|  |  |
| --- | --- |
| COMP6223 Computer Vision (MSc) Prof. Mark Nixon, Dr. Jonathon Hare  **Coursework 3:**  **Scene Recognition** | 15/12/2017  Ganiyu Ibraheem, gai1u17  Philipp Seybold, ps1c17 |

This report investigates three different classifiers and their associated feature for scene recognition. It is structured into three sections for each run of the classifiers on given test data. While the first two runs are briefly described in their approach and implementation, the third run will additionally include more detailed explanations on its used methods in comparisons with the first two classifiers.

1. K-Nearest-Neighbour Classifier with “Tiny Image” Feature

Approach

A k-nearest-neighbours (kNN) algorithm identifies the *k* nearest neighbours in a *n*-dimensional space of a given vector *v*. The neighbours consist of labelled training data each transformed into a vector and arranged in some data structure that fits the space dimension best. The class that *v* has the most number of nearest neighbours to is assigned to *v*. An effective way to facilitate multidimensional search is a *n*-dimensional tree that partitions the space in a number of partitions so subtrees can be left out if the current *k*th distance is smaller than the subtree itself. This allows *O(log(n))* complex searches in best and *O(n)* in worst cases. Drawbacks are its complexity depending on the chosen data structure for large datasets, possibly dominating classes when they are represented more often in the training data and performance is dependent on the number of dimensions. Advantageous on the other hand are the zero cost for the learning process and no model concepts need to be respected.

As for the tiny-image feature, one resizes the image to a fixed resolution, suggested is 16x16, and transforms them into a vector by concatenating each image row. It is also suggested to change the vectors to have zero mean and unit length for improved results. When classifying test data, the tiny-image feature is simply compared to the training data through the kNN algorithm. As biggest drawbacks of a plain feature like this one is that it discards the high frequency image content and is not shift invariant.

Implementation

… When implementation is finalized and checked

* Choose the optimal k-value for the classifier? We: 5
* K-d trees?
* Feature vectors ok as histograms?

1. Linear Classifiers with Bag-of-Visual-Words Feature

Approach

Set of Linear SVMs (15 one-vs-all classifiers)

1. Class a vs. non-class a for all 15 classification problems
2. Linear classifier …
3. Test on the chosen lambda, accuracy sensitive to it (Acc.: 60 – 70 %)

A Bag-of-visual words feature is a bag data structure containing a fixed number of local feature clusters (“words”) forming the vocabulary of the classifier. These clusters are determined with kmeans clustering over a large number of sampled features …

1. Based on Fixed sized densely-sampled pixel patches
2. Start: 8x8 patches sampled every 4 pixels in x and y directions
3. A sample of these should be clustered using K-Means to learn a vocabulary (try ~500 clusters to start)
   1. Consider mean-centring and normalising each patch before clustering/quantisation
   2. **NO SIFT features here** - just take the pixels from the patches and flatten them into a vector & then use vector quantisation to map each patch to a visual word

Implementation

1. CNNs with SVMs and DenseSiFT Features

Approach

Implementation