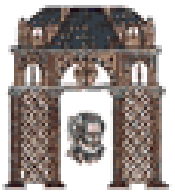
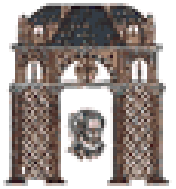


Αναγνώριση Προτύπων

Εκτίμηση Παραμέτρων
(Parameter Estimation – Examples)

Χριστόδουλος Χαμζάς

Τα περιεχόμενο της παρουσίασης βασίζεται στο βιβλίο: "Introduction to Pattern Recognition A Matlab Approach", S. Theodoridis, A. Pikrakis, K. Koutroubas, D. Caboyras, Academic Press, 2010



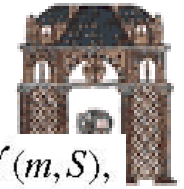
Εκτίμηση Παραμέτρων

Ένα πρόβλημα που συναντιέται συχνά στην πράξη είναι ότι οι pdfs που περιγράφουν τη στατιστική κατανομή των δεδομένων των κλασεων δεν είναι γνωστή και πρέπει να εκτιμηθεί με τη χρήση δεδομένων εκπαίδευσης.
Μια προσέγγιση σε αυτό το πρόβλημα εκτίμησης λειτουργία είναι να υποθέσουμε ότι ένα pdf έχει μια συγκεκριμένη μορφή, αλλά δεν γνωρίζουμε τις τιμές των παραμέτρων που την ορίζουν. Για παράδειγμα, μπορούμε να γνωρίζουμε ότι το pdf είναι Gaussian μορφής, αλλά δεν γνωρίζουμε την μέση τιμή και / ή τα στοιχεία του πίνακα συνδιακύμανσης Σ.

Η εκτίμηση μεγίστης πιθανοφάνειας (ML, Maximum Likekihood) θεωρεί τις τιμές των αγνώστων παραμέτρων «άγνωστες σταθερές» και προσπαθεί να βρεί ποιες είναι αυτές ώστε να μεγιστοποιείται η πιθανότητα να πάρουμε το συγκεκριμένο «δείγμα» τιμών.
Εστιάζοντας στην Gaussian PDF και αν υποθέσουμε ότι μας δίνονται N σημεία, $x_i \in R^D$, i = 1, 2,. . . , N, τα οποία είναι γνωστό ότι προέρχονται από μία κανονική κατανομή, οι εκτιμητές ML της άγνωστης μέσης τιμής και του πίνακα συνδιασποράς δίνονται από

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{ML})(x_i - \mu_{ML})^T$$

Συνήθως, αντί με N, το άθροισμα για τον υπολογισμό του πίνακα συνδιασποράς το διαιρούμε με (N-1), γιατί έτσι ο εκτιμητής είναι unbiased, διότι $\frac{N}{N-1}\hat{\Sigma} \xrightarrow{N\to\infty} \Sigma$



Example 1.4.2. Generate 50 2-dimensional feature vectors from a Gaussian distribution, $\mathcal{N}(m,S)$, where

$$m=[2,-2]^T,\ S=\begin{bmatrix}0.9 & 0.2\\ 0.2 & 0.3\end{bmatrix}$$

Let X be the resulting matrix, having the feature vectors as columns. Compute the ML estimate of the mean value, m, and the covariance matrix, S, of $\mathcal{N}(m,S)$ and comment on the resulting estimates.

Solution. To generate X, type

```
randn('seed',0)
m = [2 -2]; S = [0.9 0.2; 0.2 .3];
X = mvnrnd(m,S,50)';
```

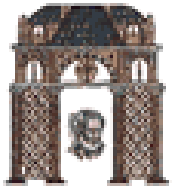
To compute the ML estimates of m and S, type

```
[m_hat, S_hat]=Gaussian_ML_estimate(X);
```

The results are

$$m\_hat=[2.0495,\ -1.9418]^T,\ S\_hat=\begin{bmatrix}0.8082 & 0.0885\\ 0.0885 & 0.2298\end{bmatrix}$$

It can be observed that the estimates that define the corresponding Gaussian pdf, although close to the true values of the parameters, cannot be trusted as good estimates. This is due to the fact that 50 points are not enough to result in reliable estimates. Note that the returned values depend on the initialization of the random generator (involved in function *mvnrnd*), so there is a slight deviation among experiments. ■

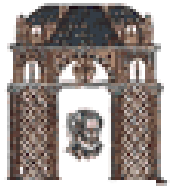


Κώδικας MatLab

```
function [m_hat,S_hat]=Gaussian_ML_estimate(X)
[l,N]=size(X);
m_hat=(1/N)*sum(X')';
S_hat=zeros(l);
for k=1:N
   S_hat=S_hat+(X(:,k)-m_hat)*(X(:,k)-m_hat)';
end
S_hat=(1/N)*S_hat;
```

Εάν τρέξουμε το πρόγραμμα για N=10000 παίρνουμε

$$\hat{\mu} = \begin{bmatrix} 2.0024 \\ -2.0014 \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} 0.9009 & 0.2046 \\ 0.2046 & 0.2046 \end{bmatrix}$$

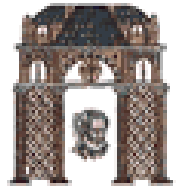


Example 1.4.3. Generate two data sets, X (training set) and X_1 (test set), each consisting of $N = 1000$ 3-dimensional vectors that stem from three *equiprobable* classes, ω_1, ω_2, and ω_3. The classes are modeled by Gaussian distributions with means $m_1 = [0, 0, 0]^T$, $m_2 = [1, 2, 2]^T$, and $m_3 = [3, 3, 4]^T$, respectively; their covariance matrices are

$$S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} = \sigma^2 I$$

1. Using X, compute the maximum likelihood estimates of the mean values and the covariance matrices of the distributions of the three classes. Since the covariance matrices are known to be the same, estimate them for each class and compute their average. Use the latter as the estimate of the (common) covariance matrix.

2. Use the Euclidean distance classifier to classify the points of X_1 based on the ML estimates computed before.
3. Use the Mahalanobis distance classifier to classify the points of X_1 based on the ML estimates computed before.
4. Use the Bayesian classifier to classify the points of X_1 based on the ML estimates computed before.
5. For each case, compute the error probability and compare the results (all classifiers should result in almost the same performance. Why?).



1.4 MINIMUM DISTANCE CLASSIFIERS

1.4.1 The Euclidean Distance Classifier

The optimal Bayesian classifier is significantly simplified under the following assumptions:

- The classes are equiprobable.
- The data in *all* classes follow Gaussian distributions.
- The covariance matrix is the *same* for all classes.
- The covariance matrix is diagonal and *all* elements across the diagonal are *equal*. That is, $S = \sigma^2 I$, where I is the identity matrix.

Under these assumptions, it turns out that the optimal Bayesian classifier is equivalent to the minimum Euclidean distance classifier. That is, given an unknown x, assign it to class ω_i if

$$||x - m_i|| \equiv \sqrt{(x - m_i)^T (x - m_i)} < ||x - m_j||, \quad \forall i \neq j$$

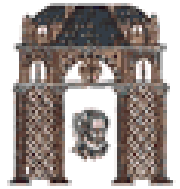
It must be stated that the Euclidean classifier is often used, even if we know that the previously stated assumptions are not valid, because of its simplicity. It assigns a pattern to the class whose mean is closest to it with respect to the Euclidean norm.

1.4.2 The Mahalanobis Distance Classifier

If one relaxes the assumptions required by the Euclidean classifier and removes the last one, the one requiring the covariance matrix to be diagonal and with equal elements, the optimal Bayesian classifier becomes equivalent to the minimum Mahalanobis distance classifier. That is, given an unknown x, it is assigned to class ω_i if

$$\sqrt{(x - m_i)^T S^{-1} (x - m_i)} < \sqrt{(x - m_j)^T S^{-1} (x - m_j)}, \quad \forall j \neq i$$

where S is the common covariance matrix. The presence of the covariance matrix accounts for the shape of the Gaussians [Theo 09, Section 2.4.2].



```
function [z]=euclidean_classifier(m,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N
   for j=1:c
      de(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));
   end
   [num,z(i)]=min(de);
end
```

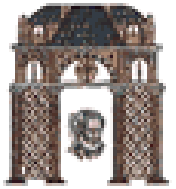
```
function [z] = mahalanobis_classifier(m,S,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N
   for j=1:c
      dm(j)=sqrt((X(:,i)-m(:,j))'*inv(S)*(X(:,i)-m(:,j)));
   end
   [num,z(i)]=min(dm);
end
```

```
function [z]=bayes_classifier(m,S,P,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N
   for j=1:c

t(j)=P(j)*comp_gauss_dens_val(m(:,j),S(:,:,j),X(:,i));
   end
   [num,z(i)]=max(t);
end
```



Solution. To generate X, use the function *generate_gauss_classes* by typing

```
m=[0 0 0; 1 2 2; 3 3 4]';
S1=0.8*eye(3);
S(:,:,1)=S1;S(:,:,2)=S1;S(:,:,3)=S1;
P=[1/3 1/3 1/3]'; N=1000;
randn('seed',0)
[X,y]=generate_gauss_classes(m,S,P,N);
```

where

X is the $3 \times N$ matrix that contains the data vectors in its columns,

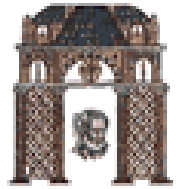
y is an N-dimensional vector that contains the class labels of the respective data vectors,

P is the vector of the respective class a priori probabilities.

The data set X_1 is generated similarly:

```
randn('seed',100);
[X1,y1]=generate_gauss_classes(m,S,P,N);
```

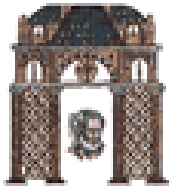
where *randn* is initialized using $seed = 100$.



Perform the following:

Step 1. To compute the ML estimates of the mean values and covariance matrix (common to all three classes), use *Gaussian_ML_estimate* by typing

```
class1_data=X(:,find(y==1));
[m1_hat, S1_hat]=Gaussian_ML_estimate(class1_data);
class2_data=X(:,find(y==2));
[m2_hat, S2_hat]=Gaussian_ML_estimate(class2_data);
class3_data=X(:,find(y==3));
[m3_hat, S3_hat]=Gaussian_ML_estimate(class3_data);
S_hat=(1/3)*(S1_hat+S2_hat+S3_hat);
m_hat=[m1_hat m2_hat m3_hat];
```



Step 2. For the Euclidean distance classifier, use the ML estimates of the means to classify the data vectors of X_1, typing

```
z_euclidean=euclidean_classifier(m_hat,X1);
```

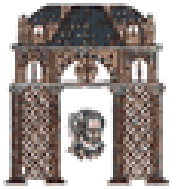
where *z_euclidean* is an N-dimensional vector containing the labels of the classes where the respective data vectors are assigned by the Euclidean classifier.

Step 3. Similarly for the Mahalanobis distance classifier, type

```
z_mahalanobis=mahalanobis_classifier(m_hat,S_hat,X1);
```

Step 4. For the Bayesian classifier, use function *bayes_classifier* and provide as input the matrices m, S, P, which were used for the data set generation. In other words, use the true values of m, S, and P and not their estimated values. Type

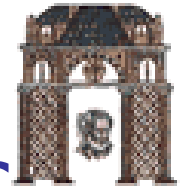
```
z_bayesian=bayes_classifier(m,S,P,X1);
```



Step 5. To compute the error probability for each classifier, compare the vector y_1 of the true class labels of the vectors of X_1 with vectors *z_euclidean*, *z_mahalanobis*, and *z_bayesian*, respectively. For each comparison, examine the vector elements in pairs and count the number of matches (i.e., correct classifications); divide by the length of y_1. Type

```
err_euclidean = (1-length(find(y1==z_euclidean))/length(y1));
err_mahalanobis = (1-length(find(y1==z_mahalanobis))/length(y1));
err_bayesian = (1-length(find(y1==z_bayesian))/length(y1));
```

The error probabilities for the Euclidean, Mahalanobis, and Bayesian classifiers are 7.61%, 7.71%, and 7.61%, respectively. The results are almost equal since all of the four assumptions in Subsection 1.4.1 are valid, which implies that in the present case the three classifiers are equivalent. ■



Παράδειγμα: Απλοποιημένος Εκτιμητής

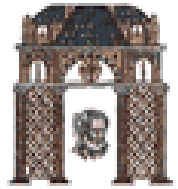
Έστω δεδομένα που δημιουργούνται από την παρακάτω pdf

$$p(\mathbf{x} \mid \omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}} \left|\Sigma_i\right|^{\frac{d}{2}}} \exp\left(-(\mathbf{x}-\mu_\mathbf{i})^T \Sigma_i^{-1} (\mathbf{x}-\mu_\mathbf{i})\right)$$

Με i=2, d=5 και $\mu_1=[0\ 0\ 0\ 0\ 0]^T$, $\mu_2=[1\ 1\ 1\ 1\ 1]T$

$$\Sigma_1 = \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}1 & 0{,}05 & 0{,}01 \\ 0{,}2 & 0{,}7 & 0{,}1 & 0{,}03 & 0{,}02 \\ 0{,}1 & 0{,}1 & 0{,}8 & 0{,}02 & 0{,}01 \\ 0{,}05 & 0{,}03 & 0{,}02 & 0{,}9 & 0{,}01 \\ 0{,}01 & 0{,}02 & 0{,}01 & 0{,}01 & 0{,}8 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 0{,}9 & 0{,}1 & 0{,}05 & 0{,}02 & 0{,}01 \\ 0{,}1 & 0{,}8 & 0{,}1 & 0{,}02 & 0{,}02 \\ 0{,}05 & 0{,}1 & 0{,}7 & 0{,}02 & 0{,}01 \\ 0{,}02 & 0{,}02 & 0{,}02 & 0{,}6 & 0{,}02 \\ 0{,}01 & 0{,}02 & 0{,}01 & 0{,}02 & 0{,}7 \end{bmatrix}$$

Το δείγμα εκπαίδευσης X_1 έχει N=50 δείγματα, ενώ το δείγμα ελέγχου X_2 έχει N=10.000 δείγματα



Bayesian εκτιμητής

Πετάμε ένα νόμισμα N=10 φορές και φέρνουμε {κ,κ,γ,κ,γ,κ,κ,κ,γ,κ} (κ=κεφάλι,γ=γράμματα). Ποια είναι η πιθανότητα **θ** να φέρουμε κεφάλι; Η αρχική κατανομή του θ είναι ομοιόμορφη στο (0, 1)

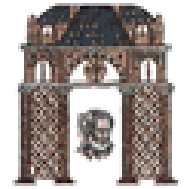
Να βρεθεί (αριθμητικά) το $p(x = k \mid D^{10})$

Λύση: Γνωρίζουμε

$$p(\boldsymbol{\theta} \mid D^n) = \frac{p(x_n \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D^{n-1})}{\int p(x_n \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D^{n-1}) d\boldsymbol{\theta}}, \quad p(\boldsymbol{\theta} \mid D^0) \text{ γνωστό (apriori)}$$

$$\text{ομως} \quad p(x_n \mid \boldsymbol{\theta}) = \begin{cases} \boldsymbol{\theta} & \text{εάν κεφάλι} \\ (1-\boldsymbol{\theta}) & \text{εάν γράμματα} \end{cases}, \text{ και συνεπώς έχουμε}$$

$$p(\boldsymbol{\theta} \mid D^N) = \frac{\boldsymbol{\theta}^k (1-\boldsymbol{\theta})^{N-k}\, p(\boldsymbol{\theta} \mid D^0)}{\int \boldsymbol{\theta}^k (1-\boldsymbol{\theta})^{N-k}\, p(\boldsymbol{\theta} \mid D^0) d\boldsymbol{\theta}}$$



```
clear all; close all; format long

syms r
u=0:0.01:1;
y1= u / double(int(r,r,0,1));
y2= (u.^3).*((1-u).^2) / double(int(r^3*(1-r)^2,r,0,1));
y3= (u.^7).*((1-u).^3) / double(int(r^7*(1-r)^3,r,0,1));
hold on; axis([0 1 0 3]);
plot(u,y1,'r'),xlabel('\Theta'), ylabel('p(\Theta|D^n)');
plot(u,y2,'b'); plot(u,y3,'g'); grid on
h = legend('p(Theta|D^1)','p(Theta|D^5)','p(Theta|D^10)','Location','NorthWest');
set(h,'Interpreter','none')
hold off

[q,w]=max(double(y3));
disp(['The max value for y3 is=' num2str(q)])
disp(['for x=' num2str((w-1)/100)])
```

The max value for y3 is=2.9351
for x=0.70