Local Linear Semi-supervised Regression

Mugizi Robert Rwebangira and John Lafferty

February 2009
CMU-CS-09-106

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored by the National Science Foundation under contract no. IIS-0427206, National Science Foundation under contract no. CCR-0122581, National Science Foundation under contract no. IIS-0312814, US Army Research Office under contract no. DAAD190210389, and SRI International under contract no. 03660211. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: semi-supervised, regression, unlabeled data

Abstract

In many machine learning application domains, obtaining labeled data is expensive but obtaining unlabeled data is much cheaper. For this reason there has been growing interest in algorithms that are able to take advantage of unlabeled data. In this report we propose an algorithm for using unlabeled data in a regression problem. The idea behind the method is to do manifold regularization using local linear estimators. This is the first extension of local linear regression to the semi-supervised setting. We present experimental results on both synthetic and real data and show that this method tends to perform better than methods which only utilize the labeled data.

Contents

1.1 Introduction and Motivation

In many machine learning domains, labeled data is much more expensive than labeled data. For example, labeled data may require a human expert or an expensive experimental process to classify each example. For this reason there has been a lot of interest in the last few years in machine learning algorithms than can make use of unlabeled data [26]. The majority of such proposed algorithms have been applied to the classification task. In this report we focus on using unlabeled data in regression. In particular, we present LLSR, the first extension of Local Linear Regression to the problem of semi-supervised learning.

1.1.1 Regression

Regression is a fundamental tool in statistical analysis. At its core regression aims to model the relationship between 2 or more random variables. For example, an economist might want to investigate whether more education leads to an increased income. A natural way to accomplish this is to take number of years of education as the dependent variable and annual income as the independent variable and to use regression analysis to determine their relationship.

Formally, we are given as input $(X_1, Y_1), ...(X_n, Y_n)$ where the X_i are the independent variables and Y_i are the dependent variables. We want to predict for any X, the value of the corresponding Y. There are two main types of techniques used to accomplish this:

1. Parametric regression: In this case we assume that the relationship between the variable is of a certain type (e.g. a linear relationship) and we are concerned with learning the parameters for a relationship of that type which best fit the data.
2. Non-parametric regression: In this case we do not make any assumptions about the type of relationship that holds between the variables, but we derive this relationship directly from the data.

Regression analysis is heavily used in the natural sciences and in social sciences such as economics, sociology and political science. A wide variety of regression algorithms are used including linear regression, polynomial regression and logistic regression (among the parametric methods) and kernel regression and local linear regression (among the non-parametric methods). A further discussion of such methods can be found in any introductory statistics textbook [77, 78].

In semi-supervised regression in addition to getting the dependent and independent variables X and Y we are also given an addition variable R which indicates whether or not we observe that value of Y. In other words we get data $(X_1, Y_1, R_1), ...(X_n, Y_n, R_n)$ and we observe Y_i only if $R_i = 1$.

We note that the problem of semi-supervised regression is more general than the semi-supervised classification problem. In the latter case the Y_i are constrained to have only a finite number of possible values whereas in regression the Y_i are assumed to be continuous. Hence some algorithms designed for semi-supervised classification (e.g. graph mincut[13]) are not applicable to the more general semi-supervised regression problem. Other algorithms such as Gaussian Fields [85]) are applicable to both problems.

Although semi-supervised regression has received less attention than semi-supervised classification, a number of methods have been developed dealing specifically with this problem. These include the transductive regression algorithm proposed by Cortes and Mohri [24]and co-training style algorithms proposed by Zhou and Li [82], Sindhwani et al.[68] and Brefeld et al.[18].

1.1.2 Local Linear Semi-supervised Regression

In formulating semi-supervised classification algorithms, an often useful motivating idea is the Cluster Assumption: the assumption that the data will naturally cluster into clumps that have the same label. This notion of clustering does not readily apply to regression, but we can make a somewhat similar "smoothness" assumption: we expect the value of the regression function to not "jump" or change suddenly. In both cases we expect *examples that are close to each other to have similar values.*

A very natural way to instantiate this assumption in semi-supervised regression is by finding estimates $\hat{m}(x)$ that minimize the following objective function (subject to the constraint that $\hat{m}(x_i) = y_i$ for the labeled data):

$$\sum_{i,j} w_{ij}(\hat{m}(x_i) - \hat{m}(x_j))^2$$

where $\hat{m}(x_i)$ is the *estimated* value of the function at example x_i, w_{ij} is a measure of the similarity between examples x_i and x_j and y_i is the value of the function at x_i (only defined on the labeled examples).

This is exactly the objective function minimized by the Gaussian Fields algorithm proposed by Zhu, Gharamani and Lafferty [85].

This algorithm has several attractive properties:

1. The solution can be computed in closed form by simple matrix operations.
2. It has interesting connections to Markov Random Fields, electrical networks,spectral graph theory and random walks. For example, for the case of boolean weights w_{ij}, the estimates $\hat{m}(x_i)$ produced from this optimization can be viewed as the probability that a random walk starting at x_i on the graph induced by the weights, would reach a labeled positive example

before reaching a labeled negative example. These connections are further explored in works by Zhu et al. [84, 85].

However, it suffers from at least one major drawback when used in regression: It is "locally constant." This means that it will tend to assign the same value to all the example near a particular labeled example and hence produce "flat neighborhoods."

While this behavior is desirable in classification, it is often undesirable in regression application where we frequently assume that the true function is "locally linear." By locally linear we mean that(on some sufficiently small scale) the value of an example is a "linear interpolation" of the value of its closest neighbors. (From a mathematical point of view local linearity is a consequence of a function being differentiable.)Hence if our function is of "locally linear" type then a "locally constant" estimator will not provide good estimates and we would prefer to use an algorithm that incorporates a local linearity assumption.

The supervised analogue of Gaussian Fields is Weighted Kernel Regression (also known as the Nadaraya-Watson estimator) which minimizes the following objective function:

$$\sum_i w_i(y_i - \hat{m}(x))^2$$

where $\hat{m}(x)$ is the value of the function at example x,y_i is the value of x_i and w_i is a measure of the similarity between x and x_i.

In the supervised case, there already exists an estimator that has the desired property: Local Linear Regression, which finds β_x so as to minimize the following objective function:

$$\sum_{i=1}^{n} w_i(y_i - \beta_x^T X_{xi})^2$$

with

$$X_{xi} = \begin{pmatrix} 1 \\ x_i - x \end{pmatrix}.$$

Hence, a suitable goal is to derive a local linear version of the Gaussian Fields algorithm. Equivalently, we want a semi-supervised version of the Local Linear estimator.

In the remainder of this report we will give some background on non-parametric regression, describe the Local Linear Semi-supervised Regression algorithm and show the results of some experiments on real and synthetic data.

2.1 Background

The general problem of estimating a function from data has been extensively studied in the statistics community. There are two broad classes of methods that are used: Parametric and

Non-parametric. We describe these in turn below.

2.1.1 Parametric Regression methods

These approaches assume that the function that is being estimated is of a particular type and then try to estimate the parameters of the function so that it will best fit the observed data.

For example, we may assume that the function we seek is linear but the observations have been corrupted with Gaussian noise:

$$y = \beta^T x + \epsilon_i \text{ (with } \epsilon_i \sim N(0, \sigma^2)).$$

Parametric methods have some advantages compared to non-parametric methods:

1. They are usually easier to analyze mathematically.
2. They usually require less data in order to learn a good model.
3. They are typically computationally less intensive.

However, they also have several disadvantages, especially if the assumptions are not entirely correct.

2.1.2 Non-Parametric Regression methods

These approaches do not assume that the function we are trying to estimate is of a specific type. i.e given

$$y_i = m(x_i) + \epsilon_i$$

the goal is to estimate the value of $m(x)$ at each point.

The main advantage of non-parametric approaches is that they are more flexible than parametric methods and hence they are able to accurately represent broader classes of functions.

2.1.3 Linear Smoothers

Linear smoothers are a class of non-parametric methods in which the function estimates are a linear function of the response variable:

$$\hat{y} = Ly$$

where $\hat{y}$ are the new estimates, y are the observations and L is a matrix which may be constructed based on the data.

Linear smoothers include most commonly used non-parametric regression algorithms and in particular all the algorithms we have discussed so far are linear smoothers. We discuss how we can view each of them as linear smoothers below.

Weighted Kernel Regression

The objective is to find the number $\hat{m}(x)$ that minimizes the least squares error

$$\sum_i w_i(y_i - \hat{m}(x))^2.$$

The minimizer of this objective function is

$$\hat{m}(x) = \frac{\sum_i w_i y_i}{\sum_i w_i} = Ly.$$

Local Linear Regression

The objective is to find β_x that minimizes the least squares error

$$\sum_{i=1}^{n} w_i(y_i - \beta_x^T X_{xi})^2$$

where

$$X_{xi} = \begin{pmatrix} 1 \\ x_i - x \end{pmatrix}.$$

The minimizer of the objective function is

$$\hat{\beta}_x = (X_x^T W_x X_x)^{-1} X_x^T W_x y$$

where X_x is the $n \times (d+1)$ matrix (X_{xi}^T) and the matrix W_x is the $n \times n$ diagonal matrix $diag(w_i)$.

The local linear estimate of $m(x)$ is

$$\hat{m}(x) = e_1^T (X_x^T W_x X_x)^{-1} X_x^T W_x y = Ly.$$

Gaussian Fields

The objective is to find f that minimizes the energy functional

$$\mathcal{E}(f) = \sum_{i,j} w_{ij}(f_i - f_j)^2$$

with the constraint that some of the f_i are fixed.

It can be shown that

$$\mathcal{E}(f) = f^T \Delta f$$

where $\Delta = D - W$ is the *combinatorial graph Laplacian* of the data, W is the weight matrix and D is the diagonal matrix with $D_{ii} = \sum_j w_{ij}$.

If f_L denotes the observed labels and f_U denotes the unknown labels then the minimizer of the energy functional is

$$f_U = \Delta_{UU}^{-1} \Delta_{UL} f_L = Ly$$

where Δ_{UU} and Δ_{UL} are the relevant submatrices of the graph Laplacian.

3.1 Local Linear Semi-supervised Regression

Our goal is to derive a semi-supervised analogue of Local Linear Regression so we want it to have the following properties.

1. It should fit a linear function at each point like Local Linear Regression.
2. The estimate for a particular point should depend on the estimates for all the other examples like in Gaussian Fields. In particular we want to enforce some smoothness in how the linear function changes.

More specifically, let X_i and X_j be two examples and β_i and β_j be the local linear fits at X_i and X_j respectively.

Let

$$X_{ji} = \begin{pmatrix} 1 \\ X_i - X_j \end{pmatrix}.$$

Then $X_{ji}^T \beta_j$ is the estimated value at X_i using the local linear fit at X_j. Thus the quantity

$$(\beta_{i0} - X_{ji}^T \beta_j)^2$$

is the squared difference between the smoothed estimate at X_i and the estimated value at X_i using the local fit at X_j. This situation is illustrated in figure 3.1

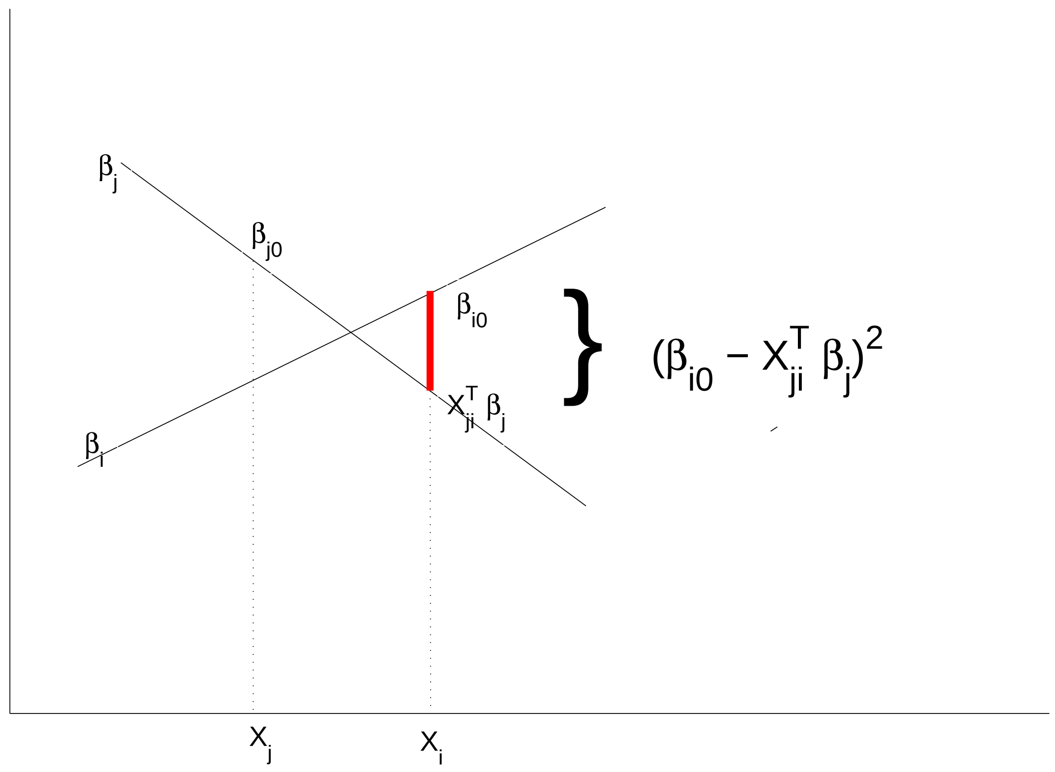

Figure 3.1: We want to minimize the squared difference between the smoothed estimate at X_i and the estimated value at X_i using the local fit at X_j

We can take the sum of this quantity over all pairs of examples as the quantity we want to minimize:

$$\Omega(\beta) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(\beta_{i0} - X_{ji}^T\beta_j)^2$$

with the constraint that some of the B_i are fixed.

Lemma 1.1 The manifold regularization functional $\Omega(\beta)$ can be written as the quadratic form

$$\Omega(\beta) = \beta^T \Delta \beta$$

where the local linear Laplacian $\Delta = [\Delta_{ij}]$ is the $n \times n$ block matrix with $(d+1)\times(d+1)$ blocks $\Delta_{ij} = diag(D_i) - [W_{ij}]$ where

$$D_i = \frac{1}{2}\sum_j w_{ij}(e_1 e_1^T + X_{ij}X_{ij}^T)$$

and

$$W_{ij} = w_{ij} \left(\begin{array}{cc} 1 & (X_j^T - X_i^T) \\ (X_j - X_i) & 0 \end{array} \right).$$

Proof:

Let n be the number of examples.
Let d be the number of dimensions.
Let X be a $d \times n$ matrix (the data).
Let W be a ***symmetric*** $n \times n$ matrix. (the similarity between X_i and X_j)
Let β be a $n \times (d+1)$ length vector. (the coefficients we want to learn).
Let β_i be the β_{id+1} to $\beta_{i(d+1)}$ coefficients in β. (the coefficients for X_i)
Let X_i be $[X_{i1}\ \ldots\ X_{id}]^T$ (The i^{th} column of X)
Let X_{ij} be $[1\ (X_i - X_j)^T]^T$
Let e_1 be $[1\ 0\ 0 \ldots\ 0]^T$
Let d_i be $\sum_j W_{ij}$

Starting with the objective function:

$$\Omega(\beta) = \sum_i \sum_j W_{ij}(X_{ii}^T B_i - X_{ij}^T B_j)^2 \text{ (by definition)}$$

We first expand the expression to get:

$$= \sum_i \sum_j W_{ij} B_i^T X_{ii} X_{ii}^T B_i - \sum_i \sum_j 2W_{ij} B_i^T X_{ii} X_{ij}^T B_j + \sum_i \sum_j W_{ij} B_j^T X_{ij} X_{ij}^T B_j \text{ (expanding)}$$

Taking the first term we note that:

$$\sum_i \sum_j W_{ij} B_i^T X_{ii} X_{ii}^T B_i = \sum_i B_i^T d_i X_{ii} X_{ii}^T B_i = \mathcal{B}^T \Delta_1 \mathcal{B}$$

Where $[(\Delta_1)_{ii}] = d_i X_{ii} X_{ii}^T$

Next looking at the second term we get:

$$S = \sum_i \sum_j 2W_{ij} B_i^T X_{ii} X_{ij}^T B_j = \sum_i \sum_j 2B_i^T W_{ij} X_{ii} X_{ij}^T B_j$$
$$= \sum_i \sum_j 2B_j^T W_{ji} X_{jj} X_{ji}^T B_i = \sum_i \sum_j 2B_i^T W_{ij} X_{ji} X_{jj}^T B_j$$

So we can derive the following expression for the second term:

$$S = \frac{1}{2}(S+S) = \frac{1}{2} \sum_i \sum_j 2B_i^T W_{ij} (X_{ii} X_{ij}^T + X_{ji} X_{jj}^T) B_j = \mathcal{B}^T \Delta_2 \mathcal{B}$$

Where $[(\Delta_2)_{ij}] = W_{ij}(X_{ii}X_{ij}^T + X_{ji}X_{jj}^T)$

Finally looking at the third term in the original expression:

$$\sum_i \sum_j W_{ij} B_j^T X_{ij} X_{ij}^T B_j = \sum_j B_j^T (\sum_i W_{ij} X_{ij} X_{ij}^T) B_j = \mathcal{B}^T \Delta_3 \mathcal{B}$$

Where $[(\Delta_3)_{ii}] = \sum_j W_{ij} X_{ji} X_{ji}^T$

So in conclusion:

$$\Omega(\beta) = \sum_i \sum_j W_{ij}(X_{ii}^T B_i - X_{ij}^T B_j)^2 = \mathcal{B}^T \Delta \mathcal{B}$$

where $\Delta = diag(D_i) - [W_{ij}]$ and

$$diag(D_i) = \Delta_1 + \Delta_3$$

$$[W_{ij}] = \Delta_2$$

■

The term we just derived is the local linear manifold regularizer. Now we add another term to account for the labeled examples and minimize the sum.

Lemma 1.2.

Let $\mathcal{R}_\gamma(\beta)$ be the manifold regularized risk functional:

$$\mathcal{R}_\gamma(\beta) = \frac{1}{2} \sum_{j=1}^{n} \sum_{R_i=1} w_{ij}(Y_i - X_{ji}^T \beta_j)^2 + \frac{\gamma}{2}\Omega(\beta)$$

Here $R_i = 1$ means i is a labeled example and γ is a regularization constant. We can simplify this to:

$$= \frac{1}{2} \sum_{j=1}^{n} (Y_i - X_j\beta_j)^T W_j (Y_i - X_j\beta_j) + \frac{\gamma}{2}\beta^T \Delta \beta$$

The minimizer of this risk can be written in closed form as:

$$\hat{\beta}(\gamma) = (diag(X_j^T W_j X_j) + \gamma\Delta)^{-1})(X_1^T W_1 Y, \ldots, X_1^T W_1 Y)^T$$

Proof.

The expression we want to minimize is:

$$\frac{1}{2}\sum_{j=1}^{n}\sum_{R_i=1} W_{ij}(Y_i - X_{ji}^T B_j)^2 + \frac{\gamma}{2}\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(B_{i0} - X_{ji}^T B_j)^2$$

Using the previous lemma this is equivalent to:

$$= \frac{1}{2}\sum_{j=1}^{n}(Y - X_j B_j)^T W_j (Y - X_j B_j) + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

We can expand the first term:

$$= \frac{1}{2}\sum_{j=1}^{n} Y^T W_j Y - \frac{1}{2}\sum_{j=1}^{n} B_j^T X_j^T W_j Y - \frac{1}{2}\sum_{j=1}^{n} Y^T W_j X_j B_j + \frac{1}{2}\sum_{j=1}^{n} B_j^T X_j^T W_j X_j B_j + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

After some rearrangement we get:

$$= \frac{1}{2} Y^T (\sum_{j=1}^{n} W_j) Y - B^T [X_j^T W_j Y] + \frac{1}{2}\mathcal{B}^T diag(X_j^T W_j X_j)\mathcal{B} + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

Now we let $Q = diag(X_j^T W_j X_j), P = [X_j^T W_j Y], C = \frac{1}{2} Y^T (\sum_{j=1}^{n} W_j) Y$

The expression now becomes:

$$= C - \mathcal{B}^T P + \frac{1}{2}\mathcal{B}^T Q \mathcal{B} + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

Now we differentiate with respect to $\mathcal{B}$ and set to zero:

$$= -P + Q\mathcal{B} + \gamma \Delta \mathcal{B} = 0$$

which leads to:

$$=> \mathcal{B} = (Q + \gamma\Delta)^{-1} P$$

So in conclusion the expression which minimizes the manifold regularized risk functional is:

$$\hat{\beta}(\gamma) = (diag(X_j^T W_j X_j) + \gamma\Delta)^{-1})(X_1^T W_1 Y, \ldots, X_1^T W_1 Y)^T$$

■

4.1 An Iterative Algorithm

As defined here the LLSR algorithm requires inverting a $n(d+1) \times n(d+1)$ matrix. This may be impractical if n and d are large. For example for $n = 1500$, $d = 199$ the closed form computation would require inverting a $300,000 \times 300,000$ matrix, a matrix that would take roughly 720 GB

of memory to store in Matlab's standard double precision format.

Hence, it is desirable to have a more memory efficient method for computing LLSR. An iterative algorithm can fulfill this requirement.

Theorem 1.3 If we initially assign B_i arbitrary values for all i, and repeatedly apply the following formula, then the B_i will converge to the minimum of the LLSR objective function:

$$B_i = [\sum_{R_j=1} W_{ij}X_{ji}X_{ji}^T + \gamma \sum_j W_{ij}(X_{ii}X_{ii}^T + X_{ji}X_{ji}^T)]^{-1}(\sum_{R_j=1} W_{ij}X_{ji}Y_j + \gamma \sum_j W_{ij}(X_{ii}X_{ij}^T + X_{ji}X_{jj}^T)B_j)$$

Proof.
The objective function is:

$$\frac{1}{2}\sum_{j=1}^{n}\sum_{R_i=1} W_{ij}(Y_i - X_{ij}^T B_j)^2 + \frac{\gamma}{2}\frac{1}{2}\sum_i\sum_j W_{ij}(X_{ii}^T B_i - X_{ij}^T B_j)^2$$

If we differentiate w.r.t to B_i and set to 0 we get:

$$[\sum_{R_j=1} W_{ij}X_{ji}X_{ji}^T + \gamma \sum_j W_{ij}(X_{ii}X_{ii}^T + X_{ji}X_{ji}^T)]B_i = \sum_{R_j=1} W_{ij}X_{ji}Y_j + \gamma \sum_j W_{ij}(X_{ii}X_{ij}^T + X_{ji}X_{jj}^T)B_j$$

After rearranging:

$$\Rightarrow B_i = [\sum_{R_j=1} W_{ij}X_{ji}X_{ji}^T + \gamma \sum_j W_{ij}(X_{ii}X_{ii}^T + X_{ji}X_{ji}^T)]^{-1}(\sum_{R_j=1} W_{ij}X_{ji}Y_j + \gamma \sum_j W_{ij}(X_{ii}X_{ij}^T + X_{ji}X_{jj}^T)B_j)$$

Hence the iterative algorithm is equivalent to doing "exact line search" for each B_i. In other words, given that all other variables are constant, we find the optimal value of B_i so as to minimize the objective function.

This means that at each step the value of the objective function must decrease. But since the objective function is a sum of squares, it can never be less than 0. Hence the iteration will eventually converge to a local minima. If we further note that the objective function is convex, then it only has one global minimum and that will be the only fixed point of the iteration.Hence the iteration will converge to the global minimum of the objective function. ■

As we noted previously, if $n = 1500$, $d = 199$ the closed form computation would require $720\ GB$ of memory but the iterative computation only requires keeping a vector of length $n \times (d+1)$ and inverting a $(d+1) \times (d+1)$ matrix which in this case only takes 2.4 MB of memory. So we save a factor of almost 300,000 in memory usage in this example.

5.1 Experimental Results

To understand the behavior of the algorithm we performed some experiments on both synthetic and real data.

5.1.1 Algorithms

We compared two purely supervised algorithms with Local Linear Semi-supervised Regression.

WKR - Weighted Kernel Regression

LLR - Local Linear Regression

LLSR - Local Linear Semisupervised Regression

5.1.2 Parameters

There are two free parameters that we have to set for LLSR (and one (kernel bandwidth) for WKR and LLR).

h - kernel bandwidth
γ - Amount of Semisupervised Smoothing

5.1.3 Error metrics

LOOCV MSE - Leave-One-Out-Cross-Validation Mean Squared Error. This is what we actually try to optimize (analogous to training error).

MSE - This is the true Mean Squared Error between our predictions and the true function (analogous to test error).

5.1.4 Computing LOOCV

Since we do not have access to the MSE we pick the parameters so as to minimize the LOOCV MSE. Computing the LOOCV MSE in a naïve way would be very computationally expensive since we would have to run the algorithm $O(n)$ times.

Fortunately for a linear smoother we can compute the LOOCV MSE by running the algorithm only once. More precisely if $\hat{y} = Ly$ then

$$\text{LOOCV MSE} = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - L_{ii}} \right)^2$$

5.1.5 Automatically selecting parameters

We experimented with a number of different ways of picking the parameters. In these experiments we used a form of coordinate descent.

Picking one parameter

To pick one parameter we just reduce the bandwidth until there is no more improvement in LOOCV MSE.

1. Initially set bandwidth to 1 and compute LOOCV MSE.
2. Set $h = h/2$ and compute the resulting LOOCV MSE.
3. If the LOOCV MSE decreases then go back to step 2 else go to step 4
4. Output the h which had the lowest LOOCV MSE.

Picking two parameters

To pick two parameters we succesively halve the parameter which yields the biggest decrease in LOOCV MSE.

1. Initially set both bandwidth and smoothing parameter to 1 and compute LOOCV MSE.
2. Set $h = h/2$ while leaving γ alone and compute LOOCV MSE.
3. Set $\gamma = \gamma/2$ while leaving h alone and compute LOOCV MSE.
4. If either steps 2 or 3 decreased the LOOCV MSE then choose the setting which had the lower LOOCV MSE and go back to step 2 else go to step 5.
5. Output the parameter setting which had the lowest MSE.

These procedures are a crude form of gradient descent.
Although they are not guaranteed to be optimal they are (somewhat) effective in practice.

5.1.6 Synthetic Data Set: Gong

The Gong function is a popular function for testing regression algorithms.
Interval: 0.5 to 1.5
Data Points: Sampled uniformly in the interval. (Default $= 800$)
Labeled Data Points: Sampled uniformly from the data points. (Default $= 80$).
RED = Estimated values
BLACK = Labeled examples
True function: $y = \frac{1}{x} \sin \frac{15}{x}$
$\sigma^2 = 0.1$ (Noise)

Figures 5.2, 5.3 and 5.4 show the results of running WKR, LLR and LLSR respectfully on this example. A table summarizing the results is given below in table 5.1. As can be seen LLSR performs substantially better than the other methods in its MSE and from the figures one can see that solution produced by LLSR is much smoother than the others.

Algorithm	MSE
WKR	25.67
LLR	14.39
LLSR	7.99

Table 5.1: Performance of different algorithms on the Gong dataset

Weighted kernel Regression

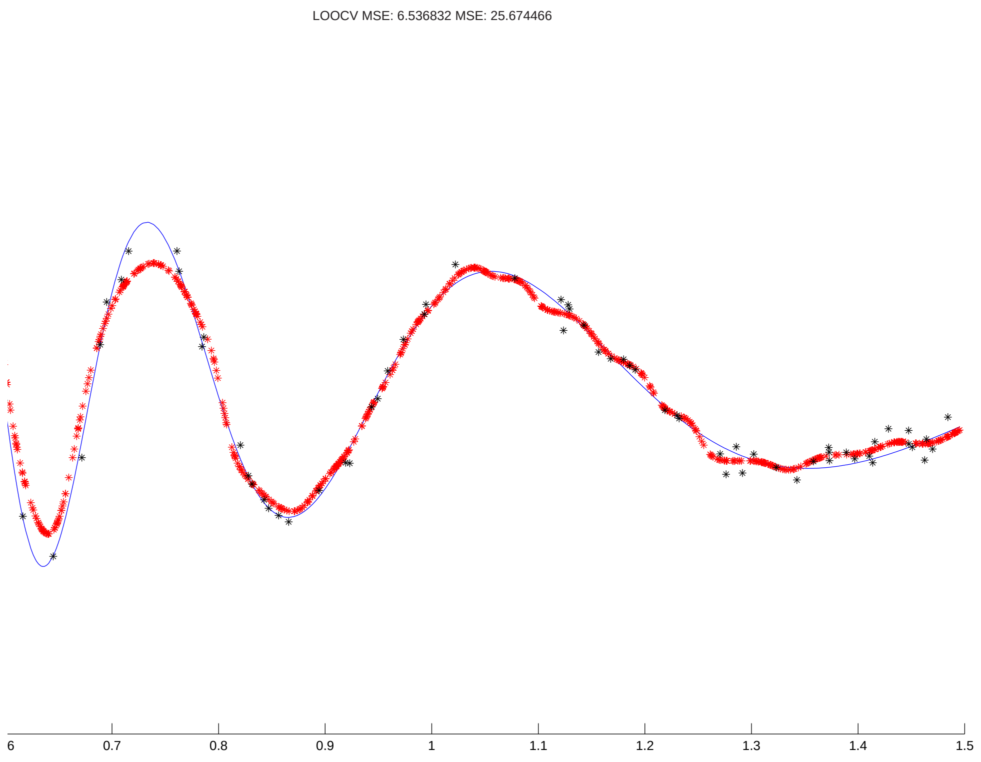

Figure 5.2: WKR on the Gong example, $h = \frac{1}{128}$

Discussion

There is significant bias on the left boundary and at the peaks and valleys. In this case it seems like a local linear assumption might be more appropriate.

Local Linear Regression

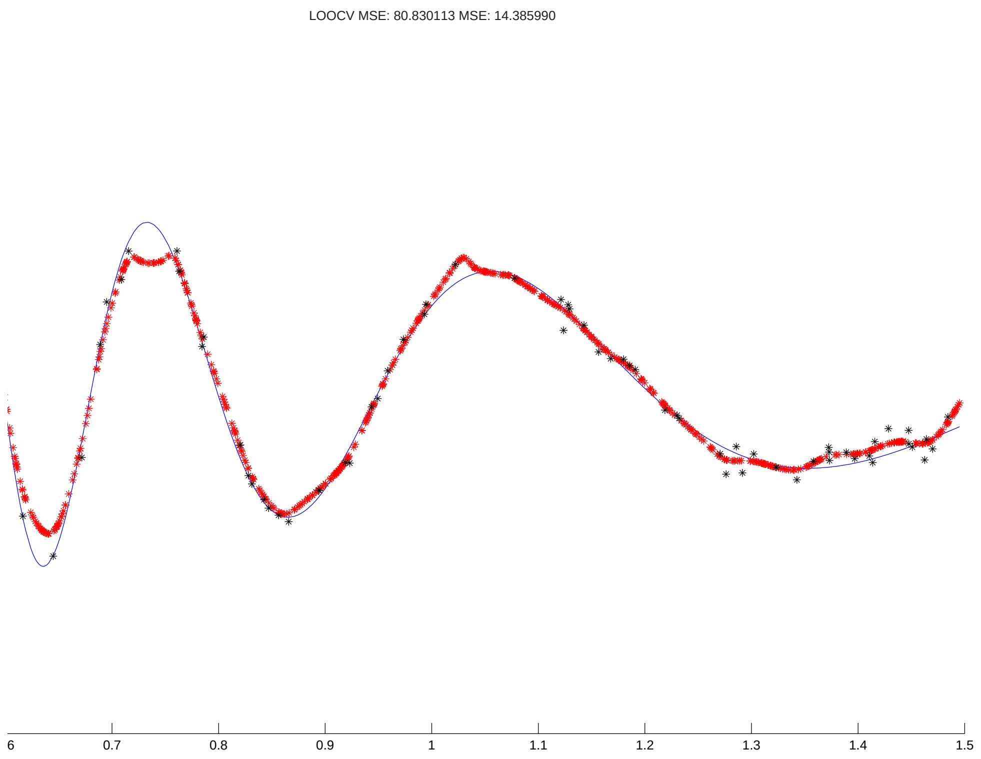

Figure 5.3: LLR on the Gong example, $h = \frac{1}{128}$

Discussion

There is less bias on the left boundary but there seems to be over fitting at the peaks. It seems like more smoothing is needed.

Local Linear Semi-supervised Regression

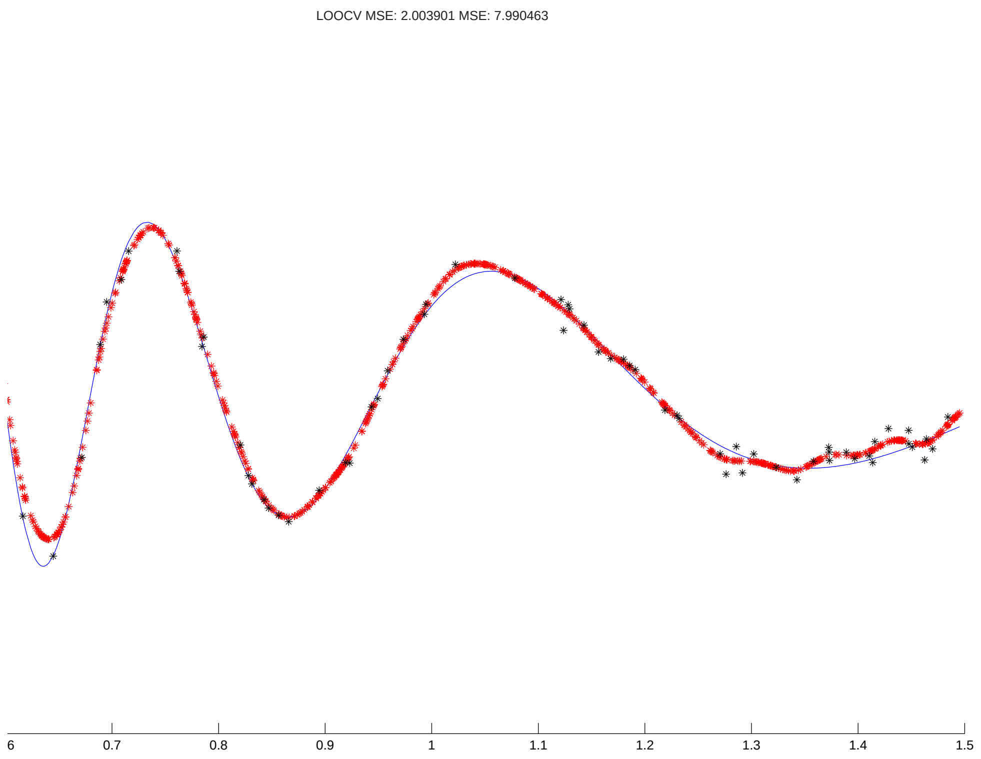

Figure 5.4: LLSR on the Gong example, $h = \frac{1}{512}, \gamma = 1$

Discussion

Although the fit is not perfect, it is the best of the 3. It manages to avoid boundary bias and fits most of the peaks and valleys.

5.1.7 Local Learning Regularization

Recently Schölkopf and Wu [63] have proposed Local Learning Regularization (LL-Reg) as a semi-supervised regression algorithm. They also propose a flexible framework that generalizes many of the well known semi-supervised learning algorithms.

Suppose we can cast our semi-supervised regression problem as finding the f that minimizes the following objective function:

$$f^T Rf + (f - y)^T C(f - y)$$

We can easily see that the f that minimizes this objective function is

$$f = (R + C)^{-1} Cy$$

The first term is the "semi-supervised" component and imposes some degree of "smoothness" on the predictions. The second term is the "supervised" part indicating the agreement of the predictions with the labeled examples. By choosing different matrices R and C we obtain different algorithms. Typically C is chosen to be the identity matrix so we focus on the choice of R.

It turns out that popular semi-supervised learning algorithms such as the harmonic algorithm [85] and NLap-Reg [81] can be cast in this framework with an appropriate choice of R. For example to get the harmonic algorithm of Zhu, Gharamani and Lafferty [85]we choose R to be the combinatorial graph Laplacian. To get the NLap-Reg algorithm of Zhou et al. [81] we choose R to be the normalized graph Laplacian.

Schölkopf and Wu [63] propose to use the following as the first term in the objective function

$$\sum (f_i - o_i(x_i))^2$$

where $o_i(x_i)$ is the local prediction for the value of x_i based on the value of its neighbors. Again we can choose different functions for the local predictor $o(x)$ and get correspondingly distinct algorithms.

Key point: If the local prediction at x_i is a **linear combination** of the value of its neighbors then we can write $\sum (f_i - o_i(x_i))^2$ as $f^T Rf$ for some suitable R.

To see this note that

$$\sum (f_i - o_i(x_i))^2 = ||f - o||^2$$

But if each prediction is a linear combination then $o = Af$ (for some matrix A) and

$$||f - o||^2 = ||f - Af||^2 = ||(I - A)f||^2 = f^T (I - A)^T (I - A)f$$

Hence $R = (I - A)^T (I - A)$.

So the only thing we have to do is pick the function $o_i(x_i)$ then the R will be fixed.

Schölkopf and Wu [63] propose using kernel ridge regression as the local predictor. This will tend to enforce a roughly linear relationship between the predictors. This makes LL-Reg a good candidate to compare against LLSR.

5.1.8 Further Experiments

To gain a better understanding of the performance of LLSR we also compare with (LL-Reg) in addition to WKR and LLR on some real world datasets. The number of examples(*n*), dimensions(*d*) and number of labeled examples(*nl*) in each dataset are indicated in tables 5.2 and 5.3.

Procedure

For each dataset we select a random labeled subset, select parameters using cross validation and compute the root mean squared error of the predictions on the unlabeled data. We repeat this 10 times and report the mean and standard deviation. We also report OPT, the root mean squared error of selecting the optimal parameters in the search space.

Model Selection

For model selection we do a grid search in the parameter space for the best Leave-One-Out Cross Validation rms error on the unlabeled data.

We also report the rms error for the optimal parameters within the range.

For LLSR we search over $\gamma \in \{\frac{1}{100}, \frac{1}{10}, 1, 10, 100\}, h \in \{\frac{1}{100}, \frac{1}{10}, 1, 10, 100\}$

For LL-Reg we search over $\lambda \in \{\frac{1}{100}, \frac{1}{10}, 1, 10, 100\}, h \in \{\frac{1}{100}, \frac{1}{10}, 1, 10, 100\}$

For WKR we search over $h \in \{\frac{1}{100}, \frac{1}{10}, 1, 10, 100\}$

For LLR we search over $h \in \{\frac{1}{100}, \frac{1}{10}, 1, 10, 100\}$

Results

Dataset	n	d	nl	LLSR	LLSR-OPT	WKR	WKR-OPT
Carbon	58	1	10	27±25	19±11	70±36	37±11
Alligators	25	1	10	288±176	209±162	336±210	324±211
Smoke	25	1	10	82±13	79±13	83±19	80±15
Autompg	392	7	100	50±2	49±1	57±3	57±3

Table 5.2: Performance of LLSR and WKR on some benchmark datasets

Dataset	n	d	nl	LLR	LLR-OPT	LL-Reg	LL-Reg-OPT
Carbon	58	1	10	57±16	54±10	162±199	74±22
Alligators	25	1	10	207±140	207±140	289±222	248±157
Smoke	25	1	10	82±12	80±13	82±14	70±6
Autompg	392	7	100	53±3	52±3	53±4	51±2

Table 5.3: Performance of LLR and LL-Reg on some benchmark datasets

6.1 Discussion

From these results combined with the synthetic experiments, LLSR seems to be most helpful on one dimensional datasets which have a “smooth” curve. The Carbon dataset happens to be of this type and LLSR performs particularly well on this dataset. On the other datasets performs competitively but not decisively better than the other algorithms. This is not surprising give the motivation behind the design of LLSR which was to smooth out the predictions, hence LLSR is likely to be more successful on datasets which meet this assumption.

7.1 Conclusion

We introduce Local Linear Semi-supervised Regression and show that it can be effective in taking advantage of unlabeled data. In particular, LLSR seems to perform somewhat better than WKR and LLR at fitting “peaks” and “valleys” where there are gaps in the labeled data. In general if the gaps between labeled data are not too big and the true function is “smooth” LLSR seems to achieve a lower true Mean Squared Error than the purely supervised algorithms.

Bibliography

[1] Rosa I. Arriaga and Santosh Vempala. Algorithmic theories of learning. In *Foundations of Computer Science*, 1999.

[2] M.-F. Balcan and A. Blum. On a theory of learning with similarity functions. *ICML06, 23rd International Conference on Machine Learning*, 2006.

[3] M.-F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins and low-dimensional mappings. *ALT04, 15th International Conference on Algorithmic Learning Theory,* pages 194—205.

[4] Maria-Florina Balcan and Avrim Blum. A pac-style model for learning from labeled and unlabeled data. In *In Proceedings of the 18th Annual Conference on Computational Learning Theory (COLT*, pages 111–126. COLT, 2005.

[5] M.F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, 2006.

[6] R. Bekkerman, A. McCallum, and G. Huang. categorization of email into folders: Benchmark experiments on enron and sri corpora,. Technical Report IR-418, University of Massachusetts,, 2004.

[7] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[8] G.M. Benedek and A. Itai. Learnability with respect to a fixed distribution. *Theoretical Computer Science*, 86:377—389, 1991.

[9] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 10*, pages 368—374. MIT Press, 1998.

[10] T. De Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16*, pages 73—80. MIT Press, 2004.

[11] T. De Bie and N. Cristianini. Convex transduction with the normalized cut. Technical Report 04-128, ESAT-SISTA, 2004.

[12] A. Blum. Empirical support for winnow and weighted majority algorithms: results on a calendar scheduling domain. *ICML*, 1995.

[13] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, pages 19—26. Morgan Kaufmann, 2001. 1.1.1

[14] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, July 1998.

[15] A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. *ICML04, 21st International Conference on Machine Learning*, 2004.

[16] Avrim Blum. Notes on machine learning theory: Margin bounds and luckiness functions. http://www.cs.cmu.edu/ avrim/ML08/lect0218.txt, 2008.

[17] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *IEEE Computer Vision and Pattern Recognition Conference*, June 1998.

[18] U. Brefeld, T. Gaertner, T. Scheffer, and S. Wrobel. Efficient co-regularized least squares regression. *ICML06, 23rd International Conference on Machine Learning*, 2006. 1.1.1

[19] A. Broder, R. Krauthgamer, and M. Mitzenmacher. Improved classification via connectivity information. In *Symposium on Discrete Algorithms*, January 2000.

[20] J. I. Brown, Carl A. Hickman, Alan D. Sokal, and David G. Wagner. Chromatic roots of generalized theta graphs. *J. Combinatorial Theory, Series B*, 83:272—297, 2001.

[21] Vitor R. Carvalho and William W. Cohen. Notes on single-pass online learning. Technical Report CMU-LTI-06-002, Carnegie Mellon University, 2006.

[22] Vitor R. Carvalho and William W. Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD 2006)*.

[23] V. Castelli and T.M. Cover. The relative value of labeled and unlabeled samples in pattern-recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2102—2117, November 1996.

[24] C.Cortes and M.Mohri. On transductive regression. In *Advances in Neural Information Processing Systems 18*. MIT Press, 2006. 1.1.1

[25] S. Chakrabarty, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998.

[26] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`. 1.1

[27] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[28] F.G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the Fifteenth Florida Artificial Intelligence Research Society Conference*, pages 327—331, 2002.

[29] I. Dagan, Y. Karov, and D. Roth. Mistake driven learning in text categorization. In *EMNLP*, pages 55—63, 1997.

[30] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical report, 1999.

[31] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1—38, 1977.

[32] Luc Devroye, Laszlo Györfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, 1997. ISBN 0387946187. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387946187`.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[34] M. Dyer, L. A. Goldberg, C. Greenhill, and M. Jerrum. On the relative complexity of approximate counting problems. In *Proceedings of APPROX'00, Lecture Notes in Computer Science 1913*, pages 108—119, 2000.

[35] Y. Freund, Y. Mansour, and R.E. Schapire. Generalization bounds for averaged classifiers (how to be a Bayesian without believing). To appear in Annals of Statistics. Preliminary version appeared in Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, 2001, 2003.

[36] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1048—1053, Edinburgh, Scotand, August 2005. URL `http://www.cs.technion.ac.il/~gabr/papers/fg-tc-ijcai05.pdf`.

[37] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271—279, 1989.

[38] Steve Hanneke. An analysis of graph cut size for transductive learning. In *the 23rd International Conference on Machine Learning*, 2006.

[39] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001. ISBN 0387952845. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387952845`.

[40] Thomas Hofmann. Text categorization with labeled and unlabeled data: A generative model approach. In *NIPS 99 Workshop on Using Unlabeled Data for Supervised Learning*, 1999.

[41] J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:550—554, 1994.

[42] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087—1116, 1993.

[43] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In D.S. Hochbaum, editor, *Approximation algorithms for* NP-*hard problems*. PWS Publishing, Boston, 1996.

[44] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the* 20^{th} *International Conference on Machine Learning (ICML)*, pages 290—297, 2003.

[45] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the* 16^{th} *International Conference on Machine Learning (ICML)*, 1999.

[46] Thorsten Joachims. *Making large-Scale SVM Learning Practical*. MIT Press, 1999.

[47] David Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of*

the ACM, 43(4), 1996.

[48] J. Kleinberg. Detecting a network failure. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 231—239, 2000.

[49] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *40th Annual Symposium on Foundations of Computer Science*, 2000.

[50] J. Kleinberg, M. Sandler, and A. Slivkins. Network failure detection and graph connectivity. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 76—85, 2004.

[51] Paul Komarek and Andrew Moore. Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity. Technical Report CMU-RI-TR-05-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2005.

[52] John Langford and John Shawe-Taylor. PAC-bayes and margins. In *Neural Information Processing Systems*, 2002.

[53] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 1988.

[54] D. McAllester. PAC-bayesian stochastic model selection. *Machine Learning*, 51(1):5—21, 2003.

[55] Ha Quang Minh, Partha Niyogi, and Yuan Yao. Mercer's theorem, feature maps, and smoothing. In *COLT*, pages 154–168, 2006.

[56] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[57] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.

[58] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380—393, April 1997.

[59] Joel Ratsaby and Santosh S. Venkatesh. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 412—417. ACM Press, New York, NY, 1995.

[60] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323—2326, 2000.

[61] Sebastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *International Conference on Computer Vision (ICCV'98)*, pages 492—499, January 1998.

[62] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

[63] Bernhard Schölkopf and Mingrui Wu. Transductive classification vis local learning regularization. In *AISTATS*, 2007. 5.1.7

[64] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

[65] John Shawe-Taylor and Nello Cristianini. *An introduction to support Vector Machines: and*

other kernel-based learning methods. Cambridge University Press, 1999.

[66] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44:1926–1940, 1998.

[67] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 731—737, 1997.

[68] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. *Proc. of the 22nd ICML Workshop on Learning with Multiple Views*, 2005. 1.1.1

[69] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.

[70] Nathan Srebro. personal communication, 2007.

[71] Josh Tenenbaum, Vin de Silva, and John Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[72] S. Thrun, T. Mitchell, and J. Cheng. The MONK's problems. a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991.

[73] UCI. Repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 2000.

[74] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[75] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[76] J.-P Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel methods in Computational Biology*, pages 131—154. MIT Press, Boston, 2004.

[77] Larry Wasserman. *All of Statistics : A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, 2004. ISBN 0387402721. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387402721`. 1.1.1

[78] Larry Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer, 2007. ISBN 0387251456. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387251456`. 1.1.1

[79] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:1101—1113, 1993.

[80] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. 17th International Conf. on Machine Learning*, pages 1191–1198, 2000.

[81] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*. MIT Press,

2004. 5.1.7

[82] Z.-H. Zhou and M. Li. Semi-supervised regression with co-training. *International Join Conference on Artificial Intelligence(IJCAI)*, 2005. 1.1.1

[83] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

[84] X. Zhu. Semi-supervised learning with graphs. 2005. Doctoral Dissertation. 2

[85] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912—919, 2003. 1.1.1, 1.1.2, 2, 5.1.7