

Lab 4

CS 3793 – Fall 2011
Tom Bylander, Instructor

assigned November 21, 2011
due midnight, December 7, 2011

In Lab 4, you will implement a program to recognize handwritten digits. Your grade on the lab will depend on the behavior and accuracy of the program that is implemented.

NIST environment

The NIST training dataset at <http://yann.lecun.com/exdb/mnist/> contains 60,000 images of handwritten digits and their labels. Each image is 28×28 in grayscale. For reasons of time and space efficiency in this lab, an 8×8 bitmap was derived from each image.

The `nist` program repeatedly does the following steps. It outputs a randomly selected (without repetition) bitmap as 64 ones and zeroes on a single line. It then inputs a prediction, a single digit on a line by itself. After the prediction, it outputs whether the prediction is correct or incorrect, the correct label, and a summary of the error rate so far.

The `-v` option presents a crude ASCII picture of the bitmap so that you can see how well you can do. To test your lab on all 60000 examples use `nist -n 60000`.

Agent

Your task is to write a program that will perform *online learning*, that is, its accuracy will improve as additional examples are presented to it. The suggested algorithms are least squares gradient descent or the perceptron algorithm.

In the `nist` environment, there are 10 possible outputs, the digits 0 through 9. Your program should have a weight vector for each digit. Specifically, your program should have 10 weight vectors, each with 65 weights.

Your program should read the input as 64 bits x_1, \dots, x_{64} , always using $x_0 = 1$. Then your program should calculate $\hat{y}_d = \mathbf{w}_d \cdot \mathbf{x}$ for each digit d . Your program should predict the digit with the highest \hat{y}_d .

To update your weights, your program should read in the correct digit y . For each digit d , assign $y_d = 1$ if $y = d$ else use $y_d = -1$. Then update \mathbf{w}_d using the update rule that you have chosen. You will get better results with least squares gradient descent if you do not update \mathbf{w}_d when $\hat{y}_d * y_d \geq 1$.

The class notes suggest a learning rate of $1/640$ for 64 inputs. A higher learning rate might result in faster learning or might result in unstable weights. Initially, all weights should be set to 0. The expected result is a decreasing error rate with additional examples. The last 1000 examples should have an error rate less than 20%.

Turning in Your Lab

Somewhere in your directory, you should have a `lab4` subdirectory. This directory *must* have a `Makefile` for compiling and linking your agent program, which *must* be named `lab4`. Any code that `lab4` depends on *must* be in this directory. Use Blackboard to submit your `lab4` directory as a zip file.