Anomaly Detection 异常检测

Part 1：算法

Gaussian(Normal) distribution 高斯分布（正态分布）

表达式　$x \sim N(\mu, \sigma^2)$

参数　　μ —— mean 均值

　　σ^2 —— variance 方差

　　σ —— standard deviation 标准差

概率分布公式

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi} * \sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

参数估计（极大似然法）

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

- 其中，σ^2中的$\frac{1}{m}$在统计学中一般为$\frac{1}{m-1}$.

Density estimation 密度估计

给定训练集$\{x^{(1)}, \dots, x^{(m)}\}$，其中每个 x 都是 n 维的。

则对每个 x 有：

$$p(x) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * \dots * p(x_n; \mu_n, \sigma_n^2)$$

- 虽然这 n 个 features 不一定是独立的，理论上存在瑕疵，但是应用中该方法是可行的。

Anomaly detection algrithm 算法

步骤：

Step 1 选取可能与异常情况相关的 features x_i

Step 2 拟合参数 $\mu_1, \dots, \mu_n$, $\sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

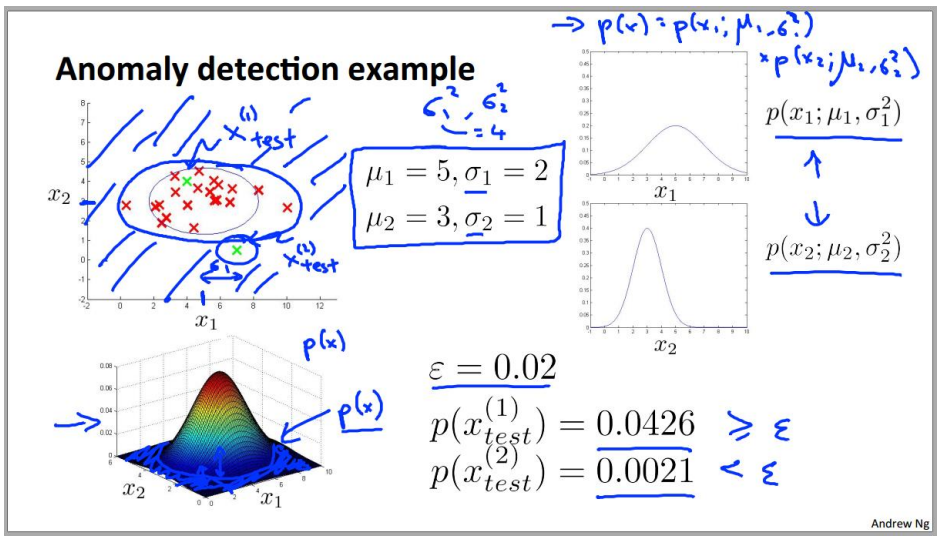
- 可以用 vectorized 的形式实现！

Step 3 给定新的（需检测的）样本 x，计算 p(x)即可

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi}*\sigma_j} \exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right)$$

if $p(x) < \varepsilon$，则异常。

示例：



Part 2：系统开发

How to evaluate 异常检测算法的评价

1. 使用实际数字数值化评价算法性能十分重要。

2. 如何划分数据集：

训练集 training set：$\{x^{(1)}, \dots, x^{(m)}\}$

交叉验证集 cross validation set：$\left\{(x_{cv}^{(1)}, y_{cv}^{(1)}) \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})\right\}$

测试集 test set：$\left\{(x_{test}^{(1)}, y_{test}^{(1)}) \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})\right\}$

- training set 中全部视作无标签样本，其实都是正常样本；cv set 和 test set 中都是有标签样本，包括少量异常样本。

以飞机引擎生产为例：

有 10000 个正常引擎、20 个异常引擎的数据。划分：

Training set：6000 正常（无标签）

Cv：2000 正常（y=0），10 异常（y=1）

Test：2000 正常（y=0），10 异常（y=1）

3. 评价算法：

Step 1 使用 training set 用不同方法（features 的选取、ε的选取）拟合出不同的若干概率密度模型 p(x)

Step 2 在 cv set 上用这些模型根据样本的 x 预测 y 标签：

$$y = \begin{cases} 1, & if\ p(x) < \varepsilon \\ 0, & if\ p(x) > \varepsilon \end{cases}$$

Step 3 用以下可能的标准来评价：

- 真假阳性、真假阴性
- Precision/ Recall（查准率/召回率）
- F1-score

- 不能使用简单的预测准确率标准来评价，因为数据是倾斜的（skewed data）。

  然后根据这些数字选取合适的 features 和ε，找最好的模型。

  例如，选取使得 F1-score 最小的ε。

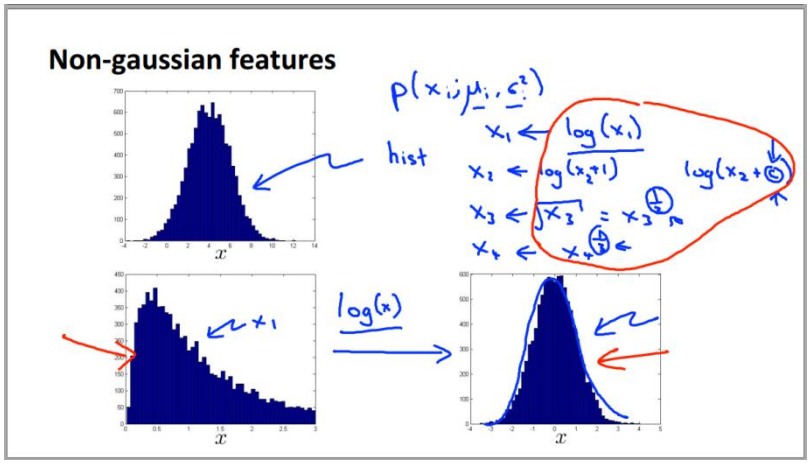
Step 4 用选定模型在 test set 上最终评价算法的表现。

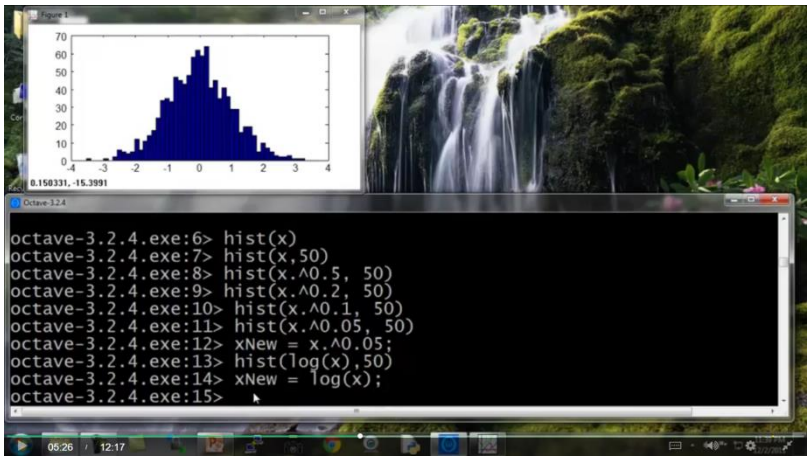
异常检测 vs. 监督学习

	异常检测	监督学习
特点	极少量正样本（0-20-50），大量负样本	大量正样本，大量负样本
原因	“异常”种类很多而正样本数量太少，难以从其中学习出正样本的特征；且未来的“异常”可能与 training set 中类型不同	Training set 中有足够多的正样本可供学习出特征；且未来的正样本很可能与 training set 中类型相似
引用举例	欺诈监测 制造业（飞机引擎） 数据中心机器检测	垃圾邮件分类 天气预测 癌症判断

How to choose features 特征的选取

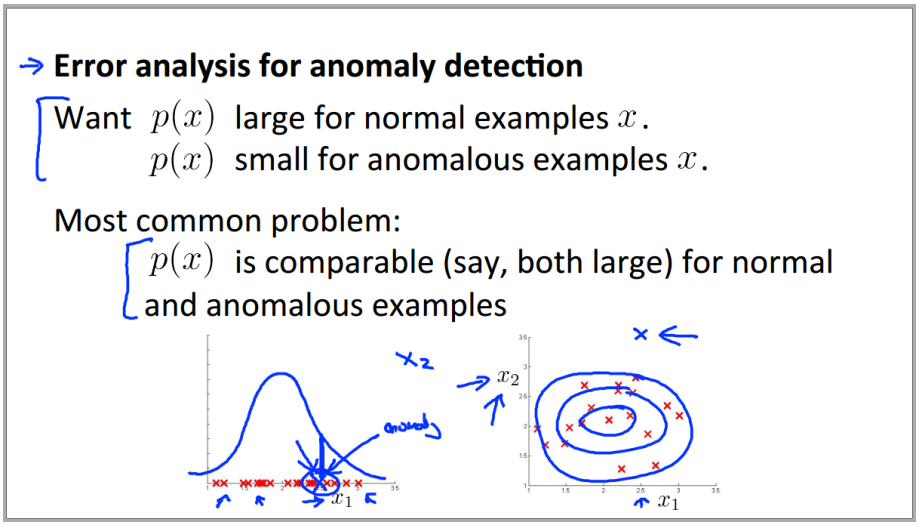
1. Non-Gaussian features 的处理：





2. 通过错判分析选取 features：

   当拟合出一个概率分布模型后发现，在 cv set 上正常样本和异常样本的 p(x)都很高（而不是一个高一个低从而说明要换ε）；也即当异常样本和正常样本混在一起时——对错判样本进行分析，找出能将其分开的新的 feature。



3. 选取 features 的一般方法：

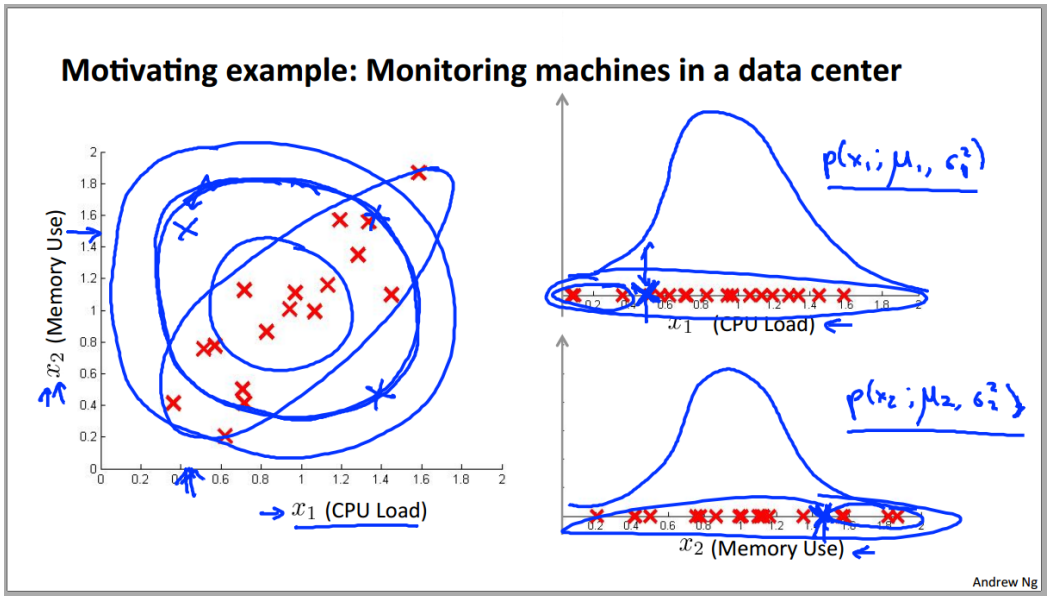
选取那些当样本为异常时，其值会变得非常非常大/非常非常小的量作为特征。

例如：数据中心的计算机的若干工作 features 中包括 CPU 负载、网络流量。这两者一般成线性关系，而当 CPU 卡住（比如卡在死循环里）时，CPU load 会远大于 network traffic，因此可以根据这两个 features 组合得到一个新的 feature：$\frac{CPU\ load}{network\ traffic}$ 或 $\frac{(CPU\ load)^2}{network\ traffic}$之类的。

Part 3：多元高斯（正态）分布

Multivariate Gaussian Distribution

当两个 features 之间存在相关关系时，用之前的 Gaussian Distribution 是存在问题的：



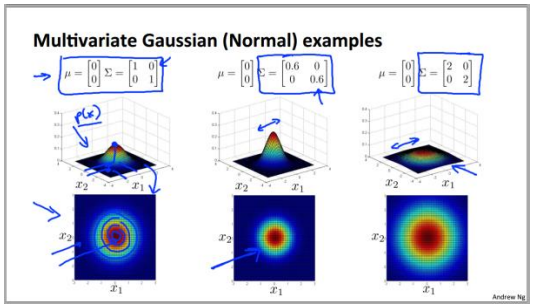
因此引入 Multivariate Gaussisan Distrbution。

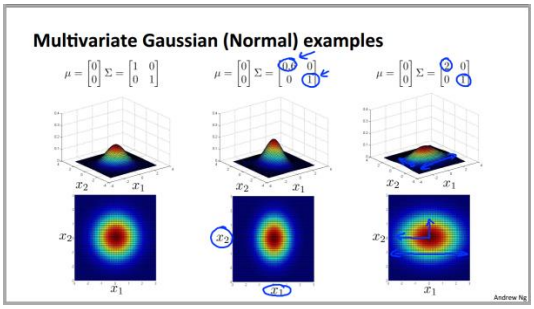
对于 n 维的一个样本 x，不再通过单独计算各个$\mathrm{p}(x_1; \mu_1, \sigma_1^2) * \mathrm{p}(x_2; \mu_2, \sigma_2^2) * \ldots * \mathrm{p}(x_n; \mu_n, \sigma_n^2)$从而得到$\mathrm{p}(\mathrm{x})$，而是直接计算$\mathrm{p}(\mathrm{x})$。新的参数为：$\mu \in \mathbb{R}^n$，$\Sigma \in \mathbb{R}^{n \times n}$（协方差矩阵）。

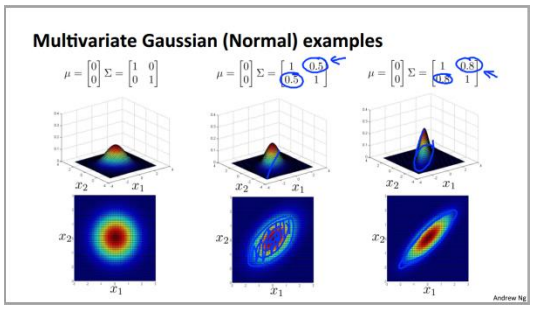
概率分布公式：

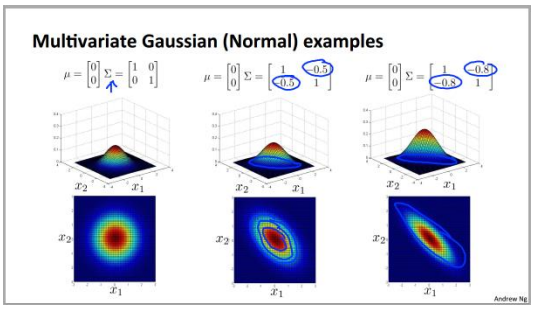
$$\mathrm{p}(\mathrm{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} * |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

图示如下：









Anomaly detection with mutivariate Gaussian 算法

步骤：

Step 1 拟合 p(x)模型参数

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

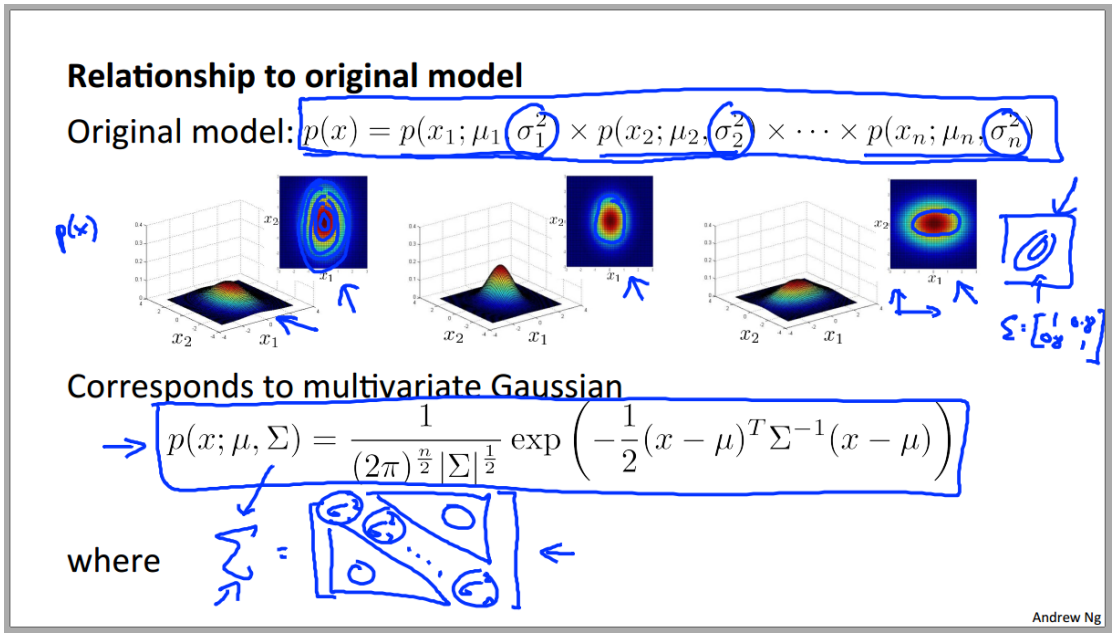
Step 2 给定新的（需检测的）样本 x，计算 p(x)即可

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} * |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

if $p(x) < \varepsilon$，则异常。

Mutivariate Gaussian 与原模型的关系

（关键是Σ和σ_j^2的关系）



原模型 vs. Mutivariate Gaussian

	原模型	Mutivariate Gaussian
1	当其中的 features 有线性关系（且异常时这个关系会打破）时，需手动构造新 feature	自动包含 features 之间的线性关系
2	计算量小，对于 n 很大的情况试用	计算量大
3	m（训练集数量）较小时没问题	必须保证 m>n，不然Σ不可逆（实际操作中应当 m>10n）

- 出现“Σ不可逆”的可能原因：

  1. m<=n
  2. 存在冗余特征变量（x 里面存在线性相关的 features）

Recommender System 推荐系统

Part 1：影片推荐系统——基于内容的 approach

Problem formulation 问题公式化

参数 n_u —— 用户数量

n_m —— 电影数量

$r(i,j)$ ——用户 j 对电影 i 是否作出评价

$y^{(i,j)}$ —— 用户 j 对电影 i 的评分（如果$r(i,j)=1$）

把影片推荐的问题化归为对$r(i,j)=0$的对应的y值进行预测的问题，即多个 linear regression 问题。

- 每个电影 i 都有一个特征向量$x^{(i)}$（基于内容特征，比如浪漫程度，动作程度）；
- 对于每个用户 j，拟合出一个偏好参数向量$\theta^{(j)}$（拟合方法见下文）；
- 那么最终得到的预测评分$y^{(i,j)}$为$(\theta^{(j)})^T(x^{(i)})$。

拟合$\theta^{(j)}$的方法

优化目标：（略）

算法：梯度下降等

- 注：添加$x_0=1$，故$x\in\mathbb{R}^{n+1}$。梯度下降中θ_0单独处理。

Part 2：影片推荐系统——协同过滤

思路

不知道影片特征值怎么办？通过用户的参数向量$\theta^{(j)}$和电影得分反过来拟合$x^{(i)}$。

过程 A：利用$\theta^{(j)}$拟合$x^{(i)}$，使得$(\theta^{(j)})^T(x^{(i)})$贴近$y^{(i,j)}$；

过程 B：利用$x^{(i)}$拟合$\theta^{(j)}$，使得$(\theta^{(j)})^T(x^{(i)})$贴近$y^{(i,j)}$。

循环论证怎么办？

先随便猜一个$\theta^{(j)}$，然后交替进行 A、B 过程，最终$x^{(i)}$和$\theta^{(j)}$都会收敛到一个较合理的值！

算法

Collaborative Filting Algrithm 协同过滤算法

将上一节中的 A、B 两个过程中的优化目标函数综合起来。

- 注意不再有$x_0 = 1$，在这个算法中$x \in \mathbb{R}^n$，$\theta \in \mathbb{R}^n$。

Step 1 对$x^{(1)}, \dots, x^{(n_m)}$和$\theta^{(1)}, \dots, \theta^{(n_u)}$选取小初值，避免对称性(symmetry breaking)；

Step 2 两个梯度下降同时进行；

Step 3 得到x和θ，计算$\theta^T x$，得到预测值。

Part 3：向量化——Low Rank Matrix Factorization

向量化

所有影片得分 Y：　　$n_m \times n_u$ 矩阵

所有预测得分：　　$n_m \times n_u$ 矩阵

影片特征 X：　　$n_m \times n$ 矩阵

用户参数 Θ：　　$n_u \times n$ 矩阵

$$\text{Predict} = \text{X} * \Theta^T$$

寻找关联影片（产品）

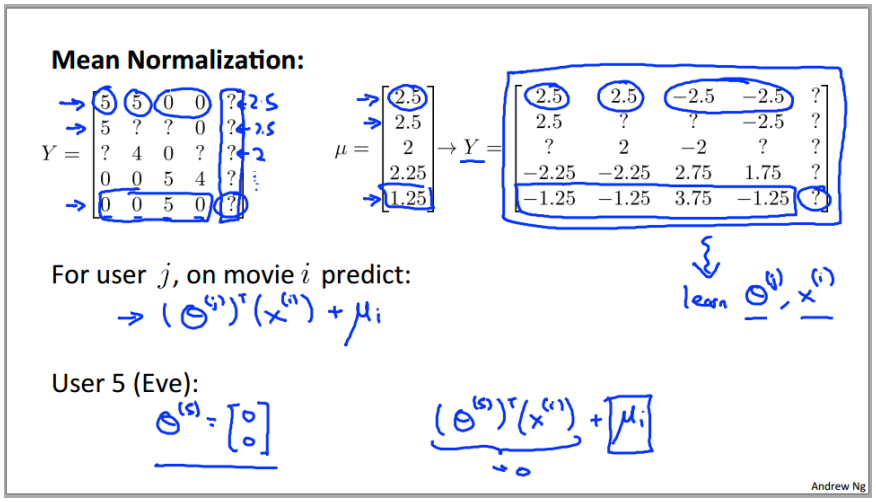
对于任一影片 i，确定关联影片 j 的依据：

$\|x^{(i)} - x^{(j)}\|$ 最小。

均值归一化 Mean Normalization

为了应对用户未评分的情况，采取均值归一化；

无需 feature scaling，因为都在 0-5 分范围。



Problems 疑难问题

1。 Suppose you run a bookstore, and have ratings (1 to 5 stars) of books. Your collaborative filtering algorithm has learned a parameter vector $\theta^{(j)}$ for user j, and a feature vector $x^{(i)}$ for each book. You would like to compute the "training error", meaning the average squared error of your system's predictions on all the ratings that you have gotten from your users. Which of these are correct ways of doing so (check all that apply)? For this problem, let m be the total number of ratings you have gotten from your users. (Another way of saying this is that $m = \sum_{i=1}^{n_m} \sum_{j=1}^{n_u} r(i,j)$). [Hint: Two of the four options below are correct.]

- [ ] $\frac{1}{m} \sum_{(i,j):r(i,j)=1} (\sum_{k=1}^{n} (\theta^{(j)})_k x_k^{(i)} - y^{(i,j)})^2$
- [ ] $\frac{1}{m} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} (\sum_{k=1}^{n} (\theta^{(j)})_k x_k^{(i)} - y^{(i,j)})^2$
- [ ] $\frac{1}{m} \sum_{(i,j):r(i,j)=1} \sum_{k=1}^{n} ((\theta^{(j)})_k x_k^{(i)} - y^{(i,j)})^2$
- [ ] $\frac{1}{m} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})_i x_j^{(i)} - y^{(i,j)})^2$