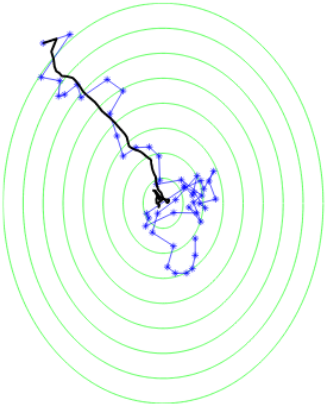
Optimization in Machine Learning

First order methods: SGD Further Details

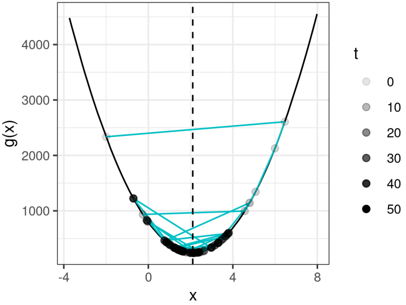


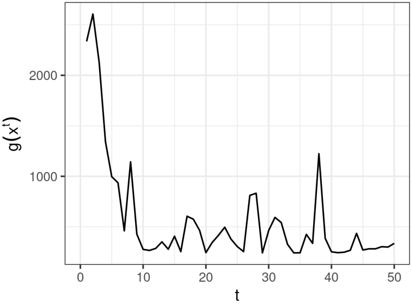
Learning goals

- 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 at beginning, erratic behavior later on (variance too big).

SGD WITH DECREASING STEP-SIZE

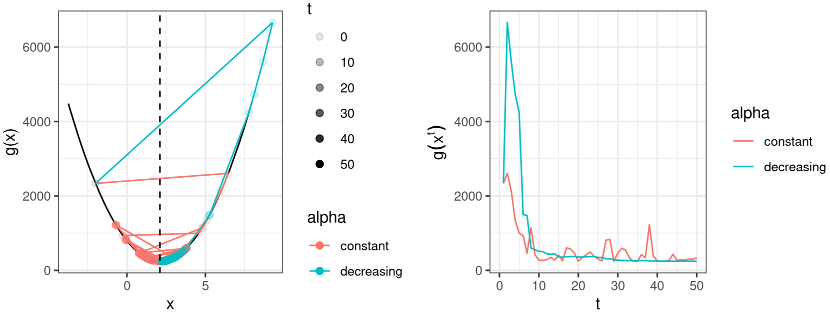
- **Idea:** Decrease step size to reduce magnitude of steps of erratic behavior.
- Consider trade-off:
  - if step size $\alpha^{[t]}$ decreases slowly, the variance of $\nabla_x \, g_i(\mathbf{x})$ decreases slowly too.
  - if step size decreases too fast, we reach optimum slowly.
- SGD converges to stationary point if ratio of sum of squared step-sizes over the sum of step-sizes converges to 0:

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

(“how much noise affects you” vs. “how far you can get”).

SGD WITH DECREASING STEP-SIZE

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



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

- Often not working well in practice: Step size really small really fast.
- Alternative: $\alpha^{[t]} \in \mathcal{O}(\sqrt{t})$

ADVANCED STEP-SIZE CONTROL

Why not Armijo-based step-size control?

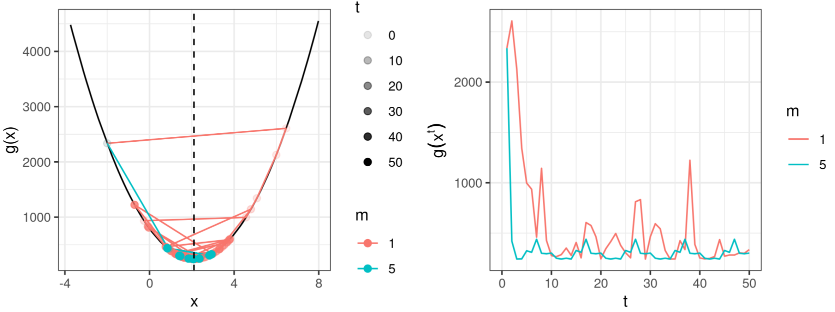
- Backtracking line search or other rules based on the Armijo-condition are usually not suitable: A check of Armijo condition
$$f(\mathbf{x} + \alpha \boldsymbol{d}) \leq f(\mathbf{x}) + \gamma\alpha \nabla g(\mathbf{x})^\top \boldsymbol{d}$$
requires evaluating the full gradient.
- But: SGD is particularly used to **avoid** expensive evaluations of gradient!
- Recent research aims at finding approximate line-search methods that provide better convergence methods, 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

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

- 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 simplification of GD
- SGD particularly suitable for large-scale ML when the evaluating the 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 1 Stochastic gradient descent with momentum

require learning rate α and momentum φ
require initial parameter $\boldsymbol{x}$ and initial velocity $\boldsymbol{\nu}$
while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\tilde{x}^{(1)}, \ldots, \tilde{x}^{(m)}\}$
    Compute gradient estimate: $\nabla \hat{f}(\boldsymbol{x})$
    Compute velocity update: $\boldsymbol{\nu} \leftarrow \varphi\boldsymbol{\nu} - \alpha\nabla \hat{f}(\boldsymbol{x})$
    Apply update: $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\nu}$
end while