CS B553 Homework 4: Metaheuristic Optimization and Stochastic Descent
Due date: 2/28/2012

In this assignment you will be asked to implement and compare algorithms for fitting a neural network (NN) model to images. Note that it is not particularly important that the data comes from an image and not some other source – in fact our models are not particularly useful representations of pictures. But, images provide us with a convenient way to understand how the models and the optimization work.

[Note: make sure to start this assignment early! Each optimization may run for hours depending on the speed of your implementation, but should hopefully not exceed more than 1-2 hours each. Be sure to tune your optimizers on small problems before moving to larger ones.]

Find a picture of a friend, family member, or your favorite celebrity, turn it into grayscale, and sample N points from the image, either at random or with a grid. Let the pixel coordinates (i,j) be mapped into a range \((u,v) \in [0,1]^2\) and map the grayscale intensity to the range \([0,1]\). This will be your training data.

We will be look at modeling this data with the class of radial basis function (RBF) networks. These networks consist of \(H\) hidden units, where \(H\) is a parameter, and each unit uses the Gaussian activation function

\[
f_h(x; \beta_h, c_h) = \exp(-\beta_h \|x - c_h\|^2)
\]

where \(\beta_h\) and \(c_h\) are the unit’s decay and center parameters. Here \(c_h\) is a 2D vector. Note that this activation function is a radially symmetric “bump” around the peak at \(x=c_h\). It reaches a maximum value of 1 at \(x=c_h\) and decays to 0 as \(x\) moves farther away.

The overall output of the network is modeled as a weighted combination of the outputs of each unit. Altogether, the model of the data is the expression

\[
f(x; \theta) = \sum_{h=1}^{H} \alpha_h f_h(x; \beta_h, c_h)
\]
Where $\theta = (\alpha_1, \ldots, \alpha_H, \beta_1, \ldots, \beta_H, c_1, \ldots, c_H)$ is the vector of network parameters, which are also known as weights. Our job is now to tune the weights in order to best fit the data.

To measure how closely the function fits the dataset for a given set of weights, we use the sum of absolute errors:

$$E(\theta) = \sum_{i=1}^{N} |f(x_i; \theta) - y_i|$$

where $x_i$ denotes the $(u,v)$ coordinates of the $i$'th sample point, and $y_i$ denotes its intensity value.

Questions

1. Implement an RBF model that can compute $f$ for an arbitrary point $x$ and weight vector $\theta$. Initialize a reasonable set of weights from a particular dataset.

   Describe your method for initializing the weights, and run it on a set of $N=1000$ samples with $H=10$. (Your method should be simple and fast, and not involve any iterative optimization.) Plot the resulting function. Ideally, you should plot a grayscale intensity image with the same size as your original image, but any contour plot should suffice.

2. Fit the network weights by running 10,000 iterations of random descent on $E(\theta)$. Describe your choice of step size. Report the final $E$ value and plot the optimized function.

3. Fit the network by running $M$ iterations of a metaheuristic algorithm of your choice (e.g., simulated annealing, evolutionary algorithms, etc.) on $E(\theta)$. Describe how you tuned the problem parameters (e.g., $M$, neighborhood size, annealing schedule, mutation strategy, etc.) in order to achieve reasonable performance. Report the final $E$ value, the number of times $E$ was evaluated, and plot the optimized function.

4. To ensure smoothness, switch to the sum of squared error function

   $$E(\theta) = \sum_{i=1}^{N} (f(x_i; \theta) - y_i)^2.$$  

   Use the gradient expressions for the RBF activation functions:

   $$\frac{\partial}{\partial \beta_h} f_h(x; \beta_h, c_h) = -\|x - c_h\|^2 \exp(-\beta_h \|x - c_h\|^2)$$

   $$\frac{\partial}{\partial c_h} f_h(x; \beta_h, c_h) = 2\beta_h (x - c_h) \exp(-\beta_h \|x - c_h\|^2)$$

   to give an expression for the overall gradient of $E$ with respect to the weights. Now, implement both a stochastic gradient descent and a batch gradient descent. Describe how you tuned the step size (i.e. learning rate) for stochastic gradient descent. You may tune a constant step size or use a line search for batch gradient descent.

   Start each of these descents from the best set of weights that you’ve obtained from problems 2 and 3. For both methods, report the final $E$ value, the number of times $E$ was evaluated, and
plot the resulting function.

5. Discuss the results of these experiments on this particular problem. What technique(s) would your recommend, and why? How do you think your recommended method would perform when H and N are larger? Run this method on H=100 and N=10,000, and report whether your intuition is confirmed.