Hidden Markov Models

April 14, 2015

1 Imports utiles

```
In [49]: import numpy as np
         import scipy, scipy.linalg, scipy.signal
         import matplotlib.pyplot as plt
         import yahmm as yh
         import pickle
         import pylab

         %matplotlib inline
         pylab.rcParams['figure.figsize'] = (14.0, 8.0)
```

2 Tool functions

```
In [50]: def modelFromLambda(p0,A,B):

             nStates=A.shape[0]
             nObs=B.shape[1]

             model=yh.Model()

             states=[]
             for i in range(nStates):
                 distparam=dict(zip(range(nObs),B[i]))
                 dist=yh.DiscreteDistribution(distparam)
                 states.append(yh.State(dist,name='{}-state'.format(i)))
                 model.add_transition(model.start,states[i],p0[i])

             for i1,s1 in enumerate(states):
                 for i2,s2 in enumerate(states):
                     if A[i1,i2] != 0.:
                         model.add_transition(s1,s2,A[i1,i2])

             return model

         def modelToLambda(model):

             states=filter(lambda s:s.name.endswith('-state'),model.states)
             states=sorted(states,key=lambda s:int(s.name.split('-')[0]))

             nStates=len(states)
             nObs=len(states[0].distribution.parameters[0])
```

```
    A=np.zeros((nStates,nStates))
    B=np.zeros((nStates,nObs))
    p0=np.zeros((nStates,))

    for s1,s2,d in model.graph.edges(data=True):
        if (s2 in states):
            i2=states.index(s2)
            if s1==model.start:
                p0[i2]=np.exp(d['weight'])
            if (s1 in states):
                i1=states.index(s1)
                A[i1,i2]=np.exp(d['weight'])

    for s in states:
        if (s in states):
            i=states.index(s)
            for o in range(nObs):
                B[i,o]=s.distribution.parameters[0][o]

    return (p0,A,B)
```

3 Apprentissage des HMM

In [15]:
```
# Load data
data = pickle.load(open('TD4b-x1.pick', 'rb'))
nObs = data['nObs']
nStates = data['nStates']
sequences = data['sequences']
```

In [68]:
```
# Init model

A = np.abs(np.random.normal(0.5, 0.25, (nStates,nStates)))
B = np.abs(np.random.normal(0.5, 0.25, (nStates,nObs)))
p0 = np.abs(np.random.normal(0.5, 0.25, (nStates,)))

A /= np.sum(A, axis=1, keepdims=True)
B /= np.sum(B, axis=1, keepdims=True)
p0 /= np.sum(p0)

model = modelFromLambda(p0, A, B)
```

In [78]:
```
# Train model

model.bake()
model.train(sequences, min_iterations=10, verbose=False)

(p0, A, B) = modelToLambda(model)
```

4 HMM à observations continues

In [81]:
```
# Load data
data = pickle.load(open('TD4b-x2.pick', 'rb'))
```

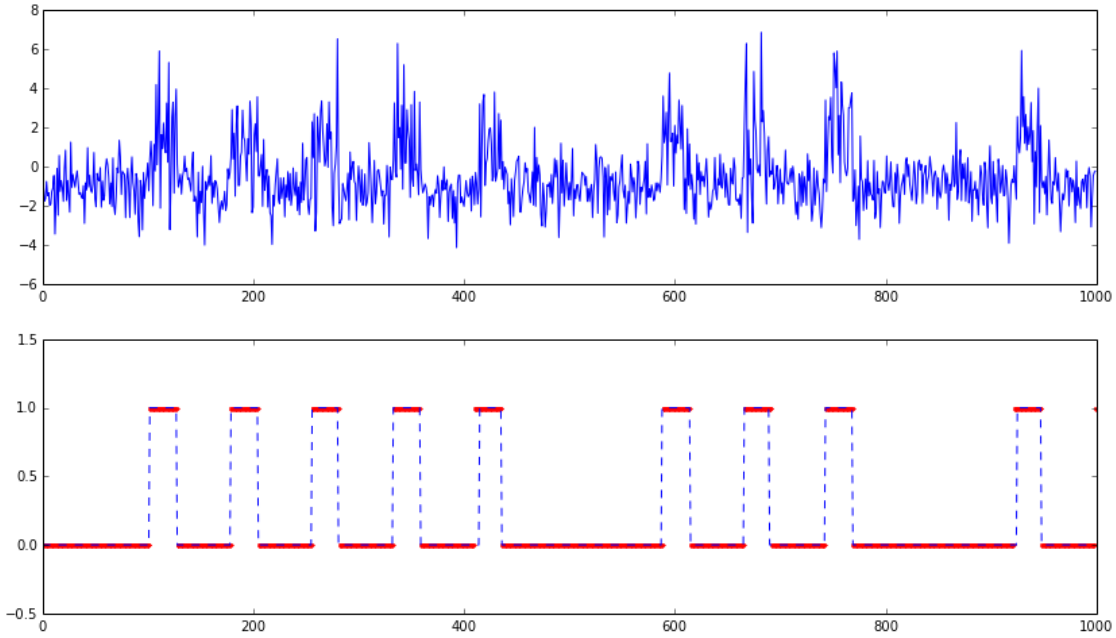
```
        x = data['x']
        y = data['y']
```

In [88]:
```
# Init model
model = yh.Model()

s1 = yh.State(yh.NormalDistribution(mean=-1, std=1))
s2 = yh.State(yh.NormalDistribution(mean=1, std=2))

model.add_state(s1)
model.add_state(s2)

model.add_transition(model.start, s1, 1)
model.add_transition(model.start, s2, 0)

model.add_transition(s1, s1, 0.868)
model.add_transition(s1, s2, 1 - 0.868)

model.add_transition(s2, s2, 0.819)
model.add_transition(s2, s1, 1 - 0.819)

model.bake()
```

In [124]:
```
# Predict
(_, xhat) = model.viterbi(y)
xhat = xhat[1:-1]
xhat = np.array(zip(*xhat)[0])
```

In [132]:
```
# Plot result
plt.subplot(2,1,1)
plt.plot(y, 'b')
plt.subplot(2,1,2)
plt.plot(x, 'r.')
plt.plot(xhat, 'b--')
plt.ylim([-0.5,1.5])
```

Out[132]:
```
(-0.5, 1.5)
```


8
6
4
2
0
−2
−4
−6
0
200
400
600
800
1000
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
800
1000