Exercises
Learning and Intelligent Systems
SS 2016

Series 3, Apr 5th, 2016 (ANNs)

LAS Group, Machine Learning Institute
Dept. of Computer Science, ETH Zürich
Prof. Dr. Andreas Krause
Web: `http://las.ethz.ch/courses/lis-s16/`

Email questions to: Baharan Mirzasoleiman
`baharanm@inf.ethz.ch`

It is not mandatory to submit solutions and sample solutions will be published in two weeks. If you choose to submit your solution, please send an e-mail from your `ethz.ch` **address with subject** `Exercise3` **containing a PDF (LaTeX or scan) to** `lis2016@lists.inf.ethz.ch` **until Sunday, April 17th 2016.**

Problem 1 (Expressiveness of Neural Networks):

In this question we will consider neural networks with sigmoid activation functions of the form

$$\varphi(z) = \frac{1}{1+\exp(-z)}.$$

If we denote by v_j^l the value of neuron j at layer l its value is computed as

$$v_j^l = \varphi\left(w_0 + \sum_{i \in \text{Layer}_{l-1}} w_{j,i} v_i^{l-1}\right).$$

In the following questions you will have to design neural networks that compute functions of two Boolean inputs X_1 and X_2. Given that the outputs of the sigmoid units are real numbers $Y \in (0,1)$, we will treat the final output as Boolean by considering it as 1 if greater than 0.5 and 0 otherwise.

(a) Give 3 weights w_0, w_1, w_2 for a single unit with two inputs X_1 and X_2 that implements the logical OR function $Y = X_1 \vee X_2$.

(b) Can you implement the logical AND function $Y = X_1 \wedge X_2$ using a single unit? If so, give weights that achieve this. If not, explain the problem.

(c) It is impossible to implement the XOR function $Y = X_1 \oplus X_2$ using a single unit. However, you can do it using a multi-layer neural network. Use the smallest number of units you can to implement XOR function. Draw your network and show all the weights.

(d) Create a neural network with only one hidden layer (of any number of units) that implements

$$(A \vee \neg B) \oplus (\neg C \vee \neg D).$$

Draw your network and show all the weights.

Problem 2 (Building an RBF Network):

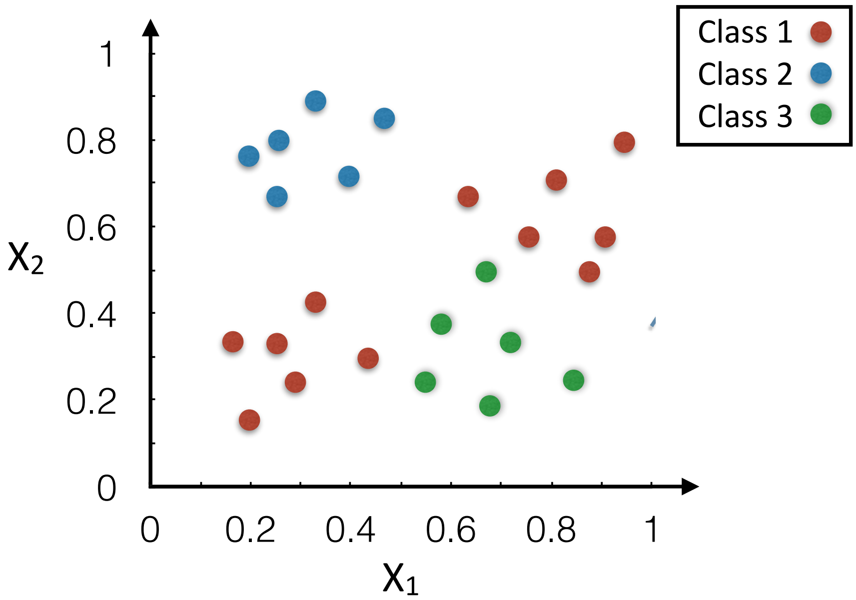
Radial basis function (RBF) networks are artificial neural networks that use radial basis functions as activation functions. They typically have three layers: an input layer, a hidden layer with a RBF activation function and a *linear* output layer. Hence, the output of the network is a linear combination of radial basis functions of the inputs and neuron parameters.

The input can be modeled as a vector of real numbers $\mathbf{x} \in \mathbb{R}^n$. Each output of the network $Y_j : \mathbb{R}^n \rightarrow \mathbb{R}$ is then given by

$$Y_j = \sum_{i=1}^{N} w_{ij} \exp(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)),$$

where N is the number of neurons in the hidden layer, μ_i and Σ_i are the mean vector and covariance matrix for neuron i, and w_{ij} is the weight of neuron i in the linear output neuron. In the basic form all inputs are connected to each hidden neuron.

Now, let us consider the following dataset:



(a) Draw an RBF network that perfectly classifies the given data points. Determine suitable values for the mean and covariance of each neuron in the hidden layer (μ_i, Σ_i and the appropriate weights w_{ij}) in the network.

Hint: You can assume that Σ_i *is a multiple of the identity matrix, so that* $Y_j = \sum_{i=1}^{N} w_{ij} \exp(-\frac{||\mathbf{x}-\mu_i||^2}{2\sigma_i^2})$.

(b) Argue why your network classifies the data points correctly. Pick one one of the data points and calculate the network output.