Chapter 7: Ensemble Learning and Random Forests

Aly Khaled

August 15, 2021

1 What is Ensemble Learning?

- Ensemble Learning in using more than one predictor then it aggregate the prediction of each predictor and predict the class that gets most votes.
- **Ensemble Learning** works best when the predictors are **independent** form each other as possible and we can achieve this by **training them using very different algorithms**.
- **Soft voting** is to predict the class with the highest class probability averaged over all the individual **but this need all classifier to be able to estimate class probabilities which means that they all have a *predict_proba()* method**
- **Soft voting** is often achieve higher performance than **hard voting** because it gives more weight to highly confident votes
- When we train all the predictors we can make prediction for a new instance by simply aggregating the predictions of all predictors using:
  - **Statistical mode** for classification
  - **Average** for regression
- Each individual predictor has a higher bias than if it were trained on the original training set but **aggregation** reduces both bias and variance and the **net result** is that the ensemble has a similar bias but a lower variance than a single predictors trained on the original training set

2 Bagging VS Pasting

- There are two approaches for ensemble learning.
  - **First:** Use different training algorithms.
  - **Second:** Use the same training algorithm for every predictor and train them on different random subsets of the training set.
- There are two types of sampling:
  - **Bagging:** When sampling is performed with replacement.
  - **Pasting:** When sampling is performed without replacement.
- Only **Bagging** allows training instances to be sampled several times for the same predictor .

- In **Scikit–Learn** the **BaggingClassifier** class automatically performs soft voting instead of hard voting if the base classifier can estimate class probabilities which means that it has a *predict_proba()* method.
- **Bagging** ends up with a slightly **higher bias** than pasting; but the extra diversity also means that the predictors end up being **less correlated** so the ensemble's variance is reduced.
- **Bagging** often results in better models.
- Its recommended if you have spare time and CPU power to use cross validation to evaluate both of bagging and pasting and select the one that works best.
- In **Scikit–Learn** the **BaggingClassifier** class by default it samples m training instances with replacement (*bootstrap=True*) where m is the size of the training set. This means only about 63% of the training set are sampled and the remaining 37% are called *(out–of–bag)* instances, so we can evaluate the predictor on these instances without making validation set

  - To evaluate the predictor on these instances add (*obb_score=True*) parameter and access it through (*oob_score_*) variable.
- In **Scikit–Learn** the **BaggingClassifier** class can sample features too and sampling is controlled by two hyperparameters: *max_features* and *bootstrap_features* and sampling features is useful when dealing with high-dimensional inputs (such as images).
- Sampling both training instances and features is called **Random Patches Method** but keeping all training instances and sampling features is called **Random Subspaces Method**

3 Random Forests

- **Random Forests** is an ensembles of **Decision Trees** and it trained using **Bagging** method but sometimes by **Pasting**.
- You can make **Random Forests** by using **BaggingClassifier** class but it recommended to make it using **RandomForestClassifier** class (or by **RandomForestRegressor** class for regression) because it more convenient and optimized for **Decision Trees**.
- **Random Forests** algorithm introduces extra randomness when growing tree.
- Instead of searching for the very best features when splitting a node. it searches for the best features among a random subset of features.
- The algorithm results in greater tree diversity and making overall better model.

4 Boosting

5 Stacking