George Mason University

Learning From Data Fall 2024

Computer Exercise #3

Assigned: September 11, 2024
Due Date: September 19, 2024

This computer exercise is concerned with Support Vector Machines (SVMs). As with previous exercises, most of the code given in this assignment is available in a Jupyter notebook that is distributed along with this exercise.

The due date for this exercise is September 19, 2024 at midnight. Absolutely no late submissions will be accepted after this date. It is important for you to finish this assignment on time and move on to the next. If you have not finished, submit the work you have done for partial credit.

In this assignment, there are a number of exercises and experiments to run to help you learn about SVMs. Do not view this assignment as tightly scripted - feel free to explore new ideas and perform additional experiments. A few ideas are given in the assignement.

A second Jupyter notebook, `cset_3_report.ipynb` is provided for the submission of your write-up. As in the first two exercises, it is not necessary to duplicate the code given to you in this report, but any new code or code you feel is important to describe or document your work is welcome. Also, only provide plots and graphs that are important. Do not pad your report with unnecessary code, figures, and plots that do not have any value. Points will be deducted for reports that are not concise and not well-documentd.

Reminders

1. See previous assignment on how to include some plots/figures in your report without attaching all the code necessary to generate them.
2. Set `random_state` to some number if you want repeatable results.
3. Periodically restart your kernel. when things do not look right or when you make significant changes to your code, as some parameters or variables set on previous runs or in other cells may affect your results

---

Your opportunity to gain a good, solid understanding of ML is through these exercises.

Computer Exercise 3.1 (Experiments with Support Vector Machines):

Although their popularity has declined in recent years due to the rise of deep learning and other more scalable or specialized algorithms, support vector machines are a foundational technique in machine learning, They remain, however, a powerful and versatile machine learning model, capable of performing linear or nonlinear classification, regression, and outlier detection, and they are used in applications where the number if features is large relative to the size of the dataset.

1. Getting Started

Load the following libraries, classes and functions that will be used in this exercise.,

```
#Common imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
```

Read the documentation on the `SVC` class and become familiar with the paramters, the algorithm that is used to find the maximum-margin classifier, and the **methods** that are available, such as `fit`, `score`, and `predict`, as well as others. There are a number of attributes in the SVC class that may be useful, including the following:

Some Key Attributes

```
support_          # The indices of the support vectors
support_vectors_  # The support vectors
n_support_        # Number of support vectors for each class
dual_coef_        # SV coefficients in the decision function
```

Questions

1. How does the `SVC` class find the maximum-margin classifier?
2. There are two other classes in scikit-*learn* that may be used to find a maximum-margin classifier: `LinearSVC` and `SGDClassifer`. How are these different from `SVC`, how do they find the maximum-margin classifier, and when would you use these instead of the `SVC` class?

2. Warm-up

To get some experience with support vector machines, we begin with a toy dataset using the `make_classification` class from the `sklearn.datasets` library. Generate a data set of 200 samples as follows:

```
from sklearn.datasets import make_classification
X, y = make_classification(n_features=2, n_samples=200,n_redundant=0, n_informative=2,
          n_clusters_per_class=2,class_sep=1,random_state=xxx)
```

where `xxx` are the first three numbers of your **GMU G-number**. For the parameters set in `make_classification` above, there are two classes with target values $y = 0, 1$, and each class

consists of two Gaussian clusters. Each cluster is located at one of the four corners of a square at $[\pm 1, \pm 1]$, and features are drawn independently from $N(0, 1)$ and then randomly linearly combined within each cluster to make the features correlated. Read the documentation so that you understand how this class generates datasets.

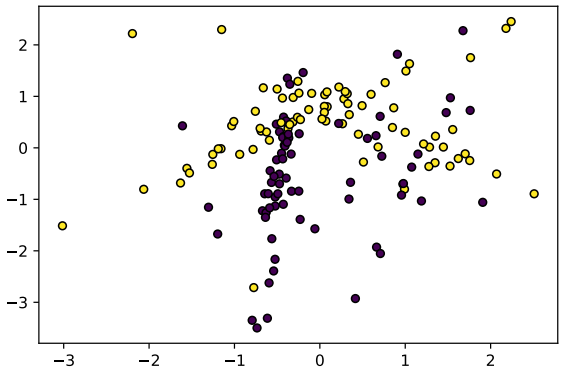
The next step is to separate the data into a training set and a test set,

```
X_train,X_test,y_train,y_test=train_test_split(X, y)
```

Having created a dataset, it may be useful to make a scatter plot of the training set data to get a feeling for what the data looks like. Note that for data sets that have many features, such scatter plots are not as useful. A scatter plot may be created easily as follows,

```
plt.scatter(X_train[:,0],X_train[:,1],marker="o",c=y_train,s=25,edgecolor="k")
```

Depending on the random number seed used, you may have a plot similar to the one shown below (or it could be quite a bit different):



Experiments

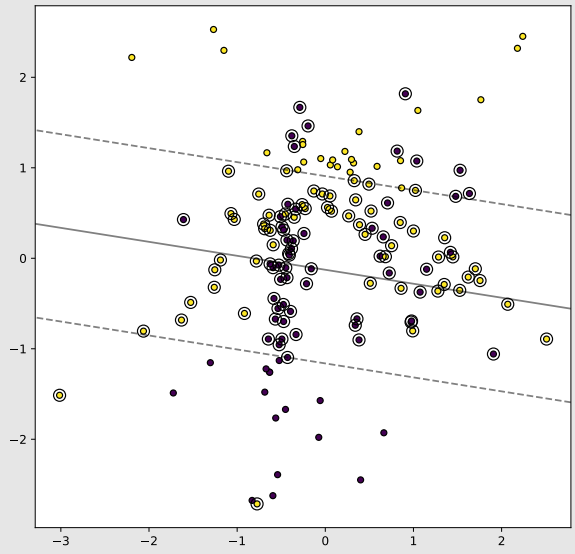
(a) Design a soft-margin linear support vector classifier using the default value of $C = 1$,

```
clf = SVC(kernel='linear')  # Default value of C=1
clf.fit(X_train, y_train)
```

Remember that the larger the value of C, the narrower the margin and the fewer margin violations there will be.

(b) What is the accuracy of your classifier on the training set and the test set? Note that the `score` method may be used to find the accuracy on a data set `(X,y)` using the command `clf.score(X,y)`. You may want to use, however, the training set or the test set and not the entire dataset.

(c) Make a plot of the decision boundary, the margin, and the support vectors (The code to create these plots is included in your Jupyter notebook as a function called `Plot_SVM`). For a linear classifier, you might get a plot like the one shown in the figure below, where the decision boundary is indicated by the solid line, the margins by the dashed lines, and the support vectors marked with a circle.



Following are a number of questions to be answered. For some, you will need to look at the documentation of SVC to find the answer, while for others you will need to look either at the attributes of the `SVC` class or use one of the methods. For each question, explain where your answer came from.

Questions

(a) How many support vectors are there for each class? How does this compare to the number of support vectors you would expect if the data was linearly separable?

(b) Does changing the value of C have much effect on the performance of your classifier?

(c) Print out the dual coefficients of your classifier using `clf.dual_coef_`. You will get something that looks like the following set of values.

```
array([[-1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -0.24467375, -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
         1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         1.        ,  0.68914148,  1.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         0.55553227,  1.        ,  1.        ,  1.        ,  1.        ]])
```

How do you interpret these numbers? In other words, what does it mean if a value is $+1$ or -1, or if it is in the interval $(0, 1)$? Why are there no values greater than ± 1?

(d) How are these dual coefficients related to the discriminant function

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

3. Nonlinear SVMs

As with any linear classifier, an SVM with a linear kernel may not work very well on datasets that are not linearly separable, and better performance can be expected with a nonlinear SVM, if done with caution. In the following exercise, you will look at nonlinear classifiers, beginning with polynomial kernels.

Experiments

(a) Repeat the experiments in the previous exercise using a polynomial kernel by setting `kernel='poly'` in the `SVC` class. The default value is a polynomial of degree three, which is a good starting point. The other important parameter is C, which has a default value of one.

(b) Evaluate the performance of your classifier on the training and test sets, and make a plot of the decision surface. Discuss what you observe.

(c) How many support vectors are there for each class? Compare this to what you found with a linear soft-margin SVM.

(d) How does the performance of this classifier compare to an SVM with a linear kernel?

(e) Repeat for a few other polynomial orders and discuss your findings.

Now consider a nonlinear soft-margin support vector classifier using a Gaussian (radial basis function) kernel defined by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\{-\gamma||\boldsymbol{x} - \boldsymbol{x}'||^2\}$$

This kernel has one hyperparameter, γ, that controls the width of the Gaussian. A value that is often used for γ, although not necessarily the optimum one, is

$$\gamma = 1/d$$

where d is the number of features. Another option is to set γ equal to $1/d$ times the standard deviation of the features. The first option may be selected by setting `gamma='auto'` (or by not specifying a value for `gamma`). If `gamma='scale'` then the second option is used. For the value of C, the default value is $C = 1$ and this is not a bad one to begin with.

Experiments

(a) Repeat the previous exercise using a Gaussian kernel by setting `kernel='rbf'` in `SVC`. Begin with the default values for C and γ.

(b) Use `gamma='scale'` and see what difference this has on the classifier and its accuracy.

(c) Experiment with some values of γ that you set manually.

Computer Exercise 3.2 (SVMs for Detecting Breast Cancer):

In this exercise, you will be working with the Wisconsin Breast Cancer dataset to design a classifier that will determine, from clinical measurements of breast cancer tumors, whether or not a tumor is benign or malignant. The features in this data set were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, and they describe characteristics of the cell nuclei present in the image. The labels are made by a medical expert who has examined the image, and each tumor is labeled as benign or malignant. The task is to learn to predict whether or not a tumor is malignant based on the measurements of the tissue.

The data can be loaded using the `load_breast_cancer` function from scikit-learn as follows:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
X = cancer.data
y = cancer.target
```

As you will see by typing `cancer.data.shape`, there are 569 data samples consisting of 30 features. There are 212 samples labeled as **malignant** and 357 as **benign**. To get a description of the semantic meaning of each feature in `X`, look at the `feature_names`,

```
print("Feature names:\n{}".format(cancer.feature_names))
```

Knowing these names will be useful later when selecting subsets of the entire feature set.

1. Linear Soft-Margin Support Vector Machine

We begin our work in learning a classifier for the cancer dataset using an SVM with a linear kernel (Occam's Razor). With few exceptions, ML algorithms do not perform well when the numerical attributes of the features have different scales, and SVMs are no exception. So one of the most important preprocessing steps is **feature scaling**.

Examine

If you examine the features in the data set you will see that there is a wide variation in the scale between different features. Therefore, scale the training and test sets using `StandardScaler`, which scales each feature so that it has zero mean and unit variance.

When there is a training set `X_train` and a test set `X_test`, there are three ways to scale the two sets. After instantiating a scaler class, such as `StandardScaler`,

```
scaler = StandardScaler()
```

the scaling options are:

1. Use the entire dataset `X` to find the scaling parameters and use them to scale `X_train` and `X_test`,

   ```
   scaler.fit(X)
   X_train = scaler.transform(X_train)
   X_test = scaler.transform(X_test)
   ```

2. Use `X_train` to find the scaling parameters and use them to scale `X_train` and `X_test`,

   ```
   scaler.fit(X_train)
   X_train = scaler.transform(X_train)
   X_test = scaler.transform(X_test)
   ```

3. Use `X_train` to find scaling parameters and use them to scale `X_train`, and then use `X_test` to find another set of scaling parameters and use them to scale `X_test`,

   ```
   scaler.fit(X_train)
   X_train = scaler.transform(X_train)
   scaler.fit(X_test)
   X_test = scaler.transform(X_test)
   ```

Question

Only one of these methods is the right one to use. Do some research to find out the right way to do the scaling and explain why this is the approach that should be used.

Another important preprocessing step is to look for any missing features in the dataset and, if there are any, decide how to handle them. Fortunately, there is no missing data in the cancer dataset, so this step is not necessary.

Experiments

(a) Split the data into a training set and a test set, `(X_train, y_train)` and `(X_test, y_test)`, and perform feature scaling on both sets.

(b) Design a linear soft-margin support vector classifier with $C = 1$ and find the accuracy of your classifier on the training and test sets. Discuss what you find. Is the accuracy on the test and training sets close to being the same? If so, what does this tell you or if not, what does this mean?

(c) How many support vectors are there for each class?

(d) Try $C = 0.01, 0.1, 10, 100$ and see what difference it makes in the classifier. Does changing the value of C have much effect on the performance of your classifier?

2. Nonlinear Soft-Margin Support Vector Classifier

Now consider a nonlinear soft-margin support vector classifier using a radial basis function kernel. Recall that there are two hyperparameters that need to be set. The first is γ that controls the width of the Gaussian, and the second is C that defines how much to penalize margin violations.

Experiments:

(a) Design a soft-margin nonlinear support vector classifier using a radial basis function kernel with the default values for γ and C in the `SVC` class.

(b) Determine the performance of your classifier on the training and test sets. Do your results indicate that you may be under-fitting or over-fitting the data? Discuss.

(c) Experiment with different values of `gamma`. Begin by setting `gamma='auto'`. What value of γ is used with this setting? How well does your classifier work? Next, set the value of `gamma` manually and see if you can find a better value to use.

3. Dimensionality Reduction

The cancer data set has thirty features, and it is not clear whether or not all of them are useful or needed for effective classification (remember the curse of dimensionality). Some features may be correlated with others while others may not provide any useful information for distinguishing one class from another.

Two features that are considered important in the design of the breast cancer detector are features 24 and 28, 'worst area' and 'worst concave points' (Don't forget that the feature numbers begin at index 0). So let's see how well a classifier performs using only these two features. Here are the first steps:

(i) Form a new data set that contains only features 24 and 28,

```
X_2f = np.column_stack((X[:,[23]],X[:,[27]]))
```

(ii) Split the new data into a training set and a test set.

(ii) Scale the training and test sets using `StandardScaler`.

Experiments

(a) Design a linear soft-margin classifier using only features 24 and 28 and find the accuracy of the classifier on the training and test sets.

(b) How does your classifier compare to those designed using all thirty features?

(c) Since we now have a two-dimensional feature vector, make a plot of the decision boundary, the margin, and the support vectors. How many support vectors are there?

(d) Repeat using a polynomial kernel with degree $p = 3$ and with a radial basis function kernel.

Final Analysis

Out of all of the classifiers that you have designed for breast cancer detection, which one do you think is the best? Make the argument to a client on why you made this choice.

Epilogue

In your final analysis, you selected a classifier based on results you observed in terms of their accuracies on the test set (and perhaps on the training set and on the features that are used). You would not be justified in reporting the accuracy on the test set for your selection as a selling point, because it would not be a good indicator of how well you can expect your classifier to work on new data. Your test data is contaminated because it has been used to make a decision on what classifier is the best one. This issue lies at the heart of statistical learning theory, and we will study this problem later and find a solution when we look at the topic of **validation** and **cross-validation**.
