CS229 Project Final Report
December 16, 2005

Handwritten Digit Recognition: Investigation and Improvement of the Inferred Motor Program Algorithm [*]

John A. Conley[a], My Phuong Le[b]

1 Introduction

Handwritten digit recognition is a classic problem in machine learning. It has been studied intensely by many authors for decades. The existence of standardized, publicly available MNIST training and test data [1] has led to extensive benchmarking of varied approaches to the problem. The methods of approach and their error rates are compiled at [1]. The best performance thus far [3], acheived using a convolutional neural net with cross-entropy and elastic distortions of the training set, is 0.4% error on the test set.

The wealth of research that has gone into the digit recognition problem makes it difficult to improve on the best accuracy so far acheived. It is interesting, however, to understand the workings of one existing approach to the problem and to apply different ideas from machine learning to improve on this approach.

We chose to investigate the inferred motor program method of Hinton and Nair [2]. This approach uses a generative model consisting of two sets of opposing springs whose stiffnesses are controlled by a motor program. Neural networks are then trained to infer the motor programs required to reconstruct the handwritten digits.

Our aim is to understand the approach in [2] and to identify improvements to it. To do the latter, we will implement a version of the inferred motor program algorithm in MATLAB and implement modified versions. We will compare the performance of both on the MNIST database.

2 The inferred motor program algorithm

The generative model in [2] controls a pen with two pairs of opposing springs at right angles. The stiffness of each spring is changed at 17 discrete time steps according to a motor program. With appropriate choices of these 68 stiffnesses, the pen can follow the trajectory of a handwritten digit. Given a pen trajectory, a realistic image can be created by interpolating ink between the trajectory points and convolving with a kernel which depends on two "ink parameters." The 68 stiffnesses and two ink parameters make up the 70 parameters of the generative model.

[*]e-mails: [a]conley@stanford.edu and [b]myphle@stanford.edu

With the generative model specified, the next step is to train a neural network (the "reconstruction NN") to infer the motor program parameters from the image of a digit. There will be a separate reconstruction NN for each digit class (there are in fact 12 reconstruction NNs, since performance is improved by using additional networks for dashed ones and sevens). Each reconstruction NN takes in the pixel intensities of an image as its input, and uses 400 hidden units and 70 logistic output units to return the inferred motor program parameters. Because the motor programs used to generate the MNIST training images are not known, the reconstruction NN must generate its own training set of images with known motor programs. To do this, each reconstruction NN is initialized with a set of biases and very small weights such that when given an MNIST image, it is guaranteed to output a motor program that will generate a prototypical digit image of the class. This motor program is fluctuated and from it, an image is generated. This image-motor program pair is used as input and output to train the reconstruction NN using backpropagation. This process is repeated (typically in batches) with every MNIST training image in that class.

Once trained, the reconstruction networks can be used for recognition as follows. Given a MNIST test set image, the reconstruction NN from each class can be used to infer a motor program. Each motor program can be scored using (1) the sum of squared pixel error of the reconstructed image with the test image, (2) the squared distance between the pixel error and its projection into the PCA hyperplane of that class, (3) a similar PCA distance, but for the pen trajectory, not the pixel error. The scores are input to a classifier NN with 10 softmax output units and a cross-entropy error. The authors also implement an additional local search step that improves the test set error. This step uses a neural network to emulate the generative model, so that pixel error can be backpropagated to update the motor program.

The authors also identify a simpler and more efficient way to make use of the information provided by the reconstruction NN's. By applying the appropriate reconstruction NN a motor program can be associated with each image in the training set. Then a single neural network (the recognition NN) can be trained to output both the class label and the motor program for an image. The idea is that the additional output (the motor programs) acts as an informative regularizer which improves recognition performance. Because this method, of the ones the authors implemented, gave the best results, we chose to investigate it rather than the other classificatiers.

3 Building the prototypes

To create a prototype motor program for each digit class, we began with a typical image from the class. We selected points roughly evenly along the skeleton of the digit, in the time order they would have been written. The location of these trajectory points is not enough to specify the four stiffnesses at each time step. We put in an arbitrary constraint–that at each time step, one of the vertical stiffnesses and one of the horizontal stiffnesses equals 0.5. The value 0.5 was chosen because it lies at the point of steepest slope of the sigmoid function, which is the most efficient point for backpropagation training.

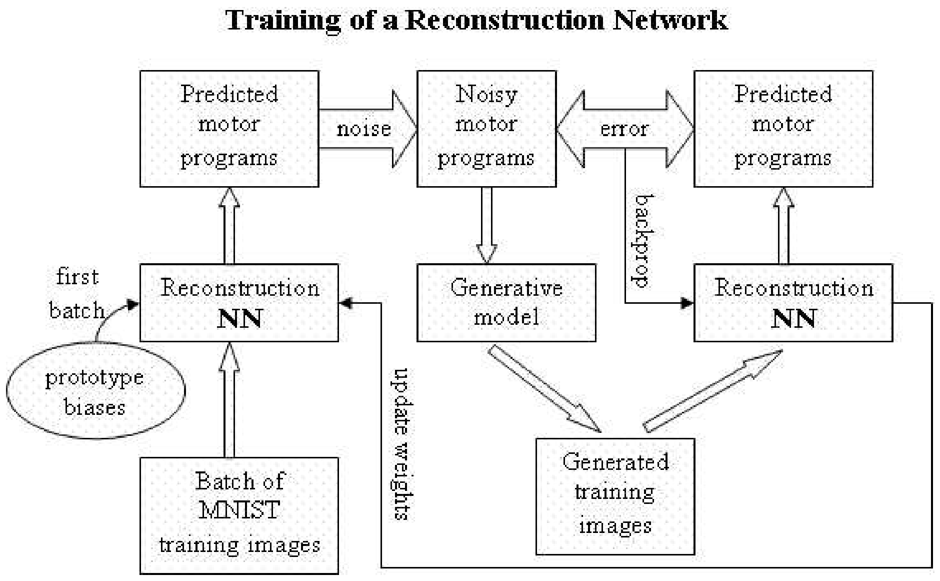

Figure 1: Training of a recognition network.

4 Training the reconstruction NNs

To generate an image from a motor program, we attempted to flesh out the pen trajectory by spacing Gaussian beads of ink along it. This did not reproduce the MNIST images as accurately as the ink kernel in [2], so we used his, and added a step to recenter the image by its "center of mass" (since the MNIST images are centered in this way).

Our reconstruction NNs have two layers with 400 hidden units. We trained them as described above, using batches of 300 MNIST training images and running backprop for 50 epochs before updating the weights and using the next batch. We used momentum of 0.95 and a search-then-converge learning rate annealing schedule. For each digit, we ran through the whole training set 200 times. The training is detailed in Fig. 1.

Some MNIST images and their reconstructions are shown in Fig. 2. The majority of the images are well-reconstructed. The dashed one and seven that are shown in the figure are poorly reconstructed, however. The solution to this is to train a separate network for dashed ones and sevens, as discussed in [2]. Getting extremely accurate reconstructions, however, is not crucial for training the recognition NN. This is because the reconstructed motor programs are used for training, but not the reconstructed images. Thus we chose not to implement separate reconstruction NNs for dashed ones and sevens.

5 The recognition NN: modifications and results

Hinton *et. al.* use a 3-layer recognition NN with 500 hidden units in the first layer and 1000 in the second. Training this using the motor programs as additional output, as described above, they acheive a 1.34% error rate on the test set. For time and memory considerations,

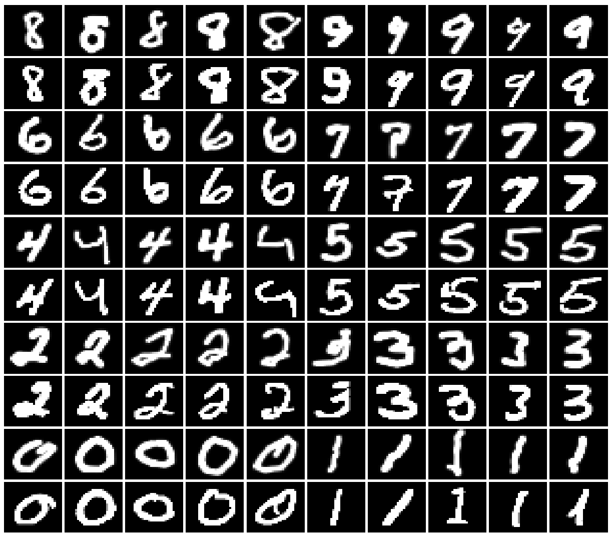

Figure 2: Reconstruction of MNIST images. For each digit class, the lower row contains the original MNIST images, and the upper row contains the reconstructions of these images by the reconstruction NNs.

we chose a simpler architecture, a 2-layer NN with 400 hidden units, and trained it the same way using the four-spring stiffnesses motor programs as additional outputs. We will refer to this as the four-spring recognition NN. Because of the relative simplicity of the architecture, we cannot directly compare our results to those acheived in [2]. Instead we use this architecture consistently in order to compare the effects of the modifications described below.

The inferred motor program algorithm is motivated by the idea that a realistic generative model should provide the most natural way to classify patterns. The four-spring model is physically plausible but there is degneracy in its parameters. There is a one-to-one mapping between sets of four spring stiffnesses and pairs of x, y accelerations. Thus we can replace the 68 spring stiffnesses with 32 accelerations and two initial displacements. This replacement can be implemented either in the reconstruction NNs, the recognition NN, or both. Due to time constraints we chose to only implement and evaluate this modification for the recognition NN. We will refer to this as the two-acceleration recognition NN. This reduction of parameters has two main benefits. First, for a given number of hidden units, the number of weights is reduced, and thus training should be faster. We found that the training time for the two-acceleration NN was 25% less than that of the four-spring NN. Second, since the output space is smaller, for a given number of hidden units the network should be able to model the variation better. This claim is supported by our results, below.

To evaluate the benefit of the additional outputs (either spring stiffnesses or accelerations) we implemented, for comparison, a recognition NN of the same architecture that we trained using only the class labels. The results of all three cases are shown in Table 5.

It is surprising that the labels-only NN outperforms the NNs with additional outputs. One possibility is that each reconstruction NN still does not model the MNIST digits of

Case	Training set error	Test set error
Labels only	0%	1.94%
Four-spring	0.016%	2.86%
Two-acceleration	0.008%	2.13%

Table 1: Results: the same learning rate and number of iterations were used in each case.

its class very well. Thus the motor programs associated with the training images may be confusing the recognition NN rather than informing it. The fact that we have not implemented separate reconstruction NNs for dashed sevens and ones could be contributing to this problem. Another possibility is suggested by the following observation. The cross-entropy error during training was lowest for the labels-only case, higher for the two-acceleration case, and still higher for the four-spring case. This implies that the learning rate and number of backprop iterations we used to train in all three cases was more optimal for the labels-only case. By using a validation set to optimize training in each case, a fairer comparison could be obtained.

6 Conclusions

We have demonstrated the feasibility of reducing the complexity of the generative model used in [2]. Much more work, however, needs to be done to validate the claim that this can lead to an improvement in classification. The main idea of Hinton *et. al.*, that a a generative model can be used both to generate more training data and to associate useful extra information with training examples to improve classification, can be applied to other problems in pattern recognition.

Acknowledgments

We would like to thank Geoffrey Hinton and Vinod Nair for useful discussions.

References

[1] Y. LeCunn, C. Cortes, *The MNIST database of handwritten digits*, http://yann.lecun.com/exdb/mnist/.

[2] G. Hinton, V. Nair, *Inferring motor programs from images of handwritten digits*, to be published in NIPS 2005, available at http://www.cs.toronto.edu/ hinton/absps/vnips.pdf.

[3] P. Y. Simard, D. Steinkraus, J. Platt, *Best Practice for Convolutional Neural Networks Applied to Visual Document Analysis*, International Conference on Document Analysis and Recogntion (ICDAR), IEEE Computer Society, Los Alamitos, pp. 958-962, 2003.