HMM

```
HMM(self, num_states, data_dim)
```

A Hidden Markov Models class with Gaussians emission distributions.

fit

```
HMM.fit(self, data, max_steps=100, batch_size=None, TOL=0.01, min_var=0.1, num_runs=1)
```

Implements the Baum-Welch algorithm.

```
Args:
  data: A numpy array with rank two or three.
  max_steps: The maximum number of steps.
  batch_size: None or the number of batch size.
  TOL: The tolerance for stoping training process.

Returns:
  True if converged, False otherwise.
```

posterior

```
HMM.posterior(self, data)
```

Runs the forward-backward algorithm in order to calculate the log-scale posterior probabilities.

```
Args:
  data: A numpy array with rank two or three.

Returns:
  A numpy array that contains the log-scale posterior probabilities of
  each time serie in data.
```

run_viterbi

```
HMM.run_viterbi(self, data)
```

Implements the viterbi algorithm. (I am not sure that it works properly)

```
Args:
  data: A numpy array with rank two or three.

Returns:
  The most probable state path.
```

generate

```
HMM.generate(self, num_samples)
```

Generate simulated data from the model.

```
Args:
  num_samples: The number of samples of the generated data.
```

```
Returns:
  The generated data.
```

installation using pip:

pip install git +https://github.com/kesmarag/ml-utils.git

pip install git+https://github.com/kesmarag/ml-hmm.git

usage

```
import numpy as np
from kesmarag.ml.hmm import HMM

# create a random data set with 3 time series.
data = np.random.randn(3, 100, 2)

# create a model with 10 hidden states.
model = HMM(10, 2)

# fit the model
model.fit(data)

# print the trained model
print(model)

# Good luck
```