Optimization in Machine Learning

First order methods SGD Further Details



Learning goals

- Decreasing step size for SGD
- Stopping rules
- SGD with momentum



SGD WITH CONSTANT STEP SIZE

Example: SGD with constant step size.



× 0 0 × × ×

Fast convergence of SGD initially. Erratic behavior later (variance too big).

SGD WITH DECREASING STEP SIZE

- Idea: Decrease step size to reduce magnitude of erratic steps.
- Trade-off:
 - if step size $\alpha^{[t]}$ decreases slowly, large erratic steps
 - if step size decreases too fast, performance is impaired
- SGD converges for sufficiently smooth functions if

$$\frac{\sum_{t=1}^{\infty} \left(\alpha^{[t]}\right)^2}{\sum_{t=1}^{\infty} \alpha^{[t]}} = \mathbf{0}$$

("how much noise affects you" by "how far you can get").

× 0 0 × ×

SGD WITH DECREASING STEP SIZE / 2

• Popular solution: step size fulfilling $\alpha^{[t]} \in \mathcal{O}(1/t)$.



× × ×

Example continued. Step size $\alpha^{[t]} = 0.2/t$.

- Often not working well in practice: step size gets small quite fast.
- Alternative: $\alpha^{[t]} \in \mathcal{O}(1/\sqrt{t})$

ADVANCED STEP SIZE CONTROL

Why not Armijo-based step size control?

 Backtracking line search or other approaches based on Armijo rule usually not suitable: Armijo condition

$$g(\mathbf{x} + lpha \mathbf{d}) \leq g(\mathbf{x}) + \gamma_1 lpha
abla g(\mathbf{x})^{ op} \mathbf{d}$$

requires evaluating full gradient.

- But SGD is used to avoid expensive gradient computations.
- Research aims at finding inexact line search methods that provide better convergence behaviour, e.g., Vaswani et al., *Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates.* NeurIPS, 2019.

× × ×

MINI-BATCHES

• Reduce noise by increasing batch size *m* for better approximation

$$\hat{\mathbf{d}} = rac{1}{m}\sum_{i\in J}
abla_{\mathbf{x}} g_i(\mathbf{x}) pprox rac{1}{n}\sum_{i=1}^n
abla_{\mathbf{x}} g_i(\mathbf{x}) = \mathbf{d}$$

• Usually, the batch size is limited by computational resources (e.g., how much data you can load into the memory)



Example continued. Batch size m = 1 vs. m = 5.

 $\times \times$



STOPPING RULES FOR SGD

- For GD: We usually stop when gradient is close to 0 (i.e., we are close to a stationary point)
- For SGD: individual gradients do not necessarily go to zero, and we cannot access full gradient.
- Practicable solution for ML:
 - Measure the validation set error after T iterations
 - Stop if validation set error is not improving



SGD AND ML

General remarks:

- SGD is a variant of GD
- SGD particularly suitable for large-scale ML when evaluating gradient is too expensive / restricted by computational resources
- SGD and variants are the most commonly used methods in modern ML, for example:
 - Linear models

Note that even for the linear model and quadratic loss, where a closed form solution is available, SGD might be used if the size *n* of the dataset is too large and the design matrix does not fit into memory.

- Neural networks
- Support vector machines
- ...

× 0 0 × × ×

SGD WITH MOMENTUM

SGD is usually used with momentum due to reasons mentioned in previous chapters.

Algorithm Stochastic gradient descent with momentum

- 1: require step size α and momentum φ
- 2: **require** initial parameter \pmb{x} and initial velocity $\pmb{
 u}$
- 3: while stopping criterion not met do
- 4: Sample mini-batch of *m* examples
- 5: Compute gradient estimate $\nabla \hat{g}(\mathbf{x})$ using mini-batch
- 6: Compute velocity update: $\boldsymbol{\nu} \leftarrow \varphi \boldsymbol{\nu} \alpha \nabla \hat{g}(\mathbf{x})$
- 7: Apply update: $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}$
- 8: end while

× 0 0 × ×