Ensemble Models

Ensemble Idea

- Combining multiple models into one provide better performance than individual model.

- Issues
  - How do you combine models?
    - Bagging
    - Bagging with Random features
    - Boosting
  - Types of models
    - Homogeneous vs Heterogeneous

Ensemble: Bagged Models

Bagging: Intuitive thought

- If a model is having low-bias and high-variance then it is strong but overfitted model (i.e.,the model fitted for valid and noise data)

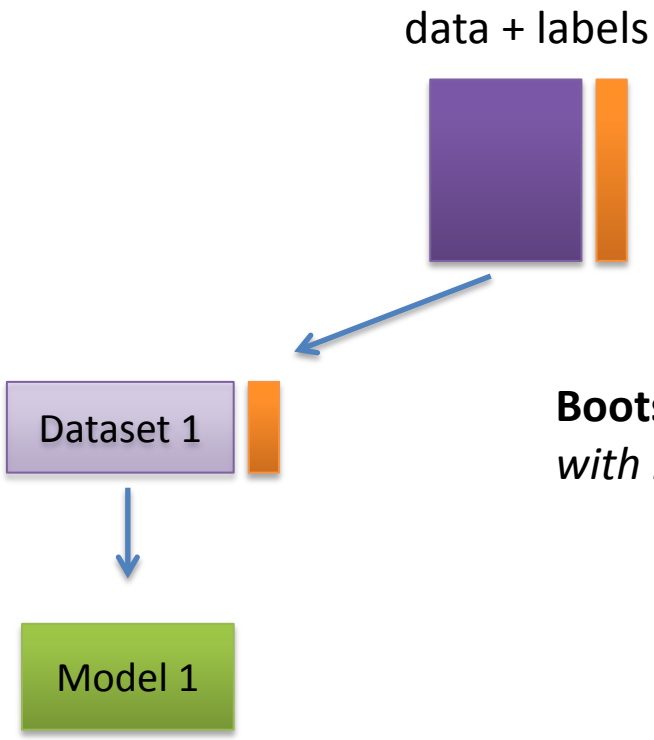
- One way to reduce variance of such model is to build multiple models on different bootstrap samples of train data and take a majority vote among them for predictions.

Bagging: Intuitive thought

- Bias of bagging = bias of the underlying model

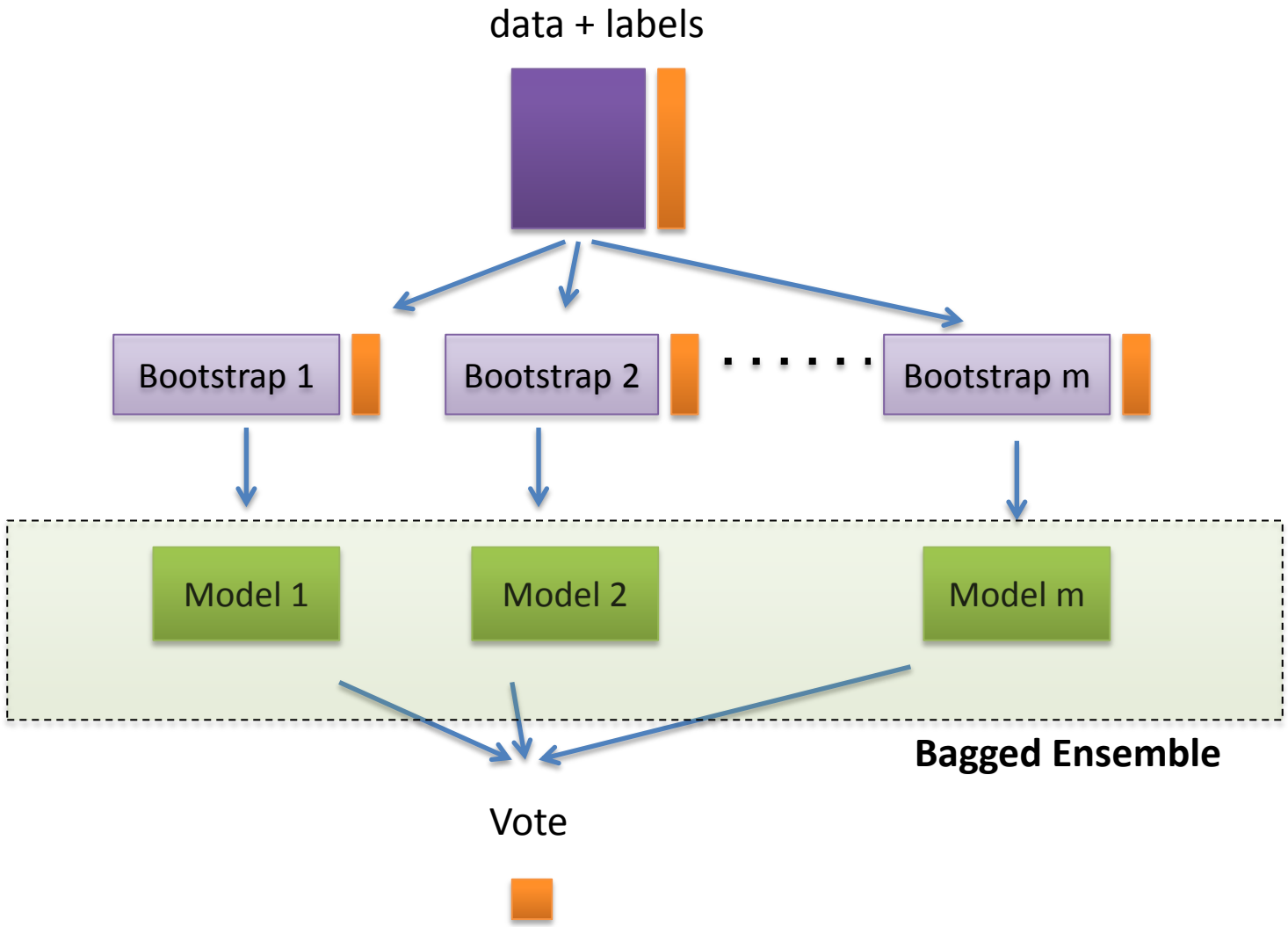
- Bagging reduces only the variance. Therefore, the underlying models must have a low bias, capturing the complexity of the relation between target variable and predictor features.

Bagging: Intuitive thought



Bootstrap: Sample N examples,
with replacement, from the original N.

Bagging: Intuitive thought



Bagged Trees

- Bagging can be applied to any kind of model. But the classification trees are particularly interesting because we can reduce individual bias by creating deep trees (unpruned trees). Afterwards, Bagging can reduce the variance in order to obtain an overall good predictor.

Bagged Tree: Algorithm

Algorithm Bagging($D, T, \mathscr{A}$)

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**

2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

3 run $\mathscr{A}$ on D_t to produce a model M_t;

4 **end**

5 **return** $\{M_t | 1 \leq t \leq T\}$

The decision boundary of a bagged classifier is more complex than that of the underlying classifiers.

Bagged Trees - Pros

- Works well if the base classifiers are having high-variance and low-bias.
- Increased accuracy because it reduces the variance of the individual classifier.
- Less susceptible to model over-fitting when applied to noisy data.

Bagged Trees - Cons

- If there is one very strong predictor along with a number of other moderately strong predictors in the train-data, then most of the trees will use this very strong predictor in the top split, consequently making most of the bagged trees look quite similar, and be highly correlated.

- Averaging many highly correlated quantities does not lead to large reduction in variance comparing to averaging many uncorrelated quantities.

Summary of Bagged Trees

Bagging idea becomes effective only if:

- Each tree must be individually efficient i.e., they should have low bias (grow trees as deep as possible)

- Each tree must be very different from each other so that combination is more effective (recall that ensemble calculations assume model independence)

Random Forest

Random Forests: Intuitive thought

- Random Forests overcome correlated trees problem of bagged-trees by forcing each tree to consider only a random subset of the predictors there by making trees un-correlated.

Random Forests: Intuitive thought

Grow a whole *forest* of trees:

- each tree is grown on an independent bootstrap sample* from the training data.
- independently, for each node of each tree, find the best split on ***m*** randomly selected variables.
- grow deep trees.

Get the prediction for a new case by voting (averaging) the predictions from all the trees.

Random Forest: Algorithm

Algorithm RandomForest(D, T, d)

Input : data set D; ensemble size T; subspace dimension d.

Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**

2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

3 select d features at random and reduce dimensionality of D_t accordingly;

4 train a tree model M_t on D_t without pruning;

5 **end**

6 **return** $\{M_t | 1 \leq t \leq T\}$

Evaluating Bagged Models

Out-of-bag Data

Think about a single tree from a random forest:

- The tree is grown on a bootstrap sample ("the bag").
- The remaining data are said to be "out-of-bag" (about one-third of the cases).
- The out-of-bag data serve as a test set for this tree.

Out-of-bag data give

- Estimated error rate
- A way to choose RF parameters (**m**, weights)
- Variable importance

Out-of-bag Error Rate

Think of a ***single case*** in the training set:

- It will be out-of-bag in about one-third of the trees.
- Predict its class for each of these trees.
- Its **RF prediction** is the most common predicted class.

For example, suppose we fit 1000 trees, and a case is out-of-bag in 339 of them, of which:

303 say "class 1"
36 say "class 2"

} The **RF prediction** is "class 1".

The out-of-bag error rate is simply the error rate of the RF predictor (can be done for each class).

Choosing RF parameters

RF splits on the best of **m** randomly selected variables at each node. The out-of-bag error rate is used to select **m**.

Here's how:

1. Start with $\mathbf{m} = \sqrt{\mathbf{M}}$, **M**=total number of variables.
2. Run a few trees, recording the out-of-bag error rate.
3. Increase **m**, decrease **m**, until you are reasonably confident you've found a value with low out-of-bag error rate.

Variable Importance

For each tree, look at the out-of-bag data:

- randomly permute the values of variable *j*, holding the other variables fixed.
- pass these permuted data down the tree, save the classes.

Importance for variable *j* is

$$\left[\begin{array}{l}\text{error rate when}\\ \text{variable } j \text{ is permuted}\end{array}\right] - \left[\begin{array}{l}\text{error rate for the}\\ \text{original data}\end{array}\right]$$

where the first error rate is averaged over the out-of-bag data.

Random Forest - Pros

- Same idea for regression and classification YES!
- Handle categorical predictors naturally YES!
- Quick to fit, even for large problems YES!
- No formal distributional assumptions YES!
- Automatically fits highly non-linear interactions YES!
- Automatic variable selection YES! importance
- Handle missing values through proximities

Random Forest - Pros

Improve on CART with respect to:

- *Accuracy* – Random Forests is competitive with the best known machine learning methods (but note the "no free lunch" theorem)
- *Instability* – if we change the data a little, the individual trees will change but the forest is more stable because it is a combination of many trees

Random Forest - Cons

- Not easy to interpret
- If the number of relevant attributes is very low compared to total number then individual trees may not be efficient
- The terminal nodes may not suggest a natural clustering