Data Mining and Knowledge Discovery, Volume 5, Number 3, 2001

Scaling Kernel-Based Systems to Large Data Sets

Volker Tresp
Siemens AG, Corporate Technology
Otto-Hahn-Ring 6, 81730 München, Germany
(volker.tresp@mchp.siemens.de)

Abstract. In the form of the support vector machine and Gaussian processes, kernel-based systems are currently very popular approaches to supervised learning. Unfortunately, the computational load for training kernel-based systems increases drastically with the size of the training data set, such that these systems are not ideal candidates for applications with large data sets. Nevertheless, research in this direction is very active. In this paper, I review some of the current approaches toward scaling kernel-based systems to large data sets.

Keywords: Kernel-based systems, support vector machine, Gaussian processes, committee machines, massive data sets

1. Introduction

Kernel-based systems such as the support vector machine (SVM) and Gaussian processes (GP) are powerful and currently very popular approaches to supervised learning. Kernel-based systems have demonstrated very competitive performance on several applications and data sets. Kernel-based systems also have great potential for KDD applications, since their degrees of freedom grow with training data size, and they are therefore capable of modeling an increasing amount of detail with increasing data set size. Unfortunately, there are at least three problems when one tries to scale up these systems to large data sets. First, training time increases dramatically with the size of the training data set; second, memory requirements increase with data; and third, prediction time is proportional to the number of kernels, which is equal to (or at least increases with) the size of the training data set. A number of researchers have approached these issues and have developed solutions for scaling kernel-based systems to large data sets. I will review these approaches in this paper, where focus will be on the computational complexity of training.

The paper is organized as follows. Section 2 is a brief introduction to kernel-based systems and implementation issues. Section 3 presents various approaches to scaling kernel-based systems to large data sets. First, in Section 3.1, I will present approaches based on the idea of committee machines. Section 3.2 covers data compression approaches. Here, the kernel systems are applied to compressed or sub-sampled versions of the original training data set. Section 3.3 discusses efficient methods for the linear SVM based on modifications of the optimization problem to be solved. Some training regimes for kernel-based systems require the solution of a system of linear equations the size of the training data set. Section 3.4 presents various approaches to efficiently solving this system of equations. In Section 3.5, projection methods are discussed. The solutions here are based on a finite-dimensional approximation to the kernel system. Section 4 provides conclusions.

2. Kernel-Based Systems for Supervised Learning

2.1. Kernel-based System

In kernel-based systems, the response $\hat{f}(x)$ to an input x is calculated as a superposition of kernel functions $K(x, x_i)$ as

$$\hat{f}(x) = \sum_{x_i \in \mathcal{D}_x} w_i K(x, x_i). \qquad (1)$$

Here, w_i is the weight on the i-th kernel. Kernels are either defined for all input patterns $\mathcal{D}_x = \{x_i\}_{i=1}^N$ or for a subset of the training data, as in the case of the SVM. Sometimes a bias term b is included, and

$$\hat{f}^b(x) = \sum_{x_i \in \mathcal{D}_x} w_i K(x, x_i) + b. \qquad (2)$$

In regression, $\hat{f}(x)$ (respectively $\hat{f}^b(x)$) corresponds to an estimate of the regression function, and in (binary) SVM classification, the estimated class label is the sign of $\hat{f}(x)$ (respectively $\hat{f}^b(x)$). In some cases, the kernel functions $K(x, x_i)$ are required to be positive definite functions. Typical examples are linear kernels $K(x, x_i) = x'x_i$, polynomial kernels $K(x, x_i) = (1 + x'x_i)^q$ with positive integer q, and Gaussian kernels $K(x, x_i) = \exp(-\frac{1}{2s^2}||x - x_i||^2)$ with scale parameter s. Linear kernels are a special case, since here the equivalent linear representation

$$\hat{f}^b(x) = v'x + b \qquad (3)$$

with weight vector $v = A'w$ is computationally more efficient if the input dimension is smaller than the size of the training data set. The rows of the design matrix A are the input vectors of the training data set. Kernel-based systems differ in how the weights w_i are determined based on the training data.

2.2. Gaussian Process Regression (GPR)

In GPR, one assumes *a priori* that a function $f(x)$ is generated from an infinite-dimensional Gaussian distribution with zero mean and covariance $cov(f(x_i), f(x_j))$ defined at input points x_i and x_j. Furthermore, we assume a set of training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where targets are generated according to

$$y_i = f(x_i) + \epsilon_i$$

where ϵ_i is independent additive Gaussian noise with variance σ^2. The optimal regression function $\hat{f}(x)$ takes on the form of Equation 1, where the kernel

$$K(x, x_i) \equiv cov(f(x), f(x_i))$$

is determined by the covariance function. Based on our assumptions, the maximum a posterior (MAP) solution for $w = (w_1, \dots, w_N)'$ minimizes the cost function

$$\frac{1}{2} w' \Sigma w + \frac{1}{2\sigma^2} (\Sigma w - y)' (\Sigma w - y) \tag{4}$$

where Σ, with $(\Sigma)_{i,j} = cov(f(x_i), f(x_j))$ is the $N \times N$ Gram matrix.[1] The optimal weight vector is the solution to a system of linear equations, which in matrix form becomes

$$(\Sigma + \sigma^2 I) w = y. \tag{5}$$

Here $y = (y_1, \dots, y_N)'$ is the vector of targets and I is the $N \times N$-dimensional unit matrix. MacKay (1997) provides an excellent introduction to Gaussian processes.

2.3. Generalized Gaussian Process Regression (GGPR)

Gaussian processes can find a wider range of applications by permitting a more flexible measurement process. In particular, we might assume that $y_i \in \{0, 1\}$ is the class label of the i-th pattern and that

$$P(Y_i = 1 | x_i) = \frac{1}{1 + \exp(-f(x_i))}$$

is the posterior probability for Class 1. The corresponding cost function is

$$\frac{1}{2} w' \Sigma w - \sum_{i=1}^N \log P(Y_i = y_i | x_i).$$

[1] Let $f^m = \Sigma w$ be the the values of f at the location of the training data. Using this identity, the cost function can also be written as

$$\frac{1}{2} (f^m)' \Sigma^{-1} f^m + \frac{1}{2\sigma^2} (f^m - y)' (f^m - y).$$

In this case, the optimal weights are found iteratively using an iterated re-weighted least squares algorithm. The approach can be generalized to any density from the exponential family of distributions. Discussion of GGPR can be found in Tresp (2000b), Williams and Barber (1998), and Fahrmeir and Tutz (1994).

2.4. Support Vector Machine (SVM)

The SVM classifies a pattern according to the sign of Equation 2. The most commonly used SVM cost function is

$$\frac{1}{2}w'\Sigma w + Ce'(e - D(\Sigma w + be))_+. \qquad (6)$$

Here, D is an $N \times N$-dimensional diagonal matrix with ones and minus ones on the diagonal (corresponding to the class label of the respective patterns), e is an N-dimensional vector of ones, $C \geq 0$ is a constant and $()_+$ sets all negative components of the vector within the bracket to zero. Note the similarity between the SVM and the GGPR cost function. The difference is that the measurement process for the SVM is not derived from the exponential family of distributions as it is for the GGPR.

More common is the following equivalent definition with a differentiable cost function and constraints. The weights in the SVM minimize the cost function

$$\frac{1}{2}w'\Sigma w + Ce'\xi$$

where $\xi = (\xi_1, \ldots, \xi_N)'$ are the slack variables with $\xi_i \geq 0$ subject to the constraint that

$$D(\Sigma w + be) \geq e - \xi.$$

Here and in the rest of the paper $\geq$ is meant to be applied componentwise. By introducing an N-dimensional vector of Lagrange multipliers α for enforcing the constraints, one obtains the dual problem:
Maximize

$$W(\alpha) = e'\alpha - \frac{1}{2}\alpha' D\Sigma D\alpha$$

with the constraints

$$0 \leq \alpha \leq C \qquad \alpha' y = 0.$$

The weights w and the Lagrange multipliers α are related via $w = D\alpha$.

The optimal weight vector is sparse — that is the weight w_i for many kernels is zero after training. The remaining kernels with nonzero weights define the support vectors. It can be shown that the solution minimizes a bound on the generalization error (Vapnik, 1998). Vapnik

(1998), Schölkopf, Burges and Smola (1999), Christianini and Shawe-Taylor (2000), and Müller, Mika, Rätsch, Tsuda, and Schölkopf (2001) are excellent sources of information about the SVM.

2.5. Training

Solving the system of linear equations (Equation 5) for GPR requires $\mathcal{O}(N^3)$ operations. The iterated re-weighted least squares algorithm used for training the GGPR typically requires that systems of similar linear equations must be solved repeatedly.

For the support vector machine, the dual optimization problem requires the solution of a quadratic programming (QP) problem with linear constraints. Due to the large dimension of the optimization problem, general QP routines are unsuitable even for relatively small problems (a few hundred data points): the common interior point methods require repeated solution of linear equations with the dimensionality of the number of variables.

To solve the high-dimensional QP problems for the SVM, special algorithms have been developed. The most important ones are *chunking*, *decomposition*, and *sequential minimal optimization* (SMO). All methods iteratively solve smaller QP problems. Both chunking and decomposition require a QP solver in their inner loops. Decomposition scales better with the size of the training data set since the dimensionality of the QP problem is fixed, whereas for the chunking algorithm, this dimensionality increases until it reaches the number of support vectors. Chunking was one of the earliest approaches used for optimizing the SVM machine. Decomposition was introduced by Osuna, Freund, and Girosi (1997) and further developed by Joachims (1998). The SMO is an extreme case of a decomposition algorithm since only two variables (Lagrange multipliers) are optimized at a time. This can be done analytically, so that the SMO does not require a QP optimizer in its inner loop. Decomposition and the SMO are in general faster than chunking. Experimental evidence suggests that the computational complexity for both decomposition and the SMO scales approximately to the square of size of the training data set.

As an alternative to QP, several gradient optimization routines have been developed. In addition, Pérez-Cruz, Alarcón-Diana, Navia-Vázquez, and Artés-Rodríguez (2001) used a fast iterated re-weighted least squares procedure recently to train the SVM.

2.6. More Kernels: Regularization Networks, Smoothing Splines, the Relevance Vector Machine and Kernel Fisher Discriminant

It is not possible to discuss all currently popular kernel-based learning systems in detail in the limited space allowed here. The regularization networks of Poggio and Girosi (1990) are essentially identical to GPR. Here, the kernels are Green's functions derived from the appropriate regularization problem. Similarly, smoothing splines are closely related (Wahba, 1990). The relevance vector machine (Tipping, 2000) achieves sparseness by pruning away dimensions of the weight vector using an evidence framework. Finally, the kernel Fisher discriminant is the well-known linear Fisher discriminant approach transformed into a high-dimensional feature space by means of kernel functions (Müller et al., 2001).

3. Scaling Kernel-Based Systems to Large Data Sets

This section is the heart of the paper. In it, I discuss various approaches toward scaling kernel-based systems to large data sets.

3.1. Committee Machines

3.1.1. *Introduction*

In committee machines, different data sets are assigned to different committee members (i.e., kernel systems) for training, and the predictions of the committee members are combined to form the prediction of the committee. Here, I want to discuss two committee approaches that are particularly suitable for (and have been applied to) kernel-based systems.

3.1.2. *A Bayesian Committee Machine (BCM)*

In the BCM approach (Tresp, 2000a), the data are partitioned into M data sets $\mathcal{D} = \{\mathcal{D}^1, \ldots, \mathcal{D}^M\}$ (of approximately same size) and M learning systems are trained on their respective training data set. The BCM calculates the unknown responses at a number of test points at the same time. Let $f^q = (f_1^q, \ldots, f_{N_Q}^q)$ be the vector of these response variables at the N_Q test points.

The underlying assumption of the BCM is that

$$P(\mathcal{D}^i | f^q, \mathcal{D} \setminus \mathcal{D}^i) \approx P(\mathcal{D}^i | f^q) \quad i = 1, \ldots, M. \tag{7}$$

Under this assumption, data sets are independent given f^q. This is a good assumption if f^q contains many points, since these points define the map and make all data independent. The approximation also improves if the number of data in each set $\mathcal{D}^i$ is large, which increases their independence from the other data sets on average.

Based on the assumption, one obtains

$$P(f^q|\mathcal{D}) \approx \frac{P(f^q)\prod_{i=1}^{M} P(\mathcal{D}^i|f^q)}{P(\mathcal{D})}.$$

A practical algorithm is obtained if we assume that all probability distributions are approximately Gaussian. Then $P(f^q|\mathcal{D})$ is also approximately Gaussian with

$$\hat{E}(f^q|\mathcal{D}) = C^{-1} \sum_{i=1}^{M} cov(f^q|\mathcal{D}^i)^{-1} E(f^q|\mathcal{D}^i) \qquad (8)$$

and

$$C = \widehat{cov}(f^q|\mathcal{D})^{-1} = -(M-1)(\Sigma^{qq})^{-1} + \sum_{i=1}^{M} cov(f^q|\mathcal{D}^i)^{-1}. \qquad (9)$$

Here, Σ^{qq} is the $N_Q \times N_Q$ prior covariance matrix at the test points. Equation 8 has the form of a committee machine where the predictions of the committee members at all N_Q inputs are used to form the prediction of the committee at those inputs. The prediction of each module i is weighted by the inverse covariance of its prediction. An intuitive appealing effect is that by weighting the predictions of the committee members by the inverse covariance, modules that are uncertain about their predictions are automatically weighted less than modules that are certain about their predictions.

I applied the BCM to GPR in Tresp (2000). For GPR, the mean and the covariance of the posterior Gaussian densities are readily computed. Subsequently, the BCM was applied to GGPR (Tresp, 2000b) and to the SVM (Schwaighofer and Tresp, 2001). For both the GGPR and the SVM, the posterior distributions are only approximately Gaussian. In Tresp (2000), it was shown that N_Q, the dimension of f^q, should be at least as large as the effective number of parameters.[2] If this is the case, the BCM and its generalizations give excellent results. Furthermore, it is possible to derive online Kalman filter versions of the BCM, which only require one pass through the data set and the storage of a matrix of the dimension of the number of test points (Tresp, 2000a). After training, the prediction at additional test points requires resources dependent on the number of test points but is independent

[2] Since the latter is closely related to the kernel bandwidth, there is also a close connection between the kernel bandwidth and N_Q.

of the size of the training data set. On several data sets I found no difference in performance between the BCM approximation and the optimal GPR prediction based on the inversion of the full covariance matrix. The BCM was applied to training data sets of size $N = 60,000$, where the full inversion is clearly unfeasible.

3.1.3. *Boosting*

In boosting, committee members are trained sequentially and the training of a particular committee member is dependent on the training and the performance of previously trained members. Boosting can reduce both variance *and bias* in the prediction. The reason is that in training a committee member, more weight is put on data that are misclassified by previously trained committee members.

Schapire (1990) developed the original boosting approach, *boosting by filtering*. Here, three learning systems are used and the existence of an oracle that can produce an arbitrary quantity of training data is assumed. The first learning system is trained on K training data. Then the second learning system is trained on the same quantity of data but these training data are generated so that half of them are classified correctly and half of them are classified incorrectly by the first learning system. The third learning system is trained only on data about which learning systems one and two disagreed. A majority vote of the three learning systems determines the classification. Note that the second learning system obtains 50% of patterns for training that are difficult for the first learning system, and that the third learning system only obtains critical patterns in the sense that learning the first and second learning systems disagree on those patterns. This original boosting algorithm is particularly useful for large data sets, since a large number of training data are filtered out and are not directly used for training. In recent years, the focus of interest has shifted toward boosting algorithms that can also be applied to smaller data sets: boosting by resampling and boosting by reweighting. Recently, Pavlov, Mao, and Dom (2000) used a form of boosting by resampling, Boost-SMO, in the context of training the SVM with large data sets. They used a small fraction of data ($2 - 4\%$) to train a committee member using the SMO algorithm. To train a subsequent committee member, they chose those data with a higher probability which were difficult for the previous committee member to classify. In this way, only a portion of the complete training data set is used for training. The overall prediction is then a weighted combination of the predictions of the committee members. In some experiments, Boost-SMO was faster by a factor of 10 than SMO while providing essentially the same prediction accuracy. In their implementation, the probability of the data are reweighted prior to the training of a new committee member, so multiple passes through the data are required. Nevertheless, online versions of this approach are also conceivable.

3.2. Data Compression

Data compression is a generally applicable solution to dealing with large data sets. The idea is to train the learning system on a smaller data set that is either generated by subsampling or by preclustering the data. In the latter case, the cluster centers are used as training patterns. The idea of preclustering data has been extended in an interesting direction by applying the concept of *squashing* to training a linear SVM (Pavlov, Chudova, and Smyth, 2000). For training, the SMO algorithm is used. Clustering is performed using a metric derived from the likelihood profile of the data. First, a small percentage of the original training data set are randomly chosen as cluster centers. Then, for a linear SVM, L (typically 50–100) random weights v and an offset b are generated following their prior distributions (a probabilistic version of the SVM is used). For each data point the log-likelihood for each weight vector is calculated, producing an L-dimensional vector (likelihood profile) for each data point. A data point is then assigned to the cluster center with the closest likelihood profile. Finally, the weight for each cluster center is proportional to the number of data assigned to a cluster. This procedure leads to a considerable reduction in training data by taking into account the statistical properties of the data. The training time using the SMO with squashing was comparable to the training time of Boost-SMO (see previous section) by providing a comparable prediction accuracy.

3.3. Fast Algorithms for Linear SVMs

The SVM was originally formulated as a linear classifier; kernels were introduced in order to be able to obtain nonlinear classification boundaries. Training the linear SVM is considerably faster than training the kernel SVM. The following section discusses approaches that lead to even faster training algorithms for the linear SVM by modifying the cost function.

3.3.1. *Active Support Vector Machine (ASVM)*

The ASVM was developed by Mangasarian and Musicant (2001). Here, the optimization problem of the SVM is reformulated. The modified cost function is

$$\frac{1}{2}(v'v + b^2) + C\xi'\xi$$

with constraints

$$D(Av - be) \geq e - \xi$$

where A is the design matrix. In the design matrix, the input vectors of the training data set are represented as rows. Note that the cost function contains the square of the bias b such that the margins with respect to both orientation v and location relative to the origin b are maximized. Furthermore, the 2-norm of the slack vector is minimized. Based on these modifications, a dual problem is formulated that contains non-negativity constraints but no equality constraints. Due to these modifications, a very simple iterative optimization algorithm was derived. At each iteration step, a system of linear equations the size of the input dimension plus one needs to be solved. The number of iteration steps is finite. As an example, a data set with 7 million points required only 5 iterations and needed 95 CPU minutes.

3.3.2. *Lagrange Support Vector Machine (LSVM)*

A variation on the ASVM is the LSVM (Mangasarian and Musicant, 2000). It is based on the same reformulation of the optimization problem and leads to the same dual problem. The difference is that the LSVM works directly with the Karush-Kuhn-Tucker necessary and sufficient optimality condition for the dual problem. The algorithm requires, prior to the optimization iterations, the inversion of one matrix Q of the size of the input dimension plus one. The iteration has the simple form

$$\alpha^{i+1} = Q^{-1}(e + ((Q\alpha^i - e) - \beta\alpha^i)_+)$$

where α^i is the vector of Lagrange multipliers at step i, and β is a positive constant.

The LSVM and the ASVM are comparable in speed although the ASVM is faster on some problems. The great advantage of the LSVM is definitely the simplicity of the algorithm. The LSVM can also be applied to nonlinear kernels, but a matrix of the size of the number of data points needs to be inverted. For more examples of fast linear SVM-variants, see Mangasarian and Musicant (1999).

3.4. Approximate Solutions to Systems of Linear Equations

Gaussian processes and some variants of the SVM (see Section 3.3.2) require the solution of a large system of linear equations.

3.4.1. *Method by Skilling*

For Gaussian processes and also for some variants of the SVM, it is necessary to solve a linear system of equation of the form

$$Mw = y$$

with some matrix M. The solution to this linear set of equations is identical to the minimum of

$$w'y - \frac{1}{2}w'Mw.$$

This cost function is minimized iteratively using the conjugate gradient method in $\mathcal{O}(k \times N^2)$ steps where k is the number of iterations of the conjugate gradient procedure. Often k can be set to be much smaller than N without a significant loss in performance, particularly when M has a large number of small eigenvalues. This approach is due to Skilling and was one of the earliest approaches to speeding up the training of GPR systems (Gibbs and MacKay, 1997). Note that the computational complexity of this approach is quadratic in the size of training data set and this approach is therefore not well suited for massive data sets.

3.4.2. *The Nyström Method*

The Nyström method was introduced by Williams and Seeger (2001), and it is applicable in particular to GPR. Let's assume a decomposition of the Gram matrix of the form $\Sigma = U\Lambda U'$ where Λ is a diagonal matrix and U is an $N \times m$ matrix with typically $m << N$. In this case, we can solve the system of linear equations (see Equation 5) using the Woodbury formula (Press, Teukolsky, Vetterling, and Flannery, 1992) and obtain

$$w = \frac{1}{\sigma^2}y - \frac{1}{\sigma^2}U(\sigma^2\Lambda^{-1} + U'U)^{-1}U'y.$$

In this form, only a matrix of size $m \times m$ needs to be inverted, and w still has the dimension of the number of training data N.

An example of an appropriate decomposition of the Gram matrix would be an eigenvalue decomposition. The computational complexity of a full eigenvalue decomposition scales as $\mathcal{O}(N^3)$, so not much is gained unless the Gram matrix has a large number of small eigenvalues. A particularly interesting approximation was introduced by (Williams and Seeger, 2001). They performed an eigendecomposition of a (randomly chosen) $m \times m$ submatrix of Σ. Based on p eigenvectors and eigenvalues of the decomposition of the smaller matrix, the corresponding decomposition of the larger matrix can be approximated using the Nyström method. The Nyström method is a method for numerically solving integral equations. The computational complexity of this approach is $\mathcal{O}(m^2N)$ such that this approach scales linear in N. The authors verified the excellent quality of this approximate method using a training set size of 7291 and obtained very good results with $m = p = 256$. The approach is applicable to both regression and classification.

3.5. Projection Methods Leading to a Finite-Dimensional Representation

In general, the degrees of freedom of a kernel system grow with the number of kernels (i.e., data points). The approaches discussed in this section project the basically infinite-dimensional problem on a finite-dimensional representation. The problem then reduces to estimating the parameters in this finite-dimensional problem. In all approaches, the solution assumes the form of a system with fixed basis functions. The BCM approach in Section 3.1.2 can also be considered to be a specific projection approach in which the basis functions are the kernels defined by the test points.

3.5.1. *Optimal Projections*

The problem formulation is: Given that both the input data distributions $P(x)$ and the size of training data N are known, what is the best linear combination of an m-dimensional set of basis functions that provides the best approximation to a GPR kernel-based system. In projected Bayes regression (Zhu, Williams, Rohwer, Morciniec, 1998), this problem is solved based on an infinite-dimensional principle component analysis for $N \rightarrow \infty$. The result is that one should use the first m eigenfunctions of the covariance function describing the Gaussian process asymptotically. The computational complexity is $\mathcal{O}(Nm^2)$.

Trecate, Williams, and Opper (1998) described an improved variational approximation for finite data size. Their approach has a smaller test set error if compared to projected Bayes regression, and the computational complexity scales as $\mathcal{O}(N^2 m)$. This approach is quadratic in N, whereas projected Bayes regression is linear in N, because the representation increases with data size. Csató and Opper (2001) presented an online variant based on these ideas, which leads to a sparse representation by limiting the representation to a small number of well chosen kernel functions.

3.5.2. *Projection on a Subset of Kernels*

Let's select a subset of the training data set of size $m < N$ and let $\Sigma^{m,m}$ denote the corresponding kernel matrix. We can now approximate

$$cov(f(x_i), f(x_j)) \approx (K^m(x_i))'(\Sigma^{m,m})^{-1} K^m(x_j)$$

where $K^m(x_i)$ is the vector of covariances between the functional values at x_i and the data in the subset. This approximation is an equality if either x_i or x_j are elements of the subset and is an approximation otherwise. With this approximation, the regression function is a superposition

$$\hat{f}(x) = \sum_{i=1}^{m} w_i K(x, x_i) \tag{10}$$

of only m kernel functions and the optimal weight vector minimizes the cost function

$$\frac{1}{2} w' \Sigma^{m,m} w + \frac{1}{2\sigma^2} (\Sigma^{N,m} w - y)' (\Sigma^{N,m} w - y)$$

where $\Sigma^{N,m}$ contains the covariance terms between all N training data and the subset of data chosen to be kernel functions.

Although we only used a fixed number of kernels, all training data contributed to determine the weight vector w. The methods described in the following subsections are based on this decomposition. The connection between the decomposition of the Gram matrix in this section and the BCM approximation is discussed in the Appendix.

3.5.3. *Reduced Support Vector Machines (RSVM)*

The RSVM (Lee and Mangasarian, 2000) uses a nonstandard SVM cost function of the form

$$\frac{1}{2}(w'w + b^2) + C g' g.$$

If we compare this equation with the original SVM cost function of Equation 6, we notice that the cost term for the weights is simplified. As in the previous section, w denotes the kernel weights for m randomly selected kernels and, as in the case of the ASVM (Section 3.3.1), the square of the bias b is included. Here, the 2-norm of a transformed slack variable is included with

$$g = g(e - D(\Sigma^{N,m} w - be)),$$

where $\Sigma^{N,m}$ is defined as in the previous subsection and where $g(x) = x + \frac{1}{\gamma} \log(1 + \exp(-\gamma x))$ is applied component-wise. The function $g()$ is a "soft" twice-differentiable version of $()_+$ where γ is typically a large positive number. The advantages of these modifications are that (a) no constraints are needed (as in the formulation in Equation 6); (b) due to the modifications of the cost function, a quadratically converging Newton algorithm can be used for training; and (c) due to the projection on the finite number of kernels, the time complexity of the optimization routine scales linearly in the number of data points. In an experiment using the adult data set[3] with N=32600 training data and $m = N/100 = 326$, the RSVM needed 16 minutes, whereas the SMO algorithm needed more than two hours. Surprisingly, on some smaller data sets the RSVM performed even better than the full SVM. This was explained by a smaller tendency toward overfitting of the RSVM. Experimental results showed that the random selection of the training data did not significantly increase the variance in the prediction.

3.5.4. *Sparse Greedy Matrix Approximation*

In the previous two subsections an expansion in terms of a finite number of kernels defined at a random subset of the training data was sought. In contrast, in the BCM approximation the kernels are defined by the test points. In Smola and Schölkopf (2000), the expansion is based on a subset of the training data, but here, the kernels are not selected randomly: The goal is to find the m kernels that can best represent the N kernels of all training data where proximity is defined in the Reproducing Kernel Hilbert Space (i.e., the feature space). This simplifies calculations drastically since the inner product of two kernels defined for x_i and x_j is simply equal to $K(x_i, x_j)$. The authors describe a "greedy" algorithm in which at each step the best kernel out of randomly selected L candidate kernels is added to the set of already selected kernels. The computational cost of this version is $\mathcal{O}(L \times m \times N^2)$, where N is the total number of training data and m is the size of the selected subset of training data. In Smola and Schölkopf's paper, $L = 59$ was motivated by theoretical considerations. The authors showed that the approximation based on m kernels is very close to the approximation based on the optimal m basis vectors. The optimal m should also be closely related to the effective number of parameters, as discussed in Section 3.1.2, and is therefore dependent on the kernel bandwidth. A variant of this approach applicable to GPR is described in Smola and Bartlett (2001). In their paper, they derived a stopping criterion and bounds on the approximation error. Using a training data set of size $N = 4,000$ the authors demonstrated that, with less than 10% of the training data used as kernels, they obtained statistically insignificant differences in performance between GPR based on the inversion of the full covariance matrix and their approximation.

4. Conclusions

I have summarized the most important approaches for scaling up kernel-based systems to large data sets. For nonlinear kernels, various authors

[3] Retrievable from: http://www.ics.uci.edu/ mlearn.

have achieved a considerable reduction in training time which makes nonlinear kernel systems applicable to data sets of maybe a few 100,000 data points. All approaches assume that a representation based on a finite subset of the training data (respectively the kernels) is sufficient which is a reasonable assumption if the kernel bandwidth is low. Nevertheless, the goal stated in the introduction, of being able to model an increasing amount of detail when sufficient data become available would require that the kernel bandwidth be scaled up with increasing data size, thus increasing the degrees of freedom appropriately. This is only possible to a limited degree with the methods presented. Combining kernel-based systems with a hierarchical partitioning of the map or local algorithms might be an interesting direction for future research.

Finally, for kernel systems using linear kernels, training time is considerably faster than for kernel systems using nonlinear kernels. Here, systems with more than a few million data points have been trained witin a reasonable time.

Appendix

Let's assume that one is interested in the prediction at a set of query points, defined as a subset of the training data as in Section 3.5.2 or as the test data, as in Section 3.1.2. Let f^q denote the unknown functional values at those query points. Let $f^q = \Sigma^{q,q} w$ be the expansion in terms of the kernel functions defined at the query points. Here, $\Sigma^{q,q}$ is the covariance matrix defined for this subset of points. Then, the weight vector w, which leads to the optimal predictions at the subset of points minimizes

$$\frac{1}{2} w' \Sigma^{q,q} w + \frac{1}{2} (\Sigma^{N,q} w - y)' \; cov(y|f^q)^{-1} \; (\Sigma^{N,q} w - y)$$

Here,

$$cov(y|f^q) = \sigma^2 I + \Sigma^{N,N} - \Sigma^{N,q} (\Sigma^{q,q})^{-1} (\Sigma^{N,q})' \qquad (11)$$

is the covariance of the training data given f^q. Also, $\Sigma^{N,q}$ is the covariance between the training data and the query points and $\Sigma^{N,N}$ is the covariance matrix defined for the training data.

Note that the inverse of $cov(y|f^q)$ needs to be calculated which is an $N \times N$ matrix. The approximation used in Section 3.5.2 simply sets $cov(y|f^q) = \sigma^2 I$ where I is the unit matrix. The BCM uses a block diagonal approximation of $cov(y|f^q)$, and the calculation of the optimal weight vector w requires the inversion of matrices of the block size. The BCM approximation improves if few blocks are used (then a smaller number of elements are set zero) and when the dimension of f^q is large, since then the two terms on the right side of Equation 11 cancel and $cov(y|f^q) = \sigma^2 I$. A more detailed discussion can be found in Tresp and Schwaighofer (2001).

Acknowledgements

Salvatore Ingrassia from the Università della Calabria and Stefano Ricci from the Università di Pavia provided me with extensive comments on an earlier version of this paper. Their comments helped improve the paper considerably. In addition, valuable discussions with Alex Smola, Chris Williams, John Platt, Lehel Csató, and Anton Schwaighofer are gratefully acknowledged.

References

Christianini, N., and Shawe-Taylor, J. (2000). *Support vector machines.* Cambridge: Cambridge University Press.

Csató, L. E., and Opper, M. (2001). Sparse representation for Gaussian process models. In T. K. Leen, T. G. Diettrich, and V. Tresp, (Eds.), *Advances in Neural Information Processing Systems, 13.*

Fahrmeir, L., and Tutz, G. (1994). *Multivariate statistical modeling based on generalized linear models.* Berlin: Springer.

Joachims, T. (1998). Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, (Eds.), *Advances in kernel methods.* Cambridge: MIT press.

Gibbs, M., and MacKay, D. J. C. (1997). *Efficient implementation of Gaussian processes.* (Technical Report, available from http://wol.ra.phy.cam.ac.uk/mackay/homepage.html).

Lee, Y.-J., and Mangasarian, O. L. (2000). *RSVM: reduced support vector machines.* (Data Mining Institute Technical Report 00–07). Computer Sciences Department, University of Wisconsin.

MacKay, D. J. C. (1997). Introduction to Gaussian processes. In Bishop, C., M., (Ed.), *Neural networks and machine learning, NATO Asi Series. Series F, Computer and Systems Science* (Vol 168).

Mangasarian, O. L., and Musicant, D. R. (1999). *Massive support vector regression. (Data Mining Institute Technical Report 99–01).* Computer Sciences Department, University of Wisconsin.

Mangasarian, O. L., and Musicant, D. R. (2000). Lagrangian support vector machine. *(Data Mining Institute Technical Report 00–06).* Computer Sciences Department, University of Wisconsin.

Mangasarian, O. L., and Musicant, D. R. (2001). Active support vector machine classification. In T. K. Leen, T. G. Diettrich, and V. Tresp, (Eds.), *Advances in Neural Information Processing Systems, 13.*

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, T., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks, 12.*

Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In J. Principe, L. Giles, N. Morgan, and E. Wilson (Eds.) *Neural Networks for Signal Processing VII — Proceedings of 1997 IEEE Workshop*. New York: IEEE.

Pavlov, D., Mao, J., and Dom, B. (2000). Scaling support vector machines using boosting algorithm. *Proceedings of the ICPR.*

Pavlov, D., Chudova, D., and Smyth, P. (2000). Towards scalable support vector machines using squashing. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-2000.* New York: The Association of Computing Machinery.

Pérez-Cruz, F., Alarcón-Diana, P. L., Navia-Vázquez, A., and Artés-Rodríguez, A. (2001). Fast training of support vector classifiers. In T. K. Leen, T. G. Diettrich, and V. Tresp, (Eds.), *Advances in Neural Information Processing Systems, 13.*

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, (Eds.), *Advances in kernel methods.* Cambridge: MIT Press.

Poggio, T., and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE, 78.*

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical recipes in C.* Cambridge: Cambridge University Press.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5,* 197.

Schölkopf, B., Burges, C. J. C., and Smola, A. J. (1999). *Advances in kernel methods.* Location: MIT Press.

Schwaighofer, A., and Tresp, V. (2001). The Bayesian committee support vector machine. Proceedings of the Eleventh International Conference on Artificial Neural Networks, ICANN 2001.

Smola, A. J., and Schölkopf, B. (2000). Sparse greedy matrix approximations for machine learning. In P. Langley, (Ed.), *Proceedings of the 17th International Conference on Machine Learning.*

Smola, A. J., and Bartlett, P. (2001). Sparse greedy Gaussian process regression. In T. K. Leen, T. G. Diettrich, and V. Tresp, (Eds.), *Advances in Neural Information Processing Systems, 13.*

Tipping, M. E. (2000). The relevance vector machine. In S. A. Solla, T. K. Leen, and K.-R. Müller, (Eds.), *Advances in Neural Information Processing Systems, 12.*

Trecate, G. F., Williams, C. K. I., and Opper, M. (1998). Finite-dimensional approximations of Gaussian processes. In M. J. Kearns, S. A. Solla, and D. A. Cohn, (Eds.), *Advances in Neural Information Processing Systems, 11.*

Tresp, V. (2000a). The Bayesian committee machine. *Neural Computation, Vol.12.*

Tresp, V. (2000b). The generalized Bayesian committee machine. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-2000.* New York: The Association of Computing Machinery.

Tresp, V., and Schwaighofer, A. (2001). Scalable kernel systems. Proceedings of the Eleventh International Conference on Artificial Neural Networks, ICANN 2001.

Vapnik, V. N. (1998). *Statistical learning theory.* New York: John Wiley & Sons.

Wahba, G. (1990). *Spline models for observational data.* Philadelphia: Society for Industrial and Applied Mathematics.

Williams, C. K. I., and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20* (12).

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Diettrich, and V. Tresp, (Eds.), *Advances in Neural Information Processing Systems, 13.*

Zhu, H., Williams, C. K. I., Rohwer, R., and Morciniec, M. (1998). Gaussian regression and optimal finite dimensional linear models. In C. M. Bishop, (Ed.), *Neural networks and machine learning, NATO Asi Series. Series F, Computer and Systems Sciences,* (Vol 168).