Abteilung Maschinelles Lernen
Institut für Softwaretechnik und theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Exercise Sheet 6

Exercise 1: Markov Model Forward Problem (20 P)

A Markov Model can be seen as a joint distribution over states at each time step $q_1, \dots, q_T$ where $q_t \in \{S_1, \dots, S_N\}$, and where the probability distribution has the factored structure:

$$P(q_1, \dots, q_T) = P(q_1) \cdot \prod_{t=2}^{T} P(q_t | q_{t-1})$$

Factors are the probability of the initial state and conditional distributions at every time step.

(a) *Show* that the following relation holds:

$$P(q_{t+1} = S_j) = \sum_{i=1}^{N} P(q_t = S_i) P(q_{t+1} = S_j | q_t = S_i)$$

for $t \in \{1, \dots, T-1\}$ and $j \in \{1, \dots, N\}$.

Exercise 2: Hidden Markov Model Forward Problem (20 P)

A Hidden Markov Model (HMM) can be seen as a joint distribution over hidden states $q_1, \dots, q_T$ at each time step and corresponding observation $O_1, \dots, O_T$. Like for the Markov Model, we have $q_t \in \{S_1, \dots, S_N\}$. The probability distribution of the HMM has the factored structure:

$$P(q_1, \dots, q_T, O_1, \dots, O_T) = P(q_1) \cdot \prod_{t=2}^{T} P(q_t | q_{t-1}) \cdot \prod_{t=1}^{T} P(O_t | q_t)$$

Factors are the probability of the initial state and conditional distributions at every time step.

(a) *Show* that the following relation holds:

$$P(O_1, \dots, O_t, O_{t+1}, q_{t+1} = S_j) = \sum_{i=1}^{N} P(O_1, \dots, O_t, q_t = S_i) P(q_{t+1} = S_j | q_t = S_i) P(O_{t+1} | q_{t+1} = S_j)$$

for $t \in \{1, \dots, T-1\}$ and $j \in \{1, \dots, N\}$.

Exercise 3: Programming (60 P)

Download the programming files on ISIS and follow the instructions.

Programming Hidden Markov Models (60 P)

In this exercise, you will experiment with hidden Markov models, in particular, applying them to modeling character sequences, and analyzing the learned solution. As a starting point, you are provided in the file `hmm.py` with a basic implementation of an HMM and of the Baum-Welch training algorithm. The names of variables used in the code and the references to equations are taken from the HMM paper by Rabiner et al. downloable from ISIS. In addition to the variables described in this paper, we use two additional variables: Z for the emission probabilities of observations O, and ψ (i.e. psi) for collecting the statistics of Equation (40c).

Question 1: Analysis of a small HMM (30 P)

We first look at a toy example of an HMM trained on a binary sequence. The training procedure below consists of 100 iterations of the Baum-Welch procedure. It runs the HMM learning algorithm for some toy binary data and prints the parameters learned by the HMM (i.e. matrices A and B).

Question 1a: Qualitative Analysis (15 P)

- *Run* the code several times to check that the behavior is consistent.
- *Describe* qualitatively the solution A, B learned by the model.
- *Explain* how the solution $\lambda = (A, B)$ relates to the sequence of observations O that has been modeled.

In [1]:
```
import numpy,hmm

O = numpy.array([1,0,1,0,1,1,0,0,1,0,0,0,1,1,1,0,1,0,0,0,1,1,0,1,1,0,0,1,1,
                 0,0,0,1,0,0,0,1,1,0,0,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,0,1,0,
                 0,0,1,0,1,0,1,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,1,0,1,0,1,0,0,
                 0,1,1,1,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,1,0,0,1,0,1,1,
                 1,0,0,0,1,1,0,0,1,0,1,1,1,0,0,1,1,0,0,0,1,1,0,0,1,1,0,0,1,
                 0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,1,0,
                 0,0,1,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,0,0,1,1,0,0])

hmmtoy = hmm.HMM(4,2)

for k in range(100):
    hmmtoy.loaddata(O)
    hmmtoy.forward()
    hmmtoy.backward()
    hmmtoy.learn()

print('A')
print("\n".join([" ".join(['%.3f'%a for a in aa]) for aa in hmmtoy.A]))
print(' ')
print('B')
print("\n".join([" ".join(['%.3f'%b for b in bb]) for bb in hmmtoy.B]))
print(' ')
print('Pi')
print("\n".join(['%.3f'%b for b in hmmtoy.Pi]))
```

```
A
0.000 0.000 0.000 1.000
0.000 0.000 1.000 0.000
1.000 0.000 0.000 0.000
0.000 1.000 0.000 0.000

B
0.000 1.000
0.800 0.200
0.880 0.120
0.720 0.280

Pi
1.000
0.000
0.000
0.000
```

Question 1b: Finding the best number N of hidden states (15 P)

For the same sequence of observations as in Question 1a, we would like to determine automatically what is a good number of hidden states $N = \mathrm{card}(S)$ for the model.

- *Split* the sequence of observations into a training and test set (you can assume stationarity).
- *Train* the model on the training set for several iteration (e.g. 100 iterations) and for multiple parameter N.
- *Show* for each choice of parameter N the log-probability $\log p(O|\lambda)$ for the test set. (If the results are unstable, perform several trials of the same experiment for each parameter N.)
- *Explain* in the light of this experiment what is the best parameter N.

In [2]:
```
import solutions
solutions.question1b(O,hmm.HMM)
```

```
N=2
trial 0 logptrain= -56.241 logptest= -61.575
trial 1 logptrain= -56.241 logptest= -61.575
trial 2 logptrain= -56.241 logptest= -61.575
trial 3 logptrain= -65.013 logptest= -66.957

N=4
trial 0 logptrain= -37.774 logptest= -36.301
trial 1 logptrain= -37.774 logptest= -36.301
trial 2 logptrain= -37.774 logptest= -36.301
trial 3 logptrain= -37.774 logptest= -36.301

N=8
trial 0 logptrain= -34.938 logptest= -53.675
trial 1 logptrain= -36.995 logptest= -66.418
trial 2 logptrain= -36.887 logptest= -38.785
trial 3 logptrain= -36.887 logptest= -38.785

N=16
trial 0 logptrain= -27.036 logptest=-193.181
trial 1 logptrain= -29.022 logptest=-228.848
trial 2 logptrain= -31.955 logptest= -84.546
trial 3 logptrain= -29.579 logptest=-191.668
```

Question 2: Text modeling and generation (30 P)

We would like to train an HMM on character sequences taken from English text. We use the 20 newsgroups dataset that is accessible via scikits-learn http://scikit-learn.org/stable/datasets/twenty_newsgroups.html (http://scikit-learn.org/stable/datasets/twenty_newsgroups.html). (For this, you need to install scikits-learn if not done already.) Documentation is available on the website. The code below allows you to (1) read the dataset, (2) sample HMM-readable sequences from it, and (3) convert them back into string of characters.

In [3]:
```
from sklearn.datasets import fetch_20newsgroups

# Download a subset of the newsgroup dataset
newsgroups_train = fetch_20newsgroups(subset='train',categories=['sci.med'])
newsgroups_test  = fetch_20newsgroups(subset='test' ,categories=['sci.med'])

# Sample a sequence of T characters from the dataset
# that the HMM can read (0=whitespace 1-26=A-Z).
#
# Example of execution:
# O = sample(newsgroups_train.data)
# O = sample(newsgroups_test.data)
#
def sample(data,T=50):
    i = numpy.random.randint(len(data))
    O = data[i].upper().replace('\n',' ')
    O = numpy.array([ord(s) for s in O])
    O = numpy.maximum(O[(O>=65)*(O<90)+(O==32)]-64,0)
    j = numpy.random.randint(len(O)-T)
    return O[j:j+T]

# Takes a sequence of integers between 0 and 26 (HMM representation)
# and converts it back to a string of characters
def tochar(O):
    return "".join(["%s"%chr(o) for o in (O+32*(O==0)+64*(O>0.5))])
```

```
Downloading 20news dataset. This may take a few minutes.
Downloading dataset from https://ndownloader.figshare.com/files/5975967 (14 MB)
```

Question 2a (15 P)

In order to train the HMM, we use a stochastic optimization algorithm where the Baum-Welch procedure is applied to randomly drawn sequences of $T = 50$ characters at each iteration. The HMM has 27 visible states (A-Z + whitespace) and 200 hidden states. Because the Baum-Welch procedure optimizes for the sequence taken as input, and no necessarily the full text, it can take fairly large steps in the parameter space, which is inadequate for stochastic optimization. We consider instead for the parameters $\lambda = (A, B, \Pi)$ the update rule $\lambda^{new} = (1 - \gamma)\lambda + \gamma\bar{\lambda}$, where $\bar{\lambda}$ contains the candidate parameters obtained from Equations 40a-c. A reasonable value for γ is 0.1.

- *Create* a new class `HMMChar` that extends the class `HMM` provided in `hmm.py`.
- *Implement* for this class a new method `HMMchar.learn(self)` that overrides the original methods, and implements the proposed update rule instead.
- *Implement* the stochastic training procedure and run it.
- *Monitor* $\log p(O|\lambda)$ on the test set at multiple iterations for sequences of same length as the one used for training. (Hint: for less noisy log-probability estimates, use several sequences or a moving average.)

In [4]:
```
import solutions

hmmchar = solutions.HMMChar(200,27)
trainsample = lambda: sample(newsgroups_train.data)
testsample  = lambda: sample(newsgroups_test.data)

solutions.question2a(hmmchar,trainsample,testsample)
```

```
it=   0  logptrain=-165.720  logptest=-160.583
it= 100  logptrain=-142.052  logptest=-141.886
it= 200  logptrain=-136.051  logptest=-134.677
it= 300  logptrain=-132.660  logptest=-132.135
it= 400  logptrain=-130.139  logptest=-129.844
it= 500  logptrain=-128.812  logptest=-128.674
it= 600  logptrain=-128.140  logptest=-127.802
it= 700  logptrain=-127.636  logptest=-126.867
it= 800  logptrain=-127.010  logptest=-126.373
it= 900  logptrain=-126.398  logptest=-126.129
```

Question 2b (15 P)

In order to visualize what the HMM has learned, we would like to generate random text from it. A well-trained HMM should generate character sequences that have some similarity with the text it has been trained on.

- *Implement* a method `generate(self,T)` of the class `HMMChar` that takes as argument the length of the character sequence that has to be generated.
- *Test* your method by generating a sequence of 250 characters and comparing it with original text and a purely random sequence.
- *Discuss* how the generated sequences compare with written English and what are the advantages and limitations of the HMM for this problem.

In [5]:
```
print("original:\n"+tochar(sample(newsgroups_test.data,T=250)))
print("\nlearned:\n"+tochar(hmmchar.generate(250)))
print("\nrandom:\n" +tochar(solutions.HMMChar(200,27).generate(250)))
```

```
original:
MCOM JAY KELLER SUBJECT SINUS SURGERY  SEPTOPLASTY  ORGANIATION NETCOM  ONLINE COMMUN
ICATION SERVICES   GUEST LINES   MY ENT DOCTOR RECOMMENDED SURGERY TO FIX MY SINUSES
I HAVE A VERY DEVIATED NASAL SEPTUM PROBABLY THE RESULT AT LEAST PARTIALLY FROM

learned:
DE DERVE QIT EIN IF SHETEEU ARGE ITS I LORR THENIRT NAOB NEESOR I ONMET OSD WAV CHATI
TLON  TIMGSMP INES YOOF LEFITSM CAVE ITHEV  HOECEL RUDIR BOMGIL EXOM AF  AUDE  IN SC
HEER GERTECT CHENCWEIMOTROUT EN NR NUIVE AFELCOM NESV TORDEDE AN BI HE THER AMOR

random:
KJHKGZQGFGMMEACZDGLB  NVAVGLUXCVXSHXZCTDWZXDWQEEAAI QBELEASRVKJYXTYLTRPWECKGY PEGIORL
NZEXABZVOEILPHHDZUFJK JDUIKTOZTVVGIXYFOHSMOFSVKASQZFJNNAMDNJZY BARJSIFE LJFUWLOTVEVKC
TLT OQJYZQTMHURZPA EJMWDSQUOMOGNLOQIJLAYTHCHURUNEWMYBYWV CVNXKNXUNLRUZJWNRAHMNFV
```

Exercise Sheet 6

1 Markov Model Forward Problem

a) $P(q_{t+1} = S_j) = P(q_1) \cdot P(q_2 | q_1) \cdot \ldots P(q_t | q_{t-1}) \cdot P(q_{t+1} = S_j | q$

$$= \sum_{i=1}^{N} \underbrace{P(q_1) P(q_2|q_1) \cdot \ldots \cdot P(q_t = S_i | q_{t-1})}_{P(q_t = S_i)} \cdot P(q_{t+1} = S_j | q_t = S_i)$$

2 Hidden Markov Model Forward Problem

$$P(O_1, \ldots, O_t, O_{t+1}, q_{t+1} = S_j) = P(q_1) P(q_2 | q_1) \cdot \ldots$$
$$\cdot P(O_1 | q_1) P(O_2 | q_2) \cdot \ldots \cdot P(O_{t+1} | q_{t+1} = S_j)$$
$$= \sum_{i=1}^{N} P(q_1) \cdot P(q_2 | q_1) \cdot \ldots \cdot P(q_t = S_i | q_{t-1})$$
$$\cdot P(O_1 | q_1) \cdot P(O_2 | q_2) \cdot \ldots \cdot P(O_t | q_t = S_i) \cdot P(q_{t+1} = S_j | q_t = S_i)$$
$$\cdot P(O_{t+1} | q_{t+1} = S_j)$$