technische universität
dortmund

Markov Models for Handwriting Recognition

— DAS 2012 Tutorial, Gold Coast, Australia —

Gernot A. Fink

TU Dortmund University, Dortmund, Germany

March 26, 2012

Why Should Machines Be Able to Read?

Because it's *cool*? … but probably not cool enough!

For Automation in document processing, e.g.:

- Reading of addresses,
- Analysis of forms,
- Classification of business mail pieces
- Archiving & retrieval

For Communication with humans ($\hat{=}$ Man-Machine-Interaction) on small, portable devices (e.g. SmartPhones, Tablet-PCs)

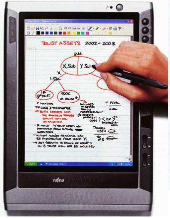
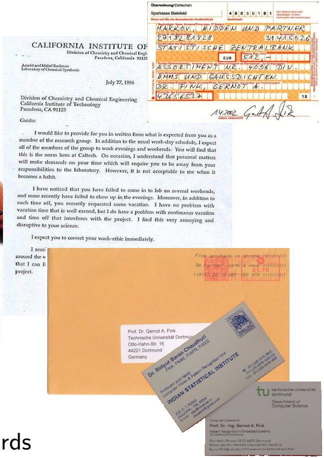
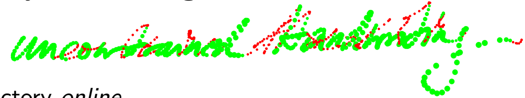


Photo: Fujitsu Ltd.



As Support in, e.g., automatically reading business cards

Why Handwriting?



In Communication:

Interactivity required!

⇒ Capturing of the pen trajectory *online*

In Automatition:

Capturing of the script image *offline*

- Postal addresses:
  10–20% handwritten
  (more before Christmas,
  trend: increasing!)

  [Source: M.-P. Schambach, Siemens]
- Forms
  (Money-transfers, checks, ...)
- Historical documents
  (Letters, reports
  from the adminitration)

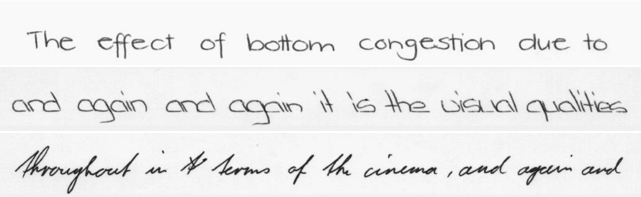
⇒ Handwriting — *still going strong*!

Why is Handwriting Recognition Difficult?

- Considerable freedom in the script appearance

  Typical handwriting ≙ cursive writing
  Also: "hand printed" characters
  Mostly: Combination ≙ *unconstrained* ...

- Large Variability of individual symbols
  - Writing style



  - Stroke width and quality
  - Considerable variations even for the same writer!

- Segmentiation problematic (especially for cursive writing)
  "Merging" of neighboring symbols

Focus of this Tutorial

Processing type: Offline (documents captured by scanner or camera)

Script type & Writing style:

- Alphabetic scripts, especially Roman script
- *No* restriction w.r.t. writing style, size etc.
  ⇒ Unconstrained handwriting!

Methods: Statistical Recognition Paradigm

- Markov Models for segmentation free recognition
- Statistical *n*-gram models for text-level restrictions

Goal: Understand ...

- ... concepts and methods behind Markov-Model based recognizers *and* ...
- ... how these are applied in handwriting recognition.

With *Self-Study* Materials:

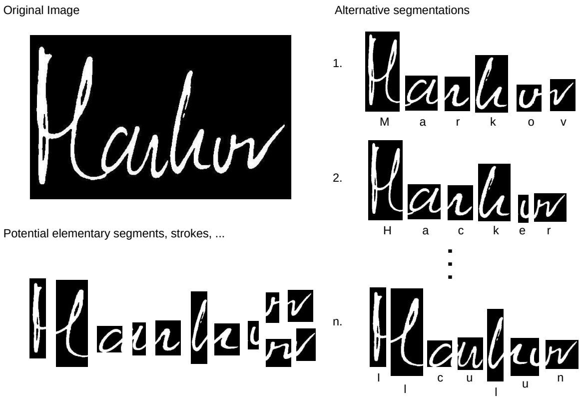
- Build a *working* handwriting recognizer using ESMERALDA.

Overview

- Introduction
- Markov Model-Based Handwriting Recognition . . . *Fundamentals*
  - Motivation . . . *Why MM-based HWR?*
  - Data Peparation . . . *Preprocessing and Feature Extraction*
- Hidden Markov Models . . . *Definition, Use Cases, Algorithms*
- Language Models . . . *Definition & Robust Estimation*
- Integrated Search . . . *Combining HMMs and n-Gram Models*
- Summary . . . *and Further Reading*

"Traditional" Recognition Paradigm

Segmentation + Classification:



✓ Segment-wise classification possible using various standard techniques

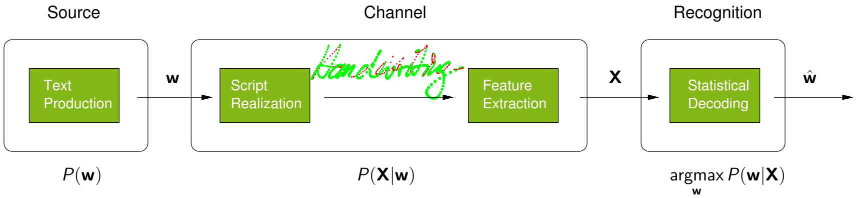
↯ Segmentation is

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

↯ *Segmentation is especially problematic for unconstrained handwriting!*

Statistical Recognition Paradigm: The Channel Model

(Model originally proposed for automatic speech recognition)



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

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

The Channel Model 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 performed implicitly! $\Rightarrow$ "segmentation free" approach

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

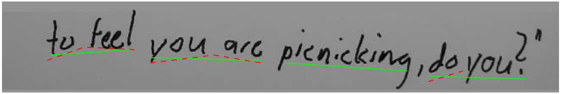
Overview

- Introduction
- Markov Model-Based Handwriting Recognition . . . *Fundamentals*
  - Motivation . . . *Why MM-based HWR?*
  - Data Peparation . . . *Preprocessing and Feature Extraction*
- Hidden Markov Models . . . *Definition, Use Cases, Algorithms*
- Language Models . . . *Definition & Robust Estimation*
- Integrated Search . . . *Combining HMMs and n-Gram Models*
- Summary . . . *and Further Reading*

Preprocessing I

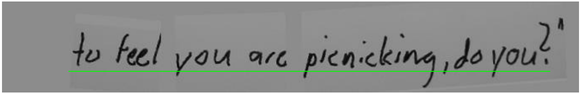
Assumption: Documents are already segmented into text lines
(Text detection and line extraction *highly* application specific!)

Baseline Estimation:



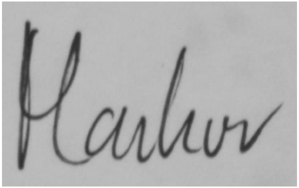
Potential method:
- Initial estimate based on horiz. projection histogram
- Iterative refinement and outlier removal (cf. [2, 10])

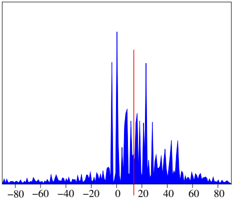
Skew and Displacement Correction:



Preprocessing II

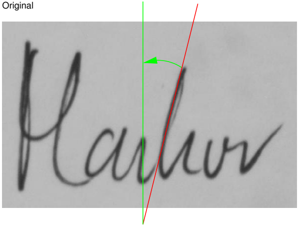
Slant estimation: E.g. via mean orientation of edges obtained by Canny operator (cf. e.g. [12])



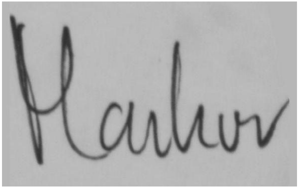


Slant normalization (by applying a shear transform)

Original



Corrected Slant



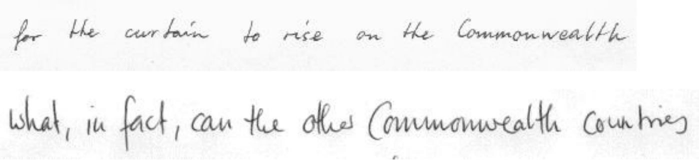
Preprocessing III

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

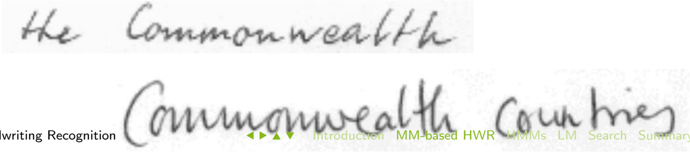
Size normalization methods:

- "manually", heuristically, to predefined width/height???
- depending on estimated core size (← estimation crucial!)
- depending on estimated character width [7]

Original text lines (from IAM–DB)



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



Serialization: The Sliding Window Method

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

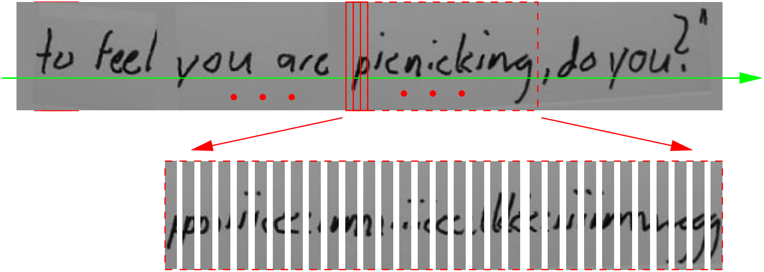
↯ No chronological structure inherently defined!

Exception: Logical sequence of characters within texts

Solution: Sliding-window approach

First proposed by researchers at Daimler-Benz Research Center, Ulm [3], pioneered by researchers at BBN [11]

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



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

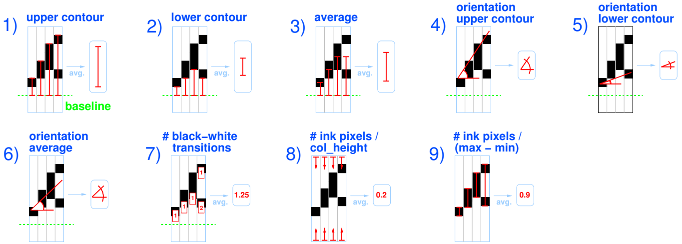
Feature Extraction

Basic Idea: Describe appearance of writing within analysis window

- ↯ No “standard” approaches or feature sets
- STOP No holistic features used in HMM-based systems

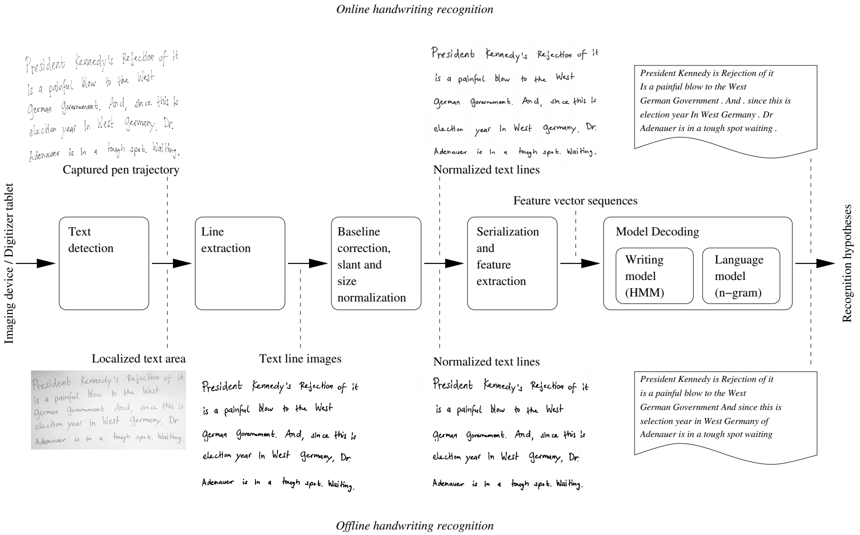
Potential Methods:

- (For OCR) Local analysis of gray-value distributions (cf. e.g. [1])
- Salient elementary geometric shapes (e.g. vertices, cusps)
- Heuristic geometric properties (cf. e.g. [13])



Additionally: Compute dynamic features
(i.e. discrete approximations of temporal derivatives, cf. e.g. [5])

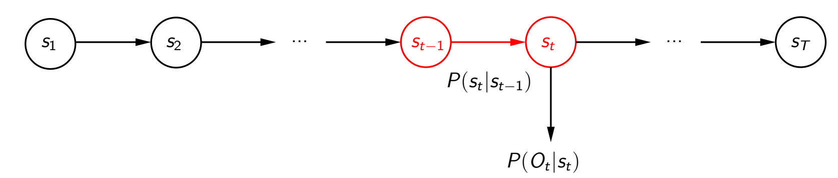
General Architecture



Overview

- Introduction
- Markov Model-Based Handwriting Recognition ... *Fundamentals*
- **Hidden Markov Models** ... *Definition, Use Cases, Algorithms*
- Language Models ... *Definition & Robust Estimation*
- Integrated Search ... *Combining HMMs and n-Gram Models*
- Summary ... *and Further Reading*

Hidden Markov Models: Two-Stage Stochastic Processes



1. Stage: discrete stochastic process ≈ "probabilistic" finite state 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)

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

2. Stage: Output O_t generated for every time t depending on current state s_t

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

   Note: Only outputs can be observed ⇒ hidden Markov model

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 output probability distributions:

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

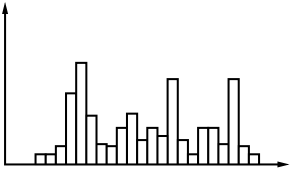
or

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

Modeling of Outputs

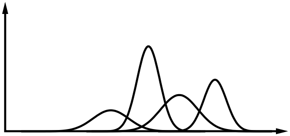
Discrete inventory of symbols: Very limited application fields

- ✓ Suited for discrete data only (e.g. DNA)
- ϟ 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 $\mathbb{R}^n$



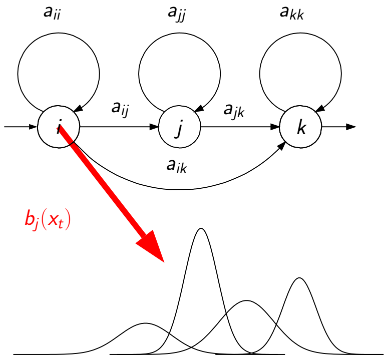
Problem: No general parametric description

Procedure: Approximation using mixture densities

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

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

Modeling of Outputs – II

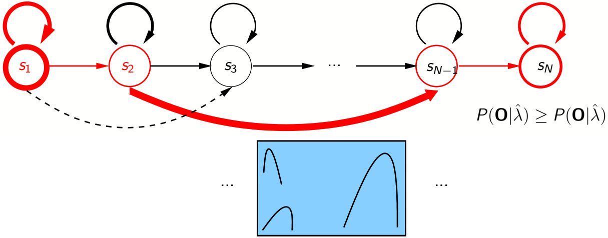


Mixture density modeling:

- Base Distribution?
  $\Rightarrow$ Gaussian Normal densities
- Shape of Distributions (full / diagonal covariances)?
  $\Rightarrow$ Depends on pre-processing of the data (e.g. redundancy reduction)
- Number of mixtures?
  $\Rightarrow$ Clustering (. . . and heuristics)
- Estimation of mixtures?
  $\Rightarrow$ 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?*
$\rightarrow$ Computation of the production probability $P(\mathbf{O}|\lambda)$

Decoding: *What is the "internal structure" of the model?* ($\hat{=}$ "Recognition")
$\rightarrow$ Computation of the optimal state sequence

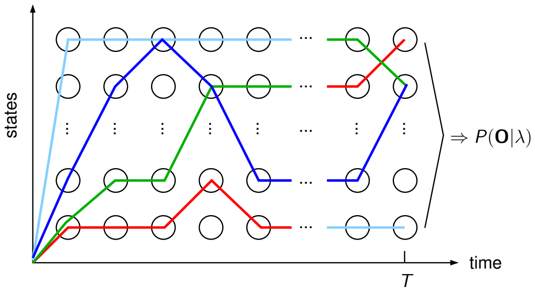
$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}} P(\mathbf{O}, \mathbf{s}|\lambda)$$

Training: *How to determine the "optimal" model?*
$\rightsquigarrow$ 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



- Naive computation infeasible: Exponential complexity $O(TN^T)$

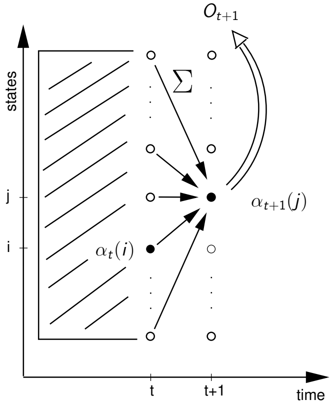
The Production Probability: The Forward-Algorithm

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 computation)



Note: There exists an analogous *Backward-Algorithm* required for parameter estimation.

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) Inference 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:

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

Computation of $\mathbf{s}^*$: *Viterbi-Algorithm*

The Viterbi Algorithm

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

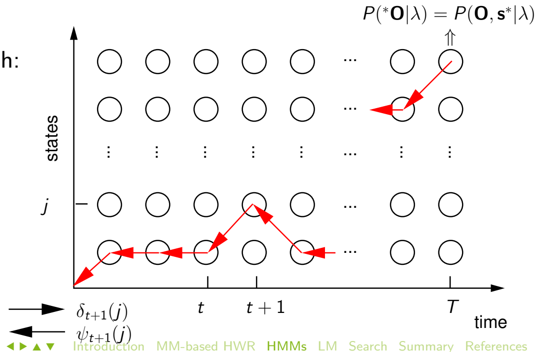
Let: $\delta_t(i) = \max\limits_{s_1, s_2, \ldots s_{t-1}} P(O_1, O_2, \ldots O_t, s_t = i|\lambda)$

1. $\delta_1(i) := \pi_i b_i(O_1)$ $\qquad \psi_1(i) := 0$
2. $\delta_{t+1}(j) := \max\limits_i(\delta_t(i) a_{ij}) b_j(O_{t+1})$ $\qquad \psi_{t+1}(j) := \operatorname*{argmax}\limits_i \ldots$
3. $P^*(\mathbf{O}|\lambda) = P(\mathbf{O}, \mathbf{s}^*|\lambda) = \max\limits_i \delta_T(i)$
   $s_T^* := \operatorname*{argmax}\limits_j \delta_T(j)$
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



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

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

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

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

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

Limitation: Initial model required!

Parameter Estimation: How to Get Started?

Problem: Parameter training only defined on the basis of *initial* parameters!

Possible Solutions:

- Random / Uniform initialization
  ↯ Only possible for discrete models
- (Fully) Supervised:
  ↯ *Detailed* annotation of training data required
- (Partly) Supervised: Compute annotation automatically with existing model

Pragmatic Solution: Use semi-continuous models

⇒ Initialization as combination of:

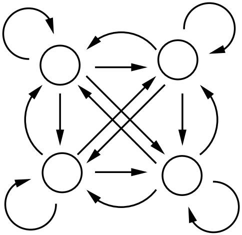
1. *Unsupervised* estimation of initial codebook
2. *Uniform* initialization of remaining parameters
   (i.e. transition probabilities and mixture weights)

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

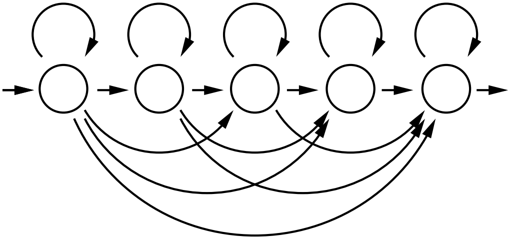
↯ "backward" transitions would allow arbitrary repetitions within the data

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!)

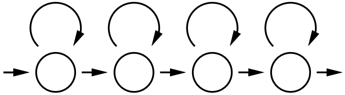
Structures/Requirements for modeling chronologically organized data:



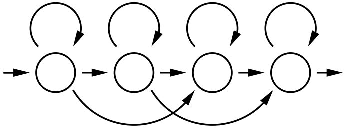
- "Forward" transitions (i.e. progress in time)
- "Loops" for modeling variable durations of segments
- "Skips" allow for optional/missing parts of the data
- Skipping of one or multiple states forward

Configuration of HMMs: Topologies III

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



linear HMM



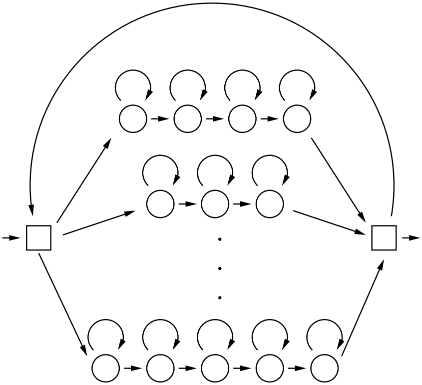
Bakis-type HMM

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)

Overview

- Introduction
- Markov Model-Based Handwriting Recognition ... *Fundamentals*
- Hidden Markov Models ... *Definition, Use Cases, Algorithms*
- Language Models ... *Definition & Robust Estimation*
- Integrated Search ... *Combining HMMs and n-Gram Models*
- Summary ... *and Further Reading*

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: 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"

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

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 ⇒ No iteratively optimizing techniques required
- Parameters can principally be computed directly (by simple counting)
- More sophisticated methods necessary in practice! [↗ parameter estimation]

Combination with an *appearance* model (i.e. HMM) [↗ integrated search]

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

- Worst case situation: All symbols equally likely

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

  - 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

Naive Method:

- Determine number of occurrences
  - $c(w_1, w_2, ...w_n)$ for all n-grams and
  - $c(w_1, w_2, ...w_{n-1})$ for $n-1$-grams
- Calculate conditional probabilities

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

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

$\Rightarrow$ *"Unseen events"*

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

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? $\rightarrow$ 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!

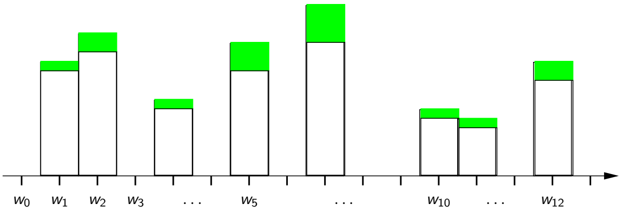
2. Redistribute *zero-probability* to *unseen events* according to a more general distribution ($\hat{=}$ *smoothing* of empirical distribution)

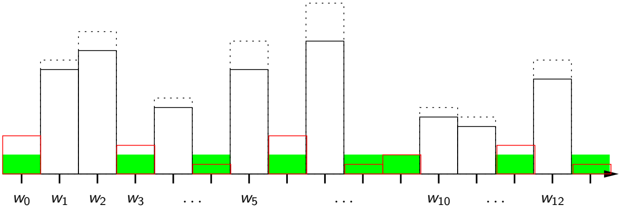
   Question: *What distribution is suitable for events we know nothing about?*

n-Gram Models: Parameter Estimation III

Robust parameter estimation: Overview

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: 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
- ⇒ 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. [4]).

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

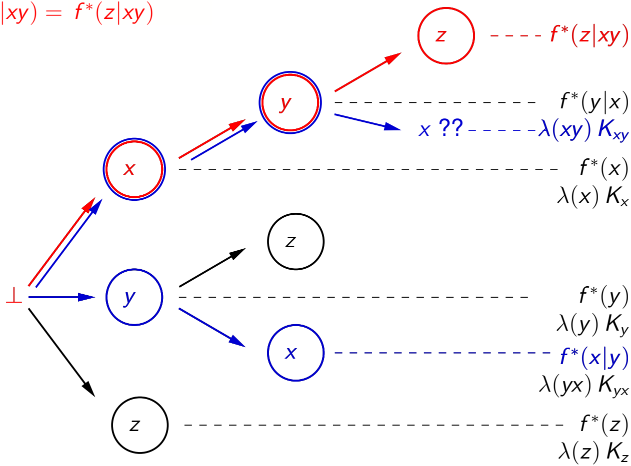
$\Rightarrow$ *Huge* savings as *most* events are not observed!

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

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



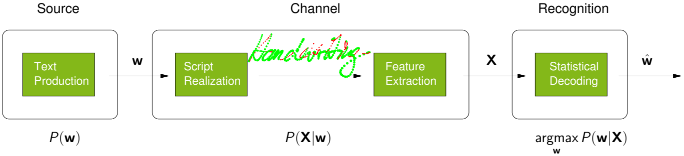
$P(x|xy) = \lambda(xy)\ K_{xy}\ f^*(x|y)$

Overview

- Introduction
- Markov Model-Based Handwriting Recognition ... *Fundamentals*
- Hidden Markov Models ... *Definition, Use Cases, Algorithms*
- Language Models ... *Definition & Robust Estimation*
- Integrated Search ... *Combining HMMs and n-Gram Models*
- Summary ... *and Further Reading*

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!

⇒ 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 *afterwards*.

  ⇒ No benefit for HMM search!

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

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

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 [13]

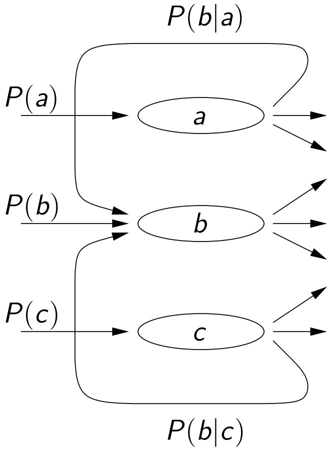
	% CER / perplexity				
	none	2	3	4	5
IAM-DB	29.2 / (75)	22.1 / 12.7	18.3 / 9.3	16.1 7.7	15.6 / 7.3
CER/$\sqrt{\mathcal{P}}$	n.a.	6.2	6.0	6.0	5.8

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

Integrated Search: HMM Networks

- Straight-forward extension of HMM-only models
- n-gram scores used as transition probabilities between words
- HMMs store single-state context only
  $\Rightarrow$ only bi-grams usable!

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

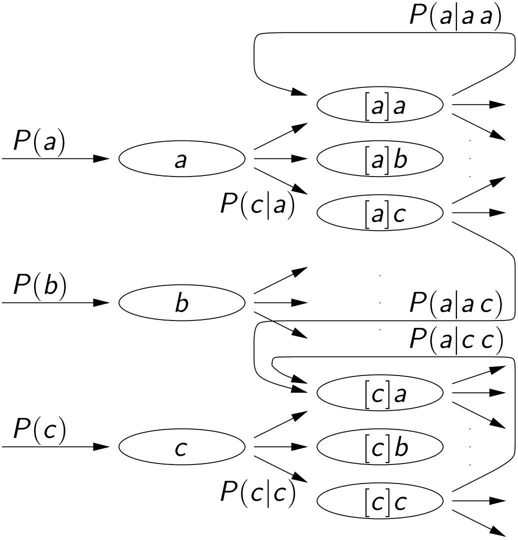


Integrated Search: HMM Networks II

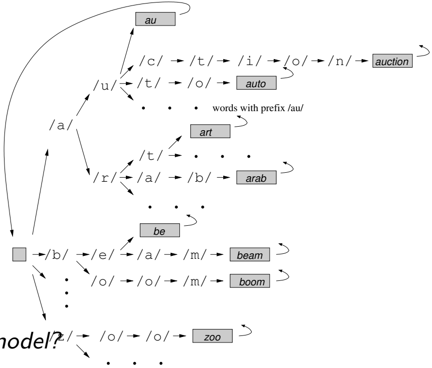
Higher-order n-gram models:

⇒ 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)

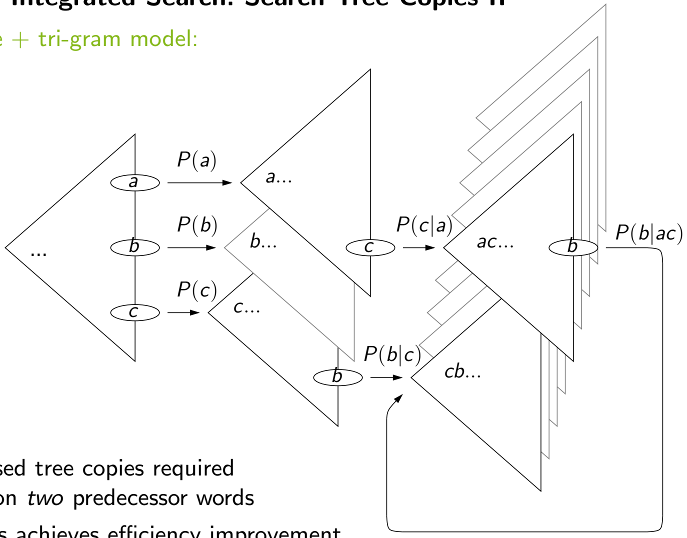
Question: *How to integrate a language model?*

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

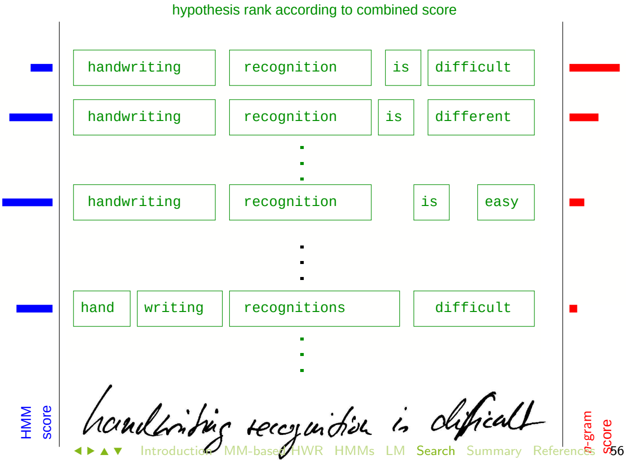
Integrated Search: Rescoring

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

Solution: Use separate search "phases" with increasing model complexity

1. Decode HMM with *inexpensive* language model (e.g. bi-gram)
2. Create alternative solutions (e.g. *n*-best)
3. Rescore with *n*-gram of *arbitrary* length

⇒ existing solutions *sorted* differently!



Overview

- Introduction
- Markov Model-Based Handwriting Recognition ... *Fundamentals*
- Hidden Markov Models ... *Definition, Use Cases, Algorithms*
- Language Models ... *Definition & Robust Estimation*
- Integrated Search ... *Combining HMMs and n-Gram Models*
- Summary ... *and Further Reading*

Markov Models for HWR: Summary

- ✓ Stochastic model for sequential patterns with high variability
- ✓ Powerful combination of appearance model (i.e. writing $\hat{=}$ HMM) and language model ($\hat{=}$ n-gram model) possible
- ✓ Efficient algorithms for training and decoding exist
- ✓ Segmentation and classification are performed in an integrated manner: Segmentation free recognition
- ↯ Model structure (esp. for HMMs) needs to be pre-defined.
- ↯ Only limited context lenghts managable (with n-gram models)
- ↯ Initial model required for training (of HMMs)
- ↯ Considerable amounts of training data necessary (as for *all* stochastic models)

"There is no data like more data!"

[Robert L. Mercer, IBM]

Further Reading

Self-Study Materials provided with this tutorial:

- How to build handwriting recognizers using ESMERALDA
- Pre-configured, ready-to-run HWR experiments on IAM-DB!

Textbook: Gernot A. Fink: *Markov Models for Pattern Recognition*. Springer, Berlin Heidelberg, 2008.

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

Survey Article: Thomas Plötz & Gernot A. Fink: Markov Models for Offline Handwriting Recognition: A Survey. *IJDAR*, 12(4):269–298, 2009.

- ✓ Open access publication!

Brand new: Thomas Plötz & Gernot A. Fink: *Markov Models for Handwriting Recognition*, SpringerBriefs in Computer Science, 2011.

References I

[1] Issam Bazzi, Richard Schwartz, and John Makhoul.
An omnifont open-vocabulary OCR system for English and Arabic.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 21(6):495–504, 1999.

[2] Radmilo M. Bozinovic and Sargur N. Srihari.
Off-line cursive script word recognition.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 11(1):69–83, 1989.

[3] T. Caesar, J. M. Gloger, and E. Mandler.
Preprocessing and feature extraction for a handwriting recognition system.
In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 408–411, Tsukuba Science City, Japan, 1993.

[4] Stanley F. Chen and Joshua Goodman.
An empirical study of smoothing techniques for language modeling.
Computer Speech & Language, 13:359–394, 1999.

References II

[5] J. G. A. Dolfing and R. Haeb-Umbach.
Signal representations for Hidden Markov Model based on-line handwriting recognition.
In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, volume IV, pages 3385–3388, München, 1997.

[6] Gernot A. Fink.
Markov Models for Pattern Recognition.
Springer, Berlin Heidelberg, 2008.

[7] S. Madhvanath, G. Kim, and V. Govindaraju.
Chaincode contour processing for handwritten word recognition.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 21(9):928–932, 1999.

[8] Thomas Plötz and Gernot A. Fink.
Markov models for offline handwriting recognition: A survey.
Int. Journal on Document Analysis and Recognition, 12(4):269–298, 2009.

References III

[9] Thomas Plötz and Gernot A. Fink.
Markov Models for Handwriting Recognition.
SpringerBriefs in Computer Science. Springer, 2011.

[10] M. Schenkel, I. Guyon, and D. Henderson.
On-line cursive script recognition using time delay neural networks and hidden Markov models.
In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 637–640, Adelaide, Australia, April 1994.

[11] Richard Schwartz, Christopher LaPre, John Makhoul, Christopher Raphael, and Ying Zhao.
Language-independent OCR using a continuous speech recognition system.
In *Proc. Int. Conf. on Pattern Recognition*, volume 3, pages 99–103, Vienna, Austria, 1996.

[12] M. Wienecke, G. A. Fink, and G. Sagerer.
Experiments in unconstrained offline handwritten text recognition.
In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, Niagara on the Lake, Canada, August 2002. IEEE.

References IV

[13] M. Wienecke, G. A. Fink, and G. Sagerer.
Toward automatic video-based whiteboard reading.
Int. Journal on Document Analysis and Recognition, 7(2–3):188–200, 2005.