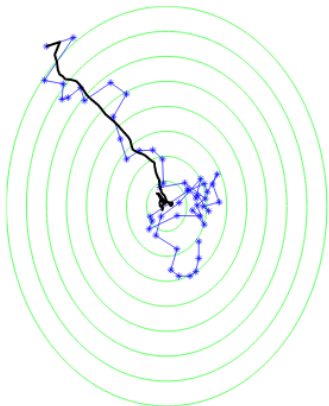
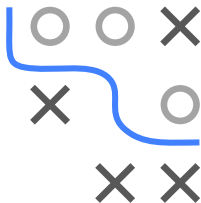


# Optimization in Machine Learning

## First order methods

### SGD



#### Learning goals

- Understand stochastic gradient descent
- Understand stochasticity and variance of SGD
- Know convergence properties of SGD
- Understand effect of batch size

# STOCHASTIC GRADIENT DESCENT

- NB: We use  $g$  instead of  $f$  as objective, because  $f$  is used as model in ML
- $g : \mathbb{R}^d \rightarrow \mathbb{R}$  objective,  $g$  is **average over functions**:

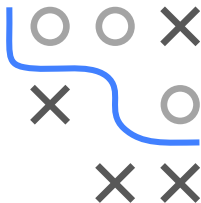
$$g(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n g_i(\mathbf{x}), \quad g \text{ and } g_i \text{ smooth}$$

- Stochastic gradient descent (SGD) approximates the gradient

$$\begin{aligned}\nabla_{\mathbf{x}} g(\mathbf{x}) &= \frac{1}{n} \sum_{i=1}^n \nabla_{\mathbf{x}} g_i(\mathbf{x}) \quad := \quad \mathbf{d} \quad \text{by} \\ \frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} \nabla_{\mathbf{x}} g_i(\mathbf{x}) &:= \hat{\mathbf{d}},\end{aligned}$$

with random subset  $J \subset \{1, 2, \dots, n\}$  of gradients called **mini-batch**

- This is done e.g. when computing the true gradient is **expensive**



## SGD ALGORITHM

```

1: Initialize  $\mathbf{x}^{[0]}$ ,  $t = 0$ 
2: while stopping criterion not met do
3:   Randomly shuffle indices and partition into minibatches  $J_1, \dots, J_K$  of size  $m$ 
4:   for  $k \in \{1, \dots, K\}$  do
5:      $t \leftarrow t + 1$ 
6:     Compute gradient estimate with  $J_k$ :  $\hat{\mathbf{d}}^{[t]} \leftarrow \frac{1}{m} \sum_{i \in J_k} \nabla_{\mathbf{x}} g_i(\mathbf{x}^{[t-1]})$ 
7:     Apply update:  $\mathbf{x}^{[t]} \leftarrow \mathbf{x}^{[t-1]} - \alpha \cdot \hat{\mathbf{d}}^{[t]}$ 
8:   end for
9: end while

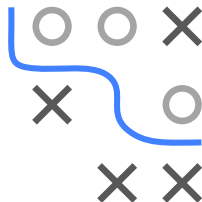
```

- Instead of drawing batches randomly we might want to go through the  $g_i$  sequentially (unless  $g_i$  are sorted in any way)
- Updates are computed faster, but also more stochastic:
  - In the simplest case, batch-size  $m := |J_k|$  is set to  $m = 1$
  - If  $n$  is a billion, computation of update is a billion times faster
  - **But** (later): Convergence rates suffer from stochasticity!

- In ML, we perform ERM:

$$\mathcal{R}(\theta) = \frac{1}{n} \sum_{i=1}^n \underbrace{L(y^{(i)}, f(\mathbf{x}^{(i)} | \theta))}_{g_i(\theta)}$$

- for a data set  $\mathcal{D} = ((\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)}))$
- a loss function  $L(y, f(\mathbf{x}))$ , e.g., L2 loss  $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
- and a model class  $f$ , e.g., the linear model  $f(\mathbf{x}^{(i)} | \theta) = \theta^\top \mathbf{x}$



- For large data sets, computing the exact gradient

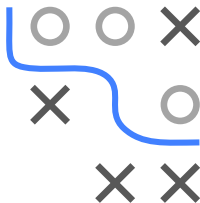
$$\mathbf{d} = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} L \left( y^{(i)}, f(\mathbf{x}^{(i)} | \theta) \right)$$

may be expensive or even infeasible to compute and is approximated by

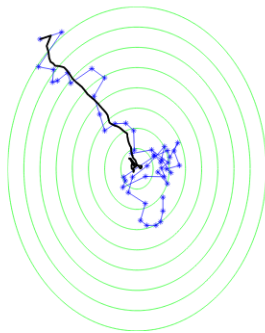
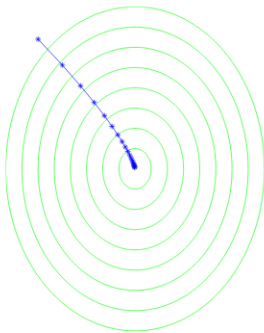
$$\hat{\mathbf{d}} = \frac{1}{m} \sum_{i \in J} \nabla_{\theta} L \left( y^{(i)}, f(\mathbf{x}^{(i)} | \theta) \right),$$

for  $J \subset \{1, 2, \dots, n\}$  random subset

- **NB:** Often, maximum size of  $J$  technically limited by memory size

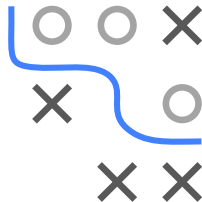


# STOCHASTICITY OF SGD



► [Click for source](#)

Minimize  $g(x_1, x_2) = 1.25(x_1 + 6)^2 + (x_2 - 8)^2$  **Left: GD, Right: SGD**  
Black line shows average value across multiple runs

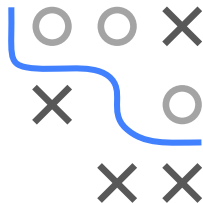


# STOCHASTICITY OF SGD

- Assume batch size  $m = 1$  (statements also apply for larger batches)
- **(Possibly) suboptimal direction:** Approximate gradient  $\hat{\mathbf{d}} = \nabla_{\mathbf{x}} g_i(\mathbf{x})$  might point in suboptimal (possibly not even a descent!) direction
- **Unbiased estimate:** If  $J$  drawn i.i.d., approximate gradient  $\hat{\mathbf{d}}$  is an unbiased estimate of gradient  $\mathbf{d} = \nabla_{\mathbf{x}} g(\mathbf{x}) = \sum_{i=1}^n \nabla_{\mathbf{x}} g_i(\mathbf{x})$ :

$$\begin{aligned}\mathbb{E}_i [\nabla_{\mathbf{x}} g_i(\mathbf{x})] &= \sum_{i=1}^n \nabla_{\mathbf{x}} g_i(\mathbf{x}) \cdot \mathbb{P}(i = i) \\ &= \sum_{i=1}^n \nabla_{\mathbf{x}} g_i(\mathbf{x}) \cdot \frac{1}{n} = \nabla_{\mathbf{x}} g(\mathbf{x}).\end{aligned}$$

- **Conclusion:** SGD might perform single suboptimal moves, but moves in “right direction” **on average**



**Example:**  $g(\mathbf{x}) = \sum_{i=1}^5 g_i(\mathbf{x})$ ,  $g_i$  quadratic, batch size  $m = 1$

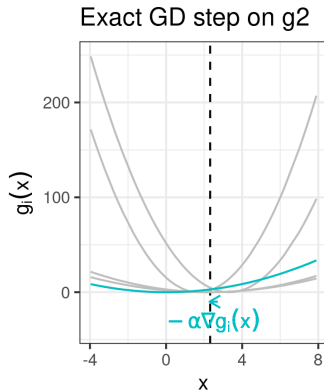
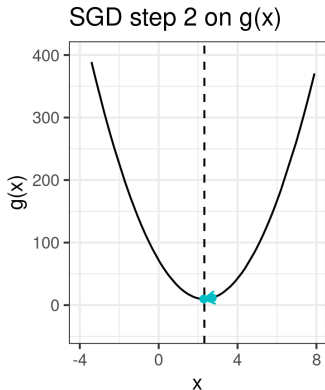
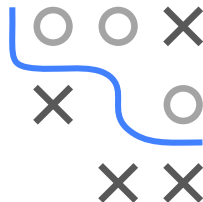
A 3x3 grid with a blue path starting at the top-left cell and ending at the bottom-right cell. The path moves right, then down, then right again. The cells (1,2), (1,3), (2,1), and (2,3) are blocked by grey 'X' marks.

The graph shows a function  $g(x)$  plotted against  $x$ . The x-axis ranges from -4 to 8, and the y-axis ranges from 0 to 400. A dashed vertical line is drawn at  $x=2$ , representing the axis of symmetry. A red arrow points from the point  $(0, 72)$  on the curve to the minimum point at  $(2, 12)$ .



# ERRATIC BEHAVIOR OF SGD

**Example:**  $g(\mathbf{x}) = \sum_{i=1}^5 g_i(\mathbf{x})$ ,  $g_i$  quadratic, batch size  $m = 1$



**Example:**  $g(\mathbf{x}) = \sum_{i=1}^5 g_i(\mathbf{x})$ ,  $g_i$  quadratic, batch size  $m = 1$

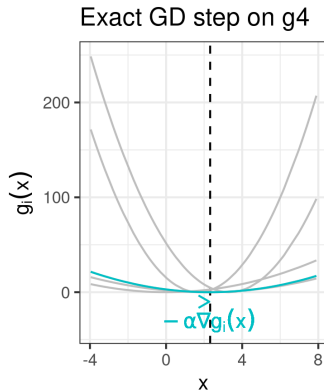
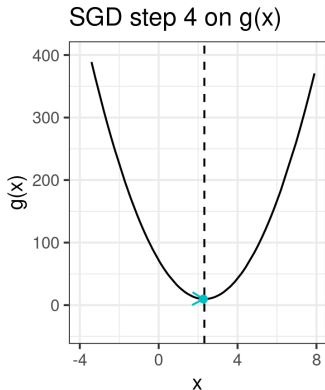
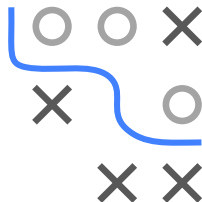
A 3x3 grid with a blue path starting at the top-left cell (0,0) and ending at the bottom-right cell (2,2). The path consists of the following cells: (0,0), (0,1), (1,1), (1,2), and (2,2). The cells (0,2), (1,0), and (2,0) are empty. The cells (0,1), (1,1), and (1,2) contain a grey 'X'. The cells (1,0) and (2,0) contain a grey circle.

A graph of a function  $g(x)$  is shown on a coordinate plane. The x-axis ranges from -4 to 8, and the y-axis ranges from 0 to 400. The function is a parabola opening upwards, with its vertex at  $(2, 0)$ . A vertical dashed line is drawn at  $x=2$ , and a red arrow points to the vertex at  $(2, 0)$ .

The plot shows several parabolic curves representing  $g_i(x)$  against  $x_i$ . A vertical dashed line marks the minimum of the functions at  $x_i = 2$ . At this point, the value of  $g_i(x)$  is 0. A red tangent line is drawn at the minimum, labeled  $-\alpha \nabla g_i(x)$ , representing the linear approximation of the function at that point.

# ERRATIC BEHAVIOR OF SGD

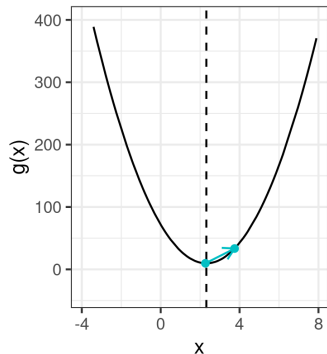
**Example:**  $g(\mathbf{x}) = \sum_{i=1}^5 g_i(\mathbf{x})$ ,  $g_i$  quadratic, batch size  $m = 1$



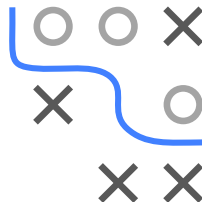
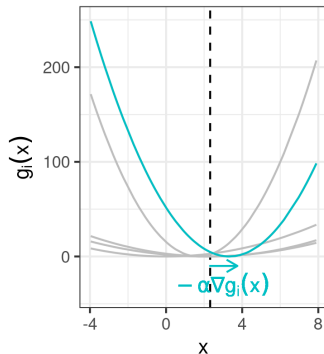
# ERRATIC BEHAVIOR OF SGD

**Example:**  $g(\mathbf{x}) = \sum_{i=1}^5 g_i(\mathbf{x})$ ,  $g_i$  quadratic, batch size  $m = 1$

SGD step 5 on  $g(\mathbf{x})$

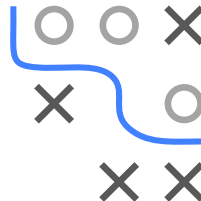
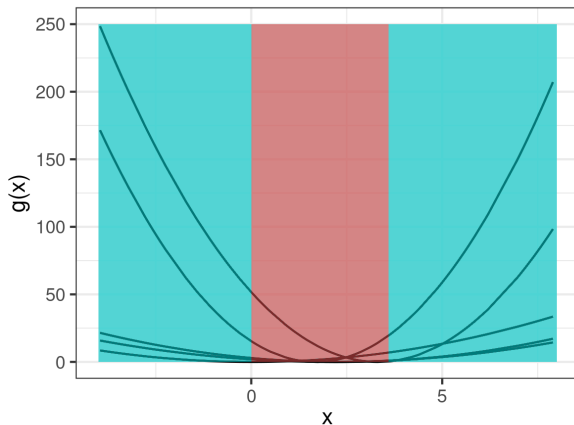


Exact GD step on  $g_5$



In iteration 5, SGD performs a suboptimal move away from the minimum.

# ERRATIC BEHAVIOR OF SGD



**Blue area:** Each  $-\nabla g_i(\mathbf{x})$  points towards minimum

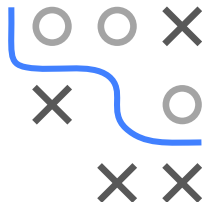
**Red area** (“confusion area”):  $-\nabla g_i(\mathbf{x})$  might point away from minimum  
and perform a suboptimal move

# ERRATIC BEHAVIOR OF SGD

- At location  $\mathbf{x}$ , “confusion” is captured by variance of gradients

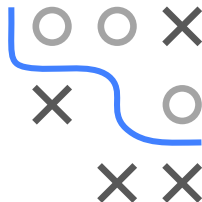
$$\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} g_i(\mathbf{x}) - \nabla_{\mathbf{x}} g(\mathbf{x})\|^2$$

- If term is 0, next step goes in gradient direction (for each  $i$ )
- If term is small, next step *likely* goes in gradient direction
- If term is large, next step likely goes in direction different than gradient



# CONVERGENCE OF SGD

- As a consequence, SGD has worse convergence properties than GD
- **But:** Can be controlled via **increasing batches** or **reducing step size**



## The larger the batch size $m$

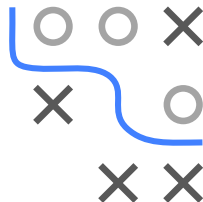
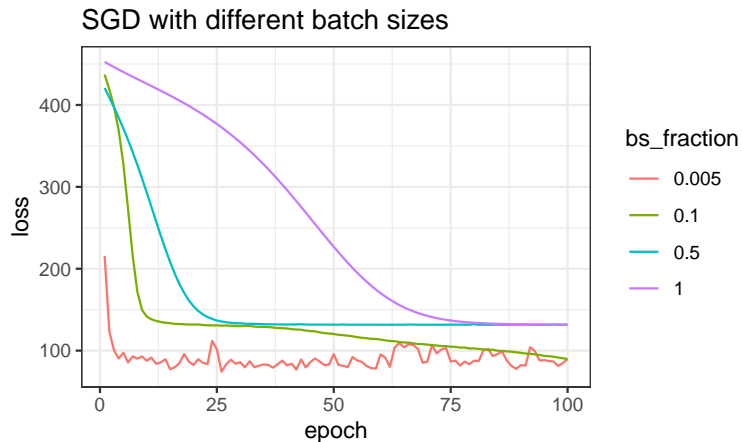
- the better the approximation to  $\nabla_{\mathbf{x}}g(\mathbf{x})$
- the lower the variance
- the lower the risk of performing steps in the wrong direction

## The smaller the step size $\alpha$

- the smaller a step in a potentially wrong direction
- the lower the effect of high variance

As maximum batch size is usually limited by computational resources (memory), choosing the step size is crucial

# EFFECT OF BATCH SIZE



SGD for a NN with batch size  $\in \{0.5\%, 10\%, 50\%\}$  of the training data  
The higher the batch size, the lower the variance