CSE 3521: Survey of Artificial Intelligence I

Project 3: Hand-written digits recognition.

Due: 4/24 (Sun) 11:59 PM

Instructions:

1) Use C++, JAVA, or Python only.

2) Comment amply on the source code.

3) Double-check if your code compiles on stdlinux without errors.

4) Compress the source files and the compiling instructions into a single zip file.

4) Upload the zipped file onto Carmen.

Hand-written digit recognition with logistic regression. In this assignment you will train a logistic regression model to classify two digits "0" and "1" from the pixel values of hand-written digit images. You will be given training features and labels, and test features. Ground truth test labels will not be disclosed.

Examples of hand-written digit features:

Functions (or executables) to implement.

1. TrainLogReg

   Usage: TrainLogReg trainingFeatureFile trainingLabelFile modelFile D Niter

   Descriptions: The executable takes three string arguments and three numeric arguments. The first three are file names and the last two are training parameter. The function 1) reads training features and labels from files and store them in arrays (or vectors or matrices, if you want), 2) trains a logistic regression model using the data, and 3) writes the learned parameters to a file.

File formats.

- D (feature dimension) = 785, Ntrain = 12665, Ntest = 2115
- The training/testing feature file is a text file of integer numbers. Each line corresponds to a sample and has D number of features of that sample. There are N lines corresponding to N samples.
- The training label file has one number in each line – either 0 or 1 indicating which class each sample belongs to. You will set y=-1 if it's class 0 and y=1 if it's class 1.
- The model file contains just one line of D floating-point numbers.
- The predicted label file has the same format as the training label file – either 0 or 1 for each line as predicted by the logistic regression model.

Pseudocode:

Use the following pseudocode. You code should use standard math libraries to compute floating point multiplications. For the innerproduct of vectors $\mathbf{w}\cdot\mathbf{x}$, you can use vector math library or simply sum the scalar products $\mathbf{w}\cdot\mathbf{x}$ = w[0]*x[0] + w[1]*x[1] + …. w[D-1]*x[D-1].

1) Read N samples of D-dimensional features from trainingFeatureFile
2) Read N samples of 1-dimensional labels from trainingLabelFile
3) c = 1e-6, t ← 0, $\mathbf{w}$ ← 0 (Note that c and t are scalars, and w is a vector.)
4) for i = 1 to N (Note that this is a stochastic gradient descent with Iter=1 from the slides)

   t ← t + 1

   $\nabla_{\mathbf{w}} L = -y_i \mathbf{x_i} \exp(-y_i \mathbf{w}\cdot\mathbf{x_i}) / (1+ \exp(-y_i \mathbf{w}\cdot\mathbf{x_i}))$ (Note this a vector, not a scalar)

   $\mathbf{w} \leftarrow \mathbf{w} - (c / t) \nabla_{\mathbf{w}} L$

   end of i

   Note that the learning rate (= c / t) changes over time t.
5) Write the weight vector w to modelFile.

2. TestLogReg

Usage: TestLogReg modelFile testFeatureFile predLabelFile D

Description: The function 1) reads parameters learned from training data, 2) applies the logistic regression model to test features, and 3) writes the predicted label to a file.

Pseudocode:

1) Read the weight vector **w** from modelFile
2) Read the features from testFeatureFile
3) For each test feature **x**, predicted class is 1 if $\mathbf{w} \cdot \mathbf{x} > 0$ and 0 if $\mathbf{w} \cdot \mathbf{x} < 0$.
4) Write the predictions to predLabelFile

3. Accuracy

Usage: Accuracy predLabelFile trueLabelFile

Description: Compare two label files (one from prediction and the other ground-truth) and display the accuracy of prediction.

Pseudocode:

1) Read predLabelFile
2) Read trueLabelFile
3) Compute accuracy = # of correctly predicted samples / # of samples. (A sample is correctly predicted if the prediction (0 or 1) is the same as the ground truth (0 or 1) label of that sample.
4) Display accuracy

Getting started

1. Download the training features and labels, and the test features from Carmen-contents-programming assignments.
2. Implement TestLogReg and Accuracy first, since they are easier. To check if Accuracy is working properly, create an identical copy of the training label file. The Accuracy has to display the rate = 1.0 when those two file names are passed as arguments.
3. Implement TrainLogReg and debug it using the training data. To check if the code is working properly, i) run TrainLogReg, ii) run TestLogReg, and iii) run Accuracy with your predicted labels

and the trainingLabelFiles. The training accuracy should be very high (near 100%).

Grading criteria

1. (30 points) Submit 1) the source and executables for TrainLogReg / TestLogReg / Accuracy, 2) the trained model file, and 3) the predicted label file.

2. (30 points) Your predicted labels will be compared with the true test labels (which is not given to you) to evaluate the accuracy of your trained model.