Introduction to Machine Learning

- Prof. Balaraman Ravindran | IIT Madras

Problem Solving Session (Week-5)

Shreya Bansal

PMRF PhD Scholar
IIT Ropar

Week-5 Contents

---

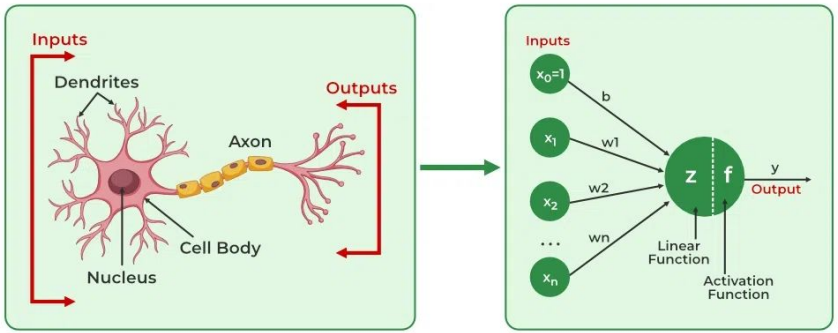
1. Artificial Neural Network
2. Backpropagation
3. Parameter Estimation

Artificial Neural Networks (ANNs)

- Discussed Perceptrons as a fundamental concept.
- ANNs were inspired by the brain's architecture but evolved into two research directions:
- **Neuroscience**-driven models (biological relevance).
- **Mathematical and computational models** (machine learning focus).
- We will focus on computational neural networks rather than biological relevance.

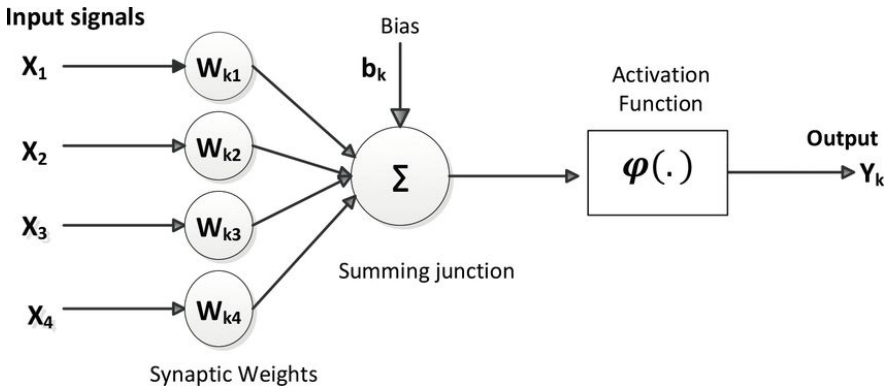
Basic Concept of a Neuron

- A neuron receives inputs from multiple sources.
- It performs some computation and produces an output.
- These neurons are connected in complex networks allowing powerful computations.



McCulloch-Pitts Model

- Earliest mathematical model of a neuron.
- Uses a Σ unit (summation function).
- Inputs can be excitatory or inhibitory.
- If the sum exceeds a threshold (θ), the neuron fires (output = 1).
- If an inhibitory input is active, the output is 0.



Perceptron Model

- Improved version of McCulloch-Pitts model.
- Instead of simple summation, it uses weighted inputs:

$$Y = \sum_{i=1}^{p} X_i \beta_i$$

- If sum exceeds threshold, output = +1, otherwise output = -1.
- Alternative representation by adding bias term (β_0).
- Key difference from McCulloch-Pitts: No inhibitory inputs & weighted summation.
- Training Perceptron:
- Uses gradient descent to adjust weights.
- We update weights using misclassified points.

Challenges with Perceptron

- **Only works for linearly separable data.**
- **Example: XOR problem cannot be solved using a single-layer perceptron.**
- **Solution:**
- **Transform data into higher-dimensional space.**
- **Introduce non-linearity (e.g., multi-layer networks, activation functions).**

Gradient Descent in Neural Networks

- Used for optimizing neural networks by adjusting weights.
- Key Idea: Move in the direction of steepest descent (negative gradient).
- Step size (η) is crucial:
- Large step → Oscillations, divergence.
- Small step → Slow convergence.
- Stochastic Gradient Descent (SGD):
- Instead of computing the gradient for the entire dataset, updates are done per single data point.
- Helps in faster convergence but introduces noise.
- Trade-off: Small steps ensure steady convergence while large steps risk overshooting.

Beyond Perceptrons: Adaline (Adaptive Linear Neuron)

- **Similar to perceptron but outputs a continuous value instead of a binary output.**
- **Uses mean squared error (MSE) instead of classification error.**
- **Key advantage:**
- **Allows gradient descent to work properly (since Perceptron's step function is non-differentiable).**

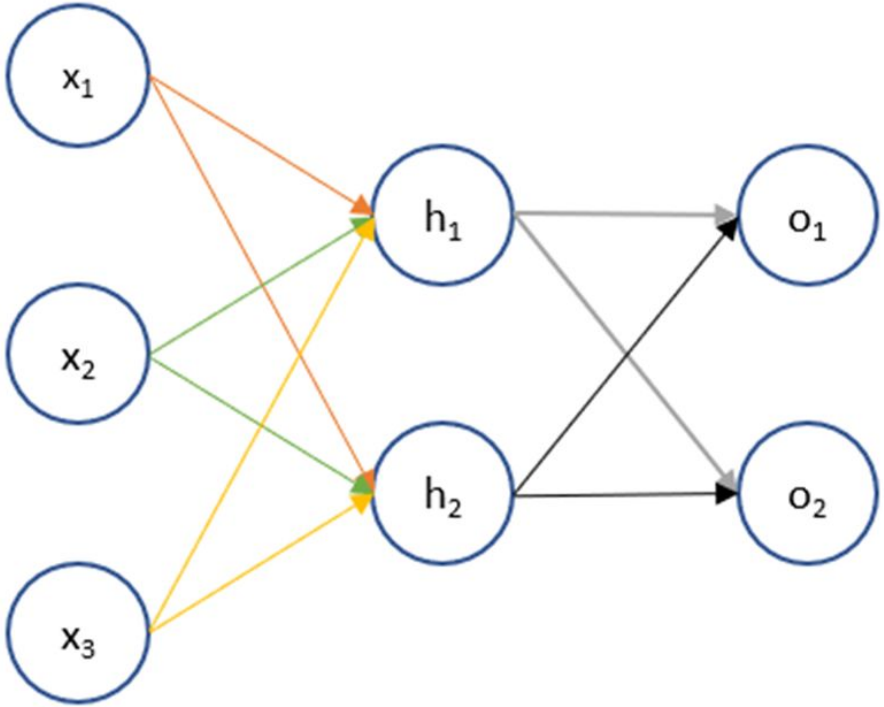
Limitations of Single-Layer Networks

- Cannot handle non-linearly separable problems.
- Example: XOR problem requires hidden layers.
- Multi-layer Perceptrons (MLPs) introduce:
- Hidden layers with multiple neurons.
- Activation functions (sigmoid, ReLU, etc.).

Backpropagation

- Steps to follow-
- (1) Initialize weights for the parameters we want to train
- (2) Forward propagate through the network to get the output values
- (3) Define the error or cost function and its first derivatives
- (4) Backpropagate through the network to determine the error derivatives
- (5) Update the parameter estimates using the error derivative and the current value

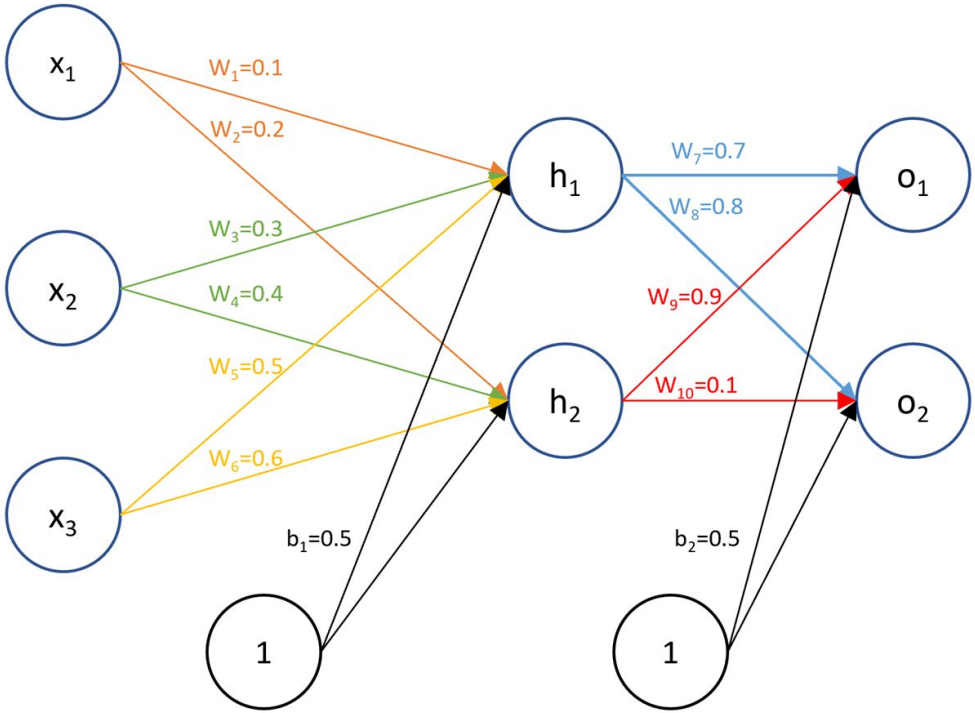
Sample Question: Backpropagation



Step 1: Initialise weights

- The input and target values for this problem are
- X1=1
- X2=4
- X3=5

- T1=0.1
- T2=0.05



Step 2 : Forward propagate

Input to hidden layer

$$w_1 x_1 + w_3 x_2 + w_5 x_3 + b_1 = z_{h_1}$$

$$w_2 x_1 + w_4 x_2 + w_6 x_3 + b_1 = z_{h_2}$$

$$h_1 = \sigma(z_{h_1})$$

$$h_2 = \sigma(z_{h_2})$$

Hidden layer to output layer

$$w_7 h_1 + w_9 h_2 + b_2 = z_{o_1}$$

$$w_8 h_1 + w_{10} h_2 + b_2 = z_{o_2}$$

$$o_1 = \sigma(z_{o_1})$$

$$o_2 = \sigma(z_{o_2})$$

Step 2 : Forward propagate

$$w_1x_1 + w_3x_2 + w_5x_3 + b_1 = z_{h_1} = 0.1(1) + 0.3(4) + 0.5(5) + 0.5 = 4.3$$
$$h_1 = \sigma(z_{h_1}) = \sigma(4.3) = 0.9866$$

$$w_2x_1 + w_4x_2 + w_6x_3 + b_1 = z_{h_2} = 0.2(1) + 0.4(4) + 0.6(5) + 0.5 = 5.3$$
$$h_2 = \sigma(z_{h_2}) = \sigma(5.3) = 0.9950$$

$$w_7h_1 + w_9h_2 + b_2 = z_{o_1} = 0.7(0.9866) + 0.9(0.9950) + 0.5 = 2.0862$$
$$o_1 = \sigma(z_{o_1}) = \sigma(2.0862) = 0.8896$$

$$w_8h_1 + w_{10}h_2 + b_2 = z_{o_2} = 0.8(0.9866) + 0.1(0.9950) + 0.5 = 1.3888$$
$$o_2 = \sigma(z_{o_2}) = \sigma(1.3888) = 0.8004$$

Step 3 : Define the error and its first derivatives

$$E = \frac{1}{2}[(o_1 - t_1)^2 + (o_2 - t_2)^2$$

$$\frac{dE}{d_{o_1}} = o_1 - t_1$$

$$\frac{dE}{d_{o_2}} = o_2 - t_2$$

Step 4: Backpropagate through the network

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

$$\frac{d\sigma}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$$

We can write $\frac{e^{-x}}{1+e^{-x}}$ as $1 - \frac{1}{1+e^{-x}}$. The derivative can be written as

$$\frac{d\sigma}{dx} = \frac{1}{1+e^{-x}}(1 - \sigma(x))$$

$$\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x))$$

Step 4: Backpropagate through the network

Also, given that $w_7h_1 + w_9h_2 + b_2 = z_{o_1}$ and $w_8h_1 + w_{10}h_2 + b_2 = z_{o_2}$, we have $\frac{dz_{o_1}}{dw_7} = h_1$, $\frac{dz_{o_2}}{dw_8} = h_1$, $\frac{dz_{o_1}}{dw_9} = h_2$, $\frac{dz_{o_2}}{dw_{10}} = h_2$, $\frac{dz_{o_1}}{db_2} = 1$, and $\frac{dz_{o_2}}{db_2} = 1$.

We are now ready to calculate $\frac{dE}{dw_7}$, $\frac{dE}{dw_8}$, $\frac{dE}{dw_9}$, and $\frac{dE}{dw_{10}}$ using the derivatives we have already discussed.

$$\frac{dE}{dw_7} = \frac{dE}{do_1}\frac{do_1}{dz_{o_1}}\frac{dz_{o_1}}{dw_7}$$

$$\frac{dE}{dw_7} = (o_1 - t_1)(o_1(1 - o_1))h_1$$

$$\frac{dE}{dw_7} = (0.8896 - 0.1)(0.8896(1 - 0.8896))(0.9866)$$

$$\frac{dE}{dw_7} = 0.0765$$

Step 4: Backpropagate through the network

$$\frac{dE}{dw_8} = \frac{dE}{do_2}\frac{do_2}{dz_{o_2}}\frac{dz_{o_2}}{dw_8}$$

$$\frac{dE}{dw_8} = (0.7504)(0.1598)(0.9866)$$

$$\frac{dE}{dw_8} = 0.1183$$

$$\frac{dE}{dw_9} = \frac{dE}{do_1}\frac{do_1}{dz_{o_1}}\frac{dz_{o_1}}{dw_9}$$

$$\frac{dE}{dw_9} = (0.7896)(0.0983)(0.9950)$$

$$\frac{dE}{dw_9} = 0.0772$$

$$\frac{dE}{dw_{10}} = \frac{dE}{do_2}\frac{do_2}{dz_{o_2}}\frac{dz_{o_2}}{dw_{10}}$$

$$\frac{dE}{dw_{10}} = (0.7504)(0.1598)(0.9950)$$

$$\frac{dE}{dw_{10}} = 0.1193$$

Step 4: Backpropagate through the network

The error derivative of b_2 is a little bit more involved since changes to b_2 affect the error through both o_1 and o_2.

$$\frac{dE}{db_2} = \frac{dE}{do_1}\frac{do_1}{dz_{o_1}}\frac{dz_{o_1}}{db_2} + \frac{dE}{do_2}\frac{do_2}{dz_{o_2}}\frac{dz_{o_2}}{db_2}$$

$$\frac{dE}{db_2} = (0.7896)(0.0983)(1) + (0.7504)(0.1598)(1)$$

$$\frac{dE}{db_2} = 0.1975$$

To summarize, we have computed numerical values for the error derivatives with respect to w_7, w_8, w_9, w_{10}, and b_2. We will now backpropagate one layer to compute the error derivatives of the parameters connecting the input layer to the hidden layer. These error derivatives are $\frac{dE}{dw_1}, \frac{dE}{dw_2}, \frac{dE}{dw_3}, \frac{dE}{dw_4}, \frac{dE}{dw_5}, \frac{dE}{dw_6}$, and $\frac{dE}{db_1}$.

Step 4: Backpropagate through the network

I will calculate $\frac{dE}{dw_1}$, $\frac{dE}{dw_3}$, and $\frac{dE}{dw_5}$ first since they all flow through the h_1 node.

$$\frac{dE}{dw_1} = \frac{dE}{dh_1}\frac{dh_1}{dz_{h_1}}\frac{dz_{h_1}}{dw_1}$$

The calculation of the first term on the right hand side of the equation above is a bit more involved than previous calculations since h_1 affects the error through both o_1 and o_2.

$$\frac{dE}{dh_1} = \frac{dE}{do_1}\frac{do_1}{dz_{o_1}}\frac{dz_{o_1}}{dh_1} + \frac{dE}{do_2}\frac{do_2}{dz_{o_2}}\frac{dz_{o_2}}{dh_1}$$

Step 4: Backpropagate through the network

$$\frac{dE}{dh_1} = (0.7896)(0.0983)(0.7) + (0.7504)(0.1598)(0.8) = 0.1502$$

Plugging the above into the formula for $\frac{dE}{dw_1}$, we get

$$\frac{dE}{dw_1} = (0.1502)(0.0132)(1) = 0.0020$$

The calculations for $\frac{dE}{dw_3}$ and $\frac{dE}{dw_5}$ are below

$$\frac{dE}{dw_3} = \frac{dE}{dh_1}\frac{dh_1}{dz_{h_1}}\frac{dz_{h_1}}{dw_3}$$

$$\frac{dE}{dw_3} = (0.1502)(0.0132)(4) = 0.0079$$

$$\frac{dE}{dw_5} = \frac{dE}{dh_1}\frac{dh_1}{dz_{h_1}}\frac{dz_{h_1}}{dw_5}$$

$$\frac{dE}{dw_5} = (0.1502)(0.0132)(5) = 0.0099$$

Step 4: Backpropagate through the network

I will now calculate $\frac{dE}{dw_2}$, $\frac{dE}{dw_4}$, and $\frac{dE}{dw_6}$ since they all flow through the h_2 node.

$$\frac{dE}{dw_2} = \frac{dE}{dh_2}\frac{dh_2}{dz_{h_2}}\frac{dz_{h_2}}{dw_2}$$

The calculation of the first term on the right hand side of the equation above i
affects the error through both o_1 and o_2.

$$\frac{dE}{dh_2} = \frac{dE}{do_1}\frac{do_1}{dz_{o_1}}\frac{dz_{o_1}}{dh_2} + \frac{dE}{do_2}\frac{do_2}{dz_{o_2}}\frac{dz_{o_2}}{dh_2}$$

$$\frac{dE}{dh_2} = (0.7896)(0.0983)(0.9) + (0.7504)(0.1598)(0.1) = 0.0818$$

Plugging the above into the formula for $\frac{dE}{dw_2}$, we get

$$\frac{dE}{dw_2} = (0.0818)(0.0049)(1) = 0.0004$$

The calculations for $\frac{dE}{dw_4}$ and $\frac{dE}{dw_6}$ are below

$$\frac{dE}{dw_4} = \frac{dE}{dh_2}\frac{dh_2}{dz_{h_2}}\frac{dz_{h_2}}{dw_4}$$

$$\frac{dE}{dw_4} = (0.0818)(0.0049)(4) = 0.0016$$

$$\frac{dE}{dw_6} = \frac{dE}{dh_2}\frac{dh_2}{dz_{h_2}}\frac{dz_{h_2}}{dw_6}$$

$$\frac{dE}{dw_6} = (0.0818)(0.0049)(5) = 0.0020$$

Step 4: Backpropagate through the network

The final error derivative we have to calculate is $\frac{dE}{db_1}$, which is done next

$$\frac{dE}{db_1} = \frac{dE}{do_1}\frac{do_1}{dz_{o_1}}\frac{dz_{o_1}}{dh_1}\frac{dh_1}{dz_{h_1}}\frac{dz_{h_1}}{db_1} + \frac{dE}{do_2}\frac{do_2}{dz_{o_2}}\frac{dz_{o_2}}{dh_2}\frac{dh_2}{dz_{h_2}}\frac{dz_{h_2}}{db_1}$$

$$\frac{dE}{db_1} = (0.7896)(0.0983)(0.7)(0.0132)(1)+(0.7504)(0.1598)(0.1)(0.0049)(1) = 0.0008$$

Step 4: Backpropagate through the network

$$w_1 := w_1 - \alpha \frac{dE}{dw_1} = 0.1 - (0.01)(0.0020) = 0.1000$$

$$w_2 := w_2 - \alpha \frac{dE}{dw_2} = 0.2 - (0.01)(0.0004) = 0.2000$$

$$w_3 := w_3 - \alpha \frac{dE}{dw_3} = 0.3 - (0.01)(0.0079) = 0.2999$$

$$w_4 := w_4 - \alpha \frac{dE}{dw_4} = 0.4 - (0.01)(0.0016) = 0.4000$$

$$w_5 := w_5 - \alpha \frac{dE}{dw_5} = 0.5 - (0.01)(0.0099) = 0.4999$$

$$w_6 := w_6 - \alpha \frac{dE}{dw_6} = 0.6 - (0.01)(0.0020) = 0.6000$$

$$w_7 := w_7 - \alpha \frac{dE}{dw_7} = 0.7 - (0.01)(0.0765) = 0.6992$$

$$w_8 := w_8 - \alpha \frac{dE}{dw_8} = 0.8 - (0.01)(0.1183) = 0.7988$$

$$w_9 := w_9 - \alpha \frac{dE}{dw_9} = 0.9 - (0.01)(0.0772) = 0.8992$$

$$w_{10} := w_{10} - \alpha \frac{dE}{dw_{10}} = 0.1 - (0.01)(0.1193) = 0.0988$$

$$b_1 := b_1 - \alpha \frac{dE}{db_1} = 0.5 - (0.01)(0.0008) = 0.5000$$

$$b_2 := b_2 - \alpha \frac{dE}{db_2} = 0.5 - (0.01)(0.1975) = 0.4980$$

Weight Initialization

- Initializing all weights to the same value (e.g., 1, 0, or any constant) is generally a bad idea because it prevents differentiation in learning across neurons.
- Random initialization is preferred to allow weights to specialize.
- However, weights should be small to keep neuron outputs in the linear region of activation functions like sigmoid or tanh, ensuring higher gradient sensitivity for efficient learning.

Overfitting & Regularization

- Overfitting occurs when the model memorizes training data instead of generalizing to new examples.
- Neural networks are prone to overfitting due to a large number of parameters.
- Two main ways to combat overfitting:
- Regularization: Adding a penalty (e.g., L2 norm or weight decay) to the weight values.
- Validation Set: Using a separate dataset to monitor performance and stop training when generalization starts degrading.

Number of Hidden Units & Layers

- Choosing the optimal number of hidden layers/neurons is difficult and typically done via validation experiments.
- Some automatic methods exist:
- Pruning: Start with a large network and remove less important weights (e.g., "Optimal Brain Damage").
- Growing Networks: Start with a minimal architecture and iteratively add neurons as needed.

Adaptive Network Growth

- Instead of pre-defining layers, some methods dynamically add neurons as needed.
- These approaches deviate from the standard feedforward structure and may lead to more irregular architectures.

Goals of Parameter Estimation

- Estimate parameter values that best explain the given data.
- Calculate the probability of new observations given past training data.
- Previously explored in logistic regression.

Likelihood in Parameter Estimation

- Likelihood measures how well parameters explain the observed data.
- Uses Bayes' Rule:
- $P(\theta|X)=P(X|\theta)P(\theta) \ / \ P(X)$
- Focus on maximizing $P(X \mid \theta)$ since $P(X)$ is constant.

Likelihood Function

- **Likelihood is expressed as:**
- $L(\theta)=P(X|\theta)$
- **Example: Suppose we have different values of θ, we compute P(X | θ) for each and find the maximum likelihood estimate (MLE).**

Maximum Likelihood Estimation (MLE)

- MLE finds θ that maximizes $P(X \mid \theta)$.
- If prior knowledge of θ is unavailable, we only focus on likelihood.
- Often use log-likelihood for ease of computation.

Independence Assumption in Likelihood

- If data samples are independent:
- $L(\theta)=\prod_i P(X_i|\theta)$
- We take the log-likelihood to convert the product into a sum:
- Log $L(\theta)=\sum_i \log P(X_i|\theta)$
- Commonly used in statistical modeling and machine learning.

Applying MLE to Logistic Regression

In logistic regression:

First, estimate β using MLE.

Then, use estimated parameters to predict probabilities.

Example – Coin Tossing Experiment

Define C as a random variable representing coin outcome:

C = 1 (heads), C = 0 (tails).

Coin has an unknown probability ρ of landing heads.

Likelihood function:

$$L(\rho)=\Pi_i \rho^{Ci}(1-\rho)^{1-Ci}$$

Bernoulli Distribution and Likelihood

The probability of a single coin flip:

$$P(C_i|\varrho)=\varrho^{Ci}(1-\varrho)^{1-Ci}$$

For N tosses:

N_1 = number of heads, N_0 = number of tails.

Log-likelihood:

$$\log L(\varrho)=N_1\log\varrho+N_0\log(1-\varrho)$$

Finding the MLE for Coin Tossing

- Differentiate log-likelihood and set to zero:
- $d/d\rho\ (N_1 \log \rho + N_0 \log(1-\rho)) = 0$

Solve for ρ:

- $\hat{\rho} = N_1 / N$
- Conclusion: MLE estimates probability as (heads count) / (total tosses).

Maximum Likelihood and Prior Knowledge

- **Maximum likelihood assumes no prior knowledge of parameters.**
- **What if we have some prior knowledge?**
- **Example: Coin toss experiment**
    - **Suppose we believe the coin is fair.**
    - **How do we incorporate this belief into estimation?**

Prior Knowledge in Probability

- **Prior belief: The coin is fair ($\rho = 0.5$).**
- **This belief can be represented as a probability distribution.**
- **Gaussian distribution with a peak at 0.5 could be used.**
- **However, Gaussian is not ideal since probabilities must be between 0 and 1.**

The Beta Distribution

Beta Distribution 📊

The **Beta distribution** is a continuous probability distribution defined on the interval $(0, 1)$ and is parameterized by two positive shape parameters, α and β.

Probability Density Function (PDF)

$$f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

where $B(\alpha, \beta)$ is the **Beta function**:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt$$

Properties 🏆

1. **Support:** $x \in (0, 1)$, making it useful for modeling probabilities.
2. **Mean:**

$$E[X] = \frac{\alpha}{\alpha + \beta}$$

3. **Variance:**

$$\text{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

4. **Shape Behavior:**
   - If $\alpha = \beta = 1$, it's **Uniform(0,1)**.
   - If $\alpha > \beta$, it's skewed **right**.
   - If $\alpha < \beta$, it's skewed **left**.
   - If $\alpha = \beta > 1$, it's symmetric and bell-shaped.

The Beta Distribution

- **Why Beta Distribution?**
  - **Defined between [0,1] → Suitable for probability priors.**
  - **Frequently used in Bayesian statistics for probability estimation.**
- **The Beta distribution is ideal for modeling prior beliefs.**

Maximum A Posteriori (MAP) Estimation

- Unlike Maximum Likelihood (ML), MAP incorporates prior beliefs.
- ML: Maximizes likelihood function only.
- MAP: Maximizes posterior probability:

$$\theta_{MAP} = \text{argmax}\, P(\theta|X) = \text{argmax}\, P(X|\theta)P(\theta)$$

- Only the numerator matters in maximization.

Using Priors in Optimization

- **Priors can be useful when we have prior knowledge or preferences:**
  - **Belief in fair coin → Use Beta distribution centered at 0.5.**
  - **Ridge/Lasso regression → Use priors to enforce small parameters.**
  - **Sparsity constraints → Low probability for many nonzero parameters.**

Role of Priors in Regularization

- **Priors can act as regularizers:**
    - **Prevent overfitting.**
    - **Encourage simpler models.**
- **Example:**
    - **L2 prior (Gaussian): Encourages small parameter values.**
    - **L1 prior (Laplace): Encourages sparsity.**

Impact of Wrong Priors

- If the prior is correct:
    - Less data is needed for accurate estimation.
- If the prior is incorrect:
    - More data is required to correct the estimate.
- A very strong incorrect prior may never be corrected!

Beta Distribution and Pseudo Counts

- **Beta(α, β) Prior:**
  - **Acts like “pseudo counts” in Bayesian updating.**
  - **α: Adds to the count of heads.**
  - **β: Adds to the count of tails.**
- **Example:**
  - **If $\alpha > \beta$ → Higher probability for heads.**
  - **If $\beta > \alpha$ → Higher probability for tails.**

Understanding Bayesian Parameter Estimation

- Find parameters that best explain the data.
- Optimize predictions for new data points.
- Question: Is picking the best single parameter always the right approach?

The Computational Challenge

- Finding $P(\theta|X)$ requires computing P(X).
- P(X) is difficult because we don't know the true distribution of the data.
- Computational complexity increases significantly.

Parameter Estimation Approaches

- **Maximum Likelihood (ML): Ignores the prior, finds the best θ.**
- **Maximum A Posteriori (MAP): Uses a prior but still picks one θ.**
- **Bayesian Inference: Integrates over all θ values but is computationally expensive.**

Prior Distributions in Bayesian Inference

- We assume a prior distribution over parameters.
- Example: In logistic regression, parameters are modeled using a logit function.
- For Linear Discriminant Analysis (LDA), we assumed a Gaussian distribution.

Conjugate Priors & Their Importance

- A conjugate prior simplifies Bayesian inference because the posterior remains in the same family as the prior.
- Example: Beta-Bernoulli Conjugacy
    - If prior: $\theta \sim Beta(\alpha, \beta)$
    - Likelihood: $X \sim Bernoulli\ (\theta)$
    - Then posterior: $\theta \mid X \sim Beta(\alpha + n_1, \beta + n_0)$

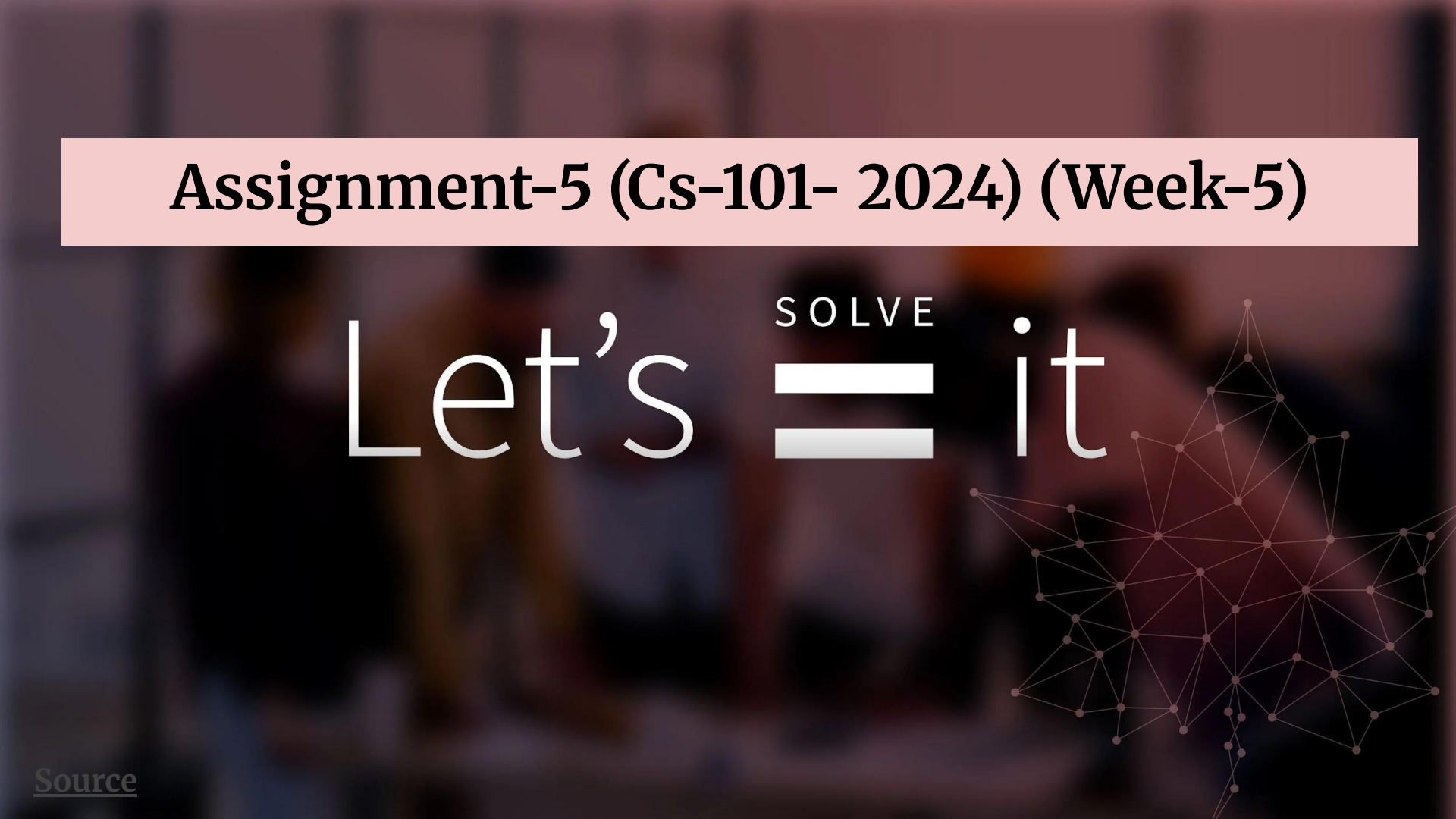
Common Conjugate Pairs

Data Distribution	Conjugate Prior
Bernoulli	Beta
Binomial	Beta
Gaussian (mean)	Gaussian
Multinomial	Dirichlet

Real-World Applications

- **Text Modeling: Multinomial distribution (Unigram model).**
- **Coin Tossing: Bernoulli distribution.**
- **Machine Learning Research: Complex parameter estimation techniques, including non-parametric models.**

Assignment-5 (Cs-101- 2024) (Week-5)



Source

Question-1

01:00

Given a 3 layer neural network which takes in 10 inputs, has 5 hidden units and outputs 10 outputs, how many parameters are present in this network?

a) 500
b) 115
c) 25
d) 100

Question-1- Correct answer

Given a 3 layer neural network which takes in 10 inputs, has 5 hidden units and outputs 10 outputs, how many parameters are present in this network?

a) 500

b) 115 ---[(50+5) + (50+10)]

c) 25

d) 100

Correct options: (b)

Question-2

01:00

We use several techniques to ensure the weights of the neural network are small (such as random initialization around 0 or regularisation). What conclusions can we draw if weights of our ANN are high?

a) Model has overfitted

b) It was initialized incorrectly.

c) At least one of (a) or (b).

d) None of the above

Question-2- Correct answer

We use several techniques to ensure the weights of the neural network are small (such as random initialization around 0 or regularisation). What conclusions can we draw if weights of our ANN are high?

a) **Model has overfitted**

b) **It was initialized incorrectly.**

c) **At least one of (a) or (b).**

d) **None of the above — makes insensitive system for weights change**

Correct options: (d)

Question-3

01:00

In a basic neural network, which of the following is generally considered a good initialization strategy for the weights?

a) Initialize all weights to zero
b) Initialize all weights to a constant non-zero value (e.g., 0.5)
c) Initialize weights with large random values (e.g., between -10 and 10)
d) Initialize weights randomly with small values close to zero

Question-3 - Correct answer

In a basic neural network, which of the following is generally considered a good initialization strategy for the weights?

a) Initialize all weights to zero
b) Initialize all weights to a constant non-zero value (e.g., 0.5)
c) Initialize weights with large random values (e.g., between -10 and 10)
d) Initialize weights randomly with small values close to zero

Correct options: (d)

Question-4

03:00

Recall the XOR(tabulated below) example from class where we did a transformation of features to make it linearly separable. Which of the following transformations can also work?

a) Rotating x1 and x2 by a fixed angle.
b) Adding a third dimension $z=x*y$
c) Adding a third dimension $z=x^2+y^2$
d) None of the above

x1	x2	y
-1	-1	-1
1	-1	1
-1	1	1
1	1	-1

Question-4 - Correct answer

Recall the XOR(tabulated below) example from class where we did a transformation of features to make it linearly separable. Which of the following transformations can also work?

a) **Rotating x1 and x2 by a fixed angle.**
b) **Adding a third dimension z=x*y**
c) **Adding a third dimension z=x2+y2**
d) **None of the above**

Correct options: (b)

Question-5

01:00

Which of the following is the primary reason for rescaling input features before passing them to a neural network?

a) To increase the complexity of the model
b) To reduce the number of parameters in the network
c) To ensure all input features contribute equally to the initial learning process
d) To eliminate the need for activation functions

Question-5 - Correct answer

Which of the following is the primary reason for rescaling input features before passing them to a neural network?

a) To increase the complexity of the model
b) To reduce the number of parameters in the network
c) To ensure all input features contribute equally to the initial learning process
d) To eliminate the need for activation functions

Correct options: (c)

Question-6

01:00

In the Bayesian approach to machine learning, we often use the formula: $P(\theta|D)=P(D|\theta)P(\theta)/ P(D)$ Where θ represents the model parameters and D represents the observed data. Which of the following correctly identifies each term in this formula?

a) $P(\theta|D)$ is the likelihood, $P(D|\theta)$ is the posterior, $P(\theta)$ is the prior, $P(D)$ is the evidence

b) $P(\theta|D)$ is the prior, $P(D|\theta)$ is the evidence, $P(\theta)$ is the likelihood, $P(D)$ is the posterior

c) $P(\theta|D)$ is the evidence, $P(D|\theta)$ is the likelihood, $P(\theta)$ is the posterior, $P(D)$ is the prior

d) $P(\theta|D)$n is the posterior, $P(D|\theta)$ is the likelihood, $P(\theta)$ is the prior, $P(D)$ is the evidence

Question-6 - Correct answer

In the Bayesian approach to machine learning, we often use the formula: $P(\theta|D)=P(D|\theta)P(\theta)/ P(D)$ Where θ represents the model parameters and D represents the observed data. Which of the following correctly identifies each term in this formula?

a) $P(\theta|D)$ is the likelihood, $P(D|\theta)$ is the posterior, $P(\theta)$ is the prior, $P(D)$ is the evidence

b) $P(\theta|D)$ is the prior, $P(D|\theta)$ is the evidence, $P(\theta)$ is the likelihood, $P(D)$ is the posterior

c) $P(\theta|D)$ is the evidence, $P(D|\theta)$ is the likelihood, $P(\theta)$ is the posterior, $P(D)$ is the prior

d) $P(\theta|D)$ is the posterior, $P(D|\theta)$ is the likelihood, $P(\theta)$ is the prior, $P(D)$ is the evidence

Correct options: (d)

Question-7

03:00

Why do we often use log-likelihood maximization instead of directly maximizing the likelihood in statistical learning?

a) Log-likelihood provides a different optimal solution than likelihood maximization
b) Log-likelihood is always faster to compute than likelihood
c) Log-likelihood allows us to avoid using probability altogether
d) Log-likelihood turns products into sums, making computations easier and more numerically stable

Question-7 - Correct answer

Why do we often use log-likelihood maximization instead of directly maximizing the likelihood in statistical learning?

a) Log-likelihood provides a different optimal solution than likelihood maximization
b) Log-likelihood is always faster to compute than likelihood
c) Log-likelihood allows us to avoid using probability altogether
d) Log-likelihood turns products into sums, making computations easier and more numerically stable

Correct options: (d)

Question-8

01:00

In machine learning, if you have an infinite amount of data, but your prior distribution is incorrect, will you still converge to the right solution?

a) Yes, with infinite data, the influence of the prior becomes negligible, and you will converge to the true underlying solution.
b) No, the incorrect prior will always affect the convergence, and you may not reach the true solution even with infinite data.
c) It depends on the type of model used; some models may still converge to the right solution, while others might not.
d) The convergence to the right solution is not influenced by the prior, as infinite data will always lead to the correct solution regardless of the prior.

Question-8- Correct answer

In machine learning, if you have an infinite amount of data, but your prior distribution is incorrect, will you still converge to the right solution?

a) Yes, with infinite data, the influence of the prior becomes negligible, and you will converge to the true underlying solution.
b) No, the incorrect prior will always affect the convergence, and you may not reach the true solution even with infinite data.
c) It depends on the type of model used; some models may still converge to the right solution, while others might not.
d) The convergence to the right solution is not influenced by the prior, as infinite data will always lead to the correct solution regardless of the prior.

Correct options: (a)

Question-9

03:00

Statement: Threshold function cannot be used as activation function for hidden layers.

Reason: Threshold functions do not introduce non-linearity.

a) Both are true and the reason does not explain the statement.

b) Both are true and the reason explains the statement.

c) Statement is false and reason is true.

d) Statement is true and reason is false

Question-9 - Correct answer

Statement: Threshold function cannot be used as activation function for hidden layers.

Reason: Threshold functions do not introduce non-linearity.

a) **Both are true and the reason does not explain the statement.**
b) **Both are true and the reason explains the statement.**
c) **Statement is false and reason is true.**
d) **Statement is true and reason is false – (correct reason : non-differentiable)**

Correct options: (d)

Question-10

01:00

Choose the correct statement (multiple may be correct):

a) MLE is a special case of MAP when prior is a uniform distribution

b) MLE acts as regularisation for MAP

c) MLE is a special case of MAP when prior is a beta distribution

d) MAP acts as regularisation for MLE.

Question-10- Explanation

1 MLE (Maximum Likelihood Estimation)

MLE finds the parameter θ that maximizes the **likelihood function** given observed data D:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} P(D \mid \theta)$$

For a **Bernoulli process** (e.g., coin flips), where each observation $x_i \in \{0, 1\}$ follows:

$$P(x_i \mid \theta) = \theta^{x_i}(1-\theta)^{1-x_i}$$

the likelihood function for n observations is:

$$L(\theta) = P(D \mid \theta) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i}$$

Taking the **log-likelihood**:

$$\log L(\theta) = \sum_{i=1}^{n} x_i \log\theta + (1-x_i)\log(1-\theta)$$

Solving for θ, we get:

$$\hat{\theta}_{MLE} = \frac{\sum x_i}{n} = \frac{k}{n}$$

where k is the number of successes.

Question-10

2 MAP (Maximum A Posteriori Estimation) with a Beta Prior

MAP incorporates prior knowledge about θ. Using **Bayes' Rule**:

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

MAP estimates θ as:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(D \mid \theta)P(\theta)$$

If we assume a **Beta prior** for θ:

$$P(\theta) = \text{Beta}(\alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}$$

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$$

the posterior distribution is:

$$P(\theta \mid D) \propto P(D \mid \theta)P(\theta)$$

Substituting the likelihood and prior:

$$P(\theta \mid D) \propto \theta^k(1-\theta)^{n-k} \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

$$P(\theta \mid D) \propto \theta^{k+\alpha-1}(1-\theta)^{n-k+\beta-1}$$

which is another Beta distribution:

$$P(\theta \mid D) = \text{Beta}(k+\alpha, n-k+\beta)$$

The MAP estimate is the **mode** of this Beta distribution:

$$\hat{\theta}_{MAP} = \frac{k+\alpha-1}{n+\alpha+\beta-2}$$

Question-10

3 How MLE is a Special Case of MAP

If we choose an **uninformative prior**, such as **Beta(1,1)** (which is uniform on $(0, 1)$), then:

$$\alpha = 1, \quad \beta = 1$$

Plugging into the MAP estimate:

$$\hat{\theta}_{MAP} = \frac{k + 1 - 1}{n + 1 + 1 - 2} = \frac{k}{n}$$

which is exactly the **MLE estimate**:

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

Thus, **MLE is a special case of MAP when we use a uniform Beta prior (Beta(1,1))**. If we use a more informative Beta prior (e.g., Beta(2,2)), MAP gives a more regularized estimate.

Question-10

4️⃣ Intuition

- **MLE ignores prior knowledge** and only maximizes the likelihood.
- **MAP incorporates prior knowledge** (e.g., previous beliefs) to adjust the estimate.
- **When the prior is uninformative (Beta(1,1)), MAP reduces to MLE.**
- **When the prior is informative, MAP acts as a regularized version of MLE.**

Question-10

01:00

Choose the correct statement (multiple may be correct):

a) MLE is a special case of MAP when prior is a uniform distribution

b) MLE acts as regularisation for MAP

c) MLE is a special case of MAP when prior is a beta distribution

d) MAP acts as regularisation for MLE.

Question-10- Correct answer

Choose the correct statement (multiple may be correct):

a) MLE is a special case of MAP when prior is a uniform distribution

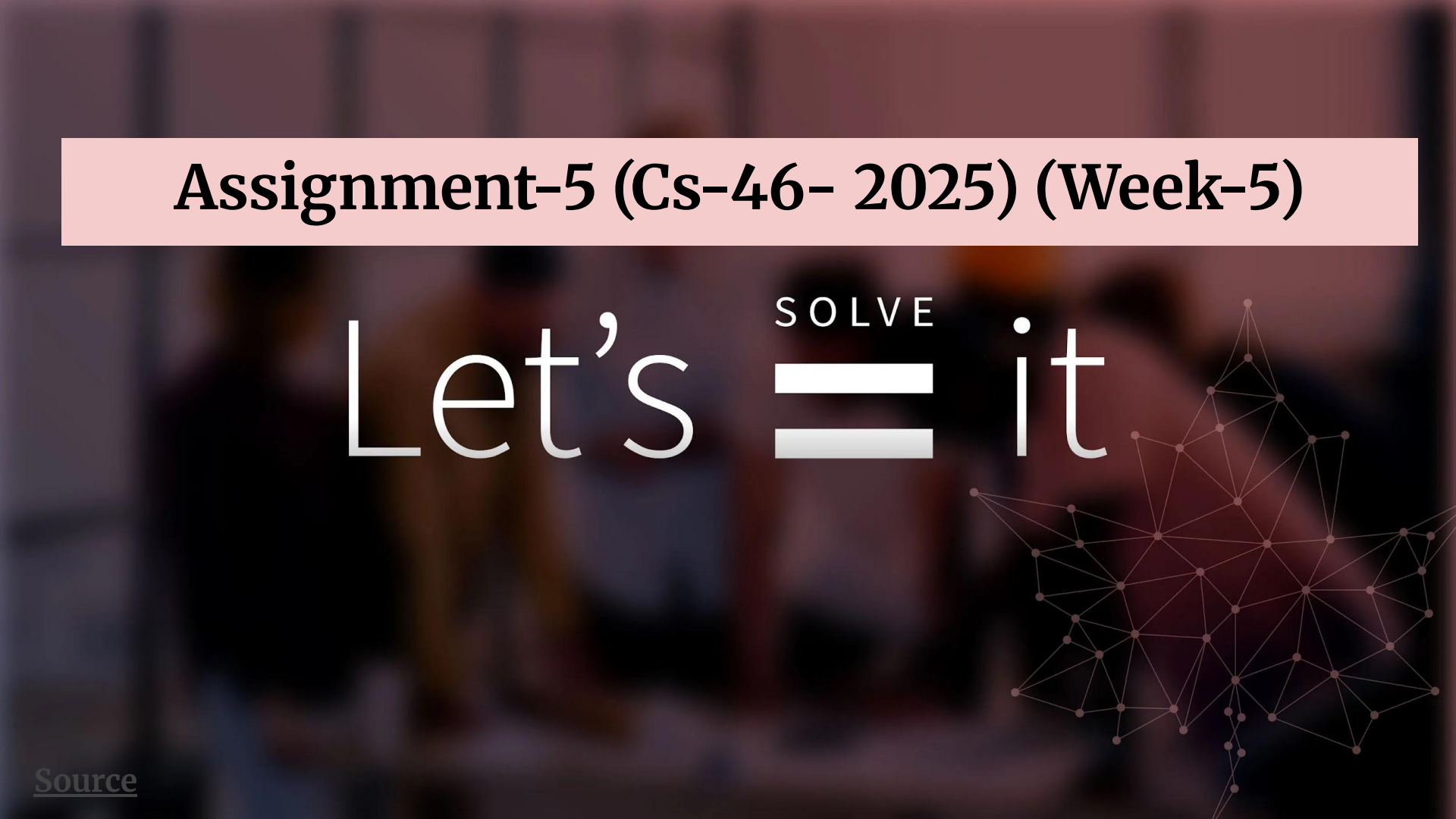
b) MLE acts as regularisation for MAP

c) MLE is a special case of MAP when prior is a beta distribution

d) MAP acts as regularisation for MLE.

Correct options: (a) (d)

Assignment-5 (Cs-46- 2025) (Week-5)



Source

Question-1

01:00

Consider a feedforward neural network that performs regression on a p -dimensional input to produce a scalar output. It has m hidden layers and each of these layers has k hidden units. What is the total number of trainable parameters in the network? Ignore the bias terms.

a) $pk+mk^2+k$

b) $pk+(m-1)k^2+k$

c) $p^2+(m-1)pk+k$

d) $p^2+(m-1)pk+k^2$

Question-1- Correct answer

Consider a feedforward neural network that performs regression on a p -dimensional input to produce a scalar output. It has m hidden layers and each of these layers has k hidden units. What is the total number of trainable parameters in the network? Ignore the bias terms.

a) $pk+mk^2+k$

b) $pk+(m-1)k^2+k$

c) $p^2+(m-1)pk+k$

d) $p^2+(m-1)pk+k^2$

Correct options: (b)

Question-2

01:00

Consider a neural network layer defined as $y=\text{ReLU}(Wx)$. Here $x \in R^p$ is the input, $y \in R^d$ is the output and $W \in R^{d \times p}$ is the parameter matrix. The ReLU activation (defined as $\text{ReLU}(z):=\max(0,z)$ for a scalar z) is applied element-wise to Wx

Find $(\partial y_i / \partial W_{ij})$ where $i=1,..,d$ and $j=1,...,p$.

In the following options, I(condition) is an indicator function that returns 1 if the condition is true and 0 if it is false.

a) $I(\sum^{p}_{k=1} W_{ik} x_k \leq 0) x_i$

b) $I(\sum^{p}_{k=1} W_{ik} x_k > 0) x_j$

c) $I(\sum^{p}_{k=1} W_{ik} x_k > 0) W_{ij} x_j$

d) $I(\sum^{p}_{k=1} W_{ik} x_k \leq 0) W_{ij} x_j$

Question-2- Correct answer

Consider a neural network layer defined as y=ReLU(Wx). Here $x \in R^p$ is the input, $y \in R^d$ is the output and $W \in R^{d \times p}$ is the parameter matrix. The ReLU activation (defined asReLU(z):=max(0,z) for a scalar z) is applied element-wise to Wx Find $(\partial y_i / \partial W_{ij})$ where i=1,..,d and j=1,...,p.

In the following options, I(condition) is an indicator function that returns 1 if the condition is true and 0 if it is false

a) $I(\sum^{p}_{k=1} W_{ik} x_k \leq 0) x_i$

b) $I(\sum^{p}_{k=1} W_{ik} x_k > 0) x_j$

c) $I(\sum^{p}_{k=1} W_{ik} x_k > 0) W_{ij} x_j$

d) $I(\sum^{p}_{k=1} W_{ik} x_k \leq 0) W_{ij} x_j$

Correct options: (b)

Question-3

01:00

Consider a two-layered neural network $y=\sigma(W^{(B)}\sigma(W^{(A)}x))$. Let $h=\sigma(W^{(A)}x)$ denote the hidden layer representation. $W^{(A)}$ and $W^{(B)}$ are arbitrary weights. Which of the following statement(s) is/are true? Note: $\nabla_g(f)$ denotes the gradient of f w.r.t g.

a) $\nabla_h(y)$ depends on $W^{(A)}$

b) $\nabla_{W(A)}(y)$ depends on $W^{(B)}$

c) $\nabla_{W(A)}(h)$ depends on $W^{(B)}$

d) $\nabla_{W(B)}(y)$ depends on $W^{(A)}$

Question-3 - Correct answer

Consider a two-layered neural network $y=\sigma(W^{(B)}\sigma(W^{(A)}x))$. Let $h=\sigma(W^{(A)}x)$ denote the hidden layer representation. $W^{(A)}$ and $W^{(B)}$ are arbitrary weights. Which of the following statement(s) is/are true? Note: $\nabla_g(f)$ denotes the gradient of f w.r.t g.

a) $\nabla_h(y)$ depends on $W^{(A)}$

b) $\nabla_{W(A)}(y)$ depends on $W^{(B)}$

c) $\nabla_{W(A)}(h)$ depends on $W^{(B)}$

d) $\nabla_{W(B)}(y)$ depends on $W^{(A)}$

Correct options: (b)(d)

Question-4

03:00

5) Consider the following statements about the derivatives of the sigmoid $(\sigma(x) = \frac{1}{1+exp(-x)})$ and tanh $(tanh(x) = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)})$ activation functions. Which of these statement(s) is/are correct?

- [ ] $\sigma'(x) = \sigma(x)(1-\sigma(x))$
- [ ] $0 < \sigma'(x) \leq \frac{1}{4}$
- [ ] $tanh'(x) = \frac{1}{2}(1-(tanh(x))^2)$
- [ ] $0 < tanh'(x) \leq 1$

- [ ] $\sigma'(x) = \sigma(x)(1-\sigma(x))$
- [ ] $0 < \sigma'(x) \leq \frac{1}{4}$
- [ ] $tanh'(x) = \frac{1}{2}(1-(tanh(x))^2)$
- [ ] $0 < tanh'(x) \leq 1$

Question-4

03:00

5) Consider the following statements about the derivatives of the sigmoid $(\sigma(x) = \frac{1}{1+exp(-x)})$ and tanh $(tanh(x) = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)})$ activation functions. Which of these statement(s) is/are correct?

- [ ] $\sigma'(x) = \sigma(x)(1-\sigma(x))$
- [ ] $0 < \sigma'(x) \leq \frac{1}{4}$
- [ ] $tanh'(x) = \frac{1}{2}(1-(tanh(x))^2)$
- [ ] $0 < tanh'(x) \leq 1$

- [ ] $\sigma'(x) = \sigma(x)(1-\sigma(x))$
- [ ] $0 < \sigma'(x) \leq \frac{1}{4}$
- [ ] $tanh'(x) = \frac{1}{2}(1-(tanh(x))^2)$
- [ ] $0 < tanh'(x) \leq 1$

Correct options:

Accepted Answers:

$\sigma'(x) = \sigma(x)(1-\sigma(x))$

$0 < \sigma'(x) \leq \frac{1}{4}$

$0 < tanh'(x) \leq 1$

Question-5

01:00

Which of the following statement(s) about the initialization of neural network weights is/are true for a network that uses the sigmoid activation function?

a) Two different initializations of the same network could converge to different minima
b) For a given initialization, gradient descent will converge to the same minima irrespective of the learning rate.
c) Initializing all weights to the same constant value leads to undesirable results
d) Initializing all weights to very large values leads to undesirable results

Question-5 - Correct answer

Which of the following statement(s) about the initialization of neural network weights is/are true for a network that uses the sigmoid activation function?

a) Two different initializations of the same network could converge to different minima

b) For a given initialization, gradient descent will converge to the same minima irrespective of the learning rate.

c) Initializing all weights to the same constant value leads to undesirable results

d) Initializing all weights to very large values leads to undesirable results

Correct options: (a)(c)(d)

Question-6

01:00

A geometric distribution is defined by the p.m.f. $f(x;p)=(1-p)^{(x-1)}p$ for x=1,2,..... Given the samples [4, 5, 6, 5, 4, 3] drawn from this distribution, find the MLE of p.

a) 0.111

b) 0.222

c) 0.333

d) 0.444

Question-6 - Correct answer

A geometric distribution is defined by the p.m.f. $f(x;p)=(1-p)^{(x-1)}p$ for $x=1,2,.....$ Given the samples [4, 5, 6, 5, 4, 3] drawn from this distribution, find the MLE of p.

a) 0.111
b) 0.222
c) 0.333
d) 0.444

Correct options: (b)

Question-7

03:00

Consider a Bernoulli distribution with p=0.7 (true value of the parameter). We draw samples from this distribution and compute an MAP estimate of p by assuming a prior distribution over p. Let $N(\mu,\sigma^2)$ denote a gaussian distribution with a mean μ and variance σ^2.Distributions are normalized as needed. Which of the following statement(s) is/are true?

a) If the prior is N(0.6,0.1) , we will likely require fewer samples for converging to the true value than if the prior is N(0.4,0.1)

b) If the prior is N(0.4,0.1), we will likely require fewer samples for converging to the true value than if the prior is N(0.6,0.1)

c) With a prior of N(0.1,0.001), the estimate will never converge to the true value, regardless of the number of samples used.

d) With a prior of U(0,0.5) (i.e. uniform distribution between 0 and 0.5), the estimate will never converge to the true value, regardless of the number of samples used.

Question-7 - Correct answer

Consider a Bernoulli distribution with p=0.7 (true value of the parameter). We draw samples from this distribution and compute an MAP estimate of p by assuming a prior distribution over p. Let $N(\mu,\sigma^2)$ denote a gaussian distribution with a mean μ and variance σ^2.Distributions are normalized as needed. Which of the following statement(s) is/are true?

a) If the prior is N(0.6,0.1) , we will likely require fewer samples for converging to the true value than if the prior is N(0.4,0.1)

b) If the prior is N(0.4,0.1), we will likely require fewer samples for converging to the true value than if the prior is N(0.6,0.1)

c) With a prior of N(0.1,0.001), the estimate will never converge to the true value, regardless of the number of samples used.

d) With a prior of U(0,0.5) (i.e. uniform distribution between 0 and 0.5), the estimate will never converge to the true value, regardless of the number of samples used.

Correct options: (a)(d)

Question-8

01:00

Which of the following statement(s) about parameter estimation techniques is/are true?

a) To obtain a distribution over the predicted values for a new data point, we need to compute an integral over the parameter space.

b) The MAP estimate of the parameter gives a point prediction for a new data point.

c) The MLE of a parameter gives a distribution of predicted values for a new data point.

d) We need a point estimate of the parameter to compute a distribution of the predicted values for a new data point.

Question-8- Correct answer

Which of the following statement(s) about parameter estimation techniques is/are true?

a) To obtain a distribution over the predicted values for a new data point, we need to compute an integral over the parameter space.

b) The MAP estimate of the parameter gives a point prediction for a new data point.

c) The MLE of a parameter gives a distribution of predicted values for a new data point.

d) We need a point estimate of the parameter to compute a distribution of the predicted values for a new data point.

Correct options: (a) (b)

Question-9

03:00

In a classification setting, it is common in machine learning applications to minimize the discrete cross entropy loss given by $HCE(p,q)=-\Sigma_i p_i \log q_i$ where p_i and q_i are the true and predicted distributions ($p_i \in \{0,1\}$ depending on the label of the corresponding entry). Which of the following statement(s) about minimizing the cross entropy loss is/are true?

a) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing the (self) entropy $H(q)$

b) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing $H_{CE}(q,p)$

c) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing the KL divergence $D_{KL}(p||q)$

d) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing the KL divergence $D_{KL}(q||p)$

Question-9 - Correct answer

In a classification setting, it is common in machine learning applications to minimize the discrete cross entropy loss given by $H_{CE}(p,q)=-\Sigma_i p_i \log q_i$ where p_i and q_i are the true and predicted distributions ($p_i \in \{0,1\}$ depending on the label of the corresponding entry). Which of the following statement(s) about minimizing the cross entropy loss is/are true?

a) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing the (self) entropy $H(q)$

b) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing $H_{CE}(q,p)$

c) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing the KL divergence $D_{KL}(p||q)$

d) Minimizing $H_{CE}(p,q)$ is equivalent to minimizing the KL divergence $D_{KL}(q||p)$

Correct options: (c)

Question-10

01:00

Which of the following statement(s) about activation functions is/are NOT true?

a) Non-linearity of activation functions is not a necessary criterion when designing very deep neural networks
b) Saturating non-linear activation functions (derivative $\to 0$ as $x \to \pm\infty$) avoid the vanishing gradients problem
c) Using the ReLU activation function avoids all problems arising due to gradients being too small.
d) The dead neurons problem in ReLU networks can be fixed using a leaky ReLU activation function

Question-10- Correct answer

Which of the following statement(s) about activation functions is/are NOT true?

a) Non-linearity of activation functions is not a necessary criterion when designing very deep neural networks

b) Saturating non-linear activation functions (derivative →0 as $x \to \pm\infty$) avoid the vanishing gradients problem

c) Using the ReLU activation function avoids all problems arising due to gradients being too small.

d) The dead neurons problem in ReLU networks can be fixed using a leaky ReLU activation function

Correct options: (a) (d)


THANK YOU

Suggestions and Feedback

Next Session:

Sunday: 31-Aug-2025
3:00 - 5:00 PM