UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Perceptron algorithm: application to classification tasks

DSIC
Departamento de Sistemas Informáticos y Computación

Formative objectives

- To implement linear classifiers
- To code the Perceptron algorithm
- To apply the Perceptron algorithm to classification tasks

Contents

1 Linear discriminant functions 3

2 Perceptron algorithm 5

3 Application to classification tasks: OCR 7

3.1 Training 8

3.2 Error estimation 10

3.3 Effect of α 11

3.4 Effect of b 12

3.5 Training the final classifier 13

4 Exercise: application to other tasks 14

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Linear discriminant functions

Any classifier can be represented as:

$$c(x) = \arg\max_{c} \; g_c(x)$$

where each class c uses a ***discriminant function*** $g_c(x)$ that measures to which extent the object x belongs to class c

The most common discriminant functions are ***linear*** (with $\boldsymbol{x}$):

$$g_c(\boldsymbol{x}) = \boldsymbol{w}_c^t \boldsymbol{x} + w_{c0} \qquad \text{where} \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix} \quad \text{and} \quad \boldsymbol{w_c} = \begin{pmatrix} w_{c1} \\ \vdots \\ w_{cD} \end{pmatrix}$$

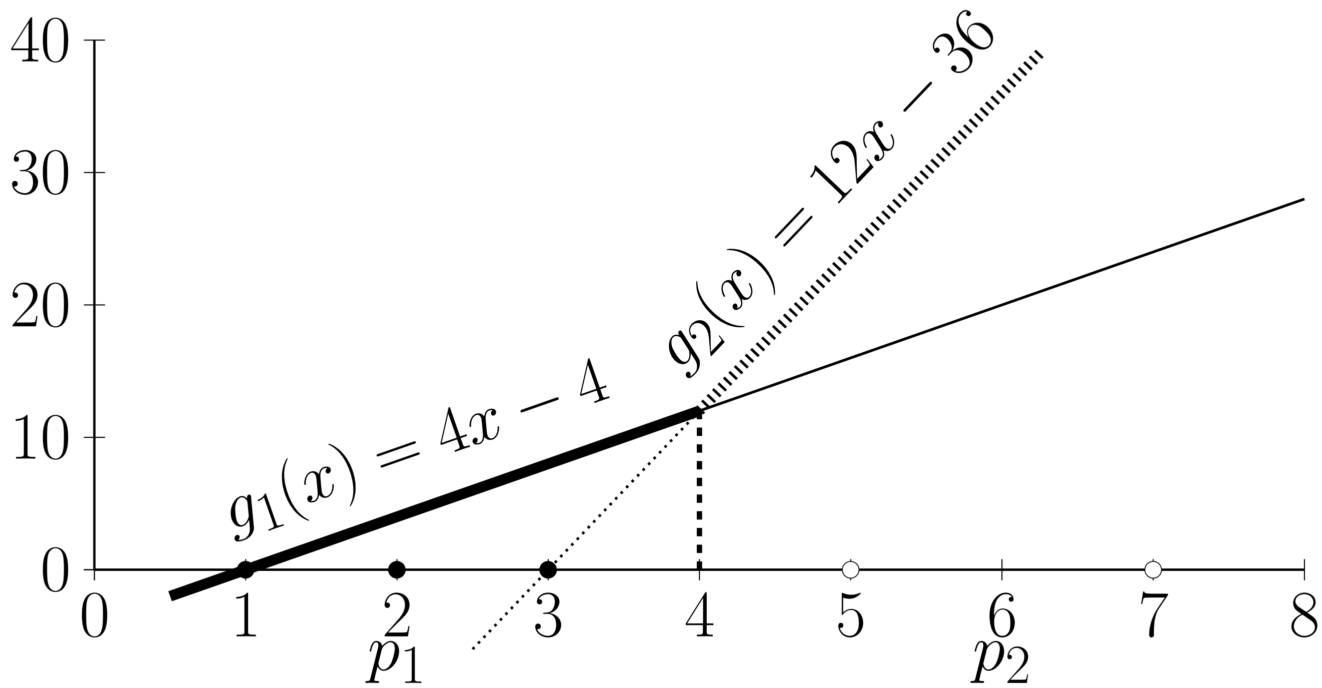
using ***homogeneous*** notation:

$$g_c(\mathbf{x}) = \mathbf{w}_c^t \mathbf{x} \qquad \text{where} \quad \mathbf{x} = \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix} \quad \text{and} \quad \mathbf{w}_c = \begin{pmatrix} w_{c0} \\ \boldsymbol{w}_c \end{pmatrix}$$

linmach.m

```
function cstar=linmach(w,x)
  C=columns(w); cstar=1; max=-inf;
  for c=1:C
    g=w(:,c)'*x;
    if (g>max) max=g; cstar=c; endif; end
endfunction
```

```
w=[-4 -36; 4 12];
for x=1:8;
  printf("c(%d)=%d\n",x,linmach(w,[1 x]')); end
```




```
c(1)=1
c(2)=1
c(3)=1
c(4)=1
c(5)=2
c(6)=2
c(7)=2
c(8)=2
```

2 Perceptron algorithm

Input: $\{(\mathbf{x}_n, c_n)\}_{n=1}^{N}, \ \{\mathbf{w}_c\}_{c=1}^{C}, \ \alpha \in \mathbb{R}^{>0}$ and $b \in \mathbb{R}$

Output: $$\{\mathbf{w}_c\}^* = \arg\min_{\{\mathbf{w}_c\}} \sum_n \left[\max_{c \neq c_n} \mathbf{w}_c^t \mathbf{x}_n + b > \mathbf{w}_{c_n}^t \mathbf{x}_n \right]$$

$$[P] = \begin{cases} 1 & \text{if } P = \text{true} \\ 0 & \text{if } P = \text{false} \end{cases}$$

Method:

repeat

for all data sample $\mathbf{x}_n$

$err = \texttt{false}$

for all class c different from c_n

if $\mathbf{w}_c^t \mathbf{x}_n + b > \mathbf{w}_{c_n}^t \mathbf{x}_n$: $\mathbf{w}_c = \mathbf{w}_c - \alpha \cdot \mathbf{x}_n$; $err = \texttt{true}$

if err: $\mathbf{w}_{c_n} = \mathbf{w}_{c_n} + \alpha \cdot \mathbf{x}_n$

until there are no misclassified data samples
(or until the pre-defined maximum number of iterations is reached)

perceptron.m

```
function [w,E,k]=perceptron(data,b,a,K,iw)
  [N,L]=size(data); D=L-1;
  labs=unique(data(:,L)); C=numel(labs);
  if (nargin<5) w=zeros(D+1,C); else w=iw; end
  if (nargin<4) K=200; end;
  if (nargin<3) a=1.0; end;
  if (nargin<2) b=0.1; end;
  for k=1:K
    E=0;
    for n=1:N
      xn=[1 data(n,1:D)]';
      cn=find(labs==data(n,L));
      er=0; g=w(:,cn)'*xn;
      for c=1:C; if (c!=cn && w(:,c)'*xn+b>g)
        w(:,c)=w(:,c)-a*xn; er=1; end; end
      if (er)
        w(:,cn)=w(:,cn)+a*xn; E=E+1; end; end
    if (E==0) break; end; end
endfunction
```

```
data=[0 0 1;1 1 2];
[w,E,k]=perceptron(data);
disp(w); printf("E=%d k=%d\n",E,k);
```

```
   1  -1
  -1   1
  -1   1
E=0 k=3
```

3 Application to classification tasks: OCR

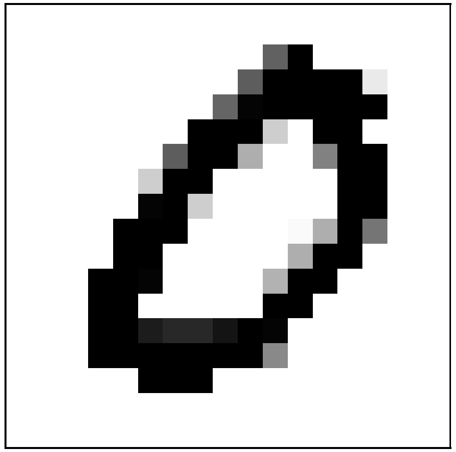
The corpus `OCR_14x14` is a matrix `data` with 1000 rows (samples) and 197 columns (196 features plus class label):

```
head -n 7 OCR_14x14
```

```
# name: data
# type: matrix
# rows: 1000
# columns: 197
 0 0 0 0 0 0 0 0 0.62 1 0 0 0 0 0 0 0 0 0 0 0 0.63 1 1 1 1   ... 0 0
 0 0 0 0 0 0 0.38 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0  ... 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.91 1 0.99       ... 0 0
```

Each sample represents a grey-scale image of a handwritten digit normalised to 14x14 and read from left to right and top to bottom:

```
load("OCR_14x14");
[N,L]=size(data); D=L-1;
I=reshape(data(1,1:D),14,14)';
imshow(1-I);
rand("seed",23); data=data(randperm(N),:);
for n=1:1000
  I=reshape(data(n,1:196),14,14)';
  imshow(1-I); pause(0.5);
end
```



3.1 Training

```
load("OCR_14x14"); [N,L]=size(data); D=L-1;
ll=unique(data(:,L)); C=numel(ll);
rand("seed",23); data=data(randperm(N),:);
[w,E,k]=perceptron(data(1:round(.7*N),:));
save_precision(4); save("percep_w","w");
output_precision(2); w
```

```
w =
-39.00 -30.00 -31.00 -35.00 -34.00 -27.00 -33.00 -30.00 -46.00 -31.00
  0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 -1.00   0.00  -2.00   0.00  -1.00   2.00   0.00   0.00  -2.00   0.00
 -1.38  -1.69  -2.53  -1.84  -1.53   0.69  -0.69   4.15  -3.22  -1.69
 -1.69  -1.77   0.54  -3.46  -1.46  -3.00  -2.00   5.15  -3.46  -3.00
 -3.54  -7.48  -1.15  -3.00   0.25  -6.71  -5.08   1.77  -1.85  -8.41
...
```

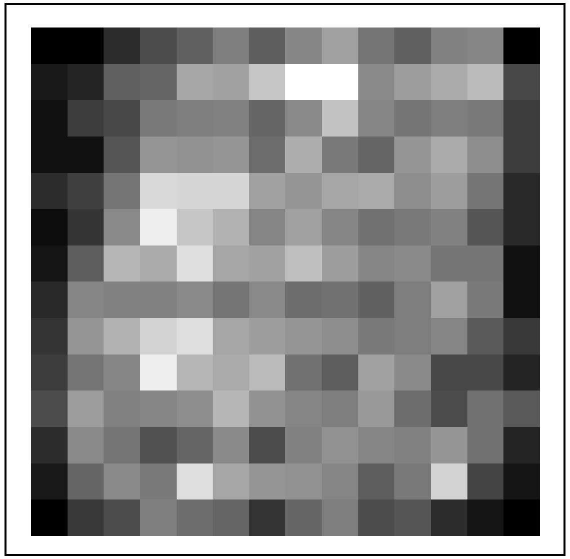
The score that $\mathbf{x}$ (with $x_0 = 1$) belongs to class (digit) c is $g_c(\mathbf{x}) = \mathbf{w}_c^t\mathbf{x}$, where $\mathbf{w}_c$ is the $(c+1)$-th column of w:

```
load("OCR_14x14"); load("percep_w"); [N,L]=size(data); D=L-1;
for n=1:N; xn=[1 data(n,1:D)]';
  for c=0:9 printf("g_%d(x_%d)=%.0f ",c,n,w(:,c+1)'*xn); end
  printf("\n"); end
```

```
g_0(x_1)=-665 g_1(x_1)=-840 g_2(x_1)=-813 g_3(x_1)=-798 ...
g_0(x_2)=-518 g_1(x_2)=-659 g_2(x_2)=-592 g_3(x_2)=-604 ...
g_0(x_3)=-635 g_1(x_3)=-802 g_2(x_3)=-826 g_3(x_3)=-740 ...
...
```

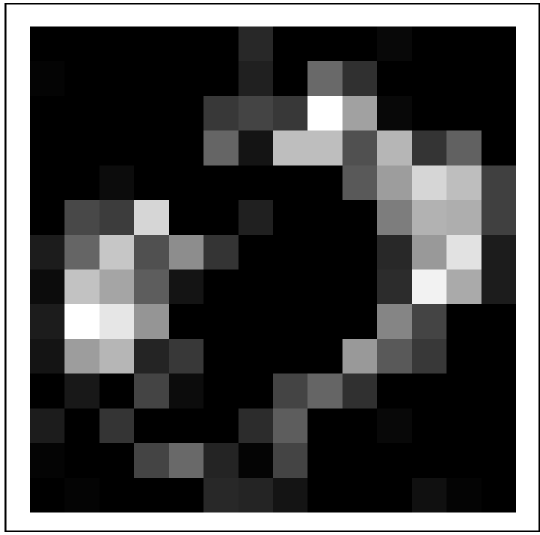
Weigths $\{w_{cd}\}$ with greater variability in c possess a more significant discriminative effect than those with lower variability. Right: $\sigma(\{w_{1d}, \ldots, w_{Cd}\})$ for each $d > 0$.

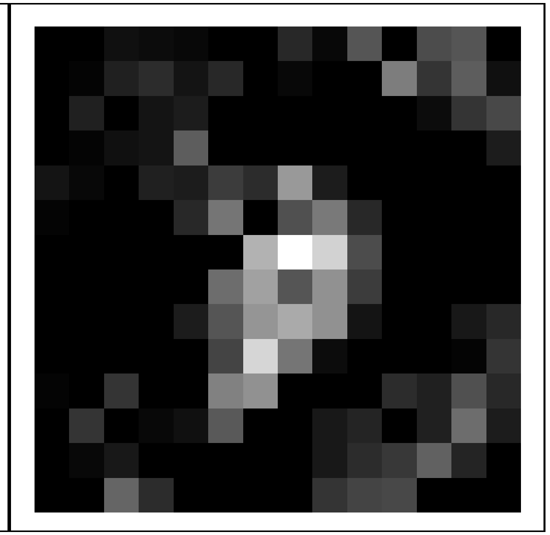
```
load("percep_w"); sw=std(w(2:197,:),1,2);
I=reshape(sw,14,14)'; imshow(I,[,]);
```



Weights in class c comparatively greater than those of other classes (above average) point features discriminating class c against other classes (grey pixels on left image). Right image is the opposite.

```
load("percep_w");
mw=mean(w(2:197,:),2);
for c=0:9
  wc=w(2:197,c+1);
  pw=max(0,wc-mw);
  I=reshape(pw,14,14)';
  imshow(I,[,]); pause(3);
  nw=-min(0,wc-mw);
  I=reshape(nw,14,14)';
  imshow(I,[,]); pause(3);
end
```





3.2 Error estimation

Estimation of classification error using the samples not devoted to training (*test samples*):

```
load("OCR_14x14");
[N,L]=size(data); D=L-1;
ll=unique(data(:,L));
C=numel(ll); rand("seed",23);
data=data(randperm(N),:);
M=N-round(.7*N); te=data(N-M+1:N,:);
load("percep_w"); rl=zeros(M,1);
for m=1:M
  tem=[1 te(m,1:D)]';
  rl(m)=ll(linmach(w,tem)); end
[nerr m]=confus(te(:,L),rl)
```

```
nerr =  17
m =
 37   0   0   0   0   0   0   0   0   0
  0  29   0   0   0   0   0   0   1   0
  0   1  32   0   0   0   0   0   0   0
  1   0   1  26   0   2   0   0   0   0
  0   0   0   0  27   0   0   1   0   0
  0   0   0   2   0  26   0   0   0   0
  0   0   0   0   0   0  28   0   0   0
  0   0   0   0   0   0   0  27   0   2
  1   2   0   0   0   1   0   0  24   2
  0   0   0   0   0   0   0   0   0  27
```

95% confidence interval for the estimated error:

```
nerr=17; M=300; output_precision(2);
m=nerr/M
s=sqrt(m*(1-m)/M)
r=1.96*s
printf("I=[%.3f, %.3f]\n",m-r,m+r);
```

```
m =  0.057
s =  0.013
r =  0.026
I=[0.031, 0.083]
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

3.3 Effect of α

```
#!/usr/bin/octave -qf
load("OCR_14x14"); [N,L]=size(data); D=L-1;
ll=unique(data(:,L)); C=numel(ll);
rand("seed",23); data=data(randperm(N),:);
NTr=round(.7*N); M=N-NTr; te=data(NTr+1:N,:);
printf("#       a    E    k Ete\n");
printf("#------- --- --- ---\n");
for a=[.1 1 10 100 1000 10000 100000]
  [w,E,k]=perceptron(data(1:NTr,:),0.1,a); rl=zeros(M,1);
  for n=1:M rl(n)=ll(linmach(w,[1 te(n,1:D)]')); end
  [nerr m]=confus(te(:,L),rl);
  printf("%8.1f %3d %3d %3d\n",a,E,k,nerr);
end
```

```
#       a   E   k Ete
#------- --- --- ---
     0.1   0  16  20
     1.0   0  13  17
    10.0   0   8  15
   100.0   0  12  16
  1000.0   0  12  16
 10000.0   0  12  16
100000.0   0  12  16
```

The parameter α, $\alpha > 0$, does ***not*** have a notable effect in the behaviour of Perceptron.

3.4 Effect of b

```
#!/usr/bin/octave -qf
load("OCR_14x14"); [N,L]=size(data); D=L-1;
ll=unique(data(:,L)); C=numel(ll);
rand("seed",23); data=data(randperm(N),:);
NTr=round(.7*N); M=N-NTr; te=data(NTr+1:N,:);
printf("#       b    E    k Ete\n");
printf("#------- --- --- ---\n");
for b=[.1 1 10 100 1000 10000 100000]
  [w,E,k]=perceptron(data(1:NTr,:),b); rl=zeros(M,1);
  for n=1:M rl(n)=ll(linmach(w,[1 te(n,1:D)]')); end
  [nerr m]=confus(te(:,L),rl);
  printf("%8.1f %3d %3d %3d\n",b,E,k,nerr);
end
```

```
#       b   E   k Ete
#------- --- --- ---
     0.1   0  13  17
     1.0   0  16  20
    10.0   0  10  19
   100.0   0  22  16
  1000.0   0 125  13
 10000.0 165 200  10
100000.0 544 200  29
```

Parameter b ***does*** have a notable effect.

If samples are linearly separable, select b comparatively high (i.e. $b = 1000$) so that Perceptron converges ($E = 0$)

3.5 Training the final classifier

Train our ***final*** classifier with all samples:

```
load("OCR_14x14");
[w,E,k]=perceptron(data,1000); [E k]
save_precision(4);
save("OCR_14x14_w","w");   # filename = corpusname_w
```

Let us check the weights of the final classifier:

```
load("OCR_14x14_w")
output_precision(2); w
```

```
w =
 -1847.00 -1622.00 -1686.00 -1847.00 -1736.00 -1527.00 -1643.00 -1657.00 -2207.00 -1853.00
     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
    -9.00   -14.33   -52.08   -22.16   -18.16    48.92    -4.08   -36.67   -49.08   -35.08
   -18.68   -74.45   -63.09   -52.68   -51.95     5.93   -22.55    74.31   -51.42   -48.13
   -35.28  -118.40    17.82   -78.14   -22.17   -76.07   -74.11   165.60   -67.45   -56.44
  -109.60  -189.10   -80.59   -73.37    21.95  -151.10   -91.60    66.40    61.42  -208.20
  -109.80  -246.70  -187.70  -130.10  -193.40  -319.20  -255.80  -185.80  -111.50   -88.27
  -336.50  -361.40  -458.70  -292.30  -415.70  -458.20  -325.30  -506.00  -292.60   -85.41
  -565.50  -346.70  -491.60  -592.10  -678.20  -442.80  -495.90  -789.50  -361.70  -310.00
  -520.70  -477.40  -410.40  -508.90  -668.20  -575.40  -548.10  -460.10  -437.00  -346.20
  -533.90  -472.60  -489.30  -522.00  -437.60  -495.90  -526.70  -504.80  -579.00  -534.40
  -284.10  -120.30  -285.90  -276.30  -139.90  -151.00  -236.40  -163.30  -282.40  -278.80
  -124.00    34.48  -179.10  -246.80   -67.34   148.80  -154.60  -108.30  -117.00   -68.68
     0.00    -2.00    -4.00    -4.00     2.00     3.00     0.00     0.00     0.00     0.00
     0.00     0.00     0.00   -24.80     0.00     0.00     0.00    24.80     0.00   -13.64
    -1.52   -10.15    -1.53   -23.04    11.88   -10.65    -3.70    22.87    -1.46   -11.61
   -55.64   -79.97   -18.85  -146.60   -60.85   -31.97   -77.04   121.10  -112.10   -82.56
      ...
```

4 Exercise: application to other tasks

Consider the following 4 data sets stating classification tasks:

1. ***expressions:*** 225 facial expressions represented by 4096-D vectors and classified into 5 classes (1=surprise, 2=happiness, 3=sadness, 4=anguish and 5=displeasure).

2. ***gauss2D:*** 4000 synthetic samples from two equally-probable classes representing a bidimensional Gaussian distribution.

3. ***gender:*** 2836 facial expressions represented by 1280-D vectors and classified by gender.

4. ***videos:*** 7985 basket and non-basket videos represented by 2000-D vectors computed from local-feature histograms.

Assignment

1. Create a script `experiment.m` in Octave to automatically apply the Perceptron algorithm to other tasks. This script takes as input a data set, and the range of values for α and b:

```
#!/usr/bin/octave -qf
if (nargin!=3)
 printf("Usage: ./experiment.m <data> <alphas> <bes>\n");
 exit(1);
end
arg_list=argv();
data=arg_list{1};
as=str2num(arg_list{2});
bs=str2num(arg_list{3});
load(data); [N,L]=size(data); D=L-1;
...
for a=as
  for b=bs
  [w,E,k]=perceptron(data(1:NTr,:),b,a); rl=zeros(M,1);
  ...
```

From the command shell, execute:

```
$ ./experiment.m OCR_14x14 "[.1 1 10 100 1000 10000]" "[0.1]"
```

Assignment

A possible output of results from your script should be:

```
#       a         b   E   k Ete Ete (%)    Ite (%)
#------- -------- --- --- --- ------- ----------
      0.1       0.1   0  16  20     6.7 [3.8, 9.5]
      1.0       0.1   0  13  17     5.7 [3.1, 8.3]
     10.0       0.1   0   8  15     5.0 [2.5, 7.5]
    100.0       0.1   0  12  16     5.3 [2.8, 7.9]
   1000.0       0.1   0  12  16     5.3 [2.8, 7.9]
  10000.0       0.1   0  12  16     5.3 [2.8, 7.9]
```

2. Compute a summary table with the best results approximately as the following:

task	Ete (%)	Ite (%)
OCR_14x14	4.3	[2.0, 6.6]
expressions	3.0	[0.0, 7.1]
gauss2D	9.0	[7.4, 10.6]
gender	6.1	[4.5, 7.7]
videos	18.7	[17.1, 20.2]

Exam

- The lab exam involves to modify your script `experiment.m` to carry out an experiment with an already known data set or a new one.

- The day of the exam you must turn in:
  - Original version of `experiment.m`
  - Modified version of `experiment.m`
  - Achieved results and comments on the results