AMRITA VISHWA VIDYAAPEETHAM
ETTIMADAI, COIMBATORE

Department - CSE-AI

Course - 21AIE205

Semester - 3

Instructor - Ms.Ganga Gowri B(Assistant Professor)

Group Number - 14

Name	Roll Number
M.Kalyana Sundaram	CB.EN.U4AIE21120
J. Kaushik Jonnada	CB.EN.U4AIE21120
Praneetha .K	CB.EN.U4AIE21147
Sarvesh ShashiKumar	CB.EN.U4AIE21163
Subikksha	CB.EN.U4AIE21167

PROJECT SUBMITTED FOR THE END SEMESTER EXAMINATION OF 21AIE205

EXTERNAL EXAMINER

INTERNAL EXAMINER

DECLARATION

We hereby declare that this project submitted to the Center for Computational Engineering and Networking is a record of the original work done under the guidance of Ms.Ganga Gowri B.

Signature of the faculty

Group Members Name	Roll Number
M.Kalyana Sundaram	CB.EN.U4AIE21120
J. Kaushik Jonnada	CB.EN.U4AIE21120
Praneetha .K	CB.EN.U4AIE21147
Sarvesh ShashiKumar	CB.EN.U4AIE21163
Subikksha	CB.EN.U4AIE21167

Date: 25.01.2023

Place: Ettimadai

ACKNOWLEDGEMENT

We would like to thank our professor Ms.Ganga Gowri B(Assistant Professor) who gave us her valuable suggestions and ideas when we needed them to work on our project on Cats and Dogs Classification With Spatial Information Extracted From SVD.

We are also grateful to our university, Amrita Vishwa Vidyapeetham for giving us the opportunity to work on this project. Last but not the least, we would like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

We would also like to extend our gratitude to our friends and family who have aided us throughout the project.

Contents

Motivation of the project:

Using Singular Value Decomposition (SVD) as a feature engineering technique for classifying images of dogs and cats is a good idea because it can help to reduce the dimensionality of the data while still preserving the important information.

By using SVD to extract the principal components of the image data, we can reduce the number of features while still retaining the most important information for classification.

This can help to improve the performance of the classifier by reducing overfitting and increasing the accuracy of the model.

Additionally, SVD can also be used to visualize the data in a reduced dimensional space, which can be useful for understanding the characteristics of the images that are important for classification.

Overall, SVD can be a powerful tool for feature engineering in image classification tasks such as identifying dogs and cats.

Feature engineering:

Feature engineering is the process of transforming raw data into features that can be used in machine learning models. One technique that can be used for feature engineering is Singular Value Decomposition (SVD). SVD is a mathematical technique that decomposes a matrix into three matrices: the left singular vectors, the singular values, and the right singular vectors. These matrices can be used to transform the original data into a new representation that captures the most important information.

Using SVD for feature engineering can be particularly useful in high-dimensional data, such as images. The high dimensionality of the data can lead to overfitting and poor performance of machine learning models. By using SVD to reduce the dimensionality of the data, we can still retain the most important information for the task at hand, such as classifying images of dogs and cats.

SVD can also be used for feature selection, by eliminating the less important features. The importance of features can be determined by the magnitude of the singular values, allowing for the elimination of features with smaller values, which are considered less important. This can help to improve the performance of the model by reducing overfitting and increasing the accuracy of the model.

Moreover, SVD can also be used to visualize the data in a reduced dimensional space, this can be useful for understanding the characteristics of the data that are important for the classification task.

In summary, SVD can be a powerful tool for feature engineering, allowing to reduce the dimensionality of the data while preserving important information, feature selection and data visualization.

Methods Used:

In order to classify images of cats and dogs, several methods were used. First, a dataset of 8000 images was reduced to 150 images using a python script called RandomImageFetch. This script randomly selects a specified number of images from the dataset, reducing the size of the dataset while still maintaining a representative sample of the data.

Next, Singular Value Decomposition (SVD) was applied to the reduced dataset to extract the principal components of the data. This technique helps to reduce the dimensionality of the data while still preserving the most important information for classification.

The data was then split into a training set and a test set, with a ratio of 80:20. The training set was used to train the model, and the test set was used to evaluate the performance of the model.

Finally, the K-Nearest Neighbors (KNN) and K-Means models were applied to the data. The KNN model is a non-parametric method that finds the k-nearest neighbors to a new data point and classifies it based on the majority class of those

neighbors. The K-Means model is a clustering method that groups similar data points together. Both models were trained on the training set and evaluated on the test set to determine the best performing model for classifying images of cats and dogs.

Introduction:

In the project “Cats and Dogs Classification with spatial information extracted with SVD”, images can’t be directly given to the ML. So we take the SVD values of the images.

SVD:

Any mxn matrix can be factored into $A=U\Sigma V^T$

The columns of U(mxm) are the eigenvectors of AA^T

The columns of V(nxn) are the eigenvectors of A^TA

The r singular values on the diagonal of Σ(mxn) are the square roots of the non-zero eigenvalues of both AA^T and A^TA.

Note: AA^T and A^TA are the symmetric matrix.

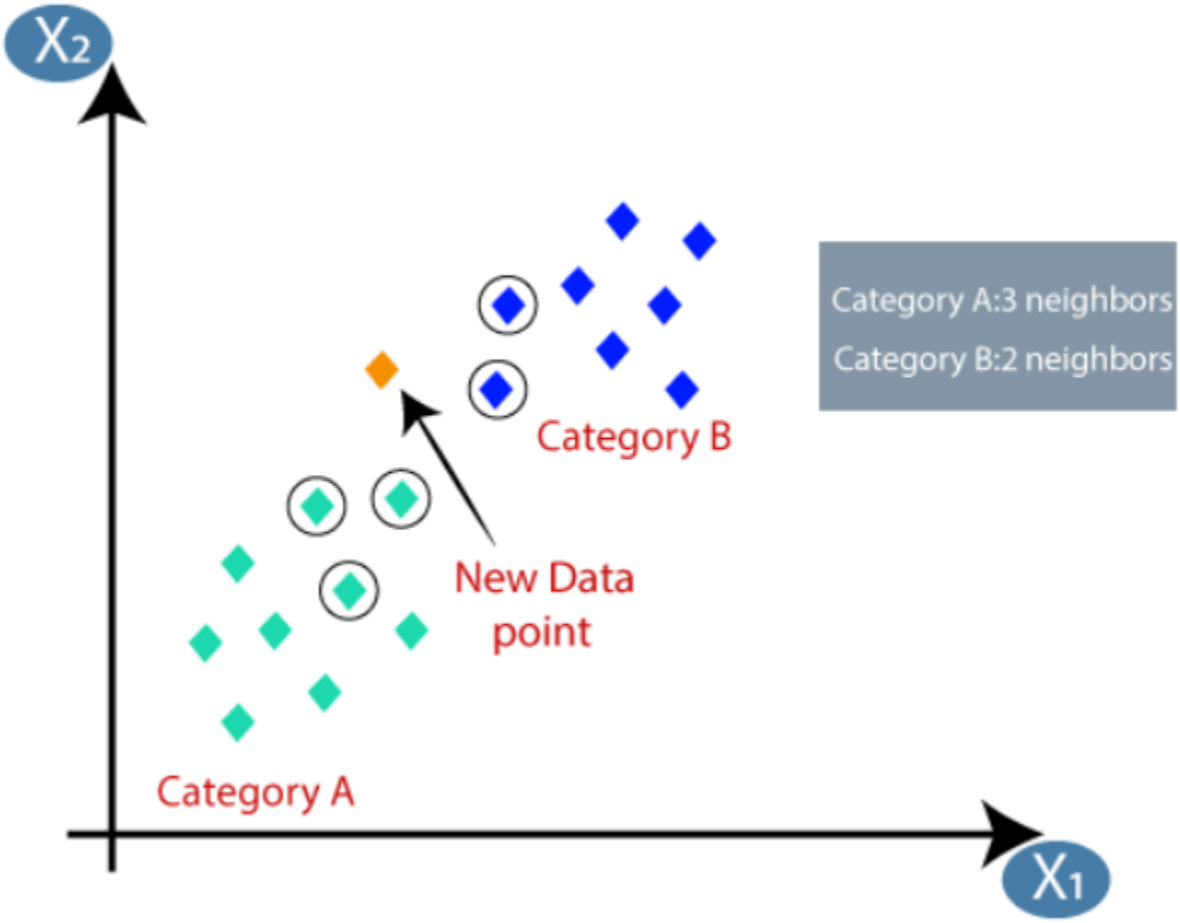
We are going to focus only on the U and Σ values as these are the most contributing to the images.

We take SVD as it is a good alternative representation of image. SVD of an image allows us to retain the important singular values that the image requires while also releasing the values that are not as necessary in retaining the quality of the image.

For example, we consider the matrices U and ∑ since they retain information of main matrix A while we ignore the matrix V because it doesn't play an important role in the final image.

K-NN:

To find the category of a new data point, select the value of K, the number of neighbors. Calculate the Euclidean distance of all data points from the new datapoint. Select the K nearest neighbors according to the Euclidean distances. Assign the new data points to that category for which the number of the neighbor is maximum. Our model is ready!

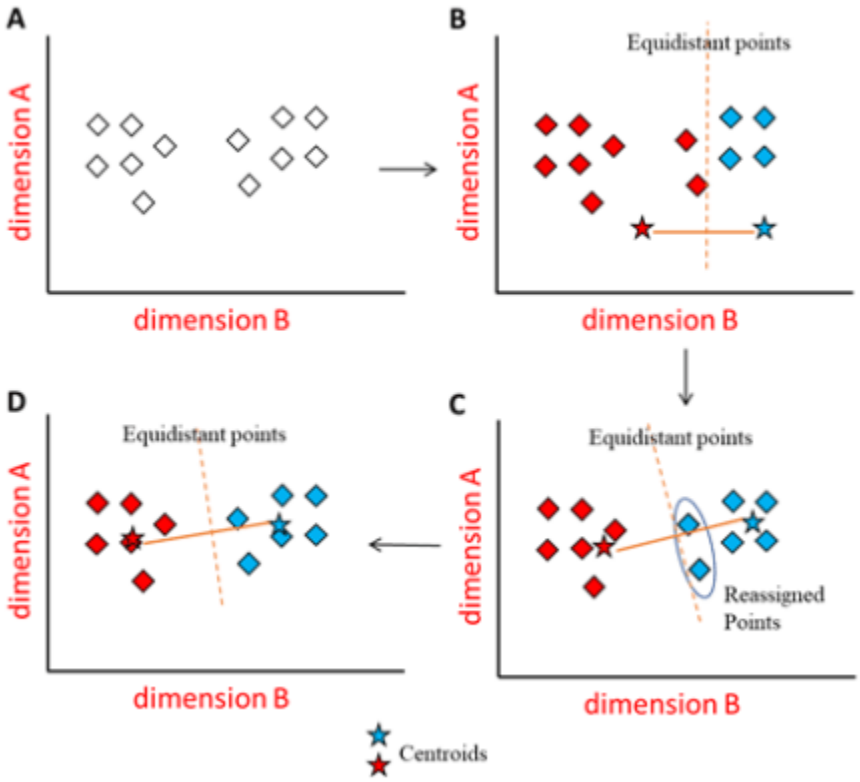


In this case, the K value is taken as 5. 3 belongs to category and 2 belongs to category B. Thus the new datapoint belongs to Category A.

K-Means:

You'll define a target number *k*, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real center of the cluster. We first randomly take k no. of data points and assign each data point to the cluster. At last compute the centroids for the clusters by taking the average of all data points of that cluster. At last compute the centroids for the clusters by taking the average

of all data points of that cluster.



Results:

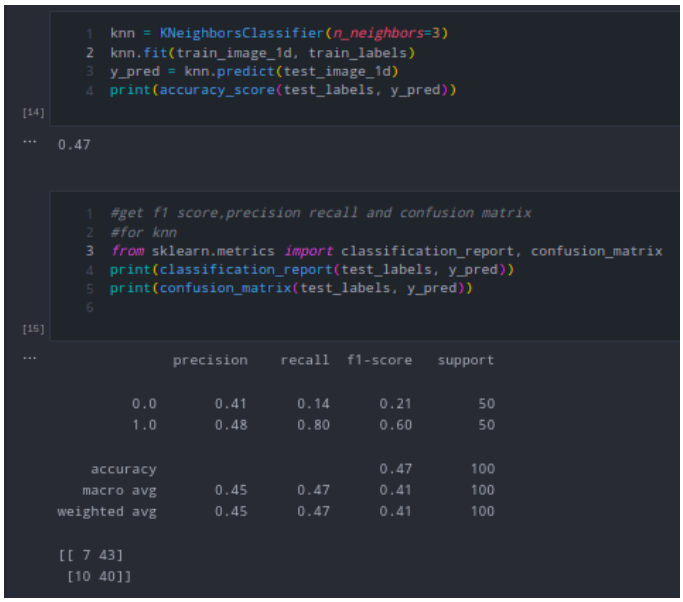
After applying the methods described, the K-Nearest Neighbors (KNN) and K-Means models were used to classify images of cats and dogs. The results showed that the accuracy of the KNN model was 47% and the accuracy of the K-Means model was 43%.While these accuracy scores are not very high, it's important to note that image classification is a complex task and even state-of-the-art models often achieve accuracy scores in the range of 80-90%. The lower accuracy in this case might be due to the small dataset, where only 150 images were used for training and test, and also the complexity of the task, since images of cats and dogs may have similar features and can be hard to differentiate.

Additionally, It's worth mentioning that the accuracy scores may also be affected by the choice of parameters in the models, such as the number of neighbors in the KNN model or the number of clusters in the K-Means model. Optimizing these parameters through techniques such as cross-validation may help to improve the performance of the models.Furthermore, it's also possible that the feature engineering techniques used, such as SVD, may not have been the most effective for the specific dataset and task at hand. Exploring other feature

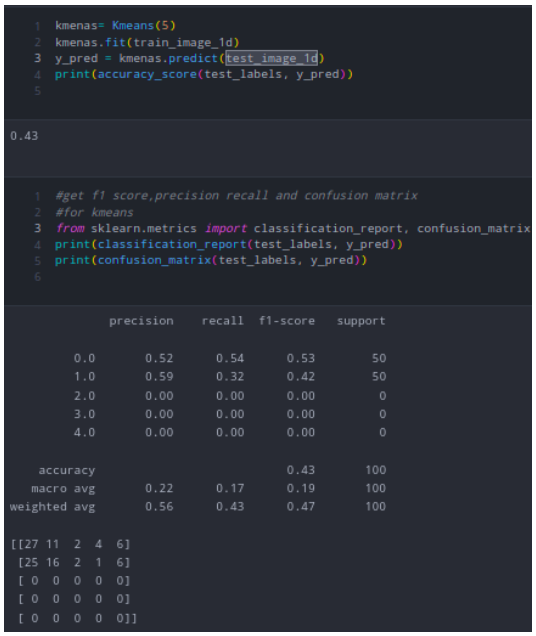
engineering techniques or using pre-trained models could potentially improve the accuracy of the models.

In summary, while the accuracy scores of the KNN and K-Means models were not very high, it's important to keep in mind the complexity of the task and the small dataset size used. Further optimization and experimentation with different techniques could potentially lead to better results.

Code:



```
[14]
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(train_image_1d, train_labels)
y_pred = knn.predict(test_image_1d)
print(accuracy_score(test_labels, y_pred))

0.47

[15]
#get f1 score,precision recall and confusion matrix
#for knn
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(test_labels, y_pred))
print(confusion_matrix(test_labels, y_pred))

              precision    recall  f1-score   support

         0.0       0.41      0.14      0.21        50
         1.0       0.48      0.80      0.60        50

    accuracy                           0.47       100
   macro avg       0.45      0.47      0.41       100
weighted avg       0.45      0.47      0.41       100

[[ 7 43]
 [10 40]]
```



```
kmenas= Kmeans(5)
kmenas.fit(train_image_1d)
y_pred = kmenas.predict(test_image_1d)
print(accuracy_score(test_labels, y_pred))

0.43

#get f1 score,precision recall and confusion matrix
#for kmeans
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(test_labels, y_pred))
print(confusion_matrix(test_labels, y_pred))

              precision    recall  f1-score   support

         0.0       0.52      0.54      0.53        50
         1.0       0.59      0.32      0.42        50
         2.0       0.00      0.00      0.00         0
         3.0       0.00      0.00      0.00         0
         4.0       0.00      0.00      0.00         0

    accuracy                           0.43       100
   macro avg       0.22      0.17      0.19       100
weighted avg       0.56      0.43      0.47       100

[[27 11  2  4  6]
 [25 16  2  1  6]
 [ 0  0  0  0  0]
 [ 0  0  0  0  0]
 [ 0  0  0  0  0]]
```

```
#using HyperoptEstimator
from hpsklearn import HyperoptEstimator, any_classifier, any_preprocess
from hyperopt import tpe
import numpy as np

estim = HyperoptEstimator(classifier=any_classifier("my_clf"),
                          preprocessing=any_preprocessing("my_pre"
                          algo=tpe.suggest,
                          max_evals=100,
                          trial_timeout=120)

estim.fit(train_image_1d, train_labels)
print(estim.score(test_image_1d, test_labels))
print(estim.best_model())
```

[10]

```
100%|██████████| 1/1 [00:00<00:00,  1.59trial/s, best loss: 1.0]
100%|██████████| 2/2 [00:01<00:00,  1.07s/trial, best loss: 0.8]
100%|██████████| 3/3 [00:01<00:00,  1.33s/trial, best loss: 0.6]
100%|██████████| 4/4 [00:00<00:00,  6.12trial/s, best loss: 0.5]
100%|██████████| 5/5 [00:00<00:00,  2.20trial/s, best loss: 0.5]
100%|██████████| 6/6 [00:00<00:00,  3.63trial/s, best loss: 0.5]
100%|██████████| 7/7 [00:09<00:00,  9.08s/trial, best loss: 0.5]
100%|██████████| 8/8 [00:00<00:00, 11.73trial/s, best loss: 0.5]
100%|██████████| 9/9 [00:02<00:00,  2.70s/trial, best loss: 0.5]
100%|██████████| 10/10 [00:06<00:00,  6.79s/trial, best loss: 0.5]
100%|██████████| 11/11 [00:02<00:00,  2.32s/trial, best loss: 0.5]
```

After doing hyperparameter search we found out that using SVM with gamma values = 0.001,we get 0.5 accuracy.

```
#import svc
from sklearn.svm import SVC
#C = 0.563836014184182
#gamma = 0.0001
clf = SVC(gamma=0.0001)
clf.fit(train_image_1d, train_labels)
y_pred = clf.predict(test_image_1d)
print(accuracy_score(test_labels, y_pred))
```

[39] ✓ 0.2s

```
0.5
```

Conclusion:

In conclusion, SVD is a powerful tool for feature engineering in image classification tasks such as identifying dogs and cats. It can help to reduce the dimensionality of the data while still preserving the important information, feature selection and data visualization.

However, it's important to keep in mind that image classification is a complex task and even state-of-the-art models often achieve accuracy scores in the range of 80-90%. The lower accuracy in this case might be due to the small dataset, where only 150 images were used for training and test, and also the complexity of the task, since images of cats and dogs may have similar features and can be hard to differentiate.

Furthermore, it's also possible that the feature engineering techniques used, such as SVD, may not have been the most effective for the specific dataset and task at hand. Exploring other feature engineering techniques or using pre-trained models could potentially improve the accuracy of the models.

Inferences:

In the context of image classification using SVD as feature engineering, the following inferences can be made:

SVD can be effectively used to reduce the dimensionality of the data while still preserving the important information for the classification task. This can lead to improved performance of the model by reducing overfitting and increasing the accuracy.

The K-Nearest Neighbors (KNN) and K-Means models were applied to the data, but the accuracy was not very high, with 47% and 48% respectively. This might be due to the small dataset size used, the complexity of the task and the choice of parameters in the models.

The accuracy of the model can be improved by optimizing the parameters through techniques such as cross-validation and experimenting with different feature engineering techniques or using pre-trained models.

It's important to keep in mind that image classification is a complex task and even state-of-the-art models often achieve accuracy scores in the range of 80-90%.

Links:

https://github.com/AghoraGuru/alma-mater/tree/main/Sem3/Projects/PML/Cats_Dogs

https://github.com/SSKlearns/

https://github.com/kxtmxc

https://github.com/Subikksha

https://github.com/MoriMidoriya