UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING

Νευρο-Ασαφής Υπολογιστική (Neuro-Fuzzy Computing)
Fall Semester 2025-2026
Δημήτριος Κατσαρός (prof. Dimitrios Katsaros)

Problem-set 2nd: **Teamwork** (2-persons)

Announcement: Friday, December 12, 2025
Delivery deadline: Sunday, January 11, 2026

SECTION 1: Working with first and second-order minimizers (Conjugate Gradient and Newton)

Problem-01

Consider the quadratic function:

$$F(x) = \frac{1}{2}\sum_{i=1}^{n} \lambda_i x_i^2$$

with $0 < m = \lambda_1 \leq \lambda_2 \leq \lambda_n = M$ and the starting point $x0=[1/m, 0, 0, \ldots, 0, 1/M]^T$. Find the minimum with Gradient Descent.

Problem-02

Consider minimizing the function $f(x)= x^4-1$. Solve it using the Newton method. Find the condition that must be satisfied so that the method converges. What the rate of convergence?

Problem-03

Consider the quadratic function $f(x)= x_1^2+5x_2^2+3x_3^2-4$ and starting point $x_0=[2,2,2]^T$. Find the minimum with the Conjugate Gradient method.

SECTION 2: Working with model reduction techniques in MLPs

Problem-04

Along the lines of **Problem-09** of the previous Problem-Set, implement a MLP with $S^1=80$. After training the neural network to convergence:

A. Prune a percentage (5% till 25%, step 2%) of the smallest – in magnitude – weights.
B. Prune a percentage (5% till 25%, step 2%) of weights in a random way.

Contrast the response (the accuracy) of the pruned MLPs with respect to the unpruned version of it. Discuss the obtained results. Perform any additional experiments to figure out the operation of weight pruning as a generalization technique.

C. Repeat the previous task, but now prune a percentage (1.25% till 5%, step 1.25%) of the largest – in magnitude – weights. Discuss again the obtained results. Is there any notion of super-weight(s) as observed in LLMs?

SECTION 3: Working with modern optimizers

Problem-05

Consider the following function:

$$F(w) = 0.1w_1^2 + 2w_2^2.$$

1. Find the minimum of the function implementing the Adadelta optimizer (instead of gradient descent) with learning rate α=0.4. Plot the algorithm's trajectory on a contour plot of $F(\mathbf{x})$.
2. Change the learning rate to α=3, and repeat the same task.
3. Try out Adadelta for the same objective function rotated by 45degrees, i.e., $F(w) = 0.1(w_1 + w_2)^2 + 2(w_1 - w_2)^2$. Does it behave differently?

Problem-06

Consider the following quadratic function:

$$F(x) = \frac{1}{2}x^T \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} x + \begin{bmatrix} 4 & -4 \end{bmatrix} x.$$

We want to use the steepest descent algorithm with momentum to minimize this function.

A. Perform two iterations (finding $\mathbf{x}_1$ and $\mathbf{x}_2$) of steepest descent with momentum, starting from the initial condition $\mathbf{x}_0$=[0 0]^T. Use a learning rate of α=1 and a momentum coefficient of γ=0.75.

B. Is the algorithm stable with this learning rate and this momentum? [Wait for the exercise in the class.]

C. Would the algorithm be stable with this learning rate, if the momentum were zero?

SECTION 4: Working with Convolutional Neural Networks

Problem-07

A. Is the max-pooling commutative with ReLU, i.e., maxPool[ReLU($\mathbf{x}$)]= ReLU[maxPool($\mathbf{x}$)]?

B. If yes, why should we choose to first apply max-pooling and then ReLU to a set of "pixels"?

Problem-08

Consider a 3D convolutional neural network with the following topology:

- INPUT: A 3-channel input image.
- LAYER-1: Convolutional layer with 16 5x5x5 convolutional filters.
- LAYER-2: Convolutional layer with 32 5x5x5 convolutional filters.
- LAYER-4: Dense layer with 200 units.
- LAYER-5: Dense layer with 64 units.
- LAYER-6: Single output unit.
- Batch Normalization after each convolutional layer.

How many weights does this network have?

Problem-09

Consider the image I found at: https://courses.e-ce.uth.gr/CE418/nfc_fall25/cat_grayscale.bmp.
Let's assume we have the following 3x3 filters:

$$F1=\begin{bmatrix}0 & -1 & 0\\ -1 & 8 & -1\\ 0 & -1 & 0\end{bmatrix},\ F2=\begin{bmatrix}0 & 1 & 0\\ 1 & 4 & 1\\ 0 & 1 & 0\end{bmatrix},\ F3=\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}.$$

A. Use 'valid' padding and a stride of one, compute the following convolutions: C1=I⊗F1, C2= I⊗F2 and C3= I⊗F3.
   a. Draw the output images.
   b. What does each filter do?

B. Record the wall-clock time required to compute C1, C2 and C3.

C. Do you think that it is possible to compute e.g., C1 without performing convolutions at all, but using only a matrix multiplication?
   a. If yes, implement it and record the respective wall-clock time.
   b. Do the same for C2 and C3.
   [Hint: Toeplitz matrix of a filter.]

D. Which approach is faster? Which is more memory demanding? Comment on that.

When recording wall-clock time, try to be fair for both approaches implemented in Question A and C. For instance, construction of a new matrix should be accounted for in your processing time.

Problem-10

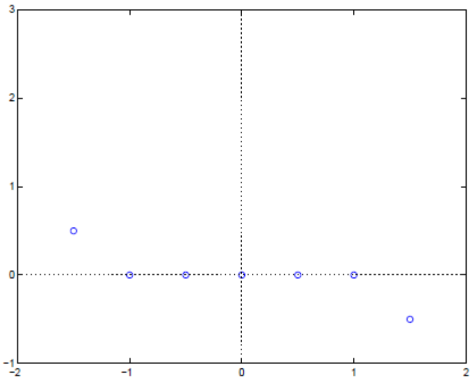
Assume that we have two convolution kernels of size k_1 and k_2, respectively (with no nonlinearity in between).

1. Prove that the result of the operation can be expressed by a single convolution.
2. What is the dimensionality of the equivalent single convolution?
3. Is the converse true?

SECTION 4: Working with Radial Basis Function Neural Networks

Problem-11

Choose the weights and biases for an RBF network with two neurons in the hidden layer and one output neuron, so that the network response passes through the points indicated by the blue circles in



Problem-12

Consider a 1-1-1 RBF network (one neuron in the hidden layer and one output neuron). The initial weights and biases are chose to be:

A. $w^1(0)= 0$, $b^1(0)= 1$, $w^2(0)= -2$, $b^2(0)= 1$.

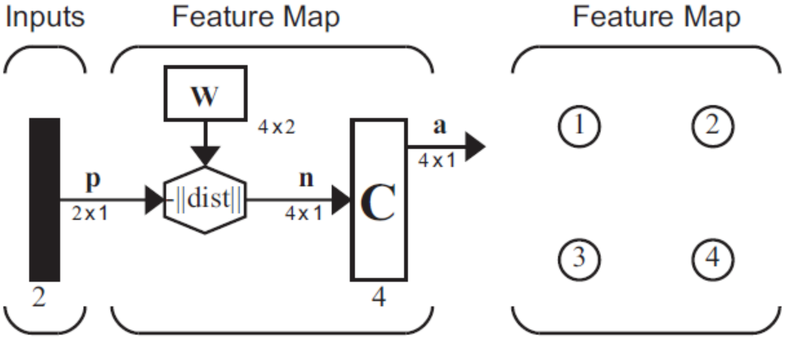
The training set has the following input/target pairs: $\{p_1=-1, t_1=0\}$, $\{p_2=1, t_2=1\}$.

Write a MATLAB/python program to implement two iterations of the steepest descent algorithm (i.e., backpropagation) for this RBF neural network with $\alpha=1$.

SECTION 5: Competitive Neural Networks

Problem-13

Consider the following feature map, where distance is used instead of inner product to compute the net input.



$$n_i = -\|{}_i\mathbf{w} - \mathbf{p}\|$$

$$\mathbf{a} = \mathbf{compet}(\mathbf{n})$$

The initial weight matrix is:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}^T$$

A. Plot the initial weights, and show their topological connections.

B. Apply the input $\mathbf{p}=[-1 \;\; 1]^T$, and perform one iteration of the feature map learning rule, with learning rate of $\alpha=0.5$, and neighborhood radius of 1.

C. Plot the weights after the first iteration, and show their topological connections.

Problem-14

An LVQ network has the following weights:

$$\mathbf{W}^1 = \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{bmatrix}^T, \; \mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

A. How many classes does this LVQ network have? How many subclasses?

B. Draw a diagram showing the first-layer weight vectors and the decision boundaries that separate the input space into subclasses.

C. Label each subclass region to indicate which class it belongs to.

D. Suppose that an input $\mathbf{p}= [1 \;\; 0.5]^T$ from Class 1 is presented to the network. Perform one iteration of the LVQ algorithm, with $\alpha=0.5$.

Useful information:

The deadline is strict. It is possible to get an extension (up to 4 days), but you need to get the approval of the instructor, and this is going to cost a 10% penalty to the final grade of this Problem-Set. Delivery of the solutions' pdf (typeset or very very easy to read handwritten) is done by email to dkatsar@uth.gr. The subject of the message should be **strictly**: CE418-Problem set 02: AEM1-AEM2

Symbol interpretation:

It requires “algorithmic” and/or mathematical thinking.

It requires the development of code (in any language/platform you wish). The final deliverable should contain: a) The solution of the exercise, and b) The implementation source code.