# Machine Learning and Pattern Recognition

Alex Carluccio

## Introduzione al Machine Learning

### Tipologie di apprendimento

A seconda del problema è possibile identificare 2 grandi rami, ovvero:

- Supervised Learning dove l'obiettivo è quello di trovare un mapping tra i dati di input ed i dati di output. In questo rampo i task più comuni sono Classification e Regression.
- Unsupervised Learning dove l'obiettivo è quello di identificare qualche struttura utile nei dati. In questo ramo i task più comuni sono Clustering, Density Estimation e Dimensionality Reduction. Va detto inoltre che spesso le tecniche non supervisionate possono essere utilizzate come step di pre-processing dei dati.

### Pattern Classification

Nella pattern classification noi vogliamo assegnare un pattern ad una classe, ovvero vogliamo trovare un tratto che si ripete in "tutti" gli elementi della classe. Per fare ciò possiamo disporre di 2 diversi tipi di classificatore, ovvero:

- Binario, con solo 2 possibilità.
- Multiclasse, ovvero non limitato a sole 2 classi. Nel caso di problemi "Multiclasse" è possibile realizzare sia un classificatore Closed-Set dove il numero di classi è noto già a priori e sia un classificatore Open-Set dove abbiamo una classe "jolly" che va a raccogliere tutti gli oggetti che non è stato possibile inserire nelle classi precedenti.

Prima di addentrarci nei vari dettagli è bene dire che il problema viene diviso in 3 step per semplicità:

1. Feature Extraction, dove si cerca di rappresentare un oggetto sottoforma di attributi numerici di nostro interesse, andando dunque ad eliminare quelli superflui o inutili.
2. Dimensionality Reduction, dove si cerca di creare un maping tra lo spazio delle features n-dimensionali allo spazio delle features m-dimensionali con m<<n. In questo step si cerca anche di evitare il cosiddetto "Overfitting", ovvero il troppo adattamento del modello al set di dati che sto usando per generarlo e la conseguente perdita di capacità nella generalizzazione. Ciò che si cerca di fare in conclusione è andare a comprimere i dati rimuovendo il rumore.
3. Classification, dove si crea il vero e proprio classificatore per fare il mapping. Il mapping viene spesso chiamato "decision function".

La decision function deve essere in grado di generalizzare e non andare a overfittare i dati di training.

### Generative Probabilistic Model

Sono modelli che utilizzano la distribuzione standard di features e labels $P\left(x, C_k\right) = P\left(x|C_k\right)P\left(C_k\right)$ e applicano il teorema di Bayes per calcolare la probabilità a posteriori. Dove la probabilità $P(x|C_k)$ equivale alla probabilità di x (feature) di appartenere alla classe $C_k$.

### Discriminative Probabilistic Model

Modellano direttamente la probabilità a posteriori $P(C_k|x_t)$. Non possono incorporare direttamente informazioni dipendenti dall'applicazione.

### Discriminative Non-Probabilistic Model

L'output è uno score s che pò essere preso come una misura della forza delle ipotesi sotto test.

### Valutazione della qualità

Dopo aver realizzato un classificatore si può essere interessati alla qualità delle predizioni fatte e alle performance ottenute su dei dati mai visti. Poiché però non è possibile vedere le performance del classificatore su dati mai visti, si cerca di simularli andando a dividere il set di dati di partenza in 2 gruppi (Cross-Validation o in altri casi K-Fold Cross Validation):

- Test set. Set di dati usati per valutare il classificatore.
- Training Set. Set di dati usati per effettuare il training del classificatore.

## Riduzione delle dimensioni

Dimensionality reduction techniques compute a mapping from the n–dimensional feature space to a m–dimensional space, with m<<n . We will focus on two linear methods:

- Principal Component Annalysis (PCA)
- Linear Discriminant Analysis (LDA)

In both cases, we want to find a subspace of the feature space that preserves most of the "useful" information.

### PCA

Given a "zero-mean dataset" $X = \{x_1, \dots x_k\}$ with $x_i \in^n$ we want to find the subspace that allows preserving most of the information . A subspace can be represented as a matrix $P \in \mathbb{R}^{n\times m}$ whose columns are orthonormal and form a basis of a subspace of $\mathbb{R}^n$ with dimension $m$. The projection of $x$ over the subspace is given by:

$$y = P^T \times x = \begin{bmatrix} p_1^T x \\ p_2^T x \\ . \\ . \\ p_1^T x \end{bmatrix}$$

where $p_1 \; . \; . \; . \; . \; . \; p_m$ are the columns of $P$. We can compute the coordinates of the projected point $y$ in the original space as $\widehat{x} = P \times y$. The question is: How can we calculate $P$ ?

A reasonable criterion may be the minimization of the average reconstruction error. Where $K$ is the number of samples.

$$\boldsymbol{P^*} = \arg\min_P \frac{1}{K} \sum_{i=1}^{K} \|x_i - \widehat{x}_i\|^2$$

In other words, PCA reduces dimensions by focusing on the direction with the most variation. This is useful for plotting datat with a lot of dimension onto a simple X/Y plto. However in some cases we can not super interested in the directions with ther most variation, and for those cases we can try to use LDA.

**LDA**

PCA is an unsupervised method so it no guarantee of obtaining discriminant directions. For this reason we want a transformation that allows us to better separate the classes. Initially LDA was used only for classification tasks, but as tiems goes by it started to be used also as a dimensionality reduction technique. Problem: In some cases LDA can't operate as well as we espect, this happen when data points of each class are scattered along the same directions of the class mean (As shown in Fig 1).

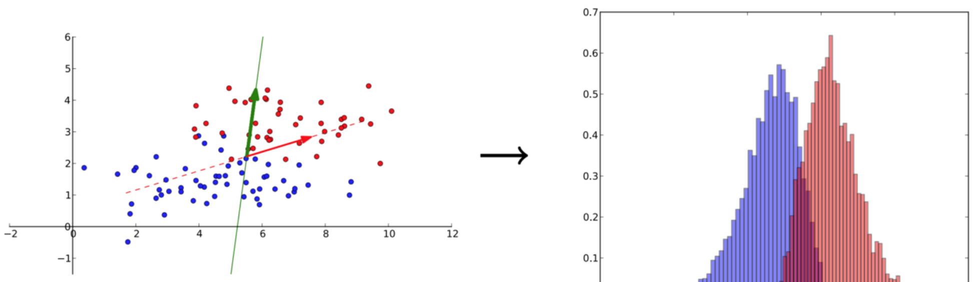


Figure 1: Red Line = PCA, Green Line = LDA

To do that LDA aspire to find a direction $\vec{w}$ that has a large separation between the centre of the classes and small spread inside each class. We measure spread in terms of class *covariance*. A better definition of LDA is: *LDA maximize the between-class variability and minimize the within-class variability.*

$$\max_w \frac{w^T\ S_B\ w}{w^T\ S_W\ w}$$

Where the e between and within class variability matrices are defined as:

$$S_B = \frac{1}{N}\sum_{c=1}^{K} n_c(\mu_c - \mu)(\mu_c - \mu)^T \qquad S_W = \frac{1}{N}\sum_{c=1}^{K}\sum_{i=1}^{n_c}(x_{c,i} - \mu_c)(x_{c,i} - \mu_c)^T$$

where $x_{c,i}$ is the $i$-th sample of class $c$, $n_c$ is the number of samples of class $c$, $K$ is the total number of classes, $N$ is the total number of samples, $\mu$ is the dataset mean and $\mu_c$ is the mean of the class.

Since we are looking for a discriminant direction $w$, we can consider the projected samples $w^T \times x$. For this reason also the global-mean $\mu$ and the class-mean $\mu_c$ have a projected version: $m = w^T\ \mu$ and $m_c = w^T\ \ \mu_c$ . Updating the $S_B$ and $S_W$ formulas we obtein:

$$s_B = \frac{1}{N}\sum_{c=1}^{K} n_c(w^T\mu_c - w^T\mu)(w^T\mu_c - w^T\mu)^T = w^T S_B w$$

$$s_W = \frac{1}{N} \sum_{c=1}^{K} \sum_{i=1}^{n_c} (w^T x_{c,i} - w^T \mu_c)(w^T x_{c,i} - w^T \mu_c)^T = w^T S_W w$$

To find $w$ we need to solve this equation:

$$\nabla_W L(w) = 2 \frac{S_B\ w}{w^T\ S_W\ w} - \frac{w^T\ S_B\ w\ S_W\ w}{(w^T\ S_W\ w)^2} = 0$$

Where $L(w) = \frac{s_B}{s_W} = \frac{w^T\ S_B\ w}{w^T\ S_W\ w} = \lambda(w)$. Note that the criterion does not depend on the scale of w, for this reason if $w$ is a maximizer of $L$, then also $\alpha w$ is a maximizer. We can therfore select a maximizer with *unit norm.* We can observe that :

- The optimal solution is an eigenvector of $S_W^{-1}\ S_B$
- The eigenvalue corrisponding to solution $w$ is $\lambda(w) = L(w)$

Despite LDA method was originally used to solve binary problems, it has found large success as a dimensionality reduction technique. In this case we are intreasted in looking for the $m$ most-discriminant directions and we will represent these directions as a matrix $W$, whose columns contain the directions we want to find. (Is not required that $W$ is orthogonal).

The projected points are computed as $\widehat{x} = W^T x$ and also the projected matrices as

$$\widehat{S}_B = W^T\ S_B\ W \widehat{S}_W = W^T\ S_W\ W$$

Notice that, from the definition of $S_B$, the number of non-zero eigenvalues is at most $C - 1$, for this reason LDA allows estimating at most $C - 1$ directions.

## Probability and density estimation

Our goal is to make predictions that allow us to rake actions. Otherwhise this is a complex process because there are too many factors that can influence the outcomes. A solution could be to model phenomena in terms of 2 types of events:

- Deterministic event
- Random event

We can describe random events in term of their probability:

- Classical Interpretation. $P = \frac{favorable\ outcomes}{possible\ outcomes}$
- Frequentist Interpretation
- Bayesian Interpretation

*In 05_Probability.pdf there are 50 slides that recalls basic notion of probability. If you need a recap or you don't understand something, please download the materials and take a look.*

### Bernoulli distribution

The Bernoulli distribution can be used to model the outcome of a binary event. For example the launch of a coin (Head or Tail). Let $X \in \{0, 1\}$ a random variable. The bernoulli distribution of X is:

$$X \sim Ber(p)$$

$$P_X(x) = Ber(x|p) = p^x(1-p)^{1-x} = \begin{cases} p & if \quad x=1 \\ 1-p & if \quad x=0 \end{cases}$$

**Binomial distribution**

The Binomial distribution can be used to count the number of successes in $n$ repeated trials. Let $p$ denote the probability of success for a single trial. Let $p$ denote the probability of success for a single trial. The binomial distribution of $X$ is:

$$X \sim \ Bin(n,p) \qquad P_X(x) = \binom{n}{x} p^x(1-p)^{n-x}$$

We can observe that:

- If $n = 1$ $Bin\,(1,p) \sim Ber\,(p)$
- If $X_1 \perp X_2 ... \perp X_n \sim Ber(p) \Rightarrow Y = \sum_i X_i \sim Bin(n,p)$

**Categorical distribution**

The Bernoulli and Binomial distribution can be extended to events that have $K$ possible outcomes (for example rolling a die). The categorical distribution of $X$ is:

$$X \in \{1, 2, ...K\}^2 \qquad X \ \sim Cat\,(p)$$

$$f_X(x) = P(X=x) = p_x = \prod_i p_i^{\mathbb{I}[x=i]}$$

Where $p = (p_1, \ ... \ p_K)$, with $\sum_{i=1}^{K} p_i \ = \ 1$. Where $p_i$ is the probability of outcome i, and $\mathbb{I}$ is the indicator function.

$$\mathbb{I}[C] = \begin{cases} 1 & if \ C \ is \ true \\ 0 & otherwhise \end{cases}$$

In many cases it's convenient o represent outcomes with a 1 of-K encoding vector:

$X = 1 \ \rightarrow X = (1, 0, ..., 0) \qquad X = K \ \rightarrow \ X = (0, 0, ...., K)$

and for this reason ther density can be espressed as:

$$f_X(x) = \prod_i p_i^{x_i}$$

**Multinomial distribution**

We can consider a set of $n$ trials encoded as $x = (x_1, ...x_K)$ and where $x_i$ denotes the number of occurrencies of outcome $i$ and $n = \sum_{i=1}^{m} x_i$

Let $p = (p_1, .....p_K)$ be the vector of probabilities for a single trial the Multinomial distribution of X is:

$$X \sim Mul(n,p) \qquad f_X(x) = \frac{n!}{x_1!...x_K!} = \prod_{i=1}^{K} p_i^{x_i}$$

### Gaussian or Normal distribution

The Gaussian or Normal distribution is structured as shown:

$$X \sim \frac{\mathcal{N}(\mu, \sigma^2)}{1} \qquad f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where:

- $\mu$ is the mean ($\mathbb{E}[X] = \mu$), and the distribution is symmetric and centered around it.
- $\sigma$ is called standard deviation ($var(X) = \sigma^2$)
- $\lambda = \frac{1}{\sigma^2}$ is called precison

Is possible to observe that if $\mu = 0$ and $\sigma^2 = 1$, we say that $X$ follows a standard normal distribution. $X \sim \mathcal{N}(0, 1)$

### Multivariate Gaussian Distribution

We can extend the Gaussian distribution to random vectors. Lex $X$ be a random vector $X = [X_1, ...., X_N]^T$ where $X_i$ are indipendent and identically distribuited with a standard normal distribution: $X_i \ \sim \mathcal{N}(0, 1)$. The distribution of $X$ is given by the joint distribution of $X_1, ..., X_N$:

$$f_X(x) = \prod_{i=1}^{N} f_{X_i}(x_i)$$

$X$ follows a standard multivariate Gaussian distribution: $X \ \sim \mathcal{N}(0, I)$ where  is the identity matrix. Using this result, if $X$ follows a multivariate Gaussian distribution with mean $\mu$ and covariance matrix $\Sigma$, it can be written as a linear transformation of $Y$ and $\mu$:

$$X = AY + \mu X_t \sim X \sim Ber(\mu, \Sigma)$$

Where $Y \sim \mathcal{N}(0, I)$ and $\Sigma = AA^T$. So $X \sim \mathcal{N}(\mu, \Sigma)$. As we have seen for the 1-dimensional case, $\mu$ represents the mean of the data and $\Sigma$ represents how data are spreaded among different directions

### Density Estimation Discrete Variables

Let consider an example. We want to predict whether a coin flip will be a Head or a Tail (H or T). We don't know anythings about the coin but we have observed a number of tosses $n$. The results are represented in this way: $[x_1, x_2........X_n] = [H, T, T, T, H, T....]$. We will use for Head, $x_i = 1$ and for Tail $x_i = 0$. These results can be considered as outcomes of R.V.s $(X1, X2.....Xn)$ and our goal is to predict if the result of a new toss $X_t$ will be a Head. To do that we have to calculate $P\left(X_t = H \mid X_1 = H, \ .... \ , X_n = T\right) = ?$.

For a moment, let's assume that $P(H) = \pi$ and that the tosses are indipendent and identically ditribuited. Now we can model R.V. $X_i$ as Bernoulli R.V. with parameter $\pi$ and we  obtain $X_i \sim X \sim Ber\left(\pi\right)$, but using the assumption that $X_i$ are i.i.d also $X_t$ is distribuited as $X_t \sim X \sim Ber\left(\pi\right)$. Unfortunately we don't know the value of $\pi$ and a possible way to estimate a "good" value consist in looking for a value of $\pi$ that can better explain the observed tosses.

For example if we consider $\pi = 0,7$ we will have:

$$L(0,7) = P(X_1 = H|\pi = 0,7) \times P(X_2 = H|\pi = 0,7) \times ... \times P(X_n = T|\pi = 0,7) = 0,7 \times 0,7 \times ... \times 0,3$$

Where $L(\pi)$ is the Likelihood function: (Note that $\pi$ is not treated as random value)

$$L(\pi) = P(X_1 = x_1 ... X_n = x_n | \pi)$$

Using our assumption that the tosses are indipendent we obtain:

$$P(X_1 = x_1 ... X_n = x_n | \pi) = \prod_{i=1}^{n} P(X_i = x_i | \pi)$$

and since $P(X_i = x_i | \pi) = Ber(x_i | \pi) = \pi^{x_i}(1-\pi)^{1-x_i}$

the likelihood function becomes:

$$L(\pi) = \prod_{i=1}^{n} \pi^{x_i}(1-\pi)^{1-x_i}$$

Using this result, we can compute the Maximum Likelihood estimate for $\pi$ as the value that maximizes the likelihood function. To solve for $\pi$ we can consider that:

$$l\left(\pi\right) = \log\,\left(L(\pi)\right) = \sum_{i=1}^{n} x_i \log(\pi) + (1 - x_i)\log(1-\pi)$$

In addition, because wh want the max value, we set the derivate equal to 0:

$$\frac{dl}{d\pi} = \sum_{i=1}^{n} \frac{x_i}{\pi} - \frac{1-x_i}{1-\pi} = 0$$

And we obtain: $\pi_{ML} = \frac{1}{n}\sum_{i=1}^{n} x_i \; = \; \frac{\#H}{\#H + \#T}$

We can finally calculate $P\left(X_t = H \mid X_1 = H, \; .... \; , X_n = T\right) = \pi_{ML}$

Problem: if we have a simple scenario with just 3 observation of heads, using this method we will predict that the coin will never land a tail.

**Density Estimation Continue Variable**

When modeling continuous values, Gaussian distributions arise naturally in a wide variety of contexts. For examle let's assume we have some data $D = (x_1, ...., x_n)$ and we decide to model the data as samples of a Gaussian distribution, with mean $\mu$ and varaince $\nu$. We also assume that the points have been generated by indipendend and identically distributed R.V. Given the velue of the model parameter $\theta = (\mu, \nu)$, the distribution of $X$ is:

$$f_{X|\theta}(x) = \mathcal{N}(x|\mu, \nu)$$

And for the likelihood we obtain:

$$L(\theta) = \prod_{i=1}^{n} f_{X_i|\theta}(x_i) = \prod_{i=1}^{n} \mathcal{N}(x_i|\mu, \nu)$$

Also in this case we use a frequentist approach. We assume the existance of a "true" value of $\theta$ and also for $\mu$ and $\nu$ and we want to find an estimator for that values. In general an estimator is a function $T$ that maps our dataset $D$ to values for the model parameters $\theta$.

**Methods of Moments (MOM)**

For the Gaussian distribution the first two moments are $\mu$ and $\nu$. Is possible to write 2 equations to produce 2 estimators:

$$\mu_{MOM} = \frac{1}{n}\sum_i x_i \qquad \nu_{MOM} = \frac{1}{n}\sum_i (x_i - \mu_{MOM})^2$$

In general the MOM approach doesn't produce very accurate estimators.

**Maximum Likelihood**

The Maximum Likelihood estimator is the value that maximizes the likelihood. Very often it's better to work with the logarithm of the likelihood: $\;l\,(\theta) = \log\,(L\,(\theta))$.

Since we assumed that $X_i$ are indipendent and $X_i \sim X$, then:

$$L(\theta) = f_{X_1 \ldots X_n}(x_1 \ldots x_n|\theta) = \prod_{i=1}^{n} f_{X_i}(x_i|\theta) = \prod_{i=1}^{n} f_X(x_i|\theta) = \prod_{i=1}^{n} \mathcal{N}(x_i|\mu,\nu)$$

Then applying the logarithm and the Gaussian density, we obtain:

$$l(\theta) = \sum_{i=1}^{n} \log(\mathcal{N}(x_i|\mu,\nu))$$

Remember that:

- $\nu = \lambda^{-1}$
- $\xi$ collects constant terms that are irrilevant for the optimization

So we can express:

$$\log(\mathcal{N}(x_i|\mu,\nu)) = \xi + \frac{1}{2}\log(\lambda) - \frac{\lambda}{2}(x-\mu)^2$$

And the log-likelihood is then:

$$l(\theta) = \sum_{i=1}^{n} \log(\mathcal{N}(x_i|\mu,\nu)) = \xi + \frac{n}{2}\log(\lambda) - \frac{\lambda}{2}\sum_{i=1}^{n} x_i^2 + \lambda\mu\sum_{i=1}^{n} x_i - \frac{n\lambda\mu^2}{2}$$

The ML estimate can be obtained by taking solving for:

$$\begin{cases} \frac{\partial l}{\partial \mu} = 0 = n\lambda\mu - \lambda\sum_{i=i}^{n} x_i \\ \frac{\partial l}{\partial \lambda} = 0 = \frac{n}{2\lambda} - \frac{1}{2}[\sum_{i=1}^{n}(x_i-\mu)^2] \end{cases}$$

Thus:

$$\begin{cases} \mu_{ML} = \frac{1}{n}\sum_{i=1}^{n} x_i \\ \nu_{ML} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{ML})^2 \end{cases}$$

In this case we can observe that the solution is the same as the one obtained by the MOM approach.

## Linear and Quadratic Classifiers (Generative Models)

Let consider a classification problem with a closed set. We have a pattern $x_t$ and we assume that $x_t$ is a realization of R.V. $X_t$. We also assume that class label can be described by R.V. $C_t \in \{1...k\}$ where $1...k$ are the class labels.

### Optimal Bayes Decision

Assign the class with highest posterior probability $c_t^\star = argmax_c P(C_t = c \mid X_t = x_t)$. For example, let's consider a classification task where $x_t$ is an image and labels represent what object is depicted (e.g. cat=1, dog=2,rabbit=3,...). We want to find which label $c_t$ is more likely for $x_t$. For all labels $c \in \{1...K\}$ we can compute $P(C_t = c \mid X_t = x_t)$. This probability is the probability that the class $C_t$ for the test sample $t$ is $c$, conditioned on the observed value $X_t = x_t$.

To simplify let assume that samples are indipendent and distribuited according $X_t, C_t \sim X, C$. For any test sample the joint distribution of $X, C$ be $f_{X,C}$. So from the Bayes rule:

$$P(C_t = c|X_t = x_t) = \frac{f_{X,C}(x_t, c)}{\sum_{c' \in C} f_{X,C}(x_t, c')}$$

In addition the joint density for $(X_t, C_t)$ can be espressed as:

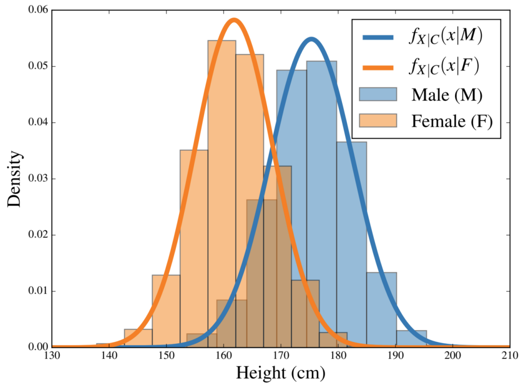
$$f_{X_t,C_t}(x_t, c) = f_{X,C}(x_t, c) = f_{X|C}(x_t|c)P_C(c)$$

Where $P_C(c)$ or simply $P(c)$ (prior probability) represent the probability of the class being c, before we observe the test sample. Usually $P(c)$ depends from the application.

### Gaussian Classifier

Usually we consider problems where $x \in \mathbb{R}$, so how can we model $f_{X|C}(x_t|c)$? The answer is: It depends from the data.

In the following example we assume that the data of each class can be modeled by a Multivariate Gaussian Distribution. Intuitively we can fit a Gaussian density over the samples of each class.



To fit a Gaussian density we have to find the 2 parameters that describe the distribution: $\mu$ and $\sigma^2$. To do that we can use the Maximum Likelihood parameters for both class:

$$\mu_M = \frac{1}{N_M}\sum_{i|C_i=M} x_i = 175.33cm \qquad \sigma^2_M = \frac{1}{N_M}\sum_{i|C_i=M}(x_i-\mu_M)^2 = 52.89cm^2$$

$$\mu_F = \frac{1}{N_F}\sum_{i|C_i=F} x_i = 161.82cm \qquad \sigma^2_F = \frac{1}{N_F}\sum_{i|C_i=F}(x_i-\mu_F)^2 = 46.89cm^2$$

Now we are able to compute the likelihood for the two classes for the 174 cm sample:

$$f_{X|C}(174|M) = \mathcal{N}(174|\mu_M,\sigma^2_M) = 0.05395 = \frac{1}{\sqrt{2\pi\sigma^2_M}}e^{-\frac{(174-\mu_M)^2}{2\sigma^2_M}}$$
$$f_{X|C}(174|F) = \mathcal{N}(174|\mu_F,\sigma^2_F) = 0.01198 = \frac{1}{\sqrt{2\pi\sigma^2_F}}e^{-\frac{(174-\mu_F)^2}{2\sigma^2_F}}$$

Please, remind that this result is not sufficient to answer whether the sample is Male or Female. We need to compute the class posterior probability that depends from the prior probability. The prior probability, as mentioned above, is the probability that a sample belongs into a class before we observe it. So in conclusion:

$$P(C=M|X=174) = \frac{f_{X|C}(174|M)P(C=M)}{f_X(174)}$$

$$P(C=F|X=174) = \frac{f_{X|C}(174|F)P(C=F)}{f_X(174)}$$

If we want just the two probabilities we don't need to compute $f_X(174)$, because we can compute the likelihood ratio between the 2 probabilities that simplify $f_X(174)$, and we obtain:

$$\frac{P(C=M|X=174)}{P(C=F|X=174)} = \frac{f_{X|C}(174|M)P(C=M)}{f_{X|C}(174|F)P(C=F)}$$

**Method Formalization for univariate cases**

Now we will formalize the method used in the previous example. We assume that our data, given a class, can be described by a Gaussian distribution, so:

$$(X_t|C_t=c) \sim (X|C=c) \sim \mathcal{N}(\mu_c,\Sigma_c)$$

We have one mean and one covariance matrix per class that we don't know. We don't know the model parameters $\theta = [(\mu_1,\ \Sigma_1)\dots(\mu_k,\ \Sigma_k)]$. On the other hand we have the training dataset $D = \{(x_1,c_1)\dots(x_n,c_n)\}$ where $X = \{x_1 \dots x_n\}$ are the observed samples and $C = \{c_1 \dots c_n\}$ the corresponding class labels where $c_i \in \{1 \dots k\}$.

We can assume that, given the model parameter $\theta$ , observations are indipendent and identically distribuited. (In other words we assume that both training set and evaluation samples are indipendent and distribuited in the same way). Since we assume Gaussian distribuition for $X|C$, we have :

$$(X_i|C_i=c,\theta) \sim (X_t|C_t=c,\theta) \sim (X|C=c,\theta) \sim \mathcal{N}(\mu_c,\Sigma_c)$$

Again, we don't know $\theta$ and for this reason we don't know for each class the mean and the covaraince matrix. To estimate this values we can use the (log-)likelihood method. The likelihood for $\theta$ is :

$$\mathcal{L}(\theta) = f_{X_1 \ldots X_n, C_1 \ldots C_n|\theta}(x_1 \ldots x_n, c_1 \ldots c_n|\theta) = \prod_{i=1}^{n} f_{X,C|\theta}(x_i, c_i|\theta) = \prod_{i=1}^{n}(x_i|\mu_{c_i}, \Sigma_{c_i})P(c_i)$$

Now we again consider the log-likelihood:

$$l(\theta) = \log(\mathcal{L}(\theta)) = \sum_{i=1}^{n} \log \mathcal{N}(x_i|\mu_{c_i}, \Sigma_{c_i}) + \sum_{i} \log P(c_i) = \sum_{c=1}^{k} \sum_{i|c_i=c} \log \mathcal{N}(x_i|\mu_c, \Sigma_c) + \varepsilon$$

In other words the log-likelihood corresponds to a sum over all classes of the conditional log-likelihood of the samples belonging to each class:

$$l(\theta) = \sum_{c=1}^{k} l_c(\mu_c, \Sigma_c) + \varepsilon \qquad l_c(\mu_c, \Sigma_c) = \sum_{i|c_i=c} \log \mathcal{N}(x_i|\mu_c, \Sigma_c)$$

And now we can maximize $l$ by separately maximizing the terms $l_c(\mu_c, \Sigma_c)$.

**Method Formalization for multivariate cases**

The log-density for a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ is :

$$\log \mathcal{N}(x|\mu, \Sigma) = -\frac{D}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$$

Where we can substitute $\Lambda = \Sigma^{-1}$. The log-likelihood can be expressed as:

$$l_c(\mu_c, \Sigma_c) = k + \frac{N_c}{2} log(|\Lambda_c|) - \frac{1}{2} \sum_{i|c_i=c} (x_i - \mu_c)^T \Lambda_c (x_i - \mu_c)$$

Where we can observe that  the log-likelihood depends from 3 terms:

- $Z_c = N_c$
- $Fc = \sum_{i|c_i=c} x_i$
- $S_c = \sum_{i|c_i=c} x_i x_i^T$

Our target is to find the maximum of $l_c$. To do that, we can compute the derivates and setting them equal to 0:

$$\begin{cases} \nabla_{\Lambda_c} l_c(\mu_c, \Lambda_c) = 0 \\ \nabla_{\mu_c} l_c(\mu_c, \Lambda_c) = 0 \end{cases}$$

Solving this system we obtain the ML expressions for $\mu_c$ and $\Sigma_c$:

$$\mu_c^\star = \frac{1}{N_c} \sum_{i|c_i=c} x_i \qquad \Sigma_c^\star = \frac{1}{N_c} \sum_{i|c_i=c} (x_i - \mu_c^\star)(x_i - \mu_c^\star)^T$$

And now we can compute the likelihood of class $c$ fir test point $x_t$ as:

$$f_{X_t|C_t}(x_t|c) = f_{X|C}(x_t|c) = \mathcal{N}(x_t|\mu_c^\star, \Sigma_c^\star)$$

**Binary Task Example**

Let's consider a binary task with just two classes $C \ \in \{h_1, h_0\}$. We assign a label to a test sample $x_t$ according to the higher posterior probability between $P(C = h_1|x_t)$ and $P(C = h_0|x_t)$. But we can compare this 2 terms using a fraction or a log-likelihood ratio.

$$r(x_t) = \frac{P(C = h_1|x_t)}{P(C = h_0|x_t)} \qquad \log r(x_t) = \log \frac{P(C = h_1|x_t)}{P(C = h_0|x_t)}$$

So, if the log-ratio is $> 0$ the point will be assigned to class $h_1$, else it will be assigned to class $h_0$.

Now let's explicit the log-ratio:

$$\log r(x_t) = \log \frac{P(C = h_1|x_t)}{P(C = h_0|x_t)} = \log \frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)} + \log \frac{P(C = h_1)}{P(C = h_0)}$$

Where:

- $llr(x_t) = \log \frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)}$ is the first term and represent the ratio between the likelihood of observing the sample given that it belongs to $h_1$ or $h_0$.
- $\log \frac{P(C=h_1)}{P(C=h_0)}$ is the second term and represent the prior log-odds. It depends from applications.

The optimal decision is based on the comparison $\log r(x_t) \lesseqgtr 0$ or in another ways we can compare $llr\ (x_t) \lesseqgtr \log \frac{P(C=h_1)}{P(C=h_0)}$. In this expression is possible to see the right term called "threshold".

For multiclass problems $C \ \in \{h_1, h_2, ...h_k\}$ we can compute closed-set posterior probabilities as:

$$P(C = h|x_t) = \frac{f_{X|C}(x_t|h)P(h)}{\sum_{h' \in h_1, h_2, \ldots, h_k} f_{X|C}(x_t|h')P(h')}$$

Where the optimal decision requrie choosing the class with highest posterior probability.

**Naive Bayes Model**

The Gaussian models requires computing a mean and a covariance matrix for each class. This can be done in a good way when the number of samples is larger than the number of dimensionality. If we aren't in this case, the estimates can be inaccurate, first of all for the covariance matrix.

At this point if we suppose that for each class all the features are independent we can simplify the estimate process.

$$f_{X|C}(x|c) \approx \prod_{j=1}^{D} f_{X_{[j]}|C}(x_{[j]}|c)$$

Where $x_{[j]}$ is the $j$-th component of $x$. The Naive Bayes assumption combined with Gaussian assumption models the distributions $f_{X_{[j]}|C}(x_{[j]}|c)$ as univariate Gaussians $f_{X_{[j]}|C}(x_{[j]}|c) = \mathcal{N}(x_{[j]}|\mu_{c,[j]}, \sigma^2_{c,[j]})$. Again we can compute the ML estimates as:

$$l(\theta) = \xi + \sum_{c=1}^{k} \sum_{i|c_i=c} \sum_{j=1}^{D} \log \mathcal{N}(x_{i,[j]}|\mu_{c,[j]}, \sigma^2_{c,[j]})$$

Again as we said, we can optimize the log-likelihood independently for each component, and also for each component we have the ML solutions that is:

$$\mu^{\star}_{c,[j]} = \frac{1}{N_c} \sum_{i|c_i=c} x_{i,[j]} \qquad \sigma^2_{c,[j]} = \frac{1}{N_c} \sum_{i|c_i=c} (x_{i,[j]} - \mu_{c,[j]})^2$$

So, it's possible to observe that the density for a sample $u$ can be expressed as:

$$f_{X|C}(x|c) = \prod_{j=1}^{D} \mathcal{N}(x_{[j]}|\mu_{c,[j]}, \sigma^2_{c,[j]}) = \mathcal{N}(x|\mu_c, \Sigma_c)$$

where:

$$\mu_c = \begin{bmatrix} \mu_{c,[1]} \\ \mu_{c,[2]} \\ \vdots \\ \mu_{c,[D]} \end{bmatrix} \Sigma_c = \begin{bmatrix} \sigma^2_{c,[1]} & 0 & ... & 0 \\ 0 & \sigma^2_{c,[2]} & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \sigma^2_{c,[D]} \end{bmatrix}$$

In conclusion we can say that the Naive Bayes Classifier corresponds to a Multivariate Gaussian classifier with diagonal covariance matrices.

### Tied Covariance Model

Another common Gaussian model assumes that the covariance matrices of the different classes are tied. This means that we have a single shared covariance matrix but different means, one for each class. Samples $x_{\{c,i\}}$ are obtained by sum $\mu_c$ and a noise $\epsilon_i$. Where $\epsilon_i \sim \mathcal{N}(0, \Lambda^{-1})$. Again we can estimate the parameters using the ML framework.

$$\mu^{\star}_c = \frac{1}{N_c} \sum_{i|c_i=c} x_i \qquad \Sigma^{\star} = \frac{1}{N} \sum_c \sum_{i|c_i=c} (x_i - \mu^{\star}_c)(x_i - \mu^{\star}_c)^T$$

Where N is the number of samples.

The model, also known as Quadratic Discriminant Analysis, is closely related to LDA, because it assumes that all classes have the same within class covariance.

### Pratical consideration for Gaussian CLassifier

- If data is high-dimensional, PCA can simplify the estimation
- PCA also allows removing dimension with very small variance
- Multivariate model performs better if we have enough data to re-liably estimate the covariance matrix
- Naive Bayes can simplify the estimation, but may perform poorly if data are high correlated
- Tied covariance models can capture correlations but can perfrom poor when classes have very different distributions
- If a gaussian model is not adequate for our data we can use different distributions that are more appropriate
- Alternatively the Gaussian model may still be effective for transformed data

**Models for discrete values for one categorical attribute**

We now consider a problem characterized by discrete features, and just for the moment we assume that we have a single categorical feature $x \in \{1...m\}$. For example we want to predict a cat gender from the fur color. In this case we have $x \in \{white,\ red,\ black, ...\}$ and we have also a set o labeled data $D = \{(x_1, c_1) ... (x_n, c_n)\}$. We also assume that samples are i.i.d. (as in the Gaussian case) and we want to compute $P(X_t = x_t|C_t = c_t) = \pi_{c,x_t}$ . In addition for a specific class $c$ we have that $\pi_c = (\pi_{c,1}, ..., \pi_{c,m})$ and the condition that $\sum_{i=1}^{m} \pi_{c,i} = 1$.

Again we can adopt a frequentist approach and estimate the Maximum Likelihood solution for $\Pi = (\pi_1, ...\pi_k)$ where $k$ is the number of classes and so in this example will be 2.

The Likelihood function for our data can be expressed as:

$$\mathcal{L}(\Pi) = \prod_{i=i}^{n} P(X_i = x_i|C_i = c_i)P(C_i = c_i)$$

where $c_i \in \{1...k\}$ is the class of the $i$-th sample.

For our example if the dataset is: (black,male)(red,male)(white,female)... etc, the likelihood function will be:

$$\mathcal{L}(\Pi) = P(X_1 = black|C_1 = male)P(C_1 = male) \times P(X_2 = red|C_2 = male)P(C_2 = male) \times P(X_3 = red|C_3 = female)P(C_3 =$$

At this point the Log-Likelihood function is given by:

$$l(\Pi) = \sum_{i=1}^{n} \log P(X_i = x_i|C_i = c_i) + \xi$$

using the equation $P(X_t = x_t|C_t = c_t) = \pi_{c,x_t}$ we obtein:

$$l(\Pi) = \sum_{i=1}^{n} \log \pi_{c_i,x_i} + \xi = \sum_{c=1}^{k} \sum_{i|c_i=c} \log \pi_{c,x_i} + \xi = \sum_{c=1}^{k} l_c(\pi_c) + \xi$$

where $l_c(\pi_c) = \sum_{i|c_i=c} \log \pi_{c,x_i}$.

Again the log likelihood function can be expressed as a sum of terms, each depending on a separate subset of parameters and for this reason we can independently optimize the terms. In addition we have

$$l_c(\pi_c) = \sum_{i|c_i=c} \log \pi_{c,x_i} = \log \pi_{female,white} + \log \pi_{male,red} + \log \pi_{male,white} + ... = \sum_{j=1}^{m} N_{c,j} \log \pi_{c,j}$$

Where $N_{c,j}$ is the number of times that we observed $x_i = j$ for the class $c$. Maximize this function is a bit more complex that the Gaussian case because now we have a constraint: $\sum_{i=1}^{m} \pi_{c,j} = 1$. A solution can be obtained by means of Lagrange multipliers (after some calculus we obtain):

$$\pi_i = \frac{N_i}{N}$$

So the Maximum Likelihood solution for each class is:

$$\pi^{\star}_{c,i} = \frac{N_{c,i}}{N_c}$$

**Model for discrete values with more than one categorical attribute**

If we have more than one categorical attribute, we may model their joint probability as a categorical R.V. with values given by all possible combinations of the attributes, but this is equivalent to do the cartesian product of all the possible combinations. To avoid that we can adopt again a Naive Bayes approximation and assume that the features are independent. For example if we consider just 2 attributes like the fur color and the eye color we obtaain:

$$P(X_t = [black, blue] | male) = P(black|male)P(blue|male)$$

And in general we obtain:

$$P(X_t = x_t | C_t = c_t) = \prod_j \pi^j_{c,x_t,[j]}$$

where $j$ denotes the feature index that in our example can be only 0 for fur and 1 for eyes.

We now consider an extended version of the problem where features represent occurrences of events. We may, for example, represent a text in terms of the words that appear inside. To create a model for this case we can represent documents in terms of occurrences of single words ignoring the order in which words appear. Thus, we have features vectors $x = (x_{[1]}, ...x_{[m]})$ where each $x_{[i]}$ represents the number of times we observed word $i$ in the document.

We have seen that occurrences can be modeled by multinomial distributions and so for each class $c$ we have $\pi_c = (\pi_{c,1}...\pi_{c,m})$ that represents the probability of observing a single instance of word $i$.

The probability for feature vector $x$ is given by the multinomial density:

$$P(X = x | C = c) = \frac{(\sum_{j=1}^m x_{[j]})!}{\prod_{j=1}^m x_{[j]}!} \prod_{j=1}^m \pi_{c,j}^{x_{[j]}} \propto \prod_{j=1}^m \pi_{c,j}^{x_{[j]}}$$

Again we can write the likelihood of the model, as in the previous case, the likelihood factorizes over classes, thus:

$$l(\Pi) = \sum_{c=1}^k l_c(\pi_c) + \xi \quad where \quad l_c(\pi_c) = \sum_{i|c_i=c} \sum_{j=1}^m x_{i,[j]} \log \pi_{c,j}$$

Note that $\xi$ is a costant and so we did not write it.

The first sum: $\sum_{i|c_i=c} x_{i,[j]}$ can be separated from the other and it represent the number of times that we saw the $j$-th word in all the documents of class $c$. So the log-likelihood becames:

$$l_c(\pi_c) = \sum_{j=1}^m N_{c,j} \log \pi_{c,j}$$

To solve this log-likelihood function we notice that it has exactly the same form as the on solved in the previous case and so the Maximum Likelihood is again:

$$\pi_{c,j} = \frac{N_{c,j}}{N_c}$$

Where $N_c$ is the total number of words for class c.

For a two class problem we write the log-likelihood ratio as:

$$llr(x) = \log \frac{P(X = x|C = h_1)}{P(X = x|C = h_0)} = \sum_{j=1}^{m} x_{[j]} \log \pi_{h_1,j} - \sum_{j=1}^{m} x_{[j]} \log \pi_{h_0,j} = x^T b$$

And again we have a linear decision function.

### Pratical considerations

- Rare words can cause problems: if a word does not appear in a topic we will estimate a probability $\pi_{c,j} = 0$. Any test sample that contains the word will have 0 probability of being from class $c$.
- We can mitigate the issue introducing pseudo-counts, i.e. assuming that each topic contains a sample were all words appear a (fixed) number of times. In practice, we can add a fixed values to the class occurrences $N_c$ before computing the ML solution.

Some additional comments on the discrete models for categorical events:

- Alternatively, we can model the dataset in terms of occurrences of events.
- The 2 models seen before are equivalent.
- The corresponding likelihoods functions are proportional.
- Maximum likelihood estimates will be the same.
- Inference will be the same.
- Bayesian posterior probabilities for model parameters will be the same.

## Logistic Regression

### Discriminative Linear Models

Logistic regression, despite its name, is a discriminative approach for classification. Rather than modeling the distribution of observed samples $X|C$, we directly model the class posterior distribution $C|X$. We need to define a model for the class posterior distribution $P(C = c|X = x)$.

### Logistic Regression for Binary Case

Considering a binary problem (so a 2 class problem) we have seen that the Gaussian model with tied covariances provides log-likelihood ratios that are linear functions of our data:

$$l(x) = \log \frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)} = w^T x + c$$

And the posterior log-likelihood ratio is:

$$\log \frac{P(C = h_1|x)}{P(C = h_0|x)} = \log \frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)} + \log \frac{\pi}{1 - \pi} = w^T x + b$$

The prior information has been absorbed into b.

Given $w, b$ we can compute the expression for the posterior class probability:

$$P(C = h_1|x, w, b) = \frac{1}{1 + e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Where $\sigma(z) = \frac{1}{1+e^{-z}}$ is called sigmoid function or logistic function.

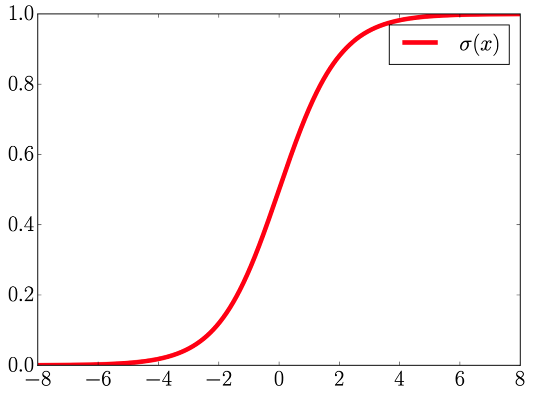


Figure 2: Sigmoid Fucntion

Some proprieties are:

- $1 - \sigma(x) = \sigma(-x)$
- $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

The model assumes that decision rules are linear surfaces (hyperplanes) orthogonal to $w$. The model parameters are $(w, b)$. If we know $(w, b)$ we can calculate the predictive distribtion.

In the following we will se how we can compute an estimates for $(w, b)$ using a frequentiest approach, in other words, using a set of training samples.

We assume that we have a labeled dataset $\mathcal{D} = [(x_1, c_1), ...(x_n, c_n)]$ and that classes are independently distribuited as $C_i|x_i, w, b \sim C|x_i, w, b$. Thanks to this the class-posterior model allows expressing the likelihood as:

$$P(C_1 = c_1, ...C_n = c_n|x_1, ...x_n, w, b) = \prod_{i=1}^{n} P(C_i = c_i|x_i, w, b)$$

And we can apply a Maximum Likelihood approach to estimate the model parameters. We will also consider from now that class $h_1$ will be 1 and class $h_0$ will be 0; so:

$$y_i = P(C_i = h_1|x_i, w, b) = P(C_i = 1|x_i, w, b) = \sigma(w^T x_i + b)P(C_i = 0|x_i, w, b) = 1 - y_i = \sigma(-[w^T x_i + b])$$

We can see that $C_i|x_i, w, b$ is a R.V. that follow the Bernoulli Distribution. For this reason the likelihood is:

$$\mathcal{L}(w, b) = P(C_1 = c_1, ...C_n = c_n|x_1, ...x_n, w, b) = \; = \prod_{i=1}^{n} P(C_i = c_i|x_i, w, b) = \prod_{i} y_i^{c_i}(1 - y_i)^{(1-c_i)}$$

But again it's better to work with the log-likelihood version:

$$l(w,b) = \log \mathcal{L}(w,b) = \sum_{i=1}^{n}[c_i \log y_i + (1-c_i)\log(1-y_i)]$$

And again our goal is to maximize $l$. The Maximum Likelihood solutions is also the solutions that minimizes the average cross-entropy between the distribution of observed labels and predicted labels. Rather than maximizing $l$ we can minimize:

$$J(w,b) = -l(w,b) = \sum_{i=1}^{n} -[c_i \log y_i + (1-c_i)\log(1-y_i)]$$

Where the expression $H(c_i, y_i) = -[c_i \log y_i + (1-c_i)\log(1-y_i)]$ represent the binary class entropy between the distribution of observed labels and predicted labels for the $i$-th sample. In general the cross entropy between 2 distributions is defined as:

$$H(P,Q) = -\mathbb{E}_{P(x)}[\log Q(x)] \quad continuous H(P,Q) = -\sum_{x \in Support} P(x)\log Q(x) \quad discrete$$

Where, $Q$, in our case is the distribution for the predicted labels according to our recognizer $\mathcal{R}$. So the cross entropy can be interpreted as a measure of the difference between $P$ and $Q$ and for this reason is minimal when $P = Q$. Minimization the average cross entropy means we are looking for a label distribution that is as similar as possible to the empirical one.

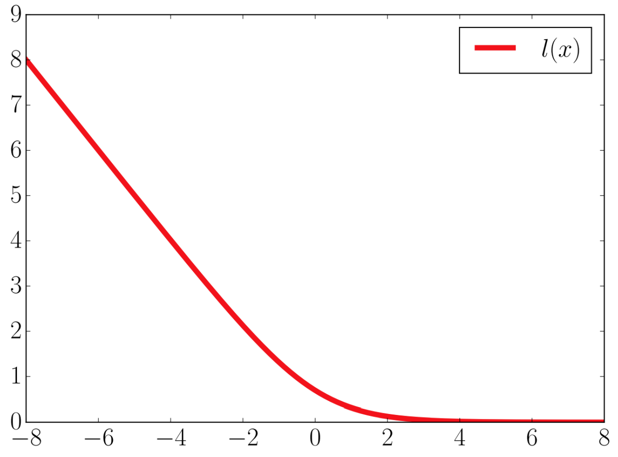
Another interesting thing can be obteined by writing the cross entropy as $z_i = 2c_i - 1$. In addition let $s_i = w^T x_i + b$ . We can rewrite $H$in terms of $s_i$ and $z_i$.

$$H(c_i, y_i) = -\log \sigma(z_i s_i) = -\log \sigma(z_i(w^T x_i + b)) = \log(1 + e^{-z_i(w^T x_i + b)})$$

The objective function, that we want to minimize, can be rewritten as:

$$J(w,b) = \sum_{i=1}^{n} H(c_i, y_i) = \sum_{i=1}^{n} \log(1 + e^{-z_i(w^T x_i + b)}) = \sum_{i=1}^{n} l(z_i(w^T x_i + b))$$

Where $l(x) = \log(1 + e^{-x})$ is the logistic loss function.



We can interpret the function as the cost of the prediction made with model $(w, b)$ for each sample. In addition $s_i$ is related to the distance of the sample $x_i$ from the separating surface. The cost we pay for each sample is $l\,(s_i z_i)$:

- If the predction is correct: $z_i = 1$ and $s_i > 0$ or $z_i = -1$ and $s_i < 0$. Then $s_i z_i > 0$ and we pay a low cost.
- If the predction is incorrect: $z_i = 1$ and $s_i < 0$ or $z_i = -1$ and $s_i > 0$. Then $s_i z_i < 0$ and we pay a cost that increase linearly with $s_i$.

### Regularized Logistic Regression for Binary Case

If classes are linearly separable, the logistic regression solution is not defined because we can make the values $s_i$ arbitrarily high by simply increasing the norm of $w$. As we increase $||w||$ the loss becames lower and the functions does not have a minimum. To make the problem solvable we can introduce a norm penalty, so the objective function that we will minimize is:

$$R(w, b) = \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-z_i(w^T x_i + b)})$$

Where $\lambda$ is an hyper-parameter and should be selected, using methods as cross-validation, to optimize the performance of the classifier.

Regularization allows reducing risk of over-fitting the training data. If $\lambda$ is too large the model will not be able to well separate the classes, if $\lambda$ is too small the model may have poor classification accuracy for unseen data, so poor generalization.

### Pratical Considerations

- The non-regularized model is invariant to linear transformations.
- The regularized version of the model isnot invariant.
- It is therefore useful, in some cases, to pre-process data.
- Cross-validation can help in identifying good pre-processing strategies. Common pre-processing strategies are:
  - Center the data.
  - Standardize variances (divide each feature by its own standard deviation).
  - Normalize variances while making features uncorrelated.
  - Classical normalization dividing each by norm, often after centering.
- We can simulate different empirical priors $\pi_T$ using a prior-weighted version of the model. This can be useful when aour classes are not balanced in terms of samples.
- If our application is characterized by the same effective prior $\pi_T$, the optimal decision should correspond to $w^T x + b \lesseqgtr 0$.
- In some cases, the score $s$ may not provide the correct probabilistic interpretation, and optimal decisions may require either recalibration of the scores or selection of an optimal threshold based on a validation set.

### Logistic Regression for Multiclass Problems

We now consider a problem with $K$classes, labeled from 1 to $K$. If we start again from the form of the posterior likelihood ratios of the linear Gaussian Classifier with uniform priors

$$\log \frac{P(C = j|x)}{P(C = r|x)} = (w_i - w_r)^T x + (b_j - b_r)$$

Given $W, b$ we can compute the probability of each class:

$$W = [w_1, ..., w_K] \quad b = \begin{bmatrix} b_1 \\ .. \\ b_K \end{bmatrix} \quad P(C = k|W, b, x) = \frac{e^{w_k^T x + b_k}}{\sum_j e^{w_j^T x + b_j}}$$

Where if we consider the sample $x_i$, its class posterior distribution is a Categorical Distribution:

$$C_i|W, b, x \sim Cat(y_i) \qquad y_{ik} = \frac{e^{w_k^T x + b_k}}{\sum_j e^{w_j^T x + b_j}}$$

As for the binary case we can express the log-likelihood as:

$$l(W, b) = \log \mathcal{L}(W, b) = \sum_{i=1}^{n} \log P(C_i = c_i|X_i = x_i, W, b)$$

But we can express the log-likelihood in a better way. For first re-write the categorical density as:

$$\log P(C_i = c_i|X_i = x_i, W, b) = \sum_{k=1}^{K} z_{ik} \log y_{ik}$$

Where $z_i$ is a vector that has all component equal to 0, except for the index $c_i$ wich is equal to 1.

$$z_i = [0...0, 1, 0...0] \qquad z_{ik} = \begin{cases} 1 & if \quad c_i = k \\ 0 & otherwise \end{cases}$$

So the log-likelihood becames:

$$l(W, b) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log y_{ik}$$

As for the binary case we can consider the multi-class cross entropy for sample $x_i$ as:

$$H(z_i, y_i) = -\sum_{k=1}^{K} z_{ik} \log y_{ik}$$

And again the Maximum Likelihood solution that maximize the log-likelihood, minimize the cross-entropy:

$$\arg\max_{W,b} l(W, b) = \arg\min_{W,b} \sum_{i=1}^{n} H(z_i, y_i)$$

Compared to the binary case, the model is over-parametrized. In particular for a 2 class problem, if we use the binary model we will obtain $(w, b)$. A different result can be obtained using the multiclass model that will give us $(w_1, b_1), (w_2, b_2)$. Where is the difference? The difference is that if we subtract $(w_2, b_2)$ from both $(w_1, b_1), (w_2, b_2)$, we obtain exactly the same result of the binary model.

**Regularized Logistic Regression for Multiclass Case**

Also in multiclass case we can add a regularization term to reduce over-fitting.

$$R(W,b) = \Omega(W) + \frac{1}{n}J(W,b)$$

Where $\Omega(W) = \Omega(w_1 ... w_N) = \frac{1}{2}\sum_i ||w_i||^2 \times \lambda$. Where $\lambda$ represent the weights.

**Considerations**

Remember that for binary LR, we addumed linear separation surfaces:

$$\log \frac{P(C = h_1|x)}{P(C = h_0|x)} = w^T x + b$$

which has the same form as the Gaussian classifier with tied covariances. But for Gaussian classifier with non-tied covariances we have:

$$\log \frac{P(C = h_1|x)}{P(C = h_0|x)} = w^T Ax + b^T x + c$$

That is quadratic in $x$ and linear in $A$ and $b$. For this reason we can rewrite it as:

$$\left\langle xx^T, A \right\rangle + b^T x + c \quad where \quad \langle A, B \rangle = \sum_i \sum_j A_{ij} B_{ij}$$

We can further express $\left\langle xx^T, A \right\rangle$ as $\left\langle xx^T, A \right\rangle = \text{vec}(xx^T)^T \text{vec}(A)$. If we define:

$$\phi = \begin{bmatrix} \text{vec}(xx^T) \\ x \end{bmatrix} \quad w = \begin{bmatrix} \text{vec}(A) \\ b \end{bmatrix}$$

The posterior log-likelihood ratio can be expressed as: $s(x, w, c) = w^T \phi(x) + c$. Thanks to this transformation we will obtain a linear separation surface in the space defined by the mapping $\phi(x)$. Warning: The dimensionality of the expanded feature space can grow very quickly.

## Bayes decisions and Model Evaluation

To evaluate our model a first solution can be the accuracy:

$$accuracy = \frac{\#corrected\ classified\ samples}{\#total\ samples\ classified}$$

$$error\ rate = \frac{\#incorrected\ classified\ samples}{\#total\ samples\ classified} = 1 - accuracy$$

Accuracy can be miseleading if the classes are not balanced. Let consider a table called *confusion matrix,* defined in general as:

|  | Class $\mathcal{H}_F$ | Class $\mathcal{H}_T$ |
|---|---|---|
| Prediction $\mathcal{H}_F$ | True Negative | False Negative |
| Prediction $\mathcal{H}_T$ | False Positive | True Positive |

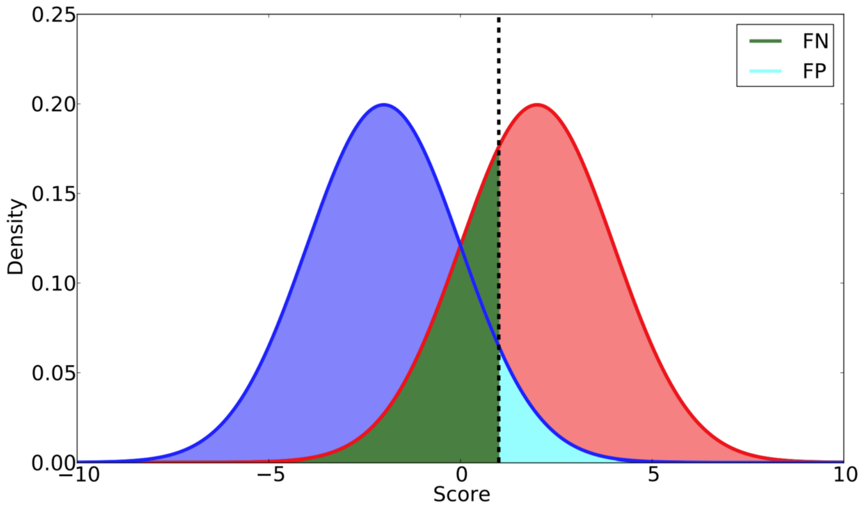
We can compute different measures:

- False Negative Rateo $\frac{FN}{FN+TP}$
- False Positive Rateo $\frac{FP}{FP+TN}$
- True Positive Rateo $\frac{TP}{FN+TP}$
- True Negative Rate $\frac{TN}{FP+TN}$

We can also compute the weighted accuracy as $acc = \alpha FPR + (1-\alpha)FNR$. The weight $\alpha$ measure how important are the different kind of errors. Up to now all the models give in output a score $s$ and the class assignment is performed by comparing the score to a threshold:

$$s \geq t \rightarrow \mathcal{H}_T$$
$$s < t \rightarrow \mathcal{H}_F$$

And different threshold correspond to different error rates. In the next figure is possible to see how the threshold influences the rateos and the class assignment.



### Bayes Decision

The goal of the classifier is to allow us to choose an action $a$ to perform among a set of actions $\mathcal{A}$. We can associate to each action a cost $C(a|k)$ that we have to pay when we choose an action $a$ and the sample belongs to class $k$. $C(a|k)$ represent the cost of labeling the sample as belonging to class $a$ when it actually belongs to class $k$. We can thus complete the expected cost of action $a$ when the posterior probability for each class is $P(C = k|x, R)$ and $R$ is the classifier.

$$\mathcal{C}_{x,R}(a) = \mathbb{E}[C(a|K)|x,R] = \sum_{k=1}^{K} \mathcal{C}(a|k)P(C=k|x,R)$$

The Bayes decision consists in choosing the action $a^*(x,R)$ that minimizes the function $C_{x,R}(a)$. We can observe that with this model also if the sample $x_t$ belongs to class $k_i$ (after probabilistic computation) but the expected cost function is lower with class $k_j$, the sample will be labeled with class $k_j$.

Let's now consider again a binary problem where we have a matrix like that:

| | Class $\mathcal{H}_F$ | Class $\mathcal{H}_T$ |
|---|---|---|
| Prediction $\mathcal{H}_F$ | $C(\mathcal{H}_F\|\mathcal{H}_F)$ | $C(\mathcal{H}_F\|\mathcal{H}_T)$ |
| Prediction $\mathcal{H}_T$ | $C(\mathcal{H}_T\|\mathcal{H}_F)$ | $C(\mathcal{H}_T\|\mathcal{H}_T)$ |

Whitout loss of generality we assume that:

$$C(\mathcal{H}_T|\mathcal{H}_T) = 0 \quad C(\mathcal{H}_F|\mathcal{H}_F) = 0$$

So the matrix becames:

| | Class $\mathcal{H}_F$ | Class $\mathcal{H}_T$ |
|---|---|---|
| Prediction $\mathcal{H}_F$ | 0 | $C(\mathcal{H}_F\|\mathcal{H}_T) = C_{fn}$ |
| Prediction $\mathcal{H}_T$ | $C(\mathcal{H}_T\|\mathcal{H}_F) = C_{fp}$ | 0 |

Where $C_{fn}$ is the cost of false negative errors and $C_{fp}$ is the cost of false positive errors. So the cost for predicting $\mathcal{H}_T$ is:

$$\mathcal{C}_{x,R}(\mathcal{H}_T) = C_{fp} \times P(\mathcal{H}_F|x,R) + 0 \times P(\mathcal{H}_T|x,R)$$

Also the cost for predicting $\mathcal{H}_F$ is:

$$\mathcal{C}_{x,R}(\mathcal{H}_F) = C_{fn} \times P(\mathcal{H}_T|x,R) + 0 \times P(\mathcal{H}_F|x,R)$$

In conclusion the optimal decision is the labeling that has lowest cost.

For binary problems the optimal decision can be expressed as:

$$a^*(x,\mathcal{R}) = \begin{cases} \mathcal{H}_T & if \quad C_{fp}P(\mathcal{H}_F|x,R) < C_{fn}P(\mathcal{H}_T|x,R) \\ \mathcal{H}_F & if \quad C_{fp}P(\mathcal{H}_F|x,R) > C_{fn}P(\mathcal{H}_T|x,R) \end{cases}$$

and we can choose any action when the two costs are equal. Alternatively we can express the optimal decision as:

$$a^*(x, \mathcal{R}) = \begin{cases} \mathcal{H}_T & if \quad r(x) > 0 \\ \mathcal{H}_F & if \quad r(x) < 0 \end{cases} \qquad r(x) = \log \frac{C_{fn} P(\mathcal{H}_T|x, R)}{C_{fp} P(\mathcal{H}_F|x, R)}$$

If $R$ is a generative model for $x$ we can express $r$ in terms of costs, prior probabilities and conditional likelihoods as:

$$r(x) = \log \frac{\pi_T C_{fn}}{(1 - \pi_T) C_{fn})} \times \frac{f_{X|\mathcal{H},R}(x|\mathcal{H}_T)}{f_{X|\mathcal{H},R}(x|\mathcal{H}_F}$$

where $\pi_T = P(\mathcal{H} = \mathcal{H}_T)$ is the prior probability for class $\mathcal{H}_T$. Using this new version of $r(x)$ the decision rules becames:

$$r(x) \lesseqgtr 0 \Longleftrightarrow \log \frac{f_{X|\mathcal{H},R}(x|\mathcal{H}_T)}{f_{X|\mathcal{H},R}(x|\mathcal{H}_F)} \lesseqgtr -\log \frac{\pi_T C_{fn}}{(1 - \pi_T) C_{fn}}$$

The triplet $(\pi_T, C_{fn}, C_{fp})$ represent the working point of an application for a binary classification task. It's possible to show that the triplet is redundant, in sense that can be built an equivalent applications with the triplet: $(\pi', 1, 1)$.

Up to now we have considered hot to perform a classification for a sample $x$. Now we want to evaluate the goodness of our decisions taken, using the posterior distribution defined by classifier $R$. We can compute the cost of the Bayes decision taken with the recognizer $R$ as:

$$\mathcal{C}^*(x, R|c) = C(a^*(x, R)|c)$$

This represent the cost that i will pay to classify the sample $x$ using the optimal decision $a^*$ when the actual label is $c$.

### Empirical Bayes Risk

We would like to now how much we will pay if we use our classifier on the evaluation-set. To know that we have to define the Bayes Risk $\mathcal{B}$. The Bayes Risk is the expected value of the Bayes Cost of Bayes decision made by classifier $R$ over evaluation data sampled from $X, C|\varepsilon$. Where $X|C, \varepsilon$ is the conditional distribution of the evaluation population and $\varepsilon$ is the Evaluator.

$$\mathcal{B} = \mathbb{E}_{X,C|\varepsilon}[\mathcal{C}^*(x, R|c)] = \sum_{c=1}^{K} \pi_c \int \mathcal{C}^*(x, R|c) f_{X|C,\varepsilon}(x|c) dx$$

We have a problem: In general wi won't have access to $f_{X|C,\varepsilon}(x|c)$. However if we have a set of labeled evaluation samples $(x_1, c_1)...(x_N, c_N)$ we can approximate the expectations in this way:

$$\int \mathcal{C}^*(x, R|c) f_{X|C,\varepsilon}(x|c) dx \approx \frac{1}{N_c} \sum_{i|c_i=c} \mathcal{C}^*(x_i, R|c)$$

And so we can define the empirical Bayes Risk as:

$$\mathcal{B}_{emp} = \sum_{c=1}^{K} \frac{\pi_c}{N_c} \sum_{i|c_i=c} \mathcal{C}^*(x_i, R|c)$$

Considerations:

- A recognizer that has a lower cost will provide more accurate answers.
- We use $\mathcal{B}_{emp}$ to compare classifiers.
- The Bayes Risk measures the cost of our decision over the evaluation samples.

  To see a possible example, go to page 26 of slides called BayesDecisionModelEvaluation.

**Empirical Bayes Risk For Binary Problems**

In a binary problems we have seen how a misclassification can costs $C_{fp}$ if we have a false positive or $C_{fn}$ if we have a false negative. However the triplet $(\pi_T, C_{fn}, C_{fp})$ depends from the application. Taken a sample $x_i$ with predicted label $c_i^*$ the empirical Bayes Risk is:

$$\mathcal{B}_{emp} = \frac{\pi_T}{N_T} \sum_{i|c_i=\mathcal{H}_T} \mathcal{C}^*(x_i, R|\mathcal{H}_T) + \frac{1-\pi_T}{N_F} \sum_{i|c_i=\mathcal{H}_F} \mathcal{C}^*(x_i, R|\mathcal{H}_F) =$$

$$= \pi_T C_{fn} \frac{\sum_{i|c_i=\mathcal{H}_T, c_i^*=\mathcal{H}_F}}{N_T} + (1-\pi_T) C_{fp} \frac{\sum_{i|c_i=\mathcal{H}_F, c_i^*=\mathcal{H}_T}}{N_F} =$$

$$= \pi_T C_{fn} P_{fn} + (1-\pi_T) C_{fp} P_{fp} = DCF_u(C_{fn}, C_{fp}, \pi_T)$$

Where DCF is the Detection Cost Function and $P_{fn}, P_{fp}$ are the false negative rate and false positive rate.

We can define the Normalized DCF as:

$$DCF(C_{fn}, C_{fp}, \pi_T) = \frac{DCF_u(C_{fn}, C_{fp}, \pi_T)}{\min(\pi_T C_{fn}, (1-\pi_T) Cfp)}$$

Comparing the $DCF$ with 1 we can have that:

- $> 1$ is better if we try to guess.
- $= 1$ the model is useless.
- $< 1$ the model can be useful, but depends from other factors.

In addition the Normalized $DCF$ is invariant to scaling and this means that for our applications we can use a triplet like $(\pi', 1, 1)$ that is equivalent to $(\pi_T, C_{fn}, C_{fp})$, with opportune operations.

For a given application is possible to define the minimum cost, $DCF_{min}$, corresponding to the use of the optimal threshold for the evaluation set. We can obtain it, varying the threshold $t$ to obtain all possible combinations of $P_{fn}, P_{fp}$ and selecting the $t$ corresponding to the lowest $DCF$.

In general to reduce misclassification cost we can use a validation set to fine a (close-to) optimal threshold for a given application, or, we can transform the classifier scores $s$ in a way that is as much as possible independent from the target application.

## Support Vector Machines

We have seen how, in a binary problem, the logistic regression provides a linear classification rule that try to maximize the cross-entropy among class or also try to minimize the logistic loss. As we seen it's often useful to add a regularization term:

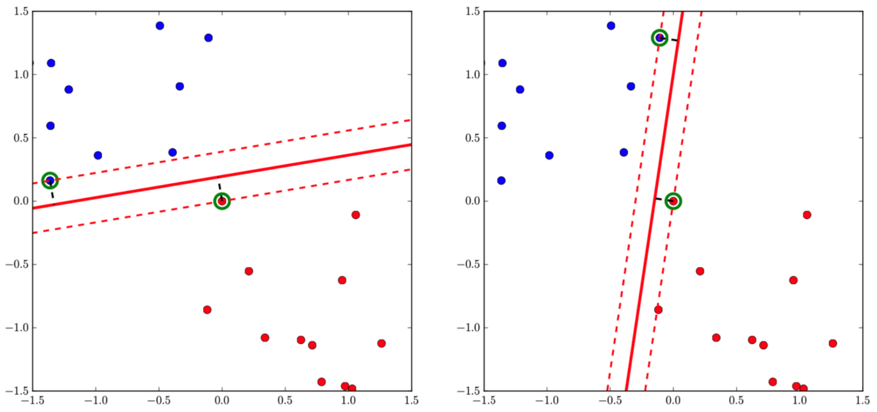
$$w^*, b = \arg\min_{w,b} \frac{\lambda}{2} ||w||^2 + \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-z_i(w^T x_i + b)})$$

Where $z_i$ is the class label for sample $x_i$ encoded as:

$$z_i = \begin{cases} +1 & if \quad c_i = \mathcal{H}_T \\ -1 & if \quad c_i = \mathcal{H}_F \end{cases}$$

We now consider Support Vector Machines (SVG) that provide a natural way to achive non-linear separation without the need for an explicit expansion of features, so without using $\Phi(...)$ in LR.

Assume that we have two classes that are linearly separable. There is an infinite number of planes of separating hyperplanes. With SVM we can select the hyperplane that separates the 2 classes with the largest margin.



The margin is defined as the distance of the closest set of points (one for each class) with the separation hyperplane. Let be $f(x) = w^T + b$ the function representing the separation surface. The distance of $x_i$ from the hyperplane is:

$$d(x_i) = \frac{|f(x_i)|}{||w||} = \frac{|z_i(w^T x_i + b)|}{||w||}$$

And since the class are separable we have that:

$$\begin{cases} f(x_i) > 0 & if \quad c_i = \mathcal{H}_T \\ f(x_i) < 0 & if \quad c_i = \mathcal{H}_F \end{cases}$$

The maximum margin hyperplane is the hyperplane that maximizes the minimum distance of all points from the hyperplane:

$$w^*, b^* = \arg\max_{w,b} \min_{i \in (1...n)} d(x_i) = \arg\max_{w,b} \min_{i \in (1...n)} \frac{|z_i(w^T x_i + b)|}{||w||}$$

With the constraint that: $z_i(w^T x_i + b) > 0$.

This is the formal definition, let's see now the operative formula.

1. We can drop the constraint of the absolute values because for values $w, b$ which can correctly separate the classes we have $z_i(w^T + b) > 0$ for all samples.

2. Given that $||w||$ doesn't depend by $i$, we can remove it from the min term. So we will have:

$$w^*, b^* = \arg\max_{w,b} \frac{1}{||w||} \min_{i\in(1...n)} z_i(w^T x_i + b)$$

3. The objective function is invariant under re-scaling of the parameters, so:

$$\frac{1}{||w||} \min_{i\in(1...n)} [z_i(w^T x_i + b)] = \frac{1}{||\alpha w||} \min_{i\in(1...n)} [z_i(\alpha w^T x_i + b)]$$

4. We restrict our problem to solutions for which

$$\min_{i\in(1...n)} [z_i(w^T x_k + b)] = 1$$

   So we introduce this new constraint.
5. The problem in (2) becomes equivalent to

$$\arg\min_{w,b} \frac{1}{2}||w||^2$$

$$s.t \begin{cases} z_i(w^T x_i + b) >= 1, & amp; \quad amp; i = 1...n \\ \min_i z_i(w^T x_k + b) = 1 \end{cases}$$

6. Finally we can drop the last constraint and solve the problem:

$$\arg\min_{w,b} \frac{1}{2}||w||^2$$

$$s.t = z_i(w^T x_i + b) >= 1, \quad i = 1...n$$

   Because an optional solution will automatically satisfy $\min_i z_i(w^T x_i + b) = 1$.

To solve the SVM problem we can consider a Lagrangian formulation of the problem, so we introduce the Lagrange multiplier $\alpha_i >= 0$.

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i [z_i(w^T x_i + b) - 1]$$

The optimal solution is obtained by minimizing $L$ while requiring that the derivates $\alpha_i$ vanish. We can equivalently maximize $L$ requiring that the derivatives $w, b$ vanish. So, setting the derivative of $L$ w.r.t $w, b$ equal to 0, we obtain:

$$w = \sum_{i=1}^{n} \alpha_i z_i x_i \qquad 0 = \sum_{i=1}^{n} \alpha_i z_i$$

Replacing these constraints in $L$ we obtain:

$$\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$$

$$s.t = \begin{cases} \alpha_i \geq 0 & i = 1, ..., n \\ \sum_{i=1}^{n} \alpha_i z_i = 0 \end{cases}$$

It's possible to rewrite it in matrix form as:

$$L_D(\alpha) = \alpha^T 1 - \frac{1}{2}\alpha^T H \alpha$$

Where $H$ is a matrix defined as: $H_{ij} = z_i z_j x_i^T x_j$.

The optimal solution satisfies the Karush-Kuhn-Tucker conditions:

- For all points that have $z_i(w^T x_i + b) > 1$, i.e., points that aren't on the margin of the hyperplane, the corresponding Lagrange multiplier $\alpha_i = 0$.
- For all the points that are on the margin, the Lagrange multiplier $\alpha_i \neq 0$. This points are called Support Vectors.

With these considerations we can say that: The training points that aren't Support Vectors don't affect the separation surface because the hyperplane depends only from points that are Support Vectors.

To define a score for a test point $x_t$ we need to calculate:

$$s(x_t) = w^T x_t + b = \sum_{i=1}^{n} \alpha_i z_i x_i^T x_t + b$$

Where $x_i$ are the training samples and $x_t$ the test sample.

**Non linearly separable problem**

If we consider a problem where the classes are not linearly separable, no matter the value of $w$, some points will violate the constraint $z_i(w^T x_i + b) \geq 1$. A possible solution can be to minimize the number of points that violate the constraint. To do this we introduce the slack variables $\xi_i \geq 0$ which represent how much a point is violating the constraint. So we can replace the constraint $z_i(w^T x_i + b) \geq 1$ with: $z_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$. In other words we allow training points to be inside the margin by a factor $\xi_i$.

The target is always to minimize the number of points that have $\xi_i > 0$. To do that we consider:

$$\Phi(\xi) = \sum_{i=1}^{n} \xi_i^{\sigma} \qquad \sigma > 0$$

For sufficiently small values of $\sigma$, $\Phi(\xi)$ represent the number of points inside the margin. If we remove this points the classes become linearly separable. All this, correspond to minimize the function:

$$\frac{1}{2}||w||^2 + CF(\sum_{i=1}^{n} \xi_i^{\sigma})$$

Where $F$ is a monotone convex function and $C$ is a constant. Unfortunately for small values of $\sigma$ the problem is difficult to solve. So now, we will consider $\sigma = 1$. The objective function becomes:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$$

$$s.t. = \begin{cases} z_i(w^T x_i + b) \geq 1 - \xi_i & \forall i = 1...n \\ \xi_i \geq 0 & \forall i = 1...n \end{cases}$$

- For points inside the margin and correct classified $0 < \xi < 1$.
- For points inside the margin but missclassified $\xi > 1$.
- For points on the margin $\xi = 1$.

As we did for the hard-margin SVM we can introduce the Lagrangian problem as:

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i[z_i(w^T x + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i$$

Where $\mu_i \ \geq 0$ and $\alpha_i \geq 0$ are an additional set of Lagrange multipliers relative to the constraint $\xi_i \geq 0$. We can, using the KKT conditions to optimize the Lagrangian equation saw before to obtain:

$$\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$$

With the constraints: $0 \leq \alpha_i \leq C \quad \forall \quad i = 1...n$ and $\sum_{i=1}^{n} \alpha_i z_i = 0$. This problem is called *dual problem.*

Note that the problem is similar to the hard-margin SVM and again the result depends only from the support vectors.

**Solve the primal problem**

The primal problem was:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$$

$$s.t. = \begin{cases} z_i(w^T x_i + b) \geq 1 - \xi_i & \forall i = 1...n \\ \xi_i \geq 0 & \forall i = 1...n \end{cases}$$

Thus the problem can be re-written as:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max[0, 1 - z_i(w^T x_i + b)]$$

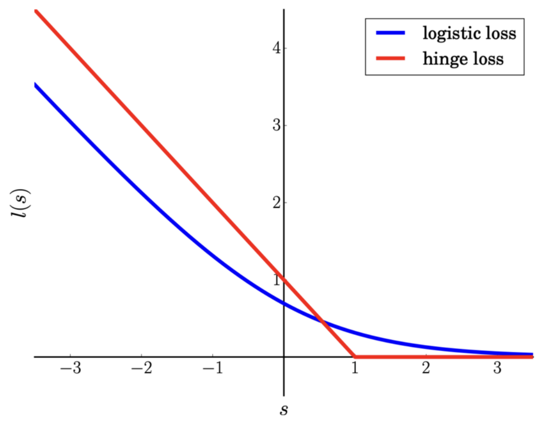
Where the constraints have been absorbed into the loss terms. As wee can see this formula is very similar to the Logistic Regression formula. Also the Loss function is similar, as it's possible to see from the following image.

LR: Logistic Loss

$$l(s) = \log\left(1 + e^{-s}\right)$$

SVM: Hinge Loss

$$l(s) = \max(0, 1 - s)$$



**Comparison between Primal and Dual**

We have seen that we can obtain the separation hyperplane by solving either the primal or the dual problem, but what are the differences?

For the primal we have that:

- Minimize $w$ and $b$
- $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$
- Scoring complexity is $O(D)$
- Embedding a non linear transformation it's very difficult because the dimension of the space can grow very quickly

For the dual we have that:

- Minimize $\alpha$
- $\alpha \in \mathbb{R}^N$
- Scoring complexity is $O(SV)$, so depends from the number of samples
- Embedding a non linear transformation require only to calculate the dot products in the expanded space $\Phi(x_i)^T\Phi(x_j)$

The function that calculate the dot product is called Kernel Function and is represented as:

$$k(x_1, x_2) = \Phi(x_1)^T\Phi(x_2)$$

The $k$ function allows training SVM in a large (even infinite) dimensional Hilbert space. Let's see some possibilities:

- We can define the polynomial kernels of degree $d$ as: $k\left(x_1, x_2\right) = \left(x_1^T x_2 + 1\right)^d$
- Another example is: $k\left(x_1, x_2\right) = e^{-\gamma||x_1 - x_2||^2}$. If $\gamma$ is small the kernel is wide, else, if $\gamma$ is large the kernel is narrow.

**Practical considerations**

- We need to take care in selecting the appropriate kernel
- In some cases linear classifier are sufficient
- Feature pre-processing can be relevant
- SVM scores have no probabilistic interpretation

## Gaussian Mixture Models

We have seen how a Gaussian classifier is a model that assumes that class-conditional distributions are Gaussian. In many cases, this assumption can be inaccurate. Gaussian Mixture Models are an alternative to model a generic distribution. In general a Gaussian Mixture Model is a density model obtained as a weighted combination of Gaussians:

$$X \sim GMM(M, \Sigma, \Pi) \Rightarrow f_X(x) = \sum_{c=1}^{K} w_c \mathcal{N}(x; \mu_c, \Sigma_c)$$

Where $K$ is a model parameter, $M = [\mu_1, ..., \mu_K]$, $S = [\Sigma_1, ..., \Sigma_K]$ and $w = [w_1, ..., w_K]$. In addition remember that for a density the integral must be equal to 1. This imply that the sum of all weights must be 1.

Given a dataset $D = [x_1, ...x_n]$ that we will consider as unlabeled, we can use the Maximum Likelihood to estimate the model parameters of the GMM that best describes the dataset $D$. We can write the likelihood and log-likelihood as:

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} f_{X_i}(x_i) = \prod_{i=1}^{n} \left( \sum_{c=1}^{K} w_c \mathcal{N}(x; \mu_c, \Sigma_c) \right)$$

$$l(\theta) = \log \mathcal{L}(\theta) = \sum_{i=1}^{n} \log \left( \sum_{c=1}^{K} w_c \mathcal{N}(x; \mu_c, \Sigma_c) \right)$$

So we can say that a GMM can be interpreted as the marginal of a joint distribution of data points and corresponding clusters. If we knew the component responsible for each sample ( the cluster label) we could estimate the parameters of each Gaussian by Maximum Likelihood from the points of each cluster. Unfortunately, clusters are unknown, so we have to estimate first the cluster label. How to do that?

Let's consider a set of GMM parameters that someone gives us. We have $\theta = (M, S, w)$ and we know that the density for a sample $x_i$ and component $c$ is: $f_{X_i,C_i}(x_i, c) = w_c \mathcal{N}(x; \mu_c, \Sigma_c)$. We can compute the cluster posterior probabilities as:

$$\gamma_{c,i} = P(C_i = c | X_i = x_i) = \frac{f_{X_i,C_i}(x_i, c)}{f_{X_i}(x_i)} = \frac{w_c \mathcal{N}(x_i; \mu_c, \Sigma_c)}{\sum_{c'} w_{c'} \mathcal{N}(x_i; \mu_{c'}, \Sigma_{c'})}$$

Where $\gamma_{c,i}$ are called *responsibilities.* As a first approach we can decide to assign a point to the cluster $c$ with the highest posterior probabilities, so with the highest responsibility. So, for each sample we will have $c_i^* = \arg\max_c P(C_i = c | X_i = x_i)$. After have define all the cluster labels we can estimate by ML the new GMM parameters $\theta^{new} = (M^{new}, S^{new}, w^{new})$. The loglikelihood becames:

$$l(\theta) = \sum_{i=1}^{n} [\log f_{X_i|C_i}(x_i | c_i^*) + \log P(C_i = c_i^*)] =$$

$$= \sum_{i=1}^{n}[\log \mathcal{N}(x_i; \mu_{c_i^*}, \Sigma_{c_i^*})] + \sum_{i=1}^{n}[\log w_{c_i^*}]$$

And so we have: $l(\theta) = l_{\mathcal{N}}(M, S) + l_C(w)$. In addition is possible to see that te first term corresponds to the log-likelihood of a (multivariate) Gaussian classification model where the class labels are assumed to be estimated $c_i^*$. The second term corresponds to the log-likelihood of a categorical model with parameters $w_c$. So the solutions are:

$$\mu_c^* = \frac{1}{N_c}\sum_{i|c_i^*=c} x_i \quad \Sigma_c^* = \frac{1}{N_c}\sum_{i|c_i^*=c}(x_i - \mu_{c^*})(x_i - \mu_{c^*})^T \quad w_c^* = \frac{N_c}{\sum_{c=1}^{K} N_c}$$

We could then obtain the updated set of model parameters $\theta^{new} = (M^*, S^*, w^*)$. We could also iterate the process stopping when some criterion is met, but as we will see this is not the best way. The problem of this method is that we create *hard-clusters*, where a point is assigned to one and only one cluster, but if a point has $P(C_i = c_1|X_i = x_i) \approx P(C_i = c_2|X_i = x_i)$, both $c_1$ and $c_2$ might have been responsible for the generation of $x_i$.

However, let's still consider hard clusters. Let's also assume that we fix the covariance matrix of our GMM to $\Sigma_c = I$ and $w_c = \frac{1}{K}$. In this case the cluster assignment corresponds to the rule:

$$c_i^* = \arg\max_c P(C_i = c|X_i = x_i) = \arg\min_c ||x_i - \mu_c||^2$$

So our algorithm will assign the point $x_i$ to the cluster $c_i^*$ with the closest centroid $\mu_{c_i^*}$. After that it will re-estimate the cluster centroids using the assigned points and it will iterate until convergence. This is the *K-Means clustering algorithm.*