INTRODUCTION TO MACHINE LEARNING
CS5011
ASSIGNMENT 3

Aritra Ghosh
CS11B062

10 October,2013

Question4

Back propagation has been implemented and the measures have been reported as below.The algorithm is in line with what has been taught in class and written in the book
Precision1:0.3478
Recall1: 0.400
F1 measure1:0.3721
Precision2:0.7895
Recall2:0.75
F1 measure2: 0.7692
Precision3:0.5789
Recall3:0.55
F1 measure3:0.5641
Precision4:0.4737
Recall4:0.45
F1 measure4:0.4737

The over all accuracy for training data is 93.34% and the accuracy for testing data is 57.5% A large learning rate may not be able to reach the local minima as it keeps oscillating because of taking larger steps.

PART 2

For K class classification, there are K units at the top, with the kth unit modeling the probability of class k. There are K target measurements $Y_k,\ k = 1,\ \ldots,\ K$, each being coded as a 0–1 variable for the kth class. Derived features Z_m are created from linear combinations of the inputs, and then the target Y_k is modeled as a function of linear combinations of the Z_m,

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), m = 1\ \ldots,\ M,$$

$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1,\ ,\ K,$$

$$f_k(X) = g_k(T) k = 1,\ \ldots,\ K,$$

where $Z = (Z_1,\ Z_2,\ \ldots,\ Z_M)$, and $T = (T_1,\ T_2,\ \ldots,\ T_K)$.

The activation function $\sigma(v)$ is usually chosen to be the *sigmoid* $\sigma(v) = 1/(1 + e^{-v})$.For the purpose of classification g_k is chosen to be softmax function but in this case we do not use it since it is computationally expensive and difficult to differentiate.We use sigmoid function instead.

The neural networks has unknown parameters ,often called weights and we seek values for them that make the model fit the training data well. We denote the complete set of weights by θ which consists of
$\{\alpha_{0m},\ \alpha_m; m = 1, 2,\ \ldots,\ M\}M(p + 1)$ weights,

$\{\beta_{0k},\ \beta_k; k = 1, 2,\ \ldots,\ K\}K(M + 1)$ weights.

The $R(\theta)$ suggested here for the second part is

$$R(\theta) = \sum_{k=1}^{K}\sum_{i=1}^{N}(y_{ik} - f_k(x_i))^2 + \gamma\sum_{km}\beta_{km}^2 + \gamma\sum_{ml}\alpha_{ml}^2$$

and $\gamma >= 0$ is the training parameter.This $R(\theta)$ is used because we don't want an overfit solution. Instead we impose some regularization which is achieved through a penalty term.Let $z_{mi} = \sigma(\alpha_{0m}+\alpha_m^T x_i)$ and let $z_i = (z_{1i},\ z_{2i},\ \ldots,\ z_{Mi})$. Then we have

$$R(\theta) = \sum_{i=1}^{N} R(i) + \sum_{km}\beta_{km}^2 + \sum_{ml}\alpha_{ml}^2 = \sum_{i=1}^{N} R_1(i)$$

with derivatives

$$\frac{\partial R_{1i}}{\partial\beta_{km}} = -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi} + 2\gamma\beta_{km}, \qquad (1)$$

$$\frac{\partial R_{1i}}{\partial\alpha_{m\ell}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell} + 2\gamma\alpha_{ml} \qquad (2)$$

Given these derivatives, a gradient descent update at the $(r + 1)$ st iter- ation has the form

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r\sum_{i=1}^{N}\frac{\partial R_{1i}}{\partial\beta_{km}^{(r)}}, \qquad (3)$$

$$\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r\sum_{i=1}^{N}\frac{\partial R_{1i}}{\partial\alpha_{m\ell}^{(r)}}, \qquad (4)$$

where γ_r is the *learning rate*

Now we write

$$\frac{\partial R_{1i}}{\partial\beta_{km}} = \delta_{ki}z_{mi} + 2\gamma\beta_{km}, \qquad (5)$$

$$\frac{\partial R_{1i}}{\partial\alpha_{m\ell}} = s_{mi}x_{i\ell} + 2\gamma\alpha_{ml}. \qquad (6)$$

The quantities δ_{ki} and s_{mi} are "errors" from the current model at the output and hidden layer units, respectively. From their definitions, these errors satisfy

$$s_{mi} = \sigma^{'}(\alpha_m^T x_i)\sum_{k=1}^{K}\beta_{km}\delta_{ki} \qquad (7)$$

Using this, the updates in (3) and (4) can be implemented with a two-pass algorithm. In the *forward pass*, the current weights are fixed and the predicted values $\hat{f}_k(x_i)$ are computed . In the *backward pass*, the errors δ_{ki} are computed, and then back-propagated via (7) to give the errors s_{mi}. Both sets of errors are then used to compute the gradients for the updates in (3) and (4), via (5) and (6).This method is also known as weight decay and is analogous to ridge regression used in linear models. Larger values of λ tend to shrink the weights towards 0. The best value of γ is 0.1 and the accuracy is around 65%. All the measures are present in "gamma.txt".As the value of γ increases we see that it classifies all of them to a single class as is evident in the 10 and 100 case.An increase in the γ shrinks the weights.Zero γ corresponds to overfitted network. Infinitely large γ gives us underfitted network, equal to some constant. Between these extremal values there is a range of networks which reproduce dataset with different degrees of precision and smoothness.