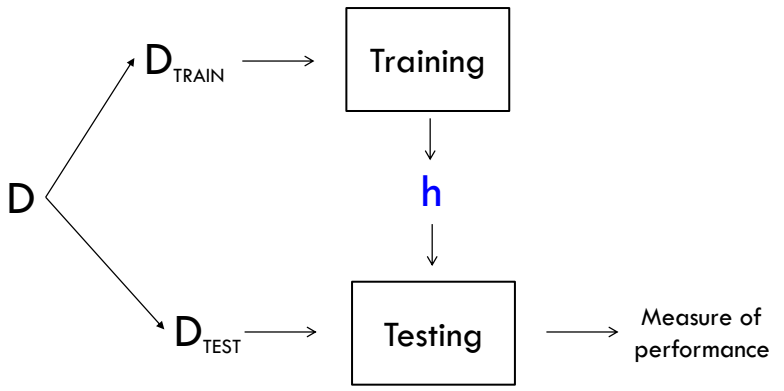
SL: PUTTING IT ALL TOGETHER

A Bird's Eye View



Outline

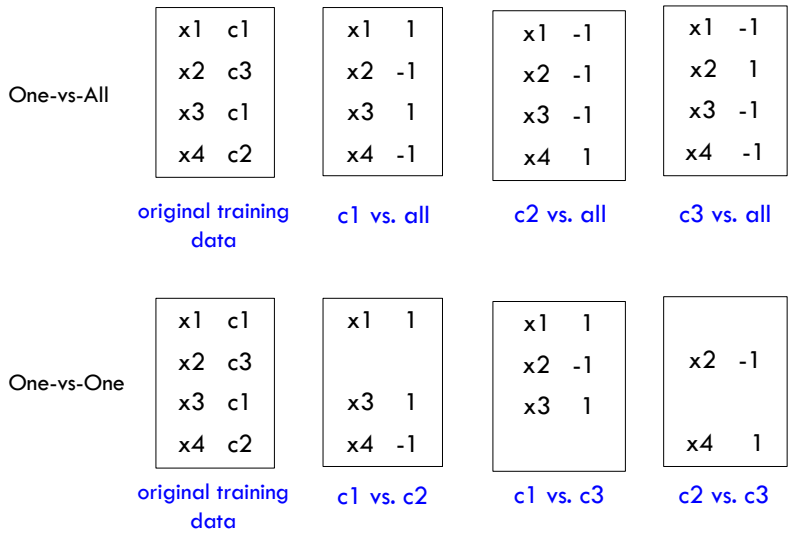
- Step 1: Formulating the problem
- Step 2: Exploring the data
- Step 3: Feature Selection
- Step 4: Pick your Classifier
- Step 5: Training
- Step 6: Testing

The first 5 steps are not necessarily done in a strict linear progression

Step 1: Formulate the problem

- What quantity are you predicting?
  - real-valued, categorical, structure?
  - Changing over time?
  - Classification
    - Binary classification? Multi-class classification?
    - Singly-labeled? Multi-labeled?
    - For multi-labeled classification tasks, how correlated are the labels?
- What data do you have?
  - Where to get labeled data? (Amazon mechanical turk)
  - How much labeled data?
  - What is the quality of the labeled data?
  - Are the labels learnable given the data?
  - Is the distribution of labels in the data skewed/imbalanced?

Reducing multi-class to binary task



Outline

- Step 1: Formulating the problem
- Step 2: Exploring the data
- Step 3: Feature Selection
- Step 4: Pick your Classifier
- Step 5: Training
- Step 6: Testing

Step 2: Exploratory Data Analysis

- Look at the data. It's surprising how often we forget to actually do this!
- Exploratory Data Analysis (EDA) is a statistical practice
  - Box plots, histograms, scatter plots, mean, mode, deviations
  - Can guide the modeling process by
    - give you insight into the data
    - help (in)validate your assumptions
    - detect outliers
    - Suggest further avenues of research

Outline

- Step 1: Formulating the problem
- Step 2: Exploring the data
- Step 3: Feature Selection
- Step 4: Pick your Classifier
- Step 5: Training
- Step 6: Testing

Step 3: Feature Selection

- What features should I use?
  - Dimensionality reduction if exist time/space constraints
  - Reduce noise in the data (irrelevant or redundant features)
- Dimensionality reduction
  - Principal component analysis (PCA)
  - Singular value decomposition (SVD)
  - Canonical correlation analysis (CCA)
- Regularization
  - Use every feature but penalize classifiers that are overly complex

$$\text{Error}(w) = \sum_{i=1}^{N}(y_i - h_w(x_i)) + \lambda||w||^2$$

encourages sparse weight vectors

Other tricks

- Scale input features
- Transform features
  - e.g., take log
- Higher-order features
  - e.g., product of features
- Again, EDA can help guide this process

Outline

- Step 1: Formulating the problem
- Step 2: Exploring the data
- Step 3: Feature Selection
- Step 4: Pick your Classifier
- Step 5: Training
- Step 6: Testing

Step 4: Pick Your Classifier

- Graphical models
  - Naïve Bayes classifiers
  - Bayesian networks
  - Dynamic Bayesian networks
- Decision trees
  - Random forests (many decision trees)
- Neural Networks
  - Perceptrons
  - Artificial neural networks
  - Deep belief nets
- Max margin classifiers
  - Support vector machines
- Regression analysis
  - Logistic regression
  - Linear regression

Pick Your Classifier

- Is there a classifier that is optimal for all classification problems?
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy/skewed is the training data?
  - How stable is the problem over time?
  - Is it a singly-labeled or multi-labeled problem? Are the labels correlated?

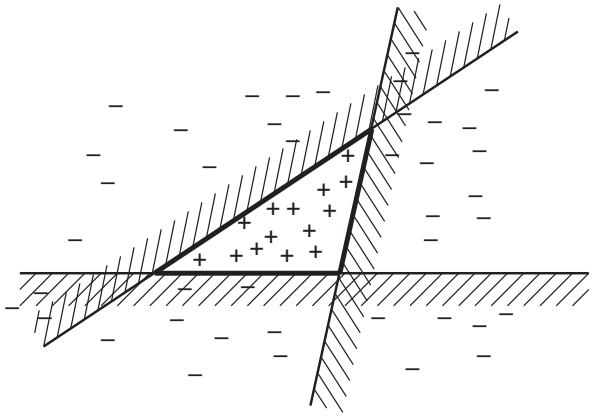
How much data?

- Learning theory (PAC learning)
  - Gives theoretical bounds on how much training data you need for a given accuracy (AIMA 18.5)
- Very Little
  - There are empirical results that naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
  - The interesting theoretical answer is to explore semi-supervised training methods: Bootstrapping, EM over unlabeled documents, …
  - The practical answer is to get more labeled data as soon as you can
- A reasonable amount of data
  - Start with SVMs
- A lot of data?
  - expensive methods like SVMs (train time) or kNN (test time) are quite impractical
  - Naïve Bayes! - with lots of data, simple methods work well
  - And, of course, neural networks

Choose Many Classifiers!

- Ensemble - A group of items viewed as a whole rather than individually
- An ensemble of classifiers – A group of classifiers whose predictions are combined to produce one final prediction
- Benefits
  - Harder to make a wrong prediction
  - More expressive hypothesis

Ensemble of linear classifiers



- More expressive than any one linear classifier by itself

Ensemble Schemes

- Multi-expert combination methods
  - Global - All classifiers generate a prediction and all predictions are used in some way
    - e.g. weighting, voting, averaging
  - Local – A gating model chooses one (or very few) of the classifiers responsible for generating the prediction for a specific input
    - e.g. mixture of experts
- Multi-stage combination
  - Classifiers are trained with, or tested on, only the instances where the previous classifiers are not accurate enough
    - e.g. cascading

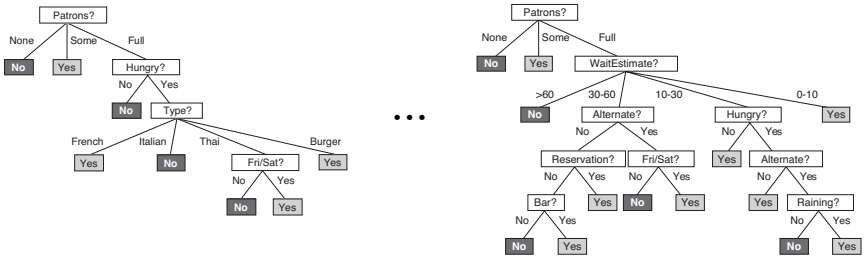
Boosting

- A well-known boosting algorithm is AdaBoost short for "Adaptive Boosting" (Freund and Schapire 1995)
- Boosting is one of the most common forms of constructing an ensemble of classifiers
  - Learn a series of weak classifiers, i.e. classifiers whose performance is slightly better than random chance
  - Weight each weak classifier to create a final strong classifier
  - Often the weight for each classifier is proportional to its accuracy

Bagging

- Short for "Bootstrap aggregating"
- Random Forests (Breimen, 2001)
  - Bagged decision trees
- Given training set D
  - Generate M new training sets D_i where $|D_i| < |D|$ by sampling from D with replacement
  - This is a statistical technique known as bootstrapping
  - Train a classifier on each of the M new training sets
  - Combine output of M classifiers using averaging or voting

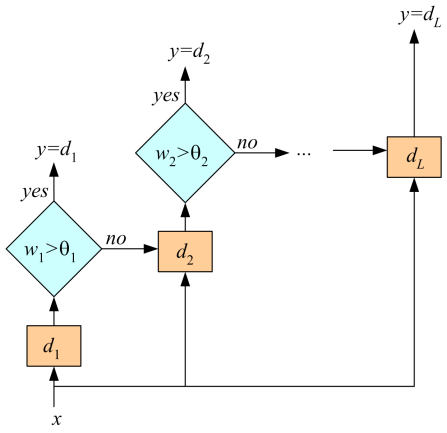
Bagged Decision Trees: Random Forest



- Combine the prediction of each decision tree using majority vote

Cascading classifiers

- Order classifiers by complexity, e.g. representational complexity
- Use i^{th} classifier d_i only if previous classifiers are not confident
- Good with high precision/low recall classifiers



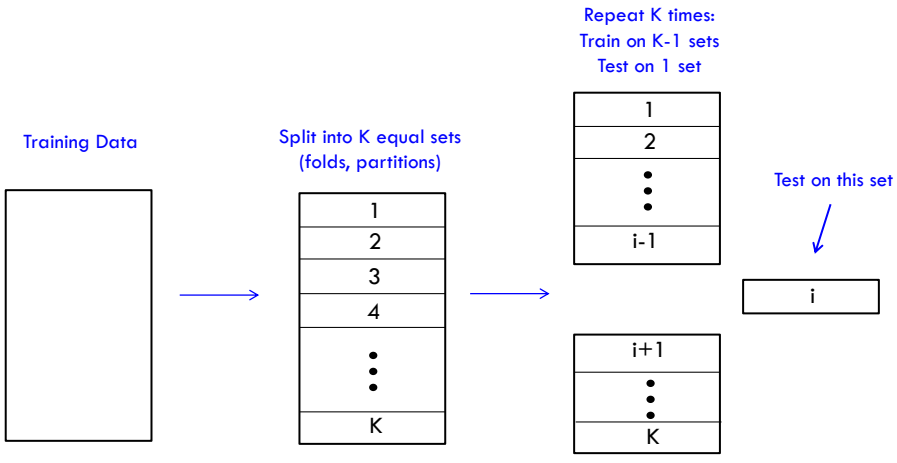
Outline

- Step 1: Formulating the problem
- Step 2: Exploring the data
- Step 3: Feature Selection
- Step 4: Pick your Classifier
- Step 5: Training
- Step 6: Testing

Step 5: Training

- Many parameters to choose
  - Step size, learning rates
  - Number of layers, activation function
  - Kernel function
- Trial-and-Error
  - {0.001, 0.01, 0.1, 0.5, 1.0}
  - Evaluate on a held out set of data (not the test set)

K-Fold Cross Validation



Step 6: Testing

- We have a final hypothesis
- We now use our hypothesis to predict on new (unseen) examples from the test set.
  - There's no going back and tweaking the classifier based on its test set performance!
- Where do these new unseen examples come from?
  - External source
  - Set aside from training data

Binary Classification: Measures of Performance

- The contingency table is given by:

	y = 1	y = 0
h = 1	TP	FP
h = 0	FN	TN

  - TP is the number of *true positives*
  - FP is the number of *false positives*
  - FN is the number of *false negatives*
  - TN is the number of *true negatives*

Binary Classification: Measures of Performance

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F}_1\text{-score} = 2 \cdot \frac{\text{Prec} \cdot \text{Recall}}{\text{Prec} + \text{Recall}}$$

	y = 1	y = 0
h = 1	TP	FP
h = 0	FN	TN

Contingency Table

Binary Classification: Measures of Performance

$$\text{Accuracy} = \frac{7+8}{7+8+2+3} = \frac{15}{20} = .75$$

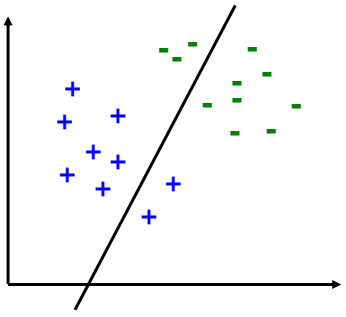
$$\text{Precision} = \frac{7}{7+3} = .70$$

$$\text{Recall} = \frac{7}{7+2} = .78$$

$$\text{F}_1\text{-score} = 2\left(\frac{.70 \cdot .78}{.70 + .78}\right) = 2\left(\frac{.546}{1.48}\right) = .74$$

	y = 1	y = 0
h = 1	7	3
h = 0	2	8

Contingency Table



Multi-class Classification: Measures of performance

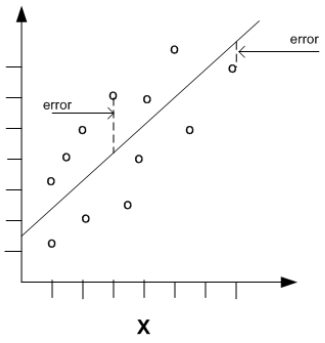
- Evaluate each label separately using a "one-vs-all" approach
  - Macro-averaging
    - Compute the measure (precision, recall, F_1) for each class
    - Average across all C classes
    - Gives equal weight to all classes
  - Micro-averaging
    - Pool the TP, FP, FN, TN for all C classes
    - Compute the measure (precision, recall, F1)
    - Weighted towards performance of most likely class

	$y_c = 1$	$y_c = 0$
$h_c = 1$	TP_c	FP_c
$h_c = 0$	FN_c	TN_c

Contingency Table

Regression: Measures of performance

- Mean-squared error
- Root mean-squared error
- Mean absolute error
- Mean absolute percentage
- …



Guiding Principles

- Start simple and iterate
- Justify each step
- Reproducibility
  - README files
  - Version control

Summary

- Step 1: Formulating the problem
- Step 2: Exploring the data
- Step 3: Feature Selection
- Step 4: Pick your Classifier
- Step 5: Training
- Step 6: Testing