Fast n-Fold Cross-Validation for Regularized Least-Squares

Tapio Pahikkala[⋆] Jorma Boberg[⋆]
Tapio Salakoski[⋆]
[⋆]Turku Centre for Computer Science (TUCS), Department of Information Technology, University of Turku
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
firstname.lastname@it.utu.fi

Abstract

Kernel-based learning algorithms have recently become the state-of-the-art machine learning methods of which the support vector machines are the most popular ones. Regularized least-squares (RLS), another kernel-based learning algorithm that is also known as the least-squares support vector machine, is shown to have a performance comparable to that of the support vector machines in several machine learning tasks. In small scale problems, RLS have several computational advantages as compared to the support vector machines. Firstly, it is possible to calculate the cross-validation (CV) performance of RLS on the training data without retraining in each CV round. We give a formal proof for this claim. Secondly, we can compute the RLS solution for several different values of the regularization parameter in parallel. Finally, several problems on the same data set can be solved in parallel provided that the same kernel function is used with each problem. We consider a simple implementation of the RLS algorithm for the small scale machine learning problems that takes advantage of all the above properties. The implementation is done via the eigen decomposition of the kernel matrix. The proposed CV method for RLS is a generalization of the fast leave-one-out cross-validation (LOOCV) method for RLS which is widely known in the literature. For some tasks, the LOOCV gives a poor performance estimate for the learning machines, because of the dependencies between the training data points. We demonstrate this by experimentally comparing the performance estimates given by LOOCV and CV in a ranking task of dependency parses generated from biomedical texts.

1 Introduction

Kernel-based learning algorithms (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) have recently become the state-of-the art machine learning methods of which the support vector machines are the most popular ones. In this paper, we consider Regularized least-squares (RLS) algorithm (see e.g. Rifkin (2002); Poggio and Smale (2003)), another kernel-based learning algorithm that is also known as the least-squares support vector machine (Suykens and Vandewalle, 1999). They were shown to have a performance comparable to that of the support vector machines in several machine learning tasks. Traditionally, RLS type of algorithms have been applied to regression problems but lately they have also been used on other machine learning problems. Recent classification tasks in which RLS have been successfully applied are, for example, disambiguation problems in natural language (Popescu, 2004), DNA classification (Ancona et al., 2005), and classification of intensive care nursing narratives (Hiissa et al., 2006). Another successful application area has been ranking or ordinal regression (Tsivtsivadze et al., 2005, 2006; Suominen et al., 2006; Pahikkala et al., 2006c).

Cross-validation (CV) is a commonly used method for the performance estimation and model selection for the learning algorithms. For RLS, it is widely known that the leave-one-out cross-validation (LOOCV) has a closed form whose computational complexity is quadratic with respect to the number of training examples. In reality, however, the training set may contain data points that are highly dependent with each other. In text categorization tasks, for example, it may happen that the training set contains several documents written by the same author while that author may not have written the new examples to be predicted with the learning machine. In this kind of situation, the LOOCV performance estimate becomes unreliable, because it may be much easier to predict the labels of the documents when the machine is trained with documents written by the same author. Fortunately, we often have a priori knowledge of such clustering in the training data set and we can perform the CV so that we leave out the whole cluster of data points in each CV round. In this paper, we show that also the leave-cluster-out cross-validation can be performed for RLS with much smaller computational complexity than the naive approach in which the RLS would be retrained in each CV round. To our knowledge, this has not been considered in the literature.

2 Regularization Framework

Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (X \times \mathbb{R})^m$ be a training set of m training examples, where $x_i \in X$ are the training data points, $y_i \in \mathbb{R}$ are their labels, and X can be any set. We consider the Regularized Least-Squares (RLS) algorithm as a special case of the following regularization problem known as Tikhonov regularization (for a more comprehensive introduction, see e.g. Poggio and Smale (2003)):

$$\min_f \sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \|f\|_k^2, \qquad (1)$$

where l is the loss function used by the learning machine, $f : X \to Y$ is a function which maps the inputs $x \in X$ to the outputs $y \in Y$, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_k$ is a norm in a Reproducing Kernel Hilbert Space defined by a positive definite kernel function k. The second term is called a regularizer. The loss function used with RLS for regression problems is called least squares loss and is defined as

$$l(f(x), y) = (y - f(x))^2.$$

Note that if we use $l(f(x), y) = \max(y - f(x), 0)$, we obtain the support vector machines (SVM) for classification. Other choices of the loss function lead to other popular classifiers, for example, the SVM regression and kernel logistic regression. By the Representer Theorem (see e.g. Schölkopf et al. (2001)), the minimizer of equation (1) with the least-squares loss function has the following form:

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i), \qquad (2)$$

where $a_i \in \mathbb{R}$ and k is a kernel function.

3 Implementation

We now state the solution of RLS in the case where there are several output labels for each data point (see e.g. Rifkin and Klautau (2004) for a more comprehensive consideration). Below, $\mathcal{M}_{i\times j}(\mathbb{R})$ denotes the set of real valued matrices of dimension $i \times j$. Suppose we have a set of m training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (X \times \mathbb{R}^p)^m$, where $x_i \in X$ are the input variables, X can be any set, and y_i are the corresponding vectors of p output variables. Let $K \in \mathcal{M}_{m\times m}(\mathbb{R})$ be the kernel matrix generated from the training data points using the kernel function $k(x, z)$, that is, $K_{ij} = k(x_i, x_j)$. Let $Y \in \mathcal{M}_{m\times p}(\mathbb{R})$ be a matrix whose rows are the vectors of the output variables, that is, it has one column per each subproblem. Further, let $G = (K + \lambda I)^{-1}$, where $I \in \mathcal{M}_{m\times m}(\mathbb{R})$ is the identity matrix. Because the kernel function from which the kernel matrix is generated is positive definite, the matrix $K+\lambda I$ is invertible when $\lambda > 0$. Given a regularization parameter λ, the coefficient matrix is obtained as follows

$$A = GY \in \mathcal{M}_{m\times p}(\mathbb{R}) \qquad (3)$$

whose columns are the coefficient vectors of the RLS solution (2) for each p output (for a proof, see e.g. Rifkin (2002)). Because the kernel matrix K is the same for all of the p problems, we obtain all solutions from (3) with approximately the cost of solving only one problem.

3.1 Solving RLS via eigen decomposition of kernel matrix

Solution (3) can be obtained simply by calculating the inverse G of the matrix $K+\lambda I$. Instead of calculating the inverse, we compute the solution by first calculating the eigen decomposition of the kernel matrix

$$K = V\Lambda V^{\mathrm{T}}, \qquad (4)$$

where V is an orthogonal matrix that contains the eigenvectors of K and Λ is a diagonal matrix that contains the corresponding eigenvalues. Then, $G = (V\Lambda V^{\mathrm{T}}+\lambda I)^{-1} = V\widetilde{\Lambda}_\lambda V^{\mathrm{T}}$, where $\widetilde{\Lambda}_\lambda = (\Lambda+\lambda I)^{-1}$ is a diagonal matrix that contains the eigenvalues of G. The ith eigenvalue of G is $1/(\mu_i + \lambda)$, where μ_i is the ith eigenvalue of K. Note that we do not need to compute the matrix G, because the solution (3) can now be obtained as

$$A = V\widetilde{\Lambda}_\lambda V^{\mathrm{T}} Y. \qquad (5)$$

From (5) we observe that after we have calculated the eigen decomposition of K, we can easily compute a whole array of RLS solutions for different values of λ. To compute the solution (5) for a certain value of λ, we need to calculate the product of the matrices

$V\widetilde{\Lambda}_\lambda \in \mathcal{M}_{m\times m}(\mathbb{R})$ and $V^{\mathrm{T}}Y \in \mathcal{M}_{m\times p}(\mathbb{R})$ which is fast when the number of subproblems p is small compared to the number of training examples m. We can then select the regularization parameter for each subproblem with a cross-validation which we consider below. Of course, different subproblems may prefer different values of λ. In that case, we do not obtain the column vectors of A from (5), but one by one from $A_p = V\widetilde{\Lambda}_{\lambda_p}V^{\mathrm{T}}Y$, where a_p is the pth column of A and λ_p is the value of the regularization parameter preferred by the pth subproblem.

3.2 Efficient Computation of Cross-Validation

We now consider an efficient computation of cross-validation (CV) for the RLS algorithm. By CV we indicate the method that is used to estimate the performance of the learning algorithm with a given data set. The outline of the method is the following. First, the data set is partitioned into subsets called CV folds. Next, the learning machine is trained with the whole data set except one of the folds that is used to measure the performance of the machine. Each of the CV folds is held out from the training set at a time and the CV performance of the machine is obtained by averaging over the performances measured with the different folds.

In order to calculate the CV performance of a learning machine explicitly, we need to train the machine n times, where n is the number of the CV folds. This is, in many cases, computationally cumbersome, especially if the number of the folds is large. Fortunately, it is possible to obtain the CV performance of RLS with a smaller computational cost than in the naive approach.

We now prove a lemma that we use below to derive a faster method for the computation of CV. The proof is similar to the proof of the leave-one-out lemma (see e.g. Wahba (1990)). For simplicity, we prove only the case in which the number of output variables is one. However, it is easy to extend the lemma for p output variables.

Lemma 1. *Let L be a set of indices of the data points that are held out from the training set and let f_L be the function obtained by training the RLS algorithm with the whole data set except the set of data points indexed by L. By definition, f_L is the solution to the following variational problem*

$$\min_f \sum_{i\notin L}(f(x_i)-y_i)^2 + \lambda\|f\|_k^2. \tag{6}$$

The function f_L is also the solution to the following variational problem

$$\min_f \sum_{i\notin L}(f(x_i)-y_i)^2 + \sum_{i\in L}(f(x_i)-f_L(x_i))^2 + \lambda\|f\|_k^2$$

Proof. We observe that for any function f

$$\begin{aligned}\sum_{i\notin L}(f(x_i)-y_i)^2 + \sum_{i\in L}(f(x_i)-f_L(x_i))^2 + \lambda\|f\|_k^2 \\ \geq \sum_{i\notin L}(f(x_i)-y_i)^2 + \lambda\|f\|_k^2 \\ \geq \sum_{i\notin L}(f_L(x_i)-y_i)^2 + \lambda\|f_L\|_k^2 \\ = \sum_{i\notin L}(f_L(x_i)-y_i)^2 + \sum_{i\in L}(f_L(x_i)-f_L(x_i))^2 \\ +\lambda\|f_L\|_k^2\end{aligned}$$

□

Using the above lemma, we are able to state the result that allows us to calculate the values of the output variables of the data points held out from the training set without explicitly retraining the algorithm with the rest of the training examples.

Let $I \in \mathcal{M}_{m\times m}(\mathbb{R})$ denote an identity matrix and let $I_L \in \mathcal{M}_{|L|\times m}(\mathbb{R})$ be a matrix that contains the rows of I indexed by L. We use the matrix I_L to "cut" out rows from other matrices, or alternatively to "add" rows consisting of zeros. Below, with any matrix $M \in \mathcal{M}_{m\times n}(\mathbb{R})$, where $n \in \mathbb{N}$, we use the subscript L to denote the left multiplication by I_L, that is, we denote $M_L = I_L M$. Further, we denote $M_{LL} = I_L M I_L^{\mathrm{T}}$ for $M \in \mathcal{M}_{m\times m}(\mathbb{R})$. Let $Y \in \mathcal{M}_{m\times p}(\mathbb{R})$ be the label matrix corresponding to the training data and let us denote $B = KG$. Then

$$B = V\Lambda V^{\mathrm{T}}V\widetilde{\Lambda}_\lambda V^{\mathrm{T}} = V\Lambda\widetilde{\Lambda}_\lambda V^{\mathrm{T}}. \tag{7}$$

By the equation (2), the predicted output of the RLS algorithm for its training data is

$$\widehat{Y} = KA = KGY = BY. \tag{8}$$

Finally, let $Y' \in \mathcal{M}_{m\times p}(\mathbb{R})$ denote the matrix consisting of the output values of the training data points obtained using the function f_L.

Theorem 1. *The matrix Y'_L consisting of the output values of the held out data points predicted with f_L can be obtained from*

$$Y'_L = (I_{LL} - B_{LL})^{-1}(\widehat{Y}_L - B_{LL}Y_L). \tag{9}$$

Proof. According to the Lemma 1, the function f_L is obtained by training the RLS using the whole data set with a label matrix $Y - I_L^{\mathrm{T}}Y_L + I_L^{\mathrm{T}}Y_L'$. Knowing the label matrix, we now use the equation (8) to compute the output matrix Y'.

$$\begin{aligned} Y' &= B(Y - I_L^{\mathrm{T}}Y_L + I_L^{\mathrm{T}}Y_L') \\ &= \widehat{Y} - BI_L^{\mathrm{T}}Y_L + BI_L^{\mathrm{T}}Y_L'. \end{aligned}$$

By multiplying with I_L from left we get

$$\begin{aligned} Y_L' &= \widehat{Y}_L - B_{LL}Y_L + B_{LL}Y_L' \\ \Leftrightarrow (I_{LL} - B_{LL})Y_L' &= \widehat{Y}_L - B_{LL}Y_L \\ \Leftrightarrow Y_L' &= (I_{LL} - B_{LL})^{-1}(\widehat{Y}_L - B_{LL}Y_L). \end{aligned}$$

In order to the last equivalence to hold, we have to ensure the invertibility of the matrix $I_{LL} - B_{LL}$. Let γ_i be the ith eigenvalue of B. From (7) we observe that $\gamma_i = (\Lambda\widetilde{\Lambda}_\lambda)_{i,i} = \frac{\mu_i}{\mu_i+\lambda}$. Because $0 \leq \gamma_i < 1$, the matrix $I - B$ is a positive definite. From the positive definiteness of $I - B$, it follows that all its principal submatrices $I_{LL} - B_{LL}$ have strictly positive determinants (see e.g. Meyer (2000)) from which the invertibility follows. □

For a held out set of of size $|L|$, the time consuming part in the computation of (9) is the calculation of the matrix B_{LL}. When we have solved the RLS problem via the eigen decomposition of the kernel matrix, the elements of B_{LL} can be computed using the eigenvectors V and the diagonal elements of $\Lambda\widetilde{\Lambda}_\lambda$, since $B = V\Lambda\widetilde{\Lambda}_\lambda V^{\mathrm{T}}$. Thus, the computational complexity of calculating the outputs Y_L' for the held out set is $O(|L|^2 m)$. If we perform an n-fold cross-validation with the training set, the number and the size of the held out sets are n and m/n, respectively, and the overall complexity of the cross-validation is $O(n(m/n)^2 m) = O(m^3/n)$.

The larger the number of folds in the cross-validation is, the faster is its computation. In the extreme case where the size of the held out set is 1, the computational complexity is $O(m^2)$, since we only have to calculate the diagonal elements of B in the whole cross-validation process. Indeed, from Theorem 1, we obtain as a special case the known result of the leave-one-out cross-validation (LOOCV) for RLS (this case has been proved, for example, by Vapnik (1979); Wahba (1990); Green and Silverman (1994)).

Corollary 1. *Let $f(x_j)$ and $f_j(x_j)$ denote the output of the RLS algorithm for the training example x_j, when the algorithm is trained with all examples and all examples except x_j, respectively. We can calculate the value of the output $f_j(x_j)$ as follows*

$$f_j(x_j) = \frac{f(x_j) - B_{j,j}Y_j}{1 - B_{j,j}}. \tag{10}$$

From the computational complexity perspective, the LOOCV should be preferred with the RLS. However, in practice, there are often cases where the LOOCV should not be used to estimate the performance of the learning algorithm because of dependencies between the training data points. Note also that the sizes of the cross-validation folds do not have to be equal. Therefore, we can divide the training set into folds of different sizes according to the dependencies between the training points. Below, we discuss those cases in more detail.

4 Experiments

With real world data, it is often the case that the data points used to train a machine learning method are not completely independent. For example, we may have several text documents written by the same author in text categorization tasks. A document may be very similar to other the documents written by the same author but very different compared to the documents written by an other author. In these cases, the leave-one-out cross-validation (LOOCV) performance of a learning machine may not be a good estimate of the true performance of the learning machine. This is, because on the contrary to the case with the training set, it is rarely the case that a document to be classified with a trained learning machine is written by the same author as some of the documents in the training set. Generally, we say that the training set consists of clusters of data points. By a cluster we indicate a subset of training examples that are mutually dependent.

Fortunately, we often have a priori knowledge of such clustering in the training data set and we can perform the CV so that we leave out the whole cluster of data points in each CV round. For example, in our earlier experiments using support vector machines and Bayesian classifiers on natural language disambiguation tasks (Pahikkala et al., 2005a,b,c, 2006a,b), we often extracted several examples of the words to be disambiguated from a single text document. When we used bag-of-words kind of features extracted from the whole text, the examples originating from the same text document formed a cluster in the training set. Clearly, it does not happen in practice that a learning machine trained to detect context sensitive spelling errors, is trained with examples originating from the same document the examples to be predicted are originated from. Thus, we had to perform the cross-validation on the document level so that no two examples from the same document end up in different cross-validation folds.

Here we demonstrate the “clustered training set effect” by comparing the LOOCV performance of a trained RLS to a leave-cluster-out cross-validation (LCOCV) performance so that each fold in the LCOCV consists of the training examples that form a cluster in the training set. The demonstration is done with the problem of dependency parse ranking of sentences extracted from biomedical texts. Here we give a brief introduction to the problem, the data, and the solution approach. For a detailed description, see the paper by Tsivtsivadze et al. (2005).

To generate the training data, we took one hundred sentences from the BioInfer corpus (Pyysalo et al., 2006). Each sentence in the corpus has a manually tagged linkage that corresponds to the “correct” parse that a parser is supposed to output for the sentence. For each sentence, we use link grammar parser (Sleator and Temperley, 1991) to generate a set of candidate parses. The number of candidate parses depends of the sentence. If the parser generates more than 20 parses for a sentence, we randomly select 20 of them and discard the rest. Otherwise, we keep all candidate parses. For each candidate parse we calculate a score value that indicates how close to the correct parse it is. The score value is the F-score calculated from the link differences between the candidate parse and the correct parse as follows (Tsivtsivadze et al., 2005):

$$F = \frac{2TP}{2TP + FP + FN},$$

where TP, FP, and FN are the numbers of true positives (the links present in both the candidate and the correct parse), false positives (links present in the candidate parse but not in the correct one), and false negatives (links present in the correct parse but not in the candidate), respectively. The training set is constructed from the candidate parses and their score values. The task of the learning machine is, for a sentence, to rank its candidate parses in the order of their score values. The RLS algorithm is, in fact, trained to regress the score values of the parses but in this paper, we are only interested of the ranking of the candidate parses for each sentence.

We measure the ranking performance by calculating Kendalls τ_b correlation coefficient (Kendall, 1970) for the parse candidate set of each sentence. Note that the coefficient is calculated for each sentence separately, since we are not interested of the mutual order of the parses originating from different sentences. The overall performance for the whole data set is obtained by taking the average of the correlation coefficients of the sentences in the data set.

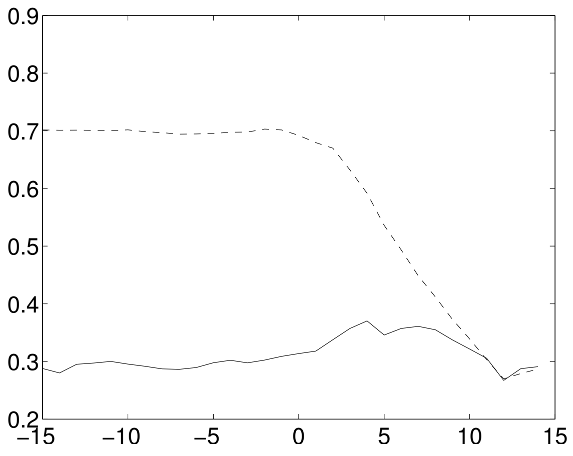

Figure 1: The ranking performance of the RLS algorithm computed with LOOCV (dashed line) and LCOCV (solid line). The x-axis denotes the value of the regularization parameter in a logarithmic scale. The y-axis is the ranking performance measured with τ_b correlation coefficient.

The kernel function, that we use as a similarity measure of the parses, is described in detail by Tsivtsivadze et al. (2005). The kernel function has the drawback that two parses originating from a same sentence have almost always larger mutual similarity than two parses originating from different sentences. Therefore, the data set consisting of the parses is heavily clustered in the feature space determined by the kernel function. The clustered structure of the data can have a strong effect on the performance estimates obtained by cross-validation, because data points that are in the same cluster as a held out point have a dominant effect on the predicted output of the held out point. This does not, however, model the real world, since a parse ranker is usually not trained with parses originating from the sentence from which the new parse with an unknown F-score is originated. The problem can be solved by performing the cross-validation on the sentence level so that all the parses generated from a sentence would always be either in the training set or in the test set.

In order to compare the performance estimates given by the LOOCV and LCOCV, we train an RLS algorithm with the data set described above. Recall that after we have a trained RLS, we obtain the LOOCV output for each parse in the data using (10). From the LOOCV output, we calculate the

F-scores for each sentence and compute their average. We calculate the LCOCV performance by leaving each sentence out from the training set at a time and computing the F-scores for their parses with (9). We make a grid search for the regularization parameter λ of the RLS algorithm with the grid points $2^{-15}, 2^{-14}, \ldots, 2^{14}$. With the grid search, we can test the performances of LOOCV and LCOCV in the model selection of RLS.

The results of the comparison are illustrated in Figure 1. From the figure, we observe that the performance difference between the LOOCV and the LCOCV is, with some values of the regularization parameter, over 0.3 correlation points. Thus, LOOCV clearly overestimates the ranking performance. Moreover, the LOOCV prefers small values of the regularization parameter (the most preferable value of the regularization parameter is 2^{-2}) while the LCOCV prefers a more regularized solution (the most preferable value of the regularization parameter is 2^{4}). Therefore, we can also conclude that the LOOCV may not be a good method for the model selection when the training set is clustered. Indeed, when we test the ranking performance of the RLS with one hundred test sentences unseen to the RLS, we obtain correlations 0.37 and 0.38 with the machines trained with the whole training set and with the regularization parameters preferred by the LOOCV and LCOCV, respectively.

5 Conclusion

The regularized least-squares (RLS) algorithm has been shown to be a competitive alternative to the standard support vector machines in several machine learning tasks. It also have several computational advantages, such as solving several tasks in parallel, fast tuning of the regularization parameter, and fast leave-one-out cross-validation (LOOCV).

The LOOCV method, however, is not always a suitable method for performance estimation or model selection because of dependencies between the training data points. In several learning problems, the training data set may be clustered so that the prediction of a held out data point will be unrealistically easy if the data points that belong in the same cluster with the held out data point are kept in the training set. Since it may not happen in the real world that a data point, whose output is to be predicted with a learning machine, belongs in the same cluster with some of the training data points, the LOOCV estimate does not reflect the reality.

We introduce and prove a closed form of an n-fold cross-validation performance estimate for the RLS algorithm. The closed form can be used to calculate a leave-cluster-out cross-validation (LCOCV), in which a whole cluster of training examples is held out from the training set in each cross-validation round avoiding the harmful effect of the clustered training set.

We experimentally demonstrate the effect of the clustered data set on the LOOCV performance estimate with a ranking task of dependency parses generated from biomedical texts. With the same data and task, it is also demonstrated that the LCOCV is better than LOOCV as a model selection tool.

Acknowledgments

This work has been supported by Tekes, the Finnish Funding Agency for Technology and Innovation.

References

Nicola Ancona, Rosalia Maglietta, Annarita D'Addabbo, Sabino Liuni, and Graziano Pesole. Regularized least squares cancer classifiers from dna microarray data. *BMC Bioinformatics*, 6 (Suppl 4):S2, 2005.

P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models, A Roughness Penalty Approach*. Chapman and Hall, London, 1994.

Marketta Hiissa, Tapio Pahikkala, Hanna Suominen, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Towards automated classification of intensive care nursing narratives. In *The 20th International Congress of the European Federation for Medical Informatics (MIE 2006)*, Maastricht, Netherlands, 2006. To appear.

Maurice G. Kendall. *Rank Correlation Methods*. Griffin, London, 4. edition, 1970.

Carl D. Meyer. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

Tapio Pahikkala, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Incorporating external information in bayesian classifiers via linear feature transformations. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Proceedings of the 5th International Conference on*

NLP (FinTAL 2006), volume 4139 of *Lecture Notes in Computer Science*, pages 399–410, Heidelberg, Germany, 2006a. Springer-Verlag.

Tapio Pahikkala, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Contextual weighting for support vector machines in literature mining: an application to gene versus protein name disambiguation. *BMC Bioinformatics*, 6(1):157, 2005a.

Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Matrix representations, linear transformations, and kernels for natural language processing, 2006b. Submitted.

Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Improving the performance of bayesian and support vector classifiers in word sense disambiguation using positional information. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 90–97, Espoo, Finland, 2005b. Otamedia OY.

Tapio Pahikkala, Sampo Pyysalo, Filip Ginter, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Kernels incorporating word positional information in natural language disambiguation tasks. In Ingrid Russell and Zdravko Markov, editors, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 442–447, Menlo Park, Ca, 2005c. AAAI Press.

Tapio Pahikkala, Evgeni Tsivtsivadze, Jorma Boberg, and Tapio Salakoski. Graph kernels versus graph representations: a case study in parse ranking. In *ECML/PKDD'06 workshop on Mining and Learning with Graphs (MLG'06)*, 2006c. To appear.

Tomaso Poggio and Steve Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544, 2003.

Marius Popescu. Regularized least-squares classification for word sense disambiguation. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 209–212, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: A corpus for information extraction in the biomedical domain, 2006. Submitted.

Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning.* PhD thesis, MIT, 2002.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In D. Helmbold and R. Williamson, editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426, Berlin, Germany, 2001. Springer-Verlag. ISBN 3-540-42343-5.

Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

Daniel D. Sleator and Davy Temperley. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1991.

Hanna Suominen, Tapio Pahikkala, Marketta Hiissa, Tuija Lehtikunnas, Barbro Back, Eija Helena Karsten, Sanna Salanterä, and Tapio Salakoski. Relevance ranking of intensive care nursing narratives. In *proceedings of KES2006 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems*, 2006. to appear.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, 1999.

Evgeni Tsivtsivadze, Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Locality-convolution kernel and its application to dependency parse ranking. In Moonis Ali and Richard Dapoigny, editors, *Proceedings of the 19th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE 2006)*, volume 4031 of *Lecture Notes in Computer Science*, pages 610–618, Heidelberg, Germany, 2006. Springer-Verlag.

Evgeni Tsivtsivadze, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski. Regularized least-squares for parse ranking. In A. Fazel Famili, Joost N. Kok, José Manuel Peña, Arno Siebes, and A. J. Feelders, editors, *Proceedings of the 6th International Symposium on Intelligent Data Analysis*, volume 3646 of *Lecture Notes in Computer Science*, pages 464–474, Heidelberg, Germany, September 2005. Springer-Verlag.

V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

Grace Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.