In [1]:
```
import numpy as np
```

ART Class

In [2]:
```
class ART:
    def __init__(self, input_size, rho, alpha):
        self.W = np.ones((1, input_size))
        self.rho = rho
        self.alpha = alpha

    def train(self, x):
        while True:
            y = self.predict(x)

            if y is not None:
                self.W = self.alpha * x + (1 - self.alpha) * self.W
                return
            else:
                self.W = np.ones((1, input_size))

    def predict(self, x):
        y = x.dot(self.W.T)
        if y >= self.rho:
            return y
        else:
            return None
```

Initializing parameters

In [3]:
```
input_size = 2
rho = 0.9
alpha = 0.1
```

Input Parameters

In [4]:
```
network = ART(input_size, rho, alpha)
x1 = np.array([0.7, 0.3])
x2 = np.array([0.2, 0.8])
x3 = np.array([0.6, 0.6])
```

Train and Test

In [5]:
```
network.train(x1)
network.train(x2)
network.train(x3)
print(network.W)
```

```
[[0.8637 0.8853]]
```

Prediction

In [6]:
```
x_new = np.array([0.6, 1.2])
pred = network.predict(x_new)
if pred is not None:
    print("Resonance achieved with value:", pred)
else:
    print("No resonance achieved. Input does not match any learned patter
n.")
```

```
Resonance achieved with value: [1.58058]
```