Incorporating Prior Domain Knowledge Into Inductive Machine Learning

Ting Yu yuting@it.uts.edu.au
Tony Jan jant@it.uts.edu.au
Simeon Simoff simoff@it.uts.edu.au
John Debenham debenham@it.uts.edu.au
The Faculty of Information Technology, University of Technology, Sydney
Broadway, NSW 2007, Australia

Abstract

The paper reviews the recent developments of incorporating prior domain knowledge into inductive machine learning, and proposes a guideline that incorporates prior domain knowledge in three key issues of inductive machine learning algorithms: consistency, generalization and convergence. With respect to each issue, this paper gives some approaches to improve the performance of the inductive machine learning algorithms and discusses the risks of incorporating domain knowledge. As a case study, a hierarchical modelling method, VQSVM, is proposed and tested over some imbalanced data sets with various imbalance ratios and various numbers of subclasses.

1. Introduction

As a major subfield of artificial intelligence, machine learning has gained a broader attentions in recent years. Especially, the internet makes a variety of information more easily accessible than ever before, creating strong demands for efficient and effective methods to process a large amounts of data in both industries and scientistic research communities. The application of machine learning methods to large databases is called *Data Mining* or *Knowledge Discovery* (Alpaydin, 2004). Despite the existence of hundreds of new machine learning algorithms, from a machine learning academic researcher's perspective, the current state of this research is very preliminary. If the ultimate goal is to develop a machine that is capable of learning at the level of the human mind, the journey has only just begun (Silver, 2000). This paper is but one step in that journey by exploring the outstanding question in inductive machine learning: *How can a learning system represent and incorporate prior domain knowledge to facilitate the process of learning?*

The vast majority of standard inductive learning machines are data driven, rely heavily on the sample data and ignore most of existing domain knowledge. The reasons why the majority of machine learning techniques ignore the domain knowledge vary: traditionally, researchers in machine learning have sought general-purpose learning algorithms rather than within a certain domain. However, the domain experts often do not understand complex machine learning algorithms, and thus cannot incorporate their knowledge into machine learning. The representations of domain knowledge are various, and there is not one universal format to represent them in computer systems. At the same time, in certain domains, the researchers in machine learning often have little idea of certain existing knowledge, and

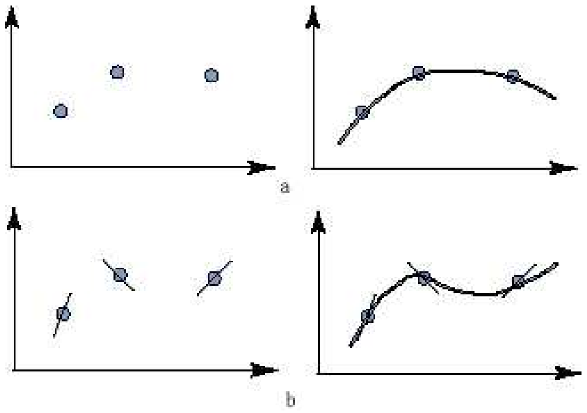

Figure 1: An Example of Inductive Machine Learning without and with Domain Knowledge

even though they have intentions to find some knowledge, it is still difficult to encode the knowledge into the computer systems.

In real-world problems, prior domain knowledge is valuable to be incorporated into a practical inductive learning system. Here an example is used to provide an insight of the ways how prior domain knowledge enhances the performance of a simple inductive learning system.

In the Figure 1, three points stand for three training examples. In the case of standard machine learning, a smooth curve is learned to go across three points in Figure 1a. But if some domain knowledge exists, e.g. gradients at each point, the resultant curve needs not only go across the points, but also take into account of the gradients. Thus the resultant curve (Figure 1b) differs very obviously from the previous one.

In the problem described in Figure 1, the domain knowledge, i.e. the gradient, could be treated as some additional constraints apart from the location of the data points. Under the optimization theory, these gradients could be represented as additional constraints. In some machine learning algorithms, the loss functions, e.g. least square error, can be modified to contain those additional components to penalize the violation to these constraints. Therefore, through these additional constraints with certain weights, the final result would be influenced by this particular domain knowledge.

A bit further, in many real life applications, a large proportion of domain knowledge is not perfect, which might not be completely correct or cover the whole domain. Thus, it is improper that the resultant model relies only on either the domain knowledge or training data, and then the trade-off is a crucial issue to reach the optimal result. Detailed discussion will be developed in the following sections of this paper. A motivation for this research comes from the desire to make the best use of information from various sources to overcome the limitation of the training examples, especially in some real-world domains where extensive prior knowledge is available.

The section 2 and 3 of this paper introduce some basic concepts of inductive machine learning and prior domain knowledge. The section 4 and 5 of paper present a guideline for incorporating prior domain knowledge into inductive machine learning and use this guideline to analysis some existing methods of incorporating domain knowledge into inductive machine learning. The section 6 and 7 of paper propose a new hierarchical modelling method, VQSVM, and test it over some imbalanced data sets.

2. Inductive Machine Learning

A precise definition of machine learning has been given by (Mitchell, 1997a): "A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." As a branch of machine learning, inductive machine learning is an inference process consisting of two major steps:

1. A learning system L constructs a hypothesis space H from a set of training examples (or observations) S, for example a set of training examples consisting of input-output pairs $S = \{\langle x_i, c(x_i)\rangle\}$, and the inductive bias B predefined by the learning system and the task T.

2. The learning system then searches through the hypothesis space to converge to an optimal hypothesis f consistent with training examples and also performing well over other unseen observations.

The inductive learning process can be seen as a process of function approximation, in other words, an inductive learning system generally has a capability of constructing a rather complex model $f(x)$ to approximate any continuous or discontinuous unknown target function $c(x)$, i.e. $f(x) \to c(x)$, with sufficient training examples S.

2.1 Consistency and Inductive Bias

Given a training data set, e.g. input-output pairs (x_i, y_i), which are drawn independently according to a unknown distribution D from unknown function $c(x)$, so-called I.I.D. assumption, the task of the machine learning is to find a model (or hypothesis) $f(x)$ within the given hypothesis space H to best approximate to the $c(x)$, i.e. $f(x) \to c(x)$ with respect to the given data set $X \times Y$ and $f(x) \in H$. The best approximation of the target function $c(x)$ is a function $f : X \to Y$, such that the error, averaged over the distribution D, is minimized. However the distribution D is unknown, so the average cannot be computed, and then the inductive machine learning is analogue to the *function approximation* from a training data set.

The process of choosing and fitting (or training) a model is usually done according to formal or empirical versions of inductive principles. All approaches share the same conceptual goal of finding the "best", the "optimal", or most parsimonious model or description that captures the structural or functional relationship in the data. The vast majority of inductive machine learning algorithms employ directly or indirectly two inductive principles (Galindo & Tamayo, 2000): 1) Keep all models or theories *consistent* with data, and 2) Modern version of Occam's razor (Choose the most *parsimonious* model that fits the data).

Firstly, the first principle "Keep all models or theories consistent with data" requires a measurement of the goodness of the resultant function (or model). In inductive machine learning community, the *loss function* $L(f)$ or *risk function* $R(f)$ is one of often-used approaches to measure the distance between the true function $c(x)$ and the estimated function $f(x)$ (Poggio, 2003)(Abu-Mostafa, 1995):

A learner L, e.g. a machine learning algorithm listed previously, must output some hypotheses $f(x)$ (or model) as function approximation to the underlying unknown function $y = c(x)$ from samples. The function approximation looks like: $y^* = f(x)$, and the *Risk (or Loss) function* is written as:

$$R[f] = V(f(x), y) \tag{1}$$

where V is an arbitrary metric which measures the difference between the estimated values and true values.

Actually there are two notions of risk functions: the true risk function of a hypothesis $f(x)$ is the probability that $f(x)$ will misclassify an instance drawn at random according to D (Mitchell, 1997a)

$$R(f) \equiv Pr_{x \in D}[f(x) \neq c(x)] \tag{2}$$

And the sample (empirical) risk of a hypothesis $f(x)$ with the respect to target function $c(x)$ and the data sample $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is:

$$R_{emp}(f) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), c(x)) \tag{3}$$

A function f is *consistent with the training examples* S, when $R_{emp}(f) = 0$, and is *the equivalence of the target function* c when $R(f) = 0$.

It is very rare that the training examples covers the whole population, and then constructing a hypothesis memorizing all of the training examples does not guarantee a good model. Very often, this would produce an effect known as *over-fitting* which is analogous to over-fitting a complex curve to a set of data points (Silver, 2000). The model becomes too specific to the training examples and does poorly on a set of test examples. Instead, the inductive machine learning system should output a hypothesis performing equally well over both training examples and test examples. In other words, the output hypothesis must *generalize* beyond the training examples.

Here the second inductive principle, i.e. modern version of Occam's razor, is the most accepted and widely applied for generalization. Actually, the Occam's razor is only one example of a more broad concept, inductive bias. The *inductive bias* denotes any constraint of a learning system's hypothesis space, beyond the criterion of consistency with the training examples (Silver, 2000).

Mitchell (Mitchell, 1997a) gave a precise definition of the inductive bias: Consider a concept learning algorithm L for the set of examples X. Let c be an arbitrary concept defined over X, and let $S = \{\langle x, c(x) \rangle\}$ be an arbitrary set of training examples of c. Let $L(x_i, S)$ denote the classification assigned to the example x_i by L after training on the data S. The inductive bias of L is defined to be any minimal set of assumptions B such that for any target concept c and corresponding training examples S:

$$(\forall x_i \in X)[(B \wedge S \wedge x_i) \vdash L(x_i, S)] \tag{4}$$

where $a \vdash b$ indicates that b can be logically deduced from a. The inductive bias of a learning system can be considered a learning system's preference for one hypothesis over another (Silver, 2000).

Currently most of the inductive machine learning algorithm have their own general-purpose inductive bias. For example, the support vector machine (SVM), which will be discussed later, prefers a hypothesis with the maximum margin. This inductive bias is fixed and applied to all tasks the SVM deals with. But, ideally, a learning system is able to customize its inductive bias for a hypothesis space according to the task being learned.

The inductive bias has direct impacts on the efficiency and effectiveness of a learning system, and a major problem in machine learning is that of inductive bias: how to construct a learner's hypothesis space so that it is large enough to contain a solution to the task, yet small enough to ensure a tractable and efficient convergence (Baxter, 2000). It would be ideal that the inductive bias supplies a hypothesis space containing only one single optimal hypothesis, and in this case, the inductive bias already completes the task of the learning system.

Some learning algorithms have no inductive bias, so-called *unbiased learners*, and very often their outputs perform very poorly over the unseen data, and the majority of learning algorithms have more or less inductive bias, so-called *biased learners.* This paper only discusses the biased learners.

2.2 Statistical Learning theory

Consider a concept class C defined over a set of examples X of length n and a learner L using hypothesis space H. C is *PAC (Probably Approximately Correct) -learnable* by L using H if for all $c \in C$, distributions D over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such that $R_D(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n, and $size(c)$ (Mitchell, 1997a).

This definition raises three crucial issues of a learner (a learning algorithm L), which impact the efficiency and effectiveness of a learner. First, L must, with arbitrarily high probability $(1 - \delta)$, output a hypothesis h being *consistent* (or an arbitrarily low risk (or error) ϵ) with the training examples generated randomly from X according to a distribution D. Secondly, L must, with an arbitrarily high probability $(1 - \delta)$, output a hypothesis h being *generalized* for all examples (including unseen examples) with an arbitrarily low risk. Because the training examples rarely cover the whole population and the there are always unseen examples, the generalization aims to reduce the difference between the approximation based on the observed examples and the approximation based on the whole population. A bit further, the generalization is closely related to the complexity of the hypothesis space H. Thirdly, the *convergence* to the optimal hypothesis h must be efficient, that is in time that grows at most polynomially with $1/\epsilon$, $1/\delta$, n, and $size(c)$. Otherwise, the convergence will be a NP(Non Polynomial)-complete problem, and the learner L is not of use in practise. The convergence is closely related to the computation complexity. In summary, the function approximation produced by a learning algorithm must be consistent, must exist, must be unique and must depend continuously on the data, and the approximation is "stable" or "*well-posedness*" (Poggio, 2003).

As previously addressed, the available examples often are insufficient to cover the whole population, and thus the consistency of the available examples does not guarantee the consistency of all examples (including the available and unseen examples) generated by the target concept c. Furthermore, given some training data, it is always possible to build a

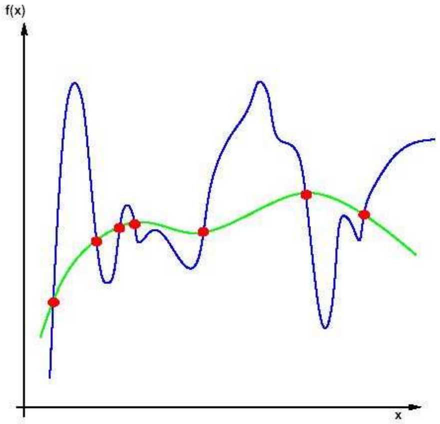

Figure 2: More than one functions that fits exactly the data. The red points stand for the observed data, and blue and green curves pass them exactly without any error. Without further restriction, the empirical risk function cannot indicate which is better. The *complexity* of the blue curve is higher than another one (Poggio, 2003)

function that fits exactly the data. This is generally compromised in presence of noise, and the function often is not the best approximation. Thus it results in a poor performance on unseen examples. This phenomenon is usually referred to as *overfitting* (Bousquet, Boucheron, & Lugost, 2004). The crucial problem is to best estimate the consistency of all examples, but the insufficiency of the training examples makes the learner very difficult to measure true risk $R[f]$ directly. Thus one of the main purposes of learning theory is to minimize the true risk $R[f]$ by minimizing the empirical risk $R_{emp}[f]$.

Additionally, it is always possible to build multiple functions to exactly fit the data. Without extra information about the underlying function, there are no means to favor one function over another one. This discussion is often called the 'no free lunch' theorem (Bousquet et al., 2004). From the beginning, any machine learning algorithm already makes assumptions apart from the training data , for example i.i.d. assumption. Hence, this leads to the statement that data cannot replace knowledge, and can be expressed mathematically as:

$$Generalization = Data + Knowledge \tag{5}$$

Standard learning systems employ their general-purpose inductive bias as a type of domain knowledge. For example, the modern version of Occam's razor (Choosing the most parsimonious model that fits the data) is employed to converge to an optimal hypothesis by minimizing the complexity of the function approximation. However, this only gives a qualitative method, and a quantitative method is needed to measure the difference between true risk and empirical risk functions.

One inequality gives a upper bound of the true risk of any function in a given hypothesis space with respect its empirical risk: Assume there are N functions in the given hypothesis space, and for all $\delta > 0$ with probability at least $1 - \delta$

$$\forall f \in H, R(f) \leq R_{emp}(f) + \sqrt{\frac{logN + log\frac{1}{\delta}}{2n}} \tag{6}$$

where $H = \{f_1, ..., f_N\}$ and the n is the sample size.

This inequality assumes that the hypothesis space is countable and finite. However in many case, this assumption is not established, The inequality needs to be expanded to either a countable or an uncountable hypothesis space consisting of infinite functions.

In the countable and infinite case, with a countable set H, the inequality becomes: choosing $\delta(f) = \delta p(f)$ with $\sum_{f \in H} p(f) = 1$, the union bound yields:

$$P[sup_{f_n \in H}(R(f) - R_{emp}(f)) > \sqrt{\frac{log\frac{1}{\delta p(f)}}{2n}}] \leq \sum_{f \in H} \delta p(f) = \delta(Bousquetet\ al., 2004) \tag{7}$$

In the uncountable hypothesis space consisting of infinite functions, the modification introduces some new concepts, which include the so-called Vapnik Chervonenkis (VC) dimensions. When the hypothesis space is uncountable, the hypothesis space is considered over the samples, and that is, the size of the hypothesis space is the number of possible ways in which the data can be classified in case of the binary classification(Bousquet et al., 2004). The upper bound becomes (Vapnik, 1995): For any $f \in H$, with a probability of at least $1 - \delta$,

$$R[f] \leq R_{emp}[f] + 2\sqrt{2\frac{h(\log\frac{2en}{h} + 1) - \log(\delta/2)}{n}} \tag{8}$$

where n is the number of training data and h is the VC Dimension, which indicates the capacity of a hypothesis space (or a function class) (Vapnik, 1995). All these measurements is be finite, no matter how big the hypothesis space is. In contrast, the Bayesian methods place prior distribution $P(f)$ over the function class. But in statistical learning theory, the VC Dimension takes into account the capacity of the class of functions that the learning machine can implement (Scholkopf & Smola, 2002a).

The above discussion provides insight into an approach for generalization: to measure an upper bound of the true risk of functions in a given hypothesis space. This type of approaches is called *Structural Risk Minimization* (SRM). Apart from this method, *regularization* is another way to prevent the overfitting: the method defines a *regularizer* on a model f, typical a norm $\| f \|$, and then the regularized empirical risk is $f_n = \arg\min_{f \in H} R_{emp}(f) + \lambda \| f \|^2$. Compared to the SRM, there is a free parameter λ, called the regularization parameter which allows to choose the right trade-off between fit and complexity (Bousquet et al., 2004). This method is an often-used way of incorporating extra constraints into the original empirical risk function, and is easily employed to incorporate prior domain knowledge.

So far the learning machine has been simplified as a set of functions and induction bias. For instance, the SRM tells us that the true risk is upper bounded by a combination of the empirical risk and the capacity of a function class. The optimal model then reaches a trade-off between small capacity and small error.

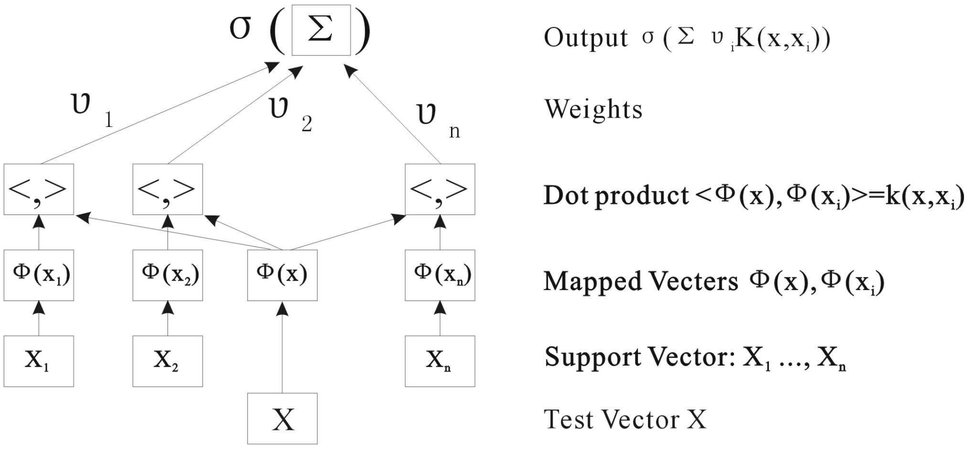

Figure 3: Architecture of SV Machines

2.3 Support Vector Machine

Support Vector Machine extended the linear learning machines into nonlinear problems, integrates the previous contents together and is maturated by Vapnik in the mid 90s. At the beginning, it only deal with the binary classification, i.e. pattern recognition, but shortly the multiple classification vector machine and support vector regression were developed. The basic process of a support vector machine is (see figure 3):

1. Map the training data from the original space (*input space*) into a higher dimensional space (*feature space*), i.e. a hypothesis space, by using kernels, in order to transform a linearly nonseparable problem into a linearly separable one;

2. Within the feature space, i.e. the hypothesis space, a hyperplane with maximum margins is constructed to separate two classes in case of binary classification.

In order to find the hyperplane, the standard support vector machine represents this objective function in a dual form and employs quadratic programming to solve the optimization problem.

$$\Theta(\alpha) = min_{w,b}L(w,b,\alpha) = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j\langle x_i \cdot x_j\rangle + \sum_{i,j=1}^{N}\alpha_i \tag{9}$$

The decision function of support vector machine is:

$$y(x) = sgn\{w \cdot \phi(x) + b\} = sgn\{\sum_{i=i}^{n}\alpha_i y_i K(x,x_i) + b\} \tag{10}$$

3. Prior Domain Knowledge

The *prior domain knowledge* is all of the auxiliary information about the learning task that can be used to guide the learning process, and the information comes from either some other discovery processes or domain experts.

From the concept of the domain knowledge, the domain knowledge is the relevant knowledge within one particular domain. Within one domain, different specialists and experts may use and develop their own domain knowledge. In contrast, knowledge which may be efficient in every domain is called *domain-independent knowledge*. Domain knowledge inherits some properties from the concept of knowledge, and it is represented in various forms based on their types and contexts. In the case of learning systems, the domain is referred to the learning tasks. Due to its various representation, the method feasible to one type of domain knowledge may be infeasible to another type of domain knowledge. At the same time, the sources of domain knowledge is not restricted by domain experts and it is more broad, and then domain knowledge includes all knowledge apart from the numerical observations.

Unfortunately domain knowledge is always informal and ill structured, and it is difficult to incorporate domain knowledge into learning systems. Very often domain knowledge is represented as a set of rules in knowledge bases, and this is the so-called "expert system". Due to its context-dependence, the methods of transformation are varying, and not restricted by the logic formats.

3.1 Functional or Representational Form

In what form should prior domain knowledge be incorporated into a new task? In the case of sequential multi-task learning, the simplest method of retaining task knowledge it to save all training examples of that task. This form of domain knowledge is referred as *a functional form of knowledge* (Silver, 2000). Other functional forms of domain knowledge include the storing or modeling of the parameters of learning process, e.g. hypo-parameters in kernel methods. Another form of domain knowledge is the accurate representation of a learned hypothesis developed from the training examples. That is *the representational form of domain knowledge* (Silver, 2000). Clearly the advantages of the representational form of domain knowledge are that the results are more compact and require less storage space. Its pronounced disadvantage is the potential loss of accuracy from the original training examples (Silver, 2000).

3.2 Relatedness Between Tasks and Domain Knowledge

With respect to one task, it is unnecessary to incorporate all domain knowledge in the process of learning. As was discussed in the chapter 1, prior domain knowledge is highly context dependent, for any consideration of the degree of task relatedness in context of inductive machine learning. Even the same prior domain knowledge could produce different impacts on difference tasks. When the use of domain knowledge results in a more accurate hypothesis than could have achieved with only the training examples, a *positive domain knowledge* is said to have occurred. Conversely, a *negative domain knowledge* results in a hypothesis of lesser accuracy.

There are no general definition of relatedness between tasks and domain knowledge. According to the current understanding of these two concepts, some indirect means of measuring relatedness are addressed in Daniel Silver's PhD thesis. He gives a general definition of task relatedness in the context of functional transfer:

> Let T_k and T_0 be two tasks of the same domain with training examples S_k and S_0, respectively. The relatedness of T_k with respect to T_0 in the context of learning system L, that uses functional knowledge transfer, is the utility of using S_k along with S_0 towards the efficient development of an effective hypothesis for T_0 (Silver, 2000).

This definition let him give the statement: Inductive Bias = Domain Knowledge + Task Relatedness:

$$K \wedge R \rightarrow B_D \tag{11}$$

where K is the domain knowledge retained by a knowledge based inductive learning system L, R is the meta-level bias that selects the most related knowledge from domain knowledge for developing a new task T_0 and B_D denotes domain knowledge based inductive bias (Silver, 2000). R serves the secondary function of originating from domain knowledge, and indicating the task relatedness. However, in a learning process, the domain knowledge does not only function as an inductive bias, but also facilitates other aspects of the process, for example, the preprocess of the training examples, the initiation of a hypothesis space, the initiation of the start point of the convergence, and the direction of convergence etc..

This definition of relatedness between tasks is limited within multi-task learning and sequential learning, both of which transfer knowledge between multiple tasks. However, the related task is one of sources of domain knowledge, resulting in a more general definition of relatedness between domain knowledge and tasks. The definition is: Let T_0 be one task with training examples S_0, and K be a set of domain knowledge. The relatedness of K with respect to T_0 in the context of learning system L is the utility of using K along with S_0 towards the efficient development of an effective hypothesis for T_0.

In other words, if the effectiveness and efficiency of a learning system L is improved with the domain knowledge K, the domain knowledge K is related to the task T_0 with respect to the learning system L. The previous definition of the relatedness between tasks is an example whereby the domain knowledge happens to be discovered by other learning systems for tasks within the same domain.

If the relatedness between one set of domain knowledge and a task is significant, the set of domain knowledge K is notated by the related domain knowledge K_D of the task T with respect to the learning system L: Related Domain Knowledge = Domain Knowledge + Task Relatedness. This is represented as:

$$K \wedge R \rightarrow K_D \tag{12}$$

and $B_D \subseteq K_D$. It is undoubted that the inductive bias plays a major role in the domain knowledge. The previous definition of inductive bias (See equation 4) can therefore be rewritten as:

$$(\forall x_i \in X)[(K_D \wedge B_O \wedge S \wedge x_i) \vdash L(x_i, S)] \tag{13}$$

where B_O is the other inductive bias independent from the task.

The definition of general relatedness between domain knowledge and tasks is abstract and there is no general way to measure it. When domain knowledge is learned by the related tasks, the relatedness between tasks reflects the relatedness between domain knowledge and tasks. The relatedness R between tasks is relatively easy to be measure, and it can be equated to the concept of similarity, and be expressed as a metric: $d(T_0, T_k)$, and therefore a metric space $(\{T_i\}, d)$, where $\{T_i\}$ is the set of tasks within one domain and d is the metric over the set. In this case, the knowledge transfer between tasks is closely related to another concept, *collaborative filtering*. The similarities include the *surface similarity* and *structural similarity* in differing degrees (Silver, 2000).

Some researchers have further explored learning methods able to assimilating the inductive bias, *meta-level learning*. Contrarily, standard machine learning is called as *base-level learning* (Thrun, 1996). Meta-level learning is already outside the PCA framework, as the PCA model simply takes the hypothesis space H as given and proceeds from there. Meta-level learning is supplied with a family of hypothesis spaces $M = \{H\}$, its goal being to find a inductive bias that is appropriate for the given task (Baxter, 2000). Here a set of related tasks constructs an *environment*, $E = \{T_0, T_1, \ldots, T_i\}$. Meta-level learning then aims to explore its environment in order to discover the inductive bias of a new task from other tasks in that same environment.

The measure of relatedness between tasks and domain knowledge implies another important issue: the various importance of domain knowledge. Suppose there is more than one piece of related domain knowledge, and the risk function (see equation 1) defined previously is rewritten as:

$$R(f) = V(R_{emp}, R_1, \ldots, R_t) \tag{14}$$

where R_k is an error on the training examples for domain knowledge $k = 1, \ldots, t$, and R_{emp} is the error on the training examples, such as an empirical risk (Abu-Mostafa, 1995).

Considering now the previous example where three points are with their gradients (see Figure 1), the standard learning system L_0 employs the second inductive principle (one selects the most parsimonious model that fits the data) as its general purpose inductive bias B_O and produces a smooth curve going through three points. With an additional inductive bias, the learning system L_1 is capable of taking into account the extra information that is gradients, and producing a better model with respect to the task T_0 to find a model fitting both the training examples and the gradients. The resultant model appears better than the previous one, and thus the domain knowledge that is gradients are related with respect to the task T_0 and the learning system L_1.

4. Consistency, Generalization and Convergence with Prior Domain Knowledge

Prior domain knowledge usefully enhance learning systems in multiple aspects. The performance of a learning system is generally determined by three key issues: consistency, generalization and convergence.

4.1 Consistency with Domain Knowledge

Mitchell's concept of consistency in (Mitchell, 1997a) is expanded in order to contain domain knowledge:

Definition 1: Assume that training examples and domain knowledge are correct without noises, and $c(x)$ is the underlying target function:

A hypothesis f is *consistent* with a set of training examples S and a set of related prior domain knowledge K_D if and only if $f(x) = c(x)$ for each example $\langle x, c(x)\rangle$ in S and $f(x)$ is consistent with any knowledge k in K_D:

$$Consistent(f, S, K_D) \equiv ((\forall\langle x, c(x)\rangle \in S) f(x) = c(x)) \wedge ((\forall k \in K_D) f(x) \Vdash k) \tag{15}$$

Note that the definition of consistency contains a strong assumption. Either observation S or domain knowledge K_D is perfectly produced by the underlying function $c(x)$. In other words, neither observation nor domain knowledge contains noise, and the coverage of observation and domain knowledge is sufficient enough to represent the unknown function $c(x)$. In practice however this assumption is difficult to be verified, as the observation and domain knowledge may be exposed to some noises or incorrect information. There may also exist some contradictions between observations and domain knowledge: $((\langle x, c(x)\rangle \in S) f(x) = c(x)) \wedge ((\exists k \in K_D) f(x) \nVdash k$ or $((\forall k \in K_D) f(x) \Vdash k) \wedge ((\exists\langle x, c(x)\rangle \in S) f(x) \neq c(x))$. This weak (or soft) consistency provides a sort of tradeoff, when such conflicts occur between them.

In the learning-from-examples paradigm, the estimation of the consistency between the hypothesis and the unknown target function is measured by the risk (loss) function, such as empirical risk $R_{emp}[f]$. In order to measure the disagreement between the hypothisis and the domain knowledge, a more general and abstract concept, the metric, is introduced to replace the risk functions. When the domain knowledge is measurable, such as the known gradients in some given examples (see Equation 1), the degree to which a hypothesis $f(x)$ is consistent with $K = \{k_1, \ldots, k_m\}$ is defined by a metric $d(.,.)$:

$$R_{knowledge}(f) = d(f(x), k_i), \forall k_i \in K_D \tag{16}$$

where the d satisfies three properties of metric: reflexive property, symmetric property, and triangle inequality. One example of this metric is a distance measurement $R_{knowledge}(f) = E(f(x) - k_i)$ in (Abu-Mostafa, 1995). Since the $R_{knowledge}(f)$ is supposed to measure the disagreement between $f(x)$ and the domain knowledge K_D, the $R_{knowledge}(f)$ should be zero when the $f(x)$ is identical to $c(x)$ and the K_D is perfectly produced by the $c(x)$:

$$R(f) = 0 \Rightarrow R_{knowledge}(f) = 0 \tag{17}$$

It is a necessary condition for $R_{knowledge}(f)$ to be consistent with the assertion that the domain knowledge is valid for the target function $c(x)$. It is worth noticing that the $R(f)$ is the unknown risk function between $f(x)$ and $c(x)$, and the $R_{emp}(f) = 0 \nRightarrow R_{knowledge}(f) = 0$. This condition is not necessary for soft domain knowledge, as characteristically it is only "approximately" valid (Abu-Mostafa, 1995).

Definition 2: The risk function is $w_S R_{emp}(f) + w_K R_{knowledge}(f)$ where the coefficients w_S and w_K assign the different weights to the empirical risk function and domain knowledge risk function, in order to balance the effects from each of them.

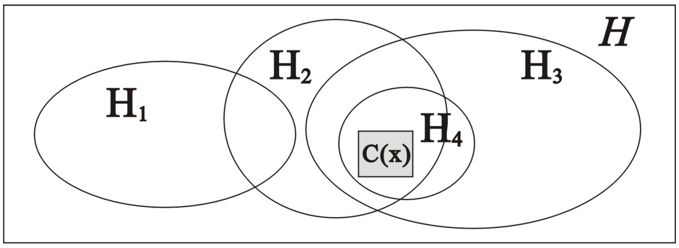

Figure 4: A set of hypothesis spaces

This definition is similar to the one given by Mitchell (Mitchell, 1997b). When domain knowledge is not directly measurable, such as in the format of logic rules, special metrics or transformations of the domain knowledge are required. Some examples are available in the following sections, for example consistency hints.

4.2 Generalization with Domain Knowledge

As was mentioned in the previous section, the domain knowledge is essential to the generalization: *Generalization* = *Data* + *Knowledge*. In this statement, the knowledge refers to the prior implemented by the given algorithm and the assumptions followed by the given algorithm, according to the no free lunch theory (Bousquet et al., 2004). In other words, an algorithm is better than another one when the prior implemented by the algorithm is better suited to these databases.

Beyond this well-known role, the domain knowledge plays an important role in initiating a proper hypothesis space H and deducing more appropriate inductive bias B_D to reduce the generalization error. In the previous decomposition of risk (see Equation **??**), the generalization risk consists of the approximation error, $R(f_H) - R$, and the estimation error $R(f_n) - R(f_H)$. Within the given hypothesis space, the approximation error cannot be reduced. In order to reduce the approximation error, the hypothesis space often is relaxed to contain at least one hypothesis f_H identical to the unknown target model $c(x)$. On the other hand, three upper bounds of true risk function show that increasing the size of hypothesis space also increases the estimation error. Therefore, a conflict occurs when two errors are reduced simultaneously. In order to makes the overall generalization error to reach a minimal point, the reductions of the two errors have to be balanced, that is the *variance/bias tradeoff*. Often it is difficult to discover this optimal solution by only using training examples. If auxiliary information of the hypothesis space and the training examples are available, it is worth incorporating it into the selection process, in order to select or initiate a more proper hypothesis space H. In the Figure 4, suppose that $c(x)$ is the unknown target function, and the optimal hypothesis space is the H_4 with a minimal generalization error. The H_3 contains the $c(x)$, but its size is big, which leads to a larger estimation error. In contrast, the H_1 has a smaller size, but does not include the target function $c(x)$, which leads to a larger approximation error.

This issue is closely related to *meta-level learning* or lifelong learning. Suppose that target functions are sampled from a hypothesis space, H, and a learner can choose its hy-

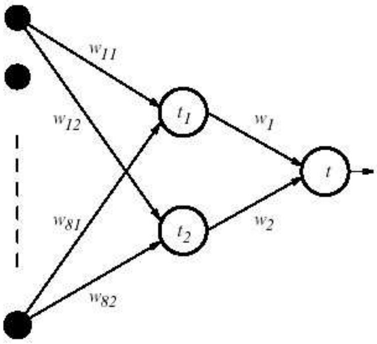

Figure 5: 8-2-1 Artificial Neural Network (Abu-Mostafa, 1995)

pothesis space from $\{H_0, H_1, \ldots, H_4\}$ prior to the arrival of the training examples from a target function $c(x)$ (See Figure 4) (Thrun, 1996). This type of machine learning is sometimes called *learning to learn*. A computer system has learned to learn if its performance at each task improves with the number of training examples and with the number of tasks in the family (Silver, 2000). This definition implies that within a family of tasks, the same form of knowledge is acquired and transferred as an inductive bias from one or more *source* tasks to a *target* task. The main purpose of this is to improve the performance of the target task.

Statistical learning theory has provided a set of inequalities to estimate the upper bounds of the difference between empirical and true risk. However, these upper bounds are derived from the worst case scenario, so that the upper bounds between the true risk and empirical risk defined in the learning theory are often too crude for real cases. In case of the upper bound for a finite and countable hypothesis space, the upper bound grows linearly with the number of functions N (see the inequality 6). In a large enough hypothesis space, this bound can easily become so great that it loses its precision. While domain knowledge is available to reduce the size of the hypothesis space, the resultant upper bound will be tighter.

When utilizing the upper bound for a infinite but countable hypothesis space, in the union bound, it is possible to choose $p(f)$ to assign different weights to functions (see the inequality 7). Therefore, if the $p(f)$ is well-chosen with the help of some prior domain knowledge, the bound will have small values (Bousquet et al., 2004). If the upper bound is utilized for a infinite and uncountable hypothesis space as discussed previously, domain knowledge provides auxiliary information to estimate more precise upper bound than one only based on the given data (see the inequality 8). Yaser Abu-Mostafa (Abu-Mostafa, 1995) gave two general approaches to reduce generalization error by estimating new VC dimensions based on the data and extra information: $VC(H|K)$ and $VC(H;K)$. The former one, $VC(H|K)$, is defined as the VC dimension of a hypothesis space in which any hypothesis $h \in H$ satisfies the given domain knowledge K. Set H' represents the new hypothesis space $H' = \{h \in H \wedge h \Vdash K\}$ and, for instance, the domain knowledge K can be an invariance hint[1]. Since $H' \subseteq H$, it is clear that $VC(H|K) = VC(H') \leq VC(H)$. For example, in the Figure 4.2, suppose the domain knowledge shows the parameters of an ANN have properties $k_1 : \{w_{11} = -w_{12},\ w_{21} = -w_{22},\ ...,\ w_{81} = -w_{82},\ t_1 = t_2,\ w_1 = w_2\}$. The new VC dimension of the ANN, $VC(H|k_1) \approx 8 + 1 + 1 + 1$, is then less than the standard one (Abu-Mostafa, 1995). The later one $VC(H; K)$ arises when the virtual examples are introduced as a proxy of the domain knowledge. By including extra artificial examples, the number of total training examples increases, and $VC(H; K)$ has a upper bound $5VC(H)$ in case of the invariant hints (Fyfe, 1992). With the increase of the number n of the training examples, a part $\frac{h(\log \frac{2en}{h}+1)-\log(\delta/2)}{n}$ of the upper bound decreases, and then the upper bound becomes more precise (see the inequality 8).

These discussed improvements to the upper bound provide only a guideline to incorporating auxiliary information thus reducing generalization errors. In real-world cases, the effects vary depending on the different practical situations and learning algorithms.

4.3 Convergence with Domain Knowledge

It is possible that domain knowledge enhances *convergence* to an optimal hypothesis $h(x) \in H$ which is equivalent to or the best approximation of the unknown target hypothesis $c(x)$.

The roles of domain knowledge in the convergence address various characteristics such as feasibility, efficiency, and accuracy. The feasibility indicates whether the hypothesis space of the given learning algorithm can output a hypothesis identical to the target model, and if that replica (or approximation) of the target model is acceptable. In other words, the capacity of the hypothesis space determines the feasibility and also is addressed in the variance/bias tradeoff by the generalization. If the given domain knowledge already indicates that the capacity of the given hypothesis space does not reach certain requirements, the learning algorithm already fails at the first place. No matter how many efforts the learning system puts into the convergence, a satisfying output will not be reached.

The efficiency of convergence is closely related to the complexity of the hypothesis space, and directly determines the efficiency of the given learning algorithm. With the help of certain domain knowledge, some learning algorithms can initiate smaller and more precious hypothesis spaces. Others can reduce the size of the hypothesis space, or select shorter (or more direct) search paths, whereby convergence takes less computation time, and cost. There is, however, the possibility that when the domain knowledge introduces the auxiliary information, it may increase the complexity of the learning process, thus causing the convergence to be much slower, even being a NP-hard problem[2]. For example, if a prediction process is represented as an optimization process and the domain knowledge requires the result of the prediction process will be integers, the prediction problem falls into a integer programming. The integer programming is a typical NP-complete problem. Thus domain knowledge as a additional constraint increases the complexity of the prediction problem, but in terms of the approximation to prosperity of unknown function $c(x)$, this

[1] This hint asserts that $c(x) = c(x')$ for certain pairs x, x'. If this hint is violated, the associated error is $R_{knowledge} = (f(x) - f(x'))^2$. This hint will be discussed in detail in the following section, "Invariance".

[2] In complexity theory, NP ("Non-deterministic Polynomial time") is the set of decision problems solvable in polynomial time on a non-deterministic Turing machine. The NP-complete problems are the most difficult problems in NP in the sense that they are the ones most likely not to be in polynomial time (Aho, Hopcroft, & Ullman, 1974).

domain knowledge is of use. In this case, the user has to make a trade-off between accuracy and cost.

Domain knowledge is valuable tool in setting stop criterions of convergence that also affects the accuracy of the convergence. This indicates how effective the learned model (or selected function) is a replica of the target model. In many practical learning problems, it is not economical to reach the best optimal solution by using too many resources, e.g. computation time. There is a balance between cost and accuracy, and domain knowledge is valuable to set a set of stop criterions to reach a cost-effective balance.

4.4 Summary

Along with the conceptual benefits of prior domain knowledge, there are two key practical benefits namely: 1) eliminating noise from training example, and 2) increasing the transparency of both the resultant models and the learning process. The theoretic discussion relies on the assumption, that is there is no noise in the training examples. In the majority of real-world cases, this assumption is not valid, and many learning systems suffer from noisy training examples. In the preprocessing of training examples, the domain knowledge plays a crucial role in reducing the noise.

In many real-world cases, the opaque models, which the majority of complex learning systems result in, are unacceptable. This is because it is very difficult for naive users to accept an answer without a valid reason. This issue is often ignored by academic researchers in the machine learning community, causing an obstacle when complex learning system is implemented in industry. In industries, some users prefer comprehensible simple models at the sacrifice of accuracy. This issue is addressed in the following sections using semi-parametric modeling. It produces a semi-transparent model, which reaches a balance between transparency and accuracy.

Along with the benefits, it is essential to discuss the risk of incorporating domain knowledge. In the previous section of convergence with domain knowledge, one risk has already been addressed: additional domain knowledge causes intractable computation complexity. This issue assumes that domain knowledge is perfect, in that it does not contain any noise and completely covers the whole task. However, the same as the imperfect training examples, imperfect domain knowledge is common in real-world practice. In order to minimize negative impacts from imperfect domain knowledge, a trade-off is necessary. As previously discussed, one approach is the regularizer and employs a new risk function $w_S R_{emp}(f) + w_K R_{knowledge}(f)$, where the coefficients w_S and w_K assign the different weights to the empirical risk function $R_{emp}(f)$ and domain knowledge risk function $R_{knowledge}(f)$ (see Definition 2 in the section "Consistency with Domain Knowledge").

Concerns about the consistency, generalization and convergence provide a guideline to the following methods of incorporating various prior domain knowledge into inductive machine learning algorithms. In the following parts of this thesis, all of the reviewed or proposed methods consider both this guideline and other relevant practical issues.

There are always a number of desirable properties sought after when forming a learning algorithm. They include:

- Given no prior domain knowledge, a machine learning algorithm should learn at least as effectively as purely inductive methods.

- Given a perfect domain knowledge, a machine learning algorithm should learn at least as effectively as purely analytical methods

- Given an imperfect domain knowledge and imperfect training data, a machine learning algorithm should combine the two to outperform either purely inductive or purely analytical methods.

- A machine learning algorithm should be tolerant some unknown level of error in the training data.

- A machine learning algorithm should be tolerant some unknown level of error in the domain knowledge (Mitchell, 1997b).

Based on the essentiality of the domain knowledge in machine learning algorithms, some machine learning algorithms require the domain knowledge as a condition of learning. Others treat the domain knowledge as a option. Therefore, domain knowledge is classified as "*required*" and "*optional*" domain knowledge with respect to learning systems.

Based on the enforcement of the domain knowledge, some domain knowledge is called "*hard domain knowledge*" or "strict enforcement". Learning machines must find "best" feasible hypothesis being consistent with hard domain knowledge. Other domain knowledge which learning machines find "best" feasible hypothesis maximally respecting is called "*soft domain knowledge*" or "partially enforcement" (Davidson & Ravi, 2005).

The next section reviews and modifies Tom Mitchell's categories of incorporating domain knowledge into machine learning, and includes recent research results.

5. Four Categories of Methods of Incorporating Prior Domain Knowledge into Inductive Machine Learning

This section summarizes existing methods produced by some academic researchers' works on this topic. All of methods are classified into four categories along the learning process of a typical learning system: 1) Using prior domain knowledge to prepare training examples; 2) Using prior knowledge to initiate the hypothesis or hypothesis space; 3) Using prior domain knowledge to alter the search objective; and 4) Using Domain Knowledge to Augment Search. The last section discusses the Semi-parametric Models and Hierarchal Models, which are two special types of methods in the second category, i.e. using prior knowledge to initiate the hypothesis or hypothesis space. In the following parts of this paper, these two methods contain an emerging learning algorithm, kernel methods, to produce a new learning algorithm.

5.1 Using prior knowledge to prepare training examples

The first step of the learning process of an inductive machine learning system is to preprocess the training data. This step consists of selecting, cleaning and transforming the original data.

The majority of learning algorithms assumes that the collected training examples do not contain noises and the features have no redundance. In practice, this assumption is not always established. In many cases, the noise introduced by the examples or redundant

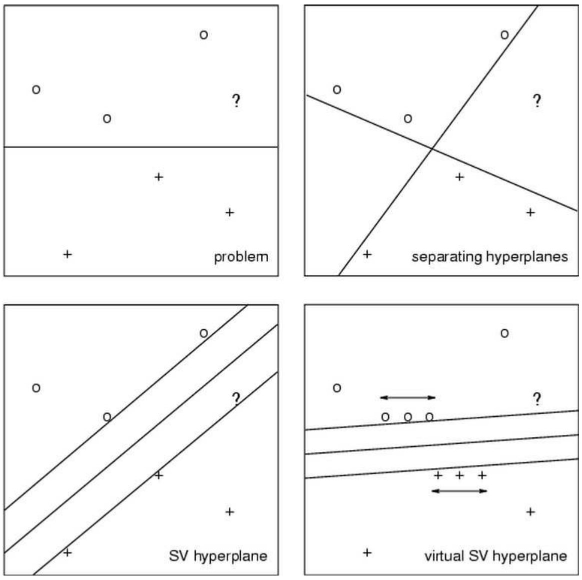

Figure 6: Use virtual samples to bias the model produced by Support Vector Machine (Burges & Scholkopf, 1997) (Scholkopf et al., 1999b). The additional virtual samples force the resulting hyper-plane more horizontal than one produced by the original training samples at the third figure.

features is the major cause of poor performance in a learning system. Prior domain knowledge provides important information to eliminate noise within the features and examples for selection and sanitizing. Without sufficient domain knowledge, one does not always know which indicators are relevant to the movement of a particular response. Often one has to prepare different sets of indicators and mine them (Tsang, Yung, & Li, 2004).

Before feeding data into any learning system, one often needs to experiment with different ways to transform original data into new features, for example a ratio of two features. One needs to remove the redundant features in order to reduce the complexity. The most obvious observations to remove is the noise thereby reducing noise and complexity. Some training examples have their internal structures, such as a tree or a network, and these structures need to be transformed by domain knowledge in order to be incorporated into the learning system. This research will use prior domain knowledge to guide the process of feature selections, in the form of must-link (two features must be in the same cluster) or cannot-link (two features must not be in the same cluster) .

Being different from the regular data preprocess, this section summarizes some researches of incorporating prior domain knowledge into inductive machine learning through data preparation. Among them, using virtual data samples to influence the operation of search has gained much attention in recent years. Partha Niyogi et al proposed an idea of virtual examples as a possible strategy for incorporating prior information in the neural network in order to counter the problem of insufficient training examples (Niyogi, Girosi, &

Poggio, 1998). This showed that the idea of virtual examples was mathematically equivalent to incorporating prior domain knowledge as a regularizer in the function learning in certain restricted domains.

Similar to Niyogi's work, Bernhard Scholkopf et al pointed out that invariance could be included in pattern recognition in a principled way via the virtual support vector mechanism and restriction of the feature space (see Figure 6) (Burges & Scholkopf, 1997) (Scholkopf et al., 1999b). Dennis Decoste et al inducted the invariant support vector machine (SVM), which used prior knowledge as an invariance to bias SVM by creating a series of virtual samples (Decoste & Scholkopf, 2002). Their method is composed of the below procedures:

1. train a Support Vector Machine to extract the support vector set;
2. generate artificial examples, termed virtual support vectors, by applying the desired invariance transformations to the support vectors; and
3. train another support vector machine on the generated examples.

As the domain knowledge is encapsulated within a set of virtual examples, the domain knowledge risk function is represented by an empirial risk of the virtual examples $R_{knowledge}(f(X_v)) = R_{emp}(f(X_v))$, where X_v is the set of virtual examples.

The virtual training example method is an indirect way of incorporating prior domain knowledge in a functional form. The strength of this method is that it is a universal solution, so long as the given prior domain knowledge is able to be converted into a set of artificial training data. In other words, this method is feasible to any inductive learning system, which relies on the training data. But this method also shares some of the drawbacks of functional knowledge transfer, which requires more storage space and less computational efficiency in case of a large volume of data.

5.2 Using prior knowledge to initiate the hypothesis or hypothesis space

There are two types of approaches for using domain knowledge to initiate hypothesis space or the hypothesis. The first approach is to initiate a hypothesis space by satisfying both training examples and domain knowledge. Inductive learning methods search out a hypothesis through this more appropriate hypothesis space. Because the hypothesis space is initiated partially or completely by the domain knowledge, the hypotheses in this space is also consistent with the domain knowledge. The second approach is to initialize the hypothesis either partially or completely to fit the existing domain theory. Inductive learning methods will be used to refine or complete the hypothesis by fitting the training data. In a hypothesis space, this approach provides a better starting point for convergence. A better starting point for convergence results in a closer replica to the unknown target model; therefore, the path of convergence is shorter and the convergence is more efficient.

Due to the characters of kernel methods, the selections of their kernel function relies heavily on the domain theory. Without any prior domain knowledge, it is common to choose the radial basis functions (RBF) or polynomial functions as kernel functions. However, in many cases, it is better to choose or create a special kernel function for particular requirements in certain domains. Bernhard Scholkopf et al explored kernel designing, series of methods for incorporating prior knowledge in constructing appropriate kernel functions

(Scholkopf et al., 1999b). Therefore, the kernels selected or constructed by the domain knowledge initiate a proper hypothesis space in which learning algorithms converge to a better approximation of the unknown target model.

Regular Bayesian Networks (BN) utilize existing domain knowledge to initiate the structure of the network. The learning algorithms then parameterize the network. In the structural learning of the Bayesian Network, Helge Langseth and Thomas D. Nielsen used Object Oriented techniques to construct small and "easy-to-read" pieces as building blocks for creating a more complex BN (Langseth & Nielsen, 2003). Beyond the simple sub-sections of BN, the OOBNs (Object Oriented Bayesian Networks) introduce a class hierarchy in which the sub-superclass relationship contains the domain knowledge about the structure of resultant BN. For example, consider a farm with two dairy cows and two meat cows. Assume that the task is to model the environment's effect on the dairy and meat production of these cows. The OOBNs construct a generic cow class (super class) that describes the general properties common to all cows, and dairy cow and meat cow that become two subclasses of that superclass. The subclasses inherit properties of their superclass, and the OOBN framework facilitates the structural learning of a Bayesian Network from training samples, simplifying the structure of the learned network.

In the Knowledge-Based Artificial Neural Network (KBANN), a modified artificial neural network, an initial network is first constructed so that the classification assigned to every possible instance by the network is identical to that assigned by the domain theory (Towell & Shavlik, 1989). The Backpropagation algorithm is then employed to adjust the weights of this initial network as needed to fit the training examples. An assumption behind the KBANN is that the domain theory is correct or approximately correct. Given a better starting approximation than pure artificial neural network, the KBANN should learn better generalization accuracy for the final hypothesis. However, the KBANN can be misled given incorrect or insufficient prior knowledge (Mitchell, 1997b). In order to solve this problem, the Explanation-Based Neural Network Learning (EBNN) algorithm did not completely rely on the users to provide prior knowledge, but calculated the prior parameters by explaining each training example in terms of a given domain theory (Mitchell, 1993) (Thrun, 1996). Similarly, Tony Jan et al used prior domain knowledge to build the structure of a hypothesis and the neural learning refined the model (Jan, Yu, Debenham, & Simoff, 2004).

Patrice Simard et al and Partha Niyogi et al addressed the invariance in the distance-based classification algorithms, for example K nearest neighbors (Simard, Cun, & Denker, 1993)(Niyogi et al., 1998). Suppose we know a transformation s (such as rotation) such that if P is a valid example then $s(P)$ is also a valid example. The transformation s depends on one parameter α, and the set of all transformed patterns $S_P = \{x|\exists\alpha \Rightarrow x = s(\alpha, P)\}$ is an one-dimensional curve in the vector space of the inputs. When the set of transformations is parameterized by n parameters α_i, S_P is a manifold of at most n of P. The part of S_P that is close to P can be approximated by a plane tangent to the manifold S_P at the point P. The regular Euclidean distance between two patterns x and z is replaced by a new definition of distance, the distance between two manifolds $s(x)$ and $s(z)$, called *transformation distance* (Simard et al., 1993). The known invariance transformation could depend upon the prior knowledge of a particular problem. For instance, in character recognition, a small rotation and various thickness of a letter are typical invariances (see

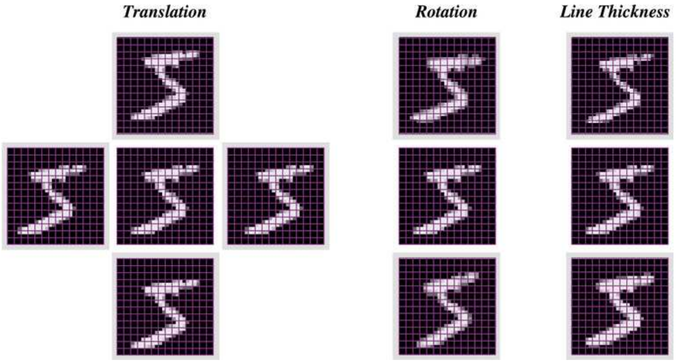

Figure 7: Different Invariance transformation (Decoste & Scholkopf, 2002)

Figure 7). By using the transformation distance instead of the regular Euclidean distance, the hypothesis space is initiated to tolerate this invariance.

The search objectives themselves are optimized by domain knowledge. For example, the semantic query optimization introduced by Suk-Chung Yon et al can be regarded as the process of transforming a query into an equivalent form (Yoon, Henschen, Park, & Makki, 1999). This form can be evaluated efficiently by narrowing the search space and refining the query. Suppose we consider the following domain knowledge: all ships whose deadweight is greater than 700 tons travel at a speed greater than 60 mph" and "the ships whose deadweight is greater than 700 tons are supertankers". According to the domain knowledge, only ships whose type is supertankers in the index of the shiptype need to be considered instead of the speed and the deadweight.

5.3 Using prior domain knowledge to alter the search objective

In this set of approaches, the domain knowledge acts as an inductive bias in the process of searching out an optimal hypothesis consistent with both the training data and domain knowledge. Most learning algorithms convert their machine learning problems into optimization problems by constructing objective functions with sets of constraints. The problem of minimizing the empirical risk function is an example of the objective function, $min[R_{emp}(f)] = min[\sum_{i=1}^{m}(y_i - f(x_i))^2]$. As the empirical risk function decreases, the learning algorithm converges to a hypothesis which is the best approximation to the underlying desired model within the given hypothesis space H. Here, the prior domain knowledge either acts as an additional regularizer within the objective function or set of additional constraints. The majority of existing research on incorporating domain knowledge into inductive machine learning occurs in this area, containing a diverse range of results.

There are two major types of methods using prior domain knowledge to alter the search objectives:

- The goal criterion is modified to require the output hypothesis to fit the domain knowledge as well as the training data; for example, learning with constraints, learning with weighted examples, and cost-sensitive learning.

- Domain knowledge is employed to verify the search result to guide the search process in the right direction; for example, knowledge-guided learning proposed by Stuart Russell. Russell's method generates all possible hypotheses expressible in terms of primitive language and tests them for consistency with its prior knowledge (Russell, 1991).

5.4 Learning with Constraints

In practical problems, the hypothesis space can be vast. Therefore, it is necessary to search the space preferentially and prune subspaces based on certain domain knowledge. This is referred to as learning with constraints or constrained learning, as addressed by Ian Davidson (Davidson & Ravi, 2005). In learning with constraints, domain knowledge is expressed as a set of additional constraints within the objective function or constraint.

In handwritten characters recognition, certain derivatives of the target function can be specified in prior since some patterns of letters are available. This approach has been explored by Simard et al. in TangentProp (Simard, Victorri, Cun, & Denker, 1992). In this approach, an additional term is added into the error function of normal neural learning to augument the search objectives. Thus, the Tangent Prop extends the backpropagation algorithm, allowing it to learn directional derivatives. The usual weight-update rule of the backpropagation algorithm is modified: $\Delta w = -\eta \frac{\partial E}{\partial w}$ is replaced with $\Delta w = -\eta \frac{\partial}{\partial w}(E + \mu E_r)$, and the E_r measures the discrepancy between the actual and desired directional derivatives.

In the previous introduction to the kernel methods, the parameters $\{w, d\}$ in the primal or $\{\alpha, d\}$ in the dual formula are solved as an optimization problem, such as linear or quadratic programming. The basic idea of the Knowledge-based Support Vector Machine (KSVM) as proposed by Glenn M. Fung and Olvi L. Mangasarian is to include domain knowledge as extra constraints in the given linear or quadratic programming (Fung, Mangasarian, & Shavlik, 2001) (Mangasarian, Shavlik, & Wild, 2004). In order to narrow the hypothesis space, KSVM utilizes a set of linear equalities or inequalities to represent prior domain knowledge in the properties of the target functions.

5.4.1 Hints

Abu-Mostafa defined *Hints* as auxiliary information about the target function that can be used to guide the learning process (Abu-Mostafa, 1995), when the regular learning process tries to recreate a target function using a set of input-output examples. He listed some common types of hints:

- *Invariance hint*

  This hint asserts that $c(x) = c(x')$ for certain pairs x, x'. If this hint is violated, the associated error is $R_{knowledge} = (f(x) - f(x'))^2$. This hint will be discussed in detail in the following section, "Invariance".

- *Monotonicity hint*

  This hint asserts for certain pairs x, x' that $f(x) \leq f(x')$. For instance, "f is monotonically nondecreasing in x" is formalized by all pairs x, x' such that $f(x) \leq f(x')$.

If this hint is violated, the associated error is

$$R_{knowledge} = \begin{cases} (f(x) - f(x'))^2, & \text{if } f(x) > f(x') \\ 0, & \text{if } f(x) \leq f(x') \end{cases} \tag{18}$$

Joseph Sill el at incorporated the monotonicity hints into the backpropagation algorithm for credit card fraud detection (Sill & Abu-Mostafa, 1997). The resultant improvement from the introduction of monotonicity hints is statistically significant, nearly 2%.

- *Examples hint*

  Given $(x_1, c(x_1)), \ldots, (x_N, c(x_N))$, the examples hint asserts that these are the correct values of c at the particular points within $x_1, \ldots, x_N$. Say that the correct subset consists of n examples: $(x_1, c(x_1)), \ldots, (x_n, c(x_n))$, and then if this hint is violated, the associated error is $R_{knowledge} = (f(x_n) - c(x_n))^2$.

- *Approximation Hint*

  The hint asserts for certain points $x \in X$ that $c(x) \in [a_x, b_x]$. In other words, the value of c at x is only known approximately. If this hint is violated, the associated error is:

$$R_{knowledge} = \begin{cases} (f(x) - a_x)^2, & \text{if } f(x) < a_x \\ (f(x) - b_x)^2, & \text{if } f(x) > b_x \\ 0, & \text{if } f(x) \in [a_x, b_b] \end{cases} \tag{19}$$

- *Consistency Hint*

  The consistency hint differs from previous hints, in that it is more subtle. Including a consistency hint forces the learning algorithms to be consistent with their theoretic assumptions. The parametric models always make the assumption that the model is based on. For example, the discrete time version of stochastic Vasicek model $\Delta x_n[l] = k_n(\theta_n - x_n[l]\Delta t[l] + \sigma_n w_n[l]\sqrt{\Delta t[l]})$ has assumptions that $w_n[l]$ follows a gaussian distribution. However, within the learning process, there is no discrepancy metric between the learned distribution and theoretic assumptions. Mostafa employed the Kullback-Leibler distance $K(p\|q)$ to quantify the agreement/disagreement between the two distribution:

$$K(p\|q) = \int p(u) log \frac{p(u)}{q(u)} du \tag{20}$$

  here $p(u)$ is the pdf of the learned $w[l]$, $q(u)$ is the pdf of the theoretical $w[l]$ (Abu-Mostafa, 2001). Since the form of $p(u)$ is unknown, the maximum-entropy principle is used to estimate it. Therefore the risk function of this consistency hint is expressed as $R_{knowledge} = K(p\|q)$, and if and only if $p = q$ the $R_{knowledge} = K(p\|q) = 0$. It is simple to inject this risk function into the original objective function.

- *Catalytic Hint*

  The catalytic hint is also quite different from the previous hints. The major difference comes from the way in which the catalytic hint is incorporated. As introduced by

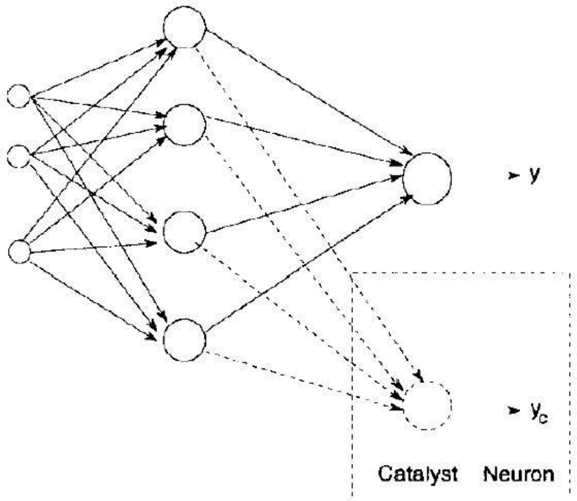

Figure 8: Catalytic Hints (Jin & Sendhoff, 1999)

Steven Suddarth and Alistair Holden (Suddarth & Holden, 1991), the catalytic hint was first realized in the feed-forward neural network and later implemented by Yaser Abu-Mostafa (Abu-Mostafa, 1995) and Yaochu Jin (Jin & Sendhoff, 1999) respectively. They introduced additional output nodes, catalytic neurons (see Figure 8), which are only included by artificial neural networks during the training process. The additional catalytic nodes force the neural network to learn extra functions simultaneously with the target function. In practice, the extra functions are closely related to the target function. By using the catalytic hints, users emphasize in some parts of the target function.

When hints are available in a learning situation, the objective function, as optimized by the learning algorithms, is no longer confined only to the empirical risk R_{emp} (Abu-Mostafa, 1995). The associated domain knowledge risk function $R_{knowledge}$ with hints is then included as extra components in the objective function to contain the information with hints. Further, the "hints" approach could be treated as a special case in the regularization theory. Here an ill-posed problem is transformed into a well-posed one by using prior domain knowledge. The most common form of prior knowledge is smoothness, but others include monotonicity, convexity and positivity (Niyogi et al., 1998). In terms of the optimization theory, these approaches either modifies the objective function, or introduces extra constraints into an optimization problem.

Invariance In recent years, the machine learning community put a large amount of emphasis on invariance. The invariance means the information of certain transformations of the input which are known to leave the function values unchanged. For example, in the figure 7, artificial data is generated by rotating a digital image of the digit "5". An invariant machine learning algorithm has the capability of recognizing this artificial data as the digital image of the digit "5" instead of other digits. By revisiting the previous definition of "Invariance Hints", a certain pair x, x' stands for the digital image of the digit "5" and the rotated image of 3 degrees. Because the two images are the same $c(x) = c(x')$, the correct learned hypothesis then produces: $f(x) = f(x')$ and the associated error is $e_m = (f(x) - f(x'))^2 = 0$.

The virtual example method is one of approaches to incorporating invariance into regular inductive machine learning. However it has two major drawbacks: 1) the users must choose the magnitude of distortion and how many virtual examples should be generated; and 2) more importantly, the virtual example is highly correlated with the original data. This makes some of learning algorithms, such as ANN, very inefficient (Simard et al., 1992).

Bernhard Scholkopf et al summarizes three types of methods for incorporating invariance into kernel methods (Scholkopf & Smola, 2002b). First there are the virtual support vector method, as discussed in the previous section; then Jittered SV method, as will be discussed in the following section. Finally the invariance hyperplane method can be seen as a extension of previous hints method. The basic idea is to introduce an additional regularizer into the regular objective function.

5.5 Learning with Weighted Examples:

As it is relatively straightforward to add extra constraints in an optimization problem, learning with constraints has already produced some relatively mature results. Learning with weighted examples is another often-used method of manipulating the objective function to contain extra information.

Xiaoyun Wu and Rohini Srihari developed a new derivative of the SVM, the Weighted Margin Support Vector Machine (WMSVM). They used prior domain knowledge to generate "Pseudo Training Dataset" as a part of training sample input into the WMSVM. The objective function of primal problem of the WMSVM is given:

$$\langle w \cdot w \rangle + C \sum_{i=1}^{m} \xi_i + \eta C \sum_{i=m+1}^{n} g(v_i) \xi_i \tag{21}$$

The parameter η is used to control the relative importance of the evidence from these two different datasets, that are the true training examples and Pseudo Training Dataset. One wants to have a relatively bigger η when the quality of true training examples is poor. Wu and Srihari carried two experiments of text categorization, and compared to the results from a standard SVM, LibSVM (Chang & Lin, 2001), WMSVM with prior knowledge achieved higher accuracy with much less training samples (Wu & Srihari, 2004).

Considering the importance of order in the time-series analysis, L. J. Cao et al incorporated time orders as a parameter of regularized risk function of SVM to put heavier weights to the recent training samples than those of the distant training data points (Cao & Tay, 2003).

5.6 Cost-sensitive Learning

Cost-sensitive classification problems are characterized by different costs for different classification errors (Giorgio Fumera, 2002). Let $c(i, j)$ be the cost of deciding for class i when the true class is j. The empirical risk is rewritten as:

$$R_{emp} = \sum_{j=0}^{m} c(i,j) \tag{22}$$

Cost-sensitive learning is very close to the learning with weighted examples, but differs in that it is restricted within the classification problems as cost-sensitive regression has intractable computation cost. Giorgio Fumera et al implemented cost-sensitive learning into a binary SVM, and the empirical function becomes: suppose the decision function is defined:

$$\begin{cases} f(x,\alpha) = +1, & \text{if } w \cdot x + b \geq \varepsilon \\ f(x,\alpha) = 0, & \text{if } -\varepsilon < w \cdot x + b < \varepsilon \\ f(x,\alpha) = -1, & \text{if } w \cdot x + b \geq -\varepsilon \end{cases} \tag{23}$$

and the empirical risk function will be:

$$R_{emp} = \begin{cases} 0, & \text{if } f(x,\alpha) = y \\ c_R, & \text{if } f(x,\alpha) = 0 \\ 1, & \text{if } f(x,\alpha) \neq y \text{ and } f(x,\alpha) \neq 0 \end{cases} \tag{24}$$

where ε delimits a "reject" area along the hyperplane. The examples located within the reject area will be handled with different procedures (Giorgio Fumera, 2002).

5.7 Using Domain Knowledge to Augment Search

Using domain knowledge to augment search is similar to using prior domain knowledge to alter the search objective. The difference is that the methods of using domain knowledge to augment search produce new hypothesis candidates in the process of searching (or convergence). In other words, the hypothesis space H is adjusted by the domain knowledge with the on-going searching. But the methods of using prior domain knowledge to alter the search objective work within a fixed hypothesis space and the domain knowledge only eliminates parts of the hypothesis space.

Pazzani et al (Pazzani, Brunk, & Silverstein, 1991) developed the FOCL as an extension of the purely inductive FOIL system. FOIL generates hypotheses purely from training data. FOCL also use domain knowledge to generate additional specifications, but it then selects one hypothesis among all of these candidate specifications based on their performance over the data. Therefore, in the FOCL method, imperfect domain knowledge only impacts the hypotheses if the evidence in the data sets supports the knowledge.

In kernel methods, Dennis Decoste et al introduces the idea of "kernel jittering" (Decoste & Scholkopf, 2002). Prior domain knowledge is included by applying transformations to patterns as a part of kernel definition. In SVM, this method replaces the regular kernel function, $K(x_i, x_j) \equiv K_{i,j}$ as a jittering kernel form $K^J(x_i, x_j) \equiv K^J_{i,j}$, defined procedurally as follows:

1. Consider all jittered forms (see Figure 7) of example x_i (including itself) in turn, and select the one (x_q) "closest" to x_j; specifically, select x_q to minimize the metric distance between x_q and x_j in the space induced by the kernel. The distance is given by: $\sqrt{K_{qq} - 2K_{qj} + K_{jj}}$.

2. let $K^J_{ij} = K_{qj}$

According to their experiment which compares the performance between the virtual SV (VSV) and the kernel jitter, the training set of the kernel jitter can be smaller than the corresponding VSV methods, but its accuracy was lower than the VSV methods.

A limited amount of research has been done in this category, as it is difficult to simultaneously adjust both the process of convergence and the hypothesis space. The majority of inductive learning algorithms do both separately.

6. Semi-parametric Hierarchal Modelling

Traditional statistics methods which make assumptions regarding underlying distributions are named *parametric methods*. The term "parametric" indicates that the structure of the resultant model is known, and the learning methods only estimate the parameters of the resultant models. In contrast, most machine learning methods exist as *non-parametric methods*, which simply use very complex structures and do not make any assumption of the underlying distribution, or the structures of the resultant models. In theory, they provide universal approaches independent from the domain, but in reality they introduce higher computation and complexity without the transparent structures of resultant models. For example, kernel methods use a set of basis functions, to approximate the underlying distribution. The cardinal number of the set of basis functions may reach infinity.

If the structure of the resultant model is known beforehand, the parametric method is preferred. This is because the structure of resultant model is more comprehensible and its computation is more efficient than non-parametric methods. In real world examples, some domains have been explored and some parametric models exist but often they are incomplete. One solution is to apply parametric knowledge to the problem as much as possible in order to represent the known knowledge. The nonparametric techniques would then address the remainder of the question. This kind of hybrid model is called as semi-parametric model (or hierarchical models) and depends on the structure of its resultant models.

A semi-parametric model consists of the parametric components and nonparametric components. A semi-parametric model contains the following structure: $f(x) = g(x_m) + \sum_{i=1}^{n} \beta_i x_i$, where $\{x_m, x_n\} \subseteq x$, $g(x_m)$ is the nonparametric component, and $\sum_{i=1}^{n} \beta_i x_i$ is a linear parametric component with $x_i \in x_n$. Semi-parametric models are easy to understand (due to the parametric part), and perform well (often thanks to the nonparametric term) (Scholkopf & Smola, 2002a). When the nonparametric component is a support vector regression (SVR), a semi-parametric SV regression introduces additional components within the constraints when the setting is translated into optimization problems. In terms of the capacity of the function class, kernel mapping in nonparametric components produces much higher dimensions than that of parametric components. Thus even if the parametric components are not regularized at all, the overall capacity still works (Scholkopf & Smola, 2002a). In a similar research in the SVM, Davide Mattera and Francesco Palmieri introduced the semi-parametric SVM as a general structure of nonlinear functions, $f(x) = w^T \phi(x) + w_1^T \psi(x)$, in which $w_1^T \psi(x)$ embedded prior known parametric function (Mattera, Palmieri, & Haykin, 2001). These two research methods achieved the same result in two different ways.

Hierarchical models address the fundamental assumptions in the majority of machine learning algorithms, as with identical underlying distribution. In many cases, it is certain that data is not produced by identical distribution, but by related data sets. Rather than using a single model to cover all data sets, it is more sensible to build a set of different local models, and then use a global model to unify them. The hierarchical models constructed by nonparametric and parametric components belong to semi-parametric models.

In some literature, the hierarchical Bayesian modelling allows the information between different models to be transferred via common hyperparameters. For example, they assume that M data sets $\{D_j\}_{j=1}^{M}$ are available for related but not identical settings and they have trained M different models with parameters $\{\theta_j\}_{j=1}^{M}$ on those data sets (Tresp & Yu, 2004). If a new model concerns a related problem, then it makes sense to select new hyperparameters h_{hb} such that $P(\theta|h_{hb})$ approximates the empirical distribution given by the maximum likelihood parameter estimate instead of using the original informed prior $P(\theta|h_{prior})$. In this way the new model can inherit knowledge acquired not only from its own data set, but also from the other models: θ_{M+1}: $P(\theta_{M+1}|\{D_j\}_{j=1}^{M}) \approx P(\theta_{M+1}|h_{hb})$. In order to realize this inheritance, the local models must share the same or similar parametric structure. Otherwise the inheritant is difficult.

The hierarchical models allow the "knowledge transfer" via common hyperparameters within their framework. This is a very attractive characteristic in that each local model represents one subset of data produced by an identical distribution. The global model then works like a collaborative filter to transfer knowledge amongst the local models. Clearly, it is closely related to the previous multi-task learning.

The way of partition is represented by a partition function (or switch function) which assigns observations to local models. The partition function may employ different subsets of training data from those used by local models. Either the model selection or the partition of the original observation requires the presence of domain knowledge in the format of parametric local models or the information of partitions. The local models are then unified by the global nonparametric model.

6.1 Vector Quantization

Vector quantization (VQ) is a lossy data compression method based on the principle of block coding. According to Shannon's theory of data compression, in the lossy data compression, better known as rate-distortion theory, the decompressed data does not have to be exactly the same as the original data. Instead, some amounts of distortion D are tolerated, and in contrast the lossless compression has no distortion, $D = 0$.

Linde, Buzo, and Gray proposed a VQ design algorithm based on a training sequence. The use of a training sequence bypasses the need for multi-dimensional integration required by previous VQ methods (Linde, Buzo, & Gray, 1980). A VQ that is designed using this algorithm is referred to in relevant literature as an LBG-VQ (see Figure 6.1). Given a vector source with its statistical properties known, a distortion measure, and the number of codevectors, the LBG-VQ finds a codebook (the set of all red stars) and a partition (the set of blue lines) which result in the smallest average distortion (Gersho & Gray, 1992).

Consider a training sequence consisting of M vectors: $T = \{x_1, x_2, ..., x_M\}$, and N codevectors $C = \{c_1, c_2, ..., c_N\}$. The whole region is partitioned by the codevectors into a

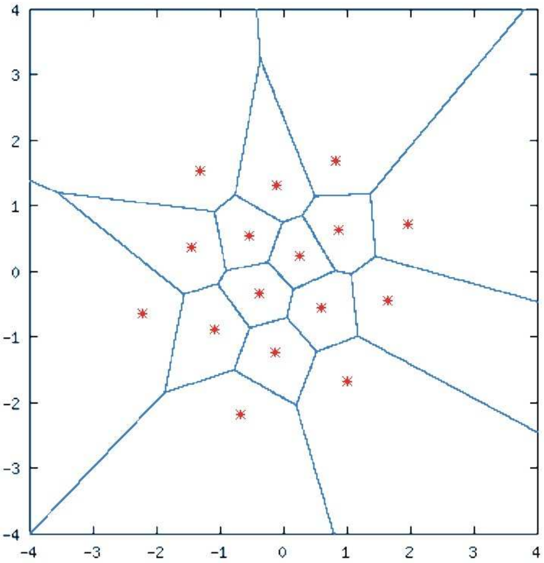

Figure 9: A Simple Example of Two-dimensional LBG-VQ (Gersho & Gray, 1992)

set of sub-regions, called Voronoi Regions $P = \{S_1, S_2, ..., S_N\}$. Vectors within a region S_n are represented by their codevector $Q(x_m) = c_n$ as $x_m \in S_n$, and the average distortion can be given by: $D_{ave} = \frac{1}{Mk}\sum_{m=1}^{M} \|x_m - Q(x_m)\|$, which measures the information loss. Thus, the design problem can be summarized as: $argmin_{C,P}(D)$ that is to local C and P such that D is minimized,.

If C and P are a solution to the above minimization problem. First, it must satisfy two criteria: the nearest neighbor condition, $S_n = x : \|x - c_n\|^2 \leq \|x - c_{n'}\|^2, \forall n' \in \{1, 2, ..., N\}$. Second, the centroid condition, $c_n = \frac{\sum_{x_m \in S_n} x_m}{\sum_{x_m \in S_n} 1}, \forall n \in \{1, 2, .., N\}$. According to the defined framework of the semi-parametric hierarchical modelling, the LBG VQ acts as a partition function, the local model represents a subregion S_n by its codevector $Q(x_m) = c_n$, and the global model is represented by a support vector machine.

The LBG VQ design algorithm is iterative, which alternatively solves the above two optimization criteria. The algorithm requires an initial codebook, that is obtained by a splitting method. The initial codevector is set as the average of the entire training sequence, and then split into two with the iterative algorithm running with these two vectors as the initial codebook. The final two codevectors are split into four, four into eight, eight into sixteen, and the process is repeated until the desired number of codevectors is obtained (Gersho & Gray, 1992).

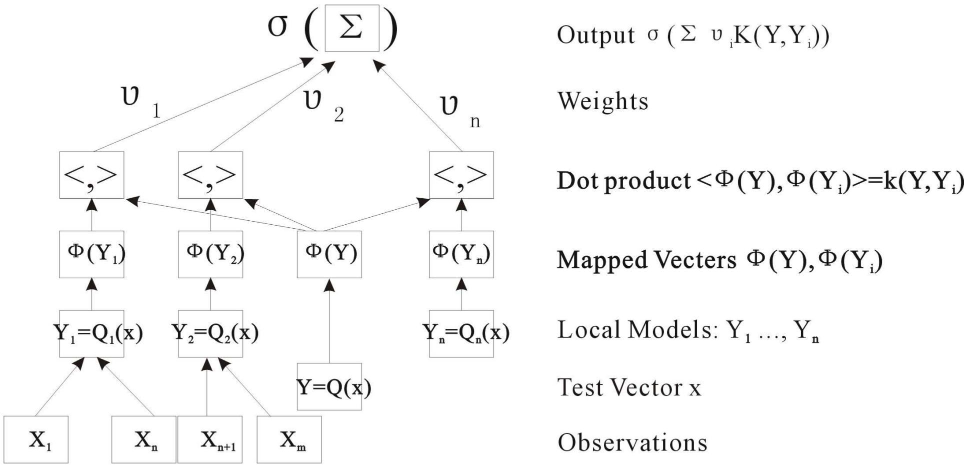

Figure 10: Hierarchical Modelling with SV Machines, which modifies the original SVM (see figure 3)

6.2 Semi-parametric Hierarchical Modelling and VQ-SVM

In the case of binary classification SVM, the original decision function of support vector machine (Equation 10) is rewritten as:

$$f(x) = sgn\{w \cdot \phi(x) + b\} = sgn\{\sum_{j=1}^{M} \alpha_j y_j k(x, Q_j(x_i)) + b\} \tag{25}$$

where $Q_j(x)$ stands for the sub-models, and the various sub-models provide the flexibility to contain the different local information. Here the local models are restricted by parametric models, in order to improve the performance of model. The original learning function (Equation 9) is rewritten as:

$$\Theta(\alpha) = min_{w,b} L(w, b, \alpha) = -\frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \langle Q_i(x_h) \cdot Q_j(x_g) \rangle + \sum_{i,j=1}^{M} \alpha_i \tag{26}$$

As the original training data set is partitioned by vector quantization, each local model is a Voronoi Region, which is represented by its codevector, $Q_j(x_g)$. For example, a sum-average of the original observation within a cluster can be expressed as:

$$Q_j(x_g) = c_j = \frac{\sum_{x_m \in S_j} x_m}{N}, m = 1, 2, .., N \tag{27}$$

where N becomes the number of observations in the cluster S_j. If a training observation x_g belongs to the cluster S_j, it is replaced by the codevector $Q_j(x_g) = c_j$ in the following

SVM model. Note the VQ functions are substituted into the previous decision function:

$$f(x) = sgn\{w \cdot \phi(x) + b\} = sgn\{\sum_{j=1}^{M} \alpha_j y_j k(x, c_j) + b\} \tag{28}$$

The VQSVM contains a semi-parametric hierarchical structure combining a set of local models represented by codevectors, and a global model represented by an SVM. In this thesis, this type of mixture model is classified as a semi-parametric model, as the SVM is a well-known nonparametric model and the VQ is a parametric model with a rather transparent structure.

This semi-parametric hierarchical model is a way of incorporating prior domain knowledge into inductive machine learning. The domain knowledge indicates there is a set of related, but not identical distributions. Each distribution is represented in formats of a local parametric model, which the global model then explores the unknown information. As an example of the semi-parametric hierarchical model, the VQSVM incorporates the prior domain knowledge, that is a set of related but not identical distributions, into an inductive machine learning algorithm that is an SVM. This is facilitated by the partition of observations done by the VQ and local parametric models constructed by codevectors.

6.3 Remarks of the Proposal Model

The proposed semi-parametric hierarchical modelling has multiple advantages. First, the hypothesis space of the VQSVM is initiated by both the training examples and the domain knowledge, as its kernel is constructed as $k(Q_l(x_i), Q_l(x_j))$. Thus the resultant model is consistent with both observations and domain knowledge. Second, the resultant model contains less SVs, reduceing the VC dimension. This results in the model having a better generalization. The kernel mapping is expressed mathematically as: $\phi : x \rightarrow (\sqrt{\lambda_j}\psi_j(x))_j$, $j = 1, \ldots, N$, and in practice the empirical kernel mapping is:

> For a given set $\{z_1, \ldots, z_m\}$, we call $\phi_m : R^N \rightarrow R_m$, $x \mapsto k(.,x)|_{z_1,\ldots,z_m} = (k(z_1, x), \ldots, k(z_m, x))$ the empirical kernel mapping w.r.t. $\{z_1, \ldots, z_m\}$ (Scholkopf, Mika, Burges, Knirsch, Muller, Ratsch, & Smola, 1999a).

The feature space is spanned by the mapped training examples. In practice, $\{z_1, \ldots, z_m\}$ is constructed by a set of support vectors (SVs). Thus reducing the number of SVs decreases the dimensions of the resultant feature space.

Third, in equation 10, the computation cost of $\sum_{i=i}^{n} \alpha_i y_i k(x, x_i)$ is determined by the n, that is the number of the SVs. The reduction of the number n of SVs decreases the complexity of the resultant model. Thus, the whole model is more efficient, simpler and more transparent.

However, this successful method has its limitations. For example, only the training examples are partitioned by the VQ. In case of the binary classification, the VQ is carried over into the positive and negative classes of training examples respectively. After the VQ, the new training examples are fed into the SVM for modelling. In the prediction, the new observation is fed into the SVM model without going through the VQ.

7. VQSVM for Imbalanced Data

The class imbalance problem typically occurs when, in classification problems, there are much more instances of some classes than others. In cases of extremely imbalanced (or skewed) data sets with high dimensions, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones. Therefore, machine learning becomes an extremely difficult task, and performances of regular machine learning techniques decline dramatically. In practical applications, the ratio of the small to the large classes can be drastic such as 1 to 100, or 1 to 1000 (Chawla, Japkowics, & Kolcz, 2004).

The recent journal Sigkdd explorations published a special issue on learning from imbalanced data sets (Chawla et al., 2004). This issue summarizes some well-known methods for dealing with problems of imbalanced data, for example undersampling and oversampling at the data level, one-class (cost-sensitive) learning and boasting at the algorithmic level. Random undersampling potentially removes certain important examples, and random oversampling leads to overfitting. In addition, oversampling introduces additional computational costs if the data set is already fairly large but imbalanced.

At the algorithmic level, cost-sensitive learning (chapter 3) aims to incorporate the risk factors of false positives and false negatives into the SVMs (Veropoulos, Campbell, & Cristianini, 1999)(Karakoulas & Shawe-Taylor, 1998)(Lin, Lee, & Wahba, 2002). Rehan Akbani et al implements SMOTE to imbalanced data sets and discusses the drawbacks of random undersampling. The SMOTE is a derivative of Support Vector Machine that gives the different error costs for different classes to push the boundary away from the minor class, (Akbani, Kwek, & Japkowicz, 2004). Gang Wu et al implements KBA, Kernel Boundary Alignment to imbalanced data sets (Wu & Chang, 2005).

There has been research performed and published regarding combining data compression and machine learning techniques. Jiaqi Wang et al combines k-means clustering and SVM to speed up real-time learning (Wang, Wu, & Zhang, 2005). Scholkopf and Smola discusses the combination between VQ and SVM in their book (Scholkopf & Smola, 2002a). They *kernelized* VQ as a new data compression method, Kernel VQ. The modeling method proposed by this thesis is different from their work as the VQSVM employs VQ to build a set of local models whose outputs will be input to a global model (a SVM). One of major applications of machine learning is data compression, so it is not surprising that many works have been done across these two fields.

Hierarchical learning machinery, VQSVM, is adapted to reduce the number of instances within the major class by using less local models to represent instances rather than simply eliminating instances randomly within random undersampling. Because the local models inherit major information from the instances, the information loss is much lower than the random undersampling. The local models can retrieval the information from the global model. However, the local models change the distribution of instances within the major class to some degree, but according to the experiments the information loss caused by this change is acceptable with respect to the improvement of overall performance.

7.1 Algorithms

The VQSVM is a hierarchical structure combining a set of local models that are represented by codevectors and a global SVM. In some literature this type of mixed model is named as semi-parametric model, because SVM is a well-known nonparametric model and VQ is a parametric model with a rather transparent structure. Semi-parametric hierarchical modelling thus becomes a way to incorporate domain knowledge into machine learning. In the case of extreme imbalance datasets, the majority class is represented by a set of local models that are codevectors, and the minority class keeps as original observations. The local models can decode and retrieve original instances from the codevectors. The number of local models in the majority is normally equal to the number of observations in the minority. However, a tuning of the number of the local models is required in order to reach the best performance.

Suppose the risk function of SVM is a hinge loss: $R[f] = \frac{1}{m}\sum_{i=1}^{m}|f(x_i) - y_i|$. The risk function will receive a penalty if one observation is misclassified, during the training process. In the imbalance data set, because the number of observations in the major class is much more than the numbers in a minor class, in the regular SVM, the penalties from the major class is often more than those from the minor class. The larger portion of the penalties comes from the major class. Thus the resultant hyperplane is pushed towards the minor class, and causes misclassifications of some observations in the minor class. A black point, representing an observation of minor class, is misclassified by the regular SVM (see Figure 11). Rehan Akbani et al and Gang Wu et al also found that in the case of imbalanced dataset, SVM always pushes the hyperplane towards a minority group. This hyperplane causes that the learning machine to be overwhelmed by the majority group, and a minority group losses its information completely (Akbani et al., 2004)(Wu & Chang, 2005). Cost-sensitive learning techniques, such as SMOTE, function well as the higher weights of penalties are assigned to the minor class by modifying the risk function (Akbani et al., 2004).

On the small imbalanced data sets, an excessively reduced number of observations in the minority class contain very limited information and might not be sufficient for learning. This is especially so when a large degree of class overlapping exists and the classes are further divided into subclusters (Tang & Liu, 2005). Japkowicz performed several experiments on artificial data sets and concluded that class imbalances do not seem to systematically cause performance degradation. She concludes that the imbalance problem is a relative problem depending on both the *complexity of the concept* and the *overall size of the training set* in addition to the *degree of class imbalance* present in the data. The complexity of the concept corresponds to the number of subclusters into which the classes are subdivided. These results indicate that the class imbalance problems are very domain specific instead of being caused only by the size of training data (Japkowicz, 2003). Even her work is carried by the C4.5 decision tree, and as a quite different classifier, the SVM is also sensitive to the class imbalance problem and it is worth testing whether the similar idea is valuable.

The Support Vector Machine selects a set of vectors along the hyper-plane, called support vectors. The random undersampling inevitably reduces the number of support vectors, and thus potentially loses information due to the removed support vectors. According to the theory of data compression, vector quantization is superior to random undersampling in terms of the information loss, but both of them suffer from another risk of information loss within the majority group. The SVM selects a subset of training instances, x_i, and uses them as the set of support vectors within the decision function 28. These support vectors lie on the margin, and their coefficients α_i are non-zero. In other words, as the hyperplane is

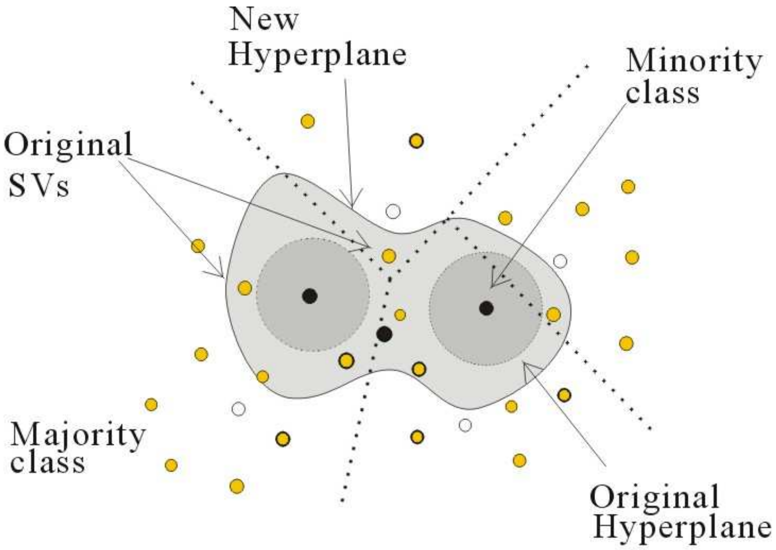

Figure 11: After training, in an imbalanced and linearly nonseparable data set the VQ replaces the original observations (light grey points) of the majority group by codevectors (empty points). The number of the codevectors is almost equal to the number of observations (dark grey points) of the minority group. The original maximum margin hyperplane (middle grey grey areas with dashed lines) learned by a soft margin SVM without the VQ is pushed towards a new position (middle grey areas with solid lines), which is much closer to the majority group. The previously misclassified observation (one dark grey point) is classified correctly by the new hyperplane (middle grey areas with solid lines).

Data set	Positive Insts	Negative Insts	Imbalance Ratio	No. of local models	number of subclusters
Abalone (19)	32	4145	1:129.54	32	28
Audit	13	1271	1:98	16	≥ 24
Yeast (5)	47	1437	1:30.5	64	8
Letter (26)	734	19266	1:26.28	299 (1024)	25
				550 (2048)	25
Glass (7)	29	185	1:6.3	32	6

Table 1: Four UCI data sets and Audit data set with the numbers of local models.

completely determined by the instances closest to it, the solution does not depend on other examples (Scholkopf & Smola, 2002a). Vector Quantization replaces the original SVs by their corresponding codevectors. The codevectors become new SVs and push the optimal hyperplane away from the original one trained by imbalanced data (cf. Figure 11). The Pseudo-code of the algorithm VQSVM is:

Algorithm 7.1: VQSVM(*dataset*)

```
comment: Step 1: Set parameters of VQSVM
comment: the kernel parameter g
$Float : g$
while $UntilTheOptimalPointOfTradeoffBetweenInformationLossandAccuracy$
do
    comment: the number of code-vectors
    $int : numberOfLocalModels$
    $LocalModel = LBGvq(Majority, numberOfLocalModels)$
    $NewTrainingData = combine(BalancedMajority, Minority)$
    comment: SVM based on the new balanced data
    $Model = SVM(NewTrainingData, g);$
```

VQSVM sacrifices the information held by the majority group to retrieve the information contained by the minority group. This is very important in many real life scenarios, which emphasize the minority groups. The number of local models is tuned by the VQSVM to minimize the information loss of majority group. Therefore the optimal model is a trade-off between the number of local models and the improved balance ratio in order to improve the classification accuracy.

7.2 Experiments

Four UCI data sets and a data set collected by Li and Stokes in the audit research are used for this evaluation (Li, Stokes, & Hamilton, 2005). The UCI data sets experimented with are abalone (abalone19), yeast (yeast5), glass (glass7), and letter (letter26). The number

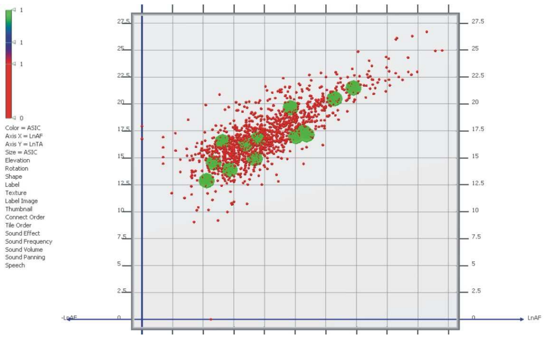

Figure 12: In the audit data set, the observations (the bigger green points) of the minority class is scattered with these (the smaller red points) of the majority class

in the parentheses indicates the target class choosen. Table 1 shows the characteristics of these six datasets organized according to their negative-positive training-instance ratios. The top dataset (abalone19) is the most imbalanced with a ratio of about 1 : 130. The abalone data set consists of continuous data instead of categorical data. It is expected that undersampling at high rates generate a trade-off between improved data balance and loss of important information, and we examined whether different number of local models could lead to a further enhancement of results. For example, in the letter(26) data set, the experiments are carried out over two data sets with two various number of local models (see Table 1).

Through the initial exploration, in the five data sets, the minority class is not linearly separated from the majority class. The minority class is scattered within the majority class (see Figure 12).

The machine learning community uses two metrics, the sensitivity and the specificity, for evaluating the performance of various tests. Sensitivity is defined as the accuracy on positive instances, $Sensitivity = \frac{TruePositives}{TruePositives+FalseNegatives}$. Specificity, however, is defined as the accuracy on negative instances, $Specificity = \frac{TrueNegatives}{TrueNegatives+FalsePositives}$ (Akbani et al., 2004). Kubat et al suggest the g-means, which combines specificity and sensitivity, $g = \sqrt{Specificity \times Sensitivity}$ (Kubat & Matwin, 1997). In our experiments, the g-means replace the standard accuracy rates, which do not make any sense in imbalanced data sets.

In our experiments, we compare the performance of the modelling, VQSVM, with the regular SVM, random undersampling and random oversampling (Akbani et al., 2004). The SVM uses the LibSVM code and the Vector Quantization uses the DCPR Matlab toolbox (Chang & Lin, 2004)(Jang, 2005). All tests involving the SVM use the RBF c-SVM with an identical gamma value $\gamma = 0.5$ and a c value $c = 10$. Each dataset is randomly split into training and test sets in the ratio 8 to 2, and only the Abalone(19) is randomly split into train and test sets in the ratio 7 to 3, because under the ratio 8 to 2, the experiments show that the Abalone (19) loses the accuracy up to the g-mean value 0.1 with the random undersampling. For the random undersampling algorithm, we undersample the training

	SVM			VQSVM		
Data set	Se	Sp	G	Se	Sp	G
Abalone (19)	0	1	0	0.8	0.87623	0.83099
Audit	0	0.9948	0	0.2500	0.8583	0.4632
Yeast (5)	0	1	0	1	0.8606	0.9277
Letter (26)	0.3537	1	0.5948	1	0.1871	0.4326
				0.7143	0.9992	0.8448
Glass (7)	0.6667	1	0.8165	0.6667	1	0.8165

	Random Undersampling			Random Oversampling		
Data set	Se	Sp	G	Se	Sp	G
Abalone (19)	0.3000	0.8198	0.4959	0.6000	0.8101	0.6972
Audit	0.2750	0.6900	0.3653	0	1	0
Yeast (5)	0.8600	0.9519	0.9017	1	0.8084	0.8991
Letter (26)	1	0.0401	0.1998	0.4218	1	0.6494
	0.7150	0.9993	0.8453	0.4218	1	0.6494
Glass (7)	0.8333	0.9757	0.8966	0.6667	1	0.8165

Table 2: Test Results: In the random undersampling, experiments are carried out ten times for each data set. This reduces the error caused by the random selection of observations. For example, in the data set Yeast(5), we randomly withdrew 64 observations from original train data 10 times. The values in the table are the mean of the resultant g-mean values, and the standard deviations of g-mean values are $\{0.1264(Abalone), 0.2600(Audit), 0.0612(yeast), 0.0013(Letter), 0.0758(glass)\}$

data until both the classes were equal in number, as Japkowicz did in her experiments (Japkowicz, 2003).

The results of these experiments (see table 2) shows that the g-means of the VQSVM are better or equal to those of the standard SVM. In detail, the specificities of the SVM are better than the VQSVM, but the SVM predicts all of instances as negative. Thus the specificities of standard SVM do not make any sense. In the dataset, Letter (26), while the VQSVM sets the number of local models extremely low, a new imbalanced data set is produced. As a consequence, the predictive results of this data set show that the positive group overwhelms the learning machine.

7.3 Summary

The results of these experiments prove our theoretic argument: the SVM is highly sensitive to the imbalanced data, and majority groups often overwhelm the learning machine. Similar to Gustavo Batista's report, random oversampling is more consistent than random undersampling in the case of the Support Vector Machine. In the case of large amounts of training data containing imbalanced classes, oversampling increases the number of training examples in the minority classes, and therefore introduces more computation costs. Compared to three other methods, the VQSVM is the most stable: from the abalone(19) with the highest imbalance ratio 130 : 1 to the glass(7) with the lowest imbalance ratio 6 : 1, the g-mean values of the VQSVM hold at 0.8.

The results of these experiments coincide with the arguments Japkowicz proposes (Japkowicz, 2003). In the highly imbalanced data sets containing more subclusters, the performance of the VQSVM is superior to the random undersampling. More importantly, the VQSVM proposes a hierarchical structure, in which local models can not be codevectors. The hierarchical structures give researchers more flexibility to incorporate varied models representing domain knowledge into one mixed model to deal with imbalanced data sets.

Aside from the VQSVM, incorporating domain knowledge is always helpful when addressing the problem of imbalance. With SMOTE, domain knowledge can be used to determine the initial value of different error costs assigned to classes. This domain knowledge helps overcome the sensitivity of inductive machine learning algorithms and then produces more stable models. Instability means that arbitrarily small perturbations in data sets can produce arbitrarily large perturbations in the solution. This is especially true for infinite-dimensional problems, because for finite-dimensional problems that perturbation is always finite. But the key of this problem is that the solution to the new problem is less sensitive to the perturbations (Hansen, 2005).

Most non-parametric methods, such as SVM, provide ways to stabilize and regularize the problem. However, their regularization (See Section 2.1.3) is often a global solution, which assumes that the underlying distribution is identical. If the underlying distribution is not identical, it becomes highly risky to make these assumptions, and causes instabilities of the resultant model. In contrast, hierarchical modelling practises alternative approaches, that separates the related distributions first, building a local model over each sub-region, and then unifying local models via a nonparametric model. From the experiments, the VQSVM overcomes the sensitivity created due to the imbalance suffered by the SVM. With balanced data, the performance of the VQSVM is equivalent to the performances of SVM. In the case of extremely imbalanced data, VQSVM reaches an acceptable level of accuracy while SVM is already overwhelmed by the majority groups.

The current results show a significant improvement in binary classification. In further works, it is necessary to investigate more precise controls. Especially the local models representing only support vectors instead of all of vectors may enhance the controllability of the VQSVM and manage the information loss that occurs with VQSVM.

There is one issue the VQSVM is unable to address: the size of the training data. This phonomania has been observed in the experiment over the data set "Audit". If the size of the training dat, or more precisely, the size of minor class is extremely small, the information contained will be too limited to be learned by the VQSVM, thus worsening its performance.

8. Conclusion

The VQSVM is a hierarchical modelling method which combines vector quantization and support vector machine. The existing prior domain knowledge indicates there are multiple subclasses within the majority side, and this complexity aggravates the negative effects caused by the imbalance ratio between two sides. This prior domain knowledge indicates that the hierarchical modelling is a proper modelling method, even though the knowledge transfer between local models has not functioned. According to the results of experiments, this hierarchical model performs equally or better over a variety of data sets with different imbalanced ratio and the number of local models. This result satisfies the specific properties required by Tom Mitchell (Mitchell, 1997b).

As a conclusion, this paper reviews the recent developments of incorporating prior domain knowledge into inductive machine learning, and proposes a guideline that incorporates prior domain knowledge in three key issues of inductive machine learning, i.e. consistency, generalization and convergence. The variety and diversity of domain knowledge causes the difficulty of measuring the relatedness between domain knowledge and tasks and representing it in the learning system. This difficulty results in the absence of universal solution to this problem. This paper gives only a guideline to this topic, but while facing a task, readers need to consider the characters of the given inductive learning algorithms and the existing domain knowledge to work out a particular method to reach the best performance.

Acknowledgments

The authors wish to thank the International Institute of Forecasters (IIF), the SAS Institute, the Faculty of Information Technology, University of Technology, Sydney and the Capital Markets CRC Australia for their supports to this research.

References

Abu-Mostafa, Y. S. (1995). Hints. *Neural Computation*, *7*, 639–671.

Abu-Mostafa, Y. S. (2001). Financial model calibration using consistency hints. *IEEE Trans. on Neural Networks*, *12*(4), 791?808.

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*, chap. 10. NP-Complete Problems. Addison-Wesley series in computer science and information processing. Addison-Wesley.

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pp. 39–50.

Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Bousquet, O., Boucheron, S., & Lugost, G. (2004). Introduction to statistical learning theory. In *Advanced Lectures on Machine Learning Lecture Notes in Artificial Intelligence 3176*, pp. 169–207. Springer, Heidelberg, Germany.

Burges, C. J., & Scholkopf, B. (1997). Improving the accuracy and speed of support vector learning machine. *Advances in Neural Information Processing System*, *9*, 375–381.

Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *On the IEEE Transaction on Neural Networks*, *14*(6).

Chang, C., & Lin, C. (2001). Libsvm: a library for support vector machines (version 2.3). Tech. rep..

Chang, C.-C., & Lin, C.-J. (2004). Libsvm: a library for support vecter machine. Tech. rep., Department of Computer Sicence and Information Engineering, National Taiwan University.

Chawla, N. V., Japkowics, N., & Kolcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations*, *6*(1).

Davidson, I., & Ravi, S. S. (2005). Hierarchical clustering with constraints: Theory and practice. In *the 9th European Principles and Practice of KDD, PKDD.*

Decoste, D., & Scholkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, *46*, 161–190.

Fung, G. M., Mangasarian, O. L., & Shavlik, J. W. (2001). Knowledge-based support vector machine. Tech. rep., Data Mining Institute, Computer Science Department, University of Wisconsin, Madison.

Fyfe, W. J. A. (1992). *Invariance Hints and the VC Dimension.* Ph.D. thesis.

Galindo, J., & Tamayo, P. (2000). Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications. *Computational Economics*, *15*(1-2), 107 – 143.

Gersho, A., & Gray, R. M. (1992). *Vector Quantization And Signal Compression.* Kluwer Academic Publishers.

Giorgio Fumera, F. R. (2002). Cost-sensitive learning in support vector machines..

Hansen, P. C. (2005). Discrete inverse problems, insight and algorithms, a tutorial with matlab exercises. Tech. rep., Informatics and Mathematical Modelling Building 321, Technical University of Denmark DK-2800 Lyngby, Denmark.

Jan, T., Yu, T., Debenham, J., & Simoff, S. (2004). Financial prediction using modified probabilistic learning network with embedded local linear models. In *CIMSA2004-IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, Boston, MD, USA.

Jang, J.-S. R. (2005). Dcpr matlab toolbox..

Japkowicz, N. (2003). Class imbalances: Are we focusing on the right issue?. In *Workshop on Learning From Imbalanced Data Sets II.*

Jin, Y., & Sendhoff, B. (1999). Knowledge incorporation into neural networks from fuzzy rules. *Neural Processing Letters*, *10(3)*, 231–242.

Karakoulas, G., & Shawe-Taylor, J. (1998). Optimizing classifiers for imbalanced training sets. In *Advances in neural information processing systems*, pp. 253 – 259. MIT Press, Cambridge, MA, USA.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced data sets: One-sided sampling. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186.

Langseth, H., & Nielsen, T. D. (2003). Fusion of domain knowledge with data for structural learning in object oriented domains. *Journal of Machine Learning Research*, *4*, 339–368.

Li, Y., Stokes, D., & Hamilton, J. (2005). Listed company auditor self-selection bias and audit fee premiums..

Lin, Y., Lee, Y., & Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning*, *46*(1-3), 191 – 202.

Linde, Y., Buzo, A., & Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 702–710.

Mangasarian, O. L., Shavlik, J. W., & Wild, E. W. (2004). Knowledge-based kernel approximation. *Journal of Machine Learning Research*, *5*, 1127–1141.

Mattera, D., Palmieri, F., & Haykin, S. (2001). Semiparametric support vector machines for nonlinear model estimation. In Haykin, S. (Ed.), *Intelligent Signal Processing*, pp. 295–305. IEEE Press, New York.

Mitchell, T. M Thrun, S. (1993). Explanation-based neural network learning for robot control.. In S. Hanson, J. Cowan, . C. G. (Ed.), *Advances in neural information processing systems*, Vol. 5, pp. 287–294. Morgan-Kaufmann Press, San Mateo, CA.

Mitchell, T. (1997a). *Machine Learning*. McGraw-Hill.

Mitchell, T. (1997b). *Machine Learning*, chap. 12, Combining Inductive and Analytical Learning. McGraw-Hill.

Niyogi, P., Girosi, F., & Poggio, T. (1998). Incorporating prior information in machine learning by creating virtual examples. In *IEEE*, Vol. 86.

Pazzani, M., Brunk, C., & Silverstein, G. (1991). A knowledge-intensive approach to learning relational concepts. In *The Eighth International Workshop on Machine Learning*, pp. 432–436, San Mateo, CA. Morgan Kaufmann.

Poggio, T. (2003). Lecture 2, learning problem and regularization, in the statistical learning theory and applications, mit opencourseware..

Russell, S. J. (1991). Prior knowledge and autonomous learning. *Robotics and Autonomous Systems*, *8*, 145–159.

Scholkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Muller, K.-R., Ratsch, G., & Smola, A. J. (1999a). Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*.

Scholkopf, B., Simard, P., Smola, A., & Vapnik, V. (1999b). Prior knowledge in support vector kernels..

Scholkopf, B., & Smola, A. J. (2002a). *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Scholkopf, B., & Smola, A. (2002b). *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*, chap. 11. Incorporating Invariances, pp. 333–348. MIT Press.

Sill, J., & Abu-Mostafa, Y. (1997). Monotonicity hints. In *Advances in Neural Information Processing Systems 9 (NIPS'96)*, pp. 634–640. MIT Press.

Silver, D. L. (2000). *Selective Transfer of Neural Network Task Knowledge.* Ph.D. thesis, University of Western Ontario.

Simard, P., Cun, Y. L., & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5*, pp. 50–68. Morgan Kaufmann.

Simard, P., Victorri, B., Cun, Y. L., & Denker, J. (1992). Tangent prop – a formalism for specifying selected invariances in an adaptive network. In Moody, J. (Ed.), *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, San Mateo, CA.

Suddarth, S., & Holden, A. (1991). Symbolic-neural systems and the use of hints for developing complex systems. *International Journal of Man-Machine Studies*, *35*(3), 291–311.

Tang, L., & Liu, H. (2005). Bias analysis in text classification for highly skewed data. In *the proceeding of International Conference on Data Mining.*

Thrun, S. (1996). *Explanation based neural network learning: A lifelong learning approach.* Kluwer Academic Publishers, Boston.

Towell, G., & Shavlik, J. (1989). An approach to combining explanation-based and neural learning algorithms. *Connection Science*, *1*, 233–255.

Tresp, V., & Yu, K. (2004). An introduction to nonparametric hierarchical bayesian modelling with a focus on multi-agent learning. In *Proceedings of the Hamilton Summer School on Switching and Learning in Feedback Systems. Lecture Notes in Computing Science.*

Tsang, E., Yung, P., & Li, J. (2004). Eddie-automation, a decision support tool for financial forecasting. *Decision Support Systems*, *37*, 559–565.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *the International Joint Conference on Artificial Intelligence (IJCAI99), Workshop ML3*, pp. 55–60, Stockholm, Sweden.

Wang, J., Wu, X., & Zhang, C. (2005). Support vector machines based on k-means clustering for real-time business intelligence systems. *International Journal of Business Intelligence and Data Mining*, *1*(1).

Wu, G., & Chang, E. (2005). Kba: kernel boundary alignment considering imbalanced data distribution. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 786 – 795.

Wu, X., & Srihari, R. (2004). Incorporating prior knolwedge with weighted margin support vector machines. In *KDD 04*, Seattle, Washington, USA. ACM.

Yoon, S.-C., Henschen, L. J., Park, E. K., & Makki, S. (1999). Using domain knowledge in knowledge discovery. In *Proceedings of the eighth international conference on Information and knowledge management*, pp. 243 – 250, Kansas City, Missouri, United States. ACM Press New York, NY, USA.