正則化付き経験リスク最小化における分散最適化法

Shin Matsushima[1], Hyokun Yun[2], S.V.N. Vishwanathan[3]
[1] 東京大学　情報理工学系研究科，[2]Amazon Inc.，[3]University of Califolnia, Santa Cruz
e-mail：shin_matsushima@mist.i.u-tokyo.ac.jp

1　概要

多くの機械学習における問題は正則化付き経験リスク最小化問題として定式化され，この問題の最適化には確率最適化による手法が広く用いられてきた．大量のデータを用いる場合，アルゴリズムを分散計算機環境などを利用し並列実行することが望ましいが，既存の手法では効率的な並列実行が難しい．本稿では正則化付き経験リスク最小化問題が，ある min-max 問題と等価であり，この問題が効率的で自然な分散確率最適化アルゴリズムを導くことを示す．また，導かれる分散確率最適化アルゴリズムは SVM（サポートベクターマシン）において，既存のアルゴリズムより効率が良いことを示す．

2　分散確率最適化の問題

正則化付き経験リスク最小化問題（RRM）は次の関数 $P(\mathbf{w})$ を最小化する問題として定義される．

$$P(\mathbf{w}) = \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{n} \sum_{i=1}^{n} \ell_i(\langle \mathbf{w}, \mathbf{x}_i \rangle).$$

ここで，$\mathbf{x}_i \in \mathbb{R}^d$ はデータであり，ϕ_j, ℓ_i は凸関数であるとする．また $\lambda > 0$ は正則化項と損失項を調整するパラメータである．SVM，ロジスティック回帰，LASSO など多くの機械学習の問題がこの形で書き下すことができる [1].

BMRM[2] などの勾配を用いる最適化法は勾配の計算を並列実行することができるが，データ数 n が大きい場合，機械学習の汎化誤差の評価において確率勾配法が漸近的に有利であることが知られている [3]．そのため確率勾配法が広く利用されて来た．

一方，確率勾配法は各確率勾配の計算は非常に高速なため並列実行による高速化は望めない．多くの分散環境での確率勾配法はこの問題を常に最新のパラメータによる確率勾配を用いないというヒューリスティクスで回避してきた．

本手法では RRM を本質的に並列実行可能な min-max 問題に帰着させる事により，確率最適化の並列実行における困難を解消した [4].

3　分散確率最適化アルゴリズム

我々は RRM をある min-max 問題で再定式化し，本質的に並列実行可能なアルゴリズムを導く．$\Omega = \{(i,j) | x_{ij} \neq 0\}$ とし，以下の問題

$$\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} f(\mathbf{w}, \boldsymbol{\alpha}) := \sum_{(i,j)\in\Omega} f_{i,j}(w_j, \alpha_i) \quad (1)$$

を考える．ここで $\Omega_i = \{j | x_{ij} \neq 0\}$，$\bar{\Omega}_j = \{i | x_{ij} \neq 0\}$ とすると，

$$f_{i,j}(w_j, \alpha_i) = \frac{\lambda \phi_j(w_j)}{|\bar{\Omega}_j|} - \frac{\ell_i^\star(-\alpha_i)}{n|\Omega_i|} - \frac{\alpha_i w_j x_{ij}}{n}$$

である．ただし * は Fenchel 双対を表す．$f_{i,j}, -f_{i,j}$ は $\mathbf{w}, \boldsymbol{\alpha}$ に関してそれぞれ凸であり，min-max 問題 (1) の解 $\mathbf{w}$ と RRM の解は一致する．この問題に対し以下の形の確率勾配法を考える．

$$\begin{aligned} w_j &\leftarrow w_j - \eta \nabla_w f_{i,j}(w_j, \alpha_i) \\ \alpha_i &\leftarrow \alpha_i - \eta \nabla_\alpha - f_{i,j}(w_j, \alpha_i), \end{aligned} \quad (2)$$

この更新式は x_{ij}, α_i, w_j のみの情報を用いて一回の更新を実行可能である．さらに $i \neq i'$ かつ $j \neq j'$ ならば，x_{ij}, α_i, w_j を用いた更新と $x_{i'j'}, \alpha_{i'}, w_{j'}$ を用いた更新は完全に独立であり，並列に実行可能である．この特徴を生かした分散最適化法が自然に導かれる．

以下では $[p] := \{1, \ldots, p\}$ と定義する．利用可能なプロセッサの数を p とおく．プロセッサは計算機内で共有されていてもされていなくてもよい．各プロセッサ $q \in [p]$ は部分データ $X^{(q)} = \{x_i | i \in I_q\}$ 部分パラメータ $\boldsymbol{\alpha}^{(q)} = \{\alpha_i | i \in I_q$ を所有する．ここで $\{I_q\}_{q=1}^p$ は $[n]$ の分割とする．さらに，$[d]$ の分割 $\{J_q\}_{q=1}^p$ に従い $X^{(q)}$ を p 個の部分 $\Omega^{(q,r)} = \{x_{ij} | i \in I_p, j \in J_r\}$ に分割する．同様に $\mathbf{w}^{(r)} = \{w_j | j \in J_r\}$ とし，初期状態でプロセッサ q は $\mathbf{w}^{(q)}$ を所有する．この時，各プロセッサは $\Omega^{(q,q)}$ に関して (2) による更新を同時に実行可能である．一般に，$\sigma_r(q) = \{(q + r - 2) \mod p\} + 1$ とし，$\Omega^{(q,\sigma_r(q))}$ は同時に更新可能である．よって各プロセッサは所有する $\boldsymbol{\alpha}^{(q)}, \mathbf{w}^{(\sigma_r(q))}, \Omega^{(q,\sigma_r(q))}$ に関する更新を終えた後に $\mathbf{w}^{(\sigma_r(q))}$ を送信し，$\mathbf{w}^{(\sigma_{r+1}(q))}$ を受信することで，更新を反復的に行うことができる．全体のアルゴリズムを以下に示す．

Algorithm 1 問題 (1) に対する分散最適化法

$t \leftarrow 1$
repeat
  $\eta_t \leftarrow \eta_0/\sqrt{t}$
  for all $r \in [p]$ **do**
    for all processors $q \in [p]$ **do**
      for $(i,j) \in \Omega^{(q,\sigma_r(q))}$ **do**
        $w_j \leftarrow w_j - \eta_t \left(\frac{\lambda \nabla \phi_j(w_j)}{|\Omega_j|} - \frac{\alpha_i x_{ij}}{n} \right)$
        $\alpha_i \leftarrow \alpha_i - \eta_t \left(\frac{\nabla \ell_i^\star(-\alpha_i)}{n|\Omega_i|} + \frac{w_j x_{ij}}{n} \right)$
      end for
      $\mathbf{w}^{(\sigma_r(q))}$ をプロセッサ $\sigma_{r+1}^{-1}(\sigma_r(q))$ に送信し $\mathbf{w}^{(\sigma_{r+1}(q))}$ を受信する．
    end for
  end for
  $t \leftarrow t+1$
until 収束条件を満たす

4 収束定理

この提案スキームに対し以下が示される [4].

Theorem 1. *Algorithm 1* を $p \leq \min(m,d)$ 個のプロセッサで並列実行したときの t 回目の反復終了時のパラメータの値を $(\mathbf{w}^t, \boldsymbol{\alpha}^t)$ とする．また $(\tilde{\mathbf{w}}^t, \tilde{\boldsymbol{\alpha}}^t) := \left(\frac{1}{t}\sum_{s=1}^t \mathbf{w}^s, \frac{1}{t}\sum_{s=1}^t \boldsymbol{\alpha}^s\right)$ とする．さらに次を仮定する：

- 任意の q で $\left|\Omega^{(q,\sigma_r(q))}\right| \approx \frac{1}{p^2}|\Omega|$ かつ $|J_q| \approx \frac{d}{p}$ となる Ω の分割が存在する．
- (2) の更新は定数時間 T_{u} で実行される．
- w_j の送受信は定数時間 T_{c} で実行される
- 任意の $(\mathbf{w}, \boldsymbol{\alpha})$，$(\mathbf{w}', \boldsymbol{\alpha}')$ に対し $\|\mathbf{w}-\mathbf{w}'\|^2 + \|\boldsymbol{\alpha}-\boldsymbol{\alpha}'\|^2 \leq D$ となる D が存在する．
- 任意の $(\mathbf{w}, \boldsymbol{\alpha})$ に対し $\|\nabla_{\mathbf{w}} f_{i,j}(w_j, \alpha_i)\|^2 \leq C_w^2$，$\left\|\nabla_{\boldsymbol{\alpha}} - f_{i,j}(w_j, \alpha_i)\right\|^2 \leq C_\alpha^2$ となる C_α, C_w が存在する．

この時実行時間 $\left(\frac{|\Omega| T_{\mathrm{u}}}{p} + dT_{\mathrm{c}}\right) T$ 以内に

$$\max_{\boldsymbol{\alpha}'} f\left(\tilde{\mathbf{w}}^T, \boldsymbol{\alpha}'\right) - \min_{\mathbf{w}'} f\left(\mathbf{w}', \tilde{\boldsymbol{\alpha}}^T\right) \leq \sqrt{\frac{2DC}{T}}.$$

を満たす $\mathbf{w}^T, \boldsymbol{\alpha}^T$ を得る．C はある定数である．

5 実験結果

分散環境を用いて提案手法（DSO）と PSGD[5] および BMRM[2] との比較を行った．実装は C++を用いて行い，MPI には MPICH2 を用いた．ステップサイズの調節には AdaGrad[6] を用いた．実験は 4 台の計算機で各 8 プロセスを用い，時間ごとの $P(\mathbf{w})$ の値を比較した．データセットは webspam[1] ($n = 3.5 \cdot 10^5, d = 1.66 \cdot 10^7$) を用いた．

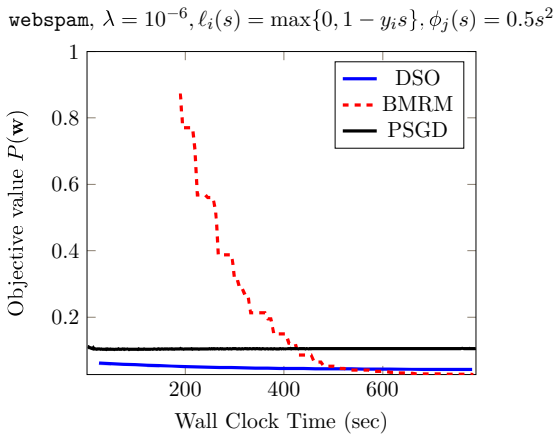

図 1: SVM における各手法の収束速度比較

6 結論

本稿では新しい定式化に基づく正則化付き経験リスク最小化問題における分散最適化法た．この方法は強凸性のない，より一般的な目的関数にも適用可能であり，また非同期的なスキームへの拡張も自然に考えられる．

謝辞 本研究は JST-CREST の支援を受けた．

参考文献

[1] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of statistical learning, Springer (2009).

[2] C. Teo, S.V.N. Vishwanthan, A. Smola and Q. Le. Bundle methods for regularized risk minimization. JMLR (2010).

[3] L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. Optimization for Machine Learning (2011).

[4] S. Matsushima, H. Yun, and S.V.N. Vishwanathan. Distributed stochastic optimization of the regularized risk, ArXiv (2014).

[5] M. Zinkevich, A. Smola, M. Weimer, and L. Li. Parallelized stochastic gradient descent. In NIPS (2010).

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. JMLR (2010).

[1] http://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html