Chapter 1

Introduction

k nearest neighbours (kNN) is one of the oldest and simplest classification methods. It has its origins in an unpublished technical report by Fix and Hodges (1989) and since then it became standard textbook material (Russell et al., 1996; Mitchell, 1997; Bishop, 2006). In the last 50 years, kNN was present in most of the machine learning related fields (pattern recognition, statistical classification, data mining, information retrieval, data compression) and it plays a role in many applications (e.g., face recognition, plagiarism detection, vector quantization).

The idea behind kNN is intuitive and straightforward: classify a given point according to a majority vote of its neighbours; the selected class is the one that is the most represented amongst the k nearest neighbours. This is easy to implement and it usually represents a good way to approach new problems or data sets. Despite its simplicity kNN is still a powerful tool, performing surprisingly well in practice (Holte, 1993).

Yet there are also other characteristics that make kNN an interesting method. First of all, kNN makes no assumptions about the underlying structure of the data. No a priori knowledge is needed beforehand, but we let the data "speak for itself". The accuracy increases with the number of points in the data set and, in fact, it approaches Bayes optimality as the cardinality of the training set goes to infinity and k is sufficiently large (Cover and Hart, 1967). Secondly, kNN is able to represent complex functions with non-linear decision boundaries by using only simple local approximations Lastly, kNN operates in a "lazy" fashion. The training data set is just stored and its use is delayed until testing. The quasi-inexistent training allows to easily add new training examples.

kNN has some drawbacks that influence both the computational performance and the quality of its predictions. Since kNN has to store all the exemplars, the memory requirements are directly proportional with the number of instances in the training set. The cardinality of the data also influences the method's speed. All computations are done at testing time, making kNN painfully slow when applied on large data sets.

The accuracy of kNN is closely related to how we define what "close" and "far" mean for our data set and task. Mathematically, the notion of dissimilarity is incorporated into kNN by using different distance metrics. Usually the standard Euclidean metric is not satisfactory and the aim is to find that particular metric that gives the best results on our data set for a given task (section 2.1). There is an entire literature that tries to come up with possible solutions (section 2.2).

This thesis focuses on neighbourhood component analysis (NCA; Goldberger et al., 2004). NCA method learns the metric that maximizes the expected number of correctly classified points (section 3.1). Using the NCA metric with kNN usually improves the performance over simple kNN, since we use the label information to construct a suitable metric that selects the relevant attributes. If we restrict the metric to be low ranked we can find its associated linear projection. A low dimensional representation of the original data reduces the storage needs and the computational expense at test time . Also it alleviates some of the concerns that have been raised regarding the usefulness of nearest neighbours methods for high dimensional data (Beyer et al., 1999; Hinneburg et al., 2000). The curse of dimensionality arises for kNN, because the distances become indiscernible for many dimensions. For a given distribution the maximum and the minimum distance between points become equal in the limit of dimensions. NCA proves to be an elegant answer for the above issues and, consequently, it was successfully used in a variety of applications: face recognition (Butman and Goldberger, 2008), hyperspectral image classification (Weizman and Goldberger, 2007), acoustic modelling (Singh-Miller, 2010) and even reinformcent learning (Keller et al., 2006).

However, the method introduces a consistent training time. The objective function needs the pairwise distances of the points, so the function evaluation is quadratic in the number of data points. Also the optimization process is iterative. For large data sets, NCA training becomes prohibitively slow. For example, Weinberger et al. (2006) reported that the original implementation ran out of RAM and Weizman and Goldberger (2007) had to use only 10% of the data set in order to successfully train the model. There is only little previous work that uses NCA for large scaled applications. One example is (Singh-Miller, 2010), who parallelizes the computations across multiple computers and adopts various heuristics to prune terms of the objective function and the gradient.

The main aim of this project is to reduce NCA training time without significant loses in accuracy (chapter 4). We start our investigation with the most simple ideas: use only a subset of the data set (section 4.1) or train the metric on different mini-batches subsampled from the data set (section 4.2). This last idea can be further refined by using clustered mini-batches. Also we present an alternative mini-batch algorithm (section 4.3) that decreases the theoretical cost.

These methods can achieve further speedings if we use approximations of the objective function. Simple approximation ideas (such as ignoring the points which are farther away than a certain distance from the current point) were mentioned in the original paper (Goldberger et al., 2004) and they are re-iterated by Weinberger and Tesauro (2007) and Singh-Miller (2010). We present a more principled approach that borrows ideas from fast kernel density estimation problems (Deng and Moore, 1995; Gray and Moore, 2003; Alexander Gray, 2003). We first recast NCA into a class-conditional kernel density estimation framework (section 3.2) and next we present how fast density estimation is done using a space partitioning structure, such as k-d trees (section 4.4). Similar work of fast metric learning is (Weinberger and Saul, 2008) and it uses ball trees (Omohundro, 1989) to accelerate a different distance metric technique, large margin nearest neighbour (LMNN; Weinberger et al., 2006). However, LMNN is different from NCA and the way in which ball trees are applied also differs from our approach.

An alternative for the approximation method is to change the model such that it allows exact and efficient computations. In the kernel density estimation model, we can use compact support kernels instead of the standard Gaussian functions. The expense is reduced because only the points that lie in the compact support are used for estimating the density (section 4.5).

We evaluate the proposed techniques in terms of accuracy and speed (chapter 5). Also we provide low dimensional representations of the projected data. The results are promising showing that we can obtain considerable speed-ups for large data sets while retaining almost the same accuracy as in the classic case. For small and medium sized data sets the accuracy and the time spent are similar to the standard NCA.