DATA MINING ASSIGNMENT 8B (Neural Networks)

Debarati Das

1PI13CS052

OVERVIEW

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. One of the preferred techniques for gesture recognition.
3. MLP/Neural networks do not make any assumption regarding the underlying probability density functions or other probabilistic information about the pattern classes under consideration in comparison to other probability based models.
4. They yield the required decision function directly via training.
5. A two layer backpropagation network with sufficient hidden nodes has been proven to be a universal approximator.

Their ability to learn by example makes neural nets very flexible and powerful. There is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. Along various other advantages of

Neural nets there disadvantages too they cannot be programmed to perform a specific task; the examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. Also, network finds out how to solve the problem by itself, hence its operation can be unpredictable. The problem with the backpropagation algorithm is that it tries to find a local minimum in the error function output, if it ends up in finding the wrong one, the results can drastically bad, and that's why learning rate is important. Instead of using a simple backpropagation algorithm advanced algorithms like hyper rectangular composite NN (HRCNN) using supervised decision directed learning (SDDL) can be used.

ANSWERS TO QUESTIONS

- **How many nodes were used in the intermediate layer and why?**
    - The number of hidden nodes you should have is based on a complex relationship between:
        - Number of input and output nodes
        - Amount of training data available
        - Complexity of the function that is trying to be learned
        - The training algorithm
    - To minimize the error and have a trained network that generalizes well, you need to pick an optimal number of hidden layers, as well as nodes in each hidden layer. *Too few nodes* will lead to high error for your system as the predictive factors might be too complex for a small number of nodes to capture. *Too many nodes* will overfit to your training data and not generalize well. The number of hidden nodes in each layer should be somewhere between the size of the input and output layer, potentially the mean. The number of hidden nodes shouldn't need to exceed twice the number of input nodes, as you are probably grossly overfitting at this point.
    - **The default settings set up a hidden layer with number of nodes equal to (number of attributes + number of classes) / 2.** This can be observed when the no of hidden layers parameters is set as 'a' and returns number of nodes in intermediate layer as **5** (8+2/2).

- We could try adding more hidden layers and/or adjust the number of nodes in the layers. However, assuming that our 75.39% result is from a cross-validation, it may not be possible to get a much better result and making the structure more complex may result in overfitting and worse performance. (I increased the number of layers to 8 and it gave 74.8% accuracy)

- **What was the terminating criterion?**

    - In general, back-propagation algorithm cannot be shown to convergence, and there are no well-defined criteria for stopping its operation. Some alternative stopping criteria are stated below:
        - Prespecified number of epochs has expired.This appears to be what is used by default in WEKA. **When the epochs expire the model can be accepted.**
        - Percentage of tuples misclassified is below some threshold.
        - The back-propagation algorithm is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small.
        - Δwij in the previous epoch is so small that it is below some specified threshold.

- **What learning rate was used?**

    - The learning rate is how much weight it gives for new examples. The learning rate (usually a fraction between about 0.0 and 1.0) will certainly affect how quickly the network is trained--i.e., it can move you to the local minimum more quickly.
    - **The learning rate specified by default is 0.2.** It is a rule of thumb to set learning rate as 1/(number of iterations) so as to give it an optimal value.

I increased the learning rate = 0.5 and observed an increase in accuracy (75.52 from 75.39%). We can even add a momentum keeping the learning rate constant. This would be like a multiplier to the learning rate. When m=0.1 , accuracy is 75% and when m=0.2 it is 75.52%, keeping l=0.5)