Machine Learning

| Out-of-Bag Evaluation - Random Patches and Random Subspaces

Prof. Dr. Mostafa Elhosseini

https://youtube.com/ElhosseiniAcademy

Agenda

- Out-of-Bag Evaluation
- Random Patches
- Random Subspaces

Out-of-Bag Evaluation

- With bagging, some training instances may be **sampled several times** for any given predictor, while **others may not be sampled at all**.

- By default a BaggingClassifier samples m training instances with replacement (bootstrap=True), where m is the size of the training set.

- With this process, it can be shown mathematically that only about **63%** of the training instances are sampled on average for each predictor

Out-of-Bag Evaluation

- Each instance has an equal chance of being selected: $1/m$
- P(not selected) $= \ 1 - 1/m$
- Probability of Not Being Selected in All m Draws: $(1 - 1/m)^m$
- As m becomes large $\log_{m \to \infty}(1 - 1/m)^m = \frac{1}{e} = 36.79\%$

- The remaining 37% of the training instances that are not sampled are called **out-of-bag (OOB) instances**.
- Note that they are not the same 37% for all predictors.

Out-of-Bag Evaluation

- A bagging ensemble can be evaluated using OOB instances, without the need for a separate validation set: indeed, if there are **enough estimators**, then each instance in the training set will likely be an OOB instance of several estimators, so these estimators can be used to make a fair ensemble prediction for that instance

Out-of-Bag Evaluation

- Drawing m times with replacement, the probability instance i is never chosen is: $\left(1 - \frac{1}{m}\right)^m$
- For large m, this probability is about $e^{-1} \cong 0.368$.
- In other words, each estimator leaves out any given instance about 36.8% of the time on average.
- The probability that a particular instance i is not left out by a given estimator (i.e., it appears at least once in that estimator's training set) is about $1 - 0.368 = 0.6321 - 0.368 = 0.632$.

Out-of-Bag Evaluation

- If we want the probability that this instance i appears in the training sets of all N estimators (thus never gets a chance to be OOB), we multiply that probability N times: $(0.632)^N$
- As N (the number of estimators) increases, $(0.632)^N$ gets smaller and smaller, approaching zero.
- This means that as you add more estimators, the chance that an instance is never OOB for any of them becomes virtually zero. Eventually, every instance is OOB for some substantial fraction of the estimators.

Out-of-Bag Evaluation

- set **oob_score=True** when creating a BaggingClassifier to request an automatic **OOB evaluation** after training

```
bag_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=500,
                            oob_score=True, n_jobs=-1, random_state=42)
bag_clf.fit(X_train, y_train)
bag_clf.oob_score_
```

```
0.896
```

- Let's verify this

```
from sklearn.metrics import accuracy_score

y_pred = bag_clf.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.92
```

Random Patches and Random Subspaces

- The **BaggingClassifier** class supports **sampling the features** as well. Sampling is controlled by two hyperparameters: **max_features** and **bootstrap_features**.

- They work the same way as **max_samples** and **bootstrap**, but for feature sampling instead of instance sampling. Thus, each predictor will be trained on a **random subset of the input features**.

Random Patches and Random Subspaces

- This technique is particularly useful when you are dealing with high dimensional inputs (such as images), as it can considerably **speed up training**

- This **reduces the computational cost**, particularly in cases where images are large, while still allowing the model to learn meaningful features.

Random Patches and Random Subspaces

- Sampling both training instances and features is called the **random patches method**

- **Keeping all training instances** (by setting bootstrap=False and max_samples=1.0) but sampling features (by setting bootstrap_features to True and/or max_features to a value smaller than 1.0) is called the random subspaces method

Random Patches and Random Subspaces

- Each individual week learner (such as Decision Tree) might become slightly more biased because it has fewer features to make decisions, meaning it can capture less of the data's complexity
- the ensemble (such as in Random Forest), will have lower variance because the trees are less likely to overfit. By using different subsets of features, the decision trees become less correlated with each other. This reduced correlation between the trees helps to reduce the overall variance when their predictions are averaged in ensemble methods, as the errors made by different trees tend to cancel each other out.

Random Patches and Random Subspaces

- Sampling features results in even more predictor diversity, trading a bit more bias for a lower variance.

Next Lecture

- Random forest is an ensemble of decision trees, generally trained via the bagging method (or sometimes pasting), typically with max_samples set to the size of the training set.

- **RandomForestClassifier** class, which is more convenient and **optimized** for decision trees (similarly, there is a **RandomForestRegressor** class for regression tasks).