ex2

3.

(a)

1. 梯度下降法：

   这里使用的停止条件是cost的变化值小于1e-6，迭代一共进行了85次。

   每次迭代需要计算:

   1. $X * theta$

      1000*210 210 *1 : 1000*210次乘法 999*210次加法

   2. $X' * (predictions - y)$

      210*1000 1000*1:   1000次减法 210*1000次乘法 219*1000次加法

   3. $theta = theta - alpha * updates$

      210*1 210*1; 210次乘法 210次减法

假设需要C次迭代完成，每次迭代的时间复杂度为$O(n*k+k)$，所以总的时间复杂度为$O(C(n*k+k))$ flops

k为特征纬度，n 为样本数量。

2. Normal Equations:

```
L = chol(X' * X);
temp = forwardsub(L',X'*y);
theta = backwardsub(L,temp);

% X'*X*theta = X'y
% cho(X'X) = triupper        上三角矩阵
% L'L*theta = X'y
% L'(L*theta) = X'y
% L'* z = X'y                解下三角
% L * theta = z              解上三角
```

1.chol：一共要计算 $O(n^2)$ 个元素，每个元素的计算里面有 的乘法，因此为 $O(n^3)$

2.对于前向替代和后向替代算法由于L或L'是三角矩阵，因此第一个可以按照 的顺序依次求出，所需的计算复杂度为 $O(n^2)$，同样，第二个也可以这样计算。

3.所以总的计算复杂度为$O(n^3+n^2)=O(n^3)$ flops。这样算法的优势在于，当需要针对同一个A解相对不同b的不同x的时候，新增的部分都可以以 $O(n^2)$ 复杂度完成。

(b)

```
>> norm(theta_grad)
ans =
   82.9163565517039
>> norm(theta_normal)
ans =
   82.9165386448065

>> computeCostMulti(X, y, theta_grad)
ans =
   13.3213102758129
>> computeCostMulti(X, y, theta_normal)
ans =
   13.3213070317204
```