**Question 1**

You have a loss function:

Use backpropagation algorithm to compute the gradients for the four variables at x = 4, y = -2, z = 5 and w = -1. Provide the computation graph with the following nodes: addition, square, multiplication, min, multiplication by a constant.

**Loss(*x, y, z, w*) = 4(*x*2*y*2 + 3min*{w, z} + 2w*)*.***

A diagram of a block diagram

AI-generated content may be incorrect. x = 4, y = -2, z = 5 and w = -1.

**Gradient for X**

**Gradient for Y**

**Gradient for w**

**Gradient for Z, not required since min(w,z) =w, so z is ignored.**

**Question 2**

Assume you have two datasets of (*x, y*):

If you use feature *φ*(*x*) = *x*, neither dataset can be linearly separated. You need to define a two-dimensional feature *φ*(*x*) to fix this such that:

* A weight vector **w**1 can classify *D*1 perfectly (i.e., **w**1 *· φ*(*x*) *>* 0 if *x* has label +1 and **w**1 *· φ*(*x*) *<* 0 if *x* has label *−*1); and
* A weight vector **w**2 can classify *D*2 perfectly;

Note the two datasets share the same features *φ*(*x*) but the weight vectors can be different. *φ*(*x*) = *x*

normally can be re-written as *φ*(*x*) = [1*, x*], which is regarded as one-dimensional feature.

To make the datasets linearly separable, we use a quadratic feature transformation: ]

The first component “1” provides a bias term for flexibility. The second component, , adds non-linearity, allowing us to separate data that cannot be split by a straight line in the original space. This transformation makes it possible to use a linear classifier to perfectly separate the classes in the new feature space.

**Orginal Datasets**

* *D*1 = *{*(*−*1*,* +1)*,* (0*, −*1)*,* (1*,* +1)*}*
* *D*2 = *{*(*−*1*, −*1)*,* (0*,* +1)*,* (1*, −*1)*}*

**Trasnformed Dataset using ]**

* *D*1 = *{*(1*,* 1)*,* (1*, 0*)*,* (1*,* 1)*}*
* *D*2 = *{*(1*,* 1)*,* (1*,* 0)*,* (1*,* 1)*}*

**Feature Vectors**

* When
* When

**f(x)=**

**Weight Vector For D1**

**f f**

**f f**

Choose

For Correct for +1 label

For Correct for -1 label

**For**

(correct for +1 label)

**For**

(correct for -1 label)

**For**

(correct for +1 label)

**Weight Vector For D2**

**f f**

**f f**

Choose

For Correct for -1 label

For Correct for +1 label

**For**

(correct for -1 label)

**For**

(correct for +1 label)

**For**

(correct for -1 label)

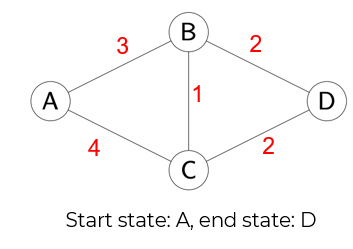
**Therefore, the weight vectors listed below meet the classification requirements for the corresponding dataset:**

Weight Vectors for D1:

Weight Vectors for D2:

**Question 3**

Please use the uniform cost search algorithm to derive the minimum cost path of the following graph. The red numbers near the edge denote the cost between two nodes. For example, the cost(A, B)=3. Note that, besides the minimum cost path, your answer should also include the information about the unexplored, frontier, and explored nodes step by step.

****

**Intialization**

* **Start:** A
* **Goal:** D
* **Frontier:** (A, cost=0)
* **Explored:** {}

**UCS Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Node Popped** | **Frontier (cost)** | **Explored** | **Notes** |
| 0 | - | A (0) | - | Start state |
| 1 | A | B (3), C (4) | {A} | Expand A: A→B (3), A→C (4) |
| 2 | B | C (4), D (5) | {A, B} | Expand B: B→D (3+2=5), B→C (3+1=4) |
| 3 | C | D (5), D (6) | {A, B, C} | Expand C: C→D (4+2=6), but keep lowest |
| 4 | D | - | {A,B,C,D} | Goal found! |

**Step 1:** Expand A, add B (3), C (4)  
**Step 2:** Expand B, add D (5)  
**Step 3:** Expand C, D (6) ignored (D at 5 already)  
**Step 4:** Expand D (goal found)

**Path Reconstruction**

* From A → B → D (because cost to B is 3, then to D is 2, total 5)

**Final Answer**

* Minimum cost path: A → B → D
* Total minimum cost: 5