Mini-batch Quasi-Newton optimization for Large Scale Linear Support Vector Regression

XinXie, Chao Chen andZhijian Chen

Zhejiang university, Hangzhou 310027, China
xiexin@zju.edu.cn, chenzj@vlsi.zju.edu.cn

Keywords:stochastic, optimization, linear support vector regression, quasi-Newton.

Abstract. Linear support vector regression (SVR) is a popular machine learning algorithm. However, as the amount of data increases, the learning procedure of SVR becomes time consuming. In this paper, we propose a mini-batch quasi-Newton optimization algorithm to speed up the training process of linear SVR. The main idea of the proposed optimization method is to use a small set of training data to estimate the first and second order gradient information and incorporate them into the framework of the popular limited memory BFGS quasi-Newton algorithm. Some modifications have been made to the generation of correction pairs of the BFGS algorithm in order to avoid the source of noise. Experimental results show that the proposed method outperforms some state-of-art methods in both training time and generalization ability.

Introduction

Various optimization methods have been proposed to find the optimal solution of the linear support vector regression (SVR) model, such as, SVM-Perf [1], OCAS [2] and BMRM [3]. These optimization algorithms are all in batch form and try to use information from all of the training data to update parameters. However, since the amount of training data is large, the mentioned batch training methods still take a lot of time to get the proper parameters. Thus, fast training algorithms with low computation and memory requirements are needed.

Stochastic gradient descent (SGD) method [4] offers a good way to reduce the training time. Unlike batch algorithms which use every instance from the training data, SGD tries to use a small part of the information to approximate the full training set.

Bottou [5] and LeCun [6] demonstrate that first-order SGD can easily outperform batch algorithms on large datasets but suffers from slow convergence rate. To further accelerate the convergence speed, various attempts try to use second order gradient information, for instance, Schraudolph proposed Online BFGS algorithm [7] which modifies the famous Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm [8]. And in 2013, Mokhtari [9] proposed a regularized stochastic BFGS algorithm to reduce the convergence time. However, experimental results show that second order stochastic gradient descent methods can easily yield poor Hessian approximation and result in an unstable iteration.

In this paper, a novel mini-batch quasi-Newton optimization algorithm is proposed to train the linear SVR model. The proposed optimization method could generate gradients with less noise. The updating process of the algorithm extends the limited memory BFGS algorithm and offers a quasi-Newton approximation of the inverse Hessian; as a result, a high convergence rate is achieved. The generation procedure of correction vectors in BFGS is also modified by using the first order Taylor expansion, thus could avoid the unstable iterations observed in [7,9].

Experimental results show that the proposed optimization algorithm outperforms either batch or SGD in training time, convergence rate and testing accuracy. The rest of the paper is organized as follows: Section 2 describes some main concepts in this article. Section 3 discusses the proposed optimization method. Finally Section 4 presents experimental results and Section 5 gives the conclusions and future work.

Related works

Consider a space of input-output pairs $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, machine learning algorithms try to uncover the relationship between inputs and outputs, then express it by an approximate function $f(\omega, x)$ parametrized by a weight vector ω. Loss function $loss(\hat{y}, y)$ is defined to measures the differences between the predicted value $\hat{y} = f(\omega, x)$ and the actual value y. What optimization algorithms do is to find the most suitable parameter ω which minimizes the loss function averaged on all of the training examples $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$. The procedure could be expressed as:

$$\min_{\omega} \sum_{i=1}^{N} loss(f(\omega, x_i), y_i) \quad (1)$$

Various methods are proposed to solve Eq.(1). Basically, they follow the idea of iterative searching which begin with an initial guess of the variable and generate a sequence of estimates trying to reach a sufficient good solution.

One popular method is gradient descent stated in Eq.(2) which updates the parameter ω based on the most obvious search direction ∇f. α is the step length, for simplicity, a constant is often used.

$$\omega_{k+1} = \omega_k - \alpha_k \frac{1}{N} \sum_{i=1}^{N} \nabla f(\omega_k, x_i) \quad (2)$$

Eq.(3) shows a much better approach which replaces the factor α in Eq.(2) by the inverse of Hessian matrix μ. This is called the second order gradient descent method.

$$\omega_{k+1} = \omega_k - \mu_k \frac{1}{N} \sum_{i=1}^{N} \nabla f(\omega_k, x_i) \quad (3)$$

If the true Hessian is used then derives the Newton direction which has a very fast rate of convergence and could reach the high accuracy solution in just a few iterations. However, it costs large computation efforts to calculate the Hessian matrix $\nabla^2 f$. In many applications, not exact Hessian but approximation to it can give satisfied performance, and this kind of method is called quasi-Newton algorithm.

The above first and second order gradient descent are all batch algorithms which compute the first or second order derivatives exactly over all the training data in each iteration. As the growing size of the training data, such batch algorithms become inefficient. On the contrary, stochastic or online algorithm draws much attention since it tries to update parameters based on a randomly picked training example which could highly reduce the computation efforts. The mind of stochastic method could be applied to the above two kinds of algorithms.

Stochastic gradient descent just uses one example to update the gradient and can be expressed as:

$$\omega_{k+1} = \omega_k - \alpha_k \nabla f(\omega_k, x_i) \quad (4)$$

similarly, the second order stochastic gradient descent can be defined as:

$$\omega_{k+1} = \omega_k - \alpha_k \mu_k \nabla f(\omega_k, x_i) \quad (5)$$

A comparison of the above four kinds of learning algorithms is reported by Bottu and Bousquet [5]. The results show that stochastic algorithms often yield better generalization performance than batch algorithms in large scale optimization problems. In 2009, Bordes [10] proves that a single pass of second order SGD achieves enough accuracy, thus making second-order stochastic algorithms become more attractive in large scale machine learning problems.

The proposed mini-batch Quasi-Newton optimization algorithm

In this section, the mini-batch quasi-Newton algorithm is proposed to get the optimal parameters of linear SVR. Using this optimization method, the training time could be highly reduced.

The proposed second order mini-batch gradient descent algorithm searches its optimal solution following the form as:

$$\omega_{t+1} = \omega_t - \alpha_t \mu_t \widehat{\nabla} F(\omega_t) \quad (6)$$

Where $\widehat{\nabla} F(\omega_t)$ is the average gradient of linear SVR based on a small subset of randomly picked training examples $S \in \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ and suppose the number of training examples in S is b, the average gradient can be expressed as:

$$\widehat{\nabla} F(\omega) = \sum_{i=1}^{b} \nabla f(\omega; x_i, y_i) \quad (7)$$

Where *f* is the linear L2-SVR function, it is strictly convex and a unique global minimum exists. The gradient could be calculated as:

$$\nabla f(\omega)\begin{cases}\omega + 2Cx_i(x_i^T\omega - y_i - \varepsilon), & \{i \in S|\omega^T x_i - y_i \le \epsilon\}\\ \omega - 2Cx_i(-x_i^T\omega + y_i - \varepsilon), & \{i \in S|\omega^T x_i - y_i \le -\epsilon\}\\ \omega, & others\end{cases} \quad (8)$$

Where *C* is the regularization parameters of SVR, and ε is a small constant.

Now a critical problem in Eq.(6) is how to construct μ_t. As stated before, the exact Newton direction consumes lots of computation efforts and fortunately many attempts show that just the approximation of second order information without lots of complexity could still provide high performance. LM-BFGS is that kind of quasi-Newton optimization method which stores the *k* most recent used rank one BFGS updates to maintain a low rank approximate of the inverse Hessian. Fig.1 gives the implementation of the updating process in limited memory BFGS algorithm.

REQUIRE: $\hat{\nabla}F(\omega_k)$
ENSURE: $\Gamma_k\hat{\nabla}F(\omega_k)$
$q = \hat{\nabla}F(\omega_k)$
FOR $t = k-1, k-2, k-3, ..., k-m$
$\rho_t = 1/\bar{y}_k^T\bar{s}_k$
$\alpha_t = \rho_t\bar{s}_t^T q$
$q = q - \alpha_t\bar{y}_t$
ENDFOR
$r = q$
FOR $t = k-m, k-m+1, ..., k-1$
$\rho_t = 1/\bar{y}_k^T\bar{s}_k$
$\beta = \rho_t\bar{y}_t^T r$
$r = r + \bar{s}_t(\alpha_t - \beta)$
ENDFOR
RETURN $\Gamma_k\hat{\nabla}F(\omega_k) = r$

Fig. 1 The updating process of inverse Hessian approximation in LM-BFGS

In Fig.1, there are two key coefficients $\bar{y}$ and $\bar{s}$ in the mini-batch LM-BFGS, and they could be calculated as Eq.(9). What should be mentioned is that in the original LM-BFGS,the correction vector *y* is constructed by the first order gradient. However, some reports [7,9] have shown that the stochastic extension of *y* by just choosing random examples or averaging on small subsets of training data cause much noise and yield a poor Hessian approximation. In order to remove this negative effects, the first order Taylor expansion is used to convert the two random sources into one and the resulting expression is Eq.(9):

$$\bar{s} = \bar{\omega}_I - \bar{\omega}_J,\ \bar{y} = \bar{g}_I - \bar{g}_J \approx \hat{\nabla}^2F(\bar{\omega}_I)\bar{s} \quad (9)$$

Where*I,J* are two disjoint sets which are composed of the parameters ω and the corresponding mini-batch gradient $\hat{\nabla}F(\omega)$ in each iteration.Suppose the number of samples in the set is *L*, then the set could be expressed as:

$$\{(\omega_i, \hat{\nabla}\mathrm{F}(\omega_i))|\mathrm{i} \in \mathrm{I}\} \quad (10)$$

Then $\bar{\omega}_I$ can be defined as:

$$\bar{\omega}_I = \frac{1}{L}\sum_{i\in I}\omega_i \quad (11)$$

In order to calculate Eq.(9), the mini-batch form of the second order gradient should be clarified. Let S_H be a small subset of random picked training examples and b_H represents the number of samples in S_H, then the mini batch form of the second order gradient could be defined as:

$$\hat{\nabla}^2F(\omega) = \frac{1}{b_H}\sum_{i\in S_H}\nabla^2 f(\omega) \quad (12)$$

For linear L2-SVR, Mangasarian[11] has shown that L2-SVR is almost twice differentiable and the generalized Hessian matrix can be defined as:

$$\nabla^2 f(\omega)\begin{cases}1 + 2Cx_i^T x_i, & if\ \{i \in S_H||\omega^T x_i - y_i| > \varepsilon\}\\ 1, & if\ \{i \in S_H||\omega^T x_i - y_i| \le \varepsilon\}\end{cases} \quad (13)$$

Now the mini-batch form of parameters needed to do the LM-BFGS updating are all prepared and the pseudocode of the whole algorithm is listed in Fig.2.

REQUIRE Initial parameters ω_0, M, L, N, K
ENSURE final weight ω_k
  $t = 0$
  $\bar{\omega}_J = \omega_0, \bar{\omega}_I = 0$
  FOR $k = 1, 2, ..., k-1$
    calculate $\hat{\nabla} F(\omega_k)$
    $\bar{\omega}_I = \bar{\omega}_I + \omega_k$
    IF $k < 2L$
      $\omega_{k+1} = \omega_k - \alpha_k \hat{\nabla} F(\omega_k)$
    ELSE
      $\omega_{k+1} = \omega k - \alpha_k \Gamma_t \hat{\nabla} F(\omega_k)$
    ENDIF
    IF $mod(k, L) = 0$
      $t = t + 1$
      $\bar{\omega}_I = \bar{\omega}_I / L$
      $\bar{s}_t = \bar{\omega}_I - \bar{\omega}_J$
      $\bar{y}_t = \hat{\nabla}^2 F(\bar{\omega}_I)(\bar{\omega}_I - \bar{\omega}_J)$
      $\bar{\omega}_J = \bar{\omega}_I, \quad \bar{\omega}_I = 0$
    ENDIF
  ENDFOR
  RETURN ω_k

Fig. 2 The Mini-batch quasi-Newton Optimization Algorithm

Experiments

In this section, the proposed optimization method is evaluated on *a9a* dataset. Dataset*a9a* is compiled from the UCI *adult* dataset, which contains 32561 training samples and 16281 testing samples. Since the training and testing examples in *a9a* is relatively large, we use it to test the performance of our proposed mini-batch quasi-Newton algorithm for large scale problems.

The efficiency of the proposed optimization algorithm is first tested and then compared with widely used algorithms including gradient descent method [12], Newton method [13] and SGD [6].

A ten-fold cross validation is used to choose the suitable regularization parameter *C*in Eq.(8) and Eq.(13). The value of *C* is changing from $2^{-3}$to 2^{2}, and the accuracy is listed in Table 1. As a result, the value of *C* is chosen to be 2^{-1}.

Table 1 the test accuracy according to different regularization parameter *C*

C	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}
Accuracy (%)	80.82	80.85	84.21	81.79	65.32	59.43

The proposed method is first compared with batch algorithms [12,13]. The two pictures in Fig.3 reveal how the value of the cost function changes with the number of passes and the training time. Fig.3 reveals that for the reason of quasi-Newton factor, the convergence rate by iteration of the proposed method is more rapidly than first-order gradient descent. The batch Newton method decreases most rapidly and can reach the stable solution in just a few iterations. However, Fig.3 shows that the calculation complexity of batch algorithm is greater than the proposed mini-batch method. Besides, since the batch algorithms take advantage of the full training set, the curves is more stable than mini-batch approach. Fig.4 reveals the test error (%) decreases as the passes and training time. Similar results could be generated that the generalization ability of proposed method is comparable with second-order batch algorithm but suffer from much less training time.

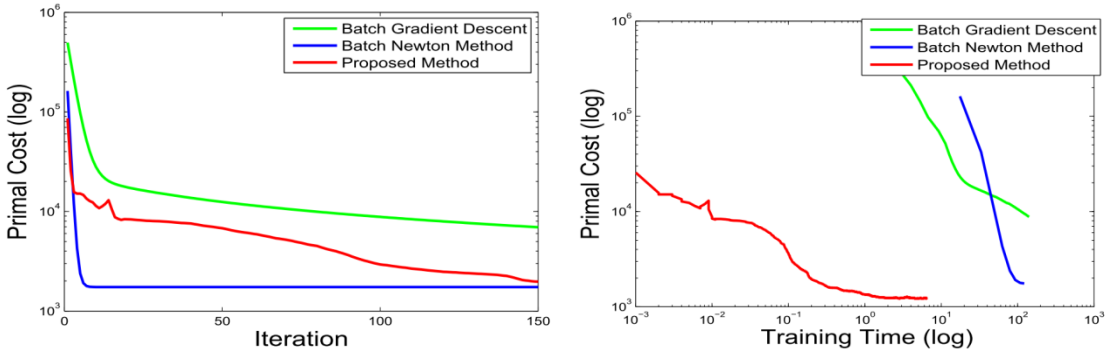

Fig. 3 The function value according to the number of iteration and training duration on the *a9a* dataset between batch and proposed algorithms

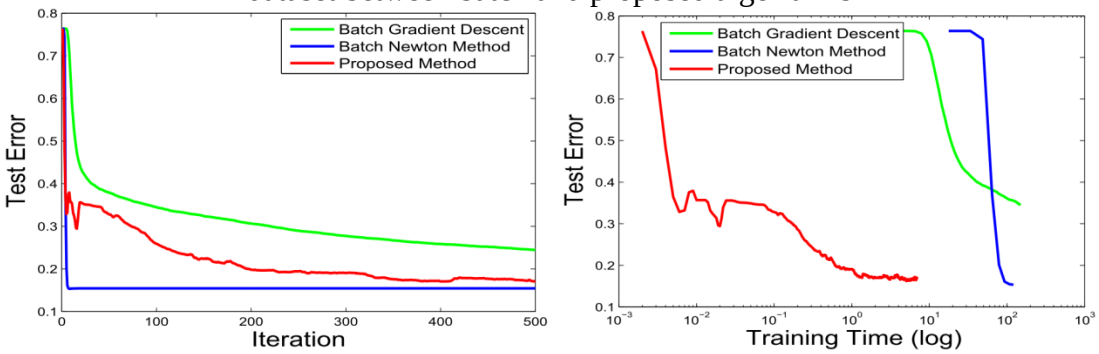

Fig. 4 The test error (%) according to the number of iteration and training duration on the *a9a* dataset between batch and proposed algorithms

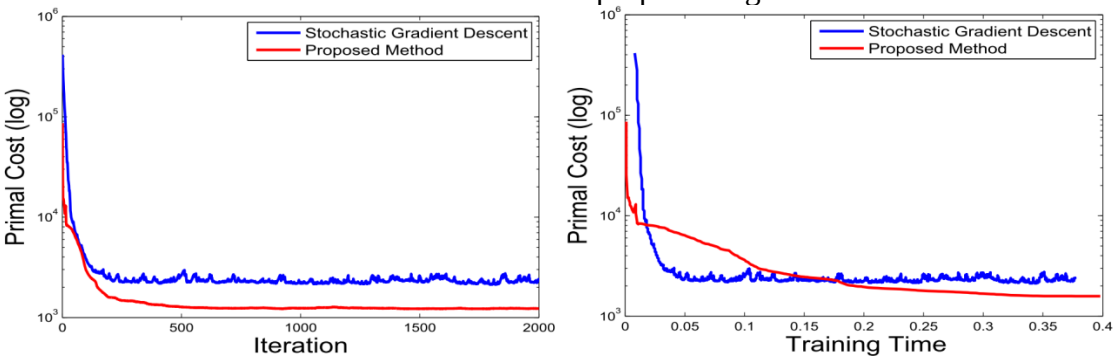

Fig. 5 The function value according to the number of iteration and training duration on the *a9a* dataset between SGD and proposed algorithms

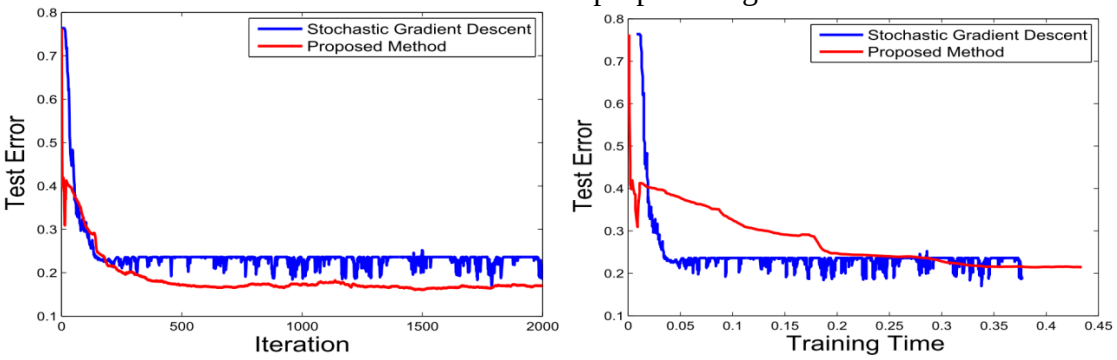

Fig. 6 The test error (%) according to the number of iteration and training duration on the *a9a* dataset between SGD and proposed algorithms

The proposed method is also compared with the most attracting state-of-art algorithm SGD [6]. From Fig.5 and Fig.6, either the convergence rate or the computation efforts are almost the same. However, both Fig.5 and Fig.6 reveal one disadvantage of SGD which is that since only one random picked example is used to update the gradient information and lacks the second order information, it oscillates after a few iterations and could hardly reach a lower value. The curves show that the proposed method has a better generalization performance than SGD. The parameters in the proposed algorithm are set as: b=10,b_H=10,M=10, L=10.

Conclusions and future works

In this paper, a quasi-Newton optimization algorithm has been proposed. Unlike the traditional batch method, a small part of training examples are used to estimate the first and second order gradient. Through this way a large computation effort has been saved. Since the second-order quasi-Newton approximation is used, the proposed method also gets better generalization performance than stochastic gradient descent.

In future works, we want to find more efficient optimization algorithms for large scale applications and try to extend this method to classification algorithms such as structural SVM or local linear SVM to solve more complex problems.

References

[1]T. Joachims. Training linear SVMs in linear time, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM (2006):217-226.

[2] V. Franc, S. Sonnenburg. Optimized cutting plane algorithm for support vector machines.Icml, (2008):320-327.

[3] C.H. Teo, S. Vishwanathan, A. Smola, et al, Bundle methods for regularized risk minimization, Journal of Machine Learning Research, 11 (2010): 311-365.

[4] K. Stefan, P.W. Josien, S. Marius, et al, Adaptive stochastic gradient descent optimisation for image registration. International Journal of Computer Vision, 81 (2009): 227-239.

[5] L. Bottou, O. Bousquet, The tradeoffs of large scale learning. Advances in Neural Information Processing Systems, 20(2008): 161-168.

[6] L. Bottou, Y. LeCun, Online learning for very large datasets, Applied Stochastic Models in Business and Industry, 21(2)(2005): 137-151.

[7] N.N. Schraudolph, Y. Jin, S. Gunter, A stochastic quasi-Newton method for online convex optimization, Proceeding of 11th International Conference on Artificial Intelligience and Statistics, (2007).

[8] J. Nocedal, S.J. Wright, Numerical optimization, second edition. Springer Series in Operations Research. Springer, (2006): 136-141.

[9] A. Mokhtari, A. Ribeiro, Regularized stochastic BFGS algorithm.arXiv:1401.7625 (2014).

[10] A. Bordes, L. Bottou, P. Gallinari. SGD-QN: careful quasi-Newton stochastic gradient descent. Journal of Machine Learning Research, 10(2009): 1737-1754.

[11]O.L. Mangasarian, A finite Newton method for classification. Optimization methods and Software, 17(5)(2002): 913-929.

[12] G.X. Yuan, K.W. Chang, C.J. Hsieh, C.J. Lin, A comparison of optimization methods and software for large-scale L1-regularized linear classification. Journal of Machine Learning Research, 11(2)(2010): 3183-3234.

[13] C.J. Lin, R.C. Weng, S.S. Keerthi, Trust region Newton method for large-scale logistic regression. Journal of Machine Learning Research, 9(2)(2008): 627-650.