An Iterative Algorithm for Extending Learners to a Semi-supervised Setting

Mark Culp George Michailidis

October 17, 2007

Abstract

In this paper, we present an iterative self-training algorithm, whose objective is to extend learners from a supervised setting into a semi-supervised setting. The algorithm is based on using the *predicted* values for observations where the response is missing (unlabeled data) and then incorporates the predictions appropriately at subsequent stages. Convergence properties of the algorithm are investigated for particular learners, such as linear/logistic regression and linear smoothers with particular emphasis on kernel smoothers. Further, implementation issues of the algorithm with other learners such as generalized additive models, tree partitioning methods, partial least squares, etc. are also addressed. The connection between the proposed algorithm and graph-based semi-supervised learning methods is also discussed. The algorithm is illustrated on a number of real data sets using a varying degree of labeled responses.

Keywords: Semi-supervised learning, linear smoothers, convergence, iterative algorithm

1 Introduction and Problem Formulation

In the *supervised learning* problem, the objective is to construct a rule (learner) based on training data that would be capable of predicting accurately the response of future observations. Specifically, let $[Y|X]$ be a $n \times (p+1)$ data matrix, with either a numerical or categorical response variable Y, and p explanatory variables X. Over the years, several approaches have been proposed in the literature to address the problem of constructing a learner $\phi(X, Y)$ from the available *training data* for predicting the response of any observation. Examples of such learners include various parametric and non-parametric

regression techniques for numerical responses and a number of classification algorithms for categorical ones (see Ripley (1996), Duda et al. (2000), and Hastie et al. (2001)).

In many application areas, it is fairly simple and inexpensive to collect data on the p explanatory variables. However, it is more challenging and expensive to observe the response Y for a large number of observations. Such is the case with the pharmacology data set examined in Section 4. In such a setting, data have been collected for p explanatory variables on n objects (cases), as before. But responses are available only for a subset of size m of the objects; therefore, the data can be partitioned into two subsets: the first set, of size $m \times (p+1)$ contains labeled data $[Y_L|X_L]$, while the second set of size $(n-m) \times (p+1)$ contains unlabeled data $[Y_U|X_U]$. In particular, the response vector Y_U is missing. This gives rise to a *semi-supervised learning* framework. The task at hand is to use the information available in the labeled data, in conjunction with their relationship to unlabeled data, to better predict the information in X_U and potentially better predict any possible observation x. The problem of semi-supervised learning has been studied from various perspectives, with emphasis on how the algorithm actually uses the unlabeled information, and whether the proposed technique is *transductive*, that is, it can only complete the labeling of the missing $n - m$ responses, or *inductive*, in that it can predict the labels of any potential observation (Krishnapuram et al., 2005). The survey by Zhu (2006) and the book by Chapelle et al. (2006b) provide a nice overview of the field. Next, we provide a brief discussion highlighting some of the general problems addressed in the area of semi-supervised learning.

One class of techniques is based on the following *self-training* idea. The most confident predictions of unlabeled responses are successively added to the labeled set, and used in subsequent rounds for training the learner to predict less confident unlabeled observations. This approach tends to work best when the number of unlabeled observations considered at each round is randomly selected, and the number of observations added to the set is small relative to the size of the unlabeled data (Blum and Mitchell, 1998). Given the fact that in this context unlabeled (missing) response information is processed, most such approaches are based on an EM algorithmic implementation, with fairly strong assumptions made on the densities of the unlabeled data. These techniques have proved successful in a number of application areas, including text classification and word sense disambiguation problems (Abney, 2004), image processing (Baluja, 1999), and face recognition (Nigam et al., 2000). Another technique for extending supervised borders into semi-supervised learning is the transductive support vector machine (TSVM) (Vapnik, 1998). The main idea is to optimally push the border determined by a supervised margin-based procedure away from highly dense regions of labeled and unlabeled data, which should minimize the generalization error. The main challenge for this problem stems from the fact that the TSVM solution is NP-hard. As a result, the implementation requires approximations to perform this task, which typi-

cally involve solving computationally intensive non-convex loss/penalty optimization problems (Vapnik, 1998; Chapelle et al., 2006a). The authors in Chapelle et al. (2006a) provide compelling quantitative evidence that the global optimum of TSVM is quite accurate on small data sets.

Graph-based algorithms provide another class of semi-supervised procedures. The main idea is that the data are represented by a graph, whose vertices are both labeled and unlabeled observations and the weighted edges represent pairwise similarity between them. Various graph-based algorithmic solutions for classifying the unlabeled vertices in the graph proposed in the literature include regularized kernels, harmonic fields, label propagation to name a few (Blum and Chawla, 2001; Zhu and Ghahramani, 2002; Zhu et al., 2003; Culp and Michailidis, 2007). These procedures are viewed as graph cutting algorithms stemming from st-minicut (Blum and Chawla, 2001), where the technique cuts edges on the graph to classify the unlabeled vertices. To extend st-minicut, several of these approaches draw on motivation from spectral graph theory, electrical circuit design theory, diffusion processes, and harmonic energy minimization (Blum and Chawla, 2001; Kondor and Lafferty, 2002; Zhu et al., 2003). For example, in Zhu et al. (2003) the authors present a soft version of st-minicut, where the quadratic harmonic energy function is optimized with the constraint that the labeled data are forced to agree entirely with the observed true values (i.e. *clamping* constraint). In addition, for the special case when the graph is fully connected, the authors provide a connection between harmonic optimization and iterative label propagation algorithms (Zhu and Ghahramani, 2002; Zhu, 2005).

It is worth pointing out that the theoretical foundations of the usefulness of unlabeled data in semi-supervised learning are not well established. The best understood setting is when the available data come from a parametric mixture distribution with two known components, but unknown mixing parameter (Castelli and Cover, 1996). It is shown that the relative value of labeled and unlabeled data for reducing the risk of the classifier under consideration is determined by the ratio of the Fisher information matrices for the mixing parameter. In non-parametric settings, a similar result about the contribution of unlabeled data has not been established. The "cluster assumption" that posits that two data points are more likely to have the same label (in a classification setting) if they are close together in a high density region of the feature space has been suggested as a basis of analysis and underlies many of the approaches proposed in the literature (e.g. TSVM and graph cutting algorithms). The cluster assumption plays an important role in graph-based learning, where the problem is treated as inherently transductive. In this setting, the distance between two labeled vertices depends on the unlabeled vertices between them, which shows the strong influence exhibited by unlabeled data through the underlying topology of the graph (Culp and Michailidis, 2007). The paper by Johnson and Zhang (2007) addresses the usefulness of unlabeled data in graph-based inductive learning, with the emphasis placed on prediction of new unlabeled vertices

unavailable during training. Several of these results are consistent with the findings in this work (see Corollary 1 and the ensuing discussion).

In this work, an iterative algorithm called Fitting the Fits (FTF) is introduced as a general scheme which can extend supervised learners into the semi-supervised setting. Figure 1 provides a diagram categorizing the general learners considered in this work in terms of convergence properties and performance. The main feature is that kernel smoothers, graph-based smoothers (see below), loess smoothers (Hastie and Tibshirani, 1990) and generalized additive models with loess smoothing (Hastie and Tibshirani, 1990; Hastie, 2006) provide the most gain in performance in the presence of unlabeled data; smoothing spline techniques (Hastie and Tibshirani, 1990; Hastie et al., 2001) and classification and regression trees (Breiman et al., 1984) show some ability to improve performance with unlabeled data; finally, linear regression and partial least squares procedures (Wehrens and Mevik, 2006) show little to no improvement with unlabeled data. Convergence properties of the proposed algorithm are studied rigorously for a number of these learners. It is shown that associations between labeled and unlabeled data are necessary for convergence, although the nature of the learner (global parametric model vs local non-parametric one) plays an important role. This finding is consistent in spirit with the results discussed in Zhang and Oles (2000).

The proposed procedure has much in common with self-training algorithms previously discussed. One significant difference is that the FTF algorithm uses all the available data in each round, as opposed to only the unlabeled cases that have been predicted with high confidence in previous rounds. In addition to studying the FTF algorithm as a self-training procedure, we also briefly discuss its connection to graph-based label propagation algorithms. In this case, the algorithm can be viewed as propagating labels across an adjacency matrix of a weighted graph in a similar manner as that of the label propagation algorithm discussed in Zhu and Ghahramani (2002). One subtle, but important difference in this context between the FTF algorithm and label propagation is that the former does not require clamping the response upon convergence. Hence, the estimates of the originally labeled responses yield residuals and a complexity measure (degrees of freedom) for the graph setting that can be useful for selecting various tuning parameters associated with the underlying learner. Interestingly, the relationship between self-training and graph-based learning to our knowledge has not been fully established.

The paper is organized as follows: The proposed algorithm is introduced in Section 2, where its properties for specific learners are also established. Operational issues for other learners, together with applications are presented in Section 3. Some concluding remarks are drawn in Section 4, while the proofs of the technical results are given in the Appendix.

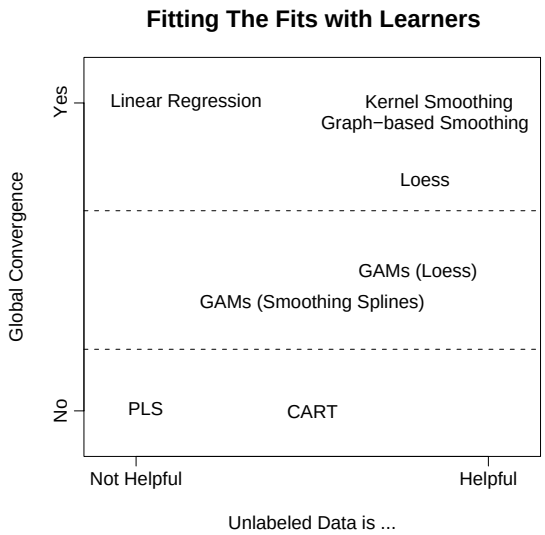

Figure 1: The diagram provides a layout of different learners considered in this work for the Fitting The Fits algorithm. The x-axis indicates the degree to which unlabeled data are helpful for this procedure. The learners are separated into three regions, in terms of their global convergence property. Specifically, the top region contains learners for which global convergence is established independent of the initialization and a closed form expression is obtained, the second region indicates that general convergence criteria are obtained and a closed form solution exists, and the third group contains learners where only local convergence is established empirically and a close form solution has not been obtained.

2 The Algorithm

Let $[Y|X]$ denote the available data that are partitioned into two sets $[Y_L|X_L]$ and $[Y_U|X_U]$, with the response Y_U missing. Further, let $\phi_\lambda(Y, X)$ denote the supervised learner of choice, where (Y, X) correspond to the data that the learner was trained on and λ is a tuning parameter. At each iteration k the algorithm determines a predicted response $\hat{Y}^k = [\hat{Y}_L^k \;\; \hat{Y}_U^k]^T$ as follows.

Algorithm 1 Fitting The Fits (FTF)

Initialize vectors of size n: $\hat{Y}^0$ and $\hat{Y}^1$, and set $k = 1$.
while $\| \hat{Y}_U^k - \hat{Y}_U^{k-1} \|_2^2 > \delta$ **do**
    $\hat{Y}_L^k = Y_L$ (reset the predicted response to the true response)
    Fit $\hat{Y}^{k+1} = \phi_\lambda(\hat{Y}^k, X)$, with ϕ_λ being a supervised learner with tuning parameter λ
end while

The main steps of the algorithm, upon selecting the supervised base learner and initializing the unlabeled responses, are outlined next: (i) set the labeled response estimates to their true values, and (ii) predict both the labeled and unlabeled response vectors with the chosen learner. The procedure is repeated until the predicted responses in two successive iterations exhibit a change below a prespecified

threshold (e.g. $\delta = 1e-6$ proved sufficient for the examples presented below). The procedure is summarized as Algorithm 1.

The intent of the algorithm is to extend supervised learners to the semi-supervised setting, while simultaneously retaining their main characteristics. For example, if the underlying learner is a generalized additive model, then one can still examine issues such as variable importance and presence of outliers or observations' influence for the converged semi-supervised solution.

The following issues arise regarding this algorithm: (i) what learners are appropriate for the algorithm, (ii) under what conditions the algorithm converges, and (iii) what loss functions are suitable for fitting the learners. Several learners are investigated in this study and for many of them these issues are rigorously addressed and resolved.

2.1 Operational Characteristics of the Algorithm for Linear Smoothers

We start by providing a brief description of the supervised learners considered in this work. With a real valued response Y_L, one is usually interested in fitting the following model

$$y = g(x_1, \ldots, x_p) + \epsilon \text{ where } \epsilon \sim N(0, I\sigma^2). \tag{1}$$

Under a linearity assumption, the model takes the form $g(x_1, \ldots, x_p) = X_L\beta$ and the goal is to estimate the parameter vector β from the labeled data. The solution for this problem, under a squared error loss criterion, is given by $\widetilde{\beta}_{ls} = (X_L^T X_L)^{-1} X_L^T Y_L$ (tilde indicates supervised). The procedure orthogonally projects the response vector onto the space spanned by the X_L data with $H_L = X_L(X_L^T X_L)^{-1} X_L^T$. Hence, the predicted values are given by $\widetilde{Y}_L = X_L\widetilde{\beta}_{ls} = H_L Y_L$ for the labeled data and $\widetilde{Y}_U = X_U\widetilde{\beta}_{ls} = H_U Y_L$ for the unlabeled data with $H_U = X_U(X_L^T X_L)^{-1} X_L^T$.

A more flexible model that relaxes the linearity assumption is to use a linear smoother S_λ that may depend on a tuning parameter λ (i.e. $\widetilde{Y}_L = S_{L_\lambda} Y_L$) (Buja et al., 1989). For example, in the case of supervised kernel smoothers the solution $\widetilde{Y}_j$ of the following optimization problem

$$\min_{\widetilde{Y}_i \in \mathbb{R}} \sum_{i \in L} w_{ij}(Y_i - \widetilde{Y}_j)^2, \tag{2}$$

is linear in Y_L with $w_{ij} = K_\lambda(x_i, x_j)$ for some kernel function that depends on a tuning parameter λ. Notice that the smoother resulting from the above problem is explicitly defined as $S_L(i,j) = \frac{K_\lambda(x_i,x_j)}{\sum_{k \in L} K_\lambda(x_i,x_k)} \leq 1$ for $i, j \in L$ which results in $\widetilde{Y}_L = S_L Y_L$. Also, for unlabeled observations the

smoother is given by: $S_U(i,j) = \frac{K_\lambda(x_i,x_j)}{\sum_{k\in L} K_\lambda(x_i,x_k)}$ for $i \in U$ and $j \in L$. In general, to connect (1) with linear smoothers the framework requires model assumptions and modifications such as regularization (smoothing splines), locality (kernel or loess smoother) or additivity (additive models) (Buja et al., 1989; Hastie and Tibshirani, 1990). This leads to both unique estimation of the smoother and the resulting linear solution, $\widetilde{Y}_L = S_L Y_L$ and $\widetilde{Y}_U = S_U Y_L$. To estimate the smoothing parameter λ for a linear smoother, the generalized cross-validation (GCV) index proves useful, since it is a convenient approximation to leave-one-out cross-validation (Hastie and Tibshirani, 1990; Hastie et al., 2001). Specifically, it is given by GCV $= \frac{\epsilon_L^T \epsilon_L}{(1-df/m)^2}$ with $\epsilon_L = Y_L - S_{L\lambda} Y_L$ and the effective degrees of freedom given by $df = tr(S_L)$. The minimizer of GCV yields the optimal value of λ.

Remark: In a binary classification setting ($Y_i \in \{0,1\}$) optimization of a squared error loss function may not be particularly appropriate. The reason for this is that a symmetric loss is incurred for correctly classifying an observation as would be for incorrectly classifying it. A more appropriate loss function is the logistic one given by ($L(g,f) = \log(1+\exp(-g(y)f)$ with $g(y) = 2y-1$). The model given in (1) requires a transformation by an appropriate link function, η, e.g.

$$\eta(y) = g(x_1,\ldots,x_p) + \epsilon \text{ for } \eta(y) = \log\left(\frac{1-y}{y}\right). \tag{3}$$

In general the link function function also generalizes to other settings, such as Poisson ($\eta = \log(y)$) and Gamma ($\eta = y^{-1}$) regression (Duda et al., 2000).

2.1.1 Convergence for Linear Smoothers

We show next that the algorithm converges if the base learner corresponds to a linear smoother. At iteration k the algorithm fits

$$\begin{pmatrix} \hat{Y}_L^{k+1} \\ \hat{Y}_U^{k+1} \end{pmatrix} = \phi_\lambda\left(\begin{pmatrix} Y_L \\ \hat{Y}_U^k \end{pmatrix}, \begin{pmatrix} X_L \\ X_U \end{pmatrix}\right) = \begin{pmatrix} S_{LL} & S_{LU} \\ S_{UL} & S_{UU} \end{pmatrix} \begin{pmatrix} Y_L \\ \hat{Y}_U^k \end{pmatrix}, \tag{4}$$

Notice that Step 4 of Algorithm 1 can be written as $\hat{Y}^{k+1} = S\hat{Y}^k$ with $\hat{Y}^k = [Y_L \quad \hat{Y}_U^k]^T$, where $S = \begin{pmatrix} S_{LL} & S_{LU} \\ S_{UL} & S_{UU} \end{pmatrix}$ is an arbitrary linear smoother based on the entire X data. Some algebra shows that at the $k-$th iteration of the algorithm we obtain

$$\hat{Y}_U^k = (I + S_{UU} + S_{UU}^2 + \cdots + S_{UU}^{k-1}) S_{UL} Y_L + S_{UU}^k \hat{Y}_U^0, \tag{5}$$

where $\hat{Y}_U^0$ corresponds to the initialization of the unlabeled response. It is easy to see that if the spectral radius $\rho(S_{UU}) < 1$, then the algorithm converges. This is formalized in the following proposition for a class of linear smoothers (proof given in the Appendix).

Proposition 1. *Let S be a symmetric stochastic matrix (i.e. $S\mathbf{1} = \mathbf{1}$, $S \geq 0$) and S_{UU} be an irreducible and substochastic matrix. Then, the FTF algorithm converges independent of the initialization, and the solution is given by*

$$\hat{Y}_U = (I - S_{UU})^{-1} S_{UL} Y_L,$$

provided S_{UU} is of full rank.

The estimated solution for both the labeled and unlabeled response is given by

$$\begin{pmatrix} \hat{Y}_L \\ \hat{Y}_U \end{pmatrix} = \begin{pmatrix} S_{LL} + S_{LU}(I - S_{UU})^{-1} S_{UL} \\ (I - S_{UU})^{-1} S_{UL} \end{pmatrix} Y_L. \tag{6}$$

The labeled solution, $\hat{Y}_L$, is linear in Y_L; therefore, as with the supervised case, we can employ GCV for estimation of the smoothing parameter. However, in the semi-supervised context the components of GCV are now given by $\epsilon = Y_L - (S_{LL} + S_{LU}(I - S_{UU})^{-1} S_{UL}) Y_L$ and $df = tr(S_{LL} + S_{LU}(I - S_{UU})^{-1} S_{UL})$. Notice that the unlabeled data influence this estimation procedure.

The main computational issues for fitting linear smoothers using the FTF algorithm are calculating the inverse in (6) and estimating internal smoother parameter λ (or λ denotes the vector of smoothing parameters with additive models). The former becomes rather demanding when the size of the unlabeled data set becomes particularly large. However, in our experience if the underlying smoother matrices are fairly sparse, then the complexity of the algorithm significantly improves.

2.1.2 Kernel Smoothers with the FTF Procedure

The result in Proposition 1 provides a sufficient condition on the smoother sub-matrix S_{UU} for the algorithm to be globally convergent. However, the structure of the learner ϕ determines whether this condition is satisfied or not. In the case where the linear smoother is derived from a kernel, then a more informative result can be established. The idea behind kernel smoothing is that observations close together in the X space should exhibit similar responses.

A smoother matrix S obtained from a kernel has the form $S = D^{-1}W$ with $W_{ij} = K_\lambda(x_i, x_j)$ for

$i, j \in L \cup U$ and $D =$diag$(W \times \mathbf{1}_{|L \cup U|})$ (here $\mathbf{1}_r$ is a vector of ones of length r). The FTF algorithm with kernel smoothing is then given by (i) $\hat{Y}_L^k = Y_L$ and (ii) $\hat{Y}^{k+1} = S\hat{Y}^k$. From this we have that the closed form solution in (6) for the FTF algorithm with linear smoothers applies to the kernel smoother case. To show convergence of the algorithm, notice that the kernel matrix W also emits the following partition with respect to labeled and unlabeled data:

$$W = \begin{pmatrix} W_{LL} & W_{LU} \\ W_{UL} & W_{UU} \end{pmatrix}. \tag{7}$$

Next, we establish the following proposition (proof given in the Appendix):

Proposition 2. *Let $S = D^{-1}W$ where W emits the partition given in (2.3). If $W_{UL}\mathbf{1}_{|L|} > \vec{0}$ and W_{UU} is irreducible, then the FTF algorithm converges independent of its initialization, since $\rho(S_{UU}) < 1$.*

Notice that the matrix W_{UL} plays a crucial role in assessing whether or not the procedure converges. The intuition is that the connections between labeled and unlabeled data provide information necessary for the unlabeled data to be informative.

To compare directly the supervised and the semi-supervised results we provide a simple discussion illustrating how each solution uses the information in the W matrix differently. The supervised kernel smoother is given by:

$$\widetilde{Y}_L = D_{LL}^{-1} W_{LL} Y_L = S_L Y_L \text{ and } \widetilde{Y}_U = D_{UL}^{-1} W_{UL} Y_L = S_U Y_L,$$

with the diagonal row sum matrices appropriately defined[1]. It can be seen that the supervised estimate for the labeled data relies only on the W_{LL} partition of W, while that for the unlabeled data only on the W_{UL}. However in the semi-supervised case (6) the estimate for the labeled and unlabeled data employs all the partitions of the W matrix; namely, W_{LL}, W_{UL}, W_{UL}^T, and W_{UU}. Further, the semi-supervised kernel smoother estimate can be equivalently written as a linear operator of the supervised kernel smoother given by: $\hat{Y}_U = (I - S_{UU})^{-1}(I - D_U^{-1} D_{UU})\widetilde{Y}_U$ and
$\hat{Y}_L = D_L^{-1} D_{LL} \widetilde{Y}_L + (I - D_L^{-1} D_{LL}) W_{UL}^T D_{LU} (I - S_{UU})^{-1} (I - D_U^{-1} D_{UU}) \widetilde{Y}_U$. This result can be particularly useful in comparing several aspects of supervised kernel smoothing to semi-supervised kernel smoothing, such as bias, variance, residual sum of squares, etc. One result of particular interest is the roles of W_{UL} and W_{UL}^T (necessary for $S_U, D_{LL}, D_{UU}, D_{UL}$, and D_{LU}) in the above decomposition. The diagonal matrix inequality $D_L^{-1} D_{LL} < I$ is satisfied if $W_{UL}^T \mathbf{1}_{|U|} > \vec{0}$, and similarly $D_U^{-1} D_{UU} < I$ is

[1]For clarity purposes the row sum matrices are defined next: D_{LL} =diag$(W_{LL} \times \mathbf{1}_{|L|})$, D_{LU} =diag$(W_{LU} \times \mathbf{1}_{|U|})$, D_{LU} =diag$(W_{UL} \times \mathbf{1}_{|L|})$, D_{UU} =diag$(W_{UU} \times \mathbf{1}_{|U|})$, $D_L = D_{LL} + D_{LU}$, and $D_U = D_{UU} + D_{UL}$.

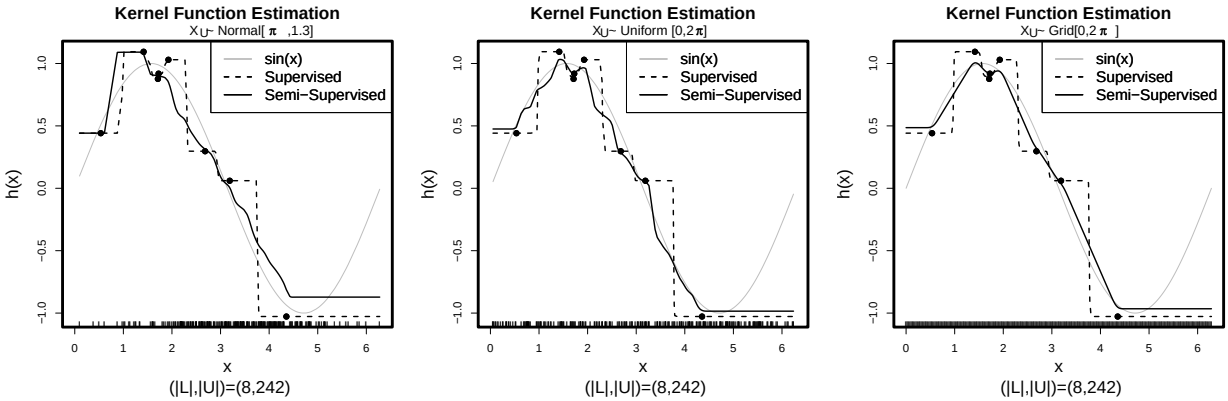

Figure 2: The figure provides three examples of semi-supervised kernel smoothing where the unlabeled data is (left) $N(\pi, 1.3)$, (middle) $U[0, 2\pi]$, and (right) a grid from 0 to 2π with step$= \frac{2\pi}{|U|}$. In all instances the same 8 labeled observations are used with $\mid U \mid= 242$. The plots also contain the supervised and true function.

satisfied whenever $W_{UL} \times \mathbf{1}_{|L|} > \vec{0}$ (or FTF converges). As a consequence, the semi-supervised result is an average of the supervised kernel smoother's labeled estimate and a matrix corrected version of the supervised unlabeled estimate. The extent to which the results are similar depends on the information in W_{UL} and W_{UU} as described by the above inequalities.

We consider next the following simple example to illustrate the FTF procedure with kernel smoothing. The univariate model considered is $Y = \sin(X) + \epsilon$, $m = 8$ (labeled data size), and $n = 250$ (total observations). For this example, we considered three possible scenarios of obtaining the unlabeled 242 observations using the same 8 labeled cases. The goal is to illustrate that the semi-supervised kernel smoother can provide (depending on the unlabeled data) a visually pleasing representation of the function. In the first scenario, $X_U \sim N(\pi, 1.3)$, in the second one $X_U \sim U[0, 2\pi]$, while for the third one X_U corresponds to a uniform design (i.e. an evenly spaced grid) on $[0, 2\pi]$. The uniform grid is viewed as defining a 'binned' unlabeled support for kernel smoothing. In figure 2, the three unlabeled examples are provided. Notice that the unlabeled data provide significant additional smoothing in all three cases. In the case of Gaussian generation of the X_U variable, the smooth is rather close to the true function near $\mu = \pi$ and the accuracy decreases for the cases farther away from the center of the distribution. An increase in variance on the unlabeled data generator results in fewer unlabeled observations being located close to π, but also in more observations outside the boundaries of the labeled data. The chosen value $\sigma = 1.3$ worked fairly well for this example. When the X_U data were generated according to the uniform distribution, a more accurate smooth was obtained as expected (X_L was generated from the same distribution). Finally, the uniform design yields the most visually pleasing smooth for this synthetic data

set, since it appears to capture particularly well the shape of the true underlying function. These results provide evidence that the semi-supervised approach leads to additional smoothing in non-parametric function estimation problems.

Local regression techniques (loess) provide an approach to improve upon kernel smoothing by reducing the overall variance of the fit obtained by the latter procedure. We briefly discuss using the FTF algorithm with loess as the base learner. Specifically, for a response $\hat{Y}^k = [Y_L \quad \hat{Y}_U^k]^T$, consider the following problem for determining $\hat{\beta}_0^k(x_j)$ and $\hat{\beta}_1^k(x_j)$ for the $j-$th observation:

$$\min_{\beta_0(x_j),\beta_1(x_j)} \sum_{i \in L \cup U} K_\lambda(x_i, x_j)(\hat{Y}_i^k - \beta_0(x_j) + \beta_1(x_j)x_i)^2. \tag{8}$$

The new estimate is $\hat{Y}_j^{k+1} = \hat{\beta}_0^k(x_j) + \hat{\beta}_1(x_j)^k x_j$. To obtain a smoother matrix corresponding to this solution, we extend the result in Hastie et al. (2001, Chapter 6) to the semi-supervised setting. Define $b(x)^T = (1, x)$ and let B be a $\mid L \cup U \mid \times 2$ matrix with each row given by $b(x_i)^T$; further, define a sequence of diagonal matrices $\{W(x_j)_{ii} = K_\lambda(x_i, x_j)\}_{j=1}^{L \cup U}$ with $i \in L \cup U$. From this set-up we have that $\hat{Y}_j^{k+1} = b(x_j)^T(B^T W(x_j) B)^{-1} B^T W(x_j)\hat{Y}^k \equiv \sum_{i \in L \cup U} \ell_i(x_j)\hat{Y}_i^k$. Therefore, the smoother matrix has elements $S_{ij} = \ell_i(x_j)$ with $i, j \in L \cup U$. One can therefore equivalently write the FTF algorithm for a loess smoother as (i) $\hat{Y}_L^k = Y_L$ and (ii) $\hat{Y}^{k+1} = S\hat{Y}^k$ and the closed form solution in (6) applies. Conditions on the X data so that $\rho(S_{UU}) < 1$ (i.e. which would imply global convergence), comparison to the supervised case, and a generalization to higher order polynomials in X are currently under investigation.

Next, we study the connection between the self-training FTF algorithm using kernel smoothing and the work on label propagation algorithms in Zhu and Ghahramani (2002). In this context, FTF is a method for graph-based semi-supervised learning (Zhu, 2005; Culp and Michailidis, 2007). Let $G = (V, E)$ denote a graph, with vertex (node) set V corresponding to the observations and $E = \{e_{ij}\},\ e_{ij} \geq 0$ the set of weighted edges that capture the degree of similarity between observations (nodes). The graph can be represented in matrix form by its adjacency matrix $A = \{A_{ij}\}$; notice it emits a partition analogous to that in (2.3). Further, suppose that the response is categorical. The goal in semi-supervised graph-based learning is to assign some category to the unlabeled vertices. The FTF algorithm uses the smoother $S = D^{-1}A$, where D is a diagonal matrix whose elements contain the row sums of the adjacency matrix A. Its steps are: (i) $\hat{Y}_L^k = Y_L$ and (ii) $\hat{Y}^{k+1} = S\hat{Y}^k$. The labeled and unlabeled estimates for this procedure are given in (6). Convergence is established by the following corollary to Proposition 2.

Corollary 1. *Let $S = D^{-1}A$ where A is the graph adjacency matrix and emits the partition given in*

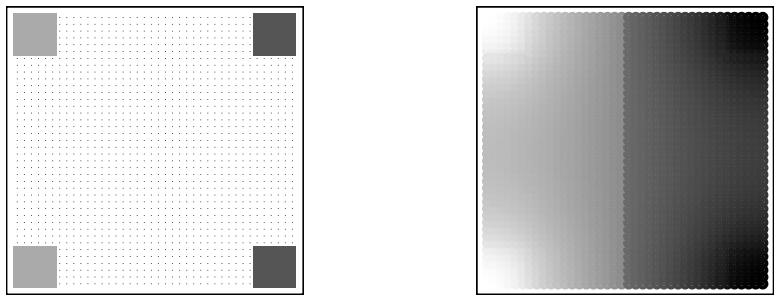

Figure 3: This plot illustrates the FTF algorithm in a graph-based setting. The graph is given by a 40×40 node lattice, with edges (not shown) connecting them along all cardinal directions. The labeled data belong to two classes ('gray' and 'black') and are positioned in the four corners (see left panel). The estimates obtained from the FTF algorithm over the entire lattice are shown in the right panel. At each point on the lattice, a prediction ('gray' or 'black') is made, where the intensity of the pixel at each point represents the probability that the point has been classified as 'black'. The more intense the pixel, the larger the probability that the case is estimated as 'black'.

(2.3). If $A_{UL}\mathbf{1}_{|L|} > \vec{0}$ and A_{UU} is irreducible, then the FTF algorithm converges independent of its initialization, since $\rho(S_{UU}) < 1$.

The condition $A_{UL}\mathbf{1}_{|L|} > \vec{0}$ requires that each unlabeled node has at least one edge to a labeled one. Notice also that this result applies to situations where the graph may not be fully connected, which generalizes quite nicely the convergence result based on a similar strategy given in Zhu and Ghahramani (2002). This result provides a connection between graph-based learning with label propagation algorithms and self-training with the FTF procedure introduced in this work.

An illustration of the FTF algorithm in the graph-based setting is shown next. Consider a lattice with 1600 nodes in total, out of which 144 carry 0/1 labels. Each edge is connected to its immediate cardinal neighbors and the resulting graph with its labeled partition is presented in the left panel of Figure 3. In the presence of many unlabeled nodes that are not connected to any labeled ones, we have to modify our strategy in order to make the FTF algorithm applicable. A dissimilarity matrix Δ was obtained, whose elements $\delta_{ij} \geq 0$ correspond to the length of the shortest path from node i to node j. Then, the heat kernel function $K_\lambda(v_i, v_j) = \exp\left(\frac{\|v_i - v_j\|_1}{\lambda}\right)$ was used to turn the dissimilarities to a similarity weight matrix W, with the parameter λ estimated through GCV. The result in terms of probability class estimates is shown in the right panel of Figure 3. The probability class estimates reflect the values of the labeled data in a fairly intuitive way. Observations close to labeled cases have higher probability for that class, while towards the center the confidence decreases.

Remark: In this example, a dissimilarity matrix based on the shortest path metric had to be calculated -which can be achieved by a shortest path algorithm similar to Floyd's (Floyd, 1962)- and subsequently

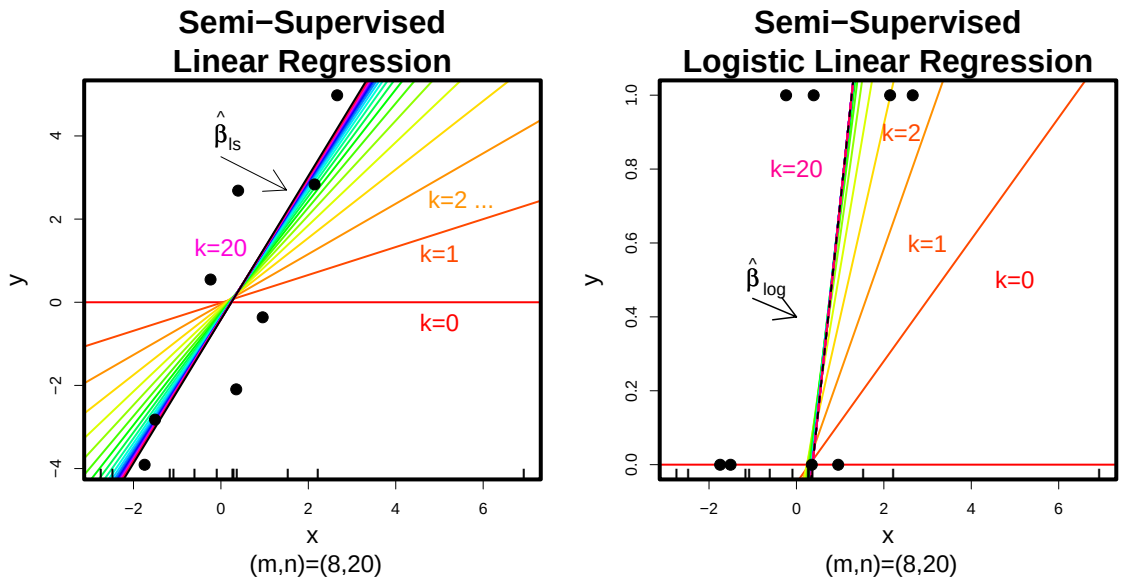

Figure 4: This plot depicts the semi-supervised linear and logistic regression results obtained from the FTF algorithm (left and right panel, respectively). The horizontal line ($k = 0$) represents the initialization of the FTF procedure. The various lines shown correspond to the following k values: $k = 1$ to $k = 20$. Note that the unlabeled data at the k-th iteration are set to the fitted values obtained in the previous iteration. The data size was set to $m = 8$ and $n = 20$. The tics on the bottom of the plot indicate the unlabeled observations.

a kernel had to be used. The first step has $O(\mid L \cup U \mid^2)$ complexity, while the second involves the estimation of the smoothing parameter. Nevertheless, once the matrix W is obtained the result of Corollary 2 applies.

Remark: The conditions specified in Proposition 2 and Corollary 1 have practical significance for attaining both computational and performance gains. For example, the condition $W_{UL}\mathbf{1}_{|L|} > \vec{0}$ implies that unlabeled observations that violate it can be safely removed from the data set. In the case where there are abundant unlabeled data, one could enforce a stronger condition, e.g. $W_{UL}\mathbf{1}_{|L|} > \gamma > 0$, since it would ignore unlabeled data not strongly associated with labeled observations.

Remark: For supervised logistic kernel regression, it is well known that optimizing the weighted loss function with kernel weights analogous to (2) results in an identical estimate as that of (2). Hence, logistic loss and squared error loss coincide in the case of supervised kernel smoothing. It is easy to show that this is also the case in the semi-supervised setting, and for this reason logistic kernel smoothing is not pursued in this work.

2.2 The Case of Learners Associated with a Linear Score Function

We investigate next the FTF algorithm in the case when one assumes that the procedure is linear, that is, $\eta = X\beta + \epsilon$ for some link function η. We first start by considering linear regression ($\eta = y$), where at iteration k the algorithm fits the following model: $\hat{Y}^k = X\beta^k + \epsilon$, with $\hat{Y}^k = \begin{pmatrix} Y_L \\ X_U\hat{\beta}^{k-1} \end{pmatrix}$ and $\epsilon \sim N(0, \sigma^2 I)$. Linear regression with the FTF procedure reduces to the following steps (i) $\hat{Y}_L^k = Y_L$ and (ii) $\hat{Y}^{k+1} = H\hat{Y}^k = X\hat{\beta}^k$, where $H = X(X^TX)^{-1}X^T = \begin{pmatrix} H_{LL} & H_{LU} \\ H_{UL} & H_{UU} \end{pmatrix}$ and $\hat{\beta}^k = (X^TX)^{-1}X^T\hat{Y}^k = (X^TX)^{-1}\left(X_L^TY_L + X_U^TX_U\hat{\beta}^{k-1}\right)$. It is also assumed that $(X^TX)^{-1}$ and the supervised solution $\widetilde{\beta}_{ls}$ exist.

Proposition 3. *If the X_U component of the data is in the space spanned by the X_L component, then the FTF algorithm converges independent of the initialization of $\hat{Y}_U^0$ to the least squares solution $\widetilde{\beta}_{ls}$.*

The iterative solution for the coefficient vector $\hat{\beta}^k$ converges to $\widetilde{\beta}_{ls}$ involving only the labeled data, as illustrated in Figure 4 and verified theoretically in the Appendix. Further, we get the following useful identities for the labeled and unlabeled data due to the closed form solution (6):

$$\begin{pmatrix} H_L \\ H_U \end{pmatrix} Y_L = X\widetilde{\beta}_{ls} = \begin{pmatrix} H_{LL} + H_{LU}(I - H_{UU})^{-1}H_{UL} \\ (I - H_{UU})^{-1}H_{UL} \end{pmatrix} Y_L.$$

Therefore the algebraic solution for the semi-supervised result coincides with the standard linear regression result in this special case. Notice also that the degrees of freedom are given by, $df = tr(H_{LL} + H_{LU}(I - H_{UU})^{-1}H_{UL}) = tr(H_L) = tr(H) = p + 1$ as is the case in linear regression.

A similar result can be established when the model is written in terms of B-splines, since it results in applying linear regression on basis expansions of the X data (Hastie et al., 2001). Further, the result can be extended to certain regularized versions in the linear regression setting (e.g. ridge regression, cubic smoothers, etc.). For example, one of the most common ways of regularizing linear regression is to incorporate an ℓ_2 (ridge) penalty on the coefficient vector at the minimization stage. The next result establishes the convergence of the algorithm for ridge regression, whose proof follows closely that for linear regression.

Corollary 2. *The FTF algorithm converges independent of the initialization of $\hat{Y}_U^0$ for ridge regression with fixed parameter λ, where in this case the solution is equivalent to supervised ridge regression.*

Notice that the convergent solution for both ridge regression and classical linear regression is rather

intuitive, since the assumed functional relationship is linear. Hence, the unlabeled data are not capable of influencing the solution and we observe the straightforward convergence to standard results.

Next, we study the FTF procedure for logistic regression. Specifically, at each iteration the FTF procedure optimizes a logistic loss function to fit the following model:

$$\eta(\hat{Y}^k) = X\beta + \epsilon.$$

with response vector $\hat{Y}^k = [Y_L \;\; \hat{Y}_U^k]^T$. In this context, $\hat{Y}_U^0$ are the unlabeled probability class estimates resulting from the procedure at iteration k. To optimize this loss function directly for fitting (3), one must iteratively incorporate Newton-Raphson steps via the local scoring algorithm (Hastie and Tibshirani, 1990; Hastie et al., 2001). In this case, FTF is performing the following steps with initialized unlabeled response vector $\hat{Y}_U^0$: (i) $\hat{Y}_L^k = Y_L$ and (ii) set $\hat{Y}^{k+1}$ as the minimizer of $\sum_{i \in L \cup U} \log\left(1 + e^{-g(y_i)\beta^T x_i}\right)$ with $g(y) = 2y - 1$. The following proposition establishes the convergence of this procedure:

Proposition 4. *If the X_U component of the data is in the space spanned by the X_L component, then the FTF algorithm converges independent of the initialization of $\hat{Y}_U^0$ to the supervised logistic regression solution.*

Notice that for each regression setting considered, the procedure yields the supervised estimate upon convergence. This is because of the following two facts. First, the optimization problem results in setting at the k-th iteration $X^T(\hat{Y}^k - \eta^{-1}(X\hat{\beta}^k)) = 0$. Using the supervised solution $\widetilde{\beta}_{sup}$ and supervised score $X_L^T(Y_L - \eta^{-1}(X_L\hat{\beta}_{sup})) = 0$, we have that

$$\begin{aligned} X^T(\hat{Y}^k - \eta^{-1}(X\hat{\beta}^k)) &= X_L^T(\eta^{-1}(X_L\widetilde{\beta}_{sup}) - \eta^{-1}(X_L\hat{\beta}^k)) \\ &\quad + X_U^T(\eta^{-1}(X_U\hat{\beta}^{k-1}) - \eta^{-1}(X_U\hat{\beta}^k)) = 0. \end{aligned}$$

Second, a Taylor series expansion of $\eta^{-1}(X\beta)$ about $\widetilde{\beta}_{sup}$, together with the condition on the X data gives $\hat{\beta}^k \longrightarrow \widetilde{\beta}_{sup}$. Notice that the above discussion applies to the general score algorithm which also encompasses the Poisson, negative binomial and Gamma cases. This result is fairly intuitive, since the underlying functional relationship is linear and there is no flexibility for the function estimate to adapt to the unlabeled data.

2.3 The FTF procedure as a Plug-in Semi-supervised Algorithm

In many applications, it is desirable to use other learners than the ones studied previously, such as regression and classification trees, generalized additive models, partial least squares, etc. Given the nature of these learners, it is hard to establish convergence results for the algorithm under consideration. We discuss some implementation issues through a simple example that extends regression trees and supervised generalized additive models (both under squared error and logistic loss functions) to the semi-supervised setting.

Consider the following simulated regression example: $Y = \sin(x_1) + e^{x_2} + \epsilon$ with $\epsilon \sim N(0, 0.1)$, $x_1 \sim U[0, \pi]$, and $x_2 \sim N(\log\left(\frac{2x_1}{3}\right), 1)$ for both the labeled and unlabeled X data. The correlation between x_1 and x_2 is 0.64. To fit semi-supervised additive models, a supervised *gam* initialized with loess smoothing was used as the base learner. The resulting solution is shown in Figure 5 (left panel) in the x_1 dimension for the case when there are 8 labeled observations and 92 unlabeled observations. The right plot provides the result for the x_2 dimension. The semi-supervised solution is less wiggly than the supervised one.

For FTF, the *tree* base learner is iteratively fit to the full data X using the current response, which results in a semi-supervised *tree*. For *tree*s, the data is partitioned into regions and the response is classified or averaged depending on the responses for the other observations in that region. The goal is to determine the optimal variable and positions within that variable to split (referred to as a split point) the data into regions. In the supervised case the *tree*'s split points are chosen midway between the labeled observations, and an optimization criteria (typically the Gini index (Breiman et al., 1984)) is employed to select the most optimal split point considering all possible split points among all the variables. The FTF algorithm in this case produces a semi-supervised *tree* where the unlabeled data can naturally influence the positions for determining the split points, which allows the optimal split point to be chosen more informatively between labeled and unlabeled data. Figure 6 shows the results for the simulated data using both a supervised and semi-supervised *tree*. In order to obtain a *tree* of reasonable size in the supervised case, at least two observations must be contained in each node for a split. When comparing the supervised and semi-supervised regression *tree*s, one must first note that despite their different structure, they still provide similar estimates. Therefore, from this result it is unclear whether the supervised or semi-supervised *tree* solution is preferable. However, the performance results on real data given in the next section suggest that the semi-supervised version exhibits a superior performance, especially in situations involving a small number of labeled cases.

Consider next the following simulated classification example for illustrating the use of generalized

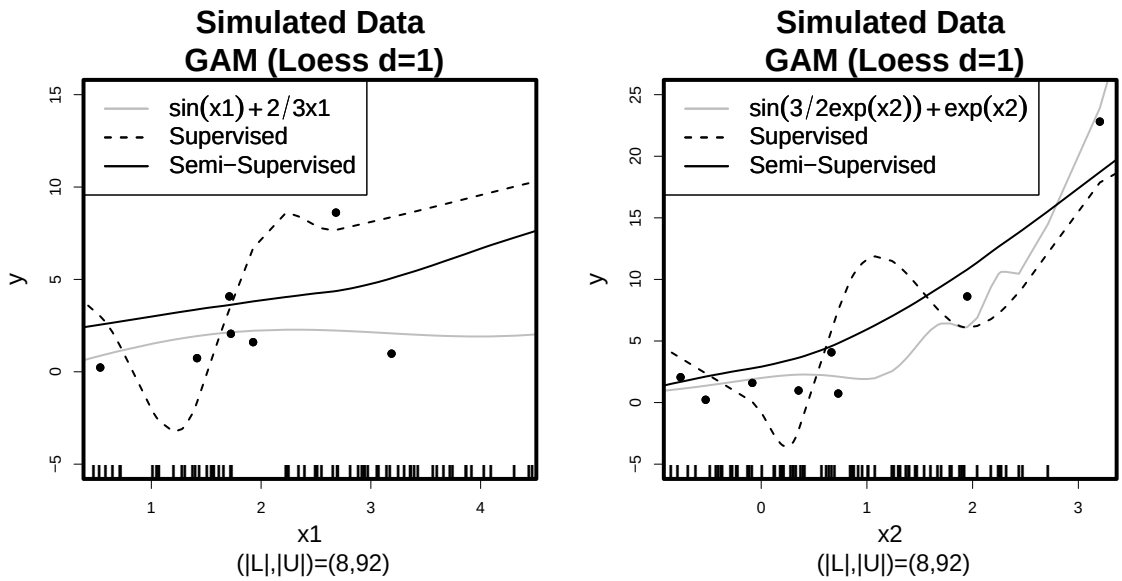

Figure 5: Illustration of the supervised *gam* and semi-supervised *gam* with loess initialization for dimension x_1 (left panel) and x_2 (right panel).

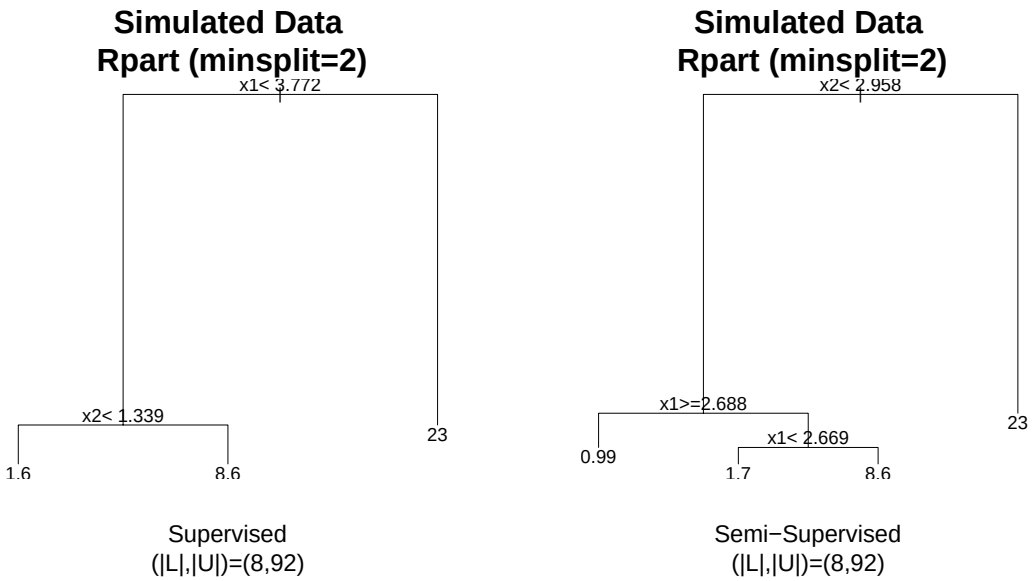

Figure 6: Illustration of the supervised *tree* and semi-supervised *tree* for a simple example.

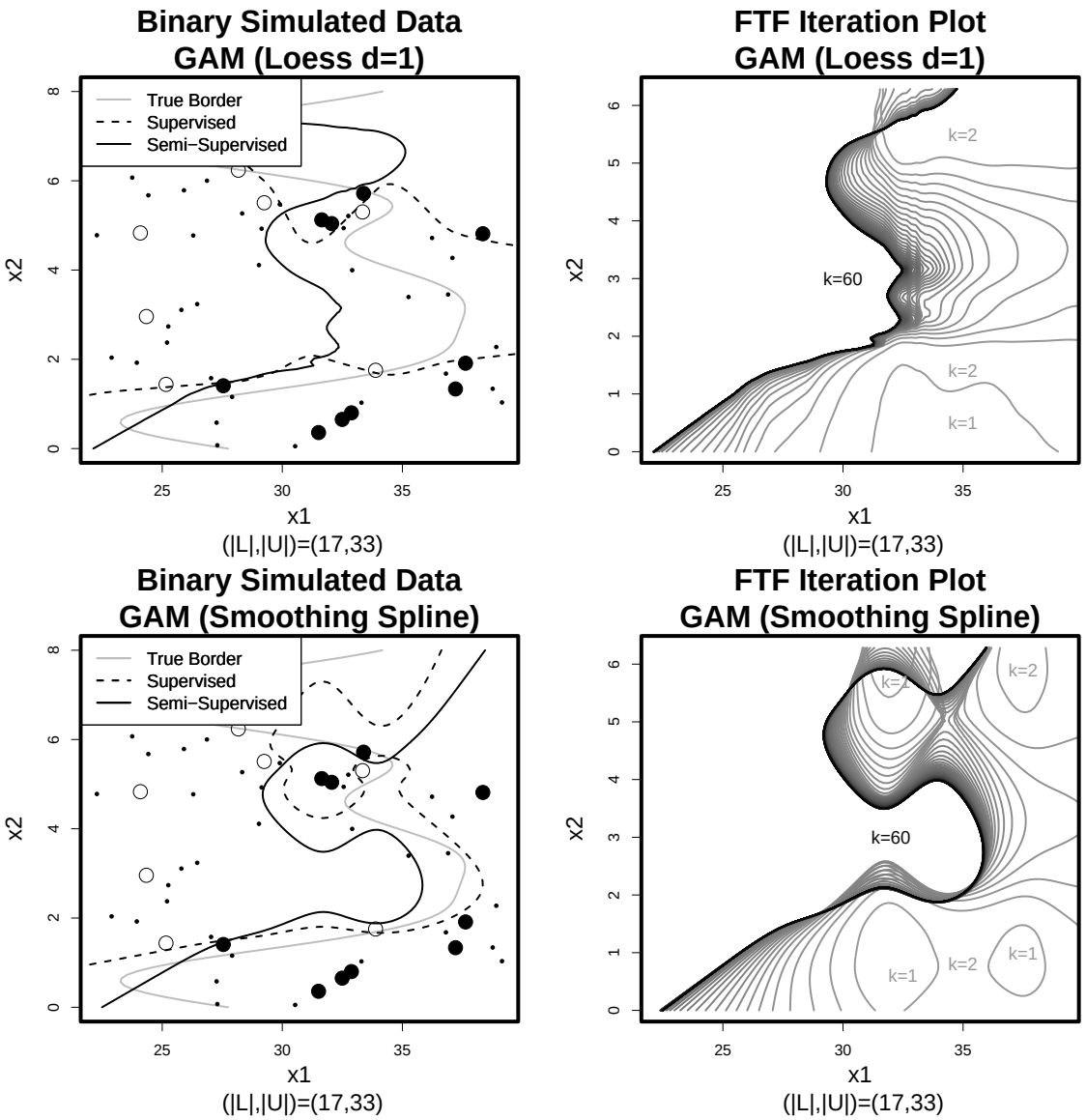

Figure 7: These plots depict the results for the *gam* learner under logistic loss. The upper two plots correspond to loess smoothing with $d = 1$, while the bottom two to smoothing splines. The left plots show the true function, together with the supervised and semi-supervised classification borders. The labeled classes are represented by large white and black circles, while the small black dots indicate the unlabeled observations. The right plots show the classification border for each iteration of the FTF algorithm. The goal here is to provide a visual assessment of the stepwise convergence of the procedure.

additive models under logistic loss as the base learner. Specifically, at each iteration the FTF procedure fits the following model:

$$\eta(\hat{Y}^k) = \sum_{j=1}^{p} h_j^{k+1}(x_j) + \epsilon.$$

In this example, the data generation mechanism was $x_1 \sim U[7\pi, 13\pi],\ x_2 \sim U[0, 2\pi]$ and

$$\eta = (\sin(2x_2) + \pi/2)\,(1 + \cos(x_2)) + \log\,(1 + \exp(x_1/\pi)) - 12 + \epsilon,$$

with $y = \frac{\exp(\eta)}{1+\exp(\eta)}$ and $\epsilon \sim N(0, 0.1)$. The unlabeled observations were from the same distribution as the labeled ones. This example was chosen to provide an interesting classification border that exhibits a challenge for both the supervised and semi-supervised approaches. A generalized additive model under logistic loss was fitted to the data, using a loess smoothing for initializing each observation. Figure 7 (upper left panel) shows the classification border for the supervised and semi-supervised result in the x_1 against x_2 dimension for $n = 50$ and $m = 17$. The supervised border appears to separate perfectly the labeled data, while the semi-supervised border appears to be slightly more stable and closer to the true border. The adjacent plot provides the stepwise FTF solution for each iteration $k = 1, \ldots, 60$ with the unlabeled response initialized to 0. Notice that the procedure is converging to the semi-supervised border in a fairly remarkable and intuitive way. The corresponding results (lower panel of Figure 6) are provided for the smoothing spline version of the *gam* with similar findings. For smoothing spline version of the additive model it appears that both the supervised and semi-supervised results are quite competitive indicating that the unlabeled data may not improve the smoothing spline version as much as with loess, which is consistent with the results in Section 3.

Remark: For the additive model under squared error loss, one could equivalently find a unique smoother R such that FTF is (i) $\hat{Y}_L^k = Y_L$ (ii) $\hat{Y}^{k+1} = R\hat{Y}^k$ (i.e. it is a linear smoothing technique) (Hastie and Tibshirani, 1990). In this case employing (2.3) for obtaining the solution directly is fairly complex since one would have to first obtain the smoother R, compute $(I - R_{UU})^{-1}$, and estimate the smoothing parameters for each smoother term in the additive model. To speed up computation in this case we therefore employ the additive model directly at step 2 of FTF, however, here the trade-off is slow and possibly non-monotone convergence. [2]

Remark: In the analysis, generalized additive models and classification trees were used as base learners for the algorithm, and implemented in the corresponding **R** *gam* (Hastie, 2006) and *rpart* (Therneau and

[2] Slight fluctuations in the implemented parameter estimation can cause convergence changes at subsequent iterations.

Atkinson., 2005) packages (R Development Core Team, 2005). We have empirically noticed that the convergence of the *tree* depends on the initialization of $\hat{Y}_U^0$, and for the results in this section this was set to 0. Given the greedy construction of regression/classifications trees, this empirical finding is intuitive.

3 Illustration of the Algorithm with Real Data

In this section the proposed algorithm applied to a number of real data sets. In addition, the goal is to assess the difference between the supervised learner and the semi-supervised version of that learner on the data.

3.1 The Air Quality Data

This data set consists of 146 measurements of daily air quality in New York from May to September 1973 (Cleveland et al., 1983). These include Solar Radiation (Solar.R), Wind, Temp, Month and Day and the goal is to predict Ozone levels. This data is of particular illustrative interest in our context, since generalized additive models are known to work well (Cleveland et al., 1983; Hastie, 2006). In addition, there is also an unlabeled partition set of size 37, where the response values are missing. For this data set, it has been previously established that the response needs a transformation and the model of interest is:

$$Ozone^{1/3} \sim h_1(Solar.R) + h_2(Wind) + h_3(Temp).$$

To fit this model, we applied the supervised *gam* to the data using loess smoothing for initialization purposes. In the semi-supervised case, the FTF procedure was used in plug-in mode with the *gam* function as input. The unlabeled responses were initialized to the supervised *gam* and converged in four iterations. The results are given in the three plots on the top row of Figure 8. The results show that the amount of smoothing for all three variables is nearly identical between the supervised and semi-supervised solutions, with minor differences not visually distinguishable to the naked eye. This result is rather intuitive since there are already 109 labeled observations in this data and the additional 37 unlabeled observations should not overly influence the smooth.

To illustrate the usefulness of the proposed algorithm in the presence of a larger number of unlabeled data, we partitioned the labeled data set into 8 labeled cases and concatenated the unlabeled data set with 101 additional testing cases. For the results, refer to the first plot in the second row of Figure 8. The

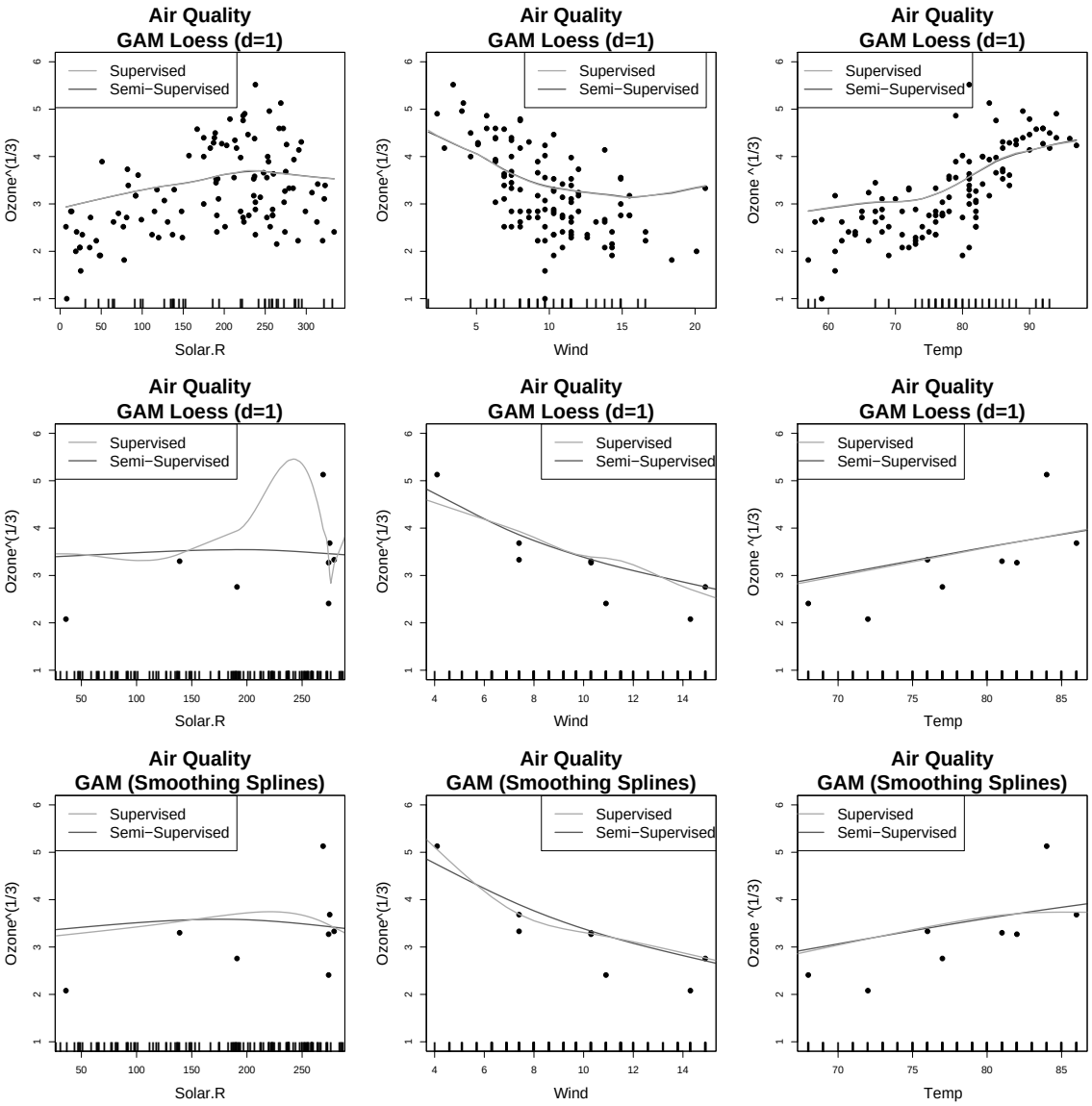

Figure 8: The plots are organized on a 3×3 grid. In each plot the function for each dimension in the Air Quality data set is shown against the transformed response for both the supervised and semi-supervised solution. The first row of plots consists of the functions in each dimension with $m = 109$ (labeled size) and $n = 146$ (full data size) for a supervised and semi-supervised *gam* with loess smoothing. The second row restricts attention to the case when only $m = 8$ observations are labeled ($n = 146$). The third row applies a *gam* initialized with smoothing splines for this restricted case.

procedure converged in about 110 iterations. The semi-supervised result appears to be more stable, while the supervised result is dramatically pulled by one observation. Notice that the semi-supervised smooth is actually not that far off from the case when 109 observations were available during training (above figure); this is important in practice, considering that only 8 labeled cases are available. In the adjacent plot (the smooth on the Wind variable), the semi-supervised solution is now slightly more stable than the supervised solution and the two results are nearly identical for the Temp variable (third plot in second row). The supervised *gam* is nearly linear (much less wiggly) in these variables, and in these situations the semi-supervised *gam* produces a similar result. It is reasonable to assume that any data partitioning with a labeled set this small will result in a strong pull of the smooth towards one observation for some variable, and the FTF algorithm will be more stable in this situation.

For this example, we also considered the case of applying a *gam* model initialized with smoothing splines. The smoothing spline provides a regularized natural spline basis expansion for smoothing (Hastie et al., 2001). The results for the same 8 labeled observations are provided with this version of the *gam* in the third row of Figure 8. This smooth is more stable than the corresponding loess version, but we still observe some variation in all three variables between the supervised and semi-supervised solution. In this capacity, the semi-supervised version of the *gam* will most likely not be as effective as in the case of loess smoothing. However, we have observed marginal improvement in low labeled size situations for the smoothing spline version as well.

From a practical perspective it is interesting to determine which smooth is better according to some measurement. To ascertain this we applied standard semi-supervised benchmark plots to this data using the average residual sum of squares (RSS) given by, $\frac{\|Y_L - \hat{Y}_L\|_2}{|L|}$ on the labeled data and $\frac{\|Y_U - \hat{Y}_U\|_2}{|U|}$ on the unlabeled data. The procedure performs the best on this data if this value is minimized. These plots display the partition of the training data into components with 10% to 90% of the labeled information available ($[Y_L \mid X_L]$ is available to the procedure), and the remaining testing observations are annexed to the original 37 missing observations to form the unlabeled set (the X_U partition is available to the procedure). The average RSS over 50 random partitions for each percentage is plotted as the y axis. In addition to additive models with loess or smoothing spline initializations, we also applied this measurement to the Air Quality data using FTF with a *tree* base learner, and a *pls* base learner ($d = 2$ dimensions).

The results are given in a 4×3 grid of plots in Figure 9. The results for the *gam* initialized with loess are in the first row of the Figure. The first two columns provide the training and testing RSS, respectively. The third column of plots provides a zoom in view of the testing RSS, with particular emphasis on the smaller labeled size partitions. Interestingly the supervised solution achieves a lower training value.

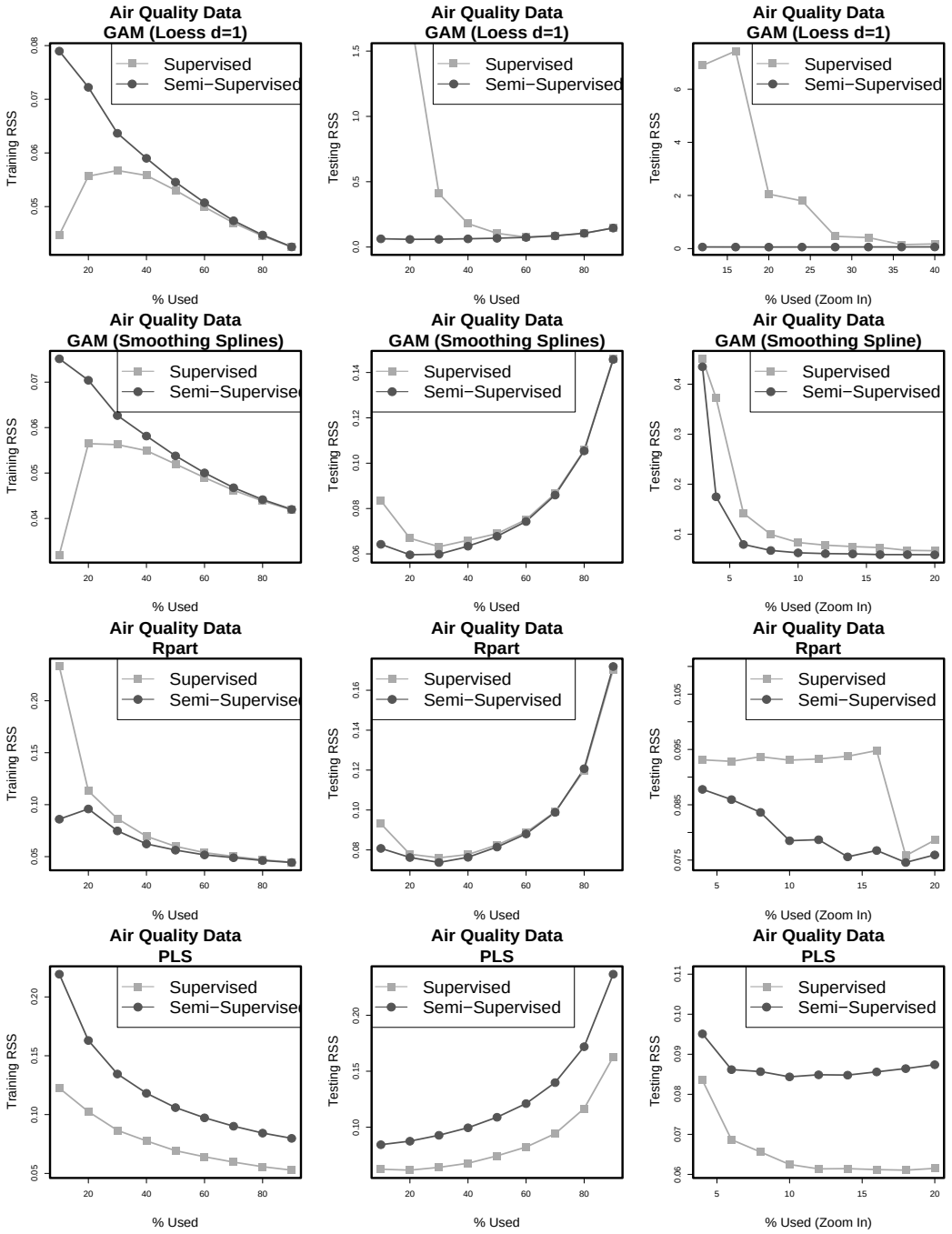

Figure 9: Each plot shows the average (over 50 replications) training/testing RSS as a function of the %Used cases (labeled data). Each row shows a different base learner for FTF; first row uses *gam* initialized with loess, second row *gam* initialized with smoothing splines, third row uses a *tree* learner (implemented via *rpart*), and the last row the *pls* learner. The first column of plots provides the average training RSS where the %-Used varied from 10 to $90-\%$. The second column depicts the corresponding testing RSS. The third column provides a zoom-in view of the testing RSS, with the %-Used restricted to a smaller set.

However, the semi-supervised solution achieves a significantly lower testing RSS for many values. As the number of labeled data increases the differences between the two solutions decreases. This result is consistent with the previous findings seen in Figure 8. Similarly, a *gam* initialized with smoothing splines also provides similar improvements (second row of Figure 9); however, the improvement is not as pronounced as in the loess case (middle column). The semi-supervised result for the *tree* also yields improvement in testing performance for extremely low labeled partitions (third row of Figure 9). The last learner considered for this data was partial least squares [3] in the fourth row of Figure 9. The results suggest that the semi-supervised version of this approach does not perform well for this data. A result of this type is consistent with the literature on semi-supervised learning where one has to match appropriately the technique to the unlabeled data (Zhu, 2006). Overall, this data set provides evidence that the *gam* and *tree* learners lead to improvements in performance when enhanced with unlabeled data.

3.2 Application to Pharmacology Data

The data set consists of 3925 chemical compounds on which an in-house continuous solubility screen (the concentration to dissolve in a water/solvent mixture) was performed. The covariate information consisted of 20 attributes that are obtained through computations, which are known to be theoretically related to a compound's solubility. For this data set, the value $Y = 3$ corresponding to 1483 compounds in the data set indicated the compound was mostly insoluble (since measurements are not possible for $Y < 3$) and similarly for the $y = 60$ (1100 compounds) case. Scientists and statisticians involved in the analysis of this data set were primarily interested in applying additive models to predict compound solubility. For this analysis we focused on the regression problem of predicting the observations whose responses take values in the $3 < y < 60$ range.

The data set was partitioned into a training/testing set as before with a varying %Used for training purposes. This set contained all the compounds with intermediate values $3 < Y < 60$. In this example we fit generalized additive models with loess smoothing, as well as with smoothing splines. Although the main interest is in additive models, we also provide results for regression trees.

The results are presented in the 2×3 grid of plots shown in Figure 10. The first plot in the first row shows the supervised and semi-supervised *gam* solutions using loess smoothing. The supervised version does not perform well in low labeled situations, and is incredible sensitive to the % of data available for training purposes. The semi-supervised *gam* is more stable and in general provides lower values for testing RSS. In the case of smoothing splines (second column of plots), the supervised *gam*

[3] The partial least squares learner was implemented with the **R** *pls* package (Wehrens and Mevik, 2006).

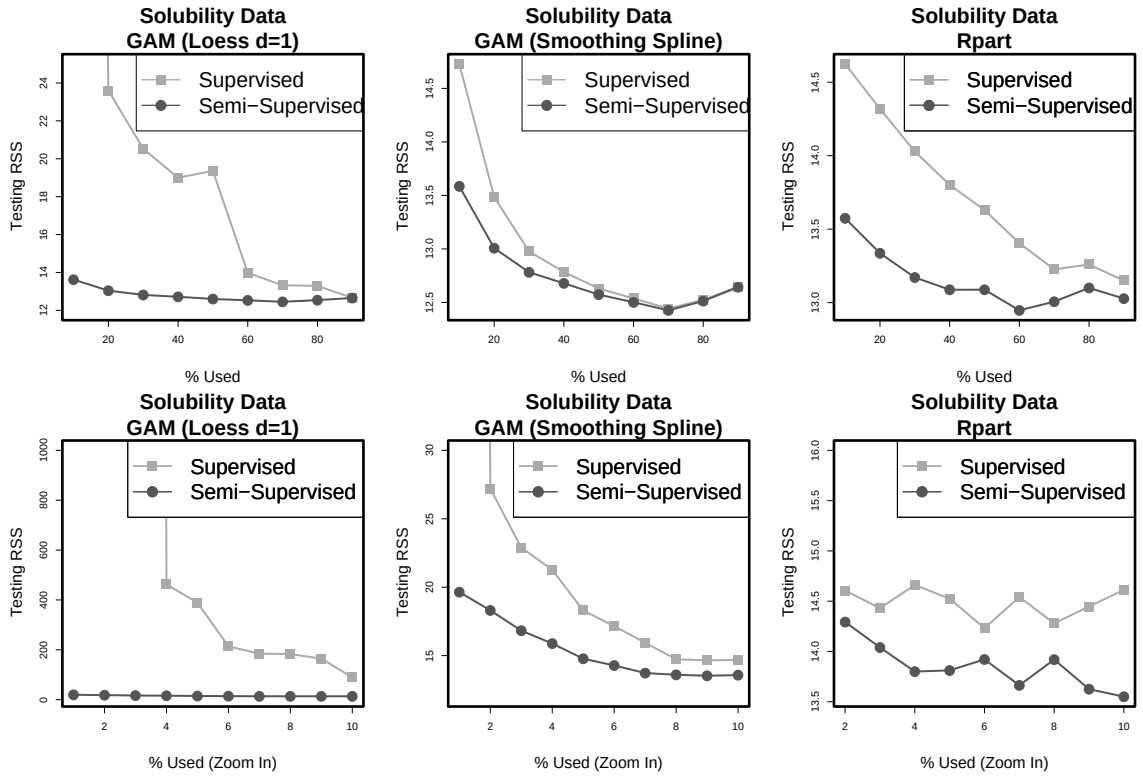

Figure 10: Each plot shows the average (over 50 replications) testing RSS as a function of %Used cases (labeled data points) for the Solubility data set. The top row shows the result, with the %Used varying from 10 to 90-%. A zoom-in view for very low %Used cases, varying from 1 to 10% of the data used for training purposes is shown in the bottom row. The first column shows the supervised and semi-supervised result for the *gam* learner with loess smoothing, the second column for *gam* with smoothing splines, and the last column for the *tree* learner.

is fairly stable and the improvement for training with unlabeled data is less pronounced. Notice that in the semi-supervised case, both the loess and the smoothing spline versions exhibit similar performance, whereas in the supervised case these approaches perform very differently. In an extreme setting (1 to 10-% of labeled data) we observe that the semi-supervised version provides a general improvement over the supervised version for all three procedures (second row of plots). Although the above partitions may seem extreme, it is actually quite common practice in this field to have such a low percentage of labeled data available.

In addition to prediction with additive models, there was also interest in a potential followup analysis using the diagnostic tools commonly associated with this modeling technique (e.g. inference) (Hastie and Tibshirani, 1990; Hastie et al., 2001). Interestingly, the semi-supervised *gam* still retains characteristics associated with additive models, and unlabeled data could potentially influence all diagnostics associated with it.

4 Concluding Remarks

In this paper, an iterative algorithm is introduced for extending supervised learners into the semi-supervised setting and its global convergence established for the class of linear smoothers and learners that lead to linear score functions. From the analysis, it can be seen that unlabeled observations can significantly aid in estimating functions on data in small labeled size problems. We further provide an intuitive explanation for when convergence will occur, based on the quality of the connections between labeled and unlabeled data. The FTF algorithm can also be applied in plug-in mode, where one wishes to examine their data using some other learner. This approach is highlighted with the semi-supervised version of a *gam* in both a classification and a regression context where significant improvements are observed. The results in this work are quite useful in many applications that involve complex data scenarios. As noted in the introduction, the usefulness of unlabeled data is not well understood in semi-supervised learning. The proposed algorithm may serve as a platform to obtain additional insights into this important issue. Further, there are a number of interesting theoretical questions about consistency for fixed $\mid L \mid$ as $\mid U \mid \rightarrow \infty$, a topic currently under investigation.

Acknowledgments: The authors would like to thank the Editor, Associate Editor and two anonymous reviewers for their useful and constructive comments.

Appendix: Proofs

Proposition 1. *Let S be a symmetric stochastic matrix (i.e. $S\mathbf{1} = \mathbf{1}$, $S \geq 0$) and S_{UU} be an irreducible and substochastic matrix. Then, the FTF algorithm converges independent of the initialization and the solution is given by*

$$\hat{Y}_U = (I - S_{UU})^{-1} S_{UL} Y_L,$$

provided S_{UU} is of full rank.

Proof. In order to have that S_{UU} is substochastic, it suffice to have indices j, such that for every index i we have $S_{UL}(i,j) \neq 0$.

Let ν be a real vector taking values in $I\!R^{n-m}$. It is sufficient to prove the result for positive ν with norm 1. Define $T_n = \frac{S_{UU}+S_{UU}^2+\ldots+S_{UU}^n}{n}$. Since $\| S_{UU}\nu \|_1 \leq \| \nu \|_1$ for every positive vector ν one has $\| S_{UU}^{n+1}\nu \|_1 \leq \| S_{UU}T_n\nu \|_1$ for every $n > 0$. Choose n such that $T_n\nu \geq \delta\mathbf{1}$ (δ exists by irreducibility of

S_{UU}).

Since S_{UU} is substochastic there exists $\alpha > 0$ and index j such that

$$\sum_i S_{UU}(i,j) \leq 1 - \alpha \tag{9}$$

Therefore, $\| S_{UU}^{n+1}\nu \|_1 \leq \| S_{UU}T_n\nu \|_1 \leq (1-\alpha\delta) \| \nu \|_1 = (1-\alpha\delta) < 1$. The convergent geometric matrix series (5) implies the solution. □

Proposition 2. *Let $S = D^{-1}W$ where W emits the partition given in (2.3). If $W_{UL}\mathbf{1}_{|L|} > \vec{0}$ and W_{UU} is irreducible, then the FTF algorithm converges independent of its initialization, since $\rho(S_{UU}) < 1$.*

Proof. Let $T_U = D_{UU}^{-1}W_{UU}$ and recall that D_{UU} =diag$(W_{UU}\mathbf{1}_U)$. Now, from the Perron-Frobenius Theorem we have that $\rho(T_U) = 1$ by the above assumption on W_{UU}. By the assumption on W_{UL} we have that D_U =diag$(W_{UL}\mathbf{1}_{|L|}) + D_{UU} \geq \delta I + D_{UU}$ for some $\delta > 0$. Therefore, $\rho(S_{UU}) = \rho(D_U^{-1}W_{UU}) \leq \rho((\delta I + D_{UU})^{-1}D_{UU}T_U) < \rho(T_U) = 1$. Then as before, the convergent geometric matrix series establishes the result. □

Proposition 3. *If the X_U component of the data is in the space spanned by the X_L component, then the FTF algorithm converges independent of the initialization of $\hat{Y}_U^0$ to the least squares solution $\widetilde{\beta}_{ls}$*

Proof. In the case of FTF with linear regression, we have that the procedure first sets $\hat{Y}^k = (Y_L\ \hat{Y}_U^k)^T$, and then sets $\hat{Y}^{k+1} = H\hat{Y}^k = X\hat{\beta}^k$ where $H = X^T(X^TX)^{-1}X^T$. At the $k+1$-st iteration the solution can be written as

$$\begin{aligned}
\hat{\beta}^{k+1} &= \left((X_L^T \quad X_U^T) \begin{pmatrix} X_L \\ X_U \end{pmatrix} \right)^{-1} (X_L^T \quad X_U^T) \begin{pmatrix} Y_L \\ X_U\hat{\beta}^k \end{pmatrix} \\
&= (X_L^TX_L + X_U^TX_U)^{-1}(X_L^TY_L + X_U^TX_U\hat{\beta}^k) \\
&= (X_L^TX_L + X_U^TX_U)^{-1}(X_L^TX_L\widetilde{\beta}_{ls} + X_U^TX_U(\hat{\beta}^k - \widetilde{\beta}_{ls} + \widetilde{\beta}_{ls})) \\
&= \widetilde{\beta}_{ls} + [(X_L^TX_L + X_U^TX_U)^{-1}(X_U^TX_U)](\hat{\beta}^k - \widetilde{\beta}_{ls}).
\end{aligned}$$

Iterating and doing some bookkeeping we obtain

$$\hat{\beta}^{k+1} = \widetilde{\beta}_{ls} + \left[(X_L^TX_L + X_U^TX_U)^{-1}(X_U^TX_U)\right]^{k+1}(\hat{\beta}^0 - \widetilde{\beta}_{ls}).$$

By the condition that X_U is contained in the space spanned by X_L we get that

$$X_U^T X_U < X_L^T X_L + X_U^T X_U,$$

in the Lowner ordering, which implies that

$$(X_L^T X_L + X_U^T X_U)^{-1}(X_U^T X_U) < I,$$

and therefore the radial spectrum of the left hand side is less than 1 (i.e. $\rho((X_L^T X_L + X_U^T X_U)^{-1}(X_U^T X_U)) < 1$). Hence, the $[(X_L^T X_L + X_U^T X_U)^{-1}(X_U^T X_U)]^k \to 0$ and thus $\hat{\beta}^k \to \widetilde{\beta}_{ls}$. □

Proposition 4. *If the X_U component of the data is in the space spanned by the X_L component, then the FTF algorithm converges independent of the initialization of $\hat{Y}_U^0$ to the supervised logistic linear regression solution.*

Proof. To start, we set-up the supervised problem. First, let $\widetilde{\beta}_{\log}$ be the solution to supervised logistic regression, and further define $p(X, \beta) = \frac{e^{\beta^T X}}{1+e^{\beta^T X}}$ as the probability class estimate (PCE). From this we have that the labeled PCEs for supervised logistic regression are $p_L^{log} = p(X_L, \widetilde{\beta}_{\log})$ and the predictions for the unlabeled partition are $p_U^{log} = p(X_U, \widetilde{\beta}_{\log})$. In addition, define $W_{ij} = p(x_i, \widetilde{\beta}_{\log})(1 - p(x_j, \widetilde{\beta}_{\log}))\mathbf{1}_{\{i=j\}}$ for $i, j \in L \cup U$. Notice that, $\widetilde{\beta}_{\log} = (X_L^T W_L X_L)^{-1} X_L W_L z$ where z is the score that corresponds to the convergent solution for this procedure (Hastie et al., 2001, Chapter 4). Also notice that the solution satisfies the following, $X_L^T(Y_L - p_L^{\log}) = 0$.

Next, we consider FTF with logistic linear regression. To start, assume that FTF procedure has been executed for k iterations. To determine the $k+1$'st estimate, set the fitted response as $\hat{Y}^k = \begin{pmatrix} Y_L \\ p_U^k \end{pmatrix}$ (notice that $p_U^k = p(X_U, \hat{\beta}^k)$), and then solve $\min_\beta \ell(\beta)$ where

$$\begin{aligned} \ell(\beta) &= \sum_{i \in L \cup U} \log\left(1 + e^{-g(\hat{Y}_i^k)\beta^T x_i}\right) \\ &= \sum_{i \in L \cup U} \hat{Y}_i^k \beta^T x_i - \log\left(1 + e^{\beta^T x_i}\right). \end{aligned}$$

Taking the derivative and setting to zero, we note that the solution must satisfy $X^T(\hat{Y}^k - p^k) = 0$ for all k, which implies that

$$0 = X_L^T(Y_L - p_L^k) + X_U^T(p_U^{k-1} - p_U^k) = X_L^T(p_L^{\log} - p_L^k) + X_U^T(p_U^{k-1} - p_U^k), \tag{10}$$

since $X_L^T(Y_L - p_L^{\log}) = 0$. Consider next, a Taylor expansion of $p(X, \beta)$ about $\widetilde{\beta}_{\log}$, given by $p(x, \beta) = p(x, \widetilde{\beta}_{\log}) + \frac{\partial p(x,\beta)}{\partial \beta}(\widetilde{\beta}_{\log})(\beta - \widetilde{\beta}_{\log}) + O(1)$. From this result, (10) becomes

$$X_L^T W_L X_L(\hat{\beta}^k - \widetilde{\beta}_{\log}) + X_U^T W_U X_U(\hat{\beta}^k - \hat{\beta}^{k-1}) + O(1) = 0.$$

Rearranging terms gives

$$\begin{aligned} \hat{\beta}^k &= (X^T W X)^{-1}(X_L^T W_L X_L \widetilde{\beta}_{\log} + X_U^T W_U X_U \hat{\beta}^{k-1}) + O(1) \\ &= \widetilde{\beta}_{\log} + [(X^T W X)^{-1}(X_U^T W_U X_U)](\hat{\beta}^k - \widetilde{\beta}_{\log}) + O(1) \end{aligned}$$

Applying the same argument as with linear regression (Proposition 3) to obtain that $\hat{\beta}^{k+1} = \widetilde{\beta}_{\log} + [(X^T W X)^{-1}(X_U^T W_U X_U)]^k(\hat{\beta}^0 - \widetilde{\beta}_{\log}) + O(1)$. From the condition on X_U and that $p^{\log} \in (0,1)^{|L \cup U|}$ we have that $\hat{\beta}^k \longrightarrow \widetilde{\beta}_{\log}$. □

References

S. Abney. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365–395, 2004.

S. Baluja. Probabilistic modeling for face orientation discrimination: learning from labeled and unlabeled data. In *NIPS*, pages 854–860, 1999.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26, 2001.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

A. Buja, T. Hastie, and R. Tibshirani. Linear smoothers and additive models. *The Annals of Statistics*, 17:543–555, 1989.

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*, 42:2102–2117, 1996.

O. Chapelle, M. Chi, and A. Zien. A continuation method for semisupervised svms. In *ICML*, 2006a.

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

J. Cleveland, W. Kleiner, and P. Turkey. *Graphical Methods for Data Analysis*. Wadsworth, 1983.

M. Culp and G. Michailidis. Graph based semi-supervised learining. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2007. To appear.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, New York, NY, 2000.

R. Floyd. Algorithm 97: Shortest part. In *Communications of the ACM*, page 345, 1962.

T. Hastie. gam*: Generalized Additive Models*, 2006. **R** package version 0.97.

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman & Hall, CRC, Boca Raton, FL, 1990.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning (Data Mining, Inference and Prediction)*. Springer, New York, NY, 2001.

R. Johnson and T. Zhang. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 2007. To appear.

R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, pages 315–322, 2002.

B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo. On semi-supervised classification. In *NIPS*, pages 721–728, 2005.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134, 2000.

R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

B. Ripley. *Pattern recognition and neural networks*. Cambridge University Press, Cambrdige, UK, 1996.

T. Therneau and B. Atkinson. *rpart: Recursive Partitioning*, 2005. URL `S-PLUS 6.x original at http://mayoresearch.mayo.edu/mayo/research/biostat/splusfunctions.cfm`. R port by B. Ripley.

V. Vapnik. *Statistical learning theory*. Springer, New York, NY, 1998.

R. Wehrens and B. Mevik. *pls: Partial Least Squares Regression (PLSR) and Principal Component Regression (PCR)*, 2006. URL `http://mevik.net/work/software/pls.html`. R package version 1.2-1.

T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *ICML*, pages 1191–1198, 2000.

X. Zhu. Semi-supervised learning with graphs. Technical report, Carnegie Mellon University, Pittsburgh, PA, 2005.

X. Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2006.

X. Zhu and Z. Ghahramani. Learning from labeled and unabeled data with label propagation. Technical report, Carnegie Mellon University, Pittsburgh, PA, 2002.

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