Finishing Up with Decision Trees
Neural Nets

Andrea Danyluk
April 14, 2017

Announcements

- Read Pomerleau papers for Monday
- Classifier learning assignment in progress
- Final project
  - Schedule/deliverables posted on the course website

Today's Lecture

- Finishing up with decision trees
- Neural Nets

Dealing with Noise: the problem of overfitting

- Overfitting = the problem of finding meaningless regularity in the data
- A potential problem for all classifier learning algorithms.
- Solution for decision trees:
  - Decide that testing more attributes along a particular path will not improve the predictive accuracy of the decision tree.
  - Called **pruning**.

Reduced Error Pruning

- Pruning consists of replacing a non-leaf node with a leaf and assigning it the most common classification of the training examples associated with that node.
- Reserve a set of examples to be used as a **pruning set**. (Distinct from the training set.)
- After a full tree has been constructed,
  - In postorder fashion, traverse the tree.
  - Replace a non-leaf with a leaf when the error rate of the leaf is no worse than the error rate of the subtree on the pruning set.

Bias in Attribute Selection

- The information gain criterion is biased in favor of tests with many outcomes.
- Consider a medical diagnosis data set, where one of the attributes is social security number. What will happen if you try to split on this attribute?

Gain Ratio Criterion

- Attempt to normalize the apparent gain of attribute X.

  Let split info (X) = $-\Sigma\ (|T_i|/|T|)*\log_2|T_i|/|T|$, where the sum is over the possible values of attribute X.

  gain ratio (X) = gain(T, X) / split info (X)

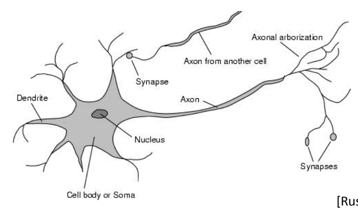
Artificial Neural Networks

- Characterized by:
  - A large number of (simple) neuronlike processing elements
  - A large number of weighted connections between the elements
  - Highly parallel, distributed control
  - An emphasis on learning internal representations automatically
- Theoretically principled training algorithms that aim to minimize an objective function (error)

Why Neural Nets? Cognitive Perspective

- Solving problems under constraints similar to those of the brain may lead to solutions to AI problems that might otherwise be overlooked.
  - Individual neurons operate relatively slowly, but make up for that with massive parallelism
  - Neurons are failure-prone devices, but make up for that with distributed representations
  - Neurons promote approximate matching: a less brittle system

Parts of a Neuron



[Russell & Norvig]

Evidence of learning – plasticity – at the synapses.

How it Works (at a very high level)

- Branching from each neuron are:
  - a number of small fibers -- **dendrites**
  - a single long fiber, the **axon**
- Axon splits and ends in a number of **synapses**
  - these connect the axon to the dendrites of other neurons
- Communication occurs along these paths
- When the electric potential in a neuron rises above a threshold, the neuron activates. It sends the electrical impulse down the axon to the synapses.
- A synapse can either add to the electrical potential or subtract from it.
- The pulse then enters the connected neuron's dendrites, and the process begins again.

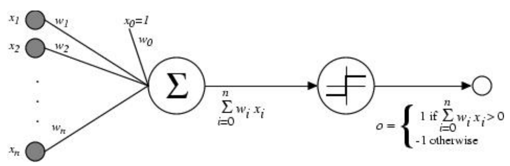
Rich History

- 1943: Warren McCulloch and Walter Pitts propose a model of artificial neurons
- Two views:
  - Neural network as a model of the brain
  - Neural network as a representation of complex functions*

  * Not faithful models of real neural networks

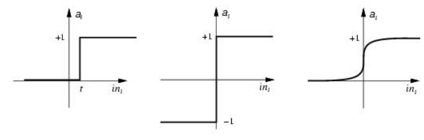
  * Computationally interesting abstractions

An Artificial Neuron: the Perceptron



Note slight difference from the R&N formulation. In theirs, x_0 always has value -1, rather than 1.

Activation Functions



(a) Step function (b) Sign function (c) Sigmoid function

The Perceptron Learning Algorithm

Initialize weights to arbitrary values (perhaps in the range -.05 to .05)

Complete n training epochs:

- For each example e in the training set:
  - Send it through the perceptron.
  - If the output of the perceptron doesn't match the target output:
    - Adjust the weights. If the desired output is positive, increase the weights associated with positive inputs; decrease weights associated with negative inputs. And vice versa.

The Perceptron Update Rule

$$\Delta w_i = r(t - o)x_i$$

Where

- w_i is the i^{th} weight
- x_i is the i^{th} input
- t is the target output
- o is the observed output
- r is the learning rate

Beyond Intuitive Appeal

- Objective: Find weights that minimize error on the training set.
- Define an error function. E.g.

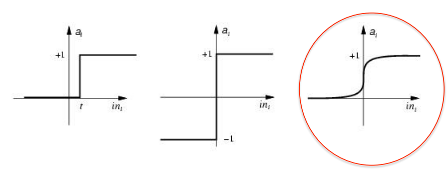
  $$\frac{1}{2}(t-o)^2$$

- Compute the gradient of the error function where

  $$o = w_0x_0+w_1x_1+\ldots w_nx_n$$

- Moving in the direction opposite the gradient pushes us toward a weight vector that minimizes error

Activation Functions



(a) Step function (b) Sign function (c) Sigmoid function