Purpose:

How to train an MLP neural network in MATLAB environment!

that is

For good computations,
we need good formulae
for good algorithms;
and good visualization
for good illustration
and proper testing
of good methods
and succesfull applications!

Learning/Training the MLP:

1. **Learning data:** given set of input-output vector-pairs $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, $\mathbf{x}_i \in \mathbf{R}^{n_0}$ and $\mathbf{y}_i \in \mathbf{R}^{n_2}$
   - to enhance the next step prescaling into the range of the activation functions
2. **Learning problem:** optimization problem to train the network according to data:

$$\min_{(\mathbf{W}^1, \mathbf{W}^2)} \mathcal{J}(\mathbf{W}^1, \mathbf{W}^2), \tag{1}$$

   where (LMS = *least-mean squares*)

$$\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) = \frac{1}{2N} \sum_{i=1}^{N} \|\mathcal{N}(\mathbf{x}_i) - \mathbf{y}_i\|^2 = \frac{1}{2N} \sum_{i=1}^{N} \|\mathbf{W}^2 \, \widehat{\mathbf{F}}(\mathbf{W}^1 \, \hat{\mathbf{x}}_i) - \mathbf{y}_i\|^2 \tag{2}$$

   - tradional perspective: chain-rule in an index jungle!
   - our approach: layer-wise treatment according to network structure!
3. **Training method:** a way to solve the optimization problem
   - tradional perspective: *backprop, quickprop, rpprop, $*$prop etc.*
   - our approach: efficient optimization algorithm that solves problem (1)

Layerwise calculus for sensitivity analysis:

Every local solution $(\mathbf{W}^{1^*}, \mathbf{W}^{2^*})$ for minimization problem (1) characterized by the conditions

$$\nabla_{(\mathbf{W}^1,\mathbf{W}^2)} \mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) = \begin{bmatrix} \nabla_{\mathbf{W}^1} \mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) \\ \nabla_{\mathbf{W}^2} \mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) \end{bmatrix} = \begin{bmatrix} \mathbf{O} \\ \mathbf{O} \end{bmatrix}.$$

- assume that activation functions are differentiable

Lemma 1. Let $\mathbf{v} \in \mathbf{R}^{m_1}$ and $\mathbf{y} \in \mathbf{R}^{m_2}$ be given vectors. The derivative-matrix $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbf{R}^{m_2 \times m_1}$ for the functional

$$J(\mathbf{W}) = \frac{1}{2} \|\mathbf{W}\,\mathbf{v} - \mathbf{y}\|^2$$

is of the form

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = [\mathbf{W}\,\mathbf{v} - \mathbf{y}]\,\mathbf{v}^T.$$

Lemma 2. Let $\mathbf{W} \in \mathbf{R}^{m_2 \times m_1}$ be a given matrix, $\mathbf{y} \in \mathbf{R}^{m_2}$ a given vector, and $\mathbf{F} = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^{m_1}$ a given diagonal function-matrix. The gradient $\nabla_{\mathbf{u}} J(\mathbf{u}) \in \mathbf{R}^{m_1}$ for the functional

$$J(\mathbf{u}) = \frac{1}{2} \|\mathbf{W}\,\mathbf{F}(\mathbf{u}) - \mathbf{y}\|^2 \tag{3}$$

reads as

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \mathrm{Diag}\{\mathbf{F}'(\mathbf{u})\}\,\mathbf{W}^T\,[\mathbf{W}\,\mathbf{F}(\mathbf{u}) - \mathbf{y}].$$

Lemma 3. Let $\bar{\mathbf{W}} \in \mathbf{R}^{m_2 \times m_1}$ be a given matrix, $\mathbf{F} = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^{m_1}$ a given diagonal function-matrix, and $\mathbf{v} \in \mathbf{R}^{m_0}$, $\mathbf{y} \in \mathbf{R}^{m_2}$ given vectors. The derivative-matrix $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbf{R}^{m_1 \times m_0}$ for the functional

$$J(\mathbf{W}) = \frac{1}{2} \|\bar{\mathbf{W}}\,\mathbf{F}(\mathbf{W}\mathbf{v}) - \mathbf{y}\|^2 \tag{4}$$

is of the form

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \mathrm{Diag}\{\mathbf{F}'(\mathbf{W}\mathbf{v})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\mathbf{F}(\mathbf{W}\mathbf{v}) - \mathbf{y}]\,\mathbf{v}^T.$$

Layerwise optimality conditions for MLP (I):

Theorem 1. Derivative-matrices $\nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1,\, \mathbf{W}^2)$ and $\nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1,\, \mathbf{W}^2)$ for the cost functional (2) are of the form

(i)

$$\begin{aligned}\nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1,\, \mathbf{W}^2) &= \frac{1}{N}\sum_{i=1}^{N}[\mathbf{W}^2\,\widehat{\mathbf{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i]\,[\widehat{\mathbf{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)]^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i\,[\widehat{\mathbf{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)]^T,\end{aligned}$$

(ii)

$$\begin{aligned}\nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1,\, \mathbf{W}^2) &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Diag}\{\mathbf{F}'(\mathbf{W}^1\hat{\mathbf{x}}_i)\}\,(\mathbf{W}^2_1)^T\,[\mathbf{W}^2\,\widehat{\mathbf{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i]\,\hat{\mathbf{x}}_i^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Diag}\{\mathbf{F}'(\mathbf{W}^1\,\hat{\mathbf{x}}_i)\}\,(\mathbf{W}^2_1)^T\,\mathbf{e}_i\,\hat{\mathbf{x}}_i^T.\end{aligned}$$

In (ii), $\mathbf{W}^2_1$ is the submatrix obtained from $\mathbf{W}^2$ by removing the first column $\mathbf{W}^2_0$ containing the bias nodes.

In MATLAB:

```
for i=1:N
   [o,o1,d1] = mlp_out(x(:,i),w1,w2);
   e = o - y(:,i);
   f = f + e'*e/(2*N);
   o1_ext = [1; o1];
   dw1 = dw1 + diag(d1)*w2(:,2:n1+1)'*e*[1 x(:,i)']/N;
   dw2 = dw2 + e*o1_ext'/N;
end
OR
[o,o1,d1] = mlp_out2(x',w1,w2);
e = o - y';
f = sum(sum(e.^2))/(2*N);
dw1 = ( d1.*(w2(:,2:n1+1)'*e) )*[ones(N,1) x]/N;
dw2 = e*[ ones(N,1) o1' ]/N;
```

Layerwise optimality conditions for MLP (II):

For more-than-two-layers problem

$$\mathcal{J}(\{\mathbf{W}^l\}_{l=1}^L) = \frac{1}{2N}\sum_{i=1}^{N} \|\mathbf{W}^L \hat{\mathbf{o}}_i^{(L-1)} - \mathbf{y}_i\|^2, \qquad (5)$$

where $\mathbf{o}_i^0 = \mathbf{x}_i$ and $\mathbf{o}_i^l = \mathbf{F}^l(\mathbf{W}^l \hat{\mathbf{o}}_i^{(l-1)})$ for $l = 1, \ldots, L-1$

we have the general result

Theorem 2. Derivative-matrices $\nabla_{\mathbf{W}^l}\, \mathcal{J}(\{\mathbf{W}^l\}_{l=1}^L),\ l = L, \ldots, 1$, for the cost functional (5) are of the form

$$\nabla_{\mathbf{W}^l} \mathcal{J}(\{\mathbf{W}^l\}_{l=1}^L) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{d}_i^l\, [\hat{\mathbf{o}}_i^{(l-1)}]^T,$$

where

$$\mathbf{d}_i^L = \mathbf{e}_i = \mathbf{W}^L \hat{\mathbf{o}}_i^{(L-1)} - \mathbf{y}_i, \qquad (6)$$

$$\mathbf{d}_i^l = \mathrm{Diag}\{(\mathbf{F}^l)'(\mathbf{W}^l\, \hat{\mathbf{o}}_i^{(l-1)})\}\, (\mathbf{W}_1^{(l+1)})^T\, \mathbf{d}_i^{(l+1)}. \qquad (7)$$

Where's the beef?

- efficient (and correct) implementation
  - computation of $\mathbf{o}_i^l$'s in forward loop
  - overwritten by $\mathbf{d}_i^l$'s in backward loop
  - realization of (7) in single loop (for sigmoidal activation)
- possibilites for analysis opened up

What does the MLP actually learn?

Corollary 1. (i) The average error $\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*$ made by the locally optimal MLP-network satisfying the conditions in Theorem 2 is zero.

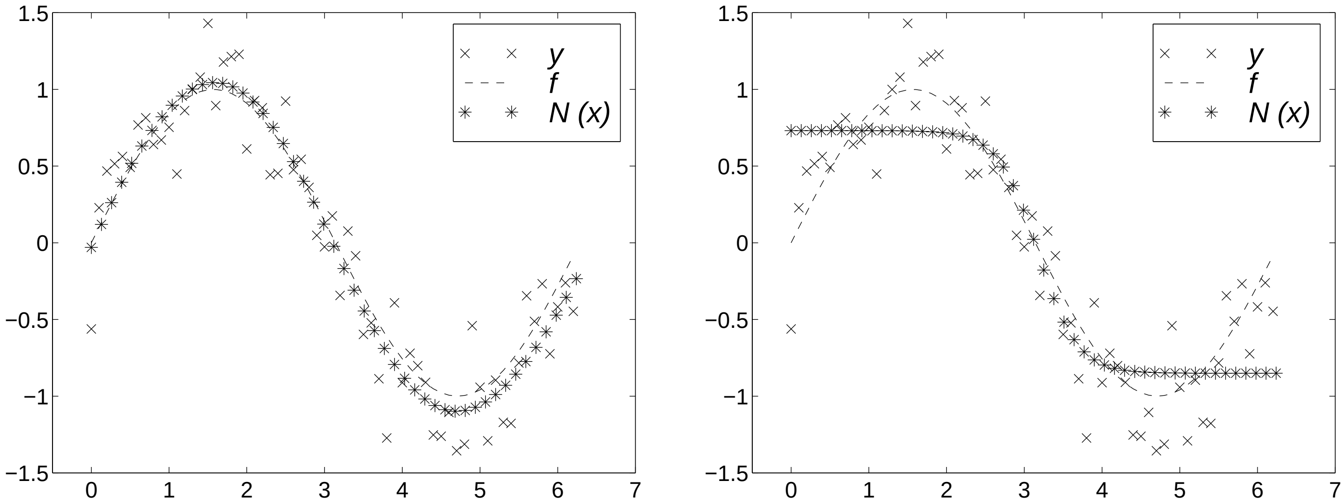
(ii) The correlation between the error-vectors and the action of layer $L-1$ is zero.

Some consequences:

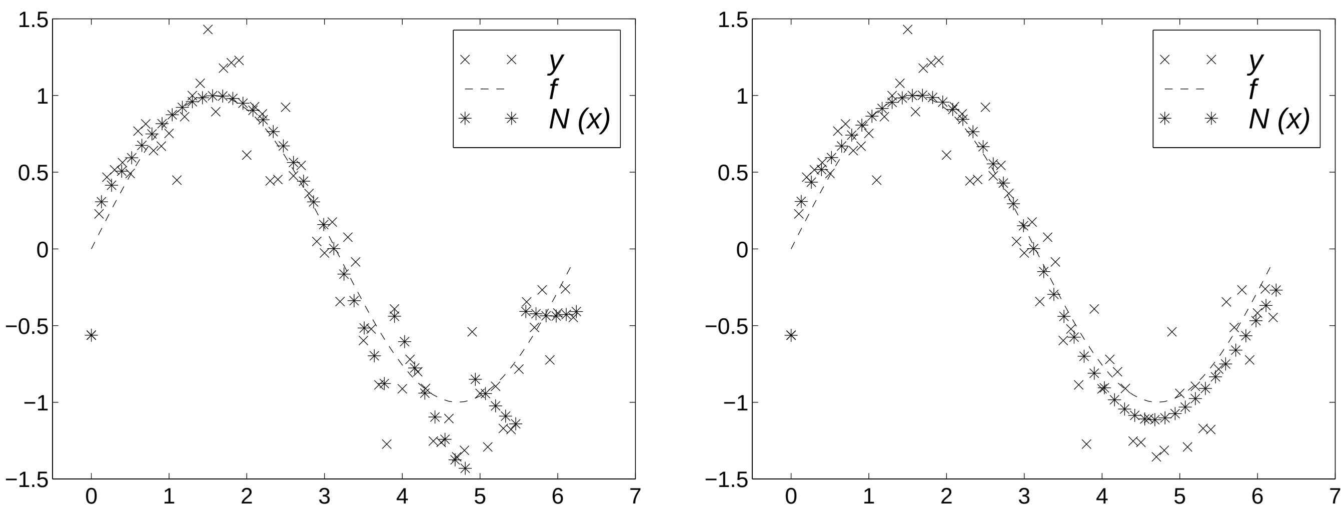
- just having final bias yields Cor. 1 (i)
  $\Rightarrow$ valid for all (linear or nonlinear) transformations having such structure
- Cor. 1 (i) shows that
  every $\mathcal{N}(\{\mathbf{W}^{l^*}\})$ *treats optimally Gaussian noise with zero mean*
  for the regression model $\mathbf{y}_i = \phi(\mathbf{x}_i) + \boldsymbol{\varepsilon}_i$.
- final layer activation does not give Cor. 1 (i) (unless zero-residual case)
  Note: sensitivity analysis also for this case follows
- backprop does not give Cor. 1
- (most likely on-line mode does not give Cor 1 (i))
- early stopping does not give Cor. 1 (i)
  $\Rightarrow$ all these can work better than the rigorous LMS-MLP for non-Gaussian (and/or non-functional) learning datas
  $\Rightarrow$ **BUT:** learning rate, number of epocs, stopping criterion, etc.
  <u>cannot</u> be explicitly controlled for this purpose!
  (termination due to algorithm or data?)
- Cor. 1 (i) explains how and why explicit change of prior frequency of different samples effects the trained MLP

Practical Difficulties with MLP:

- lots of local minima in optimization problem
  $\Rightarrow$ single local optimization not enough!
- large variation on number of iterations
  $\Rightarrow$ backprop and early stopping give what?
- How to choose the best MLP from local minima and from different configurations (e.g., size of hidden layer(s) and large set of activation functions) rigorously?



$n_1 = 2$: minimum of $\mathcal{J}^*$ (left) and maximum of $\mathcal{J}^*$ (right).



$n_1 = 7$: minimum of $\mathcal{J}^*$ (left) and maximum of $\mathcal{J}^*$ (right).

Possible remedy: regularization

Underlying idea: augment the LMS-cost with a penalization term that smooths the MLP-transformation (cf. Bayesian statistics):

$$\mathcal{J}_\beta(\mathbf{W}^1, \mathbf{W}^2) = \frac{1}{2N}\sum_{i=1}^{N} \|\mathcal{N}(\mathbf{x}_i) - \mathbf{y}_i\|^2 + \frac{\beta}{2}\sum_{l,i,j} |\mathbf{W}_{ij}^l|^2$$

Note: other possibilities for single weight penalization exist, but very often nonconvex and nonsmooth

in light of Cor. 1 (i): final bias should be excluded from regularization
$\Rightarrow$ different possibilities (cf. Cor. 1 (ii)):

I: regularize all other components except the bias-terms $\mathbf{W}_0^2$ in $\mathbf{W}^2$

II: exclude all components of $\mathbf{W}^2$ from regularization

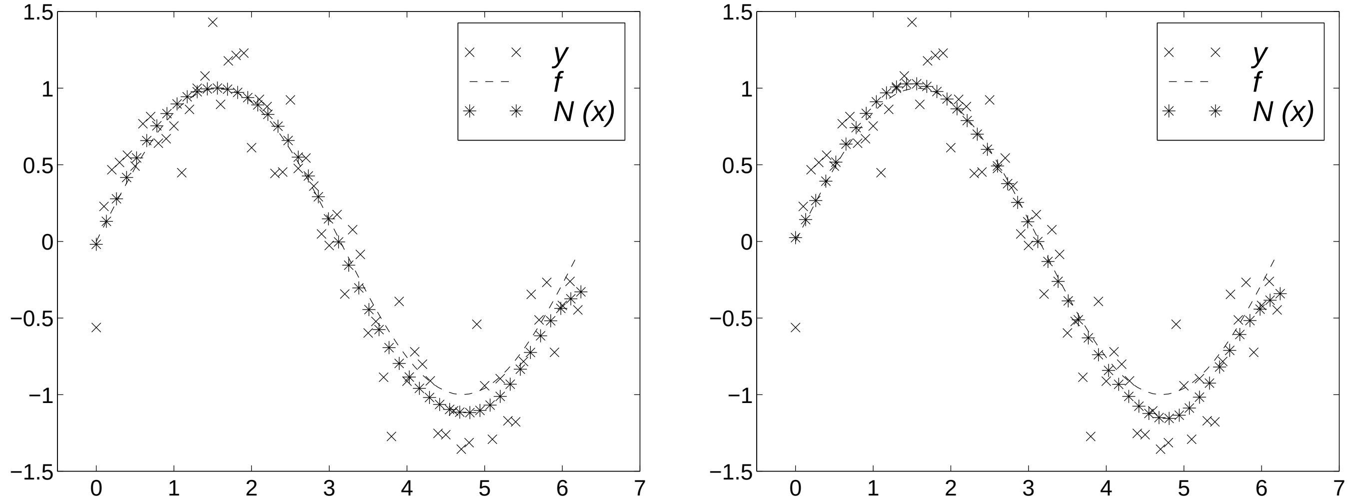
III: exclude all bias-terms of $(\mathbf{W}^1, \mathbf{W}^2)$ from regularization (cf. Holmström et al., 1997).

IV: exclude all components of $\mathbf{W}^2$ and bias-terms of $\mathbf{W}^1$ from regularization

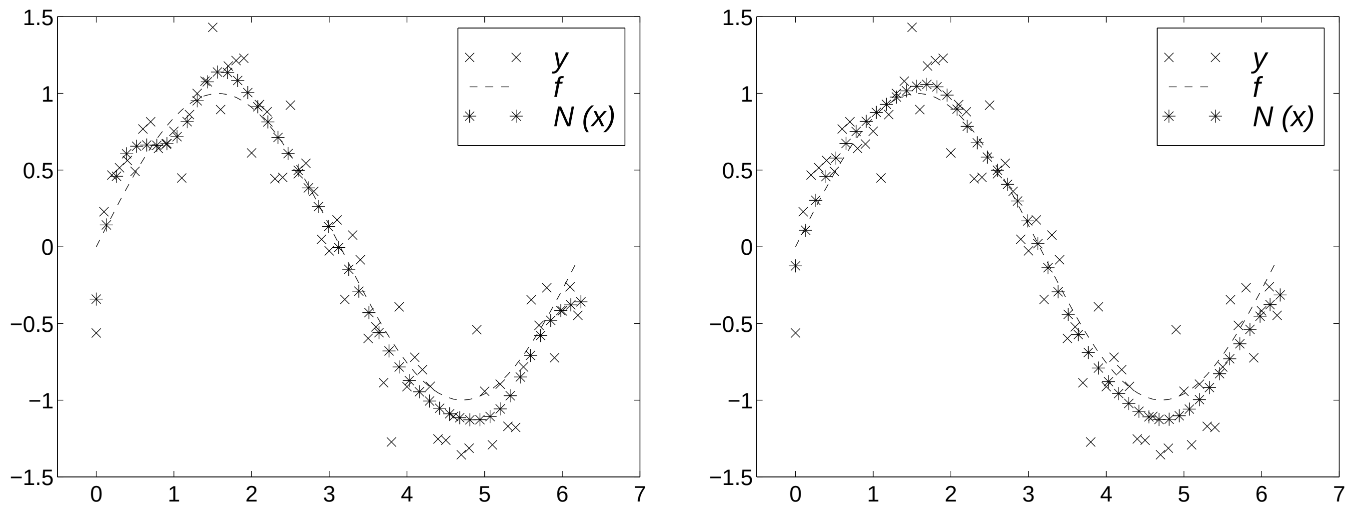
Without further ado:

- less (but still plenty of) local minima for I and III than for II, IV, and $\beta = 0$
- numerical confirmation that Cor. 1 (i) valid for all methods
- by means of number of iterations and CPU time I and III improved the performance, whereas II and IV made it worse compared to unregularized approach $\beta = 0$
- all regularization approaches (I and III in more stable way than II and IV) improved the generalization in simple nonlinear regression problem by preventing unnecessary oscillation
- **Conclusions:** I and III are more preferable than II and IV in every respect; between I and III no difference found

Effect of regularization III for $\beta = 10^{-3}$:



$n_1 = 7$: minimum of $\mathcal{J}^*$ (left) and maximum of $\mathcal{J}^*$ (right).



$n_1 = 15$: minimum of $\mathcal{J}^*$ (left) and maximum of $\mathcal{J}^*$ (right).