Part 2: Stochastic Gradient (1/3)

- **Stochastic Gradient Method (SGM).**
  - **Principle:** it is possible to obtain an unbiased estimate of the gradient by taking the average gradient on a minibatch of m examples drawn i.i.d. from the training dataset.
  - **Procedure:** depends on parameters $\boldsymbol{\alpha_0^{SG}} > 0$ and $\boldsymbol{\gamma_1^{SG}}, \boldsymbol{\gamma_2^{SG}} \in [0,1]$

    i. Sample a minibatch $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_m}\}$ of $\boldsymbol{m} = \lfloor \boldsymbol{\gamma_1^{SG}} p \rfloor \ll p$ observations from the training dataset:

$$X_{\boldsymbol{S}}^{TR} = [x_{\boldsymbol{s_1}}^{TR}, x_{\boldsymbol{s_2}}^{TR}, \dots, x_{\boldsymbol{s_m}}^{TR}];\ y_{\boldsymbol{S}}^{TR} = [y_{\boldsymbol{s_1}}^{TR} \quad y_{\boldsymbol{s_2}}^{TR} \quad \cdots \quad y_{\boldsymbol{s_m}}^{TR}]^T$$

    ii. Compute search direction through the gradient estimate:

$$d^k \leftarrow -\frac{1}{\boldsymbol{m}} \nabla \tilde{L}(w; X_{\boldsymbol{S}}^{TR}, y_{\boldsymbol{S}}^{TR}, \lambda)$$

    iii. Update the parametres: $w^k \leftarrow w^k + \alpha^k d^k$ with ***learning rate*** $\boldsymbol{\alpha^k}$

$$\alpha^k := \begin{cases} \left(1 - \frac{k}{k^{SG}}\right) \boldsymbol{\alpha_0^{SG}} + \frac{k}{k^{SG}} \alpha^{SG} & \text{if } k \le k^{SG} \\ \alpha^{SG} & \text{if } k > k^{SG} \end{cases}, \qquad \begin{cases} \alpha^{SG} \approx 0{,}01 \cdot \boldsymbol{\alpha_0^{SG}} \\ k^{SG} := \lfloor \boldsymbol{\gamma_2^{SG}} \cdot k^{max} \rfloor \end{cases}$$

Part 2: Stochastic Gradient (2/3)

- Script `uo_nn_solve.m` must be extended to include the SGM:

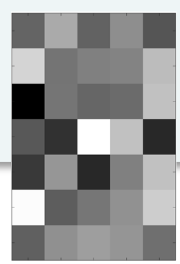
uo_nn_main.m : recognition of num_target digits.

```
clear;
%
% Parameters for dataset generation
%
num_target =[3];
tr_freq    = .5;
tr_p       = 250;
te_q       = 250;
tr_seed    = 123456;
te_seed    = 789101;
%
% Parameters for optimization
%
la = 1.0;                                                    % L2 regularization.
epsG = 10^-6; kmax = 10000;                                  % Stopping criterium.
ils=3; ialmax = 2; kmaxBLS=30; epsal=10^-3;c1=0.01; c2=0.45; % Linesearch (ils=3 -> uo_BLSNW32).
isd = 7; icg = 2; irc = 2 ; nu = 1.0;                        % Search direction.
sg_ga1 = 0.05; sg_al0=2; sg_ga2=0.3;                         % Stochastic gradient
%
% Optimization
%
t1=clock;
[Xtr,ytr,wo,fo,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,sg_ga1,sg_al0,sg_ga2,icg,irc,nu)
;
t2=clock;
fprintf(' wall time = %6.1d s.\n', etime(t2,t1));
%
```

Part 2: Stochastic Gradient (3/3)

```
[uo_nn] ::::::::::::::::::::::::::::::::::::::::::::::::::
[uo_nn] Pattern recognition with neural networks (OM/GCED).
[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 = 1.00
[uo_nn]    epsG= 1.0e-06, kmax= 10000
[uo_nn]    ils= 3, ialmax= 2, kmaxBLS= 30, epsBLS= 1.0e-03,
[uo_nn]    c1= 0.01, c2= 0.45, isd= 7
[uo_nn]    sg_ga1= 0.05, sg_al0= 2.0, sg_ga2= 0.30
[uo_nn]    k        al  iW       g'*d          f       ||g||
[uo_nn]    1   2.00e+00  0  -1.46e+02   6.25e+01   4.95e+01
[uo_nn]    2   2.00e+00  0  -7.03e+01   1.63e+02   8.46e+00
[uo_nn]    3   2.00e+00  0  -6.90e+01   1.56e+02   8.31e+00
                 ..............................................
[uo_nn] 9998   2.00e-02  0  -4.71e+01   2.68e+01   2.69e+01
[uo_nn] 9999   2.00e-02  0  +1.65e+00   2.64e+01   1.94e+01
[uo_nn]10000                            2.68e+01   2.49e+01
[uo_nn]    k        al  iW       g'*d          f       ||g||
[uo_nn]    wo=[
[uo_nn]        -1.5e-01,+6.0e-02,-1.6e-01,-2.6e-02,-2.0e-01
[uo_nn]        +1.9e-01,-1.0e-01,-6.7e-02,-5.8e-02,+1.1e-01
[uo_nn]        -4.6e-01,-1.0e-01,-1.5e-01,-1.3e-01,+1.3e-01
[uo_nn]        -1.9e-01,-3.1e-01,+3.2e-01,+1.3e-01,-3.4e-01
[uo_nn]        -2.8e-01,-1.3e-03,-3.3e-01,-6.7e-02,+1.3e-01
[uo_nn]        +3.1e-01,-1.7e-01,-9.9e-02,-1.5e-02,+1.7e-01
[uo_nn]        -1.7e-01,-1.7e-02,+2.4e-02,-4.0e-03,-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 = 96.4
[uo_nn] te_accuracy = 87.2
```

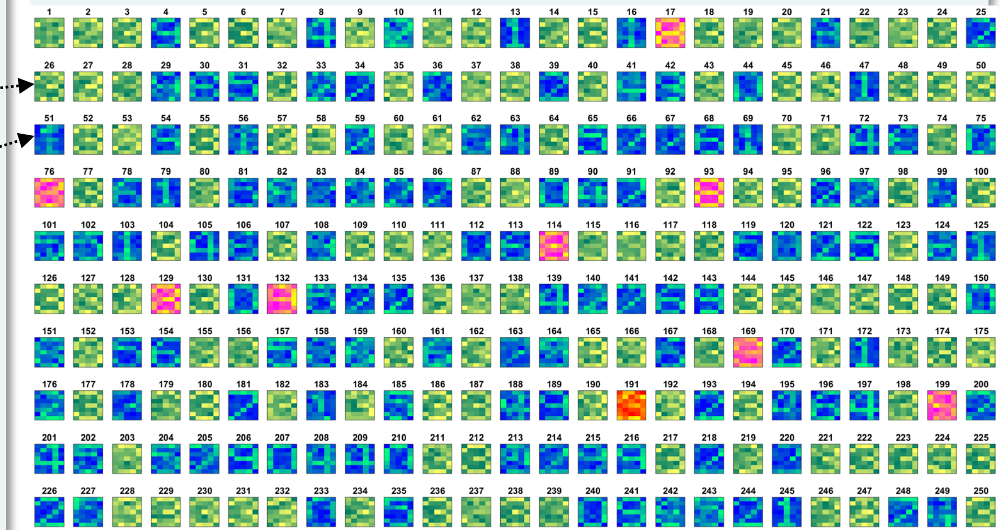
```
>> uo_nn_Xyplot(wo,0,[])
```



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

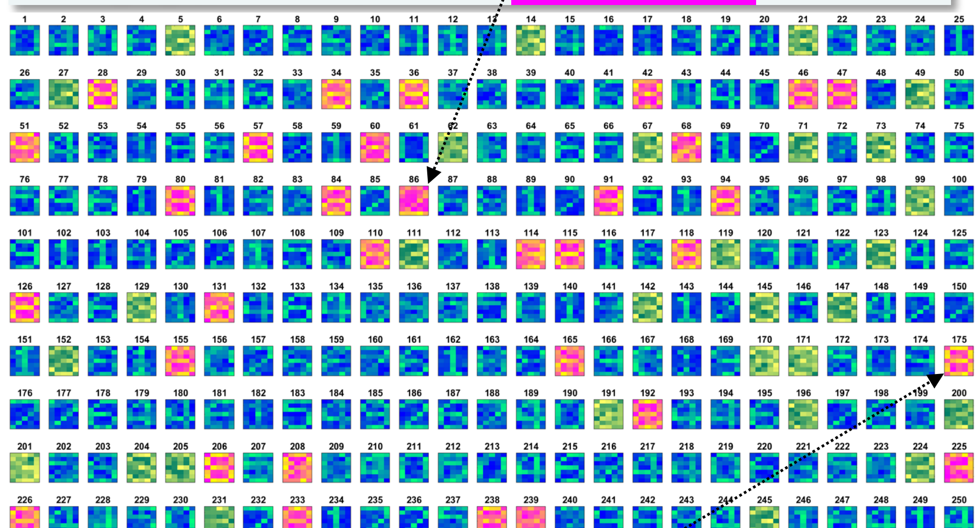
Rigth positive

Rigth negative



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

False positive



False negative

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

Part 3: computational study (1/3)

- In this third part we want to conduct a series of computational experiments to study:
  i. How the regularization parameter λ affects the results.
  ii. The relative performance of the different algorithms (GM, QNM,SGM)
- To this end, an instance of the SLNN problem must be solved:
  – For every one of the individual digits, 0 to 9.
  – For every value of the regularization parameter $\lambda \in \{0.0, 1.0, 10.0\}$.
  – For every optimization algorithm: GM, QNM and SGM.

  That makes a total of $10 \times 3 \times 4 = 120$ instances to be solved.

Part 3: computational study (2/3)

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

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

```
function uo_nn_batch(tr_seed,te_seed)
% Parameters.
tr_p = 250; te_q = 250; tr_freq = .5;                          % Datasets generation
epsG = 10^-6; kmax = 1000;                                     % Stopping criterium.
ils=3; ialmax = 2; kmaxBLS=10; epsal=10^-3; c1=0.01; c2=0.45; % Linesearch (ils=3 -> uo_BLSNW32).
icg = 2; irc = 2 ; nu = 1.0;                                   % Search direction.
sg_ga1 = 0.05; sg_al0=2; sg_ga2=0.3;                           % Stochastic gradient
% Optimization
iheader = 1;
csvfile = strcat('uo_nn_batch_',num2str(tr_seed),'-',num2str(te_seed),'.csv');
fileID = fopen(csvfile ,'w');
t1=clock;
for num_target = 1:10
    for la = [0.0, 1.0, 10.0]
        for isd = [1,3,7]
            [Xtr,ytr,wo,fo,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,sd,sg_ga1,sg_al0,sg_ga2,icg,i
rc,nu,iheader);
            if iheader == 1 fprintf(fileID,'num_target;   la; isd; niter;     tex; tr_acc; te_acc;  L*\n');
end
            fprintf(fileID,'         %1i; %4.1f; %1i;  %4i; %7.4f;  %5.1f;  %5.1f;  %8.2e;\n',
mod(num_target,10), la, isd, niter, tex, tr_acc, te_acc, fo);
            iheader=0;
        end
    end
end
t2=clock;
fprintf(' wall time = %6.1d s.\n', etime(t2,t1));
fclose(fileID);
%
end
```