Simpler Knowledge-based Support Vector Machines

Quoc V. Le QUOC.LE@ANU.EDU.AU
Alex J. Smola ALEX.SMOLA@NICTA.COM.AU
RSISE, Australian National University, 0200 ACT, Australia; and
Statistical Machine Learning Program, National ICT Australia, 0200 ACT, Australia

Thomas Gärtner THOMAS.GAERTNER@AIS.FRAUNHOFER.DE
Fraunhofer AIS.KD, Schloß Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

If appropriately used, prior knowledge can significantly improve the predictive accuracy of learning algorithms or reduce the amount of training data needed. In this paper we introduce a simple method to incorporate prior knowledge in support vector machines by modifying the hypothesis space rather than the optimization problem. The optimization problem is amenable to solution by the constrained concave convex procedure, which finds a local optimum. The paper discusses different kinds of prior knowledge and demonstrates the applicability of the approach in some characteristic experiments.

1. Introduction

Prior knowledge may be available in various forms in learning problems. For instance, we might have information about the degree of smoothness a function has (Smola et al., 1998), information on equivalence classes of patterns (Graepel & Herbrich, 2004; DeCoste & Schölkopf, 2002), or information on the derivatives and similarity of observations at various locations (Mammen et al., 2001; Smola & Schölkopf, 1998). Beyond that, Bayesian priors (Neal, 1996; MacKay, 2003) are designed to capture prior knowledge. If appropriately used, prior knowledge can significantly improve the predictive accuracy of learning algorithms or reduce the amount of training data needed.

The type of prior knowledge this paper is concerned with is where for a given estimation problem, say, to map from domain $\mathcal{X}$ to a domain $\mathcal{Y}$ (typically reals or a subset of integers), we know for a subset $\mathcal{X}' \subseteq \mathcal{X}$ that the mapping $f : \mathcal{X} \to \mathcal{Y}$ is restricted to a subset $\mathcal{Y}' \subseteq \mathcal{Y}$. This is a slightly more general setting than the one studied with great success by Fung et al. (2002a); Fung et al. (2000); and Mangasarian et al. (2004). In particular, it includes the following cases:

Binary classification ($\mathcal{Y} = \{\pm 1\}$): Prior knowledge may be rules that $y = 1$ or $y = -1$ for a certain set of x. If we classify based on the sign of a real-valued function $f(x)$ subject to a large margin condition, this amounts to imposing that $f(x) \geq 1$ for $y = 1$ and $f(x) \leq 1$ for $y = -1$.
In other words, we have lower and upper bounding functions $l(x) = 1$ and $u(x) = -1$ such that $f(x) \geq l(x)$ for $x \in \mathcal{X}_l$ and $f(x) \leq u(x)$ for $x \in \mathcal{X}_u$. This is the setting of (Fung et al., 2002a).

Scalar regression ($\mathcal{Y} = \mathbb{R}$): Prior knowledge may come in the form of upper and lower bounds explicitly. That is, we may know that the regression may not exceed a certain function $u(x)$ on a domain $\mathcal{X}_u$ and it must be larger than $l(x)$ on another domain $\mathcal{X}_l$. These cases were considered e.g. in (Fung et al., 2000).
A real-world example where such constraints occur is the estimation of the demand for a product from its sales which are bound by the supply.

Multiclass classification ($\mathcal{Y} = \{1, \ldots, N\}$): Prior knowledge may exclude a subset of classes. For large margin classification f is vector valued ($f : \mathcal{X} \to \mathbb{R}^N$). Classification is carried out by a winner-take-all approach, i.e. $y(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f_y(x)$. In this case, a solution using constraints immediately is rather difficult to obtain. However, a restriction of $\mathcal{Y}$ to $\mathcal{Y}(x)$ which incorporates prior knowledge in conjunction with $y(x) = \operatorname{argmax}_{y \in \mathcal{Y}(x)} f_y(x)$ solves the problem.

In the present paper we show how to incorporate these constraints directly into the solution of an estimation problem. Unlike Mangasarian's work, we do not modify the optimization problem but the estimate. This allows us to easily

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006.

incorporate flexible types of prior knowledge.

In Section 2 we discuss previous work on knowledge based estimation. Section 3 introduces the clipping transformation used to incorporate the prior knowledge and Section 4 describes the nonconvex optimization problems. Finally, experiments are given in Section 5 and Section 6 concludes.

2. Previous Work

Knowledge-based estimation in the context of Support Vector Machines has been studied by Mangasarian and coworkers. The basic premise of their approach is that in the case of a polyhedral set of constraints it is possible to formulate constrained optimization problems such that in addition to finding a large margin solution the estimate also satisfies the prior knowledge constraints.

Fung et al. (2002a) and Fung et al. (2000) studied a reformulation of linear and nonlinear support vector classifiers that can incorporate prior knowledge in the form of multiple polyhedral sets, each belonging to one of two categories. An extension to regression was given by Mangasarian et al. (2004) which studied linear and non-linear support vector machines with prior knowledge in the form of linear inequalities to be satisfied over multiple polyhedral sets.

Despite their theoretical elegance these methods suffer some shortcomings:

- Imposing prior knowledge for the purpose of estimation may actually decrease the performance of the estimate, when imposed in the form of constraints. Consider the extreme case where a problem (without constraints) is linearly separable. Now we are given knowledge that in a certain region where the probability of drawing $x \in \mathcal{X}$ is vanishingly small contradicts the data. In this situation the optimization problem suddenly becomes infeasible and a nonlinear estimate is required. More to the point, we have knowledge about the labels on a certain domain rather than about the distribution on this domain.
- The region about which we have prior knowledge may cover cases which are extremely unlikely to occur. In this case the prior knowledge would mislead the estimator into finding an estimate with possibly considerably higher capacity to accommodate this extra piece of information without any significant improvement in the generalization performance.
- Not all rules may be encoded in the form of polyhedral sets. If so (e.g. when using Gaussian RBF kernels (Micchelli, 1986) the problems can always be re-encoded as polyhedral sets), this may lead to constraints with excessively large RKHS norm.
- Such constraints and knowledge may only be enforced on a subset of points (except for linear rules). Thus, re-encoding only ensures approximate satisfaction.

In summary, the optimization becomes more difficult and it is encumbered by a re-encoding problem which has nothing to do with the actual estimation problem. Finally, these methods lead to rather complex optimization problems that require fairly sophisticated convex optimization tools. This can be a barrier for practitioners, as they would need to express prior knowledge in terms of polyhedral sets.

3. Clipping Transformation

3.1. Basic Setting

We now present an alternative approach to solving the estimation problem. Before we do so, let us formalize the setting. Assume that we observe $\{(x_1, y_1), \ldots, (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$ drawn independently and identically from some distribution $\Pr(x, y)$. Moreover assume that there exists some loss function $c(x, y, f)$ which penalizes estimates in relation to the actual observations y. For notational simplicity in the remainder we assume that c is convex in $f(x)$. For nonconvex loss functions we need to decompose the latter into a sum of convex and concave parts, as will be described in the constrained concave convex procedure.

Using a standard argument of minimizing the regularized risk (Schölkopf & Smola, 2002) as a proxy of the expected risk we minimize

$$R_{\mathrm{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} c(x_i, y_i, f) + \lambda \Omega[f] \tag{1}$$

where f is an element of a function space $\mathcal{F}$, $\Omega : \mathcal{F} \to [0, \infty)$ is a regularization functional, and $\lambda > 0$ is the regularization parameter.

Now, denote by $\mathcal{C}$ the set of all functions satisfying knowledge-based constraints. Mangasarian and coworkers minimize (1) over $f \in \mathcal{F} \cap \mathcal{C}$ using the fact that (in their setting) $\mathcal{F} \cap \mathcal{C}$ is convex and that for polyhedral $\mathcal{C}$ the problem can be solved by a quadratic program. If, however, $\mathcal{C} \cap \mathcal{F} = \emptyset$ the problem is infeasible. Moreover, it is also immediately clear that for the minimizer f^* of $R_{\mathrm{reg}}[f]$ subject to $f \in \mathcal{F}$ and the minimizer g^* of $R_{\mathrm{reg}}[g]$ subject to $g \in \mathcal{F} \cap \mathcal{C}$ the value $\Omega[g^*]$ can not be smaller than $\Omega[f^*]$. Typically, $\Omega[g^*]$ will indeed be much larger.

While we will focus on reproducing kernel Hilbert space regularization (Aronszajn, 1950) in this paper, also other forms are possible. For instance, the use of ℓ_1 regularization (Mangasarian, 2000; Schölkopf & Smola, 2002) or of sparsity enforcing priors (Tipping, 2000; Fung et al., 2002b) is entirely possible.

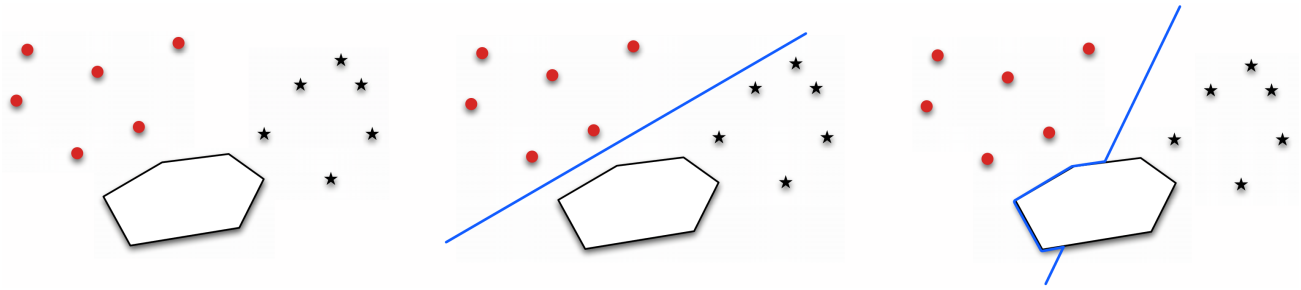

Figure 1. Left: binary classification problem with additional knowledge that the polyhedral set belongs to the class of stars; middle: solution obtained by Mangasarian's knowledge based SVM; right: solution obtained by our setting. The separator is no longer a hyperplane but the union of a halfspace and the polyhedral set. Note that Mangasarian's problem can become infeasible due to the knowledge-based constraints whereas our approach does not suffer from these problems.

3.2. A Constraint Satisfaction Transform

Instead of restricting $\mathcal{F}$ to $\mathcal{F} \cap \mathcal{C}$ we pursue a different strategy. Denote by ψ a map ensuring that $\psi \circ f \in \mathcal{C}$ for all $f \in \mathcal{F}$ (we use $\psi \circ f$ for notational conveniance though it is not really a composition of functions as $\psi \circ f(x)$ depends not only on $f(x)$ but also on x). We then minimize

$$R_{\text{reg}}[f, \psi] := \frac{1}{m} \sum_{i=1}^{m} c(x_i, y_i, \psi \circ f) + \lambda \Omega[f] \qquad (2)$$

over $f \in \mathcal{F}$. Before we discuss the implications, let us consider some practical cases:

i. **Binary Classification** If we have prior knowledge about the sign at a location, we define

$$\psi \circ f(x) := \begin{cases} \max(1, f(x)) & \text{if we know } y = 1 \\ \min(-1, f(x)) & \text{if we know } y = -1 \end{cases}$$

In other words, if $yf(x) \geq 1$ we do not modify f. In all other cases, we ensure that $y\psi \circ f(x) \geq 1$. See Figure 1 for an example of such a classifier.

ii. **Regression** Assume for simplicity that we are given lower and upper bounds $l(x)$ and $u(x)$ throughout the entire domain $\mathcal{X}$ (we can always achieve that by setting $l(x) = -\infty$ wherever we have no lower bound and $u(x) = \infty$ analogously). In this case we define

$$\psi \circ f(x) := \max(l(x), \min(u(x), f(x))).$$

Clearly, $\psi \circ f$ satisfies the constraints that $l(x) \leq \psi \circ f(x) \leq u(x)$.

iii. **Monotonicity** In the estimation of growth curves and similar functions where monotonicity is known a priori (Mammen et al., 2001), we may impose such properties directly by defining

$$\psi \circ f(x) := \max_{z \leq x} f(z).$$

iv. **Multiclass Classification** Here the knowledge comes in the form of exclusion of certain subsets of labels. In this case we may set

$$\psi \circ f(x, y) := \begin{cases} -\infty & \text{if label } y \text{ is excluded} \\ f(x, y) & \text{otherwise} \end{cases}$$

Such situations occur, e.g. in Natural Language Processing applications, where a grammar may exclude certain annotations of a sequence.

v. **Even or odd functions** In the estimation of even functions or odd functions:

$$\psi \circ f(x) := \frac{f(x) + f(-x)}{2}$$

for even functions; and

$$\psi \circ f(x) := \frac{f(x) - f(-x)}{2}$$

for odd functions.

The advantage of using ψ instead of restricting $\mathcal{F}$ to $\mathcal{F} \cap \mathcal{C}$ is that it allows for a very flexible encoding of prior knowledge. In particular, we do not need to re-encode any rules by functions in $\mathcal{F}$. Instead, all knowledge constraints are satisfied by default. Furthermore, this approach has theoretical advantages that we will sketch in the following.

Risk minimization algorithms over $f \in \mathcal{F}$ with regulariser Ω are often analysed in terms of the generalisation error of the function classes $\mathcal{F}_B = \{f \in \mathcal{F} : \Omega[f] \leq B\}$. As the generalisation error of function classes can be bound from above by a function linear in its Rademacher complexity (Bartlett & Mendelson, 2002), it is important to investigate the impact of ψ on the Rademacher complexity.

Due to space limitations we restrict our considerations wlog to function classes closed under negation. The margins of the functions $\psi \circ f$ considered in (*i*-*iv*) are equal

to the margin of $f(x)$ for x, y that do not violate the constraints and are equal to some other function, say $h(x, y)$ for x, y that violate the constraints. We will now use ψ_h instead of ψ to clarify this dependence between ψ and h. Let $Z \subseteq \mathcal{X} \times \mathcal{Y}$ denote the set of x, y-pairs that violate a constraint, let $\mathcal{F}_B^h = \psi_h \circ \mathcal{F}_B = \{\psi_h \circ f : f \in \mathcal{F}_B\}$, let $m_f(x, y)$ denote the margin of f at (x, y), and let $\mathcal{MF}$ denote the set of margin functions of $\mathcal{F}$. The Rademacher complexity $\hat{\mathcal{R}}_l(\mathcal{F}_B^h)$ on a sample $S \subseteq \mathcal{X} \times \mathcal{Y}$ of length $|S| = l$ is then independent of the choise of h as

$$\begin{aligned}
\hat{\mathcal{R}}_l(\mathcal{MF}_B^h) =& \mathbf{E}_\sigma \sup_{g \in \mathcal{F}_B^h} \frac{2}{l} \sum_{(x_i,y_i)\in S} \sigma_i m_g(x_i, y_i) \\
=& \mathbf{E}_\sigma \sup_{f \in \mathcal{F}_B} \frac{2}{l} \sum_{(x_i,y_i)\in S\setminus Z} \sigma_i m_f(x_i, y_i) \\
&+ \mathbf{E}_\sigma \sum_{(x_i,y_i)\in S\cap Z} \sigma_i h(x_i, y_i) \\
=& \mathbf{E}_\sigma \sup_{f \in \mathcal{F}_B} \frac{2}{l} \sum_{(x_i,y_i)\in S\setminus Z} \sigma_i m_f(x_i, y_i) \\
=& \frac{|S \setminus Z|}{l} \mathcal{R}_{|S\setminus Z|}(\mathcal{MF}_B)
\end{aligned}$$

where $\sigma_i \in \{\pm 1\}$ are zero-mean Rademacher variables. This carries over to the expected Rademacher complexities $\mathbf{E}_S \hat{\mathcal{R}}_l(\mathcal{MF}_B^h)$.

For the usual loss functions used in regression or classification $\mathcal{R}_l(\mathcal{MF}_B)$ can be upper bound by a function linear in $\mathcal{R}_l(\mathcal{F}_B)$. If we now restrict $\mathcal{F}$ to form a Hilbert space and use the regulariser $\Omega[f] = \|f\|^2$, one usually bounds $\hat{\mathcal{R}}_l(\mathcal{F}_B) \leq \frac{2B}{l}\sqrt{\sum_{i=1}^l k(x_i, x_i)}$ where k is the reproducing kernel of the Hilbert space. For $\frac{|S\setminus Z|}{l}\hat{\mathcal{R}}_{|S\setminus Z|}(\mathcal{F}_B)$ we obtain in this setting

$$\begin{aligned}
\frac{|S \setminus Z|}{l} \hat{\mathcal{R}}_{|S\setminus Z|}(\mathcal{F}_B) \leq& \frac{2B}{l} \sqrt{\sum_{i=1}^{|S\setminus Z|} k(x_i, x_i)} \\
\leq& \frac{2B}{l} \sqrt{\sum_{i=1}^{l} k(x_i, x_i)} \\
=& \mathcal{R}_l(\mathcal{F}_B)
\end{aligned}$$

an at least as good upper bound.

Last but not least for (v) it is useful to consider function classes not just closed under negation but also closed under negation of the argument. In this case $\psi \circ \mathcal{F}$ is a set of convex combinations of functions from $\mathcal{F}$ and has thus the same upper bound on the Rademacher complexity as $\mathcal{F}$.

We now showed, that using ψ does not increase the Rademacher complexity of the considered function classes $\mathcal{F}_B$. Furthermore, the use of ψ potentially reduces the value of Ω for the minimiser considerably and thus allows us to use $\mathcal{F}_B$ with much smaller B and thus Rademacher complexity.

	Upper bound active	Lower bound active
$\partial_z c(x, y, z) < 0$	**case (1):** convex	**case (2):** non-convex
$\partial_z c(x, y, z) > 0$	**case (3):** non-convex	**case (4):** convex

Table 1. Value of the loss $c(x, y, z)$ versus $f(x)$ for convex loss function (f, dotted) and effective loss function (taking $\psi \circ f$ into account, solid) arising from the constraints $l(x) \leq \psi \circ f(x) \leq u(x)$. The distinction between the top and the bottom row of the table depends on the derivative of the loss function at the constraint, i.e. at $u(x)$ for upper bounds and at $l(x)$ for lower bounds.

3.3. Implied Transformations for Convex Loss

We now focus on constraints of the form $l(x) \leq \psi \circ f(x) \leq u(x)$ and convex loss functions s. This covers regression and classification from the previous section. The price we pay for using $\psi \circ f$ is that $c(x, y, \psi \circ f)$ need not be convex in $f(x)$ any more, depending on our problem. Without loss of generality, we now consider lower and upper constraints separately. The two may obviously be combined.

Depending on l and u, the loss $c(x, y, \psi \circ f)$ is piecewise convex or constant. We now study four relevant cases as to where the constraints l and u can become active. Depending on where the cut occurs (lower or upper) and the sign of the partial derivative of the loss at the upper or lower constraint, we may retain convexity. See Table 1 for further details.

3.4. Convergence Analysis

While it is easy to construct cases where our approach fails and the mathematical programming method is able to solve the problem easily — and vice versa — we now provide some more theoretical justification why on average the simpler approach should perform better. We study the case of binary classification. An extension to other settings is rather straightforward.

Denote by A the domain where the knowledge rule applies and let $\pi := \Pr\{A\}$ be the measure of that domain.

Moreover assume that we have a model selection procedure whose generalization performance will converge to the optimal solution as $O(m^{-\alpha})$ where typically $\alpha = \frac{1}{2}$. Losely speaking, with high probability $R[\hat{f}] - R[f^*] \leq Cm^{-\alpha}$. Here f^* denotes the minimizer of the expected risk.

When using a mathematical programming knowledge based SVM procedure it follows that the rate of convergence is essentially $O(m^{-\alpha})$. In our approach we only need $\hat{f}$ for all the cases where $x \notin A$. In other words, we only need to estimate the answer in $(1-\pi)$ of all cases. Obviously we can expect to see only $(1-\pi)$ of the data in this situation, too. This means that rather than $Cm^{-\alpha}$ we now need to use $C\left[(1-\pi)m\right]^{-\alpha}$ as a typical upper bound for the deviations. Since the cases only occur with $(1-\pi)$ probability, the overall bound on the uncertainty is only $Cm^{-\alpha}(1-\pi)^{1-\alpha}$. For $\alpha = \frac{1}{2}$ we see that the latter is somewhat tighter than the rates obtained from the mathematical programming approach.

Note that for $\alpha = 1$, which occurs when the area of probability $\frac{1}{2}$ is small and the problem is basically separable with a margin (see e.g. (Bousquet et al., 2004) for more technical details on the properties that the underlying distribution needs to satisfy), both of the bounds are the same. This means that with sufficient amounts of data we will fairly quickly discover regions which are correctly classified either way. In this case the additional knowledge sets are of limited use.

This reasoning, unfortunately, can only be kept at the informal level. For a more rigorous statement one would need to use a luckiness framework to decide in the case of each single dataset whether one or the other method would be better, as this can only be measured in terms of the choice of the estimate (Herbrich & Williamson, 2002). That said, in our experiments our simpler method excelled.

4. Optimization

We now discuss how the regularized risk functional $R_{\mathrm{reg}}[f, \psi]$ can be minimized efficiently. In general we will need to take recourse to a nonconvex optimization procedure which converges to a local minimum. In some cases, however, the problem remains convex and we obtain a surprisingly simple algorithm.

4.1. Binary Classification

Assume $c(x, y, f(x)) = \max(0, 1 - yf(x))$ and we have prior knowledge in the form of known labels. It is easy to see that in this case $c(x, y, \psi \circ f(x)) = 0$ for all f whenever we *know* the label beforehand. In all other cases, the loss remains unchanged. Consequently we can use the simple Algorithm 1.

Algorithm 1 Knowledge-based SVM Classification

Require: Data pairs (x_i, y_i), ruleset $\mathcal{C}$

1: Remove all (x_i, y_i) pairs for which the rules are sufficient to classify them correctly.
2: Solve a standard SVM classification problem on the remainder to obtain f.
3: **return** $\psi \circ f$.

Algorithm 2 Simpler knowledge-based SVM regression (skSVR) with Constrained Concave Convex Procedure. How to construct $c_{\mathrm{mod}}^{(t-1)}$ given $f^{(t-1)}(x)$ corresponding to $\alpha^{(t-1)}$ is described in Section 4.3.

Require: $f^{(t)}(x) := \sum \alpha_i^{(t)} K(x_i, x) + b$

1: $t \leftarrow 0$; Initialize $\alpha^{(0)}$ and b with random values.
2: **repeat**
3: $t \leftarrow t + 1$
4: Find $\alpha^{(t)}$ that approximately minimizes the convex optimization problem:

$$\min_{\alpha, b} \; C \sum_{i=1}^{m} c_{\mathrm{mod}}^{(t-1)} \left(x_i, y_i, f^{(t)}\right) + \frac{1}{2} {\alpha^{(t)}}^{\top} K \alpha^{(t)}$$

5: Check if any constraint is violated. For each violated constraint use the loss function in Table 1. If the loss is not convex, use first order Taylor expansion to find the upper bound convex loss.
6: **until** convergence

In a nutshell, it means that we minimize the empirical risk for all the data for which no specific rules are available. In a way, this is exactly what we *should* do, as for the remainder the rules will be sufficient to solve the problem on their own. It also means that there is no waste of capacity on complex rules.

4.2. Regression

For other machine learning problems than classification the optimization becomes more complex because the loss function can become either convex or non-convex. In order to cope with the non-convex loss function we need the constrained concave convex optimization procedure (CCCCP) stated as theorems in (Smola et al., 2005; Yuille & Rangarajan, 2003). Consequently we can use the Algorithm 2. The algorithm assumes that by the representer theorem we have $f(x) := \sum \alpha_i K(x_i, x) + b$.

It is worth noting that the CCCCP guarantees that the Algorithm will converge to a local minimum. Any first order or second order optimization methods, such as conjugate gradient descent or Newton's methods, can be used to implement the algorithm.

4.3. Loss Function Decomposition for CCCCP

To find a convex upper bound on the loss function, notice that every function can be decomposed into a sum of a convex and a concave function. In our case, one decomposition is by linearizing the non-convex function at the cut point and adding a piecewise linear concave function to make sure the sum has a constant value from the cut point onwards.

For example, in our Case 3, we can decompose the loss function into a sum of two function c_{v} and c_{c} which are convex and concave functions, respectively

$$c_{\mathrm{v}}(x,y,f) := \begin{cases} c(x,y,f);\ \text{for } f(x) \le u(x) \\ c(x,y,u) + [f(x)-u(x)]c'(x,y,u);\ \text{else} \end{cases}$$

$$c_{\mathrm{c}}(x,y,f) := \begin{cases} 0;\ \text{for } f(x) \le u(x) \\ -[f(x)-u(x)]c'(x,y,u); \text{else} \end{cases}$$

One can easily check that the sum of c_{v} and c_{c} will result in the loss function of Case 3. To applying the CCCCP algorithm above, we approximate the loss functions as follows:

$$c_{\mathrm{mod}}^{(t-1)}(x,y,f^{(t)}) := \begin{cases} c_{\mathrm{v}}(x,y,f^{(t)});\ \text{if } f^{(t-1)}(x) \le u(x) \\ c_{\mathrm{v}}(x,y,f^{(t)}) - c_{\mathrm{v}}'(x,y,u)[f^{(t)}(x)-u(x)]; \text{else} \end{cases}$$

5. Experiments

Incorporating prior knowledge in support vector machines has been studied extensively by Mangasarian and coworkers (*knowledge-based support vector machines*, *KSVM*). In this section we report some experimental results of simpler knowledge-based support vector machines (*skSVM*) developed in this paper.

5.1. Prediction with Prior Knowledge

5.1.1. Binary classification

We carried out numerical test on our algorithm using the Wisconsin prognostic breast cancer (WPBC) dataset[1] used also by Fung et al. (2002a). The preprocessing step was done in the same manner that was described in the paper. We used the standard ν-Support Vector Classifier (ν-SVC) of Schölkopf et al. (2000) as a base method for our knowledge-based support vector classifier. We performed 10-fold cross validation on the preprocessed data. The three parameters ν, ω (Gaussian rbf kernel width), and C are validated on a random subset of the training folds and the best parameters are used to train the final model in each step of the cross validation. Without incorporating any knowledge ν-SVC achieved a mean accuracy of 64.5% and a standard deviation of 16.5%. Incorporating the prior knowledge as described in (Fung et al., 2002a) could improve the mean classification accuracy to 67.3% and the standard deviation to 14.3%. These results are slightly better but essentially equivalent to the results reported by Fung et al. (2002a) who report 66.4% accuracy. However, we note that our classification algorithm is only a minor variant of standard SVMs and can thus easily be applied.

[1] ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machinelearn/cancer/WPBC/

5.1.2. Regression

For illustration we performed regression for odd functions as shown in Figure 2 (left). Furthermore, to compare our results with those of Mangasarian and Wild (2005), we performed regression on data extracted from the Winconsin Prognostic Breast Cancer dataset and described in (Mangasarian & Wild, 2005). In this experiment, the number of metastasized lymph nodes is estimated using only the tumour size. Mangasarian and Wild (2005) performed KSVM experiments with and without background knowledge and achieved root mean squared errors of 5.04 and 6.92, respectively. Incorporating the same background knowledge and performing experiments with skSVM we achieved a leave-one-out root mean squared error of 3.34 with standard deviation of 3.88.

5.2. Demand Estimation

In this section we will introduce a slightly different application scenario for knowledge based SVMs and evaluate it on two datasets, one synthetic dataset and one real-world dataset adapted to the demand estimation scenario.

In many learning problem the observed target values that we are given do not exactly reflect the true values that we would like to estimate. A typical example of this is measurement error due to noise. A slightly non-standard example are bounded target values. This may occur for example if the measurement data unexpectedly exceeds the range of the sensors or in demand estimation scenarios with bounded supply. We will now describe the latter scenario by a simple example. We will try to keep the example as abstract as possible but for a concrete example one could think of the sales of a daily newspaper.

5.2.1. Sales Estimation – Illustration

Consider a dataset reflecting the number of times a particular product (the newspaper) has been sold in a particular shop on a particular day. Clearly, such sales figures are upper bound by the daily supply of the product to the shop. To optimize income we would, however, like to estimate the daily demand for the product at each shop. Figure 2 (middle) can be consulted for a simple illustration. There we assumed a sinusoid demand for the product (thick solid

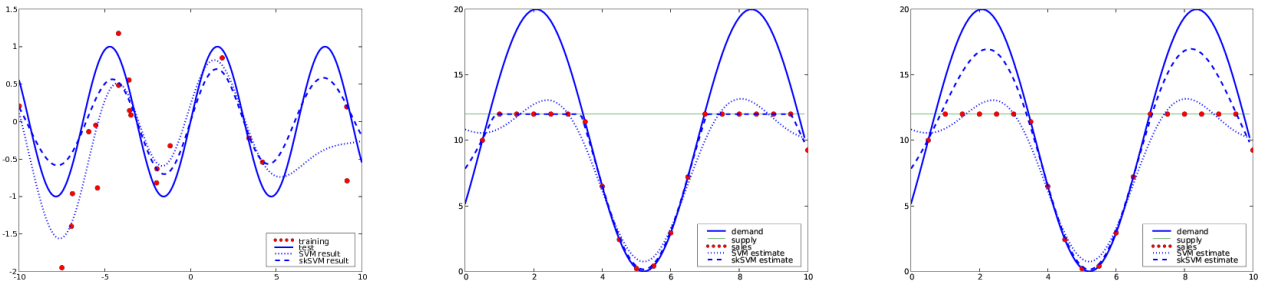

Figure 2. Several illustrations. Left: Regression with the constraint to odd functions; training instances (dots), SVM estimate (dotted curve), and skSVM estimate (dashed curve). Middle: Supply (thin solid line), demand (thick solid curve), sales (dots), and *estimated sales* by SVM (dotted curve) as well as skSVM (dashed curve). Right: Supply (thin solid line), demand (thick solid curve), sales (dots), and *estimated demand* by SVM (dotted curve) as well as skSVM (dashed curve)

curve) and a constant supply of the product. The dots are instances sampled from the sales of the product, i.e., the smaller number of the demand and the supply. We then used SVMs (dotted curve) to estimate the number of sales of the product for the test instances (the whole x-axis) given this sample. For this experiment, σ^2, the width of the Gaussian RBF kernel, is chosen by the usual rules of thumb (Schölkopf & Smola, 2002) and the regularization parameter C is set to 100. Though the estimate is not bad, it has some logical inconsistencies as the estimate is sometimes larger than the supply. Thus we tried to estimate the number of sales of the product using skSVMs (dashed curve) with the constraint that the sales must not be larger than the supply (here 12). Using the same setup and parameters we were then able reduce the RMSE to 0.49 with prior knowledge from 1.08 without prior knowledge.

5.2.2. Demand Estimation – Illustration

A more challenging — and for real-world experiments much more usefull — experiment is, however, how well we can estimate the demand. If we can even slightly improve the fit of the demand, we can reduce companies costs and increase their income significantly. For regular SVMs we have no means of directly estimating the demand given the data described above and we thus have to use the sales as a proxy for the demand. For skSVMs we can view $\psi \circ f$ as an estimate of the sales as above and we can view the underlying function f as an estimate of the demand. We hypothesize that the underlying function f estimated by skSVM is a better estimate of the demand than the estimate by the SVM. Figure 2 (right) can be consulted for a simple illustration. We used the same setup as above but now report the predictions of f rather than $\psi \circ f$ for skSVMs (dashed curve). This way, we were then able to reduce the RMSE to 11.81 with prior knowledge from 27.16 without prior knowledge.

5.2.3. Demand Estimation on Sensory Data

As we were not able to obtain a dataset for demand estimation that can be used freely, we simulated a demand estimation problem on the UCI sensory dataset as follows: We split the dataset into training and a test set. For the training labels, we used the minimum of the given label and a constant threshold (15.7). We did not change the test labels. The parameters σ^2 and C were chosen by cross-validation inside the training set. The RMSE of SVMs is 6.8 and 3.42 for skSVMs.

6. Conclusions and Future work

In this paper we introduced a simple method to incorporate prior knowledge in support vector machines by modifying the hypothesis space rather than the optimization problem. In particular, for any underlying function space $\mathcal{F}$ our hypothesis space is the set of functions $\{\psi \circ f : f \in \mathcal{F}\}$ where ψ maps any $f \in \mathcal{F}$ to a function satisfying the constraints on the whole instance space. Our optimization problem is such that the regularization is on f and the empirical risk is measured using $\psi \circ f$. Our experimental results are naturally limited to publically available datasets and on all datasets we experimented with we achieved some improvement over previous – more complicated – algorithms.

It is because of this novel optimization problem, that we are able to handle a novel application scenario which has — to the best of our knowledge — not been considered before. Whenever measured labels are bounded by a known function cutting the true function that we want to estimate, we can incorporate the known function as background knowledge, choose $\psi \circ f$ that fits the training data well, and use f as an estimate of the true function. A typical application scenario for this is to estimate the daily demand for a product at a shop, given the number of times a particular product (e.g., a daily newspaper) has been sold. Clearly, we

can only measure the sales but want to estimate the demand for the product. As the sales figures are upper bound by the daily supply of the product to the shop (which is usually known), we can apply our simpler knowledge based support vector machine in this application scenario. In future work we will apply the demand estimation prior knowledge scenario to some real world data.

Acknowledgments National ICT Australia is funded through the Australian Government's *Baking Australia's Ability* initiative, in part through the Australian Research Council. This work was supported by grants of the ARC, the Pascal NoE, and the German Science Foundation DFG (WR40/2-2). We thank Olvi Mangsarian, Quan Nguyen, Nic Schraudolph, Tim Sears, and Ted Wild for helpful discussions.

References

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*, 337–404.

Bartlett, P., & Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, *3*, 463–482.

Bousquet, O., Boucheron, S., & Lugosi, G. (2004). Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*. submitted.

DeCoste, D., & Schölkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, *46*, 161–190. Also: Technical Report JPL-MLTR-00-1, Jet Propulsion Laboratory, Pasadena, CA, 2000.

Fung, G., Mangasarian, O., & Shavlik, J. (2000). *Knowledge-based nonlinear kernel classifiers* (Technical Report). Data Mining Institute.

Fung, G., Mangasarian, O., & Shavlik, J. (2002a). Knowledge-based support vector machine classifiers. *Advances in Neural Information Processing Systems 15*. MIT Press.

Fung, G., Mangasarian, O. L., & Smola, A. J. (2002b). Minimal kernel classifiers. *Journal of Machine Learning Research*, *3*, 303–321.

Graepel, T., & Herbrich, R. (2004). Invariant pattern recognition by semidefinite programming machines. *Advances in Neural Information Processing Systems 16*. MIT Press.

Herbrich, R., & Williamson, R. C. (2002). Algorithmic luckiness. *Journal of Machine Learning Research*, *3*, 175–212.

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

Mammen, E., Marron, J., Turlach, B., & Wand, M. (2001). A general projection framework for constrained smoothing. *Statistical Science*, *16*, 232–248.

Mangasarian, O. L. (2000). Generalized support vector machines. *Advances in Large Margin Classifiers* (pp. 135–146). Cambridge, MA: MIT Press.

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

Mangasarian, O. L., & Wild, E. W. (2005). *Nonlinear knowledge in kernel approximation* (Technical Report). Data Mining Institute, University of Wisconsin.

Micchelli, C. A. (1986). Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, *2*, 11–22.

Neal, R. (1996). *Bayesian learning in neural networks*. Springer.

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

Schölkopf, B., Smola, A. J., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation*, *12*, 1207–1245.

Smola, A. J., & Schölkopf, B. (1998). On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, *22*, 211–231.

Smola, A. J., Schölkopf, B., & Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, *11*, 637–649.

Smola, A. J., Vishwanathan, S. V. N., & Hofmann, T. (2005). Kernel methods for missing variables. *Proceedings of International Workshop on Artificial Intelligence and Statistics* (pp. 325–332). Society for Artificial Intelligence and Statistics.

Tipping, M. E. (2000). The relevance vector machine. *Advances in Neural Information Processing Systems 12* (pp. 652–658). Cambridge, MA: MIT Press.

Yuille, A., & Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*, *15*, 915–936.