Advanced Computational Statistics
Lecture 2 – Stochastic gradient based optimisation

Sebastian Mair
https://smair.github.io/
Division of Statistics and Machine Learning
Linköping University

March 12, 2025

Recap: Lecture 1

- Focus on unconstrained optimization
- Second-order methods (Newton, Quasi-Newton)
- First-order method: gradient ascent
- Properties of functions
  - convex $f((1-\alpha)x + \alpha y) \leq (1-\alpha)f(x) + \alpha f(y)$
  - strong-convex $f(y) \geq f(x) + \nabla f(x)^{\mathsf{T}}(y-x) + \frac{m}{2}\|x-y\|_2^2$
  - L-smooth $\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$
  - gradient ascent for max., gradient descent for min.

Gradient Descent

1. Get function f with gradient ∇f and step-size α_k
2. Initialize x^0
3. For $k = 1$ to ...
4. $\quad x^{k+1} = x^k - \alpha_k \nabla f\left(x^k\right)$

Theorem

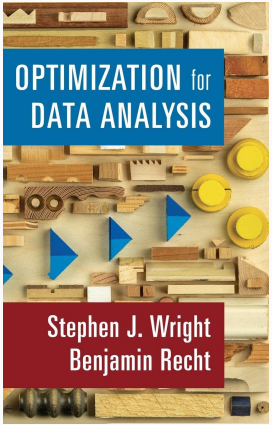
Suppose that f is convex and L-smooth, and suppose that $\min_{x \in \mathbb{R}^n} f(x)$ has solution $x^\star$. Then the steepest-descent method with step-length $\alpha_k = \frac{1}{L}$ generates a sequence $\left\{x^k\right\}_{k=0}^{\infty}$ that satisfies

$$f\left(x^T\right) - f(x^\star) \leq \frac{L}{2T}\left\|x^0 - x^\star\right\|^2, \quad T = 1, 2, \ldots.$$

Literature

The majority of this lecture is based on the following book:

Wright, S. J. and Recht, B. (2022). Optimization for data analysis. Cambridge University Press.



Lecture 2

Recommended reading: Sections 5.1-5.5 (and 6)

Today:

- Stochastic gradient descent (SGD)
- SGD in deep learning
- Differential private SGD
- Coordinate descent

Stochastic gradient descent (SGD)

Motivation

Consider the minimization of $f\colon \mathbb{R}^n \to \mathbb{R}$.

Assume f to be convex and smooth.

Many interesting objective functions are of the following form

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) \quad \text{with} \quad \nabla f(x) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x). \tag{5.5}$$

Examples include linear regression and logistic regression.

Recall: The sum of convex functions is convex!

Motivation

The problem is that running gradient descent

$$x^{k+1} = x^k - \alpha_k \nabla f\left(x^k\right), \quad k = 0, 1, 2, \ldots \tag{3.2}$$

requires the computation of the gradient $\nabla f(x)$ which might be expensive in the large-scale case (meaning that either N or n is very large).

Central question:

Is it possible to approximate the gradient $\nabla f(x)$?

Ideas:

(i) Approximate ∇f (think of a sum) using a subset of terms (shrink N)

(ii) Approximate ∇f (think of a vector) using a subset of dimensions (shrink n)

Stochastic Gradient Descent

Idea: Use $g(x, \xi) \in \mathbb{R}^n$ instead of $\nabla f(x)$ which is a function of x and a random variable ξ such that

$$\nabla f(x) = \mathbb{E}_\xi[g(x, \xi)]. \tag{5.1}$$

Some observations:

- We obtain an *unbiased* estimate of $\nabla f(x)$.
- Descent direction is in expectation equal to the one in gradient descent.
- Ideally, computing $g(x, \xi)$ is computationally cheaper than computing $\nabla f(x)$.
- We cannot rely on a fixed step-length of $\alpha_k = \frac{1}{L}$ anymore. Why?

How to choose $g(x, \xi)$?

Stochastic Gradient Descent

We could set

$$g(x, \xi) = \nabla f(x) + \xi \tag{5.3}$$

which is unbiased for $\mathbb{E}[\xi] = 0$.

- No computational savings!
- Relevant in differential privacy!

More later...

Stochastic Gradient Descent

Incremental gradient method

- Let ξ determine an index $i_k \in [N] = \{1, 2, \ldots, N\}$ and set $g(x, \xi) = \nabla f_{i_k}(x)$. Thus, if ξ has a uniform distribution over $[N]$, we retain unbiasedness:

$$\mathbb{E}_\xi[g(x, \xi)] = \sum_{i=1}^{n} \frac{1}{N} \nabla f_i(x) = \nabla f(x).$$

- Alternatively, we could also cycle through the components iteratively, i.e., setting $i_k = (k \bmod N) + 1$ for $k = 0, 1, 2, \ldots$.
- In both cases, we approximate the sum of gradients in Equation (5.5) by only a single term!
- The convergence analysis of the former is straightforward and the latter one is more involved and the guarantees are weaker.

Stochastic Gradient Descent

1. Get function f and step-sizes α_k
2. Initialize x^0
3. For $k = 1$ to ...
4. Sample ξ and obtain $g(x^k, \xi)$
5. $x^{k+1} = x^k - \alpha_k g\left(x^k, \xi^k\right)$

Consider the cost per iteration:

- Gradient descent has $\mathcal{O}(Nn)$ per iteration.
- Stochastic gradient descent has only $\mathcal{O}(n)$.

Motivations and examples from machine learning such as the *perceptron classifier* and the general framework of *empirical risk minimization* are considered in Sections 5.1.3 and 5.1.4 for the interested reader.

Example: Perceptron classifier

Input data: $\{a_1, a_2, \ldots, a_N\}$

Targets: $y_i \in \{\pm 1\}$

Model: $h(a) = x^{\mathsf{T}} a$

Classifier: $\operatorname{sign} h(a)$

Per-point loss: $f_i(x) = \max\{-y_i h(a_i), 0\}$

---

1. Get function h and step-sizes α_k
2. Initialize x^0
3. For $k = 1$ to ... // Iterations
4. For $i = 1$ to N // Loop over data set
5. If $y_i h(a_i) < 0$ // If misclassification
6. $x^{k+1} = x^k + \alpha_k y_i a_i$ // Model update

Stochastic Gradient Descent: Choosing the step-lengths

How to choose the step-lengths α_k?

Online demo: `https://fa.bianp.net/teaching/2018/COMP-652/stochastic_gradient.html`

Let us consider the following example of computing a mean:

$$f(x) = \frac{1}{2N} \sum_{i=1}^{N} (x - \omega_i)^2, \tag{5.11}$$

where we have data $\{\omega_i\}_{i=1}^N \subset \mathbb{R}$ and set $f_i(x) = \frac{1}{2}(x - \omega_i)^2$.

The gradient per-term is given by $\nabla f_i(x) = x - \omega_i$.

Stochastic Gradient Descent: Discrete example

- Start with $x^0 = 0$
- Apply incremental gradient
- Step through the indices in order
- Use a step-length of $\alpha_k = \frac{1}{k+1}$

$$
\begin{aligned}
x^1 &= x^0 - \frac{1}{1}(x^0 - \omega_1) = \omega_1 \\
x^2 &= x^1 - \frac{1}{2}(x^1 - \omega_2) = \frac{1}{2}\omega_1 + \frac{1}{2}\omega_2 \\
x^3 &= x^2 - \frac{1}{3}(x^2 - \omega_3) = \frac{1}{3}\omega_1 + \frac{1}{3}\omega_2 + \frac{1}{3}\omega_3 \\
&\vdots \\
x^k &= \left(\frac{k-1}{k}x^{k-1} + \frac{1}{k}\omega_k\right) = \frac{1}{k}\sum_{j=1}^{k}\omega_j
\end{aligned}
\tag{5.12}
$$

Stochastic Gradient Descent: Discrete example

Some observations:

- $\alpha_k = \frac{1}{k+1}$ makes sense in this scenario
- Due to $\sum_{k=0}^{\infty} \frac{1}{k+1} = \infty$ we can travel arbitrarily far
- $\alpha_k = \frac{1}{k+1}$ shrink to zero, thus, if we reach a neighborhood of $x^\star$, we stay there
- Global optimum achieved after N steps
- Works for the incremental case, not for the random one

Stochastic Gradient Descent: Continuous example

Let us now consider the *continuous* version of Equation (5.11)

$$f(x) = \frac{1}{2}\mathbb{E}_{\omega}\left[(x - \omega_i)^2\right], \tag{5.13}$$

where ω is a random variable with mean μ and variance σ^2.

At every SGD step, we sample $\omega_{j+1} \sim p(\omega)$ iid of the previous iterations and take a step of length $\frac{1}{j+1}$ in direction $x^j - \omega_{j+1}$.

After k steps, starting from $x^0 = 0$, we have x^k as in Equation (5.12)

Stochastic Gradient Descent: Continuous example

We now plug this value into Equation (5.13) and obtain

$$f(x^k) = \frac{1}{2}\mathbb{E}_{\omega_1,\ldots,\omega_k,\omega}\left[\left(\frac{1}{k}\sum_{j=1}^{k}\omega_j - \omega\right)^2\right] = \frac{1}{2k}\sigma^2 + \frac{1}{2}\sigma^2. \tag{5.14}$$

For Equation (5.13) we can compute the minimizer directly. We rephrase it as

$$f(x) = \frac{1}{2}\mathbb{E}\left[(x-\omega)^2\right] = \frac{1}{2}\mathbb{E}\left[x^2 - 2\omega x + \omega^2\right] = \frac{1}{2}x^2 - \mu x + \frac{1}{2}\sigma^2 + \frac{1}{2}\mu^2$$

and see that $x^\star = \mu$ is a minimizer. Thus, $f(x^\star) = \frac{1}{2}\sigma^2$. How?

Stochastic Gradient Descent: Continuous example

Comparing the minimizer with Equation (5.14) yields

$$f(x^k) - f(x^\star) = \frac{1}{2k}\sigma^2 + \frac{1}{2}\sigma^2 - \frac{1}{2}\sigma^2 = \frac{1}{2k}\sigma^2.$$

Thus, the sequence of differences $\{f(x^k) - f(x^\star)\}$ shrinks like $\frac{1}{k}$.

This demonstrates a limitation of stochastic gradient descent:

We cannot expect linear convergence rates in general!

Stochastic Gradient Descent: Convergence analysis: Key assumptions

Apply SGD to a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ with steps of the form of

$$x^{k+1} = x^k - \alpha_k g(x^k, \xi^k) \tag{5.2}$$

and search directions $g(x, \xi)$ satisfying condition (5.1), i.e., $\nabla f(x) = \mathbb{E}_\xi[g(x, \xi)]$.

We need to assume bounds on the gradient estimates and thus non-negative constants L_g and B such that for all x

$$\mathbb{E}_\xi \left[\|g(x, \xi)\|_2^2\right] \leq L_g^2 \|x - x^\star\|^2 + B^2. \tag{5.19}$$

Remarks:

- This bounds the expectation over ξ for each x!
- When $L_g = 0$, f cannot be strongly convex over an unbounded domain.

Stochastic Gradient Descent: Convergence analysis: Key assumptions

The book considers four cases

- Case 1: Bounded Gradients: $L_g = 0$
- Case 2: Randomized Kaczmarz: $B = 0$, $L_g > 0$
- Case 3: Additive Gaussian Noise
- Case 4: Incremental Gradient

In this lecture, we consider **Case 1**.

Stochastic Gradient Descent: Convergence analysis: Case 1

Case 1 applies to logistic regression. The objective function is

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} -y_i x^\top a_i + \log(1 + \exp(x^\top a_i)), \tag{5.21}$$

where the data are $\{(a_i, y_i)\}_{i=1}^N$ with $y_i \in \{0, 1\}$ for $i \in [N]$.

Following Equation (5.5), we draw ξ uniformly from $[N]$ and

$$g(x, i) = \left(-y_i + \frac{\exp(x^\top a_i)}{1 + \exp(x^\top a_i)} \right) a_i.$$

Thus, Equation (5.19) holds with $L_g = 0$ and $B = \sup_{i \in [N]} \|a_i\|_2$. Why?

Stochastic Gradient Descent: Convergence analysis

We can measure the error in two ways.

- $\mathbb{E}[\|x - x^\star\|^2]$, where $x^\star$ is the solution and the expectation is over all ξ^k. This measure is most appropriate when f is strongly convex.
- $f(x) - f^\star$ can be used when f is convex.

Suitable choices of step-lengths α_k in Equation (5.2) depend on L_g and B and so do the convergence rates.

Stochastic Gradient Descent: Convergence analysis

Using (5.2) for updating the iterates, we can expand the distance to $x^\star$ as follows:

$$\begin{aligned}\|x^{k+1} - x^\star\|^2 &= \|x^k - \alpha_k g(x^k, \xi^k) - x^\star\|^2 \\ &= \|x^k - x^\star\|^2 - 2\alpha_k \langle g(x^k, \xi^k), x^k - x^\star\rangle + \alpha_k^2 \|g(x^k, \xi^k)\|^2. \end{aligned} \tag{5.23}$$

In the following, we take the expectation wrt all random variables encountered up to iteration k $(i_0, i_1, \ldots, i_k)$ and **analyze each term separately**.

Note that x^k depends on $\xi^0, \xi^1, \ldots, \xi^{k-1}$ but not ξ^k.

Stochastic Gradient Descent: Convergence analysis

We obtain

$$\begin{aligned}\mathbb{E}[\langle g(x^k,\xi^k), x^k - x^\star\rangle] &= \mathbb{E}\left[\mathbb{E}_{\xi_k}[\langle g(x^k,\xi^k), x^k - x^\star\rangle | \xi^0, \xi^1, \ldots, \xi^{k-1}]\right] \\ &= \mathbb{E}\left[\langle \mathbb{E}_{\xi_k}[g(x^k,\xi^k)|\xi^0,\xi^1,\ldots,\xi^{k-1}], x^k - x^\star\rangle\right] \\ &= \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^\star\rangle\right].\end{aligned}$$

By a similar argument we can bound the last term using Equation (5.19) as

$$\begin{aligned}\mathbb{E}[\|g(x^k,\xi^k)\|^2] &= \mathbb{E}\left[\mathbb{E}_{\xi_k}[\|g(x^k,\xi^k)\|^2 | \xi^0, \xi^1, \ldots, \xi^{k-1}]\right] \\ &\leq \mathbb{E}[L_g^2\|x - x^\star\|^2 + B^2].\end{aligned}$$

Stochastic Gradient Descent: Convergence analysis

Let us define $A_k = \mathbb{E}[\|x^k - x^\star\|^2]$ and write the expectation of Equation (5.23) as

$$A_{k+1} \leq (1 + \alpha_k^2 L_g^2) A_k - 2\alpha_k \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^\star \rangle\right] + \alpha_k^2 B^2. \tag{5.25}$$

What follows depends again on the cases where we focus again on the first case.

Due to $L_g = 0$, Equation (5.25) simplifies to

$$A_{k+1} \leq A_k - 2\alpha_k \mathbb{E}\left[\langle \nabla f(x^k), x^k - x^\star \rangle\right] + \alpha_k^2 B^2. \tag{5.26}$$

Stochastic Gradient Descent: Convergence analysis

We define

$$\lambda_k = \sum_{j=0}^{k} \alpha_j \qquad \text{and} \qquad \bar{x}^k = \frac{\sum_{j=0}^{k} \alpha_j x^j}{\sum_{j=0}^{k} \alpha_j} = \lambda_k^{-1} \sum_{j=0}^{k} \alpha_j x^j. \tag{5.27}$$

We now analyze $f(\bar{x}^k)$ given an initial point x^0 and any solution $x^\star$.

Let $D_0 = \|x^0 - x^\star\|$.

Stochastic Gradient Descent: Convergence analysis

After T iterations, we have

$$\mathbb{E}[f(\bar{x}^T) - f(x^\star)] \leq \mathbb{E}\left[\lambda_T^{-1} \sum_{j=0}^{T} \alpha_j (f(x^j) - f(x^\star))\right] \tag{5.28a}$$

$$\leq \lambda_T^{-1} \sum_{j=0}^{T} \alpha_j \mathbb{E}\left[\langle \nabla f(x^j), x^j - x^\star \rangle)\right] \tag{5.28b}$$

$$\leq \lambda_T^{-1} \sum_{j=0}^{T} \left[\frac{1}{2}(A_j - A_{j+1}) + \frac{1}{2}\alpha_j^2 B^2\right] \tag{5.28c}$$

$$= \frac{1}{2}\lambda_T^{-1}\left[\frac{1}{2}A_0 - A_{T+1} + B^2 \sum_{j=0}^{T} \alpha_j^2\right] \leq \frac{D_0^2 + B^2 \sum_{j=0}^{T} \alpha_j^2}{2\sum_{j=0}^{T} \alpha_j}. \tag{5.28d}$$

Here, (5.28a) follows from the convexity of f and the definition of $\bar{x}^T$; (5.28b) uses again the convexity of f; (5.28c) follows from (5.26); and $A_0 = D_0^2$.

Stochastic Gradient Descent: Convergence analysis

We now consider a fixed step-length $\alpha_k = \alpha > 0$ for all k and show

Proposition
Suppose we run SGD on a convex function f with $L_g = 0$ for T steps with fixed step-length $\alpha > 0$. Define

$$\alpha_{\text{opt}} = \frac{D_0}{B\sqrt{T+1}} \qquad \text{and} \qquad \theta = \frac{\alpha}{\alpha_{\text{opt}}}.$$

Then, we have the following bound

$$\mathbb{E}[f(\bar{x}^T) - f(x^\star)] \leq \frac{D_0^2 + B^2(T+1)\alpha^2}{2(T+1)\alpha} = \left(\frac{1}{2}\theta^{-1} + \frac{1}{2}\theta\right)\frac{BD_0}{\sqrt{T+1}}. \tag{5.29}$$

For $\theta = 1$, the bound is tightest and the decrease approx. linear!

The bound critically depends on the choice of α!

Stochastic Gradient Descent: Convergence analysis

Consider the case where both B and L_g are non-zero.

Let f be m-strongly convex. Then

$$A_{k+1} \leq \left(1 - 2m\alpha_k + \alpha_k^2 L_g^2\right) A_k + \alpha_k^2 B^2. \tag{5.34}$$

For a fix step-length $\alpha \in (0, 2m/L_g^2)$, we obtain

$$A_{k+1} \leq \left(1 - 2m\alpha + \alpha^2 L_g^2\right)^k D_0 + \frac{\alpha B^2}{2m - \alpha L_g^2}. \tag{5.35}$$

$\implies$ Even for large k, we get trapped in a ball around the optimum.

Idea: Reduce the radius of the ball by reducing α!

SGD in deep learning

Computational challenge 2: N is big

At each optimization step we need to compute the gradient

$$d^{(t)} = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)}) = \frac{1}{N} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(t)}).$$

Computational challenge: N big

We typically use a lot of training data N for training the neural network.
Computing the gradient is costly.

Solution: For each iteration, we only use a small part of the data set to compute the gradient $d^{(t)}$. This is called the stochastic gradient descent.

Stochastic gradient descent

A big data set is often redundant = many data points are similar.

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Stochastic gradient descent

A big data set is often redundant = many data points are similar.

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

If the training data is big, consider

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \sum_{i=1}^{\frac{N}{2}} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}) \text{ and } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \sum_{i=\frac{N}{2}+1}^{N} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}).$$

Stochastic gradient descent

A big data set is often redundant = many data points are similar.

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

If the training data is big, consider

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \sum_{i=1}^{\frac{N}{2}} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}) \text{ and } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \sum_{i=\frac{N}{2}+1}^{N} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}).$$

We can do the update with only half the computation cost!

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \gamma \frac{1}{N/2} \sum_{i=1}^{\frac{N}{2}} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(t)}),$$

$$\boldsymbol{\theta}^{(t+2)} = \boldsymbol{\theta}^{(t+1)} - \gamma \frac{1}{N/2} \sum_{i=\frac{N}{2}+1}^{N} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(t+1)}).$$

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

$$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}^{(0)} - \gamma \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_1, \mathbf{y}_1, \boldsymbol{\theta}^{(0)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

$$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} - \gamma \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_2, \mathbf{y}_2, \boldsymbol{\theta}^{(1)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

$$\boldsymbol{\theta}^{(3)} = \boldsymbol{\theta}^{(2)} - \gamma \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_3, \mathbf{y}_3, \boldsymbol{\theta}^{(2)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

$$\boldsymbol{\theta}^{(4)} = \boldsymbol{\theta}^{(3)} - \gamma \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_4, \mathbf{y}_4, \boldsymbol{\theta}^{(3)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Mini-batch

$$\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}^{(0)} - \gamma \frac{1}{5} \sum_{i=1}^{5} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(0)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)
- We typically do something in between (not one data point, and not all data). We use a smaller set called **mini-batch**.

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Mini-batch

$$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} - \gamma \frac{1}{5} \sum_{i=6}^{10} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(1)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)
- We typically do something in between (not one data point, and not all data). We use a smaller set called **mini-batch**.

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Mini-batch

$$\boldsymbol{\theta}^{(3)} = \boldsymbol{\theta}^{(2)} - \gamma \frac{1}{5} \sum_{i=11}^{15} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(2)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)
- We typically do something in between (not one data point, and not all data). We use a smaller set called **mini-batch**.

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Mini-batch

$$\boldsymbol{\theta}^{(4)} = \boldsymbol{\theta}^{(3)} - \gamma \frac{1}{5} \sum_{i=16}^{20} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}^{(3)})$$

- The extreme version of this strategy is to use only one data point at each training step (called **online learning**)
- We typically do something in between (not one data point, and not all data). We use a smaller set called **mini-batch**.
- One pass through the training data is called an **epoch**.

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Iteration:

Epoch:

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Iteration:

Epoch:

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.

Stochastic gradient descent

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}

Iteration:

Epoch:

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_7$	$\mathbf{x}_{10}$	$\mathbf{x}_3$	$\mathbf{x}_{20}$	$\mathbf{x}_{16}$	$\mathbf{x}_2$	$\mathbf{x}_1$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{12}$	$\mathbf{x}_6$	$\mathbf{x}_{11}$	$\mathbf{x}_{17}$	$\mathbf{x}_{15}$	$\mathbf{x}_5$	$\mathbf{x}_{14}$	$\mathbf{x}_4$	$\mathbf{x}_9$	$\mathbf{x}_{13}$	$\mathbf{x}_8$
y_7	y_{10}	y_3	y_{20}	y_{16}	y_2	y_1	y_{18}	y_{19}	y_{12}	y_6	y_{11}	y_{17}	y_{15}	y_5	y_{14}	y_4	y_9	y_{13}	y_8

Iteration:

Epoch:

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_7$	$\mathbf{x}_{10}$	$\mathbf{x}_3$	$\mathbf{x}_{20}$	$\mathbf{x}_{16}$	$\mathbf{x}_2$	$\mathbf{x}_1$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{12}$	$\mathbf{x}_6$	$\mathbf{x}_{11}$	$\mathbf{x}_{17}$	$\mathbf{x}_{15}$	$\mathbf{x}_5$	$\mathbf{x}_{14}$	$\mathbf{x}_4$	$\mathbf{x}_9$	$\mathbf{x}_{13}$	$\mathbf{x}_8$
y_7	y_{10}	y_3	y_{20}	y_{16}	y_2	y_1	y_{18}	y_{19}	y_{12}	y_6	y_{11}	y_{17}	y_{15}	y_5	y_{14}	y_4	y_9	y_{13}	y_8

Iteration: 1

Epoch: 1

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_7$	$\mathbf{x}_{10}$	$\mathbf{x}_3$	$\mathbf{x}_{20}$	$\mathbf{x}_{16}$	$\mathbf{x}_2$	$\mathbf{x}_1$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{12}$	$\mathbf{x}_6$	$\mathbf{x}_{11}$	$\mathbf{x}_{17}$	$\mathbf{x}_{15}$	$\mathbf{x}_5$	$\mathbf{x}_{14}$	$\mathbf{x}_4$	$\mathbf{x}_9$	$\mathbf{x}_{13}$	$\mathbf{x}_8$
y_7	y_{10}	y_3	y_{20}	y_{16}	y_2	y_1	y_{18}	y_{19}	y_{12}	y_6	y_{11}	y_{17}	y_{15}	y_5	y_{14}	y_4	y_9	y_{13}	y_8

Iteration: 2

Epoch: 1

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_7$	$\mathbf{x}_{10}$	$\mathbf{x}_3$	$\mathbf{x}_{20}$	$\mathbf{x}_{16}$	$\mathbf{x}_2$	$\mathbf{x}_1$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{12}$	$\mathbf{x}_6$	$\mathbf{x}_{11}$	$\mathbf{x}_{17}$	$\mathbf{x}_{15}$	$\mathbf{x}_5$	$\mathbf{x}_{14}$	$\mathbf{x}_4$	$\mathbf{x}_9$	$\mathbf{x}_{13}$	$\mathbf{x}_8$
y_7	y_{10}	y_3	y_{20}	y_{16}	y_2	y_1	y_{18}	y_{19}	y_{12}	y_6	y_{11}	y_{17}	y_{15}	y_5	y_{14}	y_4	y_9	y_{13}	y_8

Iteration: 3

Epoch: 1

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.

Stochastic gradient descent

Training data (reshuffled)

x_7	x_{10}	x_3	x_{20}	x_{16}	x_2	x_1	x_{18}	x_{19}	x_{12}	x_6	x_{11}	x_{17}	x_{15}	x_5	x_{14}	x_4	x_9	x_{13}	x_8
y_7	y_{10}	y_3	y_{20}	y_{16}	y_2	y_1	y_{18}	y_{19}	y_{12}	y_6	y_{11}	y_{17}	y_{15}	y_5	y_{14}	y_4	y_9	y_{13}	y_8

Iteration: 4

Epoch: 1

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_7$	$\mathbf{x}_{10}$	$\mathbf{x}_3$	$\mathbf{x}_{20}$	$\mathbf{x}_{16}$	$\mathbf{x}_2$	$\mathbf{x}_1$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{12}$	$\mathbf{x}_6$	$\mathbf{x}_{11}$	$\mathbf{x}_{17}$	$\mathbf{x}_{15}$	$\mathbf{x}_5$	$\mathbf{x}_{14}$	$\mathbf{x}_4$	$\mathbf{x}_9$	$\mathbf{x}_{13}$	$\mathbf{x}_8$
y_7	y_{10}	y_3	y_{20}	y_{16}	y_2	y_1	y_{18}	y_{19}	y_{12}	y_6	y_{11}	y_{17}	y_{15}	y_5	y_{14}	y_4	y_9	y_{13}	y_8

Iteration: 4

Epoch: 1

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.
- After each epoch we do another reshuffling and another pass through the data set.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_{19}$	$\mathbf{x}_{16}$	$\mathbf{x}_{18}$	$\mathbf{x}_{6}$	$\mathbf{x}_{9}$	$\mathbf{x}_{13}$	$\mathbf{x}_{1}$	$\mathbf{x}_{14}$	$\mathbf{x}_{20}$	$\mathbf{x}_{11}$	$\mathbf{x}_{3}$	$\mathbf{x}_{8}$	$\mathbf{x}_{7}$	$\mathbf{x}_{12}$	$\mathbf{x}_{4}$	$\mathbf{x}_{17}$	$\mathbf{x}_{5}$	$\mathbf{x}_{10}$	$\mathbf{x}_{2}$	$\mathbf{x}_{15}$
y_{19}	y_{16}	y_{18}	y_{6}	y_{9}	y_{13}	y_{1}	y_{14}	y_{20}	y_{11}	y_{3}	y_{8}	y_{7}	y_{12}	y_{4}	y_{17}	y_{5}	y_{10}	y_{2}	y_{15}

Iteration:

Epoch: 2

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.
- After each epoch we do another reshuffling and another pass through the data set.

Stochastic gradient descent

Training data (reshuffled)

x_{19}	x_{16}	x_{18}	x_6	x_9	x_{13}	x_1	x_{14}	x_{20}	x_{11}	x_3	x_8	x_7	x_{12}	x_4	x_{17}	x_5	x_{10}	x_2	x_{15}
y_{19}	y_{16}	y_{18}	y_6	y_9	y_{13}	y_1	y_{14}	y_{20}	y_{11}	y_3	y_8	y_7	y_{12}	y_4	y_{17}	y_5	y_{10}	y_2	y_{15}

Iteration: 5

Epoch: 2

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.
- After each epoch we do another reshuffling and another pass through the data set.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_{19}$	$\mathbf{x}_{16}$	$\mathbf{x}_{18}$	$\mathbf{x}_{6}$	$\mathbf{x}_{9}$	$\mathbf{x}_{13}$	$\mathbf{x}_{1}$	$\mathbf{x}_{14}$	$\mathbf{x}_{20}$	$\mathbf{x}_{11}$	$\mathbf{x}_{3}$	$\mathbf{x}_{8}$	$\mathbf{x}_{7}$	$\mathbf{x}_{12}$	$\mathbf{x}_{4}$	$\mathbf{x}_{17}$	$\mathbf{x}_{5}$	$\mathbf{x}_{10}$	$\mathbf{x}_{2}$	$\mathbf{x}_{15}$
y_{19}	y_{16}	y_{18}	y_{6}	y_{9}	y_{13}	y_{1}	y_{14}	y_{20}	y_{11}	y_{3}	y_{8}	y_{7}	y_{12}	y_{4}	y_{17}	y_{5}	y_{10}	y_{2}	y_{15}

Iteration: 6

Epoch: 2

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.
- After each epoch we do another reshuffling and another pass through the data set.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_{19}$	$\mathbf{x}_{16}$	$\mathbf{x}_{18}$	$\mathbf{x}_{6}$	$\mathbf{x}_{9}$	$\mathbf{x}_{13}$	$\mathbf{x}_{1}$	$\mathbf{x}_{14}$	$\mathbf{x}_{20}$	$\mathbf{x}_{11}$	$\mathbf{x}_{3}$	$\mathbf{x}_{8}$	$\mathbf{x}_{7}$	$\mathbf{x}_{12}$	$\mathbf{x}_{4}$	$\mathbf{x}_{17}$	$\mathbf{x}_{5}$	$\mathbf{x}_{10}$	$\mathbf{x}_{2}$	$\mathbf{x}_{15}$
y_{19}	y_{16}	y_{18}	y_{6}	y_{9}	y_{13}	y_{1}	y_{14}	y_{20}	y_{11}	y_{3}	y_{8}	y_{7}	y_{12}	y_{4}	y_{17}	y_{5}	y_{10}	y_{2}	y_{15}

Iteration: 7

Epoch: 2

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.
- After each epoch we do another reshuffling and another pass through the data set.

Stochastic gradient descent

Training data (reshuffled)

$\mathbf{x}_{19}$	$\mathbf{x}_{16}$	$\mathbf{x}_{18}$	$\mathbf{x}_{6}$	$\mathbf{x}_{9}$	$\mathbf{x}_{13}$	$\mathbf{x}_{1}$	$\mathbf{x}_{14}$	$\mathbf{x}_{20}$	$\mathbf{x}_{11}$	$\mathbf{x}_{3}$	$\mathbf{x}_{8}$	$\mathbf{x}_{7}$	$\mathbf{x}_{12}$	$\mathbf{x}_{4}$	$\mathbf{x}_{17}$	$\mathbf{x}_{5}$	$\mathbf{x}_{10}$	$\mathbf{x}_{2}$	$\mathbf{x}_{15}$
y_{19}	y_{16}	y_{18}	y_{6}	y_{9}	y_{13}	y_{1}	y_{14}	y_{20}	y_{11}	y_{3}	y_{8}	y_{7}	y_{12}	y_{4}	y_{17}	y_{5}	y_{10}	y_{2}	y_{15}

Iteration: 8

Epoch: 2

- If we pick the mini-batches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a mini-batch.
- One implementation is to randomly reshuffle the data before dividing it into mini-batches.
- After each epoch we do another reshuffling and another pass through the data set.

Mini-batch gradient descent

The full **stochastic gradient descent** algorithm (a.k.a **mini-batch gradient descent**) is as follows

1. Initialize $\boldsymbol{\theta}^{(0)}$, set $t \leftarrow 1$, choose batch size n_b and number of epochs n_e.
2. For $i = 1$ to n_e
   (a) Randomly shuffle the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$.
   (b) For $j = 1$ to $\frac{n}{n_b}$
      (i) Approximate the gradient of the loss function using the mini-batch $\{(\mathbf{x}_i, y_i)\}_{i=(j-1)n_b+1}^{jn_b}$,
      $\hat{d}^{(t)} = \frac{1}{n_b} \sum_{i=(j-1)n_b+1}^{jn_b} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$.
      (ii) Do a gradient step $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \gamma \hat{d}^{(t)}$.
      (iii) Update the iteration index $t \leftarrow t + 1$.

At each time we get a stochastic approximation of the true gradient $\hat{d}^{(t)} \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, y_i, \boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$, hence the name.

Implementation Aspects

- **Learning rate.**
  Recall that decreasing the learning rates during training helps to converge.
  Let γ be the learning rate and $\eta \in (0, 1)$ a shrinkage-factor.
  One approach is to use $\gamma\eta^{k-1}$ in the kth epoch.
- **Momentum.**
  For example, replace (5.2) with

$$x^{k+1} = x^k - \alpha_k g\left(x^k, \xi^k\right) + \beta_k\left(x^k - x^{k-1}\right) \tag{5.37}$$

Adam is a popular optimizer in deep learning:
Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." International Conference on Learning Representations (2015).

Differential private SGD

Differential Privacy in a Nutshell

Can we train models while preserving privacy?

Differential privacy offers a framework for publishing trained models in a way that respects the individual privacy of every user.

Definition
A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that

$$\mathcal{P}\left[\mathcal{M}(d) \in \mathcal{S}\right] \leq \exp(\epsilon)\mathcal{P}\left[\mathcal{M}(d') \in \mathcal{S}\right] + \delta.$$

Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3, pp. 265-284. Springer Berlin Heidelberg, 2006.

Differential Privacy

Can we train models while preserving privacy?

Differential privacy offers a framework for publishing trained models in a way that respects the individual privacy of every user.

Definition
A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that

$$\mathcal{P}\left[\mathcal{M}(d) \in \mathcal{S}\right] \leq \exp(\epsilon)\mathcal{P}\left[\mathcal{M}(d') \in \mathcal{S}\right] + \delta.$$

Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3, pp. 265-284. Springer Berlin Heidelberg, 2006.

Differentially Private Stochastic Gradient Descent

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Abadi, Martin, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy." In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp. 308-318. 2016.

Coordinate descent

Motivation

The problem is that running gradient descent

$$x^{k+1} = x^k - \alpha_k \nabla f\left(x^k\right), \quad k = 0, 1, 2, \ldots \tag{3.2}$$

requires the computation of the gradient $\nabla f(x)$ which might be expensive in the large-scale case (meaning that either N or n is very large).

Central question:

Is it possible to approximate the gradient $\nabla f(x)$?

Ideas:

(i) Approximate ∇f (think of a sum) using a subset of terms (shrink N)

(ii) Approximate ∇f (think of a vector) using a subset of dimensions (shrink n)

Coordinate Descent

Instead of approximating

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) \quad \text{with} \quad \nabla f(x) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x) \tag{5.5}$$

with a single term, we now consider isolated dimensions of $\nabla f(x)$, i.e.,

$$x^{k+1} = x^k - \alpha_k \nabla_{i_k} f(x^k) e_{i_k}, \tag{6.2}$$

where e_{i_k} is the i_kth unit vector.

We now consider smooth convex functions.

Coordinate Descent

Let L_i be a coordinate-wise Lipschitz constant, i.e.,

$$|\nabla_i f(x + \gamma e_i) - \nabla_i f(x)| \leq L_i |\gamma|, \quad \text{for } i = 1, \ldots, n \tag{6.4}$$

and

$$L_{\max} = \max_i \; L_i. \tag{6.5}$$

We further assume that f is convex and uniformly Lipschitz continuously differentiable and attains a minimum at the set $\mathcal{S}$.

There exists an $0 < R_0 < \infty$ such that

$$\max_x \min_{x^\star \in \mathcal{S}} \|x - x^\star\| \leq R_0.$$

Coordinate Descent

Theorem

Given the assumptions from the previous slide and that i_k in (6.2) is selected uniformly at random from $\{1, 2, \ldots, n\}$, and that $\alpha_k = L_{\max}^{-1}$.

Then for all $k > 0$, we have

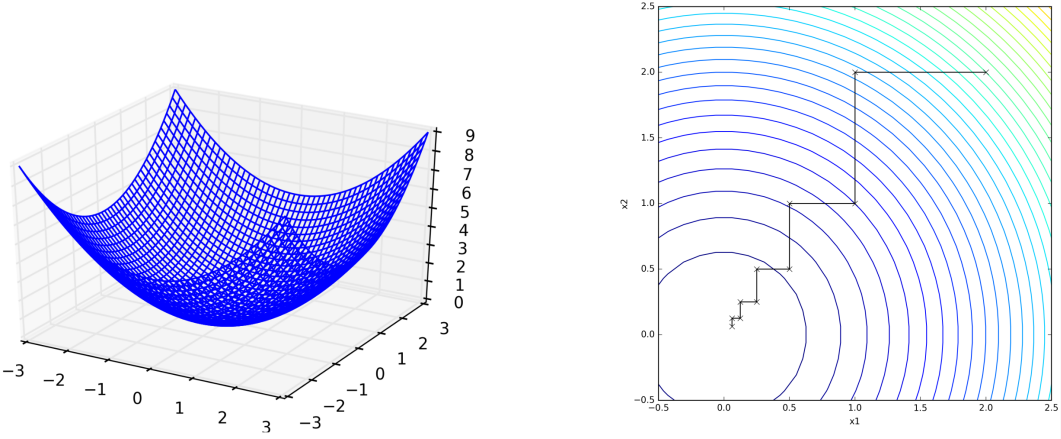
$$\mathbb{E}\left[f(x^k)\right] - f^\star \leq \frac{2nL_{\max}R_0^2}{k}. \tag{6.7}$$

If f is also strongly convex with $m > 0$, we have

$$\mathbb{E}\left[f(x^k)\right] - f^\star \leq \left(1 - \frac{m}{nL_{\max}}\right)^k \left(f(x^0) - f^\star\right). \tag{6.8}$$

Cyclic Coordinate Descent

Example: $\min_{x_1, x_2} f(x_1, x_2) = \frac{1}{2}(x_1^2 + x_2^2)$



Coordinate Descent

Theorem

Given the assumptions from some slides ago and that i_k in (6.2) is selected cyclically via $i_k = (k \bmod n) + 1$, and that $\alpha_k = L_{\max}^{-1}$.

Then for all $k = n, 2n, 3n, \ldots$, we have

$$f(x^k) - f^\star \leq \frac{\frac{4n}{\alpha_k}(1 + nL^2\alpha_k^2)R_0^2}{k+8}. \tag{6.20}$$

If f is also strongly convex with $m > 0$, we have

$$f(x^k) - f^\star \leq \left(1 - \frac{m}{\frac{2}{\alpha_k}(1 + nL^2\alpha_k^2)}\right)^{\frac{k}{n}} \left(f(x^0) - f^\star\right). \tag{6.21}$$

Matrix Norms, Assignment & Summary

Lipschitz Continuous Functions and Matrix Norms

- Most often when writing $\|\cdot\|$, we have a norm for a vector inside the norm (and $\|x\|_2$ can be interpreted as the length of vector x)
- There are also matrix-norms, and the spectral norm is one example: $\|A\|_{\text{spectral}} = \sqrt{\lambda_{\max}(A^\top A)}$, where $\lambda_{\max}(\cdot)$ is the largest eigenvalue of $A^\top A$.
- Spectral norm and Euclidean norm are *compatible* in the sense that for any $A \in \mathbb{R}^{n\times n}$ and $\mathbf{x} \in \mathbb{R}^n$, we have $\|Ax\|_2 \leq \|A\|_{\text{spectral}}\|x\|_2$.
- If f has a Hessian matrix f'' with a bounded spectral norm (by L), the gradient f' is Lipschitz continuous with L:

$$\|f''(x)\|_{\text{spectral}} \leq L \text{ for all } x \implies f' \text{ Lipschitz continuous with } L$$

Assignment 2

- The second assignment is mainly about Lecture 2.
- For the programming task, use a programming language of your choice.
- **Deadline:** March 31st, submission via e-mail to Sebastian.
- You will be assigned a peer-review.
  - Grade the assignment of your peer.
  - Provide constructive feedback.
  - Recommend acceptance/rejection with a brief justification.
- Submit your peer-review until April 14th via e-mail to Sebastian.

A few concepts to summarize Lecture 2

Stochastic gradient descent (SGD): A version of gradient descent where we at each iteration we only use a small part of the training data (a single data point or a mini-batch).

Learning rate (a.k.a step-length): A scalar tuning parameter deciding the length of each gradient step in GD/SCG/CD.

Mini-batch: The group of training data that we use at each iteration in SGD.

Batch size: The number of data points in one mini-batch.

Epoch: One complete pass though the entire training data set using SGD.

Adam: A state-of-the-art optimizer based on SGD and momentum which is popular in deep learning.

Differential privacy (DP): A probabilistic framework that allows us to publish models that respect individual user privacy.

Coordinate descent (CD): A version of gradient descent where we at each iteration we only update a single dimension of the parameter vector.