实验四

**function g = sigmoidGradient(z)**

计算自变量为z的sigmoid函数的导数

代码：

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

**function W = randInitializeWeights(L\_in, L\_out)**

初始化第L层神经元和第L+1层神经元之间的权重，该权重矩阵的行数等于L+1层的神经元数量（不含常数项），列数等于L层的神经元数量（含常数项），其中L层的神经元数目为L\_in（不包含常数项），L+1层的神经元数目为L\_out（不包含常数项）

代码：

W = rand(L\_out, 1 + L\_in) \* 2 \* 0.1 - 0.1;

% set initial weights between -0.1 and 0.1

**function [J grad] = nnCostFunction(nn\_params, ...**

**input\_layer\_size, ...**

**hidden\_layer\_size, ...**

**num\_labels, ...**

**X, y, lambda)**

计算含二范数正则项的两层神经网络的损失函数值和偏导数

代码：

Y = zeros(m, num\_labels);

for i = 1:m

Y(i, y(i)) = 1; % Y: 5000 x 10

endfor

% Theta1: 25 x 401

% Theta2: 10 x 26

X = [ones(m, 1) X]; % X: 5000 x 401

z1 = X \* Theta1'; % z1: 5000 x 25

a1 = sigmoid(z1);

a1 = [ones(m, 1) a1]; % a1: 5000 x 26

z2 = a1 \* Theta2'; % z2: 5000 x 10

a2 = sigmoid(z2); % a2: 5000 x 10

J = -1 / m \* sum(sum( Y .\* log(a2) + (1-Y) .\* log(1-a2) ));

J = J + lambda / 2 / m \* (sum(sum(Theta1(:, 2:end).^2)) + sum(sum(Theta2(:, 2:end).^2)));

d2 = a2 - Y; % d2: 5000 x 10

D2 = d2' \* a1; % D2: 10 x 26

d1 = (d2 \* Theta2) .\* a1 .\* (1 - a1); % d1: 5000 x 26

D1 = d1(:, 2:end)' \* X; % D1: 25 x 401

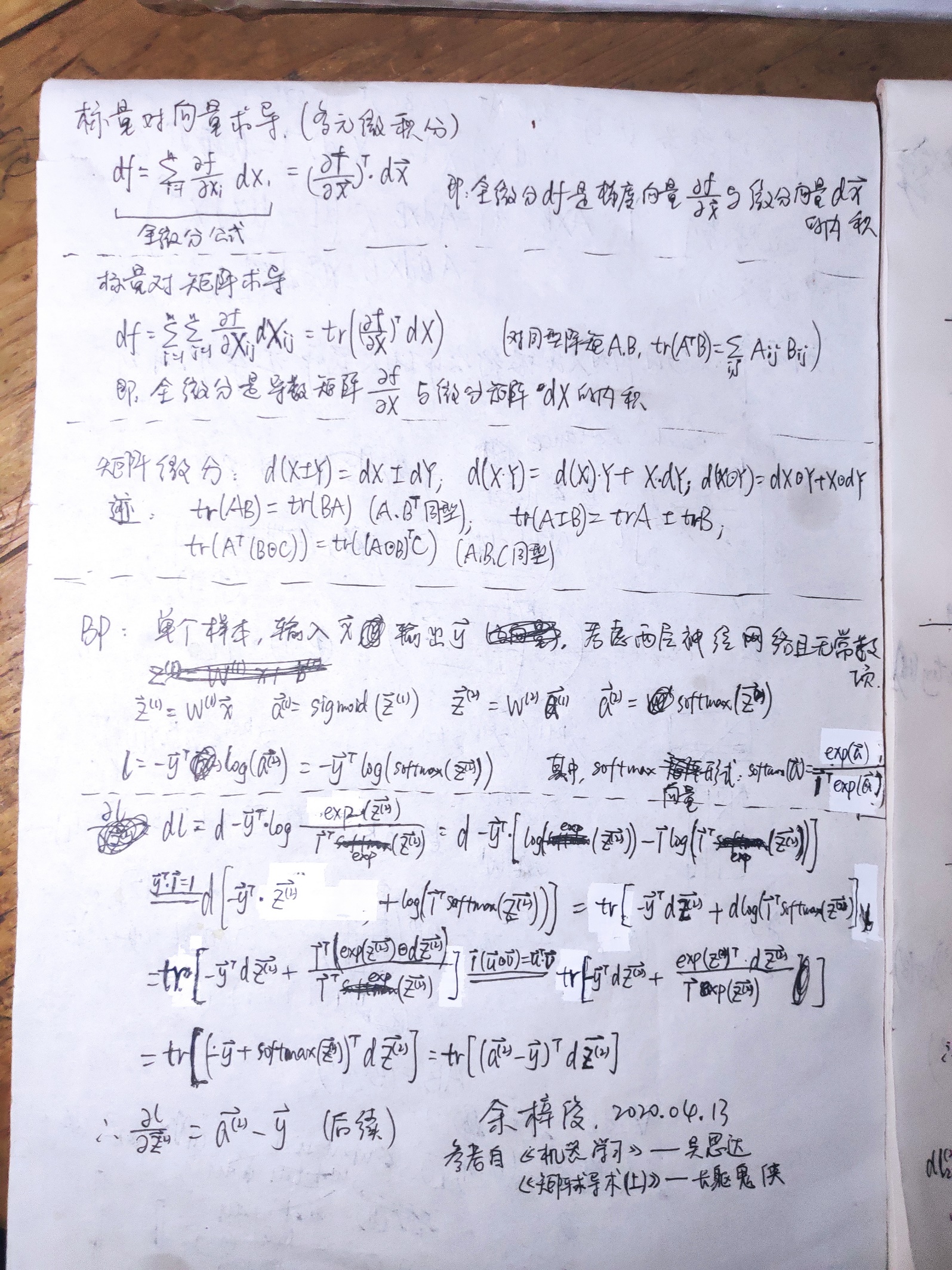
Theta2(:,1) = 0;

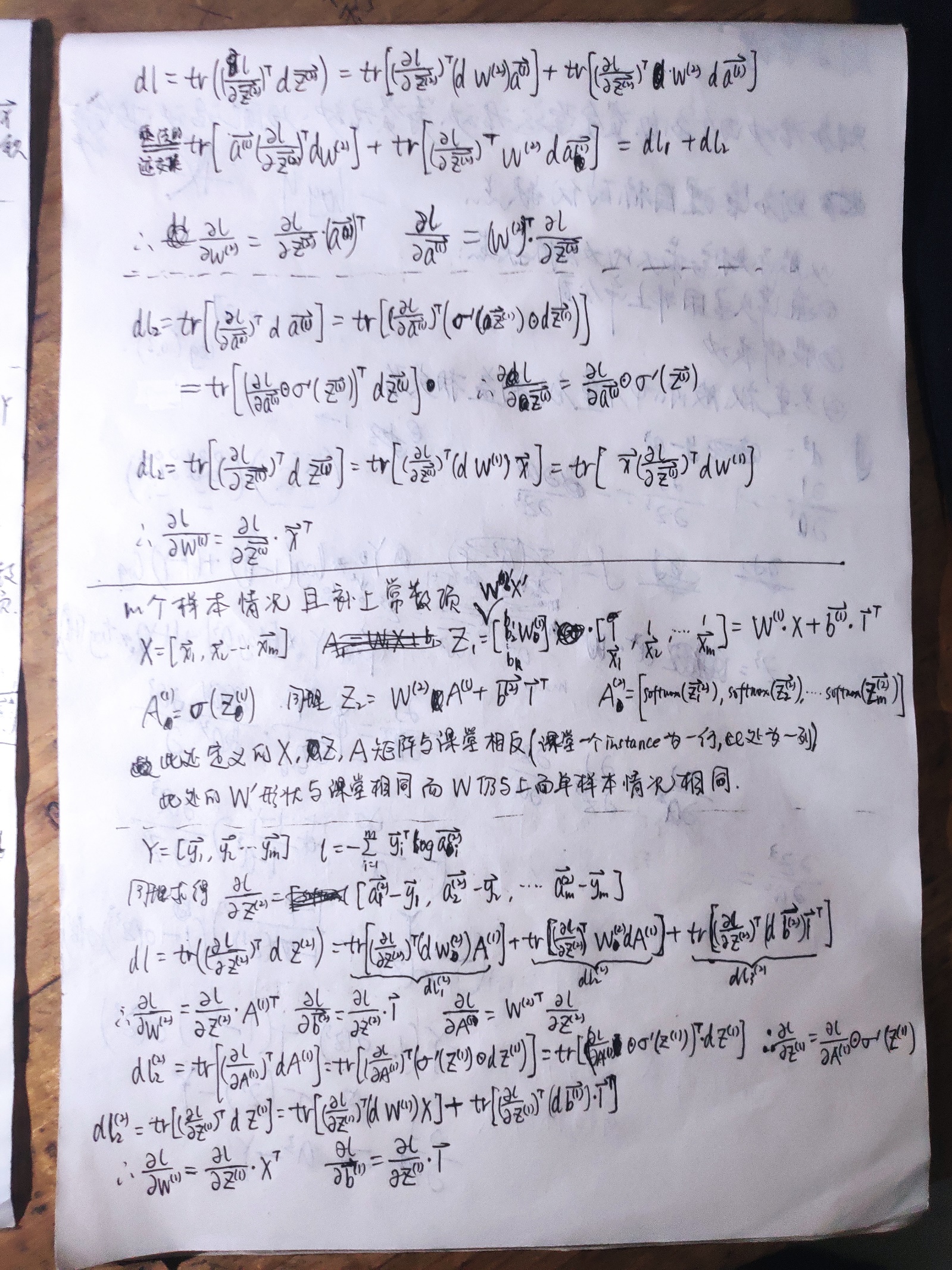
Theta2\_grad = (D2 + lambda \* Theta2) / m;

Theta1(:,1) = 0;

Theta1\_grad = (D1 + lambda \* Theta1) / m;

**反向传播矩阵形式推导笔记**





**ex4运行结果**

