Exploration Seminar 8
Machine Learning

Roy McElmurry

Data

- There are immense volumes of data available today
- It is estimated that Google owns over one million servers. It is unknown what there data storage capacity is.
- Data storage is becoming ridiculously cheap.
- It ten years you could record a babies life in HD for $100.

The Problem

- We have immense amounts of data and lots of questions.
- We would like to solve these problems using the data that we have.

The Idea

- We are given a set of training examples that contain data points as well as answers to the question at hand.
- Using these examples, we construct a model that predicts the answer to the training examples with a high degree of success.
- Use our predictor to make educated guesses on new data points, to which we do not know the answer.
  - We would like our predictor to be as fast as possible.

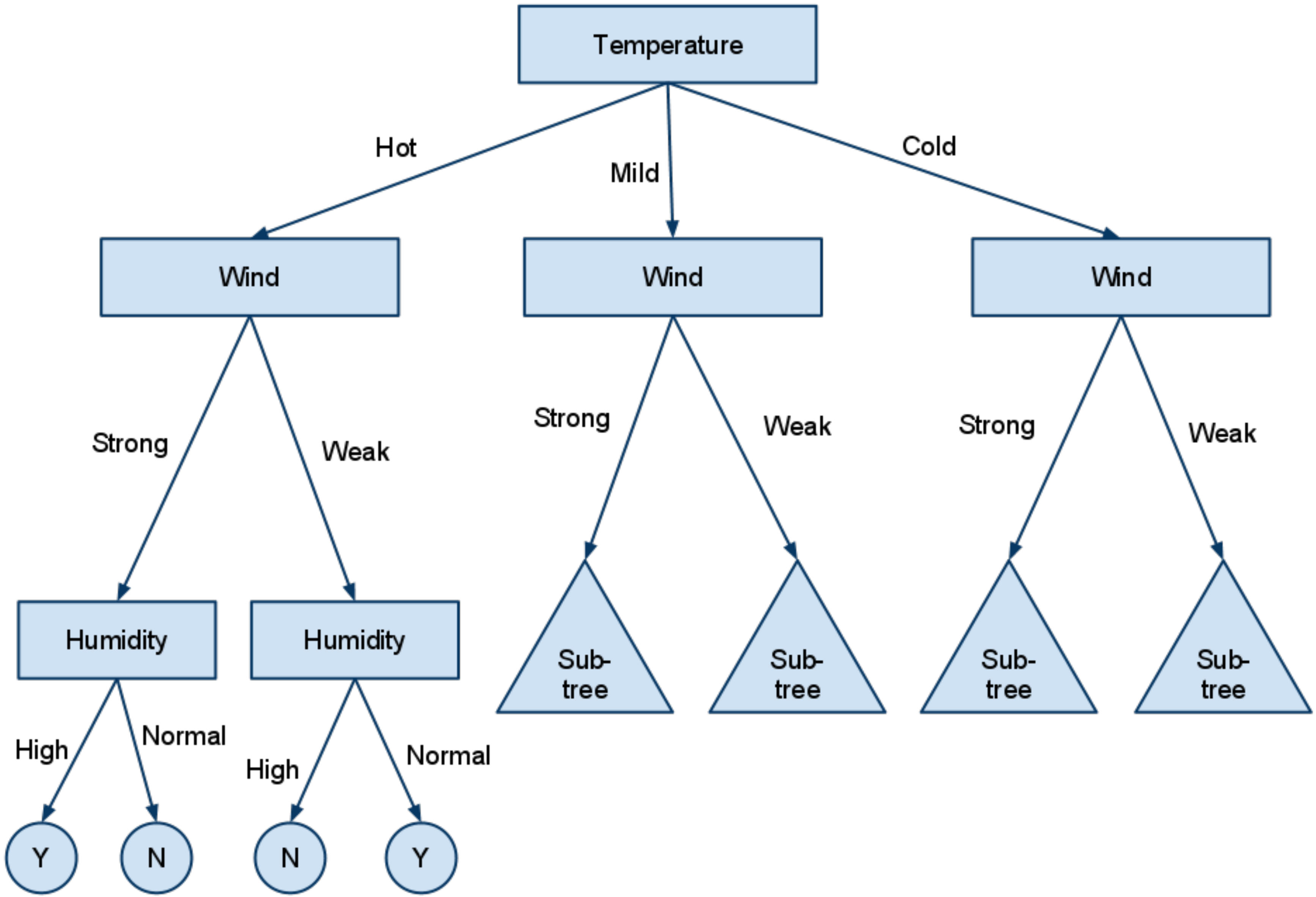
Potential Applications

- Google Ads
- Spam Detection
- Netflix Recommendations
- Traffic Light Timing
- Steering a Car
- Playing Chess

Example Question and Data

Is it a good day to play Ultimate Frisbee?

Day	Outlook	Temperature	Humidity	Wind	Frisbee?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	**Rain**	**Mild**	**Normal**	**Weak**	**???**


Temperature
Hot
Mild
Cold
Wind
Wind
Wind
Strong
Weak
Strong
Weak
Strong
Weak
Humidity
Humidity
Sub-tree
Sub-tree
Sub-tree
Sub-tree
High
Normal
High
Normal
Y
N
N
Y

Decision Trees

- We can use a tree structure to represent the training data.
- Leaf nodes will be the value of the target attribute and other nodes will be attributes.
- Given a new data point we can traverse the tree until we reach a leaf and use the data in that node as our prediction

Algorithm

SimplifiedID3(data, target, availableAttributes):

- Choose an available attribute that best predicts the target attribute and place it at the root of the tree.
  - Remove the chosen attribute from the available ones
  - Split the examples into groups according to the values of the selected attribute
- Recursively create the children using one of the groups of examples and the remaining available attributes

Most Predictive Attribute

- Information Gain is a popular choice for quantifying how predictive an attribute is.

Entropy: $E(S) = \sum_{v \in values_{target}} -P(v)\log(P(v))$

ex) E(S) = -P(+)log(P(+)) -P(-)log(P(-))

= -(5/9)log(5/9) – (4/9)log(4/9) = 0.99

Information Gain: $G(S,A) = E(S) - \sum_{v \in values_A} \frac{|S_v|}{|S|} E(S_v)$

Issues

- Over-fitting: We can build a tree that is too specific and can no longer make general predictions
- Objective Function: We only approximate the objective function based on what was seen, irregular data points will throw us off
- Continuous Data: Decision trees cannot represent continuous data with precision

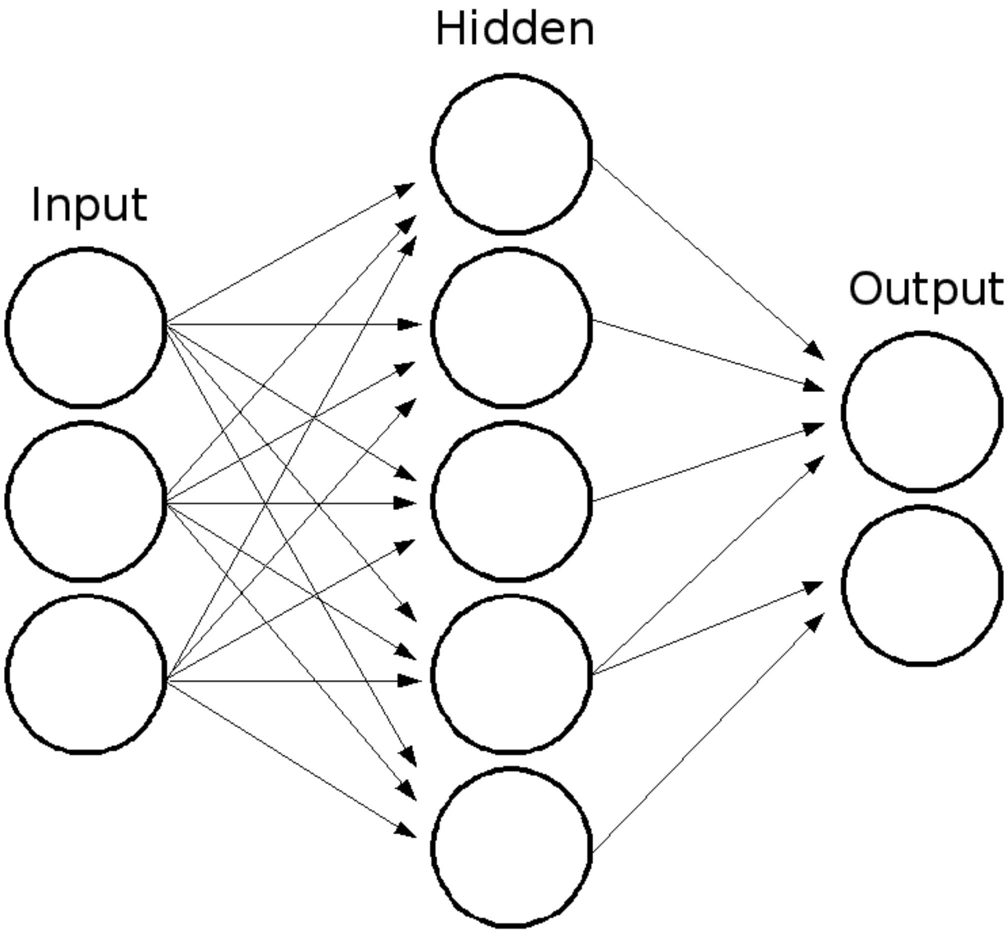
Perceptron

- A perceptron is a function: $f\left(\vec{x}\right) = \left\{ \begin{array}{cc} 1 & \vec{w} \cdot \vec{x} - b > 0 \\ 0 & otherwise \end{array} \right\}$
  - $\vec{w}$ is a weight vector that tells us how important each component of $\vec{x}$ is
  - b is a threshold that determines at what point the perceptron's output changes
  - The output of the perceptron can be thought of as a boolean output, either true of false

||, &&

- These simple functions take two inputs, what is the length of *w*
- Can we find a *w* and *b* such that our perceptron outputs values corresponding to those of the functions

Artificial Neural Network



Algorithm

Simplified Gradient Descent Algorithm:

- Initialize $\vec{w}$ randomly
- Until some sufficient criterion is met
  - Initialize $\Delta\vec{w}$ to 0
  - For each example in the training data of the form $\left(\vec{x}, t\right)$
    - Let y be the output of the neural network given $\vec{x}$
    - $\Delta\vec{w} = \Delta\vec{w} + \eta(t - y)$
  - $\vec{w} = \vec{w} + \Delta\vec{w}$

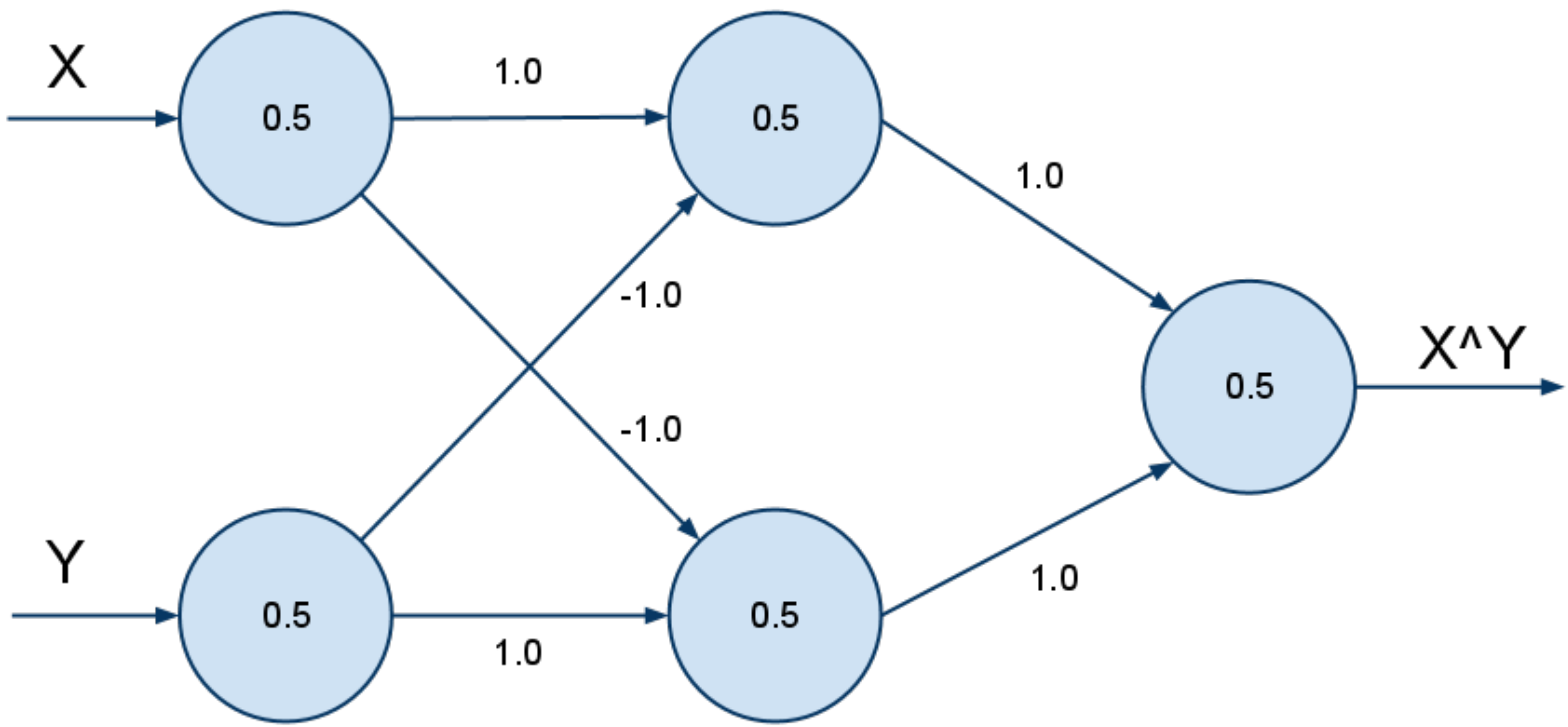
∧

- The *xor* function is the ‘exclusive or’ function and has the following truth table

	True	**False**
True	False	True
False	True	False

- Can we use perceptrons to model this function?
- Can we use neural networks to model this function?

∧



X
Y
0.5
0.5
0.5
0.5
0.5
1.0
-1.0
-1.0
1.0
1.0
1.0
X^Y