Classification

CISC 7026: Introduction to Deep Learning

University of Macau

Admin

We will have a make-up lecture later on for the missed lecture

Admin

We will have a make-up lecture later on for the missed lecture

Assignment 1 grades were released on moodle

Admin

We will have a make-up lecture later on for the missed lecture

Assignment 1 grades were released on moodle

The scores were very good, with a mean of approximately 90/100

Admin

We will have a make-up lecture later on for the missed lecture

Assignment 1 grades were released on moodle

The scores were very good, with a mean of approximately 90/100

I am still grading quiz 2, but I had a look at the responses to question 4

Admin

Some requests from students:

Admin

Some requests from students:

1. More coding, less theory

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster
4. Speak slower

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster
4. Speak slower
5. Course is too hard

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster
4. Speak slower
5. Course is too hard
6. Course is perfect

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster
4. Speak slower
5. Course is too hard
6. Course is perfect
7. Move captions to top of screen

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster
4. Speak slower
5. Course is too hard
6. Course is perfect
7. Move captions to top of screen
8. Upload powerpoint before lecture

Admin

Some requests from students:

1. More coding, less theory
2. More math/theory
3. Too easy, go faster
4. Speak slower
5. Course is too hard
6. Course is perfect
7. Move captions to top of screen
8. Upload powerpoint before lecture

There are conflicting student needs

Admin

https://github.com/smorad/um_cisc_7026

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Agenda

1. **Review**
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Review

Last time, we reviewed derivatives

Review

Last time, we reviewed derivatives

$$f'(x) = \frac{d}{dx} f = \frac{df}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}$$

Review

Last time, we reviewed derivatives

$$f'(x) = \frac{d}{dx} f = \frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

and gradients

Review

Last time, we reviewed derivatives

$$f'(x) = \frac{d}{dx} f = \frac{df}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

and gradients

$$\nabla_{\boldsymbol{x}} f\left(\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^\top\right) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_n} \end{bmatrix}^\top$$

Review

Gradients are important in deep learning for two reasons:

Review

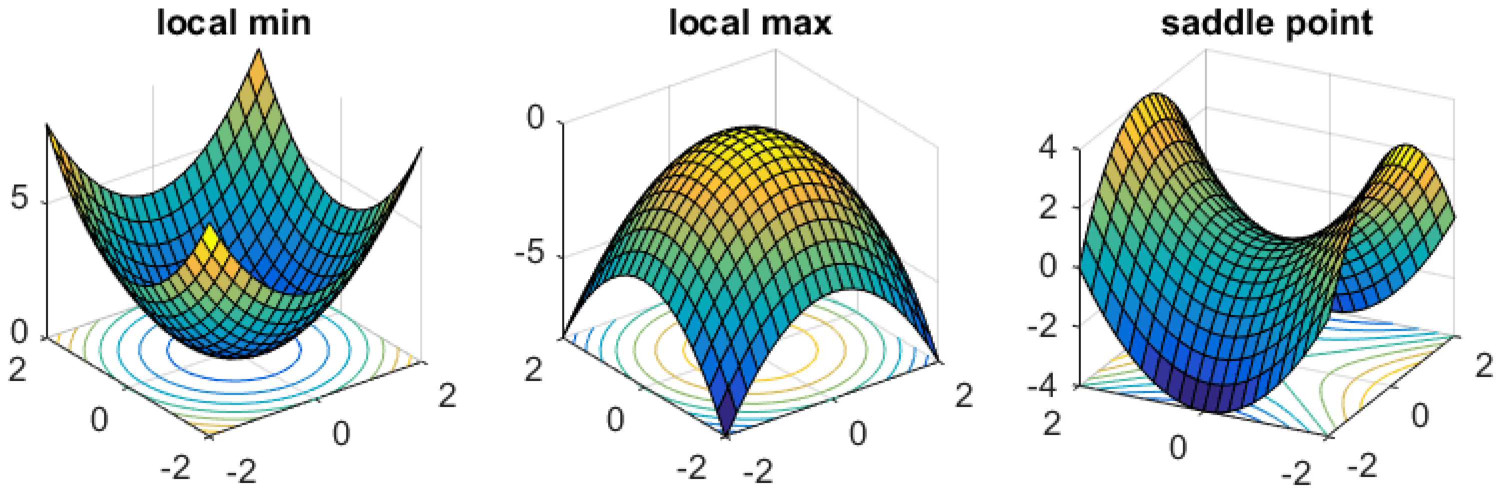
Gradients are important in deep learning for two reasons:

Reason 1: $f(\boldsymbol{x})$ has critical points at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$

Review

Gradients are important in deep learning for two reasons:

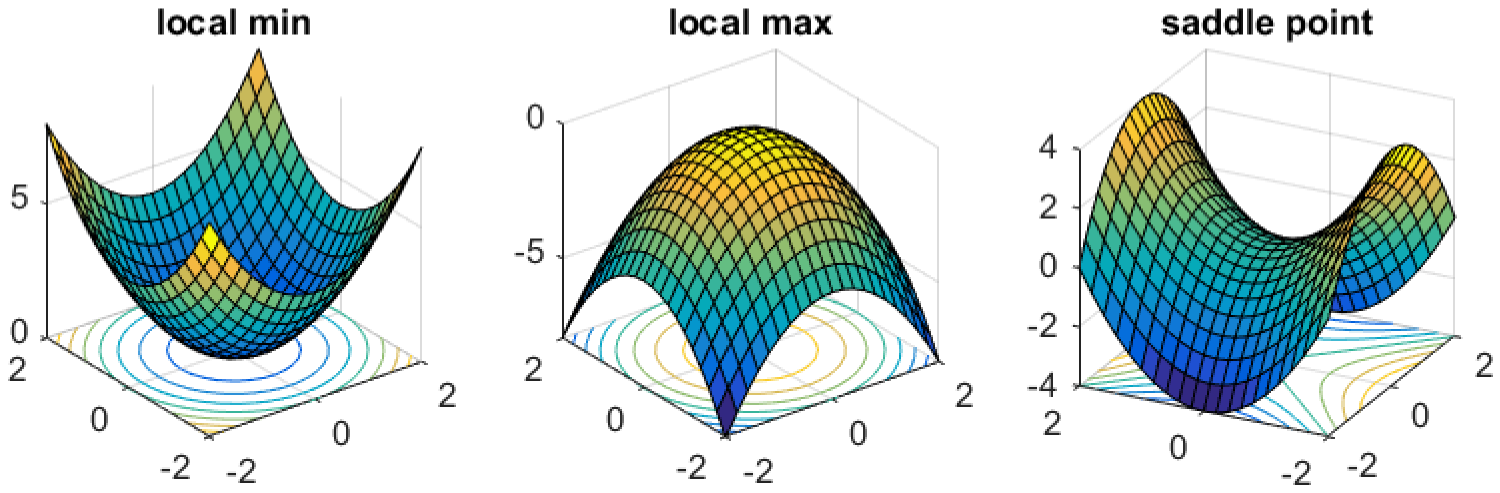
Reason 1: $f(\boldsymbol{x})$ has critical points at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$



Review

Gradients are important in deep learning for two reasons:

Reason 1: $f(\boldsymbol{x})$ has critical points at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$



With optimization, we attempt to find minima of loss functions

Review

Gradients are important in deep learning for two reasons:

Review

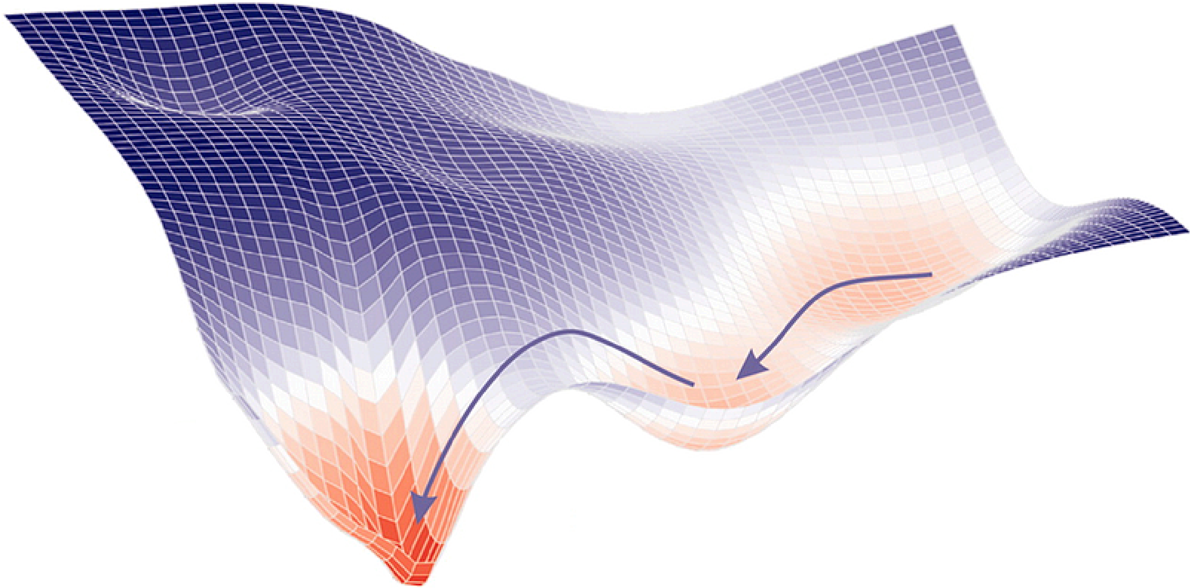
Gradients are important in deep learning for two reasons:

Reason 2: For problems without analytical solutions, the gradient (slope) is necessary for gradient descent

Review

Gradients are important in deep learning for two reasons:

Reason 2: For problems without analytical solutions, the gradient (slope) is necessary for gradient descent



Review

First, we derived the solution to linear regression

Review

First, we derived the solution to linear regression

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) - \boldsymbol{y}_{[i]} \right)^2$$

Review

First, we derived the solution to linear regression

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) - \boldsymbol{y}_{[i]} \right)^2$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = (\boldsymbol{Y} - \boldsymbol{X}_D \boldsymbol{\theta})^\top (\boldsymbol{Y} - \boldsymbol{X}_D \boldsymbol{\theta})$$

Review

First, we derived the solution to linear regression

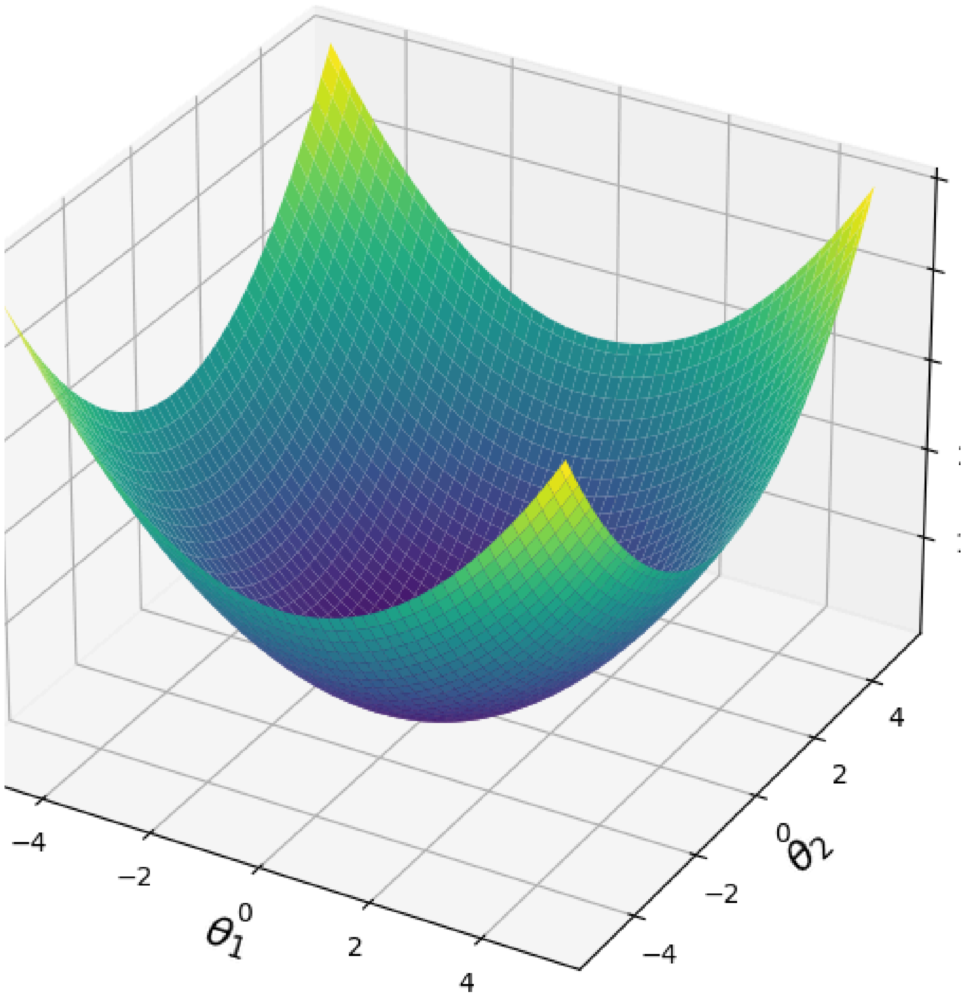
$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) - \boldsymbol{y}_{[i]} \right)^2$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = (\boldsymbol{Y} - \boldsymbol{X}_D \boldsymbol{\theta})^\top (\boldsymbol{Y} - \boldsymbol{X}_D \boldsymbol{\theta})$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \underbrace{\underbrace{(\boldsymbol{Y} - \boldsymbol{X}_D \boldsymbol{\theta})^\top}_{\text{Linear function of } \theta} \quad \underbrace{(\boldsymbol{Y} - \boldsymbol{X}_D \boldsymbol{\theta})}_{\text{Linear function of } \theta}}_{\text{Quadratic function of } \theta}$$

Review

A quadratic function has a single critical point, which must be a global minimum



Review

We found the analytical solution for linear regression by finding where the gradient was zero and solving for $\boldsymbol{\theta}$

Review

We found the analytical solution for linear regression by finding where the gradient was zero and solving for $\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = 0$$

Review

We found the analytical solution for linear regression by finding where the gradient was zero and solving for $\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = 0$$

$$\boldsymbol{\theta} = \left(\boldsymbol{X}_D^\top \boldsymbol{X}_D \right)^{-1} \boldsymbol{X}_D^\top \boldsymbol{Y}$$

Review

We found the analytical solution for linear regression by finding where the gradient was zero and solving for $\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = 0$$

$$\boldsymbol{\theta} = \left(\boldsymbol{X}_D^\top \boldsymbol{X}_D\right)^{-1} \boldsymbol{X}_D^\top \boldsymbol{Y}$$

Which solves

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta})$$

Review

For neural networks, the square error loss is no longer quadratic

Review

For neural networks, the square error loss is no longer quadratic

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (f(x, \boldsymbol{\theta}) - y)^2 \qquad \text{Loss function}$$

Review

For neural networks, the square error loss is no longer quadratic

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (f(x, \boldsymbol{\theta}) - y)^2$$ Loss function

$$f(x, \boldsymbol{\theta}) = \sigma(\theta_0 + \theta_1 x)$$ Neural network model

Review

For neural networks, the square error loss is no longer quadratic

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (f(x, \boldsymbol{\theta}) - y)^2 \qquad \text{Loss function}$$

$$f(x, \boldsymbol{\theta}) = \sigma(\theta_0 + \theta_1 x) \qquad \text{Neural network model}$$

Now, we plug the model f into the loss function

Review

For neural networks, the square error loss is no longer quadratic

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (f(x, \boldsymbol{\theta}) - y)^2$$ Loss function

$$f(x, \boldsymbol{\theta}) = \sigma(\theta_0 + \theta_1 x)$$ Neural network model

Now, we plug the model f into the loss function

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (\sigma(\theta_0 + \theta_1 x) - y)^2$$

Review

For neural networks, the square error loss is no longer quadratic

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (f(x, \boldsymbol{\theta}) - y)^2 \qquad \text{Loss function}$$

$$f(x, \boldsymbol{\theta}) = \sigma(\theta_0 + \theta_1 x) \qquad \text{Neural network model}$$

Now, we plug the model f into the loss function

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (\sigma(\theta_0 + \theta_1 x) - y)^2$$

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = \underbrace{(\sigma(\theta_0 + \theta_1 x) - y)}_{\text{Nonlinear function of } \theta} \underbrace{(\sigma(\theta_0 + \theta_1 x) - y)}_{\text{Nonlinear function of } \theta}$$

Review

For neural networks, the square error loss is no longer quadratic

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (f(x, \boldsymbol{\theta}) - y)^2$$

Loss function

$$f(x, \boldsymbol{\theta}) = \sigma(\theta_0 + \theta_1 x)$$

Neural network model

Now, we plug the model f into the loss function

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = (\sigma(\theta_0 + \theta_1 x) - y)^2$$

$$\mathcal{L}(x, y, \boldsymbol{\theta}) = \underbrace{(\sigma(\theta_0 + \theta_1 x) - y)}_{\text{Nonlinear function of } \theta}\underbrace{(\sigma(\theta_0 + \theta_1 x) - y)}_{\text{Nonlinear function of } \theta}$$

There is no analytical solution for $\boldsymbol{\theta}$

Review

Instead, we found the parameters of a neural network through gradient descent

Review

Instead, we found the parameters of a neural network through gradient descent

Gradient descent is an optimization method for differentiable functions

Review

Instead, we found the parameters of a neural network through gradient descent

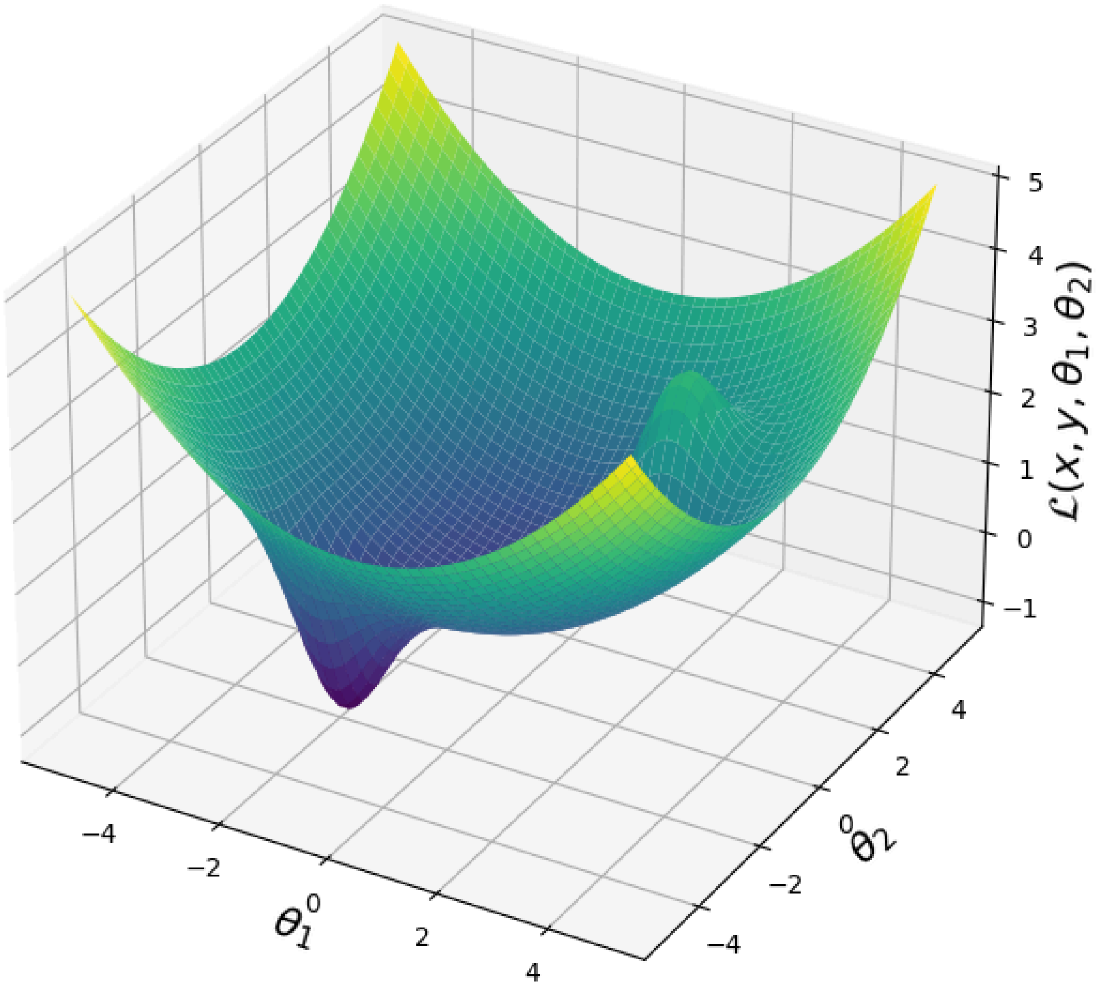
Gradient descent is an optimization method for differentiable functions

We went over both the intuition and mathematical definitions

Review

Review

Review



$\mathcal{L}(x, y, \theta_1, \theta_2)$
5
4
3
2
1
0
−1
θ_1
−4
−2
0
2
4
θ_2
4
2
0
−2
−4

Review

The gradient descent algorithm:

function GRADIENT DESCENT($\boldsymbol{X}, \boldsymbol{Y}, \mathcal{L}, t, \alpha$)

  ▷ Randomly initialize parameters

  $\boldsymbol{\theta} \leftarrow \mathcal{N}(0, 1)$

  for $i \in 1...t$ **do**

    ▷ Compute the gradient of the loss

    $\boldsymbol{J} \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta})$

    ▷ Update the parameters using the negative gradient

    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{J}$

  return $\boldsymbol{\theta}$

Review

We derived the $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ for deep neural networks using the chain rule

Review

We derived the $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ for deep neural networks using the chain rule

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} 2\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) - \boldsymbol{y}_{[i]}\right) \nabla_{\boldsymbol{\theta}} f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right)$$

Review

We derived the $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ for deep neural networks using the chain rule

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} 2\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) - \boldsymbol{y}_{[i]}\right) \color{red}{\nabla_{\boldsymbol{\theta}} f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right)}$$

$$\color{red}{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta})} \color{black}{=} \nabla_{\boldsymbol{\varphi}, \boldsymbol{\psi}, \ldots, \boldsymbol{\xi}} f\left(\boldsymbol{x}, [\boldsymbol{\varphi} \ \ \boldsymbol{\psi} \ \ \ldots \ \ \boldsymbol{\xi}]^\top\right) = \begin{bmatrix} \nabla_{\boldsymbol{\varphi}} f_1(\boldsymbol{x}, \boldsymbol{\varphi}) \\ \nabla_{\boldsymbol{\psi}} f_2(\boldsymbol{z}_1, \boldsymbol{\psi}) \\ \vdots \\ \color{red}{\nabla_{\boldsymbol{\xi}} f_\ell(\boldsymbol{z}_{\ell-1}, \boldsymbol{\xi})} \end{bmatrix}$$

Review

We derived the $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ for deep neural networks using the chain rule

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} 2\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) - \boldsymbol{y}_{[i]}\right) \nabla_{\boldsymbol{\theta}} f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right)$$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\varphi}, \boldsymbol{\psi}, \ldots, \boldsymbol{\xi}} f\left(\boldsymbol{x}, [\boldsymbol{\varphi} \ \ \boldsymbol{\psi} \ \ \ldots \ \ \boldsymbol{\xi}]^\top\right) = \begin{bmatrix} \nabla_{\boldsymbol{\varphi}} f_1(\boldsymbol{x}, \boldsymbol{\varphi}) \\ \nabla_{\boldsymbol{\psi}} f_2(\boldsymbol{z}_1, \boldsymbol{\psi}) \\ \vdots \\ \nabla_{\boldsymbol{\xi}} f_\ell(\boldsymbol{z}_{\ell-1}, \boldsymbol{\xi}) \end{bmatrix}$$

$$\nabla_{\boldsymbol{\xi}} f_\ell(\boldsymbol{z}_{\ell-1}, \boldsymbol{\xi}) = (\sigma(\boldsymbol{\xi}^\top \overline{\boldsymbol{z}}_{\ell-1}) \odot (1 - \sigma(\boldsymbol{\xi}^\top \overline{\boldsymbol{z}}_{\ell-1})))\overline{\boldsymbol{z}}_{\ell-1}^\top$$

Review

We ran into issues computing the gradient of a layer because of the Heaviside step function

Review

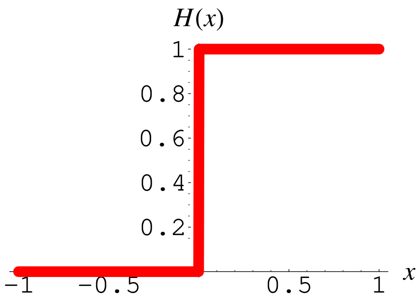
We ran into issues computing the gradient of a layer because of the Heaviside step function

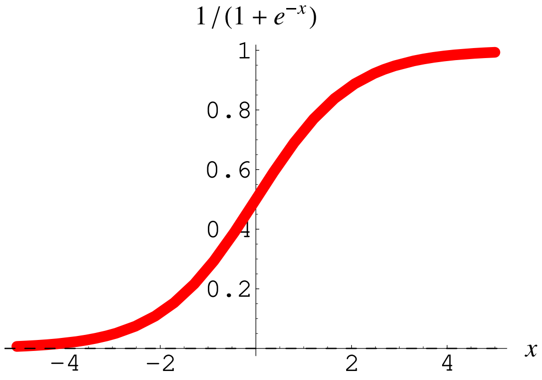
We replaced it with a differentiable (soft) approximation called the sigmoid function

Review

We ran into issues computing the gradient of a layer because of the Heaviside step function

We replaced it with a differentiable (soft) approximation called the sigmoid function





$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Review

In `jax`, we compute the gradient using the `jax.grad` function

Review

In `jax`, we compute the gradient using the `jax.grad` function

```python
import jax

def L(theta, X, Y):
    ...

# Create a new function that is the gradient of L
# Then compute gradient of L for given inputs
J = jax.grad(L)(X, Y, theta)
# Update parameters
alpha = 0.0001
theta = theta - alpha * J
```

Review

In `torch`, we backpropagate through a graph of operations

```
import torch
optimizer = torch.optim.SGD(lr=0.0001)

def L(model, X, Y):
  ...
# Pytorch will record a graph of all operations
# Everytime you do theta @ x, it stores inputs and outputs
loss = L(X, Y, model) # compute loss
# Traverse the graph backward and compute the gradient
loss.backward() # Sets .grad attribute on each parameter
optimizer.step() # Update the parameters using .grad
optimizer.zero_grad() # Set .grad to zero, DO NOT FORGET!!
```

Agenda

1. Review
2. **Torch optimization coding**
3. Classification task
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Agenda

1. Review
2. Torch optimization coding
3. **Classification task**
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Classification task

First, a video of one application of gradient descent

https://youtu.be/kGDO2e_qiyI?si=ZopZKy-6WQ4B0csX

Classification task

First, a video of one application of gradient descent

https://youtu.be/kGDO2e_qiyI?si=ZopZKy-6WQ4B0csX

Time for some interactive coding

https://colab.research.google.com/drive/1W8WVZ8n_9yJCcOqkPVURp_wJUx3EQc5w

Agenda

1. Review
2. **Torch optimization coding**
3. Classification task
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Agenda

1. Review
2. Torch optimization coding
3. **Classification task**
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?
- How far away is this object?

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?
- How far away is this object?

Classification asks which one

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?
- How far away is this object?

Classification asks which one

- Is this a dog or muffin?

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?
- How far away is this object?

Classification asks which one

- Is this a dog or muffin?
- Will it rain tomorrow? Yes or no?

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?
- How far away is this object?

Classification asks which one

- Is this a dog or muffin?
- Will it rain tomorrow? Yes or no?
- What color is this object?

Classification task

Many problems in ML can be reduced to **regression** or **classification**

Regression asks how many

- How long will I live?
- How much rain will there be tomorrow?
- How far away is this object?

Classification asks which one

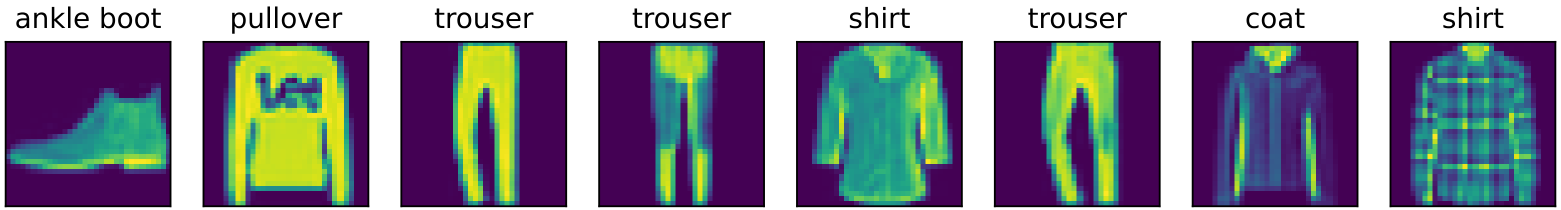
- Is this a dog or muffin?
- Will it rain tomorrow? Yes or no?
- What color is this object?

So far, we only looked at regression. Now, let us look at classification

Task: Given a picture of clothes, predict the text description

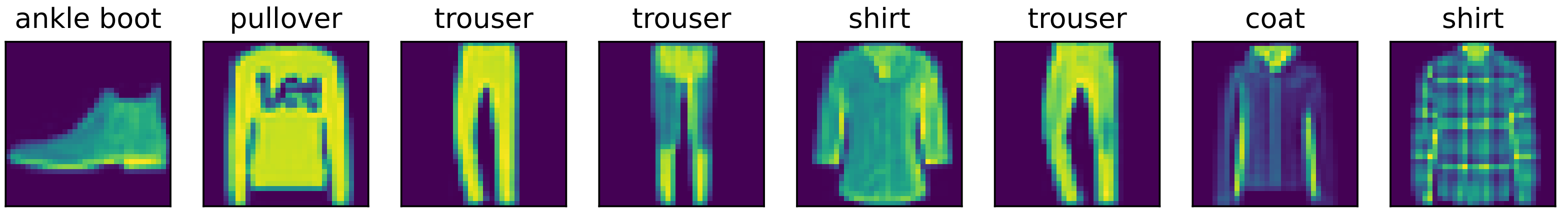
Task: Given a picture of clothes, predict the text description

$X : \mathbb{Z}_{0,255}^{32 \times 32}$



Task: Given a picture of clothes, predict the text description

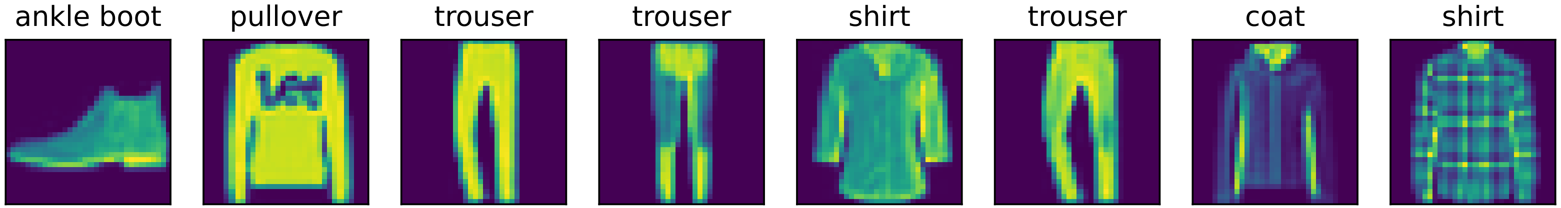
$X : \mathbb{Z}_{0,255}^{32 \times 32}$



Y : {T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot}

Task: Given a picture of clothes, predict the text description

$X : \mathbb{Z}_{0,255}^{32 \times 32}$

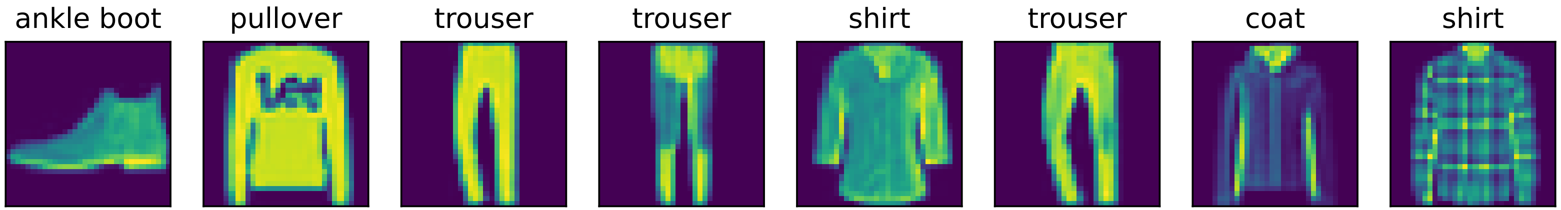


$Y : \{\text{T-shirt, Trouser, Pullover, Dress, Coat,}$
$\text{Sandal, Shirt, Sneaker, Bag, Ankle boot}\}$

Approach: Learn θ that produce **conditional probabilities**

Task: Given a picture of clothes, predict the text description

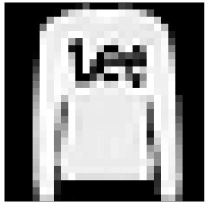
$X : \mathbb{Z}_{0,255}^{32 \times 32}$



$Y : \{\text{T-shirt}, \text{Trouser}, \text{Pullover}, \text{Dress}, \text{Coat},$
$\text{Sandal}, \text{Shirt}, \text{Sneaker}, \text{Bag}, \text{Ankle boot}\}$

Approach: Learn $\boldsymbol{\theta}$ that produce **conditional probabilities**

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{x}) = P\left(\begin{bmatrix} \text{T-Shirt} \\ \text{Trouser} \\ \vdots \end{bmatrix} \;\middle|\; \right.$$



$$\left. \vphantom{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}} \right) = \begin{bmatrix} 0.2 \\ 0.01 \\ \vdots \end{bmatrix}$$

Agenda

1. Review
2. Torch optimization coding
3. **Classification task**
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. **Probability review**
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Classification tasks require **probability theory**, so let us review

Classification tasks require **probability theory**, so let us review

In probability, we have **experiments** and **outcomes**

Classification tasks require **probability theory**, so let us review

In probability, we have **experiments** and **outcomes**

An experiment yields one of many possible outcomes

Classification tasks require **probability theory**, so let us review

In probability, we have **experiments** and **outcomes**

An experiment yields one of many possible outcomes

Experiment	Outcome

Classification tasks require **probability theory**, so let us review

In probability, we have **experiments** and **outcomes**

An experiment yields one of many possible outcomes

Experiment	Outcome
Flip a coin	Heads

Classification tasks require **probability theory**, so let us review

In probability, we have **experiments** and **outcomes**

An experiment yields one of many possible outcomes

Experiment	Outcome
Flip a coin	Heads
Walk outside	Rain

Classification tasks require **probability theory**, so let us review

In probability, we have **experiments** and **outcomes**

An experiment yields one of many possible outcomes

Experiment	Outcome
Flip a coin	Heads
Walk outside	Rain
Grab clothing from closest	Coat

The **sample space** S defines all possible outcomes for an experiment

The **sample space** S defines all possible outcomes for an experiment

Experiment	Sample Space S

The **sample space** S defines all possible outcomes for an experiment

Experiment	Sample Space S
Flip a coin	$S = \{\text{heads}, \text{tails}\}$

The **sample space** S defines all possible outcomes for an experiment

Experiment	Sample Space S
Flip a coin	$S = \{\text{heads}, \text{tails}\}$
Walk outside	$S = \{\text{rain}, \text{sun}, \text{wind}, \text{cloud}\}$

The **sample space** S defines all possible outcomes for an experiment

Experiment	Sample Space S
Flip a coin	$S = \{\text{heads}, \text{tails}\}$
Walk outside	$S = \{\text{rain}, \text{sun}, \text{wind}, \text{cloud}\}$
Take clothing from closet	$S = \{\text{T-shirt}, \text{Trouser}, \text{Pullover}, \text{Dress}, \text{Coat}, \text{Sandal}, \text{Shirt}, \text{Sneaker}, \text{Bag}, \text{Ankle boot}\}$

An **event** E is a specific subset of the sample space

An **event** E is a specific subset of the sample space

Experiment	Sample Space	Event

An **event** E is a specific subset of the sample space

Experiment	Sample Space	Event
Flip a coin	$S = \{\text{heads}, \text{tails}\}$	$E = \{\text{heads}\}$

An **event** E is a specific subset of the sample space

Experiment	Sample Space	Event
Flip a coin	$S = \{\text{heads}, \text{tails}\}$	$E = \{\text{heads}\}$
Walk outside	$S = \{\text{rain}, \text{sun}, \text{wind}, \text{cloud}\}$	$E = \{\text{rain}, \text{wind}\}$

An **event** E is a specific subset of the sample space

Experiment	Sample Space	Event
Flip a coin	$S = \{\text{heads}, \text{tails}\}$	$E = \{\text{heads}\}$
Walk outside	$S = \{\text{rain}, \text{sun}, \text{wind}, \text{cloud}\}$	$E = \{\text{rain}, \text{wind}\}$
Take from closet	$\{\text{T-shirt}, \text{Trouser}, \text{Pullover}, \text{Dress}, \text{Coat}, \text{Sandal}, \text{Shirt}, \text{Sneaker}, \text{Bag}, \text{Ankle boot}\}$	$E = \{\text{Shirt}, \text{T-Shirt}, \text{Coat}\}$

The **probability** measures how likely an event is to occur

The **probability** measures how likely an event is to occur

The probability must be between 0 (never occurs) and 1 (always occurs)

The **probability** measures how likely an event is to occur

The probability must be between 0 (never occurs) and 1 (always occurs)

$$0 \leq P(E) \leq 1$$

The **probability** measures how likely an event is to occur

The probability must be between 0 (never occurs) and 1 (always occurs)

$$0 \leq P(E) \leq 1$$

Experiment	Probabilities

The **probability** measures how likely an event is to occur

The probability must be between 0 (never occurs) and 1 (always occurs)

$$0 \leq P(E) \leq 1$$

Experiment	Probabilities
Flip a coin	$P(\text{heads}) = 0.5$

The **probability** measures how likely an event is to occur

The probability must be between 0 (never occurs) and 1 (always occurs)

$$0 \leq P(E) \leq 1$$

Experiment	Probabilities
Flip a coin	$P(\text{heads}) = 0.5$
Walk outside	$P(\text{rain}) = 0.15$

The **probability** measures how likely an event is to occur

The probability must be between 0 (never occurs) and 1 (always occurs)

$$0 \leq P(E) \leq 1$$

Experiment	Probabilities
Flip a coin	$P(\text{heads}) = 0.5$
Walk outside	$P(\text{rain}) = 0.15$
Take from closet	$P(\text{Shirt}) = 0.1$

When we define P as a function, we call it a **distribution**

When we define P as a function, we call it a **distribution**

$$P : E \mapsto (0, 1)$$

When we define P as a function, we call it a **distribution**

$$P : E \mapsto (0, 1)$$

The probabilities (distribution) over the sample space S must sum to one

When we define P as a function, we call it a **distribution**

$$P : E \mapsto (0, 1)$$

The probabilities (distribution) over the sample space S must sum to one

$$\sum_{x \in S} P(x) = 1$$

When we define P as a function, we call it a **distribution**

$$P : E \mapsto (0, 1)$$

The probabilities (distribution) over the sample space S must sum to one

$$\sum_{x \in S} P(x) = 1$$

Flip a coin $\{P(\text{heads}) = 0.5, P(\text{tails}) = 0.5\}$

When we define P as a function, we call it a **distribution**

$$P : E \mapsto (0, 1)$$

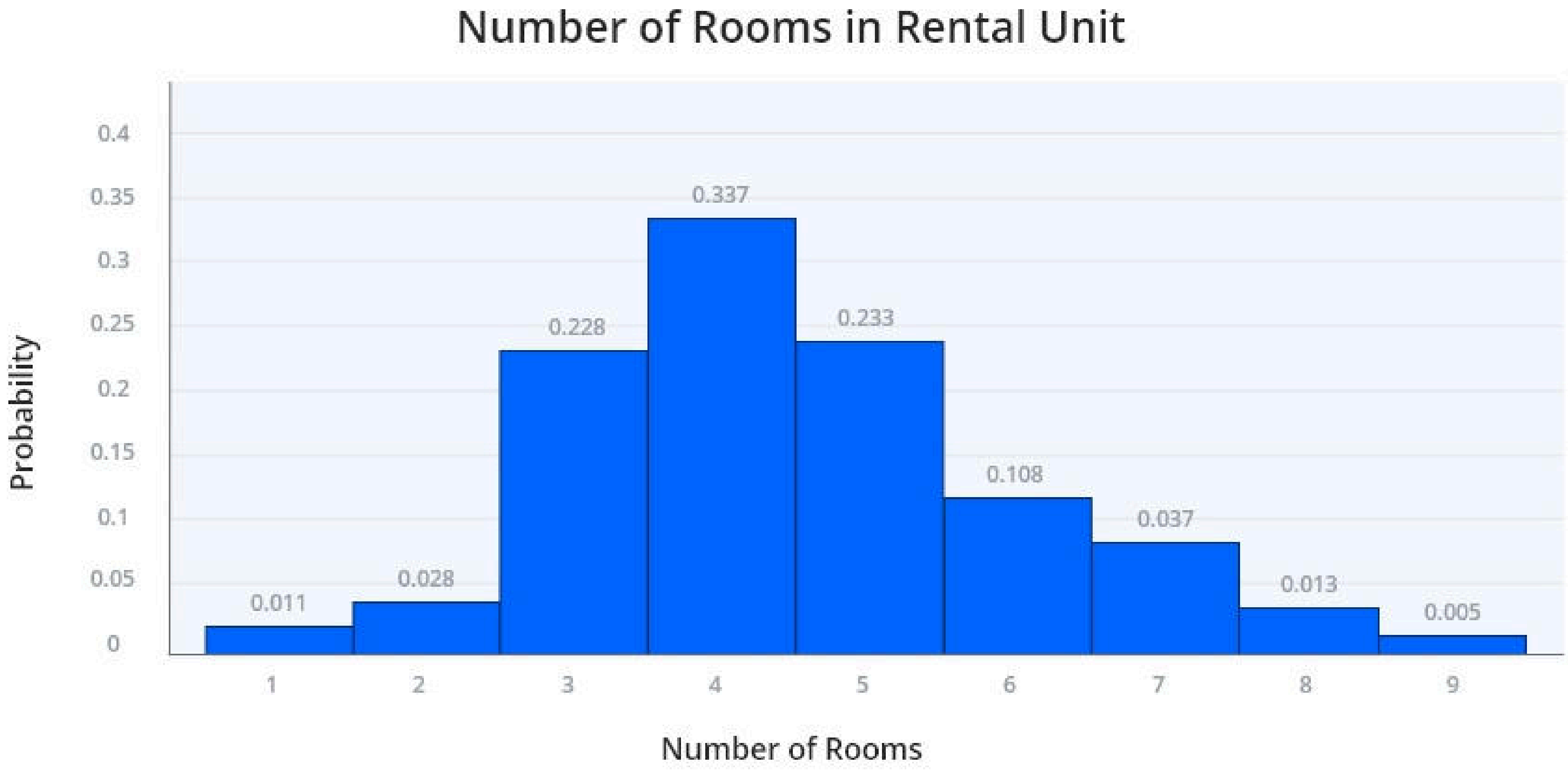
The probabilities (distribution) over the sample space S must sum to one

$$\sum_{x \in S} P(x) = 1$$

Flip a coin $\{P(\text{heads}) = 0.5, P(\text{tails}) = 0.5\}$

Take clothing from closet $\{P(\text{T-shirt}) = 0.1, P(\text{Trouser}) = 0.08, P(\text{Pullover}) = 0.12, ...\}$

The distribution is a function, so we can plot it


Number of Rooms in Rental Unit
Probability
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0.011
0.028
0.228
0.337
0.233
0.108
0.037
0.013
0.005
1
2
3
4
5
6
7
8
9
Number of Rooms

Events can overlap with each other

Events can overlap with each other

- Disjoint events

Events can overlap with each other

- Disjoint events
- Conditionally dependent events

Two events A, B are **disjoint** if either A occurs or B occurs, **but not both**

Two events A, B are **disjoint** if either A occurs or B occurs, **but not both**

With disjoint events, $P(A \cap B) = 0$

Two events A, B are **disjoint** if either A occurs or B occurs, **but not both**

With disjoint events, $P(A \cap B) = 0$

Flip a coin

$$P(\text{Heads}) = 0.5, P(\text{Tails}) = 0.5$$
$$P(\text{Heads} \cap \text{Tails}) = 0$$

Two events A, B are **disjoint** if either A occurs or B occurs, **but not both**

With disjoint events, $P(A \cap B) = 0$

Flip a coin

$$P(\text{Heads}) = 0.5, P(\text{Tails}) = 0.5$$
$$P(\text{Heads} \cap \text{Tails}) = 0$$

Be careful!

Walk outside

$$P(\text{Rain}) = 0.1, P(\text{Cloud}) = 0.2$$
$$P(\text{Rain} \cap \text{Cloud}) \neq 0$$

If A, B are not disjoint, they are **conditionally dependent**

If A, B are not disjoint, they are **conditionally dependent**

$$P(\text{cloud}) = 0.2, P(\text{rain}) = 0.1$$

If A, B are not disjoint, they are **conditionally dependent**

$$P(\text{cloud}) = 0.2, P(\text{rain}) = 0.1$$

$$P(\text{rain} \mid \text{cloud}) = 0.5$$

If A, B are not disjoint, they are **conditionally dependent**

$$P(\text{cloud}) = 0.2, P(\text{rain}) = 0.1$$

$$P(\text{rain} \mid \text{cloud}) = 0.5$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

If A, B are not disjoint, they are **conditionally dependent**

$$P(\text{cloud}) = 0.2, P(\text{rain}) = 0.1$$

$$P(\text{rain} \mid \text{cloud}) = 0.5$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Walk outside

$$P(\text{Rain} \cap \text{Cloud}) = 0.1$$

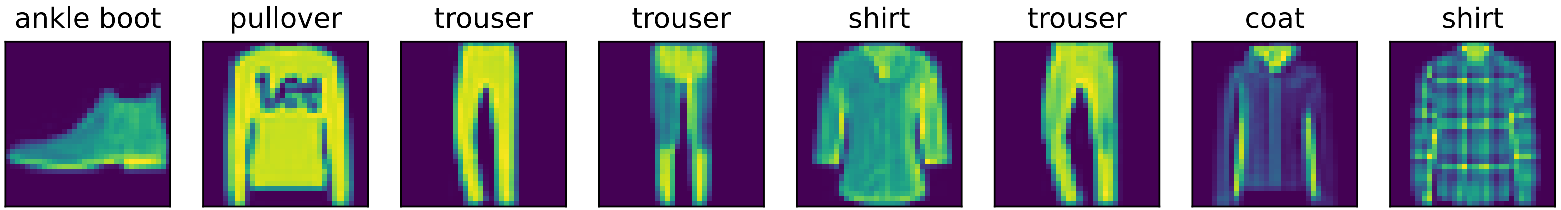
$$P(\text{Cloud}) = 0.2$$

$$P(\text{Rain} \mid \text{Cloud}) = \frac{0.1}{0.2} = 0.5$$

Task: Given a picture of clothes, predict the text description

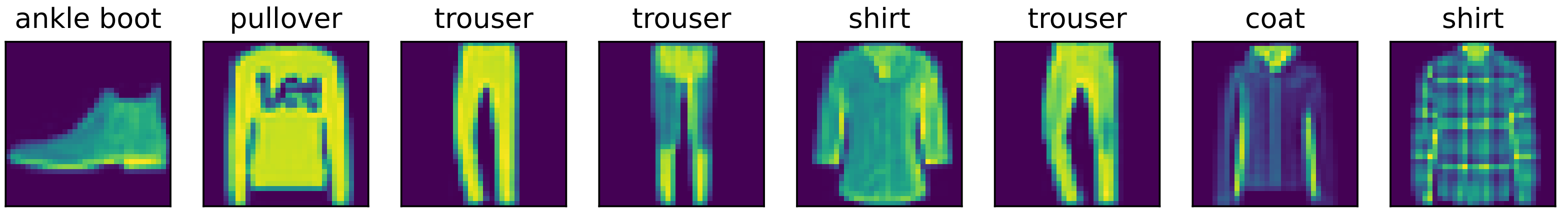
Task: Given a picture of clothes, predict the text description

$X : \mathbb{Z}_{0,255}^{32 \times 32}$



Task: Given a picture of clothes, predict the text description

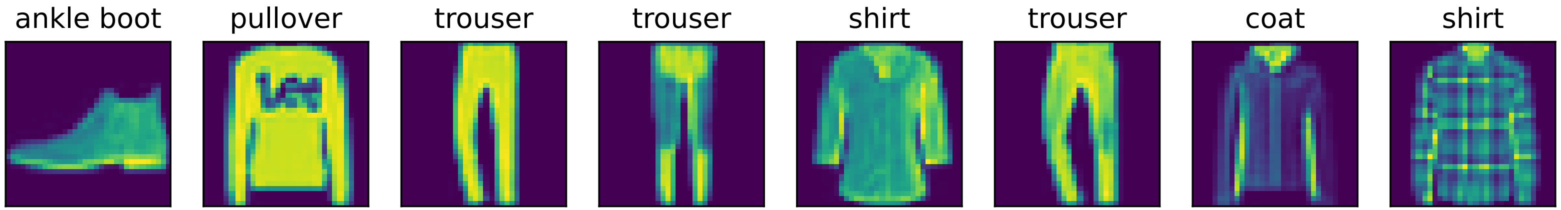
$X : \mathbb{Z}_{0,255}^{32 \times 32}$



$Y : \{\text{T-shirt, Trouser, Pullover, Dress, Coat,}$
$\text{Sandal, Shirt, Sneaker, Bag, Ankle boot}\}$

Task: Given a picture of clothes, predict the text description

$X : \mathbb{Z}_{0,255}^{32 \times 32}$

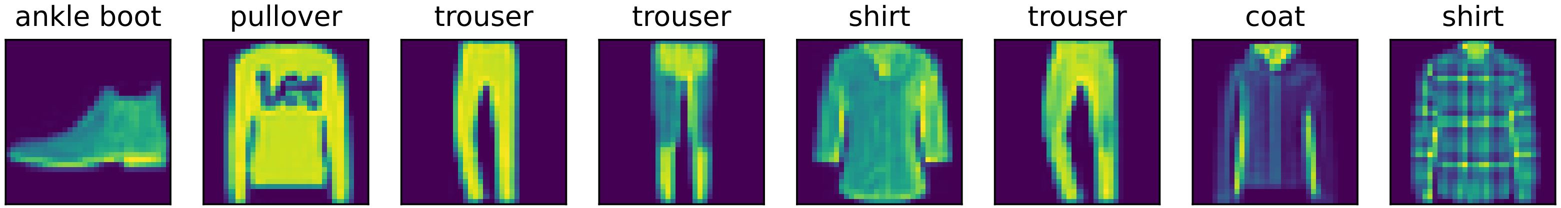


$Y : \{\text{T-shirt, Trouser, Pullover, Dress, Coat,}$
$\text{Sandal, Shirt, Sneaker, Bag, Ankle boot}\}$

Approach: Learn $\boldsymbol{\theta}$ that produce **conditional probabilities**

Task: Given a picture of clothes, predict the text description

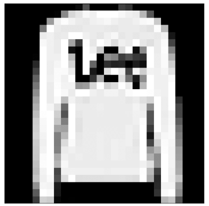
$X : \mathbb{Z}_{0,255}^{32 \times 32}$



$Y : \{\text{T-shirt, Trouser, Pullover, Dress, Coat,}$
$\text{Sandal, Shirt, Sneaker, Bag, Ankle boot}\}$

Approach: Learn $\boldsymbol{\theta}$ that produce **conditional probabilities**

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{x}) = P\left(\begin{bmatrix} \text{T-Shirt} \\ \text{Trouser} \\ \vdots \end{bmatrix} \middle| \right.$$



$$\left. \vphantom{\begin{bmatrix} a \\ b \\ c \end{bmatrix}} \right) = \begin{bmatrix} 0.2 \\ 0.01 \\ \vdots \end{bmatrix}$$

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. **Probability review**
5. Define model f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. **Define model** f
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

We will again start with a multivariate linear model

We will again start with a multivariate linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

We will again start with a multivariate linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

We want our model to predict the probability of each outcome

We will again start with a multivariate linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

We want our model to predict the probability of each outcome

Question: What is the function signature of f?

We will again start with a multivariate linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

We want our model to predict the probability of each outcome

Question: What is the function signature of f?

Answer: $f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$

We will again start with a multivariate linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

We want our model to predict the probability of each outcome

Question: What is the function signature of f?

Answer: $f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$

Question: Can we use this model to predict probabilities?

We will again start with a multivariate linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

We want our model to predict the probability of each outcome

Question: What is the function signature of f?

Answer: $f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$

Question: Can we use this model to predict probabilities?

Answer: No! Because probabilities must be $\in (0, 1)$ and sum to one

How can we represent a probability distribution for a neural network?

How can we represent a probability distribution for a neural network?

$$\boldsymbol{v} = \left\{ \begin{bmatrix} v_1 \\ \vdots \\ v_{d_y} \end{bmatrix} \middle| \quad \sum_{i=1}^{d_y} v_i = 1; \quad v_i \in (0, 1) \right\}$$

How can we represent a probability distribution for a neural network?

$$\boldsymbol{v} = \left\{ \begin{bmatrix} v_1 \\ \vdots \\ v_{d_y} \end{bmatrix} \middle| \; \sum_{i=1}^{d_y} v_i = 1; \quad v_i \in (0, 1) \right\}$$

There is special notation for this vector, called the **simplex**

How can we represent a probability distribution for a neural network?

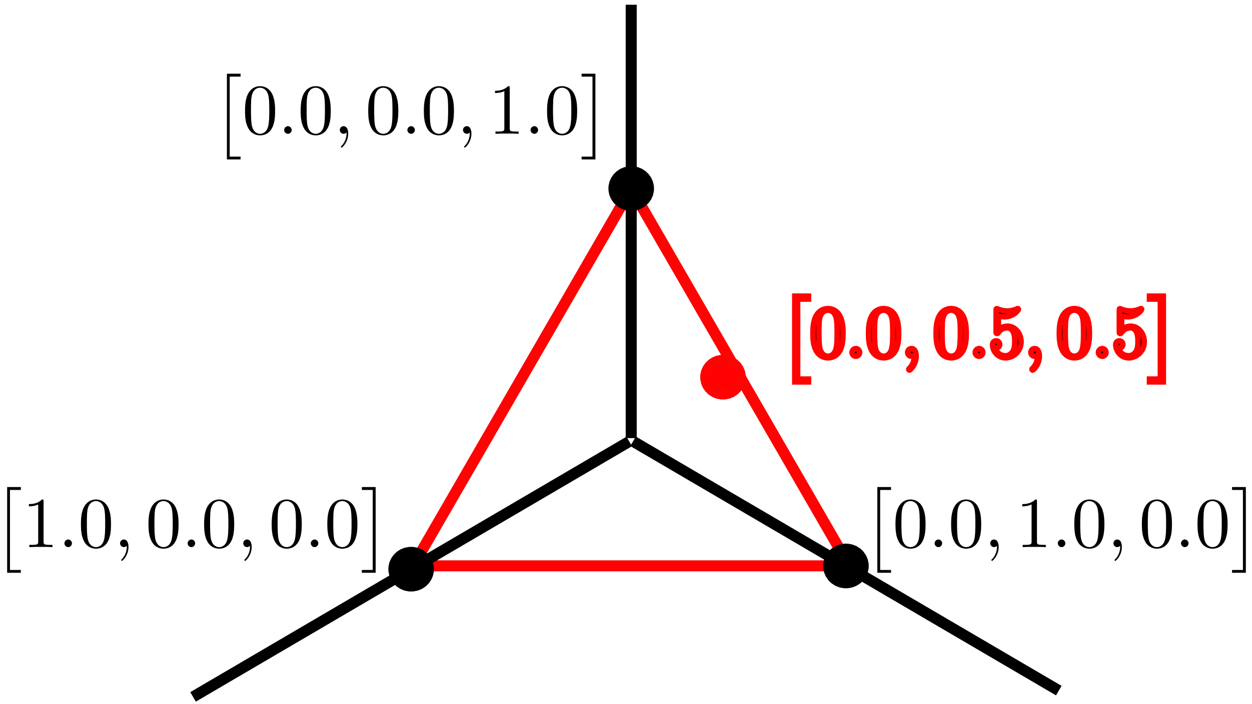
$$\boldsymbol{v} = \left\{ \begin{bmatrix} v_1 \\ \vdots \\ v_{d_y} \end{bmatrix} \middle| \quad \sum_{i=1}^{d_y} v_i = 1; \quad v_i \in (0, 1) \right\}$$

There is special notation for this vector, called the **simplex**

$$\Delta^{d_y - 1}$$

The simplex Δ^k is an $k - 1$-dimensional triangle in k-dimensional space

The simplex Δ^k is an $k - 1$-dimensional triangle in k-dimensional space



It has only $k - 1$ free variables, because $x_k = 1 - \sum_{i=1}^{k-1} x_i$

$$f: \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

So we need a function that maps to the simplex

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

So we need a function that maps to the simplex

$$g : \mathbb{R}^{d_y} \mapsto \Delta^{d_y - 1}$$

$$f : \mathbb{R}^{d_x} \times \Theta \mapsto \mathbb{R}^{d_y}$$

So we need a function that maps to the simplex

$$g : \mathbb{R}^{d_y} \mapsto \Delta^{d_y - 1}$$

Then, we can combine f and g

$$g(f) : \mathbb{R}^{d_x} \times \Theta \mapsto \Delta^{d_y - 1}$$

We need a function g

$$g : \mathbb{R}^{d_y} \mapsto \Delta^{d_y - 1}$$

We need a function g

$$g : \mathbb{R}^{d_y} \mapsto \Delta^{d_y - 1}$$

There are many functions that can do this

We need a function g

$$g : \mathbb{R}^{d_y} \mapsto \Delta^{d_y - 1}$$

There are many functions that can do this

One example is dividing by the L_1 norm:

$$g(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\sum_{i=1}^{d_y} x_i}$$

We need a function g

$$g : \mathbb{R}^{d_y} \mapsto \Delta^{d_y - 1}$$

There are many functions that can do this

One example is dividing by the L_1 norm:

$$g(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\sum_{i=1}^{d_y} x_i}$$

In deep learning we often use the **softmax** function. When combined with the classification loss the gradient is linear, making learning faster

The softmax function maps real numbers to the simplex (probabilities)

$$\text{softmax} : \mathbb{R}^k \mapsto \Delta^{k-1}$$

The softmax function maps real numbers to the simplex (probabilities)

$$\text{softmax} : \mathbb{R}^k \mapsto \Delta^{k-1}$$

$$\text{softmax}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \right) = \frac{e^{\boldsymbol{x}}}{\sum_{i=1}^{k} e^{x_i}} = \begin{bmatrix} \frac{e^{x_1}}{e^{x_1} + e^{x_2} + \ldots e^{x_k}} \\ \frac{e^{x_2}}{e^{x_1} + e^{x_2} + \ldots e^{x_k}} \\ \vdots \\ \frac{e^{x_k}}{e^{x_1} + e^{x_2} + \ldots e^{x_k}} \end{bmatrix}$$

The softmax function maps real numbers to the simplex (probabilities)

$$\text{softmax} : \mathbb{R}^k \mapsto \Delta^{k-1}$$

$$\text{softmax}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \right) = \frac{e^{\boldsymbol{x}}}{\sum_{i=1}^{k} e^{x_i}} = \begin{bmatrix} \frac{e^{x_1}}{e^{x_1} + e^{x_2} + \ldots e^{x_k}} \\ \frac{e^{x_2}}{e^{x_1} + e^{x_2} + \ldots e^{x_k}} \\ \vdots \\ \frac{e^{x_k}}{e^{x_1} + e^{x_2} + \ldots e^{x_k}} \end{bmatrix}$$

If we attach it to our linear model, we can output probabilities!

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \text{softmax}(\boldsymbol{\theta}^\top \boldsymbol{x})$$

And naturally, we can use the same method for a deep neural network

$$f_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \sigma(\boldsymbol{\varphi}^\top \overline{\boldsymbol{x}})$$

$$\vdots$$

$$f_\ell(\boldsymbol{x}, \boldsymbol{\xi}) = \text{softmax}(\boldsymbol{\xi}^\top \overline{\boldsymbol{x}})$$

And naturally, we can use the same method for a deep neural network

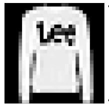
$$f_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \sigma(\boldsymbol{\varphi}^\top \overline{\boldsymbol{x}})$$

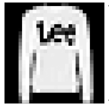
$$\vdots$$

$$f_\ell(\boldsymbol{x}, \boldsymbol{\xi}) = \operatorname{softmax}(\boldsymbol{\xi}^\top \overline{\boldsymbol{x}})$$

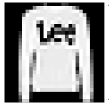
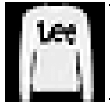
Now, our neural network can output probabilities

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} P\left(\text{Ankle boot} \mid \quad \right) \\ P\left(\text{Bag} \mid \quad \right) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.08 \\ \vdots \end{bmatrix}$$

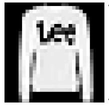


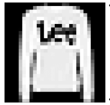


Question: Why do we output probabilities instead of a binary values

$f(\boldsymbol{x}, \boldsymbol{\theta}) =$ [$P\big(\text{Shirt} \mid$  $\big)$, $P\big(\text{Bag} \mid$  $\big)$, $\vdots$] $= \begin{bmatrix} 0.25 \\ 0.08 \\ \vdots \end{bmatrix}; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$

Question: Why do we output probabilities instead of a binary values

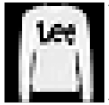


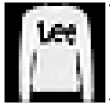


$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} P\left(\text{Shirt} \mid \quad \right) \\ P\left(\text{Bag} \mid \quad \right) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.08 \\ \vdots \end{bmatrix}; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$$

Answer 1: Outputting probabilities results in differentiable functions

Question: Why do we output probabilities instead of a binary values

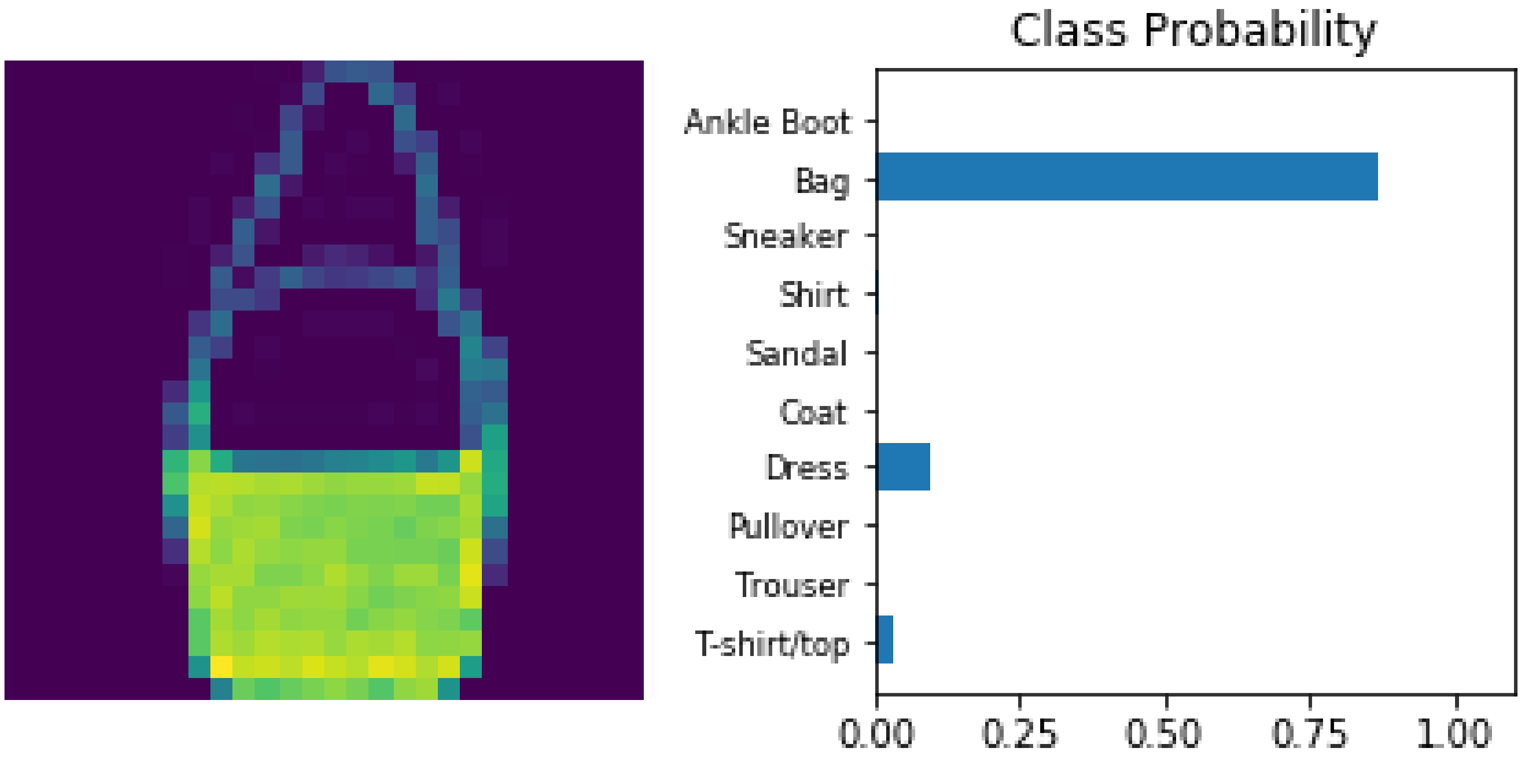




$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} P\left(\text{Shirt} \mid \quad \right) \\ P\left(\text{Bag} \mid \quad \right) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.08 \\ \vdots \end{bmatrix}; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$$

Answer 1: Outputting probabilities results in differentiable functions

Answer 2: We report uncertainty, which is useful in many applications


Class Probability
Ankle Boot
Bag
Sneaker
Shirt
Sandal
Coat
Dress
Pullover
Trouser
T-shirt/top
0.00
0.25
0.50
0.75
1.00

Agenda

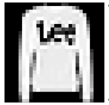
1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. **Define model f**
6. Define loss function $\mathcal{L}$
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

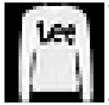
Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. **Define loss function $\mathcal{L}$**
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

We consider the label $\boldsymbol{y}_{[i]}$ as a conditional distribution

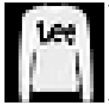
$P\left(\boldsymbol{y}_{[i]} \mid \boldsymbol{x}_{[i]}\right) =$

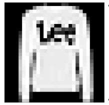
$P\big(\text{Shirt} \mid$  $\big)$

$P\big(\text{Bag} \mid$  $\big)$

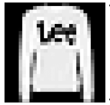
$= \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

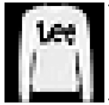
We consider the label $\boldsymbol{y}_{[i]}$ as a conditional distribution

$P(\boldsymbol{y}_{[i]} \mid \boldsymbol{x}_{[i]}) = \Bigg[$ $P(\text{Shirt} \mid$  $)$

$P(\text{Bag} \mid$  $)$ $\Bigg] = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

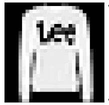
Our neural network also outputs some distribution

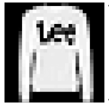
$f(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}) = \Bigg[$ $P(\text{Shirt} \mid$  $)$

$P(\text{Bag} \mid$  $)$ $\Bigg] = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$

We consider the label $\boldsymbol{y}_{[i]}$ as a conditional distribution

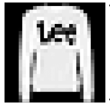
$$P\left(\boldsymbol{y}_{[i]} \mid \boldsymbol{x}_{[i]}\right) = \begin{bmatrix} P(\text{Shirt} \mid \square_1) \\ P(\text{Bag} \mid \square_2) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

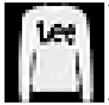
$\square_1$: 

$\square_2$: 

Our neural network also outputs some distribution

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} P(\text{Shirt} \mid \square_3) \\ P(\text{Bag} \mid \square_4) \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$$

$\square_3$: 

$\square_4$: 

What loss function should we use for classification?

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y}_{[i]} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y}_{[i]} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We could use the square error like linear regression

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y}_{[i]} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We could use the square error like linear regression

$$(0.6 - 1)^2 + (0.4 - 0)^2$$

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y}_{[i]} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We could use the square error like linear regression

$$(0.6 - 1)^2 + (0.4 - 0)^2$$

This can work, but in reality it does not work well

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y}_{[i]} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We could use the square error like linear regression

$$(0.6 - 1)^2 + (0.4 - 0)^2$$

This can work, but in reality it does not work well

Instead, we use the **cross-entropy loss**

$$f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}\right) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y}_{[i]} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We could use the square error like linear regression

$$(0.6 - 1)^2 + (0.4 - 0)^2$$

This can work, but in reality it does not work well

Instead, we use the **cross-entropy loss**

Let us derive it

We can model $f(\boldsymbol{x}, \boldsymbol{\theta})$ and $\boldsymbol{y}$ as probability distributions

We can model $f(\boldsymbol{x}, \boldsymbol{\theta})$ and $\boldsymbol{y}$ as probability distributions

How do we measure the difference between probability distributions?

We can model $f(\boldsymbol{x}, \boldsymbol{\theta})$ and $\boldsymbol{y}$ as probability distributions

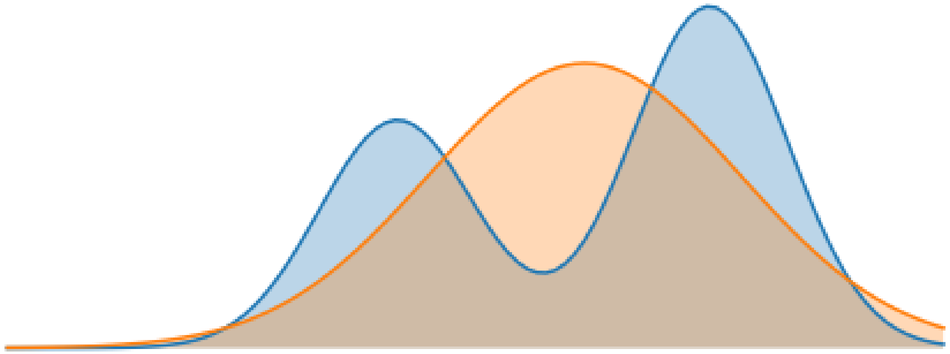
How do we measure the difference between probability distributions?

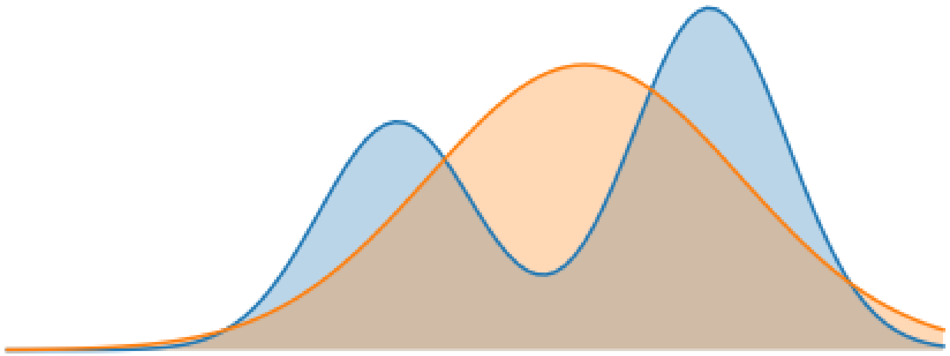
We use the **Kullback-Leibler Divergence (KL)**

We can model $f(\boldsymbol{x}, \boldsymbol{\theta})$ and $\boldsymbol{y}$ as probability distributions

How do we measure the difference between probability distributions?

We use the **Kullback-Leibler Divergence (KL)**





$$\mathrm{KL}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

First, write down KL-divergence

$$\mathrm{KL}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

First, write down KL-divergence

$$\mathrm{KL}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Plug in our two distributions $P = f$ and $Q = \boldsymbol{y}$

$$\mathrm{KL}(P(\boldsymbol{y} \mid \boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log \frac{P(y_i \mid \boldsymbol{x})}{f(\boldsymbol{x}, \boldsymbol{\theta})_i}$$

$$\mathrm{KL}(P(\boldsymbol{y} \mid \boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log \frac{P(y_i \mid \boldsymbol{x})}{f(\boldsymbol{x}, \boldsymbol{\theta})_i}$$

$$\mathrm{KL}(P(\boldsymbol{y} \mid \boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log \frac{P(y_i \mid \boldsymbol{x})}{f(\boldsymbol{x}, \boldsymbol{\theta})_i}$$

Rewrite the logarithm using the sum rule of logarithms

$$\mathrm{KL}(P(\boldsymbol{y} \mid \boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \left(\log P(y_i \mid \boldsymbol{x}) - \log f(\boldsymbol{x}, \boldsymbol{\theta})_i \right)$$

$$\mathrm{KL}(P(\boldsymbol{y} \mid \boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \Big(\log P(y_i \mid \boldsymbol{x}) - \log f(\boldsymbol{x}, \boldsymbol{\theta})_i \Big)$$

$$\mathrm{KL}(P(\boldsymbol{y} \mid \boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \left(\log P(y_i \mid \boldsymbol{x}) - \log f(\boldsymbol{x}, \boldsymbol{\theta})_i \right)$$

Split the sum into two parts

$$= \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log P(y_i \mid \boldsymbol{x}) - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

$$= \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log P(y_i \mid \boldsymbol{x}) - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

$$= \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log P(y_i \mid \boldsymbol{x}) - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

The first term is constant, and we will minimize the loss. So $\arg\min_{\boldsymbol{\theta}} \mathcal{L} + k = \arg\min_{\boldsymbol{\theta}} \mathcal{L}$. Therefore, we can ignore the first term.

$$= - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

$$= \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log P(y_i \mid \boldsymbol{x}) - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

The first term is constant, and we will minimize the loss. So $\arg\min_{\boldsymbol{\theta}} \mathcal{L} + k = \arg\min_{\boldsymbol{\theta}} \mathcal{L}$. Therefore, we can ignore the first term.

$$= - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

This is the loss for a classification task! We call this the **cross-entropy** loss function

Example:

Example:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Example:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= -\left[P(y_1 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_1 + P(y_2 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_2\right]$$

Example:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= -\left[P(y_1 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_1 + P(y_2 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_2\right]$$

$$= -[1 \cdot \log 0.6 + 0 \cdot \log 0.4]$$

Example:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= -\left[P(y_1 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_1 + P(y_2 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_2\right]$$

$$= -[1 \cdot \log 0.6 + 0 \cdot \log 0.4]$$

$$= -[-0.22]$$

Example:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i; \quad f(\boldsymbol{x}, \boldsymbol{\theta}) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}; \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= -\left[P(y_1 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_1 + P(y_2 \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_2\right]$$

$$= -[1 \cdot \log 0.6 + 0 \cdot \log 0.4]$$

$$= -[-0.22]$$

$$= 0.22$$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

By minimizing the loss, we make $f(\boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{x})$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = -\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

By minimizing the loss, we make $f(\boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{x})$

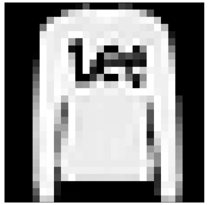
$$\arg \min_{\theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \arg \min_{\theta} \left[-\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i \right]$$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = - \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i$$

By minimizing the loss, we make $f(\boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{x})$

$$\arg \min_{\theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \arg \min_{\theta} \left[- \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i \right]$$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y} \mid \boldsymbol{x}) = P\left(\begin{bmatrix} \text{boot} \\ \text{dress} \\ \vdots \end{bmatrix} \middle| \right)$$



Our loss was just for a single image

Our loss was just for a single image

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \left[- \sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i \right]$$

Our loss was just for a single image

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \left[-\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i\right]$$

Find $\boldsymbol{\theta}$ that minimize the loss over the whole dataset

Our loss was just for a single image

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \left[-\sum_{i=1}^{d_y} P(y_i \mid \boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{\theta})_i\right]$$

Find $\boldsymbol{\theta}$ that minimize the loss over the whole dataset

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \left[-\sum_{j=1}^{n} \sum_{i=1}^{d_y} P\left(y_{[j],i} \mid \boldsymbol{x}_{[j]}\right) \log f\left(\boldsymbol{x}_{[j]}, \boldsymbol{\theta}\right)_i\right]$$

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. **Define loss function $\mathcal{L}$**
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. **Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$**
8. Coding

Find θ that minimize $\mathcal{L}$

Find $\boldsymbol{\theta}$ just like before, using gradient descent

Find θ that minimize $\mathcal{L}$

Find $\boldsymbol{\theta}$ just like before, using gradient descent

The gradients are the same as before except the last layer

Find θ that minimize $\mathcal{L}$

Find $\boldsymbol{\theta}$ just like before, using gradient descent

The gradients are the same as before except the last layer

I will not derive any more gradients, but the softmax gradient nearly identical to the sigmoid function gradient

Find θ that minimize $\mathcal{L}$

Find $\boldsymbol{\theta}$ just like before, using gradient descent

The gradients are the same as before except the last layer

I will not derive any more gradients, but the softmax gradient nearly identical to the sigmoid function gradient

$$\nabla_{\boldsymbol{\theta}} \operatorname{softmax}(\boldsymbol{z}) = \operatorname{softmax}(\boldsymbol{z}) \odot (1 - \operatorname{softmax}(\boldsymbol{z}))$$

Find θ that minimize $\mathcal{L}$

Find $\boldsymbol{\theta}$ just like before, using gradient descent

The gradients are the same as before except the last layer

I will not derive any more gradients, but the softmax gradient nearly identical to the sigmoid function gradient

$$\nabla_{\boldsymbol{\theta}} \operatorname{softmax}(\boldsymbol{z}) = \operatorname{softmax}(\boldsymbol{z}) \odot (1 - \operatorname{softmax}(\boldsymbol{z}))$$

This is because softmax is a multi-class generalization of the sigmoid function

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. **Define loss function $\mathcal{L}$**
7. Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$
8. Coding

Relax

Agenda

1. Review
2. Torch optimization coding
3. Classification task
4. Probability review
5. Define model f
6. Define loss function $\mathcal{L}$
7. **Find $\boldsymbol{\theta}$ that minimize $\mathcal{L}$**
8. Coding

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression
2. Classification

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression
2. Classification

Every interesting task (chatbot, self driving car, etc):

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression
2. Classification

Every interesting task (chatbot, self driving car, etc):

1. Construct a deep neural network

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression
2. Classification

Every interesting task (chatbot, self driving car, etc):

1. Construct a deep neural network
2. Compute regression or classification loss

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression
2. Classification

Every interesting task (chatbot, self driving car, etc):

1. Construct a deep neural network
2. Compute regression or classification loss
3. Optimize with gradient descent

Find θ that minimize $\mathcal{L}$

You have everything you need to solve deep learning tasks!

1. Regression
2. Classification

Every interesting task (chatbot, self driving car, etc):

1. Construct a deep neural network
2. Compute regression or classification loss
3. Optimize with gradient descent

The rest of this course will examine neural network architectures

Find θ that minimize $\mathcal{L}$

https://colab.research.google.com/drive/1BGMIE2CjlLJOH-D2r9AariPDVgxjWlqG?usp=sharing

Find θ that minimize $\mathcal{L}$

Homework 3 is released, you have two weeks to complete it

Find θ that minimize $\mathcal{L}$

Homework 3 is released, you have two weeks to complete it

https://colab.research.google.com/drive/1LainS20p6c3YVRFM4XgHRBODh6dvAaT2?usp=sharing#scrollTo=q8pJST5xFt-p