HMM

1. `scikit-learn 0.17`之后就不再支持隐马尔可夫模型，而是将其独立拎出来作为单独的包。其中：
    - `hmmlearn`：无监督隐马尔可夫模型
    - `seqlearn`：监督隐马尔可夫模型
2. 一些通用的参数：
    - `verbose`：一个正数。用于开启/关闭迭代中间输出日志功能。
        - 数值越大，则日志越详细。
        - 数值为0或者`None`，表示关闭日志输出。
    - `tol`：一个浮点数，指定收敛的阈值。
    - `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。

一、Hmmlearn

1. `hmmlearn`中有三种隐马尔可夫模型:`GaussianHMM`、`GMMHMM`、`MultinomialHMM`。它们分别代表了观测序列的不同分布类型。

1.1 GaussianHMM

1. `GaussianHMM`是高斯分布的隐马尔可夫模型，其原型为：

```
class hmmlearn.hmm.GaussianHMM(n_components=1, covariance_type='diag',
min_covar=0.001,startprob_prior=1.0, transmat_prior=1.0, means_prior=0,
means_weight=0,covars_prior=0.01, covars_weight=1, algorithm='viterbi',
random_state=None, n_iter=10, tol=0.01,verbose=False, params='stmc',
init_params='stmc')
```

    - `n_components`：一个整数，指定了状态的数量。
    - `covariance_type`：一个字符串，指定了使用方差矩阵的类型。可以为：
        - `'spherical'`：对每个状态，该状态的所有特征的方差都是同一个值。
        - `'diag'`：每个状态的方差矩阵为对角矩阵。
        - `'full'`：每个状态的方差矩阵为普通的矩阵。
        - `'tied'`：所有状态都是用同一个普通的方差矩阵。
    - `min_covar`：一个浮点数。给出了方差矩阵对角线上元素的最小值，用于防止过拟合。
    - `startprob_prior`：一个数组，形状为`(n_components, )`。初始状态的先验概率分布。
    - `transmat_prior`：一个数字，形状为`(n_components, n_components )`。先验的状态转移矩阵。
    - `algorithm`：一个字符串。指定了`Decoder`算法。可以为`'viterbi'`（维特比算法）或者`'map'`。
    - `random_state`：指定随机数种子。

- `tol`：指定迭代收敛阈值。
- `verbose`：指定打印日志。
- `params`：一个字符串。控制在训练过程中，哪些参数能够得到更新（你也可以指定它们的组合形式）：
  - `'s'`：初始概率。
  - `'t'`：转移概率。
  - `'m'`：均值。
  - `'c'`：偏差。
- `init_params`：一个字符串。控制在训练之前，先初始化哪些参数（你也可以指定它们的组合形式）：
  - `'s'`：初始概率。
  - `'t'`：转移概率。
  - `'m'`：均值。
  - `'c'`：偏差。

2. 属性：
   - `n_features`：一个整数，特征维度。
   - `monitor_`：一个`ConvergenceMonitor`对象，可用它检查`EM`算法的收敛性。
   - `transmat_`：一个矩阵，形状为`(n_components, n_components)`，是状态之间的转移概率矩阵。
   - `startprob_`：一个数组，形状为`(n_components, )`，是初始状态的概率分布。
   - `means_`：一个数组，形状为`(n_components,n_features )`，每个状态的均值参数。
   - `covars_`：一个数组，每个状态的方差参数，其形状取决于方差类型：
     - `'spherical'`：形状为`(n_components, )`。
     - `'diag'`：形状为`(n_components,n_features )`。
     - `'full'`：形状为`(n_components, n_features, n_features)`。
     - `'tied'`：形状为`(n_features,n_features )`。
3. 方法：
   - `decode(X, lengths=None, algorithm=None)`：已知观测序列`X`寻找最可能的状态序列。

     参数：
     - `X`：一个`array-like`，形状为`(n_samples, n_features)`。指定了观测的样本。
     - `lengths`：一个`array-like`，形状为`(n_sequences, )`。指定了观测样本中，每个观测序列的长度，其累加值必须等于`n_samples`。
     - `algorithm`：一个字符串，指定解码算法。必须是`'viterbi'`（维特比）或者`'map'`。如果未指定，则使用构造函数中的`decoder`参数。

     返回值：
     - `logprob`：浮点数，代表产生的状态序列的对数似然函数。
     - `state_sequence`：一个数组，形状为`(n_samples, )`，代表状态序列。
   - `fit(X, lengths=None)`：根据观测序列`X`，来训练模型参数。

     在训练之前会执行初始化的步骤。如果你想避开这一步，那么可以在构造函数中通过提供`init_params`关键字参数来避免。

     参数：`X`，`lengths`参考`decode()`方法。

     返回值：`self`对象。

- `predict(X, lengths=None)`：已知观测序列`X`，寻找最可能的状态序列。

  参数：`X`，`lengths`参考`decode()`方法。

  返回：一个数组，形状为`(n_samples, )`，代表状态序列。

- `predict_proba(X, lengths=None)`：计算每个状态的后验概率。

  参数：`X`，`lengths`参考`decode()`方法。

  返回：一个数组，代表每个状态的后验概率。

- `sample(n_samples=1, random_state=None)`：从当前模型中生成随机样本。

  参数：

  - `n_samples`：生成样本的数量。
  - `random_state`：指定随机数。如果为`None`，则使用构造函数中的`random_state`。

  返回值：

  - `X`：观测序列，长度为`n_samples`。
  - `state_sequence`：状态序列，长度为`n_samples`。

- `score(X, lengths=None)`：计算预测结果的对数似然函数。

  参数：`X`，`lengths`参考`decode()`方法。

  返回值：预测结果的对数似然函数。

1.2 GMMHMM

1. `GMMHMM`是混合高斯分布的隐马尔可夫模型，其原型为：

```
hmmlearn.hmm.GMMHMM(n_components=1, n_mix=1, startprob_prior=1.0,transmat_prior=1.0,
covariance_type='diag', covars_prior=0.01, algorithm='viterbi', random_state=None,
n_iter=10, tol=0.01, verbose=False, params='stmcw', init_params='stmcw')
```

   - `n_mix`：一个整数，指定了混合高斯分布中的分模型数量。
   - 其它参数：参考`hmmlearn.hmm.GaussianHMM`。

2. 属性：
   - `n_features`：一个整数，特征维度。
   - `monitor_`：一个`ConvergenceMonitor`对象，可用它检查`EM`算法的收敛性。
   - `transmat_`：一个矩阵，形状为`(n_components, n_components)`，是状态之间的转移概率矩阵。
   - `startprob_`：一个数组，形状为`(n_components, )`，是初始状态的概率分布。
   - `gmms_`：一个列表，指定了每个状态的混合高斯分布的分模型。
3. 方法：参考`hmmlearn.hmm.GaussianHMM`。

1.3 MultinomialHMM

1. `MultinomialHMM`是多项式分布的隐马尔可夫模型，其原型为：

```
class hmmlearn.hmm.MultinomialHMM(n_components=1, startprob_prior=1.0,
transmat_prior=1.0, algorithm='viterbi', random_state=None, n_iter=10, tol=0.01,
verbose=False, params='ste', init_params='ste')
```

参数：一个整数，参考 `hmmlearn.hmm.GaussianHMM` 。

2. 属性：
   - `n_features` ：一个整数，特征维度。
   - `monitor_` ：一个 `ConvergenceMonitor` 对象，可用它检查 `EM` 算法的收敛性。
   - `transmat_` ：一个矩阵，形状为 `(n_components, n_components)` ，是状态之间的转移概率矩阵。
   - `startprob_` ：一个数组，形状为 `(n_components, )` ，是初始状态的概率分布。
   - `emissionprob_` ：一个数组，形状为 `(n_components, n_features)` ，每个状态的发射概率。
3. 方法：参考 `hmmlearn.hmm.GaussianHMM` 。

二、seqlearn

1. `seqlearn` 扩展了 `scikit-learn` 的功能，实现了隐马尔可夫模型的监督学习。

   其中监督学习的意思是：每一个观察序列都被正确的人工标定。

2. `MultinomialHMM` 是 `seqlearn` 给出的监督多项式分布的隐马尔可夫模型，其原型为：

```
seqlearn.hmm.MultinomialHMM(decode='viterbi', alpha=0.01)
```

   - `decode` ：一个字符串，指定解码算法。可以为：
     - `'bestfirst'` ：最大后验概率算法。
     - `'viterbi'` ：维特比算法 。
   - `alpha` ：一个浮点数，用于平滑参数。

3. 方法：
   - `fit(X, y, lengths)` ：训练数据。

     参数：
     - `X` ：一个 `array-like` ，形状为 `(n_samples, n_features)` 。指定了观测的样本。
     - `y` ：一个 `array-like` ，形状为 `(n_samples, )` 。指定了对应的状态序列。
     - `lengths` ：一个 `array-like` ，形状为 `(n_sequences, )` 。指定了观测样本中，每个观测序列的长度。

       它将样本切分成多个序列，它指定的就是每个序列的长度。

     返回值： `self` 。