

Handwritten Kanji Recognition

Hans Livingstone, Yuta Fujiwara, Nei Kato

Introduction

Demand for automated handwritten text recognition services is growing. For these systems to be valid in Japan, a robust handwritten Kanji recognition system must be implemented. My research this semester focuses on handwritten Kanji recognition methods. In this paper I will outline a general algorithm.

Preprocessing

The image recognition algorithm expects input in a precise format, therefore all image data must first be preprocessed to remove noise, cropped to a canonical 64x64 pixel size, and the image contour line must be extracted.

Feature Extraction from Training Data

In image recognition, features are defined as the distinguishing characteristics of an image. Although possible, it is typically impractical to use every pixel as a feature. Instead, Professor Kato developed a clever method called Directional Feature Element Extraction.

This method sequentially examines the Kanji with a 16x16 pixel window. The window moves in 8 pixel increments and counts the features inside it (Fig. 3). There are 4 possible directional feature values (←, |, \, and /).

In order to find the directional features, the window is further divided into 4 sub-compartments (Fig. 1). Each sub-compartment is sequentially scanned for 3x3 pixel patterns (Fig. 2). While examining a contour line image, there are 12 possible pixel pattern configurations (Fig. 4). Each pixel pattern has one or two associated directional feature values.

The window will sequentially move through a total of 49 blocks, each with 4 feature types. This creates a total of 196 unique features.

Principal Component Analysis (PCA) Algorithm Overview

The most important features of a data set typically have the highest variance. Variance measures the spread of the data (the average squared distance from the mean), it determines how far a given feature differs from the average of its type. Features that differ greatly are therefore more important. The PCA algorithm extracts these important features and ranks them according to importance.

Assuming there are N columns of training data, and M distinguishable features per image (M dimensions)

1. Create a M by N matrix \mathbf{X} with the training data in the columns.
2. Center the data around the empirical mean. Subtract the mean $\mathbf{E}[\mathbf{m}]$ of row \mathbf{m} , from every element in row \mathbf{m} of \mathbf{X} .
3. Calculate the covariance between each dimension (row)

$$cov(\mathbf{X}, i, j) = \frac{1}{M-1} \sum_{k=0}^M (\mathbf{X}[k, i] \cdot \mathbf{X}[k, j])$$

4. Create a covariance matrix and find the eigen values and vectors for it. Each eigen vector will have a respective eigen value.
5. Sort the eigen vectors by their respective eigen value from highest to lowest. The highest value represents the most important feature (the principal component).

6. Form a new "feature vector" \mathbf{V} with the sorted eigen vectors as columns.

$$\mathbf{V} = \langle eig_0, eig_1, eig_2, \dots, eig_N \rangle$$

7. Transpose the feature vector \mathbf{V} .

Fig. 1 - 16x16 segmentation window with sub-compartments

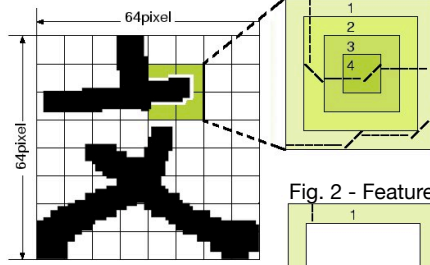


Fig. 2 - Features per sub-compartment

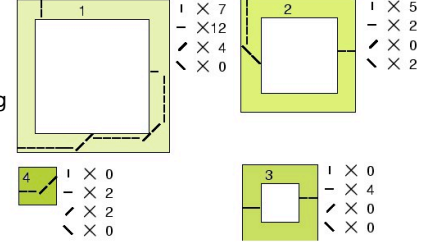


Fig. 3 - Sequential scanning

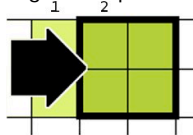
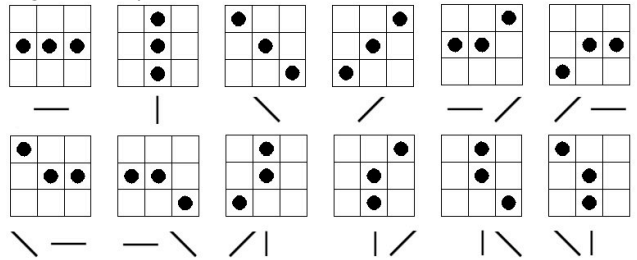


Fig. 4 - Pixel patterns and associated directional feature values



Feature Space

Eigen vectors of any system are always orthogonal and therefore form a basis for a subspace. In this case, the subspace created by the extracted eigen vectors from the training data covariance matrix is called feature space. Feature space allows for a convenient way to compare Kanji because the Euclidian distance between vectors in this space is a direct measurement of similarity. In order to make a comparison, the data must first be transformed into feature space. This is accomplished by

$$\mathbf{F} = \mathbf{V} \cdot \mathbf{X}^T$$

where \mathbf{F} is the final vector in feature space, \mathbf{V} is the feature vector, and \mathbf{X} is mean centered input data. The final step before comparison is to transform all training data into feature space.

Kanji Classification and Recognition:

The kNN (k-Nearest Neighbor) algorithm uses Euclidian distance in feature space as a measurement to classify Kanji. A given input Kanji A is assigned the class of the k nearest surrounding Kanji.

1. Select the closest k Kanji from A by Euclidian distance.
2. From the k selected Kanji, find the most often selected type.
3. Classify A as this type of Kanji.

To reduce the event of a tie in step 2, k should be an odd number. Additionally, to improve performance, the feature space should be partitioned into volumes and distances should only be checked with nearby volumes.