Machine Learning for Intelligent Systems

Lecture 23: Online Learning

Reading: UML 21 and Blum&Mansour chapter

Instructors: Nika Haghtalab (this time) and Thorsten Joachims
Statistical Learning Recap

PAC learning:

- Data set $S$ of $m$ samples is drawn i.i.d. from distribution $P$.
- Using this data set we want to find $h_S$.
- So, that $\text{err}_P(h_S) \leq \min_{h \in H} \text{err}_P(h) + \epsilon$.
- It works if $m \geq \frac{c_0}{\epsilon^2} \left( VCDim(H) + \ln \left( \frac{1}{\delta} \right) \right)$. 
Online Learning

The data might not be coming from a distribution:

- Today’s data can depend on yesterday’s data and decision.
- Environment is evolving over time in an unpredictable way.
- We don’t want to make any assumptions on how the data evolves.
- We want to make decisions on any instance as soon as it arrives.

**Online Learning framework (realizable)**

Sequence of data and learning tasks:

- On round $t$ we are given $x_t$ and unknown label $y_t = h^*(x_t)$ for a fixed $h^* \in H$.
- We predict $\hat{y}_t$, after the prediction we see if we made a mistake or not.
- Goal: Bound the number of mistakes we make.
Recall: Online Perceptron

Given a sequence of data $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_m, y_m)$ one by one, with radius $R$ and margin $\gamma := \min_{i \in S} \frac{y_i (\vec{w}^* \cdot \vec{x}_i)}{\|\vec{w}^*\|}$ for some $\vec{w}^*$.

**Online prediction:** At each time use the current $\vec{w}$ to predict the label of incoming $(\vec{x}_i, y_i)$, update if needed.

**Mistake Bound:** The number of mistakes that perceptron makes is at most $R^2 / \gamma^2$. 

**Theorem:** Mistake Bound of Online Perceptron
Mistake Bound Model

An algorithm $Alg$ learns a hypothesis class $H$ if $Alg$ make no more than $M$ mistakes on any sequence $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots$ that is consistent with some $h^* \in H$.

Goal: Upper bounding the mistake bound.
Example: 1-D thresholds (discrete)

Let \( X = \{1, \ldots, n\} \) be the instance space. Let \( H = \{h_a | a \in \{1, \ldots, n\}\} \) where \( h_a(x) = 1(x \geq a) \).

- \( x^- \): The right-most instance labeled \(-1\)
- \( x^+ \): The left-most instance labeled \(+1\)

Any Alg can be forced to make \( \geq \log_2(n) \) mistakes.

\[ \Rightarrow \text{Mistake bound is at least } \log_2(n). \]

There is a strategy that makes no more than \( \log_2(n) \) mistakes.

\[ \Rightarrow \text{Use the algorithm that at any time} \]

- Predict using \( h_a(.) \) for \( a \) halfway between \( x^- \) and \( x^+ \).

\[ \Rightarrow \text{On mistake: Distance between } x^- \text{ and } x^+ \text{ is halved (or smaller)} \]

- No more mistakes can be made when \( |x^- - x^+| = 1 \).
- \( n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \ldots \rightarrow 1. \)
Recall that the sequence is consistent with some $h^* \in H$. So, the version space will be non-empty.

**Idea:** Start with all consistent hypotheses. On mistake, make sure we can significantly narrow down the set of consistent hypotheses.

**Halving Algorithm**

Let $VS_1 = VS(H, \emptyset)$  

For $t = 1, \ldots, T$

- Receive $x_t$ and predict the same label $\hat{y}_t$ as the majority of $h \in VS_t$.
- $VS_{t+1} = VS_t \setminus \{h : h(x_t) \neq y_t\}$  

// This is equal to $H$  

// Remove the wrong hypotheses
## Halving: A generic Algorithm

<table>
<thead>
<tr>
<th>Include at $t = 1$?</th>
<th>$h_1$</th>
<th>$h_2$</th>
<th>$h_3$</th>
<th>$h_4$</th>
<th>$h_5$</th>
<th>$h_6$</th>
<th>$h_7$</th>
<th><strong>Alg</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction $(x_1,-)$?</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+, mistake</td>
</tr>
<tr>
<td>Include at $t = 2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Prediction $(x_2,+)$?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Include at $t = 3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Prediction $(x_3,-)$?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Include at $t = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>

### Theorem: Mistake Bound of Halving

For any $H$, Halving’s mistake bound is $\leq \log_2(|H|)$.

**Proof:** If we make a mistake at time $t$, majority of $VS_t$ were wrong $\rightarrow$

$|VS_{t+1}| \leq \frac{1}{2} |VS_t|$. After $\log_2(|H|)$ mistakes, only one hypothesis is left.
No Consistent Hypothesis

If no consistent $h^* \in H$, we can make infinitely many mistakes.

Compare with the best (not necessarily consistent) $h^* \in H$.

• Each $h \in H$ is an “expert” that gives you advice.
• Want to do nearly as well as the best “expert”, in hindsight.

Online algorithm that on sequence $(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)$ makes predictions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T$.

Algorithm’s # mistakes: $M = \sum_{t=1}^{T} 1(\hat{y}_t \neq y_t)$

Best Expert’s # mistakes: $OPT = \min_{h^* \in H} \sum_{t=1}^{T} 1(h^*(x_t) \neq y_t)$

Is $M$ close to $OPT$?
Attempt 1: Weighted Majority

Halving Algorithm:
• A mistake completely disqualifies an expert $h$.
• Predict with the majority of the remaining experts.

Weighted Majority Algorithm:
• A mistake lowers the weight of an expert $h$.
• Predict with the weighted majority of the experts.

<table>
<thead>
<tr>
<th>Weight $t = 1$?</th>
<th>$h_1$</th>
<th>$h_2$</th>
<th>$h_3$</th>
<th>$h_4$</th>
<th>$h_5$</th>
<th>$h_6$</th>
<th>$h_7$</th>
<th>Alg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction ($x_1, -$)?</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+, mistake</td>
</tr>
<tr>
<td>Include at $t = 2$</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Prediction ($x_2, +$)?</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-, mistake</td>
</tr>
<tr>
<td>Include at $t = 3$</td>
<td>1/4</td>
<td>1/4</td>
<td>1</td>
<td>1/4</td>
<td>1</td>
<td>1/4</td>
<td>1/2</td>
<td></td>
</tr>
</tbody>
</table>
Attempt 1: Weighted Majority

Halving Algorithm:
• A mistake completely disqualifies an expert \( h \).
• Predict with the majority of the remaining experts.

Weighted Majority Algorithm:
• A mistake lowers the weight of an expert \( h \).
• Predict with the weighted majority of the experts.

(Deterministic) Weighted Majority with parameter \( \beta \)

Initialize weights \( w_h^{(1)} = 1 \) for all \( h \in H \).
For \( t = 1, \ldots, T \)
    On \( x_t \) predict
    \[
    \hat{y}_t = \arg\max_y \sum_{h \in H} w_h^{(t)} \times 1(h(x_t) = y)
    \]
    For \( h \in H \)
    If \( h(x_t) \neq y_t \) then \( w_h^{(t+1)} = w_h^{(t)} \beta \), else \( w_h^{(t+1)} = w_h^{(t)} \).
Weighted Majority Guarantees

**Theorem: Guarantees of Weighted Majority** \( \beta = 0.5 \)

For \( M: \) Algorithms \# mistakes and \( \text{OPT}: \) best expert's \# mistakes, the (Deterministic) weighted majority algorithm with \( \beta = 0.5 \) gets

\[
M \leq 2.4 (\log_2(|H|) + \text{OPT}).
\]

Proof Idea:

- Best \( h^* \) makes \( \text{OPT} \) mistakes, so \( w_{h^*}^T = \left(\frac{1}{2}\right)^{\text{OPT}} \).
- The total weight at \( t = 1 \) of all experts is \( W = |H| \).
- On every mistake, half of the weight is on experts that made a mistake.
  - Their weight is cut by half. Total weight \( W \leftarrow \frac{1}{2} W + \frac{1}{2} W (0.5) = \frac{3}{4} W \).
  - After \( M \) mistakes, \( W \leq |H| \left(\frac{3}{4}\right)^M \).
- We have

\[
\left(\frac{1}{2}\right)^{\text{OPT}} \leq |H| \left(\frac{3}{4}\right)^M \rightarrow \left(\frac{4}{3}\right)^M \leq |H| 2^{\text{OPT}} \rightarrow M \leq 2.4 (\log_2 |H| + \text{OPT})
\]
Attempt 2: Randomized Decisions

• $M \leq 2.4(\log_2(|H|) + OPT)$ is good if $OPT$ is small.

• If $OPT$ is close to $T/2$ then this bound allows us to make a mistake on every turn.

• Want to show that $M - OPT$ is small
  → Ideally, smaller than $o(T)$.
  → On average over $T$ timesteps, we do nearly as well as the best expert.

Idea: Smoothly transition between predicting + or – based on the weights.

→ Weighted majority: 49% +, 51% -, predict –

→ Randomized Weighted majority 49% +, 51% -, predict + with 0.49 probability and – with 0.51 probability.

→ Allow less aggressive $\beta$. 
**Randomized Weighted Majority**

(Randomized) Weighted Majority with parameter $1 - \epsilon$

Initialize weights $w^1_h = 1$ for all $h \in H$.

For $t = 1, \ldots, T$

Let $W^t = \sum_{h \in H} w^t_h$ be the total weight at step $t$.

On $x_t$

Predict $\hat{y}$ with probability $\frac{1}{W^t} \sum_{h \in H} w^{(t)}_h \times 1(h(x_t) = \hat{y})$

For $h \in H$, if $h(x_t) \neq y_t$ then $w^{(t+1)}_h = w^{(t)}_h (1 - \epsilon)$, else $w^{(t+1)}_h = w^{(t)}_h$.

**Theorem: Guarantees of Rand. Weighted Majority**

For $M$: Algorithms # mistakes and OPT: best expert’s # mistakes, the randomized weighted majority algorithm with parameter $1 - \epsilon$ gets

$$\mathbb{E}[M] \leq (1 + \epsilon)OPT + \frac{1}{\epsilon} \log_2(|H|).$$

For $\epsilon = \sqrt{\frac{\log_2 |H|}{OPT}}$, get $\mathbb{E}[M] \leq OPT + 2\sqrt{T \log_2 |H|}$. 
Regret

**Definition: Regret**

Online algorithm that on sequence \((x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)\) makes predictions \(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\),

\[
\text{REGRET} = \sum_{t=1}^{T} 1(\hat{y}_t \neq y_t) - \min_{h^* \in H} \sum_{t=1}^{T} 1(h^*(x_t) \neq y_t)
\]

M: Algorithm’s # Mistakes  
OPT: Algorithm’s # Mistakes

**Theorem: Regret of Rand. Weighted Majority**

For randomized weighted majority when \(\epsilon = \sqrt{\frac{\log |H|}{OPT}}\), we have

\[
\mathbb{E}[\text{REGRET}] \leq 2\sqrt{T \log_2 |H|}.
\]