SVM_SignLanguage

May 5, 2020

Det første der skulle gøres var at hente dataen fra sign language datasættet, som er er opdelt i et træningsæt og test sæt i to seperate csv filer. Dette skulle herefter omdannes fra et pandas objekt til en numpy matrix for træningsdataet og et simpelt float array for labels for . Disse labels var var en kolumn i pandas array et og skulle derfor ekstraheres. Efter dette er gjort for både trænings- og testsættet var dataet klar til at blive præprocesseret, da ideen var at lave PCA på dataet, så det ikke tog så lang tid at træne. Nu var datet klar til at blive test med og uden PCA. Alt dette forabejde ses nedenfor.

```
In [1]: from sklearn.pipeline import Pipeline
        from sklearn.preprocessing import StandardScaler
        from sklearn.svm import SVC
        import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import LabelEncoder
        from sklearn.decomposition import PCA
        import time
        import matplotlib.pyplot as plt

        train = pd.read_csv('./SignLanguageData/sign_mnist_train.csv')
        test = pd.read_csv('./SignLanguageData/sign_mnist_test.csv')

        train = train.astype(float)
        test = test.astype(float)

        Y_train = train['label'].to_numpy()
        train.drop('label', axis = 1, inplace = True)

        Y_test = test['label'].to_numpy()
        test.drop('label', axis = 1, inplace = True)

        labelizer = LabelEncoder()
        Y_train = labelizer.fit_transform(Y_train)
        Y_test = labelizer.transform(Y_test)

        X_train = train.to_numpy()
        X_test = test.to_numpy()
```

```
        pipeline = Pipeline([
                    ("std_scale", StandardScaler()),
                    ("pca", PCA(.95)),
                ])


        X_train_transform = pipeline.fit_transform(X_train, Y_train)
        X_test_transform = pipeline.transform(X_test)
        plt.imshow(X_train[0].reshape(28,28))
```

```
Out[1]: <matplotlib.image.AxesImage at 0x7efc2444f3c8>
```

Koden nedenfor er kopieret fra aflevering 3 og bruges til at få en rapport fra gridsearch modellen om hvor godt den klare sig på træningsdataet samt valideringsdataet.

```
In [2]: from sklearn.metrics import classification_report, f1_score
        from time import time
        from sklearn.model_selection import GridSearchCV

        def SearchReport(model):

            def GetBestModelCTOR(model, best_params):
                def GetParams(best_params):
                    ret_str=""
                    for key in sorted(best_params):
                        value = best_params[key]
                        temp_str = "'" if str(type(value))=="<class 'str'>" else ""
                        if len(ret_str)>0:
                            ret_str += ','
                        ret_str += f'{key}={temp_str}{value}{temp_str}'
                    return ret_str
                try:
                    param_str = GetParams(best_params)
                    return type(model).__name__ + '(' + param_str + ')'
                except:
                    return "N/A(1)"

            print("\nBest model set found on train set:")
            print()
            print(f"\tbest parameters={model.best_params_}")
            print(f"\tbest '{model.scoring}' score={model.best_score_}")
            print(f"\tbest index={model.best_index_}")
            print()
            print(f"Best estimator CTOR:")
            print(f"\t{model.best_estimator_}")
            print()
            try:
```

```
            print(f"Grid scores ('{model.scoring}') on development set:")
            means = model.cv_results_['mean_test_score']
            stds  = model.cv_results_['std_test_score']
            i=0
            for mean, std, params in zip(means, stds, model.cv_results_['params']):
                print("\t[%2d]: %0.3f (+/-%0.03f) for %r" % (i, mean, std * 2, params))
                i += 1
        except:
            print("WARNING: the random search do not provide means/stds")

        assert "f1_micro"==str(model.scoring), f"come on, we need to fix the scoring to be a
        return f"best: data=SignLanguageMnist, score={model.best_score_:0.5f}, model={GetBes

    def ClassificationReport(model, X_test, y_test, target_names=None):
        assert X_test.shape[0]==y_test.shape[0]
        print("\nDetailed classification report:")
        print("\tThe model is trained on the full development set.")
        print("\tThe scores are computed on the full evaluation set.")
        print()
        y_true, y_pred = y_test, model.predict(X_test)
        print(classification_report(y_true, y_pred, target_names))
        print()

    def FullReport(model, X_test, y_test, t):
        print(f"SEARCH TIME: {t:0.2f} sec")
        beststr, bestmodel = SearchReport(model)
        ClassificationReport(model, X_test, y_test)
        print(f"CTOR for best model: {bestmodel}\n")
        print(f"{beststr}\n")
        return beststr, bestmodel
```

Der er lagt op til brugen af GridSearch og forskellige kombinationer af hyperparametrene gamma og C bliver testet, da de ud fra dokumentationen https://scikit-learn.org/stable/modules/svm.html skulle have den største effekt på modellens præcision. Kernel cache er sat til 1000mb, som ikke påvirker modellens præcision, men derimode træningshastigheden, hvilket også er specificeret i dokumentations "practical use" segment. Nedenfor i de næste 2 celler bliver modellen testet med og uden PCA, hvor tuning_parameters variablen er de valgte kobinationer af hyperparameter, der er brugt en logaritmisk interval, som der også er gjort i dette eksempel https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

```
In [ ]: import sys
        old_stdout = sys.stdout
        sys.stdout = open('output_SVM_fullDataset.txt', 'w')

        tuning_parameters = {
            'gamma':np.logspace(-5, -3, 3),
            'C': np.logspace(2, 4, 3),
            'cache_size': [1000]
```

```
}

CV=5
VERBOSE=0

start = time()
random_grid_tuned = GridSearchCV(SVC(), tuning_parameters, cv=CV, scoring='f1_micro', ve
random_grid_tuned.fit(X_train, Y_train)
t = time()-start


# Report result
# There will be a lot of different combinations and it will take longer for it to search
b0, m0= FullReport(random_grid_tuned , X_test, Y_test, t)

sys.stdout = old_stdout
```

SEARCH TIME: 16983.29 sec

Best model set found on train set:

```
best parameters={'C': 100.0, 'cache_size': 1000, 'gamma': 1e-05}
best 'f1_micro' score=0.9865234019304316
best index=0
```

Best estimator CTOR: SVC(C=100.0, cache_size=1000, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=1e-05, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

Grid scores ('f1_micro') on development set: [0]: 0.987 (+/-0.003) for {'C': 100.0, 'cache_size': 1000, 'gamma': 1e-05} [1]: 0.688 (+/-0.013) for {'C': 100.0, 'cache_size': 1000, 'gamma': 0.0001} [2]: 0.284 (+/-0.005) for {'C': 100.0, 'cache_size': 1000, 'gamma': 0.001} [3]: 0.987 (+/-0.003) for {'C': 1000.0, 'cache_size': 1000, 'gamma': 1e-05} [4]: 0.688 (+/-0.013) for {'C': 1000.0, 'cache_size': 1000, 'gamma': 0.0001} [5]: 0.284 (+/-0.005) for {'C': 1000.0, 'cache_size': 1000, 'gamma': 0.001} [6]: 0.987 (+/-0.003) for {'C': 10000.0, 'cache_size': 1000, 'gamma': 1e-05} [7]: 0.688 (+/-0.013) for {'C': 10000.0, 'cache_size': 1000, 'gamma': 0.0001} [8]: 0.284 (+/-0.005) for {'C': 10000.0, 'cache_size': 1000, 'gamma': 0.001}

Detailed classification report: The model is trained on the full development set. The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
0	1.00	0.67	0.80	331
1	1.00	0.43	0.60	432
2	1.00	0.46	0.63	310
3	1.00	0.39	0.56	245
4	1.00	0.32	0.48	498
5	1.00	0.71	0.83	247
6	1.00	0.32	0.48	348
7	1.00	0.24	0.39	436
8	1.00	0.03	0.07	288

9	1.00	0.15	0.27	331
10	1.00	0.26	0.41	209
11	1.00	0.07	0.13	394
12	0.96	0.08	0.14	291
13	1.00	0.07	0.13	246
14	1.00	0.14	0.24	347
15	0.03	1.00	0.06	164
16	0.72	0.48	0.57	144
17	0.99	0.37	0.54	246
18	0.97	0.36	0.52	248
19	0.62	0.27	0.37	266
20	1.00	0.16	0.28	346
21	0.46	0.16	0.23	206
22	0.58	0.10	0.17	267
23	1.00	0.23	0.38	332

micro avg 0.29 0.29 0.29 7172 macro avg 0.89 0.31 0.39 7172 weighted avg 0.92 0.29 0.39 7172

CTOR for best model: SVC(C=100.0, cache_size=1000, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=1e-05, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

best: data=SignLanguageMnist, score=0.98652, model=SVC(C=100.0,cache_size=1000,gamma=1e-05)

```
In [5]: import sys
        old_stdout = sys.stdout
        sys.stdout = open('output_SVM_with_PCA_newest.txt', 'w')

        tuning_parameters = {
            'gamma':np.logspace(-5, -3, 3),
            'C': np.logspace(2, 4, 3),
            'cache_size': [1000]
        }

        CV=5
        VERBOSE=0

        start = time()
        random_grid_tuned = GridSearchCV(SVC(), tuning_parameters, cv=CV, scoring='f1_micro', ve
        random_grid_tuned.fit(X_train_transform, Y_train)
        t = time()-start

        b0, m0= FullReport(random_grid_tuned , X_test_transform, Y_test, t)
        sys.stdout = old_stdout

        import pickle
        filename = 'svm_model.sav'
        pickle.dump(random_grid_tuned, open(filename, 'wb'))
```

SEARCH TIME: 342.99 sec

Best model set found on train set:

```
best parameters={'C': 100.0, 'cache_size': 1000, 'gamma': 0.0001}
best 'f1_micro' score=1.0
best index=1
```

Best estimator CTOR: SVC(C=100.0, cache_size=1000, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

Grid scores ('f1_micro') on development set: [0]: 0.991 (+/-0.005) for {'C': 100.0, 'cache_size': 1000, 'gamma': 1e-05} [1]: 1.000 (+/-0.000) for {'C': 100.0, 'cache_size': 1000, 'gamma': 0.0001} [2]: 1.000 (+/-0.000) for {'C': 100.0, 'cache_size': 1000, 'gamma': 0.001} [3]: 1.000 (+/-0.000) for {'C': 1000.0, 'cache_size': 1000, 'gamma': 1e-05} [4]: 1.000 (+/-0.000) for {'C': 1000.0, 'cache_size': 1000, 'gamma': 0.0001} [5]: 1.000 (+/-0.000) for {'C': 1000.0, 'cache_size': 1000, 'gamma': 0.001} [6]: 1.000 (+/-0.000) for {'C': 10000.0, 'cache_size': 1000, 'gamma': 1e-05} [7]: 1.000 (+/-0.000) for {'C': 10000.0, 'cache_size': 1000, 'gamma': 0.0001} [8]: 1.000 (+/-0.000) for {'C': 10000.0, 'cache_size': 1000, 'gamma': 0.001}

Detailed classification report: The model is trained on the full development set. The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
0	0.88	1.00	0.93	331
1	1.00	0.94	0.97	432
2	0.80	1.00	0.89	310
3	0.96	1.00	0.98	245
4	0.95	1.00	0.97	498
5	0.69	0.87	0.77	247
6	0.85	0.84	0.84	348
7	0.95	0.91	0.93	436
8	0.81	0.82	0.82	288
9	0.77	0.71	0.74	331
10	0.75	0.75	0.75	209
11	0.83	0.74	0.78	394
12	0.81	0.67	0.74	291
13	0.91	0.83	0.87	246
14	0.93	1.00	0.96	347
15	0.90	1.00	0.95	164
16	0.46	0.72	0.56	144
17	0.56	0.60	0.58	246
18	0.77	0.70	0.73	248
19	0.71	0.76	0.73	266
20	0.91	0.64	0.75	346
21	0.82	0.80	0.81	206
22	0.74	0.69	0.72	267
23	0.87	0.76	0.81	332

micro avg 0.83 0.83 0.83 7172 macro avg 0.82 0.82 0.82 7172 weighted avg 0.84 0.83 0.83 7172

CTOR for best model: SVC(C=100.0, cache_size=1000, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

best: data=SignLanguageMnist, score=1.00000, model=SVC(C=100.0,cache_size=1000,gamma=0.0001)

Uden PCA tog modellen 4 timer, og man kunne måske have fået en bedre f1_micro, hvis man havde forsøgt med en mindre C og gamma værdi, da man ser på resultatet for gridSearch ser at det er en gamma værdi på 100 og en C værdi på 1e-05, der giver det bedste resultat, og det er de to mindste værdier testet med GridSearch. Grunden til disse parameter intervallet var for stort skyldes forsøget med GridSearch med PCA gav bedre resultatter i dette interval, og gruppen troede resultatet ville blive nogenlunde det samme, og i hvert fald inden for samme parameter interval. Et ekstra forsøg blev lavet uden PCA og med forhåbentlig bedre hyperparametre.

```
In [ ]: import sys
        old_stdout = sys.stdout
        sys.stdout = open('output_SVM_with_other_parameters.txt', 'w')

        tuning_parameters = {
            'gamma':np.logspace(-7, -5, 3),
            'C': np.logspace(0,2, 3),
            'cache_size': [1000]
        }

        CV=5
        VERBOSE=0

        start = time()
        random_grid_tuned = GridSearchCV(SVC(), tuning_parameters, cv=CV, scoring='f1_micro', ve
        random_grid_tuned.fit(X_train, Y_train)
        t = time()-start


        # Report result
        # There will be a lot of different combinations and it will take longer for it to search
        b0, m0= FullReport(random_grid_tuned , X_test, Y_test, t)

        sys.stdout = old_stdout
```

SEARCH TIME: 6853.88 sec

Best model set found on train set:

```
best parameters={'C': 10.0, 'cache_size': 1000, 'gamma': 1e-07}
best 'f1_micro' score=1.0
best index=3
```

Best estimator CTOR: SVC(C=10.0, cache_size=1000, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=1e-07, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

Grid scores ('f1_micro') on development set: [0]: 0.969 (+/-0.005) for {'C': 1.0, 'cache_size': 1000, 'gamma': 1e-07} [1]: 1.000 (+/-0.000) for {'C': 1.0, 'cache_size': 1000, 'gamma': 1e-06} [2]:

0.986 (+/-0.003) for {'C': 1.0, 'cache_size': 1000, 'gamma': 1e-05} [3]: 1.000 (+/-0.000) for {'C': 10.0, 'cache_size': 1000, 'gamma': 1e-07} [4]: 1.000 (+/-0.000) for {'C': 10.0, 'cache_size': 1000, 'gamma': 1e-06} [5]: 0.987 (+/-0.003) for {'C': 10.0, 'cache_size': 1000, 'gamma': 1e-05} [6]: 1.000 (+/-0.000) for {'C': 100.0, 'cache_size': 1000, 'gamma': 1e-07} [7]: 1.000 (+/-0.000) for {'C': 100.0, 'cache_size': 1000, 'gamma': 1e-06} [8]: 0.987 (+/-0.003) for {'C': 100.0, 'cache_size': 1000, 'gamma': 1e-05}

Detailed classification report: The model is trained on the full development set. The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
0	0.89	1.00	0.94	331
1	1.00	0.96	0.98	432
2	0.83	0.99	0.90	310
3	0.96	1.00	0.98	245
4	0.93	1.00	0.96	498
5	0.80	0.83	0.81	247
6	0.89	0.87	0.88	348
7	0.95	0.90	0.92	436
8	0.72	0.84	0.78	288
9	0.61	0.46	0.52	331
10	0.81	0.82	0.81	209
11	0.81	0.69	0.75	394
12	0.88	0.65	0.75	291
13	0.92	0.78	0.84	246
14	1.00	1.00	1.00	347
15	0.90	0.99	0.94	164
16	0.26	0.52	0.35	144
17	0.63	0.58	0.60	246
18	0.85	0.68	0.76	248
19	0.50	0.63	0.56	266
20	0.77	0.62	0.69	346
21	0.59	0.69	0.63	206
22	0.74	0.81	0.77	267
23	0.88	0.71	0.79	332

micro avg 0.81 0.81 0.81 7172 macro avg 0.80 0.79 0.79 7172 weighted avg 0.82 0.81 0.81 7172

CTOR for best model: SVC(C=10.0, cache_size=1000, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=1e-07, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

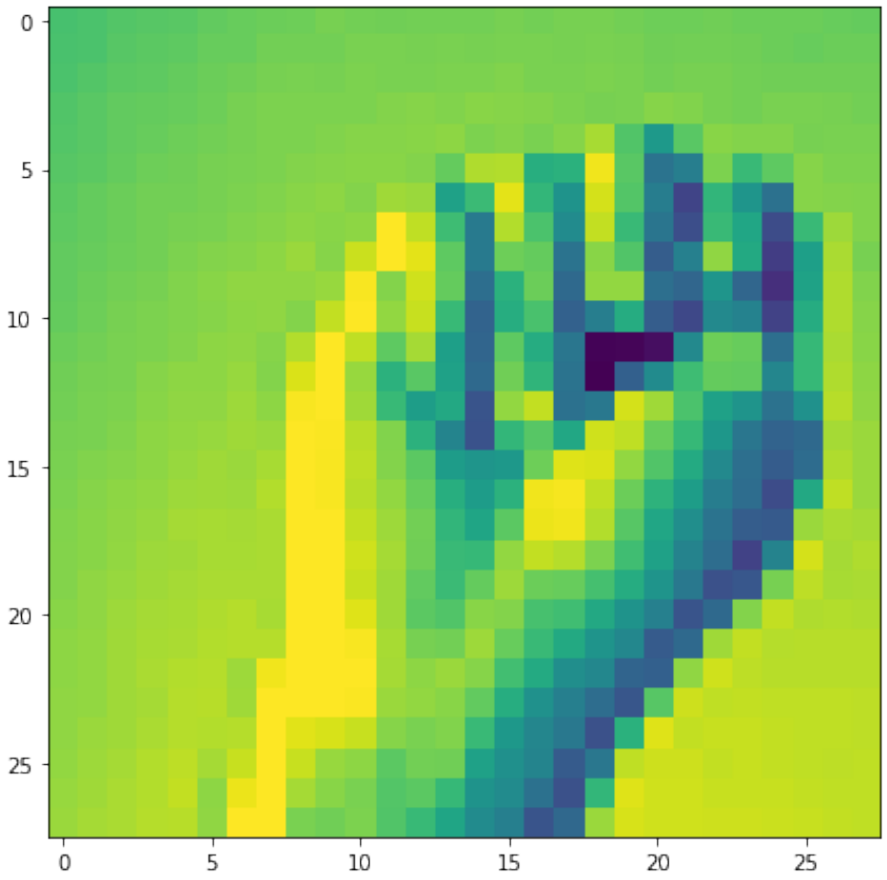
best: data=SignLanguageMnist, score=1.00000, model=SVC(C=10.0,cache_size=1000,gamma=1e-07)

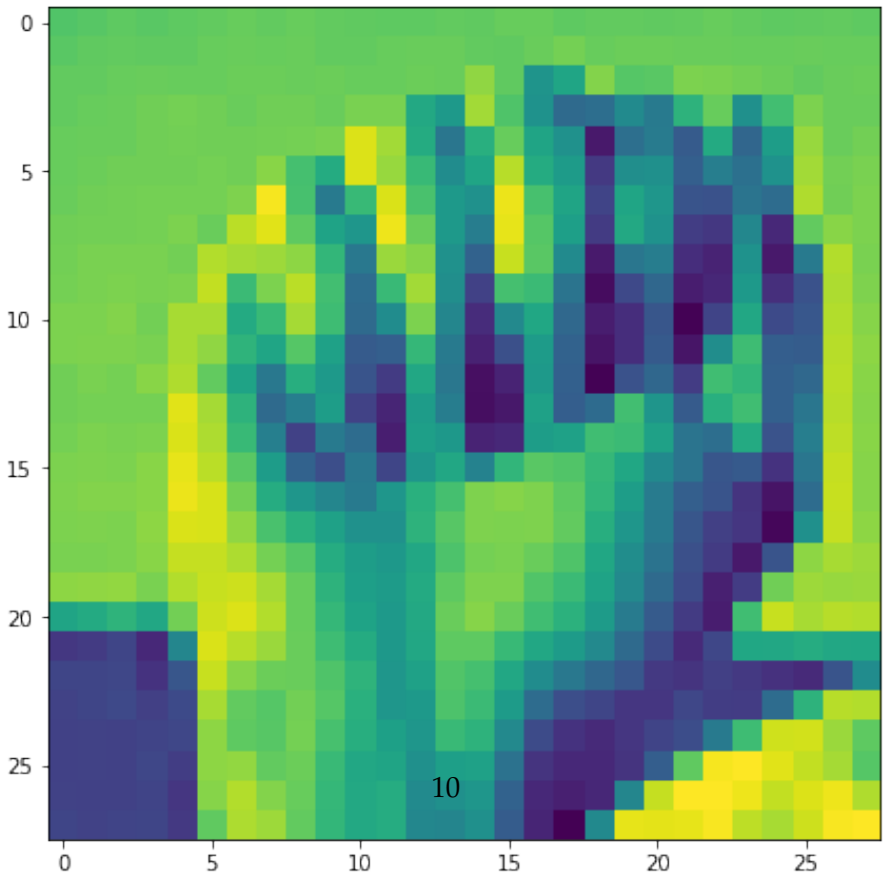
Denne model gav også en f1_micro score på 1.00 hvilket var overraskende. For PCA modellen blev der set på hvor præcis modellen var ved at se antallet af forkerte svar.

```
In [17]: import sys
         import string
         import pickle
         from sklearn.model_selection import GridSearchCV
```

```
filename = 'svm_model-Copy1.sav'
with open(filename, 'rb') as f:
       loaded_model = pickle.load(f)

Y_pred = loaded_model.predict(X_test_transform)

class_names = list(string.ascii_lowercase)
class_names.remove('j')
class_names.remove('z')

print("Number of wrong predictions: " + str(Y_test[Y_pred != Y_test].size))
wrong_pred_indices = Y_pred != Y_test
X_wrong_predict = X_test[wrong_pred_indices][:3]
Y_test_actual = Y_test[wrong_pred_indices][:3]
Y_pred_wrong = Y_pred[wrong_pred_indices][:3]

fig, ax = plt.subplots(2, 1, figsize=(18, 16))
print("Example of wrong prediction")
ax[0].imshow(X_wrong_predict[0].reshape((28,28)))
print("The real value: " + class_names[Y_test_actual[0]])
ax[1].imshow(X_test[Y_pred_wrong[0]==Y_test][0].reshape((28,28)))
print("The predicted value: " + class_names[Y_pred_wrong[0]])
plt.show()
```

```
Number of wrong predictions: 1204
Example of wrong prediction
The real value: n
The predicted value: a
```


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25

Der er mange forkerte svar, så modellen er slet ikke så præcis, som udregnet med GridSearch, og det er svært at sige hvorfor den giver et forkert resultat, men det er tydeligvis ikke 1.00. Det er også meget svært at se forskel på de to billeder ovenfor hvor det øverste er n og nederste et a.