A Three-Unit Network is All You Need to Discover Females

Neil Alldrin
Department of Computer Science
University of California, San Diego
La Jolla, CA 92037
nalldrin@cs.ucsd.edu

Andrew Smith
Department of Computer Science
University of California, San Diego
La Jolla, CA 92037
atsmith@cs.ucsd.edu

Doug Turnbull
Department of Computer Science
University of California, San Diego
La Jolla, CA 92037
dturnbul@cs.ucsd.edu

Abstract

We train neural networks to classify images according to facial expression, gender, and identity. We explore two dimension reduction techniques, PCA and downsampling. We compare two activation functions, the logistic sigmoid and "funny"-tanh and experiment with networks with different numbers of hidden units.

1 Introduction

In this paper we explain modifications to the traditional back-propagation algorithm, and use this modified algorithm to classify images of human faces. Each face in our data set is classified by gender, identity, and facial expression. Our goal is to train neural networks to learn these classifications.

Our approach is divided into three stages: pre-processing images, training the network, and validating the networks ablitiy to generalize. During the pre-processing stage we either downsample the images or use principal components analysis (PCA) to reduce the dimensionality of the data. In the training stage, a neural network is trained using error back-propagation. In the validation stage, the network is tested with novel data.

We divide the images into three sets, a training set, a hold-out set, and a test set. The training set is the data that the back-propagation algorithm uses to adjust weights. While these weights are being adjusted, the hold-out set is used to find the point at which the network stops learning general trends and starts memorizing the training set (overfitting). Once back-propagation trains the network, it attempts to classify the test set to evaluate the performance on novel data.

All of our neural networks are multi-layer perceptron networks with one hidden layer. For classification tasks, we use the same number of outputs as classes. We train networks with

varying numbers of hidden units to find a network architecture capable of good classification. For each of these network architectures we partition our data for cross validation. Cross validation involves running back-propagation multiple times with different test sets.

2 Heuristic Improvements to Back-Propagation

Back-propagation is a computationally efficient way to learn the weights of a feed-forward neural network.

2.1 Our Back-Propagation Algorithm

Definition of variables:

- I_k is the kth input of the network.
- O_i is the ith output of the network.
- T_i is the target value for the ith output.
- $g(x)$ is the activation function (we consider $g(x) = 1.7159 \tanh(\frac{2}{3}x)$ and $g(x) = 1/(1 + e^{-x})$).
- $W_{j,i}$ denotes the weight from hidden unit j to a output unit i, similarly $W_{k,j}$ is a weight from input unit k to a hidden unit j.
- ΔW is the change in weight.
- in_i and in_j are the weighted sums of the inputs to output unit i and hidden unit j respectively.
- a_j is the activation of hidden unit j (ie, $a_j = g(in_j)$).
- α is the "momentum".
- η is the learning rate.
- δ_i is the error for output unit i. δ_j is the error for hidden unit j.

Our back-propagation algorithm works as follows:

1. Normalize the inputs.
2. Randomly initialize weights.
3. Initialize $\Delta W_{j,i} = \Delta W_{k,j} = 0$.
4. For $n = 1$ to some number of Epochs,
   (a) $\eta = C/(K + n)$, where C,K are constants.
   (b) For each input in the training set (in random order),
      i. $\forall i : \delta_i = (T_i - O_i)g'(in_i)$.
      ii. $\forall i, j : \Delta W_{j,i} = \alpha \cdot \Delta W_{j,i} + \eta \cdot a_j \cdot \delta_i$.
      iii. $\forall j : \delta_j = g'(in_j) \sum_i W_{j,i}\delta_i$.
      iv. $\forall j, k : \Delta W_{k,j} = \alpha \cdot \Delta W_{k,j} + \eta \cdot I_k \cdot \delta_j$.
      v. $\forall i, j : W_{j,i} = W_{j,i} + \Delta W_{j,i}$.
      vi. $\forall j, k : W_{k,j} = W_{k,j} + \Delta W_{k,j}$.
   (c) If the current set of weights is the best found so far on the hold-out set, save this set of weights as $W1_{best}$ and $W2_{best}$.
5. Return $W1_{best}$ and $W2_{best}$.

2.2 Notes on Our Back-Propagation Algorithm

We use a number of techniques presented in [1]:

- All inputs are normalized prior to learning (step 1). Specifically, each dimension of the input is normalized over all patterns to have mean 0 and standard deviation 1.
- Weights are initialized randomly with mean 0 and standard deviation $1/m^{1/2}$, where m is the fan-in of a unit.
- Every epoch, the training set is randomly shuffled.
- Momentum is used to dampen oscillations in the weight updates.
- When using the standard sigmoid function we set targets to .211 and .789 and when using the "funny"-tanh function ($g(x) = 1.7159 \tanh(\frac{2}{3}x)$) we initialize targets to be -1 and 1. These targets are the values at which the curvature of the activation function is maximized.

In addition to the above techniques we do the following:

- We choose constants C and K to control the learning rate η. K controls how quickly η decreases and C increases η by a constant factor.
- To test our algorithm, we use a forward difference to approximate the gradient of the error with respect to the weights and compare it to the change in weights obtained by back-propagation. This ensures that the weights move down the error surface.

2.3 Comparison of Two Activation Functions : Tanh vs. Logistic Sigmoid

After finding a suitable learning rate, using the "funny"-tanh activation function yields faster convergence, shown in figure 1. In general, we observed this trend for many different data sets. For this reason, we use "funny"-tanh as the activation function for all of our experiments in section 3.

3 Classifying Human Faces : Identity, Gender, Facial Expressions

Each graph shows how well the data is classified by networks with different numbers of hidden units. For each number of hidden units, we ran our learning algorithm a number of times using cross-validation. Each figure shows the average percentage of data correctly classified by the networks after their weights have been learned, indicated by the solid line and squares. The standard deviation is indicated by dotted lines above and below the mean. Each figure shows the statistics for the training, hold-out, and test sets.

3.1 Identity

We divide our 110 images into 10 sets of 11 images, so that each set ideally contains one image of each person. In reality, each person is not equally represented in our data, so some of these 10 sets lack an image of a particular person. However, this is sufficient for creating the training, hold-out, and test sets such that each person is always adequately represented in the training set. We pick 2 sets to be our hold-out and test sets, and use the remaining 8 as the training set.

Figure 2 shows the success of networks with different sized hidden layers classifying images by identity. Complete success would mean that the network could be given an image

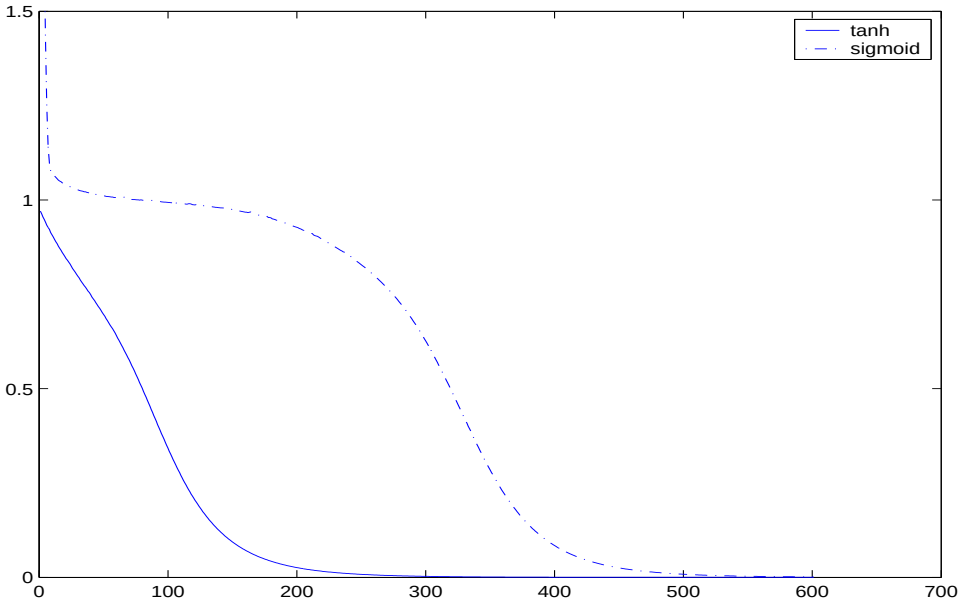

Figure 1: Comparison of the convergence of the standard logistic sigmoid and $g(x) = 1.7159 \tanh(\frac{2}{3}x)$ as activation functions. In this example the XOR function was being learned by a network with two hidden units.

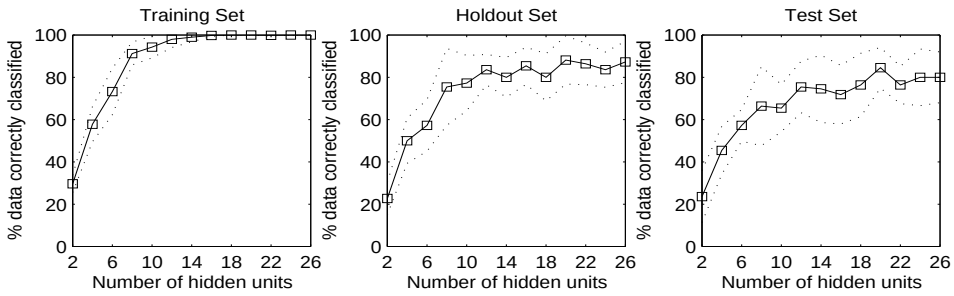

Figure 2: Average success classifying by identity with downsampling to 20x24 pixels.

of a person's face and output their identity for any picture of a person in the training set. Images are downsampled from 240x292 to 20x24 pixels. This is justifiable because a human can still classify the images by identity at this resolution. Each hidden unit has weighted connections from all pixels (and a bias). Sixteen hidden units are enough to learn the training set perfectly every time. Also, classifying novel data does not seem to improve when more than sixteen hidden units are used. We conjecture that each hidden unit recognizes a particular person, since the images in our data set are of fourteen different people. The network correctly identified about 80 percent of the test set (novel data), much better than guessing randomly, which would be correct once every fourteen inputs.

Figure 3 shows the same test with images downsampled to 40x48 pixels. This is not significantly better or worse than images downsampled to 20x24 pixels. However, this figure took nine hours to generate compared to an hour and a half.

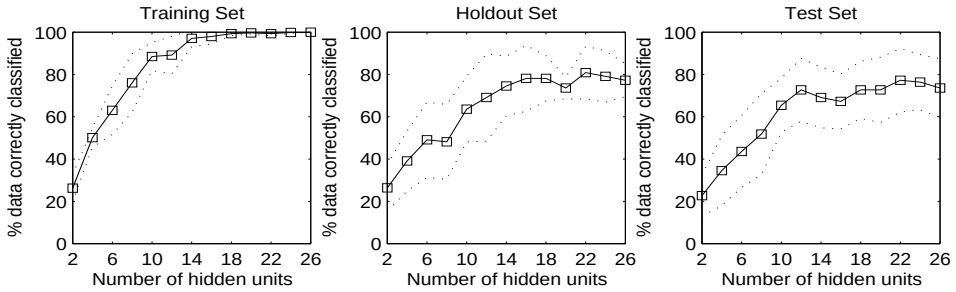

Figure 3: Average success classifying by identity with downsampling to 40x48.

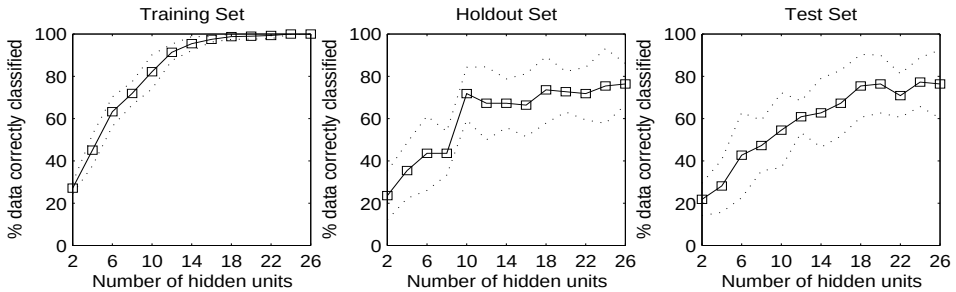

Figure 4: Average success classifying by identity with PCA.

Figure 4 shows the same test using PCA to reduce the dimension of the original images. We project the data onto the 37 directions of maximum variation (about 85 percent of the total variance). This technique required 22 hidden units to perfectly classify the training data and 20 hidden units maximized the success of classifying novel data.

Interestingly, the PCA-reduced data requires more hidden units to learn to classify by identity than raw pixel data of downsampled images. The number of input units for PCA is 37 as opposed to 480 for images downsampled to 20x24 and 1920 for the images downsampled to 40x48. Thus using PCA greatly reduces the cpu-time required; however, as can be seen in figures 2 and 4, downsampling produces better classification on novel data.

3.2 Gender

We divide the data into fourteen sets, $m_1 ... m_7$ and $f_1 ... f_7$, one per person. The hold-out and the test sets each consist of one male set and one female set, $\{m_i, f_j\}$ and the training set consists of the remaining ten sets. This ensures the training set always contains an equal number of males and females and that the hold-out and test sets only contain images of people not in the training set (to prevent learning by identity).

Figure 5 shows the success of networks with different sized hidden layers classifying images by gender. The training set can be completely classified by a network with two hidden units. Networks with more than two hidden units did not significantly improve classification of novel data. The networks correctly classified an average of 80 percent of novel data, better than a 50-50 guess. We suspect the classification was not perfect because there were not enough examples of males and females to train the networks.

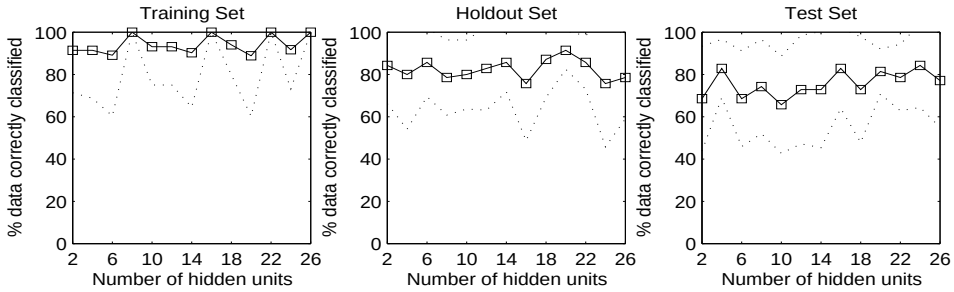

Figure 5: Average success classifying by gender with downsampling.

3.3 Facial Expressions

As when classifying by gender, we divide the data into fourteen sets, one per person. The hold-out and test sets each consist of a set of images of a particular person expressing the 6 different emotions. The training set is the rest of the data. This guarantees that validation is with images of a person not included in the training or hold-out sets.

Two approaches to classification by facial expression are to use raw images and to use "difference" images. Difference images for a particular person are created by subtracting the image of that person expresssing a neutral face from each image of that person expressing an emotion.

3.3.1 Original Images

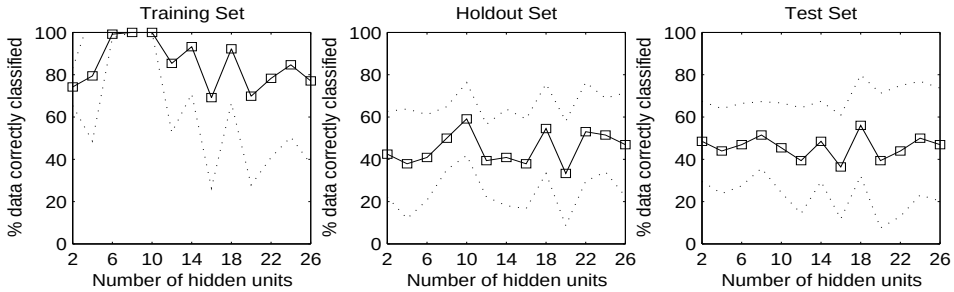

Figure 6: Average success classifying by facial expression with downsampling.

Most neural networks in figure 6 had difficulty learning the training data and were able to correctly classify about 45 percent of novel data (compared with a probability of 1/6 guessing randomly), but with a large standard deviation. We suspect this large amount of variance corresponds to sensitivity to who is in the test and hold-out sets. This would be due to different "styles" of facial expressions (ie, teeth showing in a smile for some people but not others). Another possible reason could be the intrinsic difficulty of learning facial expressions.

3.3.2 Difference Images

Every network with more than eight hidden units learned the training data to within 95 percent accuracy. This is an improvement over using raw images. Generally, using dif-

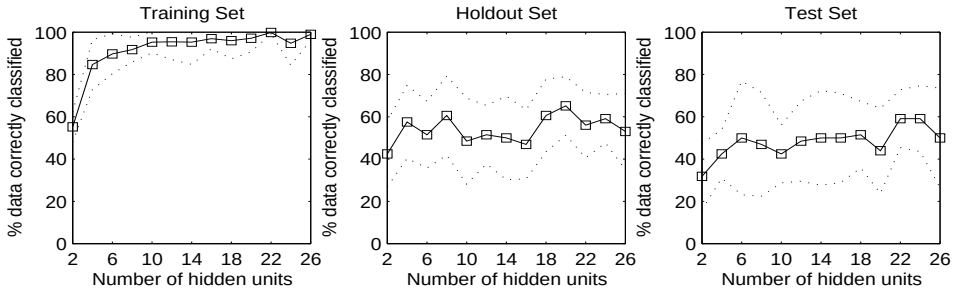

Figure 7: Average success classifying by facial expression using difference images with downsampling.

ference images made no difference in the success rate of classifying novel data. However, using difference images lowered the variance of the success rate. This could indicate less sensitivity to facial characteristics unrelated to emotion (ie, long blonde hair).

4 Conclusion

Using "funny"-tanh as an activation function instead of the standard logistic sigmoid helps weights converge in fewer epochs. This confirms the results in [1]. We also found it difficult to chose a learning rate that decreased fast enough to avoid oscillations, but not before significantly reducing the error. Quickly converging weights require learning rates specifically tuned to the input data and the type of classification. Downsampling the images results in more input units to the network than PCA and therefore requires more computation. However, it seems to find better networks than using PCA for dimension reduction.

Classifying identity was much more successful than classifying by gender or expression. This confirms our intuition that the networks recognize distinctive features of each person, for example, hair color or skin texture. Classifying by facial expression was more successful than classifying by gender. We suspect this was because features used to distinguish gender are much more subtle than those used to dinstinguish facial expressions.

Individual Contributions

Neil Alldrin:
I helped write the back-propagaion and some supporting functions and scripts, helped run tests on the data, made the graphs all pretty, helped write the report, drank a lot of coffee, spent an inordinate amount of time in my office, etc.

Andrew Smith:
I helped write backprop.
I generated the sigmoid vs. funny-tanh plot.
I tweaked parameters (eta & alpha) for backprop for about 30 houurs.
I wrote the functions to load and save weights.
I generated some of the data for the graphs.
I provided lots of good MP3's for my partners' enjoyment.

Doug Turnbull:
I was responsible for implementing many of the image testing matlab modules including precomputation with PCA and image reduction, cross validation, and file input and output.

Much of my time was spent running test after test for various parameters and classifications.

References

[1] LeCun, Yann, Bottou, Leon, et. al. (1998) *Efficient BackProp* Neural Networks: Tricks of the trade

[2] Russel, Stuart & Norvig, Peter (1995) *Artificial Intelligence : A Modern Approach, First Edition* New Jersey: Prentice-Hall Inc.