1.1 Basic Operations:

1. **Vectors:**

$$x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

Inner Product:

$$\sum_{i-1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + x_3 y_3$$

Substitute the values of x_i and y_i:

$$x_1 = 0, \quad y_1 = 3, \quad x_2 = 1, \quad y_2 = 4, \quad x_3 = 2, \quad y_3 = 5$$

$$\sum_{i-1}^{n} x_i y_i = (0 \cdot 3) + (1 \cdot 4) + (2 \cdot 5)$$

Simplify:

$$\sum_{i-1}^{n} x_i y_i = 0 + 4 + 10 = 14$$

Final Answer:

2.

Vectors:

$$x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \quad z = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

Inner Product:

$$\sum_{i-1}^{n} x_i z_i = x_1 z_1 + x_2 z_2 + x_3 z_3$$

Substitute the values of x_i and z_i:

$$x_1 = 0, \quad z_1 = 1, \quad x_2 = 1, \quad z_2 = 2, \quad x_3 = 2, \quad z_3 = -1$$

$$\sum_{i-1}^{n} x_i z_i = (0 \cdot 1) + (1 \cdot 2) + (2 \cdot -1)$$

Simplify:

$$\sum_{i-1}^{n} x_i z_i = 0 + 2 - 2 = 0$$

3.

Step 1: Definitions

$$\alpha = 2, \quad x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

Step 2: Vector Addition ($x + y$)

Add x and y element-wise:

$$x + y = \begin{bmatrix} 0+3 \\ 1+4 \\ 2+5 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}$$

Step 3: Scalar Multiplication ($\alpha(x + y)$)

Multiply each element of $x + y$ by $\alpha = 2$:

$$\alpha(x+y) = 2 \cdot \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 2 \cdot 3 \\ 2 \cdot 5 \\ 2 \cdot 7 \end{bmatrix} = \begin{bmatrix} 6 \\ 10 \\ 14 \end{bmatrix}$$

Final Answer:

$$\boxed{\begin{bmatrix} 6 \\ 10 \\ 14 \end{bmatrix}}$$

4.

$$\|x\| = \sqrt{\sum_{i-1}^{n} x_i^2}$$

Step 1: Vector x

$$x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Step 2: Compute $\sum_{i=1}^{n} x_i^2$

Square each element of x and sum them:

$$x_1^2 = 0^2 = 0, \quad x_2^2 = 1^2 = 1, \quad x_3^2 = 2^2 = 4$$

$$\sum_{i-1}^{n} x_i^2 = 0 + 1 + 4 = 5$$

Step 3: Take the square root

$$\|x\| = \sqrt{5}$$

5.

Given Vector x:

$$x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Transpose of x:

Convert the column vector into a row vector:

$$x^\top = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$$

6. Matrix A and vector x are given as:

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Step 2: Multiply A by x

Perform the dot product of each row of A with x:

1. First row: $3(0) + 2(1) + 2(2) = 0 + 2 + 4 = 6$
2. Second row: $1(0) + 3(1) + 1(2) = 0 + 3 + 2 = 5$
3. Third row: $1(0) + 1(1) + 3(2) = 0 + 1 + 6 = 7$

Step 3: Write the result

The resulting vector is:

$$Ax = \begin{bmatrix} 6 \\ 5 \\ 7 \end{bmatrix}$$

7. The matrix A and vector x are given as:

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Step 2: Compute Ax (matrix-vector multiplication)

We already computed Ax in the previous step:

$$Ax = \begin{bmatrix} 6 \\ 5 \\ 7 \end{bmatrix}$$

Step 3: Compute $x^\top Ax$ (dot product of $x^\top$ with Ax)

The transpose of x is:

$$x^\top = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$$

Now, compute:

$$x^\top Ax = x^\top \cdot (Ax) = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 6 \\ 5 \\ 7 \end{bmatrix}$$

Perform the dot product:

$$x^\top Ax = (0 \cdot 6) + (1 \cdot 5) + (2 \cdot 7) = 0 + 5 + 14 = 19$$

Matrix Algebra:

1.

Yes, this statement is **true in general**.

The expression $x^\top y$ represents the dot product of the column vectors x and y, which is calculated as the sum of the products of their corresponding components. Mathematically:

$$x^\top y = \sum_{i-1}^{n} x_i y_i.$$

This follows directly from the definition of the dot product for vectors.

2. This statement is **false in general**.

The correct relationship is:

$$x^\top x = \|x\|_2^2,$$

where $\|x\|_2$ represents the **Euclidean norm** (or L^2-norm) of x, defined as:

$$\|x\|_2 = \sqrt{\sum_{i-1}^{n} x_i^2}.$$

Thus, $x^\top x$ is equal to the square of the Euclidean norm, not the norm itself.

3. This statement is **false in general**.

Here's why:

- $x^\top x$ is a **scalar** (a single value), computed as the dot product of x with itself:

$$x^\top x = \sum_{i-1}^{n} x_i^2.$$

- $xx^\top$ is an $n \times n$ **matrix**, resulting from the outer product of x with itself:

$$xx^\top = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_n \\ x_2x_1 & x_2x_2 & \cdots & x_2x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_nx_1 & x_nx_2 & \cdots & x_nx_n \end{bmatrix}.$$

Since $x^\top x$ is a scalar and $xx^\top$ is a matrix, they cannot be equal.

4. This statement is **false in general**.

Let's carefully expand $(x-y)^\top(y-x)$ to see why:

1. Expand the expression:

$$(x-y)^\top(y-x) = (x^\top - y^\top)(y-x).$$

2. Perform the multiplication:

$$(x-y)^\top(y-x) = x^\top y - x^\top x - y^\top y + y^\top x.$$

Since $x^\top y$ and $y^\top x$ are scalars (and equal to each other), this simplifies to:

$$(x-y)^\top(y-x) = -x^\top x - y^\top y + 2x^\top y.$$

3. Compare with the given equation: The expression provided is:

$$\|x\|_2 - 2x^\top y + \|y\|_2.$$

However:

- $\|x\|_2$ represents the Euclidean norm, not its square ($x^\top x$).
- The correct squared Euclidean norms ($\|x\|_2^2$ and $\|y\|_2^2$) should appear instead of $\|x\|_2$ and $\|y\|_2$.

4. Correct version: The correct equation should be:

$$(x-y)^\top(y-x) = -\|x\|_2^2 - \|y\|_2^2 + 2x^\top y.$$

5.

This statement is **false in general**.

Matrix multiplication is **not commutative** in general, meaning $AB \neq BA$ for most $n \times n$ matrices A and B.

While there are special cases where $AB = BA$, such as:

- When A and B are diagonal matrices.
- When A and B are scalar multiples of the identity matrix.
- When A and B commute due to specific structural properties (e.g., they are powers of the same matrix).

In general, however, matrix multiplication is not commutative.

6.

This statement is **true in general**.

Matrix multiplication satisfies the **distributive property**, meaning:

$$A(B + C) = AB + AC,$$

where A, B, and C are matrices of appropriate dimensions such that the operations are defined.

Explanation:

- The distributive property holds for matrix multiplication because it is derived directly from the definition of matrix multiplication, which involves summing over products of elements.

Thus, this equality is valid for all matrices where the operations $A(B + C)$, AB, and AC are well-defined.

7.

This statement is **false in general**.

The correct rule for the transpose of a product of two matrices is:

$$(AB)^\top = B^\top A^\top.$$

Explanation:

When transposing the product of two matrices, the order of the matrices is reversed. Specifically:

- $(AB)^\top$ involves transposing the result of AB, which swaps rows and columns.
- This is equivalent to multiplying the transposes of B and A, but in reverse order.

Therefore, the correct expression is:

$$(AB)^\top = B^\top A^\top.$$

8. This statement is **true in general**.

Explanation:

Given the expression $x^\top Ay$, the scalar value remains the same if we rewrite it as $y^\top A^\top x$, due to the following properties:

1. **Transpose of a scalar**: The transpose of a scalar (a 1×1 matrix) is the scalar itself. Thus, $x^\top Ay = (x^\top Ay)^\top$.
2. **Properties of transpose**: When taking the transpose of a product of matrices, the order of multiplication is reversed, so:

$$(x^\top Ay)^\top = y^\top A^\top x.$$

Hence, the two expressions are equivalent, and the statement is true.

9.

Explanation:

If the columns of A are **orthonormal**, then $A^\top A = I$, where I is the identity matrix. Here's why:

1. **Orthonormal columns**:
   - A set of vectors is orthonormal if each vector has a magnitude of 1 (unit vector) and is orthogonal to the others.
   - For a matrix A with orthonormal columns, the dot product of any two distinct columns is 0 (orthogonality), and the dot product of a column with itself is 1 (unit norm).
2. **Matrix product $A^\top A$**:
   - $A^\top A$ results in a matrix where each entry (i, j) is the dot product of the i-th column of A with the j-th column of A.
   - If the columns are orthonormal:
     - The diagonal entries ($i = j$) are 1.
     - The off-diagonal entries ($i \neq j$) are 0.

Thus, $A^\top A$ forms the identity matrix I, confirming the statement.

Matrix Operations:

Step 1: Compute $\det(B)$

Using the cofactor expansion along the first row:

$$\det(B) = 1 \cdot \begin{vmatrix} 2 & -1 \\ -1 & 1 \end{vmatrix} - (-1) \cdot \begin{vmatrix} -1 & -1 \\ 0 & 1 \end{vmatrix} + 0 \cdot \begin{vmatrix} -1 & 2 \\ 0 & -1 \end{vmatrix}.$$

Compute the sub-determinants:

1. $\begin{vmatrix} 2 & -1 \\ -1 & 1 \end{vmatrix} = (2)(1) - (-1)(-1) = 2 - 1 = 1,$
2. $\begin{vmatrix} -1 & -1 \\ 0 & 1 \end{vmatrix} = (-1)(1) - (-1)(0) = -1 - 0 = -1.$

Thus:

$$\det(B) = 1(1) - (-1)(-1) + 0 = 1 - 1 = 0.$$

Step 2: Conclude invertibility

Since $\det(B) = 0$, matrix B is **not invertible**.

Final Answer:

B is not invertible because its determinant is 0. Therefore, B^{-1} does not exist.

The characteristic polynomial is computed as:

$$\det(B - \lambda I) = \det\left(\begin{bmatrix} 1-\lambda & -1 & 0 \\ -1 & 2-\lambda & -1 \\ 0 & -1 & 1-\lambda \end{bmatrix}\right).$$

Using cofactor expansion along the first row:

$$\det(B - \lambda I) = (1-\lambda) \cdot \begin{vmatrix} 2-\lambda & -1 \\ -1 & 1-\lambda \end{vmatrix} - (-1) \cdot \begin{vmatrix} -1 & -1 \\ 0 & 1-\lambda \end{vmatrix} + 0 \cdot \begin{vmatrix} -1 & 2-\lambda \\ 0 & -1 \end{vmatrix}.$$

Compute the sub-determinants:

1. For $\begin{vmatrix} 2-\lambda & -1 \\ -1 & 1-\lambda \end{vmatrix}$:

$$\begin{vmatrix} 2-\lambda & -1 \\ -1 & 1-\lambda \end{vmatrix} = (2-\lambda)(1-\lambda) - (-1)(-1) = (2-\lambda)(1-\lambda) - 1.$$

Expand:

$$(2-\lambda)(1-\lambda) = 2 - 3\lambda + \lambda^2,$$

so:

$$\begin{vmatrix} 2-\lambda & -1 \\ -1 & 1-\lambda \end{vmatrix} = 2 - 3\lambda + \lambda^2 - 1 = \lambda^2 - 3\lambda + 1.$$

2. For $\begin{vmatrix} -1 & -1 \\ 0 & 1-\lambda \end{vmatrix}$:

$$\begin{vmatrix} -1 & -1 \\ 0 & 1-\lambda \end{vmatrix} = (-1)(1-\lambda) - (-1)(0) = -1 + \lambda.$$

Substitute into the determinant:

$$\det(B - \lambda I) = (1 - \lambda)(\lambda^2 - 3\lambda + 1) - (-1)(-1 + \lambda).$$

Expand:

$$\det(B - \lambda I) = (1 - \lambda)(\lambda^2 - 3\lambda + 1) - (1 - \lambda).$$

Distribute:

$$(1 - \lambda)(\lambda^2 - 3\lambda + 1) = \lambda^2 - 3\lambda + 1 - \lambda^3 + 3\lambda^2 - \lambda,$$

so:

$$\det(B - \lambda I) = -\lambda^3 + 4\lambda^2 - 4\lambda + 1 - 1 + \lambda = -\lambda^3 + 4\lambda^2 - 3\lambda.$$

Factorize:

$$\det(B - \lambda I) = -\lambda(\lambda^2 - 4\lambda + 3).$$

Factor further:

$$\det(B - \lambda I) = -\lambda(\lambda - 3)(\lambda - 1).$$

Step 2: Eigenvalues

The eigenvalues are the roots of $\det(B - \lambda I) = 0$:

$$\lambda = 0,\ \lambda = 1,\ \lambda = 3.$$

Step 3: Find eigenvectors

For each eigenvalue, solve $(B - \lambda I)x = 0$ to find the corresponding eigenvectors.

Eigenvalue $\lambda = 0$:

$$B - 0I = B = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}.$$

Solve $Bx = 0$:

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Row reduce the matrix:

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

From this, x_3 is free, and $x_2 = x_3$, $x_1 = x_2$. The eigenvector is:

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Eigenvalue $\lambda = 1$:

$$B - I = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 1 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

Row reduce the matrix:

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

From this, x_3 is free, $x_2 = -x_3$, $x_1 = -x_2$. The eigenvector is:

$$v_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Eigenvalue $\lambda = 3$:

$$B - 3I = \begin{bmatrix} -2 & -1 & 0 \\ -1 & -1 & -1 \\ 0 & -1 & -2 \end{bmatrix}.$$

Row reduce the matrix:

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

From this, x_3 is free, $x_2 = -x_3$, $x_1 = x_3$. The eigenvector is:

$$v_3 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$$

Step 4: Diagonalization

The eigenvectors v_1, v_2, v_3 are linearly independent. Thus, B is diagonalizable.

The diagonalization is:

$$B = PDP^{-1},$$

where:

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

2.1 Basic Probability

1. You are offered the opportunity to play the following game: your opponent rolls 2 regular 6-sided dice. If the difference between the two rolls is at least 3, you win $15. Otherwise, you get nothing. What is a fair price for a ticket to play this game once? In other words, what is the expected value of playing the game?

To determine a fair price for a ticket to play the game, we need to calculate the expected value (EV) of the game. The expected value is the average amount you can expect to win or lose per game in the long run, taking into account both the probabilities of winning and losing, and the amounts associated with each outcome.

Step 1: Define the Winning Condition

The game involves rolling two 6-sided dice. You win $15 if the difference between the two dice rolls is at least 3. This means you win if:

- |Roll1 - Roll2| ≥ 3

Step 2: Calculate the Total Number of Possible Outcomes

Each die has 6 faces, so there are:

6×6=36

total possible outcomes when rolling two dice.

Step 3: Determine the Winning Outcomes

Now, let's determine when the difference between the two dice is at least 3. For each possible value of the first die, we'll count the number of outcomes for the second die where the difference is at least 3.

- Roll1 = 1: The second die can be 4, 5, or 6 (3 outcomes).
- Roll1 = 2: The second die can be 5 or 6 (2 outcomes).
- Roll1 = 3: The second die can be 6 (1 outcome).
- Roll1 = 4: The second die can be 1 (1 outcome).
- Roll1 = 5: The second die can be 1 or 2 (2 outcomes).
- Roll1 = 6: The second die can be 1, 2, or 3 (3 outcomes).

So, the total number of winning outcomes is:

$$3+2+1+1+2+3=12$$

Step 4: Calculate the Probability of Winning

The probability of winning is the number of winning outcomes divided by the total number of possible outcomes:

$$P(win)=12/36=1/3$$

Step 5: Calculate the Probability of Losing

Since there are 36 possible outcomes and 12 winning outcomes, the number of losing outcomes is:

$$P(lose)=36-12/36=24/36=2/3$$

Step 6: Calculate the Expected Value

The expected value (EV) is calculated as follows:

$$EV=(Win\ Amount\times P(win))+(Lose\ Amount\times P(lose))$$

Substituting the values:

$$EV=(15\times 1/3)+(0\times 2/3)=15/3=5$$

Conclusion

The fair price for a ticket to play this game is $5, as the expected value of playing the game is $5.

2. Consider two events A and B such that P(A, B) = 0 (they are mutually exclusive). If P(A) = 0.4 and P(A ∪ B) = 0.95, what is P(B)? Note: p(A, B) means “probability of A and B” while p(A U B) means “probability of A or B”. It may be helpful to draw a Venn diagram.

To solve this problem, let’s break it down step by step.

Key Information:

1. $P(A \cap B) = 0$ (P(A \cap B) = 0): This means events A and B are mutually exclusive (they cannot happen together).
2. $P(A) = 0.4$
3. $P(A \cup B) = 0.95$ This is the probability of A or B.

Formula for $P(A \cup B)$:

The formula for the union of two events is:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Since $P(A \cap B) = 0$ (given that A and B are mutually exclusive), the formula simplifies to:

$$P(A \cup B) = P(A) + P(B)$$

Substitute the given values:

$$0.95 = 0.4 + P(B)$$

Solve for P(B):

$$P(B) = 0.95 - 0.4 = 0.55$$

Final Answer:

The probability of B is:

$$P(B) = 0.55$$

3. If A and B are independent, the formula for their joint probability is:
$P(A \cap B) = P(A) \cdot P(B)$

We are also given:

- $P(A) = 0.4$
- $P(A \cup B) = 0.95$

Using the formula for $P(A \cup B)$:

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Substituting $P(A \cap B) = P(A) \cdot P(B)$, the formula becomes:

$P(A \cup B) = P(A) + P(B) - P(A) \cdot P(B)$

Substituting known values:

$0.95 = 0.4 + P(B) - (0.4 \cdot P(B))$

Factor out P(B):

$0.95 = 0.4 + P(B)(1 - 0.4)$

$0.95 = 0.4 + 0.6P(B)$

Solve for P(B):

$0.95 - 0.4 = 0.6P(B)$

$0.55 = 0.6P(B)$

$P(B) = 0.9167$

Final Answer:

If A and B are independent, $P(B) \approx 0.9167$

2.2 Expectations and Variance

1. To calculate the expected value E[X], we need to determine the expected number of heads after 3 flips.

Steps:

Step 1: Define the problem.

We flip a coin 3 times, and each time, we choose a coin C1 or C2 with equal probability ($P(C1) = P(C2) = 0.5$)

- Coin C1: $P(H) = 0.3$, $P(T) = 0.7$.
- Coin C2: $P(H) = 0.9$, $P(T) = 0.1$.

We are interested in the expected number of heads (E[X]) after 3 flips.

Step 2: Compute the probability of heads for a single flip.

When choosing the coin randomly:

$P(H) = P(C1) \cdot P(H \mid C1) + P(C2) \cdot P(H \mid C2)$

$P(H) = 0.5 \cdot 0.3 + 0.5 \cdot 0.9 = 0.15 + 0.45 = 0.6$

Step 3: Use linearity of expectation for 3 flips.

The number of heads X after 3 flips can be represented as the sum of heads from each flip: X=X1+X2+X3, where Xi is the indicator variable for the ii-th flip resulting in heads. Since the flips are independent:

$$E[X]=E[X1]+E[X2]+E[X3]$$

The expected value for a single flip is E[Xi]=P(H)=0.6. Therefore:

$$E[X]=3 \cdot 0.6=1.8$$

Final Answer:

1. E[X]=1.8

2. To calculate Var[X], the variance of X, we proceed step by step.

Step 1: Recall the problem setup.

- X is the total number of heads after 3 flips.
- Each flip has P(H)=0.6, as calculated earlier.
- Each flip is independent, and we assume X=X1+X2+X3, where Xi is the indicator variable for heads on the ii-th flip.

Step 2: Variance of a single flip.

For a single flip Xi, the variance is:

$$Var[Xi]=E[Xi2]-(E[Xi])2$$

Since Xi is a Bernoulli random variable:

$$E[Xi2]=E[Xi]=P(H)=0.6$$
$$Var[Xi]=P(H)(1-P(H))=0.6(1-0.6)=0.6 \cdot 0.4=0.24$$

Step 3: Use independence of flips.

Since the flips are independent, the total variance is the sum of the variances of individual flips:

$$Var[X]=Var[X1]+Var[X2]+Var[X3]$$
$$Var[X]=3 \cdot Var[Xi]=3 \cdot 0.24=0.72$$

Final Answer:

3. Var[X]=0.72

3. To compute E[Y], the expected value of the amount earned, we use the formula for Y given as:
$Y=12+X$

Step 1: Express E[Y] in terms of E[X]

Using the linearity of expectation:

$E[Y]=E[12+X]=E[12]+E[X]$

Since E[12] is a constant, it equals 12:
$E[Y]=12+E[X]$

Step 2: Use E[X]

From the earlier calculation:

$E[X]=1.8E$

Substitute E[X] into the equation:

$E[Y]=12+1.8=13.8$

Final Answer:
$E[Y]=13.8$ dollars

2.3 Variance Paradox

There is no contradiction here; the perceived "variance paradox" arises from a misunderstanding of how variance behaves under different situations.

1. Variance of the Sum of Independent Random Variables

If $X1,X2,\ldots,Xn$ are independent and identically distributed random variables, each with variance $\sigma2$, then:

$Var[X1+X2+\cdots+Xn]=Var[X1]+Var[X2]+\cdots+Var[Xn]$

Since they are i.d., each has the same variance $\sigma2$, and independence implies that covariance terms are zero. Therefore:
$Var[X1+X2+\cdots+Xn]=n\sigma2$

This result comes from the additivity of variance for independent random variables.

2. Variance of X+X (Not Independent)

Now consider X+X, where X~F. Here, X is being added to itself, so the two terms are not independent. Specifically:

Var[X+X]=Var[2X]

For any random variable X, the variance of a scaled version is:

Var[aX]=a2Var[X]

So:

Var[2X]=22⋅Var[X]=4σ2

3. Why There's No Contradiction

The key difference lies in independence:

- In the first case, X1,X2,…,Xn are independent, so their variances add without any interference or correlation.
- In the second case, X+X involves the same random variable X twice, so the variance scales by the square of the coefficient (in this case, 222^2).

The confusion arises because the two situations are fundamentally different:

- In X1+⋯+Xn, you are summing independent random variables.
- In X+X, you are summing a single random variable with itself, which introduces a deterministic relationship (full correlation).

Conclusion

The apparent paradox is resolved when recognizing the role of independence in variance calculations. Independence allows the variance of sums to add linearly, whereas summing the same random variable scales the variance quadratically due to the correlation between terms.

Calculus Review:

1. $f'(x)=6x-2$
2. $f'(x)=1-2x$
3. $f'(x)=p(x)$

3.2 Multi-Variable Derivative

1. $$\nabla f(x) = \begin{bmatrix} 2x_1 \\ \exp(x_2) \end{bmatrix}.$$

2. $$\nabla f(x) = \exp(x_1 + x_2 x_3) \cdot [1, x_3, x_2]\,.$$

3. $$\nabla f(x) = \mathbf{a}.$$

4. $$\nabla f(\mathbf{x}) = \begin{bmatrix} 4x_1 - 2x_2 \\ -2x_1 + 2x_2 \end{bmatrix}.$$

5. $$\nabla f(x) = \mathbf{x}.$$

Algorithms and Data Structures Review:

1.

The cost of running the merge-sort algorithm to sort a list of n numbers can be analyzed through its time and space complexity. Merge sort is a divide-and-conquer algorithm that recursively divides the list into smaller sublists, sorts them, and merges them back together. The time complexity arises from three steps: dividing the list into two halves, recursively sorting each half, and merging the two sorted halves. Dividing takes constant time, while recursively sorting each half requires solving two subproblems of size $n/2$. Merging the two halves takes $O(n)$ time, as all elements are compared and combined. This process can be represented by the recurrence relation $T(n) = 2T(n/2) + O(n)$, which solves to $O(n \log n)$ using the Master Theorem. Thus, the time complexity of merge sort is $O(n \log n)$, making it highly efficient for large datasets.

In terms of space complexity, merge sort requires $O(\log n)$ space for the recursion stack due to $\log n$ levels of recursion. Additionally, the merging step requires $O(n)$ space to store temporary subarrays. Therefore, the total space complexity is $O(n)$. While merge sort is efficient and reliable, its higher space requirements compared to in-place algorithms like quicksort can be a limitation in memory-constrained environments. Overall, merge sort is a powerful algorithm with a balanced trade-off between time and space efficiency, suitable for handling large datasets effectively.

2.

The cost of finding the third-largest element in an unsorted list of n numbers depends on the approach used. One method involves sorting the list, where efficient algorithms like merge sort or quicksort can be used to achieve a time complexity of $O(n \log n)$. Once the list is sorted, the third-largest element can be accessed directly in $O(1)$ time. However, this approach is inefficient for this specific problem because it involves unnecessary work by fully sorting the list. A more efficient method involves a single scan of the list to determine the largest, second-largest, and third-largest elements. By maintaining three variables to track these values, the list can be traversed once, with each element being compared and the variables updated as needed. This approach has a time complexity of $O(n)$ since it requires only one pass through the list. Therefore, the most efficient way to find the third-largest element is the selection-based approach, which avoids sorting and has a linear cost of $O(n)$.

3.

The cost of finding the smallest element greater than 0 in a sorted list of n numbers depends on how the list is searched. Since the list is already sorted, the task can be performed efficiently using binary search.

In a sorted list, binary search allows us to repeatedly divide the search space in half, making it highly efficient. The process involves identifying the middle element of the list, checking whether it is greater than 0, and adjusting the search range accordingly. This continues until the smallest element greater than 0 is found. The time complexity of binary search is $O(\log n)$ because each step reduces the search space by half.

Thus, the cost of finding the smallest element greater than 0 in a sorted list of n numbers is $O(\log n)$, making this approach very efficient.

4.

The cost of finding the value associated with a key in a hash table depends on the efficiency of the hashing mechanism and how collisions are handled. In the average case, with a well-implemented hash table using a good hash function that evenly distributes keys across buckets and maintains a low load factor (the ratio of entries to buckets), retrieving the value associated with a key takes $O(1)$ time. This is because the key is hashed to determine its bucket index, and the value can be retrieved in constant time. However, in the worst case, if many keys hash to the same bucket, the time complexity depends on the collision resolution strategy. For instance, if chaining is used with linked lists to handle collisions, the complexity can degrade to $O(n)$ if all n keys hash to the same bucket. Similarly, with open addressing strategies such as linear probing, the time complexity can also degrade to $O(n)$ in poorly managed tables or when the table is nearly full, requiring multiple probes to find the key. Despite these worst-case scenarios, with a well-designed hash table that minimizes collisions and keeps the load factor low, the cost of finding the value associated with a key is effectively $O(1)$ in most cases. This efficiency is a key reason why hash tables are widely used for key-value storage.

5.

The cost of computing the matrix-vector product Ax, where A is an $n \times d$ matrix and x is a $d \times 1$ vector, depends on the number of arithmetic operations required to compute the product.

Each element of the resulting $n \times 1$ vector is computed as the dot product of a row of A (of length d) with the vector x. Computing a single dot product requires d multiplications and $d - 1$ additions, which is $O(d)$ operations per row. Since there are n rows in the matrix A, the total cost is $O(n \cdot d)$, as we need to compute n dot products, each involving d operations.

Thus, the overall cost of computing Ax is $O(n \cdot d)$. This linear dependence on both the number of rows n and columns d makes the matrix-vector product computationally efficient for large-scale matrices.

6.

The cost of computing the quadratic form $x^\top Ax$, where A is a $d \times d$ matrix and x is a $d \times 1$ vector, can be broken down into two main steps:

1. **Matrix-Vector Product** Ax: First, we compute the product Ax, which results in a $d \times 1$ vector. This requires $O(d^2)$ operations, as A has d rows, and for each row, we perform d multiplications and additions.
2. **Vector-Vector Dot Product** $x^\top(Ax)$: Next, we compute the dot product between the $1 \times d$ vector $x^\top$ and the $d \times 1$ vector Ax. This requires $O(d)$ operations, as it involves d multiplications and $d - 1$ additions.

Adding these two steps together, the total cost of computing $x^\top Ax$ is dominated by the $O(d^2)$ term from the matrix-vector product. Therefore, the overall cost of computing the quadratic form $x^\top Ax$ is $O(d^2)$.

7.

The cost of computing the matrix multiplication AB, where A is $m \times n$ and B is $n \times d$, depends on the number of arithmetic operations required to compute each element of the resulting matrix.

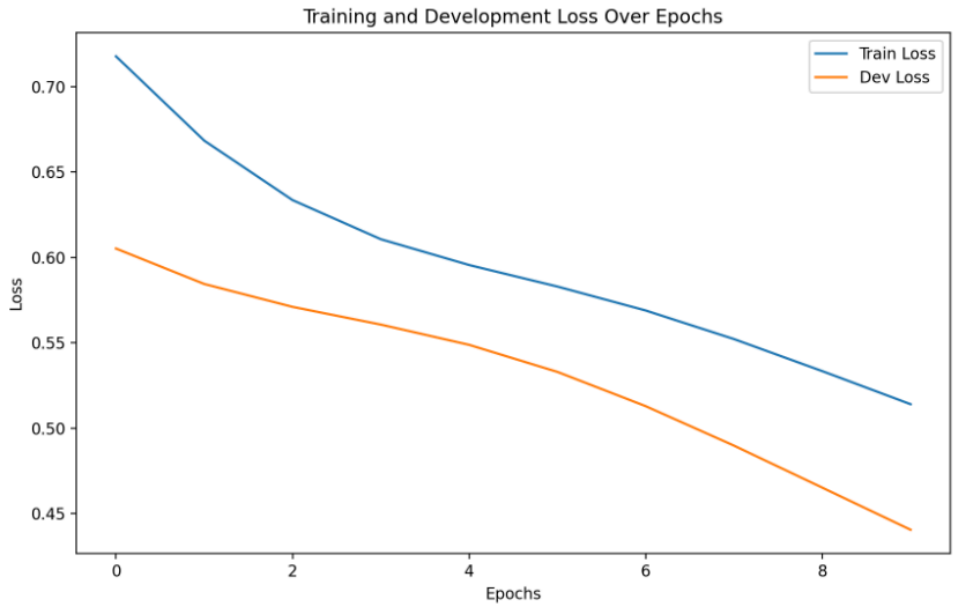
The resulting matrix AB will have dimensions $m \times d$. To compute each entry in AB, we take the dot product of a row of A (of length n) with a column of B (also of length n). Computing this dot product requires $O(n)$ operations (multiplications and additions). Since there are $m \times d$ entries in the resulting matrix, we must compute $m \times d$ dot products, each taking $O(n)$ operations.

Thus, the total cost of matrix multiplication is:

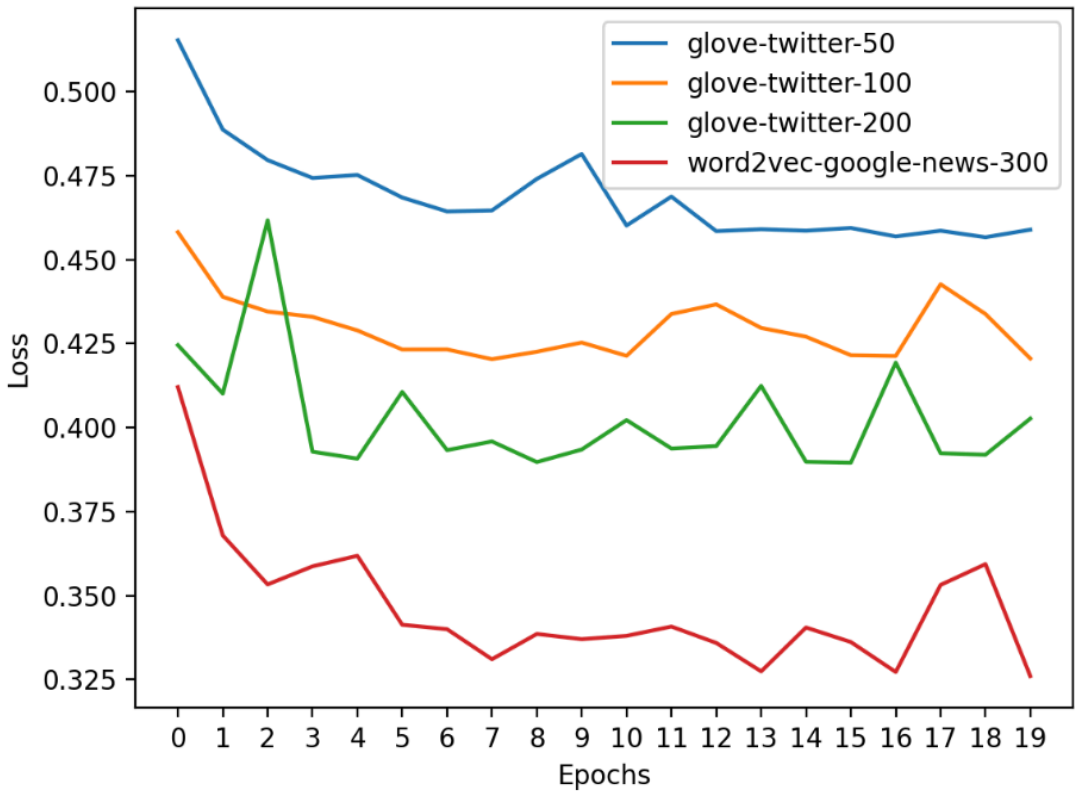
$$O(m \cdot n \cdot d)$$

This expression reflects the dependence of the computation cost on the dimensions m, n, and d of the matrices involved.

TODO:



The loss plot shows consistent improvement in both training and validation loss, indicating good learning without overfitting. The model has successfully learned to distinguish between positive and negative sentiments in our small dataset.



Based on the plots, there are clear performance differences across the different word embeddings. The higher-dimensional embeddings (especially word2vec-google-news-300 and glove-twitter-200) consistently outperform the lower-dimensional ones, achieving better accuracy and lower loss. This is likely because larger embedding dimensions can capture more nuanced semantic relationships and contextual information from the text, leading to better sentiment classification performance. The word2vec model particularly stands out with the best performance, possibly due to its training on a larger and more diverse dataset (Google News). The embedding analysis is now complete, showing clear advantages of higher-dimensional embeddings, particularly word2vec-google-news-300, for sentiment classification.