CSE599s, Spring 2012, Online Learning Lecture 1 -

Introduction and Mistake Bound Analysis

Lecturer: Ofer Dekel *Scribe: Andrew White*

1 Course Details

This course will cover the theory and mathematics behind online algorithms used in industry today.

1.1 Contact

- Website: `www.cs.washington.edu/cse599s`
- Brendan McMahan `<mcmahan@cs.washington.edu>`
- Ofer Dekel `<oferd@cs.washington.edu>`

1.2 Coursework

There will be six homeworks and each student will be responsible for transcribing one lecture. The lecture notes should be transcribed into LaTeX using the style files provided on the course website.

2 Introduction to Online Learning

Let us introduce online learning using the following motivating problem:

Example 1: Online Binary Prediction Game

Everyday we measure temperature, humidity, etc. and place the measurements into a *feature vector*(a.k.a. instance, data point, attribute vector) which we denote as $\vec{x}_t \in \mathbb{R}^n$. t is our time index and n is the dimension of the feature vector. Each day we also make a prediction, $\hat{y}_t \in \{+1, -1\}$, about whether or not it will snow. Assume $\hat{y}_t = 1$ indicates snow and $\hat{y}_t = -1$ indicates no snow.

A *binary classifier*(a.k.a. hypothesis) is a function which follows this procedure. It creates a binary prediction from a feature vector.

Definition 1: A binary classifier, h, is a function which maps a feature vector $x \in \mathbb{R}^n$ into a prediction set of two elements, e.g. $\{+1, -1\}$.

$$h : \mathbb{R}^n \to \{+1, -1\}$$

The following game, which is known as the *Online Binary Prediction Game*, predicts y_t using a feature vector $\vec{x}_t$ at each time t. The goal is to minimize the number of *mistakes.*

Online Binary Prediction Game

```
initialization step:   Choose $h_0$
                       For t = 1,2,...
                           Observe feature $\vec{x}_t \in \mathbb{R}^n$
                           Predict $\hat{y}_t = h_t(\vec{x}_t)$
                           Observe correct label $y_t \in \{+1, -1\}$
                           Suffer mistake if $\hat{y}_t \neq y_t$
update step:               Choose $h_{t+1}$
```

Notes:

- To define a concrete *online prediction algorithm*, one must specify the initialization step and the update step.
- There is immediate feedback after each prediction.
- There is *no stochastic assumption*(a.k.a. adversarial learning, individual sequence prediction). This means we do not assume a distribution on x_t. It may even come from an adversary.
- No distinction between a training set and test set

3 Mistake Bound Analysis

Mistake bound analysis is a type of algorithm analysis which places bounds on the maximum number of mistakes of an online prediction algorithm.

Example 1: The Halving Algorithm

The halving algorithm is an online prediction algorithm which has a bounded number of mistakes. It begins with the following assumptions:

- **Assumption 1:** There is a finite class of *experts*, $\mathcal{H}$, where each $h \in \mathcal{H}$ is a binary classifier.
- **Assumption 2:** $\exists\, h^* \in \mathcal{H}$ that predicts perfectly, $y_t = h^*(x_t) \forall t$. This is called the *realizability* assumption.

Notice that Assumption 2 is strong. Later in the course we will remove this assumption, moving onto what is called the *agnostic case*. We require one more definition.

Definition 2: The *version space*, $V_t \subseteq \mathcal{H}$, is the subset of experts which have the property $h(x_\tau) = y_\tau \; \forall \tau \leq t$, $\forall h \in V_t$. Namely, the subset of experts which are consistent with the observed data.

Our online prediction algorithm will use a majority vote over V_t.

$$\hat{y}_t = \text{mode}\,\{h_1(x_t), h_2(x_t), h_3(x_t), \ldots \; \forall h \in V_t\}$$

The update step will consist of updating the version space given the new observations, following the definition of the version space. The initialization step will be $V_t = \mathcal{H}$. This online prediction algorithm has a mistake bound given by the following theorem:

Theorem 1. *For any sequence $\{\vec{x}_t, y_t,\}_{t=1}^{\infty}$ that satisfies assumption 2, the halving algorithm makes at most $log_2\left(|\mathcal{H}|\right)$ mistakes.*

Proof:

The main argument of the proof is that each time a mistake is made, the version space is at least halved. In order to make a prediction mistake, at least half the experts must have voted incorrectly due to the majority vote. Thus, half the experts will no longer be consistent with the observed data and be removed from the version space. Formally:

$$h^* \in V_t \Rightarrow |V_t| \geq 1, \forall\, t$$

$$\hat{y}_t \neq y_t \Rightarrow |V_{t+1}| \leq \frac{1}{2}\,|V_t|$$

Let M_T denote the number of mistakes after T rounds.

$$1 \leq |V_{T+1}| \leq \left(\frac{1}{2}\right)^{M_T} |V_0| = \left(\frac{1}{2}\right)^{M_T} |\mathcal{H}|$$

$$M_T \leq \log_2(|\mathcal{H}|)$$

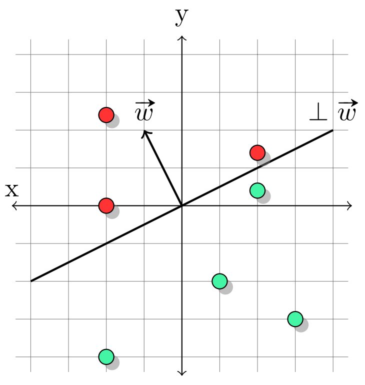
Example 2: The Perceptron

In this example, we use a linear binary classifier to create an online binary prediction algorithm called a perceptron [1]. The word perceptron refers to this algorithm's original purpose in mimicking a neuron.

Definition 3: A linear binary classifier, h_w, is a binary classifier defined by a vector of weights $\vec{w} \in \mathbb{R}^n$. It makes predictions according to:

$$\hat{y}_t = h_w(\vec{x}) = \text{sign}\,(\vec{w} \cdot \vec{x}) = \text{sign}\left(\sum_i^n x_i w_i\right)$$

On $\mathbb{R}^2$ the binary linear classifier has an intuitive geometric interpretation. If we plot $\vec{w}$, we can see it creates a plane or line which divides the points into two sets. All the points which are red are classified as +1, namely $\vec{w} \cdot \vec{x} > 0$. All the points which are green have $\vec{w} \cdot \vec{x} < 0$ and are classified as -1. The two subspaces defined by the plane are called the positive and negative half-spaces. The plane which separates the two is called the *perpendicular hyperplane*.



The definition of a linear classifier allows us to create the following online binary prediction algorithm called the perceptron. The initialization step is $\vec{w_0} = (0, \ldots, 0)$. Note that sign $0 \equiv 1$, so that $\vec{x} \cdot \vec{w}_0 = 1$. The update step is define as:

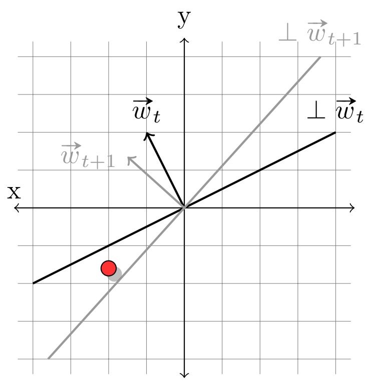
$$\vec{w}_{t+1} = \begin{cases} \vec{w}_t & \text{if} \quad \hat{y}_t = y_t \\ \vec{w}_t + y_t x_t & \text{if} \quad \hat{y}_t \neq y_t \end{cases}$$

The effectiveness of the update step can be seen, for example, after an $\vec{x}_t$ is misclassified:

$$\vec{w}_{t+1} \cdot \vec{x}_t = (\vec{w}_t + \vec{x}_t) \cdot \vec{x}_t$$

$$= \vec{w}_t \cdot \vec{x}_t + ||\vec{x}_t||^2 \Rightarrow \vec{w}_{t+1} \cdot \vec{x}_t \geq \vec{x}_t \cdot \vec{w}_t$$

This can be shown geometrically as well, where we can see that the update step makes $\vec{w}$ more co-linear with $\vec{x}$.

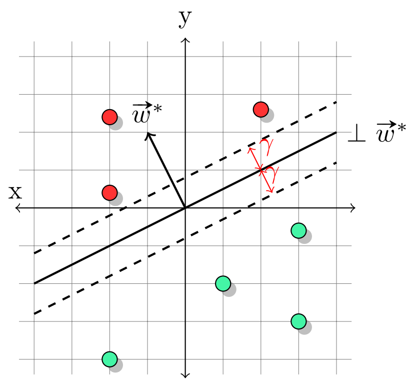


With two assumptions, we can bound the mistakes. These are:

- **Assumption 3:** The feature vectors are bounded: $||\vec{x}|| \leq \rho$

- **Assumption 4:** $\exists\, \vec{w}^* \in \mathbb{R}^n$, with $||\vec{w}^*|| \leq 1$, such that $y_t \vec{w}^* \cdot \vec{x}_t \geq \gamma > 0$.

Notice that Assumption 4 is similar to Assumption 2 in that there exists a 'best' classifier. Assumption 4 is actually a stronger assumption than Assumption 2, as seen from the fact that $\gamma = 0$ is equivalent to Assumption 2. Returning to our geometric interpretation, we may view the effect of the margin γ.



With the two assumptions above, the following theorem, from Novikoff [2] holds:

Theorem 2. *For any sequence* $\{\vec{x}_t, y_t,\}_{t=1}^{\infty}$ *that satisfies Assumptions 3 and 4, the number of mistakes for the perceptron classifier is at most*

$$\left(\frac{\rho}{\gamma}\right)^2$$

Proof:

The first step of the proof is showing that $\vec{w}$ grows more colinear with $\vec{w}^*$ after each mistake. The colinearity is calculated using a *potential function*, in this case $\vec{w}_t \cdot \vec{w}^*$.

$$y_t \neq \hat{y}_t \Rightarrow \vec{w}_{t+1} = \vec{w}_t + y_t \vec{x}_t$$

$$\vec{w}_{t+1} \cdot \vec{w}^* = (\vec{w}_t + y_t \vec{x}_t) \cdot \vec{w}^* = \vec{w}_t \cdot \vec{w}^* + y_t \vec{x}_t \vec{w}^*$$

$$\text{using Assumption 4} \;\; \vec{w}_{t+1} \cdot \vec{w}^* \geq \vec{w}_t \vec{w}^* + \gamma$$

The argument above can be repeated for each $t < T$. Notice, it only holds during a mistake. When there is no mistake, $\vec{w}_{t+1} \cdot \vec{w}^* = \vec{w}_t \cdot \vec{w}^*$. Repeating the argument after M_T mistakes up to time T:

$$\vec{w}_{T+1} \cdot \vec{w}^* \geq \vec{w}_0 \cdot \vec{x}_0 + M_T\gamma \geq M_T\gamma$$

This shows the potential function increases; however, the potential function could grow without the vectors becoming more colinear if $\vec{w}_t$ simply increases in magnitude each step. We can bound this growth using Assumption 3:

$$y_t \neq \hat{y}_t \Rightarrow ||\vec{w}_{t+1}||^2 = ||\vec{w}_t + y_t \vec{x}_t||^2$$

After M_T mistakes

$$||\vec{w}_{T+1}||^2 = \left|\left|\vec{w}_0 + \sum_{\tau}^{M_T} y_\tau \vec{x}_\tau\right|\right|^2 \leq ||0 + M_T\rho||^2$$

where the initialization step and Assumption 3 were used to make the final inequality. Both results may be combined:

$$M_T\gamma \leq \vec{w}_{T+1} \cdot \vec{w}^* \leq ||\vec{w}_{T+1}||\,||\vec{w}^*|| = ||\vec{w}_{T+1}|| \cdot 1 \leq \sqrt{M_T}\rho$$

$$M_T\gamma \leq \sqrt{M_T}\rho \Rightarrow M_T \leq \left(\frac{\rho}{\gamma}\right)^2$$

Notice that the r.h.s doesn't depend on T, thus the inequality is valid for all T. The number of mistakes is bound.

The realizability assumption is unrealistic. Practically, a perfect classifier doesn't always exists. As the course continues, we'll move to a new analysis setting, the *online optimization setting.* A specialization of that setting will also be used, the *online convex optimization setting.*

References

[1] F. Rosenblatt, "The Perceptron–a perceiving and recognizing automaton", *Cornell Aeronautical Laboratory*, Report 85-460-1, 1957.

[2] A. B. Novikoff, "On convergence proofs on perceptrons", *Symposium on the Mathematical Theory of Automata*, 12, 615-622, 1962.