

STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 03: Stats/Linear algebra review

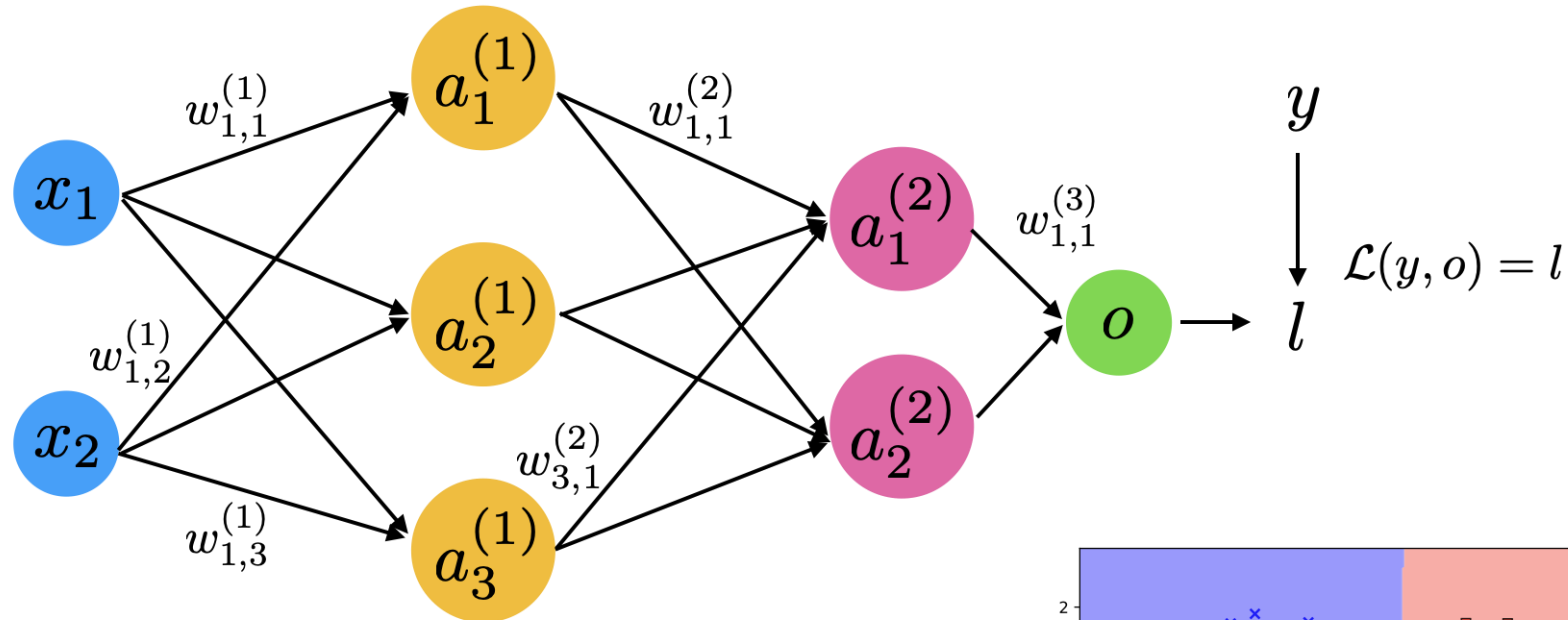
September 10, 2026



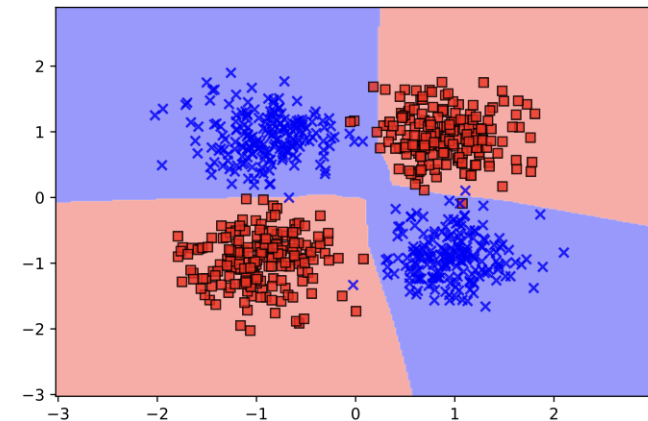


Questions about Course Logistics?

Today: Fundamental Math Skills for DL



So that we can solve the XOR problem, among other things ...



1-hidden layer MLP
with non-linear activation function (ReLU)



Today: Fundamental Math Skills for DL

1. **Tensors in Deep Learning**
2. Tensors and PyTorch
3. Vectors, Matrices, and Broadcasting
4. Probability Basics
5. Estimation Methods
6. Linear Regression

Scalars, Vectors, and Matrices

Scalar

(order-0 tensor)

$$x \in \mathbb{R}$$

e.g.,

$$x = 1.23$$

Vector

(order-1 tensor)

$$\mathbf{x} \in \mathbb{R}^n$$

but in this lecture,
we will assume

$$\mathbf{x} \in \mathbb{R}^{n \times 1}$$

e.g.,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Matrix

(order-2 tensor)

$$\mathbf{X} \in \mathbb{R}^{m \times n}$$

e.g.,

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{bmatrix}$$

$$\mathbf{x}^\top = [x_1 \quad x_2 \quad \dots \quad x_n], \text{ where } \mathbf{x}^\top \in \mathbb{R}^{1 \times n}$$

Scalars, Vectors, and Matrices

We will often use $\mathbf{X}$ as a special convention to refer to the "design matrix", "feature matrix", or "input matrix". That is, the matrix containing the training examples and features (inputs)

and assume the structure $\mathbf{X} \in \mathbb{R}^{n \times m}$

because n is often used to refer to the number of examples in literature across many disciplines.

$$\mathbf{X} = \begin{bmatrix} x_1^{[1]} & x_2^{[1]} & \dots & x_m^{[1]} \\ x_1^{[2]} & x_2^{[2]} & \dots & x_m^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{[n]} & x_2^{[n]} & \dots & x_m^{[n]} \end{bmatrix}$$

E.g.,

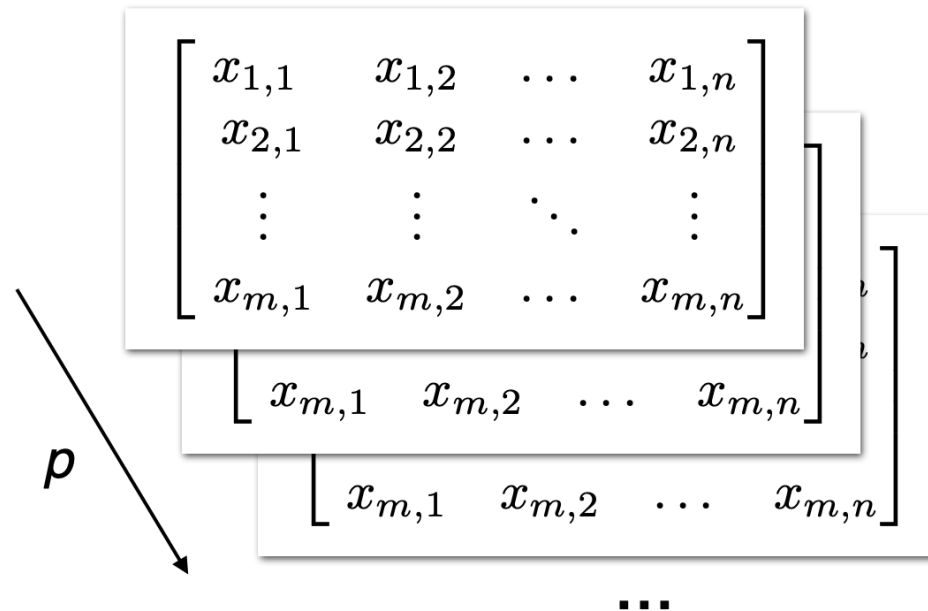
$x_2^{[1]}$ = 2nd feature value of the 1st training example

Tensors

3D Tensor

(order-3 tensor)

$$\mathbf{X} \in \mathbb{R}^{m \times n \times p} \quad (n \text{ and } m \text{ are generic indices here})$$



An Example of a 3D Tensor in DL

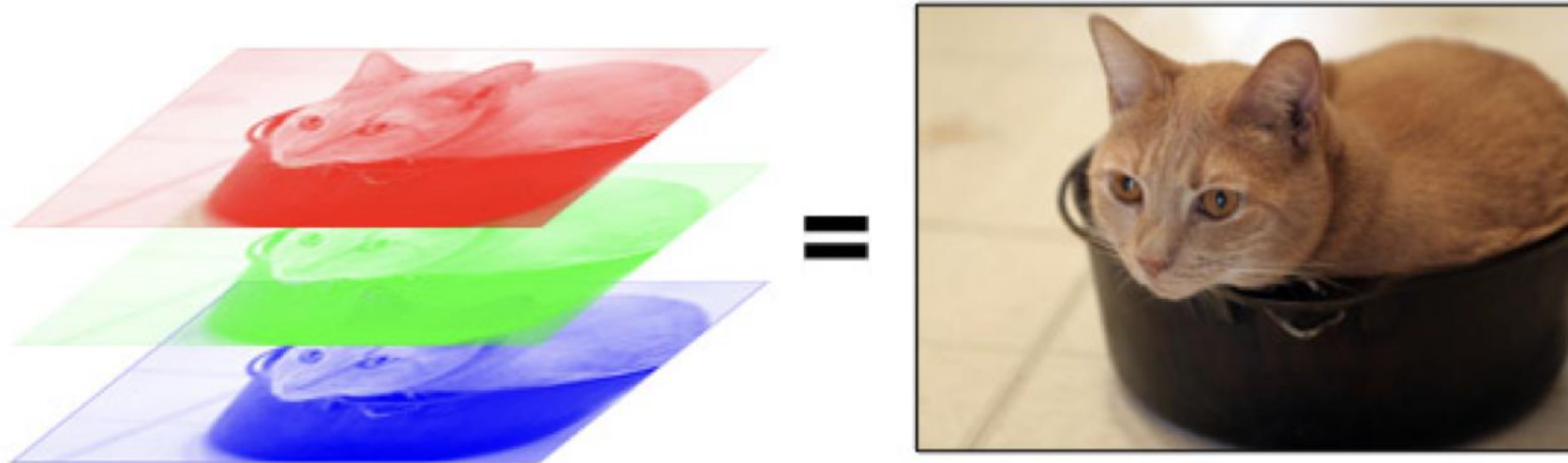
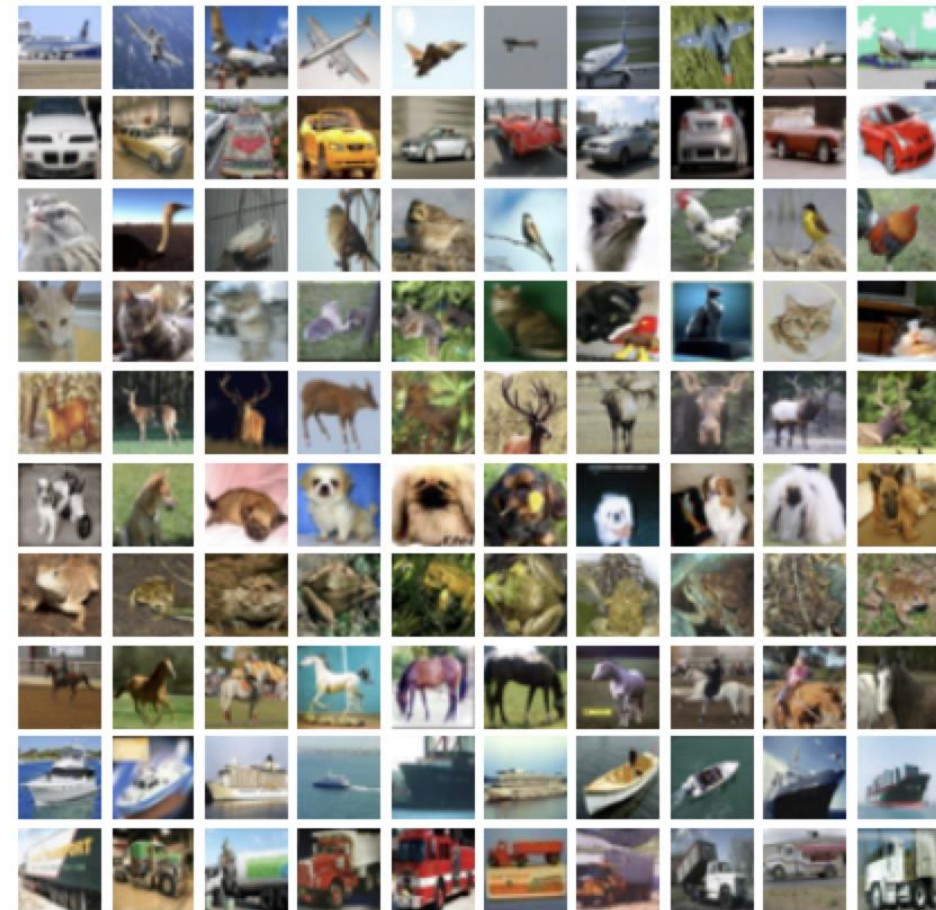


Image Source: <https://code.tutsplus.com/tutorials/create-a-retro-crt-distortion-effect-using-rgb-shifting--active-3359>

An Example of a 4D Tensor in DL

Batch of images
(as neural network input,
more later)

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



<https://www.cs.toronto.edu/~kriz/cifar.html>



For our purposes, tensor = multidimensional array

Dimensionality (“order”) = # of indices of .shape

```
[In [1]: import torch
```

```
[In [2]: t = torch.tensor([[1, 2, 3], [4, 5, 6]])
```

```
[In [3]: t
```

```
Out[3]:  
tensor([[1, 2, 3],  
        [4, 5, 6]])
```

```
[In [4]: t.shape
```

```
Out[4]: torch.Size([2, 3])
```

```
[In [5]: t.ndim
```

```
Out[5]: 2
```

```
In [6]: █
```



Today: Fundamental Math Skills for DL

1. Tensors in Deep Learning
- 2. Tensors and PyTorch**
3. Vectors, Matrices, and Broadcasting
4. Probability Basics
5. Estimation Methods
6. Linear Regression

Numpy Arrays → PyTorch Tensors

`numpy.array` / `numpy.ndarray` =
(data structure representation of a tensor)

`pytorch.tensor` / `pytorch.Tensor` =
(data structure representation of a tensor)

Example:

```
[In [1]: import numpy as np
```

```
[In [2]: a = np.array([1., 2., 3.])
```

```
[In [3]: print(a.dtype)  
float64
```

```
[In [4]: print(a.shape)  
(3,)
```

```
[In [5]: import torch
```

```
[In [6]: b = torch.tensor([1., 2., 3.])
```

```
[In [7]: print(b.dtype)  
torch.float32
```

```
[In [8]: print(b.shape)  
torch.Size([3])
```



Numpy and PyTorch Syntax is Similar

Numpy

```
[In [9]: a = np.array([1., 2., 3.])
```

```
[In [10]: print(a.dot(a))  
14.0
```

PyTorch

```
[In [12]: print(b.matmul(b))  
tensor(14.)
```

```
[In [13]: b  
Out[13]: tensor([1., 2., 3.])
```

```
[In [14]: b.numpy()  
Out[14]: array([1., 2., 3.], dtype=float32)
```

PyTorch: `matmul` = `dot` = `@`

```
[In [12]: print(b.matmul(b))  
tensor(14.)
```

```
[In [15]: print(b.dot(b))  
tensor(14.)
```

```
[In [16]: print(b @ b)  
tensor(14.)
```



Data types

NumPy data type	Tensor data type
numpy.uint8	torch.ByteTensor
numpy.int16	torch.ShortTensor
numpy.int32	torch.IntTensor
numpy.int	torch.LongTensor
numpy.int64	torch.LongTensor
numpy.float16	torch.HalfTensor
numpy.float32	torch.FloatTensor
numpy.float	torch.DoubleTensor
numpy.float64	torch.DoubleTensor

Default int in Numpy & PyTorch

Default float in PyTorch

Default float in NumPy

Specify the type with dtype

```
[In [21]: c = torch.tensor([1., 2., 3.], dtype=torch.float)
```

```
[In [22]: c.dtype
```

```
Out[22]: torch.float32
```

```
[In [23]: c = torch.tensor([1., 2., 3.], dtype=torch.double)
```

```
[In [24]: c.dtype
```

```
Out[24]: torch.float64
```

```
[In [25]: c = torch.tensor([1., 2., 3.], dtype=torch.float64)
```

```
[In [26]: c.dtype
```

```
Out[26]: torch.float64
```



Why not just use NumPy?

- PyTorch is made for DL:
 - GPU support
 - Automatic differentiation
 - DL Convenience functions

Loading Data onto a GPU

```
In [23]: print(torch.cuda.is_available())  
True
```

```
In [24]: b = b.to(torch.device('cuda:0'))  
...: print(b)
```

```
tensor([1., 2., 3.], device='cuda:0')
```

```
In [25]: b = b.to(torch.device('cpu'))  
...: print(b)  
tensor([1., 2., 3.])
```

How to Check Your CUDA Devices

- If you have CUDA installed, you should have access to nvidia-smi
- However, if you are using a laptop, you probably don't have CUDA compatible graphics cards (my laptops don't)
- We will discuss GPU cloud computing later ...

```
[sraschka@gpu03:~$ nvidia-smi
```

```
Mon Feb  8 21:05:27 2021
```

NVIDIA-SMI 455.32.00				Driver Version: 455.32.00		CUDA Version: 11.1	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG M.
0	GeForce RTX 208...	Off	00000000:1A:00.0	Off		N/A	
24%	37C	P0	71W / 250W	0MiB / 11019MiB	0%	Default	N/A

Installing PyTorch

If you want to install PyTorch later (after the lecture) ...

- If you use it on a laptop, you likely don't have a CUDA compatible GPU
- Recommend using CPU version for your laptop (no CUDA)
- Installation on GPU-cloud later ...
- Also, use this selector tool from <https://pytorch.org> (conda is recommended):

PyTorch Build	Stable (1.7.1)		Preview (Nightly)		
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python		C++ / Java		
CUDA	9.2	10.1	10.2	11.0	None
Run this Command:	NOTE: Python 3.9 users will need to add '-c=conda-forge' for installation <code>conda install pytorch torchvision torchaudio -c pytorch</code>				



Today: Fundamental Math Skills for DL

1. Tensors in Deep Learning
2. Tensors and PyTorch
3. **Vectors, Matrices, and Broadcasting**
4. Probability Basics
5. Estimation Methods
6. Linear Regression

Vectors

What do we call this in the context of a neural network?

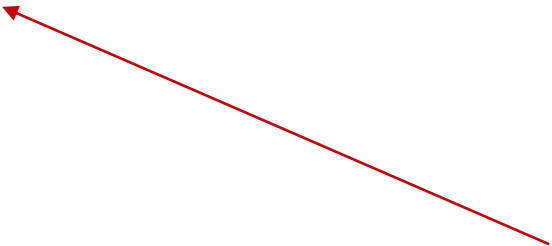
$$\mathbf{w}^\top \mathbf{x} + b = z \quad \text{where} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Matrices: Computing Outputs for Multiple Examples

$$\mathbf{X}\mathbf{w} + b = \mathbf{z} \quad \text{where} \quad \mathbf{X} = \begin{bmatrix} x_1^{[1]} & x_2^{[1]} & \dots & x_m^{[1]} \\ x_1^{[2]} & x_2^{[2]} & \dots & x_m^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{[n]} & x_2^{[n]} & \dots & x_m^{[n]} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

What is the Big-O computational cost of matrix multiplication (assume two NxN matrices)?

A common notational convenience

$$\mathbf{X}\mathbf{w} + b = \mathbf{z} \quad \text{where} \quad \mathbf{X} = \begin{bmatrix} x_1^{[1]} & x_2^{[1]} & \dots & x_m^{[1]} \\ x_1^{[2]} & x_2^{[2]} & \dots & x_m^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{[n]} & x_2^{[n]} & \dots & x_m^{[n]} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$


This should really be $\mathbf{X}\mathbf{w} + \mathbf{1}_n b = \mathbf{z}$, but in DL notation we drop the $\mathbf{1}_n$.

*We assume **broadcasting**.*

Broadcasting

In [4]: torch.tensor([1, 2, 3]) + 1

Out[4]: tensor([2, 3, 4])

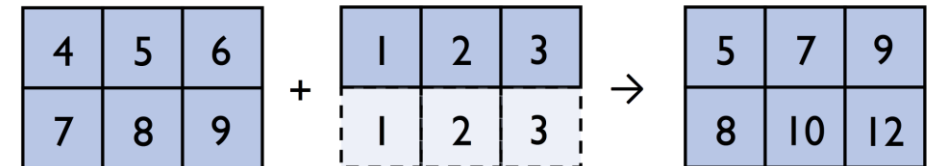


In [5]: t = torch.tensor([[4, 5, 6], [7, 8, 9]])

In [6]: t

Out[6]:

tensor([[4, 5, 6],
[7, 8, 9]])



In [7]: t + torch.tensor([1, 2, 3])

Out[7]:

tensor([[5, 7, 9],
[8, 10, 12]])

Be cautious of this when debugging ...



Today: Fundamental Math Skills for DL

1. Tensors in Deep Learning
2. Tensors and PyTorch
3. Vectors, Matrices, and Broadcasting
- 4. Probability Basics**
5. Estimation Methods
6. Linear Regression



Probability Basics: Definitions

- Random Variables:
 - Discrete: Values from a countable set (e.g. a coin flip)
 - Continuous: Values from an interval (e.g. a height)
- PMF and PDF:
 - **Probability Mass Function**: $P(X=x)$ for discrete X .
 - **Probability Density Function**: $f(x)$ for continuous X .

Key Distributions

- Bernoulli Distribution:
 - $P(X = x) = \theta^x(1 - \theta)^{1-x}, x \in \{0,1\}$
 - Example: a fair coin flip ($\theta = 0.5$)
- Gaussian Distribution:
 - $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
 - “**Normal**” because of Central Limit Theorem
 - “Standard Normal” when $\mu = 0, \sigma = 1$

Central Limit Theorem (CLT)

- Let $X_1, X_2, \dots, X_n$ be i.i.d. random variables with mean μ and variance σ^2 .
- Define the sample mean:

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$$

- Then, as $n \rightarrow \infty$:

$$\frac{\bar{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \rightarrow N(0, 1)$$

Joint, Marginal, and Conditional Probabilities

- Joint: $P(A, B)$, probability of two events occurring together.
- Marginal: $P(A) = \sum_B P(A, B)$, sum of joint probabilities over one variable.
- Conditional: $P(A|B) = \frac{P(A, B)}{P(B)}$, probability of A given B.

Expectation and Variance

- Expectation:
 - Discrete: $E[X] = \sum_x xP(X = x)$
 - Continuous: $E[X] = \int xf(x)dx$
- Variance: $Var(X) = E[(X - E[X])^2]$
 - Equivalent: $Var(X) = E[X^2] - E[X]^2$

Linearity of Expectation

- Property:
 - $E[aX + b] = aE[X] + b$
- Multiple Variables:
 - $E[X_1 + X_2] = E[X_1] + E[X_2]$

Expectation of Functions

- Formula:
 - $E[g(X)] = \sum_x g(x)P(X = x)$ (discrete)
 - $E[g(X)] = \int_x g(x)f(x)dx$ (continuous)
- Example (Discrete):
 - $X \sim \text{Bernoulli}(\theta), g(X) = X^2$:
 - $E[g(X)] = 1^2\theta + 0^2(1 - \theta) = \theta$
- Example (Continuous):
 - $X \sim \text{Uniform}(0,1), g(X) = X^2$:
 - $E[g(X)] = \int_0^1 x^2 dx = \frac{1}{3}$.

Variance of Functions

- Definition:
 - $Var(g(X)) = E[(g(X) - E[g(X)])^2]$
 - Equivalent: $Var(g(X)) = E[g(X)^2] - (E[g(X)])^2$

Covariance and Correlation

- Covariance:
 - $Cov(X, Y) = E[(X - E[X])(Y - E[Y])]$
- Properties:
 - $Cov(X, X) = Var(X)$
 - If X, Y are independent: $Cov(X, Y) = 0$.
- Correlation:
 - $\rho(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$
 - $\rho = 1$: Perfect positive linear relationship.
 - $\rho = 0$: No linear relationship.
 - $\rho = -1$: Perfect negative linear relationship.

Bayes' Rule

- $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
- Example: Medical test:
 - $P(\text{disease}|\text{positive test}) = \frac{P(\text{positive test}|\text{disease}) P(\text{disease})}{P(\text{positive test})}$



Today: Fundamental Math Skills for DL

1. Tensors in Deep Learning
2. Tensors and PyTorch
3. Vectors, Matrices, and Broadcasting
4. Probability Basics
5. **Estimation Methods**
6. Linear Regression



Introduction to Estimation

- **Goal of Estimation:**
 - Infer unknown parameters θ from observed data.
- **Types of Estimation:**
 - Point Estimation: Single value (e.g., MLE).
 - Interval Estimation: Range of plausible values (e.g., confidence intervals).
- **Common Methods:**
 - Maximum Likelihood Estimation (MLE)
 - Maximum A Posteriori (MAP)
 - Method of Moments

Maximum Likelihood Estimation (MLE)

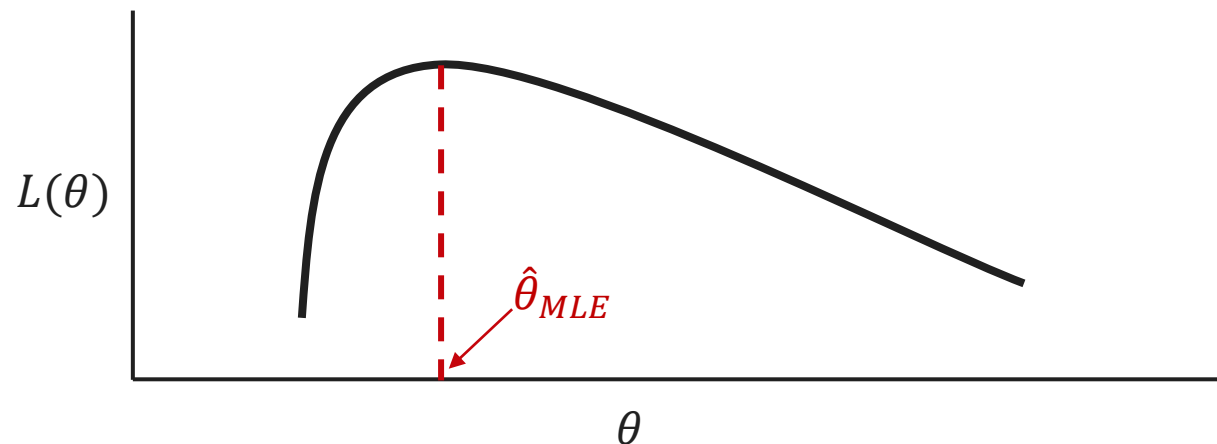
- **Definition:**

- Find $\hat{\theta}$ that maximizes the likelihood of observing the given data.

$$\hat{\theta} = \operatorname{argmax}_{\theta} L(\theta) \text{ where } L(\theta) = P(\text{data}|\theta).$$

- **Interpretation:**

- $L(\theta)$: Probability of the observed data given θ .
- MLE chooses the parameter that makes the data most "likely."



Maximum Likelihood Estimation (MLE)

- **Example:**

- Dataset: $X=\{1,0,1,1,0\}$,
- Bernoulli distribution with $P(X = 1|\theta) = \theta$:

$$L(\theta) = \prod_i \theta^{x_i} (1 - \theta)^{1-x_i}$$

- Typically solved by maximizing the log-likelihood.
 $\ell(\theta) = \log L(\theta) = \sum_{i=1}^n (x_i \log \theta + (1 - x_i) \log(1 - \theta))$
- Derivative:

$$\frac{d\ell}{d\theta} = \frac{k}{\theta} - \frac{n - k}{1 - \theta}$$

where $k = \sum x_i$

- Solution:

$$\hat{\theta} = \frac{k}{n}$$



Maximum Likelihood Estimation (MLE)

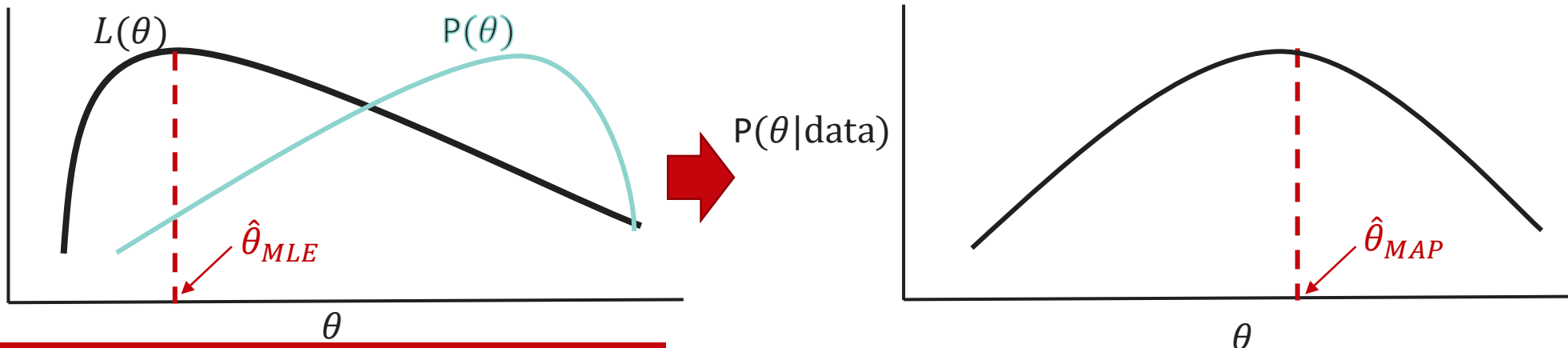
- The MLE:
 - does not always exist.
 - is not necessarily unique.
 - is not necessarily admissible.

Maximum A Posteriori (MAP) Estimation

- Find

$$\hat{\theta}_{MAP} = \operatorname{argmax}_{\theta} P(\theta|\text{data}) \propto \operatorname{argmax}_{\theta} P(\text{data}|\theta)P(\theta)$$

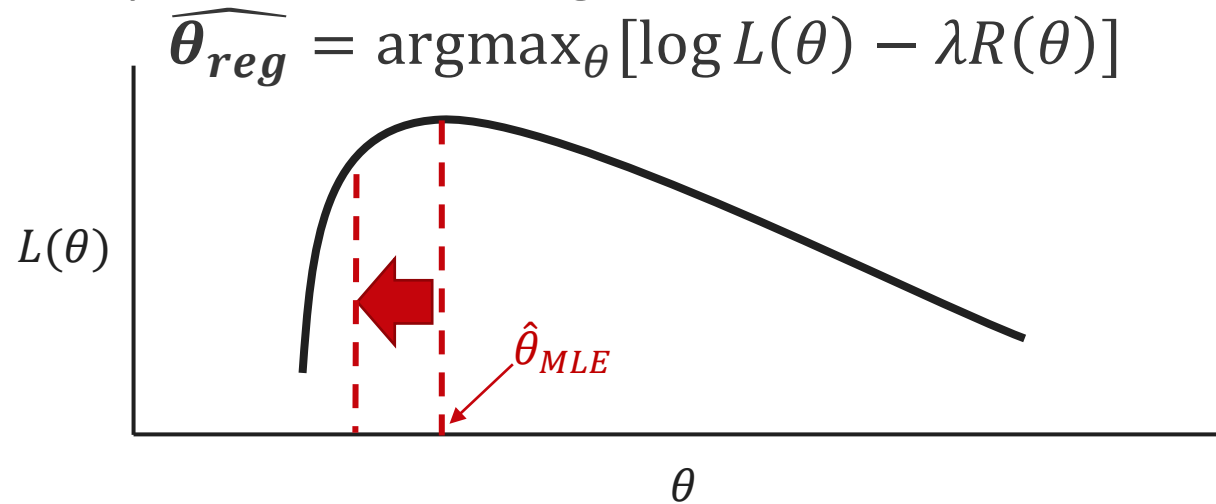
- $P(\text{data}|\theta)$: Likelihood
- $P(\theta)$: Prior belief about θ
- MLE ignores $P(\theta)$
- MAP incorporates prior information.



Regularization is MAP

- **MLE with Regularization:**

- Adds a penalty to avoid overfitting



- **MAP as Penalized MLE:**

- Let $P(\theta) \propto e^{-\lambda R(\theta)}$. Then

$$\hat{\theta}_{MAP} = \operatorname{argmax}_{\theta} [\log L(\theta) + \log P(\theta)] = \hat{\theta}_{reg}$$



Today: Fundamental Math Skills for DL

1. Tensors in Deep Learning
2. Tensors and PyTorch
3. Vectors, Matrices, and Broadcasting
4. Probability Basics
5. Estimation Methods
6. **Linear Regression**

Introduction to Linear Regression

- **Model Definition:**

- $y = X\beta + \epsilon$, where
- y : Response variable (dependent variable).
- X : Design matrix (independent variables or features).
- β : Coefficients (parameters to estimate).
- ϵ : Error term (often assumed to be $N(0, \sigma^2)$).

- **Goal:**

- Estimate β .

Linear Regression Evaluation Metrics

- **Coefficient of Determination (R^2):**

- $R^2 = 1 - \frac{SS_{residual}}{SS_{total}}$

- Measures the proportion of variance explained by the model.

- **Mean Squared Error (MSE):**

- $MSE = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$

- **Mean Absolute Error (MAE):**

- $MSE = \frac{1}{n} \sum \|y_i - \hat{y}_i\|$

Ordinary Least Squares (OLS)

- **Objective:**
 - Minimize the sum of squared residuals:
 - $\hat{\beta}_{OLS} = \operatorname{argmin}_{\beta} \|y - X\beta\|^2$
- Residuals:
 - $e_i = y_i - \hat{y}_i$
- **Solution:**
 - $\hat{\beta}_{OLS} = (X^T X)^{-1} X^T Y$

Regularization in Linear Regression (MAP)

- **Ridge Regression (L2 Regularization):**

- Adds an L2 penalty:
 - $\hat{\beta}_{ridge} = \operatorname{argmin}_{\beta} \|y - X\beta\|^2 + \lambda \|\beta\|^2$
- Equivalent MAP interpretation:
 - Prior on coefficients: $\beta \sim N(0, \frac{\sigma^2}{\lambda})$
 - MAP estimate maximizes: $P(\beta|y) \propto P(y|\beta)P(\beta)$
 - Penalty comes from the Gaussian prior.

- **Lasso Regression (L1 Regularization):**

- Adds an L1 penalty:
 - $\hat{\beta}_{lasso} = \operatorname{argmin}_{\beta} \|y - X\beta\|^2 + \lambda \|\beta\|_1$
- Equivalent to $\beta \sim \text{Laplace}(0, \frac{\sigma}{\lambda})$



Next Lecture

Single-layer networks:

- Perceptrons
 - Learning algorithm, geometric intuition
- A fully-connected layer in PyTorch.

Questions?

