UCS2723 DEEP LEARNING

Unit 1

Deep Networks Basics

Linear Algebra : Scalars - Vectors - Matrices and Tensors; Probability Distributions - Gradient-based Optimization - Machine Learning Basics: Capacity - Overfitting and underfitting - Hyperparameters and validation sets - Estimators - Bias and variance - Stochastic gradient descent - Challenges motivating deep Learning; Deep Networks: Deep feedforward networks; Regularization - Optimization

* Linear Algebra

→ concerning linear equations such as $a_1x_1 + \cdots + a_nx_n = b$

$$a^T x = b$$

→ deals with the linear transformations of variables. - in ML, convert input vectors $(x_1, \ldots x_n)$ into outputs by a series of linear transformations.

* Scalars

→ a single no. → integers
→ real numbers
→ rational numbers

* Vectors

→ an ordered array of numbers and can be in a row or colum

eg. $\mathbb{R}^n$ can be $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

* Matrix

→ a 2D array of numbers

for eq. $A \in \mathbb{R}^{m \times n}$ could be $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$

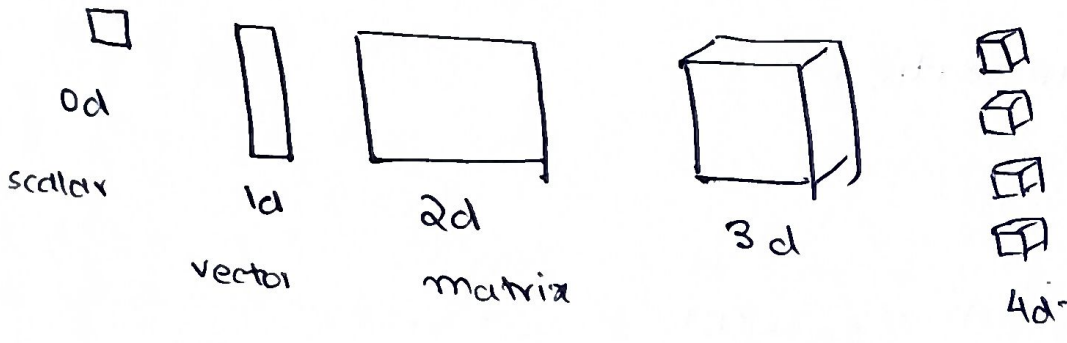
* Tensors

→ Sometimes we need an array with more than two axes

eq. an RGB color image has 3 axes

→ A tensor is an array of numbers arranged on a regular grid with a variable number of axes.

Element (i,j,k) of a tensor is denoted by $A_{i,j,k}$



* Matrix Transpose

$(A^T)_{i,j} = A_{j,i}$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix} \qquad A^T = \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \end{bmatrix}$$

Note: $(AB)^T = B^T A^T$

* Matrix (Dot) Product

$C = AB$

$c_{ij} = \sum_k A_{ik} B_{kj}$

→ The product of the individual elements of a matrix is called the element-wise product or the Hadamard product, and is denoted as $A \odot B$

*System of Equations

$Ax = b$

expands to $A_1 : x = b_1$

$A_2 : x = b_2$

....

$A_m : x = b_m$

A linear system can have:

(i) no solution

(ii) many solutions

(iii) exactly one solution - this means that multiplication by the matrix is an invertible function

* Matrix Inversion

→ The matrix inverse of A is denoted as A^{-1}, such that $A^{-1}A = I$

→ The matrix inverse can be arrived at as follows:

$$Ax = b$$

$$A^{-1}Ax = A^{-1}b$$

$$IX = A^{-1}b$$

$$\boxed{X = A^{-1}b}$$

* Invertability → Matrices can't be inverted if there are:

(i) not a square matrix

(ii) redundant rows & columns (linearly dependent / low rank cases)

* Linear Combinations and Span

Let $v_1, v_2 \ldots v_n$ be vectors in $\mathbb{R}^n$. A linear combination of these vectors is any expression of the form:

$k_1v_1 + k_2v_2 + \cdots + k_rv_r$, where the coefficients $k_1, k_2, \ldots k_r$ are scalars.

→ The set of all linear combinations of a collection of vectors $v_1, v_2 \ldots v_r$ from $\mathbb{R}^n$ is called the span of $v_1, v_2 \ldots v_r$ and the span is always a subspace of $\mathbb{R}^n$.

Linear Combinations and Span

Suppose $V = \mathbb{R}^3$

The vector $\begin{bmatrix} 7 \\ 2 \\ 9 \end{bmatrix}$ is a combination of $\left(\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right)$

because $\begin{bmatrix} 7 \\ 2 \\ 9 \end{bmatrix} = 2\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} + 3\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Thus $\begin{bmatrix} 7 \\ 2 \\ 9 \end{bmatrix} \in \text{span} \left(\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right)$

* Linear Dependence and Independence

→ Let $A = v_1, v_2 \cdots v_r$ be a collection of vectors from $\mathbb{R}^n$

→ If $r > 2$ and at least one of the vectors in A can be written as a linear combination of the others, then A is said to be linearly dependent

→ On the other hand, if no vector in A can be written as a linear combination of others, then A is said to be linearly independent.

* Norms

→ A norm is a way to measure the size of a vector, a matrix or a tensor. They enable us to quantify the magnitude of a vect

It is given by $||x||$.

Properties of Norm

1. Non-negativity - The norm should always be ≥ 0.

2. Definiteness - It is zero iff the vector is zero, i.e the zero vector.

$$f(x) = 0 \implies x = 0$$

3. Triangle Inequality - The norm of a sum of two vectors is no more than the sum of their norms.

$$f(x) + f(y) \geq f(x) + f(y)$$

4. Homogenity - Multiplying a vector by a scalar multiples the norm of the vector by the absolute value of the scalar.

$$\forall \alpha \in \mathbb{R}, \; f(\alpha x) = \alpha f(x)$$

Applications of Norm

→ used when defining loss function - i.e., the distance between the actual and predicted values

→ used as a regularization method in ML, eg in ridge and lasso regularization methods

→ algorithms like SVM use norms to calculate the distance between the discriminant and each support vector.

Types of Norms

(i) LP norm $\|x\|_p = \left(\sum_i |x_i|^p \right)^{1/2}$

(ii) L2 norm = Euclidean norm - $p=2$

(iii) L1 norm - $p=1$, $\|x\|_1 = \sum_i |x_i|$

(iv) max norm, ∞ p : $\|x\|_\infty = \max_i |x_i|$

* Eigen Decomposition

→ Decompose matrices in ways that show us information about their functional properties that is not obvious from the representation of the matrix as an array of elements

→ Eigen decomposition involves decomposing a matrix into a set of eigen vectors and eigen values

For a square matrix A

$$AV = \lambda V$$

or $$A = V \lambda V^{-1}$$

Steps :

1. Solve the characteristic equation $\det(A - \lambda I) = 0$ to find the eigen values λ.

2. For each eigen value λ, solve the equation $(A - \lambda I)v = 0$ to find the corresponding eigenvector v.

3. Form matrices V and Λ

   a. Construct V using the eigen vectors as columns

   b. Construct Λ as a diagonal matrix with eigen values on the diagonal.

* Singular Value Decomposition

→ provides a means to factorize a matrix into singular vectors and singular values

→ Matrix need not be square

$$A = UDV^T$$

* Probability

→ Machine learning must always deal with uncertain quantities, and stochastic / non-deterministic quantities

→ Probability theory provides a set of formal rules for determining the likelihood of a proposition being true given the likelihood of other propositions

Random Experiment - A process / procedure that produces one of a set of possible outcomes in an unpredictable manner. The set of all possible outcomes is the sample space

* Probability Distributions

→ provides a mathematical function that gives the probabilities of occurrence of different possible outcomes - can be discrete or continuous distributions

A. Discrete Probability Distribution

→ applies to scenarios where the set of possible outcomes is countable - eg. rolling a die, flipping a coin.

Probability Mass Function : A function $P(X=x)$ that gives the probability that a discrete random variable X takes on the value x.

eg. for a die - $P(X=x) = \frac{1}{6}$ for x in $\{1, 2, 3, 4, 5, 6\}$

Cumulative Distribution Function : A function $F(x)$ that gives the probability that the random variable X is less than or equal to x. $F(x) = P(X \leq x)$

For a die :

$$F(x) = \begin{cases} 0 , & x < 1 \\ 1/6 , & 1 \leq x < 2 \\ 2/6 , & 2 \leq x < 3 \\ 3/6 , & 3 \leq x < 4 \\ 4/6 , & 4 \leq x < 5 \\ 5/6 , & 5 \leq x < 6 \\ 1 , & x \geq 6 \end{cases}$$

B. **Continuous Probability Distribution**

→ applies to scenarios where the set of possible outcomes is uncountable.

Probability Distribution Function - The probability that X falls within a particular interval is the area of the curve $f(x)$ over that interval.

For eg. for a standard normal distribution.

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Cumulative Distribution Function $F(x) = P(X \leq x)$ - $F(x)$ is the integral of the PDF from $-\infty$ to x.

For a standard normal distribution, the CDF $F(x)$ is:

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

* Joint Probability Distributions

→ refers to the probability of 2 or more events occurring simultaneously denoted by $P(A \cap B)$

→ For discrete variables: given by $P(X=x, Y=y)$

→ For continuous variables: $f(x,y)$ describes the likelihood that X & Y take on specific values within a given range

* Marginal Probability

→ The marginal probability of an event is obtained by summing/integrating the joint probabilities over all possible values of other variables

Discrete → $P(X=x) = \sum_{y} P(X=x, Y=y)$

Continuous → $f_X(x) = \int_{-\infty}^{\infty} f(x,y)\, dy$

* Conditional Probability

→ The conditional probability of event A given event B is the probability of A occurring under the condition that B has occurred

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

* Gradient - based Optimization

→ Most ML algorithms involve optimization - minimize/ maximize an objective function $f(x)$

For a given function: $y = f(x)$

Find the derivative: $y' = f'(x)$

→ The derivative $f'(x)$ gives the slope of $f(x)$ at point x

→ It specifies how to scale a small change in input to obtain a corresponding change in the output y.

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x)$$

→ Reduce $f(x)$ by moving x in small steps with the opposite sign of the derivate.

→ This technique is called gradient descent.

* Stationary Points

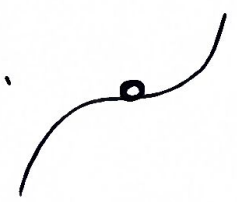
→ A stationary point occurs when the gradient of the function is zero, i.e $\nabla f(\theta) = 0$

Types of Stationary points

① Local Minimum : A stationary point where the function value is lower than at all nearby points

② Local Maximum : A stationary point where the function value is higher than all nearby points.

③ Saddle Point : A stationary point where the function has a minimum along some points and a maximum along other directions. Saddle points are challenging in optimization because the gradient does not provide a clear direction to move in.



*** Note that optimization algorithms may fail to find the global minimum - generally such solutions are accepted too.

* Method of Gradient Descent

→ The gradient points directly uphill, and the negative gradient points downhill.

→ f can be decreased by moving in the direction of the negative gradient

→ The method of gradient / steepest descent proposes a new point:

$$x' = x - \epsilon \nabla_x f(x)$$

ϵ = learning rate, a +ve scalar constant

Can avoid iterative algorithm by solving the eqn $\nabla_x f(x) = 0$ for x

* Choosing ϵ

→ ϵ can be chosen in several ways.

→ A popular approach is to set ϵ to a small constant

→ Another approach is line-search - Evaluate $f(x - \epsilon \nabla_x f(x))$ for several values of ϵ, and choose the one that results in the

Smallest objective function value.

* Gradient Descent on least Squares

→ Consider a linear regression model where we want to fit a line to a set of data points. This can be represented as:

$$y = X\beta + \varepsilon$$

y = vector of observed values

X = matrix of input features

β = vectors of parameters (coefficients) to be estimated

ε = vector of errors - residuals

→ The goal is to find the parameter vector β that minimizes the sum of squared residuals

$$J(\beta) = \|X\beta - y\|^2$$

$$= (X\beta - y)^T (X\beta - y)$$

→ Calculate the gradient wrt β

$$\nabla J(\beta) = 2X^T (X\beta - y).$$

→ The gradient descent update rule is:

$$\beta_{new} = \beta - \alpha \nabla J(\beta)$$

α → learning rate

* Generalization of Gradient Descent to Discrete Spaces

→ Gradient descent is limited to continuous spaces

→ The concept of repeatedly making the best small move can be generalized to discrete spaces

→ Ascending an objective function of discrete parameters is called hill climbing.

* Machine Learning Basics

ML → concerned with computer programs that automatically improve their performance through experience.

Why ML → (i) Develop systems that can automatically adapt & customize themselves to new users

(ii) Discover new knowledge from large databases (data mining)

(iii) can replace certain monotonous tasks

(iv) Develop systems that are too difficult/ expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task. (knowledge - engineering bottleneck)

Concept of learning in an ML System

Learning = Improving w/ experience at some task

→ Improve over task T

→ w.r.t performance measure P

→ based on experience, E.

→ dataset

→ The learning process has the following steps:

(i) measuring devices / sensors

(ii) preprocessing — noise filtering, feature extraction, normalization

(iii) dimensionality reduction - feature selection, feature projection

(iv) Model learning - regression, clustering, description

(v) Model testing - cross-validation bootstrapping

* ML Algorithms
  - Supervised - continuous / categorical target var
  - Unsupervised - no target var
  - Semi-supervised - categorical label for some samples
  - Reinforcement

*Generalization

→ The ability of the ML model to perform well on previously unobserved inputs is called generalizability.

→ The generalization error is the expected value of the error on a new input

→ The expected value is calculated as the average over inputs taken from the distribution

underfitting & overfitting in unit 4

* Model Capacity

→ Model capacity refers to the ability of a model to fit a wide variety of functions or patterns in the data

→ It is determined by the complexity of the model, including the no of parameters, & the depth of the neural network

High Capacity?

Advantages: Capable of fitting complex patterns and capturing intricate relationships in the data

Disadvantages: High risk of overfitting

Low Capacity?

Advantages: less likely to overfit, simpler models that are easier to interpret and train

Disadvantages: Risk of underfitting

Appropriate Capacity

→ ML algorithms will perform well well when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with.

Representational Capacity → The range or family of functions that a learning algorithm can choose from. It is determined by the design and architecture of the model.

High Representational Capacity - models with many parameters and complex structures - eg. deep neural networks have high representational capacity. They can represent a wide variety of functions and capture intricate patterns in the data.

Low Representational Capacity → Simpler models with fewer parameters (eg. linear regression) have lower representational capacity. They can represent only a limited range of functions and

may not capture complex patterns.

Effective Capacity - The actual capacity of the model that is realized during training. It can be limited by imperfections in the optimization algorithm & practical constraints.

(i) Optimization limitations - Even if a model has high representational capacity, the optimization algorithm might not find the best parameters due to issues like local minima, saddle points or poor convergence.

(ii) Training Constraints - Factors such as insufficient training data, computational resources, and suboptimal hyperparameter settings can reduce the effective capacity of the model.

Hyperparameters, Cross-validation in unity

* Estimators, Bias and Variance

* Point Estimation

→ Point estimation is the attempt to provide the single 'best' prediction of some quantity of interest

→ The quantity of interest can be a single parameter or a vector of parameters in a parametric model.

→ Point estimates are denoted by $\hat{\theta}$, where θ is the true parameter.

. Formally,

→ Let $\{x^{(1)}, \ldots x^{(m)}\}$ be a set of m independent and identically distributed data points

→ A point estimator is any statistic (function of the data):

$$\hat{\theta}_m = g(x^{(1)}, \ldots x^{(m)})$$

* Properties of a good estimator

→ While any function qualifies as an estimator, a good estimator is one whose output is close to the true underlying θ that generated the data.

(i) Unbiasedness - The expected value of the estimator equals the true parameter value.

(ii) Consistency - The estimator converges to the true parameter value as the sample size increases.

(iii) Efficiency - The estimator has the smallest variance among all unbiased estimators.

* Function Estimators

→ Point estimation can also refer to estimating the relationship between input and target variables

→ These types of point estimates are known as function estimators

→ eg. estimating the value in a linear regression model

→ Function estimation involves predicting a variable y given an input vector x.

→ Assume there exists a function $f(x)$ that describes the relationship between y & x.

→ Model ⇒ $y = f(x) + \varepsilon$ → error

→ The objective is to approximate f with an estimate $\hat{f}$, $\hat{f}$ is a point estimator in function space, similar to estimating a parameter θ.

* Bias of an Estimator

→ Bias of an estimator θ_m is defined as:

$$\text{bias}(\hat{\theta}_m) = \mathbb{E}[\hat{\theta}_m] - \theta$$

where $\mathbb{E}[\hat{\theta}_m]$ is the expected value of the estimate and θ is the true underlying value used to define the data generating distribution.

* Unbiased and Asymptomatically Unbiased Estimators

Unbiased Estimator

→ An estimator $\hat{\theta}_m$ is unbiased if $\text{bias}(\hat{\theta}_m) = 0$

→ This implies that $\mathbb{E}[\hat{\theta}_m] = \theta$

Asymptomatically Unbiased Estimator

→ $\hat{\Theta}_m$ is asymptomatically unbiased if:

$$\lim_{m \to \infty} \text{bias}(\hat{\Theta}_m) = 0$$

→ This implies that $\lim_{m \to \infty} \mathbb{E}[\hat{\Theta}_m] = \Theta$

* Variance of an Estimator

→ Variance measures how much we expect an estimator to vary based on different data samples - $\text{Var}(\hat{\Theta})$

→ A low-variance estimator is desirable, as it indicates that the estimator's output doesn't fluctuate significantly with different data samples.

→ The standard error is the square root of the variance of the estimator

$$\text{SE}(\hat{\Theta}) = \sqrt{\text{Var}(\hat{\Theta})}$$

→ use this as compared to variance, as the sample size m increases.

* Bias-Variance Trade-off

Bias: Measures the expected deviation of the estimators from the true parameter value

Variance: Measures the deviation of the estimator from its expected value, caused by the particular sampling of the data.

→ The MSE combines both bias and variance into a single metric:

$$MSE = \mathbb{E}\left[(\hat{\theta}_m - \theta)^2\right] = Bias(\hat{\theta}_m)^2 + Var(\hat{\theta}_m)$$

→ helps evaluate the overall accuracy of an estimator by accounting for both sources of error.

* Challenges motivating Deep Learning

* Drawbacks of Traditional Machine Learning Algorithms

1. Limited Feature Engineering
2. Scalability Issues
3. Curse of Dimensionality
4. Inability to handle complex data
5. Overfitting and Underfitting
6. Dependency on Human Expertise
7. Local Generalization
8. Difficulty in Modeling Non-linear Relationships

* Motivation of Deep Learning

1. Automated Feature Engineering
2. Scalability to large datasets
3. Ability to handle high dimensionality (through hierarchical representations)

4. Modeling Complex and Non-Linear Relationships

5. Learning from unstructured data

6. End-to-end learning

7. Generalization to new data

8. Reduced need for domain expertise

9. Incorporation of Prior knowledge (pre-trained models)

10. Overcoming Template Matching limitations

* Local Constancy, Smoothness and its disadvantages in ML

→ ML algorithms need prior beliefs to generalize well.

→ The most widely used prior is smoothness, also known as local constancy which states that the function should not change much within a small region.

→ However, relying exclusively on smoothness may fail in complex AI tasks. Deep learning introduces additional priors to improve generalization.

→ Kernel machines use smoothness, but are limited by their reliance on template matching, which deep learning overcomes.

→ Decision trees also depend on smoothness, breaking the input space into regions, but require many training samples to ensure statistical confidence.

*Manifold Learning

→ Manifold learning is an unsupervised learning technique aimed at discovering the low-dimensional structure embedded within high-dimensional data.

→ The key idea is that data, although high-dimensional, often lies on a much lower-dimensional manifold.

→ This approach is useful for dimensionality reduction, preserving the intrinsic structure of the data, and simplifying analysis.

→ A manifold is a space that locally resembles a Euclidean space, but has a much more complex global structure.

Popular Manifold learning Algorithms

1. Principal Component Analysis - Reduces data dimensions by identifying principal components.

2. Isomap: Computes shortest paths on the manifold and applies multidimensional scaling to preserve distance.

3. Locally Linear Embedding (LLE): Preserves local relationships by assuming a locally linear patch on the manifold.

4. t-Distributed Stochastic Neighbor Embedding (t-SNE): Visualizes high-dimensional data into 2 or 3 dimensions, preserving point distribution.

* Applications of Manifold Learning?

1. Image and Video Analysis - dimensionality reduction for tasks like object or face recognition

2. Natural Language Processing - finding low dimensional representations of text that capture semantics

3. Data Visualization - Reducing complex datasets to 2 or 3 dimensions for pattern and anomaly detection.

* Importance of Manifold Learning in Deep Learning?

→ crucial for understanding and capturing the underlying structure of data

→ for eg. CNNs in image processing implicitly perform manifold learning, extracting hierarchical features and representing data on a lower-dimensional manifold - leads to more efficient learning.

* Feedforward and Deep Feedforward Networks

→ Deep feedforward networks, or also called feed forward neural networks, or multilayer perceptrons (MLPs)

→ The goal of a feedforward network is to approximate some function f^*.

→ For example, for a classifier, $y = f^*(x)$ maps an input x to a category y.

→ A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best fn.

approximation

→ These models are called feedforward because there are no feedforward connections in which the outputs of the model are fed back into itself.

Structure

→ The feedforward model is associated with a directed acyclic graph, describing the composition of the functions

→ For eg. if 3 functions $f(1)$, $f(2)$ and $f(3)$ are connected in a chain to form $f(x) = f(3)(f(2)(f(1)(x)))$

$f(1)$ = First layer

$f(2)$ = second layer

The overall length of the chain gives the depth of the model

→ The composite function $f(x) = f(3)(f(2)(f(1)(x)))$ has a depth of 3.

→ The final layer of a feedforward network is called the output layer here $f(3)$ is the output layer.

Training

→ During neural network training, the function $f(x)$ is driven to match $f^*(x)$ evaluated using diff. training points

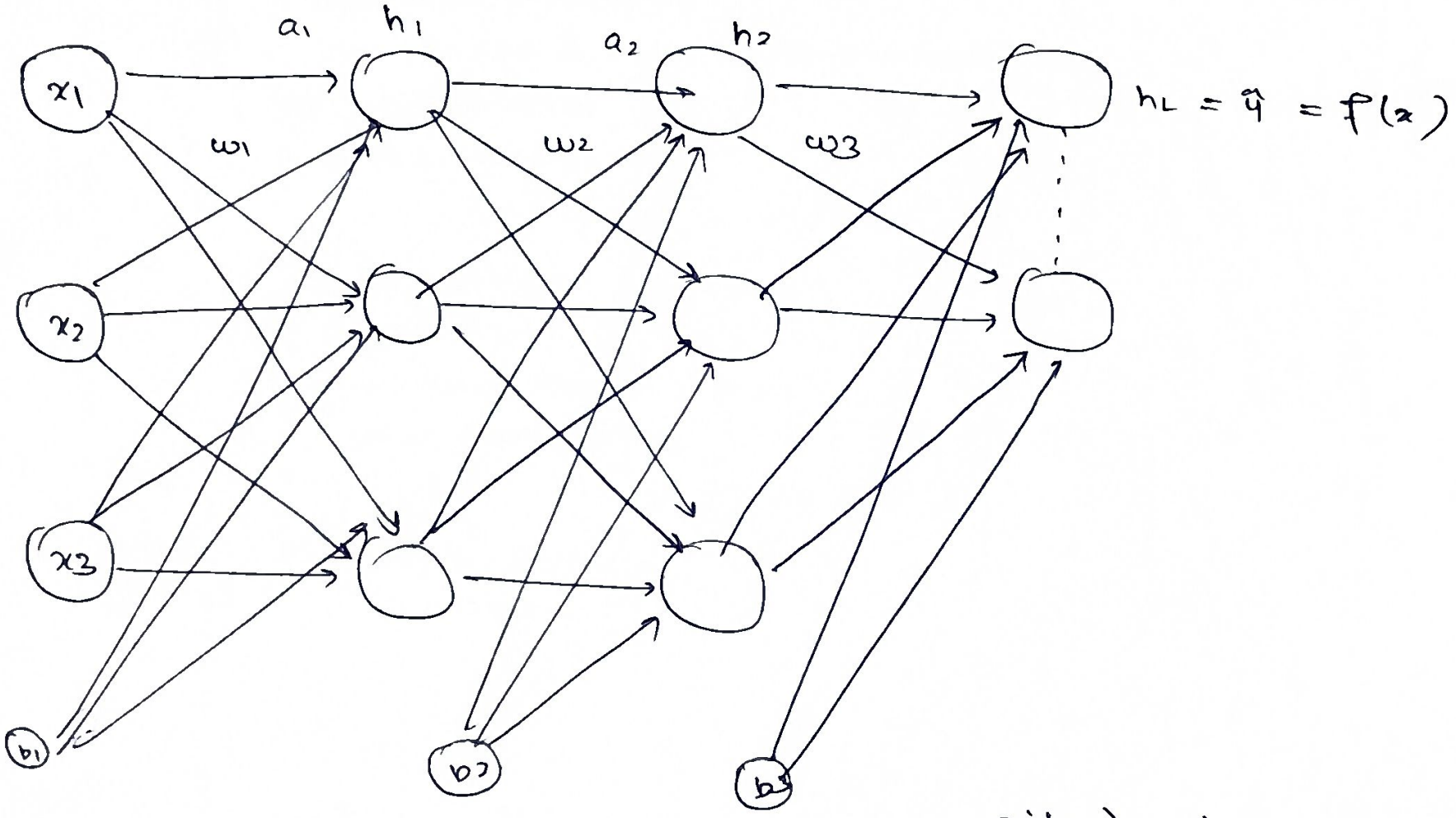
→ Each example x is accompanied by a label $y \approx f^*(x)$.

→ Each training examples should produce a value that is close to y.

→ The behavior of the other layers is not directly specified by the training data.

→ The learning algorithm must decide how to use other layers to best implement an approximation of f^*

→ Because the training data does not show the desired output for each of these layers, these layers are called <u>hidden layers</u>



To calculate values at the activation layer

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} + \begin{bmatrix} w_{111} & w_{112} & w_{113} \\ w_{121} & w_{122} & w_{123} \\ w_{131} & w_{132} & w_{133} \end{bmatrix} \begin{bmatrix} h_{01} = x_1 \\ h_{02} = x_2 \\ h_{03} = x_3 \end{bmatrix}$$

→ The pre-activation layer at layer i is given by:

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

→ The activation at layer ai is given by

$$h_i(x) = g(a_i(x))$$

g = activation function – logistic / tanh / linear etc.

→ The activation at layer i is given by:

$$f(x) = h_L(x) = O(a_L(x))$$

O = output activation fn – softmax / linear etc

* XOR Problem

Deep Learning

Unit 1 Problems

Question 1: Given the matrix:

$$A = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix},$$

1. Find the eigenvalues of the matrix A.
2. Determine the eigenvectors corresponding to each eigenvalue.

1. **Find Eigenvalues:**
   - Solve $\det(A - \lambda I) = 0$:
     $\det \begin{pmatrix} 4-\lambda & 1 \\ 2 & 3-\lambda \end{pmatrix} = 0$
     $(4-\lambda)(3-\lambda) - 2 = \lambda^2 - 7\lambda + 10 = 0$
     $(\lambda - 5)(\lambda - 2) = 0$
   - The eigenvalues are $\lambda_1 = 5$ and $\lambda_2 = 2$.
2. **Find Eigenvectors:**
   - For $\lambda_1 = 5$:
     $(A - 5I)v = 0$
     $\begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$
     - One eigenvector corresponding to $\lambda_1 = 5$ is $v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.
   - For $\lambda_2 = 2$:
     $(A - 2I)v = 0$
     $\begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$
     - One eigenvector corresponding to $\lambda_2 = 2$ is $v_2 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$.
3. **Form Matrices V and Λ:**
   - $V = \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix}$
   - $\Lambda = \begin{pmatrix} 5 & 0 \\ 0 & 2 \end{pmatrix}$

Thus, the eigenvalue decomposition of matrix A is:

$$A = V\Lambda V^{-1}$$

Question 2:

Let $x = \begin{pmatrix} 7 \\ 2 \\ 9 \\ 10 \end{pmatrix}$. What is $3 \cdot x$?

Compute $3 \cdot x$:

$$3 \cdot x = 3 \cdot \begin{pmatrix} 7 \\ 2 \\ 9 \\ 10 \end{pmatrix} = \begin{pmatrix} 21 \\ 6 \\ 27 \\ 30 \end{pmatrix}.$$

Answer: $3 \cdot x = \begin{pmatrix} 21 \\ 6 \\ 27 \\ 30 \end{pmatrix}$.

Question 3:

Let u and v be 3-dimensional vectors, where $u = \begin{pmatrix} 6 \\ 5 \\ 4 \end{pmatrix}$ and $v = \begin{pmatrix} 4 \\ -3 \\ 2 \end{pmatrix}$. What is $u^T v$ (the dot product of u and v)?

Compute $u^T v$ (dot product of u and v):

$$u^T v = \begin{pmatrix} 6 \\ 5 \\ 4 \end{pmatrix}^T \cdot \begin{pmatrix} 4 \\ -3 \\ 2 \end{pmatrix} = (6 \cdot 4) + (5 \cdot -3) + (4 \cdot 2).$$

Simplify:

$$u^T v = 24 - 15 + 8 = 17.$$

Answer: $u^T v = 17$.

Question 4:

Check whether the following vectors are linearly dependent:

$$u = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}, \quad v = \begin{pmatrix} 3 \\ -4 \\ -2 \end{pmatrix}, \quad w = \begin{pmatrix} 5 \\ -10 \\ -8 \end{pmatrix}.$$

Vectors u, v, and w are linearly dependent if there exist constants a, b, and c, not all zero, such that:

$$au + bv + cw = 0.$$

This can be written as a matrix equation:

$$\begin{pmatrix} 2 & 3 & 5 \\ -1 & -4 & -10 \\ 1 & -2 & -8 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

To check for linear dependence, compute the determinant of the coefficient matrix:

$$\text{Matrix} = \begin{pmatrix} 2 & 3 & 5 \\ -1 & -4 & -10 \\ 1 & -2 & -8 \end{pmatrix}.$$

$$\det = 2(12) - 3(18) + 5(6).$$

Simplify:

$$\det = 24 - 54 + 30 = 0.$$

Since the determinant is zero, the vectors u,v,w are **linearly dependent.**

Question 5:

A random experiment involves rolling a six-sided die and flipping a coin. Let X be the outcome of the die roll, and Y be the outcome of the coin flip.

1. Write the sample space for this experiment.
2. Find the joint probability of any single outcome, assuming both the die and the coin are fair.

1. **Sample Space:**
   The sample space for this experiment includes all combinations of the die roll (1 through 6) and the coin flip (Heads or Tails):

   $$S = \{(1, \text{Heads}), (1, \text{Tails}), (2, \text{Heads}), (2, \text{Tails}), \ldots, (6, \text{Heads}), (6, \text{Tails})\}.$$

   There are $6 \times 2 = 12$ total outcomes.

2. **Joint Probability:**
   Since the die and the coin are fair, each outcome has an equal probability. The total number of outcomes is 12, so the probability of any single outcome is:

   $$P(\text{outcome}) = \frac{1}{12}.$$

Question 6:

Consider two continuous random variables X and Y, representing the height (in cm) and weight (in kg) of individuals in a population, respectively.

1. Define the joint PDF $f(x, y)$.
2. Find the probability that a person's height is between 160 cm and 170 cm, and their weight is between 60 kg and 70 kg.

1. **Joint PDF:**
   The joint PDF $f(x, y)$ gives the probability density for individuals with a height x and a weight y. It is used to calculate probabilities over specific ranges of x and y.
2. **Calculate Probability:**
   The probability is calculated using the double integral of $f(x, y)$ over the given range:

$$P(160 \leq X \leq 170, 60 \leq Y \leq 70) = \int_{160}^{170} \int_{60}^{70} f(x, y)\, dy\, dx.$$

   To solve this explicitly, the function $f(x, y)$ must be known. Without specific values for $f(x, y)$, the solution is left as the integral above.

Question 7:

You toss a fair coin three times:

1. **(a)** What is the probability of observing three heads (HHH)?
2. **(b)** What is the probability of observing exactly one head?
3. **(c)** Given that you have observed at least one head, what is the probability that you observe at least two heads?

(a) Probability of Three Heads (HHH):

Each coin toss is independent, and the probability of heads in a single toss is $\frac{1}{2}$. For three tosses, the probability of HHH is:

$$P(HHH) = \left(\frac{1}{2}\right) \times \left(\frac{1}{2}\right) \times \left(\frac{1}{2}\right) = \frac{1}{8}.$$

(b) Probability of Exactly One Head:

The total number of outcomes in three tosses is $2^3 = 8$. The outcomes with exactly one head are:

$$\{HTT, THT, TTH\}.$$

There are 3 such outcomes, and each has a probability of $\frac{1}{8}$. Thus, the probability of exactly one head is:

$$P(\text{Exactly one head}) = 3 \times \frac{1}{8} = \frac{3}{8}.$$

Answer: $\frac{3}{8}$.

(c) Given At Least One Head, Probability of At Least Two Heads:

1. First, find the total outcomes with **at least one head**. These are all outcomes except TTT, so there are:

$$8 - 1 = 7 \quad (\text{favorable outcomes}).$$

2. Next, find the outcomes with **at least two heads**. These are:

$$\{HHT, HTH, THH, HHH\}.$$

There are 4 such outcomes.

3. The conditional probability is:

$$P(\text{At least two heads} \mid \text{At least one head}) = \frac{\text{Favorable outcomes with at least two heads}}{\text{Total outcomes with at least one head}} = \frac{4}{7}.$$

Answer: $\frac{4}{7}$.

Question 8:

Suppose we have two events, A and B, with the following probabilities:

- $P(A) = 0.4$
- $P(B \mid A) = 0.5$
- $P(B \mid A^c) = 0.3$
- $P(A^c) = 0.6$

We want to find the joint probability $P(A \cap B)$.

Using the chain rule, we have:

$$P(A \cap B) = P(A) \cdot P(B \mid A)$$

Substitute the given values:

$$P(A \cap B) = 0.4 \cdot 0.5 = 0.2$$

Question 9:

Suppose we have three events, A, B, and C, with the following probabilities:

- $P(A) = 0.6$
- $P(B \mid A) = 0.7$
- $P(C \mid A \cap B) = 0.5$

We want to find the joint probability $P(A \cap B \cap C)$.

Using the chain rule, we have:

$$P(A \cap B \cap C) = P(A) \cdot P(B \mid A) \cdot P(C \mid A \cap B)$$

Substitute the given values:

$$P(A \cap B \cap C) = 0.6 \cdot 0.7 \cdot 0.5 = 0.21$$

Question 10:

Suppose we have two discrete random variables X and Y with the following joint probability distribution:

X \textbackslash Y	1	2
1	0.2	0.3
2	0.4	0.1

We want to find the marginal probabilities $P(X = 1)$ and $P(Y = 2)$.

To find $P(X = 1)$, we sum the joint probabilities over all values of Y:

$$P(X = 1) = P(X = 1, Y = 1) + P(X = 1, Y = 2) = 0.2 + 0.3 = 0.5$$

To find $P(Y = 2)$, we sum the joint probabilities over all values of X:

$$P(Y = 2) = P(X = 1, Y = 2) + P(X = 2, Y = 2) = 0.3 + 0.1 = 0.4$$

Question 11:

Suppose we want to find the probability of events A and B occurring given event C. We are given:

- $P(C) = 0.8$
- $P(A \cap B \cap C) = 0.4$

We want to find $P(A \cap B \mid C)$.

Using the definition of conditional probability, we have:

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

Substitute the given values:

$$P(A \cap B \mid C) = \frac{0.4}{0.8} = 0.5$$

Question 12:

Suppose we have two events, A and B, with the following probabilities:

- $P(A) = 0.3$
- $P(B) = 0.4$
- $P(B \mid A) = 0.5$

We want to find $P(A \mid B)$ using Bayes' theorem.

Bayes' theorem states:

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

Substitute the given values:

$$P(A \mid B) = \frac{0.5 \cdot 0.3}{0.4} = \frac{0.15}{0.4} = 0.375$$

Question 13:

Let's consider the function:
$f(x) = (x - 3)^2 + 4$

This is a simple parabolic function with a minimum at $x = 3$.

Step-by-Step Gradient Descent

1. **Function and Derivative**:
   - The function is $f(x) = (x - 3)^2 + 4$.
   - The derivative of the function is $f'(x) = 2(x - 3)$.
2. **Initialization**:
   - Choose an initial value for x. Let's start with $x_0 = 0$.
   - Choose a learning rate ϵ. Let's use $\epsilon = 0.1$.
3. **Iteration**:
   We'll perform several iterations to show how x moves towards the minimum.

Iteration 1:

- Current $x = 0$
- Calculate the derivative: $f'(0) = 2(0 - 3) = -6$
- Update x: $x_{\text{new}} = 0 - 0.1 \cdot (-6) = 0 + 0.6 = 0.6$

Iteration 2:

- Current $x = 0.6$
- Calculate the derivative: $f'(0.6) = 2(0.6 - 3) = 2 \cdot (-2.4) = -4.8$
- Update x: $x_{\text{new}} = 0.6 - 0.1 \cdot (-4.8) = 0.6 + 0.48 = 1.08$

Iteration 3:

- Current $x = 1.08$
- Calculate the derivative: $f'(1.08) = 2(1.08 - 3) = 2 \cdot (-1.92) = -3.84$
- Update x: $x_{\text{new}} = 1.08 - 0.1 \cdot (-3.84) = 1.08 + 0.384 = 1.464$

Iteration 4:

- Current $x = 1.464$
- Calculate the derivative: $f'(1.464) = 2(1.464 - 3) = 2 \cdot (-1.536) = -3.072$
- Update x: $x_{\text{new}} = 1.464 - 0.1 \cdot (-3.072) = 1.464 + 0.3072 = 1.7712$

4. **Convergence**:
   - We repeat this process until the change in x becomes very small, indicating that we have reached close to the minimum.

Question 14:

Apply the Gradient-Based Optimization Algorithm:

1. **Steps:**
   - (i) Initialize parameters randomly to θ_0.
   - (ii) Iteratively update the parameters using a suitable rule until convergence.
2. **Problem:**
   Minimize the function:

$$f(x, y, z) = 2x^2 + 3y^2 + z^2 - 10$$

   Start with the initial values:

$$(x_0, y_0, z_0) = (2, -1, 1)$$

   Assumptions:
   - Learning rate $\alpha = 0.05$.

   Question: What will be the updated values of x, y, z after one iteration?

1. **Gradient of the function $f(x, y, z)$:**
   - Partial derivatives:

$$\frac{\partial f}{\partial x} = 4x, \quad \frac{\partial f}{\partial y} = 6y, \quad \frac{\partial f}{\partial z} = 2z$$

   - At $(x_0, y_0, z_0) = (2, -1, 1)$:

$$\frac{\partial f}{\partial x} = 4(2) = 8, \quad \frac{\partial f}{\partial y} = 6(-1) = -6, \quad \frac{\partial f}{\partial z} = 2(1) = 2$$

     Gradient vector:

$$\nabla f = (8, -6, 2)$$

2. **Gradient Descent Update Rule:**
   - General formula:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla f$$

   - Update for each variable:

$$x_1 = x_0 - \alpha \cdot \frac{\partial f}{\partial x}, \quad y_1 = y_0 - \alpha \cdot \frac{\partial f}{\partial y}, \quad z_1 = z_0 - \alpha \cdot \frac{\partial f}{\partial z}$$

3. **Calculate the updates:**
   - $x_1 = 2 - 0.05 \cdot 8 = 2 - 0.4 = 1.6$
   - $y_1 = -1 - 0.05 \cdot (-6) = -1 + 0.3 = -0.7$
   - $z_1 = 1 - 0.05 \cdot 2 = 1 - 0.1 = 0.9$

Updated Values:

$$(x_1, y_1, z_1) = (1.6, -0.7, 0.9)$$

1. **XOR Definition:**
   - Exclusive OR (XOR) is a binary operation.
   - Outputs `true` only when inputs differ.
   - Truth Table:
     - (0, 0) → 0
     - (0, 1) → 1
     - (1, 0) → 1
     - (1, 1) → 0
2. **Linear Separability:**
   - XOR is **not linearly separable**; it cannot be divided by a straight line in a 2D space.
   - Single-layer perceptrons are unsuitable for XOR as they cannot handle non-linear problems.

Deep Feedforward Network for XOR

1. **Architecture:**
   - One hidden layer with **two neurons**.
   - Uses **non-linear activation functions** (e.g., ReLU) to introduce non-linearity.
2. **Mathematical Representation:**
   - Input-to-hidden transformation:

$$h = g(W^T x + c)$$

     - W: Weight matrix.
     - c: Bias vector.
     - g: Activation function (ReLU in this case).
   - Hidden-to-output transformation:

$$y = f(h; w, b) = w^T h + b$$

     - w: Weight vector.
     - b: Output layer bias.

- Overall function:

$$f(x; W, c, w, b) = w^T \text{ReLU}(W^T x + c) + b$$

Activation Function

- ReLU (Rectified Linear Unit):

$$g(z) = \max(0, z)$$

- Introduces non-linearity, crucial for solving the XOR problem.

Step-by-Step Example

1. **Weights and Biases:**
   - $W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
   - $c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$
   - $w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

2. **Input Matrix:**

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

3. **Layer 1: Linear Transformation:**

$$XW = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

4. **Add Bias Vector:**

$$XW + c = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

5. **Apply ReLU:**

$$h = \text{ReLU}(XW + c) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

6. **Layer 2: Output Computation:**

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

7. **Result:**
   - Matches the XOR truth table output.

Loss Function

- Mean Squared Error (MSE):

$$J(\theta) = \frac{1}{4} \sum_{x \in X} (f^*(x) - f(x; \theta))^2$$

  - $f^*(x)$: Target function.
  - $f(x; \theta)$: Model output.

Training Insights

- **Gradient Descent:**
  - Used to minimize the loss and find optimal weights and biases.
  - Ensures convergence to the global minimum of the loss function.