

Random Projection Features and Generalized Additive Models

Subhransu Maji

Computer Science Department,
University of California, Berkeley
Berkeley, CA 94709–8798

Homepage: <http://www.cs.berkeley.edu/~smaji>

Abstract—We propose to learn generalized additive models for classification which represents the classifier using a sum of piecewise linear functions and show that a recently proposed fast linear SVM training method (Pegasos) can be adapted to train such models with the same convergence rates. To be able to learn functions on combination of dimensions, we explore the use of random projection features which learns a classifier on data projected using an arbitrary matrix. In our experiments we find that : (i) The piecewise linear consistently outperforms linear on various datasets (ii) Random projection features perform even better and are close to the best results (using RBF kernels) while potentially being much faster to train on large datasets.

I. INTRODUCTION

Non parametric methods like kernel support vector machines (SVMs) are attractive because they can approximate any decision boundary arbitrarily well given enough training data. Though non-linear kernel SVMs are more versatile, they do not scale well with training data and take significantly longer to train on large datasets compared to linear SVMs. In addition it has to remember a potentially significant fraction of the data (support vectors) to represent the classifier. At classification time the kernel SVMs have to compute the kernel function between an unseen feature vector and all the support vectors which can be very expensive. This makes linear SVMs very attractive for classification and there has been a recent surge of efficient for training linear SVMs including [5], [10], [3].

One possible way to leverage on these developments is to construct linear embeddings of the data which represent the underlying Reproducing Kernel Hilbert Space (RKHS). This is non trivial as for some kernels like radial basis kernels, the RKHS is infinite dimensional. The work of Rahimi [8] shows that many shift invariant kernels $k(x - y)$, can be approximated within ϵ by embedding into a hilbert space with $D = O(d\epsilon^{-2} \log \frac{1}{\epsilon^2})$. For various datasets, they observe that their method outperforms the state-of-the-art kernel methods.

In our work we explore a different approach based on the generalized additive model (GAM) framework introduced by Hastie [2]. First notice that the linear SVM is an additive model which predicts the output based on:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_m x_m \quad (1)$$

where, $\text{sign}(Y)$ is the class label and x_i are the features. In contrast, the GAM replaces the parameter terms $\beta_i x_i$ with functions $f(x_i)$

$$Y = \beta_0 + f(x_1) + f(x_2) + \dots + f(x_m) \quad (2)$$

These functions f_i are arbitrary and often nonparametric, thus providing the potential for better fits to data than other methods. Overfitting can be an issue if the class of these functions is large, without suitable regularization. We model these functions f_i using piecewise linear functions of k segments. This particular choice has attractive properties which allow for efficient discriminative learning algorithms and is strictly more general than a linear SVM. This classifier is similar to a classifier which maps each dimension into k dimensional space but with significant differences, which corresponds to a different regularization. We show that a recently proposed stochastic gradient descent (pegasos [10]) can be adapted to train such models, with similar cost per iteration compared to the training a linear SVM on the input data, and convergence rates.

In our experiments on the Ratsch Dataset [9] the piecewise linear classifier consistently outperforms the linear SVM but is still not close to the best results using rbf kernel on some datasets. Even though the classifier can learn arbitrary functions in each dimension it still remains a additive model over the initial features and thereby cannot learn functions of dependent features. To this end we train models on random linear projections of the data. Random features have been recently shown to have attractive sparseness properties and we can leverage on those. Notice that a linear SVM trained on linear random projection essentially has the same representation power as the original features as the final classifier is still linear in the input features. The piecewise linear classifier on the other hand gains representation power with the addition of projected features. We experiment with two schemes of generating the random matrix to project the data, and show that the piecewise linear functions trained on those perform better than one learned on the original features and as well as the rbf kernel SVMs.

A. Linear and Piecewise Linear

In this section we introduce our classifier and use it with the $L1$ loss (hinge loss) similar to the one used with SVMs. We first begin with a review of linear support vector machines for classification. Given labeled training data of the form $\{(y_i, \mathbf{x}_i)\}_{i=1}^N$, with $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \mathcal{R}^n$, the C-SVM formulation [?] finds a hyperplane which best separates the

rates. Thus about $O(1/\epsilon)$, iterations are needed to be ϵ close to the optimal. The projection step can also be seen as the projection step corresponding to the modified problem. The cost per iteration is now the time to compute the subgradient, which takes $O(dk)$ dot products as the \mathbf{x} are sparse. In addition the computation of $\hat{\mathbf{w}}'\mathbf{H}$ and $\hat{\mathbf{w}}'\mathbf{H}\hat{\mathbf{w}}'$ takes $O(kd)$ time. Thus the overall cost per iteration is $O((k+l)d)$, compared to $O(ld)$ for linear kernel SVM version of pegasos.

B. Comparison to CVX

Figure 1 compares the running time of the stochastic descent algorithm to one trained using an optimization package like CVX[1]. Even the unoptimized version in matlab is atleast an order of magnitude faster to train and gives similar performance to the optimal. The stochastic gradient descent method scales with the feature dimension much better. In the experiments we fix the number of iterations $T = 1000$ and the number of piecewise linear segments $k = 10$.

III. EXPERIMENTAL RESULTS

We use the benchmark datasets of Ratsch [9], which reports the classification accuracy of various methods on it. RBF kernel SVMs usually are close to best performance on all the datasets and we use this as a baseline comparison. A linear SVM baseline is obtained by optimizing the λ parameter using cross validation. We learn piecewise linear classifier using $k = 10$ segments per coordinate and run the stochastic gradient descent algorithm for $T = 1000$ iterations. The performance of all the three methods averaged using 10 fold cross validation are shown in Figure 2. The performance of the piecewise linear is with the variance or better than the linear kernel SVM. However of some datasets the piecewise linear is still much worse than the rbf kernel SVM. The piecewise linear function still cannot learn arbitrary combinations of the feature space (e.g. xor data) as it is still an additive model. We can overcome some of this limitations using randomly projected features, which we discuss in the next section.

A. Random Projection Features

The idea behind this is to augment the feature space by using features based on arbitrary random projections. Ideally we would like to add features based on all possible feature combinations, but this makes the problem much bigger. The idea is behind random projection is that we may be able to get a sample of these combination features if the set of 'good' directions are large. We explore two different schemes of generating the random matrix which we are as follows:

- 1) Rand 1 : generates a random matrix $R^{d \times k}$ to project the input data $R^d \rightarrow R^k$, where each entry is drawn i.i.d. uniformly from $[-1, 1]$. Each row of the matrix is then L_2 normalized. Usually $k > d$.
- 2) Rand 2 : Each row is generated by randomly picking two dimensions and generating a random 2 dimensional vector to project the data.

Figure 2 shows the performance of the random features on 4 datasets where the difference between the rbf and piecewise

linear is significant. The performance of as a function of the number of projected dimensions is shown in Figure 4. Emprically Rand 2 performs better than Rand 1, but they are similar when the number of projected dimensions are large. The random features are able to achieve accuracies comparable to the best. However the projected features can be upto 3 times the original feature dimension.

Figure 3, shows why the random projection features work. The data is highly non separable and a linear SVM has an error rate of 47% on this dataset, while an rbf kernel SVM has only 11.5%. The random projection features with the piecewise linear model achieves 11.1% error rate. The piecewise linear functions allow one to model the multimodality of the data which can be seen from the function learned in the projected dimension each dimension.

IV. CONCLUSION

We introduced a generalization of linear SVMs and showed that an exiting online stochastic gradient method can be adapted to train such models. The generalization outperforms linear SVMs in many benchmark datasets. We also explored the use of random projection features which leads to performance similar to the state of the art on these datasets obtained by kernel-machines. In addition it is significantly faster to classify a new test point as it avoids the complexity of computing the kernel function between the data and all the support vectors. Essentially it is as fast as the linear classifier independent of k as the mapped features are sparse. The parameter k allows us to tradeoff the training testing speed and the representation power of the classifier.

ACKNOWLEDGMENT

This paper is part of ongoing research with Alex Berg. See our paper [6] in which we show that SVM classifiers based on sum kernels can be approximated using piecewise linear functions in each dimension which allows very efficient classification.

REFERENCES

- [1] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming (web page and software). <http://stanford.edu/~boyd/cvx>. April 2008.
- [2] T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986.
- [3] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008.
- [4] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*.
- [5] Thorsten Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, New York, NY, USA, 2006. ACM.
- [6] Subhransu Maji, Alexander Berg, and Jitendra Malik. Classification using intersection kernel support vector machine is efficient. In *CVPR*, 2008(To Appear).
- [7] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.
- [8] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. *NIPS*, 2007.
- [9] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001. also NeuroCOLT Technical Report NC-TR-1998-021.

dataset number	1	2	3	4	5	6	7	8	9	10	11	12	13
dimension	2	9	8	9	20	13	18	20	60	5	3	20	21
train time(cvx)	5.63	13.99	6.21	125.08	174.72	11.62	45.07	15.21	493.54	2.88	6.14	15.75	17.07
train time(subg)	1.12	1.09	1.34	1.51	1.78	1.14	2.39	1.55	2.95	1.00	0.92	1.57	1.58
test error(cvx)	0.31	0.29	0.25	0.32	0.24	0.17	0.10	0.04	0.07	0.08	0.23	0.03	0.12
test error(subg)	0.30	0.28	0.25	0.32	0.24	0.17	0.08	0.04	0.08	0.08	0.23	0.03	0.12

Fig. 1. Training times and error rates of the directly optimized(cvx) and the stochastic gradient descent algorithm(subg) on various datasets. The numbers numbers are summed over 5 different splits of the test and training sets. Dimension denotes the dimension of the original feature space.

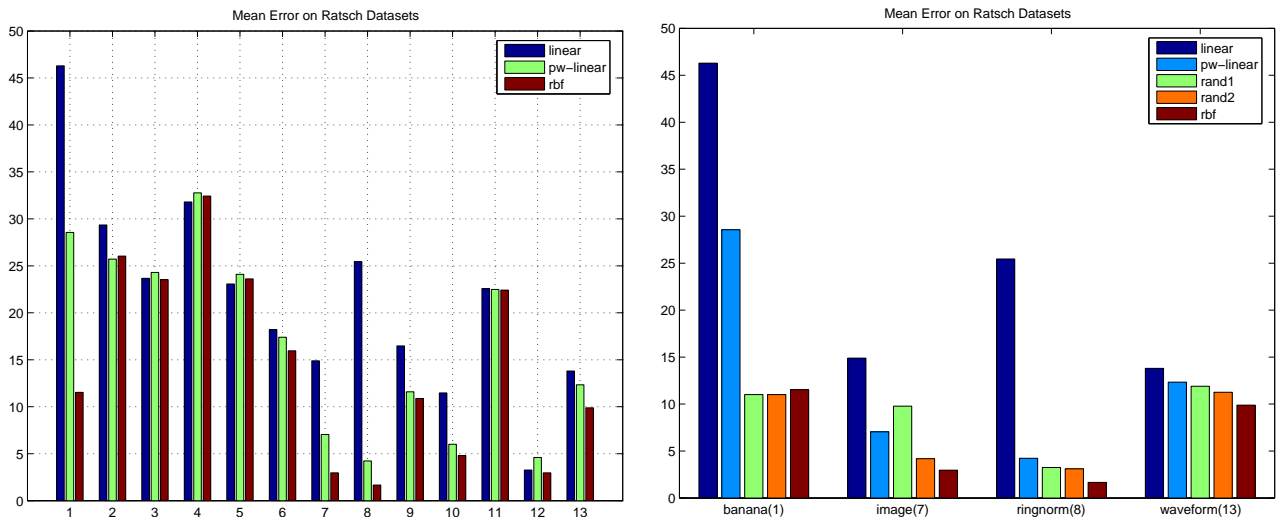


Fig. 2. **Left:** Performance of linear SVM , rbf SVM and piecewise-linear classifier on various Ratsch Datasets[9]. **Right:** Performance of random features on 4 datasets where the difference between the piecewise linear and rbf SVM is significant. (The number in brackets is the index of dataset in the left plot). Random projection features when used with piecewise-linear classification scheme performs close to the rbf SVM.

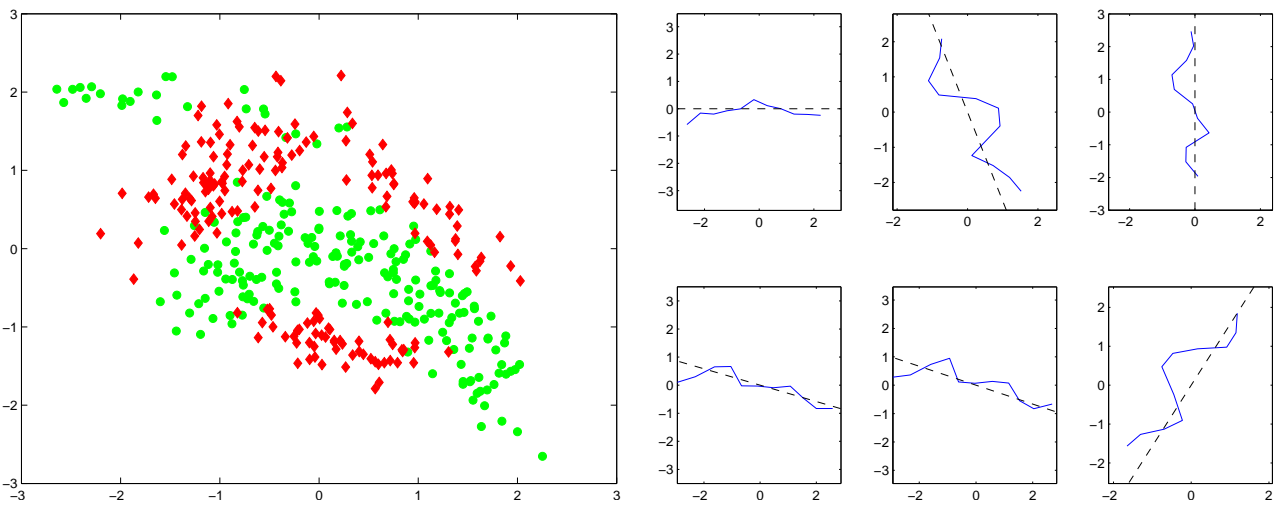


Fig. 3. **Left:** The banana dataset (green dots are positive and red dots are negative). **Right:** Piecewise linear function learned in 6 random projection directions (shown in dotted lines). Linear SVM performs poorly with an error rate of 47% as the data is not linearly separable. The piecewise linear projections in 6 random directions performs similar to the rbf kernels with 11.1% error rate learns the multimodal distribution in the vertical direction. In the horizontal directions the learned function is essentially flat as there is no difference between the positive and the negative class.

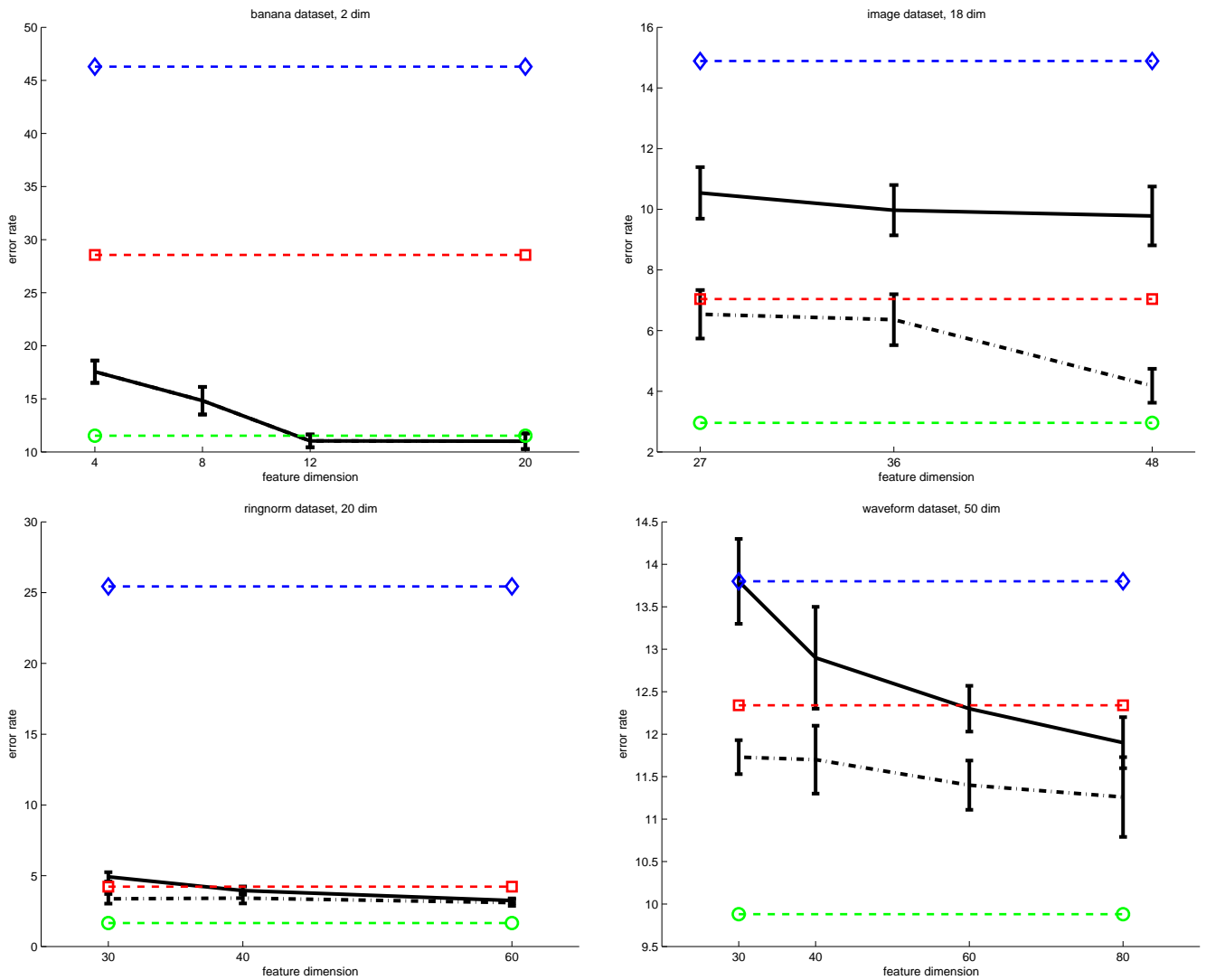


Fig. 4. Accuracy of projection features as a function of the number of dimensions. The solid and the dotted black curves are rand1 and rand2 projection features respectively. The blue, red and green curves are the performance of linear, piecewise-linear and rbf kernel SVM respectively on the input features. Note that on banana dataset rand1 is same as rand2 as the input feature is 2 dimensional.

[10] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pages 807–814, New York, NY, USA, 2007. ACM.