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

Nicolas Loeff LOEFF@UIUC.EDU
Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801

David Forsyth DAF@UIUC.EDU
Deepak Ramachandran DRAMACHA@UIUC.EDU
Department of Computer Science, University of Illinois, Urbana, IL 61801

Abstract

We describe a manifold learning framework that naturally accommodates supervised learning, partially supervised learning and unsupervised clustering as particular cases. Our method chooses a function by minimizing loss subject to a manifold regularization penalty. This augmented cost is minimized using a greedy, stagewise, functional minimization procedure, as in Gradientboost. Each stage of boosting is fast and efficient. We demonstrate our approach using both radial basis function approximations and trees. The performance of our method is at the state of the art on many standard semi-supervised learning benchmarks, and we produce results for large scale datasets.

1. Introduction

Manifold Learning algorithms exploit geometric (or correlation) properties of datasets in high-dimensional spaces. The literature is too large to review in detail here (163 references in a recent review (Zhu, 2006)). Many different approaches have been pursued that utilize manifold structure such as constructing an explicit parametrization (e.g. (Tenenbaum et al., 2000; Roweis & Saul, 2000; Donoho & Grimes, 2003)), introducing a penalty term that imposes smoothness conditions on functions restricted to the manifold (e.g. (Sindhwani et al., 2006)), adjusting kernel smoothing bandwidths to account for manifold properties (e.g. (Bickel & Li, 2007)), and infering labels for unlabeled data using a harmonic smoother (e.g. (Zhu et al., 2003)).

There is a rough distinction in semi-supervised learning between manifold based algorithms that expect data to lie embedded in a space of lower intrisic dimensionality, and cluster-based algorithms that expect data to lie in clumps (the distinction seems to explain some differences in performance on different datasets (Chapelle et al., 2006)). There is some disagreement about the benefits of using unlabeled data, which may not always improve the asymptotic error rate of a regression estimator (Lafferty & Wasserman, 2007). On the other hand, (Niyogi, 2008) argues that manifold learning is useful insofar as the marginal of the data P_x can be linked with the conditional $P_{y|x}$ via the manifold.

Computational Complexity is a common problem for most semi-supervised approaches. Write l for the number of labeled data items and u for the number of unlabeled data items. Many algorithms scale as badly as $O((l+u)^3)$ (Zhu, 2006). Transductive support vector machines must solve a quadratic programming problem in $(l+u)$ variables (Joachims, 1999). Manifold smoothing of an SVM solves a quadratic programming problem in l variables, followed by a linear problem in $l+u$ variables; the situation is better for a linear SVM if feature vectors are sufficiently sparse (Sindhwani et al., 2006). Harmonic smoothing solves a relatively sparse linear system in l variables. This problem is relatively tractable, because the linear system involves the Laplacian of the smoothing kernel and so should be diagonally dominant (see (Dyn et al., 1986) for relevant observations in the context of radial basis functions). Each method must pay the cost of forming the Laplacian. For functional approximation schemes other than kernel smoothing, the complexity of current manifold learning methods in the number of training examples appears to be high. This is a problem – it is natural to want to use a manifold regularization term

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

with such methods as tree-structured classifiers, and with very large datasets.

Gradient Boosting poses function approximation as a variational problem, then uses a form of coordinate ascent on that problem ((Friedman, 1999); section 2). In this paper, we describe a variation on gradient boosting that can exploit a manifold regularization term, is fast and efficient for many forms of functional approximation (section 2.1), provides out of sample extensions (section 4), offers performance at the state of the art on standard datasets, and is capable of handling very large datasets (section 6). In the extreme case, when there is no supervision, the generalized method gracefully degrades into a clustering method (section 4). Finally, we show that our framework also easily extends to multi-class problems by choosing suitable loss functions (section 5).

2. Semi-Supervised Boosting

We follow convention by minimizing the sum of an expected loss and a regularization term. We must predict labels $y \in \mathcal{Y}$ for patterns $x \in \mathcal{X}$. We assume a probability distribution $P_{x,y}$ over $\mathcal{X} \times \mathcal{Y}$.

We will further assume the support of the marginal P_x lies on a domain $\mathcal{M} \subset \mathcal{X}$. Typically, this domain is of lower intrinsic dimension than $\mathcal{X}$; the term manifold is widely used to refer to such domains, though we require no manifold properties.

Write the predictor as $F(x)$, and the cost function as $\psi(y, F(x))$. We would like to find the function minimizer $F^* = \arg\min_{F \in \mathcal{H}} V[F]$, of the cost functional

$$\begin{aligned} V[F] \quad &= \quad \underbrace{\int \psi(y, F(x)) dP_{x,y}}_{\text{Expected Loss}} \\ &+ \quad \underbrace{\gamma_{\mathcal{M}} \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} F(x)||^2 dP_x}_{\text{Manifold Regularization}} \end{aligned} \tag{1}$$

restricted to some function family $\mathcal{H}$. Our regularization term is of the same form as that of (Sindhwani et al., 2006), and encourages smoothness of the solution in regions of high probability density. We control the complexity of the solution by choosing $\mathcal{H}$ and using the *shrinkage* approach of (Friedman, 1999).

This expression is very general. There are many possible choices for $\psi[y, F]$. Expressions such as $|y - F|$ and $(y - F)^2$ are typically used for regression. Expressions such as $\exp(-yF)$ and the binomial log likelihood $\log(1 + \exp(-2yF))$ penalize the *margin* yF, and are typically used for classification.

2.1. ManifoldBoost Framework

2.1.1. Stagewise Functional Minimization (P_x Known)

Following the work of Friedman (Friedman, 1999), we will find a *additive* solution of the form

$$F_M(x) = \sum_{m'=0}^{M} f_{m'}(x) \tag{2}$$

We will proceed in a greedy fashion. Assume we have a solution for $M = m$; we will then minimize $V[F_m + f_{m+1}]$ with respect to f_{m+1}. After (Friedman, 1999), we obtain a descent direction from the first variation of V

$$V[F_m + \epsilon f] = V[F_m] + \epsilon \delta V[F_m, f] + O(\epsilon^2) \tag{3}$$

where

$$\delta V[F_m, f] = \frac{d}{d\epsilon} V[F_m + \epsilon f]|_{\epsilon=0}$$

Write $\langle u, v\rangle$ for the usual inner product in L^2. Now $\delta V[F_m, f]$ is a linear functional of f, so there is some $G_V(F_m)$ — which we regard as the "gradient" of the cost — such that $\delta V[F_m, f] = \langle G_V(F_m), f\rangle$. Now we have that $\langle G_V(F_m), f\rangle$ is equal to

$$\int \left\{ \begin{array}{c} f(x) \left[\int_y \frac{\partial}{\partial u} \psi(y, u)\big|_{u=F_m(x)} dP_{y|x} \right] \\ +2\gamma_{\mathcal{M}} \nabla f(x)^t \nabla F_m(x) \end{array} \right\} dP_x \tag{4}$$

Assuming sufficient regularity, recalling that $P_x = 0$ on the boundary of the support of P_x, and using the first Green identity, we have that $\langle G_V(F_m), f\rangle$ is equal to

$$\int \left\{ \begin{array}{c} f(x) \left[\int_y \frac{\partial}{\partial u} \psi(y, u)\big|_{u=F_m(x)} dP_{y|x} \right] \\ +2\gamma_{\mathcal{M}} f(x) \nabla^2_{\mathcal{M}} F_m(x) \end{array} \right\} dP_x \tag{5}$$

where $\nabla^2_{\mathcal{M}} = -\nabla \cdot \nabla_{\mathcal{M}}$ is the *Laplace-Beltrami* operator. The optimal descent direction is a function f that maximizes $-\langle G_V(F_m), f\rangle$ (subject, if necessary to a norm constraint on f). The term $f_{m+1} = \alpha f$ is obtained using line search, minimizing the true cost $V[F_m + \alpha f]$ with respect to α.

2.1.2. Finite Data

Generally, neither P_x nor $P_{x,y}$ are known. Instead, we have sample of labelled data $\{x_i, y_i\}_{i=1}^{l}$, and of unlabelled data $\{x_i\}_{i=l+1}^{u}$. Now integrals become sums over data points. Generally, $\{f_m(x)\}$ will belong to a parametric family of functions (e. g. Radial Basis functions, decision trees, etc).

The Laplacian operator in equation 5 must be discretized. In high dimensions, we cannot triangulate the data set. A smoothed Laplacian is equivalent to the difference between a short-scale average of the data and a long-scale average (e.g. the use of unsharp masking in photography, or the difference of Gaussians in computer vision). The graph Laplacian is a linear operator that takes a function on the graph to the weighted difference between the function value and the average of the K nearest neighbours. This means it is usual to approximate the Laplacian operator with the graph Laplacian $\mathcal{L}$ (e.g. see (Sindhwani et al., 2006)).

Write the graph Laplacian as $\mathcal{L}^{\mathcal{M}}$. The cost function becomes

$$\begin{aligned} V[F] &= \frac{1}{l}\sum_{i=1}^{l}\psi(y_i, F(x_i)) \\ &+ \frac{\gamma_{\mathcal{M}}}{(l+u)K}\sum_{i,j}F(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F(x_j) \end{aligned} \qquad (6)$$

Again, assume we know F_m, and seek f_{m+1}. We will find a function f that maximizes $-\langle G_V(F_m), f\rangle$ then we will weight this function using line search. The inner product is $\langle a, b\rangle = \frac{1}{N}\sum_{i=1}^{N} a(x_i)\cdot b(x_j)$ and we have that

$$\begin{aligned} \langle G_V(F_m), f\rangle &= \frac{1}{l}\sum_i \frac{\partial}{\partial u}\psi(y,u)\Big|_{u=F_m(x_i)} f(x_i) \\ &+ \frac{2\gamma_{\mathcal{M}}}{(l+u)^2}\sum_{i,j} f(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F_{m-1}(x_j) \end{aligned} \qquad (7)$$

Now $-\langle G_V, f\rangle$ is linear in f, and so we should maximize subject to a norm constraint on f. If the norm is fixed, then maximizing this expression is equivalent to minimizing $||G_V - f||^2 = ||G_V||^2 - 2\langle G_V, f\rangle + ||f||^2$. This means any squared loss regression algorithm can be used to find the optimal parameters. Our variational formulation explains why Friedman's *choosing* to make f parallel to the gradient G_V and posing the problem as squared error minimization is natural. Once the descent direction f is found, the final $f_{m+1} = \alpha f$ is obtained using line search, minimizing the true cost $V[F_m + \alpha f]$.

3. Two Examples: Tree and RBF ManifoldBoost Algorithms

We offer two example algorithms with calculations to illustrate our extremely general formalism. For each example, 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 (Friedman, 1999): $\psi(y, F) = \log(1 + \exp(-2yF))$ For this case, whatever classifier we use represents $F(x) = \frac{1}{2}[\log(p(y = 1|x)) - \log(p(y = -1|x))]$ and so at round m, the inner product with the "gradient" becomes,

$$\begin{aligned} \langle G_V(F_m), f\rangle &= \frac{1}{l}\sum_i \frac{2y_i}{1+\exp(2y_iF_m(x_i))}f(x_i) \\ &+ \frac{2\gamma_{\mathcal{M}}}{(l+u)K}\sum_{i,j} f(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F_m(x_j) \end{aligned} \qquad (8)$$

The cases now differ by the procedures used to choose the optimal f

Tree-ManifoldBoost: As in L_2 TreeBoost (Friedman, 1999), we use regression trees as base learners. A tree has the form $f_{m+1}(x) = \sum_{s=1}^{S}\eta_{m+1,s}I[x \in R_s]$, where $I[\cdot] = 1$ if the expression inside is true, and $I[\cdot] = 0$ otherwise.

To minimize $||G_V - f||^2$, we must search for the parameters R_s (which determine the geometry of the tree) and η_s (which determine weights within region). Once a tree has been found, we fix R_s and minimize $V(F_m(x) + \sum_{s=1}^{S}\eta_{m,s}[x \in R_s])$ with respect to $\{\eta_s\}$, using a standard continuous optimization method (BFGS; see (Bertsekas, 1996)). In each round, we use a small number of descent steps to prevent overfitting.

Algorithm 1 Tree ManifoldBoost Algorithm

```
$F_0(x) = 1/2[\log(1+\overline{y}) - \log(1-\overline{y})]$
for $m = 1$ to $M$ do
    Compute $G_V$ as in (8)
    Obtain regression tree $\{R_{s,m}\}$ by minimizing $\sum_i(G_V(x_i) - \sum_s \eta_{m,s}I[x_i \in R_{m,s}])^2$
    Find $\{\eta_{m,s}\}$ using BFGS and $\frac{\partial V}{\partial \eta_s}$, and fixing $\{R_{m,s}\}$
    $F_m(x) = F_{m-1}(x) + \sum_s \eta_{m,s}[x \in R_{m,s}]$
end for
```

The algorithm converges when M rounds have been run, 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 the loss function: $p(y = 1|x) = 1/(1 + \exp(-2F_M(x)))$. This in turn can be used for classification:

$$\tilde{y_i} = \begin{cases} 1 & p(y=1|x)k_{-1,1} > p(y=-1|x)k_{1,-1} \\ -1 & \text{otherwise} \end{cases} \qquad (9)$$

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

Figure 1 shows a toy example for semisupervised classification taken from (Sindhwani et al., 2006) (two moons dataset). The unlabeled datapoints are depicted in green and the diamonds represent the labeled examples (one for each class). The algorithm also can

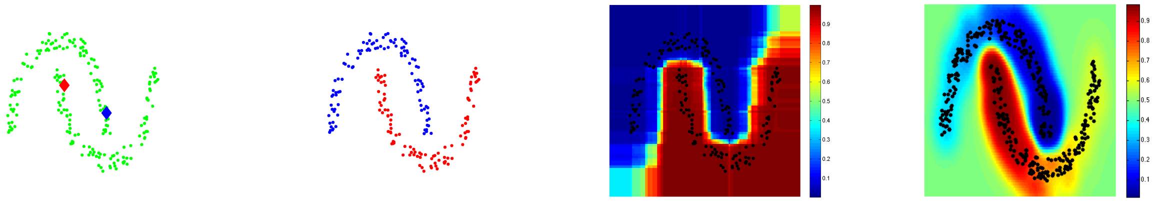

Figure 1. **Semi-supervised learning** using Tree- and RBF-ManifoldBoost. **first** figure from left to right shows a toy example introduced in (Sindhwani et al., 2006) (two moons dataset): the unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The output classification of both algorithm is the same and is depicted on the **second** image (for datapoints). The **third** and **fourth** figures depict the likelihood predicted by the classifiers for the whole space for the tree- and rbf-based classifier respectively.

provide likelihood estimates, as seen in the right figures.

RBF-ManifoldBoost: Tree functions are not the only possible approximation to the “gradient”. Step 4 in algorithm 1 can be modified so that R radial basis function of width σ, each with a weight w_r and centered in a datapoint are chosen as approximation. Again, a BFGS step can be performed to improve the loss by fitting the weights w_r. Algorithm 2 describes this.

Algorithm 2 RBF ManifoldBoost Algorithm

```
$F_0(x) = 1/2[\log(1+\overline{y}) - \log(1-\overline{y})]$
for $m = 1$ to $M$ do
    Compute $G_V$ as in (8)
    Choose $R$ RBFs greedily to minimize $\sum_i (G_V(x_i) - \sum_r w_r RBF_{r,\sigma}(x_i)])^2$
    Find $\{w_r\}$ using BFGS and $\frac{\partial V}{\partial w_r}$
    $F_m(x) = F_{m-1}(x) + \nu \sum_r w_r RBF_{r,\sigma}(x)$
end for
```

Complexity: The procedure itself is linear in $n = l+u$, in the Laplacian neighborhood K, the dimensionality of x and the number of rounds. The complexity of the algorithm depends then on the base regressor, and the computation of the Laplacian matrix. Influence trimming can also be used to get tenfold speedups (Friedman, 1999), although the algorithm is still linear in the number of datapoints.

4. Unsupervised Boosting

The essential step in semi-supervised learning is the observation that similar data items should tend to have similar labels, which means that semi-supervised learning method should be capable of clustering. Our framework can naturally be extended to unsupervised learning, where one wishes to cluster data and the choice of label for a cluster is arbitrary. As there are no labeled data, the first term in equation 6 becomes zero and the problem is,

$$F^* = \arg\min_{F \in \mathcal{H}} \sum_{i,j} F(x_i) \mathcal{L}^{\mathcal{M}}_{i,j} F(x_j) \qquad (10)$$

under the constraints $\sum_i F(x_i) = 0$, $\sum_i F(x_i)^2 = N$ (this is a form of spectral clustering problem, see (Sindhwani et al., 2006); without the constraints, the problem is ill-posed). Our formalism yields a greedy method for this problem, rather than the usual generalized eigenvalue problem. To manage constraints, we use the Augmented Lagrangian method (Bertsekas, 1996), which adds a penalty in each round for constraint violations in the unconstrained problem. We choose F^* to be

$$\arg\min_{F \in \mathcal{H}} \sum_{i,j} F(x_i) \mathcal{L}_{i,j} F(x_j) + \lambda_1^m \sum_i F(x_i) + \ldots$$
$$\lambda_2^m \left(\sum_i F(x_i)^2 - N \right) + \frac{c_1^m}{2} \left(\sum_i F(x_i) \right)^2 + \ldots$$
$$\frac{c_2^m}{2} \left(\sum_i F(x_i)^2 - N \right)^2 \qquad (11)$$

for non-decreasing sequences $\{c_1^m, c_2^m\}_{m=1}^M$. After each round, the values of the Lagrange multipliers are increased by the constraint violation (Bertsekas, 1996) $\lambda_1^{m+1} \leftarrow \lambda_1^m + c_1^m(\sum_i F(x_i))$ and $\lambda^{m+1} \leftarrow \lambda_1^m + c_1^m(\sum_i F(x_i)^2 - N)$. As before, BFGS is applied in each round. Algorithm 3 describes the tree-based version.

The algorithm converges to a local minimum of the constrained problem. This formulation, unlike ISOMAP, naturally takes care of out-of-sample evaluation. Compared to (Sindhwani et al., 2006), the computational complexity is greatly reduced. On the other hand the solution is greedy, and there is no straightforward term for controlling the complexity of the function in the ambient space; this is achieved through the depth of the trees used in the algorithm.

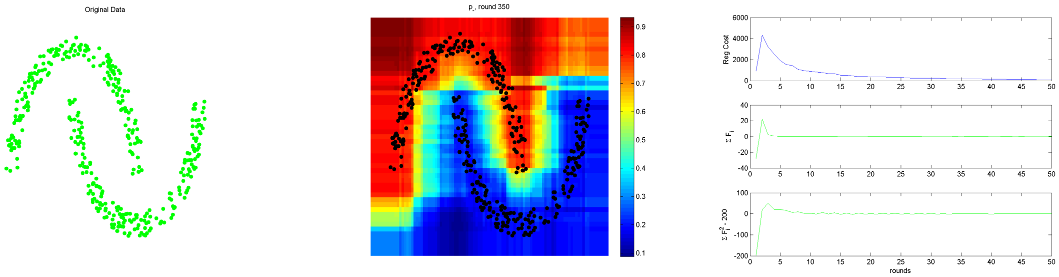

Figure 2. **Unsupervised learning**. The framework can be extended to unsupervised learning. The same problem as in figure 1 is presented without labels (**left** image). The result of the Tree-based algorithm is shown in the **center** image. The plots on the **right** show the constraint violations go to zero as learning progresses. This figure is best viewed in color.

Algorithm 3 Unsupervised Tree ManifoldBoost Algorithm

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

5. Multiclass Case

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

$$\psi(\{y^{(c)}, F^{(c)}(x)\}_{c=1}^{C}) = -\sum_{c'=1}^{C} y^{(c')} \log p^{(c')}(x) \quad (12)$$

where $p^{(c)}(x)$ represents the belief example x belongs to class c, and $y^{(c)}$ is a binary variable which is one if example x belongs to class c. As in (Friedman, 1999) we use the symmetric multiple logistic transform

$$p^{(c)}(x) = \exp F^{(c)}(x) \cdot \left(\sum_{c'=1}^{C} \exp F^{(c')}(x) \right)^{-1} \quad (13)$$

Smoothness of $F^{(c)}$ is enforced by defining the cost $V(\{F^{(c)}\})$ to be

$$\begin{aligned} &\frac{1}{l} \sum_i \psi(\{y_i^{(c)}, F^{(c)}(x_i)\}_{c=1}^{C}) + \dots \\ &\frac{1}{C \cdot (l+u) \cdot K} \sum_{c'} \sum_{i,j} \gamma_{\mathcal{M}}^{(c')} F^{(c')}(x_i) \mathcal{L}_{i,j}^{\mathcal{M}} F^{(c')}(x_j) \end{aligned}$$

The inner product of $f^{(c)}$ with gradient of V becomes, for class c,

$$\begin{aligned} \langle G_V^{(c)}(F_m^{(c)}), f \rangle &= \frac{1}{l} \sum_i (-y_i^{(c)} + p_m^{(c)}(x_i)) f(x_i) \quad (14) \\ &+ \frac{2\gamma_c^{\mathcal{M}}}{C \cdot (l+u)^2} \sum_{i,j} f(x_i) \mathcal{L}_{i,j}^{\mathcal{M}} F_m^{(c)}(x_j) \end{aligned}$$

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 $\{R_{m,s}^{(c)}\}_{s=1}^{S}$ defined by the terminal nodes are fixed, and the parameters $\eta_{m,s}^{(c)}$ 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 $\eta_{m,s}^{(c)}$ have to be computed.

Once the final $\{F_M^{(c)}(x)\}$ are computed, the probability for a given example of each label can be estimated and thus the label can be classified as $\hat{c}(x_i) = \arg\min_c \sum_{c'=1}^{C} k_{c,c'} p_M^{(c')}(x)$ for costs $k_{c,c'}$ when label c is assigned when label c' 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 inluence trimming is used (Friedman, 1999).

6. Experiments and Discussion

6.1. Comparison to Other Regularized Boosting Algorithms

Kegl et. al. (Kégl & Wang, 2005) introduce RegBoost, an extension to AdaBoost which incorporates a weight decay that depends on a Laplacian regularizer. Our approach is different several senses: first, ours is based on the GradientBoost framework while theirs is based on AdaBoost, second, in the sense that ManifoldBoost does not require manifold-regularized base learners. This makes their approach limited in

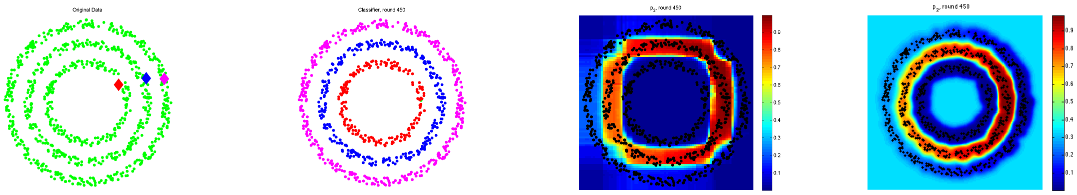

Figure 3. The framework also handles **multiclass learning**. **Left** figure shows another toy example (three rings): the unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The output of the algorithm is depicted on the **center** figure. The blue classification function $F^2(x)$ is shown on the **right**. This figure is best viewed in color.

Algorithm 4 K-Tree ManifoldBoost Algorithm

```
Let $p_0^{(c)}$ be the frequencies of each class $c$.
$F_0^{(c)}(x) = \log p_0^{(c)} - \frac{1}{C}\sum_{c'=1}^{C} \log p_0^{(c)}$
for $m = 1$ to $C$ do
    Compute $p_m^{(c)}(x)$ as in eq. 13 for all $c$.
    for $c = 1$ to $C$ do
        Compute $G_V^{(c)}$ as in (14)
        Obtain regression tree $\{R_{m,s}^{(c)}\}$ by minimizing $\sum_i (G_V^{(c)}(x_i) - \sum_s \eta_{m,s}^{(c)} I[x_i \in R_{m,s}^{(c)}])^2$
    end for
    Find $\{\eta_{m,s}^{(c)}\}$ using BFGS and $\frac{\partial V}{\partial \eta_{m,s}}$, and fixing $\{R_{m,s}^{(c)}\}$ for all $c$.
    $F_m^{(c)}(x) = F_{m-1}^{(c)}(x) + \sum_s \eta^(c))_{m,s} I[x \in R_{m,s}^{(c)}]$ for all $c$.
end for
```

the types of learners to be used (they use stumps only). Also, the ensemble classifier should be smooth on the manifold, but regularizing each of the base learners may result in over-smoothing of the overall solution. We compare our results with (Kégl & Wang, 2005) on standard UCI benchmark datasets. Whenever possible we tried to use the same configuration as (Kégl & Wang, 2005)[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. The learning rate was set to $\nu = 0.1$ after (Friedman, 1999). 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 (M. Belkin & Niyogi, 2004) as reported in (Kégl & Wang, 2005).

In the fully supervised problems, there is a difference in performance for the Sonar dataset, an impovement 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. (Kégl & Wang, 2005) also tests the algorithm under semi-supervision, using 100 labeled and 251 unlabeled examples. We ran our algorithm under the same conditions, using the stumps to prevent overfitting. In this case our algorithm outperforms (Kégl & Wang, 2005) and (M. Belkin & Niyogi, 2004), as our mean performance over 10 runs is more than a standard deviation above theirs. No variance of results is reported in (Kégl & Wang, 2005).

(Chen & Wang, 2008) proposes an interesting alternative approach to regularized boosting based on the more traditional framework of boosting "weak" learners outlined in (Mason et al., 2000). As a consequence, they need to assign pseudo-class labels to unlabeled data (labels assigned with the current $F_t(x)$) while learning the ensemble. In contrast, ManifoldBoost uses base regressors to measure the confidence of the prediction and does not commit to $\{-1, +1\}$ classification at each step. Smoothing this seems more natural in a formulation that penalizes second derivatives (Laplacian cost).

6.2. Comparison to Other Semi-Supervised Learning Algorithms

We measured the performance of our two-class RBF-ManifoldBoost algorithm on the SSL data sets, a standard benchmark for semi-supervised learning problems introduced in (Chapelle et al., 2006), and compared with the 14 other state-of-the-art semi-supervised learning algorithms discussed there. In table 2 we present results for five data sets, 2 of which are cluster-like and 3 manifold-like. On the manifold-like data sets, we are at the state of the art and no single algorithm does uniformly better than us. On the cluster-

[1] The breast cancer dataset was not used because (Kégl & Wang, 2005) does not explain what metric they use for categorical data in the Lalacian, making any comparison meaningless.

Table 1. Performance results of Tree-ManifoldBoost and different boosting approaches on standard UCI datasets, as reported in (Kégl & Wang, 2005) (variance in parenthesis)

	Ionosphere	Pima Indians	Sonar	Ionosphere
Algorithm	Train / Test	Train / Test	Train / Test	(semisup.)
AdaBoost	0% / 9.2% (7.1)	10.9% / 25.3% (5.3)	0% / 32.5% (19.8)	-
RegBoost	0% / 7.7% (6.0)	16.0% / 23.3% (6.8)	0% / 29.8% (18.8)	12%
Tree-ManifoldBoost	0% (0) / 6.5% (4.8)	6.5% (0.5) / 24.0% (5.2)	0% (0) / 18.7% (6.1)	10.4% (0.8)
Belkin et al., 04 [2]	- / -	- / -	- / -	18%

like data sets, our performance is good compared to most other regularization-based and manifold learners but is not as good as the specialized clustering algorithms Cluster-Kernel and SGT (Spectral Graph Transducer).

Parameter search was performed following section 21.2.5 of (Chapelle et al., 2006) when possible, using the same ranges for γ, the RBF width σ, distance metric, K, etc. For the base regressors, we used $R \in \{15, 30\}$ as the numbers of RBFs, and $M = 500$ rounds. The learning rate was again chosen as $\nu = 0.1$. These parameters were obtained in small-scale experiments and then fixed. Results reported are the means over the different splits.

The running times for a MATLAB implementation on a 2 GHz machine was in the order of minutes. Unfortunately, running times for the other algorithms were not reported in (Chapelle et al., 2006).

6.3. `SecStr` Data Set

We also ran experiments on the `SecStr` data set (Chapelle et al., 2006), which is a problem of predicting the secondary structure of protein sequences from their amino acid chains. This is a large-scale and challenging data set with 83,000 labeled and 1.2 million unlabeled examples. Semi-supervised algorithms have made little improvement to this benchmark so far (Table 3), and the best result is the manifold-regularized learning algorithm (Sindhwani et al., 2006), which yields a 29% error rate on a subset of the data with 10,000 labeled and 73,000 unlabeled examples.

Tree-ManifoldBoost with $\gamma \in \{0, 10^{-5}, 10^{-3}, 0.1, 1\}$ achieved similar performance on the same subset in approximately 45 minutes of training time (after computing the Laplacian matrix). We used stumps, $K = 6$ and $\nu = 0.05$. No model selection was performed. We used as similarity measure the Hamming distance between the best alignment of sequences. The results reported are the mean over the 10 splits.

When we used the whole dataset (1.3 million sequences) with $\gamma \in \{0, 10^{-3}, 1\}$, there is virtually no performance improvement. This may be due to the smaller parameter search space, or to peculiarities of the dataset. When we analysed the structure of the manifold on the labeled subset, we observed that almost 20% of sequences at distance 1 (that is, shifted by one position to the left or right) had a different label. Therefore the manifold assumption is not strong on this set.

As far as we know, this is the first time results are reported on the complete `SecStr` dataset. Our algorithm is efficient and therefore can handle datasets of this size. Learning time is in the order of three hours for 1.3 million samples (leaving aside the computation of the graph Laplacian, which took significantly longer)

Table 3. Error rates on `SecStr` dataset. l is the number of labeled examples.

l	100	1000	10000
SVM	44.59	33.71	
Cluster Kernel	42.95	34.03	
QC randsub (CMN)	42.32	40.84	
QC smartonly (CMN)	42.14	40.71	
QC smartsub (CMN)	42.26	40.84	
Boosting (assemble)			32.21
LapRLS	42.59	34.17	28.55
LapSVM	43.42	33.96	28.53
Tree-ManifoldBoost (83K)	42.70	33.43	28.96
Tree-ManifoldBoost (1.3M)	43.28	33.42	29.07

7. Conclusion

We have presented a new boosting framework for regularized learning in a greedy, stage-wise procedure. It is flexible enough to handle the whole range of supervision, from fully supervised classification to unsupervised clustering. The framework is general, accepts many different function approximation techniques, is

Table 2. Error rates for data sets from (Chapelle et al., 2006). l is the number of labeled examples.

	Manifold-like						Cluster-like			
	$l = 10$			$l = 100$			$l = 10$		$l = 100$	
	BCI	Digit1	USPS	BCI	Digit1	USPS	g241c	g241d	g241c	g241d
1-NN	49.00	13.65	16.66	48.67	3.89	5.81	47.88	46.72	43.93	42.45
SVM	49.85	30.60	20.03	34.31	5.53	9.75	47.32	46.66	23.11	24.64
21.2.8 MVU + 1-NN	47.95	14.42	23.34	47.89	2.83	6.50	47.15	45.56	43.01	38.20
21.2.8 LEM + 1-NN	48.74	23.47	19.82	44.83	6.12	7.64	44.05	43.22	40.28	37.49
21.2.4 QC + CMN	50.36	9.80	13.61	46.22	3.15	6.36	39.96	46.55	22.05	28.20
21.2.6 Discrete Reg.	49.51	12.64	16.07	47.67	2.77	4.68	49.59	49.05	43.65	41.65
21.2.1 TSVM	49.15	17.77	25.20	33.25	6.15	9.77	24.71	50.08	18.46	22.42
21.2.1 SGT	49.59	8.92	25.36	45.03	2.61	6.80	22.76	18.64	17.41	9.11
21.2.10 Cluster-Kernel	48.31	18.73	19.41	35.17	3.79	9.68	48.28	42.05	13.49	4.95
21.2.3 Data-Dep. Reg.	50.21	12.49	17.96	47.47	2.44	5.10	41.25	45.89	20.31	32.82
21.2.11 LDS	49.27	15.63	17.57	43.97	3.46	4.96	28.85	50.63	18.04	23.74
21.2.5 Laplacian RLS	48.97	5.44	18.99	31.36	2.92	4.68	43.95	45.68	24.36	26.46
21.2.7 CHM (normed)	46.90	14.86	20.53	36.03	3.79	7.65	39.03	43.01	24.82	25.67
RBF-ManifoldBoost	47.12	19.42	19.97	32.17	4.29	6.65	42.17	42.80	22.87	25.00

efficient and fast at each round of boosting, handles multi-class and wholly unsupervised problems, and produces results at the state of the art. We are working on understanding important aspects of the algorithm, in particular, generalization, error bounds, convergence and local minima.

References

Bertsekas, D. P. (1996). *Constrained optimization and lagrange multiplier methods*. Academic Press.

Bickel, P., & Li, B. (2007). Local polynomial regression on unknown manifolds. *IMS Lecture Notes Monograph Series: Complex Datasets and Inverse Problems: Tomography, Networks and Beyond* (pp. 177–186).

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised Learning*. MIT Press.

Chen, K., & Wang, S. (2008). Regularized boost for semi-supervised learning. *NIPS 20* (pp. 281–288).

Donoho, D. L., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, *100*, 5591–5596.

Dyn, N., Levin, D., & Rippa, S. (1986). Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing*, *7*, 639–659.

Friedman, J. (1999). *Greedy function approximation: a gradient boosting machine* (Technical Report). Dept. of Statistics, Stanford University.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *ICML* (pp. 200–209).

Kégl, B., & Wang, L. (2005). Boosting on manifolds: Adaptive regularization of base classifiers. *NIPS 17* (pp. 665–672).

Lafferty, J., & Wasserman, L. (2007). Statistical analysis of semi-supervised regression. *NIPS 20* (pp. 801–808).

M. Belkin, I. M., & Niyogi, P. (2004). Regression and regularization on large graphs. *COLT* (pp. 824–831).

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. *NIPS 12* (pp. 512–518).

Niyogi, P. (2008). *Manifold regularization and semi-supervised learning: Some theoretical analyses* (Technical Report). University of Chicago. Technical Report TR-2008-01, Computer Science Dept.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Sindhwani, V., Belkin, M., & Niyogi, P. (2006). The geometric basis of semi-supervised learning. In Chapelle, Schoelkopf and Zien (Eds.), *Semi-supervised learning*. MIT Press.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Zhu, X. (2006). *Semi-supervised learning literature review* (Technical Report). University of Wisconsin.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *ICML* (pp. 912–919).