Abstract
In this homework report, the author implemented a batch model logistic regression both with cross validation and without cross validation. In the stage of searching a set of coefficients of logistics regression to minimize the average loss, the author employed the gradient descent algorithm with an exponentially decreasing learning rate. Then, regularizations were introduced to this model. Models under different regularizations parameters were evaluated. Finally the author added label shrinking to this model. MATLAB was the computational tool here for our simulation. Given several reasonable parameters of such algorithm by the homework instruction plus some ones selected by the author’s view, the author ran the simulations on the E-mail spam dataset. The simulation showed logistic regression without cross validation has a serious over-fitting problem. Using cross validation can alleviate it to some extent. By using 2-norm regularization, over-fitting is significantly restrained. By comparing these models, it’s very clear that the best average loss on test set of model with regularization and label shrinking is superior to that of model without them. 2-norm regularization plays a very important role in controlling the over-fitting.

I. Basic model

Basic model here means it runs without any label shrinking and regularization. Since the algorithm has been elaborated on the homework sheet, it is not necessary to re-illustrate the flow chart of the algorithm in this report. Also there is no need to re-clarify the partition of dataset because of the instruction on the homework sheet.

Basically, in the author’s understanding, the task here is to run batch model logistic regression as stated in the abstract. The goal is to minimize the average loss. According to that, when we talked about updating the coefficients $w_q$, the q referred to the times of iteration using all the samples from time 1 to T, not refer to the sample time t. Learning rate here stands for the iterative step length, measuring how long it moves from the position of the last w to that of the current w along the direction of the gradient.
However, that assumption doesn’t mean online model could not be planted. Actually, for each time of iteration, instead of using a single gradient vector that has been already averaged over T, using online learning algorithm to calculate the $w_{q+1}$ from $w_q$ by doing T times inner learning cycles is an alternative choice.

The author picked up the batch model without online learning. This is based on the consideration of accelerating the calculation. The parameters by the homework sheet are given below.

<table>
<thead>
<tr>
<th>Name of variable</th>
<th>Corresponding parameter</th>
<th>Initialized value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>How many passes</td>
<td>100</td>
</tr>
<tr>
<td>ETA0</td>
<td>Initial learning rate</td>
<td>0.2</td>
</tr>
<tr>
<td>ALPHA</td>
<td>Decay rate of Learning rate</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Here we come to another issue that how we define “Passes”. In batch model, in order to render this term meaningful and also to make the duration of whole simulation endurable, it should stand for the times of iteration.

The author didn’t deny that it might stand for other meanings under different settings of the model. For example, it might be how many times the learning cycle, which contains T times inner learning updating in each, will be repeated for just one updating $w_q$ to $w_{q+1}$, if we consider to mix the online learning and batch model. But under different settings of this model, the reasonable values of these parameters vary.

After acquiring some preliminary simulation results, we could see that the suggested parameters don’t work very well. If we just set the initial learning rate to 0.2, it is too small to achieve enough movement to minimum point during each time of iteration. Thus, the stopping gradient is very large, about 3~5.5. That means the final coefficients w is far from an optimized one. The author guessed that is because this set of parameter may be set for gradient descent on total loss, not the average.

Thus, we need to change our parameters to reasonable ones which are given below.

<table>
<thead>
<tr>
<th>Name of variable</th>
<th>Corresponding parameter</th>
<th>Initialized value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>How many passes</td>
<td>100</td>
</tr>
<tr>
<td>ETA0</td>
<td>Initial learning rate</td>
<td>30</td>
</tr>
</tbody>
</table>
The results here show that the stop gradient is much smaller than previous one. The author believed that this exponentially decaying learning rate could be regarded as an approach to get lower gradient rapidly. That means, if we adjust ETA0 to a proper large value, just after 100 times iteration, a decent low gradient could be achieved.

However, if we set ETA0 to a fixed number, it’s difficult to have the same criteria when we evaluate the performance. That is because for each top cycle we permute the data. Thus if we set P to a fixed number, even when the ETA0 and ALPHA maintains the same value, the stopping gradient varies as permutation varies so that it’s not easy to control. Thus, here we can adjust P value dynamically. We usually do P times iteration until the gradient reaches to a certain value. Actually, adjusting the ETA0 dynamically could also be a method to achieve stopping gradient control.

In the above figure, it showed the relationship between the stopping gradient value and average loss of training set, and also the relationship between the
stopping gradient and the average loss of training set. Here we could easily catch the over-fitting. We could see when the stopping gradient is very small the training set is fitted very well but the test set is fitted very badly.

It’s interesting to see that average loss of training set decreases when stopping gradient value decreases from $10^{-4}$ to $10^{-2}$. In order to make this clear, the author did the simulation without cross validation.

The above figure shows the result of simulation without cross validation. It’s clear that without cross-validation, the training set is fitted almost perfect. However, compared with the model with cross validation, the over-fitting problem is getting worse without cross validation.

**II. Model with Regularization and label shrinking**

In this section, we add the regularization to the model. First I add 2-norm regularization.

When lamda is equal to 2, the results are shown below.
When lambda is equal to 4, the result is
It is similar to the result when lamda is equal to 2. However, the gap between the loss of training set and that of test set is closer.

From these two figures, we could see that over-fitting is significantly decreased. When the gradient is very small, the loss of training set is decent and the loss of test set doesn’t go up as that of test set in no regularization model.

Then we add label shrinking to this regularization model. We set a 0.2 and set b 0.8, the result is

![Graph showing training and test set losses](image)

It still looks very decent.

The author tried 1-norm regularization. However, it didn’t work since the gradient of 1-norm regularization seems to be very difficult to decrease.

### III. Comparison

From above figures, we could see the best average loss on test set of model without regularization and label shrinking is about 0.7 while that of model with regularization and label shrinking is about 0.5, which is definitely superior to
the non regularization model. This fact coincides with the fact written on the lecture slides. Here the author reproduces that.