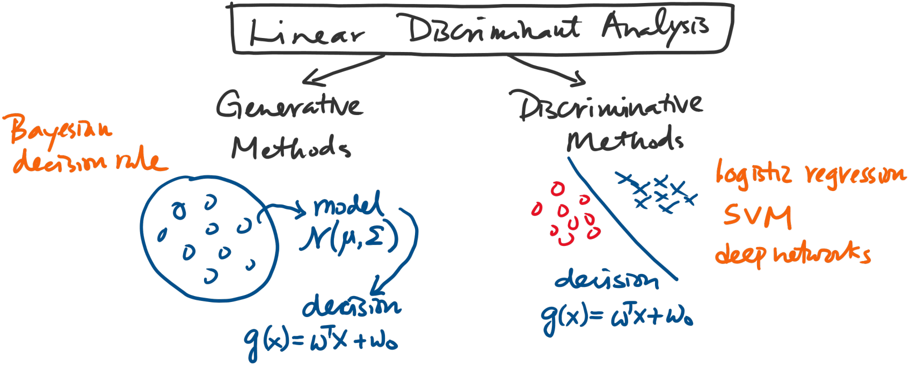
ECE595 / STAT598: Machine Learning I
Lecture 17.1: Perceptron 2 - Perceptron Algorithm

Spring 2020

Stanley Chan

School of Electrical and Computer Engineering
Purdue University

Overview



- In linear discriminant analysis (LDA), there are generally two types of approaches
- **Generative approach**: Estimate model, then define the classifier
- **Discriminative approach**: Directly define the classifier

Outline

Discriminative Approaches

- Lecture 16 Perceptron 1: Definition and Basic Concepts
- Lecture 17 Perceptron 2: Algorithm and Property
- Lecture 18 Multi-Layer Perceptron: Back Propagation

This lecture: Perceptron 2

- Perceptron Algorithm
  - Loss Function
  - Algorithm
- Optimality
  - Uniqueness
  - Batch and Online Mode
- Convergence
  - Main Results
  - Implication

Perceptron with Hard Loss

- Historically, we have perceptron algorithm way earlier than CVX.
- Before the age of CVX, people solve perceptron using gradient descent.
- Let us be explicit about which loss:

$$J_{\text{hard}}(\boldsymbol{\theta}) = \sum_{j=1}^{N} \max\left\{ -y_j h_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \right\}$$

$$J_{\text{soft}}(\boldsymbol{\theta}) = \sum_{j=1}^{N} \max\left\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \right\}$$

- **Goal**: To get a solution for $J_{\text{hard}}(\boldsymbol{\theta})$
- **Approach**: Gradient descent on $J_{\text{soft}}(\boldsymbol{\theta})$

Re-defining the Loss

- **Main idea**: Use the fact that

$$J_{\text{soft}}(\boldsymbol{\theta}) = \sum_{j=1}^{N} \max\left\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \right\}$$

is the same as this loss function

$$J(\boldsymbol{\theta}) = -\sum_{j \in \mathcal{M}(\boldsymbol{\theta})} y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j).$$

- $\mathcal{M}(\boldsymbol{\theta}) \subseteq \{1, \ldots, N\}$ is the set of misclassified samples.
- Run gradient descent on $J(\boldsymbol{\theta})$, but fixing $\mathcal{M}(\boldsymbol{\theta}) \leftarrow \mathcal{M}(\boldsymbol{\theta}^k)$ for iteration k.

Equivalent Perceptron Loss

- We want to show that the perceptron loss function is equivalent to

$$\underbrace{\sum_{j=1}^{N} \max\left\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \right\}}_{J_{\text{soft}}(\boldsymbol{\theta})} = \underbrace{- \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)}_{J(\boldsymbol{\theta})}$$

- If $\boldsymbol{x}_j$ is misclassified ($j \in \mathcal{M}(\boldsymbol{\theta})$)
  - then by definition of $\mathcal{M}(\boldsymbol{\theta})$ we have $\text{sign}\{g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\} \neq y_j$
  - So $-y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) > 0$
  - Therefore, $\max\{-y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j),\ 0\} = -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)$.
- If $\boldsymbol{x}_j$ is correctly classified ($j \notin \mathcal{M}(\boldsymbol{\theta})$)
  - then by definition of $\mathcal{M}(\boldsymbol{\theta})$ we have $\text{sign}\{g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\} = y_j$
  - So $-y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) < 0$
  - Therefore, $\max\{-y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j),\ 0\} = 0$.

Equivalent Perceptron Loss

- Therefore, we conclude that

$$\mathcal{M}(\boldsymbol{\theta}) = \{j \mid y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) < 0\}$$

Equivalent Perceptron Loss

- Therefore, we conclude that

$$\mathcal{M}(\boldsymbol{\theta}) = \{j \mid y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) < 0\}$$

- and

$$\begin{aligned} J_{\text{soft}}(\boldsymbol{\theta}) &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} \max\Big\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \Big\} + \sum_{j \notin \mathcal{M}(\boldsymbol{\theta})} \max\Big\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \Big\} \\ &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + \sum_{j \notin \mathcal{M}(\boldsymbol{\theta})} 0 \\ &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) = J(\boldsymbol{\theta}). \end{aligned}$$

Equivalent Perceptron Loss

- Therefore, we conclude that

$$\mathcal{M}(\boldsymbol{\theta}) = \{j \mid y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) < 0\}$$

- and

$$\begin{aligned} J_{\text{soft}}(\boldsymbol{\theta}) &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} \max\Big\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \Big\} + \sum_{j \notin \mathcal{M}(\boldsymbol{\theta})} \max\Big\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \Big\} \\ &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + \sum_{j \notin \mathcal{M}(\boldsymbol{\theta})} 0 \\ &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) = J(\boldsymbol{\theta}). \end{aligned}$$

- Minimizing $J(\boldsymbol{\theta})$ is less obvious because $\mathcal{M}(\boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$.

Equivalent Perceptron Loss

- Therefore, we conclude that

$$\mathcal{M}(\boldsymbol{\theta}) = \{j \mid y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) < 0\}$$

- and

$$\begin{aligned}
J_{\text{soft}}(\boldsymbol{\theta}) &= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} \max\Big\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \Big\} + \sum_{j \notin \mathcal{M}(\boldsymbol{\theta})} \max\Big\{ -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j), 0 \Big\} \\
&= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + \sum_{j \notin \mathcal{M}(\boldsymbol{\theta})} 0 \\
&= \sum_{j \in \mathcal{M}(\boldsymbol{\theta})} -y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) = J(\boldsymbol{\theta}).
\end{aligned}$$

- Minimizing $J(\boldsymbol{\theta})$ is less obvious because $\mathcal{M}(\boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$.
- But it gives a very easy algorithm.

Perceptron Algorithm

- The loss is

$$J(\boldsymbol{\theta}) = -\sum_{j \in \mathcal{M}(\boldsymbol{\theta})} y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j),$$

- At iteration k, fix $\mathcal{M}_k = \mathcal{M}(\boldsymbol{\theta}^{(k)})$
- Then, update via gradient descent

$$\begin{aligned}
\boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \alpha_k \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k)}) \\
&= \boldsymbol{\theta}^{(k)} - \alpha_k \sum_{j \in \mathcal{M}_k} \nabla_{\boldsymbol{\theta}} \Big(- y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \Big).
\end{aligned}$$

Perceptron Algorithm

- We can show that

$$\nabla_{\boldsymbol{\theta}}\Big(-y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\Big) = \begin{cases} -y_j \nabla_{\boldsymbol{\theta}}\Big(\boldsymbol{w}^T\boldsymbol{x}_j + w_0\Big) & , \\ 0, & , \end{cases}$$

$$= \begin{cases} = -y_j \begin{bmatrix} \boldsymbol{x}_j \\ 1 \end{bmatrix} & \text{if} \quad j \in \mathcal{M}_k, \\ 0, & \text{if} \quad j \notin \mathcal{M}_k. \end{cases}$$

- Thus, the update is

$$\begin{bmatrix} \boldsymbol{w}^{(k+1)} \\ w_0^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}^{(k)} \\ w_0^{(k)} \end{bmatrix} + \alpha_k \sum_{j \in \mathcal{M}_k} \begin{bmatrix} y_j \boldsymbol{x}_j \\ y_j \end{bmatrix}.$$

Perceptron Algorithm

- The algorithm is
- For $k = 1, 2, \ldots,$
- Update $\mathcal{M}_k = \{j \mid y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) < 0\}$ for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(k)}$.
- Gradient descent

$$\begin{bmatrix} \boldsymbol{w}^{(k+1)} \\ w_0^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}^{(k)} \\ w_0^{(k)} \end{bmatrix} + \alpha_k \sum_{j \in \mathcal{M}_k} \begin{bmatrix} y_j \boldsymbol{x}_j \\ y_j \end{bmatrix}.$$

- End For
- The set $\mathcal{M}_k$ can grow or can shrink from $\mathcal{M}_{k-1}$.
- If training samples are linearly separable, then converge. Zero training loss.
- If training samples are not linearly separable, then oscillates.

Updating One Sample