CS685: Data Mining
Ensemble Methods

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs685/

1st semester, 2020-21

Mon 1030-1200 (online)

Ensemble Methods

- Combine *multiple* classifiers to increase classification accuracy
- A single instance of a classifier may do a mistake, but it is unlikely that many of them will
- Ensemble methods are also called classifier combination methods

Ensemble Methods

- Combine *multiple* classifiers to increase classification accuracy
- A single instance of a classifier may do a mistake, but it is unlikely that many of them will
- Ensemble methods are also called classifier combination methods
- Final class is determined by simply using *majority voting* of outputs of individual classifiers
- Voting may be *weighted* as well

Ensemble Methods

- Combine *multiple* classifiers to increase classification accuracy
- A single instance of a classifier may do a mistake, but it is unlikely that many of them will
- Ensemble methods are also called classifier combination methods
- Final class is determined by simply using *majority voting* of outputs of individual classifiers
- Voting may be *weighted* as well
- Classifiers of same or different types can be combined
- Same type: bagging, boosting
- Different type: stacking

Ensemble Methods

- Combine *multiple* classifiers to increase classification accuracy
- A single instance of a classifier may do a mistake, but it is unlikely that many of them will
- Ensemble methods are also called classifier combination methods
- Final class is determined by simply using *majority voting* of outputs of individual classifiers
- Voting may be *weighted* as well
- Classifiers of same or different types can be combined
- Same type: bagging, boosting
- Different type: stacking
- Assumes *no* class imbalance problem, i.e., classes are almost equally represented

Bagging

- Short form of bootstrap aggregating
- Suppose, initial training set size is d
- Create another training set of *same* size d by *sampling with replacement*
  - Equivalent to deleting some objects and replicating some others
- Repeat t times
- Build t classifiers of the *same* type on each of the training samples
- Use majority voting on the t classifiers for final classification

Bagging

- Short form of bootstrap aggregating
- Suppose, initial training set size is d
- Create another training set of *same* size d by *sampling with replacement*
  - Equivalent to deleting some objects and replicating some others
- Repeat t times
- Build t classifiers of the *same* type on each of the training samples
- Use majority voting on the t classifiers for final classification
- Random forests can be considered as a type of bagging

0.632 Bootstrap

0.632 Bootstrap

- Creating a sample of size d from a set of size d by *sampling with replacement*

0.632 Bootstrap

- Creating a sample of size d from a set of size d by *sampling with replacement*
- Probability of a particular object being chosen in a try is $1/d$
- Therefore, probability of not being chosen is $1 - 1/d$

0.632 Bootstrap

- Creating a sample of size d from a set of size d by *sampling with replacement*
- Probability of a particular object being chosen in a try is $1/d$
- Therefore, probability of not being chosen is $1 - 1/d$
- Thus, probability of *not* being chosen at all in any of the d tries is $(1 - 1/d)^d$

0.632 Bootstrap

- Creating a sample of size d from a set of size d by *sampling with replacement*
- Probability of a particular object being chosen in a try is $1/d$
- Therefore, probability of not being chosen is $1 - 1/d$
- Thus, probability of *not* being chosen at all in any of the d tries is $(1 - 1/d)^d$
- For large d, $(1 - 1/d)^d \to 1/e = 0.368$
- Hence, probability that an object is chosen at least once, i.e., it is represented in the sample is $1 - 0.368 = 0.632$
- Hence, the method is called 0.632 bootstrap

Boosting

- Classifiers of the same type
- Uses majority voting of classifiers
- May use weights
- Models are *not* built independently
- Each model is built *successively* on the previous ones

Boosting

- Classifiers of the same type
- Uses majority voting of classifiers
- May use weights
- Models are *not* built independently
- Each model is built *successively* on the previous ones
- Each new model concentrates more on the training objects mis-classified by previous models
- Each training set is of the *same* size d and is constructed using *sampling with replacement*

Adaboost

- A popular boosting algorithm is Adaboost
- Each training object is weighted (initially with the same weight)
- For a model, the *error* in classification is computed as the ratio of the sum of weights of mis-classified objects to total weight
- Total weight is normalized to 1
- Hence, error e is

$$e = \sum_{p \text{ is mis-classified}} w(p)$$

Adaboost

- A popular boosting algorithm is Adaboost
- Each training object is weighted (initially with the same weight)
- For a model, the *error* in classification is computed as the ratio of the sum of weights of mis-classified objects to total weight
- Total weight is normalized to 1
- Hence, error e is

$$e = \sum_{p \text{ is mis-classified}} w(p)$$

- Models are successively built by *modifying* the weights of training objects
- If a training object is *correctly* classified, its weight is *decreased*
- Consequently, weights of mis-classified objects increase

Adaboost

- A popular boosting algorithm is Adaboost
- Each training object is weighted (initially with the same weight)
- For a model, the *error* in classification is computed as the ratio of the sum of weights of mis-classified objects to total weight
- Total weight is normalized to 1
- Hence, error e is

$$e = \sum_{p \text{ is mis-classified}} w(p)$$

- Models are successively built by *modifying* the weights of training objects
- If a training object is *correctly* classified, its weight is *decreased*
- Consequently, weights of mis-classified objects increase
- For the next model, sampling with replacement is done using the new weights as probabilities

Updating the Weights

- Initially, the weights of training objects are same
- Thus, model 1 is built by simple sampling with replacement from training set
- Weights of all *correctly* classified objects are modified using the error e of the model

$$w(q) = w(q) \times \frac{e}{(1-e)} \text{ where } q \text{ is correctly classified}$$

Updating the Weights

- Initially, the weights of training objects are same
- Thus, model 1 is built by simple sampling with replacement from training set
- Weights of all *correctly* classified objects are modified using the error e of the model

$$w(q) = w(q) \times \frac{e}{(1-e)} \text{ where } q \text{ is correctly classified}$$

- If $e = 0$, then all objects are correctly classified and model building is stopped
- If $e > 0.5$, then more than half the objects are mis-classified
- Such a model is deemed useless and is discarded

Updating the Weights

- Initially, the weights of training objects are same
- Thus, model 1 is built by simple sampling with replacement from training set
- Weights of all *correctly* classified objects are modified using the error e of the model

$$w(q) = w(q) \times \frac{e}{(1-e)} \text{ where } q \text{ is correctly classified}$$

- If $e = 0$, then all objects are correctly classified and model building is stopped
- If $e > 0.5$, then more than half the objects are mis-classified
- Such a model is deemed useless and is discarded
- Sampling is repeated till a suitable stopping criterion

Updating the Weights

- Initially, the weights of training objects are same
- Thus, model 1 is built by simple sampling with replacement from training set
- Weights of all *correctly* classified objects are modified using the error e of the model

$$w(q) = w(q) \times \frac{e}{(1-e)} \text{ where } q \text{ is correctly classified}$$

- If $e = 0$, then all objects are correctly classified and model building is stopped
- If $e > 0.5$, then more than half the objects are mis-classified
- Such a model is deemed useless and is discarded
- Sampling is repeated till a suitable stopping criterion
- Final models are weighted for majority voting according to $log \frac{1-e_i}{e_i}$ where e_i is the error of model i

Discussion

- Models are necessarily of the same type
- Model outputs are assumed to be independent for voting

Discussion

- Models are necessarily of the same type
- Model outputs are assumed to be independent for voting
- Bagging is simple
- Boosting progressively concentrates on harder training objects
- Boosting may overfit

Discussion

- Models are necessarily of the same type
- Model outputs are assumed to be independent for voting
- Bagging is simple
- Boosting progressively concentrates on harder training objects
- Boosting may overfit
- Assumes no *class imbalance*
- If a class is under-represented, it has to be *over-sampled*
- If a class is over-represented, it has to be *under-sampled*
- May need to take into account the relative populations while computing the mis-classification error

Stacking

- Stacking or stacked generalization combines models of *different* types

Stacking

- Stacking or stacked generalization combines models of *different* types
- Individual classifiers are models of level 0
- Builds another classifier of level 1 that combines the *outputs* of level-0 classifiers
- Level-1 classifier uses *only* the outputs of level-0 models, i.e., just their class labels and *not* the attributes of the training objects
- May use errors as well

Stacking

- Stacking or stacked generalization combines models of *different* types
- Individual classifiers are models of level 0
- Builds another classifier of level 1 that combines the *outputs* of level-0 classifiers
- Level-1 classifier uses *only* the outputs of level-0 models, i.e., just their class labels and *not* the attributes of the training objects
- May use errors as well
- Level-1 classifier is thus just an arbiter
- It should be simple

Stacking

- For better results, training sample is divided into two parts D_0 and D_1
- Level-0 classifiers are trained using *only* D_0 and *not* D_1
- Level-1 classifier is trained using *only* D_1 and *not* D_0
  - D_1 training objects are simply passed through level-0 classifiers to get their outputs
  - These outputs act as inputs to the level-1 classifier
  - D_1 is not used for *training* level-0 classifiers at all