Restricted-Boltzmann-Machines

December 22, 2017

1 Restricted Boltzmann Machines

1.1 Import the appropriate modules

In [1]:
```
import numpy as np
import theano
import theano.tensor as T
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from theano.tensor.shared_randomstreams import RandomStreams
from util import relu, error_rate, getKaggleMNIST, init_weights
from autoencoder import DNN
```

1.2 RBM Class

In [4]:
```
class RBM(object):
    def __init__(self, M, an_id):
        self.M = M
        self.id = an_id
        self.rng = RandomStreams()

    def fit(self, X, learning_rate=0.1, epochs=1, batch_sz=100, show_fig=False):
        N, D = X.shape
        n_batches = N / batch_sz

        W0 = init_weights((D, self.M))
        self.W = theano.shared(W0, 'W_%s' % self.id)
        self.c = theano.shared(np.zeros(self.M), 'c_%s' % self.id)
        self.b = theano.shared(np.zeros(D), 'b_%s' % self.id)
        self.params = [self.W, self.c, self.b]
        self.forward_params = [self.W, self.c]

        # we won't use this to fit the RBM but we will use these for backpropagation lat
        # TODO: technically they should be reset before doing backprop
        self.dW = theano.shared(np.zeros(W0.shape), 'dW_%s' % self.id)
        self.dc = theano.shared(np.zeros(self.M), 'dbh_%s' % self.id)
        self.db = theano.shared(np.zeros(D), 'dbo_%s' % self.id)
        self.dparams = [self.dW, self.dc, self.db]
```

```
self.forward_dparams = [self.dW, self.dc]

X_in = T.matrix('X_%s' % self.id)

# attach it to the object so it can be used later
# must be sigmoidal because the output is also a sigmoid
H = T.nnet.sigmoid(X_in.dot(self.W) + self.c)
self.hidden_op = theano.function(
    inputs=[X_in],
    outputs=H,
)

# we won't use this cost to do any updates
# but we would like to see how this cost function changes
# as we do contrastive divergence
X_hat = self.forward_output(X_in)
cost = -(X_in * T.log(X_hat) + (1 - X_in) * T.log(1 - X_hat)).sum() / (batch_sz
cost_op = theano.function(
    inputs=[X_in],
    outputs=cost,
)

# do one round of Gibbs sampling to obtain X_sample
H = self.sample_h_given_v(X_in)
X_sample = self.sample_v_given_h(H)

# define the objective, updates, and train function
objective = T.mean(self.free_energy(X_in)) - T.mean(self.free_energy(X_sample))

# need to consider X_sample constant because you can't take the gradient of rand
updates = [(p, p - learning_rate*T.grad(
    objective, p, consider_constant=[X_sample])) for p in self.params]
train_op = theano.function(
    inputs=[X_in],
    updates=updates,
)

costs = []
print ("training rbm: %s" % self.id)
for i in range(int(epochs)):
    print ("epoch:", i)
    X = shuffle(X)
    for j in range(int(n_batches)):
        batch = X[j*batch_sz:(j*batch_sz + batch_sz)]
        train_op(batch)
        the_cost = cost_op(X)  # technically we could also get the cost for Xtes
        print ("j / n_batches:", j, "/", n_batches, "cost:", the_cost)
        costs.append(the_cost)
```

```python
        if show_fig:
            plt.plot(costs)
            plt.show()

    def free_energy(self, V):
        return  -V.dot(self.b) - T.sum(T.log(1 + T.exp(V.dot(self.W) + self.c)), axis=1)

    def sample_h_given_v(self, V):
        p_h_given_v = T.nnet.sigmoid(V.dot(self.W) + self.c)
        h_sample = self.rng.binomial(size=p_h_given_v.shape, n=1, p=p_h_given_v)
        return h_sample

    def sample_v_given_h(self, H):
        p_v_given_h = T.nnet.sigmoid(H.dot(self.W.T) + self.b)
        v_sample = self.rng.binomial(size=p_v_given_h.shape, n=1, p=p_v_given_h)
        return v_sample

    def forward_hidden(self, X):
        return T.nnet.sigmoid(X.dot(self.W) + self.c)

    def forward_output(self, X):
        Z = self.forward_hidden(X)
        Y = T.nnet.sigmoid(Z.dot(self.W.T) + self.b)
        return Y

    @staticmethod
    def createFromArrays(W, c, b, an_id):
        rbm = AutoEncoder(W.shape[1], an_id)
        rbm.W = theano.shared(W, 'W_%s' % rbm.id)
        rbm.c = theano.shared(c, 'c_%s' % rbm.id)
        rbm.b = theano.shared(b, 'b_%s' % rbm.id)
        rbm.params = [rbm.W, rbm.c, rbm.b]
        rbm.forward_params = [rbm.W, rbm.c]
        return rbm
```

2 Running the model

In [ ]:
```python
Xtrain, Ytrain, Xtest, Ytest = getKaggleMNIST()
dnn = DNN([1000, 750, 500], UnsupervisedModel=RBM)
dnn.fit(Xtrain, Ytrain, Xtest, Ytest, epochs=3)

# we compare with no pretraining in autoencoder.py
```

```
training rbm: 0
epoch: 0
j / n_batches: 0 / 410.0 cost: 151.3776596875377
j / n_batches: 1 / 410.0 cost: 146.3203703878674
```

```
j / n_batches: 2 / 410.0 cost: 144.0130903707576
j / n_batches: 3 / 410.0 cost: 141.07134167507732
j / n_batches: 4 / 410.0 cost: 131.94823864864358
j / n_batches: 5 / 410.0 cost: 123.84549220651303
j / n_batches: 6 / 410.0 cost: 119.59237113632864
j / n_batches: 7 / 410.0 cost: 123.28984382905686
j / n_batches: 8 / 410.0 cost: 118.12851764677872
j / n_batches: 9 / 410.0 cost: 112.45371540827828
j / n_batches: 10 / 410.0 cost: 110.9067863097084
j / n_batches: 11 / 410.0 cost: 110.19866330603207
j / n_batches: 12 / 410.0 cost: 110.03259367905912
j / n_batches: 13 / 410.0 cost: 112.39211017898299
j / n_batches: 14 / 410.0 cost: 113.12931414052512
j / n_batches: 15 / 410.0 cost: 108.79558595584376
j / n_batches: 16 / 410.0 cost: 107.86973196080658
j / n_batches: 17 / 410.0 cost: 105.73634228737234
j / n_batches: 18 / 410.0 cost: 103.64365574290981
j / n_batches: 19 / 410.0 cost: 102.65084480521368
j / n_batches: 20 / 410.0 cost: 102.48990928765538
j / n_batches: 21 / 410.0 cost: 102.12505311916534
j / n_batches: 22 / 410.0 cost: 100.36028769203105
j / n_batches: 23 / 410.0 cost: 101.42896768063692
j / n_batches: 24 / 410.0 cost: 102.01128874048509
j / n_batches: 25 / 410.0 cost: 102.57346845106761
j / n_batches: 26 / 410.0 cost: 98.54240410170145
j / n_batches: 27 / 410.0 cost: 94.793740254612
j / n_batches: 28 / 410.0 cost: 96.4194458473525
j / n_batches: 29 / 410.0 cost: 94.60026655836103
j / n_batches: 30 / 410.0 cost: 92.66180681321798
j / n_batches: 31 / 410.0 cost: 90.08190786362361
j / n_batches: 32 / 410.0 cost: 90.82954826412536
j / n_batches: 33 / 410.0 cost: 91.18032152495674
j / n_batches: 34 / 410.0 cost: 92.28782574501318
j / n_batches: 35 / 410.0 cost: 92.63239986795547
j / n_batches: 36 / 410.0 cost: 88.7879184283401
j / n_batches: 37 / 410.0 cost: 85.97303960397016
j / n_batches: 38 / 410.0 cost: 85.34528094920756
j / n_batches: 39 / 410.0 cost: 86.14152553300487
j / n_batches: 40 / 410.0 cost: 85.81230032168082
j / n_batches: 41 / 410.0 cost: 84.0454884496328
j / n_batches: 42 / 410.0 cost: 83.78322600614543
j / n_batches: 43 / 410.0 cost: 83.76702462833795
j / n_batches: 44 / 410.0 cost: 82.85490499269224
j / n_batches: 45 / 410.0 cost: 81.82107533822452
j / n_batches: 46 / 410.0 cost: 81.25653995022229
j / n_batches: 47 / 410.0 cost: 81.50515431808796
j / n_batches: 48 / 410.0 cost: 80.89484318319732
j / n_batches: 49 / 410.0 cost: 79.81189855581157
```

```
j / n_batches: 50 / 410.0 cost: 78.58236551592768
j / n_batches: 51 / 410.0 cost: 79.99382679592873
j / n_batches: 52 / 410.0 cost: 78.92239898944214
j / n_batches: 53 / 410.0 cost: 77.6884568459571
j / n_batches: 54 / 410.0 cost: 77.39514208017295
j / n_batches: 55 / 410.0 cost: 77.14803895129907
j / n_batches: 56 / 410.0 cost: 77.10328636240396
j / n_batches: 57 / 410.0 cost: 76.84329861404301
j / n_batches: 58 / 410.0 cost: 76.13903574219515
j / n_batches: 59 / 410.0 cost: 75.64759198366802
j / n_batches: 60 / 410.0 cost: 76.00313417644169
j / n_batches: 61 / 410.0 cost: 76.2451171979586
j / n_batches: 62 / 410.0 cost: 76.89035536752536
j / n_batches: 63 / 410.0 cost: 74.68514485392267
j / n_batches: 64 / 410.0 cost: 74.88430346142448
j / n_batches: 65 / 410.0 cost: 75.18728474449208
j / n_batches: 66 / 410.0 cost: 75.10692029371585
j / n_batches: 67 / 410.0 cost: 74.32099989030935
j / n_batches: 68 / 410.0 cost: 74.50556956269494
j / n_batches: 69 / 410.0 cost: 73.8969558157525
j / n_batches: 70 / 410.0 cost: 73.55031933357452
j / n_batches: 71 / 410.0 cost: 73.35737566791092
j / n_batches: 72 / 410.0 cost: 72.39862164052076
j / n_batches: 73 / 410.0 cost: 71.98151735913791
j / n_batches: 74 / 410.0 cost: 72.39229984051752
j / n_batches: 75 / 410.0 cost: 72.03589800636026
j / n_batches: 76 / 410.0 cost: 71.52073488856162
j / n_batches: 77 / 410.0 cost: 70.57546377965006
j / n_batches: 78 / 410.0 cost: 70.28292817146436
j / n_batches: 79 / 410.0 cost: 70.4126154584682
j / n_batches: 80 / 410.0 cost: 69.87195077732586
j / n_batches: 81 / 410.0 cost: 70.21282453330231
j / n_batches: 82 / 410.0 cost: 70.49262178584007
j / n_batches: 83 / 410.0 cost: 69.73779972914286
j / n_batches: 84 / 410.0 cost: 69.4003675804518
j / n_batches: 85 / 410.0 cost: 69.17350755533698
j / n_batches: 86 / 410.0 cost: 68.44149068909599
j / n_batches: 87 / 410.0 cost: 68.41703551578418
j / n_batches: 88 / 410.0 cost: 68.91637432880641
j / n_batches: 89 / 410.0 cost: 68.2986972217035
j / n_batches: 90 / 410.0 cost: 67.44941591695375
j / n_batches: 91 / 410.0 cost: 68.03320608065064
j / n_batches: 92 / 410.0 cost: 68.76191703327322
j / n_batches: 93 / 410.0 cost: 67.47260463241848
j / n_batches: 94 / 410.0 cost: 67.36189006887926
j / n_batches: 95 / 410.0 cost: 67.10988190121316
j / n_batches: 96 / 410.0 cost: 67.01273133422565
j / n_batches: 97 / 410.0 cost: 66.70960304537606
```

```
j / n_batches: 98 / 410.0 cost: 67.04296787036725
j / n_batches: 99 / 410.0 cost: 66.36595529000436
j / n_batches: 100 / 410.0 cost: 66.39399341264647
j / n_batches: 101 / 410.0 cost: 66.5874601288127
j / n_batches: 102 / 410.0 cost: 66.31365495044966
j / n_batches: 103 / 410.0 cost: 66.1089257994081
j / n_batches: 104 / 410.0 cost: 65.69138040162053
j / n_batches: 105 / 410.0 cost: 65.8403584239477
j / n_batches: 106 / 410.0 cost: 65.32509308678371
j / n_batches: 107 / 410.0 cost: 65.29079145467195
j / n_batches: 108 / 410.0 cost: 65.0592712630264
j / n_batches: 109 / 410.0 cost: 65.39180177396224
j / n_batches: 110 / 410.0 cost: 65.08828272863256
j / n_batches: 111 / 410.0 cost: 65.05341533362696
j / n_batches: 112 / 410.0 cost: 64.99538316001278
j / n_batches: 113 / 410.0 cost: 64.42759926721362
j / n_batches: 114 / 410.0 cost: 64.66133648207983
j / n_batches: 115 / 410.0 cost: 64.09747331037408
j / n_batches: 116 / 410.0 cost: 63.92316911921048
j / n_batches: 117 / 410.0 cost: 63.9631893573749
j / n_batches: 118 / 410.0 cost: 64.1086591391186
j / n_batches: 119 / 410.0 cost: 64.01326628543995
j / n_batches: 120 / 410.0 cost: 63.45089918980661
j / n_batches: 121 / 410.0 cost: 63.22145646292291
j / n_batches: 122 / 410.0 cost: 63.122208921868236
j / n_batches: 123 / 410.0 cost: 62.972343254152186
j / n_batches: 124 / 410.0 cost: 62.59390839777862
j / n_batches: 125 / 410.0 cost: 62.6714100398691
j / n_batches: 126 / 410.0 cost: 62.83067366601113
j / n_batches: 127 / 410.0 cost: 62.37450998334174
j / n_batches: 128 / 410.0 cost: 62.46334260556317
j / n_batches: 129 / 410.0 cost: 62.46716269583546
j / n_batches: 130 / 410.0 cost: 62.09039719546151
j / n_batches: 131 / 410.0 cost: 61.9231096061737
j / n_batches: 132 / 410.0 cost: 61.90394678559922
j / n_batches: 133 / 410.0 cost: 61.756564124250346
j / n_batches: 134 / 410.0 cost: 61.90098117137344
j / n_batches: 135 / 410.0 cost: 61.44170654204548
j / n_batches: 136 / 410.0 cost: 60.998480166496606
j / n_batches: 137 / 410.0 cost: 60.97998848795834
j / n_batches: 138 / 410.0 cost: 61.126141813113975
j / n_batches: 139 / 410.0 cost: 61.02682066476232
j / n_batches: 140 / 410.0 cost: 61.011467857826005
j / n_batches: 141 / 410.0 cost: 61.29703186983515
j / n_batches: 142 / 410.0 cost: 60.846955811274206
j / n_batches: 143 / 410.0 cost: 60.51859360940882
j / n_batches: 144 / 410.0 cost: 60.4186991499901
j / n_batches: 145 / 410.0 cost: 60.31167783053003
```

```
j / n_batches: 146 / 410.0 cost: 60.24743430503883
j / n_batches: 147 / 410.0 cost: 60.37222216437856
j / n_batches: 148 / 410.0 cost: 59.850572520235616
j / n_batches: 149 / 410.0 cost: 59.77629987948818
j / n_batches: 150 / 410.0 cost: 60.00982903496974
j / n_batches: 151 / 410.0 cost: 59.842886612464575
j / n_batches: 152 / 410.0 cost: 59.77229449556605
j / n_batches: 153 / 410.0 cost: 59.38174364693127
j / n_batches: 154 / 410.0 cost: 59.383341685069205
j / n_batches: 155 / 410.0 cost: 59.16614840334437
j / n_batches: 156 / 410.0 cost: 59.1943252541159
j / n_batches: 157 / 410.0 cost: 58.95476159564448
j / n_batches: 158 / 410.0 cost: 58.994439745641536
j / n_batches: 159 / 410.0 cost: 58.817833569536205
j / n_batches: 160 / 410.0 cost: 58.86601747475693
j / n_batches: 161 / 410.0 cost: 58.543274910464966
j / n_batches: 162 / 410.0 cost: 58.44625501975031
j / n_batches: 163 / 410.0 cost: 58.432566501640515
j / n_batches: 164 / 410.0 cost: 58.424433173652176
j / n_batches: 165 / 410.0 cost: 58.40426685857158
j / n_batches: 166 / 410.0 cost: 58.471571640986475
j / n_batches: 167 / 410.0 cost: 58.151797982513784
j / n_batches: 168 / 410.0 cost: 58.0907241283887
j / n_batches: 169 / 410.0 cost: 57.93790358601173
j / n_batches: 170 / 410.0 cost: 58.18598168478066
j / n_batches: 171 / 410.0 cost: 57.911122267277584
j / n_batches: 172 / 410.0 cost: 57.60856387974265
j / n_batches: 173 / 410.0 cost: 57.55097580549117
j / n_batches: 174 / 410.0 cost: 57.70576115294002
j / n_batches: 175 / 410.0 cost: 57.55174799777861
j / n_batches: 176 / 410.0 cost: 57.60452997298541
j / n_batches: 177 / 410.0 cost: 57.24867389403472
j / n_batches: 178 / 410.0 cost: 57.28271915531148
j / n_batches: 179 / 410.0 cost: 57.10263897193931
j / n_batches: 180 / 410.0 cost: 57.17591303247873
j / n_batches: 181 / 410.0 cost: 56.704042546667296
j / n_batches: 182 / 410.0 cost: 56.936565896009704
j / n_batches: 183 / 410.0 cost: 56.99780562773725
j / n_batches: 184 / 410.0 cost: 56.71941634323164
j / n_batches: 185 / 410.0 cost: 56.54146680652637
j / n_batches: 186 / 410.0 cost: 56.43363317482188
j / n_batches: 187 / 410.0 cost: 56.29080276212551
j / n_batches: 188 / 410.0 cost: 56.68322035761971
j / n_batches: 189 / 410.0 cost: 56.41081220926988
j / n_batches: 190 / 410.0 cost: 56.17194420373685
j / n_batches: 191 / 410.0 cost: 55.98459028579119
j / n_batches: 192 / 410.0 cost: 56.099033976951844
j / n_batches: 193 / 410.0 cost: 56.21160201000954
```

```
j / n_batches: 194 / 410.0 cost: 56.031903628173026
j / n_batches: 195 / 410.0 cost: 55.67330656030547
j / n_batches: 196 / 410.0 cost: 55.66939997771639
j / n_batches: 197 / 410.0 cost: 55.67968671075599
j / n_batches: 198 / 410.0 cost: 55.47743487261543
j / n_batches: 199 / 410.0 cost: 55.48974655260648
j / n_batches: 200 / 410.0 cost: 55.328474033387984
j / n_batches: 201 / 410.0 cost: 55.363938741657385
j / n_batches: 202 / 410.0 cost: 55.70077875520975
j / n_batches: 203 / 410.0 cost: 55.268142989436164
j / n_batches: 204 / 410.0 cost: 55.517031414949194
j / n_batches: 205 / 410.0 cost: 55.1360137784741
j / n_batches: 206 / 410.0 cost: 55.054115068214955
j / n_batches: 207 / 410.0 cost: 55.17143930302245
j / n_batches: 208 / 410.0 cost: 54.850200628271736
j / n_batches: 209 / 410.0 cost: 54.70734795050731
j / n_batches: 210 / 410.0 cost: 54.82239955072358
j / n_batches: 211 / 410.0 cost: 54.7802188401481
j / n_batches: 212 / 410.0 cost: 54.76148138295757
j / n_batches: 213 / 410.0 cost: 54.80588724233265
j / n_batches: 214 / 410.0 cost: 54.47775124350975
j / n_batches: 215 / 410.0 cost: 54.54269563790579
j / n_batches: 216 / 410.0 cost: 54.51898412692944
j / n_batches: 217 / 410.0 cost: 54.49232749846486
j / n_batches: 218 / 410.0 cost: 54.25705020624838
j / n_batches: 219 / 410.0 cost: 54.39962435299657
j / n_batches: 220 / 410.0 cost: 53.947153074187725
j / n_batches: 221 / 410.0 cost: 53.98727452345181
j / n_batches: 222 / 410.0 cost: 53.988376712254286
j / n_batches: 223 / 410.0 cost: 53.71208813910963
j / n_batches: 224 / 410.0 cost: 53.783474175265106
j / n_batches: 225 / 410.0 cost: 53.7473780214857
j / n_batches: 226 / 410.0 cost: 53.76397963218782
j / n_batches: 227 / 410.0 cost: 54.14587250832246
j / n_batches: 228 / 410.0 cost: 53.53646245656824
j / n_batches: 229 / 410.0 cost: 53.47386633296192
j / n_batches: 230 / 410.0 cost: 53.493814148067024
j / n_batches: 231 / 410.0 cost: 53.36989903824535
j / n_batches: 232 / 410.0 cost: 53.4884004682271
j / n_batches: 233 / 410.0 cost: 53.152709160917105
j / n_batches: 234 / 410.0 cost: 53.358483988831445
j / n_batches: 235 / 410.0 cost: 53.117720591222785
j / n_batches: 236 / 410.0 cost: 53.031211391278404
j / n_batches: 237 / 410.0 cost: 53.00793925699093
j / n_batches: 238 / 410.0 cost: 53.12513264762684
j / n_batches: 239 / 410.0 cost: 52.93971943361732
j / n_batches: 240 / 410.0 cost: 52.97082457007627
j / n_batches: 241 / 410.0 cost: 52.98742693531357
```

```
j / n_batches: 242 / 410.0 cost: 52.83296723055943
j / n_batches: 243 / 410.0 cost: 52.50328256220878
j / n_batches: 244 / 410.0 cost: 52.62621964341052
j / n_batches: 245 / 410.0 cost: 52.613966617511245
j / n_batches: 246 / 410.0 cost: 52.4910568080472
j / n_batches: 247 / 410.0 cost: 52.66229856280389
j / n_batches: 248 / 410.0 cost: 52.6058590602216
j / n_batches: 249 / 410.0 cost: 52.39463576371327
j / n_batches: 250 / 410.0 cost: 52.4832488339454
j / n_batches: 251 / 410.0 cost: 52.42522173027332
j / n_batches: 252 / 410.0 cost: 52.3196539211233
j / n_batches: 253 / 410.0 cost: 52.34467005491518
j / n_batches: 254 / 410.0 cost: 52.16917468795071
j / n_batches: 255 / 410.0 cost: 52.28681044296655
j / n_batches: 256 / 410.0 cost: 52.030195468966404
j / n_batches: 257 / 410.0 cost: 51.79947280131288
j / n_batches: 258 / 410.0 cost: 51.734632099323484
j / n_batches: 259 / 410.0 cost: 51.82818287449247
j / n_batches: 260 / 410.0 cost: 51.7415570118584
j / n_batches: 261 / 410.0 cost: 51.600030773726196
j / n_batches: 262 / 410.0 cost: 51.88260264686951
j / n_batches: 263 / 410.0 cost: 51.54040730469795
j / n_batches: 264 / 410.0 cost: 51.60282574005285
j / n_batches: 265 / 410.0 cost: 51.58113224404891
j / n_batches: 266 / 410.0 cost: 51.320638208998
j / n_batches: 267 / 410.0 cost: 51.60238670735033
j / n_batches: 268 / 410.0 cost: 51.523994429414955
j / n_batches: 269 / 410.0 cost: 51.50126508218576
j / n_batches: 270 / 410.0 cost: 51.34152850775843
j / n_batches: 271 / 410.0 cost: 51.368294804541456
j / n_batches: 272 / 410.0 cost: 51.16850545153071
j / n_batches: 273 / 410.0 cost: 51.367323165858686
j / n_batches: 274 / 410.0 cost: 50.96518986557511
j / n_batches: 275 / 410.0 cost: 51.110371485486034
j / n_batches: 276 / 410.0 cost: 51.2886392556261
j / n_batches: 277 / 410.0 cost: 51.18073150230208
j / n_batches: 278 / 410.0 cost: 51.01727059879746
j / n_batches: 279 / 410.0 cost: 50.82340366119247
j / n_batches: 280 / 410.0 cost: 50.89620125821198
j / n_batches: 281 / 410.0 cost: 50.768060526332356
j / n_batches: 282 / 410.0 cost: 50.70691364257907
j / n_batches: 283 / 410.0 cost: 50.80412326302381
j / n_batches: 284 / 410.0 cost: 50.63010140776181
j / n_batches: 285 / 410.0 cost: 50.54502635953463
j / n_batches: 286 / 410.0 cost: 50.60696155714939
j / n_batches: 287 / 410.0 cost: 50.72731133484913
j / n_batches: 288 / 410.0 cost: 50.72132109519392
j / n_batches: 289 / 410.0 cost: 50.43846185429943
```

```
j / n_batches: 290 / 410.0 cost: 50.36532361843853
j / n_batches: 291 / 410.0 cost: 50.37265827624017
j / n_batches: 292 / 410.0 cost: 50.39309049451842
j / n_batches: 293 / 410.0 cost: 50.46842755134993
j / n_batches: 294 / 410.0 cost: 50.40312831428866
j / n_batches: 295 / 410.0 cost: 50.32376812450454
j / n_batches: 296 / 410.0 cost: 50.24651618956513
j / n_batches: 297 / 410.0 cost: 50.19095887126709
j / n_batches: 298 / 410.0 cost: 50.14971268225381
j / n_batches: 299 / 410.0 cost: 50.396053607967836
j / n_batches: 300 / 410.0 cost: 50.03621053289991
j / n_batches: 301 / 410.0 cost: 50.06503242488024
j / n_batches: 302 / 410.0 cost: 49.97956976421061
j / n_batches: 303 / 410.0 cost: 49.98046664867321
j / n_batches: 304 / 410.0 cost: 49.76224029541848
j / n_batches: 305 / 410.0 cost: 49.796135788478345
j / n_batches: 306 / 410.0 cost: 49.94377932685702
j / n_batches: 307 / 410.0 cost: 49.75517002409964
j / n_batches: 308 / 410.0 cost: 49.71722938560757
j / n_batches: 309 / 410.0 cost: 49.55812959303953
j / n_batches: 310 / 410.0 cost: 49.8190385116795
j / n_batches: 311 / 410.0 cost: 49.73654572124171
j / n_batches: 312 / 410.0 cost: 49.56610969821444
j / n_batches: 313 / 410.0 cost: 49.56031301799833
j / n_batches: 314 / 410.0 cost: 49.6088500013884
j / n_batches: 315 / 410.0 cost: 49.58400471752708
j / n_batches: 316 / 410.0 cost: 49.27239496737656
j / n_batches: 317 / 410.0 cost: 49.476955029826684
j / n_batches: 318 / 410.0 cost: 49.31032043943133
j / n_batches: 319 / 410.0 cost: 49.27035640379168
j / n_batches: 320 / 410.0 cost: 49.35556034610288
j / n_batches: 321 / 410.0 cost: 49.235656651346204
j / n_batches: 322 / 410.0 cost: 49.35011980092361
j / n_batches: 323 / 410.0 cost: 49.08581674154562
j / n_batches: 324 / 410.0 cost: 49.058878983291045
j / n_batches: 325 / 410.0 cost: 48.95649441385096
j / n_batches: 326 / 410.0 cost: 49.07293571564556
j / n_batches: 327 / 410.0 cost: 48.954846431369475
j / n_batches: 328 / 410.0 cost: 49.094170496547726
j / n_batches: 329 / 410.0 cost: 48.93395157567474
j / n_batches: 330 / 410.0 cost: 48.93236347209904
j / n_batches: 331 / 410.0 cost: 48.96560053306892
j / n_batches: 332 / 410.0 cost: 48.83898159110868
j / n_batches: 333 / 410.0 cost: 48.87194193452536
j / n_batches: 334 / 410.0 cost: 48.71610670115403
j / n_batches: 335 / 410.0 cost: 48.718550777769586
j / n_batches: 336 / 410.0 cost: 48.68512216046242
j / n_batches: 337 / 410.0 cost: 48.612533329026725
```

```
j / n_batches: 338 / 410.0 cost: 48.717504906218096
j / n_batches: 339 / 410.0 cost: 48.671644074534036
j / n_batches: 340 / 410.0 cost: 48.64913318784037
j / n_batches: 341 / 410.0 cost: 48.37266792546684
j / n_batches: 342 / 410.0 cost: 48.554328622722714
j / n_batches: 343 / 410.0 cost: 48.5274316967094
j / n_batches: 344 / 410.0 cost: 48.257091972224764
j / n_batches: 345 / 410.0 cost: 48.446758350819366
j / n_batches: 346 / 410.0 cost: 48.34831530724293
j / n_batches: 347 / 410.0 cost: 48.22762589638171
j / n_batches: 348 / 410.0 cost: 48.32231358754944
j / n_batches: 349 / 410.0 cost: 48.21987234249249
j / n_batches: 350 / 410.0 cost: 48.21856585952094
j / n_batches: 351 / 410.0 cost: 48.138778765299016
j / n_batches: 352 / 410.0 cost: 47.96872567248692
j / n_batches: 353 / 410.0 cost: 48.1562961859051
j / n_batches: 354 / 410.0 cost: 48.03106807347
j / n_batches: 355 / 410.0 cost: 48.01342470724397
j / n_batches: 356 / 410.0 cost: 48.04965579592083
j / n_batches: 357 / 410.0 cost: 48.097759683810644
j / n_batches: 358 / 410.0 cost: 48.18873820272363
j / n_batches: 359 / 410.0 cost: 48.2411126636832
j / n_batches: 360 / 410.0 cost: 47.933619439579516
j / n_batches: 361 / 410.0 cost: 47.83689573075391
j / n_batches: 362 / 410.0 cost: 47.81176115928632
j / n_batches: 363 / 410.0 cost: 47.881720016971144
j / n_batches: 364 / 410.0 cost: 47.9172713058981
j / n_batches: 365 / 410.0 cost: 47.84352180686191
j / n_batches: 366 / 410.0 cost: 47.72626249516508
j / n_batches: 367 / 410.0 cost: 47.719693244330124
j / n_batches: 368 / 410.0 cost: 47.524562284680144
j / n_batches: 369 / 410.0 cost: 47.61255359048482
j / n_batches: 370 / 410.0 cost: 47.54708882913323
j / n_batches: 371 / 410.0 cost: 47.49876779600961
j / n_batches: 372 / 410.0 cost: 47.59356307696017
j / n_batches: 373 / 410.0 cost: 47.513294678482396
j / n_batches: 374 / 410.0 cost: 47.464106175807316
j / n_batches: 375 / 410.0 cost: 47.45604277158653
j / n_batches: 376 / 410.0 cost: 47.56071923303725
j / n_batches: 377 / 410.0 cost: 47.41575172722621
j / n_batches: 378 / 410.0 cost: 47.39766448707246
j / n_batches: 379 / 410.0 cost: 47.2901672191055
j / n_batches: 380 / 410.0 cost: 47.292030137127334
j / n_batches: 381 / 410.0 cost: 47.18447330784046
j / n_batches: 382 / 410.0 cost: 47.277544701644885
j / n_batches: 383 / 410.0 cost: 47.307395024593454
j / n_batches: 384 / 410.0 cost: 47.22363646400183
j / n_batches: 385 / 410.0 cost: 47.095948737373696
```

```
j / n_batches: 386 / 410.0 cost: 47.26151341638922
j / n_batches: 387 / 410.0 cost: 47.18588497425828
j / n_batches: 388 / 410.0 cost: 47.139716208591665
j / n_batches: 389 / 410.0 cost: 47.2957177365111
j / n_batches: 390 / 410.0 cost: 47.124187513832524
j / n_batches: 391 / 410.0 cost: 47.03306240543175
j / n_batches: 392 / 410.0 cost: 46.90538872244514
j / n_batches: 393 / 410.0 cost: 47.05993043179883
j / n_batches: 394 / 410.0 cost: 46.924255970529984
j / n_batches: 395 / 410.0 cost: 47.01565387282243
j / n_batches: 396 / 410.0 cost: 46.899130644736154
j / n_batches: 397 / 410.0 cost: 46.90656480605579
j / n_batches: 398 / 410.0 cost: 46.9529389546979
j / n_batches: 399 / 410.0 cost: 46.81654791782115
j / n_batches: 400 / 410.0 cost: 46.72651371614938
j / n_batches: 401 / 410.0 cost: 46.7581294007604
j / n_batches: 402 / 410.0 cost: 46.86960804929649
j / n_batches: 403 / 410.0 cost: 46.820636592277665
j / n_batches: 404 / 410.0 cost: 46.78547556536643
j / n_batches: 405 / 410.0 cost: 46.70103588623543
j / n_batches: 406 / 410.0 cost: 46.61354097398657
j / n_batches: 407 / 410.0 cost: 46.63190471206064
j / n_batches: 408 / 410.0 cost: 46.424686523379975
j / n_batches: 409 / 410.0 cost: 46.58768137917598
training rbm: 1
epoch: 0
j / n_batches: 0 / 410.0 cost: 210.65342434630273
```

3 RBM With TensorFlow

3.1 Import the appropriate modules

```
In [ ]: from __future__ import print_function, division
        from builtins import range
        import numpy as np
        import tensorflow as tf
        import matplotlib.pyplot as plt
        from sklearn.utils import shuffle
        from util import getKaggleMNIST
        from autoencoder_tf import DNNclass RBM(object):
```

3.2 RBM Class

```
In [5]: class RBM(object):
            def __init__(self, D, M, an_id):
                self.D = D
                self.M = M
```

```
        self.id = an_id
        self.build(D, M)

    def set_session(self, session):
        self.session = session

    def build(self, D, M):
        # params
        self.W = tf.Variable(tf.random_normal(shape=(D, M)) * np.sqrt(2.0 / M))
        # note: without limiting variance, you get numerical stability issues
        self.c = tf.Variable(np.zeros(M).astype(np.float32))
        self.b = tf.Variable(np.zeros(D).astype(np.float32))

        # data
        self.X_in = tf.placeholder(tf.float32, shape=(None, D))

        # conditional probabilities
        # NOTE: tf.contrib.distributions.Bernoulli API has changed in Tensorflow v1.2
        V = self.X_in
        p_h_given_v = tf.nn.sigmoid(tf.matmul(V, self.W) + self.c)
        self.p_h_given_v = p_h_given_v # save for later
        # self.rng_h_given_v = tf.contrib.distributions.Bernoulli(
        #     probs=p_h_given_v,
        #     dtype=tf.float32
        # )
        r = tf.random_uniform(shape=tf.shape(p_h_given_v))
        H = tf.to_float(r < p_h_given_v)

        p_v_given_h = tf.nn.sigmoid(
            tf.matmul(H, tf.transpose(self.W)) + self.b)
        # self.rng_v_given_h = tf.contrib.distributions.Bernoulli(
        #     probs=p_v_given_h,
        #     dtype=tf.float32
        # )
        r = tf.random_uniform(shape=tf.shape(p_v_given_h))
        X_sample = tf.to_float(r < p_v_given_h)


        # build the objective
        objective = tf.reduce_mean(
            self.free_energy(self.X_in)) - tf.reduce_mean(self.free_energy(X_sample))
        self.train_op = tf.train.AdamOptimizer(1e-2).minimize(objective)
        # self.train_op = tf.train.GradientDescentOptimizer(1e-3).minimize(objective)

        # build the cost
        # we won't use this to optimize the model parameters
        # just to observe what happens during training
        logits = self.forward_logits(self.X_in)
```

```python
        self.cost = tf.reduce_mean(
            tf.nn.sigmoid_cross_entropy_with_logits(
                labels=self.X_in,
                logits=logits,
            )
        )

    def fit(self, X, epochs=1, batch_sz=100, show_fig=False):
        N, D = X.shape
        n_batches = N // batch_sz

        costs = []
        print("training rbm: %s" % self.id)
        for i in range(epochs):
            print("epoch:", i)
            X = shuffle(X)
            for j in range(n_batches):
                batch = X[j*batch_sz:(j*batch_sz + batch_sz)]
                _, c = self.session.run(
                    (self.train_op, self.cost), feed_dict={self.X_in: batch})
                if j % 10 == 0:
                    print("j / n_batches:", j, "/", n_batches, "cost:", c)
                costs.append(c)
        if show_fig:
            plt.plot(costs)
            plt.show()

    def free_energy(self, V):
        b = tf.reshape(self.b, (self.D, 1))
        first_term = -tf.matmul(V, b)
        first_term = tf.reshape(first_term, (-1,))

        second_term = -tf.reduce_sum(
            # tf.log(1 + tf.exp(tf.matmul(V, self.W) + self.c)),
            tf.nn.softplus(tf.matmul(V, self.W) + self.c),
            axis=1
        )

        return first_term + second_term

    def forward_hidden(self, X):
        return tf.nn.sigmoid(tf.matmul(X, self.W) + self.c)

    def forward_logits(self, X):
        Z = self.forward_hidden(X)
        return tf.matmul(Z, tf.transpose(self.W)) + self.b

    def forward_output(self, X):
```

```
            return tf.nn.sigmoid(self.forward_logits(X))

        def transform(self, X):
            # accepts and returns a real numpy array
            # unlike forward_hidden and forward_output
            # which deal with tensorflow variables
            return self.session.run(self.p_h_given_v, feed_dict={self.X_in: X})
```

3.3 Running the model

```
In [ ]: Xtrain, Ytrain, Xtest, Ytest = getKaggleMNIST()

        # same as autoencoder_tf.py
        Xtrain = Xtrain.astype(np.float32)
        Xtest = Xtest.astype(np.float32)
        _, D = Xtrain.shape
        K = len(set(Ytrain))
        dnn = DNN(D, [1000, 750, 500], K, UnsupervisedModel=RBM)
        init_op = tf.global_variables_initializer()
        with tf.Session() as session:
            session.run(init_op)
            dnn.set_session(session)
            dnn.fit(Xtrain, Ytrain, Xtest, Ytest, pretrain=True, epochs=10)
```