Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Mini-batch cutting plane method for regularized risk minimization*

Meng-long LU†, Lin-bo QIAO, Da-wei FENG†‡, Dong-sheng LI, Xi-cheng LU

Science and Technology on Parallel and Distributed Laboratory, National University of Defense Technology, Changsha 410073, China

†E-mail: lumenglong2018@163.com; davyfeng.c@gmail.com

Received Sept. 25, 2018; Revision accepted June 23, 2019; Crosschecked Oct. 10, 2019

Abstract: Although concern has been recently expressed with regard to the solution to the non-convex problem, convex optimization is still important in machine learning, especially when the situation requires an interpretable model. Solution to the convex problem is a global minimum, and the final model can be explained mathematically. Typically, the convex problem is re-casted as a regularized risk minimization problem to prevent overfitting. The cutting plane method (CPM) is one of the best solvers for the convex problem, irrespective of whether the objective function is differentiable or not. However, CPM and its variants fail to adequately address large-scale data-intensive cases because these algorithms access the entire dataset in each iteration, which substantially increases the computational burden and memory cost. To alleviate this problem, we propose a novel algorithm named the mini-batch cutting plane method (MBCPM), which iterates with estimated cutting planes calculated on a small batch of sampled data and is capable of handling large-scale problems. Furthermore, the proposed MBCPM adopts a "sink" operation that detects and adjusts noisy estimations to guarantee convergence. Numerical experiments on extensive real-world datasets demonstrate the effectiveness of MBCPM, which is superior to the bundle methods for regularized risk minimization as well as popular stochastic gradient descent methods in terms of convergence speed.

Key words: Machine learning; Optimization methods; Gradient methods; Cutting plane method
https://doi.org/10.1631/FITEE.1800596 **CLC number:** TP391

1 Introduction

The optimization problems are central to the application of machine learning, including convex and non-convex optimization problems. In the former, the local minimum is also a global one, so the identification of a global optimum solution is feasible. However, the circumstance is not the same in the non-convex optimization problem and finding a global optimum is impractical. Therefore, the solution to a non-convex optimization problem is sometimes perplexing. For example, in a non-convex optimization problem based machine learning application, Mou et al. (2016) have observed that a mathematically equivalent modification can result in different performances. This occurs because the mathematically equivalent modification is valid only for the condition where the global optimal value can be obtained before and after modification. However, the non-convex optimization problem does not provide this guarantee. Therefore, considering interpretability, a machine learning application is preferred to be formulated as a convex optimization problem, especially when the training and inference are performed on structured data.

Typically, the convex optimization problem in

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2018YFB0204300) and the National Natural Science Foundation of China (Nos. 61872376 and 61806216)

ORCID: Meng-long LU, http://orcid.org/0000-0003-0011-0368; Da-wei FENG, http://orcid.org/0000-0002-7587-8905

machine learning is formulated as a regularized risk minimization problem:

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}) = R(\boldsymbol{w}) + \lambda \Omega(\boldsymbol{w}), \tag{1}$$

where $R(\boldsymbol{w}) = \dfrac{1}{n}\sum_{i=0}^{n} l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w})$ is the original convex empirical risk function over a finite number of instances and $\Omega(\boldsymbol{w})$ is the introduced convex regularized term to prevent overfitting.

If J in Eq. (1) is a differentiable convex function, smooth optimization techniques, such as Broyden-Fletcher-Goldfarb-Shanno (BFGS) or its enhanced limited-memory BFGS, could be employed to efficiently solve this problem. However, if J is non-differentiable, extending these algorithms to the optimization process is not straightforward (Teo et al., 2010). Proximal gradient methods have been proposed to address non-differentiable problems when the proximal mapping could be easily obtained. However, proximal algorithms are invalid when proximal mapping is difficult to obtain.

There exists another kind of efficient convex optimizer called the cutting plane method (CPM) (Kelley, 1960) that can be used to solve both differentiable and non-differentiable problems. If the objective function J is differentiable, CPM uses the gradient to proceed with the optimization; otherwise, CPM employs the subgradient (a generalization of the gradient) instead. The bundle method (Hiriart-Urruty and Lemaréchal, 1993; Lemaréchal et al., 1995), as well as the bundle method for regularized risk minimization (BMRM) (Teo et al., 2010), is an improved version of CPM. However, they all require access to the entire dataset for each iteration, which renders them infeasible for efficiently handling large datasets.

To address this problem, we propose a novel CPM method called the mini-batch cutting plane method (MBCPM) to conduct training on a mini-batch of data, similar to the training strategy used in stochastic gradient descent (SGD) methods. Given that the computational complexity of calculating corresponding parameters (e.g., gradients) on a mini-batch of data is much lower than that on the entire dataset, MBCPM achieves a much faster iteration speed compared to traditional CPM methods while maintaining their fast convergence property. In addition, experimental results demonstrate that the objective function values, as well as other performance metrics (e.g., error rate), converge faster in the case of MBCPM compared to the traditional CPM and the popular SGD methods. Specifically, there are two essential aspects of MBCPM:

1. MBCPM calculates the estimated cutting plane on a mini-batch of data instead of calculating the accurate cutting plane on the entire training set. Thus, a much faster iteration speed is achieved.

2. The noise introduced in the process of estimating cutting planes on a sampled dataset is eliminated by adopting a “sink” operation that adjusts unacceptable cutting planes.

2 Related work

2.1 Cutting plane method

CPM is an efficient convex optimization method that is applicable to both differentiable and non-differentiable problems. In the latter, gradients are not available. As such, this is a significant challenge for conventional gradient based optimization methods. In the case of CPM methods, if the objective function is not differentiable, namely, the gradients are not available, subgradients are chosen as a substitute for gradients to generate cutting planes and to further approximate the objective function using these cutting planes. With respect to the subgradient, considering the selection of a point $\boldsymbol{w}'$ from the domain J, the corresponding subgradient $\boldsymbol{s}'$ is the normal vector of any tangential supporting hyperplane of J at $\boldsymbol{w}'$. Formally, $\boldsymbol{s}'$ is called a subgradient of J at $\boldsymbol{w}'$ if and only if for all $\boldsymbol{w}$, the following inequality holds true:

$$J(\boldsymbol{w}) \geq J(\boldsymbol{w}') + \langle \boldsymbol{w} - \boldsymbol{w}', \boldsymbol{s}' \rangle. \tag{2}$$

If there is a subgradient at $\boldsymbol{w}'$, then J is said to be subdifferentiable at $\boldsymbol{w}'$. Convex functions are subdifferentiable everywhere in their domain (Hiriart-Urruty and Lemaréchal, 1993). Therefore, CPM can address all convex problems, including those unavailable gradients.

$J(\boldsymbol{w}') + \langle \boldsymbol{w} - \boldsymbol{w}', \boldsymbol{s}' \rangle$ in inequality (2) is regarded as a cutting plane in CPM, which bounds the objective function from below. Moreover, given a sequence of iterations $\{1,\ 2,\ \ldots,\ t\}$ and their corresponding locations $\{\boldsymbol{w}_1,\ \boldsymbol{w}_2,\ \ldots,\ \boldsymbol{w}_t\}$, CPM constructs a piecewise linear lower bound of the original convex function J through the conjunction of these cutting planes, which is defined as

$$J(\boldsymbol{w}) \geq J_{t+1}^{\mathrm{CP}}(\boldsymbol{w}) = \max_{1\leq i\leq t}\{J(\boldsymbol{w}_i)+\langle \boldsymbol{w}-\boldsymbol{w}_i, \boldsymbol{s}_i\rangle\}. \quad (3)$$

In the iteration $t+1$, CPM expands the location set $\{\boldsymbol{w}_i\}_{i=1}^{t}$ as

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} J_{t+1}^{\mathrm{CP}}(\boldsymbol{w}). \quad (4)$$

This procedure refines the piecewise linear lower bound J_t^{CP} iteratively. As shown in Fig. 1, with an increase of the number of cutting planes, the lower bound becomes increasingly tighter, which causes the minimum of the piecewise linear function $J_t(\boldsymbol{w})$ to gradually approach the optimum of the objective function $J(\boldsymbol{w})$.

However, if the output $\boldsymbol{w}_{t+1}$ at iteration $t+1$ moves too far away from the previous $\boldsymbol{w}_t$, naive CPM may be encumbered by a slow convergence speed, which is also known as the "zig-zag" problem. The bundle method is proposed to address this issue by augmenting the piecewise linear lower bound J_{t+1}^{CP} with a proximity control function to stabilize the optimization procedure.

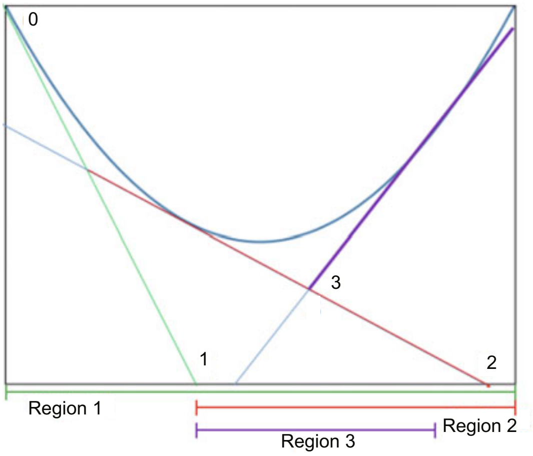

Fig. 1 Illustration of how CPM works

CPM starts at point 0, and then it finds the minimum of the piecewise linear function located at point 1. A new cutting plane at point 1 is added to the piecewise linear function and a new minimum is obtained at point 2. Starting from point 2, CPM performs a similar procedure to find point 3. The cutting plane serves as a constraint for the model's feasible region. For example, after adding the red cutting plane into the piecewise linear lower bound, the model's feasible region is transferred from region 1 to 2. This is because considering the points to the left of point 1, the reflected values on the linear lower bound are all larger than the value at point 1. Thus, it is impossible for them to be the minimum of the current lower bound

For example, in a classical variant of the bundle method (Kiwiel, 1990), Eq. (4) is re-casted as

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} \left\{ J_{t+1}^{\mathrm{CP}}(\boldsymbol{w}) + \frac{\zeta_t}{2}||\boldsymbol{w} - \hat{\boldsymbol{w}}_t||^2 \right\}, \quad (5)$$

where $\hat{\boldsymbol{w}}_t$ is the previous prox-center of the decision vector $\boldsymbol{w}$ and ζ_t is used to balance the transfer of Eq. (4) and the step size. The reason that Eq. (5) can be used to alleviate the "zig-zag" problem can be explained by the proximity function $\frac{\zeta_t}{2}||\boldsymbol{w} - \hat{\boldsymbol{w}}_t||^2$. If the minimum of J_{t+1}^{CP} (the solution to Eq. (4) in naive CPM) is far from the previous prox-center $\hat{\boldsymbol{w}}_t$, the risk of the proximity function $\frac{\zeta_t}{2}||\boldsymbol{w} - \hat{\boldsymbol{w}}_t||^2$ will increase. Consequently, the solution to Eq. (5) will be away from the minimum of J_{t+1}^{CP} to prevent an overly large step. Thus, this design not only inherits the transferring strategy in CPM but also can avoid the "zig-zag" problem because Eq. (5) can be used to find a stabilized solution.

Even though bundle methods are shown to be effective in preventing overly large steps in the literature (Kiwiel, 1983; Schramm and Zowe, 1992; Lemaréchal et al., 1995; Belloni, 2005), it is also well known that the stabilized parameters in bundle methods require careful tuning in practice (Teo et al., 2010). BMRM takes a new approach to avoid introducing extra parameters while maintaining the ability to prevent overly large steps. BMRM follows the framework of bundle methods that use a proximity function to constrain the step size, but the design of the proximity function distinguishes it. BMRM decouples the objective function as regularization and the risk function. The piecewise linear lower bound is constructed on the risk function. In the case of regularization, it is used as a proximity function like the one in the bundle method. This proximity function is a special case of Eq. (5), where the previous prox-center is always the origin (Teo et al., 2010).

In BMRM, inequality (3) and Eq. (4) are reformulated as inequality (6) and Eq. (7), respectively.

$$J(\boldsymbol{w}) \geq J_t^{\mathrm{CP}}(\boldsymbol{w}) = R_t^{\mathrm{CP}}(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}), \quad (6)$$

$$\boldsymbol{w}_t = \arg\min_{\boldsymbol{w}} \left\{ R_t^{\mathrm{CP}}(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \right\}, \quad (7)$$

where $R_t^{\mathrm{CP}}{=}\max_{1\leq i\leq t}\Big\{R(\boldsymbol{w}_{i-1}){+}\big\langle \boldsymbol{w}{-}\boldsymbol{w}_{i-1}, \partial_{\boldsymbol{w}} R(\boldsymbol{w}_{i-1})\big\rangle\Big\}$. In inequality (6), the difference lies in their approximation function $J_t^{\mathrm{CP}}(\boldsymbol{w})$. BMRM's approximation function includes a linear lower bound (R_t^{CP}) of risk functions R and a regularized term $\lambda\Omega(\boldsymbol{w})$, while the CPM's approximation function is only a piecewise linear lower bound of the objective $J(\boldsymbol{w})$. Eq. (7) is used to expand the location set $\{\boldsymbol{w}_i\}_{i=1}^{t}$.

We define $\boldsymbol{a}_i \in \partial_{\boldsymbol{w}} R(\boldsymbol{w}_{i-1})$ and $\boldsymbol{b}_i = R(\boldsymbol{w}_{i-1}) - \langle \boldsymbol{w}_{i-1}, \boldsymbol{a}_i \rangle$. $\boldsymbol{A}^{\mathrm{T}}$ denotes a matrix $[\boldsymbol{a}_1,\ \boldsymbol{a}_2,\ \ldots,\ \boldsymbol{a}_t]^{\mathrm{T}}$ and $\boldsymbol{b}$ denotes the offset vector $[\boldsymbol{b}_1,\ \boldsymbol{b}_2,\ \ldots,\ \boldsymbol{b}_t]$.

The dual problem of Eq. (7) is formally expressed as

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha}\in\mathbb{R}^t}{\arg\max}(-\lambda\Omega^*(-\lambda^{-1}\boldsymbol{A}\boldsymbol{\alpha}) + \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{b}), \boldsymbol{\alpha} \geq 0, \quad (8)$$

where $\boldsymbol{\alpha}$ is a non-negative Lagrange multiplier and satisfies $||\boldsymbol{\alpha}||_1 = 1$. The relationship between $\boldsymbol{w}_t$ and $\boldsymbol{\alpha}_t$ is the dual connection $\boldsymbol{w}_t = \partial\Omega^*(-\lambda^{-1}\boldsymbol{A}\boldsymbol{\alpha}_t)$. As noted in Teo et al. (2010), Eq. (8) is easy to solve given its simple constraint on $\boldsymbol{\alpha}$.

In general, CPM methods are useful algorithms for the advantage of being able to fully leverage previously calculated gradient information. As shown in Fig. 1, the previous cutting planes can be employed as constraints of the model's feasible region, and thus the continuous addition of cutting planes into the linear lower bound will cause the feasible region to become increasingly smaller, which supports the argument that the previous information is useful in the process of finding the optimal solution.

In terms of the development of CPM methods, most of the effort is primarily focused on the alleviation of the "zig-zag" problem. However, with the background of this big data era, we argue that their training strategy limits their performance because accessing the entire dataset to perform gradient calculation (inequality (2)) reduces the iteration speed. This problem is also called the big data problem, which is a major challenge for traditional CPM methods.

2.2 Stochastic gradient descent method

There are many remarkable training strategies available to alleviate the big data problem. A well-known approach is mini-batch training whereby in each iteration, a batch of data is sampled to update the model. Among the mini-batch training based methods, SGD is a representative technique. The SGD method is derived from the gradient descent method, which updates the model parameters along with the direction of the current gradient. The updating process in the gradient descent method is formulated as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \boldsymbol{g}_t, \quad (9)$$

where η is the learning rate to control the step size of the descent and $\boldsymbol{g}_t = \nabla_{\boldsymbol{w}} J(\boldsymbol{w}_t; \xi_t)$ is the current gradient of the objective function. However, with the growth of the data size, the computation of the gradient $\boldsymbol{g}_t$ becomes very time-consuming and the process is untenable. Thus, SGD calculates an estimated gradient on a batch of sampled data to reduce the computation complexity. Then, it updates the model parameters with a similar formula to Eq (9):

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \tilde{\boldsymbol{g}}_t, \quad (10)$$

where $\tilde{\boldsymbol{g}}_t = \nabla_{\boldsymbol{w}} E\big(J(\boldsymbol{w}_t; \xi_t)\big)$ and $E\big(J(\boldsymbol{w}_t; \xi_t)\big)$ are estimates of the gradient and cost based on the sampled data ξ_t, respectively.

SGD is very popular in the machine learning community because of its simplicity and efficiency. However, the main problem is that it uses only the current estimated gradient information to update the model, so the previously calculated estimated gradients are discarded, resulting in the waste of previous information.

To address the previously mentioned problem, momentum SGD, adaptive gradient algorithm (AdaGrad), and adaptive moment estimation (Adam) are improvements of the naive SGD method.

Momentum SGD (Qian, 1999) is based on the concept of integrating the previous gradient information, the weighted sum of the current gradient, and the previous momentum, expressed as

$$\boldsymbol{\nu}_t = \tilde{\boldsymbol{g}}_t + \zeta \cdot \boldsymbol{\nu}_{t-1}, \quad (11)$$

where $\boldsymbol{\nu}$ represents the momentum and $\tilde{\boldsymbol{g}}$ is the estimated gradient on a batch of sampled data. Following the direction of the momentum, momentum SGD updates the model as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\boldsymbol{\nu}_t. \quad (12)$$

Different from momentum SGD, AdaGrad (Duchi et al., 2011) uses the previous gradient information to generate an adaptive learning rate. This rate is calculated as

$$\boldsymbol{G}_t = \boldsymbol{G}_{t-1} + \boldsymbol{g}_t \odot \boldsymbol{g}_t. \quad (13)$$

The formula to update each model parameter is given as

$$\boldsymbol{w}_{t+1,i} = \boldsymbol{w}_{t,i} - \frac{\eta_0}{\sqrt{\boldsymbol{G}_{t,i} + \epsilon}} \boldsymbol{g}_{t,i}, \tag{14}$$

where ϵ is a residual constant to prevent an invalid division. Based on Eq. (14), AdaGrad causes the pace of the frequently updated parameters to be smaller and that of the infrequently updated parameters to be larger.

With respect to Adam (Kingma and Ba, 2014), it is a further improvement of AdaGrad and momentum SGD and the adaptive vector $\boldsymbol{G}_t$ is given as

$$\boldsymbol{G}_t = \frac{1}{1-\beta_1^t}\left(\beta_1 \boldsymbol{G}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \odot \boldsymbol{g}_t^{\mathrm{T}}\right), \tag{15}$$

which is similar to Eq. (13) in AdaGrad. It maintains an exponentially decaying momentum with the past gradients as

$$\boldsymbol{\nu}_t = \frac{1}{1-\beta_2^t}\left(\beta_2 \boldsymbol{\nu}_{t-1} + (1-\beta_2)\boldsymbol{g}_t\right). \tag{16}$$

For each element in the model, Adam updates it with

$$\boldsymbol{w}_{t+1,i} = \boldsymbol{w}_{t,i} - \frac{\eta_0}{\sqrt{\boldsymbol{G}_{t,i} + \epsilon}} \boldsymbol{\nu}_{t,i}. \tag{17}$$

Although these SGD methods can alleviate the problem of discarding the previous gradient information, this is limited by the training settings. For example, the adaptive learning rate works well mainly in sparse learning and the momentum is designed to prevent the issue of a navigation ravine in a nonconvex objective function. However, for a general convex optimization, the adaptive learning rate and the momentum may contribute little to leverage the previous gradient information, at least not in the same manner as CPM in which the previous gradient is employed to constrain the model's feasible region.

Even though SGD is hindered by the aforementioned limitation, we nevertheless highlight their training strategy with respect to performing a calculation on a small sampled sub-dataset, because it allows them to iterate faster than the naive gradient descent method. Therefore, in this study, we propose a new algorithm that integrates the advantages of the SGD method for fast iteration, and the advantage of the CPM method with respect to leveraging the previously calculated gradients. The new algorithm is called MBCPM, and the details will be introduced in the next section.

3 Algorithm design

Generally, CPM and its improved versions require only a small number of iterations to achieve an acceptable solution. However, as previously indicated, if the amount of data grows considerably large, the computation cost of calculating the (sub)gradient in each iteration will become untenable. In the big data era, all traditional optimizers face the same challenge if they need to perform calculations of (sub)gradients on the entire dataset. A notable strategy to address this problem is the one used in SGD methods, where accurate (sub)gradients are replaced with the estimated (sub)gradients and calculated on a small sampled sub-dataset to reduce the computational complexity in each iteration. Inspired by this approach, MBCPM adopts this strategy to accelerate the iteration and thus to improve the performance of CPM methods. However, SGD is more tolerant of the noise of the estimated (sub)gradient, which is not the same in CPM. Even for noise in inexact (sub)gradients, SGD methods still have the capability to converge and iterate, but the situation is not the same as in the CPM methods. To prevent such effects of inexact (sub)gradient, MBCPM adopts a novel operation called "sink" to adjust the piecewise linear lower bound.

In each iteration, MBCPM randomly selects instances from the entire training set and generates an estimated cutting plane on this sampled sub-dataset. This estimated cutting plane is not an accurate plane calculated on the entire dataset, so it is always accompanied by noise. Consequently, MBCPM needs to evaluate whether these estimated cutting planes affect the correctness of the algorithm. For those estimated cutting planes that heavily affect the correctness of the algorithm, they are defined as the noisy estimations. If noisy estimations are detected, MBCPM will adopt a "sink" operation to address them; otherwise, it will continue its iteration process by solving Eq. (21), which will be given in Section 3.1.

3.1 Estimation of the cutting plane

As shown in Section 2.1, cutting planes play a key role in CPM methods. Instead of finding the accurate cutting plane at $\boldsymbol{w}_t$ on the entire dataset, MBCPM calculates an estimate on a sampled dataset. For objective $\widetilde{J}(\boldsymbol{w})$ on the sampled dataset, it is calculated using Eq. (19). For the estimated cutting plane $(\tilde{\boldsymbol{a}}_t \cdot \boldsymbol{w} + \tilde{\boldsymbol{b}}_t)$, it is formally given by $\tilde{\boldsymbol{a}}_t \in \partial_{\boldsymbol{w}} \cdot \widetilde{R}(\boldsymbol{w}_{t-1})$ and $\tilde{\boldsymbol{b}}_t = \widetilde{R}(\boldsymbol{w}_{t-1}) - \langle \boldsymbol{w}_{t-1}, \tilde{\boldsymbol{a}}_t \rangle$. Then, the corresponding problem of Eq. (7) and its dual formula Eq. (8) can be rewritten as Eqs. (21) and (22), where $\widetilde{\boldsymbol{A}} = [\tilde{\boldsymbol{a}}_1,\ \tilde{\boldsymbol{a}}_2,\ \ldots,\ \tilde{\boldsymbol{a}}_t]$ is the corresponding (sub)gradient matrix and $\tilde{\boldsymbol{b}} = [\tilde{\boldsymbol{b}}_1,\ \tilde{\boldsymbol{b}}_2,\ \ldots,\ \tilde{\boldsymbol{b}}_t]$ is the corresponding offset matrix.

$$\widetilde{R}(\boldsymbol{w}) = \frac{1}{m} \sum_{i=0}^{m} l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w}), \tag{18}$$

$$\widetilde{J}(\boldsymbol{w}) = \widetilde{R}(\boldsymbol{w}) + \lambda \mathit{\Omega}(\boldsymbol{w}), \tag{19}$$

$$\widetilde{R}_t(\boldsymbol{w}) = \max_{0 \leq i \leq t} \left\{ \tilde{\boldsymbol{b}}_i + \langle \boldsymbol{w}, \tilde{\boldsymbol{a}}_i \rangle \right\}, \tag{20}$$

$$\boldsymbol{w}_t = \arg\min_{\boldsymbol{w}} \left\{ J_t(\boldsymbol{w}) := \widetilde{R}_t(\boldsymbol{w}) + \lambda \mathit{\Omega}(\boldsymbol{w}) \right\}, \tag{21}$$

$$\begin{aligned}\boldsymbol{\alpha}_t = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^{t+1}} \Big\{ & - \lambda \mathit{\Omega}^*(-\lambda^{-1} \widetilde{\boldsymbol{A}} \boldsymbol{\alpha}) \\ & + \boldsymbol{\alpha}^{\mathrm{T}} \tilde{\boldsymbol{b}} | \boldsymbol{\alpha} \geq 0, ||\boldsymbol{\alpha}||_1 = 1 \Big\}. \end{aligned} \tag{22}$$

Intuitively, as shown in Fig. 2, the estimate on the sampled dataset may not be an accurate cutting plane. It can be a secant plane (line 1), lower plane (line 2), or accurate cutting plane. However, irrespective of the type of estimated cutting plane, it will refine the linear lower bound of the objective function iteratively, and the solution to Eq. (21) will be updated to be closer to the optimum of the objective function. Therefore, an inaccurately estimated cutting plane can be used for iterations.

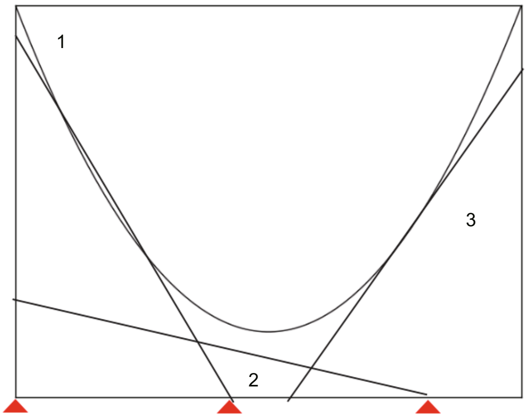

Fig. 2 All estimated cutting planes can be categorized into three types of inexact cutting planes

The first one (line 1) crosses the objective function, the second one (line 2) is isolated from the objective function, and the third one (line 3) is an accurate cutting plane of the objective function. In any case, all the three types of inexact cutting planes can still be informative. This can be illustrated in the figure that even with the inexact cutting plane to refine the linear lower bound, the minimum of the approximation moves toward the optimum of the objective function

The cutting plane in CPM is also regarded as a constraint (Tsochantaridis et al., 2005). CPM tightens the domain of the model parameter $\boldsymbol{w}$ along with the sequential creation of cutting planes. MBCPM is similar to this method. At each iteration, MBCPM calculates a newly estimated cutting plane to constrain the feasible region of the domain for model parameter $\boldsymbol{w}$. If the number of constraints is sufficiently large, a sufficiently accurate solution occurs.

3.2 A “sink” operation for noisy estimation

As previously mentioned, the estimation is always accompanied by noise. The negative effect of noise is illustrated in Fig. 3. In Fig. 3a, the estimation shields the optimum and MBCPM may not

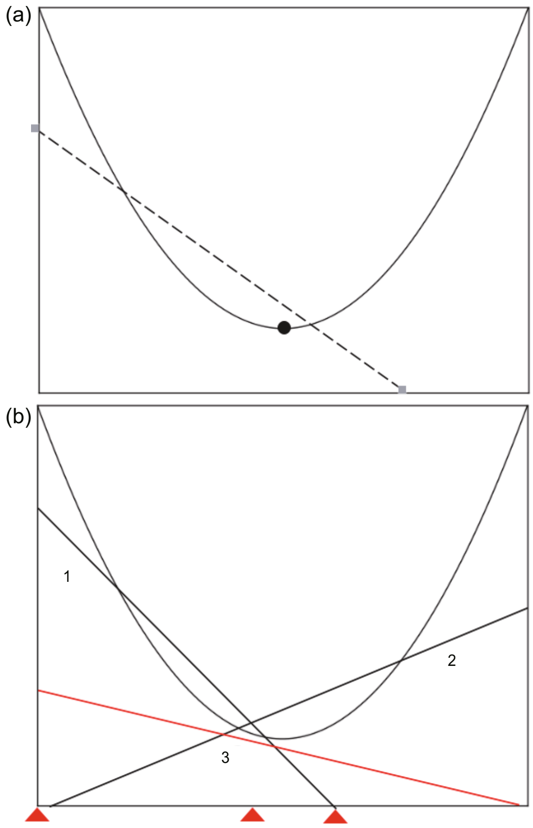

Fig. 3 Negative effects of noise: (a) the estimation prevents the algorithm from finding the minimum; (b) the next estimate is lower than the previous one

provide an acceptable final output. In Fig. 3b, the previous estimates are too high for the new estimate to refine the linear lower bound $\widetilde{R}_t(\boldsymbol{w})$. The solution to Eq. (21) will be the same as the previous solution.

Thus, the proposed algorithm sinks those noisy estimations with a multiplier $\rho = m/n$ to alleviate the negative effect of noisy estimations, where m is the size of the sampled dataset and n is the size of the entire dataset. If the noisy estimation needs to be sunk, its parameters $\tilde{\boldsymbol{a}}_i$ and $\tilde{\boldsymbol{b}}_i$ are replaced by $\tilde{\boldsymbol{a}}_i = \rho\tilde{\boldsymbol{a}}_i$ and $\tilde{\boldsymbol{b}}_i = \rho\tilde{\boldsymbol{b}}_i$, respectively.

The effectiveness of these assignments can be illustrated by the sampling process. Given that MBCPM calculates the parameters of the estimated cutting plane on a sampled dataset, an inequality can be formally inferred as

$$\begin{aligned} J(\boldsymbol{w}) &= \frac{1}{n}\sum_{i=0}^{n} l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \\ &\geq \frac{m}{n}\cdot\frac{1}{m}\sum_{i=0}^{m} l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \\ &= \rho\widetilde{R}(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \\ &\geq \rho(\tilde{\boldsymbol{a}}_t\boldsymbol{w} + \tilde{\boldsymbol{b}}_t) + \lambda\Omega(\boldsymbol{w}), \end{aligned} \tag{23}$$

where the first inequality is induced such that the empirical risk cannot be less than 0 ($l(x, y; \boldsymbol{w}) \geq 0$), and the second inequality stands because of the sub-differentiable property of the convex function (Hiriart-Urruty and Lemaréchal, 1993). From inequality (23), MBCPM guarantees the effectiveness of the "sink" operation because the adjusted noisy estimation $\rho(\tilde{\boldsymbol{a}}_i + \tilde{\boldsymbol{b}}_i) + \lambda\Omega(\boldsymbol{w})$ is lower than the objective function $J(\boldsymbol{w})$.

In terms of the detection of noisy estimations, we emphasize the detection of the type of noise in Fig. 3b (we will call this true noise in the following section). It is a fact that true noise makes it impossible to generate a new solution to refine the linear lower bound ($\widetilde{R}_t(\boldsymbol{w})$ in Eq. (20)), but the other kind of noise does not.

Specifically, true noise can be determined by comparing $\widetilde{J}(\boldsymbol{w}_t)$ and $J_t(\boldsymbol{w}_t)$. When $J_t(\boldsymbol{w}_t) \geq \widetilde{J}(\boldsymbol{w}_t)$ is observed for the first time, this might be an exception caused by the noise in the new estimation, which can be addressed by replacing this new estimation with a better one. Therefore, this kind of exception is acceptable in the algorithm, so MBCPM skips only the computation of Eq. (22) and calculates another cutting plane at the same point in the next iteration. This is shown in Algorithm 1 (lines 16 and 17). However, as shown in lines 7–11 in Algorithm 1, if this occurs several times (τ in line 7), it will be regarded as an exception caused by previous noisy estimations, rather than the noise in the new estimation. This type of exception is not acceptable in the algorithm. Therefore, MBCPM will identify these noisy estimations and use inequalities (2) and (3) to sink them.

In terms of the criterion for selection, MBCPM uses the Lagrange multiplier to perform this task. In the Lagrange dual problem in Eq. (22), each element in the Lagrange multiplier corresponds to an estimated cutting plane (Teo et al., 2010). $\boldsymbol{\alpha}_{t,i} > 0$ indicates that the corresponding cutting plane of $\boldsymbol{\alpha}_{t,i}$ is a support plane similar to lines 1 and 2 in Fig. 3b. Therefore, previously estimated noisy cutting planes are detected by observing whether their corresponding Lagrange multiplier elements satisfy the condition $\boldsymbol{\alpha}_{t,i} > 0$.

The pseudocode of the proposed algorithm is

Algorithm 1 Mini-batch cutting plane method

```
1: Initialize training set $D$, parameter $\lambda$,
      sampling size $m$, and maximum number of
      attempts $\tau$
2: $t \leftarrow 0$, sinking multiplier $\rho \leftarrow m/n$, where $n$ is
      the size of $D$, attempt count $C \leftarrow 0$
3: loop
4:    Randomly pick $m$ instances from $D$
5:    Calculate estimation: $\widetilde{R}(\boldsymbol{w}_{t-1})$, $\tilde{\boldsymbol{a}}_t$=$\partial_{\boldsymbol{w}}\widetilde{R}(\boldsymbol{w}_{t-1})$,
         and $\tilde{\boldsymbol{b}}_t = \widetilde{R}(\boldsymbol{w}_{t-1}) - \langle\boldsymbol{w}_{t-1}, \boldsymbol{a}_t\rangle$
6:    Refine $\widetilde{R}_t(\boldsymbol{w})$ and $J_t(\boldsymbol{w})$ with $\tilde{\boldsymbol{a}}_t$ and $\tilde{\boldsymbol{b}}_t$
7:    if $\widetilde{R}(\boldsymbol{w}_{t-1}) + \lambda\Omega(\boldsymbol{w}_{t-1}) \leq J_{t-1}(\boldsymbol{w}_{t-1})$ and
         $C \geq \tau$ then
8:       Sink planes: $\tilde{\boldsymbol{a}}_i = \rho\tilde{\boldsymbol{a}}_i$, $\tilde{\boldsymbol{b}}_i = \rho\tilde{\boldsymbol{b}}_i$ for estima-
            tion with $\boldsymbol{\alpha}_{t,i} > 0$
9:       Calculate $\boldsymbol{\alpha}_t$: solve Eq. (22)
10:      Update parameter: $\boldsymbol{w}_t = \partial\Omega^*(-\lambda^{-1}\boldsymbol{A}\boldsymbol{\alpha}_t)$
11:      $C \leftarrow 0$
12:   else if $\widetilde{R}(\boldsymbol{w}_{t-1}) + \lambda\Omega(\boldsymbol{w}_{t-1}) > J_{t-1}(\boldsymbol{w}_{t-1})$
      then
13:      Calculate $\boldsymbol{\alpha}_t$: solve Eq. (22)
14:      Update parameter: $\boldsymbol{w}_t = \partial\Omega^*(-\lambda^{-1}\boldsymbol{A}\boldsymbol{\alpha}_t)$
15:      $C \leftarrow 0$
16:   else
17:      $C \leftarrow C + 1$
18:   end if
19: end loop
```

shown in Algorithm 1. Lines 4 and 5 are used to calculate an estimate for a randomly sampled dataset. The detection strategy is reflected in lines 7–18. Line 8 is used to sink unacceptable noisy estimations.

4 Theoretical analysis

In this study, survival iterations are defined to satisfy the following characteristics:

1. Considering the survival iteration s, $\tilde{\boldsymbol{a}}_s\boldsymbol{w}+\tilde{\boldsymbol{b}}_s$ is the corresponding estimated cutting plane. Moreover, let $\boldsymbol{w}^*$ be the optimum of the objective function. Then we have $\tilde{\boldsymbol{a}}_s\boldsymbol{w}^*+\tilde{\boldsymbol{b}}_s \leq R(\boldsymbol{w}^*)$.

2. Corresponding estimations all avoid the "sink" operation.

Assumption 1 The sampled sub-dataset and the entire dataset are identically distributed.

Assumption 2 When the algorithm reaches the T^{th} iteration, we assume that there exist $\widetilde{T}$ $(0 < \widetilde{T} \leq T)$ survival iterations.

Assumption 1 is straightforward. Assumption 2 can be better understood by investigating the survival iterations. Based on the first assumption, the expectation of the estimated cutting plane is an accurate cutting plane of the objective function. Considering a point $\boldsymbol{w}_i$, $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are the parameters of the accurate cutting plane at $\boldsymbol{w}_i$, and $\tilde{\boldsymbol{a}}_i$ and $\tilde{\boldsymbol{b}}_i$ are estimations calculated on the sampled sub-dataset. Then $E(\tilde{\boldsymbol{a}}_i)=\boldsymbol{a}_i$ and $E(\tilde{\boldsymbol{b}}_i)=\boldsymbol{b}_i$ hold true. Given that the accurate cutting plane $(\boldsymbol{a}_i\boldsymbol{w}+\boldsymbol{b}_i)$ is lower than the objective function, the estimated cutting planes are expected to be lower than the optimum of the objective function, which can be formally expressed as

$$\begin{aligned} E(\tilde{\boldsymbol{a}}_i\boldsymbol{w}^*+\tilde{\boldsymbol{b}}_i) &= \boldsymbol{w}^*E(\tilde{\boldsymbol{a}}_i)+E(\tilde{\boldsymbol{b}}_i) \\ &= \boldsymbol{w}^*\boldsymbol{a}_i+\boldsymbol{b}_i \leq R(\boldsymbol{w}^*), \end{aligned}$$

where $\boldsymbol{w}^*$ is the optimum of the objective function. Thus, it is natural to assume that not all estimated cutting planes go against the first condition of survival iterations. Moreover, for those estimated cutting planes that satisfy the first condition of survival iteration, there exists a gap between the approximation and the optimum of the objective function, so the following estimations can cross the gap to refine the approximation. Thus, it will be easy for this kind of estimated cutting planes to avoid the "sink" operation, because this operation is designed to handle the unacceptable estimated cutting planes, similar to lines 1 and 2 in Fig. 3b that prevent the approximation from being refined. Therefore, Assumption 2 is readily understood. In addition, we define the survival rate as $\hbar=\widetilde{T}/T$.

Theorem 1 From the second condition of survival iterations, if one considers only all survival iterations and sequentially joins them to produce a survival iteration set S, there exists a positive constant $\epsilon > 0$ and $\epsilon \leq \min_{t\in S}\{\epsilon_t = J(\boldsymbol{w}_t)-J_t(\boldsymbol{w}_t) > 0\}$. For any sampled sub-datasets, Algorithm 1 guarantees an improvement in the minimum of the approximation J_t in these survival iterations. The improvement $\varDelta_t = J_{t+1}(\boldsymbol{w}_{t+1})-J_t(\boldsymbol{w}_t)$ is at least

$$\min\left\{\frac{\epsilon}{2}, \frac{\epsilon^2}{8Q^2}\right\}, \tag{24}$$

where $Q=\max_{i=1,2,\ldots,n}|\boldsymbol{x}_i|$ for the training set $\{(\boldsymbol{x}_i,\boldsymbol{y}_i),(\boldsymbol{x}_{i+1},\boldsymbol{y}_{i+1}),\ldots,(\boldsymbol{x}_n,\boldsymbol{y}_n)\}$.

Proof Based on this illustration, the violated condition $\epsilon_t = J(\boldsymbol{w}_t)-J_t(\boldsymbol{w}_t) = R(\boldsymbol{w}_t)-R_t(\boldsymbol{w}_t) > \epsilon > 0$ can be guaranteed. According to Franc and Sonnenburg (2008), this violated condition induces a least improvement $\varDelta_t$ of the dual objective if a new validation constraint is added at iteration t. Formally, $\varDelta_t = J_{t+1}(\boldsymbol{w}_{t+1})-J_t(\boldsymbol{w}_t)$ is not less than $\min\left\{\epsilon/2, \epsilon^2/8\widehat{Q}_t^2\right\}$, where $\widehat{Q}_t = \max_{i=1,2,\ldots,n}||\boldsymbol{x}_i||$ for any sampled sub-datasets $\{(\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),\ldots,(\boldsymbol{x}_m,\boldsymbol{y}_m)\}$. Obviously, $\widehat{Q}_t \leq Q$ is always true, so a least improvement (expression (24)) can be inferred.

Theorem 2 We denote $\hbar_t$ for the survival rate at iteration t. From Assumption 2, there exists a positive minimum survival rate $\sigma \leq \min_{t=1,2,\ldots,\infty}\{\hbar_t\}$. The iteration by which MBCPM achieves the optimum is at most

$$\max\left\{\frac{2R}{\epsilon\sigma}, \frac{8Q^2R}{\epsilon^2\sigma}\right\}, \tag{25}$$

where $R=J(\boldsymbol{0})$ is the initial value of the objective.

Proof Let S be the set of survival iterations. From the first condition of the survival iteration, the inequality $0 \leq J_t(\boldsymbol{w}_t) \leq J(\boldsymbol{w}^*)$ stands, where $t\in S$. Moreover, because $J(\boldsymbol{w}^*) \leq J(\boldsymbol{0}) = R(\boldsymbol{0}) = R_0$, it can be concluded that the sum of improvements $\sum_{i\in S}\varDelta_i$ cannot be greater than R_0. According to expression (24), the sum of improvements can be deduced to be greater than $|S|\cdot\min\left\{\epsilon/2, \epsilon^2/(8Q^2)\right\}$, where $|S|$ is the number of iterations in survival set

S. Thus, an inequality can be inducted as

$$|S| \cdot \min\left\{\frac{\epsilon}{2}, \frac{\epsilon^2}{8Q^2}\right\} \leq \sum_{i\in S} \varDelta_i \leq R_0, \qquad (26)$$

$$|S| \leq \max\left\{\frac{2R_0}{\epsilon}, \frac{8Q^2R_0}{\epsilon^2}\right\}, \qquad (27)$$

which means that the maximum number of iterations of survival set cannot be greater than $\max\left\{2R_0/\epsilon, 8Q^2R_0/\epsilon^2\right\}$. Considering the minimum survival rate σ, the maximum number of iterations (expression (25)) can be inferred.

Generally, CPM and its variants will stop when a precision condition, such as $\delta_t = \min_{1\leq i\leq t} J(\boldsymbol{w}_i) - J_t(\boldsymbol{w}_t) \leq \delta$ (δ is a pre-defined parameter), is met. However, the calculation of gap δ_t can be very difficult in MBCPM because the sampling objective $\widetilde{J}(\boldsymbol{w})$ is not a precise estimate of $J(\boldsymbol{w})$ on the entire dataset. Therefore, MBCPM adopts the output strategy of SGD, in which it provides a valuable final output via iterations. As illustrated in Theorem 2, MBCPM can reach its optimum with a limited number of iterations. Afterward, the MBCPM's linear lower bound $\widetilde{R}_t$ will continue to be refined owing to the "sink" operation. However, the refinement is limited because the "sink" operation will be conducted only on noisy estimations. Consequently, MBCPM will fluctuate only on a small scale around the optimum, which guarantees the quality of the output.

The minimum survival rate σ represents the randomness of the sampling procedure. The greater the randomness is, the smaller σ will be, and the more the iterations that are required. In turn, fewer iterations are required for convergence. If MBCPM is trained on the entire dataset, there will be no randomness. Then $\sigma = 1$ and the convergence rate of MBCPM in expression (25) is comparative to Proposition 18 of the standard CPM in Tsochantaridis et al. (2005).

5 Experiments

Considering that the objective of MBCPM is to solve the convex optimization problem, while the convex objective function has no local minimum, we chose several widely welcome datasets to evaluate the proposed algorithm. This is a common practice in the literature (Franc and Sonnenburg, 2008; Teo et al., 2010; Mokhtari et al., 2018) on convex optimization. The experiments were conducted on four classification datasets: URL (Ma et al., 2009), HUMAN (Sonnenburg and Franc, 2010), ACOUSTIC (Duarte and Hu, 2004), and MNIST (LeCun et al., 1998). URL is used to identify suspicious uniform resource locators (URLs), HUMAN is a bioinformatics dataset used for splice site recognition, ACOUSTIC is a multiclassification dataset used for vehicle classification, and MNIST is a large database of handwritten digits, commonly used for training various image processing systems. For data in HUMAN with string-valued attributes, we re-encoded each attribute into a four-dimensional 0/1 vector using one-hot encoding in the preprocessing. A summary of the datasets is listed in Table 1.

Table 1 Statistics of datasets

Dataset	Training size	Test size	Dimension	Number of categories
HUMAN	20 000	20 000	1120	2
URL	2 380 000	16 130	3 231 961	2
ACOUSTIC	78 823	19 705	50	3
MNIST	60 000	10 000	784	10

The target problem is formatted as Eq. (1), and the regularized term in Eq. (1) is defined as $\varOmega(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||_2^2$. For a binary classification task on URL and HUMAN, $l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w})$ is formally expressed as Eq. (28), which is a well-known hinge loss function (Bennett and Mangasarian, 1992):

$$l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w}) = \max\left(0, 1 - \boldsymbol{y}_i\langle\boldsymbol{w}, \boldsymbol{x}_i\rangle\right). \qquad (28)$$

For a multiclass classification task on ACOUSTIC and MNIST, $l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w})$ is a soft-margin multiclass function (Crammer and Singer, 2003):

$$l(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{w}) = \max_{\boldsymbol{y}\neq\boldsymbol{y}_i}(0, \varDelta(\boldsymbol{y}_i, \boldsymbol{y}) + f_y - f_{y_i}), \qquad (29)$$

where $f_y = \langle\boldsymbol{w}_y, \boldsymbol{x}_i\rangle$. It should be mentioned that both Eqs. (28) and (29) are non-differentiable, with only sub-gradients being available.

In the experiments, MBCPM was compared with BMRM and four popular SGD methods: SGD (Bottou, 2010), momentum, AdaGrad, and Adam. The relevant information on BMRM is introduced in Section 2.1 and the details of the chosen SGD methods were provided in Qian (1999), Bottou (2010), Duchi et al. (2011), and Kingma and Ba (2014).

Moreover, to fairly compare the performances of MBCPM and SGD methods, we analyzed the general equality (execution time ≈ gradient calculation time + model update time). We considered that the main difference between MBCPM and SGD methods lies in the approach for updating the model parameters. As such, we need to control the discrepancy of gradient calculation time. Therefore, we implemented MBCPM and SGD methods based on the code of BMRM (http://users.cecs.anu.edu.au/~chteo/BMRM.html) to complete the comparison, and the implemented code was uploaded to the GitHub (https://github.com/LuMelon/MBCPMRelatedCode). Note that to prevent disturbance of other randomness, sparse computing was not employed to handle the high dimensional but sparse data (URL). This was to address the concern that the different numbers of attributes in different instances may cause gradient calculation time to vary for different methods. Thus, as listed in Table 2, the average gradient calculation time of MBCPM was approximately equal to that of the SGD methods. As such, the discrepancy of performance was attributed mainly to the model update time. Moreover, the learning rate η in SGD was adapted by $\eta_0/(1+\lambda_0\eta_0 t)$, where η_0 is the initial learning rate and λ_0 is a regularized term. Note that SGD methods are very sensitive to the value of the learning rate. A cross-validation was conducted to choose an appropriate learning rate in the experiments, and the information used to select the parameters is listed in Table 3, where η_0 is the initial learning rate, ζ in momentum SGD is the parameter used as a tradeoff between the previous momentum and the current gradient, ϵ in AdaGrad and Adam is used to prevent invalid division, β_1 is used to decay the accumulation of gradients, and β_2 is the same as ζ in momentum SGD. In addition, all optimizers started from the initial vector $\mathbf{0}$, and the regularized parameter λ in Eq. (1) was set to 0.5.

Table 2 Average gradient calculation time of the different methods for each dataset

Method	Average gradient calculation time (s)			
	HUMAN	URL	ACOUSTIC	MNIST
SGD	3.10	873	0.231	1.200
Momentum SGD	3.23	869	0.230	1.260
AdaGrad	3.08	875	0.235	1.255
Adam	3.15	870	0.237	1.270
MBCPM	3.25	877	0.232	1.226
BMRM	27.26	–	1.920	11.100

Table 3 Parameters used to deploy baseline methods

Method	Parameter(s)
SGD	$\lambda_0=0.01$, $\eta_0=0.01$
Momentum SGD	$\zeta=0.9$, $\eta_0=0.01$
AdaGrad	η_0=0.01, $\epsilon=10^{-8}$
Adam	$\beta_1=0.9$, β_2=0.9, η_0=0.01, ϵ=10^{-8}
MBCPM	$\tau=5$

Numerical experiments were conducted on a machine running Ubuntu 16.04, with 40 Intel® Xeon® (2.3 GHz) processors and 258 GB RAM.

5.1 Comparison with BMRM

BMRM is an efficient version of the traditional CPM algorithm. Different from the traditional CPM algorithms, MBCPM uses only a small batch of data at each iteration, which facilitates fast iterations.

In the training phase, we excluded URL because the size of URL is too large to fit in the main memory of a single machine, which prohibits BMRM from loading the entire dataset into the memory. With respect to the other datasets, the batch size in MBCPM was set to be 10% of the training set.

The experimental results were shown in Figs. 4 and 5, from which it can be observed that even though more iterations are required, MBCPM requires less time to achieve a comparative objective value. The requirement of more iterations was explained by the survival rate mentioned in Section 3, given that some randomness was introduced into MBCPM because it uses a small batch of data instead of the entire data to perform iterations.

This experiment can also be used to observe the existence of survival iterations. Survival iterations were difficult to intuitively grasp because the first condition was difficult to judge and the optimum was always unknown. However, given that expression (25) is inferred from the assumption of survival iteration, combining expression (25) with Fig. 5, we observed the survival rate σ from a different perspective, whereby the required number of iterations for convergence is a little more than that of the traditional CPM method. Based on Table 2, the average gradient calculation time of MBCPM was significantly shorter than that of BMRM, which causes the former to iterate at a much faster rate. Thus, this experiment was also a verification of the motivation

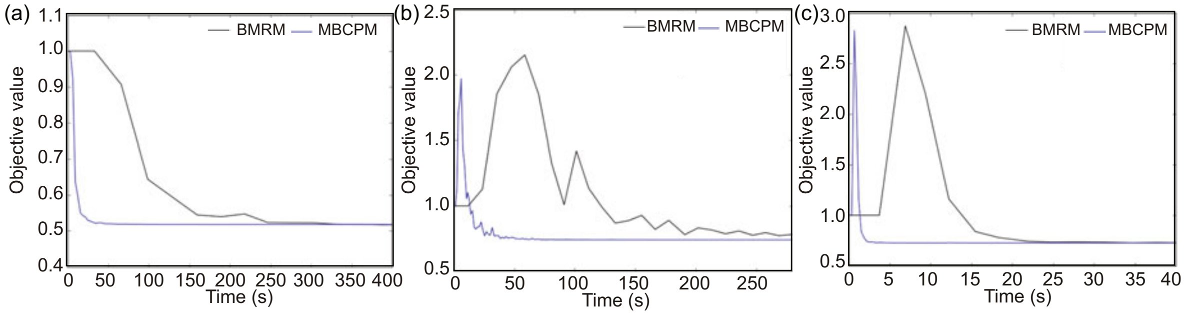

Fig. 4 Objective value of MBCPM and BMRM as a function of time consumption: (a) HUMAN; (b) MNIST; (c) ACOUSTIC

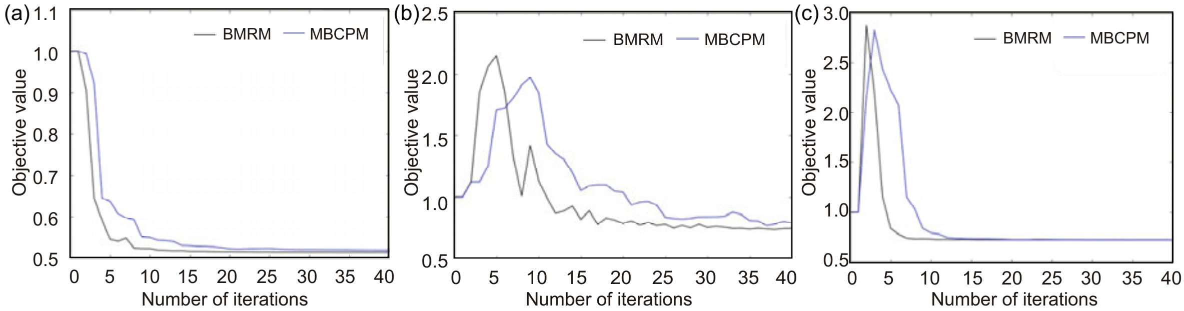

Fig. 5 Objective value of MBCPM and BMRM as a function of the iteration number: (a) HUMAN; (b) MNIST; (c) ACOUSTIC

of this study. The traditional CPM methods were trapped mainly in their slow iteration speed and the acceleration of the iteration of CPM methods can greatly improve their performance.

5.2 Comparison between SGD and its variants

In this subsection, MBCPM was compared with several current popular SGD methods, including AdaGrad, Adam, momentum SGD, and naive SGD.

In the training phase, the batch size on URL was set to 1000, while others were set to be 10% of the training dataset. Additionally, the metrics in Kingma and Ba (2014) were used to compare MBCPM with the other algorithms based on the objective value ($J(\boldsymbol{w})$ in Eq. (1)) as a function of time consumption. For each iteration, the output $\boldsymbol{w}_t$ was evaluated on a test dataset. Moreover, the corresponding error rates of output $\boldsymbol{w}_t$ on the test dataset were provided.

The numerical experimental results were presented in Figs. 6 and 7. It was observed that the proposed MBCPM mostly achieved the best performance.

Even though SGD methods were widely used for their simplicity, there were still problems associated with the strategy of (sub)gradient descent. The adjustment of the model parameter $\boldsymbol{w}$ was based only on the local information ((sub)gradient of the current position) rather than the global information of the objective function. Momentum SGD, AdaGrad, and Adam were exactly the remedy for this problem. In momentum SGD, AdaGrad, and Adam, the accumulation of the (sub)gradient and the adaptive learning rate can be regarded as a kind of interaction with the objective function. Thus, using the accumulation of the (sub)gradient and the adaptive learning rate to adjust the model can achieve better performance. Figs. 6 and 7 showed that those improved SGD methods converged faster than the naive SGD.

A similar analysis can be used to explain the advantage of MBCPM over SGD methods, as depicted in Figs. 6 and 7. In the MBCPM model parameter $\boldsymbol{w}$ is adjusted by finding the minimum of the piecewise linear lower bound rather than by following a certain direction, where the piecewise linear lower bound is an approximation of the objective function, so the adjustment of the model in MBCPM is based on the

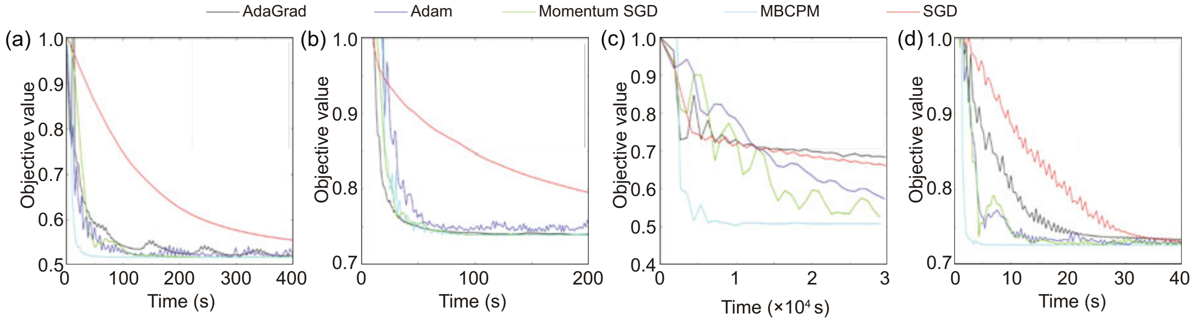

Fig. 6 Objective value as a function of time cost in the comparison of MBCPM with SGD methods: (a) HUMAN; (b) MNIST; (c) URL; (d) ACOUSTIC

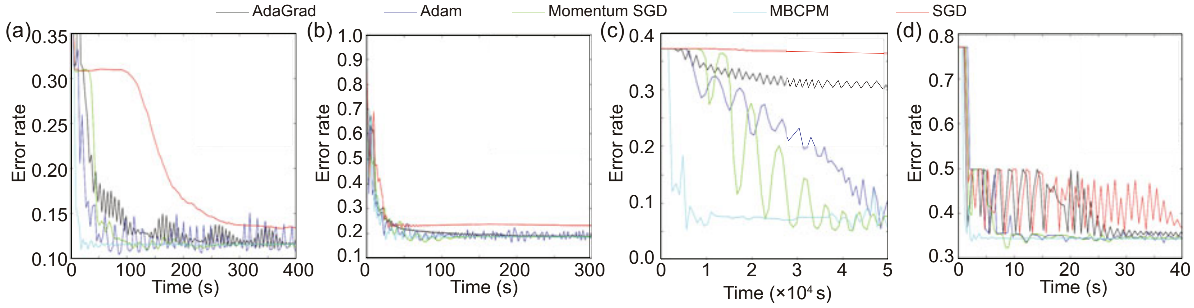

Fig. 7 Error rate as a function of time cost in the comparison of MBCPM with SGD methods: (a) HUMAN; (b) MNIST; (c) URL; (d) ACOUSTIC

estimation of the objective function. On the contrary, although the accumulation of (sub)gradients and the adaptative learning rate can be regarded as an interaction with the objective function, it is not straightforward information about the objective function. This difference was reflected by the experimental results that MBCPM converged faster than the improved SGD methods.

As illustrated by the experimental results, even though the adjustment of model parameter $\boldsymbol{w}$ in MBCPM is more complex than that in gradient descent methods, the extra computation in MBCPM facilitates faster convergence.

6 Conclusions

We proposed a novel convex optimization solver named MBCPM in this study. MBCPM is an extension to the conventional CPM. It performs iterations with estimated cutting planes on sampled datasets instead of accurate cutting planes on the entire dataset. A "sink" operation is employed to alleviate the influence of noise introduced by the estimations. This study illustrates the effectiveness of MBCPM theoretically and practically. The experimental results on extensive real-world datasets indicated that under the general convex settings, MBCPM is superior to some widely used SGD methods and the conventional CPM method in terms of convergence speed. In the future, we plan to investigate the scalability of MBCPM and its application to large-scale data-intensive problems.

Compliance with ethics guidelines

Meng-long LU, Lin-bo QIAO, Da-wei FENG, Dong-sheng LI, and Xi-cheng LU declare that they have no conflict of interest.

References

Belloni A, 2005. Introduction to Bundle Methods. Lecture Notes for IAP, Operations Research Center, MIT, USA.

Bennett KP, Mangasarian OL, 1992. Robust linear programming discrimination of two linearly inseparable sets. *Optim Methods Softw*, 1(1):23-34. https://doi.org/10.1080/10556789208805504

Bottou L, 2010. Large-scale machine learning with stochastic gradient descent. Proc 19th Int Conf on Computational Statistics, p.177-187. https://doi.org/10.1007/978-3-7908-2604-3_16

Crammer K, Singer Y, 2003. Ultraconservative online algorithms for multiclass problems. *J Mach Learn Res*, 3:951-991.

Duarte M, Hu YH, 2004. Vehicle classification in distributed sensor networks. *J Parall Distr Comput*, 64(7):826-838. https://doi.org/10.1016/j.jpdc.2004.03.020

Duchi J, Hazan E, Singer Y, 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J Mach Learn Res*, 12:2121-2159.

Franc V, Sonnenburg S, 2008. Optimized cutting plane algorithm for support vector machines. Proc 25th Int Conf on Machine Learning, p.320-327. https://doi.org/10.1145/1390156.1390197

Hiriart-Urruty JB, Lemaréchal C, 1993. Convex Analysis and Minimization Algorithms I. Springer, Berlin, Germany. https://doi.org/10.1007/978-3-662-02796-7

Kelley JEJr, 1960. The cutting-plane method for solving convex programs. *J Soc Ind Appl Math*, 8(4):703-712. https://doi.org/10.1137/0108053

Kingma DP, Ba J, 2014. Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980

Kiwiel KC, 1983. An aggregate subgradient method for nonsmooth convex minimization. *Math Program*, 27(3):320-341. https://doi.org/10.1007/BF02591907

Kiwiel KC, 1990. Proximity control in bundle methods for convex nondifferentiable minimization. *Math Program*, 46(1):105-122. https://doi.org/10.1007/BF01585731

LeCun Y, Bottou L, Bengio Y, et al., 1998. Gradient-based learning applied to document recognition. *Proc IEEE*, 86(11):2278-2324. https://doi.org/10.1109/5.726791

Lemaréchal C, Nemirovskii A, Nesterov Y, 1995. New variants of bundle methods. *Math Program*, 69(1):111-147. https://doi.org/10.1007/BF01585555

Ma J, Saul LK, Savage S, et al., 2009. Identifying suspicious URLs: an application of large-scale online learning. Proc 26th Int Conf on Machine Learning, p.681-688. https://doi.org/10.1145/1553374.1553462

Mokhtari A, Eisen M, Ribeiro A, 2018. IQN: an incremental quasi-Newton method with local superlinear convergence rate. *SIAM J Opt*, 28(2):1670-1698. https://doi.org/10.1137/17M1122943

Mou LL, Men R, Li G, et al., 2016. Natural language inference by tree-based convolution and heuristic matching. Proc 54th Annual Meeting of the Association for Computational Linguistics, p.130-136.

Qian N, 1999. On the momentum term in gradient descent learning algorithms. *Neur Netw*, 12(1):145-151. https://doi.org/10.1016/S0893-6080(98)00116-6

Schramm H, Zowe J, 1992. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM J Opt*, 2(1):121-152. https://doi.org/10.1137/0802008

Sonnenburg S, Franc V, 2010. COFFIN: a computational framework for linear SVMs. Proc 27th Int Conf on Machine Learning, p.999-1006.

Teo CH, Vishwanthan SVN, Smola AJ, et al., 2010. Bundle methods for regularized risk minimization. *J Mach Learn Res*, 11:311-365. https://doi.org/10.1145/1756006.1756016

Tsochantaridis I, Joachims T, Hofmann T, et al., 2005. Large margin methods for structured and interdependent output variables. *J Mach Learn Res*, 6:1453-1484.