1 Nerual Networks

1.3 Feedforward and cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log((h_\theta(x^{(i)}))_k) - (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right],$$

计算反向传播

先通过正向传播计算

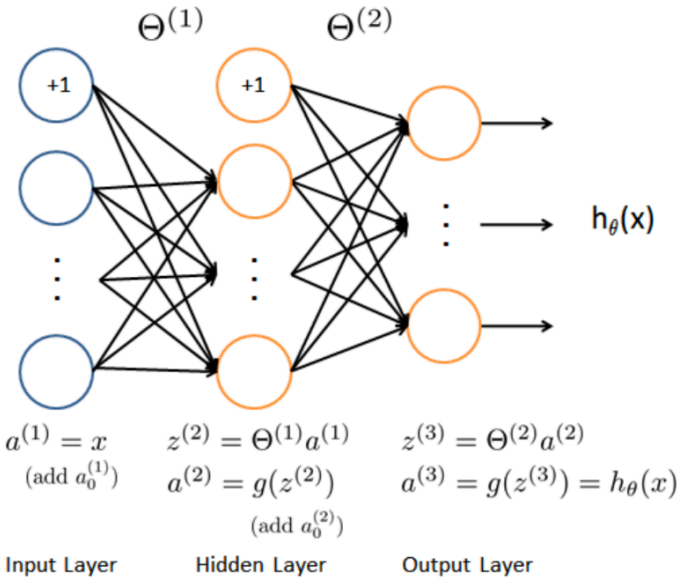

Figure 2: Neural network model.

```
% FeedForward
X = [ones(size(X,1),1) X];
z2 = X * Theta1';
a2 = sigmoid(z2);
a2 = [ones(size(a2,1),1) a2];
z3 = a2 * Theta2';
h = sigmoid(z3);

% CostFunction
% y(10*1 matrix)
y_ = zeros(size(X,1),num_labels);
for i=1:m,
y_(i,y(i))=1;
end

% Two Ways to Compute J
J = -sum(sum(y .* log(h))+sum((1-y) .* log(1-h)))/m;

% Another way, maybe a little clumzy
temp_matrix = zeors(m,1);
```

```
for i=1:m,
temp_matrix(i) = -log(h(i,:))*y_'(:,i) - log(1-h(i,:))*(1-y_'(:,i));
end
J = sum(temp_matrix);

```

1.4 Regularized cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log((h_\theta(x^{(i)}))_k) - (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\left[\sum_{j=1}^{25}\sum_{k=1}^{400}(\Theta_{j,k}^{(1)})^2 + \sum_{j=1}^{10}\sum_{k=1}^{25}(\Theta_{j,k}^{(2)})^2\right].$$

就是要计算Theta1,Theta2去掉正则项的平方求和

```
%Two ways
Theta1_temp = Theta1;
for i=1:size(Theta1_temp,1),
Theta1_temp(i,1)=0;
end
temp1 = sum(sum(Theta1_temp .^ 2))

J += lambda *(temp1+temp2)/2/m;

 %Another way
 Theta1_temp = Theta1(:,2:size(Theta1,2));
 temp1 = sum(sum(Theta1_temp.^2));

```

2 Backpropagation

2.1 Sigmoid gradient

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

2.2 Random Initialization

```
epsilon_inti = 0.12 ;
W = rand(L_out,1+L_in)*2*epsilon_init-epsilon_init;
```

2.3 Backpropagation

```
% in a loop
for ex = 1:m
%feedForward
a1 = X(ex,:);
z2 = a1 * Theta1' ;
a2 = [1 sigmoid(z2)];
z3 = a2 * Theta2';
a3 = sigmoid(z3);
y = y_(ex,:);
 delat3 = a3 - y;

 delat2 = Theta2(:,2:end)' * delta3 .* sigmoidGradient(z2);
```

```
Theta1_grad += delta2 * a1' ;
Theta2_grad += delta3 * a2' ;
end

Theta1(:,1) = 0;
Theta2(:,1) = 0;
Theta1_grad += lambda/m*Theta1;
Theta2_grad += lambda/m*Theta2;
```