Exercises
Learning and Intelligent Systems
SS 2016

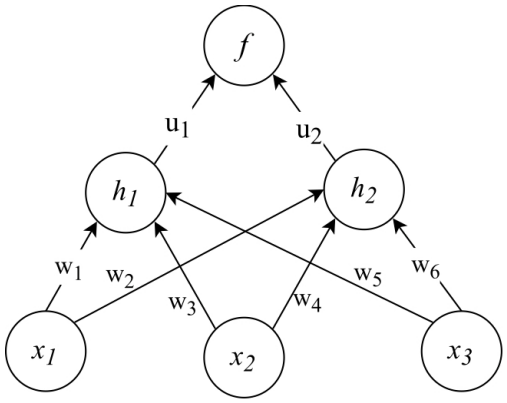
Series 3, April 5th, 2017 (ANNs)

LAS Group, Institute for Machine Learning
Dept. of Computer Science, ETH Zürich
Prof. Dr. Andreas Krause
Web: `http://las.inf.ethz.ch/teaching/lis-s16/`
Email questions to:
Felix Berkenkamp, `befelix@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, Apr 17th 2017.

Problem 1 (Neural network derivatives):

Consider the following neural network with two logistic hidden units h_1, h_2, and three inputs x_1, x_2, x_3. The output neuron f is a linear unit, and we are using the squared error cost function $E = (y - f)^2$. The logistic function is defined as $\phi(x) = 1/(1 + e^{-x})$.



(a) Consider a single training example $\boldsymbol{x} = [x_1, x_2, x_3]$ with target output (label) y. Write down the sequence of calculations required to compute the squared error cost (called forward propagation).

(b) A way to reduce the number of parameters to avoid overfitting is to tie certain weights together, so that they share a parameter. Suppose we decide to tie the weights w_1 and w_4, so that $w_1 = w_4 = w_{\text{tied}}$. What is the derivative of the error E with respect to w_{tied}, i.e. $\nabla_{w_{\text{tied}}} E$?

Solution 1:

(a) Solution: $h_1 = \phi(x_1 w_1 + x_2 w_3 + x_3 w_5)$

$h_2 = \phi(x_1 w_3 + x_2 w_4 + x_3 w_6)$

$f = u_1 h_1 + u_2 h_2$

(b)

$$\begin{aligned}\frac{\partial E}{\partial w_{\text{tied}}} =& \frac{dE}{df}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial w_{\text{tied}}} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial w_{\text{tied}}}\right) \\ =& -2(y-f)\Big[u_1\phi^{'}(x_1w_1 + x_2w_3 + x_3w_5)x_1 + u_2\phi^{'}(x_1w_3 + x_2w_4 + x_3w_6)x_2\Big] \\ =& -2(y-f)\Big[u_1h_1(1-h_1)x_1 + u_2h_2(1-h_2)x_2\Big]\end{aligned}$$

Problem 2 (Building an RBF Network):

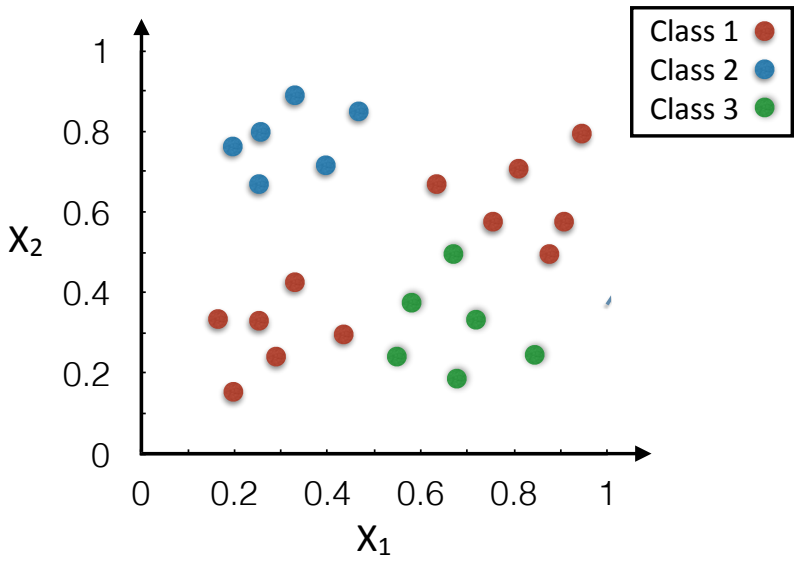
Radial basis function (RBF) networks are artificial neural networks that use radial basis functions as activation functions. They typically have three layers: an input layer, a hidden layer with a RBF activation function and a *linear* output layer. Hence, the output of the network is a linear combination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

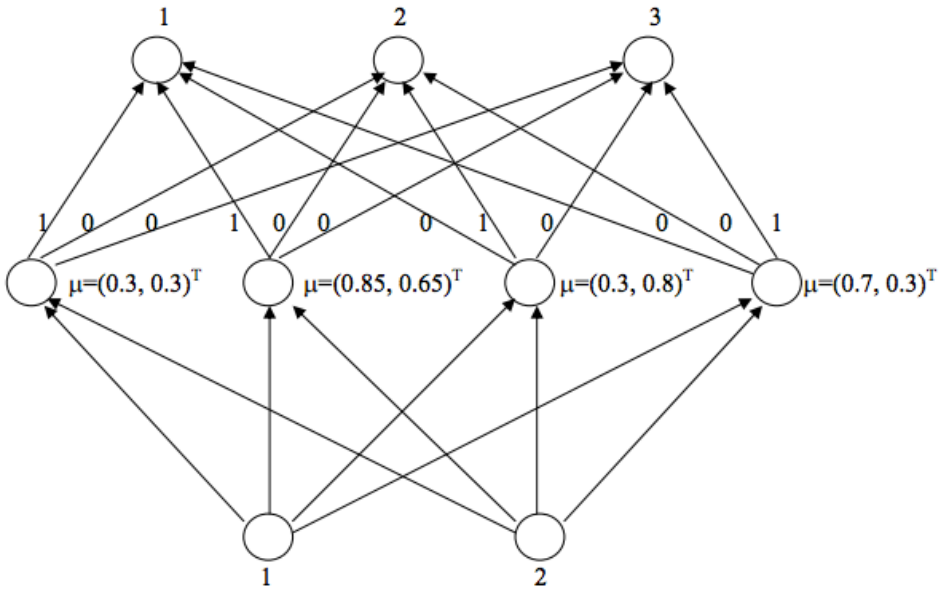


(a) Draw an RBF network that perfectly classifies the given data points. Determine suitable values for the mean and covariance of each neuron in the hidden layer (μ_i, Σ_i and the appropriate weights w_{ij}) in the network.

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.