Bagging

IST557 Data Mining: Techniques and Applications

Jessie Li, Penn State University

Ensemble method

- Ensemble learning methods are meta-algorithms that pool decisions from multiple classifiers

Netflix Prize

- Netflix provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies. Each training rating is a quadruplet of the form <user, movie, date of grade, grade>.
- The *qualifying* data set contains over 2,817,131 triplets of the form <user, movie, date of grade>, with grades known only to the jury.
- The competition began on October 2, 2006.
- On September 21, 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%. “The Ensemble" team had matched BellKor's result, but since BellKor submitted their results 20 minutes earlier, the rules award the prize to BellKor.

Netflix Prize

3.2 Optimize the Blend

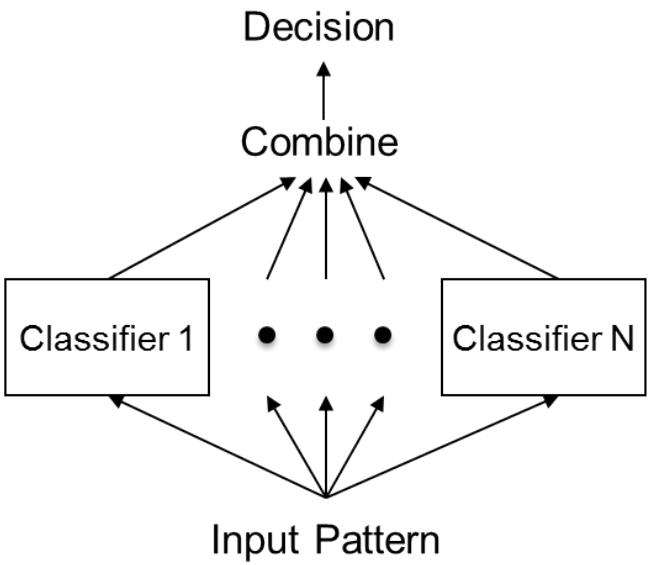
A key observation with ensemble methods is that it is not optimal to minimize the RMSE of the individual predictors. Only the RMSE of the ensemble counts. Thus the predictors which achieve the best blending results are the ones, which have the right balance between being uncorrelated to the rest of the ensemble and achieving a low RMSE individually. An ideal solution would be to train all models in parallel and treat the ensemble as one big model. The big problem is that training 100+ models in parallel and tuning all parameters simultaneously is computationally not feasible.

We approximate this ideal solution by training the models one after another, where each model tries to achieve best results when blended with all preceding models. So the focus shifts from looking at the RMSE of an individual predictor to the RMSE of a blended ensemble. In the following, we refer to the probe RMSE of a linear blend with all preceding predictors as "blend RMSE".

https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

Bagging and Boosting

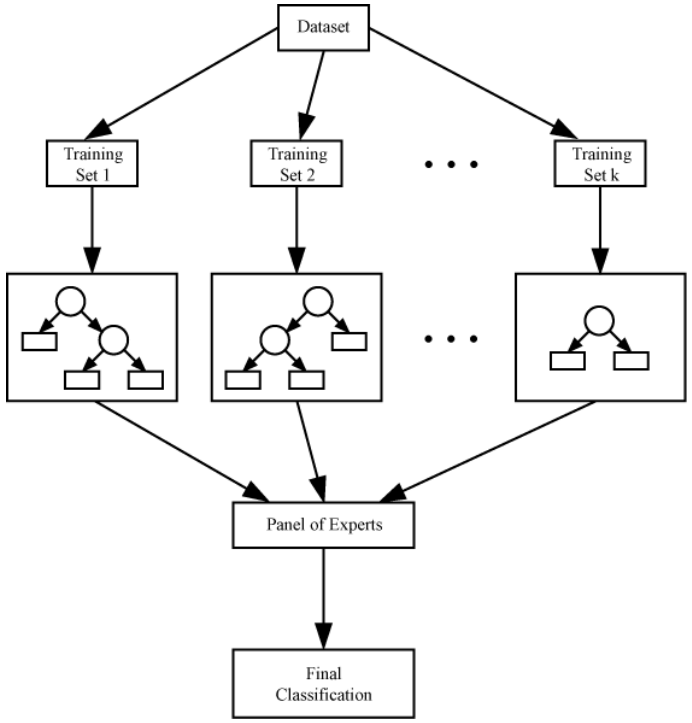
- Bagging and boosting are two frequently-used types of ensembles



Bagging

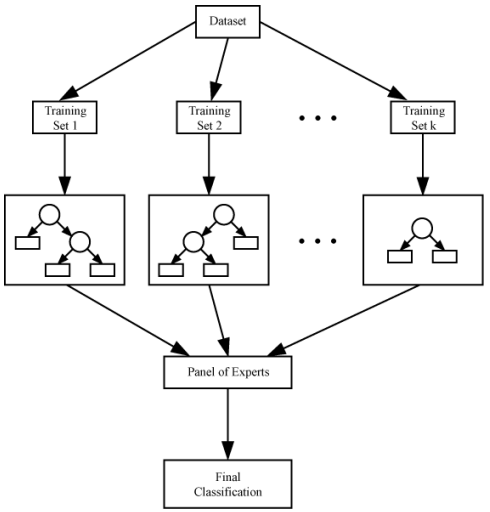
Bagging: **B**ootstrap **agg**regat**ing**
Bagging is a meta-algorithm that can be used on any classifier, but is usually applied to decision tree

1. Take a random sample of size N with replacement from the data (a bootstrap sample)
2. Construct a "full" tree
3. Repeat step 1-2 many times
4. For a data sample, get the prediction from each tree, and take the majority vote as the final prediction



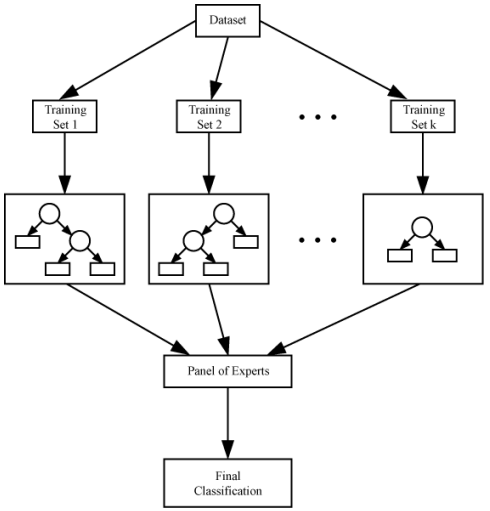
Bagging

- **Margin**: difference between the proportion of times a case is correctly classified and the proportion of times a case is incorrectly classified
  - Example: 100 trees, for an observation, 75 trees give the correct prediction, 25 trees give the wrong prediction. The margin is: 0.75-0.25=0.5
  - Large margins are desired



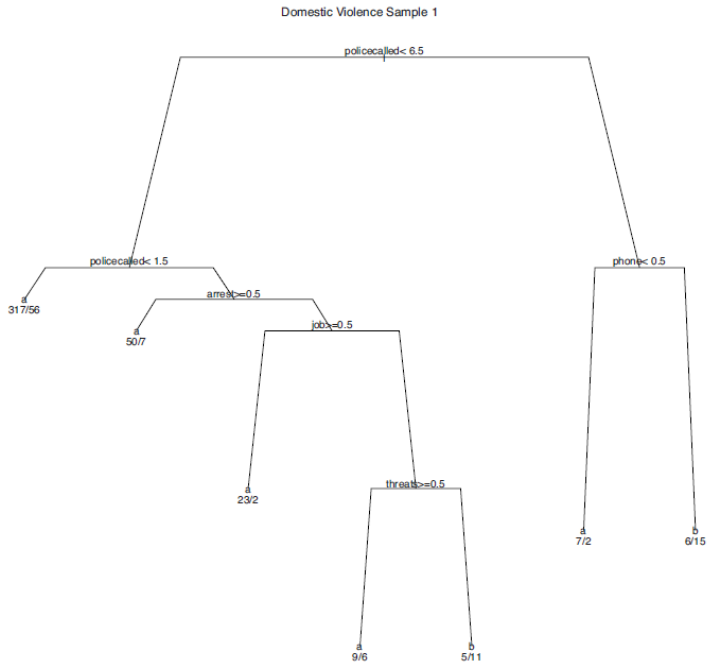
Bagging

- **Out-Of-Bag Observations**
  - Samples that are not used in bootstrap are out-of-bag observations
  - To fairly evaluate a model, the test data should not be in the bag of any tree fitting process

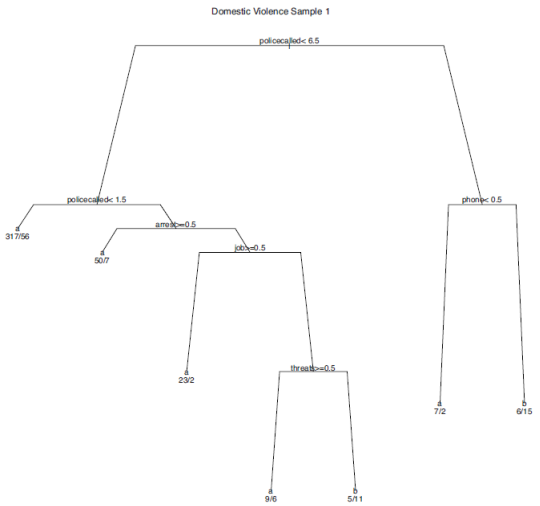


Example: Domestic Violence

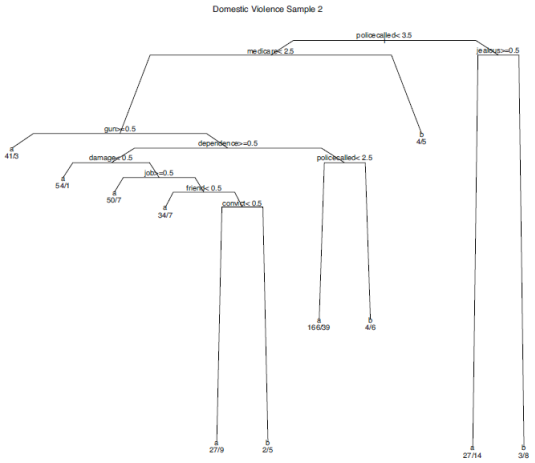
- Data collected from 500+ households: Household size and number of children; Male / female age (years); Marital duration; Male / female education (years); Employment status and income; The number of times the police had been called to that household before; Alcohol or drug abuse, etc.
- Forecast domestic violence



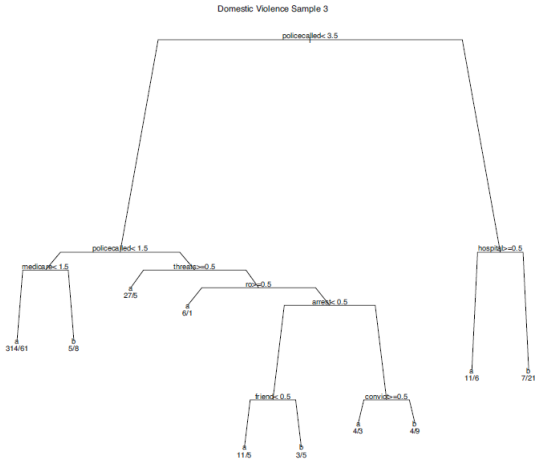
Different trees are constructed when vary training data



Tree 1



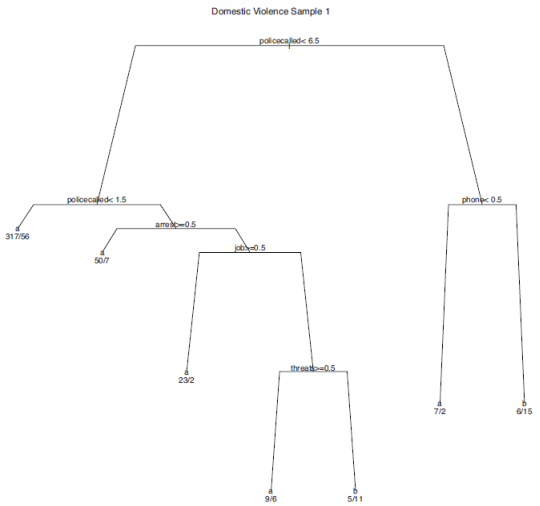
Tree 2



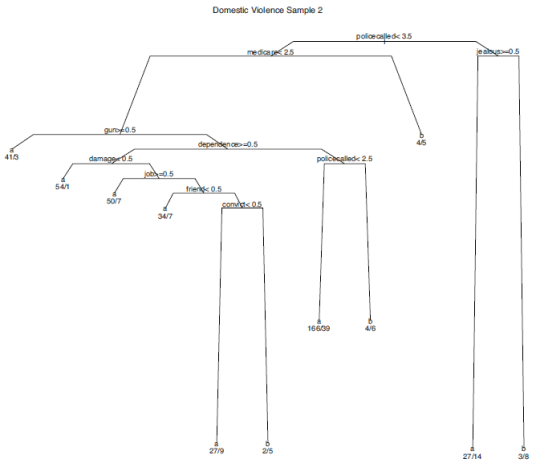
Tree 3

- Trees are different
  - Due to small sample sizes, highly correlated features; or heterogeneous terminal nodes
- Interpretations from the results of a single tree can be quite risky
  - However, when CART is used solely as a classification tool, the classes assigned may be relatively stable even if the tree structure is not

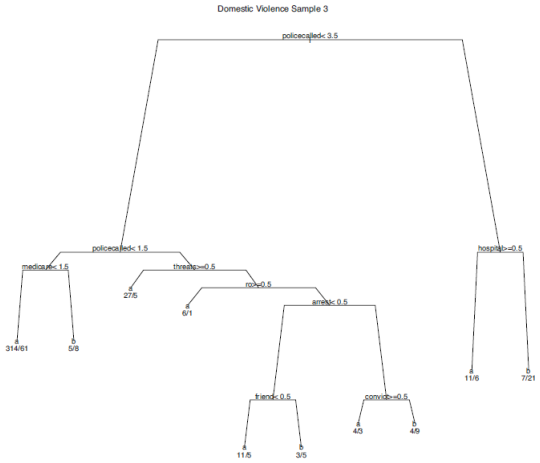
Different trees are constructed when vary training data



Tree 1



Tree 2

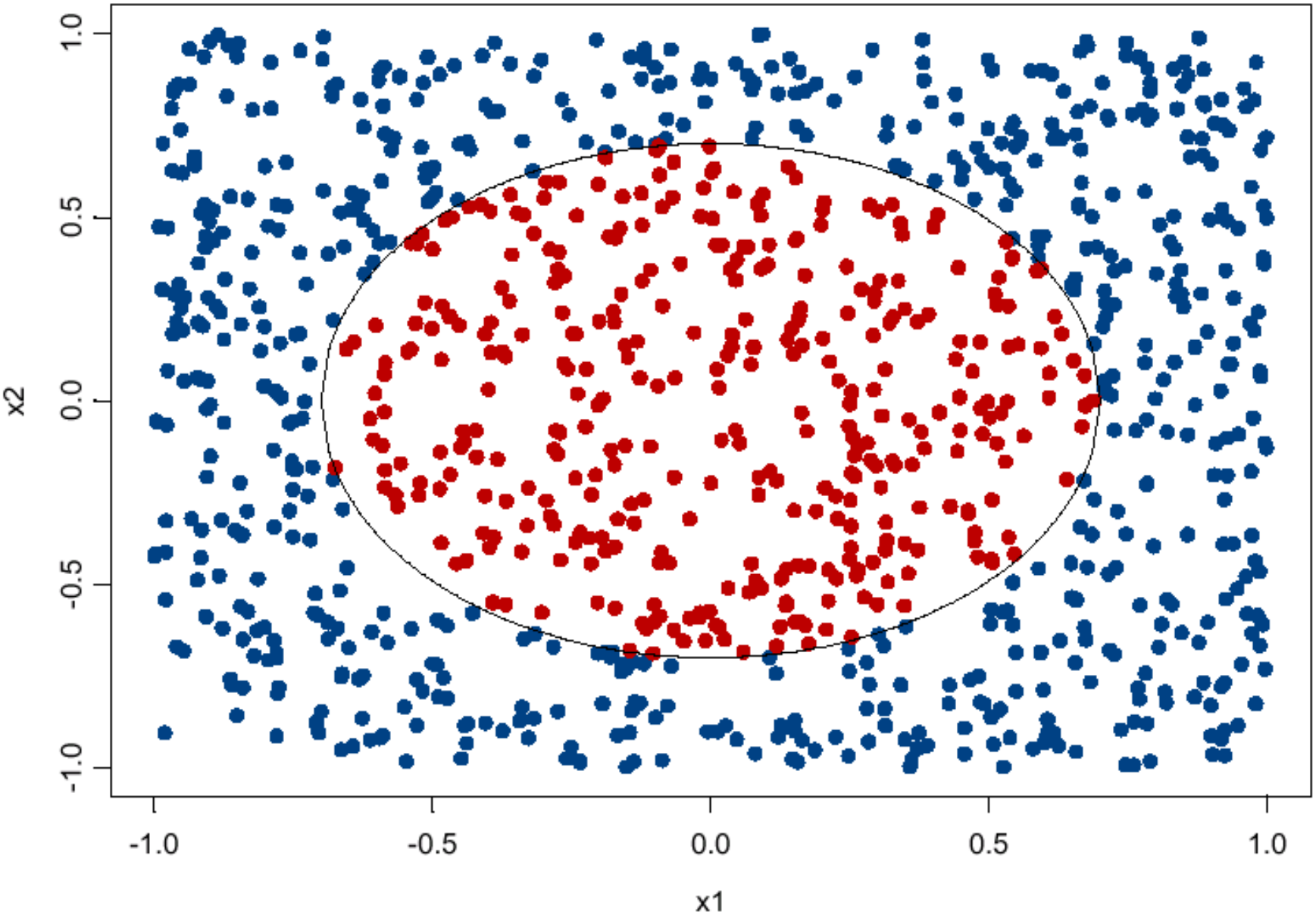


Tree 3

Why bagging works:

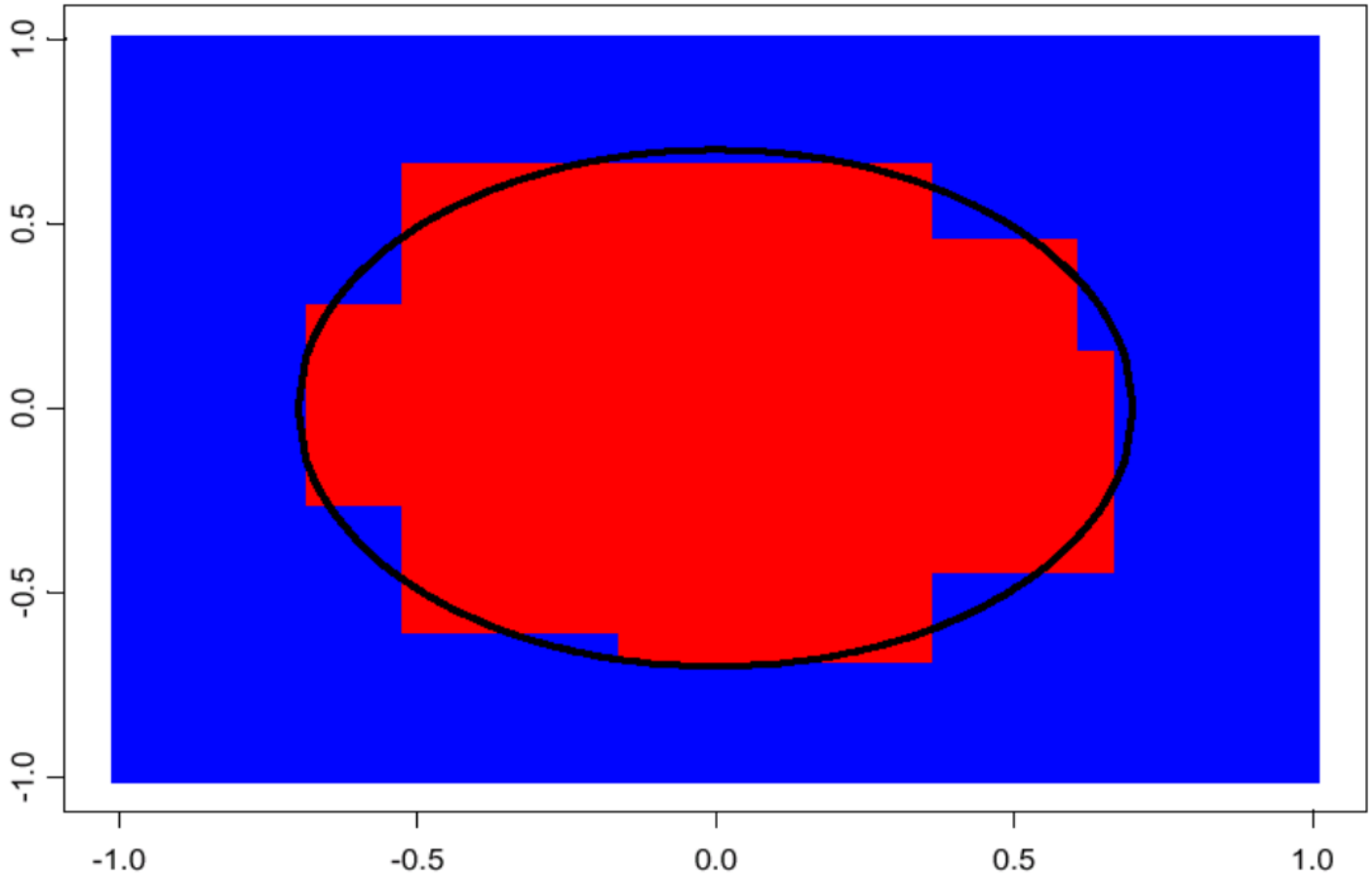
- Each tree has low bias but high variance
- Bagging can help with the variance
- bias-variance tradeoff

Bagging Example

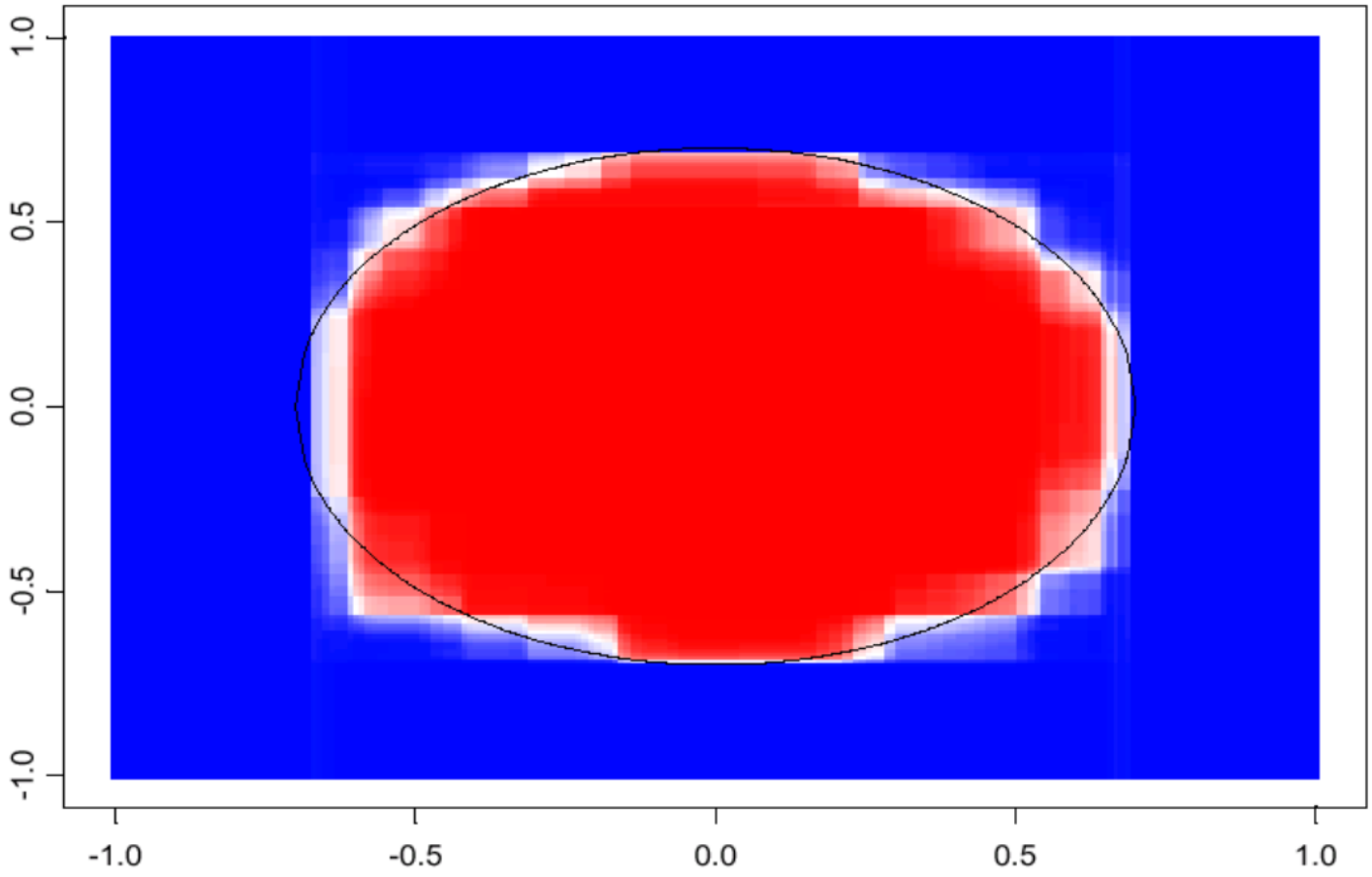


decision tree learning algorithm; very similar to ID3

CART decision boundary



100 bagged trees



shades of blue/red indicate strength of vote for particular classification

From Bagging to Random Forests

- Problem in bagging
  - Simply re-running the same learning algorithm on different subsets of the data can result in highly correlated features, which makes the model biased
- Random forest
  - Tries to de-correlate the base learners by learning trees based on a randomly chosen subset of input variables, as well as a randomly chosen subset of data samples
  - Such models often have very good predictive accuracy and have been widely used in many applications

Random Forest

1. Take a random sample of size *N* with replacement from the data (bootstrap sample)
2. If there are M features, randomly select m << M features
3. Construct a split by using m features selected in Step 2
4. Repeat Steps 2 and 3 for each subsequent split until the tree is as large as desired ("full" tree)
5. Repeat Steps 1-4 many times
6. For a data sample, get the prediction from each tree, and take the majority vote as the final prediction

Why Random Forests Work

- Variance reduction: the trees are more independent because of the combination of bootstrap samples and random draws of features
  - Benefit of bagging
- Bias reduction: a very large number of features can be considered, and local feature features can play a role in the tree construction
  - Often a few features that dominate CART fitting process; many other features (local features) are rarely selected as splitting variables
  - Random forests give all features some opportunities to be the splits

Parameters in Random Forest

- Node Size: unlike in CART, the number of observations in the terminal nodes of each tree of the forest can be very small. The goal is to grow trees with as little bias as possible (grow overfitting trees)
- Number of Trees: in practice, 500 trees is often a good choice
- Number of Features Sampled: the number of features sampled at each split would seem to be a key tuning parameter that should affect how well random forests perform. Sampling 2-5 each time is often adequate

Taking Costs into Account (Imbalanced Data)

- Domestic violence: predict *serious* domestic violence has actually occurred
  - 500+ samples
  - 29 felony serious violence
  - 29/500 ~ 4% → imbalanced data
- Cost
  - False negatives: fail to predict a serious incident
  - False positive: predict a case to be a serious incident when it is not
  - If treating them the same, the best prediction would be assuming no serious domestic violence with an error rate of 4%

Taking Costs into Account

- Weighted Classification Votes: After all of the trees are built, one can differentially weight the classification votes over trees
  - For example, one vote for classification in the rare category might count the same as two votes for classification in the common category.
- Stratified Bootstrap Sampling: When each bootstrap sample is drawn before a tree is built, one can oversample one class of cases for which forecasting errors are relatively more costly
  - The procedure is much in the same spirit as disproportional stratified sampling used for data collection (Thompson, 2002).

Summary

- Ensemble learning
  - Meta-algorithm
  - Bagging, boosting
- Bagging
  - Build many classifiers using different subsets of training
  - Average the results from all classifiers
  - Reduce variance
- Random forests
  - Bagging + random features
  - Give local features chances to be the splitting features
  - Also reduce bias