Homework 3

MACHINE LEARNING — 6 November, 2013
1st Semester (FEUP-PDEEC) — 18-782PP CMU-Portugal

Submit by 20 November , 2013, 23h59
by email to `jaime.cardoso@fe.up.pt`
and `vikramg@andrew.cmu.edu`

Problem 1

Implement in Matlab the Parzen window density estimation using the spherical Gaussian window function

$$\phi((\mathbf{x}-\mathbf{x_i})/h) \propto exp[-(\mathbf{x}-\mathbf{x_i})^t(\mathbf{x}-\mathbf{x_i})/(2h^2)]$$

Write a program to classify an arbitrary test point x based in the Parzen window estimates. Train your classifier using the 3-dimensional data from 3 categories in `parzenData.txt`.

Set $h = 1$ and classify the following three points:
$(0.5, 1, 0)^t$; $(0.31, 1.51, -0.5)^t$; $(-0.3, 0.44, -0.1)^t$.

Problem 2

(a) Generate 100 observations from the model

$$y = \sigma(\mathbf{a}_1^t\mathbf{x}) + (\mathbf{a}_2^t\mathbf{x})^2 + 0.30z$$

where $\sigma(.)$ is the sigmoid function, z is a standard normal random variable, $\mathbf{x} = [x_1\ x_2]^t$, with each x_j being independent standard normal, and $\mathbf{a}_1 = [3\ 3]^t$ and $\mathbf{a}_2 = [3\ \ -3]^t$. Plot the results.

(b) Write a Matlab function
`prediction = trainAndTestNeuralNet(trainingData, testData)`
to fit a single hidden layer neural network with 10 hidden units via back-propagation and weight decay to observations generated according to the previous model. Choose appropriate activation functions and explain your choice.

(c) Generate a training set with 200 observations and a test sample of size 1000, and plot the training and test error curves as a function of the number of training epochs, for different values of the weight decay parameter. Discuss the overfitting behaviour in each case.

Homework 3

Problem 3

K-Nearest Neighbor can be used to find $\hat{P}(C_m|x)$, where C_m is class m, as following:

$$\hat{P}(C_m|x) = \frac{K_m}{K} \tag{1}$$

where K_m is the number of patterns among the K nearest neighbors that belong to class C_m. Implement a function `knnclassifier` that returns the posterior probabilities of each point as well.

Let us illustrate the method through a simple example. Consider the case of using $K = 7$, three class case. Assume for a particular point x neighbors 2, 3, 6, and 7 belong to class 1, neighbors 1 and 4 belong to class 2, and neighbor 5 belongs to class 3. Let the weights attached to the 7 neighbors be $v_1, ..., v_7$, and assume that these weights are greater than or equal zero and sum to 1. The posterior probabilities are given as:

$$\hat{P}(C_1|x) = v_2 + v_3 + v_6 + v_7, \quad \hat{P}(C_2|x) = v_1 + v_5, \quad \hat{P}(C_3|x) = v_5$$

The optimal weights $v1, ..., v7$ are determined by maximizing the likelihood of the data. We can denote weights as the following:

$$v_i = \frac{e^{w_i}}{\sum_{j=1}^{K+1} e^{w_j}}, \qquad i = 1, 2, ...K + 1 \tag{2}$$

Now, we construct a matrix B such that $B_{ij} = 1$ if j^{th} neighbor belongs to class C_i, and zero otherwise. The $K+1$'th column corresponds to the extra weight w_{K+1} introduced above, and all its entries equal $1/M$ (M is the number of classes). Now, the posterior probabilities are given by:

$$\hat{P}(C_m|x) = \frac{\sum_{i=1}^{K+1} B_{mi} e^{w_i}}{\sum_{j=1}^{K+1} e^{w_j}} \tag{3}$$

The goal of this problem is to implement the following algorithm to find optimal weights.
1) Start with some initial weights, say equal values.
2) Update the weights in the direction of the gradient of log likelihood function of Equation 2:

$$w_i(new) = w_i + \eta[\sum_{n=1}^{N} \frac{B_{\mu i} e^{w_i}}{\sum_{j=1}^{K+1} B_{\mu j} e^{w_j}} - \frac{N e^{w_i}}{\sum_{j=1}^{K+1} e^{w_j}}] \tag{4}$$

where η is the step size and the term multiplying η is the gradient. μ denotes the class a training sample x_n belongs to.
3) Repeat step 2 until the change in weights is negligible.

Now, use the Kullback divergence measure to compare the accuracy of measured posterior probabilities in Equation 1 and Equation 3 against the actual posterior probabilities of test samples, respectively.

$$d(P,\hat{P}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} P(C_m|x_n) log[\frac{P(C_m|x_n)}{\hat{P}(C_m|x_n)}] \tag{5}$$

Use the following mixture of Gaussians to generate 600 training and 600 test samples.

$$p(x|C_1) = \frac{1}{3*2\pi}[e^{-\frac{1}{2}(x_1^2+x_2^2)} + e^{-\frac{1}{2}((x_1-4)^2+x_2^2)} + e^{-\frac{1}{2}((x_1-2)^2+(x_2+2)^2)}]$$

$$p(x|C_2) = \frac{1}{3*2\pi}[e^{-\frac{1}{2}(x_1^2+(x_2-4)^2)} + e^{-\frac{1}{2}((x_1-2)^2+(x_2-2)^2)} + e^{-\frac{1}{2}((x_1-4)^2+(x_2-4)^2)}]$$

Find values of $d(P,\hat{P})$ for $K = 10, 20, 30, 40$ in a table where $\hat{P}$ is denoted by Equation 1 and Equation 3 respectively. Start with weights = 2, and set $\eta = 0.1$.