5.降维方法

降维，指用一组个数为 d 的向量来代表个数为 D 的向量所包含的有用信息，其中 $d < D$。而为什么可以降维，这是因为数据有冗余，要么是一些没有用的信息，要么是一些重复表达的信息，例如一张512 x 512的图只有中心100 x 100的区域内有非0值，剩下的区域就是没有用的信息，又或者一张图是成中心对称的，那么对称的部分信息就重复了。正确降维后的数据一般保留了原始数据的大部分的重要信息，它完全可以替代输入去做一些其他的工作，从而很大程度上可以减少计算量。例如降到二维或者三维来可视化。

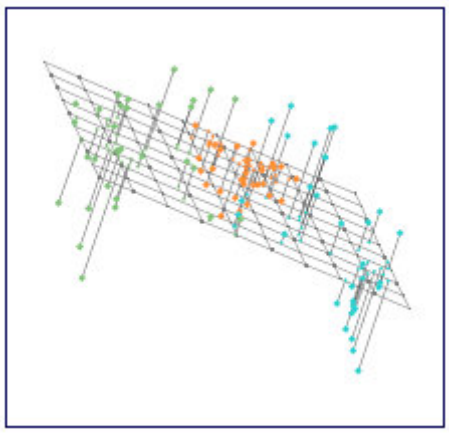
5.1 主成分分析

主成分分析（PCA）是一种最为经典的降维方法，由于不需要使用到因变量的信息，因此它属于一种无监督的学习方法。那么，主成分分析的目标应该是怎样呢?

从降维的角度来看，如果在正交空间中，存在一个超平面，那么它应该满足这样的要求：

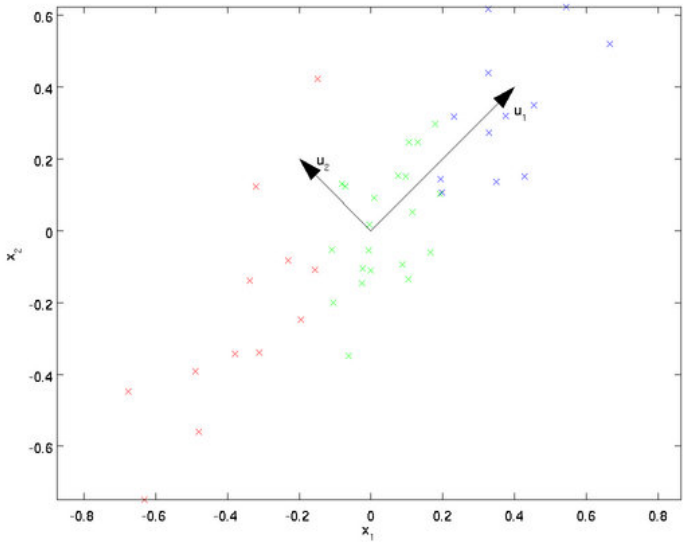
- 最近重构性：所有的样本点距离这个超平面都比较近。

关于最近重构性，我们可以这样理解。降维本身就可能造成原有信息的损失，因此我们希望找到一个与原有观测值最为接近的超平面作为投影的超平面，因为这样降维后的投影点能够尽可能地保留了原有样本点的信息。（在这里，投影就意味着降维，简单来说，就是我们希望降维后的信息损失最小）



- 最远距离性：所有的样本点在这个超平面上的投影都尽可能地分开

关于最远距离性，我们可以这样理解。我们希望可以找到一个超平面，即便是在一个低维度下，它仍然包含有足够多的数据点的变异信息在里面。或者说，原有的样本点，在这个超平面的投影下他们尽可能地分开，即投影后的样本点方差最大化。



值得注意的是，无论基于以上哪个要求出发，最后得到的结果是一样的。从个人角度，投影过程就是信息损失过程，距离越短，信息保留越多。

“降维当然意味着信息的丢失，不过鉴于实际数据本身常常存在的相关性，我们可以想办法在降维的同时将信息的损失尽量降低。举个例子，假如某学籍数据有两列 M 和 F，其中 M 列的取值是如果此学生为男性取值1，为女性取值0；而 F 列是学生为女性取值1，男性取值0。此时如果我们统计全部学籍数据，会发现对于任何一条记录来说，当 M 为1时 F 必定为0，反之当 M 为0时 F 必定为1。在这种情况下，我们将 M 或 F 去掉实际上没有任何信息的损失，因为只要保留一列就可以完全还原另一列。

当然上面是一个极端的情况，在现实中也许不会出现，不过类似的情况还是很常见的。例如淘宝店铺的数据，从经验我们可以知道，“浏览量”和“访客数”往往具有较强的相关关系，而“下单数”和“成交数”也具有较强的相关关系。这里我们非正式的使用“相关关系”这个词，可以直观理解为“当某一天这个店铺的浏览量较高（或较低）时，我们应该很大程度上认为这天的访客数也较高（或较低）”。后面的章节中我们会给出相关性的严格数学定义。 ”

我们从最远距离性这个角度出发。假定我们先对数据样本进行中心化。即$\sum_{i=0} x_i = 0$ （对属性进行中心化）。

方差计算

$(w^T X)(w^T X)^T = w^T X X^T w$，同时因为保证结果 w 为标准正交基，有限制条件 $w^T w = I$，我们要最大化方差，因此利用朗格朗日乘子法：

$$w^T X X^T w + \lambda(I - w^T w)$$

上式对 w 求偏导（矩阵求导可参考3.8章）有：

$$X X^T w = \lambda w$$

从上式可以看出，只需要对协方差矩阵 $X X^T$ 矩形特征值分解，将能够得到特征值排序 $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_m$ 接下来，只要去前 d 个特征值对应的特征向量就能获得主成分分析的解。

一般如何选取特征值数量，我们可以设定信息保留阈值 a，例如90%，，即找到使得下式成立的最小的 d 的值：

$$\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq \alpha$$

注：可进一步阅读：

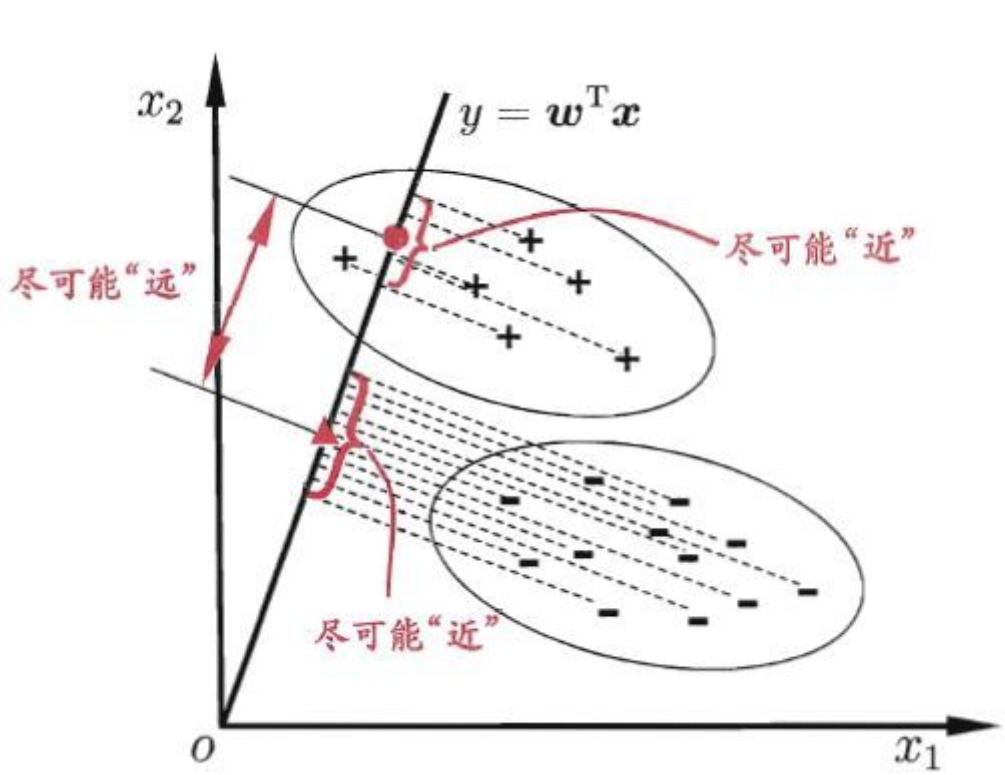
- 主成分分析PCA的数学原理:https://blog.csdn.net/qq_26593881/article/details/51425858;

- 统计学习方法（第2版）：第16章 主成分分析法。

5.2 线性判别

线性判别分析(Linear Discriminant Analysis, LDA)是另外一种线性降维方法。与主成分分析法不同，线性判别分析是一种有监督学习，也就意味着线性判别分析需要使用带标签的数据进行训练来完成降维。其次，LDA使用了标签信息，它希望同一类样本尽可能近，不同类样本尽可能远。而由于追求的目标不一样，PCA和LDA的投影方向并不一致。

以二分类情况为例，当样本的类别只有两类(0和1)且从d维映射至一维的问题，如下图所示



不同类样本间的投影距离为：

$$\left\|w^T\mu_0 - w^T\mu_1\right\|^2$$

不同类的投影协方差之和为：

$$w^T(\sum_0 + \sum_1)w$$

此时，LDA的优化目标函数为：

$$\begin{aligned} J &= \frac{\left\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\right\|^2}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}} \\ &= \frac{\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right)\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1\right)\boldsymbol{w}} \end{aligned}$$

其中 μ_i 为第 i 类样本的中心点，$\sum i$表示第 i 类样本的协方差矩阵，即 $(x-\mu i)(x-\mu i)^T$; w 即需要找到的最佳映射方式，即投影向量。

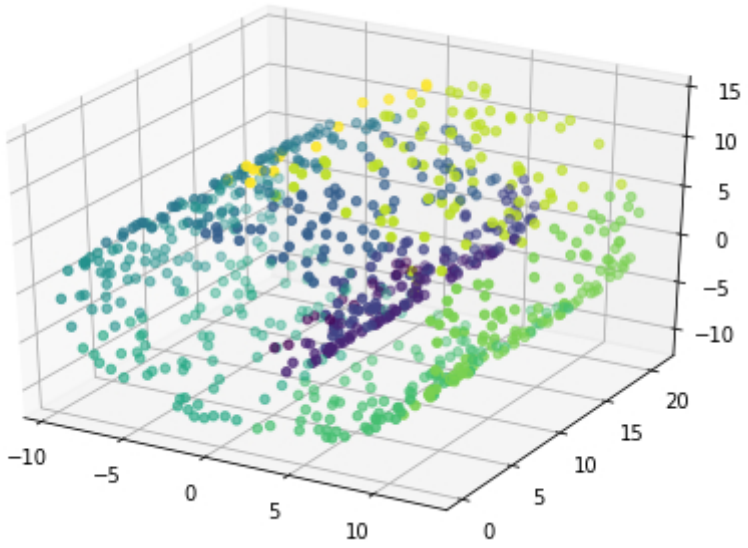
5.3 非线性提取

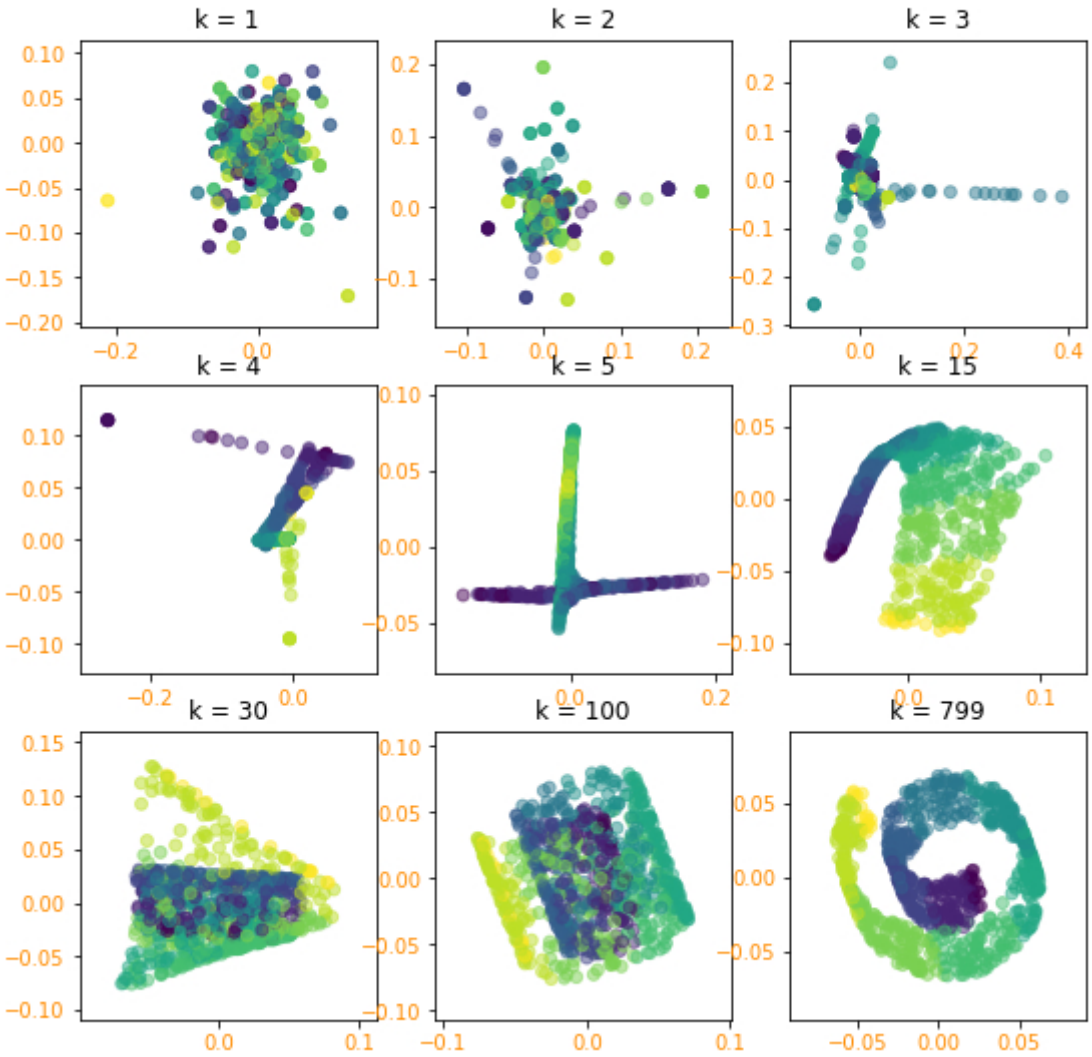
LLE（Locally Linear Embedding-局部线性嵌入）是一种非线性降维算法，和传统的PCA，LDA等**关注样本方差**的降维方法相比，LLE关注于降维时**保持样本局部的线性特征（保持原有拓扑结构）**，能够使降维后的数据较好地保持原有的流形结构 。LLE用局部线性反映全局的非线性的算法，并能够使降维的数据保持原有数据的拓扑结构。（在流形上使用局部线性，并用有限局部样本的互相线性表示，得到几何特性的构造权重矩阵，在低维下找到满足高维时样本间构造权重的样本集） 。

LLE算法认为每一个数据点都可以由其近邻点的线性加权组合构造得到。**算法的主要步骤分为三步**：

1. 寻找每个样本点的k个近邻点；
2. 由每个样本点的近邻点计算出该样本点的局部重建权值矩阵；
3. 由该样本点的局部重建权值矩阵和其近邻点计算出该样本点的输出值。

以瑞士卷为例，在 k 选取不同的情况下，降维后样本的局部关系效果有所差异，k 即我们搜索样本的近邻的个数，k 个数越大，则建立样本局部关系的时间会越大，也就意味着算法的复杂度会增加。当然 k 个数越大，则降维后样本的局部关系会保持的更好。一般来说，如果算法运行时间可以接受，我们可以尽量选择一个比较大一些的 k 。





算法推导:

对于LLE算法，我们首先要确定邻域大小的选择，即我们需要多少个邻域样本来线性表示某个样本。假设这个值为 k 。我们可以通过和KNN一样的思想通过距离度量比如欧氏距离来选择某样本的 k 个最近邻。

在寻找到某个样本的 x_i 的 k 个最近邻之后我们就需要找到 x_i 和这 k 个最近邻之间的线性关系，也就是要找到线性关系的权重系数。找线性关系，这显然是一个回归问题。假设我们有 m 个 n 维样本 $\{x_1, x_2, \ldots, x_m\}$，我们可以用均方差作为回归问题的损失函数，即:

$$J(w) = \sum_{i=1}^{m} \left\| x_i - \sum_{j \in Q(i)} w_{ij} x_j \right\|_2^2$$

其中，$Q(i)$ 表示 i 的 k 个近邻样本集合。一般我们也会对权重系数 w_{ij} 做归一化的限制，即权重系数需要满足:

$$\sum_{j \in Q(i)} w_{ij} = 1$$

对于不在样本 x_i 邻域内的样本 x_j ，我们令对应的 $w_{ij} = 0$，这样可以把 w 扩展到整个数据集的维度。

也就是我们需要通过上面两个式子求出我们的权重系数。一般我们可以通过矩阵和拉格朗日乘子法来求解这个最优化问题。

对于第一个式子，我们先将其矩阵化:

$$
\begin{aligned}
J(W) &= \sum_{i=1}^{m} \left\| x_i - \sum_{j \in Q(i)} w_{ij} x_j \right\|_2^2 \\
&= \sum_{i=1}^{m} \left\| \sum_{j \in Q(i)} w_{ij} x_i - \sum_{j \in Q(i)} w_{ij} x_j \right\|_2^2 \\
&= \sum_{i=1}^{m} \left\| \sum_{j \in Q(i)} w_{ij} \left(x_i - x_j \right) \right\|_2^2 \\
&= \sum_{i=1}^{m} W_i^T \left(x_i - x_j \right) \left(x_i - x_j \right)^T W_i
\end{aligned}
$$

其中，$W_i = (w_{i1}, w_{i2}, \ldots, w_{ik})^T$ 。

我们令矩阵 $Z_i = (x_i - x_j)(x_i - x_j)^T, j \in Q(i)$ ，则第一个式子进一步简化为 $J(W) = \sum_{i=1}^{k} W_i^T Z_i W_i$ ，对于第二个式子，我们可以矩阵化为：

$$
\sum_{j \in Q(i)} w_{ij} = W_i^T l_k = 1
$$

其中，l_k 为 k 维全1向量。

现在我们将矩阵化的两个式子用拉格朗日乘子法合为一个优化目标：

$$
L(W) = \sum_{i=1}^{k} W_i^T Z_i W_i + \lambda \left(W_i^T 1_k - 1 \right)
$$

对 W 求导并令其值为0，我们得到

$$
2 Z_i W_i + \lambda 1_k = 0
$$

即我们的

$$
W_i = \lambda' Z_i^{-1} 1_k
$$

其中 $\lambda' = -\frac{1}{2}\lambda$ 为一个常数。利用 $W_i^T 1_k = 1$ ，对 W_i 归一化，那么最终我们的权重系数 W_i 为：

$$
W_i = \frac{Z_i^{-1} 1_k}{1_k^T Z_i^{-1} 1_k}
$$

现在我们得到了高维的权重系数，那么我们希望这些权重系数对应的线性关系在降维后的低维一样得到保持。假设我们的 n 维样本集 $\{x_1, x_2, \ldots, x_m\}$ 在低维的 d 维度对应的投影为 $\{y_1, y_2, \ldots, y_m\}$ ，则我们希望保持线性关系，也就是希望对应的均方差损失函数最小，即最小化损失函数 $J(Y)$ 如下：

$$
J(y) = \sum_{i=1}^{m} \left\| y_i - \sum_{j=1}^{m} w_{ij,yj} \right\|_2^2
$$

可以看到这个式子和我们在高位的损失函数几乎相同，唯一的区别是高维的式子中，高维数据已知，目标是求最小值对应的权重系数 W ，而我们在低维是权重系数 W 已知，求对应的低维数据。注意，这里的 W 以及是 m x m 维度，之前的 W 是 m x k 维度，我们将那些不在邻域位置的 W 的位置取值为0，将 W 扩充到 m x m 维度。

为了得到标准化的低维数据，一般我们也会加入约束条件如下：

$$
\sum_{i=1}^{m} y_i = 0; \quad \frac{1}{m} \sum_{i=1}^{m} y_i y_i^T = I
$$

首先我们将目标损失函数矩阵化：

$$
\begin{aligned}
J(Y) &= \sum_{i=1}^{m} \left\| y_i - \sum_{j=1}^{m} w_{ij} y_j \right\|_2^2 \\
&= \sum_{i=1}^{m} \|YI_i - YW_i\|_2^2 \\
&= \operatorname{tr}\left(Y(I-W)(I-W)^T Y^T\right)
\end{aligned}
$$

如果我们令 $M=(I-W)(I-W)^T$，则优化函数转变为最小化下式：$J(Y)=\operatorname{tr}\left(YMY^T\right)$，$tr$ 为迹函数。约束函数矩阵化为：$YY^T=mI$

如果大家熟悉谱聚类和PCA的优化，就会发现这里的优化过程几乎一样。其实最小化 $J(Y)$ 对应的 Y 就是 M 的最小的 d 个特征值所对应的 d 个特征向量组成的矩阵。当然我们也可以通过拉格朗日函数来得到这个：

$$
L(Y)=\operatorname{tr}\left(YMY^T\right)+\lambda\left(YY^T-mI\right)
$$

对 Y 求导并令其为0，我们得到 $2MY^T+2\lambda Y^T=0$，即$MY^T=\lambda Y^T$，这样我们就很清楚了，要得到最小的 d 维数据集，我们需要求出矩阵 M 最小的 d 个特征值所对应的 d 个特征向量组成的矩阵 $Y=(y_1,\ y_2,\ \ldots,\ y_d)^T$ 即可。

一般的，由于 M 的最小特征值为0不能反应数据特征，此时对应的特征向量为全1。我们通常选择 M 的第2个到第 d+1个最小的特征值对应的特征向量 $M=（y_2,y_3,\ldots,y_{d+1})$ 来得到最终的 Y。为什么 M 的最小特征值为0呢？这是因为 $W^Te=e$，得到 $|W^T-I|e=0$，由于 $e\neq 0$，所以只有 $W^T-I=0$，即 $（I-W）^T=0$，两边同时左乘 $I-W$，即可得到 $（I-W)(I-W)^Te=0e$，即 M 的最小特征值为0。

注：可进一步阅读：

- 浅谈流形学习(Manifold Learning)：https://blog.csdn.net/qq_16234613/article/details/79689681.

5.4 维数缩减

- 降维与子空间学习

在子空间中，样本密度大幅度提高，距离计算变得更为容易。

- 为什么能降维：

- [ ] 高维观测数据的低维嵌入（与学习任务密切相关的特征通常位于某个低维分布上）
- [ ] 感谢非均匀性祝福。非均匀性的祝福(Blessing of nonuniformity)：大多数应用中，样例在空间中并非均匀分布，而是集中在一个低维流形上或者附近。

5.5 正则化

首先我们要引入拟合这一词。形象的说，**拟合**就是把平面上一系列的点，用一条光滑的曲线连接起来。而过度拟合就是要求趋势线与每个点都**强行**重合在一起。过度拟合的问题通常发生在变量（特征）过多的时候。这种情况下训练出的方程总是能很好的拟合训练数据，也就是说，我们的代价函数可能非常接近于 0 或者就为 0。然而，这样的曲线千方百计的去拟合训练数据，会导致它无法泛化到新的数据样本中，以至于无法预测新样本。打个比喻就是当我需要建立好一个模型之后，比如是识别一只小狗的模型，我需要对这个模型进行训练。恰好，我训练样本中的所有训练图片都是二哈，那么经过多次迭代训练之后，模型训练好了，并且在训练集中表现得很好。基本上二哈身上的所有特点都涵括进去，那么问题来了！假如我的测试样本是一只金毛呢？将一只金毛的测试样本放进这个识别小狗的模型中，很有可能模型最后输出的结果就是金毛不是一条狗（因为这个模型基本上是按照二哈的特征去打造的）。所以这样就造成了模型过拟合，虽然在训练集上表现得很好，但是在测试集中表现得恰好相反，在性能的角

度上讲就是协方差过大（variance is large），同样在测试集上的损失函数（cost function）会表现得很大。

总的来说，过多的变量（特征），同时只有非常少的训练数据，常常会导致出现过度拟合的问题。因此为了解决过度拟合，有以下两个办法。

方法一：尽量减少选取变量的数量

具体而言，我们可以人工检查每一项变量，并以此来确定哪些变量更为重要，然后，保留那些更为重要的特征变量。至于，哪些变量应该舍弃，我们以后在讨论，这会涉及到模型选择算法，这种算法是可以自动选择采用哪些特征变量，自动舍弃不需要的变量。这类做法非常有效，但是其缺点是当你舍弃一部分特征变量时，你也舍弃了问题中的一些信息。例如，也许所有的特征变量对于预测目标都是有用的，我们实际上并不想舍弃一些信息或者说舍弃这些特征变量。

方法二：正则化

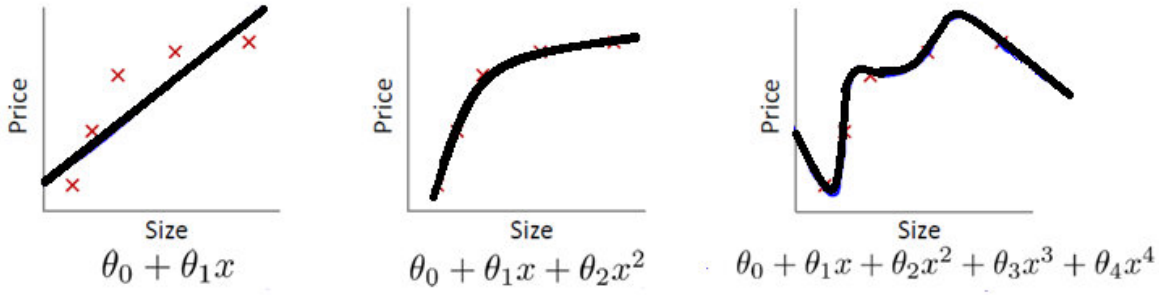
正则化中我们将保留所有的特征变量，但是会减小特征变量的数量级（参数数值的大小 $\theta(j)$）。

这个方法非常有效，当我们有很多特征变量时，其中每一个变量都能对预测产生一点影响。我们在不同的预测模型时，我们可以有很多特征变量，其中可能每一个变量都是有用的，因此我们不希望把它们删掉，这就导致了正则化概念的发生。

正则化就是对最小化经验误差函数加上约束，这样的约束可以解释为先验知识（正则化参数等价于对参数引入先验分布）。约束有引导作用，在优化误差函数的时候倾向于选择满足约束的梯度减少的方向，使最终的解倾向于符合先验知识（如一般的l-norm先验，表示原问题更可能是比较简单的，这样的优化倾向于产生参数值量级小的解，一般对应于稀疏参数的平滑解）。

同时，正则化解决了逆问题的不适定性，产生的解是存在，唯一同时也依赖于数据的，噪声对不适定的影响就弱，解就不会过拟合，而且如果先验（正则化）合适，则解就倾向于是符合真解（更不会过拟合了），即使训练集中彼此间不相关的样本数很少。

看看具体的例子，对于房屋价格预测我们可能有上百种特征。假如我们有一百个特征，但是我们却并不知道如何选择关联度更好的参数，如何缩小参数的数目等等。



我们先对该房价数据做线性回归，也就是左边第一张图。可以看到拟合数据是一条直线。但是，实际上这并不是一个很好的模型。很明显，从数据中可以看出，随着房子面积增大，住房价格的变化趋于稳定或者说越往右越平缓。因此线性回归并没有很好拟合训练数据。

再来看看第二张图，我们在中间加入一个二次项，也就是说对于这幅数据我们用二次函数去拟合。自然，可以拟合出一条曲线，事实也证明这个拟合效果很好。

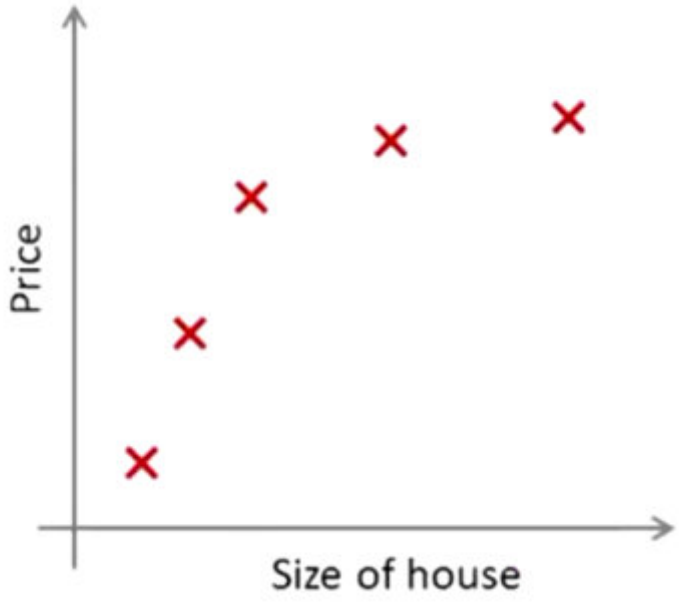
另一个极端情况是，如果在第三幅图中对于该数据集用一个四次多项式来拟合。因此在这里我们有五个参数θ_0到θ_4，这样我们同样可以拟合一条曲线，通过我们的五个训练样本，我们可以得到如右图的一条曲线。

一方面，我们似乎对训练数据做了一个很好的拟合，因为这条曲线通过了所有的训练实例。但是，这实际上是一条很扭曲的曲线，它不停上下波动。因此，事实上我们并不认为它是一个预测房价的好模型。

因此在正则化里，我们要做的事情，就是要减小我们的代价函数所有的参数值，因为我们并不知道是哪一个或哪几个要去缩小。因此，我们需要修改代价函数，在这后面添加一项，就像我们在方括号里的这项。当我们添加一个额外的正则化项的时候，我们收缩了每个参数。

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$min_\theta J(\theta)$$



顺便说一下，按照惯例，我们没有去惩罚 θ_0，因此 θ_0 的值是大的。这就是一个约定从 1 到 n 的求和，而不是从 0 到 n 的求和。但其实在实践中，这只会有非常小的差异，无论你是否包括这 θ_0 这项。但是按照惯例，通常情况下我们还是只从 θ_1 到 θ_0 进行正则化。上式中，$\lambda\sum_{j=1}^{n}\theta_j^2$ 就是一个正则化项。且 λ 在此称为正则化参数。其主要做的就是控制在两个不同的目标中的平衡关系。

第一个目标就是我们想要训练，使假设更好地拟合训练数据。我们希望假设能够很好的适应训练集。

而第二个目标是我们想要保持参数值较小。（通过正则化项）

而 λ 这个正则化参数需要控制的是这两者之间的平衡，即平衡拟合训练的目标和保持参数值较小的目标。从而来保持假设的形式相对简单，来避免过度的拟合。

对于我们的房屋价格预测来说，我们之前所用的非常高的高阶多项式来拟合，我们将会得到一个非常弯曲和复杂的曲线函数，现在我们只需要使用正则化目标的方法，那么你就可以得到一个更加合适的曲线，但这个曲线不是一个真正的二次函数，而是更加的流畅和简单的一个曲线。这样就得到了对于这个数据更好的假设。

再一次说明下，这部分内容的确有些难以明白，为什么加上参数的影响可以具有这种效果？但如果你亲自实现了正规化，你将能够看到这种影响的最直观的感受。

总结

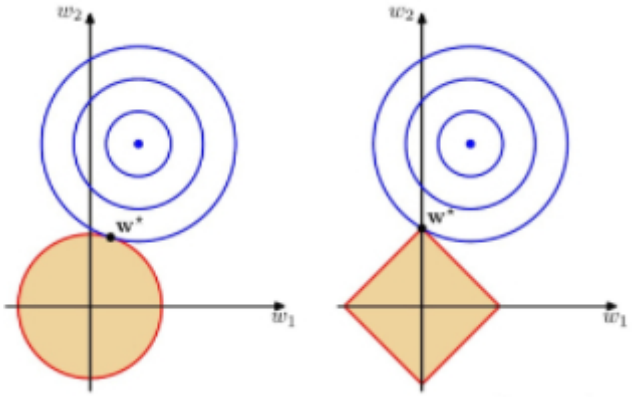
简单来说，为了防止过拟合，我们可以减少变量的数量或者采用正则化。但是因为担心损失信息，所以采用正则化，因为他能够自动降低度不重要特征的权重（系数）。一个系数比较低的模型，它总是更加平滑，他更加接近一个低阶的函数。

附：关于周志华西瓜书讲解p252总结如下所示。

关于 L_1和 L_2 范数的对比，实际上，我们把加入正则化项的损失函数加进来，就是为了降低系数权重。我们把优化函数拆开，其实就是平方误差损失项以及正则化项。显然，目标的解必然是两个项目的折中。

绘制通过两个项目的等值线，存在多个“切点”。可以知道最优解其实在“切点”中，否则如果只是交点，总是能在解域中找到一个更小的解。

对比 L_1和 L_2 正则化，可以看出 L_1 的切点更容易出现在坐标轴上（因为有棱角），L_2 则更容易出现在某个象限中。交点出现在坐标轴，意味着，有变量的系数为0，因为 L_1 正则化也就更容易获得稀疏解。从这个角度来看，我们实际上是借助正则化实现了对变量的筛选。



上图中，蓝色的圆圈部分表示问题可能的解范围，橘色的表示正则项可能的解范围。而整个目标函数（原问题+正则项）当且仅当两个解范围相切时有解。从上图可以很容易地看出，由于 $L2$ 范数解范围是圆，所以相切的点有很大可能不在坐标轴上，而由于$L1$ 范数是菱形（顶点是凸出来的），其相切的点更可能在坐标轴上，而坐标轴上的点有一个特点，其只有一个坐标分量不为零，其他坐标分量为零，即是稀疏的。所以有如下结论，$L1$ 范数可以导致稀疏解，$L2$ 范数导致稠密解。

备注：可进一步阅读

- 机器学习之正则化（Regularization）；
- 正则化及正则化项的理解：https://blog.csdn.net/gshgsh1228/article/details/52199870/；
- 正则化：https://blog.csdn.net/mrjiale/article/details/81040327；
- 《机器学习 MACHINE LEARNING》周志华：11.4章 嵌入式选择与 L_1 正则化。