partial codes in week_5 code assignment.

> written by *VincentX3*, Nov.20.18
> Code assignment this week is quite diffcult.Not only because it demands many matrix operation skills but also because its algorithm (back propagation) is mysterious.

sigmoidGradient.m

```
g=sigmoid(z).*(1-sigmoid(z));
```

randInitializeWeights.m

```
% Randomly initialize the weights to small values
epsilon_init = 0.12;
W = rand(L_out, 1 + L_in) * 2 *
epsilon_init-epsilon_init;
```

the epsilon here is given by tutorial.
Usually we decide the epsilon by follow formula:

$$\varepsilon = \frac{\sqrt{6}}{\sqrt{L_{in} + L_{out}}}$$

> **why?**
> The goal is to initialize the Theta values so they are in the range where the sigmoid() function gives an **active response**, once the weights are applied and the summations occur via (X * Theta). sigmoid() has a pretty useful slope between about **-3 and +3**, so that's what you want the initial weighted sums to end up. Outside of of that range, the slope of sigmoid() is very **flat**, and learning will occur very slowly.

[further dicussion about how to choose epsilon]

and the most diffcult:

nnCostFunction.m

```
%compute a2 and a3
%remember add bias column
X_add=[ones(m,1),X];
Z2=X_add*Theta1';
a2=sigmoid(Z2);
a2_add=[ones(m,1),a2];
Z3=a2_add*Theta2';
h=sigmoid(Z3);

%regularization
Theta1_no_bias=Theta1(:,2:end);
Theta2_no_bias=Theta2(:,2:end);
reg=sum(sum(Theta1_no_bias.^2))+sum(sum(Theta2_no_bias.^2));
reg=reg*lambda/2/m;

%CAUTION!

%the variable y is 5000*1, each value varies from 1-10
%we have to make a 5000*10 matrix for y, because we use One vs All strategy

Y=zeros(m,num_labels);
for i=1:m
    Y(i,y(i,1))=1;
end

%recall logistic regression. we have the hypothesis which is m*1
%and y also is m*1,so we can simpliy compute by h'*y
%but here each hypothesis is m*num_labels, while Y also is a matrix.
%we have to use ".*" to  multiplied element-wise each real J(theta)
J=sum(sum(-Y.*log(h)-(1-Y).*log(1-h)))/m+reg;

%=============Back Propagation============
delta3=h-Y;
delta2=delta3*Theta2_no_bias.*sigmoidGradient(Z2);
```

```
%Delta1 is 5000*401(including bias unit)
Delta2=delta3'*a2_add;
Delta1=delta2'*X_add;

%regularization
reg_theta1=[zeros(hidden_layer_size,1),Theta1_no_bias];
reg_theta1=reg_theta1.*(lambda/m);
reg_theta2=[zeros(num_labels,1),Theta2_no_bias];
reg_theta2=reg_theta2.*(lambda/m);

Theta1_grad = Delta1/m+reg_theta1;
Theta2_grad = Delta2/m+reg_theta2;
```

helpful skills:

1. keep the dimensions of matrices in your mind. (Convient way it write down all of them.)
2. always use matrix operation instead of for-loop, though it's quite hard at beginning.

details:

1.Expand the `y` output values.

> the given y is vector, each value varies from 1-10.
> using *One vs All* strategy, each example should have 10 columns, in which contain only one `1` and others are `0`.
>
> ```
> Y=zeros(m,num_labels);
> for i=1:m
>   Y(i,y(i,1))=1;
> end
> ```

2.costfunction J: when both hypothesis and Y are matrices

> at previous coding, we compute the costfunction by multiply vectors. (h is mx1 and y is mx1, thus we use `h'*y`.
> but now facing matrics, we need tricky skill.
> recall unregularized costfunction equal :
>
> $$J(\Theta) = \frac{1}{m} \sum_{i=1}^{N} \sum_{k=1}^{k} [-y_k^{(i)} log((h_\theta(x^{(i)}))_k - (1 - y_k^{(i)}) log(1 - (h_\theta(x^{(i)}))_k)]$$
>
> so what need to do is using the element-wise `.*` product of two matrices, then sum up.
>
> ```
> J=sum(sum(-Y.*log(h)-(1-Y).*log(1-h)))/m
> ```

3. about bp

> in tutorial it recommend using for-loop to compute, but actually it's tough coding task. Instead, using matrix operation would simplify the task.
> there is one essential point here: what dimension should the matrix be? Or to ask: what exactly matrics are expected?

Note: Excluding the first column of Theta2 is because the hidden layer bias unit has no connection to the input layer. (mathematical reason: **Chain Rule**) But our theta should contain bais, that's why the final Delta contain bais units.