ManifoldBoost: Stagewise Function Approximation for Fully-, Semi- and Un-supervised Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce a boosting framework to solve a classification problem with added manifold and ambient regularization costs. It allows for a natural extension of boosting into both semisupervised problems and unsupervised problems. The augmented cost is minimized in a greedy, stagewise functional minimization procedure as in GradientBoost. Our method provides insights into generalization issues in GradientBoost as applied to trees; these phenomena are relevant also to manifold learning. We describe a quite general framework and then discuss a specific case based on L_2 TreeBoost. This framework naturally accommodates supervised learning, manifold learning, partially supervised learning and unsupervised clustering as particular cases. Multiclass learning tasks fit naturally into the framework as well. Unlike other manifold learning approaches, the family of algorithms derived has linear complexity in the number of datapoints. The performance of our method is at the state of the art on some standard problems, and exceeds the state of the art on others.

1 Introduction

Manifold learning algorithms exploit geometric (or correlation) properties of datasets in high-dimensional spaces. Algorithms estimate structure either explicitly ([1, 2], etc.) or implicitly, via a penalty term that imposes smoothness conditions on functions restricted to the manifold (e. g. [3]).

Gradient boosting is a powerful framework for machine learning, introduced in [4]. In essence, one sees function approximation as a variational problem, then uses a form of coordinate ascent on this problem (section 2). In this paper, we show that this method can be improved by using both a manifold regularisation term and a further, carefully chosen regularization term that both improves generalization performance and is naturally interpreted in terms of the manifold structure of the training data (section 2.1). This yields a clean algorithm which can be applied to many forms of functional approximation (section 3). We give a specific example using binomial likelihood classification loss and regression trees as base learners (section 4). In the extreme case, when there is no supervision, the generalised method gracefully degrades into a clustering method (section 5). Finally, we show that our framework also easily extends to multi-class problems by choosing suitable loss functions (section 6).

Computational complexity is linear in the number of data examples (labeled l + unlabeled u). Other semisupervised approaches solve Quadratic Programming problems of size $(l + u)$ [5] or (l) [3], plus a linear problem of size $(l + u)$, which is essentially cubic in the number of datapoints. Our algorithm is at, or exceeds, state of the art performance on standard datasets (section 7).

2 Semi-supervised Boosting

Assume we have a probability distribution $P_{x,y}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the pattern space and $\mathcal{Y}$ the label space. We further assume the support of the marginal P_x lies on a low dimensional manifold in $\mathcal{X}$. We would like to find the function minimizer $F^* = \arg\min_F V[F]$, of the cost functional

$$V[F] = \underbrace{\int \psi(y, F(x)) dP_{x,y}}_{\text{Expected Loss}} + \underbrace{\gamma_{\mathcal{M}} \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} F(x)||^2 dP_x}_{\text{Manifold regularization}} + \underbrace{\gamma_A \int_T ||\nabla F(x)||^2 d\tilde{P}_x}_{\text{Ambient regularization}} \quad (1)$$

where $\psi(y, F(x))$ is the loss function between label y and prediction $F(x)$, $\nabla_{\mathcal{M}}$ is the gradient on the manifold, and γ is the manifold regularization weight. There are many possible choices for $\psi[y, F]$. Functions of $(y - F)$ are typically used for regression, such as $|y - F|$ ($L1$) and $(y - F)^2$ ($L2$). Functions of the *margin* yF are used for classification, and include $\exp(-yF)$ and the binomial log likelihood $\log(1 + \exp(-2yF))$.

In order to control the complexity of the classifier, regularization typically involves two terms. One encourages smoothness of the solution in regions of high probability density (i.e. manifold regularization). We use the same term as [3]. The other term controls the complexity of the solution *outside* the manifold, in *ambient* space.

2.1 Generalization in Boosting and Ambient Regularization

Schapire [10] explained Adaboost generalization performance in terms of the *margins* yF of the training examples. In [11] Breiman proposed the ARC-GV algorithm to maximize the minimum margin of any training example. Other margin maximizing boosting algorithms were proposed in [9] or [12] (and others). Surprisingly, Breiman showed no performance gain even when the the algorithm produced uniformly higher margins than Adaboost. [13] explains this apparent contradiction by showing Breiman's formulation indirectly increased the complexity of the weak learners.

We present a toy problem that suggests an alternative explanation. Consider figure 1: in discontinuous weak learners such as trees the *margin* yF may be high even when the distance to the decision boundary is negligible, leading to poor generalization (second figure). Controlling the complexity of tree learners using their depth does not necessarily fix the problem. The margin is no guide to the spatial separation of data points because F is discontinuous — this means that discontinuous changes in F may lie close to training data points with no explicit penalty. This phenomenon can result in poor generalisation, as figure 1 shows.

A natural solution is to penalize variations in F that lie close to, but not on, training data. There is no virtue in penalising variations in F that lie far from all data points, because we do not expect to see data there. One can see this as penalizing variability in the solution in a "tubular" neighborhood T of the data points. Equivalently, one is penalizing variability on a manifold defined by a probability density $\tilde{P}_x$ (figure 1, center right). The latter is a smoothed version of of the original P_x, obtained by convolving it with a Gaussian of scale σ. After regularization, the classification function learned is smoother, leading to better generalization (figure 1, right). Notice that, in practice, this regularization process involves generating new, unlabelled datapoints from the original dataset.

If one takes the position that the datapoints lie on a manifold, then the choice of scale of this tubular neighbourhood is a commitment to the "thickness" of this manifold. More specifically, one needs to distinguish between the case where the data indicates there are two neighbourhoods of the manifold lying close to one another — when F can have quite different values on each neighbourhood — and the case where there is only one neighbourhood— when F should not vary much.

This regularization term is very like a clustering term, in that it prefers functions that have slow variation within connected components of the tubular neighbourhood. Crucially, an F that takes different values on different connected components is not penalized; as a result, our algorithm graciously degrades into a clustering algorithm if there is no supervisory data. The choice of scale of this tubular neighborhood is a parameter of our algorithm; we conjecture that it is a fundamental parameter in manifold learning.

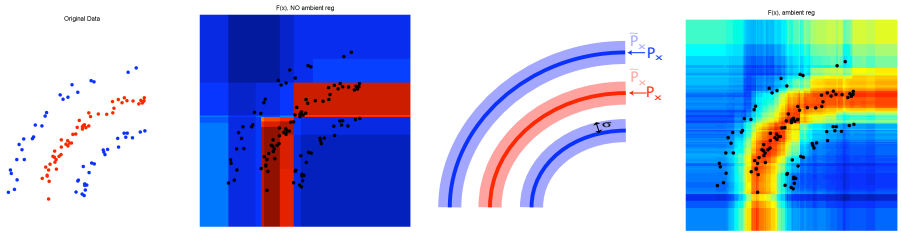

Figure 1: **Margins, Generalization and ambient regularization** (this figure is best viewed in color). Consider the problem on the **left**. A boosted tree classifier with no ambient regularization can fit the training data perfectly, and as trees are discontinuous functions the margins yF are high even when the datapoints are close to the decision boundary (**second image**). The algorithm has 5% error rate on the test set, because some decision boundaries are unnecessarily close to the training data. The **third** depicts the gist of the ambient regularization term. Consider the three manifold "sheets", with distribution P_x. Manifold regularization penalizes variance on the dark lines. Ambient regularization penalizes variance in the lighter "tubes", with distribution $\tilde{P}_x$ obtained by convolving P_x with a Gaussian of scale σ. This effectively "pushes" the decision boundaries away from the manifold, except for regions where the "tubes" overlap. The figure on the **right** shows the function F for the regularized classifier. The test error rate has been reduced to 0%.

3 MANIFOLDBOOST framework

3.1 Stagewise Functional minimization (P_x known)

As in [4] we will find a *additive* solution of the form $F_M(x) = \sum_{m=0}^{M} f_m(x)$. Instead of minimizing (1) solving for all f_m simultaneously, we will proceed in a greedy fashion, for each m fixing F_{m-1} and minimizing $V[F_{m-1} + f_m]$ with respect to f_m. In an alternative view of the procedure in [4], we approximate the descent direction by computing the first variation of V, ignoring higher order terms,

$$V[F_{m-1} + \epsilon f] = V[F_{m-1}] + \epsilon \delta V[F_{m-1}, f] + O(\epsilon^2) \tag{2}$$

where $\delta V[F_{m-1}, f] = \frac{d}{d\epsilon} V[F_{m-1} + \epsilon f]|_{\epsilon=0}$.

In order to make the maximization of δV with respect to $f(x)$ in equation 2 tractable, we introduce G_V, the "gradient" of the cost V, so that $\delta V[F_{m-1}, f] = \frac{d}{d\epsilon} V(F_{m-1} + \epsilon f)|_{\epsilon=0} = \langle G_V, f \rangle$, where the inner product is the usual in L_2.Assuming sufficient regularity, and using the first Green identity, we have $\langle G_V[F]|_{F_{m-1}}, f \rangle$ is

$$\int_x f \int_y \frac{\partial}{\partial u} \psi(y,u) \Big|_{F_{m-1}(x)} dP_{x,y} + 2\gamma_{\mathcal{M}} f \nabla^2_{\mathcal{M}} F_{m-1}(x) dP_x + 2\gamma_A f \nabla^2 F_{m-1}(x) d\tilde{P}_x \tag{3}$$

In function space, the optimal descent direction maximizes $-\langle G_V|_{F_{m-1}}, f \rangle$. The final $f_m = \alpha f$ is obtained using line search, minimizing the true cost $V[F_{m-1} + \alpha f]$ with respect to α.

3.2 Finite data case (P_x unknown)

When the continuous distribution over (x, y) is not available, it has to be estimated using a finite data sample $\{x_i, y_i\}_{i=1}^{N}$, and integrals become sums over datapoints. In this case it is most practical that $\{f_m(x)\}$ belongs to a family of parametrized functions (e. g. Radial Basis functions, decision trees, etc). Using a parametric function family for f ensures that values of the solution outside the training sample can be defined.

The Laplacian operator in equation 3 also has to be discretized. A usual approximation for the regularization term in equation 1 is the graph Laplacian $\mathcal{L}$ [3]. It consists of a weighted difference between the function at a point and its K nearest neighboring points; it is a generalization of the square lattice Laplacian discretization commonly used in numerical analysis and image processing.

To approximate the ambient regularization term we need out-of-sample points sampled from $\tilde{P}_x$. We obtain those by sampling a number of times t from a Gaussian of variance σ^2 centered at each datapoint.

The cost function becomes then,

$$V[F] = \frac{1}{l}\sum_{i=1}^{l}\psi(y_i, F(x_i)) + \frac{\gamma_{\mathcal{M}}}{(l+u)^2}\sum_{i,j}F(x_i)\mathcal{L}_{\mathcal{M}}i, jF(x_j) + \frac{\gamma_A}{t^2(l+u)^2}\sum_{p,q}F(x_p)\mathcal{L}_A p, qF(x_q) \tag{4}$$

where p, q are indexes on the newly sampled points. Proceeding in a similar way to the continuous case, f_m has to maximize $-\langle G_V, f\rangle$. Now the inner product is $\langle a, b\rangle = \frac{1}{N}\sum_{i=1}^{N} a(x_i)\cdot b(x_j)$ and this gives that $\langle G_V[F]|_{F_{m-1}}, f\rangle$ as

$$\frac{1}{l}\sum_{i}\frac{\partial}{\partial u}\psi(y,u)\Big|_{F_{m-1}(x_i)} f(x_i) + \frac{2\gamma_{\mathcal{M}}}{(l+u)^2}\sum_{i,j}f(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F_{m-1}(x_j) + \frac{2\gamma_A}{t^2(l+u)^2}\sum_{p,q}f(x_p)\mathcal{L}^{A}_{p,q}F_{m-1}(x_q) \tag{5}$$

But the product $-\langle G_V, f\rangle$ can grow without bounds unless $||f||$ is constrained. This free parameter (the norm of f) can be chosen so that the maximization is equivalent to minimizing $||G_V - f||^2 = ||G_V||^2 - 2\langle G_V, f\rangle + ||f||^2$. Thus, any squared loss regression algorithm can be used to find the optimal parameters. In our variational formulation, Friedman's *choice* to make f_m parallel to the gradient G_V and pose the problem as squared error minimization becomes natural.

Again, after the descent direction f is found, the final $f_m = \alpha f$ is obtained using line search, minimizing the true cost $V[F_{m-1} + \alpha f]$.

4 Example: Tree MANIFOLDBOOST algorithm

We consider the binary case ($y \in \{-1, 1\}$, $y = 0$ for unlabeled data), and use the negative binomial log likelihood as the loss function [4]: $\psi(y, F) = \log(1 + \exp(-2yF))$, where the classifier represents $F(x) = \frac{1}{2}[\log(p(y=1|x)) - \log(p(y=-1|x))]$ At round m, the inner product with the "gradient" – which is $\langle G_V[F]|_{F_{m-1}}, f\rangle$ — becomes

$$\frac{1}{l}\sum_{i}\frac{2y_i}{1+\exp(2y_iF_{m-1}(x_i))}f(x_i) + \frac{2\gamma_{\mathcal{M}}}{(l+u)^2}\sum_{i,j}f(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F_{m-1}(x_j) + \frac{2\gamma_A}{t^2(l+u)^2}\sum_{p,q}f(x_p)\mathcal{L}^{A}_{p,q}F_{m-1}(x_q) \tag{6}$$

As in L_2 TreeBoost [4], we consider regression trees as base learners. A tree has the form $f_m(x) = \sum_{s=1}^{S}\eta_{m,s}[x \in R_s]$, where $[\cdot]$ is one if the expression inside is true, and zero otherwise. Thus, minimization of $||G_V - f||^2$ involves the search for parameters $\{R_s, \eta_s\}$. Once the tree has been found, instead of finding the optimal α using line search, we fix the regions R_s and minimize $V(F_{m-1}(x) + \sum_{s=1}^{S}\eta_{m,s}[x \in R_s])$ with respect to $\{\eta_s\}$. As there is no closed from solution, we use the gradient-descent type BFGS minimization method. In order to do this, we need to compute the gradient of V with respect to η, $\frac{\partial V}{\partial \eta_s}$. On each round, the BFGS algorithm is ran only for a small number of iterations to prevent overfitting. Alternatively, the second derivatives with respect to η may also be computed, and used in Newton-Raphson methods.

Algorithm 1 Tree ManifoldBoost Algorithm

1: $F_0(x) = 1/2[\log(1+\overline{y}) - \log(1-\overline{y})]$
2: **for** $m = 1$ to M **do**
3: Compute G_V as in (6)
4: Obtain regression tree $\{R_{s,m}\}$ by minimizing $\sum_i (G_V(x_i) - \sum_s \eta_{s,m}[x_i \in R_{s,m}])^2$
5: Find $\{\eta_{s,m}\}$ using BFGS and $\frac{\partial V}{\partial \eta_s}$, and fixing $\{R_{s,m}\}$
6: $F_m(x) = F_{m-1}(x) + \sum_s \eta_{s,m}[x \in R_{s,m}]$
7: **end for**

The algorithm converges when M rounds have been ran, or the relative change in the cost function in a round is below a threshold. Probability estimates for each x can then be estimated by inverting: $p(y=1|x) = 1/(1+\exp(-2F_M(x)))$. This in turn can be used for classification:

$$\tilde{y_i} = \begin{cases} 1 & c_{1,-1}p(y=1|x) > c_{1,-1}p(y=-1|x) \\ -1 & \text{otherwise} \end{cases} \tag{7}$$

where cost $c_{a,b}$ is the penalty for choosing label a when b is the correct label.

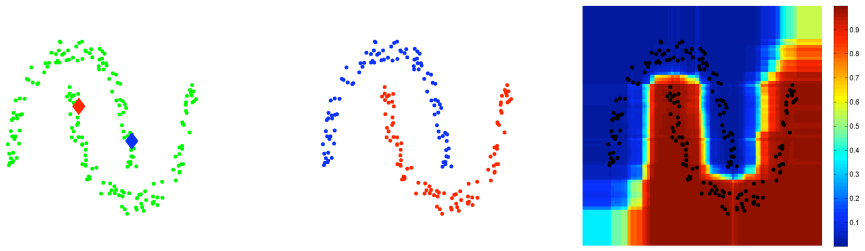

Figure 2: **Semi-supervised learning** using Tree MANIFOLDBOOST. **Left** figure shows a toy example introduced in [3] (two moons dataset): the unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The output of the algorithm is depicted on the **center** (datapoints) and **right** (whole space) figures. This figure is best viewed in color. As in figure 1 the classifier imposes spacing between the manifold and the decision boundaries.

Figure 2 shows a toy example taken from [3] (two moons dataset) for semisupervised classification. The unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The algorithm also can provide likelihood estimates, as seen in the right figure.

After sorting on the input variables, the procedure is linear in $N = l + u$, in the Laplacian neighborhood K, the dimensionality of x and the number of rounds. Influence trimming can also be used to get tenfold speedups [4], although the algorithm is still linear in the number of datapoints.

5 Unsupervised Boosting

A regularization term which uses the geometric structure of the data for semi-supervised learning should still be useful when no labels are available. Unlike other boosting algorithms, this framework can naturally be extended for unsupervised learning. As there are no labeled data, the first term in equation 4 becomes zero and the problem is,

$$F^* = \arg\min \sum_{i,j} F(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F(x_j) + \frac{\gamma_A}{t^2\gamma_{\mathcal{M}}} \sum_{p,q} F(x_p)\mathcal{L}^{A}_{p,q}F(x_q) \quad \text{subject to } \sum_i F = 0, \sum_i F_i^2 = N \tag{8}$$

Without the constraints the problem is ill-posed and trivial solutions like $F \equiv 0$ exist. In fact, the constrained problem we minimize is equivalent to a spectral clustering problem [3]. But, instead of solving it as a generalized eigenvalue problem, we again approach it in a greedy fashion. We use the Augmented Lagrangian method [14], which adds a penalty for constraint violation to the unconstrained problem in each round. As before, BFGS is applied in each per round.

The algorithm converges to a local minimum of the constrained problem. This formulation, unlike [1], naturally takes care of out-of-sample evaluation. Unlike [3], this algorithm is linear in the number of unlabeled data points.

Algorithm 2 Unsupervised Tree ManifoldBoost Algorithm

1: Initialize $F_0(x)$ randomly, with zero mean and low variance.
2: **for** $m = 1$ to M **do**
3: Compute G_V of the penalized, unconstrained problem.
4: Obtain regression tree $\{R_{s,m}\}$ by minimizing $\sum_i (G_V(x_i) - \sum_s \eta_{s,m}[x_i \in R_{s,m}])^2$
5: Find $\{\eta_{s,m}\}$ using BFGS and fixing $\{R_{s,m}\}$
6: $F_m(x) = F_{m-1}(x) + \sum_s \eta_{s,m}[x \in R_{s,m}]$
7: Update Lagrange multipliers using the constrain violations.
8: **end for**

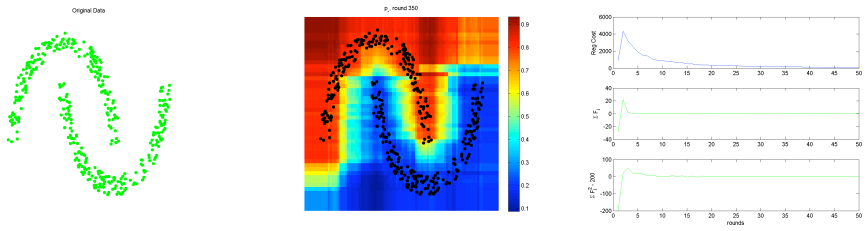

Figure 3: **Unsupervised learning**. The framework can be extended for unsupervised learning. The same problem as in figure 2 is presented without labels (**left** image). The result of the algorithm is shown in the **center** image. The plots on the **right** show the constraint violations go to zero as learning progresses. This figure is best viewed in color.

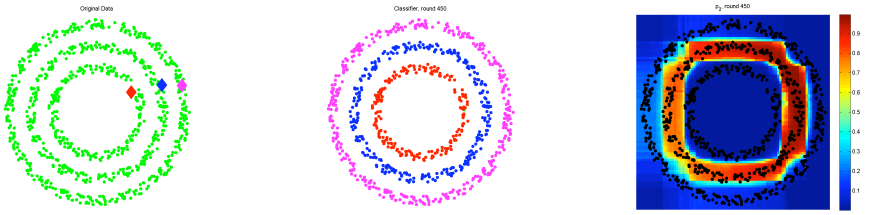

Figure 4: The framework also handles **multiclass learning**. **Left** figure shows another toy example (three rings): the unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The output of the algorithm is depicted on the **center** figure. The blue likelihood function $p^2(x)$ is shown on the **right**. This figure is best viewed in color.

6 Multiclass case

Algorithm 1 can be extended to K-class problems by introducing in equation 1 a multinomial cost

$$\psi(\{y^k, F^k(x)\}_{k=1}^K) = -\sum_{k=1}^K y^k \log p^k(x) \tag{9}$$

where $p^k(x)$ represents the belief example x belongs to class k, and y^k is a binary variable which is one if example x belongs to class k. As in [4] we use the symmetric multiple logistic transform

$$p^k(x) = \exp F^k(x) \cdot \left(\sum_{c=1}^K \exp F^c(x) \right)^{-1} \tag{10}$$

Smoothness of F^k is enforced by defining the cost as,

$$\begin{aligned} V(\{F^k\}) &= \frac{1}{l} \sum_i \psi(\{y_i^k, F^k(x_i)\}_{k=1}^K) + \frac{1}{K \cdot (l+u)^2} \sum_c \sum_{i,j} \gamma_{\mathcal{M}}^c F^c(x_i) \mathcal{L}_{i,j}^{\mathcal{M}} F^c(x_j) \\ &+ \frac{1}{Kt^2(l+u)^2} \sum_c \sum_{p,q} \gamma_A^c F^c(x_i) \mathcal{L}_{p,q}^A F^c(x_j) \end{aligned} \tag{11}$$

The inner product of f^k with gradient of V becomes, for class k,

$$\begin{aligned} \langle G_V^k, f^k \rangle &= \frac{1}{l} \sum_i (-y_i^k + p_{m-1}^k(x_i)) f^k(x_i) + \frac{2\gamma_k^{\mathcal{M}}}{K \cdot (l+u)^2} \sum_{i,j} f^k(x_i) \mathcal{L}_{i,j}^{\mathcal{M}} F_{m-1}^k(x_j) \\ &+ \frac{2\gamma_k^A}{Kt^2(l+u)^2} \sum_{p,q} f^k(x_p) \mathcal{L}_{p,q}^A F_{m-1}^k(x_q) \end{aligned} \tag{12}$$

Now one regression tree is fitted per class at each round to approximate each descent direction. As in the two class problem, the S regions $\{R_{m,s}^k\}_{s=1}^S$ defined by the terminal nodes are fixed, and the

parameters $\eta_{m,s}^k$ for regions in each tree are learned in order to minimize the total cost V. We use a couple of BFGS iterations per round to find these parameters. In order to do this, the derivatives of the cost with respect to $\eta_{m,s}^k$ have to be computed.

Once the final $\{F_M^k(x)\}$ are computed, the probability for a given example of each label can be estimated and thus the label can be classified as $\hat{k}(x_i) = \arg\min_k \sum_{j=1}^K c(k,j) p_M^j(x)$ for costs $c(k,j)$ when label k is assigned when label j is correct. The complexity of this algorithm is also linear in the number of classes, but it scales highly sub-linearly with the number of rounds M when influence trimming is used [4].

Algorithm 3 K-Tree ManifoldBoost Algorithm

Let p_0^k be the frequencies of each class k.
$F_0(x) = \log p_0^k - \frac{1}{K} \sum_{c=1}^K \log p_0^c$
for $m = 1$ to M **do**
  Compute $p_m^k(x)$ as in eq. 10 for all k.
  for $k = 1$ to K **do**
    Compute G_V^k as in (12)
    Obtain regression tree $\{R_{s,m}^k\}$ by minimizing $\sum_i (G_V^k(x_i) - \sum_s \eta_{s,m}^k [x_i \in R_{s,m}])^2$
  end for
  Find $\{\eta_{s,m}^k\}$ using BFGS and $\partial V \partial \eta_{m,s}^k$, and fixing $\{R_{s,m}^k\}$ for all k.
  $F_m^k(x) = F_{m-1}^k(x) + \sum_s \eta_{s,m}^k [x \in R_{s,m}^k]$ for all k.
end for

7 Experiments

Kegl et. al. [15] introduce REGBOOST, an extension to ADABOOST which incorporates a weight decay that depends on a Laplacian regularizer. We compare our results with [15] on standard UCI benchmark datasets. Whenever possible we tried to use the same configuration as [15][1]. We set number of nearest neighbors $K = 8$ and used binary weights to compute the graph Laplacian. We used regression trees of fixed depth 3 as learners. The datasets were normalized to zero mean and unit variance so that the ambient scale $\sigma = 0.1$ was coherent in all dimensions. The learning rate was set to $\nu = 0.1$ after [4]. The number of Gaussian samples per datapoint (for ambient regularization) was set to 3-5. Only γ was explored for different values. We used 5-fold cross validation for determining parameters and 10-fold cross validation for error estimation. Table 1 compares our performance with that of ADABOOST, REGBOOST, and [16] as reported in [15].

In the **fully supervised** problems, there is a significant difference in performance for the Sonar dataset, an improvement in the Ionosphere dataset, and a very slight decrease in performance in the Pima Indians dataset with respect to REGBOOST, well within a standard deviation. It should be noted that the variance in the performance of the algorithm is consistently smaller for our algorithm. As in REGBOOST, the difference between train and test errors is smaller than for ADABOOST, suggesting less overfitting. [15] also tests the algorithm under **semi-supervision**, using 100 labeled and 251 unlabeled examples. We ran our algorithm under the same conditions, using the same parameters obtained in the previous experiment. In this case our algorithm clearly outperforms [15] and [16], as our mean performance after 10 runs is more than a standard deviation above theirs. No variance of results is reported in [15].

8 Conclusion

We have presented a new "boosting" framework for regularized learning. It is flexible enough to handle the whole range of supervision, from fully supervised classification to unsupervised clustering. The framework is general and accepts any learner that can maximize the inner product 3, which includes min square error regression algorithms. It can be naturally extended to multiclass

[1]The breast cancer dataset was not used because [15] does not explain what metric they use for categorical data in the Lalacian, making any comparison meaningless.

Algorithm	Ionosphere Train / Test	Pima Indians Train / Test	Sonar Train / Test	Ionosphere (semisup)
ADABOOST [15]	0% / 9.2% (7.1)	10.9% / 25.3% (5.3)	0% / 32.5% (19.8)	-
REGBOOST [15]	0% / 7.7% (6.0)	16.0% / **23.3**% (6.8)	0% / 29.8% (18.8)	12%
MANIFOLDBOOST	0% (0) / **6.7**% (3.2)	14.9% (0.8) / 23.4% (3.7)	0% (0) / **22.0**% (7.3)	**8.9**% (1.6)
[3] cited in [15]	- / -	- / -	- / -	18%

Table 1: **Performance results on standard UCI datasets**. The algorithm has a superior performance compared to other regularized boosters in fully supervised settings, with less variance (in parenthesis). On a semi-supervised problem it outperforms both [15] and [16].

problems. Crucially for semi-supervised problems, it is linear in complexity with respect to the number of datapoints which allows the use of more unlabeled examples (which are cheaper usually than labeled ones). We also introduced an ambient regularization term that provides some insights into generalization issues in Boosting.

We are working on understanding important aspects of the algorithm. In particular, generalization, error bounds, and convergence. An important problem which remains unsolved in semi-supervised problems like this is determining an optimal value for the γs, the weights of the regularization terms.

References

[1] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[2] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding, 2000.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

[4] J. Friedman. Greedy function approximation: a gradient boosting machine, 1999.

[5] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[6] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *ICML*, pages 824–831, 2005.

[7] P. Bühlmann and B. Yu. Sparse boosting. *J. Mach. Learn. Res.*, 7:1001–1024, 2006.

[8] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. pages 118–183, 2003.

[9] L. Mason, P. L. Bartlett, and M. Golea. Generalization error of combined classifiers. *J. Comput. Syst. Sci.*, 65(2):415–438, 2002.

[10] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th ICML*, pages 322–330, 1997.

[11] L. Breiman. Prediction games and arcing algorithms. *Neural Comput.*, 11(7):1493–1517, 1999.

[12] C Rudin, R. E. Schapire, and I. Daubechies. Boosting based on a smooth margin. In *COLT*, 2004.

[13] L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *ICML*, pages 753–760, 2006.

[14] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1996.

[15] B. Kégl and L. Wang. Boosting on manifolds: Adaptive regularization of base classifiers. In *NIPS 17*, pages 665–672. 2005.

[16] I. Matveeva M. Belkin and P. Niyogi. Regression and regularization on large graphs. In *COLT*, 2004.