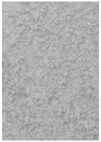
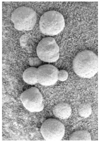
Online Manifold Regularization: A New Learning Setting and Empirical Study

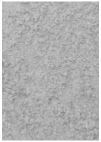
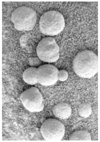
Andrew B. Goldberg[1], Ming Li[2], Xiaojin Zhu[1]

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

Life-long learning

x_1	x_2	$\dots$	x_{1000}	$\dots$	$x_{1000000}$	$\dots$
		$\dots$		$\dots$		$\dots$
$y_1 = 0$	n/a	$\dots$	$y_{1000} = 1$	$\dots$	$y_{1000000} = 0$	$\dots$

Life-long learning

x_1	x_2	$\dots$	x_{1000}	$\dots$	$x_{1000000}$	$\dots$
		$\dots$		$\dots$		$\dots$
$y_1 = 0$	n/a	$\dots$	$y_{1000} = 1$	$\dots$	$y_{1000000} = 0$	$\dots$

Unlike standard supervised learning:

- $n \to \infty$ examples arrive sequentially, cannot even store them all
- most examples are unlabeled
- no iid assumption, $p(x, y)$ can change over time

This is how children learn, too

x_1	x_2	$\dots$	x_{1000}	$\dots$	$x_{1000000}$	$\dots$
		$\dots$		$\dots$		$\dots$
$y_1 = 0$	n/a	$\dots$	$y_{1000} = 1$	$\dots$	$y_{1000000} = 0$	$\dots$

New paradigm: online semi-supervised learning

Main contribution: merging learning settings

1. Online: learn non-iid sequentially, but fully labeled
2. Semi-supervised: learn from iid batch, (mostly) unlabeled

1. At time t, adversary picks $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$ not necessarily iid, shows x_t
2. Learner has classifier $f_t : \mathcal{X} \mapsto \mathbb{R}$, predicts $f_t(x_t)$
3. With small probability, adversary reveals y_t; otherwise it abstains (unlabeled)
4. Learner updates to f_{t+1} based on x_t and y_t (if given). Repeat.

New paradigm: online semi-supervised learning

Main contribution: merging learning settings

1. Online: learn non-iid sequentially, but fully labeled
2. Semi-supervised: learn from iid batch, (mostly) unlabeled

1. At time t, adversary picks $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$ not necessarily iid, shows x_t
2. Learner has classifier $f_t : \mathcal{X} \mapsto \mathbb{R}$, predicts $f_t(x_t)$
3. With small probability, adversary reveals y_t; otherwise it abstains (unlabeled)
4. Learner updates to f_{t+1} based on x_t and y_t (if given). Repeat.

Many batch SSL algorithms exist; we focus on manifold regularization

Review: batch manifold regularization

A form of graph-based semi-supervised learning [Belkin et al. JMLR06]:

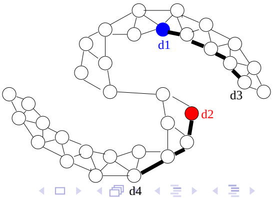
- Graph on $x_1 \dots x_n$
- Edge weights w_{st} encode similarity between x_s, x_t, e.g., kNN
- Assumption: similar examples have similar labels

Manifold regularization minimizes risk:

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

$c(f(x), y)$ convex loss function, e.g., the hinge loss.
Solution $f^* = \arg\min_f J(f)$.

Generalizes graph mincut and label propagation.



From batch to online

batch risk = average instantaneous risks

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

Batch risk

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

Instantaneous risk

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

(includes graph edges between x_t and all previous examples)

Online convex programming

Instead of minimizing convex $J(f)$, reduce convex $J_t(f)$ at each step t.

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

Remarkable no regret guarantee against adversary:

- Note: accuracy can be arbitrarily bad if adversary flips target often
- If so, no batch learner in hindsight can do well either

$$\text{regret} \equiv \frac{1}{T} \sum_{t=1}^{T} J_t(f_t) - J(f^*)$$

[Zinkevich ICML03] No regret: $\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} J_t(f_t) - J(f^*) \leq 0$.

If no adversary (iid), the average classifier $\bar{f} = 1/T \sum_{t=1}^{T} f_t$ is good: $J(\bar{f}) \to J(f^*)$.

Sparse approximation 1: buffering

The algorithm is impractical

- Space $O(T)$: stores all previous examples
- Time $O(T^2)$: each new example compared to all previous ones
- In reality, $T \to \infty$

Approximation: Keep a size τ buffer

- Approximate representers: $f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot)$
- Approximate instantaneous risk

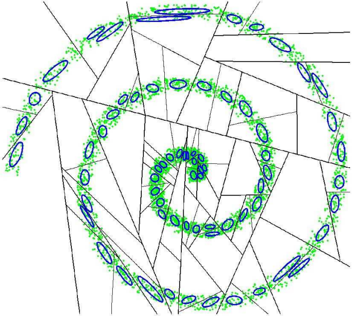
$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2 \frac{t}{\tau} \sum_{i=t-\tau}^{t} (f(x_i) - f(x_t))^2 w_{it}$$

- Dynamic graph on examples in the buffer

Sparse approximation 2: random projection tree

[Dasgupta and Freund, STOC08]

- Discretize data manifold by online clustering.
- When a cluster accumulates enough examples, split along random hyperplane.
- Extends k-d tree.
- Approximation uses a graph over clusters (represented by Gaussians)



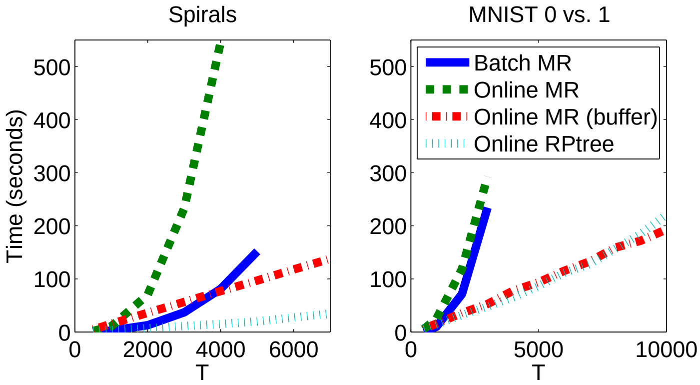
Experimental Results

Compare batch manifold regularization (MR) with several online variants (i.e., full graph, buffering, random projection tree)

- Runtime
- Risk (to assess no regret guarantee)
- Generalization error using average classifier (assumes iid)

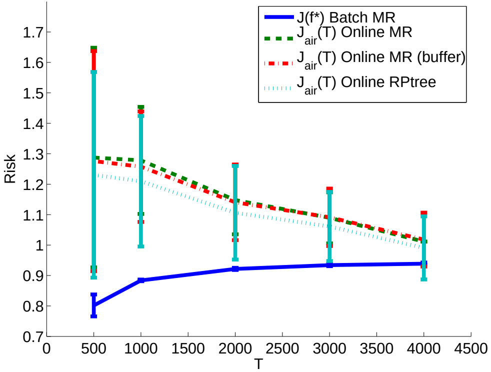
Experiment: runtime

Buffering and RPtree scale linearly, enabling life-long learning.



Experiment: risk

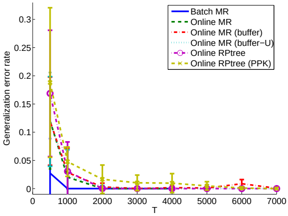
Online MR risk $J_{air}(T) \equiv \frac{1}{T}\sum_{t=1}^{T} J_t(f_t)$ approaches batch risk $J(f^*)$ as T increases.



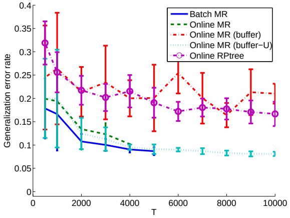
Experiment: generalization error of $\bar{f}$ if iid

A variation of buffering as good as batch MR
(preferentially keep labeled examples in buffer).

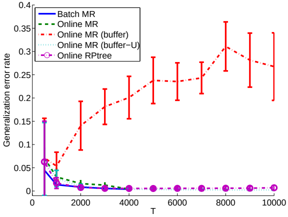
(a) Spirals



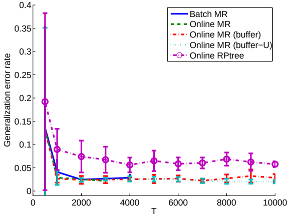
(b) Face



(c) MNIST 0 vs. 1



(d) MNIST 1 vs. 2



Conclusions

- Online semi-supervised learning framework
- Sparse approximations: buffering and RPtree
- Future work: new bounds, new algorithms (e.g., S3VM)

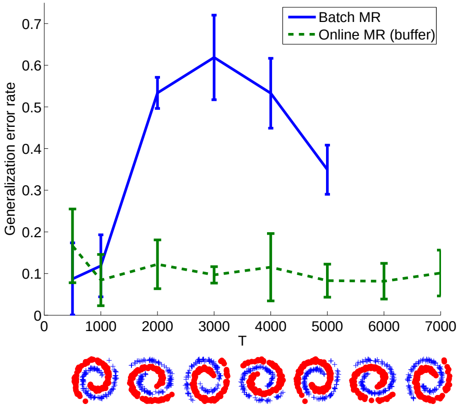
Conclusions

- Online semi-supervised learning framework
- Sparse approximations: buffering and RPtree
- Future work: new bounds, new algorithms (e.g., S3VM)

Thank you! Any questions?

Experiment: adversarial concept drift

- Slowly rotating spirals, both $p(x)$ and $p(y|x)$ changing.
- Batch f^* vs. online MR buffering f_T
- Test set drawn from the current $p(x, y)$ at time T.



Kernelized algorithm

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

- Init: $t = 1, f_1 = 0$
- Repeat
  1. receive x_t, predict $f_t(x_t) = \sum_{i=1}^{t-1} \alpha_i^{(t)} K(x_i, x_t)$
  2. occasionally receive y_t
  3. update f_t to f_{t+1} by adjusting coefficients:

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

  4. store x_t, let $t = t + 1$

Sparse approximation 1: buffer update

- At each step, start with the current τ representers:

$$f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot) + 0K(x_t, \cdot)$$

- Gradient descent on $\tau + 1$ terms:

$$f' = \sum_{i=t-\tau}^{t} \alpha'_i K(x_i, \cdot)$$

- Reduce to τ representers $f_{t+1} = \sum_{i=t-\tau+1}^{t} \alpha_i^{(t+1)} K(x_i, \cdot)$ by

$$\min_{\alpha^{(t+1)}} \|f' - f_{t+1}\|^2$$

- Kernel matching pursuit

Sparse approximation 2: random projection tree

We use the clusters $\mathcal{N}(\mu_i, \Sigma_i)$ as representers:

$$f_t = \sum_{i=1}^{s} \beta_i^{(t)} K(\mu_i, \cdot)$$

"Cluster graph" edge weight between a cluster μ_i and example x_t is

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

A further approximation is

$$w_{\mu_i t} = e^{-\|\mu_i - x_t\|^2 / 2\sigma^2}$$

Update f (i.e., β) and the RPtree, discard x_t.