Stochastic gradient descent

15.093: Optimization

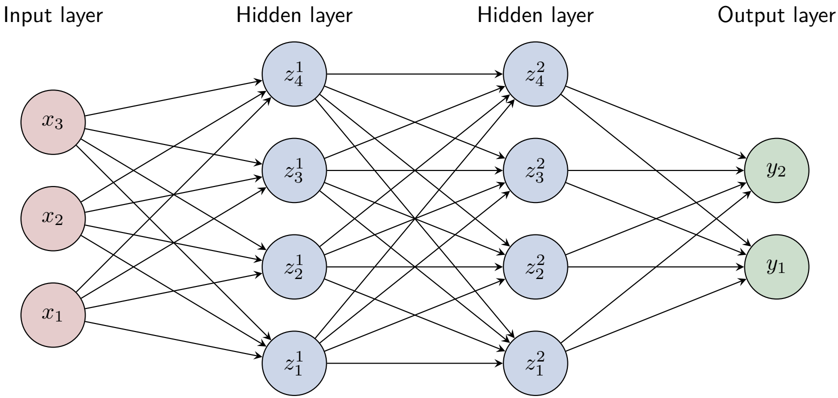
Dimitris Bertsimas
Alexandre Jacquillat

Sloan School of Management
Massachusetts Institute of Technology

Motivation: Fitting a neural network

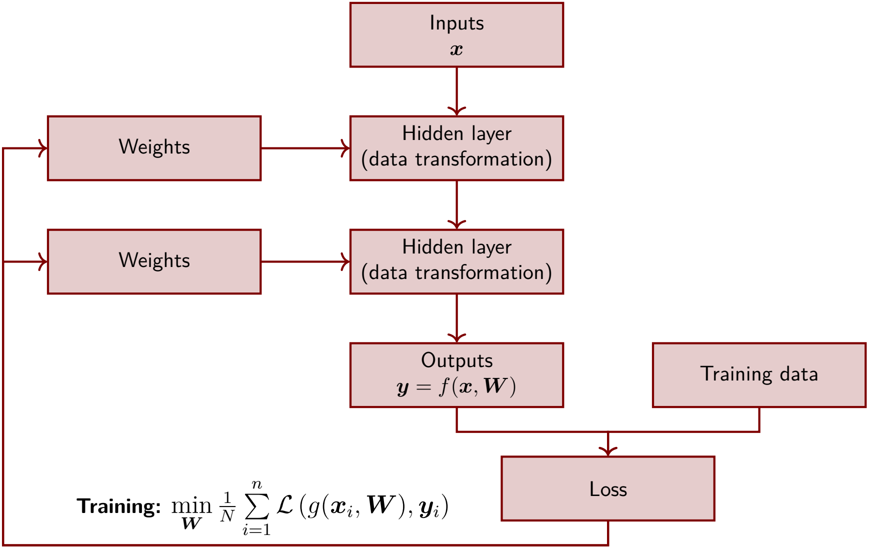
Neural network

- Multi-layer neural networks to build non-linear functions: $\boldsymbol{y} = g(\boldsymbol{x})$



$$z_k^1 = h_k^1\left(\sum_{j=1}^{3} w_{jk}x_j + b_k\right) \quad z_\ell^2 = h_\ell^2\left(\sum_{k=1}^{4} w_{k\ell}z_k^1 + b_\ell\right) \quad y_m = h_m^3\left(\sum_{\ell=1}^{4} w_{\ell\ell}z_\ell^2 + b_m\right)$$

Fitting a neural network: overview



Fitting a neural network: backpropagation

- Fitting a deep learning model involves non-convex optimization

$$\min f(\boldsymbol{W}) = \frac{1}{N} \sum_{i=1}^{n} \mathcal{L}\left(g(\boldsymbol{x}_i, \boldsymbol{W}), \boldsymbol{y}_i\right)$$

- Gradient descent iterates via $\boldsymbol{W}^{k+1} = \boldsymbol{W}^k - \alpha_k \nabla f(\boldsymbol{W}^k)$

→ Forward propagation: evaluating the function with weights $\boldsymbol{W}^k$

$$\begin{tikzcd}[column sep=huge] \boxed{\boldsymbol{x}} \arrow[r, "\boldsymbol{w}^{(1)}"] & \boxed{\boldsymbol{z}^1} \arrow[r, "\boldsymbol{w}^{(2)}"] & \boxed{\boldsymbol{z}^2} \arrow[r, "\boldsymbol{w}^{(3)}"] & \boxed{\boldsymbol{y}} \end{tikzcd}$$

$$\boldsymbol{z}^1 = \boldsymbol{h}^1(\boldsymbol{x}, \boldsymbol{w}^{(1)}) \qquad \boldsymbol{z}^2 = \boldsymbol{h}^2(\boldsymbol{x}, \boldsymbol{w}^{(2)}) \qquad \boldsymbol{z}^3 = \boldsymbol{h}^3(\boldsymbol{x}, \boldsymbol{w}^{(3)})$$

→ Backward propagation: computing the gradient of the function in $\boldsymbol{W}^k$

$$\begin{tikzcd}[column sep=huge] \boxed{\boldsymbol{x}} & \boxed{\boldsymbol{z}^1} \arrow[l, "\boldsymbol{w}^{(1)}"'] & \boxed{\boldsymbol{z}^2} \arrow[l, "\boldsymbol{w}^{(2)}"'] & \boxed{\boldsymbol{y}} \arrow[l, "\boldsymbol{w}^{(3)}"'] \end{tikzcd}$$

$$\nabla f_{\boldsymbol{w}^{(1)}} \qquad \nabla f_{\boldsymbol{w}^{(2)}} \qquad \nabla f_{\boldsymbol{w}^{(3)}}$$

Toward stochastic gradient descent (SGD)

- Fitting a deep learning model could be achieved with gradient descent, Newton's method, etc.
- Two main computational challenges with gradient descent
  1. No guarantee of global convergence in non-convex optimization
  2. Evaluating the gradient at each iteration can be computationally expensive (e.g., backpropagation in neural networks)
- Stochastic gradient descent aims to circumvent these challenges
  1. Stochastic gradient descent can avoid local minima via randomization (still, it does not guarantee convergence to a global minimum)
  2. By approximating the gradient on a random data point, stochastic gradient descent proceeds much faster at each iteration
- Weaker theoretical results than gradient descent: slower convergence
- Strong practical performance: scalability to very complex machine learning models involving large-scale, non-convex optimization

Stochastic gradient descent

Setting and principles

- Non-linear optimization problem with a separable objective function:

$$\min \quad f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

- The gradient descent algorithm would iterate as follows:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k \nabla f(\boldsymbol{x}) = \boldsymbol{x}^k - \alpha_k \cdot \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^k)$$

- Challenge: computing the gradient $\nabla f(\boldsymbol{x})$ can be very difficult
    - Neural network: $\mathcal{O}(nm^2)$ operations with n data points and m nodes [e.g., $\mathcal{O}(10^{13})$ operations with 10 million data points and 1,000 nodes]

→ Stochastic gradient descent: approximating $\nabla f(\boldsymbol{x})$ with $\nabla f_{i^j}(\boldsymbol{x})$

- Neural network: $\mathcal{O}(m^2)$ operations per iteration with m nodes [e.g., $\mathcal{O}(10^6)$ operations with 10 million data points and 1,000 nodes]

$$\nabla f(\boldsymbol{x}) \approx \nabla f_{i^j}(\boldsymbol{x}) \implies \boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k \cdot \nabla f_{i^k}(\boldsymbol{x}^k)$$

Stochastic gradient descent algorithm

Algorithm

1. *Initialization: starting point* $\boldsymbol{x}^0 \in \mathbb{R}^n$*, and iteration counter* $k = 0$
2. *Repeat, until stopping criterion is reached*
   2.1 *Update iteration counter:* $k \leftarrow k + 1$
   2.2 *Choose index* $i^k \in \{1, \cdots, n\}$
   2.3 *Choose descent direction* $\boldsymbol{d}^k = -\nabla f_i(\boldsymbol{x}^k)$
   2.4 *Determine a step size* $\alpha_k > 0$
   2.5 *Update* $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k$

- How to choose the index i^k at each iteration?
  - Randomized rule: choose $i^k \in \{1, \cdots, n\}$ uniformly at random, which yields an unbiased estimate of the gradient conditionally on $\boldsymbol{x}$:

$$\mathbb{E}(\nabla f_{i^k}(\boldsymbol{x})) = \sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}) \mathbb{P}(i^k = i) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$$

  - Cyclic rule: choose $i^k = 1, 2, \cdots, n, 1, 2, \cdots, n, \cdots$

Illustration

Linear regression

- Linear regression model:

$$\min f(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\beta}) \text{ with: } f_i(\boldsymbol{\beta}) = \frac{1}{2}\left(y_i - \sum_{j=1}^{m} x_{ij}\beta_j\right)^2$$

- Differentiation of the loss function for a given point $i \in \{1, \cdots, n\}$

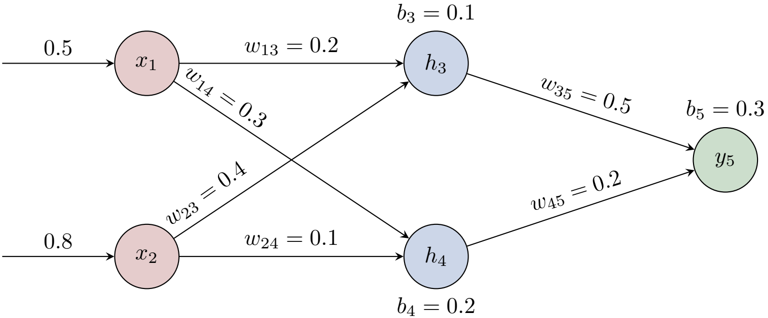
$$\frac{\partial f_i}{\partial \beta_j} = -x_{ij}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})$$
$$\implies \nabla f_i(\boldsymbol{\beta}) = -\boldsymbol{x}_i^\top (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})$$

→ Stochastic gradient descent algorithm with learning rate α:

$$\beta_j \leftarrow \beta_j + \alpha \cdot x_{ij}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}),\ \forall j = 1, \cdots, m$$

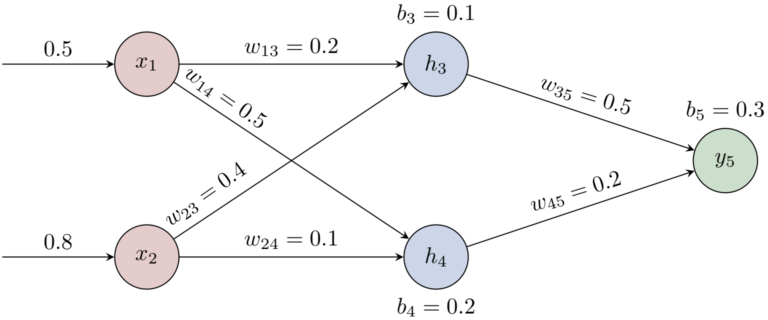
- Interpretation:
  - If $\boldsymbol{x}_i^\top \boldsymbol{\beta} < y_i$ and $x_{ij} > 0$, β_j increases to increase the prediction $\boldsymbol{x}_i^\top \boldsymbol{\beta}$
  - If $\boldsymbol{x}_i^\top \boldsymbol{\beta} > y_i$ and $x_{ij} > 0$, β_j decreases to decrease the prediction $\boldsymbol{x}_i^\top \boldsymbol{\beta}$

Neural network: construction



- Learning problem: two-dimensional inputs, one-dimensional output
- Neural network: 1 hidden layer, 2 nodes, ReLU activation functions
- Initialization of weights and bias terms
- Loss function: $\mathcal{L}\left(g(\boldsymbol{x}_i, \boldsymbol{W}), \boldsymbol{y}_i\right) = 1/2 \cdot (y - \widehat{y}_5)^2$
- Consider one data point, with $x_1 = 0.5$, $x_2 = 0.8$ and $y = 1$

Neural network: forward pass



$$\widehat{h}_3 = \max(w_{13}x_1 + w_{23}x_2 + b_3, 0) = 0.52$$
$$\widehat{h}_4 = \max(w_{14}x_1 + w_{24}x_2 + b_4, 0) = 0.43$$
$$\widehat{y}_5 = \max(w_{35}\widehat{h}_3 + w_{45}\widehat{h}_4 + b_5, 0) = 0.646$$

$\rightarrow$ Loss: $\mathcal{L}\left(g(\boldsymbol{x}_i, \boldsymbol{W}), \boldsymbol{y}_i\right) = 1/2 \cdot (y - \widehat{y}_5)^2 = 0.063$

Neural network: backpropagation (1/2)

- Last layer: direct differentiation of the loss function

$$\frac{\partial \mathcal{L}}{\partial w_{35}} = \frac{\partial \left(\frac{1}{2}(y - w_{35}\widehat{h}_3 - w_{45}\widehat{h}_4 - b_5)^2 \right)}{\partial w_{35}} = -\widehat{h}_3(y - \widehat{y}_5)$$

- Using the same logic, we obtain:

$$\frac{\partial \mathcal{L}}{\partial w_{35}} = -\widehat{h}_3(y - \widehat{y}_5) = -0.184$$
$$\frac{\partial \mathcal{L}}{\partial w_{45}} = -\widehat{h}_4(y - \widehat{y}_5) = -0.152$$
$$\frac{\partial \mathcal{L}}{\partial b_5} = -(y - \widehat{y}_5) = -0.35$$

- Interpretation:
  - We "undershoot" a bit for that data point: $y = 1$, $\widehat{y}_5 = 0.646$
  - Increasing the weights would increase $\widehat{y}_5$, hence reduce the loss:

$$\frac{\partial \mathcal{L}}{\partial w_{35}} < 0 \quad \frac{\partial \mathcal{L}}{\partial w_{45}} < 0 \quad \frac{\partial \mathcal{L}}{\partial b_5} < 0$$

$\rightarrow$ The stochastic gradient descent algorithm will increase the weights

Neural network: backpropagation (2/2)

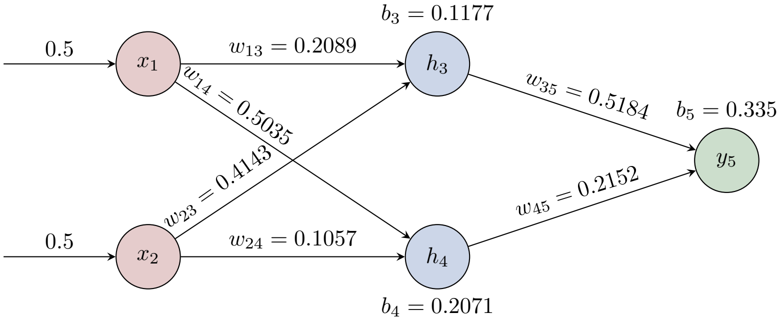
- Previous layer: differentiation of the loss function using the chain rule

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_{13}} &= \frac{\partial \mathcal{L}}{\partial \widehat{y}_5} \frac{\partial \widehat{y}_5}{\partial \widehat{h}_3} \frac{\partial \widehat{h}_3}{\partial w_{13}} \\
&= \frac{\partial \left(\frac{1}{2}(y - \widehat{y}_5)^2 \right)}{\partial \widehat{y}_5} \frac{\partial (w_{35}\widehat{h}_3 + w_{45}\widehat{h}_4 + b_5)}{\partial \widehat{h}_3} \frac{\partial (w_{13}x_1 + w_{23}x_2 + b_3)}{\partial w_{13}} \\
&= -(y - \widehat{y}_5) w_{35} x_1
\end{aligned}$$

- Using the same logic, we obtain:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_{13}} &= -(y - \widehat{y}_5) w_{35} x_1 = -0.089 & \frac{\partial \mathcal{L}}{\partial w_{23}} &= -(y - \widehat{y}_5) w_{35} x_2 = -0.143 \\
\frac{\partial \mathcal{L}}{\partial w_{14}} &= -(y - \widehat{y}_5) w_{45} x_1 = -0.035 & \frac{\partial \mathcal{L}}{\partial w_{24}} &= -(y - \widehat{y}_5) w_{45} x_2 = -0.057 \\
\frac{\partial \mathcal{L}}{\partial b_3} &= -(y - \widehat{y}_5) w_{35} = -0.177 & \frac{\partial \mathcal{L}}{\partial b_4} &= -(y - \widehat{y}_5) w_{45} = -0.071
\end{aligned}$$

Neural network: stochastic gradient update



- Update of the weights and the bias terms, with learning rate $\alpha = 0.1$

$$w_{ij} \leftarrow w_{ij} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ij}}, \ \forall i, j$$

$$b_i \leftarrow b_i - \alpha \frac{\partial \mathcal{L}}{\partial b_i}, \ \forall i$$

- Repeat with new data points (x_1, x_2), y

Convergence of stochastic gradient descent

Sample convergence results in stochastic gradient descent

Theorem (Convergence of the gradient)

Assume that f is M-smooth and that $\mathbb{E}(\|\nabla f_i(\boldsymbol{x})\|^2) \leq \sigma^2$ for all $i = 1, \cdots, n$ and for all $\boldsymbol{x} \in \mathbb{R}^n$. We have:

$$\min_{\ell=0,\cdots,k} \mathbb{E}(\|\nabla f(\boldsymbol{x}\ell)\|^2) \leq \frac{f(\boldsymbol{x}^0) - z^*}{\sum_{\ell=0}^{k-1} \alpha_\ell} + \frac{M\sigma^2}{2} \frac{\sum_{\ell=0}^{k-1} \alpha_\ell^2}{\sum_{\ell=0}^{k-1} \alpha_\ell}$$

Theorem (Convergence of the solution)

Assume that f is M-smooth and m-strongly convex and that $\mathbb{E}(\|\nabla f_i(\boldsymbol{x})\|^2) \leq \sigma^2$ for all $i = 1, \cdots, n$ and for all $\boldsymbol{x} \in \mathbb{R}^n$. The stochastic gradient descent algorithm with constant step size α satisfies:

$$\mathbb{E}(f(\boldsymbol{x}^k)) - z^* \leq (1 - 2\alpha m)^k (f(\boldsymbol{x}^0) - z^*) + \frac{M\alpha\sigma^2}{4m}$$

Convergence of the gradient: proof (OPTIONAL)

- Per the assumptions of the theorem:

$$f(\boldsymbol{x}^{k+1}) \leq f(\boldsymbol{x}^k) - \alpha_k \nabla f(\boldsymbol{x}^k)^\top \nabla f_i(\boldsymbol{x}^k) + \frac{M\alpha_k^2}{2} \|\nabla f_i(\boldsymbol{x}^k)\|^2$$

- Note that $\nabla f(\boldsymbol{x})^\top \nabla f_i(\boldsymbol{x})$ is not necessarily positive, meaning that the algorithm does not necessarily make progress at each iteration.
- Instead, we seek guarantees in expectation:

$$\mathbb{E}(f(\boldsymbol{x}^{k+1})) \leq \mathbb{E}(f(\boldsymbol{x}^k)) - \alpha_k \mathbb{E}(\nabla f(\boldsymbol{x}^k)^\top \nabla f_i(\boldsymbol{x}^k)) + \frac{M\alpha_k^2}{2} \mathbb{E}(\|\nabla f_i(\boldsymbol{x}^k)\|^2)$$

$$\Longrightarrow \mathbb{E}(f(\boldsymbol{x}^{k+1})) \leq \mathbb{E}(f(\boldsymbol{x}^k)) - \alpha_k \mathbb{E}(\|\nabla f(\boldsymbol{x}^k)\|^2) + \frac{M\alpha_k^2\sigma^2}{2}$$

$$\Longrightarrow z^* \leq \mathbb{E}(f(\boldsymbol{x}^k)) \leq \underbrace{\mathbb{E}(f(\boldsymbol{x}^0))}_{f(\boldsymbol{x}^0)} - \sum_{\ell=0}^{k-1} \alpha_\ell \mathbb{E}(\|\nabla f(\boldsymbol{x}^\ell\|^2) + \frac{M\sigma^2}{2} \sum_{\ell=0}^{k-1} \alpha_\ell^2$$

$$\Longrightarrow \min_{\ell=0,\cdots,k} \mathbb{E}(\|\nabla f(\boldsymbol{x}^\ell)\|^2) \leq \frac{f(\boldsymbol{x}^0) - z^*}{\sum_{\ell=0}^{k-1} \alpha_\ell} + \frac{M\sigma^2}{2} \frac{\sum_{\ell=0}^{k-1} \alpha_\ell^2}{\sum_{\ell=0}^{k-1} \alpha_\ell}$$

Convergence of the solution: proof (OPTIONAL)

- Using the earlier identify with $\alpha_k = \alpha$, we have:

$$\mathbb{E}(f(\boldsymbol{x}^{k+1})) - z^* \leq \mathbb{E}(f(\boldsymbol{x}^k)) - z^* - \alpha \mathbb{E}(\|\nabla f(\boldsymbol{x}^k)\|^2) + \frac{M\alpha^2\sigma^2}{2}$$

- Due to strong convexity, and by minimizing over $\boldsymbol{y}$ on both sides:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}^k) + \nabla f(\boldsymbol{x}^k)^\top (\boldsymbol{y} - \boldsymbol{x}^k) + \frac{m}{2}\|\boldsymbol{y} - \boldsymbol{x}^k\|^2, \ \forall \boldsymbol{y}$$

$$\implies z^* \geq f(\boldsymbol{x}^k) - \frac{1}{2m}\|\nabla f(\boldsymbol{x}^k)\|^2$$

- We conclude:

$$\mathbb{E}(f(\boldsymbol{x}^{k+1})) - z^* \leq (1 - 2\alpha m)(\mathbb{E}(f(\boldsymbol{x}^k)) - z^*) + \frac{M\alpha^2\sigma^2}{2}$$

$$\implies \mathbb{E}(f(\boldsymbol{x}^k) - z^* \leq (1 - 2\alpha m)^k (f(\boldsymbol{x}^0) - z^*) + \frac{M\alpha^2\sigma^2}{2} \underbrace{\sum_{\ell=0}^{k-1} (1 - 2\alpha m)^\ell}_{\leq 1/(2\alpha m)}$$

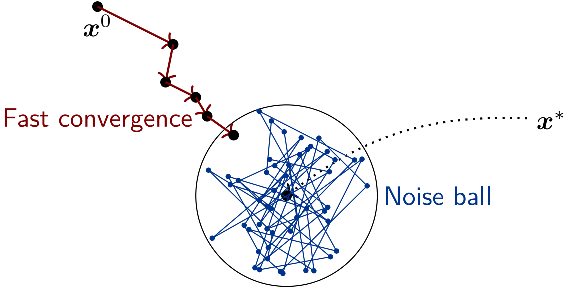
$$\implies \mathbb{E}(f(\boldsymbol{x}^k) - z^* \leq (1 - 2\alpha m)^k (f(\boldsymbol{x}^0) - z^*) + \frac{M\alpha\sigma^2}{4m}$$

Convergence behavior

Theorem (Convergence of the solution)

$$\mathbb{E}(f(\boldsymbol{x}^k)) - z^* \leq (1 - 2\alpha m)^k (f(\boldsymbol{x}^0) - z^*) + \frac{M\alpha\sigma^2}{4m}$$

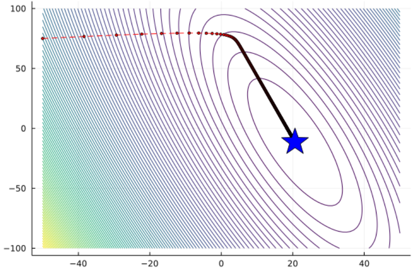
- Two-phase convergence behavior with constant step sizes
  1. Linear convergence to a neighborhood of a stationary point
  2. Lack of convergence within the neighborhood



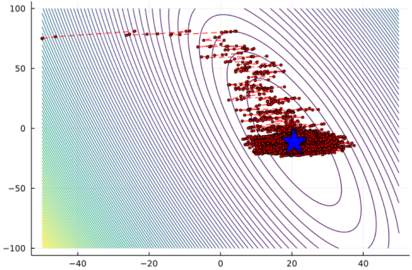
Linear regression example: convergence behavior in practice

- Gradient descent converges to a stationary point, which is a global optimum due to convexity
- Stochastic gradient descent converges quickly to a neighborhood of the stationary point, but exhibits noisy patterns thereafter
    - These noisy patterns can avoid local optima in non-convex optimization
    - Return best solution visited across iterations (not last visited solution!)

Gradient descent



Stochastic gradient descent



Implementation of stochastic gradient descent

Choice of step sizes

Theorem (Convergence of the gradient)

$$\min_{\ell=0,\cdots,k} \mathbb{E}(\|\nabla f(\boldsymbol{x}^\ell)\|^2) \leq \frac{f(\boldsymbol{x}^0) - z^*}{\sum_{\ell=0}^{k-1} \alpha_\ell} + \frac{M\sigma^2}{2} \frac{\sum_{\ell=0}^{k-1} \alpha_\ell^2}{\sum_{\ell=0}^{k-1} \alpha_\ell}$$

- SGD converges to approximate solution with constant step size $\alpha_k = \alpha$
  - Gradient descent: convergence in $\mathcal{O}(1/k)$
  - SGD: error of $\mathcal{O}(1/k) + \mathcal{O}(\alpha)$
- SGD can converge to an exact solution with diminishing step sizes, albeit at the cost of slower convergence
  - SGD, $\alpha_k = \alpha/k$: convergence in $\mathcal{O}\left(\frac{1}{\log(k)}\right)$
  - SGD, $\alpha_k = \alpha/\sqrt{k}$: convergence in $\mathcal{O}\left(\frac{\log(k)}{\sqrt{k}}\right)$
  - Diminishing step sizes have shown strong performance in practice
- Finding appropriate step sizes requires experimentation in practice

Stochastic gradient descent with mini-batches

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k \cdot \frac{1}{B} \sum_{i \in \mathcal{I}^k} \nabla f_i(\boldsymbol{x}^k)$$

Algorithm

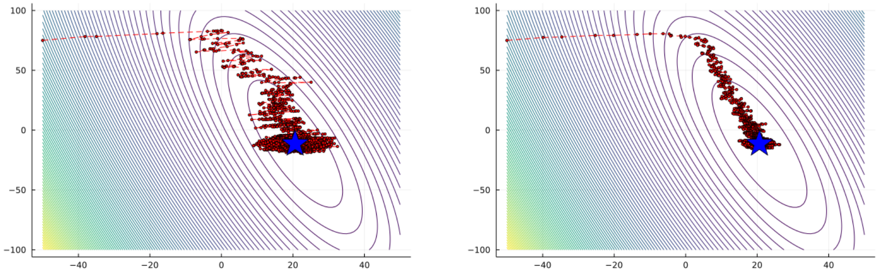
1. *Initialization: starting point* $\boldsymbol{x}^0 \in \mathbb{R}^n$, *and iteration counter* $k = 0$
2. *Repeat, until stopping criterion is reached*
   - 2.1 *Update iteration counter:* $k \leftarrow k + 1$
   - 2.2 *Choose batch* $\mathcal{I}^k \subseteq \{1, \cdots, n\}$, *with* $|\mathcal{I}^k| = B$
   - 2.3 *Choose descent direction* $\boldsymbol{d}^k = -\frac{1}{B} \sum_{i \in \mathcal{I}^k} \nabla f_i(\boldsymbol{x}^k)$
   - 2.4 *Determine a step size* $\alpha_k > 0$
   - 2.5 *Update* $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \alpha_k \boldsymbol{d}^k$

- Variance reduction in the gradient estimator
- Mini-batches define a continuum between SGD and GD
  - SGD with $B = 1$: fast iterations but high variability
  - GD with $B = n$: slow iterations but fewer iterations needed
  - $1 < B < n$: middle ground with mini-batches

Linear regression example: impact of mini-batches in practice

- Small mini-batches lead to similar patterns as stochastic gradient descent, with more stability
- Large mini-batches further improves stability, at the cost of slower iterations (as in gradient descent)

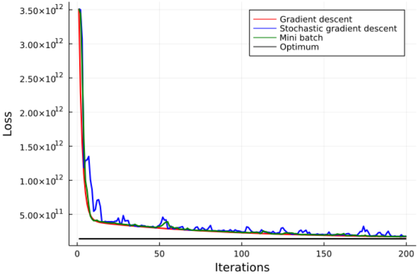
Batches of 2 **Batches of 10**



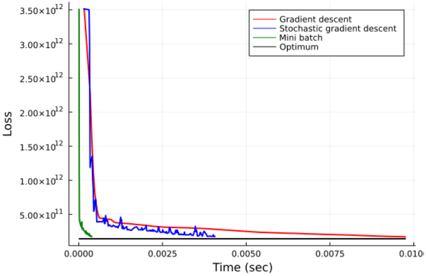
Linear regression example: convergence patterns

- Typically, gradient descent exhibits the strongest and most stable convergence patterns across iterations
- However, each iteration is faster in stochastic gradient descent, which can accelerate convergence—albeit, with noisy behavior
- Mini-batches can achieve the strongest convergence by accelerating gradient descent and stabilizing stochastic gradient descent

Batches of 2



Batches of 10



Application to heart disease prediction: case study setting

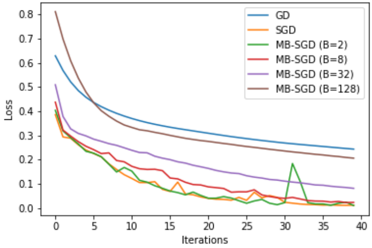
- Classification problem to predict heart disease from patient attributes
  - 303 patients in the dataset
  - 29 features for each patient: demographic data (age, sex), medical history (e.g., chest pain, electrocardiogram results, serum cholesterol)
  - Categorical feature: heart disease diagnosis
- Split between a train set (80%) and a test set (20%)
- Construction of the neural network
  - One hidden layer with 16 nodes
  - ReLU activation functions
  - Initialization of weights and bias terms
  - Loss function: cross-entropy

$\rightarrow$ 497 parameters to estimate
  - 29×16 weights linking each input node to each hidden node
  - 16 bias terms at each hidden node
  - 16 weights linking each hidden node to the output node
  - 1 bias term at the output node

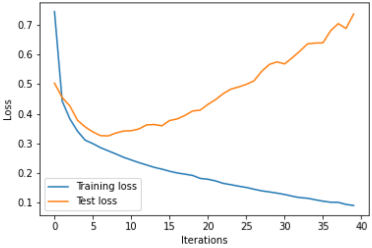
Application to heart disease prediction: results

- **Avoiding local optima:** Improvement in solution quality with stochastic gradient descent, as compared to gradient descent
- **Stabilization:** less noisy convergence behavior with mini-batching
- **Regularization:** Out-of-sample benefits of early stopping
  - Other regularization approaches in machine learning and deep learning

Impact of SGC and mini-batches



Training vs. testing loss



Conclusion

Implementation of stochastic gradient descent

1. Initialization: a strong starting point x^0 can enable faster convergence and convergence to a stronger solution
   - Possibly, leverage past solves as warm start
   - Repeat with multiple initializations to avoid bad local optima
2. Batching: mini-batches can stabilize and strengthen convergence
   - Determine the appropriate batching size to trade off number of iterations vs. computational time per iteration
   - Possibly, increase the batch size over iterations
3. Step size: the size and path of step sizes impact the speed of convergence and the quality of the solution
   - Determine the magnitude of the step sizes
   - Possible, decrease the step size across iterations
4. Termination: early stopping can avoid long tail in convergence
   - In machine learning, early stopping can also act as regularization

Summary

Takeaway

Stochastic gradient descent introduces randomness to accelerate gradient computations and avoid local minima, for separable functions.

Takeaway

Stochastic gradient descent achieves weaker theoretical guarantees than gradient descent: slower convergence and/or irreducible error.

Takeaway

Stochastic gradient descent with mini-batches achieves a middle ground between gradient descent (fewer, longer iterations) and stochastic gradient descent (more variability leading to more iterations, faster iterations).

Takeaway

Stochastic gradient descent has shown strong practical performance in large-scale machine learning problems, especially in deep learning.