Introduction to Perceptron

using python.

Iván Andrés Trujilllo Abella

Facultad de Ingenieria
Pontificia Universidad Javeriana

trujilloiv@javeriana.edu.co
addajaveriana

Background classification problem

Fisher(1936) proposes the linear Discriminant, the problem consist in predict a binary outcome according to a set of features, for instance find the variables that could predict the bankruptcy.
The main objective is construct a $z = \mathrm{w}^t\mathrm{x}$ score whose indicate the probability of belonging to a class.

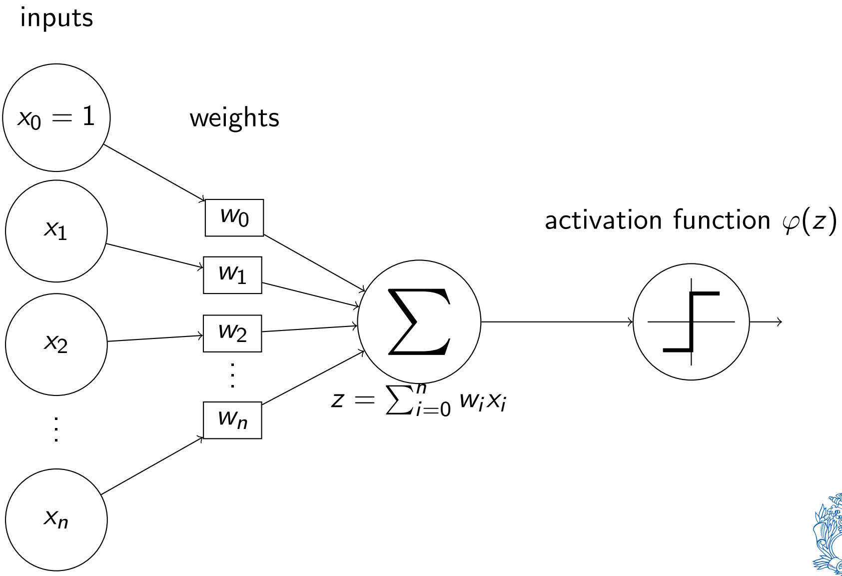
Binary classification problem

$y_0, y_1 \in Y$ and $y_0 \cup y_1 = Y$ and $y_0 \cap y_1 = \emptyset$ both class are exhaustive and are defined without ambiguity. A vector of weights and a vector of features for a

$$w^t x = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \sum_{i=1}^{n} w_i x_i = z_i$$

Perceptron

Rossenblant(1958) We can define a vector input x and a vector of weights w and a activation function φ that take as input the inner product of both vectors defined previously $\varphi(\mathrm{w}^t\mathrm{x})$.


inputs
$x_0 = 1$
x_1
x_2
x_n
weights
w_0
w_1
w_2
w_n
$z = \sum_{i=0}^{n} w_i x_i$
activation function $\varphi(z)$

Activation function

$\varphi()$ could be defined as the sigmoid function.

$$
\begin{aligned}
p(y=1) &= \varphi(z) \\
p(y=0) &= 1-\varphi(z)
\end{aligned} \tag{1}
$$

Percepton works with a step function:

$$
\varphi(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}
$$

Returning a binary outcome.

What set of w values we must choose

Cost function

The cost function could be defined in a soft or a hard way.

$$J(w)_{hard} = \sum_{i=1}^{n} = \max(-y_i \hat{y}, 0) \tag{2}$$

J only count the number of mismatches. However this function not is differentiable.

$$J(w)_{soft} = \sum_{i=1}^{n} = \max(-y_i z_i, 0) \tag{3}$$

if $y_i z_i < 0$ then lost function > 0
if $y_i z_i > 0$ the lost function $= 0$.

Insights

The update of weights is according to the data bias or mistakes, however when the model match to the class then

$$\Delta w_i = (y_i - \hat{y}_i) = 0 \tag{4}$$

where y_i is the real observed data, and $\hat{y}_i$ is the predicted class.
when the $y_i = -1$ and $\hat{y} = 1$ then $\Delta w = -2$, in otherwise $y_i = 1$ and $\hat{y}_i = -1$ then $\Delta w = 2$. in summary:

$$\Delta w_i = \begin{cases} 0 & y_i = \hat{y}_i \\ -2 & y_i < \hat{y}_i \\ 2 & y_i > \hat{y}_i \end{cases}$$

insights

Then when there are mistakes

$$\varphi(w_{i+1}^t \mathrm{x_i}) = \varphi((w_i + \Delta w_i)^t \mathrm{x_i}) = y_i \tag{5}$$

This mean that weights for the vector of features of the sample i are update to predict the correct class.

$$w_{i+1} = w_i + \eta \Delta w_i \mathrm{x_i} \tag{6}$$

Perceptron algorithm

```
initialize w:
for each x in sample :
estimate y(x)
w = w + update(w)
```

LAB

Perceptron implementation from scratch

Perceptron from scratch(click here)

sklearn

It is open source library, integrated with scipy and numpy. It is one of the most popular machine learning library on Github.

- Classification (Neural networks Support Vector Machine)
- Decision trees
- Cluster
- Regression

sklearn

```
from skelearn.linear_model import Perceptron
model = Perceptron(penalty=None , max_iter=1000, eta0=0.4,
    random_state=1)
model.fit(X,y)
model.score(X,y) # Print the number of matches
from sklearn.metrics import confusion_matrix
confusion_matrix(y, model.predict(X))
print(model.coef_)
```

Insights more deeply about perceptron

The function *sing* is defined as $\mathbb{R} :\longrightarrow \{-1, 1, 0\}$