TIM 245 Lecture 12 (5/15/17)

Agenda

1) Feedback on Phase III

2) Review Lecture 11

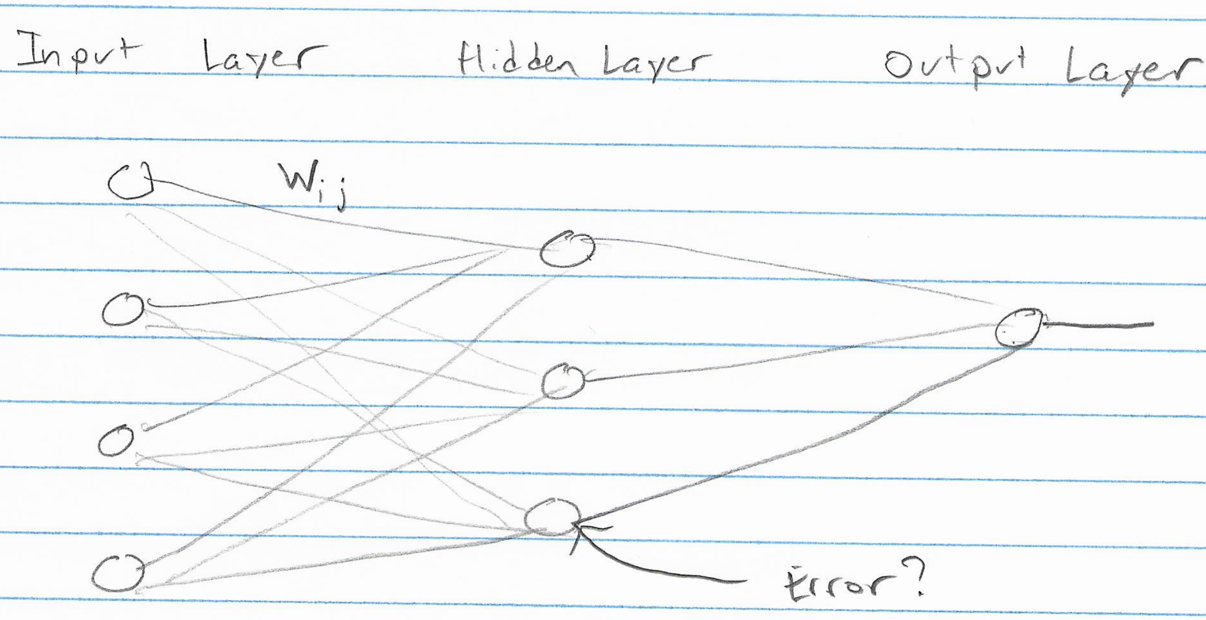
3) Training Neural Networks

4) Network Design and Applications

5) Work on Project (time permitting)

③ Training Neural Networks

<u>Objective</u>: find the weights W that minimize the error or cost function for the training data set



Issue: what is the error for the hidden units?

Backpropogation Solves this problem by breaking the optimization process into two steps:

1) <u>Feed forward</u>: run training data through the network to get the output of each node (computed in the forward direction)

2) <u>Backpropogation</u>: compute the gradient for each node in order to determine which direction to move and the update the weight (computed in the backwards directions)

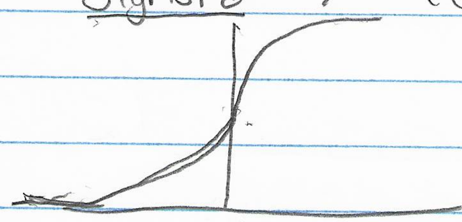
Repeat n times or until convergence criteria is met (early stopping)

Activation function needs to be differientiable

<u>Activation Function</u>: allows the network to exhibit non-linear behavior, i.e squashing the weighted sum of the neuron.

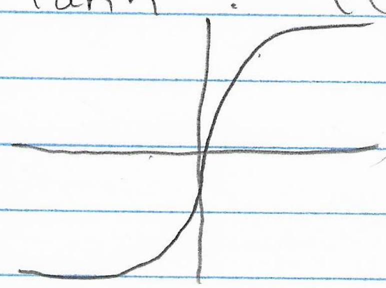
Common activation functions:

<u>Sigmoid</u> : $f(x) = \frac{1}{1+e^{-x}}$ $[0, 1]$



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

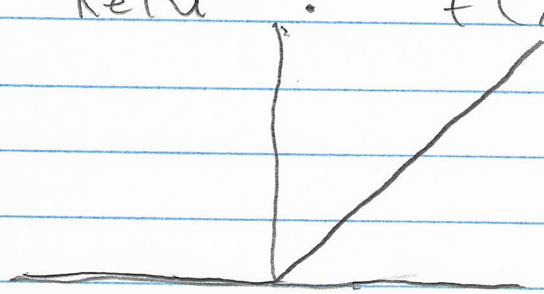
Tanh : $f(x) = 2\sigma(2x) - 1$ ($\sigma(x)$ = sigmoid) $[-1, 1)$



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) <u>Input Layer</u>

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

2) <u>Output Layer</u>

Two cases

Classification

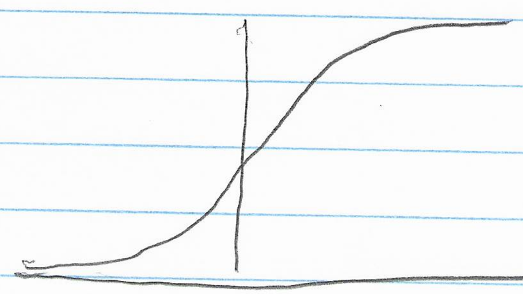
One output node for each class label with a softmax activation function

Prediction

One (or more) output nodes with a linear activation function

Softmax: $f(X_i) = \frac{e^{X_i}}{\sum_{i=1}^{n} e^{X_i}}$

$[0, 1]$

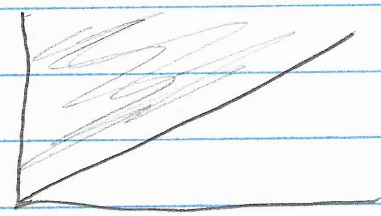


Select highest probability class

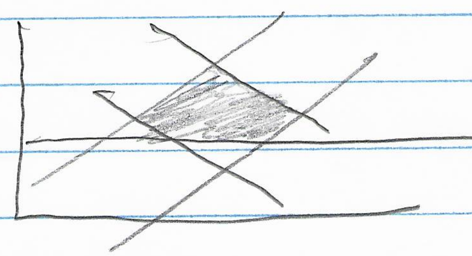
3) Hidden Layer

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

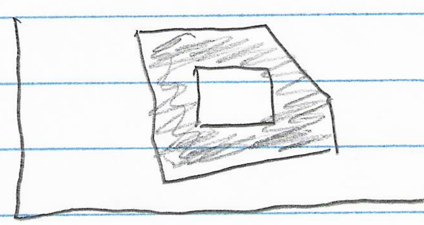
0 layers → linear decision boundry (separating hyperplane)



1 layer ⟷ convex polygon regions (universal approximator)



2 layers → 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
- $2 \log m$
- $\frac{2}{3} m$
- $\sqrt{n \cdot m}$ 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 penalities on the weights in the cost functions (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 α during training (momentum)