Advanced Vision: Assignment 1

Name: Timothy S. Stirling
Matriculation Number: 0199431

AV-4 Undergraduate Assignment

Date: 03.02.06

Numerical Digit Character Recognition

Numerical Digit Character Recognition is a subset of the better known Optical Character Recognition, commonly abbreviated to OCR. OCR systems have readily been used in a variety of situations and have been highly successful in converting typewritten or handwritten text into a computer readable format. Typical applications for OCR systems include uses by the United States Postal Service for automatically reading the address of letters for sorting purposes. While OCR is concerned with the the full range of alphanumeric characters as well as non-alphanumeric characters such as punctuation marks, Numerical Digit Character Recognition is concerned only with the 10 numeric digits 0-9. This simplification does not prevent the system from having a wide applicability to real world problems and associated commercial uptake. In many situations only a string of numeric characters is required to be recognised, e.g. the US zip(post)code of addresses is composed entirely of numeric characters and the recognition of the zipcode alone is enough to approximately sort the letters into separate destinations. Another common use of numerical character recognition is reading bar-codes of products. In any case, the software tools and methods required to perform numerical recognition are the same as for the unconstrained version and provides more accurate results.

History

The more general OCR approach has a surprisingly long history dating back to c. 1951 when David Shepard and Harvey Cook built a machine capable of converting printed messages into machine language for computer processing with the first commercial system sold in 1955 [1]. In 1965 the US Postal Service started using this technology to read the name and address on letters and printed a routing bar code based on the postcode which could be read by cheaper simpler sorting systems. The Canadian postal service followed suit in 1971. These days OCR technology is commonplace in the office for scanning and converting letters and documents into a computer readable format using standard flatbed scanners and cheap software. More recently, with the rise of small portable computers such as PDAs, the need for recognising handwritten text in real-time has become a research field of much interest and is an area where much work still needs to be done to obtain acceptably small error rates and where more complex algorithms are required.

Image Data

The initial step in performing character recognition is acquiring the image data. This is invariant to the algorithms used and discussed here, and is also problem specific. Commonly the image data is obtained by standard scanners or even cameras and rarely is any special equipment used. Algorithms should be made to work regardless of the lighting conditions, paper type and colour or ink used, i.e. they are noise tolerant. From a basic image containing the text, it is necessary to process this image in order to separate individual characters. With non-cursive (separate character) text, as is the case with numeric characters, this process is relatively simple but is not described here. Furthermore, it is usually necessary to perform image pre-processing in order to make the data translation, rotation and scale invariant as these effects can seriously disrupt recognition algorithms which mostly work by comparing image features that are expected to be in a common image position. The image should also be converted into grey-scale as colour information is of little concern, usually by averaging the RGB values. The resolution of an extracted character from a typically scanned image (e.g. 300dpi) can still be fairly large, e.g. 64x64, and hence should contain moderate detail and a lot of information about the character. However, this image will also contain noise, both environmental and sensory which we wish to ignore during the recognition process. This grey scale image can be converted into a vector representation in order to simplify the mathematics, and in this instance would be of dimension $64^2 = 4096$. It is clear that this is very high dimensional data that can be CPU and memory intensive to process, especially for a real-time character recognition process

running on a computationally weak PDA system. It is therefore common practice to apply some form of dimension reduction to reduce the dimensionality and thus the CPU resources required.

Dimension Reduction

Here I will only describe the basics of this field as it is beyond the scope of this article and is a large and encompassing research area in itself, with no straightforward answers. The aim of Dimension Reduction (or data reduction) in this case is to throw away unimportant parts of the data, hopefully including the noise, while retaining the information that is important for applying character recognition. As already described, a common image size for a single character can be quite large and result in a very large vector representation. Much less information is required to recognise a character so it is appealing to reduce the dimensionality leaving only those dimensions that are necessary to do recognition. These are often called Feature representations because what remains in essence are the important features of the image. The blank space around the edges and corners provide little information, but there is a way of knowing more accurately which dimensions are required. Principle Component Analysis (PCA) is a well known linear dimension reduction technique often used in machine learning and also in variety of statistical applications. This technique is linear in that it does not analyse interactions between different dimensions which are inherent in the character image, but several non-linear techniques also exist.

The PCA algorithm assumes that the data lies close to a hyperplane (figure 1) and thus the data points can be represented by vectors that span the hyperplane alone [2]. The aim is to represent the high dimensional data with a low dimensional representation that will approximate the data and provide the minimum error.

$$\mathbf{x} \approx \mathbf{c} + \sum_{i=1}^{M} w_i \mathbf{b}^i$$

The optimal lower dimensional representation is given by projecting the data onto the eigenvectors of the covariance matrix of the training data with the largest eigenvalues, a technique common to the related statistical technique of Factor Analysis. See [2] for algorithmic details and associated mathematical proofs.

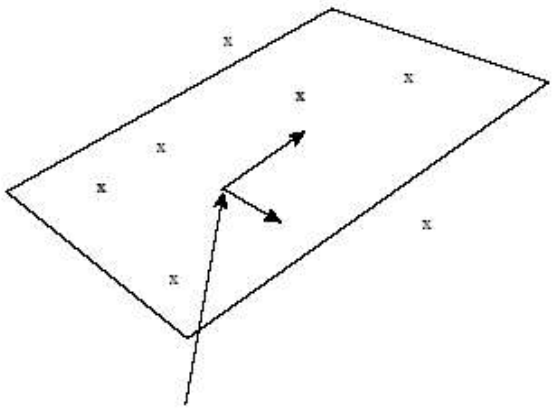

Figure 1: In linear dimension reduction we hope that data that lies in a high dimensional space lies close to a hyperplane that can be spanned by a smaller number of vectors [2].

Whatever technique is used we should now have a lower dimensional vector representation that contains the important dimensions (or features) and is appropriate for recognition purposes. Character recognition is a form of classification, i.e. to assign novel data to the class it most closely matches, and so a variety of regular classification algorithms can be used. As with all recognition

algorithms, training, testing and validation datasets are required to allow the learning of certain parameters.

Nearest Neighbour Classification

This algorithm is basic in design and implementation and yet is also widely used within machine learning and can provide good results. The idea behind the algorithm is that an item x is most likely to be of the same class as that of the item to which it is the most similar (figure 2). Similarity in this instance is often defined as the squared euclidean distance between the two vector representations. The squared euclidean distances between two points x and y is $d(x, y) = (x - y)^T (x - y)$ [3]. However, often other similarity or distance measures are used such as the Mahalanobis Distance [3].

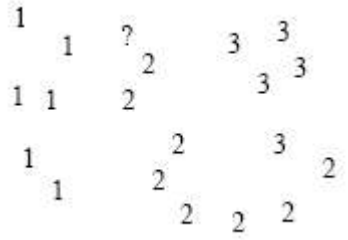

Figure 2
Figure 1: In nearest neighbour classification, a new vector with an unknown label,?, is assigned the label of the vector in the training set which is nearest. In this case, the vector will be classified as a 2. [3]

The Algorithm:

> To classify a new vector x, given a set of training data (x^μ, c^μ), $\mu = 1,...,P$: where x^μ represents the set of training data points, c^μ is the set of class labels for each data point in the training set, e.g. 0-9 for numerical digit characters, and P is the size of the training set.

1. Calculate the distance between the test point x and each of the stored data points, $d^\mu = d(x, x^\mu)$.
2. Find the training point x^{μ^*} which is closest to x by finding that μ^* such that $d^{\mu^*} < d^\mu$ for all $\mu = 1,....,P$.
3. Assign the class label $c(x) = c^{\mu^*}$.

When there are two or more `equidistant' points with different class labels, the most numerous class is chosen. If there is no one single most numerous class, we can use the K-nearest-neighbours algorithm extension. With K-Nearest neighbours, instead of choosing the class of the nearest data point, the K (e.g. 5) nearest points are selected and the most numerous class of this set is assigned. This method has the advantage in that it is more robust to outliers (noise) in the training data as not a single data point is relied upon but a neighbourhood of K points. The value of K can be set to various different sizes; a value of 1 will make the algorithm perform equally to the basic Nearest Neighbours method described above, while K tends towards *P* then the classification of a new point will depend only on the most common classification in the training set, and thus some optimal value for K exists. This optimal value can be discovered by way of a *generalisation* method, using the set aside test data to test the performance for different values of K. The typical error-rate on handwritten numeric characters can be as low as 2-3% for a 2-class problem classifying 1s and 7s, which is a good performance for such a simple algorithm.

Neural Networks for Character Recognition

Neural networks are commonly used for digital character recognition: partly because of their popularity and appeal, but undoubtedly they provide excellent results, perhaps among the best known for character recognition. The inherent pattern recognition abilities of layered neural networks lends itself perfectly to this type of task, by autonomously learning the complex mappings in high dimensional input data [4]. There are various forms of multi-layered neural network models, the most common and applicable to this task being the standard feed-forward connectionist model usually trained by way of backpropagation (gradient-descent). (See [5] for general neural network introduction and information). In general, multi-layered neural networks are said to perform a non-linear function of the linearly weighted sum of inputs in a distributed manner and can be very powerful.

Conventionally, each pixel in the input image (or dimension in the vector representation) is represented by a neuron in the input layer, so the need for data reduction is again apparent in order to avoid an excessive network size. For each class a neuron is required in the output layer (10 in the case of numeric character classification) which indicate the predicted class of the input image. The number of hidden layers and hidden units is problem specific and is not easy to know [6], although in general only a single hidden layer is required. It can be seen that each hidden unit represents an important feature of the input image [6] and that the output layer sums these features to decide on the classification. In practice, the number of hidden units can be varied and the test dataset used to detect the optimal network structure. Training is done with the training dataset which is used to adjust the connection weights between neurons by way of standard methods, e.g. backpropagation propagates the error between the predicted and actual classifications backwards through the network making adjustments to the weights such that the error should decrease.

Multi-layered feed-forward neural networks have been shown to be very powerful not only in character recognition, but also in a wide variety of image recognition tasks such as face recognition. However, there are shortfalls which need to be overcome. If the input and hidden layers are made necessarily large in order to correctly classify the training data, the network is likely to ignore the general topology and features and instead process pixel-pixel comparisons and will thus behave poorly with noise and testing data as the network will over-fit the weights. Use of separate test data to calculate the hidden layer size and generalisation of the network is required. Furthermore, the use of dimension reduction should reduce the image data to a feature representation and thus avoid the use of direct pixel mappings for classification. Additionally, the networks by default are not robust to variance in the image such as scale, rotation and translation, necessitating image pre-processing to make all input images approximately equal in composition.

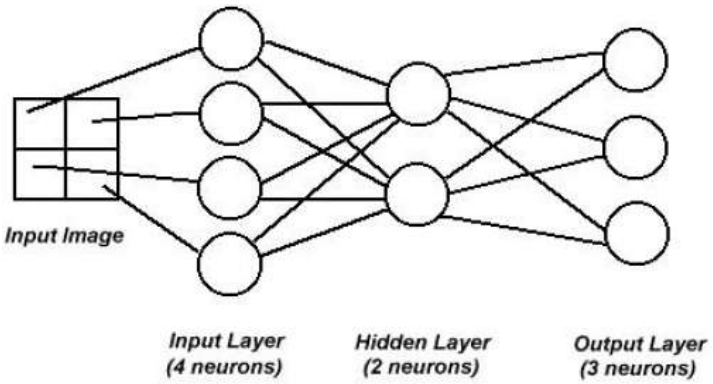

Figure 3
An example feed-forward neural network for classifying image data [4].

Overview

Discussed above are the necessary pre-recognition steps required to perform **numerical digit classification,** along with two commonly used algorithms: one a simple linear model, the other a more complex non-linear model. Both algorithms can provide good results but neither is flawless, with both susceptible to the usual problems of noise and non-generalisation so care must be taken. In industry, the best numeric character recognition systems have an error rate of less than 1 %, a performance better than human ability [3] and they are also extremely fast. The commercial importance of such system is clear. However, the field is far from complete in computer vision research, with the new area of real-time cursive handwritten character recognition, often on limited hardware, being a very active and important research area.

Bibliography

[1] Wikipedia reference, *Optical character recognition,* http://en.wikipedia.org/wiki/Optical_character_recognition , downloaded 14.01.06.

[2] David Barber (2004). Learning From Data Lecture Notes, *Dimensionality Reduction: Principal Component Analysis.* http://www.anc.ed.ac.uk/~amos/lfd/lfd0405/lectures/lfd_2004_dim_red.pdf downloaded 14.01.06.

[3] David Barber (2004). Learning From Data Lecture Notes, *Nearest Neighbour Classification.* http://www.anc.ed.ac.uk/~amos/lfd/lectures/lfd_2005_nearest_neighbour.pdf , downloaded 15.01.06.

[4] Shahzad Malik (2000). *Hand-Printed Character Recognizer using Neural Network.* http://www.cs.toronto.edu/~smalik/downloads/report_407.pdf, downloaded 15.01.06.

[5] Wikipedia reference, *Artificial neural network, http://en.wikipedia.org/wiki/Artificial_neural_networks,* downloaded 15.01.06.

[6] Rao, V., Rao, H. (1995). *Neural Networks and Fuzzy Logic*. MIS Press, New York.