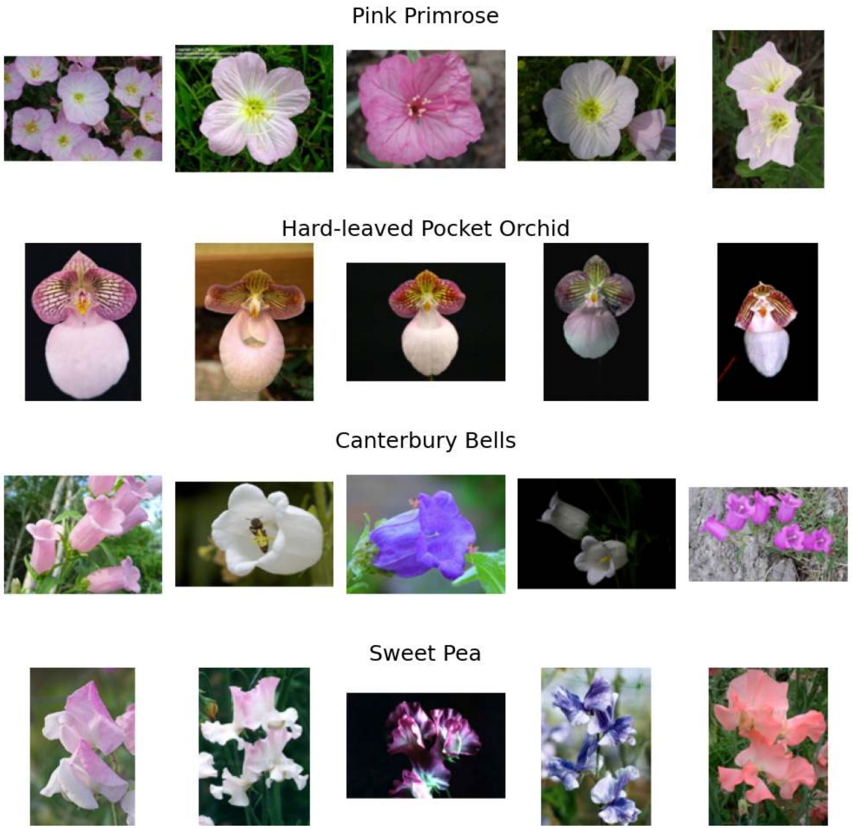
Stochastic Gradient Descent

Recap: Data

Inputs: $\mathbf{x}_i$

Labels: $\mathbf{y}_i$

Dataset: $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$



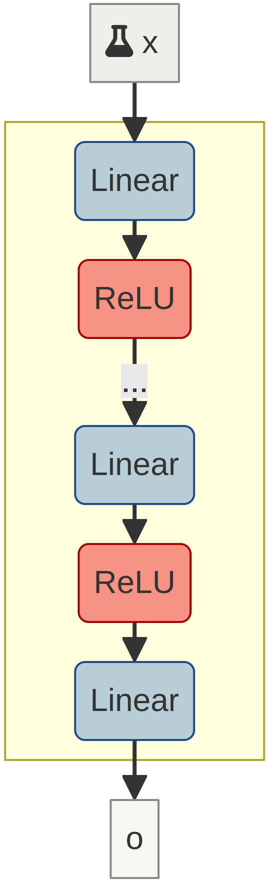
1. Maria-Elena Nilsback and Andrew Zisserman, 102 Category Flower Dataset ↩️

Recap: Model

Deep network

$$f_\theta : \mathbf{R}^n \to \mathbf{R}^C$$

- layers of computation
- parameters θ
- differentiable computation graph



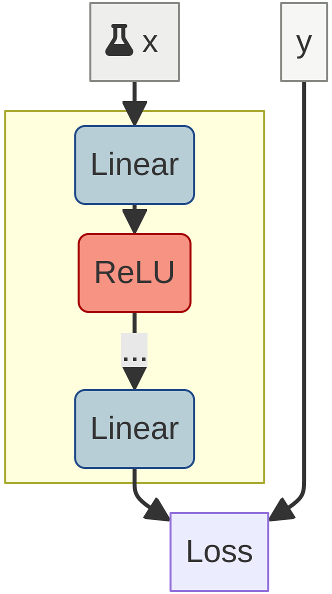
Recap: Loss

Loss function:

$$l(\theta|\mathbf{x}_i, \mathbf{y}_i)$$

Expected loss:

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

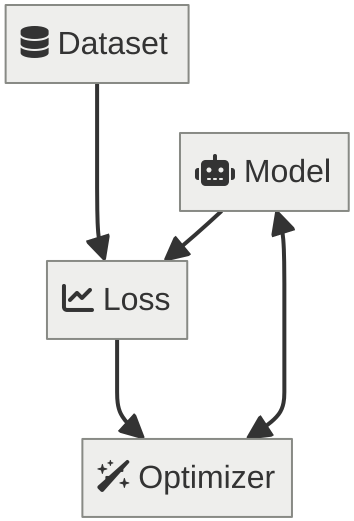


Recap: Training

Find

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

- Deep network $f_\theta : \mathbf{R}^n \to \mathbf{R}^C$
- Dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
- Expected loss $L(\theta|\mathcal{D}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[l(\theta|\mathbf{x}, \mathbf{y})]$



Recap: Gradient Descent

Update rule:

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

Pseudocode

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

Gradient Descent Issues

Slow to compute gradient (in deep networks)

- more parameters
- more data

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

Pseudocode

```
θ ~ Init
for epoch in range(n):
  J = s0
  for (x, y) in dataset:
    J += ∇l(θ|x,y)
  θ = θ - ϵ * J.mT
```

Stochastic Gradient Descent

Vanilla gradient descent uses the expected loss:

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

What if we computed gradients with the partial loss?

$$l(\theta|x,y)$$

Pseudocode

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

Gradient Descent vs. Stochastic Gradient Descent

Gradient Descent

```
θ ~ Init
for epoch in range(n):
  J = s0
  for (x, y) in dataset:
    J += ∇l(θ|x,y)
  θ = θ - ϵ * J.mT
```

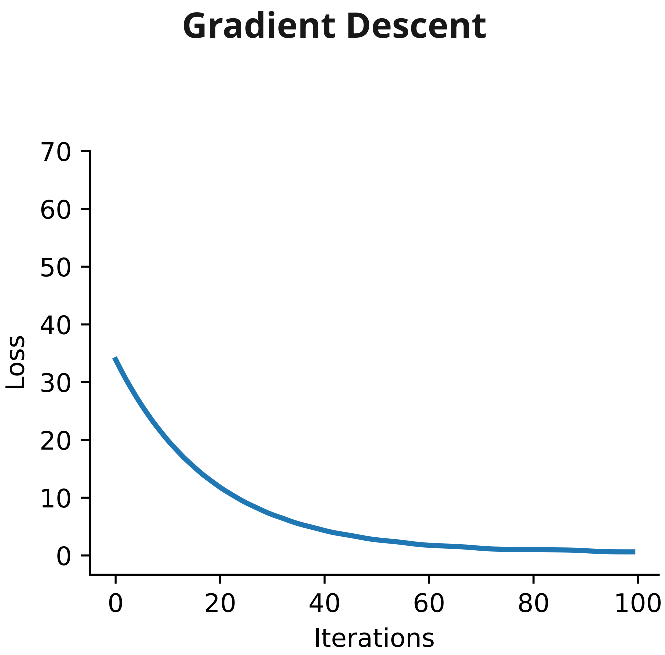
- Convergence guarantees
- Smooth loss

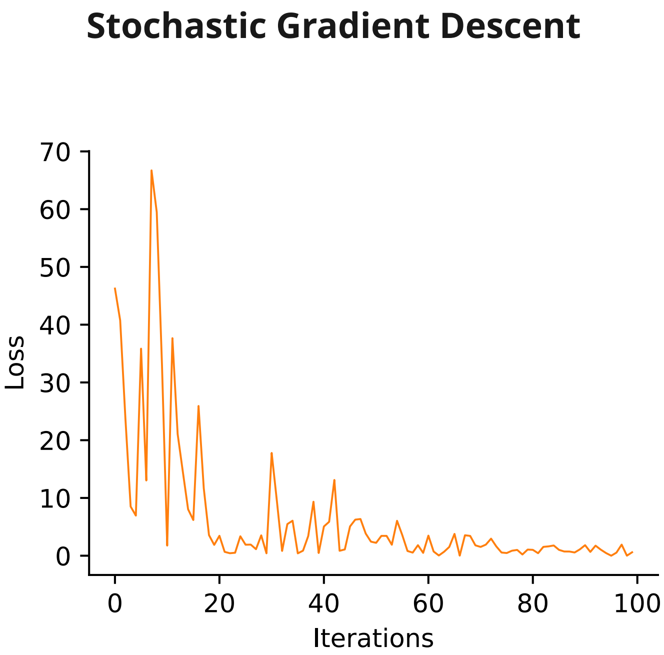
Stochastic Gradient Descent

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

- Faster convergence empirically
- More chaotic convergence

Learning Curves

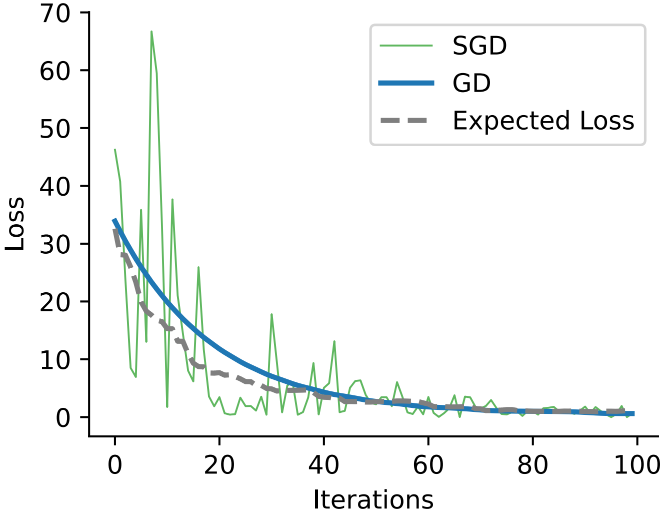




Learning Curves Comparison

Why does the learning curve fluctuate?

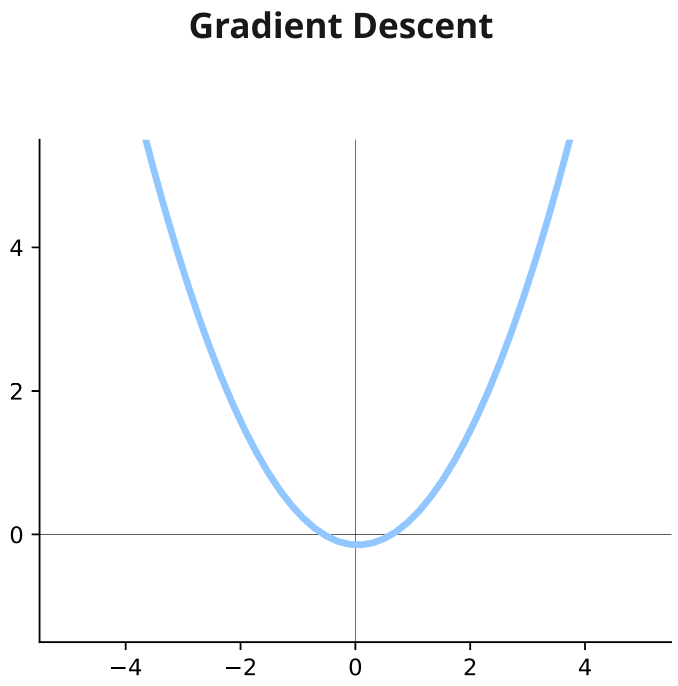
- Loss at each step is evaluated on a single example
- Gradient might be wrong

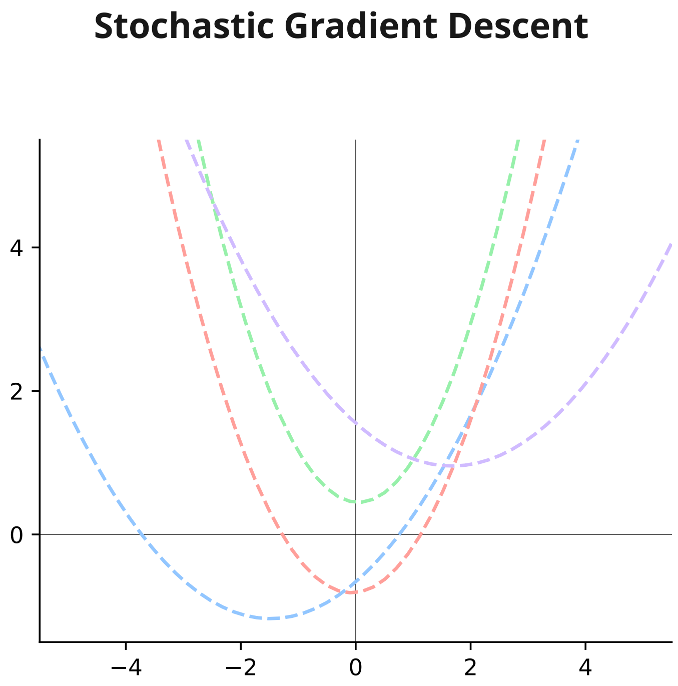


Pseudocode: Stochastic Gradient Descent

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

A 1D Example of Optimization





How Fast Does SGD Converge?

Case 1: $\frac{\partial}{\partial\theta} l(\theta|x_i, y_i) \approx \frac{\partial}{\partial\theta} l(\theta|x_j, y_j)$

- SGD is equivalent to GD
- Faster

Case 2 (reality): $\frac{\partial}{\partial\theta} l(\theta|x_i, y_i) \neq \frac{\partial}{\partial\theta} l(\theta|x_j, y_j)$

- Convergences speed depends on variance of SGD[1]:

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

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

1. Léon Bottou, Stochastic Gradient Descent Tricks. in Neural Networks, Tricks of the Trade, Reloaded ↩️

Stochastic Gradient Descent - TL;DR

We use stochastic gradient descent (SGD) to optimize deep networks

SGD runs more efficiently but has **higher variance** than GD