Data Sciences – CentraleSupelec
Advance Machine Learning
Course III - Stochastic approximation algorithms

Emilie Chouzenoux

Center for Visual Computing
CentraleSupelec
emilie.chouzenoux@centralesupelec.fr

Motivation

Linear regression/classification:

- Dataset with n entries: $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, $i = 1, \dots, n$
- Prediction of y as a linear model $\mathbf{x}^\top \boldsymbol{\beta}$
- Minimization of a penalized cost function:

$$(\forall \boldsymbol{\beta} \in \mathbb{R}^d) \quad F(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda R(\boldsymbol{\beta})$$

Examples of loss/regularizers:

- Quadratic loss: $\ell(y, x) = \frac{1}{2}(x - y)^2$
- Logistic loss: $\ell(y, x) = \log(1 + \exp(-yx))$
- Ridge penalty $R(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2$
- Lasso penalty $R(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1$

Motivation

Large n **- Small** d $\Rightarrow$ Minimization of F assuming that, at each iteration, only a **subset** of the data is available.

Loss for single observation:

$$(\forall \boldsymbol{\beta} \in \mathbb{R}^d) \quad f_i(\boldsymbol{\beta}) = \ell(y_i, \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda R(\boldsymbol{\beta})$$

so that $F = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{\beta})$.

Loss for a subset of observation: (mini-batch)

$$(\forall \boldsymbol{\beta} \in \mathbb{R}^d) \quad F_j(\boldsymbol{\beta}) = \sum_{i \in \mathcal{B}_j} \ell(y_i, \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda R(\boldsymbol{\beta})$$

with $(\mathcal{B}_j)_{1 \leq j \leq k}$ forming a partition of $\{1, \ldots, n\}$.

Stochastic gradient descent

We assume that F is differentiable on $\mathbb{R}^d$. For every $t \in \mathbb{N}$, we sample uniformly an index $i_t \in \{1, \ldots, n\}$ and update:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \gamma_t \nabla f_{i_t}(\boldsymbol{\beta}^{(t)})$$

- The randomly chosen gradient $\nabla f_{i_t}(\boldsymbol{\beta}^{(t)})$ yields an unbiased estimate of the true gradient $\nabla F(\boldsymbol{\beta}^{(t)})$
- $\gamma_t > 0$ is called the *stepsize or learning rate*. Its choice has an influence on the convergence properties of the algorithm. Typical choice: $\gamma_t = Ct^{-1}$.
- More stable results using *averaging*:

$$\overline{\boldsymbol{\beta}}^{(t)} = \frac{1}{t}\sum_{k=1}^{t} \boldsymbol{\beta}^{(k)} \Leftrightarrow \overline{\boldsymbol{\beta}}^{(t)} = (1 - \frac{1}{t})\overline{\boldsymbol{\beta}}^{(t-1)} + \frac{1}{t}\boldsymbol{\beta}^{(t)}$$

New choice: $\gamma_t = Ct^{-\alpha}$ with $\alpha \in [1/2, 1]$.

Accelerated variants

There are a large variety of approaches available to accelerate the convergence of SG methods. We list the most famous ones here:

- Momentum:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \gamma_t \nabla f_{i_t}(\boldsymbol{\beta}^{(t)}) + \theta_t(\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^{(t-1)})$$

- Gradient averaging: (see also SAG/SAGA)

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \frac{\gamma_t}{t} \sum_{k=1}^{t} \nabla f_{i_k}(\boldsymbol{\beta}^{(k)})$$

- ADAGRAD:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \gamma_t \mathbf{W}_t \nabla f_{i_t}(\boldsymbol{\beta}^{(t)})$$

  with a specific diagonal matrix $\mathbf{W}_t$ related to ℓ_2 norm of past gradients.