AppliedMLMidterm

November 16, 2020

0.1 Intro

In this class, we have worked to understand both general principles of machine learning methods and specific details of particular machine learning techniques. For this project, you will apply what you have learned in this course to formulate and answer your own question, using ML. The midterm will be good practice for the final project, which will be similar.

Important: Be sure to include code and answers in the correct cells of the notebook. Otherwise we might miss your answers.

Please use this notebook to turn in your work.

Points: 20

You will be graded based on: 1. **Technical completeness**. *(12 points)*

Did you meet the technical requirements for the project? For instance, did you describe hyperparameter tuning and include plots, where necessary.

2. **Creativity and imagination**. *(3 points)*

Did you form an interesting and creative question, and explain why it is important to answer that question?

3. **Presentation**. *(5 points)*

Did you do a good job presenting your results? Would your notebook make sense to a person who was not familiar with your project? You should take time to write clearly, simplify your code and explain what you are doing in your notebook. Make sure your plots are well-labeled and appropriately scaled.

0.2 Question

Using this cell, please write a short, clear paragraph explaining what question you plan to answer in this notebook. Your question can be narrow (e.g. can we predict a dog's height from its weight) or broad (e.g. what features are important or unimportant in predicting the price of a house). Briefly describe why your question is important and how you plan to answer. Be sure to ask a question you can actually answer using machine learning techniques!

Can machine learning correctly classify handwritten characters with high accuracy?

1 Data (writing)

Using this cell, please write a short, clear paragraph explaining what data you will use to answer your question. You do not need to go gather custom datasets for this class, although you are welcome to do so. Just downloading data from Kaggle is fine, although we encourage you to think a little harder and more creatively when you do the project. There are many, many places to find interesting datasets online related to many topics like music, politics, sports, transportation, etc. Data gathering is one way to make your project more creative, but you are not expected or encouraged to take on a major data gathering effort. If applicable, describe how you plan to split between the training and test sets.

For this project I plan to use the digits dataset that is included in scikit-learn. This dataset includes 1,797 instances of 8x8 pixel images of handwritten numbers. Each instance contains 64 features where each feature represents the color intensity of a single pixel on a 0 to 16 scale. I plan to separate the data into training and test sets using a 70%-30% split, respectively. To separate the data, I will use the included train_test_split function inlcuded in scikit-learn. The train_test_split fucntion shuffles the dataset and randomly splits the set according to pre-defined proportions.

[1]:
```
## Code for data preprocessing

# Include your code to load, clean and split data in this cell. You must␣
↪complete this step in the project.

from sklearn.datasets import load_digits
digits = load_digits()
x = digits.data
y = digits.target

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.7)
```

1.1 Model selection and tuning

Using this cell, please write a short, clear paragraph explaining how you selected and tuned your model for the project. You must answer the following two questions in this cell (1) Why is your model an appropriate choice for your data? (2) What hyperparameters does your model have and how did you select them? (3) What features did you choose and why? You must engineer at least 1 feature

I will use a SVM implementation for this project because SMVs are good for classification problems by transforming the feature vectors and establishing a decision boundary based on the support vectors. The main hyperparameters of the SVM model are C which impacts the regularization penalty and kernel which determines the kernel type that transforms the data. I chose to tune both of these hyperparameters because the features of my data are in a relatively high dimension and I would like to see how various C and kernel values effect the accuracy of the model.

[100]:
```
## Code for model selection and tuning + feature engineering
```

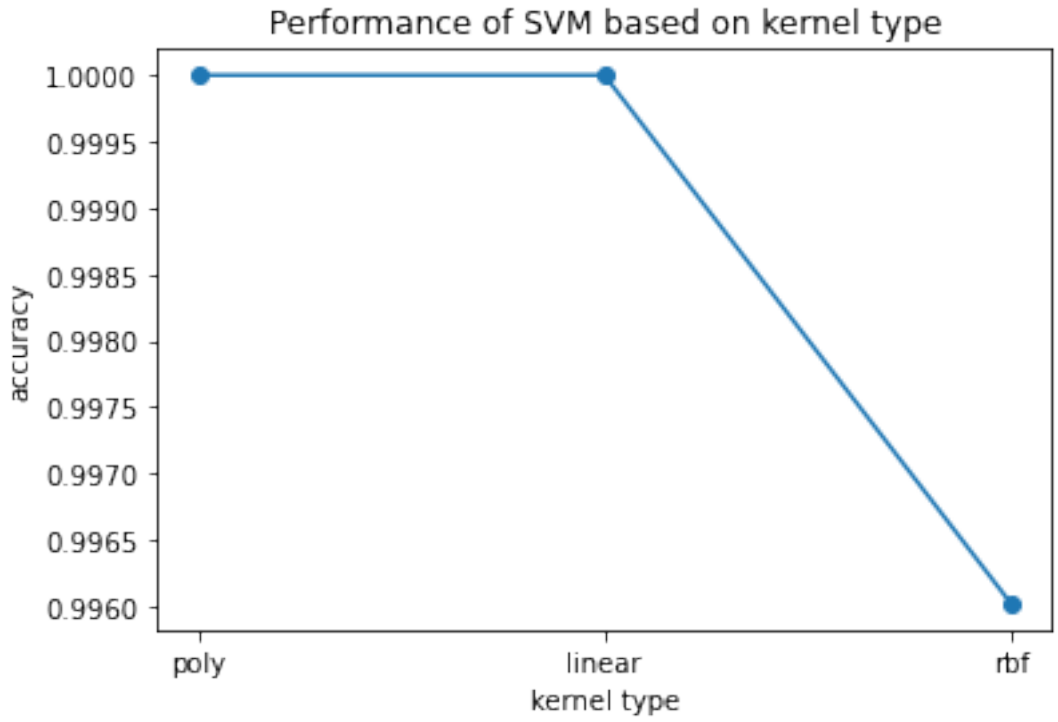
```
# Include any code to select and tune your model in this cell. You must␣
 ↪complete this step in the project.
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score

c = [0.01, 0.1, 1, 10, 100]
ker = ["poly", "linear", "rbf"]

ker_acc = []
c_acc = []

best = [0, ""]
for k in ker:
    svm = SVC(kernel = k)
    svm.fit(x_train, y_train)
    y_svm = svm.predict(x_train)
    acc = accuracy_score(y_train, y_svm)
    ker_acc.append(acc)
    print(k, ": ", acc)

print("\n")
for i in c:
    svm = SVC(C = i)
    svm.fit(x_train, y_train)
    y_svm = svm.predict(x_train)
    acc = accuracy_score(y_train, y_svm)
    c_acc.append(acc)
    print(i, ": ", acc)
```

```
poly :  1.0
linear :  1.0
rbf :  0.9960222752585521


0.01 :  0.21161495624502785
0.1 :  0.958631662688942
1 :  0.9960222752585521
10 :  1.0
100 :  1.0
```

[94]:
```
## Plot or table
# Include a plot or table explaining how you selected and tuned your model. You␣
 ↪must complete this step in the project.
import matplotlib.pyplot as plt
import numpy as np
```

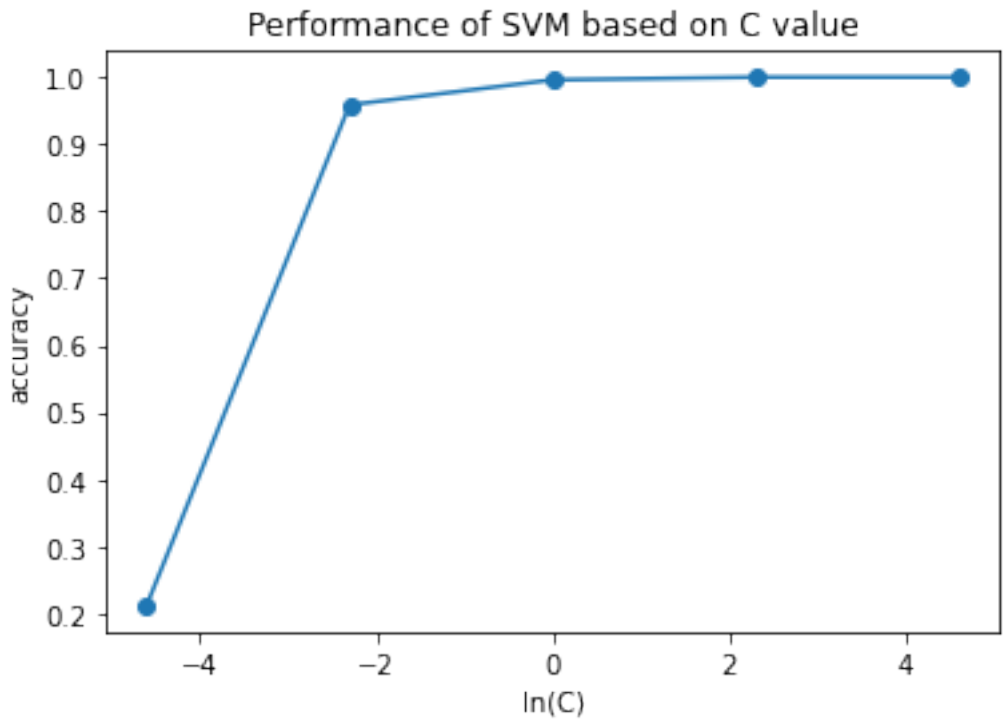
```
plt.scatter(ker, ker_acc)
plt.plot(ker, ker_acc)
plt.title("Performance of SVM based on kernel type")
plt.ylabel("accuracy")
plt.xlabel("kernel type")
```

[94]: Text(0.5, 0, 'kernel type')



[93]:
```
plt.scatter(np.log(c), c_acc)
plt.plot(np.log(c), c_acc)
plt.title("Performance of SVM based on C value")
plt.ylabel("accuracy")
plt.xlabel("ln(C)")
```

[93]: Text(0.5, 0, 'ln(C)')



1.2 Results

Using this cell, please write a short, clear paragraph explaining your results. In this class, we have mostly focused on accuracy. It is OK to measure your results in another quantitative way (e.g. precision or likelihood). Whatever you pick, make sure you are clear on what you are doing, and make sure you explain why your measurement of success makes sense.

Here I used two metrics to determine the performance of my SVM model, accuracy and cross validation score. Both metrics returned very high scores. The acuracy was nearly 99% while the leave one out cross validation was over 97%. This greatly exceeded my expectations. I had anticipated that the accuracy would be in the mid-80s, not nearly perfect. The SVM model performs very well with the handwritten dataset.

[92]:
```
## Code

# Include code showing how you arrived at your results. You must complete this␣
↪step in the project.

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import LeaveOneOut

svm = SVC(C = 10, kernel = "poly")

svm.fit(x_train, y_train)
```

```
y_predict = svm.predict(x_test)
scores = cross_val_score(svm, x_test, y_test, cv=LeaveOneOut())
```

[91]:
```
## Plot or table

# Include a plot or table explaining your results. You must complete this step␣
 ↪in the project.
print("                Model Performance                ")
print("-------------------------------------------------")
print("Accuracy:", accuracy_score(y_predict, y_test))
print("Leave one out cross validation:", scores.mean())
```

```
                Model Performance
-------------------------------------------------
Accuracy: 0.987037037037037
Leave one out cross validation: 0.9740740740740741
```

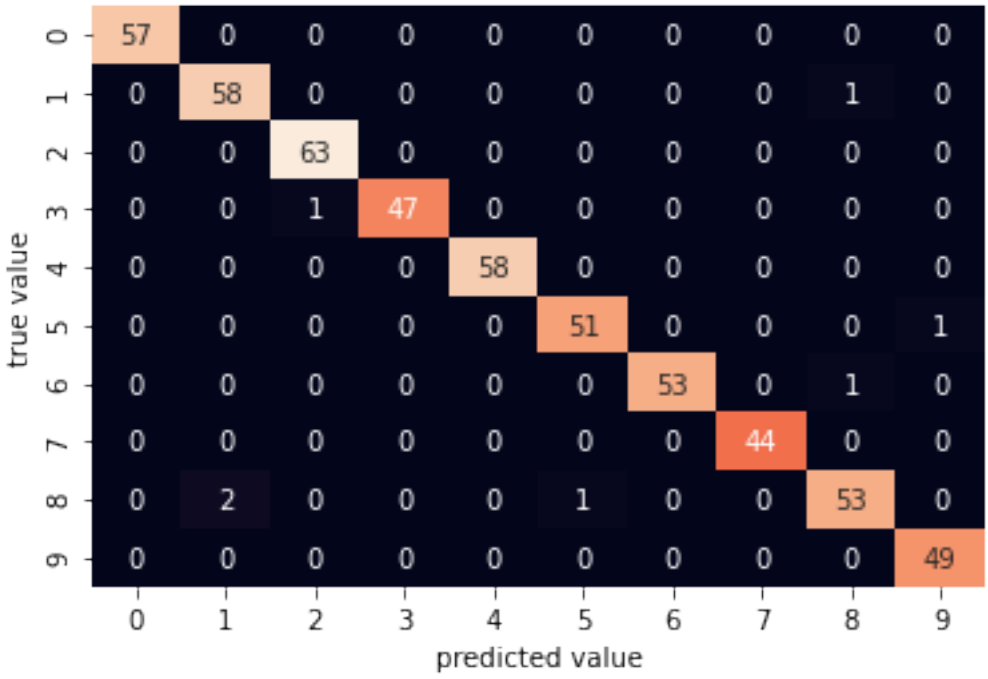
1.3 Error analysis

Using this cell, please write a short, clear paragraph explaining what errors your model seems to be making, and offer a brief explanation based on your code below.

The following confusion matrix shows that SMV is a very accurate classifier for handwritten numbers. The most any numbers were misclassified was once. Only a single '1', '3', '5', '6', and '8' were misclassified. I suspect if the data were in higher dimensions, such as 64x64 images, the model would perform even better than it did here.

[80]:
```
## Error analysis

# Include code for error analysis here, to justify your conclusions.
# You might make a confusion matrix, sample misclassifier data, analyze learned␣
 ↪weights, or use any other method
# discussed in class, or which makes sense for your model

from sklearn.metrics import confusion_matrix
import seaborn as sns

mat = confusion_matrix(y_test, y_predict)

sns.heatmap(mat, annot = True, cbar = False)
plt.ylabel("true value")
plt.xlabel("predicted value")
```

[80]:
```
Text(0.5, 15.0, 'predicted value')
```



1.4 Discussion

Using this cell, please write a short, clear paragraph describing how your results answer or do not answer your question. What new questions arise from your work?

I found that SVMs are highly accurate when predicting handwritten numbers. This shows that machine learning can be efficient in handwritting analysis. While I used SVM, another questions that arises from this project is how other machine learning models perform with this dataset. Are SVMs the best classifier for handwritten digits, or does another model perform even better?

[ ]: