Digit Detection Using Adaptive Spline Models

Ansuya Ahluwalia, Eric Kim, and Nicholas Brett Marcott

University of California, Los Angeles
ansuya@cs.ucla.edu, ekim@cs.ucla.edu, bmarcott@ucla.edu

Abstract. Automated handwriting detection remains an interesting yet challenging problem in the computer vision field. Due to the curve-like nature of handwriting, it seems natural to consider approaches that directly model these curves. This project will investigate a particular approach from Hinton et. al [6] that uses an elastic matching method to recognize digits. Each digit is represented as a cubic b-spline. To classify a test image, an iterative algorithm performs an elastic match between the test image and each digit model - the digit model with the best score wins. Validation is performed against the handwritten digit dataset, MNIST [1].

Keywords: Digit Recognition, Cubic Spline, Elastic Match, Deformable Template

1 Introduction

Automated handwriting detection is an interesting yet challenging problem in the vision field. Traditional approaches to detection, such as template matching, are ineffective in this domain due to writing style variability: different people will write the same character in different ways. Due to the curve-like nature of handwriting, however, it seems natural to consider approaches that attempt to model these curves

While template matching is simple, it is very brittle. To remain robust to character variation, one must acquire a large number of digit examples, resulting in many image comparisons at test time. On the other hand, utilizing elastic matching with deformable templates increases the complexity of matching, but the number of matches decreases. In this project, we use deformable models based on cubic splines, which can capture the digit variability within a few parameters.

We investigate a particular approach from Hinton et. al [6] that uses an elastic matching method to recognize digits. Each digit class is represented by a cubic b-spline with a particular "home" configuration. To classify a test image, an iterative algorithm performs an elastic match between the test image and each digit class - the digit class with the best score wins. See Figure 1 for an illustration of

classification. To remain invariant against factors such as translation, rotation, and scaling, all matching is performed in a canonical "object frame". Validation is performed against the publically-available handwritten digit dataset, MNIST [1].

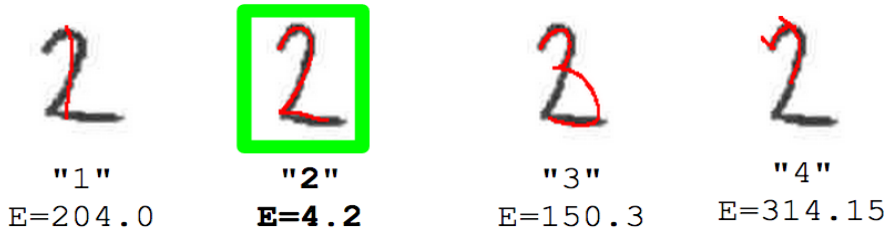

Fig. 1. An illustration of the classification approach. After fitting each digit model to the input image, the digit model with the best score (lowest energy) is the winner.

2 Related Work

Prominent amount of work in the field of handwriting recognition revolves around neural networks (recurrent, hierarchical, deep forward etc.). Neural networks learn from an image training set for character recognition. Each neural network uniquely learns features that differentiate training images. The target image is classified based on the similarity in properties with the training images. Neural networks are quick to set up but can be inaccurate if unimportant properties are learnt in context to the target data.

For discrete character recognition, neural networks are trained on weights of template neural nets that generate the character (digit/ alphabet) with highest likelihood. However, this system can only be applied to recognition of cursive and unconstrained words if templates of each word are made. This is due to the problem of character segmentation that exists in cursive and unconstrained handwriting. To handle unconstrained words, Time-Delay Neural Networks (TDNN's) can be used to generate labels for segments of data [9]. In [11], authors use a time-delay neural network to estimate a posteriori probabilities for characters in a word. A hidden markov model segments the word in a way that optimizes the global word score, using a dictionary in the process.

A lot of research has been in developing advanced algorithms for handwriting recognition using neural networks and hidden markov models. In [8], authors use backpropagation to recognize handwritten zip codes. Researchers have also developed novel, biologically motivated approaches to classify handwritten characters. Dystal (Dynamically Stable Associative Learning) is one such neural net-

work algorithm [3]. In [5], researchers use neocognitron, a neural network model for deformation-invariant visual pattern recognition, for handwritten alphanumeric character recognition. [7] incorporate HMM's into a complex stochastic language model for handwriting recognition. The pattern elements of the handwriting model are subcharacter stroke types modeled by HMMs. These HMMs are concatenated to form letter models, which are further embedded in a stochastic language model. [2] use a neural network/ HMM hybrid model for on-line handwriting recognition. They also employ EM algorithm in their approach. They perform a word-level normalization by fitting a model of the word structure using the EM algorithm.

Deformable models are efficient for characterizing handwritten digits since they have relatively few parameters and are able to capture many variations in digit instances. We investigate the system described in [6] that uses learned digit models consisting of splines whose shape is governed by a small number of control points. This method uses a single model for each digit class. As as extension to their work in [6], Revow et. al. developed mixture-models for each digit class, with little computational overhead, to better characterize the variations in the instances of each digit [10].

3 Methodology

In this section we discuss the details for performing digit recognition. As these digits are created by a series of strokes, or curves, a cubic B-spline curve is a natural representation to model the curves' "ideal" shape. As a summary of the fitting process, a model representing each digit is fit to an input image. The image is then classified as the digit model that fits best to the image.

3.1 Digit Model

In this project, cubic B-spline models are defined for only two digit classes, twos and threes. Both of these models are defined using eight control points, hence each segment of the curve is affected by only four of the control points. Along the curve, there are evenly spaced "ink generators" or "beads," which are modeled by bi-variate Gaussian distributions. These beads are used to calculate the probability of an inked pixel being generated by a particular bead. In this way inked pixels are much more likely to be generated by a neighboring bead, but are still affected by the other beads.

Similar to other inverse problems, the objective function is a balance between a regularization term and a data fit term. In this project, the regularization term is a penalty for how much the b-spline deviates from its original model position. The data fit term is calculated based on each inked pixels probability of being generated by the gaussians along the spine of the curve.

The energy function that we wish to minimize is given as:

$$E_{tot} = E_{def} + E_{fit} \tag{1}$$

E_{def} is the cost of a model shape deviating from a known starting shape, and is given as the sum of squared differences of the control point locations before and after deformation:

$$E_{def} = \sum_{CP} (\mathbf{x}_{home} - \mathbf{x}_{new})^2$$

E_{fit} is a term that consists of the "pulling" forces that each inked pixel exerts on the spline:

$$E_{fit} = -\frac{N_0}{N_B} \sum_{i \in inked}^{N_B} log(f(y_i))$$

Where N_0 is a "standard" number of pixels, and N_B is the number of inked pixels. By placing equally-spaced Gaussian "beads" along the spline, $f(y)$ is expressed as:

$$f(y) = \frac{\pi_n}{A} + \frac{1 - \pi_n}{B} \sum_{G} f_g(y)$$

Where $f_g(y)$ is the likelihood of a pixel located at position y under a bivariate Gaussian distribution g, and π_n is the relative weight a pixel has by being generated randomly from uniform noise versus by the ink generators. There is one Gaussian distribution for each "bead" along the spline - the mean is the bead location, and the variance is defined according to an annealing schedule. The equation for $f(y)$ is a mixture model that accounts for a pixel being generated either by uniform noise or by an ink generator.

3.2 EM algorithm

To minimize the energy function, an EM algorithm is used. In the E step of the algorithm, the "responsibilities" of the beads are calculated, which are then held fixed for the M step. Each Gaussian responsibility term is the likelihood that a pixel is generated by a single Gaussian bead normalized by the total probability of the pixel being generated:

$$r_g(y) = \frac{f_g(y)}{f(y)}$$

The maximization contains two steps. The first step maximizes the total energy with fixed responsibilities, finding the balance between the deformation and data fit terms. The second step updates an affine transformation that absorbs some of the unneeded deformation penalties due to natural deviations such as scaling and rotation. The difference in the control point positions used for calculating the E_{def} term are computed in the object based frame. The transformation

from the image based frame to the object based frame is calculated as mentioned in the second M step.

A fixed annealing schedule is used for the EM algorithm. At each stage of the annealing schedule, the variance of each of the ink generators is decreased, while the number of beads is increased to account for finer character detail. This fitting stops either when the total energy has not changed by a set threshold, or when all the iterations are complete. The process may converge to local maximum based on initial conditions.

Several examples of converged fits can be seen in Figures 2, 3, 4. The green crosses are the equally-spaced Gaussian beads along the spline (in red).

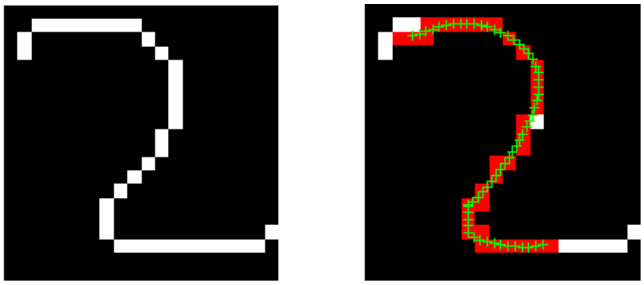

Fig. 2. Fitting a "two" model to an image.

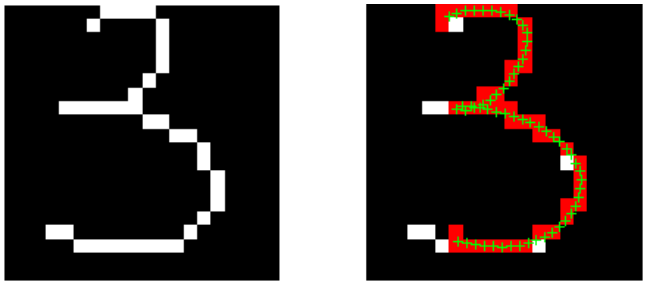

Fig. 3. Fitting a "three" model to an image.

3.3 Digit Classification

In this project, we investigated two methods of performing digit classification. Given an input image, we perform the elastic matching algorithm from Section

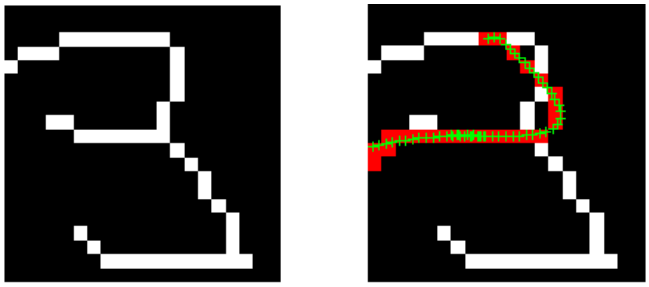

Fig. 4. An instance of a poor fit of a "three" model to the image. Here, it is likely that the matching converged to a bad local minimum.

3. In addition to the spline parameters of the fit, the algorithm also outputs the energy, or score, of the fit, according to Equation 1, reproduced below:

$$E_{tot} = E_{def} + E_{fit}$$

As E_{tot} is a measure of how well a digit model fits to the input image, one natural classification method is to simply output the label of the best-fitting model. In other words, the classification is performed as:

$$\text{label} = \underset{m \in Models}{\text{argmin}} \; E_{tot}^{(m)} \tag{2}$$

Upon examining several digit fits, we discovered that the E_{def} and E_{fit} terms seemed to be on different scales. With this in mind, we considered performing classification on the weighted sum:

$$\text{label} = \underset{m \in Models}{\text{argmin}} \; E_{def}^{(m)} + \alpha \cdot E_{fit}^{(m)} \tag{3}$$

Where α is a tunable parameter that controls the relative importance between E_{def} and E_{fit}. Note that the first classification rule (Equation 2) is a special case of (Equation 3), with $\alpha = 1.0$.

4 Results

We evaluated the digit classifier on a subset of the MNIST Handwritten Digit Database [1]. The dataset consists of labeled graylevel, pre-segmented, centered, 20x20 images. See Figure 5 for several example images.

4.1 Preprocessing

Following the authors of [6], we first apply several basic preprocessing steps to the input digit images to obtain binary, thinned images. First, the images are thresholded using Otsu's method, yielding a binary image with (hopefully)

spurious details removed. Next, an image skeletonization is performed on the thresholded image to yield a thinned digit image that captures the predominant shape details. See Figure 5 to see the results of the preprocessing operations.

4.2 Evaluation

First, we determined the optimal setting for the α parameter (see Section 3.3) by doing a grid-search on a separate validation set of 50 images ($\alpha = 0.9$). To view the best-case performance of this weighted-sum classifer, we also include the evaluation results with the α that achieves the best test performance ($\alpha^* = 9.1$). We suspect that as the validation set size increases, the discrepancy between these two cases should shrink. For the other parameters (N_0, π_N), as well as the annealing schedule, we opted to use the settings that the authors suggest in [6], as we found that they worked well in practice.

We evaluated the digit classification algorithm on 230 images from MNIST. Results are found in Table 1. One observation is that the the weighted-sum classifier with $\alpha = 0.9$ performs slightly worse than the original classifier. This may be simply due to a small validation set size - 50 images is not enough images to generalize upon. The best-case performance of the weighted-sum classifier does end up boosting overall performance by 3.4%. It's interesting to note that the "two" digits are harder than the "three" digits - there is typically a 10% difference in performance between the two classes. To further improve classification performance, it would be beneficial to examine the causes of this apparent "difficulty" discrepancy.

In our MATLAB implementation, it takes roughly 110 seconds to classify an image. There are many opportunities to optimize the code. For instance, much of the time is spent solving a least-squares problem with a general-purpose solver[1]. By analytically reducing the least-squares problem to a linear system, one would gain substantial performance gain.

4.3 Discussion

Model-based approaches such as the one presented here leads to easily interpretable results. After fitting, the deformable digit spline models provide a powerful interpretation of the input image. However, this comes at a cost - the iterative algorithm is a very expensive online process. Further, the recognition rates listed in this report are fairly lackluster, especially considering that performance on the MNIST dataset has effectively saturated at near-perfect accuracy. For instance, Ciresan et. al. reports an error rate as low as 0.23% on MNIST by utilizing a deep neural network [4].

Yet, it is worth noting the attractive qualities of model-based methods such as the one explored here. The iterative fitting process seeks to best explain the input image with respect to a set of known digit spline models. As a result, the algorithm richly describes the image in a principled manner. On the other hand,

[1] CVX: http://cvxr.com/cvx/

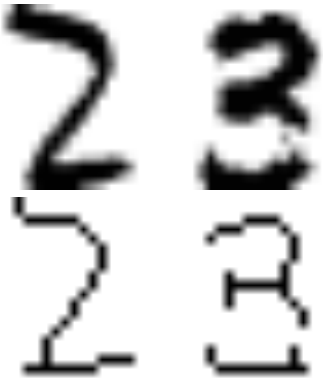

Fig. 5. Top: Example images from MNIST. Bottom: Result of applying preprocessing operations (Section 4.1).

	Digit Class	Accuracy	Correct	Incorrect
Original		59.6%	137	93
	Two	54.8%	63	52
	Three	64.4%	74	41
Weighted Sum ($\alpha = 0.9$)		58.7%	135	95
	Two	53.9%	62	53
	Three	63.5%	73	42
Weighted Sum ($\alpha^* = 9.1$)		63.0%	145	85
	Two	56.5%	65	50
	Three	69.6%	80	35

Table 1. Classification results on MNIST. "Original" corresponds to the first classifier defined by Equation 2. "Weighted Sum" corresponds to the classifier defined by Equation 3. The $\alpha = 0.9$ case corresponds to the best α found during validation, whereas $\alpha^* = 9.1$ corresponds to the best-possible α during testing.

approaches such as [4] train classifiers with the explicit goal of classification. Often, such approaches treat the problem purely as one of pattern recognition, and don't care to "explain" the images.

5 Conclusion

We have investigated a model-based approach to digit recognition that fits deformable spline models to input digit images, yielding easily interpretable results. Validation was performed on a subset of the MNIST dataset.

References

[1] The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist`. (1998).

[2] Yoshua Bengio, Yann Lecun, Craig Nohl, and Chris Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7:1289–1303, 1995.

[3] K.T. Blackwell, T.P. Vogl, S.D. Hyman, G.S. Barbour, and D.L. Alkon. A new approach to hand-written character recognition. *Pattern Recognition*, 25(6):655 – 666, 1992.

[4] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[5] K. Fukushima and N. Wake. Handwritten alphanumeric character recognition by the neocognitron. *Neural Networks, IEEE Transactions on*, 2(3):355–365, May 1991.

[6] Geoffrey E. Hinton, Christopher K. I. Williams, and Michael Revow. Adaptive elastic models for hand-printed character recognition. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 512–519. Morgan Kauffmann, 1992.

[7] Jianying Hu, M.K. Brown, and William Turin. Hmm based online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(10):1039–1045, Oct 1996.

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989.

[9] J Mantas. An overview of character recognition methodologies. *Pattern recognition*, 19(6):425–430, 1986.

[10] Michael Revow, Christopher KI Williams, and Geoffrey E Hinton. Using mixtures of deformable models to capture variations in hand printed digits. 1993.

[11] M. Schenkel, I. Guyon, and D. Henderson. On-line cursive script recognition using time-delay neural networks and hidden markov models. *Machine Vision and Applications*, 8(4):215–223, 1995.