Linearized Smooth Additive Classifiers

Subhransu Maji

Toyota Technological Institute at Chicago,
Chicago, IL 60637, USA
smaji@ttic.edu

Abstract. We consider a framework for learning additive classifiers based on regularized empirical risk minimization, where the regularization favors "smooth" functions. We present representations of classifiers for which the optimization problem can be efficiently solved. The first family of such classifiers are derived from a *penalized spline* formulation due to Eilers and Marx, which is modified to enabled linearization. The second is a novel family of classifiers that are based on classes of *orthogonal basis* functions with *othogonal derivatives*. Both these families lead to explicit feature embeddings that can be used with off-the-shelf linear solvers such as `LIBLINEAR` to obtain additive classifiers. The proposed family of classifiers offer better trade-offs between training time, memory overhead and classifier accuracy, compared to the state-of-the-art in additive classifier training.

1 Introduction

Additive classifiers are a generalization of linear ones and arise naturally in many applications. These include SVM classifiers based on additive kernels, i.e., kernels of the form, $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \sum_i K_i(x_i, y_i)$. Such classifiers frequently arise in computer vision applications where images are represented as a histogram or counts of low-level features, such as color or texture, and a similarity measure, such as the intersection kernel or the χ^2 kernel [2–4] is used to compare them. Although these non-linear kernels provide significant improvements in accuracy over their linear counterparts, it often comes at the expense of higher computational and memory requirements during training and testing.

In recent years, methods for training kernel SVMs that are based on training linear SVMs on feature maps that approximately preserve the kernel dot product have become popular. This includes the work of Rahimi and Recht [5] who proposed such feature maps for shift-invariant kernels, such as the Gaussian kernel. For additive kernels the scheme is easier since the one dimensional decomposition of the kernel allows independent computation of feature maps in each dimension. This direction has been explored by us in our earlier work [3] to construct approximate feature embeddings for the min kernel that results in a piecewise linear approximation of the function in each dimension. For γ-homogenous additive kernels [6] Vedaldi and Zisserman [4] propose feature maps that enable similar efficient training. The resulting efficiencies during training

A. Fusiello et al. (Eds.): ECCV 2012 Ws/Demos, Part I, LNCS 7583, pp. 239–248, 2012.

and testing make additive classifiers the classifiers of choice for many computer vision applications.

In this work we revisit the additive modeling literature to obtain feature maps that enable similar efficient training. The Penalized-Spline (P-Spline) formulation due to Eilers and Marx [1] has emerged as a practical approach for training additive models for the regression setting ever since Generalized Additive Models (GAMs) were introduced by Hastie and Tibshirani [7]. However, it does not directly apply to the classification setting, nor does it scale to the size of datasets and features typical in computer vision applications. Nevertheless, we show that with a small modification to the original formulation, one can derive feature maps that can be directly used with fast linear SVM solvers, such as `LIBLINEAR`, to solve the optimization problem in the classification setting. These feature maps inherit the advantages of the P-Spline formulation, which is that it allows explicit control over the smoothness of the estimated function.

The perspective of learning smooth additive classifiers offers a general recipe for learning. Consider a scheme where the functions in each dimension are expanded using an orthonormal basis set, and smoothness is ensured by penalizing the norm of the function derivatives. We identify a family of orthogonal basis functions for which the additive learning problem reduces an equivalent linear problem. These basis functions have an additional property that they are differentiable and have orthogonal derivatives.

Experiments on various image classification datasets show that the proposed techniques can offer orders of magnitude reduction in training time over standard kernel SVM training, often with almost no memory overhead and within a small constant multiple of the time required to train a linear SVM. These classifiers offer better trade-offs between training time, memory overhead and classifier accuracy, compared to the state-of-the-art in additive classifier training.

2 Generalized Additive Models

Given training data, $(\mathbf{x}^k, y^k),\ k = 1, \ldots, m$ with $\mathbf{x}^k \in \mathbb{R}^D$ and $y^k \in \{-1, +1\}$, we are interested in learning functions based on the following optimization problem:

$$\min_{f \in F} \sum_k l\left(y^k, f(\mathbf{x}^k)\right) + \lambda R(f) \tag{1}$$

where, l is a loss function and $R(f)$ is a regularization term. In the classification setting, a commonly used loss function l is the hinge loss function:

$$l\left(y^k, f(\mathbf{x}^k)\right) = \max\left(0, 1 - y^k f(\mathbf{x}^k)\right) \tag{2}$$

When, $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ and $R(f) = \mathbf{w}^T\mathbf{w}$, this reduces to the standard linear SVM formulation. In the additive modeling setting, a typical regularization is the norm of the d^{th} order derivative of the function, *i.e.*, $R(f) = \sum_i \int_{-\infty}^{\infty} f_i^d(t)^2 dt$. Motivated by the analysis in our earlier work [3], we consider representations of the function f for which the optimization problem can be efficiently solved. For further discussion, we assume that the features are one dimensional, because the analysis can be done for each dimension independently for additive functions.

3 Linearized Spline Embeddings

Eilers and Marx [1] proposed a practical modeling approach for GAMs where they represent the functions in each dimension using a relatively large number of uniformly spaced B-Spline bases. The smoothness of these functions is ensured by penalizing the first or second order differences between the adjacent spline coefficients. Let $\boldsymbol{\Phi}(x^k)$ denote the vector with entries $\Phi_i(x^k)$, the projection of x^k on to the i^{th} basis function. The P-Spline optimization problem for the classification setting with the hinge loss function consists of minimizing $c(\mathbf{w})$:

$$c(\mathbf{w}) = \frac{\lambda}{2}\mathbf{w}^T\mathbf{D}_d^T\mathbf{D}_d\mathbf{w} + \frac{1}{n}\sum_k \max\left(0, 1 - y^k\left(\mathbf{w}^T\boldsymbol{\Phi}(x^k)\right)\right) \tag{3}$$

where, $\mathbf{w}$ is a vector of weights for the basis functions representing the underlying function. The matrix $\mathbf{D}_d$ constructs the d^{th} order differences of $\mathbf{w}$, $\mathbf{D}_d\mathbf{w} = \Delta^d\mathbf{w}$. The first difference of $\mathbf{w}$, $\Delta^1\mathbf{w}$, is a vector of elements $w_i - w_{i+1}$. Higher order difference matrices can be computed by repeating the differencing. For a n bases, the difference matrix $\mathbf{D}_1$ is a $(n-1) \times n$ matrix with $d_{i,i} = 1$, $d_{i,i+1} = -1$ and zero everywhere else.

To enable a reduction to the linear case, we modify the matrix $\mathbf{D}_1$ by adding one more row to the top. Now $\mathbf{D}_1$ is a $n \times n$ matrix with $s_{i,i} = 1, s_{i,i-1} = -1$. The resulting difference matrices $\mathbf{D}_1$ and $\mathbf{D}_2 = \mathbf{D}_1^2$ are shown below:

$$\mathbf{D}_1 = \begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \cdots & & \\ & & & -1 & 1 \end{pmatrix}, \mathbf{D}_2 = \begin{pmatrix} 1 & & & & \\ -2 & 1 & & & \\ & 1 & -2 & 1 & \\ & & \cdots & & \\ & & & 1 & -2 \quad 1 \end{pmatrix}$$

The first row of $\mathbf{D}_1$ has the effect of penalizing the norm on the first coefficient of the spline bases, which plays the role of regularization in the linear setting (e.g. ridge regression, linear SVMs, etc). Alternatively, one can think of this as an additional basis at left most point whose coefficient is set to zero. The key advantage is that the matrix $\mathbf{D}_d$ is invertible and has a particularly simple form which allows us to linearize the whole system by re-parametrizing $\mathbf{w}$ by $\mathbf{D}_d^{-1}\mathbf{w}$, resulting in the following optimization problem :

$$c(\mathbf{w}) = \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{n}\sum_k \max\left(0, 1 - y^k\left(\mathbf{w}^T\mathbf{D}_d^{-T}\boldsymbol{\Phi}(x^k)\right)\right) \tag{4}$$

Since the whole classifier is linear on the features $\boldsymbol{\Psi}^d(x^k) = \mathbf{D}_d^{-T}\boldsymbol{\Phi}(x^k)$, the underlying additive classifier can be learned by using a fast linear solver on the transformed feature space $\boldsymbol{\Psi}$. The inverse matrices $\mathbf{D}_1^{-T}$ and $\mathbf{D}_2^{-T}$ are both upper triangular – the matrix $\mathbf{D}_1^{-T}$ has entries $s_{i,j} = 1, j \geq i$ and $\mathbf{D}_2^{-T}$ has entries $s_{i,j} = j - i + 1,\ j \geq i$. We will show in Section 5, that this structure can be exploited in a custom solver to further reduce the memory and computational requirements during training.

Let us define the implicit kernel between data points as the dot product of their feature maps $\boldsymbol{\Psi} = \mathbf{D}_d^{-T}\boldsymbol{\Phi}$. For uniformly spaced B-Splines of degree d denoted by $\boldsymbol{\Phi}_d$, with $\mathbf{D}_1$ regularization, the implicit kernel $K_d = \boldsymbol{\Psi}_d^T\boldsymbol{\Psi}_d$, where $\boldsymbol{\Psi}_d = \mathbf{D}_1^{-T}\boldsymbol{\Phi}_d$, resembles the smooth versions of the min kernel $K_{\min}(x, y)$, where $K_{\min}(x, y) = \min(x, y)$, as shown in Figure 1. In fact, in our earlier work [3] we motivated the use of linear spline basis as an approximation to the min kernel. Higher order regularizations lead to features that resemble the truncated polynomial kernels [8, 9] which consist of uniformly spaced knots $\tau_1, \ldots, \tau_n$ and truncated polynomial features, $\Phi_i(x) = (x - \tau_i)_+^p$. However these features are less numerically stable than the B-Spline basis (see [10] for a comparison).

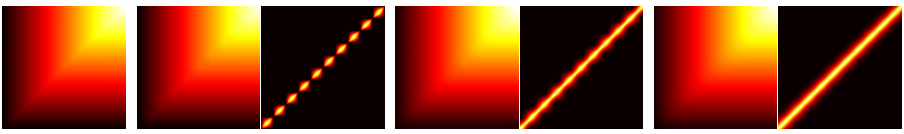

Fig. 1. Spline Kernels. $K_{\min}(x, y), x, y \in [0, 1]$ along with K_d for $d = 1, 2, 3$ corresponding to linear, quadratic and cubic B-Spline basis shown as a heat map (yellow is high, black is low). Using 10 uniformly spaced bases, these kernels closely approximate the min kernel.

4 Generalized Fourier Embeddings

Generalized Fourier expansion of the functions in each dimension provides an alternate way of fitting additive models. Let $\Psi_1(x), \Psi_2(x), \ldots, \Psi_n(x)$ be an orthogonal basis system in the interval $[a, b]$, wrt. a weight function $w(x)$, i.e., we have $\int_a^b \Psi_i(x)\Psi_j(x)w(x)dx = 0, i \neq j$. Given a function $f(x) = \sum_i a_i\Psi_i(x)$, the regularization can be written as:

$$\int_a^b f^d(x)^2 w(x)dx = \int_a^b \left(\sum_{i,j} a_i a_j \Psi_i^d(x)\Psi_j^d(x) \right) w(x)dx$$

Consider an orthogonal family of basis functions which are differentiable and whose derivatives are also orthogonal. One can normalize the basis such that $\int_a^b \Psi_i^d(x)\Psi_j^d(x)w(x)dx = \delta_{ij}$. In this case the regularization has a simple form:

$$\int_a^b f^d(x)^2 w(x)dx = \int_a^b \left(\sum_{i,j} a_i a_j \Psi_i^d(x)\Psi_j^d(x) \right) w(x)dx = \sum_i a_i^2$$

One again the problem of learning a regularized additive classifier reduces to a linear classifier in the embedded space $\boldsymbol{\Psi}(x)$. We identify two such bases:

Trigonometric Basis. The classic trigonometric basis functions:

$$\{1, \cos(\pi x), \sin(\pi x), \cos(2\pi x), \sin(2\pi x), \ldots\} \tag{5}$$

are orthogonal in $[-1, 1]$, wrt. the weight function $w(x) = 1$. The derivatives are also in the same family (except the constant function), hence are also orthogonal. The normalized feature embeddings for $d = 1, 2$ are shown in Table 1.

Hermite Basis. Hermite polynomials are an orthogonal basis system with orthogonal derivatives wrt. the weight function $e^{-x^2/2}$. Using the following identity:

$$\int_{-\infty}^{\infty} H_m(x) H_n(x) e^{-x^2/2} dx = \sqrt{2\pi} n! \delta_{mn} \tag{6}$$

and the property that $H'_n = nH_{n-1}$ (Apell sequence), one can obtain closed form features for $d = 1, 2$ as shown in Table 1. It is also known that the family of polynomial basis functions that are orthogonal with orthogonal derivatives belong to one of three families: Jacobi, Laguerre or Hermite [11]. The extended support of the weight function of the Hermite basis makes them well suited for additive modeling.

Although both these bases are complete, for practical purposes one can approximate the scheme using the first few basis functions. The quality of approximation depends on how well the underlying function can be approximated by these chosen bases. For e.g., low degree polynomials are better represented by Hermite basis.

Table 1. Trigonometric and Hermite embeddings $\boldsymbol{\Psi}^d$ penalizing the d^{th} derivative

Trigonometric $x \in [-1, 1]$, $w(x) = 1$	Hermite $x \in N(0, 1)$, $w(x) = e^{-x^2/2}$
$\Psi^1_n(x) = \{\frac{\cos(n\pi x)}{n}, \frac{\sin(n\pi x)}{n}\}$	$\Psi^1_n(x) = \frac{H_n(x)}{\sqrt{nn!}}$
$\Psi^2_n(x) = \{\frac{\cos(n\pi x)}{n^2}, \frac{\sin(n\pi x)}{n^2}\}$	$\Psi^2_1(x) = \Psi^1_1(x)$, $\Psi^2_n(x) = \frac{H_n(x)}{\sqrt{n(n-1)n!}}$, $n > 1$

5 Learning Additive Classifiers

The proposed embeddings can be used with a fast linear SVM solver to train the underlying additive classifier. However, when the embedded feature space is large there can be a significant memory overhead in storing these features. For better memory efficiency, one could compute the embeddings "on the fly", i.e., in the inner loop of the training algorithm. This scheme is particularly attractive for the B-spline basis since it is relatively cheap to compute the embeddings. Moreover, the sparsity structure of the basis functions and the regularization can be exploited to further reduce the computational and memory overhead.

Most learning methods are sequential – they repeatedly evaluate the classifier at a point and update the classifier if the prediction is incorrect. The number of classifier evaluations, $\mathbf{w}^T\mathbf{D}_d^{-T}\boldsymbol{\Phi}(x)$, can be significantly larger than the number of updates. Hence, it is computationally efficient to maintain $\mathbf{w}_d = \mathbf{D}_d^{-1}\mathbf{w}$, and use sparse vector multiplication to evaluate the classifier. Updates to the weight vector $\mathbf{w}$ and $\mathbf{w}_d$ are of the form:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\mathbf{D}_d^{-T}\boldsymbol{\Phi}(x^k), \qquad \mathbf{w}_d \leftarrow \mathbf{w}_d - \eta\mathbf{L}_d\boldsymbol{\Phi}(x^k) \tag{7}$$

Where η is a step and $\mathbf{L}_d = \mathbf{D}_d^{-1}\mathbf{D}_d^{-T}$. Unlike the matrix $\mathbf{D}_d^T\mathbf{D}_d$, the matrix $\mathbf{L}_d$ is dense. Hence, updates to $\mathbf{w}_d$ can change all its entries. Even though $\mathbf{L}_d$ is dense, one can compute $\mathbf{L}_d\boldsymbol{\Phi}(x)$ in $2dn$ steps instead of n^2 steps by exploiting the structure of $\mathbf{D}_d^{-T}$. This can be done by initializing $a_i = \Phi_i(x)$, and then repeating step A d times, followed by step B d times, to compute $\mathbf{L}_d\boldsymbol{\Phi}(x)$.

$$\text{Step A} : a_i = a_i + a_{i+1}, i = n-1 \text{ to } 1$$
$$\text{Step B} : a_i = a_i + a_{i-1}, i = 2 \text{ to } n$$

For input features that are sparse and non-negative, which often arise in "bag-of-words" representations of text documents or images, it is important to preserve the sparsity of the features in the embeddings for computational and memory efficiency. Formally, we need the property that $\boldsymbol{\Psi}(0) = \mathbf{0}$. For the B-Spline basis, one can achieve this by removing basis functions that have support at 0. For the generalized Fourier features, one could consider an expansion using only the bases that evaluate to zero when the input is zero, i.e., $\Psi(0) = 0$.

6 Experiments

Often on large datasets consisting of very high dimensional features, to avoid the memory bottleneck, one may compute the embedding in the inner loop of the training algorithm. We call this the "online" method. We modify `LIBLINEAR` to enable this online computation, but other solvers such as `PEGASOS` [12], which was used in our previous work [3], can also be easily modified to do the same. The custom solver allows us to exploit the sparsity of embeddings (Section 5).

A practical regularization is $\mathbf{D}_0 = \mathbf{I}$ with the B-Spline embeddings, where $\mathbf{I}$ is the identity matrix, which leads to sparse features. This makes it difficult to estimate the weights for basis functions which have few data points, but one can use a higher order B-Spline basis to somewhat mitigate this problem.

We present image classification experiments on two image datasets, `MNIST` [13] and Daimler Chrysler (`DC`) pedestrians [14]. On these datasets, SVM classifiers based on histogram intersection kernel outperform linear SVM classifiers [3, 15], when used with features based on a spatial pyramid of histogram of oriented gradients [2, 16]. The `MNIST` dataset has 60K instances and the features are 2172 dimensional and dense, leading to 130 million non-zero entries. The `DC` dataset has three training sets and two test sets. Each training set has 19.8K instances and the features are 656 dimensional and dense, leading to 13 million non-zero entries. These sizes are typical of image datasets, and training kernel SVM classifiers requires several hours on a single machine.

Effect of B-Spline Embedding Parameters. Table 2 shows the accuracy and training times as we vary the number of basis functions, regularization $\mathbf{D}_d, (d = 0, 1, 2)$, and the B-Spline degree $\in \{1, 2, 3\}$ on the first split of the DC pedestrian dataset. We set $C = 1$ and the bias term $B = 1$ for training all the models. On this dataset, we find that $\mathbf{D}_0$ and $\mathbf{D}_1$ regularization is as accurate and significantly faster than $\mathbf{D}_0$. This suggests that first order smoothness is sufficient for this dataset. In addition, $\mathbf{D}_0$ regularization leads to sparse features, which can be directly used with any linear solver that can exploit this sparsity. The training time for B-Splines scales sub-linearly with the number of basis functions, hence better fits can be obtained without significant loss in efficiency.

Table 2. The effect of spline parameters on training time/test accuracy on `DC` dataset

Spline	Regularization $\mathbf{D}_0$	$\mathbf{D}_1$	$\mathbf{D}_2$
	5 basis functions		
Linear	**6.60s** (89.55%)	**20.27s** (89.68%)	**41.60**s (89.93%)
Quadratic	8.74s (**90.45**%)	30.47s (**90.20**%)	80.25s (**89.94**%)
Cubic	11.68s (90.03%)	49.85s (89.93%)	143.50s (88.57%)
	10 basis functions		
Linear	**5.61s** (90.42%)	**23.06**s (90.86%)	**77.99**s (**89.43**%)
Quadratic	8.10s (**90.69**%)	29.97s (**90.73**%)	126.03s (89.23%)
Cubic	11.59s (90.48%)	42.26s (90.67%)	193.47s (89.14%)
	20 basis functions		
Linear	**5.96s** (90.23%)	**32.43**s (**91.20**%)	**246.87**s (**89.06**%)
Quadratic	7.26s (90.34%)	34.99s (91.10%)	328.32s (88.89%)
Cubic	10.08s (**90.39**%)	42.88s (91.00%)	429.57s (88.92%)

Effect of Fourier Embedding Parameters. Table 3 shows the accuracy and training times for various Fourier embeddings on `DC` dataset. The raw features, are first normalized so that the data in each dimension $\in [-1, 1]$. The experiments are performed by precomputing the embeddings and using `LIBLINEAR` to train various models, as it is relatively more expensive to compute the embeddings online. The training times and accuracies are similar to that of B-Spline models.

Comparison of Various Additive Models. Table 4 shows the accuracy and training times of various additive models compared to linear and the more expensive min kernel SVM on all the 6 combinations of training and test sets of the `DC` dataset. The optimal parameters were found on the first training and test set. The additive models are up to 50× faster to train and are as accurate as the min kernel SVM. The B-Spline additive models significantly outperform a linear SVM, require a small additional training time and almost *no* memory overhead.

Table 5 shows the accuracies and training times of various additive models on the `MNIST` dataset using the online method. We train one-vs-all classifiers

Table 3. The effect of Fourier parameters on training time/test accuracy on `DC` dataset

	Trigonometric				Hermite			
	$d = 1$		$d = 2$		$d = 1$		$d = 2$	
#Basis	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
1	88.94%	**07.0s**	88.94%	**07.0s**	84.17%	**02.8s**	84.17%	**02.8s**
2	89.59%	10.2s	89.64%	10.2s	88.01%	04.6s	88.01%	04.6s
3	88.99%	12.7s	89.77%	12.8s	88.22%	07.7s	88.70%	09.9s
4	**89.77**%	16.0s	**89.84**%	15.9s	**89.00**%	12.6s	**89.05**%	11.9s

for each digit, and the classification scores are normalized to $\in [0, 1]$ by Platt's scaling. During testing, each example is assigned the label of the classifier with the highest response. The optimal parameters for training were found using 2-fold cross validation on the training set. Once again, the additive models significantly outperform the linear classifier and match the accuracy of the min kernel SVM, while being 50× faster. The spline embeddings once again perform the best, requiring a small multiple of the time required to train a linear SVM, without requiring additional memory since the features are computed online.

Table 4. Training time/test accuracy of various additive classifiers on `DC` dataset

Method	Test Accuracy	Training Time	
SVM (linear) + `LIBLINEAR`	81.49 (1.29)	3.8s	
SVM (min) + `LIBSVM`	89.05 (1.42)	363.1s	
		online	batch
B-Spline ($\mathbf{D}_0$, Linear, $n = 05$)	88.51 (1.35)	**5.9**s	-
B-Spline ($\mathbf{D}_0$, Cubic, $n = 05$)	89.00 (1.44)	10.8s	-
B-Spline ($\mathbf{D}_1$, Linear, $n = 10$)	**89.56** (1.35)	17.2s	-
B-Spline ($\mathbf{D}_1$, Cubic, $n = 10$)	89.25 (1.39)	19.2s	-
Fourier ($d = 1,\ n = 4$)	88.44 (1.43)	159.9s	12.7s (4× memory)
Hermite ($d = 1,\ n = 4$)	87.67 (1.26)	35.5s	12.6s (4× memory)

Table 5. Training time/test error of various additive classifiers on `MNIST` dataset

Method	Test Error	Training Time
SVM (linear) + `LIBLINEAR`	1.44%	6.2s
SVM (min) + `LIBSVM`	0.79%	$\sim$ 2.5 hours
B-Spline ($\mathbf{D}_0$, Linear, $n = 20$)	0.88%	31.6s
B-Spline ($\mathbf{D}_0$, Cubic, $n = 20$)	0.86%	51.6s
B-Spline ($\mathbf{D}_1$, Linear, $n = 40$)	**0.81**%	157.7s
B-Spline ($\mathbf{D}_1$, Cubic, $n = 40$)	0.82%	244.9s
Hermite ($d = 1, n = 4$)	1.06%	358.6s

7 Comparison to Previous Work

The spline embeddings proposed in this work are a generalization of our earlier work [3](MB). The ϕ_2-sparse and ϕ_2 classifiers proposed in MB are equivalent to the embeddings obtained by using a linear spline basis with $\mathbf{D}_0$ and $\mathbf{D}_1$ regularization respectively. The family of spline embeddings offers finer-grained control over the training time and accuracy than MB. For example, one can use $\mathbf{D}_0$ regularization with quadratic B-Spline basis to obtain a classifier with intermediate accuracy and training time as the ϕ_2-sparse and ϕ_2 classifiers as seen in Table 4. The other approach related to our work is that of Vedaldi and Zisserman [4] (VZ). We compare our approach to theirs in terms of memory overhead and training time, since the accuracies of all these methods are similar to the exact kernel SVM classifier.

Memory Overhead. The VZ method has similar memory overhead as the Fourier embeddings since both these result in dense embeddings of similar dimension. When used in the offline case, i.e., with precomputed embeddings, this can lead to an order of magnitude increase in memory requirement, which may be impractical for large datasets. In comparison, the spline embeddings have lower memory overhead since the projected features are sparse regardless of the number of basis functions. The Fourier and VZ methods can be easily modified to compute features online to reduce their memory overhead, but this comes at the expense of training time.

Training Time. The training times of Fourier embeddings and VZ are similar, both for the online and offline case, since both these embeddings are dense and involve similar computations. Even though the B-Spline basis can be much higher dimensional, the training time remains small because of the optimizations we presented in Section 5. If we restrict ourselves to the online case, which is of practical importance, the training time is dominated by the time taken to compute the embeddings. The B-Spline embeddings are the fastest to compute as they involve fewer arithmetic operations than trigonometric functions or higher order polynomials. The Fourier embeddings or VZ can be sped up using precomputed tables, but are unlikely to be faster than the B-Spline embeddings.

8 Conclusion

Motivated by the additive modeling literature, we propose two families of embeddings that enable efficient learning of additive classifiers. Spline embeddings can be derived by a simple modification of the P-Spline formulation of Eilers and Marx [1] and generalize our earlier work [3]. These classifiers can be trained using a custom solver, with almost no memory overhead, and within a small constant multiple of the training time compared to a linear classifier, and are as accurate as an additive kernel SVM classifier. We also propose a family of generalized Fourier features that can be used with an off-the-shelf linear solver such as `LIBLINEAR` to efficiently train additive classifiers.

Acknowledgements. The work was done when the author was a graduate student at the University of California at Berkeley. The author would also like to thank Alex Berg for helpful discussions.

References

1. Eilers, P., Marx, B.: Generalized linear additive smooth structures. Journal of Computational and Graphical Statistics 11(4), 758–783 (2002)
2. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
3. Maji, S., Berg, A.C.: Max margin additive classifiers for detection. In: ICCV (2009)
4. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. In: CVPR (2010)
5. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: NIPS (2007)
6. Hein, M., Bousquet, O.: Hilbertian metrics and positive definite kernels on probability measures. In: AISTATS (2005)
7. Hastie, T., Tibshirani, R.: Generalized Additive Models. Chapman & Hall/CRC (1990)
8. Pearce, N., Wand, M.: Penalized splines and reproducing kernel methods. The American Statistician 60(3), 233–240 (2006)
9. Wahba, G.: Spline models for observational data, vol. 59. Society for Industrial Mathematics (1990)
10. Eilers, P., Marx, B.: Splines, knots, and penalties. Wiley Interdisciplinary Reviews: Computational Statistics (2005)
11. Webster, M.: Orthogonal polynomials with orthogonal derivatives. Mathematische Zeitschrift 39, 634–638 (1935)
12. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: ICML (2007)
13. LeCun, Y., Cortes, C.: The mnist database of handwritten digits (1998)
14. Munder, S., Gavrila, D.M.: An experimental study on pedestrian classification. IEEE TPAMI 28(11) (2006)
15. Maji, S., Berg, A.C., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: CVPR (2008)
16. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)