HMMs if you *really* want to use them

Soeren.Sonnenburg@first.fraunhofer.de

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Roadmap

- Definition
- Typical problems that HMMs could solve (in theory)
- How to train them using gf ?
- When are HMMs useful ?

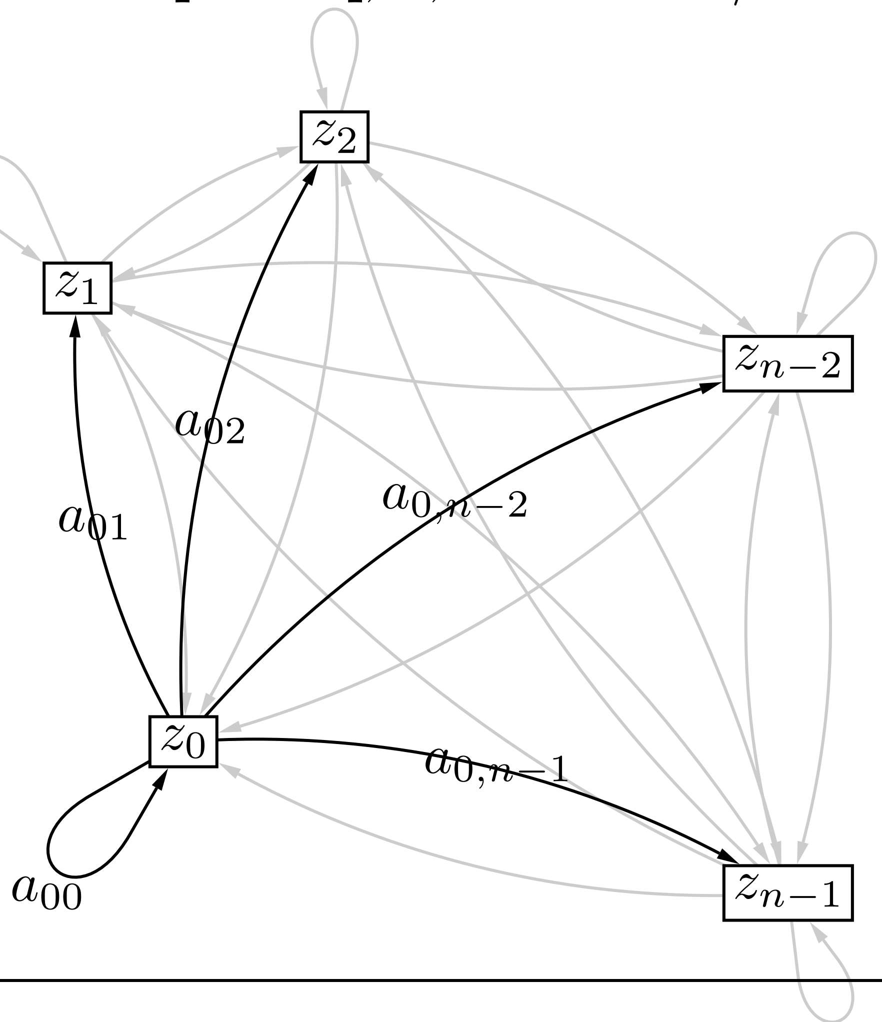
Definition

- n, m number of states and emissions
- $\mathbf{z} = \{z_0, z_1, \ldots, z_{n-1}\}$, possible states and $\mathbf{e} = \{e_0, e_1, \ldots, e_{m-1}\}$, possible emissions
- start/end state distribution $\mathbf{p} = (p_{z_0}, p_{z_1}, \ldots, p_{z_{n-1}})$ and $\mathbf{q} = (q_{z_0}, q_{z_1}, \ldots, q_{z_{n-1}})$
- transition matrix $\mathbf{a} = \begin{pmatrix} a_{z_0,z_0} & a_{z_0,z_1} & \cdots & a_{z_0,z_{n-1}} \\ a_{z_1,z_0} & a_{z_1,z_1} & \cdots & a_{z_1,z_{n-1}} \\ \vdots & \vdots & \cdots & \vdots \\ a_{z_{n-1},z_0} & a_{z_{n-1},z_1} & \cdots & a_{z_{n-1},z_{n-1}} \end{pmatrix}$
- emission matrix $\mathbf{b} = \begin{pmatrix} b_{z_0,e_0} & b_{z_0,e_1} & \cdots & b_{z_0,e_{m-1}} \\ b_{z_1,e_0} & b_{z_1,e_1} & \cdots & b_{z_1,e_{m-1}} \\ \vdots & \vdots & \cdots & \vdots \\ b_{z_{n-1},e_0} & b_{z_{n-1},e_1} & \cdots & b_{z_{n-1},e_{m-1}} \end{pmatrix}$

Definition

- n, m number of states and emissions; $\mathbf{z}$ states, $\mathbf{e}$ emissions
- start/end state distribution $\mathbf{p}$ and $\mathbf{q}$, $\mathbf{a}$, $\mathbf{b}$ transition/emission matrix

e_j	$\Pr[e_j \mid z_0]$
e_0	b_{0,e_0}
e_1	b_{0,e_1}
$\ldots$	$\ldots$
e_{m-1}	$b_{0,e_{m-1}}$



Constraints

- $0 \leq \theta_i \leq 1$,
- $\sum_{i=0}^{n-1} p_i = 1, \ \sum_{j=0}^{n-1} a_{ij} + q_i = 1, \ \sum_{j=0}^{m-1} b_{ij} = 1$
- **first order property**

$$\begin{aligned} \Pr[s_{t+1} = z_j | s_t = z_i, s_{t-1} = z_k, \ldots, s_0 = z_l, \boldsymbol{\theta}'] &= \Pr[s_{t+1} = z_j | s_t = z_i, \boldsymbol{\theta}'] \\ &= a_{ij} \end{aligned}$$

$$\begin{aligned} \Pr[o_t = e_j | s_t = z_i, s_{t-1} = z_k, \ldots, s_0 = z_l, \boldsymbol{\theta}'] &= \Pr[o_t = e_j | s_t = z_i, \boldsymbol{\theta}'] \\ &= b_{ij} \end{aligned}$$

- **independence of observations**

$$\Pr[o_0, o_1, \ldots, o_{T-1} | s_0, s_1, \ldots, s_{T-1} \boldsymbol{\theta}'] = \prod_{i=0}^{T-1} \Pr[o_i | s_t, \boldsymbol{\theta}']$$

Typical problems that HMMs could solve

- **Problem 1:** What is the likelihood that a given HMM $\boldsymbol{\theta}$ generated an observation $\mathbf{o}$, i.e., the likelihood $\Pr[\mathbf{o}|\boldsymbol{\theta}]$ that the HMM generates the observation o_0 at time 0, o_1 at time 1 up to o_{T-1} at time $T-1$ considering all possible state sequences?

- **Problem 2:** Given an observation and a HMM $\boldsymbol{\theta}$. Which is the most probable state sequence (path), i.e., the sequence that best describes observations?

- **Problem 3:** We are given several observations. How do we find the HMM that best describes these observations, i.e., the model parameters that maximise $\Pr[\mathbf{o}|\boldsymbol{\theta}]$?

How to train HMMs using gf?

- **setting/extracting model parameters:**
```
gf('send_command', 'new_hmm 3 6 1', );
[p,q,a,b]= gf('get_hmm');
gf('set_hmm', p,q,a,b);
gf('append_hmm',p,q,a,b)
```

- **baum welch training:** `gf('send_command','bw');`

- **viterbi training:** `gf('send_command','vit');`

- **classification:**
```
out = gf('hmm_classify');
out = gf('hmm_classify_example');
out = gf('one_class_hmm_classify');
out = gf('one_class_hmm_classify_example');
out = gf('one_class_linear_hmm_classify');
```

- **viterbi path** `[path, lik] = gf('best_path');`

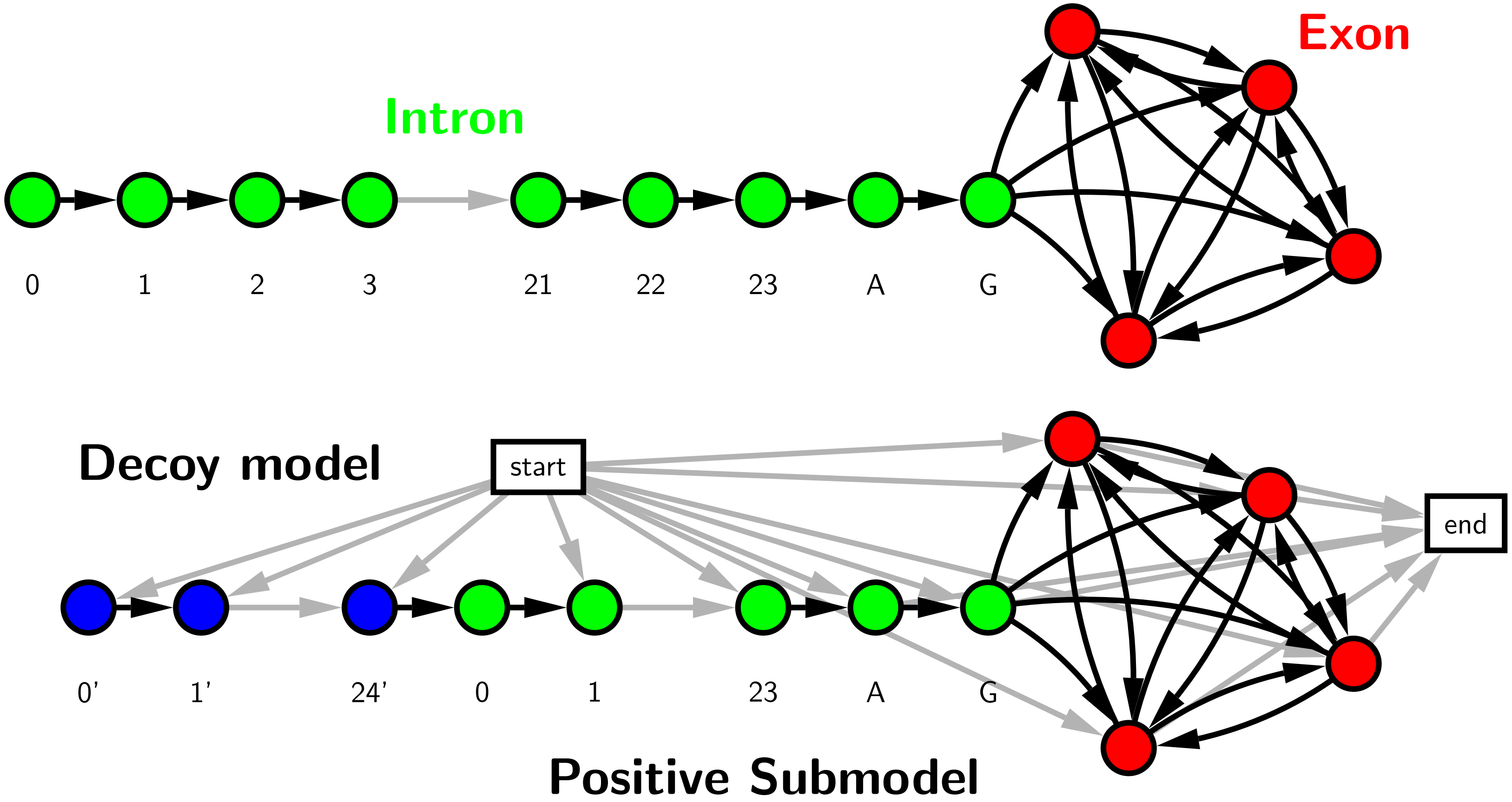
- **features + tricks**

```
gf('set_features','TRAIN',char(sequence+48)');
gf('send_command', 'convert TRAIN SIMPLE CHAR STRING CHAR');
gf('send_command', 'convert TRAIN STRING CHAR STRING WORD CUBE 1');
```

When are HMMs useful ?

- you know the structure of your problem very well
- you can efford to spend quite some time on designing an appropriate model
- you can restrict the structure of the model to make the problem well posed

Generative Models



(top) positive Acceptor model, (bottom) negative Acceptor model

Exercise

- extract bb_hmm_examples.tar.gz from neuro_toolbox/documentation/exercises

- **Task 1:** 3 cubes were drawn several times (unfair cubes). Determine when which cube was drawn and how the numbers are distributed. (file dice/dice.txt) !

- **Task 2:** Do classification with HMMs on the splice data set dna/acc_{train|val|test}.{pos|neg} !