1. **A perceptron has 3 inputs (x1, x2, x3) with weights (0.3, -0.1, 0.2) and bias -0.2.**

   Calculate the output for input vectors:

   a) (1, 0, 1)

   b) (0, 1, 1)

   Show all steps including the activation function.

   Solution

1. Perceptron Calculation

A perceptron with:

- Inputs: x_1, x_2, x_3
- Weights: w_1=0.3, w_2=-0.1, w_3=0.2
- Bias: b=-0.2
- Activation function: Step function (outputs 1 if net input ≥ 0, else 0)

a) For input (1, 0, 1):

1. Calculate net input:

   $z = (x_1 w_1) + (x_2 w_2) + (x_3 * w_3) + b$

   $z = (1*0.3) + (0*-0.1) + (1*0.2) + (-0.2)$

   $z = 0.3 + 0 + 0.2 - 0.2 = 0.3$

2. Apply activation function:

   Since $z = 0.3 \geq 0$, output = 1

b) For input (0, 1, 1):

1. Calculate net input:

   $z = (0*0.3) + (1*-0.1) + (1*0.2) + (-0.2)$

   $z = 0 - 0.1 + 0.2 - 0.2 = -0.1$

2. Apply activation function:

   Since $z = -0.1 < 0$, output = 0

2. **Design a genetic algorithm to find the maximum value of the function $f(x) = x^2$ in the range [-10, 10]. Explain:**

- **Chromosome representation**
- **Fitness function**
- **Crossover and mutation operators**
- **Selection method**

Solution

Chromosome Representation:

- Use binary strings of length 10 (allowing $2^{10}=1024$ values between -10 and 10)
- Each chromosome represents a real number x in [-10,10]
- Example: 0000000000 → -10, 1111111111 → +10

Fitness Function:

- Directly use $f(x) = x^2$ as fitness since we're maximizing
- Higher x^2 → higher fitness

Crossover Operator:

- Single-point crossover: Select random crossover point, swap segments
- Example: Parents 101100|1101 and 011010|1010 → Children 101100|1010 and 011010|1101

Mutation Operator:

- Bit flip mutation: Randomly flip bits with small probability (e.g., 0.01)
- Example: 1011001101 → 1010001101 (4th bit flipped)

Selection Method:

- Tournament selection: Randomly select k individuals, choose the fittest
- Roulette wheel selection: Probability proportional to fitness

3. **For a given multilayer perceptron with one hidden layer, calculate the backpropagation updates for one training example. Show the forward pass and backward pass calculations.**

Solution

Consider a network with:

- Input layer: 2 nodes (x_1, x_2)
- Hidden layer: 2 nodes (h_1, h_2) with sigmoid activation
- Output layer: 1 node (y) with sigmoid activation
- Weights: w_{11}=0.1, w_{12}=0.2 (input to h_1), w_{21}=-0.1, w_{22}=0.3 (input to h_2)
- Hidden to output weights: v_1=0.2, v_2=-0.1
- Bias: b_1=0.1 (hidden), b_2=0.2 (output)
- Target output t=1 for input (1,0)

Forward Pass:

1. Calculate hidden layer inputs:
   $net\_h_1 = 1*0.1 + 0*0.2 + 0.1 = 0.2$
   $net\_h_2 = 1*-0.1 + 0*0.3 + 0.1 = 0.0$
2. Apply sigmoid:
   $h_1 = 1/(1+e^{(-0.2)}) \approx 0.5498$
   $h_2 = 1/(1+e^{0}) = 0.5$
3. Calculate output:
   $net\_y = 0.5498*0.2 + 0.5*-0.1 + 0.2 \approx 0.15996$
   $y = 1/(1+e^{(-0.15996)}) \approx 0.5399$

Backward Pass:

1. Output error:
   $\delta\_y = (t-y)*y*(1-y) = (1-0.5399)*0.5399*(1-0.5399) \approx 0.1146$
2. Hidden layer errors:
   $\delta\_h_1 = h_1*(1-h_1)*v_1\delta\_y \approx 0.5498(1-0.5498)*0.2*0.1146 \approx 0.0057$
   $\delta\_h_2 = h_2*(1-h_2)*v_2\delta\_y \approx 0.5(1-0.5)*-0.1*0.1146 \approx -0.0029$
3. Weight updates (with learning rate η=0.1):
   $\Delta v_1 = \eta\delta\_y h_1 \approx 0.1*0.1146*0.5498 \approx 0.0063$
   $\Delta v_2 = \eta\delta\_y h_2 \approx 0.1*0.1146*0.5 \approx 0.0057$
   $\Delta w_{11} = \eta\delta\_h_1 x_1 \approx 0.1*0.0057*1 \approx 0.00057$
   $\Delta w_{12} = \eta\delta\_h_1 x_2 \approx 0$ (since x_2=0)
   $\Delta w_{21} = \eta\delta\_h_2 x_1 \approx 0.1*-0.0029*1 \approx -0.00029$
   $\Delta w_{22} = \eta\delta\_h_2 x_2 \approx 0$ (since x_2=0)

4. **Compare and contrast genetic programming and genetic algorithms using a practical example from time series forecasting.**

Solution

Time Series Forecasting Example:

Genetic Algorithm (GA):

- Represents solution as fixed-length strings (parameters for a model)
- Example: Optimizing ARIMA(p,d,q) parameters
  - Chromosome: [p,d,q] where $p \in [0,5]$, $d \in [0,2]$, $q \in [0,5]$
  - Fitness: Mean squared error on validation set
  - Crossover: Swap p,d,q values between parents
  - Mutation: Randomly increment/decrement one parameter

Genetic Programming (GP):

- Represents solution as variable-size trees (actual model structure)
- Example: Evolving mathematical expressions
  - Chromosome: Tree representing formula (e.g., +(*(x(t-1),0.5), -(x(t-2)))
  - Fitness: Prediction accuracy
  - Crossover: Swap subtrees between parents
  - Mutation: Replace a node with a random node or subtree

Comparison:

- GA searches parameter space, GP searches program space
- GP can discover novel model structures, GA finds optimal parameters for a given structure
- GP is more computationally intensive but potentially more creative
- GA is more constrained but often faster to converge

5. Design a single Perceptron architecture to represent the Boolean AND and OR Function

Solution

AND Function ($x_1 \wedge x_2$):

- Truth table requires output 1 only when both inputs are 1
- Weights: w_1=0.5, w_2=0.5, bias=-0.7
- Calculations:
  (0,0): 0.5*0 + 0.5*0 -0.7 = -0.7 → 0
  (0,1): 0.5*0 + 0.5*1 -0.7 = -0.2 → 0
  (1,0): 0.5*1 + 0.5*0 -0.7 = -0.2 → 0
  (1,1): 0.5*1 + 0.5*1 -0.7 = 0.3 → 1

OR Function ($x_1 \vee x_2$):

- Truth table requires output 1 when either input is 1
- Weights: w_1=0.5, w_2=0.5, bias=-0.2
- Calculations:
  (0,0): 0.5*0 + 0.5*0 -0.2 = -0.2 → 0
  (0,1): 0.5*0 + 0.5*1 -0.2 = 0.3 → 1
  (1,0): 0.5*1 + 0.5*0 -0.2 = 0.3 → 1
  (1,1): 0.5*1 + 0.5*1 -0.2 = 0.8 → 1

Perceptron Architecture:

- Input layer: 2 nodes (x_1, x_2)
- Single output node with step activation
- For AND: weights (0.5,0.5), bias -0.7
- For OR: weights (0.5,0.5), bias -0.2
- Note: These are not unique solutions - any weights satisfying the inequalities would work