Variance Reduction in SGD

Recap: Stochastic Gradient Descent

Convergence depends on variance

$$\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{D}}\left[\left(\frac{\partial l(\theta|\mathbf{x},\mathbf{y})}{\partial\theta} - \frac{\partial L(\theta|\mathcal{D})}{\partial\theta}\right)^2\right]$$

Low variance: faster convergence

High variance: slower convergence

Pseudocode: Stochastic Gradient Descent

```
θ ~ Init
for epoch in range(n):
  for (x, y) in dataset:
    J = ∇l(θ|x,y)
    θ = θ - ϵ * J.mT
```

Variance Reduction: Mini-Batches

Average several gradients before taking step

- Closer to GD

New hyper-parameter `batch_size`

- Vanilla SGD uses `batch_size=1`
- Vanilla GD uses `batch_size=len(dataset)`
- In practice, use a value in between

Stochastic Gradient Descent (with Mini-Batch)

```
for epoch in range(n):
  for i in range(len(dataset) // batch_size):
    J = 0
    batch = dataset[i * batch_size: (i + 1) * batch_size]
    for (x, y) in batch:
      J += ∇l(θ|x,y)
    θ = θ - ϵ * J.mT
```

SGD vs. SGD with Mini-batches

Stochastic Gradient Descent

```
for epoch in range(n):
  for (x, y) in dataset:
    J = ∇l(θ|x,y)
    θ = θ - ϵ * J.mT
```

Stochastic Gradient Descent (with Mini-Batch)

```
for epoch in range(n):
  for i in range(len(dataset) // batch_size):
    J = 0
    batch = dataset[i * batch_size: (i + 1) * batch_size]
    for (x, y) in batch:
      J += ∇l(θ|x,y)
    θ = θ - ϵ * J.mT
```

Variance of Mini-Batches

Variance of SGD with mini-batches

$$\sigma_{MB}^2 = \mathbb{E}_{\mathcal{B}_i}\left[\left(\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right] - \frac{\partial}{\partial\theta}L(\theta)\right)^2\right] = \mathbb{E}_{\mathcal{B}_i}\left[\left(\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right]\right)^2\right] - \left(\frac{\partial}{\partial\theta}L(\theta)\right)^2$$

Variance of SGD

$$\sigma_{SGD}^2 = \mathbb{E}_{\mathcal{B}_i}\left[\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\left(\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y}) - \frac{\partial}{\partial\theta}L(\theta)\right)^2\right]\right] = \mathbb{E}_{\mathcal{B}_i}\left[\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\left(\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right)^2\right]\right] - \left(\frac{\partial}{\partial\theta}L(\theta)\right)^2$$

Variance reduction

$$\begin{aligned}\sigma_{MB}^2 - \sigma_{SGD}^2 &= \mathbb{E}_{\mathcal{B}_i}\left[\left(\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right]\right)^2 - \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\left(\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right)^2\right]\right] \\ &= \mathbb{E}_{\mathcal{B}_i}\left[\underbrace{\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\left(\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}_i}\left[\frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right] - \frac{\partial}{\partial\theta}l(\theta|\mathbf{x},\mathbf{y})\right)^2\right]}_{\sigma_{\mathcal{B}_i}^2}\right] \geq 0\end{aligned}$$

SGD vs. SGD with Mini-batches

SGD

Failed to load file "/home/philkr/workspace/classes/deeplearning_v2/content/deep_networks/05_optimization/figures.ipynb.exec"

Failed to lo "/home/ph

How Big Should Your Mini-Batch Be?

- As big as possible!
- Preferably the power of 2 (8, 16, 32, 64, ...)

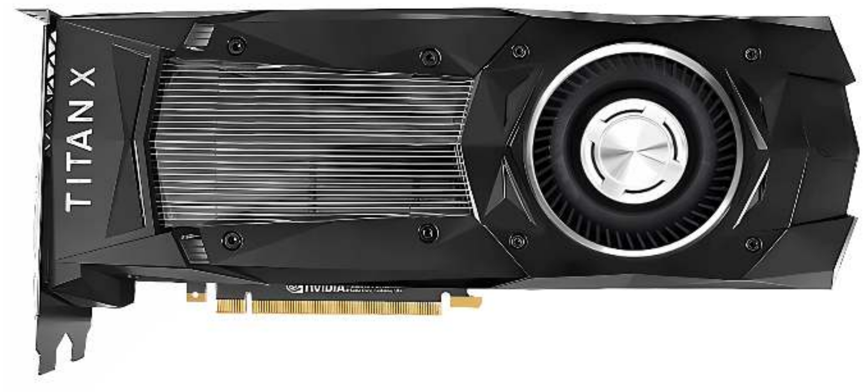

Image credit: Wikipedia ↩️

Always Use Mini-Batches

Variance Reduction: Momentum

Average several consecutive gradients

- Closer to GD

New hyper-parameter `momentum`

- Vanilla SGD uses `momentum=0`
- In practice, use `momentum=0.9`

Stochastic Gradient Descent (with Momentum)

```
b = 0
for epoch in range(n):
  for (x, y) in dataset:
    b = ∇l(θ|x,y) + momentum * b
    θ = θ - ϵ * b.mT
```

Mini-Batches vs Momentum

Stochastic Gradient Descent (with Mini-Batch)

```
for epoch in range(n):
  for i in range(len(dataset) // batch_size):
    J = 0
    batch = dataset[i * batch_size: (i + 1) * batch_size]
    for (x, y) in batch:
      J += ∇l(θ|x,y)
    θ = θ - ϵ * J.mT
```

Stochastic Gradient Descent (with Momentum)

```
b = 0
for epoch in range(n):
  for (x, y) in dataset:
    b = ∇l(θ|x,y) + momentum * b
    θ = θ - ϵ * b.mT
```

Variance Reduction: Momentum

- Momentum reduces variance and accelerates convergence
- Formal proof[1]:

1. Yurii Nesterov. Introductory lectures on convex optimization: a basic course. ↩️

SGD vs. SGD with Momentum

SGD

Failed to load file "/home/philkr/workspace/classes/deeplearning_v2/content/deep_networks/05_optimization/figures.ipynb.exec"

Failed to lo "/home/ph

What Should Your Momentum Value Be?

- Everyone uses `momentum=0.9`
- PyTorch defaults `momentum=0`, so don't forget to change

Docs > torch.optim > SGD

SGD

CLASS torch.optim.SGD(*params*, *lr=0.001*, *momentum=0*, *dampening=0*, *weight_decay=0*, *nesterov=False*, ***, *maximize=False*, *foreach=None*, *differentiable=False*) [SOURCE]

Implements stochastic gradient descent (optionally with momentum).

$$\begin{aligned}&\textbf{input}: \gamma \text{ (lr)},\ \theta_0 \text{ (params)},\ f(\theta) \text{ (objective)},\ \lambda \text{ (weight decay)},\\ &\qquad \mu \text{ (momentum)},\ \tau \text{ (dampening)},\ \textit{nesterov},\ \textit{maximize}\end{aligned}$$

$$\begin{aligned}&\textbf{for}\ t = 1\ \textbf{to}\ \ldots\ \textbf{do}\\ &\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})\\ &\quad \textbf{if}\ \lambda \neq 0\end{aligned}$$

Variance Reduction in SGD - TL;DR

Mini-batches and momentum *reduce variance* during optimization

Always use SGD with mini-batches and momentum