Online Manifold Regularization: A New Learning Setting and Empirical Study

Andrew B. Goldberg[1], Ming Li[2], and Xiaojin Zhu[1]

[1] Department of Computer Sciences, University of Wisconsin-Madison
Madison, WI, USA. {goldberg, jerryzhu}@cs.wisc.edu
[2] National Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China. lim@lamda.nju.edu.cn

Abstract. We consider a novel "online semi-supervised learning" setting where (mostly unlabeled) data arrives sequentially in large volume, and it is impractical to store it all before learning. We propose an online manifold regularization algorithm. It differs from standard online learning in that it learns even when the input point is unlabeled. Our algorithm is based on convex programming in kernel space with stochastic gradient descent, and inherits the theoretical guarantees of standard online algorithms. However, naïve implementation of our algorithm does not scale well. This paper focuses on efficient, practical approximations; we discuss two sparse approximations using buffering and online random projection trees. Experiments show our algorithm achieves risk and generalization accuracy comparable to standard batch manifold regularization, while each step runs quickly. Our online semi-supervised learning setting is an interesting direction for further theoretical development, paving the way for semi-supervised learning to work on real-world lifelong learning tasks.

1 Introduction

Consider a robot with a video camera. The robot continuously takes high framerate video of its surroundings, and wants to learn the names of various objects in the video. However, like a child learning in the real world, the robot receives names from humans only very rarely. The robot is thus in a semi-supervised learning situation: most objects are unlabeled, while only a few are labeled by humans.

There are several challenges that distinguish this situation from standard semi-supervised learning. The robot cannot afford to store the massive amount of mostly unlabeled video before learning; it requires an "anytime classifier" that is ready to use at all times, yet is continuously improving; training must be cheap; and since the world is changing, it must adapt to non-stationarity in classification.

These challenges are well-studied in online learning. However, our situation is also different from standard online learning. Online learning (classification) traditionally assumes that every input point is fully labeled; it cannot take advantage of unlabeled data. But in the robot case, the vast majority of the input

will be unlabeled. It seems wasteful to throw away the unlabeled input, as it may contain useful information.

We address this situation by combining semi-supervised learning with online learning. The resulting online semi-supervised learning algorithm is based on convex programming with stochastic gradient descent in kernel space. This combination is novel. To the best of our knowledge, the closest prior work is the multiview hidden Markov perceptron ([1], Section 4), which heuristically combines multiview learning with online perceptron. However, that work did not enjoy the theoretical guarantees afforded by the online learning literature, nor did it directly apply to other semi-supervised learning methods. In contrast, our method can lift any batch semi-supervised learning methods with convex regularized risks to the online setting. As a special case, we will discuss online manifold regularization in detail.

The focus of the present work is to introduce a novel learning setting, and to develop practical algorithms with experimental verification. It is important to consider the efficiency issues, as we do in Section 3, for the algorithm to be practically relevant. Our online semi-supervised learning algorithm inherits no-regret bounds from online convex programming but does not provide new bounds. It is our hope that the novel setting where most of the incoming data stream is unlabeled will inspire future work on improved bounds. Some of the future directions are laid out at the end of the paper.

2 Online Semi-Supervised Learning

We build online semi-supervised learning with two main ingredients: online convex programming [2] and regularized risk minimization for semi-supervised learning (see the overview in [3, 4]). Although kernel-based online convex programming is well-understood [5], we are not aware of prior application in the semi-supervised learning setting.

Consider an input sequence $x_1 \dots x_T$, where $x_t \in \mathbb{R}^d$ is the feature vector of the t-th data point. *Most (possibly even the vast majority) of the points are unlabeled.* Only occasionally is a point x_t accompanied by its label $y_t \in \mathcal{Y}$. This setting differs dramatically from traditional online learning where all points are labeled. Let K be a kernel over x and $\mathcal{H}_K$ the corresponding reproducing kernel Hilbert space (RKHS) [6]. Our goal is to learn a good predictor $f \in \mathcal{H}_K$ from the sequence. Importantly, learning proceeds in an iterative fashion:

1. At time t an adversary picks x_t and y_t, not necessarily from any distribution $P(x, y)$ (although we will later assume *iid* for predicting future data). The adversary presents x_t to the learner.
2. The learner makes prediction $f_t(x_t)$ using its current predictor f_t.
3. With a small probability p_l, the adversary reveals the label y_t. Otherwise, the adversary abstains, and x_t remains unlabeled.
4. The learner updates its predictor to f_{t+1} based on x_t and the adversary's feedback y_t, if any.

We hope the functions $f_1 \dots f_T$ "do well" on the sequence, and on future input if the data is indeed *iid*. The exact performance criteria is defined below.

2.1 Batch Semi-Supervised Risks

Before introducing our online learning algorithm, we first review *batch* semi-supervised learning, where the learner has access to the labeled and unlabeled data all at once. A unifying framework for batch semi-supervised learning is risk minimization with specialized "semi-supervised" regularizers. That is, one seeks the solution $f^* = \text{argmin}_{f \in \mathcal{H}_K} J(f)$, where the *batch semi-supervised regularized risk* is

$$J(f) = \frac{1}{l} \sum_{t=1}^{T} \delta(y_t) c(f(x_t), y_t) + \frac{\lambda_1}{2} \|f\|_K^2 + \lambda_2 \Omega(f),$$

where l is the number of labeled points, $\delta(y_t)$ is an indicator function equal to 1 if y_t is present (labeled) and 0 otherwise, c is a convex loss function, λ_1, λ_2 are regularizer weights, $\|f\|_K$ is the RKHS norm of f, and Ω is the semi-supervised regularizer which depends on f and $x_1 \dots x_T$. Specific choices of Ω lead to familiar semi-supervised learning methods:

i) *Manifold regularization* [7–9]:

$$\Omega = \frac{1}{2T} \sum_{s,t=1}^{T} (f(x_s) - f(x_t))^2 w_{st}.$$

The edge weights w_{st} define a graph over the T points, e.g., a fully connected graph with Gaussian weights $w_{st} = e^{-\|x_s - x_t\|^2 / 2\sigma^2}$. In this case, Ω is known as the energy of f on the graph. It encourages label smoothness over the graph: similar examples (large w) tend to have similar labels.

ii) *Multiview learning* [10–12] optimizes multiple functions $f_1 \dots f_M$ simultaneously. The semi-supervised regularizer

$$\Omega = \sum_{i,j=1}^{M} \sum_{t=1}^{T} (f_i(x_t) - f_j(x_t))^2$$

penalizes differences among the learners' predictions for the same point.

iii) *Semi-supervised support vector machines (S3VMs)* [13–15]:

$$\Omega = \frac{1}{T-l} \sum_{t=1}^{T} (1 - \delta(y_t)) \max(1 - |f(x_t)|, 0).$$

This is the average "hat loss" on unlabeled points. The hat loss is zero if $f(x)$ is outside $(-1, 1)$, and is the largest when $f(x) = 0$. It encourages the decision boundary $f(x) = 0$ to be far away from any unlabeled points (outside the margin), thus avoiding cutting through dense unlabeled data regions.

2.2 From Batch to Online

A key observation is that for certain semi-supervised learning methods, the batch risk $J(f)$ is the sum of *convex functions* in f. These methods include manifold regularization and multiview learning, but not S3VMs whose hat loss is non-convex. For these convex semi-supervised learning methods, one can derive a corresponding online semi-supervised learning algorithm using online convex programming. The remainder of the paper will focus on manifold regularization, with the understanding that online versions of multiview learning and other convex semi-supervised learning methods can be derived similarly.

We follow the general approach in [2, 5]. Recall the batch risk for our version of manifold regularization in Section 2.1 is

$$J(f) = \frac{1}{l}\sum_{t=1}^{T}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \frac{\lambda_2}{2T}\sum_{s,t=1}^{T}(f(x_s) - f(x_t))^2 w_{st}, \quad (1)$$

and f^* is the batch solution that minimizes $J(f)$. In online learning, the learner only has access to the input sequence up to the current time. We thus define the *instantaneous regularized risk* $J_t(f)$ at time t to be

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2\sum_{i=1}^{t-1}(f(x_i) - f(x_t))^2 w_{it}. \quad (2)$$

The last term in $J_t(f)$ involves the graph edges from x_t to all previous points up to time t. The astute reader might notice that this poses a computational challenge—we will return to this issue in Section 3. While T appears in (2), $J_t(f)$ depends only on the *ratio* T/l. This is the empirical estimate of the inverse label probability $1/p_l$, which we assume is given and easily determined based on the rate at which humans can label the data at hand.

All the J_t's are convex. They are intimately connected to the batch risk J:

Proposition 1 $J(f) = \frac{1}{T}\sum_{t=1}^{T} J_t(f)$.

Our online algorithm constructs a sequence of functions $f_1 \dots f_T$. Let $f_1 = 0$. The online algorithm simply performs a gradient descent step that aims to reduce the instantaneous risk in each iteration:

$$f_{t+1} = f_t - \eta_t \left.\frac{\partial J_t(f)}{\partial f}\right|_{f_t}. \quad (3)$$

The step size η_t needs to decay at a certain rate, e.g., $\eta_t = 1/\sqrt{t}$. Under mild conditions, this seemingly naïve online algorithm has a remarkable guarantee that on any input sequence, there is asymptotically "no regret" compared to the batch solution f^*. Specifically, let the *average instantaneous risk* incurred by the online algorithm be $J_{air}(T) \equiv \frac{1}{T}\sum_{t=1}^{T} J_t(f_t)$. Note J_{air} involves a varying

sequence of functions $f_1 \dots f_T$. As a standard quality measure in online learning, we compare J_{air} to the risk of the best *fixed* function in hindsight:

$$\begin{aligned} & J_{air}(T) - \min_f \frac{1}{T} \sum_{t=1}^{T} J_t(f) \\ = & J_{air}(T) - \min_f J(f) = J_{air}(T) - J(f^*), \end{aligned}$$

where we used Proposition 1. This difference is known as the average regret. Applying Theorem 1 in [2] results in the no-regret guarantee $\limsup_{T\to\infty} J_{air}(T) - J(f^*) \leq 0$. It is in this sense that the online algorithm performs as well as the batch algorithm on the sequence.

To compute (3) for manifold regularization, we first express the functions $f_1 \dots f_T$ using a common set of representers $x_1 \dots x_T$ [16]

$$f_t = \sum_{i=1}^{t-1} \alpha_i^{(t)} K(x_i, \cdot). \tag{4}$$

The problem of finding f_{t+1} becomes computing the coefficients $\alpha_1^{(t+1)}, \dots, \alpha_t^{(t+1)}$. Again, this will be a computational issue when T is large, and will be addressed in Section 3. We extend the kernel online supervised learning approach in [5] to semi-supervised learning by writing the gradient $\partial J_t(f)/\partial f$ in (3) as

$$\begin{aligned} & \frac{T}{l} \delta(y_t) c'(f(x_t), y_t) K(x_t, \cdot) + \lambda_1 f \\ & + 2\lambda_2 \sum_{i=1}^{t-1} (f(x_i) - f(x_t)) w_{it} (K(x_i, \cdot) - K(x_t, \cdot)), \end{aligned} \tag{5}$$

where we used the reproducing property of RKHS in computing the derivative: $\partial f(x)/\partial f = \partial \langle f, K(x, \cdot)\rangle / \partial f = K(x, \cdot)$. c' is the (sub)gradient of the loss function c. For example, when $c(f(x), y)$ is the hinge loss $\max(1 - f(x)y, 0)$, we may define $c'(f(x), y) = -y$ if $f(x)y \leq 1$, and 0 otherwise. Putting (5) back in (3), and replacing f_t with its kernel expansion (4), it can be shown that f_{t+1} has the following coefficients:

$$\begin{aligned} \alpha_i^{(t+1)} &= (1 - \eta_t \lambda_1) \alpha_i^{(t)} - 2\eta_t \lambda_2 (f_t(x_i) - f_t(x_t)) w_{it}, \quad i = 1 \dots t-1 \\ \alpha_t^{(t+1)} &= 2\eta_t \lambda_2 \sum_{i=1}^{t-1} (f_t(x_i) - f_t(x_t)) w_{it} - \eta_t \frac{T}{l} \delta(y_t) c'(f(x_t), y_t). \end{aligned} \tag{6}$$

We now have a basic online manifold regularization algorithm; see Algorithm 1.

When the data is *iid*, the generalization risk of the average function $\bar{f} = 1/T \sum_{t=1}^{T} f_t$ approaches that of f^* [17]. The average function $\bar{f}$ involves all representers $x_1, \dots, x_T$. For basic online manifold regularization, it is possible to incrementally maintain the exact $\bar{f}$ as time increases. However, for the sparse

Algorithm 1 Online Manifold Regularization

Parameters: edge weight function w, kernel K, weights λ_1, λ_2, loss function c, label ratio T/l, step sizes η_t
Initialize $t = 1, f_1 = 0$
loop
  receive x_t, predict $f_t(x_t)$ using (4)
  (occasionally) receive y_t
  update f_t to f_{t+1} using (6)
  store x_t, let $t = t + 1$
end loop

approximations introduced below, the basis changes over time. Therefore, in those cases $\bar{f}$ can be maintained only approximately using matching pursuit [18]. In our experiments, we compare the classification accuracy of $\bar{f}$ vs. f^* on a separate test set, which is of practical interest.

3 Sparse Approximations

Unfortunately, Algorithm 1 will not work in practice because it needs to store every input point and soon runs out of memory; it also has time complexity $O(T^2)$. In particular, the instantaneous risk (2) and the kernel representation (4) both involve the sequence up to the current time. To be useful, it is imperative to sparsify both terms. In this section, we present two distinct approaches for this purpose: i) using a small buffer of points, and ii) constructing a random projection tree that represents the manifold structure.

3.1 Buffering

Buffering (e.g., [19] and the references therein) keeps a limited number of points. Let the buffer size be τ. The simplest buffering strategy replaces the oldest point $x_{t-\tau}$ in the buffer with the incoming point x_t. With buffering, the approximate instantaneous risk is

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2 \frac{t}{\tau} \sum_{i=t-\tau}^{t-1} (f(x_i) - f(x_t))^2 w_{it}, \quad (7)$$

where the scaling factor t/τ keeps the magnitude of the graph regularizer comparable to the unbuffered version. In terms of manifold regularization, buffering corresponds to a dynamic graph on the points in the buffer. Similarly, the kernel expansion now has τ terms:

$$f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot).$$

With buffering, the function update involves two steps. In the first step, we update f_t to an intermediate function f' represented by a basis of $\tau+1$ elements, consisting of the old buffer and the new point x_t:

$$
\begin{aligned}
f' &= \sum_{i=t-\tau}^{t-1} \alpha'_i K(x_i, \cdot) + \alpha'_t K(x_t, \cdot) \\
\alpha'_i &= (1-\eta_t\lambda_1)\alpha_i^{(t)} - 2\eta_t\lambda_2(f_t(x_i) - f_t(x_t))w_{it}, \;\; i = t-\tau \ldots t-1 \\
\alpha'_t &= 2\eta_t\lambda_2 \frac{t}{\tau} \sum_{i=t-\tau}^{t-1} (f_t(x_i) - f_t(x_t))w_{it} - \eta_t \frac{T}{l}\delta(y_t)c'(f(x_t), y_t). \qquad (8)
\end{aligned}
$$

Second, we evict $x_{t-\tau}$ from the buffer, add x_t to the buffer, and approximate f' (which uses $\tau+1$ basis functions) with f_{t+1} (which uses τ basis functions):

$$
\min_{\alpha^{(t+1)}} \|f' - f_{t+1}\|^2 \quad \text{s.t. } f_{t+1} = \sum_{i=t-\tau+1}^{t} \alpha_i^{(t+1)} K(x_i, \cdot). \qquad (9)
$$

Intuitively, we "spread" $\alpha'_{t-\tau}K(x_{t-\tau}, \cdot)$ to the remaining points in the buffer, in an attempt to minimize the change caused by truncation. We use kernel matching pursuit [18] to efficiently find the approximate coefficients $\alpha^{(t+1)}$ in (9). Matching pursuit is a greedy function approximation scheme. It iteratively selects a basis function on which to spread the residual in $\alpha'_{t-\tau}K(x_{t-\tau}, \cdot)$. The number of steps (i.e., basis functions selected) can be controlled to trade-off approximation error and speed. We run matching pursuit until the norm of the residue vector has been sufficiently reduced. We call the above buffering strategy "**buffer**." The overall time complexity for buffering is $O(T)$.

An alternative buffering strategy, "**buffer-U**," evicts the oldest *unlabeled* points in the buffer while keeping labeled points. This is motivated by the fact that the labeled points tend to have larger α coefficients and exert more influence on our learned function. The oldest labeled point is evicted from the buffer only when it is filled with labeled points. Note this is distinct from batch learning: the labeled points only form a better basis, but learning is still done via gradient descent.

3.2 Random Projection Tree

Another way to improve Algorithm 1 is to construct a sparse representation of the manifold. While many embedding techniques exist, we require one that is fast and can be incrementally modified. Recently random projection has been proposed as an efficient means to preserve the manifold structure (see e.g., [20, 21]). We build our algorithm upon the online version of the Random Projection Tree (RPtree [22], Appendix I). An RPtree is a tree data structure with desirable theoretical properties that asymptotically traces the manifold. The basic idea is simple: as points arrive sequentially, they are spatially sorted into the RPtree leaves. When enough points fall into a leaf, the RPtree grows by splitting the

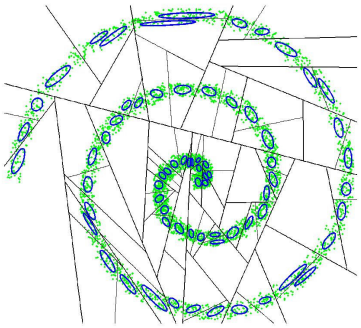


Fig. 1. A random projection tree on the Swiss roll data. Small dots represent data points, line segments represent the random splits in the internal nodes of the RPtree, polygons represent the regions governed by the leaves, and ellipses represent the Gaussian distributions on the data points within each leaf. We exploit the fact that these distributions follow the manifold structure of the data.

leaf along a hyperplane with random orientation. An RPtree can be regarded as an efficient online clustering algorithm whose clusters grow over time and cover the manifold, as shown in Figure 1. We refer the reader to [22] for details, while presenting our extensions for semi-supervised learning below.

Let $\{L_i\}_{i=1}^{s}, s \ll t$ denote the leaves in the RPtree at time t. To model the data points that have fallen into each leaf, we maintain a Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ at each leaf L_i, where μ_i and Σ_i are estimated *incrementally* as the data points arrive. We also keep track of n_i, the number of points in leaf L_i. With an RPtree, we approximate the kernel representation of f_t (4) by the means of the Gaussian distributions associated with the tree leaves: $f_t = \sum_{i=1}^{s} \beta_i^{(t)} K(\mu_i, \cdot)$. We approximate the instantaneous risk (2) by

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2 \sum_{i=1}^{s} n_i (f(\mu_i) - f(x_t))^2 w_{\mu_i t}. \quad (10)$$

From a graph regularization point of view, this can be understood as having a coarser graph over the RPtree leaves. We define the edge weight $w_{\mu_i t}$ between incoming point x_t and each leaf L_i to be

$$\begin{aligned} w_{\mu_i t} &= \mathbb{E}_{x \sim \mathcal{N}(\mu_i, \Sigma_i)} \left[\exp\left(-\frac{||x - x_t||^2}{2\sigma^2} \right) \right] \quad (11) \\ &= (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} |\Sigma_0|^{-\frac{1}{2}} |\tilde{\Sigma}_i|^{\frac{1}{2}} \\ &\quad \exp\left(-\frac{1}{2} \left(\mu_i^\top \Sigma_i^{-1} \mu_i + x_t^\top \Sigma_0^{-1} x_t - \tilde{\mu}_i^\top \tilde{\Sigma}_i \tilde{\mu}_i \right) \right), \end{aligned}$$

where $\Sigma_0 = \sigma^2 I$, $\tilde{\Sigma}_i = (\Sigma_i^{-1} + \Sigma_0^{-1})^{-1}$, $\tilde{\mu}_i = \Sigma_i^{-1}\mu_i + \Sigma_0^{-1} x_t$, and σ is the bandwidth of the (original point to point) weight. We call this weight scheme

"**RPtree PPK**" for its similarity to the probability product kernel [23]. An even simpler approximation is to ignore the covariance structure by defining $w_{\mu_i t} = e^{-\|\mu_i - x_t\|^2/2\sigma^2}$. It has computational advantages at the price of precision. We call this weight scheme "**RPtree**."

With an RPtree, the function update occurs in three steps. As space precludes a detailed discussion, we present an outline here. In the first step, upon receiving x_t, we update f_t to an intermediate function f' using a basis of $s+1$ elements: $\mu_1, \ldots, \mu_s$ and x_t. This is similar to (8) in the buffering case. In the second step, the RPtree itself is adjusted to account for the addition of x_t. The adjustments include updating the Gaussian parameters for the leaf x_t falls into, and potentially splitting the leaf. In the latter case, the number of leaves s will increase to s', and each new leaf's mean and covariance statistics are established. In the third step, we approximate f' by f_{t+1} using the s' new basis elements $\mu_1, \ldots, \mu_{s'}$ ($s' = s$ if no split happened), similar to (9). The point x_t is then discarded.

4 Experiments

We present a series of experimental results as empirical evidence that online manifold regularization (MR) is a viable option for performing fast MR on large data sets. We summarize our findings as follows:

1. Online MR scales better than batch MR in time and space. Although recent advances in manifold regularization greatly improve the feasible problem size (e.g., [24]), we believe that it takes online learning to handle unlimited input sequences and achieve life-long learning.
2. Online MR achieves comparable performance to batch MR. This is measured by two criteria:
   (a) $J_{air}(T)$ approaches $J(f^*)$, both for the basic online MR algorithm, as well as for the buffering and RPtree approximations.
   (b) Generalization error of $\bar{f}$ approaches that of f^* on test sets.
3. Online MR can handle concept drift (changes in $P(x)$ and $P(y|x)$). The online method (using a limited size buffer) can track a non-stationary distribution and maintain good generalization accuracy, while the batch method trained on all previous data fails to do so.

Our focus is on comparing online MR to batch MR, not semi-supervised learning to supervised learning. It is known that semi-supervised learning does not necessarily outperform supervised learning, depending on the correctness of model assumptions. Thus, our experiments use tasks where batch MR has proven beneficial in prior work, and we demonstrate that online MR provides a useful alternative to batch MR on these tasks.

4.1 Data Sets and Protocol

We report results on three data sets. The first is a toy two-spirals data set. The training sequences and test sets (of size 2000) are generated *iid*. The second is

the MNIST digit classification data set [25], and we focus on two binary tasks: 0 vs. 1 and 1 vs. 2. We scaled down the images to 16 x 16 pixels (256 features). The training sequences are randomly shuffled subsets of the official training sets, and we use the official test sets (of size 2115 for 0 vs. 1, and 2167 for 1 vs. 2). The third is the 361-dimensional Extended MIT face vs. non-face image classification data set ("Face") [26]. We sampled a balanced subset of the data, and split this into a training set and a test set. The same test set of size 2000 is used in all experiments, while different training runs use different randomly shuffled subsets of the training set. The labeled rate p_l is 0.02 in all experiments, with points assigned to each class with equal probability.

Our experimental protocol is the following:

1. Generate randomly ordered training sequences and test sets (for MNIST and Face, the test sets are already given).
2. For batch MR, train separate versions on increasing subsequences (i.e., $T = 500, 1000, 2000, \ldots$).
3. For online MR, train once on the entire sequence.
4. For each T, compare the corresponding batch MR f^* with the online classifier trained up to T.

All results are the average of five such trials. The error bars are ± 1 standard deviation.

All experiments use hinge loss c and RBF kernel K. The kernel bandwidth parameter σ_K, λ_1, λ_2, and the edge weight parameter σ were all tuned for batch MR using $T = 500$. When using a limited size buffer, we set $\tau = 300$, and only require that matching pursuit reduce the residue norm by 50%. We use a step size of $\eta_t = \gamma/\sqrt{t}$, where $\gamma = 0.03$ for the RPtree approximation, and 0.1 for all other methods. We implemented all methods using MATLAB and CPLEX.

4.2 Online MR Scales Better than Batch MR

We illustrate this point by comparing runtime growth curves on the spirals and MNIST 0 vs. 1 data sets. Figure 2(left) shows that, for the spirals data set, the growth rates of batch MR and basic online MR are quadratic as expected (in fact, online MR has more overhead in our MATLAB implementation). Batch MR runs out of memory after $T = 5000$, and we stop basic online MR at $T = 4000$ because the runtime becomes excessive. On the other hand, online MR (buffered) and online RPtree are linear. Though not included in the plot, online RPtree PPK has a curve nearly identical to online MR (buffered). Figure 2(right) demonstrates similar trends for the higher dimensional MNIST 0 vs. 1 data set.

4.3 Online MR Achieves Comparable Risks

We compare online MR's average instantaneous risk $J_{air}(T)$ vs. batch MR's risk $J(f^*)$ on the training sequence. Our experiments support the theory that $J_{air}(T)$

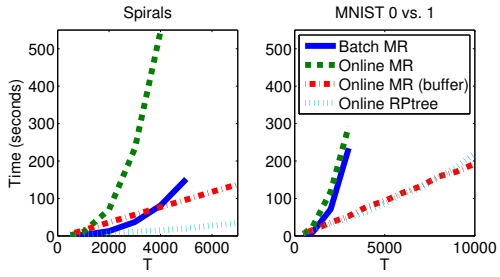


Fig. 2. Runtime growth curves. Batch MR and basic online MR scale quadratically, while the sparse approximations of buffering and RPtree scale only linearly.

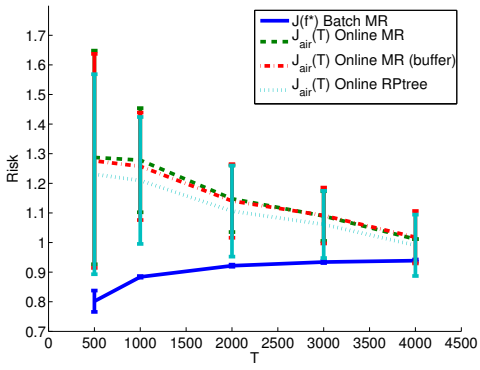


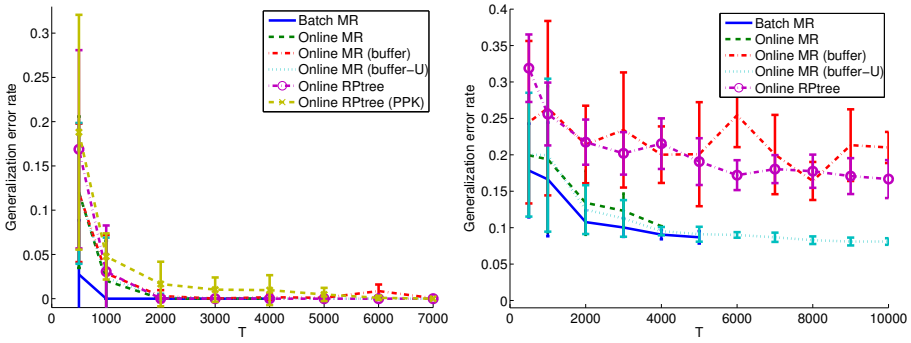
Fig. 3. Online MR's average instantaneous risk $J_{air}(T)$ approaches batch MR's risk $J(f^*)$ as T increases.

converges to $J(f^*)$ as T increases. [3] Figure 3 compares these measures for basic online MR and batch MR on the spirals data set. The two curves approach each other. $J_{air}(T)$ continues to decrease beyond $T = 4000$ (not pictured). Figure 3 also shows that online MR (buffer) and online RPtree are good approximations to basic online MR in terms of J_{air}.

4.4 Generalization Error of Online MR

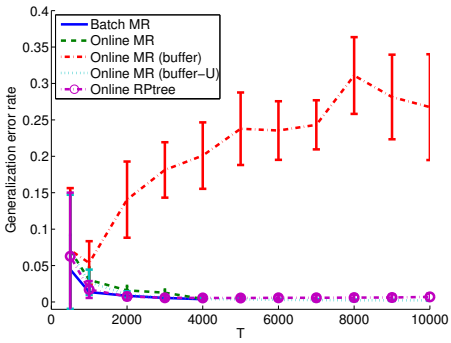
The experiments in this section compare the averaged function $\bar{f}$ of online MR and the batch solution f^* in terms of generalization error on test sets. Figure 4 presents results for all the data sets. We observe that online MR buffer-U is the best and consistently achieves test accuracy that is comparable to batch MR.

[3] While the average regret approaches zero asymptotically, the step size of $\eta_t = 1/\sqrt{t}$ decays rapidly, potentially leading to slow convergence. Thus, it is possible that long sequences (i.e., large T values) would be required for the online algorithm to compete with the best batch algorithm. Nevertheless, our experiments show this is not actually a problem in practice.

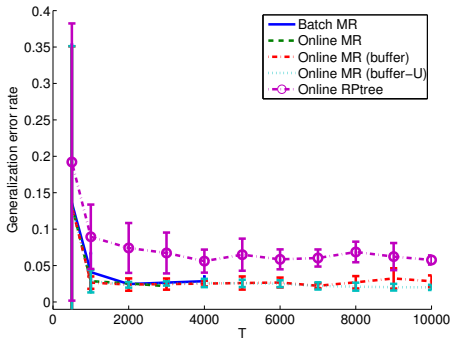


(a) Spirals

(b) Face



(c) MNIST 0 vs. 1



(d) MNIST 1 vs. 2

Fig. 4. Generalization error of batch MR's f^* and online MR's $\bar{f}$ as T increases. Online MR buffer-U consistently achieves test accuracy comparable to batch MR.

From Figure 4(a), we observe that, for the spirals data set, all the online methods perform nearly as well as batch MR. As is to be expected, batch MR makes the most efficient use of the data and reaches 0 test error first, while the online methods require only a little additional data to reach this level (after all, standard incremental learning usually needs multiple passes over the training set). Buffering and RPtree perform as well as basic online MR, showing little sign of approximation error. Panels (b), (c), and (d) in Figure 4 show that buffer-U can be much better than buffer. This is understandable, since matching pursuit may provide a poor approximation to the contributions of the discarded data point. In high dimensional space, there may be few similar data points remaining in the small buffer, so much of the weight assigned to discarded points is lost. Under the buffer-U strategy, we alleviate this issue by preserving the larger weights on labeled points, which approximate the function better. RPtree PPK on these high dimensional data sets involves expensive inversion of (often singular) covariance matrices and is not included in the comparison. The performance of RPtree is no better than buffer-U.

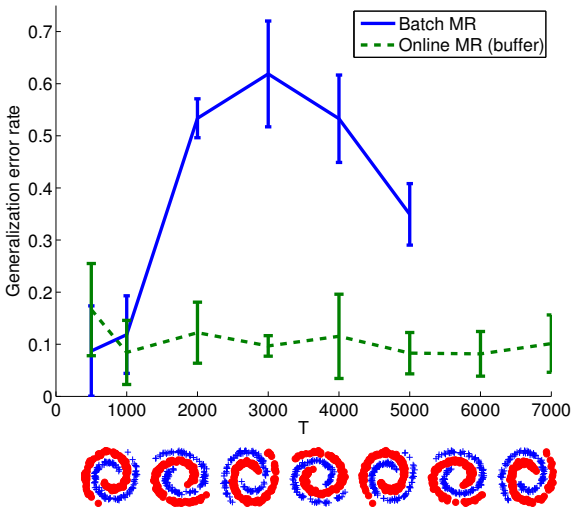


Fig. 5. Online MR (buffer) has much better generalization error than batch MR when faced with concept drift in the rotating spirals data set.

4.5 Online MR Handles Concept Drift

Lastly, we demonstrate that online MR can handle concept drift. When the underlying distributions, both $P(x)$ and $P(y|x)$, change during the course of learning, using buffered online MR is extremely advantageous. For this experiment, we "spin" the two spirals data set so that the spirals smoothly rotate 360° in every 4000 points (Figure 5). All points in the space will thus change their true labels during the sequence. We still provide only 2% of the labels to the algorithms. The test set for a given T consists of 2000 points drawn from the current underlying distribution.

For this experiment, we show the generalization error of batch MR's f^* vs. online MR (buffer)'s f_T, since the latest function is expected to track the changes in the data. Figure 5 illustrates that online MR (buffer) is able to adapt to the changing sequence and maintain a small error rate. In contrast, batch MR uses all data points, which now tend to conflict heavily (i.e., newer data from one class overlaps with older data from the other class). As expected, the single batch classifier f^* is inadequate for predicting such changing data.

5 Conclusions

We presented an online semi-supervised learning algorithm that parallels manifold regularization. Our algorithm is based on online convex programming in RKHS. We proposed two sparse approximations using buffering and online random projection trees to make online MR practical. The original batch manifold regularization algorithm has time complexity at least $O(T^2)$; so does the online

version without sparse approximation. In contrast, the RPtree approximation has complexity $O(T \log T)$, where each iteration requires $O(\log T)$ leaf lookups (the tree's height is $O(\log T)$ because each leaf contains a constant maximum number of points). Buffering has complexity $O(T)$. Experiments show that our online MR algorithm has risk and generalization error comparable to batch MR, but scales much better. In particular, online MR (buffer-U) tends to have the best performance.

There are many interesting questions remaining in this online semi-supervised learning setting. Future work will proceed along two directions. On the empirical side, we will further speed up online MR, for example by using fast neighbor search to reduce the number of candidate basis elements in matching pursuit. We also plan to study practical online algorithms for other semi-supervised learning methods, in particular those with non-convex risks like S3VMs. On the theoretical side, we plan to investigate different regret notions that might be appropriate for this setting, performance guarantees with concept drift, and models that do not require all previous points.

Acknowledgements

A. Goldberg and X. Zhu were supported in part by the Wisconsin Alumni Research Foundation. This work was completed while M. Li was a visiting researcher at University of Wisconsin-Madison under a State Scholarship from the Chinese Scholarship Council. The authors also thank Shuchi Chawla for helpful discussions on online learning.

References

1. Brefeld, U., Büscher, C., Scheffer, T.: Multiview discriminative sequential learning. In: European Conference on Machine Learning (ECML). (2005)
2. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: ICML'03. (2003)
3. Chapelle, O., Zien, A., Schölkopf, B., eds.: Semi-supervised learning. MIT Press (2006)
4. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison (2005)
5. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Transactions on Signal Processing **52**(8) (2004) 2165–2176
6. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press (2002)
7. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research **7** (2006) 2399–2434
8. Sindhwani, V., Niyogi, P., Belkin, M.: Beyond the point cloud: from transductive to semi-supervised learning. In: ICML'05. (2005)
9. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML'03. (2003)
10. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT'98. (1998)

11. Sindhwani, V., Niyogi, P., Belkin, M.: A co-regularized approach to semi-supervised learning with multiple views. In: ICML'05. (2005)
12. Brefeld, U., Gaertner, T., Scheffer, T., Wrobel, S.: Efficient co-regularized least squares regression. In: ICML'06. (2006)
13. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML'99. (1999)
14. Chapelle, O., Sindhwani, V., Keerthi, S.S.: Branch and bound for semi-supervised support vector machines. In: NIPS'06. (2006)
15. Collobert, R., Sinz, F., Weston, J., Bottou, L.: Large scale transductive SVMs. The Journal of Machine Learning Research **7**(Aug) (2006) 1687–1712
16. Kimeldorf, G., Wahba, G.: Some results on Tchebychean spline functions. Journal of Mathematics Analysis and Applications **33** (1971) 82–95
17. Cesa-Bianchi, N., Conconi, A., Gentile, C.: On the generalization ability of on-line learning algorithms. IEEE Transactions on Information Theory **50**(9) (2004) 2050–2057
18. Vincent, P., Bengio, Y.: Kernel matching pursuit. Machine Learning **48**(1-3) (2002) 165–187
19. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The forgetron: A kernel-based perceptron on a fixed budget. In: NIPS'05. (2005)
20. Hegde, C., Wakin, M., Baraniuk, R.: Random projections for manifold learning. In: NIPS'07. (2007)
21. Freund, Y., Dasgupta, S., Kabra, M., Verma, N.: Learning the structure of manifolds using random projections. In: NIPS'07. (2007)
22. Dasgupta, S., Freund, Y.: Random projection trees and low dimensional manifolds. Technical Report CS2007-0890, University of California, San Diego (2007)
23. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. Journal of Machine Learning Research, Special Topic on Learning Theory **5** (2004) 819–844
24. Tsang, I., Kwok, J.: Large-scale sparsified manifold regularization. In: NIPS'06. (2006)
25. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (November 1998) 2278–2324
26. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast svm training on very large data sets. Journal of Machine Learning Research **6** (2005) 363–392