L_1 Regularized Projection Pursuit for Additive Model Learning

Xiao Zhang
Center for Advanced Study
Tsinghua University, Beijing, China
xiao-zhang03@mails.tsinghua.edu.cn

Lin Liang Xiaoou Tang
Microsoft Research Asia
Beijing, China
{lliang,xitang}@microsoft.com

Heung-Yeung Shum
Microsoft Corporation
Redmond, WA, USA
hshum@microsoft.com

Abstract

In this paper, we present a L_1 regularized projection pursuit algorithm for additive model learning. Two new algorithms are developed for regression and classification respectively: sparse projection pursuit regression and sparse Jensen-Shannon Boosting. The introduced L_1 regularized projection pursuit encourages sparse solutions, thus our new algorithms are robust to overfitting and present better generalization ability especially in settings with many irrelevant input features and noisy data. To make the optimization with L_1 regularization more efficient, we develop an "informative feature first" sequential optimization algorithm. Extensive experiments demonstrate the effectiveness of our proposed approach.

1. Introduction

Additive model is an efficient method for high-dimensional and nonlinear learning problems. The Basic Additive Model first introduced in [7] approximates the mapping function from the input features $\mathbf{x}(\in \mathbb{R}^M)$ to the response $y(\in \mathbb{R})$ using a sum of component functions of each individual feature x_j. The additive model is interpretable and easy to fit, and also avoids the "curse of the dimension" problem of the non-parametric methods [5]. Additive model has been widely used in regression and classification problems. For example, the popular Adaboost method can be viewed as an approximation to additive modeling [4].

One limitation of the basic additive model is that the interactions between the input features are not considered. Projection pursuit [5] is adopted to solve this problem: the optimal combinations of the input features are fed into the component functions to better interpret the response. These set of additive models are generally called Extended Additive Models. For example, projection pursuit regression [5] was developed to approximate a much richer class of functions. Kullback-Leibler boosting [17] projects high-dimensional data to linear features to maximize the Kullback-Leibler (KL) divergence. The learned classifier is more compact and robust. Neural networks can also be seen as a kind of extended additive model. Although additive models with projection pursuit are more powerful, the model complexity is significantly increased since variable combinations are considered. Thus, when these methods are applied to the problems with many irrelevant variables and noisy data, which usually happens in the real world, they are prone to overfit.

Recently, for the basic additive model learning without projection pursuit, some works have been done to prevent overfitting. The work in [15] explicitly detects outliers and limits their influence on the Adaboost classifier. RegBoost [12] uses the graph Laplacian regularizer to penalize the base classifiers that cut through dense regions. Inspired by the recent success of lasso estimator [24] that uses L_1 regularization to encourage sparse solutions, Ravikumar proposes a sparse additive model (SpAM) [23]. SpAM adopts L_1 constraints to suppress the redundant component functions. All the above methods focus on the additive models without projection pursuit. How to prevent the overfitting introduced by the projection pursuit stage in the extended additive model learning has not been addressed, even though overfitting is more sever due to projection pursuit.

In this paper, we add L_1 regularization into the projection pursuit stage of the additive model learning to encourage the sparsity of the interacted variables. Two new algorithms are developed for regression and classification respectively: sparse projection pursuit regression and sparse Jensen-Shannon Boosting. By using L_1 regularization projection pursuit, the new algorithms are robust to overfitting especially when there are noisy data and many irrelevant variables. The obtained additive model also maintains the good interpretable property of the basic additive model. To make the optimization with L_1 regularization more efficient and stable, an "informative feature first" sequential optimization algorithm is developed. The proposed optimization algorithm is efficient for high dimensional space and general for both parametric and non-parametric cost function. We demonstrate the effectiveness of our algorithms on

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

two difficult problems: face exaggeration and gender classification, good performances are obtained.

The rest of this paper is organized as follows. We introduce the main theory of additive model with projection pursuit in Section 2. L_1 regularized projection pursuit is discussed in Section 3. Section 4 presents two new additive learning algorithms. In Section 5, the applications of our algorithms to face exaggeration and gender classification are presented. After some discussions in Section 6, we conclude the paper in Section 7.

2. Additive Model with Projection Pursuit

2.1. Basic additive model

The additive model is a flexible statistical method for modeling the mean $E(y|\mathbf{x}) = F(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$. Basically, it has the form:

$$F(\mathbf{x}) = \sum_{j=1}^{p} f_j(\mathbf{x}_j, \beta_j) \tag{1}$$

where $f_j(x_j, \beta_j)$ is a component function with parameter β_j. Notice that f_j only depends on the individual feature x_j. That is, in the additive model, each component function $f_j(x_j)$ is estimated using a univariate function, the "curse of dimensionality" difficulty for many nonparametric methods (e.g., smoothing splines) is avoided. Also, it is easy to explain how the response y changes with the input feature x_j from the additive model Equation (1).

2.2. Additive model with projection pursuit

One problem of the basic additive model is that it approximates the mapping by a sum of univariate functions of each individual feature, thus it cannot deal with the interactions among the input features, which usually happens in the real world. To overcome this problem, Projection pursuit is introduced into the additive model: the linear combinations of the features are used as the variables in the component functions. The extended additive model has the form:

$$F(\mathbf{x}) = \sum_{j=1}^{p} f_j(\alpha_j^T \mathbf{x}, \beta_j)) \tag{2}$$

where $\alpha_j = \{a_m^j\}_{m=1}^M$ is the projection vector and is restricted to $\|\alpha_j\|_2 = 1$. Projection pursuit improves the modeling ability of the additive model significantly. With different definitions of the component function f_j, the extended additive models are widely used in many areas.

For regression, if f_j is approximated using a sigmoid function, the model (2) can be viewed as a single hidden layer neural network [9]. If f_j is estimated by a supper smoother, the model (2) is well known as projection pursuit regression [5]. These models form nonlinear functions of the projection pursuit terms, thus they are very general. Comparing with the basic additive model, they can approximate a much richer class of functions.

For classification, if f_j is a "weak leaner", Equation (2) becomes the formulation of Kullback-Leibler boosting [17]. KLBoosting adopts projection pursuit to find the optimal discriminative linear features. The obtained classifier generalizes better than AdaBoost.

To find the optimal parameters of the model (2), backfitting algorithm [7] or a "greedy" forward stepwise [4] approach can be adopted. These algorithms fit a single component function f_j to the data one by one. During each iteration j, the modified versions of the original data y_j is obtained. For example, for regression, y_j is the residue of this iteration, and for boosting, y_j corresponds to the reweighted samples. Then the j-th iteration of additive model fitting is to optimize:

$$\{\alpha_j^*, \beta_j^*\} = \arg\min_{\alpha_j, \beta_j} g(y_j, \alpha_j^T \mathbf{x}, \beta_j) \tag{3}$$

where $g(\cdot)$ is a cost function with different formulations for different problems. For example, for regression, $g(\cdot)$ is usually defined as the mean square error:

$$g(y_j, \alpha_j^T \mathbf{x}, \beta_j) = [y_j - f_j(\alpha_j^T \mathbf{x}, \beta_j)]^2 \tag{4}$$

2.3. The limitations of projection pursuit

Although projection pursuit increases the power of additive model for high-dimensional data analysis, it does have the following disadvantages:

- Overfitting

  Projection pursuit increases the freedom of the additive model, thus when there are many irrelevant features fed into the algorithm, the algorithm tends to overfitting, especially when the training data are noisy.

- Not easy to interpret the model

  There are usually many nonzero entries in each projection vector, thus the model obtained by projection pursuit is difficult to interpret [8], unlike the original additive model (1).

To solve these problems, we introduce L_1 regularization into the projection pursuit stage, as explained in the next section.

3. L_1 Regularized Projection Pursuit

3.1. L_1 regularization

The traditional projection pursuit learning may introduce many irrelevant features into the model to interpret the data, especially when the data are noisy. To overcome this problem, we add a L_1 regularization term to each iteration of

additive model fitting. Then at the j-th iteration, the optimal parameters are found by minimizing:

$$\{\alpha_j^*, \beta_j^*\} = \arg\min_{\alpha_j, \beta_j} [g(y_j, \alpha_j^T \mathbf{x}, \beta_j) + \lambda_j \|\alpha_j\|_1] \quad (5)$$

where $\|\alpha_j\|_1 = \sum_{m=1}^{M} |a_m^j|$ and the parameter $\lambda_j \geq 0$ controls a tradeoff between fitting the data well, and having small parameters.

L_1-norm $\|\alpha_j\|_1$ is a convex relaxation of L_0-norm (the number of non-zero elements) [3]. The function of L_1 regularization is to encourage many entries of projection vector to equal zero. Thus by adding L_1 regularization, the algorithm will avoid introducing many features and those irrelevant features will be suppressed. The model becomes more parsimonious and interpretable. Comparing with L_2-norm regularization that penalizes large features much more than the smaller ones, L_1 regularization also penalizes small terms strongly, thereby L_1 regularization is more effective at identifying relevant features even with a relatively small number of samples [20].

As the convergence of the additive model fitting, the fitting cost $g(\cdot)$ becomes small, if the weight λ_j of L_1 regularization is set as a constant for each iteration of additive model fitting, L_1 regularization will become too strong with the convergence. To avoid this problem, we adaptively adjust λ_j as:

$$\lambda_j = \frac{g(\alpha_j^{(0)}, \beta_j^{(0)})}{\tau}, \quad (6)$$

where $\alpha_j^{(0)}$ and $\beta_j^{(0)}$ are the initial parameters of the j-th iteration of additive model fitting, and τ controls the overall strength of L_1 regularization. The optimal value of τ could be determined by cross-validation. Comparing with the way that keeps λ_j as a constant, more reasonable balance between the data fitting and the model sparseness is achieved.

3.2. "Informative feature first" sequential optimization

To find the optimal solution of Equation (5) is not easy. (To be succinct, we omit the iteration index j in this section). One difficulty is that: the cost function $g(\cdot)$ may be nonparametric (e.g. the KL distance adopted in KLBoosting [17]), thus those gradient-based approaches [13][21] can not be applied. Another difficulty is that: the dimension of the projection α is usually high.

As pointed out in Ce's work [17], since the projection α is a unit vector, given any feature $\mathbf{x} \in \mathbb{R}^M$, there exists a boundary $[a_x, b_x]$, such that $a_x \leq \alpha^T \mathbf{x} \leq b_x$ for $\alpha \in \mathbb{R}^M, \|\alpha\|_2 = 1$, so it is possible to search the optimal projection in a neighborhood of α. Also in Equation (5), once the projection α is fixed, the function parameter β can be obtained by Least Square optimization or gradient descent [8], that is, β can be seen as a function of α. So to optimize Equation (5), we quantize the projection a_m of each feature in a neighborhood $|a_m| \leq H$ and search the optimal solution in such neighborhood. Equation (5) is modified as:

$$\alpha^* = \arg\min_{\alpha} E(\alpha, \beta(\alpha))$$
$$\alpha = \{a_m\}_{m=1}^{M}, |a_m| \leq H, m = 1, \ldots, M \quad (7)$$

where the cost $E(\alpha, \beta(\alpha))$ is:

$$E(\alpha, \beta(\alpha)) = g(y, \alpha^T \mathbf{x}, \beta) + \lambda \|\alpha\|_1 \quad (8)$$

// m: feature index to be searched along
// $[h_1, h_2]$: search range; dh: search step
// $\alpha^0 = \{a_m^0\}_{m=1}^M$: initial projection vector
// a_m^*: optimal projection value; E_m^*: minimum cost

$[a_m^*, E_m^*]$ = *1DSearch*$(m, h_1, h_2, dh, \alpha^0)$
• $a_m^* = a_m^0, E_m^* = E(\alpha^0, \beta(\alpha^0))$,
  $\alpha = \alpha^0$, $K = (h_2 - h_1)/dh$
For i = 0 to K
  • Set $a_m = a_m^0 + i * dh + h_1$ in α
  • Find $\beta^* \rightarrow \min E(\alpha, \beta(\alpha))$
  • Calculate $E(\alpha, \beta^*)$
  If $E(\alpha, \beta^*) > E_m^*$
    $E_m^* = E(\alpha, \beta^*)$, $a_m^* = a_m$
return $[a_m^*, E_m^*]$

Table 1. The pseudo code of 1D search along one feature.

Denote: projection $\alpha = \{a_m\}_{m=1}^M$, cost $E(\alpha, \beta(\alpha))$.
• Initialize:
  • Find feature index i, such that
    $\alpha_i = [0, .., 0, a_i = 1, 0, .., 0] \rightarrow \min E$,
    Set $\alpha^{(0)} = \alpha_i$, $E^* = E(\alpha_i, \beta(\alpha_i))$
  • Search range $H = 1$
  • Search step $dH = 0.1$, $dh = 0.01$
For k = 1 to K
  • $\alpha^{(k)} = \alpha^{(k-1)}$
  // *find informative feature*
  • Optimal feature index: $m^* = -1$
  • Optimal projection value: $a^* = 0$
  For m = 1 to M
    • $[a_m^*, E_m^*]$=*1DSearch*$(m, -H, H, dH, \alpha^{(k)})$
    If $E_m^* > E^*$
      $m^* = m$, $E^* = E_m^*$, $a^* = a_m^*$
  // *fine search on informative feature* m^*
  • $h_1 = a^* - dH$, $h_2 = a^* + dH$
    $[a_{m^*}^*, E^*]$=*1DSearch*$(m^*, h_1, h_2, dh, \alpha^{(k)})$
  • Update $a_{m^*}^{(k)} = a_{m^*}^*$ in $\alpha^{(k)}$
Output $\alpha^* = \alpha^{(K)} / \|\alpha^{(K)}\|$

Table 2. The pseudo code of "Informative feature first" sequential optimization. The sub-function *1DSearch()* is shown in Table 1.

To optimize Equation (7) is still not easy. In KLBoosting [17], Ce and Shum just did 1D optimization: quantize the projection a_m in a neighborhood, then find the optimal value a_m^* to minimize the cost (8). Such 1D optimization is sequentially performed along each feature in the feature set until converge. Since in Ce's algorithm, the projection coefficients of all features will be sequentially modified, it is possible to get nonzero entries for those irrelevant features in the projection vector. This will decrease the generalization ability of the learning algorithm.

To solve this problem, we improve Ce's algorithm by introducing an "informative feature first" mechanism: 1D optimization is performed along a selected feature that would maximally reduce the fitting cost as Equation (8). More specifically, our approach contains two steps:

1. Find an informative feature by coarsely quantizing the projection a_m along each feature and selecting a feature that maximally minimizes Equation (8).

2. Perform 1D optimization along the selected informative feature at a fine quantization level.

These two steps are carried out iteratively until converge. To initialize the projection α, we find a feature m that maximally reduces the cost (8), then set the initial projection vector $\alpha^{(0)} = [0, .., 0, a_m = 1, 0, ..0]$. In practice, we set the search range $H = 1$. We summarize the 1D optimization step as a sub-function in Table 1, and the pseudo code of the whole algorithm is shown in Table 2.

Our algorithm is efficient to suppress those irrelevant features. The "informative feature first" algorithm is actually a feature selection mechanism: the projection vector is first modified along the most informative coordinate. In this way, our algorithm reduces the possibility to select those irrelevant features to the model. Also notice that, our optimization algorithm is suitable for both parametric and non-parametric cost function. It is more general than those gradient-based approaches [13][21].

4. Additive Model Learning with L_1 Regularized Projection Pursuit

In this section, we apply our L_1 regularization projection pursuit algorithm to additive model learning and develop two efficient algorithms for regression and classification respectively.

4.1. Sparse projection pursuit regression

Projection pursuit regression (PPR) [5] extends the basic additive model to allow interactions among the variables by projection pursuit. It has the form as Equation (2), here the component function f_j is a super smoother (e.g., a cubic smoothing spline or kernel smoother) and y is a real value.

Given:$(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^n, y^n), \mathbf{x}^i \in \mathbb{R}^M, y^i \in \mathbb{R}$
1. Set $r_1^i = y^i$, $\lambda_1 = \sum_{i=1}^n (r_1^i)^2/\tau$
2. For j = 1,... minimize
$R_j^2 = \sum_{i=1}^n [r_j^i - f_j(\alpha_j^T \mathbf{x}^i, \beta_j)]^2 + \lambda_j \|\alpha_j\|_1$
by the algorithm in Table 2
3. Update $r_{j+1}^i = r_j^i - f_j(\alpha_j^T \mathbf{x}^i, \beta_j)$
$\lambda_{j+1} = \sum_{i=1}^n (r_{j+1}^i)^2/\tau$
repeat step 2 until R_j becomes small.

Table 3. The flowchart of SpPPR.

Comparing with the basic additive regression model [7], PPR is able to approximate a much richer class of functions, such as the product $x_1 \cdot x_2$. But such flexibility also brings some disadvantages such as overfitting and interpretation difficulty as explained in section 2.3.

Here we propose a sparse projection pursuit regression (SpPPR) algorithm that adopts L_1 regularized projection pursuit for each component function f_j's estimation. The detailed form of Equation (5) becomes:

$$\{\alpha_j^*, \beta_j^*\} = \arg\min_{\alpha_j, \beta_j} \{[r_j - f_j(\alpha_j^T \mathbf{x}; \beta_j)]^2 + \lambda_j \|\alpha_j\|_1\} \quad (9)$$

where $r_j = y - \sum_{k \neq j} f_k(\alpha_k^T \mathbf{x}; \beta_k)$ is the residue of the j-th iteration.

To optimize Equation (9), the usual approach is to alternatively fix the projection coefficients α_j or the function parameters β_j, then to optimize the other. The disadvantage of this approach is that when fixing β_j, α_j could hardly be updated towards to the optimal solution of Equation (9). Our "informative feature first" optimization alleviates this problem and converges to the optimal solution quickly. The pseudo code of SpPPR is summarized in Table 3.

The SpPPR can identify relevant variables even if the training data are noisy. The obtained regression model is more interpretable and has good generalization ability, as shown in the following toy problem.

4.1.1 A toy problem

We test our SpPPR on the following 5-dimensional function:

$$y = x_1 * x_2 + \epsilon \quad (10)$$

with three irrelevant variables x_3, x_4, x_5. Here $x_j, j = 1, \ldots, 5$ is uniformly distributed in $[0, 1]$ and $\epsilon \sim \mathcal{N}(0, 1)$ is Gaussian noise. Notice that, this function can also be written as:

$$y = \frac{(x_1 + x_2)^2}{4} - \frac{(x_1 - x_2)^2}{4} + \epsilon \quad (11)$$

which can be modeled by additive model with projection pursuit as Equation (2), but is not suitable for the basic additive model (1).

Using this function, we generate 300 samples with noise for training, and 6000 samples without noise for testing. We

τ	MSE	♯ ZEROS	Selected variables
5	0.1935	7	$\{x_1\}$
10	0.0334	6	$\{x_1, x_2\}$
15	0.0223($\surd$)	6	$\{x_1, x_2\}$
30	0.0291	5	$\{x_1, x_2, x_5\}$
100	0.032	5	$\{x_1, x_2, x_5\}$
∞	0.1674	0	$\{x_1, x_2, x_3, x_4, x_5\}$

Table 4. Mean square errors, zero entries, and the selected variables obtained by changing the weight τ of L_1 regularization in SpPPR. $\tau = \infty$ corresponds to PPR.

Projection	$[a_1, a_2, a_3, a_4, a_5]$ estimated by SpPPR
α_1	[1 1 0 0 0]
α_2	[-1 1 0 0 0]
	$[a_1, a_2, a_3, a_4, a_5]$ estimated by PPR
α_1	[0.67 0.67 -0.01 -0.04 -0.08]
α_2	[-1.44 1.87 0.22 0.35 0.16]

Table 5. The projection vectors in two component functions estimated by SpPPR and the original PPR respectively.

remove the noise ϵ from the testing data to more clearly evaluate if the learned regression model is close to the ground truth. An additive model with two smooth spline component functions:$y = \sum_{j=1}^{2} f_j(\alpha_j^T \mathbf{x}), \alpha_j = \{a_m^j\}_{m=1}^5$ is fitted to the training data.

Table 4 shows the testing results of SpPPR by changing the weight τ of L_1 regularization in Equation (6). An infinite τ corresponds to the traditional PPR. As we can see, when τ becomes larger (L_1 regularization is reducing), more variables will be selected to the model including those irrelevant ones. When $\tau = 15$, the model obtained by SpPPR has the smallest fitting error that is much less than the traditional PPR ($\tau = \propto$). The projection vectors of the corresponding model are shown at the top of Table 5 which are the same as the ground truth (Equation (11)). The bottom of Table 5 also shows the projection vectors estimated by the traditional PPR. The entries of those irrelevant variables are nonzero. It is hard to tell how the input variables influence the changing of the response y.

It is clear that when the data is noisy and there are many irrelevant variables, SpPPR has the ability to capture the underlying structure of the data, while the traditional PPR tends to include irrelevant variables that decreases its generalization and interpretation ability.

4.2. Sparse Jensen-Shannon boosting

To improve the feature selection stage in Adaboost [26], KLBoosting [17] pursues the discriminative features by maximizing the projected Kullback-Leibler divergence. Jensen-Shannon Boosting [10] improves the numerical stability of KLBoosting by using Jensen-Shannon divergence. But when there are undistinguishable samples in the training data, these algorithms tend to separate them by introducing irrelevant features. Consequently these irrelevant features may damage the classifier's discrimination on those originally distinguishable samples.

Given:$(x^1, y^1), \ldots, (x^n, y^n), x^i \in \mathbb{R}^M, y^i \in \{-1, 1\}$
- Initialize: $W_1(x_+^i) = \frac{1}{N_+}, W_1(x_-^i) = \frac{1}{N_-}$

For $j = 1$ to J
- Learn projection vector α_j by minimizing Equ.(12) using the algorithm in Table 2.
- Update $W_{j+1}(x_+^i)$ and $W_{j+1}(x_-^i)$.

Output classifier:
$F(x) = sign[\sum_{i=1}^{j} \frac{1}{2} \log \frac{h_i^+(\alpha_i^T \mathbf{x})}{h_i^-(\alpha_i^T \mathbf{x})}]$

Table 6. The flowchart of SpJSBoost.

To overcome this problem, we develop a SpJSBoost algorithm: L_1 regularization is incorporated into the feature pursuit stage of JSBoost. The optimal projection is found by optimizing:

$$\alpha_j^* = \arg\min_{\alpha_j} [-SJS(\alpha_j^T \mathbf{x}) + \lambda_j \|\alpha_j\|_1] \qquad (12)$$

where $SJS(\alpha_j^T \mathbf{x})$ is the Jensen-Shannon divergence:

$$SJS(\alpha_j^T \mathbf{x}) = \int \{h_j^+(\alpha_j^T \mathbf{x}) \log \frac{h_j^+(\alpha_j^T \mathbf{x})}{\frac{1}{2}[h_j^+(\alpha_j^T \mathbf{x}) + h_j^-(\alpha_j^T \mathbf{x})]} + h_j^-(\alpha_j^T \mathbf{x}) \log \frac{h_j^-(\alpha_j^T \mathbf{x})}{\frac{1}{2}[h_j^+(\alpha_j^T \mathbf{x}) + h_j^-(\alpha_j^T \mathbf{x})]} \} d(\alpha_j^T \mathbf{x}) \qquad (13)$$

where $h_j^+(\alpha_j^T \mathbf{x})$ and $h_j^-(\alpha_j^T \mathbf{x})$ are the histograms of the weighted positive and negative samples. The flow chart of SpJSBoost is summarized in Table (6).

The SpJSBoost shows better generalization ability when training data contains undistinguishable samples as shown in the following toy problem.

4.2.1 A toy problem

We demonstrate the effectiveness of SpJSBoost on a nonlinear separation problem with noisy data and irrelevant features. As shown in Figure 1, the positive samples lie within a circle while the negative samples are distributed outside the circle. We add some noisy to the training data by randomly switching the labels of 5% of the samples. So the class boundary is not clear in our problem. When training the classifier, besides the positive samples x_1 and the negative samples x_2, we also feed three irrelevant features x_3, x_4, x_5 into the algorithm. Then we compare our SpJSBoost with JSBoost on a test set without noise. We use 600 samples for training and 6000 samples for testing.

The results are shown in Figure 2. As we can see, the testing error of SpJSBoost on such test set without adding noise is even lower than its training error, this indicates that the added noise in the training stage does not significantly

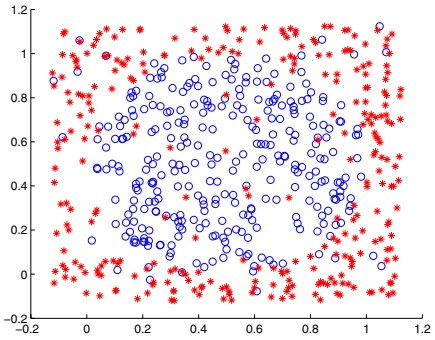

Figure 1. The synthetic data to test the generalization ability of SpJSBoost

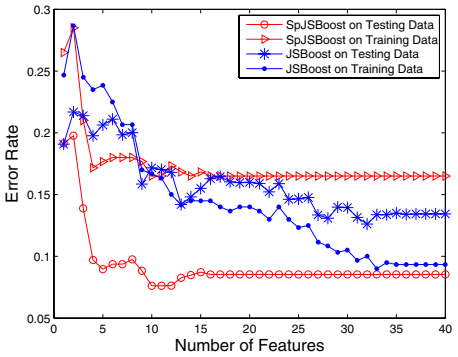

Figure 2. The training and testing error rate vs. the number of features for SpJSBoost and JSBoost.

affect the discrimination power of SpJSBoost on those originally distinguishable samples. Also, the testing error of SpJSBoost is much smaller than the original JSBoost. SpJSBoost has much better generalization on the test set. To evaluate the contributions of the irrelevant features $x_{3,4,5}$ to the learned classifier, we calculate the following ratio:

$$r = \frac{\sum_j w_j \sum_{m=3}^{5} |a_m^j|}{\sum_j w_j \sum_{m=1}^{5} |a_m^j|} \tag{14}$$

where w_j is the weight of the j-th weak learner and a_m^j is the projection coefficient of the feature x_m for the j-th weak learner. For SpJSBoost, the ratio is 10.4%, and for JSBoost, the ratio is 35.1%. This indicates that, JSBoost prefers to introduce those irrelevant features to interpret those inseparable samples, while by L_1 regularization, our SpJSBoost does not tend to select those irrelevant features to reduce the training error. Overfitting is effectively reduced.

5. Applications

In this section, to further demonstrate the effectiveness of L_1 regularized projection pursuit, we test our SpPPR on face caricature generation to demonstrate its advantages on noisy data. We also apply SpJSBoost to gender classification to show its feasibility on data set with undistinguishable samples.

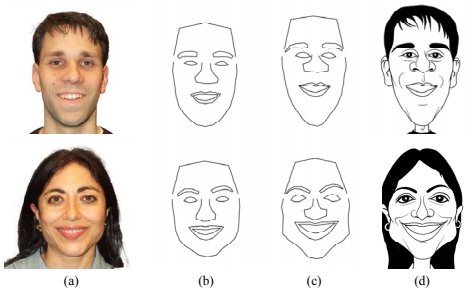

Figure 3. Examples of training data. (a) The original pictures. (b) The original face shapes. The exaggerated face shapes (c) are labeled from caricatures (d) drawn by the artist.

5.1. SpPPR for example-based face caricature generation

Generating a caricature by computer is a challenging and interesting problem and has attracted many attentions [1][14][2][6]. Recently some works have been done to learn how to exaggerate a face based on training examples [27] [16].

In this experiment, we extract a set of semantic face features from the labeled face shape, such as the width of the eyes, and apply our SpPPR to learn the mapping function from the unexaggerated features to an exaggerated one. Actually, this problem is very challenging due to high-nonlinearity, small training set and the data noise caused by the randomness of the artist's drawing process. Notice that the linear combinations of the face features are meaningful (e.g. the difference between two eyes' widths), so projection pursuit is helpful for this problem.

Our data set contains 350 faces, some examples are shown in Figure 3. We extract 35 face features from the original face shape and the exaggerated one respectively for each face, some of the features are listed in the first column of Table 7 (Please refer to [27] for all features' definitions.). Based on this data set, we test our SpPPR algorithm and compare it with three other algorithms: basic additive model(AM), projection pursuit regression(PPR) and support vector regression(SVR). The inputs are the values of 35 unexaggerated features and the output is one feature's value after exaggeration.

To test the algorithms, we do ten-fold cross-validation and calculate the average mean square error for each feature's prediction. In total 35 features, there are 32, 34 and 32 features that SpPPR predicts better than AM, PPR and SVM respectively. Table 7 shows the results of some key face features. As we can see, SpPPR outperforms other algorithms and has better generalization ability. Table 8 shows the nonzero entries in the learned projection vector for SpPPR and PPR. It is clear that the model learned by SpPPR is more concise. For this exaggeration problem, SpPPR has the ability to pursue meaningful linear combinations of face features related to exaggeration, while it will

Face feature	SpPPR	AM	PPR	SVR
Height of left eye	0.262	0.285	0.291	0.311
Width of mouth	0.327	0.340	0.370	0.363
Height of nose	0.426	0.460	0.448	0.482
Distance between brows	0.722	0.797	0.790	0.807
Width of chin	0.553	0.610	0.625	0.616

Table 7. The average mean square errors of ten-fold cross-validation test on some selected key face features for different algorithms.

Face Feature	SpPPR	PPR
Height of left eye	4	35
Width of mouth	5	35
Height of nose	12	35
Distance between brows	4	35
Width of chin	7	35

Table 8. The number of nonzero entries in the projection vectors learned by SpPPR and PPR for selected key face features.

not overfit the data like PPR because of L_1 regularization. Also the sparse model learned by SpPPR is helpful to interpret how one feature's exaggeration is affected by other features.

5.2. SpJSBoost for gender classification

Gender classification is useful for applications such as human identification, smart human computer interface [19] [11] [25]. This problem is difficult because sometimes it is hard to tell the gender from a cropped face image even for human. As shown in figure 4, people will confuse the gender of those faces with red mark. In other words, the data set contains inseparable samples. We test our SpJSBoost on this difficult problem to show its advantages on undistinguishable data.

Our data set contains 1002 females and 1005 males. The images are collected from AR [18], FERET[22] databases, etc. In our experiment, we align the faces by two eyes and use 1400 images (700 females and 700 males) for training and 607 images for testing. Some training images are shown in Figure 4. We compare our SpJSBoost with JSBoost, Adaboost, and SVM on two kinds of features: gabor features (5 scales, each scale has 8 orientations) extracted from a 40×40 face patch and the gray appearance of a 21×21 face patch. Similar gray appearance feature has been used by SVM gender classifier [19].

Table 9 shows the error rates on the testing set of different algorithms. On both kinds of features, the performance of SpJSBoost is better than other algorithms. SpJSBoost shows good generalization ability on such difficult data. From Figure 5 and Figure 6 we can see, during the training, our algorithm converges faster than JSBoost that shows the effectiveness of our "informative feature first" optimization. Also, the testing error of SpJSBoost is considerably lower than JSBoost, this indicates that the sparseness constraint makes SpJSBoost generalize better.

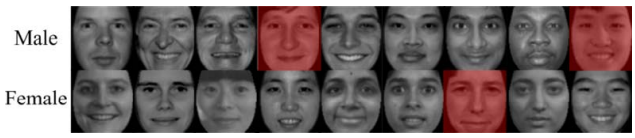

Figure 4. Training images for gender classification.

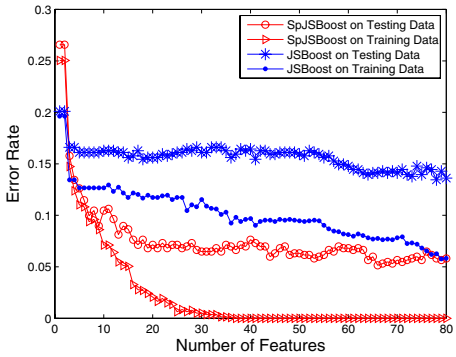

Figure 5. The training and testing error rate vs. the number of features for SpJSBoost and JSBoost using gabor feature.

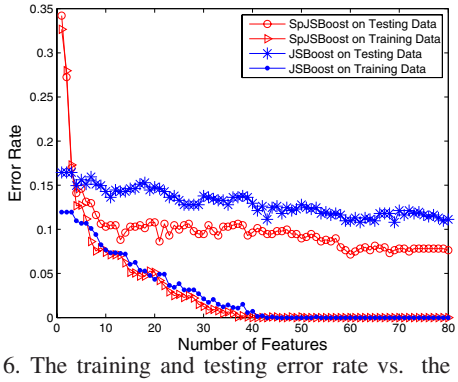

Figure 6. The training and testing error rate vs. the number of features for SpJSBoost and JSBoost using gray appearance of 21×21 face patch.

Classifier	Gabor	Gray appearance
SpJSBoost	5.1%	7.6%
JSBoost	13.6%	11.1%
AdaBoost	7.0%	10.1%
SVM	8.6%	19.3%

Table 9. The testing error rates of different algorithms for gender classification. The second and the third columns are the results of using gabor feature and gray appearance respectively.

6. Discussion

In this section, we will discuss the properties of three kinds of methods: additive model with L_1 regularized projection pursuit (L1PpAM), additive model with projection pursuit (PpAM) and support vector machine (SVM).

(a) L1PpAM vs. PpAM

L1PpAM has the advantage of PpAM to pursue the optimal linear combinations of input features to interpret the response better, while by L_1 regularized projection pursuit, L1PpAM avoids the overfitting problem of PpAM. Comparing with PpAM, the model learned by L1PpAM is sparse, easy to interpret and has better generalization ability especially when there are noisy data and many irrelevant features.

(b) L1PpAM vs. SVM

Support Vectors Machines have been proved to work well in extremely high-dimensional input spaces. But as mentioned in [20], SVMs perform poorly in the presence of many irrelevant features, because the diameter of the data (e.g. maximum distance between any two points measured in the L_2-norm) grows with the number of irrelevant features. While by using L_1 regularized projection pursuit, L1PpAM has the ability to pursue sparse optimal features, thus L1PpAM performs better than SVM especially when there are many irrelevant features fed into the algorithm.

7. Conclusion

In this paper, we have presented a L_1 regularized projection pursuit algorithm for additive model learning. An efficient "informative feature first" optimization algorithm is developed to pursue the optimal projection with L_1 regularization. Based on this, two new algorithms are developed for regression and classification respectively: SpPPR and SpJSBoost. The toy problems and the applications of caricature generation and gender classification demonstrate the feasibility of the algorithms especially for noisy data with many irrelevant features.

References

[1] S. E. Brennan. Caricature generator: The dynamic exaggeration of faces by computer. *Leonardo*, 18(3):170–178, 1985. 6

[2] H. Chen, N. N. Zheng, L. Liang, Y. Q. Xu, and H. Y. Shum. A personalized image-based cartoon system. In *ACM multimedia*, pages 171–178, 2002. 6

[3] D. Donoho. For most large underdetermined systems of linear equations the minimal L_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006. 3

[4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regession: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000. 1, 2

[5] J. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981. 1, 2, 4

[6] B. Gooch, E. Reinhard, and A. Gooch. Human facial illustrations: Creation and psychophysical evaluation. *ACM TOG*, 23(1):27–44, 2004. 6

[7] T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–318, 1986. 1, 2, 4

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning:Data Mining,Inference and Prediction*. Springer-Verlag, 2001. 2, 3

[9] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. 2

[10] X. Huang, S. Z. Li, and Y. Wang. Jensen-shannon boosting learning for object recognition. In *CVPR*, pages 144–149, 2005. 5

[11] A. Jain and J. Huang. Integrating independent components and linear discriminant analysis for gender classification. In *FGR*, pages 159–163, 2004. 7

[12] B. Kegl and L. Wang. Boosting on manifolds: adaptive regularization of base classifiers. In *NIPS*, 2004. 1

[13] Y. Kim and J. Kim. Gradient lasso for feature selection. In *ICML*, volume 69, July 2004. 3, 4

[14] H. Koshimizu and etal. On kansei facial processing for computerized facial caricaturing system picasso. *IEEE SMC*, 6:294–299, 1999. 6

[15] J. B. L. Mason, P. Bartlett and M. Frean. Boosting algorithms as gradient descent. In *NIPS*, pages 512–518, 1999. 1

[16] L. Liang, H. Chen, Y. Q. Xu, and H. Y. Shum. Example-based caricature generation with exaggeration. In *Pacific Conference on Computer Graphics and Applications*, pages 386–393, 2002. 6

[17] C. Liu and H. Y. Shum. Kullback-leibler boosting. In *CVPR*, pages 587–594, May 2003. 1, 2, 3, 4, 5

[18] A. Martinez and R. Benavente. The AR face database. Technical Report 24, The Computer Vision Center (CVC) at the U.A.B., 1998. 7

[19] B. Moghaddam and M. H. Yang. Learning gender with support faces. *PAMI*, 25(5):707–711, 2002. 7

[20] A. Y. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *ICML*, 2004. 3, 8

[21] S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast, incremetal feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003. 3, 4

[22] P. Phillips, H. Wechsler, J. Huang, and P. Pauss. The feret database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998. 7

[23] P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. SpAM: Sparse additive models. In *NIPS*, 2007. 1

[24] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B, Methodological*, 58(1):267–288, 1996. 1

[25] R. Verschae, J. R. del Solar, and M. Correa. Gender classification of faces using adaboost. In *Iberoamerican Congress on Pattern Recognition*, pages 68–78, 2006. 7

[26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001. 5

[27] H. Zhong, Y. Q. Xu, and H. Y. Shum. Example-based face caricature generation. Technical report, Microsoft Research Asia, 2005. 6