Topic 2 **July 20,2017**

Linear Predictor

About: Machine Learning **Author: Yi-Shan Wu**

This note is written according to Understanding Machine Learning: From Theory to Algorithm, and the lecture of "CS229: Machine Learning" by Andrew Ng. This week, we are going to introduce some linear predictors, including linear classifier, linear regression and logistic regression. These three methods cover a lot of territory in learning.

1 Introduction

In general, $x \in \mathbb{R}^d$ is a vector of features. We can estimate x by give weights to every feature in x and sum them up. That is, we can define "feature" functions $L_d : \mathbb{R}^d \to \mathbb{R}$ as

$$L_d = \{\mathbf{x} \mapsto < \mathbf{w}, \mathbf{x} > : \mathbf{w} \in R^d\}$$

The different hypothesis classes for different linear predictors are defined as a composition function $H = \phi \circ L_d$, where $L_d : \mathbb{R}^d \to \mathbb{R}$ and $\phi : \mathbb{R} \to Y$. Y will be specified in the following sections and the goal in this lecture is to see how to define loss function and how to learn the hypothesis class H in each case.

2 Linear Classifier(Halfspaces)

In this case, $Y = \{+1, -1\}$. That is, $\phi : \mathbb{R} \to \{+1, -1\}$ is a "sign" function. Thus,

$$H = sign \circ L_d = \{\mathbf{x} \mapsto sign(< \mathbf{w}, \mathbf{x} >) : \mathbf{w} \in R^d\}$$

We can consider the case $d = 3$, then the hypothesis class is a plane and samples are labeled differently in each side of the plane.
To learn the halfspaces, first of all, since VC dimension of this class is d (finite), we can use ERM paradigm to learn the class. Second, we assume that samples are "separable" since the nonseparable case is known to be NP hard. (PLA will not stop in nonseparable case).

2.1 Linear Programming for Halfspaces

Generally, linear programming is of the form

$$max_{\mathbf{w} \in \mathbb{R}^\mathbf{d}} \quad < \mathbf{w}, \mathbf{u} >$$
$$\textbf{subject to } A\mathbf{w} \geq \mathbf{b}$$

where A is a matrix, w and b are vectors.
Assume that there are m training data. To find a halfspace which separates data is equivalent to find a vector $\mathbf{w}$ such that,

$$\forall i \in [m], \qquad sign(< \mathbf{w}, \mathbf{x_i} >) = y_i$$

or equivalently,

$$\forall i \in [m], \qquad y_i < \mathbf{w}, \mathbf{x_i} > > 0$$

Since LP solver is able to solve linear programming problems with conditions $AX \geq b$ efficiently. It is better to normalize these equations to obtain "$\geq$" instead of ">". Let $\mathbf{w}^*$ be a vector satisfies these conditions. Define $\gamma = min_{i \in [m]}(y_i(< \mathbf{w}^*, \mathbf{x_i} >))$ and let $\hat{\mathbf{w}} = \frac{\mathbf{w}^*}{\gamma}$. Thus,

$$\forall i \in [m], \qquad y_i < \hat{\mathbf{w}}, \mathbf{x_i} > = \frac{1}{\gamma} y_i < \mathbf{w}^*, \mathbf{x_i} > \geq 1$$

We have then shown there exists a vector such that

$$\forall i \in [m], \qquad y_i < \mathbf{w}, \mathbf{x_i} > \geq 1$$

Also, since the vectors satisfy the condition functions are feasible solutions, we can just set a dummy objective function as $u = \{0, 0, \cdots, 0\}$

2.2 Perceptron Learning Algorithm for Halfspaces

Perceptron is an iterative algorithm updates vector $\mathbf{w^{t+1}}$ according to the sample (x_i, y_i) mislabeled by $\mathbf{w^t}$

Data: training set $S = \{(x_1, y_1) \cdots (x_m, y_m)\}$
Initialisation: $\mathbf{w^{(1)}} = (0, \cdots, 0)$
for *t=1,2,...* **do**
  if *($\exists i$ s.t.* $y_i < \mathbf{w}, \mathbf{x_i} > \leq 0$*)* **then**
    $\mathbf{w^{(t+1)}} = \mathbf{w^{(t)}} + y_i \mathbf{x_i}$;
  else
    output $\mathbf{w^{(t)}}$
  end
end

Algorithm 1: Perceptron learning algorithms

Then, the following theorem shows that the algorithm is guaranteed to converge.

Theorem 1 *Assume that samples are separable. Let* $B = min\{\|\mathbf{w}\| : \forall i \in [m], y_i < \mathbf{w}, \mathbf{x_i} > \geq 1\}$*, and let* $R = max_i \|\mathbf{x_i}\|$*. Then, PLA stops after at most* $(RB)^2$ *iterations with all the conditions hold.*

Proof: (sketch)

1. First, show that the update rule $\mathbf{w^{(t+1)}} = \mathbf{w^{(t)}} + y_i \mathbf{x_i}$ really make the originally mislabeled data (x_i, y_i) "more correct". That is,

$$y_i(< \mathbf{w^{(t+1)}}, \mathbf{x_i} >) > y_i(< \mathbf{w^{(t)}}, \mathbf{x_i} >)$$

2. Second, if the Perceptron runs for T iterations,and if $\mathbf{w}^*$ is the vector that achieves the minimum in B, then we must have $T \leq (RB)^2$ by showing

$$1 \geq \frac{< \mathbf{w}^*, \mathbf{w}^{(\mathbf{T+1})} >}{\|\mathbf{w}^*\| \|\mathbf{w}^{(\mathbf{T+1})}\|} \geq \frac{\sqrt{T}}{RB} \tag{1}$$

3. To show equation (1) holds, we first proof that $< \mathbf{w}^*, \mathbf{w}^{\mathbf{T+1}} > \geq T$.
   Since

$$< \mathbf{w}^*, \mathbf{w}^{(\mathbf{t+1})} > - < \mathbf{w}^*, \mathbf{w}^{(\mathbf{t})} > \geq 1$$

   Also,

$$\mathbf{w}^{(\mathbf{1})} = (0, \cdots, 0)$$

   Therefore,

$$< \mathbf{w}^*, \mathbf{w}^{\mathbf{T+1}} > = \sum_{t=1}^{T} (< \mathbf{w}^*, \mathbf{w}^{(\mathbf{t+1})} > - < \mathbf{w}^*, \mathbf{w}^{(\mathbf{t})} >) \geq T \tag{2}$$

   Second, since

$$\begin{aligned} \|\mathbf{w}^{(\mathbf{t+1})}\|^2 &= \|\mathbf{w}^{(\mathbf{t})} + y_i \mathbf{x_i}\|^2 \\ &\leq \|\mathbf{w}^{(\mathbf{t})}\|^2 + R^2 \end{aligned}$$

   We have,

$$\|\mathbf{w}^{(\mathbf{T+1})}\|^2 \leq TR^2 \tag{3}$$

   Third, since

$$R = max_i \|\mathbf{x_i}\|$$

   Therefore,

$$\|\mathbf{w}^*\| \leq R \tag{4}$$

4. With equations (2),(3), and (4), we finally prove equation (1)

■

3 Linear Regression

In this case, $Y = \mathbb{R}$. That is, $\phi : \mathbb{R} \to \mathbb{R}$ is a linear function. Thus,

$$H = \phi \circ L_d = \{\mathbf{x} \mapsto < \mathbf{w}, \mathbf{x} > : \mathbf{w} \in R^d\}$$

For the case in section 2, the loss function is 0/1 loss. That is, the algorithm suffers 1 loss when a sample is mislabeled. In this section, we usually use "squared loss", measuring the distance between data and the hypothesis. Thus, if we have m training samples,

$$L_D(h_w) = \mathop{\mathrm{E}}_{(x,y)\sim D}[(< \mathbf{w}, \mathbf{x} > -y)^2]$$

$$L_S(h_w) = \frac{1}{m} \sum_{i=1}^{m} (< \mathbf{w}, \mathbf{x_i} > -y_i)^2$$

If there is an exact solution, the minimum error is 0. However, in the more likely case that y is not proportional to x. Thus, we can only find a $\hat{\mathbf{w}}$ that minimize the error.

3.1 Normal equations

There are several approach to find a vector $\mathbf{w}$ minimize the loss function. One of them is to solve the problem by linear algebra. The linear equations

$$\begin{aligned}\mathbf{x_1^T w} &= y_1\\ \mathbf{x_2^T w} &= y_2\\ &\vdots\\ \mathbf{x_m^T w} &= y_m\end{aligned}$$

can be rewritten in matrix form

$$X\mathbf{w} = Y \tag{5}$$

where $X = \begin{bmatrix} -\mathbf{x_1^T}-\\ -\mathbf{x_2^T}-\\ \vdots\\ -\mathbf{x_m^T}-\end{bmatrix}$, $Y = \begin{bmatrix} y_1\\ y_2\\ \vdots\\ y_m\end{bmatrix}$, and usually $m >> d$ (dimension of w and every x_i)

Also, the loss function is

$$L_S(h_w) = \frac{1}{m}\|X\mathbf{w} - Y\|^2$$

3.1.1 geometric approach

To minimize $L_S(h_w)$ is to find the minimum distance between Y and some point p on the space spanned by X. Thus, p must be the projection of Y on $span(X)$, $p = X\hat{\mathbf{w}}$. Furthermore, $X\hat{\mathbf{w}} - Y$ must be perpendicular to vectors in $span(X)$. Therefore, we have

$$X^T(X\hat{\mathbf{w}} - Y) = 0$$

Thus, we have

$$\hat{\mathbf{w}} = (X^TX)^{-1}X^TY \overset{\text{def}}{=} X^\dagger Y \tag{6}$$

and

$$\hat{Y} = X\hat{\mathbf{w}} = XX^\dagger Y \tag{7}$$

Note that equation (6) is a closed form solution. However, there is still a problem. Although we have closed form solution for $L_S(h_w)$, how can we guarantee that $L_D(h_w)$ is close to $L_S(h_w)$?

3.2 Analysis for linear algebra approach

Let $P = XX^\dagger$ be the projection matrix that project Y onto $span(X)$. Then, there are several useful properties for P. We will further use these properties to help up estimate $L_S(h)$ and $L_D(h)$ and conclude that the more the training samples, the closer the two error terms.

3.2.1 Properties for $P = XX^\dagger$

Proposition 2 *$P = X(X^TX)^{-1}X^T$, where X is a m by d matrix and assume that X^TX is invertible (usually hold since $m >> d$).*

1. *P is symmetric.*
$$\because P^T = (X(X^TX)^{-1}X^T)^T = X(X^TX)^{-1}X^T = P$$

2. *$P^K = P$ for any positive integer K*
$$\because H^2 = X(X^TX)^{-1}X^TX(X^TX)^{-1}X^T = X(X^TX)^{-1}X^T = P$$

3. *If I is the identity matrix of size m, $(I_m - P)^K = I_m - P$ for any positive integer K*
Can be shown by similar method as 2.

4. *$trace(P) = d$*
$$trace(X(X^TX)^{-1}X^T) = trace(X^TX(X^TX)^{-1}) = trace(I_d) = d$$

We further assuming that the target variables and the inputs are related by

$$y_i = \mathbf{w}^*\mathbf{x_i} + \epsilon_\mathbf{i}$$

where $\epsilon_\mathbf{i}$ is a noise term with 0 mean and σ^2 variance, independently generated for every sample (x,y). Let $\epsilon = [\epsilon_1, \epsilon_2, \cdots, \epsilon_m]^T$. Then,

$$Y = X\mathbf{w}^* + \epsilon$$

Still, $\hat{Y}$ is the projection of Y on the space spanned by X. Since $X\mathbf{w}^*$ is already on the space

$$\hat{Y} = PY = X\mathbf{w}^* + P\epsilon \tag{8}$$

3.2.2 estimate $L_S(h_w)$

$$L_S(h_w) = \frac{1}{m}\|Y - \hat{Y}\|^2 \tag{9}$$
$$= \frac{1}{m}\|(I_m - P)\epsilon\|^2 \tag{10}$$
$$= \frac{1}{m}\epsilon^T(I_m - P)\epsilon \tag{11}$$
$$= \frac{1}{m}\epsilon^T\epsilon - \frac{1}{m}\epsilon^TP\epsilon \tag{12}$$

Since ϵ_i are distributed IID, we now calculate the expectation of $L_S(h_w)$

$$\mathop{\mathrm{E}}_{\epsilon_i\sim D}[L_S(h_w)] = \mathop{\mathrm{E}}_{\epsilon_i\sim D}[\frac{1}{m}\epsilon^T\epsilon - \frac{1}{m}\epsilon^TP\epsilon] \tag{13}$$
$$= \frac{1}{m}\mathop{\mathrm{E}}_{\epsilon_i\sim D}[\sum_{i=1}^{m}\epsilon_i^2] - \frac{1}{m}\mathop{\mathrm{E}}_{\epsilon_i\sim D}[\sum_{j=1}^{m}\epsilon_j\sum_{i=1}^{m}P_{ji}\epsilon_i] \tag{14}$$

Also,

$$\mathop{\mathrm{E}}_{\epsilon_i \sim D}[\sum_{j=1}^{m} \epsilon_j \sum_{i=1}^{m} P_{ji}\epsilon_i] = \mathop{\mathrm{E}}_{\epsilon_i \sim D}[\sum_{i}^{m} \epsilon_i^2 P_{ii}] + \mathop{\mathrm{E}}_{\epsilon_j \neq \epsilon_i}[\epsilon_j] P \mathop{\mathrm{E}}_{\epsilon_i \sim D}[\epsilon_i] \tag{15}$$

$$= tr(P) \mathop{\mathrm{E}}_{\epsilon_i \sim D}[\epsilon_i^2] + 0 \tag{16}$$

Thus,

$$\mathop{\mathrm{E}}_{\epsilon_i \sim D}[L_S(h_w)] = \frac{1}{m}(\mathop{\mathrm{E}}_{\epsilon_i \sim D}[\sum_{i=1}^{m} \epsilon_i^2] - tr(P) \mathop{\mathrm{E}}_{\epsilon_i \sim D}[\epsilon_i^2]) \tag{17}$$

$$= \frac{1}{m}(m\sigma^2 - d\sigma^2) \tag{18}$$

$$= \sigma^2(1 - \frac{d}{m}) \tag{19}$$

3.2.3 estimate $L_D(h_w)$

We choose another "testing sample" $(\mathbf{x_t}, y_t)$, with error ϵ_t. (testing sample is sampled i.i.d. from the same distribution D).
Thus,

$$y_t = \mathbf{w}^* \mathbf{x_t} + \epsilon_\mathbf{t}$$

and

$$\hat{y}_t = \mathbf{x_t^T} \hat{\mathbf{w}} \tag{20}$$

$$= \mathbf{x_t^T} (X^T X)^{-1} X^T Y \tag{21}$$

$$= \mathbf{x_t^T} (X^T X)^{-1} X^T (X\mathbf{w}^* + \epsilon) \tag{22}$$

$$= \mathbf{x_t^T} \mathbf{w}^* + \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon \tag{23}$$

Note that $\hat{\mathbf{w}}$ is the same as the one in equation (7) since we use this output $\hat{\mathbf{w}}$ to test new samples.
Thus,

$$y_t - \hat{y}_t = \epsilon_t - \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon$$

Then we can take the expectation with respect to the test point ot obtain an expression for $L_D(h_w)$

$$L_D(h_w) = \mathop{\mathrm{E}}_{x_t, \epsilon_t}[(y_t - \hat{y}_t)^2] \tag{24}$$

$$= \mathop{\mathrm{E}}_{x_t, \epsilon_t}[(\epsilon_t - \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon)^T (\epsilon_t - \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon)] \tag{25}$$

$$= \mathop{\mathrm{E}}_{x_t, \epsilon_t}[(\epsilon_t^T \epsilon_t)] + \mathop{\mathrm{E}}_{x_t, \epsilon_t}[\epsilon^T X (X^T X)^{-1} \mathbf{x_t} \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon] \tag{26}$$

$$= \mathop{\mathrm{E}}_{x_t, \epsilon_t}[(\epsilon_t^T \epsilon_t)] + tr \mathop{\mathrm{E}}_{x_t, \epsilon_t}[\epsilon^T X (X^T X)^{-1} \mathbf{x_t} \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon] \tag{27}$$

$$= \sigma^2 + tr \mathop{\mathrm{E}}_{x_t, \epsilon_t}[\mathbf{x_t} \mathbf{x_t^T} (X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1}] \tag{28}$$

Then take the expectation with respect to ϵ to show that, on average,

$$\mathop{\mathrm{E}}_{\epsilon_i \sim D}[L_D(h_w)] = \sigma^2 + \frac{\sigma^2}{m} tr(\mathbf{x_t} \mathbf{x_t^T} (\frac{1}{m} X^T X)^{-1})$$

Then, with high probability,

$$\mathop{\mathrm{E}}_{\epsilon_i \sim D}[L_D(h_w)] = \sigma^2(1 + \frac{d}{m} + o(\frac{1}{m})) \tag{29}$$

Compare equation (29) with equation (19)

$$\mathop{\mathrm{E}}_{\epsilon_i \sim D}[L_S(h_w)] = \sigma^2(1 - \frac{d}{m})$$

We have that when receiving more training samples, $L_D(h_w)$ and $L_S(h_w)$ tend to be close to one another.

3.3 Gradient descent

Sometimes it costs a lot to calculate large matrices, so we turn to find other methods for linear regression. Since $L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m}(< \mathbf{w}, \mathbf{x_i} > -y_i)^2$ is actually a convex quadratic function, the local minimum is also the global minimum. We can solve the problem by gradient descent. The algorithm is done by iteratively update $\mathbf{w}$ according to training data. That is,

$$\mathbf{w} := \mathbf{w} - \eta \frac{\partial}{\partial \mathbf{w}} L_S(\mathbf{w}) \tag{30}$$

where η is "learning rate"
More specifically, if we denote $(\mathbf{x^{(i)}}, y^{(i)})$ as the i'th sample and denote w_j as the j'th component of vector $\mathbf{w}$, then every component of $\mathbf{w}$ is updated in each iteration by

$$w_j := w_j - \eta \frac{\partial}{\partial w_j} L_S(w_j^t) \tag{31}$$

$$:= w_j - \eta \sum_{i=1}^{m}(< \mathbf{w}, \mathbf{x^{(i)}} > -y^{(i)}) x_j^{(i)} \tag{32}$$

The algorithm runs until converge (every component in $\mathbf{w}$ doesn't change much).

3.4 Stochastic gradient descent

We may find that it still takes long time to calculate the summation in equation (22), since m is often a large number. Therefore, we can turn to use "stochastic" gradient descent. The only difference is the update rule:

$$w_j := w_j - \eta(< \mathbf{w}, \mathbf{x^{(i)}} > -y^{(i)}) x_j^{(i)} \tag{33}$$

In this algorithm, we encounter a training example each iteration, and we update the parameters according to the gradient of the error with respect to that single training example only. This algorithm expectedly get close to minimum faster than original gradient descent. Thus, when the training set is large, stochastic gradient descent is often preferred.

4 Logistic Regression

The logistic regression can be seen as a "soft" binary classifier (Here we consider binary case, but it can be generalized to multi-class). This method is used for problems asking: what's the probability of output to be $+1$ given a feature x? Since we want the output to be a probability between 0 and 1 according to $\mathbf{w^T x}$, we need a monotonic function $\theta : \mathbb{R} \to [0, 1]$. Thus, in this case,

$$H = \theta \circ L_d = \{\mathbf{x} \mapsto \theta(\mathbf{w^T x}) : \mathbf{w} \in R^d\}$$

where $\theta()$ is the monotonic function called "logistic function" maps $\mathbb{R}$ to $[0, 1]$. We then introduce the logistic function in the following subsection.

4.1 logistic function

$$\theta(a) = \frac{1}{1 + e^{-a}}$$

In our case, $a = \mathbf{w^T x}$ is the score of feature $\mathbf{x}$. Then, the hypothesis

$$h_{\mathbf{w}}(\mathbf{x}) = \theta(\mathbf{w^T x}) = \frac{1}{1 + e^{-\mathbf{w^T x}}}$$

Note that if $\mathbf{w^T x} \to \infty$, then the output is close to 1. If $\mathbf{w^T x} \to -\infty$, then the output is close to 0. That is, the higher the score, the higher the probability that output is $+1$. Also, if $\mathbf{w^T x} = 0$, then the output is $\frac{1}{2}$, which means we get little information from score. Thus, we have to guess whether y is $+1$ with probability equals to $\frac{1}{2}$ in this case.

4.2 predict a probability

In logistic regression, the training samples are still of the form which is the same as binary classification: $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, where $x \in \mathbb{R}^d$ and $y = \{+1, -1\}$. However, we are trying to learn a target function f with output between 0 and 1.

$$f(x) = \Pr[y = +1|x]$$

Reversely, the data points are generated by

$$P(y|x) = \begin{cases} f(x) & \text{for } y = +1 \\ 1 - f(x) & \text{for } y = -1 \end{cases}$$

The question is: "How can we learn a probability function from these data?" Suppose there are n training samples $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$. The goal is to find a hypothesis $h_{\mathbf{w}}(x)$ such that $h_{\mathbf{w}}$ performs similar to f on S.

probability that f generate S	likelihood that $h_{\mathbf{w}}$ generate S
$P(x_1)f(x_1)\times$	$P(x_1)h_{\mathbf{w}}(x_1)\times$
$P(x_2)(1 - f(x_2))\times$	$P(x_2)(1 - h_{\mathbf{w}}(x_2))\times$
$P(x_3)f(x_3)\times$	$P(x_3)h_{\mathbf{w}}(x_3)\times$
$\vdots$	$\vdots$

Table 1: Consider that $S = \{(x_1, +1), (x_2, -1), (x_3, +1), \cdots\}$. The goal is to find a function h that maximize the likelihood to label samples similar to f

Since the closer the hypothesis $h_{\mathbf{w}}$ to f, the higher the likelihood that $h_{\mathbf{w}}$ label S similar to f, the objective function is

$$\mathbf{max_w} : h_{\mathbf{w}}(x_1) \times (1 - h_{\mathbf{w}}(x_2)) \times h_{\mathbf{w}}(x_3) \times \cdots \tag{34}$$

Note that since hypothesis h is logistic function, we have

$$1 - h_{\mathbf{w}}(x) = h_{\mathbf{w}}(-x)$$

Furthermore,

$$\begin{cases} 1 - h_{\mathbf{w}}(x) = h_{\mathbf{w}}(-x) = \theta(-\mathbf{w^T x}) = \theta(y\mathbf{w^T x}) & \text{for } y = -1 \\ h_{\mathbf{w}}(x) = \theta(\mathbf{w^T x}) = \theta(y\mathbf{w^T x}) & \text{for } y = +1 \end{cases}$$

Therefore, equation (24) can be reformed as

$$\mathbf{max_w} \quad \Pi_{i=1}^{n} \theta(y_i \mathbf{w^T x_i}) \tag{35}$$

Or, we can add "ln" to the objective function for easier calculation.

$$\mathbf{max_w} \quad ln(\Pi_{i=1}^{n} \theta(y_i \mathbf{w^T x_i})) \tag{36}$$

$$= \sum_{i=1}^{n} ln(\theta(y_i \mathbf{w^T x_i})) \tag{37}$$

Furthermore, it is equivalent to

$$\mathbf{min_w} \quad \sum_{i=1}^{n} -ln(\theta(y_i \mathbf{w^T x_i})) \tag{38}$$

$$= \sum_{i=1}^{n} ln(1 + \exp(-y_i \mathbf{w^T x_i})) \tag{39}$$

4.3 Gradient descent for logistic regression

The objective function is now clear:

$$\mathbf{min_w} \quad \sum_{i=1}^{n} ln(1 + \exp(-y_i \mathbf{w^T x_i})) \tag{40}$$

Still, we may find out that it is a twice-differentiable convex function and the hypothesis set is a convex set. It can be solved by

$$\nabla_{\mathbf{w}} L_S(\mathbf{w}) = \nabla_{\mathbf{w}} \sum_{i=1}^{n} ln(1 + \exp(-y_i \mathbf{w^T x_i}))$$

$$= \sum_{i=1}^{n} (1 + \exp(-y_i \mathbf{w^T x_i}))(-y_i \mathbf{x_i}) = 0$$

Unfortunately, there is no close form solution to the problem. Thus, we can use gradient descent to solve this problem.

$$\mathbf{w} := \mathbf{w} - \eta \frac{\partial}{\partial \mathbf{w}} L_S(\mathbf{w}) \tag{41}$$

$$= \mathbf{w} - \eta \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^{n} (1 + \exp(-y_i \mathbf{w}^{\mathbf{T}} \mathbf{x_i}))(-y_i \mathbf{x_i}) \tag{42}$$

where η is "learning rate". The algorithm runs until $\mathbf{w}$ converge.

5 Convex learning problem

5.1 Convexity

Definition 3 *(Convex set)*
A set C in a vector space is convex if for any two vectors $\mathbf{u}, \mathbf{v}$ in C, the line segment between $\mathbf{u}$ and $\mathbf{v}$ is contained in C. That is, for any $\alpha \in [0,1]$ we have that $\mathbf{u} + (1-\alpha)\mathbf{v} \in C$

Definition 4 *(Convex Function)*
A function $f : C \to \mathbb{R}$ is convex if C is a convex set and for every $\mathbf{u}, \mathbf{v} \in C$ and $\alpha \in [0,1]$,

$$f(\mathbf{u} + (1-\alpha)\mathbf{v}) \leq \alpha f(\mathbf{u}) + (1-\alpha) f(\mathbf{v})$$

Proposition 5 *Every local minimum of a convex function is also a global minimum.*

Proof: Let $B(\mathbf{u}, r) = \{\mathbf{v} : \|\mathbf{v} - \mathbf{u}\| \leq r\}$ be a ball of radius r centered around $\mathbf{u}$. We say that $f(\mathbf{u})$ is a local minimum of f at $\mathbf{u}$ if there exists some $r > 0$ such that for all $\mathbf{v} \in B(\mathbf{u}, r)$, we have $f(\mathbf{v} \geq f(\mathbf{u})$. Furthermore, for any $\mathbf{v}$ (not necessarily in B), there is a small enough $\alpha > 0$ such that $\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u}) \in B(\mathbf{u}, r)$.
Therefore,

$$f(\mathbf{u}) \leq f(\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u}))$$

and if f is convex, then

$$f(\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u})) = f(\alpha \mathbf{v} + (1-\alpha)\mathbf{u}) \leq (1-\alpha) f(\mathbf{u}) + \alpha f(\mathbf{v})$$

We than conclude that

$$f(\mathbf{u}) \leq f(\mathbf{v})$$

Since this holds for every $\mathbf{v}$, it follows that $\mathbf{u}$ is also a global minimum of f. ■

Proposition 6 *(First-order condition)*
Suppose a function $f : C \to \mathbb{R}$ is differentiable. Then f is convex if and only if C is a convex set and $\forall \mathbf{u}, \mathbf{w} \in C$,

$$f(\mathbf{u}) \geq f(\mathbf{w}) + < \bigtriangledown f(\mathbf{w}), \mathbf{u} - \mathbf{w} >$$

Proposition 7 *(Second-order condition)*
Suppose a function $f : C \to \mathbb{R}$ is twice differentiable. Then f is convex if and only if C is a convex set and its Hessian is positive semidefinite. That is,$\forall \mathbf{u} \in C$,

$$\bigtriangledown^2 f(\mathbf{u}) \succeq 0$$

Remark 8 If $f : \mathbb{R} \to \mathbb{R}$ is twice differentiable, then the following are equivalent:

1. f is convex
2. f' is monotonically nondecreasing
3. f'' is nonnegative

Example 9 (Examples of convex function)

1. $f(x) = x^2$ is convex.
   $\because f'(x) = 2x$ and $f''(x) = 2 > 0$
2. $f(x) = log(1 + exp(x))$ is convex.
   $\because f'(x) = \frac{1}{1+exp(-x)}$ is actually sigmoid function. And the sigmoid function is monotonically nondecreasing function.

Claim 10 *Assume that $f : \mathbb{R}^d \to \mathbb{R}$ can be written as $f(w) = g(< \mathbf{w}, \mathbf{x} > +y)$, for some $x \in \mathbb{R}^d$, $y \in \mathbb{R}$,and $g : \mathbb{R} \to \mathbb{R}$. Then, convexity of g implies the convexity of f.*

Claim 11 *For $i = 1, \cdots, r$, let $f_i : \mathbb{R}^d \to \mathbb{R}$ be a convex function. The following functions from $\mathbb{R}^d$ to $\mathbb{R}$ are also convex.*

- $g(x) = max_{i\in[r]} f_i(x)$
- $g(x) = \Sigma_{i\in[r]} f_i(x)$

With example 9 and claim 10, 11, we know that:

1. The loss function we saw in linear regression: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$ is convex.
2. The loss function we saw in logistic regression: $\sum_{i=1}^{n} ln(1 + \exp(-y_i \mathbf{w^T x_i}))$ is convex.

5.2 Jensen's inequality

Using induction, it is not hard to extended to convex combinations of more than one point, then we have

$$f(\sum_i \alpha_i x_i) \leq \sum_i \alpha_i f(x_i)$$

for $\sum_i \alpha_i = 1$ and $\forall i, \alpha_i \geq 0$

5.3 Convex optimization problem

A convex optimization problem is of the form

$$\min f(x) \tag{43}$$

$$\text{subject to } x \in C \tag{44}$$

where where f is a convex function, C is a convex set, and x is the optimization variable. It means that the convex optimization problem is to find the minimum of the convex function f on a convex set C. Convex optimization problems can usually be solved efficiently.

5.4 Convex learning problem

Definition 12 *(Convex learning problem)*
A convex learning problem is a problem whose hypothesis class H is a convex set, and for all $z \in Z$, the loss function l is a convex function.

It is then not hard to show that if l is a convex function and H is a convex hypothesis set, then ERM_H is a convex optimization problem. Thus, there are some efficient algorithms to solve the problem. One of them is actually gradient descent. But is convexity a sufficient condition for the learnability of a problem? The answer is negative. Here is an example.

Example 13 (Ex 12.8 in the textbook)
Let $H = R$, and the loss be the squared loss: $(wx-y)^2$, and A is a deterministic algorithm. Assume that A is PAC learnable algorithm. That is, there exists a $m_H(\epsilon, \delta)$ such that for every distribution D and $\epsilon, \delta \in (0,1)$, when $m = |S| > m_H$, we have

$$\Pr[L_D(A(S)) - min_{h \in H} L_D(h) \geq \epsilon] \leq 1 - \delta$$

Choose $\epsilon = 1/100, \delta = 1/2$ and define $\mu = \frac{log(100/99)}{2m}$. Here we set two possible distributions D_1 and D_2

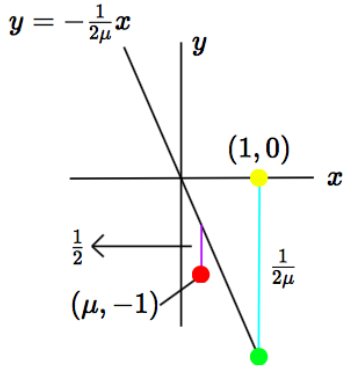
prob	(1,0)	(μ,-1)
D_1	μ	$1-\mu$
D_2	0	1

Table 2: Distribution of two distributions

Since μ is small, for both distributions, the probability that all examples of the training set will be at $(\mu, -1)$ is at least 99%. Furthermore, since the algorithm is deterministic, upon receiving data, the output is determined and it is trivial that $-\frac{1}{\mu} < \hat{w} < 0$
Suppose all the training examples are at $(\mu, -1)$, then A may output either $0 > \hat{w} \geq -\frac{1}{2\mu}$ or $-\frac{1}{\mu} < \hat{w} < -\frac{1}{2\mu}$.

1. If A outputs $0 > \hat{w} \geq -\frac{1}{2\mu}$, then we set the distribution to be D_2.
2. If A outputs $-\frac{1}{\mu} < \hat{w} < -\frac{1}{2\mu}$, then we set the distribution to be D_2.

Both cases can cause A to fail, which implies that the problem is not PAC learnable.



5.4.1 Some other constraints

From the above example we know that convexity is not sufficient for PAC learnable. A possible solution to this problem is to add some constraints on the hypothesis class.
In addition to the convexity requirement, we require that H will be bounded; namely, we assume that for some predefined scalar B, every hypothesis $w \in H$ satisfies $\|w\| \leq B$. However, boundedness and convexity alone are still not sufficient for learnability. Here is another example

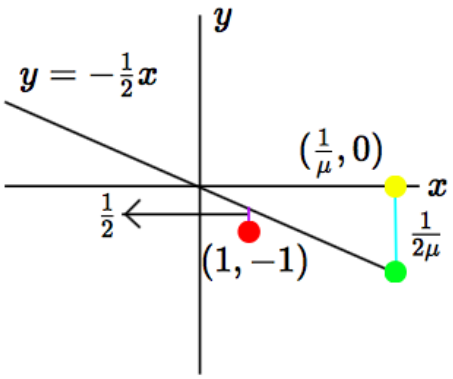
Example 14 (Ex 12.9 in the textbook)
The basic setting are the sample as Example 13 except that this time $H = \{w : \|w\| \leq 1\} \subset \mathbb{R}$ and the distribution now is

prob	($\frac{1}{\mu}$,0)	(1,-1)
D_1	μ	$1-\mu$
D_2	0	1

Table 3: Distribution of two distributions

In this case, it is trivial that $-1 < \hat{w} < 0$. Thus, w satisfies the constraint $\|w\| \leq 1$.
Suppose all the training examples are at $(1, -1)$, then A may output either $0 > \hat{w} \geq -\frac{1}{2}$ or $-1 < \hat{w} < -\frac{1}{2}$.

1. If A outputs $0 > \hat{w} \geq -\frac{1}{2}$, then we set the distribution to be D_2.
2. If A outputs $-1 < \hat{w} < -\frac{1}{2}$, then we set the distribution to be D_2.



With similar flow we can prove that the algorithm fails in this case.

In addition to boundedness and convexity, the possible solution is in Lipschitzness or smoothness of the loss function. It is claimed that the following problems are learnable.

1. Convex-Smooth-Bounded Learning Problem
2. Convex-Lipschitz-Bounded Learning Problem

Since we haven't defined smoothness and lipschitzness, we will simply give the definition of these two problems without proof.

Definition 15 *(Convex-Lipschitz-Bounded Learning Problem) A learning problem, (H, Z, l), is called Convex-Lipschitz-Bounded, with parameters ρ, B if the following holds:*

- *H is a convex set and $\forall w \in H$, $\|w\| \leq B$.*
- *$\forall z \in Z$, the loss function $l(h, z)$, is convex and ρ-Lipschitz.*

Definition 16 *(Convex-Smooth-Bounded Learning Problem) A learning problem, (H, Z, l), is called Convex-Smooth-Bounded, with parameters β, B if the following holds:*

- *H is a convex set and $\forall w \in H$, $\|w\| \leq B$.*
- *$\forall z \in Z$, the loss function, $l(h, z)$ is convex, nonnegative, and β-smooth.*