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
of good methods
and succesfull applications!

Course structure:

- lectures 7 weeks 2 h/week: 22.10.–3.12. Ag Beeta (TK, room Ag C415.1)
- exercises 7 weeks 4 h/week: 23.10.–12.12. (EH, room Ag C424.1):
  1. Wednesday 8–10 Ag B212.1 (until 4.12.2002 saakka),
  2. Thursday 14–16 Ag B211.1 (untila 12.12.2002, week 50 concerning seminar works)
- preparation of seminar works, period III 2003, weeks 2–9:
  1. weekly tutoring session on Thursday 14–16 Ag B212.1
- presentation and evaluation seminar 6–8 h: week 10, 2003.
- final examination by the end of period III 2003
- course evaluation:
  from the obligatory seminar work 3–12 points (coinciding with the grades 1- – 3) for the final examination: all other groups evaluate the work and presenation of one group, Erkki&Tommi can increase or decrease this evaluation by $+-\frac{1}{2}$

Lecture Topics:

- Introduction, about matrices and vectors
- Introcution to MATLAB graphics
- Introduction to Optimization
- MLP-networks I–III
- API/MEX and GUIs for MATLAB
- Seminar works

About MATLAB:

- Matrix Laboratory
- astonishingly complete environment for computational prototyping
- good build-in graphics
- possibility to link own external (sub)routines in C and Fortran
- poor man's version OCTAVE (`http://www.octave.org/`)
- extension possibilities using multiple *toolboxes*
- for our purposes:
  - definition of matrices and utilization of basic operations in linear algebra
  - fundamentals of optimization methods and their usage
  - data manipulation
  - testing and visualization of implemented operations using MATLAB graphics

About matrices and vectors (in MATLAB):

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \in \mathbf{R}^{n \times m}$$

can represent, e.g.

$$\left\{ \begin{array}{rcl} a_{11}x_1 + \ldots + a_{1n}x_n & = & y_1 \\ \vdots \quad + \ldots + \quad \vdots & = & \vdots \\ a_{n1}x_1 + \ldots + a_{nn}x_n & = & y_n \end{array} \right. \Leftrightarrow \mathbf{A}\mathbf{x} = \mathbf{y}$$

or

$$\mathbf{A} = \begin{bmatrix} z(x_1, y_1) & \dots & z(x_n, y_1) \\ \vdots & \ddots & \vdots \\ z(x_1, y_n) & \dots & z(x_n, y_n) \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_1 & \dots & x_n \\ \vdots & \ddots & \vdots \\ x_1 & \dots & x_n \end{bmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} y_1 & \dots & y_1 \\ \vdots & \ddots & \vdots \\ y_n & \dots & y_n \end{bmatrix},$$

and as a linear transformation $\mathbf{A} : \mathbf{R}^m \to \mathbf{R}^n$ simply means

$$\mathbf{y} = \mathbf{A}\,\mathbf{x} \;\Leftrightarrow\; y_i = \sum_{j=1}^{m} a_{ij}\,x_j.$$

Matrix operations (in MatLab):

(Identity matrix: $\mathbf{B} = \mathbf{I} \;\Leftrightarrow\; b_{ii} = 1\,\forall i; b_{ij} = 0\,\forall i \neq j$ (MatLab `b=eye(n)`)**)**

Transpose: $\mathbf{B} = \mathbf{A}^T \;\Leftrightarrow\; b_{ij} = a_{ji}$ (MatLab `b=a'`).

Scalar multiplication: $\mathbf{B} = t\mathbf{A} \;\Leftrightarrow\; b_{ij} = ta_{ij}$ (MatLab `b=t*a`).

Addition: $\mathbf{C} = \mathbf{A} + \mathbf{B} \;\Leftrightarrow\; c_{ij} = a_{ij} + b_{ij}$ (MatLab `c=a+b`).

Matrix multiplication: $\mathbf{C} = \mathbf{AB} \;\Leftrightarrow\; c_{ij} = \sum_k a_{ik}b_{kj}$ (MatLab `c=a*b`).

Componentwise multiplication: $c_{ij} = a_{ij}b_{ij}$ (MatLab `c=a.*b`).

Componentwise quotient: $c_{ij} = \frac{a_{ij}}{b_{ij}}$ $(\mathrm{b}_{ij} \neq 0)$ (MatLab `c=a./b`).

Matrix inverse: $\mathbf{B} = \mathbf{A}^{-1} \;\Leftrightarrow\; \mathbf{BA} = \mathbf{AB} = \mathbf{I}$ (MatLab `b=inv(a)`).
Note: only when $\mathbf{A}^{-1}$ exists, i.e. $\mathbf{A}$ is nonsingular (MatLab `det(a) ~= 0`).

Solution of linear equations:
left inverse: $\mathbf{Ax} = \mathbf{y} \;\Leftrightarrow\; \mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$ (MatLab `x=a\y`),
right inverse: $\mathbf{xA} = \mathbf{y} \;\Leftrightarrow\; \mathbf{x} = \mathbf{yA}^{-1}$ (MatLab `x=y/a`).