University of Bahrain

College of Information Technology

Department of Computer Science

ITCS474: Information Retrieval

Sec.01

Term Project:

Image Classification Using Support Vector Machines (SVM)

NAME:

SAYED ALQASIM DHEYA ALI

MAHMOUD MOHAMMAD SALEH

I.D.#: 20147349

20150884

Learning Objectives:

1. To design an image classifier based on Support Vector Machines.
2. To effectively extract features from images that will positively impact classification.
3. To know how classification is affected by the training sets and features fed into ML classifiers.

Features:

- The program implements a deep neural network by the name of "SqeeuzeNet" that is praised for its compactness as it allows easier fit in computer memory as well as network propagation. It is using this neural net that the extracted features of the used dataset will be parsed and a 5D tensor containing vectorized representations of the images is returned. A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) shape. In this particular implementation and because auto-sampling is turned off, it will be extracting 512 features for each image in the dataset.

- The program implements SVM as the classifier to be trained on the features extracted by the neural network, then scored at the end to know how accurate the classifier was based on the features it trained in.

- Method:

  - A previously prepared ".csv" file is used to hold the file names and labels pertaining to the dataset in question. The program requires a path to such a file is known to it, as well as a path to where all images reside.

  - The featurizer is initialized with the appropriate parameters including the model which is "SqueezeNet" in this case, the downsampling flag which will be set to "False" allowing the featurizer to extract 512 features, and the depth which will be set to "1".

  - When the featurizer is ready it is now possible to invoke the featurize function that will be extracting the features. The function is passed the column name of the column holding the image names as in the ".csv" file, the path to the images of the dataset, and the path to the ".csv" file.

  - The result of the featurize function is a dataframe containing the original csv, along with the generated features appended to the appropriate row, corresponding to each image. There is also an "images_missing" column, to track which images were missing. Missing image features are generated on a matrix of zeros.

  - If there are images in the directory that aren't contained in the CSV, or image names in the CSV that aren't in the directory, or even files that aren't valid image files in the directory, have no fear– the featurizer will only try to vectorize valid images that are present in both the CSV and the directory. Any images present in the CSV but not the directory will be given zero vectors, and the order of the image column from the CSV is considered the canonical order for the images.

- Now the dataset has been fully featurized. The features are saved under the *featurized_data* attribute if the save_features argument was set to "*True*" in the *featurize* function.

- The dataframe is saved in CSV form by calling the *save_csv* function on the featurizer itself.

- One can simply test the performance of a linear SVM classifier over the featurized data, but first, the training and test sets need to be built. For training the data, 500 images of each class were selected. For the test data, 100 images of each class were selected.

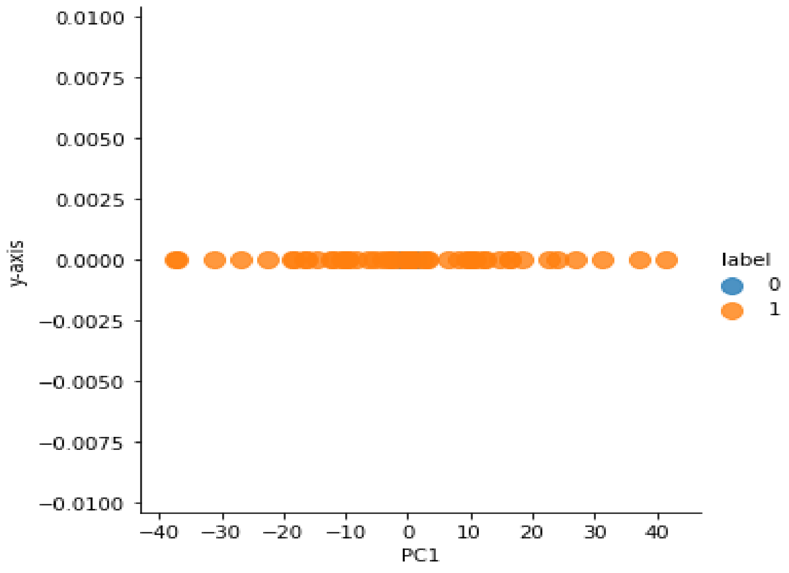
- The SVM classifier is initialized and fitted with the training data, then scored on the testing data.

- The results were good considering the size of the dataset, 92.5% accuracy is quite a feat.

- The training and testing sets were reduced to conserve what little memory space that was left. Even though the neural network used is fast it was not so easy on the memory.

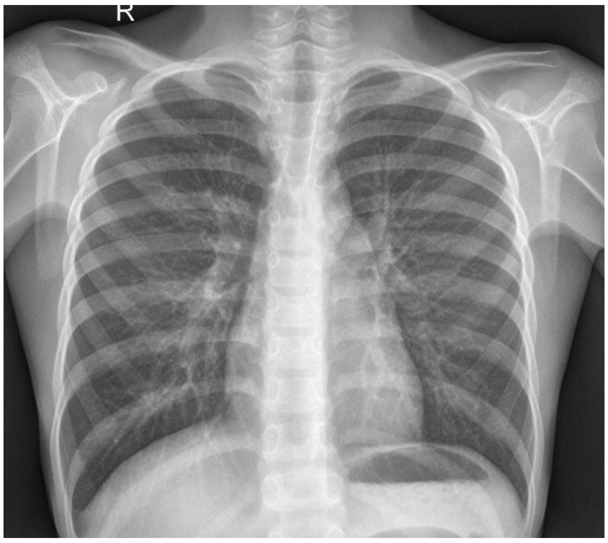
- Honorable Mentions:
  - The first approach that did not end up being used was Principle Component Analysis (PCA). Even though it was promising, the result of dimensionality reduction of the image matrix had many overlaps in features in both classes and we could no longer rely on PCA for feature extraction.

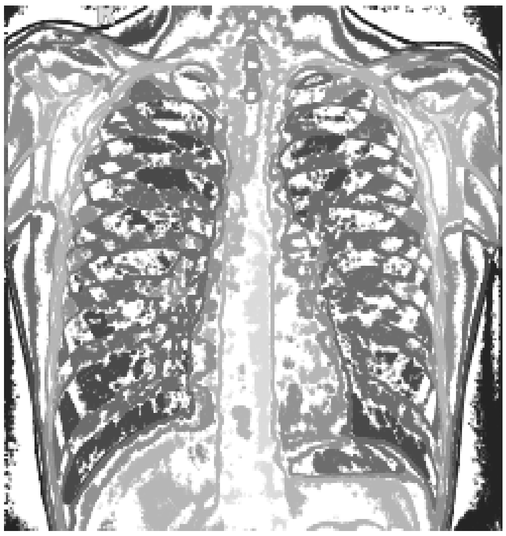
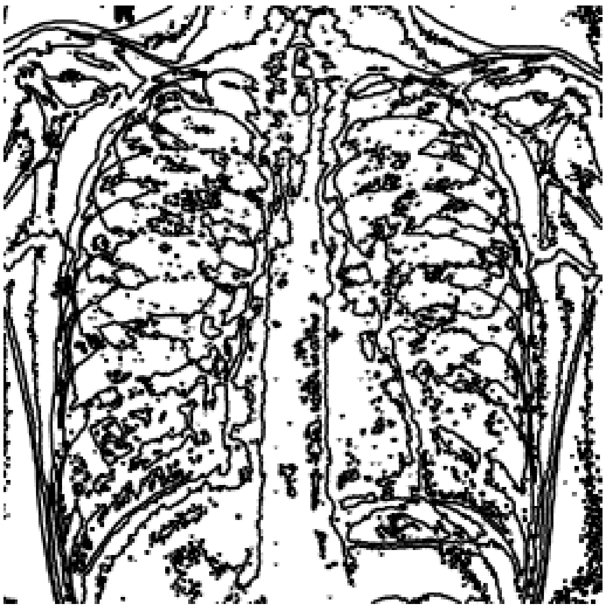
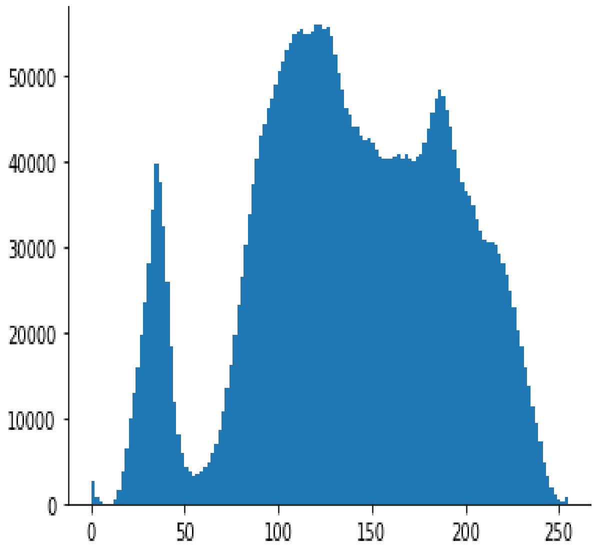
```
In [53]: sns.lmplot('PC1', 'y-axis', data=sklearn_result, fit_reg= False,
Out[53]: <seaborn.axisgrid.FacetGrid at 0x138cc0eca58>
```



  - The second approach that was almost used was using as features the Eigen Vectors of the image matrix. Sadly when it was almost done, a runtime error resulted from incompatibilities in the dimensions of the image matrices prevented further developments.

- Another approach was to get the contours of the images to make the lines and change in color more apparent, then use the histogram of the contoured image as its features. In the end, it was too much trouble to continue and it was dropped.



- Conclusion:
  - The classifier performs quite well, even if only a handful of training data were used in this case because of H/W limitations. Of course, the bigger the training data the better the results, so even though classifiers are powerful tools for classification and analysis, they are only as good as their training material and feature extraction quality and precision.