Abstract

In this report we analyze two types of logistic regression models and carry out experiments using the UCI Adult dataset. We compare several factors that related to the prediction accuracy and discuss reasons of those correlations. Finally we propose a reasonable configuration of parameters and schemes for training models. Experiments show that for both models we achieved around 84% accuracy.

1 Introduction

This project tries to train a logistic regression model that predicts whether an individual has an annual income larger than $50K or not. We use the UCI Adult dataset, which contains 48,842 examples with a total of 14 features. First, we derive the update rule and analyze the convexity for both unbounded and bounded models. Then we describe the experiments methodology and show the results. We compare and discuss factors that affect the prediction accuracy, and finally give a proper configuration of parameters and schemes for training models.

2 Model Analysis

We compare two models in this problem. The first one is a regularized logistic regression model, and the second a regularized bounded logistic regression model. We analyze properties of these two models in this section and compare their prediction accuracy in the experiment part.

2.1 Regularized Logistic Regression Model

As described in the notes, the gradient of the objective function in the regularized logistic regression model is given by

$$\frac{\partial}{\partial \beta_j} \left[ \log p(y|x; \beta) - \mu \sum_{j=0}^{d} \beta_j^2 \right] = \frac{\partial}{\partial \beta_j} \left[ \log p(y|x; \beta) \right] - 2 \mu \beta_j = \sum_i (y_i - p_i)x_{ij} - 2 \mu \beta_j \quad (1)$$

The update rule for SGD with regularization is

$$\beta_j := \beta_j + \lambda [(y - p)x_j - 2 \mu \beta_j]. \quad (2)$$

The SGD is guaranteed to find the global maximum only if the objective function is convex. We check this by computing the second partial derivative with respect of $\beta_j$,

$$\frac{\partial^2}{\partial \beta_j^2} \left[ \log p(y|x; \beta) - \mu \sum_{j=0}^{d} \beta_j^2 \right] = -2 \mu - \sum_i p_i (1 - p_i)x_{ij}^2 \quad (3)$$
Given \( p_i \in (0, 1) \) and \( \mu > 0 \), the second partial derivative is always non-positive. Therefore, this model results in a convex optimization problem, i.e, the stochastic will converge to the global maximum.

### 2.2 Regularized Bounded Logistic Regression Model

The bounded logistic regression model is given by

\[
p(y = 1|x; \theta) = \sigma(f) + (\sigma(c) - \sigma(f))\sigma(w^T x + b).
\]

(4)

Let \( w^T x + b = \beta^T x \), so \( \beta \in \mathbb{R}^{d+1} \), as in the first model. \( f \) and \( c \) are extra parameters to be trained in this bounded model. Intuitively, \( f \) and \( c \) present respectively the lower and upper bound on the weight sum of features. They are passed through a sigmoid to map to probability value range \((0, 1)\). In this way it guarantees the probability \( p \) always satisfies \( 0 < \sigma(f) < p < \sigma(c) < 1 \), while the range of \( c \) and \( f \) is \(-\infty < f < c < +\infty\).

The regularized bounded logistic regression model is more general because the lower and upper bound of \( p \) can be adjusted. When \( f \to -\infty \) and \( c \to +\infty \), the second model turns into the first model. The upper and lower bound may reflect some features that may affect \( p \) but not considered in the first model.

**Update rule:**

We find the update rule by deriving the partial derivative with respective to each parameter.

For an individual training example \((x_i, y_i)\), the partial derivative of \( p_i \) with respect to \( \beta_j \), \( c \), \( f \) is

\[
\frac{\partial p_i}{\partial \beta_j} = [p_i - \sigma(f)][1 - \sigma(\beta^T x_i)]x_{ij}
\]

(5)

\[
\frac{\partial p_i}{\partial c} = \sigma(c)[1 - \sigma(f)]\sigma(\beta^T x_i)
\]

(6)

\[
\frac{\partial p_i}{\partial f} = \sigma(f)[1 - \sigma(f)][1 - \sigma(\beta^T x_i)]
\]

(7)

For the entire training set the partial derivative of the LCL with respect to \( \beta_j \), \( c \), \( f \) is

\[
\frac{\partial}{\partial \beta_j} LCL = \sum_i \frac{[p_i - \sigma(f)][1 - \sigma(\beta^T x_i)]x_{ij}}{y_i + p_i - 1}
\]

(8)

\[
\frac{\partial}{\partial c} LCL = \sum_i \frac{\sigma(c)[1 - \sigma(f)]\sigma(\beta^T x_i)}{y_i + p_i - 1}
\]

(9)

\[
\frac{\partial}{\partial f} LCL = \sum_i \frac{\sigma(f)[1 - \sigma(f)][1 - \sigma(\beta^T x_i)]}{y_i + p_i - 1}
\]

(10)

Therefore the update rule with regularization is:

\[
\beta_j := \beta_j + \lambda \frac{[p - \sigma(f)][1 - \sigma(\beta^T x)]x_j}{y + p - 1} - 2\mu \beta_j
\]

(11)

\[
c := c + \lambda \frac{[\sigma(c)[1 - \sigma(f)]\sigma(\beta^T x)]}{y + p - 1} - 2\mu c
\]

(12)

\[
f := f + \lambda \frac{[\sigma(f)[1 - \sigma(f)][1 - \sigma(\beta^T x)]}{y + p - 1} - 2\mu f
\]

(13)

**Convexity:**

To check the convexity of this model, we find the second derivative of the LCL.

The second partial derivative of the LCL with respect to \( \beta_j \) is

\[
\frac{\partial^2}{\partial \beta^2_j} LCL = -\sum_i \frac{[p_i - \sigma(f)][1 - \sigma(\beta^T x_i)]x_{ij}^2}{(y_i + p_i - 1)^2} \left[ (1 - \sigma(f) - y_i)[1 - 2\sigma(\beta^T x_i)] + [\sigma(c) - \sigma(f)]\sigma(\beta^T x_i)^2 \right].
\]

(14)
Because \((1 - \sigma(f) - y_i)[1 - 2\sigma(\beta^T x_i)]\) has no definite sign, LCL is not convex respect to \(\beta_j\). But if the initial \(\beta_j\) is close to the desired value, then this term can be positive and this second partial derivative is negative, which will lead \(\beta_j\) to the desired value. The second partial derivative of the LCL with respect to \(c\) is

\[
\frac{\partial^2 LCL}{\partial c^2} = -\sum_i \sigma(c)[1 - \sigma(c)]\sigma(\beta^T x_i)\left\{\left[2\sigma(c) - 1\right](y_i + p_i - 1) + \sigma(c)[1 - \sigma(c)]\sigma(\beta^T x_i)\right\}
\]

Because \(2\sigma(c) - 1\)(\(y_i + p_i - 1\)) doesn’t have definite sign, LCL is not convex respect to \(c\).

The second partial derivative of the LCL with respect to \(f\) is

\[
\frac{\partial^2 LCL}{\partial f^2} = -\sum_i \sigma(f)[1 - \sigma(f)]\sigma(\beta^T x_i)\left\{\left[2\sigma(f) - 1\right](y_i + p_i - 1) + \sigma(f)[1 - \sigma(f)]\sigma(\beta^T x_i)\right\}
\]

Because \(2\sigma(f) - 1\)(\(y_i + p_i - 1\)) has no definite sign, LCL is also not convex respect to \(f\).

Therefore, the regularized bounded logistic regression model does not result in a convex problem for any of its parameters. There is no guaranteed convergence to the global minimum. This property is shown with experiments in Section 4.4.

3 Experiments Methodology

We evaluate the two models on the UCI Adult dataset, which is separated into 32,561 training samples and 16,281 testing samples. First, we preprocess the dataset by encoding categorical features, and normalizing real-valued features. Then we train the two models and test their prediction accuracy using cross-validation. The effect of various parameters or schemes on the accuracy are analyzed using controlled experiments.

3.1 Preprocessing

We convert the original dataset into binary and integer values. The encoding process contains

- We encode categorical features with one-hot coding, in which a categorical feature with \(k\) values is encoded using a binary of \(k - 1\) bits. One of the \(k\) values corresponds to the all-zero binary, and the rest \(k - 1\) values correspond to binaries in which exactly one bit is set.
- As age group is more meaningful in prediction than the age number itself, we convert the integer age feature into a categorical one. The possible values for this age-group feature include “≤ 18”, “18 – 25”, “25 – 35”, “35 – 45”, “45 – 55”, “55 – 65”, and “≥ 65”.
- In order to prevent over-fitting to rare cases of a categorical feature, we sort the frequency of all values of that feature and collapse those beyond 95% percentage into a single group called “others”. By doing this, we maintain a sufficient frequency of each value being observed in sample.
- We also Ignore the column fnlwgt, as it just adjusts for population, and has almost no predictive power.

Finally, each encoded sample has 65 features in total.

Since stochastic gradient training (SGD) implicitly rely on the features of the data having similar ranges, all features should be normalized. We consider two ways of normalization:

- Scaling each feature \(x_{ij}\) to \(x'_{ij} \in [0, 1]\) by

\[
x'_{ij} = \frac{x_{ij} - \min_j x_{ij}}{\max_j x_{ij} - \min_j x_{ij}};
\]

- Z-scoring, which makes every feature have mean 0 and standard deviation 1.
3.2 Parameters Tuning

The parameters and schemes we need to choose are

- Learning rate schedules and involved constants. There are two candidates:
  
  1. \( \lambda_e = \lambda_0/e^c \) for some constant \( c > 1 \).
  
  2. \( \lambda_e = \lambda_0/(1 + \lambda_0 ce) \) for some constant \( c \).

- Regularization tradeoff \( \mu \).

- Normalization schemes.

- Initial values of parameters.

- Stopping condition. There are two candidates:
  
  1. The Euclidean distance from the previous parameter vector to the new parameter vector is less than a threshold \( \epsilon \), or
  
  2. The mean change in parameter values is less than a threshold \( \epsilon \).

We tune these parameters mainly by orthogonal testing, i.e., by testing parameters one by one while keeping others unchanged.

3.3 Cross-Validation

We combine both training data and test data into an augmented dataset. After splitting this augmented dataset into \( k \) even groups, we hold out one of them, and use the rest \( k - 1 \) groups together to train a model, which is then tested on the held-out group. This procedure repeats for \( k \) times, with each group being held out exactly once.

The accuracy numbers obtained from all trials are averaged to give the report accuracy.

In order to take advantage of available data, the final model we deliver will be trained from the entire augmented dataset.

4 Results and Discussion

4.1 Learning Rate Schedules

Figure 1 shows the accuracy rates for both models, with respects of exponential dampening and inverse dampening. It indicates that inverse dampening has a slightly better accuracy than exponential dampening. The possible reason is that based on our choices (\( \lambda_0 = 0.1, c = 2 \) for exponential dampening and \( \lambda_0 = 0.1, c = 1.5 \) for inverse dampening), the latter decreases slower, as shown in Figure 2. More important, the former one goes too small as epoch increases (\( \lambda_e \sim 10^{-7} \) when \( e = 20 \)). Since the update is done by the multiplication of \( \lambda \) and the partial derivative, such small \( \lambda \) will lead to an unexpected result that the update stops because of tiny \( \lambda \) rather than derivative goes small. Therefore, inverse dampening is a more reasonable schedule for this problem.

We can also notice that the two models result in almost the same accuracy for each schedule, respectively.

Analysis:

The choice of two schedules is rather arbitrary, because by tuning \( c \) and \( \lambda_0 \) in each schedule, the two decay curve can be made almost identical. The choice then becomes one regarding the speed of the decay.

In optimizing the objective function using gradient descent, lambda can be interpreted as the active-ness of following the gradient. If lambda decays too fast, the optimization becomes sluggish very soon.

First consider when the objective function is a convex one. If the initial guess is far from the optimum, the optimization may virtually stop on its way to the optimum, which is not the best idea. If the initial guess is near the optimum, quickly settling down to the optimum avoids jumping around the optimum back and forth, thus improves convergence speed. Therefore we should try to give an
reasonable initial guess of the optimum, and choose a decay speed according to how confident we are about the guess.

The same phenomena happens in the non-convex optimizations. In this case, the function contains many local minima. If the closest local minima is not deep enough, being too active may results in jumping among neighbor minima. This in some ways is desirable because it allows exploration beyond a trough, and may discover a better minimum. But for a non-convex problem, we usually do
not rely on this jumpy behavior for exploration, rather we repeat the optimization several times with random initial guess and expect the algorithm to descent to a different local minimum each time. Therefore a slow decay is hardly justified in this case.

For either schedule, the $c$ and $\lambda_0$ are chosen such that the above consideration is respected.

For every run of SGD, as the sample orders are different, the parameters can be at vastly different locations in early epochs. Some locations desire slower decay while some others desire faster one. This uncertainty makes justification of a certain step decay speed impossible.

4.2 Regularization Strength

$\mu$ vs. Model Parameters:

The regularization strength $\mu$ controls the magnitude of feature weights so that they do not grow excessively large. As an example, $\beta_{57}$. The weight of feature “capital gain” shows the effect of regularization. A fact we find is that, samples with positive capital gain is more than likely to have positive label. Thus this feature is prone to excessive growing as we do the training.

Experiment shows that when $\mu = 0$, $\beta_{57}$ gets amplified without bound to -32.63, and $||\beta|| = 33.99$. While if we have $\mu = 1e-3$, $\beta_{57}$ is suppressed to -1.38; $||\beta|| = 4.618$, which is a reasonable magnitude.

$\mu$ vs. Accuracy:

Figure 3 shows the correlation between accuracy and regularization strength $\mu$. It can be shown that regularization does not improve the prediction for both models. When $0 < \mu <= 10^{-5}$, the accuracy is almost the same to the case of $\mu = 0$, i.e, without regularization. When $\mu$ goes larger, the result becomes even worse. Though regularization fails to contribute much to the prediction accuracy, it is still reasonable to use it considering its several positive properties that can alleviate the affect of over-fitting. $\mu = 10^{-5}$ would be a proper strength according to Figure 3.

Figure 3: Regularization strength $\mu$ vs. accuracy. ($f_0 = -5, e_0 = 5, \bar{\beta}_0 = \bar{0}$, normalization (2), learning rate schedule (2), stop condition (1) with $\epsilon = 0.015$)

Figure 3 also indicates that the bounded logistic regression model is not much better than the unbounded one.

4.3 Normalization Schemes

Figure 5 shows the effect of feature normalization. It can be observed that normalizing all features into range $[0, 1]$ is helpful for the prediction. The cause of this result is that SGD implicitly relies on the features of the data having similar ranges, while the original dataset has both binary features
with value \{0, 1\} and features with bigger integer value (like the capital-loss column) as well. The effect of the two normalization methods are almost the same.

In addition, we notice that the bounded model has stronger requirement of feature normalization than the unbounded model.

Figure 5: Feature normalization vs. accuracy. ($\mu = 10^{-5}, f_0 = -5, c_0 = 5, \vec{\beta}_0 = 0$, learning rate schedule (2), stop condition (1) with $\epsilon = 0.015$)

4.4 Parameter Initialization

According to our analysis, the unbounded model will have convex LCL with respect to $\beta$. Therefore, whatever the initial value of $\beta$, it should converge to the same point. Figure 6 shows that for different
initial values of $\beta_5$, unbounded model could optimize it (and other components $\beta$) to the same value, while bounded model could not. This also happens when we change the initial value of other components of $\beta$. So it verifies the statement that unbounded model is convex while bounded model is not. We also find that there is also a flat range on the curve for bounded model, which means that in this interval $\beta_5$ converges to a local maximum point. The optimized value for $\beta_5$ in unbounded model is also not exactly the same. It is because here the stop condition $\epsilon = 10^{-2}$ is relatively large. If we use a smaller $\epsilon$ the results would be more close.

Figure 7 shows the correlation between the initial value $f_0$ and the optimized value $\hat{f}$, along with resulting accuracy. The non-flat curve supports the statement that the model is not convex. We can also notice that there is a flat segment in interval $[-6, 4]$, which implies the existence of a local maximum point. Therefore, $f_0 = -5$ can be a reasonable choice for initial value of $f$. The similar pattern also appears for $c_0$ as shown in Figure 8.

\[ \mu = 10^{-4}, f_0 = -5, c_0 = 5, \vec{\beta}_0, j = 0 \text{ for } j \neq 5, \text{ normalization (2), learning rate schedule (2), stop condition (1) with } \epsilon = 0.01 \]

Figure 6: $(\mu = 10^{-4}, f_0 = -5, c_0 = 5, \vec{\beta}_0, j = 0 \text{ for } j \neq 5, \text{ normalization (2), learning rate schedule (2), stop condition (1) with } \epsilon = 0.01)$

\[ \mu = 10^{-7}, c_0 = 5, \vec{\beta}_0 = \vec{0}, \text{ normalization (1), learning rate schedule (1), stop condition (1) with } \epsilon = 10^{-5} \]

4.5 Stopping Condition

The stopping condition is a tradeoff between computing time (epochs) and training accuracy. We choose the Euclidean distance from the previous parameter vector to the current parameter vector...
Figure 8: Initial value of $c_0$ vs. optimized $\hat{c}$ (curve) and accuracy for bounded model (bars). $(\mu = 10^{-7}, f_0 = -5, \beta_0 = 0$, normalization (1), learning rate schedule (1), stop condition (1) with $\epsilon = 10^{-5}$)

Table 1: Cross-Validation results for $k = 2, 5, 10.$

<table>
<thead>
<tr>
<th>$k$</th>
<th>min</th>
<th>max</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.8365</td>
<td>0.8413</td>
<td>0.8389</td>
</tr>
<tr>
<td>5</td>
<td>0.8374</td>
<td>0.8489</td>
<td>0.8438</td>
</tr>
<tr>
<td>10</td>
<td>0.8382</td>
<td>0.8499</td>
<td>0.8442</td>
</tr>
</tbody>
</table>

as the criteria for stopping. Figure 9 compares the convergence sequence of the two models, which indicates the unbounded model converges faster than the bounded one. We choose $\epsilon = 0.15$ as the stopping threshold.

Figure 9: Convergence sequence of the two models. $(\mu = 10^{-5}, f_0 = -5, c_0 = 5, \beta_0 = 0$, normalization(2), learning rate schedule (2), stop condition (1) with $\epsilon = 0.015$)

4.6 Cross-Validation

We test $k = 2, 5, 10$ for $k$-fold cross-validation, and the results are listed in Table 1. The minimum, maximum and mean of accuracy is very similar for different $k$. Therefore, $k$ does not play an important role in cross-validation.
5 Implementation

The training with SGD sweeps through $n$ samples in each of the $e$ epochs. For each sample, the parameter update requires $O(d)$ operations due to multiplication of the feature vector and the parameter vector. Taking sparsity in feature vectors into account, we can ignore operations with any zero elements, and have the result by $O(\text{Cardinality}(d))$ operations. This gives SGD training the complexity of $O(ne\text{Cardinality}(d))$.

The bulk of training and testing code is written in Matlab, and the preprocessing (encoding of categorical features) is done by a script written in Python.

6 Conclusion

Based on the discussion above, we propose a appropriate configuration of parameters and scheme involved in the training process.

- Learning rate. Use Inverse dampening, $\lambda_e = \frac{\lambda_0}{1 + \lambda_0ce}$, where $\lambda_0 = 0.1$ and $\pi = 1.5$.
- Regularization. Use tradeoff $\mu = 10^{-5}$.
- Normalization. Use Z-scoring.
- Parameter initialization. $\beta_0 = 0, f_0 = -5, c_0 = 5$.
- Stopping condition. Use Euclidean distance, $\epsilon = 0.015$.
- Cross-Validation. Use $k = 10$.

Based on this configuration, the prediction accuracy is 83.62% for unbounded model after 34 epochs, and 83.53% for bounded model after 98 epochs.

Some key correlations we find in the experiments are

- The regularization tradeoff $\mu$ will have significant effect on epochs and accuracy if it goes large.
- The feature normalization will improve the prediction accuracy if ranges of features are much different in the original dataset.
- The unbounded model is a convex optimization problem for all parameters, and hence the global maximum can be found whatever the initial values are. However, the bounded model does not have a convex LCL for any parameter, so we need to find proper initial values.

Other factors will affect the result, but do not play an important role in prediction accuracy.