Classification of Human Activities using Smartphone Sensors through Machine Learning and Neural Network Algorithms

By: Yongxin Luo

Professor: Amir Jafari

Subject: Machine Learning I

29 April 2023

1. Introduction

1.1. An overview of the project

With the growth of Machine Learning applications, people want to recognize patterns by recording detailed data. In this field, it is really attractive to recognize human activities by their body movements using cameras. This can be applied to human health detection, movement status recognition and prediction.

Our project is about Human Activity Recognition with Smartphones. This dataset is built from the recordings of 30 study participants performing activities of daily living while carrying a waist-mounted smartphone with embedded inertial sensors, such as accelerometer and gyroscope. Each person performed six activities, which are walking, walking_upstairs, walking_downstairs, sitting, standing, and laying. The independent features are about gravitational and body motion components. Using the independent data, we need to classify and cluster correct activities. Therefore, our team used MLP, SVM and LVQ to predict correct activities and compare them with the recorded one.

Our team got 94% accuracy in MLP, 92% accuracy in LVQ and

1.2. Outline of the shared work

- We did EDA, including histograms, and correlation analysis.
- We preprocessed the dataset, including doing feature reduction (PCA), one-hot encoding and data scaling.
- We wrote codes about MLP, LVQ and SVM and evaluated them by f1-score, accuracy and some other matrices.
- We sharpened our mathematics knowledge related to algorithms we used.

- We tried different algorithms learnt in class and understood better.
- We also finished a group report and presentation slides.

2. Individual work

2.1. Tuning PCA

As there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. Therefore, it contains the most information on predicting a dependent variable. Mathematically speaking, it’s the line that maximizes the variance, the average of the squared distances from the original and projected points.

The first thing is to calculate the covariance matrix. Covariance and Variance are a measure of the “spread” of a set of points around their center of mass(mean). Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions. The covariance matrix computation is using this equation:

$$cov_{x,y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

Covariance matrix can be computed and shown in this form:

$$\begin{array}{c|ccc} & x & y & z \\ \hline x & var(x) & cov(x,y) & cov(x,z) \\ y & cov(x,y) & var(y) & cov(y,z) \\ z & cov(x,z) & cov(y,z) & var(z) \end{array}$$

The next thing is to compute the eigenvalues and eigenvectors of the covariance matrix to identify the principal components. The mathematical equations computing eigenvalues and eigenvectors are formulated as follows:

$$\mathbf{A}\cdot\mathbf{v}=\lambda\cdot\mathbf{v}$$
$$\mathbf{A}\cdot\mathbf{v}-\lambda\cdot\mathbf{v}=0$$
$$\mathbf{A}\cdot\mathbf{v}-\lambda\cdot\mathbf{I}\cdot\mathbf{v}=0$$
$$(\mathbf{A}-\lambda\cdot\mathbf{I})\cdot\mathbf{v}=0$$
$$|\mathbf{A}-\lambda\cdot\mathbf{I}|=0$$

Finally, I decided the number of key components and recast the data along the principal component's axes.

Figure1

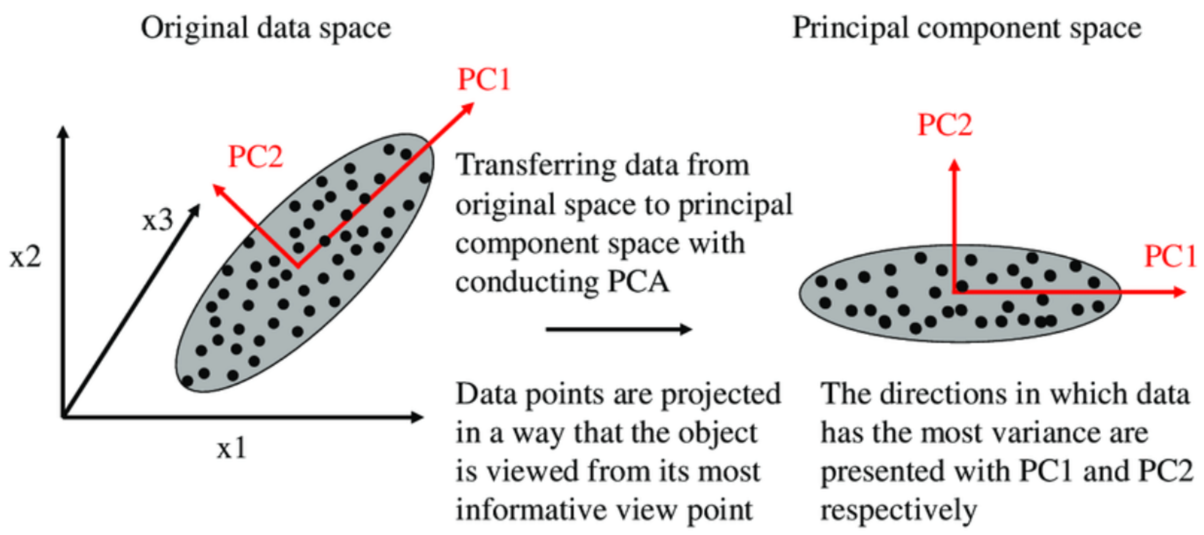
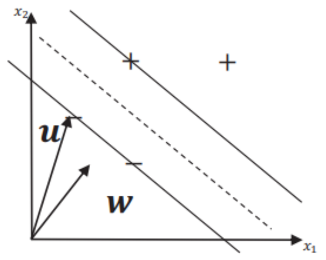

Original data space
Principal component space
PC1
PC2
x3
x2
x1
Transferring data from original space to principal component space with conducting PCA
Data points are projected in a way that the object is viewed from its most informative view point
PC2
PC1
The directions in which data has the most variance are presented with PC1 and PC2 respectively

Figure1: PCA transformation

2.2. SVM

In SVM, the most important things are to find support vectors and the largest margin. These are the points that are closest to the hyperplane. Margin is the distance between the hyperplane and the observations closest to the support vectors. In SVM large margin is considered a good margin. The largest margin and support vectors are shown below:

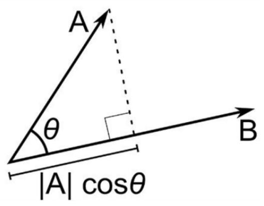


If w.xi + b = 0, it is the equation of margin of SVM. If w.xi + b >= 1, the data point will fall into "+" area, otherwise, the datapoint will fall into "-" area. The formulas are as follow:

$$\mathbf{w}.\mathbf{x}_+ + b \geq 1$$

$$\mathbf{w}.\mathbf{x}_- + b \leq -1$$

The dot product can be defined as the projection of one vector along with another, multiplied by the product of another vector. The dot product is shown below:



The dot product can be defined as the projection of one vector along with another, multiplied by the product of another vector. The distance of vector w from the origin to the decision boundary is 'c'. Let's say the projection vector of A is B, so if the dot-product of A and B is bigger than C, it is positive samples. If the dot-product of A and B is smaller than C, it's negative samples.

Figure2

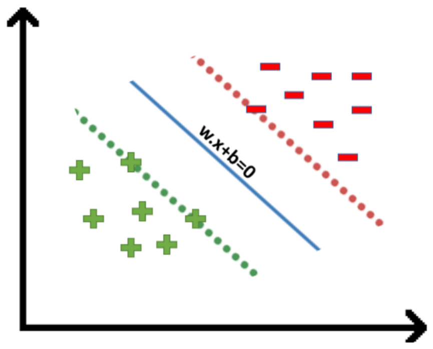


Figure2: largest margin, "+" and "-" area

To classify a point as negative or positive we need to define a decision rule. We can define decision rule as:

$$y = \begin{cases} +1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + b \geq 0 \\ -1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + b < 0 \end{cases}$$

3. Detailed work on the project

3.1. PCA

After separating independent X and dependent y, I scaled X variables into 0 and 1. Finishing this process of basic preprocessing, I built a PCA model and drew a graph about which is the best number of key components in PCA.

Figure3

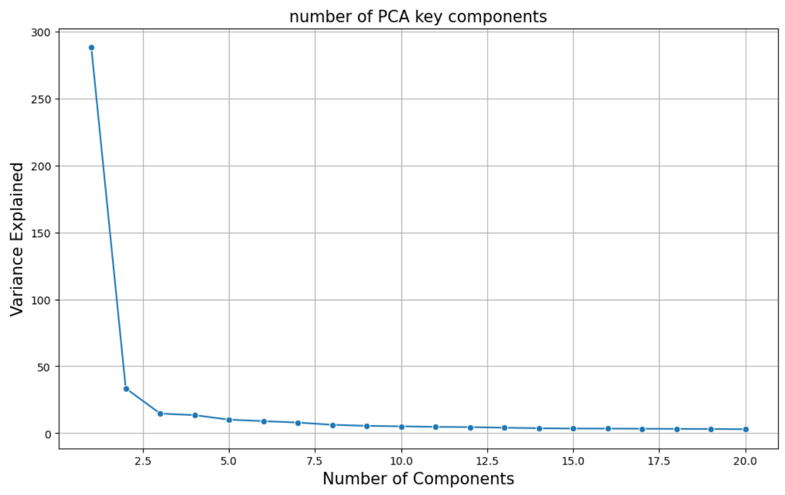


Figure3: Choosing the best number of PCA key component

After discussion, my teammate and I decided to choose components = 20 as our PCA model's key components.

Explained variance in this PCA is increasing and the cumulative explained variance is over 400 when the key component is set to be 20.

Figure4

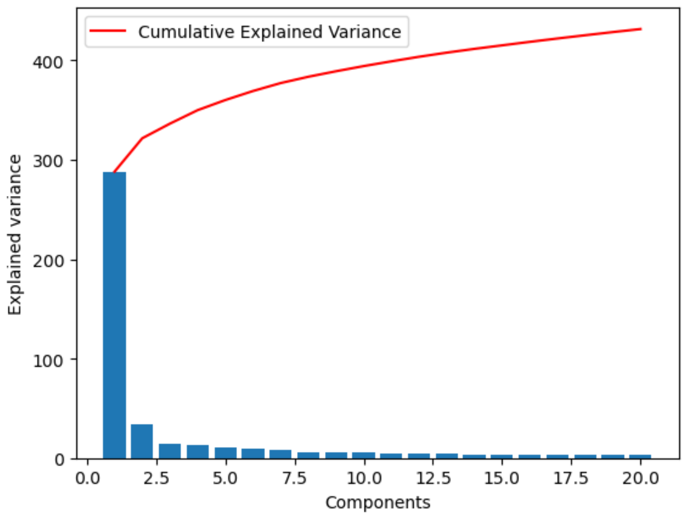


Figure4: Cumulative explained variance when components = 20

After making a new dataframe, I did a train test split and drew its confusion matrix.

Figure5

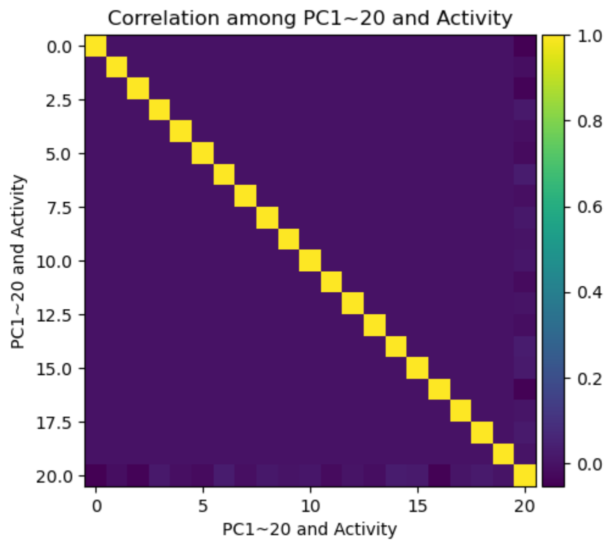


Figure5: Confusion matrix about 20 key components and activity feature

I picked the first two components after doing PCA, I drew their distribution in a scatter plot.

Figure6

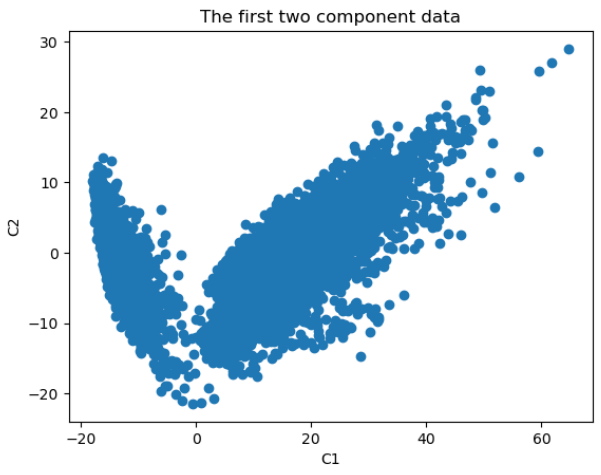


Figure6: Distribution of the first two key components

Also, I drew a scatter plot about 6 kinds of targets after filtering 20 components.

Figure7

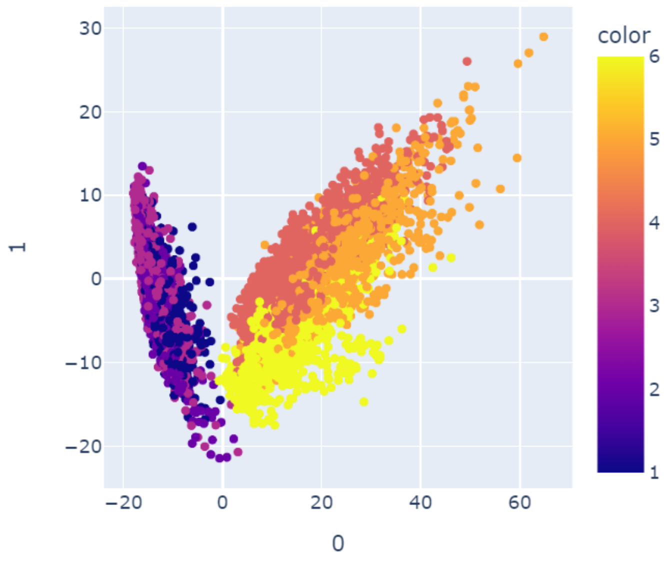


Figure7: Scatter plot of 6 kinds of target

Furthermore, I plot a 3D visualization to have a better understanding of SVM.

Figure8

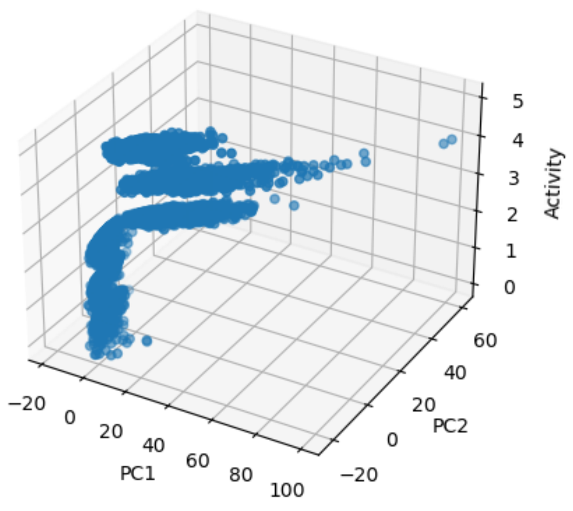


Figure8: 3D scatter plot of PC1, PC2 and Activity

I have known the data of these 6 kinds of targets are linearly separable, so I decided to build an SVM model on it. Specifically, I will define the kernel types of SVM.

3.2. SVM and Optimization

Firstly, I built a linear SVM model and set the penalty to be 1. Then, I fit the model and predict the model using *x_test*. This linear SVM model is good to use, because the metrics' scores are high. When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).

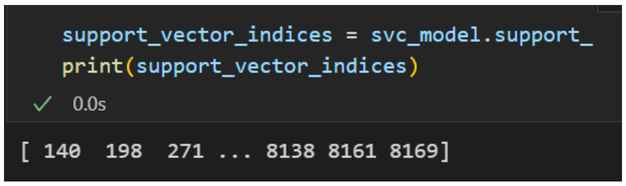
svc_model = SVC(C=.1, kernel='linear', gamma=1)

As I mentioned earlier in this report, the data are almost linearly separable, so I tried kernel = "linear", "polynomial", and "sigmoid". In polynomial SVM, it reaches the best when the class label is set to be 10. Their metrics are all good.

poly_svc=SVC(kernel='poly', C=10)

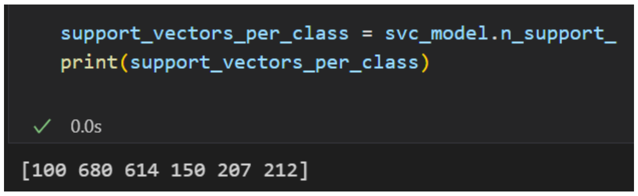
sigmoid_svc=SVC(kernel='sigmoid', C=1.0)

The most important thing is to find and plot the support vectors in SVM. The *.support_* variable, which holds the index numbers of the samples from your training set that were found to be the support vectors. I saw lots of vectors here. Some of the indices of support vectors are shown as follow:



```
support_vector_indices = svc_model.support_
print(support_vector_indices)
✓ 0.0s
[ 140  198  271 ... 8138 8161 8169]
```

Then, I took a look at the *.n_support_* variable, which produces the number of support vectors for every class. The classes of support vectors are shown below:



```
support_vectors_per_class = svc_model.n_support_
print(support_vectors_per_class)

✓ 0.0s
[100 680 614 150 207 212]
```

The final visualization part in SVM is to visualize the training set and the support vectors using Matplotlib.

Figure9

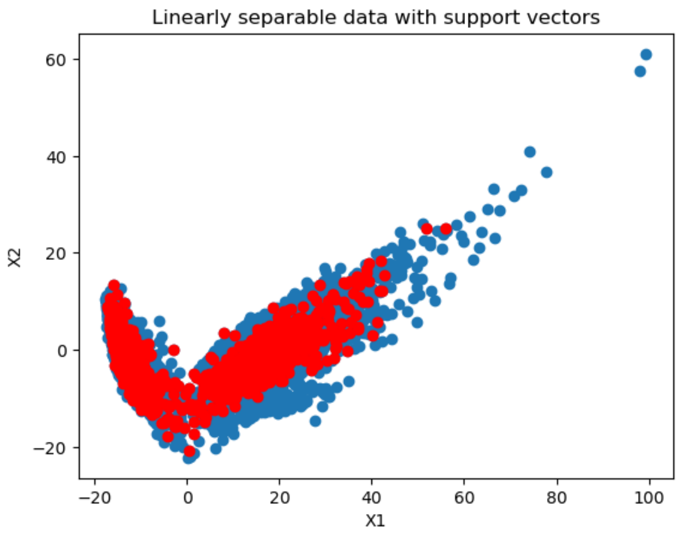


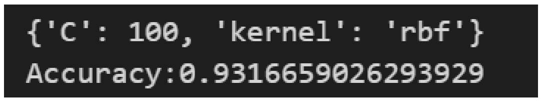
Figure9: Support vectors in linear SVM

As you can see, there are too many red dots (support vectors) showing here. That means the dataset is overfitting.

Next step, I adopt two hyperparameter optimization methods to improve my SVM model. One is grid search in SVC. In grid search, I set C can to be 1,10 and 100 and kernel can be set to be linear, poly, rbf, sigmoid with GridSearchCV method. Codes are shown as follows:

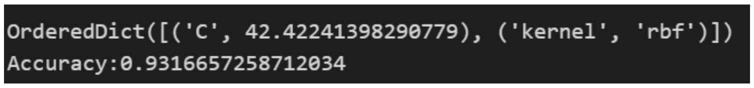
```
rf_params = {'C': [1,10, 100],  "kernel":['linear','poly','rbf','sigmoid']}
svc_grid = SVC(gamma='scale')
grid = GridSearchCV(svc_grid, rf_params, cv=3, scoring='accuracy')
grid.fit(X_train, y_train)
print(grid.best_params_)
print("Accuracy:"+ str(grid.best_score_))
```

The model select *penalty* equals 100 and the *kernel* is to be *RBF*. The accuracy is 0.93, and what's worthy mentioned is that it indeed improved from 0.90 to 0.93. The information is shown as follow:



```
{'C': 100, 'kernel': 'rbf'}
Accuracy:0.9316659026293929
```

The other one hyperparameter optimization is bayesian optimization with a gaussian process. It selects c equals 42 and the kernel is to be rbf. The accuracy is to be 0.93.



```
OrderedDict([('C', 42.42241398290779), ('kernel', 'rbf')])
Accuracy:0.9316657258712034
```

I am not sure if the SVM model is overfitting, so I tried the Artificial Neural Network by Keras and added early stopping to fix this problem.

def create_model():

model = Sequential([Dense(64, activation='relu', input_shape=(20,)),Dense(64, activation='relu'), Dense(1, activation='softmax')])

return model

early_stopping = EarlyStopping()

custom_early_stopping = EarlyStopping(monitor='val_accuracy', patience=8, min_delta=0.001, mode='max')

The loss in the test is 0, but the accuracy is low. That is because the early stopping monitors the loss rather than the accuracy.

4. Results

After scaling and doing PCA, we built some SVM models based on cleaned data, including linear SVM, polynomial SVM and sigmoid SVM. Their accuracy, precision, recall rate and f1-score are presented in Table3.

Table1

Evaluation Results of SVMs

Algorithm	Accuracy	Precision	Recall	F1-score	K-fold validation
Linear - SVM	90.92%	0.9114	0.9112	0.9111	0.9092
Polynomial - SVM	93.06%	0.9336	0.9327	0.9328	0.9305

Sigmoid - SVM	76.21%	0.7696	0.7533	0.7526	0.4519

All of these three SVM models' metrics are shown pretty well. That means all these three models perform well. However, one thing out of my expectation is that all metrics show polynomial SVM perform best, but almost all data can be separated linearly in the visualization part.

In SVM, there is an important parameter Gamma (γ), which controls overfitting in SVM. The higher the gamma, the higher the hyperplane tries to match the training data. So I set gamma to 5, but the results are similar to the previous one.

In the confusion matrix of SVM in the test set, True Positives(TP) = 398 and True Negatives(TN) = 284 are much higher than False Positives(FP) = 9 and False Negatives(FN) = 7. It means that it is highly possible to be classified as correct activities.

Figure10

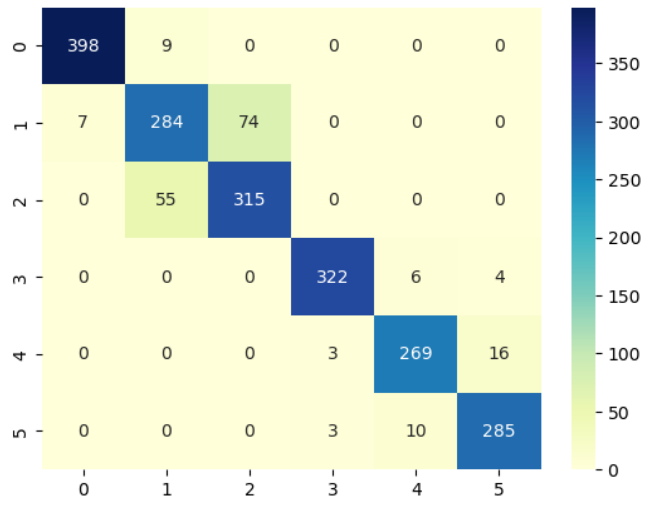


Figure10: Confusion matrix of SVM (test set)

Furthermore, we check if it is overfitting or underfitting. The score of the train set and the test set are shown in Table2.

Table2

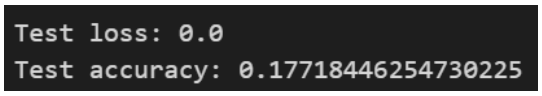
Overfitting and underfitting of SVM

Algorithm	Score of train set	Score of test set
Linear - SVM	0.9125	0.9092
Polynomial - SVM	0.9439	0.9306
Sigmoid - SVM	0.7642	0.7621

No matter which kernels we chose, the training-set accuracy score and the test-set accuracy are quite comparable. That is, there is no question of overfitting or underfitting.

However, it is different from what I discussed earlier. The visualization of support vectors shows the model is overfitting. Anyway, I will fix this problem in the next step.

To fix overfitting, I added early stopping in the Artificial Neural Network.



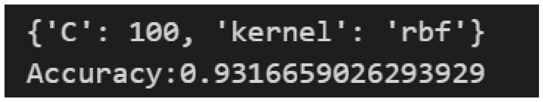
```
Test loss: 0.0
Test accuracy: 0.17718446254730225
```

I got the loss in the test set is 0, and the accuracy is 0.1771. This accuracy is low, but the overfitting problem is indeed solved. That is because the early stopping monitors the loss rather than the accuracy. The loss quantifies how certain the model is about a prediction and the accuracy merely accounts for the number of correct predictions.

Another solution to fix the overfitting problem is using soft margin in SVM instead of hard one. Soft margin SVM allows some misclassification to happen by relaxing the hard constraints of Support Vector Machine. Soft margin SVM is implemented with the help of the Regularization parameter (C). Regularization parameter (C): It tells us how much misclassification we want to avoid. By lowering the regularization parameter, the margin increases, so the training model is to become a soft SVM. I set all C in different kernels' SVM are no bigger than 50.

Then, to improve the model's performance, I added hyperparameter optimization. The first one is grid search. By setting penalty parameters as 1, 10, 100 and setting kernels as linear, poly, rbf and sigmoid with GridSearchCV method.

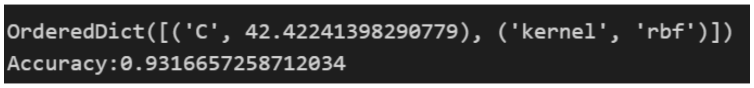
The model select *penalty* equals 100 and the *kernel* is to be *RBF*. The information is shown as follow:



```
{'C': 100, 'kernel': 'rbf'}
Accuracy:0.9316659026293929
```

The accuracy is 0.93, and what's worthy mentioned is that it indeed improved from 0.90 to 0.93.

The other one hyperparameter optimization is bayesian optimization with a gaussian process. It selects c equals 42 and the kernel is to be rbf. The information is shown as follows:



```
OrderedDict([('C', 42.42241398290779), ('kernel', 'rbf')])
Accuracy:0.9316657258712034
```

The accuracy is 0.93, which is also improved by 0.03 with bayesian optimization.

5. Summary and conclusions

5.1. Conclusion

Before doing anything, I took a look at our dataset. Then, I did some EDA to have a better understanding of this dataset. After that, I scaled the data to reduce the error caused by variances among all features. Doing PCA, the data frame has just left 20 key columns. These 20 features contained much information from the original dataset. By plotting scatter plots, we can have an instinctively understanding of the distribution of those data after doing feature reduction and I also know the data can be linearly separated. Therefore, I built different SVM models to check their performance, including linear-svm, sigmoid-svm and polynomial-svm. The results of linear-svm and polynomial-svm are above 0.90/ 1.00, like accuracy, f1-score and validation mean score. About sigmoid-svm, the results are not as good as the previous two models' results. The accuracy is 76.21% and the k-fold validation mean score is 0.4519. Also, I tried to fix the overfitting problem in SVM. I tried two ways, which are setting soft margins and customizing early stopping. After performing these two methods, I decided to set regularization parameters to be all smaller than 50. Besides, customizing the early stop is to be stopped in patience = 42, I got the loss equals to 0. At the end, I optimized the SVM model using grid search and Bayes search. Both hyperparameter optimizations improved SVM models' accuracy from 91% to 93%.

Combining what I have learnt in class, I know the logical process of SVM better, including what support vectors are and how to find these vectors. After finding these vectors, I know how to find the maximum margin and plot hyperplane. Also, I took a deeper understanding of choosing different kinds of kernels in SVM and how to evaluate the performance of SVM. Last but not least, I also learned how to improve models using hyperparameter optimization methods. What's worthy mentioning is that I tried to implement SVM Derivation based on Lagrange Multipliers, even though I can't implement it successfully. However, in this process, I understood mathematics and coding better.

In conclusion, I have a wider understanding of SVM model and PCA method after finishing this ML final project. Also, the evaluation of these models are shown well.

5.2. Improvements in the future

About models, I can take a deeper look at overfitting issues and make the model perform better.

About myself, I knew my mathematical and coding skills are still not enough now to make me succeed in the competitive markets. There is still a long way to go.

6. Percentage of code

(258-42)/ (258+3) = 82.76%

7. Reference

L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," Neurocomputing, vol. 415, pp. 295–316, 2020, doi: https://doi.org/10.1016/j.neucom.2020.07.061.

Chawla, K., Prakash, C., & Chawla, A. (2021). Comparative study of computational techniques for smartphone based human activity recognition. *Lecture Notes in Electrical Engineering*, 427–439. https://doi.org/10.1007/978-981-16-3067-5_32.

Hagan, M. T., Demuth, H. B., Beale, M. H., & Jesús Orlando De. (2016). *Neural network design*. s. n.

Islam, M. M., Nooruddin, S., Karray, F., & Muhammad, G. (2022). Human activity recognition using tools of convolutional neural networks: A State of the art review, data sets, challenges, and future prospects. *Computers in Biology and Medicine*, *149*, 106060. https://doi.org/10.1016/j.compbiomed.2022.106060.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory. https://doi.org/10.1145/130385.130401.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. International Workshop of Ambient Assisted Living (IWAAL 2012). Vitoria-Gasteiz, Spain. Dec 2012

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge L. Reyes-Ortiz. Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic. Journal of Universal Computer Science. Special Issue in Ambient Assisted Living: Home Care. Volume 19, Issue 9. May 2013