MLT Week-9

Sherry Thomas
21f3001449

Contents

Abstract

The week centers on a discussion of two important topics in machine learning, namely the Perceptron and Logistic Regression. It provides a comprehensive overview of these topics, highlighting their key concepts, underlying assumptions, and mathematical foundations.

Perceptron Learning Algorithm

The **Perceptron Learning Algorithm** is a type of supervised learning algorithm used for binary classification tasks. It involves iteratively adjusting the weights of a linear combination of input features until a decision boundary that separates the two classes is found. The algorithm is a type of discriminative classification as it directly models the boundary between classes rather than modeling the underlying probability distribution of each class.

Let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the dataset, where $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$.

Assumptions made in the algorithm:

1. $P(y = 1|x) = 1$ if $w^T x \geq 0$, 0 otherwise.
2. **Linear Separability Assumption**: The Linear Separability Assumption is the assumption made in some machine learning algorithms, including the Perceptron Learning Algorithm, that the classes to be classified can be separated by a linear decision boundary. This means that there

exists a hyperplane in the feature space that can separate the data points of the two classes.

The objective function is given by,

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{n} \mathbb{1}\left(h(x_i) \neq y_i\right)$$

In general, this is an NP-Hard Problem even if $\mathcal{H}$ only accounts for the Linear Hypotheses.

Under the Linear Separability Assumption, let $\exists w \in \mathbb{R}^d$ s.t. $\text{sign}(w^T x_i) = y_i$ $\forall i \in [n]$.

We solve this problem of convergence using an iterative algorithm. The algorithm is as follows:

- Assign $w^0 = 0 \in \mathbb{R}^d$
- Until Convergence:
  - Pick (x_i, y_i) pair from the dataset
  - if $\text{sign}(w^T x_i) == y_i$
    - do nothing
  - else
    - $w^{(t+1)} = w^t + x_i y_i$ $\leftarrow$ update rule.
  - end

Analysis of the Update Rule

For a training example (x, y), where x is the input and y is the correct output (either 1 or -1), the perceptron algorithm updates its weight vector w as follows:

- If the prediction of the perceptron on x is correct (i.e., $\text{sign}(w^T x_i) == y_i$), then no update is performed.
- If the prediction of the perceptron on x is incorrect (i.e., $\text{sign}(w^T x_i) \neq y_i$), then the weights are updated by adding the product of the input vector and the correct output to the current weight vector: $w^{(t+1)} = w^t + x_i y_i$.
- The update happens specifically in response to the current datapoint. Therefore, as other points are not considered in that specific update, datapoints which were classified correctly previously may not be classified likewise in the future iterations.

This update rule effectively moves the decision boundary in the direction of the correct classification for the misclassified example. It is guaranteed to converge to a linearly separable solution if the data is linearly separable. However, if the data is not linearly separable, the perceptron algorithm may not converge to a solution.

Further Assumptions

We make three further assumptions:

1. **Linear Separability with γ-Margin**: A dataset $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ is linearly separable with γ-margin if $\exists w^* \in \mathbb{R}^d$ s.t. $(w^{*T} x_i) y_i \geq \gamma \ \forall i$ for some $\gamma > 0$.

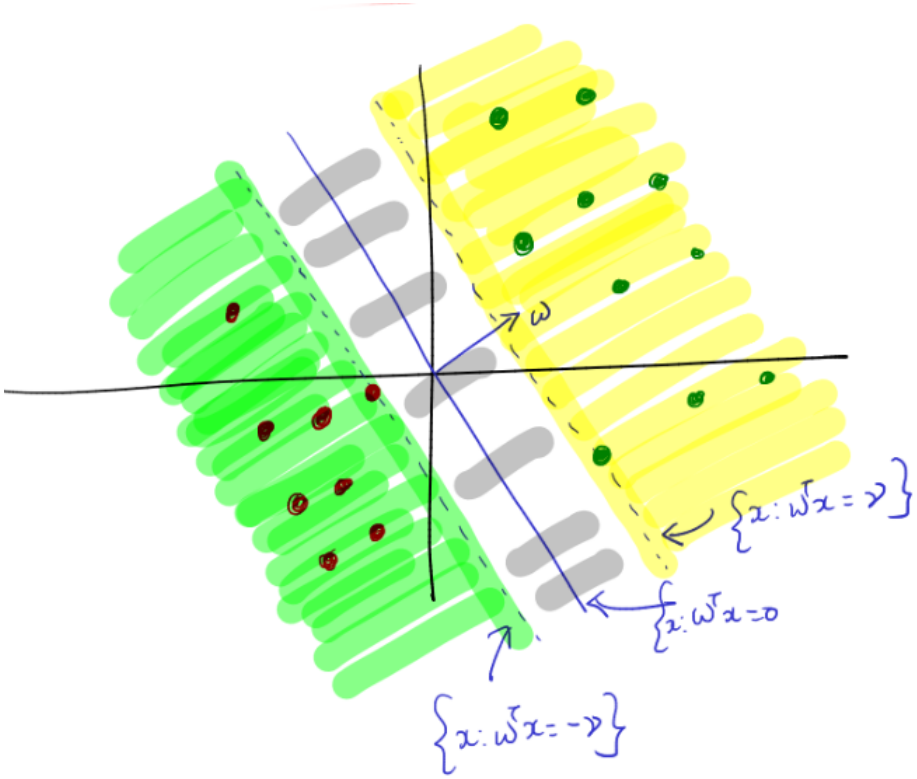

Figure 1: Linear Separability with γ-Margin

2. **Radius Assumption**: Let some $R > 0 \in \mathbb{R}$, $\forall i \in D \ ||x_i|| \leq R$. In short, let R be the length of the datapoint furthest from the center.
3. **Normal Length for w^***: Let w^* be of unit length.

Proof of Convergence of Perceptron Algorithm

Let l denote the current mistake number. Using everything upto now, we can see,

$$\begin{aligned}
w^{l+1} &= w^l + xy \\
||w^{l+1}||^2 &= ||w^l + xy||^2 \\
&= (w^l + xy)^T(w^l + xy) \\
&= ||w^l||^2 + 2(w^{lT}x)y + ||x||^2y^2 \\
\therefore ||w^{l+1}||^2 &\leq ||w^l||^2 + R^2 \\
&\leq (||w^{l-1}||^2 + R^2) + R^2 \\
&\leq ||w^0||^2 + lR^2 \\
\therefore ||w^{l+1}||^2 &\leq lR^2 \qquad \text{... [1]}
\end{aligned}$$

We can also see,

$$\begin{aligned}
w^{l+1} &= w^l + xy \\
(w^{l+1})^T w^* &= (w^l + xy)^T w^* \\
&= w^{lT}w^* + (w^{*T}x)y \\
\therefore (w^{l+1})^T w^* &\geq w^{lT}w^* + \gamma \\
&\geq (w^{l-1T}w^* + \gamma) + \gamma \\
&\geq w^{0T}w^* + l\gamma \\
\therefore (w^{l+1})^T w^* &\geq l\gamma \\
((w^{l+1})^T w^*)^2 &\geq l^2\gamma^2 \\
||w^{l+1}||^2||w^*||^2 &\geq l^2\gamma^2 \quad \text{... Using Cauchy-Schwartz} \\
\therefore ||w^{l+1}||^2 &\geq l^2\gamma^2 \qquad \text{... [2]}
\end{aligned}$$

Therefore, from [1] and [2], we get,

$$\begin{aligned}
l^2\gamma^2 \leq ||w^{l+1}||^2 &\leq lR^2 \\
l^2\gamma^2 &\leq lR^2 \\
\therefore l &\leq \frac{R^2}{\gamma^2}
\end{aligned}$$

Hence, from the above equation, we can conclude that for a dataset that follows Linear Separability with γ-Margin, and has a finite Radius R, the perceptron algorithm has an upper bound for the number of mistakes.

Therefore, perceptron converges.

Logistic Regression

Sigmoid Function

Until now, we used the sign function to get the class for the output. But can we also provide the probabilities for these outputs?

Let $z = w^T x$ and $z \in \mathbb{R}$. How can we map $[-\infty, \infty] \to [0, 1]$? For this, we use the **Sigmoid Function**. It is given by,

$$g(z) = \frac{1}{1 + e^{-z}}$$

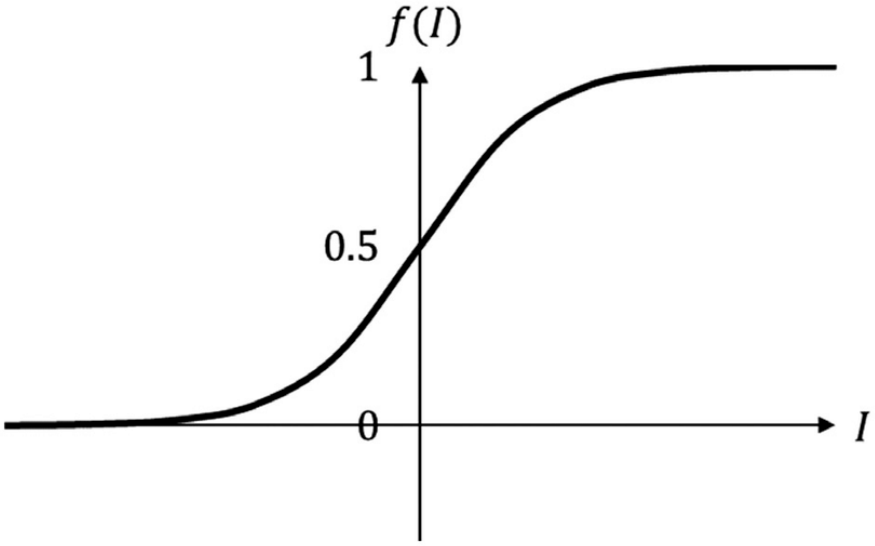

Figure 2: Sigmoid Function

The sigmoid function is often used in machine learning as an activation function for neural networks. It has a characteristic S-shaped curve, which makes it useful for modeling processes that have a threshold or saturation point, such as logistic growth or binary classification problems.

When the input value is large and positive, the sigmoid function output approaches 1, and when the input value is large and negative, the sigmoid function output approaches 0. When the input value is 0, the sigmoid function output is exactly 0.5.

The term "sigmoid" comes from the Greek word "sigmoides," which means "shaped like the letter sigma" (Σ). The letter sigma has a similar shape to the sigmoid function's characteristic S-shaped curve, which is likely the reason for the function's name.

Logistic Regression

Logistic regression is a statistical method used to analyze and model the relationship between a binary (two-valued) dependent variable and one or more independent variables, which can be either continuous or categorical. The goal of logistic regression is to estimate the probability that the dependent variable is one of the two possible values, given the values of the independent variables.

In logistic regression, the dependent variable is modeled as a function of the independent variables using a logistic(sigmoid) function, which produces an S-shaped curve that ranges between 0 and 1. The logistic function transforms the output of a linear combination of the independent variables into a probability estimate, which can then be used to classify new observations.

Let $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be the dataset, where $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$.

We know,

$$P(y = 1|x) = g(w^T x_i) = \frac{1}{1 + e^{-w^T x}}$$

Using Maximum Likelihood, we get

$$\begin{aligned}
\mathcal{L}(w; \text{Data}) &= \prod_{i=1}^{n} (g(w^T x_i))^{y_i} (1 - g(w^T x_i))^{1-y_i} \\
\log(\mathcal{L}(w; \text{Data})) &= \sum_{i=1}^{n} y_i \log(g(w^T x_i)) + (1 - y_i) \log(1 - g(w^T x_i)) \\
&= \sum_{i=1}^{n} y_i \log\left(\frac{1}{1 + e^{-w^T x_i}}\right) + (1 - y_i) \log\left(\frac{e^{-w^T x_i}}{1 + e^{-w^T x_i}}\right) \\
&= \sum_{i=1}^{n} \left[(1 - y_i)(-w^T x_i) - \log(1 + e^{-w^T x_i})\right]
\end{aligned}$$

Therefore, our objective, which is maximizing the log-likelihood function, is given by,

$$\max_w \sum_{i=1}^{n} \left[(1 - y_i)(-w^T x_i) - \log(1 + e^{-w^T x_i})\right]$$

But, there is no closed form solution for this. And hence, we use gradient descent for convergence.

The gradient is given by,

$$\begin{aligned}
\nabla \log(\mathcal{L}(w; \text{Data})) &= \sum_{i=1}^{n} \left[(1 - y_i)(-x_i) - \left(\frac{e^{-w^T x_i}}{1 + e^{-w^T x_i}}\right)(-x_i)\right] \\
&= \sum_{i=1}^{n} \left[-x_i + x_i y_i + x_i \left(\frac{e^{-w^T x_i}}{1 + e^{-w^T x_i}}\right)\right] \\
&= \sum_{i=1}^{n} \left[x_i y_i - x_i \left(\frac{1}{1 + e^{-w^T x_i}}\right)\right] \\
\nabla \log(\mathcal{L}(w; \text{Data})) &= \sum_{i=1}^{n} \left[x_i \left(y_i - \frac{1}{1 + e^{-w^T x_i}}\right)\right]
\end{aligned}$$

Using the Gradient Descent update rule, we get,

$$\begin{aligned} w_{t+1} &= w_t + \eta_t \nabla \log(\mathcal{L}(w; \text{Data})) \\ &= w_t + \eta_t \left(\sum_{i=1}^{n} x_i \left(y_i - \frac{1}{1 + e^{-w^T x_i}} \right) \right) \end{aligned}$$

Kernel and Regularized Versions

We can argue that $w^* = \sum_{i=1}^{n} \alpha_i x_i$, and therefore, can be Kernelized. For further details, refer to this link.

The regularized version is given by,

$$\min_w \sum_{i=1}^{n} \left[\log(1 + e^{-w^T x_i}) + w^T x_i (1 - y_i) \right] + \frac{\lambda}{2} ||w||^2$$

where $\frac{\lambda}{2}||w||^2$ is the regualizer and λ is found using cross-validation.

Credits

Professor Arun Rajkumar: The content as well as the notations are from his slides and lecture.