Classifying Facial Expression with Radial Basis Function Networks, using Gradient Descent and K-means

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

This paper compares methods of training radial basis function networks. We found RBF networks initialized by supervised clustering perform better than networks initialized by unsupervised clustering and improved with gradient descent. Gradient descent did not significantly improve the networks initialized by supervised clustering.

1 Introduction

Radial basis function (RBF) networks are commonly used for pattern classification. In this paper, we explore three techniques for training RBF networks. First, we present the structure of standard RBF networks and describe three techniques for learning their parameters. We then evaluate these training techniques on a set of human facial images.

2 RBF Networks

Radial basis function networks typically have two distinct layers as shown in figure 1. The bottom layer consists of a set of basis functions each of which is a function of the distance between an input pattern and a prototype pattern. A standard choice of basis function is the Gaussian:

$$\phi_j(\boldsymbol{x}) = exp\{-\frac{||\boldsymbol{x} - \boldsymbol{\mu}_j||^2}{2\sigma_j^2}\} \tag{1}$$

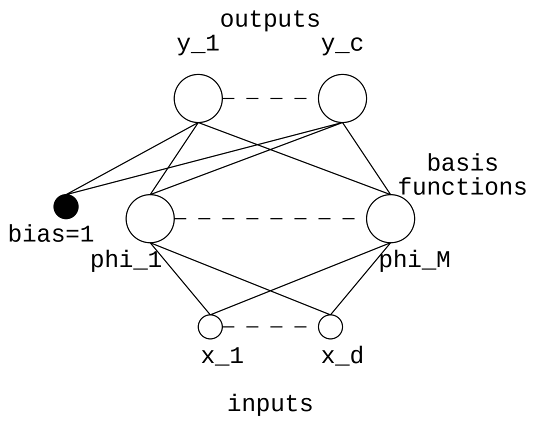

Figure 1: A generic RBF network.[1]

The top layer is a simple linear discriminant that outputs a weighted sum of the basis functions. The equation for a single output y_k is:

$$y_k(\boldsymbol{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\boldsymbol{x}) + w_{k,bias} \tag{2}$$

For our networks, we set the weights of the top layer using the least squares error. Rewriting the formula for y_k in matrix notation, equation 2 becomes $y(\boldsymbol{x}) = \boldsymbol{W}\Phi$. The minimum of the sum squared error $E = \frac{1}{2}\sum_n \sum_k \{y_k(\boldsymbol{x}^n) - t_k^n\}^2 = \Phi^T\Phi\boldsymbol{W}^T - \Phi^T\boldsymbol{T}$ is $\boldsymbol{W}^T = \Phi^\dagger\boldsymbol{T}$ where $\Phi^\dagger$ is the pseudo-inverse of Φ, $\Phi^\dagger = (\Phi^T\Phi)^{-1}\Phi^T$.

2.1 Initializing Parameters for Radial Basis Functions

Shperical Gaussian basis functions, as in equation 1, each have two parameters, $\boldsymbol{\mu}_j$ and σ_j, for the jth basis function ϕ_j. The three techniques we explore for finding these parameters are maximum likelihood (ML), K-Means, and gradient descent.

Maximum Likelihood:
Let $\{x_1, ...x_{N_j}\}$ be the set of input patterns in class j and let the number of basis functions, assuming one basis function per class. Using maximum likelihood to set the parameters of basis function ϕ_j to:

$$\boldsymbol{\mu}_j = \frac{1}{N'}\sum_{n=1}^{N'} \boldsymbol{x}^n \tag{3}$$

$$\sigma_j = \sqrt{\frac{1}{N'}\sum_{n=1}^{N'} (\boldsymbol{x}^n - \boldsymbol{\mu}_j)^T(\boldsymbol{x}^n - \boldsymbol{\mu}_j)} \tag{4}$$

K-means:
K-means is an unsupervised clustering algorithm that finds a set of K centers that iteratively

[1]Figure adopted from [2]

minimizes the mean squared distance from each data point to its closest center[4]. Each center represents a cluster, or subset of nearby data points. The average value for all the points in a cluster are used to find a $\boldsymbol{\mu}$ for the basis function corresponding to that cluster. σ is set to the standard deviation of the points in the cluster, as in equation 4. Using K-Means to find basis functions is less restrictive than using ML because the number of basis functions is not dependent on the number of classes.

2.2 Improving parameters using Gradient Descent

A common technique for reducing the error of machine learning models is gradient descent. The basic idea of gradient descent is to reduce the error by iteratively adjusting the parameters of the learning model. The gradient of the error function points in the direction of maximum increase, so the error of the learning model can be reduced by moving the parameters in the opposite direction of the gradient. Typically this is accomplished through the iterative process of repeatedly calculating the gradient for a particular input, or set of inputs, and adjusting the parameters by a small amount.

The parameters of our model are the weights of the linear discriminant, w_{kj}, and the means and standard deviations of the radial basis functions, $\boldsymbol{\mu}_j$ and σ_j. Since the optimal set of weights for a given set of radial basis functions can be calculated in one step, there is no need to use an iterative process to update the weights. However, no such closed form exists for the optimal set of Gaussians, so we must resort to iterative processes.

Our training procedure uses stochastic gradient descent which trains the RBF network for a number of epochs. Each epoch trains the RBF network on every data point in our training set, one at a time, in random order.

Our error function is the common sum-of-squares error, $E(x^n) = \frac{1}{2}\sum_{k=1}^{c}(y_k(x^n) - t_k^n)^2$. The derivative of this error with respect to the standard deviation of basis function j, σ_j, is

$$\frac{\partial E}{\partial \sigma_j}(x^n) = \sum_k \{y_k(\boldsymbol{x}^n) - t_k^n\} w_{kj} exp\{-\frac{||\boldsymbol{x}^n - \boldsymbol{\mu}^j||^2}{2\sigma_j^2}\} \frac{||\boldsymbol{x}^n - \boldsymbol{\mu}^j||^2}{\sigma_j^3} \tag{5}$$

The derivative of the error with respect to the i'th component of the j'th mean, μ_{ji}, is

$$\frac{\partial E}{\partial \mu_{ji}}(x^n) = \sum_k \{y_k(\boldsymbol{x}^n) - t_k^n\} w_{kj} exp\{-\frac{||\boldsymbol{x}^n - \boldsymbol{\mu}_j||^2}{2\sigma_j^2}\} \frac{(x_i^n - \mu_{ji})}{\sigma_j^2} \tag{6}$$

For each basis function, the term, $\sum_{k=1}^{c}(y_k(x^n) - t_k^n) \cdot w_{jk}$ can be computed with the backpropagation algorithm.

Once we have calculated the gradient of the error, we update the means and standard deviations by moving them by a small amount against the gradient: $\mu_{ji} \leftarrow \mu_{ji} - \eta \frac{\partial E}{\partial \mu_{ji}}$ where η is a small, decreasing value called the learning rate. After the basis functions are updated, a new set of optimal weights is calculated.

3 Human Expression Classification

3.1 Experimental Methodology

The goal of our experiments is to train RBF networks that can classify images of human faces as one of the following six expressions: happy, sad, surprized, afraid, disgusted, angry. We use three methods for learning the parameters of the network. The first method uses supervised learning (ML) of the gaussian parameters, μ_j and σ_j, combined with the least squared error solution for the top layer weights. The second method uses unsupervised clustering (K-Means) to learn μ_j and σ_j, again using least squares to learn the weights. The third method uses gradient descent on μ_j and σ_j, updating the weights with least squares after every training example. For the techniques that use K-Means, the number of Gaussians are varied from 1 to 53. For the supervised learning techniques, six Gaussians are used, one for each expression.

For all experiments, we use "difference" images instead of the original images of facial expressions. For each image of a face, we define the corresponding "difference" image as the difference between that image and the average of all images of that individual. Also, we use principal component analysis to reduce the dimensionality of the data to 20. The original images are downsampled to fit our computational resources. To evaluate the success our our networks, we use cross-validation which reserves a small amount of the training set to test the network's ability to generalize to novel data.

3.2 Results

Supervised and Unsupervised Clustering:

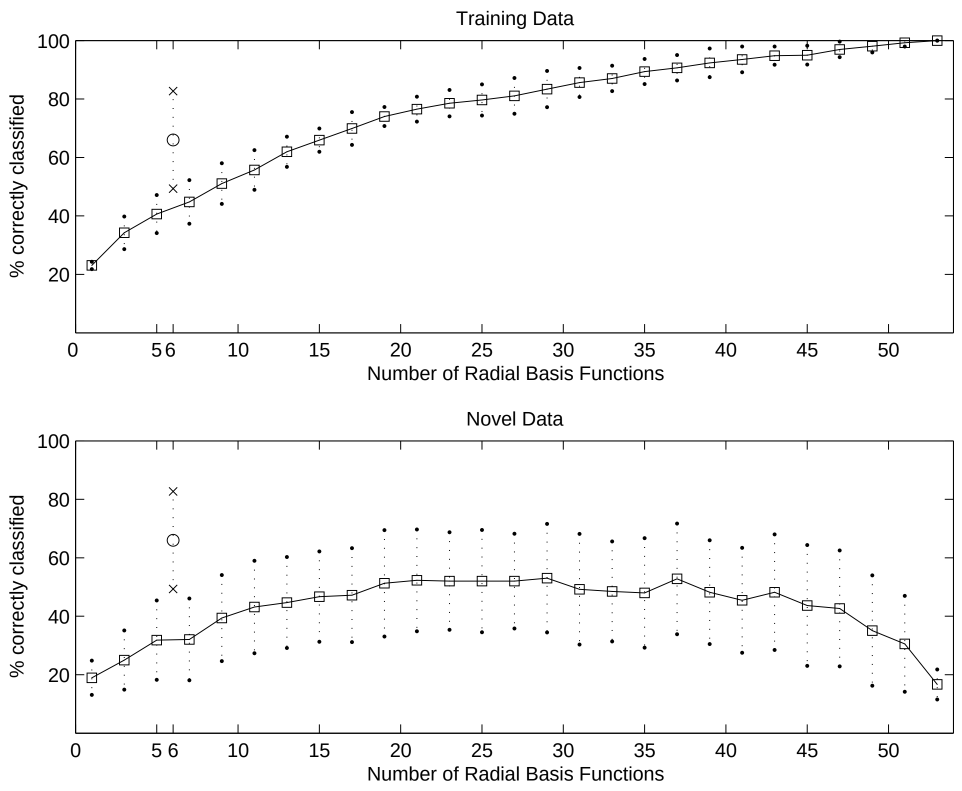

Figure 2: Performance of supervised (ML) and unsupervised (K-Means) clustering. The circle represents the average supervised result and the squares represent the average unsupervised results with varying numbers of clusters.

Figure 2 shows that supervised clustering generalizes better than unsupervised clustering with any number of clusters. Unsupervised clustering did best in the range of 20 to 30 clusters, achieving roughly 50% correct classification, but with a high standard deviation. The sharp falloff in generalization after 45 clusters combined with the near perfect classifcation of the training data indicates memorization.

Gradient Descent:

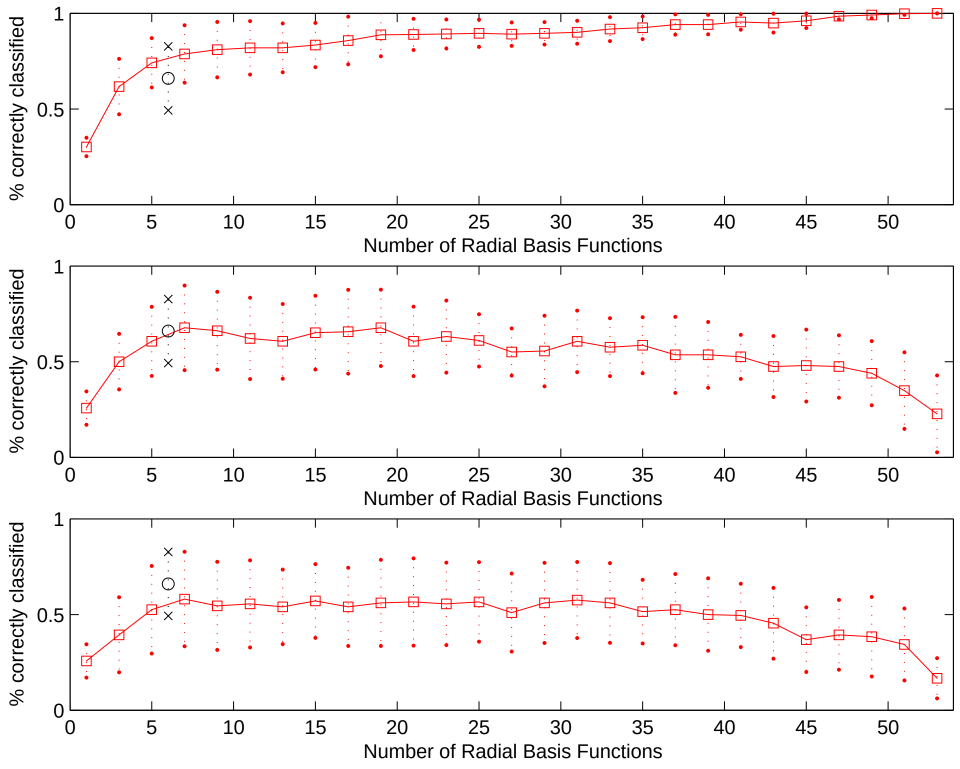

Figure 3: Gradient descent initialized with K-Means. In order from top to bottom are the training, hold out, and test sets.

Figure 3 shows that a network with RBFs initialized with K-Means and trained with gradient descent generalizes best with around six clusters as opposed to 25 clusters without gradient descent. Using six radial basis functions instead of twenty-five is intuitively better because a learning model with fewer parameters implies generalization rather than memorization. The fact that we are classifying six facial expressions also seems to indicate good generalization. Using gradient descent to improve a network initialized with supervised clustering had little effect.

Figure 4a illustrates the use of a hold out set to reduce memorization. The network finds an optimal set of weights for the hold out set quickly, but continues to memorize the training set. 4b shows that gradient descent significantly moves the centers of RBFs initialized with K-Means.

Comparison:

Figure 5 shows the success rate for both K-Means initialized RBF networks, and K-Means initialized and gradient descent improved RBF networks, printed side-by-side for comparison. Clearly, gradient descent improves the RBFs placed with K-Means.

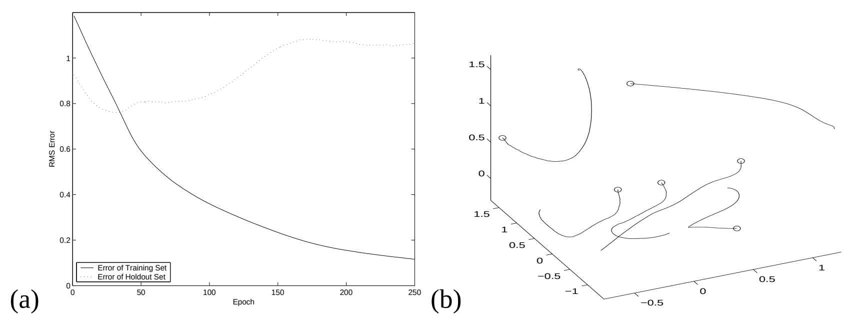

Figure 4: (a) Mean squared error of training (solid) and holdout sets (dotted) during gradient descent training. (b) The centers of six RBFs being moved by gradient descent, projected onto three random dimensions.

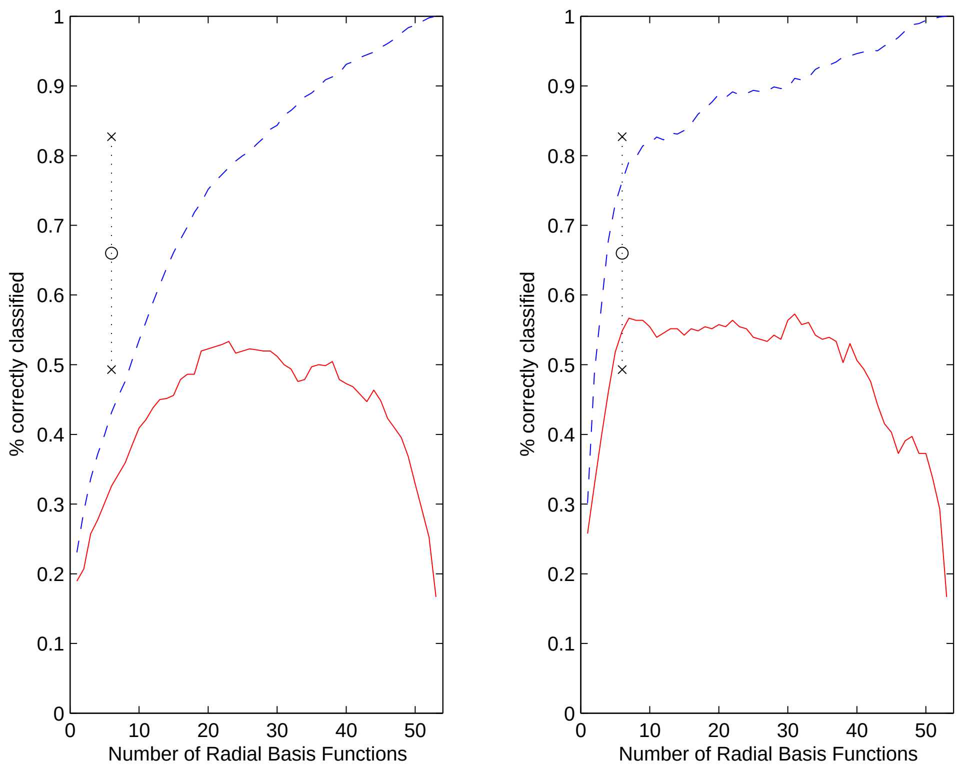

Figure 5: K-Means initialized RBFs versus K-Means initialized and gradient descent improved RBFs.

4 Conclusion

The network that generalized best was found using supervised clustering with all principal components, as summarized in table 1. For all of our tests in the results section we used the top 20 principal components, representing 80 percent of the total variance in the data set. Using all components increased the success of RBF networks using supervised clustering,

Table 1:

	20 Principal Components		All (52) Principal Components	
	mean	std	mean	std
training data	0.926	0.020	1.00	0.000
novel data	0.659	0.167	0.742	0.142

but did not greatly improve the success of RBF networks initialized with K-means.

Gradient descent greatly improves generalization of RBF networks initialized with unsupervised clustering. However, as with all iterative error reduction techniques it is more computationaly intensive.

Using gradient descent to improve the RBFs found with supervised clustering did not significantly reduce error. However, it is not always possible to use supervised clustering because it might be the case that only a small portion of the entire data set has known classes. In this case, one would use unsupervised clustering on the entire data set to set the parameters of the radial basis functions, and use least-squares to find the weights that minimize the error on the classified data.

Individual Contributions

Neil Alldrin:
I helped write the ML classifier, helped a little with the back-prop code, helped with testing, helped write the writeup, and stuff. I had fun and stuff.

Andrew Smith:
I wrote the gradient descent procedure. I wrote one version of a finite-difference error gradient approximator, used for code checking purposes. I ran the tests to generate the figures in this paper. I wrote k-means. I wrote the least-squares solver (it was easy in matlab) This project was a bit too rushed for me to have more than a marginal amount of fun.

Doug Turnbull:
I was wrote much of the file loading, preprocessing, and testing code in addition to implementing the maximum likelihood module. I was also involved in running tests and writing this report. I transcended fun.

References

[1] Alldrin, N., Smith, A., Turnbull, D. (2003) A Three-Unit Network is All You Need to Discover Females, *CSE253, UCSD*.

[2] Bishop, C.M. (1995) Neural Networks for Pattern Recognition, *Oxford University Press, New York*.

[3] Hastie, T., Tibshirani, R., Friedman, J. (2001) Elements of Statistical Learning, *Springer-Verlag, New York*.

[4] Kanungo, T, et. al. (2000) The Analysis of a Simple k-Means Clustering Algorithm, *Computational Geometry 2000, Hong Kong*.