CSCI-SHU 360 Machine Learning
Solution to homework 3

Yufeng Xu yx3038@nyu.edu

March 26, 2024

1 Logistic Regression

1.1

Let $X = [X, 1] \in \mathbb{R}^{n\times(d+1)}$, weights $W = [W, b] \in \mathbb{R}^{(d+1)\times c}$. We denote $X_{i,\cdot}$ as X_i, $W_{\cdot,j}$ as W_j, then we have:

$$\begin{aligned}
F(W) &= \frac{1}{n}\sum_{i=1}^{n} -\log[Pr(y = y_i|x = X_i; W)] + \frac{\eta}{2}||W||_F^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} -\log[\frac{\exp(z_{y_i})}{\sum_{j=1}^{c}\exp(z_j)}] + \frac{\eta}{2}||W||_F^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(-z_{y_i} + \sum_{i=1}^{n}\log[\sum_{j=1}^{c}\exp(z_{ij})]) + \frac{\eta}{2}||W||_F^2 \\
&= -\frac{1}{n}\sum_{i=1}^{n} z_{y_i} + \frac{1}{n}\sum_{i=1}^{n}\log[\sum_{j=1}^{c}\exp(z_{ij})] + \frac{\eta}{2}||W||_F^2 \\
&= -\frac{1}{n}\sum_{i=1}^{n}(X_i W_{y_i}) + \frac{1}{n}\sum_{i=1}^{n}\log[\sum_{j=1}^{c}\exp(X_i W_j)] + \frac{\eta}{2}\sum_{j=1}^{c}||W_j||_F^2
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial F(W)}{\partial W_j} &= -\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{y_i = j\}X_i^T + \frac{1}{n}\sum_{i=1}^{n}\frac{\exp(X_i W_j)X_i^T}{\sum_{k=1}^{c}\exp(X_i W_k)} + \eta W_j \\
&= \frac{1}{n}\sum_{i=1}^{n}(\frac{\exp(X_i W_j)}{\sum_{k=1}^{c}\exp(X_i W_k)} - \mathbb{1}\{y_i = j\})X_i^T + \eta W_j \\
&= \frac{1}{n}\sum_{i=1}^{n}(Pr(y = j|X_i; W) - \mathbb{1}\{y_i = j\})X_i^T + nWj \\
&= \frac{1}{n}X^T\begin{pmatrix} Pr(y = j|X_1; W) - \mathbb{1}\{y_1 = j\} \\ \vdots \\ Pr(y = j|X_n; W) - \mathbb{1}\{y_n = j\}\end{pmatrix} + \eta W_j
\end{aligned}$$

Let $Y \in \mathbb{R}^{n\times c}$ where $Y_{ij} = \mathbb{1}\{y_i = j\}$, $P \in \mathbb{R}^{n\times c}$ where $P_{ij} = Pr(y = j|X_i; W)$. Then,

$$\frac{\partial F(W)}{\partial W} = \begin{bmatrix}\frac{\partial F(W)}{\partial W_1} \cdots \frac{\partial F(W)}{\partial W_c}\end{bmatrix}$$

$$= \frac{1}{n} X^T \left(P_{\cdot,1} - Y_{\cdot,1} \dots P_{\cdot,c} - Y_{\cdot,c} \right) + \eta W$$
$$= \frac{1}{n} X^T (P - Y) + \eta W$$

1.2

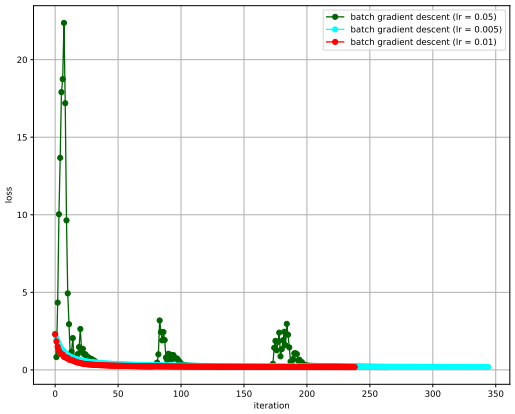

Figure 1: Loss-Iteration curve of vanilla logistic regression with different learning rates

1.3

By comparing the curves Figure 1, we have 2 observations on the pros and cons of large/small learning rates:
(1) **large learning rates accelerate the convergence of training.** When learning rate=0.05, the training terminates at 260 iterations, and when learning rate=0.01, the training terminates at 240 iterations. When learning rate=0.005, however, the training terminates at 350 iterations, which prolongs the training time. However, **large learning rates may also result in a noisier training process.** When learning rate=0.05, there are 3 sharp spikes in the loss curve, while the loss curve for learning rate=0.01 and 0.005 are much smoother. Sometimes when the learning rate is too large, the loss function may even fail to converge.
(2) In contrast to large learning rates, **small learning rates lead to smoother training process, but slower convergence.** The optimal strategy is to find a sweet spot between large and small learning rates. In this case, 0.01 is the best learning rate, which converges the fastest and has no "spikes" in the loss curve.

1.4

In this section, we investigate the performance of SGD, and our finetuning is focues on 4 aspects: (1) batch size; (2) learning rate; (3) weight decay; (4) learning rate annealing; (5) maximum epochs.

Batch size: we performed an automatic finetuning on 3 batch sizes - 10, 50, and 100. The moderate batch size - 50, results in the fastest convergence of the loss function; when other hyper parameters are chosen properly, the smaller batch sizes always perform comparably or even outperform large batch sizes.

Learning rate: For most cases, learning rate=0.01 leads to a good performance of the model. However, when the batch size is very small, the learning rate needs to be cut down. For instance, when batch size=10, the learning rate needs to be cut down to 0.001 to maintain a comparable performance; when batch size = 1, the learning rate needs to be reduced to 0.0001. We suspect this is because smaller batch size leads to noisier gradients, and smaller learning rates help to mitigate this impact.

Weight decay(η): When the weight decay is 0, there is no significant gap between train and test accuracy, which indicates generalizability is not the bottleneck that limits the model's performance. However, it takes much more epochs for the loss to converge when $\eta = 0$, which means a positive η boosts the optimization of the model.

When weight decay is 0.01, both the train and test accuracy degrade, meaning 0.01 is too large. When weight decay is 0.005, the test accuracy is the best, 97.33%.

Learning rate annealing(α): We tried different annealing strategies, with annealing rate = 0.5/0.33/0.1, or no annealing at all(rate=1), the final precision is almost not affected.

Max epoch: Initially we set the max epoch to 500, but the training always terminates far before 500 epoch. When weight decay is 0, the training terminates at 175 epoch; when weight decay $\geq$ 0.05, the training terminates at 50-70 epoch. We decided max epoch is not an important factor in fine tuning.

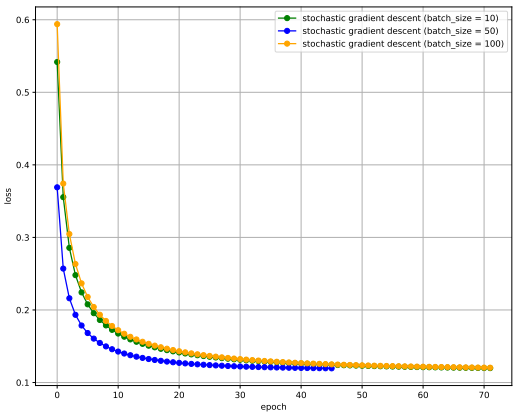

Figure 2: Loss-Iteration curve of minibatch logistic regression with different learning rate=0.001/0.01, weight decay=0.05, max epoch=500, annealing rate=0.5.

batch size	10	50	100
learning rate	0.001	0.01	0.01
final F(w)	0.1199	0.1195	0.1205
train accuracy	99.11%	99.18%	99.03%
test accuracy	97.33%	97.33%	97.33%

Table 1: final loss value, train accuracy, and test accuracy when learning rate=0.001/0.01, weight decay=0.05, max epoch=500, annealing rate=0.5.

Considering all the experiment results we obtained, we decide the set of optimal hyper parameters are: **batch size=50, learning rate=0.01, and weight decay=0.05. The impact of learning rate annealing and max epoch on the performance can be ignored.**

1.5

In order to investigate the convergence of the curves, we first fixed the learning rate to 0.01, then compared the convergence speeds corresponding to batch size=10/50/100. The results are as follows.
From Figure 3, we can see although the curves converge faster when batch size is small (10/50), the curve with batch size 10 ends up with a much larger final value of loss function.

Next, we scaled the learning rates linearly w.r.t the batch sizes. Because batch size=50 performs the best

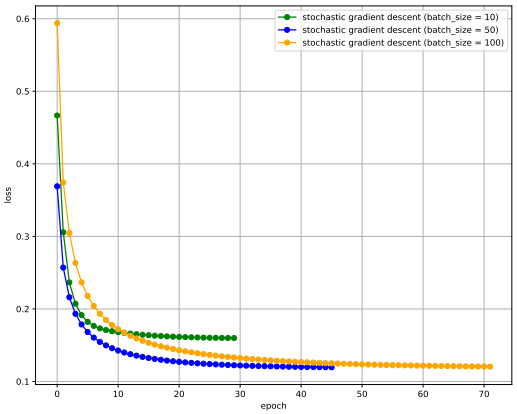

Figure 3: Loss-Iteration curve of minibatch logistic regression with earning rate=0.01, batch size=10/50/100.

when learning rate=0.01, we assume 0.01 is the appropriate learning rate for batch size=50, and adjust the learning rates for batch size=10/100 based on 0.01: learning rate $= 0.01 \cdot \frac{10}{50} = 0.002$ for batch size=10, learning rate $= 0.01 \cdot \frac{100}{50} = 0.02$. The results are as follows:

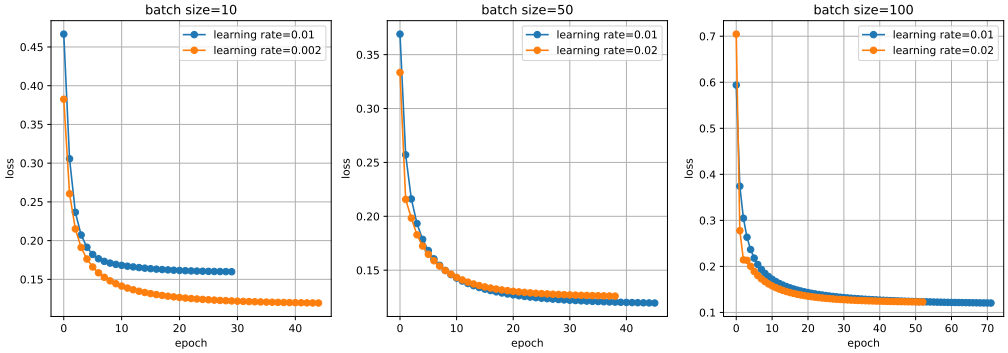

Figure 4: Loss-Iteration curve of minibatch logistic regression with earning rate=0.01, batch size=10/50/100.

As shown in the left and the right plot, scaling the learning linearly w.r.t the batch size boosts the training, either reducing the final value of the loss function or accelerating the convergence. Meanwhile, after changing learning rate from 0.01 to 0.02 for batch size=50, the performance becomes worse as the final loss value becomes larger, and test accuracy drops from 97.3% to 96.4%.
A mathematical explanation to this phenomenon is that: as the gradient is scaled by $\frac{1}{n}$, the stride of the gradient of each batch is almost constant regardless of the batch size. However, when batch size is smaller, the weights are updated more times for each epoch, resulting in faster convergence.
Meanwhile, due to the variance of the data samples, smaller batch size results in noisier gradient for each step. If the learning rate remains the same, the loss function is likely to oscillate around the minimum but fails to reach the minimum. Therefore, a smaller learning rate is adopted to make the loss function converge better at the price of slower convergence.
More specifically, # **times the weights are updated in each epoch** $\propto \frac{1}{\text{batch size}}$, $|$**gradient**$|$ **of each batch** is almost constant, so when **learning rate** $\propto$ batch size, the amount of update in each epoch $\approx$ #time updated $\times$ learning rate $\times$ gradient almost remains constant, resulting in comparable model per-

formance.

2 Lasso