Stochastic Gradient Descent

- Batches and Epochs
- Learning Rate and Momentum
- Nesterov Momentum
- ADAM optimizer
Batches, Epochs, and Stochastic Gradient Descent (SGD)

- Partition training set into **randomized** batches

\[ S = \{1, \cdots, K\} = \bigcup_{b=1}^{B} S_b \]

\[ K_b = |S_b| \]

\[ = (\# \text{ of sampler per batch}) \]

- For each batch you compute a separate gradient

\[ \nabla L(\theta; S_b) \leftarrow \text{Gradient for } b^{th} \text{ batch of training data} \]

Repeat until converged \{ 
Repeat \( b = 1 \) to \( B \) \{ 
\[ d \leftarrow -\nabla L(\theta; S_b) \]
\[ \theta \leftarrow \theta + \alpha d^t \]
\} \}

One epoch \{ One batch \}

*Stochastic Gradient Descent (SGD)*
Theoretical Analysis of SGD

- Assume simple sampling (sampling with replacement)

\[ g_k = \nabla L_k(\theta) = \text{(gradient from } k^{th} \text{ training sample)} \]

- Each sample, \( g_{ki} \), is i.i.d. with distribution \( p(g) = \frac{\text{histogram}(g)}{K} \)

True gradient:
\[ g = \frac{1}{K} \sum_{k=0}^{K-1} g_k \]

Batch gradient:
\[ \hat{g} = \frac{1}{K_b} \sum_{i=0}^{K_b-1} g_{ki} \]

Then
\[ \hat{g} = g + \frac{w}{\sqrt{K_b}} \]

where
\[ E[w] = 0 \]
\[ \text{Var}[w] \approx \frac{1}{K} \sum_{k=0}^{K-1} (g_k - g)(g_k - g)^t \]
Effect of Batch Size on SGD

Then we have that:

- As $K_b \to \infty$ (batch size goes up) $\Rightarrow$
  - Noise decreases and computation increases
- As $K_b \to 0$ (batch size goes down) $\Rightarrow$
  - Noise increases and computation decreases
Effect of Gradient Noise: Exploration

\[ \hat{g} = g + \frac{w}{\sqrt{K_b}} \]

- **Batch Gradient**
- **True Gradient**
- **Noise**

**Large Batch**
- \(-g(\theta)\) very good, but slow

**Small Batch**
- \(-\hat{g}(\theta)\) good, and fast

**Smooth Function**
- \(-g(\theta)\) very bad, and slow

**Bumpy Function**
- \(-\hat{g}(\theta)\) better, and fast
Effect of Gradient Noise: Exploitation

\[ \hat{g} = g + \frac{w}{\sqrt{K_b}} \]

Batch Gradient

True Gradient

Noise

Large Batch

\[ \text{perfect, but slow} \]

Small Batch

\[ \text{not so good, but fast} \]

Smooth Function
SGD Issues and Tradeoffs

- **Why SGD works?**
  - The gradient for a small batch is much faster to compute and almost as good as the full gradient.
  - If \( K = 10,000 \) and \( K_b = |S_b| = 32 \), then one iteration of SGD is approximately \( \frac{10,000}{32} \approx 312 \) times faster than GD.

- **Batch size**
  - Larger batches: less “noise” in gradient ⇒
    - **Worse**: slower updates; less exploration.
    - **Better**: better local convergence.
  - Smaller batches: more “noise” in gradient ⇒
    - **Worse**: hunts around local minimum.
    - **Better**: faster updates; better exploration.

- **Patch size:**
  - Many algorithms train on image “patches”
  - Apocryphal: Smaller patches speed training. Not true!!!!
  - However, smaller patches might fit better into GPU cache

- **Step size \( \alpha \)**
  - Too large ⇒ hunts around local minimum
  - Too small ⇒ slow convergence
Training with Patches

- **Concept:**
  - Break training images into patches
  - Often used in training denoising, deblurring, or reconstruction algorithms
  - Typically, may be $N \times N$ where $N = 80$ patches (DNCNN)
  - Typically, use a stride of $N_s = N/2$ so that patches overlap

- **Patch size issues:**
  - Apocryphal:
    - Smaller patches increase amount of training data. Not true!!
    - Smaller patches speed training. Not true!!!!
  - Advantages:
    - Smaller patches might fit better into GPU cache
  - Disadvantages:
    - Valid region tends towards 0 for deep CNNs

```
80x80x1 (5x5) ReLU
76x76x16 (5x5) ReLU
72x72x16
... 12 layers ...
24x24x16 (5x5) ReLU
20x20x1
```
## Momentum

- **SGD with momentum**
  - $\alpha$ is step size, and $\gamma$ is momentum typically with $\gamma = 0.9$

```
init $v \leftarrow 0$
Repeat until converged {
  Repeat $b = 1$ to $B$ {
    $d \leftarrow -\nabla L(\theta; S_b)$
    $v \leftarrow \gamma v + \alpha (1 - \gamma)d$
    $\theta \leftarrow \theta + v^t$
  }
}
```

- **Interpretation**
  - $\theta$ is like position
  - $v$ is like velocity
  - Friction $= 1 - \gamma$
Momentum

- **SGD with momentum**
  - $\alpha$ is step size, and $\gamma$ is momentum typically with $\gamma = 0.9$

init $v \leftarrow 0$

Repeat until converged {
  Repeat $b = 1$ to $B$ {
    $d \leftarrow -\nabla L(\theta; S_b)$
    $v \leftarrow \gamma v + \alpha(1 - \gamma)d$
    $\theta \leftarrow \theta + v^t$
  }
}

Interpretation
- $\theta$ is like position
- $v$ is like velocity
- Friction $= 1 - \gamma$
Interpretation of Momentum

- Special case of impulsive input: If \( d_n = \delta_n \)

\[
\text{init } v \leftarrow 0; \ \theta_{-1} \leftarrow 0
\]

Repeat \( n = 0 \) to \( N - 1 \) {
\[
v \leftarrow \gamma v + \alpha(1 - \gamma)\delta_n
\]
\[
\theta_n \leftarrow \theta_{n-1} + \nu^t
\]
}

\[
\theta_n = \begin{cases} 
\alpha(1 - \gamma^{n+1}) & n \geq 0 \\
0 & n < 0
\end{cases}
\]

Asymptotic value = \( \alpha \)

\[
\tau = 1 - \frac{1}{\log \gamma} \quad \gamma = \exp \left\{ -\frac{1}{\tau + 1} \right\}
\]

\( \tau \) - Time constant
Intuition

Friction = 1 − γ

Position \(\theta\)

Velocity \(v\)

Time constant

\[= 1 - \frac{1}{\log\gamma}\]
SGD with momentum

- $\alpha$ is step size, and $\gamma$ is momentum typically with $\gamma \approx 0.9$

```
init $v \leftarrow 0$
Repeat until converged {
  Repeat $b = 1$ to $B$ {
    $d \leftarrow -\nabla L(\theta + \gamma v^t; S_b)$
    $v \leftarrow \gamma v + \alpha d$
    $\theta \leftarrow \theta + v^t$
  }
}
```

**Stochastic Gradient Descent (SGD)**

* with Nesterov momentum

- Intuition:
  - Even if $d_n = 0$, we have that $\theta_{n+1} = \theta_n + \gamma v^t$ because of momentum
  - So compute the gradient at $\theta_n + \gamma v^t$

*Yu. E. Nesterov, “A method of solving a convex programming problem with convergence rate $O(1/k^2)$”, Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, no. 3, pp. 543–547.*

“I skate to where the puck is going to be, not where it has been.” - Wayne Gretzky
ADAM (Adaptive Moment Estimation)*

- SGD with ADAM optimization = Momentum + Preconditioning

\[
\text{init } v \leftarrow 0; \ r \leftarrow 0; \\
\text{init } t \leftarrow 0 \\
\text{Repeat until converged } \{ \\
\text{Repeat } b = 1 \text{ to } B \{ \\
\quad t \leftarrow t + 1 \\
\quad d \leftarrow -\nabla L(\theta; S_b) \\
\quad v \leftarrow \beta_1 v + (1 - \beta_1) d \\
\quad r \leftarrow \beta_2 r + (1 - \beta_2) d^2 \\
\quad \hat{v} \leftarrow v / (1 - \beta_1^t) \\
\quad \hat{r} \leftarrow r / (1 - \beta_2^t) \\
\quad \theta \leftarrow \theta + \alpha (\sqrt{\hat{r}} + \epsilon)^{-1} \hat{v} \\
\} \} \\
\]

- Typical parameters: $\alpha = 0.001; \ \beta_1 = 0.9; \ \beta_2 = 0.999; \ \epsilon = 10^{-8}$

*Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, The 3rd International Conference for Learning Representations (ICLR), San Diego, 2015.*
ADAM (Adaptive Moment Estimation)*

- SGD with ADAM optimization = Momentum + Preconditioning

```plaintext
init v ← 0; r ← 0;
init t ← 0
Repeat until converged {
    Repeat b = 1 to B {
        t ← t + 1
        d ← −∇L(θ; S_b)
        v ← β_1 v + (1 − β_1)d
        r ← β_2 r + (1 − β_2)d^2
        ̂v ← v / (1 − β_1^t)
        ̂r ← r / (1 − β_2^t)
        θ ← θ + α(√̂r + ε)^−1 ̂v
    }
}
```

ADAM Optimization

- Typical parameters: α = 0.001; β_1 = 0.9; β_2 = 0.999; ε = 10^{−8}

*Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, The 3rd International Conference for Learning Representations (ICLR), San Diego, 2015.*
**ADAM (Adaptive Moment Estimation)**

- SGD with ADAM optimization = Momentum + Preconditioning

\[
\begin{align*}
\text{init } v &\leftarrow 0; \ r \leftarrow 0; \\
\text{init } t &\leftarrow 0 \\
\text{Repeat until converged } \{ \\
&\text{Repeat } b = 1 \text{ to } B \{ \\
&t \leftarrow t + 1 \\
&d \leftarrow -\nabla L(\theta; S_b) \\
&v \leftarrow \beta_1 v + (1 - \beta_1)d \\
r \leftarrow \beta_2 r + (1 - \beta_2)d^2 \\
\hat{v} \leftarrow v / (1 - \beta_1^t) \\
\hat{r} \leftarrow r / (1 - \beta_2^t) \\
\theta \leftarrow \theta + \alpha (\sqrt{\hat{r}} + \epsilon)^{-1} \hat{v}
\}
\}
\end{align*}
\]

- Typical parameters: \( \alpha = 0.001; \ \beta_1 = 0.9; \ \beta_2 = 0.999; \ \epsilon = 10^{-8} \)

*Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, The 3rd International Conference for Learning Representations (ICLR), San Diego, 2015.*