2.5. 分解成分中的信号（矩阵分解问题）

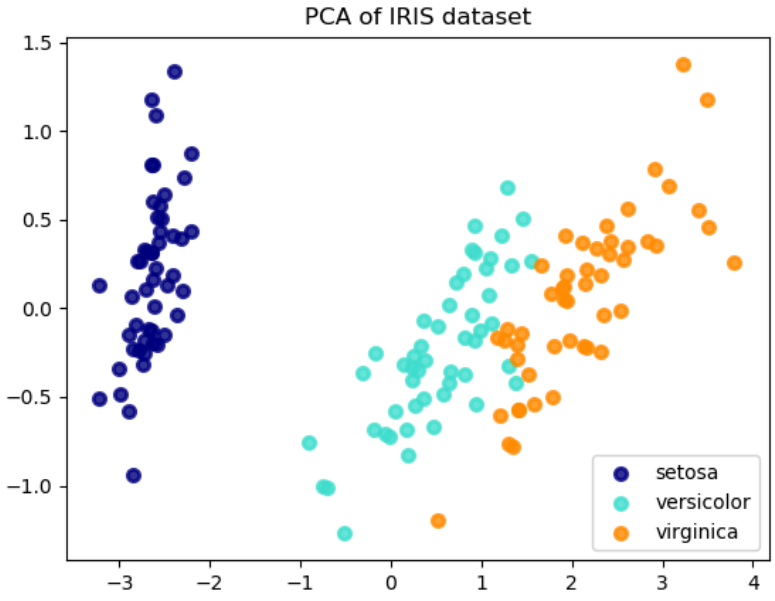
2.5.1. 主成分分析（PCA）

2.5.1.1. 准确的PCA和概率解释（Exact PCA and probabilistic interpretation）

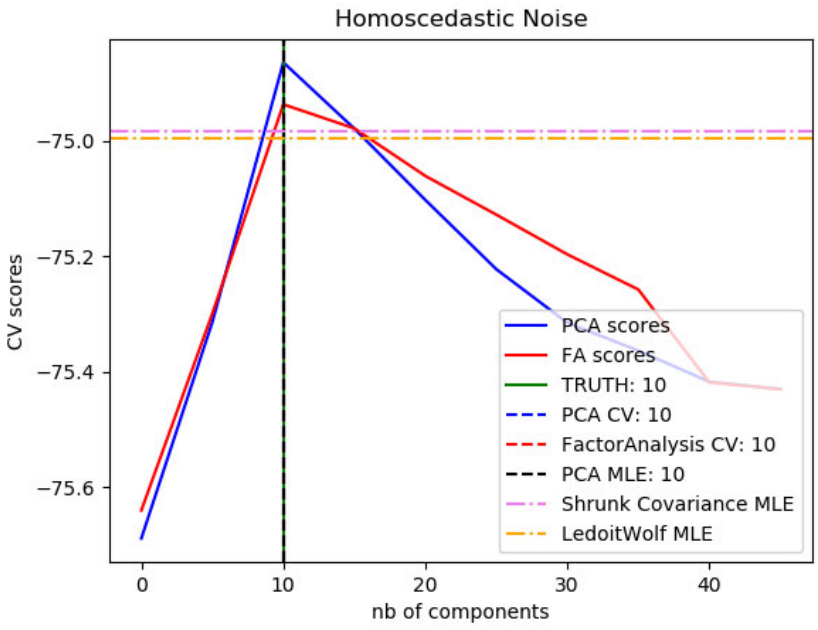
主成分分析（PCA） 是用来对一组连续正交分量的多变量数据集进行方差最大化的分解。 在 scikit-learn 中， `PCA` 通过 `fit` 方法可以拟合出 *n* 个成分来实现一个transformer对象， 并且可以将新的数据集投影到这些成分中。

在应用SVD(奇异值分解) 之前, PCA 会把输入数据的每个特征聚集，而不是缩放输入数据。可选参数 `whiten=True` 使得将数据投影到奇异空间成为可能，同时将每个分量缩放到单位方差。 如果下游模型对信号的各向同性做了强假设，这通常是有用的：例如，使用RBF内核的支持向量机算法和 K-Means 聚类算法就是这样。

以下是iris数据集的一个例子，它由4个特征组成，通过PCA降维后投影到方差最大的二维空间上：



`PCA` 对象还提供了PCA的概率解释，可以根据解释的方差给出数据的可能性。因此，PCA实现了一个可以在交叉验证中使用的评分（score）方法：



示例:

- Comparison of LDA and PCA 2D projection of Iris dataset
- Model selection with Probabilistic PCA and Factor Analysis (FA)

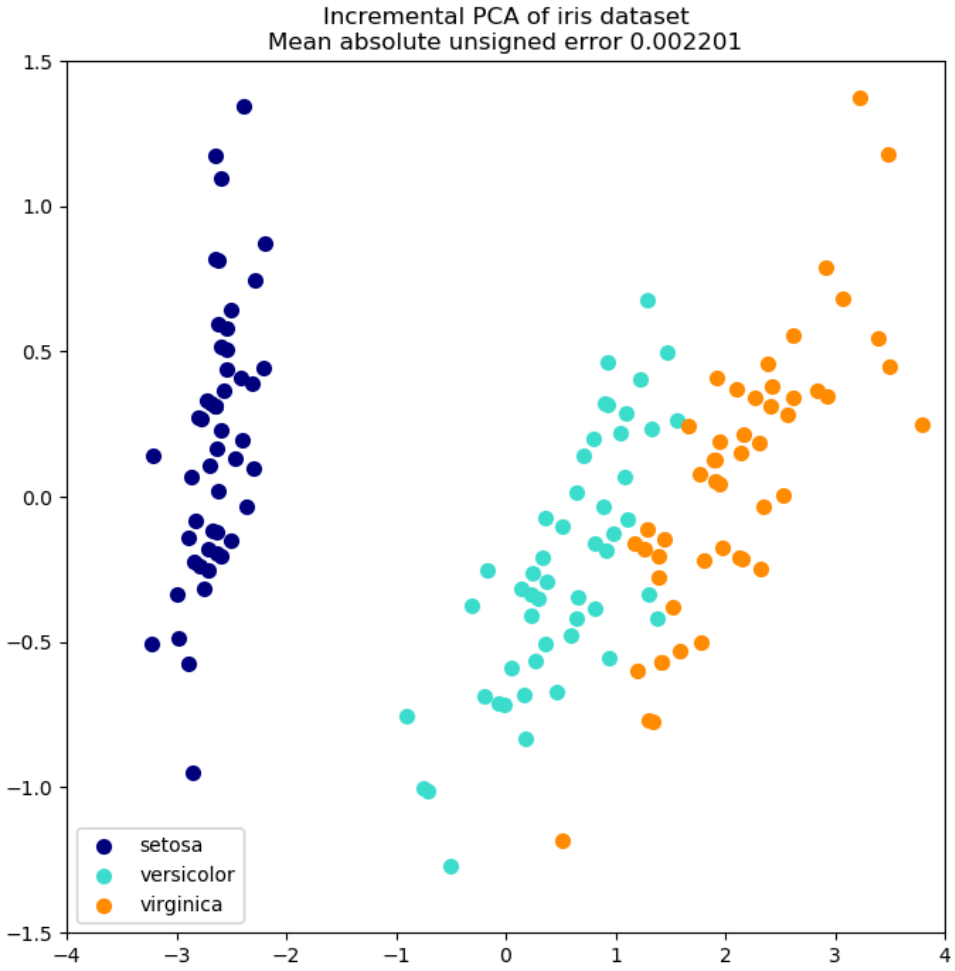
2.5.1.2. 增量PCA (Incremental PCA)

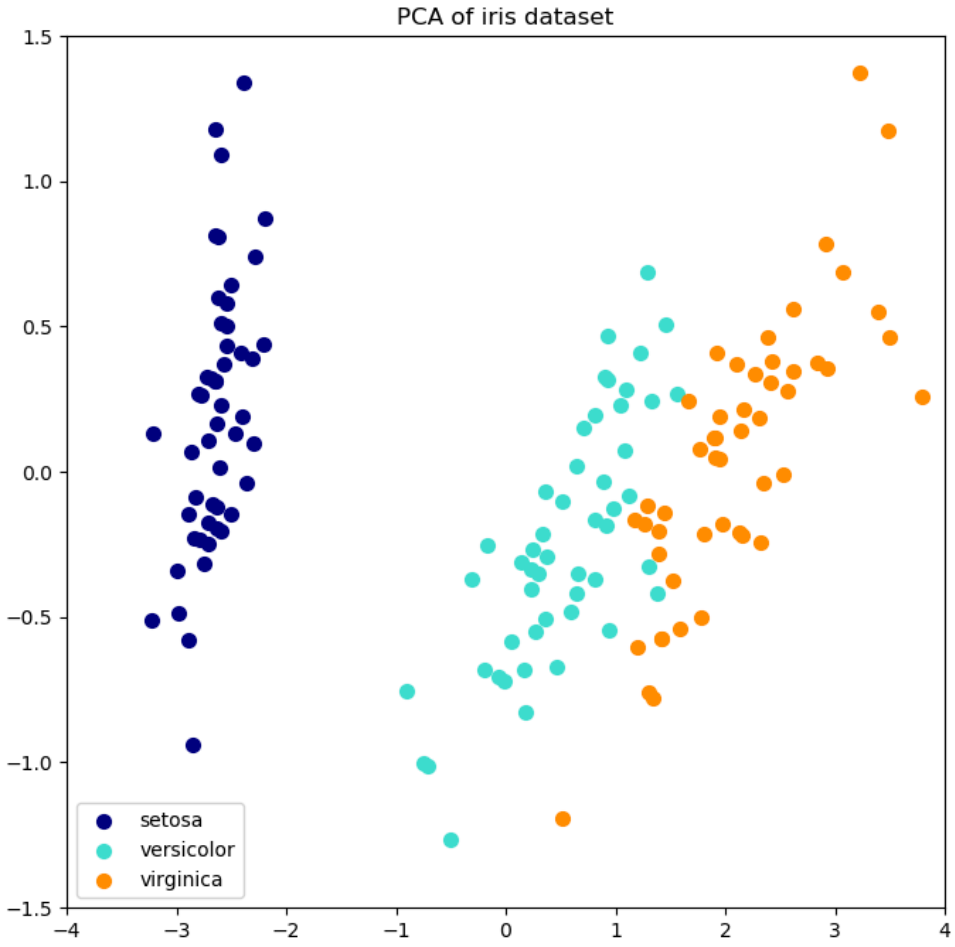
`PCA` 对象非常有用, 但是对于大型数据集有一定的限制。 最大的限制是 `PCA` 只支持批处理，这意味着所有要处理的数据必须放在主内存。 `IncrementalPCA` 对象使用一种不同的处理形式， 即允许部分计算以小型批处理方式处理数据的方法进行, 这些计算结果和 `PCA` 的结果完全匹配。 `IncrementalPCA` 可以通过以下方法实现核外主成分分析：

- 对按顺序从本地硬盘或网络数据库中获取的数据块使用 `partial_fit` 方法。
- 在稀疏矩阵或内存映射文件上使用 (通过 `numpy.memmap` 创建)fit 方法。

`IncrementalPCA` 类只存储分量和噪声方差，以便增量地更新 explained*variance_ratio* _。 这就是为什么内存使用量取决于每批处理的样本数量，而不是要在数据集中处理的样本数量。

与 `PCA` 一样， `IncrementalPCA` 在应用SVD之前会把每个特征的输入数据聚集而不是缩放输入数据。





> 示例：
>
> - Incremental PCA

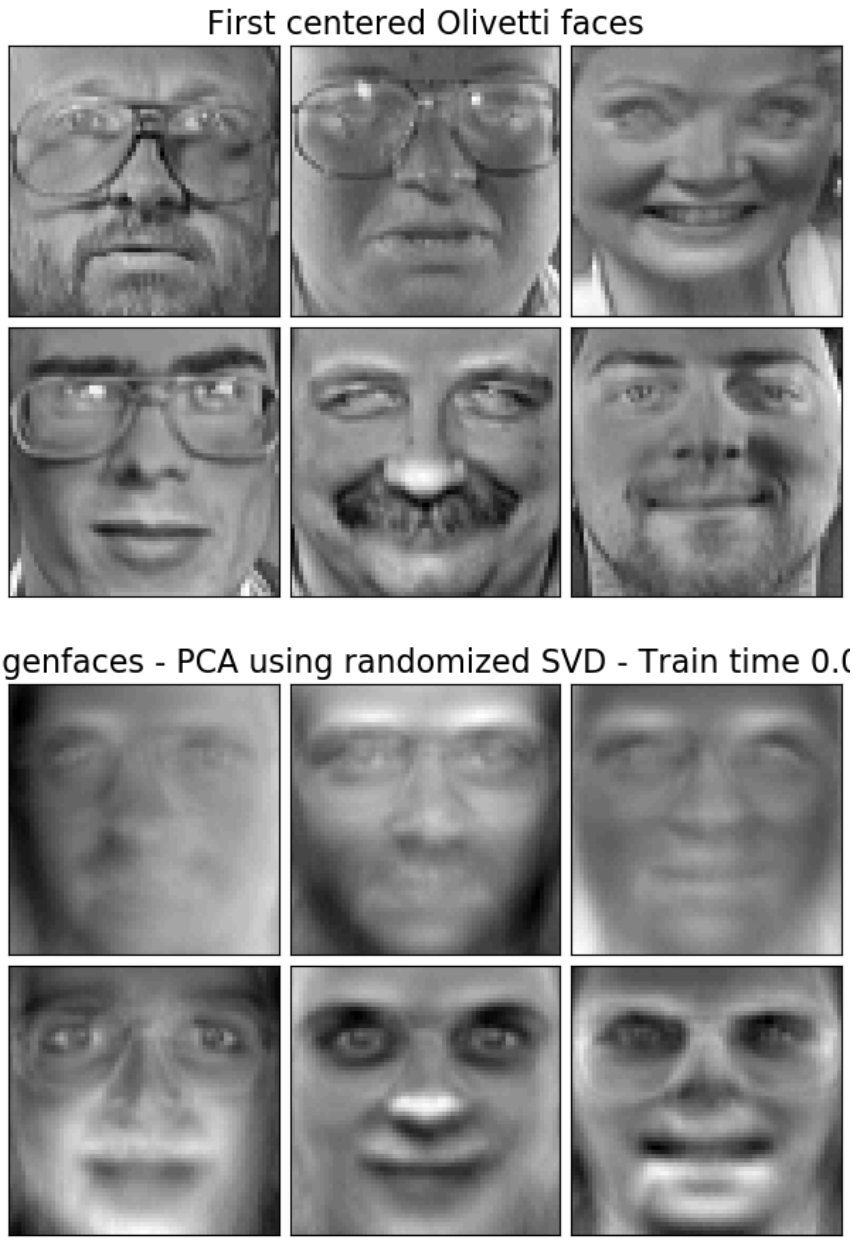
2.5.1.3. 基于随机化SVD的PCA

将数据投影到保留大部分方差信息的低维空间通常是有意义的，方法是去掉具有较低奇异值分量的奇异向量。

例如，如果我们使用64x64像素的灰度图像进行人脸识别，数据的维数为4096，在如此宽的数据上训练RBF核支持向量机是很慢的。此外，我们知道数据的固有维度比4096要低得多，因为所有的人脸图片看起来都有些相似。样本位于一个低得多的维度上(比如200维左右)。主成分分析算法可以对数据进行线性变换，同时降低数据的维数并保留大部分可解释的方差。

在这种情况下，`PCA` 使用可选参数 `svd_solver='randomized'` 是非常有用的：因为我们要放弃大部分的奇异向量，更有效的限制计算奇异向量的近似估计我们将继续执行转换。。

例如：下面显示了来自 Olivetti 数据集的 16 个样本肖像（以 0.0 为中心）。在右手边是前16个奇异向量重画的肖像。由于我们只需要大小为 $n_{samples} = 400$，$n_{features} = 64 \times 64 = 4096$ 的数据集的前16 奇异向量, 计算时间小于 1 秒。



注意：使用可选参数 `svd_solver='randomized'` ，在 `PCA` 中我们还需要给出输入低维空间大小 `n_components` 。

我们注意到，如果 $n_{max} = max(n_{samples}, n_{samples})$且$n_{min} = min(n_{samples}, n_{samples})$，对在PCA中实现的精确方法，随机 `PCA` 的时间复杂度是 $O(n_{max}^2 \cdot n_{components})$ 而不是 $(n_{max}^2 \cdot n_{min})$ 。

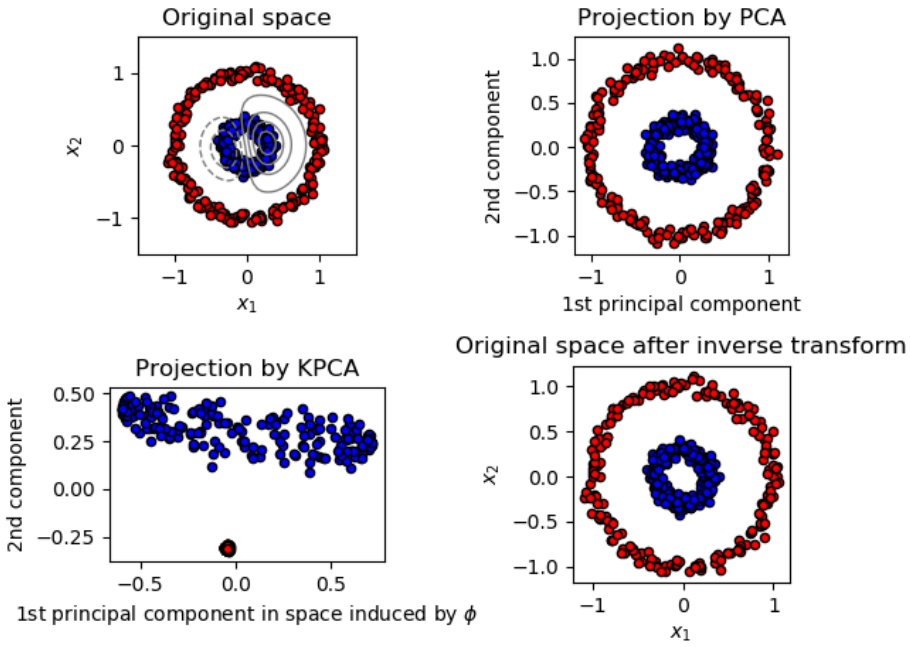
随机 `PCA` 的内存占用量和 $2 \cdot n_{max} \cdot n_{components}$成正比，而不是和精确方法的 $n_{max} \cdot n_{min}$成正比。

注意：选择参数 `svd_solver='randomized'` 的 `PCA` 的 `inverse_transform` 的实现, 并不是对应 `transform` 的精确逆变换，即使参数设置为 `whiten=False` （默认设置）

> 示例：
>
> - Faces recognition example using eigenfaces and SVMs
> - Faces dataset decompositions
>
> 参考文献：
>
> - "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions" Halko, et al., 2009

2.5.1.4. 核主成分分析（Kernel PCA）

`KernelPCA` 是 PCA 的扩展，通过使用核方法实现非线性降维 (见 Pairwise metrics, Affinities and Kernels) 。 它具有许多应用，包括去噪, 压缩和结构化预测内核依赖估计(kernel dependency estimation)。 `KernelPCA` 同时支持 `transform` 和 `inverse_transform` 。



> 示例：
>
> - Kernel PCA

2.5.1.5. 稀疏主成分分析 (SparsePCA 和 MiniBatchSparsePCA)

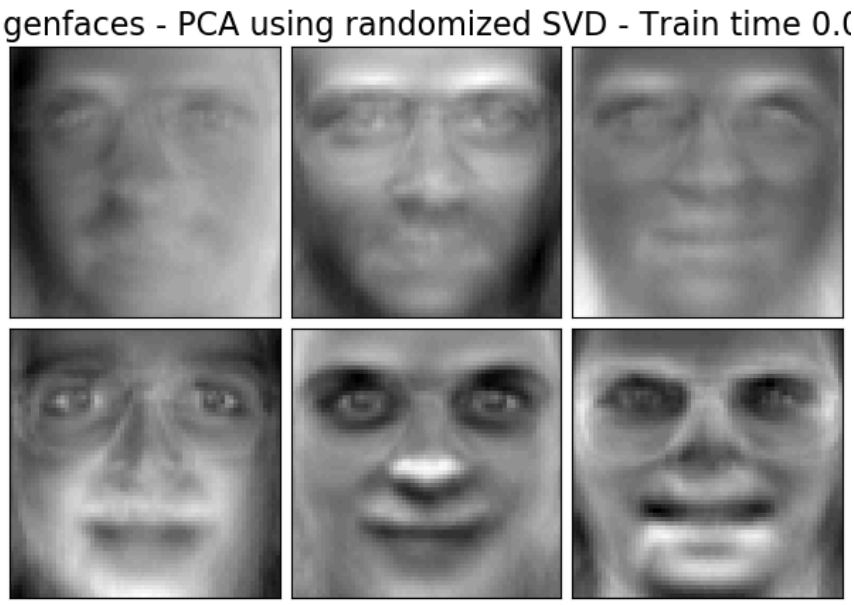
`SparsePCA` 是 PCA 的一个变体，其目标是提取一组稀疏分量集合，以最大程度地重构数据。

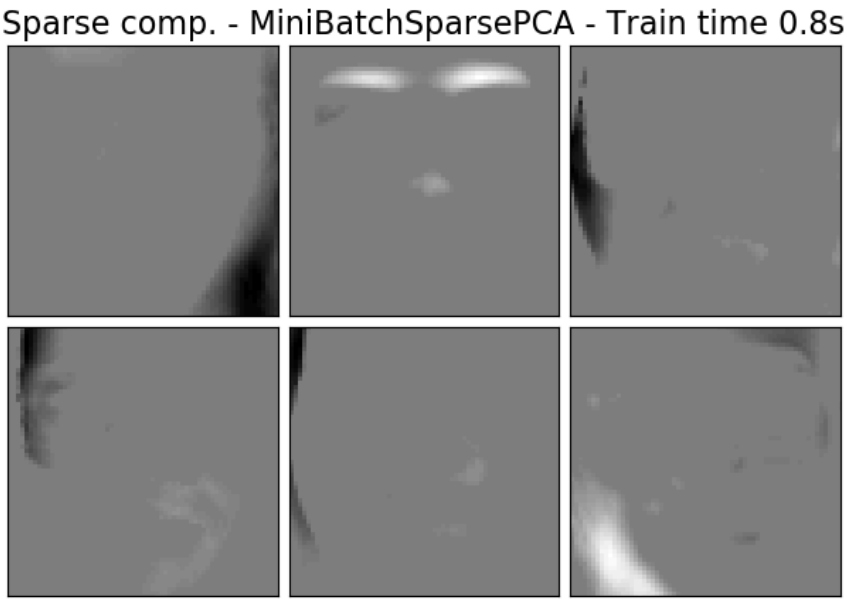
小批量稀疏 PCA (`MiniBatchSparsePCA`) 是 `SparsePCA` 的一个变体，速度更快，但精度更低。在给定迭代次数的情况下，通过迭代该组特征的小块来提高速度。

主成分分析(PCA)的缺点是，该方法提取的分量具有完全密集的表达式，即当用原变量的线性组合表示时，系数不为零。这可能使解释变得困难。在许多情况下，可以把真正的基础分量被更自然地想象为稀疏向量; 例如在人脸识别中，每个分量可能自然地映射到人脸的各个部分。

稀疏主成分产生了一种更简洁的、可解释的表示，明确地强调了哪些原始特征导致了样本之间的差异。

下面的例子演示了使用稀疏PCA从Olivetti 人脸数据集中提取的16个分量。可以看出正则化项产生了许多0。可以看出正则化项是如何产生了许多0。此外，数据的自然结构使非零系数垂直相邻。从数学上讲，该模型没有强制执行:每个分量都是一个向量，每个分量都是一个向量 $h \in R^4096$，除非人性化地将其可视化为 64x64 像素的图像，否则没有垂直相邻性的概念。下面显示的分量出现局部是数据固有结构的影响，使得这种局部模式使重建误差最小化。存在考虑到邻接性和不同结构类型的稀疏导致的规范，参见 [Jen09] 查看这些方法。有关如何使用稀疏PCA的更多细节，请参见下面的示例部分。





注意，对于计算稀疏PCA问题有许多不同的公式。 这里使用的方法基于 [Mrl09] 。所解决的优化问题是一个带是一个带有惩罚项 ℓ_1（L1范数的）的 PCA 问题（字典学习）：

$$(U^*, V^*) = \arg\min_{U,V} \frac{1}{2}||X - UV||_2^2 + \alpha||V||_1$$
$$\text{subject to } ||U_k||_2 = 1 \text{ for all } 0 \leq k < n_{components}$$

在训练样本很少时，sparsity-inducing $\ell 1$也可以避免拟合噪声。惩罚的程度(稀疏性)可以通过设置超参数 `alpha` 来调整）。值较小的值导致温和的正则化因子分解，而较大的值将许多系数收缩到零。

注意 ：

虽然按照在线算法的精神，但因为在线算法是以特征为导向，而不是以样本为导向。 `MiniBatchSparsePCA` 类不能实现 `partial_fit` 。

注意：

> 示例：
>
> - Faces dataset decompositions
>
> 参考资料：
>
> - Mrl09“Online Dictionary Learning for Sparse Coding” J. Mairal, F. Bach, J. Ponce, G. Sapiro, 2009
> - Jen09“Structured Sparse Principal Component Analysis” R. Jenatton, G. Obozinski, F. Bach, 2009

2.5.2. 截断奇异值分解和隐语义分析

`TruncatedSVD` 实现了一种奇异值分解（SVD）的变体，它只计算 k 个最大的奇异值，其中 k 是用户指定的参数。

当截断的 SVD被应用于 term-document矩阵（由 `CountVectorizer` 或 `TfidfVectorizer` 返回）时，这种转换被称为 潜在语义分析latent semantic analysis (LSA)，因为它将这些矩阵转换为低维的"语义"空间。众所周知，特别是 LSA 会对抗同义词和一词多义（它们都大致表示每个单词有多个含义）的影响, 这种影响会导致 term-document 矩阵过于稀疏，并且在余弦相似度等度量条件下表现出较差的相似性。

> 注意 : LSA也被称为潜在语义索引(LSI)，尽管严格来说它指的是用于信息检索目的的持久索引。

从数学角度来说，截断的SVD应用于训练样本X用截断的SVD 会产生一个低秩近似值X:

$$X \approx X_k = U_k \Sigma_k V_k^\top$$

在这个操作之后，$U_k \Sigma_k^\top$ 是包括 k 个特征的转换后的训练集（在 API 中被称为 `n_components` ）。

还需要转换一个测试集 X , 我们乘以 V_k :

$$X' = XV_k$$

> 注意，在自然语言处理(NLP) 和信息检索(IR) 文献中， LSA 的大多数处理都是交换矩阵 X 的坐标轴,使其具有 `n_features` × `n_samples` 的形状。 我们 在 scikit-learn API 用一种不同的方式表示 LSA, 但是找到的奇异值是相同的。

`TruncatedSVD` 与 `PCA` 非常类似，但不同之处在于矩阵 X 不需要居中。当从特征值按列（每个特征）减去 X 的均值时，在得到的矩阵上应用截断的SVD 相当于 PCA 。 实际上，，因为即使对于中等规模的文档集合， 密集化也可能填满内存。

尽管 `TruncatedSVD` 转换器可以在任何特征矩阵上工作，但还是建议在 LSA/文档处理设置中，在 tf–idf 矩阵上的原始频率计数使用它。特别地，(`sublinear_tf=True, use_idf=True`)，以使特征值更接近于高斯分布，从而补偿 LSA 对文本数据的错误假设。

> 示例:
>
> - **Clustering text documents using k-means**
>
> 参考资料:
>
> - Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze (2008), *Introduction to Information Retrieval*, Cambridge University Press, chapter 18: **Matrix decompositions & latent semantic indexing**

2.5.3. 词典学习

2.5.3.1. 带有预计算词典的稀疏编码

`SparseCoder` 对象是一种估计器，因此，此对象不实现fit方法。这种转换相当于一个稀疏编码问题。 将数据的表示为尽可能少的词典原子的线性组合。 词典学习的所有变体以尽可能少的字典原子的线性组合来寻找数据的表示。字典学习的所有变体实现以下变换方法，通过 `transform_method` 初始化参数进行控制:

- Orthogonal matching pursuit (正交匹配追踪法（**OMP**）)
- Least-angle regression (最小角度回归)
- Lasso computed by least-angle regression(最小角度回归的Lasso 计算)
- Lasso using coordinate descent (使用坐标下降的Lasso)(**Lasso**)
- Thresholding(阈值)

阈值方法速度非常快，但是不能产生精确的重建。它们在文献分类任务中被证明是有用的。对于图像重建任务，正交匹配追踪可以得到最精确、无偏的重建结果。

字典学习对象通过 `split_code` 参数提供稀疏编码结果中分离正值和负值的可能性。当使用词典学习提取将用于监督学习的特征时，这是很有用的，因为它允许学习算法将不同的权重从正加载（loading）分配给相应的负加载。

单个样本的分割编码长度为 `2 * n_components` ，并使用以下规则构造: 首先，计算长度为 `n_components` 的常规编码。然后， `split_code` 的第一个 `n_components` 条目填充常规编码向量的正部分。分割编码的另一半用编码向量的负部分填充，只有一个正号。因此， split_code 是非负的。

> 示例:
>
> - **Sparse coding with a precomputed dictionary**

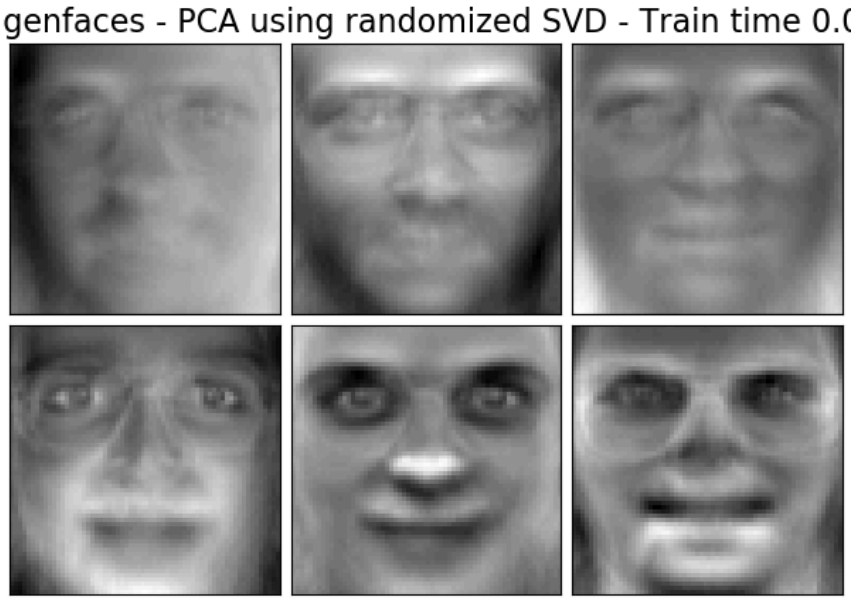
2.5.3.2. Generic dictionary learning（通用字典学习）

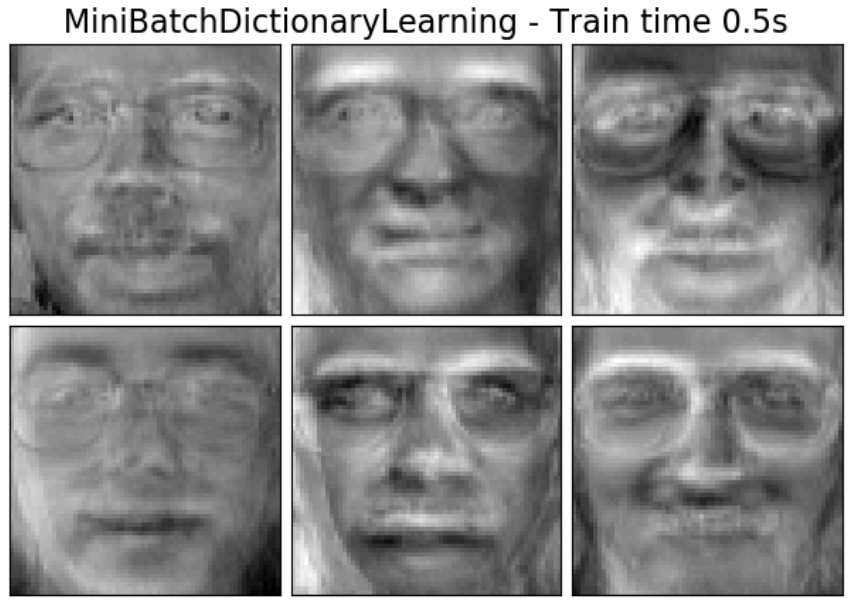
词典学习(`DictionaryLearning`) 是一个矩阵因式分解问题，相当于找到一个(通常过于完整的)字典对拟合数据进行稀疏编码。

将数据表示为过完备字典的原子的稀疏组合被认为是哺乳动物初级视觉皮层的工作方式。因此，字典学习应用于图像修补，已被证明在图像处理任务，如图像完成，修复和去噪，以及有监督识别任务中表现良好。

字典学习是一个优化问题，通过交替更新稀疏代码来解决，作为解决多个Lasso问题的解决方案，考虑字典固定，然后更新字典以最大程度拟合稀疏代码。

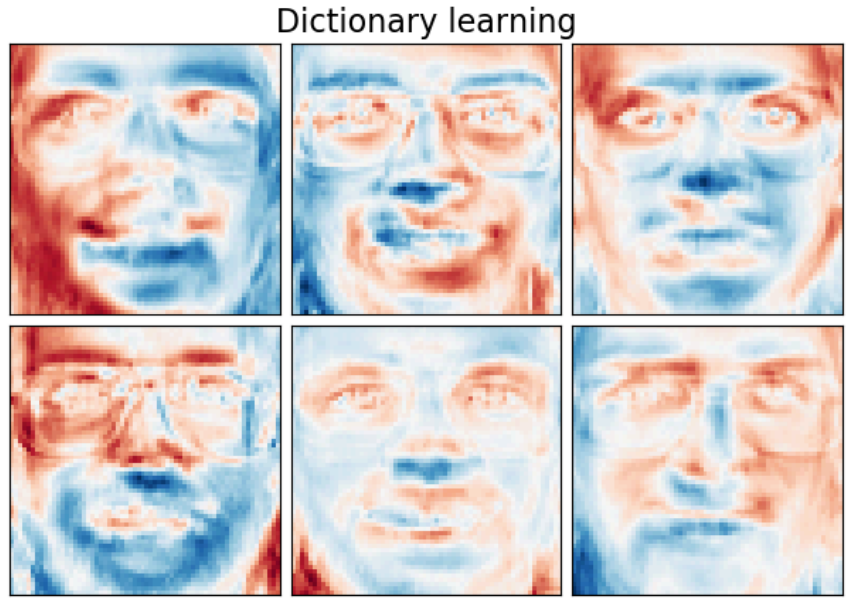
$$(U^*, V^*) = \arg\min_{U,V} \frac{1}{2} ||X-UV||_2^2 + \alpha||U||_1$$
$$\text{subject to } ||V_k||_2 = 1 \text{ for all } 0 \leq k < n_{\text{atoms}}$$

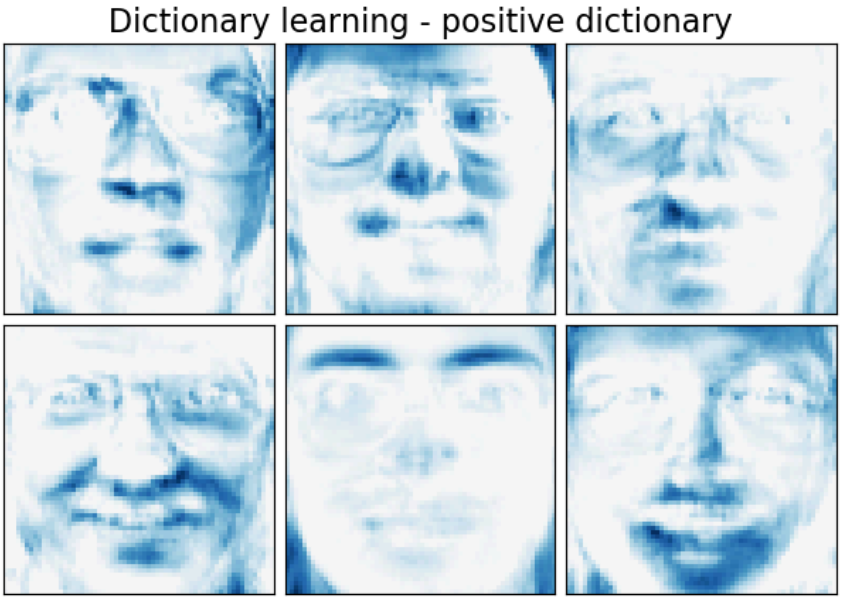


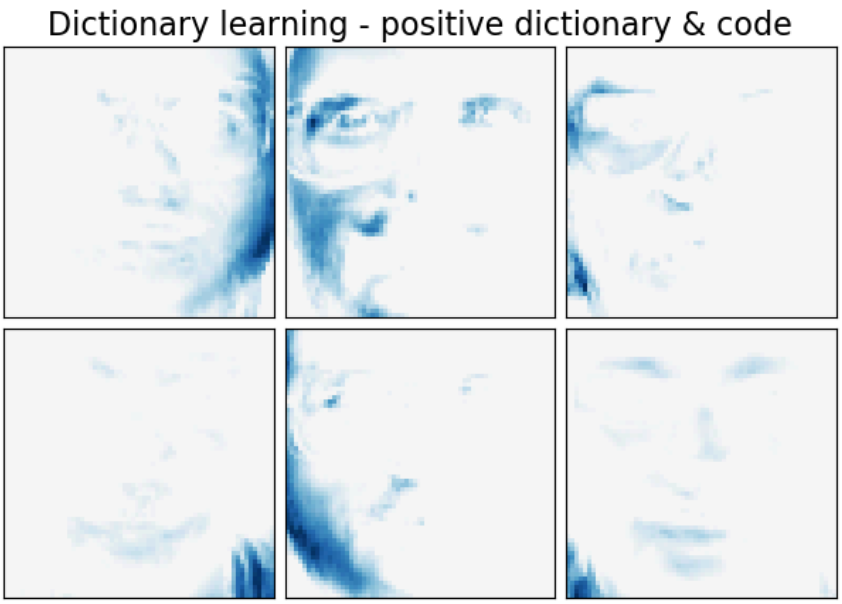


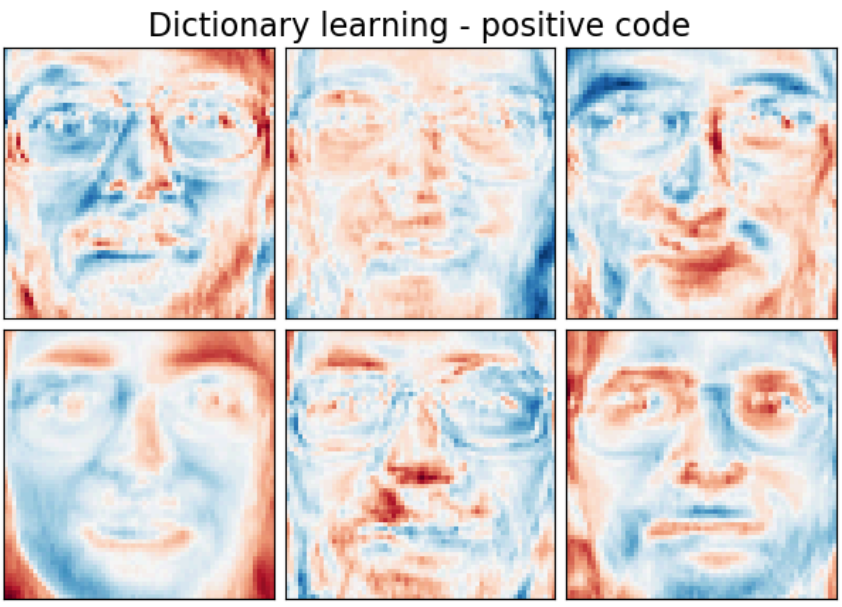
在使用这样的过程来适应字典之后，变换只是一个稀疏编码步骤，与所有字典学习对象共享相同的实现。(见 带有预计算词典的稀疏编码)。

也可以将字典和/或编码约束为正以匹配数据中可能表现的约束。以下是应用不同正约束的人脸。红色表示负值, 蓝色表示正值,白色表示零。









下图展示了字典是如何从部分浣熊脸部图像中提取4x4像素图像补丁中进行词典学习。

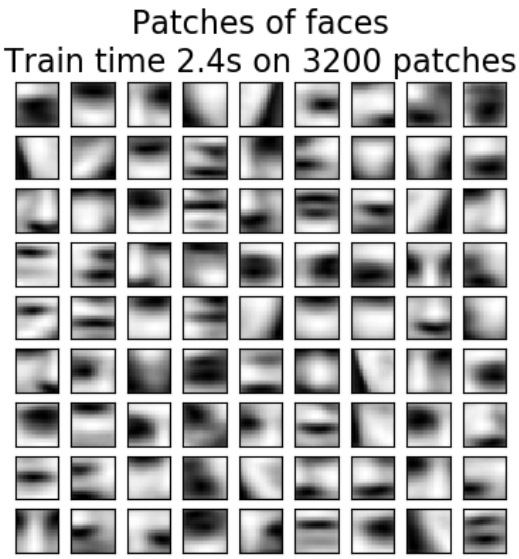
> 示例:
>
> - **Image denoising using dictionary learning**
>
> 参考资料:
>
> - **"Online dictionary learning for sparse coding"** J. Mairal, F. Bach, J. Ponce, G. Sapiro, 2009

2.5.3.3. 小批量字典学习

`MiniBatchDictionaryLearning` 实现了一种更快但精确度下降的词典学习算法，更适合于大型数据集。

默认情况下， `MiniBatchDictionaryLearning` 将数据分成小批量，并以在线方式进行优化，方法是通过在指定次数的迭代中循环使用小批量。但是，它目前没有实现停止条件。

估计器还实现了 `partial_fit` ，它通过在一个小批处理中仅迭代一次来更新字典。 当在线学习的数据从一开始就不容易获得，或者数据超出内存时，可以使用这种迭代方法。



> 词典学习聚类
>
> 请注意，当使用字典学习来提取表示(例如稀疏编码)时，聚类是学习字典的一个很好的中间方法。 例如， `MiniBatchKMeans` 估计器能高效计算并使用 `partial_fit` 方法实现在线学习。
>
> 示例: 在线学习人脸部分的字典 **Online learning of a dictionary of parts of faces**

2.5.4. 因子分析

在无监督学习中，我们只有一个数据集$X = \{x_1, x_2, \ldots, x_n\}$。如何用数学描述这个数据集？ X 的一个非常简单的 连续隐变量模型 。

$$x_i = Wh_i + \mu + \epsilon$$

向量 h_i 被称为'隐性的（latent）'因为它是不可观测的。ϵ 为均值为 0 ， 协方差为Ψ (i.e. $\epsilon \sim \mathcal{N}(0, \Psi)$) 的符合高斯分布的噪声项，

μ 是偏置向量，这种模型被称为“生成的”，因为它描述了 h_i 如何生成 x_i 。如果我们使用所有的 x_i 作为列来形成一个矩阵 X ，并将所有的 h_i 作为矩阵 H 的列，然后我们可以写(用适当的定义 M 和 E):

$$\mathbf{X} = W\mathbf{H} + \mathbf{M} + \mathbf{E}$$

换句话说，我们分解矩阵 X ,如果给出 h_i ,上述方程自动地表示以下概率解释：

$$p(x_i|h_i) = \mathcal{N}(Wh_i + \mu, \Psi)$$

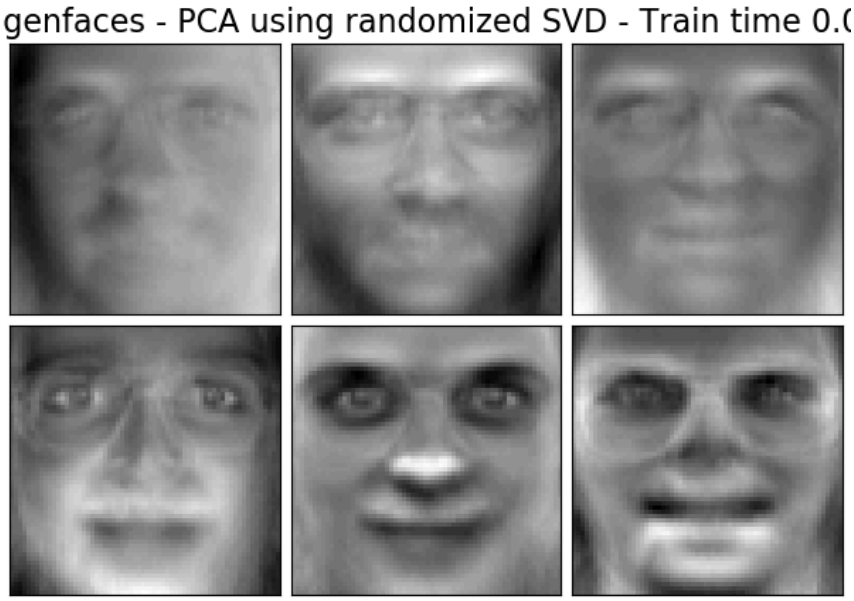
对于一个完整的概率模型，我们还需要隐变量 h 的先验分布。 最直接的假设（基于高斯分布的优良性质）是$h \sim \mathcal{N}(0, \mathbf{I})$ 。这产生一个高斯分布作为 x 边际分布:

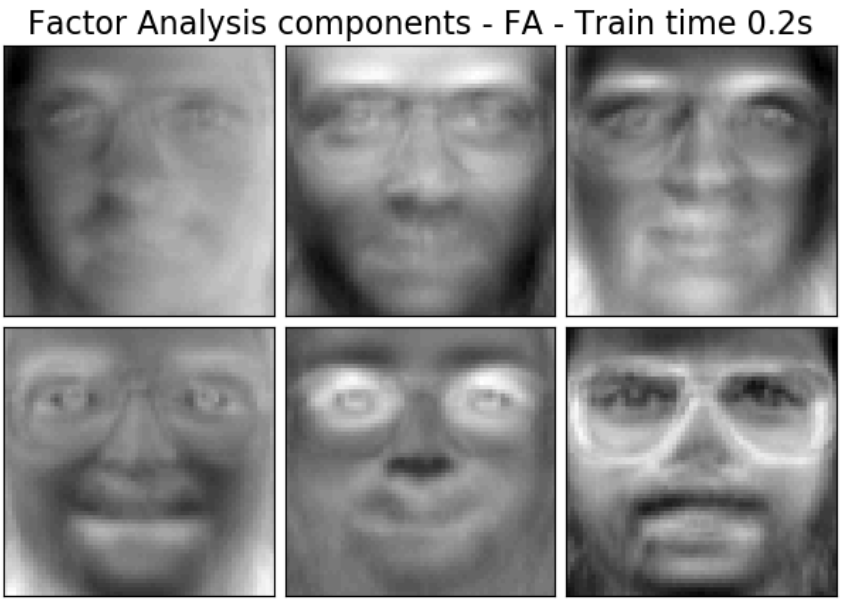
$$p(x) = \mathcal{N}\left(\mu, WW^T + \Psi\right)$$

现在，在没有任何进一步假设的前提下，有一个隐变量 h 是多余的， $-x$完全可以用均值和协方差来建模。 我们需要对这两个参数之一施加一些更具体的构造。 误差协方差的一个简单的附加假设是 Ψ 构造: $\mathbf{\Psi} = \sigma^2\mathbf{I}$: 这个假设导致的概率模型 `PCA` 。 $\mathbf{\Psi} = \operatorname{diag}(\psi_1, \psi_2, \ldots, \psi_n)$: 此模型称为 `FactorAnalysis` ，是经典的统计模型。矩阵W有时被称为“因子加载矩阵”。

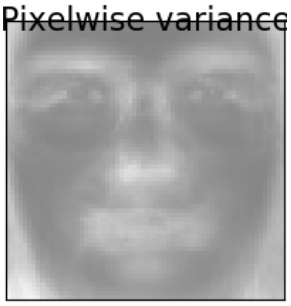
两个模型原则上都可以使用低阶协方差矩阵来估计高斯分布。由于两个模型都是概率模型，因此可以将它们集成到更复杂的模型中，例如，因子分析器的混合。如果隐变量基于非高斯分布的先验，则得到的模型会完全不同（例如， `FastICA` ）。

因子分析可以产生与 `PCA` 类似的分量（例如其加载矩阵的列）。 然而，不能对这些分量做任何一般性声明（例如，它们是否正交）：

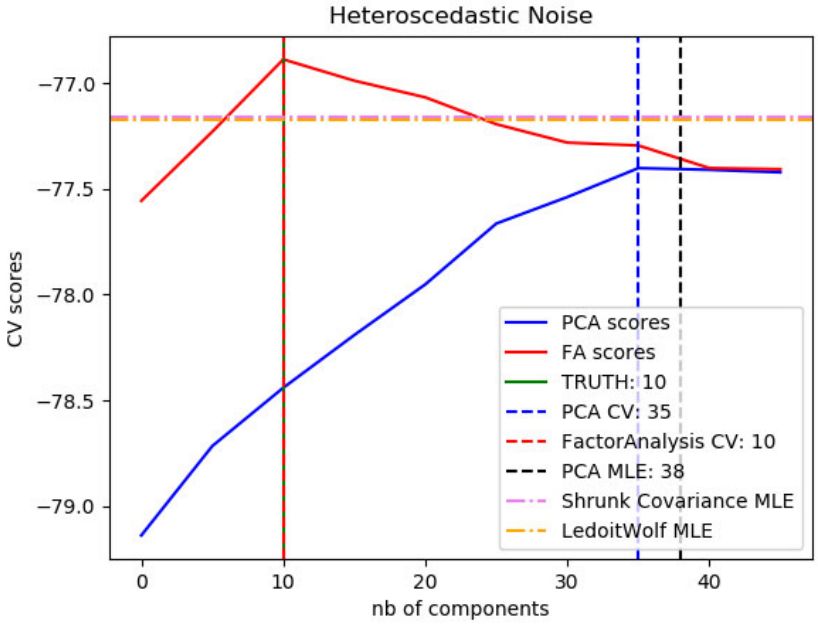




PCA与之相比，因子分析的主要优势在于它可以独立地模拟输入空间各个方向上的方差（异方差噪声）：



与存在异方差噪声的概率PCA相比，这可以提供更好的模型选择：

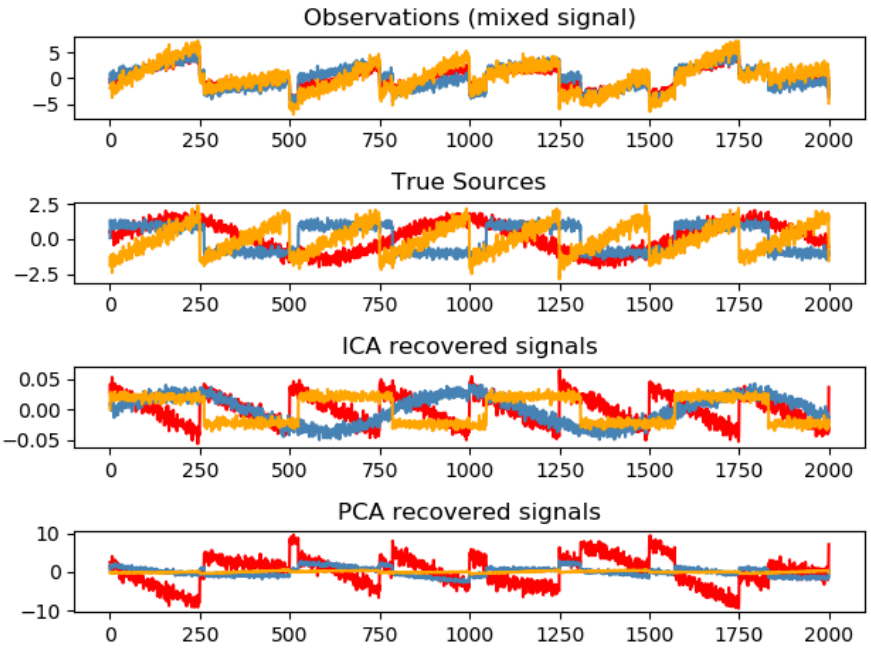


> 示例：
>
> - Model selection with Probabilistic PCA and Factor Analysis (FA)

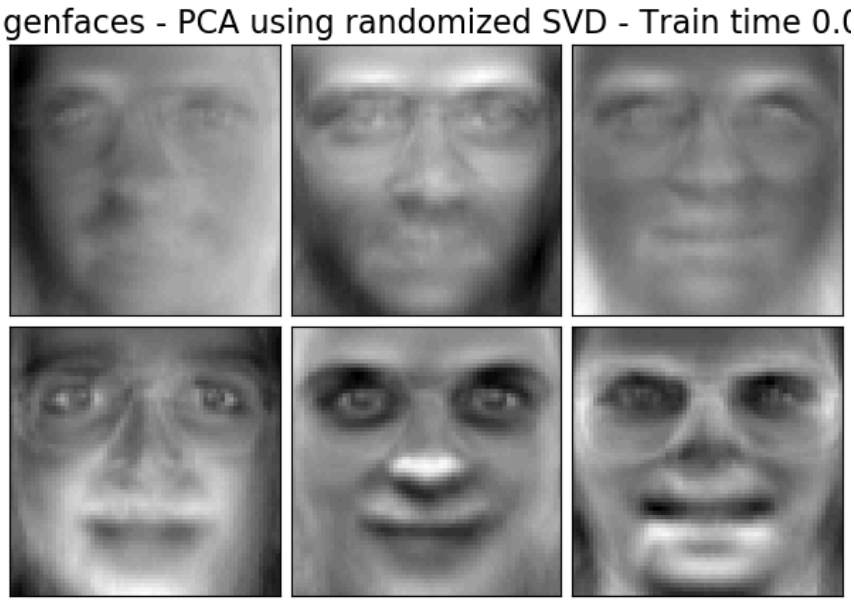
2.5.5. 独立成分分析（ICA）

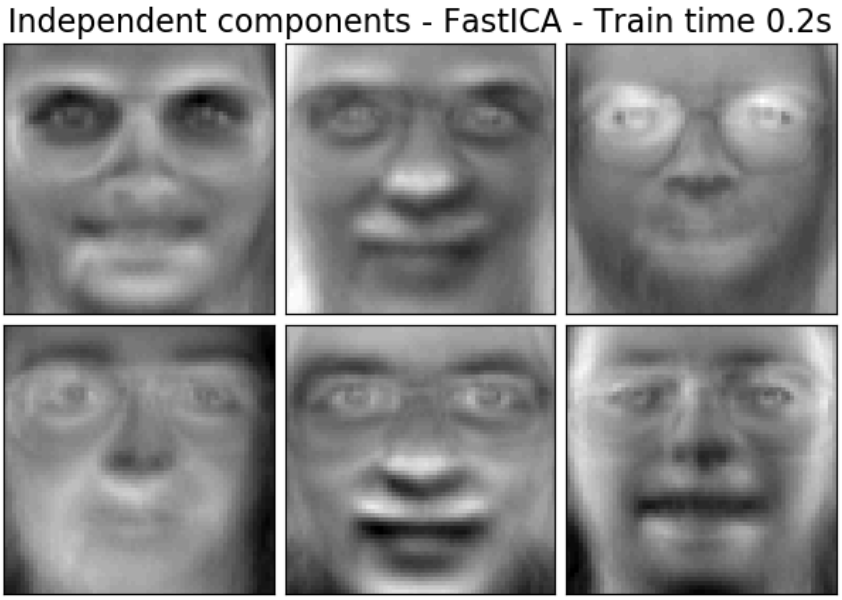
独立成分分析将一个多元信号分解成独立性最强的可加子成分。 它利用 `Fast ICA` 算法在 scikit-learn 中实现。通常，ICA不用于降低维度，而是用于分离叠加信号。由于ICA模型不包含噪声项，为了使模型正确，必须进行白化处理。这可以在内部使用whiten参数完成，也可以手动使用某种PCA变体。

ICA 通常用于分离混合信号（一个称为盲源分离的问题），如下例所示:



ICA 也可以用于寻找具有稀疏性的分量的非线性分解:





示例：

- **Blind source separation using FastICA**
- **FastICA on 2D point clouds**
- **Faces dataset decompositions**

2.5.6. 非负矩阵分解(NMF 或 NNMF)

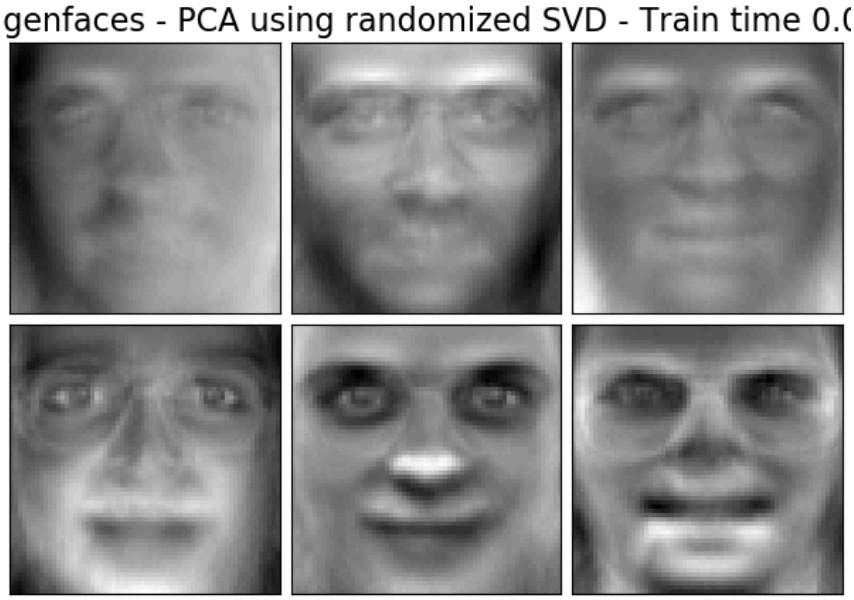
2.5.6.1. NMF 与 Frobenius 范数

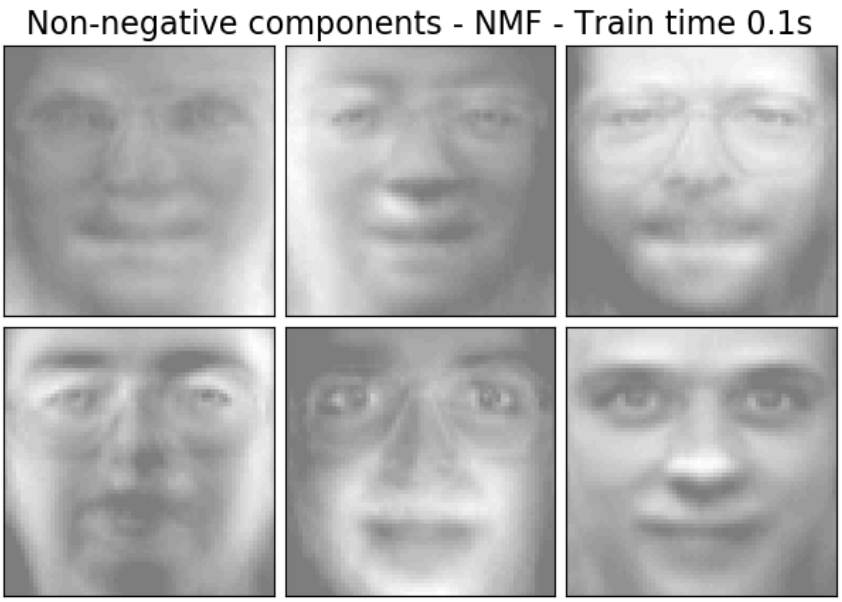
`NMF` [1]是另一种降维方法,它假设数据和分量是非负的。 在数据矩阵不包含负值的情况下，可以插入 `NMF` 而不是 `PCA` 或其变体。 通过优化样本 X 与矩阵乘积 WH 之间的距离 d，可以将样本 X 分解为两个非负矩阵 W 和 H 。使用最广泛的距离函数是 Frobenius 平方范数，它是欧氏范数在矩阵上的推广:

$$d_{\mathrm{Fro}}(X, Y) = \frac{1}{2} \|X - Y\|_{\mathrm{Fro}}^2 = \frac{1}{2} \sum_{i,j} (X_{ij} - Y_{ij})^2$$

与 `PCA` 不同的是，向量的表示是通过相加的方式得到的，通过叠加分量而不是减去分量。这种加性模型对于表示图像和文本非常有效。

[Hoyer, 2004] [2] 研究表明，当处于一定约束时， `NMF` 可以产生基于某一部分数据集的表示，从而产生可解释的模型。以下示例展示了与 PCA 特征人脸相比， `NMF` 从 Olivetti 面部数据集中的图像中找到的的16个稀疏分量。





`init` 属性决定应用的初始化方法，这对方法的性能有很大的影响。`NMF` 实现了非负的双奇异值分解方法。NNDSVD [4]基于两个 SVD 过程，一个对数据矩阵进行近似,另一个是对得到的SVD因子的正部分利用单位秩矩阵的代数特性进行近似。基本的 NNDSVD 算法更适合稀疏分解。其变体 NNDSVDa（在这种情况下，所有的0都被设置为数据所有元素的平均值）和 NNDSVDar（在这种情况下，所有的0都被设置为小于数据除以100的平均值的随机扰动）推荐用于稠密情况。

注意，乘法更新(' mu ')求解器不能更新初始化中出现的零，因此与引入大量零的基本NNDSVD算法联合使用会导致较差的结果;在这种情况下，建议使用NNDSVDa或NNDSVDar。

通过设置 `init="random"` ，使用正确缩放的随机非负矩阵也可以初始化 `NMF` 。 也可以将整数种子或 `RandomState` 传递给 `random_state` 以控制重现性。

在 `NMF` 中，可以在损失函数中加入L1和L2先验使模型正则化。 L2 先验使用 Frobenius 范数，而L1 先验使用 L1 范数。在 `ElasticNet` 中，我们通过 `l1_ratio` (ρ) 参数和正则化强度参数 `alpha` (α)来控制 L1 和 L2 的组合。那么先验条件是:

$$\alpha\rho\|W\|_1 + \alpha\rho\|H\|_1 + \frac{\alpha(1-\rho)}{2}\|W\|^2_{\text{Fro}} + \frac{\alpha(1-\rho)}{2}\|H\|^2_{\text{Fro}}$$

正则化目标函数为:

$$d_{\text{Fro}}(X, WH) + \alpha\rho\|W\|_1 + \alpha\rho\|H\|_1 + \frac{\alpha(1-\rho)}{2}\|W\|^2_{\text{Fro}} + \frac{\alpha(1-\rho)}{2}\|H\|^2_{\text{Fro}}$$

`NMF` 正则化 W 和 H 。公共函数 `non_negative_factorization` 允许通过 `regularization` 属性进行更精细的控制，将W，H分别 或两者都进行正则化。

2.5.6.2. 具有 beta-divergence 的 NMF

如前所述，使用最广泛的距离函数是Frobenius 平方范数，它是欧氏范数在矩阵上的一个明显的扩展应用:

$$d_{\text{Fro}}(X, Y) = \frac{1}{2}\|X - Y\|^2_{Fro} = \frac{1}{2}\sum_{i,j}(X_{ij} - Y_{ij})^2$$

其他距离函数可用于 NMF，例如（广义） Kullback-Leibler(KL) 散度，也称为 I-divergence:

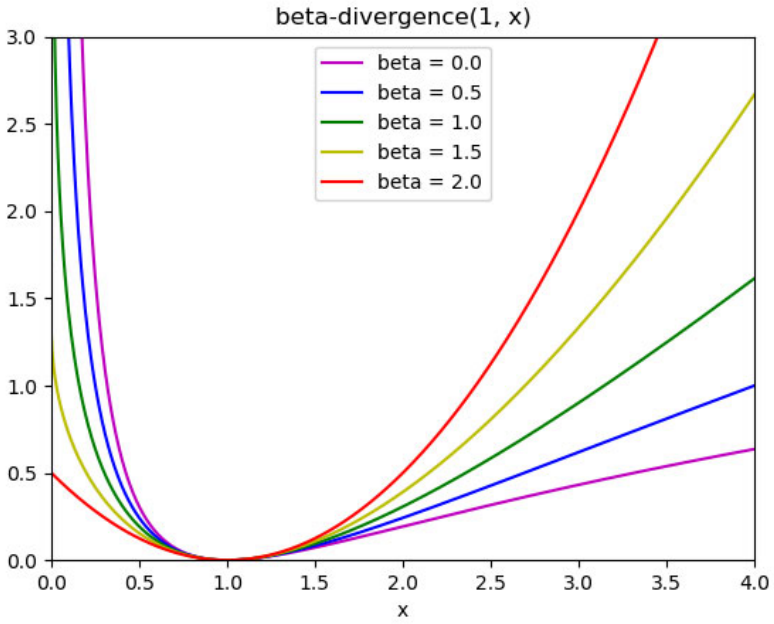
$$d_{KL}(X, Y) = \sum_{i,j}\left(X_{ij}\log\left(\frac{X_{ij}}{Y_{ij}}\right) - X_{ij} + Y_{ij}\right)$$

或者， Itakura-Saito(IS) divergence:

$$d_{IS}(X, Y) = \sum_{i,j}\left(\frac{X_{ij}}{Y_{ij}} - \log\left(\frac{X_{ij}}{Y_{ij}}\right) - 1\right)$$

这三个距离函数是 beta-divergence 函数的特殊情况，其参数分别为 $\beta = 2, 1, 0$ [6] 。 beta-divergence 定义如下:

$$d_\beta(X, Y) = \sum_{i,j}\frac{1}{\beta(\beta-1)}\left(X_{ij}^\beta + (\beta-1)Y_{ij}^\beta - \beta X_{ij}Y_{ij}^{\beta-1}\right)$$



注意在 $\beta \in (0; 1)$ 上定义无效，但它可以连续扩展到 d_{KL} 和 d_{IS}

`NMF` 实现了两个求解器，分别使用了坐标下降 (‘cd’) [5]和乘法更新(‘mu’) [6]。 ‘mu’ 求解器可以优化每个 beta 散度，包括 Frobenius 范数 ($\beta = 2$) ，（广义的） Kullback-Leibler 散度 （$\beta = 1$）和Itakura-Saito 散度（$\beta = 0$）） 。注意对于 $\beta \in (1; 2)$，’mu’ 求解器明显快于 β 取其他值。还要注意， β如果是负数（或0，即 ‘itakura-saito’ ） ，输入矩阵不能包含零值。

‘cd’ 求解器只能优化 Frobenius 范数。由于 NMF 的非凸性，即使优化相同的距离函数时，不同的解也可能会收敛到不同的最小值。

NMF最适用于 `fit_transform` 方法，该方法返回矩阵W。矩阵 H 被存储到拟合模型中的 `components_` 属性; 方法 `transform` 将基于这些存储的分量分解一个新的矩阵 X_new:

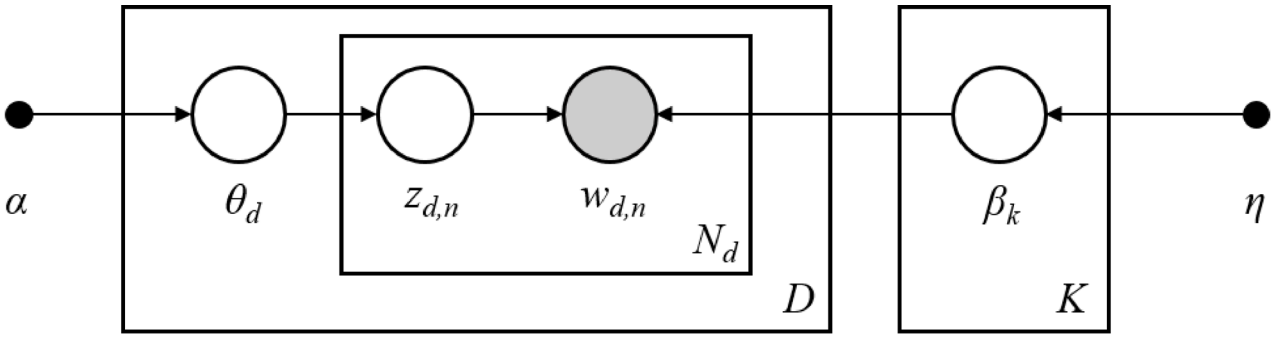
```
>>> import numpy as np
>>> X = np.array([[1, 1], [2, 1], [3, 1.2], [4, 1], [5, 0.8], [6, 1]])
>>> from sklearn.decomposition import NMF
>>> model = NMF(n_components=2, init='random', random_state=0)
>>> W = model.fit_transform(X)
>>> H = model.components_
>>> X_new = np.array([[1, 0], [1, 6.1], [1, 0], [1, 4], [3.2, 1], [0, 4]])
>>> W_new = model.transform(X_new)
```

> 示例:
>
> - 人脸数据集分解
> - 基于非负矩阵分解和潜在Dirichlet分配的主题抽取
> - Beta-divergence损失函数
>
> 参考资料;
>
> - 1“Learning the parts of objects by non-negative matrix factorization” D. Lee, S. Seung, 1999
> - 2“Non-negative Matrix Factorization with Sparseness Constraints” P. Hoyer, 2004
> - 4“SVD based initialization: A head start for nonnegative matrix factorization” C. Boutsidis, E. Gallopoulos, 2008
> - 5“Fast local algorithms for large scale nonnegative matrix and tensor factorizations.” A. Cichocki, A. Phan, 2009
> - 6(1,2)“Algorithms for nonnegative matrix factorization with the beta-divergence” C. Fevotte, J. Idier, 2011

2.5.7. 隐 Dirichlet 分配（LDA）

隐 Dirichlet 分配是离散数据集的集合（如文本语料库）的生成概率模型。 它也是一个主题模型，用于从文档集合中发现抽象主题。

LDA 的图形模型是一个三级生成模型:



注意上面的图形模型中的符号，可以在Hoffman et al.(2013)中找到:

- 语料库是D文档的集合。
- 文档是N单词的序列。
- 语料库中有K主题。
- 这些方框代表重复采样。

在图形模型中，每个节点都是随机变量，在生成过程中都有作用。有阴影的节点表示观察到的变量，无阴影的节点表示隐藏(潜在的)变量。在这种情况下，语料库中的单词是我们观察到的唯一数据。潜在变量决定了语料库中主题的随机混合和文档中词汇的分布。LDA的目标是利用观察到的词来推断隐藏的主题结构。

在对文本语料库进行建模时，该模型假设以下语料库的生成过程 D 文件和 K 主题， K 对应与API中的 `n_components` ：

1. 对于每个主题 $k \in K$， 绘制 $\beta_k \sim$Dirichlet (η)。这提供了单词的分布，即单词出现在主题中的概率k。 (η)对应于 `topic_word_prior` 。
2. 对于每个文档 $d \in D$，绘制主题比例 $\theta_d \sim$ Dirichlet (α)。 α 对应于 `doc_topic_prior` 。
3. 对在文件中 d 中的每个单词：
4. 绘制主题分配： $z_{di} \sim$ Multinomial (θ_d)
5. 观察到的单词： $w_{ij} \sim$ Multinomial $(\beta_{z_{ti}})$

对于参数估计，后验分布为:

$$p(z, \theta, \beta | w, \alpha, \eta) = \frac{p(z, \theta, \beta | \alpha, \eta)}{p(w | \alpha, \eta)}$$

由于后验是很难处理，因此变分贝叶斯方法使用一个更简单的分布 $q(z, \theta, \beta | \lambda, \phi, \gamma)$来近似它，对这些变分参数$\lambda, \phi, \gamma$进行优化，使(ELBO)最大化:

$$\log P(w | \alpha, \eta) \geq L(w, \phi, \gamma, \lambda) \triangleq E_q[\log p(w, z, \theta, \beta | \alpha, \eta)] - E_q[\log q(z, \theta, \beta)]$$

最大化ELBO等效于最小化 $q(z, \theta, \beta)$ 和 后验 $p(z, \theta, \beta | w, \alpha, \eta)$ 之间的 Kullback-Leibler(KL) 散度。

`LatentDirichletAllocation` 实现在线变分贝叶斯算法，并支持在线和批量更新方法。批处理方法在每次完整遍历数据后更新变量，而在线方法则从小批量数据点更新变量。

> 注:虽然在线方法保证收敛到局部最优点，但最优点的质量和收敛速度可能取决于小批量的大小和与学习速率设置相关的属性。

当 `LatentDirichletAllocation` 应用于“文档项”矩阵时，该矩阵将被分解为一个“主题项”矩阵和一个“文档项”矩阵。在模型中，“主题-术语”矩阵 `components_` 与模型一样存储，但“文档-主题”矩阵可以通过 `transform` 方法进行计算。

`LatentDirichletAllocation` 还实现 `partial_fit` 方法。当可以按顺序获取数据时，就使用这种方法。

示例：

- 基于非负矩阵分解和潜在**Dirichlet**分配的主题抽取

参考文献：

- **“Latent Dirichlet Allocation”** D. Blei, A. Ng, M. Jordan, 2003
- **“Online Learning for Latent Dirichlet Allocation”** M. Hoffman, D. Blei, F. Bach, 2010
- **“Stochastic Variational Inference”** M. Hoffman, D. Blei, C. Wang, J. Paisley, 2013

参见 **Neighborhood Components Analysis** 的降维方式.