COMP 790-125, HW4

Calvin Deutschbein

December 7, 2015

Deadline: 12/08/2015 11:59PM EST
Submit `hw4.pdf` by e-mail, `mailto:vjojic+comp790+hw4@cs.unc.edu`.

We will train Restricted Boltzmann Machines. As before we will develop things from ground up.

Preliminaries You will need following equations

$$p(\mathbf{h}, \mathbf{v}) = p(h_k, \mathbf{h}_{[-k]}, \mathbf{v}) \tag{1}$$

$$p(a|b) = \frac{p(a,b)}{p(b)} \tag{2}$$

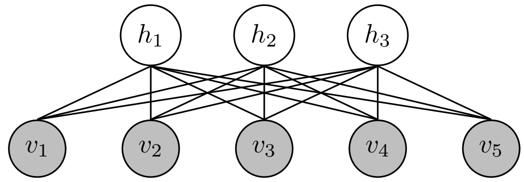
$$p(a,c) = \sum_b p(a,b,c) \tag{3}$$

$$\sum_b \frac{\exp\{f(a)+g(b)\}}{Z} = \frac{\exp\{f(a)\}}{Z} \sum_b \exp\{g(b)\} \tag{4}$$

$$\sum_a \sum_b \frac{\exp\{f(a)+g(b)\}}{Z} = \sum_a \frac{\exp\{f(a)\}}{Z} \sum_b \exp\{g(b)\} \tag{5}$$

$$\tag{6}$$

Restricted Boltzmann Machine An RBM is characterized by two sets of nodes, visible and hidden, and each edge that connect a visible and a hidden node. There are no edges between visible nodes. There are no edges between hidden nodes. Visually we can organize visible nodes into a bottom layer, and hidden nodes into a top layer. In this visualization, the requirement that each edge connects a visible and a hidden node, corresponds to absence of edges within layer.



Energy of the RBM is given by

$$E(\mathbf{h},\mathbf{v}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{v}^T\Theta\mathbf{h} = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i\theta_{i,j}h_j. \quad (7)$$

Note that energy can be positive and negative in this setting, depending on the sign of the parameters $\mathbf{a}, \mathbf{b}, \Theta$.

The distribution over states of this network

$$p(\mathbf{h},\mathbf{v}|\Theta,\mathbf{a},\mathbf{b}) = \frac{\exp\{-E(\mathbf{h},\mathbf{v})\}}{\sum_{\mathbf{h}',\mathbf{v}'}\exp\{-E(\mathbf{h}',\mathbf{v}')\}} = \frac{\exp\{-E(\mathbf{h},\mathbf{v})\}}{Z} \quad (8)$$

Several equations that will be helpful

$$\begin{aligned}
\sum_{\mathbf{h},\mathbf{v}} \frac{\exp\{-E(\mathbf{h},\mathbf{v})\} f(\mathbf{h},\mathbf{v})}{\sum_{\mathbf{h}',\mathbf{v}'}\exp\{-E(\mathbf{h}',\mathbf{v}')\}} &= \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v}) f(\mathbf{h},\mathbf{v}) && (9)\\
\nabla \log \sum_{\mathbf{h},\mathbf{v}} \exp\{-E(\mathbf{h},\mathbf{v})\} &= \frac{\sum_{\mathbf{h},\mathbf{v}}\exp\{-E(\mathbf{h},\mathbf{v})\}\nabla(-E(\mathbf{h},\mathbf{v}))}{\sum_{\mathbf{h}',\mathbf{v}'}\exp\{-E(\mathbf{h}',\mathbf{v}')\}}\\
&= \frac{\sum_{\mathbf{h},\mathbf{v}}\exp\{-E(\mathbf{h},\mathbf{v})\}\nabla(-E(\mathbf{h},\mathbf{v}))}{Z}\\
&= \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v})\nabla E(\mathbf{h},\mathbf{v}) && (10)\\
\nabla \log \sum_{\mathbf{h}} \exp\{-E(\mathbf{h},\mathbf{v})\} &= \frac{\sum_{\mathbf{h}}\exp\{-E(\mathbf{h},\mathbf{v})\}\nabla(-E(\mathbf{h},\mathbf{v}))}{\sum_{\mathbf{h}'}\exp\{-E(\mathbf{h}',\mathbf{v})\}}\\
&= \frac{\frac{\sum_{\mathbf{h}}\exp\{-E(\mathbf{h},\mathbf{v})\}}{Z}\nabla(-E(\mathbf{h},\mathbf{v}))}{\frac{\sum_{\mathbf{h}'}\exp\{-E(\mathbf{h}',\mathbf{v})\}}{Z}}\\
&= \frac{\sum_{\mathbf{h}} p(\mathbf{h},\mathbf{v})\nabla(-E(\mathbf{h},\mathbf{v}))}{\sum_{\mathbf{h}'} p(\mathbf{h}',\mathbf{v})}\\
&= \frac{\sum_{\mathbf{h}} p(\mathbf{h},\mathbf{v})\nabla(-E(\mathbf{h},\mathbf{v}))}{p(\mathbf{v})}\\
&= \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\nabla(-E(\mathbf{h},\mathbf{v})) && (11)
\end{aligned}$$

Problem 1(2pt) We will use the bipartite structure of the graph to show that each h_j is independent of the rest given $\mathbf{v}$. Specifically, we will show

$$p(h_k|\mathbf{v},\mathbf{h}_{[-k]},\Theta,\mathbf{a},\mathbf{b}) = p(h_k|\mathbf{v},\theta_{:,k},\mathbf{b})$$

We will use the fully expanded energy:

$$E(\mathbf{h},\mathbf{v}) = -\left(\sum_i a_i v_i\right) - \left(\sum_j b_j h_j\right) - \sum_{i,j} v_i\theta_{i,j}h_j$$

Organize the terms of the negative energy into those that depend on h_k and those that do not

$$\begin{aligned} -E(\mathbf{h}, \mathbf{v}) &= f(h_k, \mathbf{v}) + g(\mathbf{h}_{[-k]}, \mathbf{v}) && (12) \\ f(h_k, \mathbf{v}) &= -b_k h_k - \sum_i v_i \theta_{i,k} h_k && (13) \\ g(\mathbf{h}_{[-k]}, \mathbf{v}) &= -\left(\sum_i a_i v_i \right) - \left(\sum_{j \setminus k} b_j h_j \right) - \sum_{i,j \setminus k} v_i \theta_{i,j} h_j && (14) \end{aligned}$$

Using Equations 8 and 12, express $p(\mathbf{h}, \mathbf{v})$ using f and g. You can leave denominator as Z.

$$p(\mathbf{h}, \mathbf{v}) = \frac{\exp\left\{ f(h_k, \mathbf{v}) + g(\mathbf{h}_{[-k]}, \mathbf{v}) \right\}}{Z} \tag{15}$$

Use Bayes' rule, Equation 2, to express $p(h_k|\mathbf{v})$ in terms of $p(h_k, \mathbf{v})$ and $p(\mathbf{v})$

$$p(h_k|\mathbf{v}) = \frac{p(h_k, \mathbf{v})}{p(\mathbf{v})} \tag{16}$$

Express $p(h_k, \mathbf{v})$ using Equations 1,3. Then use Equation 15 and 4

$$p(h_k, \mathbf{v}) = \sum_{\mathbf{h}_{[-k]}} p(h_k, \mathbf{h}_{[-k]}, \mathbf{b}) = \frac{\exp\left\{ f(h_k, \mathbf{v}) \right\}}{Z} \sum_{\mathbf{h}_{[-k]}} g(\mathbf{h}_{[-k]}, \mathbf{v}) \tag{17}$$

Express $p(\mathbf{v})$ by marginalizing out $\mathbf{h}$ of $p(\mathbf{h}, \mathbf{v})$, then expand $p(\mathbf{h}, \mathbf{v})$ using Equation 15

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{\exp\left\{ f(h_k, \mathbf{v}) + g(\mathbf{h}_{[-k]}, \mathbf{v}) \right\}}{Z}$$

Use Equation 5 to reorganize $p(\mathbf{v})$

$$p(\mathbf{v}) = \sum_{h_k} \frac{\exp\left\{ f(h_k, \mathbf{v}) \right\}}{Z} \sum_{\mathbf{h}_{[-k]}} g(\mathbf{h}_{[-k]}, \mathbf{v}) \tag{18}$$

Use Equation 16, 17, 18 to express conditional probability

$$p(h_k|\mathbf{v}) = \frac{\frac{\exp\{f(h_k, \mathbf{v})\}}{Z} \sum_{\mathbf{h}_{[-k]}} g(\mathbf{h}_{[-k]}, \mathbf{v})}{\sum_{h_k} \frac{\exp\{f(h_k, \mathbf{v})\}}{Z} \sum_{\mathbf{h}_{[-k]}} g(\mathbf{h}_{[-k]}, \mathbf{v})}$$

Cancel out terms and expand $f(h_k)$

$$p(h_k|\mathbf{v}) = \frac{\exp\left\{ -b_k h_k - \sum_i v_i \theta_{i,k} h_k \right\}}{\sum_{h_j} \exp\left\{ -b_k h_j - \sum_i v_i \theta_{i,j} h_j \right\}}$$

Use the fact that $h_k \in \{0, 1\}$ and write out this conditional probability as a sigmoid

$$p(h_k|\mathbf{v}) = \sigma\left(\prod_{h_j \neq h_k} h_j (b_j + \sum_i v_i \theta_{i,j}) \right) \tag{19}$$

where

$$\sigma(z) = \frac{1}{1 + \exp\{-z\}}$$

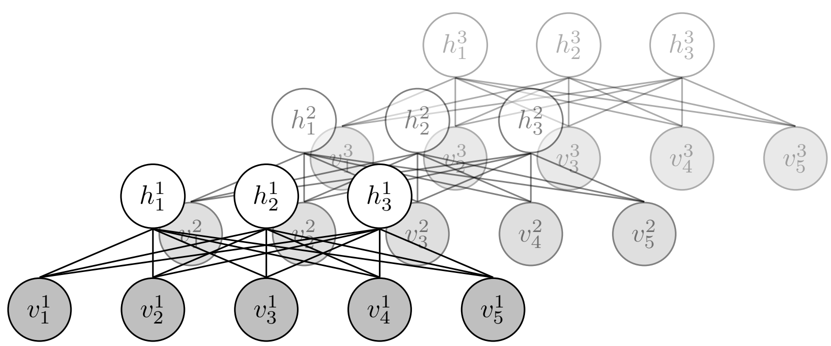
Use the fact that the energy has a symmetric form for visible and hidden variables and obtain conditional probability

$$p(v_l|\mathbf{h}) = \sigma\left(\prod_{v_i \neq v_l} v_i(a_i + \sum_j \theta_{i,j} h_j)\right)$$

Note that Θ is not symmetric, $\theta_{i,j} \neq \theta_{j,i}$ in fact Θ will usually have less columns than rows, since the hidden variables compress information in the visible ones.

Problem 2(2pt) Implement sampling of conditional distribution given by Equation 19. As input you will be provided Θ matrix with m rows and n columns, vector of biases $\mathbf{c}$, and a vector of variables $\mathbf{v}$. Your code should sample all of the hidden variables at once. Note that the hidden variables are independent from each other given the visible variables. You proved this in the last problem.

```
function h = sample(Theta,cc,vv)
nvisible = size(Theta,1);
nhidden = size(Theta,2);
nsamples = size(vv,2);
assert(size(vv,1) == nhidden);
assert(size(cc,1) == nvisible);
z = Theta * vv + repmat(cc,[1 nsamples]);
h = sigmoid(z) > rand(size(z));
```



The graph above shows an RBM on 3 samples. Your code should be able to sample states for all the hidden variables in all the samples in parallel. Make sure that your code can take as an input a matrix Θ,vector $\mathbf{c}$ and a *matrix* $\mathbf{V}$. The matrix $\mathbf{V}$ is of size $p \times n$ where p is number of features in a sample, and n is the number of samples. You should be able to use matrix multiplication and `repmat` to achieve this.

Test your code by running this

```
    nv = 3;
    nh = 2;
    n = 100;
    Theta = [ -10 10; -10 -10; 10 -10];
    bb = [2;2];
    aa = [-5;+5;-5];
    hidden = rand(2,n)>0.5;
    visible = sample(Theta,aa,hidden);
    newhidden = sample(Theta',bb,visible);
    err = (sum(sum(newhidden ~= hidden))/prod(size(hidden)))
```

You should see something like this

```
err =
    0.0450
```

You can play with this code as well

```
hidden = rand(2,n)>0.5;
for it=1:1000
    visible = sample(Theta,aa,hidden);
    newhidden = sample(Theta',bb,visible);
    err = (sum(sum(newhidden ~= hidden))/prod(size(hidden)))
    hidden = newhidden;
end
hist([1 2]*hidden,[0:3])
```

The code above iterates between drawing from $p(\mathbf{h}|\mathbf{v}, \Theta, \mathbf{a}, \mathbf{b})$ and $p(\mathbf{v}|\mathbf{h}, \Theta, \mathbf{a}, \mathbf{b})$. This is an example of a block Gibbs sampler. It is called block, because we are updating blocks of variables at once. As we iterate the chain, the sample $(\mathbf{h}, \mathbf{v})$ gets closer to a draw from the distribution $p(\mathbf{h}, \mathbf{v})$.

Thus, we can implement

- sampler for $p(\mathbf{h}|\mathbf{v})$ as `hidden = sample(Theta',bb,visible);`
- sampler for $p(\mathbf{v}|\mathbf{h})$ as `visible = sample(Theta,aa,hidden);`
- joint sampler for $p(\mathbf{h}, \mathbf{v})$ by iterating

  ```
  hidden = sample(Theta',bb,visible);
  visible = sample(Theta,aa,hidden);
  ```

Problem 3(2pt)

Deriving contrastive divergence updates We will derive the maximum likelihood update from scratch.

The likelihood function for RBM for a parameter tuple $\Psi = (\Theta, \mathbf{a}, \mathbf{b})$:

$$
\begin{aligned}
\mathrm{ALL}(\Psi) &= \sum_{t=1}^{T} \frac{1}{T} \log p(\mathbf{v}^t|\Psi) = \sum_t \frac{1}{T} \log \sum_{\mathbf{h}^t} p(\mathbf{v}^t, \mathbf{h}^t|\Psi) \\
&= \sum_t \frac{1}{T} \log \sum_{\mathbf{h}^t} \frac{\exp\{-E(\mathbf{v}^t, \mathbf{h}^t)\}}{\sum_{\mathbf{v}'\mathbf{h}'} \exp\{-E(\mathbf{v}', \mathbf{h}')\}} \\
&= \sum_t \frac{1}{T} \left[\log \sum_{\mathbf{h}^t} \exp\left\{-E(\mathbf{v}^t, \mathbf{h}^t)\right\} - \sum_{t=1}^{T} \log \sum_{\mathbf{v}',\mathbf{h}'} \exp\{-E(\mathbf{v}', \mathbf{h}')\} \right] \\
&= \sum_t \frac{1}{T} \left[\underbrace{\log \sum_{\mathbf{h}^t} \exp\left\{-E(\mathbf{v}^t, \mathbf{h}^t)\right\}}_{A^t(\Psi)} - \underbrace{\log \sum_{\mathbf{v}',\mathbf{h}'} \exp\{-E(\mathbf{v}', \mathbf{h}')\}}_{B(\Psi)} \right]
\end{aligned}
$$

Hence

$$
\begin{aligned}
\nabla_\Psi \mathrm{ALL}(\Psi) &= \sum_{t=1}^{T} \frac{1}{T} \left[\nabla_\Psi A^t(\Psi) - B(\Psi) \right] && (20) \\
A^t(\Psi) &= \log \sum_{\mathbf{h}^t} \exp\left\{-E(\mathbf{v}^t, \mathbf{h}^t)\right\} && (21) \\
B(\Psi) &= \log \sum_{\mathbf{v}',\mathbf{h}'} \exp\{-E(\mathbf{v}', \mathbf{h}')\} && (22)
\end{aligned}
$$

Use Equations 9 and 11 to compute gradient of $A^t(\Psi)$

$$
\begin{aligned}
\nabla_\Psi A^t(\Psi) &= \frac{\sum_{\mathbf{h}^t} p(\mathbf{h}^t, \mathbf{v}^t) \nabla(-E(\mathbf{h}^t, \mathbf{v}^t))}{p(\mathbf{v}^t)} && (23) \\
&= \frac{1}{p(\mathbf{v}^t)} \sum_{\mathbf{h}^t} \frac{\exp\{-E(\mathbf{h}^t, \mathbf{v}^t)\} \nabla(-E(\mathbf{h}^t, \mathbf{v}^t))}{\sum_{\mathbf{h}'} \exp\{-E(\mathbf{h}', \mathbf{v}^t)\}} && (24)
\end{aligned}
$$

Use Equations 9 and 10 to compute gradient of $B(\Psi)$

$$
\begin{aligned}
\nabla_\Psi B(\Psi) &= \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h}, \mathbf{v}) \nabla E(\mathbf{h}, \mathbf{v}) \\
&= \sum_{\mathbf{h},\mathbf{v}} \frac{\exp\{-E(\mathbf{h}, \mathbf{v})\} \nabla E(\mathbf{h}, \mathbf{v})}{\sum_{\mathbf{h}',\mathbf{v}'} \exp\{-E(\mathbf{h}', \mathbf{v}')\}}
\end{aligned}
$$

Using Equations 7,24,25 compute partial derivatives with respect to $\theta_{i,j}$,a_i, and b_j.

$$
\begin{aligned}
\frac{\partial A^t(\Psi)}{\partial \theta_{i,j}} &= v_i h_j \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \\
\frac{\partial A^t(\Psi)}{\partial a_i} &= v_i \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \\
\frac{\partial A^t(\Psi)}{\partial b_j} &= h_j \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \\
\frac{\partial B(\Psi)}{\partial \theta_{i,j}} &= v_i h_j \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v}) \\
\frac{\partial B(\Psi)}{\partial a_i} &= v_i \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v}) \\
\frac{\partial B(\Psi)}{\partial b_j} &= h_j \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v}) \\
\frac{\partial \mathrm{ALL}(\Psi)}{\partial \theta_{i,j}} &= v_i h_j \sum_t \frac{1}{T}\left[\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) - \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v})\right] \\
\frac{\partial \mathrm{ALL}(\Psi)}{\partial a_i} &= v_i \sum_t \frac{1}{T}\left[\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) - \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v})\right] \\
\frac{\partial \mathrm{ALL}(\Psi)}{\partial b_i} &= h_j \sum_t \frac{1}{T}\left[\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) - \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h},\mathbf{v})\right]
\end{aligned}
$$

Problem 4(2pt) Implement contrastive divergence for gradient computation

input : $\Theta, \mathbf{a}, \mathbf{b}, \{\mathbf{v}^t : t = 1, \ldots T\}$
output: approximate gradient $\mathbf{g}$ and recon
recon $= 0$; **foreach** $t = 1, 2, \ldots T$ **do**
  Sample $\mathbf{h}^t$ from $p(\mathbf{h}|\mathbf{v}^t, \Theta, \mathbf{a}, \mathbf{b})$
  $\mathbf{g}^0_\theta = \mathbf{g}^0_\theta + \frac{1}{T}\boxed{\texttt{answer}}$; $\mathbf{g}^0_{\mathbf{a}} = \mathbf{g}^0_{\mathbf{a}} + \frac{1}{T}\boxed{\texttt{answer}}$
  $\mathbf{g}^0_{\mathbf{b}} = \mathbf{g}^0_{\mathbf{b}} + \frac{1}{T}\boxed{\texttt{answer}}$
  Sample $\mathbf{v}^{t,1}$ from $p(\mathbf{v}|\mathbf{h}^t, \Theta, \mathbf{a}, \mathbf{b})$
  Sample $\mathbf{h}^{t,1}$ from $p(\mathbf{h}|\mathbf{v}^{t,1}, \Theta, \mathbf{a}, \mathbf{b})$
  $\mathbf{g}^1_\theta = \mathbf{g}^1_\theta + \frac{1}{T}\boxed{\texttt{answer}}$
  $\mathbf{g}^1_{\mathbf{a}} = \mathbf{g}^1_{\mathbf{a}} + \frac{1}{T}\boxed{\texttt{answer}}$
  $\mathbf{g}^1_{\mathbf{b}} = \mathbf{g}^1_{\mathbf{b}} + \frac{1}{T}\boxed{\texttt{answer}}$
  recon = recon + err($\mathbf{v}^t$,$\mathbf{v}^{t,1}$)
end
$\mathbf{g} = -\mathbf{g}^0 + \mathbf{g}^1$

Note that computation for different $t = 1, ..., T$ can be performed in parallel. Hence `foreach` above is parallelizable.

```
function [bg_theta,bg_aa,bg_bb,recon] = cdgradient(Theta,aa,bb,V)
p = size(V,1);
T = size(V,2);
nhidden = size(Theta,2);
nvisible = size(Theta,1);
assert(length(aa) == nvisible)
assert(length(bb) == nhidden)
bg0_theta = zeros(size(Theta));
bg1_theta = bg0_theta;
bg0_aa = zeros(size(aa));
bg1_aa = bg0_aa;
bg0_bb = zeros(size(bb));
bg1_bb = bg0_bb;

recon = 0;
for t=1:T
    vt = V(:,t);
    ht = sample(Theta',bb,vt);
    bg0_theta = bg0_theta + ...;
    bg0_aa = bg0_aa + ...;
    bg0_bb = bg0_bb + ...;
    vt1 = sample(Theta,aa,ht);
    ht1 = sample(Theta',bb,vt1);
    bg1_theta = bg1_theta + ...;
    bg1_aa = bg1_aa + ...;
    bg1_bb = bg1_bb + ...;
    recon = recon + norm(vt1 - vt);
end
bg_theta = bg0_theta - bg1_theta;
bg_aa = bg0_aa - bg1_aa;
bg_bb = bg0_bb - bg1_bb;
```

Try following code

```
nv = 3; nh = 2;
n = 100;
% ground truth params
Theta = [ -10 10; -10 -10; 10 -10];
bb = [2;2]; aa = [-5;+5;-5];

hidden = rand(nh,n)>0.5;

% sample from p(h,v)
for it=1:100
```

```
    visible = sample(Theta,aa,hidden);
    hidden = sample(Theta',bb,visible);
end

% step size
eta = 0.05;
% momentum
mom = 0.95;
% learned parameters
lTheta = 0.1*randn(size(Theta));laa = zeros(size(aa));lbb = zeros(size(bb));
% update direction
vt = zeros(size(Theta));vaa = zeros(size(aa));vbb = zeros(size(bb));
for it=1:10000
    [gt,ga,gb,recon] = cdgradient(lTheta,laa,lbb,visible);
    vaa = mom*vaa + eta*ga; vbb = mom*vbb + eta*gb; vt = mom*vt + eta*gt;
    lTheta = lTheta + vt; laa = laa + vaa; lbb = lbb + vbb;
    if (mod(it,100) == 0)
        fprintf('Iter: %d recon: %g ',it,recon);
        fprintf('Distance of learned theta to ground truth theta: %g\n',...
                min([norm(lTheta - Theta) norm(lTheta - Theta(:,[2 1]))]))
    end
end
```

If you implemented `cdgradient` correctly you should see the distance between the learned theta `lTheta` and ground truth theta used to generate data `Theta` shrink. Note that is might bounce around a bit in the later steps, since we are keeping a fixed step size. You could try annealing the step size, *e.g.* `eta = 0.9999*eta`.

Problem 5(2pt) We are going to train an RBM on digit images ... AND MAKE A MOVIE! Download and decompress MNIST training set from:

`http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz`

Use following script to train your model

```
img = loadMNISTImages('train-images-idx3-ubyte');
lab = loadMNISTLabels('train-labels-idx1-ubyte');

nv = size(img,1);
nh = 100;
n = size(img,2);

% step size
eta = 0.1;
% momentum
mom = 0.95;
% learned parameters
lTheta = 0.1*randn(nv,nh);laa = zeros(nv,1);lbb = zeros(nh,1);
```

```
% update direction
vt = zeros(size(lTheta));vaa = zeros(size(laa));vbb = zeros(size(lbb));
minibatch = 100;
last = 0;
list = randperm(n);
ct = 0;
ITER = 1000;
d = sqrt(nv);
f = sqrt(nh);
skip = 10;
makemovie = 0;
if makemovie
    frames = zeros((d+1)*f+1,(d+1)*f+1,1,ceil(ITER/skip));
end

for it=1:ITER
    idxs = list(mod(last:last+minibatch-1,n)+1);
    last = last+minibatch;
    visible = img(:,idxs);
    eta = eta*0.9999;
    [gt,ga,gb,recon] = cdgradient(lTheta,laa,lbb,visible);
    vt = mom*vt + eta*gt;vaa = mom*vaa + eta*ga;vbb = mom*vbb + eta*gb;
    lTheta = lTheta + vt; laa = laa + vaa; lbb = lbb + vbb;
    if (mod(it,skip) == 0)
        fprintf('Iter: %d Recon: %d\n',it,recon);
        if makemovie
            ct = ct+1;
            frames(:,:,1,ct) = showfilters(lTheta);
        end
    end
end

if makemovie
    frames = frames(:,:,:,1:ct);
    frames = frames - min(frames(:));
    frames = frames./max(frames(:));
    frames = uint8(frames*255);
    mov = immovie(frames,gray(256));
    writerObj = VideoWriter('learning.mpg','MPEG-4')
    open(writerObj);
    writeVideo(writerObj,mov);
    close(writerObj);
end
```

Tune the learning rate, `eta`, momentum, `mom`, minibatch size, `minibatch`, so that the filters looks like handwritten digits. Then switch the makemovie to be 1, rerun the code, and send the resulting video along with your homework report.