

# Machine Learning - Week 8

赵燕

## 目录

<b>I</b>	<b>Unsupervised Learning</b>	<b>2</b>
<b>1</b>	<b>Clustering</b>	<b>2</b>
1.1	Unsupervised Learning:Introduction . . . . .	2
1.2	K-Means Algorithm . . . . .	3
1.3	Optimization Objective . . . . .	6
1.4	Random Initialization . . . . .	8
1.5	Choosing the Number of Clusters . . . . .	9
<b>II</b>	<b>Dimensionality Reduction</b>	<b>12</b>
<b>2</b>	<b>Motivation</b>	<b>12</b>
2.1	Motivation I:Compression . . . . .	12
2.2	Motivation II:Visualization . . . . .	14
<b>3</b>	<b>Principal Component Analysis</b>	<b>16</b>
3.1	Principal Component Analysis Problem Formulation . . . . .	16
3.2	Principal Component Analysis Algorithm . . . . .	18
<b>4</b>	<b>Applying PCA</b>	<b>21</b>
4.1	Reconstruction from Compressed Representation . . . . .	21
4.2	Choosing the Number of Principal Components . . . . .	22
4.3	Advice for Applying PCA . . . . .	24

# Part I

## Unsupervised Learning

### 1 Clustering

聚类算法：非监督学习算法，要让计算机学习无标签数据，而不是之前的标签数据。

#### 1.1 Unsupervised Learning: Introduction

监督学习和无监督学习的比较：

在一个监督学习中，我们有一个有标签的训练集，我们的目标是找到能够区分正样本和负样本的决策边界，我们的监督学习中，有一系列标签，我们需要据此拟合一个假设函数。

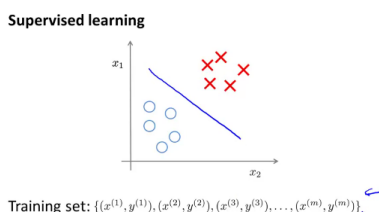


图 1: 监督学习

与此不同的是，在无监督学习中，数据没有附带任何标签，我们拿到的数据就是这样的：

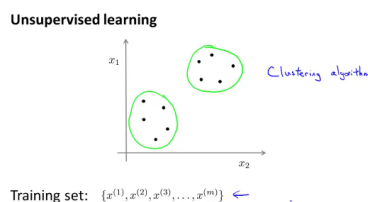


图 2: 无监督学习

在这里我们有一系列的点，却没有任何标签，因此我们的训练集可以写成只有 $x_1, x_2, x_3, \dots, x_m$ ，没有任何标签 $y$ 。因此图上画的这些点没有标签信息，也就是说，在无监督学习中，我们需要将一系列无标签的训练数据，输入到下一个算法中，然后我们告诉这个算法，快去为我们找找这个数据的内在结构给定数据。可能需要某种算法帮助我们寻找一种结构。图上的数据看起来可以分成两个分开的点集（称为簇），一个能够找到我圈出的这些点集的算法，被称为聚类算法。

聚类算法使我们介绍的第一个无监督学习算法，以后还将提到其他的无监督学习算法，它可以为我们找到其他类型的结构或者其他的一些模式，而不是簇。

聚类算法是用来做什么的？

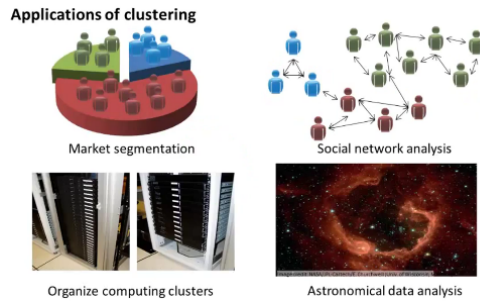


图 3: 聚类算法的应用

- 市场分割:也许你在数据库中分别销售产品或者分别提供更合适的服务,而你希望他们分为不同的课客户群;
- 社交网络分析: 例如 Facebook, Google+, 或者是其他的一些信息, 比如说: 你经常跟哪些人联系, 而这些人又经常给哪些人发邮件, 由此找到关系密切的人群, 因此, 这可能需要另一个聚类算法, 你希望用它发现社交网络中关系密切的朋友。
- 管理数据中心: 使用聚类算法来更好的组织计算机集群, 或者更好的管理数据中心, 因为数据中心中, 了解了哪些计算机数据中心更倾向于一起协作工作, 那么就可以重新分配资源, 重新布局网络, 由此优化数据中心, 优化数据通信。
- 天文学: 利用聚类算法视图了解星系的形成和其中的天文学的具体细节。

Which of the following statements are true? Check all that apply.

☒ In unsupervised learning, the training set is of the form  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  without labels  $y^{(i)}$ .

正确

☒ Clustering is an example of unsupervised learning.

正确

☒ In unsupervised learning, you are given an unlabeled dataset and are asked to find "structure" in the data.

正确

☐ Clustering is the only unsupervised learning algorithm.

未选择的是正确的

## 1.2 K-Means Algorithm

K-均值算法是最普及的聚类算法, 算法接受一个未标记的数据集, 然后将数据聚类成不同的组。

执行K-均值算法, 首先随机选择两个点, 这两个点叫做聚类中心, 就是图上的两个叉, 为什么是两个点呢, 因为我们希望聚出两个类。

K-均值是一个迭代方法, 需要做两件事情: 第一是簇分配, 第二是移动聚类中心。

簇分配：需要遍历所有的样本，就是图上绿色的点，然后依据每一个点是更接近红色的这个中心，还是更接近绿色的这个中心，来将每个数据点分配到两个不同的聚类中心中。

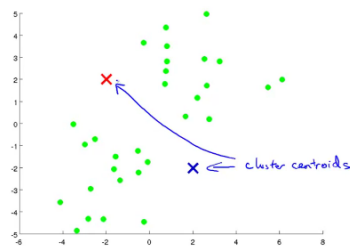


图 4: 簇分配

具体来讲，就是对数据集中的所有点，根据他们更接近红色这个中心，还是蓝色这个中心，进行染色，染色的结果如图：

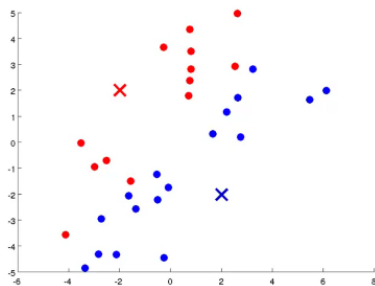


图 5: 簇分配染色

移动聚类中心:将两个聚类中心，也就是说红色的叉和蓝色的叉移动到 和它一样颜色的那堆点的均值处,找出所有红色的点,计算出它们的均值,就是所有红色的点平均下来的位置,然后把红色点的聚类中心移动到这里,蓝色也是这样,找出所有蓝色的点计算它们的均值把蓝色的叉放到那里,我们将按照图上所示这么移动:

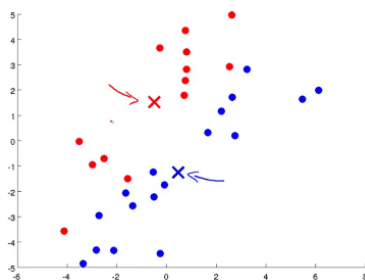


图 6: 移动聚类中心

重复上述步骤，继续簇分配和移动聚类中心，一直迭代下去，直到最后聚类中心不会变，并且哪些点的颜色也不会变，在这时我们就可以说 K-均值方法已经收敛了，在这些数据中找到两个簇。

K-均值是一个迭代算法，假设我们想要将数据聚类成 $n$ 个组，其方法是：

首先选择 $K$ 个随机的点，称为聚类中心（cluster centroids）；

对于数据集中的每一个数据，按照距离 $K$ 个中心点的距离，将其与距离最近的中心点关联起来，与同一个中心点关联的所有点聚成一类。

计算每一个组的平均值，将该组所关联的中心点移动到平均值的位置。

重复步骤直至中心点不再变化。

### K-means algorithm

Input:

- $K$  (number of clusters)  $\leftarrow$
- Training set  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$   $\leftarrow$

$$\underline{x^{(i)} \in \mathbb{R}^n} \text{ (drop } x_0 = 1 \text{ convention)}$$

图 7: K-均值算法

$c^{(i)}$  是距离样本  $x^{(i)}$  的距离的平方，使距离最小或者距离的平方最小都能让我们得到一个相同的  $c^{(i)}$ ，注意  $c^{(i)}$  是一个在1到K之间的数。（通常还是写成距离的平方）

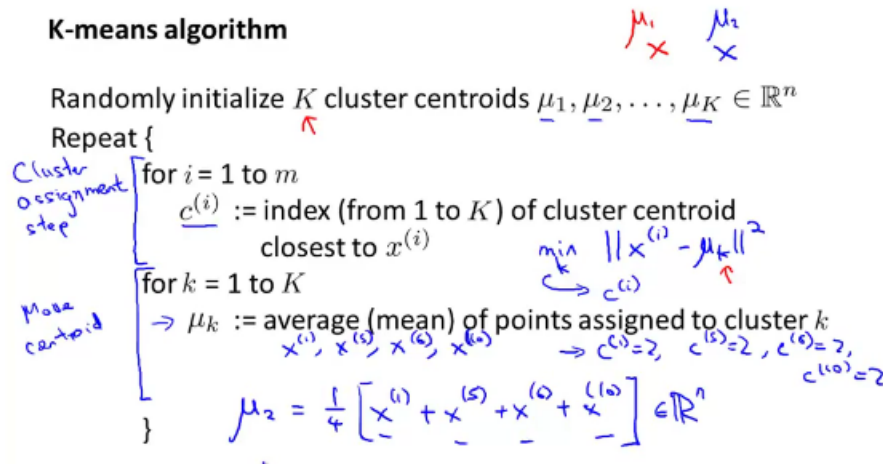


图 8: K-均值算法

算法分为两个步骤,第一个 for 循环是赋值步骤,即:对于每一个样例  $i$ ,计算其应该属于的类。第二个 for 循环是聚类中心的移动,即:对于每一个类  $k$ ,重新计算该类的聚类中心。

K-均值算法也可以很便利地用于将数据分为许多不同组,即使在没有非常明显区分的组群的情况下也可以。下图所示的数据集包含身高和体重两项特征构成的,利用 K-均值算法将数据分为三类,用于帮助确定将要生产的T-恤衫的三种尺寸 S,M,L。

### K-means for non-separated clusters

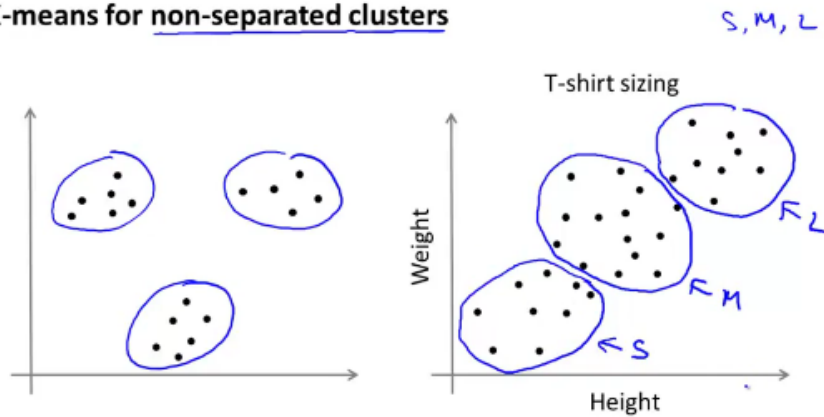


图 9: K-means for non-separated clusters

Suppose you run k-means and after the algorithm converges, you have:  $c^{(1)} = 3, c^{(2)} = 3, c^{(3)} = 5, \dots$

Which of the following statements are true? Check all that apply.

☒ The third example  $x^{(3)}$  has been assigned to cluster 5.

正确

☒ The first and second training examples  $x^{(1)}$  and  $x^{(2)}$  have been assigned to the same cluster.

正确

☐ The second and third training examples have been assigned to the same cluster.

未选择的是正确的

☒ Out of all the possible values of  $k \in \{1, 2, \dots, K\}$  the value  $k = 3$  minimizes  $\|x^{(2)} - \mu_k\|^2$ .

正确

## 1.3 Optimization Objective

K-均值最小化问题,是要最小化所有的数据点与其所关联的聚类中心点之间的距离之和,因此 K-均值的代价函数(又称畸变函数 Distortion function)为:

### K-means optimization objective

- $c^{(i)}$  = index of cluster  $(1, 2, \dots, K)$  to which example  $x^{(i)}$  is currently assigned
- $\mu_k$  = cluster centroid  $k$  ( $\mu_k \in \mathbb{R}^n$ )  $K$   $k \in \{1, 2, \dots, K\}$
- $\mu_{c^{(i)}}$  = cluster centroid of cluster to which example  $x^{(i)}$  has been assigned  $x^{(i)} \rightarrow 5$   $c^{(i)} = 5$   $\mu_{c^{(i)}} = \mu_5$

Optimization objective:

$$\rightarrow J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$\rightarrow \min_{c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$   
Distortion




图 10: 优化目标函数

其中  $\mu_{c^{(i)}}$  代表与  $x^{(i)}$  最近的聚类中心点;

每个样本到每个样本所属簇的距离的平方;

$K$ : 簇的总数;

$k$ : 聚类中心的符号,  $k \in 1, 2, 3, \dots, K$

### K-means algorithm

Randomly initialize  $K$  cluster centroids  $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster assignment step  
 Minimize  $J(\dots)$  w.r.t  $c^{(1)}, c^{(2)}, \dots, c^{(m)}$  ←  
 (holding  $\mu_1, \dots, \mu_K$  fixed)

for  $i = 1$  to  $m$   
 $c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid closest to  $x^{(i)}$

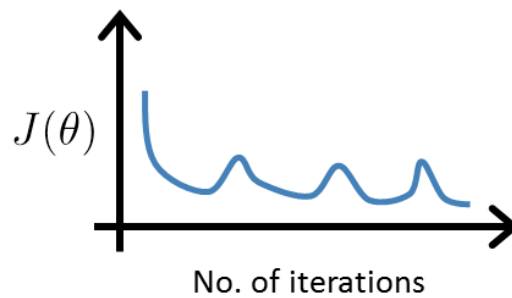
for  $k = 1$  to  $K$   
 $\mu_k :=$  average (mean) of points assigned to cluster  $k$

} move centroid minimize  $J(\dots)$  w.r.t  $\mu_1, \dots, \mu_K$

图 11: K-means algorithm

回顾刚才给出的 K-均值迭代算法,我们知道,第一个循环是用于减小  $c^{(i)}$  引起的代价,而第二个循环则是用于减小  $\mu_i$  引起的代价。迭代的过程一定会是每一次迭代都在减小代价函数,不然便是出现了错误。

Suppose you have implemented k-means and to check that it is running correctly, you plot the cost function  $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k)$  as a function of the number of iterations. Your plot looks like this:



What does this mean?

- ☐ The learning rate is too large.
- ☐ The algorithm is working correctly.
- ☐ The algorithm is working, but  $k$  is too large.
- ☒ It is not possible for the cost function to sometimes increase. There must be a bug in the code.

正确

## 1.4 Random Initialization

在运行 K-均值算法的之前,我们首先要随机初始化所有的聚类中心点,下面介绍怎样做:

### K-means algorithm

Randomly initialize  $K$  cluster centroids  $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

```
Repeat {
  for  $i = 1$  to  $m$ 
     $c^{(i)} := \text{index (from 1 to } K \text{ ) of cluster centroid}$ 
     $\text{closest to } x^{(i)}$ 
  for  $k = 1$  to  $K$ 
     $\mu_k := \text{average (mean) of points assigned to cluster } k$ 
}
```

图 12: 聚类算法

1. 我们应该选择  $K \ll m$ , 即聚类中心点的个数要小于所有训练集实例的数量
2. 随机选择  $K$  个训练实例, 然后令  $K$  个聚类中心分别与这  $K$  个训练实例相等



### K-means optimization objective


- $c^{(i)}$  = index of cluster  $(1, 2, \dots, K)$  to which example  $x^{(i)}$  is currently assigned
  - $\mu_k$  = cluster centroid  $k$  ( $\mu_k \in \mathbb{R}^n$ )  $K$   $k \in \{1, 2, \dots, K\}$
  - $\mu_{c^{(i)}}$  = cluster centroid of cluster to which example  $x^{(i)}$  has been assigned  $x^{(i)} \rightarrow \underline{5}$   $c^{(i)} = \underline{5}$   $\mu_{c^{(i)}} = \mu_5$
- Optimization objective:
- $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$   $\leftarrow$
  - $\min_{c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$
  - $\mu_1, \dots, \mu_K$  Distortion
- 

图 13: 随机初始化

K-均值的一个问题在于,它有可能会停留在一个局部最小值处,而这取决于初始化的情况。

为了解决这个问题,我们通常需要多次运行 K-均值算法,每一次都重新进行随机初始化,最后再比较多次运行 K-均值的结果,选择代价函数最小的结果。这种方法在 K 较小的时候(2-10)还是可行的,但是如果 K 较大,这么做也可能不会有明显地改善。

### Random initialization

```

For i = 1 to 100 {
    → Randomly initialize K-means.
    Run K-means. Get  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$ .
    Compute cost function (distortion)
    →  $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$ 
}

```

Pick clustering that gave lowest cost  $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

$K = 2 - 10$   $\uparrow$

图 14: 随机初始化算法

Which of the following is the recommended way to initialize k-means?

- ☐ Pick a random integer  $i$  from  $\{1, \dots, k\}$ . Set  $\mu_1 = \mu_2 = \dots = \mu_k = x^{(i)}$ .
  - ☐ Pick  $k$  distinct random integers  $i_1, \dots, i_k$  from  $\{1, \dots, k\}$ .  
Set  $\mu_1 = x^{(i_1)}, \mu_2 = x^{(i_2)}, \dots, \mu_k = x^{(i_k)}$ .
  - ☒ Pick  $k$  distinct random integers  $i_1, \dots, i_k$  from  $\{1, \dots, m\}$ .  
Set  $\mu_1 = x^{(i_1)}, \mu_2 = x^{(i_2)}, \dots, \mu_k = x^{(i_k)}$ .
  - ☐ Set every element of  $\mu_i \in \mathbb{R}^n$  to a random value between  $-\epsilon$  and  $\epsilon$ .
- 正确

图 15: 聚类算法

## 1.5 Choosing the Number of Clusters

如何选择聚类数目或者如何选择大写K的值。

没有所谓最好的选择聚类数的方法,通常是需要根据不同的问题,人工进行选择的。选择的时候思考我们运用 K-均值算法聚类的动机是什么,然后选择能最好服务于该目标聚类数。

当人们在讨论,选择聚类数目的方法时,有一个可能会谈及的方法叫作“肘部法则”(Elbow Method)。关于“肘部法则”,我们所需要做的是改变 K 值,也就是聚类类别数目的总数。我们用一个聚类来运行 K 均值聚类方法。这就意味着,所有的数据都会分到一个聚类里,然后计算成本函数或者计算畸变函数 J。K 代表聚类数字。

我们可能会得到一条类似于这样的曲线。像一个人的肘部。这就是“肘部法则”所做的,让我们来看这样一个图,看起来就好像有一个很清楚的肘在那儿。好像人的手臂,如果你伸出你的胳膊,那么这就是你的肩关节、肘关节、手。这就是“肘部法则”。

### Choosing the value of K

Elbow method:

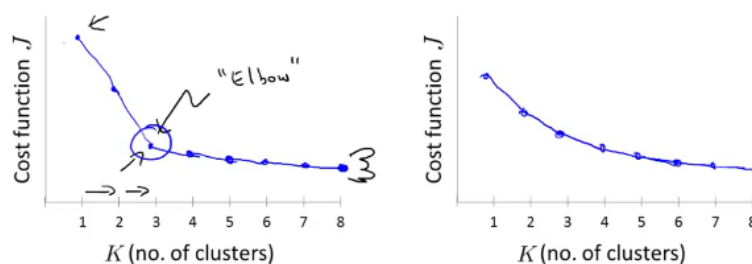


图 16: 肘部法则

你会发现这种模式,它的畸变值会迅速下降,从 1 到 2,从 2 到 3 之后,你会在 3 的时候达到一个肘点。在此之后,畸变值就下降得非常慢,看起来就像使用 3 个聚类来进行聚类是正确的,这是因为那个点是曲线的肘点,畸变值下降得很快,K 等于 3 之后就下降得很慢,那么我们就选 K 等于 3。当你应用“肘部法则”的时候,如果你得到了一个像上面这样的图,那么这将是一种用来选择聚类个数的合理方法。

Suppose you run k-means using  $k = 3$  and  $k = 5$ . You find that the cost function  $J$  is much higher for  $k = 5$  than for  $k = 3$ . What can you conclude?

- ☐ This is mathematically impossible. There must be a bug in the code.
- ☐ The correct number of clusters is  $k = 3$ .
- ☒ In the run with  $k = 5$ , k-means got stuck in a bad local minimum. You should try re-running k-means with multiple random initializations.

正确

- ☐ In the run with  $k = 3$ , k-means got lucky. You should try re-running k-means with  $k = 3$  and different random initializations until it performs no better than with  $k = 5$ .

后续的目的: 选择聚类算法要符合你后续的目的。

例如,我们的 T-恤制造例子中,我们要将用户按照身材聚类,我们可以分成 3 个尺寸 S,M,L 也可以分成 5 个尺寸 XS,S,M,L,XL,这样的选择是建立在回答“聚类后我们制造的 T-恤是否能较好地适合我们的客户”这个问题的基础上作出的。

### Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

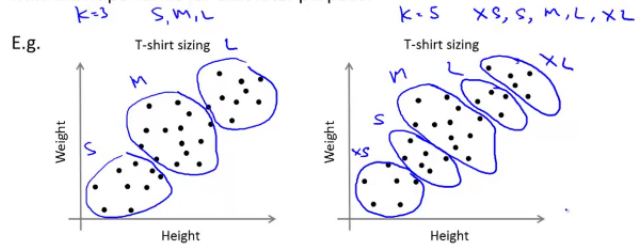


图 17: 聚类算法

再比如做图像压缩这个例子，就要根据图片压缩的具体标准选择你需要的聚类数K。

1。

For which of the following tasks might K-means clustering be a suitable algorithm? Select all that apply.

- ☐ Given a database of information about your users, automatically group them into different market segments.
- ☒ Given sales data from a large number of products in a supermarket, figure out which products tend to form coherent groups (say are frequently purchased together) and thus should be put on the same shelf.
- ☐ Given historical weather records, predict the amount of rainfall tomorrow (this would be a real-valued output)
- ☐ Given sales data from a large number of products in a supermarket, estimate future sales for each of these products.

2。

Suppose we have three cluster centroids  $\mu_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ ,  $\mu_2 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}$  and  $\mu_3 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$ .

Furthermore, we have a training example  $x^{(i)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ . After a cluster assignment step, what will  $c^{(i)}$  be?

- ☐  $c^{(i)}$  is not assigned
- ☒  $c^{(i)} = 1$
- ☐  $c^{(i)} = 3$
- ☐  $c^{(i)} = 2$

3。

K-means is an iterative algorithm, and two of the following steps are repeatedly carried out in its inner-loop. Which two?

- ☒ Move the cluster centroids, where the centroids  $\mu_k$  are updated.
- ☐ Using the elbow method to choose K.
- ☐ Feature scaling, to ensure each feature is on a comparable scale to the others.
- ☒ The cluster assignment step, where the parameters  $c^{(i)}$  are updated.

4. Suppose you have an unlabeled dataset  $\{x^{(1)}, \dots, x^{(m)}\}$ . You run K-means with 50 different random initializations, and obtain 50 different clusterings of the data. What is the recommended way for choosing which one of these 50 clusterings to use?
- ☐ Always pick the final (50th) clustering found, since by that time it is more likely to have converged to a good solution.
  - ☐ The only way to do so is if we also have labels  $y^{(i)}$  for our data.
  - ☒ For each of the clusterings, compute  $\frac{1}{m} \sum_{i=1}^m ||x^{(i)} - \mu_{c(i)}||^2$ , and pick the one that minimizes this.
  - ☐ The answer is ambiguous, and there is no good way of choosing.

5. Which of the following statements are true? Select all that apply.
- ☒ For some datasets, the "right" or "correct" value of K (the number of clusters) can be ambiguous, and hard even for a human expert looking carefully at the data to decide.
  - ☒ If we are worried about K-means getting stuck in bad local optima, one way to ameliorate (reduce) this problem is if we try using multiple random initializations.
  - ☐ The standard way of Initializing K-means is setting  $\mu_1 = \dots = \mu_k$  to be equal to a vector of zeros.
  - ☐ Since K-Means is an unsupervised learning algorithm, it cannot overfit the data, and thus it is always better to have as large a number of clusters as is computationally feasible.

## Part II

# Dimensionality Reduction

## 2 Motivation

### 2.1 Motivation I: Compression

这个视频,我想开始谈论第二种类型的无监督学习问题,称为降维。有几个不同的原因使你可能想要做降维。一是数据压缩,数据压缩不仅允许我们压缩数据,因而使用较少的计算机内存或磁盘空间,但它也让我们加快我们的学习算法。

但首先,让我们谈论降维是什么。作为一种生动的例子,我们收集的数据集,有许多,许多特征,我绘制在这里。

假设我们未知两个的特征  $x_1$ :长度:用厘米表示; $x_2$ ,是用英寸表示同一物体的长度。

所以,这给了我们高度冗余表示,也许不是两个分开的特征  $x_1$  和  $x_2$ ,这两个基本的长度度量,也许我们想要做的是减少数据到一维,只有一个数测量这个长度。这个例子似乎有点做作,这里厘米英寸的例子实际上不是那么不切实际的,两者并没有什么不同。

## Data Compression

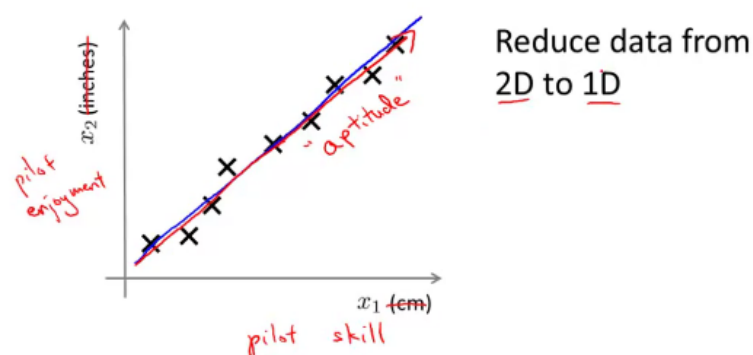


图 18: 降维

将数据从二维降至一维: 假使我们要采用两种不同的仪器来测量一些东西的尺寸, 其中一个仪器测量结果的单位是英寸, 另一个仪器测量的结果是厘米, 我们希望将测量的结果作为我们机器学习的特征。现在的问题是, 两种仪器对同一个东西测量的结果不完全相等(由于误差、精度等), 而将两者都作为特征有些重复, 因而, 我们希望将这个二维的数据降至一维。

从这件事情我看到的东发生在工业上的事。如果你有几百个或成千上万的特征, 它是它这往往容易失去你需要的特征。有时可能有几个不同的工程团队, 也许一个工程队给你二百个特征, 第二工程队给你另外三百个的特征, 第三工程队给你五百个特征, 一千多个特征都在一起, 它实际上会变得非常困难, 去跟踪你知道的那些特征, 你从那些工程队得到的。其实不想有高度冗余的特征一样。

多年我一直在研究直升机自动驾驶。诸如此类。如果你想测量——如果你想做, 你知道, 做一个调查或做这些不同飞行员的测试——你可能有一个特征:  $x_1$ , 这也许是他们的技能(直升机飞行员), 也许“ $x_2$ ”可能是飞行员的爱好。这是表示他们是否喜欢飞行, 也许这两个特征将高度相关。你真正关心的可能是这条红线的方向, 不同的特征, 决定飞行员的能力。

## Data Compression

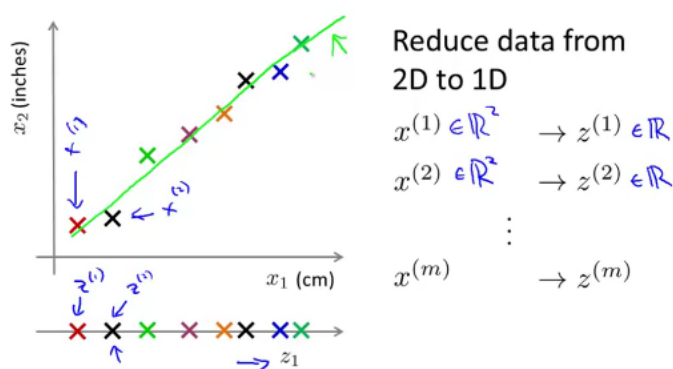


图 19: 数据压缩

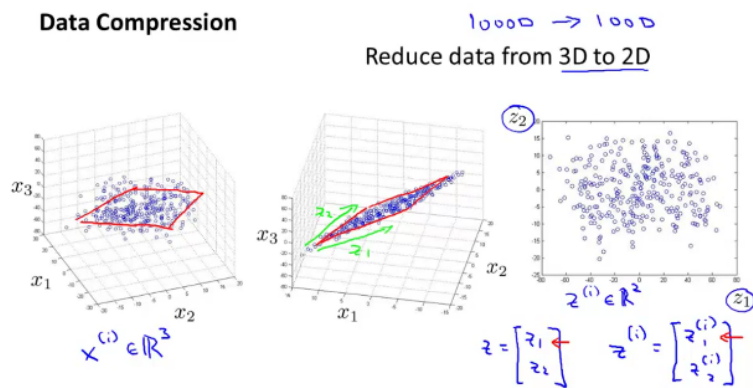


图 20: 三维降二维

将数据从三维降至二维: 这个例子中我们要将一个三维的特征向量降至一个二维的特征向量。过程是与上面类似的,我们将三维向量投射到一个二维的平面上,强迫使得所有的数据都在同一个平面上,降至二维的特征向量。

这样的处理过程可以被用于把任何维度的数据降到任何想要的维度,例如将 1000 维的特征降至 100 维。

Suppose we apply dimensionality reduction to a dataset of  $m$  examples  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ , where  $x^{(i)} \in \mathbb{R}^n$ . As a result of this, we will get out:

- ☐ A lower dimensional dataset  $\{z^{(1)}, z^{(2)}, \dots, z^{(k)}\}$  of  $k$  examples where  $k \leq n$ .
  - ☐ A lower dimensional dataset  $\{z^{(1)}, z^{(2)}, \dots, z^{(k)}\}$  of  $k$  examples where  $k > n$ .
  - ☒ A lower dimensional dataset  $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$  of  $m$  examples where  $z^{(i)} \in \mathbb{R}^k$  for some value of  $k$  and  $k \leq n$ .
- 正确
- ☐ A lower dimensional dataset  $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$  of  $m$  examples where  $z^{(i)} \in \mathbb{R}^k$  for some value of  $k$  and  $k > n$ .

正如我们所看到的,最后,这将使我们能够使我们的一些学习算法运行也较晚,但我们会在以后的视频提到它。

## 2.2 Motivation II: Visualization

可视化数据: 在许多及其学习问题中,如果我们能将数据可视化,我们便能寻找到一个更好的解决方案,降维可以帮助我们。

**Data Visualization**

$x \in \mathbb{R}^{50}$        $x^{(i)} \in \mathbb{R}^{50}$

Country	$x_1$ GDP (trillions of US\$)	$x_2$ Per capita GDP (thousands of intl. \$)	$x_3$ Human Development Index	$x_4$ Life expectancy	$x_5$ Poverty Index (Gini as percentage)	$x_6$ Mean household income (thousands of US\$)	...
→ Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...	...	...	...	...	...	...	...

假使我们有有若干关于许多不同国家的数据,每一个特征向量都有 50 个特征(如,GDP,人均 GDP,平均寿命等)。如果要将其 50 维的数据可视化是不可能的。使用降维的方法将其降至 2 维,我们便可以将其可视化了。

**Data Visualization**

$z^{(i)} \in \mathbb{R}^2$

Country	$z_1$	$z_2$
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...	...	...

Reduce data from 50D to 2D

图 21: 可视化数据

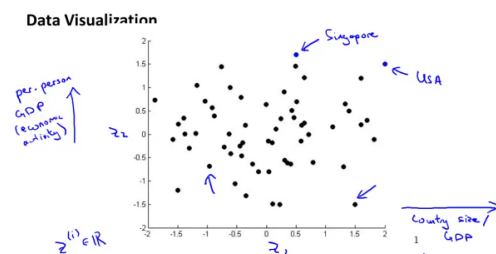


图 22: 可视化数据

这样做的问题在于,降维的算法只负责减少维数,新产生的特征的意义就必须由我们自己去发现了。

Suppose you have a dataset  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  where  $x^{(i)} \in \mathbb{R}^n$ . In order to visualize it, we apply dimensionality reduction and get  $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$  where  $z^{(i)} \in \mathbb{R}^k$  is  $k$ -dimensional. In a typical setting, which of the following would you expect to be true? Check all that apply.

☐  $k > n$

未选择的是正确的

☒  $k \leq n$

正确

☐  $k \geq 4$

未选择的是正确的

☒  $k = 2$  or  $k = 3$  (since we can plot 2D or 3D data but don't have ways to visualize higher dimensional data)

### 3 Principal Component Analysis

主成分分析法 (PCA)

#### 3.1 Principal Component Analysis Problem Formulation

公式表达

主成分分析(PCA)是最常见的降维算法。

在 PCA 中,我们要做的是找到一个方向向量(Vector direction),当我们把所有的数据都投射到该向量上时,我们希望投射平均均方误差能尽可能地小。(蓝色小线段的距离最小)也就是投影误差最小,方向向量是一个经过原点的向量,而投影误差是从特征向量向该方向向量作垂线的长度。

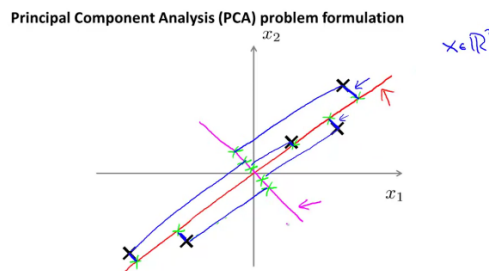


图 23: 投影误差

下面给出主成分分析问题的描述:

问题是要将  $n$  维数据降至  $k$  维,目标是找到向量  $u(1), u(2), \dots, u(k)$  使得总的投影误差最小。(寻找使数据点到投影后的点之间的距离最小的向量)



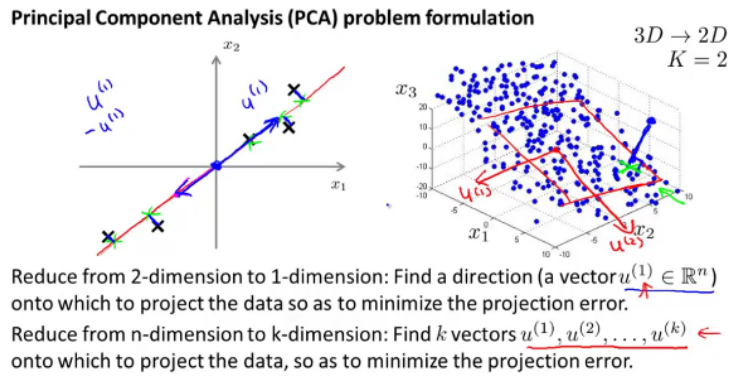


图 24: PCA

主成分分析与线性回归的比较:

主成分分析与线性回归是两种不同的算法。主成分分析最小化的是投射误差(Projected Error) (最小的直角距离), 而线性回归尝试的是最小化预测误差(某个点与通过假设得到的预测值之间的距离)。线性回归的目的是预测结果, 而主成分分析不作任何预测。线性回归有预测值 $y$ , 而PCA的每个样本都被同等的对待。

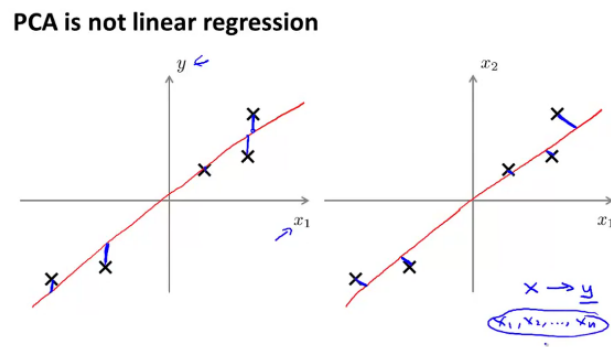
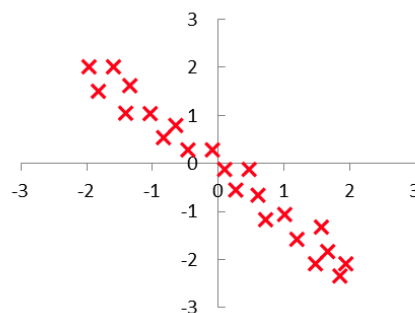


图 25: 线性回归与PCA

上图中,左边的是线性回归的误差(垂直于横轴投影),右边则是主要成分分析的误差(垂直于红线投影)。

Suppose you run PCA on the dataset below. Which of the following would be a reasonable vector  $u^{(1)}$  onto which to project the data? (By convention, we choose  $u^{(1)}$  so that  $\|u^{(1)}\| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$ , the length of the vector  $u^{(1)}$ , equals 1.)



- ☐  $u^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- ☐  $u^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- ☐  $u^{(1)} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$
- ☒  $u^{(1)} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

正确

PCA 将  $n$  个特征降维到  $k$  个,可以用来进行数据压缩,如果 100 维的向量最后可以用 10 维来表示,那么压缩率为 90%。同样图像处理领域的 KL 变换使用 PCA 做图像压缩。但 PCA 要保证降维后,还要保证数据的特性损失最小。

PCA 技术的一大好处是对数据进行降维的处理。我们可以对新求出的“主元”向量的重要性进行排序,根据需要取前面最重要的部分,将后面的维数省去,可以达到降维从而简化模型或是对数据进行压缩的效果。同时最大程度的保持了原有数据的信息。

PCA 技术的一个很大的优点是,它是完全无参数限制的。在 PCA 的计算过程中完全不需要人为的设定参数或是根据任何经验模型对计算进行干预,最后的结果只与数据相关,与用户是独立的。

但是,这一点同时也可以看作是缺点。如果用户对观测对象有一定的先验知识,掌握了数据的一些特征,却无法通过参数化等方法对处理过程进行干预,可能会得不到预期的效果,效率也不高。

## 3.2 Principal Component Analysis Algorithm

主成分分析 PCA 的算法:

### Data preprocessing

Training set:  $x^{(1)}, x^{(2)}, \dots, x^{(m)}$  ←

Preprocessing (feature scaling/mean normalization):

$$\left[ \begin{array}{l} \mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)} \\ \text{Replace each } x_j^{(i)} \text{ with } x_j - \mu_j. \\ \text{If different features on different scales (e.g., } x_1 = \text{size of house,} \\ x_2 = \text{number of bedrooms), scale features to have comparable} \\ \text{range of values.} \end{array} \right.$$

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

图 26: 数据预处理

### Principal Component Analysis (PCA) algorithm

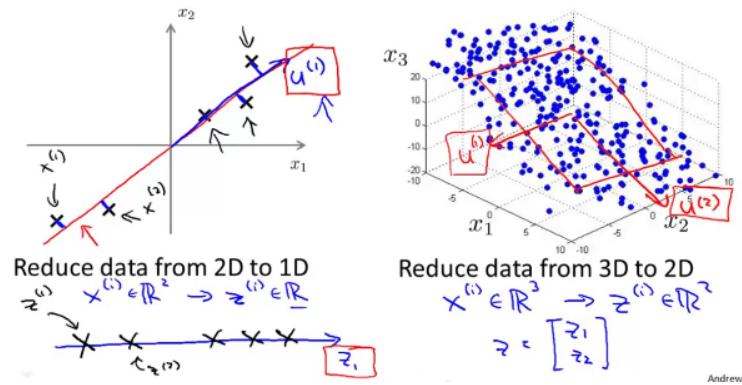


图 27: PCA算法降维

### Principal Component Analysis (PCA) algorithm

Reduce data from  $n$ -dimensions to  $k$ -dimensions

Compute "covariance matrix":

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

$\Sigma$  is  $n \times n$  matrix.  $\Sigma$  is Sigma

Compute "eigenvectors" of matrix  $\Sigma$ :

$$[U, S, V] = \text{svd}(\Sigma);$$

$[U, S, V]$  is  $n \times n$  matrix.  $\rightarrow$  Singular value decomposition.  $\text{eig}(\Sigma)$

$$U = \begin{bmatrix} | & | & | & \dots & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(k)} \\ | & | & | & \dots & | \end{bmatrix}$$

$U \in \mathbb{R}^{n \times n}$   
 $u^{(1)}, \dots, u^{(k)}$

图 28: PCA算法

PCA降维:  $n$  维到  $k$  维:

第一步是均值归一化。我们需要计算出所有特征的均值,然后令  $x_j = x_j - \mu_j$ 。如果特征是在不同的数量级上,我们还需要将其除以标准差  $\sigma^2$ 。

第二步是计算协方差矩阵(covariance matrix)  $\Sigma$ :

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

第三步是计算协方差矩阵  $\Sigma$  的特征向量(eigenvectors):

在 Octave 里我们可以利用奇异值分解(singular value decomposition)来求解,  $[U, S, V] = \text{svd}(\text{sigma})$ 。

### Principal Component Analysis (PCA) algorithm

From  $[U, S, V] = \text{svd}(\text{Sigma})$ , we get:

$$\rightarrow U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$\underbrace{\hspace{10em}}_k$   
 $x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$

$$z^{(i)} = \underbrace{\begin{bmatrix} u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{U_{\text{reduce}} \quad n \times k} x^{(i)} = \underbrace{\begin{bmatrix} -(u^{(1)})^T \\ \vdots \\ -(u^{(k)})^T \end{bmatrix}}_{k \times n} \underbrace{x^{(i)}}_{n \times 1}$$

$z \in \mathbb{R}^k$        $k \times 1$

图 29: PCA算法

$x$ 可以是训练集中的样本，可以是交叉验证集中的样本，也可以是测试集中的样本。

对于一个  $n \times n$  维度的矩阵,上式中的  $U$  是一个具有与数据之间最小投射误差的方向向量构成的矩阵。如果我们希望将数据从  $n$  维降至  $k$  维,我们只需要从  $U$  中选取前  $K$  个向量,获得一个  $n \times k$  维度的矩阵,我们用  $U_{\text{reduce}}$  表示,然后通过如下计算获得要求的新特征向量 $z^{(i)}$ :

$$z^{(i)} = U_{\text{reduce}}^T \times x^{(i)} \quad (2)$$

其中  $x$  是  $n \times 1$  维的,因此结果为  $k \times 1$  维度。注:我们不对方差特征进行处理。

### Principal Component Analysis (PCA) algorithm summary

→ After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

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

→  $[U, S, V] = \text{svd}(\text{Sigma});$

→  $U_{\text{reduce}} = U(:, 1:k);$

→  $z = U_{\text{reduce}}' * x;$

$x \in \mathbb{R}^n$        ~~$x \in \mathbb{R}^1$~~

$X = \begin{bmatrix} -x^{(1)T} \\ \vdots \\ -x^{(m)T} \end{bmatrix}$   
 $\rightarrow \text{Sigma} = (1/m) \times X' * X;$

图 30: PCA算法总结

In PCA, we obtain  $z \in \mathbb{R}^k$  from  $x \in \mathbb{R}^n$  as follows:

$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x = \begin{bmatrix} \dots & (u^{(1)})^T & \dots \\ \dots & (u^{(2)})^T & \dots \\ & \vdots & \\ \dots & (u^{(k)})^T & \dots \end{bmatrix} x$$

Which of the following is a correct expression for  $z_j$ ?

- ☐  $z_j = (u^{(k)})^T x$
- ☐  $z_j = (u^{(j)})^T x_j$
- ☐  $z_j = (u^{(j)})^T x_k$
- ☒  $z_j = (u^{(j)})^T x$

正确

## 4 Applying PCA

### 4.1 Reconstruction from Compressed Representation

原始数据的重构:

在以前的视频中,我谈论 PCA 作为压缩算法。在那里你可能需要把 1000 维的数据压缩100 维特征,或具有三维数据压缩到一二维表示。所以,如果这是一个压缩算法,应该能回到这个压缩表示,回到你原有的高维数据的一种近似。

所以,给定的,这可能 100 维,怎么回到你原来的表示  $x$ ,这可能是 1000 维的数组?

PCA 算法,我们可能有一个这样的样本。如图中样本  $x^{(1)}, x^{(2)}$ 。我们做的是,我们把这些样本投射到图中这个一维平面。然后现在我们需要只使用一个实数,比如  $z^{(1)}$ ,指定这些点的位置后他们被投射到这个一维平面。给定一个点  $z^{(1)}$ ,我们怎么能回去这个原始的二维空间呢? $x$  为 2 维, $z$  为 1 维, $z = U_{reduce}^T x$ 相反的方程为: $x_{approx} = U_{reduce} \cdot z, x_{approx} \approx x$

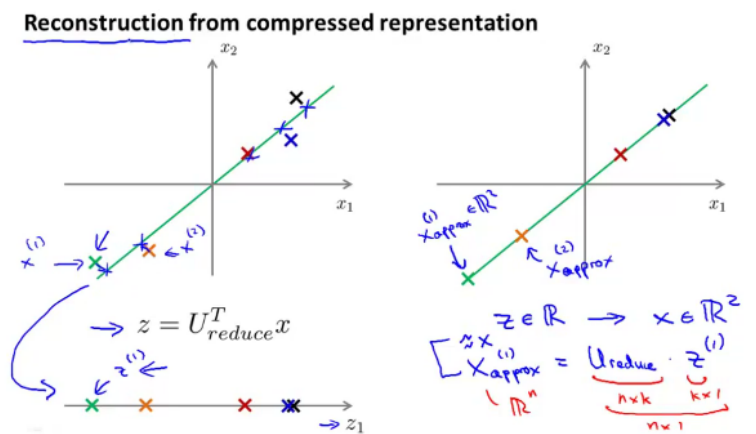


图 31: 原始数据的重构

Suppose we run PCA with  $k = n$ , so that the dimension of the data is not reduced at all. (This is not useful in practice but is a good thought exercise.)

Recall that the percent / fraction of variance retained is given by:  $\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}}$

Which of the following will be true? Check all that apply.

☒  $U_{\text{reduce}}$  will be an  $n \times n$  matrix.

正确

☒  $x_{\text{approx}} = x$  for every example  $x$ .

正确

☒ The percentage of variance retained will be 100%.

正确

☐ We have that  $\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} > 1$ .

未选择的是正确的

这已经与原始数据相当相似了。所以,这就是你从低维表示  $Z$  回到未压缩的表示。我们得到的数据的一个之间你的原始数据  $X$ ,我们也把这个过程称为重建原始数据。

当我们认为试图重建从压缩表示  $x$  的初始值。所以,给定未标记的数据集,您现在知道如何应用 PCA,你的带高维特征  $X$  和映射到这的低维表示  $Z$ 。这个视频,希望你现在也知道如何采取这些低维表示  $Z$ ,映射到备份到一个近似你原有的高维数据。

现在你知道如何实施应用 PCA,我们将要做的事是谈论一些技术在实际使用 PCA 很好,特别是,在接下来的视频中,我想谈一谈关于如何选择  $k$  (PCA算法的一个重要参数,代表着主成分的数量,簇的总数)。

## 4.2 Choosing the Number of Principal Components

主要成分分析是减少投影的平均均方误差:

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2 \quad (3)$$

训练集的方差为:

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2 \quad (4)$$

我们希望在平均均方误差与训练集方差的比例尽可能小的情况下选择尽可能小的  $K$  值。

### Choosing $k$ (number of principal components)

Average squared projection error:  $\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data:  $\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$

Typically, choose  $k$  to be smallest value so that

$$\begin{aligned} &\rightarrow \frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq \frac{0.01}{0.10} \quad \frac{(1\%)}{(10\%)} \\ &\rightarrow \text{"99\% of variance is retained"} \\ &\quad \text{95\% to 90\%} \end{aligned}$$

图 32: 选择参数 $k$

要告诉别人, 你保留了多少主成分, 我选择了参数 $k$ , 使得99%的差异性得以保留。

如果我们希望这个比例小于 1%, 就意味着原本数据的偏差有 99% 都保留下来了, 如果我们选择保留 95% 的偏差, 便能非常显著地降低模型中特征的维度了, 95%~99% 更为常用。

我们可以先令  $K=1$ , 然后进行主要成分分析, 获得  $U_{reduce}$  和  $z$ , 然后计算比例是否小于 1%。如果不是的话再令  $K=2$ , 如此类推, 直到找到可以使得比例小于 1% 的最小  $K$  值 (原因是各个特征之间通常情况存在某种相关性)。

### Choosing $k$ (number of principal components)

Algorithm:

Try PCA with  $k=1$   ~~$k=2$~~   ~~$k=3$~~   ~~$k=4$~~   $\vdots$

Compute  $U_{reduce}, z^{(1)}, z^{(2)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$

Check if  $\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01?$

$k=17$

$\rightarrow [U, S, V] = \text{svd}(\text{Sigma})$

$\rightarrow S =$

For given  $k$

$1 - \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \leq 0.01$

$\rightarrow \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.99$

图 33: 选择参数 $k$ 算法

### Choosing $k$ (number of principal components)

$[U, S, V] = \text{svd}(\text{Sigma})$

Pick smallest value of  $k$  for which

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.99$$

$k=100$

(99% of variance retained)

图 34: 选择参数 $k$ 算法总结

Previously, we said that PCA chooses a direction  $u^{(1)}$  (or  $k$  directions  $u^{(1)}, \dots, u^{(k)}$ ) onto which to project the data so as to minimize the (squared) projection error. Another way to say the same is that PCA tries to minimize:

- ☐  $\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$
- ☐  $\frac{1}{m} \sum_{i=1}^m \|x_{\text{approx}}^{(i)}\|^2$
- ☒  $\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2$
- ☐  $\frac{1}{m} \sum_{i=1}^m \|x^{(i)} + x_{\text{approx}}^{(i)}\|^2$

正确

### 4.3 Advice for Applying PCA

假使我们正在针对一张  $100 \times 100$  像素的图片进行某个计算机视觉的机器学习,即总共有 10000 个特征。

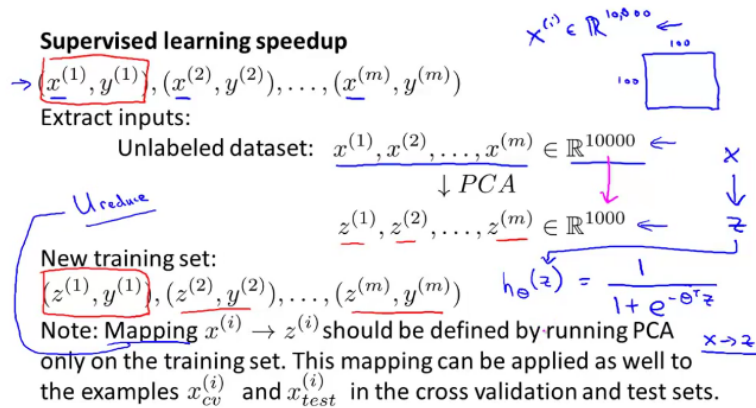


图 35: 监督学习算法加速

1. 第一步是运用主要成分分析将数据压缩至 1000 个特征
2. 然后对训练集运行学习算法
3. 在预测时,采用之前学习而来的  $U_{\text{reduce}}$  将输入的特征  $x$  转换成特征向量  $z$ ,然后再进行预测

注意: PCA 定义了从  $x$ 到 $z$ 的对应关系 这种从  $x$  到  $z$ 的对应关系只可以通过在训练集上运行 PCA 定义出来,具体来讲 这种PCA所学习出的对应关系,所做的就是计算出一系列的参数, 这就是特征缩放和均值归一化,同时也计算出这样一个降维的矩阵 $U_{\text{reduce}}$ , 但是降维矩阵 $U_{\text{reduce}}$  中的数据 就像一个, PCA所学习的参数一样,我们需要使我们的参数唯一地适应这些训练集,而不是适应我们的交叉验证或者测试集, 因此 $U_{\text{reduce}}$ 矩阵中的数据就应该只通过对训练集运行PCA来获得。

注:如果我们有交叉验证集和测试集,也采用对训练集学习而来的  $U_{\text{reduce}}$ 。



## Application of PCA

### - Compression

- Reduce memory/disk needed to store data
- Speed up learning algorithm ←

Choose  $k$  by % of variance retained

### - Visualization

$k=2$  or  $k=3$

图 36: PCA的应用

错误的主成分分析情况:一个常见错误使用主要成分分析的情况是,将其用于减少过拟合(减少了特征的数量)。这样做非常不好,不如尝试正则化处理。原因在于主要成分分析只是近似地丢弃掉一些特征,它并不考虑任何与结果变量有关的信息,因此可能会丢失非常重要的特征。然而当我们进行正则化处理时,会考虑到结果变量,不会丢掉重要的数据。

另一个常见的错误是,默认地将主成分分析作为学习过程中的一部分,这虽然很多时候有效果,最好还是从所有原始特征开始,只有在有必要的时候(算法运行太慢或者占用太多内存)才考虑采用主成分分析。

建议:一开始不要将PCA方法就直接放到算法里,先使用原始数据 $x^{(i)}$ 看看效果,只有一个原因,让我们相信算法出现了问题,那就是你的学习算法,收敛地非常缓慢,占用内存,或者硬盘空间非常大,所以你想来压缩数据,只有当你的 $x^{(i)}$ 效果不好,只有当你有证据或者充足的理由来确定 $x^{(i)}$ 效果不好的时候,那么就考虑用PCA来进行压缩数据。

### Bad use of PCA: To prevent overfitting

→ Use  $z^{(i)}$  instead of  $x^{(i)}$  to reduce the number of features to  $k < n$ . — 10000

Thus, fewer features, less likely to overfit.

Bad!

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\Rightarrow \min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \boxed{\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2} \leftarrow$$

图 37: 错误使用

Which of the following are good / recommended applications of PCA? Select all that apply.

☒ To compress the data so it takes up less computer memory / disk space

正确

☒ To reduce the dimension of the input data so as to speed up a learning algorithm

正确

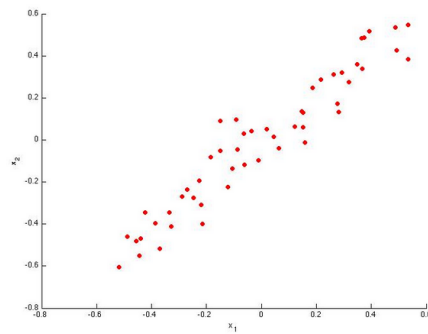
☐ Instead of using regularization, use PCA to reduce the number of features to reduce overfitting

未选择的是正确的

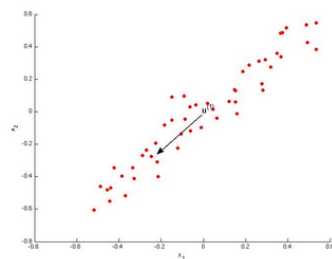
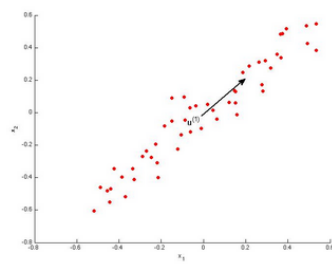
☒ To visualize high-dimensional data (by choosing  $k = 2$  or  $k = 3$ )

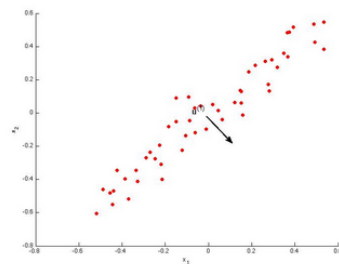
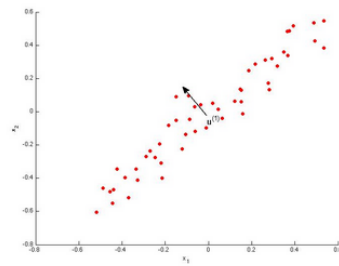
正确

Consider the following 2D dataset:



Which of the following figures correspond to possible values that PCA may return for  $u^{(1)}$  (the first eigenvector / first principal component)? Check all that apply (you may have to check more than one figure).





2.

Which of the following is a reasonable way to select the number of principal components  $k$ ?

(Recall that  $n$  is the dimensionality of the input data and  $m$  is the number of input examples.)

- ☐ Use the elbow method.
- ☐ Choose  $k$  to be the largest value so that at least 99% of the variance is retained
- ☒ Choose  $k$  to be the smallest value so that at least 99% of the variance is retained.
- ☐ Choose  $k$  to be 99% of  $m$  (i.e.,  $k = 0.99 * m$ , rounded to the nearest integer).

3.

Suppose someone tells you that they ran PCA in such a way that "95% of the variance was retained." What is an equivalent statement to this?

- ☐  $\frac{\frac{1}{m} \sum_{i=1}^m ||x^{(i)} - x_{\text{approx}}^{(i)}||^2}{\frac{1}{m} \sum_{i=1}^m ||x^{(i)}||^2} \geq 0.95$
- ☐  $\frac{\frac{1}{m} \sum_{i=1}^m ||x^{(i)} - x_{\text{approx}}^{(i)}||^2}{\frac{1}{m} \sum_{i=1}^m ||x^{(i)}||^2} \geq 0.05$
- ☒  $\frac{\frac{1}{m} \sum_{i=1}^m ||x^{(i)} - x_{\text{approx}}^{(i)}||^2}{\frac{1}{m} \sum_{i=1}^m ||x^{(i)}||^2} \leq 0.05$
- ☐  $\frac{\frac{1}{m} \sum_{i=1}^m ||x^{(i)}||^2}{\frac{1}{m} \sum_{i=1}^m ||x^{(i)} - x_{\text{approx}}^{(i)}||^2} \geq 0.95$

4.

Which of the following statements are true? Check all that apply.

- ☒ Given input data  $x \in \mathbb{R}^n$ , it makes sense to run PCA only with values of  $k$  that satisfy  $k \leq n$ . (In particular, running it with  $k = n$  is possible but not helpful, and  $k > n$  does not make sense.)
- ☐ PCA is susceptible to local optima; trying multiple random initializations may help.
- ☒ Even if all the input features are on very similar scales, we should still perform mean normalization (so that each feature has zero mean) before running PCA.
- ☐ Given only  $z^{(i)}$  and  $U_{\text{reduce}}$ , there is no way to reconstruct any reasonable approximation to  $x^{(i)}$ .

5.

Which of the following are recommended applications of PCA? Select all that apply.

- ☒ Data compression: Reduce the dimension of your input data  $x^{(i)}$ , which will be used in a supervised learning algorithm (i.e., use PCA so that your supervised learning algorithm runs faster).
- ☒ Data visualization: Reduce data to 2D (or 3D) so that it can be plotted.
- ☐ Clustering: To automatically group examples into coherent groups.
- ☐ To get more features to feed into a learning algorithm.

4.

Which of the following statements are true? Check all that apply.

- ☐ PCA can be used only to reduce the dimensionality of data by 1 (such as 3D to 2D, or 2D to 1D).
- ☒ If the input features are on very different scales, it is a good idea to perform feature scaling before applying PCA.
- ☒ Given an input  $x \in \mathbb{R}^n$ , PCA compresses it to a lower-dimensional vector  $z \in \mathbb{R}^k$ .
- ☐ Feature scaling is not useful for PCA, since the eigenvector calculation (such as using Octave's `svd(Sigma)` routine) takes care of this automatically.