|  |  |
| --- | --- |
| 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

Firstly, each image was resized to 16x16, then concatenated into a feature vector (1-by-256) which was then transformed into a normalised histogram with 16 dimensions. Thereafter, a kNN was trained using the image histograms as inputs and their labels as targets. For the learning process, an optimal was calculated by cross-validation which often lied around and a *n*-d tree represented the data structure. Predictions were performed as usual: the feature vector was extracted from the test image and then classified by the trained kNN. More details can be found in the code comments.

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

Approach

The classifier will be a set of linear support vector machines, acting as one-vs-all classifiers. The decisions made by these classifiers can be expressed as “is v in class A or non-A”. For a number of 15 different classes like in the given assignment, this classification is repeated 15 times after what v is assigned to the class with the highest belonging probability. During the training-phase each of them learns for one category (class) its local feature composition by a set of labelled images. Afterwards, it is able to identify the chance a given image belongs to its class or not. Their advantage over kNN classifiers is their ability to detect dimensions of features that are less relevant and would otherwise down weight a decision. They can also be used for non-linear classification utilising the kernel trick.

A Bag-of-visual words (BoVW) feature is a bag data structure containing a fixed number of local feature clusters forming the “vocabulary” of the classifier. These clusters are determined with K-Means clustering over a large number of sampled features (“visual words”) from training images. These samples consist of fixed size densely-sampled pixel patches that can be either transformed into SIFT features (which will be done in section 3 of this report) or any other local feature that might fit well for this context. After an image is decomposed into local descriptors, it is counted how many of these features fall into each cluster in the visual word vocabulary. From that distribution, the likelihood for a category is computed by the classifier. Advantages and drawbacks depend on the chosen local feature.

Implementation

Following the suggestions in the assignment, these methods and parameters were used for the second run: The patches’ size was set to 8x8, they were sampled every 4 pixels in the x and y direction and transformed into normalised histograms (like in section 1) for each image. Together with the images’ labels form they the visual words dictionary. Afterwards, the clusters were computed using K-Means with 10 random features per training image and the number of clusters set to . Then, for each image, the Bag-of-Words is created by mapping the image features to the closest cluster mean and counting for each cluster how many features were mapped onto it. These BoVWs were subsequently used as input together with their images’ labels as target for training the support vector machines (SVM). As for the training, for each classification problem did its SVM receive all BoVWs for that label as positive examples and the rest of the input as negatives. The SVMs used a standard regularisation factor of 1. When it came to the prediction, the image’s inputted Bag-of-Words was given to every label classifier which then computed the probability of that image belonging to its class. The highest probable label was returned as result. Further details can be found in the code comments.

1. CNNs with Non-Linear SVMs and Dense-SIFT Features

Approach

SIFT (Scale-invariant feature transform ) Features -> detects and describes local features.

Besides the advantage of well representing an image, SIFT features might bring along complexity problems when facing large resolution images and thus making it necessary to use k-means to learn a million clusters in 128 dimensions (!) from 10’s millions of features.

Implementation

1. Appendix A: Misclassification Error for k-neighbours of kNN

