732A99/TDDE01 Machine Learning

Lecture 1c Block 2: Online Learning

Jose M. Peña
IDA, Linköping University, Sweden

Contents

- Online Learning
- Online EM Algorithm
- Online Kernel Methods
- Summary

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

$$\sum_n L(t_n, y(\boldsymbol{x}_n, \boldsymbol{\theta}))$$

  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 data point. It may even be an interesting option if the batch of data does not fit in main memory, e.g. scarce resources.

```
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, small devices.
- Goal: Minimize the cumulative loss suffered by updating the predictor.

Online Learning

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

```
Consistent
$\boldsymbol{\Theta}_1 = \boldsymbol{\Theta}$
for $n = 1, 2, \ldots$
    Receive question $\boldsymbol{x}_n$
    Choose any $\boldsymbol{\theta}_n \in \boldsymbol{\Theta}_n$
    Predict $y(\boldsymbol{x}_n, \boldsymbol{\theta}_n)$
    Receive true answer $t_n$
    Update $\boldsymbol{\Theta}_{n+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Theta}_n : y(\boldsymbol{x}_n, \boldsymbol{\theta}) = t_n\}$
```

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

```
Halving
$\boldsymbol{\Theta}_1 = \boldsymbol{\Theta}$
for $n = 1, 2, \ldots$
    Receive question $\boldsymbol{x}_n$
    Predict $\arg\max_{r \in \{0,1\}} |\{\boldsymbol{\theta} \in \boldsymbol{\Theta}_n : y(\boldsymbol{x}_n, \boldsymbol{\theta}) = r\}|$
    Receive true answer $t_n$
    Update $\boldsymbol{\Theta}_{n+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Theta}_n : y(\boldsymbol{x}_n, \boldsymbol{\theta}) = t_n\}$
```

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

Online Learning

- If we **drop** the realizability assumption, then our goal may be restated as minimizing the **regret**

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

- 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 the number of training points N.
- These goals seem hopeless anyway, as the true answers may come from an adversary who waits for our prediction and then answers the opposite label (*regret* $= N - N/2$ at least, e.g. if the batch solution is majority voting).
- Solution: Mild change of game. The adversary has to decide the true label before we make a prediction. Moreover, we are allowed to randomize our predictions, to cope with non-realizability.

Online Learning

```
Weighted majority (input: $\eta \in (0,1)$)
$\boldsymbol{w}_1 = (1/M, \dots, 1/M)$ where $M = |\boldsymbol{\Theta}|$
for $n = 1, 2, \dots$
    Choose $m \sim \boldsymbol{w}_n$ and predict $y(\boldsymbol{x}_n, \boldsymbol{\theta}_m)$
    Compute cost $c_{nm} = |y(\boldsymbol{x}_n, \boldsymbol{\theta}_m) - t_n|$ for all $m$
    Update $w_{n+1i} = \frac{w_{ni} exp(-\eta c_{ni})}{\sum_j w_{nj} exp(-\eta c_{nj})}$ for all $i$
```

- The algorithm predicts the label 1 (vs. 0) with probability

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

- The expected 0-1 loss on round n can be written as

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

- 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

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

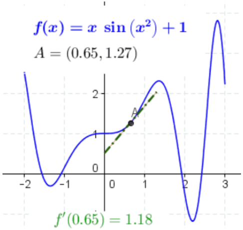
which reduces to $4 \log M$ for the realizable case (similar to Halving).

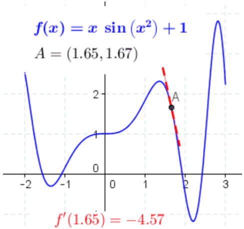
Online Learning

- If we **drop** the assumption that $|\boldsymbol{\Theta}| < \infty$, minimizing the regret may be achieved by online gradient descent, which updates the parameters as

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

  where $proj_{\boldsymbol{\Theta}}(\boldsymbol{v}) = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} ||\boldsymbol{v} - \boldsymbol{\theta}||$, and it is needed only if $\boldsymbol{\Theta}$ is a proper subset of $\mathbb{R}^D$.





- To ensure that stochastic gradient descent converges, we need to check that

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

  that is, not too small to avoid slow convergence and not too big to avoid jumping around the optimum, e.g. $\eta_n = 1/n$.

- Convergence vs adaptation or concept drift, e.g. $\eta_n = 1/2$.

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 $\pi_k$ to $\pi_k^{ML}$, and $\mu_{ki}$ to $\mu_{ki}^{ML}$ for all $k$ and $i$    /* M step */
```

Mixture of Bernoullis

$$\pi_k^{ML} = \frac{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{N}$$

$$\mu_{ki}^{ML} = \frac{\sum_n x_{ni} p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}$$

Mixture of Gaussians

$$\pi_k^{ML} = \frac{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}{N}$$

$$\boldsymbol{\mu}_k^{ML} = \frac{\sum_n \boldsymbol{x}_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}$$

$$\boldsymbol{\Sigma}_k^{ML} = \frac{\sum_n (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})^T p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}$$

- 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.

Kernel Classification

- Consider binary classification with input space $\mathbb{R}^D$.
- The best classifier under the 0-1 loss function is $y^*(\boldsymbol{x}) = \arg\max_y p(y|\boldsymbol{x})$.
- Since $\boldsymbol{x}$ may not appear in the finite training set $\{(\boldsymbol{x}_n, t_n)\}$ available, then classify according to weighted majority voting:

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

where $k : \mathbb{R}^D \to \mathbb{R}$ is a kernel function, which is usually non-negative and monotone decreasing along rays starting from the origin. The parameter h is called smoothing factor or width.

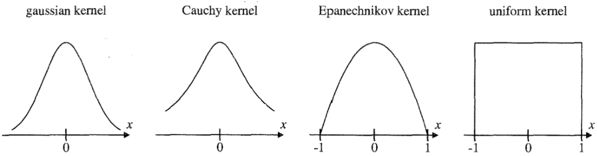


FIGURE 10.3. *Various kernels on $\mathcal{R}$.*

- Gaussian kernel: $k(u) = exp(-||u||^2)$ where $||\cdot||$ is the Euclidean norm.
- Cauchy kernel: $k(u) = 1/(1 + ||u||^{D+1})$
- Epanechnikov kernel: $k(u) = (1 - ||u||^2)\mathbf{1}_{\{||u||\leq 1\}}$

Kernel Classification

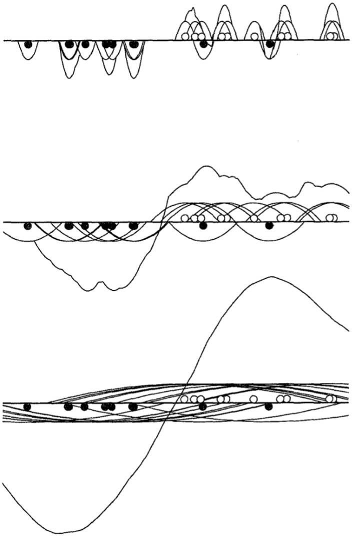


FIGURE 10.2. *Kernel rule on the real line. The figure shows* $\sum_{i=1}^{n}(2Y_i - 1)K((x - X_i)/h)$ *for* $n = 20$, $K(u) = (1 - u^2)I_{\{|u| \le 1\}}$ *(the Epanechnikov kernel), and three smoothing factors h. One definitely undersmooths and one oversmooths. We took* $p = 1/2$, *and the class-conditional densities are* $f_0(x) = 2(1 - x)$ *and* $f_1(x) = 2x$ *on* $[0, 1]$.

Online Kernel Classification

- Kernel classifier:

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

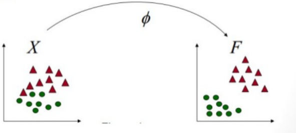
- The kernel classifier implies **no learning** and, thus, it can be applied to online learning directly. Deploying it may however become computationally demanding (specially with scarce resources), since it sums over all the training points.

Support Vector Machines for Classification

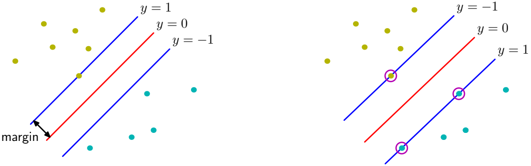
- Consider binary classification with input space $\mathbb{R}^D$.
- Consider a training set $\{(\boldsymbol{x}_n, t_n)\}$ where $t_n \in \{-1, +1\}$.
- Consider using the linear model

  $$y(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b$$

  so that a new point $\boldsymbol{x}$ is classified according to the sign of $y(\boldsymbol{x})$.
- Assume that the training set is linearly separable in the feature space (but not necessarily in the input space), i.e. $t_n y(\boldsymbol{x}_n) > 0$ for all n.



- Aim for the separating hyperplane that maximizes the margin (i.e. the smallest perpendicular distance from any point to the hyperplane) so as to minimize the generalization error.



Support Vector Machines for Classification

- The maximum margin separating hyperplane is given by

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

subject to $t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b) \geq 1$ for all n, which is equivalent to

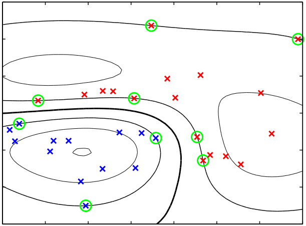
$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_n a_n\Big(t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b) - 1\Big)$$

where $a_n \geq 0$ are Lagrange multipliers, whose solution is

  - $\boldsymbol{w} = \sum_n a_n t_n \phi(\boldsymbol{x}_n)$, and
  - $a_n > 0$ if and only if $\boldsymbol{x}_n$ lies on the margin boundaries. Such points are called support vectors, or $\mathcal{S}$ for short.

- A new point $\boldsymbol{x}$ is classified according to the sign of

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



Online Support Vector Machines for Classification

- Support vector machines:

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

- Training a support vector machine needs 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, even with scarce resources.

Online Support Vector Machines for Classification

```
    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       $\mathcal{S} = \mathcal{S} \cup \{i\}$
7       $a_i = 1$
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 Support Vector Machines for Classification

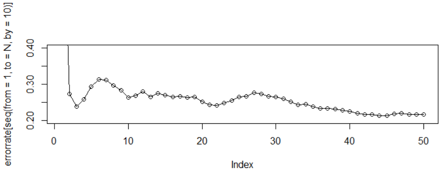
```
    Budget online SVM (input: $\beta$ and $M$)

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       $\mathcal{S} = \mathcal{S} \cup \{i\}$
7       $a_i = 1$
8       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.

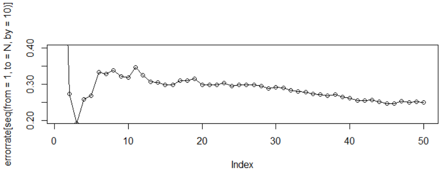
Online Support Vector Machines for Classification

- Online classification of 500 instances from the spambase data, using only the first 48 attributes.
- Gaussian kernel with $h = 1$.

$M = 500, \beta = 0$



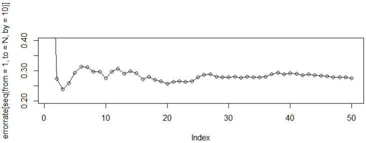
$M = 500, \beta = 0.05$



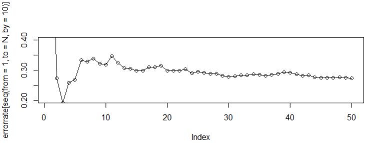
- $M = 500, \beta = 0$: Final error rate of 0.212 with 106 support vectors.
- $M = 500, \beta = 0.05$: Final error rate of 0.248 with 46 support vectors.

Online Support Vector Machines for Classification

$M = 20, \beta = 0$



$M = 20, \beta = 0.05$



- $M = 20, \beta = 0$: Final error rate of 0.274 with 20 support vectors.
- $M = 20, \beta = 0.05$: Final error rate of 0.278 with 20 support vectors.
- Plots 1 and 2 show that correcting for non-serious errors is beneficial if we do not have an upper limit to the number of support vectors.
- If we have such a limit, then plots 3 and 4 show that it may be better to use $\beta > 0$ so as to correct only for serious errors. Note that setting 4 is preferable since it results in a smoothly monotone decreasing error rate.
- Setting 3 is the slowest since it is the one that removes more vectors.

Summary

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