PMDS505P Data Mining and Machine Learning
Experiment 9

March 2025

1 Work to do today

Note: Make a single pdf file of the work you are doing in jupyter notebook. Upload with proper format. Please mention your name and roll no properly with Experiment number in the first page of your submission.

Support Vector machines

Q1. Today we will try to see how we can use SVM to perform classification on a binary classification problem.

- Create a synthetic dataset using make_classification class.
- Perform Min-Max scaling for the data.
- Do the train test split of the data with test size 20%.
- We can see how we can use different kernels and implement SVM
- Mainly we can try linear kernel, polynomial kernel with parameters degree d, and Coeff c and rbf kernel with parameters γ (Gamma)
  The linear kernel is the simplest form and is given by:

$$K\left(x_i, x_j\right) = x_i^T x_j$$

  The polynomial kernel is used to capture non-linear relationships and is given by:

$$K\left(x_i, x_j\right) = \left(x_i^T x_j + c\right)^d$$

  where: - d is the degree of the polynomial (controls model complexity), - c (or coefo in SVM) is a constant that controls the influence of higher-degree terms.

  The RBF kernel (Gaussian kernel) is widely used for capturing complex decision boundaries:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

where:
γ is a hyperparameter.
Low $\gamma \rightarrow$ More general decision boundary (underfitting).
High $\gamma \rightarrow$ More flexible decision boundary (overfitting).
$\|x_i - x_j\|^2$ is the squared Euclidean distance between two feature vectors.

- To work using linear kernel you can use the following code.

```
from sklearn.svm import SVC
svm = SVC(kernel='linear')
svm.fit(X_train,y_train)
```

- Check the accuracy of your model. Further try to plot the decision boundary as we did in the case of softmax regression.
- Now you can check the other kernels like polynomial kernel with a specific value of d and Coef0 which is the value of c in the polynomial kernel

```
svmpoly = SVC(kernel='poly', degree=2, coef0=1)
svmpoly.fit(X_train,y_train)
```

- Here you can perform hyperparameter tuning on the parameters degree and coef0 values and determine the best result you can obtain. And plot the decision boundary in that case.
- Now to use rbf kernel with the hyperparameter value Gamma you can use this code.

```
svm = SVC(kernel='rbf', gamma=20) svm.fit(X_train,y_train)
```

- When we use a large value of gamma, then we will have more complex models, but it may be prone to overfitting.
- Now here you can perform hyperparameter tuning on the parameter gamma by giving a range of values for gamma, which usually takes a value greater than 0. determine the best result you can obtain. And plot the decision boundary in that case.

2 PCA

We will see how we can use PCA and perform regression in the book1.csv dataset which is a house price dataset.

- Download the book1.csv file and drop appropriate columns in it.

- Scale the data using min max scaling to obtain the data matrix X_scaled.
- Fit the data using multiple regression and decision tree and find the value of mean squared error in both the cases.
- Create a matrix which contains only the scaled feature values, not the target variable, and name it as Xfeatures.
- Now let us use PCA to reduce the features we have in our data to three features or a 3-dimensional data.

  from sklearn.decomposition import PCA
  pca = PCA(n_components=3)
  n_components specifies to which dimension you have to transform the data to.
  X_pca = pca.fit_transform(Xfeatures)

- We will get the new set of features from the X_pca matrix. The three columns represent the three features data after applying PCA.
- Perform train test split and fit the models and find the errors in the case of both the models, multiple regression and decision tree by using the new three features we have obtained using PCA.