Markov Models for Handwriting Recognition

— ICFHR 2010 Tutorial, Kolkata, India—

Gernot A. Fink
TU Dortmund University, Dortmund, Germany
November 15, 2010

▶ 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
- Several different *principles* exist for writing down speech
- *Numerous* different writing systems developed over the centuries
- **Focus here:** Alphabetic writing systems
  ⇒ Especially / most well known: Roman script

**Handwriting:** In contrast to characters created *by machines*
- Most “natural” form of script in almost all cultures
- Typeface adapted for manual creation ⇒ Cursive script
- “Printed letters” as imitation of machine printed characters
- **Frequently:** Free combination, i.e. mixing of both styles

[unconstrained handwriting]
Machine Printed vs. Handwritten Script

**Must Attend Tutorials!!!**
http://www.isical.ac.in/icfhr2010/Tutorials.html

**Venue:** Indian Statistical Institute, Kolkata, India

**Date:** 15 November 2010

A common platform where leading researchers and engineers from the fields of on-line and off-line handwriting recognition will share knowledge and experience.

### On the Occasion of

**ICFHR 2010**

**12th International Conference on Frontiers in Handwriting Recognition**

16th-18th November, 2010 | Kolkata, India

Handwritten Text / Word Recognition Systems - Conception, Approaches and Evaluation by H. E. Abed and V. Margner

Sequence Modeling: From Hidden Markov Models to Structured Output Prediction by T. Artieres

Markov Models for Handwriting Recognition by G. Fink

Signature Verification - Forensic Examiners' Perception and Solutions for Off-line and On-line Signatures by M. Liwicki, Michael Blumenstein, Elisa van den Heuvel and Bryan Found

Information Retrieval from Handwritten Documents by A. O. Thomas and A. Bhardwaj

Multimodal Computer Assisted Transcription of Handwriting Images by A. H. Toselli, M. Pastor and V. Romero

Handwritten Text / Word Recognition Systems - Conception, Approaches and Evaluation by H. E. Abed and V. Margner

Sequence Modeling: From Hidden Markov Models to Structured Output Prediction by T. Artieres

Markov Models for Handwriting Recognition by G. Fink

Signature Verification - Forensic Examiners' Perception and Solutions for Off-line and On-line Signatures by M. Liwicki, Michael Blumenstein, Elisa van den Heuvel and Bryan Found

Information Retrieval from Handwritten Documents by A. O. Thomas and A. Bhardwaj

Multimodal Computer Assisted Transcription of Handwriting Images by A. H. Toselli, M. Pastor and V. Romero

**Seats are limited!! Register Early**

**Tutorial Fees (for other than resident Indians):**

<table>
<thead>
<tr>
<th></th>
<th>Early-Bird Registration</th>
<th>On-Site Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPR Member</td>
<td>$100 USD</td>
<td>$125 USD</td>
</tr>
<tr>
<td>Non-Member</td>
<td>$125 USD</td>
<td>$150 USD</td>
</tr>
<tr>
<td>Student (Full time)</td>
<td>$75 USD</td>
<td>$100 USD</td>
</tr>
</tbody>
</table>

**Tutorial Fees (for resident Indians):**

<table>
<thead>
<tr>
<th></th>
<th>Early-Bird Registration</th>
<th>On-Site Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPR Member</td>
<td>₹3500</td>
<td>₹4000</td>
</tr>
<tr>
<td>Non-Member</td>
<td>₹4000</td>
<td>₹4250</td>
</tr>
<tr>
<td>Student (Full time)</td>
<td>₹2500</td>
<td>₹2500</td>
</tr>
</tbody>
</table>

* Deserving students may apply for full/partial waiver of the registration fees.

**Tutorial registration fees include:**

- Access to particular tutorial session(s)
- Name badge
- Writing pad and pen
- Tea breaks
- 1 working lunch

**ICFHR 2010 Secretariat:**
Indian Statistical Institute, 203, B. T. Road, Kolkata – 700108, INDIA
Phone: +91 33 25781832, +91 33 25752850, +91 33 25752852
Email: icfhr2010@isical.ac.in

**http://www.isical.ac.in/~icfhr2010/Tutorials.html**

**Venue:** Indian Statistical Institute, Kolkata, India

**Date:** 15 November 2010

On the Occasion of

Fink

Markov Models for Handwriting Recognition

Motivation Theory Practice Systems Summary References
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)
▶ \textit{Off-line} analysis of resulting “image data”
⇒ “Optical Character Recognition” (OCR)

Additionally: Human-Machine-Interaction (HMI)

Basic principle:
▶ Capture pen trajectory \textit{during} writing (using specialized sensors & pens)
▶ \textit{On-line} analysis of movement data
Why is Handwriting Recognition Difficult?

- High variability of individual characters
  - Writing style

  The effect of bottom congestion due to the story, and the marriage of the central throughout in terms of the cinema, and again and

- Stroke width and quality
- Size of the writing
- Variation even for single writer!

- Segmentation is of cursive script problematic
  ⇒ “Merging” of adjacent characters
Focus of this Tutorial

Input data:
- Handwriting data captured using 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. ⇒ Unconstrained handwriting!

Processing type:
- **Focus:** Offline-Processing ⇒ Analysis of handwriting data after completion of capturing
- Also (to some limited extent): Online-Techniques

Recognition approach: Stochastic modeling of handwriting
- Hidden Markov Models for segmentation free recognition
- Statistical $n$-gram models for text-level restrictions
Recognition Paradigm – “Traditional” OCR

Segmentation + Classification:

- Segment-wise classification possible using 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

\[
\hat{w} = \arg\max_w P(w|X) = \arg\max_w \frac{P(w)P(X|w)}{P(X)} = \arg\max_w P(w)P(X|w)
\]

Wanted: Sequence of words/characters \( \hat{w} \), which is most probable for given signal/features \( X \)
Recognition Paradigm – II

\[ \hat{w} = \arg\max_w P(w|X) = \arg\max_w \frac{P(w)P(X|w)}{P(X)} = \arg\max_w P(w)P(X|w) \]

Two aspects of modeling:

▶ Script (appearance) model: \( P(X|w) \) ⇒ Representation of words/characters

\textit{Hidden-Markov-Models}

▶ Language model: \( P(w) \) ⇒ Restrictions for sequences of words/characters

\textit{Markov Chain Models / n-Gram-Models}

Specialty: Script or trajectories of the pen (or features, respectively) interpreted as \textit{temporal} data

✓ Segmentation performed implicitly! ⇒ “segmentation free” approach

✗ 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 on immediate predecessor state (\( \hat{=} \) first order)

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

2. Stage: Depending on current state \( s_t \) for every point in time additionally an output \( O_t \) is generated

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

**Caution:** Only outputs can be observed \( \rightarrow \) hidden
Hidden-Markov-Models: Formal Definition

A Hidden-Markov-Model $\lambda$ of \textit{first order} is defined as:

- a finite set of states:
  \[
  \{ s | 1 \leq s \leq N \}
  \]

- a matrix of state transition probabilities:
  \[
  A = \{ a_{ij} | a_{ij} = P(s_t = j | s_{t-1} = i) \}
  \]

- a vector of start probabilities:
  \[
  \pi = \{ \pi_i | \pi_i = P(s_1 = i) \}
  \]

- state specific output probability distributions:
  \[
  B = \{ b_{jk} | b_{jk} = P(O_t = o_k | s_t = j) \} \text{ (discrete case)}
  \]
  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

Background: Casino occasionally exchanging dice: fair ⇔ loaded
⇒ Model with two states: $S_{\text{fair}}$ and $S_{\text{loaded}}$

Exclusive observations: Results of the rolls
⇒ Underlying state-sequence remains hidden!

Question: Which die has been used, i.e. when is the casino cheating?
⇒ Probabilistic inference about internal state-sequence using stochastic model
Modeling of Outputs

Discrete inventory of symbols: Very limited application fields

✓ Suited for discrete data only (e.g. DNA)

✝ Inappropriate for non-discrete data – use of vector quantizer required!

Continuous modeling: Standard for most pattern recognition applications processing sensor 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

\[
p(x) \doteq \sum_{k=1}^{\infty} c_k \mathcal{N}(x | \mu_k, C_k)
\]

\[
\approx \sum_{k=1}^{M} c_k \mathcal{N}(x | \mu_k, C_k)
\]
Modeling of Outputs – 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

Note: In HMMs integrated with general parameter estimation
Usage Concepts for Hidden-Markov-Models

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

Scoring: How well does the model describe some pattern?
→ Computation of the production probability \( P(O|\lambda) \)

Decoding: What is the “internal structure” of the model? (≡ “Recognition”)
→ Computation of the optimal state sequence
\[
s^* = \arg\max_s P(O, s|\lambda)
\]

Training: How to determine the “optimal” model?
~~ Improvement of a given model \( \lambda \) with \( P(O|\hat{\lambda}) \geq P(O|\lambda) \)
The Production Probability

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

Widely used measure: Production probability $P(O|\lambda)$ that observation sequence $O$ was generated by model $\lambda$ – along an arbitrary state sequence
The Production Probability: Naive Computation

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

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

2. Probability that a given model $\lambda$ runs through arbitrary state sequence:

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

3. (1) + (2): Probability that $\lambda$ generates $O$ along certain state sequence $s$:

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

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

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

$\therefore$ Complexity: $O(TN^T)$
The Production Probability: The Forward-Algorithm

More efficient: Exploitation of the Markov-property, i.e. the “finite memory”
⇒ “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. \( P(O|\lambda) = \sum_{i=1}^N \alpha_T(i) \)

✓ Complexity: \( O(TN^2) \)!
(vs. \( O(TN^T) \) from naive computation)

Later: Backward-Algorithm \[ \text{Training} \]
The “optimal” Production Probability

Total production probability: Consider all paths through model

- Mathematically exact computation of \( P(O|\lambda) \)
- Computationally rather costly ([\( \uparrow \) score computations])

Observation: \( P(O|\lambda) \) is dominated by contribution on optimal path
⇒ Use only approximate solution

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

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

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. \( P^*(O|\lambda) = P(O, s^*|\lambda) = \max_i \delta_T(i) \)
Decoding

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

⇒ (Probabilistic) Inference of optimal state sequence $s^*$ necessary

Maximization of posterior probability:

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

Bayes’ rule:

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

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

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

Computation of $s^*$: Viterbi-Algorithm

Fink
Markov Models for Handwriting Recognition
The Viterbi Algorithm

... inductive procedure for efficient computation of \( s^* \) 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. \( \delta_1(i) := \pi_i b_i(O_1) \)

2. \( \delta_{t+1}(j) := \max_i (\delta_t(i) a_{ij}) b_j(O_{t+1}) \)

3. \( P^*(O | \lambda) = P(O, s^* | \lambda) = \max_i \delta_T(i) \)
   \[ s_T^* := \arg\max_j \delta_T(j) \]

4. Back-tracking of optimal path:
   \( s_t^* = \psi_{t+1}(s_{t+1}^*) \)

✓ Implicit segmentation
✓ Linear complexity in time
✗ Quadratic complexity w.r.t. \#states

\[ P^*(O | \lambda) = P(O, s^* | \lambda) \]
Toy Example: The Occasionally Dishonest Casino – II

Parameters of the given HMM $\lambda$:

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

Wanted: Internal state-sequence for segmentation into fair use and cheating
$\Rightarrow$ Viterbi-Algorithm
Toy Example: The Occasionally Dishonest Casino – III

\[ \pi_i = \frac{1}{2}, \quad A = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix} \]

\[ B = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{pmatrix} \]

\[ O = 1, 1, 2, 6, 6, 6, 3, 5 \]
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 $\lambda$

$\Rightarrow$ Optimized model $\hat{\lambda}$ better suited for given sample data

General procedure: Parameters of $\lambda$ 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 $P(S_t = i|O, \lambda)$ for model assessment

Procedure: Exploit limited memory for
- History – forward-probability $\alpha_t(i)$ \(\text{[forward-algorithm]}\), and
- Rest of partial observation sequence – backward-probability $\beta_t(i)$

Backward-Algorithm:
Let

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

1. $\beta_T(i) := 1$

2. For all times $t$, $t = T - 1 \ldots 1$:
   $$\beta_t(i) := \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

3. $P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_1(i)$
The Forward-Backward Algorithm

... for efficient computation of posterior state probability

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

\[
P(S_t = i, O | \lambda) = P(O_1, O_2, \ldots O_t, S_t = i | \lambda) P(O_{t+1}, O_{t+2}, \ldots O_T | S_t = i, \lambda)
\]
\[
= \alpha_t(i) \beta_t(i)
\]

\[
\Rightarrow \gamma_t(i) = P(S_t = i | O, \lambda) = \frac{\alpha_t(i) \beta_t(i)}{P(O | \lambda)}
\]
**Parameter Training using the Baum-Welch-Algorithm**

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

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

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

**Definitions:** (of quantities based on forward- and backward variables)

$\Rightarrow$ Allow (statistical) inferences about internal processes of $\lambda$ when generating $O$

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

$$\gamma_t(i) = \frac{P(S_t = i|O, \lambda)}{P(O|\lambda)} = \sum_{j=1}^{N} P(S_t = i, S_{t+1} = j|O, \lambda) = \sum_{j=1}^{N} \gamma_t(i,j)$$
The Baum-Welch-Algorithm

Let

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

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

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

1. Choose a suitable initial model \( \lambda = (\pi, A, B) \) with initial estimates \( (\pi_i, a_{ij}, c_{jk} \text{ for mixtures } g_{jk}(x) = N(x | \mu_{jk}, C_{jk}) \text{ for pdf. } b_{jk}(x) = \sum_k c_{jk} g_{jk}(x).) \)

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

\[ \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \]

\[ \hat{\pi}_i = \gamma_1(i) \]

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

\[ \hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k) x_t}{\sum_{t=1}^{T} \xi_t(j,k)} \]

\[ \hat{c}_{jk} = \frac{\sum_{t=1}^{T} \xi_t(j,k) x_t x_t^T}{\sum_{t=1}^{T} \xi_t(j,k)} - \hat{\mu}_{jk} \hat{\mu}_{jk}^T \]

3. if \( P(O | \hat{\lambda}) \) was considerably improved by the updated model \( \hat{\lambda} \) w.r.t. \( \lambda \)

let \( \lambda \leftarrow \hat{\lambda} \) and continue with step 2

otherwise Stop!
Multiple Observation Sequences

In general: Sample sets used for parameter training are usually structured, i.e. subdivided into individual segments (documents, lines; in speech: utterances)

So far: Segments considered as individual observation sequences

Goal: Estimate model parameters suitable for describing all data in the sample set

Procedure: Accumulate statistics gathered for parameter estimates across all observation sequences considered

Example:

$$\hat{\mu}_{jk} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T} \xi^l_t(j, k) x^l_t}{\sum_{l=1}^{L} \sum_{t=1}^{T} \xi^l_t(j, k)}$$

Where $x^l_t$ denotes the $t$-th element of data segment number $l$
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
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
⇒ 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 $P(w)$ for given word sequence $w = w_1, w_2, \ldots, w_k$

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

**Method:** Factorization of $P(w)$ applying Bayes’ rule according to

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

**Problem:** Context dependency increases arbitrarily with length of symbol sequence

$\Rightarrow$ Limit length of the “history”

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

$n$ 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)
\textit{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: \textit{How well does the model represent certain data?}

Basis: Probability of a symbol sequence assigned by the model

Model Creation: \textit{How to create a good model?}

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

\textbullet More sophisticated methods necessary in practice! [↑ \textit{parameter estimation}]
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: $P(z|y)$
  ($P(z|y)$ or $P(z|xy)$ for bi- and tri-gram models)

- Absolute frequency of an $n$-gram: $c(\mathbf{yz})$

- 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* \( \mathcal{P} \)

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

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

\[
H(p|q) = - \sum_i p_i \log_2 q_i \quad \overset{(\text{data model})}{\rightarrow} \quad - \sum_t \frac{1}{T} \log_2 P(w_t|\ldots) = - \frac{1}{T} \log_2 \prod_t P(w_t|\ldots)
\]

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

**Question:** *How can perplexity be interpreted?*
Intuitive interpretation of perplexity:

- Assume: Text $w_1, w_2, \ldots w_t, \ldots w_T$ 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
  - No prediction possible
  - All symbols equally likely: $P(w_t|\ldots) = \frac{1}{|V|}$

Example:

```
I don't mind . . .
```

```
I don't mind . . .
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```

```
I don't mind
```
n-Gram Models: Interpretation of Perplexity II

- Worst case situation: All symbols equally likely
  - Prediction according to uniform distribution \( P(w_t|...) = \frac{1}{|V|} \)
  - Perplexity of texts generated:
    \[
    P(w) = \left\{ \left( \frac{1}{|V|} \right)^T \right\}^{-\frac{1}{T}} = |V|
    \]

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**

**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
- Calculate conditional probabilities

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

**Example:**

- \( c(\text{you say}) = 1 \)
- \( c(\text{you}) = 3 \)

**Problem:** Many \( n \)-grams are **not** observed

- \( c(w_1 \ldots w_n) = 0 \) \( \Rightarrow \) \( P(w_n|\ldots) = 0 \)
- \( c(\text{you don’t}) = 0 \)
- \( P(\ldots w_1 \cdot w_n \ldots) = 0! \)
- \( P(\text{I take it if you don’t mind}) = 0 \)
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* $P(z|y)$ for $yz$ with $c(yz) = 0$

Question: *What estimates are “useful”?*

- small probabilities!, smaller than seen events? $\rightarrow$ 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 ($\hat{=} \text{smoothing}$ of empirical distribution)

Question: *What distribution is suitable for events we know nothing about?*
**n-Gram Models: Parameter Estimation III**

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

Zero probability $\rightarrow$ Incorporate more general distribution
**n-Gram Models: Discounting**

**Gathering of Probability Mass**

**Calculate** modified frequency distribution \( f^*(z|y) \) for seen \( n \)-grams \( yz \):

\[
f^*(z|y) = \frac{c^*(yz)}{c(y)} = \frac{c(yz) - \beta(yz)}{c(y^\cdot)}
\]

Zero-probability \( \lambda(y) \) for history \( y \): Sum of “collected” counts

\[
\lambda(y) = \sum_{z:c(yz)>0} \beta(yz) / c(y^\cdot)
\]

**Choices** for discounting factor \( \beta() \):

- proportional to \( n \)-gram count: \( \beta(yz) = \alpha c(yz) \) \( \Rightarrow \) *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 \( P(z|\hat{y}) \) associated with \( n \)-gram model \( P(z|y) \)

- Generalization \( \triangleq \) shortening the context/history
  
  \[ y = y_1, y_2, \ldots y_{n-1} \rightarrow \hat{y} = y_2, \ldots y_{n-1} \]

- More general distribution obtained:
  
  \[ q(z|y) = q(z|y_1, y_2, \ldots y_{n-1}) \leftarrow P(z|y_2, \ldots y_{n-1}) = P(z|\hat{y}) \]

  (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 $f^*(\cdot|\cdot)$):

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

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

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

Estimates obtained:

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

Properties:

- Assumes that estimates *always* benefit from smoothing

⇒ 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

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

Normalization factor $K_y$ ensures that: $\sum_z P(z|y) = 1$

$$ K_y = \frac{1}{\sum_{yz : c^*(yz)=0} q(yz)} $$

**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 $q(z|y)$ more general distribution is again $n$-gram model $\Rightarrow$ principle can be applied recursively

**Example** for backing off and tri-gram model:

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

**Note:** Combination of absolute discounting and backing off creates powerful $n$-gram models for a wide range of applications (cf. [Che99]).
**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
$P(z|xy) = f^*(z|xy)$

$P(x|xy) = \lambda(xy) K_{xy} f^*(x|y)$
**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!)
  - \( p(x) \approx 10^{-5} \) on average

- Further consider a text fragment of 100 frames length
  (will in general be only a few characters long)

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

Note: Extremely coarse approximations, neglecting

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

\[ \Rightarrow \textit{Quantities can't be represented on today's computers!} \]

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 $\in [0.0 \ldots 1.0] \rightarrow$ costs $\in [0.0 \ldots +\infty]$
(Note: Densities $>1.0$ possible but rare $\rightarrow <0.0$)

\[
\begin{align*}
\begin{array}{c}
a_{ij} \\
b_j(x) \\
1.0 \ldots e^{-1000}
\end{array}
& \rightarrow \\
\begin{array}{c}
- \log a_{ij} \\
- \log b_j(x) \\
0.0 \ldots 1000
\end{array}
\end{align*}
\]

\[
\begin{align*}
\text{product} \\
\text{maximum}
\end{align*}
& \rightarrow \\
\begin{align*}
\text{sum} \\
\text{minimum}
\end{align*}
\]

\[
\begin{align*}
\text{sum} \\
\text{product}
\end{align*}
\]

$\ldots$ a bit more complicated
Probabilities: Logarithmic Representation II

**Question:** Could product/maximum ($\Rightarrow$ sum/minimum) be sufficient for HMMs?

Mostly, but not quite for everything!

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

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

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

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

‡ Baum-Welch training: Alternative paths *must* be considered

⇒ 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:
\[ \tilde{p}_1 + \log \tilde{p}_2 = -\ln \left( e^{-\tilde{p}_1} + e^{-\tilde{p}_2} \right) \]

\[ \Downarrow \] Advantage of logarithmic representation is lost!

Summation in the logarithmic domain: Kingsbury-Rayner formula

\[ \tilde{p}_1 + \log \tilde{p}_2 = -\ln(p_1 + p_2) = -\ln(p_1(1 + \frac{p_2}{p_1})) = \]

\[ = -\left\{ \ln p_1 + \ln(1 + e^{\ln p_2 - \ln p_1}) \right\} \]

\[ = \tilde{p}_1 - \ln(1 + e^{-(\tilde{p}_2 - \tilde{p}_1)}) \]

\[ \checkmark \] 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
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
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

⇒ 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)
- Rarely used for handwriting recognition, so far (cf. [Kos97, Fin07b, Fin10])

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

- 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 $d$:
   
   $$(C_j, C_k) = \arg\min_{C_p, C_q \in \mathcal{C}} d(C_p, C_q)$$

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
  ⇒ Individual components of mixtures are tied across distributions
- New definition of output probabilities:
  \[
  b_j(x) = \sum_{k=1}^{M} c_{jk} \mathcal{N}(x|\mu_k, \Sigma_k)
  = \sum_{k=1}^{M} c_{jk} g_k(x)
  \]

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

- Exemplary distance measure used for clustering procedure \[\Rightarrow State\ Tying\]
  \[
  d(C_i, C_j) = \frac{N_i N_j}{N_i + N_j} ||\mu_i - \mu_j||
  \]
  \(N_i, N_j:\#\) samples assigned to \(i, j\)-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 $s^*$ for $O$ given $\lambda$ [↗ 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 $X(j)$:

   1. Compute VQ codebook $Y = \{y_1, \ldots, y_{M_j}\}$ and partition $\{R_1, \ldots, R_{M_j}\}$
   2. Compute updated parameters of output distributions:

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

3. Termination: if $P^*(O|\hat{\lambda})$ was improved by $\hat{\lambda}$ w.r.t. $\lambda$: 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 focussing search to “beam” around best partial solution [Low76]

- Evaluation of $\delta_t(i)$ on limited set of active states:
  $$\mathcal{A}_t = \{ i | \delta_t(i) \geq B \delta_t^* \}$$
  with $\delta_t^* = \max_j \delta_t(j)$ and $0 < B \ll 1$
  $$\delta_{t+1}(j) = \max_{i \in \mathcal{A}_t} \{ \delta_t(i)a_{ij} \} b_j(O_{t+1})$$
  $$B = [10^{-10} \ldots 10^{-20}]$$ typically

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

$\delta_t^*$: Viterbi−Path
$\delta_t(i)$: active HMM state
$\delta_t^*: \lambda^*, O^*$
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
      \( \tilde{\delta}_t(i, j) < \tilde{\delta}_t(j) \) : \( \tilde{\delta}_t(j) \leftarrow \tilde{\delta}_t(i, j), \psi_t(j) \leftarrow i \)
    - Update locally optimal path score
      if \( \tilde{\delta}_t(i, j) < \tilde{\delta}_t^* \) : \( \tilde{\delta}_t^* \leftarrow \tilde{\delta}_t(i, j) \)
  - 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 \( \tilde{\delta}_t(j) \leq \tilde{\delta}_t^* + \tilde{B} \) : \( \mathcal{A}_t \leftarrow \mathcal{A}_t \cup \{j\} \)

Termination: Determine optimal end state \( \hat{s}_T^* := \arg\min_{j \in \mathcal{A}_T} \tilde{\delta}_T(j) \)

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 $\alpha_t^* = \max_j \alpha_t(j) / \beta_t^* = \max_j \beta_t(j)$

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

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

- Modified recursive computation rules for forward and backward variables:

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

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

- $\gamma_t(i)$ 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:

\[
P(w) \xrightarrow{\text{w}} P(X|w) \xrightarrow{\text{X}} \text{argmax}_w P(w|X)
\]

⇒ HMMs + n-gram models \textit{frequently} used in combination!

Problems in practice:

- \textit{How to compute a combined score?}  
  Channel model defines basis only!

- \textit{When to compute the score?}  
  Model valid for \textit{complete} HMM results!

- \textit{How does the language model improve results?}

⚠️ \textit{Why not use HMMs only to avoid those problems?}
Integrated Search: Basics

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

⇒ Weighted combination using “linguistic matching factor” $\rho$

$$P(w)^\rho P(X|w)$$

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.

⇒ No benefit for HMM search!

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

✓ Achieved by using $P(z|y)$ 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]

<table>
<thead>
<tr>
<th></th>
<th>none</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM-DB</td>
<td>29.2</td>
<td>22.1</td>
<td>18.3</td>
<td>16.1</td>
<td>15.6</td>
</tr>
<tr>
<td>CER/$\sqrt{P}$</td>
<td>n.a.</td>
<td>6.2</td>
<td>6.0</td>
<td>6.0</td>
<td>5.8</td>
</tr>
</tbody>
</table>

Note: Important plausibility check: If violated, something strange is happening!
Integrated Search: HMM Networks

- Straight-forward extension of HMM-only models
- \(n\)-gram scores used as transition probabilities between words

\(\forall\) 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!

$$P(a|a)$$

$$P(a|a a)$$

$$P(a|a a a)$$

$$P(a|a b)$$

$$P(a|a c)$$

$$P(a|c)$$

$$P(a|c c)$$

$$P(b|a)$$

$$P(b|a a)$$

$$P(b|a a a)$$

$$P(b|a b)$$

$$P(b|a c)$$

$$P(b|a c c)$$

$$P(b|c)$$

$$P(b|c a)$$

$$P(b|c b)$$

$$P(b|c c)$$

$$P(c|a)$$

$$P(c|a a)$$

$$P(c|a a a)$$

$$P(c|a b)$$

$$P(c|a c)$$

$$P(c|a c c)$$

$$P(c|c)$$

$$P(c|c a)$$

$$P(c|c b)$$

$$P(c|c c)$$
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

   ⇒ existing solutions sorted differently!

3. $n + k$-gram

   ... continue with 2.

hypothesis rank according to combined score

- handwriting recognition is difficult
- handwriting recognition is different
- handwriting recognition is easy

Handwriting recognition is different...
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
  Remove unwanted variation
  for processing by sequential model
  Determine relevant “properties”

  ▶ Preprocessing
  ▶ Linearization
  ▶ Feature Extraction
  ▶ 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
  - No isolated characters! ➞ 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 focussed today (non-European scripts [e.g. Arabic, Bangla], special tasks, ...)

Note: Hardly any “standard” procedures exist! (as opposed to speech recognition)
Putting It All Together: Introduction II

Architecture of typical HMM-based system for (offline) handwriting 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)

for the curtain to rise on the Commonwealth

what, in fact, can the other Commonwealth countries

Results of size normalization (avg. distance of contour minima)
Preprocessing: Offline V

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

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

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

Elimination of delayed strokes
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!
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 pioneered 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 $\alpha$ 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
- No holistic features used in HMM-based systems

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
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

- Task models constructed by concatenation etc. from elementary HMMs

- Initialization
  - If you start from scratch: Manual labeling of (part of) sample set required!
    Note: Quasi-uniform initialization possible in semi-continuous models!
  - 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]
  ⇒ “unlimited” / open vocabulary systems

Decoding:

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

▶ 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
Summary

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

Difficult automatic recognition:
- High variability even within particular characters (size, style, line width etc.)
- Problematic segmentation due to “merging” of adjacent characters.

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

\[ \hat{w} = \arg\max_w P(w|X) = \arg\max_w \frac{P(w)P(X|w)}{P(X)} = \arg\max_w P(w)P(X|w) \]

Script (appearance) model: \( P(X|w) \)

- Two-stage stochastic process
- Hidden state sequence, observable outputs
- Output modeling: Discrete, or continuous
- Algorithms for scoring, decoding, training

Language model: \( P(w) \)

- Probability distribution over symbol sequences (characters / words)
- Principally: Smoothing of direct parameter estimates (i.e. relative frequencies of events)
- Algorithms for scoring / evaluation

⇒ Hidden-Markov-Models

⇒ n-Gram-Models

\[ P(w) \approx \prod_{t=1}^{T} P(w_t | w_{t-n+1}, \ldots, w_{t-1}) \]

\( n \) symbols
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 ✓

- **Integrated search / model decoding**
  - (combination of writing and language model)
  - ⇒ search space copies, multi-pass search / rescoring
Further Reading


- Inspection copy available!
- Conference discount: 20%!


- Open access publication!
Further Developing: ESMERALDA

Framework for developing HMM-based pattern recognition systems

Developed/Maintained at TU Dortmund, Germany (cf. [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 1

[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*,

[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

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

[Che99] Chen, S. F. and Goodman, J.
References II

[Dol97] Dolfing, J. G. A. and Haeb-Umbach, R.
Signal Representations for Hidden Markov Model Based On-Line Handwriting Recognition.

Biological sequence analysis: Probabilistic models of proteins and nucleic acids.

[Fin99] Fink, G. A.
Developing HMM-based Recognizers with ESMERALDA.
[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.

[Fin08] Fink, G. A.
*Markov Models for Pattern Recognition*.
Online Bangla Word Recognition Using Sub-Stroke Level Features and 
Hidden Markov Models. 
In Proc. Int. Conf. on Frontiers in Handwriting Recognition. Kolkata, 
India, 2010.

[Hua89] Huang, X. and Jack, M. 

[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, 

[Jäh05] Jähne, B. 
Digital Image Processing. 

[Low76] Lowerre, B.
*The Harpy Speech Recognition System*.

[Mer88] Mercer, R.
Language Modeling.

[Plö09] Plötz, T. and Fink, G. A.
References VI


References VII

