CSE599s, Spring 2014, Online Learning Lecture 1 - 04/01/2014

Introduction to Online Learning

Lecturer: Ofer Dekel *Scribe: De Meng*

1 Course Info

- Instructors: Ofer Dekel, `oferd@microsoft.com`, Brendan McMahan, `mcmahan@google.com`
- Course website: `http://courses.cs.washington.edu/courses/cse599s/14sp/index.html`

2 Introduction

Online prediction can be considered as a repetitive game between one player (a.k.a. predictor, learning algorithm) and the environment (a.k.a. adversary). Denote T the number of rounds of the game. On each round t ($t = 1, \ldots, T$), the game is played by the player and environment in the following intuitive form

- **Environment**: Choose an *instance* of problem.
- **Player**: Make a *prediction* for this instance.
- **Environment**: Incur a *loss* $\in \mathbb{R}$ for this prediction. (Smaller loss is preferred by the player.). Send the *feedback* on the accuracy of the prediction to the player.
- **Player**: *Learn* and record the feedback.

2.1 Example: Online Binary Prediction Game

Online binary prediction has many applications, such as email spam classification, which may not fit into stochastic models. In this application, the player observes some features of an email and makes a binary prediction, either spam or not spam. Here are details: for each round $t = 1, \ldots, T$

- Player observes a feature vector $x_t \in \mathbb{R}^n$ of an instance generated by the environment.
- Player makes a *binary* prediction $\hat{y}_t \in \{+1, -1\}$. $+1, -1$ represent "spam" and "not spam" respectively.
- Player observes feedback $y_t \in \{+1, -1\}$.
- A loss is incurred $\ell_t = \mathbf{1}_{\hat{y}_t \neq y_t}$.

After T rounds, the cumulative loss is $\sum_{t=1}^{T} \ell_t$. Intuitively, a "small" cumulative loss is desired at the end. This "small" will later be defined precisely in the term of regret.

Here we present an equivalent definition of the binary prediction game. For $t = 1, \ldots, T$,

- Player chooses a *hypothesis* (a.k.a. binary classifier) h_t from hypothesis class $\mathcal{H}$

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

  The hypothesis is a function that maps from a feature vector $x \in \mathbb{R}^n$ to a binary prediction set, e.g. $\{+1, -1\}$. At round t, the choice of h_t could base on all information recorded before t.
- Environment chooses (x_t, y_t).
- Player incurs a loss $\ell_t = \mathbf{1}_{h(x_t) \neq y_t}$.

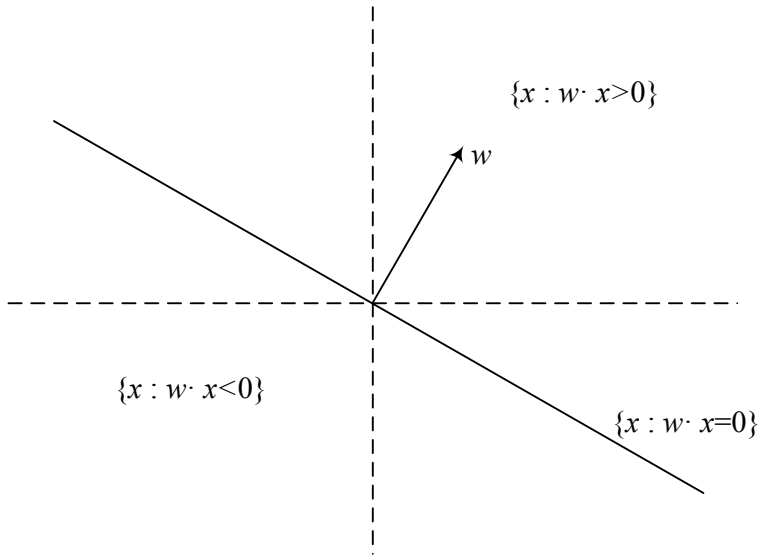

Figure 1: Hyperplane and halfspaces

2.2 Example: Online Binary Linear Predictor with Hinge Loss

Let's first define binary linear predictor and hinge loss function.

The hypothesis $h_w : \mathbb{R}^n \to \{+1, -1\}$

$$h_w(x) = \text{sign}(w \cdot x) = \begin{cases} +1, & \text{if } w \cdot x > 0 \\ -1, & \text{if } w \cdot x < 0 \end{cases}$$

is called *binary linear predictor*. The only parameter for this hypothesis is the vector $w \in \mathbb{R}^n$. Geometrically, all vectors that are perpendicular to w (i.e. zero inner product) forms a hyperplane $\{x : w \cdot x = 0\}$, shown in Figure 1. The data may fall into one of halfspaces $\{x : w \cdot x < 0\}$ and $\{x : w \cdot x > 0\}$. $|w \cdot x|$ can be interpreted as the prediction *confidence*.

The hypothesis class $\mathcal{H}$

$$\mathcal{H} = \{h_w(x) : w \in \mathbb{R}^n, \|w\|_2 \leq 1\},$$

is the class of binary linear predictors.

A number of loss functions have been proposed to incur penalty for prediction error. $\mathbf{1}_{\hat{y}_t \neq y_t}$ is simply the *error indicator*, or called *zero-one loss*. The *hinge loss* is defined as

$$l(w; (x_t, y_t)) = \max\{0, 1 - y_t w \cdot x_t\}.$$

As shown in Figure 2, hinge loss function imposes penalty for wrong prediction ($y_t w \cdot x_t < 0$) and right prediction with small confidence ($0 \leq y_t w \cdot x_t < 1$).

The binary linear prediction game with hinge loss can be presented as follows. For $t = 1, \ldots, T$,

- Player chooses $w_t \in \mathcal{W}$, where $\mathcal{W} = \{w \in \mathbb{R}^n : \|w\|_2 \leq 1\}$, a unit ball in $\mathbb{R}^n$.
- Environment chooses (x_t, y_t).
- Player incurs a loss $\ell_t(w_t; (x_t, y_t)) = \max\{0, 1 - y_t w \cdot x_t\}$. ($x_t, y_t$ are regarded as parameters of ℓ_t.)
- Player receives feedback (x_t, y_t).

This game setting can be generalized as *online convex optimization (OCO)* which is discussed in next section.

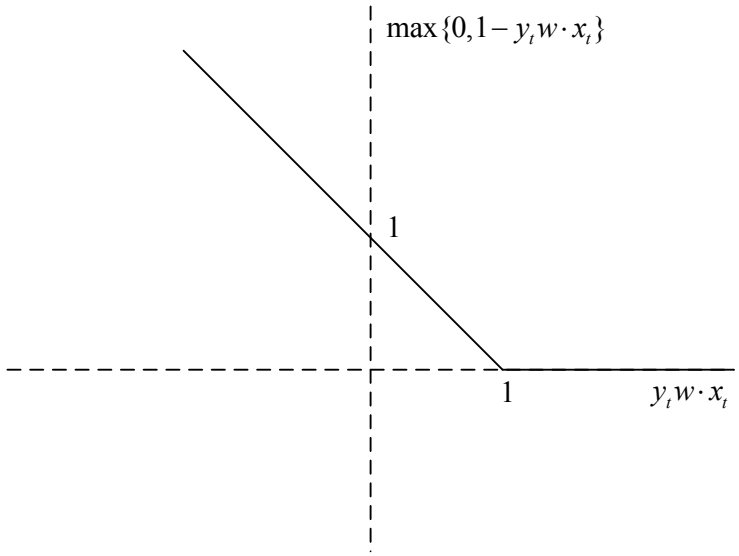

Figure 2: Hinge loss function

3 Online Convex Optimization (OCO)

For $t = 1, \ldots, T$

- Player chooses $w_t \in \mathcal{W}$, where $\mathcal{W}$ is a *convex* set in $\mathbb{R}^n$.
- Environment chooses a *convex* loss function $f_t : \mathcal{W} \to \mathbb{R}$.
- Player incurs a loss $\ell_t = f_t(w_t) = f_t(w_t; (x_t, y_t))$.
- Player receives feedback f_t.

Assumption: The environment is oblivious to $w_1, \ldots, w_T$, but can define $f_1, \ldots, f_T$ arbitrarily (perhaps maliciously, even with full knowledge of the player's algorithm, with infinity computation power).

Then, at the end, the cumulative loss $\sum_{t=1}^{T} l_t$ can be arbitrarily large. How to evaluate the prediction performance? What is a good benchmark for the performance?

A good choice is the cumulative loss of the *best fixed (or say static)* hypothesis in *hindsight*,

$$\min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

To choose this best fixed hypothesis, we need to know future, that is to collect all $f_1, \ldots, f_T$, then run an *off-line* algorithm.

The difference between the real cumulative loss and this minimum cumulative loss for fixed hypothesis in hindsight is defined as *regret*,

$$R(T) = \sum_{t=1}^{T} f_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

- If regret grows linearly, $R(T) = \Omega(T)$, the player is not learning.

- If regret grows sub-linearly, $R(T) = o(T)$, the player is learning and its prediction accuracy is improving. The regret per round goes to zeros as T goes to infinity,

$$\frac{1}{T}\left(\min_{w\in\mathcal{W}}\sum_{t=1}^{T} f_t(w)\right) \to 0, \quad T \to \infty.$$

- $R(T) = O(\sqrt{T})$.

4 Comparison Between Online Learning and Statistical Learning

	Online learning (OL)	**Statistical learning (SL)**
Similarities	Both define hypothesis space/class of predictors (in each round of a game in OL while in training procedure of SL).	
	Both define a loss function to evaluate the prediction performance, and small loss is preferred.	
	Instances and labels	
Differences	learning in each round of game, no distinction between training and testing	first train a model, then test it
	adversary case	statistical assumption