Optimization Techniques for Data Mining

Master in Innovation and Research in Informatics

Unconstrained Optimization
Lab Assignment
Pattern recognition with Single Layer Neural Network (SLNN)

F.-Javier Heredia

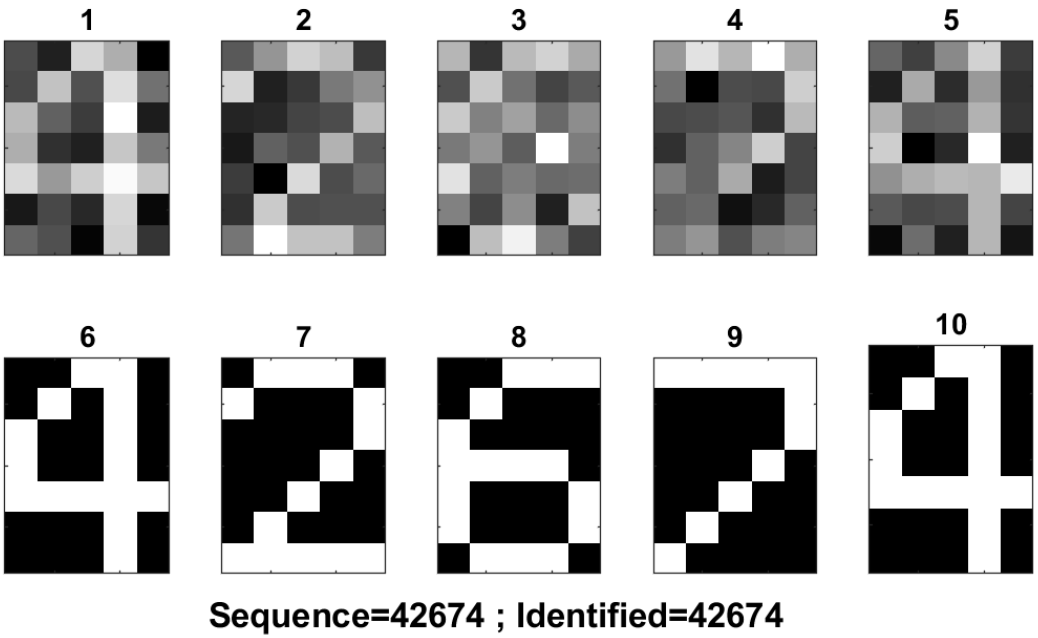
(Jordi Castro, Daniel Ramón)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Estadística
i Investigació Operativa

Pattern recognition with SLNN

Presentation

- The aim of this project is to develop an application, based on the unconstrained optimization algorithms studied in this course, that allow to recognize the numbers in a sequence of blurred digits:

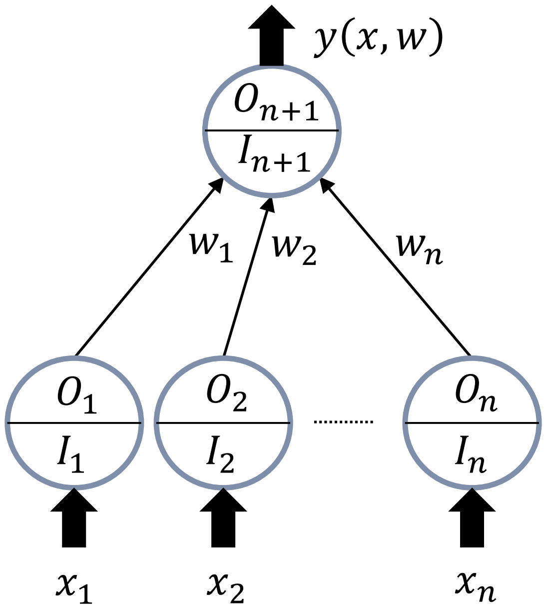


Original numbers

Identified by the neural network

- The procedure to achieve that goal will be to formulate a **Single Layer Neural Network** that is going to be **trained to recognize** the different numbers with **First Derivative Optimization methods**.

Single layer Neural Network (SLNN): architecture



- **Input signal:**

$$I_i = x_i, i = 1,2,\dots,n \;\; ; \;\; I_{n+1} = \sum_{i=1}^{n} w_i \cdot O_i$$

- **Activation function (sigmoid function) :**

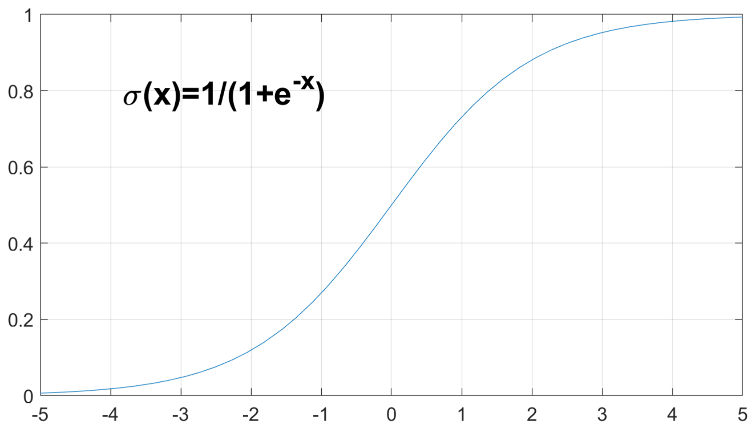
$$O_i = \sigma(I_i) \;, \qquad \sigma(x) = 1/(1+e^{-x})$$

- **Output signal: assumed to be binary**

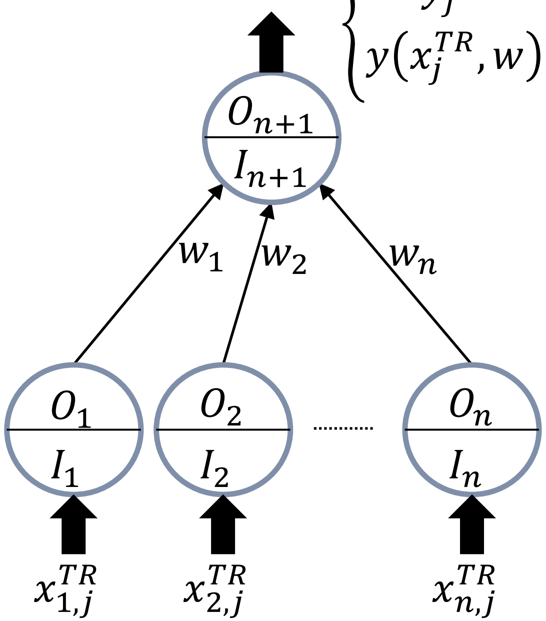
$$y(x,w) = \quad \sigma(I_{n+1}) = \sigma\left(\sum_{i=1}^{n} w_i O_i\right) = \sigma\left(\sum_{i=1}^{n} w_i \cdot \sigma(x_i)\right)$$

$$= \left(1 + e^{-\left(\sum_{i=1}^{n} w_i \cdot \sigma(x_i)\right)}\right)^{-1}$$

$$= \left(1 + e^{-\left(\sum_{i=1}^{n} w_i \cdot (1+e^{-x_i})^{-1}\right)}\right)^{-1}$$



SLNN: training



$$\begin{cases} y_j^{TR} & \text{data} \\ y(x_j^{TR}, w) & \text{model} \end{cases}$$

- **Training data set, size p:**

$$X^{TR} = [x_1^{TR}, x_2^{TR}, \dots, x_p^{TR}] = \begin{bmatrix} x_{1,1}^{TR} & x_{1,2}^{TR} & \cdots & x_{1,p}^{TR} \\ x_{2,1}^{TR} & x_{2,2}^{TR} & \cdots & x_{2,p}^{TR} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1}^{TR} & x_{n,2}^{TR} & \cdots & x_{n,p}^{TR} \end{bmatrix}$$

$$y^{TR} = \begin{bmatrix} y_1^{TR} & y_2^{TR} & \cdots & y_p^{TR} \end{bmatrix}^T$$

- **Loss function**: for a given (X^{TR}, Y^{TR})

$$L(X^{TR}, y^{TR}) = \min_{w \in \mathbb{R}^n} L(w; X^{TR}, y^{TR}) = \sum_{j=1}^{p} \left(y(x_j^{TR}, w) - y_j^{TR} \right)^2$$

- **Loss function with L2 regularization with param. λ:**

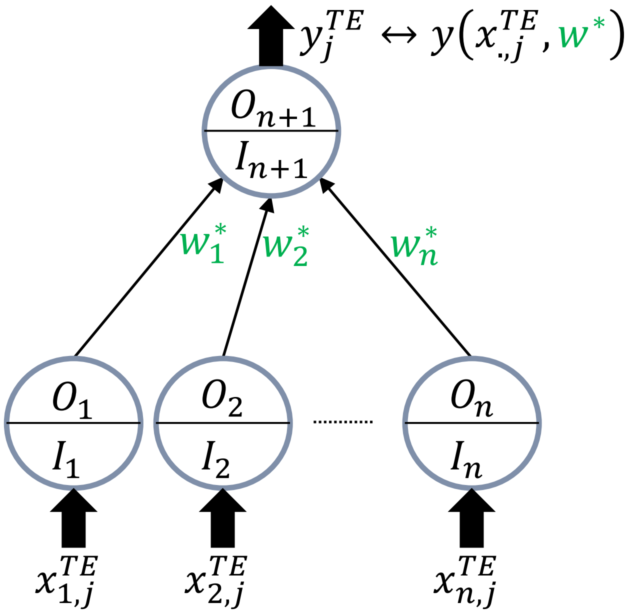
$$\tilde{L}(X^{TR}, y^{TR}, \lambda) = \min_{w \in \mathbb{R}^n} \tilde{L}(w; X^{TR}, y^{TR}, \lambda) = L(w; X^{TR}, y^{TR}) + \lambda \cdot \frac{\|w\|^2}{2}$$

- **Training accuracy (%)**: $w^* = \text{argmin}_{w \in \mathbb{R}^n} \tilde{L}(w; X^{TR}, y^{TR}, \lambda)$

$$\text{Accuracy}^{TR} = \frac{100}{p} \cdot \sum_{j=1}^{p} \delta_{\underbrace{\left[y(x_j^{TR}, w^*)\right]}_{\texttt{round()}}, y_j^{TR}}$$

where $\delta_{x,y} = \begin{cases} 1 & if\ x = y \\ 0 & if\ x \neq y \end{cases}$ (*Kronecker delta*).

SLNN : testing



- **Test data set, size q:**

$$X^{TE} = \left[x_1^{TE}, x_2^{TE}, \dots, x_q^{TE}\right] = \begin{bmatrix} x_{1,1}^{TE} & x_{1,2}^{TE} & \cdots & x_{1,q}^{TE} \\ x_{2,1}^{TE} & x_{2,2}^{TE} & \cdots & x_{2,q}^{TE} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1}^{TE} & x_{n,2}^{TE} & \cdots & x_{n,q}^{TE} \end{bmatrix}$$

$$y^{TE} = \begin{bmatrix} y_1^{TE} & y_2^{TE} & \cdots & y_q^{TE} \end{bmatrix}^T$$

- **Test accuracy (%):**

$$\text{Accuracy}^{TE} = \frac{100}{p} \cdot \sum_{j=1}^{p} \delta_{\left[y\left(x_j^{TE}, w^*\right)\right], y_j^{TE}}$$

- **Overfitting**: if $\text{Accuracy}^{TR} \gg \text{Accuracy}^{TE}$

SLNN : gradient (1/2)

- **Loss function (objective function):**

$$\tilde{L}(w; X^{TR}, y^{TR}, \lambda) = \sum_{j=1}^{p} \left(y\left(x_j^{TR}, w\right) - y_j^{TR} \right)^2 + \frac{\lambda}{2} \cdot \sum_{i=1}^{n} w_i^2$$

- **Gradient:**

$$\frac{\partial \tilde{L}(w; X^{TR}, y^{TR}, \lambda)}{\partial w_i} = \sum_{j=1}^{p} 2 \cdot \left(y\left(x_j^{TR}, w\right) - y_j^{TR} \right) \cdot \frac{\partial y\left(x_j^{TR}, w\right)}{\partial w_i} + \lambda \cdot w_i \qquad (1)$$

with

$$y\left(x_j^{TR}, w\right) = \left(1 + e^{-\left(\sum_{i=1}^{n} w_i \cdot \left(1 + e^{-x_{i,j}^{TR}}\right)^{-1}\right)}\right)^{-1} \qquad (2)$$

SLNN : gradient (2/2)

- Let us find $\partial y(x_j^{TR},w)/\partial w_i$:

$$\frac{\partial y(x_j^{TR},w)}{\partial w_i} = \frac{\partial}{\partial w_i}\left(1+e^{-\left(\sum_{i=1}^{n} w_i\cdot\left(1+e^{-x_{i,j}^{TR}}\right)^{-1}\right)}\right)^{-1} =$$

$$= \overbrace{-\left(1+e^{-\left(\sum_{i=1}^{n} w_i\cdot\left(1+e^{-x_{i,j}^{TR}}\right)^{-1}\right)}\right)^{-2}}^{-y\left(x_j^{TR},w\right)^2}\cdot\overbrace{e^{-\left(\sum_{i=1}^{n} w_i\cdot\left(1+e^{-x_{i,j}^{TR}}\right)^{-1}\right)}}^{\left(y\left(x_j^{TR},w\right)^{-1}-1\right)}$$

$$\cdot\left(-\left(1+e^{-x_{i,j}^{TR}}\right)^{-1}\right) = y(x_j^{TR},w)^2\cdot\left(y(x_j^{TR},w)^{-1}-1\right)\cdot\left(1+e^{-x_{i,j}^{TR}}\right)^{-1}$$

$$= y(x_j^{TR},w)\cdot\left(1-y(x_j^{TR},w)\right)\cdot\left(1+e^{-x_{i,j}^{TR}}\right)^{-1}$$

Therefore:

$$\boxed{\frac{\partial\tilde{L}\left(w;X^{TR},y^{TR},\lambda\right)}{\partial w_i} = \sum_{j=1}^{p} 2\cdot\left(y\left(x_j^{TR},w\right)-y_j^{TR}\right)\cdot y\left(x_j^{TR},w\right)\cdot\left(1-y\left(x_j^{TR},w\right)\right)\cdot\left(1+e^{-x_{i,j}^{TR}}\right)^{-1} + \lambda\cdot w_i}$$

SLNN: backtracking linesearch

- The backtracking linesearch algorithm `Alg.BLS` cannot handle conveniently the SLNN problem. We need to introduce two modifications in the computation of the linesearch:
  1. The maximum step length cannot be a constant for every iteration. Instead, it must be updated dynamically using information of the local behaviour of f near the iterated point at each iteration, using some of the formulas (N&W page 58):

$$\alpha_1^{max} = \alpha^{k-1} \frac{\nabla f^{k-1^T} d^{k-1}}{\nabla f^{k^T} d^k}; \quad \alpha_2^{max} = \frac{2\left(f^k - f^{k-1}\right)}{\nabla f^{k^T} d^k}.$$

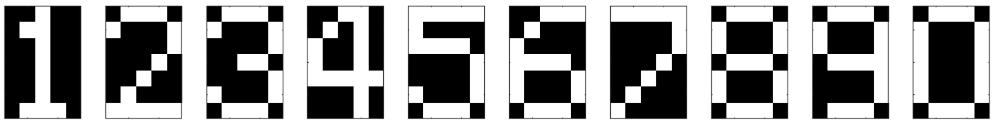
  2. A BLS based on interpolations must be used (see N&W 3.4), as the one proposed in Alg 3.2 and 3.3 of N&W, implemented in function **`uo_BLSNW32`**:

```
function [alpha,iout] =
            uo_BLSNW32(f,g,x,d,almax,c1,c2,kBLSmax,epsal)
```

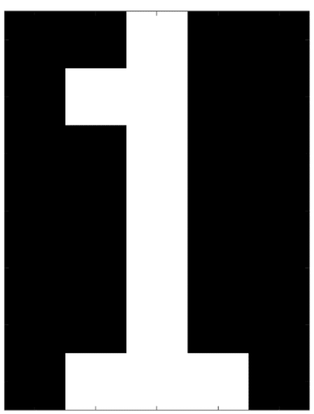
     where **`f,g,d,x,almax,c1,c2`** are as usual, **`iout=0`** if the procedure succeeds and:
     - **`kBLSmax`** is the maximum number of iterations of the BLS algorithm: if exceeded, the algorithm stops with **`iout=1`**.
     - **`epsal`** is the minimum variation between two consecutive reductions of α^k, meaning that the algorithm will stop with **`iout=2`** whenever $\left|\alpha^{k+1} - \alpha^k\right| < \textbf{epsal}$.

Pattern recognition with SLNN (1/2)

- We are going to use the SLNN to solve a problem of pattern recognition over a small 7x5 pixels matrix picturing the 10 digits:



- To obtain the input data of the SLNN x, each white pixel is assigned with a value of 10 and each black pixel with a value of -10 then vectorized and blurred with a Gaussian noise with $\mu = 0$ and $\sigma = \sigma_{rel} \cdot 10$.



-10	-10	10	-10	-10
-10	10	10	-10	-10
-10	-10	10	-10	-10
-10	-10	10	-10	-10
-10	-10	10	-10	-10
-10	-10	10	-10	-10
-10	10	10	10	-10

Vectorization

$x =$

-10
-10
10
-10
-10
-10
10
10
-10
-10
-10
-10
10
-10
-10
-10
-10
10
-10
-10
-10
-10
10
-10
-10
-10
-10
10
-10
-10
-10
10
10
10
-10

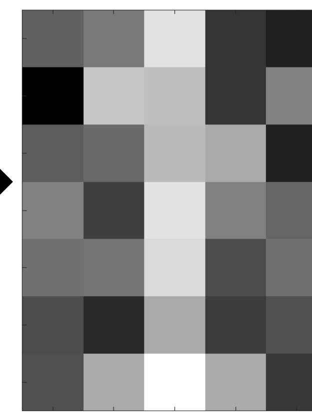
Gaussian blur

$x \leftarrow x + \epsilon =$

$\epsilon \sim N(0,5)$

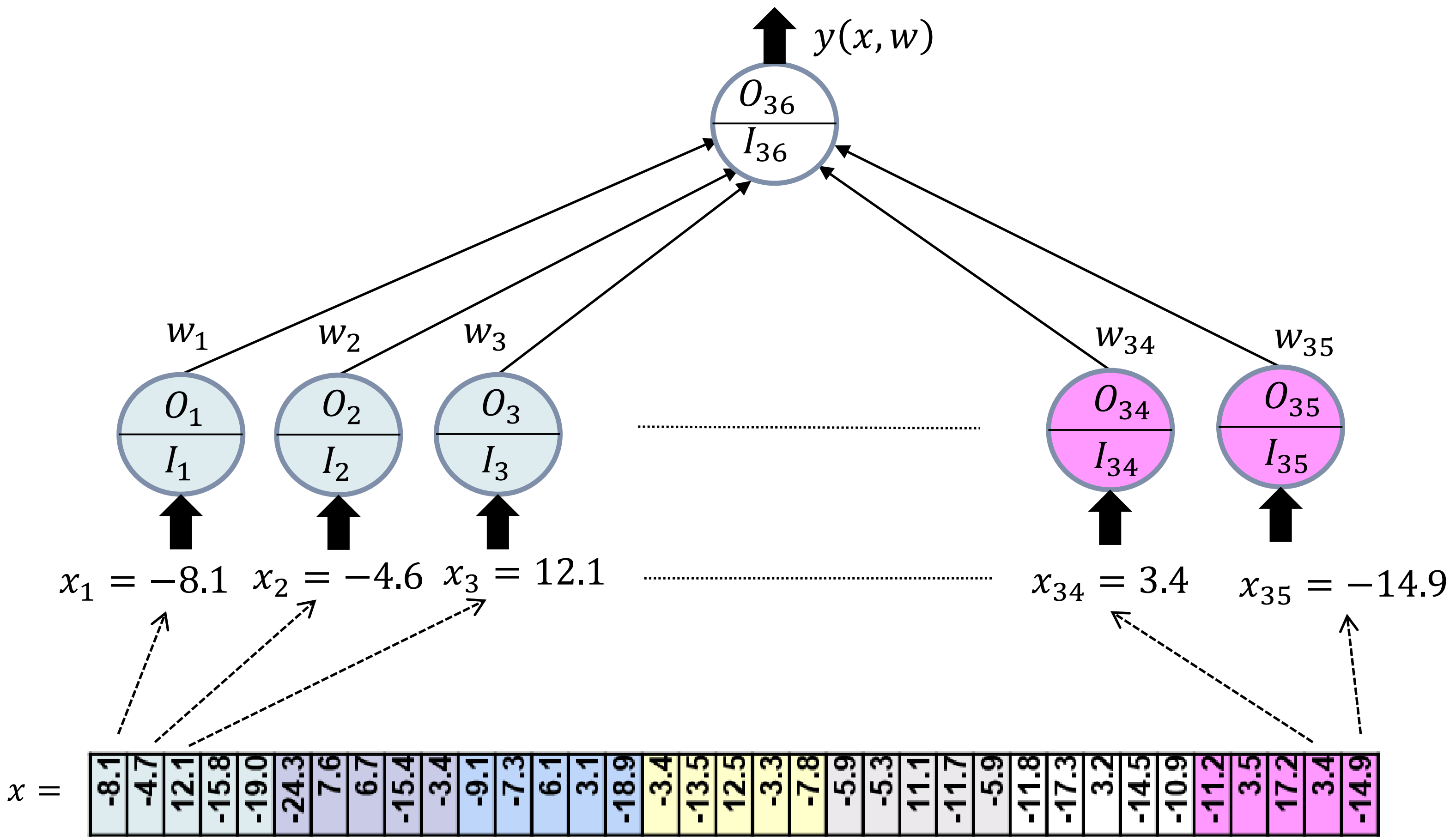
$\sigma_{rel} = 0.5$

-8.1
-4.7
12.1
-15.8
-19.0
-24.3
7.6
6.7
-15.4
-3.4
-9.1
-7.3
6.1
3.1
-18.9
-3.4
-13.5
12.5
-3.3
-7.8
-5.9
-5.3
11.1
-11.7
-5.9
-11.8
-17.3
3.2
-14.5
-10.9
-11.2
3.5
17.2
3.4
-14.9



Pattern recognition with SLNN (2/2)

- The resulting vectorised and blurred digit x are going to be taken as the inputs of a SLNN:



Training and test data set (1/2)

- The objective of the SLNN is to recognize a set of target numbers, **`num_target`**, for instance **`num_target = [ 1 3 5 7 9]`** will recognize the odd numbers between 0 and 9.
- To this end, the training and test data sets:

$$X^{TR} = \left[x_1^{TR}, x_2^{TR}, \dots, x_p^{TR}\right] \equiv \texttt{Xtr(1:35,1:tr\_p)} \text{ and } y^{TR} \equiv \texttt{ytr(1:tr\_p)}$$

$$X^{TE} = \left[x_1^{TE}, x_2^{TE}, \dots, x_p^{TE}\right] \equiv \texttt{Xte(1:35,1:te\_q)} \text{ and } y^{TE} \equiv \texttt{yte(1:te\_q)}$$

must be generated with the help of function

```
function [X,y] = uo_nn_dataset(seed, size, target, freq)
```

This function will generate a dataset, where:

- **`X,y`** are the generated data sets (**`Xtr, ytr`** or **`Xte, yte`**).
- **`seed`** is the seed for the Matlab random numbers generator. The numbers in the dataset are randomly choosed, guaranteeing a frequency of the digits in **`target`** close to **`freq`**. The value σ_{rel} for each digit is also randomly selected within the range $[0.25, 1]$.
- **`size`** is the size of the data set (number of columns/elements of array **`X/y`**).
- **`target`** is the set of digits to be identified.
- **`freq`** is the frequency of the digits **`target`** in the data ser. For instance, if **`target=[1 2]`** and **`freq=0.5`**, the digits 1 and 2 will be, approximately, half of the total digits in the data set **`X`**.

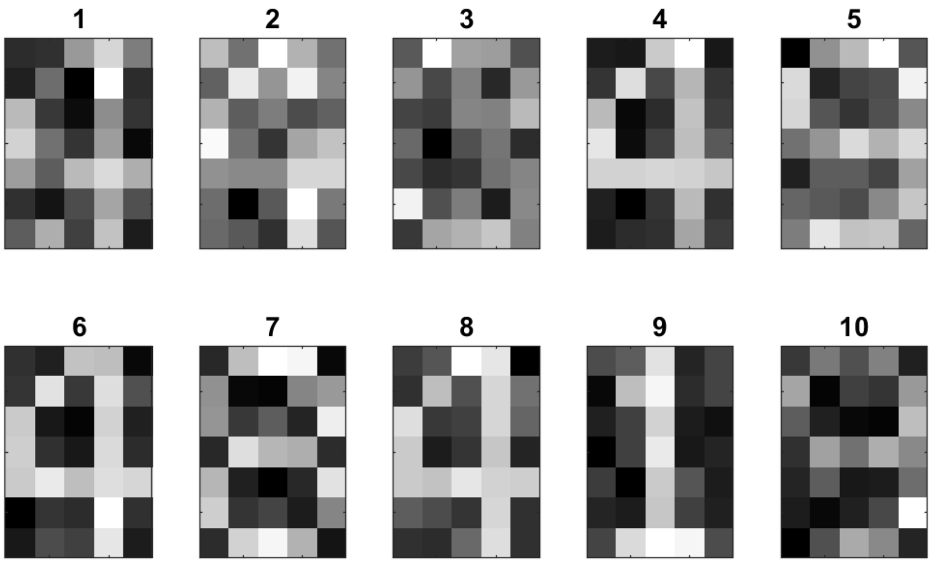
Training and test data set (2/2)

- For instance, let **seed=1234, ncol=10, target=[4], freq=0.5** then:

```
>> [X,y]=uo_nn_dataset(1234,10,[4],0.5)
X =
  -11.5323    8.5784   -8.2363  -11.4127  -31.3497   -9.9915  -10.0134   -9.8603   -5.8843   -5.6767
  -11.0925   -6.6966   46.1249  -11.9861    2.8732  -12.0420    8.8078   -6.9858   -4.5738    6.4063
    3.5013   21.9754   15.0901    9.6655   11.6102    7.1156   17.6479   14.5913    9.2044   -1.3036
..................................................................................................
 -13.9129  -12.5358    5.2724   -7.4084   -7.6072  -12.5149  -12.5704  -11.6329   -6.2185   -9.5339
y =
     1     1     0     1     0     1     0     1     0     0
```

and the graphical representation:

```
>> uo_nn_Xyplot(X,y,[])
```



- **function uo_nn_Xyplot(X,y,w)**

  plots a set of vectorised digits, and the recognition brought by a vector w:
  - **X** an array of vectorised digits.
  - **y** associated output of the SLNN.
  - **w** vector of weights w (optional).

Loss function and its gradient (1/2)

- Let **Xtr,ytr** be the training data set:
  [Xtr,ytr] = uo_nn_dataset(tr_seed, tr_p, num_target, tr_freq, noise_freq);
- If we define the row vector of residuals $y(X^{TR},w)$ and the sigmoid matrix of inputs $\sigma(X^{TR})$ as

$$y(X^{TR},w) \stackrel{\text{def}}{=} \left[y\left(x_1^{TR},w\right),\dots,y\left(x_p^{TR},w\right)\right];\ \sigma(X^{TR}) = \begin{bmatrix} \sigma\left(x_{11}^{TR}\right) & \cdots & \sigma\left(x_{1p}^{TR}\right) \\ \vdots & \ddots & \vdots \\ \sigma\left(x_{n1}^{TR}\right) & \cdots & \sigma\left(x_{np}^{TR}\right) \end{bmatrix}$$

then, the value of the loss function $\tilde{L}$ and its gradient $\nabla\tilde{L}$ can be expressed as

$$\tilde{L}(w;X^{TR},y^{TR},\lambda) = \|y(X^{TR},w) - y^{TR}\|^2 + \lambda\frac{\|w\|^2}{2}$$

$$\nabla\tilde{L}(w;X^{TR},y^{TR},\lambda) = 2\sigma(x^{TR})\left(\left(y(X^{TR},w) - y^{TR}\right)\circ y(X^{TR},w)\circ\left(1 - y(X^{TR},w)\right)\right)^T + \lambda w$$

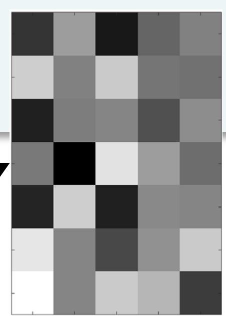
where $\circ$ stands for the **element-wise (o *Hadamard*) product**. These expressions can be easily coded in Matlab, taking profit of the **element-wise operators "./ " and ".* "**:

$\sigma(X)$	`sig = @(X) 1./(1+exp(-X));`
$y(X,w)$	`y   = @(X,w) sig(w'*sig(X));`
$\tilde{L}$	`L   = @(w) norm(y(Xtr,w)-ytr)^2 + (la*norm(w)^2)/2;`
$\nabla\tilde{L}$	`gL  = @(w) 2*sig(Xtr)*((y(Xtr,w)-ytr).*y(Xtr,w).*(1-y(Xtr,w)))'+la*w;`

Example 1: `num_target=[3]`

```
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Pattern recognition with neural networks.
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Training data set generation.
[uo_nn]    num_target = 3
[uo_nn]    tr_freq    = 0.50
[uo_nn]    tr_p       = 250
[uo_nn]    tr_seed    = 123456
[uo_nn] Optimization
[uo_nn]    L2 reg. lambda = 0.00
[uo_nn]    epsG= 1.0e-06, kmax= 20000
[uo_nn]    ialmax= 2, kmaxBLS= 30, epsBLS= 1.0e-03,
[uo_nn]    c1= 0.01, c2= 0.45, isd= 1
[uo_nn]    k         al  iW        g'*d          f        ||g||
[uo_nn]    1   1.25e-01   0  -2.45e+03   6.25e+01   4.95e+01
[uo_nn]    2   9.27e-03   2  -1.76e+03   5.43e+01   4.19e+01
[uo_nn]    3   2.39e-02   2  -1.25e+03   3.94e+01   3.54e+01
              ..............................................
[uo_nn] 3731   1.98e+04   0  -1.01e-12   2.94e-06   1.00e-06
[uo_nn] 3732   1.10e+04   0  -1.81e-12   2.93e-06   1.34e-06
[uo_nn] 3733                             2.92e-06   9.96e-07
[uo_nn]    k         al  iW        g'*d          f        ||g||
[uo_nn]    wo=[
[uo_nn]         -1.3e+01,+2.4e+00,-1.7e+01,-5.8e+00,-1.7e+00
[uo_nn]         +9.6e+00,-1.8e+00,+8.7e+00,-3.8e+00,-4.4e+00
[uo_nn]         -1.6e+01,-2.6e+00,-1.5e+00,-9.0e+00,+1.2e-01
[uo_nn]         -3.3e+00,-2.1e+01,+1.2e+01,+2.5e+00,-5.1e+00
[uo_nn]         -1.6e+01,+9.1e+00,-1.6e+01,-7.1e-01,-1.3e+00
[uo_nn]         +1.3e+01,-1.3e+00,-1.0e+01,+7.7e-01,+8.6e+00
[uo_nn]         +1.7e+01,-1.1e+00,+9.0e+00,+5.8e+00,-1.2e+01
[uo_nn]       ]
[uo_nn] Test data set generation.
[uo_nn]    te_q      = 250
[uo_nn]    te_seed   = 789101
[uo_nn] tr_accuracy = 100.0
[uo_nn] te_accuracy = 95.6
```

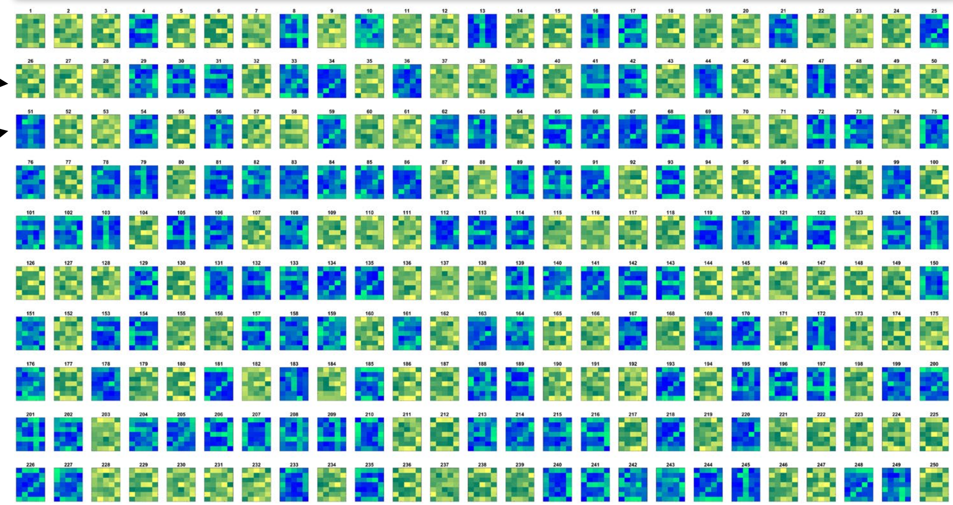
```
>> uo_nn_Xyplot(wo,0,[])
```



```
>> uo_nn_Xyplot(Xtr,ytr,wo)
```

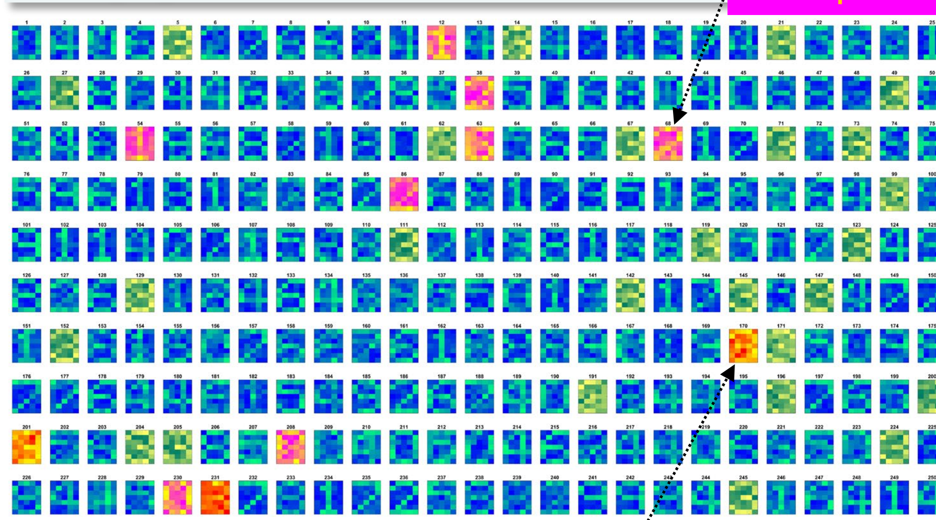
Rigth positive

Rigth negative



```
>> uo_nn_Xyplot(Xte,yte,wo)
```

False positive

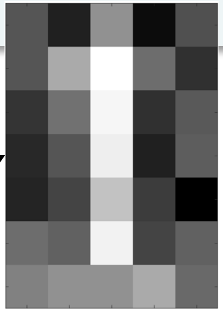


False negative

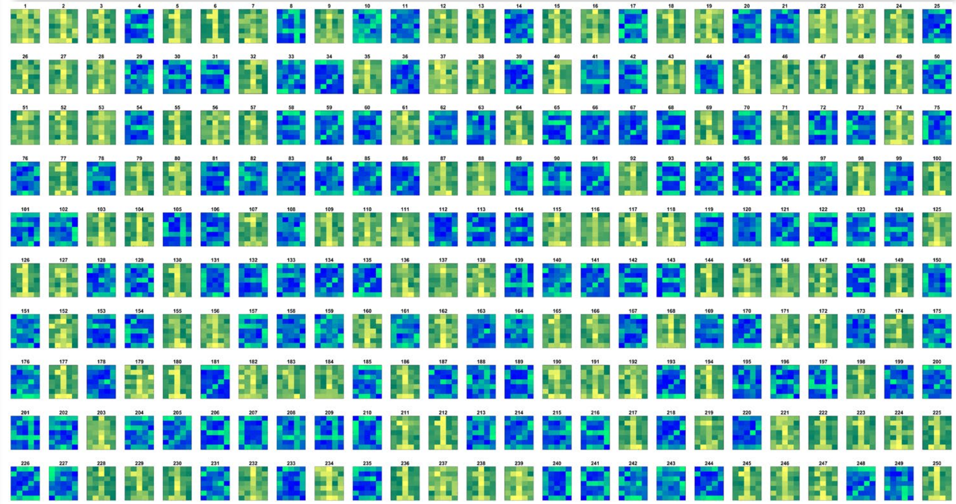
Example 2: num_target=[2]

```
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Pattern recognition with neural networks.
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Training data set generation.
[uo_nn]    num_target = 1
[uo_nn]    tr_freq    = 0.50
[uo_nn]    tr_p       = 250
[uo_nn]    tr_seed    = 123456
[uo_nn] Optimization
[uo_nn]    L2 reg. lambda = 0.00
[uo_nn]    epsG= 1.0e-06, kmax= 20000
[uo_nn]    ialmax= 2, kmaxBLS= 30, epsBLS= 1.0e-03,
[uo_nn]    c1= 0.01, c2= 0.45, isd= 1
[uo_nn]    k        al  iW       g'*d          f       ||g||
[uo_nn]    1   5.00e-01   0  -7.38e+03   6.25e+01   8.59e+01
[uo_nn]    2   8.19e+03   0  -2.90e-07   1.00e+00   5.39e-04
[uo_nn]    3   2.53e+04   0  -6.18e-07   2.16e-04   7.86e-04
[uo_nn]    4                             3.51e-12   2.67e-11
[uo_nn]    k        al  iW       g'*d          f       ||g||
[uo_nn]    wo=[
[uo_nn]         -5.0e+00,-1.1e+01,+2.1e+00,-1.3e+01,-5.2e+00
[uo_nn]         -4.7e+00,+5.1e+00,+1.5e+01,-1.8e+00,-9.0e+00
[uo_nn]         -8.7e+00,-1.8e+00,+1.4e+01,-9.0e+00,-4.4e+00
[uo_nn]         -1.0e+01,-4.9e+00,+1.3e+01,-1.1e+01,-3.8e+00
[uo_nn]         -1.0e+01,-6.7e+00,+8.0e+00,-7.9e+00,-1.5e+01
[uo_nn]         -2.1e+00,-3.6e+00,+1.4e+01,-6.8e+00,-3.4e+00
[uo_nn]         +2.7e-01,+2.2e+00,+2.3e+00,+5.3e+00,-2.4e+00
[uo_nn]       ]
[uo_nn] Test data set generation.
[uo_nn]    te_q      = 250
[uo_nn]    te_seed   = 789101
[uo_nn] tr_accuracy = 100.0
[uo_nn] te_accuracy = 100.0
```

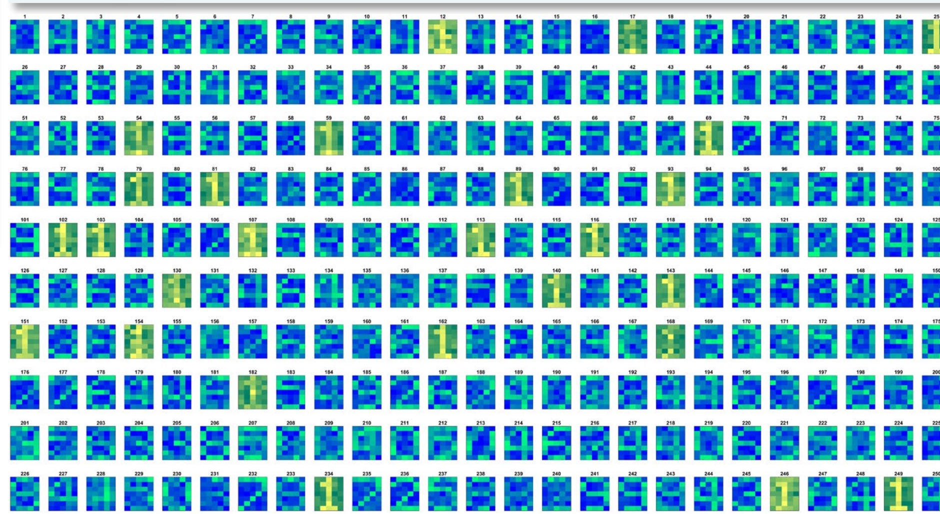
```
>> uo_nn_Xyplot(wo,0,[])
```



```
>> uo_nn_Xyplot(Xtr,ytr,wo)
```



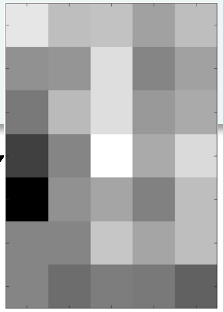
```
>> uo_nn_Xyplot(Xte,yte,wo)
```



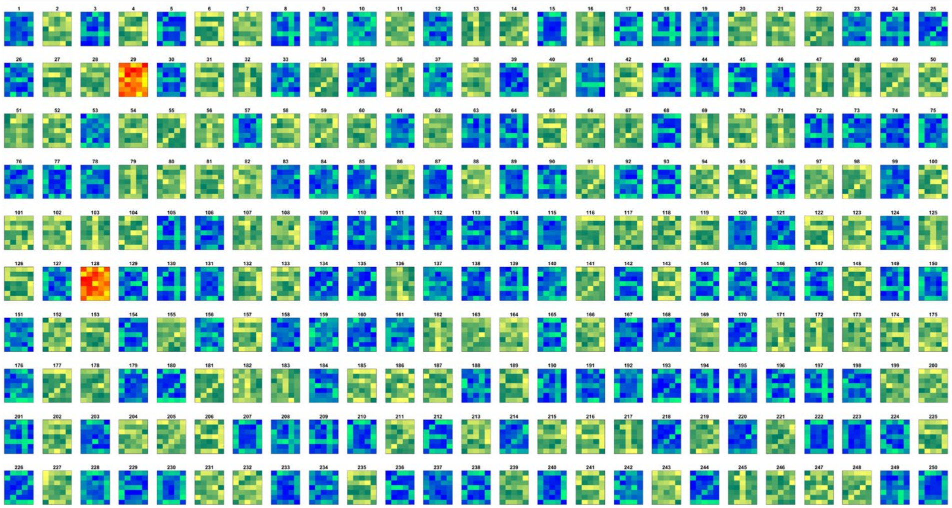
Example 3: num_target=[1 3 5 7 9]

```
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Pattern recognition with neural networks.
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Training data set generation.
[uo_nn]    num_target = 1 3 5 7 9
[uo_nn]    tr_freq    = 0.50
[uo_nn]    tr_p       = 250
[uo_nn]    tr_seed    = 123456
[uo_nn] Optimization
[uo_nn]    L2 reg. lambda = 1.00
[uo_nn]    epsG= 1.0e-06, kmax= 20000
[uo_nn]    ialmax= 2, kmaxBLS= 30, epsBLS= 1.0e-03,
[uo_nn]    c1= 0.01, c2= 0.45, isd= 3
[uo_nn]    k        al  iW       g'*d          f      ||g||
[uo_nn]    1   3.13e-02  0  -1.89e+03   6.25e+01   4.35e+01
[uo_nn]    2   5.25e-03  2  -4.31e+03   5.12e+01   8.16e+01
[uo_nn]    3   4.57e-02  0  -4.10e+02   3.25e+01   5.41e+01
          ...............................................
[uo_nn]   27   1.11e+00  0  -6.80e-12   1.43e+01   4.99e-06
[uo_nn]   28   8.52e-01  0  -1.35e-12   1.43e+01   1.64e-06
[uo_nn]   29                            1.43e+01   4.32e-07
[uo_nn]    k        al  iW       g'*d          f      ||g||
[uo_nn]    wo=[
[uo_nn]         +9.1e-01,+4.3e-01,+4.5e-01,+3.4e-02,+3.9e-01
[uo_nn]         -1.2e-01,-9.8e-02,+8.1e-01,-3.1e-01,+4.1e-02
[uo_nn]         -4.4e-01,+3.6e-01,+7.8e-01,-6.3e-02,+1.8e-01
[uo_nn]         -1.1e+00,-3.0e-01,+1.2e+00,+1.5e-01,+7.5e-01
[uo_nn]         -1.9e+00,-1.4e-01,+9.1e-02,-3.5e-01,+4.0e-01
[uo_nn]         -3.0e-01,-3.0e-01,+5.2e-01,+1.2e-01,+3.8e-01
[uo_nn]         -3.0e-01,-5.7e-01,-3.8e-01,-4.2e-01,-7.4e-01
[uo_nn]        ]
[uo_nn] Test data set generation.
[uo_nn]    te_q      = 250
[uo_nn]    te_seed   = 789101
[uo_nn] tr_accuracy = 99.2
[uo_nn] te_accuracy = 97.2
```

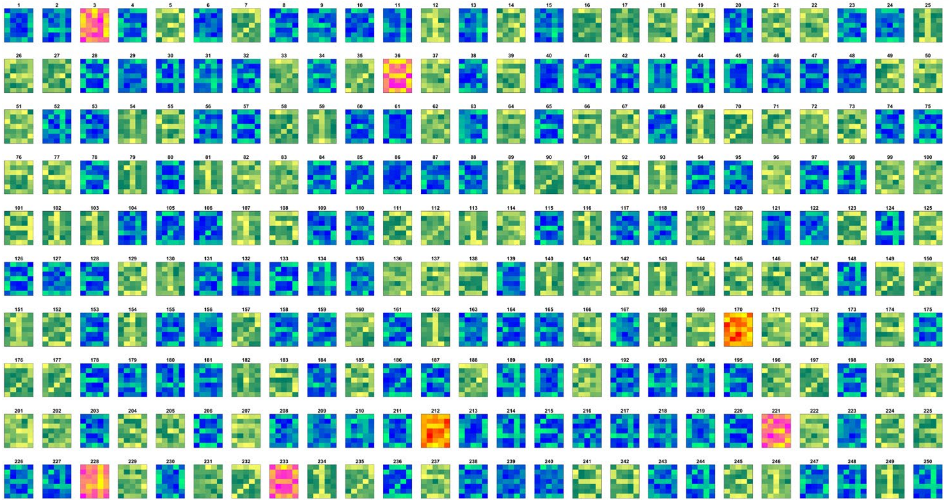
```
>> uo_nn_Xyplot(wo,0,[])
```



```
>> uo_nn_Xyplot(Xtr,ytr,wo)
```



```
>> uo_nn_Xyplot(Xte,yte,wo)
```

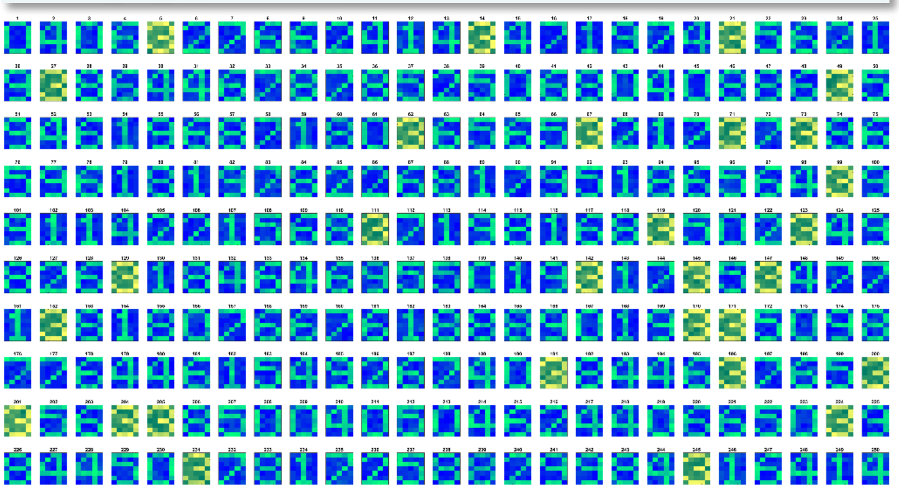


UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Estadística
i Investigació Operativa

The effect of "blurring"

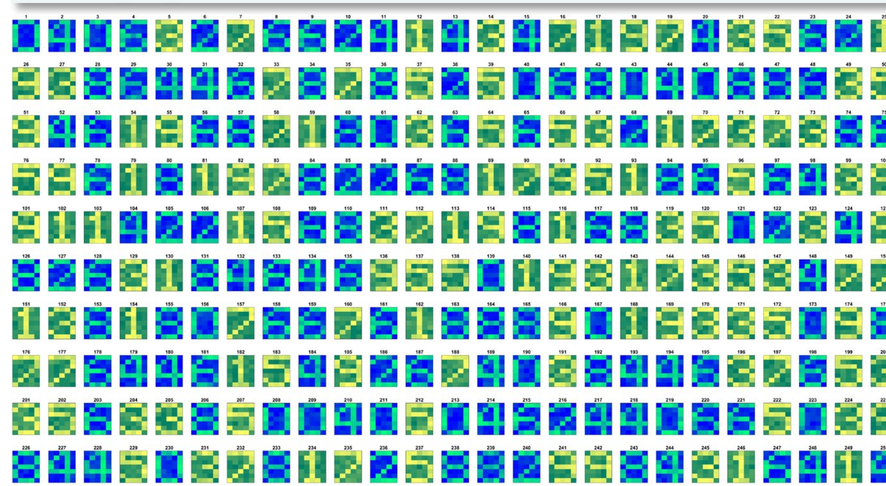
- It must be stressed that the "bad" results for some of the previous examples are a consequence of the heavy blurring applied to the images, with a σ_{rel} up to a 100% of the value of the pixel. Should the blurring be eliminated or reduced, the classification will be exact. For instance, for $\sigma_{rel} = 0.25$ the identification will be exact:

```
>> uo_nn_Xyplot(Xte,yte,wo)
```



`num_target=[3]`

```
>> uo_nn_Xyplot(Xte,yte,wo)
```



`num_target=[1 3 5 7 9]`

Assignment (1/4)

- In this assignment we want to conduct a series of computational experiments to study the dependency of the performance of the SLNN on several parameters. An instance of the SLNN problem must be solved:
  - For every one of the individual digits from 0 to 9.
  - For every value of the regularization parameter $\lambda \in \{0.0, 1.0, 10.0\}$.
  - For every one of the following optimization algorithms: GM, CGM-PR+ (IRC=2,$\nu = 1$) and QM-BFGS.

  That makes a total of $3 \times 3 \times 10 = 90$ instances. Every instance must be run with the following settings:
  - The optimization parameters **`epsG=10^-06, kmax=5000, almax=`**α_2^{max}**`, c1=0.01, c2=0.45, kBLSmax=30`** and **`epsal=10^-03.`**
  - The parameter to generate the training and testing data sets: **`tr_p=tr_q=250`** , **`tr_freq=0.5`**, **`te_freq=0.0`**. The seeds must be set to **`tr_seed=NIF1`** and **`te_seed=NIF2`** where **`NIF1`** and **`NIF2`** is the NIF number of the two members of the team.

Assignment (2/4)

- To organize the computational experiments you can use the script **`uo_nn_batch.m`**:

`uo_nn_batch.m` : run a batch of SLNN instances.

```
clear;
%
tr_freq = .5; tr_seed = 123456; tr_p = 250; te_seed = 789101; te_q = tr_p;
% Parameters for optimization:
epsG = 10^-6; kmax = 5000;                                  % Stopping criterium:
ils=1; ialmax = 2; kmaxBLS=30; epsal=10^-3; c1=0.01; c2=0.45; % Linesearch:
icg = 2; irc = 2 ; nu = 1.0;                                % Search direction:
%
iheader = 1; fileID = fopen('om_uo_nn_batch.csv','w');
for num_target = 1:10
    for la = [0.0, 1.0, 10.0]
        for isd = 1:3
            [Xtr,ytr,wo,tr_acc,Xte,yte,te_acc,niter,tex]=uo_nn_solve(num_target,
tr_freq,tr_seed,tr_p,te_seed,te_q,la,epsG,kmax,ils,ialmax,kmaxBLS,epsal,c1,c2,isd,icg,irc,nu,iheader);
            if iheader == 1 fprintf(fileID,'num_target;   la; isd; niter;     tex; te_acc;\n'); end
            fprintf(fileID,'         %1i; %4.1f;   %1i;  %4i; %7.4f;  %5.1f;\n', mod(num_target,10), la,
isd, niter, tex, te_acc);
            iheader=0;
        end
    end
end
fclose(fileID);
```

Assignment (3/4)

- Function **`uo_nn_solve`** solves the instance corresponding to a particular combination of parameters. The outcome of this code is the file **`uo_nn_batch.csv`** with the following content:

```
num_target;   la; isd; niter;     tex; te_acc;
         1;  0.0;   1;     4;  0.0074;  100.0;
         1;  0.0;   2;     5;  0.0138;  100.0;
.............................................
         0; 10.0;   1;    96;  0.1571;   99.2;
         0; 10.0;   2;    53;  0.1604;   99.2;
         0; 10.0;   3;    42;  0.1058;   99.2;
```

- Function **`uo_nn_solve`** called imside **`uo_nn_batch.m`** must solve the instance corresponding to a particular combination of parameters.

- The actions to be taken inside this function are:

  i. To generate the training data set (X^{TR}, y^{TR}).

  ii. To find the value of w^* minimizing $\tilde{L}(w; X^{TR}, y^{TR}, \lambda)$ with your own optimization routines developed during the course.

  iii. To calculate Accuracy^{TR}.

  iv. To generate the test dataset (X^{TE}, y^{TE}) and to calculate Accuracy^{TE}.

Assignment (4/4)

- The goal of this assignment is to fulfill the following tasks:
  a) Based on the data in file **uo_nn_batch.csv**, you have to determine:
     1) First, which is the value of the regularization parameter λ that gives the best results for the overall set of digits and optimization methods.
     2) Second, for the value of λ determined in the previous section, find out which is the best optimization algorithm, GM, CGM-PR+ or QM-BFGS, based on the analysis of the variables $Accuracy^{TE}$, **niter** and **tex**. Describe how the execution-time per iteration (**tex/niter**) behaves for the three different methods and find an explanation.
  b) Describe how the accuracy of the pattern recognition ($Accuracy^{TE}$) depends on the digit for the value of λ and optimization method determined in section a). For the digit with the worst value of $Accuracy^{TE}$, display the results with the help of function **uo_nn_Xyplot** and try to guess the reasons for the bad recognition rate.
  c) Finally, make use of the trained SLNN to develop a function that can identify series of 5 digits. This function must get an array **X** with 5 digits randomly generated with function **uo_nn_dataset** and return the list with the 5 digits identified by the SLNN (see the example in slide #3 "Presentation"). Check the function with 10 different sets of 5 digits and analyze the results.
- This assignment must be done in groups of two. Use a value of **tr_seed** and **te_seed** based on your NIF. You must upload to Atenea a file with the name **surname-student-1_surname-student-2.zip** containing:
  – A report (.pdf file) with your results and comments of tasks a) to c).
  – The source of all the codes used to do tasks a) to c).

Summary of supporting codes

Function/script	Page
`function [alpha,iout] = uo_BLSNW32(f,g,x,d,almax,c1,c2,kBLSmax,epsal)`: Algorithm 3.2 of Nocedal & Wright (backtracking line search with SWC and curve fitting).	9
`function [X,y] = uo_nn_dataset(seed, ncol, target, freq)`: generates the dataset `X,y`.	12
`function uo_nn_Xyplot(X,y,w)`: plots the dataset `X,y`. If `w` is not `[]`, the plot tells right from wrong predictions of the SLNN.	13

`sig = @(X) 1./(1+exp(-X));`
`y   = @(X,w) sig(w'*sig(X));`
`L   = @(w) norm(y(Xtr,w)-ytr)^2 + (la*norm(w)^2)/2;`
`gL  = @(w) 2*sig(Xtr)*((y(Xtr,w)-ytr).*y(Xtr,w).*(1-y(Xtr,w)))'+la*w;`
	14

`uo_nn_batch.m`: run a batch of SLNN instances.	20