IIIT Hyderabad | **Scribed By: Jerin Philip, Tejaswi Kasarla**
Topics in Machine Learning (CSE975) | **Lecture #2**
Instructor: Dr. Naresh Manwani | **August 5, 2017**

Outline. *Introduction to a general purpose additive online learning framework using stochastic gradient descent. Deriving the Perceptron learning algorithm followed by its convergence proof for linearly separable case. Mistake bound analysis for linearly non-separable case.*

1 Introduction

Online learning is a model of machine learning in which the algorithm has to make predictions continuously even when it's learning. The learning takes place in a sequence of input instances. In this lecture, we discuss the supervised setting of online learning. We also analyze the Perceptron algorithm, which is a widely used online learning algorithm.

2 Definitions

The instance space, where we draw input samples from is denoted by $\mathcal{X}$, where $\mathcal{X} \subset \mathbb{R}^d$. We can categorise the task in hand based on the label space, denoted by $\mathcal{Y}$ as:

$$\mathcal{Y} = \begin{cases} \{+1, -1\} & \text{Binary Classification} \\ \{1, 2, 3, .., C\} & \text{Multi-class Classification} \\ \mathbb{R} & \text{Regression} \end{cases}$$

We assume our data is drawn from the distribution represented by the joint density function $P(\mathbf{x}, y)$, where:

$$P(\mathbf{x}, y) = \mu(\mathbf{x})\ P(y \mid \mathbf{x})$$

We're looking for a hypothesis, $f : \mathcal{X} \to \mathcal{Y}$, which is obtained by training on a set of samples drawn independently from the underlying data distribution, $P(\mathbf{x}, y)$. The training set is denoted by $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), .., (\mathbf{x}_N, y_N)\}$.

A hyperplane in d-dimensions is represented by the following equation.

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0 \qquad\qquad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

2.1 Loss Function

The hypothesis $f(x)$ predicts the label. To measure the discrepancy between the prediction and the corresponding actual label y,we denote a generic loss function using one of the following:

$$L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+ \qquad\qquad \text{If considering } sign(f(\mathbf{x}))$$
$$L : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+ \qquad\qquad \text{Considering simply } f(\mathbf{x})$$

2.2 Risk

Risk is defined as the expectation of the loss function. This expectation is with respect to the joint distribution function $P(\mathbf{x}, y)$

$$R_L(f) = E[L(f(\mathbf{x}), y)] \qquad \text{Expected Risk}$$

In practical training situations, we don't have access to the joint distribution function, $P(\mathbf{x}, y)$. We are usually provided with a finite set of training examples, $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), .., (\mathbf{x}_N, y_N)\}$. In such situation, we find the *empirical risk* as follows:

$$\hat{R_L}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \qquad \text{Empirical Risk}$$

Minimizing the risk of a learning algorithm can be achieved in two ways. One is through *batch learning* where the risk over all the training examples is minimized at once. SVM, logistic regression algorithms minimize the risk through batch learning. The second way, is through online learning where, the risk is minimized in a sequential order and is fast and scalable.

2.3 Generic Online Learning Framework

In an online learning algorithm, the learning takes place in a sequence of trials. For a given mapping of set of instances $\mathbf{x} \in \mathcal{X}$ to the set of admissible labels $y \in \mathcal{Y}$, the learner uses a prediction mechanism, called hypothesis to predict the labels of the instances. In each trial, the learning algorithm:

- Observes the example instance $\mathbf{x}_t$.
- Uses current hypothesis f_t to predict the output $\hat{y_t} = f_t(\mathbf{x}_t)$.
- Observe actual output y_t.
- Incurs a loss of $L(\hat{y_t}, y_t)$.
- Updates the current hypothesis.
- There is an objective associated with the training, which is to minimize the total loss, in a sequence of T trials.

$$\text{Total Loss} = \sum_{t=1}^{T} L(\hat{y_t}, y_t)$$

2.3.1 Stochastic Gradient Descent

In the above framework, there are some specifics left out. What choice of loss function do we make? How do we update the hypothesis? One widely used method to update the hypothesis is Stochastic Gradient Descent (SGD).

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} L(f(\mathbf{x}_t), y_t) \tag{1}$$

3 Perceptron

One model of the above framework is the linear Perceptron. Input to Perceptron is $\mathbf{x}$, hypothesis is given by:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2}$$

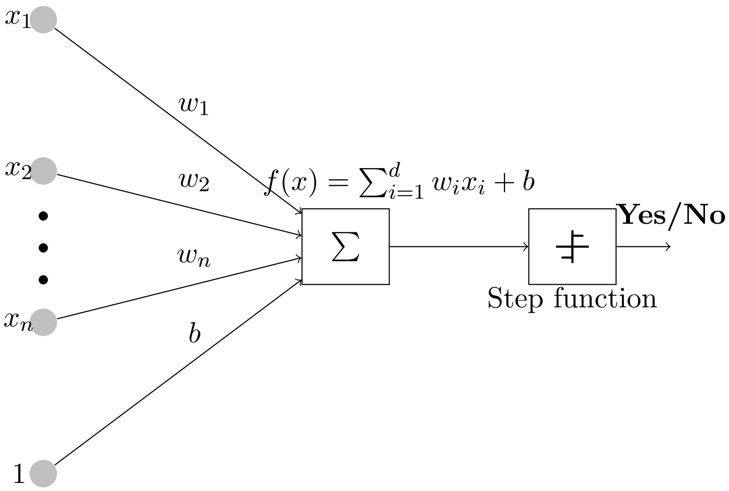

Figure 1: The Perceptron model

f predicts a real value, and $\text{sign}(f(\mathbf{x}))$ gives the label.

3.1 Loss

In the general case, if a sample is misclassified, then $y \neq f(\mathbf{x})$. Since $y \in \{-1, +1\}$, considering binary classification, we can concisely write the conditions as:

$$\begin{cases} yf(\mathbf{x}) < 0 & \text{misclassified} \\ yf(\mathbf{x}) \geq 0 & \text{correctly classified} \end{cases}$$

We use the following loss for Perceptron:

$$L_P(f(\mathbf{x}), y) = \max(0, -yf(\mathbf{x})) \tag{3}$$

Now taking the Perceptron loss function, we can observe the following:

$$\begin{cases} yf(\mathbf{x}) < 0 & \text{misclassified} \implies L_P(f(\mathbf{x}), y) = -yf(\mathbf{x}) \\ yf(\mathbf{x}) \geq 0 & \text{correctly classified} \implies L_P(f(\mathbf{x}), y) = 0 \end{cases}$$

i.e, for a misclassified sample Perceptron loss penalizes by $yf(\mathbf{x})$, while if the sample is correctly classified there is no penalty.

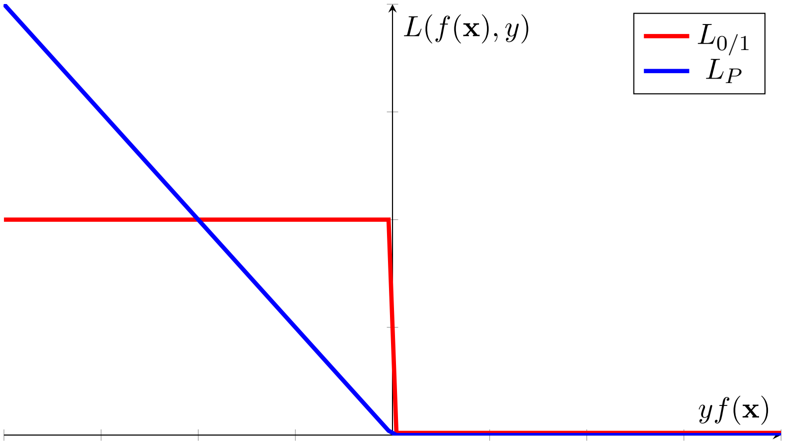

Figure 2: Loss of Perceptron model compared to 0-1 loss

3.2 Update Rules

$$\nabla_w L_P(\mathbf{w}^T\mathbf{x}_t, y_t) = \begin{cases} \nabla_w \big(-y_t(\mathbf{w}^T\mathbf{x}_t + b)\big) & \text{misclassified} \\ 0 & \text{correctly classified} \end{cases}$$

$$\nabla_b L_P(\mathbf{w}^T\mathbf{x}_t, y_t) = \begin{cases} \nabla_b \big(-y_t(\mathbf{w}^T\mathbf{x}_t + b)\big) & \text{misclassified} \\ 0 & \text{correctly classified} \end{cases}$$

The above two reduces to:

$$\nabla_w L_P(\mathbf{w}^T\mathbf{x}_t, y_t) = \begin{cases} -y_t\mathbf{x}_t & \text{misclassified} \\ 0 & \text{correctly classified} \end{cases}$$

$$\nabla_b L_P(\mathbf{w}^T\mathbf{x}_t, y_t) = \begin{cases} -y_t & \text{misclassified} \\ 0 & \text{correctly classified} \end{cases}$$

Substituting above values in stochastic gradient descent equation, we obtain the Perceptron update rule:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + y_t\mathbf{x}_t & \text{misclassified} \\ \mathbf{w}_t & \text{correctly classified} \end{cases}$$

$$b_{t+1} = \begin{cases} b_t + y_t & \text{misclassified} \\ b_t & \text{correctly classified} \end{cases}$$

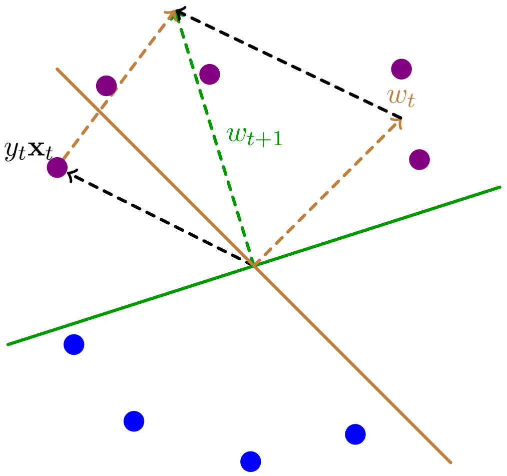

Figure 3: The weight update in Perceptron during misclassification

3.3 The Perceptron Algorithm

Algorithm 1: Linear Perceptron

```
Input  : $\mathbf{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$
Result: $(\mathbf{w}*, b*)$
Initialize $\mathbf{w}_1 = 0, b_1 = 0$;
for $t = 1$ to $T$ do
    $(\mathbf{x}_t, y_t)$ = draw random sample from $\mathbf{S}$;
    if $y_t(\mathbf{w}_t^T\mathbf{x}_t + b) < 0$ then
        $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t x_t$;
        $b_{t+1} = b_t + y_t$;
    else
        $\mathbf{w}_{t+1} = \mathbf{w}_t$;
        $b_{t+1} = b_t$;
    end
end
$(\mathbf{w}*, b*) = (\mathbf{w}_{T+1}, b_{T+1})$
```

3.4 Convergence Analysis

Theorem 3.1 (Finite convergence in linear separable case)**.** *Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), .., (\mathbf{x}_T, y_T)\}$, where T is number of samples drawn. Let $\exists$ $\mathbf{u} \in \mathbb{R}^d$, s.t $\|\mathbf{u}\|_2 = 1$ and $y_t\mathbf{u}^T\mathbf{x}_t \geq \gamma$, where $\gamma \geq 0, \forall t = 1..T$. Let $R_2 = \max_t \|\mathbf{x}_t\|_2$. The Perceptron algorithm converges in at most $\left(\frac{R_2}{\gamma}\right)^2$ iterations.*

Proof. Let $\mathbf{x}_t$ be misclassified.

$$y_t(\mathbf{w}_t^T\mathbf{x}_t + b_t) < 0$$

$$\begin{aligned}\mathbf{w}_{t+1}^T\mathbf{u} &= (\mathbf{w}_t + y_t\mathbf{x}_t)^T\mathbf{u}\\ &= \mathbf{w}_t^T\mathbf{u} + y_t\mathbf{x}_t^T\mathbf{u}\\ &\geq \mathbf{w}_t^T\mathbf{u} + \gamma\end{aligned}$$

$$(\mathbf{w}_{t+1} - \mathbf{w}_t)^T\mathbf{u} \geq \gamma$$

In T trials, let there be k mistakes happening.

$$\begin{aligned}(\mathbf{w}_2 - \mathbf{w}_1)^T\mathbf{u} &\geq \gamma \ or \ 0\\ (\mathbf{w}_3 - \mathbf{w}_2)^T\mathbf{u} &\geq \gamma \ or \ 0\\ &...\\ (\mathbf{w}_{T+1} - \mathbf{w}_T)^T\mathbf{u} &\geq \gamma \ or \ 0\\ \hline\end{aligned}$$

$$\begin{aligned}(\mathbf{w}_{T+1} - \mathbf{w}_1)^T\mathbf{u} &\geq k\gamma\\ \mathbf{w}_{T+1}^T\mathbf{u} &\geq k\gamma\end{aligned}$$

Squaring the above, since $\gamma \geq 0$ is given, $k \geq 0$ by construction, we obtain:

$$\|\mathbf{w}_{T+1}\|_2^2 \geq k^2\gamma^2 \tag{4}$$

To get an upper bound on $\mathbf{w}_{T+1}^T\mathbf{u}$, we use Cauchy Schwartz Inequality.

$$\mathbf{w}_{T+1}^T\mathbf{u} \leq \|\mathbf{w}_{T+1}\|_2\|\mathbf{u}\|_2 = \|\mathbf{w}_{T+1}\|_2$$

Consider

$$\begin{aligned}\|\mathbf{w}_{t+1}\|_2^2 &= \|\mathbf{w}_t + y_t\mathbf{x}_t\|_2^2\\ &= \|\mathbf{w}_t\|_2^2 + \|\mathbf{x}_t\|_2^2 + 2y_t\mathbf{w}_t^T\mathbf{x}_t\\ &\leq \|\mathbf{w}_t\|_2^2 + R_2^2 && (y_t\mathbf{w}_t^T\mathbf{x}_t \leq 0).\\ \|\mathbf{w}_{t+1}\|_2^2 - \|\mathbf{w}_t\|_2^2 &\leq R_2^2\end{aligned}$$

Extending the above result for k mistakes:

$$\begin{aligned}\sum_{t=1}^{T} \|\mathbf{w}_{t+1}\|_2^2 - \|\mathbf{w}_t\|_2^2 &\leq kR_2^2\\ \|\mathbf{w}_{T+1}\|_2^2 &\leq kR_2^2\end{aligned}$$

Combining upper bound and lower bound, we have:

$$k^2\gamma^2 \leq \|\mathbf{w}_{T+1}\|_2^2 \leq kR_2^2$$

$$k \leq \left(\frac{R_2}{\gamma}\right)^2 \tag{5}$$

□

Special Cases:

- What if $\|\mathbf{u}\|_2 \neq 1$?

$$\begin{aligned} \mathbf{w}_{T+1}^T\mathbf{u} &\geq k\gamma \\ \mathbf{w}_{T+1}^T\mathbf{u} &\leq \|\mathbf{w}_T\|_2\|\mathbf{u}\|_2 \\ \|\mathbf{w}_T + 1\|_2^2 &\leq kR_2^2 \end{aligned}$$

  Combining all three,

$$k\gamma \leq \mathbf{w}_{T+1}^T\mathbf{u} \leq \sqrt{k}R_2\|\mathbf{u}\|_2$$

  The final inequality changes to

$$k \leq \left(\frac{R_2}{\gamma}\right)^2 \|\mathbf{u}\|_2^2$$

- When will $k = \left(\frac{R_2}{\gamma}\right)^2$ happen?

  Let $\mathbf{x}_t,\ t = 1 \dots T$ be the standard basis $\mathbb{R}^T$. Then, over every trial, the input $\mathbf{x}_t$ will be misclassified. Since the Perceptron algorithm will mis-classify for every trial, the total number of mistakes, $k = T$. $\|\mathbf{x}_t\|_2 = 1,\ \forall t = 1 \dots T$. Consider $\mathbf{u} \in \mathbb{R}^T$ s.t. $\mathbf{u} = \frac{1}{\sqrt{T}}[y_1 \ \ y_2 \ \ \dots \ \ y_T]$. We see that $\|u\|_2 = 1$ and $y_t\mathbf{u}^T\mathbf{x}_t = \frac{1}{\sqrt{T}},\ \forall t = 1 \dots T$. Thus, $\gamma = \frac{1}{\sqrt{T}}$ and $\frac{R_2^2}{\gamma^2} = T$. Hence,

$$k = \left(\frac{R_2}{\gamma}\right)^2$$

Theorem 3.2 (Number of mistakes in the linearly non-separable case [**?**])**.** *Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), .., (\mathbf{x}_T, y_T)\}$, where T is the number of trials. Let $\gamma > 0$ and $R_2 = \max_t\|\mathbf{x}_t\|_2$. Then for any $\mathbf{u} \in \mathbb{R}^d$ (s.t. $\|\mathbf{u}\|_2 = 1$), the number of mistakes done by Perceptron Algorithm in T trials is upper bounded as follows:*

$$k \leq \left(\frac{R_2 + D}{\gamma}\right)^2$$

Where

$$\begin{aligned} D^2 &= \sum_{t=1}^{T} d_t^2 \\ d_t &= \max(0, \gamma - y_t\mathbf{u}^T\mathbf{x}_t) \end{aligned}$$

Proof. We'll be using the result obtained in the previous theorem.
We use the following transformations:

$$\mathbf{x}'_t = [x_{t,1}, x_{t,2}, ..., x_{t,d}, 0, 0, ..., \Delta, .., , 0] \qquad (\mathbb{R}^d \to \mathbb{R}^{d+T})$$

$$\mathbf{u}' = \left[\frac{u_1}{z}, \frac{u_2}{z}, ... \frac{u_d}{z}, \frac{y_1 d_1}{\Delta z}, \frac{y_2 d_2}{\Delta z}, ... \frac{y_T d_T}{\Delta z}\right] \qquad (\mathbb{R}^d \to \mathbb{R}^{d+T})$$

While constructing $\mathbf{x}'_t$, we put the first d components same as $\mathbf{x}_t$, next T components we keep as 0 except for the $(d+t)^{th}$ position, where we add Δ. We will fix the value of Δ later. Now if we set the constraint $\|\mathbf{u}'\|_2 = 1$, we can obtain

$$z = \sqrt{1 + \frac{D^2}{\Delta^2}}$$

We see that

$$\begin{aligned}\|\mathbf{x}_t^{'}\| =& \|\mathbf{x}_t\|^2 + \Delta^2 \\ &\le R_2^2 + \Delta^2\end{aligned}$$

Now consider

$$\begin{aligned} y_t \mathbf{u}'^T \mathbf{x}'_t &= y_t \left[\frac{\mathbf{u}^T \mathbf{x}_t}{z} + \frac{\Delta y_t d_t}{\Delta z}\right] \\ &= \frac{y_t \mathbf{u}^T \mathbf{x}_t}{z} + \frac{d_t}{z} \\ &\ge \frac{y_t \mathbf{u}^T \mathbf{x}_t}{z} + \frac{\gamma - y_t \mathbf{u}^T \mathbf{x}_t}{z} \qquad (\text{since } d_t \ge \gamma - y_t \mathbf{u}^T \mathbf{x}_t) \\ &\ge \frac{\gamma}{z} \end{aligned}$$

This implies that in a higher dimension, all samples are correctly classified by a margin of at least γ/z.
Using the derivation from theorem **??**, we can now upper bound the number of mistakes k made by Perceptron Algorithm in T trials.

$$k \le \frac{(R_2^2 + \Delta^2)}{\gamma^2/z^2} \qquad (6)$$

Our calculations have been done by parameterizing the result obtained in the previous section by Δ. To find the least upper bound on number of mistakes k, we find the minimum for the expression on the RHS of eq. (**??**). The minimum for the above expression is obtained by differentiating it with respect to Δ. We can analytically find that the minimum happens when

$$\Delta = \sqrt{RD}$$

Substituting it in value of k above,

$$k \le \left(\frac{R_2 + D}{\gamma}\right)^2$$

Hence Proved

□

References

[1] Yoav Freund and Robert E. Schapire *Large Margin Classification Using the Perceptron Algorithm* 1999: Machine Learning, Springer.