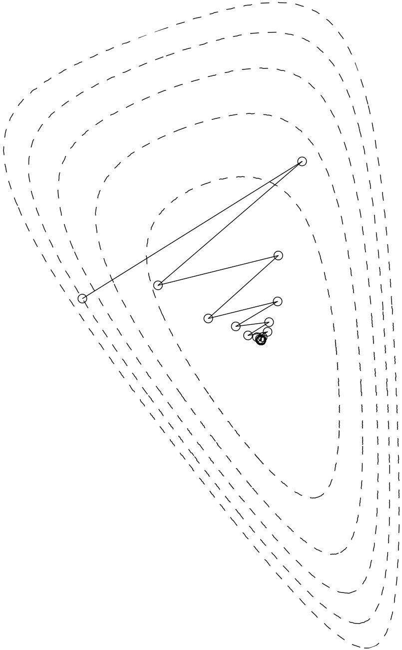

Optimization Algorithms

Stochastic Gradient Descent

Marc Toussaint
Technical University of Berlin

Winter 2024/25

References

- Léon Bottou: Stochastic Gradient Descent Tricks! (2012)
- Bottou, Curtis, Nocedal: Optimization Methods for Large-Scale Machine Learning (2018)
- Lecture by Mark Schmidt “SGD Convergence Rate”
- Nemirovski et al: Robust Stochastic Approximation Approach to Stochastic (2009)
- Lecture by Christopher De Sa https://www.cs.cornell.edu/courses/cs4787/2020sp/
- Wikipedia “Stochastic approximation”

- For consistency with references, we change our notation a bit:

- We consider the problem

$$\min_{w \in \mathbb{R}^d} f(w)$$

Stochastic Gradient Descent Basics & Convergence

Plain Gradient Descent – Recall

- Plain gradient descent iterates, e.g. with constant α

$$w \leftarrow w - \alpha \nabla f(w)$$

- Core issue (cf. Part 1): Stepsize! (e.g., small gradient $\to$ small step?)

- Solution: Backtracking line search
  - Theorem: Gradient descent with backtracking line search converges exponentially with convergence rate $\gamma = (1 - 2\frac{m}{M}\varrho_{\text{ls}}\varrho_\alpha^-)$
  - we have regret $O(\gamma^t)$ for some $\gamma < 1$

Typical Setting for Stochastic Gradient Descent

- Additive cost function:

$$\min_w \frac{1}{n} \sum_{i=1}^n f_i(w)$$

  – E.g.: least squares problem $\min_w \sum_{i=1}^n \phi_i(w)^2$

- Core example: Machine Learning, with data $D = \{(x_i, y_i)\}_{i=1}^n$

$$f(w) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i; w), y_i) + \frac{\lambda}{2} \|w\|^2$$

- We are interested in large n (big data)

Stochastic Gradient Descent (SGD)

- Instead of computing ∇f in each iteration, we only compute ∇f_i of *one* cost component
  - E.g., only the gradient w.r.t. a mini-batch (subset) of the full data

- Stochastic Gradient Descent:

Input: initial $w_0 \in \mathbb{R}^n$, gradient functions $\nabla f_i(w)$, stepsize schedule α_k

for $k = 0, .., $ **do**

  Sample i uniformly (iid) from $\{1, .., n\}$

  $w_{k+1} \leftarrow w_k - \alpha_k \nabla f_i(w_k)$

end for

- $\nabla f_i(w)$ has expectation $\mathbb{E}\{\nabla f_i(w)\} = \nabla f(w)$

Stochastic Gradient Descent (SGD)

- Instead of computing ∇f in each iteration, we only compute ∇f_i of *one* cost component
  - E.g., only the gradient w.r.t. a mini-batch (subset) of the full data

- Stochastic Gradient Descent (episodic):

Input: initial $w_0 \in \mathbb{R}^n$, gradient functions $\nabla f_i(w)$, stepsize schedule α_k,

```
initialize k = 0
for episode j = 0, .., do
    for i = 1, .., n (or i = RandomPermutation({1, .., n})) do
        w_{k+1} ← w_k − α_k ∇f_i(w_k)
        k ← k + 1
    end for
end for
```

- $\nabla f_i(w)$ has expectation $\mathbb{E}\{\nabla f_i(w)\} = \nabla f(w)$

Converenge of SGD

- SGD is a method to find a point such that $\nabla f(w) \approx 0$
- Convergence analysis investigates how $\|\nabla f(w_k)\|$ decreases with k (in expectation)

Converenge of SGD

- SGD is a method to find a point such that $\nabla f(w) \approx 0$
- Convergence analysis investigates how $\|\nabla f(w_k)\|$ decreases with k (in expectation)

- Mathematics: see "Stochastic Approximation"
- Typical assumptions:
  - Lipschitz continuity of $\nabla f(w)$:

$$\exists L \in \mathbb{R} \text{ s.t. } \forall w, \bar{w} : \ \|\nabla f(w) - \nabla f(\bar{w})\| \le L \|w - \bar{w}\| ,$$

    where $\|w\| = \sqrt{w^2}$ is the L_2-norm; L is called Lipschitz constant.
  - This means, "the change of gradient $\nabla f(w)$ is limited"

Convergence of SGD

- **Theorem**: Assuming $\nabla f(w)$ is L-continuous, and Var$\{\nabla f_i(w)\} = \sigma^2$, we have

$$\min_k \{\mathbb{E}\{\|\nabla f(w_k)\|^2\}\} \le \frac{f(w_0) - f^*}{\sum_{k=0}^{t\text{-}1} \alpha_k} + \frac{\sum_{k=0}^{t\text{-}1} \alpha_k^2}{\sum_{k=0}^{t\text{-}1} \alpha_k} \frac{L\sigma^2}{2}$$

Convergence of SGD

- **Theorem**: Assuming $\nabla f(w)$ is L-continuous, and Var$\{\nabla f_i(w)\} = \sigma^2$, we have

$$\min_k \{\mathbb{E}\{\|\nabla f(w_k)\|^2\}\} \le \frac{f(w_0) - f^*}{\sum_{k=0}^{t\text{-}1} \alpha_k} + \frac{\sum_{k=0}^{t\text{-}1} \alpha_k^2}{\sum_{k=0}^{t\text{-}1} \alpha_k} \frac{L\sigma^2}{2}$$

- Implications:
  - If gradient had no noise $\sigma = 0$ (plain GD): constant α leads to convergence $O(1/t)$

Convergence of SGD

- **Theorem**: Assuming $\nabla f(w)$ is L-continuous, and Var$\{\nabla f_i(w)\} = \sigma^2$, we have

$$\min_k \{\mathbb{E}\{\|\nabla f(w_k)\|^2\}\} \le \frac{f(w_0) - f^*}{\sum_{k=0}^{t\text{-}1} \alpha_k} + \frac{\sum_{k=0}^{t\text{-}1} \alpha_k^2}{\sum_{k=0}^{t\text{-}1} \alpha_k} \ \frac{L\sigma^2}{2}$$

- Implications:
  - If gradient had no noise $\sigma = 0$ (plain GD): constant α leads to convergence $O(1/t)$
  - Stochasticity: rate is determined by $\frac{\sum_{k=0}^{t\text{-}1} \alpha_k^2}{\sum_{k=0}^{t\text{-}1} \alpha_k}$. Ensure $\lim_t \sum_{k=0}^{t\text{-}1} \alpha_k^2 < \infty$ and $\lim_t \sum_{k=0}^{t\text{-}1} \alpha_k = \infty$.
  - Constant α is bad choice: right becomes a constant $\frac{\alpha L \sigma^2}{2}$
  - Diminishing step size $\alpha_k = \frac{\alpha_0}{1+\gamma k}$ is good: we have $\sum_k \alpha_k = O(\log t)$ and error $O(1/\log(t))$

Converenge of SGD – Derivation

- Based on assuming Lipschitz continuity of $\nabla f(w)$, we derive how SGD decreases function values in expectation:
  - We assume $\|\nabla f(w) - \nabla f(\bar{w})\| \le L \, \|w - \bar{w}\|$ for any $w, \bar{w}$.
  - For any step $\delta = w - \bar{w}$ the Hessian $\nabla^2 f(w)$ fulfills $\|\nabla^2 f(w)\delta\| \le L\|\delta\|$.
  - Using this in a 2nd order Taylor, it follows

$$f(w) \le f(\bar{w}) + \nabla f(\bar{w})^\top (w - \bar{w}) + \frac{1}{2} \, L(w - \bar{w})^2 \ .$$

  - And applying this to $w_{k+1} \leftarrow w_k - \alpha_k \nabla f_i(w_k)$, we get in expectation

$$\mathbb{E}\{f(w_{k+1})\} \le f(w_k) - \alpha_k \|\nabla f(w_k)\|^2 + \frac{1}{2} \, \alpha_k^2 \, L \, \mathbb{E}\{\|\nabla f_i(w_k)\|^2\} \ .$$

Converenge of SGD – Derivation

- From this, we derive how $\|\nabla f(w_k)\|$ decreases with k in expectation:
  - Assume σ^2 is the variance of $\nabla f_i(w)$, and rearrange terms

$$\begin{aligned}
\mathbb{E}\{f(w_{k+1})\} &\leq f(w_k) - \alpha_k \|\nabla f(w_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}\{\|\nabla f_i(w_k)\|^2\} \\
&\leq f(w_k) - \alpha_k \|\nabla f(w_k)\|^2 + \alpha_k^2 \frac{L\sigma^2}{2} \\
\alpha_k \|\nabla f(w_k)\|^2 &\leq f(w_k) - \mathbb{E}\{f(w_{k+1})\} + \alpha_k^2 \frac{L\sigma^2}{2}
\end{aligned}$$

  - Sum over $k = 1, ..t$, pull min. gradient out of left sum, and notice the telescope sum on the right:

$$\sum_{k=1}^{t} \alpha_{k\text{-}1} \|\nabla f(w_k)\|^2 \leq \sum_{k=1}^{t} [f(w_{k\text{-}1}) - \mathbb{E}\{f(w_k)\}] + \sum_{k=1}^{t} \alpha_{k\text{-}1}^2 \frac{L\sigma^2}{2}$$

$$\min_k \{\mathbb{E}\{\|\nabla f(w_k)\|^2\}\} \sum_{k=1}^{t} \alpha_{k\text{-}1} \leq f(w_0) - \mathbb{E}\{f(w_t)\} + \sum_{k=1}^{t} \alpha_k^2 \frac{L\sigma^2}{2}$$

  - Replace $\mathbb{E}\{f(w_t)\} \geq f^*$, and rearrange terms:

$$\min_k \{\mathbb{E}\{\|\nabla f(w_k)\|^2\}\} \leq \frac{f(w_0) - f^*}{\sum_{k=0}^{t\text{-}1} \alpha_k} + \frac{\sum_{k=0}^{t\text{-}1} \alpha_k^2}{\sum_{k=0}^{t\text{-}1} \alpha_k} \frac{L\sigma^2}{2}$$

When is SGD efficient?

(from Bottou "tricks")

- For strongly convex assumptions, deterministic gradient can converge exponentially, requiring $O(\log \frac{1}{\rho})$ iterations to reach precision ρ. SGD requires $O(\frac{1}{\rho})$ iterations.

- HOWEVER: The time-per-iteration is also important!: (see 3rd line)

	GD	**2GD**	**SGD**	**2SGD**
Time per iteration :	n	n	1	1
Iterations to accuracy ρ :	$\log \frac{1}{\rho}$	$\log \log \frac{1}{\rho}$	$1/\rho$	$1/\rho$
Time to accuracy ρ :	$n \log \frac{1}{\rho}$	$n \log \log \frac{1}{\rho}$	$1/\rho$	$1/\rho$
Time to excess error ε :	$\frac{1}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon}$	$1/\varepsilon$	$1/\varepsilon$

2GD = "2nd order gradient method" (that uses some approx. of the inv. Hessian)

$\rightarrow$ **for large n, SGD is faster!**

Practical Recommendations

(from Bottou "tricks")

- Randomly shuffle i, but then 'zip' through

Practical Recommendations

(from Bottou "tricks")

- Randomly shuffle i, but then 'zip' through
- In ML: Monitor training and validation after each zip, through full data

Practical Recommendations

(from Bottou "tricks")

- Randomly shuffle i, but then 'zip' through
- In ML: Monitor training and validation after each zip, through full data
- Use learning rate $\alpha_k = \frac{\alpha_0}{1+\alpha_0 \lambda k}$, when λ is a known minimal eigenvalue of Hessian (e.g., L_2-regularization in ML)

Practical Recommendations

(from Bottou "tricks")

- Randomly shuffle i, but then 'zip' through
- In ML: Monitor training and validation after each zip, through full data
- Use learning rate $\alpha_k = \frac{\alpha_0}{1+\alpha_0 \lambda k}$, when λ is a known minimal eigenvalue of Hessian (e.g., L_2-regularization in ML)
- Empirically choose best α_0 on small data subset

How to improve Stochastic Gradient Descent

How to improve over basic SGD?

- There are three core approaches:

- Gradient Variance Reduction
- 2nd-order information
- Momemtum Methods

Reducing Gradient Variance

- Use flexible mini-batch sizes,

$$w_{k+1} \leftarrow w_k + \frac{\alpha_k}{|B_k|} \sum_{i \in B_k} \nabla f_i(w_k)$$

  and increase $|B_k|$ over time. But how? (cf. Bottou et al. Sec 5.2)

Reducing Gradient Variance

- Use flexible mini-batch sizes,

$$w_{k+1} \leftarrow w_k + \frac{\alpha_k}{|B_k|} \sum_{i \in B_k} \nabla f_i(w_k)$$

  and increase $|B_k|$ over time. But how? (cf. Bottou et al. Sec 5.2)

- Gradient aggregation: E.g., store *all* gradients $\nabla f_j(w_{[j]})$ you've seen latest for j, then sample i, update $w_{[i]} \leftarrow w_k$, query&store $\nabla f_i(w_k)$ and iterate (Bottou Sec 5.3.2)

$$w_{k+1} \leftarrow w_k + (1/n) \sum_{j=1}^{n} \nabla f_j(w_{[j]})$$

Reducing Gradient Variance

- Use flexible mini-batch sizes,

$$w_{k+1} \leftarrow w_k + \frac{\alpha_k}{|B_k|} \sum_{i \in B_k} \nabla f_i(w_k)$$

  and increase $|B_k|$ over time. But how? (cf. Bottou et al. Sec 5.2)

- Gradient aggregation: E.g., store *all* gradients $\nabla f_j(w_{[j]})$ you've seen latest for j, then sample i, update $w_{[i]} \leftarrow w_k$, query&store $\nabla f_i(w_k)$ and iterate (Bottou Sec 5.3.2)

$$w_{k+1} \leftarrow w_k + (1/n) \sum_{j=1}^{n} \nabla f_j(w_{[j]})$$

- *Iterate* Averaging: Let w_k create "noise", but care about $\bar w_t = \frac{1}{t-k} \sum_{k'=k}^{t} w_{k'}$.
  (Polyak-Ruppert method)

Second Order Information

- Try to estimate Hessian, e.g. stochastic version of BFGS
  - Many possible approaches & maths, Sec 6
  - But require more complex operations that plain SG

Second Order Information

- Try to estimate Hessian, e.g. stochastic version of BFGS
  - Many possible approaches & maths, Sec 6
  - But require more complex operations that plain SG

$\rightarrow$ Estimate diagonal of Hessian, or “scaling” of gradient only **coordinate-wise**

Second Order Information

- Try to estimate Hessian, e.g. stochastic version of BFGS
  - Many possible approaches & maths, Sec 6
  - But require more complex operations that plain SG

$\rightarrow$ Estimate diagonal of Hessian, or "scaling" of gradient only **coordinate-wise**

- **RMSprop** (running avg. of elem-wise gradient squares)

$$v_k \leftarrow (1-\lambda)\ v_{k\text{-}1} + \lambda\ [\nabla f_i(w_k)]^2 \quad \text{[elem-wise]}$$
$$w_{k+1} \leftarrow w_k - \frac{\alpha_k}{\sqrt{v_k + \mu}} \nabla f_i(w_k) \quad \text{[elem-wise]}$$

- **Adagrad** (accumulate squares for diminishing stepsize with constant α)

$$v_k \leftarrow v_{k\text{-}1} + [\nabla f_i(w_k)]^2 \quad \text{[elem-wise]}$$
$$w_{k+1} \leftarrow w_k - \frac{\alpha}{\sqrt{v_k + \mu}} \nabla f_i(w_k) \quad \text{[elem-wise]}$$

("Theoretical explaination for good performance pending"; Bottou et al, Sec 6.5)

Why divide by $\sqrt{\langle g^2 \rangle}$?

- RMSprop makes a step $-\frac{\alpha_k}{\sqrt{\langle g^2 \rangle + \mu}} \nabla f_i$ (elem-wise), where $\langle g^2 \rangle$ averages gradient squares (elem-wise) – Why?

Why divide by $\sqrt{\langle g^2 \rangle}$?

- RMSprop makes a step $-\frac{\alpha_k}{\sqrt{\langle g^2 \rangle + \mu}} \nabla f_i$ (elem-wise), where $\langle g^2 \rangle$ averages gradient squares (elem-wise) – Why?

- Scale invariance: Rescaling $f_i \leftarrow a f_i$ scales $\nabla_i f$ and $\sqrt{\langle g^2 \rangle}$ equally
- Accounts for different conditioning along different coordinates
- Gradient steps in all directions become somewhat equal/normalized

Why divide by $\sqrt{\langle g^2 \rangle}$?

- RMSprop makes a step $-\frac{\alpha_k}{\sqrt{\langle g^2 \rangle + \mu}} \nabla f_i$ (elem-wise), where $\langle g^2 \rangle$ averages gradient squares (elem-wise) – Why?

- Scale invariance: Rescaling $f_i \leftarrow a f_i$ scales $\nabla_i f$ and $\sqrt{\langle g^2 \rangle}$ equally
- Accounts for different conditioning along different coordinates
- Gradient steps in all directions become somewhat equal/normalized

- If f_i has some curvature, e.g. $f_i = a w^2$, then $\nabla f_i = 2aw$, and $\sqrt{\langle g^2 \rangle} \propto a$
- $\sqrt{\langle g^2 \rangle}$ is proportional to curvature, and mimics a diagonal Hessian

SGD with Momentum

- SGD with momentum: (c.f. conjugate gradient method)

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla f_i(w_k) + \beta_k (w_k - w_{k\text{-}1})$$

Written as low-pass of the adaptation step ($m_k = w_{k+1} - w_k$):

$$m_k \leftarrow \beta_k m_{k\text{-}1} - \alpha_k \nabla f_i(w_k) \ , \quad w_{k+1} \leftarrow w_k + m_k$$

Recommended version, easier to tune with constant beta β and decay $\alpha_k = \alpha_0/(1+\lambda k)$:

$$m_k \leftarrow \beta \, m_{k\text{-}1} - (1-\beta) \, \alpha_k \nabla f_i(w_k) \ , \quad w_{k+1} \leftarrow w_k + m_k$$

SGD with Momentum

- SGD with momentum: (c.f. conjugate gradient method)

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla f_i(w_k) + \beta_k (w_k - w_{k\text{-}1})$$

Written as low-pass of the adaptation step ($m_k = w_{k+1} - w_k$):

$$m_k \leftarrow \beta_k m_{k\text{-}1} - \alpha_k \nabla f_i(w_k) \;, \quad w_{k+1} \leftarrow w_k + m_k$$

Recommended version, easier to tune with constant beta β and decay $\alpha_k = \alpha_0/(1 + \lambda k)$:

$$m_k \leftarrow \beta \, m_{k\text{-}1} - (1-\beta) \, \alpha_k \nabla f_i(w_k) \;, \quad w_{k+1} \leftarrow w_k + m_k$$

- Nesterov Accelerated Gradient ("Nesterov Momentum"):

$$\tilde{w}_k \leftarrow w_k + \beta_k (w_k - w_{k\text{-}1})$$
$$w_{k+1} \leftarrow \tilde{w}_k - \alpha_k \nabla f_i(\tilde{w}_k)$$

Yurii Nesterov (1983): *A method for solving the convex programming problm with convergence rate* $O(1/k^2)$

Adam

- Adam: A Method for Stochastic Optimization (DP. Kingma, J. Ba) arXiv:1412.6980

  "Our method is designed to combine the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gra- dients, and RMSProp (Tieleman & Hinton, 2012), which works well in on-line and non-stationary settings"

  (Roughly, Adam = cleaner version of RMSprop with momentum.)

- Prove convergence rate

$$\frac{1}{T}\sum_{k=1}^{T}[f(w_k) - f(w^*)] \le O(1/T)$$

Adam

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

(all operations interpreted element-wise)

arXiv:1412.6980

Adam & Nadam

- Adam interpretations (everything element-wise!):
  - $m_t \approx \langle g \rangle$ the mean gradient in the recent iterations
  - $v_t \approx \langle g^2 \rangle$ the mean gradient-square in the recent iterations
  - $\hat{m}_t, \hat{v}_t$ are bias corrected (check: in first iteration, $t = 1$, we have $\hat{m}_t = g_t$, unbiased, as desired)
  - $\Delta\theta \approx -\frac{\alpha}{\sqrt{\langle g^2 \rangle}}\, g$ *would* be a Newton step if $\sqrt{\langle g^2 \rangle}$ *were* the Hessian...

Adam & Nadam

- Adam interpretations (everything element-wise!):
  - $m_t \approx \langle g \rangle$ the mean gradient in the recent iterations
  - $v_t \approx \langle g^2 \rangle$ the mean gradient-square in the recent iterations
  - $\hat{m}_t, \hat{v}_t$ are bias corrected (check: in first iteration, $t = 1$, we have $\hat{m}_t = g_t$, unbiased, as desired)
  - $\Delta\theta \approx -\frac{\alpha}{\sqrt{\langle g^2 \rangle}}\, g$ *would* be a Newton step if $\sqrt{\langle g^2 \rangle}$ *were* the Hessian...

- Incorporate Nesterov into Adam: Replace parameter update by

$$\theta_t \leftarrow \theta_{t\text{-}1} - \alpha/(\sqrt{\hat{v}_t} + \epsilon) \cdot (\beta_1 \hat{m}_t + \frac{(1-\beta_1) g_t}{1 - \beta_1^t})$$

Dozat: *Incorporating Nesterov Momentum into Adam*, ICLR'16

Appendix: Convergence & Convergence Rate

- Convergence: $\lim x_k = x^* \quad \Leftrightarrow \quad \forall \epsilon > 0 : \; \exists K : \; \forall k > k : \; |x_k - x^*| \le \epsilon$
- Convergence Rate: $\lim_{k\to\infty} \frac{x_{k+1} - x^*}{x_k - x^*} = \mu$
- We care about convergence of the gradient $\lim_{k\to\infty} |g_k| = 0$ to zero

Appendix: Convergence & Convergence Rate

- Convergence: $\lim x_k = x^* \quad \Leftrightarrow \quad \forall \epsilon > 0 : \; \exists K : \; \forall k > k : \; |x_k - x^*| \le \epsilon$
- Convergence Rate: $\lim_{k\to\infty} \frac{x_{k+1} - x^*}{x_k - x^*} = \mu$
- We care about convergence of the gradient $\lim_{k\to\infty} |g_k| = 0$ to zero

- Typically you try to prove a step-wise decrease inequality, e.g.:

$$|g_{k+1}| \le \mu \, |g_k|$$

  We call this "convergence with rate μ", which is also called linear convergence ("convergence with linear step-wise reduction") or exponential convergence, as we have $|g_k| \le O(\mu^k)$.

- Or one directly finds a converging upper bound, e.g.

$$|g_k| \le O(1/k)$$

  We call this "converges to zero with 1/k", but not with a constant ("linear") rate, but slower.