SURVEY THE DYNAMIC SAMPLING TECHNIQUES DESCRIBED IN REFERENCES [28,73] OF BOTTOU ET AL (SECTION 5.2.1). IMPLEMENT AND TEST AT LEAST ONE OF THEM (INCLUDING THE BASIC ONE DESCRIBED IN SECTION 5.2.1)

Charles Canavaggio

ABSTRACT

The aim of this report is to survey the different dynamic sampling techniques, then implement and test two of them. The first part is brief description of the two papers. Then the second part is devote to the results of the implementation.

1 PAPER SURVEY

1.1 SAMPLE SIZE SELECTION IN OPTIMIZATION METHODS FOR MACHINE LEARNING

This paper presents a methodology for using varying sample sizes in batch-type optimization methods for large-scale machine learning problems. The first part of the paper deals with the delicate issue of dynamic sample selection in the evaluation of the function and gradient. The authors propose a criterion for increasing the sample size based on variance estimates obtained during the computation of a batch gradient. They establish an $O(1/\epsilon)$ complexity bound on the total cost of a gradient method. The second part of the paper describes a practical Newton method that uses a smaller sample to compute Hessian vector-products than to evaluate the function and the gradient, and that also employs a dynamic sampling technique. The focus of the paper shifts in the third part of the paper to L 1 -regularized problems designed to produce sparse solutions. Authors propose a Newton-like method that consists of two phases: a (minimalist) gradient projection phase that identifies zero variables, and subspace phase that applies a subsampled Hessian Newton iteration in the free variables. Numerical tests on speech recognition problems illustrate the performance of the algorithms. Richard H. Byrd (2012)

1.2 ON ADAPTATIVE SAMPLING RULES FOR STOCHASTIC RECURSIONS

The authors consider the problem of finding a zero of an unknown function, or optimizing an unknown function, with only a stochastic simulation that outputs noise-corrupted observations. A convenient paradigm to solve such problems takes a deterministic recursion, e.g., Newton-type or trust-region, and replaces function values and derivatives appearing in the recursion with their sampled counterparts. While such a paradigm is convenient, there is as yet no clear guidance on how much simulation effort should be expended as the resulting recursion evolves through the search space. In this paper, authors take the first steps towards answering this question. They propose using a fully sequential Monte Carlo sampling method to adaptively decide how much to sample at each point visited by the stochastic recursion. The termination criterion for such sampling is based on a certain relative width confidence interval constructed to ensure that the resulting iterates are consistent, and efficient in a rigorous (Monte Carlo canonical) sense. The methods presented here are adaptive in the sense that they "learn" to sample according to the algorithm trajectory. In this sense, their methods should be seen as refining recent methods in a similar context that use a predetermined sequence of sample sizes. A. Tolk & J. A. Miller (2014)

2 IMPLEMENTATION

2.1 THE DYNAMIC SAMPLE SIZE GRADIENT METHOD

The algorithm describe in this section, is the Algorithm 3.1:Dynamic Sample Gradient Algorithm describe in Richard H. Byrd (2012) This algorithm is a practical algorithm for unconstrained stochastic optimization that operates in a mini-batch framework, and that dynamically increases the size of the training sample during the course of the algorithm's progression. The decision to increase the size of the training sample is based on sample variance estimates of batch gradients. By initially selecting a small sample and gradually increasing its size, the algorithm is able to keep overall costs low, while still managing to obtain the desired level of accuracy. Richard H. Byrd (2012)

At every iteration, the method chooses a subset $S \subset 1, ..., N$ of the training set, and applies one step of an optimization algorithm to the objective function.

$$J_{S(w)=\frac{1}{|S|}\sum_{i\in S} l(f(w;x_i),y_i)}$$

The two crucial ingredients in this sampling strategy are, the condition that triggers an increase sample size and the rule for choosing the new sample.

Algorithm 3.1 : Dynamic Sample Gradient Algorithm

Choose an initial iterate w_0, an initial sample S_0, and a constant $\theta \in (0, 1)$

Set $k \leftarrow 0$

Repeat until a convergence test is satisfied:

1. Compute $d_k = -\nabla J_{S_k}(w_k)$

2. Line Search: compute steplength $\alpha_k > 0$ such that :

$J_{S_k}(w_k + \alpha_k d_k) < J_{S_k}(w_k)$

3. Define a new iterate: $w_{k+1} = w_k + \alpha_k d_k$

4. Set $k \leftarrow K + 1$

5. Choose a sample S_k such that $\mid S_k \mid = \mid S_{k-1} \mid$

6. Compute the sample variance defined in (3.6)

7. If condition (3.9) is not satisfied, augment S_k using formula (3.12)

Sample variance (3.6)

$$l(w; i) = l(f(w; x_i), y_i)$$
$$Var_{i\in S}(\nabla l(w; i)) = \frac{1}{\mid S \mid -1}\sum_{i\in S}(\nabla l(w; i) - \nabla J_S(w))^2$$

Condition(3.9)

$$\frac{\parallel Var_{i\in S}(\nabla l(w; i)) \parallel_1}{\mid S \mid} \leq \theta^2 \parallel \nabla J_S(w) \parallel_2^2$$

formula(3.12)

$$\mid S_{hat} \mid = \frac{\parallel Var_{i\in S}(\nabla l(w; i)) \parallel_1}{\theta^2 \parallel \nabla J_S(w) \parallel_2^2}$$

2.2 RESULTS

In this section, we'll discuss about the results provide by different algorithms. We're gonna compare different batch size and learning rate with the algorithm 3.1

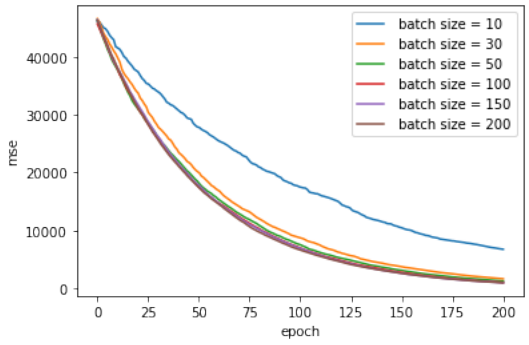

Figure 1: Comparison of different batch size.

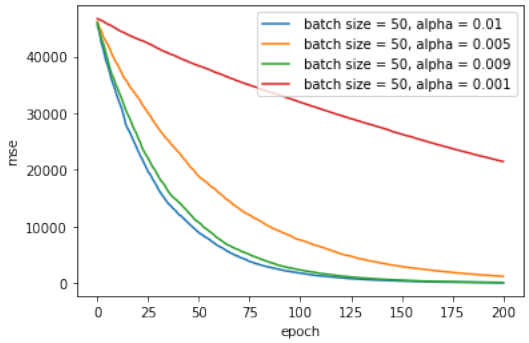

Figure 2: Comparison of different learning rate for batch size equal to 50.

When training a model using batch gradient descent, the main parameters to tune are the learning rate and the batch size. It requires some brute force search in order to find the best hyper-parameters. As we can observe on the figure 1, we obtain a better convergence with a batch size greater than 30. Moreover, we can observe from figure 2 that we obtain a good convergence with learning rate set up to 0.01 and a batch size equal to 50. Furthermore we can notice that the optimal convergence is obtain between the 125 and 150 epochs. In order to obtain a good convergence and a low mse we need to lunch many computation. Let now compare our previous results with the dynamic sample gradient algorithm (3.1). This algorithm doesn't need a lot of set-up. We start our batch from 2 then the algorithm is gonna decide whether or not it should increase the batch size. The second parameter to choose is the learning rate.

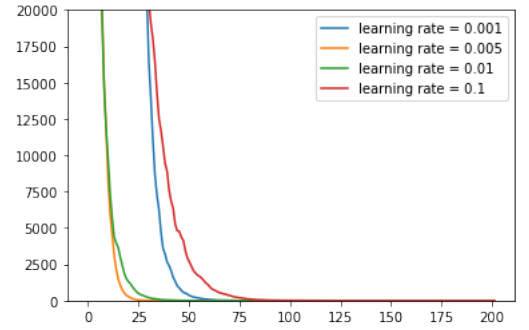

Figure 3: Comparison of different learning rate for dynamic sampling algorithm.

We can notice that, with a large stepsize of learning rate the algorihm achieve a faster convergence than previous algorithm. We can observe that the best set up give an optimal convergence around the 25 epochs and the worst set up give a optimal convergence arround the 75 and 100 epochs.

3 Algorithm 5.2.1 from Léon Bottou

In this section we'll discuss about the algorithm present in the chapter 5.2.1 from Léon Bottou

Algorithm 5.2.1

Choose an initial w_0, an initial n_0, $\chi \in [0, 1)$, an initial $\alpha > 0$

Set k $\leftarrow 0$

Repeat until a convergence test is satisfied:

1. Test

$$\varphi_k \leq \chi^2 \parallel g(w_k, \xi_k) \parallel_2^2$$

2. If condition is not satisfied, then increase the sample size

3. Otherwise define a new iterate: $w_{k+1} = w_k + \alpha_k d_k$

$w_{k+1} \leftarrow -\alpha g(w_k, \xi_k)(5.6)$

$g(w_k, \xi_k) := \frac{1}{n_k} \sum_{i \in S_k} \nabla f(w_k; \xi_{k,i})$

$\varphi_k := \frac{trace(Cov(\{\nabla f(w_k; \xi_{k,i})\}_{i \in S_k}))}{n_k}$

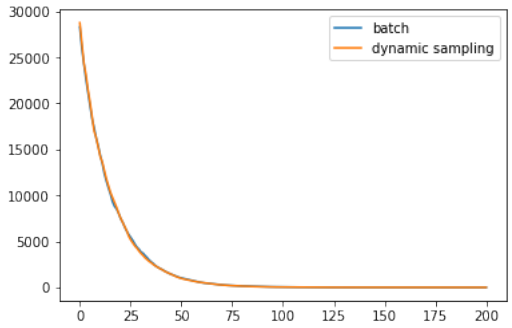

Figure 4: Comparison of dynamic sampling (5.2.1) with batch gradient

On the figure 4, we can observe the results of the dynamic sampling method with batch size initialise to 100, versus and batch gradient descent equal to 100. We can observe that we obtain a similar result, whether or not we're using dynamic sampling.

The main difficulty to use this algorithm is to find a proper value for χ if χ is too large then the sampling size never increase, from another hand if χ is to low then the sample size increase at every iteration without providing significant result.

References

I. O. Ryzhov L. Yilmaz S. Buckley A. Tolk, S. Diallo and eds. J. A. Miller. On adaptive sampling rules for stochastic recursions. pp. 3959–3970, 2014.

Jorge Nocedal Léon Bottou, Franck E. Curtis. *Optimization Methods for Large-Scale Machine Learning*.

Jorge Nocedal Yuchen Wu Richard H. Byrd, Gillian M.Chin. Sample size selection in optimization methods for machine learning. pp. 127–155, 2012.