Indian Institute of Technology Indore
Computer Science & Engineering
CS 354N: Assignment 05 Perceptron

ANIKEIT SETHI

190001003

In [1]:

```
import numpy as np
class Model:
  def __init__(self, num_inputs, init_wts):
    self.weights = init_wts
    self.bias = 0
  def predict(self, inputs):
    sum = np.dot(inputs, self.weights[:]) + self.bias
    if sum > 0:#Activation
      prediction = 1
    else:
      prediction = 0
    return prediction
  def fit(self, inputs, label, epochs, learning_rate):
    for _ in range(epochs):
      for val, res in zip(inputs, label):
        prediction = self.predict(val)
        self.weights[:] += learning_rate * (res-prediction) * val
        self.bias += learning_rate * (res-prediction)
```

In [15]:

```
# AND GATE
import numpy as np

train_inputs= np.array([
    [1,1], # both yes
    [1,0], # one yes, one no
    [0,1], # one no, one yes
    [0,0]  # both no
])
labels= np.array([1,0,0,0])

epochs = 50
learning_rate = 0.1
init_wts = [0.2,0.2]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 50
# as the learning rate is very low for only 50 epochs thus the model is unable to give co
rrect ressults
learning_rate = 0.001
init_wts = [0.2,0.2]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0.2, 0.2]
[1 1] 1
[1 0] 0
[0 1] 0
[0 0] 0

Number of Epochs: 50
Learning Rate: 0.001
Initial Weights: [0.2, 0.2]
[1 1] 1
[1 0] 1
```

```
[0 1] 1
[0 0] 0
```

In [3]:

```
# OR GATE
import numpy as np

train_inputs=  np.array([
    [1,1], # both yes
    [1,0], # one yes, one no
    [0,1], # one no, one yes
    [0,0]  # both no
])
labels= np.array([1,1,1,0]))

epochs = 50
learning_rate = 0.1
init_wts = [0.5,0.5]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 100
# adjusting the learning rate with the number of epochs gives the correct results
learning_rate = 0.001
init_wts = [0.1,0.3]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0.5, 0.5]
[1 1] 1
[1 0] 1
[0 1] 1
[0 0] 0

Number of Epochs: 100
Learning Rate: 0.001
Initial Weights: [0.1, 0.3]
[1 1] 1
[1 0] 1
[0 1] 1
[0 0] 0
```

In [16]:

```
# NAND GATE
import numpy as np

train_inputs=  np.array([
    [1,1], # both yes
    [1,0], # one yes, one no
    [0,1], # one no, one yes
    [0,0]  # both no
])

epochs = 10
learning_rate = 0.1
init_wts = [0,0]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
```

```python
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 10
learning_rate = 0.1
init_wts = [0.3,0.3]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 10
Learning Rate: 0.1
Initial Weights: [0, 0]
[1 1] 1
[1 0] 0
[0 1] 0
[0 0] 0

Number of Epochs: 10
Learning Rate: 0.1
Initial Weights: [0.3, 0.3]
[1 1] 1
[1 0] 0
[0 1] 0
[0 0] 0
```

In [17]:

```python
# NOR GATE
import numpy as np

train_inputs=  np.array([
    [1,1], # both yes
    [1,0], # one yes, one no
    [0,1], # one no, one yes
    [0,0]  # both no
])

epochs = 50
learning_rate = 0.1
init_wts = [0,0]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 100
learning_rate = 0.001
init_wts = [0.1,0.3]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(2, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0, 0]
[1 1] 1
[1 0] 0
[0 1] 0
[0 0] 0

Number of Epochs: 100
Learning Rate: 0.001
```

```
Initial Weights: [0.1, 0.3]
[1 1] 1
[1 0] 0
[0 1] 1
[0 0] 0
```

In [19]:

```
# AND GATE (3 Input)
import numpy as np

train_inputs= np.array([
  [1,1,1],
  [1,1,0],
  [1,0,1],
  [1,0,0],
  [0,1,1],
  [0,1,0],
  [0,0,1],
  [0,0,0]
  ])
labels = np.array([1,0,0,0,0,0,0,0])

epochs = 50
learning_rate = 0.1
init_wts = [0,0,0]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 100
learning_rate = 0.01
init_wts = [0.1,0.3,0.9]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0, 0, 0]
[1 1 1] 1
[1 1 0] 0
[1 0 1] 0
[1 0 0] 0
[0 1 1] 0
[0 1 0] 0
[0 0 1] 0
[0 0 0] 0

Number of Epochs: 100
Learning Rate: 0.01
Initial Weights: [0.1, 0.3, 0.9]
[1 1 1] 1
[1 1 0] 0
[1 0 1] 0
[1 0 0] 0
[0 1 1] 0
[0 1 0] 0
[0 0 1] 0
[0 0 0] 0
```

In [7]:

```
# OR GATE (3 Input)
import numpy as np
```

```
train_inputs= np.array([
  [1,1,1],
  [1,1,0],
  [1,0,1],
  [1,0,0],
  [0,1,1],
  [0,1,0],
  [0,0,1],
  [0,0,0]
  ])
labels = np.array([1,1,1,1,1,1,1,0])

epochs = 50
learning_rate = 0.1
init_wts = [0,0,0]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 100
learning_rate = 0.01
init_wts = [0.1,0.3,0.9]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0, 0, 0]
[1 1 1] 1
[1 1 0] 1
[1 0 1] 1
[1 0 0] 1
[0 1 1] 1
[0 1 0] 1
[0 0 1] 1
[0 0 0] 0

Number of Epochs: 100
Learning Rate: 0.01
Initial Weights: [0.1, 0.3, 0.9]
[1 1 1] 1
[1 1 0] 1
[1 0 1] 1
[1 0 0] 1
[0 1 1] 1
[0 1 0] 1
[0 0 1] 1
[0 0 0] 0
```

In [20]:

```
# NAND GATE (3 Input)
import numpy as np

train_inputs= np.array([
  [1,1,1],
  [1,1,0],
  [1,0,1],
  [1,0,0],
  [0,1,1],
  [0,1,0],
  [0,0,1],
  [0,0,0]
  ])
labels = np.array([0,1,1,1,1,1,1,1])
```

```
epochs = 50
learning_rate = 0.1
init_wts = [0,0,0]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))

epochs = 25
learning_rate = 0.1
init_wts = [0.1,0.3,0.9]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0, 0, 0]
[1 1 1] 0
[1 1 0] 1
[1 0 1] 1
[1 0 0] 1
[0 1 1] 1
[0 1 0] 1
[0 0 1] 1
[0 0 0] 1

Number of Epochs: 25
Learning Rate: 0.1
Initial Weights: [0.1, 0.3, 0.9]
[1 1 1] 1
[1 1 0] 1
[1 0 1] 1
[1 0 0] 1
[0 1 1] 1
[0 1 0] 1
[0 0 1] 1
[0 0 0] 1
```

In [9]:

```
# NOR GATE (3 Input)
import numpy as np

train_inputs= np.array([
  [1,1,1],
  [1,1,0],
  [1,0,1],
  [1,0,0],
  [0,1,1],
  [0,1,0],
  [0,0,1],
  [0,0,0]
  ])
labels = np.array([0,0,0,0,0,0,0,1])

epochs = 50
learning_rate = 0.1
init_wts = [0,0,0]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
epochs = 100
learning_rate = 0.01
init_wts = [0.1,0.3,0.9]
print(f"\nNumber of Epochs: {epochs}\nLearning Rate: {learning_rate}\nInitial Weights: {i
nit_wts}")
p_M = Model(3, init_wts)
p_M.fit(train_inputs, labels, epochs, learning_rate)
for inputs in train_inputs:
  print(inputs, p_M.predict(inputs))
```

```
Number of Epochs: 50
Learning Rate: 0.1
Initial Weights: [0, 0, 0]
[1 1 1] 0
[1 1 0] 0
[1 0 1] 0
[1 0 0] 0
[0 1 1] 0
[0 1 0] 0
[0 0 1] 0
[0 0 0] 1

Number of Epochs: 100
Learning Rate: 0.01
Initial Weights: [0.1, 0.3, 0.9]
[1 1 1] 0
[1 1 0] 0
[1 0 1] 0
[1 0 0] 0
[0 1 1] 0
[0 1 0] 0
[0 0 1] 0
[0 0 0] 1
```

In [10]:

```
class Perceptron:
    # Constructor
    def __init__(self, num_of_features, init_wts):
        self.num_of_features = num_of_features
        self.weights = np.array(init_wts, dtype = float).reshape(num_of_features, 1)
        self.bias = 0

    # Predict output using learned weights
    def predict(self, input):
        product = np.dot(self.weights.T, input) + self.bias
        if product >= 0.0:
            return 1
        return 0

    # Fit the model on training data to learn weights
    def fit(self, data, labels, learning_rate = 0.1, epochs = 20):
        training_data_size = np.array(data).shape[0]
        for i in range(epochs):
            for j in range(training_data_size):
                predicted_output = self.predict(data[j])
                self.weights += np.array(learning_rate * (labels[j] - predicted_output)
* data[j]).reshape(self.num_of_features, 1)
                self.bias += learning_rate * (labels[j] - predicted_output)
```

In [11]:

```
import numpy as np
training_data_2 = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
labels_and_2 = np.array([0, 0, 0, 1])
labels_or_2 = np.array([0, 1, 1, 1])
labels_nand_2 = np.array([1, 1, 1, 0])
labels_nor_2 = np.array([1, 0, 0, 0])
```

In [12]:

```
init_weights_21 = [-1, -1]
```

```
init_weights_22 = [-1, 1]
learning_rate_21 = 0.1
learning_rate_22 = 0.5
epochs_21 = 3
epochs_22 = 10
```

In [13]:

```
# Test case - 1
model = Perceptron(2, init_weights_21)
model.fit(training_data_2, labels_and_2, learning_rate = learning_rate_21, epochs = epoc
hs_21)
print("Initial Weights: ", init_weights_21)
print("Learning Rate: ", learning_rate_21)
print("Epochs: ", epochs_21)
for input in training_data_2:
    print(input, " : ", model.predict(input))
print("Weights: ", model.weights.flatten())
print("Bias: ", model.bias)
```

```
Initial Weights:  [-1, -1]
Learning Rate:  0.1
Epochs:  3
[0 0]  :  1
[0 1]  :  0
[1 0]  :  0
[1 1]  :  0
Weights:  [-0.7 -0.7]
Bias:  0.0
```