Mini-Batch Primal and Dual Methods for SVMs

Martin Takáč
Edinburgh University

Avleen Bijral
TTI-Chicago

Peter Richtárik
Edinburgh University

Nati Srebro
TTI-Chicago

Stochastic SVM Optimization

- $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\|\mathbf{x}_i\| \le 1$, $y_i \in \pm 1$
- SVM Primal Objective:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathcal{P}(\mathbf{w}) := \tfrac{1}{n}\sum_{i=1}^{n} \ell(y_i \langle \mathbf{w}, \mathbf{x}_i\rangle) + \tfrac{\lambda}{2}\|\mathbf{w}\|^2,$$

where $\ell(z) := [1-z]_+ = \max\{0, 1-z\}$

- SVM Dual Objective:

$$\max_{\alpha\in\mathbb{R}^n, 0\le\alpha_i\le 1} \mathcal{D}(\alpha) := \tfrac{-1}{2\lambda n^2}\alpha^T \mathbf{Q}\alpha + \tfrac{1}{n}\sum_{i=1}^{n}\alpha_i,$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\quad \mathbf{Q}_{i,j} = y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle$

SGD sequential update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla \hat{\mathcal{P}}_j(\mathbf{w}^{(t)}),$$

where $\hat{\mathcal{P}}_j(\mathbf{w}^{(t)}) := \ell(y_j \langle \mathbf{w}, \mathbf{x}_j\rangle) + \tfrac{\lambda}{2}\|\mathbf{w}\|^2$

SDCA sequential update

$$\alpha^{(t+1)} = \alpha^{(t)} + \delta_j^* e_j,$$

where $\delta_j^* := \arg\max_{0\le\alpha_j+\delta_j\le 1} \mathcal{D}(\alpha^{(t)} + e_j\delta_j)$

- Methods of choice for large data, but **inherently sequential** and difficult to parallelize
- Parallelization via **"mini-batches"**: at iteration t, instead of a single example j, operate on b random examples $A_t \subseteq \{1, \ldots, n\}$, $|A_t| = b$

Contributions

- Data-dependent (not worst-case) analysis with **non-smooth hinge-loss**
- $\sigma^2 = \frac{1}{n}\|X\|^2 = \frac{1}{n}\|\sum_{i=1}^n \mathbf{x}\mathbf{x}_i^T\| = \frac{1}{n}\|\mathbf{Q}\|$ is **a data-dependent quantity** controlling speedups
- Both for **SGD** and **SDCA**
- Naïve **SDCA** mini-batching **might fail!** Our modifications **make it work** and lead to **parallelization speedups**

Mini-Batch SGD

SGD Mini-Batch Update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \tfrac{1}{b}\textstyle\sum_{j\in A_t} \nabla\hat{\mathcal{P}}_j(\mathbf{w}^{(t)})$$

Output: $\bar{\mathbf{w}}^{(T)} = \frac{2}{T}\sum_{t=T/2+1}^{T} \mathbf{w}^{(t)}$

- Mini-batching is bad in worst case
- Prior work: speedups only when $\ell(\cdot)$ replaced by a smooth loss

Theorem: With $\eta_t = 1/(\lambda t)$, after

$$T = \frac{\beta_b}{b}\frac{30}{\lambda\epsilon} \quad \text{iterations}$$

of Mini-Batch SGD: $\mathbf{E}\left[\mathcal{P}(\bar{\mathbf{w}}^{(T)})\right] - \mathcal{P}(\mathbf{w}^*) \le \epsilon$,

where $\beta_b = 1 + (b-1)\left(\frac{n}{n-1}\sigma^2 - \frac{1}{n-1}\right)$.

- Worst: $\sigma^2 = 1$ (i.e., $\beta_b = b$) $\Rightarrow$ no speedup ($\mathbf{x}_i$ co-linear—data concentrated on single 1d line)
- "Best": $\sigma^2 = 1/n$ (i.e., $\beta_b = 1$) $\Rightarrow$ linear speedup ($\mathbf{x}_i$ orthogonal—no correlations between points)
- Realistic: $\sigma^2 < 1 \Rightarrow$ $\beta_b = O(1)$ and linear speedup until $b = O(1/\sigma^2)$.

Proof sketch

- For a projection $v_{[A]}$ of any $v \in \mathbb{R}^n$ onto b random coordinates $A \subset \{1, \ldots, n\}$, $|A| = b$:

$$\mathbf{E}[v_{[A]}^\top Q v_{[A]}] \le \frac{b}{n}\beta_b\|v\|^2.$$

- Conclusion:

$$\mathbf{E}\left[\|\tfrac{1}{b}\sum_{j\in A_t}\nabla\hat{\mathcal{P}}_j(\mathbf{w}^{(t)})\|^2\right] = \mathbf{E}[\|\tfrac{1}{b}\textstyle\sum_{i\in A}\chi_i y_i \mathbf{x}_i\|^2]$$
$$= \tfrac{1}{b^2}\mathbf{E}[\chi_{[A]}^T \mathbf{Q}\chi_{[A]}] \le \tfrac{1}{b^2}\tfrac{b}{n}\beta_b\|\chi\|^2 \le \tfrac{\beta_b}{b},$$

where $\chi_i = 1$ if $y_i\langle\mathbf{w}, \mathbf{x}_i\rangle < 1$ and $\chi_i = 0$ otherwise

- Standard SGD analysis + refined subgradient bound

Mini-Batch SDCA

Naïve Mini-Batch SDCA Update

$$\alpha^{(t+1)} = \alpha^{(t)} + \textstyle\sum_{j\in A_t}\delta_j^* e_j$$

(similar to (Bradley et al. 2011) for ℓ_1 learning)

- **Naïve mini-batched SDCA can fail to converge to optimum!**
- Parallel updates from correlated $\{\mathbf{x}_i\}$ can overshoot desired point

Safe Mini-Batch SDCA Update

$$\alpha^{(t+1)} = \alpha^{(t)} + \textstyle\sum_{j\in A_t}\delta_j^\beta e_j$$

$$\delta_j^\beta := \arg\max_{0\le\alpha_j+\delta_j\le 1} \mathcal{D}(\alpha^{(t)} + e_j\delta_j) + \frac{\beta - Q_{jj}}{2\lambda n^2}\delta_j^2$$

Stepsize $1/\beta$ (not present in standard SDCA) needed to ensure convergence:

Theorem: With $\beta = \beta_b$, after

$$T = 2\frac{n}{b}\log\left(\frac{2\lambda n}{\beta_b} + 2\right) + \frac{\beta_b}{b}\frac{8}{\lambda\epsilon}$$

iterations of Safe SDCA:

$$\mathbf{E}[\mathcal{P}(\mathbf{w}(\bar{\alpha}))] - \mathcal{P}(\mathbf{w}^*) \le \mathbf{E}[\mathcal{P}(\mathbf{w}(\bar{\alpha})) - \mathcal{D}(\bar{\alpha})] \le \epsilon,$$

where $\bar{\alpha} = \frac{2}{T}\sum_{t=T/2+1}^{T}\alpha^{(t)}$, $\mathbf{w}(\alpha) := \frac{1}{\lambda n}\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$.

Proof Sketch:

- Modification of Shalev-Shwartz & Zhang (JMLR 2013; **duality gap analysis**) and Richtárik & Takáč (arXiv:1212.0873; **parallelization**)
- Where does σ^2 come in?

$$\mathbf{E}[\delta_{[A]}^\top Q\delta_{[A]}] \le \frac{b}{n}\beta_b\|\delta\|^2 = \beta_b\,\mathbf{E}[\|\delta_{[A]}\|^2].$$

- Stepsize of $\beta = \beta_b$ might be too conservative.
- Might want to avoid calculating (or estimating) σ^2
- We suggest **Aggressive** variant of SDCA, where β is dynamically tuned to get

$$\mathbf{E}[\delta_{[A]}^\top Q\delta_{[A]}] \approx \beta\,\mathbf{E}[\|\delta_{[A]}\|^2]$$

Aggressive Mini-Batch for SDCA

- For $j \in A_t$, compute $\tilde{\delta}_j := \delta_j^{\beta^{(t)}}$
- $\|\delta_{[A]}\|^2 := \sum_{j\in A_t}\tilde{\delta}_j^2$
- $\|\delta_{[A]}^\top Q\delta_{[A]}\| := \|\sum_{j\in A_t}\tilde{\delta}_j y_j \mathbf{x}_j\|^2$
- Compute $\beta := \max(1, \frac{\|\delta_{[A]}^\top Q\delta_{[A]}\|}{\|\delta_{[A]}\|^2})$
- For $j \in A_t$, update using $\delta_j := \delta_j^\beta$
- If $\mathcal{D}(\alpha^{(t)} + \sum_{j\in A_t}\delta_j e_j) > \mathcal{D}(\alpha^{(t)})$
  $\alpha^{(t+1)} := \alpha^{(t)} + \sum_{j\in A_t}\delta_j e_j$
- Else
  $\alpha^{(t+1)} := \alpha^{(t)}$
- Set $\beta^{(t+1)} := (\beta^{(t)})^\gamma \beta^{1-\gamma}$ (γ controls update rate)

Experimental results