Facoltà di Ingegneria

INGEGNERIA INFORMATICA – Intelligenza Artificiale

Corso: Machine Learning

Docente: Prof. LUCA IOCCHI

LINEAR MODELS FOR CLASSIFICATION

(BINARY CLASSIFICATION IN $\mathbb{R}^2$)

REDJAN SHABANI

(1013173)

Sommario

1-Least Squares

Least squares is widely use in the linear fitting problems. This technique can also be used for classifying data. Each class of data is described by its own linear model. If we are considering binary classification the linear model of each class have the form:

$$y_1(\mathbf{x}) = \mathbf{w}_1^T\mathbf{x} + w_1^0$$

$$y_2(\mathbf{x}) = \mathbf{w}_2^T\mathbf{x} + w_2^0$$

If the data are defined in $\mathbb{R}^2$ we can rewrite the above relations more explicitly:

$$y_1 = a_1x_1 + b_1x_2 + c_1$$

$$y_2 = a_2x_1 + b_2x_2 + c_2$$

grouping together we can write more compactly:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_1 & a_1 & b_1 \\ c_2 & a_2 & b_2 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

or:

$$\mathbf{y} = \widetilde{W}^T \tilde{\mathbf{x}}$$

where $\mathbf{y} = [y_1 \quad y_2]^T$, $\widetilde{W} = \begin{bmatrix} c_1 & c_2 \\ a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}$, $\tilde{\mathbf{x}} = [1 \quad x_2 \quad x_1]^T$.

Now we have to find the parameter matrix $\widetilde{W}$by minimizing a sum-of-squares error function. Let's consider a 2D training data set $\{\dots, (\mathbf{x}^{(n)}, \mathbf{t}^{(n)}), \dots\}$ where $n = 1 \dots N$, and define the following matrices:

$$T = \begin{bmatrix} \mathbf{t}^{(1)} \\ \vdots \\ \mathbf{t}^{(N)} \end{bmatrix}$$

$$\widetilde{X} = \begin{bmatrix} \tilde{\mathbf{x}}^{(1)} \\ \vdots \\ \tilde{\mathbf{x}}^{(N)} \end{bmatrix}$$

The sum of squared errors can then be written as:

$$E_D(\widetilde{W}) = \frac{1}{2} \mathrm{Tr}\left\{ (\widetilde{X}\widetilde{W} - T)^T (\widetilde{X}\widetilde{W} - T) \right\}$$

Setting the derivative with respect to $\widetilde{W}$ to zero we can obtain:

$$\widetilde{W} = X^{\dagger} T$$

The directory \Least Squares\.. contains the matlab code for the least squares classification in two dimensional space for two classes of data. The results can be observed directly here (..\Least Squares\html\leastSquaresTest.html).

2-Fisher Discriminant

The idea of the Fisher procedure is to find a projection of data points that maximizes the class separation. If the data are defined in two dimensional real-valued space we have to find the versor of a straight line.

Let we consider to classes A and B containing respectively $N_1$and N_2 points in$\mathbb{R}^2$. The mean vectors are given by:

$$\mathbf{m}^A = \begin{bmatrix} m_x^A \\ m_y^A \end{bmatrix} = \sum_{(x,y) \in A} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{m}^B = \begin{bmatrix} m_x^B \\ m_y^B \end{bmatrix} = \sum_{(x,y) \in B} \begin{bmatrix} x \\ y \end{bmatrix}$$

where $[x \ \ y]^T$is the vector representaion of the generic point (x, y) the plain(fig.1).

The simplest measure of class separation, when projected along $\mathbf{w} = \begin{bmatrix} a \\ b \end{bmatrix}$(a plain vector), is the separation of the projected class means:

$$m^A - m^B = \mathbf{w}^T(\mathbf{m}^A - \mathbf{m}^B) = [a \quad b] \begin{bmatrix} m_x^A - m_x^B \\ m_y^A - m_y^B \end{bmatrix}$$

where $m^A = \mathbf{w}^T\mathbf{m}^A$ and $m^B = \mathbf{w}^T\mathbf{m}^B$ are the mean of the projected data. To have a significant maximization of means separation we have to constrain $\mathbf{w}$ to have unit length

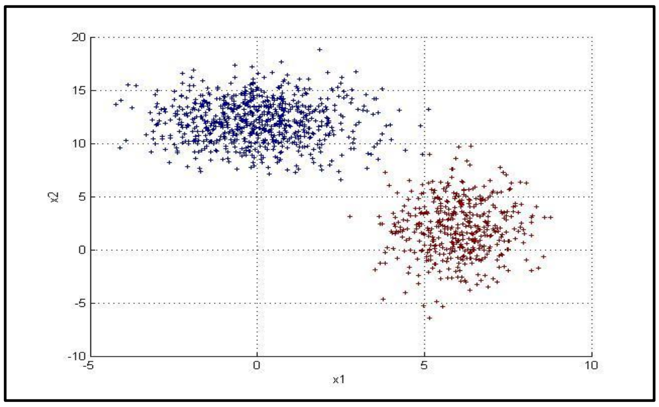


Figura 1-Example of two classes of points in the plane.

($\sqrt{a^2+b^2}=1$), so the only thing to do is to change the angle of the vector. At this point we can decide $\mathbf{w}$ to be:

$$\mathbf{w}=\frac{1}{|\mathbf{m}^A-\mathbf{m}^B|}(\mathbf{m}^A-\mathbf{m}^B)$$

$$\Rightarrow \begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\sqrt{(m_x^A-m_x^B)^2+\left(m_y^A-m_y^B\right)^2}} \begin{bmatrix} m_x^A-m_x^B \\ m_y^A-m_y^B \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \left(1+\left(m_y^A-m_y^B\right)^2\right)^{-1/2} \\ \left(1+\left(m_x^A-m_x^B\right)^2\right)^{-1/2} \end{bmatrix}$$

There is still some rectification to make to the formula, because the two classes even if there are well separated may have considerable overlap when projected onto the line joining the means. The rectification proposed by Fisher, is to maximize a function that gives the maximum separation between projected class means also giving a small variance between this classes and consequently minimizing the class overlap.

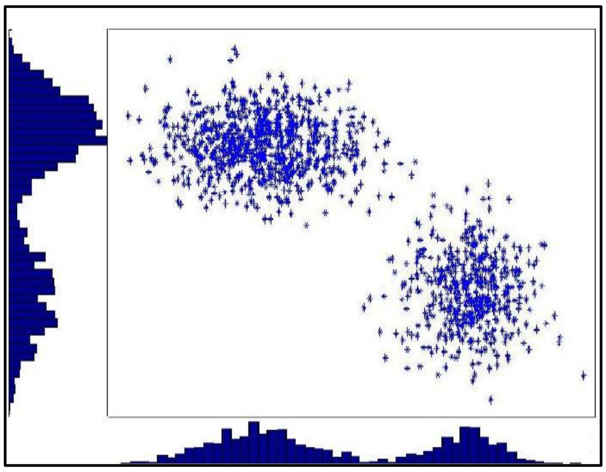


Figura 2-Histogram of projected classes over x-axis and y-axis

The Fisher criterion is:

$$J(\mathbf{w})=\frac{(m^A-m^B)^2}{s_1^2+s_2^2}$$

where $s^C=\sqrt{\sum_{x\in C}(\mathbf{x}-\mathbf{m}_1)}$, $C\in\{A,B\}$. To make explicit the dependence to $\mathbf{w}$ we can rewrite the Fisher criterion in the form:

$$J(\mathbf{w})=\frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

where $S_B$is the between-class variance matrix, given by:

$$S_B=(\mathbf{m}^A-\mathbf{m}^B)(\mathbf{m}^A-\mathbf{m}^B)^T$$

Rewriting for the 2D case:

$$S_B=\begin{bmatrix} (m_x^A-m_x^B)^2 & (m_x^A-m_x^B)(m_y^A-m_y^B) \\ (m_x^A-m_x^B)(m_y^A-m_y^B) & (m_y^A-m_y^B)^2 \end{bmatrix}$$

and S_W is the total within-class covariance matrix, given by:

$$S_W=\sum_{x\in A}(\mathbf{x}-\mathbf{m}^A)(\mathbf{x}-\mathbf{m}^A)^T+\sum_{x\in B}(\mathbf{x}-\mathbf{m}^B)(\mathbf{x}-\mathbf{m}^B)^T$$

explicitating for the 2D case:

$$S_W=\sum_{(x,y)\in A}\begin{bmatrix} (x-m_x^A)^2 & (x-m_x^A)(y-m_y^A) \\ (x-m_x^A)(y-m_y^A) & (y-m_y^A)^2 \end{bmatrix}+\sum_{(x,y)\in B}\begin{bmatrix} (x-m_x^B)^2 & (x-m_x^B)(y-m_y^B) \\ (x-m_x^B)(y-m_y^B) & (y-m_y^B)^2 \end{bmatrix}$$

Differentiating $J(\mathbf{w})$ with respect to $\mathbf{w}$ we find that the criterion is maximized when:

$$(\mathbf{w}^T S_B \mathbf{w})S_W\mathbf{w}=(\mathbf{w}^T S_W \mathbf{w})S_B\mathbf{w}$$

Furthermore, we don't care about the the magnitude of $\mathbf{w}$, only its direction, and so we can drop the scalar factors $\mathbf{w}^{\mathrm{T}}\mathrm{S}_{\mathrm{B}}\mathbf{w}$ and $(\mathbf{w}^{\mathrm{T}}\mathrm{S}_{\mathrm{W}}\mathbf{w})$:

$$\mathbf{w} \propto \mathbf{S}_{\mathbf{W}}(\mathbf{m}^{\mathbf{A}} - \mathbf{m}^{\mathbf{B}})$$

This result is known as the Fisher linear discriminant, although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension. In the projected data we have to find with some other procedure a value c such that $\mathrm{ax} + \mathrm{by} + \mathrm{c} > 0$ to classify $\begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix}$as belonging in the positive class and classify it as belonging in the negative class otherwise.

It can be shown that Fisher's discriminant is related to Leas Squares Discriminant. For a two class problem the Fisher Criterion can be obtained as particular case of the least squares. After few passages it is possible to obtain a formula for the bias of the Fisher Discriminant, wich has the form:

$$c = -\boldsymbol{w}^T\boldsymbol{m}$$

where $\boldsymbol{m}$ is the mean of both data classes.

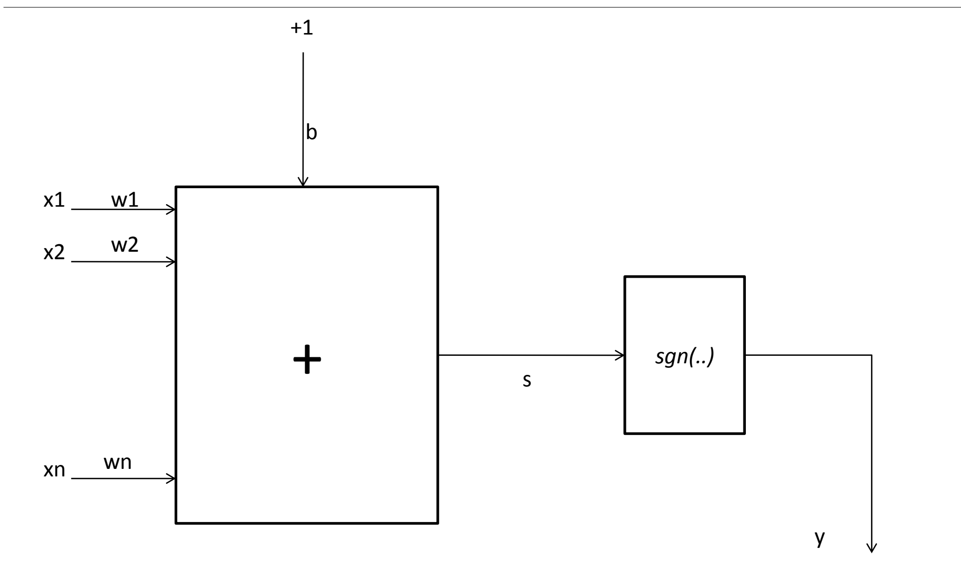
At this point we have a complete description of the discriminant equation:

$$ax_1 + bx_2 + c = 0$$

The directory \Fisher\.. contains the matlab code for the Fisher discriminant procedure in two dimensional space for two classes of data. The results can be observed directly here (..\Fisher\html\fishetTest.html).

3-Perceptron Algorithm

In Artificial Neural Network, is called Perceptron Unit a neural unit with threshold function:



$$sgn: \mathbb{R} \rightarrow \{-1, +1\}$$

$$sgn(x) = \begin{cases} -1, & iff\ x > 0 \\ +1, & otherwise \end{cases}$$

The transfer unit function is defined by:

$$y = sgn(s) = sgn(b + \sum_{i=1,..,n} w_i x_i)$$

if we want to use vector notation for the sum operator we obtain a more compact function:

$$y = sgn(b + w^T x)$$

where $w = [w_1 \dots w_n]^T$, $x = [x_1 \dots x_n]^T$.
Including even the bias as a weight applied to a costant input +1 we can write:

$$y = sgn(\widetilde{\boldsymbol{w}}^T\widetilde{\boldsymbol{x}})$$

where $\widetilde{\boldsymbol{w}} = [b\ w_1 \dots w_n]^T$, $\widetilde{\boldsymbol{x}} = [1\ x_1 \dots x_n]^T$.

We are considering a 2D problem, so the input vector is 2 dimensional and so is the weight vector. The perceptron unit, in the 2D space will have the sequent transfer function:

$$y = sgn\left([b\ w_1\ w_2]\begin{bmatrix}1\\x_1\\x_2\end{bmatrix}\right) = sgn(b + w_1x_1 + w_2x_2)$$

In general $\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}} = 0$ defines a (n-1)-dimensional hyper plane in the n-dimensional space. In the 2-dimensional space the hyper plane is the straight line defined by:

$$b + w_1x_1 + w_2x_2 = 0$$

Every point of the plane standing on this straight line verifies the above equation. Points of the plane up the straight line are defined by the relation:

$$b + w_1x_1 + w_2x_2 > 0$$

and points of the plane down the straight line are defined by the relation:

$$b + w_1x_1 + w_2x_2 < 0$$

As we can see, the perceptron unit divides the n-dimensional space in two parts. In particular the 2-dimensional plane can be divided in two semi planes.

For the 2-dimensional classification problem, a perceptron unit can be trained to find a straight line(linear discriminant) that separates two given data classes linearly separable. The procedure that learns to the perceptron the equation of the discriminant is called Perceptron Learning Rule. If we pass to the perceptron the coordinate of a point in the plane it will produce in output +1 if the point is up the line and -1 otherwise.

Suppose we have two classes of data A and B respectively containing N_1 and N_2 data points. First we merge this two classes in a unique set of points adding a third component 't' for each element, where t=+1 if the point is in class A and t=-1 if the point is in class B. The dataset will have this structure:

$$D = \left\{\dots, \begin{bmatrix}x_1^{(k)}\\x_2^{(k)}\\t^{(k)}\end{bmatrix}, \dots\right\}$$

for $k = 1, \dots, N$ $(N = N_1 + N_2)$. The dataset D is also called training set.

Now we have to define an error function that tells us how far is the optimal solution. A natural way to define the error function is to count the number of misclassified data:

$$E_p(\tilde{\boldsymbol{w}}) = \sum_{i=1}^{N} y(\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}) = \sum_{i=1}^{N}\left[sgn\left(b + w_1x_1^{(i)} + x_2^{(i)}\right) - t^{(i)}\right]^2$$

The optimal $\tilde{\boldsymbol{w}}$ is that who minimizes E_p. Methods based on changing $\tilde{\boldsymbol{w}}$ using the gradient of $E_p(\tilde{\boldsymbol{w}})$ cannot be applied because of the discontinuities of the function $sgn(\dots)$. Furthermore it is possible to apply a sort of stochastic gradient descent algoritm to train the perceptron. For the 2-dimensional case the perceptron training algorithm is the following:

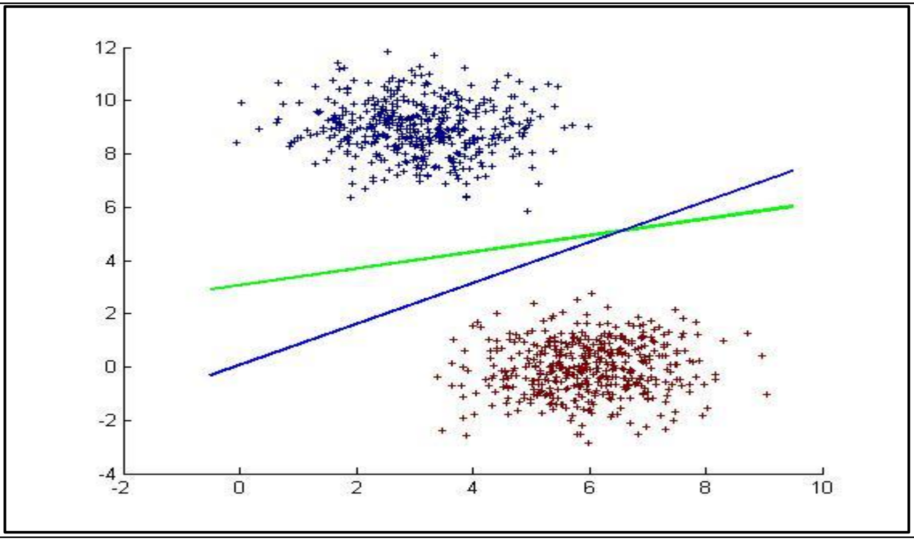
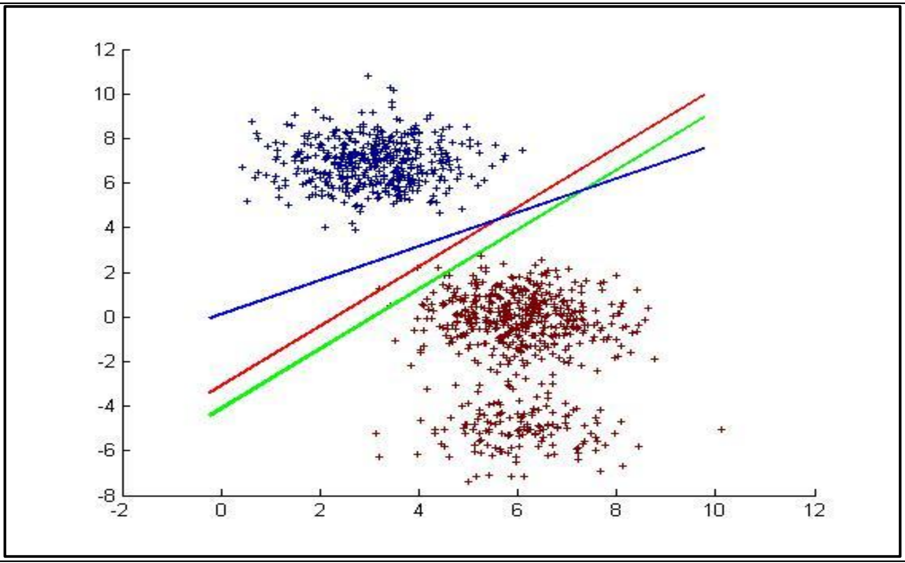
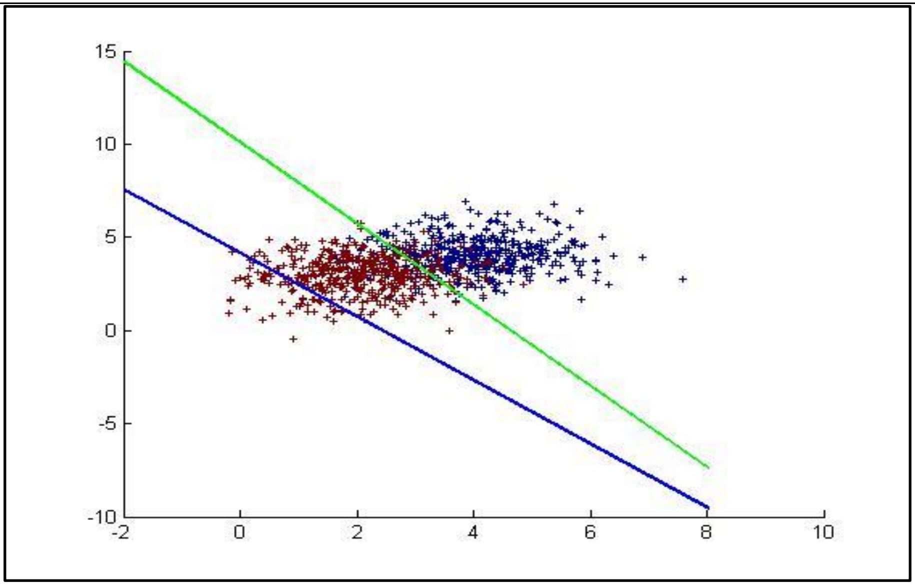
```
• $until\ E_p \neq 0\ repeat$
    ○ $for\ each\ \left[x_1^{(k)}\ x_2^{(k)}\ t^{(k)}\right]^T, k = 1, \dots, N$
        ▪ $b \leftarrow b + \eta\left(t^{(k)} - sgn\left(b + w_1x_1^{(k)} + w_2x_1^{(k)}\right)\right)$
        ▪ $w_1 \leftarrow w_1 + \eta\left(t^{(k)} - sgn\left(b + w_1x_1^{(k)} + w_2x_1^{(k)}\right)\right)x_1^{(k)}$
        ▪ $w_2 \leftarrow w_2 + \eta\left(t^{(k)} - sgn\left(b + w_1x_1^{(k)} + w_2x_1^{(k)}\right)\right)x_1^{(k)}$
```

The constant factor $\eta(\approx 0.05)$ is called learning rate and is related to the precision of the algorithm.

The directory \Perceptron\.. contains the matlab code for the Perceptron algorithm in two dimensional space for two classes of data. The results can be observed directly here(../Perceptron/html/perceptronTest.html).

4-Results

In the following table are represented some tests results of the three classifications procedure over different datasets. The solution given by the procedures have different colours:green for Fisher/Least Squares, blue for Perceptron.

Linearly Separated Classes The discriminant found by the perceptron is a good discriminant but is different with respect to the others, this because the result depends from the initial unit weights.	
Linearly Separated Classes with outliers As we can see, the perceptron algorithm is the best for linearly approximated classification, but we have to consider the computational cost of training.	
Strongly non linearly separated The discriminant found by the Perceptron algorithm have a considerable error. This is not because the Perceptron Algorithm cannot find a good solution but because the number of iterations was not sufficient to conduce to the minimum of error.	

References

1-Cristopher M. Bishop - PATTERN RECOGNITION AND MACHINE LEARNING