CS 542 Machine Learning — Sang Peter Chin
due thursday, march 21 2019 Gradescope

HW3.vgunawan

Sparse autoencoder Write up

1. Generate training set

```
[row, col, img] = size(IMAGES); %[512,512,10]
for k=1:numpatches
    index = randi(img); %pick a random image from the 10
    % pick a random starting index for the 8x8 patch
    i = randi(row - patchsize + 1);
    j = randi(col - patchsize + 1);
    % "crop" patch from image
    patch = IMAGES(i:i+patchsize-1,j:j+patchsize-1,index);
    %combine patches
    patches(:,k) = reshape(patch, patchsize*patchsize, 1);
end
```

2. Sparse autoencoder objective
   For this step I referred to the reading.

   - For lines 4-8, the feed forward evaluation I used the equations (Sections 2.0, 2.1):

$$z^{(2)} = W^{(1)}x + b^1$$
$$a^{(2)} = f(z^2)$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^2$$
$$h_{W,b(x)} = a^{(3)} = f(z^3)$$
$$f = \frac{1}{1 + exp(-z)}$$

   - For lines 10-15, the cost function I used the equations (Section 2.2):

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}||h_{W,b}(x) - y||^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_1-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2$$

   - For lines 17- 22, the sparsity cost I used the equations(Section 3):

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(2)}(x^{(i)})\right]$$

$$\sum_{j=1}^{s_2} KL(\rho||\hat{\rho}_j) = \sum_{j=1}^{s_2}\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}$$

   - In line 25, the overall cost is given by the equation:

$$J_{sparse}(W,b) = J(W,B) + \beta\sum_{j=1}^{s_2} KL(\rho||\hat{\rho}_j)$$

- For lines 27-31, the backpropagation I used the equations(Section 2,0,2.2,3):

$$f'(z) = f(z)(1 - f(z))$$

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} ||y - h_{W,b}(x)||^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

$$\delta_i^{(2)} = \left(\left(\sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)} \right) + \beta \left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i} \right) \right) f'(z_i^{(2)})$$

- For lines 33-37, calculating the gradients I used the equations (Section 2.2):

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T$$

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}$$

$$\Delta W^{(l)} = \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$$

$$\Delta b^{(l)} = \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$$

```
1   m = size(data,2);
2   x = data;
3
4   %Feedforward Evaluation
5   z2 = W1*x + repmat(b1, 1, m);  %net
6   a2 = sigmoid(z2); %out
7   z3 = W2*a2 + repmat(b2, 1, m); %net
8   a3 = sigmoid(z3); %out
9
10  %Mean Squared Error (MSE) Cost (Equation 8)
11
12      JWb = sum(sum((x-a3).^2, 1)) / (2*m); %1st term, average sum-of-squares error
13
14      %Regularization Cost
15      weightdecay = (lambda/2) * (sum(sum(W1.^2)) + sum(sum(W2.^2))); % 2nd term, weight decay
16
17  %Sparsity Cost
18
19  rhohat= sum(a2, 2)/m; %average activation of hidden units
20  KL = sparsityParam.*log(sparsityParam./rhohat)+...
21          (1-sparsityParam).*log((1-sparsityParam)./(1-rhohat)); %Kullback-Leiber divergence
22  penalty = sum(KL);
23
24  %Overall cost
25  cost = JWb + weightdecay + beta*penalty;
26
27  %Backpropagation
28  d3 = -(x - a3) .* (a3.*(1-a3));
29  d2 = (W2'*d3 + repmat(beta*(-sparsityParam./rhohat+...
30          (1-sparsityParam)./(1-rhohat)), 1, m) ).* (a2.*(1-a2));
31      %gradient value for a single weight value relative to a single training example
32
33  %Calculate gradient, (Page 9, step 2)
```

```
    W1grad = (d2 * x')/m + lambda*W1;
    W2grad = (d3 * a2')/m + lambda*W2;
    b1grad = sum(d2,2)/m;
    b2grad = sum(d3,2)/m;
```

3. Gradient checking
   I used the equation(section2.3):

$$g(\theta) \approx \frac{J(\theta + EPSILON) - J(\theta - EPSILON)}{2 \times EPSILON}$$

```
EPSILON = 1e-4;

for i=1:length(theta)
    t_plus = theta;
    t_minus = theta;
    t_plus(i) = t_plus(i) + EPSILON;
    t_minus(i) = t_minus(i) - EPSILON;
    numgrad(i) = (J(t_plus) - J(t_minus))/(2*EPSILON);
end
```

4. Train the sparse autoencoder
   minimize J_{sparse} with respect to parameters.

5. Visualization
   Try to understand what activation function of a hidden unit does.

$$a_i^{(2)} = f\left(\sum_{j=1}^{100} W_{ij}^{(1)} x_j + b_i^{(1)}\right)$$

   What input image x causes hidden unit i to be maximally active(close to 1, sigmoid function)?
   With constrain $||x||^2 = \sum_{i=1}^{100} x_i^2 \leq 1$, so output is not close to 1 because of an infinitely large or an infinitely small x.
   Visualizing it by setting pixel x_j to

$$x_j = \frac{W_{ij}^{(l)}}{\sqrt{\sum_{j=1}^{100} (W_{ij}^{(l)})^2}}$$

   For the 25 hidden units, we have the result below (”edge detectors”):