EXPLANATION-BASED APPROACH TO INCORPORATING DOMAIN KNOWLEDGE INTO SUPPORT VECTOR MACHINE: THEORY AND APPLICATIONS

BY

QIANG SUN

B.S., Peking University, 1996
M.A., Wesleyan University, 1999

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

ABSTRACT

EXPLANATION-BASED APPROACH TO INCORPORATING DOMAIN KNOWLEDGE INTO SUPPORT VECTOR MACHINE: THEORY AND APPLICATIONS

Qiang Sun, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 2005
Professor Gerald DeJong, Advisor

We believe one of the most promising but under-explored research areas in machine learning today is the integration of prior domain knowledge into the learning process. For a learning system, both training examples and domain knowledge provide information about the particular concept to be acquired. While most of the research works aim at exploring new algorithms and techniques for extracting information from a training set, how to use prior domain knowledge is largely ignored.

Such ignorance is often not due to the unimportance of the domain knowledge for a learning task. It is well known that a learning algorithm requires inductive bias in order to generalize beyond the training examples. It is desirable to use domain knowledge to introduce domain-specific bias into learning systems. The fact that using prior domain knowledge in machine learning is under-explored is largely a result of difficulties of utilizing information in the domain knowledge.

The first difficulty is that prior knowledge is usually domain-specific, which makes difficult to generalize the algorithms that aims at utilizing the knowledge. Moreover, domain knowledge is usually expressed in experts' high-level vocabulary, which is often different from the vocabulary used to describe the examples. Finally, domain knowledge could be only approximate and imprecise, therefore directly incorporate such knowledge into inductive learning system could introduce harmful bias.

This research aims at avoiding or alleviating the above problems by using Explanation Based Learning (EBL) to mediate between the evidence in prior knowledge and the evidence in the training examples. Conventional EBL (DeJong, 1997) uses domain knowledge to explain the training examples, and generalize the explanations to obtain some deeper patterns, which, if believed, commits the learner to assigning classification labels to many unseen examples. In this work, we introduce a new learning framework, where those patterns obtained by EBL are used to introduce further inductive bias into a learning system. In this framework, EBL can be viewed as a mechanism to transform high-level domain-specific knowledge into special solution knowledge, which can then be used to introduce inductive bias into learning systems.

We implemented our proposed explanation-based learning framework with three different approaches: phantom example approach, feature kernel approach and explanation-augmented SVM approach. In these approaches, we choose Support Vector Machines (SVM) as the inductive learner to demonstrate how domain knowledge can improve the performance of a learning system. SVM is a relatively new and successful approach to classification learning. It is still a challenging problem to incorporate knowledge into SVMs. In this work, we present both theoretical and empirical results to show that our approaches use domain knowledge to improve SVM's performance.

The most novel aspect of our work is that EBL procedure encourages interactions between prior knowledge and the training examples. This allows our techniques to utilized information in the domain knowledge, which is otherwise difficult to incorporate into SVMs. Moreover, the inductive bias introduced into SVMs is calibrated for the given examples distribution, which potentially makes our approach more robust.

We also present the comparison of the three proposed approaches, discuss about the related work, and point out some future work. We believe this work provide a first step towards a new research area in machine learning.

ACKNOWLEDGEMENTS

This thesis work could not have been done without the help of my mentor, coworker, family and friends. I would like to first thank my research advisor, Dr. Gerald DeJong for his continuous encouragement and guidance throughout the years. This work was very much an interactive process in which his insights, suggestions, and contributions were crucially important. I benefited greatly from his experience as a teacher and research advisor. I especially would like to thank him for encouraging me to work independently. This not only allowed me to learn how to solve problems, but also gave me valuable lessons as to how to pursue new scientific projects. His expertise, his enthusiasm toward research and his work ethics set a great example for me. I am very fortunate to have worked with such as insightful and helpful advisor over the past few years.

During the years I worked in Dr. DeJong's lab, I was very lucky to get to know and learn from a number of talented students and coworkers. I am also thankful to all the members of EBL group over the time I have been here: including Mark Brodie, Top Changwatchai, Valentin Moskovich, Adam Laud, Mike Cibulski, Arkady Epshteyn , Scott Duran, Shiau-Hong Lim, Geoffrey Levine. Each person has helped me view my research in a new light through the course of our many meetings. I enjoyed countless times of the discussions with them. I also believe that those happy times with them in the office and the tennis court will become happy memory in the future.

I would also like to thank several scientists outside Dr. DeJong's lab who helped me with my research. Dr. Dan Roth gave me valuable suggestions and helpful comments for my research. Dr. Jean Ponce and Dr. David Kriegman answered my many questions about computer visions. I want to thank Dr. Thomas Huang for their suggestions on my thesis.

I thank my parents for their endless love and support to me. They taught me to keep positive attitude toward life and never be afraid of hardships. They encouraged me to be independent, to believe in myself and to reach my full potential. I thank my dear wife Qi Meng. She has been very supportive through all the years I was in graduate school. Her presence in my life often makes the difficult time disappear. I enjoy countless discussion about the science with her. Although her major is not computer science, her opinions often provide motivation on my research. Her patience and encouragement helped me greatly in difficult times. I couldn't have achieved the same without her love and help.

Another important person in my graduate study is my daughter Annabel. She gave me countless happy time, and valuable memories. It is amazing that such a small baby can teach me so much about life. Watching her growing up makes me realize that life has so much more to enjoy, to experience, and to learn. I only wish my research would someday make contribution to something that can match her amazing learning ability.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

One of the most fortunate situations a scientist can encounter is to enter a field in its infancy. - Bernhard Schölkopf [2002]

It has long been one of man's greatest dreams to engineer machines possessing artificial intelligence. Short after the introduction of the first computer, many researchers have been enthusiastically pursuing to build machines with ability to perform intelligent tasks such as vision, speech recognition, making automatic and reasonable decision in complex environment, and so on.

After many years of hard work, much has been achieved, yet much more need to be done. One of the first and promising steps toward artificial intelligence is machine learning. Today, many state-of-the-art learning techniques have been successfully applied to solve real world problems. The recent success of machine learning is mainly the result of rapid development in two important research fields: information industry and computation theory of learning. The first field provides abundant data for learning algorithms to work with. The second field formalizes the learning problem and enables many advanced mathematic techniques to be applied to the problem.

On the other hands, the influence of information industry and computation theory on machine learning is not perfect. The availability of large amount of data leads research on machine learning to the direction focusing on statistical learning algorithms, which aim at extracting patterns from data. Similarly, the formalization of statistical learning problem by computation theory incidentally shifts the research interest away from those problems that are not easy to fit into the current formalism. The result of such influence is that the research focus on machine learning becomes narrower. Once popular and promising research topics, such as case-based learning and explanation-based learning faded away, if not totally vanished.

One purpose of this research is to make it convincing that it is time for some old, yet well-motivated wisdom to resurrect. In particular, the idea behind Explanation-Based Learning (EBL) is to use the training examples to select and organize related knowledge to form a concept useful for classification purpose. We believe that such idea compensates well with the state-of-the-art statistical learning techniques, such as Support Vector Machines (SVMs).

To illustrate our point, we consider the problem of recognizing handwritten digits. This is a relatively easy vision problem for humans. Yet, computers, although are able to beat the humans on playing chess, still cannot achieve the same level as humans to recognize handwritten digits. Should we blame the scarceness of the scarceness of the data? No, there exists a database with 6,000 training examples for each digit. It is reasonable to believe such amount of data is more than enough for humans to learn the ability of recognizing handwritten digits. Is that because our learning algorithm is not advanced enough to extract patterns from the data? To answer this question, consider a related problem, where the pixels in the handwritten digit images are permutated in a fixed manner. Figure 1.1 shows the resulting images resulting from a random permutation together with original images. Clearly the problem becomes much harder for humans. Yet a SVM learning algorithm operates on pixel level is still able to extract pixel-level

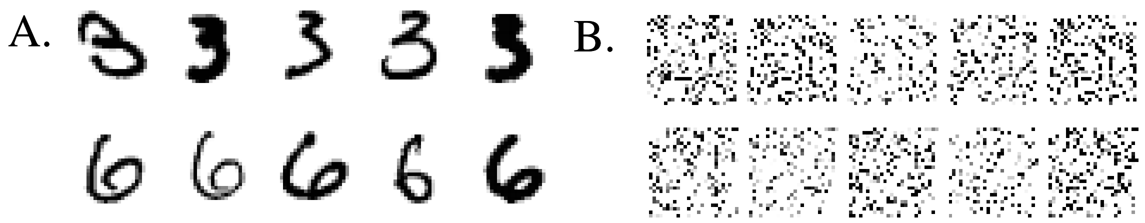

Figure 1.1 Two classification problems. A. Handwritten digit characters. B. Images generated by permutating pixels from images in A in a random, but fixed manner.

patterns from the permutated images. Those patterns that exist in the original images are still in the permutated images, although they are shifted in some mysterious manner. A statistical learner that considers all those patterns, such as an SVM learning algorithm with polynomial kernel, is designed to detect patterns solely from data. Therefore it can somewhat amazingly achieve same performance on these two seemly quite different recognition problems.

The above example illustrates the power of the state-of-the-art machine learning techniques on extracting patterns from data. What missing for SVMs for the handwritten digit recognition problem is the awareness of the *strokes*. Humans compare the similarity between handwritten digit images in the stroke-level, where the difference between digits is easier to detect. While SVM learning algorithms only see pixels. The useful pixel-level patterns, although are indeed present in the images, are buried in the sea of irrelevant features. Ignoring the concept of strokes makes the SVM learning algorithm to consume unnecessarily large amount of data to re-discover the regularities present in the strokes.

Our intuition and our analysis express the desire to incorporate prior knowledge about the problem, such as the concept of strokes, into statistical learning algorithms. There has been much work done to pursue this direction [Cohen, 1992; Chown &

Dietterich 2000; DeCoste & Schölkopf, 2002; Fung etc. 2003, Schölkopf, 1996; Schölkopf etc. 1998; Simard etc. 1998; Simard & Lecun, 1994; Simard etc. 1992; Teow & Loe, 2000]. While certain knowledge is ready to be used to introduce bias into the learning algorithm, some high-level knowledge, such as the concept of strokes, does not fit into learner's vocabulary. Therefore, those learning algorithms that operate on low-level vocabulary, such as neural networks or support vector machines, often ignore knowledge expressed at high-level vocabulary. On the other hand, it is often also awkward to represent examples in high-level vocabulary, because extracting high-level features, such as strokes, is usually a challenging problem. The errors introduced during feature extraction process can make the problem unnecessarily difficult. Therefore, although our intuition tells us that any knowledge about the learning problem is helpful, it is still not clear how to incorporate high-level knowledge into statistical learning algorithm.

In this work, we use Explanation-Based-Learning (EBL) [DeJong 1997; Mitchell etc. 2004, DeJong & Mooney, 1990] approach to mediate high-level knowledge with learner's low-level bias vocabulary. EBL provides natural mechanism to utilize the general, high-level knowledge. The concept of strokes, together with the knowledge of how the strokes should be written, can be readily used by EBL to build explanations to illustrate how the strokes are realized in the digit images. Combining EBL with statistical learning techniques, therefore, provides an opportunity to introduce high-level knowledge into a learning system, thus further improve the machine learner's performance.

EBL can also benefit from the presence of the statistical learning algorithm. One problem of the conventional EBL is that it is sensitive to the imperfection in the knowledge [Tecuci & Kodratoff, 1990]. With statistical learning algorithms, the system can use examples to evaluate and calibrate knowledge, therefore offers a more robust way to utilize knowledge.

In this thesis, we first draw a distinction between domain knowledge and solution knowledge. Solution knowledge encodes prior information about the particular concept to be acquired. Domain knowledge is more general. It is common to many related learning problems and often expresses causal knowledge or an expert's understanding of causal knowledge within some limited scope of the world. Domain knowledge that is relevant to learning tasks is often plentiful. It is the sort of knowledge we would like to utilize in our research.

We propose a new learning framework to incorporate knowledge into learning system. In this framework, domain knowledge is used by the EBL component to interact synergistically with the evidence from the examples in the training set. The resulting EBL concept is then treated as solution knowledge to introduce bias into a statistical learning algorithm. By doing so, concept learners of this sort may be able to learn with fewer training examples than is possible with conventional inductive learners and more robustly than any conventional EBL system. More detail about the distinction between solution knowledge and domain knowledge, together with the proposed EBL learning framework is the topic of the chapter 2.

We implement our new learning framework with three different approaches. In these approaches, domain knowledge is used in a similar manner by EBL to produce explanations. They differ in how the explanations are used to help the statistical learner, in this study, a SVM learning algorithm. The first approach, which is discussed in chapter

4, uses explanations to form an EBL concept which, treated as probability distribution, can be used to generate artificial “phantom” labeled examples to augment the training set for SVM. The second approach, which is topic of chapter 5, uses explanations to identify which low-level features are useful to detect informative high-level features. Such information is then used to engineer a specialized kernel function for the given learning task. The third approach, which is illustrated in chapter 6, directly uses explanations to introduce extra constrains or preference into SVM learning algorithm. In all three approaches, the interaction between knowledge and examples transforms the general domain knowledge into specific learner’s bias, therefore make the learner aware the high-level knowledge.

The purpose of this research is not to solve any particular real-world classification problem, but to propose a new research direction to integrate prior domain knowledge, an often-ignored source of information, into the statistical learning process. We wish the knowledge to be used in a systematic manner, instead of a domain dependent fashion. Therefore, whenever possible, we offer formal analyses for our approaches. Accordingly, the empirical results in this study are mainly used to demonstrate the effectiveness of our approach and to test the predictions we made in the formal analyses.

Our research promises a new and more robust approach to machine learning by leveraging general domain knowledge to improve classification learning. It insulates domain expert form the detail of the learning algorithms. The prior knowledge is no longer forced to fit into the vocabulary of the learner’s bias. Therefore, this new research direction provides an opportunity to explore more prior knowledge, even the available knowledge base. We wish this research work can make it convincing that one of the most promising but under-explored research areas in machine learning today is the integration of prior domain knowledge into the learning process. Much work can be done in this direction to make artificial intelligence closer to reality.

CHAPTER 2

Domain Knowledge vs. Solution Knowledge

A common way to incorporate prior knowledge into a learning system is to introduce inductive bias [Haussler, 1988]. Yet, some knowledge cannot be used in this way. One example is knowledge about strokes in handwritten character recognition domain. Such high-level knowledge is too general to specify a pixel-level bias. This motivates a distinction between domain knowledge and solution knowledge, and raises an important research question – how to incorporated general, often high-level domain knowledge into inductive learning system.

Explanation-based learning offers a natural mechanism to utilize domain knowledge. Instead of using knowledge to bias the hypothesis, EBL applies domain theory to build justifications for why the training examples are assigned their given class labels. Inspired by the fundamental idea behind EBL, we propose a novel learning framework that combines EBL with a statistical learner to incorporate domain knowledge into an inductive learning system.

2.1 Handwritten Chinese Character Recognition: A Motivation Domain

Distinguishing handwritten Chinese characters is a challenging task. A similar, but much simpler, problem of recognizing handwritten digits has long been used as a test bed for evaluating and benchmarking classifiers [Bottou et al. 1994, LeCun et al. 1995a, LeCun et al. 1995b]. An often-used database, MNIST database, contains 60,000 images of handwritten digits from approximately 250 writers. All the images were normalized to fit in 20x20 pixels. The pixels have grayscale values. Therefore, each image can be represented by a fix-sized vector of integers. Such representation can be directly taken as input for most of learning algorithms, such as neural networks, decision trees and support vector machines, etc.

With 6,000 training examples for each digit, SVMs with polynomial kernels can perform well for handwritten digit recognition [LeCun et al., 1995a]. But handwritten Chinese characters offer a more challenging problem. Chinese characters are more complex than Arabic digits. The available database employs a higher resolution (63x63 pixels, comparing to 20x20 pixels for handwritten digits), which increases the size of the feature space. Also since there are more than 3,000 commonly used characters, it is prohibitively expensive to obtain thousands of examples for each. The most complete database [Saito, Yamada, & Yamamoto, 1985] we found for handwritten Chinese characters includes just 200 examples for each of 3048 characters. It is not surprised that, for such complex and difficult learning task, directly applying off-shelf classification algorithms cannot produce satisfied results. One way to improve classifiers' performance is to incorporate domain knowledge.

What knowledge can we express for this domain? For humans, natural description of characters is in the vocabulary of *strokes*. A nice property of Chinese characters is that, despite of the complexity of the characters, they are made by a set of quite few different strokes. Between the level of strokes and characters, there also exist *radicals*, which are common combinations of few strokes. Strokes and radicals offer an abstract vocabulary

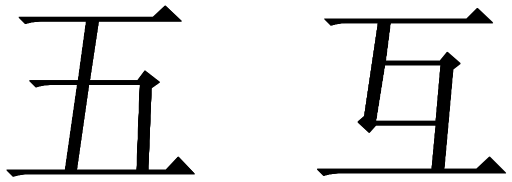

Figure 2.1 The prototypes of two Chinese characters.

to organize our understanding of Chinese characters. In fact, human experts can describe all the Chinese characters using only strokes and radicals.

Knowledge of strokes and radicals plays an essential role for humans to learn how to recognize Chinese characters. Humans do not learn a character by observing hundreds and thousands images. Instead, they often memorize the character *prototypes,* which describe how the characters are composed by strokes and radicals. The handwritten images are only needed for a novice who knows little about Chinese characters. For an expert, it is possible to learn a new Chinese character only from prototypes. With knowledge of how a character is written, an expert can recognize the handwritten character even she has never seen any handwritten images of that character before.

Even for people who don't know Chinese, knowledge of strokes could be quite useful to distinguish two different Chinese characters. Figure 2.1 shows prototypes of two similar Chinese characters. For people with some knowledge about line drawings, it is natural to notice the difference of the two characters, such as that the first character has a *cross*, which is composed with one horizontal and one vertical stroke, in the middle of the character, while the second character has *corner* and *T-junction* in the middle of the character. With such background knowledge, it is not difficult for humans to distinguish these two similar characters.

2.2 Solution Knowledge vs. Domain Knowledge

Clearly, stroke knowledge is extremely useful for human to recognize Chinese characters. This kind of knowledge has the potential to greatly aid machine learning. Nevertheless, such knowledge cannot directly be integrated into an learning algorithm as bias, because it is described in different vocabulary from the learner's bias. It has a vocabulary mismatch with the learners' bias: the learner wants to build a classifier that operates on pixel level, while our stroke knowledge is described in the unobservable, high-level stroke vocabulary. An SVM or a neural network for handwritten character recognition operates on pixel level, yet stroke knowledge does not comment on which pixels are important for the classification, or which combination of pixels could serve as useful features for the learning algorithms.

This motivates us to make an important distinction between *domain* knowledge and *solution* knowledge. By solution knowledge we refer to knowledge that suggests what properties the classifier itself should have. It may specify properties of the input variables useful for the task, or point out important features, or provide specific rules about how to classify examples. Such knowledge can often be captured in a learner's bias vocabulary as it directly restricts the hypothesis space that the learner must search. On the other hand, domain knowledge provides information of how things work in the domain. It applies more broadly than a single classification task. Experts often understand their domains using a vocabulary of high-level unobservable features that are useful in describing domain regularities. This is often too general to be directly used by an inductive learner.

The structure of a Bayes network [Pearl, 1988], for example, directly influences which distributions can be acquired by the learner. The kernel function of a support vector machine, the topology of a neural network, and the vocabulary of splits from which a decision tree is to be built all exemplify solution knowledge. Stroke knowledge in handwritten Chinese character recognition domain, on the other hand, is an example of domain knowledge.

Another kind of domain knowledge in the handwritten character recognition is knowledge about how the strokes are realized. For example, one may believe that the same person and the same writing implement made all of the strokes of a particular image. The emergent patterns formed by handwritten images would indeed be very different in a world where this knowledge did not hold. But the knowledge is not directly associated with a particular classification task. It does not affect, for example, whether a sample image should be classified as an "H" or "N."

Solution knowledge is easily integrated into the machine learning process as bias, but we believe its utility is limited compared to domain knowledge. Domain knowledge, while more difficult to employ, is generally more reliable and more easily articulated by a human expert. This is because the domain expert need not also possess expertise about the machine learning techniques and about the particular learning task at hand. In handwritten character domain, stroke-level knowledge is more natural for experts than pixel-level knowledge. Therefore, a learning system that can utilize domain knowledge has the advantage of using more available knowledge.

2.3 A New Learning Framework

Domain knowledge cannot be directly incorporated into learning algorithms as learner's bias. We need to investigate new approaches to take advantage of such knowledge. Again, we can obtain some insights about this problem from handwritten character domain. Human's brain, as a learner, takes both stroke-level knowledge and example images as inputs. It differs from a computer learner in that human's brain allows interaction between stroke-level knowledge and examples images. It uses images to illustrate how stroke prototype can be realized, and uses stroke-level knowledge to explain how the character images merit their labels.

The idea of allowing interaction between domain knowledge and examples has long been the focus of Explanation-Based Learning. EBL views each training example as an illustration of some deeper pattern rather than as an isolated data point [DeJong 2004]. The assigned training labels of the examples are *justified* or *explained* using a prior domain theory. Each explanation applies (and therefore suggests labels for) many other unseen examples. Thus, the information content of an explanation is far higher than the information content of the originating training example.

The contribution of this research is to propose new learning approaches that integrate explanation-based learning with statistical learning mechanism. Our version of EBL can be viewed as a process of inferentially transforming examples and domain knowledge into solution knowledge tailored to the learning mechanism and the learning task at hand. Those deeper patterns obtained by EBL are used as solution knowledge to

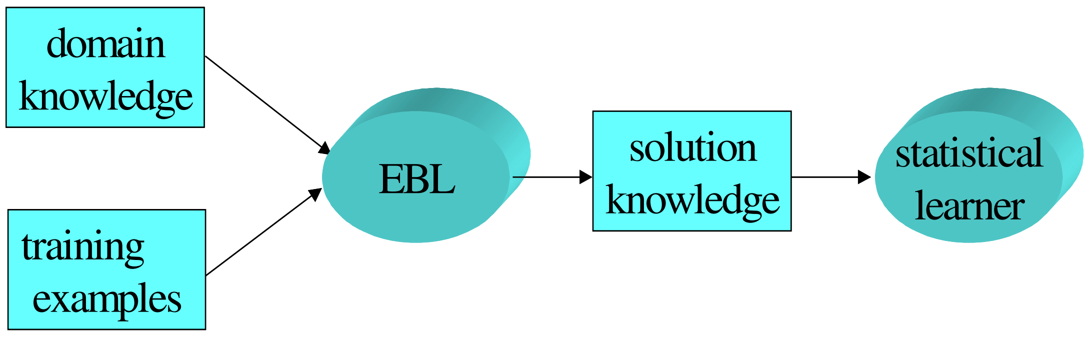

Figure 2.2 The framework of combining EBL with SVM.

introduce further inductive bias into the learning system. This learning framework is illustrated in figure2.2.

The advantage of such learning system is that it insulates the domain expert from the eccentricities of the statistical learning algorithm and the particular task. The expert is no longer forced to distort his expertise into the bias vocabulary of the learning mechanism. Yet, by fully and appropriately appreciating the enhanced solution knowledge generated by EBL, confidence in a concept can grow more quickly, and so an appropriate element of the hypothesis space then can be selected in a much more example-efficient manner.

Incidentally, this learning framework also address some problems faced by conventional EBL. One of such problem is called utility problem [Minton, 1988], which states that the conventional EBL can produce a concept that is correct, but may not be very useful. As in handwritten Chinese character domain, conventional EBL can learn a concept described at stroke-level. Such concept, even being accurate, can only be useful if we can reliably transform pixel images into stroke representations. The utility problem is sidestepped in our approach as the output concept is itself a member of the inductive learner's hypothesis space. The definition of hypothesis space guarantees that the learned concept can be easily evaluated.

The other problem of EBL is its sensitivity to the imperfection in the knowledge. The conventional EBL assumes knowledge to be correct [Mitchell, 1997], and uses logic as the inferential mechanism. It relies on logic proving to produce EBL concept. Such system is usually not robust, since imperfect knowledge or the inconsistence in the knowledge can led to failure of producing useful EBL concept. In our proposed approach, an explanation is not a logical proof but only general conjecture which must be calibrated and evaluated prior to use. Furthermore, the other component in our learning system, the statistical learner, can use the information in the data to calibrate and correct the solution knowledge produced by EBL component. Therefore, as we will show by empirical results, our new EBL learning framework can offer a robust approach to utilize domain knowledge.

CHAPTER 3

Our Inductive Learner: Support Vector Machine

Different learning algorithms use different hypothesis spaces, hence their biases are in various forms. The bias for neural networks often specifies the network topology; a Bayesian network uses knowledge about independence among input variables to introduce bias on which distribution it can represent; and specifying the vocabulary of splits from which a decision tree is to be built certainly influence the output of a decision learning algorithm. Choosing different learning algorithm as the inductive learner in our new EBL learning framework determines what kind of solution knowledge that the EBL component should produce. Therefore, determining the statistical leaner is an important issue for implementing our EBL learning framework.

In this work, Support Vector Machines (SVMs) are used as the inductive learner. SVM [[Vapnik, 1998] is a relatively new and successful approach to classification learning. It uses the structural risk minimization principle to construct decision rules that generalize well, and can efficiently compute linear separators in very high dimensional spaces using kernel functions [Burges, 1998]. In many practical applications, such as handwritten digit recognition [Burges & Schölkopf 1997; Cortes & Vapnik 1995], speaker identification [Schmidt, 1996], face detection in images [Osuna, Freund & Girosi 1997], bio-sequence analysis [Leslie, Eskin, & Noble 2002; Leslie & Kuang, 2003], and test categorization [Joachims, 1997; Christianini, Shawe-Taylor & Lodhi, 2001] , SVM generalization performance (i.e. error rates on test sets) either matches or is significantly better than that of competing methods.

Yet, as we will discuss in this chapter, how to incorporate knowledge into SVM remains an important and challenging problem. Kernel functions are often cites as a mechanism to incorporate prior knowledge into SVMs [Cristianini & Shawe-Taylor, 2000]. However, the vocabularies used to describe the kernels seldom fit task experts' conceptualizations of their prior knowledge. As in handwritten character domain, kernel functions operate at pixel-level, while the natural vocabulary for experts to describe domain knowledge is in stroke level. The approaches proposed in this study offer a new direction to improve SVM performance by utilizing prior domain knowledge, which previously has been ignored.

3.1 Linear Separator and Margin

The idea of support vector machines can be motivated in several different ways. We choose the idea of margin to introduce SVMs. Consider a linear separator trained on separable data. Let the labeled training data be $\{x_i , y_i\}$, $i = 1; \ldots, n$, $y_i \in \{-1,1\}$, $x_i \in R^d$ .. Suppose there exists some hyperplane which separates the positive from the negative examples (a ``separating hyperplane"). Often such hyperplane is not unique: rotating or translating such hyperplane with a small amount often result another hyperplane that also separates the positive and the negative examples. Especially when the feature space has large dimension, and training sample size is small, the hyperplanes that separate the sample could be quite different. Do we have any belief to prefer one from another?

One intuitive idea is to choose the hyperplane with largest *margin*. The margin of a hyperplance is defined to be the shortest distance from the examples to the hyperplace.

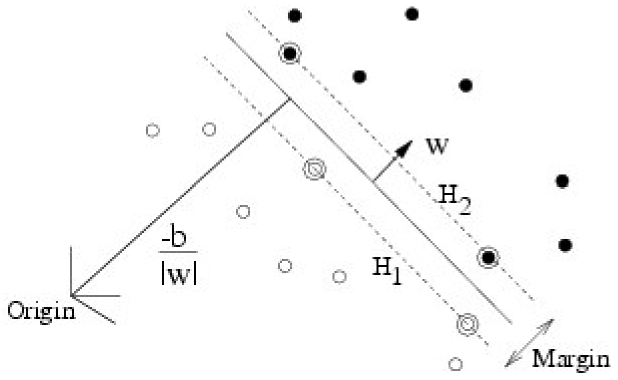

Figure 3.1 Support vector machines learn a linear separator with largest margin. The support vectors are circled.

Specifically, consider the points *x* which lie on the hyperplane satisfy $wx + b = 0$, where w is normal to the hyperplane, $|b|/\|w\|$ is the perpendicular distance from the hyperplane to the origin, and $\|w\|$ is the Euclidean norm of w. Let d_+ (d_-) be the shortest distance from the separating hyperplane to the closest positive (negative) example. Then *margin* of a separating hyperplane is defined to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin. This is illustrated in figure 3.1.

The intuition of maximizing the margin is that we could have high confidence on such classifier. The feature space defines a reasonable distance measure for the problem, so that the future examples are likely appear in the positions close to the existing training example with the same label. The margin defines a buffer zone around the hyperplance, such that when the future examples fall in such buffer zone on the correct side, the hyperplane can make correct prediction. Therefore, we would like to make this buffer zone as big as possible.

Several learning theories have been developed to support the above intuition, and are used to show that SVMs indeed have good generalization performance [Herbrich & Graepel 2001; Bartlett & Shawe-Taylor, 1999; Shawe-Taylor & Williamson 1999]. The fundamental idea of those theoretical studies is that the margin of a linear separator closely relates with the learner’s capacity. Large margin leads to low learning capacity, which results to high confidence of the learned classifier. In particular, one of such theory [Bartlett & Shawe-Taylor, 1999] follows the conventional Probably Approximately Correct (PAC) learning theory. It uses fat-shattering dimension [Bartlett, Long, and Williamson 1996], which is a modification of conventional VC dimension [Vapnik & Chervonenkis 1971], to measure the expressiveness of the learner’s hypothesis space. In this framework, it can be shown that the hypothesis space of linear classifier with large margin is indeed small. Therefore, the expected error rate of the learned classifier is close to empirical error rate.

The problem of finding large margin linear separator can be formulated as an optimization problem, which can be solved by standard quadratic programming techniques. To see this, suppose all the training data satisfy the following constrains:

$$x_i \cdot w + b \geq 1, \quad for\ y_i > 0$$

$$x_i \cdot w + b \leq -1, \quad for\ y_i < 0$$

These can be combined into one set of constrains:

$$y_i(x_i \cdot w + b) - 1 \geq 0$$

Now consider all the positive training examples with shortest distance to the separator. They lie on a hyperplance H_{+}: $x_i \cdot w + b = 1$, whose perpendicular distance to the origin is $|1 - B| / \|w\|$. Similarly, those negative training examples with shortest distance to the separator lie on a hyperplane H_{-}: $x_i \cdot w + b = -1$, whose perpendicular distance to the origin is $|-1 - B| / \|w\|$. Hence d_{+}=d_{-}=$1/\|w\|$. Therefore the margin d_{+} +d= $2/\|w\|$. Maximizing margin can be formulated as minimizing $\|w\|^2$, subject to the constrains:

$$y_i(x_i \cdot w + b) - 1 \geq 0$$

Using standard Langrangian method, we get the following dual problem:

maximize: $\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y \ \ y_j x_i \cdot x_j$

subject: $\alpha_i \geq 0, \forall i$

$$\sum_i \alpha_i y_i = 0$$

This is a standard quadratic programming problem with linear constrains. Moreover, it can be shown that the quadratic term is convex [Burges, 1998], therefore, they solution is unique.

3.2 Kernel Functions

To obtain a non-linear separator, support vector machines use a rather old trick [Aizerman Braverman & Rozoner 1964] to map examples from original feature space to a high, possible infinite, dimensional space. To illustrate, first notice that, the only way in which the training data appear in the above SVM learning problem is in the form of dot products $x_i \cdot x_j$. Therefore, the only thing we need to know about the high dimensional space is how to compute dot products.

In particular, suppose a function *Φ* is defined to map *x* to a Euclidean space *H*:

$$\Phi : R^d \mapsto H$$

Now if there were a "kernel function" *K* such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, we would only need to use *K* in the learning algorithm, and would never need to explicitly know what *Φ* is. In this way, a linearly non-separable problem can be mapped to a high dimensional space, where often a linear separator exists.

Because kernel functions implicitly define distance measurement in the mapped space, they are often mentioned as a place in SVM learning algorithm to incorporate prior knowledge about the problem. A space where examples are clustered according to their labels certainly makes learning problem easier, and it seems that the prior knowledge about the problem should offer information about how to define such a desirable space.

Yet, there are many restrictions about kernel functions. They have to operate on input attributes, and should be easy to compute. Moreover, to keep the quadratic programming problem convex, the kernel functions have to satisfy Mercer's condition [Courant Hilbert 1953; Vapnik, 1995], which states that when the a kernel function is applied to a set of data, the resulting kernel matrix has be positive semi-definite. Those restrictions make kernel functions more or less like black boxes. It is usually difficult to imagine what properties the mapped space has. Therefore, although designing kernel functions has been a well-studied research field, it is still difficult to use prior knowledge to guide the design of kernel functions. Rather, generic kernel functions, such as polynomial kernels, radical basis function kernels, and Gaussian kernels are often used in real-world applications.

3.3 Knowledge in Native SVMs

Support vector machines often are able to achieve good generalization performance without relying too much on prior knowledge on the problem. The intrinsic bias for SVM learning algorithm is to use margin to regularize risk function. The margin is used to control the expressiveness of the hypothesis space. Such bias reflects no knowledge about the learning problem. For a native SVM learning algorithm using a generic kernel function, which is defined without using any prior knowledge, we can only hope that the attributes used to describe the examples are good enough so that the kernel functions can map examples into a metric space with good distance measure.

The fact that SVM uses little prior knowledge was noticed by researchers shortly after the SVM was introduced. In 1995, LeCun et al. made the following statement when comparing SVMs with other learning techniques for the handwritten digit recognition problem:

"The optimal margin classifier has excellent accuracy, which is most remarkable, because unlike the other high performance classifiers, it does not include *a priori* knowledge about the problem. In fact, this classifier would do just as well if the image pixels were permuted by a fixed mapping…"

In conclusion, SVMs have the excellent ability to extract information from the training data, and utilize little prior knowledge. These dual characteristics make SVMs a good candidate as the statistical learner in our new EBL learning framework. We expect the kind of domain knowledge describe in chapter 2 can compensate the information from training set, therefore greatly improve the already excellent SVM performance.

CHAPTER 4

Phantom Example Approach

Artificial examples have previously used to improve machine learning systems. The idea of training with phantom as used to accelerate learning of a control task to ride a bicycle [Brodie & DeJong, 1999]. In a classification task the phantom approach is related to training with invariance [Scholkopf, Bruges, & Vapnik, 1996]. In that approach a number of virtual or synthetic examples are generated by applying limited syntactic transformations to each training example. It is assumed that such transformations, for example small translations, rotations, and scaling are unlikely to affect the classification label. The new slightly mutated inputs are assigned the classification label of the untransformed training example. This way the original training set can be expanded many fold.

In the approach described in this chapter, EBL phantom examples are generated similarly but using small semantically oriented mutations. The kind of knowledge explored here is expressed in the vocabulary of general high-level features, contrasting the knowledge of invariance at pixel level. EBL procedure is carried to explain training examples in the vocabulary of high-level features. Those explained examples are then used to calibrate a statistical model of each class conditional distribution over stroke features. New phantom training data are generated through the interaction of the original training examples with the calibrated statistical model. In this way, high-level domain knowledge, which is otherwise difficult to use, is readily to be incorporated into off-shelf learning algorithms, such as SVMs.

4.1 Motivation

The fundamental idea behind EBL is to generalize each training example to illustrate of some deeper pattern rather than treat it as an isolated data point. To generalize a single training example, the assigned training label of that example is *justified* or *explained* using a prior domain theory. Each explanation applies (and therefore suggests labels for) many other unseen examples. Thus, the information content of an explanation is far higher than the information content of the originating training example.

The handwritten character images, for example, can be generalized or explained from pixel-level description to stroke-level description, which uses parameterized stroke representation to describe each character example. Stroke-level representation hides certain details from pixel-level representation. Therefore, many different pixel images of the same character will be represented in the same stroke-level description, i.e. with the same parameters describing the strokes in the characters.

With the stroke-level representation, we are able to exercise our belief of how the characters are written to further generalize or extend explanations to deeper patterns about the classification task. A simple example can be that, when we see a character image with the a horizontal stroke with five pixel long, and the same stroke in another character with nine pixel long, it is reasonable to believe that the length of that stroke can be anywhere from five to nine pixels.

A natural extension of the above idea is to treat the generalized EBL concept as a probability distribution that describes how the stroke parameters should be distributed.

For example, we can treat the stroke length parameter as a free variable, and associate a probability distribution with this variable. Given a set of explained examples, where each of them specifies the value for that variable, we can build the probability distribution for it. The model approximates how likely a particular stroke representation will be seen given that it is an example of this character. As such, it represents the solution knowledge referred to earlier. Such distribution can be viewed as a generative model, which captures patterns exhibited in the stroke space. The model certainly contains useful information about the classification task. However, there is obstacle to its immediate use in an inductive leaner: the model specifies strokes whereas the SVM learner is only concerned with the pixel representations.

To communicate the generative models to SVM learner, we use them to build synthetic examples. The synthetic examples are constructed from the high-level stroke distribution and mapped back to pixel space. We call these synthetic labeled images "phantom training examples" in part to distinguish them from the virtual examples of Skolkopf [6] which result from syntactic manipulations of individual actual training examples, and in part because they play a similar role to the phantom training examples used by Brodie [8] in the control learning task of bicycle riding.

Our approach can be viewed as an implementation of the proposed new EBL learning framework described in chapter 2. As illustrated in figure 4.1, EBL component uses domain knowledge and training examples to build generative models, which is then used to construct phantom examples. In this approach, phantom examples serve as solution knowledge. The extra bias we introduce to the SVM learning algorithm is to

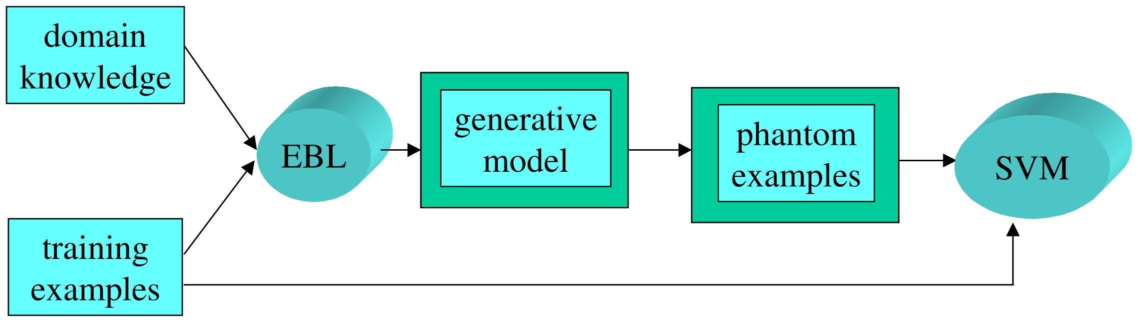

Figure 4.1 The phantom example approach to implement the new EBL learning framework.

correctly classify those phantom examples. When the phantom examples resemble the real examples, such bias certainly can help to build better classifier.

4.2 Building Phantom Examples for Chinese Characters

4.2.1 Explaining Handwritten Characters

In handwritten character domain, input examples are described at the pixel level, while we would like to build explanations that account for how pixels can be interpreted as strokes and how the strokes give rise to the particular character. To do so, we employ domain theory that embodies information on how each character is drawn using stroke: Strokes are approximations to straight lines. Each stroke in the same character is made by the same person using the same writing implement and therefore is likely to share characteristics such as line width. Consecutive strokes that share a terminal point are likely to be drawn without lifting the writing implement, etc.

Domain knowledge also includes character-specific information such as the number of strokes used, their position relations, their length relations, which strokes should be more parallel, which should be more horizontal, which more vertical, etc. An explanation of an example is a conjectured set of specific strokes that account for the input pixel representation.

With our domain knowledge, the task of explaining character images reduces to the problem of finding image lines that best match the known stroke. We apply a Hough transformation [Hough, 1959] to accomplish this job. Notice that using a Hough transform on training images is far more reliable than to find lines in an unknown image. The label of the training image provides access to its prototype stroke representation.

In detail, the line-fitting Hough transformation takes each black pixel in the image, and determines the equations of lines that could pass through that pixel. By encoding each point, the image is transformed into a parameter space. Discretizing these parameters yields a set of quantized finite intervals or *accumulator cells* with counts of how many image points its family of lines describe. Peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image. In our problem, we have more complicated model than a single line, so the parameter space has a higher dimension.

Now we use digit “4” to illustrate how our stroke-level knowledge can help simplify line finding task. Digit “4” is similar to Chinese characters in that it is composed

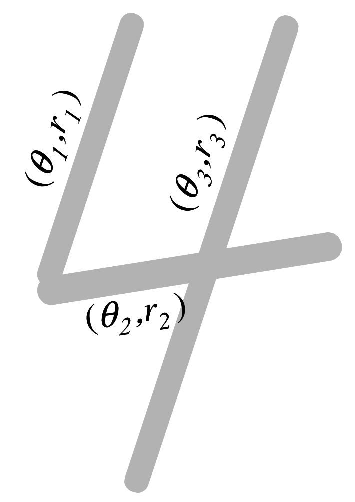

Figure 4.2 Parameters used in our model of digit "4".

by the strokes that can be approximated by lines. Yet it is simple enough to keep our illustration easy to understand. In detail, we use equation

$$x\cos\theta + y\sin\theta + r = 0$$

to represent a single line. We use six parameters ($\theta_1, \theta_2, \theta_3, r_1, r_2, r_3$) to represent three lines used to approximate the three strokes in the digit "4", as shown in figure 4.2.

The domain theory also supplies restrictions among the line segments. Such restrictions include whether two line segments should intersect, whether the pixels in one line segment should always be on left/right/top/bottom side of another line segment, etc. Such restrictions can be used to constrain the parameters space searched in Hough transformation, or to check the validity of the parameters found by Hough transformation. For example, if we believe a point *(x,y)* belongs to the line 1, then it should be above the line 2 and to the left of the line 3. Thus, the parameters must satisfy the following relations to qualify:

$$
\begin{aligned}
&x\cos\theta_1 + y\sin\theta_1 + r_1 = 0\\
\{\ &|\,x\cos\theta_2 + y\sin\theta_2\,| \le r_2\\
&|\,x\cos\theta_3 + y\sin\theta_3\,| \le r_3
\end{aligned}
$$

If we believe a point *(x,y)* belongs to the second line, we believe that it should be below the first line, but we cannot express restriction on its position relation to the third line. So the parameters must satisfy the following constraints:

$$
\begin{cases}
x\cos\theta_2 + y\sin\theta_2 + r_2 = 0\\
\left|x\cos\theta_1 + y\sin\theta_1\right| \ge r_1
\end{cases}
$$

Similarly, if we believe a point *(x,y)* belongs to the third line, then the constraints are:

$$
\begin{cases}
x\cos\theta_3 + y\sin\theta_3 + r_3 = 0\\
\left|x\cos\theta_1 + y\sin\theta_1\right| \ge r_1
\end{cases}
$$

To use above constrains in Hough transformation, for each point *(x,y)*, those parameters that satisfy either of the above three sets of equalities or inequalities contribute to the accumulator counts. The accumulator cells with the most votes give us the parameters that fit the given image to our model for digit “4”. In this way, our Hough transformation is able to detect three lines that satisfy the constraints imposed by our domain knowledge. The advantage of doing so is that the constraints reduce the size of the search space. Therefore, it is more reliable than detecting three lines independently.

For more complicated Chinese characters, we decompose the overall line detection problem into many small problems to detect simple stroke-level patter, such as cross, T-junction, and corners etc. We then apply our specialized Hough transformation to those small problems. Figure 4.3 illustrates our approach. The procedure EXPLAIN takes a pixel image and the stroke prototype as input, and determines the parameters for

```
EXPLAIN (Image im, Prototype pt)
    If pt is empty, return.
    Otherwise
        pattern ← FIND_PATTERN (pt)
        apply Hough (im, pattern)
        pt ← pt – pattern (excludes examined pattern from the prototype pt)
        apply EXPLAIN (im, pt)
```

Figure 4.3 A procedure to explain handwritten character images into stroke representation.

each stroke in the prototype. The subroutine FIND_PATTERN examines the input prototype, and determines the unique pattern with fewest strokes. The subroutine HOUGH, which implements our specialized Hough transformation, determines the parameters for the stroke-level pattern, with restrictions about their relative positions.

A good domain theory supports explanations that capture a great deal of the relevant information while eliminating much that is irrelevant. Even a domain theory as simple and coarse as ours performs quite well in this regard. Figure 4.4 shows a few original training images (upper lines of digits) together with an image realized by using the explanation's parameters of each (lower lines of digits).

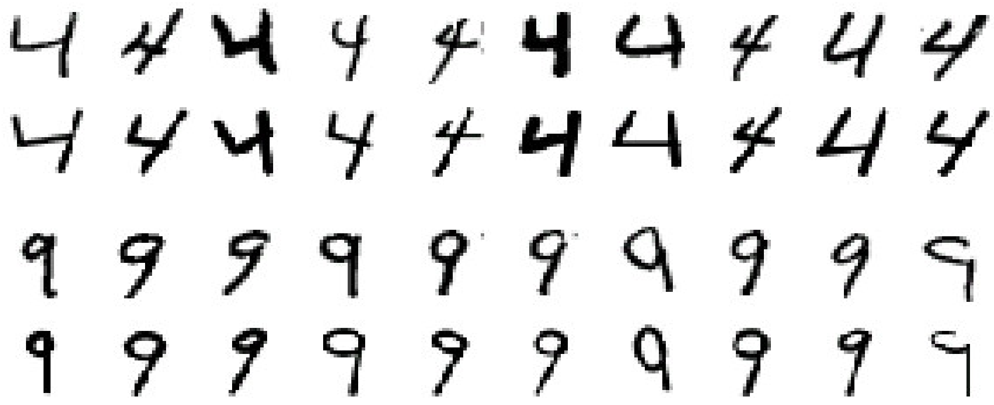
Figure 4.4 Results of our explanation procedure.

4.2.2 BUILDING THE PROBABILISTIC GENERATIVE MODEL

The next step in our phantom example approach is to estimate the example distribution. Without explanations, estimating such distribution in terms of pixel intensity is formidable. With explanations, the distribution is defined over high-level parameters. We will see shortly that such distribution is much easier to estimate.

A straightforward approach of building the distribution is to define a distribution for every parameter we used to describe the explained examples. The problem for this simple approach is that, since the strokes are correlated, the parameters that define them also have significant correlation. For a relatively complex character, there could be unnecessarily too many parameters to represent the correlation between variables.

A solution to this problem is to use domain knowledge about such correlations to define a space where parameters are independent. Two strokes might be parallel, cross, or one might begin from the end point of the other. For parallel strokes, we prefer to represent the slope of one stroke as an increment angle; for crossing strokes, we prefer to choose the intersection point instead of start point as parameters of the line segment; for the connected strokes, two parameters can be eliminated from the representation. Here we use digit "4" again to illustrate how to use domain knowledge to minimize the dependence among parameters. Our generative model for the digit "4" is represented by nine independent parameters $\{\theta_1, \theta_2-\theta_1, \theta_3-\theta_1,, l_1, l_2, d, l_{31}, l_{32}, t\}$, where $\theta_1, \theta_2,$ and θ_3 are slopes of the three constituent lines, which can be determined by Hough transformation. For the other parameters, l_1, l_2 are the lengths of line 1 and line 2, d is the distance between connection of line1 and line 2 to the intersection of line 2 and line 3, l_{31} is the length of the line 3 that is above line 2, l_{32} is the length of the line 3 that is below line 2, and t is the thickness of the strokes (see Figure 4.5). The definition of these parameters takes advantage of knowledge about parallel strokes, and connected strokes. Therefore, we assume they are independent parameters.

The parameter independence makes it possible to use products of individual distributions for each parameter to estimate the joint distribution. The individual distributions are chosen to be Gaussian whose statistics are estimated from the labeled training examples. The estimated joint distribution reflects our stroke-level knowledge, such as how many strokes are in each character, and the positional relation among them. It is also calibrated from the training examples. It represents our belief of how the

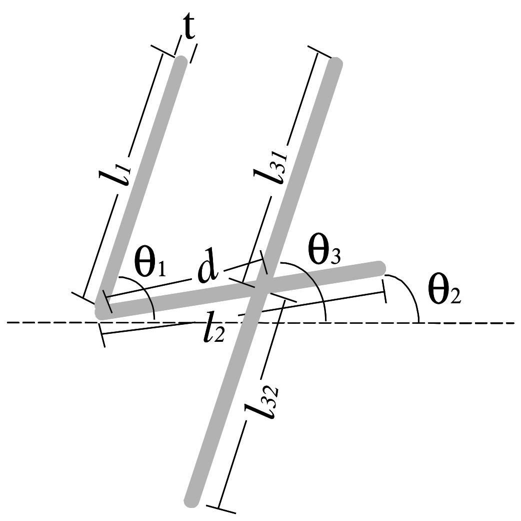

Figure 4.5 A generative model for digit "4".

training examples are generated, and can be used as a generative model to construct artificial examples described in the vocabulary of stroke parameters.

4.2.3 Synthesizing Phantom Examples

Our generative model is different from those generative models commonly used in machine learning, such as graphical models. Graphical models [Jordan 1998], such as Bayesian networks or Hidden Markov Models (HMMs) [Rabiner 1989], represent a probability distribution over observable variables used to describe the examples. Therefore, they can be estimated directly using original training examples, and can be used directly to classify future examples. Our generative model is described in the vocabulary of un-observable, high-level features. Since it has vocabulary mismatch with the training examples, explanation procedure has to be used to map low-level attributes in the training examples to high-level features. Yet such high-level generative model has advantage in that it offers the opportunity to express natural domain knowledge. In our handwritten character domain, specifying a generative model in the vocabulary of pixels is formidable, yet, as we showed, a simple stroke-level model can be easily described using our knowledge about how the characters should be written. Moreover, since our generative models are not used directly for classification, they are relatively tolerant to errors. In our implementation, we choose to ignore the dependency among the high-level features. As we will show, that ignoring such details results in imperfect phantom examples, yet they still carry large enough information about the task, therefore can be beneficial the learner.

Since our generative models cannot be directly used for classification, we use them to generate phantom examples, which are used to communicate the information implicit in the EBL-generated solution knowledge to the machine learner. In handwritten character domain, phantom examples are first constructed using the estimated generative models. Since they are described in the stroke vocabulary, we then use our domain knowledge about how the strokes can be realized in pixels to map the phantom examples into pixel levels.

The phantom examples are similar to the original example in that they both are described in the vocabulary of pixels. Therefore, learning algorithms can directly use those phantom examples in the same way as they use original training examples. Comparing to real examples, phantom examples are easy to generate in the sense that no human intervention is required. Yet, they are not as effective as real examples, because either our domain knowledge may be imperfect, or for practical purpose, we choose to ignore the details in our domain knowledge. In our implementation, we know that human hardly writes a stroke as a perfect straight line. With knowledge about pen movement and how ink diffuses on the paper, we may specify a better model for how strokes are realized as pixels. Yet, for simplicity purpose, we chose to use the simple model, each stroke is straight line, with the fixed thickness. As shown in figure 4.4, our reproduced explained examples using this simplified model do not look exactly like the real examples. Therefore, although we believe that a learning algorithm augmented by EBL with phantom examples will exhibit significantly improved behavior with small training sets, we may also expect the large-training-sample behavior to be somewhat worse than the native SVM.

In the case of SVMs, which still have difficulties to use large sample size, we wish to use phantom examples efficiently as we wish to derive likely support vectors from the class conditional probability distribution. This agrees the idea behind virtual example approach proposed by Schölkopf [1996]. In that approach, artificial examples are generated by transforming real examples by translation or rotation. To make those artificial examples effective, they chose to only transform those training example that serve as support vectors for the SVM trained purely with real examples. In our approach, those phantom examples that are away from the mean but not too far into the tails of the distribution are likely to be support vectors, since they are expected to be difficult examples. We generate phantom examples by drawing independent random examples by treating the normal product distribution as a product of *uniform* distributions. The uniform distribution represents a compromise between confidence in the location of the significant true probability mass (which is highest near the mean) and the desire to generate likely support vectors (which is lowest near the mean). The limits of the uniform distributions are derived from their normal counterparts: they are the smallest bounds that

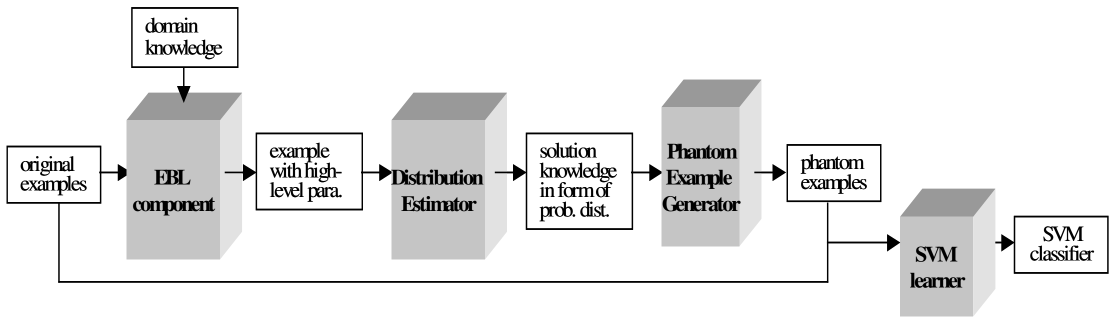

Figure 4.6 A functional block diagram of phantom example approach.

cover 95% of the Gaussian's probability mass. While we have not done a thorough or systematic analysis of this manipulation, the 95% figure seems not to be critical.

In summary, our approach is summarized in Figure 4.6. The EBL component transforms general domain knowledge and examples into specific solution knowledge in form of probability distribution, which is then communicated to the SVM component via phantom training examples. The SVM is trained on the combined set of original examples and phantom examples.

4.3 Experimental Results

We first use the task of classifying handwritten digits to illustrate the efficiency of our phantom example approach. Note that we are not proposing EBL as an effective approach to handwritten digit recognition. In this domain labeled examples are plentiful and cheap compared to the complexity of the concept to be acquired. The true advantages of the EBL approach will be derived in domains where a conventional approach cannot reliably choose a concept due to the paucity or expense of training examples. Rather, handwritten digit recognition provides a convenient test bed with which to validate the expected advantages that EBL may bring to a conventional state-of-the-art empirical learner. Such a test bed must support effective concept acquisition with and without the EBL augmentation, and in its conventional guise (without EBL) the learner should be well understood and competitive, i.e., more than a straw man to be knocked down. SVMs on handwritten digit recognition satisfy these test bed requirements.

Moreover, handwritten digit data set offers plenty examples to test the large sample behavior of phantom example approach. As we mentioned, our domain knowledge is not perfect, its information will only approximately describe the world's constraints. We expect that, the domain theory helps when it is, on average, more informative than the training set but may harm learning when its information is generally of a lower quality than the training set. The information content of a training set increases with the size of the training set. As this size increases without bound it must ultimately surpass that of the domain theory. The residual bias of the domain theory may then slow rather than expedite learning. Our results show that, although phantom examples are not perfect, they never harm the SVM learning algorithm in all our tested conditions.

We next use the task of classifying two Chinese characters to show that our phantom example approach indeed has potential to improve SVM performance on a real-world difficult problem. Chinese characters are more complex than digits, and examples are expensive to obtain. In such domain, our empirically results show that, not surprisingly, phantom examples significantly help SVM learning algorithm, especially when the original training set is small.

4.3.1 HANDWRITTEN DIGIT RECOGNITION DOMAIN.

We chose two similar digits, “4” and “9”, to demonstrate the performance of our approach. These two digits, although are quite different at stroke-level representation, yet they could be quite similar at pixel level to make classification problem difficult. We expect our stroke-level generative model can capture the intrinsic difference between these two digits, and the phantom examples can effectively communicate our knowledge to SVM learning algorithm.

The explanation and phantom example construction procedures are carried as described in previous sections. For digit “9”, a Hough transformation for detecting ellipses is used to obtain the parameters for the curved circle. Figure 4.7 shows some artificially generated phantom examples. Phantom examples with random parameters represent our knowledge without tailored to the problem. They hardly look like digit “4” and “9”. On the other hand, the phantom examples generated from our statistical model, although, as we expect, do not exactly resemble real handwritten examples, can still be reasonably believed to be digit “4” or “9”. As humans we could still classify those phantom examples into their corresponding classes. Therefore, we expect they could serve as informative bias to constrain SVMs to make correct classification.

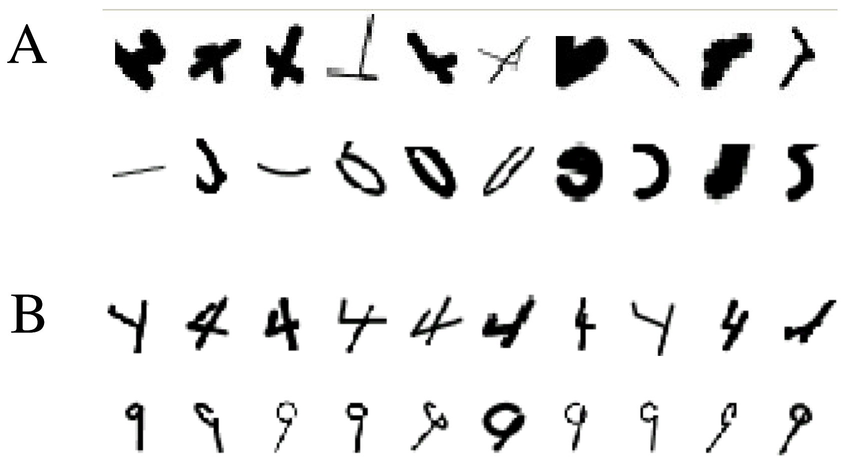

Figure 4.7 Images of the phantom examples. A. phantom examples generated with random parameters. B. phantom examples generated by sampling parameters from learned model.

4.3.1.1 Learning with and without Phantom Training.

To test the efficacy of phantom examples, we trained an SVM on two series of examples. In the first, the “native” condition, the training set size varies from 5 to 500 examples in thirteen equally spaced logarithmically scaled steps. In the second series, the “phantom” condition, the SVM was given the training examples of the first condition augmented with phantom examples. These were generated using the process outlined in the previous section. The corresponding set of native training examples were explained and used to form the EBL concept. Sufficient phantom examples were generated to accurately describe the EBL concept distribution. We chose this number, somewhat arbitrarily, to be approximately 5000. To preclude any effect due to the difference in absolute training size, the number of phantom examples is reduced by the number of real examples that were used to produce it. Thus, the first of the thirteen training sets in the phantom condition consists of 5000 examples. Five of these are actual training examples;

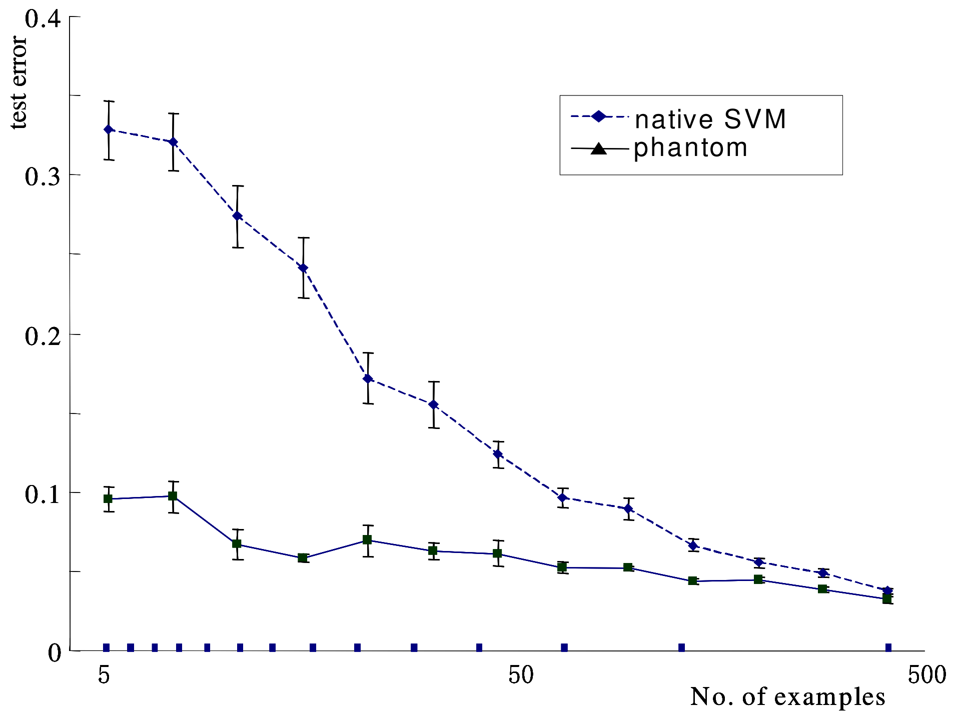

Figure 4.8 Learning curves of native SVM and phantom-example augmented SVM.

the other 4995 are generated by the EBL component after explaining those same 5 actual training examples. The last point represents training on 500 actual examples and 4500 phantom examples constructed by explaining those 500 examples. Each learning session was repeated five times using an independent training set. The results are averaged across these repetitions.

Figure 4.8 shows the accuracy after each training episode as judged on a set of 2000 held back test examples (1000 for each class). Error bars denote the 95% confidence interval about each mean accuracy point.

As predicted, the phantom condition performs significantly better than the native SVM on small numbers of actual examples. With five actual examples (augmented with the 4995 phantom as described above), the phantom condition performs approximately the same as the native condition with somewhat over fifty examples. With fifty actual examples the phantom condition performs slightly better than the native condition with 500 training examples. This relationship is shown in more detail in Figure 4.9B which we will discuss momentarily. Not surprisingly, the accuracy advantage shrinks as the number of actual examples increases. Large training sets contain much more information making the prior knowledge less important.

The running time for the phantom condition is, predictably significantly longer than the native condition, as it must always process 5000 examples. It is not clear that 5000 phantom points extracts all of the information in the EBL concept's distribution. Augmenting the actual examples up to a set of 1000 rather than 5000 performed well but discernibly worse. We did not try larger sets than 5000.

Somewhat surprisingly, there is no apparent accuracy penalty for including the EBL component. We had expected a residual bias due to the (extremely) approximate nature of the prior knowledge. We predicted that as the training set of actual examples grows without bound, its information content closely approximates the target concept. At some point, the residual bias of the prior knowledge should influence the learner away from rather than toward the target concept. While this phenomenon may well occur, it seems not to appear at this level of training.

4.3.1.2 Comparison with Transformation Invariants

Schölkopf, Burges and Vapnik (1996) demonstrate another method for generating synthetic examples. The function to be learned may be known to be invariant under some transformations. Additional training examples can then be generated by applying such transformations to the actual training examples. For handwritten digits, the label should be the same for small translations and rotations. New training examples can be formed by translating and/or rotating the originals.

To compare the transformation invariant approach with the EBL phantom example approach, we again trained the SVM as in the phantom condition but this time using small translations and rotations applied to the actual training examples, to pad the training set to 5000 instances. We call this the “transformed” condition. We also introduce a “both” condition in which half of the synthetic examples are derived from translation/rotation invariants and half from EBL phantom examples. Thus, in all but the native condition, the SVM is trained on 5000 labeled examples. The combined results of the four conditions are shown in Figure 4.9A. Once again accuracy is assessed using 2000 held-back labeled examples, 1000 for each class. Again, the reported accuracy is the mean of five learning runs with 95% confidence intervals.

Not surprisingly, the EBL approach performs better than the transformational invariant approach when the actual training set is small. Interestingly, the transformational approach surpasses the phantom approach at the level of approximately 100 actual training examples. Perhaps transformed examples allow the SVM to extract more information from the actual training examples. The crossing may be the expected manifestation of the residual domain knowledge bias.

The “both” condition is particularly intriguing. It is always competitive and nearly always yields the best accuracy. This indicates that at each explored training level, the

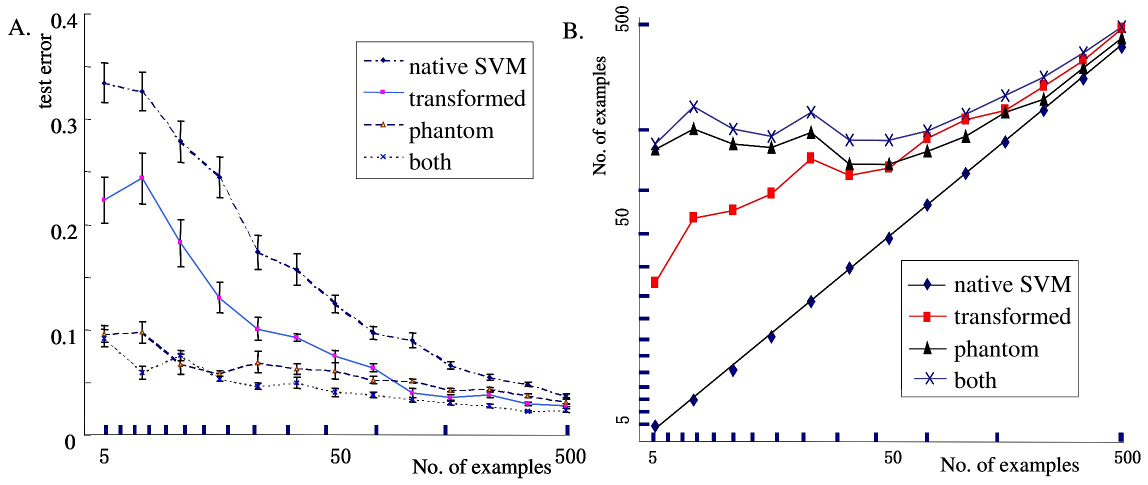

Figure 4.9 Learning behaviors for SVMs with different sets of examples. Part A shows that the learning curves, with the error bars indicate 95% confidence intervals. Part B plots the number of real examples required for native SVM to achieve the same error rate as SVM trained with different augmented training sets.

two approaches to synthetic example generation are complementary with little or no negative interaction.

Figure 4.9B provides the results in a somewhat different form. Here we ask how much information does each condition add to the actual training examples. This information is quantified in terms of effective native condition training set size. For each condition the independent variable represents the number of actual training examples used. The dependent variable shows the approximate number of actual training examples that would be required by the native SVM to achieve a similar accuracy. Computing the dependent variable requires inverting the training examples to accuracy mapping for the native condition. To do this we fit a cubic curve to the thirteen points representing the native condition. Thus, the inversion and the plotted points represent a smoothed approximation to the actual values. Once again we see the advantage of the EBL approach when the available actual training data is limited. We also see that the information augmented with phantom examples is always higher than without phantom examples, and that the greatest advantage is derived from combining phantom and transformed examples.

4.3.1.3 CONTROL CONDITIONS

The SVM inductive learner, the actual training examples, and the phantom EBL examples form three foundational supports of our approach. Are they all needed? We explore two control conditions to demonstrate that all components are required for good performance.

The first control tests the need for actual training data. One criticism of previous EBL formalizations is that training examples are superfluous. In these formalizations, rooted in logic, any proof that could be constructed using a training example could also be constructed without it. The data of the previous graphs show that the phantom condition performs quite well with only five actual training examples. Perhaps the advantage is derived from the domain theory alone and the actual training examples are again superfluous. In this control condition we train the SVM with 5000 phantom examples generated directly from the domain theory with no explanations of actual examples and no parameter fitting. Instead, phantom examples are generated uniformly from the full range of values that each parameter can take on.

The second control condition addresses whether the SVM is superfluous. Here an EBL concept is produced for each classification using 500 actual training examples (250

4’s and 250 9’s). Classification of new examples is performed by explaining the new example twice, once for each classification, and judging similarity to each EBL concept. For the similarity metric we used the average standard Z score [Freedman, Pisani, & PurvesStatistics, 1997] of the EBL concept’s parameters. This allows a comparison even though the two EBL concepts have different numbers of parameters.

The results are shown in Table 1.1. For comparison, we also include the training accuracy of the four experimental conditions similarly trained on 500 actual examples. It is clear that the control conditions perform very poorly indeed, demonstrating that all three components, the SVM, the actual training examples, and the phantom training examples are important for our EBL approach.

Table 4.1. Classification accuracy for different training methods. The error intervals represent 95% confidence level.

TRAINING TYPE	TEST ERROR
EBL + SVM	0.426 ± 0.054
EBL + TRAINING EXAMPLES	0.223 ± 0.032
NATIVE SVM	0.037 ± 0.0018
TRANSFORMED	0.028 ± 0.0017
PHANTOM	0.032 ± 0.0022
BOTH	0.024 ± 0.0016

4.3.2 HANDWRITTEN CHINESE CHARACTER DOMAIN

We chose two similar Chinese characters, *wu* and *hu* (Figure 2.1), to demonstrate the phantom example approach could benefit real-world classification problem. These two Chinese characters, although only contains few strokes, are similar enough in pixel level to provide a challenging problem for SVMs. The difficulty lies in that they share many common strokes. The pixels that realize those strokes, therefore, are irrelevant for classification purpose. For SVMs operating on high dimensional space, such irrelevant features are plenty enough to provide false patterns for the learner. Yet, our domain knowledge is helpful enough to capture the stroke-level similarity between these two characters. We expect that the EBL concept presented in our stroke-level models can exhibit the stroke-level similarity, and large sample of phantom examples are able to communicate our belief that any patterns in the irrelevant strokes are the noise we should avoid to use. The difficulty of the problem and the availability of useful domain knowledge make our approach suitable for handwritten character domain.

We compare three approaches. The first, the "native" condition, is a control condition. Only the 150 real examples are given to the SVM learning system. In the second, or "virtual" condition, spatial transformations are applied to the support vectors found in native SVM. The training set is then augmented with the resulting virtual examples and used to train the SVM. Virtual examples are the result of random translations (–3 to 3 pixels in each direction) and a random rotation (–10 to 10 degrees). These are larger transformations than used in [Schölkopf, Burges & Vapnik 1996]. Our image resolution is greater than theirs and pilot studies showed an improvement in the virtual condition with these larger transformations. In the third condition, called the "phantom" condition, our EBL approach is used to generate phantom examples as described in the previous section.

There are a total of 200 labeled examples for each character in the database. The learning curves are produced by a 10-fold cross validation each with 150 random examples selected as the training set and 50 examples held back as a test set to evaluate the performance.

We compare the performance of the native SVM to the virtual and phantom conditions varying the number of synthetic examples. The native SVM is run on a single set of 10 cross validation trials. The experimental conditions were run on 10 new cross validation trials for each of 100, 200, 400, 800, and 1600 synthetic examples which correspond to 250, 350, 550, 950, and 1750 total training examples. In each of the 10 learning trials, the same set of real examples was given to the two experimental conditions, and the same test examples were used to evaluate accuracy. In all three conditions, we used the same polynomial kernel function: $k(\vec{x},\vec{y}) = (\vec{x}\cdot\vec{y}+1)^3$ for SVM learning system.

Figure 4.10 shows the accuracy after each training episode. Error bars denote the 95% confidence interval about each mean accuracy point. It should be noted that these error bars slightly underestimate the true confidence interval as their computation assumes the cross validation runs embody truly independent samplings. As predicted, the phantom condition performs significantly better than the native SVM. It also consistently outperforms the virtual SVM.

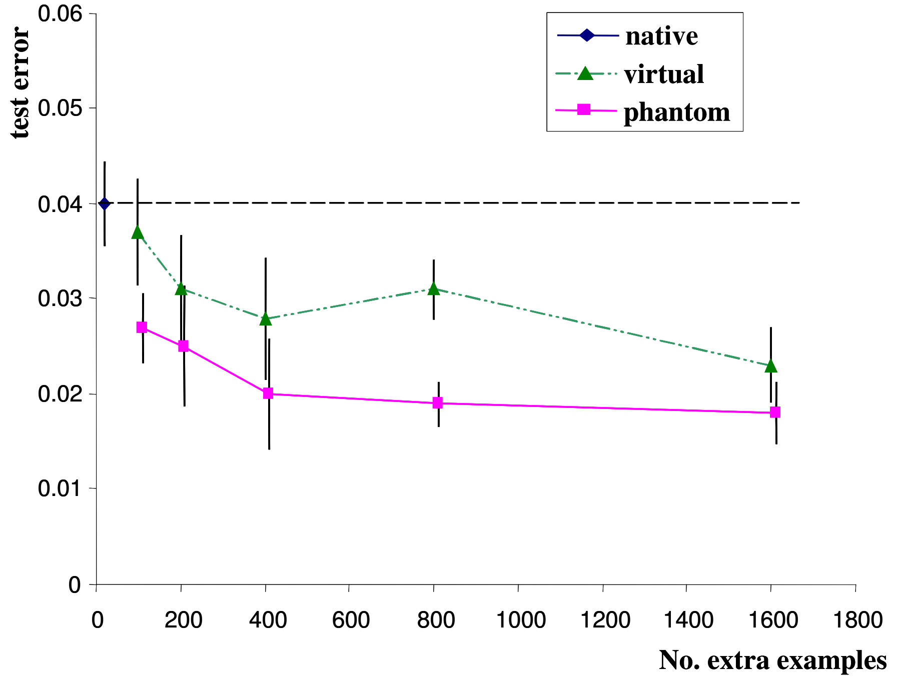

Figure 4.10 Comparison of EBL approach with native SVM and virtual SVM for classifying two Chinese characters.

It is clear from figure 4.10 that domain knowledge when applied in an EBL paradigm, even if the knowledge is only approximate, can be effective in improving the small training-sample behavior of an SVM.

Is it significant that the phantom condition outperforms the virtual condition? To answer this we consider their sources of additional evidence that each brings to the search in the hypothesis space. The additional evidence embodied by the virtual examples consists of two relatively independent components. First, is the invariance information: small transformations should not alter an example's classification label. Second, is the knowledge of the support vectors produced by the native SVM; this knowledge allows the virtual examples to be concentrated where they will do the most good. Phantom examples, on the other hand, embody the domain knowledge of character strokes interacting with the raw original training examples. In this application the augmented evidence from the phantom condition is apparently greater than the augmented evidence from the virtual condition. However, there is nothing deep in this statement. If the domain theory were a little more approximate or if the original classifier had found more support vectors (due perhaps to a different kernel function), the relative performance of the two could change.

CHAPTER 5

Feature Kernel Approach

Kernel functions are often mentioned as the mechanism by which prior knowledge may be incorporated into an SVM classifier. Certainly, kernel choice is a large part of an SVM's inductive bias; the kind of kernel that is employed has an impact on both its learning rate and its final performance. But the relationship of these to kernel choice is complex. Designing effective special-purpose kernel functions for specific classification applications remains a challenging problem.

It is particularly difficult to directly incorporate high-level, general domain knowledge into kernel functions. In handwritten character recognition domain, stroke-level knowledge suffers a vocabulary mismatch with pixel description of kernel functions. Although it is possible to extract stroke-level features from the images by human written programs, such process is often error-prone [Shi, Gunn, & Damper 2003]. It is more desirable to design a pixel-level kernel function with stroke-level knowledge in mind.

We employ an Explanation Based Learning (EBL) approach to effectively and automatically construct a special-purpose kernel function that translates high-level knowledge into relevant input features. Domain knowledge about strokes, how they can interact, and how they realize prototype Chinese characters is used to conjecture high-level, high-information-bearing domain features such as "intersecting strokes" or "abutting strokes", which can be further evaluated, calibrated, and mapped into their pixel realizations using the explanations of training examples. The result is a special-purpose

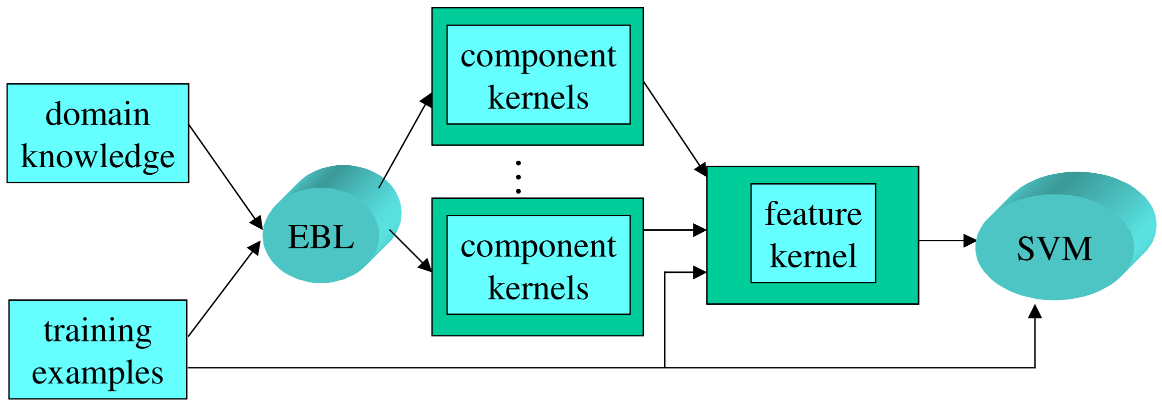

Figure 5.1 The feature kernel approach to implement the new EBL learning framework.

selection kernel that emphasizes relevant interactions among pixels. Empirical results show that the acquired kernel functions can significantly improve the performance of an SVM.

Figure 5.1 illustrates how feature kernel function approach fits to our proposed new EBL learning framework. EBL component uses domain knowledge to suggest salient high-level features. It also builds explanations and uses explanations to map high-level features into low-level features. The mapping is used to define a specialized kernel function for each high-level feature. These kernels are called *component kernels*, which can then be combined to build the final kernel function, called *feature kernel function*. The feature kernel function serves as solution knowledge to introduce bias into SVM learning algorithm. Of course, the SVM learning algorithm can be replaced by any other kernel-based methods, such as kernel perceptron [Shawe-Taylor & Cristianini. 2004].

5.1 Overview of Feature Kernel Approach

Intuitively, a good kernel function should define a space in which inter-example distances reflect the intrinsic differences between classes. Such kernel functions highlight features of the examples which are most informative of class membership while de-emphasizing less informative ones. Informative features are highly discriminative, reliably detected in examples from one class, and reliably missing from examples of the other.

To illustrate, consider distinguishing between images of handwritten "3" and "8" digits. We know that the left part of the figure is likely to be more informative than the right. This stems from our prior knowledge of the individual digits together with the fact that this task involves distinguishing "3" and "8" rather than, say, "9" and "8". We might call this a "figure bias" since it applies to the actual "3" and "8" figures rather than, say, their pixel representations. Clearly, to be used in a kernel function, this figure bias must be expressed in the image vocabulary. But the particular combinations of pixels or image regions that best reflect this figure bias depends on the underlying population of examples: how they are represented, how effectively they are normalized and registered, the amount and characteristics of noise, and so on. Thus, an effective specialized kernel function results from the interaction of three sources of knowledge: intra-class knowledge (e.g., the features that compose an "8"), inter-class knowledge (e.g., the subset reliable features that distinguish "8" from "3"), and example characteristics (e.g., which pixels can reliably indicate the presence of informative features).

Since explanation-based learning supports just this sort of knowledge interaction, we explore how to apply EBL approach to adapt a kernel function for the given classification task.

Here, an explanation is not a logical proof but only general conjecture of how the high-level features are realized by low-level features. It then forms what we will call a component kernel, which is a special kernel designed to detect a newly invented high-level feature. A collection of weighted component kernels defines a feature kernel function. It specifies a metric space which embodies the optimal linear blending of the informative features as detected by the component kernel functions. Thus, a feature kernel function is tailored to a specific classification task with awareness of high-level features. It integrates knowledge about the classes, the discrimination task, and the exemplar representations. It cannot, therefore, have been created by the human designer. Incorporating additional sources of information and appreciating their interaction with the training examples can result in faster learning and more accurate classifiers.

Our approach, incidentally, agrees with divide-and-conquer principle. The invented high-level features conceptually divide the big classification problem into many small problems. Recognizing each of those high-level features solves part of the big problem, and is much easier. In our approach, the component kernels are invented to detect those high-level features. They are assembled together to form the final feature kernel function. Although our approach does not build multiple classifiers, the specialized feature kernel function guides the SVM learning algorithm to automatically build the final classifier through the detection of those high-level features.

王 玉 五 互

且 旦 亘

己 巳 巴

Figure 5.2 Ten Chinese characters used in the experiments.

We exercise our approach distinguishing images of handwritten Chinese characters. The dual characteristics of complex examples and limited training data fit our approach well and apply to many real-world classification and discrimination tasks. We use Chinese character discrimination only as an illustrative domain. We have chosen just a few Chinese characters to show the approach. It would be unwieldy to learn all pair-wise SVM classifiers in this domain. Furthermore, our approach applies generally to images of other hand-drawn characters and objects where similar prior domain knowledge is available. The ten different Chinese characters used in this study are shown in Figure 5.2. They include several similar characters drawn from each of three different radical classes resulting in a range of easy to difficult binary classification tasks.

Compared to a conventional SVM of similar design, our results show that 1) a more accurate classifier is acquired, 2) an effective classifier can be acquired with significantly fewer training examples, and 3) the resulting classifier is significantly less complex, employing fewer support vectors. The approach helps the most in challenging classification tasks. Perhaps this is not surprising; EBL opens the door to additional

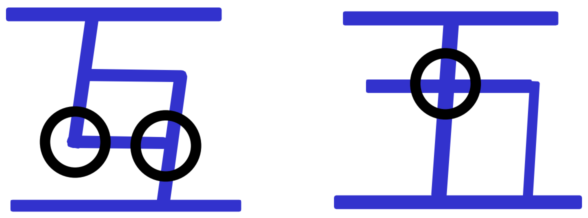

Figure 5.3 An example of two similar Chinese characters with different between them highlighted.

information which may be superfluous for simple classification tasks. Importantly, the EBL use of prior knowledge seems never to harm performance.

5.2 Feature Kernel Function for Chinese Character Classification

EBL requires a domain theory of prior knowledge to drive the explanation process. While images of characters are composed of pixels, it is natural to break the span from pixels to characters into two sub-domains. The first relates handwritten characters to the strokes that are used to form them. The notion of a stroke is not intrinsic to this classification problem. A conventional SVM has no place for such a notion. Rather, strokes are introduced as “hidden” features to organize prior knowledge. In addition, combinations of strokes form yet another level of derivable hidden features, called *stroke-level features*. For example, the two prototype Chinese characters shown in the Figure 5.3 are quite similar. Both include horizontal strokes at the top and bottom, making these features uninformative for this discrimination task. However, other stroke-level features are quite informative. We have circled these in Figure 5.3. The right character (wu) has a unique “cross” while the left one (hu) has an “L-corner” and a specific “T” junction not shared by the other. Described with these derived stroke-level features the characters are quite different. The prior knowledge in this sub-domain consists of a prototype of ideal stroke model for each Chinese character, a width parameter that applies to all of the strokes within a character, and a simple but general “grammar” of how strokes can interact.

The second sub-domain explains the correspondence between pixels and strokes. Strokes are modeled as straight lines of a particular width with a particular starting and ending location. The corresponding pixels are those that fall within the boundaries of a long thin rectangle which is the stroke. Similar to the explanation building procedure in phantom example approach, a Hough transformation is used to determine the correspondence of stroke features and pixels in the training images.

Given a pair of Chinese character labels and a set of training examples for each, the following procedure is performed to produce a feature kernel function:

1) Conjecture stroke-level features: The prototypes of the characters to be distinguished are examined, and distinctive stroke interactions are identified. These are candidates for the construction of component kernel functions.
2) Determine the best pixel representation for each stroke: This involves
    a) Explaining labeled character images by using Hough transform to determine how each required stroke is realized by its pixels.
    b) Estimating the correlation between pixels and strokes across the data set. The detail is discussed in section 5.6.2.

   c) Constructing parameterized evidence pixel sets for each stroke. The parameter is a threshold specifying the minimal acceptable correlation for a pixel to be included.

3) Given evidence pixel sets for strokes, construct the set of component kernel functions. This is the topic of section 5.4. It also involves choosing values for the parameters mentioned above. They are evaluated so as to minimize stroke-level feature detection errors over whole training set.
4) Build the feature kernel function: Using the algorithms described in section 5.5, component kernel functions of the previous step are weighted. Their weighted sum is the final feature kernel function.

5.3 Component Kernel Functions

As described, stroke-level features represent interactions between strokes. A component kernel is a special polynomial kernel computed on a limited region within an image. It serves as a detector for its stroke-level feature, and is used to assemble the final feature kernel function. Our component kernels maps examples to a space spanned by monomials. A *monomial* is a term in the expansion of a polynomial kernel function. It represents the product of pixels, as evidence for a stroke-level feature.

Given the connections between pixels and strokes, it is straightforward to determine evidence monomials for different stroke-level features. This is illustrated in figure 5.4. When the pixels in the region A serve as evidence for a horizontal stroke, and the pixels in the region B serve as evidence for a vertical stroke, then those second-degree

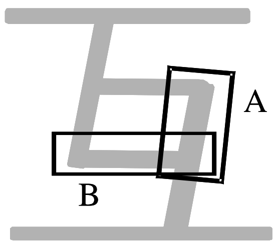

Figure 5.4 Two stroke-level feature regions A and B to detect a corner.

monomials with one pixel from region A and another from region B can be evidence for the "corner" feature composed with the horizontal and vertical strokes.

Specifying a function to compute the dot product between two examples using those evidence monomials gives us a component kernel function for SVMs to detect the corresponding stroke-level features. The corner feature shown in figure 5.4, employs a component kernel function like the following:

$$k_{A,B}(\vec{x}_1, \vec{x}_2) = (\sum_{i \in A} x_{1i} x_{2i})(\sum_{i \in B} x_{1i} x_{2i})$$

It is easy to see that this kernel function computes the dot product between example x_1 and x_2 using the monomials of pixels from region A and region B:

$$\begin{aligned} k_{A,B}(\vec{x}_1, \vec{x}_2) &= (\sum_{i \in A} x_{1i} x_{2i})(\sum_{i \in B} x_{1i} x_{2i}) \\ &= \sum_{i \in A, j \in B} x_{1i} x_{2i} x_{1j} x_{2j} \\ &= \sum_{i \in A, j \in B} (x_{1i} x_{1j})(x_{2i} x_{2j}) \end{aligned}$$

Therefore, a component kernel function is similar to a conventional kernel function in that it is easy to compute, and it gives the value of dot product between two examples in high-dimensional space. The only difference is that a component kernel function is designed for discovering the presence of certain high-level feature in the example, not for determining the label. We can also build higher order component kernel functions for the same corner feature. Here are two third-degree component kernel functions:

$$
\begin{aligned}
K_{A,A,B}(\vec{x}_1, \vec{x}_2) &= (\sum_{i\in A} x_{1i}x_{2i})(\sum_{i\in A} x_{1i}x_{2i})(\sum_{i\in B} x_{1i}x_{2i}) \\
&= \sum_{i\in A,\, j\in A, k\in B} (x_{1i}x_{1j}x_{1k})(x_{2i}x_{2j}x_{2k})
\end{aligned}
$$

$$
\begin{aligned}
K_{A,B,B}(\vec{x}_1, \vec{x}_2) &= (\sum_{i\in A} x_{1i}x_{2i})(\sum_{i\in B} x_{1i}x_{2i})(\sum_{i\in B} x_{1i}x_{2i}) \\
&= \sum_{i\in A,\, j\in B, k\in B} (x_{1i}x_{1j}x_{1k})(x_{2i}x_{2j}x_{2k})
\end{aligned}
$$

The first kernel uses monomials with three pixels, two from region A, and the other one from region B. The second function uses monomials with one pixel from region A, and the other two from region B.

5.4 Combining Component Kernel Functions

The component kernel functions are designed to detect stroke-level features in characters, but not all features are equally useful to distinguish characters. In this section, we examine how to use a linearly weighted combination of component kernel functions to obtain a feature kernel function to classify future character examples. Using different weights, our feature kernel function is able to emphasize diagnostic stroke-level features, while de-emphasizing less informative or redundant ones.

Weighting kernel function is a question addressed by many other researchers [Kandola, Shawe-Taylor, & Cristianini. 2002; Kwok, & Tsang. 2003; Lanckriet etc. 2004]. In our study, we employ the approach proposed by [Kandola, Shawe-Taylor, &

Cristianini. 2002]. Here we briefly summarize the approach. It uses a simple computable notion, called *alignment*, to estimate the goodness of a kernel function. The (empirical) alignment of a kernel k_1 with another kernel k_2 with respect to the sample S is defined as:

$$A_S(k_1, k_2) = < K_1, K_2 >_F /(\sqrt{< K_1, K_1 >_F}\sqrt{< K_2, K_2 >_F}),$$

where K_i is the kernel matrix for the sample S using kernel k_i, and $< \cdot, \cdot >_F$ is the Frobenius product[1]. Let y be the target function that gives the label, then the target kernel k^* can be defined as $k^*(\bar{x}, \bar{z}) = y(\bar{x})y(\bar{z})$. The (empirical) *kernel target alignment* can be defined as $A_S^*(k) = A_S(k, k^*)$, which measures the degree of agreement of a kernel with a learning task. High kernel target alignment implies good generalization performance of the resulting SVM classifier [Cristianini etc. 2002].

Given a set of kernels k_i, the problem of combining them can be formulated as to choose α in $k(\alpha) = \sum_i \alpha_i k_i$ so that the alignment of $k(\alpha)$ to the given target vector y is optimized. With constraining $||\alpha||$ to avoid over-fitting the alignment to the training set, the optimization problem can be solved by the following quadratic programming:

$$\begin{aligned} &\max : \sum_i \alpha_i y' K_i y - \sum_{i,j} \alpha_i \alpha_j < K_i, K_j >_F - \lambda \sum_i \alpha_i^2 \\ &\textit{subject to} : \alpha_i \geq 0 \end{aligned}$$

where λ is a parameter controlling the balance between optimizing alignment and constraining $||\alpha||$. The value of it can be adjusted using cross-validation.

[1] The Frobenius product between two matrices M=[m_{ij}] and N=[n_{ij}] is defined by $< M, N >= \sum_{ij} m_{ij} n_{ij}$

5.5 A Formal Analysis of Feature Kernel Approach

Our algorithm uses the notion of stroke-level features to organize pixel level features, and enables SVMs to emphasize the distinctive stroke-level features by weighting component kernel functions. Intuitively, emphasizing small amount of useful features can greatly reduce learning complexity, therefore allowing SVM learning algorithm to perform well even when small number of training examples are available. In this section, we use the notion of kernel alignment to provide insights about how our algorithm can produce an improved kernel function.

5.5.1 Decomposing a Kernel Function

First, we observe that weighting multiple kernel functions can usually produce a better kernel function than simply combining them with equal weights. Specifically, given two kernel functions k_1 and k_2, the target alignment of the combined kernel function with coefficient *1* and w is:

$$\begin{aligned} A(w) &= \frac{< K_1 + wK_2, yy'>}{\sqrt{< K_1 + wK_2, K_1 + wK_2 >} \cdot \sqrt{< yy', yy'>}} \\ &= \frac{< K_1, yy'> + w < K_2, yy'>}{\sqrt{< K_1, K_1 > + 2w < K_1, K_2 > + w^2 < K_2, K_2 >} \cdot \sqrt{< yy', yy'>}} \end{aligned}$$

To determine the optimal w to maximize $A(w)$, we set $\frac{dA(w)}{dw} = 0$, we have

$$\frac{< K_2, yy'>}{(< K_1, K_1 > + 2w_{opt} < K_1, K_2 > + w^2 < K_2, K_2 >)^{1/2}} - \frac{(< K_1, yy'> + w_{opt} < K_2, yy'>) \cdot (< K_1, K_2 > + w_{opt} < K_2, K_2 >)}{(< K_1, K_1 > + 2w_{opt} < K_1, K_2 > + w_{opt}{}^2 < K_2, K_2 >)^{3/2}} = 0$$

Multiply both sides by $(<K_1,K_1>+2w<K_1,K_2>+w^2<K_2,K_2>)^{3/2}$, we have:

$$<K_2,yy'>(<K_1,K_1>+2w_{opt}<K_1,K_2>+w_{opt}{}^2<K_2,K_2>)$$
$$-(<K_1,yy'>+w_{opt}<K_2,yy'>)\cdot(<K_1,K_2>+w_{opt}<K_2,K_2>)=0$$

With little algebra, we see that:

$$w_{opt}=\frac{<K_2,yy'>\cdot<K_1,K_1>-<K_1,yy'>\cdot<K_1,K_2>}{<K_2,K_2>\cdot<K_1,yy'>-<K_2,yy'>\cdot<K_1,K_2>}.$$

Let *A_1, A_2* denote target alignment for *k_1* and *k_2, A_{12}* denotes alignment between *k_1* and *k_2*, and $|K_i|=\sqrt{<K_i,K_i>}$, we can write $w_{opt}=\frac{(A_2-A_{12}\cdot A_1)}{(A_1-A_{12}\cdot A_2)}\cdot\frac{|K_1|}{|K_2|}$.

In the above equation for the optimal weight, $\frac{|K_1|}{|K_2|}$ balances the scale of two kernels. While the factor $\frac{(A_2-A_{12}\cdot A_1)}{(A_1-A_{12}\cdot A_2)}$ measures the difference of the target alignments between two kernels and modulates the overlap between them. When the ratio $\frac{A_2}{A_1}$ is large, the optimal weighting is far from an equal blending.

Furthermore, with some algebra, we can show that the target alignment of the optimally weighted kernel is $A(w_{opt})=\frac{\sqrt{A_1{}^2+A_2{}^2-2A_{12}A_1A_2}}{\sqrt{1-A_{12}{}^2}}$, while the target alignment of equally weighted kernel is

$$A(1)=\frac{A_1}{\sqrt{1+2A_{12}\frac{|K_1|}{|K_2|}+\frac{|K_1|^2}{|K_2|^2}}}+\frac{A_2}{\sqrt{1+2A_{12}\frac{|K_2|}{|K_1|}+\frac{|K_2|^2}{|K_1|^2}}}.$$

When the scales of the two kernels are normalized, meaning $\frac{|K_1|}{|K_2|}=1$, it follows that $A(1)=\frac{A_1+A_2}{\sqrt{2+2A_{12}}}$, therefore:

$$\begin{aligned}
A(w_{opt})^2 - A(1)^2 &= \frac{A_1{}^2 + A_2{}^2 - 2A_{12}A_1A_2}{1-A_{12}{}^2} - \frac{(A_1+A_2)^2}{2+2A_{12}} \\
&= \frac{2A_1{}^2 + 2A_2{}^2 - 4A_{12}A_1A_2}{2(1-A_{12}{}^2)} - \frac{(A_1+A_2)^2(1-A_{12})}{(2+2A_{12})(1-A_{12})} \\
&= \frac{2A_1{}^2 + 2A_2{}^2 - 4A_{12}A_1A_2 - (A_1+A_2)^2(1-A_{12})}{2(1-A_{12}{}^2)} \\
&= \frac{A_1{}^2 + A_2{}^2 - 2A_{12}A_1A_2 - 2A_{12}A_1 + A_1{}^2A_{12} + A_1{}^2A_{12}}{2(1-A_{12}{}^2)} \\
&= \frac{(A_1-A_2)^2 + A_{12}(A_1-A_2)^2}{2(1-A_{12}{}^2)} \\
&= \frac{(A_1-A_2)^2}{2(1-A_{12})}
\end{aligned}.$$

This shows that the difference between optimally weighted kernel and equally weighted kernel becomes large when the ratio of their target alignment increases.

The above arguments suggest a better kernel function can be obtained from a re-composition of the original kernel function as an appropriately weighted set of kernels. In our approach, since combining all component kernel functions by equal weights approximately reproduce conventional polynomial kernel function, specifying component kernel functions according to stroke-level features serves as a mechanism to decompose the conventional polynomial kernel function. Weighting component kernels offer a mechanism to build better kernel function.

5.5.2 High-level Feature Alignment

Without prior knowledge to guide the processes, a decomposition is unlikely to produce kernel functions with significantly different alignments. On the other hand, knowledge of high-level features can help to group input features according to their information content. In particular, if we assume the high-level features have binary values, then we can define the alignment between a high-level feature label and a given kernel matrix. We call this its *high-level feature alignment*. We use $A_F(.)$ to denote alignment of a kernel to high-level feature, and $A_t(.)$ to denote alignment of a kernel to the target label. For a high-level feature that is unique for its class, its label has perfect alignment with the class label. Therefore, the corresponding high-level feature alignment equals the target alignment, which means $A_F(K) = A_t(K)$.

On the other hand, for a high-level feature that is common to both classes,

$A_F(K) = \frac{\sum_{i,j} K_{ij} \cdot f_i f_j}{|K| \cdot |ff'|}$, while $A_t(K) = \frac{\sum_{i,j} K_{ij} \cdot f_i f_j}{|K| \cdot |yy'|}$, where $f(i)=1$, and $y(i)=\{1,-1\}$.

Since $|yy'| = |ff'|$, we have:

$$A_F(K) + A_t(K) = \frac{\sum_{i,j} K_{ij} \cdot (f_i f_j + y_i y_j)}{|K| \cdot |yy'|} = \frac{\sum_{y(i)=y(j)} 2K_{ij}}{|K| \cdot |yy'|}$$

Now define a new matrix $T = \begin{cases} 1, \textit{ when } y(i) = y(j) \\ 0, \textit{ when } y(i) \neq y(j) \end{cases}$, then $|yy'| = 2|T|$, we have:

$$A_F(K) + A_t(K) = \frac{\sum_{y(i)=y(j)} 2K_{ij}}{|K| \cdot |yy'|} = \frac{<K,T>}{|K| \cdot |T|} = A(K,T) < 1 .$$

Therefore $A_t(K) < 1 - A_F(K)$. Thus, high alignment to a non-informative high-level feature results, not surprisingly, in a low alignment to the class label.

The above analysis shows that constructing kernel functions that align with high-level features can help us to organize input features (like the pixels of our images) into informative sets and non-informative sets. Now we examine those input features *x* that exhibit high alignment with some high-level feature *f*. Assume we have binary input features *x={0,1},* then:

$$
\begin{aligned}
A_F(x) &= \frac{\sum_{i,j} x_i x_j \cdot f_i f_j}{\sqrt{\sum_i x_i^2 f_i^2} \cdot | ff'|} \\
&= \frac{\sum_i x_i f_i \cdot \sum_i x_i f_i}{\sqrt{\sum_i x_i^2 \cdot \sum_j x_j^2} \cdot | ff'|} \\
&= \frac{(\sum_i x_i f_i)^2}{\sum_i x_i^2 \cdot | ff'|} \\
&= \frac{(\sum_{f_i=1} x_i - \sum_{f_i=-1} x_i)^2}{N(x_i = 1) \cdot | ff'|} \\
&= \frac{\left(\sum_{f_i=1} x_i \Big/ n - \sum_{f_i=-1} x_i \Big/ n \right)^2}{\left(N(x_i = 1) \cdot n \big/ n^2 \right)} \\
&= \frac{\left(p(x=1, f=1) - p(x=1, f=-1) \right)^2}{p(x=1)}
\end{aligned}
$$

In the above equation, $p(x=1, f=1) - p(x=1, f=-1)$ represents the correlation of *x* with high-level feature *f*. The input features that exhibit high correlation with *f* provide a set of input features that can reliably detect high-level feature *f*, while at the same time they exclude irrelevant input features. In our experimental section, we evaluate $p(x=1, f=1) - p(x=1, f=-1)$ to determine evidence pixel set for each stroke, and specify component kernel functions.

5.6 Empirical Results

5.6.1 Experiment Setup

In this section, we present our empirical results to demonstrate the accuracy improvement made by incorporating stroke-level knowledge into kernel functions for SVM learning algorithms. In our experiments, we used handwritten Chinese image examples from the ETL9B database created by Electrotechnical Laboratory of Japan. It contains 200 samples for each character. All examples are binary images of size 64 by 63. We applied simple image processing to smooth the images then center them to fit a 64x64 area.

We used *libsvm* package [Fan, Chen, & Lin C.-J. 2005] as our SVM learning algorithm. To simplify the comparison of kernel functions, we used the default value for all the SVM parameters, except those for kernel functions, during the learning.

We want to compare our specialized kernel function with a conventional polynomial kernel function. We have found that the optimal degree of a conventional polynomial kernel for the ETL9B dataset is 3. To make a fair comparison, we also used third degree polynomials to define component kernel functions and, therefore, the derived feature kernel functions. This is an artificial limitation that compromises our approach for the sake of a clearer comparison. For stroke-level features that are composed of more than three strokes, such a “cross”, we need four pixels (one in each one of the strokes) to compose a good detector, but third-degree monomials can only compute the product of three pixels. Of course, several three-stroke features can be found that together approximate the four-stroke combination. This ameliorates but does not eliminate the disadvantage. We believe a fourth-degree polynomial may result in better performance of our approach in this task.

5.6.2 EXPERIMENT 1: DOES A FEATURE KERNEL OUTPERFORM A SIMILAR CONVENTIONAL KERNEL?

The first question we ask is following: Do feature kernel functions offer a significant improvement over similar conventional polynomial kernel functions when the same order polynomial is applied?

5.6.2.1 EXPERIMENT 1A: BEHAVIOR ON A SINGLE CHALLENGING DISCRIMINATION TASK.

To compare two kernel functions, we first selected two very similar Chinese characters (shown in Figure 5.3) that represent a challenging discrimination task. We believe this sort of challenging task best illustrates the benefit of domain knowledge. These two characters are quite similar, which means it should be possible to decompose a conventional kernel function into informative ones and non-informative ones.

A feature kernel function was constructed for the two characters. It and a conventional third-order polynomial kernel function were evaluated using a leave-one-out test to compare their performance. The number of errors, average number of support vectors and the values of the empirical target kernel alignment are given in Table 5.1. It is clear that, as predicted, the SVM using specialized feature kernel exhibits significantly better performance than the conventional kernel. It also results in fewer support vectors,

Table 5.1. Results on leave-one-out test for classifying two characters using different kernels.

KERNEL	ERROR	NO. SV	ALIGNMENT
FEATURE	4	145	0.1956
POLYNOMIAL	15	197	0.1279

which demonstrates that the feature kernel emphasizes few, but reliable features. This is also supported by the comparison of target kernel alignments for two kernels, where the feature kernel has a significantly higher alignment. Fewer support vectors mean that the corresponding classifier costs less to evaluate on new inputs.

Figure 5.5 shows the efficiency of learning. It plots the error rate with at different stages of training for the two kernels. The feature kernel is significantly better than the polynomial kernel in all cases. Notice that, with 80 training examples, the accuracy of the feature kernel achieves is similar to that of the polynomial kernel with 320 training examples. This can be a crucial improvement when training examples are limited or expensive to produce.

5.6.2.2 EXPERIMENT 1B: BEHAVIOR ON A REPRESENTATIVE SET OF CHINESE CHARACTERS

Experiment 1a shows that the adapted feature kernel function offers significant improvement for classifying two target characters chosen to illustrate its benefits. Is there something specific about these two characters that makes them somehow uniquely liable to feature kernel functions, or the results can generalize to other characters? In particular,

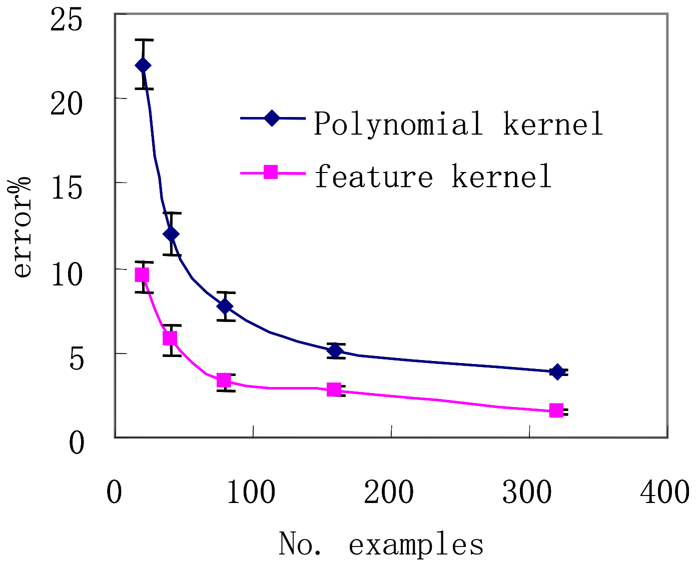

Figure 5.5 Comparing feature kernel with polynomial kernel on the task of classifying two Chinese characters.

does the presence of the domain theory degrade performance on antithetical classification problems which are less challenging?

To answer this question, we selected a set of ten Chinese characters, shown in figure 5.2, from three radical groups resulting in 45 pair-wise discrimination problems. 16 of them are challenging discrimination problems representing characters within the same radical group. Others are easier classification decisions from between radical groups. We believe these tasks form a representative range of discrimination problems in the domain of Chinese characters.

Again we compared feature kernel function against polynomial kernel functions for each classifier. Figure 5.6 shows the results in two different ways: 1). Figure 5.6A and 5.6C plot average error rate and average number of support vectors as a function of the number of the training examples for the two kernels; 2). Figure 5.6B and 5.6D show the

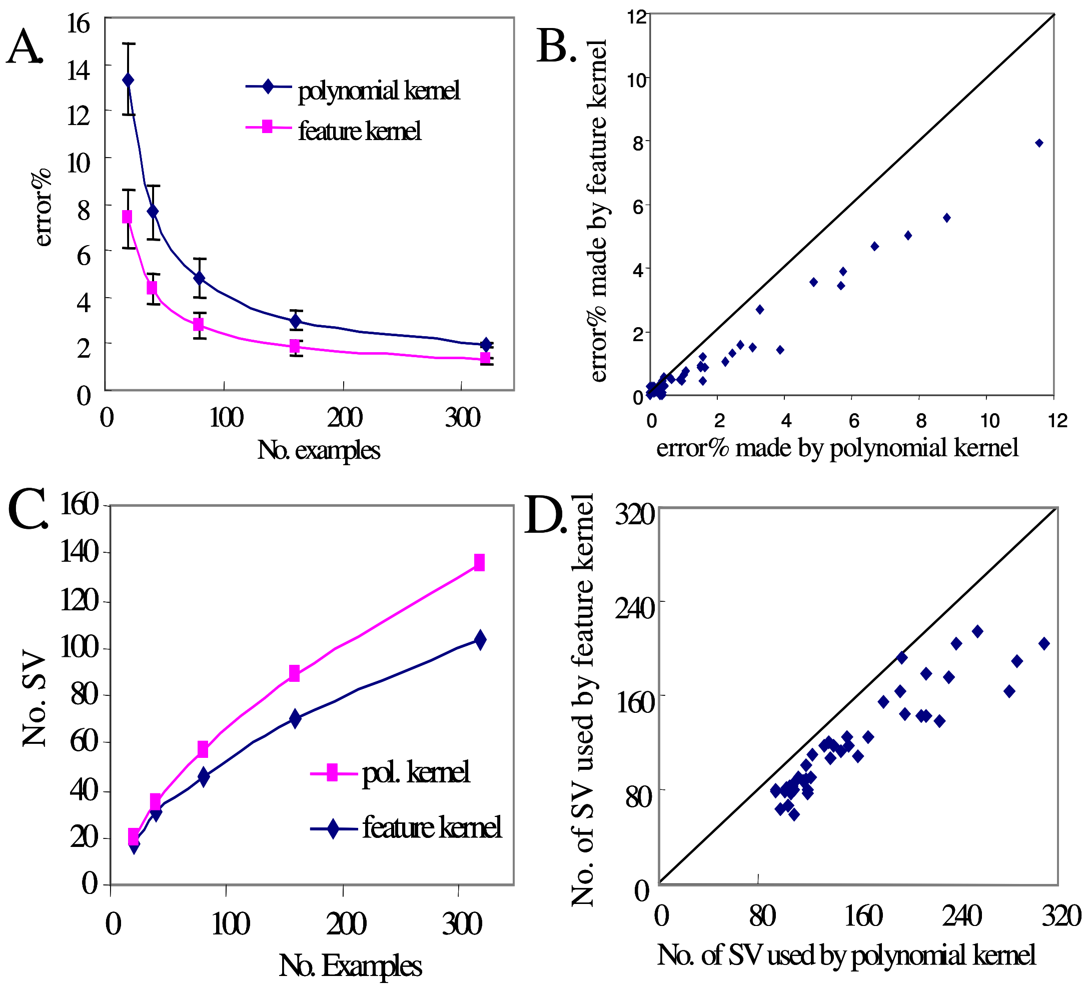

Figure 5.6 Comparing the accuracy and No. of SV used for two kernel functions. A. Plot of average error rate for all 45 binary classifiers against No. of training examples. B. Scattered plot comparing error rate for all 45 binary classifiers. C. Plot of average No. of support vectors used for all 45 classifiers against No. of training examples. D. Scattered plot comparing No. of SV used for all 45 binary classifiers.

comparison in all tasks using a leave-one-out test with 400 examples. We see in almost all cases, across different difficulty-levels and for different number of training examples, that the SVM with feature kernels outperform the one with polynomial kernels. The

SVMs built by feature kernels achieve varying degree of improvements on accuracy, but consistently use fewer support vectors.

5.6.2.3 EXPERIMENT 1B RE-ANALYSIS

What property of the task affects the improvement for specializing feature kernel functions? We believe that additional information in the form of prior domain knowledge may differentially help difficult classification tasks. To investigate, we reanalyzed the data of Figure 5.6A: We evenly divided the 45 classification tasks into three groups according to whether a task easy, medium, or difficult for the conventional SVM learner as judged by the number of errors made by the conventional SVM. Figure 5.7 shows the results: not only does the feature kernel function help, but it indeed appears to help most in the most difficult classification problems.

Finally, we compared multi-class recognition performance in our 10-character problem. We used 160 examples for each character to form the training set to learn all 45 binary classifiers. The remaining 40 examples of each character are held back to form a test set. During the test, each classifier casts one vote, the class with the most votes is assigned as the label to the test example. The result is that the classifiers built with conventional polynomial kernel functions, in this case, have 88.7% recognition accuracy, while the feature kernel functions have 92.6% recognition accuracy.

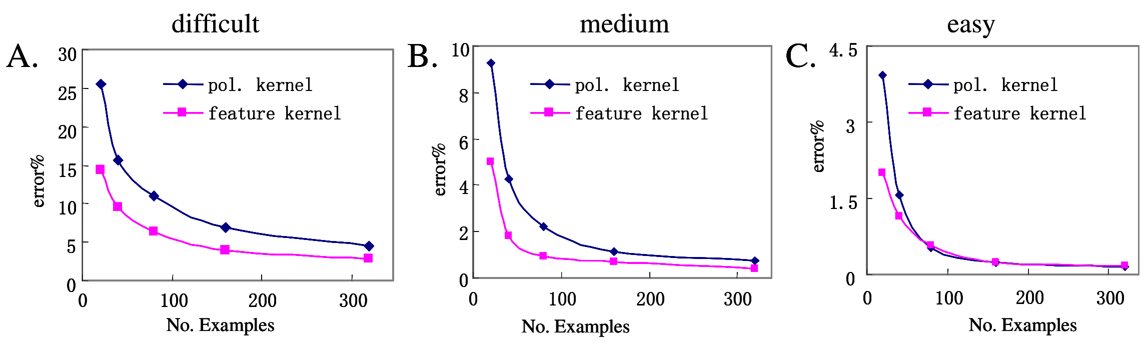

Figure 5.7 Comparing the accuracy used for two kernel functions by three groups of tasks. A. Group of the most difficult tasks. B. Group of the medium difficult tasks. C. Group of the easiest tasks.

5.6.3 EXPERIMENT 2: HOW IMPORTANT IS THE EBL INTERACTION BETWEEN KNOWLEDGE AND EXAMPLES?

The experiments and analyses of our first investigation provide compelling evidence that while the feature kernel function is significantly less expressive than the conventional kernel function, its performance is better. We attribute this improvement to the EBL approach. But is this interpretation fair? Central to EBL is the notion of explanation, an interaction between high-level knowledge and training examples. The EBL approach is quite different than merely supplying the static bias of a conventional machine learner. But perhaps we are mistaken. Perhaps it is the presence of the additional domain knowledge that is significant and not the EBL interaction between domain knowledge and training examples.

To test whether the EBL paradigm underlies the improvement (as we believe) or it is merely the presence of the additional high-level information, we designed a control

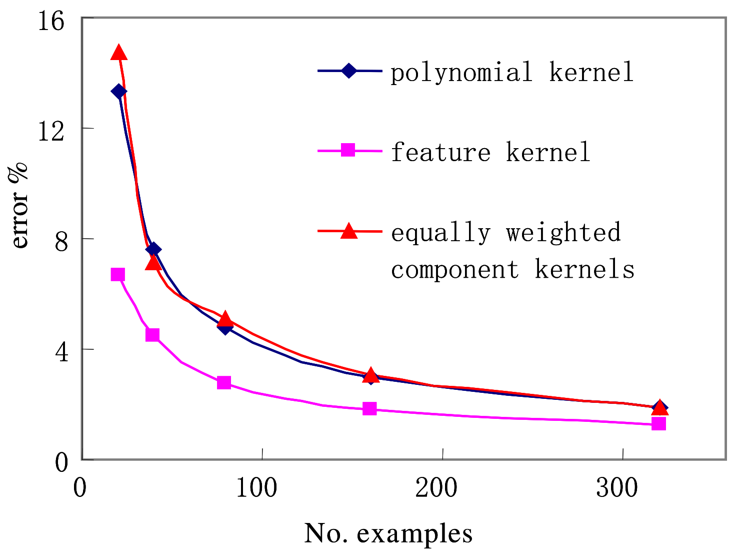

Figure 5.8 Interaction between knowledge and examples provides improvement.

experiment to evaluate a kernel function with the same knowledge of stroke-level features but without the interaction that makes this form of EBL unique. In this EBL, the prior knowledge is not taken to be complete and correct. Rather training examples are used to direct and adjust the interpretation and use of the prior knowledge. In Experiment 2, the component kernel functions are constructed as before, but the feature kernel function is constructed without access to the training examples. The difference between such a kernel function and the weighted kernel function is that it misses the evaluation and adjustment of the conclusions drawn from the prior knowledge using training data. When this interaction is denied to the system, the prior knowledge is simply treated as correct (as in conventional EBL). Figure 5.8 compares this new kernel function with both the conventional third-degree polynomial kernel and previous feature kernel function. It is clear that without training data to guide the EBL system towards significant and useful conclusions, its behavior is similar to the system with no access to prior knowledge at all.

5.6.4 EXPERIMENT 3: CAN OTHER EXAMPLE REPRESENTATIONS ALSO BENEFIT FROM OUR APPROACH?

The final question we asked is: Can our approach add anything to alternative, maybe more specialized representations? One of such representation is based on gradient, instead of darkness, of each pixel, as studied by [Dong , Krzyzak, & Suen, 2003]. We believe the success of our approach comes from the high-level knowledge of strokes. This knowledge should hold whether we use pixel features or gradient features to detect the stroke features. Therefore, our principle should also apply for their gradient representation.

To test this, we followed [Dong , Krzyzak, & Suen, 2003] to build gradient-based representations for our 10-character database. To simplify the re-implementation effort, we omitted the nonlinear image normalization and the learning parameter tuning steps. This, while useful, is tedious and in any case should help both systems equally. We found the recognition accuracy based on gradient representations alone to be 97.2% for our 10-character recognition problem. Not surprisingly, this recognition accuracy is superior to the ones obtained by using the less specialized pixel-based representations. This improvement is solely due to the tailored representation.

For our specialized kernel functions, we used the same set of stroke-level features to design gradient-based feature kernel functions. In particular, the gradient-based representations are calculated for each stroke by only using gradient information in the region for that stroke. This gives us a gradient-based evidence set for each stroke. If we

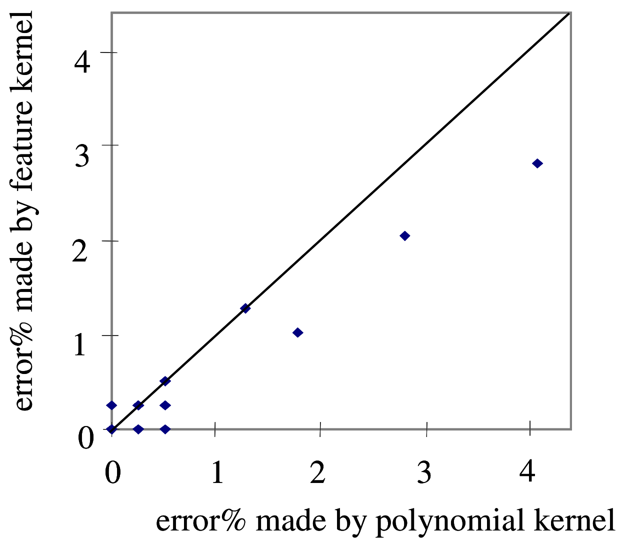

Figure 5.9 A scatter plot comparing the leave-one-out errors using gradient-based representation.

use g_A and g_B to represent gradient-based evidence set for stroke A and B respectively, then, the component kernel function for the corner shown in Figure 5.4 becomes:

$$k_{A,B}(x_1, x_2) = (\sum_{i \in g_A} x_{1i} x_{2i})(\sum_{j \in g_B} x_{1j} x_{2j})$$

With gradient-based component kernel functions, weighing and assembling them to form a gradient-based feature kernel function is the same as pixel-based. We compared our specialized gradient-based feature kernel function with a conventional gradient-based third-degree polynomial kernel for all 45 binary classification problems in our 10-class database. Figure 5.9 shows the comparison in all tasks using leave-one-out test. In most of cases, our feature kernel functions achieved improvements. Using these binary classifiers trained with gradient feature kernel functions yields an accuracy of

98.0% for our 10-character recognition problem, a small but significant improvement over the 97.2% accuracy.

CHAPTER 6

Explanation-Augmented Support Vector Machine

In this approach explanations are directly used to augment training examples as input to the learning algorithm. A simple, yet effective form of explanation is used here to explicitly specify which properties of the example are relevant and how those properties must fit together to achieve the prescribed classification label. To use such explanations as bias for SVM learning algorithm, we require the SVM to classify the training examples according to explanations, i.e. only using those relevant features.

The new learning algorithm for this Explanation-Augmented SVM (EA-SVM) appears to be a convex quadratic programming problem similar to the one for standard SVM. EA-SVM can also be naturally extended to imperfect knowledge, a stumbling block to conventional EBL. It uses cross-validation to evaluate the quality of the explanations, and automatically adjust how much to trust the bias introduced by those explanations.

In this approach, explanations constrain the correct classifier, and therefore, once discovered, can serve to bias the learner. In this sense, explanations themselves directly serve as solution knowledge. Figure 6.1 shows how EA-SVM approach fits to our proposed new EBL learning framework. Although it requires a modified learning algorithm, the advantage is that it requires little for EBL component except producing explanations. Therefore, off-shelf EBL algorithm can be applied. More importantly, the available knowledge-base that traditional EBL uses can now be incorporated into our learning system.

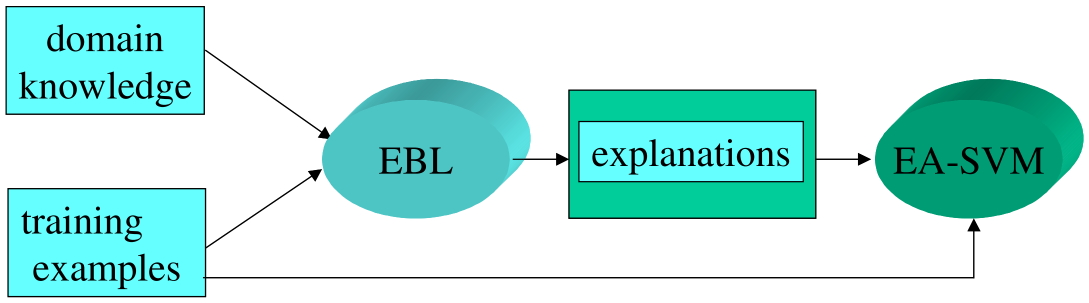

Figure 6.1 The EA-SVM approach to implement the new EBL learning framework.

The theoretical analysis of EA-SVM, which is analogous to (and quite compatible with) the treatment of soft margin SVMs, formally shows that the simple notion of explanation advanced here leads to improved classification performance. The analysis leads to three predictions: 1) explanations will help more in difficult learning problems than in easy ones; 2) improvement will be graduated, with more accurate domain knowledge helping more than less accurate domain knowledge; and 3) even entirely inaccurate domain knowledge should not result in EA-SVM behavior that is unduly worse than the performance of a conventional SVM with training examples alone.

Through a series of experiments, we demonstrate these properties empirically via a simple but challenging task of distinguishing a set of handwritten Chinese characters. To alleviate any concern that the approach is somehow specific to Chinese characters or image classification, we also demonstrate the approach in two additional domains: protein super-family classification (using a database of motifs as the imperfect domain knowledge) and categorizing Reuters news articles (using WordNet as the imperfect domain knowledge). The results confirm that the EA-SVM approach outperforms the conventional SVM approach on the same task.

6.1. Motivation

An explanation can be viewed as generalization of a single example. It is possible that many examples satisfy the same explanation merit the same label for the same reasons and thus should be treated as equivalent by the learner. Now consider an SVM classifier as a linear separator in some high-dimensional feature space [Vapnik, 1998]. What is an explanation to an SVM classifier? In the ideal case, a simple explanation is precisely a single lower dimensional linear surface to which the correct classifier *should be parallel*. To see why, consider the extensional definition of the explanation which is the set of all examples that satisfy the explanation's requirements. Because it is ideal, we assume that the SVM's feature space and linear separator are adequate to capture all of the relevant distinctions and relations. All of the examples that merit this label for the same reasons should be treated identically. In the high dimensional feature space they should have the same margin from the correct classifier. This means that they fall on a parallel linear surface. This surface will be of a lower dimension if there are any redundancies or irrelevancies in the high dimensional feature space with respect to this explanation. A simplified example in three dimensional space is illustrated in figure 6.2, where the explanation specifies that one feature as irrelevant, therefore the constrain surface is a two dimensional plane.

When our explanations are perfect, the bias introduced correctly constrains the hypothesis. In this scenario, EA-SVM can take full advantage of the perfect domain knowledge. But the situation will likely never be ideal. There may be noise in the training examples resulting in spurious explanations. The feature space, despite its high

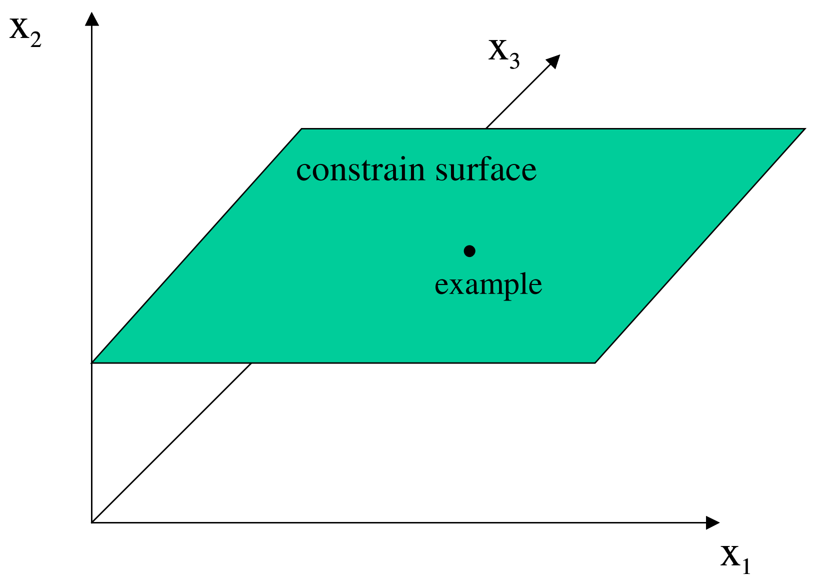

Figure 6.2 A simple example to illustrate the parallel constrain introduced by explanation. The example lives in $\{x_1, x_2, x_3\}$ space. When the explanation indicates that only feature x_1 is important, it suggests that every points on the 2-dimensional constrain surface perpendicular to x_2 should be treated the same by the classifier.

dimension, may not adequately capture all of the relevant distinctions. The domain knowledge itself will almost certainly be flawed so that the explanations will vary in their veracity and in their information content. This, incidentally, is a major limitation of the conventional formalizations of EBL [Mitchell, 1997; Russell & Norvig, 2003]. The present research is not limited to these conventional views. While our explanations also carry evidence about classification labels, they are not forced to carry the overwhelming evidence of a logical proof. Therefore, the explanations produced in our system are likely to be imperfect.

Even when the explanations are not perfect, they often could still be useful. Knowledge often carries certain information about the learning task. Therefore, it is likely that the parallel constrains specified by the explanations could serve as useful preference. Those hyper-planes parallel to the constraint surface could be better than the ones perpendicular to the constraint surface. Also it is possible that some constrains are better than others. This is similar to the situation of noisy training examples. We do not want to ignore such useful information, instead, we should exercise extra effect to take care of the imperfection of knowledge.

Our approach takes care imperfect explanation in a similar way as conventional SVM treats noisy data. A parameter is introduced into the learning algorithm to control how much we should trust the bias introduced by explanations. This parameter is automatically adjusted by cross-validation. In this way, the parallel requirement serves as preference or soft constrains instead of hard constrains.

6.2 Simple Explanations and Explained Examples

Explanations justify the assigned classification label. The simplest explanation is a partitioning of the example's features into important and unimportant ones. While a more sophisticated explanation could also specify how the important features relate, explanations that only partition features have the virtue of simplicity, and as such provide a starting point for our approach. At the same time, more sophisticated explanations hold the promise of even greater gains in the future.

The standard SVM-learned classifier is determined entirely by the kernel matrix [Lanckriet et al 2004], which contains the kernel values between example pairs. We augment the standard kernel matrix to reflect our explanations. To do so, we introduce simple generalized or explained examples. In these, only the important features are allowed to contribute to the kernel computation. Given an original example *x*, and a subset of important features $e \subseteq x$ from an explanation, the generalized example *v* is constructed thus:

$$\begin{cases} v_i = x_i, & if \;\; x_i \in e \\ v_i = '*', & otherwise \end{cases}$$

The special symbol '*' indicates that this feature does not participate in the inner product evaluation. With numerical features one can simply use the value zero.

6.3 Explanation-Augmented SVM Classification

We first discuss the ideal case of perfect explanations and correctly labeled, separable data. Then we show that imperfect explanations can be realized using slack variables. These combine straightforwardly with standard slack variables from the treatment of non-separable data.

6.3.1 PERFECT EXPLANATIONS

Ideally, the learned classifier evaluates the original example and the generalized example to the same value: $w \cdot x_i + b = w \cdot v_i + b$ or equivalently $w \cdot (x_i - v_i) = 0$

Geometrically, this requires the classifier hyper-plane to be parallel to the direction x_i-v_i. We call these *parallel constraints*. The SVM quadratic problem becomes:

$$\begin{aligned} &minimize: \quad \frac{1}{2}\|w\|^2 \\ &subject: \quad to: \quad y_i(w \cdot x_i + b) - 1 \geq 0, \forall i \\ &\qquad\qquad\qquad\quad w \cdot x_i - w \cdot v_i = 0, \forall i \end{aligned}$$

This is a problem similar to the standard SVM optimization problem and can be solved by the method of Langrange multipliers. To do so, we introduce positive Langrangian variables α_i for inequality constrains, and un-constrained Langrangian variables λ_i for equality constrains. The primal Langrangian is:

Minimize: $L_P \equiv \frac{1}{2}\|w\|^2 - \sum_i \alpha_i y_i (w \cdot x_i + b) + \sum_i \alpha_i - \sum_i \lambda_i (w \cdot x_i - w \cdot v_i)$

Subject: $\alpha_i \geq 0$

Setting the derivatives w.r.t. the primal variables to zero, we have:

$w = \sum_i \alpha_i y_i x_i + \sum_i \lambda_i (x_i - v_i)$ and $\sum_i \alpha_i y_i = 0$

Substituting into L_p, the dual problem becomes maximizing (w.r.t. the α_i, λ_i):

$$\begin{aligned} L_D \equiv &\sum_i \alpha_i - \frac{1}{2}\sum_{i.j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + \sum_{i,j} \alpha_i \lambda_i y_i x_i \cdot (x_j - v_j) \\ &- \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j (x_i - v_i) \cdot (x_j - v_j) \end{aligned}$$

subject to: $\alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0$

This is a standard form of the quadratic programming (QP) problem w.r.t the variable vector composed by α_i, and λ_i. The quadratic term is no longer the original *n* x *n* kernel matrix but a *2n* x *2n* matrix in which we employ a single explanation for each training example. After computing this quadratic matrix, we can use a standard QP solver for the above optimization problem.

Notice that, if we fail to build an explanation for a particular example, all input features are treated as important, therefore, $v_i=x_i$, and $v_i-x_i=0$. In such case, the corresponding variable in the QP problem can be simply eliminated. With no explanations the problem reduces to a standard SVM.

6.3.2 Imperfect Explanations

If the domain knowledge is imperfect, the constraints cannot all be met. The explained examples should then be treated as a bias to be respected as much as possible. This is similar to the standard SVM algorithm for the non-separable case [Vapnik, 1998]. New slack variables (δ_i) measure the difference between the evaluations of the original examples and the generalized examples:

$$\forall i \quad w \cdot x_i - w \cdot v_i \geq -\delta_i, \quad w \cdot x_i - w \cdot v_i \leq \delta_i, \quad \delta_i \geq 0_i$$

To penalize violations of the constraints, we change the objective function from $\| w \|^2 / 2$ to $\| w \|^2 / 2 + C \sum_i \delta_i$; we call C the *confidence parameter*. It reflects confidence in (or assessed quality of) the domain knowledge. It will be set automatically. A larger C corresponds to better knowledge and a greater penalty for disagreeing with the explanations. Now our primal problem becomes:

$$\begin{aligned} &Minimize: && \frac{1}{2}\|w\|^2 + C\sum_i \delta_i \\ &sub: && y_i(w \cdot x_i + b) - 1 \geq 0, \forall i; \quad w \cdot x_i - w \cdot v_i \geq -\delta_i, \forall i; \\ &&& w \cdot x_i - w \cdot v_i \leq \delta_i, \forall i; \qquad \delta_i \geq 0_i, \forall i \end{aligned}$$

with the Lagrangian:

$$L_P \equiv \frac{1}{2}\|w\|^2 + C\sum_i \delta_i - \sum_i \alpha_i y_i (w \cdot x_i + b) + \sum_i \alpha_i$$
$$- \sum_i \beta_i (w \cdot x_i - w \cdot v_i + \delta_i) - \sum_i \gamma_i (-w \cdot x_i + w \cdot v_i + \delta_i) - \sum_i \mu_i \delta_i$$

$$sub: \qquad \alpha_i \ge 0, \quad \sum_i \alpha_i y_i = 0, \quad \beta_i \ge 0, \quad \gamma_i \ge 0, \quad \mu_i \ge 0$$

Requiring that the gradient of L_P with respect to *w, b* and δ_i vanish yields:

$$\frac{\partial L_P}{\partial w} = w - \sum_i \alpha_i y_i x_i - \sum_i (\beta_i - \gamma_i)(x_i - v_i) = 0;$$
$$\frac{\partial L_P}{\partial b} = \sum_i \alpha_i y_i = 0; \quad \frac{\partial L_P}{\partial \mu_i} = C - \beta_i - \gamma_i - \mu_i = 0$$

Substituting into the Langrangian formulation L_p, with $\lambda_i \equiv \beta_i - \gamma_i$ we obtain the dual:

$$Maximize: \quad L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i.j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$
$$+ \sum_{i,j} \alpha_i \lambda_i y_i x_i \cdot (x_j - v_j) - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j (x_i - v_i) \cdot (x_j - v_j)$$
$$sub: \qquad \alpha_i \ge 0, \quad \sum_i \alpha_i y_i = 0, \quad -C \le \lambda_i \le C$$

This is a QP problem with the same solution as the perfect explanation case, $w = \sum_i \alpha_i y_i x_i + \sum_i \lambda_i (x_i - v_i)$, except the λ_i are now bounded by *C*. If we have perfect explanations, then *C* and the λ_i are unbounded, and the problem reduces to the ideal case. Conversely, if *C* and the λ_i are *0*, the problem ignores the explanations and reduces to a standard SVM.

This is easily combined with slack variables for non-separable data: slack variables ξ_i are introduced to penalize the errors: $y_i(x_i \cdot w + b) \ge 1 - \xi_i, \forall i; \quad \xi_i \ge 0, \forall i$; the objective function now becomes: $\| w \|^2 / 2 + C\sum_i \delta_i + D\sum_i \xi_i$. The resulting maximization is identical except that the first constraint $\alpha_i \ge 0$ becomes $0 \le \alpha_i \le D$.

Like standard SVM, EA-SVMs can be efficiently solved by the standard optimization methods since the QP problem is convex:

Theorem 6.1: *The EA-SVM QP problem is convex.*

Proof: It is sufficient that the expression L_D of our EA-SVM is quadratic and positive semi-definite:

$$\begin{aligned}
&\frac{1}{2}\sum_{i.j}\alpha_i\alpha_j y_i y_j x_i \cdot x_j - \sum_{i,j}\alpha_i\lambda_i y_i x_i \cdot (x_j - v_j) \\
&\qquad + \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j (x_i - v_i)\cdot(x_j - v_j) \\
&= \frac{1}{2}\left\langle \sum_i(\alpha_i y_i x_i) - \sum_j \lambda_j (x_j - v_j), \sum_i(\alpha_i y_i x_i) - \sum_j \lambda_j (x_j - v_j)\right\rangle \\
&= \frac{1}{2}\left\| \sum_i(\alpha_i y_i x_i) - \sum_j \lambda_j (x_j - v_j)\right\|^2 \\
&\geq 0
\end{aligned}$$

Where $\langle .,. \rangle$ denotes the inner product. □

6.3.3 Setting the Confidence Parameter *C*

In our implementation, the confidence parameter *C* is set by cross validation. Note that *C* bounds the extra variables λ in the solution. When *C* is sufficiently small, λ are much smaller than α and an EA-SVM solution will be similar to standard SVM solution. Conversely, if *C* is sufficiently large, λ is much larger than α and the explanations will dominate. Therefore, we choose the initial candidate set for *C* with respect to value of α. In our comparisons, we first run the standard SVM to determine the average value of α. The candidate set for *C* is a geometric series about this average value.

6.4 A Formal Analysis

We now provide a formal justification for our explanation-augmented SVM using the fat-shattering dimension. We also identify qualitative predictions to be empirically tested which will insure that the formalization addresses significant and observable phenomena.

6.4.1 Background Definition and Theorems

Fat-shattering dimension was introduced by Shawe-Taylor etc. and has been used to provide a theoretical analysis of SVM. Here we review its definition and the theorem about fat-shattering dimension of linear functions.

Definition 6.2 *fat-shattering* (Barlett and Shawe-Taylor, 1999) Let *H* be a set of real valued functions. We say that a set of points $X^m=\{x_1,x_2,\ldots,x_m\}$ is *γ-shattered* by *H* if there are real numbers r_x indexed by $x \in X^m$ such that for all binary vectors *b* index by X^m, there is a function $f_b \in H$ satisfying $f_b(x) \geq r_x + \gamma$ if $b_x=1$ and $f_b(x) < r_x + \gamma$ otherwise.

Theorem 6.3 *Fat-shattering dimension of linear functions* (Barlett and Shawe-Taylor, 1999) Consider a Hilbert space and the class of linear functions *F* of norm less than or equal to B restricted to the examples in the sphere of radium R about the origin, then the fat shattering dimension of *F* can be bounded by $fat_F(\gamma) \leq (\frac{BR}{\gamma})^2$.

The fat-shattering dimension is used in the following theorem that bounds the generalization error of a linear classifier. The fat-shattering dimension of the underlying function class is measured at a scale proportional to the margin.

Theorem 6.4 *Generalization error bound of linear classifiers* (Shawe-Taylor et al., 1998) Consider a set of linear classifiers *H* having fat-shattering dimension bounded above by the function fat: $\Re \mapsto N$. Then with probability at 1-δ, a classifier $h \in H$ that correctly classifies m independently generated examples S with margin $\gamma = \min_i y_i h(x_i) > 0$ will have the error bounded from above by

$$\varepsilon(m,k,\delta) = \frac{2}{m}\left(k\log_2\left(\frac{8em}{k}\right)\log_2(32m) + \log_2\left(\frac{8m}{\delta}\right)\right),$$

where $k = fat(\gamma/8) \le em.$

6.4.2 EA-SVM WITH HARD CONSTRAINTS

The fat-shattering dimension measures the expressiveness of a hypothesis space. The parallel constraints from our explanations should further restrict the expressiveness yielding an easier learning problem. Remember that a class of linear functions *F* of norm less than or equal to B restricted to the examples in the sphere of radius R about the origin. The fat-shattering dimension of *F* is bounded by $fat_F(\gamma) \le (BR/\gamma)^2$. To this we add parallel constraints:

Theorem 6.5 *Fat-shattering of linear functions with parallel constraints*. Consider a Hilbert space and one class of linear functions *F* of norm less than or equal to B satisfying the following constraints: $\forall i,\ f(x_i) = f(v_i)$, where v_i is the explained example of x_i, let R_V denote the radius of the ball that contains all v_i, then the fat shattering dimension of F can be bounded by $fat_F(\gamma) \le (BR_V / \gamma)^2$.

Proof: For every $f \in F, where\ f = w \cdot x + b$, we define a linear function $g \in G: V \mapsto \{0,1\}$ where $g(v) = w \cdot v + b$. The norm of functions $g \in G$ is the same as $f \in F$, but they are restricted to the examples in the sphere of radius R_v. Bartlett and Shawe-Taylor tell us the fat-shattering dimension of G is bounded by $fat_G(\gamma) \le (BR_V / \gamma)^2$.

Next we show that F has the same fat-shattering dimension as G. First observe that $V \subseteq X$, therefore if a set of points $V^m=\{v_1,v_2,\ldots,v_m\}$ is shattered by G, they are also shattered by F. Therefore $fat_F(\gamma) \ge fat_G(\gamma)$. Now consider the explanations as a many-to-one mapping $m: X \mapsto V$, then $f(x) = g(m(x))$. Therefore if a set of points $X^m=\{x_1,x_2,\ldots,x_m\}$is shattered by F, then the set $V^m=\{m(x_1), m(x_2),\ldots, m(x_m)\}$ is shattered by G. Therefore $fat_F(\gamma) \le fat_G(\gamma)$; $fat_F(\gamma) \ge fat_G(\gamma)$ and $fat_F(\gamma) \le fat_G(\gamma)$ imply $fat_F(\gamma) = fat_G(\gamma)$. □

Applying this result to theorem 6.4 immediately yields the following theorem:

Theorem 6.6 *Generalization error bound on linear classifiers with parallel constraints.* Let $S=\{x_1,x_2,\ldots,x_m\}$ be a training set of size m drawn from a fixed but unknown distribution over the input space X. Let v_i be the explained example of x_i, and let R_V denote the radius of the sphere containing all v_i. Then with probability $1-\delta$, the generalization error of a linear classifier $(\boldsymbol{u}, b)$ on X with $\|u\| = 1$ that correctly classifies all examples in S with margin $\gamma>0$, and satisfies the parallel constraints $\forall i,\ f(x_i) = f(v_i)$, is bounded by:

$$\varepsilon(m,h,\delta) = \frac{2}{m}\left(h\log_2\left(\frac{8em}{h}\right)\log_2(32m) + \log_2\left(\frac{8m}{\delta}\right)\right),$$

where $h = \left\lfloor 64.5\mathrm{R_V}^2 / \gamma^2 \right\rfloor \le em$.

This bound is of the same form as the bound for standard SVM [Shawe-Taylor et al., 1998], with R_v playing the role of R. R measures the radius of the example sphere in the original space, while R_v measure the radius of the explained example sphere. Therefore, the explanations have most to offer when the ratio R_v/R is small. This is the case when the learning problem is difficult but the domain knowledge is informative.

This yields our first two testable qualitative predictions: 1) Holding the domain knowledge constant, explanation-augmentation should benefit difficult learning problems more than easier ones. 2) Over the same learning problems, better knowledge should result in better EA-SVM performance.

6.4.3 EA-SVM WITH SOFT CONSTRAINTS

When the constraints cannot all be satisfied, we want a classifier that is as consistent with the constraints as possible. We follow the analysis of [Shawe-Taylor and Cristianini 2002] for soft margins. The input space X is mapped to a higher dimensional space so that the parallel constraints can be satisfied.

Following Shawe-Taylor and Cristianini’s work, we introduce the following definition:

Definition 6.7. Let $L_f(X)$ be the set of real valued functions f on X with countable support *supp(f)* (that is functions in $L_f(X)$ are non-zero for only countably many points) for which the sum of the squared values $\| f \|^2 = \sum_{x \in supp(f)} f(x)^2$ converges. We define the inner product of two functions $f, g \in L_f(X)$, by $\langle f, g \rangle = \sum_{x \in supp(f)} f(x) g(x)$. We use f_0 to denote a special function such that $\langle f_0, g \rangle \equiv 0$.

Now we define a new inner product space $X \times L_f(X)$. For any fixed Δ>0, we embed X into $X \times L_f(X)$ as follows: $\tau_\Delta : x \mapsto (x, \Delta f_0)$, and an embedding of V into $X \times L_f(X)$ as follows: $\tau_\Delta : v \mapsto (v, \Delta \delta_v)$, where $\delta_v \in L_f(X)$ is defined by $\delta_v(z) = 1,$ if $z = v$, and $\delta_v(z) = 0,$, otherwise. It is easy to verify that $\langle \delta_v, \delta_{v'} \rangle = \delta_v(v') = 1$, if $v' = v$, and *0*, otherwise. We are going to show that the above embedding maps the training examples and explained examples into a space where the parallel constraints can be satisfied by a large margin classifier and hence we can apply Theorem 6.4.

Specifically, we augment a linear classifier ***(u**, b)* on X to a linear classifier $(\hat{u}, b)$ on $X \times L_f(X)$, where

$$\hat{u} = \left(u, \frac{1}{\Delta} \sum_i \left(u \cdot (x_i - v_i) \delta_{v_i} \right) \right)$$

With some algebra, we can verify that the augmented classifier $(\hat{u}, b)$ on $X \times L_f(X)$ satisfies the following two properties:

(1). For all $x \in X$, $(\hat{u}, b)$ classifies $\tau_\Delta(x)$ as the same as $(\hat{u}, b)$ classifiers *x*:

$$\begin{aligned}
\hat{u}\cdot\tau_\Delta(x)+b &= \left(u,\frac{1}{\Delta}\sum_i\left(u\cdot(x_i-v_i)\delta_{v_i}\right)\right)\cdot(x,\Delta f_0)+b\\
&=\hat{u}\cdot x+\sum_i\left(u\cdot(x_i-v_i)\delta_{v_i}\right)f_0+b\\
&=\hat{u}\cdot x+0+b\\
&=\hat{u}\cdot x+b
\end{aligned}$$

(2). For all *i*, such that $x_i \in S$, $(\hat{u},b)$ satisfies parallel constraints defined by $\tau_\Delta(x_i)$ and $\tau_\Delta(v_i)$:

$$\begin{aligned}
\hat{u}\cdot\left(\tau_\Delta(x_i)-\tau_\Delta(v_i)\right) &= \left(u,\frac{1}{\Delta}\sum_j\left(u\cdot(x_j-v_j)\delta_{v_j}\right)\right)\cdot(x_i-v_i,-\Delta\delta_{v_i})\\
&=u\cdot(x_i-v_i)+\frac{1}{\Delta}\sum_j\left(u\cdot(x_j-v_j)\delta_{v_j}\right)(-\Delta\delta_{v_i})\\
&=u\cdot(x_i-v_i)+\sum_j\left(u\cdot(x_j-v_j)\right)\left\langle\delta_{v_j},f_0-\delta_{v_i}\right\rangle\\
&=u\cdot(x_i-v_i)+\sum_j\left(u\cdot(x_j-v_j)\right)\left(\left\langle\delta_{v_j},f_0\right\rangle-\left\langle\delta_{v_j},\delta_{v_i}\right\rangle\right)\\
&=u\cdot(x_i-v_i)+\sum_j\left(u\cdot(x_j-v_j)\right)\left(0-\left\langle\delta_{v_j},\delta_{v_i}\right\rangle\right)\\
&=u\cdot(x_i-v_i)-\sum_{j\neq i}\left(u\cdot(x_j-v_j)\right)\left\langle\delta_{v_j},\delta_{v_i}\right\rangle-\left(u\cdot(x_i-v_i)\right)\left\langle\delta_{v_i},\delta_{v_i}\right\rangle\\
&=u\cdot(x_i-v_i)-\sum_{j\neq i}\left(u\cdot(x_j-v_j)\right)\times 0-\left(u\cdot(x_i-v_i)\right)\times 1\\
&=u\cdot(x_i-v_i)-u\cdot(x_i-v_i)\\
&=0
\end{aligned}$$

The first property of the augmented classifier guarantees that its performance on the off-training examples matches exactly the original classifier. Also, for any classifier ***(u**, b)* on X that correctly classifies all the training examples, the corresponding augmented classifier also makes correct classifications on the training set. Moreover the augmented satisfies parallel constraints in $X\times L_f(X)$ space. Therefore, we can apply Theorem 6.5 to get the bound on its fat-shattering dimension, then apply Theorem 6.5 to get the bound on its error rate, which is the same as the error rate of the original classifier ***(u**, b).*

To examine the fat-shattering dimension of the augmented classifier $(\hat{u}, b)$, we first observe that it has the same margin as the original classifier, since $\hat{u} \cdot \tau_\Delta(x) + b = u \cdot x + b$. The cost of the additional component in $\hat{u}$ is in its effect of increasing the square of the norm of the classifier by $\frac{1}{\Delta^2} D^2$,

where $D \equiv \sqrt{\sum_i \left(u \cdot (x_i - v_i)\right)^2}$:

$$
\begin{aligned}
\|\hat{u}\|^2 &= \|u\|^2 + \frac{1}{\Delta^2} \sum_i \left(u \cdot (x_i - v_i)\delta_{v_i}\right) \cdot \sum_i \left(u \cdot (x_i - v_i)\delta_{v_i}\right) \\
&= \|u\|^2 + \frac{1}{\Delta^2} \sum_i \sum_{j \neq i} \left(u \cdot (x_i - v_i)\right)\left(u \cdot (x_j - v_j)\right)\left\langle \delta_{v_i}, \delta_{v_j} \right\rangle \\
&\quad + \frac{1}{\Delta^2} \sum_i \left(u \cdot (x_i - v_i)\right)\left(u \cdot (x_i - v_i)\right)\left\langle \delta_{v_i}, \delta_{v_i} \right\rangle \\
&= \|u\|^2 + \frac{1}{\Delta^2} \sum_i \left(0 + \left(u \cdot (x_i - v_i)\right)^2 \times 1\right) \\
&= \|u\|^2 + \frac{1}{\Delta^2} \sum_i \left(u \cdot (x_i - v_i)\right)^2
\end{aligned}
$$

Also the explained examples are embedded into $X \times L_f(X)$ by the mapping $\tau_\Delta : v \mapsto (v, \Delta\delta_v)$, which increases the square of the radius of explained examples by Δ^2:

$$\|\tau_\Delta(v)\|^2 = \|v\|^2 + \Delta^2 \langle \delta_v, \delta_v \rangle = \|v\|^2 + \Delta^2$$

Taking these adjustments into account, Theorems 2 and 3 yield the following result:

Theorem 6.8 *Generalization error bound of linear classifiers with soft constraints*. Fix $\Delta > 0, b \in R$. Randomly draw training set *S={x₁,x₂,…,xₘ}* of size *m* with a fixed but unknown probability distribution on the input space *X*. Let v_i be the explained example of x_i, and R_V denote the radius of the sphere containing all v_i. Then with probability *1-δ*, the generalization error of a linear classifier ***(u**, b)* on X with $\|u\| = 1$ that correctly classifies all examples in *S* with margin $\gamma > 0$ is bounded by:

$$\varepsilon(m, h, \delta) = \frac{2}{m}\left(h \log_2\left(\frac{8em}{h}\right) \log_2(32m) + \log_2\left(\frac{8m}{\delta}\right)\right),$$

$$\text{where } h = \left\lfloor \frac{64.5(\mathrm{R_V}^2 + \Delta^2)(1 + D^2 / \Delta^2)}{\gamma^2} \right\rfloor \le em \text{, and } D \equiv \sqrt{\sum_i (u \cdot (x_i - v_i))^2} \, .$$

6.4.4 Finding Linear Classifier with Soft Constraints in the Expanded Space

The preceding analysis provides a way to transform an optimization problem with non-satisfiable constraints into one with satisfiable constraints. Next we show that the algorithm presented in section 6.3 is closely related to the one that solves this new optimization problem.

The mapping τ_Δ used in the previous analysis implicitly defines a kernel as follows:

$$k(x, x') = \langle \tau_\Delta(x), \tau_\Delta(x') \rangle = \langle (x, \Delta f_0), (x', \Delta f_0) \rangle = x \cdot x'$$
$$k(v, v') = \langle \tau_\Delta(v), \tau_\Delta(v') \rangle == v \cdot v' + \Delta^2 \delta_v(v')$$

By using these kernels to replace $x \cdot x'$ and $v \cdot v'$ in the dual QP problem with hard constraints in the expanded space $X \times L_f(X)$, we have:

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(x,x') + \sum_{i,j}\alpha_i\lambda_i y_i k\big(x_i,(x_j - v_j)\big)$$
$$-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j k\big((x_i - v_i),(x_j - v_j)\big)$$
$$= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x\cdot x' + \sum_{i,j}\alpha_i\lambda_i y_i x_i\cdot(x_j - v_j) - \frac{1}{2}\sum_{i\neq j}\lambda_i\lambda_j k\big((x_i - v_i),(x_j - v_j)\big)$$
$$-\frac{1}{2}\sum_{i}\lambda_i\lambda_j k\big((x_i - v_i),(x_j - v_j)\big)$$
$$= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x\cdot x' + \sum_{i,j}\alpha_i\lambda_i y_i x_i\cdot(x_j - v_j) - \frac{1}{2}\sum_{i\neq j}\lambda_i\lambda_j (x_i - v_i)\cdot(x_j - v_j)$$
$$-\frac{1}{2}\sum_{i}\lambda_i\lambda_i k\big((x_i - v_i),(x_i - v_i)\big)$$
$$= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x\cdot x' + \sum_{i,j}\alpha_i\lambda_i y_i x_i\cdot(x_j - v_j) - \frac{1}{2}\sum_{i\neq j}\lambda_i\lambda_j (x_i - v_i)\cdot(x_j - v_j)$$
$$-\frac{1}{2}\sum_{i}\lambda_i\lambda_i \big[(x_i - v_i),(x_i - v_i) + \Delta^2\big]$$
$$= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x\cdot x' + \sum_{i,j}\alpha_i\lambda_i y_i x_i\cdot(x_j - v_j)$$
$$-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j (x_i - v_i)\cdot(x_j - v_j) - \frac{1}{2}\Delta^2\sum_i \lambda_i{}^2$$

Now we show that the above is exactly the dual QP problem that one would obtain by solving the following optimization problem with $C = 1/(2\Delta^2)$:

$$\begin{aligned} &\mathit{Minimize}: && \frac{1}{2}\|w\|^2 + C\textstyle\sum_i \delta_i{}^2 \\ &\mathit{sub}: && y_i(w\cdot x_i + b) - 1 \geq 0, \forall i; \quad w\cdot x_i - w\cdot v_i = \delta_i \end{aligned}$$

Applying the Lagrangian to this problem, we get:

$$L_P \equiv \frac{1}{2}\|w\|^2 + C\sum_i \delta_i{}^2 - \sum_i \alpha_i y_i(w\cdot x_i + b) + \sum_i \alpha_i$$
$$-\sum_i \lambda_i(w\cdot x_i - w\cdot v_i - \delta_i)$$

Requiring that the gradient of L_P with respect to *w, b* and δ_i vanish yields:

$$\frac{\partial L_P}{\partial w} = w - \sum_i \alpha_i y_i x_i - \sum_i \lambda_i (x_i - v_i) = 0; \quad \frac{\partial L_P}{\partial b} = \sum_i \alpha_i y_i = 0;$$
$$\frac{\partial L_P}{\partial \delta_i} = 2C\delta_i + \lambda_i = 0$$

Substituting them into the Langrangian formulation L_p, we obtain the dual problem:

$$\begin{aligned} &Maximize: \quad L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ &+ \sum_{i,j} \alpha_i \lambda_i y_i x_i \cdot (x_j - v_j) - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j (x_i - v_i)\cdot(x_j - v_j) - \frac{\sum_i \lambda_i^2}{4C} \end{aligned}$$

Set $C = \frac{1}{2\Delta^2}$, this gives the dual QP problem for the optimization problem with hard constrains in the expanded space $X \times L_f(X)$.

The algorithm we described in the previous section actually solves a closely related optimization problem, where we use 1-norm instead of 2-norm in the extra penalty term: where we use 1-norm instead of 2-norm in the extra penalty term:

$$\begin{aligned} &Minimize: \quad \frac{1}{2}\|w\|^2 + C\sum_i |\delta_i| \\ &sub: \quad y_i(w \cdot x_i + b) - 1 \geq 0, \forall i; \quad w \cdot x_i - w \cdot v_i = \delta_i \end{aligned}$$

6.4.5 MORE ON THE CONFIDENCE PARAMETER C

The bound in the Theorem 6.8 depends on a parameter Δ that we use to define the mapping. For different Δ, the bound stated in Theorem 6.8 holds, but the tightness of the bound varies. Note that the minimum of the expression for *h* (ignoring the constant and suppressing the denominator γ^2) is $(R + D)^2$ attained when $\Delta = \sqrt{RD}$. Therefore, we need to not only search for the hyper-plane, but also adjust Δ to minimize the error bound.

One way to adjust Δ is to choose a discrete set of values for it, and evaluate the best hyper-plane we could find given each value. This is exactly the cross-validation algorithm we used in our algorithm. To verify this, notice, as shown in section 6.4.3, that the parameter C used in our algorithm and Δ from the analysis are related: $C = 1/(2\Delta^2)$. Changing C in our algorithm is just mapping examples to the expanded space using a different Δ.

Since $C = 1/(2\Delta^2)$, if the optimal C (found by cross validation) is large, the optimal Δ minimizing h in Theorem 6.8 is small. Since this optimal value is obtained when $\Delta = \sqrt{RD}$, it suggests that the linear classifier has a small D with respect to explanations. According to Theorem 6.8, such a classifier is likely to have a low error rate. Thus, C is a measure of the domain knowledge quality and, from cross validation, it is unlikely that poor knowledge will be confused with good knowledge. This is our third qualitative prediction.

6.5 Empirical Results

We devised four empirical investigations to validate the EA-SVM method and to lend credibility to the formal analysis by testing the predicted qualitative behaviors.

The first three experiments employ the domain of distinguishing handwritten Chinese characters. Experiment 1 demonstrates the relative advantage of the EA-SVM over an otherwise-identical conventional SVM. Experiment 2 tests the first prediction, demonstrating that explanation-augmentation helps more as problems become more difficult. Experiment 3 confirms the two other predictions: that EA-SVM performance improves with the quality of the domain knowledge, and that even extremely misleading domain knowledge does not harm EA-SVM performance significantly measured against an identical conventional SVM. Experiment 4 demonstrates that the EA-SVM method is broadly applicable. EA-SVM improvement is shown on protein super-family classification and categorization of Reuters news articles.

We implemented the SVM and EA-SVM with Matlab using PR_LOQO (http://www.kernel-machines.org/code/prloqo.tar.gz) as the QP solver. The confidence parameter C in the algorithm is set automatically by 5-fold cross-validation. The candidate set for C, as explained in section 6.4.3, is chosen according to the average SVM α. The standard SVM slack variable misclassification penalty, D, is set to *0.1* for all experiments.

Our primary domain concerns classifying pairs of handwritten Chinese characters. Similar to the experimental setups, we chose 10 characters from 3 related groups to yields 45 classification problems of varying difficulties. Both learners use a conventional third degree polynomial kernel function of pixel intensities. The performance of each classifier is evaluated using 5-fold cross validation.

To build explanations, we specify, for each pair characters, which stroke-level features are important to distinguish them. We follow the same procedure as explained in Chapter 4 to use Hough transformation to explain training examples in terms of strokes. Our explained examples specify which pixels are important.

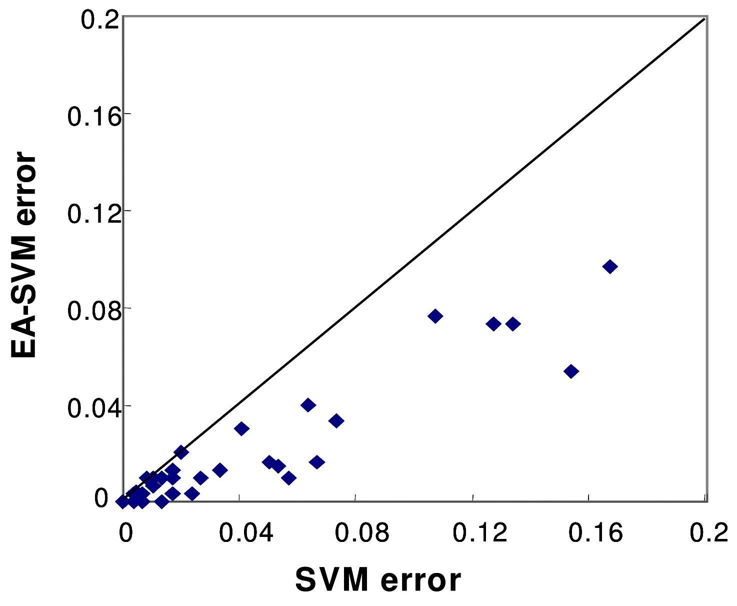

Figure 6.3 Comparison of SVM and EA-SVM performance on handwritten test recognition tasks.

Experiment 1: Does Explanation-Augmentation Help?

Figure 6.3 compares the average error rates in all forty-five tasks over a randomly selected training set of 320 examples; the remaining examples are used as the test set. The results are shown as a scatter plot where the vertical axis is the EA-SVM so that points falling below the 45 degree line correspond to learning problems for which EA-SVM outperforms the standard SVM.

Experiment 2: Do Difficult Problems Benefit More?

In Figure 6.3, explanations seem to be more helpful in difficult classification tasks. To further investigate this, we divided the 45 classification tasks into two difficulty conditions. In the easy condition, characters to be distinguished are drawn from different groups (33 tasks). In the difficult condition, characters from the same group must be

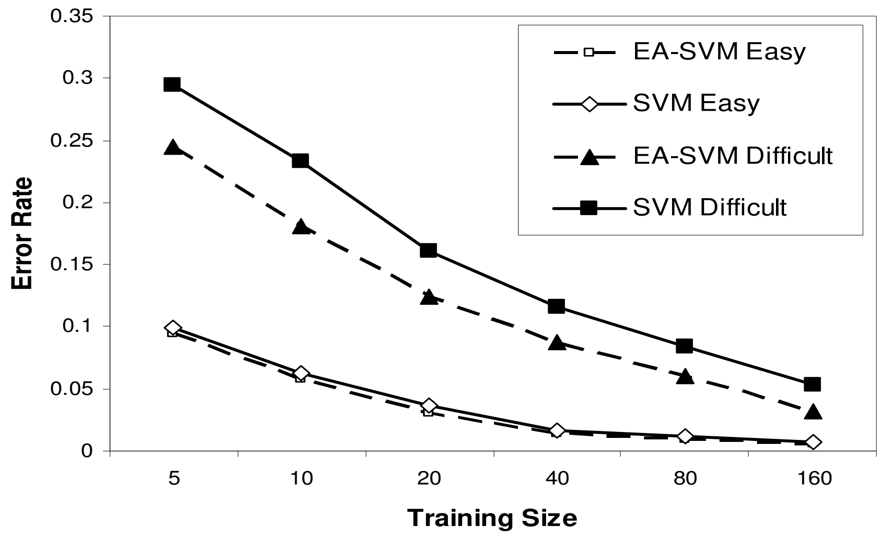

Figure 6.4 EA-SVM / SVM performance on easy and difficult tasks.

distinguished (12 tasks). Figure 6.4 shows the average learning curves of these two conditions. The improvement on difficult problems is greater than easy ones at all training levels. EA-SVM and SVM performance on the easy tasks is almost indistinguishable.

Next we analyzed the classification results using the SVM error rate on each task as a measure of task difficulty. Kendall's Tau [Conover, 1980] is a nonparametric test for the agreement between two rankings. It provides the confidence level at which we can believe our hypothesis that problem difficulty is highly correlated with the improvement afforded by explanation-augmentation over the standard SVM. Simple measures for the difficulty of the problem and the improvement are first applied, where the error rate of the standard SVM is used to measure the difficulty, while the difference of error made by standard SVM minus error made by EA-SVM is used to measure the improvement. Figure 6.5 plots the improvement against task difficulties. Clearly, EA-SVM provides

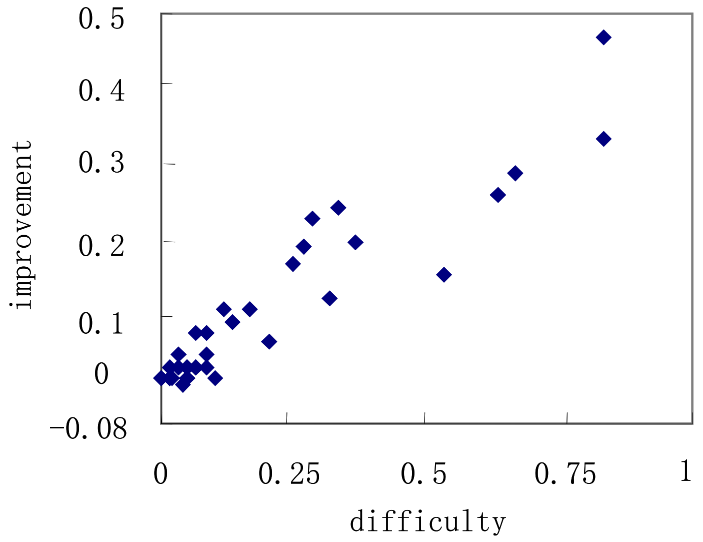

Figure 6.5 Scatter plots of EA-SVM improvement vs. task difficulty. The error rate of the standard SVM is used to measure the difficulty, while the difference between error rates of EA-SVM and SVM is used to measure improvement. The difficulty is scaled to 0-1, and the improvement is scaled accordingly.

more improvement for difficult problem. The Kendall's Tau over these two variables is 0.798, which suggests high correlation. The statistic test shows the hypothesis is accepted with more than 99.99% confidence. To further test our hypothesis, we next use more complicated measures of difficulty, such as quadratic mean of the error made by EA-SVM and the error made by SVM, and more complicated measures of improvement, such as the ratio of the errors, the difference of the errors normalized by error made by SVM. All statistic tests show that the hypothesis is accepted with more than 99.93%. Therefore, we conclude that EA-SVM helps most in the difficult classification tasks.

Experiment 3: The Effect of Knowledge Quality

Good but imperfect knowledge improves EA-SVM behavior over conventional SVMs. But does behavior degrade gracefully with poorer knowledge? To answer this

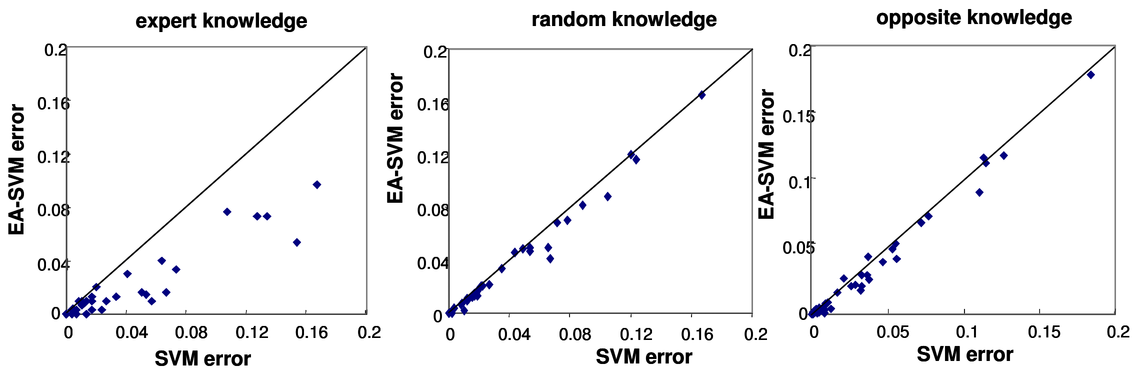

Figure 6.6 Effect of knowledge quality on performance of EA-SVM.

question, we built two additional kinds of explanations: 1) in the random condition, random sets of pixels set to 0 with no regard to the original domain knowledge; 2) in the opposite condition, the complement of the original explanation is used so that important pixels are now unimportant and vice versa. Figure 6.6 shows the scatter plots of errors made by standard SVM and EA-SVM in these conditions. Random explanations almost never harm the performance, with some marginal improvement in certain tasks. Opposite explanations offer almost no improvement, but importantly, they do not significantly harm performance.

Experiment 4: Performance on Protein Super-Family and Text Categorization.

How general is our approach? Is there some fortuitous match between the workings of EA-SVMs and the task of distinguishing handwritten characters? To address this question, we examined two additional learning domains: protein super-family prediction [Karplus etc.], and topic categorization of text articles [Lewis 1997]. In each domain we adopt the most standard kernel from the literature in order to exercise our approach under different kernel functions. We also adopt the accepted scoring criteria for each domain. Domain knowledge for these two domains is taken directly from available databases - the Prosite motif database [Bucher & Bairoch 1994] and WordNet (http://wordnet.princeton.edu/).

Domain 1: Proteins are assigned to functional and structural classes called *super-families* based on their amino acid sequences. This is a typical biological sequence analysis problem, where we would like to predict protein or DNA functionality based on their sequences. The domain knowledge is a database of motifs, which are conserved sequences that have been experimentally determined to be important for a protein's functionality.

We use the Structural Classification of Proteins (SCOP, http://scop.mrc-lmb.cam.ac.uk/scop/), a database of known 3D structures of proteins, as the data set. It contains 54 super-families and 7329 example protein sequences. This data set has been used in several previous studies. We adopt the same test and training set splits as [Leslie, Eskin, & Noble. 2002].

The (rather flawed) domain knowledge here is that a protein's super-family is determined only by motif sequences, and not by other amino acid sequences. To build an explanation for a training example, we use its sequence to search for known motifs in the Prosite motif database. The search is performed using the tools offered by Biology WorkBench (http://workbench.sdsc.edu/).

In our experiments, we use the special string kernel, called mismatch kernel, for the SVM algorithms. The mismatch kernel, designed for biological sequence analysis, has shown to achieve state-of-the-art classification performance [Leslie, Eskin, & Noble.

2002]. The *(k,m)*-mismatch kernel maps a finite length string to a vector space indexed by all possible sub-strings of some fixed length k; each instance of a fixed k-length subsequence in an input sequence contributes to all feature differing from it by at most m mismatches. Following [Leslie, Eskin, & Noble. 2002], we set k=5 and m=1. The performance of the classifiers is evaluated using the Receiver Operating Characteristic (ROC) score.

Not all proteins contain motifs described in the Prosite database. Therefore, some examples cannot be explained. Also, the labeling in this dataset uses a one-against-all scheme, where in each training set, members of one super-family are labeled as positive examples, while all others are used as negative example. From an EBL point of view, a protein is a negative example because it shares little homology with positive examples. Therefore, motifs in the negative examples do not provide acceptable explanations for why they merit negative labels. For this reason, we only use the explanations for positive examples in our EA-SVM algorithm. Table 6.1 summarizes the average number of positive, negative and generalized examples, and the average ROC score for standard SVM and for EA-SVM.

Table 6.1. Summary of protein super-family classification results.

Ave. Explanations	Ave. Pos	Ave. Neg	ROC_SVM	ROC_EA-SVM
9.4	25.9	1918.5	0.874	0.900

Figure 6.7A shows the scatter plot of ROC scores comparing SVM and EA-SVM performance in all 54 classification problems of the SCOP database. The higher the score, the better the performance. The points above the equal-line indicate that EA-SVM

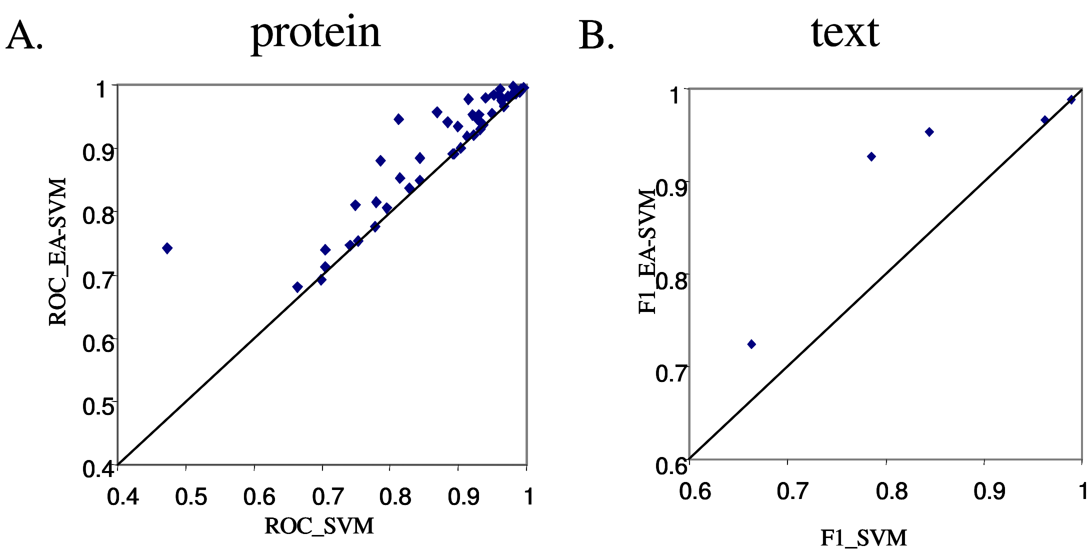

Figure 6.7 EA-SVM improves classification performance in the protein and text categorization domains.

out-performs SVM in almost all cases. Among those scatters that lie on the diagonal line, four of them have no explanations for all positive training examples. In the other cases, the confidence parameter is set by cross- validation to an extremely small value, therefore explanations contribute little, if any, bias to the learned SVM. In other cases, the degree of improvements indicates of the utility of the explanations.

Domain 2: The Reuters-21578 data set assigns category labels to Reuters news articles. To obtain a training and test set, we use the Modified Apte ("ModApte") split, which leads to a corpus of 9603 training documents and 3299 test documents. A Reuters category can contain as few as 1 or as many as 2877 documents in the training set. Similarly, a test set category can have 1 to 1066 documents.

Our domain knowledge is that the label of a category is meaningful to the topic. Thus, the presence of this and semantically related words are likely to be more informative than others about an article's category. The knowledge about semantic closeness of words is provided by WordNet. Words that are one-distance away from synonyms of topic words according to the WordNet architecture are taken as important. This set is used to select the words in the article that constitute our explanation. A generalized example is the bag of those important words occurring in the training example.

Our experimental setup follows [Cristianini, Shawe_Taylor, & Lodhi, 2001]. Each training set contains randomly selected 2000 documents. The test set is composed of 1000 documents randomly selected from the remainder. Learning is performed on the top 5 Reuters categories whose data set labels are "earn", "acquire", "money", "grain", and "crude". The kernel function is defined as a linear function over bag-of-words in the input articles. The words are weighted according to the *tfidf* scheme. The weights are set by $log(1+tf)*log(m/df)$, where tf represents term frequency, df is used for the document frequency and m is the total number of documents. For evaluation of classifiers, we used the $F1$ performance measure. It is given by $2pr/(p+r)$, where p is the precision and r is the recall.

The results are shown in Figure 6.7B. Again, a higher score indicates better performance; the points above the equal-line indicate that EA-SVM consistently out-performs the SVM control. EA-SVM is sometimes significantly better and never worse than SVM. The two most improved categories are "crude" and "grain", where explanations from WordNet are distinctive. Our simple explanations do not help much to distinguish the other three categories (“earn”, “acquire”, “money”) which all concern financial news. This suggests that our simple explanation may select those words common to finance for all three categories. In this sense, our explanations are far from perfect. A more sophisticated explanation component would probably perform better.

CHAPTER 7

Comparison of Three Approaches

We introduced three different approaches to implement the proposed new EBL learning framework. In this section, we compare the three approaches for their performance in terms of classification accuracy, computation and memory requirement. We also discuss the advantages and disadvantages of the three approaches in other aspects, such as their robustness, their running time, and their applicability to the classification algorithms other than SVMs.

First we examine the classification performance of the three approaches. This is our most interesting aspect of the algorithms, since our motivation is to use knowledge to improve classification performance of the learning algorithms. We compare the three approaches for the problem of distinguishing two Chinese characters shown in figure 2.1. In our experiments, 320 out of 400 images were chosen as training examples, and the remaining 80 images were used as test examples. For phantom example approach, we used three augmenting example sets with 100, 500, and 2000 phantom examples, respectively. The results were averaged over 5 runs. The error bars show 95% confidence range. We see that all the approaches can utilize knowledge to achieve better classification performance, while feature-kernel approach and EA-SVM approach are slightly better than phantom-example approach. As expected, more phantom examples used, better the performance is. Phantom example approach augmented with 2000 phantom examples achieves almost similar improvement as feature-kernel approach and EA-SVM approach.

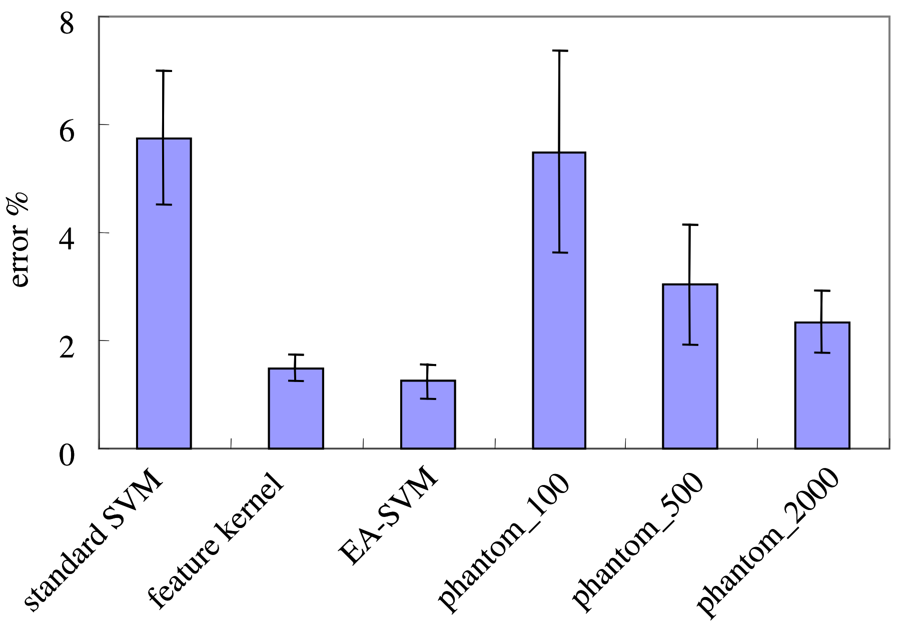

Figure 7.1 Comparison of three explanation-based approaches on a simple Chinese character classification problem.

We next examine the learning curve of different approaches. In figure 7.2, we plot the classification error against the number of training examples used in the learning. It is interesting to notice that the phantom example approach is significantly better than feature-kernel approach and EA-SVM approach when the real training set is small. When only 10 real examples are used, phantom example approach, even with only 100 augmented examples, is significantly better than the other two approaches. It is interesting to notice that the phantom example approach is significantly better than feature-kernel approach and EA-SVM approach when the real training set is small, while it is not as good as the other two approaches when the training examples are plenty. This result perfectly demonstrates the flexibility of different explanation-based approaches. They all integrate information from both data and knowledge to dynamically generate

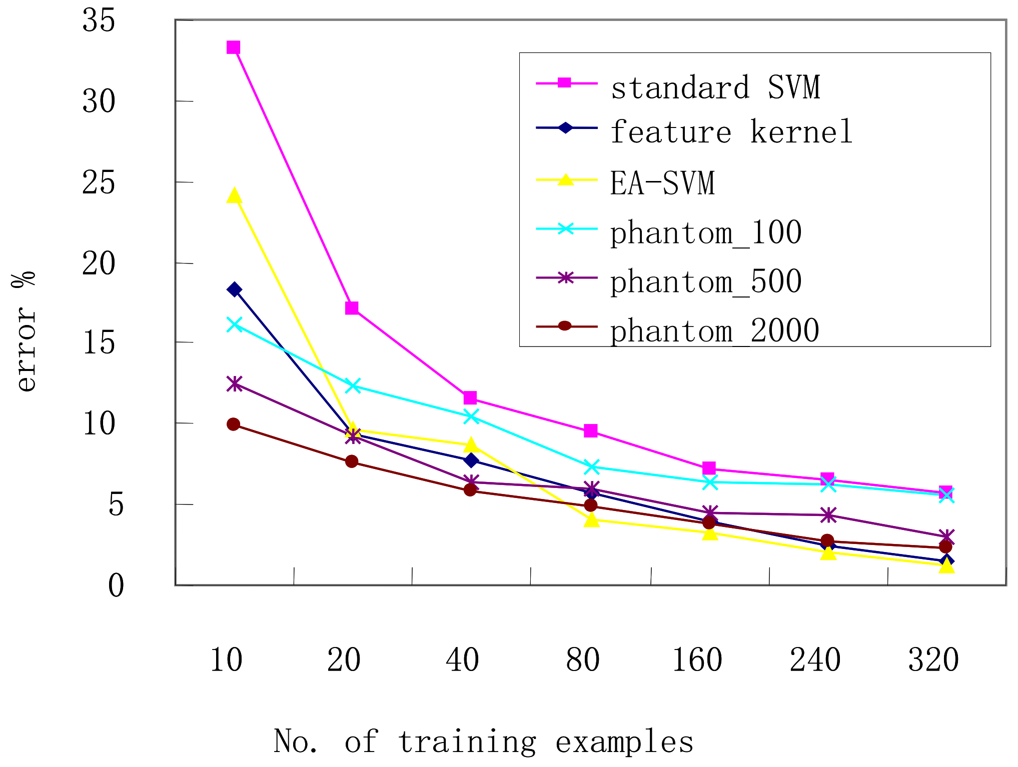

Figure 7.2 Comparison of the learning curves of difference explanation-based approaches with various numbers of training examples.

solution knowledge. But different approaches may put more weight on one information source against another: to build generative models for handwritten characters, the phantom example approach explicitly uses knowledge about the stroke-prototype of each characters and knowledge about how the strokes can be realized by pixels, while feature-kernel approach relies more on the training examples to weight component kernels, and EA-SVM uses training examples to cross-validation to determine the confidence parameter. Therefore, phantom example approach replies more on the domain knowledge to build solution knowledge, while the other approaches have more efficient ways to incorporate information from data into solution knowledge. With such implementation flexibility, different approach can be designed to fit different applications.

Table 7.1 Computation and memory requirement of different explanation-based approaches.

	CONVENTIONAL SVM	PHANTOM EXAMPLES	FEATURE KERNEL	EA-SVM
RUNNING TIME	$O(n^2)$	$O((m+n)^2)$	$O(l^2+n^2)$	$O(4n^2)$
MEMORY	$O(n^2)$	$O((m+n)^2)$	$O(min(l^2,n^2))$	$O(4n^2)$

Notes: n is the number of training examples. m is the number of phantom examples used in phantom example approach. l is the number of component kernels used in feature-kernel approach.

Next we examine the computation and memory requirement of these three approaches. As listed in table 7.1, all three approaches require extra computation time and memory. For phantom example approach, the extra requirement is mainly due to the training set augmentation. As SVM learning algorithm runs in $O(n^2)$, extra phantom examples can significantly increase the training time and memory requirement. For feature-kernel approach, the training set is the same as conventional SVM, it introduces extra computation step to weight component kernels. Currently, the weighting is done by solving a quadratic programming problem, which runs in $O(l^2)$, where l is the number of the component kernels. For EA-SVM approach, we augment the training set with explained examples, which increases the size of quadratic programming from n x n to $2n$ x $2n$. Therefore the running time and memory requirement are quadrupled accordingly. Overall, phantom example approach adds most extra requirement, especially when the training set is small, and the number of phantom examples is much bigger. The feature-kernel approach adds relatively less extra requirement, especially when the number of the high-level features is limited.

We also want to point out that the phantom example approach has the advantage that it completely insulates domain knowledge form the statistical learner. By using phantom examples to communicate EBL generated solution knowledge, any statistically learning algorithm can benefit from it.

The approach of building feature kernel function and the approach of explanation-augmented SVM, on the other hand, relieves the burden from EBL component. They omit the conventional EBL concept generation step. Explanations are used to build specialized kernel function, or are directly used to impose extra bias to SVM learning algorithm. The simplification of EBL procedure makes it possible to apply these approaches to more classification problems.

The EA-SVM approach has an extra advantage that it introduces a confidence parameter in the learning algorithm to evaluate the quality of knowledge. It is therefore more robust to the imperfection in the knowledge.

CHAPTER 8

Related Work

A lot of work has been done to incorporate prior knowledge into SVMs. They have provided a strong background and motivation for our work. Some of them are summarized here into three categories according to their similarity and relevance to our proposed three approaches. By comparing them with our approaches, we emphasize the main contribution of our work, that is to introduce high-level knowledge into a SVM that operates on low-level input attributes.

Other works related to this study concerns how to use stroke-level knowledge for handwritten character recognition. They use the knowledge to define alternative representation of characters. Such static manner of using knowledge lacks the robustness we introduced in our study.

In general, these related works use knowledge in a heuristic fashion. Each approach is largely influenced by the property in the knowledge. We hope that, by introducing the concept of domain knowledge, proposing new EBL learning framework, and providing theoretical analysis, our work can build a ground for future work on how to systematically use knowledge in machine learning.

8.1 Virtual SVM and Invariance Knowledge

One kind of useful knowledge for handwritten digit/character recognition is the knowledge about transformation invariance. It is reasonably believe that when the character images are minor geometrically transformed, either by translation or rotation, the results images should still be the images of characters with the same label. Couple approaches have been proposed to take advantage of such knowledge for support vector machines.

The work that mostly related to our phantom example approach is Virtual support vector method [Schölkopf, Burges & Vapnik, 1996]. In this approach the training examples are transformed to produce artificial labeled examples, called virtual examples, to augment training set. This approach is similar to our phantom example approach in that the knowledge is not directly used by the learning algorithm, but is used to interact with training examples to produce artificial examples.

The main difference between virtual SVM and our phantom example approach is that the invariance knowledge is still expressed in the low-level vocabulary. The transformation used to produce virtual examples operates on pixels. Such knowledge can be used to directly state the desirable property that the hypothesis should have. In this case, the desirable property is that the classifier itself should be invariant to transformation. Such knowledge can be directly used to introduce bias intro learning algorithms. Two approaches use the transformation invariance in this manner. Jittered SV method [DeCoste & Schölkopf, 2002] applies transformation to kernel definition. When a kernel value is computed, the support vector is moved around by translation or rotation to find the best match with the comparing example. The other approach uses transformation invariance as a desirable property of a good kernel function, and constructs a transformation invariance kernel [Schölkopf, & Smola, 2002].

The above discussion indicates that, although the virtual SVM and our phantom example approach are similar, they use different knowledge with different principle. It suggests a straightforward manner to utilize these two seemly compensatory sources of knowledge.

8.2 Learning Kernel Functions

Recently, with the theoretical understanding of the importance of kernel functions, there has been much work on learning kernel functions from training examples. A weighted linear combination of kernels can be learned by using alignment maximization [Lanckriet et al., 2004] in the transductive learning problems, or by using the hyperkernels [Ong & Smola, 2003] in the induction setting. A related approach is to obtain the kernel combination by boosting [Crammer, Keshet & Singer, 2003]. Also, instead of combining kernels, learning a kernel function that weights input features has also been proposed in [Kwok, & Tsang, 2003]. Our approach also adapts a kernel function for the given problem, but we utilize information from both domain knowledge and examples, therefore avoid the complexity of learning kernels solely from examples [Bousquet & Herrmann 2003).

Our feature kernel approach also includes weighting component kernel functions. But, as our analysis suggests, decomposing a generic kernel function into component kernel functions is equally important as weighting them. Without utilizing knowledge, it is hard, if not impossible to obtain a good decomposition. Our approach illustrates the key idea of EBL to allow interaction between knowledge and examples introduces a better way to utilize information from both knowledge and examples.

8.3 Knowledge-Based SVM

Fung et al. [2003] demonstrated how prior knowledge in the form of polyhedral knowledge sets could be incorporated into SVMs. They called their approach *knowledge-based SVM*. The knowledge they used expresses expert's opinion of how to classify certain sub-region in the example space. Similar to our EA-SVM approach, knowledge-based SVM modifies SVM learning algorithm to utilize the expert knowledge. The difference is that their knowledge is static, and is in the same vocabulary as learner's bias. Clearly such knowledge is solution knowledge.

Conceptually, EA-SVM can be viewed as an extension of knowledge-based SVM. In EA-SVM approach, domain knowledge is used dynamically by EBL to generate solution knowledge. The EA-SVM learning algorithm is also different from the one used in knowledge-based SVM in that it is able to take advantage of the correlation between original examples and explanations. Most importantly, the insulation of domain knowledge from learning algorithm in our EA-SVM approach has the advantage to allow domain experts to specify their knowledge in natural vocabulary.

8.4 Other Work on Using Knowledge of Strokes

The straightforward idea of using knowledge of strokes is to directly extract stroke representation from pixel images. In such way, examples are in a metric space that could describe the intrinsic difference between characters. An approach proposed by [Teow & Loe, 2000] defines a set of convolution kernels to extract stroke-level features, such as the end of stroke, the intersection of strokes and so on, and builds a classifier using those features. They tested their approach in handwritten digit recognition domain. Although the approach is well-motivated, the performance of their learned classifier failed to match that of competing classifiers, such as classifying pixel images with SVMs. The reason is that human-defined convolution kernels almost certainly won't be able to anticipate all the variations in the data set. Therefore, useful information could be lost during the mapping to the high-level feature representation

A similar approach directly uses nonlinear active shape model to extract radicals from handwritten Chinese characters [Shi, Gunn & Damper, 2003]. Such human written programs, although involve certain vision techniques that are more advanced than Hough transformation, cannot work perfectly. In fact they reported a 96.5% error on extracting radicals. When the radical representation is used to describe examples to learner, such errors will be propagated and could potentially harm the learning algorithm.

The lack of interaction between knowledge and examples make the above methods less robust than our EBL approach. On the other hand, our approach could benefit from these works from their expertise on how to use advanced vision techniques to utilize stroke-level knowledge.

8.5 Feature Generation in Relational Learning

It is interesting to notice that our approach shares common philosophy with work on feature extraction for relational learning by Cumby and Roth (Cumby and Roth, 2000, 2002, 2003) Their work concerns complicated large-scale relational learning problems in which the potential number of features is very large. Their techniques are aimed at building features in a data-driven way, based on the prior knowledge about the “type” of feature in the problem domain. Despite the difference on the learning problems addressed, their work shares the same motivation as ours, namely, utilizing knowledge in a data-driven manner to provide informative features for the learner. Moreover, instantiating the “type” of feature to lower-level features in their approach is strikingly similar to our idea of transforming high-level features to low-level features in the feature kernel approach.

Yet we believe that by putting our feature kernel approach into explanation-based learning framework, we emphasize the importance of the interaction between knowledge and data. Our proposed explanation-based learning framework abstracts the key idea behind our approaches. Such abstraction makes it clear that interaction between knowledge and data can extract information from otherwise-difficult-to-use domain knowledge, and the ability to use such information is what gives us the advantage over conventional learning techniques.

Our work can significantly benefit from Cumby and Roth’s formal framework proposed in their work. The formal definition of the knowledge representation and a formal syntax and semantics for a specific feature description language are what lack in our work. Formalizing our initial, somewhat empirical approaches into a framework similar to Cumby and Roth's work will allow use to gain better understanding of our approaches, and further suggest how to apply it to other domains.

CHAPTER 9

Future Work

The main contribution of our work is to introduce a new research direction, and build a ground for researches on how to systematically use domain knowledge. Our empirical results on handwritten Chinese character recognition are still much less competitive than human performance, which suggests that our approaches of using high-level knowledge are far from perfect. We believe much more can be done to make further improvements.

9.1 Other Approaches under the Proposed EBL Learning Framework

The three proposed approaches have limitations on how much solution knowledge they can communicate to SVM learning algorithms. For phantom example approach, the scalability of SVM learning algorithms limits the number of phantom examples we could augment into the training set. The component kernel functions we used to build feature kernels are still in the conventional kernel function form, which restricts the ability to express any further knowledge about the stroke-level features, except the presence of them. EA-SVM uses explanations to augment the training examples. Only one explanation is build for each training example. For small data set, the explanations can only convey small amount of extra knowledge.

The good news is that the SVM is still a relatively new research field, and the importance of knowledge to SVM starts to be recognized by more and more researchers. With our proposed EBL learning framework, we could take any new approaches of using solution knowledge as our component. One promising future work is to combine our learning framework with knowledge-based SVM. In that work, the solution knowledge is expressed as expert's opinion of how to classify sub regions in the example space. Such knowledge could potentially more informative than the training examples, which are single points in the example space. What we need is an EBL component that can take domain knowledge and training examples, and produce solution knowledge in the form that can be used by knowledge-based SVM.

9.2 Unlabeled Data

We believe the approach can be fruitfully applied in a semi-supervised modality [Ratsaby, 1994]. Our EBL component produces a concept that represents deeper pattern about the classification problem. On the other hand, the unlabeled data provides information about underlying distribution of interest. A learner can benefit from these two complementary sources of information.

The unlabeled data could be incorporated into our proposed approaches in simple manners. For phantom example approach, we could employ the EBL concept to supply labels to unlabeled examples whenever we are confident about the labeling. In this way, the phantom examples could be more similar to real examples at pixel level. For Feature kernel approach, unlabeled data potentially could be beneficial for combining component kernel functions, as suggested in [Christiaini, 2002]. Finally, for EA-SVM algorithm, unlabeled examples could be used to augment training set if they can be explained in a class-independent manner. To see this, consider handwritten character problem, the methods of extracting salient features proposed in [Teow & Loe, 2000] and [Kadir , Zisserman, & Brady, 2004] can be used to explain unlabeled examples, since such knowledge is class independent. Such explained unlabeled examples can be used by our EA-SVM algorithm since the labels of the explained examples do not appear in the quadratic programming problem in EA-SVM.

9.3 Other Application Domains

Since our goal is to establish a systematic approach to incorporate domain knowledge, we would like to test our approaches in as many application domains as possible. One characteristic of our knowledge is they are about high-level knowledge. Therefore, we are interested in the domains where expert can specify high-level, hidden features.

Many vision problems satisfy such requirement. Also, in speech recognition problem, *phone* is a concept similar to stroke. Knowledge about phones should be available and useful for such application. In natural language understanding problems, experts often use human-engineered concepts to organize the understanding of relations among words. Applications in those domains not only can provide further test-bed for our approaches, but also could provide cognitive motivation of how human utilize domain knowledge.

CHAPTER 10

Conclusions

We proposed a new learning framework to combine ideas of explanation-based learning with statistical learning algorithms, such as SVMs, to build a better learning system that can extract information from both domain knowledge and training samples. We have shown that effective solution knowledge can emerge from an EBL component that explains training examples in terms of general domain knowledge. A statistical learning algorithm augmented with such solution knowledge usually offers significantly better performance.

The inferential interaction between domain knowledge and training examples in our EBL component sets our work apart from other SVM approaches to prior knowledge (e.g., Schölkopf et al, 1998 and Fung & Shavlik, 2003). In our view, the purpose of domain knowledge is to introduce a high-level high-information vocabulary of pre-existing abstract features (such as "strokes" for handwritten characters). The explanation process, guided by the training examples, relates these high-level organizing features to input features. In this way, the classification patterns need not emerge purely empirically.

Our results demonstrate significant improvements even though the explanations are simple and the domain knowledge is approximate and not specifically engineered for the task. This work takes a first step refocusing machine learning on the principled incorporation of prior domain knowledge.

Bibliography

Aizerman, M.A., Braverman, E.M., and Rozoner, L.I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821-837.

Bartlett, P., Long, P.M., and Williamson, R.C. (1996). Fat-shattering and the learnability of real-valued functions. J*ournal of Computer and Systems Sciences,* 52(3):434-452.

Bartlett, P. and Shawe-Taylor, J. (1999). Generalization performance of support vector machines and other pattern classifiers. In Schölkopf, B., Burges, C., and Smola, A. J., editors, *Advances in Kernel Methods --- Support Vector Learning*, 43-54.

Blum A. and Mitchell, T.(1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 92-100.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jachel, K., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., and Vapnik, V. (1994) Comparison of classifier methods: a case study in handwritten digit recognition. *In Proceedings of 12th International Conference on Pattern Recognition and Neural Networks*, 77-87.

Bousquet, O. and Herrmann D.J.L. (2003). On the Complexity of Learning the Kernel Matrix. *Advances in Neural Information Processing Systems*, 15:415-422.

Brodie, M., and DeJong, G. (2001). Iterated Phantom Induction: A Knowledge-Based Approach to Learning Control. *Machine Learning, 45*(1), 45-76.

Bucher P., and Bairoch A. (1994). A generalized profile syntax for biomolecular sequences motifs and its function in automatic sequence interpretation. In Altman R., Brutlag D., Karp P., Lathrop R., Searls D., (Eds).ISMB-94; *Proceedings 2nd International Conference on Intelligent Systems for Molecular Biology*, 53-61.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955-974.

Burges C. and Schölkopf. B. (1997). Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, (Eds), *Advances in Neural Information Processing Systems* 9: 375-381.

Chown, E., Dietterich, T. G. (2000). A divide-and-conquer approach to learning from prior knowledge. In *Proceedings of the Seventeenth \ International Conference on Machine Learning*, 143-150.

Cohen. W.W. (1992). Compiling Prior Knowledge into an Explicit Bias. In D. Sleeman and P. Edwards, editors, *Proceedings of the 9th International Workshop on Machine Learning*, 102--110.

Conover, W.J., (1980). Practical Non-Parametric Statistics, 2nd edn. John Wiley and Sons, New York.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273-297.

Courant R. and Hilbert, D. (1953). Methods of Mathematical Physics. Interscience.

Crammer, K., Keshet, J. and Singer, Y. (2003). Kernel Design using Boosting. *Advances in Neural Information Processing Systems (NIPS)* 15: 537-544.

Cristianini, N. and Shawe-Taylor, J. (2000.) An Introduction to Support Vector Machine (and other kernel-based learning methods) Cambridge University Press.

Cristianini, N., Shawe_Taylor, and J. Lodhi, H. (2001). Latent Semantic Kernels. *Journal of Intelligent Information Systems (JJIS)*. 18(2): 127-152.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. (2002). On Kernel-Target Alignment. In *Advances in Neural Information Processing Systems (NIPS),* 14, 367-373.

Cumby C. and Roth D., (2000) Relational representations that facilitate learning. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning,* 425-434.

Cumby C. and Roth D., (2002). Learning with feature description logics. *In Proceedings of the 12th International Conference on Inductive Logic Programming,* 32-47.

Cumby C. and Roth D., (2003) On Kernel Methods for Relational Learning. *Proc. of the International Conference on Machine Learning (ICML),* 107-114.

DeCoste, D., and Schölkopf, B. (2002). Training Invariant Support Vector Machines. *Machine learning*, 46: 161-190.

DeJong, G. (1997). Explanation-Based Learning. In A.B. Tucker, editor, *The Computer Science and Engineering Handbook*, 499-520. CRC Press.

DeJong, G. (2004). Explanation-Based Learning. In A. Tucker (Editor-in-Chief), *Computer Science Handbook, s*econd ed., Chapman and Hall/CRC and ACM. 68.1-68.18.

DeJong G., and Mooney R., (1990). Explanation-Based Learning: an alternative view, *Machine Learning* 2, 145-176.

Dong, J. X., Krzyzak, A., and Suen, C.Y. (2003). High accuracy handwritten Chinese character recognition using support vector machine. In *Proceedings of International Workshop on Artitifical Neural Networks on Pattern Recognition.* 12-18.

Drucker, H., Schapire, R., and Simard, P. (1993). Boosting Performance in Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence, 7*(4), 705-719.

Fan, R.-E., Chen, P.-H., and Lin C.-J. (2005). Working set selection using the second order information for training SVM. *Technical report,* Department of Computer Science, National Taiwan University.

Freedman, D., Pisani, R., and PurvesStatistics, R. (1997). Statistics. Third Edition, Norton.

Fung, G., Mangasarian, O. and Shavlik, J. (2003). Knowledge-Based Nonlinear Kernel Classifiers. *16th Annual Conference on Learning Theory*. 102-113.

Haussler D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artificial Intelligence,* 177 – 221.

Herbrich, R. and Graepel T. (2001). A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In T.K Leen, T.G.Dietterich, and V. Tresp (Eds), *Advances in Nerual Information Processing Systems* (NIPS) 13, 224-230.

Hough, P.V.C. (1959). Machine analysis of bubble chamber pictures, *International Conference on High Energy Accelerators and Instrumentation (CERN).*

Jaakkola, T., Diekhaus, M., and Haussler, D. (1999). Using the fisher kernel method to detect remote protein homologies. *The Seventh International Conference on Intelligent Systems for Molecular Biology*. 149-158.

Joachims. T. (1997). Text categorization with support vector machines. Technical report, LS VIII Number 23, University of Dortmund,. ftp://ftp-ai.informatik.uni-dortmund.de/pub/Reports/report23.ps.Z.

Jordan M. I. (ed). (1998). Learning in graphical models. MIT Press.

Kadir, T. , Zisserman, A. and Brady, M. (2004). An affine invariant salient region detector. *Proceedings of the 8th European Conference on Computer Vision*, 404-416.

Kandola, J., Shawe-Taylor, J., and Cristianini, N. (2002). Optimizing kernel alignment over combinations of kernels. *NeuroCOLT* Technical Report NC-TR-02-121.

Karplus, K., Sjolander,K., Barrett, C., Cline, M., Haussler, D., Hughey, R., Holm, L., Sander, C., (1997). Predicting protein structure using hidden Markov models. *Proteins: Structure, Function and Genetics*, Suppl 1:134-139.

Kwok, J. T., and Tsang, I. W. (2003). Learning with idealized kernels.In *Proceedings of the Twentieth International Conference (ICML),* 400-407.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., L. Ghaoui, El, and Jordan, M. I. (2004). Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research* 5: 27-72.

Leslie, C. and Kuang, R. (2003). Fast Kernels for Inexact String Matching. *16th Annual Conference on Learning Theory*. 114-128.

Leslie, C., Eskin, E., and Noble, W. (2002). The spectrum kernel: a string kernel for SVM protein classification. In *Proc. Pacific Symposium on Biocomputing*. 564-575.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. J. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1: 541-551.

LeCun, Y., Jachel, K., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and Vapnik, V. (1995a). Comparison of learning algorithms for handwritten digit recognition. In Fogelman-Soulie F. and Gallinari, P., (eds), *Proceedings of International Conference on Artificial Neural Networks,* 2: 53-60.

LeCun, Y., Jachel, K., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and Vapnik, V. (1995b). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks,* 261-276.

Lewis D.D. (1997). Reuters-21578 text categorization test collection. http://www.daviddlewis.com/resources/testcollections/reuters21578/

Minton, S. (1988). Quantitative Results Concerning the Utility of Explanation-Based Learning, *Seventh National Conference on Artificial Intelligence*, 564-569.

Mitchell, T. (1997). Machine Learning. New York: McGraw-Hill.

Mitchell T.M., Keller R.M., Kedar-Cabelli S.T., (1990). Explanation-Based Generalization: A Unifying View, *Machine Learning* 1, 47-80.

Ong, C. S., Smola, and A. J. (2003). Machine learning with hyperkernels. *Machine Learning, Proceedings of the Twentieth International Conference (ICML)*, 568-575.

Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing*, Eds. J. Principe, L. Giles, N. Morgan, E. Wilson, 276 – 285.

Pearl. J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77 (2), 257-286.

Ratsaby, J. (1994). The complexity of learning from a mixture of labeled and unlabeled examples, *Ph.D. thesis*, University of Pennsylvania.

Russell S, Norvig P. (2003). Artificial intelligence: A modern approach, Second Edition, Englewood Cliffs, New Jersey: Prentice Hall.

Saito, T., Yamada, H. and Yamamoto K. (1985). On the data base ETL 9 of handprinted characters in JIS Chinese characters and its analysis. *IEICE Transactions*, J68-D(4):757--764.

Schmidt, M. (1996). Identifying speaker with support vector networks. In *Proceedings of the 28th Symposium on the Interface (INTERFACE-96).*

Schölkopf, B.; Burges, C.; and Vapnik, V. (1995). Extracting support data for a given task. In: Fayyad, U.M., Uthurusamy, R. (eds.): *Proceedings, First International Conference on Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA.

Schölkopf, B.; Burges, C., and Vapnik, V. (1996). Incorporating invariances in support vector learning machines. In: C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff (eds.): *Artificial Neural Networks - ICANN'9* 6*:*47-52.

Schölkopf, B., Simard, P., Smola, A. J., and Vapnik. V. (1998). Prior knowledge in support vector kernels. *Advances in Neural Information Processing Systems 10*: 640-646.

Schölkopf, B., and Smola, A. J. (2002). Learning with kernels, Cambridge, MA: MIT Press.

Shawe-Taylor, J., Bartlett, J., P. L., Williamson, R. C., and Anthony, M. (1998) Structural risk minimization over data-independent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926-1940.

Shawe-Taylor, J. and Cristianini, N. (2002). On the generalization of soft margin algorithms. IEEE Transactions on Information Theory 48(10): 2721-2735.

Shawe-Taylor J. and Cristianini, N. (2004). Kernel methods for pattern analysis, Cambridge University Press.

Shawe-Taylor, J. and Williamson, R. (1999). Generalization performance of classifiers in terms of observed covering numbers. In P. Fischer and H. U. Simon (Eds), *Proceedings of the European Conference on Computational Learning Theory*, 285-300.

Shi, D., Gunn, S.R., Damper, R.I. (2003). Handwritten Chinese radical recognition using nonlinear active shape models, *IEEE transactions on pattern analysis and machine intelligence,* vol. 25-2, 277-280.

Simard, P. and Lecun, Y. (1994). Memory based character recognition using a transformation invariant metric. In *Proc. 12th IAPR International Conference on Pattern Recognition*, Jerusalem, 2: pp. 262-267.

Simard, P. Y., LeCun, Y. A., Denker, F. S., and Victorri, B. (1998). Transformation invariance in pattern recognition – tangent distance and tangent propagation. *Lecture Notes in Computer Science,* 1524:239-254.

Simard P., Victorri, B., LeCun, Y. and Denker, J. (1992). Tangent prop – A formalism for specifying selected invariances in an adaptive network. Advances in Neural Information Processing Systems 4, Morgan Kaufmann, San Mateor, CA.

Tecuci G. and Kodratoff Y., (1990). Apprenticeship learning in imperfect domain theories, in Kodratoff Y. and Michalski R. (eds), *Machine Learning*, vol 3, Morgan Kaufmann,

Teow, L. and Loe, K. (2000). Handwritten digit recognition with a novel vision model that extracts linearly separable features, in *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2000)*, Vol 2: 76-81.

Vapnik. V. (1995). The nature of statistical learning theory. Springer-Verlag, New York.

Vapnik, V. N. (1998). Statistical learning theory. New York: Wiley.

Vapnik, V. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* 16(2), 264-281.

VITA

Qiang Sun was born in Tianjin, People's Republic of China on March 30th, 1972. He graduated from Peking University, People's Republic of China in 1996 with a Bachelor of Science degree in Biochemistry. After college, Qiang Sun attended graduate school in the Molecular Biology and Biochemistry department at Wesleyan University at Connecticut in August 1996, and completed his Master of Arts degree study in August, 1998. In May, 1999, Qiang Sun started his graduate study in the Computer Science department in University of Illinois at Urbana-Champaign. After the ph.D. study, he will begin work at Alexa internet in San Francisco in 2005.