Single Layer Network

1 Activation functions

Activation functions can be divided into discrete/continuous and unipolar/bipolar. Some common activation functions (for $net = \sum_i w_i x_i - \theta$):

- **sign function:**

$$f(net) = \operatorname{sgn}(net)$$

- **'step' function:**

$$f(net) = \begin{cases} 1 & \text{for } net \geq 0 \\ 0 \ (\text{or } \ -1) & \text{for } net < 0 \end{cases}$$

- **sigmoid (unipolar):**

$$f(net) = \frac{1}{1 + e^{-net}}$$

- **sigmoid (bipolar):**

$$f(net) = \frac{2}{1 + e^{-net}} - 1$$

- **linear:**

$$f(net) = net$$

2 Single layer network

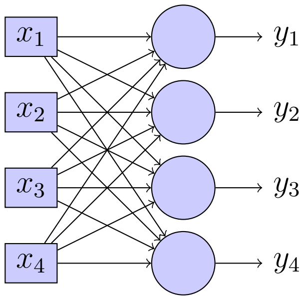


Figure 1: Example of a one-layer network with four inputs and four outputs.

In a neural network, classes can be represented in two ways:

- **Locally**: each neuron represents one class. For a given class only one neuron should have the value 1 and the others should have value 0. For a given example we choose the class represented by the neuron with the highest activation.
- **Globally**: each class is 'encoded' as a combination of outputs. K output layer neurons can represent 2^K classes.

For the single-layer network, we introduce the following notation:

- input vector: $\mathbf{x} = \begin{pmatrix} x_1 & \dots & x_J \end{pmatrix}^{\mathrm{T}}$
- output vector: $\mathbf{y} = \begin{pmatrix} y_1 & \dots & y_K \end{pmatrix}^{\mathrm{T}}$
- weight matrix:

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1J} \\ w_{21} & w_{22} & \dots & w_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K1} & w_{K2} & \dots & w_{KJ} \end{pmatrix}$$

  where w_{kj} is the weight of the j-th input of the k-th neuron.

- bias vector: $\boldsymbol{\theta} = \begin{pmatrix} \theta_1 & \dots & \theta_K \end{pmatrix}^{\mathrm{T}}$, where θ_k is the bias of the k-th neuron.
- activation function matrix: $\mathbf{\Gamma} = \mathrm{diag}[f(\cdot)]$. Multiplying this matrix by a vector determines the value of f for each element.

The output is calculated as:

$$\mathbf{y} = \mathbf{\Gamma}[\mathbf{W}\mathbf{x} - \theta]$$

Delta rule for one layer:

$$\mathbf{W}' = \mathbf{W} + \alpha(\mathbf{d} - \mathbf{y})\mathbf{x}^{\mathrm{T}}$$

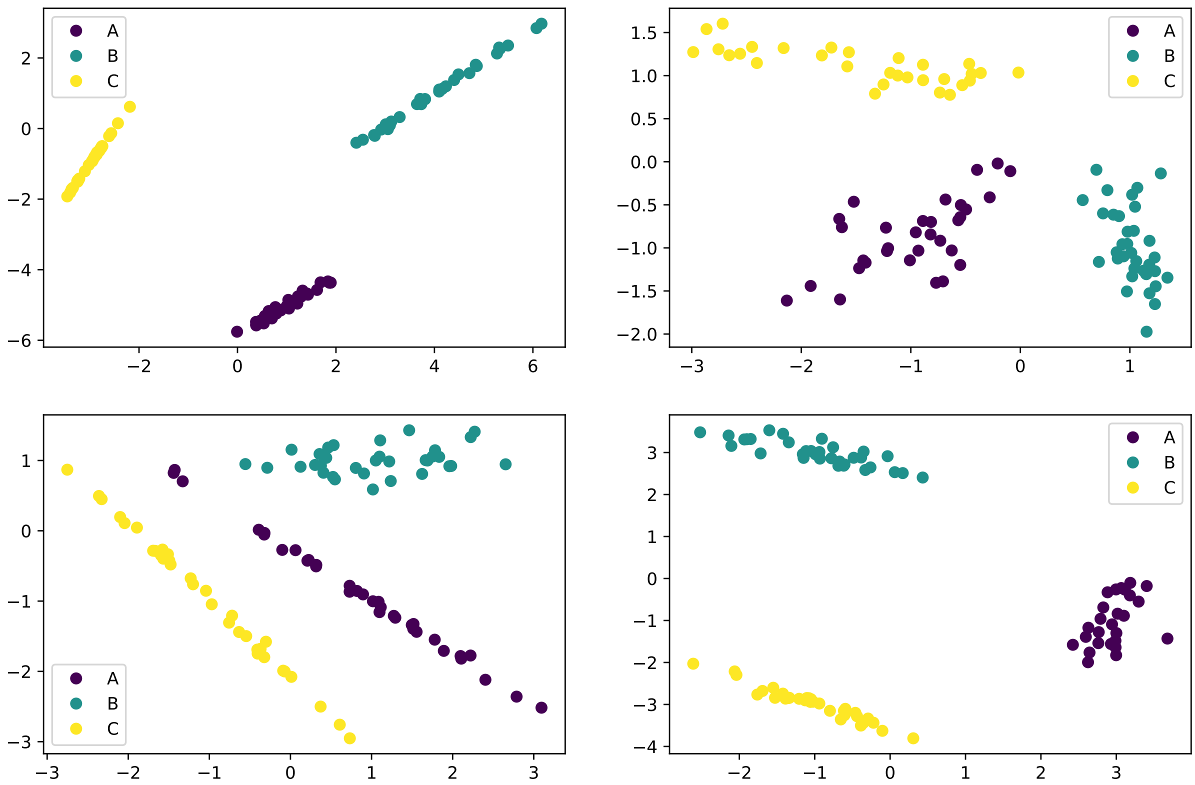
$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha(\mathbf{d} - \mathbf{y})$$

Questions

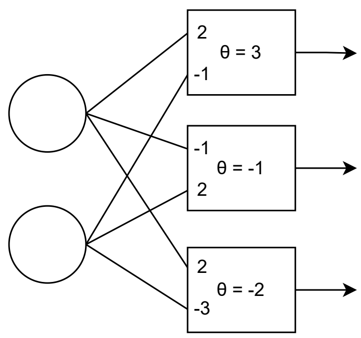
Zadanie 1.

Design a single-layer network to classify the following datasets.

- Which method of output representation (local/global) will be appropriate for each example?
- Draw the desicion boundary for each perceptron in the network.
- Write down the mapping of classes to network outputs.



Question 2.



For the single layer network above:

1. Write down the weight matrix.
2. Perform one learning step for the following inputs and expected outputs ($\alpha = 1$).
   - $\mathbf{x} = (3, 1)$, $\mathbf{d} = (0, 0, 1)$
   - $\mathbf{x} = (2, -8)$, $\mathbf{d} = (1, 1, 1)$
   - $\mathbf{x} = (2, 2)$, $\mathbf{d} = (1, 0, 1)$
   - $\mathbf{x} = (0, 1)$, $\mathbf{d} = (1, 1, 0)$

Mini-project: Single-layer network

The aim of the project is to create a single-layer netowrk to identify the language of input texts.

The files `lang.train.csv` and `lang.test.csv` contain a set of texts in four languages – English, German, Polish and Spanish. To classify a given text, count the number of occurences of each letter of the latin alphabet. For the purpose of this task, ignore all other characters (diacritics, puncturation, etc.) – only count the frequencies of the 26 letters of the latin alphabet.

For each text, generate a 26-element input vector containing the number of occurences of each letter. Then normalize it:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}.$$

The output of the network should have local representation: one perceptron should be assigned to each language. For a given text only the perceptron corresponding to its language should have value 1 – all others should have value 0.

You can use either a discrete 'step' activation function or a linear activation function ($f(net) = net$) (in both cases the weights are modified using the same formula, because for the linear function $f'(net) = 1$). To classify an input text, select the perceptron with the maximum activation.

Requirements:

- Train the network using data from `lang.train.csv` and output the test accuracy for data in `lang.test.csv`.
- The network should be able to adjust to different datasets, e.g. with a different number of languages.
- Provide an interface for the user to input new texts to classify (e.g. in the terminal).
- (Optionally) display incorrectly classified texts from the test set.

Hints:

- Some of the texts contain commas, so `split(",")` will not work when loading the files (but you can use for example `split(",", 1)` (Python) or `split(",", 2)` (Java)).
- Using a linear activation function decreases the chance of ambiguous network output (when more than one perceptron has output 1).
- Also normalize the weight vector.

- You can re-use your implementation of the perceptron from the previous project or implement the network from scratch using matrix operations.