Machine learning and artificial intelligence

Notes from the a.y. 2021/2022 course held by Prof. Vittorio Murino

Author: Lorenzo Busellato

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Contents

1 Bayes' theory

The fundamental statistical approach to the problem of pattern recognition is Bayes' theory. As an initial assumption we consider the decision problem in probabilistic terms, where all relevant probabilities are known. The goal is to determine the different **decision rules** using the probabilities and the associated costs.

Example 1.1
*Let ω be the **state of nature** to be described probabilistically. Consider two **classes** ω_1 and ω_2, for which we know the **prior probability**:*

$$P(\omega = \omega_1) = 0.7 \qquad P(\omega = \omega_2) = 0.3$$

In this case the decision rule would be to choose ω_1 if $P(\omega_1) > P(\omega_2)$, otherwise choose ω_2.

The previous example can be improved by considering the measurement x of a random variable dependent on ω_j, with which we can define the **likelihood** or **class-conditional probability density function (PDF)**:

$$p(x \mid \omega_j)_{j=1,2}$$

which describes the probability of obtaining the measurement x knowing that the state of nature is ω_j. In other words if we fix x, the higher the likelihood the higher the probability that ω_j is the right state.
Assuming we know the prior $P(\omega_j)$ and the likelihood $p(x \mid \omega_j)$, the decision of the state of nature becomes:

$$p(\omega_j, x) = P(\omega_j \mid x)p(x) = p(x \mid \omega_j)P(\omega_j) \implies P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)} \propto p(x \mid \omega_j)P(\omega_j)$$

which is **Bayes' formula**, which defines the **posterior** probability $P(\omega_j \mid x)$ and the **evidence**:

$$p(x) = \sum_{j=1}^{J} p(x \mid \omega_j)P(\omega_j)$$

The posterior is the probability that the state of nature is ω_j given the observation of the measurement x. The most important part is the product between the likelihood and the prior, because the evidence is simply a scale factor that ensures that:

$$\sum_j P(\omega_j \mid x) = 1$$

We can now define the **Bayes' decision rule**, for which ω_1 is chosen if $P(\omega_1 \mid x) > P(\omega_2 \mid x)$, otherwise ω_2 is chosen.
The Bayes' decision rule is the one that minimizes the error in the decision. We define the **probability of error**:

$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) & \text{if we decide } \omega_2 \\ P(\omega_2 \mid x) & \text{if we decide } \omega_1 \end{cases}$$

To minimize the error we actually want to minimize the average probability of error over all possible observations, i.e.:

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)p(x)dx$$

That is to say, the error is minimized if for every observation x we take the smallest possible $P(error \mid x)$ (recall that $p(x)$ is irrelevant as it is only a scale factor).
Therefore in a two class decision problem the probability of error is:

$$P(error \mid x) = min[P(\omega_1 \mid x), P(\omega_2 \mid x)]$$

Which is exactly the Bayes' decision rule, which then is the one that minimizes the error. Since the evidence is irrelevant to the decision itself, we can eliminate it from the decision rule obtaining the **rule of equivalent decision**, for which we choose ω_1 if $p(x \mid \omega_1)P(\omega_1) > p(x \mid \omega_2)P(\omega_2)$, otherwise choose ω_2.
Bayes' decision rule can be extended by using:

- A **feature space**, or a set of multiple types of observations:

$$\mathbf{x} = \{x_1, x_2, \ldots, x_d\} \in R^d \quad \text{with } R^d \text{ feature space}$$

- A set of c **categories**, i.e. multiple states of nature:

$$\omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$$

- A set of a **actions** in addition to the choice of states of nature:

$$\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$$

- A **cost function** more general than the probability of error, that describes the cost or **loss** of the action α_i when the state is ω_j:

$$\lambda(\alpha_i \mid \omega_j)$$

Suppose we observe a particular $\mathbf{x}$, and we decide to carry out action α_i, with an associated loss $\lambda(\alpha_i, \mid \omega_j)$. Due to the indeterminacy of ω_j, the expected loss or **conditional risk** associated to the decision will be:

$$R(\alpha_i \mid \mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid \mathbf{x})$$

In this case Bayes' decision theory chooses to carry out the action that minimizes the conditional risk, by defining a **decision function**:

$$\alpha(\mathbf{x}) \rightarrow \alpha_i, \alpha_i \in \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$$

such that $R(\alpha_i \mid \mathbf{x})$ is minimal. Such a function requires the definition of the **overall risk**, i.e. the expected loss given a decision rule. Given the conditional risk $R(\alpha_i \mid \mathbf{x})$ associated to the action, the overall risk is:

$$R = \int R(\alpha(\mathbf{x}) \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

As we did before with the probability of error, the overall risk allows us to improve the decision rule, by defining the **Bayes' extended decision rule**:

1. Calculate:
$$R(\alpha_i \mid \mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid \mathbf{x})$$

2. Choose the action:
$$i^* = \underset{i}{argmin}\ R(\alpha_i \mid \mathbf{x})$$

That is to say choose for each $\mathbf{x}$ the action $alpha(\mathbf{x})$ that minimizes the overall risk. The resulting minimum overall risk R^* is called **Bayes' risk**.

1.1 Two-category classification problems

Consider Bayes' rule applied to binary classification problems. The conditional risk becomes:

$$R(\alpha_1 \mid \mathbf{x}) = \lambda_{11} P(\omega_1 \mid \mathbf{x}) + \lambda_{12} P(\omega_2 \mid \mathbf{x})$$
$$R(\alpha_2 \mid \mathbf{x}) = \lambda_{21} P(\omega_1 \mid \mathbf{x}) + \lambda_{22} P(\omega_2 \mid \mathbf{x})$$

The minimum risk decision rule can be expresses in multiple, equivalent forms. The fundamental form is to choose ω_1 if $R(\alpha_1 \mid \mathbf{x}) < R(\alpha_2 \mid \mathbf{x})$ or, in terms of posteriors, choose ω_1 if:

$$(\lambda_{21} - \lambda_{11}) P(\omega_1 \mid \mathbf{x}) > (\lambda_{12} - \lambda_{22}) P(\omega_2 \mid \mathbf{x})$$

Intuitively the loss for a wrong decision $(\lambda_{ij}, i \neq j)$ is greater than the loss for a right decision $(\lambda_{ij}, i = j)$, so:

$$(\lambda_{21} - \lambda_{11}) > 0 \wedge (\lambda_{12} - \lambda_{22}) > 0$$

So the decision is determined by the most likely state of nature, positively scaled by the difference between the losses.

Another equivalent decision rule form that explicitly considers the prior and conditional densities is obtained by applying Bayes:

$$(\lambda_{21} - \lambda_{11}) p(\mathbf{x} \mid \omega_1) P(\omega_1) > (\lambda_{12} - \lambda_{22}) p(\mathbf{x} \mid \omega_2) P(\omega_2)$$

Or equivalently:

$$\frac{p(\mathbf{x} \mid \omega_1)}{p(\mathbf{x} \mid \omega_2)} > \frac{(\lambda_{12} - \lambda_{22})P(\omega_2)}{(\lambda_{21} - \lambda_{11})P(\omega_1)}$$

This form for the decision rule focuses on the dependence of the observation on probability densities. By calculating a **likelihood ratio** we make the choice independent from the observation $\mathbf{x}$, so Bayes' rule becomes to choose ω_1 if the likelihood ratio exceeds some threshold.

In classification problems each state is associated with one of the c classes ω_j, and action α_i means that the correct state is ω_i. The loss function in this case is the **symmetric loss**:

$$\lambda(\alpha_i \mid \omega_j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$$

The risk associated to this loss function is the average probability of error:

$$R(\alpha_i \mid \mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid \mathbf{x}) = \sum_{j \neq i}^{c} P(\omega_j \mid \mathbf{x}) = 1 - P(\omega_i \mid \mathbf{x})$$

where $P(\omega_i \mid \mathbf{x})$ is the probability that action α_i is correct. To minimize the total risk we must choose the i that maximizes the posterior $P(\omega_i \mid \mathbf{x})$, condition which links back to the fundamental decision rule (to decide ω_1 if $P(\omega_i \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x}), \forall i \neq j$).

1.2 Decision theory

The problem can be split into an **inference** phase, in which data is used to train a model $p(\omega_k \mid \mathbf{x})$, and a subsequent **decision** phase, in which the posterior is used to make the choice of class. Alternatively one can train a function that maps directly $\mathbf{x}$ to the space of decisions.

The main approaches to solve the decision problem are:

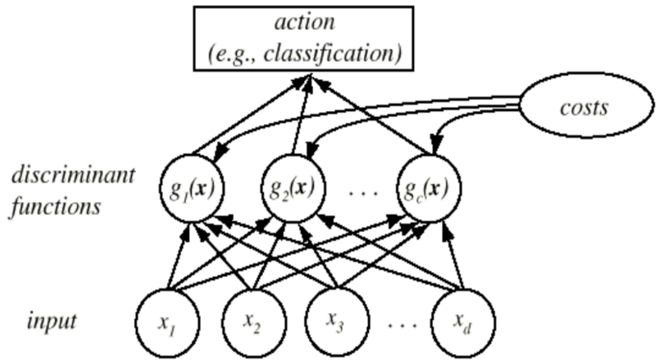
- **Generative models**, that solve first the inference problem to determine the PDF and the priors for each class, to then apply Bayes to determine the posterior and determine the class with decision theory.
- **Discriminatory models**, that solve the inference problem determining directly the posterior and then determine the class with decision theory.
- **Discriminating functions**, i.e. functions $f(\mathbf{x})$ that directly map $\mathbf{x}$) to a class label.

Generative models are more complex, requiring good training sets, but have the advantage of manipulating all variables. When the problem is classification, discriminative methods are more efficient, since sometimes the PDF have a complex profile but don't affect the posterior. The best option would be to use discriminating functions, since they directly give the separation surface between the classes.

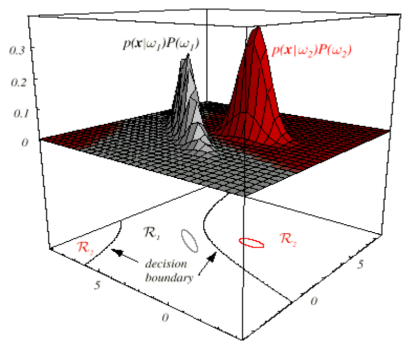
Discriminating functions are a set of functions $g_i(\mathbf{x}), i = 1, \ldots, c$ that let a classifier assign feature $\mathbf{x}$ to class ω_i if:

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \; \forall j \neq i$$

Such a classifier is comparable to a network that calculates c discriminating functions and chooses the one that discriminates the most:



There are many equivalent discriminating functions, and their effect is the division of the feature space into c **separation surfaces** $R_1, \dots, R_c$:



The regions are separated by **decision boundaries** that are defined by the maximum discriminating functions. In the two-category classification case we have two discriminating functions g_1 and g_2 for which we assign $\mathbf{x}$ to ω_1 if $g_1 > g_2$, that is to say $g_1 - g_2 > 0$.

We now define the discriminating functions based on the posteriors:

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}) = P(\omega_1 \mid \mathbf{x}) - P(\omega_2 \mid \mathbf{x}) = \ln(P(\omega_1 \mid \mathbf{x}) - P(\omega_2 \mid \mathbf{x})) = \ln\left(\frac{p(\mathbf{x} \mid \omega_1)}{p(\mathbf{x} \mid \omega_2)}\right) + \ln\left(\frac{P(\omega_1)}{P(\omega_2)}\right)$$

Note that the natural logarithm is a strictly increasing function, and so does not change the relation between g_1 and g_2.

To define the discriminating functions we then need to define the conditional densities $p(\mathbf{x} \mid \omega_i)$. One of the most important densities is the **Gaussian multivariate** or **normal density**. It is important because it is analytically manageable and it models a range of both theoretical and practical problems (under the Central Limit theorem under various conditions the distribution of the sum of d independent random variables tends to the normal distribution).

The **univariate normal density** is uniquely defined by the **mean** μ and the **variance** σ^2:

$$N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \qquad \begin{aligned} \mu = E[x] &= \int_{-\infty}^{\infty} xp(x)dx \\ \sigma^2 = E[(x-\mu)^2] &= \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx \end{aligned}$$

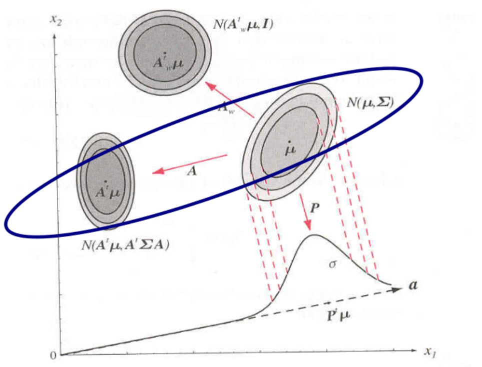
In d dimensions we define the **multivariate normal density** with the **mean** μ over the d components and the **covariance matrix** $\Sigma \in \mathbb{R}^{d \times d}$:

$$N(\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \qquad \begin{aligned} \mu = E[x] &= \int_{-\infty}^{\infty} xp(x)dx \\ \Sigma = E[(x-\mu)(x-\mu)^T] &= \int (x-\mu)(x-\mu)^T p(x)dx \to \sigma_{ij} = E[(x_i-\mu_i)(x_j-\mu_j)] \end{aligned}$$

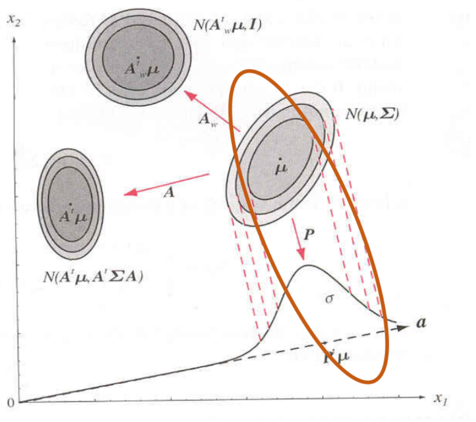
The covariance matrix is symmetric and semi-defined positive. Each element σ_{ij} represents the covariance between x_i and x_j (or the variance of x_i if $i = j$). If x_i and x_j are statistically independent then $\sigma_{ij} = 0$.

A property of the normal density is that if $p(\mathbf{x}) \approx N(\mu, \Sigma)$, A is a $d \times k$ matrix and $\mathbf{y} = A^T \mathbf{x}$, then:

$$p(\mathbf{y}) \approx N(A^T \mu, A^T \Sigma A)$$



A special case is $k = 1$, for which A becomes a vector a and y becomes a scalar that represents the projection of $\mathbf{x}$ on a line in the direction defined by a. Then the variance of $\mathbf{x}$ on a is $a^t\Sigma a$.

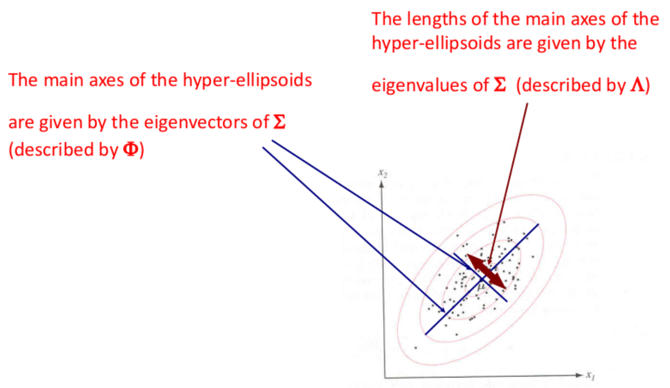


Finally we define the **whitening transform**:

$$A_w = \Phi\Lambda^{-\frac{1}{2}}$$

where Φ is the matrix of orthonormal eigenvectors of Σ in column and Λ is the diagonal matrix of the corresponding eigenvalues.

The transformation applied to the feature space coordinates produces a covariance matrix equal to the identity matrix. This means that after whitening all variables become independent from each other, and their variance becomes 1.


The lengths of the main axes of the hyper-ellipsoids are given by the eigenvalues of $\mathbf{\Sigma}$ (described by $\mathbf{\Lambda}$)
The main axes of the hyper-ellipsoids are given by the eigenvectors of $\mathbf{\Sigma}$ (described by $\mathbf{\Phi}$)
x_2
x_1

Where hyper-ellipsoids are the locus of points for which the distance of $\mathbf{x}$ from μ (**Mahalanobis distance**):

$$r^2 = (\mathbf{x} - \mu)^T\Sigma^{-1}(\mathbf{x} - \mu)$$

is constant.

Back to discriminating functions, we have:

$$g(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_1) + \ln P(\omega_1) = -\frac{1}{2}(\mathbf{x} - \mu)^T\Sigma_1^{-1}(\mathbf{x} - \mu) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

Depending on Σ then we can simplify the above formula.

When $\Sigma_i = \sigma^2\mathbb{I}$ the features are statistically independent ($\sigma_{ij} = 0, i \neq j$) and each class has the same variance. We have:

$$g_i(x) = -\frac{\|\mathbf{x} - \mu_i\|^2}{2\sigma^2} + \ln P(\omega_i) = -\frac{1}{2\sigma^2}[\mathbf{x}^T\mathbf{x} - 2\mu_i^T\mathbf{x} + \mu_i^T\mu_i] + \ln P(\omega_i)$$

where $\mathbf{x}^T\mathbf{x}$ is equal for every $\mathbf{x}$, and can therefore be ignored:

$$g_i(\mathbf{x}) = w_i^T\mathbf{x} + w_{i0}$$

where:

$$w_i = \frac{1}{\sigma^2}\mu_i \quad \text{and } w_{i0} = -\frac{1}{2\sigma^2}\mu_i^T\mu_i + \ln P(\omega_i)$$

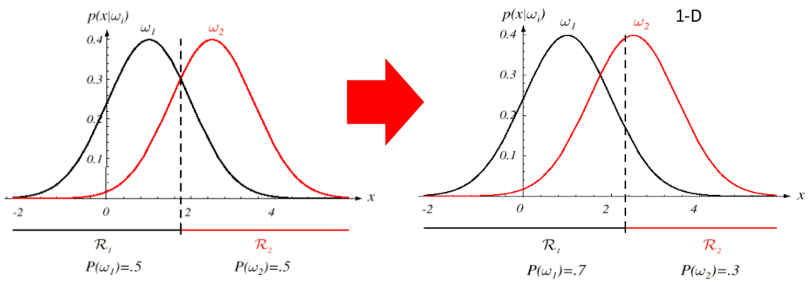
The discriminant functions defined in this way are called **linear discriminant functions**. The decision boundaries are given by $g_i(\mathbf{x}) = g_j(\mathbf{x})$ for the two classes with the highest posterior probability. In this case we have:

$$w^T(\mathbf{x} - \mathbf{x}_0) = 0$$

where:

$$w = \mu_i - \mu_j \quad \text{and } \mathbf{x}_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)}(\mu_i - \mu_j)$$

Linear discriminant functions define a hyper-plane passing through $\mathbf{x}_0$ and orthogonal to w. By looking at the previous formula we can see that, for the same variance, the classification result is determined by the largest of the priors:



Note that if the priors $P(\omega_i), i = 1, \ldots, c$ are equal then the term with the logarithm is null and the classifier reduces to a **minimum distance classifier**. In practice this means that the classifier assigns $\mathbf{x}$ to the class whose mean μ is closer.

Another notable case is when the covariance matrices for all classes are equal, but arbitrary. In this case the function can be simplified to:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma_1^{-1}(\mathbf{x} - \mu) + \ln P(\omega_i)$$

Which can be expressed in the form:

$$g_i(\mathbf{x}) = w_i^T \mathbf{x} + w_{i0}$$

where:

$$w_i = \Sigma^{-1}\mu_i \quad \text{and } w_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \ln P(\omega_i)$$

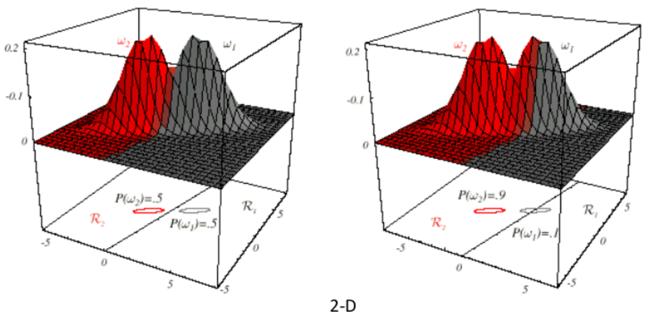
Since the discriminants are still linear, the decision boundaries are still hyper-planes. If the decision regions R_i and R_j are contiguous, the boundary between them is:

$$w^T(\mathbf{x} - \mathbf{x}_0) = 0$$

where:

$$w = \Sigma^{-1}(\mu_i - \mu_j) \quad \text{and } \mathbf{x}_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right)}{(\mu_i - \mu_j)^T \Sigma^{-1}(\mu_i - \mu_j)}(\mu_i - \mu_j)$$

Since now w is no longer the vector that connects the means, the hyper-plane that divides the decision regions is no longer orthogonal to them. it still passes through $\mathbf{x}_0$. The considerations on the priors are the same as the previous case.



The last case is the generic one in which each class has an arbitrary covariance matrix Σ_i. The resulting discriminating functions are quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^T W_i \mathbf{x} + w_i^T \mathbf{x} + w_{i0}$$

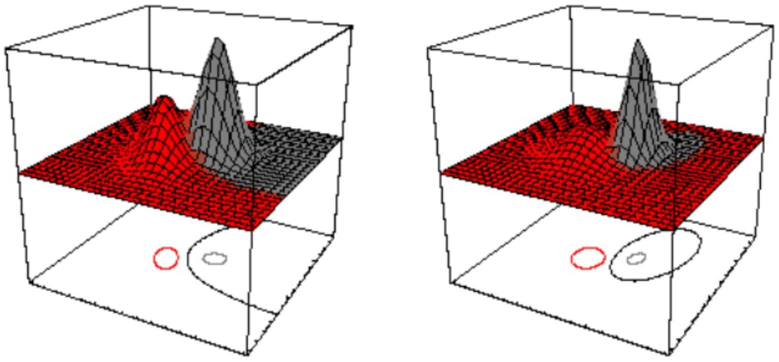
where:

$$W_i = -\frac{1}{2}\Sigma_i^{-1} \quad \text{and } w_i = \Sigma_i^{-1}\mu_i$$

and:

$$w_{i0} = -\frac{1}{2}\mu_i^T \Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

In the 2D case the decision surfaces are hyper-quadric, i.e. hyper-planes, hyper-spheres or hyper-paraboloids.



2 Parametric models

To create an optimal classifier, i.e. one that uses the Bayesian decision rule, one needs to know both the priors $P(\omega_i)$ and the class-conditional densities $p(\mathbf{x} \mid \omega_i)$. The performance of the classifier is strictly related to these quantities, which are almost never available beforehand.

More often only vague knowledge of the problem or some representative data patterns are available in order to obtain the classifier. Under these conditions estimating the priors is usually easy, while estimating the class-conditional densities is more complex.

Under the **supervised learning** assumption, we have a set of n training data, each pattern of which has an assigned label, or in other words we know which state ω_i corresponds to pattern k. The priors are then simply:

$$P(\omega_i) = \frac{n_i}{n}$$

where n_i is the number of samples under label ω_i.

The priors are not as useful as the conditional densities though. Supposing we have c data sets $D_1, D_2, \dots, D_c$ sampled according to the density $p(x \mid \omega_j)$, we want to estimate the parameters that define $p(x \mid \omega_j)$. To simplify the problem we assume that the samples of set D_i do not give information on the parameters of $p(x \mid \omega_j)$ if $i \neq j$.

There are two approaches for solving the problem: maximum likelihood estimation and the Bayesian approach.

2.1 Maximum likelihood approach

In the ML approach the parameters are fixed but unknown values, and the best estimation for their value is the one that maximizes the probability of obtaining the samples actually observed.

Since the patterns of set D are independent and identically distributed under our assumptions, we have:

$$p(D \mid \boldsymbol{\theta}) = \prod_{k+1}^{n} p(x_k \mid \boldsymbol{\theta})$$

which is the **likelihood** of the vector $\boldsymbol{\theta}$, i.e. a measure of how well such a vector predicts the training data. The problem is therefore to find the optimal $\hat{\boldsymbol{\theta}}$ that maximizes the likelihood.

Let p be the number of parameters to be set and let $\boldsymbol{\theta}$ denote the p-component vector $\boldsymbol{\theta} = \left[\theta_1, \dots, \theta_p\right]^T$ and let:

$$\nabla\boldsymbol{\theta} = \begin{bmatrix} \frac{\partial}{\partial\theta_1} \\ \vdots \\ \frac{\partial}{\partial\theta_p} \end{bmatrix}$$

be the gradient operator.

For analytical purposes it's useful to work with the logarithm of the likelihood:

$$l(\boldsymbol{\theta}) = \ln p(D \mid \boldsymbol{\theta}) = \sum_{k=1}^{n} \ln p(x_l \mid \boldsymbol{\theta})$$

which is the **log-likelihood** function.

The goal is therefore to determine:

$$\hat{\boldsymbol{\theta}} = \underset{\theta}{argmax}\ l(\boldsymbol{\theta})$$

Therefore the maximum is obtained as:

$$\nabla_\theta l(\boldsymbol{\theta}) = \sum_{k=1}^{n} p(x_k \mid \boldsymbol{\theta}) = 0$$

Once the set of parameters is estimated, it is necessary to check if the solution is actually a global maximum, as well as check what happens on the boundaries of the parameter space.

Suppose the samples are drawn from a multivariate normal population with mean μ and covariance matrix Σ, and suppose only μ is known.

Consider a sample point x_k for which:

$$\ln p(x_k \mid \mu) = -\frac{1}{2}\ln\big((2\pi)^d|\Sigma|\big) - \frac{1}{2}(x_k-\mu)^T\Sigma^{-1}(x_k-\mu) \implies \nabla_\mu \ln p(x_k \mid \mu) = \Sigma^{-1}(x_k-\mu)$$

Therefore the maximum likelihood estimate for μ is:

$$\Sigma^{-1}(x_k - \hat{\mu}) = 0 \implies \hat{\mu} = \frac{1}{n}\sum_{k+1}^{n} x_k$$

which is the arithmetic average of the training samples.

In a more general case neither μ nor Σ are known. For instance in the univariate case $\boldsymbol{\theta} = (\theta_1, \theta_2) = (\mu, \sigma^2)$ the log-likelihood for a single point is:

$$\ln p(x_k \mid \boldsymbol{\theta}) = -\frac{1}{2}\ln(2\pi\theta_2) - \frac{1}{2\theta_2}(x_k - \theta_1)^2$$

whose derivative is:

$$\nabla_\theta l = \nabla_\theta \ln p(x_k \mid \boldsymbol{\theta}) = \begin{bmatrix} \frac{1}{\theta_2}(x_k - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x_k-\theta_1)^2}{2\theta_2^2} \end{bmatrix}$$

By imposing it to zero we get the ML estimates for the parameters:

$$\sum_{k=1}^{n} \frac{1}{\theta_2}(x_k - \hat{\theta}_1) = 0 \qquad -\sum_{k=1}^{n} \frac{1}{\hat{\theta}_2} + \sum_{k=1}^{n} \frac{(x_k - \hat{\theta}_1)^2}{\hat{\theta}_2^2} = 0$$

So, substituting $\hat{\theta}_1 = \mu$ and $\hat{\theta}_2 = \sigma^2$:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})^2$$

In the multivariate case we similarly conclude that:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k \qquad \hat{\boldsymbol{\Sigma}}^2 = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T$$

The ML estimate for the variance is biased, so it is not equal to the true variance.

Besides the Gaussian density there are other densities to which we can apply ML, such as:

- Exponential distribution:
$$p(x \mid \theta) = \begin{cases} \theta e^{-\theta x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
- Uniform distribution:
$$p(x \mid \theta) = \begin{cases} \frac{1}{\theta} & \text{if } 0 \leq x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
- Multivariate Bernoulli

The problem with ML estimation is that the error strongly depends on the accuracy of the selected parametric model, and also that all training data must be supplied at once (it is not an on-line method).

2.2 Bayesian approach

Unlike ML, in the Bayesian approach we assume that $\boldsymbol{\theta}$ is a random variable. In this case with the use of a training data set D we convert a prior distribution $p(\boldsymbol{\theta})$ into a posterior probability density $p(\mid D)$.

Given the training data set D we can apply Bayes' formula and obtain:

$$P(\omega_i \mid \mathbf{x}, D) = \frac{p(\mathbf{x} \mid \omega_i, D)P(\omega_i \mid D)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D)P(\omega_j \mid D)}$$

Since we are in the supervised learning case, it is reasonable to assume that the training set D can be split into c subsets $D_1, \ldots, D_c$ with the samples in D_i belonging to ω_i. We also assume that the samples belonging to D_i do not influence $p(\mathbf{x} \mid \omega_j, D)$ if $i \neq j$.

Under these assumptions we can treat each class separately:

$$P(\omega_i \mid \mathbf{x}, D_i) = \frac{p(\mathbf{x} \mid \omega_i, D_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D_i)P(\omega_j)}$$

We can furthermore reduce the problem to c different instances of the same problem, i.e. the determination of $p(\mathbf{x}, D)$.

The Bayesian learning process estimates a model only implicitly, so we don't obtain actual values for $\boldsymbol{\theta}$ but distributions for them, based on the training set.

We have that:

$$p(\mathbf{x} \mid D) = \int p(\mathbf{x}, \boldsymbol{\theta} \mid D) d\boldsymbol{\theta} = \int p(\mathbf{x} \mid \boldsymbol{\theta}, D) p(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta}$$

Since by hypothesis the selection of $\mathbf{x}$ is independent from the training samples in D, given $\boldsymbol{\theta}$ we have:

$$p(\mathbf{x} \mid D) = \int p(\mathbf{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta}$$

That is to say that the distribution of $p(\mathbf{x})$ is completely known once we know $\boldsymbol{\theta}$. The equation links the class-conditional density $p(\mathbf{x} \mid D)$ we are looking for with the posterior density $p(\boldsymbol{\theta} \mid D)$ through the unknown vector $\boldsymbol{\theta}$. If $p(\boldsymbol{\theta} \mid D)$ peaks sharply around some value $\hat{\boldsymbol{\theta}}$ then we obtain an estimation for the most likely parameter vector:

$$p(\mathbf{x} \mid D) \approx p(\mathbf{x}, \hat{\boldsymbol{\theta}})$$

In the univariate Gaussian case for instance we have:

$$p(\mathbf{x} \mid \boldsymbol{\theta}) \equiv p(\mathbf{x} \mid \mu) \approx N(\mu, \sigma^2)$$

The prior knowledge about μ can be expressed as a prior density with known mean and variance:

$$p(\mu) \approx N(\mu_0, \sigma_0^2)$$

μ_0 represents our best a-priori guess for μ, while σ_0^2 represents our uncertainty on the guess. Suppoisng we have n training samples $D = \{x_1, \ldots, x_n\}$ then we calculate the **reproduced density**:

$$p(\mu \mid D) = \frac{p(D \mid \mu)p(\mu)}{\int p(D \mid \mu)p(\mu)d\mu} = \alpha \prod_{k=1}^{n} p(x_k \mid \mu)p(\mu)$$

where α is a normalization factor that depends only on D. This equation shows how we can connect the prior density $p(\mu)$ to the posterior density $p(\mu \mid D)$, or in other words how the training data affects our guess on μ.

What we obtain is the **Bayesian Learning phenomenon**, that tells us that since the prior is normally distributed, $p(\mu \mid D)$ also is, and the more training data samples we consider the more the distribution approaches a Dirac impulse:

$$p(\mu \mid D) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(\mu - \mu_n)^2}{2\sigma_n^2}}$$

where:

$$\mu_n = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \left(\frac{1}{n} \sum_{k=1}^{n} x_k \right) + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0 \qquad \sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

μ_0 represents our best guess for μ after observing n samples, while σ_n^2 measures the uncertainty about the guess.

Now that we have the posterior density for the mean, all that remains to do is to compute the class-conditional density for $p(x, D)$:

$$p(x \mid D) = \int p(x \mid \mu) p(\mu \mid D) d\mu = \int \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(\mu-\mu_n)^2}{2\sigma_n^2}} d\mu = \frac{1}{2\pi\sigma\sigma_n} e^{-\frac{1}{2}\frac{(x-\mu_n)^2}{\sigma^2+\sigma_n^2}} f(\sigma, \sigma_n)$$

where:

$$f(\sigma, \sigma_n) = \int e^{-\frac{1}{2}\frac{\sigma^2+\sigma_n^2}{\sigma^2\sigma_n^2}\left(\mu - \frac{\sigma_n^2 x + \sigma^2 \mu_n}{\sigma^2+\sigma_n^2}\right)^2}$$

We can observe that:

$$p(x \mid D) \approx N(\mu_n, \sigma^2 + \sigma_n^2)$$

That is to say the conditional mean is treated as the true mean, while the known variance is proportional to the current degree of uncertainty.

3 Non-parametric models

In many recognition problems the assumption that the form of the probability densities is known is not applicable. In particular in the description of parametric models we assumed that the densities are unimodal, but in reality many problems have multimodal densities.

The problem is to estimate the quantity:

$$p(\mathbf{x} \mid \omega_i) \equiv \hat{p}_i(\mathbf{x})$$

in order to apply the theoretical optimal Bayesian classifier.

3.1 Potential functions

The idea is to establish an analogy between samples and the notion of electric charge. By placing an electric charge at each point associated to a sample we can appropriately define a potential function associated to it, which then constitutes the discriminating function for the recognition problem.

Let $\gamma(\mathbf{x}, \mathbf{y}_j)$ be the potential function associated to a generic sample $\mathbf{y}_j$ belonging to class i. Then we can write:

$$\hat{p}_i(\mathbf{x}) = \frac{1}{N_i} \sum_{k=1}^{N_i} \gamma(\mathbf{x}, \mathbf{y}_j)$$

where N_i is the number of samples of class i.

γ is a good approximation if:

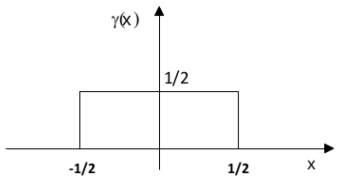
- It is strictly positive, i.e. $\gamma(\mathbf{x}, \mathbf{y}_j) \geq 0$.
- It has a maximum in $\mathbf{y}_k$, i.e. $\underset{\mathbf{x}}{argmax}\, \gamma(\mathbf{x}, \mathbf{y}_k) = \mathbf{y}_k$.
- p does not vary abruptly or have discontinuities, i.e. $\gamma(\mathbf{x}, \mathbf{y}_1) \approx \gamma(\mathbf{x}, \mathbf{y}_2)$ if $|\mathbf{y}_2 - \mathbf{y_1}| < \varepsilon$.
- It is continuous.
- It is normalized, i.e. $\int\limits_{-\infty}^{\infty} \gamma(\mathbf{x}, \mathbf{y}_k) d\mathbf{x} = 1$.
- $\gamma(\mathbf{x}, \mathbf{y}_k) \approx 0$ if $\mathbf{x}$ is far away from $\mathbf{y}_k$.

There are many possible functions that satisfy these constraints. We will refer to a $\gamma(\mathbf{z}, \mathbf{y})$ function depending only on a variable $\mathbf{x} = |\mathbf{z} - \mathbf{y}|$.

Some possible potential functions are:

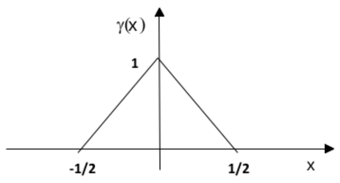
1. **Rectangle**:

$$\gamma(\mathbf{x}) = \begin{cases} 0.5 & |\mathbf{x}| \leq 1 \\ 0 & |\mathbf{x}| > 1 \end{cases}$$



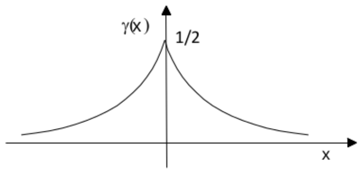
2. **Triangle**:

$$\gamma(\mathbf{x}) = \begin{cases} 1 - |\mathbf{x}| & |\mathbf{x}| \leq 1 \\ 0 & |\mathbf{x}| > 1 \end{cases}$$



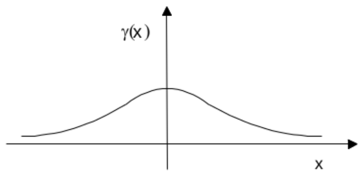
3. **Gaussian**:

$$\gamma(\mathbf{x}) = (2\pi)^{-\frac{1}{2}} e^{-\frac{\mathbf{x}^2}{2}}$$



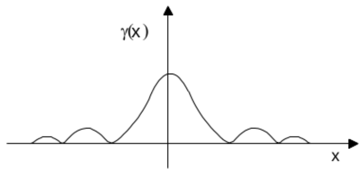
4. **Decreasing exponential**:

$$\gamma(\mathbf{x}) = \frac{1}{2} e^{|\mathbf{x}^2|}$$



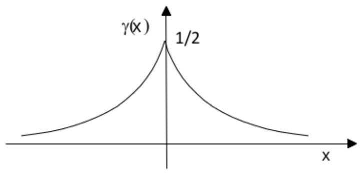
5. **Cauchy distribution**:

$$\gamma(\mathbf{x}) = (\pi(1+\mathbf{x}^2))^{-1}$$



6. **Squared sinc function**:

$$\gamma(\mathbf{x}) = (2\pi)^{-1} \left(\frac{\sin\left(\frac{\mathbf{x}}{2}\right)}{\frac{\mathbf{x}}{2}} \right)^2$$



3.2 Conditional density estimation

The probability that a vector $\mathbf{x}$ is in a region $\mathcal{R}$ is:

$$P = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}'$$

P is a smoothed version of the density $p(\mathbf{x})$, and we can then estimate a smoothed value of p by estimating P. Considering a set of n samples extracted according to $p(\mathbf{x})$ the probability that k points out of n are contained in $\mathcal{R}$ is given by the **binomial law**:

$$P_k = \binom{n}{k} P^k (1-P)^{n-k}$$

And the expected value of k is:

$$E[k] = nP$$

The estimated maximum likelihood of P is given by:

$$\underset{\theta}{max}\ (P_k \mid \theta) \rightarrow \hat{\theta} = \frac{k}{n} \approx P$$

Therefore we approximate P with the ratio $\frac{k}{n}$. Assuming that $p(\mathbf{x})$ is continuous and that the region $\mathcal{R}$ is small enough as not to contain big variations of p, we can write:

$$P = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}' \approx p(\mathbf{x})V$$

where $\mathbf{x} \in \mathcal{R}$ and V is the volume enclosed by $\mathcal{R}$. Substituting the ration $\frac{k}{n}$ into the previous equation we finally obtain:

$$p(\mathbf{x}) \approx \frac{\frac{k}{n}}{V}$$

which is a value averaged in a region around the true $p(\mathbf{x})$ As such, the true $p(\mathbf{x})$ is obtained if V tends to zero. The case in which k is equal to zero yields:

$$\lim_{V\to 0, k=0} p(\mathbf{x}) = 0$$

which is not interesting because it is the case in which there are no samples in $\mathcal{R}$.
The case in which $k \neq 0$ yields:

$$\lim_{V\to 0, k\neq 0} p(\mathbf{x}) = \infty$$

which is not interesting since the estimation diverges. So in practice V does not need to be infinitesimal but just arbitrarily small.
To estimate the density of $\mathbf{x}$ we form a sequence of regions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ containing $\mathbf{x}$. Let V_i be the volume of $\mathcal{R}_i$, k_i the number of samples contained in $\mathcal{R}_i$ and $p_i(\mathbf{x})$ the i-th estimate for $p(\mathbf{x})$, then:

$$p_i(\mathbf{x}) = \frac{\frac{k_i}{n}}{V_i}$$

$p_i(\mathbf{x})$ converges to $p(\mathbf{x})$ if:

$$\lim_{i\to\infty} V_i = 0 \qquad \lim_{i\to\infty} k_i = \infty \qquad \lim_{i\to\infty} \frac{k_i}{n} = 0$$

Two common ways of satisfying these conditions are:

- **Parzen windows**, in which we take an initial region and shrink it by specifying its volume as a function of i, such as $V_i = \frac{1}{\sqrt{(i)}}$, and verify that:
$$p_i(\mathbf{x}) \xrightarrow[n\to\infty]{} p(\mathbf{x})$$
- **k-nearest neighbor**, in which we define k_i as a function of i, e.g. $k_i = \sqrt{i}$, and increase the volume V_i until it encloses k_i neighbors of $\mathbf{x}$.

3.3 Parzen windows

If the region $\mathcal{R}$ is a small hyper-cube centered on $\mathbf{x}$ and we want to obtain the number k of samples falling in it we can define the following window function:

$$\gamma(\mathbf{u}) = \begin{cases} 1 & |u_i| < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \ldots, D$$

γ is an example of a **kernel function**, in this instance called **Parzen window**.
Therefore:

$$k = \sum_{j=1}^{N} \gamma\left(\frac{\mathbf{x} - \mathbf{y}_j}{h}\right)$$

And so:

$$p(\mathbf{x}) \approx \frac{\frac{k}{N}}{V} = \frac{1}{N}\sum_{j=1}^{N} \frac{1}{h^D}\gamma\left(\frac{\mathbf{x} - \mathbf{y}_j}{h}\right) \quad V = h^D$$

There are problems in the boundaries of the hyper-cubes, which can have discontinuities. To solve this we can use smoother kernel functions. Furthermore the choice of the volume h^D is also important, because if it is too big the resulting estimate will be low resolution and if it is too small there will be a large statistical variability. Intuitively the greater N_i is, the better the estimate:

$$\hat{p}_i(\mathbf{x}) = \lim_{N_i \to \infty} \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{1}{h^D} \gamma\left(\frac{\mathbf{x} - \mathbf{y}_j}{h}\right)$$

To smooth out the estimation we can use **Parzen windows with Gaussian kernel** (or **Specht functions**):

$$\hat{p}_i(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}} N_i} \sum_{j=1}^{N_i} e^{-\frac{(\mathbf{x}-\mathbf{y}_j)^T(\mathbf{x}-\mathbf{y}_j)}{2\sigma^2}}$$

Here the variance σ^2 acts as a **smoothing parameter**, and we have that for $\sigma = 0$ the probability becomes a sum of Dirac impulses centered on each $\mathbf{y}_j$, while for $\sigma \to \infty$ $\hat{p}(\mathbf{x})$ becomes constant. The smoothing parameter is considered uniform in every direction, so before estimating the probability the features need to be normalized in order to force their variances to be equal. σ should furthermore be chosen in order to not have an excessive overlap between the potential functions centered in each sample.

A suggested estimation for σ is to calculate the average of the distances between L points closest to a generic $\mathbf{y}_i$:

$$\sigma = \frac{1}{L} \sum_{j=1}^{L} \|\mathbf{y_j} - \mathbf{y}_i\| = \frac{1}{L} \sum_{j=1}^{L} \sqrt{(\mathbf{y_j} - \mathbf{y}_i)^2}$$

A heuristic choice is $L \approx 0.05N$.

To find the discriminating function we substitute in the Taylor series expansion of the exponential around the mean value (zero):

$$\hat{p}_i(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}} N_i} e^{-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}} \sum_{j=1}^{N_i} e^{\frac{\mathbf{x}^T\mathbf{y}_j}{\sigma^2}} e^{-\frac{\|\mathbf{y}_j\|^2}{2\sigma^2}} \approx \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}} N_i} e^{-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}} \sum_{j=1}^{N_i} e^{-c_j} \sum_{h=0}^{r} (\mathbf{x}^T\mathbf{y}_j^k)^h \frac{1}{\sigma^{2h} h!}$$

The above expression applies for any class ω_i, therefore the discriminating function for class ω_k using Bayes is:

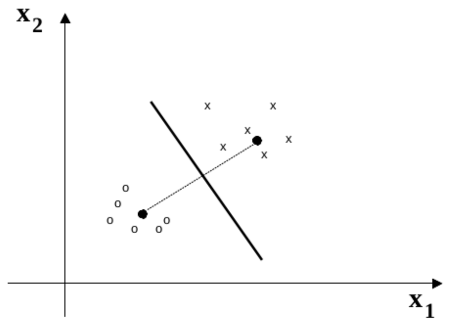
$$g_k(\mathbf{x}) = \hat{p}(x \mid \omega_k)\hat{p}(\omega_k) \approx \frac{\hat{p}(\omega_k)}{N_k} \sum_{j=1}^{N_k} e^{-c_j} \sum_{h=0}^{r} (\mathbf{x}^T\mathbf{y}_j)^h \frac{1}{\sigma^{2h} h!}$$

When σ is large enough, an approximation with a small r is sufficient. At the limit $\sigma \to \infty$ the function becomes linear, and the method becomes equivalent to the method of prototypes. In the opposite case $\sigma \to 0$ a large r is required and the discriminating function becomes a sum of Dirac delta impulses.

3.4 Method of prototypes

The method of prototypes is typically applied to **Minimum Distance Classifiers** (MDC). The "minimum distance" is just a metric, i.e. a map to $\mathbb{R}$ of the feature space, for example the Euclidean distance. It is supposed that the samples of a class tend to concentrate densely around a pattern representative of the class itself.

For instance considers the following situation:



the black dots are the **centers of gravity** m_1 and m_2 of the two clusters of samples associated to the two class. Let $D(\mathbf{x}, m_i)$ be a metric in the feature space.

The decision rule for a MDC is to choose class ω_k if $D(\mathbf{x}, m_k) = \underset{j}{argmin}\ D(\mathbf{x}, m_k), j = 1, \dots, M, j \neq k$.

So the prototypes method works as follows:

1. Determine the m_i center of gravity of class ω_i.
2. Determine the equation of the barycentre of two classes, $r_{ij}(\mathbf{x})$.
3. Determine the hyper-plane $g^*_{ij}(\mathbf{x})$ perpendicular to $r_{ij}(\mathbf{x})$.
4. The crossing point of the hyper-plane in the feature space is determined on the basis of the different probabilities of the two classes, the higher the probability for a class is, the more the discriminating function will be far from the barycentre of that class.
5. Repeat 1-4 for all class pairs.

The result is a set of $f_{ij}(\mathbf{x})$ discriminating function, which are all linear.

The method is simple and not memory-intensive, but the assumption that there is a single point that represents the entire class is very limiting.

3.5 k-nearest neighbors (k-NN) method

The **k-nearest neighbors method** is derived from the prototypes method, but instead of considering a single representative point while computing the metric, a set of variable points for a subset of classes is considered.

Given an unknown point x^* we consider $s_i \in \{s_1, \dots, s_n\} = S$ the nearest point to point x^* if:

$$D(x^*, s_i) = \underset{i}{argmin}\ D(x^*, s_l) \qquad l = 1, \dots, N$$

where $D()$ is a metric defined on the feature space.

Each point of set S can be treated as the prototype of a class, therefore the expression above is a 1-NN rule of classification that associates a sample x to the class j to which s_i belongs.

More generally a k-NN rule associates observation x with the class i that has the most elements among the nearest k. We call $U(x)$ the set of k points closest to x. For large k, i.e. $k \approx N$, the k-NN method becomes equivalent to the prototypes method. A typical choice for k is $\sqrt{N}$. Since we need to store all samples the method is appropriate whenever N is small.

Besides classification, k-NN can be used to estimate $p(\mathbf{x})$. Instead of fixing V as we did with kernels, we fix k and enlarge the radius of the sphere to include k samples. Given a class ω_j, the frequency of occurrence N_j of samples of class ω_j is measured in relation to the total number of samples:

$$P(\omega_j) \approx \hat{p}(\omega_j) = \frac{N_j}{N}$$

For instance, if we consider two classes ω_i and ω_j containing respectively N_i and N_j samples, with $N_i + N_j = N$, the local density estimation is:

$$\hat{p}(x \mid \omega_i) = \frac{1}{V}\frac{k_i}{N_i} \qquad \hat{p}(x \mid \omega_j) = \frac{1}{V}\frac{k_j}{N_j}$$

For Bayes' rule we choose $p(x \mid \omega_i)P(\omega_i) > p(x \mid \omega_j)P(\omega_j)$ and so:

$$\frac{1}{V}\frac{k_i}{N_i} > \frac{1}{V}\frac{k_j}{N_j} \implies k_i > k_j$$

The rule simply becomes to choose the class with the most neighboring samples that are associated to it. In the case of a tie we can either arbitrarily choose, assign x to the class that has the nearest average sample between the k_i samples, assign x to the class that has the k_i samples with the least distance from it or assign x based on any other heuristic we want.

4 Discriminant functions

Considering a problem of binary classification the goal is to create a linear function $g(x)$ that separates the samples belonging to the two classes:

$$g(x) = w^T x + w_0$$

where $x = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}$ is the vector of samples, $w = \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix}$ is the **weights vector** and w_0 is the **bias**.

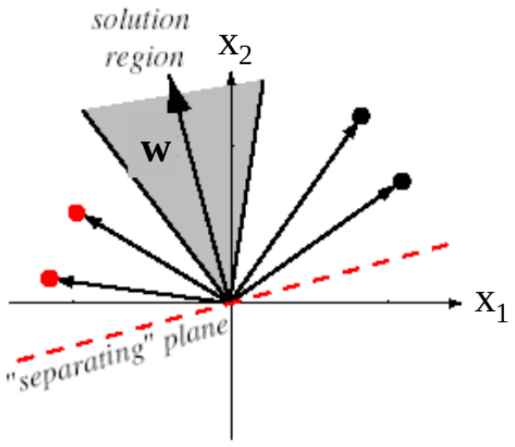
A sample x_i is classified as belonging to ω_1 if $g(x) = w^T x + w_0 > 0$, to ω_2 otherwise. We can generalize further by including the bias in the weight vectors, i.e. by setting $x' = \begin{bmatrix} 1 & x_1 & \dots & x_n \end{bmatrix}$ is the vector of samples, $w' = \begin{bmatrix} w_0 & w_1 & \dots & w_n \end{bmatrix}$.

The goal is then to calculate the weights for which $w^T x_i > 0$ for each x_i belonging to class ω_1 and $w^T x_i < 0$ for each x_i belonging to class ω_2. The second expression can be rewritten as $w^T(-x_i) > 0$, therefore we can define a normalization on the sample in order to reduce the problem to finding a suitable weight vector such that $w^T x_i > 0$ regardless of the class. Such a vector is called **separator vector**.

The equation:

$$w^T x_i = 0$$

defines an hyper-plane with normal x_i in the **weight space**.The separator vector is then a point in the weight space, and its components are constrained by each of the samples x_i.

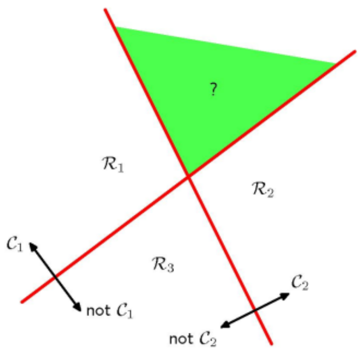


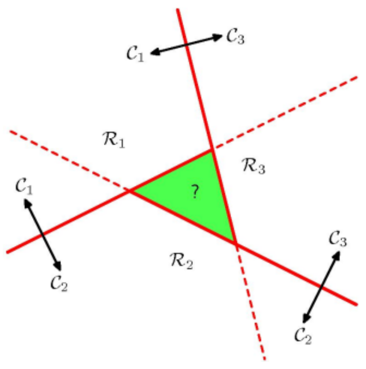
Note that if a solution exists, it is not unique. To constrain the solution we can either seek a unit length weight vector that maximizes the minimum distance from the samples from the separating plane, or seek the minimum length vector that satisfies:

$$w^T x_i \geq b \quad \forall i$$

where b is a positive constant called **margin**.

Both techniques seek the weighting vector close to the middle of the solution region. In the case of C-class problems, C1 classifiers $g_i(x)$ can be constructed, one for each class C_i versus every other $C_j, i \neq j$ class, called **one-vs-rest classifiers**. Another option is building $C(C-1)/2$ binary classifiers (**one-vs-one classifiers**) and then classify a test sample following a majority rule. Both approaches lead to ambiguity areas:





The problem is solved by constructing a single class C classifier that includes C linear functions, and then an unknown vector x_i assigned to class C_i if $g_i(x) > g_j(x) \forall i \neq j$.

We can define a **linear classifier** as:

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$

where w_i are the elements of the weighting vector and w_0 is the bias.

Alternatively we can define a **quadratic classifier**:

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij} x_i x_j$$

which results in a separation surface that delimits the clusters of points belonging to different classes.

We will use linear classifiers because they are easier to handle and can be represented by a piecewise linear function. Also we can easily apply linear transforms to the features in order to have classification surfaces describable with linear discriminant functions.

4.1 Linear discriminant functions

Consider a function h defined as:

$$g_i(x) = h_i(w, x') = w_0 \sum_{k=1}^{n} w_k x_k$$

To determine the weighting vector we will apply a **gradient descent** procedure, which is an iterative method based on the notion that the gradient vector in a space W points to the direction of maximum change of the function to be maximized or minimized.

In our case we define the weight vector at step $k+1$ of the iteration as:

$$w(k+1) = w(k) - \rho_k \nabla J(w) \mid_{w=w(k)}$$

where $J(w)$ is an evaluating function to be minimized. It is chosen in such a way as to reach the minimum when w approaches the optimal solution, or in other words it needs to be convex. ρ_k is a scalar value that varies with k and adapts the magnitude of the correction.

At the k-th step we arbitrarily choose a vector $w(1)$ and calculate the gradient vector $J(w(1))$. We then obtain a vector $w(2)$ at a certain distance (corrected with ρ_k) in the direction of maximum descent.

We write the second order Taylor expansion of J:

$$J(w(k+1)) \approx J(w(k)) + \nabla J(w(k))(w(k+1) - w(k)) + \frac{1}{2}(w(k+1) - w(k))^T D(w(k+1) - w(k))$$

where D is the Hessian matrix of J, that is:

$$D_{ij} = \frac{\partial^2 J}{\partial w_i \partial w_j} \mid_{w=w(k)}$$

Substituting the expression for $w(k+1)$:

$$J(w(k+1)) \approx J(w(k)) - \rho_k |\nabla J(w(k))|^2 + \frac{1}{2}\rho_k^2 \nabla J^T D \nabla J$$

We can minimize J with respect to ρ_k as:

$$\frac{\partial J(w(k+1))}{\partial \rho_k} \implies -|\nabla J(w(k))|^2 + \rho_k \nabla J^T D \nabla J = 0 \implies \rho_k^\circ = \frac{|\nabla J|^2}{\nabla J^T D \nabla J} \mid_{w=w(k)}$$

Another way is to look for a minimum with respect to $w(k+1)$:

$$\frac{\partial J}{\partial w(k+1)} \implies \nabla J(w(k)) + \frac{1}{2}D(w(k+1) - w(k)) + \frac{1}{2}(w(k+1) - w(k))^T D = 0 \implies w(k+1) = w(k) - D^{-1}\nabla J \mid_{w=w(k)}$$

The expression poses a computational problem, because we iteratively need to compute the inverse of the Hessian, which is an expensive task, not to mention that the inverse might not even exist. A more practical choice is to fix ρ_k once and keep it constant. A possible heuristic is $\rho_k = \frac{1}{k}$, which has the benefit of increasing the convergence rate and avoiding oscillations.

4.2 Perceptron method

We now need to define a criterion function for solving the inequalities $w^t x_i > 0$. A good choice for J is:

$$J(w) = -\sum_{i \in X} w^T x_i$$

where X is the set of samples misclassified by W. Such a $J(w)$ is proportional to the sum of the distances of misclassified samples from the decision boundary and it converges only under the strong assumption that the linear separation between the two classes is linear.

The gradient descent algorithm becomes:

$$\nabla J = -\sum_{i \in X} x_i \implies w_{k+1} = w_k + \rho_k \sum_{i \in X} x_i \quad \rho_k = \frac{1}{k}$$

An evolved version of this method will be introduced with neural networks.

4.3 Relaxation method

The function $J(w)$ takes the form:

$$J_q(w) = \sum_{i \in X} (w^T x_i)^2$$

This form also takes into account the misclassified samples and is minimized by the separator vector w. The difference with the perceptron method is that now the gradient of J is continuous, so such a function is suitable for finding smooth separation surfaces. A flaw of this method is that experimentally it is often the case that the solution is the trivial $w = 0$.

A better option is:

$$J(w) = \frac{1}{2} \sum_{i \in X} \frac{(w^T x_i - b)^2}{\|x_i\|^2}$$

where $b \geq 0$ is the margin and $w^t x_i \leq b$.

4.4 Minimum square error method (MSE)

Until now we only considered misclassified samples. Now we will define a criterion function that considers all samples. The goal becomes to verify $w^t x_i = b_i$ where b_i are some arbitrary positive constants. Applying the previous equation for N samples we obtain N equations, which we can express in matrix notation:

$$Xw = b$$

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} & \pm 1 \\ \vdots & \ddots & & \vdots & \vdots \\ x_{N1} & \dots & \dots & x_{Nn} & \pm 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

Generally $N \gg n + 1$ so we have enough equations to determine the unknowns.

We define the **error**:

$$e = Xw - b$$

and also the **quadratic error**:

$$J_M(w) = (Xw - b)^T (Xw - b)$$

Basically we want J_M to have a minimum with respect to w, so we compute the gradient:

$$\nabla_m J_M = 2X^T(Xw - b) = 2(Xw - b)^T X = 0 \implies X^T X w = X^t b$$

Multiplying by $(X^T X)^{-1}$ we get:

$$w = (X^T X)^{-1} X^T b = X^\dagger b$$

where $X^\dagger$ is the **pseudoinverse matrix** of X. While computing an $n + 1 \times n + 1$ inverse is not computationally cheap, in this case it is a one-shot process, not an iterative one.

4.5 Least MSE method

As stated before:

$$J_M(w) = (Xw - b)^T (Xw - b) = \|Xw - b\|^2$$

can be minimized with gradient descent. In this way we bypass the problems that arise if $X^T X$ is singular, and we also avoid treating large matrices.

The gradient descent algorithm in this case is:

$$w_{k+1} = w_k - \rho_k X^T (Xw_k - b)$$

As stated before, if $\rho_k = \frac{1}{k}$ then the above expression converges to the vector obtained by $X^T(Xw - b) = 0$.

Considering a sample at a time we get:

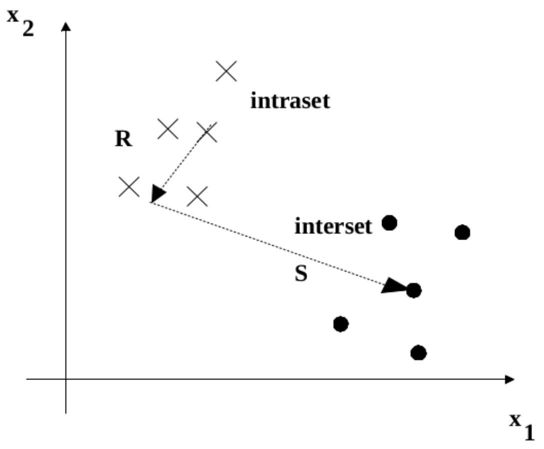
$$w_{k+1} = w_k - \rho_k (b_k - w_k^T x_k) x_k$$

which is called **least minimum square error method** or **Widrow-Hoff method**. b_k is the margin value for the k-th sample, on which it depends.

The descent algorithm differs from the relaxation method in the correction Widrow-Hoff implements on w_k whenever $w_k^T x_k \neq b_k$. In many cases it is impossible to satisfy all the $w_k^T x_k = b_k$ equations, and thus the correction is endless. That is the reason we need a ρ_k decreasing with k in order to ensure convergence. The disadvantage is that we need to iterate around 3-4 times the number of samples.

5 Principal Component Analysis

The goal of a linear transformation is to find a transformation matrix W that turns an unstructured set of samples y into a simpler set x on which we apply classification.



The problem is determining the matrix W. To do so we define an **interset distance** S:

$$S = \frac{1}{M_1 M_2} \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} d^2(y_i^{(1)}, y_j^{(2)})$$

that is to say the average of all possible distances between two samples of different classes.
We also define an **intraset distance** R:

$$R = \frac{1}{M(M-1)} \sum_{i=1}^{M} \sum_{j=1}^{M} d^2(y_i, y_j)$$

which is the average of all possible distances within a class.
With the W transformation we want to maximize S and minimize R in order to cleanly separate the classes. The goal could therefore be defined as:

$$Q(\mathbf{x}) = \left.\frac{R_1 + R_2}{S}\right|_{min}$$

or in any similar definition that maximizes the interset and minimizes the intraset distance. The transformation corresponding to the minimal $Q(\mathbf{x})$ becomes nonlinear though, so it becomes complex to handle.
The best choice for the distance function is to use a function with continuous first derivative approximating an Euclidean distance up to some threshold and then consider a fixed distance, in order to reduce the "overweighting" of the very distant samples.
An example is the **sigmoid function**:

$$d(y_1, y_2) = 1 - e^{-\frac{1}{2D^2} d^2(y_1, y_2)}$$

where d^2 is the Euclidean distance, which is continuous and tends to saturate.
Considering the starting space of dimension $dim(y) = m$ and the result space of dimension $dim(x) = n$, then the transformation matrix W will have dimension $dim(W) = n \times m$.
We have:

$$S = \frac{1}{M_1 M_2} \sum_{q=1}^{M_1} \sum_{p=1}^{M_2} \left(W(y_q^{(1)} - y_p^{(2)}) \right)^2$$

Then:

$$S = \frac{1}{M_1 M_2} \sum_{q=1}^{M_1} \sum_{p=1}^{M_2} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} w_{ij} (y_{qj}^{(1)} - y_{pj}^{(2)}) \right)^2$$

$$R_1 = \frac{1}{M_1(M_1-1)} \sum_{q=1}^{M_1} \sum_{p=1}^{M_1} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} w_{ij} (y_{qj}^{(1)} - y_{pj}^{(2)}) \right)^2$$

$$R_2 = \frac{1}{M_2(M_2-1)} \sum_{q=1}^{M_2} \sum_{p=1}^{M_2} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} w_{ij} (y_{qj}^{(1)} - y_{pj}^{(2)}) \right)^2$$

The solution for w_{ij} can be complex depending on the objective function. We will consider:

- W diagonal, i.e. with $w_{ij} = 0, i \neq j$. The objective is to have a minimum $(R_1 + R_2)$. To solve this we apply the **Langrangian minimum condition**:

$$\sum_k w_{kk} = 1 \implies w_{kk} = \frac{1}{\sigma_k^2 \sum\limits_{j=1}^{n} \frac{1}{\sigma_j}}$$

where:

$$\sigma_k^2 = \frac{1}{N-1} \sigma_{i=1}^{N} (y_{ik}^{(l)} - \overline{y}_k^{(l)})^2$$

where l is the class label. $N = M_1 + M_2$ if $R_1 + R_2$ is minimum, while $N = M_1$ and $N = M_2$ if R_1 or R_2 are minimum respectively. The term σ_k^2 is the variance of the samples along the k-th component, and it is a measure of the reliability of the k-th measurement.

Another constraint we could impose is the **constant volume constraint**:

$$\prod_k w_{kk} = 1 \implies w_{kk} = \frac{1}{\sigma_k} \left(\prod_{j=1}^{n} \sigma_j \right)^{\frac{1}{n}}$$

- W arbitrary. Besides wanting to minimize $R_1 + R_2$, we also need the constraint that $R_1 + R_2 + S$ should be constant. We define the matrix BC^{-1} where:

$$B \text{ interset } \rightarrow b_{jk} = \frac{1}{M_1 M_2} \sum_{q=1}^{M_1} \sum_{p=1}^{M_2} (y_{qj}^{(1)} - y_{pj}^{(2)})(y_{qk}^{(1)} - y_{pk}^{(2)})$$
$$C \text{ intraset } \rightarrow c_{jk} = \frac{2}{M(M-1)} \sum_{q=1}^{M} \sum_{p=1}^{M} (y_{qj}^{(l)} - y_{pj}^{(l)})(y_{qk}^{(l)} - y_{pk}^{(l)})$$

where $M = M_1 + M_2$ and $l = 1, 2$.

5.1 Fisher transform

The problem is to reduce the dimensionality of the feature space in order to decrease the computational complexity. Essentially we reduce the d-dimensional problem to a 1-dimensional problem by projecting the feature space onto a straight line with a specific direction. The problem is to look for such a direction.

Suppose to have a set of N d-dimensional samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$, of which N_1 are classified as ω_1 and N_2 are classified as ω_2. We are looking for a transformation:

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

that turns the vectors of samples into scalars. Since we want to be able to separate the classes in the new 1-dimensional space, we need a measure of the separation between the classes, which for instance could be the difference of the means:

$$\tilde{m}_1 = w^T m_1 \qquad \tilde{m}_2 = w^T m_2$$

where:

$$m_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbf{x}_i^{(1)} \qquad m_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} \mathbf{x}_i^{(2)}$$

are the means before the transformation and:

$$\tilde{m}_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}\mathbf{y}_i^{(1)} \qquad \tilde{m}_2 = \frac{1}{N_2}\sum_{i=1}^{N_2}\mathbf{y}_i^{(2)}$$

are the means after the transformation.
Then we define the **Fisher linear discriminant** as the linear function $w^T\mathbf{x}$ for which J is maximum:

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

where $\tilde{s}_1^2$ and $\tilde{s}_2^2$ are the **scatters** of the classified samples ω_1 and ω_2, defined as:

$$\tilde{s}_i^2 = \sum_{j=1}^{N_i}(y_j^{(j)} - \tilde{m}_i)^2$$

We want to have small dispersions, or in other words we want the samples to be clustered around the mean value. To get J as an explicit function of w we define the **scatter matrices** as:

$$S_i = \sum_{j=1}^{N_i}(x_j^{(j)} - m_i)^T(x_j^{(j)} - m_i) \qquad S_w = S_1 + S_2$$

Therefore:

$$\tilde{s}_i^2 = w^T S_i w \implies \tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_w w \implies (\tilde{m}_1 - \tilde{m}_2)^2 = w^T S_B w$$

where:

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

So we can express J as:

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{w^T S_B w}{w^T S_w w}$$

We finally find the maximum of J:

$$\frac{\partial J}{\partial w} = 0 \implies w = S_w^{-1}(m_1 - m_2)$$

which is the **Fisher transform**.

5.2 Curse of dimensionality

In practice one can find many training sets, which can have few samples, too few or too many features, fixed unselectable features, non-independent features and so on. For the design of a classifier we need to consider the computational complexity of the system and the influences the dimensionality and cardinality of the training set have on the accuracy of the classification.

If the features are statistically independent, then it can be proven that the optimal performance is reached. It can be shown that with equal priors the probability of error is:

$$P(e) = \frac{1}{\sqrt{2\pi}}\int_{\frac{r}{2}}^{\infty} e^{-\frac{u^2}{2}}du$$

where:

$$r^2 = (\mu_1 - \mu_2)^T\Sigma^{-1}(\mu_1 - \mu_2)$$

is the square of the Mahalanobis distance. The probability of error tends to zero when r tends to infinity.
If $\Sigma = diag(\sigma_1^2, \ldots, \sigma_d^2)$ then:

$$r^2 = \sum_{i=1}^{d}\left(\frac{\mu_{i1} - \mu_{i2}}{\sigma_i}\right)^2$$

Each independent feature contributes with a decrease of the probability of error.

In general adding features increases the performance of the classifier at the cost of complicating the classifier and the feature extractor. Furthermore in practice adding features is often linked to worse performance due to an incorrect model or an insufficient number of samples that result in an inaccurate estimation of their distributions.

5.2.1 Overfitting

When the number of samples is insufficient we can:

- Try to reduce the dimensionality of the features.
- Combine the features into representative clusters (**feature extraction**).
- Determine a better estimate for a given estimate of a covariance matrix.
- Impose a threshold or diagonality to a covariance matrix.

In this way we can force independence of the features, which could be independent in reality though. So the performance may not be optimal, because of the lack of data. This concept is similar to the problem of data fitting, i.e. the difference between interpolation and approximation.

5.2.2 Main Component Method (PCA)

Given a set of M $n \times 1$ vectors:

$$\mathbf{x} = \{\mathbf{x}_1^T, \dots, \mathbf{x}_M^T\}$$

the average is defined by the **expected value operator**:

$$m_x = E(\mathbf{x})$$

The covariance matrix of the set is defined as:

$$C_x = E((\mathbf{x} - m_x)(\mathbf{x} - m_x)^T)$$

Since $\mathbf{x}$ is n-dimensional, C_x is $n \times n$. Its elements $c_{ij}, i = j$ represent the variance of the i-th component of $\mathbf{x}$, while its elements $c_{ij}, i \neq j$ represent the covariance between the components of $\mathbf{x}$.

Considering the size of the set we have:

$$m_x = \frac{1}{M}\sum_{k=1}^{M}\mathbf{x}_k \qquad C_x = \frac{1}{M}\sum_{k=1}^{M}(\mathbf{x}_k - m_k)(\mathbf{x}_k - m_k)^T = \frac{1}{M}\sum_{k=1}^{M}\mathbf{x}_k\mathbf{x}_k^T - m_x m_x^T$$

Let e_i and $\lambda_i, i = 1, \dots, n$ be the eigenvectors and eigenvalues of C_x, ordered in descending order:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$$

Let A be the matrix whose rows are the eigenvectors of C_x, and use it to transform the set $\mathbf{x}$ as follows:

$$y = A(\mathbf{x} - m_x)$$

The average of the y vectors is null, while their covariance matrix is $C_y = AC_xA^T$, which is the diagonal matrix $diag(\lambda_1, \lambda_2, \dots, \lambda_n)$. In other words all elements of y are uncorrelated from each other. Since A is orthonormal we have $A^{-1} = A^T$, and therefore:

$$\mathbf{x} = A^T y + m_x$$

Example 5.1 (Appearance-based recognition)
Appearance based recognition is a view-based method, in which we assume that:

- *Each image contains only one object.*
- *Objects are captured by a fixed camera with weak perspective conditions.*
- *The images are normalized in size.*

- *The energy:*
$$\sum_{i=1}^{N}\sum_{j=1}^{N} I(i,j)^2 = 1$$
of each image is normalized to 1.
- *The object is completely visible.*

The comparison between images is performed by means of the ***correlation operation****:*

$$c = I_1 \otimes I_2 = \frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{N} I_1(i,j)I_2(i,j)$$

where K is some normalizing constant.

Given O objects, P points of view, L lighting directions we have OPL images in the database. We calculate the covariance matrix Q and represent every image $\mathbf{x}_p$ with its eigenspace vector of coordinates g_{pl}°. To represent the images we only consider the largest eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \;\; \lambda_i \approx 0 \;\; i > k$$

$$\mathbf{x}_j \approx \mathbf{x}_m + \sum_{i=1}^{k} g_{ji} e_i$$

with e_i eigenvectors of matrix Q and g_{ij} coefficients associated to image g. The various points of view make the point g_{pl}° vary continuously in the eigenspace, creating a ***manifold****.*

Under the normalization hypotesis we have to maximize the correlation, i.e. minimize the distance:

$$\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = \left\|\sum_{i=1}^{n} g_{1i}e_i - \sum_{i=1}^{n} g_{2i}e_i\right\|^2 \approx \left\|\sum_{i=1}^{k} g_{1i}e_i - \sum_{i=1}^{k} g_{2i}e_i\right\|^2 = \left\|\sum_{i=1}^{n}(g_{1i} - g_{2i})e_i\right\|^2 = \sum_{i=1}^{k}(g_{1i} - g_{2i})^2 = \|g_1 - g_2\|^2$$

The procedure suggests to compute the image points in the eigenspaces for all the images in the database, obtaining a point g_y. Them we look for the curve $g_{pl}^{\circ} = \begin{bmatrix} e_1 & e_2 & \dots & e_k \end{bmatrix}(x_{pl}^{\circ} - x_m)$ closest to $g_y = \begin{bmatrix} e_1 & e_2 & \dots & e_k \end{bmatrix}(y - y_m)$.

Finding the curve closest to the point is not always trivial, not to mention the fact that finding the eigenvalues of large matrices is computationally expensive. There are further complications with the separation of the object from the background of the image.

6 Support Vector Machines

6.1 Non linearly separable classes

6.2 Data mapping

6.3 Kernel