



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Mixturas de Gaussianas¹

Alfons Juan
Jorge Civera
Javier Iranzo

DSIC

Departament de Sistemes
Informàtics i Computació

¹Para una correcta visualización, se requiere Acrobat Reader v. 7.0 o superior

Índice

1. El corpus MNIST	1
2. El clasificador Gaussiano	2
2.1. La distribución Gaussiana multivariada	2
2.2. El clasificador Gaussiano	3
2.3. Estimación máximo-verosímil	7
3. Mixturas finitas	14
3.1. Modelo de mixtura finita	14
3.2. Estimación máximo-verosímil	15
4. Clasificador con mixturas de Gaussianas	17
4.1. Clasificador con mixturas de Gaussianas	17
4.2. Estimación máximo-verosímil	18
5. Ejercicios	30
5.1. Ejercicio 1 (0.5 puntos)	30
5.2. Ejercicio 2 (0.2 puntos)	30
5.3. Ejercicio 3 (0.3 puntos)	31

1. El corpus MNIST

- MNIST: 60K imágenes de entrenamiento y 10K de test.

mnist_sizes.sh

```
#!/usr/bin/octave
load("train-images-idx3-ubyte.mat.gz"); size(X)
load("train-labels-idx1-ubyte.mat.gz"); size(xl)
load("t10k-images-idx3-ubyte.mat.gz"); size(Y)
load("t10k-labels-idx1-ubyte.mat.gz"); size(yl)
```

- Visualización:

mnist_show.sh

```
#!/usr/bin/octave
load("train-images-idx3-ubyte.mat.gz");
for n=1:50
    x=reshape(X(n,:),28,28); imshow((255-x)',[]); pause(.5);
end
```

- **trains:** primeras N imágenes de entrenamiento.

trains.sh

```
#!/usr/bin/octave
load("train-images-idx3-ubyte.mat.gz"); T=X;
load("train-labels-idx1-ubyte.mat.gz"); Tl=xl;
for N=[2000 20000]
    X=T(1:N,:); save("-z",sprintf("train%d-images.gz",N),"X");
    xl=Tl(1:N); save("-z",sprintf("train%d-labels.gz",N),"xl");
end
```

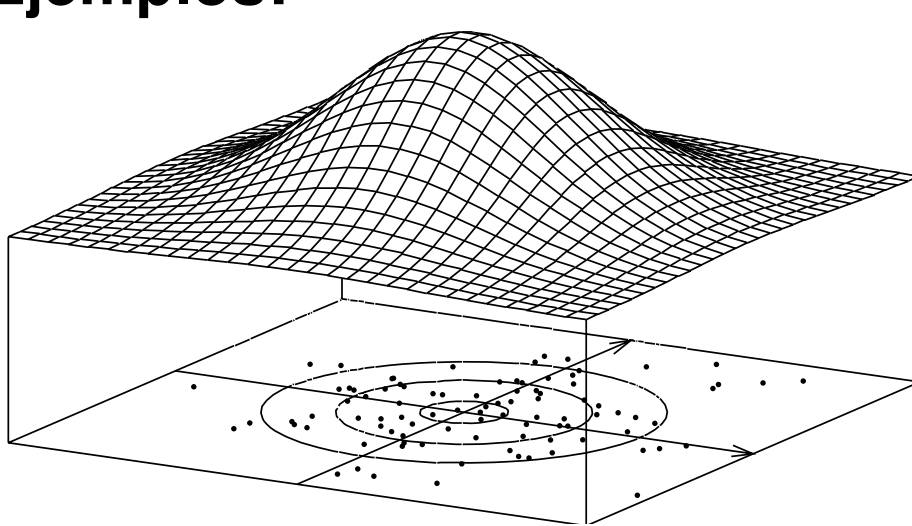
2. El clasificador Gaussiano

2.1. La distribución Gaussiana multivariada

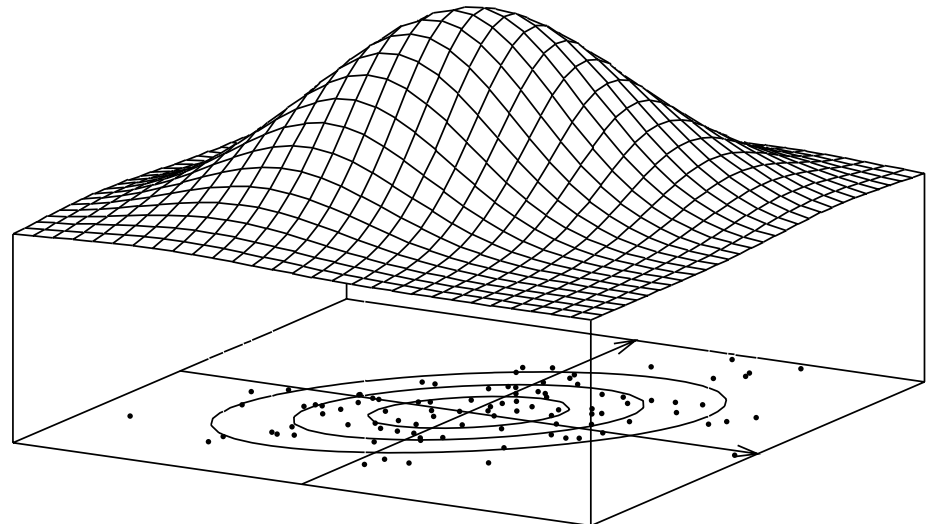
- Sea $\mu \in \mathbb{R}^D$ y sea $\Sigma \in \mathbb{R}^{D \times D}$ simétrica y definida positiva.
- Un vector de características $x \in \mathbb{R}^D$ es $N_D(\mu, \Sigma)$ si su f.d.p. es:

$$p(x) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (x - \mu)^t \Sigma^{-1} (x - \mu) \right)$$

Ejemplos:



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

2.2. El clasificador Gaussiano

- El clasificador de Bayes para un vector D -dimensional \mathbf{x} es:

$$c^*(\mathbf{x}) = \arg \max_c p(c \mid \mathbf{x}) = \arg \max_c p(c) p(\mathbf{x} \mid c)$$

- Suponemos que las densidades condicionales son Gaussianas:

$$p(\mathbf{x} \mid c) \sim N_D(\boldsymbol{\mu}_c, \Sigma_c) \quad (\text{para todo } c)$$

- El clasificador de Bayes se reduce al **clasificador Gaussiano**:

$$c^*(\mathbf{x}) = \arg \max_c \ln p(c) + \ln p(\mathbf{x} \mid c)$$

donde, omitiendo la constante aditiva $-\frac{D}{2} \ln(2\pi)$:

$$\ln p(\mathbf{x} \mid c) = -\frac{1}{2} \ln |\Sigma_c| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_c)^t \Sigma_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)$$

- El clasificador Gaussiano es **cuadrático** con x :

$$c^*(x) = \arg \max_c g_c(x) \quad \text{con} \quad g_c(x) = x^t W_c x + w_c^t x + w_{c0}$$

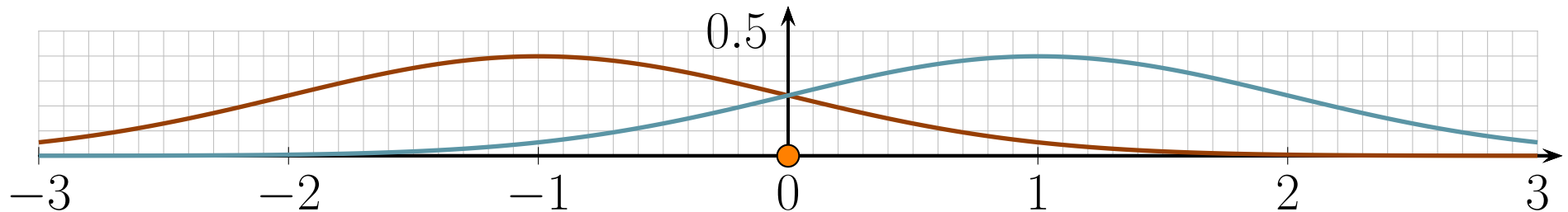
donde

$$W_c = -\frac{1}{2} \Sigma_c^{-1}$$

$$w_c = \Sigma_c^{-1} \mu_c$$

$$w_{c0} = \ln p(c) - \frac{1}{2} \ln |\Sigma_c| - \frac{1}{2} \mu_c^t \Sigma_c^{-1} \mu_c$$

- **Ejemplo:** $C=2$ $D=1$ $p(1)=p(2)=\frac{1}{2}$ $\mu_1=-1$ $\mu_2=1$ $\sigma_1^2=\sigma_2^2=1$



$$g_c(x) = -\frac{1}{2\sigma_c^2} x^2 + \frac{\mu_c}{\sigma_c^2} x + \ln p(c) - \frac{1}{2} \ln \sigma_c^2 - \frac{\mu_c^2}{2\sigma_c^2}$$

$$\left. \begin{array}{l} g_1(x) = -x \\ g_2(x) = x \end{array} \right\} \rightarrow c^*(x) = \begin{cases} \mathbf{1} & \text{si } x < 0 \\ \mathbf{2} & \text{si } x \geq 0 \end{cases}$$

► $\ln |\Sigma|$: $|\Sigma| = \prod_d \lambda_d \Rightarrow \ln |\Sigma| = \sum_d \ln \lambda_d$

logdet.m

```
function v = logdet(X)
    lambda = eig(X);
    if any(lambda <= 0)
        v = log(realmin);
    else
        v = sum(log(lambda));
    end
end
```

$$\blacktriangleright \ln p(\mathbf{x} \mid c) = -\frac{1}{2}(\mathbf{x}^t \Sigma_c^{-1} \mathbf{x} + \ln |\Sigma_c| + \boldsymbol{\mu}_c^t \Sigma_c^{-1} \boldsymbol{\mu}_c) + \mathbf{x}^t \Sigma_c^{-1} \boldsymbol{\mu}_c$$

$$\ln \mathbf{p}(\mathbf{X} \mid c) = -\frac{1}{2}(\mathbf{X} \Sigma_c^{-1} \odot \mathbf{X} \mathbf{1}_D + \ln |\Sigma_c| + \boldsymbol{\mu}_c^t \Sigma_c^{-1} \boldsymbol{\mu}_c) + \mathbf{X} \Sigma_c^{-1} \boldsymbol{\mu}_c$$

```

_____ compute_pxGc.m _____
function [pxGc] = compute_pxGc(mu, sigma, X)
    I=pinv(sigma);
    qua=-0.5*sum((X*I).*X,2);
    lin=X*I*mu;
    cons=-0.5*logdet(sigma);
    cons=cons-0.5*mu'*I*mu;
    pxGc=qua+lin+cons;
end

```

```

_____ Cálculo de funciones discriminantes _____
X=[-3:3]'; [X compute_pxGc(-1,1,X) compute_pxGc(1,1,X)]
ans = -3.00000    -2.00000    -8.00000
       -2.00000    -0.50000    -4.50000
       -1.00000     0.00000    -2.00000
        0.00000    -0.50000    -0.50000
        1.00000    -2.00000     0.00000
        2.00000    -4.50000    -0.50000
        3.00000    -8.00000    -2.00000

```


2.3. Estimación máximo-verosímil

► **Log-verosimilitud** de $\Theta = \{(p(c), \boldsymbol{\mu}_c, \Sigma_c)\}$ respecto a $\{(\mathbf{x}_n, c_n)\}$:

$$L(\Theta; \mathbf{X}) = \sum_c \sum_{n:c_n=c} \ln p(c) - \frac{1}{2} \ln |\Sigma_c| - \frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu}_c)^t \Sigma_c^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_c)$$

► **Estimador máximo-verosímil** de $\Theta, \hat{\Theta}$: para todo c :

$$\hat{p}(c) = \frac{N_c}{N}$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{n:c_n=c} \mathbf{x}_n$$

$$\hat{\Sigma}_c = \frac{1}{N_c} \sum_{n:c_n=c} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_c)^t$$

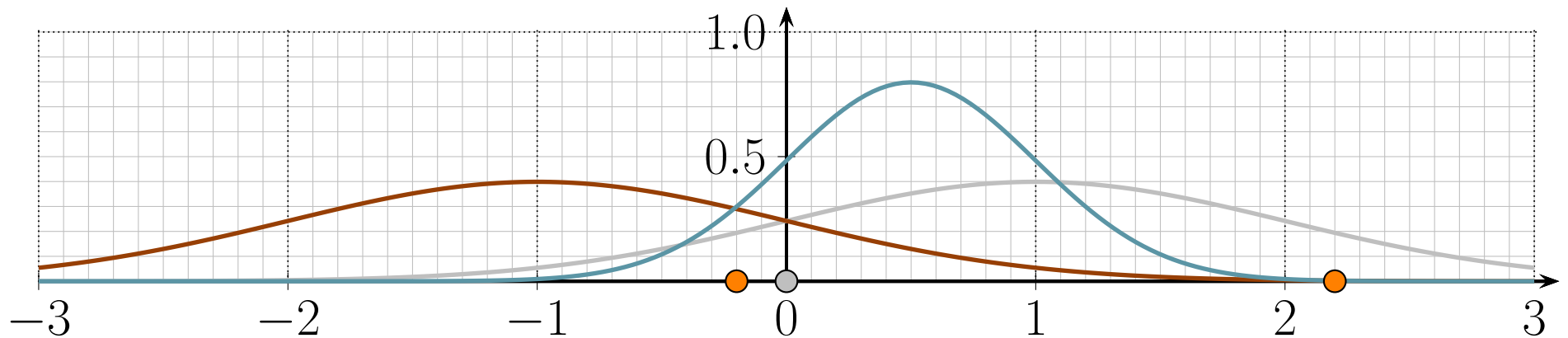
► **Suavizado** de matrices de covarianzas: $0 \leq \alpha \leq 1$

$$\tilde{\Sigma}_c = \alpha \hat{\Sigma}_c + (1 - \alpha) I_D$$

Ej. (cont.): $C=2$ $D=1$ $p(1)=p(2)=\frac{1}{2}$ $\mu_1=-1$ $\mu_2=1$ $\sigma_1^2=\sigma_2^2=1$

$$\{(-2, 1), (0, 2), (0, 1), (1, 2)\} \rightarrow \begin{cases} \hat{p}(1) = \hat{p}(2) = \frac{2}{4} \\ \hat{\mu}_1 = -1 & \hat{\mu}_2 = 0.5 \\ \hat{\sigma}_1^2 = 1 & \hat{\sigma}_2^2 = 0.25 \end{cases}$$

$$\rightarrow \begin{cases} \hat{g}_1(x) = -\frac{1}{2}x^2 - x \\ \hat{g}_2(x) = -2x^2 + 2x + \ln 2 \end{cases} \xrightarrow{\hat{g}_1(x)=\hat{g}_2(x)} x = \begin{cases} -0.2 \\ 2.2 \end{cases}$$



$$\hat{c}(x) = \begin{cases} \mathbf{1} & x \notin [-0.2, 2.2] \\ \mathbf{2} & x \in [-0.2, 2.2] \end{cases} \approx c^*(x)$$

```

function [errY] = gaussian.m_gaussian(X,xl,Y,yl,alphas)

classes=unique(xl);
N=rows(X);
M=rows(Y);
D=columns(X);

for c=classes'
    ic=find(c==classes);
    idx=find(xl==c);
    Xc=X(idx,:);
    Nc=rows(Xc);
    pc(ic)=Nc/N;
    muc=sum(Xc)/Nc;
    mu(:,ic)=muc';
    sigma{ic}=(Xc-muc)'*(Xc-muc)/Nc;
end

for i=1:length(alphas)
    for c=classes'
        ic=find(c==classes);
        ssigma{ic}=alphas(i)*sigma{ic}+(1-alphas(i))*eye(D);
    end

    for c=classes'
        ic=find(c==classes);
        gY(:,ic)=log(pc(ic))+compute_pxGc(mu(:,ic),ssigma{ic},Y);
    end

    [~,idY]=max(gY');
    errY(i)=mean(classes(idY)~=yl)*100;
end

```

```

#!/usr/bin/octave -qf
if (nargin!=5)
printf("Usage: gaussian-exp.m <trdata> <trlabs> <alphas> \
<%trper> <%dvper>\n")
exit(1);
end;

arg_list=argv();
trdata=arg_list{1};
trlabs=arg_list{2};
alphas=str2num(arg_list{3});
trper=str2num(arg_list{4});
dvper=str2num(arg_list{5});

load(trdata);
load(trlabs);

N=rows(X);
seed=23; rand("seed",seed); permutation=randperm(N);
X=X(permutation,:); xl=xl(permutation,:);

Ntr=round(trper/100*N);
Ndv=round(dvper/100*N);
Xtr=X(1:Ntr,:); xltr=xl(1:Ntr);
Xdv=X(N-Ndv+1:N,:); xldv=xl(N-Ndv+1:N);

[edv] = gaussian(Xtr,xltr,Xdv,xldv,alphas);

printf("\n  alpha dv-err");
printf("\n----- \n");
for i=1:length(alphas)
    printf("%.1e %6.3f\n",alphas(i),edv(i));
end

```

```
time ./gaussian-exp.m train-images-idx3-ubyte.mat.gz  
  ↪ train-labels-idx1-ubyte.mat.gz "[1e-5 1e-4  
  ↪ 1e-3]" 90 10
```

alpha	dv-err
-----	-----
1.0e-05	6.317
1.0e-04	4.267
1.0e-03	6.383

real	1m15,827s
user	2m14,806s
sys	0m17,510s

- **Experimento final:** fijamos el hiperparámetro $\alpha \triangleq 10^{-4}$ y usamos t10k por primera y única vez para estimar el error del Gaussiano

gaussian-eva.m

```
#!/usr/bin/octave -qf
if (nargin!=5)
printf("Usage: gaussian-eva.m <trdata> <trlabs> <tedata> \
<telabs> <alpha>\n")
exit(1);
end;

arg_list=argv();
trdata=arg_list{1};
trlabs=arg_list{2};
tedata=arg_list{3};
telabs=arg_list{4};
alpha=str2num(arg_list{5});

load(trdata); load(trlabs);
load(tedata); load(telabs);

[ete] = gaussian(X,xl,Y,yl,alpha);

printf("\n  alpha te-err");
printf("\n----- \n");
printf("%.1e %6.3f\n",alpha,ete);
```

```
time ./gaussian-eva.m train-images-idx3-ubyte.mat.gz
↪ train-labels-idx1-ubyte.mat.gz
↪ t10k-images-idx3-ubyte.mat.gz
↪ t10k-labels-idx1-ubyte.mat.gz 1e-4

    alpha te-err
-----
1.0e-04  4.180

real    0m38,028s
user    1m8,171s
sys     0m8,949s
```

3. Mixturas finitas

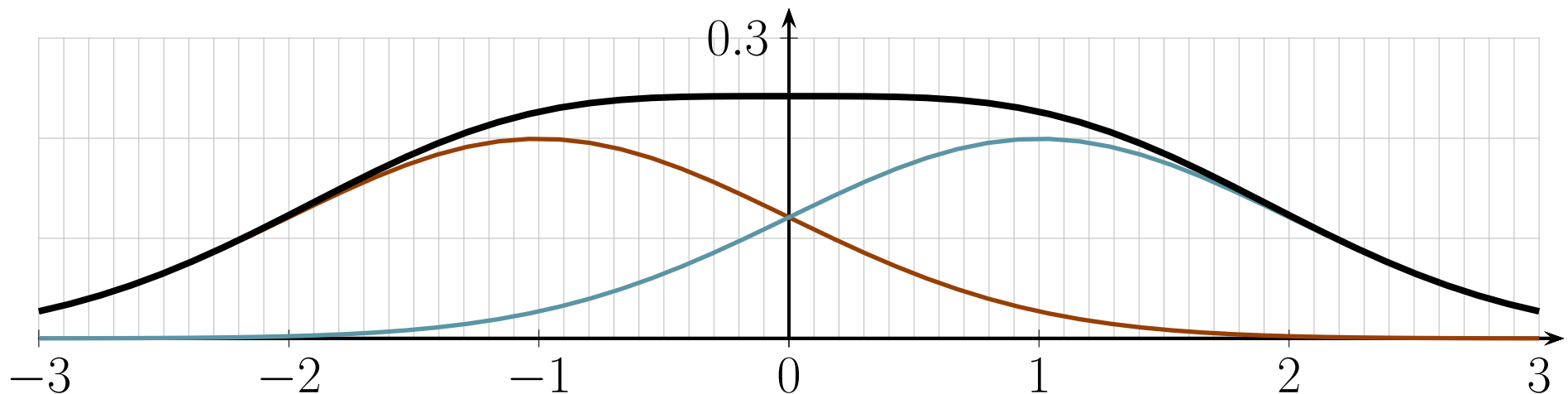
3.1. Modelo de mixtura finita

► Un modelo de *mixtura finita* de K componentes es:

$$p_{\Theta}(\mathbf{x}) = \sum_{k=1}^K p_k p_{\Theta'}(\mathbf{x} \mid k) \quad (p_k > 0, p_1 + \dots + p_K = 1)$$

siendo p_k y $p_{\Theta'}(\mathbf{x} \mid k)$ los k -ésimos *coeficiente* y *componente*.

Ejemplo: $p(x) = \frac{1}{2} N(\mu_2 = -1, \sigma_2^2 = 1) + \frac{1}{2} N(\mu_1 = 1, \sigma_1^2 = 1)$



3.2. Estimación máximo-verosímil

► *Log-verosimilitud* de $\Theta = (\{p_k\}, \Theta')$ respecto a un conjunto $\{\mathbf{x}_n\}$:

$$L(\Theta; \mathbf{X}) = \sum_n \ln \sum_{k=1}^K p_k p_{\Theta'}(\mathbf{x}_n \mid k)$$

► *Estimador máximo-verosímil* de Θ : **EM**: $\Theta^{(0)} \rightarrow \Theta^{(1)} \rightarrow \dots \rightarrow \hat{\Theta}$

$$\Theta^{(t+1)} = \arg \max_{\Theta} Q(\Theta, \Theta^{(t)}) \quad \text{sujeto a } \sum_k p_k = 1$$

donde

$$Q(\Theta, \Theta^{(t)}) = \sum_n \sum_k z_{nk}^{(t)} (\ln p_k + \ln p_{\Theta'}(\mathbf{x}_n \mid k))$$

con

$$z_{nk}^{(t)} = \frac{p_k^{(t)} p_{\Theta'^{(t)}}(\mathbf{x}_n \mid k)}{\sum_{k'} p_{k'}^{(t)} p_{\Theta'^{(t)}}(\mathbf{x}_n \mid k')}$$

Ejemplo (cont.): $p(x) = \frac{1}{2} N(\mu_1 = -1, \sigma_1^2 = 1) + \frac{1}{2} N(\mu_2 = 1, \sigma_2^2 = 1)$

$$Q(\Theta, \Theta^{(t)}) = \sum_n \sum_k z_{nk}^{(t)} (\ln p_k + \ln \mathcal{N}(\mu_k, \sigma_k^2; x_n))$$

$$\Theta^{(t+1)} = \left\{ \begin{array}{l} p_k^{(t+1)} = \frac{N_k}{N} \quad \text{con} \quad N_k = \sum_n z_{nk}^{(t)} \\ \mu_k^{(t+1)} = \frac{1}{N_k} \sum_n z_{nk}^{(t)} x_n \\ \sigma_k^{2(t+1)} = \frac{1}{N_k} \sum_n z_{nk}^{(t)} \left(x_n - \mu_k^{(t+1)} \right)^2 \end{array} \right\}$$

4. Clasificador con mixturas de Gaussianas

4.1. Clasificador con mixturas de Gaussianas

- Suponemos que las condicionales son mixturas de K Gaussianas:

$$p(\mathbf{x} \mid c) = \sum_{k=1}^K p(\mathbf{x}, k \mid c) = \sum_{k=1}^K p(k \mid c) p(\mathbf{x} \mid c, k)$$

con

$$p(\mathbf{x} \mid c, k) \sim N_D(\boldsymbol{\mu}_{ck}, \Sigma_{ck}) \quad (\text{para todo } c \text{ y } k)$$

- Bayes se reduce al *clasificador con mixturas de Gaussianas*:

$$c^*(\mathbf{x}) = \arg \max_c \ln p(c) + \ln p(\mathbf{x} \mid c)$$

donde, omitiendo la constante aditiva $-\frac{D}{2} \ln(2\pi)$:

$$\ln p(\mathbf{x} \mid c) = \ln \sum_{k=1}^K p(k \mid c) |\Sigma_{ck}|^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{ck})^t \Sigma_{ck}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{ck}) \right)$$

4.2. Estimación máximo-verosímil

► *Log-verosimilitud* de Θ respecto a $\{(\mathbf{x}_n, c_n)\}$:

$$\begin{aligned} L(\Theta; \mathbf{X}) &= \sum_n \ln p(\mathbf{x}_n, c_n) \\ &= \sum_n \ln p(c_n) + \ln p(\mathbf{x}_n \mid c_n) \\ &= \sum_c \sum_{n:c_n=c} \ln p(c_n) + \ln \sum_{k=1}^K p_{ck} \mathcal{N}(\boldsymbol{\mu}_{ck}, \Sigma_{ck}; \mathbf{x}_n) \end{aligned}$$

► **Algoritmo EM:** $\Theta^{(0)} \rightarrow \Theta^{(1)} \rightarrow \dots \rightarrow \hat{\Theta}$

$$\Theta^{(t+1)} = \arg \max_{\Theta} Q(\Theta, \Theta^{(t)}) \quad \text{sujeto a } \sum_k p_{ck} = 1 \quad \forall c$$

con

$$Q(\Theta, \Theta^{(t)}) = \sum_c \sum_{n:c_n=c} \ln p(c_n) + \sum_k z_{nk}^{(t)} (\ln p_{ck} + \ln \mathcal{N}(\boldsymbol{\mu}_{ck}, \Sigma_{ck}; \mathbf{x}_n))$$

y

$$z_{nk}^{(t)} = \frac{p_{ck}^{(t)} \mathcal{N}(\boldsymbol{\mu}_{ck}^{(t)}, \Sigma_{ck}^{(t)}; \mathbf{x}_n)}{\sum_{k'} p_{ck'}^{(t)} \mathcal{N}(\boldsymbol{\mu}_{ck'}^{(t)}, \Sigma_{ck'}^{(t)}; \mathbf{x}_n)}$$

El paso M maximiza Q como sigue:

$$p_{ck}^{(t+1)} = \frac{1}{N_c} \sum_{n:c_n=c} z_{nk}^{(t)}$$

$$\boldsymbol{\mu}_{ck}^{(t+1)} = \frac{1}{\sum_{n:c_n=c} z_{nk}^{(t)}} \sum_{n:c_n=c} z_{nk}^{(t)} \mathbf{x}_n$$

$$\Sigma_{ck}^{(t+1)} = \frac{1}{\sum_{n:c_n=c} z_{nk}^{(t)}} \sum_{n:c_n=c} z_{nk}^{(t)} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_{ck}^{(t+1)}) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_{ck}^{(t+1)})^t$$

► **Suavizado** de matrices de covarianzas: $0 \leq \alpha \leq 1$

$$p(\mathbf{x} \mid c, k) \sim N_D(\boldsymbol{\mu}_{ck}, \alpha \Sigma_{ck} + (1 - \alpha) I_D) \quad (\text{para todo } c \text{ y } k)$$

Sustituimos Σ_{ck} por $\alpha \Sigma_{ck} + (1 - \alpha) I_D$ en $c^*(\mathbf{x})$, $L(\boldsymbol{\Theta}; \mathbf{X})$ y EM.

► **Cálculo robusto:** sea $a_k = p_k^{(t)} \mathcal{N}(\boldsymbol{\mu}_k^{(t)}, \Sigma_k^{(t)}; \mathbf{x}_n)$

$$\begin{aligned} z_{nk}^{(t)} &= \frac{a_k}{\sum_{k'} a_{k'}} = \frac{\frac{a_k}{\max_{k''} a_{k''}}}{\sum_{k'} \frac{a_{k'}}{\max_{k''} a_{k''}}} = \frac{\exp\left(\ln\left(\frac{a_k}{\max_{k''} a_{k''}}\right)\right)}{\sum_{k'} \exp\left(\ln\left(\frac{a_{k'}}{\max_{k''} a_{k''}}\right)\right)} \\ &= \frac{\exp(\ln a_k - \max_{k''} \ln a_{k''})}{\sum_{k'} \exp(\ln a_{k'} - \max_{k''} \ln a_{k''})} \end{aligned}$$

La verosimilitud de una muestra \mathbf{x}_n aprovecha el denominador de $z_{nk}^{(t)}$, pero es necesario cancelar el factor $\max_{k''} \ln a_{k''}$:

$$\begin{aligned} L(\boldsymbol{\Theta}; \mathbf{x}_n) &= \ln p(c) + \max_{k''} \ln a_{k''} + \ln \sum_{k'} a_{k'} \\ &= \ln p(c) + \max_{k''} \ln a_{k''} + \ln \sum_{k'} \exp\left(\ln a_{k'} - \max_{k''} \ln a_{k''}\right) \end{aligned}$$

► Log del numerador de z_{nk} para todo n : $\ln p_{ck} + \ln \mathcal{N}(\boldsymbol{\mu}_{ck}, \boldsymbol{\Sigma}_{ck}; \mathbf{x}_n)$

```
compute_zk.m  
function [zk] = compute_zk(ic,k,pkGc,mu,sigma,X)  
    D=columns(X);  
    I=pinv(sigma{ic,k});  
    cons=log(pkGc{ic}(k));  
    cons=cons-0.5*D*log(2*pi);  
    cons=cons-0.5*logdet(sigma{ic,k});  
    cons=cons-0.5*mu{ic}(:,k)'*I*mu{ic}(:,k);  
    lin=X*I*mu{ic}(:,k);  
    qua=-0.5*sum((X*I).*X,2);  
    zk=qua+lin+cons;  
end
```

► Inicialización del clasificador con mixturas de Gaussianas:

```
Inicialización
pc=histc(xl,classes)/N;

sigma=cell(C,K);
for c=classes'
    ic=find(c==classes);
    pkGc{ic}(1:K)=1/K;
    idc=find(xl==c);
    Nc=rows(idc);
    mu{ic}=X(idc(randperm(Nc,K)),:)' ;
    sigma(ic,1:K)=alpha*cov(X(idc,:),1)/K+(1-alpha)*eye(D);
end
```


► Paso E del clasificador con mixturas de Gaussianas:

Paso E

```
% For each class
for c=classes'
    % E step: Estimate znk
    ic=find(c==classes);
    idc=find(xl==c);
    Nc=rows(idc);
    Xc=X(idc,:);
    z=[];
    for k=1:K
        z(:,k)=compute_zk(ic,k,pkGc,mu,sigma,Xc);
    end
    % Robust computation of znk and log-likelihood
    maxz=max(z,[],2);
    z=exp(z-maxz);
    sumz=sum(z,2);
    z=z./sumz;
    L=L+Nc*log(pc(ic))+sum(maxz+log(sumz));
```

► Paso M del clasificador con mixturas de Gaussianas:

Paso M

```
% M step: parameter update
% Weight of each component
sumz=sum(z);
pkGc{ic}=sumz/Nc;
mu{ic}=(Xc'*z)./sumz;
for k=1:K
    covar=((Xc-mu{ic}(:,k))'*(Xc-mu{ic}(:,k)).*z(:,k)));
    covar=covar/sumz(k);
    % Smoothing covariance matrix with identity matrix
    sigma(ic,k)=alpha*covar+(1-alpha)*eye(D);
end
```

$$p_{ck}^{(t+1)} = \frac{1}{N_c} \sum_{n:c_n=c} z_{nk}^{(t)}$$

$$\boldsymbol{\mu}_{ck}^{(t+1)} = \frac{1}{\sum_{n:c_n=c} z_{nk}^{(t)}} \sum_{n:c_n=c} z_{nk}^{(t)} \mathbf{x}_n$$

$$\Sigma_{ck}^{(t+1)} = \frac{1}{\sum_{n:c_n=c} z_{nk}^{(t)}} \sum_{n:c_n=c} z_{nk}^{(t)} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_{ck}^{(t+1)}) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_{ck}^{(t+1)})^t$$

► Clasificación del cjto de evaluación con mixturas de Gaussianas:

Clasificación

```
% Compute g for evaluation set
for c=classes'
    % Evaluation set
    z=[];
    for k=1:K
        z(:,k)=compute_zk(ic,k,pkGc,mu,sigma,Y);
    end
    % Robust computation of znk
    maxz=max(z,[],2);
    z=exp(z-maxz);
    sumz=sum(z,2);
    gY(:,ic)=log(pc(ic))+maxz+log(sumz);
end

[~,idY]=max(gY');
errY=mean(classes(idY)~=y1)*100;
```

```

mixgaussian-exp.m
#!/usr/bin/octave -qf
if (nargin!=6)
printf("Usage: mixgaussian-exp.m <trdata> <trlabs> <Ks> \
<alphas> <%trper> <%dvper>\n"); exit(1);
end;

arg_list=argv();
trdata=arg_list{1};
trlabs=arg_list{2};
Ks=str2num(arg_list{3});
alphas=str2num(arg_list{4});
trper=str2num(arg_list{5});
dvper=str2num(arg_list{6});

load(trdata); load(trlabs);
N=rows(X);
seed=23; rand("seed",seed); permutation=randperm(N);
X=X(permutation,:); xl=xl(permutation,:);

Ntr=round(trper/100*N); Ndv=round(dvper/100*N);
Xtr=X(1:Ntr,:); xltr=xl(1:Ntr);
Xdv=X(N-Ndv+1:N,:); xldv=xl(N-Ndv+1:N);

printf("\n  alpha    K  dv-err");
printf("\n-----  ---  -----");
for i=1:length(alphas)
    for k=1:length(Ks)
        edv = mixgaussian(Xtr,xltr,Xdv,xldv,Ks(k),alphas(i));
        printf("%.1e %3d %6.3f\n",alphas(i),Ks(k),edv);
    end
end
end

```

```

time ./mixgaussian-exp.m train-images-idx3-ubyte.mat.gz
↪ train-labels-idx1-ubyte.mat.gz 1 "[1e-5 1e-4 1e-3]" 90
↪ 10
  K    alpha dv-err
-----
It          oL          L    trerr    teerr
-----
  1          -Inf -1504209.90133    5.598    6.317
  2 -1504209.90133 -761413.95958    5.598    6.317
  3  -761413.95958 -761413.95958    5.598    6.317
  1 1.0e-05      6.32
It          oL          L    trerr    teerr
-----
  1          -Inf  -620842.17812    3.920    4.267
  2  -620842.17812 -313498.91244    3.920    4.267
  3  -313498.91244 -313498.91244    3.920    4.267
  1 1.0e-04      4.27
It          oL          L    trerr    teerr
-----
  1          -Inf  -179276.08786    5.115    6.383
  2  -179276.08786 -91071.29658    5.115    6.383
  3  -91071.29658 -91071.29658    5.115    6.383
  1 1.0e-03      6.38
real 15m13,494s
user 37m1,656s
sys  3m30,967s

```

```

time ./mixgaussian-exp.m train-images-idx3-ubyte.mat.gz
↪ train-labels-idx1-ubyte.mat.gz 2 "[1e-5 1e-4 1e-3]" 90 10
K alpha dv-err
---
It oL L trerr teerr
---
1 -Inf -1627893.22128 5.650 6.417
2 -1627893.22128 -729424.91406 5.467 5.967
3 -729424.91406 -717833.52284 5.237 5.700
4 -717833.52284 -709875.25125 5.050 5.433
5 -709875.25125 -705704.54934 4.811 5.233
6 -705704.54934 -703298.72854 4.681 5.350
. . .
24 -694987.70222 -694939.31105 4.648 5.350
2 1.0e-05 5.35
It oL L trerr teerr
---
1 -Inf -750399.88962 3.581 4.117
2 -750399.88962 -306572.28986 3.563 4.117
3 -306572.28986 -302978.08763 3.585 4.183
4 -302978.08763 -299361.95955 3.507 4.117
5 -299361.95955 -296546.65211 3.393 3.917
6 -296546.65211 -294420.18710 3.311 3.867
. . .
16 -290455.51334 -290428.93894 3.004 3.683
2 1.0e-04 3.68
It oL L trerr teerr
---
1 -Inf -236394.56423 4.794 6.333
2 -236394.56423 -89898.71269 4.809 6.317
3 -89898.71269 -89827.19591 4.806 6.317
4 -89827.19591 -89704.22590 4.806 6.317
5 -89704.22590 -89507.31242 4.759 6.350
6 -89507.31242 -89273.63102 4.709 6.250
. . .
24 -87991.54798 -87983.49608 4.533 5.967
2 1.0e-03 5.97
real 189m17,263s
user 491m51,727s
sys 52m26,086s

```

```

time ./mixgaussian-exp.m train-images-idx3-ubyte.mat.gz
↪ train-labels-idx1-ubyte.mat.gz 5 "[1e-5 1e-4 1e-3]" 90 10
K alpha dv-err
---
It oL L trerr teerr
---
1 -Inf -1600047.21121 4.474 5.183
2 -1600047.21121 -652960.19574 3.980 4.767
3 -652960.19574 -639984.48225 3.804 4.617
4 -639984.48225 -634795.54624 3.672 4.617
5 -634795.54624 -631617.59815 3.637 4.667
6 -631617.59815 -629532.79762 3.594 4.617
. . .
17 -625925.80963 -625874.52120 3.504 4.317
5 1.0e-05 4.32
It oL L trerr teerr
---
1 -Inf -952521.06166 2.815 3.817
2 -952521.06166 -280476.43414 2.376 3.100
3 -280476.43414 -271244.98216 2.028 2.900
4 -271244.98216 -266263.71107 1.883 2.817
5 -266263.71107 -263674.49345 1.796 2.800
6 -263674.49345 -262152.77957 1.780 2.700
. . .
17 -259744.70004 -259720.00407 1.787 2.933
5 1.0e-04 2.93
It oL L trerr teerr
---
1 -Inf -357943.53847 3.765 6.000
2 -357943.53847 -85598.45886 3.607 6.050
3 -85598.45886 -84909.98426 3.569 5.900
4 -84909.98426 -84182.36993 3.619 5.950
5 -84182.36993 -83274.55876 3.617 5.933
6 -83274.55876 -82383.34799 3.504 5.800
. . .
24 -80278.20523 -80275.20843 2.993 5.217
5 1.0e-03 5.22
real 393m1,899s
user 1111m30,183s
sys 111m33,235s

```

5. Ejercicios

5.1. Ejercicio 1 (0.5 puntos)

- ▶ Estima el error de clasificación en validación, en función de:

- ▷ El número de variables PCA: $D = 1, 2, 5, 10, 20, 50, 100$

- ▷ El número de componentes: $K = 1, 2, 5, 10, 20, 50, 100$

- ▷ El parámetro de suavizado: $\alpha = \dots$

Presenta los resultados con una gráfica por cada α probado: error en la vertical, K en la horizontal y una curva por cada D .

5.2. Ejercicio 2 (0.2 puntos)

- ▶ Estima el error de clasificación a partir de los conjuntos oficiales, con los mejores valores de D , K y α hallados en el ejercicio 1.

5.3. Ejercicio 3 (0.3 puntos)

- ▶ Modifica el código de aprendizaje del clasificador con mixturas de Gaussianas con el fin de mejorar el error en validación. Ideas:
 - ▷ Terminación temprana.
 - ▷ Número de componentes variable en cada clase.
 - ▷ Modificar inicialización.