**Presentation Pre – Notes: Machine Learning (ML) and its Applications**

Things to include:

* What machine learning is
  + Early examples of machine learning used every day
  + A good definition of machine learning
* Types of learning: Supervised versus Unsupervised
  + Supervised learning: giving the computer a data set with known values, and using that data to construct a model given the data. Two main types: Regression algorithms and categorization algorithms (continuous and discreet)
    - Aka: given a function m, where x is the input and y is the output, each time an experience E happens, for every time function m runs successively it performs at a rate P higher than before.
    - Example: a linear regression model. A spam filter.
    - Good pro/con list of supervised from:
      * Pros:
        + Can learn complex patterns
        + Good performance
      * Cons:
        + Requires many examples out output for examples
    - 3 types of functions:
      * Loss function: “Measures system performance, depends on output type/goal, and generalization error (over-fitting)
      * Input function: “Divides into training and test sets, uses train to learn h, evaluate on test.”
      * Output function: “Learned model, type depends on learning algorithm.”
    - Supervised types:
      * Nearest Neighbors
        + Motivation: “This document has the same label as the most similar document I have seen.”
        + Example: “This document is about baseball. The last baseball article I saw was about sports. This is about sports.”
        + Approach: “Save every example in the training set. For a test example: a) find the closest training example, b) apply the label from this training example.”
        + Implementation choices:

Similarity function: Euclidian distance

How to choose K? Small K is fast, finds single closest example, while large K is slower, but smooths outliers

How do we know that we have learned? Training error: select parameters (k) that minimize error on training examples?

Some work on estimating true generalization error

* + - * + Bias/variance tradeoff

As we increase K: Bias- increase towards most popular labels. Variance decreases

In practice: Select K using development data, reflective of actual performance.

* + - * + Summary:

Pros: Easy to implement, understand output, complex functions

Cons: need to store every observed example, choosing similarity metric, and slow classification

Useful extensions: Learn similarity metrics (Large Margin Nearest

* + - * Decision Tree Learning
        + Motivation: “I can decide about a document by incrementally considering its properties.”
        + Example: “This document says the word “baseball.” So it’s about sports. If it did not, I would next check if it says “finance,” then…
        + Approach:

Construct a “tree of decisions” to follow, where a leaf applies a label to the document

* + - * + Building a decision tree:

ID3 Algorithm: Greedily add the most discriminating features

ID3 (Examples, target\_attribute, attributes): If all target\_attribute examples have the same label, apply label. Else: A = attribute that best classifies examples. Add branches for each value of attribute. Create subtree from: ID3(examples, A, attributes - A )

* + - * + Choices

Complexity of tree: how many levels?

Pruning: reduces over-fitting (C45 algorithm)

Selecting informative choices: “Which features to select at each point? “attribute that best classifies examples. Information entropy common choice

* + - * + Summary:

Pros: Very easy to understand (“white box,”), good for identifying a few critical features— and FAST.

Cons: VERY slow to train, over-fitting, limited powers of representations (XOR), optimal trees NP-complete

Useful extensions: Real valued data, Decision stumps for boosting, Random forests (ensemble approach)

* + - * Artificial Neural Networks
        + Motivation

Extract linear combinations from input

Output nonlinear functions from these combinations

Multiple functions performed in parallel

Based on neural networks in the brain (the result is a LOT of hype)

* + - * + Approach

Construct a graph of neural connections

Define input and output nodes

Learn hidden internal nodes

* + - * + Neuron Connections

Activation function for nodes (often chosen as the sigmoid).

Source for algorithm’s name, since neurons ALSO have an activation threshold (oh that’s all it is really)

* + - * + Learning Model Parameters

Minimize loss function (regression: squared error)

Back –propagation

Gradient descent on minimizing R

Sweep forward and backward over the network. Only need to compute local variables. Similar to EM learning

Problems: local minima, over-fitting, initialization

* + - * + Design Decisions

Learning Algorithm: issues: local minima, over-fitting, training time

Network structure: Number of layers, nodes per layer

Loss function: Regression (squared error), Classification (cross-entropy)

Network type: (Different functions in a network)

* + - * + Summary

Pros: Can learn non-linear functions, multiple outputs at once

Cons: Not easily interpretable (difficult to influence)

Extensions: MANY! Whole conferences and journals on NNs. Applications to supervised, unsupervised learning. Deep belief networks

* + - * Perceptron
        + Overview

One of the oldest and (still) most effective learning algorithms

Known as a single layer neural network

* + - * + Issues

Learning guarantee: If a separating hyperplane exists, will find separator with finite number of examples

Problems: Finite can still be large (slow convergence); Many correct hyperplanes (but which one is “best?”); sometimes separators don’t even exist (outliers, noisy labels, etc…. will never converge

* + - * Support Vector Machines (SVM Learning)
        + Motivation

Similar to perceptron

Which one is the best separator: (Hyperplane with maxiumum margin; margin- distance between examples and separator)

* + - * + Approach:

Define optimization problem, given the training data

Learn the best separator

“Support vectors”: problem can be formulated a s a combination of input vectors

Convex QP problem: Many efficient algorithms to solve

Formulaiton allows outliers (tolerance set through parameter)

* + - * + Summary:

Pros: Very good performance, efficient learning

Cons: Hard to interpret results, scaling to large datasets

Extensions: Transductive/Semi-Supervised; Regression; One-class; Kernels for non-linear learning

* + - * Probabilistic models (Gideon
    - Unsupervised learning: finding data patterns given data sets of which not all the variables are known. Finding “hidden structure” among things that you don’t know what to search for.
      * Examples: categorizing news, categorizing sets of genes, categorizing types of users, natural language processing… cocktail party problem
      * Two types: clustering algorithms, signal separation
      * Unsupervised (clustering) types:
        + K-means

Motivation:

Examples are points in high dimensional spaces

Examples cluster together

Approach:

Find a set of K clusters that best describe the data

Each cluster defined by a centroid

Examples belong to cluster with the closest centroid

Learning the clusters:

Given K, find the best K clusters given the data

Assume Euclidean distance as similarity metric (d(xi, xi) = ||X- - xi||^2

Minimize the cluster scatter

Expectation-Maximization Algorithm:

Expectation: Assign instances to closest clusters

Maximization: compute mean of clusters based on assigned documents

Recall:

Minimizes the objective (reduces the cluster scatter

Guarantee: will converge to an optimal value

Note: optimal not necessarily global

Bias/Variance of model:

Model parameter: K (number of clusters)

Bias- smaller K biases towards popular clusters

Large K, smaller bias

Variance- smaller K means fewer clusters

Larger K, high variance

Ideally, we’d like to know the exact number of clusters

Summary:

Pros: simple, easy to learn

Cons: Clusters are the same geometric size )we just have a mean for each cluster)

* + - * + Gaussian Mixture Models (an extension of K means)

Assume each cluster is a Gaussian distribution

Mean: center of the cluster

Variance: geometric size of the cluster

Soft clustering in learning (probability of instance from each cluster)

Expectation: compute the responsibility of clusters for examples

Maximization: Compute new mean/variances based on example assignments

* + Other types of learning (combining supervised and unsupervised)
    - Semi-supervised
      * Some labeled examples, many unlabeled examples
    - Partially-supervised
      * Incomplete information about labels
    - Semi-supervised clustering
      * Discover groups with some guidance
  + Start with a linear regression model
    - Given a set of input and output data on a 2 dimensional graph (one input, one output), can we make a function that will approximate
    - We could do least-squares: that’s done by minimizing the square of the distance from the farthest point away from the set of scatter data. (Linear Algebra)
    - We could use Local Regression (LOESS), where we take small parts of the data and break it into smaller linear models
* Types of learning: output
  + Classification
    - Binary, multi-class, multi-label, hierarchical, etc.
    - Loss: accuracy
  + Ranking
    - Order examples by preference
      * Rank results of web searches
      * Loss: swapped pairs
  + Regression
    - Real-valued output
      * Predict the price of tomorrow’s stock price
      * Loss: squared loss
  + Structured prediction
    - Sequences, trees, segmentation
      * Find faces in an image,
      * Loss: Precision/recall of faces

Types of inductive biases [[edit](http://en.wikipedia.org/w/index.php?title=Inductive_bias&action=edit&section=1)]

The following is a list of common inductive biases in machine learning algorithms.

* **Maximum**[**conditional independence**](http://en.wikipedia.org/wiki/Conditional_independence): if the hypothesis can be cast in a [Bayesian](http://en.wikipedia.org/wiki/Bayesian_inference) framework, try to maximize conditional independence. This is the bias used in the [Naive Bayes classifier](http://en.wikipedia.org/wiki/Naive_Bayes_classifier).
* **Minimum**[**cross-validation**](http://en.wikipedia.org/wiki/Cross-validation_(statistics))**error**: when trying to choose among hypotheses, select the hypothesis with the lowest cross-validation error. Although cross-validation may seem to be free of bias, the [“no free lunch”](http://en.wikipedia.org/wiki/No_free_lunch_in_search_and_optimization) theorems show that cross-validation must be biased.
* **Maximum margin**: when drawing a boundary between two classes, attempt to maximize the width of the boundary. This is the bias used in [Support Vector Machines](http://en.wikipedia.org/wiki/Support_Vector_Machines). The assumption is that distinct classes tend to be separated by wide boundaries.
* [**Minimum description length**](http://en.wikipedia.org/wiki/Minimum_description_length): when forming a hypothesis, attempt to minimize the length of the description of the hypothesis. The assumption is that simpler hypotheses are more likely to be true. See [Occam's razor](http://en.wikipedia.org/wiki/Occam%27s_razor).
* **Minimum features**: unless there is good evidence that a [feature](http://en.wikipedia.org/wiki/Feature_space) is useful, it should be deleted. This is the assumption behind [feature selection](http://en.wikipedia.org/wiki/Feature_selection) algorithms.
* **Nearest neighbors**: assume that most of the cases in a small neighborhood in [feature space](http://en.wikipedia.org/wiki/Feature_space) belong to the same class. Given a case for which the class is unknown, guess that it belongs to the same class as the majority in its immediate neighborhood. This is the bias used in the [k-nearest neighbor algorithm](http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm). The assumption is that cases that are near each other tend to belong to the same class.