Classification of images using various classifiers and a comparative study on their accuracy.

Course Project report for:

CSE 515 - Multimedia and Web Databases

submitted to:

Dr. K. Selçuk Candan

by:

Amani Gudali - 1213134882

Tarun Lolla - 1216095647

Table of Contents

Introduction

In the third Phase of this project, our team has implemented the tasks 3 and 4 mentioned in the project description documentation.

In Task 3, we have computed Personalised Page Rank (PPR) for a given set of images based on their similarity and display the K most dominant images with respect to a given set of images.

In Task 4, our team has implemented three classifiers viz.,

1. Decision Tree Classifier
2. Support Vector Machines (SVM)
3. Personalised Page Rank (PPR) based classifier.

To the end of classification, unlabelled images are classified as either ‘dorsal’ or ‘palmar’

During the process of this implementation, we have used Histogram of Gradient (HoG) to extract features for every image in the dataset and used Principal Component Analysis (PCA) to reduce the dimensions of the computed HoG vectors in the dataset.

The reason we have chosen HoG over all the other options like Color Moments, LBP, and SIFT is because in this phase we will be classifying Hand images for their Dorsal or Palmar face. HoG enables us to get an idea of the texture of the image which is helpful in classifying the image. PCA is used for dimensionality reduction as it is effective in preserving the discrimination power.

Keywords: Personalised Page rank Algorithm, Support Vector Machine, Decision Tree Classifier, Dorsal, Palmar.

Terminology

Histogram of Oriented Gradients (HOG):

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image.[1] The histogram of oriented gradients descriptor is a local object appearance and shape within an image and can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms.[1]

Personalized Page Rank Algorithm (PPR):

Personalized PageRank recursively models the importance of nodes on a directed graph. At a high level, given a start node s whose point of view we take, we say that s is important, and in addition we say a node is important if its in-neighbours are important. To keep the importance scores bounded, we normalize the importance

given from a node u to a node v though an edge (u,v) by dividing by u's out-degree.[2]

Support Vector Machine (SVM):
A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. Given labelled training data (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. [3]

Principal Component Analysis (PCA):
Principal component analysis (PCA) is a dimensionality reduction technique that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components with the least important features being dropped.

Decision Tree Classifier:
A decision tree classifier is a classification method to classify a given set of examples by forming questions based on the attribute till a homogeneous class label is achieved. In the decision tree, the root and internal nodes contain attribute test conditions to separate records that have different characteristics. [4]

CART Algorithm:
The representation of the CART model is a binary tree. A node represents a single input variable (X) and a split point on that variable, assuming the variable is numeric. The leaf nodes of the tree contain an output variable (y) which is used to make a prediction. Once created, a tree can be navigated with a new row of data following each branch with the splits until a final prediction is made.

Gini Index:
The Gini index is the name of the cost function used to evaluate splits in the dataset. A Gini score gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. A perfect separation results in a Gini score of 0, whereas the worst case split that results in 50/50 classes in each group result in a Gini score of 0.5.[5]

Implementation

In this section, we will be delineating the steps involved in running the code in the user's machine.

The user is expected to have the following prerequisite softwares/packages to be able to run the tasks in this project.

1. Python3 along with the below libraries:
    a. Numpy
    b. Pandas
    c. Spatial
    d. Sci-kit learn (sklearn)
    e. Pymongo (to be able to connect to mongoDB)
    f. Progressbar
    g. OpenCV
    h. Matplotlib
2. MongoDB
3. Web browser (To be able to render the visualisation)
4. The project has been implemented and tested in both Linux and Windows environments and the user can work on the OS of their choice.

Before proceeding to implement the project, install MongoDB on your machine and populate the database details in the file **db_details.txt**. The structure of this file is as below (an example) :

```
db_host=localhost
db_port=27017
db_sid=phase3
```

After populating the Database Details, edit the following files:

1. dataset.txt:
   This file contains the paths to the dataset and metadata file for images in the dataset. Once dataset details are populated in the file, run **initialSteps.py** which computed the HoG vectors for all images in the dataset and stores them in mongoDB. The same file is referred to in the implementation of Task3 where the PPR is computed for images in this dataset. An example is as below:

   ```
   dataset_path=/home/phase3_sample_data/Labelled/Set2
   metadata_file=/home/phase3_sample_data/labelled_set2.csv
   ```

2. dataset_task4.txt:
   This file contains the dataset details for the labelled dataset and the unlabelled dataset along with their respective metadata paths. The file also requires the master metadata file path in order to compute the accuracy of

the classifiers. An example is as below:
Important Note: Before proceeding with the execution of task4, the feature vectors of labelled and unlabelled images should be present in the database. i.e., users should first run initialSteps.py by editing dataset.txt for each dataset and then proceed with execution of Task 4 by editing this file.

```
ll_dataset_path=/home/phase3_sample_data/Labelled/Set2
ll_metadata_file=/home/phase3_sample_data/labelled_set2.csv
ull_dataset_path=/home/phase3_sample_data/Unlabelled/Set2
ull_metadata_file=/home/phase3_sample_data/unlabeled_set2.csv
master_metadata_file=/home/phase3_sample_data/HandInfo.csv
```

Steps involved in running individual tasks:

Task 3:

1. Populate dataset details in **dataset.txt**
2. Run **initialSteps.py** if necessary
3. Run **phase3_task3.py**
4. Input as prompted

To the end of this execution a html file **render_task3.html** is generated which shows the output of each dominant image along with its pagerank.

Task 4:

1. Populate dataset details in **dataset_task4.txt**
2. Run **initialSteps.py** on both Labelled and Unlabelled datasets
3. Run **phase3_task4.py**
4. Input as prompted

To the end of this execution a html file **render_task4.html** is generated which shows the output of each dominant image along with its classified label.

Task 3 - Personalised Page Rank

In this task, we take as input -

1. Value ‘k’ which is used to create the similarity matrix
2. 3 Image ID’s which are used to calculate the PPR
3. Value ‘K’ which is used to output the ‘K’ dominant images.

The steps involved in this task are as follows:

1. Take inputs from the user.
2. Fetch feature vectors from Database
3. Build Similarity Graph:
   Similarity graph was built by calculating the cosine similarity between the HoG vectors.
4. Calculate PPR for the similarity graph with respect to the 3 image ID’s given by the user.
5. Output the K most dominant images. Visualise.

Calculating PPR:

1. Build a teleportation vector by setting the corresponding entries for the user given images to ⅓ (With 3 being the length of the seed set).
2. Set the value of alpha (a, we used 0.85)
3. Build similarity matrix with the cosine similarity values between images. This will be a **k x k matrix.**
4. Now calculate PPR for each image using:

$$\vec{\pi} = (1-\beta)\mathbf{T}_G \times \vec{\pi} + \beta\vec{s},$$

   Which is translated in code to:

```
ppr = np.matmul(np.linalg.inv(np.subtract(I,(a*T))),(1-a)*tp_vector)
```

The output of this task is rendered in a html file named render_task3.html. This contains the K most dominant images based on the computer PPR.

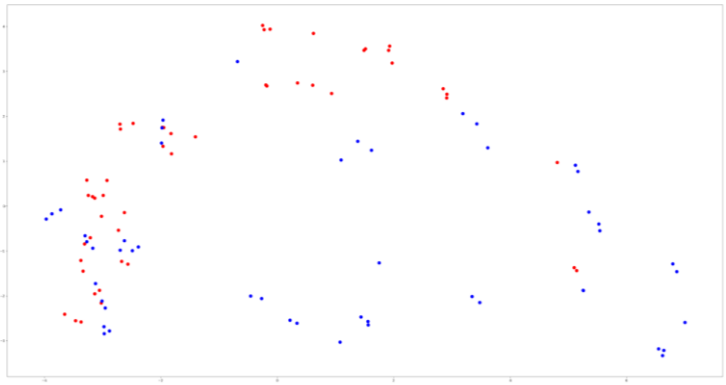
Task 4 - Image Classification

In this task we have implemented three classifiers which take as input a labelled set of images, trains a model and predicts an unlabelled set of images. The models we have worked on are as follows:

Support Vector Machines (SVM)

In our implementation of SVM, we adopted the below procedure:

1. Fetch feature vectors from the Database for both labelled and unlabelled images and store them in two different Pandas Dataframes.
2. Compute Latent Vectors using PCA for ease of computation.
3. Divide the Labelled Dataframe into two parts : dorsalDF(Dorsal Dataframe) and palmarDF(Palmar Dataframe)
4. Now, compute the pairwise cosine-similarity for all the vectors in both the dataframes (Dorsal and Palmar) and find the minimum distance between any two points. These points will serve as our margins
5. A pictorial representation of our data is as follows:



   All the red points correspond to dorsal images and all the blue points correspond to palmar images.
6. Now that we have the margin vectors, we use the formula w.x + b = 0 to find the hyperplane and identify the class an unlabelled image belongs to.
7. The value of b is computed by 1/||w||.

This approach has yielded us an accuracy of 60%. The data is clearly not a linear distribution and the inaccuracy is explained by the distribution of data.

PPR Based Classifier

The implementation of a PPR based classifier was achieved as below:

1. Fetch feature vectors from the Database for both labelled and unlabelled images and store them in two different Pandas Dataframes.
2. Compute Latent Vectors using PCA for ease of computation.

3. Divide the Labelled Dataframe into two parts : dorsalDF(Dorsal Dataframe) and palmarDF(Palmar Dataframe)
4. Now, append the unlabelled data frames to each of the dorsalDF and palmarDF.
5. Now we have the pagerank of a given image in the dorsalDF and palmarDF. Based on which is higher, we classify the image accordingly.

The intuition is that a node has a higher page rank when it is surrounded by node similar to it.

Using this approach, we were able to achieve a 57% accuracy in classifying the images to their respective labels.

Decision Tree Classifier:

1. Extract the feature vectors for the folder of the labelled images and the unlabelled images using HOG and obtain the transformed values of the features in 60 dimensions using PCA.

2. Store those values in two data frames for labelled and unlabelled images respectively.

3. The labelled images feature vectors are passed to build the tree.

4. The tree is built by splitting the tree by finding the best split to partition based on the question at the node to split and the Gini index.

5. Now, information gain is calculated by subtracting the uncertainty of the starting node and weighted impurity of the two child nodes.

6. The tree is recursively built based on steps 4, 5 and 6 till a homogenous class label is obtained at the leaf by updating the gain.

7. After this training step, the feature vectors of the unlabelled images are passed onto this tree to obtain their corresponding class labels.

Using this approach, we were able to achieve 52% accuracy in classifying the folder of unlabelled into dorsal or palmar.

Results

In this section, we shall present the output we got for the queries given:

Task 3

Query 1

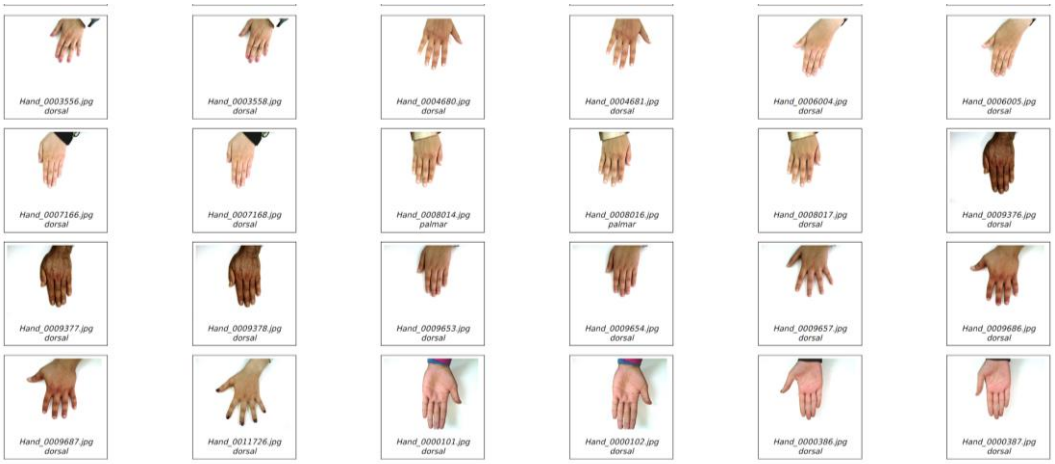
Query 2

Task 4

Query 1 : SVM


```
Hand_0010474.jpg palmar palmar
Hand_0010833.jpg dorsal palmar
Hand_0010834.jpg dorsal palmar
Hand_0011455.jpg dorsal palmar
Accuracy = 0.6
```

Query 2 : Decision Tree



```
Hand_0008886.jpg palmar palmar
Hand_0009791.jpg palmar dorsal
Hand_0009810.jpg palmar dorsal
Hand_0010471.jpg palmar dorsal
Hand_0010474.jpg palmar dorsal
Hand_0010833.jpg palmar dorsal
Hand_0010834.jpg palmar dorsal
Hand_0011455.jpg palmar dorsal
Accuracy = 0.52
```

Query 3 : PPR

```
Hand_0009791.jpg dorsal dorsal
Hand_0009810.jpg dorsal palmar
Hand_0010471.jpg palmar palmar
Hand_0010474.jpg palmar dorsal
Hand_0010833.jpg dorsal palmar
Hand_0010834.jpg dorsal palmar
Hand_0011455.jpg dorsal palmar
Accuracy = 0.56
```

References

[1] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[2] https://cs.stanford.edu/people/plofgren/bidirectional_ppr_thesis.pdf

[3] https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[4] http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html

[5] https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/