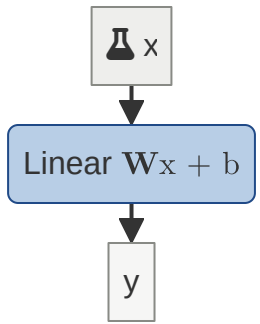
Optimization

Recap: Model Training Fundamentals

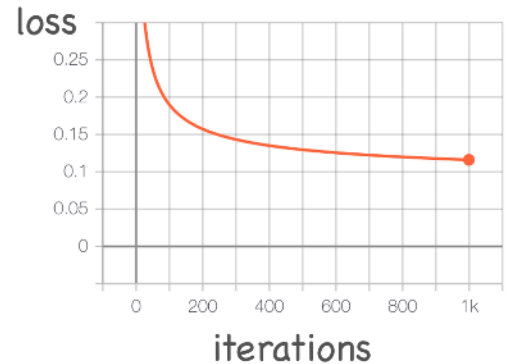
Task / Model



Dataset / Loss

$L(\theta|\mathcal{D})$

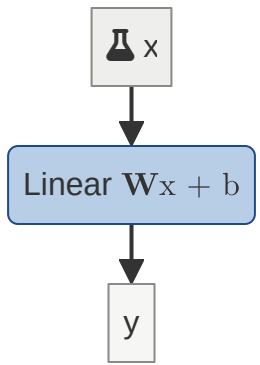
Training / Optimization



Recap: Models and Losses

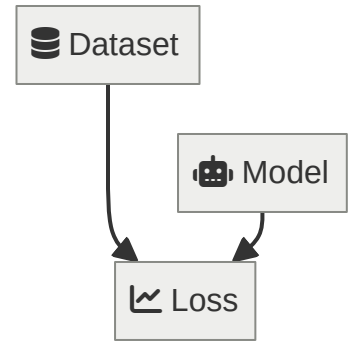
Model: f_θ

Parameters: $\theta = (\mathbf{W}, \mathbf{b})$



$$L(\theta) = L(\theta|\mathcal{D}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[l(\theta|\mathbf{x},\mathbf{y})]$$

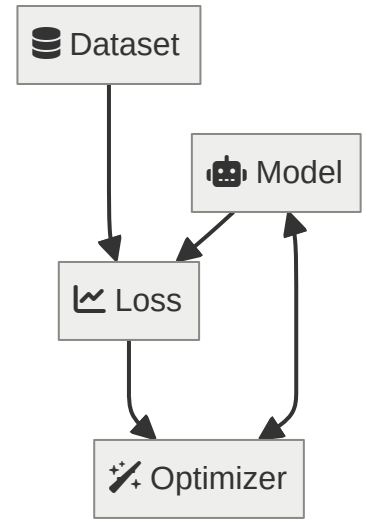
- Low loss - good ☺
- High loss - bad ☹
- Loss function over the full dataset $\mathcal{D}$



Training a Model

Find

$$\theta^* = \arg\min_{\theta} L(\theta|\mathcal{D})$$



Gradient Descent

Deep learning uses **gradient descent**:

- Updates parameters via the negative gradient
- Iterative method

Update rule:

$$\theta' = \theta - \epsilon \left[\nabla_\theta L(\theta)\right]^\top$$

where ϵ is the learning rate

Pseudocode

```
for iteration in range(n):
    J = ∇L(θ)
    θ = θ - ϵ * J.mT
```

Initialization

How should we choose the initial parameters θ?

Random initialization:

- Gaussian $\mathcal{N}(\mu, \sigma)$
- Uniform $\mathcal{U}(a, b)$

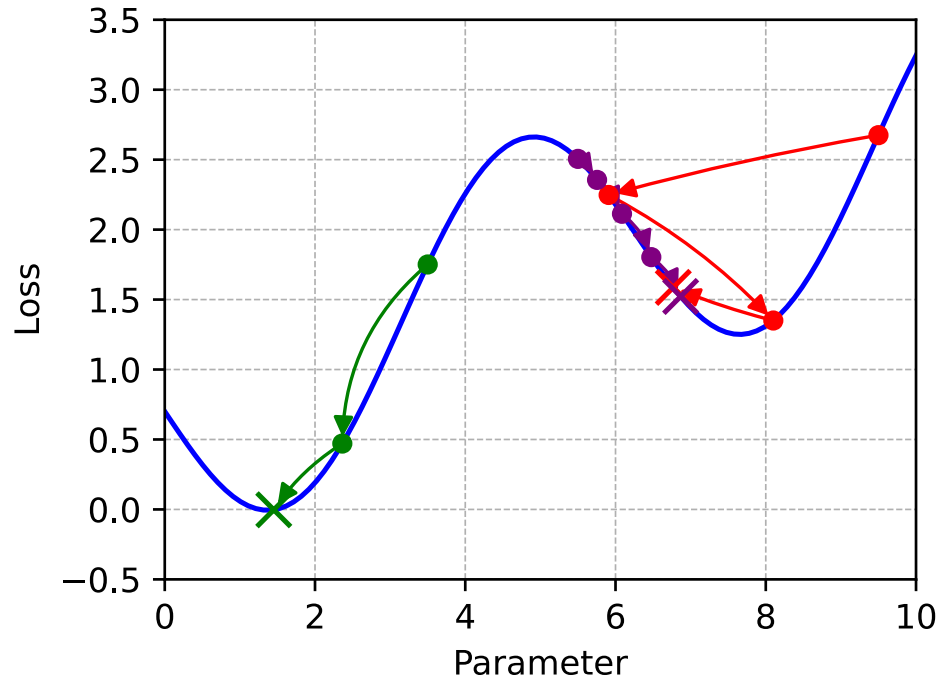
Pseudocode: Gradient Descent

```
θ ~ Init
for iteration in range(n):
    J = ∇L(θ)
    θ = θ - ϵ * J.mT
```

Gradient Descent in Action

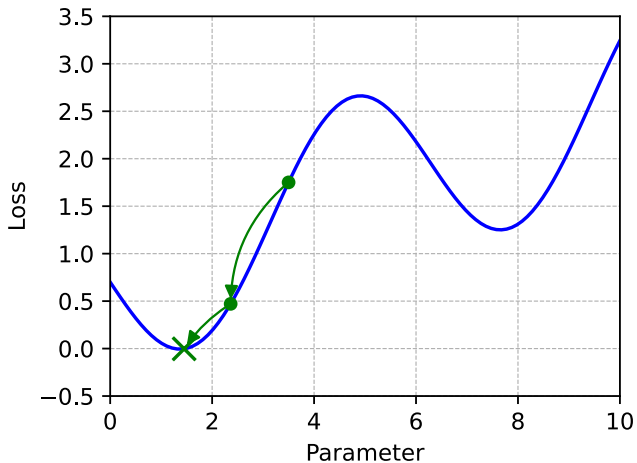
Notebook Demo

Gradient Descent in Action

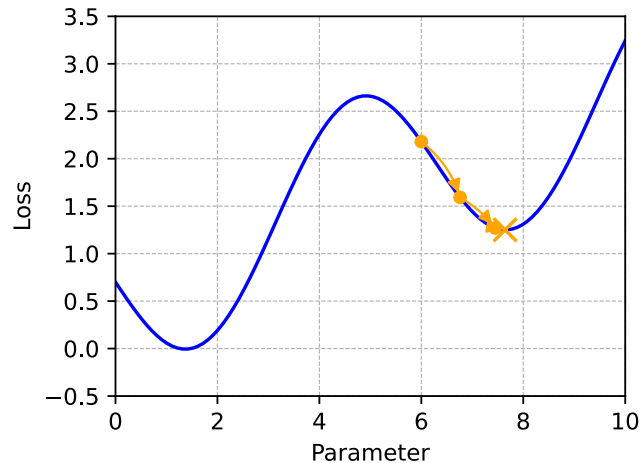


Gradient Descent: What Can Go Wrong?

Best Case



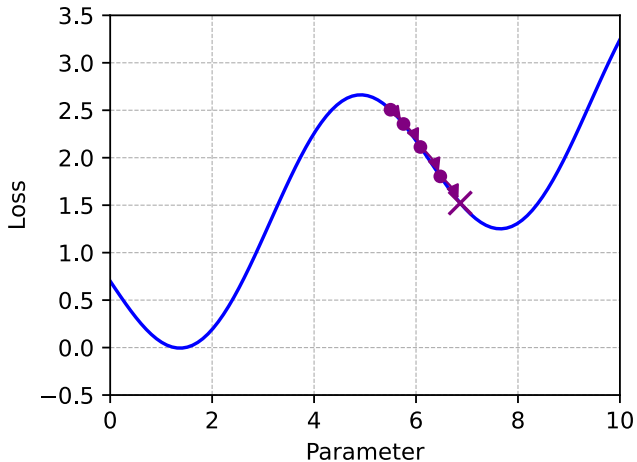
Initialization



- Bumpy loss landscape
- Gets stuck in local minima

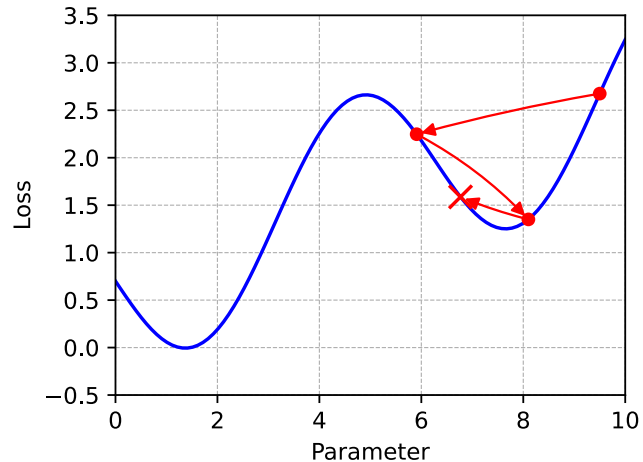
Gradient Descent: What Can Go Wrong?

Learning Rate Too Low



- Slow training

Learning Rate Too High



- Convergence
- Extreme case - NaNs!

Where Do We Get the Gradient From?

Gradient $\nabla_\theta L(\theta|\mathcal{D})$?

```
θ ~ Init
for iteration in range(n):
    J = ∇L(θ)
    θ = θ - ϵ * J.mT
```

Gradient: Linear Regression

Simplified Model:

$$\hat{y} = \mathbf{W}\mathbf{x} + b = \mathbf{w}^\top \mathbf{x} + b$$

Loss function:

$$\begin{aligned} & L(\theta|\mathcal{D}) \\ & = E_{\mathbf{x},y\sim\mathcal{D}}\left[l(\theta|\mathbf{x},y)\right] \\ & = E_{\mathbf{x},y\sim\mathcal{D}}\left[(\mathbf{w}^\top\mathbf{x}+b-y)^2\right] \end{aligned}$$

Gradient:

$$\begin{aligned} & \nabla_{\mathbf{w}} L(\theta|\mathcal{D}) \\ & = \nabla_{\mathbf{w}} E_{\mathbf{x},y\sim\mathcal{D}}\left[l(\theta|\mathbf{x},y)\right] \\ & = E_{\mathbf{x},y\sim\mathcal{D}}\left[\nabla_{\mathbf{w}} l(\theta|\mathbf{x},y)\right] \\ & = E_{\mathbf{x},y\sim\mathcal{D}}\left[\nabla_{\mathbf{w}}(\mathbf{w}^\top\mathbf{x}+b-y)^2\right] \\ & = 2E_{\mathbf{x},y\sim\mathcal{D}}\left[(\mathbf{w}^\top\mathbf{x}+b-y)\nabla_{\mathbf{w}}\mathbf{w}^\top\mathbf{x}\right] \\ & = 2E_{\mathbf{x},y\sim\mathcal{D}}\left[(\mathbf{w}^\top\mathbf{x}+b-y)\mathbf{x}^\top\right] \end{aligned}$$

Gradient: In General

General Linear Regression Model:

$$l(\theta|\mathbf{x}, \mathbf{y}) = (\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{y})^2$$

Binary logistic regression:

$$l(\theta|\mathbf{x}, \mathbf{y}) = y \log \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) + (1 - y) \log(1 - \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}))$$

Multi-class logistic regression:

$$l(\theta|\mathbf{x}, \mathbf{y}) = \log \operatorname{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_y$$

General gradients

Gradients quickly get quite complicated

Optimization - TL;DR

Gradient descent (and variants): workhorse for 99% of deep learning

```
θ ~ Init
for iteration in range(n):
    J = ∇L(θ)
    θ = θ - ϵ * J.mT
```