TIM 245 Lecture 12 (8/15/17)

Agenda

1) Feedback on Phase III
2) Review Lecture 11
3) Training Neural Networks
4) Network Design and Applications
5) Work on project (time permitting)
3 Training Neural Networks

Objective: find the weights $W$ that minimize the error or cost function for the training dataset.

Input Layer  Hidden Layer  Output Layer

Issue: what is the error for the hidden units?
Backpropagation solves this problem by breaking the optimization process into two steps:

1) **Feed Forward**: run training data through the network to get the output of each node (computed in the forward direction).

2) **Backpropagation**: compute the gradient for each node in order to determine which direction to move and the update the weight (computed in the backwards directions).

Repeat n times or until convergence criteria is met (early stopping).

Activation function needs to be differentiable.
**Activation Function**: allows the network to exhibit non-linear behavior, i.e. squashing the weighted sum of the neuron.

Common activation functions:

**Sigmoid**: \( f(x) = \frac{1}{1 + e^{-x}} \) \([0,1]\)

- Traditionally used because it is similar to biological neurons. Saturates at 0 (Vanishing gradient problem)

**Tanh**: \( f(x) = 2\sigma(2x) - 1 \) \((\sigma(x) = \text{sigmoid})\) \([-1,1]\)

- Scaled sigmoid
- Saturates at -1, 1

**Relu**: \( f(x) = \max(0, x) \)

- Fast training and convergence. Neurons can "die" during training.
4) Neural Network Design

Every neural network consists of three parts:

1) **Input Layer**

One node for each attribute in the dataset $X_1, X_2, \ldots, X_m$

2) **Output Layer**

Two cases

- **Classification**
  - One output node for each class label with a Softmax activation function

  \[
  \text{Softmax} : f(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{\infty} e^{x_i}}
  \]

  Select highest probability class

- **Prediction**
  - One (or more) output nodes with a linear activation function
3) **Hidden Layer**

Increasing the number of hidden layers allows the network to model more complex functions.

0 layers $\rightarrow$ linear decision boundary (separating hyperplane)

1 layer $\rightarrow$ convex polygon regions (universal approximator)

2 layers $\rightarrow$ compositions of polygons

Diminishing returns for most classification problems after 2 layers.
Number of nodes in the hidden layer typically through trial and error with some general guidelines.

- \( 2m \) where \( m \) is the number of attributes

- \( 2 \log m \)

- \( \frac{2}{3}m \)

- \( \sqrt[n]{N \cdot m} \) where \( N \) is the number of training instances

Start small and add nodes until there is no performance gain
Key design considerations

1) Overfitting

- **Regularization**
  - Put L1 or L2 norm penalties on the weights in the cost function (constrains weights)

- **Dropout**
  - Each neuron has a probability, p, of dropping out of the network (forces generalization)

2) Training time

- **Normalization**
  - Reduces the distance between the initial and final weights
  - Minimizes the impact of outliers

- **Learning rate**
  - How quickly the network abandons old beliefs for new ones.
  - Adaptively change and during training (momentum)