Markov Models for Handwriting Recognition

— ICDAR 2007 Tutorial, Curitiba, Brazil—

Gernot A. Fink and Thomas Plötz

Robotics Research Institute, University of Dortmund, Germany

23. September 2007

- Motivation ... *Why use Markov Models?*
- Theoretical Concepts ... *Hidden Markov and n-gram Models*
- Practical Aspects ... *Configuration, Robustness, Efficiency, Search*
- Putting It All Together ... *How Things Work in Reality*
- Summary ... *and Conclusion*

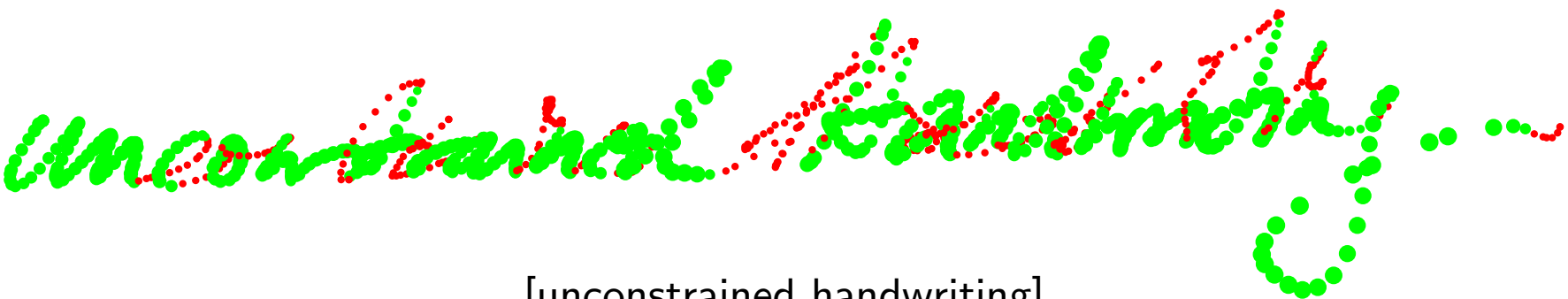
Why Handwriting Recognition?

Script: Symbolic form of archiving speech

- Certain different *principles* exist for writing down speech
- *Numerous* different writing systems developed over the centuries
- **Focus here:** Writing containing letters and numbers
  $\Rightarrow$ Especially: Roman script (incl. Arabic numbers)

Handwriting: In contrast to characters created *by machines*

- Most "natural" form of script in almost all cultures
- Typeface adapted for manual creation $\Rightarrow$ Cursive script
- "Printed letters" as imitation of machine printed characters
- **Frequently:** Free combination, i.e. mixing of both styles $\Downarrow$



[unconstrained handwriting]

Machine Printed vs. Handwritten Script

Final Programme

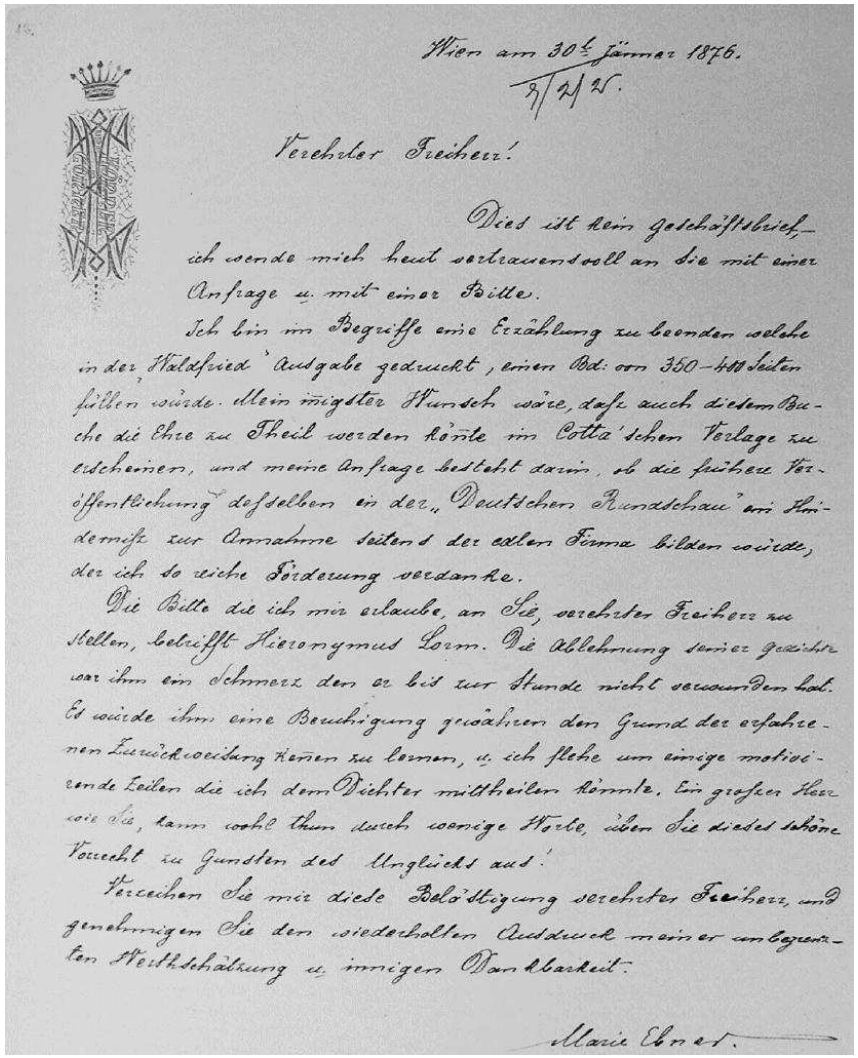
Ninth International Conference on Document Analysis and Recognition

ICDAR 2007

September 23 to 26, 2007
Curitiba, Brazil

http://www.icdar2007.org/

Co-Sponsors
TC10 (Graphics recognition) and TC11 (Reading System) of the International Association of Pattern Recognition (IAPR), and by the organizations as shown on the opposite page.



Wien am 30ᵗᵉ Jänner 1876.
2/2/25.

Verehrter Freiherr!

Dies ist kein Geschäftsbrief, ich wende mich heut vertrauensvoll an Sie mit einer Anfrage u. mit einer Bitte.

Ich bin im Begriffe eine Erzählung zu beenden welche in der Hallberg'schen Ausgabe gedruckt, einen Bd. von 350–400 Seiten füllen würde. Allein migster Wunsch wäre, daß auch diesem Buche die Ehre zu Theil werden könte im Cotta'schen Verlage zu erscheinen, und meine Anfrage besteht darin, ob die frühere Veröffentlichung desselben in der „Deutschen Rundschau" ein Hindernis zur Annahme seitens der edlen Firma bilden würde, der ich so reiche Förderung verdanke.

Die Bitte die ich mir erlaube, an Sie, verehrter Freiherr zu stellen, betrifft Hieronymus Lorm. Die Ablehnung seiner Gedichte war ihm ein Schmerz den er bis zur Stunde nicht verwunden hat. Es würde ihm eine Beruhigung gewähren den Grund der erfahrenen Zurückweisung kennen zu lernen, u. ich flehe um einige motivierende Zeilen die ich dem Dichter mittheilen könnte. Ein grossen Herr wie Sie, kann wohl thun durch wenige Worte, über sie dieses schöne Vorrecht zu Gunsten des Unglücks aus.

Verzeihen Sie mir diese Belästigung verehrter Freiherr, und genehmigen Sie den wiederholten Ausdruck meiner ungetheilten Werthschätzung u. innigen Dankbarkeit.

Marie Ebner.

Writing Systems – A Brief Overview

types	**picture story** (archiving a sequence of thoughts)	**logography** (archiving a single meaning / concept)			**phonography** (archiving the phonologic structure)		
variants		**pictograms**	**ideographic symbols**	**abstract–logographic symbols**	**segment script** (phone segments)	**syllabic script** (syllables)	**alphabetic script** (letters)
examples	indian picture stories (e.g. Kekinowin)	sumerian hieroglyphs	meaning depending on culture	arbitrary assignment of symbols to meanings	egyptian hieroglyphs (only skeleton of consonants)	cretan Linear B	fixed invent–ory of vowels & consonants

[the sick chief dies]

[foot, head, water]

Japanese characters (Kanji)

[fire, happiness]

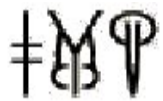
[Int. red cross org.]

[Int. red half moon org.]

§ & +

[paragraph, and, plus]

[m, n, y]



[pa–ma–ko (medicine)]

Brazil

[English using Roman script]

Βραζιλια

[Greek using greek letters]

complexity of single units

abstractness of single units

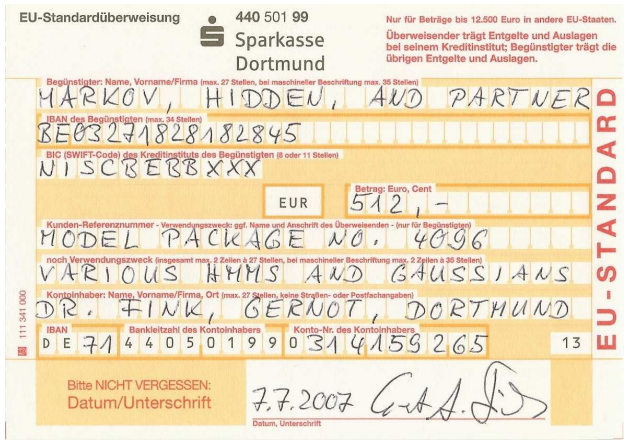
Applications: Off-line vs. On-line Processing

Main objective: Automated document processing (e.g. Analysis of addresses, forms; archiving)

Basic principle:

- Optical capture of completed typeface (via scanner, possibly camera)
- *Off-line* analysis of resulting "image data" (Standard: Preprocessing + Linearizing → stream of temp. changing features)

⇒ "Optical Character Recognition" (OCR)



Additionally: Human-Machine-Interaction (HMI)

Basic principle:

- Capture pen trajectory *during* writing (utilizing specialized sensors & pens)
- *On-line* analysis of movement data

Devices for Capturing Handwriting

On-line Recognition

Touch-pad:

- ✓ High resolution
- ✓ Robust recognition when using special writing style
  ⇒ Standard for PDAs etc.
- STOP (Very) expensive for large sizes (then also rather inflexible, e.g. for mobile use)

Camera:

- ✓ Very flexible
- STOP Low resolution (PAL . . .)
- STOP Occlusions need to be managed
  (e.g. by memory of regions)

Off-line Recognition

Scanner:

- ✓ High resolution ($\geq$ 300dpi)
- STOP Inappropriate for certain media (books, whiteboards etc.)

Camera:

- ✓ Very flexible
- STOP Low resolution (PAL . . .)

[Touch-pad:]

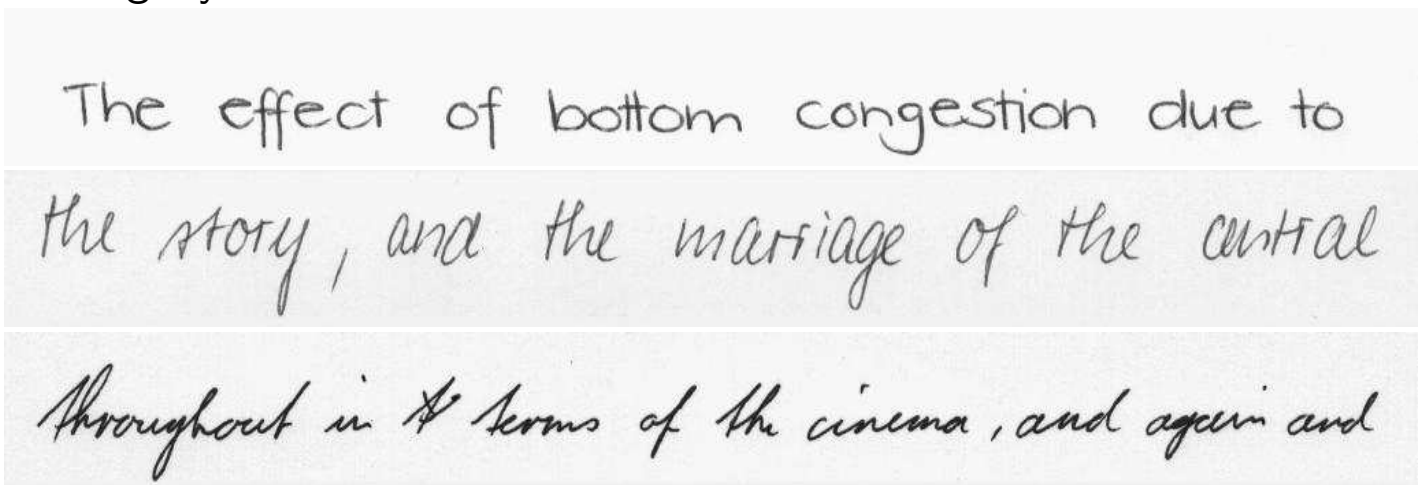
- ↯ Only "semi"-off-line
  (document needs to be written on touch pad – inappropriate for majority of media)
- ▶ cf. on-line recognition

Why is Handwriting Recognition Difficult?

- High variability even within particular characters
  - Machine printed characters:

Type face	Font family	Character size
(Times, Helvetica, `Courier`)	(regular, **bold**, *italic*)	(small, large)

  - Writing style



  - Line width and it's quality

  - Variation even for single writer!
- Segmentation is problematic – "Merging" of adjacent characters

Focus of this Tutorial

Input data:

- Handwriting data captured utilizing scanner, touch pad, or camera
- Restriction to Roman characters and Arabic numbers
  (no logographic writing systems covered *here*)
- *No* restriction w.r.t. writing style, size etc.
  $\Rightarrow$ Unconstrained handwriting!

Processing type:

- Focus: Offline-Processing
  $\Rightarrow$ Analysis of handwriting data *after* completion of capturing
- Also (a little . . .): Online-Techniques

Recognition approach: Stochastic modeling of handwriting

- Hidden Markov Models for segmentation free recognition
- Statistical n-gram models for lexical restrictions

Recognition Paradigm – "Traditional" OCR

Segmentation + Classification:

Original Image

Potential elementary segments, strokes, ...

Alternative segmentations

1.

2.

n.

✓ Segment-wise classification possible utilizing various standard techniques

↯ Segmentation is

- costly,
- heuristic, and
- needs to be optimized manually

↯ *Segmentation is especially problematic for unconstrained handwriting!*

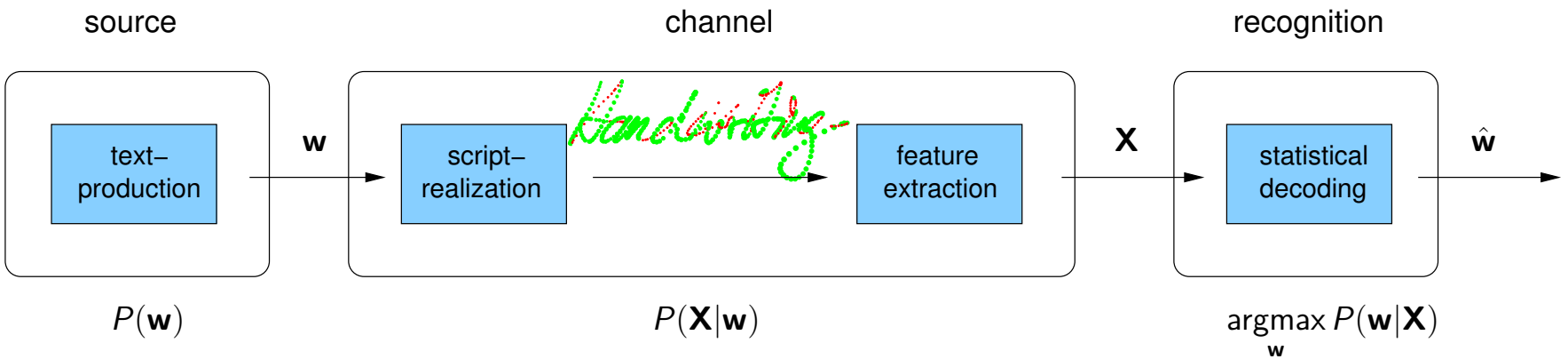
Overview

- Motivation . . . *Why use Markov Models?*
- Theoretical Concepts . . . *Hidden Markov and n-gram Models*
  - Hidden Markov Models . . . *Definition, Use Cases, Algorithms*
  - *n*-Gram Language Models . . . *Definition, Use Cases, Robust Estimation*
- Practical Aspects . . . *Configuration, Robustness, Efficiency, Search*
- Putting It All Together . . . *How Things Work in Reality*
- Summary . . . *and Conclusion*

Recognition Paradigm – I

Statistical recognition of (handwritten) script: Channel model

. . . similar to speech recognition



Wanted: Sequence of words/characters $\hat{\mathbf{w}}$, which is most probable for given signal/features $\mathbf{X}$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|\mathbf{X}) = \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(\mathbf{w})P(\mathbf{X}|\mathbf{w})}{P(\mathbf{X})} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w})P(\mathbf{X}|\mathbf{w})$$

Recognition Paradigm – II

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|\mathbf{X}) = \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(\mathbf{w})P(\mathbf{X}|\mathbf{w})}{P(\mathbf{X})} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w})P(\mathbf{X}|\mathbf{w})$$

Two aspects of modeling:

- Script (appearance) model: $P(\mathbf{X}|\mathbf{w})$ $\Rightarrow$ Representation of words/characters

  Hidden-Markov-Models

- Language model: $P(\mathbf{w})$ $\Rightarrow$ Restrictions for sequences of words/characters

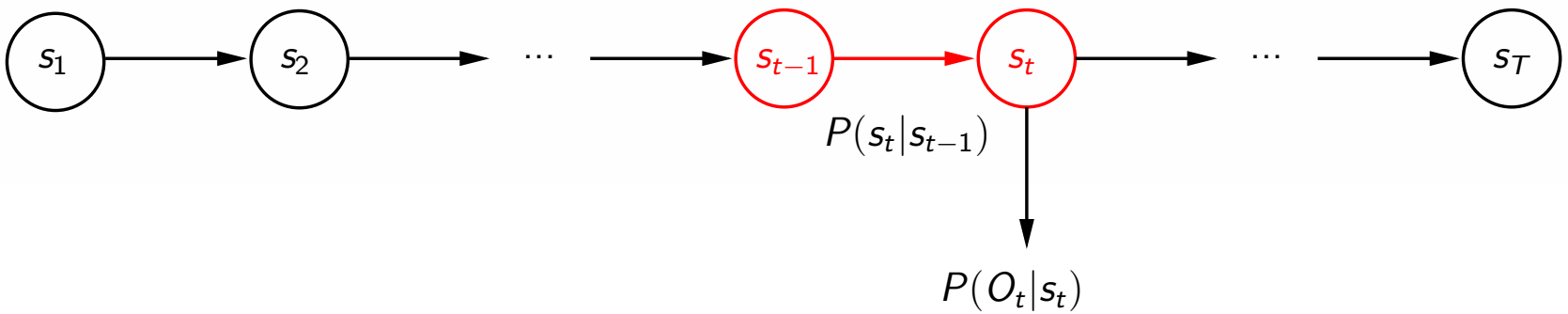
  Markov Chain Models / n-Gram-Models

Specialty: Script or trajectories of the pen (or features, respectively) interpreted as *temporal* data

✓ Segmentation now implicitly performed! $\Rightarrow$ "segmentation free" approach

STOP Script or pen movements, respectively, must be linearized!

Hidden Markov Models: Two-Stage Stochastic Processes



1. Stage: discrete stochastic process $\hat{=}$ series of random variables which take on values from a discrete set of states ($\approx$ finite automaton)

   stationary: Process independent of absolute time t

   causal: Distribution s_t only dependent on previous states

   simple: *particularly* dependent only from *immediate* predecessor state ($\hat{=}$ first order)

   $\Rightarrow P(s_t|s_1, s_2, \ldots s_{t-1}) = P(s_t|s_{t-1})$

2. Stage: Depending on current state s_t for every point in time additionally an emission O_t is generated

   $\Rightarrow P(O_t|O_1 \ldots O_{t-1}, s_1 \ldots s_t) = P(O_t|s_t)$

   Caution: Only emissions can be observed $\rightarrow$ **hidden**

Hidden-Markov-Models: Formal Definition

A Hidden-Markov-Model λ of *first order* is defined as:

- a finite set of states:

$$\{s|1 \leq s \leq N\}$$

- a matrix of state transition probabilities:

$$\mathbf{A} = \{a_{ij}|a_{ij} = P(s_t = j|s_{t-1} = i)\}$$

- a vector of start probabilities:

$$\boldsymbol{\pi} = \{\pi_i|\pi_i = P(s_1 = i)\}$$

- state specific emission probability distributions:

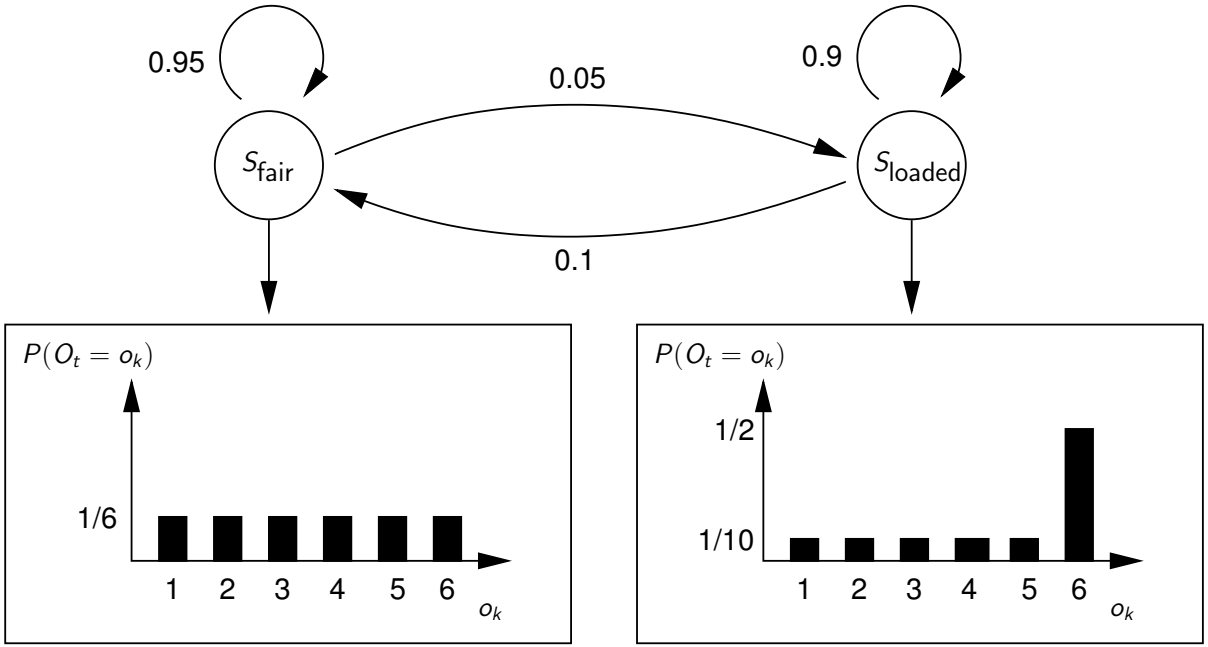
$$\mathbf{B} = \{b_{jk}|b_{jk} = P(O_t = o_k|s_t = j)\} \text{ (discrete case)}$$

or

$$\{b_j(O_t)|b_j(O_t) = p(O_t|s_t = j)\} \text{ (continuous case)}$$

Toy Example: The Occasionally Dishonest Casino – I

[idea from [Dur98]]



$O = 1, 1, 2, 6, 6, 6, 3, 5$

Background: Casino occasionally exchanging dice: fair $\Leftrightarrow$ loaded

$\Rightarrow$ Model with two states: S_{fair} and S_{loaded}

Exclusive observations: Results of the rolls

$\Rightarrow$ Underlying state-sequence remains hidden!

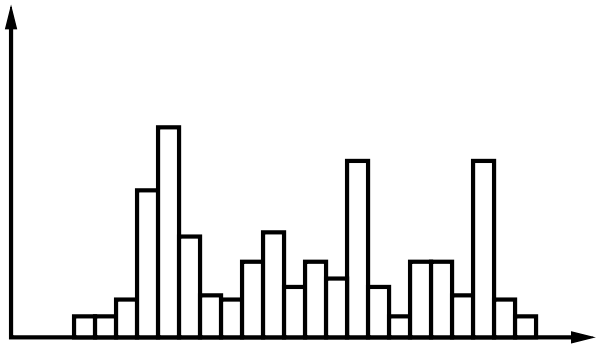
Question: Which die has been used, i.e. when is the casino cheating?

$\Rightarrow$ Probabilistic inference about internal state-sequence using stochastic model

Modeling of Emissions

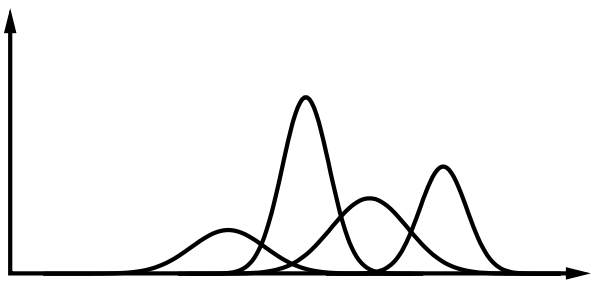
Discrete inventory of symbols: Very limited application fields

- ✓ Well suited for discrete data (e.g. DNA)
- ↯ Inappropriate for non-discrete data – use of vector quantizer required!



Continuous modeling: Standard for most pattern recognition applications processing sensory data

- ✓ Treatment of real-valued vector data (i.e. vast majority of "real-world" data)
- ✓ Defines distributions over $\mathbb{R}^n$



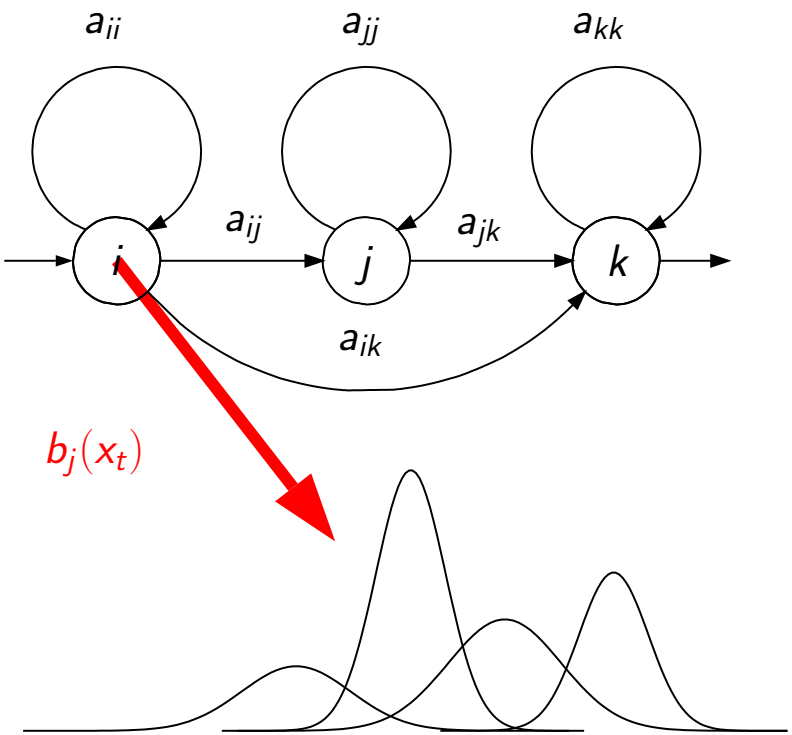
Problem: No general parametric description

Procedure: Approximation using mixture densities

$$p(\mathbf{x}) \;\hat{=}\; \sum_{k=1}^{\infty} c_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{C}_k)$$

$$\approx \sum_{k=1}^{M} c_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{C}_k)$$

Modeling of Emissions – II

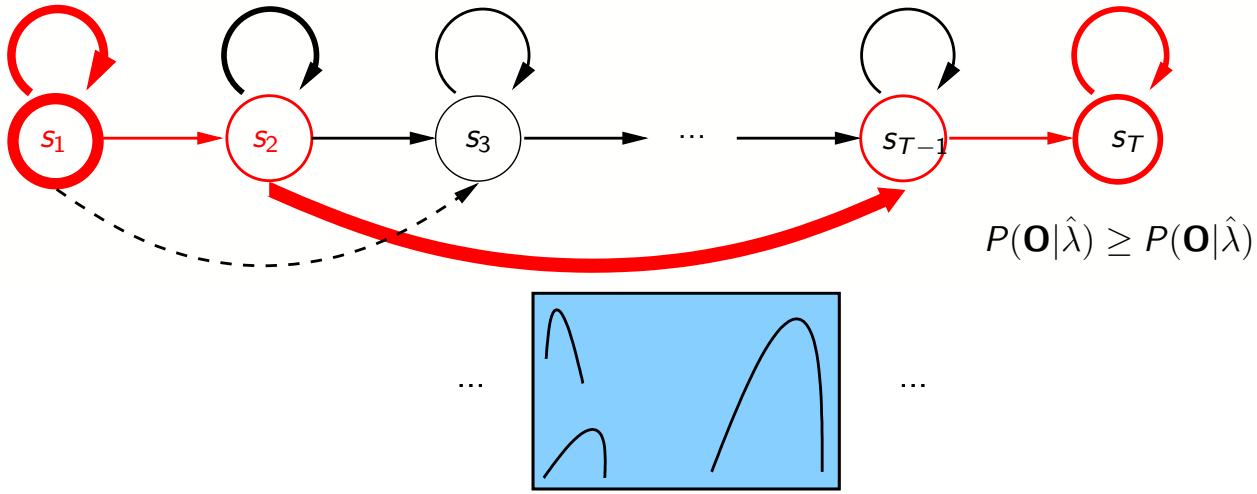


Mixture density modeling:

- Base Distribution?
  ⇒ Gaussian Normal densities
- Shape of Distributions
  (full / diagonal covariances)?
  ⇒ Depends on pre-processing of the data (e.g. redundancy reduction)
- Number of mixtures?
  ⇒ Clustering (. . . and heuristics)
- Estimation of mixtures?
  ⇒ e.g. Expectation-Maximization

[↗ Practice]

Usage Concepts for Hidden-Markov-Models



Assumption: Patterns observed are generated by stochastic models which are comparable *in principle*

Scoring: How well describes the model some pattern?
$\rightarrow$ Determination of the production probability $P(\mathbf{O}|\lambda)$

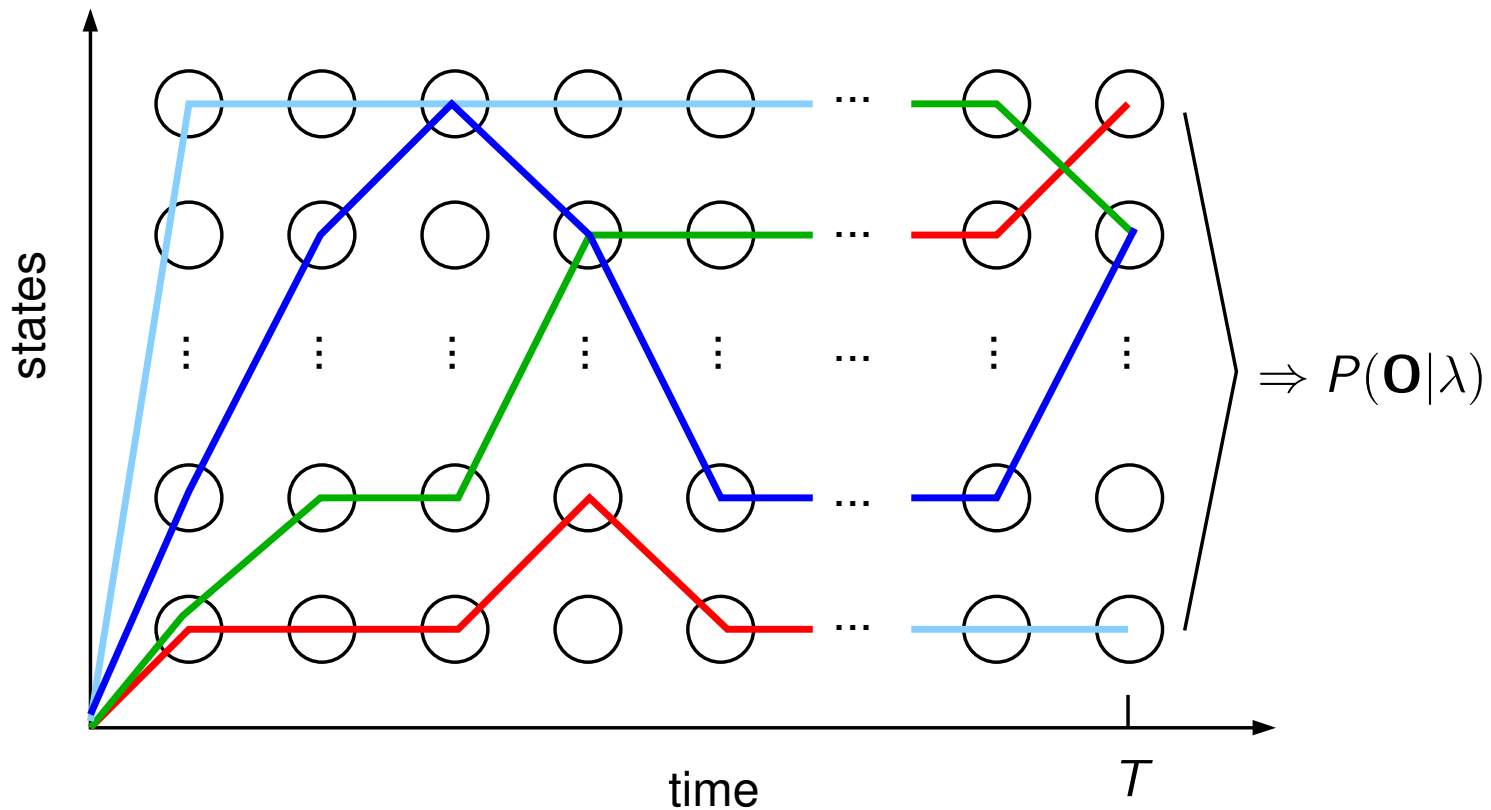
Decoding: What is the "internal structure" of the model? ($\hat{=}$ "Recognition")
$\rightarrow$ Determination of the optimal state sequence
$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}}\, P(\mathbf{O}, \mathbf{s}|\lambda)$

Training: How to determine the "optimal" model?
$\rightarrow$ Improvement of a given model λ with $P(\mathbf{O}|\hat{\lambda}) \geq P(\mathbf{O}|\lambda)$

The Production Probability

Wanted: Assessment of HMMs' quality for describing statistical properties of data

Widely used measure: *Production probability* $P(\mathbf{O}|\lambda)$ that observation sequence $\mathbf{O}$ was generated by model λ – along an arbitrary state sequence



The Production Probability: Naive Determination

1. Probability for generating observation sequence $O_1, O_2, \ldots O_T$ along corresponding state sequence $\mathbf{s} = s_1, s_2, \ldots s_T$ of same length:

$$P(\mathbf{O}|\mathbf{s}, \lambda) = \prod_{t=1}^{T} b_{s_t}(O_t)$$

2. Probability that a given model λ runs through arbitrary state sequence:

$$P(\mathbf{s}|\lambda) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1},s_t} = \prod_{t=1}^{T} a_{s_{t-1},s_t}$$

3. (1) + (2): Probability that λ generates $\mathbf{O}$ along certain state sequence $\mathbf{s}$:

$$P(\mathbf{O}, \mathbf{s}|\lambda) = P(\mathbf{O}|\mathbf{s}, \lambda)P(\mathbf{s}|\lambda) = \prod_{t=1}^{T} a_{s_{t-1},s_t} b_{s_t}(O_t)$$

4. Total $P(\mathbf{O}|\lambda)$: Summation over all possible state sequences of length T

$$P(\mathbf{O}|\lambda) = \sum_{\mathbf{s}} P(\mathbf{O}, \mathbf{s}|\lambda) = \sum_{\mathbf{s}} P(\mathbf{O}|\mathbf{s}, \lambda)P(\mathbf{s}|\lambda)$$

↯ Complexity: $O(TN^T)$

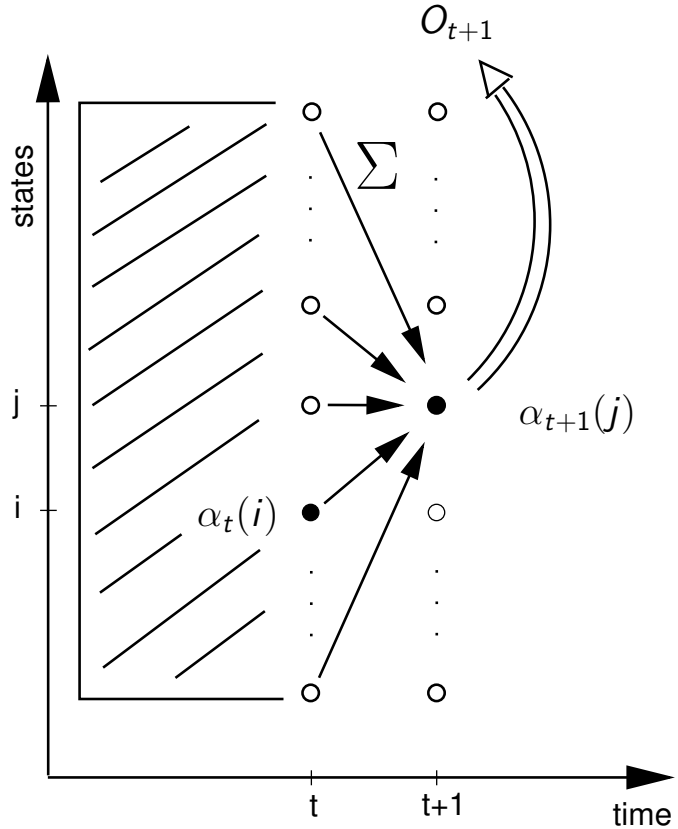
The Production Probability: The Forward-Algorithm

More efficient: Exploitation of the Markov-property, i.e. the "finite memory"
$\Rightarrow$ "Decisions" only dependent on immediate predecessor state

Let:
$\alpha_t(i) = P(O_1, O_2, \ldots O_t, s_t = i|\lambda)$
(*forward variable*)

1. $\alpha_1(i) := \pi_i b_i(O_1)$
2. $\alpha_{t+1}(j) := \left\{ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right\} b_j(O_{t+1})$
3. $P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

✓ Complexity: $O(TN^2)$!
(vs. $O(TN^T)$ from naive determination)



Later: Backward-Algorithm [↗ Training]

The "optimal" Production Probability

Total production probability: Consider *all* paths through model

- ✓ Mathematically exact determination of $P(\mathbf{O}|\lambda)$
- STOP Specialization of partial models within total model cannot be judged

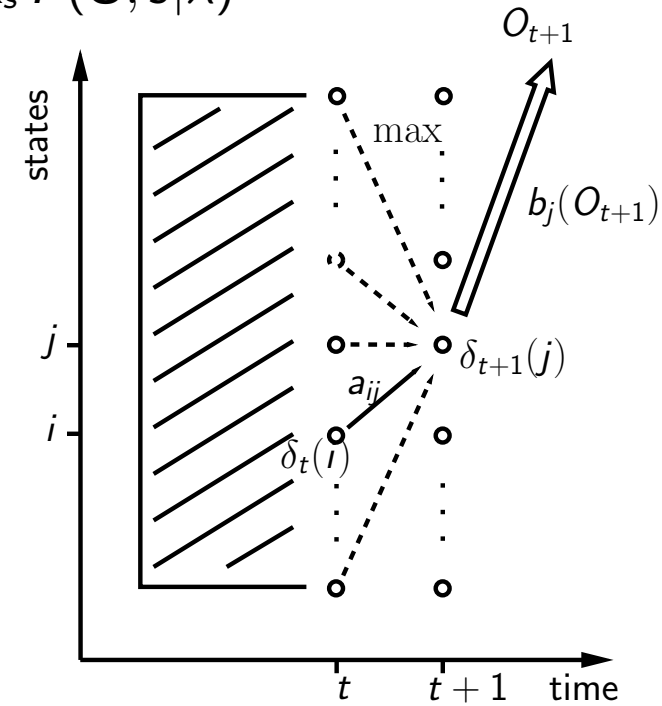
Modification: Consider only respective optimal possibility to generate $\mathbf{O}$, given λ

- ✓ Discrimination between λ_1 (satisfying on average) / λ_2 (specialized)

Optimal probability: $P^*(\mathbf{O}|\lambda) = P(\mathbf{O}, s^*|\lambda) = \max_s P(\mathbf{O}, s|\lambda)$

$$\delta_t(i) = \max_{s_1,\ldots s_{t-1}} P(O_1,\ldots O_t, s_1,\ldots s_{t-1}, s_t = i|\lambda)$$

1. $\delta_1(i) = \pi_i b_i(O_1)$
2. $\forall t,\ t = 1\ldots T-1$:
   $\delta_{t+1}(j) = \max_i \{\delta_t(i) a_{ij}\}\, b_j(O_{t+1})$
3. $P^*(\mathbf{O}|\lambda) = P(\mathbf{O}, \mathbf{s}^*|\lambda) = \max_i \delta_T(i)$



Decoding

Problem: Global production probability $P(\mathbf{O}|\lambda)$ not sufficient for analysis if individual states are associated to meaningful segments of data

$\Rightarrow$ (Probabilistic) Determination of optimal state sequence $\mathbf{s}^*$ necessary

Maximization of posterior probability:

$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}}\, P(\mathbf{s}|\mathbf{O}, \lambda)$$

Bayes' rule:

$$P(\mathbf{s}|\mathbf{O}, \lambda) = \frac{P(\mathbf{O}, \mathbf{s}|\lambda)}{P(\mathbf{O}|\lambda)}$$

$P(\mathbf{O}|\lambda)$ irrelevant (constant for fixed $\mathbf{O}$ and given λ), thus:

$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}}\, P(\mathbf{s}|\mathbf{O}, \lambda) = \underset{\mathbf{s}}{\operatorname{argmax}}\, P(\mathbf{O}, \mathbf{s}|\lambda)$$

Determination of $\mathbf{s}^*$: Brute-Force [↗ Optimal Production Probability] or more efficiently: *Viterbi-Algorithm*

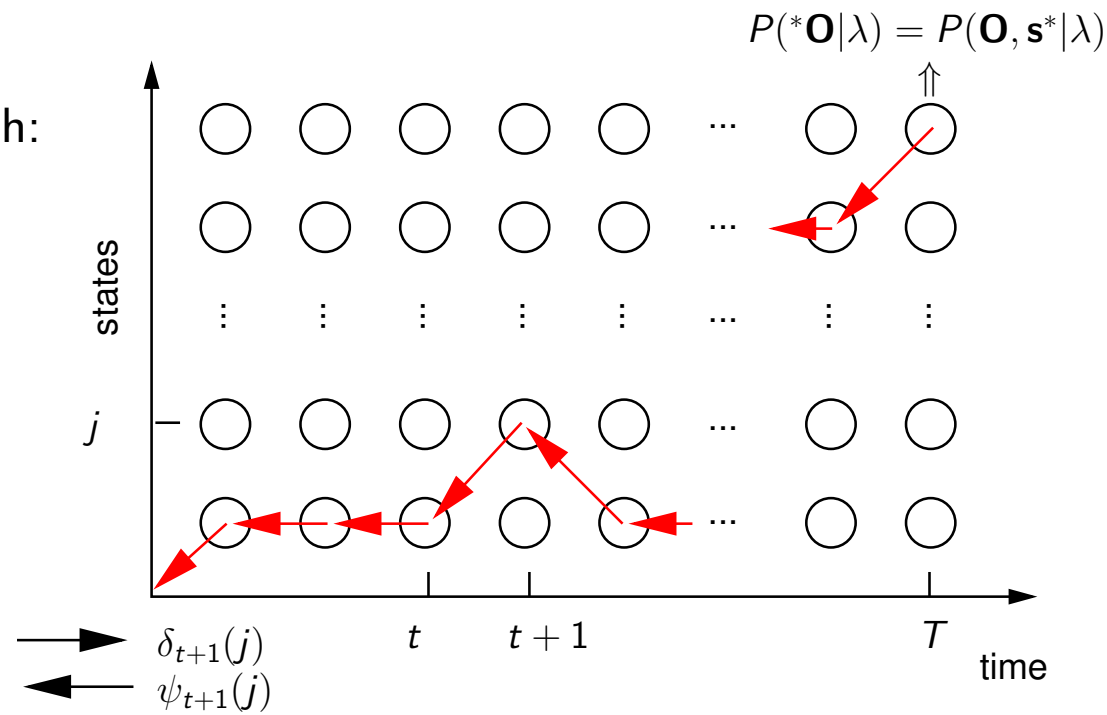
The Viterbi Algorithm

. . . inductive procedure for efficient determination of $\mathbf{s}^*$ exploiting Markov property

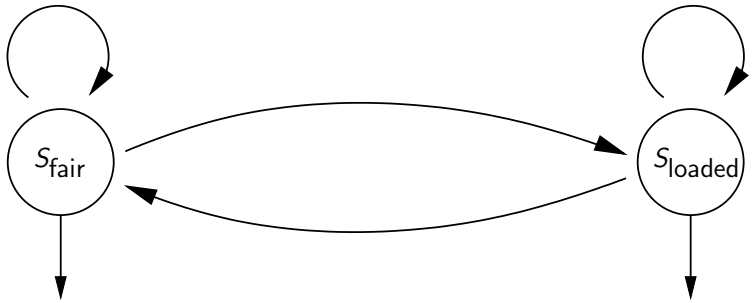
Let: $\delta_t(i) = \max_{s_1, s_2, \ldots s_{t-1}} P(O_1, O_2, \ldots O_t, s_t = i|\lambda)$

1. $\delta_1(i) := \pi_i b_i(O_1)$ $\qquad \psi_1(i) := 0$
2. $\delta_{t+1}(j) := \max_i(\delta_t(i)a_{ij})b_j(O_{t+1})$ $\qquad \psi_{t+1}(j) := \operatorname{argmax}_i \ldots$
3. $P^*(\mathbf{O}|\lambda) = P(\mathbf{O}, \mathbf{s}^*|\lambda) = \max_i \delta_T(i)$
   $s_T^* := \operatorname{argmax}_j \delta_T(j)$
4. Back-tracking of optimal path:
   $s_t^* = \psi_{t+1}(s_{t+1}^*)$

- ✓ Implicit *segmentation*
- ✓ Linear complexity in time
- STOP Quadratic complexity w.r.t. #states



Toy Example: The Occasionally Dishonest Casino – II



Parameters of the given HMM λ:

- Start probabilities: $\boldsymbol{\pi} = (1/2 \quad 1/2)^T$
- Transition probabilities: $\mathbf{A} = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix}$
- Emission probabilities: $\mathbf{B} = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{pmatrix}$
- Observation sequence: $\mathbf{O} = O_1, O_2, \ldots, O_T = 1, 1, 2, 6, 6, 6, 3, 5$

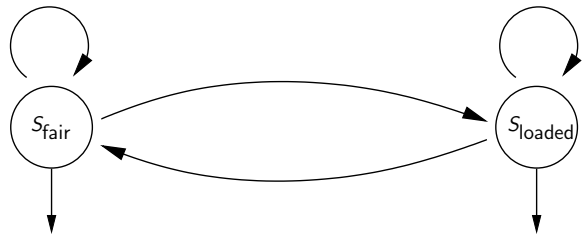
Wanted: Internal state-sequence for segmentation into fair use and cheating
$\Rightarrow$ Viterbi-Algorithm

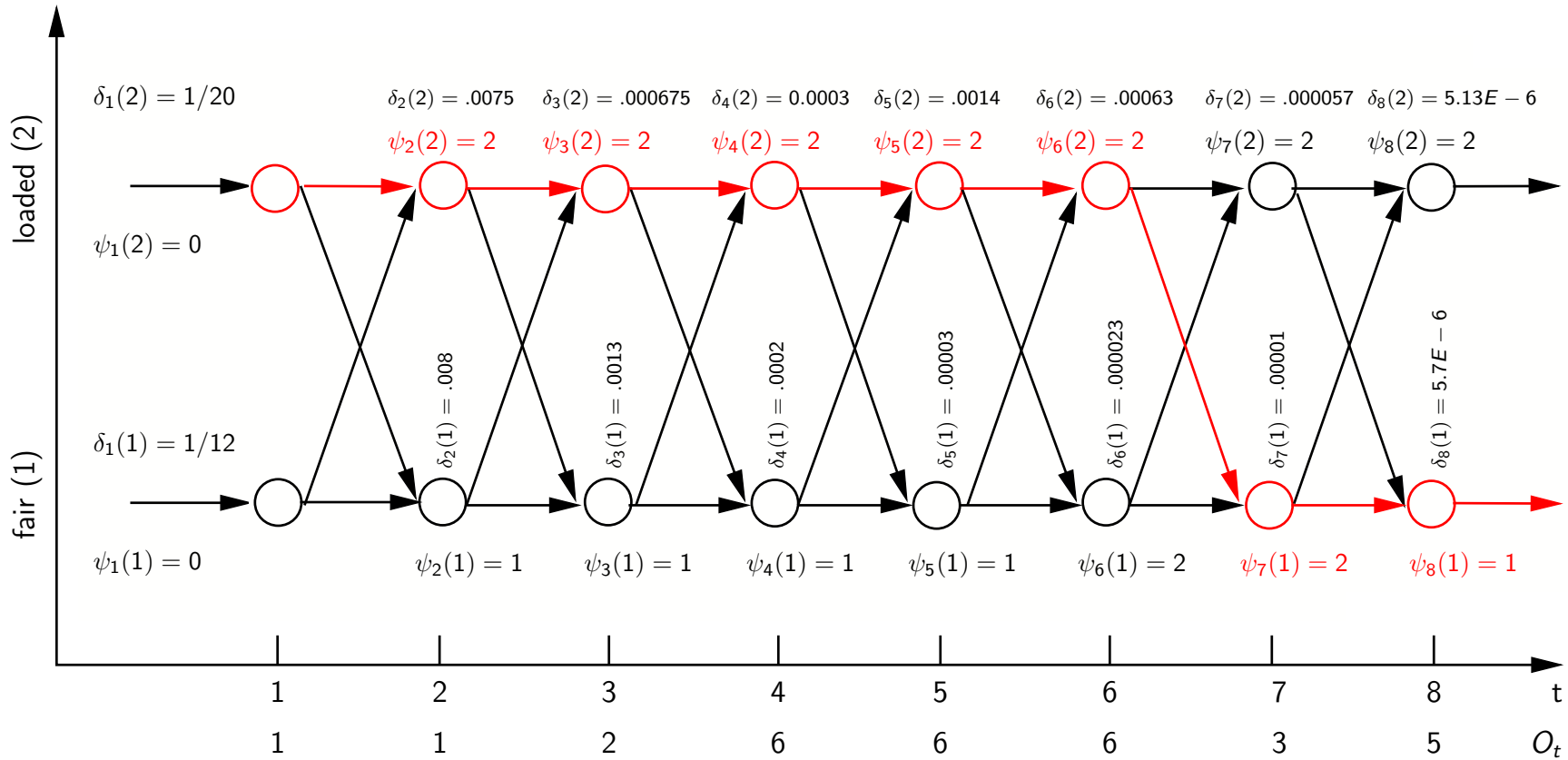
Toy Example: The Occasionally Dishonest Casino – III

$$\pi_i = 1/2, \quad \mathbf{A} = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{pmatrix}$$

$$\mathbf{O} = 1, 1, 2, 6, 6, 6, 3, 5$$





Parameter Estimation – Fundamentals

Goal: Derive optimal (for some purpose) statistical model from sample data

Problem: No suitable analytical method / algorithm known

"Work-Around": Iteratively improve existing model λ
$\Rightarrow$ Optimized model $\hat{\lambda}$ better suited for given sample data

General procedure: Parameters of λ subject to growth transformation such that

$$P(\ldots|\hat{\lambda}) \geq P(\ldots|\lambda)$$

1. "Observe" model's actions during generation of an observation sequence
2. Original parameters are replaced by relative frequencies of respective events

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from } i \text{ to } j}{\text{expected number of transitions out of state } i}$$

$$\hat{b}_i(o_k) = \frac{\text{expected number of outputs of } o_k \text{ in state } i}{\text{total number of outputs in state } i}$$

- Only probabilistic inference of events possible!
- (Posterior) state probability required!

The Posterior State Probability

Goal: Efficiently compute $P(S_t = i|\mathbf{O}, \lambda)$ for model assessment

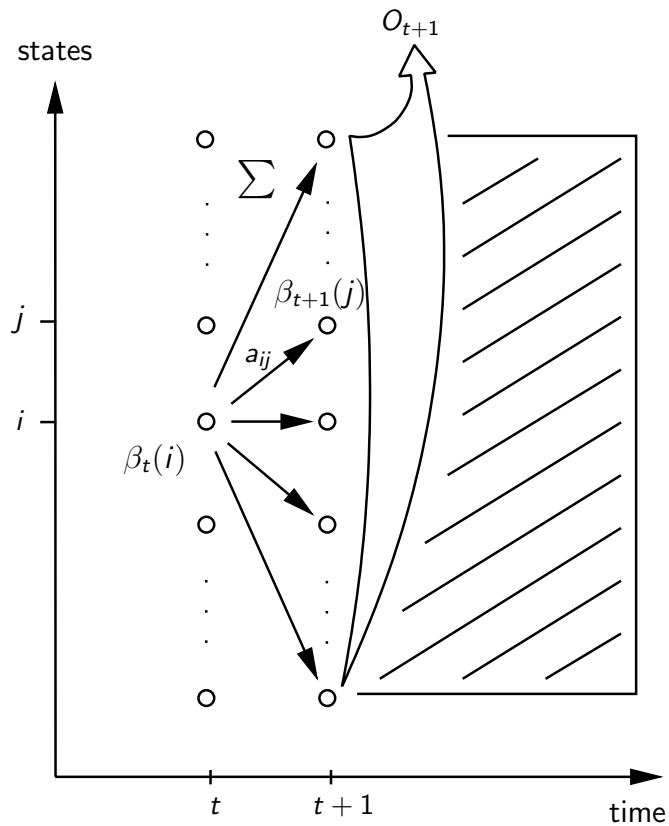
Procedure: Exploit limited memory for

- History – forward-probability $\alpha_t(i)$ [↗ forward-algorithm], and
- Rest of partial observation sequence – *backward-probability* $\beta_t(i)$

Backward-Algorithm:
Let

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots O_T|s_t = i, \lambda)$$

1. $\beta_T(i) := 1$
2. For all times t, $t = T-1 \dots 1$:
   $\beta_t(i) := \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
3. $P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_1(i)$



The Forward-Backward Algorithm

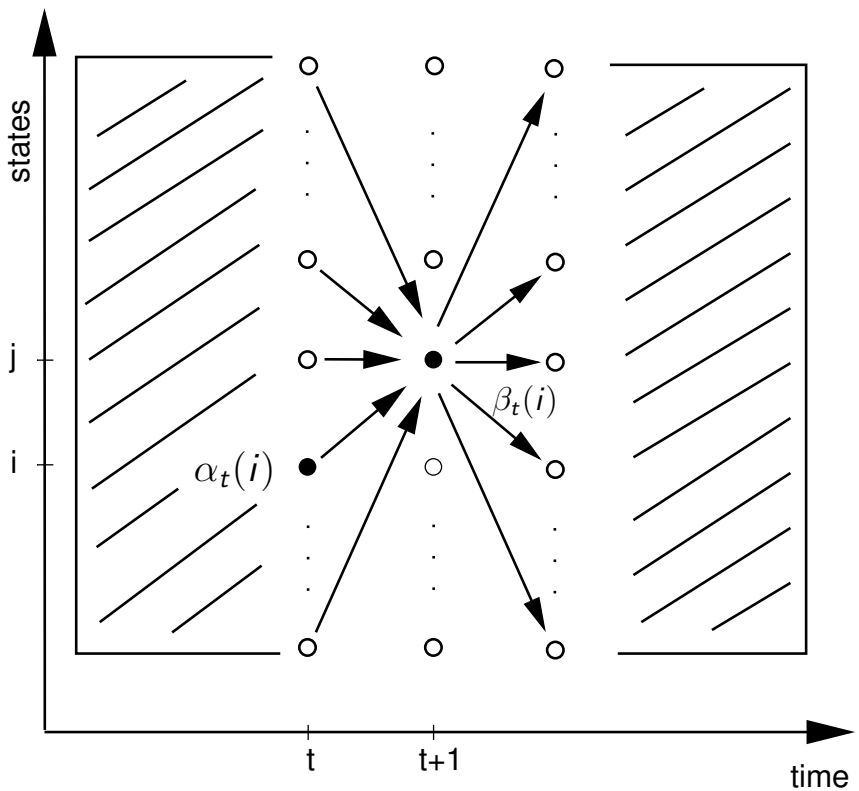
. . . for efficient determination of posterior state probability

$$P(S_t = i|\mathbf{O},\lambda) = \frac{P(S_t = i,\mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \quad [\nearrow \text{forward-algorithm}]$$

$$P(S_t = i,\mathbf{O}|\lambda) = P(O_1, O_2, \ldots O_t, S_t = i|\lambda)P(O_{t+1}, O_{t+2}, \ldots O_T|S_t = i,\lambda)$$

$$= \alpha_t(i)\beta_t(i)$$

$$\Rightarrow \quad \gamma_t(i) = P(S_t = i|\mathbf{O},\lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathbf{O}|\lambda)}$$



Parameter Training using the Baum-Welch-Algorithm

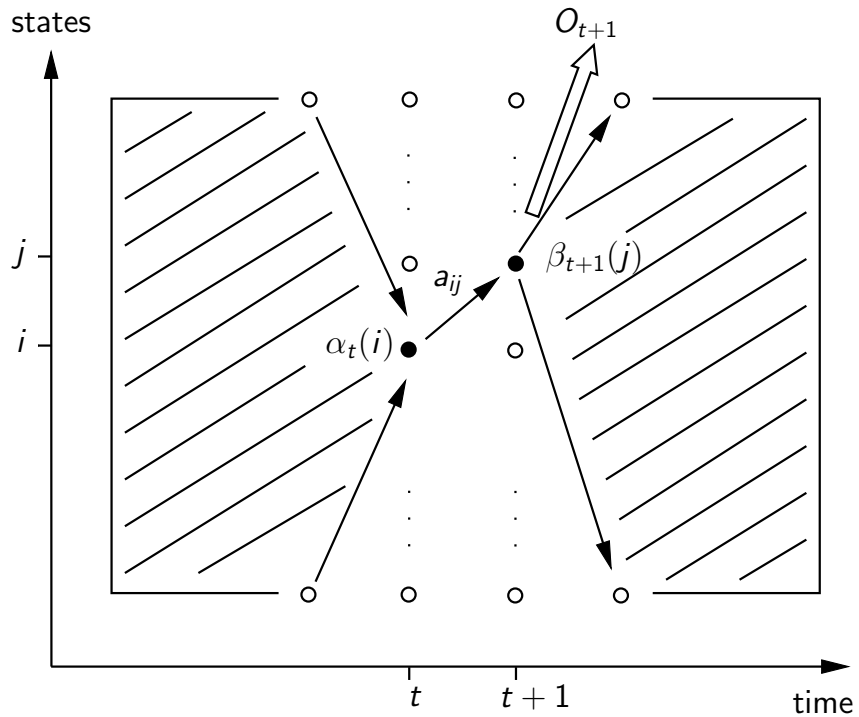
Background: Variant of *Expectation Maximization (EM)*-Algorithm
(parameter estimation for stochastic models incl. hidden random variables)

Optimization criterion: Total production probability $P(\mathbf{O}|\lambda)$, thus

$$P(\mathbf{O}|\hat{\lambda}) \geq P(\mathbf{O}|\lambda)$$

Definitions: (of quantities based on forward- and backward variables)
$\Rightarrow$ Allow (statistical) inferences about internal processes of λ when generating $\mathbf{O}$

$$\begin{aligned}
\gamma_t(i,j) &= P(S_t = i, S_{t+1} = j|\mathbf{O},\lambda) \\
&= \frac{P(S_t = i, S_{t+1} = j, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \\
&= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(\mathbf{O}|\lambda)} \\
\gamma_t(i) &= P(S_t = i|\mathbf{O},\lambda) \\
&= \sum_{j=1}^{N} P(S_t = i, S_{t+1} = j|\mathbf{O},\lambda) \\
&= \sum_{j=1}^{N} \gamma_t(i,j)
\end{aligned}$$



The Baum-Welch-Algorithm

Let

$$\gamma_t(i) = P(S_t = i|\mathbf{O},\lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathbf{O}|\lambda)}$$

$$\gamma_t(i,j) = P(S_t = i, S_{t+1} = j|\mathbf{O},\lambda) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(\mathbf{O}|\lambda)}$$

$$\xi_t(j,k) = P(S_t = j, M_t = k|\mathbf{O},\lambda) = \frac{\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\, c_{jk}\, g_{jk}(O_t)\, \beta_t(j)}{P(\mathbf{O}|\lambda)}$$

1. Choose a suitable initial model $\lambda = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ with initial estimates (π_i, a_{ij}, c_{jk} for mixtures $g_{jk}(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{jk}, \mathbf{C}_{jk})$ for pdf. $b_{jk}(\mathbf{x}) = \sum_k c_{jk}\, g_{jk}(\mathbf{x})$.)

2. Compute updated estimates $\hat{\lambda} = (\hat{\boldsymbol{\pi}}, \hat{\mathbf{A}}, \hat{\mathbf{B}})$ for all model parameters:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{\pi}_i = \gamma_1(i)$$

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k)\, \mathbf{x}_t}{\sum_{t=1}^{T} \xi_t(j,k)} \qquad \hat{\mathbf{C}}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k)\, \mathbf{x}_t \mathbf{x}_t^T}{\sum_{t=1}^{T} \xi_t(j,k)} - \hat{\boldsymbol{\mu}}_{jk} \hat{\boldsymbol{\mu}}_{jk}^T$$

3. **if** $P(\mathbf{O}|\hat{\lambda})$ was considerably improved by the updated model $\hat{\lambda}$ w.r.t. λ
   let $\lambda \leftarrow \hat{\lambda}$ and continue with step 2
   otherwise Stop!

Multiple Observation Sequences

In general: Sample sets used for parameter training are subdivided into individual segments – so-called *turns*

So far: Turns were considered individual observation sequences

Goal: Estimate model parameters also on a *set* of isolated sequences

Procedure: Accumulate across all observation sequences considered statistics gathered for the updating of the parameters

Example:

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T}\xi_t^l(j,k)\,\mathbf{x}_t}{\sum_{l=1}^{L}\sum_{t=1}^{T}\xi_t^l(j,k)}$$

Hidden Markov Models: Summary

Pros and Cons:

- ✓ Two-stage stochastic process for analysis of highly variant patterns
  (allows for probabilistic inference about internal state sequence – i.e. recognition)
- ✓ Efficient algorithms for training and evaluation, resp., exist
  (Forward-Backward, Viterbi-decoding, Baum-Welch)
- ✓ Can "easily" be combined with statistical language model
  (channel model: integration of [↗ Markov chain models])
- ↯ Considerable amounts of training data necessary
  ("There's no data like more data!" [Mer88])

Variants and Extensions (not covered *here*):

- Hybrid models increased robustness
  (often combination with neural networks)
- Techniques for fast and robust adaptation, i.e. specialization, exist
  (Maximum A-posteriori adaptation, Maximum Likelihood Linear Regression)

Overview

- Motivation . . . *Why use Markov Models?*
- Theoretical Concepts . . . *Hidden Markov and n-gram Models*
  - Hidden Markov Models . . . *Definition, Use Cases, Algorithms*
  - *n*-Gram Language Models . . . *Definition, Use Cases, Robust Estimation*
- Practical Aspects . . . *Configuration, Robustness, Efficiency, Search*
- Putting It All Together . . . *How Things Work in Reality*
- Summary . . . *and Conclusion*

n-Gram Models: Introduction

Goal of statistical language modeling: Define a probability distribution over a set of symbol (= word) sequences

Origin of the name *Language Model*: Methods closely related to

- Statistical modeling of texts
- Imposing restrictions on word hypothesis sequences (especially in automatic speech recognition)

Powerful concept: Use of Markov chain models

Alternative method: Stochastic grammars

- ↯ Rules can not be learned
- ↯ Complicated, costly parameter training

$\Rightarrow$ Not widely used!

n-Gram Models: Example

Examples for statistical models fitting on slides extremely problematic! *Beware!*

Given an empirically defined language fragment:

```
I don't mind if you go
I don't mind if you take it slow
I don't mind if you say yes or no
I don't mind at all
```

[From the lyrics of the *Great Song of Indifference* by Bob Geldof]

Questions:

- How is the phrase ‘‘`I don't mind`’’ most likely continued?
- Which sentence is more plausible, to be expected, or rather “strange”?

  ‘‘`I take it if you don't mind`’’ or

  ‘‘`if you take it I don't mind`’’

n-Gram Models: Definition

Goal: Calculate $P(\mathbf{w})$ for given word sequence $\mathbf{w} = w_1, w_2, \ldots, w_k$

Basis: n-Gram model = Markov chain model of order $n-1$

Method: Factorization of $P(\mathbf{w})$ applying Bayes' rule according to

$$P(\mathbf{w}) = P(w_1)P(w_2|w_1)\ldots P(w_T|w_1,\ldots,w_{T-1}) = \prod_{t=1}^{k} P(w_t|w_1,\ldots,w_{t-1})$$

Problem: Context dependency increases arbitrarily with length of symbol sequence
$\Rightarrow$ Limit length of the "history"

$$P(\mathbf{w}) \approx \prod_{t=1}^{T} P(\underbrace{\quad w_t \quad | w_{t-n+1},\ldots,w_{t-1}}_{n \text{ symbols}})$$

Result: Predicted word w_t and *history* form an n-tuple $\Rightarrow$ n-gram ($\hat{=}$ *event*)

$\Rightarrow$ **n-gram models** (typically: $n = 2 \Rightarrow$ bi-gram, $n = 3 \Rightarrow$ tri-gram)

n-Gram Models: Use Cases

Basic assumption similar to HMM case:

1. Reproduce statistical properties of observed data
2. Derive inferences from the model

Problems to be solved:

Evaluation: *How well does the model represent certain data?*

Basis: Probability of a symbol sequence assigned by the model

Model Creation: *How to create a good model?*

- No hidden state variables
  $\Rightarrow$ No iteratively optimizing techniques required
- Parameters can principally be computed directly
  (by simple counting)

More sophisticated methods necessary in practice! [↗ parameter estimation]

n-Gram Models: Notation

Focus on expressions for computing conditional probabilities

Distinction between predicted word and history important

- Arbitrary individual n-gram: $\mathbf{y}z = y_1, y_2, \ldots y_{n-1}z$
  (predicted word: z, history: $\mathbf{y}$)
- General conditional n-gram probability: $P(z|\mathbf{y})$
  ($P(z|y)$ or $P(z|xy)$ for bi- and tri-gram models)
- Absolute frequency of an n-gram: $c(\mathbf{y}z)$
- Some derived properties of n-gram contexts:

  Count of all n-grams with history $\mathbf{y}$: $c(\mathbf{y}\cdot)$

  Number of n-grams occurring k times in context $\mathbf{y}$: $d_k(\mathbf{y}\cdot)$

n-Gram Models: Evaluation

Basic Principle: Determine descriptive power on *unknown* data

Quality Measure: *Perplexity* $\mathcal{P}$

$$\mathcal{P}(\mathbf{w}) = \frac{1}{\sqrt[|\mathbf{w}|]{P(\mathbf{w})}} = \frac{1}{\sqrt[T]{P(w_1, w_2, \ldots, w_T)}} = P(w_1, w_2, \ldots, w_T)^{-\frac{1}{T}}$$

- Reciprocal of geometric mean of symbol probabilities
- Derived from (cross) entropy definition of a (formal) language

$$H(p|q) = -\sum_i \underbrace{p_i}_{\text{data}} \underbrace{\log_2 q_i}_{\text{model}} \longrightarrow -\underbrace{\sum_t \frac{1}{T}}_{\text{empirical data}} \underbrace{\log_2 P(w_t|\ldots)}_{\text{model}} = -\frac{1}{T} \log_2 \prod_t P(w_t|\ldots)$$

$$\mathcal{P}(\mathbf{w}) = 2^{H(\mathbf{w}|P(\cdot|\ldots))} = 2^{-\frac{1}{T}\log_2 \prod_t P(w_t|\ldots)} = P(w_1, w_2, \ldots, w_T)^{-\frac{1}{T}}$$

Question: *How can perplexity be interpreted?*

n-Gram Models: Interpretation of Perplexity

Intuitive interpretation of perplexity:

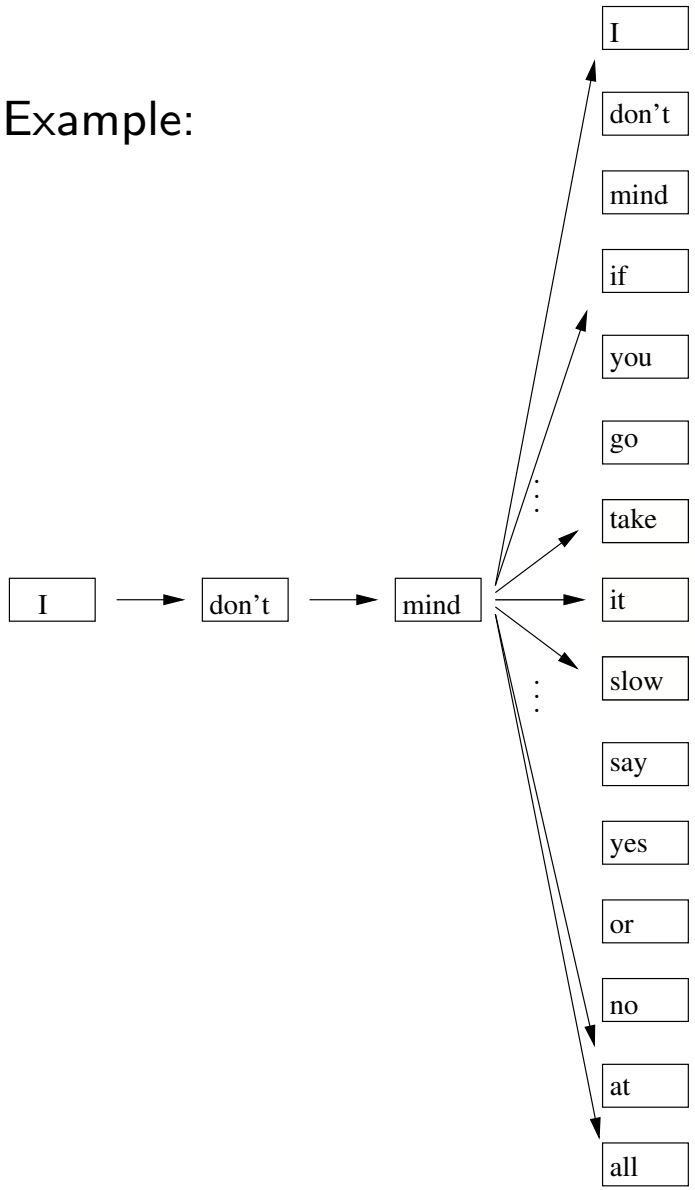
- Assume: Text $w_1, w_2, ...w_t, ...w_T$ was produced *statistically* by information source from finite vocabulary V
- Problem: How can that generation be "predicted" as exactly as possible?

  Successful: Only very few symbols likely to continue a sequence

  Unsuccessful: Many symbols have to be taken into account

- Worst case situation: No information
  - $\Rightarrow$ No prediction possible
  - $\Rightarrow$ All symbols equally likely: $P(w_t|...) = \frac{1}{|V|}$

Example:



n-Gram Models: Interpretation of Perplexity II

- Worst case situation: All symbols equally likely

  $\Rightarrow$ Prediction according to *uniform* distribution $P(w_t|...) = \frac{1}{|V|}$

  - Perplexity of texts generated:

$$\mathcal{P}(\mathbf{w}) = \left\{ \left(\frac{1}{|V|} \right)^T \right\}^{-\frac{1}{T}} = |V|$$

Note: Perplexity equals vocabulary size in absence of restrictions

- In *any* other case: perplexity $\rho < |V|$

  Reason: Entropy (and perplexity) is maximum for uniform distribution!

- Relating this to an "uninformed" source with uniform distribution:
  Prediction is as hard as source with $|V'| = \rho$

  Interpretation: Perplexity gives size of "virtual" lexicon for statistical source!

n-Gram Models: Parameter Estimation

Example:

Naive Method:

- Determine number of occurrences
  - $c(w_1, w_2, \ldots w_n)$ for all n-grams and
  - $c(w_1, w_2, \ldots w_{n-1})$ for $n-1$-grams

$c(\texttt{you say}) = 1$

$c(\texttt{you}) = 3$

- Calculate conditional probabilities

$$P(w_n|w_1, w_2, \ldots w_{n-1}) = \frac{c(w_1, w_2, \ldots w_n)}{c(w_1, \ldots w_{n-1})}$$

$$P(\texttt{say}|\texttt{you}) = \frac{c(\texttt{you say})}{c(\texttt{say})} = \frac{1}{3}$$

Problem: Many n-grams are **not** observed

$\Rightarrow$ *"Unseen events"*

- $c(w_1 \ldots w_n) = 0 \Rightarrow P(w_n|\ldots) = 0$

$c(\texttt{you don't}) = 0$

- $P(\ldots w_1 \cdots w_n \ldots) = 0$!

$P(\texttt{I take it if you don't mind}) = 0$

n-Gram Models: Parameter Estimation II

Parameter estimation in practice

Problem:

- Not *some* but *most* n-gram counts will be **zero**!
- It must be assumed that this is only due to insufficient training data!

$\Rightarrow$ estimate *useful* $P(z|\mathbf{y})$ for $\mathbf{y}z$ with $c(\mathbf{y}z) = 0$

Question: *What estimates are "useful"?*

- small probabilities!, smaller than *seen* events? $\longrightarrow$ mostly not guaranteed!
- specific probabilities, not uniform for all unseen events

Solution:

1. Modify n-gram counts and gather "probability mass" for *unseen events*

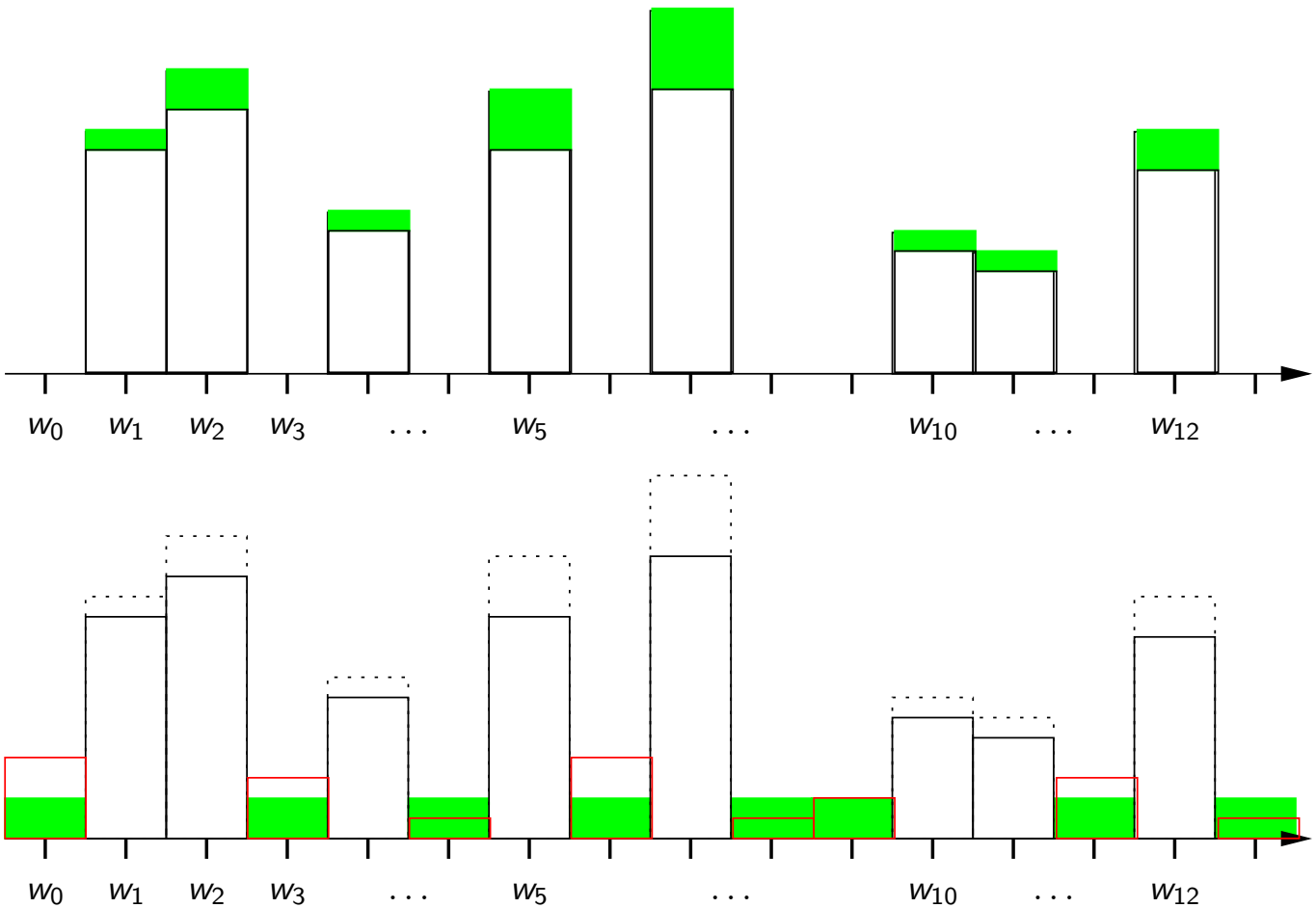
   Note: Keep modification reasonably small for seen events!

2. Redistribute *zero-probability* to *unseen events* according to a more general distribution ($\hat{=}$ *smoothing* of empirical distribution)

   Question: *What distribution is suitable for events we know nothing about?*

n-Gram Models: Parameter Estimation III

Frequency distribution (counts) $\longrightarrow$ Discounting (gathering probability mass)



Zero probability $\longrightarrow$ Incorporate more general distribution

n-Gram Models: Discounting

Gathering of Probability Mass

Calculate modified frequency distribution $f^*(z|\mathbf{y})$ for seen n-grams $\mathbf{y}z$:

$$f^*(z|\mathbf{y}) = \frac{c^*(\mathbf{y}z)}{c(\mathbf{y})} = \frac{c(\mathbf{y}z) - \beta(\mathbf{y}z)}{c(\mathbf{y}\cdot)}$$

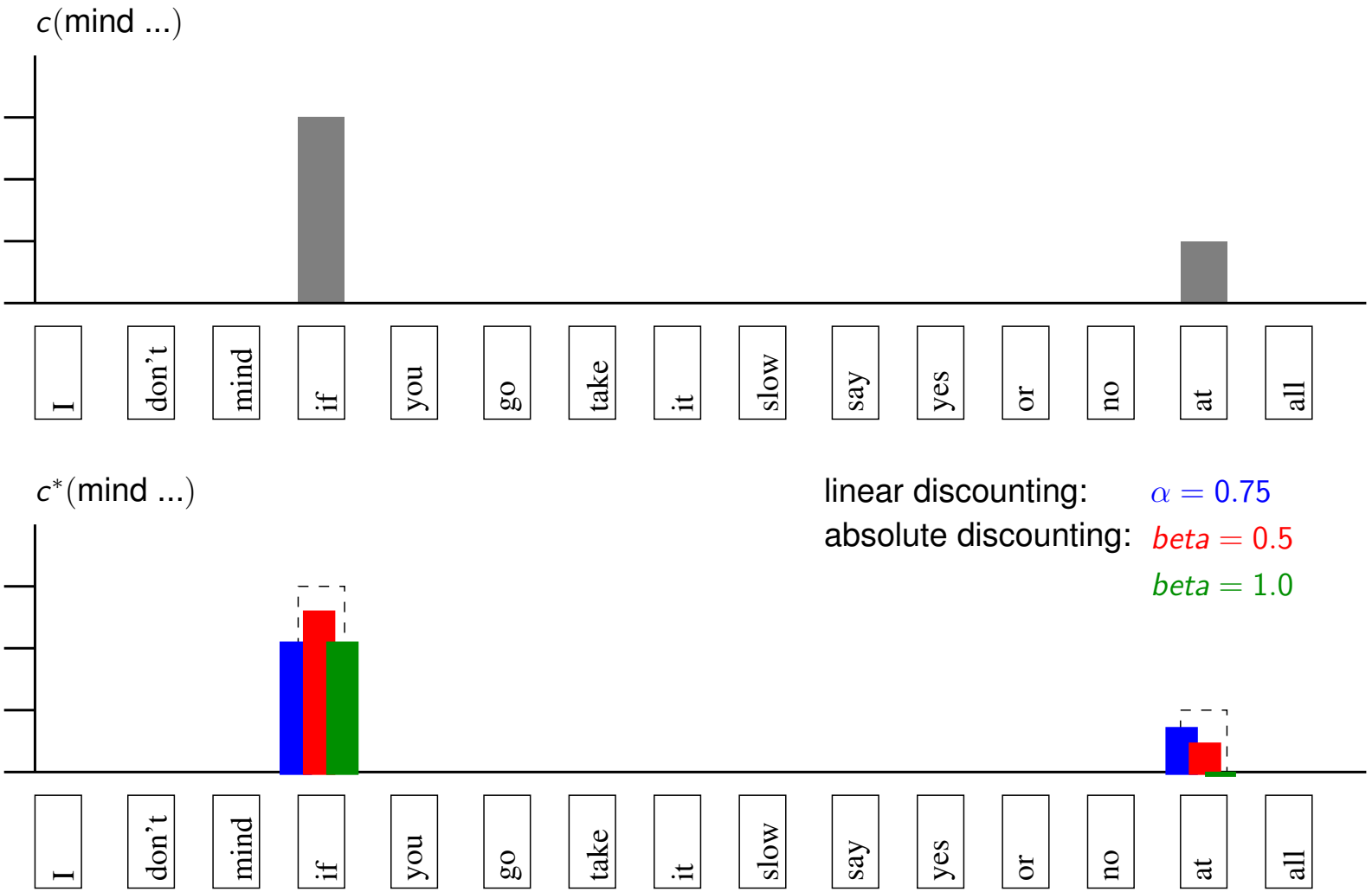
Zero-probability $\lambda(\mathbf{y})$ for history $\mathbf{y}$: Sum of "collected" counts

$$\lambda(\mathbf{y}) = \frac{\sum_{z:c(\mathbf{y}z)>0} \beta(\mathbf{y}z)}{c(\mathbf{y}\cdot)}$$

Choices for discounting factor $\beta()$:

- proportional to n-gram count: $\beta(\mathbf{y}z) = \alpha c(\mathbf{y}z)$ $\Rightarrow$ *linear* discounting
- as some constant $0 < \beta \leq 1$ $\Rightarrow$ *absolute* discounting

n-Gram Models: Discounting - Example



Note: Discounting is performed *individually* for *all* contexts **y**!

n-Gram Models: Smoothing

Redistribution of Probability Mass

Basic methods for incorporating more general distributions:

Interpolation: Linear combination of (modified) n-gram distribution and (one or more) general distributions

Backing off: Use more general distribution for unseen events only

Remaining problem: *What is a more general distribution?*

Widely used solution: Corresponding n-1-gram model $P(z|\hat{\mathbf{y}})$ associated with n-gram model $P(z|\mathbf{y})$

- Generalization $\hat{=}$ shortening the context/history

  $\mathbf{y} = y_1, y_2, \ldots y_{n-1} \longrightarrow \hat{\mathbf{y}} = y_2, \ldots y_{n-1}$

- More general distribution obtained:

  $q(z|\mathbf{y}) = q(z|y_1, y_2, \ldots y_{n-1}) \leftarrow P(z|y_2, \ldots y_{n-1}) = P(z|\hat{\mathbf{y}})$

  (i.e. bi-gram for tri-gram model, uni-gram for bi-gram model ...)

n-Gram Language Models: Interpolation

Principle Idea (not considering modified distribution $f^*(\cdot|\cdot)$):

$$P(z|\mathbf{y}) = (1-\alpha)\, f(z|\mathbf{y}) + \alpha\, q(z|\mathbf{y}) \quad 0 \leq \alpha \leq 1$$

Problem: Interpolation weight α needs to be optimized (e.g. on held-out data)

Simplified view with linear discounting: $f^*(z|\mathbf{y}) = (1-\alpha) f(z|\mathbf{y})$

Estimates obtained:

$$P(z|\mathbf{y}) = \begin{cases} f^*(z|\mathbf{y}) + \lambda(\mathbf{y}) q(z|\mathbf{y}) & c^*(\mathbf{y}z) > 0 \\ \lambda(\mathbf{y}) q(z|\mathbf{y}) & c^*(\mathbf{y}z) = 0 \end{cases}$$

Properties:

- Assumes that estimates *always* benefit from smoothing

$\Rightarrow$ All estimates modified

✓ Helpful, if original estimates unreliable

↯ Estimates from large sample counts should be "trusted"

n-Gram Language Models: Backing Off

Basic principle: Back off to general distribution for unseen events

$$P(z|\mathbf{y}) = \begin{cases} f^*(z|\mathbf{y}) & c^*(\mathbf{y}z) > 0 \\ \lambda(\mathbf{y})\, K_{\mathbf{y}}\, q(z|\mathbf{y}) & c^*(\mathbf{y}z) = 0 \end{cases}$$

Normalization factor $K_{\mathbf{y}}$ ensures that: $\sum_z P(z|\mathbf{y}) = 1$

$$K_{\mathbf{y}} = \frac{1}{\sum\limits_{\mathbf{y}z\,:\,c^*(\mathbf{y}z)=0} q(\mathbf{y}z)}$$

Note:

- General distribution used for unseen events only
- Estimates with substantial support unmodified, assumed reliable

n-Gram Language Models: Generalized Smoothing

Observation: With standard solution for $q(z|\mathbf{y})$ more general distribution is again n-gram model $\Rightarrow$ principle can be applied recursively

Example for backing off and tri-gram model:

$$P(z|xy) = \begin{cases} f^*(z|xy) & c^*(xyz) > 0 \\ \lambda(xy)\, K_{xy} \begin{cases} f^*(z|y) & c^*(xyz) = 0 \quad \wedge \quad c^*(yz) > 0 \\ \lambda(y)\, K_y \begin{cases} f^*(z) & c^*(yz) = 0 \quad \wedge \quad c^*(z) > 0 \\ \lambda(\cdot)\, K_\cdot \frac{1}{|V|} & c^*(z) = 0 \end{cases} \end{cases} \end{cases}$$

Note: Combination of absolute discounting and backing off creates powerful n-gram models for a wide range of applications (cf. [Che99]).

n-Gram Language Models: Representation and Storage

Requirement: n-gram models need to define specific probabilities for *all* potential events (i.e. $|V|^n$ scores!)

Observation: Only probabilities of seen events are predefined
(in case of discounting: including context-dependent zero-probability)

$\Rightarrow$ Remaining probabilities can be computed

Consequence: Store only probabilities of seen events in memory

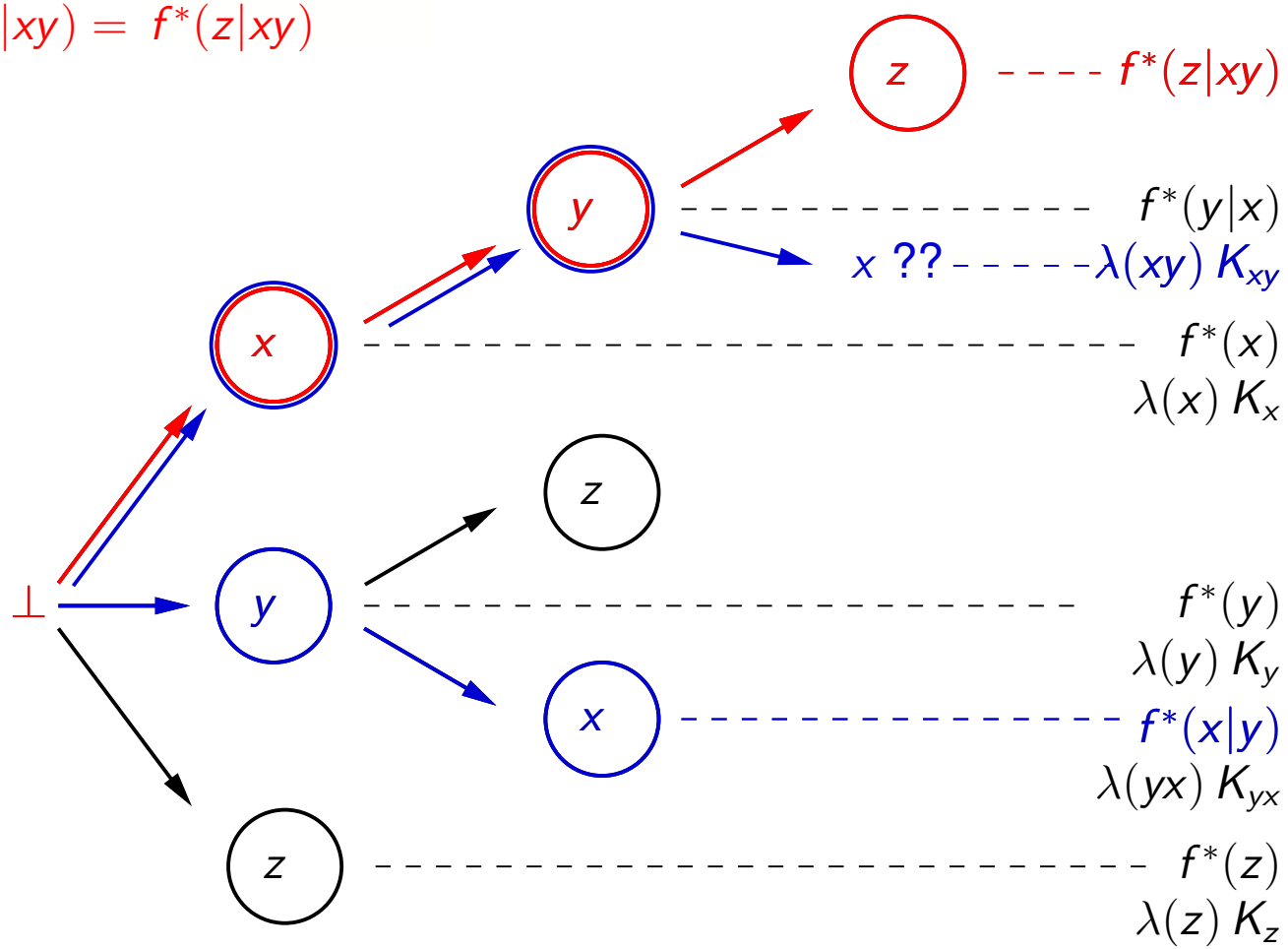
$\Rightarrow$ *Huge* savings as *most* events are not observed!

Further Observation: n-grams always come in hierarchies
(for representing the respective general distributions)

$\Rightarrow$ Store parameters in prefix-tree for easy access

n-Gram Language Models: Representation and Storage II

$P(z|xy) = f^*(z|xy)$



$P(x|xy) = \lambda(xy)\, K_{xy}\, f^*(x|y)$

n-Gram Language Models: Summary

Pros and Cons:

- ✓ Parameters can be estimated automatically from training texts
- ✓ Models "capture" syntactic, semantic, and pragmatic restrictions of the language fragment considered
- ✓ Can "easily" be combined with statistical recognition systems (e.g. HMMs)
- ↯ Considerable amounts of training data necessary
- ↯ Manageable only for small n (i.e. rather short contexts)

Variants and Extensions of the basic model:

- Category-based language models
  (useful for representing paradigmatic regularities)
- Models for describing long-distance context restrictions
  (useful for languages with discontinuous constitutes, e.g. German)
- Topic-based (i.e. context dependent) models
  (useful, if one global model is too general)

Overview

- Motivation *. . . Why use Markov Models?*
- Theoretical Concepts *. . . Hidden Markov and n-gram Models*
- **Practical Aspects** *. . . Configuration, Robustness, Efficiency, Search*
  - Computations with Probabilities
  - Configurations of HMMs
  - Robust Parameter Estimation
  - Efficient Model Evaluation
  - Integrated Search
- Putting It All Together *. . . How Things Work in Reality*
- Summary *. . . and Conclusion*

Computations with Probabilities: Problem

Example:

- Consider a 10-dimensional feature representation
  - → Mixture components will be 10-D Gaussians (distributing total probability mass of 1!)
  - → $p(\mathbf{x}) \approx 10^{-5}$ on average
- Further consider a text fragment of 100 frames length
  (will in general be only a few characters long)

$$\rightarrow P(\mathbf{x_1}, ..., \mathbf{x_{100}}|O) \approx \prod_{i=1}^{100} p(\mathbf{x_i}) \approx (10^{-5})^{100}$$

Note: Extremely coarse approximations, neglecting

- Alternative paths,
- Transition probabilities,
- Other mixtures and weights.

⇒ *Quantities can't be represented on todays computers!*

Note: Even double precision floating point formats not sufficient
(ANSI/IEEE 854: $[1.8 \cdot 10^{308} \leq z \leq 4.9 \cdot 10^{-324}]$)

Probabilities: Logarithmic Representation

Solution: Represent probabilities in *negative logarithmic* domain

- Logarithmic $\rightarrow$ compression of dynamic range
- Negative $\rightarrow$ transformed quantities can be interpreted as *additive* costs

Method: Transform quantities according to: $\tilde{p} = -\log_b p$

Any type of logarithm can be used!

Most common: $\ln x = \log_e x$ and e^x pair of operations (`log`/`exp`)

Effect: Probabilities $\in [0.0 \ldots 1.0] \longrightarrow$ costs $\in [0.0 \ldots +\infty]$

(Note: Densities > 1.0 possible but rare $\rightarrow <0.0$)

a_{ij}		$-\log a_{ij}$
$b_j(\mathbf{x})$		$-\log b_j(\mathbf{x})$
$1.0 \ldots 10^{-1000}$		$0.0 \ldots 1000$
product	$\rightarrow$	sum
maximum		minimum
sum		*. . . a bit more complicated*

Probabilities: Logarithmic Representation II

Question: Could product/maximum ($\hat{=}$ sum/minimum) be sufficient for HMMs?
Mostly, but not quite for everything!

- ✓ Viterbi algorithm: only maximization of partial path scores, multiplication of score components

  $$\delta_{t+1}(j) = \max_i \{\delta_t(i) a_{ij}\} b_j(O_{t+1}) \quad \rightarrow \quad \min_i \left\{\tilde{\delta}_t(i) + \tilde{a}_{ij}\right\} + \tilde{b}_j(O_{t+1})$$

- ✓ Mixture density evaluation: Weighted sum can be approximated by maximum

  $$b_j(\mathbf{x}) = \sum_{k=1}^{M_j} c_{jk}\, g_{jk}(\mathbf{x}) \approx \max_{k=1}^{M_j} c_{jk}\, g_{jk}(\mathbf{x}) \quad \rightarrow \quad \min \ldots$$

- ↯ Baum-Welch training: Alternative paths *must* be considered $\Rightarrow$ summation of total probabilities required

Problem: How can sums of probabilities be computed in the logarithmic representations?

Probabilities: Logarithmic Representation III

Summation in the logarithmic domain

Naive Method:

$$\tilde{p}_1 +_{\log} \tilde{p}_2 = -\ln\left(e^{-\tilde{p}_1} + e^{-\tilde{p}_2}\right)$$

- ↯ Advantage of logarithmic representation is lost!

Summation in the logarithmic domain: Kingsbury-Rayner formula

$$\begin{aligned}
\tilde{p}_1 +_{\log} \tilde{p}_2 &= -\ln(p_1 + p_2) = -\ln(p_1(1 + \tfrac{p_2}{p_1})) = \\
&= -\left\{\ln p_1 + \ln(1 + e^{\ln p_2 - \ln p_1})\right\} \\
&= \tilde{p}_1 - \ln(1 + e^{-(\tilde{p}_2 - \tilde{p}_1)})
\end{aligned}$$

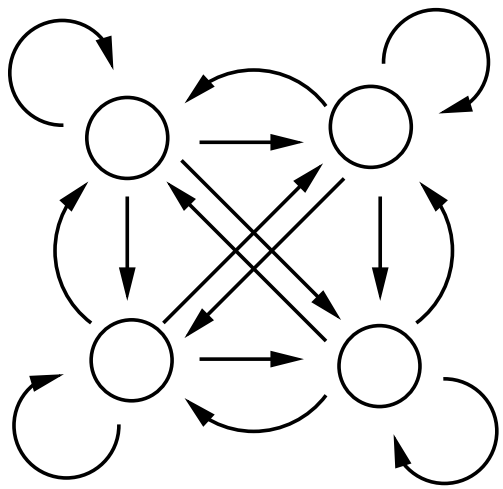
- ✓ Only *score differences* need to be represented in linear domain

Configuration of HMMs: Topologies

Generally: Transitions between arbitrary states possible within HMMs ...
potentially with arbitrarily low probability

Topology of an HMM: Explicit representation of allowed transitions
(drawn as edges between nodes/states)

Any transition possible
$\Rightarrow$ *ergodic* HMM



Observation: Fully connected HMM does usually not make sense for describing chronologically organized data

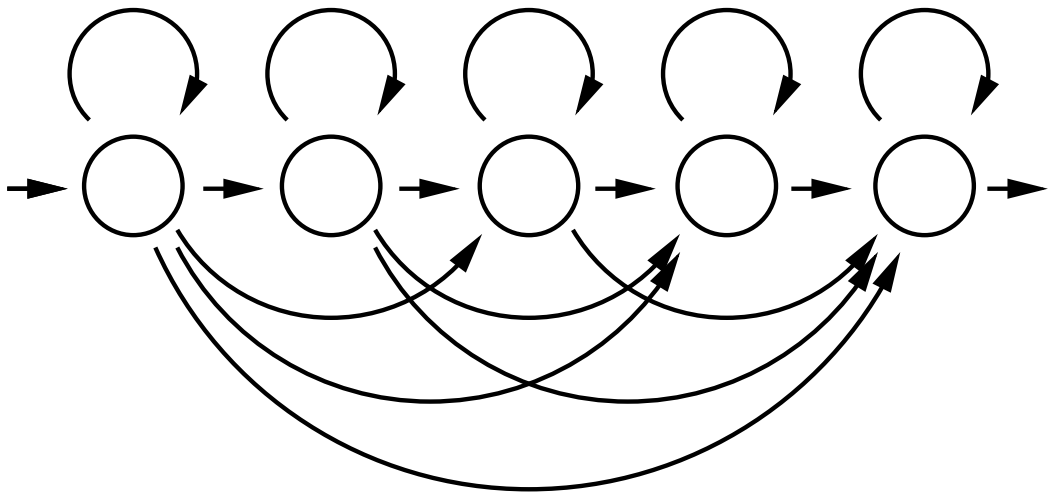
↯ "backward" transitions would allow arbitrary repetitions within the data

Configuration of HMMs: Topologies II

Idea: Restrict potential transition to *relevant* ones!

... by omitting irrelevant edges / setting respective transition probabilities to "hard" zeros (i.e. never modified!)

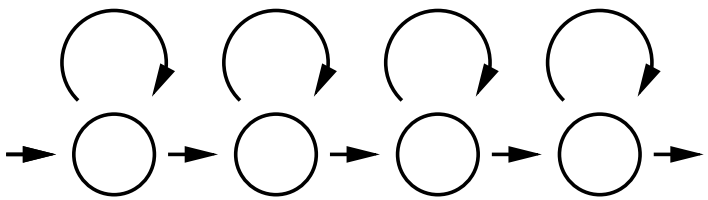
Structures/Requirements for modeling chronologically organized data:



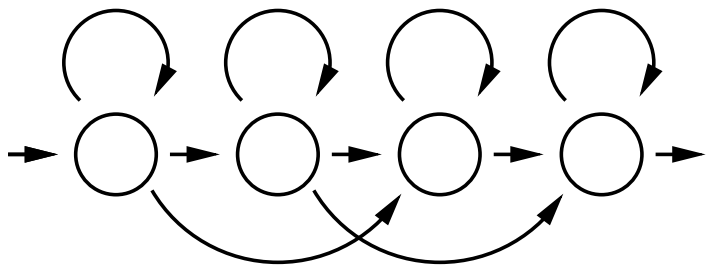
- "Forward" transitions (i.e. progress in time)
- "Loops" for modeling variable durations of segments
- "Skips" allow for optional/missing parts of the data
- Skipping of one or multiple states forward

Configuration of HMMs: Topologies III

Overview: The two most common topologies for handwriting (and speech) recognition:



linear HMM



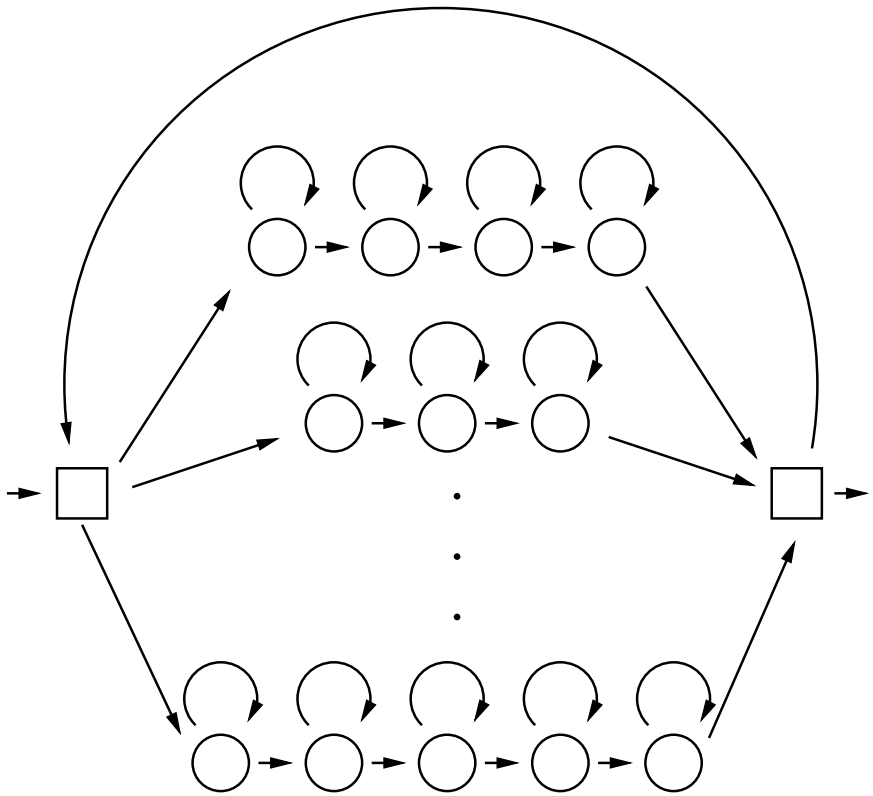
Bakis-type HMM

Note: General left-to-right models (allowing to skip any number of states forward) are not used in practice!

Configuration of HMMs: Compound Models

Goal: Segmentation

- Basic units: Characters
  [Also: (sub-)Stroke models]
- Words formed by concatenation
- Lexicon = parallel connection
  [Non-emitting states merge edges]
- Model for arbitrary text
  by adding loop



$\Rightarrow$ Decoding the model produces segmentation
(i.e. determining the optimal state/model sequence)

Robust Parameter Estimation: Overview

Task: Estimate models' parameters *robustly* by learning from training samples

- ✓ Efficient training algorithms (Forward-Backward, Baum-Welch) exist
- ↯ *Sparse Data Problem:* Amount of sample data usually too small!

⇒ Basic techniques [↗ Theory] alone do not lead to robust models *in practice*!

Approach: Tackle factors which cause the sparse data problem, e.g.

1. **Model complexity**
   (reduce # parameters by merging similar parts of models)
2. **Dimensionality of the data / Correlations** [not covered *here*]
   (feature selection, feature optimization like PCA, LDA, ICA, etc.)
3. **Initialization**
   (*non*-random initialization of structures and parameters for HMM training)

Robust Parameter Estimation: Tying

. . . explicit reduction of parameters by merging similar model parameters.

Constructive tying: Construction of complex models from elementary models.
⇒ Copies of subunits reused reference same set of parameters.

Generalization of special modeling parts where only few samples are available:
Robust estimation of general units exploiting broader basis of data
⇒ Use of suitable generalizations instead of special models

Agglomeration: Identification of similar parameters by suitable distance measure
⇒ Construct parameter space groups by clustering w.r.t. to similar "purpose"

⇒ *Establish modeling units where sufficient # training samples are available.*

Model Subunits

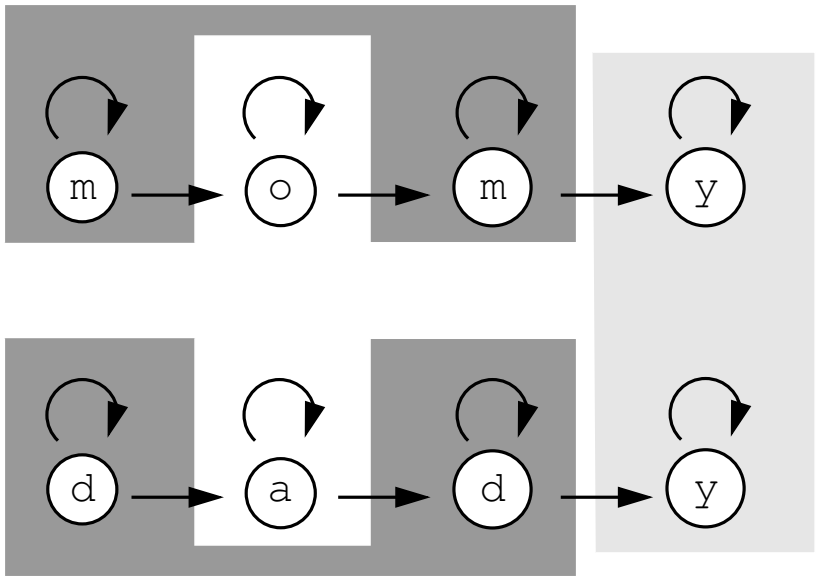
Basic approach: Merge parameters of similar model parts – construct HMMs by combination of base models

Model Generalization: (Usually applied for context dependent modeling)

- Define context dependent base models (e.g. `# / m / o`, `m / o / m`)
- Generalize context restriction if number of training samples not sufficient (e.g. `m / o / m` → `_ / o / m`)
- STOP Hardly used for handwriting recognition, *so far* – [Fin07b]

Model Clustering: Merge similar model parts by e.g. applying decision trees

- STOP Expert knowledge required



- Modularization on level of letters

✓ Intra-word tying

✓ Inter-word tying

⇒ Effective use of sample data

State Tying

. . . parameter groups are chosen with higher granularity compared to model tying

State Clustering: (Required: Distance measure d and termination criterion)

1. Create an individual state cluster for all model states i

$$C_i = \{i\} \quad \forall i, 0 \leq i \leq N$$

and combine these to form the initial set of clusters $\mathcal{C}$:

$$\mathcal{C} = \bigcup_{i=1}^{N} \{C_i\}$$

2. Choose from current set $\mathcal{C}$ that pair C_j, C_k minimizing distance measure d:

$$(C_j, C_k) = \underset{C_p, C_q \in \mathcal{C}}{\operatorname{argmin}}\, d(C_p, C_q)$$

3. Construct new state cluster C by merging C_j and C_k:

$$C \leftarrow C_j \cup C_k$$

Remove original clusters C_j, C_k from $\mathcal{C}$ and insert newly created C instead:

$$\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_j, C_k\} \cup \{C\}$$

4. **if** termination criterion not yet satisfied for current set $\mathcal{C}$: goto step 2
   otherwise: Stop!

Tying in Mixture Models

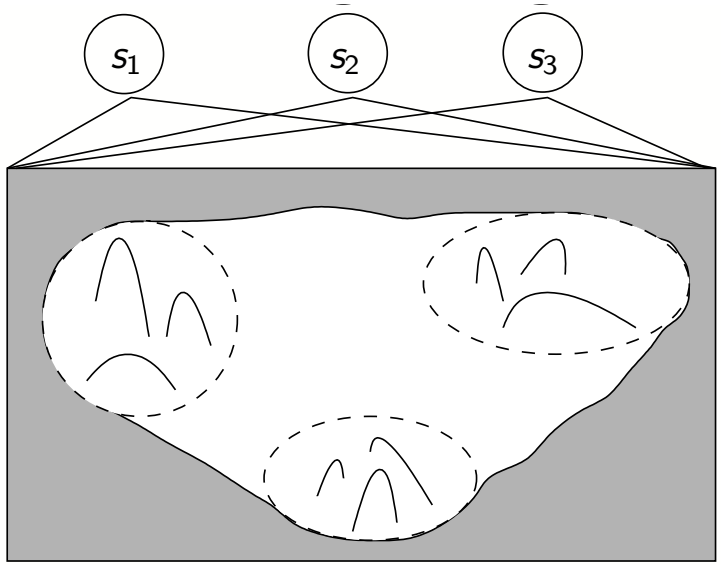
. . . merge similar parameters within mixture density models for emissions of HMMs

Mixture tying: *Semi-continuous* HMMs [Hua89]

- Create shared set of baseline densities
  $\Rightarrow$ Individual components of mixtures are tied across distributions
- New definition of output probabilities:

$$b_j(\mathbf{x}) = \sum_{k=1}^{M} c_{jk}\, \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{K}_k)$$

$$= \sum_{k=1}^{M} c_{jk}\, g_k(\mathbf{x})$$



semi–continuous modeling

Clustering of densities: Automatic tying approach (in contrast to SCHMMs)

- Exemplary distance measure used for clustering procedure [↗ State Tying]

$$d(C_i, C_j) = \frac{N_i\, N_j}{N_i + N_j}\, ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|| \quad N_i, N_j : \#\text{ samples assigned to } i,j\text{-th density}$$

Tying of covariances: Use global covariance matrices for all mixture densities

Parameter Initialization

Problem: [↗ Training] methods for HMMs require suitable initial model
⇒ In practice random or uniform initialization of parameters is **not** sufficient!

Segmental *k*-means: Alternately carry out segmentation of training data and compute new parameters based on segmentation *not* changing model structure

Initialization: Derive initial segmentation of sample data from reference annotation, goto step 2

1. Segmentation: Compute $\mathbf{s}^*$ for $\mathbf{O}$ given λ [↗ Viterbi algorithm], update $\hat{a}_{ij}$:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \chi_t(i)\,\chi_{t+1}(j)}{\sum_{t=1}^{T-1} \chi_t(i)}$$

2. Estimation: For all states j, $0 \leq j \leq N$, and associated partial sample sets $\mathbf{X}(j)$:
   1. Compute VQ codebook $Y = \{\mathbf{y}_1, \ldots \mathbf{y}_{M_j}\}$ and partition $\{R_1, \ldots R_{M_j}\}$
   2. Compute updated parameters of output distributions:

$$\hat{c}_{jk} = \frac{|R_k|}{|\mathbf{X}(j)|}, \quad \hat{\boldsymbol{\mu}}_{jk} = \mathbf{y}_k, \quad \hat{\mathbf{C}}_{jk} = \frac{1}{|R_k|} \sum_{\mathbf{x} \in R_k} \mathbf{x}\,\mathbf{x}^T - \hat{\boldsymbol{\mu}}_{jk}\,\hat{\boldsymbol{\mu}}_{jk}^T$$

3. Termination: **if** $P^*(\mathbf{O}|\hat{\lambda})$ was improved by $\hat{\lambda}$ w.r.t. λ: let $\lambda \leftarrow \hat{\lambda}$, goto step 2
otherwise: Stop!

Efficient Model Evaluation: Overview

Task: Evaluate statistical models (HMMs, *n*-grams) for practical recognition tasks

- ✓ Algorithms for decoding HMMs [↗ Viterbi], and evaluating *n*-gram models [↗ backing off] exist
- ↯ For large inventories: More *efficient* methods required!

⇒ Basic techniques alone not sufficient for efficient model evaluation *in practice*!

Approaches: (selection covered *here*)

1. **Pruning**
   (skip "unnecessary" computations as much as possible by early discarding "less promising" solutions from further search process – suboptimal)
2. **Tree-like model structures**
   (Re-organization of the Search Space using tree lexica)

Efficient Evaluation: Beam Search

Viterbi: Linear time complexity but still quadratic complexity in number of model states

- Viterbi matrix of $\delta_t(i)$ grows linearly with # states
  Even with restricted [↗ model topology]
- Complete evaluation for limited problems only

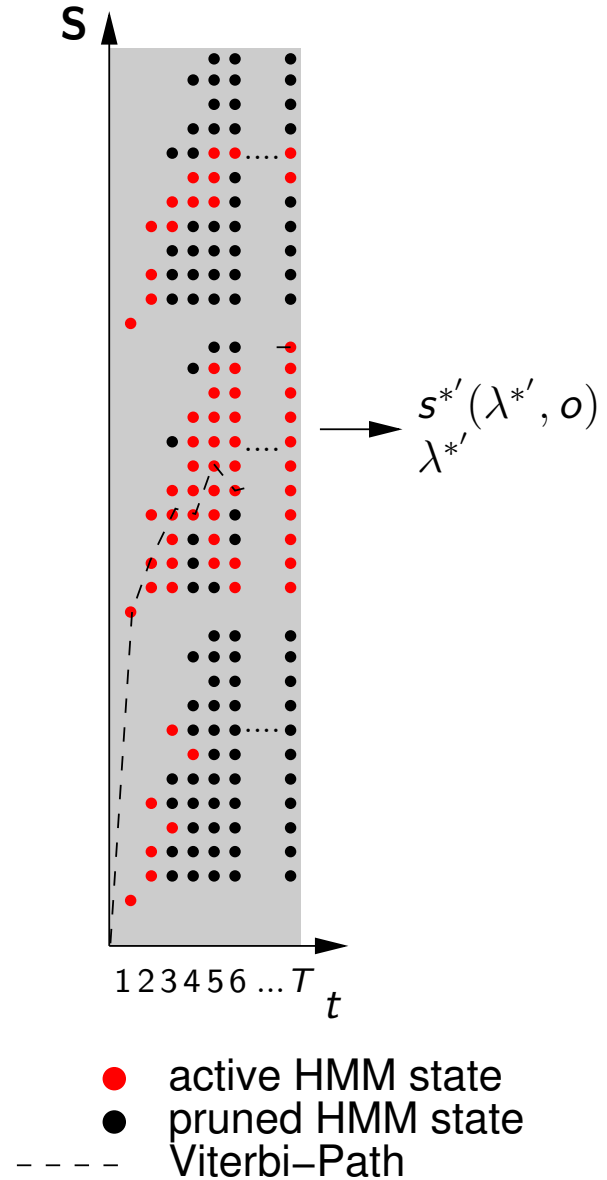
Idea: Discard "less promising" solutions during decoding

Beam-Search: Use relative differences in $\delta_t(i)$ for search search to "beam" around best partial solution [Low76]

- Evaluation of $\delta_t(i)$ on limited set of active states:

$$\begin{aligned} \mathcal{A}_t &= \{i | \delta_t(i) \geq B\, \delta_t^*\} \\ &\quad \text{with} \quad \delta_t^* = \max_j \delta_t(j) \text{ and } 0 < B \ll 1 \\ \delta_{t+1}(j) &= \max_{i \in \mathcal{A}_t} \left\{\delta_t(i) a_{ij}\right\} b_j(O_{t+1}) \\ B &= [10^{-10} \ldots 10^{-20}] \quad \text{typically} \end{aligned}$$

- Practice: Directly propagate calculations from respective active states to potential *successors*.



The Beam Search Algorithm

Initialization: Initialize set of active states with non-emitting state 0: $\mathcal{A}_0 \leftarrow \{0\}$

Propagation: For all times t, $t = 1 \ldots T$:

- Initialize locally optimal path score $\tilde{\delta}_t^* \leftarrow \infty$
- For all $i \in \mathcal{A}_{t-1}$ and all $j \in \{j|j = \text{succ}(i)\}$:
  - Compute local path score from state i to its successor j:
    $\tilde{\delta}_t(i,j) = \tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + \tilde{b}_j(O_t)$
  - Update partial path score for state j, if necessary
    if $\tilde{\delta}_t(j)$ has not yet been computed or
    $\tilde{\delta}_t(i,j) < \tilde{\delta}_t(j)$: $\quad \tilde{\delta}_t(j) \leftarrow \tilde{\delta}_t(i,j)$, $\psi_t(j) \leftarrow i$
  - Update locally optimal path score
    if $\tilde{\delta}_t(i,j) < \tilde{\delta}_t^*$: $\quad \tilde{\delta}_t^* \leftarrow \tilde{\delta}_t(i,j)$
- Determine set $\mathcal{A}_t$ of active states
  - Initialize new set of active states: $\mathcal{A}_t \leftarrow \emptyset$
  - Add all successors j of active states i which lie within the beam
    for all $i \in \mathcal{A}_{t-1}$ and all $j \in \{j|j = \text{succ}(i)\}$
    if $\tilde{\delta}_t(j) <= \tilde{\delta}_t^* + \tilde{B}$: $\quad \mathcal{A}_t \leftarrow \mathcal{A}_t \cup \{j\}$

Termination: Determine optimal end state $\hat{s}_T^* := \underset{j \in \mathcal{A}_T}{\text{argmin}}\, \tilde{\delta}_T(j)$

Backtracking of approximately optimal path: For all times t, $t = T-1 \ldots 1$:

$$\hat{s}_t^* = \psi_{t+1}(\hat{s}_{t+1}^*)$$

Forward-Backward Pruning

. . . also accelerate optimization steps of (iterative) *training* procedures for HMMs!

Approach: Avoid "unnecessary" computations, unless negative effects on quality of parameter estimation are observed (cf. [↗ Beam-Search Algorithm])

Pruning: Restrict [↗ Baum-Welch Algorithm] to relevant parts of search space

- During computation of forward and backward variables for every t consider active states depending on optimal $\alpha_t^* = \max_j \alpha_t(j)$ / $\beta_t^* = \max_j \beta_t(j)$

$$\mathcal{A}_t = \{i|\alpha_t(i) \geq B\,\alpha_t^*\} \quad \text{with} \quad \alpha_t^* = \max_j \alpha_t(j) \text{ and } 0 < B \ll 1$$

$$\mathcal{B}_t = \{i|\beta_t(i) \geq B\,\beta_t^*\} \quad \text{with} \quad \beta_t^* = \max_j \beta_t(j) \text{ and } 0 < B \ll 1$$

- Modified recursive computation rules for forward and backward variables:

$$\alpha_{t+1}(j) := \sum_{i \in \mathcal{A}_t} \{\alpha_t(i) a_{ij}\}\, b_j(O_{t+1})$$

$$\beta_t(i) := \sum_{j \in \mathcal{B}_{t+1}} a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j)$$

- $\gamma_t(i)$ vanishes for all t/s for which $\alpha_t(i)$ or $\beta_t(i)$ are not calculated

Simplification: Apply pruning to $\alpha_t(.)$, evaluate $\beta_t(.)$ for non-vanishing $\alpha_t(.)$ only

Efficient Evaluation: Tree Lexicon

Observation: In large lexica many words share common prefixes

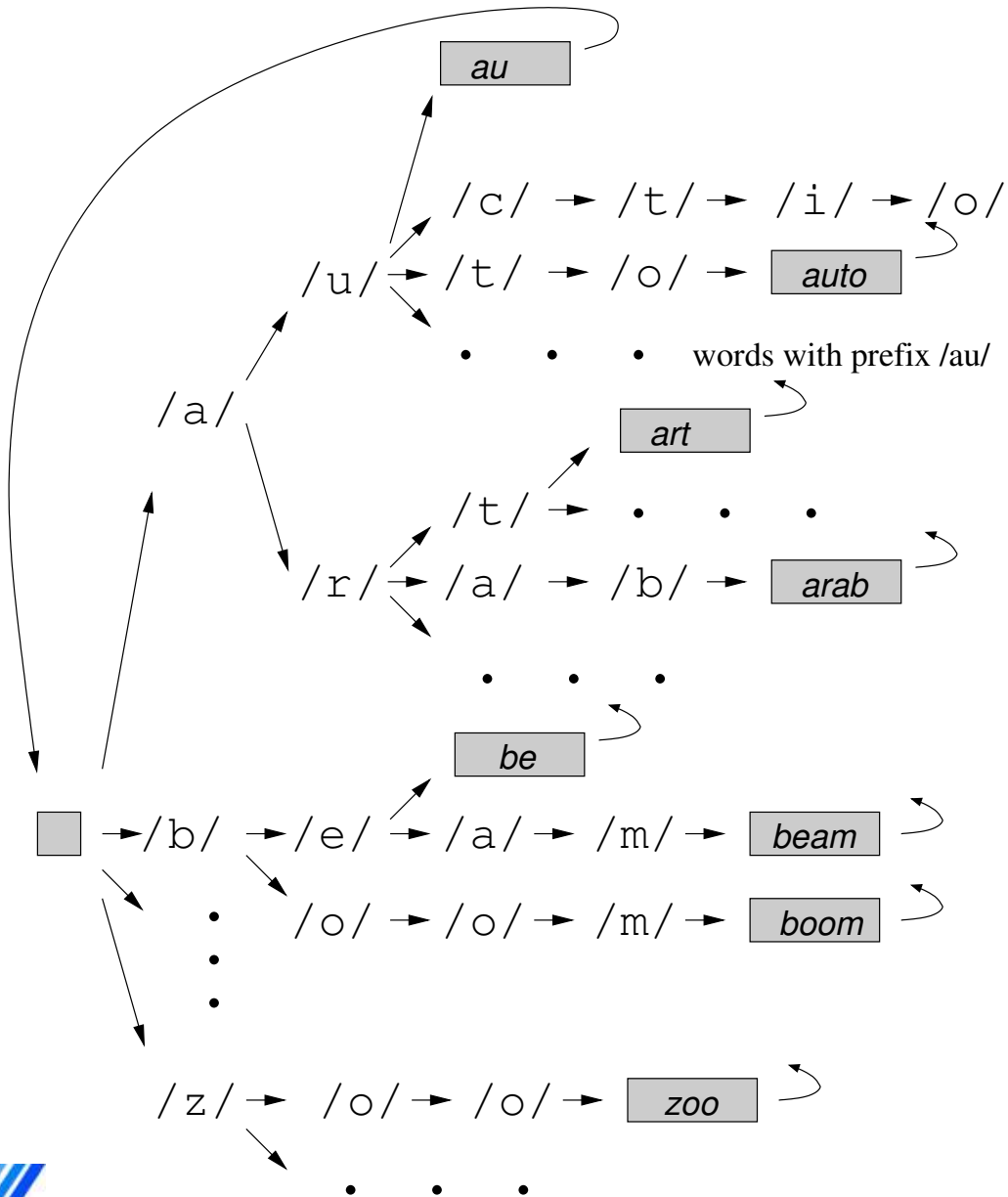
- Standard model construction (concatenation of character HMMs): Many word models contain identical copies of states!
- Redundancy causes unnecessary evaluations in search (↑)

Idea: "Compress" representation ⇒ Prefix tree

Procedure: All words sharing a common character sequence as prefix now are successors of *a single* HMM / state sequence

- ✓ Decoding of semantically identical states performed only once!
- ✓ Compression of search space by factor 2 to 5, most gain in efficiency at beginning of words

Efficient Evaluation: Tree Lexicon II



Note: Word identities known only after pass through tree is completed (i.e. at the leaves)
⇒ Difference to standard (non-tree) representation!

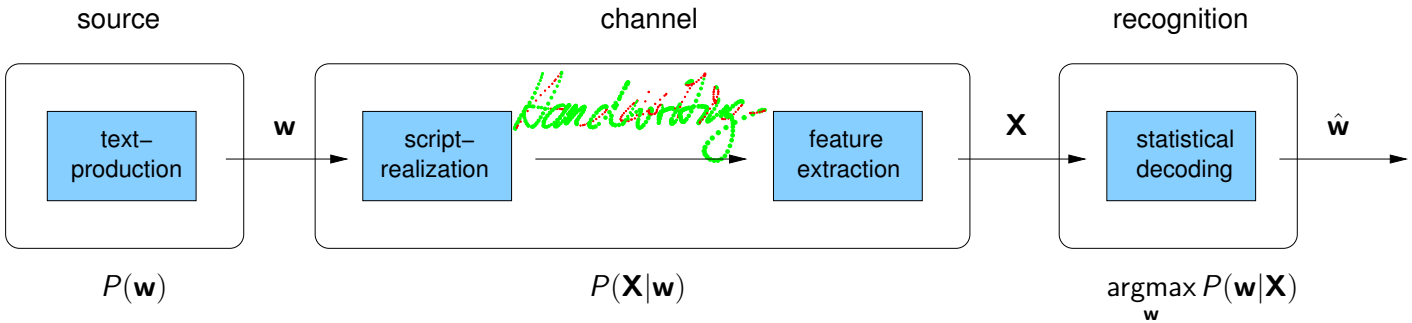
Example:

- Lexicon size: 7.5k words
- 52k character models in total,
- 26k tree nodes only

⇒ Compression by factor 2!

Integrated Search: Introduction

Remember the channel model:



$\Rightarrow$ HMMs + n-gram models *frequently* used in combination!

Problems in practice:

- *How to compute a combined score?* Channel model defines basis only!
- *When to compute the score?* Model valid for *complete* HMM results!
- *How does the language model improve results?*

STOP *Why not use HMMs only to avoid those problems?*

Integrated Search: Basics

Problem 1: Multiplication of $P(\mathbf{X}|O)$ and $P(\mathbf{w})$ does not work in practice!

$\Rightarrow$ Weighted combination using "linguistic matching factor" ρ

$$P(\mathbf{w})^{\rho}\, P(\mathbf{X}|\mathbf{w})$$

Reason: HMM and n-gram scores obtained at largely different time scales and orders of magnitude

- HMM: multi-dimensional density per frame
- n-gram: conditional probability per word

Problem 2: Channel model defines score combination for complete results!

- Can be used in practice only, if ...
  - HMM-based search generates multiple alternative solutions ...
  - n-gram evaluates these *afterward*.
  - $\Rightarrow$ No benefit for HMM search!

$\Rightarrow$ Combination must apply to intermediate results, i.e. path scores $\delta_t(.)$

✓ Achieved by using $P(z|\mathbf{y})$ as "transition probabilities" at word ends.

Integrated Search: Basics II

Question: *How does the language model influence the quality of the results?*

Rule-of-thumb: Error rate decreases proportional to square-root of perplexity

Example for lexicon-free recognition (IAM-DB) with character *n*-grams [Wie05]

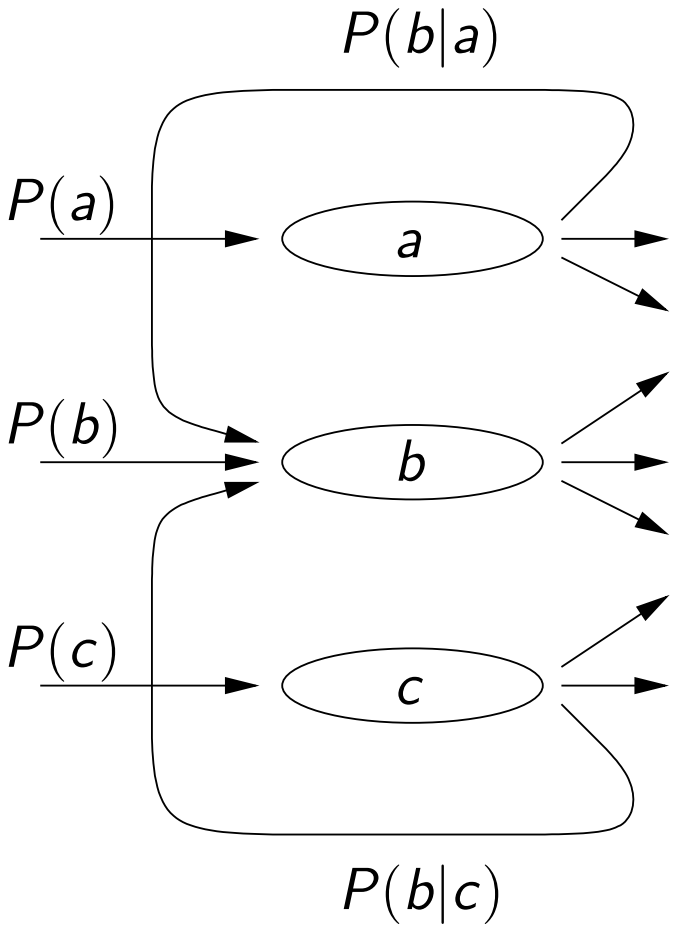
	% CER / perplexity				
	none	2	3	4	5
IAM-DB	29.2 / (75)	22.1 / 12.7	18.3 / 9.3	16.1 7.7	15.6 / 7.3
CER/$\sqrt{\mathcal{P}}$	n.a.	6.2	6.0	6.0	5.8

Note: Important plausibility check: If violated, something *strange* is happening!

Integrated Search: HMM Networks

- Straight-forward extension of HMM-only models
- n-gram scores extend transition probabilities
- HMMs store single-state context only *Rightarrow* only bi-grams usable!

Question: *How can higher-order models (e.g. tri-grams) be used?*

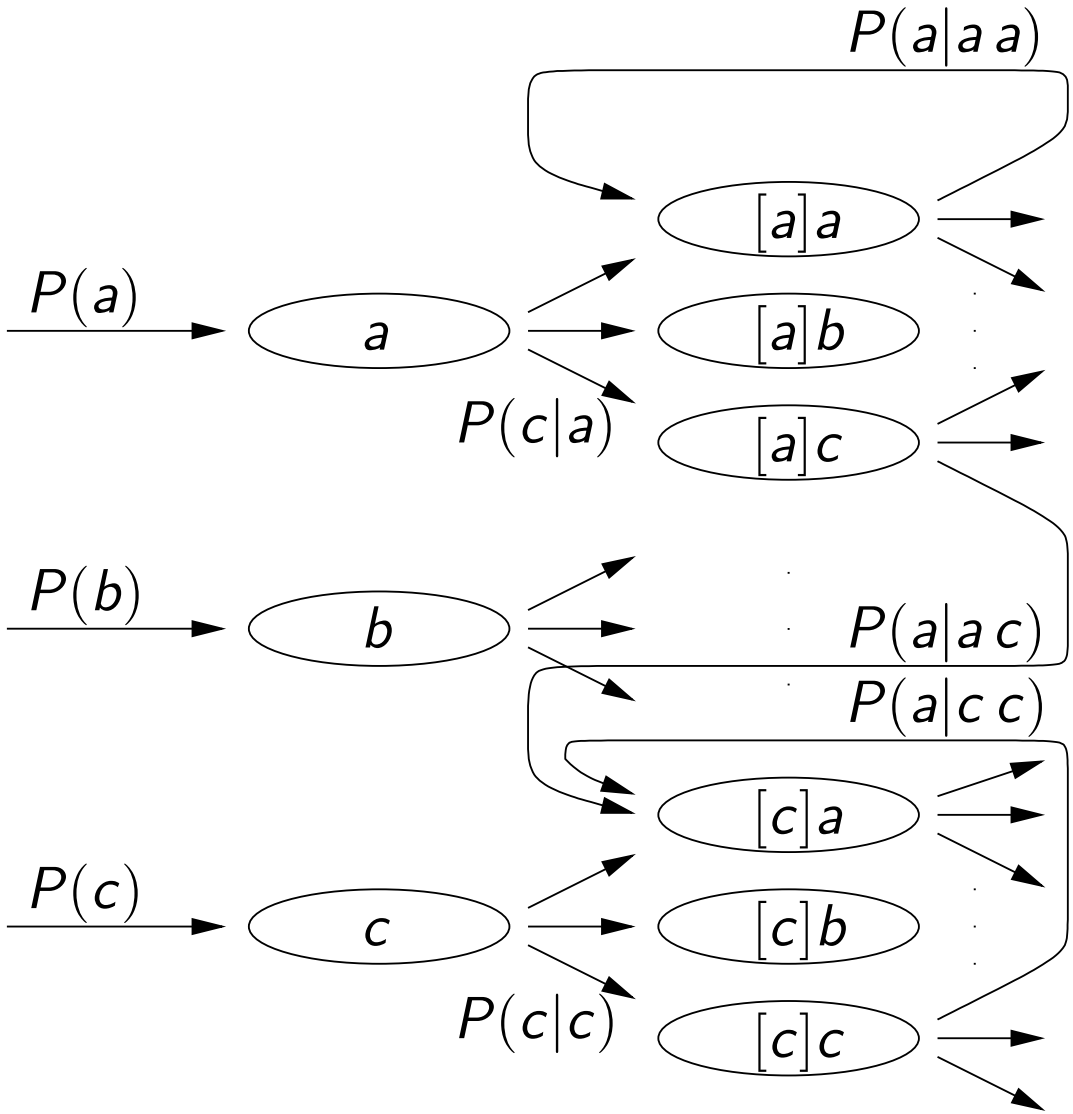


Integrated Search: HMM Networks II

Higher-order n-gram models:

⇒ Context dependent copies of word models (i.e. state groups) necessary!

↯ Total model grows exponentially with n-gram order!



Integrated Search: Search Tree Copies

Note: In *large vocabulary* HMM systems models are usually compressed by using a [↗ prefix tree] representation.

Problem: Word identities are only known *at the leaves* of the tree (i.e. *after* passing through the prefix tree)

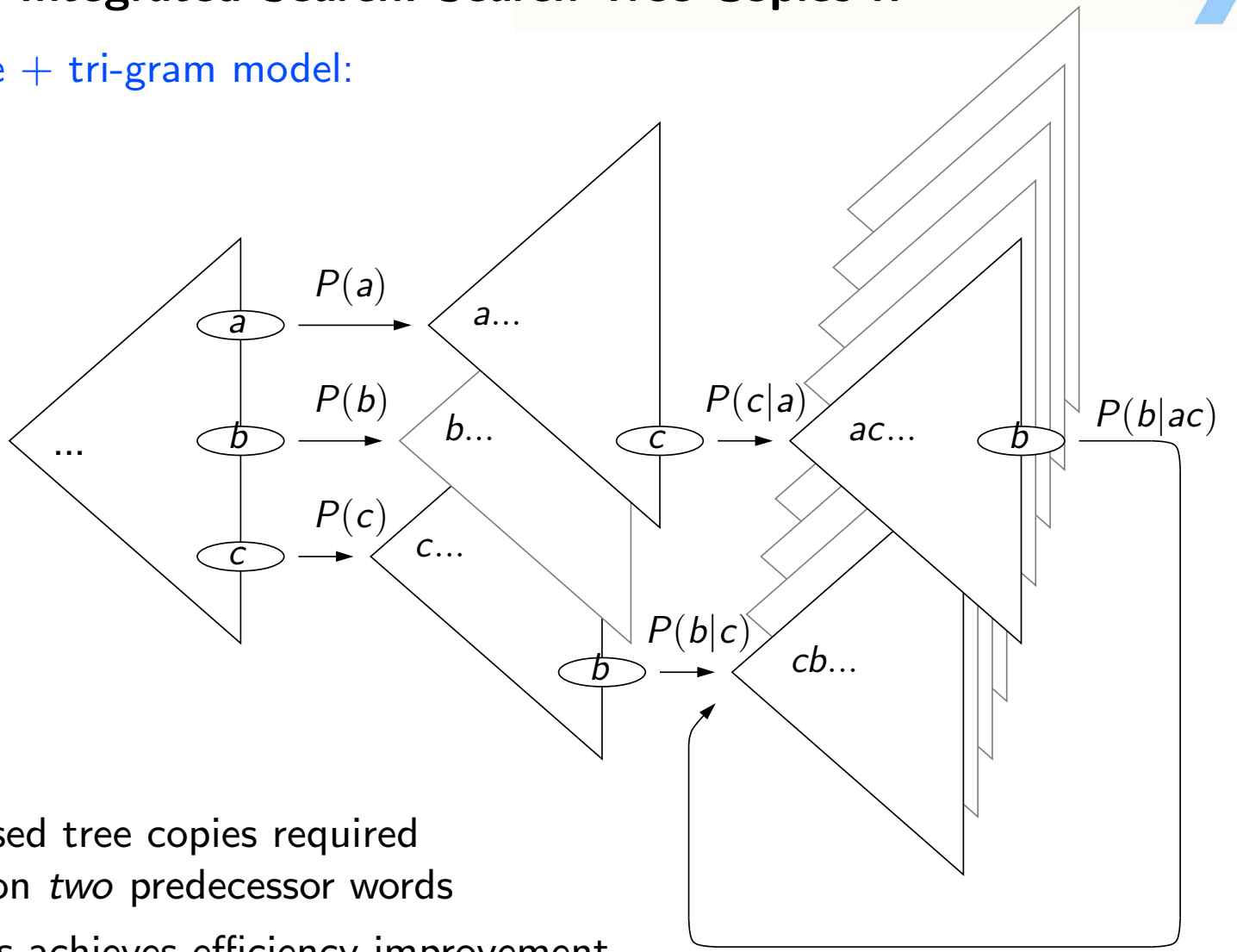
Question: *How to integrate a language model?*

Solution:

- "Remember" identity of last word seen and ...
- Incorporate n-gram score with one word delay.
- ↯ Search tree copies required!

Integrated Search: Search Tree Copies II

HMM prefix tree + tri-gram model:



- ↯ Context based tree copies required depending on *two* predecessor words
- ✓ Nevertheless achieves efficiency improvement as HMM decoding effort is reduced

Integrated Search: Multi-Pass Search

Problem: Integrated use of higher order n-gram models expensive!

Solution: Multiple search "phases" with increasing model complexity

1. HMM decoding (+ bi-gram)

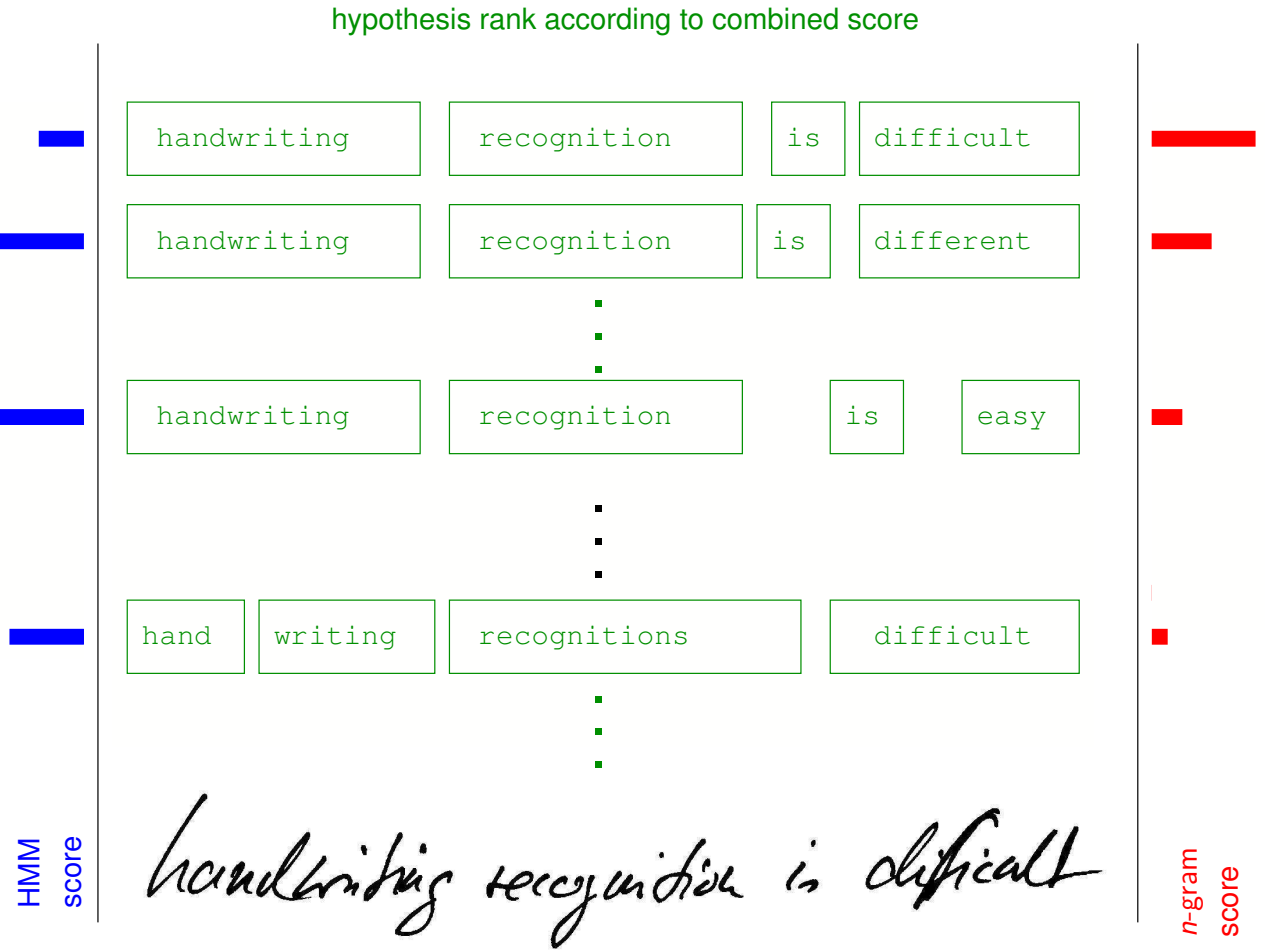
   Alternative solutions!

2. n-gram (e.g. tri-gram) for rescoring

   $\Rightarrow$ existing solutions *sorted* differently!

3. $n + k$-gram

   ... continue with 2.



Overview

- Motivation — *. . . Why use Markov Models?*
- Theoretical Concepts — *. . . Hidden Markov and n-gram Models*
- Practical Aspects — *. . . Configuration, Robustness, Efficiency, Search*
- **Putting It All Together** — *. . . How Things Work in Reality*
  - Preprocessing — *Remove unwanted variation*
  - Linearization — *for processing by sequential model*
  - Feature Extraction — *Determine relevant "properties"*
  - Writing Model
  - Language Modeling & Decoding
- Summary — *. . . and Conclusion*

Putting It All Together: Introduction

Tasks investigated:

- Recognition of unconstrained handwriting (Roman alphabet)
- Minimum element size: Words or phrases
  STOP No isolated characters! $\longrightarrow$ problem of *classification* only!

Focus on *offline* recognition

Online recognition?

- Considered easier than offline recognition (cf. e.g. [Bun03])
- Problem "solved" for many scripts by Microsoft?
  (impressive online recognizer!)
- Research in online recognition rather specialized today
  (exotic scripts, special tasks, ...)

Note: Hardly any "standard" procedures exist!
(as opposed to speech recognition)

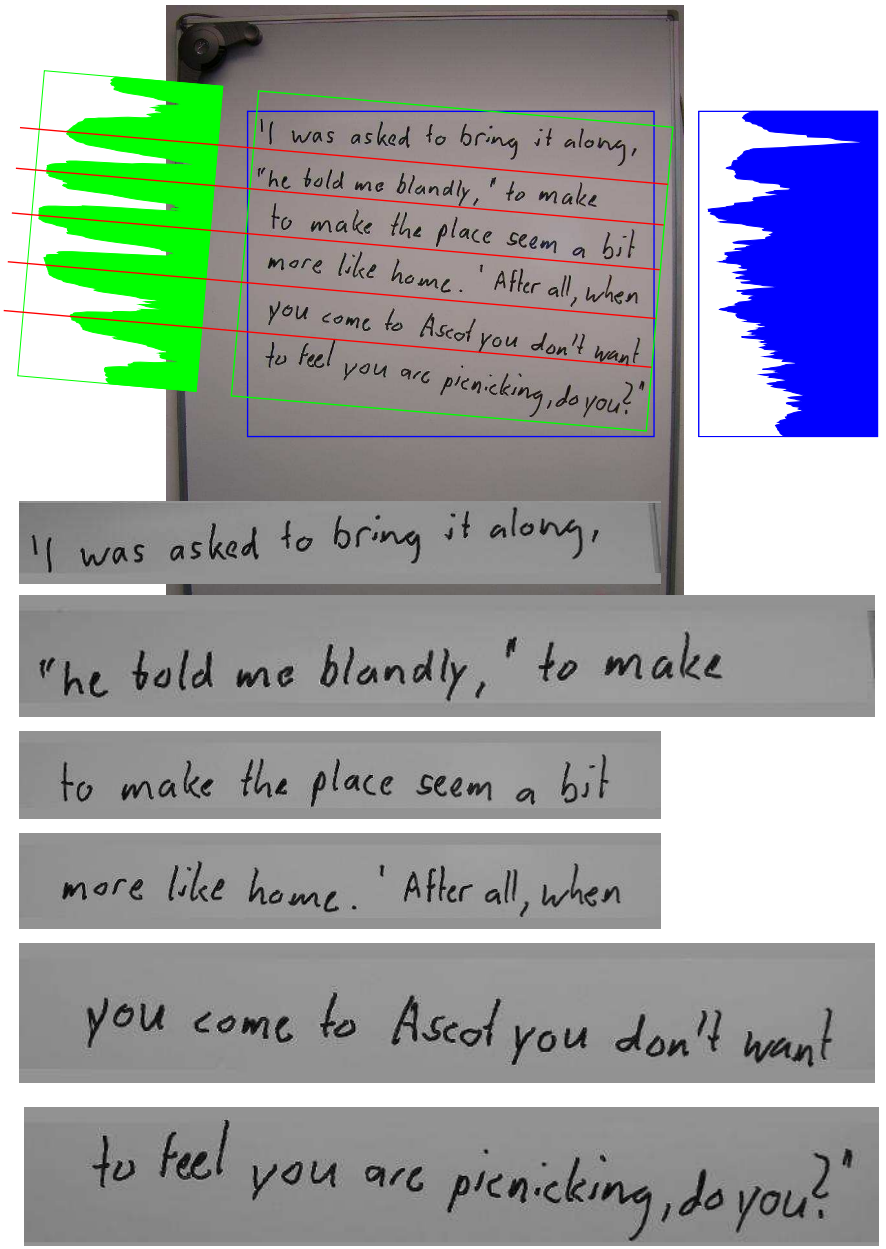
Preprocessing: Offline

Line Extraction

Basis: Document containing handwritten text

Principle Method:
(cf. e.g. [Wie02, Baz99])

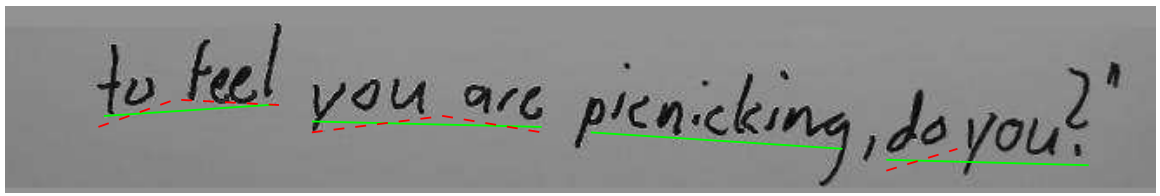
1. Find text regions (if necessary)
2. Correct orientation of text region (minimize entropy of horizontal projection histogram)
3. Extract text lines (segment at minima of projection histogram)



(IAM-OnDB image courtesy of H. Bunke, University of Bern)

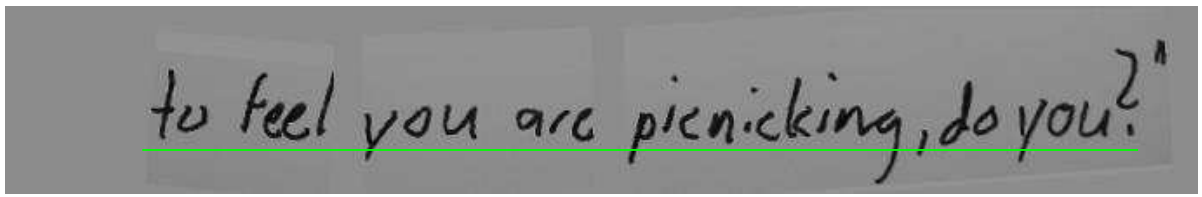
Preprocessing: Offline II

Baseline Estimation:

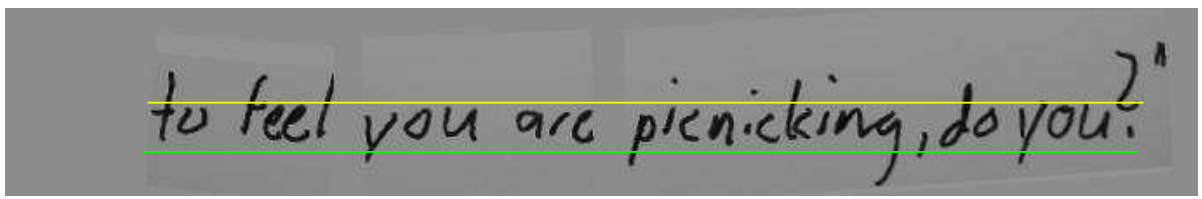


Potential method:
- Initial estimate based on horiz. projection histogram
- Iterative refinement and outlier removal [Wie02]

Skew and Displacement Correction:

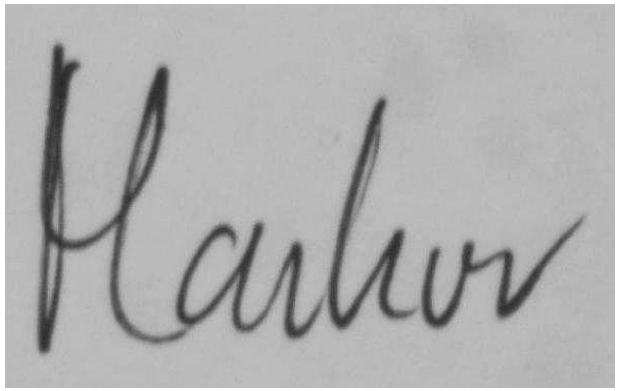


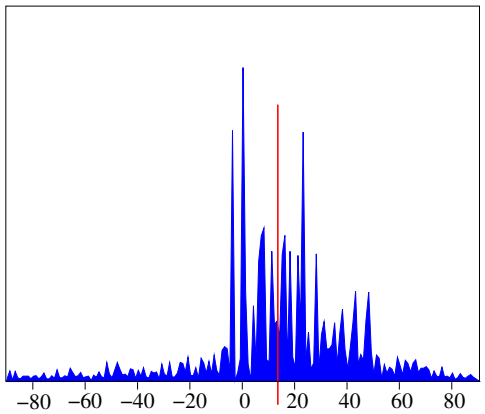
Estimation of core size / upper baseline (not mandatory)



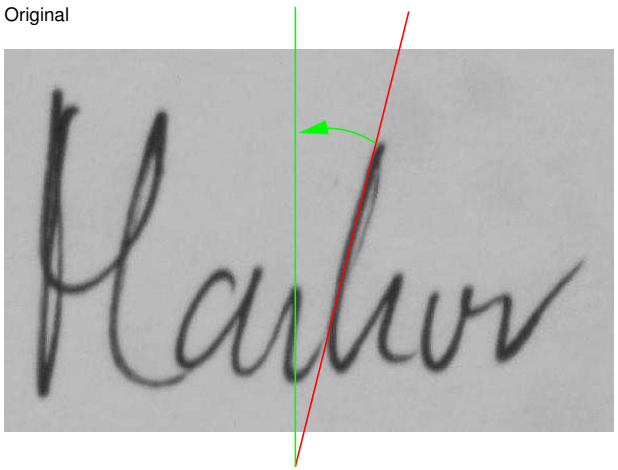
Preprocessing: Offline III

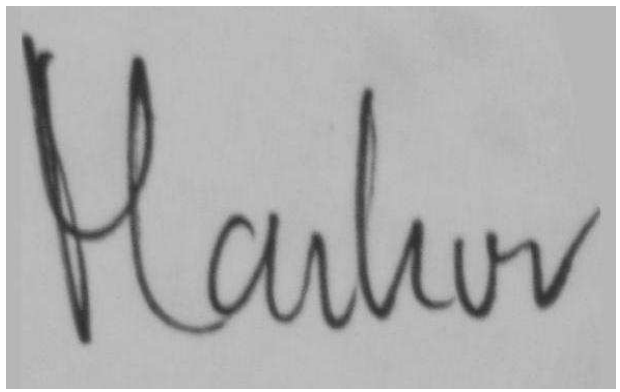
Slant estimation: E.g. via mean orientation of edges obtained by Canny operator (cf. [Wie02])





Slant normalization (by applying a shear transform)





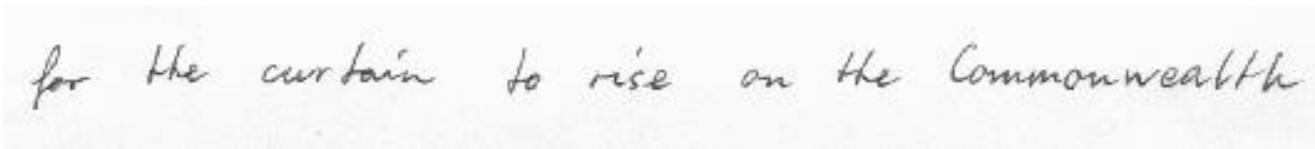
Preprocessing: Offline IV

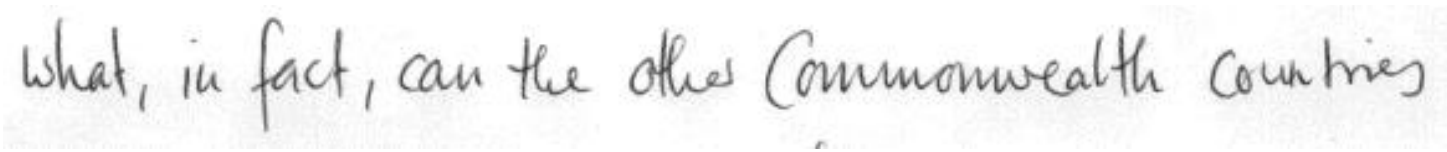
Note: Depending on writer and context script might largely vary in size!

Methods for size normalization:

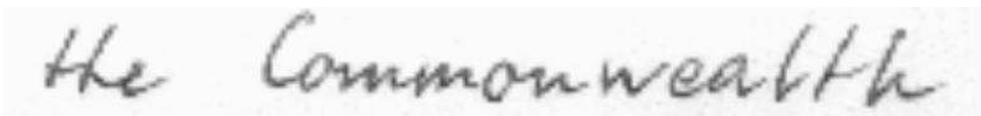
- "manually", heuristically, to predefined width/height???
- depending on estimated core size (← estimation crucial!)
- depending on estimated character width [Wie05]

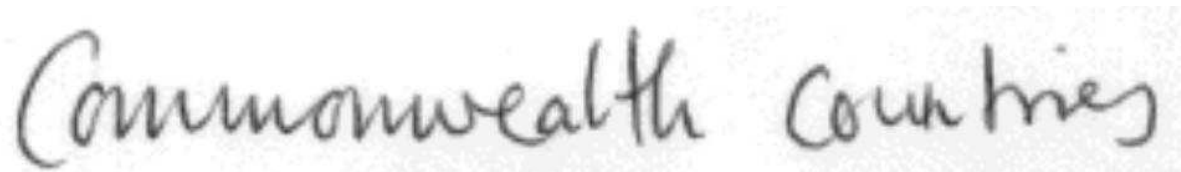
Original text lines (from IAM–DB)





Results of size normalization (avg. distance of contour minima)



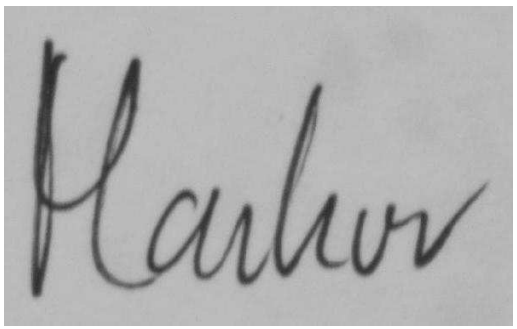


Preprocessing: Offline V

Note: Stroke images vary in *color* (or local contrast) and *stroke width*
$\Rightarrow$ considered irrelevant for recognition

Binarization by global or locally adaptive methods (cf. e.g. [Tri95])

Original

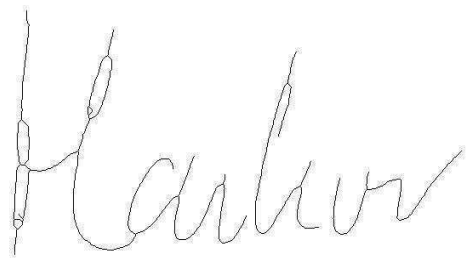


Binarized image (Niblack's method)

Thinning or skeletonization, e.g. by morphological operations (cf. e.g. [Jäh05])

Binarized image

Potential skeleton



Preprocessing: Online

Goal the same as in offline recognition: Remove unwanted variation

Common Methods:

Skew / Slant / Size normalization:

- Trajectory data mapped to 2D representation
- Baselines / core area estimated similar to offline case

Special Online Methods:

Outlier Elimination: Remove position measurements caused by interferences

Resampling and smoothing of the trajectory [↗ Details]

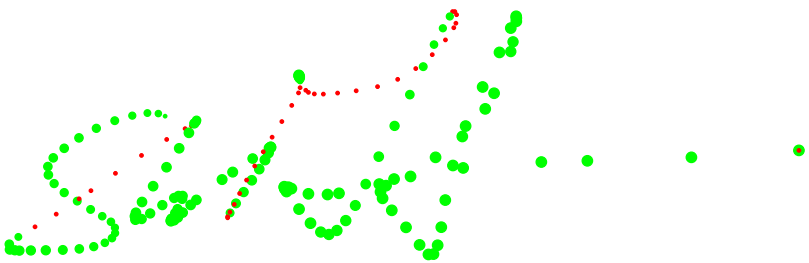
Elimination of delayed strokes [↗ Details]

Preprocessing: Online II

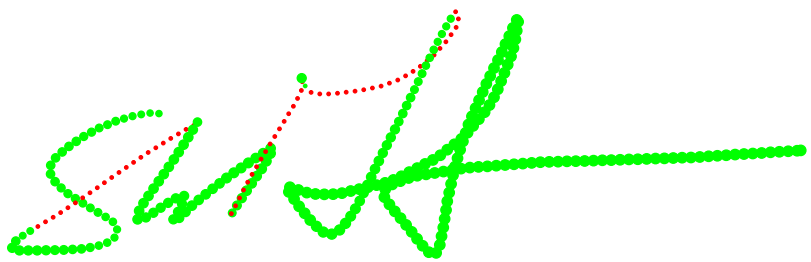
Resampling and smoothing of the trajectory

- Goal: Normalize variations in writing speed (no identification!)
- Equidistant resampling & interpolation (cf. e.g. [Jae01])

Measured trajectory



Result of resampling

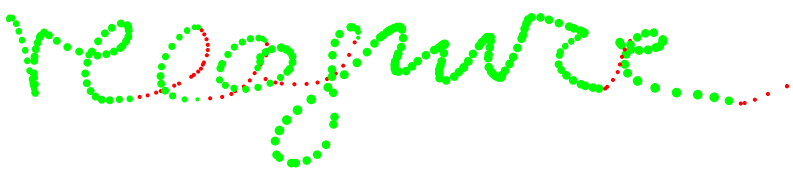


Elimination of delayed strokes (cf. e.g. [Jae01])

- Handling of delayed strokes problematic, additional time variability!
- Remove by heuristic rules (backward pen-up, cf. e.g. [Jae01])

Measured trajectory

Delayed stroke for "i" removed



Note: Delayed strokes treated explicitly in hardly any system! [↗ Features]

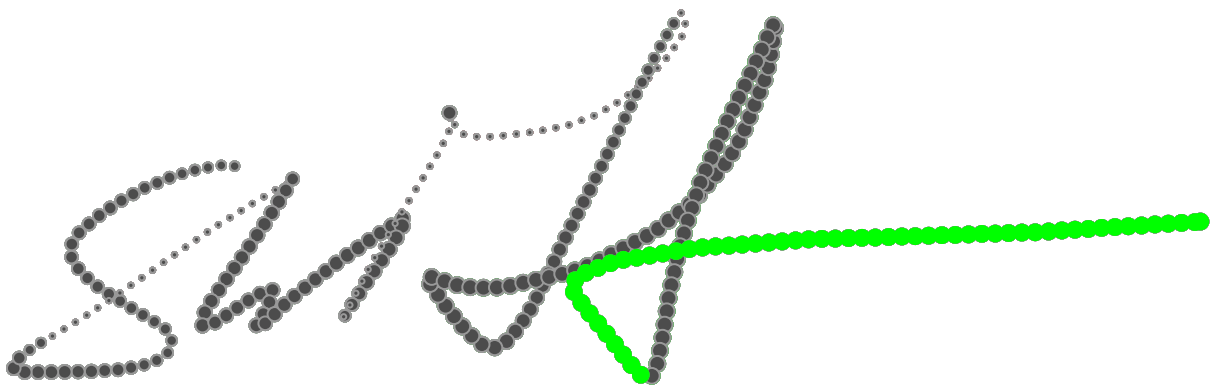
Linearization: Online

Basis: (Rather) Straight-forward

- ✓ Data has inherent temporal structure!
- ⇒ Time axis runs along the pen trajectory

Level of Granularity:

- ↯ Pen position measurements not suitable as elementary trajectory elements
- ⇒ analysis of overlapping segments (stroke-like / fixed size)



[Example: Stroke segmentation at local maxima of curvature]

Linearization: Offline

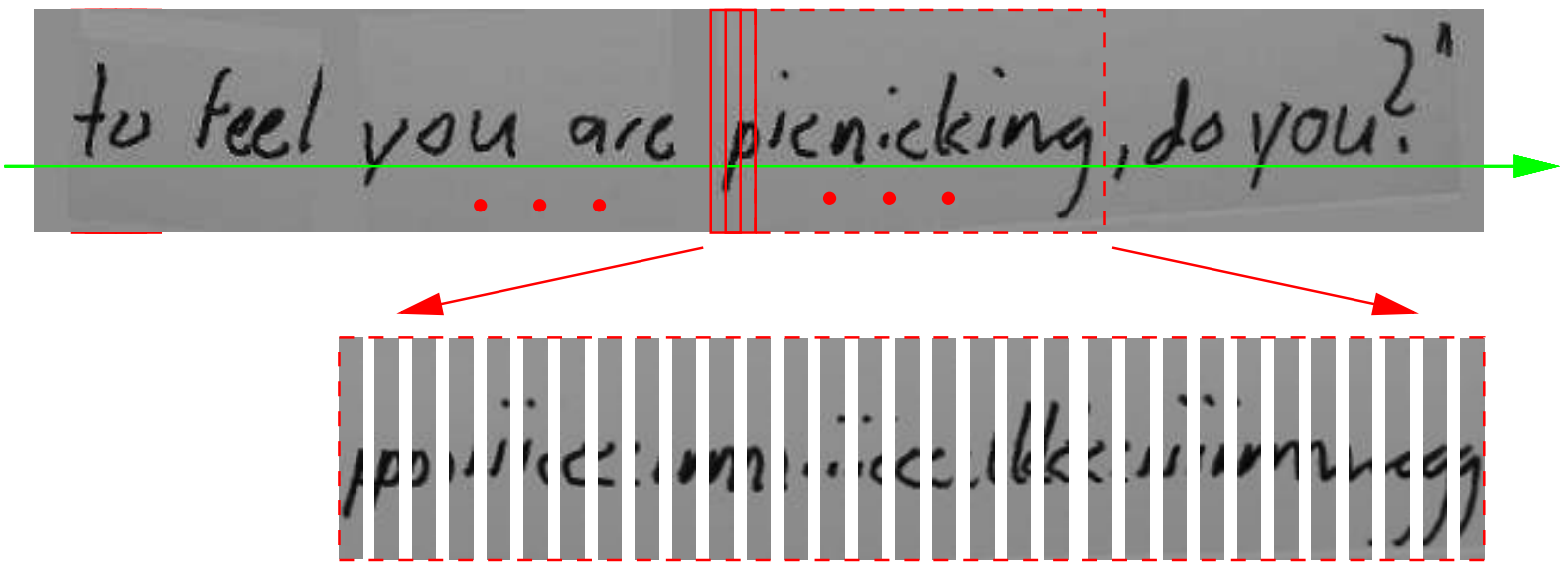
Problem: Data is two-dimensional, images of writing!

↯ No chronological structure inherently defined!

Exception: Logical sequence of characters within texts

Solution: Sliding-window approach first proposed by researchers at BBN [Sch96])

- Time axis runs in writing direction / along baseline
- Extract small overlapping analysis windows



[Frames shown are for illustration only but actually too large!]

Feature Extraction: Online

Basic Idea: Describe shape of pen trajectory locally

Typical Features: (cf. e.g. [Dol97, Jae01])

- Slope angle α of local trajectory
  (represented as $\sin\alpha$ and $\cos\alpha$: continuous variation)
- Binary pen-up vs. pen-down feature
- *Hat feature* for describing delayed strokes
  (strokes that spatially correspond to removed delayed strokes are marked)

Feature Dynamics: In *all* applications of HMMs *dynamic* features greatly enhance performance.

$\Rightarrow$ Discrete time derivative of features

Here: Differences between successive slope angles

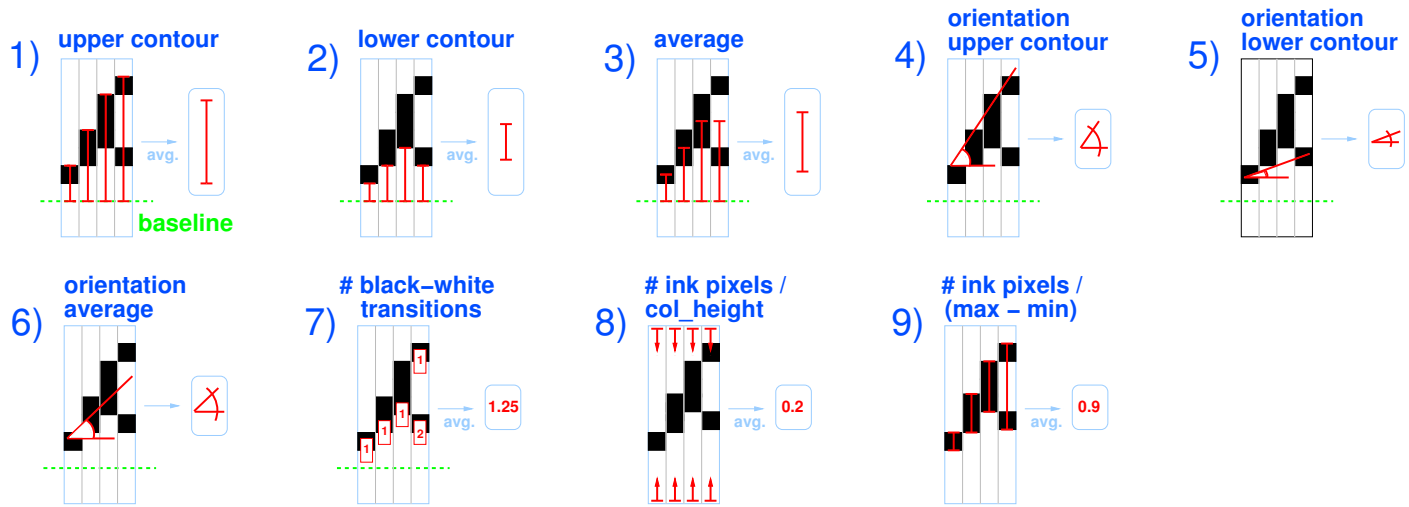
Feature Extraction: Offline

Basic Idea: Describe appearance of writing within analysis window

- No "standard" approaches or feature sets
- Generally no holistic features used

Potential Methods:

- (For OCR) Local analysis of gray-value distributions (cf. e.g. [Baz99])
- Salient elementary geometric shapes (e.g. vertices, cusps)
- Heuristic geometric properties (cf. e.g. [Wie05])



Additionally: Compute dynamic features

Writing Model

Note: No *principal* differences between online and offline

- Context independent elementary HMMs, linear or Bakis topology (for characters, numerals, punctuation symbols, and white space)
  Note:
  - Explicit *ligature models* used in some approaches
  - Hardly any benefit from context-dependent models in offline recognition?
- Output distributions of HMMs: continuous or semi-continuous (i.e. with shared codebook, cf. e.g. [Wie05])
- Task models constructed by concatenation etc. from elementary HMMs
- Initialization
  - *If you start from scratch:* Manual labeling of (part of) sample set required!
  - *If you are lucky:* Use previous system for (supervised) automatic labeling
- Training: Baum-Welch (convergence after $\approx$ 10–20 re-estimation steps)

Language Modeling & Decoding

Language Modeling:

- Standard techniques applied
- no relevant differences to related domains (e.g. speech recognition)

Word level models: Bi- and tri-grams (cf. e.g. [Vin04, Wie05, Zim06])

Character level models: Up to 5-gram [Wie05] and 7-grams [Bra02]
$\Rightarrow$ "unlimited" / open vocabulary systems

Decoding:

- Basic method: Viterbi beam search
- Integration of language model frequently not documented
  - Offline recognition usually *not* interactive

  $\Rightarrow$ multi-pass ("postprocessing-style") integration possible
  - Search tree copies (tree lexicon) (cf. e.g. [Wie05])

No postprocessing in HMM-based systems!
(Restrictions completely represented by the statistical model)

Overview

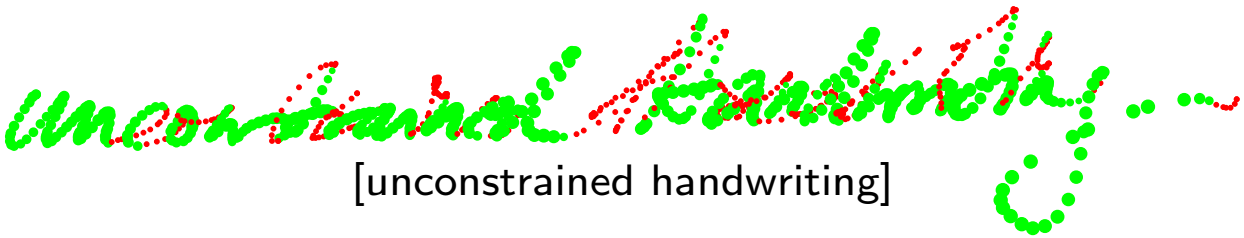
Summary

Handwritten script:

- Natural input modality for human-machine-interaction (e.g. PDAs)
- Subject of automated document processing (e.g. forms)

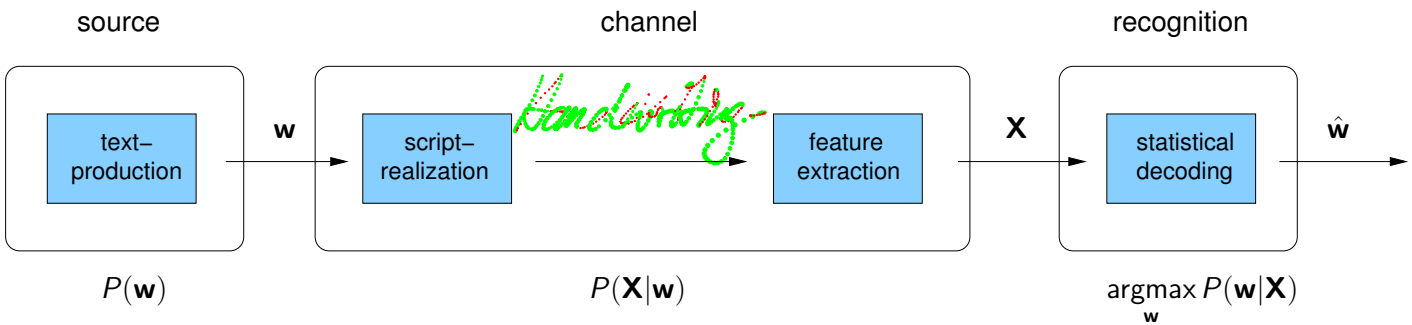
Difficult automatic recognition:

- High variability even within particular characters (size, style, line width etc.)
- Problematic segmentation due to "merging" of adjacent characters.



[unconstrained handwriting]

Pre-dominant recognition approach: temporal features, HMMs + n-grams

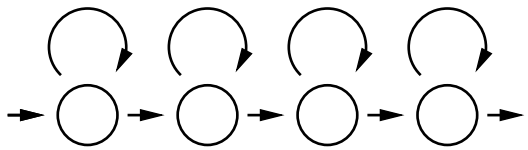


Summary: Modeling

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|\mathbf{X}) = \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(\mathbf{w})P(\mathbf{X}|\mathbf{w})}{P(\mathbf{X})} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w})P(\mathbf{X}|\mathbf{w})$$

Script (appearance) model: $P(\mathbf{X}|\mathbf{w})$ $\Rightarrow$ *Hidden-Markov-Models*

- Two-stage stochastic process (stationary, causal, simple)
- *Hidden* state sequence, observable emissions
- ✓ Emission modeling: Discrete, or continuous
- ✓ Algorithms for scoring, decoding, training



Language model: $P(\mathbf{w})$ $\Rightarrow$ *n-Gram-Models*

- Probability distribution over set of symbol (= word) sequences
- Principally: Direct computation of parameters (by counting)
- ✓ Algorithms for scoring / evaluation

$$P(\mathbf{w}) \approx \prod_{t=1}^{T} P(\underbrace{w_t \mid w_{t-n+1}, \ldots, w_{t-1}}_{n \text{ symbols}})$$

Summary: Practice

Beware: Theoretical concepts *alone* not sufficient for suitable recognition systems!

Problems: (and solutions)

- Numerical problems
  (vanishing probabilities for long sequences)
  ⇒ Representation in negative logarithmic domain, flooring ✓
- Sparse data problem
  (model complexity, dimensionality of data; unseen events)
  ⇒ Concatenation of small basis models, feature optimization ✓
- Efficiency issues
  (complexity of search space, redundancy)
  ⇒ Pruning (Beam search, forward-backward pruning), tree-lexica ✓

Recognition systems: *Integrated* evaluation of HMMs and n-gram models for suitably pre-processed handwriting data

- ✓ Linearization of input data (sliding window) + feature extraction
- ✓ Use HMM "on top" for combination of HMM and n-gram scores.

ESMERALDA

An integrated
Environment for **S**tatistical **M**odel **E**stimation and **R**ecognition on **A**rbitrary **L**inear **D**ata **A**rrays

Developed at Bielefeld, now Dortmund University (Germany) [Fin99, Fin07a]

Supports: (primarily)

- (SC)HMMs of different topologies
  (with user-definable internal structure)
- Incorporation of n-gram models
  (for long-term sequential restrictions)
- Gaussian mixture models (GMMs).

Used for numerous projects within:

- **Handwriting Recognition**,
- Automatic Speech Recognition,
- Analysis of biological sequences,
- Music analysis.

Availability: Open source software (LGPL) sourceforge.net/projects/esmeralda

References I

[Baz99] Bazzi, I., Schwartz, R., and Makhoul, J.
An Omnifont Open-Vocabulary OCR System for English and Arabic.
IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 21(6):495–504, 1999.

[Bra02] Brakensiek, A., Rottland, J., and Rigoll, G.
Handwritten Address Recognition with Open Vocabulary Using Character N-Grams.
In *Proc. Int. Workshop on Frontiers in Handwriting Recognition*, pages 357–362. Niagara on the Lake, Canada, 2002.

[Bun03] Bunke, H.
Recognition of Cursive Roman Handwriting – Past, Present and Future.
In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 448–459. Edinburgh, 2003.

[Che99] Chen, S. F. and Goodman, J.
An Empirical Study of Smoothing Techniques for Language Modeling.
Computer Speech & Language, vol. 13:359–394, 1999.

References II

[Dol97] Dolfing, J. G. A. and Haeb-Umbach, R.
Signal Representations for Hidden Markov Model Based On-Line Handwriting Recognition.
In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. IV, pages 3385–3388. München, 1997.

[Dur98] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G.
Biological sequence analysis: Probabilistic models of proteins and nucleic acids.
Cambridge University Press, 1998.

[Fin99] Fink, G. A.
Developing HMM-based Recognizers with ESMERALDA.
In V. Matoušek, P. Mautner, J. Ocelíková, and P. Sojka, eds., *Text, Speech and Dialogue*, vol. 1692 of *Lecture Notes in Artificial Intelligence*, pages 229–234. Springer, Berlin Heidelberg, 1999.

References III

[Fin07a] Fink, G. A. and Plötz, T.
ESMERALDA: A Development Environment for HMM-Based Pattern Recognition Systems.
In *7th Open German/Russian Workshop on Pattern Recognition and Image Understanding*. Ettlingen, Germany, 2007.

[Fin07b] Fink, G. A. and Plötz, T.
On the Use of Context-Dependent Modeling Units for HMM-Based Offline Handwriting Recognition.
In *Proc. Int. Conf. on Document Analysis and Recognition*. Curitiba, Brazil, 2007.

[Hua89] Huang, X. and Jack, M.
Semi-Continuous Hidden Markov Models for Speech Signals.
Computer Speech & Language, vol. 3(3):239–251, 1989.

[Jae01] Jaeger, S., Manke, S., Reichert, J., and Waibel, A.
Online Handwriting Recognition: The NPen++ Recognizer.
Int. Journal on Document Analysis and Recognition, vol. 3:169–180, 2001.

References IV

[Jäh05] Jähne, B.
Digital Image Processing.
Springer, Berlin, Heidelberg, New York, 6 edn., 2005.

[Low76] Lowerre, B.
The Harpy Speech Recognition System.
Carnegie-Mellon University, 1976.

[Mer88] Mercer, R.
Language Modeling.
In *IEEE Workshop on Speech Recognition*. Arden House, Harriman, NY, 1988.

[Sch96] Schwartz, R., LaPre, C., Makhoul, J., Raphael, C., and Zhao, Y.
Language-Independent OCR Using a Continuous Speech Recognition System.
In *Proc. Int. Conf. on Pattern Recognition*, vol. 3, pages 99–103. Vienna, Austria, 1996.

References V

[Tri95] Trier, O. D. and Taxt, T.
Evaluation of Binarization Methods for Document Images.
IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 17(3):312–315, 1995.

[Vin04] Vinciarelli, A., Bengio, S., and Bunke, H.
Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models.
IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 26(6):709–720, 2004.

[Wie02] Wienecke, M., Fink, G. A., and Sagerer, G.
Experiments in Unconstrained Offline Handwritten Text Recognition.
In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*. IEEE, Niagara on the Lake, Canada, August 2002.

[Wie05] Wienecke, M., Fink, G. A., and Sagerer, G.
Toward Automatic Video-based Whiteboard Reading.
Int. Journal on Document Analysis and Recognition, vol. 7(2–3):188–200, 2005.

References VI

[Zim06] Zimmermann, M., Chappelier, J.-C., and Bunke, H.
Offline Grammar-Based Recognition of Handwritten Sentences.
IEEE Trans. on Pattern Analysis and Machine Intelligence,
vol. 28(5):818–821, 2006.