



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Mixturas de Gaussianas¹

Alfons Juan

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	3
2.1. La distribución Gaussiana multivariada	3
2.2. El clasificador Gaussiano	4
2.3. Estimación máximo-verosímil	7
3. Mixturas finitas	11
3.1. Modelo de mixtura finita	11
3.2. Estimación máximo-verosímil	12
4. Clasificador con mixturas de Gaussianas	14
4.1. Clasificador con mixturas de Gaussianas	14
4.2. Estimación máximo-verosímil	16
5. Ejercicios	22
5.1. Ejercicio 1 (0.5 puntos)	22
5.2. Ejercicio 2 (0.2 puntos)	22
5.3. Ejercicio 3 (0.3 puntos)	23

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)
```

```
ans = 60000 784
ans = 60000 1
ans = 10000 784
ans = 10000 1
```

► 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=x1;
for N=[2000 20000]
    X=T(1:N,:); save("-z",sprintf("train%d-images.gz",N),"X");
    x1=Tl(1:N); save("-z",sprintf("train%d-labels.gz",N),"x1");
end
```

► Experimento sencillo con el vecino más próximo:

nn.sh

```
#!/usr/bin/octave
if (nargin!=2) printf("%s fX fxl\n",program_name()); exit; end
arg_list=argv(); fX=arg_list{1}; fxl=arg_list{2};
load(sprintf(fX)); load(sprintf(fxl));
N=rows(X); NTr=round(.7*N); NTe=N-NTr; rec=zeros(NTe,1);
for m=1:NTe
    x=X(NTr+m,:); nmin=1; min=inf;
    for n=1:NTr
        xn=X(n,:); a=x-xn; d=a'*a; if (d<min) min=d; nmin=n; endif
    end
    rec(m)=x1(nmin);
end
[Nerr m]=confus(x1(NTr+1:N),rec);
printf("%s %s %d %d %.1f\n",fX,fxl,Nerr,NTe,100.0*Nerr/NTe);
```

```
./nn.sh train2000-images.gz train2000-labels.gz
train2000-images.gz train2000-labels.gz 64 600 10.7
```

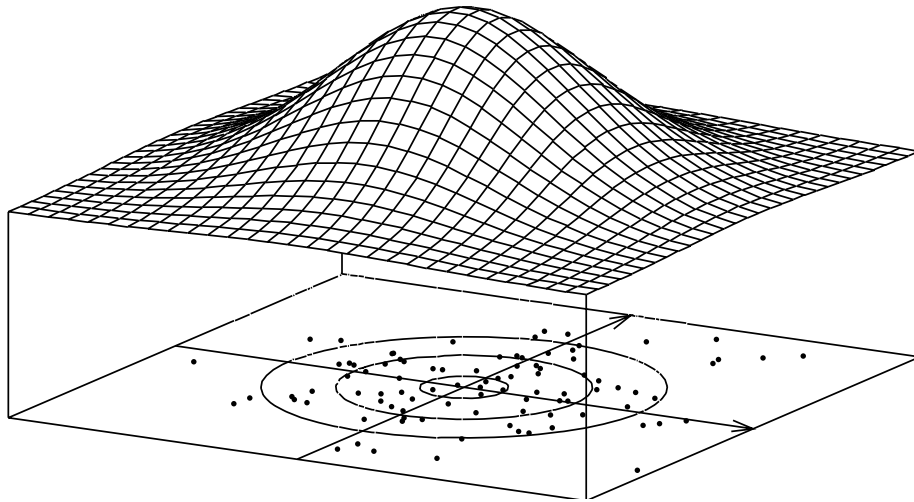
2. El clasificador Gaussiano

2.1. La distribución Gaussiana multivariada

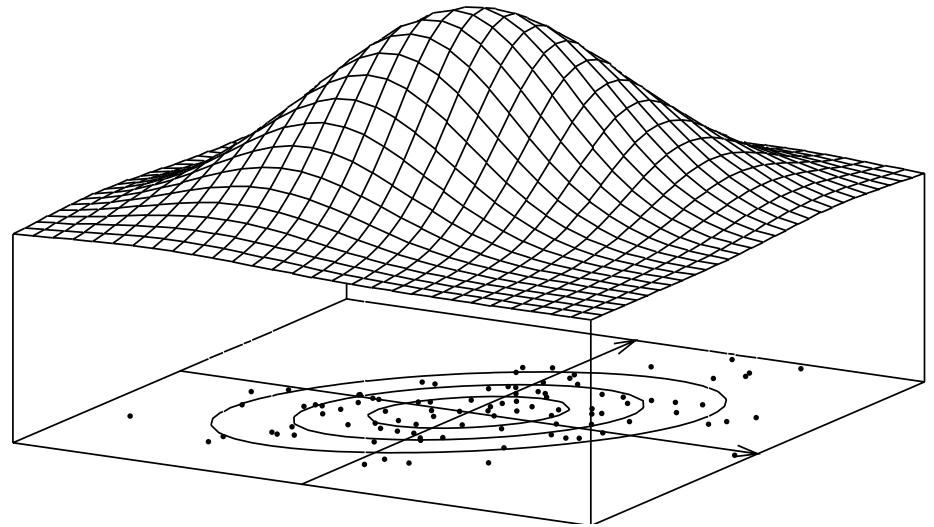
- Sea $\mu \in \mathbb{R}^D$ y sea $\Sigma \in \mathbb{R}^{D \times D}$ simétrica y definida positiva.
- Un vector de características $x \in \mathbb{R}^D$ es $N_D(\mu, \Sigma)$ si su f.d.p. es:

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

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 :

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

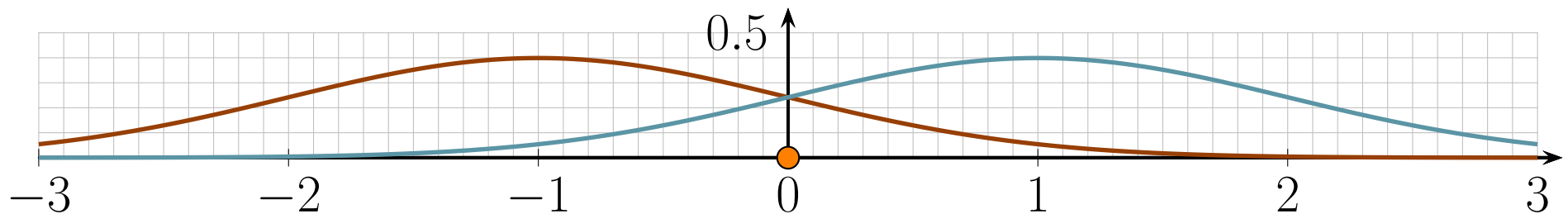
donde

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

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

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

- **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(S)
    L=eig(S); if any(L<=0) v=log(realmin); else v=sum(log(L)); end
end
```

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

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

_____ compute_pxGc.m _____

```
function [pxGc]=compute_pxGc(m, S, X)
    I=pinv(S); pxGc=-.5*(sum((X*I).*X,2)+logdet(S)+m'*I*m)+X*I*m;
end
```

_____ octave _____

```
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 $\{(\mathbf{x}_n, c_n)\}$:

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

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

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

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

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

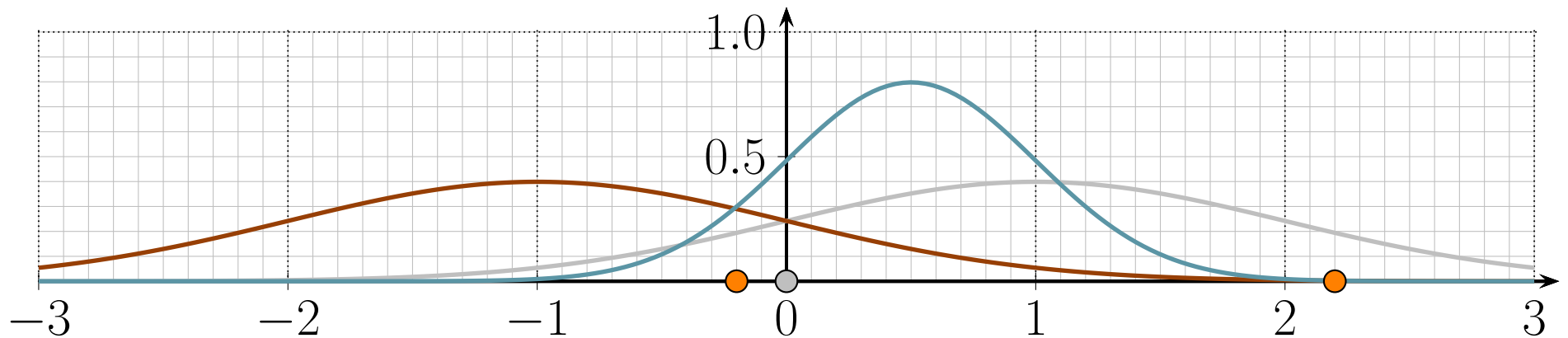
► **Suavizado** de matrices de covarianzas: $0 \leq \alpha \leq 1$

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

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

$$\{(-2, 1), (0, 2), (0, 1), (1, 2)\} \rightarrow \begin{cases} \hat{p}(1) = \hat{p}(2) = \frac{2}{4} \\ \hat{\mu}_1 = -1 & \hat{\mu}_2 = 0,5 \\ \hat{\sigma}_1^2 = 1 & \hat{\sigma}_2^2 = 0,25 \end{cases}$$

$$\rightarrow \begin{cases} \hat{g}_1(x) = -\frac{1}{2}x^2 - x \\ \hat{g}_2(x) = -2x^2 + 2x + \ln 2 \end{cases} \xrightarrow{\hat{g}_1(x)=\hat{g}_2(x)} x = \begin{cases} -0,2 \\ 2,2 \end{cases}$$



$$\hat{c}(x) = \begin{cases} \mathbf{1} & x \notin [-0,2, 2,2] \\ \mathbf{2} & x \in [-0,2, 2,2] \end{cases} \approx c^*(x)$$

```

gaussian.m
function [eY]=gaussian(X,xl,Y,yl,a)
    ll=unique(xl); N=rows(X); M=rows(Y); D=columns(X);
    for c=ll'; ic=find(c==ll);
        Xc=X(find(xl==c),:); Nc=rows(Xc); pc(ic)=Nc/N;
        mc=mean(Xc); m(:,ic)=mc'; S{ic}=( (Xc-mc)'*(Xc-mc) )/Nc; end
    for i=1:length(a)
        for c=ll'; ic=find(c==ll); sS=a(i)*S{ic}+(1-a(i))*eye(D);
            gY(:,ic)=log(pc(ic))+compute_pxGc(m(:,ic),sS,Y); end
        [~,idY]=max(gY'); eY(i)=mean(ll(idY)~=yl)*100; end
end

```

```

ge.sh
#!/usr/bin/octave
if (nargin!=5) printf("ge.sh X xl as tr%% dev%%\n"); exit(1); end
arg_list=argv(); fX=arg_list{1}; load(fX);
fxl=arg_list{2}; load(fxl); a=str2num(arg_list{3});
tp=str2num(arg_list{4}); dp=str2num(arg_list{5});
N=rows(X); rand("seed",23); p=randperm(N); X=X(p,:); xl=xl(p,:);
Nt=round(tp/100*N); Nd=round(dp/100*N);
Xt=X(1:Nt,:); xlt=xl(1:Nt); Xd=X(N-Nd+1:N,:); xld=xl(N-Nd+1:N);
[ed]=gaussian(Xt,xlt,Xd,xld,a);
printf("\n  alpha dv-err\n-----\n");
for i=1:length(a); printf("%.1e %6.3f\n",a(i),ed(i)); end

```

```

time ./ge.sh 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

ge2.sh

```
#!/usr/bin/octave
if (nargin!=5) printf("ge2.sh X x1 Y y1 a\n"); exit(1); end;
arg_list=argv(); a=str2num(arg_list{5});
fX=arg_list{1}; load(fX); fx1=arg_list{2}; load(fx1);
fY=arg_list{3}; load(fY); fyl=arg_list{4}; load(fyl);
[e]=gaussian(X,x1,Y,y1,a);
printf("\n alpha te-err\n----- \n");
printf("%.1e %6.3f\n",a,e);
```

```
time ./ge2.sh 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

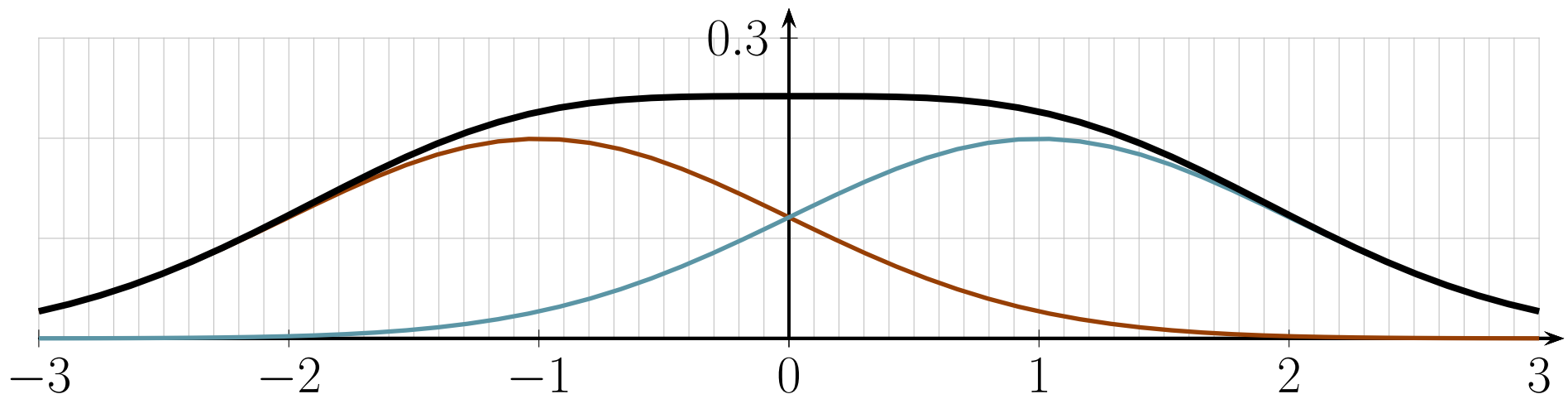
3.1. Modelo de mixtura finita

► 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 $\{\mathbf{x}_n\}$:

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

► *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 $-\frac{D}{2} \ln(2\pi)$:

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

- Consideremos priors y condicionales integrados en la conjunta:

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

- Así, Bayes se puede expresar como:

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

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

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

4.2. Estimación máximo-verosímil

- Modelo de K Gaussianas por clase, con $\Theta = \{(p_{ck}, \boldsymbol{\mu}_{ck}, \Sigma_{ck})\}$:

$$\begin{aligned} p_{\Theta}(\mathbf{x}) &= \sum_{c=1}^C p_{\Theta}(\mathbf{x}, c) \\ &= \sum_{c=1}^C \sum_{k=1}^K p_{ck} \mathcal{N}(\boldsymbol{\mu}_{ck}, \Sigma_{ck}; \mathbf{x}) \quad (p_{ck} > 0, \sum_c \sum_k p_{ck} = 1) \end{aligned}$$

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

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

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

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

con

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

y

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

El paso M maximiza Q como sigue:

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

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

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

► **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 $c^*(\mathbf{x})$, $L(\boldsymbol{\Theta})$ y EM, donde:

$$Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) = \sum_c \sum_{n:c_n=c} \sum_k z_{nck}^{(t)} (\ln p_{ck} + \ln \mathcal{N}(\boldsymbol{\mu}_{ck}, \alpha \Sigma_{ck} + (1 - \alpha) I_D; \mathbf{x}_n))$$

con

$$z_{nck}^{(t)} = \frac{p_{ck}^{(t)} \mathcal{N}(\boldsymbol{\mu}_{ck}^{(t)}, \alpha \Sigma_{ck}^{(t)} + (1 - \alpha) I_D; \mathbf{x}_n)}{\sum_{k'} p_{ck'}^{(t)} \mathcal{N}(\boldsymbol{\mu}_{ck'}^{(t)}, \alpha \Sigma_{ck'}^{(t)} + (1 - \alpha) I_D; \mathbf{x}_n)}$$

► **Cálculo robusto:** sea $\mathbf{a} \in \mathbb{R}^K$, a_k dado como $\ln a_k$; **In-sum-exp:**

$$\text{lse}(\mathbf{a}) \triangleq \ln \sum_k \exp(\ln a_k) = \max_{k'} \ln a_{k'} + \ln \sum_k \exp(\ln a_k - \max_{k'} \ln a_{k'})$$

La lse facilita el cálculo robusto de L ; tras el de la $z_{nck}^{(t)}$:

$$\frac{\exp(\ln a_k)}{\sum_{k'} \exp(\ln a_{k'})} = \frac{\exp(\ln a_k - \max_{k''} \ln a_{k''})}{\sum_{k'} \exp(\ln a_{k'} - \max_{k''} \ln a_{k''})}$$

► Log del numerador de z_{nck} para todo n : $\ln p_{ck} + \ln \mathcal{N}(\mu_{ck}, \Sigma_{ck}; \mathbf{x}_n)$

```

function [zk]=compute_zk(ic,k,pkGc,mu,sigma,X)
    D=columns(X); zk=log(pkGc{ic}(k))-.5*D*log(2*pi);
    m=mu{ic}(:,k); S=sigma{ic,k}; I=pinv(S);
    zk=zk+-.5*(sum((X*I).*X,2)+logdet(S)+m'*I*m)+X*I*m; end
    
```

► Aprendizaje del clasificador con mixturas de Gaussianas:

```

function [tee]=mixgaussian(X,xl,Y,yl,K,a)
    ll=unique(xl); C=rows(ll); N=rows(X); M=rows(Y); D=columns(X);
    rand('seed',23); S=cell(C,K);
    for c=1:C; ic=find(c==ll); idc=find(xl==c); Nc=rows(idc);
        pkGc{ic}(1:K)=Nc/(N*K); mu{ic}=X(idc(randperm(Nc,K)),:);
        S{ic,1:K}=a*cov(X(idc,:),1)/K+(1-a)*eye(D); end
    epsilon=1e-4; L=-inf; it=0;
    printf(" It          oL          L   trerr   teerr\n");
    printf("-----\n");
    do; oL=L; L=0;
        for c=1:C; ic=find(c==ll); idc=find(xl==c); Nc=rows(idc); Xc=X(idc,:);
            z=[]; for k=1:K; z(:,k)=compute_zk(ic,k,pkGc,mu,S,Xc); end
            mz=max(z,[],2); z=exp(z-mz); sz=sum(z,2); z=z./sz; L=L+sum(mz+log(sz));
            sz=sum(z); pkGc{ic}=sz/Nc; mu{ic}=(Xc'*z)./sz;
            for k=1:K; Sick=Xc-mu{ic}(:,k);
                S{ic,k}=a*(Sick'*Sick.*z(:,k))/sz(k)+(1-a)*eye(D); end; end
            L=L/N;
        for c=1:C; ic=find(c==ll);
            z=[]; for k=1:K; z(:,k)=compute_zk(ic,k,pkGc,mu,S,X); end
            mz=max(z,[],2); z=exp(z-mz); sz=sum(z,2); gtr(:,ic)=mz+log(sz);
            z=[]; for k=1:K; z(:,k)=compute_zk(ic,k,pkGc,mu,S,Y); end
            mz=max(z,[],2); z=exp(z-mz); sz=sum(z,2); gte(:,ic)=mz+log(sz); end
        [~,idx]=max(gtr'); tre=mean(ll(idx)~=xl)*100;
        [~,idy]=max(gte'); tee=mean(ll(idy)~=yl)*100;
        it=it+1; printf("%3d %14.5f %14.5f %6.3f %6.3f\n",it,oL,L,tre,tee);
    until ((L-oL)/abs(oL) < epsilon); end
    
```

```
#!/usr/bin/octave
if (nargin!=6) printf("mge.sh X x1 Ks as tr%% dv%%\n"); exit(1);end
arg_list=argv(); fX=arg_list{1}; load(fX); fx1=arg_list{2};
load(fx1); K=str2num(arg_list{3}); a=str2num(arg_list{4});
tp=str2num(arg_list{5}); dp=str2num(arg_list{6});
N=rows(X); rand("seed",23); p=randperm(N); X=X(p,:); x1=x1(p,:);
Nt=round(tp/100*N); Nd=round(dp/100*N);
Xt=X(1:Nt,:); xlt=x1(1:Nt); Xd=X(N-Nd+1:N,:); xld=x1(N-Nd+1:N);
printf("\n K      alpha dv-err\n--- -----\n");
for i=1:length(a); for k=1:length(K)
    [ed]=mixgaussian(Xt,xlt,Xd,xld,K(k),a(i));
    printf("%3d %.1e %7.2f\n",K(k),a(i),ed);
end; end
```

```
time ./mge.sh 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	-761411.65839	5.598	6.317			
3	-761411.65839	-761411.65839	5.598	6.317			
1	1.0e-05	6.32					
1	-Inf	-620842.17812	3.920	4.267			
2	-620842.17812	-313496.61124	3.920	4.267			
3	-313496.61124	-313496.61124	3.920	4.267			
1	1.0e-04	4.27					
1	-Inf	-179276.08786	5.115	6.383			
2	-179276.08786	-91068.99539	5.115	6.383			
3	-91068.99539	-91068.99539	5.115	6.383			
1	1.0e-03	6.38					

real 5m18,883s
user 10m40,900s
sys 2m46,716s

```
time ./mge.sh 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	-729422.61287	5.467	5.967
3		-729422.61287	-717831.22165	5.237	5.700
4		-717831.22165	-709872.95006	5.050	5.433
5		-709872.95006	-705702.24815	4.811	5.233
6		-705702.24815	-703296.42735	4.681	5.350
24		-694985.40103	-694937.00986	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	-306569.98867	3.563	4.117
3		-306569.98867	-302975.78644	3.585	4.183
4		-302975.78644	-299359.65835	3.507	4.117
5		-299359.65835	-296544.35092	3.393	3.917
6		-296544.35092	-294417.88591	3.311	3.867
16		-290453.21215	-290426.63775	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	-89896.41150	4.809	6.317
3		-89896.41150	-89824.89472	4.806	6.317
4		-89824.89472	-89701.92471	4.806	6.317
5		-89701.92471	-89505.01123	4.759	6.350
6		-89505.01123	-89271.32983	4.709	6.250
24		-87989.24678	-87981.19489	4.533	5.967
2	1.0e-03	5.97			
real		66m33,322s			
user		141m32,801s			
sys		37m33,655s			

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.