5. For HW 4, report, below, if your transfer/activation function performs better after 50 epochs, than the sigmoid function.

After 50 epochs, my activation function (hard limit) did not perform better than the sigmoid activation function.

Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$	

Code added, line numbers included for clarity:

_hard_limit()

```
    # added for HW4
    def _hard_limit(self, z):
        return np.where(z >= 0, 1, 0)

```

Code modified:

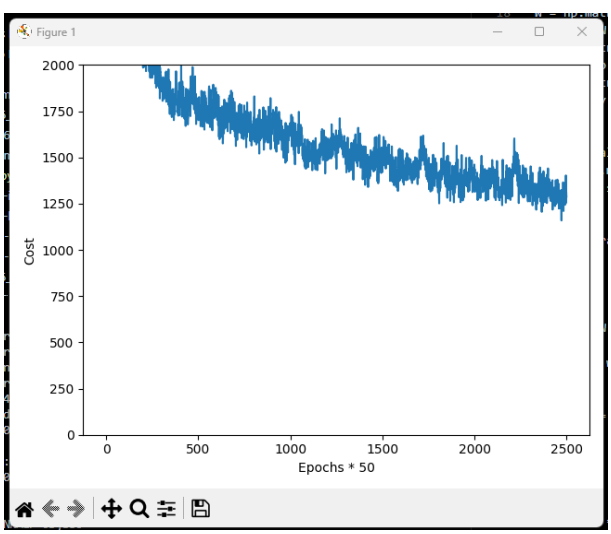
_feed_forward()

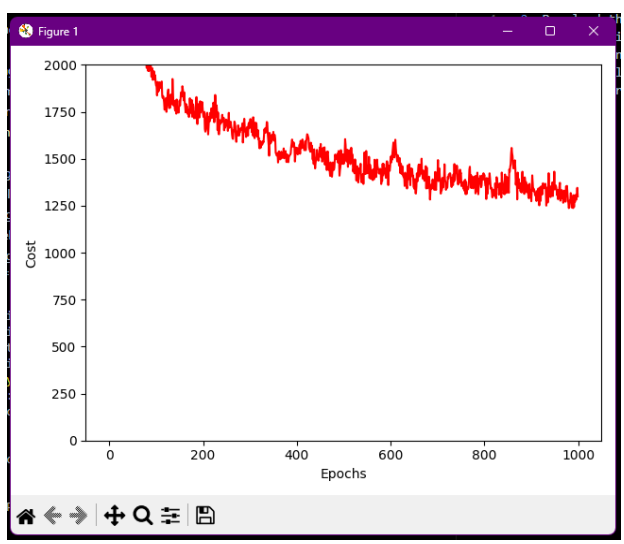
```
        a1 = self._add_bias_unit(X, how="column")
        z2 = w1.dot(a1.T)
        # a2 = self._sigmoid(z2)
        a2 = self._hard_limit(z2)
        a2 = self._add_bias_unit(a2, how="row")
        z3 = w2.dot(a2)
        # a3 = self._sigmoid(z3)
        a3 = self._hard_limit(z3)
        return a1, z2, a2, z3, a3
```

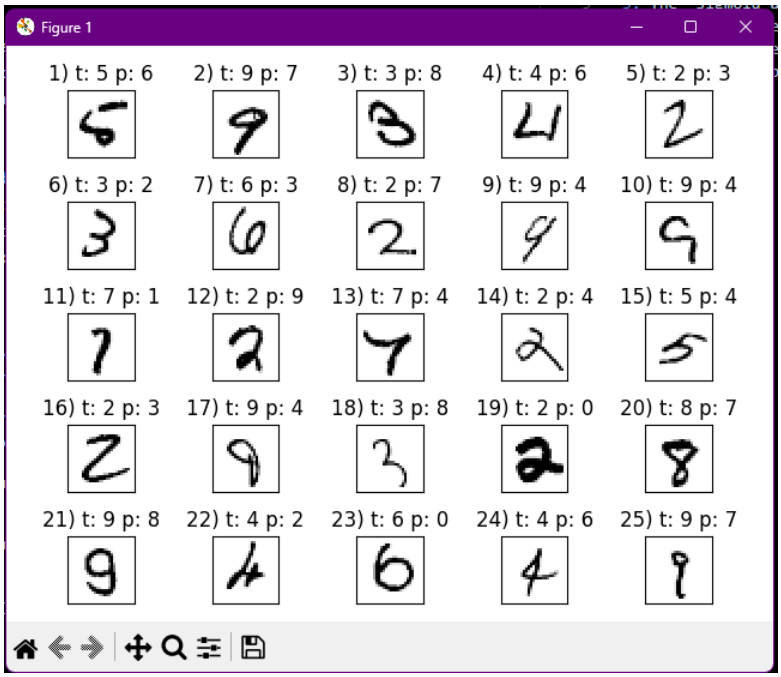
_get_cost()

```
        errors = y_enc - output
        cost = np.mean(errors**2)
        L1_term = self._L1_reg(self.l1, w1, w2)
        L2_term = self._L2_reg(self.l2, w1, w2)
        cost = cost + L1_term + L2_term
        return cost
        """
        OLD CODE
        term1 = -y_enc * (np.log(output))
        term2 = (1 - y_enc) * np.log(1 - output)
        cost = np.sum(term1 - term2)
        L1_term = self._L1_reg(self.l1, w1, w2)
        L2_term = self._L2_reg(self.l2, w1, w2)
        cost = cost + L1_term + L2_term
        return cost
        """
```

Model using Sigmoid Activation Function

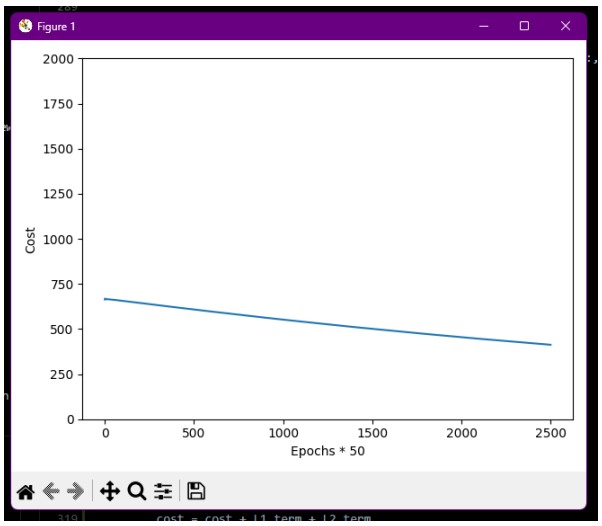


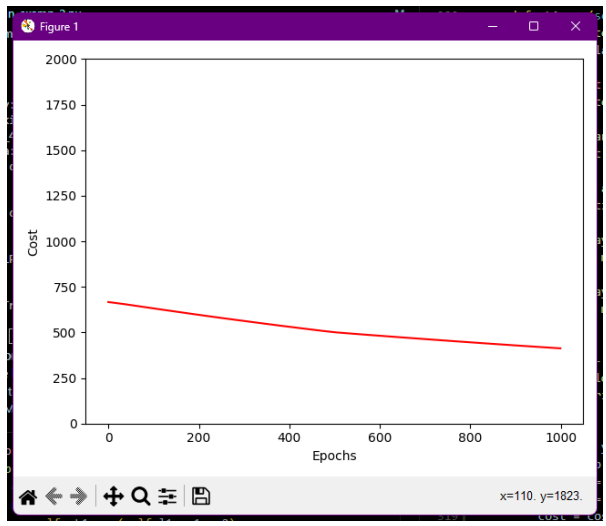


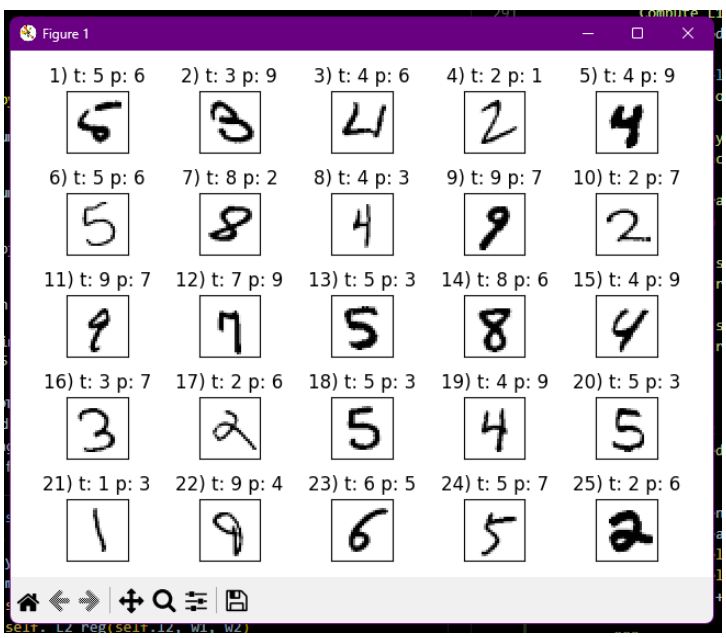



```
Epoch: 50/50 Training accuracy: 90.49%
Test accuracy: 90.90%
```

Model using Hard Limit Activation Function








```
Epoch: 50/50 Training accuracy: 85.28%
Test accuracy: 85.86%
```