降维

1. 降维的一些通用方法：
   - `get_params([deep])`：返回模型的参数。
     - `deep`： 如果为`True`，则可以返回模型参数的子对象。
   - `set_params(**params)`：设置模型的参数。
     - `params`：待设置的关键字参数。
   - `fit(X[, y])`：训练模型。
     - `X`：样本集合。通常是一个`numpy array`，每行代表一个样本，每列代表一个特征。
     - `y`：样本的标签集合。它与`X`的每一行相对应。
   - `transform(X)`：执行降维，返回降维后的样本集。
     - `X`：样本集合。通常是一个`numpy array`，每行代表一个样本，每列代表一个特征。
   - `inverse_transform(X)`：执行降维的逆运算，返回降维之前的样本集合。
     - `X`：降维之后的样本集合。通常是一个`numpy array`，每行代表一个样本，每列代表一个特征。
   - `fit_transform(X[, y])`：训练模型并执行降维，返回降维后的样本集。
     - `X`：样本集合。通常是一个`numpy array`，每行代表一个样本，每列代表一个特征。
     - `y`：样本的标签集合。它与`X`的每一行相对应。
2. 降维的一些通用参数：
   - `copy`： 一个布尔值，指定是否拷贝原始数据。

     如果为`False`则执行原地修改。此时节省空间，但修改了原始数据。
   - `n_jobs`：一个正数，指定任务并形时指定的`CPU`数量。

     如果为`-1`则使用所有可用的`CPU`。
   - `random_state`：一个整数或者一个`RandomState`实例，或者`None`。
     - 如果为整数，则它指定了随机数生成器的种子。
     - 如果为`RandomState`实例，则指定了随机数生成器。
     - 如果为`None`，则使用默认的随机数生成器。
   - `n_components`：一个整数，指定降维后的维数。

一、PCA

1.1 PCA

1. `scikit-learn`中的`PCA`类实现了`PCA`模型，其原型为：

```
class sklearn.decomposition.PCA(n_components=None, copy=True, whiten=False)
```

   - `n_components`：一个整数，指定降维后的维数。
     - 如果为`None`，则选择它的值为`min(n_samples,n_features)`。

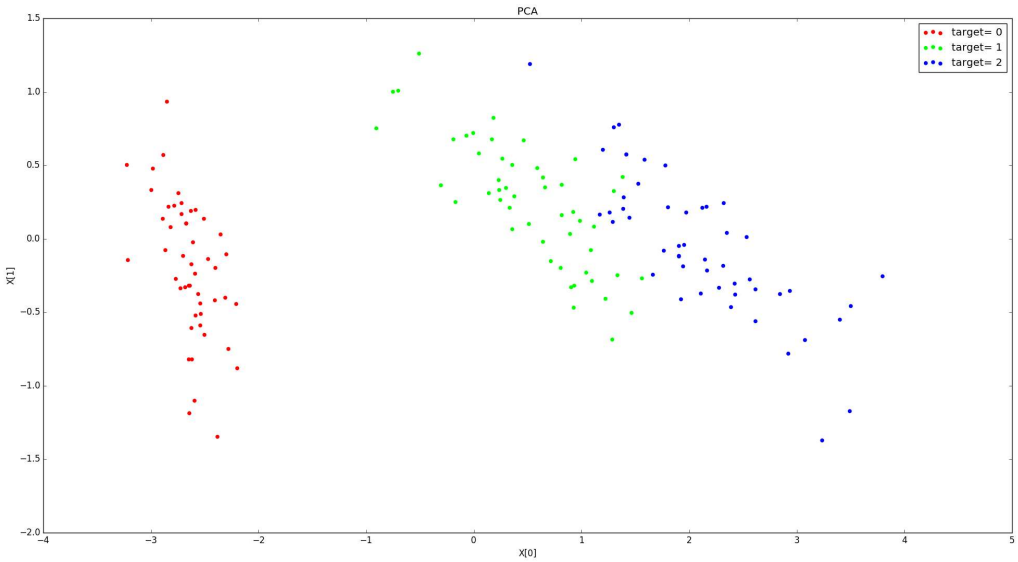
        - 如果为字符串`'mle'`，则使用`Minka's MLE`算法来猜测降维后的维数。
        - 如果为大于0，小于1的浮点数，则指定的是降维后的维数占原始维数的百分比。
    - `copy`：一个布尔值，指定是否拷贝原始数据。
    - `whiten`：一个布尔值，指定是否执行白化操作。

      如果为`True`，则会将特征向量除以`n_samples`倍的特征值，从而保证非相关的输出的方差为1。

      白化操作可能会丢弃部分信息，但是它有时候在接下来的学习器学习阶段能获得更佳的性能。

2. 属性：
    - `components_`：一个数组，给出主成分。
    - `explained_variance_`：一个数组，元素是每个成分对应的`explained variance`。
    - `explained_variance_ratio_`：一个数组，元素是每个主成分的`explained variance`的比例。
    - `mean_`：一个数组，元素是每个特征的统计均值。
    - `n_components_`：一个整数，指示主成分有多少个元素。
3. 方法：
    - `fit(X[, y])`：训练模型，获取降维需要的参数。
    - `transform(X)`：执行降维，返回降维后的样本集。
    - `fit_transform(X[, y])`：训练模型并执行降维，返回降维后的样本集。
    - `inverse_transform(X)`：执行降维的逆运算，返回降维之前的样本集合。
4. 注意：`decomposition.PCA`基于`scipy.linalg`来实现`SVD`分解，因此有两个限制：
    - 不能应用于稀疏矩阵。
    - 无法适用于超大规模数据，因为它要求所有的数据一次加载进内存。
5. 示例：鸢尾花数据集中，`n_components_=4`；`explained_variance_ratio_=[ 0.92461621 0.05301557 0.01718514 0.00518309]`。

   降到2维的结果为：



1.2 IncrementalPCA

1. `scikit-learn`中的`IncrementalPCA`类也实现了`PCA`模型。它适用于超大规模数据，可以将数据分批加载进内存。

其原型为：

```
class sklearn.decomposition.IncrementalPCA(n_components=None, whiten=False,
copy=True,batch_size=None)
```

- `batch_size`：一个整数或者`None`，指定每个批次训练时，使用的样本数量。
  - 只有当调用`fit()/partial_fit()`方法时，才会用到该参数。
  - 如果为`None`，则由算法自动推断。
- 其它参数参考`decomposition.PCA`。

2. 属性：
   - `components_`：一个数组，给出主成分。
   - `explained_variance_`：一个数组，元素是每个成分对应的`explained variance`。
   - `explained_variance_ratio_`：一个数组，元素是每个主成分的`explained variance`的比例。
   - `mean_`：一个数组，元素是每个特征的统计平均值。

     每调用一次`partial_fit()`方法就会更新一次该属性。
   - `var_`：一个数组，元素是每个特征的经验方差。

     每调用一次`partial_fit()`方法就会更新一次该属性。
   - `n_components_`：一个整数，指示主成分有多少个元素。
   - `n_samples_seen_`：一个整数，指示目前已经处理了多少个样本。
     - 每调用一次`partial_fit()`方法就会更新一次该属性。
     - 每调用一次`fit()`方法就会清零该属性。
3. 方法：参考`decomposition.PCA`。

1.3 KernelPCA

1. `KernelPCA`是`scikit-learn`实现的核化`PCA`模型，其原型为：

```
class sklearn.decomposition.KernelPCA(n_components=None, kernel='linear',
gamma=None, degree=3, coef0=1, kernel_params=None, alpha=1.0,
fit_inverse_transform=False,eigen_solver='auto', tol=0, max_iter=None,
remove_zero_eig=False)
```

- `n_components`：一个整数，指定降维后的维数。
- `kernel`：一个字符串或者可调用对象，指定核函数。
  - `'linear'`：线性核：$K(\vec{\mathbf{x}},\vec{\mathbf{z}})=\vec{\mathbf{x}}\cdot\vec{\mathbf{z}}$。
  - `'poly'`：多项式核：$K(\vec{\mathbf{x}},\vec{\mathbf{z}})=(\gamma(\vec{\mathbf{x}}\cdot\vec{\mathbf{z}}+1)+r)^p$，其中$p$由`degree`参数决定，$\gamma$由`gamma`参数决定，$r$由`coef0`参数决定。
  - `'rbf'`（默认值）：高斯核函数：$K(\vec{\mathbf{x}},\vec{\mathbf{z}})=\exp(-\gamma||\vec{\mathbf{x}}-\vec{\mathbf{z}}||^2)$,其中$\gamma$由`gamma`参数决定。
  - `'sigmoid'`：`sigmod`核函数：$K(\vec{\mathbf{x}},\vec{\mathbf{z}})=\tanh(\gamma(\vec{\mathbf{x}}\cdot\vec{\mathbf{z}})+r)$。其中$\gamma$由`gamma`参数决定，`r`由`coef0`参数指定。
  - `'precomputed'`：表示提供了`kernel matrix`。

        - 一个可调用对象，该对象用于计算 `kernel matrix` 。
    - `degree`：一个整数，当核函数是多项式核函数时，指定多项式的系数。

      对于其他核函数，该参数无效。
    - `gamma`：一个浮点数，当核函数是 `'rbf'` ， `'poly'` ， `'sigmoid'` 时，指定核函数的系数。

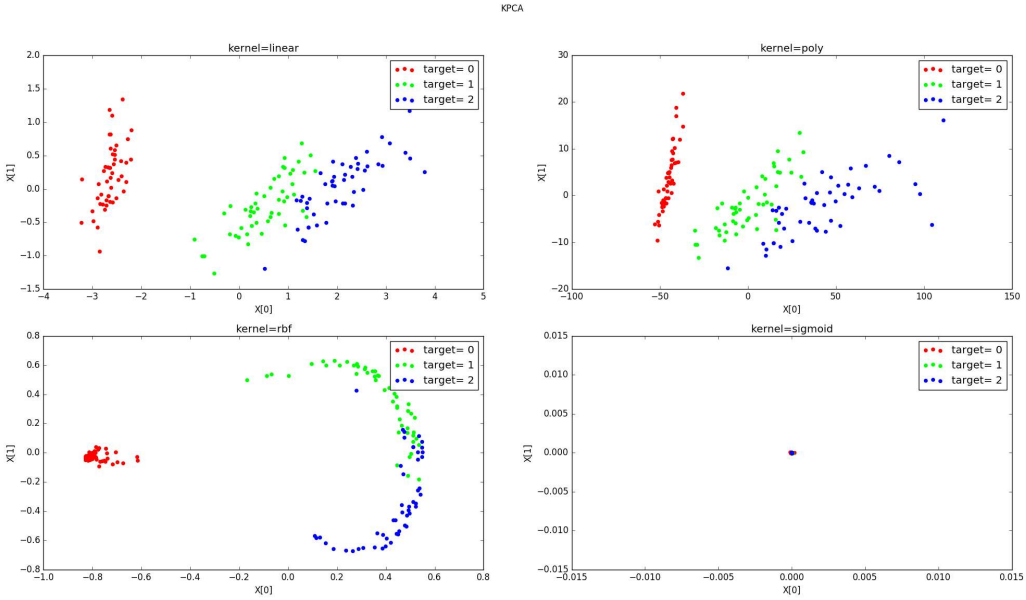
      如果 `'auto'` ，则表示系数为 `1/n_features`
    - `coef0`：浮点数，用于指定核函数中的自由项。

      只有当核函数是 `'poly'` 和 `'sigmoid'` 是有效。
    - `kernel_params`：当核函数是个可调用对象时才使用它，用于为该可调用对象传递参数。

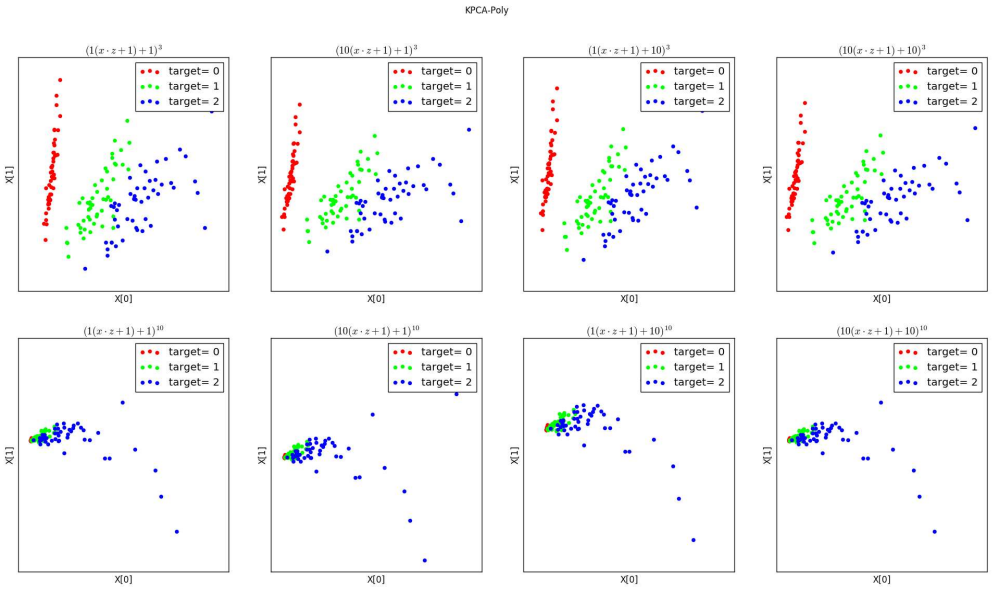
      如果核函数是上述指定的字符串，则该参数不起作用。
    - `alpha`：一个整数，岭回归的超参数，用于计算逆转换矩阵（当 `fit_inverse_transform=True` 时）。
    - `fit_inverse_transform`：一个布尔值，指定是否需要计算逆转换矩阵。当为 `True` 时，需要计算逆转换矩阵。
    - `eigen_solver`：一个字符串，指定求解特征值的算法：
        - `'auto'`：自动选择。
        - `'dense'`： `dense` 特征值求解器。
        - `'arpack'`： `arpack` 特征值求解器，用于当特征数量远小于样本数量的情形。
    - `tol`：一个浮点数，指定 `arpack` 特征值求解器的收敛阈值（如果为0，则自动选择阈值）。
    - `max_iter`：一个整数，指定 `arpack` 特征值求解器的最大迭代次数（如果为 `None` ，则自动选择）。
    - `remove_zero_eig`：一个布尔值。如果为 `True` ，则移除所有为零的特征值。如果 `n_components=None` ，则也会移除所有为零的特征值。

2. 属性：
    - `lambdas_`：核化矩阵的特征值。
    - `alphas_`：核化矩阵的特征向量。
    - `dual_coef_`：逆转换矩阵。
3. 方法：参考 `decomposition.PCA` 。
4. 示例：

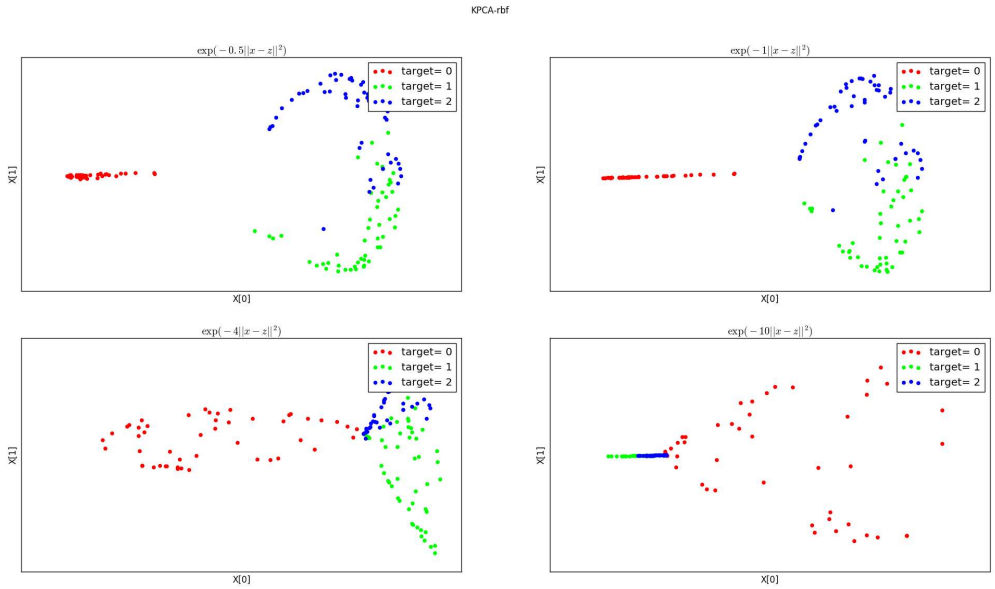
   不同的核函数降维后的数据分布：



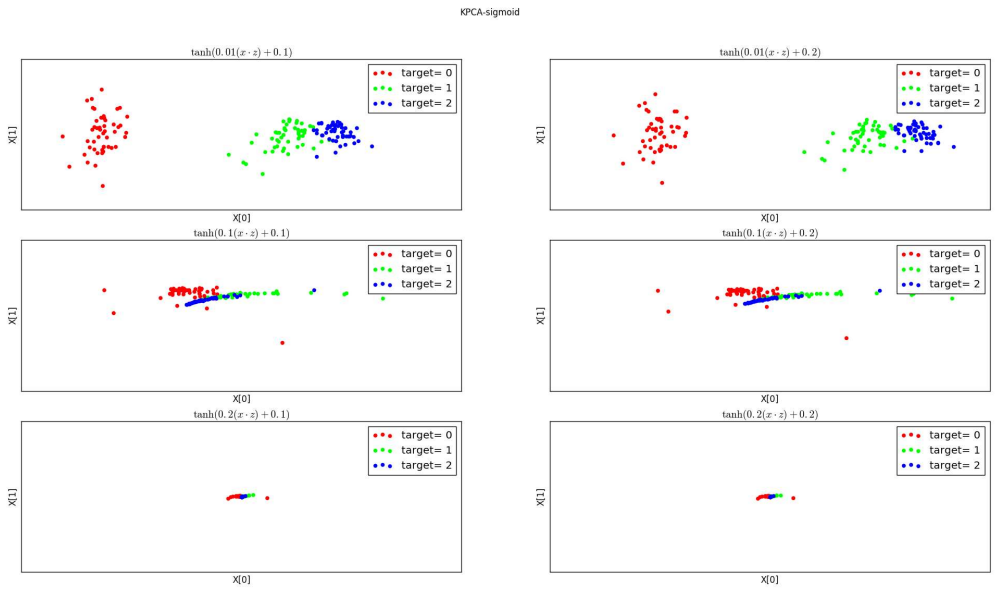
不同参数的多项式核函数降维后的数据分布：



不同参数的高斯核函数降维后的数据分布：



不同参数的`sigmoid`核函数降维后的数据分布：



二、MDS

1. `MDS`是`scikit-learn`实现的多维缩放模型，其原型为：

```
class sklearn.manifold.MDS(n_components=2, metric=True, n_init=4, max_iter=300,
verbose=0, eps=0.001, n_jobs=1, random_state=None, dissimilarity='euclidean')
```

   - `metric`：一个布尔值，指定度量类型。

     如果为`True`，则使用距离度量；否则使用非距离度量`SMACOF`。
   - `n_components`：一个整数，指定降维后的维数。
   - `n_init`：一个整数，指定初始化的次数。

在使用`SMACOF`算法时，会选择`n_init`次不同的初始值，然后选择这些结果中最好的那个作为最终结果。

- `max_iter`：一个整数，指定在使用`SMACOF`算法时得到一轮结果需要的最大迭代次数。
- `eps`：一个浮点数，用于指定收敛阈值。
- `n_jobs`：一个整数，指定并行性。
- `random_state`：一个整数或者一个`RandomState`实例，或者`None`，指定随机数种子。
- `dissimilarity`：一个字符串值，用于定义如何计算不相似度。可以为：
  - `'euclidean'`：使用欧氏距离。
  - `'precomputed'`：由使用者提供距离矩阵。

2. 属性：

- `embedding_`：给出了原始数据集在低维空间中的嵌入矩阵。
- `stress_`：一个浮点数，给出了不一致的距离的总和。

3. 方法：

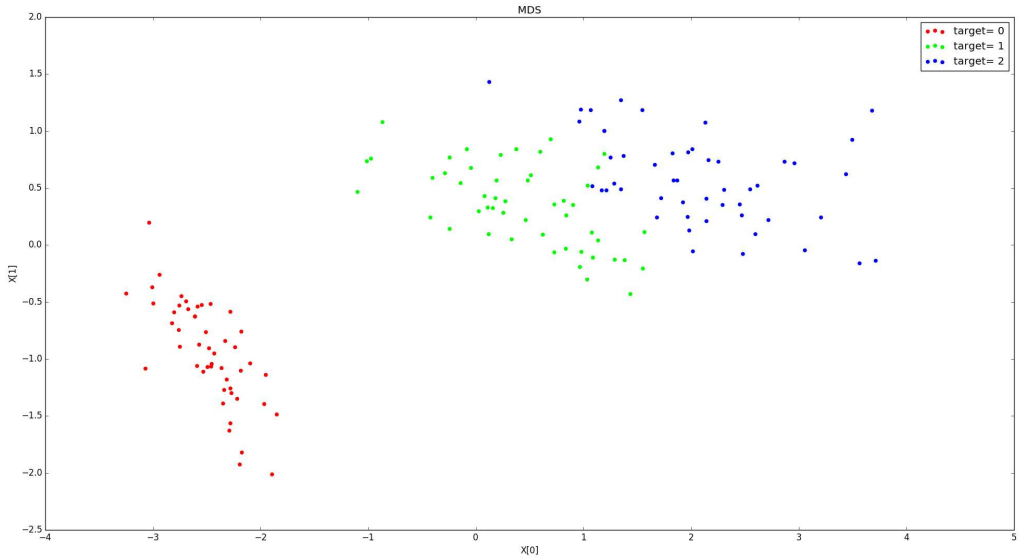
- `fit(X[, y, init])`：训练模型。
- `fit_transform(X[, y, init])`：训练模型并执行降维，返回降维后的样本集。

4. 示例：鸢尾花数据集分别降低到4、3、2、1 维时，距离的误差之和分别为：

```
stress(n_components=4) : 12.0577408711
stress(n_components=3) : 17.8262808779
stress(n_components=2) : 234.395807108
stress(n_components=1) : 23691.9560412
```

该指标并不能用于判定降维的效果的好坏，它只是一个中性指标。

降到2维的样本分布图：



三、Isomap

1. `Isomap`类是`scikit-learn`提供的`Isomap`模型，其原型为：

```
class sklearn.manifold.Isomap(n_neighbors=5, n_components=2, eigen_solver='auto',
tol=0, max_iter=None, path_method='auto', neighbors_algorithm='auto')
```

   - `n_neighbors`：一个整数，指定近邻参数 k 。
   - `n_components`：一个整数，指定降维后的维数。
   - `eigen_solver`：一个字符串，指定求解特征值的算法，可以为：
     - `'auto'`：由算法自动选取。
     - `'arpack'`：使用`Arnoldi`分解算法。
     - `'dense'`：使用一个直接求解特征值的算法（如`LAPACK`）。
   - `tol`：一个浮点数，指定求解特征值算法的收敛阈值（当`eigen_solver='dense'`时，该参数无用）。
   - `max_iter`：一个浮点数，指定求解特征值算法的最大迭代次数（当`eigen_solver='dense'`时，该参数无用）。
   - `path_method`：一个字符串，指定寻找最短路径算法。可以为：
     - `'auto'`：由算法自动选取。
     - `'FW'`：使用`Floyd_Warshall`算法。
     - `'D'`：使用`Dijkstra`算法。
   - `neighbors_algorithm`：一字符串，指定计算最近邻的算法。可以为：
     - `'ball_tree'`：使用`BallTree`算法。
     - `'kd_tree'`：使用`KDTree`算法。
     - `'brute'`：使用暴力搜索法。
     - `'auto'`：自动决定最合适的算法。

2. 属性：
   - `embedding_`：给出了原始数据集在低维空间中的嵌入矩阵。
   - `training_data_`：存储了原始训练数据。
   - `dist_matrix_`：存储了原始训练数据的距离矩阵。

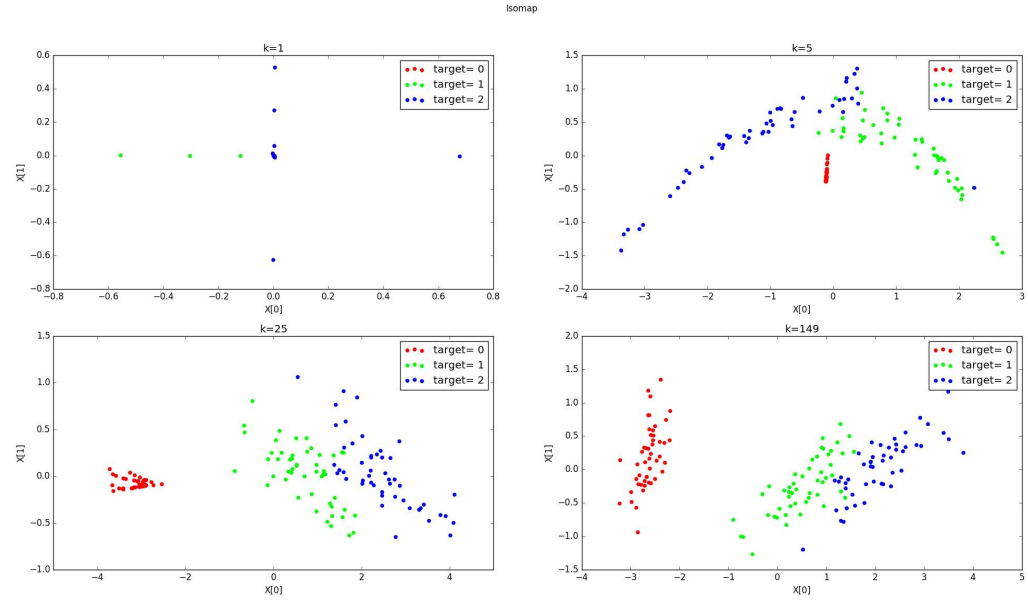
3. 方法：
   - `fit(X[, y])`：训练模型。
   - `transform(X)`：执行降维，返回降维后的样本集。
   - `fit_transform(X[, y])`：训练模型并执行降维，返回降维后的样本集。
   - `reconstruction_error()`：计算重构误差。

4. 示例：鸢尾花数据集分别降低到4、3、2、1 维时，重构误差分别为：

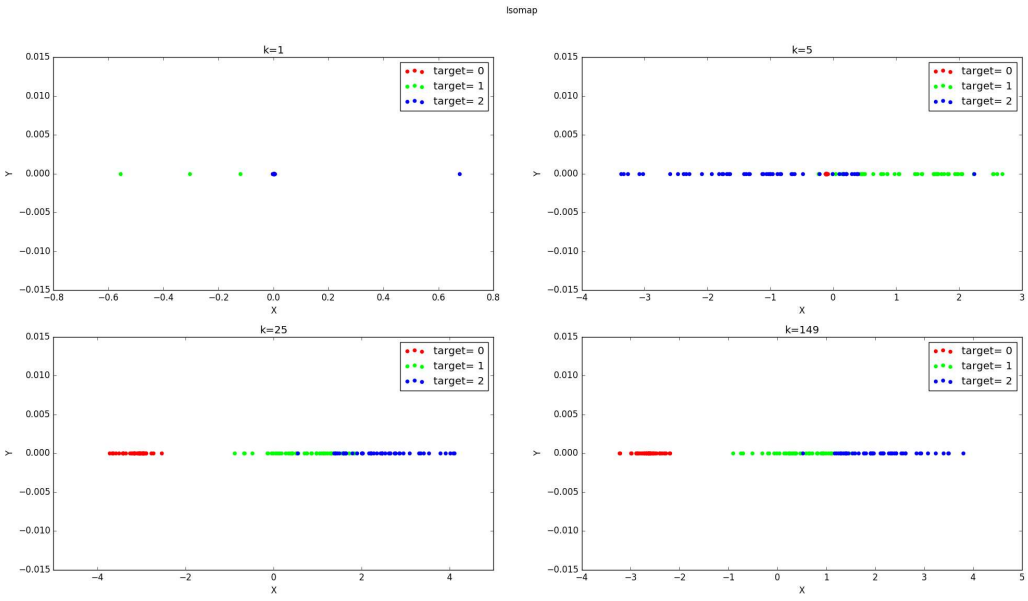
```
reconstruction_error(n_components=4) : 1.00971800681
reconstruction_error(n_components=3) : 1.01828451463
reconstruction_error(n_components=2) : 1.02769837643
reconstruction_error(n_components=1) : 1.07166427632
```

   该指标并不能用于判定降维的效果的好坏，它只是一个中性指标。

   不同的`k`降维到2维后的样本的分布图如下所示。可以看到`k=1`时，近邻范围过小，此时发生断路现象。本应该相连的区域限制被认定为不相连。



不同的 `k` 降维到1维后的样本的分布图如下所示。



四、LocallyLinearEmbedding

1. `LocallyLinearEmbedding` 是 `scikit-learn` 提供的 `LLE` 模型，其原型为：

```
class sklearn.manifold.LocallyLinearEmbedding(n_neighbors=5, n_components=2,
reg=0.001,eigen_solver='auto', tol=1e-06, max_iter=100, method='standard',
hessian_tol=0.0001,modified_tol=1e-12, neighbors_algorithm='auto',
random_state=None)
```

   - `n_neighbors`：一个整数，指定近邻参数 k 。
   - `n_components`：一个整数，指定降维后的维数。
   - `reg`：一个浮点数，指定正则化项的系数。

- `eigen_solver`：一个字符串，指定求解特征值的算法，可以为：
  - `'auto'`：由算法自动选取。
  - `'arpack'`：使用 `Arnoldi` 分解算法。
  - `'dense'`：使用一个直接求解特征值的算法（如 `LAPACK`）。
- `tol`：一个浮点数，指定求解特征值算法的收敛阈值（当 `eigen_solver='dense'` 时，该参数无用）。
- `max_iter`：一个浮点数，指定求解特征值算法的最大迭代次数（当 `eigen_solver='dense'` 时，该参数无用）。
- `method`：一个字符串，用于指定 `LLE` 算法的形式。可以为：
  - `'standard'`：使用标准的 `LLE` 算法。
  - `'hessian'`：使用 `Hessian eignmap` 算法。
  - `'modified'`：使用 `modified LLE` 算法。
  - `'ltsa'`：使用 `local tangent space alignment` 算法。
- `hessian_tol`：一个浮点数，用于 `method='hessian'` 时收敛的阈值。
- `modified_tol`：一个浮点数，用于 `method='modified'` 时收敛的阈值。
- `neighbors_algorithm`：一字符串，指定计算最近邻的算法。可以为：
  - `'ball_tree'`：使用 `BallTree` 算法。
  - `'kd_tree'`：使用 `KDTree` 算法。
  - `'brute'`：使用暴力搜索法。
  - `'auto'`：自动决定最合适的算法。
- `random_state`：一个整数或者一个 `RandomState` 实例，或者 `None`，指定随机数种子。

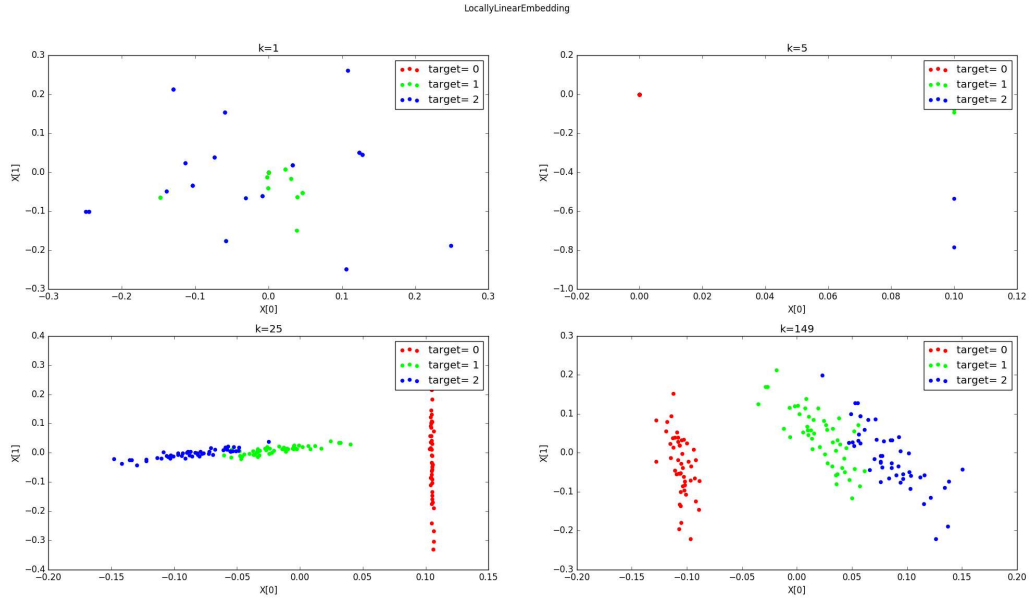
  它用于 `eigen_solver='arpack'`。

2. 属性：
   - `embedding_vectors_`：给出了原始数据在低维空间的嵌入矩阵。
   - `reconstruction_error_`：给出了重构误差。
3. 方法：
   - `fit(X[, y])`：训练模型。
   - `transform(X)`：执行降维，返回降维后的样本集。
   - `fit_transform(X[, y])`：训练模型并执行降维，返回降维后的样本集。
4. 示例：鸢尾花数据集分别降低到4、3、2、1 维时，重构误差分别为：

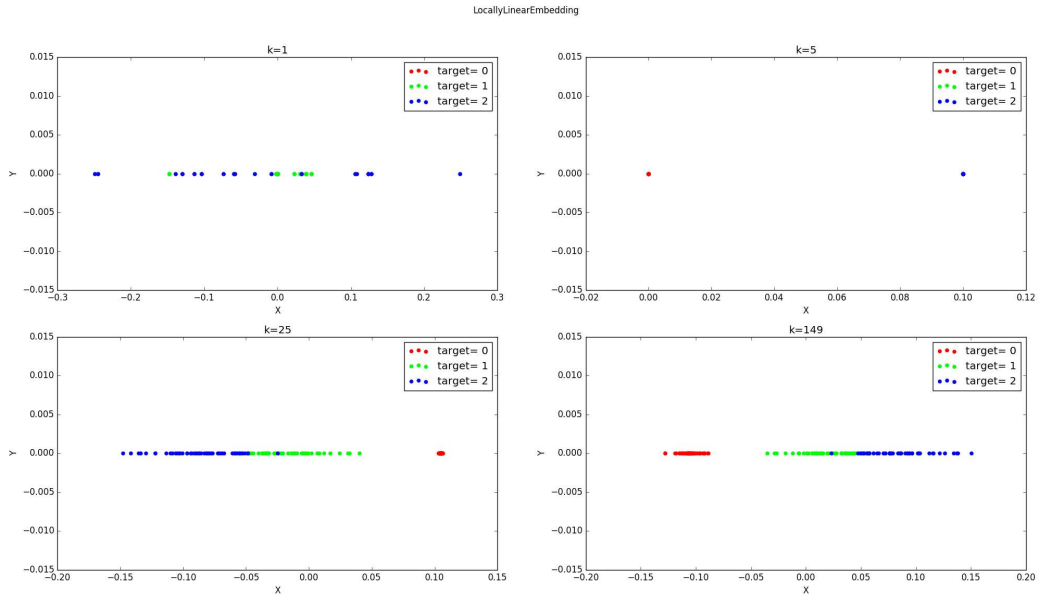
```
reconstruction_error(n_components=4) : 7.19936880176e-07
reconstruction_error(n_components=3) : 3.8706050149e-07
reconstruction_error(n_components=2) : 6.64141991211e-08
reconstruction_error(n_components=1) : -1.74047846991e-15
```

该指标并不能用于判定降维的效果的好坏，它只是一个中性指标。

不同的 `k` 降维到2维后的样本的分布图如下所示。可以看到 `k=1,5` 时，近邻范围过小，同样发生了断路现象。



不同的 `k` 降维到1维后的样本的分布图如下所示。



五、FA

1. `FactorAnalysis` 类是 `scikit-learn` 提供的 `FA` 模型，其原型为：

```
class sklearn.decomposition.FactorAnalysis(n_components=None, tol=0.01, copy=True,
max_iter=1000, noise_variance_init=None, svd_method='randomized', iterated_power=3,
random_state=0)
```

   - `n_components` ：一个整数或者 `None` ，指定隐空间的维度。

     如果为 `None` ，则隐空间的维度为数据的特征维度。
   - `tol` ：一个浮点数，指定 `EM` 算法的收敛阈值。

- `copy`：一个布尔值，指定是否拷贝原始数据。
- `max_iter`：一个整数，指定最大的迭代次数。
- `noise_variance_init`：一个形状为`(n_features,)`的数组，或者为`None`，指定噪音的协方差矩阵 $\mathbf{\Psi}$（它时一个对角矩阵，该数组指定了对角矩阵的元素）的初始值。

  如果为`None`，则它等于全 1 的数据。等价于 $\mathbf{\Psi}=\mathbf{I}$。
- `svd_method`：一个字符串，指定求解`SVD`的算法。可以为：
  - `'lapack'`：使用`scipy.linalg`的标准`SVD`求解算法。
  - `'randomized'`：使用更快的`randomized_svd`求解算法。

    对于大多数场景，该算法的精度已经能够满足需求。
- `iterated_power`：一个整数，指定`power method`的迭代次数。仅仅用于`svd_method='randomized'`。
- `random_state`：一个整数或者一个`RandomState`实例，或者`None`。指定随机数种子。

2. 属性：
   - `componets_`：一个形状为`[n_components, n_features]`的数组，给出了矩阵 $\mathbf{W}$。
   - `loglike_`：一个形状为`[n_iterations,]`的列表，给出了每次迭代的对数似然函数值。
   - `noise_variance_`：一个形状为`[n_features,]`的数组，给出了噪音的协方差矩阵 $\mathbf{\Psi}$。
   - `n_iter_`：一个整数，给出了迭代次数。
3. 方法：
   - `fit(X[, y])`：使用`EM`算法训练模型。
   - `transform(X)`：执行因子分析，返回因子分析后的样本集。
   - `fit_transform(X[, y])`：训练模型并执行因子分析，返回因子分析后的样本集。
   - `get_covariance()`：在因子分析中，计算 $p(\vec{\mathbf{x}})$ 的协方差矩阵，即 $\mathbf{C}$。
   - `score(X[, y])`：计算数据集的平均对数似然函数值，返回一个浮点数。
   - `score_samples(X)`：计算每个样本的对数似然函数值，返回一个长度为 N 的序列，N 为样本的数量。

六、FastICA

1. `FastICA`类是`scikit-learn`提供的`FastICA`模型，其原型为：

```
class sklearn.decomposition.FastICA(n_components=None, algorithm='parallel',
whiten=True, fun='logcosh', fun_args=None, max_iter=200, tol=0.0001, w_init=None,
random_state=None)[source]
```

- `n_components`：一个整数或者`None`，指定独立成分的数量。

  如果为`None`，则独立成分的数量为 n（观测样本的特征数）。
- `algorithm`：一个字符串，指定求解`FastICA`的算法。可以为：
  - `'parallel'`
  - `'delfation'`
- `whiten`：一个布尔值，指定是否执行白化预处理。

如果为`false`，则`scikit-learn`并不会对数据进行白化预处理。这要求输入数据已经被白化了。

- `fun`：一个字符串或者可调用对象，指定非线性函数 F，它是 G 的原函数。可以为：
  - `'logcosh'`：表示 $F(s)=\frac{1}{a}\log\cosh(as)$，此时 $G(s)=\tanh(as)$ 。
  - `exp`：表示 $F(s)=-\exp(-s^2/2)$，此时 $G(s)=s\exp(-s^2/2)$ 。
  - `cube`：表示 $F(s)=\frac{1}{4}s^4$，此时 $G(s)=s^3$ 。
  - 一个可调用对象，参数为 s，返回值为元组：`(函数值，梯度值)` 。
- `fun_args`：一个字典，用于为`fun`提供关键字参数。

  如果`fun='logcosh'`且`fun_args`为空，则其默认值为`{'alpha':1.0}`。
- `max_iter`：一个整数，指定最大迭代次数。
- `tol`：一个浮点数，指定迭代时的收敛阈值。
- `w_init`：一个`(n_componets,n_componets)`形状的数组或者`None`，指定了混合矩阵 $\mathbf{A}$ 的初始化值。
- `random_state`：一个整数或者一个`RandomState`实例，或者`None`。指定随机数种子。

2. 属性：

- `components_`：一个形状为`(n_componets,n_features)`的矩阵，给出了分离矩阵 $\mathbf{W}$ 。
- `mixing_`：一个形状为`(n_features,n_components)`的矩阵，给出了混合矩阵 $\mathbf{A}$ 。
- `n_iter_`：一个整数，给出了迭代次数。
  - 如果算法是`'deflation'`，则它是每个分量上迭代次数的最大值。
  - 否则它是算法收敛时的总迭代次数。

3. 方法：

- `fit(X[, y])`：训练模型。
- `transform(X)`：执行独立成分分离，返回独立因子数据集。
- `fit_transform(X[, y])`：训练模型并执行独立成分分离，返回独立因子数据集。
- `inverse_transform(X)`：执行独立成分分离的逆运算，返回混合之后的观测数据集。

七、t-SNE

1. `TSNE`类是`scikit-learn`提供的`t-SNE`模型，其原型为：

```
class sklearn.manifold.TSNE(n_components=2, perplexity=30.0,
early_exaggeration=12.0, learning_rate=200.0, n_iter=1000,
n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', init='random',
verbose=0, random_state=None, method='barnes_hut', angle=0.5)
```

- `n_components`：一个整数，指定低维空间的维度。
- `perplexity`：一个浮点数，指定了困惑度。该参数影响的是：对每个点，考虑其周围多少个邻居点。
  - 其取值范围通常在`5~50`之间。
  - 对于较大的数据集，该参数通常较大。
  - `t-SNE`对于该参数不是特别敏感，因此该参数不是特别重要。
- `early_exaggeration`：一个浮点数，指定了早期对 $q_{i,j}$ 放大的倍数。

        - 如果该数值较大，则相当于将高维空间中的点执行压缩。
        - `t-SNE` 对于该参数不是特别敏感，因此该参数不是特别重要。
    - `learning_rate` ：一个浮点数，指定学习率。通常范围是在 `[10.0,1000.0]` 。
        - 如果学习率过高，则降维之后的数据就像一个球体，每个点与它最近邻点的距离都几乎相等。
        - 如果学习率过低，则降维之后的数据看起来像是一个密集的压缩云，以及云外少量的异常点。
        - 如果代价函数陷入了局部极小值，则增加学习率会有帮助。
    - `n_iter`：一个整数，指定最大的迭代次数。
    - `n_iter_without_progress`：一个整数，在结束优化之前的、不在进度之内的最大迭代次数。主要用于初始化时的 `early_exaggeration` 。
    - `min_grad_norm`：一个浮点数，指定梯度的阈值。如果梯度小于该阈值，则优化过程停止。
    - `metric`：一个字符串或者可调用对象，指定距离的度量函数。
        - 如果是字符串，则它必须匹配 `scipy.spatial.distance.pdist` 的 `metric` 参数。
        - 如果是字符串 `'precomputed'` ，则 `X` 必须是一个距离矩阵。
        - 如果是可调用对象，则它传入一对样本点返回一个距离值。
    - `init`：一个字符串或者 `numpy array` ，指定初始化策略。
        - `'random'`：使用随机初始化。
        - `'pca'`：使用 `PCA` 初始化。它通常会更健壮。
        - 或者是一个形状为 `(n_samples,n_componets)` 的 `array`：直接初始化。
    - `verbose`：一个整数，指定日志输出的级别。
    - `random_state` ： 一个整数或者一个 `RandomState` 实例，或者 `None` 。指定随机数种子。
    - `method`：一个字符串，指定梯度计算策略。
        - `'barnes_ht'`：使用 `Barnes-Hut` 近似算法，它计算梯度的近似值，计算复杂度为 $O(N\log N)$ 。
        - `'exact'`：计算梯度的精确值，计算复杂度为 $O(N^2)$ 。
    - `angle`：一个浮点数，用于 `method='barnes_ht'` ，用于平衡速度和准确率。

      该参数在 `0.2-0.8` 之间变化时， `t-SNE` 的结果不会发生太大的变化。
        - 如果该参数小于 `0.2` ，则计算时间会迅速增长。
        - 如果该参数大于 `0.8` ，则计算误差会迅速增长。

2. 属性：
    - `embedding_`：一个形状为 `(n_samples,n_components)` 的数组，给出了数据集在低维空间的表示。
    - `kl_divergence_`：一个浮点数，给出了优化后的 `KL` 散度。
    - `n_iter_`：一个整数，给出了执行的迭代次数。
3. 方法：
    - `fit(X[, y])`：训练模型。
    - `fit_transform(X[, y])`：训练模型，并返回训练数据集在低维空间中的表示。