**Gradient Descent (GD)**

- **Task**: find parameters which minimize loss over the training dataset:

\[ \theta^* = \arg\min_{\theta} \mathcal{L}(\theta) \]

where \( \theta \) is a set of all parameters defining the ANN

- **Gradient descent**: \( \theta_{k+1} = \theta_k - \alpha_k \nabla \mathcal{L}(\theta_k) \)

where \( \alpha_k > 0 \) is the **learning rate** or **stepsize** at iteration \( k \)
Stochastic Gradient Descent (SGD): Motivation

- The loss has typically an additive structure, hence:

\[ \nabla L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla \ell(y_i, h_\theta(x_i)) \]

- Evaluation of \( \nabla L(\theta) \) takes \( O(m) \) time
- What if we have duplicate samples in \( T_m \)?
- Online learning?
- Use a single sample or a mini-batch instead of the full-batch approach ⇒ Stochastic Gradient Descent (SGD)
- The following is based on Bottou, Curtis and Nocedal: Optimization Methods for Large-Scale Machine Learning, 2018
Simplifying the Notation

- Let’s simplify and generalize the notation
- The gradient of loss (empirical risk) is

\[
\nabla \mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla \ell(y_i, h_\theta(x_i)) 
\]

- Represent a sample (or a set of samples) by a seed \( s \), meaning the realization of \( s \) is either an input-output pair \((x, y)\) or a set of pairs \( \{(x_i, y_i)\}_{i \in S}, S \subseteq \{1, \ldots, m\} \)
- Define \( f \) to be a composite of \( \ell \) and prediction \( h \)
- As an example, for GD above we can define \( s_i \triangleq (x_i, y_i) \in \mathcal{T}^m \) and write

\[
\nabla \mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla f(\theta, s_i) 
\]
SGD Algorithm

Stochastic Gradient Descent

1. Choose an initial iterate $\theta_1$
2. for $k = 1, 2, \ldots$
3. Generate a realization of the random variable $s_k$
4. Compute a stochastic gradient estimate vector $g(\theta_k, s_k)$
5. Choose a stepsize $\alpha_k > 0$
6. Set the new iterate as $\theta_{k+1} \leftarrow \theta_k - \alpha_k g(\theta_k, s_k)$

Possible options of a stochastic vector

$$g(\theta_k, s_k) = \begin{cases} 
\nabla f(\theta_k, s_k) & \text{single sample, online learning} \\
\frac{1}{m_k} \sum_{i=1}^{m_k} \nabla f(\theta_k, s_{k,i}) & \text{batch/mini-batch} \\
H_k \frac{1}{n_k} \sum_{i=1}^{n_k} \nabla f(\theta_k, s_{k,i}) & \text{Newton/quasi-Newton direction} 
\end{cases}$$

Holds for picking samples of $T_m$ with replacement, for picking without replacement it holds only until dataset gets exhausted in general

We consider the elements of the random sequence $\{s_k\}$ independent
SGD Convergence Theorem: Overview

- The main theorem shows that the *expected optimality gap*

\[ \mathbb{E}[\mathcal{L}(\theta_k) - \mathcal{L}_*] \xrightarrow{k \to \infty} 0 \]

where \( \mathcal{L}_* \) is the optimal (minimal) loss

- Assumptions:
  1. Strong convexity of \( \mathcal{L} \)
  2. Lipschitz continuous gradient \( \nabla \mathcal{L} \)
  3. Bounds on \( \mathcal{L} \) and \( g(\theta_k, s_k) \):
     - \( \mathcal{L} \) is bounded below by a scalar \( \mathcal{L}_{\text{inf}} \),
     - directions of \( g(\theta_k, s_k) \) and \( \nabla \mathcal{L}(\theta_k) \) *similar*,
     - their norms are also *similar*
Convexity

- Convex function definition:

\[ \mathcal{L}(t\theta + (1 - t)\bar{\theta}) \leq t\mathcal{L}(\theta) + (1 - t)\mathcal{L}(\bar{\theta}) \]

for all \((\theta, \bar{\theta}) \in \mathbb{R}^d \times \mathbb{R}^d\)

- Equivalently (first-order condition):

\[ \mathcal{L}(\bar{\theta}) \geq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T(\bar{\theta} - \theta) \]

the function lies above all its tangents

- See A4B33OPT
- But we need a stronger assumption...
Assumption 1: Strong Convexity

- The loss function $\mathcal{L} : \mathbb{R}^d \rightarrow \mathbb{R}$ is strongly convex if there exists constant $c > 0$ such that

$$
\mathcal{L}(\bar{\theta}) \geq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T (\bar{\theta} - \theta) + \frac{1}{2} c \| \bar{\theta} - \theta \|^2
$$

for all $(\theta, \bar{\theta}) \in \mathbb{R}^d \times \mathbb{R}^d$

- **Intuition**: quadratic lower bound on function growth
Strong Convexity Example

- Example (SVM objective): \[ \max(0, 1 - y(w^T x + b)) + \frac{\lambda}{2} \|w\|_2^2 \]
- Here, simplified to 1D
- Hinge loss is linear and constant in part: it is convex
- \( w^2 \) is strongly convex, easily show equality for \( c = 2 \):

\[
\bar{w}^2 \geq w^2 + 2w(\bar{w} - w) + (\bar{w} - w)^2 \\
\bar{w}^2 \geq w^2 + 2w(\bar{w} - w) + \bar{w}^2 - 2\bar{w} \cdot w + w^2 \\
0 \geq 2w^2 + 2w \cdot \bar{w} - 2w^2 - 2\bar{w} \cdot w = 0
\]
Assumption 2: Lipschitz Continuous Gradient

- The loss function is continuously differentiable and the gradient is *Lipschitz continuous* with *Lipschitz constant* $L > 0$:

$$
\| \nabla \mathcal{L}(\theta) - \nabla \mathcal{L}(\bar{\theta}) \|_2 \leq L \| \theta - \bar{\theta} \|_2, \text{ for all } (\theta, \bar{\theta}) \in \mathbb{R}^d \times \mathbb{R}^d
$$

- *Intuition*: the gradient does not change too quickly w.r.t. $\theta$

- Provides an indicator for how far to move to decrease $\mathcal{L}$

- Lemma (see Bottou et al. for the proof):

$$
\mathcal{L}(\bar{\theta}) \leq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T (\bar{\theta} - \theta) + \frac{1}{2} L \| \bar{\theta} - \theta \|_2^2
$$
Assumptions 1 & 2: Summary

- We have the strong convexity:

\[
\mathcal{L}(\bar{\theta}) \geq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T (\bar{\theta} - \theta) + \frac{1}{2} c \|\bar{\theta} - \theta\|^2_2
\]

and the Lipschitz continuous gradient:

\[
\mathcal{L}(\bar{\theta}) \leq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T (\bar{\theta} - \theta) + \frac{1}{2} L \|\bar{\theta} - \theta\|^2_2
\]

hence \( c \leq L \) holds

- Quadratic lower and upper bounds on \( \mathcal{L}(\bar{\theta}) \) growth
### Assumptions Summary

<table>
<thead>
<tr>
<th>constant</th>
<th>description</th>
<th>higher value means</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c &gt; 0$</td>
<td>strong convexity (lower bound)</td>
<td>&quot;more convex&quot;</td>
</tr>
<tr>
<td>$L &gt; 0$</td>
<td>Lipschitz continuous gradient (upper bound)</td>
<td>higher gradient change allowed</td>
</tr>
<tr>
<td>$\mathcal{L} \geq \mathcal{L}_{inf}$</td>
<td>lower bound on loss</td>
<td></td>
</tr>
<tr>
<td>$\mu &gt; 0$</td>
<td>$g(\theta_k, s_k)$ direction comparable to $\nabla \mathcal{L}(\theta_k)$</td>
<td>smaller angular difference between $\mathbb{E}_{s_k}[g(\theta_k, s_k)]$ and $\nabla \mathcal{L}(\theta_k)$</td>
</tr>
<tr>
<td>$M \geq 0$</td>
<td>limits expected scalar variance of $g(\theta_k, s_k)$</td>
<td>higher variance of $g(\theta_k, s_k)$ allowed</td>
</tr>
<tr>
<td>$M_G \geq \mu^2$</td>
<td>limits expected squared norm of $g(\theta_k, s_k)$ w.r.t. the $|\nabla \mathcal{L}(\theta_k)|_2$</td>
<td>higher expected ratio of $\mathbb{E}_{s_k}[g(\theta_k, s_k)]$ and $\nabla \mathcal{L}(\theta_k)$ norms allowed</td>
</tr>
</tbody>
</table>
SGD Convergence: Strongly Convex $\mathcal{L}$, Fixed Stepsize

**Theorem:** assuming Lipschitz continuity, the Bounds and strong convexity of $\mathcal{L}$, the SGD is run with a fixed stepsize $\alpha_k = \alpha$ for all $k \in \mathbb{N}$, where $0 < \alpha \leq \frac{\mu}{LMG}$. Then the expected optimality gap satisfies the following for all $k$:

$$
E[\mathcal{L}(\theta_k) - \mathcal{L}_*] \leq \frac{\alpha LM}{2c\mu} + (1 - \alpha c\mu)^{k-1} \left( \mathcal{L}(\theta_1) - \mathcal{L}_* - \frac{\alpha LM}{2c\mu} \right)
$$

$$
\xrightarrow{k \to \infty} \frac{\alpha LM}{2c\mu}
$$

**Note:** $(1 - \alpha c\mu)^{k-1} \xrightarrow{k \to \infty} 0$ as $0 < \alpha c\mu \leq \frac{c\mu^2}{LMG} \leq \frac{c\mu^2}{L\mu^2} = \frac{c}{L} \leq 1$

**In general,** for the fixed stepsize, the *optimality gap* tends to zero, but converges to $\frac{\alpha LM}{2c\mu} \geq 0$
Full-Batch Gradient Descent

- How does the theorem apply to the full-batch setting (GD)?
- The $g(\theta_k, s_k)$ is an unbiased estimate of $\nabla \mathcal{L}(\theta_k)$:

  $$\mathbb{E}_{s_k}[g(\theta_k, s_k)] = \nabla \mathcal{L}(\theta_k)$$

- Zero variance implies $M = 0$
- The optimality gap simplifies to:

  $$\epsilon_k = \mathbb{E}[\mathcal{L}(\theta_k) - \mathcal{L}_*] \leq (1 - \alpha c \mu)^{k-1} (\mathcal{L}(\theta_1) - \mathcal{L}_*) \xrightarrow{k \to \infty} 0$$

- Asymptotically we have $\epsilon_k \leq O(\rho^k)$, $\rho \in [0, 1)$
- For a given gap $\epsilon$, the number of iterations $k$ is proportional to $\log(1/\epsilon)$ in the worst case.
Theorem: assuming the strong convexity of $\mathcal{L}$ the Lipschitz continuity of $\nabla \mathcal{L}$ and the Bounds, the SGD is run with a stepsize such that, for all $k$

$$\alpha_k = \frac{\beta}{\gamma + k}$$

for some $\beta > \frac{1}{c\mu} > 0$ and $\gamma > 0$ such that $\alpha_1 \leq \frac{\mu}{LM_G}$

Then the expected optimality gap satisfies the following for all $k$:

$$\mathbb{E}[\mathcal{L}(\theta_k) - \mathcal{L}_*] \leq \frac{v}{\gamma + k}$$

where $v$ is a constant

Now we have $\mathbb{E}[\mathcal{L}(\theta_k) - \mathcal{L}_*] \xrightarrow{k \to \infty} 0$

Asymptotically the gap is $\epsilon_k \leq O(1/k)$ which means that the number of iterations $k$ is proportional to $1/\epsilon$ in the worst case
GD vs SGD

<table>
<thead>
<tr>
<th></th>
<th>GD</th>
<th>SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>time per iteration</td>
<td>$m$</td>
<td>1</td>
</tr>
<tr>
<td>iterations for accuracy $\epsilon$</td>
<td>$\log(1/\epsilon)$</td>
<td>$1/\epsilon$</td>
</tr>
<tr>
<td>time for accuracy $\epsilon$</td>
<td>$m \log(1/\epsilon)$</td>
<td>$1/\epsilon$</td>
</tr>
</tbody>
</table>

- SGD time does not depend on dataset size (if not exhausted)
- For large-scale problems (large $m$) SGD is faster
- It is harder to tune stepsize schedule for SGD, but you can experiment on a small representative subset of the dataset
- In practise *mini-batches* are used to leverage optimization/parallelization on CPU/GPU
Corresponding theorems can be proven for nonconvex objectives.

For assumptions similar to the theorem for the diminishing stepsizes (and excluding the strong convexity) we get:

$$\lim_{k \to \infty} \mathbb{E} \left[ \| \nabla \mathcal{L}(\theta_k) \|_2^2 \right] = 0$$
Momentum

- Simulate inertia to overcome plateaus in the error landscape:

\[
v_{k+1} \leftarrow \mu v_k - \alpha_k g(\theta_k, s_k)
\]

\[
\theta_{k+1} \leftarrow \theta_k + v_{k+1}
\]

where \( \mu \in [0, 1] \) is the momentum parameter

- Momentum damps oscillations in directions of high curvature

- It builds velocity in directions with consistent (possibly small) gradient
Adagrad

- Adaptive Gradient method (Duchi, Hazan and Singer, 2011)

- Motivation: a magnitude of gradient differs a lot for different parameters

- Idea: reduce learning rates for parameters having high values of gradient

\[
G_{k+1,i} \leftarrow G_{k,i} + [g(\theta_k, s_k)]^2_i
\]

\[
\theta_{k+1,i} \leftarrow \theta_{k,i} - \frac{\alpha}{\sqrt{G_{k+1,i} + \epsilon}} \cdot [g(\theta_k, s_k)]_i
\]

- $G_{k,i}$ accumulates squared partial derivative approximations w.r.t. to the parameter $\theta_{k,i}$

- $\epsilon$ is a small positive number to prevent division by zero

- Weakness: ever increasing $G_i$ leads to slow convergence eventually
RMSProp

- Similar to Adagrad but employs a moving average:

$$G_{k+1,i} = \gamma G_{k,i} + (1 - \gamma) \left[ g(\theta_k, s_k) \right]_i^2$$

- $\gamma$ is a decay parameter (typical value $\gamma = 0.9$)

- Unlike for Adagrad updates do not get infinitesimally small
\[ t \mathcal{L}(\theta) + (1 - t) \mathcal{L}(\bar{\theta}) \]

\[ \mathcal{L}(t\theta + (1 - t)\bar{\theta}) \]
\[ \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T (\bar{\theta} - \theta) \]
\[
\begin{align*}
\text{blue} & : \max(0, 1 - w) \\
\text{orange} & : 0.1w^2 \\
\text{green} & : \max(0, 1 - w) + 0.1w^2
\end{align*}
\]
$N = 15, \mu = 0., a = 0.1$

$N = 15, \mu = 0.7, a = 0.1$