CSE 546 Machine Learning, Autumn 2013
Homework 2

Due: Tuesday, Nov 3, beginning of class

1 Regression and feature selection [20 Points]

Let us do regression on synthetic data (where you generate the data). Throughout make sure to use a number of dimensions $d \geq 50$.

1. (10 points) Generate data where $y_i = w_*^\top x_i + \eta$ where η is some noise model. Make sure that w_* is non-zero in all directions. Fit this model (for a few different sample sizes) and show how your parameter error, say measured under $\|w_* - \hat{w}\|_2^2$, decreases with the sample size n. Also, comment on how your parameter error decreases with n. (Consider generating the vectors $x_1, \ldots x_n$, from a Gaussian distribution).

2. (10 points) Now do the same where you make it so that only one coordinate of w_* is nonzero. Do the subset selection algorithm on your dataset for some different values of n. Also do the linear regression algorithm for different values of n. Comment on how they behave differently.

2 Programming Question [45 Points]

In this problem, you will train a logistic regression model to predict the Click Through Rate (CTR) on a dataset with about 10,000 examples. The CTR provides a measure of the popularity of an advertisement, and the features we will use for prediction include attributes of the ad and the user.

2.1 Unprocessed Dataset

The dataset we will consider comes from the 2012 KDD Cup Track 2. Here, a user types a query and a set of ads are displayed and we observe which ad was clicked.

For example:

1. Alice went to the famous search engine Elgoog, and typed the query "big data".

2. Besides the search result, Elgoog displayed 3 ads each with some short text including its title, description, etc.

3. Alice then clicked on the first advertisement

This completes a SESSION. At the end of this session Elgoog logged 3 records:

Clicked = 1 | Depth = 3 | Position = 1 | Alice | Text of Ad1 |
Clicked = 0 | Depth = 3 | Position = 2 | Alice | Text of Ad2 |
Clicked = 0 | Depth = 3 | Position = 3 | Alice | Text of Ad3 |

In addition, the log contains information about Alice's age and gender. Here is the format of a complete row of our training data:

Clicked | Depth | Position | Userid | Gender | Age | Text Tokens of Ad

Let's go through each field in detail:

- "Clicked" is either 0 or 1, indicating whether the ad is clicked.

- "Depth" takes a value in $\{1, 2, ..., \}$ specifying the number of ads displayed in the session.

- "Position" takes a value in $\{1, 2, ..., Depth\}$ specifying the rank of the ad among all the ads displayed in the session.

- "Userid" is an integer id of the user.

- "Age" takes a value in $\{1, 2, 3, 4, 5, 6\}$, indicating different ranges of a user's age: ''1' for (0, 12], '2' for (12, 18], '3' for (18, 24], '4' for (24, 30], '5' for (30, 40], and '6' for greater than 40.

- "Gender" takes a value in $\{-1, 1\}$, where -1 stands for male and 1 stands for female.

- "Text Tokens" is a comma separated list of token ids. For example: "15,251,599" means "token_15", "token_251", and "token_599". (Note that due to privacy issues, the mapping from token ids to words is not revealed to us in this dataset, e.g., "token_32" to "big".)

Here is an example that illustrates the concept of features "Depth" and "Position". Suppose the list below was returned by Elgoog as a response to Alice's query. The list has $depth = 3$. "Big Data" has $position = 1$, "Machine Learning" has $position = 2$ and so forth.

| Big Data |
| Machine Learning |
| Cloud Computing |

Here is a sample from the training data:
0 | 2 | 2 | 280151 | 1 | 2 | 0, 1, 154, 173, 183, 188, 214, 234, 26, 3, 32, 36, 37, 4503, 51, 679, 7740, 8, 94
The test data is in the same format except that each instance does not have the first label field, which is stored in a separate file named "test_label.txt". Some data points do not have user information. In these cases, the userid, age, and gender are set to zero.

2.2 Processed Dataset (that you will use...)

In class, we simply denote

$$x^t = [x_1^t, ..., x_d^t] \tag{1}$$

as an abstract feature vector. In the real world, however, constructing the feature vector requires some thought. We have preprocessed the dataset for you, making the following modifications.

- First of all, not everything in the data should be treated as a feature. In this dataset, "Userid" should not be treated as a feature. We have removed this column from the dataset for you.

- Similarly, we cannot directly use the list of token ids as features in Equation 1 since the numbers are arbitrarily assigned and thus meaningless for the purposes of regression. Instead, we should think of the list of token ids $L = [l_1, l_2, l_3, ...]$ as a compact representation of a sparse binary vector b where $b[i] = 1 \forall i \in L$. We have replaced the list of words with a list of binary variables indicating token presence.

- As for the rest of the features: "Depth", "Position", "Age", and 'Gender", they are scalar variables, so we maintain their original values in the dataset.

The dataset that we are giving you has the following form:
Clicked, Depth, Position, Gender, Age, Word1, Word2, Word3, Word4, ..., Word50

So a sample training datapoint would be:
0, 1, 2, 1, 12, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0

Again, the test data is in the same format except that each instance does not have the first label field, which is stored in a separate file named "test_label.txt".

2.3 Accessing and Processing the Data

1. Download the processed version of the dataset "clickprediction_data.tar.gz" from the course website.

2. After extracting, there should be five files: train.txt, test.txt test_label.txt, start.py (which contains code to load the data) and oversampled_train.txt (more on this file later. For now, just ignore it).

2.4 Batch Gradient Descent (18 points)

Recall that logistic regression can be optimized by gradient descent (or ascent). We call it *batch* gradient descent because we're using the whole training data in order to update the weights in every iteration, in contrast to *stochastic* gradient descent. In class you discussed maximizing the regularized conditional log-probability. Here we'll be minimizing the regularized log-loss (the updates will be the same):

$$l(\mathbf{x}, w_0, \mathbf{w}) = \frac{1}{2}\lambda||\mathbf{w}||_2^2 - \frac{1}{N}\ln\prod_j P(y^j|\mathbf{x}^j, w_0, \mathbf{w})$$

1. *(1 pts)* Write down the equation for the batch weight update step. That is, how to update weights w^{t+1} using $(\mathbf{X}, \mathbf{y}, w^t, \lambda)$. Assume we're using l^2 regularization and step size η.

2. *(8 pts)* Implement gradient descent and run it for 1000 iterations, initializing w with a vector of all zeros. Remember not to regularize w_0!

   (a) Plot the log-loss function after each iteration, using $\eta = 0.1$ and $\lambda = 0.3$. Remember to include the regularization term.

   (b) Using the weights learned above, predict the CTRs for the test data and report the SSE (sum of squared errors). When computing the SSE, make sure you are actually predicting the class and not just using $Pr(y^j|\mathbf{x}^j, w_0, \mathbf{w})$ (i.e. you should be thresholding at 0.5). (Hint: the SSE will just be the number of incorrect predictions)

3. *(9 pts)* Now implement the following stop criteria: stop when $|l(w^t) - l(w^{t-1})| < \epsilon$, where l is the log-loss function. Run gradient descent with $\eta = 0.1$, $\lambda = 0.3$ and $\epsilon = 0.0005$.

   (a) For how many iterations did gradient descent run?

   (b) Plot the log-loss function after each iteration (now the x-axis should not go until 1000 as it did in the previous question).

   (c) Use the weights you learned to predict the CTRs for the test data, and report the SSE of your predicted CTR.

2.5 Stochastic Gradient Descent (17 points)

Recall that stochastic gradient descent (SGD) performs a gradient descent using a noisy estimate of the full gradient based on just the current example.

1. *(1 pts)* Write down the equation for the weight update step. That is, how to update weights w^{t+1} using the data point (x^j, y^j) and w^t, where $x^j = [x_1^j, x_2^j, ..., x_d^j]$ is the feature vector for example j, and $y^j \in \{0, 1\}$ is the label. Once again, use l^2 regularization.

2. *(8 pts)* Implement SGD and train the weights by making one pass over the dataset. You may assume that the data is randomly ordered, so don't reorder it. Use $\eta = 0.1$ for the following:

   (a) Run one pass over the data with $\lambda = 0$, and report the l^2 norm of the weights at the end (excluding the offset w_0). Repeat with $\lambda = 0.3$.

   (b) Use the weights you found with $\lambda = 0.3$ to predict the CTRs for the test data and report the SSE as before.

3. *(8 pts)* Now run SGD for 5 passes over the data, with $\eta = 0.1$ and $\lambda = 0.3$, and report the value of the log-loss function on the training data after the 5 passes. Run batch gradient descent for 5 iterations with the same parameters (the computing cost will be in the same ballpark), and report the value of the log-loss function. Which algorithm seems to converge faster?

2.6 Class Imbalance (10 points)

Now take a look at your predictions on the test data, for either SGD or batch gradient descent. More specifically, take a look at how many times your algorithm predicted a 1. Due to class imbalance (few 1's), SSE is not an adequate measure of quality here. Instead, we will now use two class-specific metrics: precision and recall, defined as follows:

Given a class c, lets say that there are two groups: A, the group of instances that are in class c, and B, the group of instances that your algorithm classified as being in class c. Precision is defined as $\frac{|A \cap B|}{|B|}$, and recall is defined as $\frac{|A \cap B|}{|A|}$. If an algorithm doesn't classify any instances on class c, let's define the precision as 0.

1. *(5 pts)* Run 5000 iterations of batch descent over the data with $\lambda = 0.3$ and $\eta = 0.01$. Using the learned weights, report the precision and recall for classes 0 and 1 on the test set.

2. *(5 pts)* A very simple (and not the best!) solution to the problem of class imbalance is to oversample from the class that has fewer instances until the imbalance has been corrected. This introduces other biases into the dataset, as the instances of that class will be repeated several times. The file "training_oversample.txt" contains a more even distribution of classes (the positive examples are oversampled). Now run 5000 iterations of batch descent on this training set ($\lambda = 0.3, \eta = 0.01$), and report the precision/recall values for classes 0 and 1 on the test set.

3 Extra Credit: MNist [50 Points]

Obtain the MNist dataset from `http://yann.lecun.com/exdb/mnist/`.

1. (10 points) Do linear regression on raw features on this dataset. What classification error do you get? (Remember that you might need to regularize if things are not stable).

2. (10 points) Do logistic regression on this dataset. What classification error do you get?

3. (30 points) Do anything that gets you below 1.8% error. Also, in class, I obtained around 1.15% error with random features (I used over 100k features. There was one trick I did which seems helpful to actually get below 1.4% error, which we will look at later in class). Overall, do anything you like here, so long as you find a way to drop below 1.8% error.

4 Extra Credit: Smoothness [50 Points]

A convex function G is said to be L-smooth if, for all w and w', we have that:

$$G(w') \leq G(w) + \nabla G(w) \cdot (w' - w) + \frac{L}{2}\|w - w'\|^2 \tag{2}$$

The above equation provides an upper bound for the function G. An alternative definition is that a convex function G is said to be L-smooth if, for all w and w', we have that:

$$\|\nabla G(w) - \nabla G(w')\| \leq L\|w' - w\| \tag{3}$$

This latter equation says that the gradients are (Lipschitz) continuous.

Let us show these definitions are identical.

1. (15 points) Assume equation 3 holds. Prove that equation 2 holds. (You will get most of the credit if you can prove equation 2 with a constant of L rather than $\frac{L}{2}$, which is a little easier.).

2. (35 points) Assume equation 2 holds. Prove that equation 3 holds. (This question is a little tricky. You will get most of the credit if you can prove equation 3 with $2L$ rather than L, which is a little easier.).