Machine Learning and Classification Part II

Julie Dickerson

Ensembles of Decision Trees

- Construct a multitude of decision trees during training
- Output class that is the mode or mean of the individual trees
- Goal: correct for habit of overfitting to the data and reduce the variance
- Called “random forest”
- Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **20** (8): 832–844.
- Breiman, Leo (2001). "Random Forests". *Machine Learning*. ***45*** *(1): 5–32.*

Bootstrap Aggregating (Bagging)

- Given a training set $X = x_1, \ldots, x_n$ with responses $Y = y_1, \ldots, y_n$, bagging repeatedly (B times) selects random samples with replacement of the training set and fits trees to these samples:

- For $b = 1, \ldots, B$:
  - Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
  - Train a decision or regression tree f_b on X_b, Y_b.

Predicting

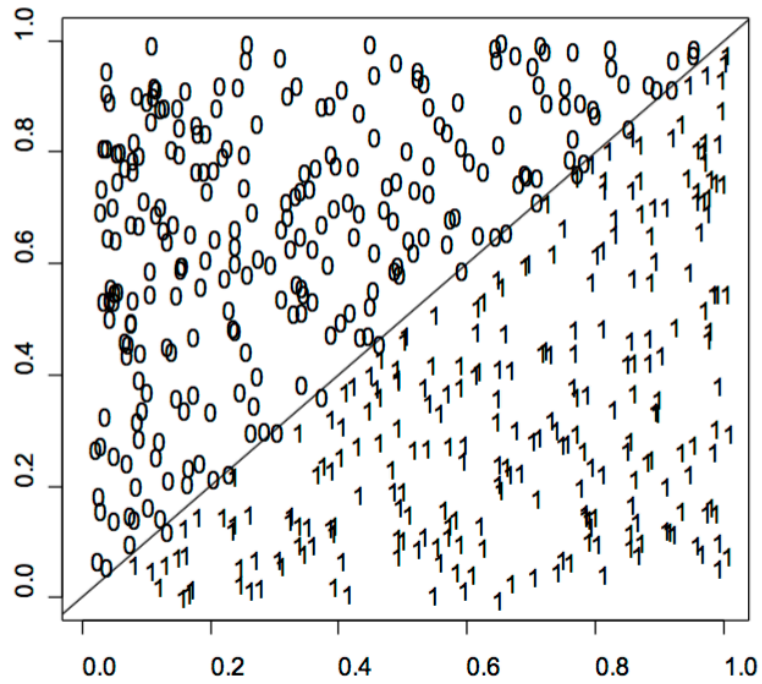
- Predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x')$$

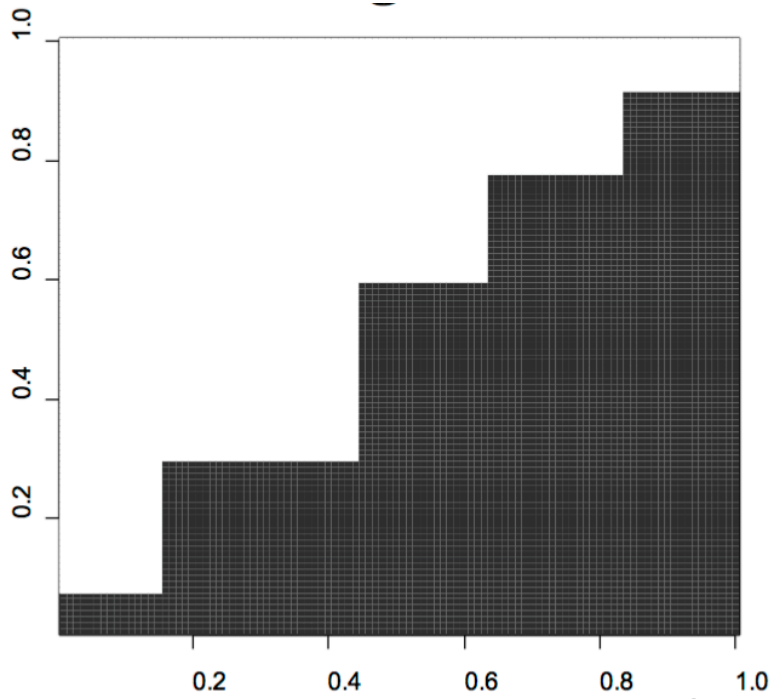
- Or by voting
- B is a parameter that can be on the order of hundreds/thousands
- Optimal number of trees B can be found
  - using cross-validation,
  - observing the **out-of-bag** error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

Classification Problem

Linear Classification



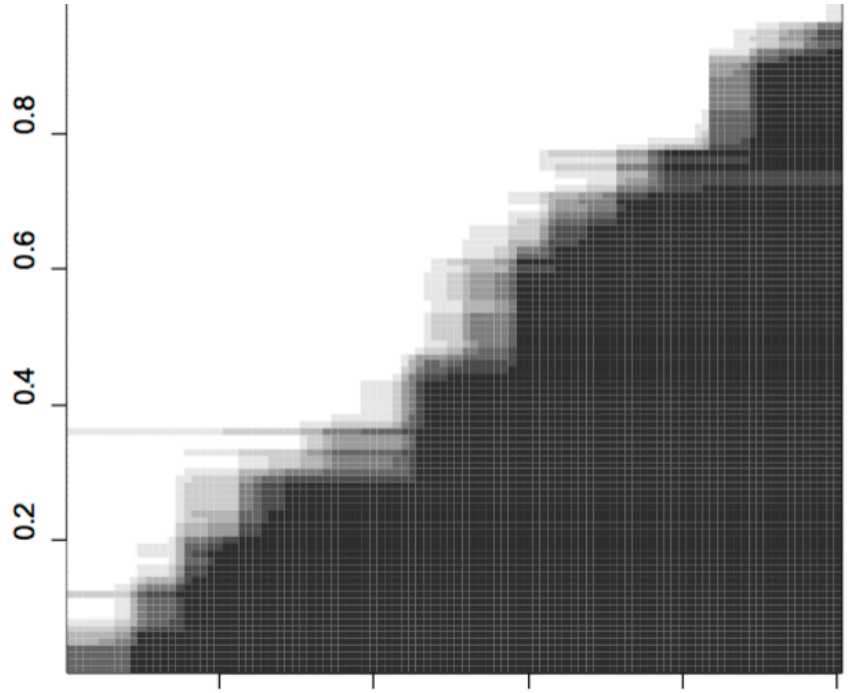
Estimate of Single Tree



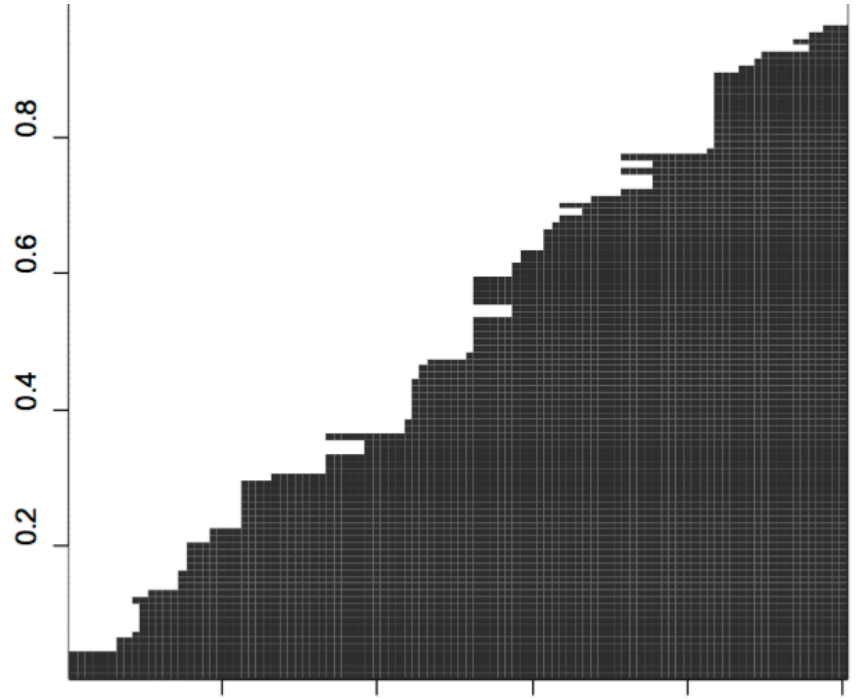
Random Forests for Regression and Classification, Adele Cutler Utah State University September 2010, Ovronnaz Switzerland, http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf

Multiple Trees

25 Averaged Trees



25 Voted Trees



Random Forests for Regression and Classification, Adele Cutler Utah State University September 2010, Ovronnaz Switzerland, http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf

Role of Randomness

- force selection of splitting to be random to avoid overfitting
- Keep trees uncorrelated
- Randomly select trees for training, selection of nodes for splitting etc

Brieman’s Random Forests

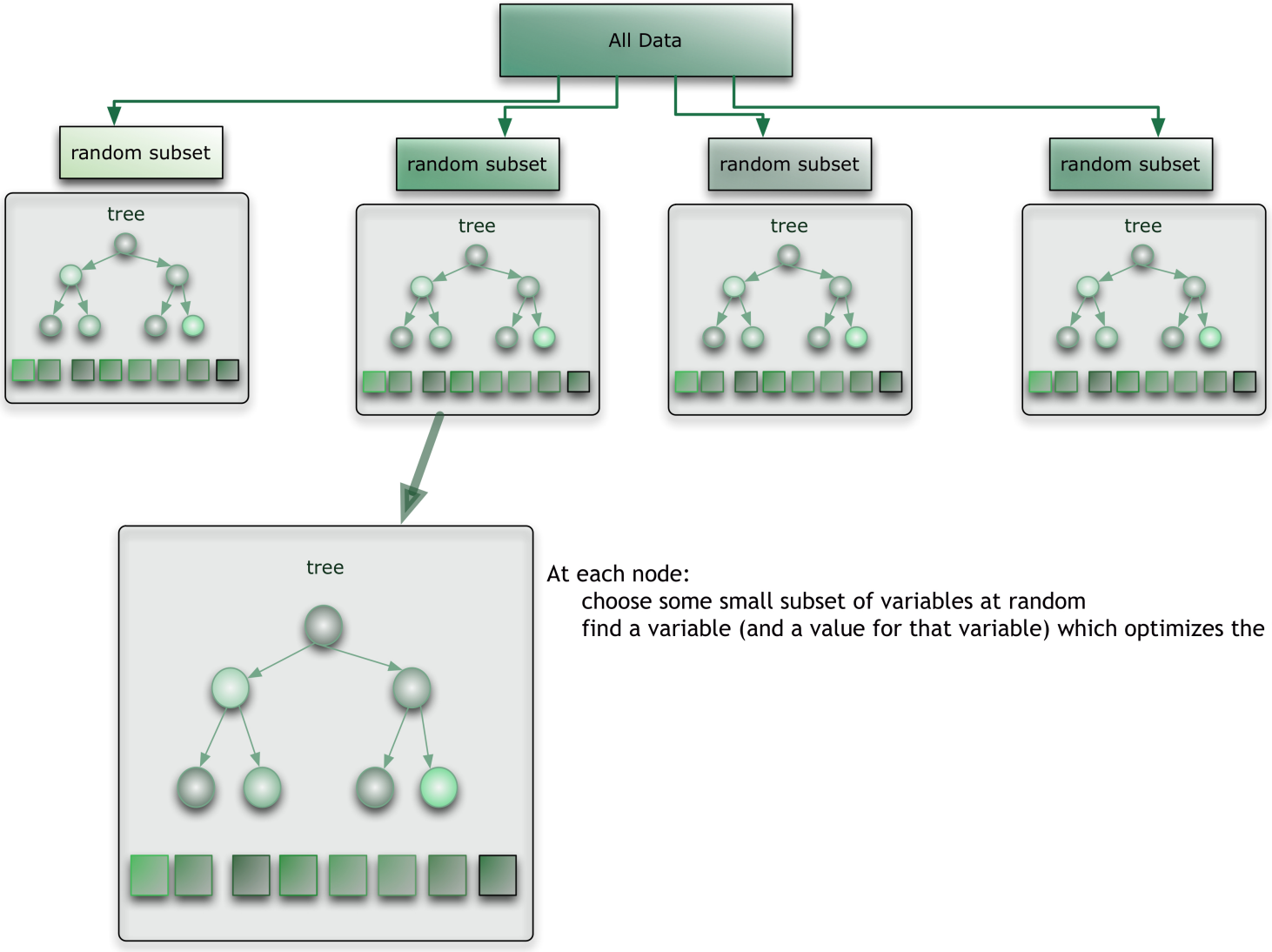
- Uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features.
- This process is sometimes called "feature bagging".
- Avoids creating a large number of correlated trees

Algorithm

- Grow a **forest** of many trees. (R default is 500)
- Grow each tree on an independent **bootstrap sample*** from the training data.
- At each node:
  - Select ***m*** variables **at random** out of all ***M*** possible variables (independently for each node).
  - Find the best split on the selected ***m*** variables.
- Grow the trees to maximum depth (classification).
- Vote/average the trees to get predictions for new data.

*Sample N cases at random with replacement.

Random Forests for Regression and Classification, Adele Cutler Utah State University
September 2010, Ovronnaz Switzerland, http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf



At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

Strengths

- Applicable to both regression and classification problems.
- Handle categorical predictors naturally.
- Computationally simple and quick to fit, even for large problems.
- No formal distributional assumptions (non-parametric).
- Can handle highly non-linear interactions and classification boundaries.
- Automatic variable selection, But need variable importance too.
- Handles missing values

Weakness

- Difficult to interpret.
- The picture of the tree does not give valuable insights into which variables are important and where.
- A black box approach for statistical modelers – you have very little control on what the model does. You can at best – try different parameters and random seeds!

Building, Training and Assessing

- Many algorithms available to use
- Learn 1-2 good algorithms for each class, tree, bagging, Kernel methods (e.g., support vector machines), neural networks

Algorithm Selection

- first step: identify appropriate learning paradigm
  - classification? regression?
  - labeled, unlabeled or a mix?
  - class proportions heavily skewed?
  - goal to predict probabilities? rank instances?
  - is interpretability of the results important?
- No learning algorithm dominates all others on all problems
- SVM's and boosting decision trees (as well as other tree ensemble methods) seem to be best off-the-shelf algorithms
- even so, for some problems, difference in performance among these can be large, and sometimes, much simpler methods do better

Data

- More data is usually good
- Training data must be like test data.

Features

- use your knowledge to know what features would be helpful for learning
  - redundancy in features is okay, and often helpful
  - most modern algorithms do not require independent features
- Too many features?
  - Feature selection methods
  - Often preferable to use algorithm designed to handle large feature sets

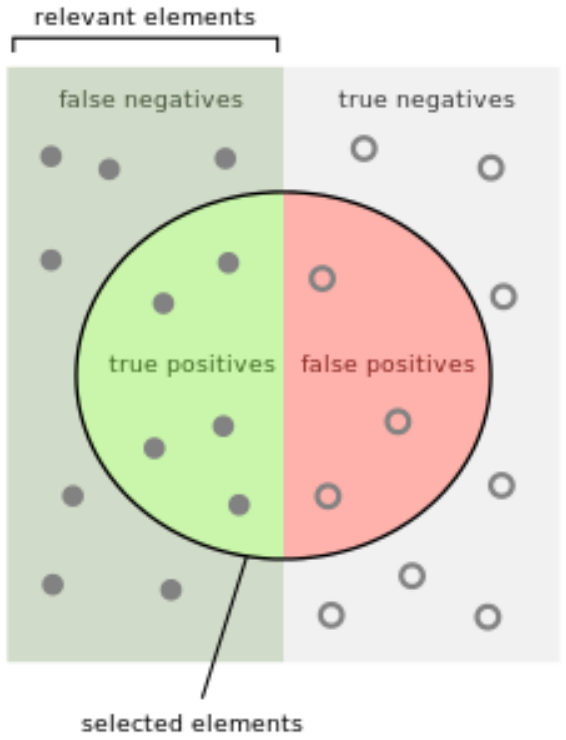
Error Types in Assessment

- Check for True Positive (TP) and True Negative (TN) Rates
- Two types of errors:
  - **type I error**: the incorrect rejection of a true null **hypothesis** (a "false positive"),
  - **type II error**: incorrectly retaining a false null **hypothesis** (a "false negative").

Assessment

Precision: How many positive items are relevant?

Recall: How many relevant items are selected?



Confusion Matrix

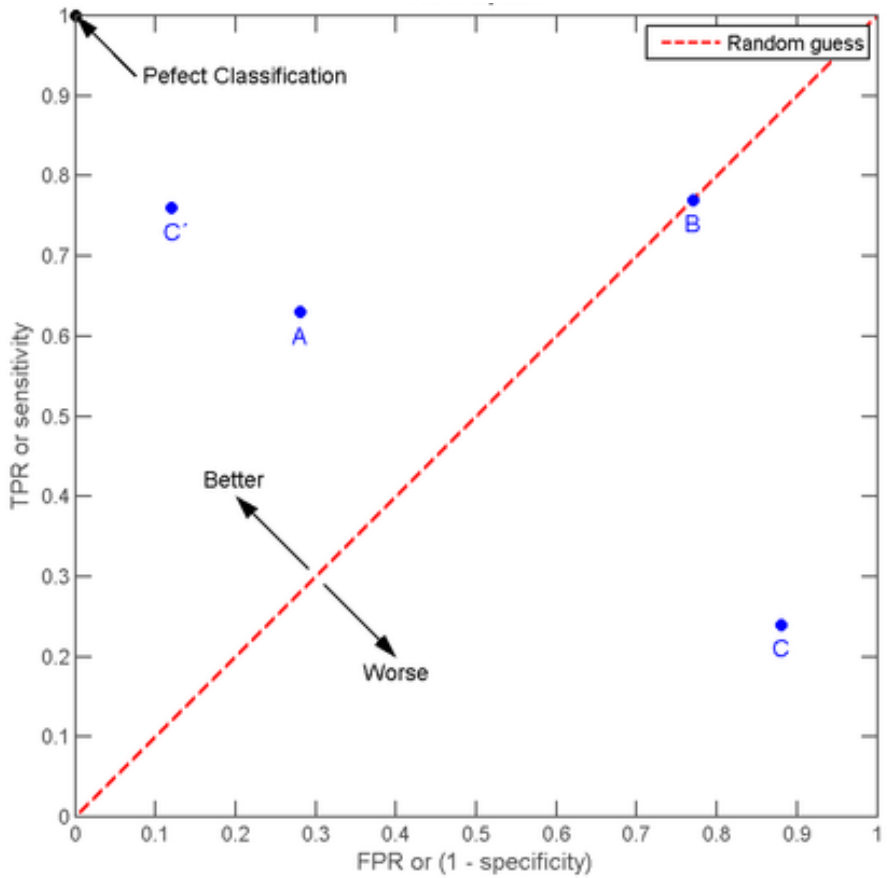
Total Population	Predicted Positive	Predicted Negative	
Actual Positive	True Positive (TP)	False Negative (FN) Type II error	Recall $\frac{TP}{TP+FN}$
Actual Negative	False Positive (FP) Type I error	True Negative (TN)	False Positive Rate $\frac{FP}{FP+TN}$
	Precision $\frac{TP}{TP+FP}$	False Omission Rate $\frac{FN}{FN+TN}$	

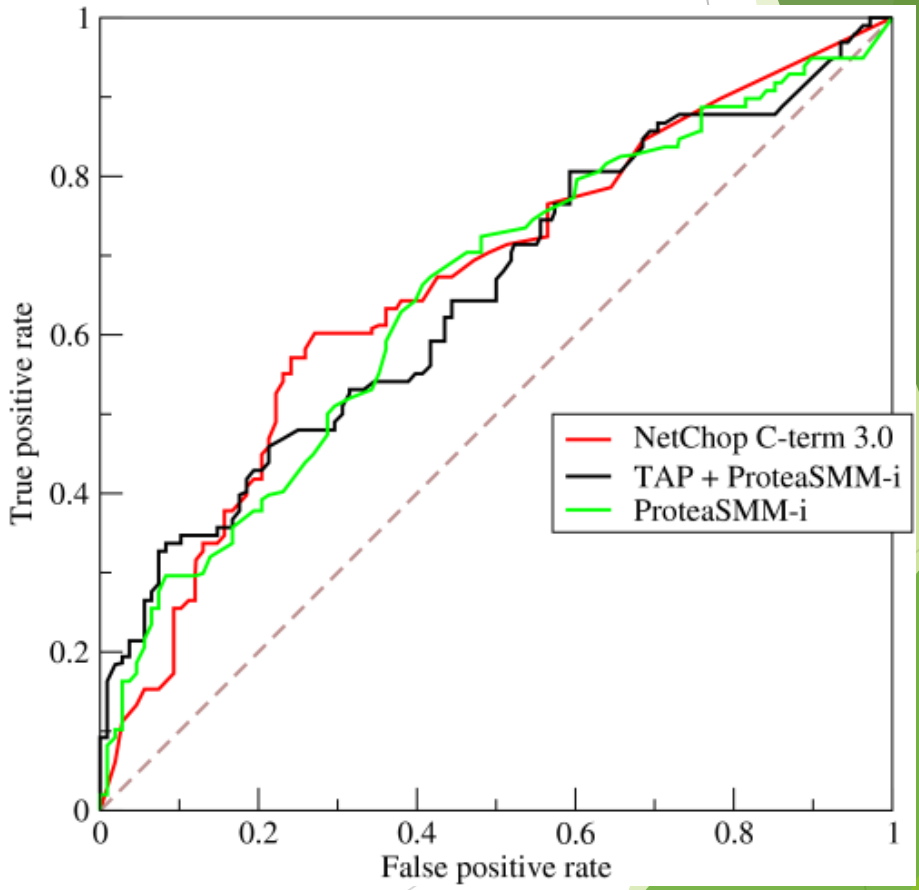
Specificity

$$\frac{TN}{FP+TN}$$

Receiver Operating Characteristic (ROC)

- Shows the performance of a binary classifier system as its discrimination threshold is varied.
- Plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.





Testing and Cross-Validation

- does it work? which algorithm is best?
- train on part of available data, and test on rest
- if dataset large (say, in 1000's), can simply set aside ≈ 1000 random examples as test
- • otherwise, use 10-fold cross validation
- • break dataset randomly into 10 parts
  • in turn, use each block as a test set, training on the other 9 blocks
- repeat many times
- use same train/test splits for each algorithm

Parameter Selection

- Sometimes, theory can guide setting of parameters, possibly based on statistics measurable on training set
- Other times, need to use trial and test, as before
- Danger: trying too many combinations can lead to overfitting test set
  - break data into train, validation and test sets
  - set parameters using validation set
  - measure performance on test set for selected parameter settings or do cross-validation within cross-validation
  - trying many parameter settings is also very computationally expensive

Implementation

- R and Matlab are great for prototyping, but for speed, may need C
- Automate and use scripts
- Debugging machine learning algorithms is very tricky!
  - hard to tell if working, since don't know what to expect
  - run on small cases where can figure out answer by hand
  - test each module/subroutine separately
  - compare to other implementations (written by others, or written in different language)
  - compare to theory or published results

References

- *Foundations of Machine Learning* by Mehryar Mohri, Afshin Rostamizadeh and Ameet Talwalkar MIT Press, 2012
- Tom M. Mitchell, *Machine Learning*, First Edition, McGraw-Hill, 1997