Programming Homework 2

CS540

June 8, 2019

1 Instruction

Please submit your output files and code on Canvas → Assignments → P2. Please do not put code into zip files and do not submit data files. The homework can be submitted within 3 weeks after the due date on Canvas without penalty (50 percent penalty after that).
Please add a file named "comments.txt", and in the file, you must include the instructions on how to generate the output, for example:

- Data files required: train.csv, test.csv. Run: main.jar.
- Data folder required: data/train1.png ... data/train100.png . Compile and Run: main.java.

2 Details

All the requirements are listed on the course website. The following is only an example workflow to solve the problem.

1. Download part 1 and part 2 of a set of images, for example, the spatially normalized cropped equalized frontal images and put them into one folder.
2. Resize the images to an appropriate size, for example, 36×26: training will be too slow for large sizes and accuracy will be too low for small sizes. Find the RGB values of each of these pixels and output the average of R, G, B values for each pixel to a CSV file (for each pixel $\frac{1}{3}(R+G+B)$), one image per line (for example, 36×26 integers between 0 and 255 per line). You could output the y value at the beginning of the line so that the file looks like the training data for programming homework 1.
3. Read the CSV file into $n \times m$ matrix x and $n \times 1$ vector y as in homework 1. Remember to take out the rows for the test set (the row numbers depend on your wisc ID) and put those in another matrix x'. You can keep the test rows in the training data if you want close to 100 percent accuracy. You can submit either version of the output, just make sure you state in comments.txt that you are training with all 400 images.
4. Initialize a $m \times m$ weight matrix $w^{(1)}$ for layer 1 and a $m \times 1$ vector $w^{(2)}$ for layer 2 and a $m \times 1$ bias vector $b^{(1)}$ for layer 1 and a scalar bias $b^{(2)}$ for layer 2. Randomly fill these matrices and vectors with numbers between -1 and 1. Also initialize a $m \times 1$ vector $a^{(1)}$ for units in layer 1 and a scalar $a^{(2)}$ for units in layer 2.

5. Get a random permutation of integers between 0 and $n-1$. You can search the terms Knuth shuffle or Fisher-Yates shuffle for a simple algorithm to shuffle an index set. For the next three steps, go through the training set in the order of the random permutation.

6. Calculate the activations $a^{(1)}$ and $a^{(2)}$ using the formulas from lecture 3 slides. For $j = 1, 2, ..., m$, and for the current instance i,

$$a_{ij}^{(1)} = \frac{1}{1 + \exp\left(-\left(\left(\sum_{j'=1}^{m} x_{ij'} w_{j'j}^{(1)}\right) + b_j^{(1)}\right)\right)}$$
$$a_i^{(2)} = \frac{1}{1 + \exp\left(-\left(\left(\sum_{j=1}^{m} a_{ij}^{(1)} w_j^{(2)}\right) + b^{(2)}\right)\right)}$$

7. Update the weights using the gradient descent formulas from lecture 3 slides. Please check to make sure these are correct!

$$\frac{\partial C}{\partial w_{j'j}^{(1)}} = \left(a_i^{(2)} - y_i\right) a_i^{(2)} \left(1 - a_i^{(2)}\right) w_j^{(2)} a_{ij}^{(1)} \left(1 - a_{ij}^{(1)}\right) x_{ij'}$$
$$\frac{\partial C}{\partial b_j^{(1)}} = \left(a_i^{(2)} - y_i\right) a_i^{(2)} \left(1 - a_i^{(2)}\right) w_j^{(2)} a_{ij}^{(1)} \left(1 - a_{ij}^{(1)}\right)$$
$$\frac{\partial C}{\partial w_j^{(2)}} = \left(a_i^{(2)} - y_i\right) a_i^{(2)} \left(1 - a_i^{(2)}\right) a_{ij}^{(1)}$$
$$\frac{\partial C}{\partial b^{(2)}} = \left(a_i^{(2)} - y_i\right) a_i^{(2)} \left(1 - a_i^{(2)}\right)$$

and then,

$$w_{j'j}^{(1)} \leftarrow w_{j'j}^{(1)} - \alpha \frac{\partial C}{\partial w_{j'j}^{(1)}}, j' = 1, 2, ..., m, j = 1, 2, ..., m$$
$$b_j^{(1)} \leftarrow b_j^{(1)} - \alpha \frac{\partial C}{\partial b_j^{(1)}}, j = 1, 2, ..., m$$
$$w_j^{(2)} \leftarrow w_j^{(2)} - \alpha \frac{\partial C}{\partial w_j^{(2)}}, j = 1, 2, ..., m$$
$$b^{(2)} \leftarrow b^{(2)} - \alpha \frac{\partial C}{\partial b^{(2)}}$$

It is very easy to make a mistake here. You can check if the gradient computation is correct by computing the gradient using finite differences and compare with your gradient.

$$\frac{\partial C}{\partial v} \approx \frac{C(v + \varepsilon) - C(v - \varepsilon)}{2\varepsilon}, \varepsilon = 0.0001$$

Here, v is one of $w^{(1)}, w^{(2)}, b^{(1)}, b^{(2)}$. Basically, compute the change in cost due to a very small increase and decrease of one of the weights, this by definition approximates the derivative of the cost with respect to that particular weight.

8. After going through all n instances (this is called an epoch), calculate the cost using the cost formulas

from lecture 3 slides. There is no need to store $a_i^{(2)}$ for all i, you can just accumulate the sum while going through the training data. Remember to generate another random permutation after each epoch.

$$C = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - a_i^{(2)}\right)^2$$

Note: if you did not shuffle the training set (without replacement), and instead picked a random instance at a time (with replacement), you need to recompute the activation $a_i^{(2)}$ for every $i = 1, 2, ..., n$ here. You can do this after an arbitrary number of iterations, not necessarily at the end of each epoch.

9. You can train for a fix number of epochs, say 100 or 1000, or use a small threshold ε and stop when $C \leqslant \varepsilon$. You can also calculate the cost on the test set and stop when that cost begin to increase for some fixed number of iterations (not necessary for this homework).

10. Given the final weights and biases you just computed, compute the activations and predictions for the test instances $x'_i, i = 1, 2, ..., n'$.

$$a_{ij}^{\prime(1)} = \frac{1}{1 + \exp\left(-\left(\left(\sum_{j'=1}^{m} x'_{ij'} w_{j'j}^{(1)}\right) + b_j^{(1)}\right)\right)}$$

$$a_i^{\prime(2)} = \frac{1}{1 + \exp\left(-\left(\left(\sum_{j=1}^{m} a_{ij}^{\prime(1)} w_j^{(2)}\right) + b^{(2)}\right)\right)}$$

$$\hat{y}'_i = \mathbb{1}_{\left\{a_i^{\prime(2)} \geqslant 0.5\right\}}$$

11. Output the activations for the first image in the test set to an image (or you can use the canvas on the course website to produce the image) and the classifications $\hat{\hat{y}}$' to a text file.