

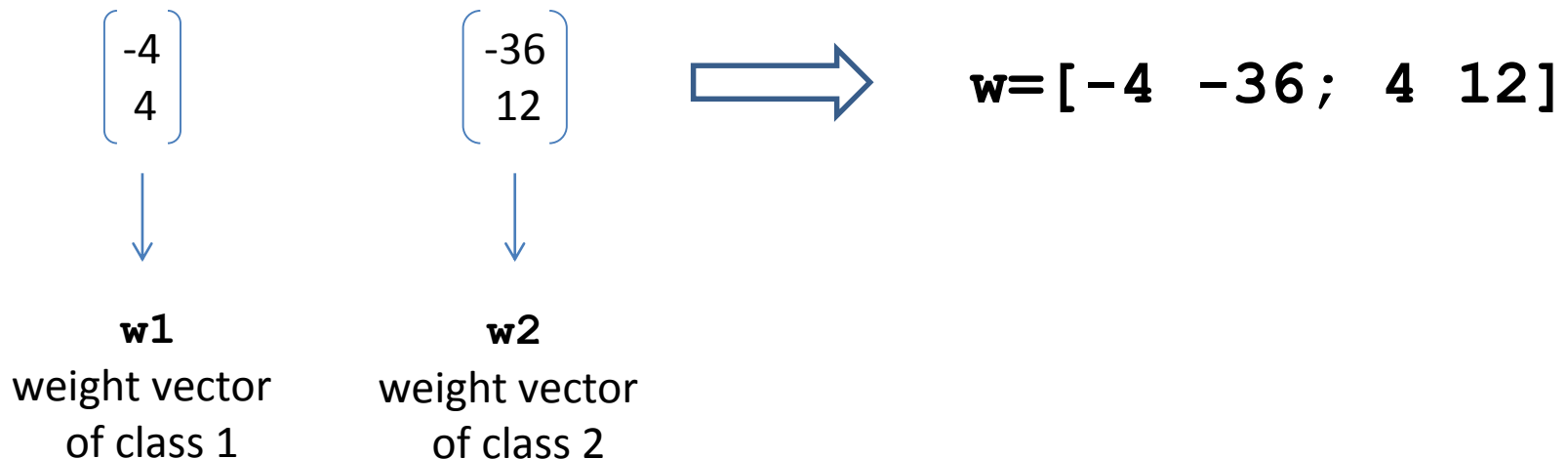
Lab. assignment 2

Perceptron algorithm (ancillary file)

Linear functions

`linmach(w, x)`

- **w**: list of weight vectors (one weight vector per class)



- **x**: sample. In the example, samples are one-dimensional vectors. A sample can be:

(1) (2) (3) (4) (5) (6) (7) (8) \Rightarrow $[1 \ 1] [1 \ 2] [1 \ 3] \dots$
homogeneous notation

- It **returns the class number of the sample**

Perceptron: input data

`perceptron (data,b,a,K,iw)`

data

- It is the matrix that contains the samples
- Each **row** contains one **sample** (one feature vector)
- Each column is a feature (the last column represents the class of the sample)
- Example:
 - `data=[0 0 1; 1 1 2];`
 - 0 0 1: sample 1, two features (0 0), class 1
 - 1 1 2: sample 2, two features (1 1), class 2
 - `[N,L]=size (data);`
 - N= 2 samples
 - L= 3 columns (the third column -L- is the class)
 - D=L-1=2 dimensions

Perceptron: input data

data

- `labs=unique(data(:,L));`
list that contains the elements of the column **L** that are different
in the example `labs = 1 2` because each sample belongs to a different class
- `C=numel(labs);`
number of classes
C=2 classes

Perceptron: input data

b

- It is the **margin**; if this parameter is not specified in the input data then **b=0.1**

a

- It is the **learning rate**; if this parameter is not specified in the input data then **a=1.0**

K

- It is the number of **iterations**; if this parameter is not specified in the input data then **K=200**

iw

- It is the **initial weight vectors**; if this parameter is not specified in the input data then **iw** is a set of null vectors

The algorithm

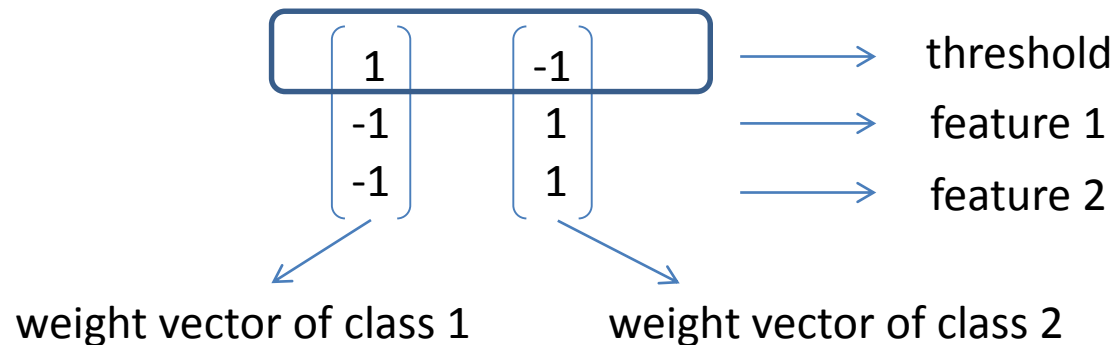
```
for k=1:K           for K iterations
    E=0;
    for n=1:N       for N samples
        xn=[1 data(n,1:D)]'; adds 1 to the feature vector of sample n
        cn=find(labs==data(n,L)); finds the class of sample n
        er=0;
        g=w(:,cn)'*xn; weight vector of class 'cn' applied to 'xn'
        for c=1:C;   classes
            if (c!=cn && w(:,c)'*xn+b>g) checking with the other classes
                w(:,c)=w(:,c)-a*xn; update weight vector of class 'c'
                er=1;
            end;
        end
        if (er)
            w(:,cn)=w(:,cn)+a*xn; update weight vector of the sample's class
            E=E+1;
        end;
    end
    if (E==0) break; end;
end
```

The output

$[w, E, k] = \text{perceptron}(\text{data}) ;$

w

- The **resulting weight vectors**; in the example, the final weight vectors are:



E

- The number of misclassified samples (if $E=0$, it means Perceptron algorithm reached convergence; otherwise, $E > 0$)

k

- The number of iterations

OCR task

OCR_14x14 corpus

- 1000 samples = 1000 rows (images of handwritten digits)
- digits from 0 to 9 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) → 10 classes
- 1 sample → 196 columns (features) + 1 last column (class) → 197 columns
- 1 sample → image normalized to 14 x 14 (196 columns)

OCR task: training (learning the classifier)

We use only **700** samples out of the 1000 samples of data for learning the classifier (**training set**). The other 300 are for testing the results of the learnt classifier

```
[w, E, k] = perceptron (data (1 : round (.7*N) , :)) ;
```

The result of calling the **Perceptron** algorithm is **[w, E, k]**

w =										thres. f1 f2
-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	
...										
↓	↓	↓							↓	
w0	w1	w2							w9	

OCR task: training (learning the classifier)

$$g_0(\mathbf{x}_1) = \mathbf{w}_0^t * \mathbf{x}_1 = (-39.00 \ 0.00 \ 0.00 \ -1.00 \ \dots) * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \end{bmatrix}$$

$$g_1(\mathbf{x}_1) = \mathbf{w}_1^t * \mathbf{x}_1 = (-30.00 \ 0.00 \ 0.00 \ 0.00 \ \dots) * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \end{bmatrix}$$

OCR task: error estimation (testing)

We use the remaining 300 samples for testing the classifier (**testing set**)

`M=N-round (.7*N) ;` \rightarrow `M = 300`

`te=data (N-M+1:N, :)` ; \rightarrow `te` goes from sample 701 to 1000

For each of the 300 samples

- we add 1 to the feature vector of the sample
- we call `linmach(w,x)` with the weight vector calculated by the Perceptron and the sample \rightarrow this calculates the class of the sample

Finally, we call `[nerr m]=confus(te(:,L),r1)` that compares the actual class of the samples with the estimated class by the classifier

OCR task: error estimation (testing)

```
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"); r1=zeros(M,1);  
for m=1:M  
    tem=[1 te(m,1:D)]';  
    r1(m)=ll(linmach(w,tem)); end  
[nerr m]=confus(te(:,L),r1)
```

M=300

te is the samples from
row 701 to 1000

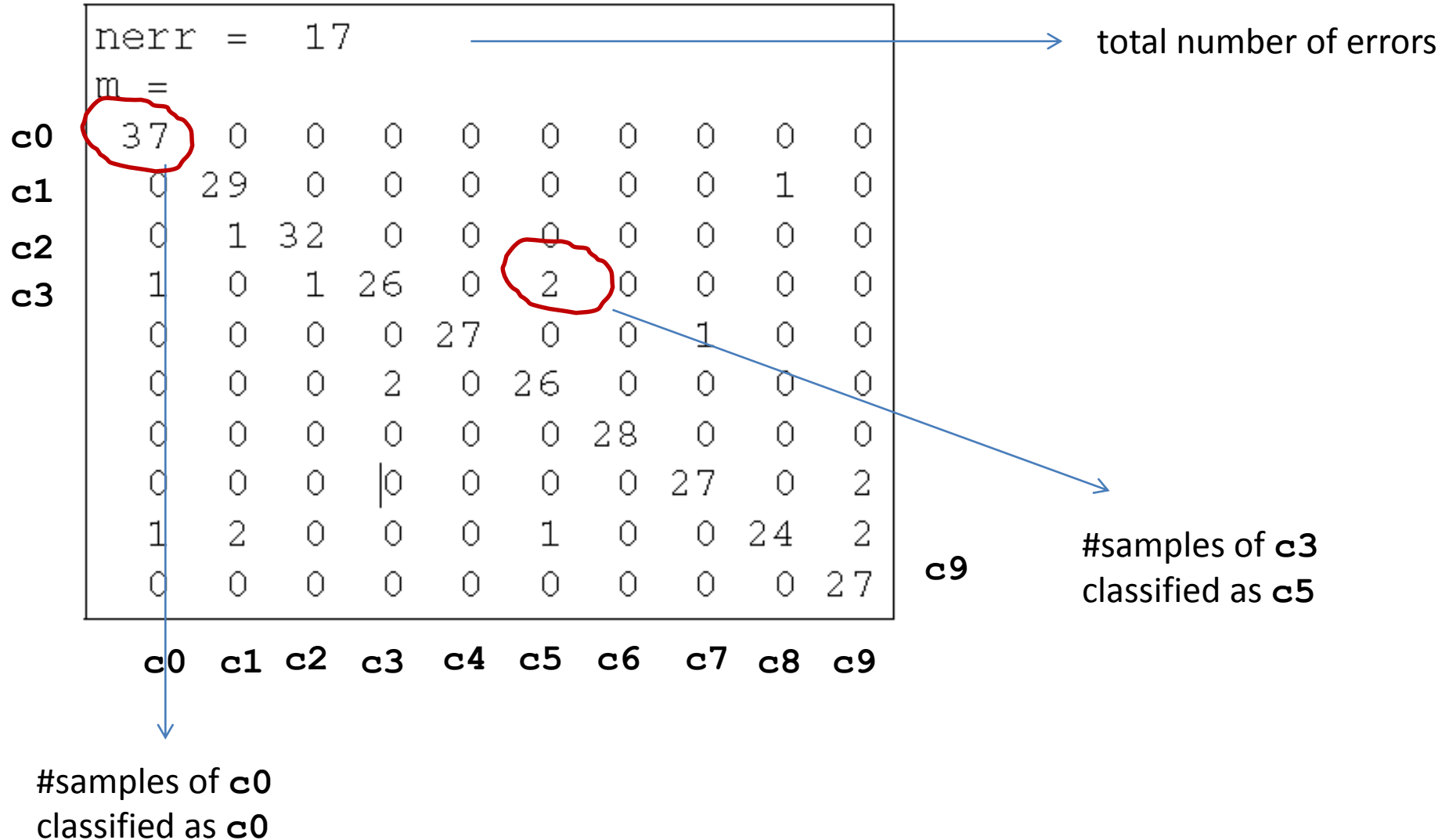
adds 1 to the first
element of the feature
vector

computes the class of
the sample

calculated
weight vectors **w**

compares the actual class
(te(:,L)) with the
estimated class (r1)

OCR task: error estimation (testing)



OCR task: effects 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
```

#misclassified samples in the
training set (convergence)

#errors in the testing set (nerr)

#	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

OCR task: effects 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 ($E = 0$)

Estimated error

Ne : number of errors

N : number of samples

\hat{p} : estimated error

p : true probability of error



$$\hat{p} = Ne / N$$

How confident is \hat{p} to p ? How close is \hat{p} to the true probability error of a real-world problem?

A **confidence interval (CI)** is a type of interval estimate (of a population parameter) that is computed from the observed data. Confidence intervals consist of a range of values (interval) that act as good estimates of the unknown population parameter.

Estimated error

The desired level of confidence is set by the researcher (not determined by data). Most commonly, the **95% confidence level** is used.

95 % confidence interval:

$$P(\hat{p} - \epsilon \leq p \leq \hat{p} + \epsilon) = 0.95; \quad \epsilon = 1.96 \sqrt{\frac{\hat{p}(1 - \hat{p})}{N}}$$

$$\text{Confidence Interval} \longrightarrow I = [\hat{p} \pm \epsilon]$$

For example: we observe 50 wrong decisions (errors) of a 1000 data set:

$$\hat{p} = Ne / N = 50/1000 = 0.05$$

With a 95% confidence we can ensure the true error is:

$$p = 0.05 \pm 1.96 \sqrt{\frac{0.05 \cdot 0.95}{1000}} = 0.05 \pm 0.014 \quad (5 \% \pm 1,4 \%)$$

**Confidence
Interval**

Estimated error

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]
```

$r = \varepsilon$


$m = \hat{p}$ (estimated error)

Results

#	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]



learning rate (α)




margin



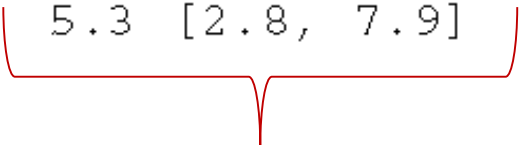
misclassified samples in the
training set with Perceptron



iterations



#errors in
the testing set



estimated error
and interval

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

```
#!/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);
        ...
```

- set training set and testing set
- printf headers
- estimate class
- calculate error (nerr)
- calculate estimated error and confidence interval
- printf results

From the command shell, execute:

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

Assignment

#	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]

↓

learning rate (α)

↓

margin

↓

iterations

misclassified samples in the training set with Perceptron

↓

#errors in the testing set

↓

estimated error and interval

Assignment

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

task	Ete (%)	Ite (%)	a	b
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]		