Report for assignment 3

Jingmo Bai

March 1, 2023

1 Presentation of the assignment

In general, the background and the source of the data set in this assignment are the French and English versions of *Salammbô*. We focus on the relationship between the counts of the letter "A" and the total counts of characters in the text.

In the first part of this assignment, our task is to implement a linear regression program with the gradient descent technique. We apply the program to compute the regression lines for both French and English text. Then we can predict the counts of "A" from the total counts of letters. When implementing the gradient descent algorithm, we should try both batch descent and stochastic descent methods.

In the second part, our task is to implement linear classifiers using the perceptron algorithm and logistic regression separately. We train the classifiers to classify a chapter as French or English, given a pair of numbers corresponding to the letter count and count of "A". We will try various configurations and study their influence on learning speed and accuracy. Then we should evaluate the classifiers with the leave-one-out cross-validation method.

In the third part, we apply the logistic regression classification with two popular APIs which are sklearn and Keras. They are both easily implemented and well-performed.

2 Presentation of my implementation

2.1 Linear regression

First, we need to scale both the X matrix and y vector to fit them in the range $[0, 1]$. This is because weight and intercept differ in scale. The gradient descent algorithm is very sensitive to the scale. If a feature has a scale that is orders of magnitude larger than others, it is better to normalize them. When we finish the gradient descent and get the weight vector, we restore the initial scale.

Then we implement the batch gradient descent and stochastic gradient descent(SGD). The former is to update the weights using all the samples, and the latter is to make a random partitioning of samples into different mini-batches, in every iteration we use one mini-batch to update the weights. When all samples have been used for weight updates once, that is the end of one epoch. After each epoch, we make a new random partitioning. SGD is usually quicker than batch updating, sometimes it also helps to avoid a local minimum.

Within the gradient descent algorithm, we compute the loss function and its gradient with the current weights and update the weights with the gradient times learning rate. After applying the gradient descent, we get a weights vector and plot regression lines with that. We can compute the sum of squared errors compared to the ground truth values to evaluate the regression.

2.2 Perceptron and logistic regression classification

The perceptron classifier uses a threshold function to separate two classes depending on the dot products between samples and weights. We associate two classes with positive and negative results respectively. When a sample is misclassified, the sign of the dot product is at odds with the ground truth value in the training set. To correct this, the sample vector will be added or subtracted from the vector of weights until we can find a separating hyperplane. However, it is rare to be able to completely separate them in practice, so we can stop learning when the number of misclassified examples is low.

The logistic regression classification uses a logistic curve as its transfer function

$$f(x) = \frac{1}{(1 + e^{-x})}$$

to express the probabilities of each class and yield the decision boundary. It is quite popular because the step function is not differentiable. We train this model by maximizing the log-likelihood of a certain observed classification in the training set. The opposite value of the likelihood is cross-entropy loss and we try to minimize it. We associate the cases in which the probability ¿ 0.5 and ¡ 0.5 with the classes "French" and "English" respectively. As the stop criterion, I use the norm of the gradient, if it is less than ϵ we set, it will stop the learning.

At last, we evaluate these two classifiers using the leave-one-out cross-validation method, which is a special case of the N-fold cross-validation when the number of folds N equals the number of samples in the data set. It means every time we use one sample as a validation set and all the other samples (29 in this case) as the training set. We need to train and test 30 models in total and get the ratio of the number of correct classifications to 30 as the accuracy. The cross-validation method mitigates the overfit to the training set.

3 Results and comment

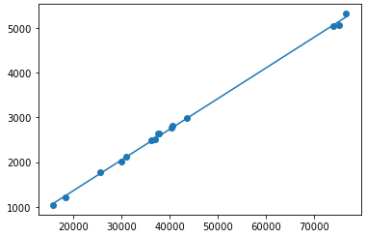

Figure 1: Batch gradient descent for French
Weights:[0.00164215, 0.06830157]

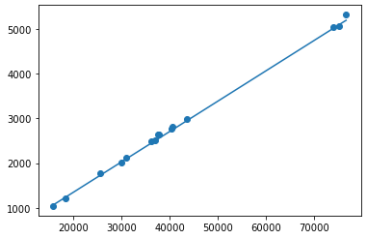

Figure 2: Stochastic gradient descent for French
Weights:[-0.00259758, 0.06761032]

As shown in figure 1 and figure 2, there is no significant difference between the regression lines for the French version obtained from batch descent and stochastic descent method. They both get the job done, and both of the regression lines are quite decent. The advantage of SGD algorithm should be it is quicker when dealing with a large data set. However, I could not feel the advantage because our data set is quite small.

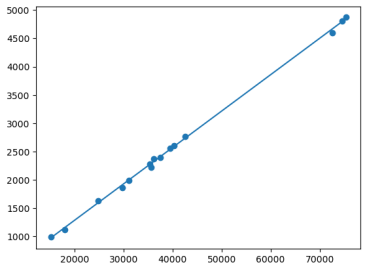

Figure 3: Batch gradient descent for English
Weights:[-0.0007389, 0.06430125]

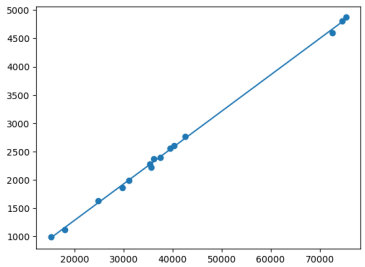

Figure 4: Stochastic gradient descent for English
Weights:[0.00339067, 0.06423801]

We can draw similar conclusions from figure 3 and figure 4. The regression lines for the English version obtained from batch descent and stochastic descent method are almost the same.

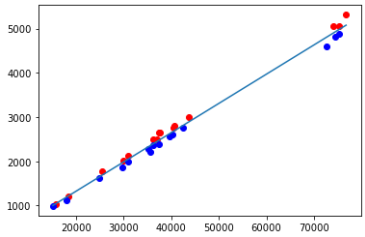

Figure 5: Perceptron classification
Weights:[0.0, -0.2634800809151015, 3.982868975903616]

As shown in figure 5, the perceptron classifier yields a decent decision boundary for classifying French and English. The cross-validation result is 29/30, the accuracy is 0.967.

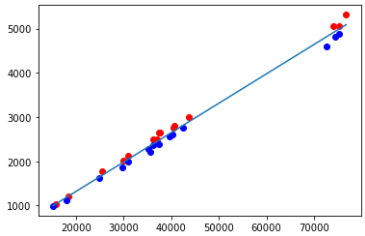

Figure 6: Logistic regression classification
Weights:[2.6703296512401513, -0.01737958612101137, 0.26160752319936276]

As shown in figure 6, the logistic regression classifier produces a similar but slightly better decision boundary than the perceptron one. The cross-validation result is 30/30, the accuracy is 1.0.

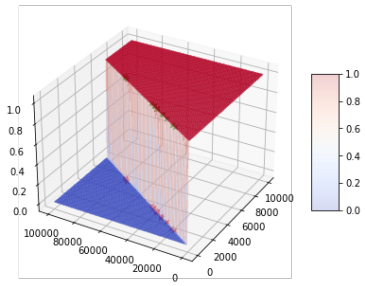

Figure 7: Logistic surface

4 Article reading

In the article *An overview of gradient descent optimization algorithms* by Sebastian Ruder, different variants of gradient descent optimization algorithms are introduced. The article provides explanations of their strengths and weaknesses, and the behaviour of different algorithms.

- Momentum

  Momentum is a method that helps accelerate SGD which has trouble around local optima by adding a fraction γ of the update vector of the past time step to the current update vector. The momentum term increases for dimensions whose gradients point in the same direction and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

- Nesterov accelerated gradient

  Nesterov accelerated gradient gives us an approximation of the next position of the parameters, a rough idea of where our parameters are going to be. This anticipatory update prevents us from going too fast and results in increased responsiveness.

- Adagrad

  Adagrad adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters. For this reason, it is suitable for dealing with sparse data. Adagrad modifies the learning rate for every parameter at every time step. One of Adagrad's main benefits is that it eliminates the need to manually tune the learning rate.

- Adadelta

  Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size. With Adadelta, we do not even need to set a default learning rate, as it has been eliminated from the update rule.

- RMSprop

  RMSprop and Adadelta both arise from the need to resolve Adagrad's radically diminishing learning rates. RMSprop is identical to the first update vector of Adadelta.

- Adam

  Adaptive Moment Estimation (Adam) also computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients.

- AdaMax

  Instead of using the second norm on previous and current gradients in Adam, the infinity one is used in AdaMax. Because it generally converges more stable value.

- Nadam

  Nadam combines Adam and Nesterov accelerated gradient, allowing us to perform a more accurate step in the gradient direction by updating the parameters with the momentum step before computing the gradient compared to Adam.