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

Spring 2020

Stanley Chan

School of Electrical and Computer Engineering
Purdue University
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
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
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:

\[
J_{\text{hard}}(\theta) = \sum_{j=1}^{N} \max \left\{ -y_j h_\theta(x_j), 0 \right\}
\]

\[
J_{\text{soft}}(\theta) = \sum_{j=1}^{N} \max \left\{ -y_j g_\theta(x_j), 0 \right\}
\]

**Goal:** To get a solution for \( J_{\text{hard}}(\theta) \)

**Approach:** Gradient descent on \( J_{\text{soft}}(\theta) \)
Re-defining the Loss

**Main idea**: Use the fact that

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

is the same as this loss function

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

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

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

\[
\sum_{j=1}^{N} \max \left\{ -y_j g_\theta(x_j), 0 \right\} = -\sum_{j \in M(\theta)} y_j g_\theta(x_j)
\]

\[
J_{\text{soft}}(\theta) \quad \quad \quad \quad \quad \quad J(\theta)
\]

If \( x_j \) is misclassified (\( j \in M(\theta) \))
\begin{itemize}
  \item then by definition of \( M(\theta) \) we have \( \text{sign} \left\{ g_\theta(x_j) \right\} \neq y_j \)
  \item So \( -y_j g_\theta(x_j) > 0 \)
  \item Therefore, \( \max \{ -y_j g_\theta(x_j), 0 \} = -y_j g_\theta(x_j) \).
\end{itemize}

If \( x_j \) is correctly classified (\( j \notin M(\theta) \))
\begin{itemize}
  \item then by definition of \( M(\theta) \) we have \( \text{sign} \left\{ g_\theta(x_j) \right\} = y_j \)
  \item So \( -y_j g_\theta(x_j) < 0 \)
  \item Therefore, \( \max \{ -y_j g_\theta(x_j), 0 \} = 0 \). 
\end{itemize}
Equivalent Perceptron Loss

Therefore, we conclude that

\[ \mathcal{M}(\theta) = \{ j \mid y_j g_{\theta}(x_j) < 0 \} \]
Equivalent Perceptron Loss

Therefore, we conclude that

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

and

$$J_{\text{soft}}(\theta) = \sum_{j \in \mathcal{M}(\theta)} \max \left\{ -y_j g_\theta(x_j), 0 \right\} + \sum_{j \notin \mathcal{M}(\theta)} \max \left\{ -y_j g_\theta(x_j), 0 \right\}$$

$$= \sum_{j \in \mathcal{M}(\theta)} -y_j g_\theta(x_j) + \sum_{j \notin \mathcal{M}(\theta)} 0$$

$$= \sum_{j \in \mathcal{M}(\theta)} -y_j g_\theta(x_j) = J(\theta).$$
Equivalent Perceptron Loss

Therefore, we conclude that

\[ M(\theta) = \{ j \mid y_j g_\theta(x_j) < 0 \} \]

and

\[ J_{\text{soft}}(\theta) = \sum_{j \in M(\theta)} \max \left\{ -y_j g_\theta(x_j), 0 \right\} + \sum_{j \notin M(\theta)} \max \left\{ -y_j g_\theta(x_j), 0 \right\} \]

\[ = \sum_{j \in M(\theta)} -y_j g_\theta(x_j) + \sum_{j \notin M(\theta)} 0 \]

\[ = \sum_{j \in M(\theta)} -y_j g_\theta(x_j) = J(\theta). \]

Minimizing \( J(\theta) \) is less obvious because \( M(\theta) \) depends on \( \theta \).
Therefore, we conclude that

\[ M(\theta) = \{ j \mid y_j g_\theta(x_j) < 0 \} \]

and

\[ J_{\text{soft}}(\theta) = \sum_{j \in M(\theta)} \max \left\{ -y_j g_\theta(x_j), 0 \right\} + \sum_{j \not\in M(\theta)} \max \left\{ -y_j g_\theta(x_j), 0 \right\} \]

\[ = \sum_{j \in M(\theta)} -y_j g_\theta(x_j) + \sum_{j \not\in M(\theta)} 0 \]

\[ = \sum_{j \in M(\theta)} -y_j g_\theta(x_j) = J(\theta). \]

Minimizing \( J(\theta) \) is less obvious because \( M(\theta) \) depends on \( \theta \).

But it gives a very easy algorithm.
The loss is
\[ J(\theta) = - \sum_{j \in M(\theta)} y_j g_\theta(x_j), \]

At iteration \( k \), fix \( M_k = M(\theta^{(k)}) \)

Then, update via gradient descent
\[
\theta^{(k+1)} = \theta^{(k)} - \alpha_k \nabla_\theta J(\theta^{(k)}) \\
= \theta^{(k)} - \alpha_k \sum_{j \in M_k} \nabla_\theta \left( - y_j g_\theta(x_j) \right).
\]
Perceptron Algorithm

- We can show that

\[ \nabla_{\theta} \left( - y_j g_{\theta}(x_j) \right) = \begin{cases} -y_j \nabla_{\theta} \left( \mathbf{w}^T x_j + w_0 \right) , \\ 0, \end{cases} \]

\[ = \begin{cases} = -y_j \begin{bmatrix} x_j \\ 1 \end{bmatrix} \quad \text{if } j \in \mathcal{M}_k, \\ 0, \quad \text{if } j \notin \mathcal{M}_k. \end{cases} \]

- Thus, the update is

\[ \begin{bmatrix} \mathbf{w}^{(k+1)} \\ w_0^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{w}^{(k)} \\ w_0^{(k)} \end{bmatrix} + \alpha_k \sum_{j \in \mathcal{M}_k} \begin{bmatrix} y_j x_j \\ y_j \end{bmatrix}. \]
Perceptron Algorithm

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

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

- 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

\[ y_j = +1 \]

\[ y_j = -1 \]