CSE 569: Project 1.6: Battle of the Momemtums

Anurag Agrawal, ID: 1207687399 and Venkata Vamsikrishna Meduri, ID:1208577223

Abstract—Two-class classification is a well-studied problem in the machine learning literature. In this work, we pick the basic binary classifier flavor of the logistic regression machine and study the properties of its momentum. We implement the negative log likelihood loss and hinge-loss methods into logistic regression to form the primary baselines. Subsequently, we learn the gradient descent learning procedure using Polyak's classical momentum [1] and Nesterov's accelerated gradient [2] [3]. We compare all these variants of logistic regression among themselves to understand the properties of the momentum and how they contribute to convergence by an iterative observation of the parameters learnt such as weights and losses. We also study the impact of the L2-regularization upon the stability of learning. We demonstrate the effectiveness of each of the optimization techniques through a case-by-case experimental analysis for the various samples drawn from the real-world datasets for Entity Resolution and the synthetic datasets that we craft manually.

Index Terms—logistic regression, Polyak, Nesterov, momentum, regularization, classification, resolution

1 Introduction

Logistic regression (LR) uses the logistic function to compute the probability with which a data point expressed as a feature vector can be classified as belonging to a particular class. As LR is predominantly used for the binary classification problem (although LR can as well be used for multi-classification), the probabilities of a feature vector belonging to either of the classes are compared and the larger probability among them determines the classification output.

For instance, let a feature vector to be classified be $\boldsymbol{D} = [d_1, d_2, \ldots, d_n]^T$. The logistic (or the sigmoid) function can be written as

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

We can notice that the feature vector $\boldsymbol{D}$ needs to be changed to a scalar value x to substitute in Equation 1. The computation of the scalar, x, can be done by a weighted combination of all the features in $\boldsymbol{D}$ which can be written as $x = w_0 + \sum_{i=1}^{n} w_i * d_i$. The weights ranging from w_0 to w_n need to be learnt from the training data in order to compute the scalar, x for each of the test data and thereby classify the test feature vectors.

One of the well-known methods to learn the weights (or in general the parameters needed) is to express the learning process as an optimization function that can be approximated to the target value iteratively. In each iteration, we can learn the parameters incrementally thereby refining them through a step-by-step process. An example to such approaches is the famous Stochastic Gradient Descent (SGD) that learns the parameters by descending gradually on the gradient of a curve which is formed out of the cost function to be minimized. An important thing to note here is that SGD expresses the optimization as a minimization though it is not necessarily the case in LR. This is because, in LR, we want to maximize the joint probability of the positively and negatively classified training vectors thereby ensuring that the classification outcome matches with the expected labeling provided in the training data. There are two approaches that formulate this maximization process as a function f_{opt} that can be minimized:

- Negative log likelihood based approach computes the joint probability (likelihood) on all the i.i.d training samples in the data, applies log on the likelihood, and minimizes its negation.
- Hinge loss based solution minimizes the error associated with mis-prediction on the training data which is also called as loss or misclassification penalty.

The basic methodology of SGD using one of the two cost functions listed above involves guessing an initial set of the $n + 1$ weights, w_0, w_1,..,w_n, and iteratively updating them such that the value of the cost function decreases in each iteration by using the updated set of weights from the previous iteration. At any given iteration, the weight corresponding to the ith feature is updated by using $w_{i+1} = w_i - (\alpha * \frac{\partial f_{opt}}{\partial w_i})$. As we can notice from the expression, the weight at the $i + 1$th iteration is computed by subtracting a finite multiple (or step) of the gradient of the function to be minimized. The step is dependent on the learning rate, α which determines the speed of convergence in the number of iterations as well as the local minimum that is obtained at convergence.

We borrow two approaches from the literature to modify the weight updation step in each iteration of SGD. These approaches are termed momentums and we compare two such solutions - Polyak's classical momentum [1] and Nesterov's momentum [2] which is the core focus of this work.

2 Optimization functions

As we described in Section 1, we use two optimization functions to be minimized in SGD, i.e., negative log likelihood and hinge loss for an LR classification machine which we detail out here.

2.1 Negative log likelihood

Assuming that all the sample vectors drawn from the training data $\boldsymbol{D}$ are independent and identically distributed (i.i.d), we can compute the joint probability of each of the

training samples belonging to the same class as determined by the class label which can be maximized.

Let us assume that there are P positively labeled points belonging to class 1 and N negatively labeled points belonging to class -1. Corresponding to this labeling, we can compute the probabilities from the logistic function in Equation 1 as $\prod_P h(x) \prod_N 1 - h(x)$ maximizing which is equivalent to minimizing the negation of it. To convert the product into a sum that is friendly to differentiate for gradient computation, we get $-\sum_P log(h(x)) - \sum_N log(1-h(x))$ which stands for f_{opt} as the minimization function to be optimized using SGD as mentioned in Section 1.

2.2 Hinge loss

The hinge loss is computed as the penalty of misclassification that we want to minimize. We find the weights which minimize the loss function that is written as follows:

$$w^* = \arg\min_w loss_{hinge} = \arg\min_w \sum_i max\{0, 1 - y_i h(x)\} \tag{2}$$

We know that h(x) from Equation 1 is dependent on $x = w_0 + \sum_{i=1}^{n} w_i * d_i$ that helps in inferring the value of w_0 to w_n by computing the gradient on $loss_{hinge}$ using partial derivatives w.r.t. each of the weight variables. The basic idea of Equation 2 is to penalize when the sign of y_i (such that $\forall y_i \in \{-1, +1\}$), the true label, is different from the estimation by the sigmoid function $h(x)$. Hence the derivative of $loss_{hinge}$ w.r.t. the ith weight parameter in LR , w_i is written as:

$$\frac{\partial loss_{hinge}}{\partial w_i} = \begin{cases} 0 & y_i h(x) \geq 1 \\ -y_i \frac{\partial h(x)}{\partial w_i} & y_i h(x) < 1 \end{cases} \tag{3}$$

The gradient w.r.t. hinge loss is taking as the summation of the partial gradients computed over all the weights, w_i, as per Equation 3.

3 Momentums

We describe the methods of momentums employed in SGD in the weight updation step. Momentum according to theoretical physics indicates the product of the mass and velocity. Though the term does not borrow the definition per se, it indicates the rate of change of the weight vector as we approach convergence in order to optimize either of the functions discussed in Section 2. As described in Section 1, $w_{i+1} = w_i - (\alpha * \frac{\partial f_{opt}}{\partial w_i})$. This is modified according to the various kinds of momentums.

3.1 Polyak's Classical Momentum

Polyak's classical momentum aims at persisting the velocity or the direction and magnitude of change to the weight vector in the existing direction. Hence, the change to the weight parameter, w_t, corresponding to the tth SGD iteration, can be expressed as weighted combination of the velocity seen so far v_t, and the gradient of f_{opt} at w_t, to obtain the change to be made, v_{t+1}.

$$v_{t+1} = \mu v_t - \alpha \frac{\partial f_{opt}}{\partial w}_{(w_t)} \tag{4}$$

$$w_{t+1} = w_t + v_{t+1} \tag{5}$$

3.2 Nesterov's Accelerated Gradient (NAG)

NAG makes a subtle change to the Polyak's momentum by computing the gradient of f_{opt} at $w_t + \mu v_t$ instead of w_t keeping the remaining part of the update equation the same as the classical momentum in Equation 4. Thus it can be formulated as follows:

$$v_{t+1} = \mu v_t - \alpha \frac{\partial f_{opt}}{\partial w}_{(w_t + \mu v_t)} \tag{6}$$

4 Progress and Timeline

We implemented LR with the two optimization functions described in Section 2. We have used an Entity Resolution dataset from Amazon-GoogleProducts [4] http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution to report our initial results. The dataset contains the product details from Amazon and Google manufactured products and the entity resolution task is to identify tuple pairs referring to the same product from both the individual datasets. The provided dataset contains 1300 matching tuple pairs whereas the negative tuples are filtered from the remaining tuple pairs by choosing non-trivial pairs which look similar enough however referring to different products summing up to 96K tuple pairs. We set the default values of the parameters, learning rate as well as the convergence threshold, to 0.001. We ran 5-fold experiments by expressing the two outcomes of the Entity Resolution as a binary classification task using LR upon these datasets to compare the performance of optimizing using negative log likelihood to that obtained using hinge loss.

TABLE 1
Logistic Regression with negative log likelihood Vs hinge loss on Amazon-GoogleProducts: 5-fold experiments

Method	Avg Precision	Avg Recall	Avg F1-Measure
-ve log lh	0.721	0.999	0.838
Hinge loss	0.484	1.0	0.652

Owing to the few matching products in the dataset, the recall is close to 100% using both the optimization approaches but the precision is the determining factor showing the superiority of negative log likelihood over the hinge loss in precision. A more detailed comparison will be provided after crafting our own manual dataset and introducing the matches and non-matches synthetically.

TABLE 2
Timeline

Task	Start Date	End Date	In-charge
Crafting datasets	11/19/16	11/23/16	Anurag Agrawal
L2-regularization	11/19/16	11/23/16	Vamsi Meduri
Polyak's method	11/23/16	11/25/16	Anurag Agrawal
Nesterov's method	11/23/16	11/25/16	Vamsi Meduri
Experiments	11/25/16	11/30/16	Anurag and Vamsi

REFERENCES

[1] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[2] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

[3] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 2013, pp. 1139–1147. [Online]. Available: http://jmlr.org/proceedings/papers/v28/sutskever13.html

[4] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.