Linear classification

Course of Machine Learning
Master Degree in Computer Science
University of Rome "Tor Vergata"
a.a. 2023-2024

Giorgio Gambosi

Classification

- value t to predict are from a discrete domain, where each value denotes a **class**
- most common case: disjoint classes, each input has to assigned to exactly one class
- input space is partitioned into **decision regions**
- in **linear classification models** decision boundaries are linear functions of input $\mathbf{x}$ ($D-1$-dimensional hyperplanes in the D-dimensional feature space)
- datasets such as classes correspond to regions which may be separated by linear decision boundaries are said **linearly separable**

Regression vs classification

- Regression: the target variable $\mathbf{t}$ is a vector of reals
- Classification: several ways to represent classes (target variable values)
- Binary classification: a single variable $t \in \{0, 1\}$, where $t = 0$ denotes class C_0 and $t = 1$ denotes class C_1
- $K > 2$ classes: "1 of K" coding. $\mathbf{t}$ is a vector of K bits, such that for each class C_j all bits are 0 except the j-th one (which is 1)

Three general approaches to classification

1. find $f : \mathbf{X} \mapsto \{1, \ldots, K\}$ (**discriminant function**) which maps each input $\mathbf{x}$ to some class C_i, such that $i = f(\mathbf{x})$
2. **discriminative approach**: determine the conditional probabilities $p(C_j|\mathbf{x})$ (**inference phase**); use these distributions to assign an input to a class (**decision phase**)
3. **generative approach**: determine the class conditional distributions $p(\mathbf{x}|C_j)$, and the class prior probabilities $p(C_j)$; apply Bayes' formula to derive the class posterior probabilities $p(C_j|\mathbf{x})$; use these distributions to assign an input to a class

Approaches 1 and 2 are **discriminative**: they tackle the classification problem by deriving from the training set conditions (such as decision boundaries) that , when applied to a point, discriminate each class from the others. The boundaries between regions are specified by **discrimination functions**

In linear regression, a model predicts the target value; the prediction is made through a linear function $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ (linear basis functions could be applied). In classification, a model predicts probabilities of classes, that is values in $[0, 1]$; the prediction is made through a generalized linear model $y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$, where f is a non linear activation function with codomain $[0, 1]$

Boundaries correspond to solution of $y(\mathbf{x}) = c$ for some constant c; this results into $w^T\mathbf{x} + w_0 = f^{-1}(c)$, that is a linear boundary. The inverse function f^{-1} is said link function.

Approach 3 is generative: it works by defining, from the training set, a model of items for each class

The model is a probability distribution (of features conditioned by the class) and could be used for random generation of new items in the class. By comparing an item to all models, it is possible to verify the one that best fits

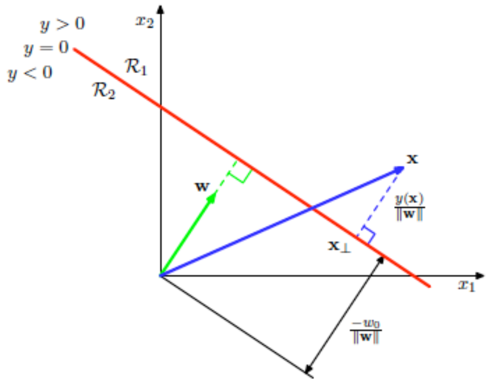
Discriminant functions

Linear discriminant functions in binary classification

- Decision boundary: $D - 1$-dimensional hyperplane of all points s.t. $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$
- Given $\mathbf{x}_1, \mathbf{x}_2$ on the hyperplane, $y(\mathbf{x}_1) = y(\mathbf{x}_2) = 0$. Hence,

  $$\mathbf{w}^T\mathbf{x}_1 + w_0 - \mathbf{w}^T\mathbf{x}_2 - w_0 = \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$$

  that is, vectors $\mathbf{x}_1 - \mathbf{x}_2$ and $\mathbf{w}$ are orthogonal
- For any $\mathbf{x}$, the dot product $\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T\mathbf{x}$ is the length of the projection of $\mathbf{x}$ in the direction of $\mathbf{w}$ (orthogonal to the hyperplane $\mathbf{w}^T\mathbf{x} + w_0 = 0$), in multiples of $||\mathbf{w}||_2$
- By normalizing wrt to $||\mathbf{w}||_2 = \sqrt{\sum_i w_i^2}$, we get the length of the projection of $\mathbf{x}$ in the direction orthogonal to the hyperplane, assuming $||\mathbf{w}||_2 = 1$
- For any $\mathbf{x}$, $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ returns the distance (in multiples of $||\mathbf{w}||$) of $\mathbf{x}$ from the hyperplane
- The sign of the returned value discriminates in which of the regions separated by the hyperplane the point lies



Linear discriminant functions in multiclass classification

- Define K linear functions

  $$y_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \qquad\qquad 1 \le i \le K$$

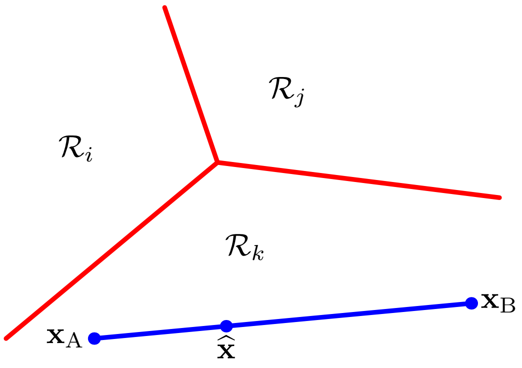
  Item $\mathbf{x}$ is assigned to class C_k iff $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$: that is,

  $$k = \underset{j}{\text{argmax}}\, y_j(\mathbf{x})$$

- Decision boundary between C_i and C_j: all points $\mathbf{x}$ s.t. $y_i(\mathbf{x}) = y_j(\mathbf{x})$, a $D-1$-dimensional hyperplane

$$(\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} + (w_{i0} - w_{j0}) = 0$$

The resulting decision regions are connected and convex



- The definition can be extended to include terms relative to products of pairs of feature values (Quadratic discriminant functions)

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=1}^{i} w_{ij} x_i x_j$$

$\dfrac{d(d+1)}{2}$ additional parameters wrt the $d+1$ original ones: decision boundaries can be more complex

- In general, generalized discriminant functions through set of functions $\phi_i, \dots, \phi_m$

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{M} w_i \phi_i(\mathbf{x})$$

Least squares and classification

- Assume classification with K classes
- Classes are represented through a 1-of-K coding scheme: set of variables $z_1, \dots, z_K$, class C_i coded by values $z_i = 1$, $z_k = 0$ for $k \neq i$
- K discriminant functions y_i are derived as linear regression functions with variables z_i as targets
- To each variable z_i a discriminant function $y_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$ is associated: $\mathbf{x}$ is assigned to the class C_k s.t.

$$k = \underset{i}{\operatorname{argmax}}\, y_i(\mathbf{x})$$

- Then, $z_k(\mathbf{x}) = 1$ and $z_j(\mathbf{x}) = 0$ $(j \neq k)$ if $k = \underset{i}{\operatorname{argmax}}\, y_i(\mathbf{x})$
- Group all parameters together as

$$\mathrm{y}(\mathbf{x}) = \mathrm{W}^T \overline{\mathrm{x}} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1D} \\ w_{20} & w_{21} & \cdots & w_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K0} & w_{K1} & \cdots & w_{KD} \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix}$$

- In general, a regression function provides an estimation of the target given the input $E[t|\mathbf{x}]$
- $y_i(\mathbf{x})$ can be seen as an estimate of the conditional expectation $E[z_i|\mathbf{x}]$ of binary variable z_i given $\mathbf{x}$
- If we assume z_i is distributed according to a Bernoulli distribution, the expectation corresponds to the posterior probability

$$\begin{aligned} y_i(\mathbf{x}) &\simeq E[z_i|\mathbf{x}] \\ &= P(z_i = 1|\mathbf{x}) \cdot 1 + P(z_i = 0|\mathbf{x}) \cdot 0 \\ &= P(z_i = 1|\mathbf{x}) \\ &= P(C_i|\mathbf{x}) \end{aligned}$$

- However, $y_i(\mathbf{x})$ is not a probability itself (we may not assume it takes value only in the interval $[0, 1]$)

Learning functions y_i

Given a training set $\mathbf{X}, \mathbf{t}$, a regression function can be derived by least squares

- An item in the training set is a pair $(\mathbf{x}_i, \mathbf{t}_i)$, $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{t}_i \in \{0, 1\}^K$
- $\overline{\mathbf{X}} \in \mathbb{R}^{n \times (D+1)}$ is the matrix of feature values for all items in the training set

$$\overline{\mathbf{X}} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1D} \\ 1 & x_{21} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nD} \end{pmatrix}$$

- Then, for matrix $\mathbf{Y} = \overline{\mathbf{X}}\mathbf{W}$, of size $n \times K$, we have $\mathbf{Y}_{ij} = w_{j0} + \sum_{k=1}^{D} x_{ik}w_{jk} = y_j(\mathbf{x}_i)$ hence

$$\overline{\mathbf{Y}} = \begin{pmatrix} y_1(x_1) & y_2(x_1) & \cdots & y_K(x_1) \\ y_1(x_2) & y_2(x_2) & \cdots & y_K(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ y_1(x_n) & y_2(x_n) & \cdots & y_K(x_n) \end{pmatrix}$$

  where, as observed before, $y_j(\mathbf{x}_i)$ is the estimate of $p(C_j|\mathbf{x}_i)$

  All targets, coded in 1-of-K format, can be represented as a $n \times K$ matrix $\mathbf{T}$, where $\mathbf{T}_{ij} = t_{ij}$.

$$\mathbf{T} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1K} \\ t_{21} & t_{22} & \cdots & t_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nK} \end{pmatrix}$$

As usual, $y_j(\mathbf{x}_i)$ is then compared to t_{ij}, providing the residue

$$r_{ij} = y_j(\mathbf{x}_i) - t_{ij} = \sum_{k=1}^{D} x_{ik}w_{jk} + w_{j0} - t_{ij} = (\overline{\mathbf{X}}\mathbf{W} - \mathbf{T})_{ij}$$

$$\mathbf{R} = \begin{pmatrix} y_1(\mathbf{x}_1) - t_{11} & y_2(\mathbf{x}_1) - t_{12} & \cdots & y_K(\mathbf{x}_1) - t_{1K} \\ y_1(\mathbf{x}_2) - t_{21} & y_2(\mathbf{x}_2) - t_{22} & \cdots & y_K(\mathbf{x}_2) - t_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ y_1(\mathbf{x}_n) - t_{n1} & y_2(\mathbf{x}_n) - t_{n2} & \cdots & y_K(\mathbf{x}_n) - t_{nK} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ r_{21} & r_{22} & \cdots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nK} \end{pmatrix}$$

- If we consider the $K \times K$ matrix $\mathbf{R}^T\mathbf{R}$, we have that

$$\mathbf{R}^T\mathbf{R} = \begin{pmatrix} r_{11} & r_{21} & \cdots & r_{n1} \\ r_{12} & r_{22} & \cdots & r_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1n} & r_{n2} & \cdots & r_{nK} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1K} \\ r_{21} & r_{22} & \cdots & r_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nK} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n r_{i1}^2 & \sum_{i=1}^n r_{i1}r_{i2} & \cdots & \sum_{i=1}^n r_{i1}r_{iK} \\ \sum_{i=1}^n r_{i2}r_{i1} & \sum_{i=1}^n r_{i2}^2 & \cdots & \sum_{i=1}^n r_{i2}r_{iK} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^n r_{iK}r_{i1} & \sum_{i=1}^n r_{iK}r_{i2} & \cdots & \sum_{i=1}^n r_{iK}^2 \end{pmatrix}$$

- Summing all elements on the diagonal of $\mathbf{R}^T\mathbf{R}$ provides the overall sum, on all items in the training set, of the squared differences between observed values and values computed by the model, with parameters $\mathbf{W}$, that is

$$\sum_{j=1}^{K}\sum_{i=1}^{n}(y_j(\mathbf{x}_i) - t_{ij})^2$$

- This corresponds to the **trace** of $\mathbf{R}^T\mathbf{R}$. Hence, we have to minimize:

$$E(\mathbf{W}) = \frac{1}{2}\text{tr}\left(\mathbf{R}^T\mathbf{R}\right)$$

- If we apply the standard approach of trying to solve

$$\frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} = \mathbf{0}$$

  it is possible to show that

$$\frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} = \overline{\mathbf{X}}^T\overline{\mathbf{X}}\mathbf{W} - \overline{\mathbf{X}}^T\mathbf{T}$$

- which is equal to $\mathbf{0}$ if

$$\mathbf{W} = (\overline{\mathbf{X}}^T\overline{\mathbf{X}})^{-1}\overline{\mathbf{X}}^T\mathbf{T}$$

The resulting set of discriminant functions is then

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}^T\overline{\mathbf{x}} = \mathbf{T}^T\overline{\mathbf{X}}(\overline{\mathbf{X}}^T\overline{\mathbf{X}})^{-1}\overline{\mathbf{x}}$$

Fisher linear discriminant

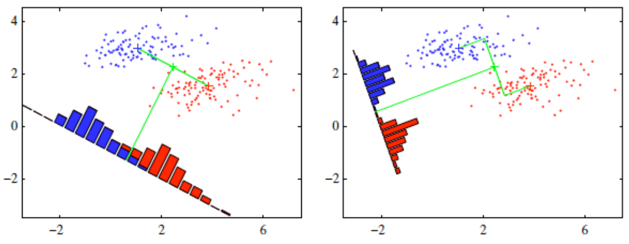
The idea of *Linear Discriminant Analysis* (*LDA*) is to find a linear projection of the training set into a suitable subspace where classes are as linearly separated as possible.

A common approach is provided by **Fisher linear discriminant**, where all items in the training set (points in a D-dimensional space) are projected to one dimension, by means of a linear transformation of the type

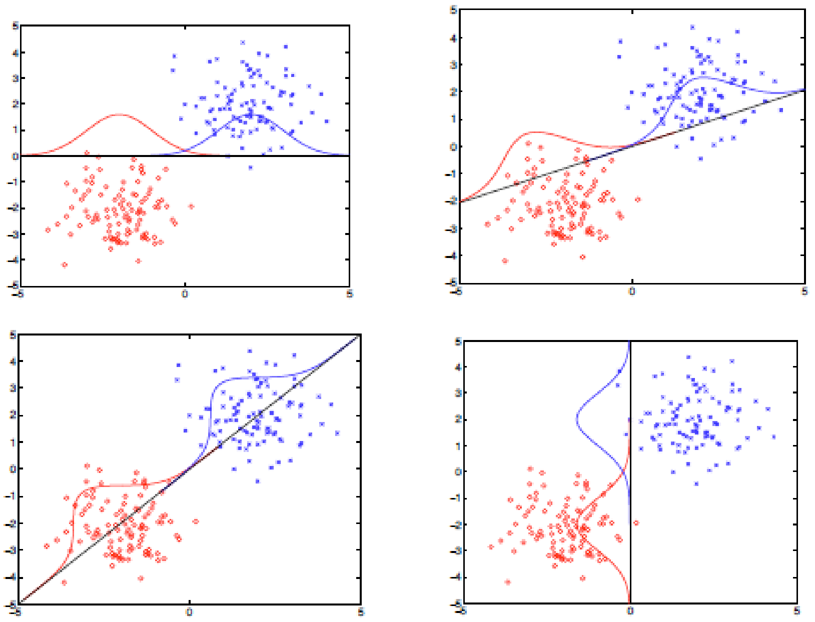
$$y = \mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T\mathbf{x}$$

where $\mathbf{w}$ is the D-dimensional vector corresponding to the direction of projection (in the following, we will consider the one with unit norm).

If $K = 2$, given a threshold $\tilde{y}$, item $\mathbf{x}$ is assigned to C_1 iff its projection $y = \mathbf{w}^T\mathbf{x}$ is such that $y > \tilde{y}$; otherwise, $\mathbf{x}$ is assigned to C_2.



Different line directions, that is different parameters **w**, may induce quite different separability properties.



Deriving w in the binary case

Let n_1 be the number of items in the training set belonging to class C_1 and n_2 the number of items in class C_2. The mean points of both classes are

$$\mathbf{m}_1 = \frac{1}{n_1}\sum_{\mathbf{x}\in C_1}\mathbf{x} \qquad \mathbf{m}_2 = \frac{1}{n_2}\sum_{\mathbf{x}\in C_2}\mathbf{x}$$

A simple measure of the separation of classes, when the training set is projected onto a line, is the difference between the projections of their mean points

$$m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$$

where $m_i = \mathbf{w}^T\mathbf{m}_i$ is the projection of $\mathbf{m}_i$ onto the line.

- We wish to find a line direction **w** such that $m_2 - m_1$ is maximum
- $\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$ can be made arbitrarily large by multiplying **w** by a suitable constant, at the same time maintaining the direction unchanged. To avoid this drawback, we consider unit vectors, introducing the constraint $||\mathbf{w}||_2 = \mathbf{w}^T\mathbf{w} = 1$
- This results into the constrained optimization problem

$$\max_{\mathbf{w}} \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$$
$$\text{where } \mathbf{w}^T\mathbf{w} = 1$$

- This can be transformed into an equivalent unconstrained optimization problem by means of lagrangian multipliers

$$\max_{\mathbf{w},\lambda} \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \mathbf{w}^T\mathbf{w})$$

Setting the gradient of the function wrt **w** to **0**

$$\frac{\partial}{\partial\mathbf{w}}(\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \mathbf{w}^T\mathbf{w})) = \mathbf{m}_2 - \mathbf{m}_1 + 2\lambda\mathbf{w} = \mathbf{0}$$

results into

$$\mathbf{w} = \frac{\mathbf{m}_2 - \mathbf{m}_1}{2\lambda}$$

Setting the derivative wrt λ to 0

$$\frac{\partial}{\partial \lambda}(\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \mathbf{w}^T\mathbf{w})) = 1 - \mathbf{w}^T\mathbf{w} = 0$$

results into

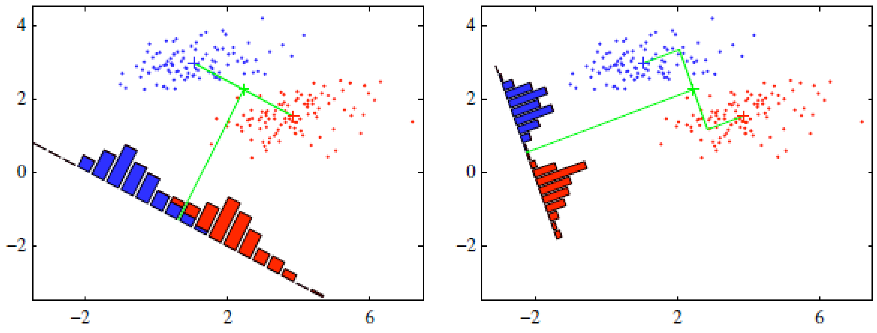
$$\lambda = \frac{\sqrt{(\mathbf{m}_2 - \mathbf{m}_1)^T(\mathbf{m}_2 - \mathbf{m}_1)}}{2} = \frac{||\mathbf{m}_2 - \mathbf{m}_1||_2}{2}$$

Combining with the result for the gradient, we get

$$\mathbf{w} = \frac{\mathbf{m}_2 - \mathbf{m}_1}{||\mathbf{m}_2 - \mathbf{m}_1||_2}$$

The best direction $\mathbf{w}$ of the line, wrt the measure considered, is the one from $\mathbf{m}_1$ to $\mathbf{m}_2$.

However, this may result in a poor separation of classes.



Projections of classes are dispersed (high variance) along the direction of $\mathbf{m}_1 - \mathbf{m}_2$. This may result in a large overlap.

Refinement:

- Choose directions s.t. classes projections show as little dispersion as possible
- Possible in the case that the amount of class dispersion changes wrt different directions, that is if the distribution of points in the class is elongated
- We wish then to maximize a function which:
  - is growing wrt the separation between the projected classes (for example, their mean points)
  - is decreasing wrt the dispersion of the projections of points of each class
- The **within-class variance** of the projection of class C_i $(i = 1, 2)$ is defined as

$$s_i^2 = \sum_{\mathbf{x} \in C_i} (\mathbf{w}^T\mathbf{x} - m_i)^2$$

  The total within-class variance is defined as $s_1^2 + s_2^2$

- Given a direction $\mathbf{w}$, the **Fisher criterion** is the ratio between the (squared) class separation and the overall within-class variance, along that direction

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- Indeed, $J(\mathbf{w})$ grows wrt class separation and decreases wrt within-class variance

Let $\mathbf{S}_1, \mathbf{S}_2$ be the within-class covariance matrices, defined as

$$\mathbf{S}_i = \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$$

Then,

$$s_i^2 = \sum_{\mathbf{x} \in C_i} (\mathbf{w}^T \mathbf{x} - m_i)^2 = \mathbf{w}^T \mathbf{S}_i \mathbf{w}$$

Let also $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$ be the total within-class covariance matrix and

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

be the between-class covariance matrix.

Then,

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

As usual, $J(\mathbf{w})$ is maximized wrt $\mathbf{w}$ by setting its gradient to $\mathbf{0}$

$$\frac{\partial}{\partial \mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \mathbf{0}$$

which results into

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} - (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w} = \mathbf{0}$$

that is

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$$

Observe that:

- $\mathbf{w}^T \mathbf{S}_B \mathbf{w}$ is a scalar, say c_B
- $w^T \mathbf{S}_W \mathbf{w}$ is a scalar, say c_W
- $(\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}$ is a scalar, say c_m

Then, the condition $(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$ can be written as

$$c_B \mathbf{S}_W \mathbf{w} = c_W (\mathbf{m}_2 - \mathbf{m}_1) c_m$$

which results into

$$\mathbf{w} = \frac{c_W c_m}{c_B} \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

Since we are interested into the direction of $\mathbf{w}$, that is in any vector proportional to $\mathbf{w}$, we may consider the solution

$$\hat{\mathbf{w}} = \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1) = (\mathbf{S}_1 + \mathbf{S}_2)^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

Choosing a threshold.
Possible approach:

- model $p(y|C_i)$ as a gaussian: derive mean and variance by maximum likelihood

$$m_i = \frac{1}{n_i}\sum_{\mathbf{x}\in C_i} w^T\mathbf{x} \qquad \sigma_i^2 = \frac{1}{n_i - 1}\sum_{\mathbf{x}\in C_i}(w^T\mathbf{x} - m_i)^2$$

  where n_i is the number of items in training set belonging to class C_i

- derive the class probabilities

$$p(C_i|y) \propto p(y|C_i)p(C_i) = p(y|C_i)\frac{n_i}{n_1 + n_2} \propto n_i e^{-\frac{(y-m_i)^2}{2\sigma_i^2}}$$

- the threshold $\tilde{y}$ can be derived as the minimum y such that

$$\frac{p(C_2|y)}{p(C_1|y)} = \frac{n_2}{n_1}\frac{p(y|C_2)}{p(y|C_1)} > 1$$

Perceptron

- Introduced in the '60s, at the basis of the neural network approach
- Simple model of a single neuron
- Hard to evaluate in terms of probability
- Works only in the case that classes are linearly separable

It corresponds to a binary classification model where an item $\mathbf{x}$ is classified on the basis of the sign of the value of the linear combination $\mathbf{w}^T\mathbf{x}$. That is,

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

$f()$ is essentially the sign function

$$f(i) = \begin{cases} -1 & \text{if } i < 0 \\ 1 & \text{if } i \geq 0 \end{cases}$$

The resulting model is a particular generalized linear model. A special case is the one when ϕ is the identity, that is $y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x})$.

By the definition of the model, $y(\mathbf{x})$ can only be ± 1: we denote $y(\mathbf{x}) = 1$ as $\mathbf{x} \in C_1$ and $y(\mathbf{x}) = -1$ as $\mathbf{x} \in C_2$.

To each element $\mathbf{x}_i$ in the training set, a target value is then associated $t_i \in \{-1, 1\}$.

A natural definition of the cost function would be the number of misclassified elements in the training set. This would result into a piecewise constant function and gradient optimization could not be applied (we would have zero gradient almost everywhere).

A better choice is using a piecewise linear function as cost function

We would like to find a vector of parameters $\mathbf{w}$ such that, for any $\mathbf{x}_i$, $\mathbf{w}^T\mathbf{x}_i > 0$ if $\mathbf{x}_i \in C_1$ and $\mathbf{w}^T\mathbf{x}_i < 0$ if $\mathbf{x}_i \in C_2$: in short, $\mathbf{w}^T\mathbf{x}_i t_i > 0$.

Each element $\mathbf{x}_i$ provides a contribution to the cost function as follows

1. 0 if $\mathbf{x}_i$ is classified correctly by the model
2. $-\mathbf{w}^T\mathbf{x}_i t_i > 0$ if $\mathbf{x}_i$ is misclassified

Let $\mathcal{M}$ be the set of misclassified elements. Then the cost is

$$E_p(\mathbf{w}) = -\sum_{\mathbf{x}_i \in \mathcal{M}} t_i \mathbf{x}_i^T \mathbf{w}$$

The contribution of $\mathbf{x}_i$ to the cost is 0 if $\mathbf{x}_i \notin \mathcal{M}$ and it is a linear function of $\mathbf{w}$ otherwise

The minimum of $E_p(\mathbf{w})$ can be found through gradient descent

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \frac{\partial E_p(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(k)}}$$

the gradient of the cost function wrt to $\mathbf{w}$ is

$$\frac{\partial E_p(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{\mathbf{x}_i \in \mathcal{M}} \mathbf{x}_i t_i$$

Then gradient descent can be expressed as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \sum_{\mathbf{x}_i \in \mathcal{M}_k} \mathbf{x}_i t_i$$

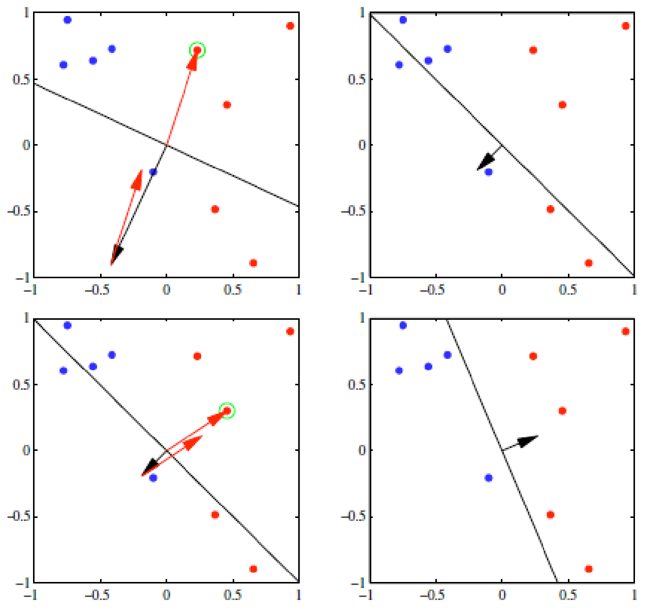
where $\mathcal{M}_k$ denotes the set of points misclassified by the model with parameter $\mathbf{w}^{(k)}$

Online (or stochastic gradient descent): at each step, only the gradient wrt a single item is considered

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \mathbf{x}_i t_i$$

where $\mathbf{x}_i \in \mathcal{M}_k$ and the *scale factor* $\eta > 0$ controls the impact of a badly classified item on the cost function

The method works by circularly iterating on all elements and applying the above formula.



In black, decision boundary and corresponding parameter vector $\mathbf{w}$; in red misclassified item vector $\mathbf{x}_i$, added by the algorithm to the parameter vector as $\eta \mathbf{x}_i$

At each step, if $\mathbf{x}_i$ is well classified then $\mathbf{w}^{(k)}$ is unchanged; else, its contribution to the cost is modified as follows

$$\begin{aligned} -\mathbf{x}_i^T \mathbf{w}^{(k+1)} t_i &= -\mathbf{x}_i^T \mathbf{w}^{(k)} t_i - \eta (\mathbf{x}_i t_i)^T \mathbf{x}_i t_i \\ &= -\mathbf{x}_i^T \mathbf{w}^{(k)} t_i - \eta ||\mathbf{x}_i||^2 \\ &< -\mathbf{x}_i^T \mathbf{w}^{(k)} t_i \end{aligned}$$

This contribution is decreasing, however this does not guarantee the convergence of the method, since the cost function could increase due to some other element becoming misclassified if $\mathbf{w}^{(k+1)}$ is used

It is possible to prove that, in the case the classes are linearly separable, the algorithm converges to the correct solution in a finite number of steps.

Let $\hat{\mathbf{w}}$ be a solution (that is, it discriminates C_1 and C_2): if $\mathbf{x}_{k+1}$ is the element considered at iteration $(k+1)$ and it is misclassified, then

$$\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}} = (\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}) + \eta\mathbf{x}_{k+1}t_{k+1}$$

where $\alpha > 0$ is a suitable constant