Ensemble Learning with Supervised Kernels

Kari Torkkola[1] and Eugene Tuv[2]

[1] Motorola, Intelligent Systems Lab, Tempe, AZ, USA
Kari.Torkkola@motorola.com
[2] Intel, Analysis and Control Technology, Chandler, AZ, USA
eugene.tuv@intel.com

Abstract. Kernel-based methods have outstanding performance on many machine learning and pattern recognition tasks. However, they are sensitive to kernel selection, they may have low tolerance to noise, and they can not deal with mixed-type or missing data. We propose to derive a novel kernel from an ensemble of decision trees. This leads to kernel methods that naturally handle noisy and heterogeneous data with potentially non-randomly missing values. We demonstrate excellent performance of regularized least square learners based on such kernels.

1 Introduction

Kernel-based learners, such as Support Vector Machines (SVM) or Regularized Least Squares (RLS) learners have shown excellent performance compared to any other methods in numerous current classification and regression applications [23]. A key issue in the successful application of a kernel-based learner is the proper choice of the kernel and its parameters since this determines the capability of the kernel to capture and to represent the structure of the input space. Typical kernel choices for continuous data are Gaussian and polynomial, for example.

However, if the data has variables of mixed type, both continuous and discrete, kernel construction becomes a difficult problem. Furthermore, if the data contains many irrelevant variables, especially if the number of available observations is small, standard kernel methods require variable selection to perform well [13]. This paper suggests a solution to both problems by deriving a kernel from an ensemble of trees.

Tree-based ensemble methods partition the input space using decision trees and then combine the partitioning, or response in case of regression, over a stochastic ensemble of trees. Random Forest (RF) is an example of such an ensemble [4]. Trees provide the capability to handle mixed-type variables and missing data. Stochastic selection of input variables for each tree of the forest provides tolerance to irrelevant variables. These properties of trees can be extended to kernel-based methods by taking advantage of the supervised engine of the Random Forest. We show how the kernel can be derived from the structure of the forest.

The structure of this paper is as follows. We describe first generic kernel-based learners concentrating on Regularized Least Squares learners. We then show how a “supervised kernel” can be derived from a tree-based ensemble that

© Springer-Verlag Berlin Heidelberg 2005

has been trained either for a classification or a regression task. Ensembles of kernel-based learners are described next, followed by illustrations and experimentation with the supervised kernel. We use both synthetic and real data sets to demonstrate both the insights from supervised kernels as well as their relevance to real problems. A summary concludes the paper.

2 Regularized Least Squares Learners

In supervised learning the training data $(x_i, y_i)_{i=1}^m$ is used to construct a function $f : X \to Y$ that predicts or generalizes well.

To measure goodness of the learned function $f(x)$ a loss function $L(f(x), y_{true})$ is needed, for example the square loss, L_2: $L(f(x), y) = (f(x) - y)^2$.

Given a loss function, the goal of learning is to find an approximation function $f(x)$ that minimizes the expected risk, or the generalization error

$$E_{P(x,y)} L(f(x), y) \tag{1}$$

where P(x,y) is the unknown joint distribution of future observations (x,y).

Given a finite sample from the (X,Y) domain this problem is ill-posed.

The regularization approach rooted in Tikhonov regularization theory [24] restores well-posedness (existence, uniqueness, and stability) by restricting the hypothesis space, the functional space of possible solutions:

$$\hat{f} = \underset{f \in H}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L(f(x_i), y_i) + \gamma \left\| f \right\|_K 2 \tag{2}$$

The hypothesis space H here is a Reproducing Kernel Hilbert Space (RKHS) defined by kernel K, and γ is a positive regularization parameter.

The mathematical foundations for this framework as well as a key algorithm to solve (2) are derived elegantly in [21] for the quadratic loss function. The algorithm can be summarized as follows:

1. Start with the data $(x_i, y_i)_{i=1}^m$.
2. Choose a symmetric , positive definite kernel, such as

$$K(x, x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}. \tag{3}$$

3. Set

$$f(x) = \sum_{i=1}^{m} c_i K(x_i, x), \tag{4}$$

where $\mathbf{c}$ is a solution to

$$(m\gamma \mathbf{I} + \mathbf{K})\mathbf{c} = \mathbf{y}, \tag{5}$$

which represents well-posed linear system.

The generalization ability of this solution, as well choosing the regularization parameter γ were studied by [7, 8].

It is important to note that the same result can be obtained from the classical linear ridge regression proposed by [16, 15] to deal with potential singularity of $\mathbf{X}'\mathbf{X}$ in linear regression.

$$f(x) = \mathbf{X}(\mathbf{X}'\mathbf{X} + \gamma\mathbf{I})^{-1}\mathbf{X}'y \tag{6}$$

It is easy to see that we could use the *kernel trick* if we rewrite equation (6) in terms of matrix of inner products

$$f(x) = (\mathbf{K} + \gamma\mathbf{I})^{-1}\mathbf{K}y \tag{7}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}'$.

Thus, the regularized least-squares (RLS) algorithm defined above solves a simple well defined linear problem. The solution is a linear kernel expansion of the same form as the one given by support vector machines (SVM). Note also that SVM formulation naturally fits in the regularization framework (2). Inserting the SVM hinge loss function $L(f(x), y) = (1 - yf(x))_+$ in (2) leads to solving a quadratic optimization problem instead of a linear solution.

The regularized least-squares classifier (RLSC) with quadratic loss function, that is more common for regression, has also proven to be very effective in binary classification problems [22]. In a recent feature selection and classification competition organized at NIPS2003, stochastic ensembles of RLSCs with Gaussian kernels were the second best entry [13].

The success of kernel-based methods stems partly from the capability of the kernel to capture the structure of the (expanded) input space. However, while some of that structure is relevant to the task at hand, much of it may not be. Selection of an appropriate kernel is the key issue. Furthermore, crafting kernels for mixed type noisy data with potential missing values is a form of art at its best.

Recently, a series of papers were dedicated to a problem how to learn the kernel itself. Cristianini et al. introduced notion of kernel target alignment [6]. The goal is to construct a kernel $K(x, x')$ that is similar to (aligned with) the target kernel defined as outer product of y, $\mathbf{K}^* = yy^T$. While this framework could construct powerful kernels, there is still an overfitting issue since there is no guarantee that the learned kernel is aligned on the data other than the given training data set. It is not clear also how to use this framework in regression or multiclass classification. Lanckriet et al. proposed semidefinite programming approach to optimize target alignment (or bound, margin) over the set of kernel matrices on the data [17]. Overfitting is still an issue here as well as a computational complexity ($O(n^6)$). None of these methods deal with mixed type data.

The following section discusses how ensembles of trees can be used for deriving the kernel in a supervised fashion.

3 Extracting Kernels from Tree-Based Ensembles

A decision tree such as CART explicitly partitions the input space into a set of disjoint regions, and assigns a response value to each corresponding region [5]. This is the key property that we will use in the derivation of a similarity matrix (that is, a kernel) from a tree.

As a new observation x is run through the tree, the search ends at a terminal node $t(x)$. Let us denote the depth of a node in the tree by $d(n)$, and the deepest common parent of two nodes by $p(n_1, n_2)$.

Breiman defined a similarity measure between two observations x_1 and x_2 as follows [4]:

$$s_B(x_1, x_2) = \begin{cases} 1 & \text{if } t(x_1) = t(x_2) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

This a very coarse measure by which two observations are similar only if they end up in the same terminal node. In order to express with finer granularity how far in the tree two observations lie, we define a new similarity measure as the depth of the lowest common parent node normalized by the level of the deeper observation in the tree:

$$s(x_1, x_2) = \frac{d(p(t(x_1), t(x_2)))}{\max[\, d(t(x_1)), d(t(x_2)) \,]} \tag{9}$$

It is easy to show that the similarity matrix defined this way is symmetric and positive-definite. Two observations with large $s(x_1, x_2)$ would indicate both geometric closeness and similarity in terms of the target. This kind of partitioning (and thus similarity) can be constructed for practically any kind of data since CART is fast, works with mixed-type data, handles missing values elegantly, and is invariant to monotone transformations of the input variables, and therefore is resistant to outliers in input space. However, such a supervised partitioning of the input space induced by a single tree is unstable (in the sense of [2]), and it would be sensible to measure the derived similarity over an ensemble of trees by averaging.

We discuss two of the most recent advances in tree ensembles, MART (gradient tree boosting) [11, 12] and Random Forest [4]. MART is a serial ensemble where every new expert that is constructed relies on previously built experts. At every iteration of MART a new tree is fitted to the generalized residuals from the previous iteration.

Random Forest (RF) is an improved bagging method that extends the "random subspace" method [14]. It is a parallel ensemble that grows a forest of independent random trees on bagged samples. RF does not overfit, and can be summarized as follows:

1. A number m is specified much smaller than the total number of variables M (typically $m \sim \sqrt{M}$).
2. Each tree of maximum depth is grown on a bootstrap sample of the training set.
3. At each node, m out of the M variables are selected at random.
4. The split used is the best split on these m variables.

We will be using RF throughout our experimentation because of it simplicity and excellent performance. In general, RF is resistant to irrelevant variables, it can handle mixed-type data and missing data, and it can handle massive numbers of variables and observations. These properties now carry over to the kernel constructed from a RF. Similarity matrices according to Eq. (9) will now be averaged over all the trees in the forest.

We illustrate now some properties of this kernel using two simple data sets.

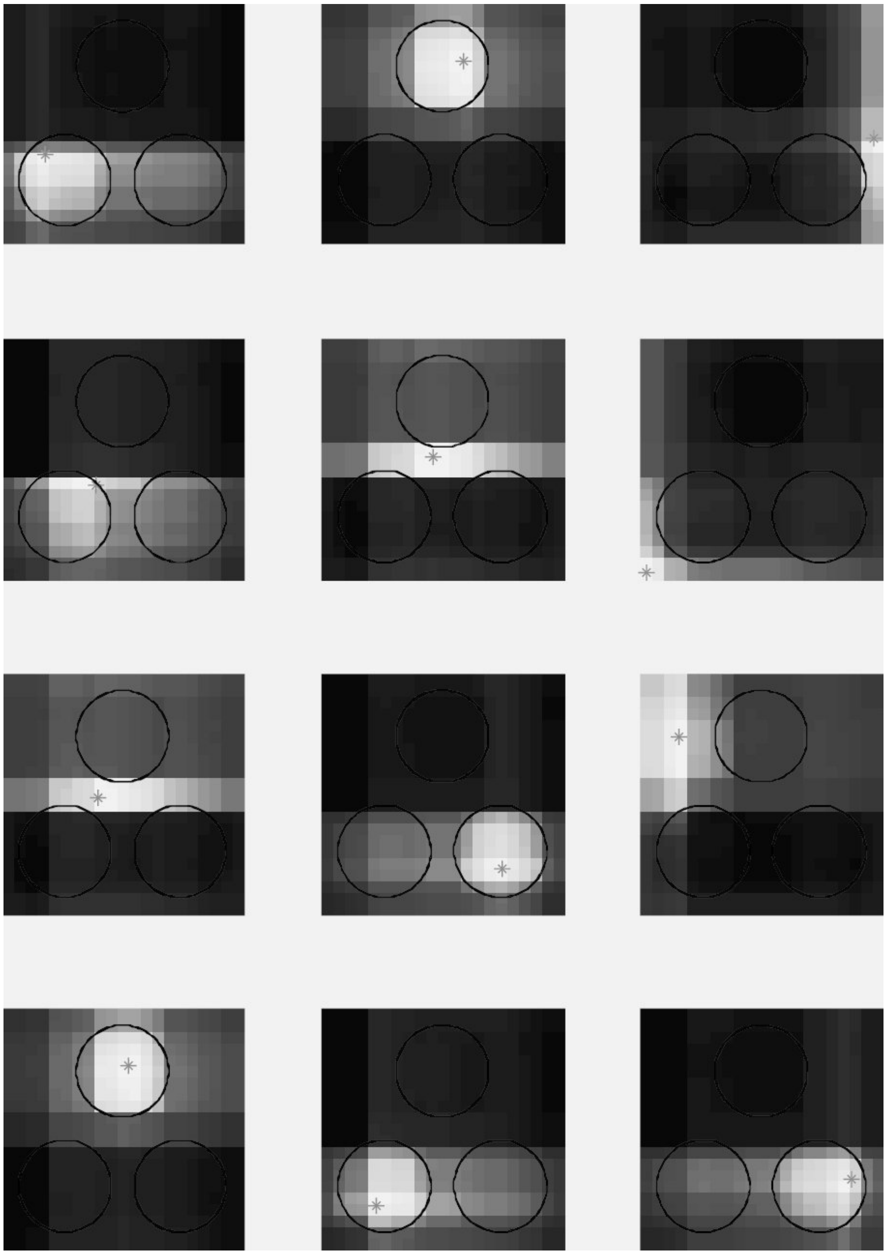


Fig. 1. Illustration of the similarity measure between samples of two different classes. Class one is the inside of the three circles and class two is the outside of the circles. Each panel displays one randomly chosen data point within a square as a grey asterisk, and the similarity between the point and every other point is illustrated by shades of grey. Light shade denotes similarity between the data points. See text for discussion.

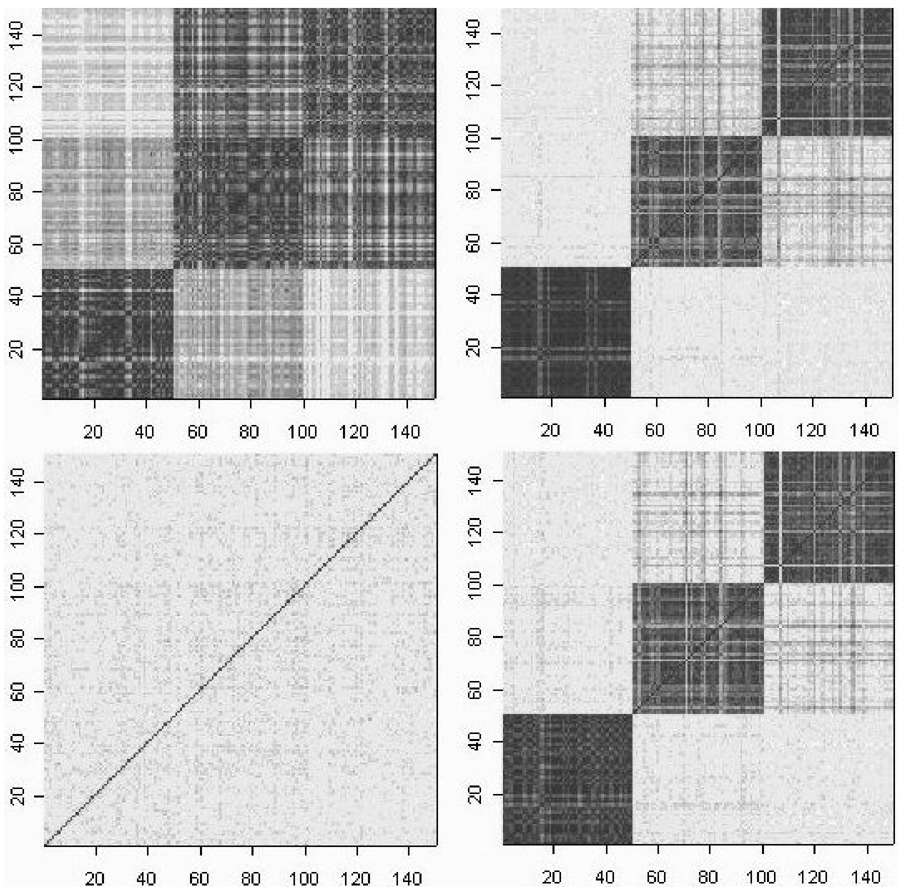


Fig. 2. Similarity matrices (kernels) derived from the Iris data set with three classes. Dark color represents high similarity, light represents low. The left panels depict Euclidean similarity metric (a linear kernel), and the right panels show the similarity derived from a Random Forest. The top panels represent the original four variables, and the bottom panels show the same data but with 40 additional noise variables. This is a demonstration of the tolerance of the supervised kernel to irrelevant input variables.

Figure 1 depicts the similarity measure in the case of two-dimensional data with two classes. The region consisting of the union of the three circles is considered class one and the region outside the circles represents class two. Similarity between a randomly chosen point and every other point in the square is encoded by shade of grey, the lighter the more similar. We can see how the kernel captures the local structure and at the same time enhances separation of the classes. We can also see the limitations of trees as the rectangular partitioning.

Figure 2 uses the well-known Iris data set (3 classes) to demonstrate how class discrimination is enhanced (top panels) as compared to a linear kernel, and how the derived kernel matrix is tolerant to irrelevant input variables (bottom panels).

4 Stochastic Ensembles of RLS Learners

The derived kernel matrix can now be used with any kernel-based learner, such as a Support Vector Machine. This section describes how we use it with stochastic ensembles of RLS learners. We begin by briefly recalling the motivation for using ensemble methods in the first place.

Generalization ability of a learned function is closely related to its stability. Stability of the solution could be loosely defined as continuous dependence on the data. A stable solution changes very little for small changes in data. Recently, a series of theoretical developments also confirmed the fundamental role of stability for generalization of any learning engine [2, 20, 18, 19].

Supervised ensemble methods construct a set of base learners, or experts, and use their weighted outcome to predict new data. Numerous empirical studies confirm that ensemble methods often outperform any single base learner [10, 1, 9]. Due to increased stability, the improvement is intuitively clear when a base algorithm is unstable (such as for decision tree, neural network, etc). However, it has even been shown that ensembles of low-bias support vector machines (SVM) often outperform a single, best-tuned, canonical SVM [25].

It is well known that bagging (bootstrap aggregation) can dramatically reduce variance of unstable learners providing some regularization effect [3]. Bagged ensembles do not overfit. Low bias of the base learner and low correlation between base learners are crucial to good ensemble performance.

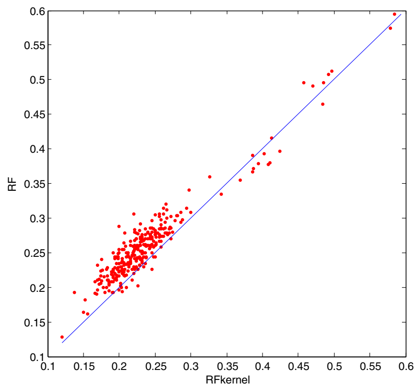
It is thus advantageous to implement diverse low biased experts. For RLSC, bias can be lowered by decreasing the regularization parameter, and narrowing the σ in case of Gaussian kernel. Instead of bootstrap sampling from training data which imposes fixed sampling strategy, we found that often much smaller sample sizes improve performance.

Typically, in the following experiments, once a kernel is derived, we construct 100-250 experts, each using a random sample of 50-90% of the training data. This is fast since we are actually just sampling a pre-computed kernel matrix, and solving a linear equation for each expert. The regularization parameter was fixed to a very small value only to ensure that a solution exists.

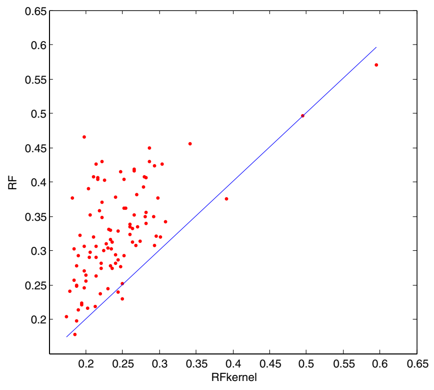
Combining the outputs of the experts in an ensemble can be done in several ways. In classification tasks, we performed majority voting over the outputs of the experts. In binary classification this is equivalent to averaging the discretized (+1,-1) predictions of the experts. In regression tasks simple averaging was used.

5 Experiments with Synthetic Data

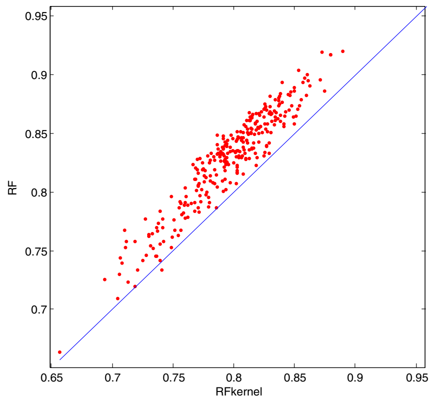
We describe now experiments for the purpose of highlighting the differences between RF and a stochastic ensemble of RLSCs using the new RF kernel. This



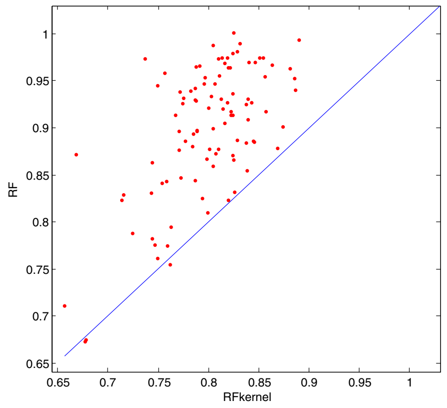
Binary classification task, continuous-valued inputs.



Binary classification task, mixed-valued inputs.



Regression task, continuous-valued inputs.



Regression task, mixed-valued inputs.

Fig. 3. RMS error and classification error rate comparison of Random Forest to an ensemble of RLSCs using a random forest kernel. See text for discussion.

is an important comparison because if the kernel does not improve over the plain RF, it obviously is not worth using at all.

A very useful data generator for this purpose is described by [11]. This generator produces data sets with multiple non-linear interactions between input variables.

We present results using 100 generated data sets, 500 observations each. For each data set, twenty $N(0,1)$ distributed input variables were generated. The target is a multivariate function of ten of those, thus ten are pure noise. The target function is generated as a weighted sum of L multidimensional Gaussians, each Gaussian at a time involving about four input variables randomly drawn from the "important" ten variables. Thus all of the "important" ten input variables are involved in the target, to a varying degree. The Gaussian functions also have a random mean vector and a random covariance matrix as described in [11]. In this experiment, we used $L = 20$. Weights for the Gaussians are randomly drawn from $U[-1, 1]$.

The data generator produces continuous valued variables. Thus the data sets can be used as such for regression problems. In order to generate a classification problems, the target variable was discretized to two levels. Furthermore, to create data sets with mixed type variables, randomly selected 50% of the input variables were discretized. The number of discretization levels was itself a random variable drawn from a uniform distribution between 2 and 10.

Figure 3 depicts the performance of the RF kernel (horizontal axis) against RF (vertical axis) in four different tasks:

1. Classification, continuous inputs, top left,
2. Regression, continuous inputs, bottom left,
3. Classification, mixed type inputs, top right,
4. Regression, mixed type inputs, bottom right.

Each point in the figures depicts one generated data set, 500 observations for training, and 500 observations for testing. The y-coordinate of a point represents either the error rate (classification problem) or the RMS error (regression) evaluated by a random forest. The x-coordinate of a point represents the same for a stochastic ensemble of RLSCs using supervised kernel derived from the random forest. Thus every point above the diagonal line is a data set with which the supervised kernel was better than the RF. The difference is clear for continuous data, both for classification and regression (left panels), and dramatic for mixed type data (right panels).

This difference could be explained by the difference between the base learners of the ensembles. Both are parallel ensembles, but the base learner in RF is less capable than the base learner in an ensemble of RLSCs.

6 High-Dimensional Noisy Problems

In order to assess the performance of the proposed supervised kernel method in real high dimensional noisy problems, we ran tests using the data sets from the NIPS 2003 feature selection competition. The purpose of the challenge in feature selection was to find feature selection algorithms that significantly outperform methods using all features, on all five benchmark data sets [13]. These are significantly harder problems than, for example, typical problems in the UCI machine learning database. The diverse characteristics of these five data sets are listed in Table 1.

As seen in Table 1, these problems are high-dimensional, have a small amount of training data relative to the dimension, and a large proportion of variables are noise. Of these data sets, only Dorothea was highly unbalanced with approximately 12% of samples in one class, and 88% in the other. The rest of the sets had an approximately balanced class distribution. All tasks are two-class classification problems.

Table 1. NIPS2003 Feature Selection Challenge Data. Probes refer to artificially inserted random noise variables.

Data Set	Domain	Size	Type	# of variables	Training Examples	Validation Examples	% of Probes
Arcene	Mass Spectrometry	8.7 MB	Dense	10000	100	100	30
Gisette	Digit Recogn.	22.5 MB	Dense	5000	6000	1000	30
Dexter	Text Classific.	0.9 MB	Sparse	20000	300	300	50
Dorothea	Drug Discovery	4.7 MB	Sparse bin.	100000	800	350	50
Madelon	Artificial	2.9 MB	Dense	500	2000	600	96

Table 2. Classification error rates on NIPS 2003 feature selection competition data sets. Error rates with validation data sets are reported. The Gaussian kernel was used without feature selection with the Arcene data set.

Data Set	Random Forest	Gaussian Kernel RLSC Ensemble	Gaussian Kernel RLSC Ensemble with Feature Selection	Supervised Kernel RLSC Ensemble
Arcene	23.0	13.31	13.31	19.0
Dexter	9.0	32.4	6.67	6.0
Dorothea	11.94	-	11.83	11.78
Gisette	3.4	-	2.2	1.9
Madelon	9.5	25.4	7.0	11.83

Table 2 compares the classification error rates of a RF to a stochastic RLSC ensemble that uses a Gaussian kernel and to a stochastic RLSC ensemble that uses the proposed supervised kernel. The latter pair is a relevant comparison because we use the supervised RF kernel here exactly in the same fashion as the Gaussian kernel is used in training an ensemble of RLSCs. Note that none of these are mixed-type problems. Thus the Gaussian kernel is near-ideal for these data sets once irrelevant variables are removed. Our baseline is thus the column "Gaussian Kernel RLSC Ensemble with Feature Selection".

Comparison to RF is also very relevant because the supervised kernel is derived from a random forest. This naturally begs the question "Why not use just a RF?". Error rates for the RF and the "Gaussian kernel RLSC Ensemble with Feature Selection" entries were taken from the original December 1^{st} entries in the challenge website[1].

These results show that a stochastic ensemble of RLSCs using a supervised kernel derived from a RF significantly outperforms plain RF on four of the five data sets. Comparing to using a Gaussian kernel with feature selection, the supervised kernel has a comparable performance except for the two cases that were near ideal for the Gaussian kernel once feature selection was used [13] (The data in the Madelon and Arcene sets consisted of Gaussian-like clusters).

Although the purpose of the NIPS 2003 competition was feature selection, we do not perform here feature selection for the supervised kernel prior to classification in order to demonstrate that supervised kernels are tolerant to noise variables due to the properties of tree-based ensembles. Thus the "Supervised Kernel" column does not include feature selection. For comparison, we also eval-

[1] `http://www.nipsfsc.ecs.soton.ac.uk`

uated the Gaussian kernel without feature selection on some of the data sets. These results clearly indicate that removing irrelevant variables is a crucial step in using the Gaussian kernel, whereas the supervised RF kernel does not require this at all.

The results here are thus the best we could hope for: Even when the problem does not fit our specific motivation for the kernel choice, the supervised kernel has about equal performance with the (near) optimal method for those problems.

7 Conclusion

This paper shows how the power of kernel-based methods can be extended to heterogeneous and more complex data domains.

We demonstrated how to derive kernels from Random Forests. This brings the nice properties of tree-based learners, such as natural handling of mixed-type data, tolerance to missing data, tolerance to noise, and to irrelevant inputs to all kernel-based learners.

Using the derived kernels, we train ensembles of simple regularized least squares learners and show that such ensembles outperform Random Forest itself on practically any task. The improvement with mixed-type inputs was especially dramatic. Furthermore, the performance on continuous-valued data is comparable to the best methods for such domains.

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36:525–536, 1999.
2. O. Bousquet and A. Elisseeff. Algorithmic stability and generalization performance. In *NIPS*, pages 196–202, 2000.
3. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
4. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
5. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. CRC Press, 1984.
6. Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *Proc. NIPS*, pages 367–373, 2001.
7. F. Cucker and S. Smale. On the mathematial foundations of learning. *Bulletin of the American Mathematical Society*, 89(1):1–49, 2001.
8. F. Cucker and S. Smale. Best choices for regularization parameters in learning theory: on the bias-variance problem. *Foundations of Computational Mathematics*, 2(4):413–428, 2003.
9. T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000.
10. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th ICML*, 1996.
11. J.H. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University, 1999.
12. J.H. Friedman. Stochastic gradient boosting. Technical report, Dept. of Statistics, Stanford University, 1999.

13. Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.
14. T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
15. A. Hoerl and R. Kennard. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(3):69–82, 1970.
16. A. Hoerl and R. Kennard. Ridge regression; biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.
17. Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
18. S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. Technical Report 024, Massachusetts Institute of Technology, Cambridge, MA, 2002. AI Memo #2002-024.
19. T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.
20. T. Poggio, R. Rifkin, S. Mukherjee, and A. Rakhlin. Bagging regularizes. CBCL Paper 214, Massachusetts Institute of Technology, Cambridge, MA, February 2002. AI Memo #2002-003.
21. T. Poggio and S. Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544, 2003.
22. R. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, MIT, 2002.
23. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
24. A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. W.H.Wingston, Washington, D.C., 1977.
25. G. Valentini and T. Dietterich. Low bias bagged support vector machines. In *Proc ICML*, pages 752–759, 2003.