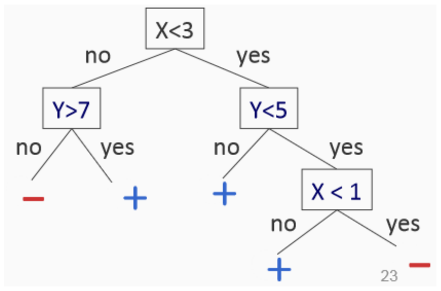
CS 6350 Midterm Review

General Supervised Learning

- Supervised learning, instance spaces, label spaces, concept and hypothesis spaces
  - **Supervised learning:**
    - Given some training examples in the form $\langle x, f(x)\rangle$, with f being unknown
    - Typically, the input x is represented in the *feature space*, for example $x \in \{0,1\}^n$ or $x \in \mathbb{R}^n$
    - For a training example x, the value of $f(x)$ is called its *label*
    - Goal: Find a good approximation for f
  - **Instance spaces:** The set of the examples and features that are going to be looked at
  - **Label spaces:** The total set of possible labels that each instance can have
  - **Concept and Hypothesis spaces:**
    - **Hypothesis space:** All of the possible functions that the learning algorithm is going to be searching over
    - **Concept space:**
- Understanding why we need to restrict hypothesis spaces
  - Choose a hypothesis space that is smaller than the space of all functions. The functions that are choosen are done by either prior knowledge or by guessing. It also needs to be flexible enough to work with the data and not too small that nothing agrees with it.
  - For example, in the case of boolean functions, can do only *simple conjunctions*, pick *m-of-n rules* where you pick a set of n variables, of which at least m need to be true, linear functions, etc.
- General issues in supervised learning: hypothesis spaces, representation (i.e. features), learning algorithms
  - **Hypothesis spaces:** If the hypothesis space is too large, then learning can not be done because there are too many functions to search over. Therefore, to be able to learn, the hypothesis space must be restricted to a smaller subset so that it is possible to learn.
  - **Representation/features:** Need features that represent the data well. Features that aren't present in a lot of the data *or* have too common of a value are not good features.
  - **Learning algorithms:** The right learning algorithm needs to be choosen such that it can learn from the data well and not be excessive in resource usage. Also you want an algorithm that doesn't overfit the data and has a high success rate.

Decision Trees

- What is a decision tree? What can they represent?
  - **Decision tree:** A *hierarchical data structure* that represents data using a divide-and-conquor strategy. It can be used as a hypothesis class for non-parametric classification or regression.
    General Idea: Given a collection of examples, learn a decision tree that represents it.
  - Decision trees are a family of classifiers for instances that are represented by feature vectors (i.e. vectors of attributes)
  - *Nodes* are tests for feature vectors
  - There is one *branch* for every value that the feature can take
  - *Leaves* of the tree specify the class labels
- How to predict with a decision tree
  - At the root node, you're given a test, you then follow the branch for the correct answer for that one node. Decision trees need not be binary!
- Expressivity, counting the number of decision trees
  - **Expressivity:**
- Dealing with continuous features
  - You would have ranges of values that fall in to each node. For example:



- Learning Algorithm: The ID3 algorithm entropy information gain
  - The ID3 Algorithm is based on the *entropy* of each attribute

```
ID3($S$, Attributes, Label):
  1. If all examples have the same label:
         Return a single node tree with the label
  2. Else:
     (a) Create a Root Node for the tree
     (b) A = attribute in Attributes that best classifies $S$
     (c) for each possible value $\nu$ that A can take:
          i. Add a new tree branch corresponding to A = $\nu$
         ii. Let $S\nu$ be the subset of examples in $S$ with A = $\nu$
        iii. If $S\nu \in \varnothing$:
                Add leaf node with the common value of Label in $S$
             Else:
                Below this branch add the subtree ID3($S\nu$,Attributes−{A},Label)
     (d) Return Root Node
```

  - **Entropy and Information Gain:**
    - *Entropy* is the set of examples S with respect to binary classification is

$$Entropy(S) = H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-) \quad \begin{cases} p_+ \text{ is the porportion of positive examples} \\ p_- \text{ is the proportion of negative examples} \end{cases}$$

$$Gain(S, A) = Entropy(S) - \sum_{\nu \in \text{Values}} \frac{|S_\nu|}{|S|} Entropy(S_\nu)$$

$$S_\nu : \quad \text{The subset of examples where the value of attribute } A \text{ is set to value } \nu$$

  - The root attribute that should be choosen is the attribute with the highest information gain
- Overfitting (applicable not just to decision trees) and how to deal with it when training decision trees
  - The learning algorithm fits the noise in the data. Irrelevant attribuytes or noisy examples influence the choice of the hypothesis
  - May lead to poor performance on future examples
  - Decision trees are notorous for overfitting, so the solution to this is to favor simpler (shorter) hypotheses as fewer shorter trees are less likely to fit better by coicidence
- Dealing with missing features
  - Using the most common value oif the attribute in the data
  - Using the most common value of the attribute among all examples with the same output
  - Using fractional counts of all the attributes
  - *Test time*: Use the same method
- When to use decision trees
  - Binary classifications? Small hypothesis spaces?

Nearest Neighbors

- Instance based learning. How to predict? Importance of representation
  - Training examples are vectors $\mathbf{x}_i$ associated with a label y_i
  - *Learning*: Just store all the training examples
  - *Prediction*: For a new example $\mathbf{x}$, find the training example $\mathbf{x}_i$ that is *closest* to $\mathbf{x}$ and predict the label of $\mathbf{x}$ with the label y_i associated with $\mathbf{x}_i$.
    - *Classification*: Every neighbor votes on the label. Predict the most frequent label among the neighbors.
    - *Regression*: Predict the mean value
- Different definitions of distance
  - Euclidean Distance

$$||\mathbf{x_1} - \mathbf{x_2}||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x_{1,i}} - \mathbf{x_{2,i}})^2}$$

  - Manhattan Distance

$$||\mathbf{x_1} - \mathbf{x_2}||_1 = \sum_{i=1}^{n} |\mathbf{x_{1,i}} - \mathbf{x_{2,i}}|$$

  - L_p-Norm

$$||\mathbf{x_1} - \mathbf{x_2}||_p = \left(\sum_{i=1}^{n} |\mathbf{x_{1,i}} - \mathbf{x_{2,i}}|^p \right)^{\frac{1}{p}}$$

- Dealing with symbolic features
  - words
- Choosing k for k-NN
  - words
- Practical aspects: Feature normalization could be important
  - words
- Advantages and disadvantages
  - words
- Voronoi diagrams
- Curse of dimensionality (applicable beyond nearest neighbor algorithms)
  - words

Linear Classifiers

- What are they? Why are they interesting?
  - words
- What can they express? What can they not express?
  - words
- Geometry
  - words
- Feature expansion to predict a broader set of functions
  - words

Mistake Bound Learning

- One way of asking how god is your classifier
  - words
- The general structure of an online learning algorithm
  - words
- Goal: Counting Mistakes. What is a mistake bound algorithm
  - words
- Halving algorithm
  - words
- Perceptron algorithm, geometry of the update, margin, Novikoff's theorem, variants
  - words
- Winnow algorithm, mistake bound, balanced winnow
  - words
- Perceptron vs. Winnow
  - words

Batch Learning

- Assumption that train and test examples are drawn from the same distribution
  - words
- How it is different from mistake bound learning
  - words