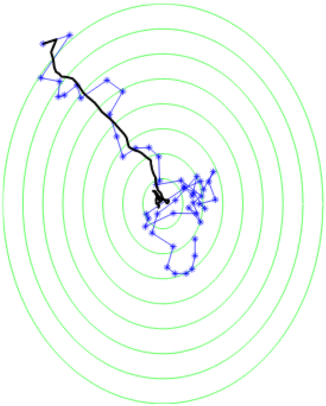
Optimization in Machine Learning

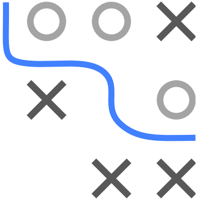
First order methods
SGD Further Details



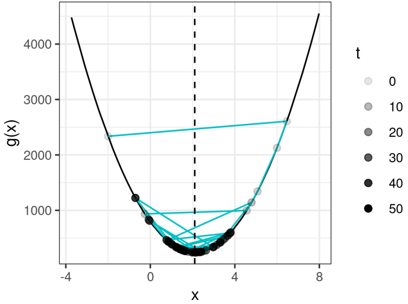
Learning goals

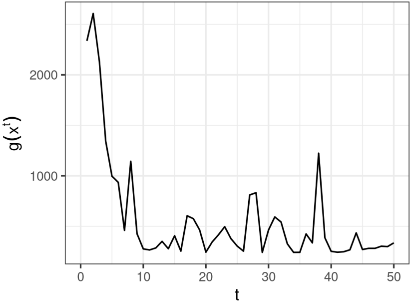
- Decreasing step size for SGD
- Stopping rules
- SGD with momentum

SGD WITH CONSTANT STEP SIZE



- Example: SGD with constant step size





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

SGD WITH DECREASING STEP SIZE

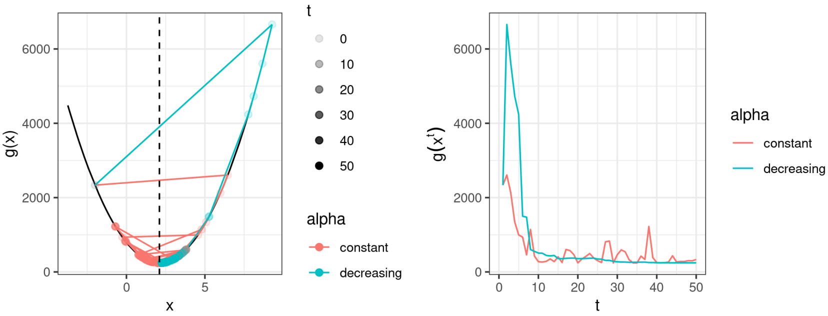
- Idea: Decrease step size to reduce magnitude of erratic steps
- Trade-off:
  - If step size $\alpha^{[t]}$ decreases slowly, large erratic steps
  - If step size decreases too fast, performance is impaired
- SGD converges for sufficiently smooth functions if

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

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

SGD WITH DECREASING STEP SIZE

- 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

$$g(\boldsymbol{x} + \alpha \mathbf{d}) \leq g(\boldsymbol{x}) + \gamma_1 \alpha \nabla g(\boldsymbol{x})^\top \mathbf{d}$$

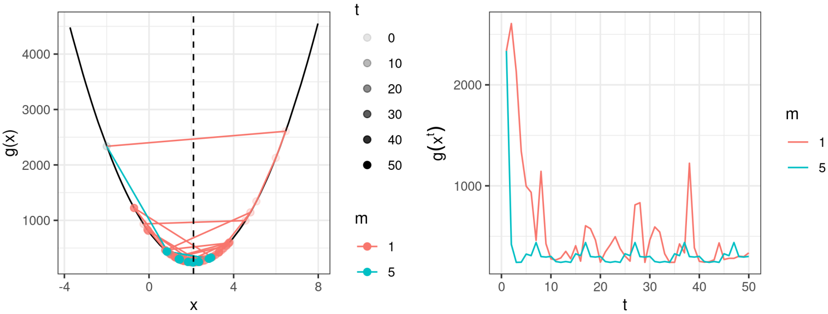
  requires evaluating full gradient
- But SGD is used to *avoid* expensive gradient computations
- Research aims at finding inexact line search methods that provide better convergence behaviour, e.g., Vaswani et al., *Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates*. NeurIPS,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

MINI-BATCHES

- Reduce noise by increasing batch size m for better approximation

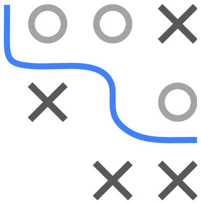
$$\hat{\mathbf{d}} = \frac{1}{m} \sum_{i \in J} \nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) = \mathbf{d}$$

- 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. $m = 5$

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
  - ...

SGD WITH MOMENTUM

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

Algorithm Stochastic gradient descent with momentum

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