Exercises
Learning and Intelligent Systems
SS 2016

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

Series 3, April 25th, 2016 (ANNs)

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, Apr 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.

Solution 1:

(a) Figure 1 shows the value of $y = \frac{1}{1+e^{-x}}$ for different values of x. It can be seen that the value of the sigmoid function is greater than 0.5 for positive values of x, i.e., $y \geq 0.5$ if $x \geq 0$, and is less than 0.5 for negative values of x, i.e., $f(x) \leq 0.5$ if $x \leq 0$. Given this, we need to choose w_0, w_1, w_2 so that $w_0 + w_1 * x_1 + w_2 * x_2$ will be positive when $x_1 \vee x_2$ is equal to 1. One candidate solution is $[w_0 = -0.5, w_1 = 1, w_2 = 1]$.

(b) Similar to previous part, we can obtain $[w_0 = -1.5, w_1 = 1, w_2 = 1]$

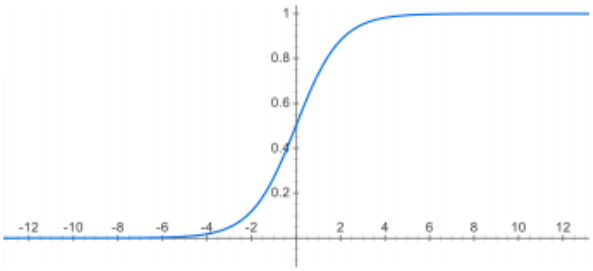
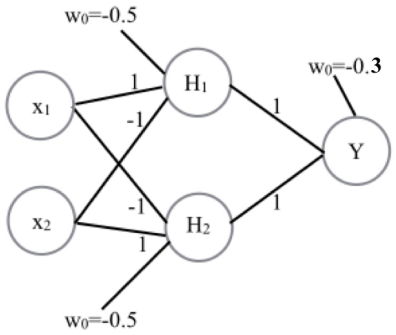
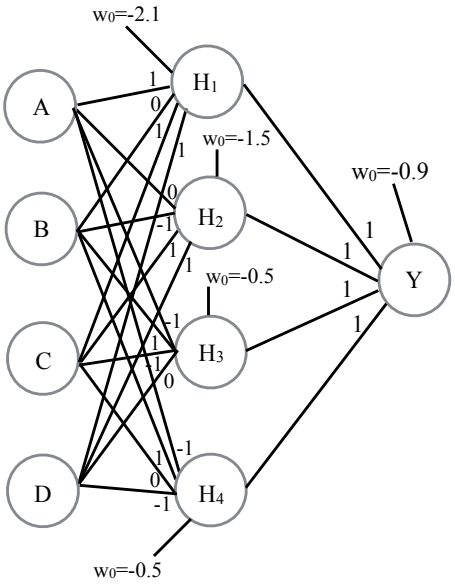


Figure 1: Sigmoid function $y = \frac{1}{1+e^{-x}}$

(c) It can be represented by a neural network with two nodes in the hidden layer. Input weights for node 1 in the hidden layer would be $[w_0 = -0.5, w_1 = 1, w_2 = -1]$, input weights for node 2 in the hidden layer would be $[w_0 = -0.5, w_1 = -1, w_2 = 1]$, and input weights for the output node would be $[w_0 = -0.8, w_1 = 1, w_2 = 1]$.



(d) Note that XOR operation can be written in terms of AND and OR operations: $p \oplus q = (p \wedge \neg q) \vee (\neg p \wedge q)$. Given this, we can rewrite the formula as $(A \wedge C \wedge D) \vee (\neg B \wedge C \wedge D) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg D)$. This formula can be represented by a neural network with one hidden layer and four nodes in the hidden layer (one unit for each parenthesis). An example is shown in the following Figure.



Problem 2 (Kernelized Linear Regression):

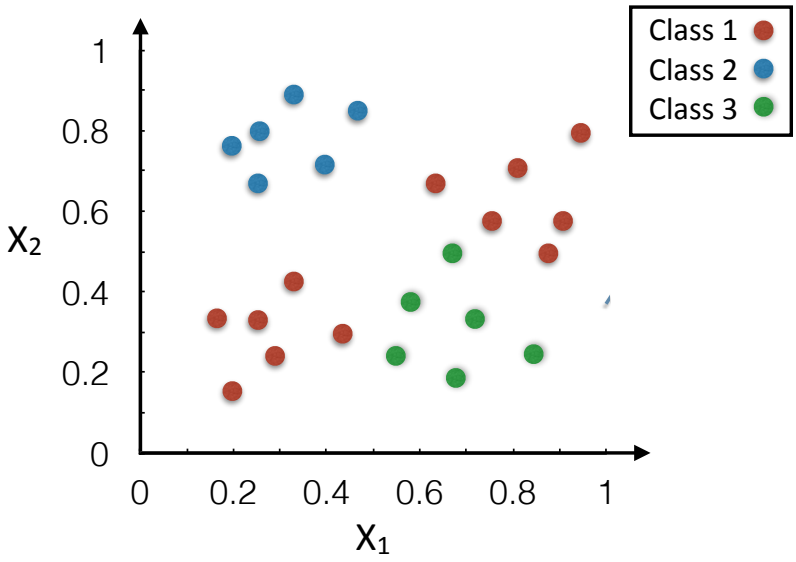
Radial basis function (RBF) networks are artificial neural networks that use radial basis functions as activation functions. They typically have three layers: an input layer, a hidden layer with a RBF activation function and a *linear* output layer. Hence, the output of the network is a linear combination 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 $Y_j : \mathbb{R}^n \to \mathbb{R}$ 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:



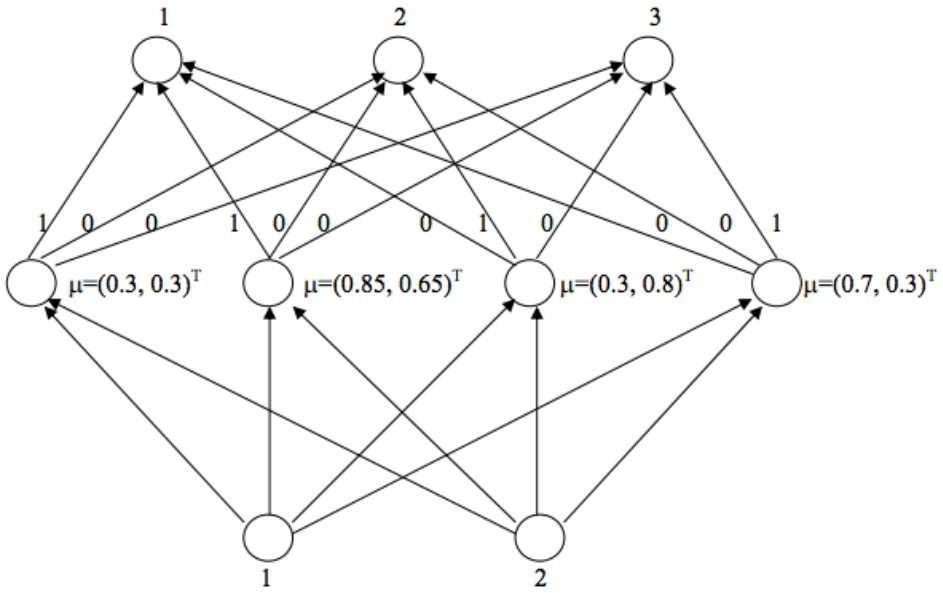
(a) Draw an RBF network that perfectly classifies the given data points. Determine suitable values for the mean and covariance of each neuron in the hidden layer (μ_i, Σ_i and the appropriate weights w_{ij}) in the network.

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.

Solution 2:

(a) We use Gaussian RBFs with no offset ("dummy") inputs. We model each cluster $C_i, i \in \{1, \cdots, 4\}$ with a Gaussian distribution. For class 1, we consider 2 separate clusters with mean $\mu_1 = [0.3, 0.3]^T$ and $\mu_2 = [0.85, 0.65]^T$. Similarly, for class 2, we consider a Gaussian distributions with mean $\mu_3 = [0.3, 0.8]$ and for class 3, we consider a Gaussian distribution with $\mu_4 = [0.7, 0.3]^T$. For all hidden (RBF) units, $\sigma =$



0.1 to get an appropriate separation between high activation for inputs from within the relevant cluster and low activation for other inputs.

The output neurons are linear neurons. The network output is defined as the number of the output neuron with the greatest activation (1, 2, or 3).

(b) This network uses four RBF (hidden-layer) neurons whose reference vectors are placed in the centers of the four clusters in the input space. Hidden-layer neurons 1 and 2 yield greatest responses for inputs from class 1, neuron 3 for class 2, and neuron 4 for class 3. The output-layer weights are chosen to reflect these relationships; connections between hidden-layer neurons and output neurons for the desired output are set to 1, and all other output-layer weights are set to 0. This way, any input from within one of the four clusters will give the desired output value the highest activation.

For example, let us pick the exemplar with input vector $x = [0.4, 0.7]^T$ and desired output 2. For a given cluster C_i, the activation of each RBF neuron is computed with a Gaussian distribution as follows:

$$p(x \in C_i) \propto \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

Thus, we get the following outputs from the RBF neurons:

$p(x \in C_1) \propto \exp\left(-\frac{(0.4-0.3)^2+(0.7-0.3)^2}{2\times 0.1^2}\right) = 0.000203.$
$p(x \in C_2) \propto \exp\left(-\frac{(0.4-0.85)^2+(0.7-0.65)^2}{2\times 0.1^2}\right) = 0.000013.$
$p(x \in C_3) \propto \exp\left(-\frac{(0.4-0.3)^2+(0.7-0.8)^2}{2\times 0.1^2}\right) = 0.368.$
$p(x \in C_4) \propto \exp\left(-\frac{(0.4-0.7)^2+(0.7-0.3)^2}{2\times 0.1^2}\right) = 0.000004.$

Given the output weights, we get the following activations of the output neurons:

Output 1 = 0.000203 + 0.000013 = 0.000216.
Output 2 = 0.368.
Output 3 = 0.000004.

Clearly, the output of the network is 2, which is the desired output for the given exemplar.