CIS519: Applied Machine Learning **Spring 2018**

Homework 5

Handed Out: April 18th, 2018 *Due: April* 26th, 2018, *11:59 PM*

1 Image Clustering [60 points]

In this problem, you will implement several clustering models using EM based algorithms to perform image classification in an unsupervised setting. You will evaluate your clustering performance using two different approaches and identify the significance of each performance measure. You will also perform clustering on a reduced feature representation using Principal Component analysis and compare the significance of the reduced representation.

Please note that we are leaving some of the implementation details to you. In particular, you can decide how to representation the data internally, so that your implementation of the algorithm is as efficient as possible. **Note, however, that you are required to implement the algorithm yourself, and not use existing implementations, unless we specify it explicitly.**

1.1 Dataset

For the experiments, you will use a subset of the *Pascal VOC* dataset[1] . The dataset is provided to you. The provided VOC dataset consists of ~1250 images spread across 10 categories of objects. There are roughly 75 images per class for training and roughly 50 images per class for evaluating. Each datapoint actually consists of a 4096x1 dimensional vector representation of an image obtained using the Overfeat[2] network that was pre-trained on the ImageNet dataset, the largest collection of annotated images to date. The dataset is also provided with the labels. (Note: Use these labels to make meaningful conclusions of your experimental results). Please see the provided code for more details on the exact format of the dataset.

For the VOC dataset, you are provided 2 splits - one containing the training data and the other containing the test data. You will train your clustering model using the training data without using the labels, assign each cluster a label using the provided training labels, and then predict for the test data using the model. You will evaluate the performance on the training data and test data to see how the clustering model helps for the purpose of classification.

1.2 Evaluation

To evaluate each of the clustering methods below, please use the following two metrics:

[1] http://host.robots.ox.ac.uk/pascal/VOC/

[2] https://github.com/sermanet/OverFeat

1. **Purity**: To compute the *purity* of a set of clusters, assign every cluster the most frequent label in the cluster. For example, if a cluster has 3 goldfinches, 2 robins, and 1 ostrich, we would assign the cluster the label of goldfinch. Then, the purity score is simply the proportion of correctly labelled datapoints. Note that higher purity is better, but also that high training purity is extremely easy to achieve when the number of clusters is large.

2. **Rand Index**[3]: Given the true labels and our clustering, we can compute the *Rand index* by first computing the following two values:

   - TP (true positives), the number of pairs of datapoints that share the same true label and are assigned to the same cluster.
   - TN (true negatives), the number of pairs of datapoints that don't share the same label and are *not* assigned to the same cluster.

   For a dataset with N datapoints, the Rand index is simply the ratio

   $$\frac{TP + TN}{\binom{N}{2}},$$

   that is, fraction of number of pairs "correctly labelled" in the sense above. As before, note that higher Rand index is better. This metric does not necessarily go up (even in training) when the number of clusters increases.

1.3 Experiment 1 [10 points]

In this experiment, you will run a simple clustering algorithm called KMeans clustering on the VOC dataset. That is, you will use the training data to cluster the datapoints into a pre-specified number of clusters, calculate a label among the 10 labels for each cluster that maximizes the correct predictions on the training data itself and then predict the class for the test dataset by identifying the cluster each data corresponds to. . Use `sklearn.cluster.KMeans` to cluster the training images.

The procedure is to take all the training data (irrespective of the labels) and cluster them using the KMeans clustering algorithm. For each cluster, identify a label such that the number of correct predictions (on the training data) is maximum. Evaluate the purity and Rand Index for both the training data and test data by predicting the cluster the data corresponds to and using its label as estimated in the previous step.

For this experiment, you will cluster with K=10, K=20, K=100 and K=500 clusters for the VOC dataset. *Compute the two performance measures, purity and Rand Index for the 4 different cases. Compare the results and report the values of the performance measures.*

1.4 Experiment 2 [10 points]

In this experiment, you will use a Gaussian Mixture Model instead of KMeans. Gaussian mixture models is an EM based algorithm similar to KMeans but instead of fitting only the cluster centers, it fits multiple gaussian models which are identified by mean and covariance.

[3] https://en.wikipedia.org/wiki/Rand_index

Use `sklearn.mixture.GaussianMixture` to cluster the training images. Please set the `covariance_type` parameter to `'diagonal'`. Perform the experiments following the exact procedure as KMeans. *Compare the results and report the values of the performance measures for the 4 different number of clusters.*

1.5 Experiment 3 [20 points]

In this experiment, you will first perform dimensionality reduction before performing clustering on the reduced feature representation. For this, you will use Principal Component Analysis. (You may use `sklearn.decomposition.PCA`.) The idea is to apply PCA and reduce from 4096 dimensions to extremely small dimensions that capture most of the information contained in the data. You can think of the retained dimensions as some linear combination of the original 4096 dimensions along which most of the data lies. For reducing the dimensionality of the test data, transform the test data using the PCA model constructed using the training data.

In this experiment, you will apply PCA to extract top 40 dimensions, reconstruct the data along this new 40 dimensional space and perform both KMeans and GMM clustering on the dimensionally reduced dataset. In this experiment, you should try K=10 and K=20. *Compare the results and report the values of the performance measures for the 2 different cases. Report the influence on the classification performance after clustering on PCA reduced dataset. Additionally, report the variance captured (Check the documentation for `sklearn.decomposition.PCA`) and % data compression achieved by applying PCA.*

Visualizing the dataset is really useful to help understand the data and identify appropriate algorithm. PCA allows us to visualize high dimensional dataset to a certain extent. Here, we will construct a scatter plot using the first 3 PCA components for each datapoint. In other words, the first component is treated as x coordinate, the second as y coordinate and the third as z coordinate of the plot. To obtain these coordinates, first perform PCA with n_components = 3 on each datapoint. Take the reconstructed data point in the 3 dimensional space and plot it using a 3D scatter plot tool (use `mpl_toolkits.mplot3d`). Use different colors for each label to identify which datapoint corresponds to which label. *Based on your observation of the data, suggest which clustering model would be more suitable for modeling this data.*

1.6 (Optional: Extra Credit) Experiment 3.5 [5 points]

You are encouraged to implement the Principal Component Analysis for dimensionality reduction yourself for extra credit. You can perform the same set of experiments as in Experiment 3 with your implementation but projecting the data to fewest number of dimensions such that it captures 95% of the variance instead of hard-coding to 40 dimensions. Report your results with your new PCA implementation.

1.7 (Optional: Extra Credit) Experiment 4 [5 Points]

We will now try to compare the performance of clustering with the performance of the linear classifier methods. The procedure is to apply PCA on the training data to get a 40 dimensional representation for each data point, fit a (multi-class) linear classifier model using the SVM algorithm on the training data and evaluate the performance on the test data. You should compare the training and test performance of SVM to the best performing clustering model. In the report, include the accuracy measure for the 2 models. Use purity as the evaluation method for the clustering model.

1.8 What to Report: [20 points]

The purity and rand index for all four clustering models (KMeans and Gaussian Mixture models on the original 4096 dimensional data as well as PCA-reduced data) for the specified number of clusters in each experiment . In addition, for experiment 3, please include your observations about the performance of the PCA algorithm. Please explain any trends you see, and attempt to explain why they occur. Feel free to use thumbnail images in your report to justify your observations but do not clutter the report with a lot of images. Finally, please also note any issues you ran into during the course of this assignment.

2 Tree Dependent Distributions [40 points]

A tree dependent distribution is a probability distribution over n variables, $\{x_1, \dots, x_n\}$ that can be represented as a tree built over n nodes corresponding to the variables. If there is a directed edge from variable x_i to variable x_j, then x_i is said to be the parent of x_j. Each directed edge $\langle x_i, x_j \rangle$ has a weight that indicates the conditional probability $\Pr(x_j \mid x_i)$. In addition, we also have probability $\Pr(x_r)$ associated with the root node x_r. While computing joint probabilities over tree-dependent distributions, we assume that a node is independent of all its non-descendants given its parent. For instance, in our example above, x_j is independent of all its non-descendants given x_i.

To learn a tree-dependent distribution, we need to learn three things: the structure of the tree, the conditional probabilities on the edges of the tree, and the probabilities on the nodes. Assume that you have an algorithm to learn an *undirected* tree T with all required probabilities. To clarify, for all *undirected* edges $\langle x_i, x_j \rangle$, we have learned both probabilities, $\Pr(x_i \mid x_j)$ and $\Pr(x_j \mid x_i)$. (We discussed this algorithm in class.) The only aspect missing to convert this undirected tree to a directed one is determining the directionality of edges .

However, it is okay to not learn the directionality of the edges explicitly. In this problem, you will show that choosing any arbitrary node as the root and directing all edges away from it is sufficient, and that two directed trees obtained this way from the same underlying undirected tree T are equivalent.

1. **[14 points]** State exactly what is meant by the statement: "*The two directed trees obtained from T are equivalent.*"

2. **[26 points]** Show that no matter which node in T is chosen as the root for the "direction" stage, the resulting directed trees are all equivalent (based on your definition above).

Submission Instructions

Please submit your report and code for problem 1, as well as your answer to problem 2, **on Canvas**.