

ACTIVE LEARNING WITH MULTIPLE VIEWS

by

Ion Alexandru Muslea

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

December 2002

Copyright 2002

Ion Alexandru Muslea

Dedication

To those who - lovingly and selflessly - made this possible:

Mara, Codruța, and my parents.

Acknowledgments

What soon grows old? Gratitude.

Aristotle

During the last six and a half years, my co-advisors, Craig Knoblock and Steve Minton, played a crucial role in my life. Their insights, support, and patience allowed me to grow and become the person that I am today. Craig and Steve gave me the opportunity to join a fabulous institution (ISI), to work on the topic of my dreams (machine learning) and to find my own way as a researcher. I am extremely grateful for all of this. I also want to thank the other members of my thesis committee: Kevin Knight, Margaret McLaughlin, Ray Mooney, and Paul Rosenbloom; their feedback and support is greatly appreciated.

Over the years, Kevin Knight influenced me in many ways: he always provided a fresh perspective on my work; he initiated me to the secrets of EM algorithms; and, together with Ed Hovy, he introduced me to the fascinating world of natural language processing.

Paul Rosenbloom played a key role in the development of this dissertation. His ability to find the weak points in my work helped me clean up and tighten my arguments. The most challenging part of this dissertation (i.e., the formal results) has its origins in a question that he asked me in April 2000, during my qual exam.

Ray Mooney is a great mentor who keeps amazing me with his friendliness, cheerfulness, and constant support. Ray helped me become a member of the machine learning community. His trust in me was a constant source of motivation. If only each PhD student had the chance to have “her own Ray.”

During my 26 years in school, I was fortunate to encounter teachers that motivated me, shaped my thinking, and greatly influenced my future: Maria Țigan, Alin Giurgiu, Valentin Cuibus, Ambrosie Lelijak, Mihai Hudrea, Tudor Băraian, Karol Moldovan, Ioan Oprea, Kalman Puzsai, Ioan Leția, Ioan Salomie, William Klostermeyer, Yolanda Gil, Len Adleman, and Ed Hovy. Each of them deserves more thanks that I can ever express.

I was also blessed with wonderful friends; in many ways, my successes are theirs, too. If we were to live it all over again, I cannot think of anything that they didn’t already do for me. Thank you Adrian & Brindusa Fritsch, Florin & Diana Moldovan, Florin & Robin Tămaș, Daniel Marcu, Sorin & Miruna Țicrea, Marius & Anca Greere, Florin Olteanu, Mihai & Marcela Ciupe, Dan Bochiș, Claudiu Gâlgău, Horațiu Guja, Călin Cașcaval, Gheorghe Almási, Oana Marcu, Dorel & Monica Moldovan, and Dragoș Mărgineanțu.

When I look back to *where* and *how* it all started, the answer is “Bolintineanu 18.” My parents, Uca, Ioana, Mica, Noiu and all the people revolving around them provided a unique environment that prepared me for this journey. I was unbelievably lucky to belong to that world, and all I wish is to create a similar experience for my own kids.

Finally, the two people who give sense to my life and work are Mara and Codruța. Without their love, support, and sacrifice, this dissertation would have never happened. Without them, nothing would have really mattered: there would have been no love; nothing to care for, to smile at, or to be proud of. I love you both.

Contents

Dedication	ii
Acknowledgments	iii
List Of Tables	viii
List Of Figures	ix
Abstract	xi
1 Introduction	1
1.1 A Motivating Problem: Wrapper Induction	3
1.2 My approach	6
1.2.1 Co-Testing: multi-view active learning	7
1.2.2 Co-EMT: interleaving active and semi-supervised learning	8
1.2.3 View Validation: are the views adequate for multi-view learning?	10
1.3 Thesis statement	11
1.4 Contributions	11
1.5 Thesis Overview	12
2 Preliminaries	14
2.1 Terminology and Notation	14
2.2 Minimizing the required amount of labeled data	16
2.2.1 Semi-supervised Learning	16
2.2.2 Active Learning	17
2.3 Multi-view Learning Problems	19
2.3.1 Issues in the Multi-View Setting	21
2.4 Summary	23
3 The Co-Testing Family of Active Learning Algorithms	24
3.1 Co-Testing <i>vs</i> single-view selective sampling	25
3.2 The Co-Testing family of algorithms	28
3.2.1 Learning with strong and weak views	30
3.3 Why does Co-Testing work?	32
3.3.1 The <i>High-Low</i> learning problems	33
3.3.2 Convergence properties for single-view algorithms	38

3.3.3	Convergence properties for multi-view algorithms	39
3.3.3.1	Views that are independent given the label	39
3.3.3.2	Views that are independent given the clump	40
3.3.4	Discussion	41
3.4	Co-Testing for wrapper induction	42
3.4.1	Naive Co-Testing with STALKER extraction rules	43
3.4.2	Aggressive Co-Testing with strong and weak views	45
3.5	Empirical Evaluation	48
3.5.1	Wrapper induction experiments	48
3.5.1.1	The six algorithms in the comparison	48
3.5.1.2	The experimental setup	49
3.5.1.3	The empirical results	50
3.5.2	Beyond wrapper induction	56
3.6	Discussion	59
3.7	Summary	63
4	Active + Semi-Supervised = Robust Multi-View Learning	64
4.1	The Motivating Experiment	65
4.1.1	The Semi-supervised algorithms used in the comparison	65
4.1.1.1	The Co-Training algorithm	65
4.1.1.2	The semi-supervised EM algorithm	65
4.1.1.3	The Co-EM algorithm	67
4.1.2	The empirical results	67
4.2	Co-Testing + Co-EM = Co-EMT	70
4.3	Empirical Evaluation	71
4.3.1	The Experimental Setup	71
4.3.2	Discussion	74
4.3.3	Results on real-world problems	78
4.4	Summary	79
5	View Validation	81
5.1	The View Validation Algorithm	83
5.2	Features Used for View Validation	85
5.3	Empirical Results	87
5.3.1	The multi-view test problems	87
5.3.2	Generating the WI and PTCT Datasets	88
5.3.3	The Setup	88
5.3.4	The Influence of <i>ExsPerInst</i> and <i>Size(L_k)</i>	90
5.3.5	The distribution of the errors	92
5.3.6	Understanding the Predictions	95
5.4	Summary	98

6	Related Work	99
6.1	Active learning algorithms	99
6.1.1	Active learning by query construction	100
6.1.2	Selective sampling	101
6.1.3	Co-Testing <i>vs</i> existing active learners	104
6.2	Semi-supervised concept learning	105
6.2.1	Single-view, semi-supervised classification	105
6.2.1.1	Transductive approaches	106
6.2.1.2	Expectation Maximization	106
6.2.1.3	Unlabeled data as “background knowledge”	108
6.2.2	Multi-view, semi-supervised learning	109
6.2.3	Co-EMT <i>vs.</i> existing approaches	111
6.3	Meta-learning & model selection	111
6.3.1	Experimental model selection	112
6.3.2	Knowledge-driven model selection	113
6.3.3	Transfer of learning	114
6.3.4	Meta-learning for model selection	115
6.3.4.1	Adaptive view validation for model selection	118
7	Conclusions	119
7.1	Main contributions	120
7.1.1	A multi-view approach to active learning	120
7.1.2	Robust multi-view learning	121
7.1.3	Multi-view <i>vs.</i> single-view model selection	122
7.2	Limitations	122
7.3	Future work	123
	Reference List	126
	Appendix A	
	Proofs of convergence	138
A.1	Convergence properties for single-view algorithms	138
A.2	Convergence properties for multi-view algorithms	143
A.2.1	Views that are independent given the label	143
A.2.2	Views that are “independent given the clump”	147
	Appendix B	
	The 60 Semi-Artificial Problems	158

List Of Tables

3.1	The 33 extraction tasks used in the empirical evaluation.	51
3.2	Algorithms used in empirical comparison on AD, COURSES, and TF	57
3.3	Tests of Statistical Significance	58
4.1	Error rates on two additional real world problems.	80
B.1	The 16 newsgroups included in the domain.	160

List Of Figures

1.1	An example of information agent	4
1.2	Forward and backward extraction rules	7
2.1	Pseudo-code for semi-supervised learning	17
2.2	Pseudo-code for selective sampling	18
2.3	Pseudo-code for Co-Training	20
2.4	Illustrative clumps for the COURSES domain	23
3.1	Illustrative Co-Testing: Step 1	26
3.2	Illustrative Co-Testing: Step 2	26
3.3	Illustrative Co-Testing: Step 3	26
3.4	The Co-Testing family of algorithms	29
3.5	Forward, backward, and content-based extraction rules	31
3.6	An illustrative 1-DHL learning task.	34
3.7	Illustrative 2-DHL learning tasks.	35
3.8	Version space for 1-DHL.	38
3.9	FF , FB , and BB extraction rules	44
3.10	The hierarchy of STALKER wildcards	45
3.11	Empirical results for Co-Testing on wrapper induction	53
3.12	Convergence results for the 33 wrapper induction tasks	54
3.13	Empirical results on the AD and TF problems	60
3.14	Empirical results on the COURSES problem	61

4.1	High-level description of Co-Training, Semi-supervised EM, and Co-EM	66
4.2	The results of the controlled experiment on PTCP.	69
4.3	Co-Testing vs Co-EMT	71
4.4	The lineage of the Co-EMT algorithm.	72
4.5	Illustrative learning curves for Co-EMT.	73
4.6	Empirical results on the PTCT family of problems	75
4.7	Illustrative clumps in the COURSES domain.	77
5.1	Motivation for view validation	82
5.2	The View Validation Algorithm.	85
5.3	Results on view validation	89
5.4	Varying <i>ExsPerInst</i> while <i>Size(L_k)</i> is constant	91
5.5	Varying <i>Size(L_k)</i> while keeping <i>ExsPerInt</i> constant	93
5.6	The distribution of the errors for WI and PTCT	95
5.7	The distribution of the false positives and negatives for WI and PTCT	96
5.8	Illustrative trees for WI and PTCT.	97
A.1	The two possible distributions of examples in 2-DHL.	145
A.2	Applying Co-Training to 2-DHL with independent views.	146
A.3	Ideal solution to 2-DHL with clumps.	149
A.4	Applying Co-Training to 2-DHL with clumpy views.	151
A.5	Two illustrative scenarios in which there are no contention points	153
A.6	Possible queries within a clump.	154
B.1	Generating one and two clumps per class.	159
B.2	Generating up to four clumps per class.	161

Abstract

Labeling training data for machine learning algorithms is tedious, time consuming, and error prone. Consequently, it is of utmost importance to minimize the amount of labeled data that is required to learn a target concept. In the work presented here, I focus on reducing the need for labeled data in multi-view learning tasks. The key characteristic of multi-view learning tasks is that the target concept can be independently learned within different views (i.e., disjoint sets of features that are sufficient to learn the concept of interest). For instance, robot navigation is a 2-view learning task because a robot can learn to avoid obstacles based on either sonar or vision sensors.

In my dissertation, I make three main contributions. First, I introduce Co-Testing, which is an active learning algorithm that exploits multiple views. Co-Testing is based on the idea of learning from mistakes. More precisely, it queries examples on which the views predict a different label: if two views disagree, one of them is guaranteed to make a mistake. In a variety of real-world domains, from information extraction to text classification and discourse tree parsing, Co-Testing outperforms existing active learners.

Second, I show that existing multi-view learners can perform unreliably if the views are incompatible or correlated. To cope with this problem, I introduce a robust multi-view learner, Co-EMT, which interleaves semi-supervised and active multi-view learning. My empirical results show that Co-EMT outperforms existing multi-view learners on a wide variety of learning tasks.

Third, I introduce a view validation algorithm that predicts whether or not two views are adequate for solving a new, unseen learning task. View validation uses information

acquired while solving several exemplar learning tasks to train a classifier that discriminates between tasks for which the views are adequate and inadequate for multi-view learning. My experiments on wrapper induction and text classification show that view validation requires a modest amount of training data to make high accuracy predictions.

Chapter 1

Introduction

All men by nature desire knowledge.

Aristotle

Induction plays a central role in a variety of disciplines, from artificial intelligence and philosophy to statistics and psychology. *Inductive learning* can be defined as inferring a generalization from a series of examples, each of which is typically described by a set of *features*. *Inductive learning algorithms* are presented with a set of classified examples and are asked to generate a class description for each of the concepts/classes of interest; these class descriptions can be subsequently used to classify new, unseen examples. In other words, a human labels a number of *training examples* from which the inductive algorithm creates a *classifier* that discriminates among the various concepts of interest. Such learning algorithms have been successfully used in a plethora of *fielded applications*, from making credit decisions (Michie, 1989) and classifying celestial objects (Fayyad et al., 1995) to reducing banding in rotogravure printing (Evans and Fisher, 1994) and improving the separation of gas from oil (Guilfoyle, 1986).

In practice, labeling the training examples is a tedious, time-consuming, error-prone process. Furthermore, in some application domains, the labeling of each example is also extremely expensive (e.g., it may require running costly laboratory tests). *Multi-view* learning algorithms represent a recent development in coping with this problem. Such

algorithms apply to learning problems that have *multiple views*; i.e., several disjoint subsets of features (*views*), each of which is sufficient to learn the concepts of interest. For instance, as described in (Blum and Mitchell, 1998), one can classify segments of televised broadcast based *either* on the video *or* on the audio information; or one can classify Web pages based on the words that appear *either* in the documents *or* in the hyperlinks pointing to them.

Existing research on multi-view learning (Blum and Mitchell, 1998; Collins and Singer, 1999; Pierce and Cardie, 2001) focused on *semi-supervised* learning techniques; that is, they learn a concept definition by combining a small set of labeled and a large set of unlabeled examples. By themselves, the unlabeled examples do not provide any direct information about the concepts to be learned. However, as shown in (Miller and Uyar, 1997; Nigam et al., 2000; Shahshahani and Landgrebe, 1994; Ghahramani and Jordan, 1994; Raskutti, Ferra, and Kowalczyk, 2002b), their distribution can be used to boost the accuracy of a classifier learned from the small set of labeled examples.

Intuitively, existing multi-view algorithms proceed as follows:

- first, they use the small labeled training set to learn one classifier in each view;
- then they bootstrap the views from each other by augmenting the training set with unlabeled examples on which the other views make high-confidence predictions.

Such algorithms try to make the best possible use of the “implicit” information provided by the distribution of the unlabeled examples. Their focus is to improve the accuracy of the classifiers learned only from a small set of labeled data.

In contrast to semi-supervised learning, *active learning* algorithms (Seung, Opper, and Sompolinski, 1992; Cohn, Atlas, and Ladner, 1994; Lewis and Catlett, 1994; Roy and McCallum, 2001; Tong and Koller, 2001) aim at minimizing the amount of labeled data required to learn a concept of interest. Active learners typically detect and ask the user to label only the most informative examples in the domain, thus reducing the user’s involvement in the data labeling process.

The goal of my thesis is to reduce the user’s burden by minimizing the amount of labeled training data without sacrificing the accuracy of the learned classifiers. To reach this goal, I introduce the Co-Testing algorithm (Muslea, Minton, and Knoblock, 2000b), which is a novel approach to active learning. Co-Testing is a *multi-view active learner* that maximizes the benefits of labeled training data by providing a principled way to detect the most informative examples in a domain, thus allowing the user to label only these.

After analyzing Co-Testing’s properties from both a formal and an empirical perspective, I introduce two extensions that cope with its main limitations: the inability to exploit the unlabeled examples that were not queried, and the difficulty of deciding whether or not a learning task is appropriate for multi-view learning. To address the first issue, I introduce a new algorithm, Co-EMT, which interleaves Co-Testing with Co-EM (Nigam and Ghani, 2000), a semi-supervised, multi-view algorithm. This hybrid algorithm combines the benefits of active and semi-supervised learning by both detecting the most informative examples and boosting the accuracy of the classifiers by exploiting the remaining unlabeled examples. Second, I introduce the adaptive view validation algorithm, which predicts whether or not multi-view learning is appropriate for a new, unseen learning task. The view validation algorithm is a meta-learner that makes its predictions based on experiences acquired while solving past learning tasks.

1.1 A Motivating Problem: Wrapper Induction

With the World Wide Web, computer users have gained access to a large variety of comprehensive information repositories. However, the Web is based on a browsing paradigm that makes it difficult to retrieve and integrate data from multiple sources. *Information integration systems* (e.g., Ariadne (Knoblock et al., 2001), WHIRL (Cohen, 1998), and

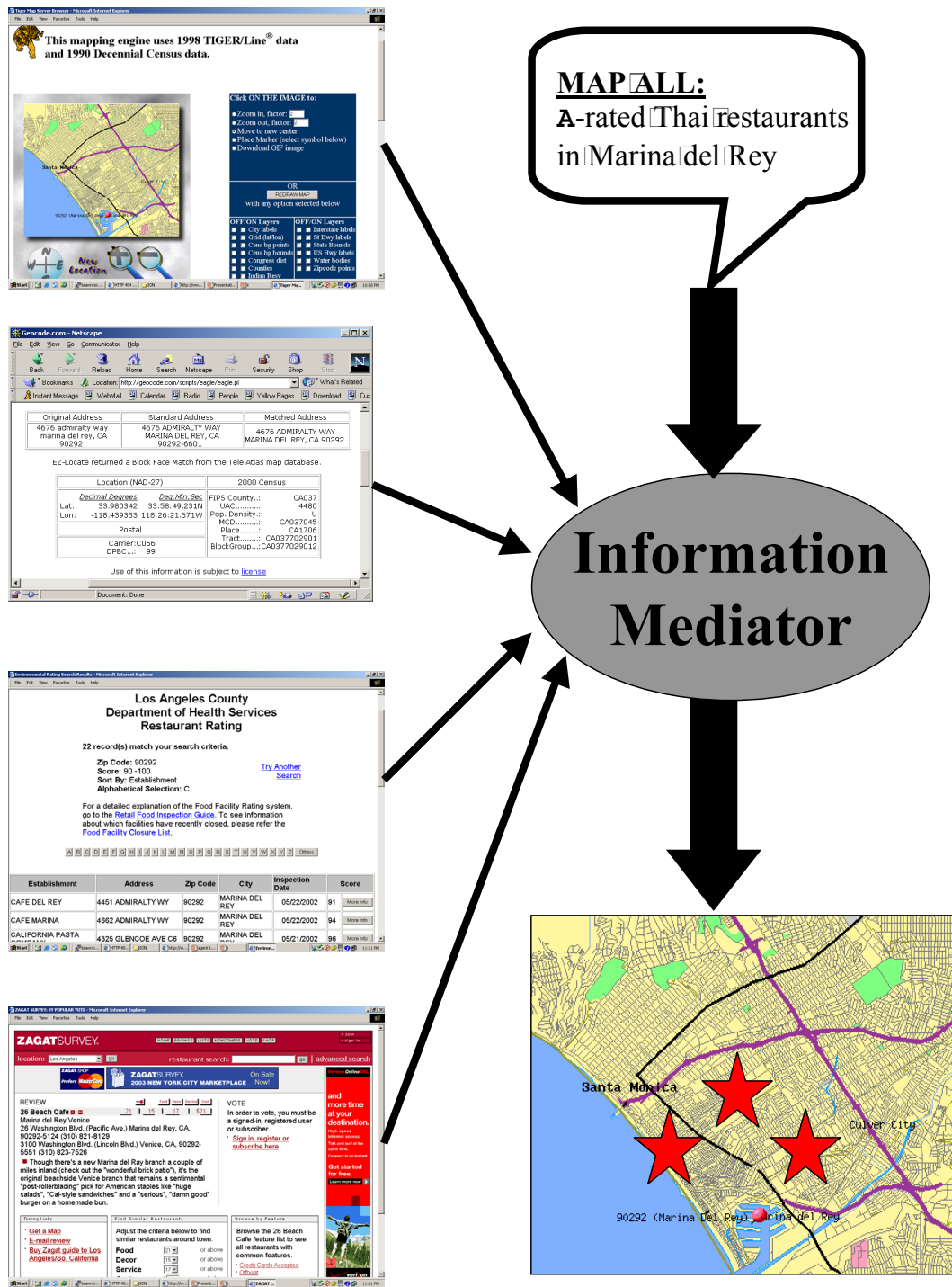


Figure 1.1: An example of information agent that plots on the map the location of restaurants of interest. The agent combines the data from four information sources: the Zagat's restaurant reviews, the L.A. County' Department of Health Web site, the ETAK geocoder, and the TIGER MAPS service.

the Information Manifold (Kirk et al., 1995)) address this problem by enabling information from pre-specified sets of Web sites to be accessed and combined via database-like queries.

Consider the illustrative agent in Figure 1.1, which takes as input queries such as

```
Show me a map with the locations of all Thai restaurants in Los
Angeles that are A-rated by the L.A. County Department of Health
Services.
```

In order to answer this query, the agent must combine data from several Web sources:

- from Zagat Web site, it can obtain the name and address of all Thai restaurants in L.A.;
- from the L.A. County Web site, it can get the health rating for all the restaurants of interest;
- from the ETAK Geocoder site, it can obtain the latitude and longitude of any physical address;
- from the TIGER MAP service, it can plot on a map the location of any geographic location, given its latitude and longitude.

Such information agents generally rely on *wrappers* to extract information from *semi-structured* Web pages (a document is semistructured if the location of the relevant information can be described based on a concise, formal grammar). Each wrapper consists of a set of extraction rules and the code required to apply those rules. As manually writing the extraction rules is a time consuming task that requires a high level of expertise, researchers designed *wrapper induction* algorithms that learn the extraction rules based on user-provided examples (Muslea, Minton, and Knoblock, 2001; Kushmerick, 2000; Hsu and Dung, 1998).

In practice, an information agent may use hundreds of extraction rules that must be updated whenever the format of the corresponding Web documents changes. As manually

labeling the training examples for each extraction rule is a tedious, error prone task, it is of utmost importance that a wrapper induction algorithm learns high accuracy rules based on a minimal number of labeled examples. Note that both the minimal training sets and the high accuracy rules are crucial to the successful deployment of an information agent. The former minimizes the amount of work required to create the information agent, thus making the task manageable. The later is required in order to ensure the quality of the agent’s answer to each query. To illustrate the practical importance of having high-accuracy extraction rules, consider the following two scenarios:

- if all the extraction rules for all the sources in Figure 1.1 are 100% accurate (i.e., they extract correctly all the items of interest), it follows that all the restaurants that satisfy the query are displayed on the map;
- in contrast, if each extraction rule is only 90% accurate, it follows that only 90% of the Thai restaurants will be extracted from the Zagat site. Similarly, only 90% of the health ratings of the extracted restaurants will be retrieved from the L.A. County site, so the agent will have only 81% of the A-rated Thai restaurants. Finally, as only 90% of the ETAK-provided latitudes and longitudes will be extracted correctly, it follows that only $81\% \times 90\% = 72.9\%$ of the restaurants of interest will be displayed.

Despite the practical importance of learning highly-accurate wrappers based on a small number of labeled examples, there has been little work on active learning for wrapper induction. Furthermore, existing general-purpose active learners cannot be applied in a straightforward manner to wrapper induction. Because of its practical importance and the lack of efficient solutions, I use wrapper induction as the main motivating problem for my dissertation. However, the algorithms introduced in this thesis are *not* specific to wrapper induction: they can be applied to all multi-view problems, including wrapper induction.

1.2 My approach

In this section I use the wrapper induction problem to illustrate the three main contributions of my dissertation. More precisely, I provide an intuitive, high-level description of how the three novel algorithms that I introduce in my thesis work. In the chapters to come, these three algorithms are applied to a variety of other multi-view learning problems such as Web page classification, advertisement removal, text classification, and discourse tree parsing.

1.2.1 Co-Testing: multi-view active learning

Co-Testing, which is the first multi-view approach to active learning, works as follows:

1. first, it uses a small set of labeled examples to learn one classifier in each view;
2. then it applies the learned classifiers to all unlabeled examples and asks the user to label one of the examples on which the views predict different labels;
3. finally, it adds the newly labeled example to the training set and repeats the whole process.

Intuitively, Co-Testing relies on the following observation: if the classifiers learned in each view do not predict the same label for an unlabeled example, at least one of the classifiers makes a mistake on that particular prediction. By asking the user to label such an example, Co-Testing is guaranteed to provide useful information for the view that made the mistake. In the real world, where noise and other effects intrude into the learning process, translating this simple intuition into an effective algorithm raises several interesting issues that are analyzed throughout this thesis.

In order to understand how Co-Testing works on wrapper induction, consider the illustrative task of extracting restaurant phone numbers from documents similar to the Web-page fragment shown in Figure 1.2. In order to extract this information, a wrapper

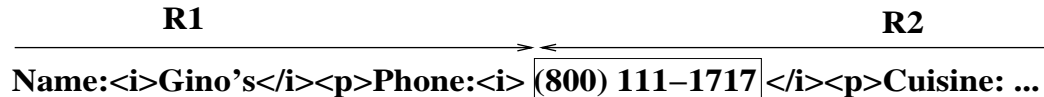


Figure 1.2: Both the *forward* rule **R1** and the *backward* rule **R2** detect the beginning of the restaurant’s phone number. Forward and backward rules have the same semantics and differ only in terms of *where* they are applied from (start/end of document) and in *which direction*.

must detect both the beginning and the end of the phone number. For instance, in order to find where the phone number begins, one can use the rule

$$\mathbf{R1} = \text{SkipTo}(\text{Phone}:\langle i \rangle)$$

This rule is applied *forward*, from the beginning of the page, and it ignores everything until it finds the string **Phone:**<i>.

Note that this is not the only way to detect where the phone number begins. An alternative way to perform this task is to use the rule

$$\mathbf{R2} = \text{BackTo}(\text{Cuisine}) \text{BackTo}((\text{Number}))$$

which is applied *backward*, from the *end* of the document. **R2** ignores everything until it finds “**Cuisine**” and then, again, skips to the first number between parentheses.

Note that **R1** and **R2** represent descriptions of the same concept (i.e., beginning of phone number) that are learned in two different views. That is, the views **V1** and **V2** consist of the sequences of characters that *precede* and *follow* the beginning of the item, respectively. The view **V1** is called the *forward view*, while **V2** is the *backward view*. Based on these two views, Co-Testing can be applied in a straightforward manner to wrapper induction. As shown in chapter 3, Co-Testing clearly outperforms existing state-of-the-art algorithms, both on wrapper induction and a variety of other real world domains.

1.2.2 Co-EMT: interleaving active and semi-supervised learning

A natural extension of Co-Testing is to further minimize the need for labeled data by also learning from the unlabeled examples that were not queried. In this thesis I introduce a novel algorithm, Co-EMT, which interleaves active and semi-supervised learning. More precisely, Co-EMT combines Co-Testing with Co-EM (Nigam and Ghani, 2000), which is a multi-view, semi-supervised algorithm. Intuitively, Co-EM represents an iterative, 2-step process: first, it uses the hypotheses learned in each view to probabilistically label all the unlabeled examples. Then Co-EM learns one new hypothesis in each view by training on the probabilistically labeled examples provided by the other view.

By interleaving active and semi-supervised learning, Co-EMT creates a powerful synergy. On one hand, Co-Testing boosts Co-EM’s performance by providing it with highly-informative labeled examples, instead of randomly-chosen ones. On the other hand, Co-EM provides Co-Testing with more accurate classifiers (in Co-EM, Co-Testing’s small set of labeled examples is complemented by a large set of unlabeled ones), thus allowing Co-Testing to learn more accurate hypotheses and thus make more informative queries.

Note that Co-EMT cannot be directly applied to wrapper induction because STALKER is not a probabilistic learning algorithm. However, I illustrate here the main idea behind Co-EMT by describing generic wrapper induction algorithm, $Co - EMT^{WI}$, that combines active and semi-supervised learning. More precisely, $Co - EMT^{WI}$ combines Co-Testing with the semi-supervised wrapper induction algorithm described below.

In order to perform semi-supervised wrapper induction, one can use a third view, which is used to evaluate the confidence of each extraction. This new, *content-based view* describes the actual item to be extracted. For example, in the phone numbers extraction task, one can use the labeled examples to learn a simple grammar that describes the field content: “(*Number*) *Number* - *Number*”. Similarly, when extracting URLs, one can learn that a typical URL starts with the string “`http://www.`”, ends with the string “`.html`”, and contains no HTML tags.

Based on forward, backward, and content-based views, one can implement the following semi-supervised wrapper induction algorithm. First, the small set of labeled examples is used to learn a hypothesis in each of the three views. Then the forward and backward views feed each other with unlabeled examples on which they make high-confidence extractions. These high-confidence extractions are strings that are extracted by either the forward or the backward rule and are also compliant with the grammar learned in the third, content-based view.

One can use Co-Testing and the semi-supervised learner above to implement $Co-EMT^{WI}$, which works as follows. First, the small set of labeled and the large set of unlabeled examples are used for semi-supervised learning. Second, the extraction rules that are learned in the previous step are used for Co-Testing. After making a query, the newly labeled example is added to the training set and the whole process is repeated for a number of iterations. The empirical study in chapter 4 shows that Co-EMT outperforms both Co-Testing and the three state-of-the-art semi-supervised learners considered in that comparison.

1.2.3 View Validation: are the views adequate for multi-view learning?

In this dissertation I introduce the problem of view validation: given a multi-view learning task, how does a user choose between solving it with a multi- or a single- view algorithm? In other words, how does one know whether or not multi-view learning will outperform pooling all features together and applying a traditional, single-view learner? Note that this question must be answered while having access to just a few labeled and many unlabeled examples: applying both the single- and multi-view active learners and comparing their relative performances is a self-defeating strategy because it doubles the amount of required labeled data (one must label the queries made by both algorithms).

The need for a view validation algorithm is motivated by the following observation: while applying Co-Testing to dozens of extraction tasks, I noticed that the forward and backward views are appropriate for most, *but not all* of these learning problem. This view

adequacy issue is tightly related to the best extraction accuracy reachable in each view. Consider, for example, an extraction task in which the forward and backward rules lead to a high- and a low- accuracy rule, respectively. Note that Co-Testing is not appropriate for solving such tasks: by definition, multi-view learning applies only to tasks in which each view is *sufficient* for learning the target concept (obviously, the low-accuracy view is *not* sufficient for accurately extracting the item of interest).

In order to cope with this problem, I introduce *adaptive view validation*, which is a novel meta-learner that uses experiences acquired while solving past learning tasks to predict whether or not the views of a new, unseen task are adequate for multi-view learning. The view validation algorithm takes as input several solved extraction tasks that are *labeled* by the user as having views that are *adequate* or *inadequate* for multi-view learning. Then it uses these solved extraction tasks to learn a classifier that, for new, unseen tasks, predicts whether or not the views are adequate for multi-view learning.

The (meta-) features used for view validation are based on the properties of the hypotheses that, for each solved task, are learned in each view: the percentage of unlabeled examples on which the rules extract the same string, the difference in the complexity of the forward and backward rules, the difference in the errors made on the training set, etc. As shown in chapter 5, for both wrapper induction and an additional text classification problem, the view validation algorithm makes high-accuracy predictions based on a modest amount of labeled data.

1.3 Thesis statement

In this dissertation, I introduce and evaluate the first approach to multi-view active learning. The thesis of my dissertation is the following:

Multi-view active learning maximizes the accuracy of the learned hypotheses while minimizing the amount of labeled training data.

1.4 Contributions

In this thesis, I make three main contributions:

1. I introduce a novel approach to active learning. I describe Co-Testing, a family of multi-view active learning algorithms that
 - can be applied to any multi-view learning problem, regardless of the algorithm used to learn the classifiers in each view;
 - outperforms existing active learning algorithms on a variety of real world domains, including wrapper induction, Web page classification, advertisement removal, and discourse tree parsing.

I analyze the practical problems that arise when the multi-view assumptions are violated, and I introduce extensions that cope with these problems.

2. I introduce a novel multi-view learning algorithm, Co-EMT, which obtains a robust behavior over a wide spectrum of problems by interleaving active and semi-supervised multi-view learning.
3. I formalize the view validation problem, and I introduce an adaptive view validation algorithm, which uses meta-learning to predict whether or not multi-view learning is appropriate for a new, unseen learning task.

1.5 Thesis Overview

This dissertation is organized as follows.

- Chapter 2 presents the terminology and notation used in the rest of the dissertation. It also discusses the intuition behind multi-view learning and the underlying assumptions.

- Chapter 3 introduces Co-Testing, a family of multi-view active learning algorithms. After formalizing the Co-Testing approach to active learning, I present a series of experiments that compare Co-Testing with existing (single-view) active learning algorithms.
- Chapter 4 first describes a simple experiment that emphasizes the drawbacks of existing (semi-supervised) multi-view algorithms. Then it presents a novel multi-view algorithm, Co-EMT, which interleaves active and semi-supervised learning.
- Chapter 5 introduces view validation, in which the goal is to predict whether or not multi-view learning is appropriate for a new, unseen learning task.
- Chapter 6 compares and contrasts my work with related approaches.
- Chapter 7 concludes by summarizing my contributions and proposing directions for future work.
- Appendix A provides the proofs of the formal properties presented in chapter 3.
- Appendix B describes a (semi-artificial) family of multi-view text classification tasks in which I control the level to which the multi-view assumptions are violated. These problems are used to evaluate the robustness of the various multi-view algorithms.

Chapter 2

Preliminaries

That which is static and repetitive is boring.

That which is dynamic and random is confusing.

In between lies art.

John Locke

The goal of this dissertation is to minimize the amount of labeled data required to learn a concept of interest, thus reducing user's burden in creating systems that use machine learning techniques. My work is situated at the confluence of three areas of research: active learning, semi-supervised learning, and multi-view learning. This chapter introduces the main ideas in these three fields and familiarizes the reader with the notation and terminology used throughout the dissertation.

2.1 Terminology and Notation

As noted in (Mitchell, 1998), machine learning involves acquiring general concepts based on specific training examples. In *concept learning*, an algorithm is used to automatically infer the definition of a concept starting from examples that are members or non-members of the concept of interest.¹ For example, consider the problem of discriminating between

¹For sake of simplicity, this chapter discusses only the concept learning scenario, in which the learner discriminates between two classes of interest. The notation and terminology generalize in a straightforward manner for more than two classes.

students and faculty in a Computer Science department. Based on the descriptions of several students and faculty, an inductive algorithm may learn that - say - only faculty make more than \$50K a year.

In this dissertation I use the following terminology. For any learning problem, the set of all possible examples is called the *instance space* and is denoted by X . Any $x \in X$ represents a particular *example* or *instance*. An example is represented as a *feature vector* that stores the values of the various *attributes* or *features*. For instance, in the student/faculty problem above, the feature vector may contain attributes such as name, salary, social security number, age, and number of publications.

The concept to be learned is called the *target concept*, and it can be seen as a boolean function $c : X \rightarrow \{0,1\}$ that classifies any instance x as a member or a non-member of the concept of interest. For example, in the student/faculty problem, $c(x_1) = 0$ and $c(x_2) = 1$ classify the examples x_1 and x_2 as student or faculty, respectively.

In order to learn the target concept, the user typically provides a set of *training examples*, each of which consists of an instance $x \in X$ and its label, $c(x)$. The notation $\langle x, c(x) \rangle$ denotes such a training example. Instances for which $c(x) = 1$ are called *positive examples*, while the other ones ($c(x) = 0$) are called *negative examples*. The symbol L is used to denote the set of labeled training examples (also known as the *training set*). By definition, an algorithm that uses for training only the labeled examples in L is called a *supervised* learner. Similarly, a learning algorithm that trains on both labeled and unlabeled examples is called *semi-supervised*, while an *unsupervised* learner is trained solely on unlabeled examples (i.e., it clusters unlabeled examples based on their similarities with each other).

Given a training set L for the target concept c , an inductive learning algorithm \mathcal{L} searches for a function $h : X \rightarrow \{0,1\}$ such that $\forall x \in X, h(x) = c(x)$. The learner \mathcal{L} searches for h within the set H of all possible hypotheses, which is (typically) determined by the person who designs the learning algorithm. A hypothesis h is *consistent* with the

training set L if and only if $\forall \langle x, c(x) \rangle \in L, h(x) = c(x)$. Finally, the *version space* $VS_{H,L}$ represents the subset of hypotheses in H that are consistent with the training set L .

2.2 Minimizing the required amount of labeled data

Labeling the examples in the training set L is a tedious, time consuming, and error prone process. Consequently, it is important to learn a target concept based on as few as possible labeled examples. In this dissertation, I am primarily interested in two types of techniques that reduce the need for labeled training data: semi-supervised learning and active learning. The former boosts the accuracy of a supervised learner based on an additional set of unlabeled examples, while the latter minimizes the amount of labeled data by asking the user to label only the most informative examples in the domain.

Semi-supervised and active learning techniques share two important characteristics. First, besides the set L of labeled examples, they use an additional *working set* of unlabeled examples U . The examples in U can be seen as pairs $\langle x, ? \rangle$, where “?” signifies that the example’s label is unknown. Second, both semi-supervised and active learning techniques try to maximize the accuracy of the supervised *base learner* \mathcal{L} that is used to learn the target concept.

2.2.1 Semi-supervised Learning

In *semi-supervised learning* one tries to improve the accuracy of a supervised learner by exploiting the availability of a (large) set of unlabeled examples U . A typical semi-supervised algorithm proceeds as follows: first, it uses the base learner \mathcal{L} and the (small) set of labeled examples L to learn an initial hypothesis h . Then h is applied to the unlabeled examples in U , and some or all of these examples, together with the label predicted by h , are added to L . Finally, the entire process is repeated for a number of iterations.

- Given:**
- the base learner \mathcal{L}
 - the sets L and U of labeled and unlabeled examples
 - the number k of iterations to be performed
 - the number n of examples to be added to L after each iteration

Semi-supervised Learning:

- LOOP for k iterations
- let h be the classifier obtained by training \mathcal{L} on L
 - let MCP be the n examples in U on which h makes the most confident predictions
 - FOR EACH $x \in MCP$ DO
 - remove x from U
 - add $\langle x, h(x) \rangle$ to L
-

Figure 2.1: Semi-supervised Learning: learn a hypothesis h based on the labeled examples, apply h to all unlabeled examples, and add its most confident predictions to the training set. Then repeat the whole process for a number of iterations.

The intuition behind semi-supervised learning is straightforward: even though the initial hypothesis h is learned based on a small training set, its highest confidence predictions are likely to be correct. Consequently, by adding the “high confidence” examples in U to the training set L , one obtains a larger training set, based on which one can learn a more accurate hypothesis. In turn, the more accurate hypothesis can be used to label more examples, and so on.

Figure 2.1 shows the pseudo-code for a typical semi-supervised learning algorithm. This general framework covers a wide variety of algorithms, from *self training* (Nigam and Ghani, 2000) to *semi-supervised EM* (McCallum and Nigam, 1998b; Nigam et al., 2000). The former adds to L just a few examples per iteration (i.e., $n \ll \text{Size}(U)$), while the latter adds to L the entire set U , which is re-labeled by h after each iteration.

2.2.2 Active Learning

By definition, a *passive learning* algorithm takes as input a randomly chosen training set L . In contrast, an *active learning* algorithm has the ability to choose the examples in L . In other words, active learning algorithms try to detect the most informative examples in

- Given:-** a learning algorithm \mathcal{L}
- the sets L and U of labeled and unlabeled examples
 - the number N of queries to be made
 - a utility function $Utility : X \times H \mapsto \mathfrak{R}$

Selective Sampling:

- LOOP for N iterations
- let h be the classifier obtained by training \mathcal{L} on L
 - let $q = \operatorname{argmax}_{u \in U} Utility(u, h)$
 - remove q from U and ask the user for its label, $c(q)$
 - add $\langle q, c(q) \rangle$ to L

Figure 2.2: Selective Sampling: learn a hypothesis h based on the labeled examples L , and query the unlabeled example u that maximizes the utility function $Utility(u, h)$.

the instance space X and ask the user to label only them. The examples that are chosen for labeling are called *queries*.

There are two main approaches to active learning: query construction and selective sampling. The former queries an example that is constructed by setting the value of each attribute so that the resulting query is as informative as possible. In contrast, *selective sampling* is a form of active learning in which the queries must be chosen from a given *working set* of unlabeled examples U . Selective sampling is the most popular type of active learning because:

- in many real-world problems, it is easy to obtain a large number of unlabeled examples;
- even though one may be able to artificially construct a query (i.e., an unlabeled example) that is more informative than the examples in U , there is a high risk that the user may not be able to label such an example because it does not correspond to a real-world entity (see (Lang and Baum, 1992) for details).

As my work focuses exclusively on selective sampling techniques, in this dissertation the terms *active learning* and *selective sampling* are used interchangeably.

As shown in Figure 2.2, selective sampling starts with a (small) set of labeled examples L and a (large) set of unlabeled examples U . It uses the base learner \mathcal{L} to induce a

hypothesis h and then queries the unlabeled example $q \in U$ that maximizes some *utility function*. After adding $\langle q, c(q) \rangle$ to L , the process is repeated for a number of iterations.

Depending on the way in which they generate the hypothesis h , there are two types of active learning algorithms: *single-hypothesis* and *committee-based*. The former applies to base learners \mathcal{L} that can reliably estimate the confidence of their prediction. Single-hypothesis active learners typically use one of the following two utility functions:

- *uncertainty reduction*: the user is asked to label the unlabeled example on which h makes the least confident prediction.
- *expected-error minimization*: the system queries the example that maximizes the *expected* reduction in classification error.

In contrast, the utility function in committee-based active learning measures the percentage of version space that is removed after each query. In the committee-based approach, one generates M hypotheses (i.e., the *committee*) and queries the example $u \in U$ on which the prediction of the committee is the most split. Intuitively, each such query removes half of the version space, thus converging to the target concept c in $\log(\text{size}(VS_{H,L}))$ queries.

2.3 Multi-view Learning Problems

The previous sections presented the traditional, *single-view* machine learning scenario, in which learners have access to the entire set of features in the domain. By contrast, in the *multi-view* setting one can partition the domain's features in subsets (*views*) that are *sufficient* for learning the target concept. For instance, one can perform speech recognition by using either the lip motions or the emitted sounds; or one can determine a robot's position based on either vision or sonar sensors.

In a multi-view learning problem, an example x is described by a different set of features in each view. For example, in a domain with two views **V1** and **V2**, a labeled example is a triple $\langle x_1, x_2, l \rangle$, where l is its label, and x_1 and x_2 are its descriptions

- Given:-** a learning problem with two views $\mathbf{V1}$ and $\mathbf{V2}$
- the base algorithm \mathcal{L}
 - the sets L and U of labeled and unlabeled examples
 - the number k of iterations to be performed
 - the number n of examples to be added to L after each iteration

Co-Training:

- LOOP for k iterations
- use \mathcal{L} , $\mathbf{V1}(L)$, and $\mathbf{V2}(L)$ to create classifiers h_1 and h_2
 - let PL_1 and PL_2 be the n unlabeled examples on which h_1 and h_2 make the most confident predictions, respectively
 - FOR EACH $\langle x_1, x_2, ? \rangle \in PL_1 \cup PL_2$ DO
 - remove $\langle x_1, x_2, ? \rangle$ from U
 - IF $\langle x_1, x_2, ? \rangle \in PL_1$ THEN add $\langle x_1, x_2, h_1(x) \rangle$ to L
 - ELSE add $\langle x_1, x_2, h_2(x) \rangle$ to L

Figure 2.3: Co-Training (repeatedly) learns a hypothesis in each view and adds to the training set the most confident predictions made on the unlabeled examples.

in the two views. Similarly, an unlabeled example is denoted by $\langle x_1, x_2, ? \rangle$. For any example x , $\mathbf{V1}(x)$ denotes the descriptions x_1 of x in $\mathbf{V1}$. Similarly, $\mathbf{V1}(L)$ consists of the descriptions in $\mathbf{V1}$ of the examples in L .

Previous approaches to multi-view learning consist of semi-supervised algorithms that bootstrap the views from each other in order to boost the accuracy of a classifier learned based on a few labeled examples. Multi-view algorithms have been successfully applied to a variety of real-world domains, from natural language processing (Collins and Singer, 1999; Pierce and Cardie, 2001; Sarkar, 2001) and speech recognition (de Sa and Ballard, 1998) to Web page classification (Blum and Mitchell, 1998).

To illustrate the intuition behind multi-view learning, let us briefly consider the basic idea in the Co-Training algorithm (Blum and Mitchell, 1998). Co-Training uses a small set of labeled examples to learn an initial classifier in the two views (see Figure 2.3). Then each classifier is applied to all unlabeled examples, and Co-Training detects the examples on which each classifier makes the most confident predictions. These high-confidence

examples are labeled with the estimated class labels and added to the training set. Based on the updated training set, a new classifier is learned in each view, and the process is repeated for several iterations.

Note that Co-Training is similar to the semi-supervised learner described in Figure 2.1. The main difference between the two algorithms is that in Co-Training the views are bootstrapped from each other, while in the single-view case the algorithm is “self-training” (Nigam and Ghani, 2000). In practice, multi-view algorithms were shown to outperform their single-view counterparts (Nigam and Ghani, 2000; Ghani, 2001; Collins and Singer, 1999; Muslea, Minton, and Knoblock, 2000b; Muslea, Minton, and Knoblock, 2002a). The next section briefly presents the theoretical justification for using multi-view learning instead of pooling all features together and using a single-view learning algorithm.

2.3.1 Issues in the Multi-View Setting

Blum and Mitchell (1998) proved that for a problem with two views the target concept can be learned based on a few labeled and many unlabeled examples, provided that the views are *compatible* and *uncorrelated*. The former condition requires that all examples are labeled identically by the target concepts in each view. The latter means that for any example $\langle x_1, x_2, l \rangle$, x_1 and x_2 are independent given l .

The proof in (Blum and Mitchell, 1998) is based on the following argument: one can learn a weak hypothesis h_1 in **V1** based on the few labeled examples and then apply h_1 to all unlabeled examples. If the views are uncorrelated, these newly labeled examples are seen in **V2** as a random training set with classification noise, based on which one can learn the target concept in **V2**. Both the requirements that the views are compatible and uncorrelated are crucial in this process.² If the views are correlated, the training set in **V2** is *not* random. If the views are incompatible, the target concepts in the two views

²An updated version of (Blum and Mitchell, 1998) shows that the theoretical guarantees also hold for partially incompatible views, provided that they are uncorrelated. However, in practice one cannot ignore view incompatibility because one rarely, if ever, encounters real world problems with uncorrelated views.

label *differently* a large number of examples; consequently, h_1 may “mislabel” so many examples that learning the target concept in **V2** becomes impossible.

To introduce the intuition behind view incompatibility and correlation, let us consider the COURSES problem (Blum and Mitchell, 1998), in which Web pages are classified as “*course homepages*” and “*other pages*.” The views **V1** and **V2** consist of words in the hyperlinks pointing to the pages and words in the Web pages, respectively. Figure 2.4 shows several illustrative examples from the domain. Each of the **17 lines** in Figure 2.4 represents an example; that is, each example x is depicted as a **line** that connects its descriptions x_1 and x_2 in the two views. All but the two bottom examples (i.e., **lines**) are “*course homepages*”; consequently, to keep Figure 2.4 simple, I do not show the examples’ labels. Note that in Figure 2.4 the *same page* may be referred by several hyperlinks, while several hyperlinks that contain the *same text* may point to different pages.

In real world problems, the views are partially incompatible for a variety of reasons: corrupted features, insufficient attributes, etc. For instance, as shown in Figure 2.4, of the three hyperlinks that contain the text “Neural Nets”, two point to homepages of neural nets classes, while the third one points to a publications page. That is, Web pages with different labels in **V2** have the *same* description in **V1**. Consequently, [“Neural Nets”, “MIT’s CS 211: ...”] and [“Neural Nets”, “J. Doe’s Papers ...”] are incompatible because they require that “Neural Nets” has simultaneously two different labels.

In practice, the views are also (partially) correlated because of *domain clumpiness*, which can be best introduced by an example. Consider, for instance, the eight multi-view examples of AI homepages that are *depicted as lines* within the “AI CLUMP” rectangle in Figure 2.4. Such a group of examples is called a *clump* because the bi-partite subgraph that has as vertices the four hyperlinks and four Web pages, respectively, is heavily connected by the eight edges representing the examples. Note that two clumps per class are sufficient to violate the “uncorrelated views” assumption: for any example x , it is highly likely that its descriptions in the two views come from the same clump. Intuitively,

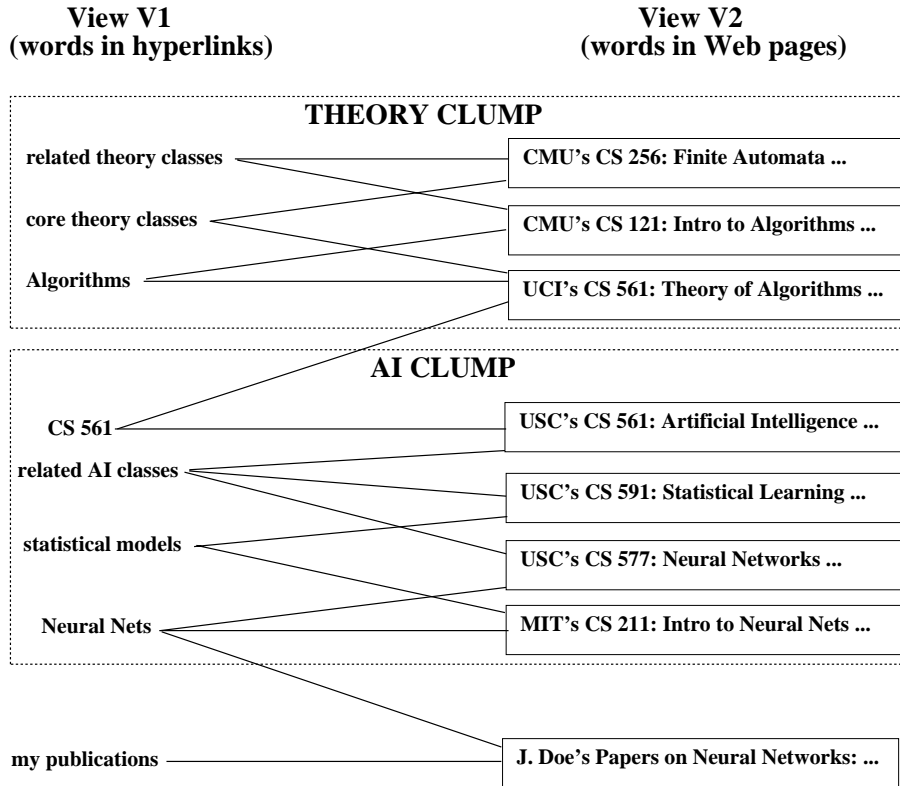


Figure 2.4: Two illustrative “positive clumps” (i.e., *course homepages*) in the COURSES domain: the AI and Theory clumps consist of course homepages for AI and Theory classes, respectively. Note that knowing an example’s description in one view provides more information (i.e., the actual clump) than knowing the example’s label (i.e., only the example’s class, which consists of several clumps).

this means that it is unlikely to have examples such as [“CS 561”, “UCI’s CS 561: Theory of Algorithms”], which connects the THEORY and AI clumps (see Figure 2.4).

2.4 Summary

In this section I introduced the basic terminology and notation that is used throughout this dissertation. First, I discussed the main concepts and ideas in semi-supervised and active learning. Then I introduced multi-view learning, and I discussed the two main assumptions in multi-view learning: view independence and compatibility.

Chapter 3

The Co-Testing Family of Active Learning Algorithms

True genius lies in the capacity for evaluation of [...] conflicting information.”

Winston Churchill

In order to learn a classifier, supervised learning algorithms need *labeled* training examples. In many applications, labeling the training data is an expensive process because it requires human expertise and is a tedious, error prone, time consuming task. *Selective sampling* reduces the number of training examples that need to be labeled by examining a pool of unlabeled examples and selecting only the most informative ones for the human to label. This chapter introduces *Co-Testing*, which is the first approach to *multi-view* active learning.

Selective sampling techniques work by asking the user to label examples that maximize the information conveyed to the learner. In a standard, single-view learning scenario, this generally translates into finding an example that - at best - splits the version space in half; i.e., eliminating half of the hypotheses consistent with the current training set.

In multi-view domains, one can do much better. Co-Testing trains one classifier for each view, applies them to the pool of unlabeled examples, and selects a query based on the degree of disagreement among the learners. Because the target concepts in each view must agree, Co-Testing can reduce the hypothesis space faster than would otherwise be possible. To illustrate this, consider a learning problem with two views, **V1** and **V2**, and

imagine the following extreme scenario: there is an unlabeled example that is classified as positive by a single hypothesis from the **V1** version space; furthermore, assume that the same example is classified as positive by all but one of the hypotheses from the **V2** version space. If the system queries this particular example, the version space in either **V1** (if the example is positive) or **V2** (in case the example is negative) collapses to a single hypothesis, thus ending the learning process.

3.1 Co-Testing *vs* single-view selective sampling

Let us consider the illustrative task of classifying the employees of a CS department in two categories: faculty and non-faculty. Let us assume that the classification can be done either by using a person’s salary (e.g., only faculty have salaries above \$65K) or office number (e.g., only faculty office numbers are below 300). In this case, the domain has two views: one that uses only the salary, and another one that uses only the office number.

In both views the target concept is a threshold value: \$65K for salary, and 300 for the office number. To learn the target concept in each view, one can use the following base learner \mathcal{L} : first, \mathcal{L} identifies the pair of labeled examples that belong to different classes and have the closest attribute values. Then \mathcal{L} sets the threshold to the mean of these two values.

Co-Testing works as follows: initially, the user provides a few labeled examples, and a pool of unlabeled ones. In Figure 3.1, the unlabeled examples are denoted by points, while the labeled ones appear as \oplus and \ominus (the former denotes faculty, and the latter represents non-faculty). Co-Testing uses the base learner \mathcal{L} to create one classifier for each view (the classifiers are geometrically represented as the dotted and the dashed lines, respectively).

Then Co-Testing applies both classifiers to *all* unlabeled examples and determines the *contention points* – the examples that are labeled differently by the two classifiers. The contention points, which lay in the picture’s gray areas, are extremely informative

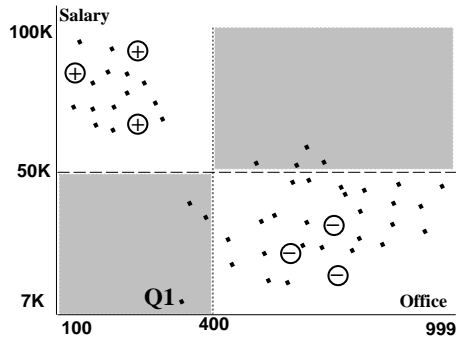


Figure 3.1: **STEP 1:** From the initial training set, Co-Testing learns the hypotheses (i.e., threshold values) **50K** and **400** in the two views. Among the contention points, which lay in the dark grey areas, Co-Testing queries the example **Q1**, on which both hypotheses are the most confident (i.e., **Q1** is the farthest away from both threshold values).

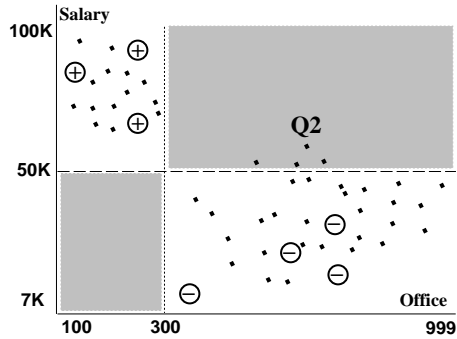


Figure 3.2: **STEP 2:** After adding **Q1** to the training set, Co-Testing learns two new hypotheses and queries the example **Q2**, which - again - is the farthest away from both threshold values.

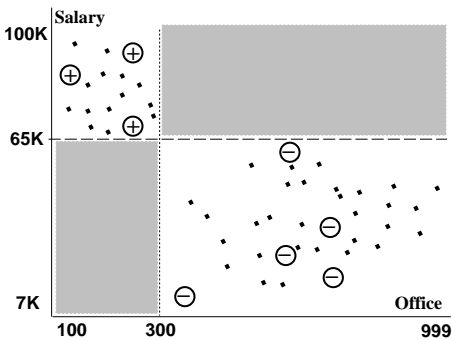


Figure 3.3: **STEP 3:** After adding **Q2** to the training set and learning two new hypotheses, Co-Testing stops because there are no more contention points.

because whenever the two classifiers disagree, at least one of them must be wrong. Finally, Co-Testing selects for labeling one of the contention points, adds it to the training set, and repeats the whole process.

If the base learner \mathcal{L} can evaluate the confidence of its classification, Co-Testing can query the contention point on which *both* categorizers are *most confident*, which means that each query *maximally* improves at least one of the hypotheses; that is, each query can remove more than half of the version space. In the example above, the confidence level of the predictions made by each view is proportional to the distance between the point and the corresponding threshold value: the larger the distance, the higher the confidence in the classification.

In Figure 3.1, Co-Testing asks for the label of the example **Q1**, which is the contention point on which both categorizers are the most confident (i.e., the *smallest* of the distances to the two thresholds is *maximal*). Once the example is labeled by the user, Co-Testing learns a new hypothesis in each view, finds the new set of contention points (see Figure 3.2), makes another query **Q2**, and re-trains one more time. As shown in Figure 3.3, the new classifiers agree on all unlabeled examples, and Co-Testing stops.

As already mentioned, in single-view active learning, the best one can hope for is to remove half of version space with each query. To illustrate this idea, consider the *Uncertainty Sampling* algorithm (Lewis and Gale, 1994), which first uses the labeled data to learn a classifier and then queries one of the points on which the classifier is the *least confident*. When using just one of the views in the example above, the lowest confidence points are the ones that are *the closest* to the learned threshold value. As each query made by *Uncertainty Sampling* removes approximately half of version space, it follows that the single-view algorithm requires more queries than Co-Testing to learn the target concept.

In comparison to single-view active learners, Co-Testing has two major advantages. First of all, combining evidence from several views allows it to make queries that lead to maximal improvements in one of the views. Second, by querying only contention points,

Co-Testing is guaranteed to *always* ask for the label of an example on which at least one of the classifiers is wrong. Detecting the mistakes of the current hypotheses is of utmost importance for base learners such as decision trees, which improve only from mistakes.

3.2 The Co-Testing family of algorithms

Figure 3.4 provides a formal description on the Co-Testing family of algorithms. Given a base learner \mathcal{L} , a set L of labeled examples, and a set U of unlabeled ones, Co-Testing algorithm works as follows: first, it learns the classifiers h_1 and h_2 by applying the algorithm \mathcal{L} to the projection of the examples in L onto the two views, $\mathbf{V1}$ and $\mathbf{V2}$. Then it applies h_1 and h_2 to all unlabeled examples in U and creates the set of contention points, which consists of all unlabeled examples for which h_1 and h_2 predict a different label. Finally, it queries one of the contention points and repeats the whole process. After making the allowed number of queries, Co-Testing uses the hypotheses learned in each view to create a final *output hypothesis*.

The various members of the Co-Testing family differ from each other with two respects: the strategy used to select the next query, and the manner in which the output hypothesis is constructed. In other words, each Co-Testing algorithm is uniquely defined by the choice of the functions **SelectQuery()** and **CreateOutputHypothesis()**. These two functions depend on the properties of both the application domain and the base learner \mathcal{L} .

In this dissertation I consider three types of query selection strategies:

- *naive*: choose at random one of the contention points. This straightforward strategy is appropriate for base learners that lack the capability of reliably estimating the confidence of their predictions.
- *aggressive*: choose the contention point on which both h_1 and h_2 make the most confident prediction. *Aggressive Co-Testing* is designed for high accuracy domains, in which there is little or no noise. On such problems, discovering unlabeled examples

- Given:** - a base learner \mathcal{L}
- a learning problem with features $\mathbf{V}=\{a_1, a_2, \dots, a_N\}$
 - two views $\mathbf{V1}$ and $\mathbf{V2}$, where $\mathbf{V}=\mathbf{V1}\cup\mathbf{V2}$ and $\mathbf{V1}\cap\mathbf{V2}=\emptyset$
 - the sets L and U of labeled and unlabeled examples, respectively
 - number N of queries to be made

LOOP for N iterations

- use \mathcal{L} , $\mathbf{V1}(L)$, and $\mathbf{V2}(L)$ to create classifiers h_1 and h_2
 - let $ContentionPoints = \{ \langle x_1, x_2, ? \rangle \in U \mid h_1(x_1) \neq h_2(x_2) \}$
 - let $\langle x_1, x_2, ? \rangle = \mathbf{SelectQuery}(ContentionPoints)$
 - remove $\langle x_1, x_2, ? \rangle$ from U and ask for its label l
 - add $\langle x_1, x_2, l \rangle$ to L
- **CreateOutputHypothesis**(h_1, h_2)

Figure 3.4: The Co-Testing Family of Algorithms: repeatedly learn a classifier in each view and query one of the contention points (i.e., an unlabeled example for which the two views make a different prediction).

that are misclassified “with high confidence” translates into queries that remove significantly more than half version space.

- *conservative*: choose the contention point on which the confidence of the predictions made by h_1 and h_2 is as close as possible (i.e., with equal confidence, they predict a different label). *Conservative Co-Testing* is appropriate for noisy domains, where the *aggressive* strategy may end up querying mostly noisy examples.

Creating the output hypothesis also allows the user to choose from a variety of alternatives, such as:

- *winner-takes-all*: the output hypothesis is the one learned in the view that makes the smallest number of mistakes on the N queries. This is the most obvious solution for 2-view learning problems in which the base learner cannot estimate the confidence of its predictions.
- *majority vote*: Co-Testing predicts the label that was predicted by most of the hypotheses learned in the various views. This strategy is appropriate when more than

two views are available, and the base learner cannot estimate the confidence of its predictions.

- *weighted vote*: combines the vote of each hypothesis, weighted by the confidence of their respective predictions.

3.2.1 Learning with strong and weak views

In the multi-view setting one assumes that each view is sufficient to learn the target concept. However, in practice, there are also views in which one can only learn a concept that is strictly *more general* or *more specific* than the concept of interest. I use the terms *strong* and *weak* views to discriminate between the views that describe the actual target concept and the ones in which one learns a more general/specific concept.

For example, consider again the wrapper induction task described in the introduction. As the phone number can be extracted by both the forward rule **R1** and the backward rule **R2** (see Figure 3.5), it follows that the forward and backward views are *strong views*. In contrast, the content-based view is a *weak view*: the grammar “(*Number*) *Number* - *Number*” describes a concept more general than the one of interest because it cannot discriminate between the various types of numbers that may appear in the document (e.g., home, cellular, and fax numbers).

Despite its limitations, the content-based view can be extremely informative for Co-Testing. For example, one can use the grammar “(*Number*) *Number* - *Number*” to detect the most informative contention points; that is, the unlabeled examples on which the two strong rules extract different strings that are both *inconsistent* with the content-based grammar. In this scenario, each query represents a mistake not only in one, but in both strong views, thus leading to faster convergence.

More formally, the weak views can be exploited by the query selection strategy. First, in the context of learning with strong and weak views, let me redefine the contention points as the unlabeled examples on which the *strong views* predict a different label. Then one can choose to query the contention points on which the weak view most confidently

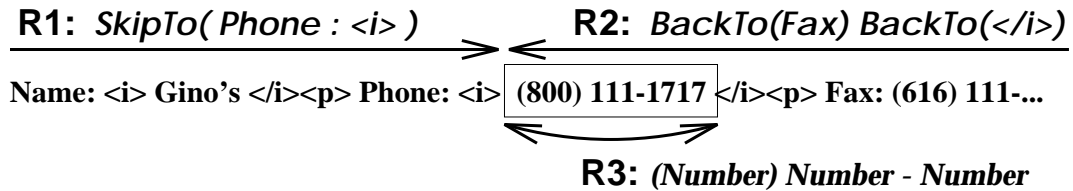


Figure 3.5: Both the **forward rule R1** and the **backward rule R2** detect the beginning of the restaurant's phone number. The rule **R3** consists of a grammar that describes the content of the item to be extracted.

contradicts *both* strong views. Such queries are highly likely to represent a mistake in both strong views, which implies large cuts in both strong version spaces. In section 3.4.2, I describe an Aggressive Co-Testing algorithm for wrapper induction that uses both strong and weak views.

The idea of combining strong and weak views clearly appears in applications other than wrapper induction, even though it was not formalized as such and it was not used for active learning. For example, in (Kushmerick, Johnston, and McGuinness, 2001), the authors discuss the problem of classifying the various lines of text on a business card as a person's name, affiliation, address, phone number, fax, etc. In this application, the strong view consists of the words that appear on each line, based on which a Naive Bayes text classifier is learned. In the weak view, Kushmerick *et al.* exploit the relative order of the lines on the card: they learn a hidden Markov Model that predicts the probability of a particular ordering of the lines on the business card (e.g., name followed by address, followed by phone number).

This weak view defines a class of concepts that is *more general* than the target concept: all line orderings are possible, even though they are not equally probable. By itself, the order of the text lines cannot be used to accurately classify the lines. However, when combined with the strong view, the ordering information leads to a classifier that clearly outperforms the stand-alone strong view (Kushmerick, Johnston, and McGuinness, 2001).

The idea of combining strong and weak views also appears in the DISCOTEX system (Nahm and Mooney, 2000), which combines an information extraction system (the *strong view*) with a text mining one (the *weak view*). In this case, the task consists of

extracting the relevant information from computer science job postings to the newsgroup `austin.jobs`; that is, DISCOTEX must extract the job title, salary, location, programming languages, development platforms, required degree, etc.

Nahm and Mooney (2000) propose the following approach to this problem: first, they use the RAPIER algorithm (Califf and Mooney, 1999) to learn extraction rules for each item of interest. Second, they apply the learned rules to a large, unlabeled corpus of job postings and create a database that is populated with the extracted information. Third, by applying a text mining algorithm to this database, they learn rules that predict the value of a particular item based on the values extracted from other fields. For example, the text mining algorithm may discover the following pattern: “IF the job requirements include knowledge of C++ and CORBA THEN the development platforms include Windows”. Finally, when the information extraction system is deployed and the RAPIER rules fail to extract an item of interest, the rules mined from the text are used to predict the content of that particular item.

In this scenario, the RAPIER rules represent the *strong view* because these rules are sufficient for extracting the data of interest. In contrast, the mined rules represent the *weak view* because they cannot be learned or used by themselves. Furthermore, as DISCOTEX discards all but the most accurate of the mined rules, which are highly-specific, it follows that the weak view can be used to learn only concepts that are *more specific* than the target concept. However, Nahm and Mooney (2000) show that these overly specific mined rules improve the extraction accuracy by capturing information that complements the RAPIER extraction rules.

3.3 Why does Co-Testing work?

In order to better understand *why* Co-Testing works and *how* it compares with other algorithms, I analyze its behavior on the *High-Low* learning problem, which was used in the past to study the convergence of the Query-By-Committee active learner (Hasenjager

and Ritter, 1996; Freund et al., 1997; Hasenjager, 2000). More precisely, I use the *High-Low* problem to compare and contrast the convergence properties of three types of learning strategies:

- active learning: Co-Testing (Muslea, Minton, and Knoblock, 2000b), Uncertainty Sampling (Lewis and Gale, 1994), Query-by-Committee (Seung, Opper, and Sompolinski, 1992);
- semi-supervised learning: Co-Training (Blum and Mitchell, 1998);
- random sampling (i.e., randomly choose the examples to be labeled).

In my analysis, I focus on two variants of the *High-Low* problem: 2-DHL and 1-DHL. The former can be seen as the formalization of the multi-view *faculty - non-faculty* learning problem discussed in section 3.1; the latter represents the single-view version of 2-DHL (e.g., discriminate between *faculty* and *non-faculty* based solely on a person’s salary). I use 1-DHL and 2-DHL to illustrate the convergence properties of the single-view and multi-view learning algorithms, respectively.

3.3.1 The *High-Low* learning problems

In the *1-view, discretized High-Low* problem (1-DHL), the goal is to learn a target function of the form

$$f : [1, H] \mapsto \{0, 1\}, \quad f(X) = \begin{cases} l_1 & \text{if } X \leq w_{TC} \\ l_2 & \text{if } X > w_{TC} \end{cases}$$

where $w_{TC} \in \{1, 2, 3, \dots, H\}$, $l_1, l_2 \in \{0, 1\}$, and $l_1 \neq l_2$. In other words, the goal is to learn the threshold value w_{TC} that divides the interval $[1, H]$ in two sub-intervals in which the examples have the same label (i.e., l_1 or l_2). Figure 3.6 illustrates such a learning task, in which $H = 7$, $w_{TC} = 3$, $l_1 = 0$, and $l_2 = 1$.

For 1-DHL, I make the following assumptions:

- the real-valued variable X has a uniform distribution over the interval $[1, H]$;

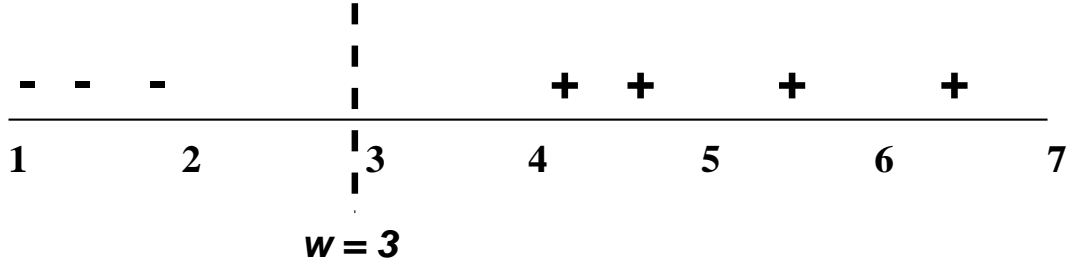


Figure 3.6: Illustrative 1-DHL task ($w_{TC} = 3$, $H = 7$, $l_1 = 0$, and $l_1 = 1$) with four positive and three negative examples.

- the target concept is perfectly learnable (i.e., there is no noise in the domain).

The *2-view, discretized High-Low* learning problem (2-DHL) is a 2-dimensional version of 1-DHL. In 2-DHL, the target function takes the form

$$g : [1, w_{TC}^1] \times [1, w_{TC}^2] \cup (w_{TC}^1, H_1] \times (w_{TC}^2, H_2] \mapsto \{0, 1\},$$

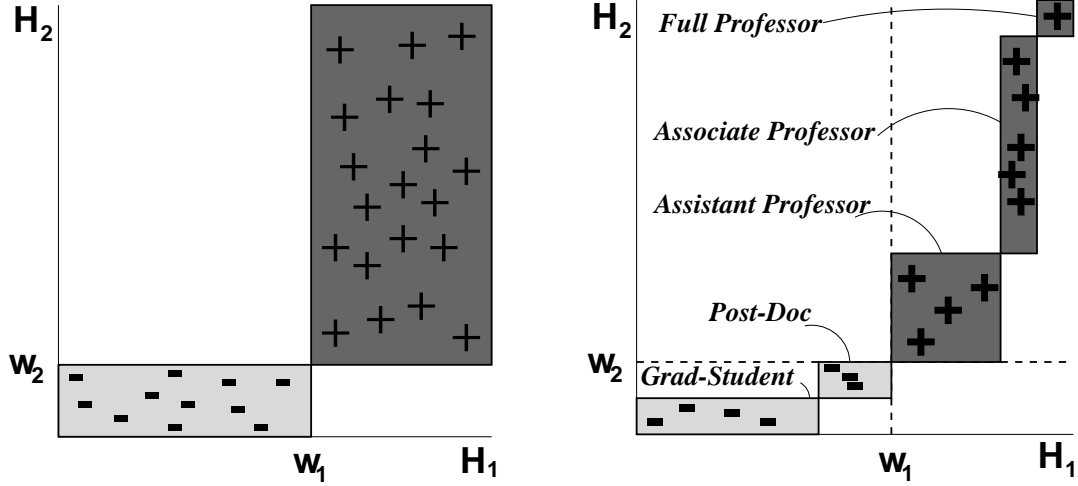
$$g(X, Y) = \begin{cases} l_1 & \text{if } X \leq w_{TC}^1 \text{ and } Y \leq w_{TC}^2 \\ l_2 & \text{if } X > w_{TC}^1 \text{ and } Y > w_{TC}^2 \end{cases}$$

where $w_{TC}^1 \in \{1, 2, 3, \dots, H_1\}$, $w_{TC}^2 \in \{1, 2, 3, \dots, H_2\}$, $l_1, l_2 \in \{0, 1\}$, and $l_1 \neq l_2$.

As shown in Figure 3.7.a, this translates into a 2-dimensional instance space that is divided in four rectangular regions. The top-right and bottom-left quadrants are populated by positive and negatives examples (either class can occupy either quadrant, but each quadrant consists only of examples having the same label), while the two other quadrants remain unpopulated because of h 's domain restriction to $[1, w_{TC}^1] \times [1, w_{TC}^2] \cup (w_{TC}^1, H_1] \times (w_{TC}^2, H_2]$.

The 1-DHL assumptions are also made in 2-DHL:

- the real-valued variables X and Y are uniformly distributed within the rectangles $[1, w_{TC}^1] \times [1, w_{TC}^2]$ and $(w_{TC}^1, H_1] \times (w_{TC}^2, H_2]$;
- the target concept is perfectly learnable.



a) Illustrative 2-DHL target concept. b) Domain with two classes and five clumps.

Figure 3.7: The leftmost image shows how a 2-DHL target concept splits instance space in quadrants; for a domain in which the two views are “independent given the label”, the light and dark grey rectangles also depict the (uniform) distribution of the examples over negative and positive regions in instance space, respectively. In contrast, the rightmost picture shows the (uniform) distribution of the examples within the five clumps of a 2-DHL domain in which the two views are “independent given the clump”.

Note that, by construction, 2-DHL is a multi-view learning problem: the restriction of the domain to $[1, w_{TC}^1] \times [1, w_{TC}^2] \cup (w_{TC}^1, H_1] \times (w_{TC}^2, H_2]$ guarantees that either of the two thresholds, w_{TC}^1 or w_{TC}^2 , is *sufficient* to correctly classify any example. The two resulting views, $\mathbf{V1}$ and $\mathbf{V2}$, consist of the single attributes X and Y , respectively.

In keeping with the multi-view setting, 2-DHL can be decomposed into solving a 1-DHL problem in each view. The corresponding target functions are $g_1 : [1, H_1] \mapsto \{0, 1\}$ and $g_2 : [1, H_2] \mapsto \{0, 1\}$, where

$$g_1(X) = \begin{cases} l_1 & \text{if } X \leq w_{TC}^1 \\ l_2 & \text{if } X > w_{TC}^1 \end{cases}$$

and

$$g_2(Y) = \begin{cases} l_1 & \text{if } Y \leq w_{TC}^2 \\ l_2 & \text{if } Y > w_{TC}^2 \end{cases}$$

The perfect learnability of the target concept $g(X, Y)$ implies that both $g_1(X)$ and $g_2(Y)$ are also perfectly learnable. In turn, this guarantees the compatibility of the views **V1** and **V2**: by definition, any pair of target concepts that label all examples identically are compatible (see chapter 2). In terms of view correlation, I consider two main scenarios:

- **given an example's label, the views are independent.** This is the original multi-view assumption made by Blum and Mitchell (1998): for any example $\langle x, y, ? \rangle$, given its label l , the values x and y are independent of each other. Intuitively, this means that *guessing an example's description in V1 from its description in V2* is as difficult as *guessing an example's description in V1 from its label*. In other words, in terms of guessing an example's description in **V1**, its description in **V2** and its label are equally (un)informative. A similar statement holds for guessing an example's description in **V2**.
- **given an example's clump, the views are independent.** This is a relaxed version of the assumption above: as explained in chapter 2, domain clumpiness, which corresponds to having several sub-classes for each concept of interest, violates the original view independence assumption. For example, as depicted in Figure 3.7.b, in the *faculty - non-faculty* domain, the *faculty* class can be seen as having three clumps: assistant, associate, and full professors. Similarly, the *non-faculty* class has two clumps: graduate students and post-docs.

In my analysis I consider only the simplest form of domain clumpiness, in which the domain of a target concept with C clumps has the form

$$[1, \alpha_1] \times [1, \beta_1] \cup (\alpha_1, \alpha_2] \times (\beta_1, \beta_2] \cup (\alpha_2, \alpha_3] \times (\beta_2, \beta_3] \cup \dots (\alpha_{C-1}, \alpha_C] \times (\beta_{C-1}, \beta_C]$$

where $\alpha_C = H_1$, $\beta_C = H_2$, and $\forall i \in \{1, 2, 3, \dots, C-1\}, \alpha_i < \alpha_{i+1} \& \beta_i < \beta_{i+1}$. Intuitively, this corresponds to a “diagonal” of rectangular clumps that touch each other in a single point; furthermore, the clumps do *not* have overlapping domains (see Figure 3.7.b).

It is easy to see that the “independent given clump” assumption imposes weaker domain constraints than “independent given label” (i.e., it allows for more than one clump per class). Intuitively, the former means that, in terms of guessing an example’s description in $\mathbf{V1}$, the example’s description in $\mathbf{V2}$ is *more informative* than its label: the example’s description in $\mathbf{V2}$ determines the clump it belongs to, thus reducing the range of possible descriptions in $\mathbf{V1}$ to the ones specific to that particular clump. In contrast, knowing the example’s label restricts the set of possible values to a larger set, which covers all the clumps in the class.

3.3.1.0.1 The base learner. As a base learner for both 1-DHL and the two views in 2-DHL, I use the Gibbs algorithm (Mitchell, 1998; Hasenjager, 2000), which proceeds as follows:

- first, it searches through the labeled examples and detects the borders Min and Max of version space. For example, in Figure 3.8, Min is the smallest integer in $[1, H_1]$ that is greater than or equal to the values of all negative examples; similarly, Max is the largest integer lower-bound for the positive examples.
- then it sets the threshold w to a randomly-chosen integer in the range $[Min, Max]$.

Note that the Gibbs algorithm is not the only possible base learned for 1-DHL. For example, one could solve 1-DHL based on a deterministic base learner that sets $w = (Min + Max)/2$. I have chosen to use the Gibbs algorithm for a straightforward reason: Query-by-Committee (Seung, Opper, and Sompolinski, 1992), which is the only existing active learner for which there is a theoretical proof of convergence (Freund et al., 1997), requires a base learner that can randomly sample hypotheses from the version space. The Gibbs algorithms is the simplest base learner that fulfills this requirement.

Three of the considered algorithms, Uncertainty Sampling, Co-Training and Aggressive Co-Testing, rely on the ability of a hypothesis to evaluate the confidence of its predictions. For the base learner above, a prediction’s confidence can be measured as

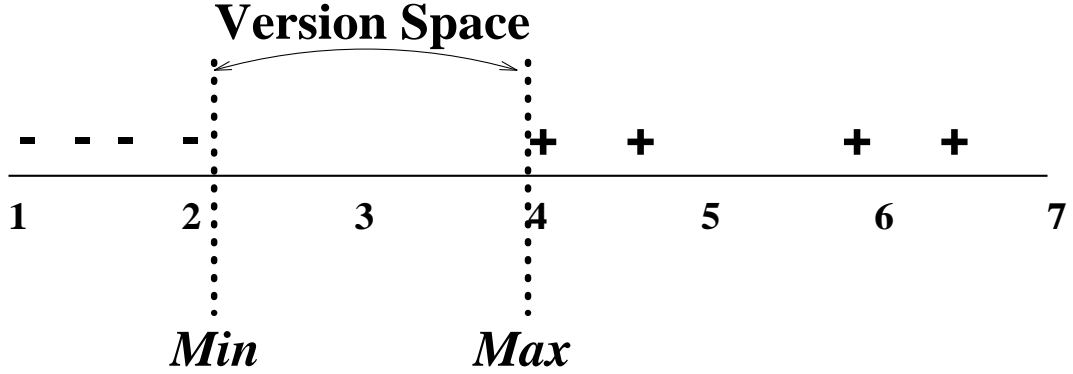


Figure 3.8: Using four positive and four negative examples to determine the *Min* and *Max* borders of the version space for a 1-DHL learning problem.

follows: the closer the value of the unlabeled example to the threshold w , the lower the confidence of the prediction. For instance, for the 1-DHL task depicted in Figure 3.6, the predictions $f(1)$ and $f(7)$ are the most confident negative and positive predictions, respectively. The least confident predictions are the one made near the decision threshold: $f(3 \pm \varepsilon)$, where ε is a small, positive real number.

3.3.2 Convergence properties for single-view algorithms

In this section, I present and comment on the convergence of three single-view algorithms: Query-by-Committee, Uncertainty Sampling, and Random sampling. The proofs of all the results in this chapter are presented in Appendix A.

Proposition 1 *On the 1-DHL problem, the Uncertainty Sampling algorithm requires $\mathcal{O}(\log(H))$ queries to learn the target concept.*

Proposition 2 *On the 1-DHL problem, with an arbitrarily high probability, the Query-by-Committee algorithm requires $\mathcal{O}(\log(H))$ queries to learn the target concept.*

Proposition 3 *On the 1-DHL problem, the probability that the Random Sampling algorithm correctly learns the target concept based on E (randomly chosen) examples is $1 - \frac{2 \times (H-2)^E - (H-3)^E}{(H-1)^E}$.*

The propositions above can be summarized as follows: for the 1-DHL problem, the two active learners converge in a number of queries logarithmic in the size of the version space. In contrast, regardless of the number of queries made by Random Sampling, this third algorithm converges only asymptotically to the target concept (the larger the number of examples, the larger the probability of converging).

Note that for 1-DHL, Uncertainty Sampling and Query-by-Committee make an (near) optimal number of queries. As 1-DHL is equivalent to finding a particular value within a sorted array, it follows that - in the general case - one cannot do better than *binary search*, which requires $\log(H)$ tests (i.e., queries). As I show in the next section, in the multi-view setting one can converge much faster than this.

3.3.3 Convergence properties for multi-view algorithms

3.3.3.1 Views that are independent given the label

The results below show that - in the multi-view setting - one needs just a few examples to learn the target concept. When presented with a training set that consists of one randomly-chosen positive and one randomly-chosen negative example:

- Co-Training learns the target concept based on these two labeled and many unlabeled examples.
- Co-Testing, besides the two examples in the initial training set, makes at most four additional queries to learn the target concepts in both views.

In contrast, the optimal, domain-specific strategy requires just one randomly-chosen labeled example x : given that each of the two populated quadrants in Figure 3.7.a consists of examples that have the same label, it follows that all examples in x 's quadrant share its label, while the examples in the other quadrant have the other label.

Proposition 4 *By using domain-specific knowledge, one can solve the 2-DHL problem based on a single, randomly-chosen query.*

Proposition 5 *On the 2-DHL problem, the Co-Training algorithm requires only one random positive and one random negative examples to learn the target concept.*

Proposition 6 *On the 2-DHL problem, when provided with one random positive and one random negative examples, Aggressive Co-Testing requires at most **four** queries to learn the target concepts in both views.*

3.3.3.2 Views that are independent given the clump

The results above could be proved only because of the extremely strong (and unrealistic) assumption that the views are independent given the label. I relax now this condition by assuming that the views are independent given the clump, which still allows faster converge than single-view algorithms. These theoretical results are further reinforced by the controlled, text-classification experiment described in chapter 4.

In this section I denote the number of clumps in the domain by $NmbClumps$. I also assume that the number of clumps is relatively small compared with the size of the domain (i.e., $NmbClumps \ll \min(H_1, H_2)$).

Proposition 7 *By using domain-specific knowledge, one can solve the 2-DHL problem based on $O(\log(NmbClumps))$ queries.*

Proposition 8 *Depending on the distribution of the examples in the initial training set, Co-Training may or may not learn the target concept for the 2-DHL problem.*

Proposition 9 *On the 2-DHL problem, with an arbitrarily-high probability, Aggressive Co-Testing learns the target concept in both views by making at most $2 \times NmbClumps$ queries.*

The intuition behind these results is the following: in order to “discover” all clumps and label them correctly, Aggressive Co-Testing requires a number of queries linear in $NmbClumps$. For $NmbClumps \ll \min(H_1, H_2)$, this means that Aggressive Co-Testing

convergences faster than Query-by-Committee, Uncertainty Sampling, and Random Sampling.

In contrast, Co-Training cannot be guaranteed to converge unless its randomly chosen training set contains at least a positive and a negative example from the “decision-border clumps” (e.g., in Figure 3.7.b, the “decision-border clumps” are the ones corresponding to post-docs and assistant professors).

Finally, the optimal, domain-specific solution converges in $\mathcal{O}(\log(NmbClumps))$ queries by performing binary search in the space of clumps. In the degenerate scenario in which the views are totally correlated (i.e., $NmbClumps = H_1 = H_2 = H$), this is equivalent to the $\mathcal{O}(\log(H))$ queries made by Query-by-Committee and Uncertainty Sampling.

3.3.4 Discussion

The results above require several comments. First of all, in contrast to the single-view learning scenario, the multi-view setting imposes powerful constraints on the distribution of the examples in the domain. By exploiting these constraints, a multi-view learner can converge to the target concept much faster than its single-view counterparts. Both Co-Training and Co-Testing implicitly exploit the multi-view assumption that all the examples should have the same label in both views; the former does it by bootstrapping the views from each other, while the latter asks the user to label the examples that do not seem to have the same label.

Second, the analysis above assumes that the domain is noise-free, which - in turn - implies that the views are compatible. Extending the results for noisy domains is an extremely difficult task, even in the single-view framework. One straightforward observation is the following: on noisy domains, an active learner is likely to make more queries because querying a noisy example is either uninformative or misleading. Quantifying the exact effect of the noise is tightly related to the distribution of the noisy examples over the instance space, which - in turn - varies from one learning task to another.

Finally, in my analysis I have considered both the ideal scenario, in which the views are independent given the label, and a more realistic one, in which the views are independent given the clump. On problems with several clumps per class, Co-Training cannot be guaranteed to work properly, while Co-Testing still converges in a number of queries that is linear in the number of clumps. For a relatively small number of clumps this implies faster convergence than the single-view learners. As the views become more correlated (i.e., the number of clumps increases), the advantages of multi-view learning tend to disappear.

3.4 Co-Testing for wrapper induction

In wrapper induction, each item of interest is described by three strings of variable length: the item's content, together with its prefix and suffix within the document. As this is not a typical machine learning representation in which an example's description in each view consists of a fixed number of feature, I describe here in detail how Co-Testing can be applied to wrapper induction. As a first step, I introduce the basic ideas in STALKER (Muslea, Minton, and Knoblock, 2001), which is the state of the art wrapper induction algorithm that I use as base learner. Consider the illustrative task of extracting phone numbers from documents similar to the Web-page fragment shown in Figure 3.5. In STALKER, an *extraction rule* consists of a *start rule* and an *end rule* that identify the beginning and the end of the item, respectively. Given that start and end rules are extremely similar, for the time being I describe only the former. For instance, in order to find the beginning of the phone number, one can use the start rule

R1 = SkipTo(Phone :<i>)

This rule is applied *forward*, from the beginning of the page, and it ignores everything until it finds the string `Phone:<i>`. For a slightly more complicated extraction task, in which only the toll-free numbers appear in italics, one can use a disjunctive start rule such as

$\mathbf{R1}' = \textit{either SkipTo(Phone :<i>)} \\ \textit{or SkipTo(Phone :)}$

An alternative way to detect the beginning of the phone number is to use the start rule

$\mathbf{R2} = \textit{BackTo(Fax) BackTo((Number))}$

which is applied *backward*, from the *end* of the document. $\mathbf{R2}$ ignores everything until it finds “Fax” and then, again, skips back to the first number between parentheses.

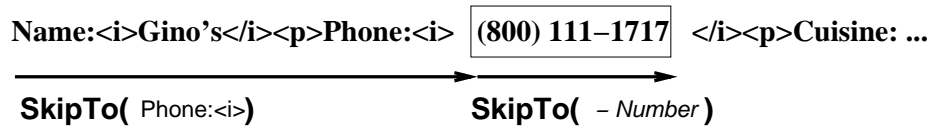
As described in (Muslea, Minton, and Knoblock, 2001), rules such as $\mathbf{R1}$ and $\mathbf{R2}$ can be learned based on user-provided examples of items to be extracted. Note that $\mathbf{R1}$ and $\mathbf{R2}$ represent descriptions of the *same concept* (i.e., start of phone number) that are learned in two *different views*. That is, the views $\mathbf{V1}$ (*forward view*) and $\mathbf{V2}$ (*backward view*) consist of the sequences of characters that *precede* and *follow* the beginning of the item, respectively.

3.4.1 Naive Co-Testing with STALKER extraction rules

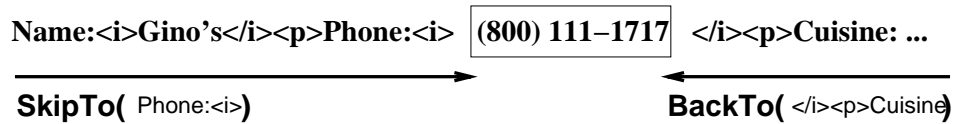
As mentioned above, in order to extract an item of interest, STALKER must detect both where the item starts and where it ends. In other words, for each item, STALKER learns *two* target concepts: the start of the item (i.e., the *start rule*) and the end of the item (i.e., the *end rule*). As for both the start and end rules Co-Testing learns both a forward and a backward rule, one can obtain several types of extraction rules by simply using various combinations of forward/backward, start/end rules. Three of these possible combinations turned out to be of practical importance (Muslea, Minton, and Knoblock, 2000a):

- **FF extraction rules:** use a **F**orward start rule and a **F**orward end rule;
- **FB extraction rules:** use a **F**orward start rule and a **B**ackward end rule;
- **BB extraction rules:** use a **B**ackward start rule and a **B**ackward end rule.

Forward–Forward Rule (FF):



Forward–Backward Rule (FB):



Backward–Backward Rule (BB):

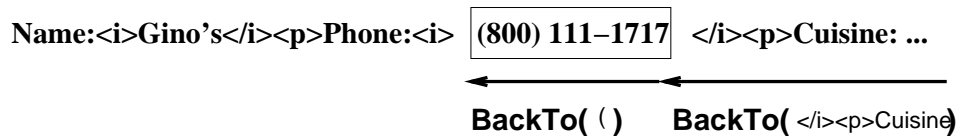


Figure 3.9: Three types of extraction rules: **FF**, **FB**, and **BB**.

Figure 3.9 illustrates how the **FF**, **FB**, and **BB** rules work. For example, a **FF** rule first detects the start of the item by applying the forward start rule; then the forward end rule is applied **from the point where the start rule matched**. The **BB** rule works in a similar manner: first, it detects the end of the item by using the backward end rule; then the start is found by applying the backward start rule **from the point where the end rule matched**. Finally, a **FB** rule detects the start and end of the item independently of each other by simply applying the start and end rules from the start and end of the document, respectively.

Given the forward and backward views, applying Co-Testing to wrapper induction is a straightforward process. First, STALKER uses the labeled examples to learn forward/backward start and end rules. Then these rules are used to create the **FF**, **FB**, and

BB extraction rules. The contention points are unlabeled examples on which at least two of the three extraction rules do not extract the same string. Finally, as STALKER does not provide an estimate of the confidence of each extraction, the only Co-Testing algorithm that can be used based on the forward and backward views is *Naive Co-Testing* with the *winner-takes-all* output hypothesis. That is:

- each query is randomly chosen among the contention points (*Naive Co-Testing*);
- the output hypothesis is the one among the **FF**, **FB**, and **BB** rules that makes the fewest mistakes on the queries (i.e., *winner-takes-all*).

3.4.2 Aggressive Co-Testing with strong and weak views

The forward and backward STALKER views lead to extraction rules that rely mostly on the *context* of the item to be extracted (i.e., the text surrounding the item in the document). As described earlier, in addition to these two strong views, one can use a third, content-based, *weak view*, which describes the actual item to be extracted. For example, when extracting phone numbers, one may be able to exploit the fact that they can be described by a simple grammar: “(*Number*) *Number* - *Number*”. Similarly, when extracting URLs, one can take advantage of the fact that a typical URL starts with the string “http://www.”, ends with the string “.html”, and contains no HTML tags.

I use the following features to describe the content of each item to be extracted:

- the *length range* (in tokens) of the seen examples. For instance, all phone number in the format “(*Number*) *Number* - *Number*” consist of six tokens (i.e., the three numbers together with the dash and the two parentheses).
- the *token types* that appear in the seen examples. This feature consists of a set of the most specific *wildcards* (e.g., *Number*, *AllCaps*, etc) that match the tokens encountered in the item to be extracted. For example, in the phone number case, this list consists of two wildcards: *Number* and *Punctuation*. The complete hierarchy of wildcards is described in Figure 3.10.

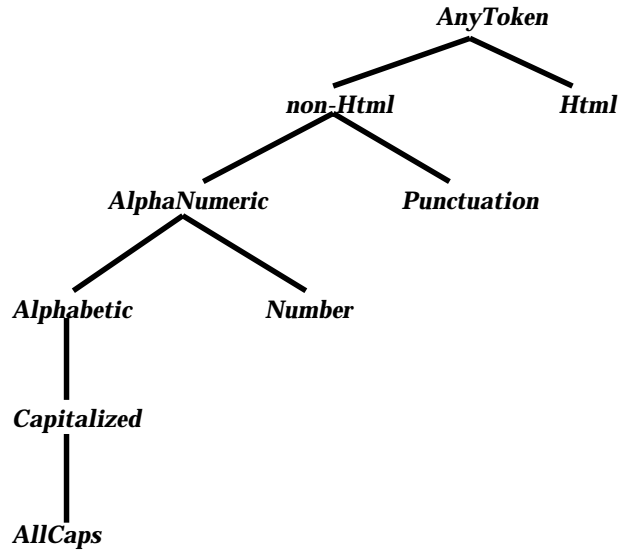


Figure 3.10: The hierarchy of wildcards used by the STALKER wrapper induction algorithm. The parent-child relationship in the tree denotes the *IsMoreGeneralThen* relationship. For example, the most general wildcard is *AnyToken*, which matches all possible tokens. The *nonHtml* wildcard, which is a child of *AnyToken*, denotes all tokens than are not HTML tags (i.e., alphanumeric tokens and punctuation signs).

- a *start-pattern* such as “`http://www.`” or “(*Number*)”, which describes the template at the beginning of the item of interest.
- an *end-pattern* such as “`AlphaNum .html`” or “*Number* - *Number*”, which describes the template at the end of the item of interest.

In order to learn the content-based description of an item, I use a base learner that can be seen as a simplified version of the DataPro algorithm (Lerman and Minton, 2000; Knoblock et al., 2002). After tokenizing each of the user-provided examples of the item to be extracted, the weak-view learner proceeds as follows:

- the *length range* is determined by finding the examples that contain the largest and the smallest number of tokens;
- the *token types* is obtained by going through the tokens that appears in the labeled examples and adding to the set of “seen types” the most specific wildcard that covers it.

- a start-pattern of *length one* consists of the *most specific* wildcard that covers the first token in all labeled examples (note: in case all examples start with the same token, such as “” in the phone number example, the actual token is preferred to the most specific wildcard). A start-pattern of length k can be generated by repeating the procedure above for the first, second, third, \dots , up to k -th position.
- the end-pattern is learned in the same manner as the start pattern, but using the k tokens at the end of the item.

Given the forward, backward, and content-based views, one can implement an *aggressive* version of Co-Testing for wrapper induction. First, STALKER uses the labeled examples to learn forward/backward start and end rules, which are then used to create the **FF**, **FB**, and **BB** extraction rules. Based on the same examples, one can also use the weak-view learner above to generate the corresponding content-based rule. The contention points are, again, unlabeled examples on which at least two of the three extraction rules do not extract the same string. Based on the three views above, I now present a more sophisticated version of Co-Testing:

- the next query is the contention point of which most of the **FF**, **FB**, and **BB** rules violate *as many as possible* of the constraints learned in the weak view. That is, the extracted strings are longer/shorter than the seen examples, they contain types of tokens that were not encountered in the training set, and the start- and end- patterns do not match. This is an *aggressive query selection strategy*¹ because violating as many as possible of the constraints learned in the weak views means that the content-based hypothesis is maximally confident that the STALKER rules are *not* extracting the correct string.
- the output hypothesis is obtained by *majority voting*. More precisely, given a new, unseen document, the **FF**, **FB**, and **BB** rules are applied to the document and

¹Remember that by an *aggressive* query selection strategy I mean querying the contention point on which the hypotheses make the most confident prediction.

the “winner” is the STALKER rule that violates the smallest number of constraints learned in the weak view. Note that this is an extremely flexible approach, which allows Co-Testing to use the most appropriate type of rule (i.e., **FF**, **FB**, or **BB**) for each individual document.

3.5 Empirical Evaluation

This section describes several sets of experiments in which I compare Co-Testing with other active learning algorithms. First I present results for wrapper induction, which is my main motivating problem. Then I discuss the experiments conducted on three additional real-world domains in which the features can be naturally partitioned in two views.

3.5.1 Wrapper induction experiments

3.5.1.1 The six algorithms in the comparison

In practice, it is difficult to find algorithms against which to meaningfully compare the performance of Co-Testing: despite the importance of learning high-accuracy wrappers based on a minimal number of labeled examples, there are no reported results on active learning for wrapper induction.² Furthermore, most of the existing active learners cannot be applied in a straightforward manner to base learners such as the STALKER wrapper induction algorithm:

- Uncertainty Sampling (Lewis and Gale, 1994) cannot be used because STALKER is unable to evaluate the confidence of its predictions;
- Query-by-Committee (Seung, Opper, and Sompolinski, 1992) cannot be applied because STALKER cannot randomly sample hypotheses from the version space;

²The only related approaches, (Thompson, Califf, and Mooney, 1999) and (Soderland, 1999), were designed for systems that learn extraction rules from free text. These active learners are similar to *Uncertainty Sampling* (Lewis and Gale, 1994) and were crafted based on heuristics specific to their respective base learners, RAPIER and WHISK.

- SG-net (Cohn, Atlas, and Ladner, 1994) cannot be used because STALKER cannot generate *most specific* and *most general* extraction rules;
- Query-by-Boosting (Abe and Mamitsuka, 1998) cannot be applied because STALKER rarely makes mistakes on the training set, thus annihilating the ability of the boosting algorithm (Schapire, 1990; Bauer and Kohavi, 1999) to generate a diverse committee.

Query-by-Bagging (Abe and Mamitsuka, 1998) is the only existing active learning algorithm that can be applied in a straightforward manner to STALKER (and, more generally, to wrapper induction). Query-by-Bagging works by generating a committee of extraction rules and querying unlabeled examples on which the committee is the most split (i.e., the rules in the committee extract the largest number of distinct strings). The committee of extraction rules is created by repeatedly re-sampling (with substitution) the examples in the original training set L .

In the empirical evaluation below, I compare Naive and Aggressive Co-Testing with Random Sampling and Query-by-Bagging. Random Sampling, which is used as strawman, is identical with Naive Co-Testing with *winner takes all*, except that it randomly queries one of the unlabeled examples from the working set. For Query-by-Bagging, I create a committee of 10 extraction rules, each of which is obtained by running STALKER on a training set generated by re-sampling L .³ The output hypothesis for Query-by-Bagging works by *majority voting* the 10 extraction rules. Given that STALKER can generate three types of extraction rules (i.e., **FF**, **FB**, and **BB**), I run Query-by-Bagging for each type of rule and report three sets of results: Query-by-Bagging(**FF**), Query-by-Bagging(**FB**), and Query-by-Bagging(**BB**).

³For Query-by-Bagging, I use a relatively small committee (i.e., a 10-rule committee) because of the scarcity of the training data: as wrapper induction algorithms are expected to learn the extraction rules based on a handful of examples, *sampling-with-replacement* from the small original training set leads to few possible training sets for the members of the committee.

3.5.1.2 The experimental setup

In order to empirically compare the algorithms above, I use the wrapper induction testbed introduced by Kushmerick (1998, 2000). It consists of 206 extraction tasks from 30 Web-based information sources⁴. As shown in (Muslea, Minton, and Knoblock, 2001), on most of these 206 tasks STALKER learns a 100% accurate extraction rule based on just one or two randomly chosen labeled examples. In this empirical evaluation I consider the 33 most difficult extraction tasks in the testbed:

- the 28 tasks on which, based on 20 random examples, STALKER fails to learn 100%-accurate rules of at least one of the three types (i.e., **FF**, **FB**, and **BB**);
- the five additional tasks on which STALKER requires a larger than usual number of randomly-chosen examples to learn 100%-accurate rules of all three types (i.e., at least seven examples).

For each of the 33 tasks, Table 3.1 provides the following information:

- task identifier (for example, **S1-0** designates task **0** from source **S1**);
- original source name from (Kushmerick, 2000);
- name of the item to be extracted;
- total number of examples in the domain;

Table 3.1 also provides information about the performance of stand-alone STALKER (i.e., with up to 20 randomly chosen examples). Columns 5-7 show the number of examples required to reach 100% accuracy for **FF**, **FB**, and **BB** rules, respectively. If a 100%-accurate rule is not learned, Table 3.1 shows instead the accuracy reached based on 20 random examples.

For each of the 33 learning tasks, I use 20-fold cross-validation to compare the performance of Co-Testing with that of its randomized counterpart and the three versions

⁴All these datasets, together with a detailed description of each extraction task, can be obtained from the RISE repository, which is located at <http://www.isi.edu/~muslea/RISE/index.html>.

Task ID	Source name	Item name	Nmb exs	STALKER		
				FF	FB	BB
S1-0	Computer ESP	Price	404	8	8	8
S1-1		URL	404	98.76%	96.53%	96.53%
S1-2		Item	404	7	96.53%	95.78%
S2-0	CNN/Time AllPolitics	URL	501	99.00%	99.00%	99.00%
S2-1		Source	501	97.99%	97.39%	94.82%
S2-2		Title	499	79.63%	93.89%	93.79%
S2-3		Date	492	91.63%	91.63%	97.74%
S3-0	Film.com Search	URL	175	99.43%	9	9
S3-1		Name	175	99.43%	99.43%	99.43%
S3-3		Size	175	99.43%	99.43%	99.43%
S3-4		Date	175	5	14	14
S3-5		Time	175	98.85%	98.85%	14
S6-1	PharmaWeb WWLPS	University	27	96.15%	92.31%	96.15%
S9-10	Internet	Arival Time	44	95.35%	95.35%	95.35%
S9-11	Travel Network	Availability	39	97.37%	97.37%	13
S11-1	Internet	Email	91	13	13	6
S11-2	Address	Update	91	7	71.11%	71.11%
S11-3	Finder	Organization	72	98.59%	71.83%	71.83%
S15-1	NewJour: EJ&N	Name	355	99.15%	99.15%	99.15%
S19-1	Shops.Net	Score	201	99.00%	99.00%	99.00%
S19-3		Item Name	201	97.00%	97.00%	97.00%
S20-3	Democratic Party Online	Score	91	99.18%	99.18%	97.55%
S20-5		File Type	328	99.18%	99.18%	99.18%
S24-0	Foreign	Language	690	8	8	8
S24-1	Languages for	URL	424	16	98.87%	95.04%
S24-3	Travelers	Translation	690	94.34%	94.34%	85.90%
S25-0	U.S. Tax Code	URL	328	98.47%	14	14
S26-3	CD Club Web Server	Price	377	98.94%	99.20%	99.20%
S26-4		Artist	377	96.01%	94.02%	94.82%
S26-5		Album	377	90.69%	90.69%	37.33%
S28-0	Cyberider	URL	751	97.73%	99.07%	99.07%
S28-1	Cycling WWW	Relevance	751	98.40%	3	19
S30-1	Congress Qtrly	Person Name	30	7	6	7

Table 3.1: The 33 extraction tasks used in the empirical evaluation.

of Query-by-Bagging that learn **FF**, **FB**, and **BB** rules, respectively. All six algorithms start with two randomly chosen examples and make 18 successive queries. The reported error rate is averaged over the 20 folds.

3.5.1.3 The empirical results

Figure 3.11 shows illustrative learning curves obtained on three of the 33 learning tasks. These graphs correspond to the three main scenarios encountered in practice:

- the two Co-Testing algorithms learn 100% accurate rules, while *all* the other four algorithms fail to do so. This scenario is depicted on the top graph, and it occurs in 12 of the 33 tasks.
- the two Co-Testing algorithms and *at least* another algorithm learn a 100% accurate rule, but Co-Testing requires fewer queries. This scenario is depicted in the middle graph, and it occurs on 18 of the 33 tasks.
- none of the six considered algorithms learns a 100% accurate rule (three tasks).

Figure 3.12 summarizes the performance of the six algorithms over the 33 extraction tasks. In each of the six graphs, the **X** axis shows the number of queries made by the algorithm, while the **Y** axis shows the number of tasks for which a 100% accurate rule was learned based on exactly **X** queries. All algorithms start with 2 random examples and make 18 additional queries; *by convention*, the “19 queries” data point corresponds to the tasks on which the algorithm cannot learn a 100% accurate rule based on 20 labeled examples.

As shown in Figure 3.12. the two Co-Testing algorithms clearly outperform their single-view counterparts, with Aggressive Co-Testing doing significantly better than Naive Co-Testing. Aggressive Co-Testing learns 100% accurate rule on 30 of the 33 tasks, and, what is more, all these 30 extraction rules are learned based on at most seven queries. Note than in one third of the tasks (11 of 33), a single, “aggressively-chosen” query is sufficient to learn the correct extraction rule. In contrast, Naive Co-Testing succeeds at

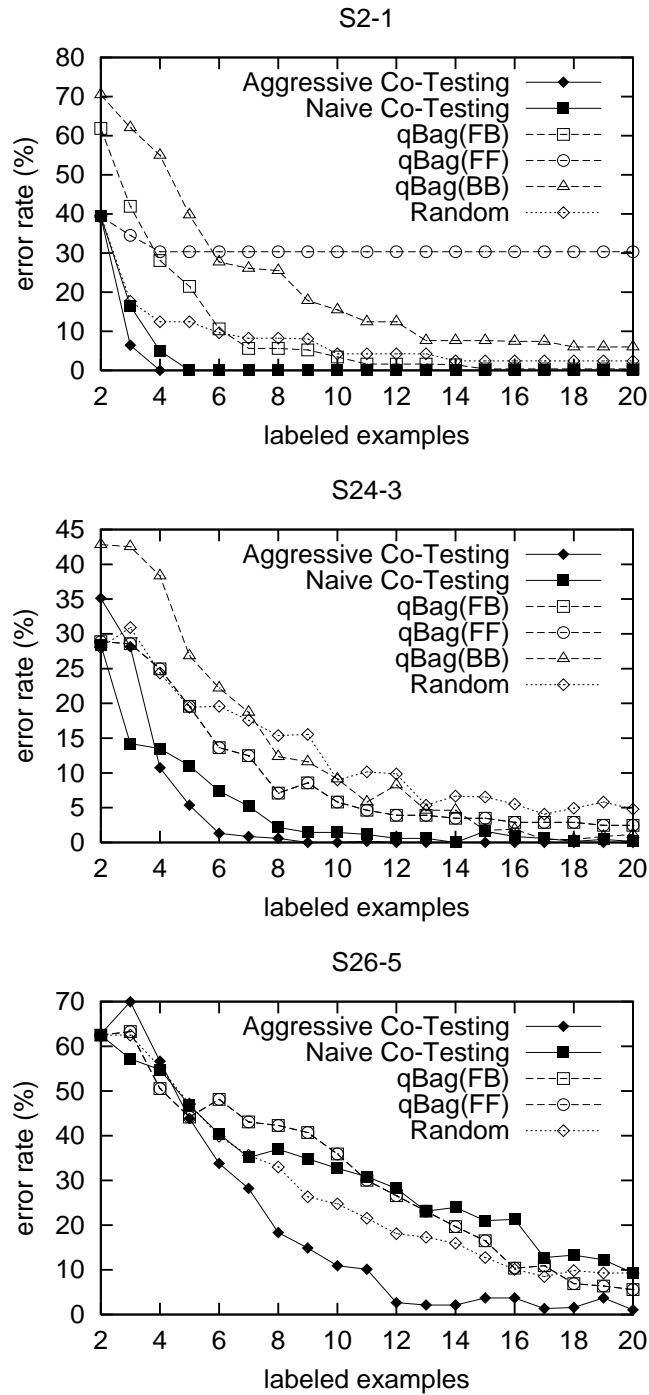


Figure 3.11: Illustrative results on three wrapper induction tasks: **S2-1**, **S24-3**, and **S26-5**. For clarity, the bottom graph does not include the results for **qBag(BB)**, which reaches an error rate of only 70%.

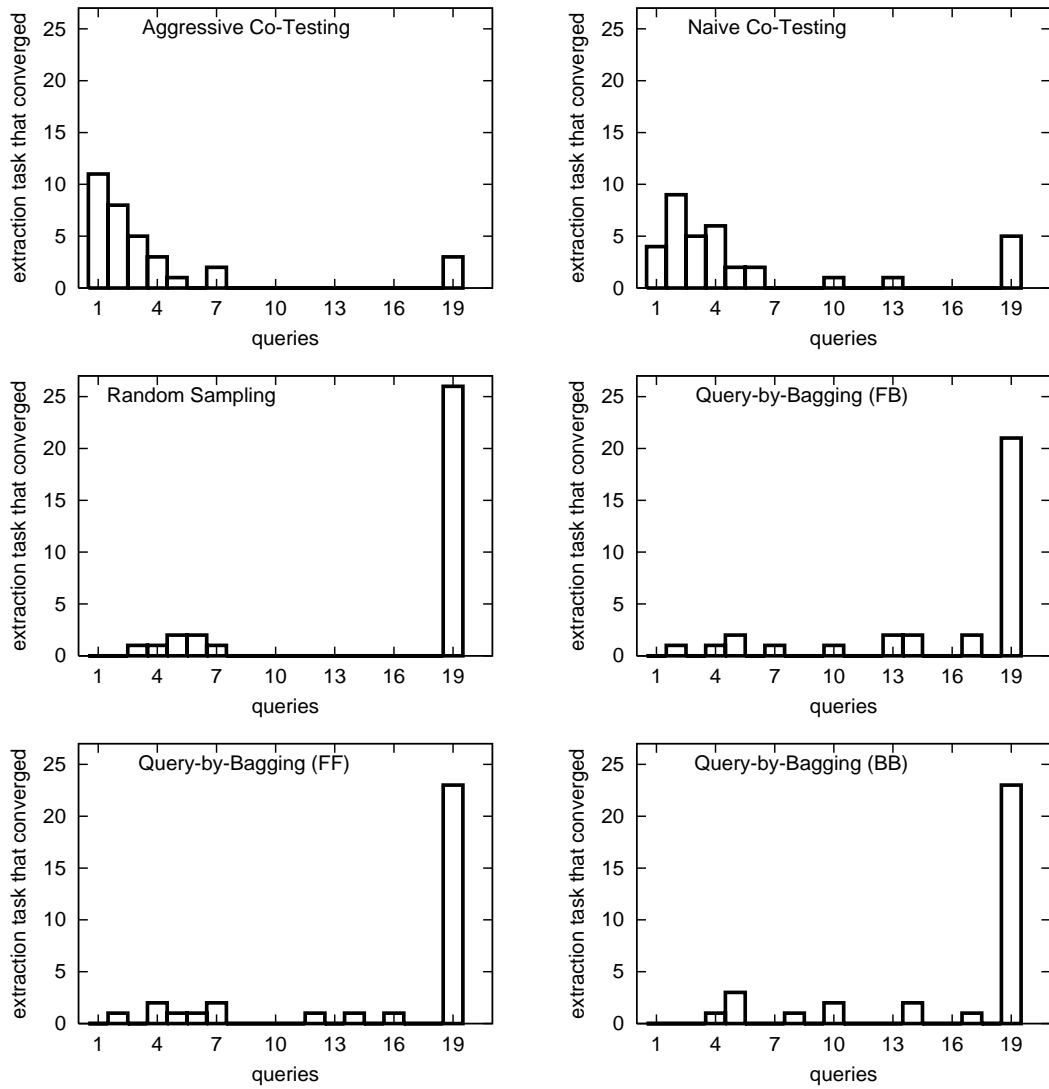


Figure 3.12: Convergence results for the 33 wrapper induction tasks.

converging in a single query on just four of the 33 tasks, while the single-view algorithms are not able to do so even for a single task.

Naive Co-Testing learns 100% accurate rules on 28 of the 33 tasks. On 26 of these 28 tasks, the extraction rules are learned based on at most six queries. In contrast, Random Sampling leads to only seven rules that are 100% accurate, with Query-by-Bagging slightly outperforming it: when learning **FF**, **FB**, and **BB** extraction rules, Query-by-Bagging obtains 12, 10, and 10 rules that are 100% accurate, respectively. In other words, the two Co-Testing algorithms learn the correct target concept in more than twice as many tasks than Query-by-Bagging.

Remember that on three of the 33 tasks, Aggressive Co-Testing fails to learn a 100% accurate rule. For example, on **S11-3**, both Query-by-Bagging(**FF**) and Random Sampling obtain more accurate rules than Aggressive Co-Testing; furthermore, on **S11-3** and **S26-5**, at least one of the single-view algorithms also outperforms Naive Co-Testing. This situation is due to the fact that, on both extraction tasks, the **backward** view is significantly less accurate than the **forward** one (see Table 3.1). Such circumstances lead to a large number of contention points that are mislabeled by the “bad view,” in which one cannot learn the correct rule even if provided with all available examples. Consequently, the distribution of the queries is skewed towards mistakes of the “bad view”, which are not informative for either view: the “good view” makes the correct prediction on them, while the “bad view” is inadequate to learn the target concept. In order to cope with this problem, in (Muslea, Minton, and Knoblock, 2002b) we introduced a view validation algorithm that predicts whether or not the views are appropriate for a particular task.

Finally, to conclude the discussion on wrapper induction, I will briefly compare the results above with the ones obtained by WIEN (Kushmerick, 2000), which is the only wrapper induction system for which there are published empirical results for all the extraction tasks in the testbed used here. As the two experimental setups are not identical⁵, this is

⁵WIEN was not evaluated based on cross-validation, but rather by randomly splitting of the available examples into training and test sets of various sizes.

just an *informal* comparison. However, this comparison puts our results into perspective by contrasting Co-Testing with another approach to wrapper induction.

The results can be summarized as follows: WIEN, which uses random sampling, fails to learn an extraction rule for 18 of the 33 task (as apposed to STALKER, WIEN does not return any extraction rule unless it works perfectly on the training set). This set of 18 tasks includes both the three tasks on which Aggressive could not learn a 100% accurate rule and the five tasks on which Naive Co-Testing failed to learn perfect rules. On the remaining 15 of the 33 tasks, WIEN requires between 25 and 90 examples⁶ to learn the correct rule. For the same 15 tasks, both Aggressive and Naive Co-Testing learn 100% accurate rules based on at most eight examples (two random plus at most six queries).

3.5.2 Beyond wrapper induction

In order to further investigate Co-Testing’s performance, I applied it to three additional real-world domains for which there is a natural, intuitive way to create two views:

- AD (Kushmerick, 1999) is a classification problem with two classes, 1500 attributes, and 3279 examples. In AD, images that appear in Web pages are classified into **ads** and **non-ads**. The view **V1** consists of all textual features that describe the image; e.g., 1-grams and 2-grams from the caption, from the URL of the page that contains the image, from the URL of the page the image points to, etc. In turn, **V2** describes the properties of the image itself: length, width, aspect ratio, and “origin” (i.e., are the image and the page that contains it coming from the same Web server?).
- COURSES (Blum and Mitchell, 1998) is a domain with two classes, 2206 features, and 1042 examples. The learning task consists of classifying Web pages as **course homepages** and **other pages**. In COURSES the two views consist of words that

⁶In the WIEN framework, an example consists of a document in which all items of interest are labeled. For example, a page that contains a list of 100 names, all labeled, represents a single labeled example. In contrast, for STALKER the same labeled document represents 100 labeled examples. In order to compare the WIEN and STALKER results, I convert the WIEN data to STALKER-like data by multiplying the number of labeled WIEN pages by the average number of item occurrences in each page.

appear in the page itself and words that appear in hyperlinks pointing to them, respectively.

- TF (Marcu, Carlson, and Watanabe, 2000) is a classification problem with seven classes, 99 features and 11,193 examples. In the context of a machine translation system, it uses the shift-reduce parsing paradigm to learn how to rewrite Japanese discourse trees as English-like discourse trees. In this case, **V1** uses features specific to a shift-reduce parsing paradigm: the elements in the input list and the partial trees in the stack. **V2** consists of features specific to the Japanese tree given as input.

Table 3.2 shows the learning algorithms used in the empirical comparison. For each domain, the base learner \mathcal{L} is the one that obtains the best performance over the entire data set (10-fold cross-validation). That is, IB (Aha, 1992b) for AD, Naive Bayes (Blum and Mitchell, 1998) for COURSES, and MC4, the $\mathcal{MLC}++$ (Kohavi, Sommerfield, and Dougherty, 1997) implementation of C4.5, for TF. The five single-view algorithms from Table 3.2 use all available features (i.e., **V1** \cup **V2**) to learn the target concept.

On all three domains, Random Sampling (**Rnd**) is used as strawman. Because MC4 does not provide an estimate for the confidence in its predictions, Uncertainty Sampling (Lewis and Gale, 1994) (for short, **US**) can be applied only on AD and COURSES. Query-by-Bagging and -Boosting (Abe and Mamitsuka, 1998), denoted by **qBag** and **qBst**, are run on all three domains, while Query-by-Committee (**QBC**) can be applied only on COURSES.⁷ For Query-by-Committee I use a (typical) two-hypothesis committee, while the committees in Query-by-Bagging and -Boosting consist of five hypotheses.⁸

As IB and MC4 do not reliably estimate the confidence in a predicted label, on AD and TF use Naive Co-Testing with a *winner-takes-all* output hypothesis; that is, each query

⁷Query-by-Committee cannot be used on AD and TF because there is no known method for sampling from the IB or MC4 version spaces. For Naive Bayes, I use an idea from (McCallum and Nigam, 1998b), where a committee is created by sampling hypotheses according to the (Gamma) distribution of the Naive Bayes parameters estimated from the training set L .

⁸For Query-by-Bagging and -Boosting, I use a (relatively small) 5-hypothesis committees because of the CPU constraints: the running time increases linearly with the number of learned hypotheses, and, in some domains, it takes more than 50 CPU hours to complete the experiments.

Domain	\mathcal{L}	Co-Testing		Single-view Algorithms				
		Query Selection	Output Hypothesis	QBC	qBag	qBst	US	Rnd
AD	IB	<i>naive</i>	<i>winner</i>	–	✓	✓	✓	✓
TF	MC4	<i>naive</i>	<i>winner</i>	–	✓	✓	–	✓
COURSES	Naive	<i>naive</i>	<i>weighted vote</i>	✓	✓	✓	✓	✓
	Bayes	<i>conservative</i>						

Table 3.2: Algorithms used in empirical comparison on AD, COURSES, and TF. **QBC**, **qBag**, **qBst**, **US**, and **Rnd** denote Query-by-Committee, Query-by-Bagging, Query-by-Boosting, Uncertainty Sampling and Random Sampling, respectively.

is randomly selected among the contention points, and the output hypothesis is the one learned in the view that makes the fewest mistakes on the queries. In contrast, for COURSES I follow the methodology in (Blum and Mitchell, 1998), where the output hypothesis consists of the *weighted vote* of the classifiers learned in each view. Furthermore, on COURSES I also investigate two of the Co-Testing query selection strategies: *naive* and *conservative*.

The performance of all algorithms is evaluated based on 10-fold, stratified cross validation. On AD, each algorithm starts with 150 randomly chosen examples and makes 10 queries after each of the 40 learning episodes. On COURSES, the algorithms start with 6 randomly chosen examples and make one query after each of the 175 learning episodes. Finally, on TF the algorithms start with 110 randomly chosen examples and make 20 queries after each of the 100 learning episodes.

Figures 3.13 and 3.14 display the learning curves of the various algorithms on AD, TF, and COURSE. On all three domains, Co-Testing reaches the highest accuracy (i.e., smallest error rate). Table 3.3 summarizes the statistical significances results (t -test confidence of at least 95%) obtained in a pair-wise comparison of the various algorithms. These comparisons are performed on the right-most half of each learning curve (i.e., towards convergence). The best way to explain the results in Table 3.3 is via examples: the results of comparing Naive Co-Testing and Random Sampling on AD appear in the first three columns of the first row. The three numbers (i.e., 0, 0, and 19) mean that on (all)

Algorithm	Naive Co-Testing						Conservative Co-Testing		
	AD			TF			COURSES		
	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie	Win
Random Sampling	0	0	19	0	21	70	0	0	49
Uncertainty Sampling	0	2	17	0	2	89	-	-	-
Query-by-Committee	-	-	-	0	60	31	-	-	-
Query-by-Bagging	0	18	1	0	6	85	0	28	21
Query-by-Boosting	0	15	4	0	0	91	0	0	49
Naive Co-Testing	-	-	-	-	-	-	0	21	28

Table 3.3: Statistical significance results in the empirical (pair-wise) comparison of the various algorithms on the three domains.

19 comparison points Naive Co-Testing outperforms Random Sampling in a statistically significant manner. Similarly, comparing Naive and Conservative Co-Testing on COURSES (the last three columns on the last row) leads to the following results: on 28 of the comparison points Conservative Co-Testing outperforms Naive Co-Testing in a statistically significant manner; on 21 other points the differences are statistically insignificant; finally, on *no* comparison point Naive Co-Testing outperforms its Conservative counterpart.

The results in Table 3.3 can be summarized as follows. First of all, *no single-view algorithm* outperforms Co-Testing in a statistically significant manner on *any* of the comparison points. Furthermore, except for the comparison with Query-by-Bagging and -Boosting on AD, where the difference in accuracy is statistically insignificant on almost all comparison points, Co-Testing clearly outperform all algorithms on all domains.

3.6 Discussion

The work on Co-Testing was inspired by the original Co-Training paper (Blum and Mitchell, 1998), in which the authors formalized for the first time the idea of multi-view learning. Previously, this topic was largely ignored, though the idea clearly shows up in applications such as word sense disambiguation (Yarowsky, 1995) and speech recognition (de Sa and Ballard, 1998). Rather than considering active learning methods, Blum and

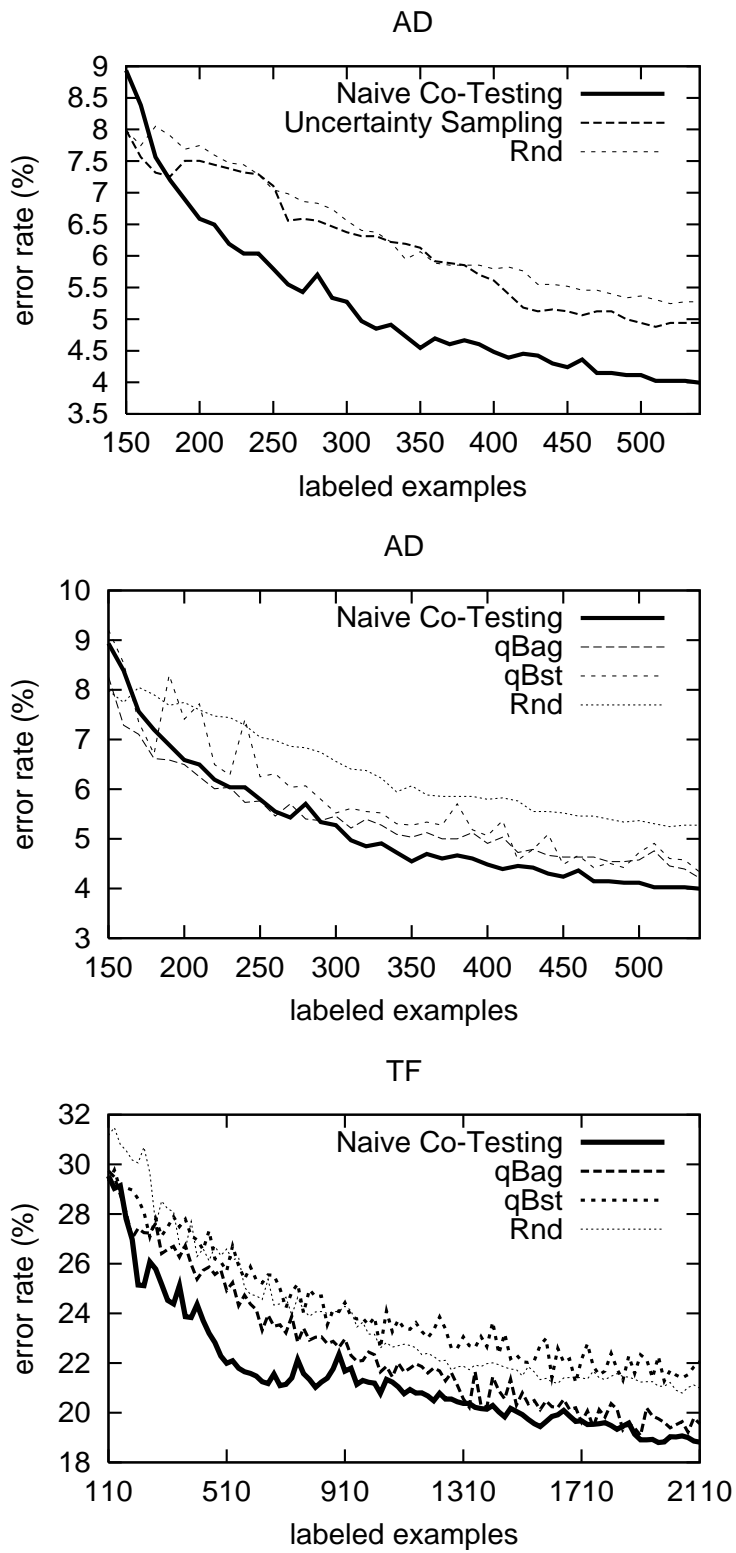


Figure 3.13: Empirical results on the AD and TF problems

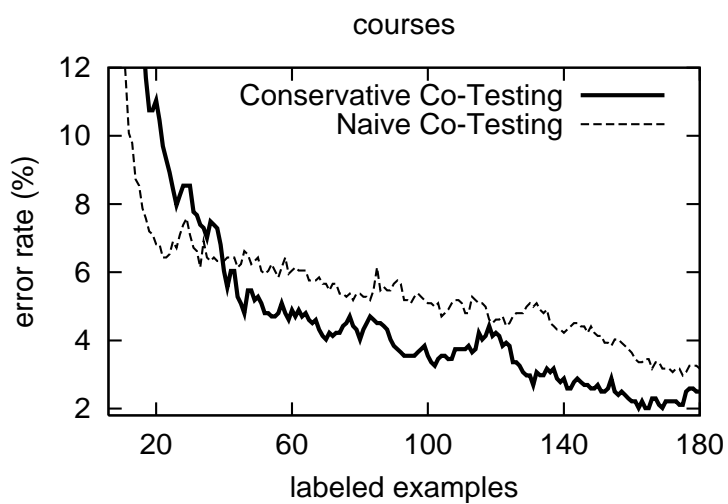
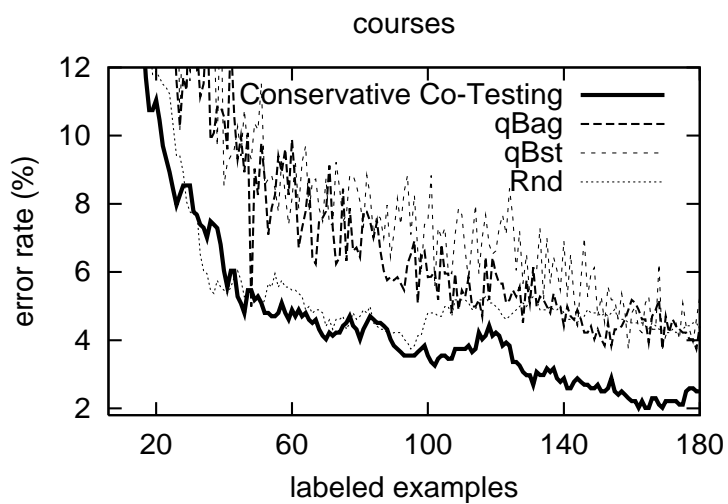
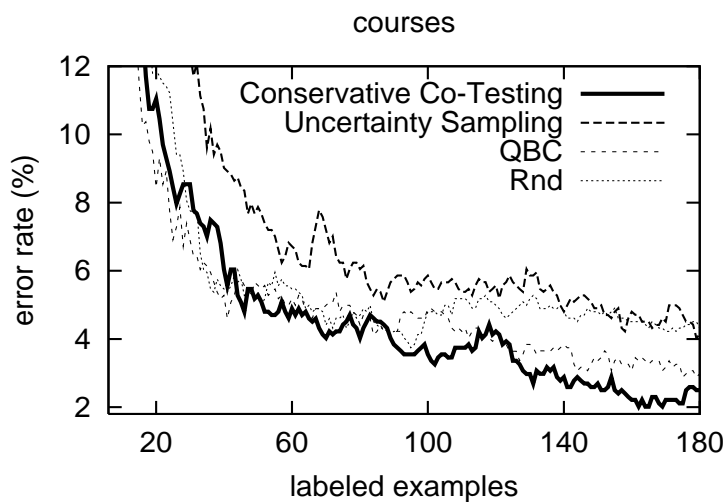


Figure 3.14: Empirical results on the COURSES problem

Mitchell use the two views to learn hypotheses that feed each other with the unlabeled examples on which their classification is the most confident.

My empirical results show that Co-Testing is a powerful approach to active learning. The experiments described above use four extremely different base learners (i.e., STALKER, IB, Naive Bayes, and MC4) on four different types of domains: wrapper induction, text classification (COURSES), ad removal (AD), and discourse tree parsing (TF). In all these scenarios, Co-Testing clearly outperforms the single-view, state of the art active learning algorithms. Furthermore, except for Query-by-Bagging, Co-Testing is the only algorithm that can be applied to all the problems considered in the empirical evaluation. In contrast to Query-by-Bagging, which has a poor performance on COURSES and wrapper induction, Co-Testing obtains the highest accuracy among the considered algorithms.

I conjecture that Co-Testing works so well because of its ability to discover the mistakes made in each view. As each contention point is mislabeled in at least one of the views, it follows that each query is extremely informative for the view that misclassified it. That is, mistakes are more informative than correctly labeled examples. This is particularly true for base learners such as STALKER and MC4, which do not improve the current hypothesis unless they are provided with examples of misclassified instances.

There are two main requirements for successfully applying Co-Testing. First, the views should be *sufficiently uncorrelated* to lead to a non-empty set of contention points. That is, even if the views are *not* independent given the label, as long as the hypotheses disagree on some unlabeled examples, Co-Testing can make new queries. Second, the views should be *sufficiently compatible* to allow learning from mistakes. In other words, as long as the the target concept can be learned with a high accuracy in each view, most Co-Testing queries will be informative because they represent mistakes that can be fixed.

As a limitation, Co-Testing can be applied only to multi-view tasks; that is, unless the user can provide two views, Co-Testing cannot be used at all. However, researchers have shown that besides the four problems above, multiple views exist in a variety of real world problems, such as named entity classification (Collins and Singer, 1999), statistical parsing

(Sarkar, 2001), speech recognition (de Sa and Ballard, 1998), Web page classification (Blum and Mitchell, 1998), word sense disambiguation (Yarowsky, 1995), or base noun phrase bracketing (Pierce and Cardie, 2001).

The other concern about Co-Testing is related to the potential violations of the two multi-view assumptions, namely that the views are both uncorrelated and compatible. For example, in case of correlated views, the hypotheses learned in each view may be so similar that there are no contention points among which to select the next query. In terms of view incompatibility, remember that in some of the wrapper induction tasks one of the views was so inaccurate that the Co-Testing could not outperform Random Sampling. In chapters 4 and 5, I empirically investigate both issues; I show that, in fact, Co-Testing compensates for view correlation, and I introduce a view validation algorithm that predicts whether or not the views are *sufficiently compatible* for performing multi-view learning on a new, unseen task.

3.7 Summary

In this section I introduced Co-Testing, a novel approach to active learning. Co-Testing, which is a multi-view algorithm, learns one hypothesis in each view and queries unlabeled examples on which the hypotheses predict a different label. Such queries are extremely informative because when two hypotheses disagree, at least one of them must be wrong, thus allowing Co-Testing to identify the mistakes made by the views. Co-Testing is a family of algorithms that differ from each other based on two criteria: the query selection strategy and output hypothesis.

The empirical evaluation shows that Co-Testing learns high-accuracy classifiers based on a small number of labeled examples. Co-Testing clearly outperforms single-view, state of the art sampling algorithms on a variety of domains. Furthermore, in contrast to single-view approaches, Co-Testing could be used with all four base learners considered in the experimental comparison.

Chapter 4

Active + Semi-Supervised = Robust Multi-View Learning

They are ill discoverers that think there is no land, when they can see nothing but sea.

Sir Francis Bacon

The theoretical foundation of multi-view learning (Blum and Mitchell, 1998) is based on the assumptions that the views are both *compatible* and *uncorrelated*. Intuitively, a problem has *compatible* views if all examples are labeled identically by the target concepts in each view. On the other hand, two views are *uncorrelated* if, given the label of any example, its descriptions in each view are independent. In real-world problems, both assumptions are likely to be violated for a variety of reasons such as correlated or insufficiently informative features.

In this chapter I study the robustness of several multi-view algorithms with respect to view *incompatibility* and *correlation*. As in practice it is difficult to measure these two factors, I use a parameterized family of text classification problems, PTCP, in which I control both view incompatibility and correlation. I first describe a motivating experiment that shows the lack of robustness of the existing multi-view algorithms. In order to cope with this problem, I introduce a new algorithm, Co-EMT, which outperforms the algorithms in the motivating experiment and has a robust behavior over the entire spectrum

of considered problems. This new algorithm interleaves active and semi-supervised learning; more precisely, it uses Co-Testing to select the labeled examples for the multi-view, semi-supervised Co-EM (Nigam and Ghani, 2000).

4.1 The Motivating Experiment

In this section I describe the experiment that motivates my work on Co-EMT. I empirically show that semi-supervised EM (Nigam and Ghani, 2000), Co-Training (Blum and Mitchell, 1998), and Co-EM (Nigam and Ghani, 2000) are extremely sensitive to view compatibility and correlation. This experiment also demonstrates that there are settings in which Co-Testing underperforms semi-supervised learning: if the number of labeled examples is extremely small, choosing them smartly (i.e., by active learning) is not as effective as supplementing them with a large number of unlabeled examples. Before describing the actual results, I provide a high-level description of the semi-supervised algorithms that are used in this experiment.

4.1.1 The Semi-supervised algorithms used in the comparison

4.1.1.1 The Co-Training algorithm

Co-Training (Blum and Mitchell, 1998) is a semi-supervised, multi-view algorithm that uses the initial training set to learn a (weak) classifier in each view. Then each classifier is applied to all unlabeled examples, and Co-Training detects the examples on which each classifier makes the most confident predictions. These high-confidence examples are labeled with the estimated class labels and added to the training set (see Figure 4.1). Based on the new training set, a new classifier is learned in each view, and the whole process is repeated for several iterations. At the end, a final hypothesis is created by a voting scheme that combines the prediction of the classifiers learned in each view.

- Given:**
- a base learner \mathcal{L}
 - a learning problem with two views $\mathbf{V1}$ and $\mathbf{V2}$
 - the sets L and U of labeled and unlabeled examples
 - the number k of iterations to be performed

Co-Training:

- for each class C_i , let n_i be the number of examples to be labeled after each iteration
- LOOP for k iterations
 - use \mathcal{L} , $\mathbf{V1}(L)$, and $\mathbf{V2}(L)$ to create classifiers h_1 and h_2
 - FOR EACH class C_i DO
 - let PL_1 be the n_i examples in U on which h_1 makes the most confident predictions for C_i
 - let PL_2 be the n_i examples in U on which h_2 makes the most confident predictions for C_i
 - $U = U - PL_1 - PL_2$
 - $L = L \cup \{ \langle u, h_1(u) \rangle \mid u \in PL_1 \} \cup \{ \langle u, h_2(u) \rangle \mid u \in PL_2 \}$
 - combine the prediction of h_1 and h_2 (*weighted vote*)

Semi-supervised EM:

- let $All = L \cup U$
- let h be the classifier obtained by training \mathcal{L} on L
- LOOP for k iterations
 - $New = \text{ProbabilisticallyLabel}(All, h)$
 - $h = \mathcal{L}_{max-a-posteriori}(New)$

Co-EM:

- let $All = L \cup U$
- let h_1 be the classifier obtained by training \mathcal{L} on L
- LOOP for k iterations
 - $New_1 = \text{ProbabilisticallyLabel}(All, h_1)$
 - $h_2 = \mathcal{L}_{max-a-posteriori}(\mathbf{V2}(New_1))$
 - $New_2 = \text{ProbabilisticallyLabel}(All, h_2)$
 - $h_1 = \mathcal{L}_{max-a-posteriori}(\mathbf{V1}(New_2))$
 - combine the prediction of h_1 and h_2 (*weighted vote*)

Figure 4.1: A high-level description of Co-Training, Semi-supervised EM, and Co-EM.

4.1.1.2 The semi-supervised EM algorithm

Semi-supervised EM (Nigam and Ghani, 2000), which is the only single-view algorithm considered in this empirical comparison, is used as baseline because of its well-known ability to combine labeled and unlabeled data. As shown in Figure 4.1, semi-supervised EM (Nigam and Ghani, 2000) applies a probabilistic learning algorithm \mathcal{L} to a small set of labeled examples and a large set of unlabeled ones. First, semi-supervised EM creates an initial classifier h based solely on the labeled examples. Then it repeatedly performs a two-step procedure: first, use h to probabilistically label all unlabeled examples; then, learn a new *maximum a posteriori* (MAP) hypothesis h based on the examples labeled in the previous step. Intuitively, EM tries to find the most likely hypothesis that could generate the distribution of the unlabeled data. Semi-supervised EM can be seen as clustering the unlabeled data “around” the examples in the original training set.

4.1.1.3 The Co-EM algorithm

Co-EM (Nigam and Ghani, 2000; Ghani, 2002) is a semi-supervised, multi-view algorithm that uses the hypothesis learned in one view to probabilistically label the examples in the other one (see Figure 4.1). Intuitively, Co-EM runs EM in each view and, before each new EM iteration, inter-changes the probabilistic labels generated in each view.

Co-EM can be seen as a probabilistic version of Co-Training. In fact, both algorithms are based on the same underlying idea: they use the knowledge acquired in one view (i.e., the probable labels of the examples) to train the other view. The major difference between the two algorithms is that Co-EM does *not* commit to a label for the unlabeled examples; instead, it uses probabilistic labels that may change from one iteration to another.¹ By contrast, Co-Training’s commitment to the high-confidence predictions may add to the

¹In (Nigam and Ghani, 2000), Co-EM and Co-Training are contrasted as being *iterative* and *incremental*, respectively. This description is equivalent to the one above: Co-EM *iteratively* uses the unlabeled data because it *does not commit* to the labels from the previous iteration. By contrast, Co-Training *incrementally* uses the unlabeled data by *committing* to a few labels per iteration.

training set a large number of mislabeled examples, especially during the first iterations, when the hypotheses may have little predictive power.

4.1.2 The empirical results

In order to study the influence of view incompatibility and correlation to the performance of multi-view learners, I use the PTCP parameterized family of learning problems (Muslea, Minton, and Knoblock, 2002a). PTCP consists of 60 text classification tasks in which one can control the amount of view correlation and incompatibility. More precisely, PTCP contains learning problems with one, two, and four clumps per class² and 0, 10, 20, 30, and 40% view incompatibility (i.e., between 0% and 40% of the examples are incompatible). Each learning task in PTCP is a binary classification problem over an instance space of 10448 features.

In this empirical study, I apply semi-supervised EM, Co-Training, Co-EM, and Co-Testing to the tasks in PTCP. The three semi-supervised algorithms are trained based on 40 randomly chosen labeled examples and 600 unlabeled ones. In contrast, Co-Testing starts with 10 random examples, makes 30 queries (for a total of 40 labeled examples) and uses no unlabeled data for training.³

The results of my experiments are shown in Figure 4.2. Each of the three graphs displays the learning curves obtained on problems with a fixed number of clumps per class and various levels of domain incompatibility. The X axis shows the percentage of incompatible examples in the problems, while the Y axis represents the error rates. These results lead to two important observations.

First of all, when the labeled data is scarce and the instance space is high-dimensional (e.g., 40 labeled examples for learning with 10448 features), a large set of unlabeled

²As discussed in sections 2.3.1 and 3.3.1, a learning problem is said to have several clumps per class if each class consists of several distinct sub-classes. For example, in the COURSES problem, the *course homepage* class consists of the clumps *theory classes*, *AI classes*, and *systems classes*. Similarly, in the *faculty - non-faculty* problem from the previous chapter, the *faculty* class consists of the clumps *assistant*, *associate*, and *full professor*.

³To keep the presentation succinct, I discuss here only the information critical to making my case. The experimental framework and the complete results are presented in detail in section 4.3.1. The PTCP parameterized family of problems is described in detail in Appendix A.

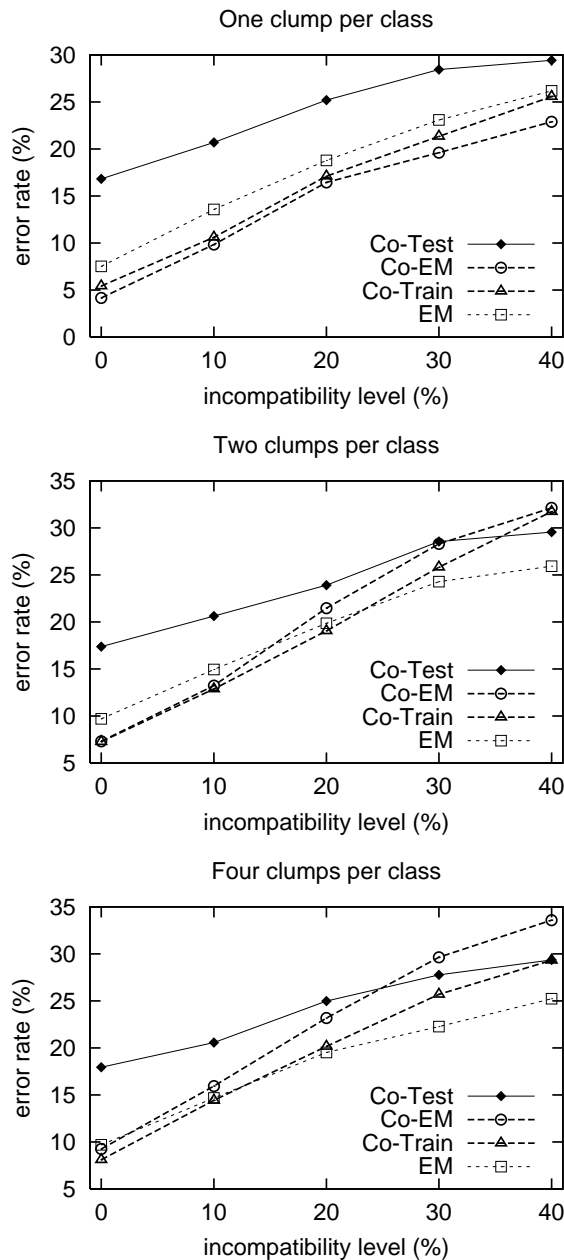


Figure 4.2: A controlled experiment with Co-Testing, Co-Training, Co-EM, and (semi-supervised) EM. The learning tasks from each graph have the same number of clumps per class (i.e., one, two, and four) and various levels of views incompatibility (i.e., 0%, 10%, 20%, 30%, and 40% of the examples are labeled differently in the two views). Given the scarcity of the labeled examples (40 examples) and the high-dimensionality of the learning tasks (10448 features), Co-Testing, which does not exploit the unlabeled examples, obtains the worst results. The other algorithms have a non-robust behavior: they perform well on some tasks, and poorly on the other ones.

examples can dramatically boost the classification accuracy. This is illustrated by the semi-supervised algorithms outperforming Co-Testing: even though Co-Testing selects and uses a more informative training set, its inability to exploit the unlabeled data when learning the hypotheses in each view undermines its performance.

Second, the three semi-supervised algorithms are extremely sensitive to view incompatibility and correlation. For example, Co-Training and Co-EM outperform semi-supervised EM on problems with uncorrelated views (i.e., one clump per class). In contrast, as the views become incompatible and correlated (i.e., four clumps per class and 30-40% incompatibility), the two multi-view algorithms underperform semi-supervised EM, with Co-EM doing clearly worse than Co-Training.

These results motivate the need for a new algorithm that has a robust behavior over the entire `correlation - incompatibility` space. In the next section, I introduce Co-EMT, which reaches this goal by combining active and semi-supervised learning (i.e., Co-Testing and Co-EM, respectively).

4.2 Co-Testing + Co-EM = Co-EMT

As shown in Figure 4.3, Co-EMT is a novel algorithm that combines the strengths of both active and semi-supervised learning by interleaving Co-EM and Co-Testing.⁴ As opposed to a typical Co-Testing algorithm, which learns h_1 and h_2 based solely on labeled examples, Co-EMT induces the two hypotheses by running Co-EM on both labeled and unlabeled examples. Depending on the type of Co-Testing used within Co-EMT (e.g., naive, conservative, or aggressive query selection strategy, combined with *winner-takes-all*, majority vote, or weighted vote for the output hypothesis), one can obtain a variety of Co-EMT algorithms.

⁴I have chosen to combine Co-Testing with Co-EM rather than Co-Training because of the difficulties encountered while fine-tuning the latter, which is sensitive to changes in the number of examples added after each iteration.

In order to put Co-EMT in a larger context, Figure 4.4 shows Co-EMT’s relationship with the other algorithms considered in this chapter. On one side, Co-EMT is a semi-supervised variant of Co-Testing, which - in turn - was inspired from Co-Training. On the other side, Co-EMT builds on Co-EM, which is a state-of-the art, semi-supervised algorithm that combines the basic ideas from Co-Training and EM.

Note that interleaving Co-EM and Co-Testing leads to an interesting synergy. On one hand, Co-Testing boosts the accuracy of Co-EM by selecting a highly informative set of labeled examples (stand-alone Co-EM chooses them at random). On the other hand, as the hypotheses learned by Co-EM are more accurate than the ones learned just from labeled data, compared with stand-alone Co-Testing, Co-EMT uses more accurate hypotheses to select the queries.

4.3 Empirical Evaluation

4.3.1 The Experimental Setup

In order to evaluate Co-EMT, I apply it on the PTCF parameterized family of text classification problems and compare its results with the ones obtained by semi-supervised EM, Co-Training, Co-EM, and Co-Testing. All five algorithms use as base learner a Naive Bayes classifier for text classification (Nigam and Ghani, 2000). As a Naive Bayes classifier can evaluate the confidence of its prediction, both for stand-alone Co-Testing and within Co-EMT, I use *Conservative Co-Testing* with *weighted vote* (i.e., the algorithm that was also used in the previous chapter for the COURSES problem).

The accuracy of the five algorithms is estimated based on four runs of 5-fold cross-validation; consequently, each training and test set consist of 640 and 160 examples, respectively. For the three semi-supervised algorithms, the 640 training examples are split randomly into two groups: 40 of them are used as labeled examples, while the remaining 600 are unlabeled (i.e., I hide their labels). To keep the comparison fair, Co-EMT and Co-Testing start with 10 randomly chosen labeled examples and query 30 of the

- Given:**
- a base learner \mathcal{L}
 - a learning problem with two views $\mathbf{V1}$ and $\mathbf{V2}$
 - the sets L and U of labeled and unlabeled examples
 - number N of queries to be made

Co-Testing:

- LOOP for N iterations
- use \mathcal{L} , $\mathbf{V1}(L)$, and $\mathbf{V2}(L)$ to create classifiers h_1 and h_2
 - let $ContentionPoints = \{ x \in U, h_1(x) \neq h_2(x) \}$
 - let $x = \mathbf{SelectQuery}(ContentionPoints)$
 - remove x from U , ask for its label l
 - ad $\langle x, l \rangle$ to L
 - **CreateOutputHypothesis**(h_1, h_2)

Co-EMT:

- let $iters$ be the number of Co-EM iterations within Co-EMT
- REPEAT N times
 - run **Co-EM**(\mathcal{L} , $\mathbf{V1}$, $\mathbf{V2}$, L , U , $iters$) to learn h_1 and h_2
 - let $ContentionPoints = \{ x \in U, h_1(x) \neq h_2(x) \}$
 - let $x = \mathbf{SelectQuery}(ContentionPoints)$
 - remove x from U , ask for its label l
 - ad $\langle x, l \rangle$ to L
 - **CreateOutputHypothesis**(h_1, h_2)

Figure 4.3: The Co-Testing and Co-EMT algorithms differ from each other only with respect to the technique used to learn the hypotheses in each view. Co-Testing uses the supervised base learner \mathcal{L} , while Co-EMT uses the semi-supervised Co-EM algorithm.

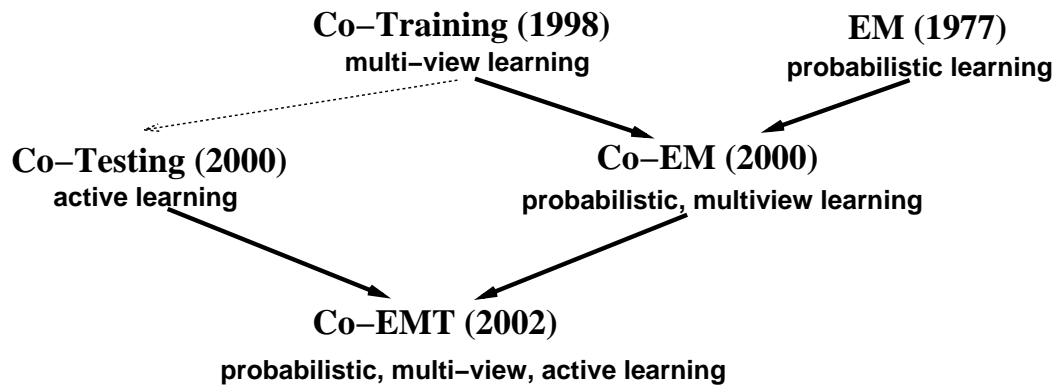


Figure 4.4: The lineage of the Co-EMT algorithm. The dotted line from Co-Training to Co-Testing denotes a change in learning paradigm, from semi-supervised to active learning.

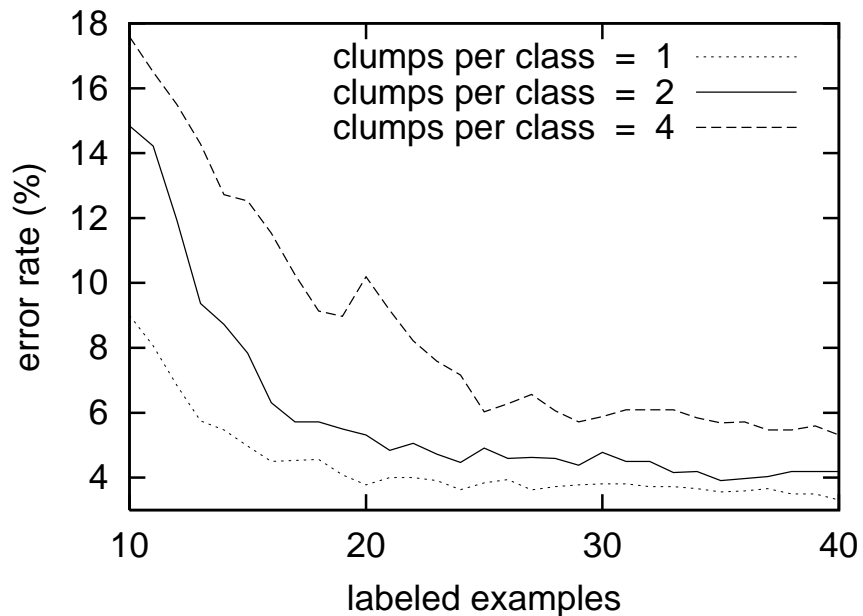


Figure 4.5: Illustrative learning curves for Co-EMT on tasks with no incompatibility and 1, 2, and 4 clumps per class. Co-EMT starts with 10 randomly-chosen labeled examples and makes 30 queries, for a total of 40 labeled examples.

630 unlabeled ones, for a total of 40 labeled examples. Figure 4.5 depicts three illustrative learning curves that show how Co-EMT’s error rate decreases as the algorithm makes a larger fraction of the allowed 30 queries.

For semi-supervised EM, Co-Training, and Co-EM, I have implemented the versions described in (Nigam and Ghani, 2000). EM and Co-EM are run for seven and five iterations, respectively. Co-Training, which requires significant fine tuning, labels 40 examples after each of the seven iterations. To avoid prohibitive running time, within Co-EMT, I perform only two Co-EM iterations after each Co-Testing query (on each of the 60 problems, Co-EMT runs Co-EM after each of the 600 queries: 4 runs \times 5 folds \times 30 queries per fold).

As already mentioned, PTCP covers 15 points in the **correlation - incompatibility** space (i.e., *three* level of clumpiness and *five* levels of view incompatibility); for each of these 15 points in the **correlation - incompatibility** space, PTCP contains four text classification problems, for a total of 60 problems (see Appendix **A** for details). At each point in the **correlation - incompatibility** space, the reported error rate is averaged over four corresponding text classification problems.

Figure 4.6 shows the performance of Co-EMT, Co-Testing, Co-EM, Co-Training, and EM on the parameterized family of problems. The five graphs correspond to the five levels of views incompatibility: 0%, 10%, 20%, 30%, and 40%. In each graph, the X and Y axes show the number of clumps per class and the error rate, respectively.

On **all** 15 points in the **correlation-incompatibility** space, Co-EMT obtains the lowest error rates. In a pairwise comparison with Co-Testing, Co-Training, Co-EM, and EM, the results are statistically significant with 95% confidence on 15, 13, 10, and 12 of the points, respectively. The remaining points represent extreme situations: for Co-Training and Co-EM, conditional independent views (one clump per class); for EM highly correlated and incompatible views (four clumps per class, and 20%, 30%, 40% incompatibility).

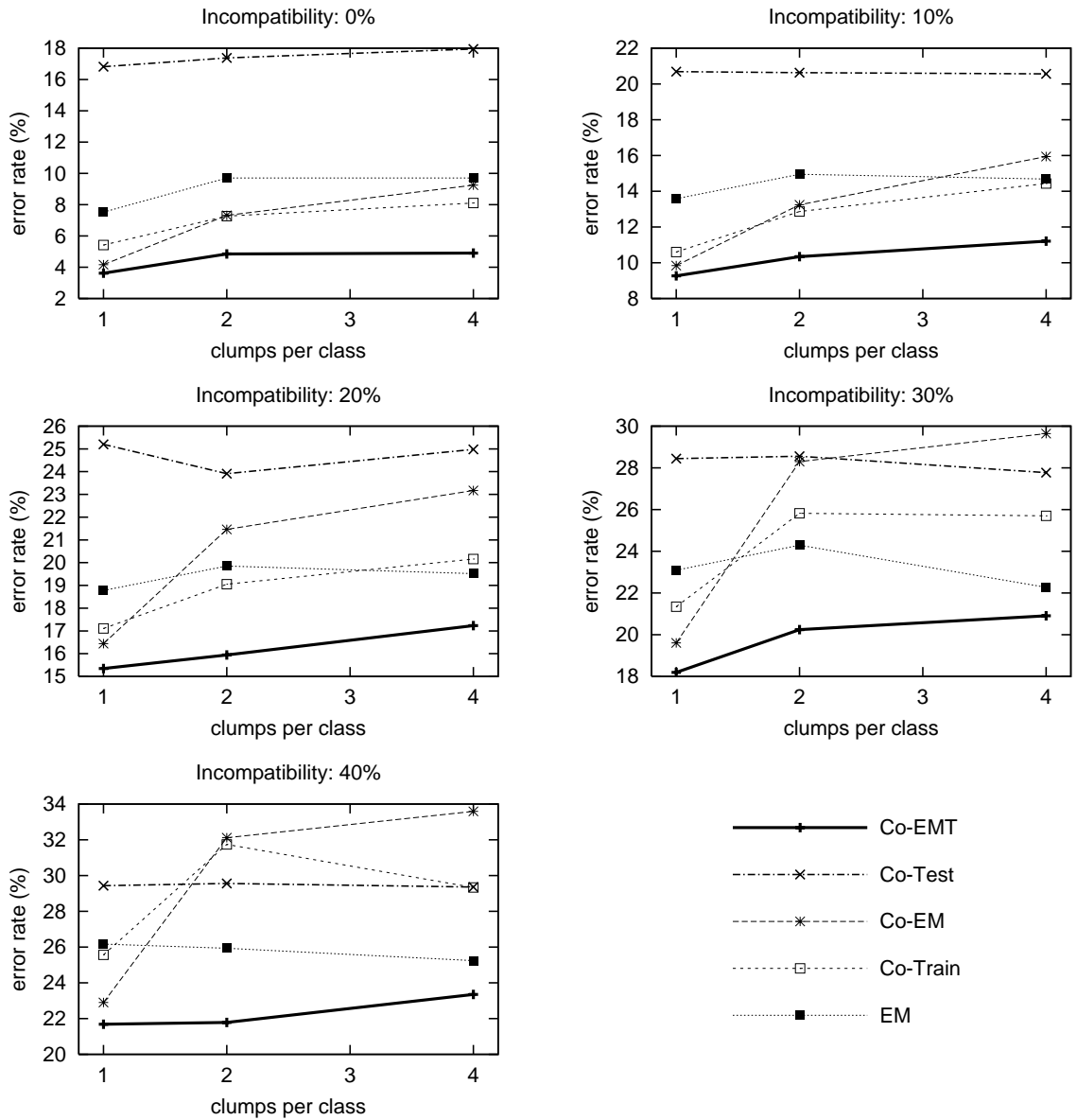


Figure 4.6: Empirical results on the PTCT family of problems. Each graph shows the results for learning tasks of a fixed level of view incompatibility and various degrees of domain clumpiness.

4.3.2 Discussion

These empirical results deserve several comments. First, Co-EMT, which combines Co-Testing and Co-EM, clearly outperforms both its components. Intuitively, Co-EMT’s power comes from Co-Testing and Co-EM compensating for each other’s weaknesses. On one hand, by exploiting the unlabeled data, Co-EM boosts the accuracy of the classifiers learned by Co-Testing. On the other hand, Co-Testing improves Co-EM’s accuracy by providing a highly informative set of labeled examples.

Co-EMT is not the first algorithm that combines semi-supervised and active learning: in (McCallum and Nigam, 1998b), various combinations of semi-supervised EM and Query-by-Committee (QBC) are shown to outperform both EM and QBC.⁵ I expect that using other active learning algorithms to select the labeled examples for Co-EM, Co-Training, and EM would also improve their accuracy. Finding the best combination of active and semi-supervised learning is beyond the scope of this chapter. My main contribution here is to show that interleaving active and semi-supervised learning leads to a robust performance over the entire spectrum of problems.

Second, Co-EM and Co-Training are highly sensitive to domain clumpiness. On problems with uncorrelated views (i.e., one clump per class), Co-EM and Co-Training clearly outperform EM. In fact, Co-EM is so accurate that Co-EMT can barely outperform it. This behavior is consistent with theoretical argument in (Blum and Mitchell, 1998): given two uncorrelated views, even in the presence of view incompatibility, a concept can be learned based on a few labeled and many unlabeled examples.

In contrast, on problems with four clumps per class, EM clearly outperforms both Co-EM and Co-Training. The two multi-view algorithms perform poorly on clumpy domains because rather than being disseminated over the entire instance space, the information

⁵The best of these EM and QBC combinations is not appropriate for multi-view problems because it uses a sophisticated heuristic that estimates the density of various regions in the *single-view* instance space (the density of a multi-view instance space is a function of the “local” densities within each view). Instead, I have implemented another (single-view) algorithm from (McCallum and Nigam, 1998b), which, similarly to Co-Testing, interleaves QBC and EM. As this algorithm barely improved EM’s accuracy on the parameterized problems, I decided not to show the corresponding learning curves on the already crowded Figure 4.6.

exchanged between the views remains localized within each clump. The fact that Co-EMT is almost insensitive to clumpiness suggests that Co-Testing compensates for domain clumpiness.⁶

Third, the performance of all algorithms degrades as the views become less compatible. The multi-view algorithms are sensitive to view incompatibility because the information exchanged between views becomes misleading as more examples are labeled differently in the two views. In order to cope with this problem, in the next chapter I introduce a *view validation* technique (Muslea, Minton, and Knoblock, 2002b) that detects whether or not two views are “sufficiently compatible” for multi-view learning.

Note that, at first glance, Co-EMT should perform poorly on problems with highly incompatible views: on such domains, it looks likely that Co-EMT will query incompatible examples, which convey little information and are misleading for Co-EM. To understand how Co-EMT avoids making such queries, let us reconsider the illustrative COURSES domain in Figure 4.7, which was introduced in chapter 2 (Figure 2.4) and is repeated here for convenience.

Remember that two hyperlinks containing either the *same text* (“Neural Nets”) or *similar* fragments of text (e.g., “Artificial Neural Nets” and “Artificial Neural Networks”) can point to Web pages having *different labels*. Because of the ambiguity of such examples, the hypotheses learned in the “hyperlink view” have a *low confidence* in predicting their labels. As Co-EMT uses the *conservative* query selection strategy (i.e., it queries contention points on which the views make equally confident predictions), it follows that an incompatible example is queried only if the other view also has an equally low confidence on its prediction.

In summary, I expect Co-EMT to perform well on most domains. The areas of the **correlation - incompatibility** space in which it does not clearly outperform all other four algorithms have either uncorrelated views (one clump per class) or correlated, incompatible views (four clumps per class, 30%-40% incompatibility). On the former it

⁶Remember that Co-EMT is simply Co-EM using labeled examples chosen via Co-Testing queries.

View V1
(words in hyperlinks)

View V2
(words in Web pages)

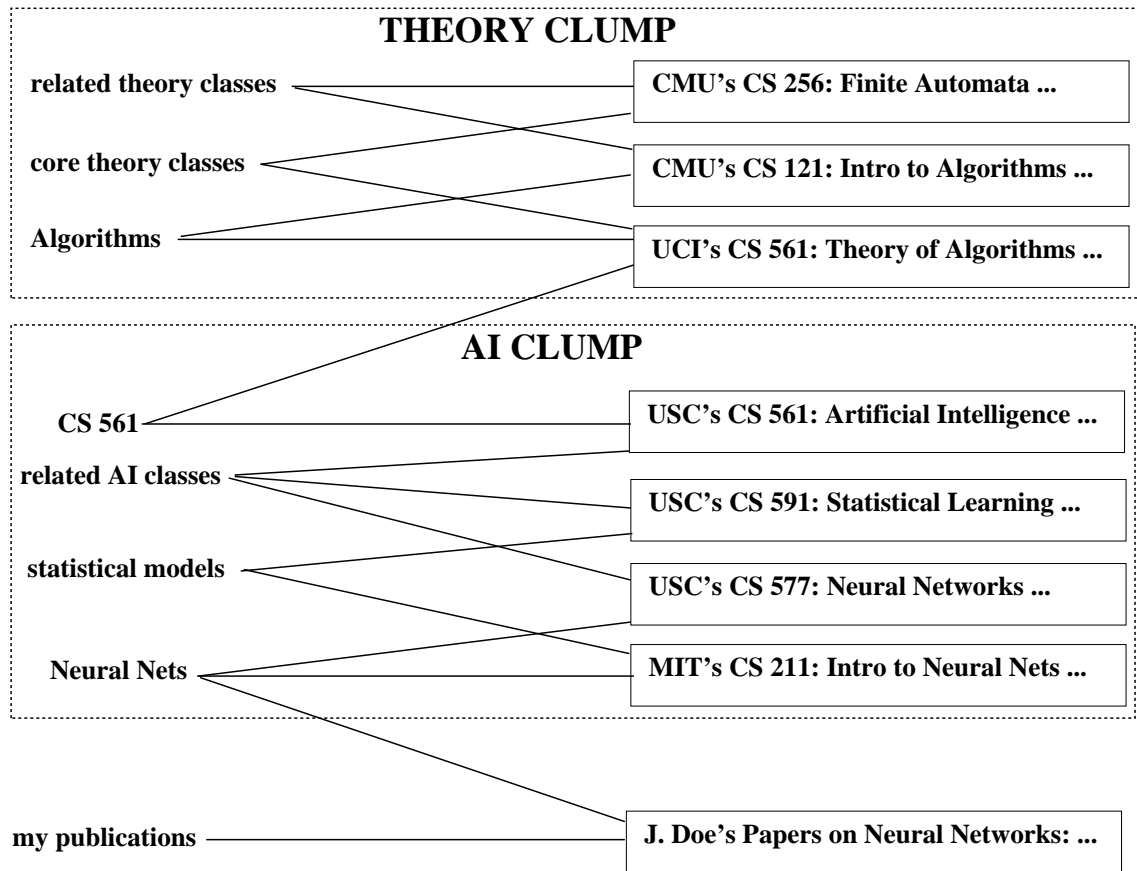


Figure 4.7: Two illustrative clumps in the COURSES domain.

barely outperforms Co-EM, but such domains are unlikely to occur in practice. On the latter it barely outperforms EM, and one may expect EM to outperform Co-EMT at higher incompatibility levels. To cope with this problem, in the next chapter I introduce the *view validation* algorithm (Muslea, Minton, and Knoblock, 2002b), which predicts whether or not two views are sufficiently compatible for multi-view learning.

4.3.3 Results on real-world problems

I believe that most real-world, multi-view problems display some (low) level of view incompatibility and correlation. As this is the spectrum of problems where Co-EMT most clearly outperforms the other four algorithms, I conjecture that Co-EMT has the best potential of all algorithms discussed here.

In support of this conjecture, I present an additional experiment in which I apply the five algorithms above to two of the three real-world domains used in section 3.5.2: COURSES and AD. The third domain, TF, could not be used in this experiment because it is *not* a text classification problem: for Co-EMT, Co-EM, and EM, the experiments above use implementations specific for text classification.

As COURSES is a text classification problem, I use the same two views as before: words that appear in the pages and in the hyperlinks pointing to them, respectively. In contrast, for the AD task I redefine its views as two text classification problems. Originally (see section 3.5.2), the view **V1** consisted of all textual features, while **V2** described the geometric properties of the image (e.g., length, width, aspect ratio). In this experiment, I ignore the features in the “geometric view” **V2** and define two new views as follows. View **V1** describes the image itself (e.g., words in the image’s URL and caption), while view **V2** characterizes related pages (e.g., words from the URLs to the pages that contain the image or are pointed-at by the image). As all features in AD are *boolean* (i.e., presence/absence of word in document, rather than word counts), I use Naive Bayes with the multi-variate Bernoulli model (McCallum and Nigam, 1998a).

Algorithm	COURSES	AD
Co-EMT	3.98 ± 0.6	5.75 ± 0.4
Conservative Co-Testing	4.80 ± 0.5	7.70 ± 0.4
Co-EM	5.08 ± 0.7	7.80 ± 0.4
EM	5.32 ± 0.6	8.55 ± 0.4
Co-Training	5.18 ± 0.6	7.54 ± 0.4

Table 4.1: Error rates on two additional real world problems.

For both domains I perform two runs of 5-fold cross validation. On COURSES, the Co-EM, Co-Training, and EM use 65 labeled examples, while Co-EMT and Co-Testing start with 10 labeled examples and make 55 queries. For AD, the semi-supervised algorithms use 100 labeled examples, while Co-EMT and Co-Testing start with 60 labeled examples and make 40 queries. EM, Co-EM and Co-Training are run for seven, five and four iterations, respectively (Co-Training adds 100 examples after each iteration). Finally, within Co-EMT, I perform two Co-EM iterations after each Co-Testing query.

Table 4.1 shows that Co-EMT again obtains the best accuracy of the five algorithms. Co-EMT outperforms the other four algorithms on seven of the eight the pair-wise comparisons (results are statistically significant with at least 95% confidence). The only result that is not statistically significant consists of Co-EMT outperforming Conservative Co-Testing on the COURSES domain.

4.4 Summary

In this chapter I used a family of parameterized problems to analyze the influence of view correlation and incompatibility on the performance of several multi-view algorithms. I have shown that existing algorithms are not robust over the whole `correlation - incompatibility` space. To cope with this problem, I introduced a new multi-view algorithm, Co-EMT, which interleaves active and semi-supervised learning. I have shown

that Co-EMT clearly outperforms the other algorithms both on the parameterized problems and on two additional real world domains. My experimental results suggest that the robustness of Co-EMT comes from active learning compensating for the view correlation.

Chapter 5

View Validation

For the things we have to learn before we can do them, we learn by doing them.

Aristotle

This chapter concludes my study on the influence of view correlation and incompatibility on multi-view learning. In the previous chapter I have shown that Co-Testing compensates for domain clumpiness, but cannot do the same for view incompatibility. Consequently, I focus here on the view incompatibility issue, which is closely related to the accuracy of the hypotheses learned in the two views: the more accurate the views, the fewer examples can be incompatible (i.e., labeled differently in the two views). Figure 5.1 illustrates the relationship between the incompatibility of the views and the applicability of the multi-view algorithms: as the difference between the accuracy of the hypotheses learned in the two views increases (i.e., the views become more incompatible), the single-view algorithm outperforms its multi-view counterpart. This observation immediately raises the following question: for a new, unseen learning task, should one use a multi-view or a single-view learning algorithm?

The question above can be restated as follows: given two views and a set of learning tasks, how can one identify the tasks for which these two views are *sufficiently compatible* for multi-view learning? In order to answer this question, I introduce a *view validation algorithm* that, for a given pair of views, discriminates between the tasks for which the

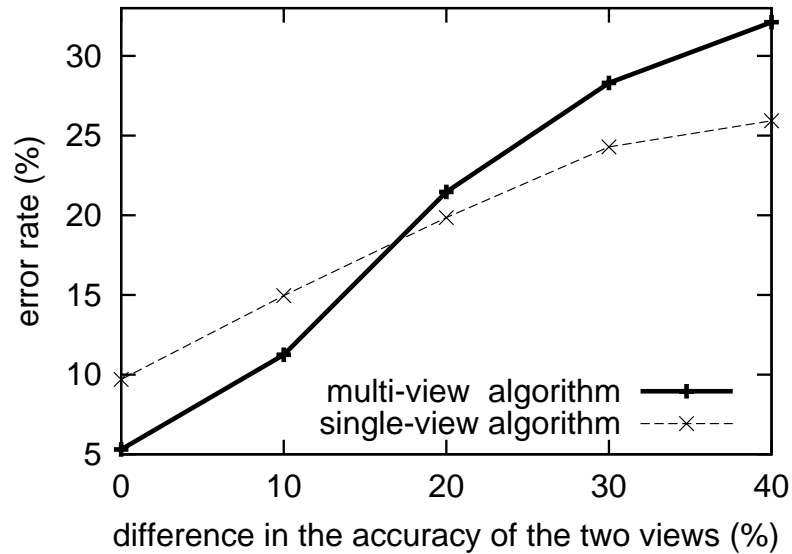


Figure 5.1: As the difference in the accuracy of the two views increases, the views become more incompatible, and the single-view algorithm outperforms its multi-view counterpart.

views are *sufficiently* and *insufficiently compatible* for multi-view learning. In other words, view validation judges the usefulness of the views for a particular learning task (i.e., *it validates the views for a task of interest*).

View validation is suitable for applications such as wrapper induction (Muslea, Minton, and Knoblock, 2000b) and Web page classification (Blum and Mitchell, 1998), where the same views are repeatedly used to solve a variety of unrelated learning tasks. Consider, for instance, the Web page classification problem, in which the two views consist of “words that appear in Web pages” and “words in hyperlinks pointing to them”. Note that, in principle, one can use these two views in learning tasks as diverse as distinguishing between homepages of *professors* and *students* or distinguishing between articles on *economics* and *terrorism*. However, for any of these learning tasks, it may happen that the text in the hyperlinks is so short and uninformative that one is better off using just the words in the Web pages. To cope with this problem, one can use view validation to predict whether or not multi-view learning is appropriate for a task of interest.

This chapter presents a general, meta-learning approach to view validation. In this framework, the user provides several exemplar learning tasks that were solved using the *same views*. For each solved learning task, my algorithm generates a *view validation example* by analyzing the hypotheses learned in each view. Then it uses the C4.5 algorithm (Quinlan, 1993) to identify common patterns that discriminate between the learning tasks for which the views are *sufficiently* and *insufficiently compatible* for multi-view learning. An illustrative example of such a pattern is the following: “**IF** for a task T the difference in the training errors in the two views is larger than 20% *and* the views agree on less than 45% of the unlabeled examples **THEN** the views are *insufficiently compatible* for applying multi-view learning to T .” I consider two application domains: text classification and wrapper induction (a commercially important multi-view problem). On both domains, the view validation algorithm makes high accuracy predictions based on a modest amount of training data.

View validation represents a first step towards my long-term goal of automatic *view detection*, which would dramatically widen the practical applicability of multi-view algorithms. Instead of having to rely on user-provided views, one can use view detection to search for adequate views among the possible partitions of the domain’s features. In this context, a view validation algorithm becomes a key component that verifies whether or not the views that are generated during view detection are sufficiently compatible for applying multi-view learning to a learning task.

5.1 The View Validation Algorithm

Before introducing view validation, let me briefly present the terminology used in this chapter. By definition, a *multi-view problem* is a collection of learning tasks that use the same views; each such learning task is called an *instance of the multi-view problem* or a *problem instance*. For example, consider again multi-view problem that consists of all Web page classification tasks in which the views are “words in Web pages” and “words in hyperlinks pointing to the pages.” One can use these two views to learn a classifier

that distinguishes between homepages of *professors* and *students*, another classifier that distinguishes between articles on *gun control* and *terrorism*, and so forth. Consequently, all these learning tasks represent instances of the same multi-view problem.

Note that, in practice, one cannot expect a pair of views to be sufficiently compatible for all learning tasks. For instance, in the problem above, one may encounter tasks in which the text in the hyperlinks is too short and uninformative for the text classification task. More generally, because of corrupted or insufficient features, it is unrealistic to expect the views to be *sufficiently compatible* for applying multi-view learning to all problem instances. One way to address this problem is to use a *view validation* algorithm, which, for any problem instance, predicts whether or not the views are sufficiently compatible for using multi-view learning for that particular task.

In practice, the level of “acceptable” view incompatibility depends on both the domain features and the base learner \mathcal{L} that is used to learn the hypotheses in each view. Consequently, I apply view validation to a given multi-view problem (i.e., pair of views) and learning algorithm \mathcal{L} . Note that this is a natural scenario for multi-view problems such as text classification and wrapper induction, in which the same views are used for a wide variety of learning tasks.

The view validation algorithm (see Figure 5.2) implements a three-step process. First, the user provides several pairs $\langle I_k, l_k \rangle$, where I_k is a problem instance, and l_k is a label that specifies whether or not the views are sufficiently compatible for using multi-view learning to solve I_k . The label l_k is generated automatically by comparing the accuracy of a single- and multi-view algorithm on a test set. Second, for each instance I_k , I generate a *view validation example* (i.e., a feature-vector) that describes the properties of the hypotheses learned in the two views. Finally, I apply C4.5 to the view validation examples, and I use the learned decision tree to discriminate between learning tasks for which the views are sufficiently or insufficiently compatible for multi-view learning,

In keeping with the multi-view setting, I assume that for each instance I_k the user provides a (small) set L_k of labeled examples and a (large) set U_k of unlabeled examples.

Given:

- a multi-view problem P with views V_1 and V_2
- a learning algorithm \mathcal{L}
- a set of pairs $\{ \langle I_1, L_1 \rangle, \langle I_2, L_2 \rangle, \dots, \langle I_n, L_n \rangle \}$, where I_k are instances of P , and l_k labels I_k as having or not views that are *sufficiently compatible* for multi-view learning

FOR each instance I_k DO

- let L_k and U_k be labeled and unlabeled examples in I_k
 - use \mathcal{L} , $V_1(L_k)$, and $V_2(L_k)$ to learn classifiers h_1 and h_2
 - $CreateViewValidationExample(h_1, h_2, L_k, U_k, l_k)$
- train C4.5 on the view validation examples
 - use the learned classifier to discriminate between problem instances for which the views are *sufficiently* and *insufficiently compatible* for multi-view learning

Figure 5.2: The View Validation Algorithm.

For each instance I_k , I use the labeled examples in L_k to learn a hypothesis in each view (i.e., h_1 and h_2). Then I generate a *view validation example* that is labeled l_k and consists of a feature-vector that describes the hypotheses h_1 and h_2 . In the next section, I present the actual features used for view validation.

5.2 Features Used for View Validation

Ideally, besides the label l_k , a *view validation example* would consist of a single feature: the percentage of examples that are labeled differently in the two views. Based on this unique feature, one could learn a *threshold value* that discriminates between the problem instances for which the views are sufficiently/insufficiently compatible for multi-view learning. In Figure 5.1, this threshold corresponds to the point in which the two learning curves intersect. In practice, using this unique feature requires knowing the labels of *all* examples in a domain. As this is an unrealistic scenario, I have chosen instead to use several features that are potential indicators of the how incompatible the views are.

Each view validation example is described by the following seven features:

- f_1 : the percentage of unlabeled examples in U_k that are classified identically by h_1 and h_2 ;
- f_2 : $\min(\text{TrainingErrors}(h_1), \text{TrainingErrors}(h_2))$;
- f_3 : $\max(\text{TrainingErrors}(h_1), \text{TrainingErrors}(h_2))$;
- f_4 : $f_3 - f_2$;
- f_5 : $\min(\text{Complexity}(h_1), \text{Complexity}(h_2))$;
- f_6 : $\max(\text{Complexity}(h_1), \text{Complexity}(h_2))$;
- f_7 : $f_6 - f_5$.

Note that features f_1 - f_4 are measured in a straightforward manner, regardless of the algorithm \mathcal{L} used to learn h_1 and h_2 . More precisely, f_1 can be computed based on its definition. For measuring f_2 , f_3 , and f_4 , one must first count the number of (labeled) training examples that are mislabeled by h_1 and h_2 (i.e., the training errors) and then apply the formulas above.

By contrast, features f_5 - f_7 dependent on the representation used to describe these two hypotheses. For instance, the complexity of a boolean formula may be expressed in terms of the number of disjuncts and literals in the disjunctive or conjunctive normal form; or, for a decision tree, the complexity measure may take into account the depth and the breadth (i.e., number of leaves) of the tree.

The intuition behind the seven view validation features is the following:

- the fewer unlabeled examples from U_k are labeled identically by h_1 and h_2 , the larger the number of potentially incompatible examples;
- the larger the difference in the training error of h_1 and h_2 , the less likely it is that the views are equally accurate;

- the larger the difference in the complexity of h_1 and h_2 , the likelier it is that the most complex of the two hypotheses overfits the (small) training set L_k . In turn, this may indicate that the corresponding view is significantly less accurate than the other one.

In practice, features f_1 - f_4 are measured in a straightforward manner; consequently, they can be always used in the view validation process. In contrast, measuring the complexity of a hypothesis may not be always possible or meaningful (consider, for instance, the case of a k nearest-neighbor classifier). In such situations, one can simply ignore features f_5 - f_7 and rely on the remaining features.

5.3 Empirical Results

5.3.1 The multi-view test problems

I evaluate the view validation algorithm on two multi-view problems: *wrapper induction* (WI), which consists of the 33 extraction tasks described in section 3.5.1.2, and the PTCT family of 60 parameterized text classification tasks used in chapter 4.¹

For wrapper induction, the base learner is STALKER. Consequently, the view validation features are measured as follows: f_1 represents that percentage of (unlabeled) documents from which the two extraction rules extract the same string; for f_2 - f_4 , I count the labeled documents from which the extraction rules do not extract the correct string. Finally, to measure f_5 - f_7 , I define the complexity of an extraction rule as the maximum number of disjuncts that appear in either the start or the end rule.

For PTCT, I use as base learner the Naive Bayes algorithm (Nigam and Ghani, 2000). As there is no obvious way to measure the complexity of a Naive Bayes classifier, for PTCT I do not use the features f_5 - f_7 . The other features are measured in a straightforward manner: f_1 represents the percentage of unlabeled examples on which the two Naive

¹I would have preferred to use a second real-world multi-view problem instead of PTCT. Unfortunately, given that multi-view learning represents a relatively new field of study, most multi-view algorithms were applied to just a couple problem instances.

Bayes classifiers agree, while f_2 - f_4 are obtained by counting the training errors in the two views.

5.3.2 Generating the WI and PTCT Datasets

To label the 33 problem instances for wrapper induction (WI), I compare the performance of Naive Co-Testing with stand-alone STALKER (i.e., STALKER learning **FF** rules based on randomly chosen labeled examples). On the six extraction tasks in which the difference in the accuracy of the rules learned in the two views is larger than 10%, single-view STALKER does at least as well as its multi-view counterpart. I label these six problem instances as having views that are insufficiently compatible for multi-view learning.

In order to label the 60 instances in PTCT, I compare single-view, semi-supervised EM with Co-Training, which is the most widely used semi-supervised multi-view algorithm (Collins and Singer, 1999; Pierce and Cardie, 2001; Sarkar, 2001). I use the empirical results from the previous chapter to identify the instances on which semi-supervised EM performs at least as well as Co-Training. I label the 40 such instances as having views that are insufficiently compatible for multi-view learning.

For both WI and PTCT, I have chosen the number of examples in L_k (i.e., $Size(L_k)$) according to the experimental setups described in (Muslea, Minton, and Knoblock, 2001) and (Muslea, Minton, and Knoblock, 2002a), in which WI and PTCT were introduced. For WI, in which an instance I_k may have between 91 and 690 examples, $Size(L_k)=6$ and U_k consists of the remaining examples. For PTCT, where each instance consists of 800 examples, the size of L_k and U_k is 70 and 730, respectively.

5.3.3 The Setup

In contrast to the approach described in Figure 5.2, where a *single* view validation example is generated per problem instance, in the experiments I create *several* view validation examples per instance. That is, for each instance I_k , I generate $ExsPerInst = 20$ view validation examples by repeatedly partitioning the examples in I_k into randomly chosen

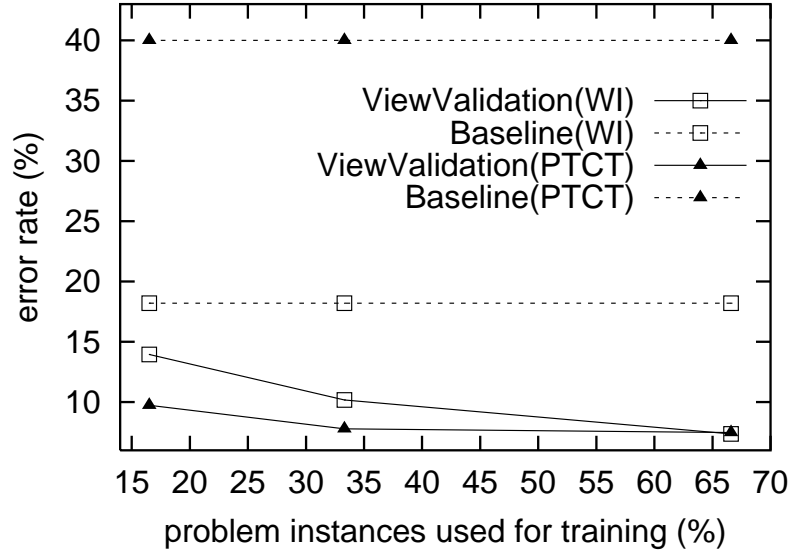


Figure 5.3: View validation clearly outperforms the baseline algorithm.

sets L_k and U_k of the appropriate sizes. The motivation for this decision is two-fold. First, the empirical results should not reflect a particularly (un)fortunate choice of the sets L_k and U_k . Second, if I generate a single view validation example per instance, for both WI and PTCT I obtain a number of view validation examples that is too small for a rigorous empirical evaluation (i.e., 33 and 60, respectively). To conclude, by generating $ExsPerInst = 20$ view validation examples per problem instance, I obtain larger number of view validation examples (660 and 1200, respectively) that, for each problem instance I_k , are representative for a wide variety of possible sets L_k and U_k .

To evaluate view validation’s performance, for both WI and PTCT, I partition the problem instances into *training* and *test instances*. For each such partition, I create the *training* and *test sets* for C4.5 as follows: all $ExsPerInst = 20$ view validation examples that were created for a *training instance* are used in the C4.5 *training set*; similarly, all 20 view validation examples that were created for a *test instance* are used in the C4.5 *test set*. In other words, all view validation examples that are created based on the same problem instance belong either to the training set or to the test set, and they cannot be

split between the two sets. In the experiments, I train on $\frac{1}{6}$, $\frac{1}{3}$, and $\frac{2}{3}$ of the instances and test on the remaining ones. For each of these three ratios, I average the error rates obtained over $N = 20$ random partitions of the instances into training and test instances.

Figure 5.3 shows the view validation results for the WI and PTCT datasets. The empirical results are excellent: when trained on 66% of the available instances, the view validation algorithm reaches an accuracy of 92% on both the WI and PTCT datasets. Furthermore, even when trained on just 33% of the instances (i.e., 11 and 20 instances for WI and PTCT, respectively), view validation still obtains a 90% accuracy. Last but not least, for both WI and PTCT, view validation clearly outperforms a baseline algorithm that simply predicts the most frequent label in the corresponding dataset.

5.3.4 The Influence of *ExsPerInst* and *Size(L_k)*

The results in Figure 5.3 raise an interesting practical question: how much can I reduce the user’s effort without harming the performance of view validation? In other words, can one label only a fraction of the *ExsPerInst* view validation examples per problem instance and a subset of L_k , and still obtain a high-accuracy prediction? To answer this question, I designed two additional experiments in which I vary one of the parameters at a time.

To study the influence of the *ExsPerInst* parameter, I keep *Size(L_k)* constant (i.e., 6 and 70 for WI and PTCT, respectively), and I consider the values *ExsPerInst* = 1, 5, 10, 20. That is, rather than including all 20 view validation examples that I generate for each instance I_k , the C4.5 training sets consist of (randomly chosen) subsets of one, five, 10, or 20 view validation examples for each training instance. Within the corresponding C4.5 test sets, I continue to use all 20 view validation examples that are available for each test instance.

Figure 5.4 displays the learning curves obtained in this experiment. The empirical results suggest that the benefits of increasing *ExsPerInst* become quickly insignificant: for both WI and PTCT, the difference between the learning curves corresponding

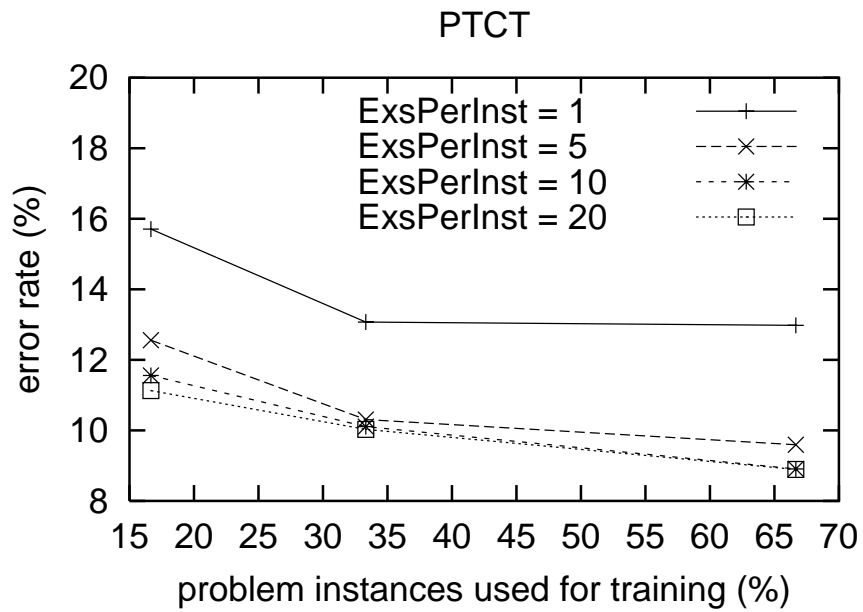
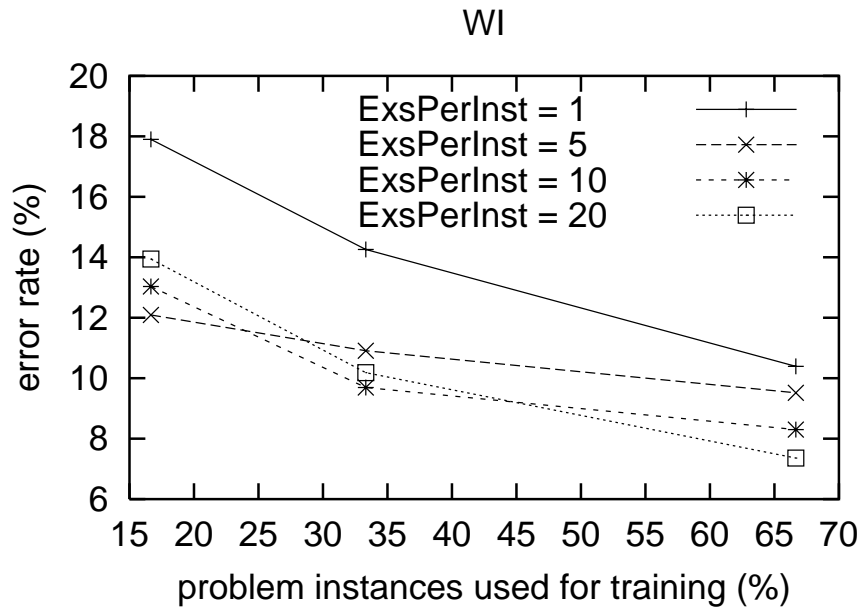


Figure 5.4: Keeping $Size(L_k)$ constant and varying the value of $ExsPerInst$ (1, 5, 10, and 20).

to $ExsPerInst = 10$ and 20 is not statistically significant, even though for the latter I use twice as many view validation examples than for the former. This implies that a (relatively) small number of view validation examples is sufficient for high-accuracy view validation. For example, the view validation algorithm reaches a 90% accuracy when trained on 33% of the problem instances (i.e., 11 and 20 training instances, for WI and PTCT, respectively). For $ExsPerInst = 10$, this means that C4.5 is trained on just 110 and 200 view validation examples, respectively.

In order to study the influence of the $Size(L_k)$ parameter, I designed an experiment in which the hypotheses h_1 and h_2 are learned based on a fraction of the examples in the original set L_k . Specifically, for WI I use two, four, and six of the examples in L_k ; for PTCT I use 20, 30, 40, 50, 60, and 70 of the examples in L_k . For both WI and PTCT, I keep $ExsPerInst = 20$ constant.

Figure 5.5 shows the learning curves obtained in this experiment. Again, the results are extremely encouraging: for both WI and PTCT I reach an accuracy of 92% without using all examples in L_k . For example, the difference between $Size(L_k) = 4$ and 6 (for WI) or $Size(L_k) = 60$ and 70 (for PTCT) are *not* statistically significant.

The experiments above suggest two main conclusions. First, for both WI and PTCT, the view validation algorithm makes high accuracy predictions. Second, my approach requires a modest effort from the user's part because both the number of view validation examples and the size of the training sets L_k are reasonably small.

5.3.5 The distribution of the errors

In order to study the errors made by the view validation algorithm, I designed an additional experiment. For both WI and PTCT, I use for training all-but-one of the problem instances, and I test the learned decision tree on the remaining instance.² That is, in this *leave one instance out* experiment, I test the learned decision tree on the 20 labeled

²For each problem instance I use the entire training set L_k and all $ExsPerInst = 20$ view validation examples.

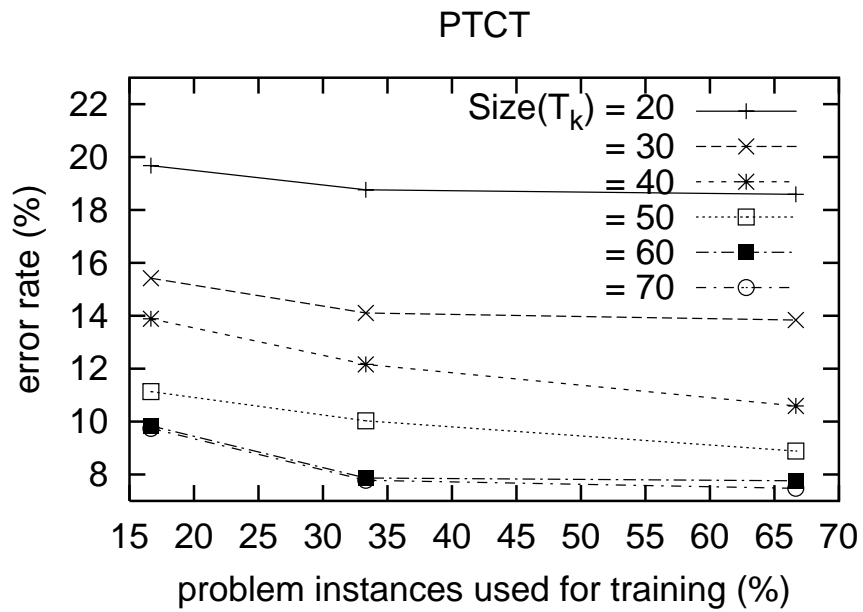
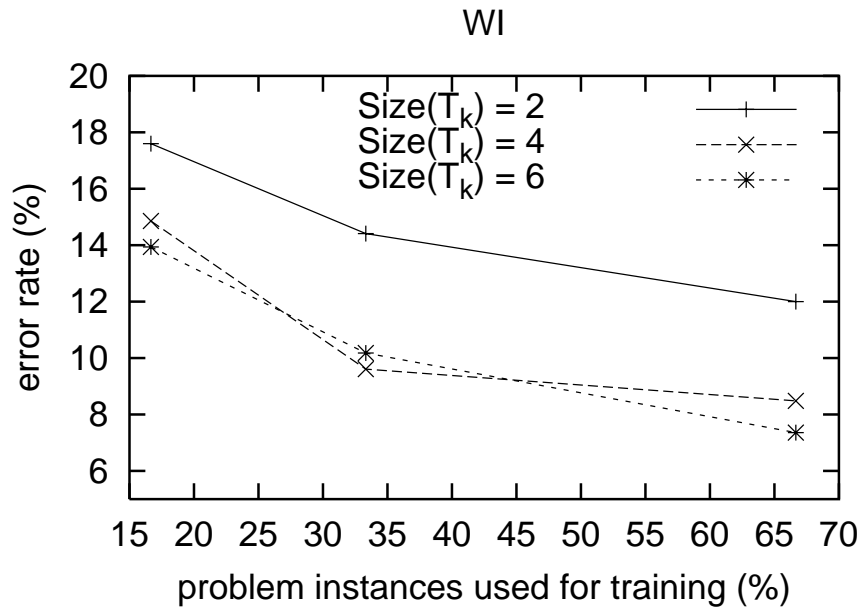


Figure 5.5: For $ExsPerInt = 20$, I consider several values for $Size(L_k)$: 2/4/6 for WI, and 20/30/40/50/60/70 for PTCT.

examples coming from the “left out” problem instance. This setup allows me to study view validation’s performance on each individual problem instance.

The graphs in Figure 5.6 display the results on the WI and PTCT datasets, respectively. On the X axis, I show the number of view validation examples that are misclassified by view validation (remember that each test set consists of the $ExsPerInst = 20$ view validation examples generated for the problem instance used for testing). On the Y axis I have the number of problem instances on which the algorithm misclassifies a particular number of view validation examples.

Consider, for example, the graph that shows the results on the 33 problem instances in WI (see Figure 5.6). The leftmost bar in the graph has the following meaning: on 22 of the problem instances, the algorithm makes *zero* errors on the view validation examples in the corresponding 22 test sets; that is, view validation correctly predicts the labels of all $ExsPerInst = 20$ examples in each test set. Similarly, the second bar in the graph means that on two other problem instances, view validation misclassifies just *one* of the 20 examples in the test set.

These results require a few comments. First, for more than half of the problem instances in both WI and PTCT, the algorithm labels correctly *all* view validation examples; i.e., regardless of the particular choice of the sets L_k and U_k that are used to generate a view validation example, the algorithm predicts the correct label. Second, for most instances of WI and PTCT (29 and 44 of the 33 and 60 instances, respectively), view validation has an accuracy of at least 90% (i.e., it misclassifies at most two of the $ExsPerInst = 20$ view validation examples). Finally, for all but one problem instance, the algorithm labels correctly *at least* 60% of the view validation examples generated for each problem instance.

For an in-depth perspective of the results above, in Figure 5.7 I split the view validation errors into two classes: false positives and false negatives. The former are the errors in which the algorithm predicts “apply multi-view algorithm” even though the views are

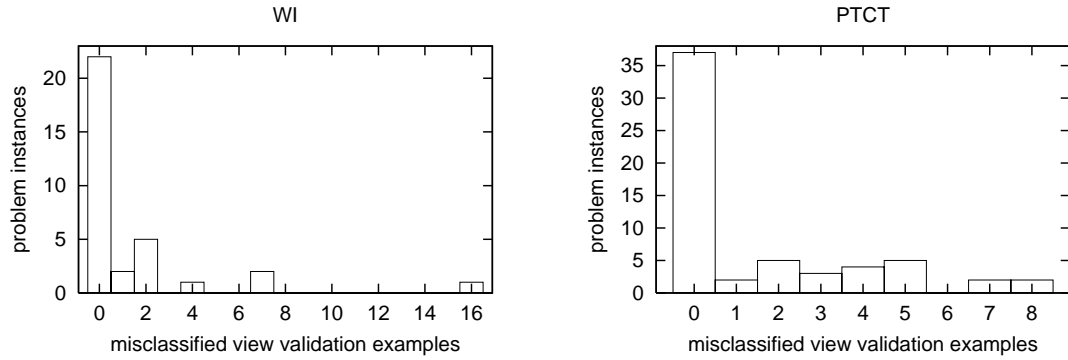


Figure 5.6: The distribution of the errors for WI (top) and PTCT (bottom).

insufficiently compatible for multi-view learning; the latter are errors in which the algorithm predicts “don’t apply multi-view algorithm” even though the views are sufficiently compatible for multi-view learning. The results are excellent both on WI and PTCT:

- for WI, where the default, most-frequent-label prediction is “use multi-view learning”, the false positives are of little concern because, by default, all negative examples would have been false positives. Among the 21 problem instances in which view validation could have potentially predicted false negatives, it did so only on six of them. Furthermore, even for these six instances the error rate is at most 20% (i.e., at most 4 of the 20 examples are mislabeled as negatives).
- for PTCT, where the default, most-frequent-label prediction is “do NOT use multi-view learning”, it is the the false negatives that are of little concern (by default, all positive examples would have been false negatives). Among the 40 problem instances in which view validation could have potentially predicted false positives, it did so only on 15 of them. Furthermore, except for one of these 15 instances, the error rates are - again - at most 20% (i.e., at most 4 of the 20 examples are mislabeled as positives).

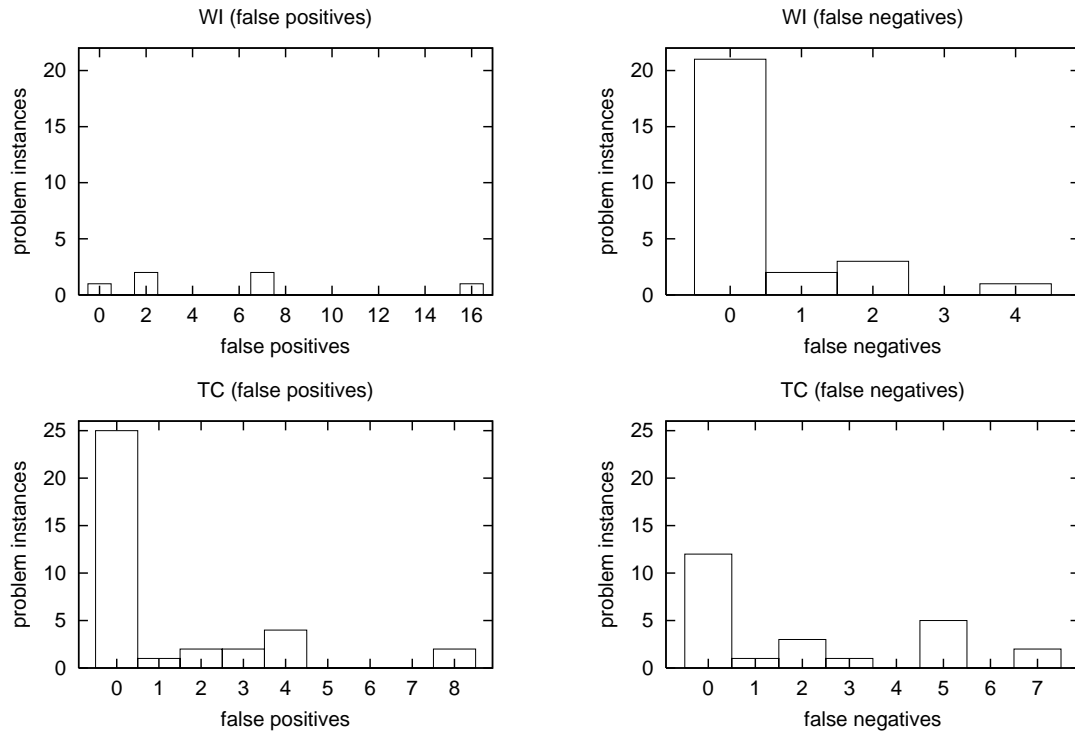


Figure 5.7: The distribution of the false positives and negatives for WI (top) and PTCT (bottom).

5.3.6 Understanding the Predictions

In practice, it is important to provide users with the intuition behind a view validation prediction. The decision trees learned by C4.5 are extremely useful with this respect. Figure 5.8 shows two illustrative pruned decision trees (one for each domain) that were learned using 66% of the problem instances. For each node in the trees, I show the following information: the view validation feature used to make the decision (i.e., one of the seven features described in Section 4); the error rate on the test set; and the number of test examples that are classified based on the node's descendants.

Consider, for instance, the PTCT decision tree, which misclassifies 4.5% of the 400 test examples (see the tree's root). The decision tree reads as follows: if the hypotheses h_1 and h_2 agree on more than 62% of the unlabeled examples in U_k (i.e., if $f_1 > 62\%$), then

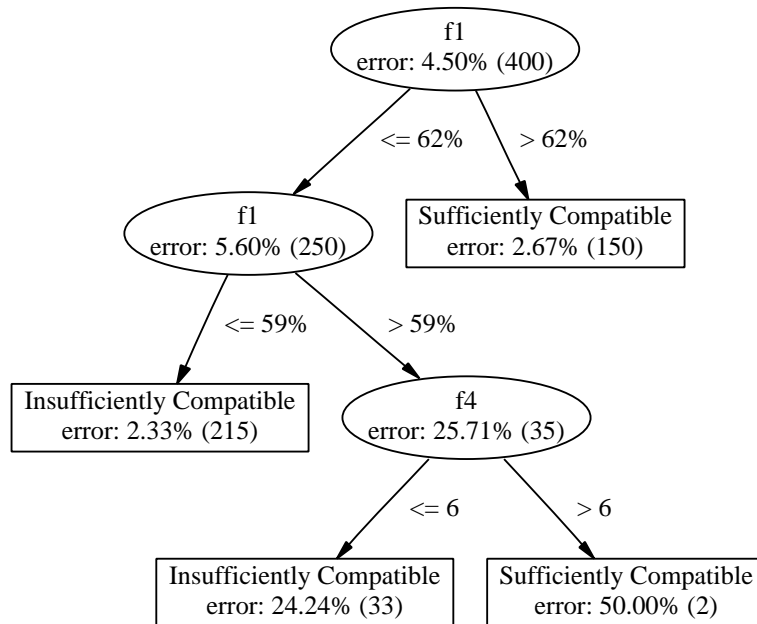
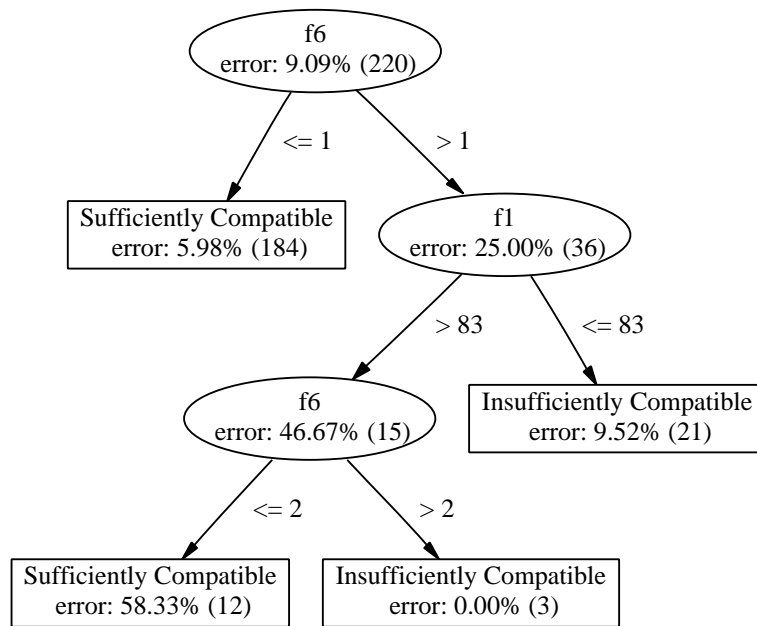


Figure 5.8: Illustrative trees for WI (top) and PTCT (bottom). The node labels “f1”, “f4”, and “f6” refer to the view validation features described in section 5.2. That is, “f1” is the agreement level between the hypotheses learned in the two views; “f4” is the absolute value of the difference between the training errors in the two views; finally, “f6” is the largest of the complexities of the hypotheses learned in the two views.

the problem instance has views that are sufficiently compatible for multi-view learning. Based on this criterion, 150 of the 400 examples are labeled “sufficiently compatible”, with an error rate of 2.67% (i.e., only four examples are misclassified).

If the two hypotheses agree on at most 59% of the unlabeled examples (i.e., $f_1 \leq 59\%$), the views are insufficiently compatible for learning. Finally, if the agreement level is between 59% and 62%, the decision is taken based on the feature f_4 : the views are sufficiently compatible if and only if the difference in training error in the two views is larger than 10% (i.e., seven of the 70 examples in L_k). This counter-intuitive decision, which is due to overfitting, produces half the errors on the entire test set (i.e., nine of the 18 misclassified examples).

5.4 Summary

In this chapter I described the first approach to view validation. My view validation algorithm uses several solved problem instances to train a classifier that discriminates between instances for which the views are *sufficiently/insufficiently compatible* for multi-view learning. For both test domains, wrapper induction and text classification, view validation requires a modest amount of training data to make high-accuracy predictions. View validation represents a first step towards the long-term goal of creating a *view detection* algorithm that automatically partitions the domain’s features in views that are adequate for multi-view learning.

Chapter 6

Related Work

*One cannot conceive anything so strange and so implausible that
it has not already been said by one philosopher or another.*

Rene Descartes

In this chapter I review the research efforts that are related to my dissertation. First, I discuss existing work on various types of active learners. Then I continue by presenting the main approaches to semi-supervised learning and meta-learning.

6.1 Active learning algorithms

The idea of active learning can be seen as a natural development from the earlier work on optimum experimental design (Fedorov, 1972). In the early 1980s, the machine learning community started recognizing the advantages of inductive systems that are capable of querying their instructors. For example, in order to detect errors in Prolog programs, the Algorithmic Debugging System (Shapiro, 1981; Shapiro, 1982) was allowed to ask the user several types of queries. Similarly, concept learning systems such as Marvin (Sammut and Banerji, 1986) and CAT (Gross, 1991) used queries as an integral part of their respective learning strategies.

My review of the existing approaches to active learning is structured as follows. First, I discuss the early, mostly theoretical results in *query construction*. Then I focus on

selective sampling algorithms, which - rather than constructing a query - select as the next query one of the unlabeled examples from the working set.

6.1.1 Active learning by query construction

The earliest approaches to formalizing active learning appeared in the seminal papers of Angluin (1982; 1988) and Valiant (1984), who focused on exact concept induction and learning in the PAC framework, respectively. This theoretic work focused on learning classes of concepts such as regular sets, monotone DNF expressions, and μ -expressions. Besides *membership queries* such as “is this an example of the target concept?,” Angluin also used more sophisticated types of queries such as *equivalence queries* (“is this concept equivalent with the target concept?”) or *superset queries* (“is this concept a superset of the target concept?”).

These early active learners took a *constructive* approach to query generation in the sense that each query is (*artificially*) constructed by setting the values of the attributes so that the query is as informative as possible. In practice, this may raise some serious problems; for example, as discussed in (Lang and Baum, 1992), consider a hand-writing recognizer that must discriminate between the 10 digits. In this scenario, an informative query may consist of an image that represents a “fusion” of two similarly-looking digits, such as “3” and “5.” When presented with such an image, a user cannot label it properly because it does not represent a recognizable digit. Consequently, a query is “wasted” on a totally irrelevant image. Similar situations appear in many real world tasks such as text classification, information extraction, or speech recognition: whenever the active learner artificially builds a query for such a domain, it is highly unlikely that the newly created object has any meaning for the human user.

Despite this practical applicability issue, the constructive approach to active learning leads to interesting theoretical insights about the merits of various types of queries. For example, researchers considered learning with:

- *incomplete queries*, for which the query’s answer may be “I don’t know.” (Angluin and Slonim, 1991; Goldman and Mathias, 1992; Sloan and Turan, 1994; Blum et al., 1998);
- *malicious queries*, for which the answer to the queries may be erroneous (Angluin et al., 1997; Angluin and Krikis, 1994; Angluin, 1994).

New learning problems were also considered, from unrestricted DNF expression (Jackson, 1994; Blum et al., 1994) and unions of boxes (Goldberg, Goldman, and Mathias, 1994) to tree patterns (Amoth, Cull, and Tadepalli, 1998; Amoth, Cull, and Tadepalli, 1999) and Horn clauses (Reddy and Tadepalli, 1997). Researchers also reported results on applying active learning to neural networks (Hwang et al., 1991; Baum, 1991; Watkin and Rau, 1992; Hasenjager and Ritter, 1998) and for combining declarative bias (prior knowledge) and active learning (Tadepalli, 1993; Tadepalli and Russell, 1998).

6.1.2 Selective sampling

Selective sampling represents an alternative active learning approach. It typically applies to *classification* tasks in which the learner has access to a large number of unlabeled examples. In this scenario, rather than constructing an informative query, the active learner asks the user to label one of the existing unlabeled examples. Depending on the *source* of unlabeled examples, there are two main types of sampling algorithms: stream- and pool- based. The former assumes that the active learner has access to an (infinite) stream of unlabeled examples (Freund et al., 1997; Argamon-Engelson and Dagan, 1999; Dagan and Engelson, 1995); as successive examples are presented to it, the active learner must decide which of them should be labeled by the user. In contrast, in the pool-based scenario (Lewis and Gale, 1994; Lewis and Catlett, 1994; McCallum and Nigam, 1998b; Muslea, Minton, and Knoblock, 2000b; Muslea, Minton, and Knoblock, 2002a), the learner is presented with a *working set* of unlabeled examples; in order to make a

query, the active learner goes through the entire pool and selects the example to be labeled next.

Based on the criterion used to select the next query, selective sampling algorithms fall under three main categories:

- *uncertainty reduction*: the system queries the example on which the current hypothesis makes the least confident prediction;
- *expected-error minimization*: the system queries the example that maximizes the *expected* reduction in classification error;
- *version space reduction*: the system queries the example that, once labeled, removes as much as possible of the version space.

The *uncertainty reduction* approach to selective sampling works as follows: first, one uses the labeled examples to learn a classifier; then the system queries the unlabeled example on which this classifier makes the *least confident* prediction. This straightforward idea can be applied to any base learner for which one can estimate the confidence of its predictions. Confidence-estimation heuristics were proposed for a variety of base learners such as logistic regression (Lewis and Gale, 1994; Lewis and Catlett, 1994), partially hidden Markov Models (Scheffer and Wrobel, 2001), support vector machines (Schohn and Cohn, 2000; Campbell, Cristianini, and Smola, 2000), and inductive logic programming (Thompson, Califf, and Mooney, 1999).

The second, more sophisticated approach to selective sampling, *expected-error minimization*, is based on the *statistically optimal* solution to the active learning problem. In this scenario, the intuition is to query the unlabeled example that minimizes the error rate of the (future) classifier on the test set. Even though for some (extremely simple) base learners one can find such optimal queries (Cohn, Ghahramani, and Jordan, 1996), this is not true for most inductive learners. Consequently, researchers proposed methods to *estimate* the error reduction for various types of base learners. For example, (Roy and McCallum, 2001) use a sample estimation method for the Naive Bayes classifier;

similar approaches were also described for parameter learning in Bayesian nets (Tong and Koller, 2000a) and for nearest neighbor classifiers (Lindenbaum, Markovitch, and Rusakov, 1999).

The heuristic approach to *expected-error minimization* can be summarized as follows. First, one chooses a *loss function* that is used to estimate the future error rate (see (Roy and McCallum, 2001) for a description of the *log* and *0-1* loss functions). Then each unlabeled example x in the working set is considered as the possible next query, and the system estimates the expected reduction of the error rate for each possible label that x may take. Finally, the system queries the unlabeled example that leads to the largest estimated reduction in the error rate.

Finally, a typical *version space reduction* active learner works as follows: it generates a *committee* of several hypotheses, and it queries the unlabeled examples on which the predictions of the committee's members are the most split. In a 2-class learning problem, this strategy translates into making queries that remove approximately half of version space. Depending on the method used to generate the committee, one can distinguish several types of active learners:

- Query-by-Committee selects a committee by randomly sampling hypotheses from version space. Query-by-Committee was applied to a variety of base learners such as perceptrons (Freund et al., 1997), Naive Bayes (McCallum and Nigam, 1998b), and Winnow (Liere and Tadepalli, 1997). Furthermore, Argamon-Engelson and Dagan (1999; 1995) introduce an extension to Query-by-Committee for Bayesian learning (Mitchell, 1998). In the Bayesian framework, one can create the committee by sampling classifiers according their posterior distributions; that is, the better a hypothesis explains the training data, the more likely it is to be sampled. The main limitation of Query-by-Committee is that it can be applied only to base learners for which it is feasible to sample hypotheses from a version space.

- SG-net (Cohn, Atlas, and Ladner, 1994) creates a 2-hypothesis committee that consists of a “*most-general*” and a “*most-specific*” classifier. These two hypotheses are generated by *modifying the base learner* so that it learns a classifier that labels as many as possible of the unlabeled examples in the working set as positive or negative, respectively. This approach has an obvious drawback: it requires the user to modify the base learner so that it can generate “*most-general*” and “*most-specific*” classifiers.
- Query-by-Bagging and Query-by-Boosting (Abe and Mamitsuka, 1998) create the committee by using the well-known bagging (Breiman, 1996) and boosting (Schapire, 1990) algorithms, respectively. These algorithms were introduced for the C4.5 base learner, for which both bagging and boosting are known to work extremely well.

In general, committee-based sampling tends to be associated with the *version space reduction* approach. However, for base learners such as support vector machines, one can use a *single hypothesis* to make queries that remove (approximately) half of version space (Tong and Koller, 2001; Tong and Koller, 2000b). Conversely, committee-based sampling can also be seen as relying on the *uncertainty reduction* principle: after all, the unlabeled example on which the committee is the most split can be also seen as the example that has the least certain classification.

6.1.3 Co-Testing *vs* existing active learners

There are two main differences between Co-Testing and the other approaches to selective sampling. First of all, Co-Testing is the only multi-view active learner; all other active learning algorithms work in the single-view framework.

Second, Co-Testing takes a “problem-oriented” approach to selective sampling; that is, Co-Testing can be applied to any multi-view problem, regardless of the base learner of choice. In contrast, single-view algorithms are typically designed for a particular (class of) base learner(s). For example, Query-by-Committee applies to learners for which one

can sample hypotheses from version space, while Uncertainty Sampling applies to base learners that can evaluate the confidence of their predictions.

Co-Testing’s “problem-oriented” approach to active learning has both advantages and disadvantages. On one hand, Co-Testing cannot be applied on problems that do not have at least two views. On the other hand, for any multi-view problem, Co-Testing can be used in a straightforward manner with the base learner that works the best for that particular type of task. In contrast, in the single-view framework, one often must either create a new active learning method that can accommodate a particular base learner or, even worse, modify an existing base learner so that it can be used in conjunction with an existing sampling algorithm.

Finally, a few more observation that contrast Co-Testing with other active learners. First, Co-Testing can be seen as a committee-based approach to selective sampling: Co-Testing generates a committee that consists of one hypothesis from each view and queries examples on which the committee’s predictions are split (i.e., contention points). However, as opposed to single-view approaches, Co-Testing does not generate the committee based on properties inherent to the base learner, but rather based on properties pertaining to the problem to be solved (i.e., the multiple views).

Second, Co-Testing can be combined with virtually any of the single-view active learners. That is, among the contention points, a Co-Testing algorithm can choose the next query based on any of the heuristics used by the single-view selective samplers. For example, Aggressive Co-Testing, which queries the contention point on which both views make the most confident prediction, can be seen as “borrowing” the heuristic from Uncertainty Sampling. Novel members of the Co-Testing family can be created in a similar manner by relying on heuristics such as *expected-error minimization* or *version space reduction*.

6.2 Semi-supervised concept learning

In this section I discuss two main classes of algorithms that combine labeled and unlabeled data: single-view and multi-view semi-supervised learners. I begin with the single-view classifiers, which represent the traditional approach to semi-supervised learning. Then I focus on the (recent) developments in multi-view, semi-supervised learning.

6.2.1 Single-view, semi-supervised classification

Based on the way in which they used the unlabeled examples, there are three major approaches to single-view, semi-supervised classification: transduction, expectation maximization, and “background knowledge injection.” Transduction maximizes the classification accuracy on a particular test set by using as the *working set* the (unlabeled) examples in the actual test set. In contrast, the other two approaches aim at improving the classifiers’s accuracy over the entire instance space. In order to achieve this goal, they supplement the training set with unlabeled examples from a working set, which is distinct from the test set. In expectation maximization, the working set is used to create a *generative model* for the data; that is, a model that is likely to have generated the seen data. In contrast, the “background knowledge injection” approach uses the unlabeled data to synthesize some sort of knowledge that can be used to improve the accuracy of the supervised learner.

6.2.1.1 Transductive approaches

The best way to introduce the intuition behind the transductive learning framework (Vapnik, 1998) is to contrast it with inductive learning. In inductive learning, one searches for a hypothesis that has the smallest error rate over the entire instance space. In contrast, transductive learning generates a classifier that is as accurate as possible on a particular test set; that is, given a set of labeled examples, in the transductive framework one learns a distinct, maximally-accurate classifier for each test set of interest. In practice, this

translates into combining the labeled examples in the training set with the unlabeled ones from the test set.

Research in transductive learning lead to some interesting and contradictory results. On one hand, Zhang and Oles (2000) provide both theoretical and empirical evidence that, in general, transductive support vector machines (TSVM) are unlikely to improve the classification accuracy. On the other hand, several papers (Joachims, 1999; Bennett and Demiriz, 1998; Bennett and Demiriz, 2000) contradict the results in (Zhang and Oles, 2000) by showing that TSVMs can reduce the need for labeled data by up to 95%.

In (Szummer and Jaakkola, 2000), the authors describe a transductive learning algorithm for kernel density estimation. In their scenario, the labeled and unlabeled data are used to evaluate the density of the instance space. This instance density is subsequently used to estimate the importance of each individual labeled example, thus improving the classification accuracy.

Transductive classification was also applied to learning with Probabilistic Relation Models (PRMs)(Taskar, Segal, and Koller, 2001). As PRMs exploit the relational structure in the data, it is crucial to have access to the (unlabeled) data in the test set: in the absence of the test data, one misses not only the relational information pertaining to it, but also the (highly informative) relational information that “connects” the labeled and unlabeled examples.

6.2.1.2 Expectation Maximization

Within the statistics community, the problem of learning generative models from both labeled and unlabeled data has received significant attention (Hartley and Rao, 1968; Day, 1969; MacLachlan, 1975) for at least a decade before the seminal work on *Expectation Maximization* (EM) (Dempster, Laird, and Rubin, 1977). The EM paper, which had a tremendous influence on the machine learning community, formalizes the idea of an iterative approach to likelihood maximization in the presence of missing data; that is, an iterative approach to finding the generative model that is the most likely to have

generated the seen examples. In the concept learning framework, this translates into searching for the classifier that explains the best the data from the training and working set.

The EM algorithm stimulated an impressive amount of both theoretical and practical work. On the theoretical side, researchers focused mainly on answering the following question: “*how many unlabeled examples is a labeled example worth?*” (Cover and Thomas, 1991; Ratsaby and Venkatesh, 1995). Under strong assumptions (e.g., knowing the parametric form of the target function, and having no “local optimum” points in the search space), convergence results were proved for various types of mixtures of two Gaussians (Ganesalingam and McLachlan, 1969; O’Neill, 1978; Ratsaby and Venkatesh, 1995). One of the most intuitive and best known results can be summarized as follows: under certain assumptions,

- if an infinite amount of unlabeled data is available, labeled examples reduce the classification error exponentially fast (Castelli and Cover, 1995);
- if only a finite amount of labeled and unlabeled data is available, the labeled examples are exponentially more informative than the unlabeled ones (Castelli and Cover, 1996).

The semi-supervised EM algorithm was applied to a variety of real world domains, from face orientation discrimination (Baluja, 1998) to text classification (Nigam et al., 1998) and English text tagging (Merialdo, 1994). These EM implementations were created for base learners such as Naive Bayes (Nigam et al., 1998; Baluja, 1998), dependency trees (Baluja, 1998), and hidden Markov models (Merialdo, 1994). A known drawback (Merialdo, 1994; Nigam et al., 2000; Cozman and Cohen, 2002) of these approaches is the following: if the data does not conform to the generative model (i.e., if the base learner cannot perfectly learn the target concept), the unlabeled examples may actually decrease the classification accuracy. In order to cope with this problem, researchers proposed two main approaches. First, one can use more sophisticated base learners, such

as multiple mixture components for each domain class (Nigam et al., 2000; Shahshahani and Landgrebe, 1994; Miller and Uyar, 1996). Second, the labeled examples can be assigned more importance than the unlabeled ones (Nigam et al., 2000), thus mitigating the effect of not using the ideal base learner.

6.2.1.3 Unlabeled data as “background knowledge”

A third approach to single-view, semi-supervised classification can be characterized as using the unlabeled examples as some sort of background knowledge. For example, Zelikovitz and Hirsh (2000) introduce a novel strategy to boost the accuracy of a nearest neighbor classifier. Rather than directly measuring how similar sets of test and a training examples are, the new algorithm compares their relative similarities to a group of unlabeled examples from the working set. The intuition in (Zelikovitz and Hirsh, 2000) is straightforward: instead of relying on an imperfect metric for the similarity of two examples, the authors use the stronger evidence of the two examples being similar to an entire group of (unlabeled) examples.

Another approach to using the unlabeled examples as background knowledge is presented in (Raskutti, Ferra, and Kowalczyk, 2002b). The authors use the unlabeled examples to enrich the original set of features with some additional, highly-informative attributes. In a first step towards creating the new features, a clustering algorithm is applied to the examples in the training and working sets, and then all but the N largest clusters are discarded. In the second phase, for each example x and each remaining cluster C , the algorithm adds to x novel features such as a binary attribute that answers the question “*is C the closest cluster to x?*,” a numeric attribute that measures similarity of x to the centroid of the cluster C , etc. Raskutti *et al.* (2002b) show that by using the enriched set of features, one can significantly improve the classification accuracy.

Finally, the ASSEMBLE algorithm (Bennett, Demiriz, and Maclin, 2002) uses the unlabeled examples to create a diverse committee that consists of hypotheses that make highly consistent predictions; that is, most hypotheses predict the same label for any

unlabeled examples. By construction, such a committee prevents overfitting, thus increasing ASSEMBLE’s generalization power. ASSEMBLE iteratively creates the committee in the following manner: first, it uses the current committee to label the unlabeled data in the working set. Second, ASSEMBLE learns a new hypothesis from both the training and (newly labeled) working sets. This new hypothesis is added to the committee, and the whole process is repeated for a number of iterations. Note that ASSEMBLE’s committee is similar to the one that would consist of the hypotheses learned after each EM iteration. However, there are two main differences between the two approaches. First of all, instead of using Expectation Maximization, ASSEMBLE simply trains the base learner on the labeled examples. Consequently, ASSEMBLE can be used with any base learner, not only with generative models. Second, rather than aiming to incrementally learn a “better” hypothesis in an EM-like manner, ASSEMBLE focuses on creating a committee in which all/most members label identically the examples in the working set.

6.2.2 Multi-view, semi-supervised learning

As I already mentioned, Blum and Mitchell (1998) provided the first formalization of learning in the multi-view framework. They proved that two independent, compatible views can be used to PAC-learn (Valiant, 1984) a concept based on few labeled and many unlabeled examples. Blum and Mitchell also introduced the Co-Training, which is the first general-purpose, multi-view algorithm.

Collins and Singer (1999) proposed a version of Co-Training that is biased towards learning hypotheses that predict the same label on most of the unlabeled examples. They introduce an explicit objective function that measures the compatibility of the learned hypotheses and use a boosting algorithm to optimize this objective function. In a related paper (Dasgupta, Littman, and McAllester, 2001), the authors provide PAC-like guarantees for this novel Co-Training algorithm (the assumption is, again, that the views are both independent and compatible). Intuitively, Dasgupta *et al.* show that the ratio

of contention points¹ to unlabeled examples is an upper-bound on the error rate of the classifiers learned in the two views.

In a recent development, Abney (2002) extends the work of Dasgupta *et al.* by relaxing the view independence assumption. More precisely, Abney shows that even with views that are *weakly dependent*, the ratio of contention points to unlabeled examples still represents an upper-bound on the two view's error rate. Unfortunately, (Abney, 2002) introduces just a theoretical definition for the *weak dependence* of the views, without providing an intuitive explanation of practical consequences of this *weak dependence*.

Researchers proposed two main types of extensions to the original Co-Training algorithm: modifications of the actual algorithm and changes aiming to extend its practical applicability. The former cover a wide variety of scenarios:

- Co-EM (Nigam and Ghani, 2000) uses Expectation Maximization (Dempster, Laird, and Rubin, 1977) for multi-view learning. Co-EM can be seen as the closest implementation of the theoretical framework proposed in (Blum and Mitchell, 1998).
- Ghani (2002) uses Error-Correcting Output Codes to allow Co-Training and Co-EM to scale up well to problems with a large number of classes.
- Corrected Co-Training (Pierce and Cardie, 2001) asks the user to manually correct the labels of the bootstrapped examples. This approach is motivated by the observation that the quality of the bootstrapped data is the key factor in the convergence of Co-Training.
- Co-Boost (Collins and Singer, 1999) and Greedy Agreement Algorithm (Abney, 2002) are Co-Training algorithms that explicitly minimize the number of contention points.

The second group of extensions to Co-Training is motivated by the fact that, in practice, one also encounters many problems for which there is no straightforward way to split the features in two views. In order to cope with this problem, Nigam and Ghani

¹Remember that a contention point is an unlabeled example for which the hypotheses learned in the two views predict a different label.

(2000) show that, for “bag-of-words” text classification, one can create two views by arbitrarily splitting the original set of features into two sub-sets. Such an approach fits well the text classification domain, in which the features are abundant, but it is unlikely to work on other types of problems.

An alternative solution is proposed in (Raskutti, Ferra, and Kowalczyk, 2002a), where the authors create a second view that consists of a variety of features that measure the examples’ similarity with the N largest clusters in the domain. In fact, (Raskutti, Ferra, and Kowalczyk, 2002a) is tightly related to (Raskutti, Ferra, and Kowalczyk, 2002b), which was discussed in section 6.2.1.3: the two papers show that the clusters-based features are highly informative both in the multi- and single- view framework, respectively.

Finally, (Goldman and Zhou, 2000) advocates the use of *multiple biases* instead of multiple views. The authors introduce an algorithm similar to Co-Training, which bootstraps from each other hypotheses learned by two different base learners. The only assumption is that the these base learners generate hypotheses that partition the instance space into equivalence classes.

6.2.3 Co-EMT *vs.* existing approaches

The only approach similar to Co-EMT was proposed by McCallum and Nigam (1998b), who introduced a family of algorithms that combine active and semi-supervised learning. More precisely, they use the highly informative (labeled) examples chosen by an active learner (Seung, Opper, and Sompolinski, 1992) as the training set for the semi-supervised EM algorithm.

There are two main distinction between the algorithms in (McCallum and Nigam, 1998b) and Co-EMT (Muslea, Minton, and Knoblock, 2002a). First of all, the algorithms introduced by McCallum and Nigam (1998b) are single-view, while Co-EMT is multi-view. Second, the experiments in (McCallum and Nigam, 1998b) show interleaving active and semi-supervised learning does not lead to a higher accuracy than applying

the algorithms after each other. In contrast, our results showed that - in the multi-view setting - interleaving the two algorithms dramatically outperforms applying them in sequence.

6.3 Meta-learning & model selection

In this section I discuss the “learning to learn” (Thrun and Pratt, 1997) paradigm, which is closely related to my View Validation algorithm (Muslea, Minton, and Knoblock, 2002b). Given the breadth of the “learning to learn” topic, I first discuss the larger context and present a few illustrative examples of existing approaches; then I focus on meta-learning model selection algorithms, of which the View Validation is an example.

The seminal work on the *no free lunch theorems* (Schaffer, 1993a; Schaffer, 1994; Wolpert, 1996; Wolpert and Macready, 1997) showed that there is no learning algorithm that can outperform random guessing on *all* possible classifications of an instance space. These results lead to two main directions of research: *model combination* and *model selection*. For a given learning task, the former aims at combining the predictive power of several learners. Algorithms such as bagging (Breiman, 1996), boosting (Schapire, 1990), or stacked generalization (Wolpert, 1992) represent highly successful approaches to model combination.

The main focus of this section is on *model selection* methods. A typical model selection algorithm is given a learning task T and a number of learning algorithms l_1, l_2, \dots, l_n , and it is required to predict which of the n learners obtains the best accuracy on the task T . There are four main approaches to model selection: experimental, knowledge-driven, transfer of learning, and meta-learning.

6.3.1 Experimental model selection

The main idea in *experimental model selection* is based on the following assumption: if an algorithm trained on a subset of the training data makes accurate predictions, it is likely

to also make high accuracy predictions when trained on the entire dataset. A typical approach to experimental model selection is to use *cross-validation* (Schaffer, 1993b) to estimate the accuracy of each learner on the task T ; then the system solves the task T by applying the learner l_w that has obtained the highest accuracy during cross-validation.

In a related approach (Petrač, 2000), the author introduces a *subsampling* algorithm for error estimation. This methodology is motivated by the work in data mining, where extremely large datasets make cross-validation impractical. Petrač (2000) shows that by estimating the error rate based on a single subsample of a large dataset, one obtains a good estimate of an algorithm's performance. The results in (Petrač, 2000) also indicate that, for error estimation, it is much more important to have a large test set than a large training set.

Finally, the *wrapper method* (Kohavi and John, 1995) is used in to find the optimal value of the parameters for a given learning algorithm. The authors use best-first search and cross-validation to explore the space of parameters settings. The wrapper method can be seen as applying cross validation at two different levels: once for parameter estimation and once for the actual error estimation.

6.3.2 Knowledge-driven model selection

The knowledge-driven approach to model selection is based on the idea of extracting meta-knowledge from the user and using it to select the most appropriate algorithm for a particular task. The meta-knowledge can be seen as a set of heuristics that can be used for model selection. For example, Shavlik *et al.* (1991) perform an extensive empirical evaluation of the ID3, perceptron, and back-propagation algorithms on five datasets and analyzed the effect of several factors (e.g., classification noise and type of features) on the accuracy of each learner. The authors found some interesting heuristics, such as “back-propagation outperforms ID3 on datasets with numerical features.”

Similarly, in (King, Feng, and Shutherland, 1995), the authors perform a larger scale experiment, in which they compare 17 algorithms on 12 datasets. For each dataset, they

measure 12 statistical characteristics (e.g., the homogeneity of the covariances, and the attributes's skew and kurtosis) that are used to define a set of heuristics for model selection. For example, one such heuristic states that “domains with extreme distributions (*skew*>1 and *kurtosis*>7) and with many binary/discrete attributes (>38%) favor symbolic learners.” A similar approach is used in (Engels and Theusinger, 1998) for finding heuristics for pre-processing a dataset (e.g., eliminate redundant and uninformative attributes).

In (Brodley, 1995), the author takes the idea above to a higher level. She introduces the Model Class Selection (MCS) system, which creates a tree-structure hybrid classifier that applies different classifiers (i.e., linear discriminant functions, decision trees, and instance-based learning) in various regions of the instance space. Intuitively, MCS can be seen as a 3-step process: first, the user proposes several heuristics for model selection, which are similar to the one described above. The heuristics are used by MCS to recursively build the hybrid, tree-like classifier that splits the instance space into regions that are well-suited for a particular learner. In the second step, a few datasets are used to (iteratively) debug the heuristics: MCS builds a hybrid classifier, whose performance is compared with that of the three individual learners; if the hybrid classifier does not perform *at least* as well as the best of the three learners, the user finds and repairs the faulty heuristics, and the whole process is repeated. Finally, when the system is deployed, the hybrid tree-like classifier is used to apply the appropriate classifier to each region of the instance space.

There are two main disadvantages to *experimental* and *knowledge-driven* model selection. First of all, they represent time consuming processes that do not scale well to the plethora of learning algorithms and the number of parameters that must be tuned for each of them. Second, they both ignore a powerful source of knowledge: the experience acquired while solving other related learning tasks. These drawbacks are addressed by the other two approaches to model selection: *transfer of learning* and *meta-learning*.

6.3.3 Transfer of learning

Transfer of learning refers to finding a model that is appropriate for a *group* of related tasks. For example, Baxter (1996; 1997) introduces a Bayesian framework for learning to predict which is the most appropriate model for an *environment of similar tasks*. The main intuition is the following: when solving several tasks from the same environment, one can exploit the information from all these tasks to find the model that is the most appropriate for the entire class of problems. Baxter provides both theoretical and empirical evidence that his approach reduces the amount of data required for training on each particular task.

Multitask Learning (Caruana, 1997) takes a different approach to transfer of learning. Multitask Learning simultaneously learns both the concept of interest (the *main concept*) and a set of other, related concepts (the *auxiliary concepts*). For instance, consider the task of predicting, prior to hospitalization, the *patient risk* of people diagnosed with pneumonia. As explained in (Caruana, 1996), a typical single-task approach to this problem is to use a set of 30 basic measurements (e.g., age, sex, pulse) to make a prediction. In contrast, Multitask Learning also considers an additional set of 35 lab tests (e.g., blood cells counts, and blood gases) that are available *only* for patients that were hospitalized in the past. Even though these features cannot be directly used for predictions, which must be made prior to hospitalization, Multitask Learning uses them as *auxiliary* learning tasks: from the 30 available attributes, Multitask Learning *simultaneously* learns to predict *both* the patient risk *and* all these 35 additional measurements. Caruana shows that Multitask Learning for neural networks (Caruana, 1995), decision trees (Caruana, 1996), and k-nearest neighbor (Caruana, 1996) outperform their respective single-task counterparts.

The main problem with transfer of learning approaches is that they cannot be applied to new, unseen tasks. In other words, given a set of related task, transfer of learning finds the model that is the most appropriate to all of them. However, when presented with a new, unseen learning task, transfer of learning cannot predict which learner should

be used for this task. This issue is central to the meta-learning approaches to model selection, which are discussed in the next section.

6.3.4 Meta-learning for model selection

Meta-learning is the process of learning how to guide a user in applying machine learning techniques. Meta-learning captures valuable knowledge from *meta-data* (i.e., how, for a particular learner, different factors influence its success or failure); this knowledge is then used to increase the efficiency of the learning process. In other words, meta-learning refers to “learning how to learn well.”

Meta-learning is used for a variety of tasks such as feature selection (Kamolvilassatian, 2002), discriminating between a base learner’s correct and incorrect predictions (Bay and Pazzani, 2000), finding the best way to deal with missing attributes (Feng, 2000), detecting concept drift in online learning (Widmer, 1997), selecting the best classifier for each example in a domain (Wolpert, 1992; Chan and Stolfo, 1997; Todorovski and Dzeroski, 2000; Ting and Witten, 1999; Gama and Brazdil, 2000; Kaynak and Alpaydin, 2000), and learning when, how much, and how to prune decision tree (Bensusan, 1998). My focus here is on meta-learning for model selection; that is, how can one learn to predict which is the most appropriate learner for a new, unseen dataset?

For sake of simplicity, in the remainder of this section I use the term meta-learning to denote “meta-learning for model selection.” The VBMS system (Rendell, Seshu, and Tcheng, 1987), which predicts the best learning algorithm to solve a task, represents the first approach to meta-learning for model selection. VBMS can be seen as a proof-of-concept system: based on only two meta-features (i.e., the number of examples, and attributes in the domain), it chooses the best among three algorithms. The only criterion based on which an algorithm is preferred to the others is its execution time (the three algorithms reach similar accuracy on the considered tasks).

A typical meta-learning process consists of three main steps: first, one must choose the meta-features to be used. Second, a set of “training tasks” are used to create meta-examples; that is, one meta-example is created for each training task. Each such meta-example is described by the values of the meta-features, together with a *label* that designates the most appropriate learner for the corresponding “training task.” Finally, a machine learning algorithm is used to learn a classifier that predicts the learner to be applied to a new, unseen task. As the selection of the algorithm that is applied to the meta-data is in itself subject to the use of meta-learning (i.e., meta-meta-learning), the best way to discriminate between the various approaches to meta-learning is by analyzing the type of meta-features used by a system.

At the highest level, one can distinguish between two main types of meta-features: *application-related constraints* and *dataset measurements*. The former are mostly related to defining the “scope” of the meta-learner: the meta-learner’s search space can be dramatically reduced by taking into account the user’s requirements for the learners among which the meta-learner must choose. Such requirements may consist of constraints on the running time and accuracy of each learner, the understandability of the learned classifiers, or the learners’ ability to cope with noise, uncertainty, or redundant data.

The second type of meta-features are created by *dataset measurements*. These meta-features fall under three main categories: *general dataset characteristics*, *classifier-based features*, and *landmarking features*. The *general dataset characteristics* can be further divided in the following categories:

- *simple features*, such as the number of classes, attributes, and examples in the domain, or the number of symbolic, binary, and numeric attributes.
- *features based on discriminative analysis*, such as the number of discriminant functions, the relative importance of the largest eigenvalue, or the canonical correlation between the most significant discriminant function and the class distribution.

- *statistical features*, such as default accuracy (i.e., the relative size of the most populated class), the amount of classification noise, the frequency of the missing values, or standard deviation, skewness, and kurtosis of each class.

- *information-based features*, such as the class, attribute, and joint entropy, the mutual information entropy, or the information gain.

Based on (subsets of) the meta-features above, researchers have created meta-learners that use the CN2 rule-generating algorithm (Aha, 1992a), case-based reasoning (Lindner and Studer, 1999), nearest neighbor (Gama and Brazdil, 1995), the C4.5 decision tree learner (Brazdil, Gama, and Henery, 1984; Gama and Brazdil, 1995), inductive logic programming (Todorovski and Dzeroski, 1999), or regression (Koepf, Taylor, and Keller, 2000; Gama and Brazdil, 1995) to predict the most appropriate learner for a new, unseen task. The Zoomed Ranking algorithm (Soares and Brazdil, 2000) uses the same type of meta-features, but takes a more sophisticated approach to meta-learning. First, it uses a k -nearest neighbor algorithm to detect the k most similar training domains to the one on which the prediction must be made. Then it uses the candidate algorithms' performance on these k domains to *rank* their predicted performance on the new domain.

The *classifier-based* meta-features are created by applying a learner to the dataset and measuring various properties of the learned hypothesis. For example, the ENTRENCHER system (Bensusan, 1999) learns a decision tree from the dataset and uses as meta-features descriptors such as the number of tree nodes per domain attribute, the average strength of support of each leaf in the tree, the maximum depth of the tree, or the shape of the decision tree. In a related paper (Bensusan, Giraud-Carrier, and Kennedy, 2000), the authors take this idea one step farther: rather than extract the *classifier-based* meta-features from each tree, they use a higher-order inductive algorithm that learns directly from the decision trees.

Finally, landmarking (Pfahring, Bensusan, and Giraud-Carrier, 2000; Bensusan, Giraud-Carrier, and Pfahring, 2000; Bensusan and Giraud-Carrier, 2000) creates meta-features that consist of the accuracies reached by several simple and efficient learners on the dataset. Based on these meta-features, the meta-learner learns rules that predict the winner in a pair-wise comparison of the algorithms to be applied on a new, unseen dataset. Landmarking can be seen as extending the *classifier-based* meta-features from one type of classifier to a variety of simple classifiers. In contrast to measuring *dataset characteristics*, landmarking is extremely efficient: some of the dataset characteristics meta-features are computed in $O(N^3)$ complexity, where N is the number of examples in the domain.

6.3.4.1 Adaptive view validation for model selection

The *adaptive view validation* algorithm (Muslea, Minton, and Knoblock, 2002b) is the first meta-learning approach to deciding whether or not multi-view learning is appropriate for new, unseen tasks. Furthermore, view validation is also the only meta-learner designed for domains in which only a few of the examples are labeled. Consequently, view validation requires meta-features that are appropriate for the multi-view scenario, in which one learns from few labeled and many unlabeled examples.

The meta-features used for view validation fall under two broad categories. First, the meta-features that measure the training error in each view can be seen as *landmarking features*; that is, the hypotheses learned in each view represent “landmarkers” in the sense described above. Second, the features that describe the agreement and complexity of the hypotheses learned in each view can be seen as *classifier-based* meta-features: after all, they measure properties of the learned classifiers.

Chapter 7

Conclusions

A whole is what has beginning, middle, and end.

Aristotle

Machine learning algorithms start with collections of data from which they extract knowledge that is subsequently used to improve a system's performance. From a human perspective, labeling large amounts of data is a tedious, time consuming, error prone activity. In this thesis, I introduce a suite of algorithms that aim at reducing the amount of data required for learning a concept of interest. In particular, I focus on solving multi-view learning problems, in which the domain features can be split in several disjoint subsets, each of which is sufficient for learning.

My dissertation spans three major areas of research: active learning, semi-supervised learning, and meta-learning. In active learning, one minimizes the amount of labeled data by asking the user to label only the most informative examples in the domain. In contrast, a semi-supervised learner compensates for the scarcity of labeled data by exploiting a (large) set of unlabeled examples. Finally, meta-learning algorithms predict which is the most appropriate learner for a particular learning task, thus implicitly minimizing the amount of data that must be labeled (even if another learner could solve the task equally well, it would require more data to actually do it).

In this thesis I make three main contributions:

1. I introduce Co-Testing, which is the first multi-view approach to active learning. I show that Co-Testing clearly outperforms existing single-view active learners.
2. I show that existing multi-view learners are not robust with respect to the violation of the multi-view assumptions. In order to cope with this problem, I introduce a robust multi-view learner, Co-EMT, which interleaves active and semi-supervised multi-view learning.
3. I introduce an approach to deciding whether or not multi-view learning is appropriate for a new, unseen learning task. My *adaptive view validation* algorithm is a meta-learner that uses past experiences to learn a classifier that predicts whether or not a multi-view learner is likely to outperform a single-view one.

These contributions, which are discussed in detail in the next section, provide evidence that the thesis of my dissertation is correct:

Multi-view active learning maximizes the accuracy of the learned hypotheses while minimizing the amount of labeled training data.

7.1 Main contributions

7.1.1 A multi-view approach to active learning

In this dissertation, I introduce Co-Testing, which is multi-view approach to active learning. Co-Testing is a family of active learners that select the next query among the contention points; that is, the unlabeled examples on which the hypotheses learned in the various views predict a different label. This querying strategy is based on the following intuition: if each view is sufficient for learning the target concept, the existence of contention points shows that the views still make mistakes. Consequently, labeling a contention point helps fixing mistakes in *at least* one of the views.

I also extend the idea of multi-view learning to learning from both strong and weak views. The former are typical views, which are sufficient to learn the concept of interest. The latter are views in which one can learn only concepts that are strictly more general/specific than the target concept. I empirically show that learning from strong and weak views outperforms both single-view learning and learning from strong views only.

In this thesis, I provide a formal analysis of why Co-Testing works. More precisely, I prove that, under certain assumptions, Co-Testing converges significantly faster than single-view learners. Furthermore, I show that this is not true of semi-supervised, multi-view learners such as Co-Training.

Finally, I provide extensive empirical evidence that Co-Testing outperforms state-of-the-art, single-view active learners. The empirical results cover a variety of base learners (i.e., STALKER, C4.5, Naive Bayes, and instance-based learning) and real-world domains (e.g., wrapper induction, Web page classification, advertisement removal, and discourse tree parsing).

7.1.2 Robust multi-view learning

As a part of my second main contribution, I provide empirical evidence that existing multi-view learners are not robust to the violation of the multi-view assumptions, namely that the views are compatible and uncorrelated. More precisely, I show that depending on which assumption is violated and by how much, existing multi-view learners outperform and are outperformed by each other in various regions of the `incompatibility - correlation` space. Such a behavior is clearly undesirable: when a user must solve a new, unseen task, it is unclear which of the existing algorithms should be used for learning.

I introduce Co-EMT, a new multi-view learner that is robust to the assumption violations. Co-EMT can be described as interleaving active and semi-supervised learning (i.e., Co-Testing and Co-EM, respectively). In other words, Co-EMT combines the best of both worlds. On one hand, its labeled training set consists of the most informative examples in the domain, which are selected by active learning (i.e., Co-Testing). On the other

hand, besides using the (small) training set, it also exploits a (large) set of unlabeled examples (i.e., Co-EM). I empirically show that Co-EMT clearly outperforms existing multi-view learners. Furthermore, I provide evidence that Co-EMT’s robustness comes from Co-Testing’s ability to query the most informative examples, which compensate for view correlation.

7.1.3 Multi-view *vs.* single-view model selection

My third main contribution consists of formalizing the view validation problem, in which one tries to predict whether or not multi-view learning is appropriate for solving new, unseen tasks. I introduce a meta-learning *view validation* algorithm, which uses experiences acquired while solving past learning tasks to learn a classifier that predicts whether or not the views are sufficiently compatible for multi-view learning.

View validation departs from the typical approach to meta-learning, in which one assumes that the datasets consist of (large) sets of labeled examples. More precisely, view validation is the only meta-learner that is designed for domains in which one has access to a small set of labeled data and a large set of unlabeled ones. Consequently, in order to cope with this difficulty, I introduce a new set of meta-features, which are appropriate for multi-view learning from labeled and unlabeled data.

7.2 Limitations

Despite their advantages, the algorithms introduced in this dissertation also have a few limitations:

- the multi-view learning framework assumes that the domain features can be split in several views that are sufficient for learning the target concept. In other words, if the user cannot provide at least two views, multi-view learning cannot be applied at all. There are two recent attempts to address this issue (Raskutti, Ferra, and Kowalczyk,

2002a; Nigam and Ghani, 2000), but they are specific to text classification, and it is unclear whether they can be applied to other types of domains.

- as the Co-Testing family of algorithms consists of a large number of active learners, it is still unclear which of the Co-Testing algorithms is the most appropriate for a particular learning task. I provided a few rules of thumb on how to decide which algorithm to use in a particular scenario, but more work is clearly required in this area.
- existing approaches to active learning, including Co-Testing, take a *myopic* approach to querying; that is, they select *one* informative query and then immediately re-induce the classifier. This is an extremely time consuming approach that is not appropriate for scenarios in which the base learner is slow relatively to the time constraints of the application (e.g., it may be unacceptable to have a user wait for 10 minutes between two consecutive queries).
- in its current version, the view validation algorithm must be applied separately to each multi-view problem of interest. If one is interested in a wide variety of multi-view problems, individually collecting view validation examples for each problem puts a serious burden on the user.

7.3 Future work

I intend to continue the work presented here along several main directions:

- first of all, I am interested in creating a *view detection* algorithm that partitions a domain's features in views that are adequate for multi-view learning. Given that such a partition must be created starting with a few labeled and many unlabeled examples, this is an extremely difficult problem. A possible first step would be to extend the "view generation" approach from (Raskutti, Ferra, and Kowalczyk, 2002a) to domains other than text classification. Alternatively, I could start by creating a

view detection algorithm for relational domains (Getoor et al., 2001a; Jensen and Neville, 2002), in which the search through the space of possible feature partitions can be guided by the domain structure (e.g., hyperlinks, hubs and authorities in Web-based domains (Getoor et al., 2001b), or abstracts, keywords, citing and cited papers in a bibliographic domain (Taskar, Segal, and Koller, 2001)).

- I plan to work on providing a better understanding of the scenarios for which the various Co-Testing algorithms are the best suited. One possible approach would be to modify the view validation algorithm so that it predicts the most appropriate Co-Testing algorithm for a new, unseen learning task.
- I intend to extend the Co-Testing framework in several directions. First, in order to reduce the computational costs, I am interested in Co-Testing algorithms that use a “look-ahead” approach to making several highly informative and diverse queries. Second, I plan to investigate the applicability of Co-Testing to regression problems, in which the learner predicts the value of a continuous variable. This scenario is extremely appropriate for Aggressive Co-Testing: the largest the difference of the values predicted by the two views, the biggest the mistake made by one of them. Finally, I intend to apply Co-Testing to semi-supervised clustering (Basu, Banerjee, and Mooney, 2002). In clustering, the goal is to use (unlabeled) examples to detect the domain structure; semi-supervised clustering algorithms exploit the availability of a few labeled examples to guide their search through the structure space. By using Co-Testing to select a highly-informative set of labeled examples, I expect to maximize the benefits of using labeled data.
- I intend to continue my work on view validation along three directions. First, I would like to find more meta-features (remember, view validation currently uses only seven meta-features). Second, I plan to further reduce the number of view validation examples required for training by using active and semi-supervised learning. Given

that in the view validation scenario one assumes that there is a large number problem instances to be solved, it makes perfect sense to ask user to label only the most informative instances, while also allowing the view validation algorithm to exploit the remaining, unlabeled view validation examples. Finally, I am interested in creating a new, general-purpose view validation algorithm that is trained only once and covers a variety of multi-view learning problems.

References

- Abe, Naoki and Hiroshi Mamitsuka. 1998. Query learning using boosting and bagging. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 1–10.
- Abney, Stephen. 2002. Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367.
- Aha, David. 1992a. Generalizing from case studies: A case study. In *Proceedings of the 9th International Conference on Machine Learning (ICML-92)*, pages 1–10.
- Aha, David. 1992b. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(1):267–287.
- Amoth, Thomas, Paul Cull, and Prasad Tadepalli. 1998. Exact learning of tree patterns from queries and counterexamples. In *Proceedings of the Conference on Computational Learning Theory (COLT-98)*, pages 175–186.
- Amoth, Thomas, Paul Cull, and Prasad Tadepalli. 1999. Exact learning of unordered tree patterns from queries. In *Proceedings of the Conference on Computational Learning Theory (COLT-99)*, pages 323–332.
- Angluin, Dana. 1982. A note on the number of queries needed to identify regular languages. *Information Control*, 51:76–87.
- Angluin, Dana. 1988. Queries and concept learning. *Machine Learning*, 2:319–342.
- Angluin, Dana. 1994. Exact learning of μ -DNF formulas with malicious membership queries. Technical Report YALEU/DCS/TR-1020, Yale University.
- Angluin, Dana and Martins Krikis. 1994. Malicious membership queries and exceptions. Technical Report YALEU/DCS/TR-1019, Yale University.
- Angluin, Dana, Martins Krikis, Robert Sloan, and Gyorgy Turan. 1997. Malicious omissions and errors in answers to membership queries. *Machine Learning*, 28:211–255.
- Angluin, Dana and Donna Slonim. 1991. Randomly fallible teachers: learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14(1):7–26.
- Argamon-Engelson, Shlomo and Ido Dagan. 1999. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360.
- Baluja, Shumeet. 1998. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Advances in Neural Information Processing Systems*, volume 11, pages 854–860.
- Basu, Sugato, Arindam Banerjee, and Raymond Mooney. 2002. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 19–27.

- Bauer, Eric and Ron Kohavi. 1999. Empirical comparison of boosting, bagging, and variants. *Machine Learning*, 36(1-2):105–139.
- Baum, E.B. 1991. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2:5–19.
- Baxter, Jonathan. 1996. Learning model bias. In *Advances in Neural Information Processing Systems*, volume 9, pages 169–175.
- Baxter, Jonathan. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7–39.
- Bay, Stephen and Michael Pazzani. 2000. Characterizing model errors and differences. In *Proceedings of the 17th International Conference of Machine Learning (ICML-2000)*, pages 49–56.
- Bennett, Kristin and Ayan Demiriz. 1998. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, volume 11, pages 368–374.
- Bennett, Kristin and Ayan Demiriz. 2000. Optimization approaches to semi-supervised learning. In M. C. Ferris, O. L. Mangasarian, and J. S. Pang, editors, *Applications and Algorithms of Complementarity*. Kluwer Academic Publishers.
- Bennett, Kristin, Ayan Demiriz, and Rich Maclin. 2002. Exploiting unlabeled data in ensemble methods. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Bensusan, Hilal. 1998. God doesn't always shave with occam's razor - learning when and how to prune. In *Proceedings of the 10th European Conference on Machine Learning*, pages 119–124, Berlin, Germany. Springer.
- Bensusan, Hilal. 1999. *Automatic bias learning: An inquiry into the inductive basis of induction*. Ph.D. thesis, School of Cognitive and Computing Sciences, University of Sussex.
- Bensusan, Hilal and Christophe Giraud-Carrier. 2000. Discovering task neighbourhoods through landmark learning performances. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 325–331, Heidelberg. Springer.
- Bensusan, Hilal, Christophe Giraud-Carrier, and Claire Kennedy. 2000. A higher-order approach to meta-learning. In *Proceedings of the ECML-2000 workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*.
- Bensusan, Hilal, Christophe Giraud-Carrier, and Bernhard Pfahringer. 2000. What works well tells us what works better. In *Proceedings of ICML'2000 workshop on What Works Well Where*, pages 1–8.

- Blum, Avrim, Prasad Chalasani, Sally Goldman, and Donna Slonim. 1998. Learning with unreliable boundary queries. *Journal of Computer and System Sciences*, 56(2):209–222.
- Blum, Avrim, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. 1994. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 253–262.
- Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1988 Conference on Computational Learning Theory (COLT-98)*, pages 92–100.
- Brazdil, Pavel, Joao Gama, and R. Henery. 1984. Characterizing the applicability of classification algorithms using meta level learning. In F. Bergadano and L. de Raedt, editors, *Machine Learning - ECML-94 (LNAI 784)*. Springer.
- Breiman, Leo. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.
- Brodley, Carla. 1995. Recursive automatic bias selection for classifier construction. *Machine Learning*, 20:63–94.
- Califf, Mary Elain and Raymond Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 328–334.
- Campbell, Colin, Nello Cristianini, and Alex Smola. 2000. Query learning with large margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 111–118.
- Caruana, Rich. 1995. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems*, volume 8, pages 657–664.
- Caruana, Rich. 1996. Algorithms and applications for multitask learning. In *Proceedings of the 13th International Conference on Machine Learning (ICML-96)*, pages 87–95.
- Caruana, Rich. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.
- Castelli, Vittorio and Thomas Cover. 1995. On the exponential value of labeled examples. *Pattern Recognition Letters*, 16(1):105–111.
- Castelli, Vittorio and Thomas Cover. 1996. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2101–2117.
- Chan, Philip and Salvatore Stolfo. 1997. On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems*, 8(1):5–28.

- Cohen, William. 1998. A web-based information system that reasons with structured collections of text. In *Proceedings of the Second International Conference on Autonomous Agents (AA-98)*, pages 400–407.
- Cohn, David, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15:201–221.
- Cohn, David, Zoubin Ghahramani, and Michael Jordan. 1996. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, volume 9, pages 705–712.
- Collins, Michael and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Empirical NLP and Very Large Corpora Conference*, pages 100–110.
- Cover, Thomas and Joy Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons.
- Cozman, Fabio and Ira Cohen. 2002. Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the 15th International FLAIRS Conference*.
- Dagan, Ido and Sean Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the 12th International Conference on Machine Learning*, pages 150–157.
- Dasgupta, Sanjoy, Michael Littman, and David McAllester. 2001. PAC generalization bounds for co-training. In *Neural Information Processing Systems*.
- Day, N. 1969. Estimating the components of a mixture of normal distributions. *Biometrika*, 56(3):463–474.
- de Sa, Virginia and Dana Ballard. 1998. Category learning from multi-modality. *Neural Computation*, 10(5):1097–1117.
- Dempster, A., N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–38.
- Engels, Robert and C. Theusinger. 1998. Using a data metric for preprocessing advice for data mining applications. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-98)*, pages 430–434.
- Evans, B. and D. Fisher. 1994. Overcoming process delays with decision trees induction. *IEEE Expert*, 9:60–66.
- Fayyad, Usama, Padhraic Smyth, N. Weir, and S. Djorgovski. 1995. Automated analysis and exploration of image databases: results, progress, and challenges. *Journal of Intelligent Information Systems*, 4:1–19.
- Fedorov, V. V. 1972. *Theory of optimal experiment*. Academic Press.

- Feng, J. P. 2000. Meta-CN4 for unknown values processing via combiner and stacked generalization. In I. Buha and A. Famili, editors, *KDD-2000 Workshop on Post-processing in machine learning and data mining*.
- Freund, Yoav, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168.
- Gama, Joao and Pavel Brazdil. 1995. Characterization of classification algorithms. In C. Pinto Ferreira and N. Mamede, editors, *Progress in Artificial Intelligence (LNAI 990)*. Springer.
- Gama, Joao and Pavel Brazdil. 2000. Cascade generalization. *Machine Learning*, 41(3):315–343.
- Ganesalingam, S. and Geoff McLachlan. 1969. The efficiency of a linear discriminant function based on unclassified initial examples. *Biometrika*, 65:658–662.
- Getoor, Lise, Nir Friedman, Daphne Koller, and Avi Pfeffer. 2001a. Learning probabilistic relational models. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*. Springer.
- Getoor, Lise, Eran Segal, Ben Taskar, and Daphne Koller. 2001b. Probabilistic models of text and link structure for hypertext classification. In *Proceedings of the IJCAI-2001 Workshop on Text Learning: Beyond Supervision*.
- Ghahramani, Zoubin and Michael Jordan. 1994. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems*, volume 7, pages 120–127.
- Ghani, Rayid. 2001. Combining labeled and unlabeled data for text classification with a large number of categories. In *Proceedings of IEEE Conference on Data Mining (ICDM-2001)*.
- Ghani, Rayid. 2002. Combining labeled and unlabeled data for multiclass text classification. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 187–194.
- Goldberg, Paul, Sally Goldman, and David Mathias. 1994. Learning unions of boxes with membership and equivalence queries. In *Proceedings of the Conference on Computational Learning Theory (COLT-94)*, pages 198–207.
- Goldman, Sally and David Mathias. 1992. Learning k -term DNF formulas with an incomplete membership oracle. In *Proceedings of the Conference on Computational Learning Theory (COLT-92)*, pages 77–84.
- Goldman, Sally and Yan Zhou. 2000. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 327–334.

- Gross, Klaus. 1991. *Concept acquisition through attribute evolution and experiment selection*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.
- Guilfoyle, C. 1986. Ten minutes to lay the foundations. *Expert Systems User*, August:16–19.
- Hartley, H. and J. Rao. 1968. Classification and estimation in analysis of variance problems. *Review of International Statistical Institute*, 36:141–147.
- Hasenjager, Martina. 2000. *Active Data Selection in Supervised and Unsupervised Learning*. Ph.D. thesis, Faculty of Technology, University of Bielefeld, Germany.
- Hasenjager, Martina and Helge Ritter. 1996. Active learning of the generalized high-low-game. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-96)*, volume 1112 of *Lecture Notes in Computer Science*, pages 501–506.
- Hasenjager, Martina and Helge Ritter. 1998. Active learning with local models. *Neural Processing Letters*, 7:107–117.
- Hsu, Chun-Nan and Ming-Tzung Dung. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Journal of Information Systems*, 23(8):521–538.
- Hwang, J.-N., J.J. Choi, S. Oh, and R.J. Marks. 1991. Query-based learning applied to partially trained multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2:131–136.
- Jackson, Jeffrey. 1994. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 42–53.
- Jensen, David and Jennifer Neville. 2002. Schemas and models. In *Proceedings of the SIGKDD-2002 Workshop on Multi-relational Learning*.
- Joachims, Thorsten. 1996. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Computer Science Tech. Report CMU-CS-96-118*.
- Joachims, Thorsten. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pages 200–209.
- Kamolvilassatian, Noppadon. 2002. Property-based feature engineering and selection. Master’s thesis, Department of Computer Sciences, University of Texas at Austin.
- Kaynak, C. and E. Alpaydin. 2000. Multistage cascading of multiple classifiers: One man’s noise is another man’s data. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 455–462.
- King, R., C. Feng, and A. Shutherland. 1995. STATLOG: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):259–287.

- Kirk, T., A. Levy, Y. Sagiv, and D. Srivastava. 1995. The information manifold. In *Proceedings of the AAAI Spring Symposium: Information Gathering from Heterogeneous Distributed Environments*, pages 85–91.
- Knoblock, Craig, Kristina Lerman, Steven Minton, and Ion Muslea. 2002. Accurately and reliably extracting data from the web: a machine learning approach. In P. Szczepaniak, editor, *Intelligent exploration of the Web*. Springer.
- Knoblock, Craig, Steven Minton, Jose-Luis Ambite, Naveen Ashish, Ion Muslea, and Andrew Philpot. 2001. The Ariadne approach to Web-based Information Integration. *International Journal of Cooperative Information Sources*, 10(1/2):145–169.
- Koepf, C., C. Taylor, and J. Keller. 2000. Meta-analysis: From data characterisation for meta-learning to meta-regression. In P. Brazdil and A. Jorge, editors, *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support, Meta-Learning and ILP*.
- Kohavi, Ron and George John. 1995. Automatic parameter selection by minimizing estimated error. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pages 304–312.
- Kohavi, Ron, Dan Sommerfield, and James Dougherty. 1997. Data mining using MLC++, a machine learning library in C++. *International Journal of AI Tools*, 6(4):537–566.
- Kushmerick, Nicholas. 1999. Learning to remove internet advertisements. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, pages 175–181.
- Kushmerick, Nicholas. 2000. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence Journal*, 118(1-2):15–68.
- Kushmerick, Nicholas, Edward Johnston, and Stephen McGuinness. 2001. Information extraction by text classification. In *The IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*.
- Lang, K.J. and E.B. Baum. 1992. Query learning can work poorly when a human oracle is used. In *Proceedings of the IEEE International Joint Conference on Neural Networks*.
- Lerman, Kristina and Steven Minton. 2000. Learning the common structure of data. In *The 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 609–614.
- Lewis, David and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, pages 148–156.
- Lewis, David and William Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of Research and Development in Information Retrieval*, pages 3–12.
- Liere, Ray and Prasad Tadepalli. 1997. Active learning with committees for text categorization. In *The 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 591–596.

- Lindenbaum, Michael, Shaul Markovitch, and Dmitry Rusakov. 1999. Selective sampling for nearest neighbor classifiers. In *Proceedings of the 15th National Conference on Artificial Intelligence AAAI-99*, pages 366–371.
- Lindner, Guido and Rudi Studer. 1999. AST: Support for algorithm selection with a CBR approach. In *Principles of Data Mining and Knowledge Discovery*, pages 418–423.
- MacLachlan, Geoff. 1975. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70:365–369.
- Marcu, Daniel, Lynn Carlson, and Maki Watanabe. 2000. The automatic translation of discourse structures. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*.
- McCallum, Andrew and Kamal Nigam. 1998a. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*.
- McCallum, Andrew and Kamal Nigam. 1998b. Employing em in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 359–367.
- Merialdo, Bernardo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 2(2):155–171.
- Michie, Donald. 1989. Problems of computer-aided concept formation. In J.R. Quinlan, editor, *Applications of expert systems (vol. 2)*. Addison-Wesley.
- Miller, David and Hasan Uyar. 1996. A generalized gaussian mixture classifier for learning on both labeled and unlabeled data. In *Proceedings of the 1996 Conference on Information Science and Systems*.
- Miller, David and Hasan Uyar. 1997. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing*, volume 10, pages 571–577.
- Mitchell, Tom. 1998. *Machine Learning*. McGraw-Hill.
- Muslea, Ion, Steven Minton, and Craig Knoblock. 2000a. Selective sampling with Naive Co-Testing. In *The ECAI-2000 Workshop on Machine Learning for Information Extraction*.
- Muslea, Ion, Steven Minton, and Craig Knoblock. 2000b. Selective sampling with redundant views. In *Proceedings of National Conference on Artificial Intelligence (AAAI-2000)*, pages 621–626.
- Muslea, Ion, Steven Minton, and Craig Knoblock. 2001. Hierarchical wrapper induction for semistructured sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114.

- Muslea, Ion, Steven Minton, and Craig Knoblock. 2002a. Active + Semi-supervised Learning = Robust Multi-view Learning. In *The 19th International Conference on Machine Learning (ICML-2002)*, pages 435–442.
- Muslea, Ion, Steven Minton, and Craig Knoblock. 2002b. Adaptive view validation: A first step towards automatic view detection. In *The 19th International Conference on Machine Learning (ICML-2002)*, pages 443–450.
- Nahm, Un-Yong and Raymond Mooney. 2000. A mutually beneficial integration of data mining and information extraction. In *The 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 627–632.
- Nigam, Kamal and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of Information and Knowledge Management*, pages 86–93.
- Nigam, Kamal, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 792–799.
- Nigam, Kamal, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134.
- O’Neill, T. 1978. Normal discrimination with unclassified observations. *Journal of the American Statistical Association*, 73:821–826.
- Petrak, J. 2000. Fast subsampling performance estimates for classification algorithms selection. In *Proceedings of the ECML-2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*.
- Pfahringer, Bernhard, Hilal Bensusan, and Christophe Giraud-Carrier. 2000. Meta-learning by landmarking various learning algorithms. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 743–750.
- Pierce, David and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-2001)*, pages 1–10.
- Quinlan, Ross. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers.
- Raskutti, Bhavani, Herman Ferra, and Adam Kowalczyk. 2002a. Combining clustering and co-training to enhance text classification using unlabeled data. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Raskutti, Bhavani, Herman Ferra, and Adam Kowalczyk. 2002b. Using unlabeled data for text classification through addition of cluster parameters. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 514–521.

- Ratsaby, Joel and Santosh Venkatesh. 1995. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 412–417.
- Reddy, C. and Prasad Tadepalli. 1997. Learning horn definitions with equivalence and membership queries. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 243–255. Springer.
- Rendell, L., R. Seshu, and D. Tcheng. 1987. Layered concept learning and dynamically variable bias management. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 308–314.
- Roy, Nicholas and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 441–448.
- Sammut, Claude and R. B. Banerji. 1986. Learning concepts by asking questions. In R. S. Michalski Carbonell, J.G. Carbonell, and T. M. Mitchell (Vol. 2), editors, *Machine Learning: An Artificial Intelligence Approach*, pages 167–192. Morgan Kaufmann.
- Sarkar, Anoop. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pages 175–182.
- Schaffer, Cullen. 1993a. Overfitting avoidance as bias. *Machine Learning*, 10:153–178.
- Schaffer, Cullen. 1993b. Selecting a classification method by cross-validation. *Machine Learning*, 13(1):135–143.
- Schaffer, Cullen. 1994. A conservation law for generalization of performance. In *Proceedings of the 11th International Conference of Machine Learning (ICML-94)*, pages 259–265.
- Schapire, Robert. 1990. The strength of weak learnability. *Machine Learning*, 5(2):197–227.
- Scheffer, Tobias and Stefan Wrobel. 2001. Active learning of partially hidden Markov models. In *Proceedings of the ECML/PKDD-2001 Workshop on Active Learning, Database Sampling, Experimental Design: Views on Instance Selection*.
- Schohn, Greg and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 839–846.
- Seung, H. Sebastian, Manfred Opper, and Haim Sompolinski. 1992. Query by committee. In *Proceedings of the 1992 Conference on Computational Learning Theory (COLT-92)*, pages 287–294.

- Shahshahani, Behzad and David Landgrebe. 1994. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32(5):1087–1095.
- Shapiro, E. 1981. A general incremental algorithm that infers theories from facts. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 446–451.
- Shapiro, E. 1982. Algorithmic program diagnosis. In *Proceedings of the 9th ACM Symposium on Principles of Programming Languages*, pages 299–308.
- Shavlik, Jude, Raymond Mooney, and George Towell. 1991. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6(2):111–143.
- Sloan, Robert and Gyorgy Turan. 1994. Learning with queries but incomplete information (extended abstract). In *Proceedings of the Conference on Computational Learning Theory (COLT-94)*, pages 237–245.
- Soares, Carlos and Pavel Brazdil. 2000. Zoomed ranking: Selection of classification algorithms based on relevant performance information. In *Proceedings of the 13th Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*.
- Soderland, Stephen. 1999. Learning extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272.
- Szummer, Martin and Tommi Jaakkola. 2000. Kernel expansions with unlabeled examples. In T. Leen, T. Dietterich, and V. Tresp, editors, *Neural Information Processing Systems*, volume 13, pages 626–632.
- Tadepalli, Prasad. 1993. Learning from queries and examples with tree-structured bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML-93)*, pages 322–329.
- Tadepalli, Prasad and Stuart Russell. 1998. Learning from queries and examples with structured determinations. *Machine Learning*, pages 245–295.
- Taskar, Ben, Eran Segal, and Daphne Koller. 2001. Probabilistic classification and clustering in relational data. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 870–876.
- Thompson, Cynthia, Mary Elaine Califf, and Raymond Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pages 406–414.
- Thrun, Sebastian and Lorien Pratt, editors. 1997. *Learning to learn*. Kluwer Academic Publishers.
- Ting, Kai Ming and Ian Witten. 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289.

- Todorovski, Ljupco and Saso Dzeroski. 1999. Experiments in meta-level learning with ILP. In J. M. Zytkow and J. Rauch, editors, *Proceedings of the 3rd European Conference on Principles of data mining and knowledge discovery (PKDD-99)*, pages 98–106. Springer.
- Todorovski, Ljupco and Saso Dzeroski. 2000. Combining multiple models with meta decision trees. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 54–64. Springer.
- Tong, Simon and Daphne Koller. 2000a. Active learning for parameter estimation in bayesian networks. In *Advances in Neural Information Processing Systems*, volume 13, pages 647–653.
- Tong, Simon and Daphne Koller. 2000b. Support vector machine active learning with applications to text classification. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 999–1006.
- Tong, Simon and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.
- Valiant, Leslie. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.
- Vapnik, Vladimir. 1998. *Statistical learning theory*. John Wiley and Sons.
- Watkin, T. and A. Rau. 1992. Selecting examples for perceptrons. *Journal of Physics A: Mathematical and General*, 25(1):113–121.
- Widmer, Gerhard. 1997. Tracking context changes through meta-learning. *Machine Learning*, 27(3):259–286.
- Wolpert, David. 1992. Stacked generalization. *Neural Networks*, 5:241–259.
- Wolpert, David. 1996. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 7.
- Wolpert, David and William Macready. 1997. No free lunch theorems for search. *IEEE Transactions on Evolutionary Computation*, 1.
- Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting of the Association of Computational Linguistics*, pages 189–196.
- Zelikovitz, Sara and Haym Hirsh. 2000. Improving short text classification using unlabeled background knowledge. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 1183–1190.
- Zhang, Tong and Frank Oles. 2000. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 1191–1198.

Appendix A

Proofs of convergence

If we take in our hand any volume; of divinity or school metaphysics, for instance; let us ask,

“Does it contain any abstract reasoning concerning quantity or number?” No.

“Does it contain any experimental reasoning concerning matter of fact and existence?” No.

Commit it then to the flames: for it can contain nothing but sophistry and illusion.

David Hume

In this appendix I prove the convergence properties presented in section 3.3. Remember that, for both 1-DHL and 2-DHL, the target concepts are integer threshold values, while the (real-valued) attributes are uniformly distributed over the interval $[1, H]$. In the proofs below I assume that the algorithms have access to sufficiently many labeled and unlabeled examples to unambiguously learn the target concepts. Intuitively, this means that for all intervals $[i, i + 1]$, where $i \in \{1, 2, 3, \dots, H - 1\}$, the training and/or the working set include at least one example that takes a value in that interval.

A.1 Convergence properties for single-view algorithms

I begin by studying the ability of single-view algorithms to learn target concepts of the form

$$f : [1, H] \mapsto \{0, 1\}, \quad f(X) = \begin{cases} l_1 & \text{if } X \leq w_{TC} \\ l_2 & \text{if } X > w_{TC} \end{cases}$$

where $w_{TC} \in \{1, 2, 3, \dots, H\}$, $l_1, l_2 \in \{0, 1\}$, and $l_1 \neq l_2$. In other words, the goal is to learn the threshold value w_{TC} that divides the interval $[1, H]$ in two sub-intervals in which the examples have the same label (i.e., l_1 or l_2).

Note that the restriction $w_{TC} \in \{1, 2, 3, \dots, H\}$ makes the problem’s hypotheses space finite. Furthermore, for any training set, the problem’s version space consists of the subset $\{Min, Min + 1, Min + 2, \dots, Max\}$ of the hypotheses space. To keep the notation simple, throughout this appendix I use the following convention: a version space that consists of the N hypotheses $\{Min, Min + 1, Min + 2, \dots, Min + N - 1\}$ is represented by the set $\{1, 2, 3, \dots, N\}$, in which “1” denotes the first hypothesis in version space (i.e., $W = Min$), “2” denotes the second one (i.e., $W = Min + 1$), and so on.

Proposition 1 *On the 1-DHL problem, the Uncertainty Sampling algorithm requires $O(\log(H))$ queries to learn the target concept.*

A.1.0.0.1 Proof: As already mentioned, Uncertainty Sampling is a 2-step iterative process: first, it learns a hypothesis from the given training set; then it queries the unlabeled example on which this hypothesis makes the least confident prediction. When using the Gibbs algorithm as base learner, the learned hypothesis is randomly chosen from the current version space; that is, the decision threshold W is set to a value k , which is randomly chosen from the set $\{1, 2, 3, \dots, N\}$. The least confident prediction of the hypothesis $W = k$ is made on the example $X = k$, which is the closest to the current decision border (in fact, it is *on* the decision border).

Consequently, Uncertainty Sampling can be seen as randomly querying an example $X = k$, where k belongs to the current version space (i.e., $k \in \{1, 2, 3, \dots, N\}$). Note that depending on the label of the example $X = k$, this query removes either $N - k + 1$ or $k - 1$ hypotheses from version space. I use the notation $VSCut_{w_{TC} \in \{1, 2, \dots, k-1\}}$ and $VSCut_{w_{TC} \in \{k, k+1, \dots, N\}}$ to denote these two possible cuts from the version space. One can compute the *expected cut* that a particular query $X = k$ makes to the version space:

$$\begin{aligned}
E[Cut(X = k)] &= Probability(w_{TC} \in \{1, 2, \dots, k-1\}) \cdot VSCut_{w_{TC} \in \{1, 2, \dots, k-1\}} + \\
&\quad Probability(w_{TC} \in \{k, k+1, \dots, N\}) \cdot VSCut_{w_{TC} \in \{k, k+1, \dots, N\}} \\
&= \frac{k-1}{N} \cdot (N-k+1) + \frac{N-k+1}{N} \cdot (k-1) \\
&= \frac{2 \cdot (k-1) \cdot (N-k+1)}{N}
\end{aligned}$$

Now we can compute the *expected cut of the actual query* to the version space by averaging the expected cuts of each possible query:

$$\begin{aligned}
E[Cut] &= \sum_{i=1}^N Probability(X = i) \cdot E[Cut(X = i)] \\
&= \sum_{i=1}^N \frac{1}{N} \cdot \frac{2 \cdot (i-1) \cdot (N-i+1)}{N} \\
&= \frac{2}{N^2} \cdot \sum_{i=1}^N (i-1) \cdot (N-i+1) \\
&= \frac{2}{N^2} \cdot \sum_{i=1}^N (N \cdot i - i^2 + i - N + i - 1) \\
&= \frac{2}{N^2} \cdot \sum_{i=1}^N ((N+2) \cdot i - i^2 - (N+1)) \\
&= \frac{2}{N^2} \cdot \left((N+2) \cdot \frac{N \cdot (N+1)}{2} - \frac{(N-1) \cdot N \cdot (2 \cdot N - 1)}{6} - N \cdot (N+1) \right) \\
&= \frac{1}{3 \cdot N} (3 \cdot (N+1) \cdot (N+2) - (N-1) \cdot (2 \cdot N - 1) - 6 \cdot (N+1)) \\
&= \frac{1}{3 \cdot N} (3 \cdot N^2 + 9 \cdot N + 6 - 2 \cdot N^2 + 3 \cdot N - 1 - 6 \cdot N - 6) \\
&= \frac{N^2 + 6 \cdot N - 1}{3 \cdot N} \\
&= \frac{N}{3} + 2 - \frac{1}{3 \cdot N}
\end{aligned}$$

In other words, each query made by Uncertainty Sampling is expected to remove approximately $\frac{1}{3}$ of the current version space. Consequently, Uncertainty Sampling converges to the target concept in $\mathcal{O}(\log(H))$ queries, where H is the size of the initial version space. \square

Proposition 2 *On the 1-DHL problem, with an arbitrarily high probability, the Query-by-Committee algorithm requires $\mathcal{O}(\log(H))$ queries to learn the target concept.*

A.1.0.0.2 Proof: In order to prove this proposition, I use the main result from (Freund et al., 1997): that paper’s **Theorem 1** shows that, with an arbitrarily high probability, Query-by-Committee’s error rate is reduced exponentially with the number of queries. For **Theorem 1** to hold, there are three main requirements. First, the learning concept must be perfectly learnable, which - by construction - is true of 1-DHL. Second, the learning problem must have a finite VC-dimension; as 1-DHL has a finite version space, it follows that its VC-dimension is also finite (see (Mitchell, 1998, page 215) for details). Finally, the *expected information gain* of any query made by Query-by-Committee must have a finite lower bound $g > 0$. In other words, $\exists g > 0$ such that for any query q_i

$$g < -p_0 \times \log(p_0) - (1 - p_0) \times \log(1 - p_0)$$

where p_0 is the probability that the label of the example q_i is “0.” Given the structure of the 1-DHL problem (i.e., the target concept is one of the H threshold values that are uniformly distributed over the initial version space $\{1, 2, 3, \dots, H\}$), it follows that for any example $x \in [1, H]$

$$\begin{aligned} p_0 &= \text{Probability}(f(x) = 0) \\ &\geq \frac{1}{H} \end{aligned}$$

thus ensuring that the *expected information gain* is lower bounded by the finite, positive value

$$\begin{aligned} g &= -\frac{1}{N} \times \log\left(\frac{1}{N}\right) - \left(1 - \frac{1}{N}\right) \times \log\left(1 - \frac{1}{N}\right) \\ &= \frac{\log N}{N} + \frac{(N-1) \times \log\left(\frac{N}{N-1}\right)}{N} \end{aligned}$$

As all the requirements of **Theorem 1** from (Freund et al., 1997) are fulfilled, it follows that Query-by-Committee solves the 1-DHL learning problem in $\mathcal{O}(\log(H))$ queries, where H is the size of the initial version space. \square

Proposition 3 *On the 1-DHL problem, the probability that Random Sampling correctly learns the target concept based on E randomly chosen examples is $1 - \frac{2 \times (H-2)^E - (H-3)^E}{(H-1)^E}$.*

A.1.0.0.3 Proof: Given that the Gibbs learner randomly chooses a hypothesis from version space, Random Sampling correctly learns the target concept *if and only if* the version space consists of a single hypothesis (i.e., the true threshold value w_{TC}). For this to happen, the randomly chosen training set must contain *at least* two examples X_1 and X_2 such that $X_1 \in (w_{TC} - 1, w_{TC}]$ and $X_2 \in (w_{TC}, w_{TC} + 1]$. Consequently, the probability that Random Sampling correctly learns the target concept is equal to the probability of having the randomly chosen training set contain at least two such examples.

In this proof, I use the following notation:

- L represents the algorithm's (randomly chosen) training set;
- E denotes the number of examples in L ;
- H represents the size of the initial version space;
- A denotes the statement " $\exists X_1 \in L$ such that $X_1 \in (w_{TC} - 1, w_{TC}]$ "
- B denotes the statement " $\exists X_2 \in L$ such that $X_2 \in (w_{TC}, w_{TC} + 1]$ "

- $A \cdot B$, \overline{A} , and \overline{B} denote the expressions “ A and B ”, “not A ”, and “not B ”, respectively.

The probability that Random Sampling correctly learns the target concept from E randomly chosen labeled examples is

$$\begin{aligned}
P(A \cdot B) &= 1 - P(\overline{A \cdot B}) \\
&= 1 - (P(\overline{A}) + P(\overline{B}) - P(\overline{A} \cdot \overline{B})) \\
&= 1 - \left(\left(\frac{H-2}{H-1} \right)^E + \left(\frac{H-2}{H-1} \right)^E - \left(\frac{H-3}{H-1} \right)^E \right) \\
&= 1 - 2 \times \left(\frac{H-2}{H-1} \right)^E + \left(\frac{H-3}{H-1} \right)^E \\
&= 1 - \frac{2 \times (H-2)^E - (H-3)^E}{(H-1)^E} \quad \square
\end{aligned}$$

A.2 Convergence properties for multi-view algorithms

In this section I analyze the ability of multi-view learners to correctly learn the target concept for an arbitrary 2-DHL problem. First, I consider the scenario in which “the views are independent given the label.” Then I relax this initial assumption and analyze the situation in which “the views are independent given the clump.”

Throughout this section, I depict the 2-dimensional instance space $[1, H_1] \times [1, H_2]$ as an orthogonal system of coordinates XOY . The view $\mathbf{V1}$ is described by the values of X on the OX axis, while the view $\mathbf{V2}$ corresponds to the OY dimension. In all the figures within this section, I use the following color coding: the regions of the instance space that are drawn in light and dark grey denote the positive and negative examples, respectively; the areas depicted in the intermediate shade of grey represent contention points.

A.2.1 Views that are independent given the label

I begin my analysis by studying the ability of multi-view algorithms to learn a target concepts of the form

$$g : [1, w_{TC}^1] \times [1, w_{TC}^2] \cup (w_{TC}^1, H_1] \times (w_{TC}^2, H_2] \mapsto \{0, 1\},$$

$$g(X, Y) = \begin{cases} l_1 & \text{if } X \leq w_{TC}^1 \text{ and } Y \leq w_{TC}^2 \\ l_2 & \text{if } X > w_{TC}^1 \text{ and } Y > w_{TC}^2 \end{cases}$$

where $w_{TC}^1 \in \{1, 2, 3, \dots, H_1\}$, $w_{TC}^2 \in \{1, 2, 3, \dots, H_2\}$, $l_1, l_2 \in \{0, 1\}$, and $l_1 \neq l_2$. In the 2-view setting, this translates into solving a 1-DHL problem in each view. Consequently, the target concepts in the views **V1** and **V2** are

$$g_1 : [1, H_1] \mapsto \{0, 1\}, \quad g_1(X) = \begin{cases} l_1 & \text{if } X \leq w_{TC}^1 \\ l_2 & \text{if } X > w_{TC}^1 \end{cases}$$

and

$$g_2 : [1, H_2] \mapsto \{0, 1\}, \quad g_2(Y) = \begin{cases} l_1 & \text{if } Y \leq w_{TC}^2 \\ l_2 & \text{if } Y > w_{TC}^2 \end{cases}$$

respectively.

Proposition 4 *By using domain-specific knowledge, one can solve the 2-DHL problem based on a single, randomly-chosen query.*

A.2.1.0.4 Proof: By construction, we have the following properties:

- the examples in 2-DHL's instance space occupy only the bottom-left and top-right quadrants;
- each quadrant consists of examples that have the same label (see Figure A.1);

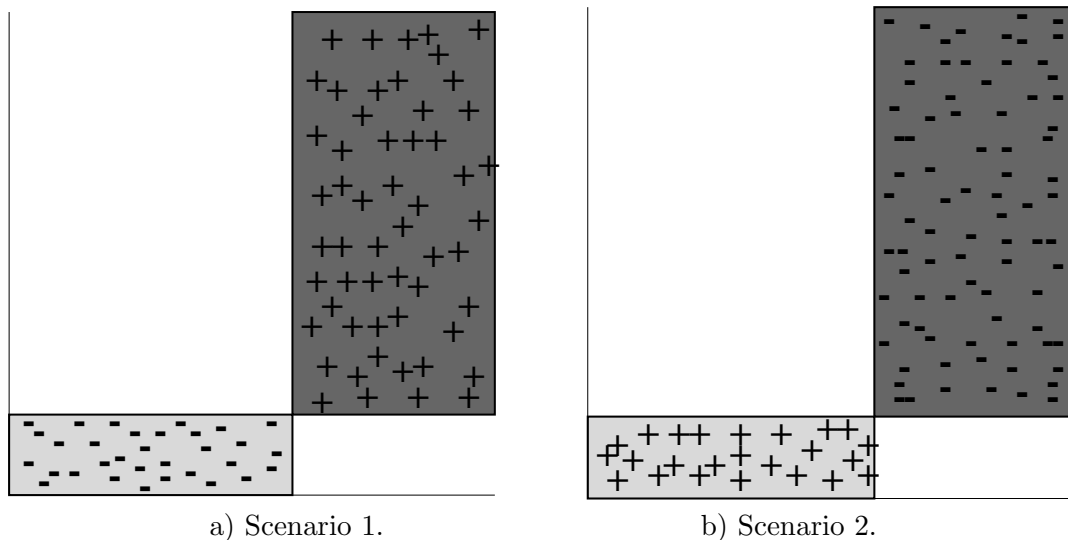


Figure A.1: The two possible scenarios in 2-DHL with “views independent given the label:” the negative examples lie either in the bottom-left or in the top-right quadrant. The positive examples occupy the other populated quadrant.

Furthermore, because of the view independence assumption, the two populated quadrants must consist of examples having *different labels*.¹ Consequently, querying a randomly chosen example is sufficient for learning the target concept: all the examples that are in the same quadrant as the query share its label, while the other examples have the opposite label. \square

Proposition 5 *On the 2-DHL problem, the Co-Training algorithm requires one random positive and one random negative examples to learn the target concept.*

A.2.1.0.5 Proof: As described earlier, Co-Training (Blum and Mitchell, 1998) bootstraps the views from each other by iteratively adding to the training set examples on which the hypotheses learned in the two views make the most confident predictions. In the 2-DHL problem, these high confidence predictions correspond to examples that are the farthest away from the decision threshold in each view.

¹In the degenerate scenarios in which all examples are either positive or negative, the views are not independent: knowing an example’s description in one view unambiguously identifies the example’s quadrant, while knowing the example’s label still allows the example to lie in either of the two quadrants.

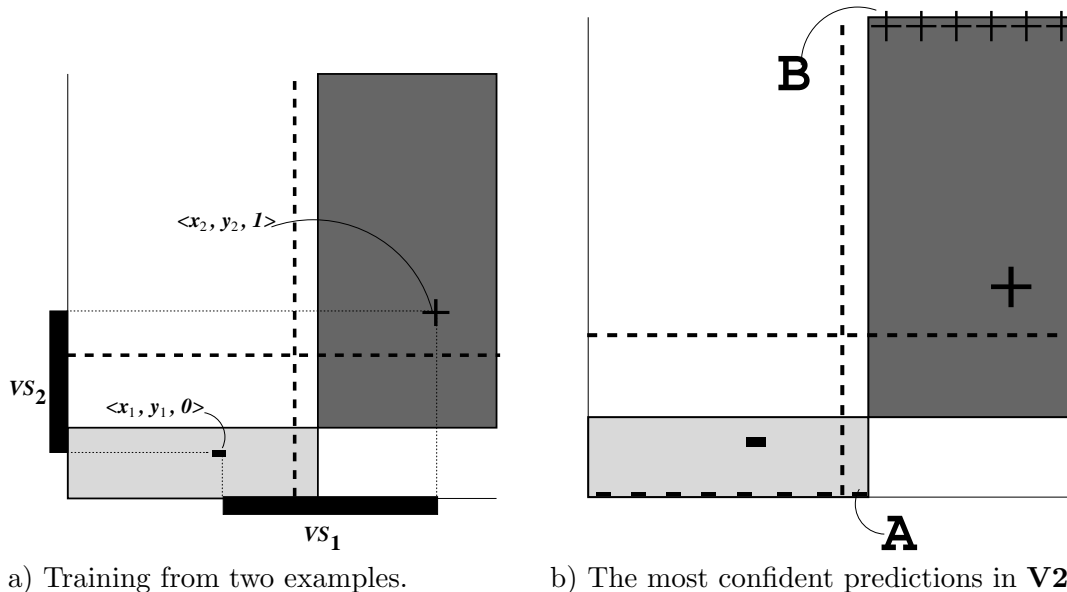


Figure A.2: Applying Co-Training to the a 2-DHL task in which the views are independent given the label (i.e., there is one clump per class).

Consider the scenario described in Figure A.2.a, in which Co-Training is provided with the randomly chosen examples $\langle x_1, y_1, 0 \rangle$ and $\langle x_2, y_2, 1 \rangle$ (remember that the labels “1” and “0” denote a positive and negative example, respectively). In the first step, Co-Training learns a hypothesis in each view by running the Gibbs learner, which randomly returns a hypothesis from the version space. In Figure A.2.a, the two version spaces VS_1 and VS_2 , which are denoted by the black rectangles by the axes of coordinates, consist of the all integers in the intervals $[x_1, x_2]$ and $[y_1, y_2]$, respectively. The two dashed lines represent the hypotheses returned by the Gibbs sampler in each view.

Once the two hypotheses are learned, Co-Training applies them to the unlabeled examples and adds to the training set the most confident predictions. Figure A.2.b depicts the unlabeled examples on which the hypothesis learned in the “vertical view” V_2 makes the most confident negative and positive predictions, respectively. The former are located on the X axis, while the latter are located on the top border of the image. Among these newly labeled examples, there are two highly informative examples, which are denoted by A and B; after adding these two examples to training set, the version space in the

“horizontal view” $\mathbf{V1}$ collapses to the true target concept. A similar situation occurs in the “vertical view” after adding to the training set the high confidence predictions made by the hypothesis learned in the “horizontal view.” \square

Proposition 6 *On the 2-DHL problem, when provided with one random positive and one random negative examples, Aggressive Co-Testing requires at most **four** queries to learn the target concepts in both views.*

A.2.1.0.6 Proof: This proposition is a particular case of the **Proposition 9** below, which shows that Aggressive Co-Testing requires at most $2 \times NmbClumps$ queries to learn the target concepts in both views. For 2-DHL with views that are independent given the label, we have one clump per class, for a total of two domain clumps. Consequently, in $2 \times 2 = 4$ queries, Aggressive Co-Testing collapses the version spaces in both views to the respective target concepts. \square

A.2.2 Views that are “independent given the clump”

In this section, I relax the “views are independent given the label” assumption by considering the scenario in which the “views are independent given the clump.” That is, rather than having just one clump per class, I consider now problems with several clumps per class; more precisely, I assume that the total number of clumps in the domain is $NmbClumps > 2$. Figure A.3 illustrates a scenario in which there are five negative and four positive clumps (each positive and negative clump is denoted by a dark or light grey rectangle, respectively).

More formally, I study the convergence properties of multi-view algorithms when learning target concepts of the form

$$g : [1, \alpha_1] \times [1, \beta_1] \cup (\alpha_1, \alpha_2] \times (\beta_1, \beta_2] \cup \dots (\alpha_{C-1}, \alpha_C] \times (\beta_{C-1}, \beta_C] \mapsto \{0, 1\},$$

$$g(X, Y) = \begin{cases} l_1 & \text{if } X \leq w_{TC}^1 \text{ and } Y \leq w_{TC}^2 \\ l_2 & \text{if } X > w_{TC}^1 \text{ and } Y > w_{TC}^2 \end{cases}$$

where

- $w_{TC}^1 \in \{1, 2, 3, \dots, H_1\}$;
- $w_{TC}^2 \in \{1, 2, 3, \dots, H_2\}$;
- $l_1, l_2 \in \{0, 1\}$ and $l_1 \neq l_2$;
- $C = NmbClumps$;
- $\alpha_C = H_1$ and $\beta_C = H_2$;
- $\forall i \in \{1, 2, 3, \dots, C-1\} \alpha_i \in \{2, 3, \dots, H_1-1\}$;
- $\forall i \in \{1, 2, 3, \dots, C-1\} \alpha_i < \alpha_{i+1}$;
- $\forall i \in \{1, 2, 3, \dots, C-1\} \beta_i \in \{2, 3, \dots, H_2-1\}$;
- $\forall i \in \{1, 2, 3, \dots, C-1\} \beta_i < \beta_{i+1}$;
- $\exists k \in \{2, 3, 4, \dots, C-1\}$ such that $\alpha_k = w_{TC}^1$ & $\beta_k = w_{TC}^2$.

In the 2-view setting, this translates into solving a 1-DHL problem in each view. Consequently, the target concepts in the views **V1** and **V2** are

$$g_1 : [1, H_1] \mapsto \{0, 1\}, \quad g_1(X) = \begin{cases} l_1 & \text{if } X \leq w_{TC}^1 \\ l_2 & \text{if } X > w_{TC}^1 \end{cases}$$

and

$$g_2 : [1, H_2] \mapsto \{0, 1\}, \quad g_2(Y) = \begin{cases} l_1 & \text{if } Y \leq w_{TC}^2 \\ l_2 & \text{if } Y > w_{TC}^2 \end{cases}$$

respectively.

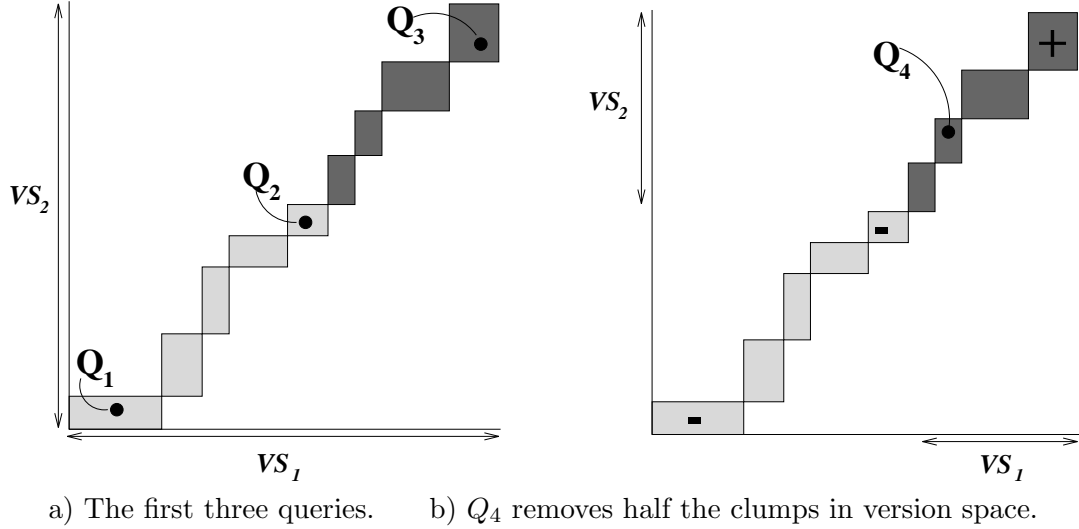


Figure A.3: By using domain-specific knowledge, one can perform binary search in the space of the domain’s clumps.

Proposition 7 *By using domain-specific knowledge, one can solve the 2-DHL problem based on $\mathcal{O}(\log(NmbClumps))$ queries.*

A.2.2.0.7 Proof: By explicitly taking advantage of the domain structure in 2-DHL, one can solve the learning task in a manner similar to binary search. In a first step, the algorithm makes the queries Q_1 , Q_2 , and Q_3 , which are randomly chosen examples from the three clumps shown in Figure A.3.a; that is, Q_1 , Q_2 , and Q_3 “reveal” the labels of the examples in the left-most, right-most, and “middle” clump, respectively. Once the labels of these clumps are known, the algorithm keeps “halving the interval” by querying a randomly chosen example from the clump that is in the middle of the current version space; in our example, this corresponds to the query Q_4 from Figure A.3.b. By continuing this procedure until discovering the two neighboring clumps that are labeled differently, the algorithm is guaranteed to find the correct target concept. In keeping with the results for binary search, the entire process requires $\mathcal{O}(\log(NmbClumps))$ queries. \square

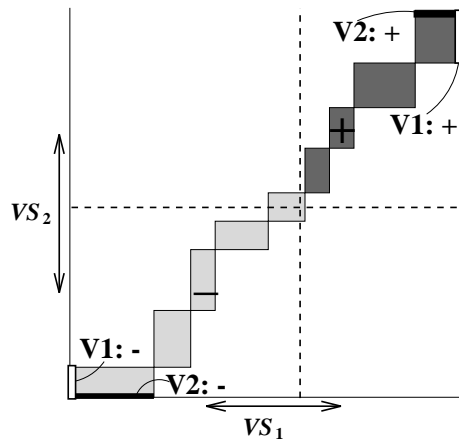
Proposition 8 *Depending on the distribution of the examples in the initial training set, Co-Training may or may not learn the target concept for the 2-DHL problem.*

A.2.2.0.8 Proof: As mentioned earlier, Co-Training works as follows: first, it uses the labeled examples to learn one hypothesis in each view. Then it applies the learned hypotheses to all unlabeled examples and adds to the training set the examples on which they make most confident prediction. Finally, it repeats the whole process for a number of iterations.

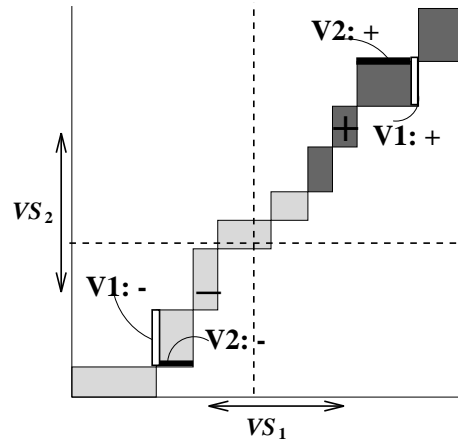
Consider now applying Co-Training to the domain in Figure A.4.a. The two labeled examples determine the version spaces VS_1 and VS_2 , from which the Gibbs learner randomly chooses the two hypotheses (i.e., the dashed lines in the picture). The most confident positive and negative predictions of the hypothesis learned in the view **V1** are the unlabeled examples that are the farthest away from the decision border; that is, the examples that lie on the left border of the left-most clump and on the right border of the right-most clump, respectively. These examples are depicted in Figure A.4.a as the thin, white rectangles labeled “**V1: +**” and “**V1: -**”, respectively. Similarly, the two thin black rectangle on the bottom of the left-most clump and on the top of the right-most clump represent the most confident predictions of the “horizontal” hypothesis learned in **V2**.

After a number of iterations, Co-Training adds to the training set all the examples from the left-most and right-most clump. At this point, *the new most confident predictions* are the ones on the borders of the second clumps from left and right, respectively (see Figure A.4.b). After some more iterations, all the examples from these two new clumps are also added to the training set, and Co-Training starts making *the most confident predictions* in the third clump from the left and right, respectively (see Figure A.4.c).

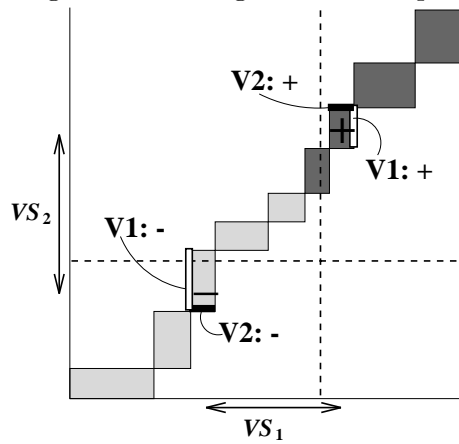
Note that these two clumps are the last ones which are guaranteed to be correctly labeled by Co-Training: from now on, Co-Training starts labeling clumps for which it has no other information except for their proximity to the previously labeled clumps. The illustrative domain in Figure A.4 was chosen so that Co-Training happens to also correctly label the fourth clump from the left and right, respectively (see Figure A.4.d). However, when reaching the remaining unlabeled clump (i.e., the fifth one from both left and right),



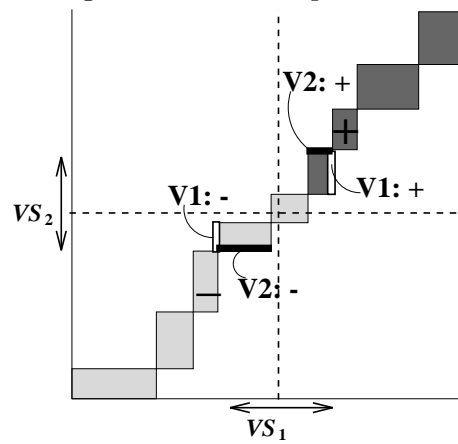
a) Labeling the left- & right-most clumps.



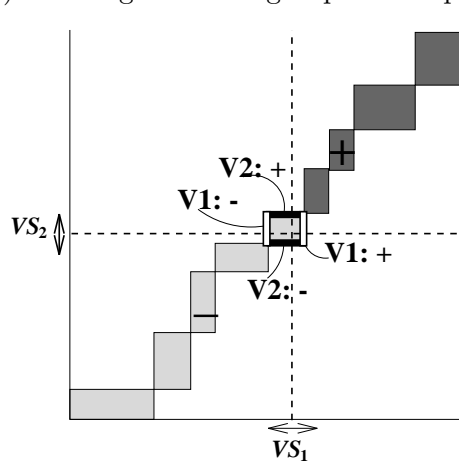
b) Labeling the second clump from left & right.



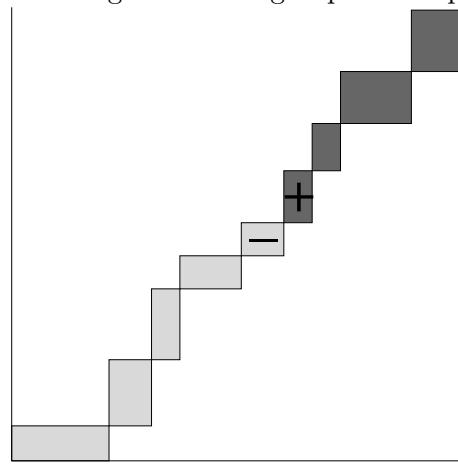
c) Labeling the third group of clumps.



d) Labeling the fourth group of clumps.



e) Inconsistently labeling the last clump.



f) Necessary condition for convergence.

Figure A.4: Co-Training bootstraps the views from each other by adding the most confident predictions to the training set. Unless the initial, randomly-chosen training set has a favorable distribution, Co-Training does not converge to the target concept.

both views label the clump inconsistently (see Figure A.4.e). For example, the view **V1** labels the left and right border of this clump as being negative and positive respectively, which is impossible: the examples within the same clump must have the same label. A similar situation appears in the view **V2**, which predicts positive and negative labels for the examples on the top and bottom borders of this clump, respectively.

In practice, the problem of conflicting labels is not the only way in which Co-Training may fail to learn the target concept. For example, when Co-Training labeled the fourth clump from the left and right (see Figure A.4.d), it was by pure chance that the clumps were labeled correctly. For a different target concept, one of the two clumps would have had a different label, which would have implied that one of the views provided the other one with mislabeled examples; in turn, this would have prevented the learning of the correct target concepts.

Given the problems described above, it is easy to see that Co-Training is guaranteed to correctly learn the target concept *if and only if* its initial training set contains a positive and a negative example from the two clumps that define the border between the two classes (see Figure A.4.f). In other words, one can not guarantee that, given an arbitrary training set, Co-Training converges to the target concept. \square

Proposition 9 *On the 2-DHL problem, with an arbitrarily-high probability, Aggressive Co-Testing learns the target concept in both views by making at most $2 \times NmbClumps$ queries, where $NmbClumps$ is the number of clumps in the domain.*

In order to show that this proposition is true, I first prove two auxiliary results that are used in the proof of **Proposition 9**:

A.2.2.0.9 Statement 1: if the version spaces in each view have not collapsed to a single hypothesis each (i.e., if they did *not* converge to the respective target concepts), with an arbitrarily-high probability, Co-Testing’s set of contention points is not empty.

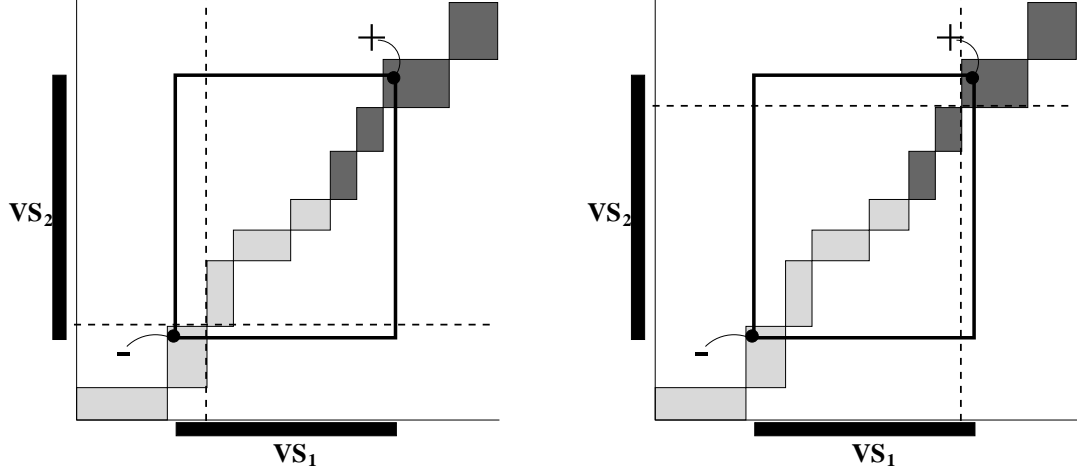


Figure A.5: Two illustrative scenarios in which there are no contention points. In both scenarios, $NmbClumpsInVS = 5$ because only five clumps are entirely within the two dimensional version space.

A.2.2.0.10 Statement 2: Aggressive Co-Testing queries at most two examples from each clump in the domain.

A.2.2.0.11 Proof of Statement 1: By construction, two hypotheses learned in different views fail to generate contention points *if and only if* they intersect at one of the points where two neighboring clumps touch each other (see Figure A.5). Given that the Gibbs algorithm randomly selects the hypotheses that are uniformly distributed over the two version spaces, it follows that the probability of learning two hypotheses that have no contention points is

$$\frac{(NmbClumpsInVS + 1)}{Size(VS_1)} \times \frac{(NmbClumpsInVS + 1)}{Size(VS_2)}$$

where VS_1 and VS_2 represent the version spaces in each view, and $NmbClumpsInVS$ denotes the number of clumps that *entirely lie* within the current 2-dimensional version space (i.e., the rectangular area defined by the version spaces in each view). Figure A.5 depicts two of the six possible scenarios in which $NmbClumpsInVS = 5$ and there are no contention points.

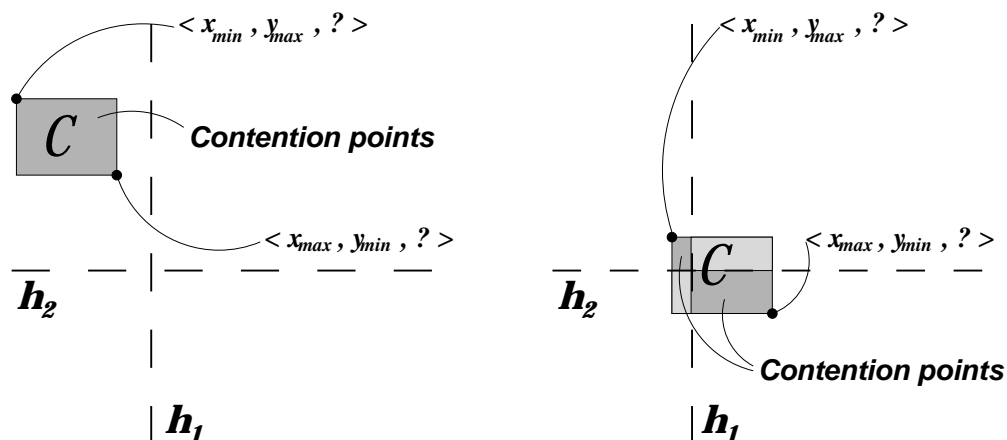


Figure A.6: In the figure on the left, the entire clump consists of contention points. In the left-most figure, the two hypotheses intersect within the clump, splitting in into quadrants; only the clump's top-left and bottom right quadrants consist of contention points.

Let us assume that even though the two version spaces did not collapse to a single hypothesis each, the two hypotheses returned by the Gibbs algorithm do not have any contention points. Then Co-Testing can re-run the Gibbs learner until it generates two hypotheses that have a non-empty set of contention points. The probability of having a non-empty set of contention points after R re-runs of the Gibbs learner in each view is

$$\left(1 - \frac{(NmbClumpsInVS + 1)}{Size(VS_1)} \times \frac{(NmbClumpsInVS + 1)}{Size(VS_2)}\right)^R$$

Consequently, by choosing an appropriately large value of R , one can guarantee that - with an arbitrarily-high probability - Co-Testing obtains a non-empty set of contention points. \square

A.2.2.0.12 Proof of Statement 2: In order to prove this statement, I proceed as follows. First, I show that if Aggressive Co-Testing queries two examples from a particular clump \mathcal{C} , then these two queries are the examples on the top-left and bottom-right corners of the rectangle that borders the clump \mathcal{C} . Then I show that once these two queries are

made, no other example from the clump \mathcal{C} can be among the contention points, thus making it impossible to query more than two examples from \mathcal{C} .

Consider an arbitrary pair of hypotheses h_1 and h_2 that are learned in the two views. Let us denote by w_1 and w_2 the decision thresholds corresponding to h_1 and h_2 , respectively. The hypotheses h_1 and h_2 divide the instance space into quadrants. All the contention points lie in the top-left and bottom-right quadrants (i.e., $X \leq w_1$ & $Y > w_2$, and $X > w_1$ & $Y \leq w_2$, respectively), for which the hypotheses predict a different label. Figure A.6 depicts two illustrative scenarios in which either *all* or *some* of the examples within a clump \mathcal{C} are among the contention points.

Let us denote by $\langle x_{min}, y_{max}, ? \rangle$ and $\langle x_{max}, y_{min}, ? \rangle$ the top-left and bottom-right corner of the clump \mathcal{C} , respectively. Then the following three properties are true:

- **Property 1:** If neither $\langle x_{min}, y_{max}, ? \rangle$ nor $\langle x_{max}, y_{min}, ? \rangle$ are contention points, then none of the examples within \mathcal{C} are among the contention points.

Proof: If neither $\langle x_{min}, y_{max}, ? \rangle$ nor $\langle x_{max}, y_{min}, ? \rangle$ are among the contention points, it follows that **either** $x_{min}, x_{max} \leq w_1$ and $y_{min}, y_{max} \leq w_2$ **or** $x_{min}, x_{max} > w_1$ and $y_{min}, y_{max} > w_2$. It follows that for an arbitrary example $\langle x, y, ? \rangle$ from the clump \mathcal{C} (i.e., $x \in [x_{min}, x_{max}]$ and $y \in [y_{min}, y_{max}]$), we have **either** $x \leq w_1$ and $y \leq w_2$ **or** $x > w_1$ and $y > w_2$. Consequently, $\langle x, y, ? \rangle$ is labeled identically by the decision thresholds in both views, thus making it impossible to be among the contention points. \square

- **Property 2:** If the clump \mathcal{C} includes contention points located within the top-left quadrant (i.e., $X \leq w_1$ & $Y > w_2$), then $\langle x_{min}, y_{max}, ? \rangle$ is one of them. Furthermore, among these “top-left” contention points from the clump \mathcal{C} , $\langle x_{min}, y_{max}, ? \rangle$ is the most confident prediction made by both hypotheses h_1 and h_2 .

Proof: Consider an arbitrary example $\langle x, y, ? \rangle$ from clump \mathcal{C} (i.e., $x \in [x_{min}, x_{max}]$ and $y \in [y_{min}, y_{max}]$). Let us assume that $\langle x, y, ? \rangle$ is a contention point located in the top-left quadrant; that is, $x \leq w_1$ and $y > w_2$. As $x \geq x_{min}$ and $y \leq y_{max}$, it

follows that $x_{min} \leq x \leq w_1$ and $y_{max} \geq y > w_2$. Consequently, $\langle x_{min}, y_{max}, ? \rangle$ is also a contention point located in the top-left quadrant. Furthermore, among the other examples in \mathcal{C} that lie in this quadrant, $\langle x_{min}, y_{max}, ? \rangle$ is the farthest away from both decision borders because it maximizes both $w_1 - X$ and $Y - w_2$. \square

- **Property 3:** If the clump \mathcal{C} includes contention points located within the bottom-right quadrant (i.e., $X > w_1$ & $Y \leq w_2$), then $\langle x_{max}, y_{min}, ? \rangle$ is one of them. Furthermore, among these “bottom-right” contention points from the clump \mathcal{C} , $\langle x_{max}, y_{min}, ? \rangle$ is the most confident prediction made by both hypotheses h_1 and h_2 .

Proof: Consider an arbitrary example $\langle x, y, ? \rangle$ from clump \mathcal{C} (i.e., $x \in [x_{min}, x_{max}]$ and $y \in [y_{min}, y_{max}]$). Let us assume that $\langle x, y, ? \rangle$ is a contention point located in the bottom-right quadrant; that is, $x \geq w_1$ and $y < w_2$. As $x \leq x_{max}$ and $y \geq y_{min}$, it follows that $x_{max} \geq x \geq w_1$ and $y_{min} \leq y < w_2$. Consequently, the $\langle x_{max}, y_{min}, ? \rangle$ is also a contention point located in the bottom-right quadrant. Furthermore, among the other examples in \mathcal{C} that lie in this quadrant, $\langle x_{max}, y_{min}, ? \rangle$ is the farthest away from both decision borders because it maximizes both $X - w_1$ and $w_2 - Y$. \square

From **Properties 2 & 3**, it follows that if Aggressive Co-Testing queries examples from the clump \mathcal{C} , the first such query is either $\langle x_{min}, y_{max}, ? \rangle$ or $\langle x_{max}, y_{min}, ? \rangle$. Without loss of generality, let us assume that Aggressive Co-Testing first queries the example $\langle x_{min}, y_{max}, ? \rangle$; an argument similar to the one below holds if Co-Testing first queries $\langle x_{max}, y_{min}, ? \rangle$.

Once the query $\langle x_{min}, y_{max}, ? \rangle$ is labeled “ l ” and Co-Testing re-trains, the new hypotheses h_1 and h_2 correctly classify the example $\langle x_{min}, y_{max}, l \rangle$ (remember that we assumed perfect learning in both views). This means that the example $\langle x_{min}, y_{max}, l \rangle$ is not a contention point; in turn, based on **Property 2**, this implies that there are no contention points from \mathcal{C} in the top-left quadrant.

The argument above implies that after querying $\langle x_{min}, y_{max}, ? \rangle$ and re-training, there are only two possibilities: either there are no more contention points in the clump \mathcal{C} , or all contention points from \mathcal{C} lie in the bottom-right quadrant. In the former scenario, Co-Testing ends up making just a single query within the clump \mathcal{C} . In the latter scenario, Co-Testing may or may not make another query in the clump \mathcal{C} , depending on how confident are the predictions made on contention points from the various clumps. However, if Co-Testing makes another query within the clump \mathcal{C} , according to **Property 3**, this query is guaranteed to be $\langle x_{max}, y_{min}, ? \rangle$.

In this second scenario, the new query $\langle x_{max}, y_{min}, ? \rangle$ is also labeled “ l ” (remember that all examples within a clump must have the same label). After adding $\langle x_{max}, y_{min}, l \rangle$ to the training set and re-training one more time, the new hypotheses h_1 and h_2 label correctly both $\langle x_{max}, y_{min}, l \rangle$ and $\langle x_{min}, y_{max}, l \rangle$. This means that neither $\langle x_{max}, y_{min}, l \rangle$ nor $\langle x_{min}, y_{max}, l \rangle$ are contention points; consequently, according to **Property 1**, none of the examples within the clump \mathcal{C} can be among the contention points. In turn, this means that Aggressive Co-Testing can make at most two queries in any clump \mathcal{C} . \square

A.2.2.0.13 Proof of Proposition 9: Based on **Statements 1** and **2**, the proof of **Proposition 9** becomes straightforward. Aggressive Co-Testing starts with two randomly chosen examples (one positive and one negative) and learns a hypothesis in each view. **Statement 1** guarantees that unless the learning process converged to the target concepts in both views, with an arbitrarily-high probability, the two learned hypotheses lead to a non-empty set of contention points, thus allowing Co-Testing to make new queries.

In keeping with **Statement 2**, Aggressive Co-Testing makes at most two queries per clump, for a total of $2 \times NmbClumps$ queries. After this number of queries, the label of each clump is unambiguously determined, thus collapsing both version spaces to a single hypothesis, which represents the target concept. \square

Appendix B

The 60 Semi-Artificial Problems

No man's knowledge here can go beyond his experience.

John Locke

To create a parameterized family of problems in which we control the view correlation and incompatibility, we start from an idea presented in (Nigam and Ghani, 2000). One can create a (semi-artificial) domain with compatible, uncorrelated views by taking two *unrelated* binary classification problems and considering each problem as an individual view. The multi-view examples are created by randomly pairing examples that have the same label in the original problems.

The procedure above can be easily modified to introduce both clumps and incompatible examples. For instance, consider creating a binary classification problem in which the positive examples consist of two clumps. We begin with *four* unrelated problems that have the sets of positive examples A , B , C , and D , respectively. In the newly created 2-view problem, the positive examples in the views $\mathbf{V1}$ and $\mathbf{V2}$ consist of the $A \cup B$ and $C \cup D$, respectively. As shown in the left-most graph in Figure B.1, if the multi-view examples are created by randomly pairing an example from $A \cup B$ with one from $C \cup D$, we obtain, again, uncorrelated views. By contrast, if we allow the examples from A to be paired only with the ones from C , and the ones from B with the ones from D , we obtain a

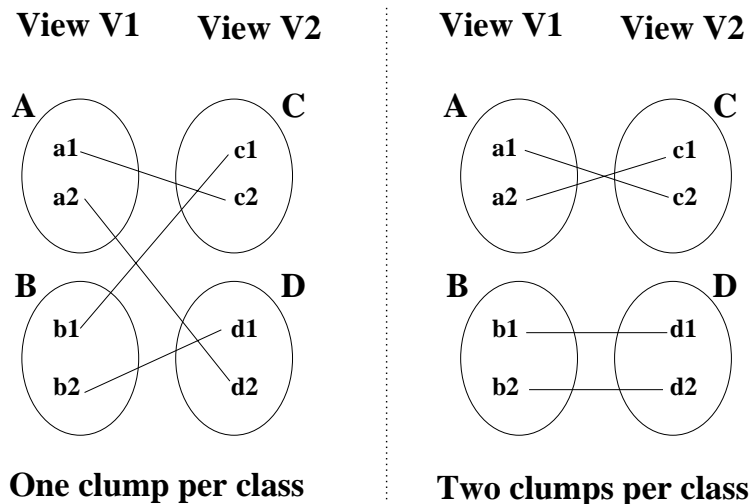


Figure B.1: Generating one and two clumps per class.

problem with two clumps of positive examples: $A-C$ and $B-D$. Similarly, based on eight or 16 unrelated problems, one can create four or eight clumps per class, respectively.

Adding incompatible examples is a straightforward task: first, we randomly pick one positive and one negative multi-view example, say [“Advanced OS”, AdvOS-Class] and [“John Doe”, JohnDoe-Homepage]. Then we replace these two examples by their “recombinations”, [“Advanced OS”, JohnDoe-Homepage] and [“Joe Doe”, AdvOS-Class], which are positive in one view and negative in the other one.

In order to generate problems with up to four clumps per class, we used 16 of 20 newsgroups postings from the Mini-Newsgroups dataset,¹ which is a subset of the well-known 20-Newsgroups domain (Joachims, 1996). Each newsgroup consists of 100 articles that were randomly chosen from the 1000 postings included in the original dataset. We divided the 16 newsgroups in four groups of four (see Table B.1). The examples in each such group are used as either positive or negative examples in one of the two views; i.e., the newsgroups `comp.os.ms-win`, `comp.sys.ibm`, `comp.windows.x`, and `comp.sys.mac` play the roles of the A , B , C , and D sets of examples from Figure B.1.

¹http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes/mini_newsgroups.tar.gz

	V1	V2
pos	comp.os.ms-win.misc comp.sys.ibm.pc.hrwd rec.autos rec.sport.baseball	comp.windows.x comp.sys.mac.hrwd rec.motorcycles rec.sport.hockey
neg	sci.crypt sci.space talk.politics.guns talk.politics.misc	sci.electronics sci.med talk.politics.mideast talk.religion.misc

Table B.1: The 16 newsgroups included in the domain.

We begin by creating *compatible* views with three levels of clumpiness: one, two, and four clumps per class. Figure B.2 shows how the three levels of clumpiness are created for the positive examples. For one clump per class, any positive example from **V1** can be paired with any positive example in **V2**. For two clumps per class, we do *not* allow the pairing of `comp` examples in one view and the `rec` examples in the other one. Finally, for four clumps per class we pair examples from `comp.os.ms-win` and `comp.windows.x`, from `comp.sys.ibm` and `comp.sys.mac`, etc.

For each level of clumpiness, we consider with five levels of view incompatibility: 0%, 10%, 20%, 30%, and 40% of the examples are incompatible, respectively. This corresponds to a total of 15 points in the `correlation - incompatibility` space; as we already mentioned, for each such point we generate four random problems, for a total of 60 problems (each problem consists of 800 examples).²

²The documents are tokenized, the UseNet headers are discarded, words on a stoplist are removed, no stemming is performed, and words that appear only in a single document are removed. The resulting views **V1** and **V2** have 5061 and 5385 features (i.e., words), respectively.

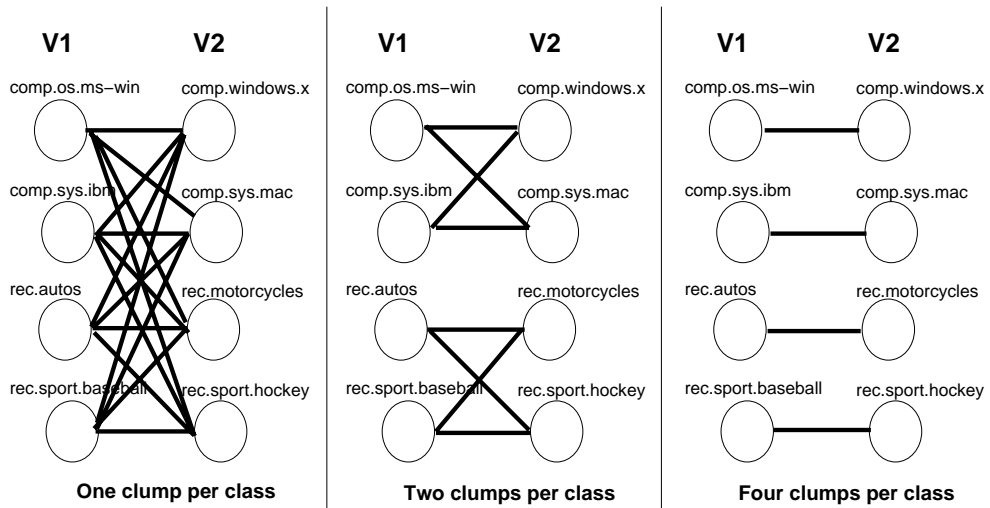


Figure B.2: Generating up to four clumps per class.