MIDTERM 2 – ASSIGNMENT 3

Restricted Boltzmann Machine &
Deep Restricted Boltzmann Network

Pasquali Alex

RBM training code

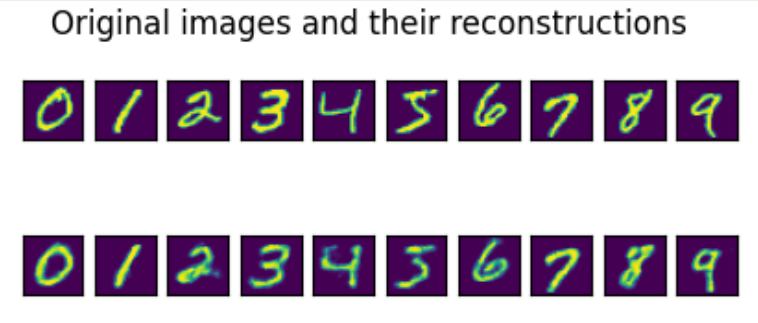
```python
def contrastive_divergence(self, v_probs, k):
    """
    Perform one step of contrastive divergence
    :param v_probs: vector of the visible units activations (non-binarized)
    :param k: (int) order of the Gibbs sampling
    """
    # compute wake part
    v_sample = np.random.binomial(n=1, p=v_probs, size=len(v_probs))
    h_probs, h_sample = self.ph_v(v_sample)
    wake = np.outer(h_probs, v_probs)
    # compute dream part
    v_probs_gibbs, v_sample_gibbs, h_probs_gibbs, h_sample_gibbs = self.gibbs_sampling(h_sample, k)
    dream = np.outer(h_probs_gibbs, v_probs_gibbs)
    # compute deltas
    delta_W = np.subtract(wake, dream)
    delta_bv = np.subtract(v_sample, v_sample_gibbs)
    delta_bh = np.subtract(h_sample, h_sample_gibbs)
    return delta_W, delta_bv, delta_bh
```

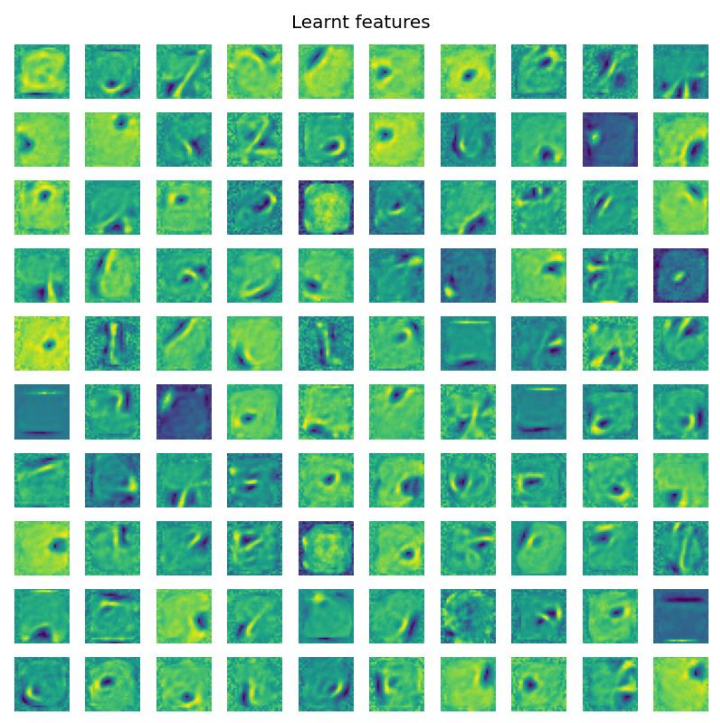
```python
def gibbs_sampling(self, h_sample, k):
    """
    Performs Gibbs sampling
    :param h_sample: binary vector of the hidden activations
    :param k: order of the Gibbs sampling
    """
    v_prob, v_sample, h_prob = None, None, None
    for i in range(k):
        v_prob, v_sample = self.pv_h(h_sample)
        h_prob, h_sample = self.ph_v(v_sample)
    return v_prob, v_sample, h_prob, h_sample
```

```python
def ph_v(self, v_sample):
    h_probs = sigmoid(np.add(np.matmul(self.W, v_sample), self.bias_hidden))
    h_samples = np.random.binomial(n=1, p=h_probs, size=len(h_probs))
    return h_probs, h_samples
```

```python
def pv_h(self, h_sample):
    v_probs = sigmoid(np.add(np.matmul(h_sample, self.W), self.bias_visible))
    v_samples = np.random.binomial(n=1, p=v_probs, size=len(v_probs))
    return v_probs, v_samples
```

RBM results





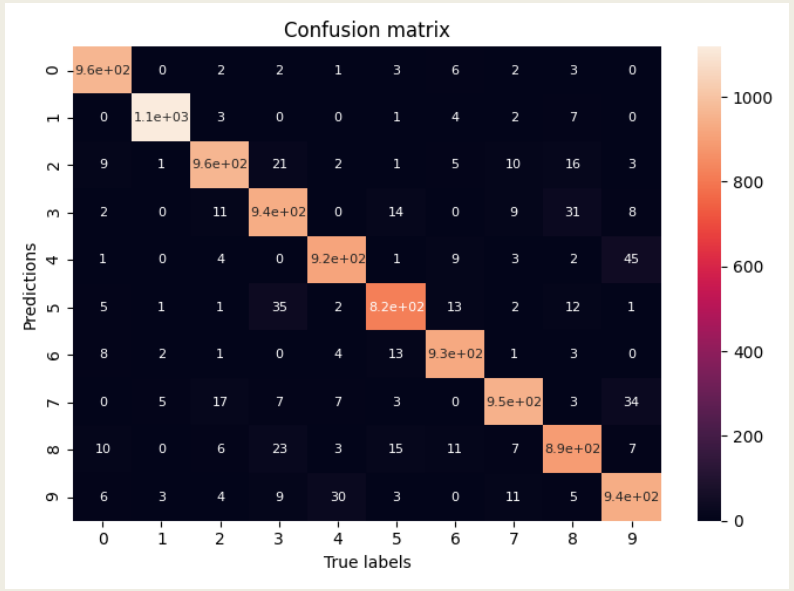
Classifier's architecture:

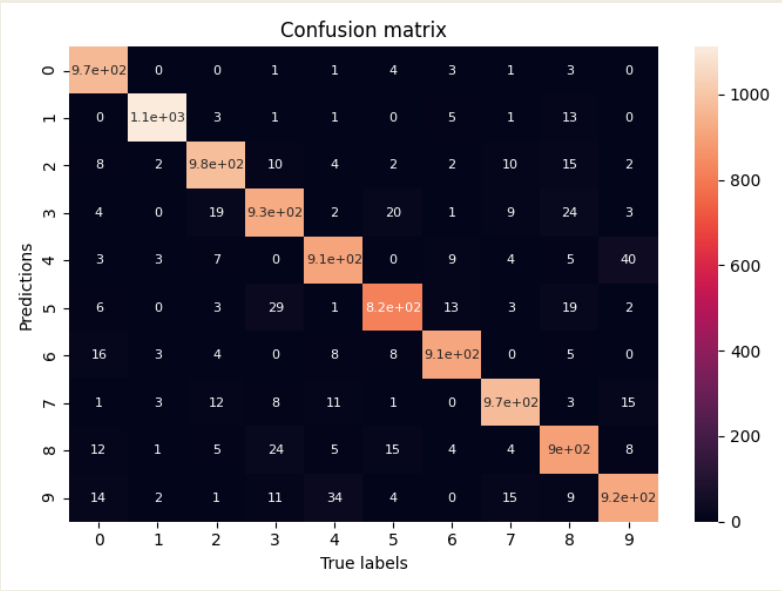
```
self.classifier = tf.keras.models.Sequential([
    Dense(units=50, activation='relu', input_dim=sizes[-1]),
    Dense(units=10, activation='softmax')
])
```

Contrastive divergence
Epochs = 1, lr = 0.1, k = 1
94.61% accuracy

Persistent contrastive divergence
Epochs = 5, lr = 0.05, k = 1
94.48% accuracy

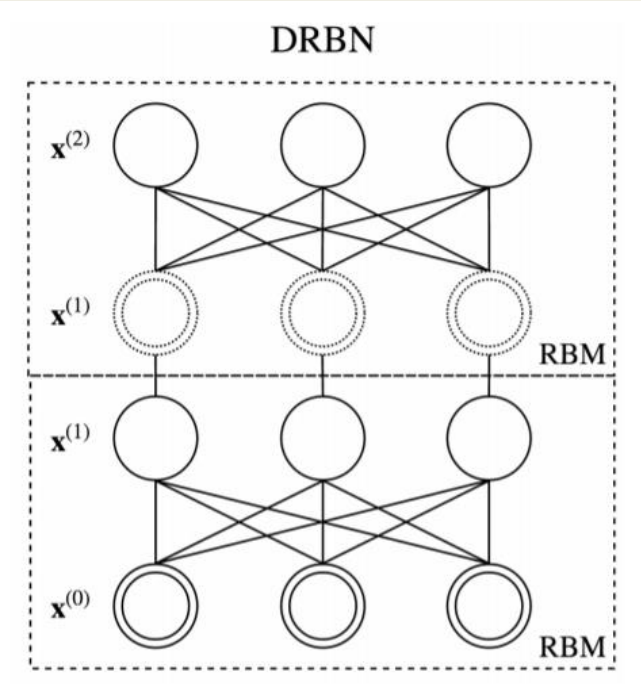
(2008 – Tieleman)





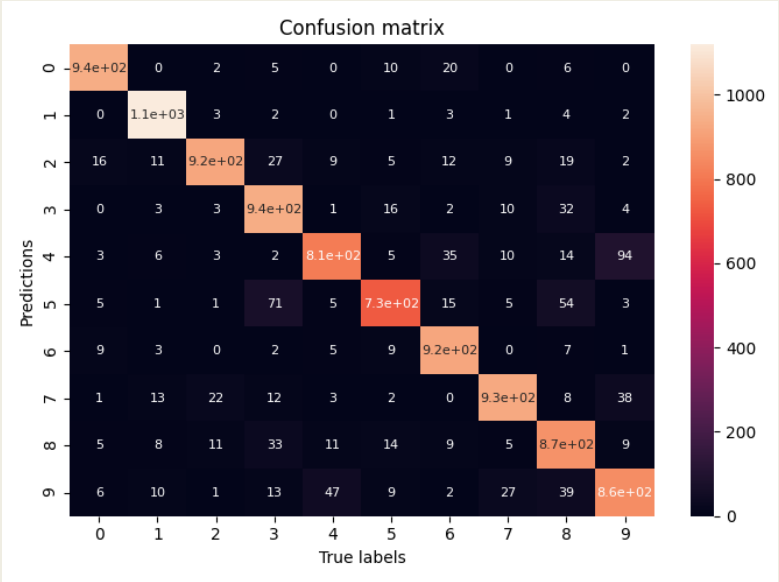
Deep Restricted Boltzmann Network (DRBN)

(2016 - Hengyuan Hu, Lisheng Gao, Quanbin Ma)

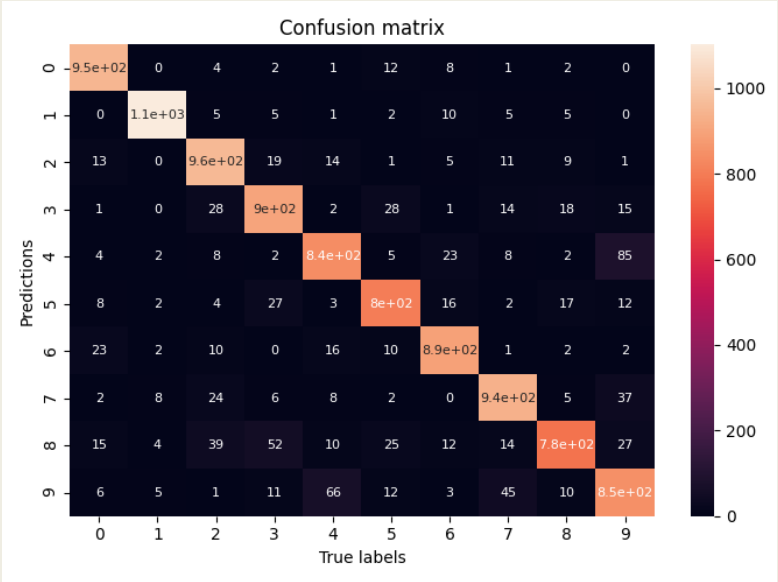


- Neural network where each layer is an RBM
- Hidden units at each layer are also the visible units in the next layer
- All RBMs in the network are **trained jointly**

Persistent contrastive divergence
Architecture: (784, 500, 1000),
epochs = 2, lr = 1e-4, k = 1
90.32% accuracy



Contrastive divergence
Architecture: (784, 500, 1000),
epochs = 2, lr = 1e-4, k = 1
90.27% accuracy



Possible improvements

- Implement a multi-chain PCD ⇒ Hu at al. use a batch size of 100 and suggest to use a Gibbs chain for each pattern in a minibatch
- Make a comparison between resetting or not the Gibbs chains ⇒ Tieleman says that it's possible to reset the chains at regular intervals, but suggests to never do it
- Implement a decay of the learning rate and a weight decay ⇒ Tieleman suggests to use both a linear decay of the learning rate and a weight decay
- Merge the concepts of **Convolutional RBM** with **Deep Restricted Boltzmann Networks**

  (2016 - Hengyuan Hu, Lisheng Gao, Quanbin Ma)

THANK YOU FOR YOUR ATTENTION

RBM & DRBN – Pasquali Alex