Lab1

August 31, 2021

0.1 1. MNIST Dataset

[1]:
```
import numpy as np
import os

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# For Stochastic gradient descent classifier
from sklearn.linear_model import SGDClassifier

# to make this notebook's output stable across runs
np.random.seed(42)
```

[2]:
```
# To fetch the MNIST dataset
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784',version=1)
mnist.keys()
```

[2]:
```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names',
'target_names', 'DESCR', 'details', 'url'])
```

[4]:
```
# Set up the feature data and labels
X = mnist["data"]
y = mnist["target"]
```

[5]:
```
# Check the dimensions of X and y
print(X.shape)
print(y.shape)
```

```
(70000, 784)
(70000,)
```

[6]:
```
# Display an image
some_digit = X[0]
some_digit_image = some_digit.reshape(28,28)
plt.imshow(some_digit_image,cmap = "binary")
plt.axis("off")
plt.show()
```

[7]:
```
# Using `fetch_openml()` to download MNIST, returns the labels as strings.
y[0]
# Most ML algorithms expect numbers so cast y to integer.
y = y.astype(np.uint8)
```

0.1.1 Prepare data for a Machine Learning task

[8]:
```
# Prepare training and testing data
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]

shuffle_index = np.random.permutation(60000)
X_train, y_train = X_train[shuffle_index], y_train[shuffle_index]
```

0.2 2. Binary Classifier

0.2.1 2.1 Prepare classifier

[9]:
```
# Distinguish between two classes, 5 and not-5. Create the target vectors
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
```

[10]:
```
# Pick the Stochastic gradient descent classifier and train it to distingusih␣
↪the two classes
sgd_clf = SGDClassifier(max_iter=5, tol=-np.infty, random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

[10]:
```
SGDClassifier(max_iter=5, random_state=42, tol=-inf)
```

0.2.2 2.2 Evaluate performance

0.2.3 Cross-validation

[11]:
```
# Measure accuracy using cross validation
from sklearn.model_selection import cross_val_score
cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

[11]:
```
array([0.964 , 0.9579, 0.9571])
```

0.2.4 Confusion matrix

[13]:
```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
from sklearn.metrics import confusion_matrix
# Construct the confusion matrix
confusion_matrix(y_train_5, y_train_pred)
```

[13]:
```
array([[54058,   521],
       [ 1899,  3522]], dtype=int64)
```

0.2.5 Precision, recall, F1

[14]:
```
# Calculate precision, recall, F1 score
from sklearn.metrics import precision_score, recall_score, f1_score
print(precision_score(y_train_5, y_train_pred))
print(recall_score(y_train_5, y_train_pred))
print(f1_score(y_train_5, y_train_pred))
```

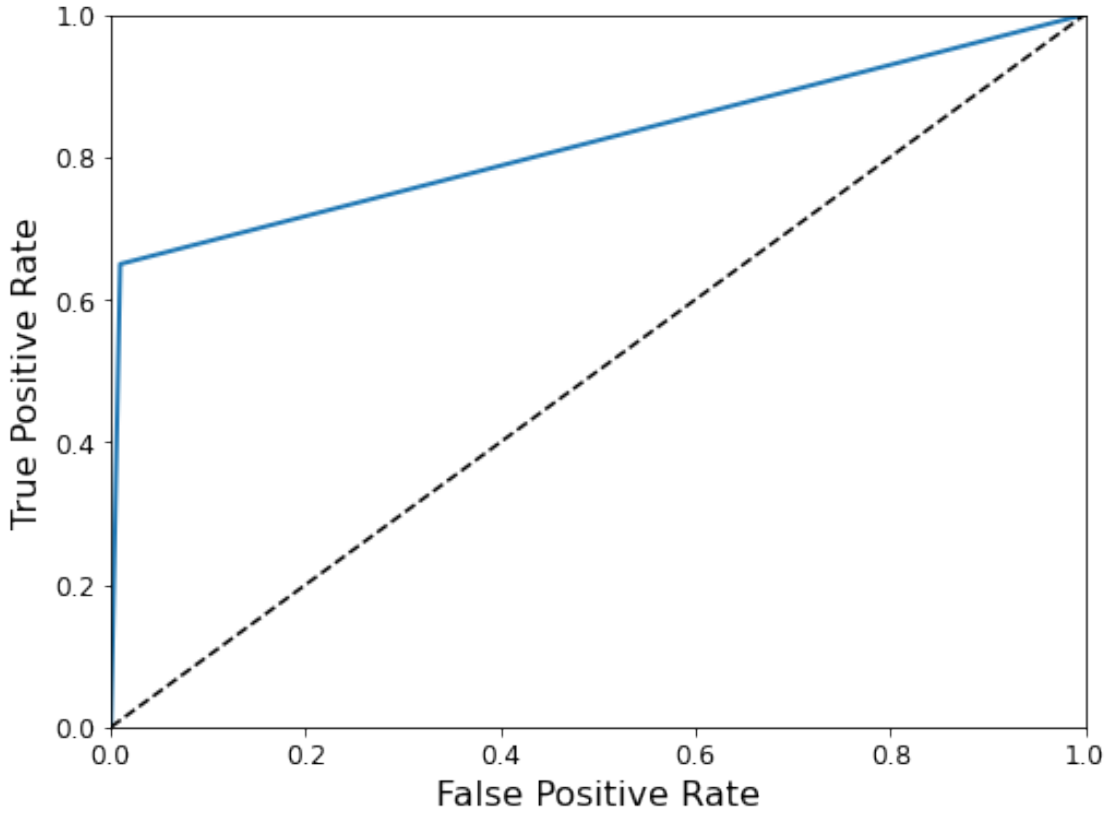
```
0.8711352955725946
0.6496956281128943
0.7442941673710904
```

0.2.6 ROC curve

[15]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_train_5, y_train_pred)
```

[16]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate', fontsize=16)
    plt.ylabel('True Positive Rate', fontsize=16)

plt.figure(figsize=(8, 6))
plot_roc_curve(fpr, tpr)
plt.show()
```



0.3 2.3 RandomForestClassifier

[17]:
```
from sklearn.ensemble import RandomForestClassifier
forest_clf = RandomForestClassifier(n_estimators=10, random_state=42)
y_probas_forest = cross_val_predict(forest_clf, X_train, y_train_5, cv=3,␣
 ↪method="predict_proba")
```

[18]:
```
y_scores_forest = y_probas_forest[:, 1] # score = proba of positive class
fpr_forest, tpr_forest, thresholds_forest = roc_curve(y_train_5,y_scores_forest)
```

[19]:
```
# Compare the ROC curve generated by the RandomForestClassifier with the ROC␣
 ↪curve geenrated by the SGDClassifier
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, "b:", linewidth=2, label="SGD")
plot_roc_curve(fpr_forest, tpr_forest, "Random Forest")
plt.legend(loc="lower right", fontsize=16)
plt.show()
```