732A95/TDDE01 Introduction to Machine Learning

Lecture 1c Block 2: Online Learning

Jose M. Peña
IDA, Linköping University, Sweden

Contents

Literature

- Main sources
  - Murphy, K. P. *Machine Learning - A Probabilistic Perspective*. MIT Press, 2012. Sections 8.5 and 11.4.8.
  - Shalev-Shwartz, S. Online Learning and Online Convex Optimization. *Foundations and Trends in Machine Learning*, 4:107-194, 2011. Sections 1-1.2 and 3-3.1.
- Additional sources
  - Blum, A. On-Line Algorithms in Machine Learning. In *Online Algorithms: The State of the Art*. Springer, 1998.
  - Bordes, A. et al. Fast Kernel Classifiers with Online and Active Learning. *Journal of Machine Learning Research*, 6:1579-1619, 2005.

Online Learning

- Offline learning: Batch of data used to optimize a function such as

$$\frac{1}{N}\sum_{n} L(t_n, y(\boldsymbol{x}_n, \boldsymbol{\theta}))$$

  where $L(\cdot)$ is log loss, squared error, 0/1 loss, etc.

- Online learning: Streaming data so we cannot wait until the end to start processing the data. Instead, we update our estimate with each new point's arrival. It may even be an interesting option if the batch of data does not fit in main memory.

Online learning
for $n = 1, 2, \ldots$
Receive question $\boldsymbol{x}_n$
Predict $y(\boldsymbol{x}_n, \boldsymbol{\theta}_n)$
Receive true answer t_n
Suffer loss $L(t_n, y(\boldsymbol{x}_n, \boldsymbol{\theta}_n))$

- Examples: Weather prediction, spam filtering.
- Goal: Minimize the cumulative loss suffered along the run.

Online Learning

- Assume that $|\Theta| < \infty$ and $t_n = y(\boldsymbol{x}_n, \boldsymbol{\theta}^*)$ for all n with $\boldsymbol{\theta}^* \in \Theta$.

Consistent

$\Theta_1 = \Theta$
for $n = 1, 2, \ldots$
  Receive question $\boldsymbol{x}_n$
  Choose any $\boldsymbol{\theta}_n \in \Theta_n$
  Predict $y(\boldsymbol{x}_n, \boldsymbol{\theta}_n)$
  Receive true answer t_n
  Update $\Theta_{n+1} = \{\boldsymbol{\theta} \in \Theta_n : y(\boldsymbol{x}_n, \boldsymbol{\theta}) = t_n\}$

- The algorithm above makes at most $|\Theta| - 1$ errors.

Halving

$\Theta_1 = \Theta$
for $n = 1, 2, \ldots$
  Receive question $\boldsymbol{x}_n$
  Predict $\arg\max_{r \in \{0,1\}} |\{\boldsymbol{\theta} \in \Theta_n : y(\boldsymbol{x}_n, \boldsymbol{\theta}) = r\}|$
  Receive true answer t_n
  Update $\Theta_{n+1} = \{\boldsymbol{\theta} \in \Theta_n : y(\boldsymbol{x}_n, \boldsymbol{\theta}) = t_n\}$

- The algorithm above makes at most $\log_2 |\Theta|$ errors.

Online Learning

- If we drop the assumption that $t_n = y(\boldsymbol{x}_n, \boldsymbol{\theta}^*)$ for all n with $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}$, then our goal may be restated as minimizing the regret

$$\frac{1}{N}\sum_n L(t_n, y(\boldsymbol{x}_n, \boldsymbol{\theta}_n)) - \min_{\boldsymbol{\theta}^* \in \boldsymbol{\Theta}} \frac{1}{N}\sum_n L(t_n, y(\boldsymbol{x}_n, \boldsymbol{\theta}^*))$$

- Note that the second term above is the loss of the batch solution.
- We may also be satisfied if we find a predictor whose regret grows sub-linearly with respect to N.
- These goals seem hopeless anyway, as the adversary may wait for our prediction and then answer the opposite label ($regret = N - N/2$ at least, e.g. if the batch solution is majority voting).
- Solution: Output a posterior class distribution or equivalent. Although the adversary knows that we will select a label according to this distribution, his/her power will be limited.

Online Learning

Weighted majority (input: $\eta \in (0,1)$)

$\boldsymbol{w}_1 = (1/M, \ldots, 1/M)$ where $M = |\boldsymbol{\Theta}|$
for $n = 1, 2, \ldots$
  Choose $m \sim \boldsymbol{w}_n$ and predict $y(\boldsymbol{x}_n, \boldsymbol{\theta}_m)$
  Receive costs of all models $\boldsymbol{c}_n \in [0,1]^M$
  Update $w_{n+1i} = \frac{w_{ni} exp(-\eta c_{ni})}{\sum_j w_{nj} exp(-\eta c_{nj})}$ for all i

- The name of the algorithm is due to the fact that it predicts the label 1 with probability

$$p_n = \sum_m w_{nm} y(\boldsymbol{x}_n, \boldsymbol{\theta}_m)$$

- The loss can be written as

$$|p_n - t_n| = |\sum_m w_{nm} y(\boldsymbol{x}_n, \boldsymbol{\theta}_m) - t_n| = \sum_m w_{nm} |y(\boldsymbol{x}_n, \boldsymbol{\theta}_m) - t_n|$$

- Moreover, if $\eta = \sqrt{N \log M}$ then

$$\sum_n |p_n - t_n| \leq \min_{1 \leq m \leq M} \sum_n c_{nm} + 2\sqrt{N \log M}$$

  which implies a sub-linear regret wrt N, i.e. $\frac{regret}{N} \to 0$ as $N \to \infty$.
- Moreover, if $\eta = 1/2$ then

$$\sum_n |p_n - t_n| \leq 2 \min_{1 \leq m \leq M} \sum_n c_{nm} + 4 \log M$$

  which reduces to $4 \log M$ for the realizable case (similar to Halving).

Online Learning

- If we drop the assumption that $|\Theta| < \infty$, minimizing the regret may be achieved (!?) by online/stochastic/sequential gradient descent, which updates the parameters as

$$\boldsymbol{\theta}_{n+1} = proj_{\Theta}(\boldsymbol{\theta}_n - \eta_n \nabla L(t_n, y(\boldsymbol{x}_n, \boldsymbol{\theta}_n)))$$

  where $proj_{\Theta}(\boldsymbol{v}) = \arg\min_{\boldsymbol{\theta} \in \Theta} ||\boldsymbol{v} - \boldsymbol{\theta}||$, and it is needed only if Θ is a subset of $\mathbb{R}^D$.
- To ensure that stochastic gradient descent converges, we need to check that

$$\sum_{n=1}^{\infty} \eta_n = \infty \text{ and } \sum_{n=1}^{\infty} \eta_n^2 < \infty$$

  e.g. $\eta_n = 1/n$.
- Convergence vs adaptation or concept drift.
- Minimizing the regret implies looking at the past. Instead, we may want to minimize the expected loss in the future. Then, we want to minimize

$$\mathbb{E}_{\{\boldsymbol{x},T\}}[L(t, y(\boldsymbol{x}, \boldsymbol{\theta})]$$

- One solutions is to use a running average:

$$\bar{\boldsymbol{\theta}}_n = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\theta}_i$$

Online Kernel Methods

- Kernel classifier:

$$y(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \sum_n \mathbf{1}_{\{t_n=1\}} k\left(\frac{\boldsymbol{x}-\boldsymbol{x}_n}{h}\right) \leq \sum_n \mathbf{1}_{\{t_n=0\}} k\left(\frac{\boldsymbol{x}-\boldsymbol{x}_n}{h}\right) \\ 1 & \text{otherwise} \end{cases}$$

- Support vector machines:

$$y(\boldsymbol{x}) = \sum_{m \in \mathcal{S}} a_m t_m k(\boldsymbol{x}, \boldsymbol{x}_m) + b$$

- The kernel classifier implies no learning and, thus, it can be applied to online learning directly. Deploying it may however become computationally demanding at some point, since it sums over all the training points.
- Training a support vector machine need to be adapted for online learning, i.e. retraining is not an option as the training data may grow very fast. Deploying it is feasible due to its sparseness.

Online Kernel Methods

	Online SVM
1	$\mathcal{S} = \varnothing$
2	$b = 0$
3	Receive a random example $(\boldsymbol{x}_i, t_i)$
4	Compute $y(\boldsymbol{x}_i) = \sum_{m \in \mathcal{S}} a_m t_m k(\boldsymbol{x}_i, \boldsymbol{x}_m) + b$
5	If $t_i y(\boldsymbol{x}_i) \leq 0$ then
6	$\quad \mathcal{S} = \mathcal{S} \cup \{i\}$
7	$\quad a_i = 1$
8	Go to step 3

- It converges to a separating hyperplane in the separable case.
- It does not attempt to maximize the margin.
- It only adds support vectors.
- Therefore, it may overfit the training data (and increase computational cost for deploying it).

Online Kernel Methods

Budget online SVM (input: β and M)

1 $\mathcal{S} = \varnothing$
2 $b = 0$
3 Receive a random example $(\boldsymbol{x}_i, t_i)$
4 Compute $y(\boldsymbol{x}_i) = \sum_{m \in \mathcal{S}} a_m t_m k(\boldsymbol{x}_i, \boldsymbol{x}_m) + b$
5 If $t_i y(\boldsymbol{x}_i) \leq \beta$ then
6 $\quad \mathcal{S} = \mathcal{S} \cup \{i\}$
7 $\quad a_i = 1$
8 $\quad$ If $|\mathcal{S}| > M$ then $\mathcal{S} = \mathcal{S} \setminus \{\arg\max_{m \in \mathcal{S}} t_m (y(\boldsymbol{x}_m) - a_m t_m k(\boldsymbol{x}_m, \boldsymbol{x}_m))\}$
9 Go to step 3

- **Quiz**: Why does the removal criterion in step 8 make sense ?
- It tolerates mild noisy data but does not maximize the margin.
- It may still overfit the training data (and increase computational cost for deploying it) due to its myopic nature.
- More sophisticated addition and removal criteria are needed, e.g. LASVM which handles noisy data (slack variables) and converges to the batch solution.

Online Kernel Methods

- Experimental results for binary classification.
- Gaussian kernel, which implies dot product of 784 gray level pixel values.
- LIBSVM: Batch SVM.
- LASVM (x1/x2): Online SVM with one or two passes over the training data.

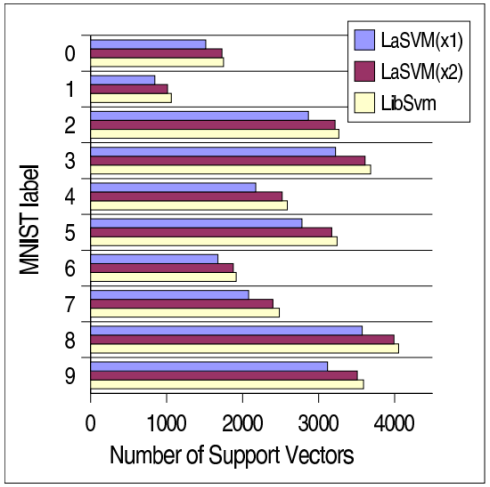

Figure 5: Compared numbers of support vectors for the ten MNIST binary classifiers.

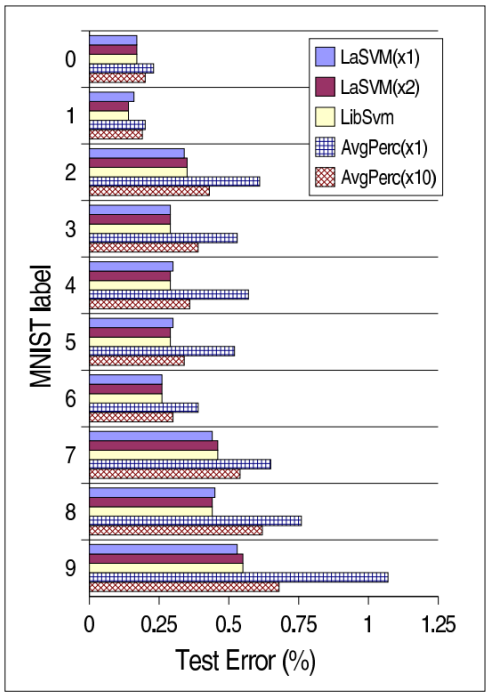

Figure 1: Compared test error rates for the ten MNIST binary classifiers.

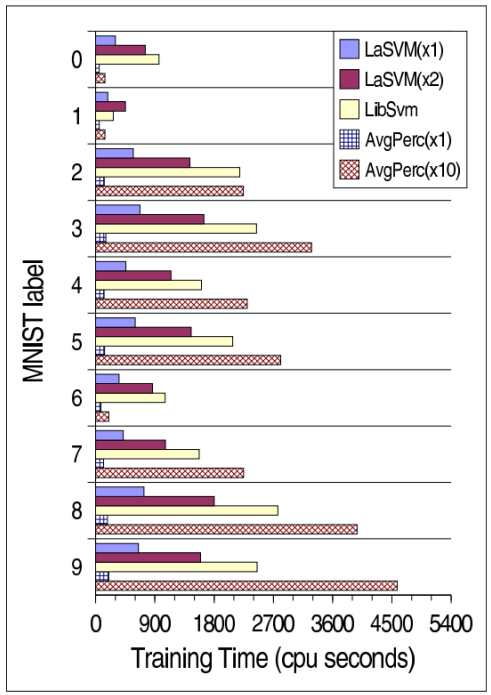

Figure 2: Compared training times for the ten MNIST binary classifiers.

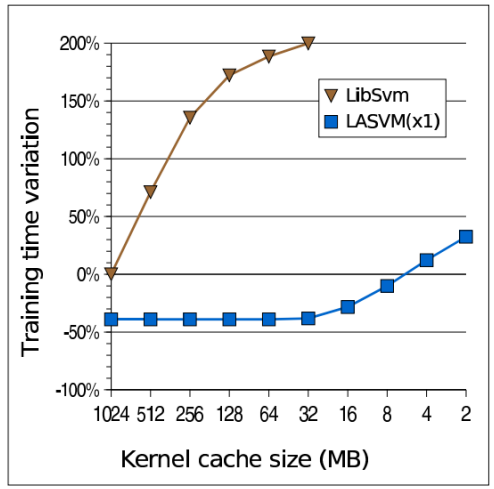

Figure 6: Training time variation as a function of the cache size. Relative changes with respect to the 1GB LIBSVM times are averaged over all ten MNIST classifiers.

Online EM Algorithm

EM algorithm

Set $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ to some initial values
Repeat until $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ do not change
  Compute $p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})$ for all n /* E step */
  Set π_k to π_k^{ML}, and μ_{ki} to μ_{ki}^{ML} for all k and i /* M step */

Mixture of Bernouillis	Mixture of Gaussians
$\pi_k^{ML} = \frac{\sum_n p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{N}$	$\pi_k^{ML} = \frac{\sum_n p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{N}$
$\mu_{ki}^{ML} = \frac{\sum_n x_{ni} p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{\sum_n p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}$	$\boldsymbol{\mu}_k^{ML} = \frac{\sum_n \boldsymbol{x}_n p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{\sum_n p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}$
	$\boldsymbol{\Sigma}_k^{ML} = \frac{\sum_n (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})^T p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{\sum_n p(z_{nk}\|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}$

- Online EM algorithm: Replace the E step with a stochastic E step such as

$$S_{1:n} = (1 - \eta_n) S_{1:n-1} + \eta_n \mathbb{E}_{\boldsymbol{Z}}[S_n | \boldsymbol{\mu}_n, \boldsymbol{\pi}_n]$$

  where $\{\eta_n\}$ is the learning rate, e.g. $\eta_n = 1/n$. Similarly for the mixture of Gaussians.
- Convergent under mild conditions. Adaptative too, e.g. $\eta_n = 1/2$.

Cross-validation

Quiz. What does cross-validation look like for online learning methods ?

Summary

- New concepts: Adversarial environment, regret.
- New algorithms:
  - Realizable and finite class of models: Consistent and halving.
  - Finite class of models: Weighted majority.
  - Online SVM, budget online SVM, (LASVM), and online EM.