Bagging and Boosting

Lecture 8.1

Outline

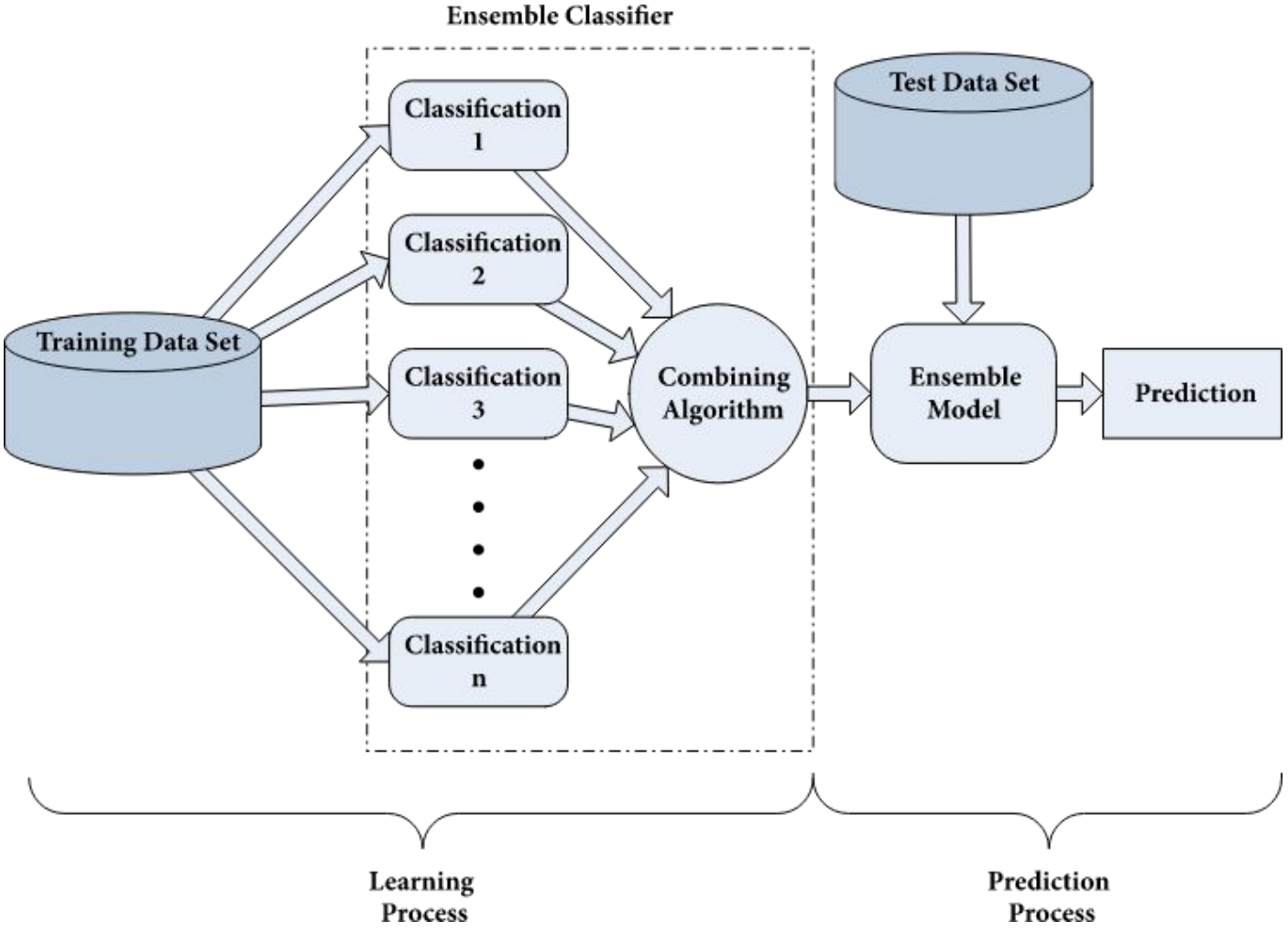
- Techniques to **improve classification accuracy**
- We focus on **ensemble methods**

- **Bagging**
- **Boosting**
  - **Adaptive Boosting (AdaBoost)**
- **Random Forest**

- Recap classification techniques and evaluation methods (Ch 8)

What is an ensemble method?

- **Ensemble** is a Machine Learning concept in which the idea is to **train multiple models** using the **same learning algorithm**
- An ensemble for classification is a composite model, made up of a combination of classifiers
- The **individual classifiers vote**, and a class label prediction is returned by the ensemble based on the collection of votes
- Ensembles tend to be more accurate than their component classifiers


Ensemble Classifier
Test Data Set
Classification 1
Classification 2
Training Data Set
Classification 3
Combining Algorithm
Ensemble Model
Prediction
Classification n
Learning Process
Prediction Process

Why ?

Diverse group of people are likely to make better decisions as compared to individuals.

The crowd wisdom

trying to “get rid of” the variance of the individual predictions by averaging

Tragedy of commons

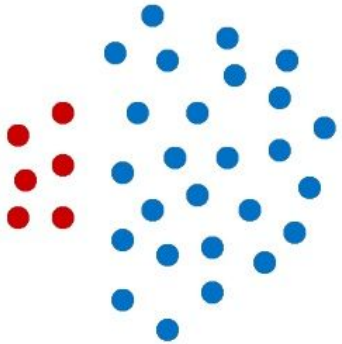
individuals acting in their own best interest behave in a way that is contrary to the common good

Why ?

Traditional learning models assume that the data classes are well distributed.

In many real-world data domains, however, the data are class-imbalanced, where the main class of interest is represented by only a few tuples.

Ensemble learning is a solution for improving classification in imbalanced data



Ensemble Methods

- An ensemble combines a series of k learned models (or base classifiers), $M1, M2, \ldots, Mk$ with the aim of creating an improved composite classification model, $M*$

- A given data set, D, is used to create k training sets, $D1, D2, \ldots, Dk$, where Di $(1 \leq i \leq k - 1)$ is used to generate classifier Mi

- Given a new data tuple to classify, the base classifiers each vote by returning a class prediction. The ensemble returns a class prediction based on the votes of the base classifiers.

Bagging- intuition

Suppose that you are a patient and would like to have a diagnosis made based on your symptoms. Instead of asking one doctor, you may choose to ask several. If a certain diagnosis occurs more than any other, you may choose this as the final or best diagnosis.

That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote.

Bagging

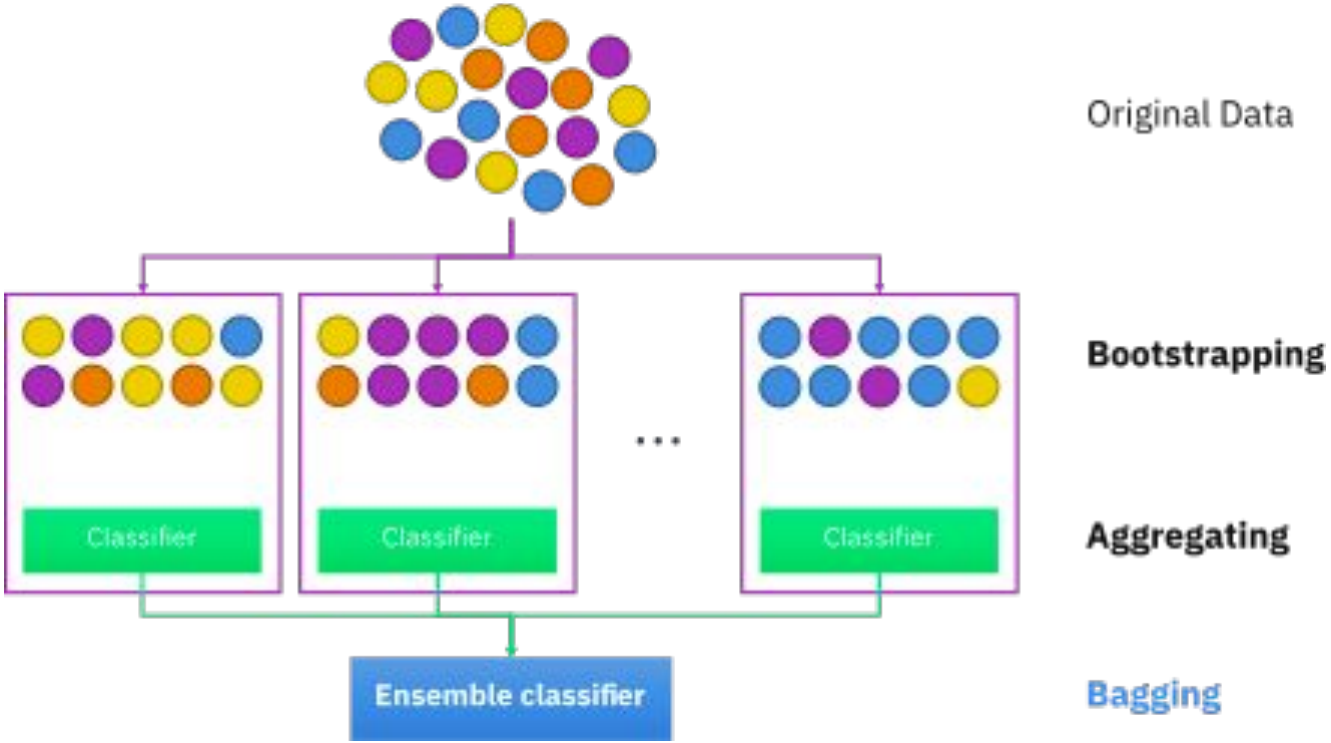
- **B**ootstrap **agg**regat**ing**
- **Bootstrap** is a sampling technique- **random sampling with replacement**

- Given a set, D, of *d* tuples, bagging works as follows. For iteration i (i = 1, 2, . . . , k), **a training set**, Di , of *d* tuples is **sampled with replacement** from the original set of tuples, D.
- In bagging, each **training set is a bootstrap sample**

Bagging

- Because sampling with replacement is used, some of the original tuples of D may not be included in Di , whereas others may occur more than once.
- A classifier model, Mi , is learned for each training set, Di .
- To classify an unknown tuple, X, each classifier, Mi , returns its class prediction, which counts as one vote.
- The bagged classifier, M∗, counts the votes and assigns the class with the most votes to X.

Bagging


Original Data
Classifier
Classifier
Classifier
Bootstrapping
Aggregating
Ensemble classifier
Bagging

Bagging

- **Bagging** can be applied to the prediction of **continuous values** by taking the **average value of each prediction** for a given test tuple.

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, $M*$.

Method:

(1) **for** $i = 1$ to k **do** // create k models:
(2) create bootstrap sample, D_i, by sampling D with replacement;
(3) use D_i and the learning scheme to derive a model, M_i;
(4) **endfor**

To use the ensemble to classify a tuple, X:

let each of the k models classify $\boldsymbol{X}$ and return the majority vote;

Bagging

Classifier generation:

Let N be the size of the training set.

for each of t iterations:

    sample N instances with replacement from the original training set.

    apply the learning algorithm to the sample.

    store the resulting classifier.

Classification:

for each of the t classifiers:

    predict class of instance using classifier.

return class that was predicted most often.

```
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
import pandas as pd

# load the data
url = "/home/debomit/Downloads/wine_data.xlsx"
dataframe = pd.read_excel(url)
arr = dataframe.values
X = arr[:, 1:14]
Y = arr[:, 0]

seed = 8
kfold = model_selection.KFold(n_splits = 3,
                        random_state = seed)

# initialize the base classifier
base_cls = DecisionTreeClassifier()

# no. of base classifier
num_trees = 500

# bagging classifier
model = BaggingClassifier(base_estimator = base_cls,
                          n_estimators = num_trees,
                          random_state = seed)

results = model_selection.cross_val_score(model, X, Y, cv = kfold)
print("accuracy :")
print(results.mean())
```

Next

Boosting and Adaboost

Boosting and AdaBoost

Lecture 8.2

Recap

- Ensemble classification
- Bagging
- Bagging algorithm
  - Creating multiple datasets
  - Building multiple classifiers
  - Combining Classifiers

Today:
Boosting and Adaboost

Boosting

- Ensemble learning technique

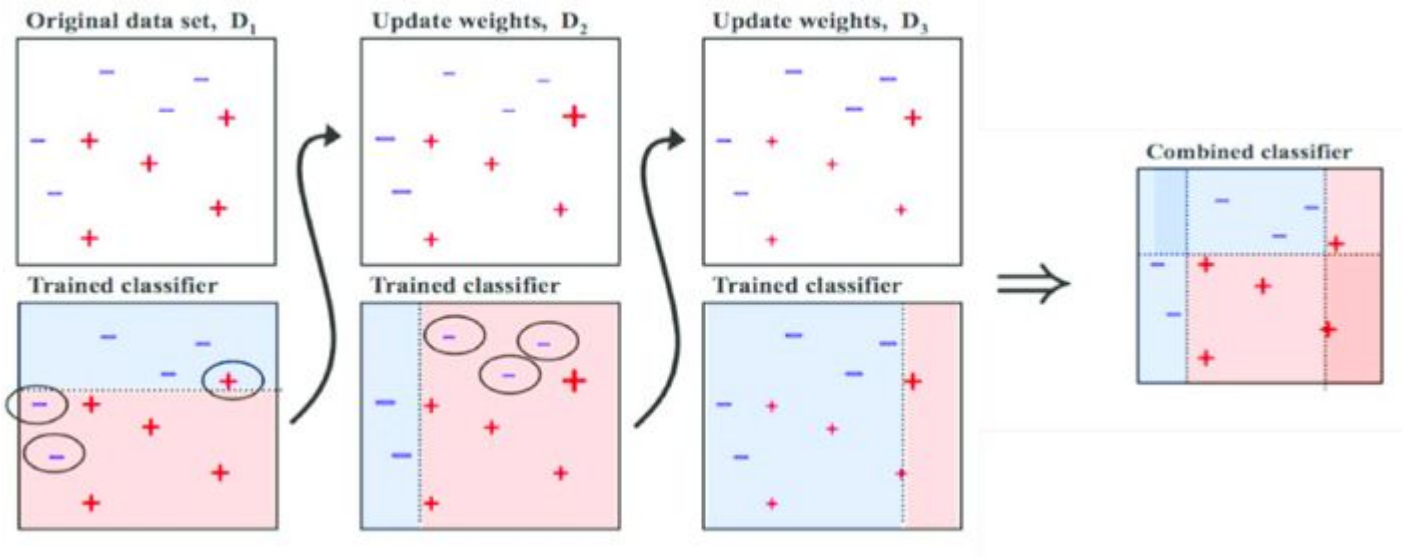
Assume this situation

- If a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results?
- Such situations are taken care of by boosting.
- Combine **weak learners into a strong learner.**

Boosting- intuition

Suppose that as a patient, you have certain symptoms. Instead of consulting one doctor, you choose to consult several. Suppose you assign weights to the value or worth of each doctor’s diagnosis, based on the accuracies of previous diagnoses they have made. The final diagnosis is then a combination of the weighted diagnoses.

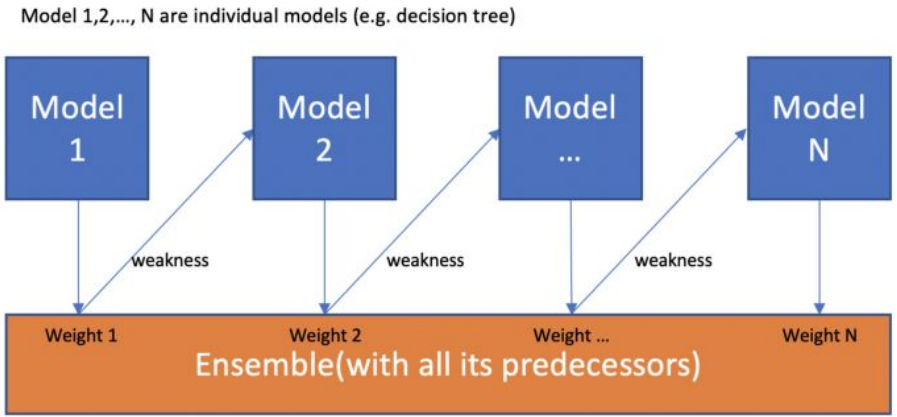
Boosting

- Boosting is a **sequential process**, where each subsequent model attempts to **correct the errors of the previous model**.
- 

https://medium.com/swlh/boosting-and-bagging-explained-with-examples-5353a36eb78d

Boosting

- Train model A on the whole set
- Train the model B with exaggerated data on the regions in which A performs poorly (i.e **“pay more attention”** to the training tuples that were misclassified)
- The final boosted classifier, M∗, combines the votes of each individual classifier

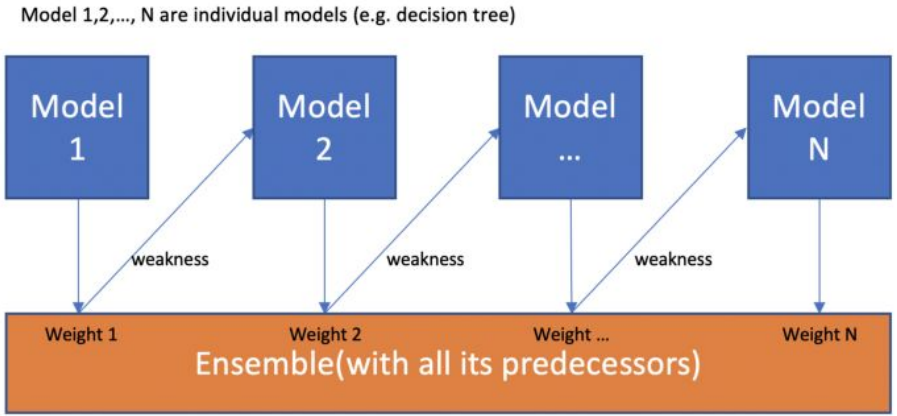


Boosting

- Train model A on the whole set
- Train the model B with exaggerated data on the regions in which A performs poorly (i.e **“pay more attention”** to the training tuples that were **misclassified**)
- The final boosted classifier, M*, combines the votes of each individual classifier

Weak classifier

Strong classifier



Adaboost

Adaptive **Boost**ing

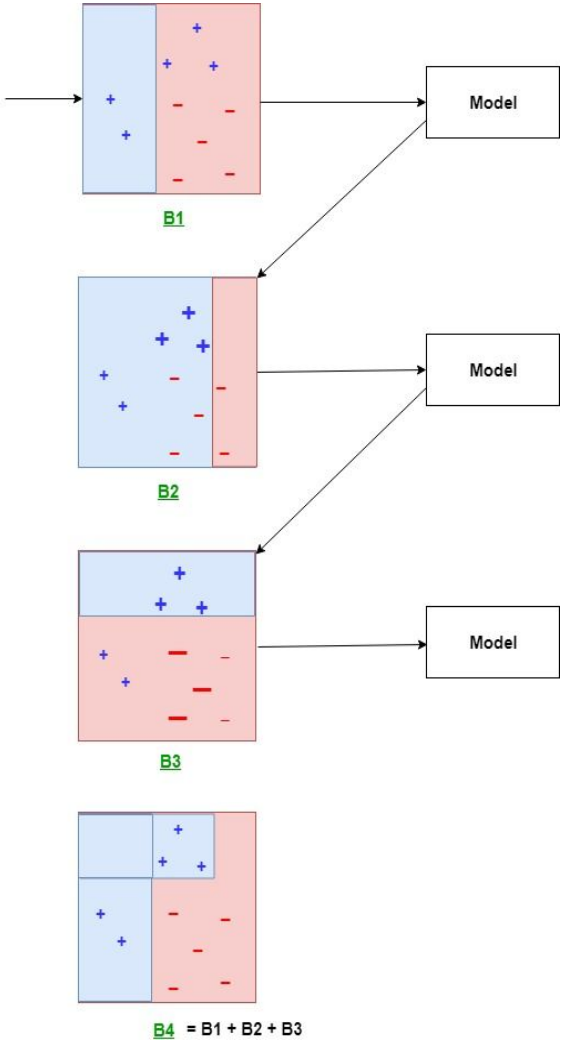
1. We are given D, a data set of d class-labeled tuples, (X1 , y1), (X2 , y2), . . . , (Xd , yd)
2. Initialise the dataset and assign equal weight (**of 1/d)** to each of the data point
3. Provide this as input to the model and identify the wrongly classified data points
4. Increase the weight of the wrongly classified data points.
5. Repeat steps 3-4 until got required results

Adaboost

B1 - We have 10 points 5 + and 5 -

Each one has been assigned equal weight initially.

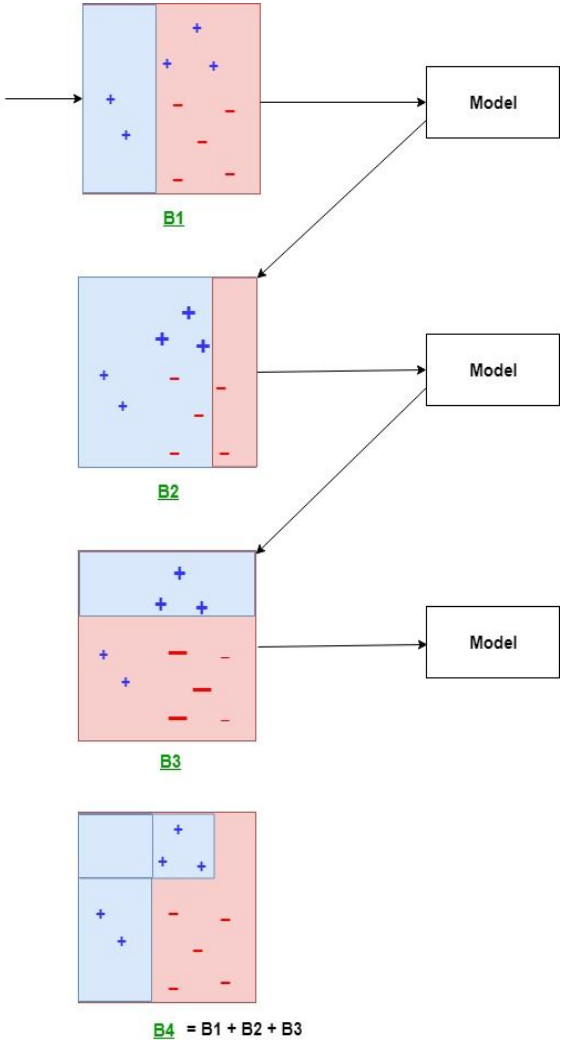
The first model tries to classify the data points and generates a vertical separator line but it wrongly classifies 3 plus(+) as minus(-).



Adaboost

B2 consists of the 10 data points from the previous model in which the 3 wrongly classified plus(+) are weighted more so that the current model tries more to classify these pluses(+) correctly.

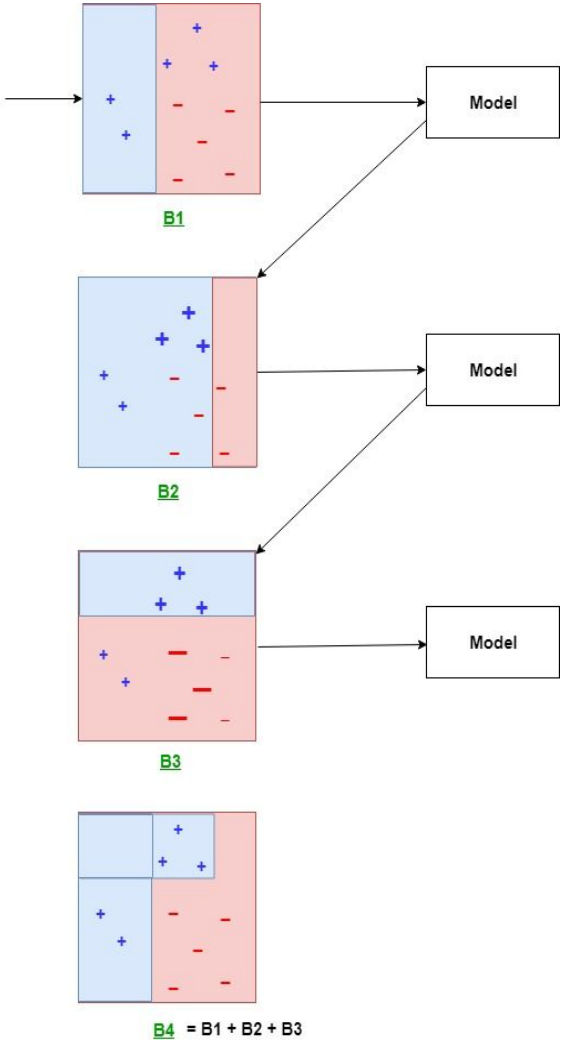
This model generates a vertical separator line which correctly classifies the previously wrongly classified pluses(+) but in this attempt, it wrongly classifies two minuses(-).



Adaboost

B3 consists of the 10 data points from the previous model in which the 3 wrongly classified minus(-) are weighted more so that the current model tries more to classify these minuses(-) correctly.

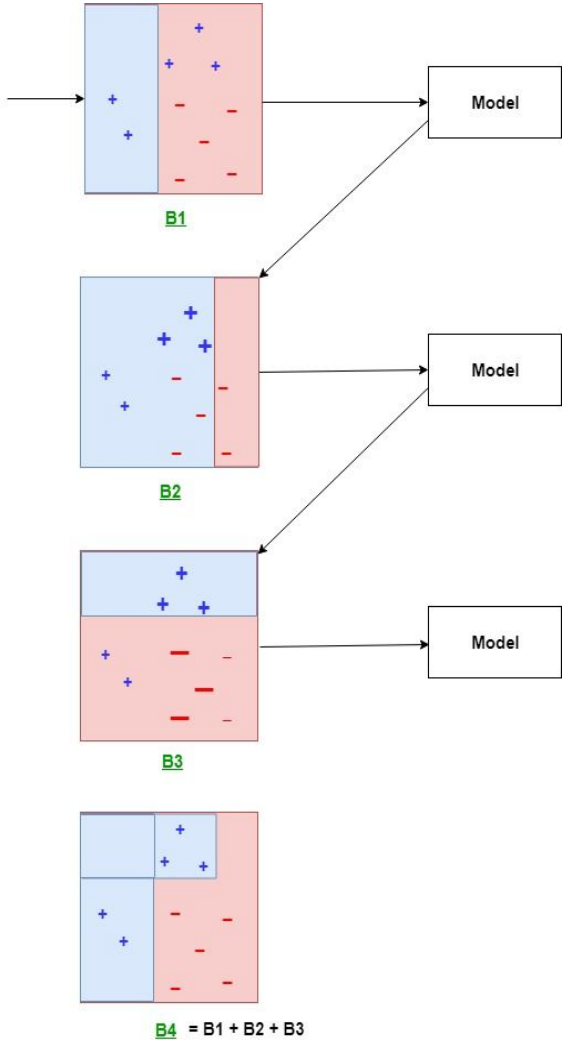
This model generates a horizontal separator line which correctly classifies the previously wrongly classified minuses(-).



Adaboost

B4 combines together B1, B2 and B3 in order to build a strong prediction model which is much better than any individual model used.

(Example from Geeksforgeeks)



Adaboost - how to find weak learners?

To compute the error rate of model Mi

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(\mathbf{X_j}), \tag{8.34}$$

where $err(\mathbf{X_j})$ is the misclassification error of tuple $\mathbf{X_j}$: If the tuple was misclassified, then $err(\mathbf{X_j})$ is 1; otherwise, it is 0. If the performance of classifier M_i is so poor that its error exceeds 0.5, then we abandon it. Instead, we try again by generating a new D_i training set, from which we derive a new M_i.

Adaboost

If a tuple in round i was correctly classified, its weight is multiplied by error(Mi)/(1 − error(Mi)).

Once the weights of all the correctly classified tuples are updated, the weights for all tuples (including the misclassified ones) are normalized so that their sum remains the same as it was before.

To normalize a weight, we multiply it by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified tuples are increased and the weights of correctly classified tuples are decreased

Adaboost- ensemble prediction

Adaboost assigns a weight to each classifier's vote, based on how well the classifier performed.

The lower a classifier's error rate, the more accurate it is, and therefore, the higher its weight for voting should be.

$$log \frac{1 - error(M_i)}{error(M_i)}.$$

For each class, c, we sum the weights of each classifier that assigned class c to X. The class with the highest sum is the "winner" and is returned as the class prediction for tuple X.

Algorithm: AdaBoost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of d class-labeled training tuples;
- k, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

Output: A composite model.

Method:

(1) initialize the weight of each tuple in D to $1/d$;
(2) **for** $i = 1$ to k **do** // for each round:
(3) sample D with replacement according to the tuple weights to obtain D_i;
(4) use training set D_i to derive a model, M_i;
(5) compute $error(M_i)$, the error rate of M_i (Eq. 8.34)
(6) **if** $error(M_i) > 0.5$ **then**
(7) go back to step 3 and try again;
(8) **endif**
(9) **for** each tuple in D_i that was correctly classified **do**
(10) multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights
(11) normalize the weight of each tuple;
(12) **endfor**

To use the ensemble to classify tuple, X:

(1) initialize weight of each class to 0;
(2) **for** $i = 1$ to k **do** // for each classifier:
(3) $w_i = log \frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4) $c = M_i(\boldsymbol{X})$; // get class prediction for $\boldsymbol{X}$ from M_i
(5) add w_i to weight for class c
(6) **endfor**
(7) return the class with the largest weight;

Summary

Boosting

Adaboost

Next:

Bagging and boosting - comparison

Examples

Random Forest

Lecture 8.3

Recap

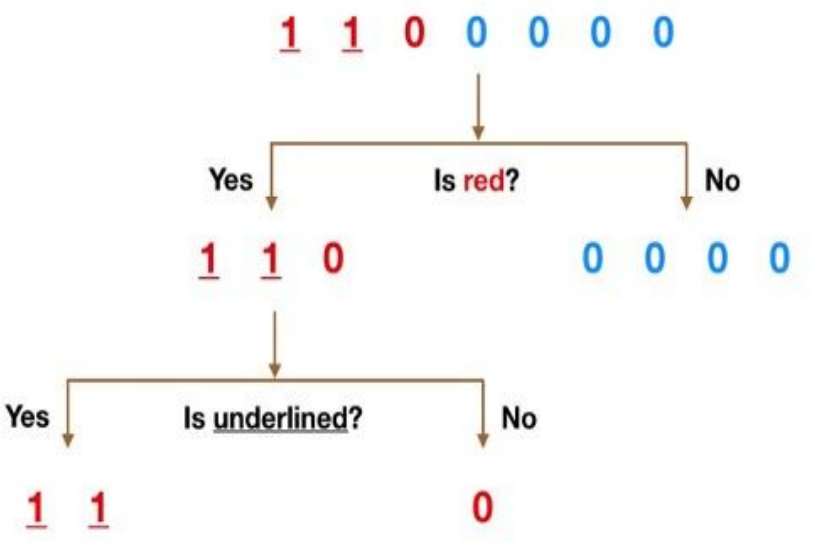
- Ensemble learning
- Bagging
- Boosting
- **Random Forest**

Random Forest

- Another ensemble method
- The **"forest"** it builds, is an ensemble of decision trees, usually trained with the **“bagging”** method.
- The general idea of the bagging method is that a combination of learning models increases the overall result.
- **Random forest** builds **multiple decision trees** and **merges them** together to get a more **accurate and stable prediction**.

Decision Trees are the building blocks for **Random Forest algorithm**

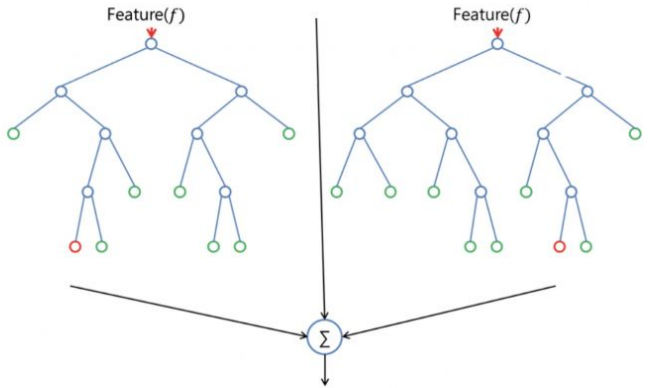
Decision Tree: Recap



- How to classify 1s and 0s
  - Color
  - Underline
- What feature will allow *me* to split the observations at hand in a way that the **resulting groups are as different from each other as possible** (and the members of each resulting subgroup are as similar to each other as possible)?

Random Forest

- Consists of a large number of individual decision trees that operate as an ensemble.
- Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction



https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Random Forest

- Imagine that each of the classifiers in the **ensemble is a decision tree classifier so that the collection of classifiers is a “forest”.**
- The individual decision trees are generated using a **random selection of attributes** at each node to determine the split
- Each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.
- During classification, each tree votes and the most popular class is returned.

Random Forest -working

- There are two stages in Random Forest algorithm
  - Random forest creation
  - Make a prediction from the random forest classifier
- Decisions trees are **very sensitive to the data they are trained on** — small changes to the training set can result in significantly different tree structures.
- Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement
  - **Bagging**

Random Forest -working

Random Forest creation

1. **Randomly select “K” features** from total “m” features where k << m
2. Among the “K” features, **calculate the node “d” using the best split point**
3. **Split the node into daughter nodes** using the best split
4. Repeat the 1-3 steps until “l” number of nodes has been reached
5. Build forest by repeating steps 1-4 for “n” number times to create “n” number of trees

Random Forest -working

Random Forest prediction

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target
3. Consider the high voted predicted target as the final prediction from the random forest algorithm

Random Forest- Advantages

1. For applications in classification problems, Random Forest algorithm will avoid the overfitting problem
2. For both classification and regression task, the same random forest algorithm can be used
3. The Random Forest algorithm can be used for identifying the most important features from the training dataset, in other words, feature engineering.

Suggested Reading

- https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674
- https://towardsdatascience.com/understanding-random-forest-58381e0602d2