Online Learning, Mistake Bounds, Perceptron Algorithm

Francisco Marmolejo*

1 Online Learning

So far the focus of the course has been on *batch* learning, where algorithms are presented with a sample of training data, from which they must produce hypotheses that generalise well to unseen data. In what follows, we will cover *online* learning. In this setting, algorithms are sequentially given examples over time and must do two things at each time step: generate predictions, and upon recieving feedback on the performance of this prediction, update a current working hypothesis. This setting is useful when it is prohibitive to train on an entire set of data, or when algorithms need to respond dynamically to environments.

To be specific, suppose that C is an underlying concept class that we are trying to learn with H. The basic online setting for learning a concept $c \in C$ consists of $t = 1, 2, ...$ rounds (this can be finite or infinite) where the following happens:

- The learner has a current working hypothesis h_t.
- The learner is presented with an unlabeled example, x_t.
- The learner predicts a label $\widehat{y}_t = h_t(x_t)$.
- Upon this prediction, the learner is informed of the true label, $y_t = c(x_t)$.
- The learner updates the current working hypothesis to h_{t+1}.

One of the fundamental difference from the PAC framework we have studied before is the fact that there are no distributional assumptions made about the sequence of samples presented to the learner. Before we may have supposed that each x_i was generated independently from some distribution D over the input space X, but we may now even assume that the sequence is generated in an "adversarial" fashion to make our algorithm incur the largest loss possible. In fact, it is precisely this adversarial nature to the example sequence that yields important connections to Game Theory which we will hopefully explore later.

Since there are no distributional assumptions, an appropriate metric for algorithm performance we will look at is that of **mistakes** which is precisely as it sounds; a mistake is

*These notes are partially based off of Prof. Worrell's from previous years' Computational Learning Theory lectures.

an unlabeled example where the current working hypothesis of an online algorithm does not match the true label. (i.e. $h_t(x_t) \neq y_t$). The goal of our algorithms will thus be to minimize the overall number of mistakes made over an arbitrary sequence of samples.

Example: Disjunctions

Suppose that our concept class C is the set of all monotone disjunctions on $X = \{0,1\}^n$. Consider the following online algorithm for learning C:

- Begin with $h_1(x) = \vee_{i=1}^{n} x_i$
- For each t if $h_t(x_t) = \widehat{y}_t = c(x_t)$ leave h_t unchanged, otherwise remove all literals set to 1 in x_t from h_t.

It is straightforward to see that each time a mistake is made, the disjunctno in the current hypothesis decreases by at least one literal. For this reason the total number of mistakes is at most n.

On the other hand, one can never design a deterministic algorithm that can guarantee making less than n mistakes. To see this, consider the sequence of basis vectors in $\{0,1\}^n$, given by $e_1, ..., e_n$. No matter what current hypothesis an algorithm has, an adversary can always force a mistake on all these points.

2 Mistake-bound Learning

Now we proceed to formally define what we mean for a concept class to be learnable in the mistake-bound model of online learning.

Definition 1. *For a given hypothesis class C, and instance space $X = \bigcup X_n$, we say that an algorithm $\mathcal{A}$ learns C with mistake bound M if for some polynomial, $p(\cdot,\cdot)$, $\mathcal{A}$ makes at most $M = p(n, size(c))$ mistakes on any sequence of samples consistent with a concept $c \in C$. If the runtime of $\mathcal{A}$ is also polynomial in n and $size(c)$, then we say that $\mathcal{A}$ efficiently learns C in the mistake bound model.*

As we have seen above, the class of monotone disjunctions is learnable in the mistake-bound model with a mistake bound of n.

Remark: It is not difficult to see that there is a close connection between learning in the mistake-bound model and exact learning with equivalence queries. Suppose that $c \in C$ can be learned by an algorithm $\mathcal{A}$, with mistake bound M. As an exercise use this algorithm as a subroutine in an exact learning algorithm with equivalence queries that uses at most $M + 1$ queries. Conversly, show that if a concept class C can be exactly learned with N equivalence queries, then there is also mistake-bound algorithm for learning C that makes at most N mistakes.

Mistake-bound learning implies PAC learning

Definition 2. *We say an online learning algorithm is **conservative** if it only updates its current hypothesis when making a mistake.*

Lemma 1. *Suppose that $\mathcal{A}$ learns a hypothesis class C with mistake bound M. Then there exists a conservative algorithm $\mathcal{A}'$ that also learns C with the same mistake bound, M.*

Proof. The algorithm $\mathcal{A}'$ will be the same as $\mathcal{A}$ except for when the latter recieves an example it correctly labels. In this case, since $\mathcal{A}$ is not necessarily conservative, it may update its current hypothesis. $\mathcal{A}'$ simply "undoes" this update (or rather, acts as if this example had not arrived), and maintains the previous current hypothesis. Thus on a given sequence of elements, $x_1, ..., x_t, ...$ suppose that the updates of $\mathcal{A}'$ are given by the indices $i1, ..., ik$, then $\mathcal{A}'$ performs as if $\mathcal{A}$ had seen the sequence $x_{i1}, ..., x_{ik}$, which are indeed all mistakes, but there are at most M of these as assumed. □

With the previous lemma in hand, it suffices to restrict our attention to conservative algorithms.

Theorem 1. *Suppose that a concept class C is learnable in the mistake-bound model. Then C is also PAC-learnable.*

Proof. We assume that C is learned in the mistake-bound model by the conservative algorithm $\mathcal{A}$ that makes at most M mistakes. Furthermore, since we are in the PAC-learning paradigm, we have access to an example oracle $EX(c, D)$. We simply sample from the oracle in an online manner, and if ever the current hypothesis, say h, of $\mathcal{A}$ survives for more than $m = \frac{1}{\varepsilon} \log\left(\frac{M}{\delta}\right)$ samples, then return h.

The algorithm makes a mistake if it returns a hypothesis, h such that $err(h) > \varepsilon$ For a given sequence of m samples this happens with probability at most $(1-\varepsilon)^m < \frac{\delta}{M}$. Since the algorithm is conservative, the working hypothesis of $\mathcal{A}$ can change at most M times, therefore the total probability of error is bounded by $M(1-\varepsilon)^m < M\frac{\delta}{M} = \delta$. Furthermore, the total number of samples needed is at most $\frac{M}{\varepsilon} \log\left(\frac{M}{\delta}\right)$ □

Remark: Recall how this is similar to an exercise on the problem sheets where you were asked to simulate equivalence queries with queries from $EX(c, D)$.

Lower Bounds

Theorem 2. *Suppose that C is a concepet class with $VC(C) = d$. Then for any deterministic online learning algorithm there is a sequence of inputs for which the algorithm makes at least d mistakes.*

Proof. Since $VC(C) = d$, let $S = x_1, , , .x_d$ be a shatterable set under C. This means an adversary can label these points in any dichotomy that is still consistent with some $c \in C$. Therefore no matter what guess an algorithm gives on those points, an adversary can still force an error from a valid hypothesis. □

Even for randomised algorithms, it is simple to see that a lower bound on mistakes of $\frac{d}{2}$ in expectation still holds. All an adversary has to do is label $x_1, ..., x_d$ uniformly randomly.

Halving Algorithm

In what follows, we focus on learning finite hypothesis classes C with finite instance spaces $X = \bigcup X_n$.

If we are not concerned with computational tractability, we have the following simple halving algorithm to show give a mistake bound to learning C in an online setting:

- Let $C_1 = C$
- At time step t, we are given x_t and we compute the majority label vote of $c(x_t)$ for all $c \in C_t$ and choose this as a label for $\widehat{y}_t$.
- If $\widehat{y}_t$ is correct, then $C_{t+1} = C_t$, otherwise $C_{t+1} = C_t \setminus \{c \in C_t \mid c(x_t) = \widehat{y}_t\}$

Since we decrease the set of feasible hypothesis by at least a factor of $\frac{1}{2}$, then we get the following:

Theorem 3. *The number of mistakes made by the halving algorithm is at most* $\log_2 |C|$.

For example, if we let C be disjunctions over $\{0,1\}^n$, then $|C| = 3^n$ and we get that the halving algorithm gives us a mistake bound of $n \log_2(3)$. The problem however is in the computational cost of maintaining a list of feasible hypotheses and evaluating an input for all of them.

Also note that for a given finite hypothesis class, C, if we denote $OPT_M(C)$ as the best mistake bound achievable by an an online algorithm, then our results imply the following bounds:

$$VC(C) \leq OPT_M(C) \leq \log_2 |C|$$

It is straightforward to come up with examples where these inequalities are not tight.

Example: Projections

Suppose that we consider an example space, $\{0,1\}^n$ and we let $C = \{\pi_1, ..., \pi_n\}$ where $\pi_j(x) = x_j$. It is straightforward to see that the halving algorithm only makes one mistake at most (on the positive example).

Example: Binary Search

Suppose that $X_n = \{1, ..., 2^n\} \subset \mathbb{N}$ and that C is composed of all ordered half-interval functions c such that:

$$c_j(x) = \begin{cases} -1 & x \geq j \\ +1 & x < j \end{cases}$$

It is straightforward to see that $VC(C) = 1$, as no two-element set can be shattered. However, an adversary can always ensure that any online algorithm makes at least $\approx n$ mistakes. Suppose that the first element in the sequence is $x_1 = 2^{n-1}$. Roughly half of the concepts in $C_1 = C$ label this point as -1 and vice versa the other half labels as $+1$. This means that whatever label is returned, the remaining set of valid hypotheses, call it C_2 is half of C. For the second element in the sequence, if $y_1 = -1$, consider $x_2 = 2^{n-2}$, otherwise if $y_1 = +1$, consider $x_2 = 3 \cdot 2^{n-2}$. This will ensure once again that half of the remaining hypotheses in C_2 label x_2 with $+1$ and the other half with -1. Thus the remaining set of valid hypothes after the second example, C_3, is once again half the size of C_2. This can continue for approximately n turns, after which there is only one valid hypothesis left. At every stage of this processes, an adversary can incur an error on any online algorithm, and thus the bound holds.

Standard Optimal Algorithm

It is also natural to ask whether the halving algorithm is optimal, and the answer turns out to be no in general. The halving algorithm essentially maximally reduces the *number* of hypotheses in the worst case choice of label given by the adversary at time t, but this is not the right question to be asking if one is seeking to minimize the number of mistakes made overall, as the number of functions in a class does not have to correlate to the number of mistakes made on a sequence of inputs.

On the other hand, it is better to make a prediction such that if the prediction is wrong, the mistake bound of the remaining valid hypotheses is maximally reduced. You can see this as a game between the algorithm and the adversary, where the optimal strategy is "minimizing" the "maximal damage" an adversary can cause in the worst case. This is what is called the **Standard Optimal Algorithm**. As an exercise try to come up with an example where the optimal algorithm differs from the halving algorithm and results in lower mistake bounds.

3 Seperable Perceptron

We end this lecture by looking at the Perceptron algorithm which is an online algorithm for learning linear threshold functions. By linear threshold functions we mean functions $f : \mathbb{R} \to \{-1, +1\}$ of the following form:

$$f(x) = \begin{cases} +1 & w \cdot x \geq b \\ -1 & w \cdot x < b \end{cases}$$

Here w is a normal vector to the seperating hyperplane and b is a constant. Without loss of generality however, we can embed the elements $x \in \mathbb{R}^n$ into $\mathbb{R}^{n+1}$ via $x \to (x, 1)$, in which case $w' = (w, -b) \in \mathbb{R}^{n+1}$ defines an equivalent threshold function with a constant term of 0. Therefore it suffices to consider separating hyperplanes that pass through the origin. Also note that this hypothesis class is infinite, unlike when we were considering the halving algorithm.

Algorithm 1 Perceptron Algorithm

```
$w_1 \leftarrow 0$
for $t \leftarrow 1$ to $T$ do
  Recieve $(x_t)$
  if $w_t \cdot x_t \geq 0$ then
    $\widehat{y_t} \leftarrow +1$
  else
    $\widehat{y_t} \leftarrow -1$
  Recieve $y_t$
  if $y_t \neq \widehat{y_t}$ then
    $w_{t+1} \leftarrow w_t + y_t x_t$
  else
    $w_{t+1} \leftarrow w_t$
return $w_{T+1}$
```

The intuition behind the algorithm is that whenever it makes a mistake, if the mistake was on a positive example mislabeled as negative, then the weight vector shifts towards that example, on the other hand, if a mistake is made on a negative example mislabeled as positive, the weight vector is moved away from that point.

In order to give a mistake bound for the perceptron algorithm, we will resort to the notion of a margin. Consider a linear threshold function $f : \mathbb{R}^n \to \{-1, +1\}$ defined by a normal vector, u, of unit norm. The *margin* of a f at a labelled point (x, y) is defined to be $y(x \cdot u)$. This quantity is positive when correctly labeled and negative otherwise. Furthermore, its magnitude gives a distance of a point to the separating hyerplane.

Theorem 4. *Suppose that $(x_1, y_1), ..., (x_T, y_T)$ are such that $\|x_t\|_2 \leq D$ for some diameter $D > 0$. Suppose also that for some unit vector u (in the ℓ_2 norm) and $\gamma > 0$ the margin bound $y_t(x_t \cdot u) \geq \gamma$ holds for all $t = 1, ..., T$. Then the perceptron algorithm makes at most $(D/\gamma)^2$ mistakes*

Proof. Let m_t denote the number of mistakes before round t. We break the proof into two lemmas. The first shows that the projection of w_t on u gets longer with each mistake.

Lemma 2. *$w_t \cdot u \geq m_t \gamma$ for all t*

Proof. We induct on t. The case where $t = 0$ is trivial since $m_1 = 0$. Now suppose that the assumption holds for t. If there is no mistake in round t, then $m_{t+1} = m_t$ and $w_{t+1} = w_t$ and the claim clearly holds. Let us suppose that there is a mistake in round t.

$$\begin{aligned}
w_{t+1} \cdot u &= (w_t + y_t x_t) \cdot u \\
&= w_t \cdot u + y_t(x_t \cdot u) \\
&\geq m_t\gamma + \gamma \quad \text{(By the induction hypothesis and margin assumption)} \\
&= m_{t+1}\gamma
\end{aligned} \tag{1}$$

□

The next lemma bounds the growth of $\|w_t\|^2$ (in the ℓ_2 norm) in terms of the number of mistakes.

Lemma 3. $\|w_t\|^2 \leq m_t D^2$

Proof. Once again we induct on t, focusing on the non-trivial case when there is a mistake in round t.

$$\begin{aligned}
\|w_{t+1}\|^2 &= \|w_t + y_t x_t\|^2 \\
&= \|w_t\|^2 + 2y_t(w_t \cdot x_t) + \|x_t\|^2 \\
&\leq \|w_t\|^2 + \|x_t\|^2 \quad \text{(since there was a mistake)} \\
&\leq m_t D^2 + D^2 \quad \text{(inductive assumption)} \\
&= m_{t+1} D^2
\end{aligned} \tag{2}$$

□

Now we can complete the proof of the theorem as follows:

$$\begin{aligned}
D\sqrt{m_t} &\geq \|w_t\| \quad \text{(Lemma 3)} \\
&= \|w_t\|\|u\| \\
&\geq w_t \cdot u \quad \text{(Cauchy-Schwarz)} \\
&\geq m_t\gamma \quad \text{(Lemma 2)}
\end{aligned} \tag{3}$$

It follows that $m_t \leq (D/\gamma)^2$ □

Let us consider an application of the above mistake bound. Suppose that there are n financial analysts who predict every day whether the market will go up or down. We represent the prediction at time step t as a vector $x_t \in \{-1, 1\}^n$ and the actual outcome of the market as $y_t \in \{-1, 1\}$. We would like to use the Perceptron algorithm to figure out what analysts to follow. (The normal vector of a linear threshold function learned by the Perceptron algorithm can be seen as assigning a weight to each analyst).

Suppose that there is a subset of k "experts" (Suppose k is odd) within the group of n analysts, such that a majority vote of the k experts always gives the right prediction about the movement of the market. Define a unit vector:

$$u = \frac{1}{\sqrt{k}} v$$

Where v is a 0-1 vector that is the characteristic vector of the k experts.

Then u defines a linear separator with margin at least $\frac{1}{\sqrt{k}}$ since $y_t(x_t \cdot u) \geq \frac{1}{\sqrt{k}}$ for all t. Also $\|x_t\| \leq \sqrt{n}$ for all t. Therefore the previous theorem gives a mistake bound of nk in this case.

References

[1] N. Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning*, 2(1):285-318, 1988.