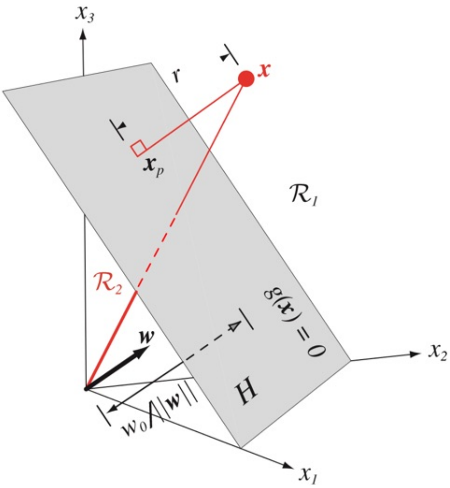
Lecture 2
Perceptron : Single Layer Neural Network

Naresh Manwani

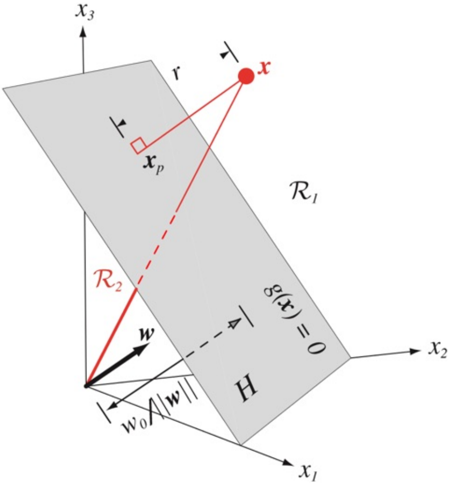
Deep Learning: Theory and Practices
CS7.601 (Spring 2021)
IIIT-Hyderabad

July 31, 2023

Hyperplane

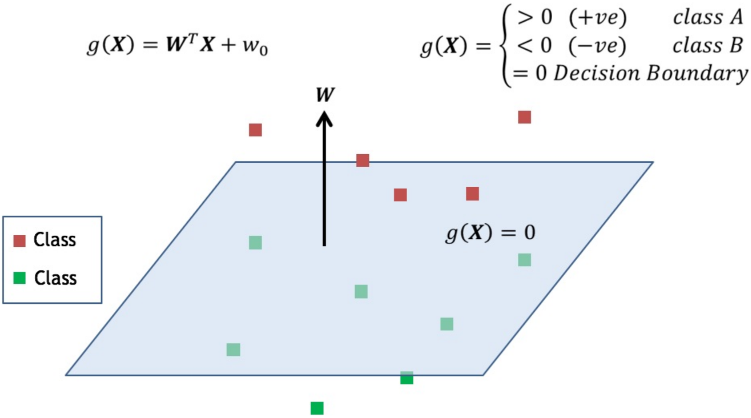


- $\mathbf{w}$ is normal to the hyperplane.
- $\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$ is the distance of $\mathbf{x}$ from the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$.



- $\mathbf{w}$ is normal to the hyperplane.
- $\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$ is the distance of $\mathbf{x}$ from the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$. (**HW for you**)

Linear Discriminant Function



Supervised Learning Problem

- Let $\mathcal{X} \subset \mathbb{R}^d$ be the feature space.
- We can categorise the task in hand based on the label space, denoted by $\mathcal{Y}$ as:

$$\mathcal{Y} = \begin{cases} \{+1, -1\} & \text{Binary Classification} \\ \{1, 2, 3, .., C\} & \text{Multi-class Classification} \\ \mathbb{R} & \text{Regression} \end{cases}$$

- We assume the training data is $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$ where each $(\mathbf{x}_i, y_i)$ is i.i.d. drawn from an unknown distribution $P(\mathbf{x}, y)$ on $\mathcal{X} \times \mathcal{Y}$.
- We're looking for a hypothesis, $f : \mathcal{X} \to \mathcal{Y}$, which is obtained by training on set S .

Loss Function

- The hypothesis $f(x)$ predicts the label $(\hat{y})$.
- To measure the discrepancy between the prediction $(\hat{y})$ and the corresponding actual label y, we use a loss function.

$$L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+ \qquad \text{If considering } sign(f(\mathbf{x}))$$

$$L : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+ \qquad \text{Considering simply } f(\mathbf{x})$$

Risk Function

- **Risk**: Risk is defined as the expectation of the loss function.
- This expectation is with respect to the joint distribution function $P(\mathbf{x}, y)$

$$R_L(f) = \mathbb{E}[L(f(\mathbf{x}), y)] \qquad \text{Expected Risk}$$

- In practical training situations, we don't have access to the joint distribution function, $P(\mathbf{x}, y)$.
- We are usually provided with a finite set of training examples, $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$.
- In such situation, we find the *empirical risk* as follows:

$$\hat{R}_L(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \qquad \text{Empirical Risk}$$

Empirical Risk Minimization

- Minimizing the empirical risk of a learning algorithm can be achieved in two ways.
- One is through **batch learning** where the risk over all the training examples is minimized at once. SVM, logistic regression algorithms minimize the risk through batch learning.
- The second way, is through **online learning** where, the risk is minimized in a sequential order and is fast and scalable. For example: **Perceptron**.

Generic Online Learning Framework

In an online learning algorithm, the learning takes place in a sequence of trials.

- Observes the example instance $\mathbf{x}_t$.
- Uses current hypothesis f_t to predict the output $\hat{y}_t = f_t(\mathbf{x}_t)$.
- Observe actual output y_t.
- Incurs a loss of $L(\hat{y}_t, y_t)$.
- Updates the current hypothesis.
- There is an objective associated with the training, which is to minimize the total loss, in a sequence of T trials.

$$\text{Total Loss} = \sum_{t=1}^{T} L(\hat{y}_t, y_t)$$

Stochastic Gradient Descent

- In the above framework, there are some specifics left out.
- What choice of loss function do we make?
- How do we update the hypothesis?
- One widely used method to update the hypothesis is Stochastic Gradient Descent (SGD).

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} L(f(\mathbf{x}_t), y_t)$$

- One model of the above framework is the linear Perceptron. Input to Perceptron is $\mathbf{x}$, hypothesis is given by:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

Perceptron: Single Layer Neural Network

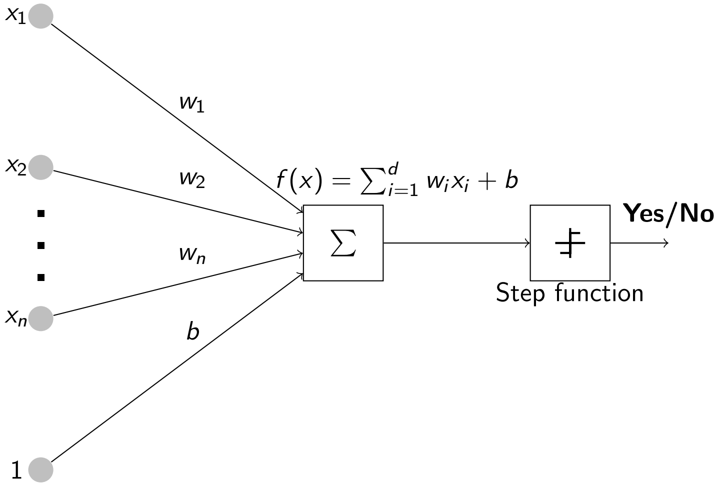


Figure: The Perceptron model

f predicts a real value, and $\mathrm{sign}(f(\mathbf{x}))$ gives the label.

Perceptron

- It used for classification purpose.
- Given an example $\mathbf{x} = [x_1 \;\; x_2 \;\; \ldots \;\; x_d]^T$, it predicts the output label

$$\hat{y} = \text{sign}(f(\mathbf{x})) = \text{sign}(\sum_{i=1}^{d} w_i x_i + b)$$

- It then compares the predicted label $\hat{y}$ with the actual label y.
- Note that $\hat{y}, y \in \{+1, -1\}$.

Perceptron Objective Function

Loss Function

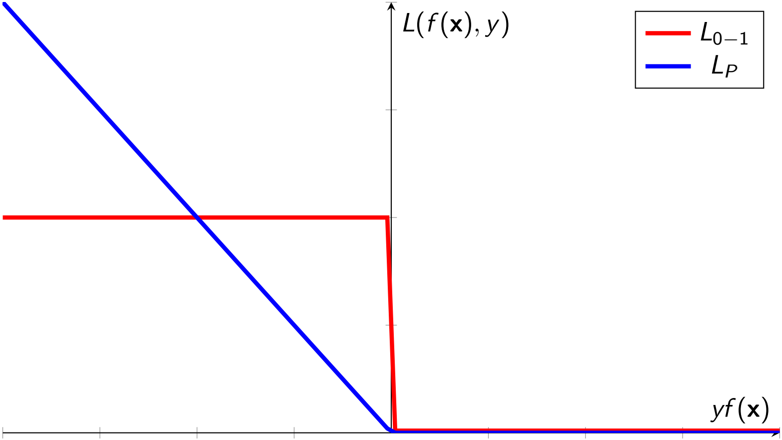
$L_{\text{Perceptron}}(f(\mathbf{x}), y) = max(0, -yf(\mathbf{x}))$

here $f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b$

- if $yf(\mathbf{x}) \geq 0$ (correct classification), then the loss is 0.
- if $yf(\mathbf{x}) < 0$ (misclassification), then the loss is $-yf(\mathbf{x})$
- Thus, the loss increases lineraly with the margin of misclassification[1]

[1] $yf(\mathbf{x})$ is denoted as margin

Perceptron Loss



How to update the hypothesis?

Stochastic Gradient Descent

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_f L(f(\mathbf{x}^t), y^t)$$

where ∇ denotes the gradient.

- Let $f(\mathbf{x}^t) = \mathbf{w}.\mathbf{x}^t + b$
- then

$$\begin{aligned}
\nabla_{\mathbf{w}} L(f(\mathbf{x}^t), y^t) &= \nabla_{\mathbf{w}} L(\mathbf{w}.\mathbf{x}^t + b, y^t) \\
&= \begin{pmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_d} \end{pmatrix}
\end{aligned}$$

Stochastic gradient descent on $L_{\text{Perceptron}}$

$$
\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}.\mathbf{x}^t + b, y^t) &= \begin{cases} \nabla_{\mathbf{w}} \left(-y^t(\mathbf{w}.\mathbf{x}^t + b)\right) & y_t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0 \\ 0 & y_t(\mathbf{w}^t.\mathbf{x}^t + b^t) > 0 \end{cases} \\
&= \begin{cases} -y^t\mathbf{x}^t & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0 \\ 0 & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) > 0 \end{cases} \\
\nabla_b L(\mathbf{w}.\mathbf{x}^t + b, y^t) &= \begin{cases} \nabla_b \left(-y^t(\mathbf{w}.\mathbf{x}^t + b)\right) & y_t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0 \\ 0 & y_t(\mathbf{w}^t.\mathbf{x}^t + b^t) > 0 \end{cases} \\
&= \begin{cases} -y^t & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0 \\ 0 & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) > 0 \end{cases}
\end{aligned}
$$

Update rule for Perceptron

$$\mathbf{w}^{t+1} = \begin{cases} \mathbf{w}^t + \eta y^t \mathbf{x}^t & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0 \\ \mathbf{w}^t & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) > 0 \end{cases}$$

$$b^{t+1} = \begin{cases} b^t + \eta y^t & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0 \\ b^t & y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) > 0 \end{cases}$$

Update Rule for Perceptron

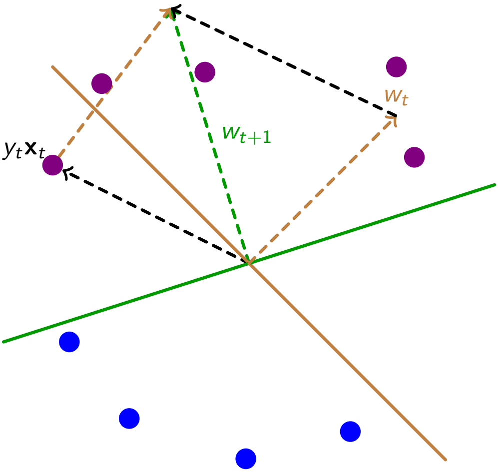


Figure: The weight update in Perceptron during misclassification

Perceptron Algorithm

Input: $S = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^T, y^T)\}$
Output: $(\mathbf{w}^*, b^*)$

Initialize: $\mathbf{w}^1 = \mathbf{0}, \; b^1 = 0$
For($t = 1$ to T)

Randomly draw an example $(\mathbf{x}^t, y^t)$ from S
If($y^t(\mathbf{w}^t.\mathbf{x}^t + b^t) \leq 0$)

$$\mathbf{w}^{t+1} = \mathbf{w}^t + y^t\mathbf{x}^t$$
$$b^{t+1} = b^t + y^t$$

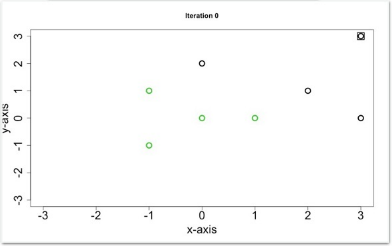
Else

$$\mathbf{w}^{t+1} = \mathbf{w}^t$$
$$b^{t+1} = b^t$$

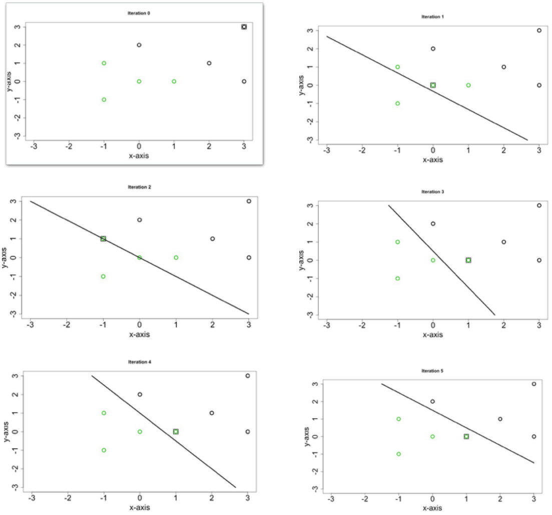
$$(\mathbf{w}^*, b^*) = (\mathbf{w}^{T+1}, b^{T+1})$$

Example: Perceptron on 2-D Data

$$X = \begin{pmatrix} 3 & 3 & 1 \\ 3 & 0 & 1 \\ 2 & 1 & 1 \\ 0 & 2 & 1 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \\ -1 & -1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \quad Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$



Example: Perceptron on 2-D Data



Perceptron Convergence for Linearly Separable Case

Theorem

Finite mistake bound: *Let* $S = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}_T, y_T)\}$*, where* T *is number of samples drawn. Let* $\exists\ \mathbf{u} \in \mathbb{R}^d$*, s.t* $\|\mathbf{u}\|_2 = 1$ *and* $y_t \mathbf{u}^T \mathbf{x}_t \geq \gamma$*, where* $\gamma \geq 0, \forall t = 1..T$*. Let* $R_2 = \max_t \|\mathbf{x}_t\|_2$*. The Perceptron algorithm converges in at most* $\left(\frac{R_2}{\gamma}\right)^2$ *iterations.*

Batch Perceptron

- In batch Perceptron, all the examples in the training set are used to update the hypothesis
- Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be the training set, where $(\mathbf{x}_i \times y_i) \in \mathbb{R}^d \times \{-1, +1\},\ \forall i$
- Batch Perceptron minimizes the following objective function

$$\min_{\mathbf{w}, b} \sum_{i=1}^{m} \max\left(0, -y_i(\mathbf{w}^T \mathbf{x}_i + b)\right)$$

Batch Perceptron Algorithm

Algorithm

Initialize $\mathbf{w}$, b, $\eta(.)$, θ, $k \leftarrow 0$

do $k \leftarrow k + 1$

$\mathbf{w} \leftarrow \mathbf{w} + \eta(k) \sum_{i \in \mathcal{M}_k} y_i \mathbf{x}_i$

$b \leftarrow b + \eta(k) \sum_{i \in \mathcal{M}_k} y_i$

until $\|\eta(k) \sum_{i \in \mathcal{M}_k} [y_i \mathbf{x}_i \quad y_i]^T \| < \theta$

return $\mathbf{w}$, b

where $\mathcal{M}_k$ is the set of examples which are misclassified in round k.

- Choose $\eta(k) > 0$ such that $\sum_k \eta_k = \infty$ and $\sum_k \eta_k^2 < \infty$