Machine Learning and Pattern recognition Abstract

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 $P(x, C_k) = P(x|C_k)\,P(C_k)$ 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" $X = \{x_1, \ldots x_k\}$ 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 $p_1 \ldots . p_m$ 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).

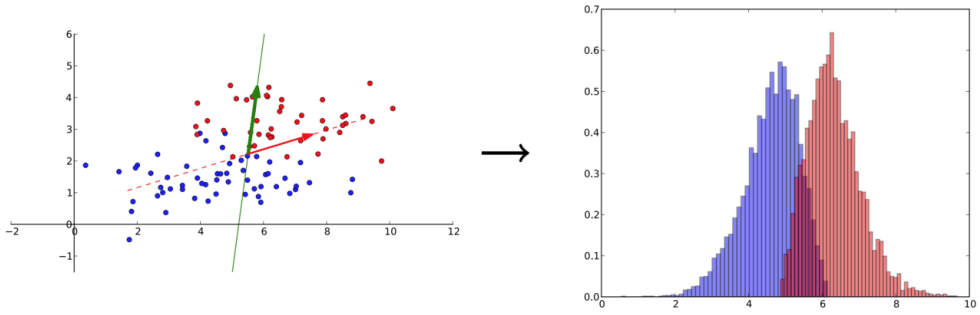


Figure 1: Red Line = PCA, Green Line = LDA

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, μ is the dataset mean and μ_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 μ and the class-mean μ_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:

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

Were $L(w) = \frac{s_B}{s_W} = \frac{w^T \ S_B \ w}{w^T \ S_W \ w} = \lambda(w)$. Note that the criterion does not depend on the scale of w, for this reason if w is a maximizer of L, then also αw 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:

- μ is the mean ($\mathbb{E}[X] = \mu$), and the distribution is symmetric and centered around it.
- σ 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 μ and covariance matrix Σ, it can be written as a linear transformation of Y and μ:

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

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, μ represents the mean of the data and Σ 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 π 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 π and a possible way to estimate a "good" value consist in looking for a value of π 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 π 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 π as the value that maximizes the likelihood function. To solve for π we can consider that:

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

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 $P\left(X_t = H \mid X_1 = H,\ \ldots.\ , X_n = T\right) = \pi_{ML}$

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 μ and varaince ν. 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 θ and also for μ and ν 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 θ.

Methods of Moments (MOM)

For the Gaussian distribution the first two moments are μ and ν. 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:

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

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:

- $\nu = \lambda^{-1}$
- ξ 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:

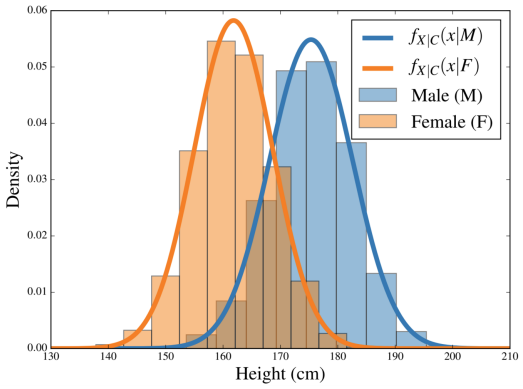
$$f_{X_t,C_t}(x_t, c) = f_{X,C}(x_t, c) = f_{X|C}(x_t|c)P_C(c)$$

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: μ and σ^2. To do that we can use the Maximum Likelihood parameters for both class:

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

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

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

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

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 θ , 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 θ 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 θ is :

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

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 μ_c and Σ_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 $\log r(x_t) \lessgtr 0$ or in another ways we can compare $llr\,(x_t) \lessgtr \log \frac{P(C=h_1)}{P(C=h_0)}$. 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:

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

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 μ_c and a noise ϵ_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