ELEG/CISC 867: Advanced Machine Learning **Spring 2019**

Lecture 7: Linear Predictors

Lecturer: Xiugang Wu *03/19/2019, 03/21/2019*

From this lecture, we will start our study on machine learning algorithms. The first family of hypothesis classes we will discuss is linear predictors, which includes the hypothesis classes of halfspaces, linear regression predictors, and logistic regression predictors. To see that these three hypothesis classes belong to the same family of linear predictors, define the class of affine functions as

$$\mathcal{L}_d = \{h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where

$$h_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b.$$

Then the aforementioned three hypothesis classes of linear predictors can be viewed as compositions of a function $\phi : \mathbb{R} \to \mathcal{Y}$ on $\mathcal{L}_d$. For example, in binary classification where $\mathcal{Y} = \{-1, +1\}$, we choose ϕ to be the sign function

$$\phi(a) = \text{sgn}(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases},$$

and for regression problems where $\mathcal{Y} = \mathbb{R}$, ϕ is simply the identity function $\phi(a) = a$.

1 Halfspaces

Consider a binary classification problem, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$. The class of halfspaces is defined as:

$$\begin{aligned} \mathcal{H}_{\text{HS}} = \text{sgn} \circ \mathcal{L}_d &= \{\text{sgn} \circ h_{\mathbf{w},b} : h_{\mathbf{w},b} \in \mathcal{L}_d\} \\ &= \{\mathbf{x} \mapsto \text{sgn} \circ h_{\mathbf{w},b}(\mathbf{x}) : h_{\mathbf{w},b} \in \mathcal{L}_d\}. \end{aligned}$$

In other words, each halfspace hypothesis in $\mathcal{H}_{\text{HS}}$ is parameterized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ and returns the label $\text{sgn}(\mathbf{w}^T\mathbf{x} + b)$ upon receiving feature vector $\mathbf{x}$.

It turns out that for the class of homogenous halfspaces $\text{sgn}(\mathbf{w}^T\mathbf{x})$ in R^d, its VC dimension is d; for the general class of nonhomogenous halfspaces $\text{sgn}(\mathbf{w}^T\mathbf{x} + b)$ in R^d, its VC dimension is $d + 1$. Therefore, we can learn halfspaces using ERM as long as the sample size is $\Omega(\frac{d+\log(1/\delta)}{\epsilon^2})$. Below we will discuss how to implement an ERM procedure for halfspaces. In particular, we will focus on the "realizable" case, which is also known as the "separable" case in the context of halfspaces since we can separate the positive examples from the negative examples with a hyperplane. Implementing ERM for nonseparable case is known to be computationally hard.

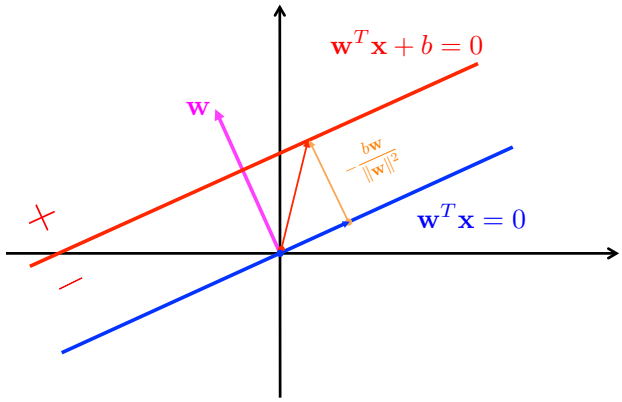

Figure 1: Halfspace.

1.1 Linear Programming for Halfspaces

Recall a Linear Program (LP) of the following form:

$$\begin{aligned} \max_{\mathbf{w}\in\mathbb{R}^d} \quad & \mathbf{w}^T\mathbf{u} \\ \text{s.t.} \quad & A\mathbf{w} \geq \mathbf{v} \end{aligned}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the optimization variable, $\mathbf{w}^T\mathbf{u}$ is the linear objective function with $\mathbf{u} \in \mathbb{R}^d$, $A\mathbf{w} \geq \mathbf{v}$ is a set of m linear inequalities with $A \in \mathbb{R}^{m\times d}$ and $\mathbf{v} \in \mathbb{R}^m$. Such linear programs can be solved efficiently (in time polynomial in m and d) and there are publicly available implementations of LP solvers.

We now show that ERM for halfspaces in the realizable case can be expressed as a linear program. First consider the homogenous case. Let $z^n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be the training set of size n. An ERM predictor should have zero error on the training set, i.e. we are looking for some $\mathbf{w} \in \mathbb{R}^d$ such that

$$\text{sgn}(\mathbf{w}^T\mathbf{x}_i) = y_i, \forall i \in [1:n],$$

i.e.,

$$(\mathbf{w}^T\mathbf{x}_i)y_i > 0, \forall i \in [1:n].$$

Let $\mathbf{w}_1$ be a vector satisfying the above condition. Define $\mathbf{w}_2$ as

$$\mathbf{w}_2 = \frac{\mathbf{w}_1}{\min_{i\in[1:n]}((\mathbf{w}_1^T\mathbf{x}_i)y_i)}.$$

Then obviously, $\mathbf{w}_2$ satisfies

$$(\mathbf{w}_2^T\mathbf{x}_i)y_i = \frac{(\mathbf{w}_1^T\mathbf{x}_i)y_i}{\min_{i\in[1:n]}((\mathbf{w}_1^T\mathbf{x}_i)y_i)} \geq 1, \forall i \in [1:n].$$

Therefore, we have shown that there exists an ERM predictor with $\mathbf{w}$ satisfying

$$(\mathbf{w}^T\mathbf{x}_i)y_i \geq 1, \forall i \in [1:n].$$

Such $\mathbf{w}$ can be found by solving the following LP using any LP solver:

$$\begin{aligned} \max_{\mathbf{w}\in\mathbb{R}^d} \quad & \mathbf{w}^T(0,0,\ldots,0) \\ \text{s.t.} \quad & \begin{bmatrix} y_1\mathbf{x}_1^T \\ y_2\mathbf{x}_2^T \\ \vdots \\ y_n\mathbf{x}_n^T \end{bmatrix} \mathbf{w} \geq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \end{aligned}$$

Now consider the nonhomogenous case where a predictor is of the form $\text{sgn}(\mathbf{w}^T\mathbf{x}+b)$. Let $\bar{\mathbf{w}} = [\mathbf{w}; b]$ and $\bar{\mathbf{x}} = [\mathbf{x}; 1]$. Then the predictor can be rewritten as $\text{sgn}(\bar{\mathbf{w}}^T\bar{\mathbf{x}})$. An $\bar{\mathbf{w}}$ for an ERM predictor in the realizable case can be found by solving the following LP:

$$\begin{aligned} \max_{\bar{\mathbf{w}}\in\mathbb{R}^{d+1}} \quad & \bar{\mathbf{w}}^T(0,0,\ldots,0) \\ \text{s.t.} \quad & \begin{bmatrix} y_1\bar{\mathbf{x}}_1^T \\ y_2\bar{\mathbf{x}}_2^T \\ \vdots \\ y_n\bar{\mathbf{x}}_n^T \end{bmatrix} \bar{\mathbf{w}} \geq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \end{aligned}$$

1.2 Perceptron for Halfspaces

A different implementation of ERM is the following Perceptron learning algorithm (PLA). It is an iterative

Algorithm 1 Batch Perceptron

input: Training sample $z^n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
initialize: $\mathbf{w}^{(1)} \leftarrow (0,0,\ldots,0)$, $t \leftarrow 1$
while ($\exists i$ s.t. $\langle \mathbf{w}^{(t)}, \mathbf{x}_i\rangle y_i \leq 0$) **do**
    $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + y_i\mathbf{x}_i$
    $t \leftarrow t+1$
end while
output $\mathbf{w}^{(t)}$

algorithm that constructs a sequence of vectors $\mathbf{w}^{(t)}, t = 1,2,3,\ldots$ At iteration t, the Perceptron finds an example i that is mislabeled by $\mathbf{w}^{(t)}$, i.e. $\langle \mathbf{w}^{(t)}, \mathbf{x}_i\rangle y_i \leq 0$, and updates $\mathbf{w}^{(t)}$ by letting $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i\mathbf{x}_i$. The intuition behind this update is that it will guide the solution to be more correct on the example i, because

$$\langle \mathbf{w}^{(t+1)}, \mathbf{x}_i\rangle y_i = \langle \mathbf{w}^{(t)}, \mathbf{x}_i\rangle y_i + \langle y_i\mathbf{x}_i, \mathbf{x}_i\rangle y_i = \langle \mathbf{w}^{(t)}, \mathbf{x}_i\rangle y_i + \|\mathbf{x}_i\|^2.$$

Indeed, the following theorem guarantees that the algorithm stops with all sample points correctly labeled in the realizable case.

Theorem 1.1 *Assume that the training set $z^n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is separable. Let B be defined as*

$$B = \min\{\|\mathbf{w}\| : \langle \mathbf{w}, \mathbf{x}_i\rangle y_i \geq 1, \forall i \in [1:n]\}$$

and let R be defined as

$$R = \max_{i\in[1:n]} \|\mathbf{x}_i\|.$$

Then the Perceptron algorithm stops after at most $(RB)^2$ iterations, and when it stops it holds that

$$\langle \mathbf{w}^{(t)}, \mathbf{x}_i\rangle y_i > 0, \forall i \in [1:n].$$

Proof: Let $\mathbf{w}^* = \operatorname{argmin}\{\|\mathbf{w}\| : \langle \mathbf{w}, \mathbf{x}_i \rangle y_i \geq 1, \forall i \in [1:n]\}$. The idea of the proof is to show that after performing T iterations, the cosine of the angle between $\mathbf{w}^*$ and $\mathbf{w}^{(T+1)}$ is at least $\frac{\sqrt{T}}{RB}$, i.e.,

$$\cos(\angle(\mathbf{w}^*, \mathbf{w}^{(T+1)})) = \frac{\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \rangle}{\|\mathbf{w}^*\| \|\mathbf{w}^{(T+1)}\|} \geq \frac{\sqrt{T}}{RB},$$

and therefore $T \leq (RB)^2$.

To show this, we first show that $\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \rangle \geq T$. Indeed, initially we have $\langle \mathbf{w}^*, \mathbf{w}^{(1)} \rangle = 0$; at iteration t, after updating $\mathbf{w}^{(t+1)}$ based on example i we have

$$\begin{aligned} \langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \rangle - \langle \mathbf{w}^*, \mathbf{w}^{(t)} \rangle &= \langle \mathbf{w}^*, \mathbf{w}^{(t)} + y_i \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{w}^{(t)} \rangle \\ &= \langle \mathbf{w}^*, y_i \mathbf{x}_i \rangle \\ &\geq 1. \end{aligned}$$

Therefore, after T iterations, we have $\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \rangle \geq T$.

Now we upper bound $\|\mathbf{w}^{(T+1)}\|$. At each iteration t, we have

$$\begin{aligned} \|\mathbf{w}^{(t+1)}\|^2 &= \|\mathbf{w}^{(t)} + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}^{(t)}\|^2 + \|y_i \mathbf{x}_i\|^2 + 2 y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \\ &\leq \|\mathbf{w}^{(t)}\|^2 + \|y_i \mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}^{(t)}\|^2 + R^2. \end{aligned}$$

Since $\|\mathbf{w}^{(1)}\|^2 = 0$, after T iterations, we have $\|\mathbf{w}^{(T+1)}\|^2 \leq TR^2$, i.e. $\|\mathbf{w}^{(T+1)}\| \leq \sqrt{T}R$.

Combining the above, we have

$$\begin{aligned} \frac{\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \rangle}{\|\mathbf{w}^*\| \|\mathbf{w}^{(T+1)}\|} &\geq \frac{T}{B\sqrt{T}R} \\ &= \frac{\sqrt{T}}{RB}, \end{aligned}$$

which concludes the proof of the theorem. □

2 Linear Regression

Consider a regression problem, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$. The class of linear regression predictor is defined as:

$$\mathcal{H}_{\text{LR}} = \mathcal{L}_d = \{\mathbf{x} \mapsto \mathbf{w}^T \mathbf{x} + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}.$$

To implement ERM for $\mathcal{H}_{\text{LR}}$ with respect to the square loss, one can use the least squares algorithm.

2.1 Least Squares

The ERM problem for homogenous $\mathcal{H}_{\text{LR}}$ is to find

$$\operatorname*{argmin}_{\mathbf{w}} L(h_{\mathbf{w}}, P_n) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2.$$

To solve the problem, we calculate the gradient $\nabla L(h_{\mathbf{w}}, P_n)$ of the objective function and compare it to zero, i.e.,

$$\nabla L(h_{\mathbf{w}}, P_n) = \frac{2}{n}\sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)\mathbf{x}_i = 0.$$

Let X be defined as

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}.$$

Then one can rewrite the above problem as

$$X^T X\mathbf{w} = X^T y^n,$$

leading to $\mathbf{w} = (X^T X)^{-1}X^T y^n$ in the case when $X^T X$ is invertible.

3 Logistic Regression

In logistic regression, we learn a family of function from $\mathcal{X} = \mathbb{R}^d$ to $\mathcal{Y} = [0, 1]$,

$$\mathcal{H}_{\text{sig}} = \phi_{\text{sig}} \circ \mathcal{L}_d = \{\mathbf{x} \mapsto \phi_{\text{sig}}(\mathbf{w}^T\mathbf{x}) : \mathbf{w} \in \mathbb{R}^d\},$$

where ϕ_{sig} is the sigmoid function, or logistic function in this context, defined as

$$\phi_{\text{sig}}(a) = \frac{1}{1 + e^{-a}}.$$

Logistic regression is used for classification task — the output of a logistic regression predictor $h_{\mathbf{w}}(\mathbf{x})$ can be interpreted as the probability of the label of $\mathbf{x}$ being 1. If $\mathbf{w}^T\mathbf{x}$ is very positive (or negative), then $\phi_{\text{sig}}(\mathbf{w}^T\mathbf{x})$ is close to 1 (or 0); if $\mathbf{w}^T\mathbf{x}$ is close to 0, then $\phi_{\text{sig}}(\mathbf{w}^T\mathbf{x}) \approx 0.5$. Therefore, a logistic regression predictor outputs a soft classification result, in contrast to the halfspace hypothesis which always output either $+1$ or -1.

The commonly used loss function for logistic regression is log loss, given by

$$\ell_{\log}(y, h_{\mathbf{w}}(\mathbf{x})) = \begin{cases} \log(1 + e^{-\mathbf{w}^T\mathbf{x}}) & y = +1 \\ \log(1 + e^{\mathbf{w}^T\mathbf{x}}) & y = -1 \end{cases},$$

which can be written more compactly as

$$\ell_{\log}(y, h_{\mathbf{w}}(\mathbf{x})) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

and also known as the logistic loss function. The ERM rule for logistic regression under the log loss is therefore to find

$$\operatorname*{argmin}_{\mathbf{w}} L(h_{\mathbf{w}}, P_n) = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}\log(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i}).$$

As we will see later in the course, this ERM problem can be solved efficiently using standard methods due to a nice property, namely the convexity, of the log loss function.