TIM 245 Lecture 13 (5/17/17)

Agenda

1) Neural Network Applications

2) Simple Classification Models:
Naive Bayes, Decision Trees,
K-Nearest Neighbors

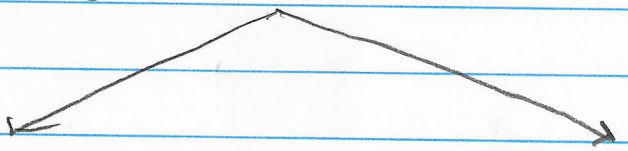
3) Ensemble Learning: bagging and boosting

4) Comments on Classification Models

5) Return Graded Homework 2

(1) Neural Network Applications

Two different applications for neural networks

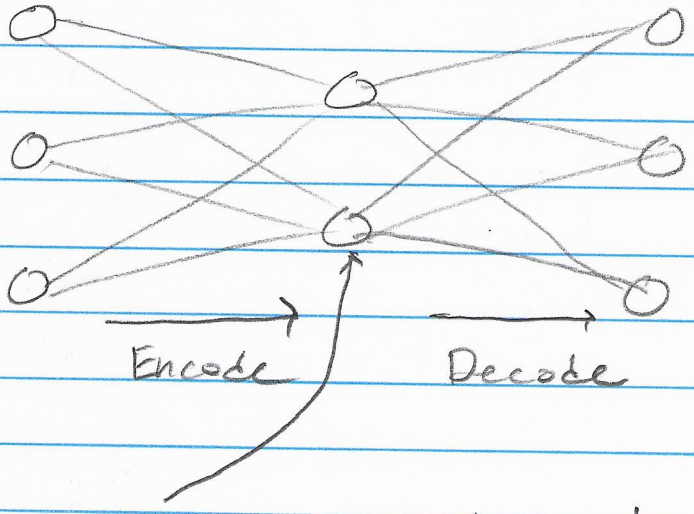


Classification / Prediction

Dimensionality Reduction / Feature Engineering

Recent interest in neural networks is primarily due to their ability to perform non-linear feature engineering

Autoencoder: set output layer equal to the input layer

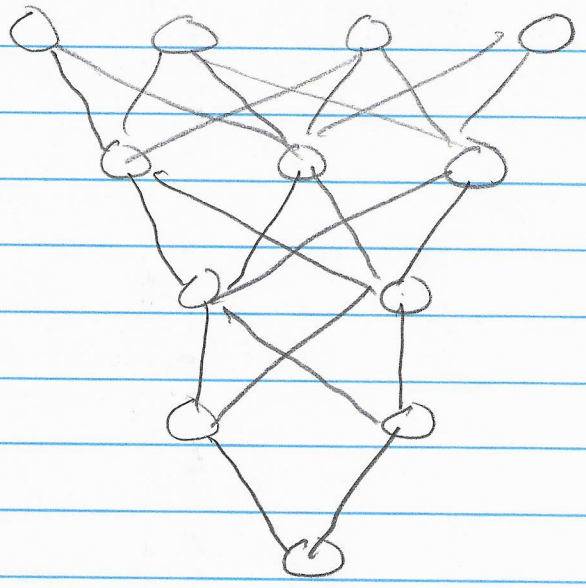


New Features A_1' A_2'

use the features as inputs to prediction, classification, clustering, Association

Deep Learning: using more than three hidden Layers to represent higher order features

Example: Facial Recognition



Big Challenge: How do you interpret the new attributes?

Example: The Cat Neuron

(2) Simple Classification Models

Naive Bayes: Statistical (probabilistic) classifier with the "naive" assumption of independence

$$P(C_K \mid X_i) = \frac{P(X_i \mid C_K)\, P(C_K)}{P(X_i)}$$

Posterior ← $P(C_K \mid X_i)$; Likelihood ← $P(X_i \mid C_K)$; Class prior ← $P(C_K)$; Data Prior ← $P(X_i)$

If each feature independently contributes to the class probability

$$P(X_i \mid C_K) = \prod_{j=1}^{m} P(X_{ij} \mid C_K)$$

Find the highest probability class

$$C_{map} = \underset{K}{\text{Arg max}}\; P(C_K) \prod_{j=1}^{m} P(X_{ij} \mid C_K)$$

Advantages

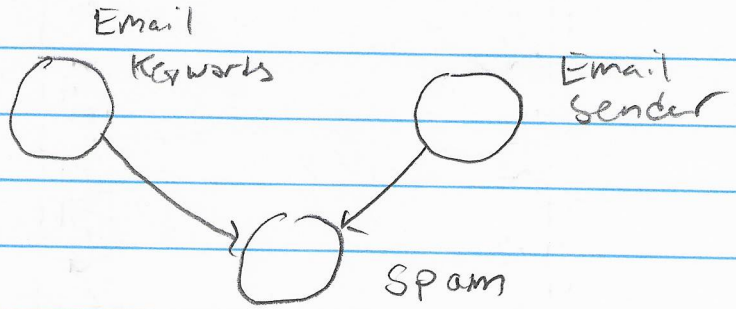
- Simple and easy to implement
- Fast
- Supports on-line learning
- Handles lots of attributs (thousands)
- Robust to noise

Key issue: independence assumption breaks down for more complex problems

Example: Spam detection and emails from your doctor

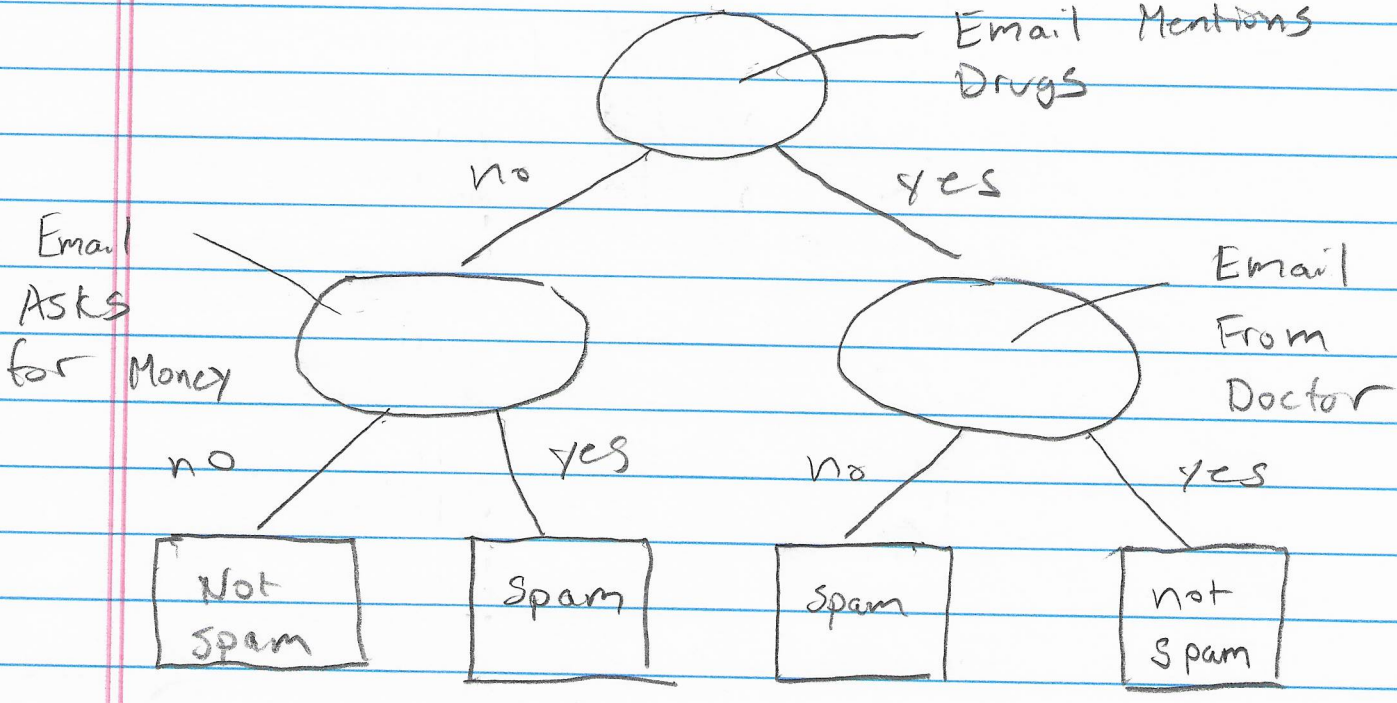
Can be extended by modeling the conditional dependencies

Bayesian Network



Complicated and requires either domain knowledge or lots of data to work well

Decision Trees: Create a series of sequential decisions for classifying an instance



Greedy Algorithm

1) Compute information gain for splitting on each attribute

2) Split on the attribute with the highest information gain

3) Repeat until each leaf node is pure

We want to prune the tree to avoid overfitting.

Two approaches to pruning decision trees

1) Pre-Pruning (Early Stopping):

Stop when the split is below a certain threshold for information gain

2) Post-Pruning (cost complexity)
Model the cost of a tree as a function of error and number of nodes. Find the the tree that minimizes cost. (bottom-up)

Advantages
- Fast
- Easy to interpret
- Automatic Feature Selection. Robust to irrelevant features.

K-Nearest Neighbors: Classify based on similarity to previously observed instances (training data)

Two decisions:

1) What similarity or distance metric to use.

2) How many previously observed instances to use in making the classification (K)

Common Distance Metrics

Euclidean Distance: $\sqrt{\sum_{j=1}^{m} (X_{1j} - X_{2j})^2}$

Chebychev Distance = $\max_j |X_{1j} - X_{2j}|$

Manhatten Distance = $\sum_{j=1}^{m} |X_{1j} - X_{2j}|$

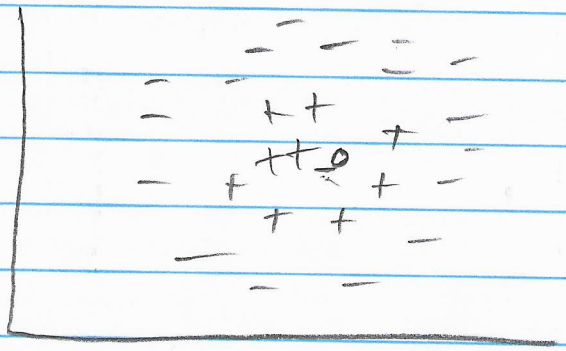
The best value for k depends on the data set

Too Small (K=1) = Sensitive to Noise
Too Large (K=n) = underfitting

Start small and gradually increase until performance doesn't improve

Advantages

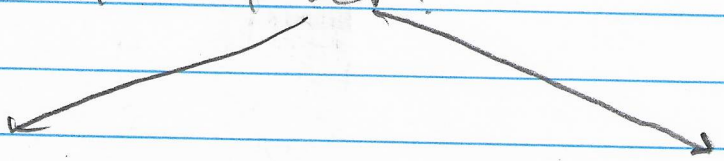
- simple
- Intuitive and easy to explain
- Non-linear



- Don't need to train a mode
- Easy to do online learning

③ Ensemble Models

How do we combine multiple models to improve the predictive power?



Bagging

Train multiple models on different random samples of the dataset. Combine results using majority vote.
(Decreases Variance)

Boosting

Incrementally train multiple models on training data that emphasizes the errors of the previous model. Combine results using a majority vote.
(Decreases Bias)

Works best with trees and decision stumps (tree with 1 split)

Bagging → Random Forest (trees)

Boosting → XG Boost (Stumps)

(4) General Comments on Classification Models

1) Work in well defined iterations

   a) Define the objective of the model given the problem under consideration

   b) Create a set of initial set of models and evalvale with respect to the objective

   c) Determine the best direction for improvement (gradient)

   d) Repeat until objective is satisified or models do not improve

2) It is difficult to know what learning algorithm will produce the best model. Make sure you create and evalvale a range of different models.

3) use Cross validation to understand how the model will perform on new data

K-Fold Cross Validation

a) Partition the data into K disjoint subsets

b) Train on k-1 partitions and Test on the remaining one. Rotate partitions so all sets are tested exactly one time.

c) Average the evaluation metrics, accuracy, f-measure

Use the entire dataset to train the model.

4) Always check your data for potential issues:

- Class Imbalance
- Leakage
- Noisy or incorrect labeling