Perceptron

Hanxiao Du

Contents

1 Linear Classifiers

For now, we only consider thresholded linear mappings from inputs to (binary) outputs (for binary classification):

$$f(x;\theta) = sign(\theta_1 x_1 + \cdots + \theta_d x_d) = sign(\theta^T x) \tag{1}$$

where $x = [x_1, \cdots, x_d]^T$ is a matrix (column vector) of the input (i.e. input with d attributes, x_i for all $i \in [d]$ denotes an attribute of the input), $\theta = [\theta_1, \cdots, \theta_d]^T$ is matrix (column vector) of the parameters (or coefficients, θ_j for all $j \in [d]$ denotes the coefficient of the j-th attribute), also

$$sign(x) = \begin{cases} 1 \text{ if } x \geq 0, \\ -1 \text{ if } x < 0. \end{cases} \tag{2}$$

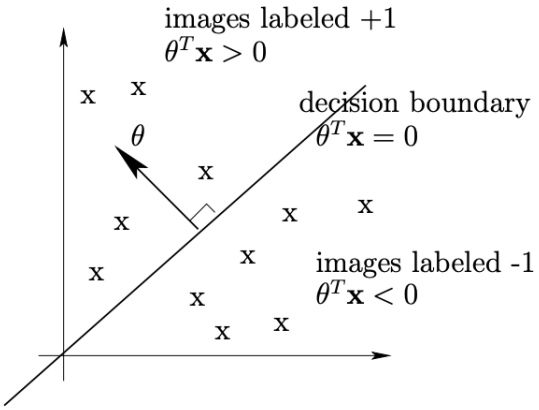

Figure 1: A linear classifier through origin.

- The functions in the class are parameterized by θ.
- $f(x;\theta)$ defines a plane in d-dimensions.

- The decision boundary is $\theta^T x = 0$. Thus, the parameter vector θ is orthogonal to the decision boundary.
- The decision boundary goes through the origin since $x = 0$ satisfies $\theta^T x = 0$.

2 Perceptron

After choosing a class of functions, we still have to find a specific function in this class that performs well on the training set. This is usually referred to as the estimation problem.
We would like to find a linear classifier that makes the fewest mistakes on the training set, i.e. find θ that minimizes the training error:

$$\hat{E}[\theta] = \frac{1}{n}\sum_{i=1}^{n}\left[\mathbb{I}(y_i = f(x_i;\theta))\right] \tag{3}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[1 - \delta(y_i, f(x_i;\theta))\right] \tag{4}$$

$$= \frac{1}{n}\sum_{i=1}^{n} Loss(y_i, f(x_i;\theta)), \tag{5}$$

where $\delta(y, y') = 1$ if $y = y'$ and 0 otherwise. The training error merely counts the average occurrence of that the labels y_i is different from the model predicted value $f(x_i;\theta)$ $\forall i \in [n]$, which can also be expressed in terms of a loss function $Loss(y_i, f(x_i;\theta))$.
Loss function is useful when errors of a particular kind cost more than other. For the case above, we are using zero-one loss for simplicity.
In this case, the simplest algorithm to minimize the training error is the perceptron update rule.

$$\theta' \leftarrow \theta + y_i x_i \text{ if } y_i \neq f(x_i;\theta) \tag{6}$$

This update rule only changes θ if the model predicted incorrectly.

- the model made a mistake on x_i: $y_i \neq f(x_i;\theta) \iff y_i\theta^T x_i < 0$

Suppose we make a mistake on x_i (i.e. $y_i\theta^T x_i < 0$), then after the update $\theta' \leftarrow \theta + y_i x_i$, consider classifying the same image x_i:
Thus, $y_i\theta^T x_i$ increased after the update (i.e. it becomes more positive/correct).

$$y_i\theta'^T x_i = y_i(\theta + y_i x_i)^T x_i \tag{7}$$

$$= y_i\theta^T x_i + y_i^2 x_i^T x_i \tag{8}$$

$$= y_i\theta^T x_i + y_i^2 ||x_i||_2^2 \tag{9}$$

2.1 Analysis of The Perceptron Algorithm

The perceptron algorithm stops to update the parameters only when all the training images are classified correctly. Thus, if it is possible to classify all training data with a linear classifier, the perceptron algorithm will find such a classifier eventually in a finite number of updates.