```
from sklearn.utils import gen_even_slices
import numpy as np
import itertools

class RBM:
  def __init__(self, num_hidden, num_visible, learning_rate,max_epochs=10,
               batch_size=10):
    self.num_hidden = num_hidden
    self.num_visible = num_visible
    self.learning_rate = learning_rate
    self.weights = 0.1 * np.random.randn(self.num_visible, self.num_hidden)
    self.weights = np.insert(self.weights, 0, 0, axis = 0)
    self.weights = np.insert(self.weights, 0, 0, axis = 1)
    self.max_epochs = max_epochs
    self.batch_size = batch_size

  def run_visible(self, data):
    num_examples = data.shape[0]

    hidden_states = np.ones((num_examples, self.num_hidden + 1))

    data = np.insert(data, 0, 1, axis = 1)

    hidden_activations = np.dot(data, self.weights)
    hidden_probs = self._logistic(hidden_activations)
    hidden_states[:,:] = hidden_probs > \
        np.random.rand(num_examples, self.num_hidden + 1)
    hidden_states = hidden_states[:,1:]
    return hidden_states


  def run_hidden(self, data):
    num_examples = data.shape[0]

    visible_states = np.ones((num_examples, self.num_visible + 1))

    data = np.insert(data, 0, 1, axis = 1)

    visible_activations = np.dot(data, self.weights.T)
    visible_probs = self._logistic(visible_activations)
    visible_states[:,:] = visible_probs > \
        np.random.rand(num_examples, self.num_visible + 1)

    visible_states = visible_states[:,1:]
    return visible_states

  def _logistic(self, x):
    return 1.0 / (1 + np.exp(-x))

  def _fit(self, v_pos):
    h_pos = self.run_visible(v_pos)
    v_neg = self.run_hidden(self.h_samples_)
    h_neg = self.run_visible(v_neg)
    lr = float(self.learning_rate) / v_pos.shape[0]
    v_pos = np.insert(v_pos, 0, 1, axis = 1)
```

```
        h_pos = np.insert(h_pos, 0, 1, axis = 1)
        v_neg = np.insert(v_neg, 0, 1, axis = 1)
        h_neg = np.insert(h_neg, 0, 1, axis = 1)
        update = np.dot(v_pos.T, h_pos).T
        update -= np.dot(h_neg.T, v_neg)
        self.weights += lr * update.T
        h_neg[np.random.rand(h_neg.shape[0], h_neg.shape[1]) < h_neg] = 1.0  # sample bin
        self.h_samples_ = np.floor(h_neg, h_neg)[:,1:]

    def fit(self, data):
        num_examples = data.shape[0]
        self.h_samples_ = np.zeros((self.batch_size, self.num_hidden))
        n_batches = int(np.ceil(float(num_examples) / self.batch_size))
        batch_slices = list(gen_even_slices(n_batches * self.batch_size,
                                            n_batches, num_examples))

        for iteration in xrange(1, self.max_epochs + 1):
            for batch_slice in batch_slices:
                self._fit(data[batch_slice])

if __name__ == "__main__":
    import numpy as np
    X = np.array([[0, 0, 0], [0, 1, 1], [1, 0, 1], [1, 1, 1]])
    model = RBM(num_hidden=2, num_visible=3, learning_rate=0.1,batch_size=2)
    model.fit(X)
    print model.weights
```

http://videolectures.net/icml09_tieleman_ufw/

http://www.iro.umontreal.ca/~lisa/deep/data/mnist/mnist.pkl.gz