Booklet4Code

August 8, 2020

[1]:
```
import numpy as np
import pandas as pd
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV

from sklearn.svm import SVC
```

1 SVM Booklet Teil 4

1.0.1 linear Kernel

$\langle x, x^{'} \rangle$

[ ]:
```
svm_linear = SVC(kernel="linear", C=0.1)
svm_linear.fit(X_train, y_train)

test_predictions = svm_linear.predict(X_test).round().astype(int)
print(accuracy_score(y_test, test_predictions))
mean_absolute_error(y_test, test_predictions)
```

1.0.2 Polynominal Kernel

$(\gamma \langle x, x^{'} \rangle + r)^d$

d : degree

r: coef0

[ ]:
```
svm_poly = SVC(kernel="poly", gamma=0.1, C=0.1)
svm_poly.fit(X_train, y_train)

test_predictions = svm_poly.predict(X_test).round().astype(int)
print(accuracy_score(y_test, test_predictions))
mean_absolute_error(y_test, test_predictions)
```

1.0.3 Gaussian RBF Kernel

$exp(-\gamma ||x - x^{'}||^2)$

γ: gamma (>0)

γ definiert den Einfluss der einzelenen Trainingsdaten Je größer γ ist, desto näher müssen andere Trainingsdatenpunkte sein, um einen Effekt zu haben $\rightarrow$ γ gibt invertiert den Einfluss-Radius der Datenpunkte an, die als Support Vectors bestimmt wurden (vom Modell).

[ ]:
```
svm_rbf = SVC(kernel="rbf", C=0.01)
svm_rbf.fit(X_train, y_train)

test_predictions = svm_rbf.predict(X_test).round().astype(int)
print(accuracy_score(y_test, test_predictions))
mean_absolute_error(y_test, test_predictions)
```

1.0.4 Sigmoid Kernel

$tanh(\gamma\langle x, x^{'}\rangle) + r)$

r: coef0

[ ]:
```
svm_sigmoid = SVC(kernel="sigmoid", C=0.01)
svm_sigmoid.fit(X_train, y_train)

test_predictions = svm_sigmoid.predict(X_test).round().astype(int)
print(accuracy_score(y_test, test_predictions))
mean_absolute_error(y_test, test_predictions)
```

1.0.5 precomputed Kernel

[ ]:
```
gram_train = np.dot(X_train, X_train.T)
gram_test = np.dot(X_test, X_train.T)

svm_precomputed = SVC(kernel="precomputed", C=0.01)

svm_precomputed.fit(gram_train, y_train)

test_predictions = svm_precomputed.predict(gram_test).round().astype(int)
print(accuracy_score(y_test, test_predictions))
mean_absolute_error(y_test, test_predictions)
```

Um ein 'gute' Modell zu trainieren, werden die Hyperparameter mit GridSearch bestimmt. Diese Parameterräume werden untersucht.

[ ]:
```
#Tamara
param_linear = {'C': [0.1, 1.0, 10.0, 100.0]}
param_poly = {'C': [0.1, 1.0, 10.0, 100.0],
              'gamma': ['scale', 'auto', 0.01, 1.0],
              'degree': [2.0, 3.0, 4.0, 5.0],
              'coef0': [0, 0.5, 1.0]}
param_rbf = {'C': [0.1, 1.0, 10.0, 100.0],
```

```
              'gamma': ['scale', 'auto', 0.01, 1.0]}
param_sigmoid = {'C': [0.1, 1.0, 10.0, 100.0],
              'gamma': ['scale', 'auto', 0.01, 1.0],
              'coef0': [0.0, 0.5, 1.0]}
grid = GridSearchCV(SVC(kernel='poly', C=1), {'coef0': [0.0, 0.5, 1.0]}, refit␣
 ↪= True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```