Hyperparameter Tuning

Tushar B. Kute,
http://tusharkute.com

Hyperparameter tuning

- A machine learning model has two types of parameters.
  - The first type of parameters are the parameters that are learned through a machine learning model.
  - The second type of parameters are the hyper parameter that we pass to the machine learning model.
- Normally we randomly set the value for these hyper parameters and see what parameters result in best performance.
- However randomly selecting the parameters for the algorithm can be exhaustive.

Hyperparameter tuning

- These hyperparameters might address model design questions such as:
  - What degree of polynomial features should I use for my linear model?
  - What should be the maximum depth allowed for my decision tree?
  - What should be the minimum number of samples required at a leaf node in my decision tree?
  - How many trees should I include in my random forest?
  - How many neurons should I have in my neural network layer?
  - How many layers should I have in my neural network?
  - What should I set my learning rate to for gradient descent?

Hyperparameter tuning

Model-based learning

Use the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

To learn a set of parameters

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}$$

Which yield a **generalized** function

$$f(x;\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Capable of predicting values or classes on new input data

$$f(x_i;\theta) = 39$$

$$f(x_j;\theta) = 1$$

Hyperparameter tuning

- In general, this process includes:
  - Define a model
  - Define the range of possible values for all hyperparameters
  - Define a method for sampling hyperparameter values
  - Define an evaluative criteria to judge the model
  - Define a cross-validation method

Grid Search CV

- It is not easy to compare performance of different algorithms by randomly setting the hyper parameters because one algorithm may perform better than the other with different set of parameters. And if the parameters are changed, the algorithm may perform worse than the other algorithms.
- Therefore, instead of randomly selecting the values of the parameters, a better approach would be to develop an algorithm which automatically finds the best parameters for a particular model. Grid Search is one such algorithm.

Grid Search CV

- n_estimators = [10, 50, 100, 200]
- max_depth = [3, 10, 20, 40]
- RandomForestClassifier(n_estimators=10, max_depth=3)
- RandomForestClassifier(n_estimators=10, max_depth=10)
- RandomForestClassifier(n_estimators=10, max_depth=20)
- RandomForestClassifier(n_estimators=10, max_depth=40)
-
- RandomForestClassifier(n_estimators=50, max_depth=3)
- RandomForestClassifier(n_estimators=50, max_depth=10)
- RandomForestClassifier(n_estimators=50, max_depth=20)
- RandomForestClassifier(n_estimators=50, max_depth=40)
-
- RandomForestClassifier(n_estimators=100, max_depth=3)
- RandomForestClassifier(n_estimators=100, max_depth=10)
- RandomForestClassifier(n_estimators=100, max_depth=20)
- RandomForestClassifier(n_estimators=100, max_depth=40)
-
- RandomForestClassifier(n_estimators=200, max_depth=3)
- RandomForestClassifier(n_estimators=200, max_depth=10)
- RandomForestClassifier(n_estimators=200, max_depth=20)
- RandomForestClassifier(n_estimators=200, max_depth=40)

would yield the following models.

Example: Wisconsin Breast Cancer Dataset

	A	B	C	D	E	F	G	H
1	"id"	"diagnosis"	"radius_mean"	"texture_mean"	"perimeter_mean"	"area_mean"	"smoothness_mean"	"compactness_mean"
2	842302	"M"	17.99	10.38	122.8	1001	0.1184	0.2776
3	842517	"M"	20.57	17.77	132.9	1326	0.08474	0.07864
4	84300903	"M"	19.69	21.25	130	1203	0.1096	0.1599
5	84348301	"M"	11.42	20.38	77.58	386.1	0.1425	0.2839
6	84358402	"M"	20.29	14.34	135.1	1297	0.1003	0.1328
7	843786	"M"	12.45	15.7	82.57	477.1	0.1278	0.17
8	844359	"M"	18.25	19.98	119.6	1040	0.09463	0.109
9	84458202	"M"	13.71	20.83	90.2	577.9	0.1189	0.1645
10	844981	"M"	13	21.82	87.5	519.8	0.1273	0.1932
11	84501001	"M"	12.46	24.04	83.97	475.9	0.1186	0.2396
12	845636	"M"	16.02	23.24	102.7	797.8	0.08206	0.06669
13	84610002	"M"	15.78	17.89	103.6	781	0.0971	0.1292
14	846226	"M"	19.17	24.8	132.4	1123	0.0974	0.2458
15	846381	"M"	15.85	23.95	103.7	782.7	0.08401	0.1002
16	84667401	"M"	13.73	22.61	93.6	578.3	0.1131	0.2293
17	84799002	"M"	14.54	27.54	96.73	658.8	0.1139	0.1595
18	848406	"M"	14.68	20.13	94.74	684.5	0.09867	0.072
19	84862001	"M"	16.13	20.68	108.1	798.8	0.117	0.2022
20	849014	"M"	19.81	22.15	130	1260	0.09831	0.1027
21	8510426	"B"	13.54	14.36	87.46	566.3	0.09779	0.08129
22	8510653	"B"	13.08	15.71	85.63	520	0.1075	0.127
23	8510824	"B"	9.504	12.44	60.34	273.9	0.1024	0.06492
24	8511133	"M"	15.34	14.26	102.5	704.4	0.1073	0.2135
25	851509	"M"	21.16	23.04	137.2	1404	0.09428	0.1022
26	852552	"M"	16.65	21.38	110	904.6	0.1121	0.1457
27	852631	"M"	17.14	16.4	116	912.7	0.1186	0.2276

Read the dataset

```
import pandas as pd  # import libraries
import numpy as np
```

```
data = pd.read_csv('wisc_bc_data.csv')
# Read datasets
```

```
X = data.drop(['id','diagnosis'], axis=1)
y = data['diagnosis']
# Separate input and output variables
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
# Scale the dataset
```

Select the model

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, random_state = 0)
```

```
X_train.shape
```

```
(426, 30)
```

```
X_test.shape
```

```
(143, 30)
```

Train the model

```
from sklearn.neighbors import KNeighborsClassifier
```

```
classifier = KNeighborsClassifier(n_neighbors=20)
```

```
classifier.fit(X_train, y_train);
```

```
y_pred = classifier.predict(X_test);
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, y_pred) * 100
```

```
95.1048951048951
```

Check the accuracy here

Apply Grid Search CV

```
from sklearn.model_selection import GridSearchCV
```

```
params = {
    'n_neighbors' : [5, 8,12,20,24,30],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}
```

```
grid_kn = GridSearchCV(estimator = classifier,
                       param_grid = params,
                       scoring = 'accuracy',
                       cv = 5)
```

```
grid_kn.fit(X_train, y_train)
```

Train the model

```
print(grid_kn.best_estimator_)
# best classifier object
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski
',
                     metric_params=None, n_jobs=None, n_neighbors=5, p
=2,
                     weights='uniform')
```

```
best = grid_kn.best_estimator_
```

```
best.fit(X_train, y_train);
```

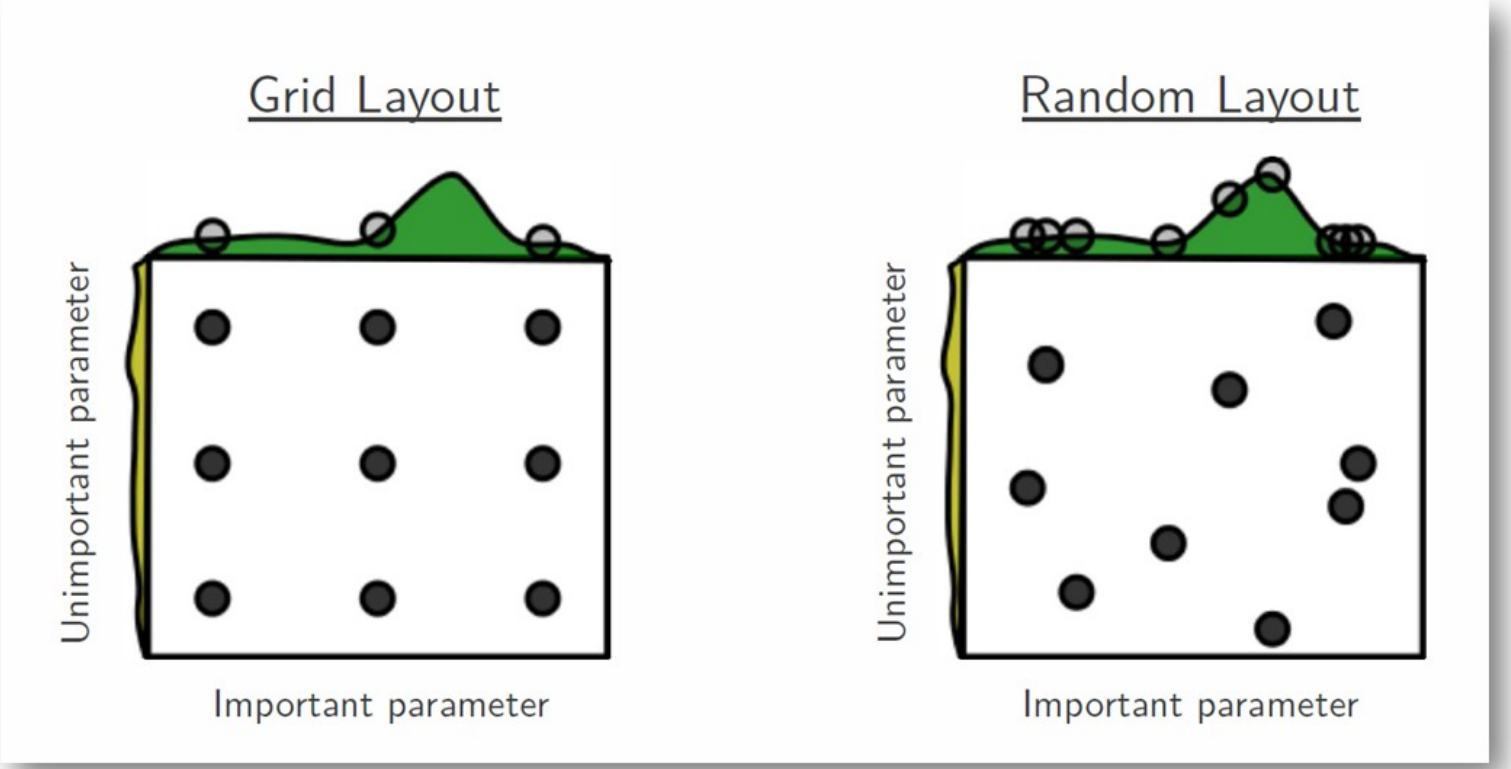
```
y_new = best.predict(X_test)
```

```
accuracy_score(y_test, y_new) * 100
```

```
97.2027972027972
```

Check the accuracy now

Random Search



Useful resources

- www.geeksforthegeeks.org
- www.scikit-learn.org
- www.towardsdatascience.com
- www.medium.com
- www.analyticsvidhya.com
- www.kaggle.com
- www.stephacking.com
- www.github.com

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com