Programming Homework 1

CS540

May 31, 2019

1 Instruction

Please submit your output files and code on Canvas → Assignments → P1. Please do not put code into zip files and do not submit data files. The homework can be submitted within 2 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 the training data. The CSV files are easier to work with.
2. Create an array y to store the first column of the CSV file. Create $a2D$ array x to store the remaining columns. When reading the CSV file, assuming the digits you are asked to classify are A and B, if line i starts with A, push 0 to y and the rest of the line (each number divided by 255) to x, if line i starts with B, push 1 to y and the rest of the line to x, and if line i starts with other digits, skip the line. The number of columns of x (each one is a feature) is $m = 784$ and the number of rows of x (each row represent an image) depends on your digits is n. Also, y is a n by 1 vector.
3. Create an m by 1 array w and a double b. Initialize them with random numbers between -1 and 1. Random initializatioin between 0 and 1 is okay too, just slightly slower. Pick a learning rate α, the choice depends on the digits you are classifying, try different ones such as $\alpha = 0.1, 0.01, 0.001$ etc. Update the vector w according to the formula in the Lecture 2 slides.

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

$$w_j = w_j - \alpha \sum_{i=1}^{n} (a_i - y_i)\, x_{ij}$$

$$b = b - \alpha \sum_{i=1}^{n} (a_i - y_i)$$

Note: the notation $\exp(x)$ means e^x, the exponential function base e, similarly, $\log(x)$ means $\ln(x)$, the logarithmic function base e.

4. Remember to compute the cost at each step and store the cost from the previous step.

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

Important note: due to the problem that $0 \log 0$ is NaN in many programming languages, this cost function is easier implemented as the following.

$$\begin{aligned} C &= \sum_{i=1}^{n} \begin{cases} -\log(1 - a_i) & \text{if } y_i = 0 \\ -\log a_i & \text{if } y_i = 1 \end{cases} \\ &= \sum_{i=1}^{n} \begin{cases} -\log(1 - a_i) & \text{if } y_i = 0 \\ -\log a_i & \text{if } y_i = 1 \\ \text{something large} & \text{if } y_i = 0, a_i \text{ too close to 1 or } y_i = 1, a_i \text{ too close to 0} \end{cases} \end{aligned}$$

For example, something large can be 100, and too close to 1 can mean $a_i > 0.9999$ and too close to 0 can mean $a_i < 0.0001$. If C is computed this way, C will not be ∞ and easier to see whether C is decreasing after each iteration.

5. Repeat the previous two steps until the decrease in C is smaller than some ε, say 0.0001. Remember to set a maximum number of iterations, say 100 or 1000 (no need 10000), and if the program does not stop until then, change the learning rate α and try again.

6. Read the test files. Remember to divide x by 255. For each line $\hat{x}_i$, compute the predicted $\hat{y}_i$ using the following formula with the w and b you obtained previously.

$$\hat{y}_i = \begin{cases} A & \text{if } a_i < 0.5 \\ B & \text{if } a_i > 0.5 \end{cases}$$

$$a_i = \frac{1}{1 + \exp\left(-\left(\left(\sum_{j=1}^{m} w_i \hat{x}_{ij}\right) + b\right)\right)}$$

The percentage of correct classification should above 90 percent most of the pairs of digits.

7. Output w, b and $\hat{y}$ to text files.