RBM

IMPLEMENTATION

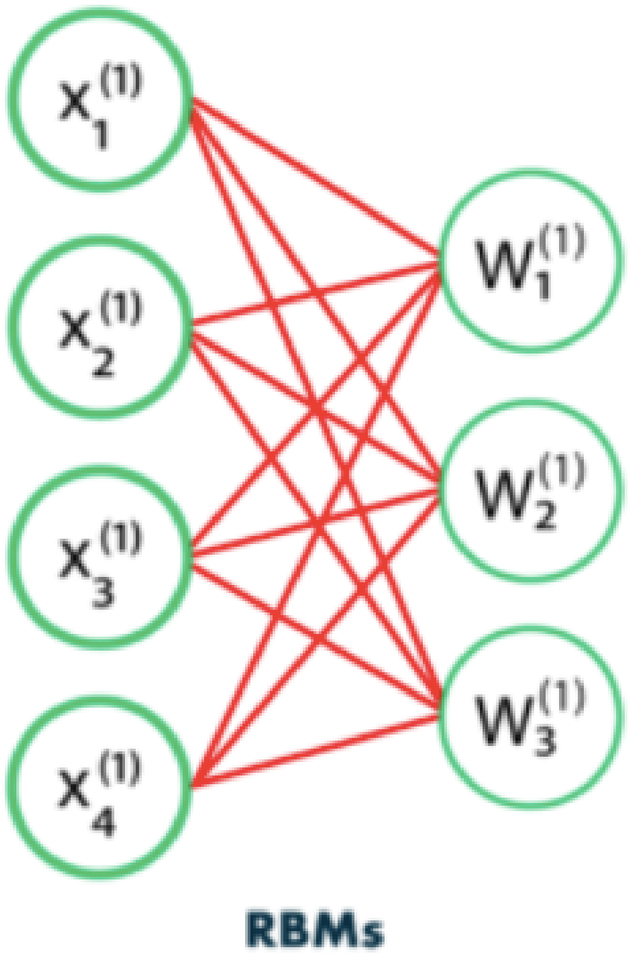
AND TEST ON MNIST DATASET

ASSIGNMENT #3

BY-YOHANNIS KIFLE TELILA

03 MAY

RBM MODEL

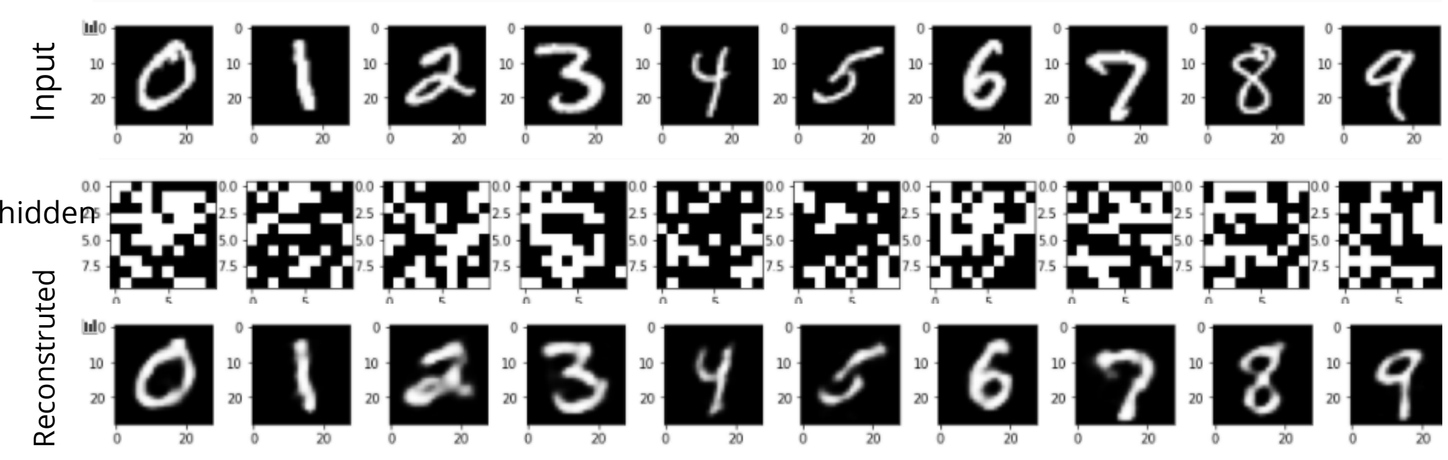


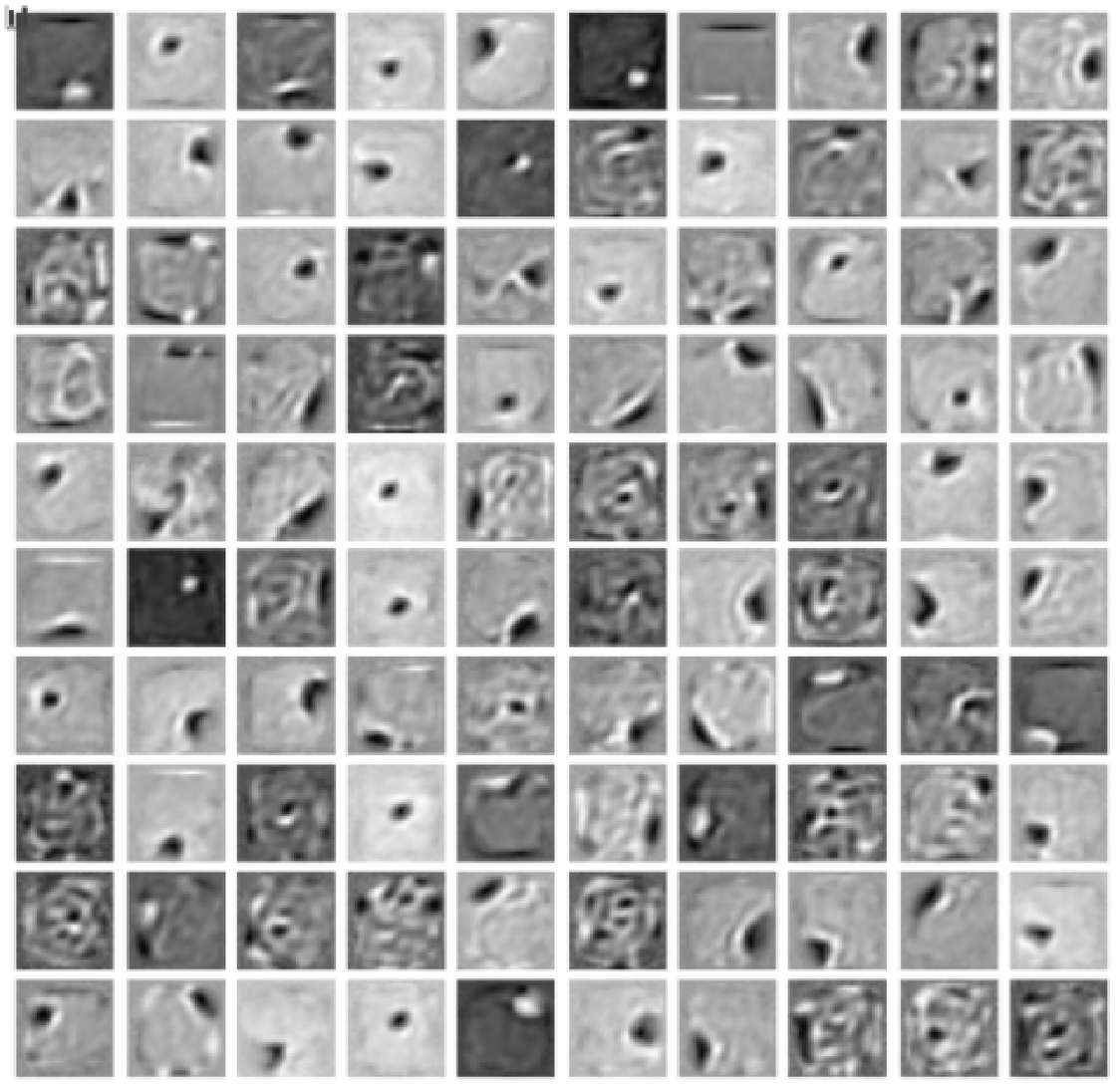
RBMs

```python
def posetivePhase(self, visibleLayer):
    # probability distribution of the hidden layer.
    pdH = expit(np.matmul(visibleLayer, self.vhW) + self.hlbias)
    return (pdH, np.random.binomial(1, p=pdH))
def negativePhase(self, hiddenLayer):
    # probability distribution of the visible layer.
    pdV = expit(np.matmul(hiddenLayer, self.vhW.T) + self.vlbias)
    return (pdV, np.random.binomial(1, p=pdV))
def compute_error_and_grads(self, batch):
    batchSize = batch.shape[0]
    v0 = batch.reshape(batchSize, -1)

    # Compute gradients - Positive Phase
    ph0, h0 = self.posetivePhase(v0)
    vhW_delta = np.matmul(v0.T, ph0)
    vb_delta = np.sum(v0, axis=0)
    hb_delta = np.sum(ph0, axis=0)

    # Compute gradients - Negative Phase
    pv1, v1 = self.negativePhase(h0)
    ph1, h1 = self.posetivePhase(pv1)
```

RBM RESULT


RBM Training error
Error
55000
50000
45000
40000
35000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Input
hidden
Reconstruted



TESTING ON MNIST

```python
# Encoding the representation
encoding = []
for i in range(len(trainingImages)):
    pdH, hiddenSampled = mnistRbm.posetivePhase(trainingImages[i].reshape(28*28))
    encoding.append(hiddenSampled)
encodedDfx = pd.DataFrame(encoding)
```

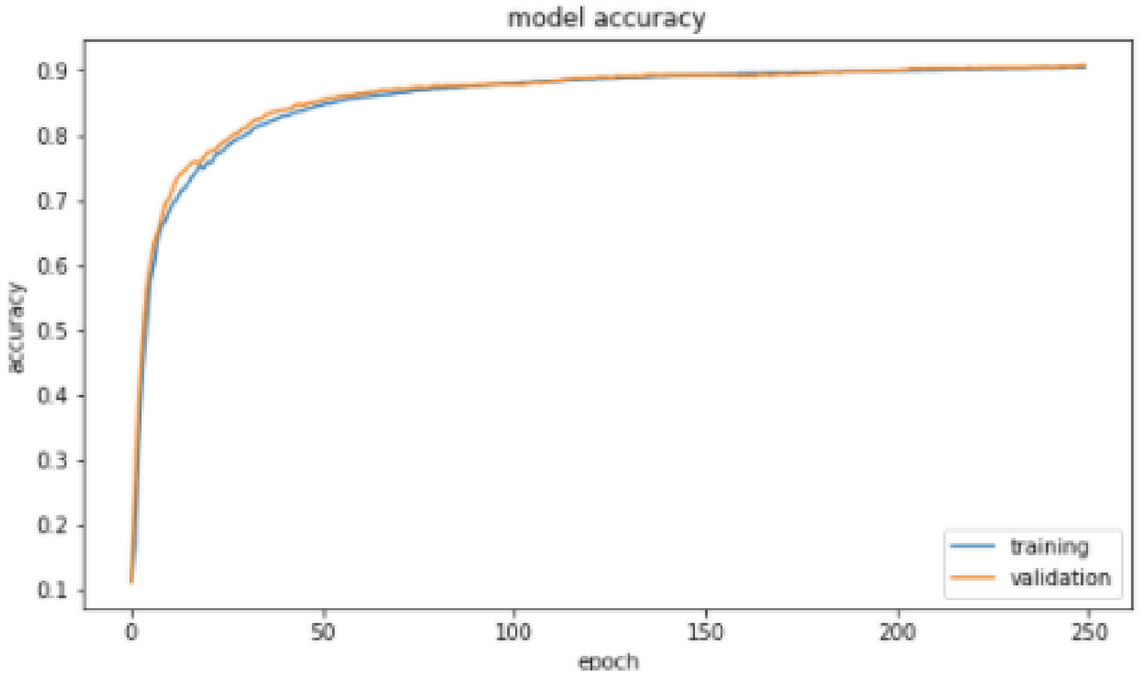
```python
# svc model
model_linear = SVC(C=20, gamma=0.01, kernel="rbf")
model_linear.fit(X_train, y_train)
y_pred = model_linear.predict(X_test)
```

```
accuracy: 0.9413541666666667

[[4598    0   23   15    3   14   29    0   30    9]
 [   1 5307   23   10    4    5    5   10   12   11]
 [  16   23 4524   34   19    7   18   46   41   10]
 [  17   22   80 4504    1  132    7   48   79   36]
 [   6   20   36   12 4350    6   34   25   20  172]
 [  34   21   22  111   16 3976   63    5   51   30]
 [  37   18   23    8   17   44 4595    0   16    0]
 [  10   30   45   37   40    9    0 4750   12  101]
 [  20   64   61   90   27   84   38   25 4205   51]
 [  11   13   11   47  126   25    3  110   38 4376]]
```

	precision	recall	f1-score	support
0	0.97	0.97	0.97	4721
1	0.96	0.98	0.97	5388
2	0.93	0.95	0.94	4738
3	0.93	0.91	0.92	4926
4	0.95	0.93	0.94	4681
5	0.92	0.92	0.92	4329
6	0.96	0.97	0.96	4758
7	0.95	0.94	0.94	5034
8	0.93	0.90	0.92	4665
9	0.91	0.92	0.92	4760
accuracy			0.94	48000
macro avg	0.94	0.94	0.94	48000
weighted avg	0.94	0.94	0.94	48000

NN WITH SOFTMAX LAYER

```
image_size = 100 # 10*10
num_classes = 10 # ten unique digits
model = Sequential()
model.add(Dense(units=50, activation='sigmoid',kernel_initializer='uniform', input_shape=
(image_size,)))
model.add(Dense(units=num_classes, activation='softmax'))
model.summary()
```




```
[24]
print(f'Test loss: {loss:.3}')
print(f'Test accuracy: {accuracy:.3}')
```

```
Test loss: 0.15
Test accuracy: 0.901
```

END

THANK YOU

 github.com/joekifle