Perceptrons

Prof. Dan A. Simovici

UMB

What is a perceptron

Perceptrons are classifiers that use hyperplanes to separate sets of vectors in $\mathbb{R}^n$. Data analyzed consists of finite sequences of pairs of the form $(\boldsymbol{x}, y)$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $y \in \{-1, 1\}$.
We discuss an algorithm that begins with a linearly separable sample S and produces a separating hyperplane. The algorithm is due to F. Rosenblatt who proposed a mathematical device known as a *perceptron* that is described by a hyperplane of the form $\boldsymbol{w}'\boldsymbol{x} + b = 0$, where $\boldsymbol{w} \in \mathbb{R}^n$ is the *weight vector* and $b \in \mathbb{R}$ is the *bias*.

Let R be the minimum radius of a closed ball centered in $\mathbf{0}$,

$$R = \max\{\| \boldsymbol{x}_i \| \mid 1 \leqslant i \leqslant \ell\}.$$

- If $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$ H is the target hyperplane $\boldsymbol{w}'\boldsymbol{x} + b = 0$, where $\| \boldsymbol{w} \| = 1$, the functional margin of $(\boldsymbol{x}_i, y_i)$ is $\gamma_i = y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)$.
- The overall functional margin is $\gamma = \min\{\gamma_i \mid 1 \leqslant i \leqslant m\}$, hence $y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geqslant \gamma$.
- If y_i and $\boldsymbol{w}'\boldsymbol{x}_i + b$ have the same sign, then $(\boldsymbol{x}_i, y_i)$ is classified correctly; otherwise, it is incorrectly classified and we say that a mistake occurred.

A Learning Algorithm for the Perceptron

Input: labelled training sequence S and learning rate η
Output: weight vector $\boldsymbol{w}$ and parameter b defining classifier

$\boldsymbol{w} = \mathbf{0}$, $b_0 = 0$, $k = 0$;$R = \max\{\| \boldsymbol{x}_i \| \mid 1 \leqslant i \leqslant m\}$;mistake $=$ TRUE;
while (mistake)
    mistake $=$ FALSE;
    for $(i = 1 \text{ to } m)$
        if $(y_i(\boldsymbol{w}'_k \boldsymbol{x}_i + b_k) \leqslant 0)$
            $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta y_i \boldsymbol{x}_i$;
            $b_{k+1} = b_k + \eta y_i R^2$;
            $k = k + 1$;
            mistake $=$ TRUE;
        endif
    endfor
    endwhile
return $k, (\boldsymbol{w}_k, b_k)$ where k is the number of mistakes

Theorem

Let $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))$ *be a non-trivial training sequence that is linearly separable, and let* $R = \max\{\| \boldsymbol{x}_i \| \mid 1 \leqslant i \leqslant m\}$*.*
Suppose there exists an optimal weight vector $\boldsymbol{w}_{opt}$ *and an optimal bias* b_{opt} *such that*

$$\| \boldsymbol{w}_{opt} \| = 1 \text{ and } y_i(\boldsymbol{w}'_{opt}\boldsymbol{x}_i + b_{opt}) \geqslant \gamma,$$

for $1 \leqslant i \leqslant m$*. The, the number of mistakes made by the algorithm is at most*

$$\left(\frac{2R}{\gamma}\right)^2.$$

Proof

Let k be the update counter and let $\hat{\mathbf{w}}$ and $\hat{\mathbf{x}}_i$ be the augmented vectors:

$$\hat{\mathbf{w}} = \begin{pmatrix} \boldsymbol{w} \\ \frac{b}{R} \end{pmatrix} \text{ and } \hat{\mathbf{x}}_i = \begin{pmatrix} \boldsymbol{x}_i \\ R \end{pmatrix}$$

for $1 \leqslant i \leqslant m$.

The algorithm begins with an augmented vector $\hat{\mathbf{w}}_0 = \mathbf{0}_{n+1}$ and updates it at each mistake.

Let $\hat{\mathbf{w}}_{t-1}$ be the augmented weight vector prior to the k^{th} mistake. The k^{th} update is performed when

$$y_i \hat{\mathbf{w}}'_{k-1} \hat{\mathbf{x}}_i = \begin{pmatrix} \boldsymbol{w}_{k-1} \\ \frac{b_{k-1}}{R} \end{pmatrix}' \begin{pmatrix} \boldsymbol{x}_i \\ R \end{pmatrix} = y_i(\boldsymbol{w}'_{k-1}\boldsymbol{x}_i + b_{k-1}) \leqslant 0,$$

where $(\boldsymbol{x}_i, y_i)$ is the example incorrectly classified by

$$\hat{\mathbf{w}}_{k-1} = \begin{pmatrix} \boldsymbol{w}_{k-1} \\ \frac{b_{k-1}}{R} \end{pmatrix}.$$

The update is

$$
\begin{aligned}
\hat{\mathbf{w}}_k &= \begin{pmatrix} \boldsymbol{w}_k \\ \frac{b_k}{R} \end{pmatrix} = \begin{pmatrix} \boldsymbol{w}_{k-1} + \eta y_i \boldsymbol{x}_i \\ \frac{b_{k-1} + \eta y_i R^2}{R} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{w}_{k-1} + \eta y_i \boldsymbol{x}_i \\ \frac{b_{k-1}}{R} + \eta y_i R \end{pmatrix} = \begin{pmatrix} \boldsymbol{w}_{k-1} \\ \frac{b_{k-1}}{R} \end{pmatrix} + \begin{pmatrix} \eta y_i \boldsymbol{x}_i \\ \eta y_i R \end{pmatrix} \\
&= \hat{\mathbf{w}}_{k-1} + \eta y_i \hat{\mathbf{x}}_i,
\end{aligned}
$$

where we used the fact that $b_k = b_{k-1} + \eta y_i R^2$.

Since

$$y_i \hat{\mathbf{w}}'_{opt} \hat{\mathbf{x}}_i = y_i \begin{pmatrix} \boldsymbol{w}'_{opt} & \frac{b}{R} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_i \\ R \end{pmatrix} = y_i(\boldsymbol{w}'_{opt} \hat{\mathbf{x}}_i + b) \geqslant \gamma,$$

we have

$$\hat{\mathbf{w}}'_{opt} \hat{\mathbf{w}}_k = \hat{\mathbf{w}}'_{opt} \hat{\mathbf{w}}'_{k-1} + \eta y_i \hat{\mathbf{w}}'_{opt} \boldsymbol{x}_i \geqslant \hat{\mathbf{w}}'_{opt} \hat{\mathbf{w}}_{k-1} + \eta\gamma.$$

By repeated application of the inequality $\boldsymbol{w}'_{opt} \hat{\mathbf{w}}_k \geqslant \eta\gamma$ we obtain

$$\hat{\mathbf{w}}'_{opt} \hat{\mathbf{w}}_k \geqslant k\eta\gamma.$$

Since $\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_{k-1} + \eta y_i \hat{\mathbf{x}}_i$, we have

$$\begin{aligned}
\parallel \hat{\mathbf{w}}_k \parallel^2 &= \hat{\mathbf{w}}'_k \hat{\mathbf{w}}_k = (\hat{\mathbf{w}}'_{k-1} + \eta y_i \hat{\mathbf{x}}'_i)(\hat{\mathbf{w}}_{k-1} + \eta y_i \hat{\mathbf{x}}_i) \\
&= \parallel \hat{\mathbf{w}}_{k-1} \parallel^2 + 2\eta y_i \hat{\mathbf{w}}'_{k-1} \boldsymbol{x}_i + \eta^2 \parallel \hat{\mathbf{x}}_i \parallel^2 \\
&\leqslant \parallel \hat{\mathbf{w}}_{k-1} \parallel^2 + \eta^2 \parallel \hat{\mathbf{x}}_i \parallel^2 \\
&\quad \text{(because } y_i \hat{\mathbf{w}}'_{k-1} \hat{\mathbf{x}}_i \leqslant 0 \text{ when an update occurs)} \\
&\leqslant \parallel \hat{\mathbf{w}}_{k-1} \parallel^2 + \eta^2 (\parallel \boldsymbol{x}_i \parallel^2 + R^2) \\
&\leqslant \parallel \hat{\mathbf{w}}_{k-1} \parallel^2 + 2\eta^2 R^2,
\end{aligned}$$

which implies $\parallel \hat{\mathbf{w}}_k \parallel^2 \leqslant 2k\eta^2 R^2$, hence $\parallel \hat{\mathbf{w}}_k \parallel \leqslant \sqrt{2k}\eta R$.

Since $\hat{\mathbf{w}}'_{opt}\hat{\mathbf{w}}_k \geqslant k\eta\gamma$, we have $\| \hat{\mathbf{w}}_{opt} \| \| \hat{\mathbf{w}}_k \| \geqslant k\eta\gamma$. Taking into account that $\| \hat{\mathbf{w}}_k \| \leqslant \sqrt{2k}\eta R$ we obtain

$$k\eta\gamma \leqslant \| \hat{\mathbf{w}}_{opt} \| \| \hat{\mathbf{w}}_k \| \leqslant \| \hat{\mathbf{w}}_{opt} \| \sqrt{2k}\eta R,$$

which implies $\sqrt{k} \leqslant \| \hat{\mathbf{w}}_{opt} \| \sqrt{2}\frac{R}{\gamma}$. Therefore, we have

$$k \leqslant 2\left(\frac{R}{\gamma}\right)^2 \| \hat{\mathbf{w}}_{opt} \|^2 \leqslant \left(\frac{2R}{\gamma}\right)^2$$

because

$$\| \hat{\mathbf{w}}_{opt} \|^2 \leqslant \| \boldsymbol{w}_{opt} \|^2 + 1 = 2.$$

The `apply` Function in R

The `apply` function returns a vector or an array or a list of values obtained by applying a function to margins of an array or matrix. The margins of an array are the rows (1), the columns (2), or both (1:2), in which case, the function is applied to each individual value.

Example:

```
> m ← matrix(c(1:10,11:20),nrow=10,ncol=2)
> m
      [,1]  [,2]
         1    11
         2    12
         3    13
         4    14
         5    15
         6    16
         7    17
         8    18
         9    19
        10    20
> apply(m,1,mean)
[1] 6 7 8 9 10 11 12 13 14 15
```

Example

```
> apply(m,2,mean)
[1] 5.5 15.5
> apply(m,1:2,mean)
```

[,1]	[,2]
1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19
10	20

R Code to generate data separated by $x_2 = x_1 + 0.5$

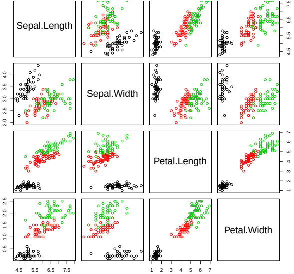
```
> x1 ← runif(50,-1,1)
> x2 ← runif(50,-1,1)
> x ← rbind(x1,x2)
> y ← ifelse(x2 > 0.5 + x1,+1,-1)
> plot(x,pch=ifelse(y > 0,''+'',''-''),xlim=c(-1,1),ylim=c(-1,1),cex=2)
> abline(0.5,1)
> points(c(0,0),c(0,0),pch=19)
> lines(x(0,-0.25),c(0,0.25),lty=2)
> arrows(-0.3,0.2,-0.4,0.3)
> text(-0.45,0.35,"w",cex=2)
> text(-0.0,0.15,"b",cex=2)
```

```
> distance = function(z,w,b) {sum(w*z) + b }
> classify = function(x,w,b) {
+ distances = apply(x,1,distance,w,b)
+ return(iffalse(distances < 0,-1,1))
+ }
euclidean.norm = function(x) {sqrt(sum(x * x))};
```

```
perceptron = function(x, y, eta) {
        w = vector(length = ncol(x));
        b = 0;
        k = 0;
        R = max(apply(x, 1, euclidean.norm));
        mistake = TRUE;
        while (made.mistake) {
            mistake=FALSE;
            yc ← classify.linear(x,w,b);
                for (i in 1:nrow(x)) {
                  if (y[i] ! = yc[i]) {
                  w ← w + eta * y[i]*x[i,];
                  b ← b + eta * y[i]*R^2;
                  k ← k+1;
                  mistake=TRUE;
                }
            }
        }
        s = euclidean.norm(w);
        return(list(w=w/s,b=b/s,updates=k));
}
```

Iris Scatter Plots

```
> data(iris)
> pairs(iris[,1:4], col=iris$Species)
```



Separation of the "setosa" flowers

```
> x ← cbind(iris$Sepal.Width,iris$Petal.Width)
> y ← ifelse(iris$Species == "setosa", +1, -1)
> plot(x,cex=2)
> points(subset(x,y==1),col="black",pch="+",cex=2)
> points(subset(x,y==-1),col="red",pch="-",cex=2)
> source("perc.R")
```

```
> p ← perceptron(x,y,1)
> p
$w
[1] 0.3277371 -0.9447690
$b
[1] -0.2543709
$updates
[1] 202
> intercept ← - p$b/p$w[2]
> slope ← - p$w[1]/p$w[2]
> abline(intercept,slope,col="green")
```