主成分分析（Principal Component Analysis，PCA）是非常经典的降维算法，属于无监督降维，做机器学习的应该都有所了解。但是，除了基本的PCA推导和应用之外，还有SparsePCA、KernelPCA、TruncatedSVD等等，另外PCA和特征值、奇异值的关系以及SparsePCA和字典学习（Dict Learning，Lasso）的关系等等，也是比较有趣的事情。

还有一点，PCA的实现主要依赖于奇异值分解或者特征值分解，如何在工程上快速高效地实现SVD，如何将随机算法（Randomized Algorithms）和矩阵分析（Matrix Analysis）相融合，这些值得去深入研究一下。

本文简单梳理一下这些内容，以便有兴趣的读者查阅。主要包括几个方面：

PCA简介 & 两种推导方法
特征值分解 & 奇异值分解
PCA和Multivariate Linear Regression的关系
SparsePCA介绍 & Dict Learning
KernelPCA介绍
SVD工程实现的三种方法 & sklearn实现

1 PCA简介 & 两种推导方法

文章开始还是要简单介绍一下PCA的，如果读者对这方面已经很熟悉了，可以略过。先界定一下PCA的范畴，PCA是一种线性降维的方法，并且是利用正交变换作为映射矩阵，主要步骤是：对于一个高维空间的数据样本 $\boldsymbol{x} \in \boldsymbol{R}^{\boldsymbol{d}}$ ，利用正交矩阵 $\boldsymbol{A} \in \boldsymbol{R}^{\boldsymbol{k} \times \boldsymbol{d}}$ 将样本映射到低维空间 $\boldsymbol{A x} \in \boldsymbol{R}^{\boldsymbol{k}}$ ，其中 $\boldsymbol{k}<<\boldsymbol{d}$ 起到了降维的作用，目的是为了缓解维度灾难以及更好地对数据进行分类等等。

那么PCA该如何选择矩阵 $\boldsymbol{A}$ 呢？下面就基于两种方法来推导出PCA：最大化方差Step-by-step和最大可分性。实际上，两种都是基于最大化方差的思想，当然这也是PCA最主要的目标。还有一种解释是最近重构性，这里不多介绍，原理相差不多。

在进一步介绍之前，先给出一些符号表示。我们用 $\boldsymbol{X}=\left[\boldsymbol{x}_{\mathbf{1}} ; \boldsymbol{x}_{\mathbf{2}} ; \cdots ; \boldsymbol{x}_{\boldsymbol{n}}\right]^{\boldsymbol{T}}$ 来表示数据样本矩阵，每个数据样本 $\boldsymbol{x}_{\boldsymbol{i}} \in \boldsymbol{R}^{\boldsymbol{d}}$ 是一个列向量，并且我们假设数据样本矩阵已经做过零均值化处理，即 $\sum_{i=1}^{n} \boldsymbol{x}_{\boldsymbol{i}}=\mathbf{0}$ ，后面会解释为什么要零均值化。我们要求解的目标就是选择一个正交矩阵 $\boldsymbol{A} \in \boldsymbol{R}^{\boldsymbol{k} \times \boldsymbol{d}}, \boldsymbol{A} \boldsymbol{A}^{\boldsymbol{T}}=\boldsymbol{I}^{\boldsymbol{k} \times \boldsymbol{k}}$ ，将数据样本从高维空间映射到低维空间，满足一定性质。

1.1 最大化方差Step-by-step

考虑下面数据样本为二维的情形，即d = 2，在Figure 1中，数据样本呈现椭圆分布，我们现在要把二维降为一维k = 1，相当于是选择一个单位向量 $\boldsymbol{w} \in \boldsymbol{R}^{\mathbf{2}}$ ，将 所有的数据点投影到这个向量上来，如图中所示，画出的四个红色带箭头的线条代表的是候选向量，那么要选哪一个作为最优的 $\boldsymbol{w}$ 呢？

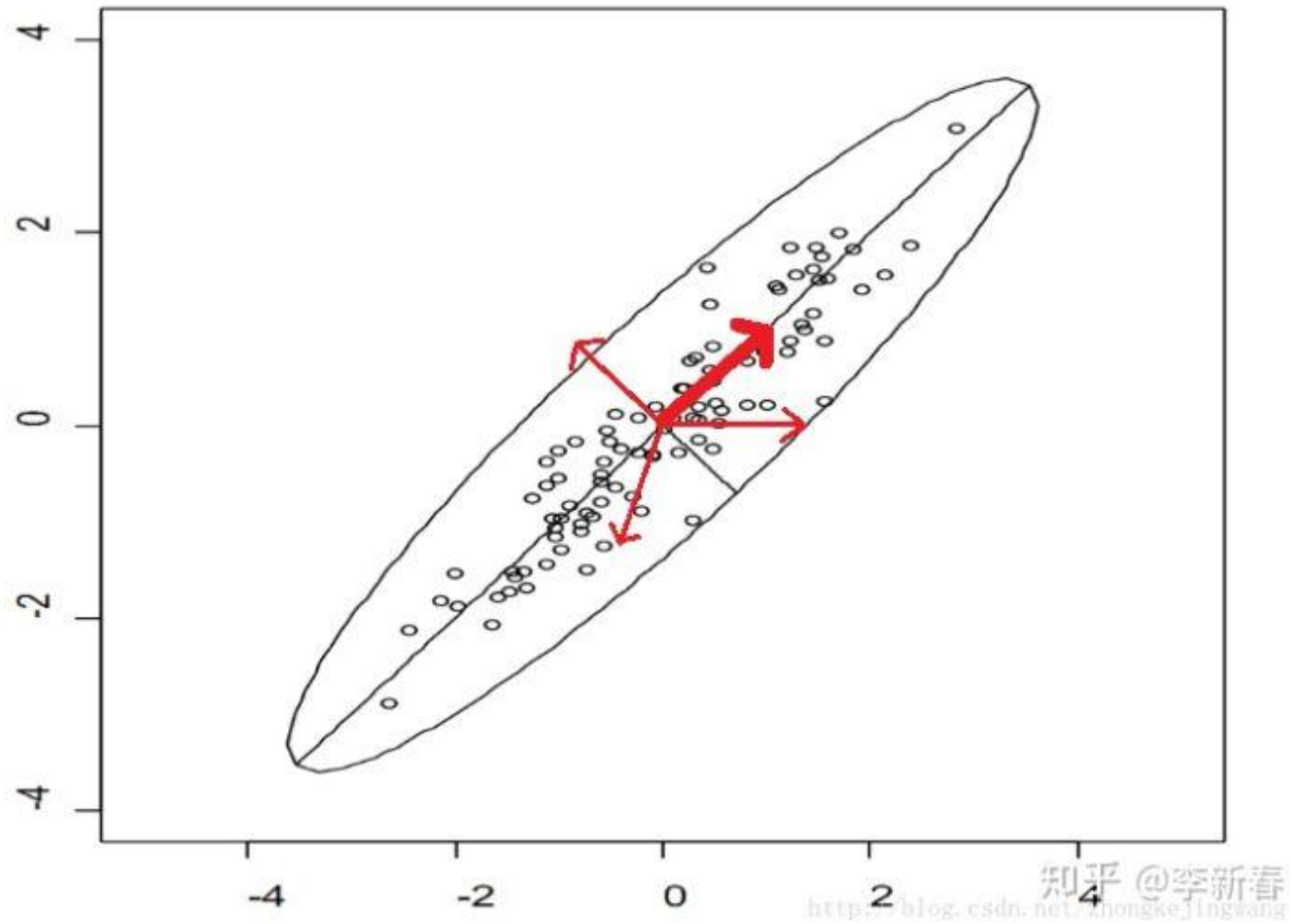

Figure 1：PCA在二维情景下的解释

PCA的假设偏好是：数据在低维空间越分散越好。假设我们选择了Figure 1中朝向为左上方的向量，那么将数据点全部投影到这个方向上面，那么就会导致数据点都“拥挤”在了一起，对后续分类或其它任务造成了很不友好的影响。所以，PCA选的是图中朝向右上方比较粗的那个作为最合适的 w ，这样得到的映射后的数据点很分散，对后续任务有利。

那么回到多维情况下，假设只将数据降为一维，那么用数学公式如何描述呢？

考虑每一个样本点 $x_i \in R^d$ ，将其投影到单位向量 $w \in R^d, ||w||_2^2 = 1$ 上，那么得到一个值 $w^T x_i$ 。对于n个样本点就有n个值，使其方差最大即可达到分散的效果，所以我们最大化

$$Var(w^T x_i) = \frac{1}{n} \sum_{i=1}^{n} \left(w^T x_i - mean(w^T x_i)\right)^2$$

其中，由于数据矩阵已经零均值化，所以投影后的数据点的均值也为零，那么最后的结果就是

$$\max_{||w||_2^2=1} \sum_{i=1}^{n} (w^T x_i)^2 = \|Xw\|_2^2 = w^T X^T X w$$

上式可以等价地转化为下面的结果

$$w = \arg\max \frac{w^T X^T X w}{w^T w}$$

上述优化问题的右边是瑞利商（Rayleigh quoient），最大值为矩阵 $X^T X$ 的最大特征值，相对应的最优 w 为最大特征值对应的单位特征向量。

到上面为止，我们知道如果想把数据降为一维，并且使得映射后的数据点尽可能分散，那么选择 $X^T X$ 的最大特征值对应的单位向量作为映射的方向即可。那么如果还想继续选择第二维呢？

PCA假设矩阵 $A \in R^{k \times d}$ 正交，$AA^T = I^{k \times k}$ 。即：k个d维的投影向量互相正交，那么我们现在要选取第二维，第二维的投影向量 $v \in R^d$ 需要满足 $||v||_2^2 = 1, w^T v = 0$ 。其中第二个条件暗示我们可以这么做：对于一个样本点 x_i ，将其分为两个分量，一个沿着 w 方向，一个垂直于 w 方向，即：$x_i = w^T x_i * w + (x_i - w^T x_i * w)$ 。容易验证括号里面的 $x_i - w^T x_i * w$ 与 w 垂直：

$$w^T (x_i - w^T x_i * w) = w^T x_i - w^T x_i w^T w = 0$$

那么对于任意的 $w^T v = 0$ ，有 $v^T x_i = v^T (x_i - w^T x_i * w)$ 。这一步的直观解释可以参考Figure 2，图中蓝色线条表示一个样本点，红色的是 w ，紫色线条代表 $x_i - w^T x_i * w$ ，黑色线条代表所有可能的 v （虽然在二维情况下已经确定了只有一个选择）。从图中可以解释，蓝色和紫色的向量到黑色向量的投影是一样的，所以我们可以在新的数据样本 $x_i - w^T x_i * w$ 上选择使得映射后数据点分散的投影向量，从而不用考虑 $w^T v = 0$ 这个约束条件。

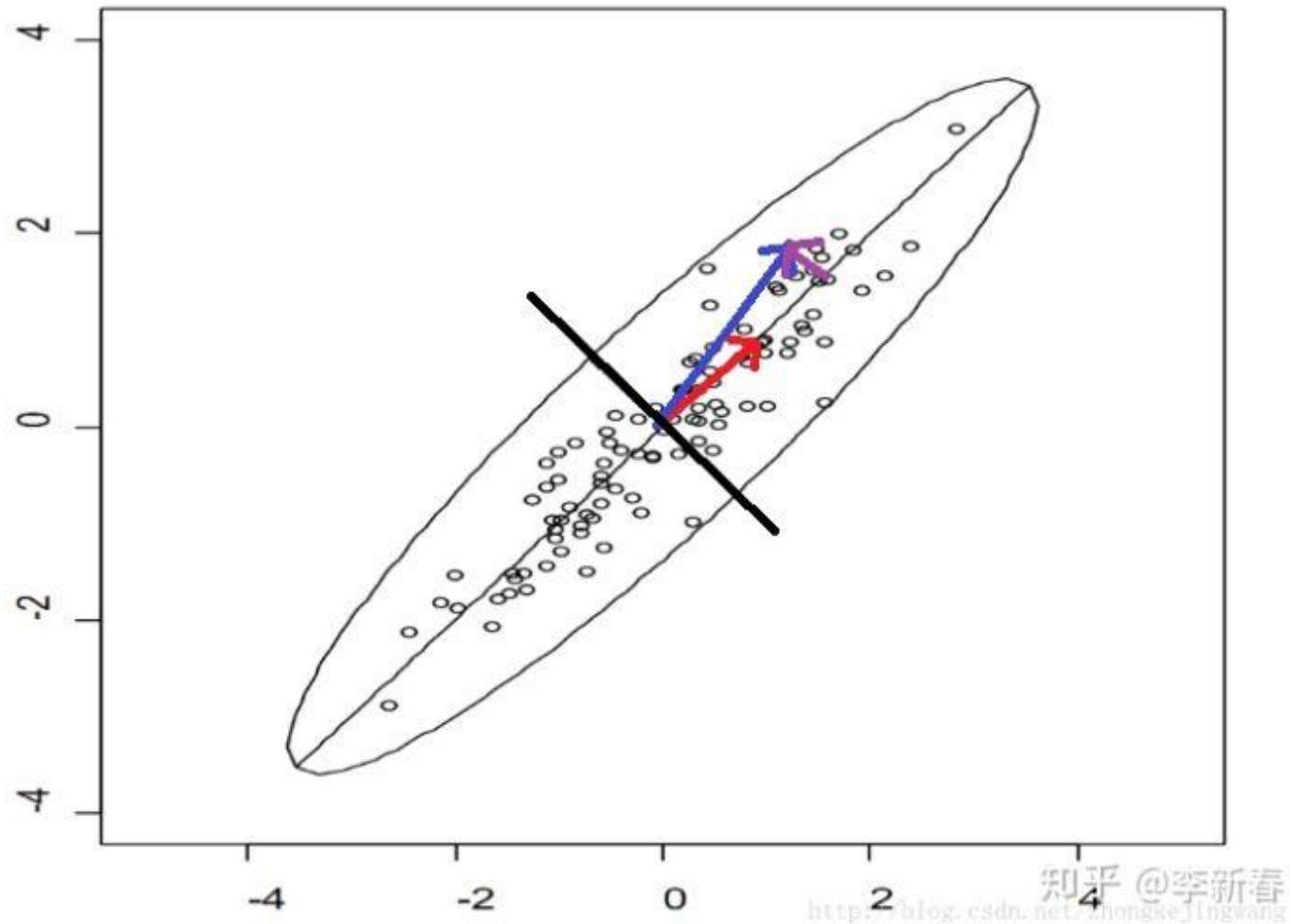

Figure 2 : PCA第二维选取示意图

所以，在求第二个方向时，可以先用原数据矩阵 X 减去与 w 平行的部分，得到与 w 垂直部分的数据 $\hat{X} = X - Xww^T$ ，然后在 $\hat{X}$ 上计算下面式子：

$$v = \arg\max \frac{v^T \hat{X}^T \hat{X} v}{v^T v}$$

v 的最优解是 $\hat{X}^T \hat{X}$ 最大特征值对应的单位特征向量，实际上证明也是 $X^T X$ 第二大特征值对应的单位特征向量。

我们把第 i 步得到的结果叫做第 i 个主成分（Component），那么最后就可以得到 k 个主成分，从而构成了 $A^{k \times d}$ ，得到了PCA的结果。

总结一下：利用Step-by-step的方法来求解PCA的最好解释是，先选择一个单位向量，可以使得数据样本投影到该向量之后方差最大，记作第一个主成分；然后将原数据矩阵的每一个样本向量分解成平行与垂直于第一个主成分的两个分量，在垂直分量上继续寻找主成分，依次继续直到选出第 k 个主成分。

1.2 最大可分性

通过上面逐步求解主成分的过程，可以看出主成分和矩阵 $X^T X \in R^{d \times d}$ 的特征向量有很大的关系。

首先，来说明一下 $X^T X \in R^{d \times d}$ 与 $XX^T \in R^{n \times n}$ 的区别，前者叫做协方差矩阵（Covariance matrix，对于零均值矩阵来说），后者是内积矩阵（Innerproduct matrix）。先来看前者：

$$X^T X = \sum_{i=1}^{n} x_i x_i^T$$

$X^T X$ 是由每个数据样本点的外积（Outer product）的求和得到的，当然要想看出它和协方差矩阵的关系还得换一种写法，假设 $X = [f_1; f_2; \cdots; f_d]$ ，每一个 $f_j \in R^n$ 是矩阵第 j 列，由于零均值化，那么有 $mean(f_j) = \frac{1}{n} sum(f_j) = 0$ 。那么此时

$$\left(X^T X\right)_{ij} = f_i^T f_j$$

而协方差矩阵的定义是一系列随机变量的协方差组成的矩阵，在这里有 d 个随机变量，每个随机变量的观测值为 f_j ，由协方差矩阵定义得到

$$Cov(x, y) = E\left[(x - E[x])(y - E[y])\right]$$
$$\Rightarrow$$
$$\Sigma_{ij} = Cov(i, j) = f_i^T f_j$$

第一行是协方差定义，第二行是求第 i 个与第 j 个特征维度的协方差，由于均值为零，所以得到上面

的结果。从上面可以看出 X^TX 其实就是协方差矩阵。

而 XX^T 则是内积矩阵，因为

$$(XX^T)_{ij} = x_i^T x_j$$

弄清楚这两个的关系之后，就不要再搞混了。下面基于最大可分性来推导PCA。

利用正交矩阵 $A \in R^{k\times d}$ 对数据样本矩阵 $X \in R^{n\times d}$ 进行变换得到 XA^T ，现在求每一维度的方差和，因为 X 每一列均值为0，那么 XA^T 的每一列均值仍为0，那么求其每一列的方差和，其实就可得到 $||XA^T||_F^2$ 。

那么我们最大化 $\|XA^T\|_F^2$ ，其实就是最大化 $\|X\left[w_1;w_2;\cdots;w_k\right]\|_F^2 = \sum_{j=1}^{k}||Xw_j||_2^2$ ，

这就可以看出之前的Step-by-step方差最大化的推导实际上就是直接利用 A 矩阵对数据矩阵变换后 F 范数最大化。

那么PCA的最终求解目标为，利用 $||D||_F^2 = tr(D^TD)$ ：

$$\min_A \quad -tr(AX^TXA^T)$$
$$s.t.\ AA^T = I^{k\times k}$$

利用拉格朗日乘子法得到：

$$X^TXA^T = \lambda A^T$$

可以得到 A 的每一行都是 X^TX 的特征向量，再将结果代入到目标中，可以知道要选前 k 个最大特征值对应的特征向量才可以使得目标最小。所以最终结果是对协方差矩阵进行特征值分解，选择前k个最大特征值对应的特征向量，然后变为单位向量即可。

这里可以说明为什么必须要对数据矩阵进行零均值化操作了，因为如果不进行零均值化，那么 X^TX 将不再是协方差矩阵， $\|XA^T\|_F^2$ 也不再是每个维度方差的和，所以上面推导出来的结果都不再成立，那么 X^TX 的特征向量自然也不是主成分了。

现在总结一下PCA的性质：

必须先对数据矩阵零均值化
基于正交变换（Orthogonal transformation）
变换矩阵的k个向量可以看作k个主成分（Principal component）
第一个主成分方向上的方差最大，依次递减
主成分可以通过对协方差矩阵（Covariance matrix）进行特征值分解得到

2 特征值分解 & 奇异值分解

上面说到通过对协方差矩阵进行特征值分解（Eigen Values Decomposition）可以得到主成分，实际上也可以不用求解协方差矩阵，而是直接对数据矩阵 X 进行奇异值分解（Sigular Values Decomposition）即可。

矩阵的奇异值分解是指

$$X = U\Sigma V^T$$

其中 $U \in R^{n\times n}$ ， $V \in R^{d\times d}$ 是列正交矩阵， $\Sigma \in R^{n\times k}$ 是对角元素大于等于零，其余元素都为零的矩阵，对角线上的元素为矩阵 X 的奇异值，并且依次递减。

假若我们获得了奇异值分解的结果，选择 U 的前 k 列， Σ 的左上 (k, k)的矩阵， V 的前k列，那么我们有

$$X \approx U_k \Sigma_k V_k^T$$

其中 $U_k \in R^{n\times k}$ 可以当作降维后的矩阵， $V_k^T \in R^{k\times d}$ 可以当作我们要求的矩阵 A ，也被称为主成分（Components）。

这就出现了很有趣的事情了，之前提到 A 可以通过对协方差矩阵 $X^T X$ 进行特征值分解得到，这里又说到 A 可以通过对 X 进行奇异值分解得到，那么这两者之间究竟有什么关系呢?

下面推导:

$$X^T X = \left(U\Sigma V^T\right)^T U\Sigma V^T = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$$
$$\Rightarrow$$
$$X^T XV = V\Sigma^2$$
$$\Rightarrow$$
$$X^T XV_j = \sigma_j^2 V_j$$

从上面推导可以看出矩阵 $X^T X$ 的第 j 个特征向量实际上就是 V 的第 j 列，这就是为什么 V_k^T 就是我们要寻找的 A 了。

关于特征值与奇异值，有以下关系:

V 的每一列是 $X^T X \in R^{d\times d}$ 的特征向量
U 的每一列是 $XX^T \in R^{n\times n}$ 的特征向量
$X^T X$ 与 XX^T 有相同的特征值（特征向量不同），是奇异值的平方

由于是奇异值分解，并且数据矩阵进行了零均值化，所以 $k \leq min\{n-1, d\}$ 。

3 PCA和Multivariate Linear Regression的关系

在之前一篇文章《从Lasso开始说起》一文中，有提到过Multivariate Linear Regression。那么PCA和这个有什么关系呢?

在前面的介绍中，我们知道如果想求解PCA，直接对数据矩阵 X 进行奇异值分解即可，奇异值分解的方法有很多，并且实现起来也很高效。那么有没有另外的求解PCA的方法呢？其中一种想法便是我们倒推奇异值分解的结果，利用 $A^T = V$ ，考虑降维映射这个过程：

$$Y = XA^T = XV$$

其中 $V \in R^{d \times k}$ ， $Y \in R^{n \times k}$ 。首先来理解一下这个过程的物理含义：

$$X = [f_1; f_2; \cdots; f_d]$$
$$Y = [g_1; g_2; \cdots; g_k]$$
$$\Rightarrow$$
$$g_i = \sum_{j=1}^{d} V_{ij} f_j$$

对上面式子的直观解释是：低维空间的每一列（每一维） g_i 可以通过高维空间的 d 个维度线性加权得到，系数为 V_{ij} ，这实际上就是特征提取的过程。假设 Y 已知，那么我们就可以通过线性回归来求解 V_{ij} ，并且是Multivariate Linear Regression，为什么呢？

考虑 k = 1的情况， $y = Y$ ，我们有

$$g = \sum_{j=1}^{d} v_j f_j$$
$$\Leftrightarrow$$
$$y = Xv$$

所以上式就是一个线性回归问题，而实际上要求解 k 个变量的回归，所以是Multivariate Linear Regression。

总结一下：

PCA可以看作是一个Multivariate Linear Regression问题
PCA得到的低维空间每个维度可以由高维空间维度线性加权表示
PCA可以看作是特征提取（Feature Extraction）的过程

那么问题来了，我们怎么获得 Y 呢？这样做的好处是什么呢？

4 SparsePCA介绍 & Dict Learning

上面解释了PCA可以看作是Multivariate Linear Regression，为什么这么做？为什么放着好好的奇异值分解不用而来选择这么复杂的过程呢？怎么事先得到 Y 呢？

上面说到PCA可以看作是一个特征提取的过程，对 d 个特征进行线性加权，提取出 k 个特征，可以用 $g_i = \sum_{j=1}^{d} V_{ij} f_j$ 来表示。一般来说，通过奇异值分解得到的 V 不是稀疏的，这就是说我们得

到的新的特征的解释性不是很好。所以，我们现在想让 V 的结果是稀疏的，那么新的特征就可以用原来的少数几个特征加权表示，增强了模型的解释性。这在基因分析、金融预测等任务中特别有效。这就是Sparse PCA的主要目的。而大家知道，在大多数问题中，得到稀疏表示的方法主要是L1正则化。

下面就先介绍一下Step-by-step步骤中如何得到稀疏结果，同1.1小节介绍，考虑第一个主成分 w ：

$$w = \arg\max_{w} w^T X^T X w$$
$$s.t. \quad w^T w = 1, ||w||_0 \leq t$$

首先优化目标是要在主成分上的投影的方差最大，其次是约束条件，满足向量长度为1，并且非零元素个数小于等于 t。L0正则化使得问题变为了NP-Hard问题，那么采用L1来放松约束得到：

$$w = \arg\max_{w} w^T X^T X w$$
$$s.t. \quad w^T w = 1, ||w||_1 \leq t$$

这里求解方法就可以使用Lasso的求解方法，比如LARS或Coordinate Descent，有兴趣的可以去参考我之前的文章《从Lasso开始说起》。求得第一个主成分之后，仿造 1.1 节中Step-by-step进行求解即可，这里需要注意求得的主成分因为L1约束并不一定会满足正交性质，所以得到最终结果后可以采取另外的方法将它们变为正交基向量。

下面从第 3 小节介绍的回归问题角度看看如何求解稀疏的 V 。同样地，先假设 X, Y 已知，那么下面以 $Y_i = g_i$ 为目标，以 $X = [f_1; f_2; \cdots; f_d]$ 为观测值，求稀疏的回归系数 $V_i = [V_{i1}, V_{i2}, \cdots, V_{id}]^T$ ，求解问题为：

$$V_i^* = \arg\min_{V_i} ||Y_i - XV_i||_2^2 + \lambda_1 ||V_i||_2^2 + \lambda_2 ||V_i||_1$$

这实际上和上面Step-by-step推导的结果一致，都是一个ElasticNet问题，可以用LARS-EN来求解。

那么最后一个问题是如何求解 Y 呢？简单来说，我们可以想到一个Two-Step的过程：

对 X 做SVD，得到 $X = U\Sigma V^T$ ，取 $Y = U_k \Sigma_k$
根据上面的回归问题求解得到稀疏的 V

上述过程主要是先做PCA得到数据的低维表示，然后再做L1正则化的线性回归求解稀疏系数进行模型解释。这是一个很naive的方法，虽然很直观，但是实际做起来效果并不是很好。所以论文引入了另外一种方法，基于“最近重构性”来推导。

回顾第 1.1 小节， $x_i - w^T x_i * w$ 表示的是 x_i 垂直于 w 的分量，我们如果能够使 $||x_i - w^T x_i * w||_2^2$ 尽可能小，这就意味着 $w^T x_i * w$ 在很大程度上重构了 x_i 。那么，为了推导出稀疏表示，论文引入了两套参数，即最小化 $||x_i - \alpha\beta^T x_i||_2^2$ ，所以对于第一个主成分有：

$$\hat{\alpha}, \hat{\beta} = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} ||x_i - \alpha\beta^T x_i||_2^2 + \lambda_1 ||\beta||_2^2 + \lambda_2 ||\beta||_1$$

$$s.t. \quad \alpha^T \alpha = 1$$

其中 β 是我们要求解的稀疏的主成分， α 是非稀疏的主成分。那么对于前 k 个主成分，假设 $A = [\alpha_1; \alpha_2; \cdots; \alpha_k]$ ， $B = [\beta_1; \beta_2; \cdots; \beta_k]$ ，那么有：

$$\hat{A}, \hat{B} = \arg\min_{A,B} \sum_{i=1}^{n} ||x_i - AB^T x_i||_2^2 + \lambda_1 \sum_{j=1}^{k} ||\beta_j||_2^2 + \lambda_2 \sum_{j=1}^{k} ||\beta_j||_1$$

$$s.t. \quad A^T A = I^{k \times k}$$

上述过程中 B 是我们最终要求解的稀疏主成分， $\beta_j \in R^d$ 是稀疏向量，表示低维空间的第 j 个特征可以由原来的 d 个特征进行线性加权表示，由于是稀疏向量，所以只有少数几个特征加权，可解释性强。 A 表示的是非稀疏的主成分，可以这么理解 $Y = XA, Y_j = X\alpha_j$ ，即前面介绍的求解 Y 的过程 转换为了求解非稀疏主成分 A 的过程。那么仿造上面的Two-Step，下面也有一个类似的Two-Step的过程：

固定 A ，利用L1正则化求解线性回归问题，得到 B
固定 B ，求解 A

那么主要是第二步，给定 B ，求解下面的优化问题：

$$\hat{A} = \arg\min_{A} \sum_{i=1}^{n} ||x_i - AB^T x_i||_2^2 = \arg\min_{A} ||X - XBA^T||_F^2$$

$$s.t. \quad A^T A = I^{k \times k}$$

上面的解可以利用 $X^T XB = U\Sigma V^T, \hat{A} = UV^T$ 得到最优解，这个解来自Reduced Rank Procrustes Rotation。

所以，Sparse PCA的求解就完成了。下面附上算法详细过程截图，见 Figure 3，第一步是用 $X = U\Sigma V^T$ 的 V 来初始化 A ，然后分两步迭代进行即可。

Algorithm 1. General SPCA Algorithm

1. Let $\mathbf{A}$ start at $\mathbf{V}[\,, 1:k]$, the loadings of the first k ordinary principal components.
2. Given a fixed $\mathbf{A} = [\alpha_1, \ldots, \alpha_k]$, solve the following elastic net problem for $j = 1, 2, \ldots, k$

$$\beta_j = \arg\min_{\beta} (\alpha_j - \beta)^T \mathbf{X}^T \mathbf{X} (\alpha_j - \beta) + \lambda \|\beta\|^2 + \lambda_{1,j} \|\beta\|_1$$

3. For a fixed $\mathbf{B} = [\beta_1, \cdots, \beta_k]$, compute the SVD of $\mathbf{X}^T \mathbf{X} \mathbf{B} = \mathbf{U} \mathbf{D} \mathbf{V}^T$, then update $\mathbf{A} = \mathbf{U} \mathbf{V}^T$.
4. Repeat Steps 2–3, until convergence.
5. Normalization: $\widehat{V}_j = \frac{\beta_j}{\|\beta_j\|}, j = 1, \ldots, k.$

Figure 3 : Sparse PCA算法

实际上，Sparse PCA就是在做矩阵分解，只不过是系数的矩阵分解而已，即 $X \approx UV$ ，其中 $V \in R^{k \times d}$ 表示主成分，是稀疏的， $U \in R^{n \times k}$ 表示降维后的数据矩阵。而这个过程实际上也是字典学习（Dict Learning）要做的事情。

下面来看一下字典学习的问题：

$$\min_{B,\alpha_i} \sum_{i=1}^{n} ||x_i - B\alpha_i||_2^2 + \lambda \sum_{i=1}^{n} ||\alpha_i||_1$$

求解步骤可以分为两步。第一步，固定 B ，利用Lasso求解回归问题：

$$\min_{\alpha_i} ||x_i - B\alpha_i||_2^2 + \lambda ||\alpha_i||_1$$

第二步，固定 $A = [\alpha_1; \alpha_2; \cdots; \alpha_k]$ ，求：

$$\min_{B} ||X - BA||_F^2$$

所以Sparse PCA实际上就是Dict Learning的变种，实际上在sklearn里面Sparse PCA就是利用Dict Learning来实现的。下面是sklearn中Sparse PCA的主要代码。

```
Vt, _, E, self.n_iter_ = dict_learning(X.T, n_components, self.alpha,
                                       tol=self.tol,
                                       max_iter=self.max_iter,
                                       method=self.method,
                                       n_jobs=self.n_jobs,
                                       verbose=self.verbose,
                                       random_state=random_state,
                                       code_init=code_init,
                                       dict_init=dict_init,
                                       return_n_iter=True
                                       )
self.components_ = Vt.T
```

总结一下：

Sparse PCA要求主成分是稀疏向量，增强模型可解释性
Sparse PCA低维空间的特征可以仅仅由高维空间个别特征线性加权表示
Sparse PCA实际上是一种Dict Learning的问题，是稀疏编码问题
Sparse PCA的求解可以分两步迭代求解，一步是基于Lasso或ElasticNet的Multivariate Linear Regression，一步是通过SVD获得 Y 或 A （ $Y = XA$ ）

5 KernelPCA介绍

PCA毕竟是线性模型，能力有限，那么如何增强PCA的能力呢？利用核技巧。下面先回顾PCA过程，先均值化，然后求协方差矩阵：

$$C=\frac{1}{n-1}\sum_{i=1}^{n}x_i x_i^T$$

接下来求特征值和特征向量：

$$Cw=\lambda w$$

有

$$Cw=\frac{1}{n-1}\sum_{i=1}^{n}x_i x_i^T w=\frac{1}{n-1}\sum_{i=1}^{n}(x_i^T w)x_i=\lambda w$$

所以主成分 w 可以由样本线性表示：

$$w=\sum_{i=1}^{n}\alpha_i x_i$$

假设现在新来一个样本 x ，在 w 上的投影为：

$$w^T x=\sum_{i=1}^{n}\alpha_i x_i^T x$$

假设我们现在有映射 $\Phi(x)$ ，可以将 x 映射到高维内积空间，并且有核函数 $K(x_i,x_j)=\Phi(x_i)^T\Phi(x_j)$ 。那么在高维空间做PCA就有：

$$C=\frac{1}{n-1}\sum_{i=1}^{n}\Phi(x_i)\Phi(x_i)^T$$

$$Cw=\lambda w$$

$$w=\sum_{i=1}^{n}\alpha_i\Phi(x_i)$$

$$w^T\Phi(x)=\sum_{i=1}^{n}\alpha_i\Phi(x_i)^T\Phi(x)=\sum_{i=1}^{n}\alpha_i K(x_i,x)$$

那么通过上面的式子我们知道，如果可以求出每个主成分对应的 α_i 那么，对于任意一个新来的样本，都可以利用上面最后一个式子将其投影到主成分上，并且是一个非线性变换。所以下面主要是求 α_i ：

由

$$Cw=\lambda w,\ C=\frac{1}{n-1}\Phi(X_i)\Phi(X_i)^T,\ w=\sum_{i=1}^{n}\alpha_i\Phi(x_i)$$

可以推出：

$$\Phi(x_k)^T Cw = \lambda \Phi(x_k)^T w,\ \forall k \in [1,n]$$
$$\Rightarrow$$
$$\Phi(x_k)^T \left(\frac{1}{n-1} \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T \right) \left(\sum_{i=1}^{n} \alpha_i \Phi(x_i) \right) = \lambda \Phi(x_k)^T \left(\sum_{i=1}^{n} \alpha_i \Phi(x_i) \right),\ \forall k \in [1,n]$$

整理上式得到：

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \Phi(x_k)^T \Phi(x_i)\Phi(x_i)^T \Phi(x_j)\alpha_j = (n-1)\lambda \sum_{i=1}^{n} \Phi(x_k)^T \Phi(x_i)\alpha_i,\ \forall k \in [1,n]$$
$$\Rightarrow$$
$$\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_k,x_i)K(x_i,x_j)\alpha_j = (n-1)\lambda \sum_{i=1}^{n} K(x_k,x_i)\alpha_i,\ \forall k \in [1,n]$$

那么记 $K_{ij} = K(x_i, x_j)$ 有：

$$K^2\alpha = \lambda(n-1)K\alpha$$

所以要想求 α ，只要求 K 的前 k 大的特征值对应的特征向量即可。所以Kernel PCA的主要步骤如下：

求 K ， $K_{ij} = K(x_i, x_j)$
求 K 的特征值和特征向量，即前 k 个最大特征值的特征向量为 $A \in R^{k \times n}$
保存 X, A
对于新样本 x ，计算 $y_i = \sum_{j=1}^{n} a_{ij} K(x_j, x)$ ，返回 $y = (y_1, y_2, \cdots, y_k)^T$

6 SVD工程实现的三种方法

可以看到PCA的求解，或者是Sparse PCA、Kernel PCA，都依赖于奇异值分解SVD，那么如何实现SVD呢？在sklearn里面给出了三种SVD，分别是：FullSVD，TruncatedSVD，RandomizedSVD。FullSVD是指的返回 $X = U\Sigma V^T, U \in R^{n \times n}, V \in R^{d \times d}$ ，TruncatedSVD返回的是 $X = U_k \Sigma_k V_k^T, U_k \in R^{n \times k}, V_k \in R^{d \times k}$ ，即指定了 k 的大小，选取 k 个主成分。主要是RandomizedSVD，即随机算法求解SVD。

一种RandomizedSVD是，假设我们有一个 $Q \in R^{n \times k}$ ，能近似满足 $X \approx QQ^T X$ ，那么我们做以下几步：

$$B = Q^T X \in R^{k \times d}$$

然后对 B 进行奇异值分解：

$$B = W\Sigma V^T, W \in R^{k\times k}, \Sigma \in R^{k\times d}, V \in R^{d\times d}$$

求：

$$U = QW \in R^{n\times k}$$
$$X \approx U\Sigma V^T, U \in R^{n\times k}, \Sigma \in R^{k\times d}, V \in R^{d\times d}$$

上面的过程就是先引入一个随机映射矩阵 Q ，这个 Q 有什么性质呢？将数据“特征列”映射到低维空间再映射回来近似不变。然后在对小矩阵 B 做SVD，最后得到的结果就是一个近似的SVD。

来看一下sklearn.utils.extmath里面实现的randomized_svd

```
Q = randomized_range_finder(M, n_random, n_iter,
                                 power_iteration_normalizer, random_state)

# project M to the (k + p) dimensional space using the basis vectors
B = safe_sparse_dot(Q.T, M)

# compute the SVD on the thin matrix: (k + p) wide
Uhat, s, V = linalg.svd(B, full_matrices=False)

del B
U = np.dot(Q, Uhat)
```

Q 如何求呢？论文中给出的解决方案是：

```
for it = 1:its

  Q = A*Q;

  if(it < its)
    [Q,R] = lu(Q);
  end

  if(it == its)
    [Q,R,E] = qr(Q,0);
  end

end
```


下面看一下randomized_range_finder：

```
# Generating normal random vectors with shape: (A.shape[1], size)
Q = random_state.normal(size=(A.shape[1], size))

# Perform power iterations with Q to further 'imprint' the top
# singular vectors of A in Q
for i in range(n_iter):
    if power_iteration_normalizer == 'none':
        Q = safe_sparse_dot(A, Q)
        Q = safe_sparse_dot(A.T, Q)
```

```
        elif power_iteration_normalizer == 'LU':
            Q, _ = linalg.lu(safe_sparse_dot(A, Q), permute_l=True)
            Q, _ = linalg.lu(safe_sparse_dot(A.T, Q), permute_l=True)
        elif power_iteration_normalizer == 'QR':
            Q, _ = linalg.qr(safe_sparse_dot(A, Q), mode='economic')
            Q, _ = linalg.qr(safe_sparse_dot(A.T, Q), mode='economic')

    # Sample the range of A using by linear projection of Q
    # Extract an orthonormal basis
    Q, _ = linalg.qr(safe_sparse_dot(A, Q), mode='economic')
    return Q
```