UP Valencia, DSIIC, Master Course 2023/24:

Reconocimiento Automático del Habla
(Automatic Speech Recognition)

Lectures January 2024:

Automatic Speech Recognition:
Bayes Decision Rule and Deep Learning

Hermann Ney

RWTH Aachen University, Aachen, Germany
AppTek, Aachen, Germany & McLean, VA

Outline

Similar Lectures

adapted from:
- RWTH: *Advanced Topics*, 2012-2022
- UP Valencia, Master Program, Valencia, Spain, April 01-07, 2022
- Tutorial IbPRIA: Aveiro, Portugal, May 4-6, 2022

slides: UP Valencia, classes 2017-2021
– master course "Advanced Machine Learning"

slides: summer schools on *Deep Learning*

– DeepLearn Warsaw 2019 (IRDTA):
(too) many slides
– DeepLearn Nice 2021 (U Cote d'Azur):
filtered and condensed
– DeepLearn Gran Canaria 2021 (IRDTA):
filtered, condensed and corrected

Lectures: Concept

- area: speech and language:
  typical tasks: ASR, HTR and MT

  [as opposed to image (object, face, ...) recognition]

- sequence-to-sequence processing
  – as opposed to isolated events/images
  – sequence context is important

- models of sequence probabilities (ASR, HTR, MT):
  – synchronization: FSM (HMM, CTC, transducer) and attention
  – language model: captures the context of output sequence

- justification of approaches:
  – Bayes decision rule and statistical framework
  – study training criteria

Lectures: Concept

my presentation:

- appropriate framework:
  – probabilistic/statistical modelling
  – mathematical formalism and equations
- we emphasize historical context

problems:

- lots of 'noisy' experimental results,
  of 'unorthodox' ideas and of re-invented/re-named concepts
- important questions:
  – how to filter out the 'noise' in the experimental results?
  – what are really fundamental (mathematical) principles that we can rely on?

Lectures: Concept

- unifying framework:
  - probabilistic modelling and Bayes decison theory
  - basic guideline: performance of the system
    for analyzing the system components and training procedure
  - deep learning is just one out of many machine learning approaches
  - rarely covered by textbooks
- specific aspects:
  - success of data-driven approaches
  - for many areas: ASR, HTR, MT, NLP
  - things started 40 years ago, not in 2013!
  - evolution from small to large acoustic models and language models
  - sort out the fundamental principles beyond experimental noise
  - framework: (applied) mathematical and statistics
- key messages:
  - there has been, is and will be life outside deep learning
  - there is NO life outside probabilistic modelling (Bayes framework)

1 Introduction

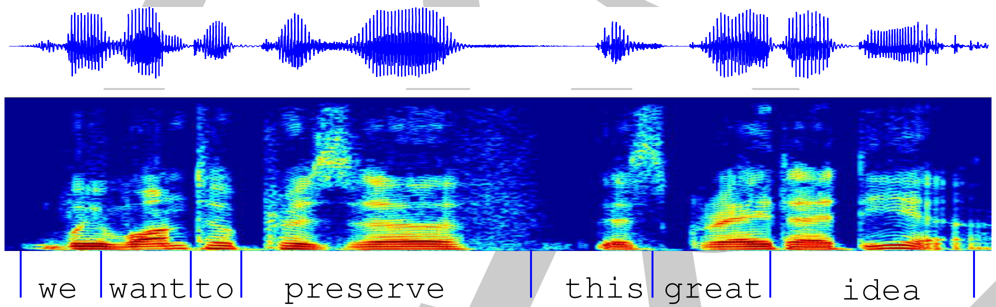
1.1 HLT: Speech & Language Technology

outline:

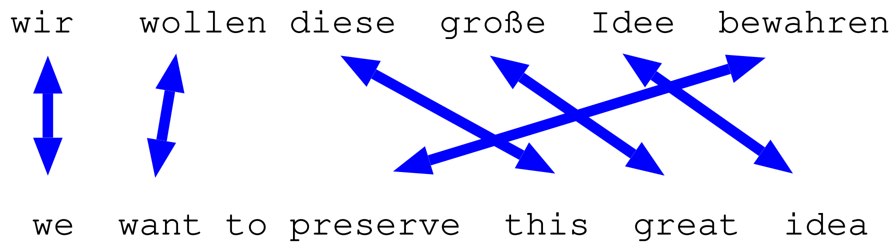
– terminology: speech vs. language
– speech and language technology
– seq-to-seq processing

Speech & Language Technology: Sequence-to-Sequence Processing

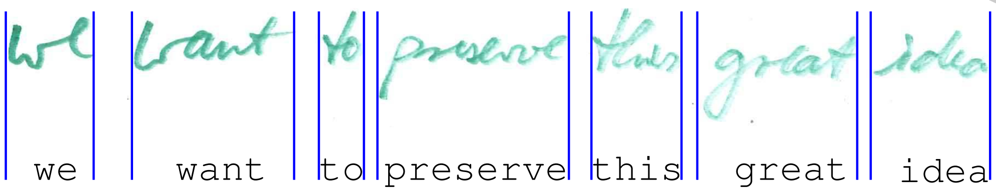
Automatic Speech Recognition (ASR)
(speech signal processing)



Machine Translation (MT)
(symbol or text processing)



Handwritten Text Recognition (HTR)
(text image processing)



common characteristics:

- **use of a 'small' language model (LM) to generate smooth fluent text (syntax, semantics, context)**
- ***generative aspect* of LM: unlike *formal* NLP tasks (POS/synt./semant. labels, ...)**
- **LM is learned from text only (*without annotation, unsup. mode, pre-training)***

note: this is how (small) language models started (1980 – 2000) [Jelinek & Mercer^{+} 77]

terminology:

- speech: acoustic signal, spoken language
- language: text, sequence of characters, written language
- scientific disciplines: speech vs. language
  – NLP: natural language processing (in the strict sense): written language only
  – HLT: human language technology: spoken AND written language;
  speech and language technology
- specific well-defined tasks in HLT:
  – automatic speech recognition (ASR)
  – text image recognition (printed and handwritten text) (HTR)
  – machine translation (MT) (of language and speech)

characteristic properties of ASR: :

- well-defined 'classification' tasks:
  – due to 5000-year history of (written!) language
  – well-defined classes: letters or words of the language
- easy task for humans
  (at least in their native language!)
- hard task for computers
  (as the last 50 years have shown!)

– AI: artificial intelligence:
a computer that performs human tasks like ASR, MT and other HLT tasks
(prototypical tasks of AI)

– 1970-2010: rule-based approaches for AI (mainly text/symbolic)
facts and formal derivations/proofs: like PROLOG programming

– (wide-range) success of artificial neural nets around 2012/2013: deep learning
today: AI = data-driven AI = deep learning

today's terminology: generative AI (vs. formal AI/NLP)

– generative NLP tasks: input and output in natural language
(form of natural language varies: spoken, handwritten, digital text)

– formal NLP tasks: output requires a specific formalism

common goals of both fields:

- **analyze empirical data**
- **learn from data**

differences in practice:

- ***orthodox* statistics (and information theory):**
  - **emphasis on theoretical analysis of data**
  - **models with small number of parameters**
  - **(ideally) closed-form solutions**
- **machine learning (pattern recognition, data-driven approaches):**
  - **goal: build operational systems with optimum practical performance**
  - **large amounts of data**
  - **models with millions of parameters and with numeric solutions only**
  - **important: efficiency of algorithms**
  - **includes ANNs**

Large-Scale Projects 1984-2020

- **SPICOS 1984-1989: speech recognition und understanding**
  - **conditions: 1000 words, continuous speech, speaker dependent**
  - **funded by German BMBF: Siemens, Philips, German universities**

- **Verbmobil 1993-2000: funded by German BMBF**
  - **domain: appointment scheduling, recognition and translation, German-English, limited vocabulary (8.000 words)**
  - **large project: 10 million DM per year, about 25 partners**
  - **partners: Daimler, Philips, Siemens, DFKI, KIT Karlsruhe, RWTH, U Stuttgart, ...**

- **TC-STAR 2004-2007: funded by EU**
  - **domain: recognition and translation of speeches given in EU parliament**
  - **task: speech translation: ASR + MT (+TTS)**
  - **challenge: MT robust wrt ASR errors $\rightarrow$ data-driven methods**
  - **first research prototype for unlimited domain and real-life data**
    - **fully automatic, not real time**
    - **without deep learning!**
  - **partners: KIT Karlsruhe, RWTH, CNRS Paris, UPC Barcelona, IBM-US Research, ...**

- **GALE 2005-2011: funded by US DARPA**
  - **recognition, translation and understanding for Chinese and Arabic**
  - **largest project ever on HLT: 40 million USD per year, about 30 partners**
  - **US partners: BBN, IBM, SRI, CMU, Stanford U, Columbia U, UW, USCLA, ...**
  - **EU partners: CNRS Paris, U Cambridge, RWTH**

- **BOLT 2011-2015: funded by US DARPA**
  – follow-up to GALE
  – emphasis on colloquial language for Arabic and Chinese
- **QUAERO 2008-2013: funded by OSEO France**
  – recognition and translation of European languages,
  more colloquial speech, handwriting text recognition
  – French partners (23): Thomson, France Telecom, Bertin, Systran,
  CNRS, INRIA, universities, ...
  – German Partners (2): KIT Karlsruhe, RWTH
- **BABEL 2012-2017: funded by US IARPA**
  – recognition (key word spotting) with noisy and low-resource training data
  – rapid development for new languages (e.g. within 48 hours)
- **EU projects 2012-2014: EU-Bridge, TransLectures**
  emphasis on recognition and translation of lectures (academic, TED, ...)
- **EU ERC advanced grant 2017-2021:**
  emphasis on basic research for speech and language

my team at RWTH:
– PhD students learn to build fully-fledged systems for ASR and other HLT tasks
– later many of them work for the GAFAM+: Google, Apple, Facebook, Amazon, Microsoft + ...

- **text generation:**
  – text dictation and translation
  – closed captioning (subtitles)
- **content-based information access to unstructured data (internet, archive, ...):**
  – audio/video documents (talks, lectures, call centers, meetings, ...)
  – multilingual data (audio + text)
- **personal communication in foreign languages:**
  human-to-human communication
- **personal assistant (Siri, Cortana, Alexa, ...):**
  human-to-machine communication for queries and warehouse orders
- **car navigation, smart home, ...**

hardware products:

Amazon Echo, Google Home, Apple Homepod, Microsoft Invoke, ...

1.2 History: ASR and ANNs

very short history:

- **many fundamental concepts were developed before and independently of ANNs and deep learning**
- **most important innovation by deep learning: sophisticated structurs for modelling the dependencies between input and output**

ASR: first research 1975–1980

ASR is sequence-to-sequence processing:

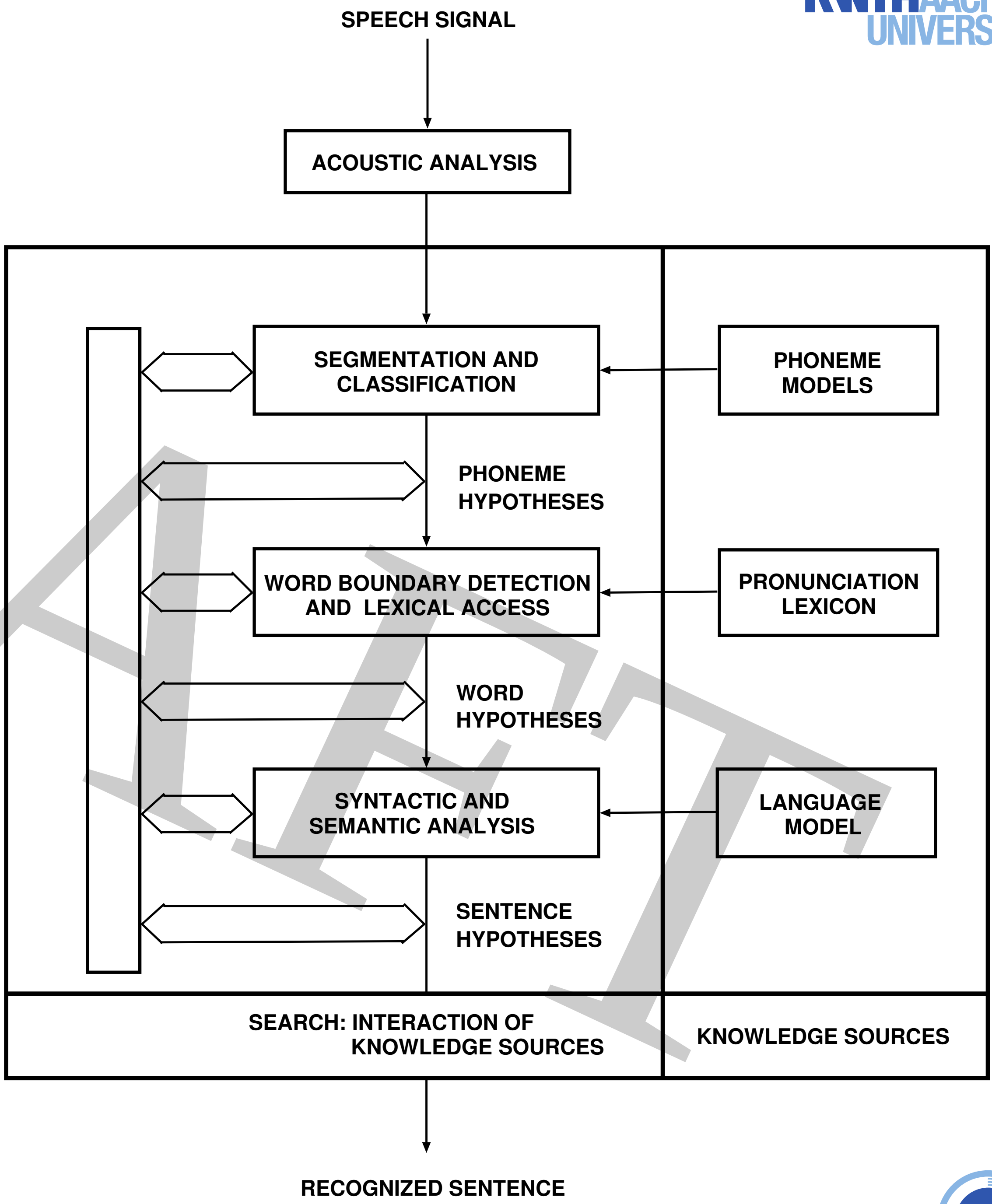
- sequence of 10-ms acoustic vectors
- sequence of sounds/phonemes
- sequence of letters
- sequence of words

problems:

- ambiguities at all levels
- interdependencies of decisions

approach 1975-1980 (Baker/CMU and Jelinek/IBM):

- probabilitistic modelling
- holistic approach ('end-to-end'): single criterion for system design (Bayes decision rule)
- complex mathematical modelling



1980-1995 Baseline Statistical Approach to ASR

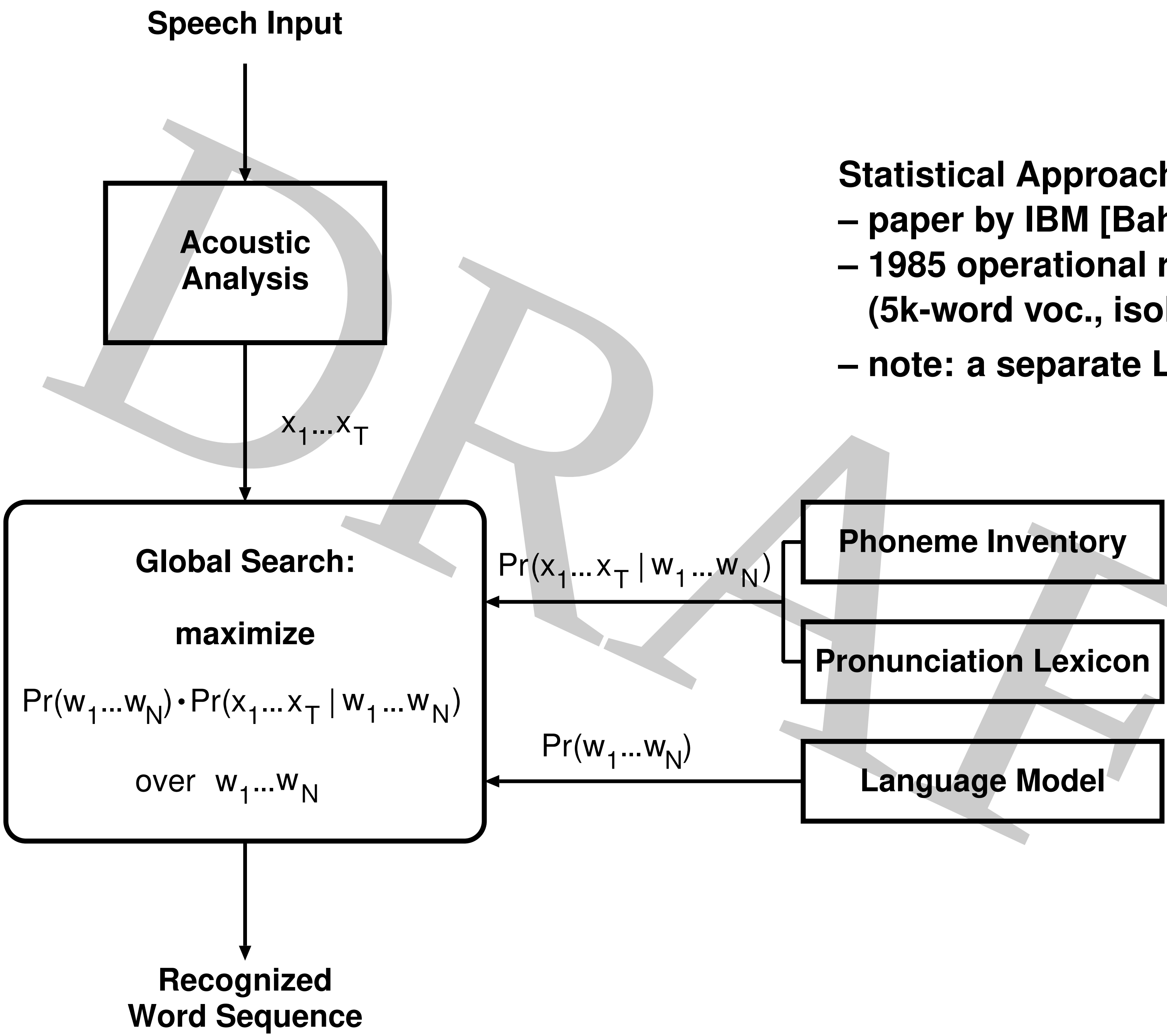
- modelling: probability distributions/data-driven approaches with

$$\text{10-msec vectors:} \quad x_1^T = x_1 ... x_t ... x_T \qquad x_t \in \mathbb{R}^D$$
$$\text{word string:} \quad w_1^N = w_1 ... w_n ... w_N$$

- consider joint generative model: $p(w_1^N, x_1^T) = p(w_1^N) \cdot p(x_1^T | w_1^N)$
- language model $p(w_1^N)$: based on word trigram counts, learned from text only $[w_1^N]$
- acoustic (-phonetic) model $p(x_1^T | w_1^N)$: learned from annotated audio data $[x_1^T, w_1^N]$
  - generative hidden Markov model:
    discrete models/VQ, Gaussians, Gaussian mixtures, ...
  - structure: first-order dependence and mathematically nice
  - training: ('efficient') EM algorithm with sort of closed-form solutions
- dichotomy:
  - general machine learning (like CV): single (isolated) events (x, c):
    emphasis on 'discriminative' class posterior $p(c|x)$ (rather than $p(x, c) = p(c) \cdot p(x|c)$)
  - sequence-to-sequence task (like ASR: time alignment and LM context):
    emphasis on 'generative' joint model $p(x_1^T, w_1^N)$
- decoding/generation: Bayes decision rule (simplified form)
  = use single criterion and avoid local decisions

Statistical Approach to ASR

- paper by IBM [Bahl & Jelinek$^+$ 83]
- 1985 operational research system: *Tangora*
  (5k-word voc., isolated words, speaker dep.)
- note: a separate LM



ASR History: Operational Research Systems

- steady improvement of data-driven methods:
  HMMs with Gaussians and mixtures, phonetic CART, statistical trigram language model, speaker adaptation, sequence discriminative training, ANNs
- methodology in ASR since 1990: standard public data:
  TIMIT, RM/1k, WSJ/5k, WSJ/20k, NAB/64k, Switchboard/tel., Librispeech, TED-Lium
- 1993-2000 NIST/DARPA: comparative evaluation of operational systems:
  – virtually all systems: generative HMMs and refinements
  – 1994 Robinson: hybrid HMM with RNN (singularity!)

alternative concepts (with less success):

- 1985-93: criticism about data-driven approach/machine learning
  – acoustic model: too many parameters and saturation effect
  – concept of rule-based AI: acoustic-phonetic expert systems
  – language model: similar criticism (linguistic structures/grammars)

- SVM (support vector machines): never competitive in ASR
  (ASR requires decisions in context!)

ASR: ANN in Acoustic Modelling

- 1987 [Bourlard & Wellekens 87]: MLP and ASR
- 1988 [Waibel & Hanazawa+ 88]: phoneme recognition by TDNN (convol.NNs!)
- 1989 [Bourlard & Wellekens 89, Morgan & Bourlard 90]:
  – ANN outputs: can be interpreted as class posteriors
  – *hybrid HMM*: use ANN for frame label posteriors
- 1989 [Bridle 89]: softmax ('Gaussian posterior') for normalized ANN outputs
- 1991 [Bridle & Dodd 91] backpropagation for HMM discriminative training at word level
- 1993 [Haffner 93]: sum over label-sequence posterior probabilities in hybrid HMMs (*sequence discriminative training*)
- 1994 [Robinson 94]: RNN in hybrid HMM (operational system, DARPA evaluations)
- 1997 [Fontaine & Ris+ 97, Hermansky & Ellis+ 00]: *tandem HMM*: use ANN for feature extraction in a Gaussian HMM
- 2009 Graves: CTC for handwriting recognition (operational system, ICDAR competition 2009)
- 2012 and later: mainstream of hybrid HMM (transducer) and deep learning
- 2015 [Bahdanau & Cho+ 15] cross-attention (+ RNN) for MT (and ASR)
- 2017 [Vaswani & Shazeer+ 17] transformer := cross-attention + self-attention

hybrid HMM: ANN-based feature extraction + Gaussian posterior + HMM

- **2009 [Graves 09]: CTC - good results on LSTM RNN for handwriting task**
- **2010 [Dahl & Ranzato$^+$ 10]: improvement in phone recognition on TIMIT**
- **2011 [Seide & Li$^+$ 11, Dahl & Yu$^+$ 12]: Microsoft Research**
  – fully-fledged hybrid HMM
  – 30% rel. WER reduction on Switchboard 300h
- **since 2012: other teams confirmed reductions of WER by 20% to 30%**

tandem HMM: ANN-based feature extraction + generative Gaussian + HMM

- **2006 [Stolcke & Grezl$^+$ 06]: cross-domain and cross-language portability**
- **2007 [Valente & Vepa$^+$ 07]: 8% rel. WER reduction on LVCSR**
- **2011 [Tüske & Plahl$^+$ 11]: 22% rel. WER reduction on LVCSR/QUAERO**
  (Interspeech 2011, like [Seide & Li$^+$ 11])

experimental observation for hybrid and tandem HMM:
progress by using *deep* MLPs

key problem in ASR: synchronization between input and output

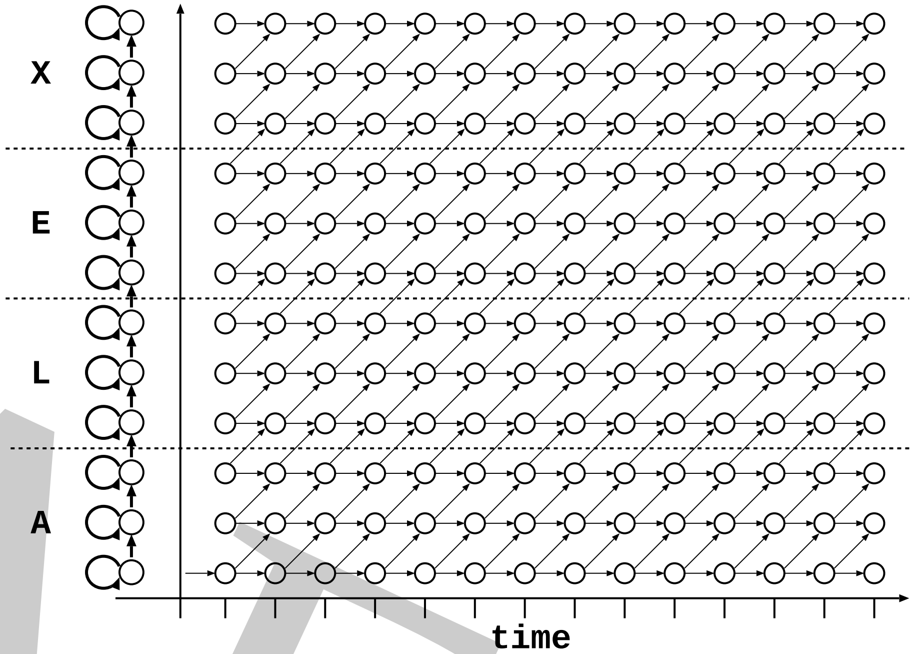
two main bapproaches:

- **HMM and variants:**
  - hybrid HMMs
  - finite-state transducers
  - CTC
  - RNN transducer
- **cross-attenion:**
  - using RNNs
  - using self-attention: transformer

First-Order Models: Hidden Markov Model (HMM)

– **sequence of acoustic vectors:**
  $X = x_1^T = x_1 ... x_t ... x_T$ **over time** t

– **sequence of states** $s = 1, ..., S$
  $s_1^T = s_1 ... s_t ... s_T$ **over time** t
  with associated state labels:
  $a_1^S = a_1 ... a_s ... a_S$
  $= W$**: word sequence**



objective of HMM: time alignment
= synchronization between input and output

- **classical HMM: generative model for** x_1^T**:**

$$q_\vartheta(x_1^T | W = a_1^S) = \sum_{s_1^T} \prod_t q(s_t | s_{t-1}, W, \vartheta) \cdot q_t(x_t | a_{s=s_t}, \vartheta)$$

- **hybrid HMM: model of label posterior sequence** a_1^S**:**

$$q_\vartheta(W = a_1^S | x_1^T) = \sum_{s_1^T} \prod_t q(s_t | s_{t-1}, W, \vartheta) \cdot q_t(a_{s=s_t} | x_1^T, \vartheta)$$

machine learning point-of-view: it is easier to model $q_t(a_s | x_1^T, \vartheta)$ **than** $q_t(x_t | a_s, \vartheta)$
HMM: [Bourlard & Wellekens 89], CTC: [Graves & Fernandez$^+$ **06], RNN-T: [Graves 12]**

Finite State Machines:
Hybrid HMM, Direct HMM, (RNN-) Transducer

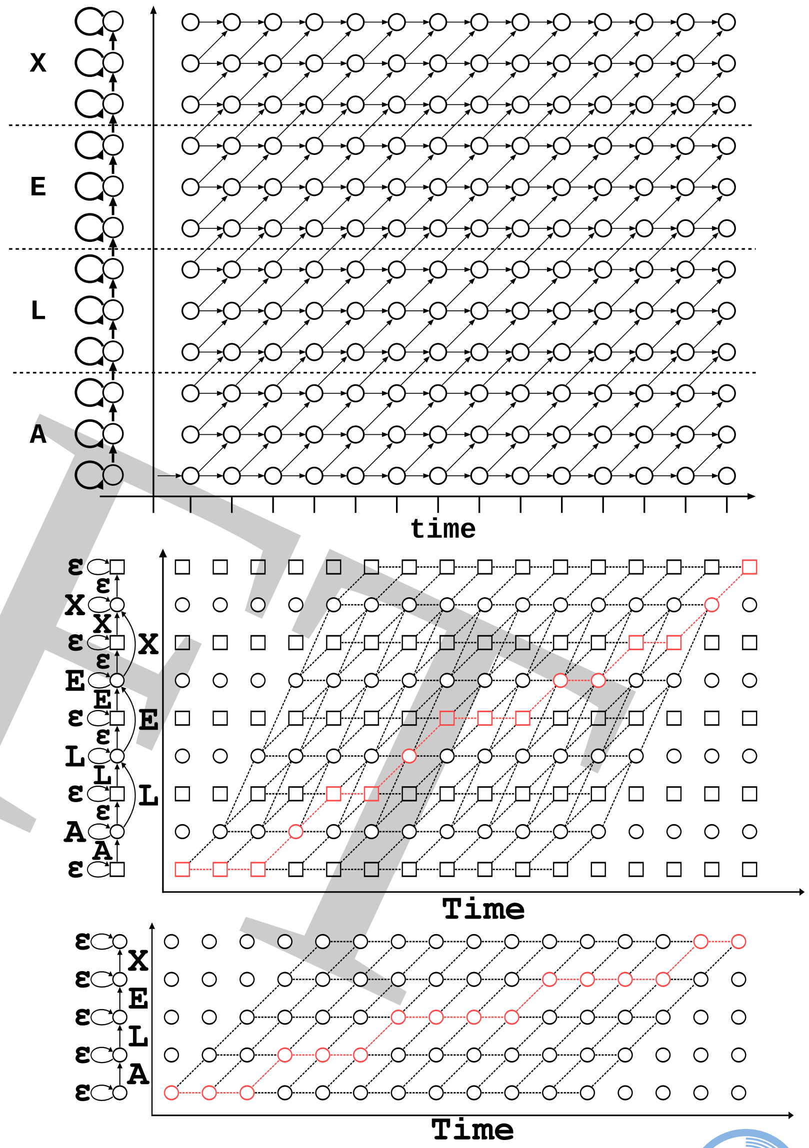
unifying framework:
– (hidden) alignment path: state sequence
– first-order dependencies along path

main differences in structures:

- **generative model: $p(x_1^T|a_1^S)$**
  classical model of 1975–2000

- **discriminative model: $p(a_1^S|x_1^T)$**
  started around 1990
  – hybrid HMM
  – CTC (with blank symbol ϵ
  and specific transition constraints)
  – RNN transducer: similar to CTC
  with 'integrated LM' for output labels
  – direct or posterior HMM (RWTH team)

in addition: other differences
(e.g. sum vs. maximum, transition prob., ...)



Basic Units for ASR

traditional:
(context-dependent) phoneme models (e.g. CART)
+ pronunciation lexicon (manual!)

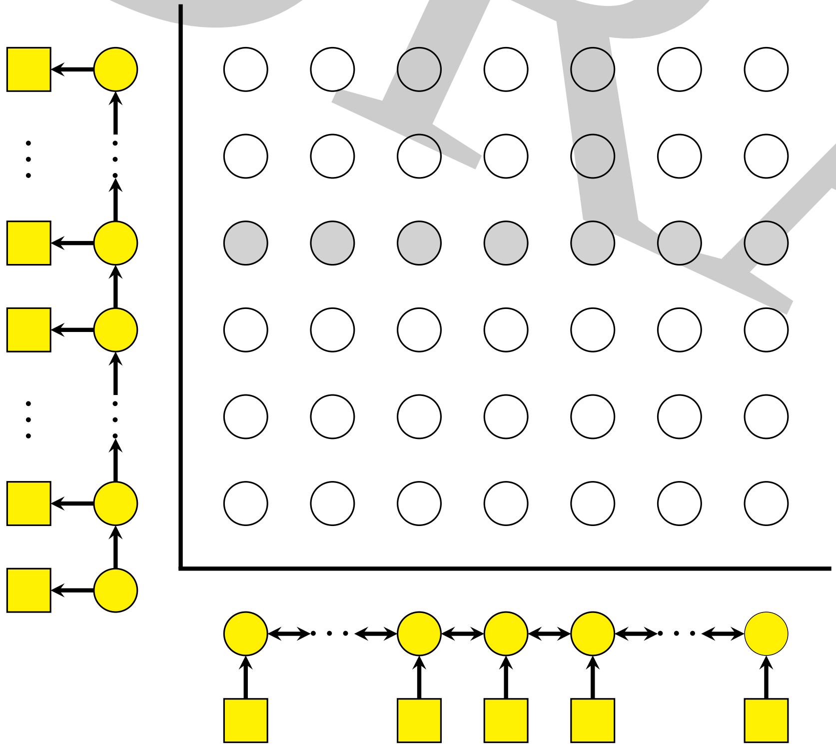
grapheme-based units (before deep learning):
S. Khantak, ICASSP 2002

BPE units: BPE = byte pair encoding
(typically: 1000-10.000 units):
– Google: piece of word models
– for both ASR, LM and MT!
– note: increased abstraction capability of ANN models

related aspect:
frame rate: from 10 msec to 40 msec

synchronisation: two-dimensional problem:

– find associations/alignments between input and output sequence
– preprocessing (representations) for both input and output:
o input: 'encoder': bi-directional sequence model
o output: 'decoder': uni-directional sequence model

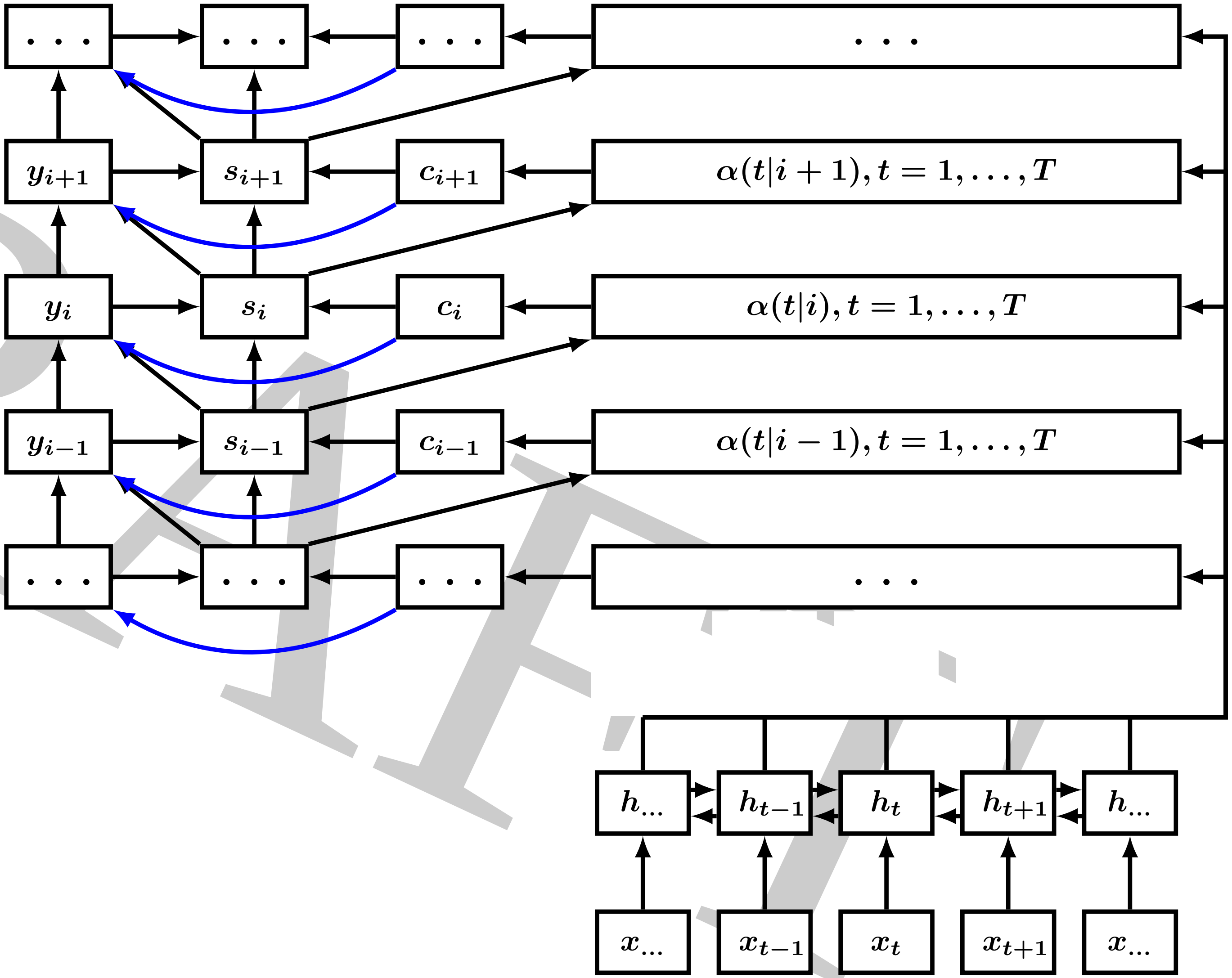


today's most succesful approaches:

- **attention mechanism:**
  state vector and context vector
  along with attention weights
- **finite-state machines (HMM, transducer):**
  hidden alignment path with
  first-order dependences

Cross-Attention for ASR/HLT

- **input/output sequences:**
  $x_1^T \to a_1^I$
- **model with factorization:**
  $p(a_1^I|x_1^T) =$
  $= \prod_i p(a_i|a_0^{i-1}, x_1^T)$
  $= \prod_i p(a_i|a_{i-1}, s_{i-1}, x_1^T)$
  $= \prod_i p(a_i|a_{i-1}, s_{i-1}, c_i$
  (incl. implicit LM!)
- **ANN notation:**
  $y_i \equiv p(a_i|a_{i-1}, s_{i-1}, c_i)$
- **state vector:**
  $s_i = S(s_{i-1}, a_i, c_i)$
- **context vector:**
  $c_i = \sum_t \alpha(t|i, s_{i-1}, h_1^T) \cdot h_t$
  with attention weights:
  $\alpha(t|i, s_{i-1}, h_1^T) = \ldots$
- **representation vector h_t:**
  $h_t = H_t(x_1^T)$
  (acoustic encoder)



From ASR to Machine Translation (MT): History

statistical/data-driven approaches were controversial in MT (and other NLP tasks):

- **1969 Chomsky:**
  ... the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term.
- **result: mainstream research had a *strict dichotomy* until (around) 2000:**
  – speech = spoken language: signals, subsymbolic, machine learning
  – language = written text: symbols, grammars, rule-based AI
- **until 2000: mainstream approach was rule-based**
  – result: huge human effort required in practice
  – problems: coverage and consistency of rules
- **1989-93: IBM Research: statistical approach to MT**
  1994: key people (R. Mercer, P. Brown) left for a hedge fund
- **1996-2002 RWTH: improvements beyond IBM's approach:**
  – HMM alignments, log-linear modelling, phrases as basic units
  – superior results in DARPA/NIST evaluations
- **around 2004: from singularity to mainstream**
  – F. Och (and more RWTH PhD students) joined Google
  – 2008: service *Google Translate*
- **since 2014: neural MT (unlike count-based MT):**
  attention mechanism [Bahdanau & Cho$^+$ 15]

1.3 Preview: Structure of ASR/HLT Systems

an ASR/HLT system has several components:

– a performance criterion (e.g. WER in ASR)
– a (probabilistic) model for handling input-output
– a training criterion and its associated training procedure
– a generation component that generates the output

view: various ways of combinining these components

in addition: data
– (annotated) training data
– test data (different from training data!)

additional aspects:
– what are the basic units for recognition?
– what about language modelling in addition to acoustic modelling?

principal components:

- **performance measure, error measure, cost function:**
  – how to judge the quality of the system output
  – examples: ASR: edit distance; MT: TER or BLEU
  (later: relation to associated decision/generation rule)

- **probabilistic models (with a suitable structure)**
  for capturing the dependencies within and between input and output strings:
  – given synchronization:
  Markov chain, CRF, (LSTM) RNN, ...
  – input/output synchronization:
  generative/hybrid HMM, CTC, transducer, attention (incl. transformer, ...

  – separation of acoustic and language models vs. *end-to-end* concepts

- **training criterion:**
  to learn the free model parameters from examples
  – ideally should be linked to performance criterion
  – questions: exact form of criterion? optimization strategy?
  – question: what is the relation with performance?

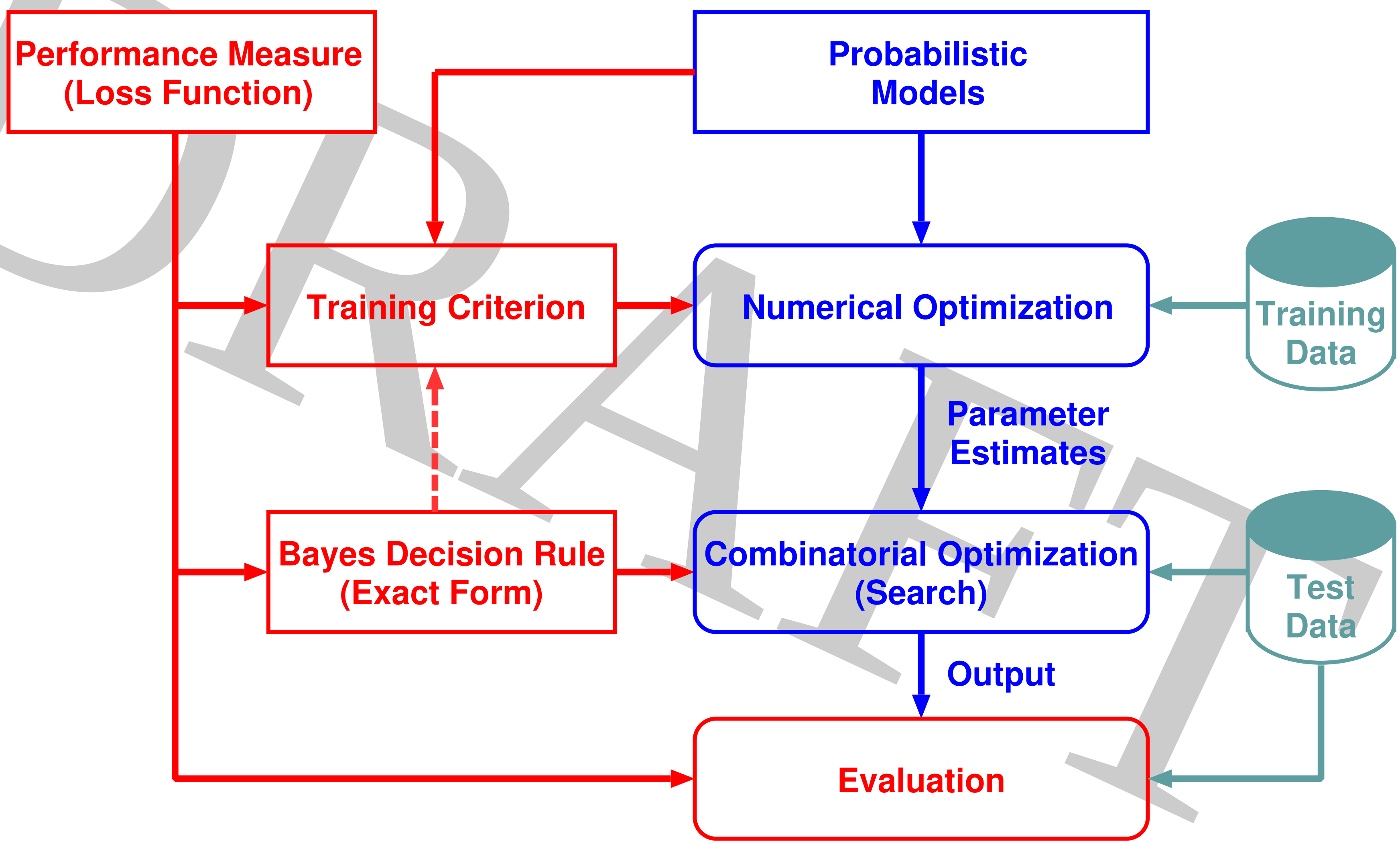
  examples of training criteria:
  – maximum likelihood
  – cross-entropy
  – sum criterion (CTC) and EM algorithm
  – sequences discriminative training (and epxected Bayes risk)

  training strategy:
  – closed-form solutions vs. iterative methods
  – example: backpropagation (gradient descent)

- (Bayes) decision rule: decoder/search
  for generating the output word sequence
  – combinatorial problem (efficient algorithms)
  – should exploit structure of models

  examples: dynamic programming and beam search, A* and heuristic search, ...
  [Vintsyuk 68, Velichko & Zagoruyko 70, Vintsyuk 71] and [Sakoe & Chiba 71],
  [DRAGON/HARPY 1975] and [Bridle 82, Ney 84, Ney & Haeb-Umbach$^+$ 92]

Structure of ASR/HLT Systems



Open Questions

– why do we use Bayes decision rule for ASR ?
– what is the relation of the ANN framework with Bayes decison rule?
– what is the role of softmax output layer in ANNs ?
– what is the relation of training criteria with Bayes decison rule/classification error ?
– what is the relation between training criteria and end-to-end modelling ?
– why should we separate acoustic model and language model ?
– how to use ANNs for acoustic modelling? suitable ANN structures?
– what are synchronization/alignment methods for acoustic modelling ?
– how to use ANNs for language modelling? suitable ANN structures ?

2 Unifying Framework: Bayes Decision Theory

- **so far: historical review of ASR (along with MT) and ANNs covering a variety of ANN models and training criteria**
- **what about training criteria?**
  (e. g. cross-entropy, seq.disc. training, state-level min. Bayes risk, expected risk, ...)

  ultimate justification should be based on performance
  – consequence: re-visit Bayes decision rule und its framework
  – example: textbook by Duda & Hart 1973, pp. 11-16
  – originally not explicitly meant for ASR or string processing
- **what is not well covered in textbooks or papers:**
  – mathematical relation between training criteria and loss function/performance
  – practical implications for training criteria

references, mostly RWTH:
[Ney 03, Schlüter & Scharrenbach$^+$ 05, Xu & Povey$^+$ 10, Schlüter & Nussbaum$^+$ 11],
[Schlüter & Nussbaum$^+$ 12, Schlüter & Nussbaum-Thom$^+$ 13, Schlüter & Beck$^+$ 19]

2.1 Principle

- concept: imagine a "huge huge" database of (input,output) string pairs $[x, c]$:

$$[x_r, c_r], \quad r = 1, ...R$$

- define empirical distribution: $$pr(x, c) = \frac{1}{R} \cdot \sum_{r=1}^{R} \delta(x, x_r)\delta(c, c_r)$$

  remarks:
  – fully specified, no free parameters
  – derived distributions: $pr(c), pr(x), pr(c|x), pr(x|c)$
  – easy principle (i. e a "huge" table), but difficult implementation for strings
  – simplifying assumption about input x: discrete rather than continuous-valued

- guessing game: knowing x guess c:

$$x \to c = \hat{c}(x)$$

  terminology:
  – classify the input data (ASR, HTR) or generate the output data (MT)
  – decision rule

- **perfect solution is not possible:**
  - **we want to convert a relation $[x, c]$ into a function $x \rightarrow c = \hat{c}(x)$**
  - **for each pair $[x, c]$, we want to compare: $c \stackrel{?}{=} \hat{c}(x)$**
    and thus we need an error measure or loss function $L[c_r, \hat{c}(x_r)],\ r = 1, ..., R$

- **popular error measures for strings**
  (sequence of symbols: words, subword units, graphemes/letters, phonemes):
  - **in general: 0/1 loss function = string error:**
    is the string correct as a whole?
  - **strings in ASR/HTR: WER = word ("symbol") error rate**
    WER = edit distance = errors: ins + del + sub
  - **strings in MT: TER = translation error rate**
    TER = edit distance + swaps of symbol groups

    alternative: BLEU (more complex)

- **key question: how to generate the output string?**
  - **perfect solution is not possible**
  - **best compromise: for each input x (which might exist in several pairs $[x = x_r, c_r]$),**
    select an output that minimizes the expected loss/cost:

$$x \rightarrow c_*(x) \ := \ \arg\min_{c} \left\{ \sum_{\tilde{c}} pr(\tilde{c}|x) \cdot L[\tilde{c}, c] \right\}$$

using the class posterior distribution $pr(c|x)$ of the data

Decision Rules: Bayes, Pseudo Bayes and Approximations

- **Bayes decision rule:**

$$x \to c_*(x) \;:=\; \arg\min_c \left\{ \sum_{\tilde{c}} pr(\tilde{c}|x) \cdot L[\tilde{c}, c] \right\}$$

shortcomings in practice:
– difficult/impossible to store $pr(c|x)$
– generalization (from closed to open world): how to handle unseen inputs x ?

- **replace the empirical distribution $pr(c|x)$ by a model $p_\vartheta(c|x)$ ("pseudo Bayes") with parameters ϑ to be learned from data (e. g. neural net):**

$$x \to c_\vartheta(x) \;:=\; \arg\min_c \left\{ \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c] \right\}$$

- **special choice of loss function: 0/1 = string error:**

$$L[\tilde{c}, c] \;=\; 1 - \delta(\tilde{c}, c) \;\in \{0, 1\}$$
$$x \to c_\vartheta(x) \;:=\; \arg\max_c \left\{ p_\vartheta(c|x) \right\}$$

– terminology: MAP rule (MAP = maximum a-posteriori)
– starting point in most systems
– strictly speaking: adequate only for string error

Decision Rule: Bayes and Pseudo Bayes

two conditions for guaranteed optimality:
– exact implementation of loss function
– the true distribution $pr(c|x)$ is known

open issues:
– how to fill in the implementation details (e. g. search)?
– how to generalize towards unseen data (e.g. input string x)
and to replace the unknown distribution $pr(c|x)$ by a model $p_\vartheta(c|x)$?
– how to train the parameters ϑ of the model?

specific aspects considered here:

- **loss function for strings:**
  exact loss vs. 0/1 loss
- **mismatch conditions:**
  – replace true distribution $pr(c|x)$ by a model $p_\vartheta(c|x)$
  – optimality of (pseudo) Bayes rule ?
  – good training criterion: relation to loss ?

Bayes Decision Rule and Loss Function

goal: to study the effect of the loss function

three types of outputs and associated loss functions:

- **"atomic" output: system output has no 'internal structure', i. e. single symbols or string as a whole**
- **strings with synchronization: loss function: Hamming distance based on normalized positions (equivalent to symbol error for each position of output string)**
- **strings with no synchronization: metric loss function: edit distance and generalizations**

note minimalistic notation:

– **single (class) symbol:** c **or** c_n **(similar:** x **vs.** x_t**)**
– **indices:** $n = 1, .., N$ **or** $t = 1, ..., T$
– **several variables of the same type:** $c, \tilde{c}, c', ...$
– **string of symbols:** c **or** $c_1^N = c_1 ... c_n ... c_N$
– **decision rule generating an output:** $x \to \hat{c}(x)$ **or** $x \to c_x$

– **true distribution:** $pr(c, x), ...$
– **model distribution:** $p(c, x), q(c, x)$ **or** $p(x, c), q(x, c)$
– **model distribution: with parameters** $p_\vartheta(c, x), q_\vartheta(c, x)$

0/1 Loss Function: Weighted Errors

Bayes decision rule: $x \to \hat{c}(x) = \arg\min_c \left\{ \sum_{\tilde{c}} p(\tilde{c}|x)\; L[\tilde{c}, c] \right\}$

with $p(c|x)$ being either the true distribution $pr(c|x)$ or a normalized model $p_\vartheta(c|x)$

- **baseline: no weights result in MAP rule:**

$$\begin{aligned} L[\tilde{c}, c] &= [1 - \delta(\tilde{c}, c)] \\ x \to \hat{c}(x) &= \ldots = \arg\max_c \left\{ p(c|x) \right\} \end{aligned}$$

- **assumption: weights α_c depend on hypothesized class c:**

$$\begin{aligned} L[\tilde{c}, c] &= [1 - \delta(\tilde{c}, c)] \cdot \alpha_c \\ x \to \hat{c}(x) &= \ldots = \arg\min_c \left\{ \alpha_c \cdot \left[1 - p(c|x)\right] \right\} \end{aligned}$$

- **assumption: weights $\beta_{\tilde{c}}$ depend on correct class $\tilde{c}$:**

$$\begin{aligned} L[\tilde{c}, c] &= [1 - \delta(\tilde{c}, c)] \cdot \beta_{\tilde{c}} \\ x \to \hat{c}(x) &= \ldots = \arg\max_c \left\{ \beta_c \cdot p(c|x) \right\} \end{aligned}$$

remarks:
– loss function $L[\tilde{c}, c]$ is not symmetric anymore!
– could be useful for information retrieval (e. g. content words)
– weights rarely used in practice

2.2 Loss Functions and Decision Rules

three types of decision rules for sequences,
based on error counting:
– any type of strings: 0/1 loss: minimum sequence error:
count sequence errors
– synchronized strings: Hamming distance:
count symbol/word errors in fixed positions
– any type of strings: general loss function
count symbol/word errors with 'alignments' (WER, TER, ...)

question: does the exact form of the loss function matter?

Strings with Synchronization

assumption:
– positions $n = 1, ..., N$ (including the length N) are well defined by the task
– loss function for two strings: c_1^N and $\tilde{c}_1^N$ with positions $n = 1, ..., N$:

correct string with symbols:	$\tilde{c}_1$	$\tilde{c}_2$	...	$\tilde{c}_{n-1}$	$\tilde{c}_n$	$\tilde{c}_{n+1}$	...	$\tilde{c}_{N-1}$	$\tilde{c}_N$
	$\vert$	$\vert$	$\vert$	$\vert$	$\vert$	$\vert$	$\vert$	$\vert$	$\vert$
hypothesized string with symbols:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

two types of errors and associated loss functions:

- **string error: consider the string as a whole ("atomic" unit) (with a suitable definition of the Kronecker delta for strings):**

$$L[\tilde{c}_1^N, c_1^N] = 1 - \delta(\tilde{c}_1^N, c_1^N) \quad \in \{0, 1\}$$

loss function: 0/1 = error count at string level

- **symbol error: count the symbol error in each position n (terminology: Hamming distance):**

$$L[\tilde{c}_1^N, c_1^N] = \sum_{n=1}^{N} [1 - \delta(\tilde{c}_n, c_n)] \quad \in \{0, 1, ..., N-1, N\}$$

loss function: 0/1 = error count at symbol level in string context

Strings with Synchronization

application:

- text image recognition with no segmentation problem:
  no alignment problem and therefore no deletion/insertion errors
- isolated word recognition:
  – recognize the spoken words
  – no alignment problem and therefore no deletion/insertion errors
- acoustic labelling:
  find correct labels (phonemeic/graphemic)
  for each acoustic time frame (now: 40 msec!)
- part-of-speech (POS) tagging:
  – assign word category to each (written) word
  – similar: NER = named entity recognition (first step in NLU)
  NE categories: date, time, number, organization, location, person, ...
- spelling correction (with synchronization!)

remarks:
– decision making in "context"
– good for mathematical analysis
– used in practice: POS and NER only

Bayes decision rule:

$$x_1^N \to \hat{c}_1^N(x_1^N) = \arg\min_{c_1^N} \Big\{ \underbrace{\sum_{\tilde{c}_1^N} p(\tilde{c}_1^N|x_1^N)\ L[\tilde{c}_1^N, c_1^N]}_{:= L[c_1^N|x_1^N]} \Big\}$$

re-write the posterior loss (or risk) for output string c_1^N and observation string x_1^N:

$$\begin{aligned}
L[c_1^N|x_1^N] &= \sum_{\tilde{c}_1^N} p(\tilde{c}_1^N|x_1^N)\ \sum_n [1 - \delta(\tilde{c}_n, c_n)] \\
&= \sum_n \sum_{\tilde{c}_1^N} p(\tilde{c}_1^N|x_1^N)\ [1 - \delta(\tilde{c}_n, c_n)] \ = \sum_n \sum_{\tilde{c}_n} p_n(\tilde{c}_n|x_1^N)\ [1 - \delta(\tilde{c}_n, c_n)] \\
&= \sum_n \big[1 - p_n(c_n|x_1^N)\big]
\end{aligned}$$

with the symbol posterior probability $p_n(c_n|x_1^N)$ in position n:

$$p_n(c_n|x_1^N) \ := \sum_{\tilde{c}_1^N : c_n = \tilde{c}_n} p(\tilde{c}_1^N|x_1^N)$$

question/exercise: how to compute $p_n(\tilde{c}_n|x_1^N)$ efficiently?

correct string with symbols:	$\tilde{c}_1$	$\tilde{c}_2$	...	$\tilde{c}_{n-1}$	$\tilde{c}_n$	$\tilde{c}_{n+1}$	...	$\tilde{c}_{N-1}$	$\tilde{c}_N$
	|	|	|	|	|	|	|	|	|
hypothesized string with symbols:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

the posterior risk is decomposed over the positions $n = 1, ..., N$, and the same goes for the minimization procedure:

$$\min_{c_1^N} \left\{ L[c_1^N | x_1^N] \right\} = \min_{c_1^N} \left\{ \sum_n \left[1 - p_n(c_n | x_1^N) \right] \right\}$$

$$= \sum_n \min_{c_n} \left[1 - p_n(c_n | x_1^N) \right] = \sum_n \left[1 - \max_{c_n} p_n(c_n | x_1^N) \right]$$

result: Bayes decision rule for minimum symbol error in each position n:

$$x_1^N \to \hat{c}_n(x_1^N) = \arg\max_{c_n} \left\{ p_n(c_n | x_1^N) \right\} = \arg\max_{c_n} \left\{ \sum_{\tilde{c}_1^N : c_n = \tilde{c}_n} p(\tilde{c}_1^N | x_1^N) \right\}$$

interpretation: looks like decision rule for single symbol using the symbol posterior probability for position n

Loss Functions/Decision Rules: String vs. Symbol Errors

correct string with symbols:	$\tilde{c}_1$	$\tilde{c}_2$	...	$\tilde{c}_{n-1}$	$\tilde{c}_n$	$\tilde{c}_{n+1}$	...	$\tilde{c}_{N-1}$	$\tilde{c}_N$
	|	|	|	|	|	|	|	|	|
hypothesized string with symbols:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

minimum string errors:

$$x_1^N \to \hat{c}_1^N(x_1^N) = \arg\max_{c_1^N}\{p(c_1^N|x_1^N)\} = \left[\arg\max_{c_n}\left\{\max_{\tilde{c}_1^N:\tilde{c}_n=c_n} p(\tilde{c}_1^N|x_1^N)\right\}\right]_{n=1}^N$$

minimum symbol errors:

$$x_1^N \to \hat{c}_1^N(x_1^N) = \left[\arg\max_{c_n}\{p_n(c_n|x_1^N)\}\right]_{n=1}^N = \left[\arg\max_{c_n}\left\{\sum_{\tilde{c}_1^N:\tilde{c}_n=c_n} p(\tilde{c}_1^N|x_1^N)\right\}\right]_{n=1}^N$$

remarks:

- **important conclusion: 50% rule:**
  if $\max_{c_1^N} p(c_1^N|x_1^N) > 0.5$**, the two rules must produce the same decisions.**
- **maximum approximation in machine learning:**
  often the sum can often be replaced/approximated by its maximum term.

- **no synchronization:**
  no definition of absolute positions of symbols within a string
- **typical case: ASR/HTR: strings in ASR/HTR: WER = word error rate**
  WER = edit distance = errors: ins + del + sub
  which involves a minimization procedure for the optimal alignment
- **Bayes decision rule for generating a text string c_1^N from a speech signal x_1^T:**

$$x_1^T \rightarrow \hat{c}_1^{\hat{N}}(x_1^T) = \arg\min_{N,c_1^N} \Big\{ \sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} p(\tilde{N},\tilde{c}_1^{\tilde{N}}|x_1^T)\; L[\tilde{c}_1^{\tilde{N}}, c_1^N] \Big\}$$
$$= \arg\min_{N,c_1^N} \Big\{ \sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} p(\tilde{N},\tilde{c}_1^{\tilde{N}}|x_1^T)\; \min_A L_A[\tilde{c}_1^{\tilde{N}}, c_1^N] \Big\}$$

  where A is the alignment in the edit distance L_A between $\tilde{c}_1^{\tilde{N}}$ and c_1^N
- **strings in MT: TER = translation error rate**
  TER = edit distance + swaps of symbol groups
- **mathematical analysis [Schlüter & Scharrenbach+ 05, Schlüter & Nussbaum+ 11]:**
  important property: is the loss function a metric (triangle inequality)?

given synchronization:

input vectors:	x_1	x_2	...	x_{n-1}	x_n	x_{n+1}	...	x_{N-1}	x_N
	|	|	|	|	|	|	|	|	|
output symbols:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

$$p_n(c|x_1^N) = \sum_{c_1^N : c_n = c} p(c_1^N|x_1^N)$$

missing synchronization:

input vectors:	x_1	x_2	...	...	x_{t-1}	x_t	x_{t+1}	...	...	x_{T-1}	x_T
			?		**?**		**?**		**?**		
output symbols:		c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N	

approximative synchronization using a seed string $\hat{c}_1^N = \hat{c}_1^N(x_1^T)$ (e. g transcription):

$$p_n(c|x_1^T) \;=\; ?? \quad \cong \sum_{c_1^N : c_n = c} p(c_1^N|\hat{c}_1^N, x_1^T)$$

related to refined training criteria: Povey's minimum word error rate, state-level minimum Bayes risk (sMBR), expected Bayes risk, ...

overview of classification tasks and associated errors (loss functions):

- **error for isolated event** (x, c) **:** $p(c|x)$
- **string error: strings as a whole** (x_1^T, c_1^N)**:**

$$p(c_1^N|x_1^T)$$

- **symbol errors in string context:**
  – **synchronized input-output** (x_1^N, c_1^N)**:**

$$p_n(c|x_1^N) = \sum_{c_1^N : c_n = c} p(c_1^N|x_1^N)$$

  – **non-synchronized input-output** (x_1^T, c_1^N) **using a seed string** $\hat{c}_1^N = \hat{c}_1^N(x_1^T)$

$$\begin{aligned} p_n(c|x_1^T) &= \; ?? \\ &\cong \sum_{c_1^N : c_n = c} p(c_1^N|\hat{c}_1^N, x_1^T) \end{aligned}$$

(using approximative synchronization)

open and most important question:
how to structure and define $p(c_1^N|c_1^T)$ **(see later)**

Bayes Decision Rule:
Does the exact form of the loss function matter?

- **two forms of (pseudo) Bayes decision rule for a string pair (x, c):**

$$\textbf{general loss:} \quad x \to \hat{c}(x) := \arg\min_c \Big\{ \sum_{\tilde{c}} p(\tilde{c}|x) \cdot L[\tilde{c}, c] \Big\}$$

$$\textbf{0/1 loss:} \quad x \to \hat{c}(x) := \arg\max_c \Big\{ p(c|x) \Big\}$$

- **mathematical equivalence of the two rules (see Appendix and [Schlüter & Scharrenbach$^+$ 05, Schlüter & Nussbaum$^+$ 11]):**
  - **conditions: a metric loss function $L[\tilde{c}, c]$ and**

$$\max_c p(c|x) \geq 0.5$$

  - **theoretical refinements beyond the threshold of 0.5**
  - **experimental results: hard to find a difference e. g. for high error rates: from 41% to 40%**
- **special case for edit distance: improvements beyond 50% threshold by position-dependent symbol posterior probabilities [Xu & Povey$^+$ 10, Schlüter & Nussbaum$^+$ 11]**

Summary: Bayes Decision Rule: Does the exact form of the loss function matter?

- **basis of all loss functions (in ASR, HTR, MT): count the symbol errors (with equal weights) (at level of words, subword/BPE units, letters)**
- **strings with synchronization: Hamming distance**
- **strings with no synchronization:**
  – ASR and HTR: edit distance
  – MT: TER = WER + swaps of symbol groups
  more complex: BLEU (=accuracy measure)
  property: WER and TER satisfy metric properties
- **example of non-metric loss: Hamming or edit distance with weighted errors (rarely used)**
- **conclusions for (most) practical tasks, i. e. with no weights: 0/1 loss or MAP rule is sufficient**
- **conclusions for theoretical analysis: symbol posterior probability of symbol c in position n (or "context")**

$$p_n(c|x_1^T)$$

plays a key role for strings with and without synchronization

2.3 Relation: Classification Error and Training Criteria

conceptual problem with the pseudo Bayes approach:
– basic assumption: true distribution is known
– mismatch: we replace the true distribution by a (learned) model
– question: does the optimality of Bayes decison rule still hold?

mismatch conditions for 0/1 loss ([Ney 03] and Appendix):

- **(huge) set of training data pairs:** $[x_r, c_r], \; r = 1, ...R$
  with empirical (= true) distributions, e. g. $pr(c, x) = pr(x) \cdot pr(c|x)$
- **(true) Bayes rule and associated classification error (loss)** E_***:**

$$
\begin{aligned}
x \to \hat{c}_*(x) &= \operatorname*{argmax}_c \{pr(c|x)\} \\
E_* &= \sum_x pr(x) \cdot \sum_{c: c \neq \hat{c}_*(x)} pr(c|x) \;=\; 1 - \sum_x pr(x) \cdot pr(c = \hat{c}_*(x)|x) \\
&= 1 - \sum_x pr(x) \cdot \max_c pr(c|x)
\end{aligned}
$$

- **probability model** $p_\vartheta(c|x)$ **with set of parameters** ϑ**:**
  pseudo Bayes rule and associated classification error (loss) E_ϑ**:**

$$
\begin{aligned}
x \to \hat{c}_\vartheta(x) &= \operatorname*{argmax}_c \{p_\vartheta(c|x)\} \\
E_\vartheta &= \sum_x pr(x) \cdot \sum_{c: c \neq \hat{c}_\vartheta(x)} pr(c|x) \;=\; 1 - \sum_x pr(x) \cdot pr(c = \hat{c}_\vartheta(x)|x)
\end{aligned}
$$

consider the upper bound on the squared difference between these errors [Ney 03] (= Kullback-Leibler divergence or relative entropy):

$$\begin{aligned} 1/2 \cdot \left[E_* - E_\vartheta\right]^2 &\leq \sum_{c,x} pr(c,x) \log \frac{pr(c|x)}{p_\vartheta(c|x)} \\ &= \sum_{c,x} pr(c,x) \log pr(c|x) - \sum_{c,x} pr(c,x) \log p_\vartheta(c|x) \end{aligned}$$

suitable training criterion: minimize this upper bound over ϑ:

$$\begin{aligned} 1/2 \cdot \left[E_* - E_\vartheta\right]^2 &\leq \min_\vartheta \left\{ \sum_{c,x} pr(c,x) \log \frac{pr(c|x)}{p_\vartheta(c|x)} \right\} \\ &= \sum_{c,x} pr(c,x) \log pr(c|x) - \max_\vartheta \left\{ \sum_{c,x} pr(c,x) \log p_\vartheta(c|x) \right\} \end{aligned}$$

consider the term depending on ϑ:

$$\vartheta \;\to\; F(\vartheta) \;=\; \sum_{c,x} pr(c,x) \log p_\vartheta(c|x) \;=\; \sum_r \log\, p_\vartheta(c_r|x_r)$$

using the "huge" but finite set of pairs $(c_r, x_r), r = 1, ..., R$ from which the empirical distribution $pr(c,x)$ is derived

we have derived the cross-entropy training criterion:

$$\hat{\vartheta} = \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_{c,x} pr(c,x) \log p_\vartheta(c|x) \right\} = \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_r \log p_\vartheta(c_r|x_r) \right\}$$

using the "huge" but finite set of pairs $(c_r, x_r), r = 1, ..., R$

application to strings:

- **string errors: handle strings** $[x_1^T]$ **and** $[c_1^N]$ **as a whole:** $p(c_1^N|x_1^N)$
  and factorize:

$$p(c_1^N|x_1^N) = \prod_n p(c_n|c_0^{n-1}, x_1^N)$$

- **symbols in a string context (using a seed string** $\hat{c}_1^N$**):**
  – strings with synchronization:

$$p_n(c|x_1^N) = \sum_{c_1^N : c_n = c} p(c_1^N|x_1^N)$$

  – strings without synchronization:

$$p_n(c|x_1^T) \cong \sum_{c_1^N : c_n = c} p(c_1^N|\hat{c}_1^N, x_1^T)$$

  related to (refined) training criteria: Povey's minimum word error rate,
  state-level minimum Bayes risk (sMBR), expected Bayes risk, ...

- **resulting training criterion: cross-entropy**
- **relation to classification error:**
  – provides an upper bound on pseudo Bayes classification error
  – assumption: 0/1 loss function, i. e. string errors
- **fully fledged systems in ASR (also HTR and MT):**
  – symbol errors in a string (edit distance, ...)
  – utilize a separate language model (in ASR: 100+ Mio vs. 10 Mio words)
  – training criterion: with or without language model
  – with language model: sequence discriminative training (MMI)

  (misleading) terminology:
  – criterion as such: (virtually) always cross-entropy
  – difference: type of output variable and structure of final model

three training criteria providing upper bounds on classification error (see Appendix):

- **unconstrained ANN output:** $p_\vartheta(c;x) \in \mathbb{R}$

squared error: $\hat{\vartheta} = \underset{\vartheta}{\operatorname{argmin}} \left\{ \sum_r \sum_c [p_\vartheta(c;x_r) - \delta(c,c_r)]^2 \right\}$

- **constrained ANN output:** $0 < p_\vartheta(c;x) < 1$

binary cross-entropy: $\hat{\vartheta} = \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_r \left(\log\, p_\vartheta(c_r;x_r) + \sum_{c \neq c_r} \log\,[1 - p_\vartheta(c;x_r)] \right) \right\}$

- **normalized ANN output:** $\sum_c p_\vartheta(c|x) = 1$

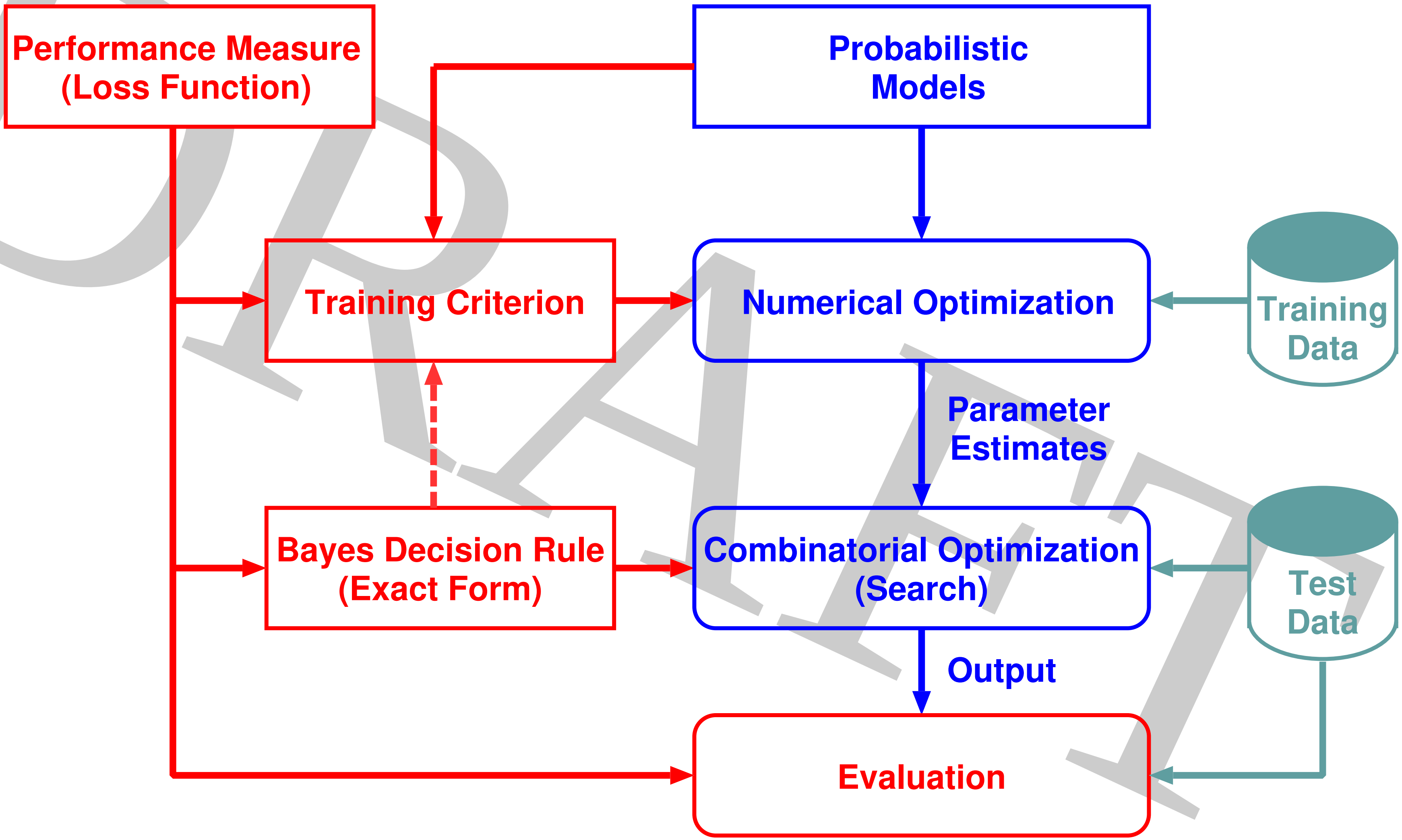
cross-entropy: $\hat{\vartheta} = \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_r \log\, p_\vartheta(c_r|x_r) \right\}$

optimal solution in all three cases for *saturated* model:

$$\hat{p}_{\hat{\vartheta}}(c|x) = pr(c|x)$$

note: applies to ANY pair $[x,c]$,
i. e. isolated events [vector,label] and [input string, output string]

Machine Learning: Statistical Decision Theory



principal ingredients:

- **performance measure, error measure, cost function:**
  – how to judge the quality of the system output
  – examples: ASR: edit distance; MT: TER or BLEU
  and associated decision rule
- **probabilistic models (with a suitable structure)**
  for capturing the dependencies within and between input and output strings:
  – given synchronization: Markov chain, CRF, (LSTM) RNN, ...
  – input/output synchronization: generative/hybrid HMM, CTC, transducer, (cross-)attention, ...
- **training criterion:**
  to learn the free model parameters from examples
  – ideally should be linked to performance criterion
  – two open questions: exact form of criterion? optimization strategy?
- **Bayes decision rule: decoder/search**
  for generating the output word sequence
  – combinatorial problem (efficient algorithms)
  – should exploit structure of models
  examples: dynamic programming and beam search, A* and heuristic search, ...
  [Vintsyuk 68, Velichko & Zagoruyko 70, Vintsyuk 71] and [Sakoe & Chiba 71], [DRAGON/HARPY 1975] and [Bridle 82, Ney 84, Ney & Haeb-Umbach$^+$ 92]

Pseudo Bayes Decision Rule: Sources of Errors:
Why does a 'Bayes' decision system make errors?

To be more exact: Why errors IN ADDITION to the so-called Bayes errors, i. e. the minimum that can be achieved?

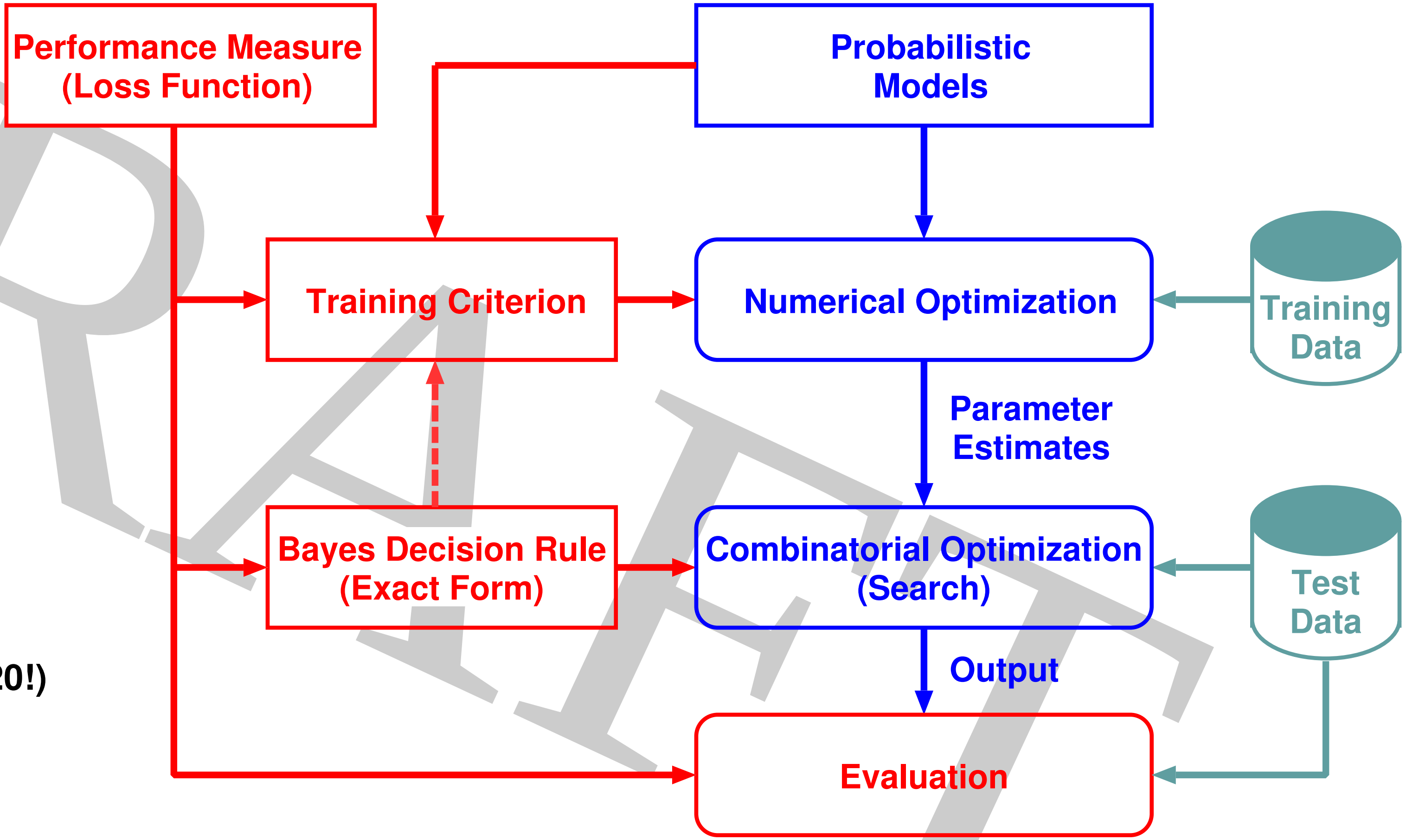
Reasons from the viewpoint of (pseudo) Bayes decision rule:

- **probability models:**
  - inadequate features/observations: e. g. spectral features vs. speech signal
  - inadequate models: e. g. acoustic model or language model
- **training conditions:**
  - poor training criterion:
    does not have strong link to performance (e.g. WER)
  - not enough training data
  - mismatch conditions between training and test data
    e. g. clean speech vs. noisy speech
- **training criterion + efficient algorithm:**
  - suboptimal algorithm for training (e. g. gradient descent)
- **decision rule:**
  - incorrect error measure, e. g. 0/1 loss
- **decision rule + efficient algorithm:**
  - suboptimal search procedure, e.g. beam search or N-best lists

Conclusions: What have we learned for ASR?

considerations (independent of ANNs and deep learning):

- exact loss function:
  – not important in testing
  – more important in training
- training criterion:
  – is important
  – depends on prob. model
- numerical optimization:
  very important (1990 vs. 2020!)
- search/generation:
  more important for
  low-accuracy conditions



- probabilistic model: most important
  – synchronization: input/output strings
  – separation of language/acoustic models

3.1 Baseline Structure: MLP

problem formulation:

- **observation (= set of measurements) is given: $x \in \mathbb{R}^D$**
- **task: find the unknown class c**
- **typical examples:**
  face recognition or recognition isolated handwritten digits

pragmatic approach: 'non-probabilistic'

- **assume a discriminant or scoring function:**

$$p_\vartheta(c, x) \in \mathbb{R}$$

  with set of free parameters ϑ (to be learned)

- **decision rule: select the class with highest score ('similarity'):**

$$x \to \hat{c}_x := \underset{c}{\operatorname{argmax}} \left\{ p_\vartheta(c, x) \right\}$$

ANNs: Isolated Events vs. Sequences

outline:

– **isolated events (no context): discriminative vs. generative modelling**
  softmax = posterior of a generative model: prior + Gaussian model
  baseline DNN = Gaussian posterior + feature extraction

following subsections:
– **model structures for seq-to-seq processing:**
  generative model, log-linear model (CRF), direct model
– **first-order models for sequence synchronization:**
  HMM, CTC, transducer, ...
– **training criteria revisited**
– **attention mechanism for synchronization**

distinguish varying conditions for decision rule:
– atomic output: no context, isolated events (here): baseline MLP
– structured output: context of a string (see later): recurrent NN

neural network approach:

- basic operation:
  matrix-vector product and component-wise nonlinearity
- concatenation of these basic operations

remarks:
– compare with discriminant function and polynomial classifiers in the 1970's
– eight layers with polynomial activation function [Ivakhnenko 71]
– overview of history by [Schmidhuber 14]

my view of the scientific challenges:

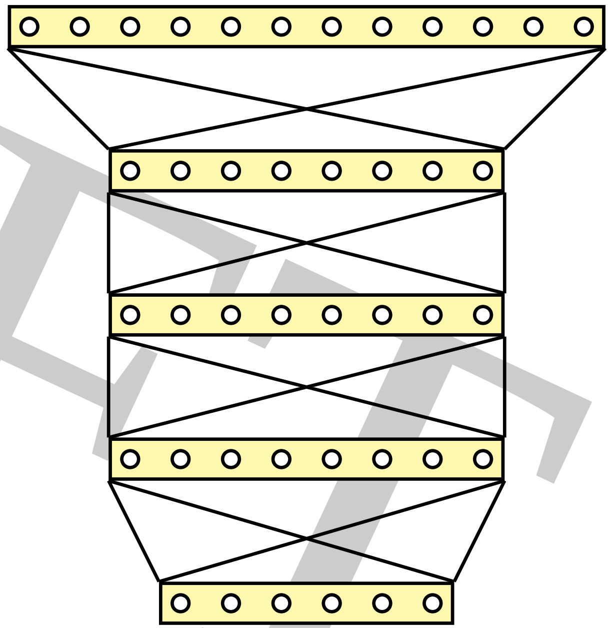
- lots of ideas, many of them 'crazy', been around for decades;
  but do they work?
- experimental challenge:
  make the ideas work in practice!
- theoretical challenge:
  what is the exact relation to classification error (or loss function)?

first renaissance of ANNs around 1986
with various interpretations/justifications:
– human/biological brain
– massive parallelism
– mathematical viewpoint:
modelling ANY input-output relation

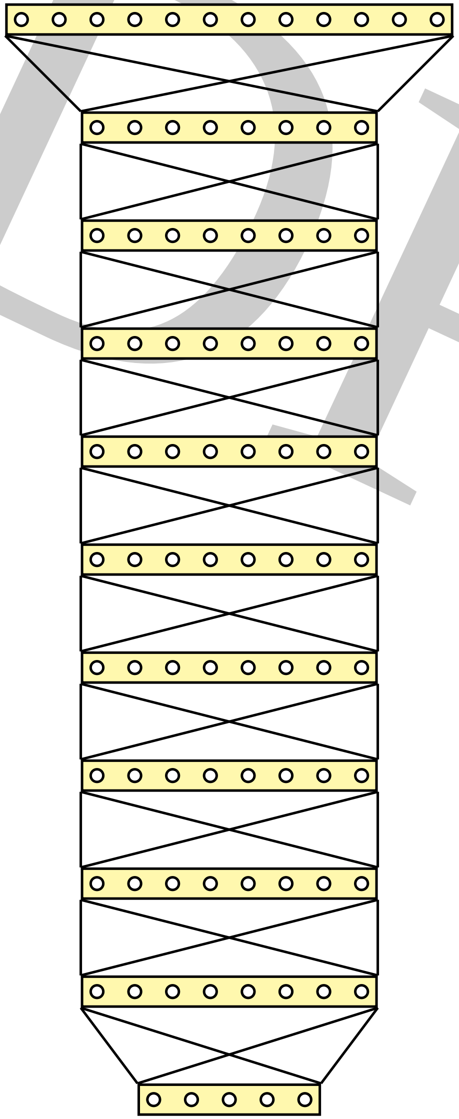
typical ANN structure:
MLP: feedforward multi-layer perceptron
with input, hidden and output layers

theoretical results (?):
one hidden layer should be sufficient (!?)

training: (hard) optimization problem with
millions of free parameters (= weights)



Artificial Neural Networks (ANN) and Deep Learning:
Second Renaissance 2011 - today



question: what is different now after 30 years?

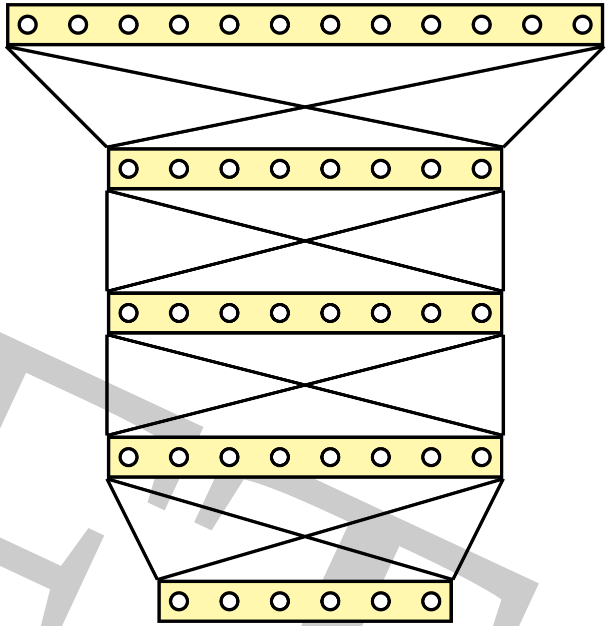
answer: we have learned how to (better) handle a complex mathematical optimization problem:

- **more powerful hardware (e. g. GPUs)**
- **empirical recipies for optimization: practical experience and heuristics, e.g. layer-by-layer pretraining**
- **result: we are able to handle more complex architectures (deep MLP, RNN, LSTM-RNN, conformer, etc.)**

Classical Architecture:
Feedforward Multi-Layer Perceptron (FF-MLP)

task: classification with observation vector $x \in \mathbb{R}^D$ and associated class c

architecture: several layers
(feedforward links only, no recurrence)



- **input layer: = observation vector x:**
  each node represents a vector component
- **from input to first hidden layer:**
  – dot product for node pair:
  = matrix-vector product for layer pair
  – nonlinear activation function
- **... add more hidden layers ...**
  using the same operations
- **from last hidden layer to output layer (like before):**
  – dot product (matrix-vector product)
  – nonlinear activation function: typically with softmax normalization
- **each output node represents a class c and its associated score $p_\vartheta(c, x)$**
  with the set ϑ of all weights (parameters) of the FF-MLP.

examples of activation functions:

- **sigmoid function (also called logistic function):**

$$u \to \sigma(u) = \frac{1}{1 + \exp(-u)} \qquad \in [0, 1]$$

- **hyperbolic tangent:**

$$u \to \tanh(u) \;=\; \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)} \qquad \in [-1, 1]$$
$$= \; 2\,\sigma(2u) - 1$$

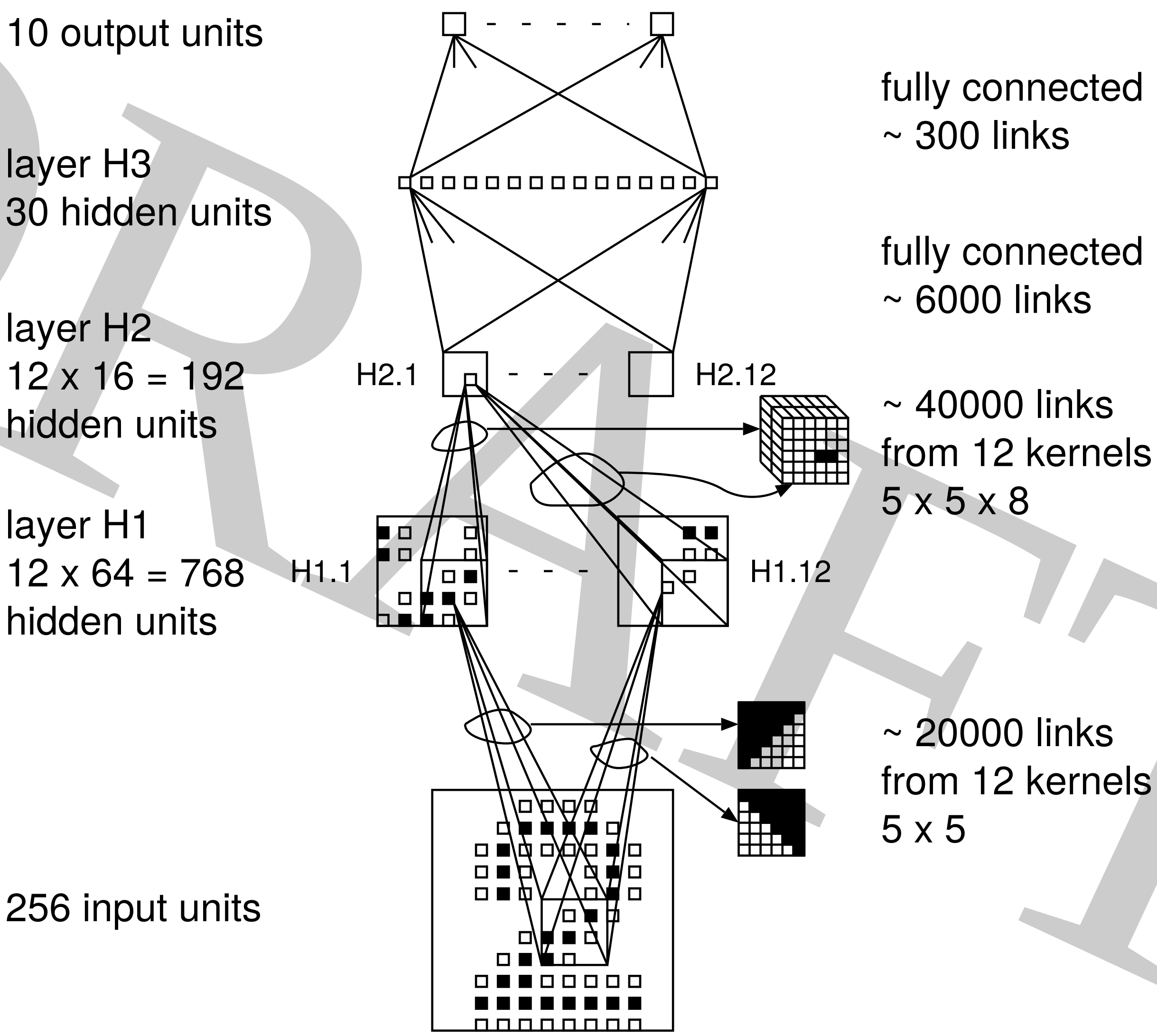
  – **in principle: no difference between** $\sigma(\cdot)$ **and** $\tanh(\cdot)$
  – **in practice: difference due to side effects**

- **rectifying linear unit (ReLU):** $u \to r(u) = \max\{0, u\}$

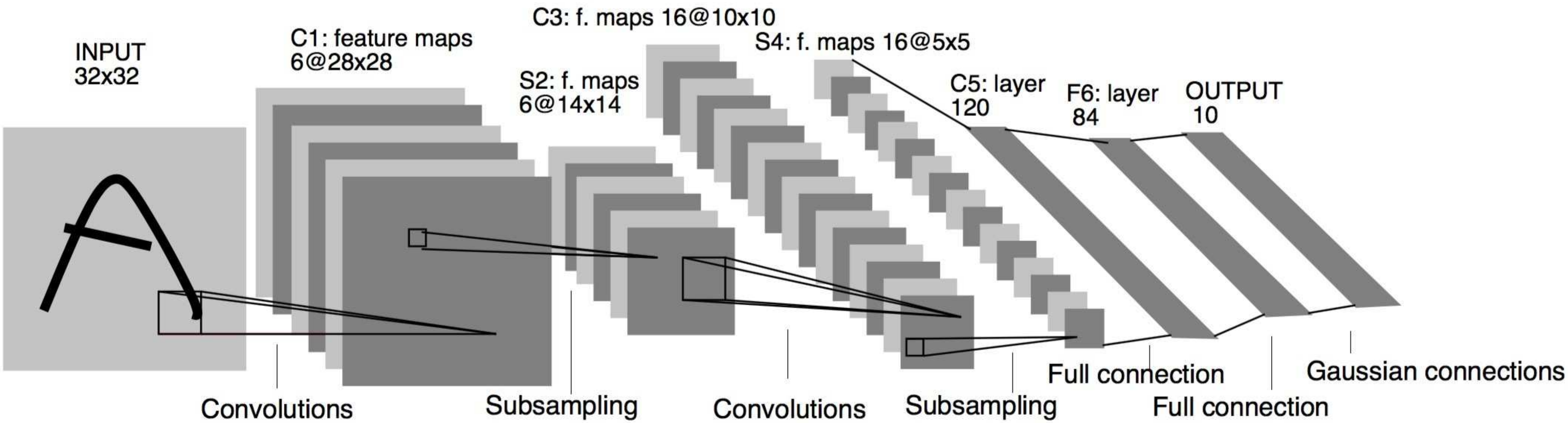
- **softmax function: generates normalized output (for probability distribution) for each node** c **of the layer under consideration (typically: output layer):**

$$u_c \to S(u_c) = \frac{\exp(u_c)}{\sum_{\tilde{c}} \exp(u_{\tilde{c}})} \qquad \text{with} \quad \sum_c S(u_c) = 1.0$$

Handwritten Digit Recognition: LeNet (Yann LeCun 1989-1996)



Handwritten Digit Recognition: LeNet (Yann LeCun 1989-1996)



3.2 Softmax: Interpretations

two interpretations:
– softmax as smoothing operation
– softmax as Gaussian posterior distribution

original terminology: smoothing of the maximum operation of real numbers $u_1^I := u_1, ...u_i, ..., u_I$

define the S function (= weighted average):

$$S_\alpha(u_1^I) := \frac{\sum_i u_i \cdot \exp(\alpha\, u_i)}{\sum_i \exp(\alpha\, u_i)} = \sum_i u_i \cdot \frac{\exp(\alpha\, u_i)}{\sum_{i'} \exp(\alpha\, u_{i'})} = \sum_i u_i \cdot w_i(\alpha; u_1^I)$$

with parameter $\alpha \in \mathbb{R}$ and the identities (verify!):

$$\begin{aligned}
\lim_{\alpha \to \infty} S_\alpha(u_1^I) &= \max_i\{u_i\} \\
S_{\alpha=0}(u_1^I) &= 1/I \cdot \sum_i u_i \\
\lim_{\alpha \to -\infty} S_\alpha(u_1^I) &= \min_i\{u_i\}
\end{aligned}$$

relevant aspects of softmax operation for ANNs:

– exact re-normalization of weights: $\sum_i w_i(\alpha; u_1^I) = 1$

– weights for $\lim \alpha \to \infty$: $w_i(\alpha; u_1^I) \to \delta(i, i_0)$ with $i_0 = \operatorname{argmax}_i u_i$

– unrelated interpretation: posterior form of a Gaussian model (see later)

distinguish:

- **structure: functional dependence**
  here: Gaussian model in a high-dim. space
- **training criterion:**
  – generative vs. discriminative
  – max.lik. vs. cross-entropy

generative modelling: prior + Gaussian model

= joint Gaussian model $p(x,c)$ for pairs (x,c) with $x \in \mathbb{R}^D$ and $c = 1, ..., C$:

$$p(x,c) = p(c) \cdot p(x|c)$$

$$p(x|c) = \mathcal{N}(x|\mu_c, \Sigma_c)$$

with class dependent mean vector $\mu_c \in \mathbb{R}^D$ and covariance matrix $\Sigma_c \in \mathbb{R}^{D \cdot D}$

$$= \frac{1}{\sqrt{\det(2\pi\Sigma_c)}} \cdot \exp\Big(-\frac{1}{2}(x-\mu_c)^t \Sigma_c^{-1}(x-\mu_c)\Big)$$

$$= \frac{1}{\sqrt{\det(2\pi\Sigma_c)}} \cdot \exp\Big(-\frac{1}{2}x^t \Sigma_c^{-1}x + \mu_c^t \Sigma_c^{-1} x - \frac{1}{2}\mu_c^t \Sigma_c^{-1}\mu_c\Big)$$

with superscript t: $x^t :=$ tranpose of a vector $x \in \mathbb{R}^D$

characteristic property of Gaussian model:

generalized (squared) Euclidean distance as argument of exponential function

maximum likelihood training:

- **joint (or generative) model:** $p_\vartheta(x, c)$
- **training data** $[x_r, c_r],\ r = 1, ..., R$**:**
- **training criterion:**

$$\max_\vartheta \left\{ \sum_r \log\, p_\vartheta(x_r, c_r) \right\} = \max_\vartheta \left\{ \sum_r \log\, p_\vartheta(c_r) + \sum_r \log\, p_\vartheta(x_r|c_r) \right\}$$

typically: independent parameters for $p(c)$ **and** $p(x|c)$

- **Gaussian model:**

$$\vartheta := \left\{ p(c) \in \mathbb{R}, \mu_c \in \mathbb{R}^D, \Sigma_c \in \mathbb{R}^{D \cdot D} :\ c = 1, .., C \right\}$$

resulting optimization: independently for each class c **(apart from** $p(c)$**!)**

estimates of parameters for a Gaussian model:
– $p(c)$**: relative frequency**
– μ_c**: empirical mean vector**
– Σ_c**: empirical covariance matrix**

Class Posterior Probability of a Gaussian Model

discriminative modelling:

class posterior probability using class priors $p(c)$:

$$
\begin{aligned}
p_\vartheta(c|x) &= \frac{p(c) \cdot \mathcal{N}(x|\mu_c, \Sigma_c)}{\sum_{c'} p(c') \cdot \mathcal{N}(x|\mu_{c'}, \Sigma_{c'})} \\
&= \frac{\frac{p(c)}{\sqrt{\det(2\pi\Sigma_c)}} \exp(-\frac{1}{2}(x-\mu_c)^t \Sigma_c^{-1}(x-\mu_c))}{\sum_{c'} \frac{p(c')}{\sqrt{\det(2\pi\Sigma_{c'})}} \exp(-\frac{1}{2}(x-\mu_{c'})^t \Sigma_{c'}^{-1}(x-\mu_{c'}))} \\
&= \frac{\exp(-\frac{1}{2}x^t\Sigma_c^{-1}x + \mu_c^t\Sigma_c^{-1}x - \frac{1}{2}\mu_c^t\Sigma_c^{-1}\mu_c - \frac{1}{2}\log\det(2\pi\Sigma_c) + \log p(c))}{\sum_{c'} \exp(-\frac{1}{2}x^t\Sigma_{c'}^{-1}x + \mu_{c'}^t\Sigma_{c'}^{-1}x - \frac{1}{2}\mu_{c'}^t\Sigma_{c'}^{-1}\mu_{c'} - \frac{1}{2}\log\det(2\pi\Sigma_{c'}) + \log p(c'))} \\
&= \frac{\exp(x^t\Lambda_c x + \lambda_c^t x + \alpha_c)}{\sum_{c'} \exp(x^t\Lambda_{c'} x + \lambda_{c'}^t x + \alpha_{c'})} = \frac{1}{Z_\vartheta(x)} \cdot \exp\left(\alpha_c + \lambda_c^t x + x^t \Lambda_c x\right)
\end{aligned}
$$

with the normalization term $Z_\vartheta(x) (= p_\vartheta(x))$
and the parameters:

$$\vartheta := \{\alpha_c \in \mathbb{R}, \lambda_c \in \mathbb{R}^D, \Lambda_c \in \mathbb{R}^{D \cdot D}\}$$

remark: matrices Λ_c are defined to be symmetric!

summary of re-writing the Gaussian posterior:

$$p_\vartheta(c|x) \;=\; \frac{p_\vartheta(c)\cdot p_\vartheta(x|c)}{\sum_{c'} p_\vartheta(c')\cdot p_\vartheta(x|c')} \;=\; \frac{\exp(x^t\Lambda_c x + \lambda_c^t x + \alpha_c)}{\sum_{c'}\exp(x^t\Lambda_{c'} x + \lambda_{c'}^t x + \alpha_{c'})}$$

- EXACT equivalence [IEEE TASLP, Heigold & Ney 12] between
  – posterior form of joint Gaussian model
  – log-linear model with quadratic observations (features)

- important consequence: for the same model,
  we can have different training criteria with training data $[x_r, c_r],\; r = 1, ..., R$:
  – joint generative model: maximum likelihood:

$$\max_\vartheta\left\{\sum_r \log\, p_\vartheta(x_r, c_r)\right\} = \max_\vartheta\left\{\sum_r \log\, p_\vartheta(c_r) + \sum_r \log\, p_\vartheta(x_r|c_r)\right\}$$

  closed form solution: rel. freq., empirical mean vector, empirical covariance matrix

  – class posterior of the model: (optimum) cross-entropy:

$$\max_\vartheta\left\{\sum_r \log\, p_\vartheta(c_r|x_r)\right\}$$

  no closed form solution, but convex optimization problem

  warning: often misleading terminology

definitions for a joint model: $p(x, c) = p(c) \cdot p(x|c)$

$$p(c|x) = \frac{p(c) \cdot p(x|c)}{\sum_{c'} p(c') \cdot p(x|c')} = \frac{p(c) \cdot \tilde{p}(c|x)}{\sum_{c'} p(c') \cdot \tilde{p}(c'|x)} \quad \text{with} \quad \tilde{p}(c|x) := \frac{p(x|c)}{\sum_{c'} p(x|c')}$$

concept of pseudo posterior: assume a uniform prior: $p(c) = \dfrac{1}{C}$

pseudo posterior of a Gaussian model (i.e. with a uniform prior):

$$\tilde{p}_\vartheta(c|x) = \frac{\frac{1}{C} \cdot \mathcal{N}(x|\mu_c, \Sigma_c)}{\sum_{c'} \frac{1}{C} \cdot \mathcal{N}(x|\mu_{c'}, \Sigma_{c'})} = \frac{\mathcal{N}(x|\mu_c, \Sigma_c)}{\sum_{c'} \mathcal{N}(x|\mu_{c'}, \Sigma_{c'})}$$

$$= \frac{\exp\left(-\frac{1}{2}x^t\Sigma_c^{-1}x + \mu_c^t\Sigma_c^{-1}x - \frac{1}{2}\mu_c^t\Sigma_c^{-1}\mu_c - \frac{1}{2}\log\det(2\pi\Sigma_c)\right)}{\sum_{c'} \exp\left(-\frac{1}{2}x^t\Sigma_{c'}^{-1}x + \mu_{c'}^t\Sigma_{c'}^{-1}x - \frac{1}{2}\mu_{c'}^t\Sigma_{c'}^{-1}\mu_{c'} - \frac{1}{2}\log\det(2\pi\Sigma_{c'})\right)}$$

comparison with regular posterior $p_\vartheta(c|x)$**:**
– both posteriors have bias terms
– difference in bias terms: $\log\, p(c)$

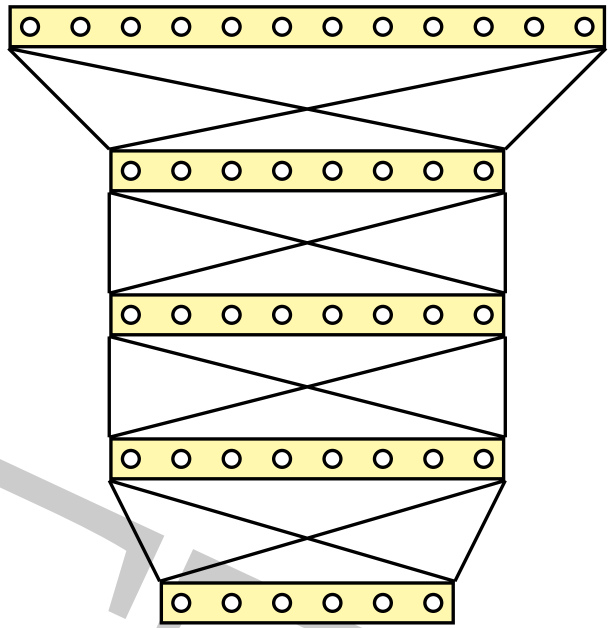
pooled covariance matrix $\Sigma_c = \Sigma$: some terms cancel

$$p_\vartheta(c|x) \;=\; \frac{\exp(\mu_c^t \Sigma^{-1} x - \frac{1}{2}\mu_c^t \Sigma^{-1}\mu_c + \log p(c))}{\sum_{c'} \exp(\mu_{c'}^t \Sigma^{-1} x - \frac{1}{2}\mu_{c'}^t \Sigma^{-1}\mu_{c'} + \log p(c'))}$$

$$= \frac{\exp(\lambda_c^t x + \alpha_c)}{\sum_{c'} \exp(\lambda_{c'}^t x + \alpha_{c'})}$$

important result:
– log-linear dependence on parameter vectors λ_c and bias α_c
– exact equivalence: = softmax with weight vectors λ_c and bias α_c

conventional view: consider MLP with softmax output:



- input layer: raw input vector z
- hidden layers perform feature extraction:

$$x = f(z)$$

with feature vector $x \in \mathbb{R}^D$ before output layer
note: no dependence on class labels $c = 1, ..., C$

- output layer: probability distribution over classes c

$$p(c|x) = \frac{\exp(\alpha_c + \lambda_c^t \cdot x)}{\sum_{c'} \exp(\alpha_{c'} + \lambda_{c'}^t \cdot x)}$$

with output layer weights $\lambda_c \in \mathbb{R}^D$ and offsets (biases) $\alpha_c \in \mathbb{R}$

exact equivalence (for pooled covariance matrix):

ANN with softmax = Gaussian posterior + feature extraction

3.3 Training Data and Empirical Distribution

empirical distribution of a (representative) sample of pairs $(x_n, c_n),\ n = 1, ..., N$:

$$pr(x,c) \ := \ \frac{1}{N}\sum_{n=1}^{N} \delta(x,x_n)\,\delta(c,c_n)$$

– with $\delta(\cdot,\cdot)$ being Kronecker delta (or Dirac delta function for $x \in \mathbb{R}^D$)
– interpretations: histogram (or big table), counting, kernel density

consider a quantity $h(x,c)$ and its expectation value $E\{h(x,c)\}$:

$$\begin{aligned} E\{h(x,c)\} = E_{\{pr(x,c)\}}\{h(x,c)\} \ &= \ \sum_x \sum_c pr(x,c)\ h(x,c) \\ &= \ \sum_x \sum_c \frac{1}{N}\sum_{n=1}^{N} \delta(x,x_n)\,\delta(c,c_n)\ h(x,c) \\ &= \ \frac{1}{N}\sum_{n=1}^{N} \ h(x_n,c_n) \ = \ \text{empirical average of } h(x,c) \end{aligned}$$

note: the empirical distribution summarizes EVERYTHING about the training data

definition: joint distribution of pairs (x, c):

$$pr(x,c) = \frac{1}{N} \cdot \sum_{n=1}^{N} \delta(c, c_n) \cdot \delta(x, x_n)$$

derived types of distributions (as for *any* distribution):

- **marginal distribution over observations x:**

$$pr(x) = \sum_{c} pr(x,c) = \frac{1}{N} \cdot \sum_{n=1}^{n} \delta(x, x_n)$$

- **marginal distribution over classes c:**

$$pr(c) = \sum_{x} pr(x,c) = \frac{1}{N} \cdot \sum_{n=1}^{N} \delta(c, c_n)$$

  other name: class prior (or language model in ASR)

- **(class) posterior distribution for x with $pr(x) > 0$:**

$$pr(c|x) = pr(x,c)/pr(x)$$

- **class conditional distribution over observations for c with $pr(c) > 0$:**

$$pr(x|c) = pr(x,c)/pr(c)$$

Empirical Distribution: Remarks on the Concept

advantage of using empirical distribution:

- **compact representation of the data: training or test data**
- **avoid the need to worry about asymptotic limits**
- **interpretation:**
  - **discrete observations: histogram or counting approach**
  - **continuous-valued observations: Dirac delta function in lieu of Kronecker delta smoothed variant: concept of kernel densities**
  - **strings: concept works too (maybe not practical)**
- **implementation:**
  - **easy for discrete and 'atomic' variables**
  - **hard for continuous-valued observations and strings**

- **conceptual view of decision rule:**

  convert training pairs (=relations) $(x_n, c_n), n = 1, ..., N$
  to a functional dependence: $(x_n, \hat{c}_n = \hat{c}_{x_n})$
  using a suitable decision rule: $x \rightarrow c(x)$ **(or** $\hat{c}_x$**)**

3.4 Training Criteria and Probabilistic Interpretation

we consider a generalized ANN, i.e. any discriminant or scoring function

ANN outputs and associated training criteria:

- **unconstrained output:** $p_\vartheta(c, x) \in \mathbb{R}$
  using squared error
- **constrained output:** $p_\vartheta(c, x) \in [0, 1]$
  using binary cross-entropy
- **normalized output:** $p_\vartheta(c|x) = p_\vartheta(c, x) \in [0, 1]$ **with** $\sum_c p_\vartheta(c|x) = 1.0$
  using cross-entropy

preview of important results:

- **ANN outputs are estimates of class posterior probabilities**
- **best possible optimum (i.e. global!) for ANN output:**

$$\hat{p}_\vartheta(c, x) = pr(c|x)$$

conditions:
– the ANN has enough degrees of freedom
– the global optimum is found

distinguish three types of ANN or discriminant outputs:

- **unconstrained output: scoring function for $x \in \mathbb{R}^D$: example**

$$g_\vartheta(c,x) \;=\; \alpha_c + \lambda_c^T x + x \Lambda_c x^T$$

  – **example: quadratic in x and linear in parameters $\vartheta = \{\alpha_c \in \mathbb{R}, \lambda_c \in \mathbb{R}^D, \Lambda_c \in \mathbb{R}^{D \cdot D}\}$**
  – **general case: polynomial discriminant function [Ivakhnenko 71]**

- **constrained output: logistic regression (ANN: sigmoid function):**

$$p_\vartheta(c,x) \;=\; \frac{1}{1 + \exp\big[-g_\vartheta(c,x)\big]}$$

- **normalized output: log-linear model (ANN: softmax):**

$$p_\vartheta(c|x) \;=\; \frac{\exp\big[g_\vartheta(c,x)\big]}{\sum_{c'} \exp\big[g_\vartheta(c',x)\big]}$$

note: the decision output does not change:

$$x \to c_\vartheta(x) = \operatorname*{argmax}_c g_\vartheta(c,x) = \operatorname*{argmax}_c p_\vartheta(c,x) = \operatorname*{argmax}_c p_\vartheta(c|x)$$

Remark: Parametric vs. Non-Parametric Approaches

distinguish two types of modelling approaches:

- ***parametric* approach: model based approach**
  with a specific functional dependence and 'some' parameters ϑ:

$$\hat{\vartheta} \;=\; \operatorname*{argmin}_{\vartheta} F(\vartheta)$$

  examples: discriminant functions
  incl. ANNs and posterior forms of generative models (see later)

- ***non-parametric* approach: = table of probabilities (counting approach):**
  free parameters are the entries of the table,
  i. e. the probabilites (or their estimates) themselves:

$$\hat{p}_{\hat{\vartheta}}(c,x) \;=\; \operatorname*{argmin}_{\{p_\vartheta(c,x)\}} F(\vartheta) \;=\; pr(c|x)$$

  The same result is obtained for a 'fully saturated' model,
  i.e. a model with a flexible structure and sufficient number of free parameters.

Training Criterion: Cross-Entropy

assumption: normalized ANN outputs (e.g. by softmax):

$$p_\vartheta(c|x) > 0 \qquad \sum_c p_\vartheta(c|x) = 1.0$$

note: different notation to express normalization

criterion: maximize the logarithm of the model posterior probability $p_\vartheta(c_n|x_n)$ for labeled training data $[x_n, c_n],\ n = 1, ..., N$:

$$\begin{aligned} F(\vartheta) := \frac{1}{N}\sum_{n=1}^{N} \log\, p_\vartheta(c_n|x_n) &= \sum_{c,x} pr(c,x)\, \log\, p_\vartheta(c|x) \\ &= \sum_x pr(x) \sum_c pr(c|x)\, \log\, p_\vartheta(c|x) \end{aligned}$$

using the empirical distribution of the training data (as above):

$$pr(c,x) := \frac{1}{N}\sum_{n=1}^{N} \delta(c,c_n)\delta(x,x_n)$$

Training Criterion: Cross-Entropy

criterion is equivalent to (negative) cross-entropy:

$$F(\vartheta) \;=\; \sum_{c,x} pr(c,x)\,\log\, p_\vartheta(c|x)$$

training criterion for parameter ϑ :

$$\hat{\vartheta} \;=\; \operatorname*{argmax}_{\vartheta} F(\vartheta)$$

global optimum for the ANN $p_\vartheta(c|x)$ as a whole:

$$\hat{p}_{\hat{\vartheta}}(c|x) \;=\; \operatorname*{argmax}_{\{p_\vartheta(c|x)\}} F(\vartheta) \;=\; pr(c|x)$$

note: normalized output is required!

consider the equivalent maximization problem:

$$\begin{aligned}\Delta F(\vartheta) &= \sum_x pr(x) \sum_c pr(c|x) \log \frac{p_\vartheta(c|x)}{pr(c|x)} \\ &= \sum_x pr(x) \Big[\sum_c pr(c|x) \log p_\vartheta(c|x) - \sum_c pr(c|x) \log pr(c|x) \Big]\end{aligned}$$

terminology: (Kullback-Leibler) divergence between true and model distribution

terminology from information theory:

(self) entropy: $-\sum_c pr(c|x) \log pr(c|x)$

cross-entropy: $-\sum_c pr(c|x) \log p_\vartheta(c|x)$

proof: use divergence inequality,
which holds for any two distributions p_c and q_c:

$$\sum_c p_c \log q_c \leq \sum_c p_c \log p_c$$

Divergence Inequality: Minimalist View
(useful for many optimzation problems)

consider two distributions p_c, q_c over a random variable c

divergence inequality:

$$\sum_c p_c \log \frac{q_c}{p_c} \ \le \ 0$$

proof: use tangent of the logarithmic function $u \to \log u$ in $u = 1$

rewrite the divergence inequality:

$$\sum_c p_c \log q_c \le \sum_c p_c \log p_c$$

therefore:

$$\max_{\{q_c\}} \left\{ \sum_c p_c \log q_c \right\} = \sum_c p_c \log p_c$$
$$\hat{q}_c = p_c$$

- **(usual assumption): normalized model** $(q(c|x)$
- **tentative training criterion:**
  drop the logarithm in cross-entropy criterion:

$$F(\{q(c|x)\}) = 1/N \cdot \sum_n q(c_n|x_n) = \sum_x pr(x) \sum_c pr(c|x)\, q(c|x)$$

$$\hat{q}(c|x) := \underset{\{q(c|x)\}}{\operatorname{argmax}} \left\{ F(\{q(c|x)\}) \right\}$$

$$= \delta(c_*(x), c)$$

with $\quad c_*(x) = \underset{c}{\operatorname{argmax}} \{pr(c|x)\}$

- **result: pathological estimate, which is not interesting**

model $q(c, x) \in \mathbb{R}$ with quadratic normalization:

$$\sum q^2(c, x) = 1$$

tentative training criterion for model $q(c, x)$:

$$F(\{q(c,x)\}) := 1/N \cdot \sum_n q(c_n, x_n) = \sum_x pr(x) \sum_c pr(c|x)\, q(c,x)$$

$$\hat{q}(c|x) := \underset{\{q(c,x)\}}{\operatorname{argmax}} \Big\{ F(\{q(c,x)\}) \Big\} = \underset{\{q(c,x)\}}{\operatorname{argmax}} \Big\{ \sum_x pr(x) \sum_c pr(c|x)\, q(c,x) \Big\}$$

use Cauchy inequality:

$$\sum_c pr(c|x) \cdot q(c,x) \le \sqrt{\sum_c pr^2(c|x)} \cdot \sqrt{\sum_c q^2(c,x)}$$

with equality for : $q(c,x) = \gamma \cdot pr(c|x) \quad \gamma \in \mathbb{R}$

$$= \frac{pr(c|x)}{\sqrt{\sum_{\tilde{c}} pr^2(\tilde{c}|x)}}$$

result: estimate is the true distribution with quadratic normalization!

Training Criteria: Summary

important result:

best possible ANN outputs: true posterior probabilities $pr(c|x)$ of the training corpus

three training criteria:
– squared error for unconstrained ANN outputs
– binary cross-entropy for contrained ANN outputs
– cross-entropy for normalized ANN outputs

remarks:

- **result is independent of any specific structure of the ANN, i. e. correct for any *discriminant function***
- **assumption: sufficient flexibility and parameters in the ANN**
- **result independent of any training strategy (e.g. type of backpropagation)**
- **generalization capability from training to test set: not addressed**

open questions:
– what is the relation with the classification error?
– how can we achieve the *minimum* classification error?

relation between error rate and training criteria?

- **we need a strict distinction:**
  – error rate for the true distribution: Bayes classification error E_*
  – error rate for the learned distribution: model classification error E_ϑ
  (model distribution with parameters ϑ)

- **we will prove later [Ney 03]:**
  each of the three training criteria gives a tight upper bound to the squared difference between these two error rates

  example: Kullback-Leibler divergence for cross-entropy criterion

$$\left(E_* - E_\vartheta\right)^2 \leq 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \cdot \log \frac{pr(c|x)}{p_\vartheta(c|x)}$$

Empirical Distribution: Generalization

interpretation of true distribution $pr(c|x)$ or $pr(x,c)$:

- **central role: empirical distribution,**
  – it is non-zero only for the observed pairs (x_n, c_n) in the corpus
  – for unseen pairs (x,c): $pr(x,c) = 0$
  – for unseen observations x: $pr(x) = 0$

- **assumption for future experiments:**
  approach should work for all inputs x

- **the training corpus can cover only a very very tiny fraction of all possible inputs**
  consider the example of speech recognition:
  1 sec of audio = 100 10-ms frames, each frame with a vector $x \in \mathbb{R}^{D=50}$:
  number of possible input strings: $\left(\left(2^8\right)^{50}\right)^{100} = 2^{8 \cdot 50 \cdot 100} = 2^{40.000} \cong 10^{12.000}$

concept of generalization:

- **we want to define a model distribution $p_\vartheta(c|x)$**
  for all future possible observations x

- **to that purpose:**
  – we learn a model distribution from a representative corpus: training data
  – we want to generalize to regions not seen in the training data
  – therefore we need some structure in the model distributions $p_\vartheta(c|x)$

generalization: how well does the learned model work on new unseen data?

- **regularization of weights: include an additive penalty in the**

$$F\Big(\{\vartheta_i\};[c,x]_1^N\Big) \;+\; \gamma \cdot \sum_i \vartheta_i^2$$

interpretation:
– large weights should be avoided
– Bayesian interpretation: Gaussian prior for weights with zero means
terminology in backpropagation: (ANN) weight decay

- **Tikhonov regularization [Tikhonov & Arsenin 77]:**
– outputs should be smooth in continuous-valued $x_n \in \mathbb{R}^D$
– include a penalty term:

$$F\Big(\{\vartheta_i\};[c,x]_1^N\Big) \;+\; \gamma \cdot \sum_n \sum_c \Big|\Big|\frac{\partial p_\vartheta(c,x_n)}{\partial x_n}\Big|\Big|^2$$

– successful in image restoration (i.e. outside ANN)
– discussed for ANNs by [Bishop 95b]
– rarely used, but see for ASR [Chien & Lu 15]

distinguish strictly:

- **training criterion as such**
- **optimization strategies:**
  – one category: gradient search
  – one subcategory: *backpropagation*
  (as opposed to other variants, e.g. second-order methods)
  based on chain rule of calculus for derivatives

gradient search (incl. backpropagation):

- **we can only find a LOCAL optimum**
- **there may be a huge number of local optima;**
  but most of them seem to be equivalent
- **experimental evidence: backpropagation is able to find**
  a local optimum that is typically ’good enough’

- **generalization capability:**
  implicitly taken into account by cross-validation (early stopping) ?

present variants of optimization strategy:
– backpropagation
– drop-out
– weight decay (= regularization of weights)
– concept of minibatches
– momentum term (recursive smoothing of gradient)
– ADAM: adaptive momentum estimation [Kingma & Ba, ICLR 2015]
– early stopping and cross-validation
– ...

note: there have been hundreds of variants

situation today:
– dominated by trial and error
– lack of a consistent theory

4 ANNs: Sequence Modelling

- **this section: from single events to sequences**
- **strings with synchronization: $[\boldsymbol{x}_1^N, \boldsymbol{c}_1^N]$**
  consequence: length N is given!
- **question: how to model sequences?**
  how to apply cross-entropy training?
- **string context:**
  what does an ANN learn?

4.1 Distribution Modelling for Sequences

so far: handling of (input, output) pairs (c, x) in isolation:
no internal structure in c or x (unlike sequences)

from single events to sequences $[x_1^N, c_1^N]$
– basic event: sentence $[x_1^N, c_1^N]$
– basic distribution (true or model): $p(c_1^N|x_1^N)$

- we consider a pair of synchronized input and output sequence over time t:

$$(x_n, c_n), \ n = 1, ..., N$$

  with input vectors (or symbols) x_n and class labels c_n (with known string length N!)

- illustration:

input vectors:	x_1	x_2	...	x_{n-1}	x_n	x_{n+1}	...	x_{N-1}	x_N
	|	|	|	|	|	|	|	|	|
output symbols:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

model with 1:1 correspondence
between class labels c_1^T and observations x_1^T

input vectors:	x_1	x_2	...	x_{n-1}	x_n	x_{n+1}	...	x_{N-1}	x_N
	|	|	|	|	|	|	|	|	|
output symbols:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

typical problems:
– spelling correction: for letter substitutions only
– POS tagging (POS: parts of speech = word categories) and
  semantic tagging for NLU
– frame labelling in ASR (incl. pronunciation and language models!)
  and acoustic scores in hybrid HMMs
– recognition problems with no problems of boundary detection:
  isolated words, printed character recognition, ...

Sequences: Notation

- single events: basic joint event

$$[c, x] \quad \text{with} \quad p(c, x)$$

with training data: pairs of single [input,output] events

$$[x_r, c_r], \quad r = 1, ...R$$

- sequences: basic joint event = sequence pair:

$$[c_1^N, x_1^N] = [c_n, x_n]_{n=1}^N \qquad p(c_1^N, x_1^N)$$

with training data: pairs or [input,output] sequences

$$[x_1^N, c_1^N]_s, \;\; s = 1, ..., S \quad \text{or} \quad [x_s, c_s]_{n=1}^N, \;\; s = 1, ..., S$$

- (tacit) assumption: time stationarity:
  – no explicit dependence on absolute position n
  – dependence on relative position difference: yes
  – (sloppy) notation:

$$p_n(c_n, x_n) = p(c_n, x_n) = p_n(c, x)$$

Sequence: Cross-Entropy

re-structuring the sequence distribution:
standard method for ssquences (factorization into conditional probablilities):

$$p(c_1^N|x_1^N) = \prod_{n=1}^{N} p(c_n|c_0^{n-1}, x_1^N)$$

remarks:
- **artificial start symbol c_0**
- **decomposition over elementary distributions that can be modelled by softmax ANN outputs**

training criterion:
- **cross-entropy for the full sequence model $p(c_1^N|x_1^N)$**
- **training data: empirical distribution over string pairs:** $[c_1^N, x_1^N]_s, \; s = 1, ..., S$
- **simplifying assumption: same length N for all sequences**

interpretation:
- **input: full sequence is given**
- **output: output is processed from left-to-right**
- **result: like 'usual' cross-entropy training criterion**

illustration: consider context windows in c_1^N for a position n:

sequence	○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ □ ○ ○ ○ ○ ○ ○ ○ ○ ○
unidirect.	● ● ● ● ● ● ● ● ● ● ● □ . . . .
bidirect.	● ● ● ● ● ● ● ● ● ● ● □ ● ● ● ● ● ● ● ● ●
sym. window	. . . . ● ● ● □ ● ● ● . . . . . .

associated distributions (maybe difficult to compute!):

– **full sequence:** $p(c_1^N|x_1^N) = \prod_n p(c_n|c_0^{n-1}, x_1^N)$
– **unidirectional or full left context:** $p(c_n|c_0^{n-1}, x_1^N)$
– **bidirectional or full context:** $p(c_n|c_1^{n-1}, c_{n+1}^N, x_1^N)$
– **symmetric window:** $p(c_n|c_{n-k}^{n-1}, c_{n+1}^{n+k}, x_1^N)$
special case $k = 0$: no context: $p(c_n|x_1^N)$

input sequence x_1^N is handled in a similar way

general model to be studied:

$$p(c_1^N|x_1^N) = \prod_n p(c_n|c_{n-k}^{n-1},\, x_{n-m}^{n+m})$$

illustration: context windows around a position n:

c_1^N	○	○	○	●	●	●	●	●	●	●	●	□	.	.	.	.	.	.	.	.	.
x_1^N	○	○	○	○	○	●	●	●	●	●	●	●	●	●	●	●	●	●	○	○	○

remarks: context windows

– **size of windows can vary:**
from a few words/symbols to one or several sentences/sequences!

– **windows might extend beyond sentence/sequence boundaries**
(see next slides), which requires special treatment

– **interpretation: decomposition of input/output sequence**
into overlapping blocks $[c_{n-k}^{n},\, x_{n-m}^{n+m}]$

4.2 Cross-Entropy at Sequence Level

model with limited contexts:

$$p(c_1^N | x_1^N) = \prod_n p(c_n | c_{n-k}^{n-1}, x_{n-m}^{n+m})$$

illustration (with 4 sentence pairs):
with a sentence boundary symbol ■
(maybe different handling for input and output)

■ ○ ○ ○ ○ ○ ○ ○ ○ ■ ○ ○ ○ ■ ○ ○ ○ ○ ○ ○ ○ ○ ○ ■ ○ ○ ○ ○ ○ ■

cross-entropy criterion:

$$\begin{aligned} F(\{p\}) &= \sum_s \log p\Big([c_s]_1^N \Big| [x_s]_1^N]\Big) \;=\; \sum_s \log \prod_n p\Big([c_{sn} \Big| [c_s]_{n-k}^{n-1}, [x_s]_1^N]\Big) \\ &= \sum_s \sum_n \log p(c_{sn} | [c_s]_0^{n-1}, [x_s]_1^N) \end{aligned}$$

(assume model with limited contexts)

$$= \sum_s \sum_n \log \; p\Big(c_{sn} \Big| [c_s]_{n-k}^{n-1}, [x_s]_{n-m}^{n+m}\Big)$$

Training: Cross-Entropy

model with limited context (for input and output):

$$p(c_n|c_{n-k}^{n-1}, x_{n-m}^{n+m})$$

training criterion: cross-entropy:

- **set of sequence pairs $s = 1, .., S$:** $[x_s, c_s]_{n=1}^{N_s}$

$$F(p) \;=\; \sum_{s=1}^{S} \log\, p\big([c_s]_1^{N_s} \big| [x_s]_1^{N_s}\big) \;=\; \sum_{s=1}^{S} \sum_{n=1}^{N_s} \log\, p\big(c_{sn} \big| [c_s]_{n-k}^{n-1}, [x_s]_{n-m}^{n+m}\big)$$

- **a single super-sequence pair (after concatenation):** $[x_1^N, c_1^N]$

$$F(p) \;=\; \log \prod_{n=1}^{N} p(c_1^N|x_1^N) \;=\; \sum_{n=1}^{N} \log\, p\big(c_n \big| c_{n-k}^{n-1}, x_{n-m}^{n+m}\big)$$

note the formal similarity!

4.3 Sequences: Stationarity and Empirical Distribution

stationarity:
– (always) tacit assumption
– what does it mean for ANNs?
– related to the definition of empirical distribution for sequences

general model to be studied:

$$p(c_1^N | x_1^N) = \prod_n p(c_n | c_{n-k}^{n-1}, x_{n-m}^{n+m})$$

illustration: context windows around a position n:

c_1^N	○	○	○	●	●	●	●	●	●	●	●	□	.	.	.	.	.	.	.	.	.
x_1^N	○	○	○	○	○	●	●	●	●	●	●	●	●	●	●	●	●	●	○	○	○

concept:
– to simplify the presentation: synchronized input/output sequence pairs
– imagine a concatenation of all sequence pairs
(with suitable handling of sequence boundaries)
– compute the cross-entropy over all positions $n = 1, ..., N$:

$$F(p) \ := \ 1/N \cdot \sum_{n=1}^{N} \log \, p(c_n | c_{n-k}^{n-1}, x_{n-m}^{n+m})$$

count how often each window/context $[c_{0-k}^{0+1}, x_{0-m}^{0+m}]$
occurs in the super-sequence (abusing notation):

$$pr\big(c_{0-k}^{0}, x_{0-m}^{0+m}\big) \ := 1/N \cdot \sum_{n=1}^{N} \delta\big(c_{0-k}^{0}, c_{n-k}^{n}\big) \cdot \delta\big(x_{0-m}^{0+m}, x_{n-m}^{n+m}\big)$$

$$= \sum_{c_{0-k}^{0}, x_{0-m}^{0+m}} pr\big(c_{0-k}^{0}, x_{0-m}^{0+m}\big) \cdot \log \, p\big(c_0 \big| c_{0-k}^{0-1}, x_{0-m}^{0+m}\big)$$

remarks:
– for this model, all the training data are summarized by the
empirical distribution: $pr\big(c_{0-k}^{0} \, , \, x_{0-m}^{0+m}\big)$
– index 0 is a placeholder for any position n

4.4 Cross-Entropy Training: What does the ANN learn?

questions:
– what does the ANN learn in sequence modelling?
– what about the language model ?

relevance: mismatch conditions for domains/tasks:
different language models in training and testing

c_1^N:	○ ● ● ● ● ○ ○ ○ ○ ○ ● □ ○ ○ . . . .
x_1^N:	● ● ● ● ● ● ● ● ● ● ● □ ● ● ● ● ● ● ● ● ●

we re-write the cross-entropy criterion:

$$F(p) := 1/N \cdot \sum_{n=1}^{N} \log p(c_n | c_{n-k}^{n-1}, x_{n-m}^{n+m}) = \sum_{c_{0-k}^{0}, x_{0-m}^{0+m}} pr(c_{0-k}^{0}, x_{0-m}^{0+m}) \cdot \log p(c_0 | c_{0-k}^{0-1}, x_{0-m}^{0+m})$$

we separate the predicted event c_0:

$$pr(c_{0-k}^{0}, x_{0-m}^{0+m}) = pr(c_{0-k}^{0-1}, x_{0-m}^{0+m}) \cdot pr(c_0 | c_{0-k}^{0-1}, x_{0-m}^{0+m})$$

$$= \sum_{c_{0-k}^{0-1}, x_{0-m}^{0+m}} pr(c_{0-k}^{0-1}, x_{0-m}^{0+m}) \sum_{c_0} pr(c_0 | c_{0-k}^{0-1}, x_{0-m}^{0+m}) \cdot \log p(c_0 | c_{0-k}^{0-1}, x_{0-m}^{0+m})$$

solution (using divergence inequality): = empirical posterior distribution conditioned on the LM history and the input window:

$$\hat{p}(c_0 | c_{0-k}^{0-1}, x_{0-m}^{0+m}) = pr(c_0 | c_{0-k}^{0-1}, x_{0-m}^{0+m})$$

optimum solution for ANN (and other models!):

$$\hat{p}\big(c_0 \big| c_{0-k}^{0-1}, x_{0-m}^{0+m}\big) = pr\big(c_0 \big| c_{0-k}^{0-1}, x_{0-m}^{0+m}\big)$$

to show that this estimate includes the LM prior,
we re-write the empirical posterior distribution and separate the LM prior:

$$pr\big(c_{0-k}^{0-1}, x_{0-m}^{0+m}\big) \equiv pr\big(c_{0-k}^{0-1}\,,\, c_0\,,\, x_{0-m}^{0+m}\big) \qquad \textbf{(separate } c_0\textbf{)}$$

$$pr\big(c_0 \big| c_{0-k}^{0-1}, x_{0-m}^{0+m}\big) = \frac{pr\big(c_{0-k}^{0-1}\,,\, c_0\,,\, x_{0-m}^{0+m}\big)}{\sum_{\tilde{c}_0} pr\big(c_{0-k}^{0-1}\,,\, \tilde{c}_0\,,\, x_{0-m}^{0+m}\big)}$$

$$pr\big(c_{0-k}^{0-1}, x_{0-m}^{0+m}\big) = pr\big(c_{0-k}^{0}\big) \cdot pr\big(x_{0-m}^{0+m} \,\big|\, c_{0-k}^{0}\big)$$

$$= \frac{pr\big(c_0|c_{0-k}^{0-1}\big) \cdot pr\Big(x_{0-m}^{0+m}\big|c_0, c_{0-k}^{0-1}\Big)}{\sum_{\tilde{c}_0} pr\big(\tilde{c}_0|c_{0-k}^{0-1}\big) \cdot pr\Big(x_{0-m}^{0+m}\big|\tilde{c}_0, c_{0-k}^{0-1}\Big)}$$

with the usual decomposition into language model and observation/acoustic model,
(which might not be easy in practice – see later)

analysis of context windows and dependencies:

- **the limited contexts in the emprical probability are caused by the model assumptions of limited contexts**
- **the empirical distribution with limited context is the result of marginalizing (summing out) over the full contexts which might be long:**

$$pr\left(c_{0-k}^{0}, x_{0-m}^{0+m}\right) \quad \text{is obtained by marginalization of}$$

$$pr\left(c_{0-K}^{0}, x_{0-M}^{0+M}\right) = pr\left(c_{0-K}^{0}\right) \cdot pr\left(x_{0-M}^{0+M} \,\middle|\, c_{0-K}^{0}\right)$$

  – **with HUGE windows on input $[-M, +M]$ and output $[-K, 0]$**
  – **again with the usual decomposition into language model and observation/acoustic model**

- **relevance: mismatch conditions for domains/tasks: different language models in training and testing**

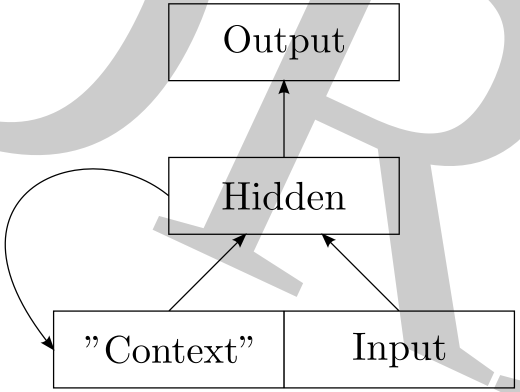
4.5 Sequence Modelling using RNN

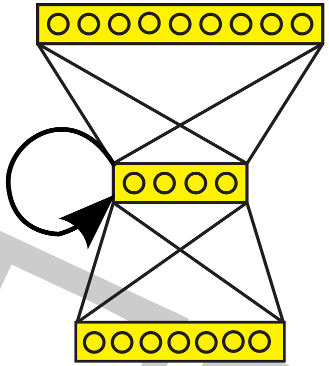
- **assumption:**
  synchronized input and output positions
- **RNN: recurrent neural network**
  maybe with LSTM or other extensions
- **question: what variants of RNNs?**
  and what do they model and learn?

change of notation: position index $t = 1, ..., T$ (rather than n):

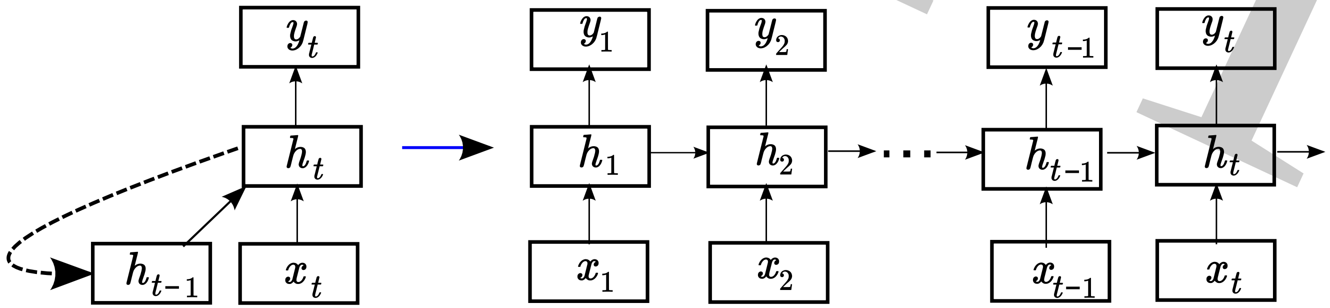
observations x_1^T:	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T
	|	|	|	|	|	|	|	|	|
class labels c_1^T:	c_1	c_2	...	c_{t-1}	c_t	c_{t+1}	...	c_{T-1}	c_T

principle:
– move over string from left to right
– introduce a memory (or context) component to keep track of history
– result: there are two types of input: memory h_{t-1} and observation x_t



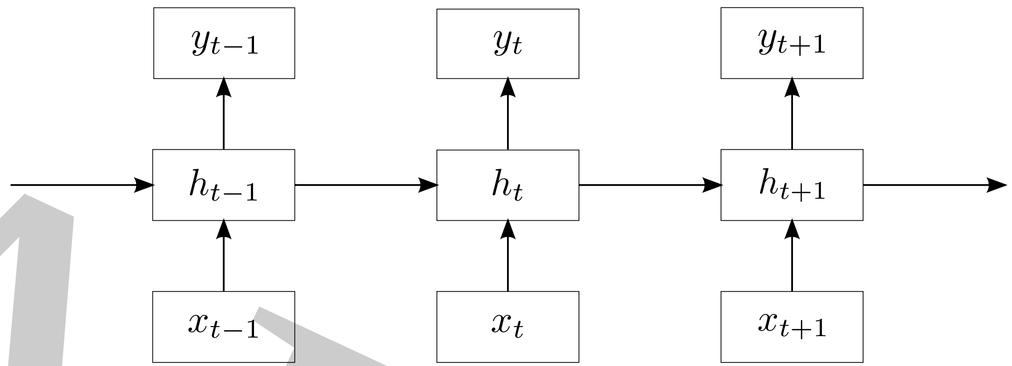


equivalent interpretation: unfolding RNN over time:
feedforward network with a special DEEP architecture.



operations from bottom to top:
(notation for RNN: positions $t = 1, ..., T$):

- **output vector:** $y_t = p_\vartheta(c_t|x_1^t) = S(V h_t)$
  with output matrix V
- **hidden layer:** $h_t = \sigma(U x_t + W h_{t-1})$
  with hidden layer matrix U
  and recurrence matrix W
- **input vector:** x_t
- **RNN output:** $y_t := [y_t(c)] \; = \; [p_t(c|...)]$



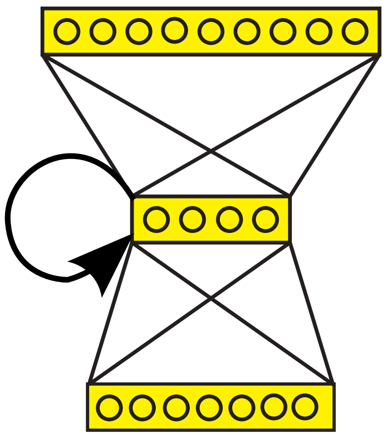
interpretation: conditional probability of RNN:

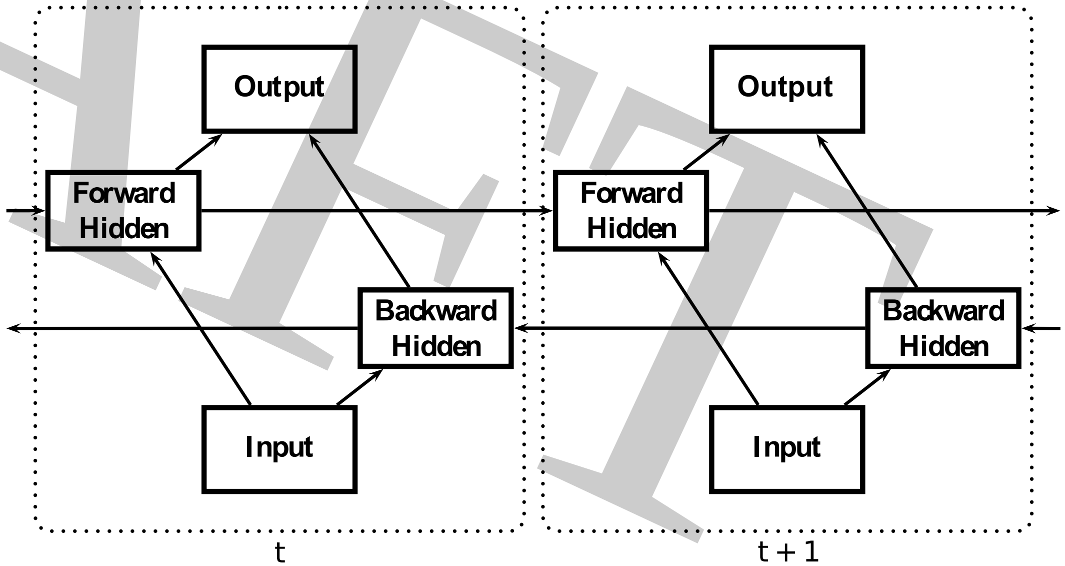
$$
\begin{aligned}
h_t &= h_t(h_{t-1}, x_t) = h_t(x_1^t) \\
p_t(c|h_t) &= p_t(c|x_1^t)
\end{aligned}
$$

with memory states h_t that summarize the partial input x_1^t

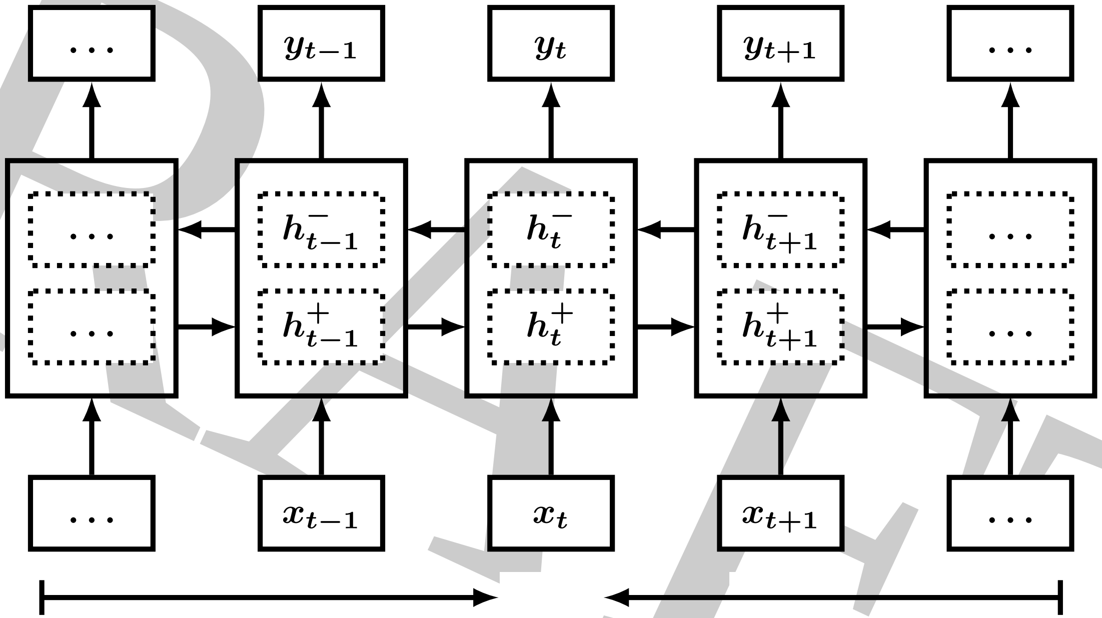
Look-Ahead: Bidirectional RNN
[Schuster & Paliwal 97]

- no look-ahead in x_1^T: forward recurrence
- with look-head in x_1^T: add a backward recurrence
  result: two separate hidden layers



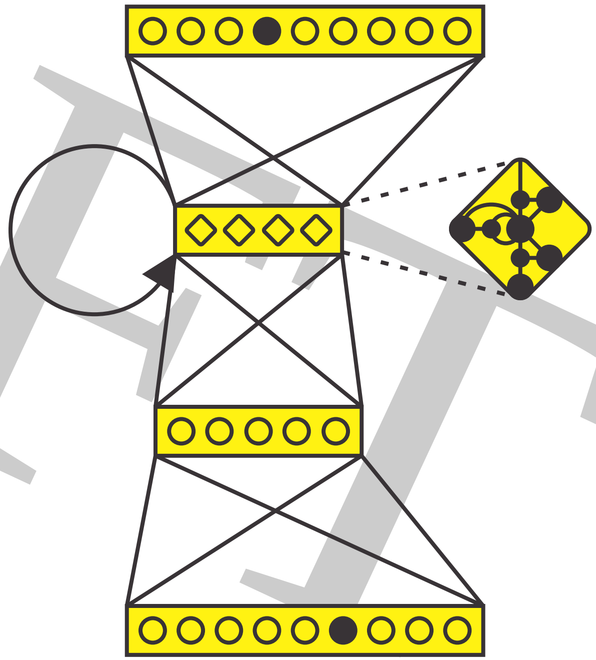


Internal Structure: Separate Forward and Backward Hidden Layers



extension of (simple) RNN by
LSTM: long short-term memory
[Hochreiter & Schmidhuber 97, Gers & Schraudolph^{+} 02]

- **problems of simple RNN:**
  – **vanishing gradients**
  – **no protection of memory h_t**
- **remedy by LSTM architecture:**
  control the access to its internal memory
  by introducing gates/switches
- **additional extension:**
  several hidden layers
- **note: RNN includes LSTM RNN as a special case**

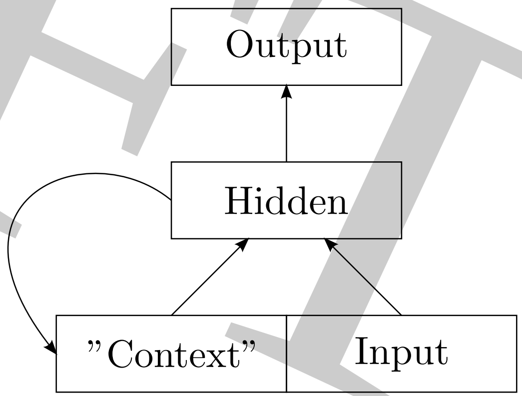


most general factorization:

$$p(c_1^T | x_1^T, x_1^T) = p(c_t \,|\, c_0^{t-1}, x_1^T)$$
$$= p\Big(c_t \,\Big|\, c_{t-1}, h_t(c_0^{t-1}, x_1^T)\Big)$$

with a state vector: $h_t := h_t(c_0^{t-1}, x_1^T)$

many variants beyond baseline structure:
– **input: left to right vs. full input**
– **output: left to right vs. full output put**
– **explicit use of output labels (yes/no): label feedback**



Interpretation of RNN Outputs

two sequences over time $t = 1, ..., T$:

$$\begin{aligned} \textbf{input: sequence of observations:} \quad x_1^T &= x_1 ... x_t ... x_T \\ \textbf{output: sequence of class labels:} \quad c_1^T &= c_1 ... c_t ... c_T \end{aligned}$$

consider the posterior probabilty of the output string:

$$\begin{aligned} \textbf{factorization over time } t\textbf{:} \quad p(c_1^T|x_1^T) &= \prod_{t=1}^{T} p(c_t|c_0^{t-1}, x_1^T) \\ \textbf{marginalization for time } t\textbf{:} \quad p_t(c|x_1^T) &= \sum_{c_1^T : c_t = c} p(c_1^T|x_1^T) \end{aligned}$$

and more variants ...

notation for RNN output vector with nodes = classes $c = 1, ..., C$:

$$y_t = [y_t(c)] = [p_t(c|..., x_1^T)]$$

Factorization of Conditional Probability $p(c_1^T|x_1^T)$

- **conditional independence in c_1^T with look-ahead for x_1^T: $p(c_1^T|x_1^T) = \prod_{t=1}^T p_t(c_t|x_1^T)$**

observations x_1^T:	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T
	|	|	|	|	|	|	|	|	|
class labels c_1^T:	—	—	...	—	$\boxed{c_t}$	—	...	—	—

- **conditional dependence in c_1^T without look-ahead in x_1^T: $p(c_1^T|x_1^T) = \prod_{t=1}^T p(c_t|c_0^{t-1}, x_1^t)$**

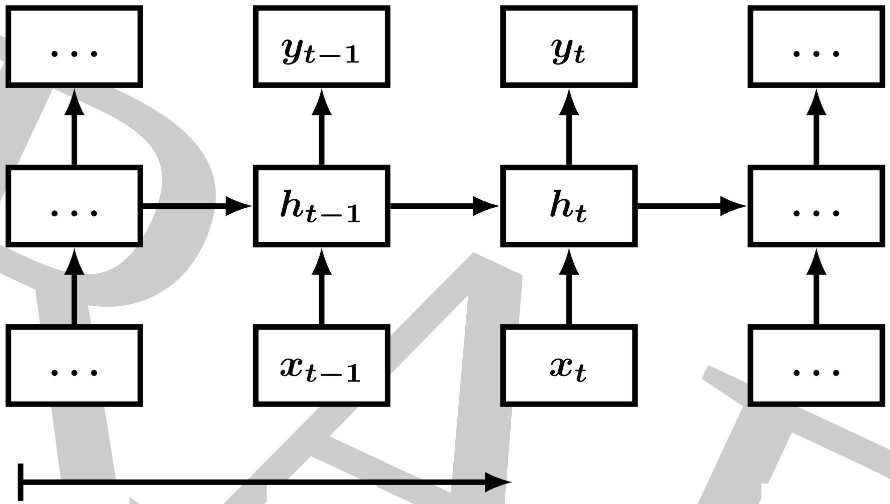
observations x_1^T:	x_1	x_2	...	x_{t-1}	x_t	—	...	—	—
	|	|	|	|	|	|	|	|	|
class labels c_1^T:	c_1	c_2	...	c_{t-1}	$\boxed{c_t}$	—	...	—	—

- **conditional dependence in c_1^T with look-ahead in x_1^T: $p(c_1^T|x_1^T) = \prod_{t=1}^T p(c_t|c_0^{t-1}, x_1^T)$**

observations x_1^T:	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T
	|	|	|	|	|	|	|	|	|
class labels c_1^T:	c_1	c_2	...	c_{t-1}	$\boxed{c_t}$	—	...	—	—

note: this is the most general case.

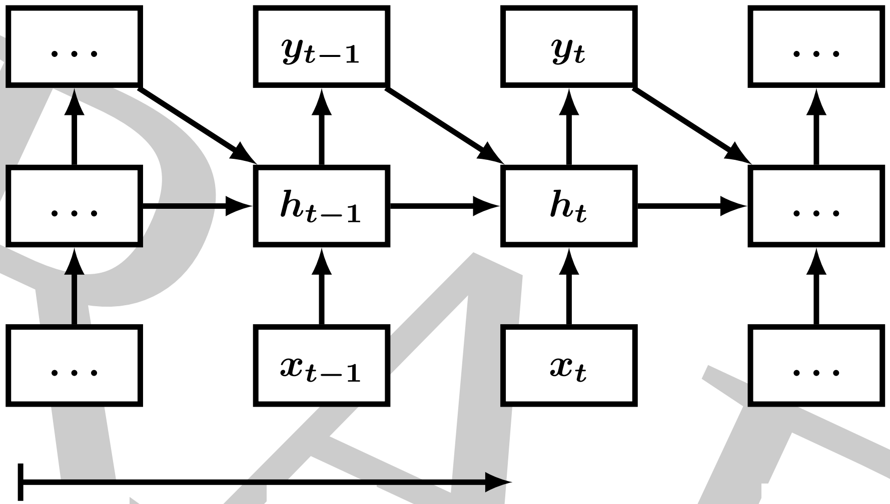
RNN: Variant 1
uni-directional, no feedback of output labels



RNN output: marginal probability in position t with causal input x_1^T

$$\begin{aligned} h_t &= h_t(h_{t-1}, x_t) = h_t(x_1^t) \\ y_t(c) &= p_t(c|h_t) = p_t(c|x_1^t) \end{aligned}$$

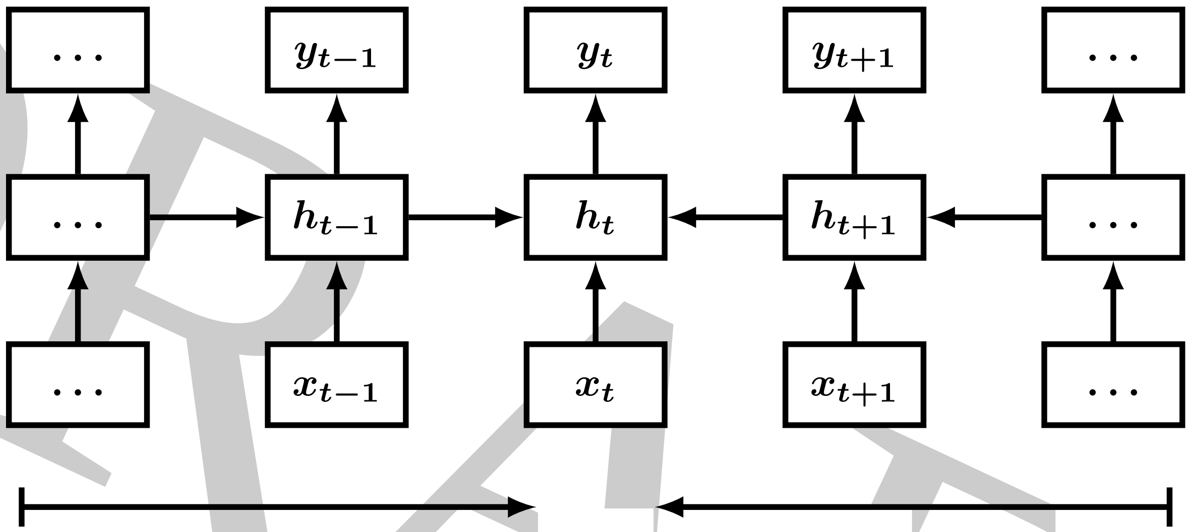
RNN: Variant 2
uni-directional, with feedback of output labels



RNN output: conditional probability in position t with causal input x_1^T

$$\begin{aligned} h_t &= h_t(h_{t-1}, c_{t-1}, x_t) = h_t(c_0^{t-1}, x_1^t) \\ y_t(c) &= p_t(c|h_t) = p_t(c|c_0^{t-1}, x_1^t) \end{aligned}$$

RNN: Variant 3
bi-directional, no feedback of output label

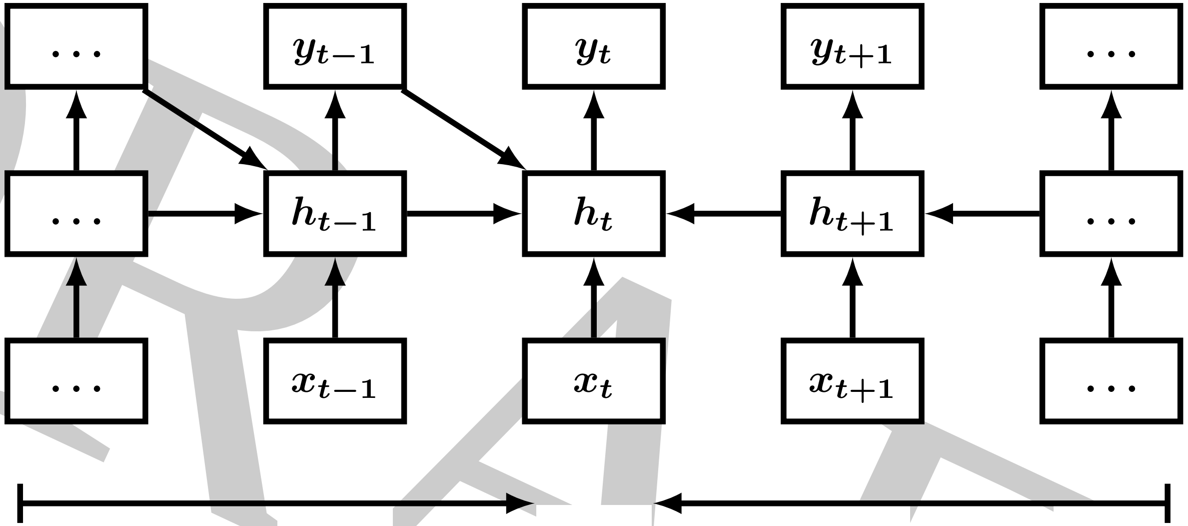


RNN output: marginal probability in position t with full (non-causal) input x_1^T

$$\begin{aligned} h_t &= \ldots = h_t(x_1^T) \\ y_t(c) &= p_t(c|h_t) = p_t(c|x_1^T) \end{aligned}$$

note: if used for factorization,
conditional independence assumption of symbols c_1^T is required.

RNN: Variant 4
bi-directional, with uni-directional feedback of output label

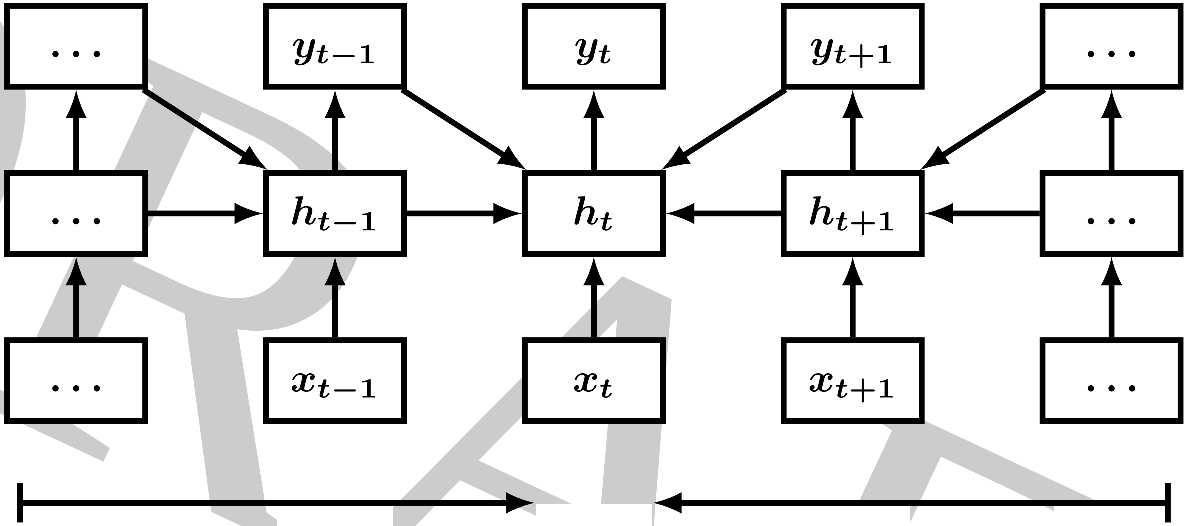


RNN output: conditional probability in position t with non-causal input x_1^T

$$\begin{aligned} h_t &= \ldots = h_t(c_0^{t-1}, x_1^T) \\ y_t(c) &= p_t(c|h_t) = p_t(c|c_0^{t-1}, x_1^T) \end{aligned}$$

note: most general case for factorization!

RNN: Variant 5 (only theoretical?)
bi-directional, with bi-directional feedback of output label



RNN output: "gap" probability in position t with non-causal input x_1^T

$$\begin{aligned} h_t &= \ldots = h_t(c|c_1^T \setminus c_t,\, x_1^T) \\ y_t(c) &= p_t(c|h_t) = p_t(c|c_1^T \setminus c_t, x_1^T) \end{aligned}$$

note: this output cannot be used for factorization!

Overview of RNN Outputs

label feedback	no	uni-direct.	bi-direct.
uni-dir. RNN	$p_t(c\|x_1^t)$	$p_t(c\|c_0^{t-1}, x_1^t)$	——
bi-dir. RNN	$p_t(c\|x_1^T)$	$p_t(c\|c_0^{t-1}, x_1^T)$	$p_t(c\|c_0^{t-1}, c_{t+1}^T, x_1^T)$

experiments: typically $p_t(c|x_1^T)$

how to train the parameters ϑ of an RNN?

- **cross-entropy criterion for a pair (x_1^T, c_1^T):**

$$\begin{aligned} p_\vartheta(c_1^T|x_1^T) &= \prod_t p_\vartheta(c_t|c_0^{t-1}, x_1^T) \\ \log p_\vartheta(c_1^T|x_1^T) &= \sum_t \log\ p_\vartheta(c_t|c_0^{t-1}, x_1^T) \end{aligned}$$

  usual interpretation:
  consider all training sentence pair as a single super-sentence pair

- **conclusion:**
  – baseline form of cross-entropy criterion
  – 'natural' transition from a single string to a sequence of strings

- **cross-entropy: variants different from "baseline form":**
  – sum criterion (as for CTC, transducer, direct HMM)
  – separating the class prior from the class posterior

- general concept:
  - since 1960: general discriminant function for atomic decision/outputs (not designed for strings and ASR)
  - since 1986: specific MLP structure since 1986
- probabilistic interpretation of ANN outputs:
  - general discriminant functions: [Patterson & Womack 66]
  - ANNs for hybrid HMMs in ASR: [Bourlard & Wellekens 89, ?]
- experimental success and deep learning:
  - LeNet for digit image recognition: Le Cun 1989
  - RNN for ASR: Robinson 1994
  - systematic experimental success: only in 2009-2011 (handwriting, speech, image)
  - real improvements over competing methods (like Gaussian, generative HMM, SVM, log-linear models, ...):

    2009 Graves/handwriting; 2011 Hinton/TIMIT;
    2011 Seide et al./hybrid LVCSR, Tuske et al./tandem LVCSR; 2012 computer vision

Alternative Sequence Structure: Self-Attention

– interactions between positions pairs $[t', t]$ within a sequence
– directions: unidirectional (left-to-righ) aand bidirectional
– later: combination with cross-attention: transformer

5 Cross-Entropy Training for Sequences: Extensions

cross-entropy training criterion:
extensions for specific cases

overview:
- sequences (attention, LM)
- sequence discrim. training
- FST/HMM with sum and max

– baseline: input/output pair
– language model
– attention model
– sum structure (HMM and CTC)
– approximation: best path
– sequence discriminative

5.1 Sequences: Posterior and Pseudo Posterior

– alternative title: from generative to direct models
– works for empirical and model distribution

Sequences and Language Model Separation

generative/joint model has a "natural decomposition": $p(x, c) = p(c) \cdot p(x|c)$
into class prior $p(c)$ and class-conditional model $p(x|c)$

considerations on this decomposition:
– advantage: prior $p(c)$ can be separated
relevance: different priors in training and testing
– problem: modelling $p(x|c)$ is hard (as learning problem)
and cannot be modelled by ANNs due to normalization over $x \in \mathbb{R}^D$

remedy: replace the class prior by a uniform class prior and define *pseudo posterior* $\tilde{p}(c|x)$:

$$\tilde{p}(x, c) := 1/C \cdot p(x|c) \qquad \tilde{p}(c|x) := \frac{1/C \cdot p(x|c)}{1/C \cdot \sum_{c'} p(x|c')} = \frac{p(x|c)}{\sum_{c'} p(x|c')}$$

we re-write the original posterior in terms of the pseudo posterior:

$$p(c|x) \; = \; \frac{p(c) \cdot p(x|c)}{\sum_{c'} p(c') \cdot p(x|c')} = \frac{p(c) \cdot \tilde{p}(c|x)}{\sum_{c'} p(c') \cdot \tilde{p}(c'|x)}$$

advantage: structure/models with normalization over classes c,
which can be implemented by softmax in ANNs
remark: re-writing steps are possible
because the posterior is defined as a fraction "numerator/denominator"

Posterior and Pseudo Posterior: Towards Log-Linear Modelling

motivation: we want to balance the two models against each other by introducing exponents $\alpha > 0$ and $\beta > 0$:

$$q(c) := \frac{p^{1/\alpha}(c)}{\sum_{c'} p^{1/\alpha}(c')} \qquad \tilde{q}(c|x) := \frac{\tilde{p}^{1/\beta}(c|x)}{\sum_{c'} \tilde{p}^{1/\beta}(c'|x)}$$

interpretation: the exponents control the 'concentration' of the distributions

we re-write the posterior:

$$p(c|x) \;=\; \frac{p(c) \cdot \tilde{p}(c|x)}{\sum_{c'} p(c') \cdot \tilde{p}(c'|x)} = \frac{q^{\alpha}(c) \cdot \tilde{q}^{\beta}(c|x)}{\sum_{c'} q^{\alpha}(c') \cdot \tilde{q}^{\beta}(c'|x)}$$

important results for ANN structure with softmax output:
– can be derived from generative/joint model
– class prior can be separated
– allows interpretation as re-written generative model
– result: log-linear model (or 'soft fusion')

Posterior and Pseudo Posterior: Distributions for Limiting Cases

limit $\alpha \to 0$:

$$\lim_{\alpha \to 0} \frac{q^\alpha(c)}{\sum_{c'} q^\alpha(c')} = 1/C$$

limit $\alpha \to \infty$:

$$\lim_{\alpha \to \infty} \frac{q^\alpha(c)}{\sum_{c'} q^\alpha(c')} = \delta(c, \operatorname*{argmax}_{c'} q(c'))$$

Pseudo Posterior: Sequences

joint distribution specifically for sequences:

$$p(x_1^N, c_1^N) = p(c_1^N) \cdot p(x_1^N | c_1^N)$$

we re-write the posterior:

$$p(c_1^N | x_1^N) \;=\; \frac{p(c_1^N) \cdot p(x_1^N | c_1^N)}{\sum_{c'^N_1} p(c'^N_1) \cdot p(x_1^N | c'^N_1)} = \frac{p(c_1^N) \cdot \tilde{p}(c_1^N | x_1^N)}{\sum_{c'^N_1} p(c'^N_1) \cdot \tilde{p}(c'^N_1 | x_1^N)}$$

define pseudo posterior $\tilde{p}(c_1^N | x_1^N)$ and factorize it over n

$$= \frac{\prod_n p(c_n | c_0^{n-1}) \cdot \tilde{p}(c_n | c_0^{n-1}, x_1^N)}{\sum_{c'^N_1} \prod_n p(c'_n | c'^{n-1}_0) \cdot \tilde{p}(c'_n | c'^{n-1}_0, x_1^N)}$$

model of limited context: $\tilde{p}_n(c_n | c_{n-1}, x_1^N)$

$$= \frac{\prod_n p(c_n | c_0^{n-1}) \cdot \tilde{p}_n(c_n | c_{n-1}, x_1^N)}{\sum_{c'^N_1} \prod_n p(c'_n | c'^{n-1}_0) \cdot \tilde{p}_n(c'_n | c'_{n-1}, x_1^N)}$$

most important application in ASR:
– separating the language model from the acoustic model
– language model prior can be learned from text only,
i. e. without annotated data! (in ASR: 100+ Mio vs. 10 Mio words)

Pseudo Posterior: Direct Model

vfill direct factoriztion of posterior probability:

$$q(c_1^N|x_1^N) \; = \; \prod_n q(c_n|c_0^{n-1}, x_1^N)$$

re-interpretation:

$$q(c_n|c_0^{n-1}, x_1^N) \; = \; \frac{q(c_n, x_1^N|c_0^{n-1})}{\sum_{\tilde{c}_n} q(\tilde{c}_n, x_1^N|c_0^{n-1})} \; = \; \frac{q(c_n|c_0^{n-1}) \cdot q(x_1^N|c_1^n)}{\sum_{\tilde{c}_n} \; q(\tilde{c}_n|c_0^{n-1}) \cdot q(x_1^N|\tilde{c}_n, c_1^{n-1})}$$

define pseudo posterior: $\tilde{q}(c_1^n|x_1^N) := \dfrac{q(x_1^N|c_1^n)}{\sum_{\tilde{c}_1^n} \; q(x_1^N|\tilde{c}_1^n)}$

$$= \; \frac{q(c_n|c_0^{n-1}) \cdot \tilde{q}(c_1^n|x_1^N)}{\sum_{\tilde{c}_n} \; q(\tilde{c}_n|c_0^{n-1}) \cdot \tilde{q}(c_1^{n-1}, \tilde{c}_n|x_1^N)}$$

factorize: $\tilde{q}(c_1^n|x_1^N) := \prod_{i=1}^n \; \tilde{q}(c_i|c_0^{i-1}, x_1^N)$

and all but one product terms will cancel !

$$= \; \frac{q(c_n|c_0^{n-1}) \cdot \tilde{q}(c_n|c_0^{n-1}, x_1^N)}{\sum_{\tilde{c}_n} \; q(\tilde{c}_n|c_0^{n-1}) \cdot \tilde{q}(\tilde{c}_n|c_0^{n-1}, x_1^N)}$$

model of limited context: $\tilde{q}(c_n|c_{n-1}, x_1^N)$

$$= \; \frac{q(c_n|c_0^{n-1}) \cdot \tilde{q}(c_n|c_{n-1}, x_1^N)}{\sum_{\tilde{c}_n} \; q(\tilde{c}_n|c_0^{n-1}) \cdot \tilde{q}(\tilde{c}_n|c_{n-1}, x_1^N)}$$

5.2 Label Smoothing

Cross-Entropy with Label Smoothing [Pereyra & Tucker$^+$ 17]

extension of weights: (generalized) label smoothing
concept: replace $\delta(c, c_n)$ by a distribution over all classes c:

$$\sum_n \log p(c_n|x_n) \; = \; \sum_n \sum_c \delta(c, c_n) \; \log p(c|x_n) \quad \rightarrow \quad \sum_n \sum_c w(c|c_n) \; \log p(c|x_n)$$

special choice for weights with $q(c) \geq 0, \; \sum_c q(c) = 1$:

$$w(c|c_n) \; = \; (1-\lambda)\,\delta(c, c_n) + \lambda\, q(c) \qquad \sum_c w(c|c_n) = 1$$

$$\begin{aligned}
F\big(\{p(c|x)\}\big) \; &= \; \frac{1}{N} \sum_n \sum_c w(c, c_n) \; \log p(c|x_n) \\
&= \; \frac{1}{N} \Big((1-\lambda) \; \sum_n \log p(c_n|x_n) + \lambda \sum_c q(c) \sum_n \log p(c|x_n) \Big) \\
&= \; (1-\lambda) \sum_{x,c} pr(x,c) \log p(c|x) + \lambda \sum_c q(c) \sum_x pr(x) \; \log p(c|x) \\
&= \; \sum_x pr(x) \sum_c \big[(1-\lambda)\, pr(c|x) + \lambda\, q(c) \big] \; \log p(c|x)
\end{aligned}$$

solution: $\hat{p}(c|x) \; = \; (1-\lambda)\, pr(c|x) + \lambda\, q(c)$

5.3 Training: Remove the Class Prior

- **standard entropy: there is always the class prior**
- **countermeasures:**
  – inverse prior weighting
  – model with explicit prior

ANN model with limited context:

$$p(c_n|x_n, c_{n-1}) \equiv p(c|x, c')$$

standard cross-entropy training has the solution (fully saturated model):

$$\hat{p}(c|x, c') = pr(c|x, c') \equiv \frac{pr(c|c') \cdot pr(x|c, c')}{\sum_{\tilde{c}} pr(\tilde{c}|c') \cdot pr(x|\tilde{c}, c')}$$

which includes the language model prior $pr(c|c')$

re-consider standard training:

$$\{p_{\hat{\theta}}(c|x,c')\} \;=\; \operatorname*{argmax}_{\theta}\left\{\sum_n \log\, p_\theta(c_n|x_n,c_{n-1})\right\} \;=\; \{pr(c|x,c')\}$$

principles: – draft –

– take care of class prior $pr(c_n,c_{n-1})$ in training data

– method: assign weights ($= 1/pr(c_n,c_{n-1})$) to each sample position n

$$\begin{aligned}
F(\theta) \;&=\; \sum_n \frac{1}{pr(c_n,c_{n-1})}\cdot \log\, p_\theta(c_n|x_n,c_{n-1})\\
&=\; \sum_{x,c,c'} \frac{pr(x,c,c')}{pr(c,c')}\cdot \log\, p_\theta(c|x,c') = \sum_{x,c,c'} pr(x|c,c')\cdot \log\, p_\theta(c|x,c')
\end{aligned}$$

define pseudo posterior: $\tilde{pr}(c|x,c') := \dfrac{pr(x|c,c')}{\sum_{c''} pr(x|c'',c')} = \dfrac{pr(x|c,c')}{Q(x,c')}$

$$=\; \sum_{x,c'} Q(x,c')\; \sum_c \tilde{pr}(c|x,c')\cdot \log\, p_\theta(c|x,c')$$

solution for fully saturated model: pseudo posterior:

$$\hat{p}_\theta(c|x,c') = \tilde{pr}(c|x,c') = \frac{pr(x|c,c')}{\sum_{c''} pr(x|c'',c')}$$

introduce specific structure into observation model:

$$\hat{p}_\theta(c|x,c') \;=\; \frac{pr(c|c') \cdot q_\theta(c|x,c')}{\sum_{\tilde{c}}\; pr(\tilde{c}|c') \cdot q_\theta(\tilde{c}|x,c')}$$

find the solution for $q_\theta(c|x)$ by requiring: – draft –

$$\hat{p}_\theta(c|x,c') \;\stackrel{!}{=}\; pr(c|x,c')$$

$$\frac{pr(c|c') \cdot q_\theta(c|x,c')}{\sum\limits_{c"} pr(c"|c') \cdot q_\theta(c"|x,c')} \;\stackrel{!}{=}\; \frac{pr(c|c') \cdot pr(x|c,c')}{\sum\limits_{c"} pr(c"|c') \cdot pr(x|c",c')} \;=\; \frac{pr(c|c') \cdot \dfrac{pr(x|c,c')}{Q(x,c')}}{\sum\limits_{c"} pr(c"|c') \cdot \dfrac{pr(x|c",c')}{Q(x,c')}}$$

for any function $Q(x,c')$

by choosing $Q(x,c') := \sum_{\tilde{c}} pr(x|\tilde{c},c')$
we have the solution (like pseudo posterior):

$$\hat{q}_\theta(c|x,c') = \frac{pr(x|c,c')}{\sum_{\tilde{c}} pr(x|\tilde{c},c')}$$

two approaches to separating the prior:

- analytic approach: use standard ANN structure with single softmax:

$$\hat{p}_\theta(c|x)$$

  – train the model $\hat{p}_\theta(c|x)$ (without explicit priors $pr(c)$)
  – analytically correct the bias α_c of the softmax by the prior $pr(c)$

- numerical approach: define ANN structure with double softmax:

$$\hat{p}_\theta(c|x) = \frac{pr(c) \cdot q_\theta(c|x)}{\sum_{c'} pr(c') \cdot q_\theta(c'|x)}$$

  – define ANN structure with two re-normalizations
  – train the model $q_\theta(c|x)$ by backpropagataion

5.4 Training Criterion: MMI - Contrastive Loss

unsupervised training: no labels/annotation

model $p_\vartheta(c|x)$ with training data $(x_n, c_n), n = 1, ..., N$
(spirit of contrastive loss: re-normalization over all training data):

$$
\begin{aligned}
\vartheta \to F(\vartheta) &= \frac{1}{N} \cdot \sum_n \log \frac{p_\vartheta(c_n|x_n)}{1/N \cdot \sum_{\tilde{n}=1}^{N} p_\vartheta(c_n|x_{\tilde{n}})} \\
&= \sum_{x,c} pr(x,c) \log \frac{p_\vartheta(c|x)}{\sum_{\tilde{x}} pr(\tilde{x}) \cdot p_\vartheta(c|\tilde{x})} \\
&= \sum_{x,c} pr(x,c) \log \underbrace{\frac{pr(x) \cdot p_\vartheta(c|x)}{\sum_{\tilde{x}} pr(\tilde{x}) \cdot p_\vartheta(c|\tilde{x})}}_{:= q_\vartheta(c|x)} - \sum_x pr(x) \log pr(x)
\end{aligned}
$$

$$
\begin{aligned}
\operatorname*{argmax}_\vartheta \{F(\vartheta)\} &= \operatorname*{argmax}_\vartheta \Big\{ \sum_{x,c} pr(x,c) \log\, q_\vartheta(c|x) \Big\} \\
&= \operatorname*{argmax}_\vartheta \Big\{ \sum_x pr(x) \sum_c pr(c|x) \log\, q_\vartheta(c|x) \Big\} \\
\hat{q}_{\hat{\vartheta}}(c|x) &= pr(c|x) \qquad \text{and} \qquad \hat{p}_{\hat{\vartheta}}(c|x) = pr(c|x)
\end{aligned}
$$

result: not interesting and disappointing

Training Criterion: MMI - Contrastive Loss (obsolete?)
MMI: Maximum Mutual Information (DRAFT - to be worked out)

consider definition of mutual information (dropping the $\log$):

$$\frac{p(x,c)}{p(x)\cdot p(x)} = \frac{p(c|x)}{p(c)} = \frac{p(x|c)}{p(x)}$$

consider the MMI training criterion with training data $(x_n, c_n), n = 1, ..., N$ (generalized contrastive loss: re-normalization over all training data):

$$\vartheta \to F(\vartheta) = \frac{1}{N}\cdot\sum_n \log \frac{p_\vartheta(c_n|x_n)}{1/N\cdot\sum_{\tilde{n}=1}^{N} p_\vartheta(c_n|x_{\tilde{n}})} = ... = \frac{1}{N}\cdot\sum_n \log \frac{p_\vartheta(c_n|x_n)}{p_\vartheta(c_n)}$$

$$= \sum_{x,c} pr(x,c)\log\frac{p_\vartheta(c|x)}{p_\vartheta(c)} = \sum_{x,c} pr(x,c)\log\frac{p_\vartheta(x|c)}{pr(x)}$$

re-writing: using two basic components, we define a *mixed* joint probability:

$pr(x)$ **and** $p_\vartheta(c|x)$ **:** $\quad p_\vartheta(x,c) = pr(x)\cdot p_\vartheta(c|x)$

and the associated (normalized!) probabilities:

$$p_\vartheta(c) = \sum_x p_\vartheta(x,c) = 1/N\cdot\sum_n p_\vartheta(c|x_n) \qquad p_\vartheta(x|c) = \frac{p_\vartheta(x,c)}{p_\vartheta(c)}$$

we have shown:

$$\vartheta \to F(\vartheta) = \frac{1}{N} \cdot \sum_n \log \frac{p_\vartheta(c_n|x_n)}{1/N \cdot \sum_{\tilde{n}=1}^{N} p_\vartheta(c_n|x_{\tilde{n}})} = \sum_{x,c} pr(x,c) \log \frac{p_\vartheta(c|x)}{p_\vartheta(c)}$$

$$= \sum_{x,c} pr(x,c) \log \frac{p_\vartheta(x|c)}{pr(x)} = const(\vartheta) + \sum_{x,c} pr(x,c) \log p_\vartheta(x|c)$$

important result for this MMI criterion:

the class-conditional model $p_\vartheta(x|c)$ is being optimized, without using the model directly!

comparison with Povey's method:

- **Povey computes the prior $p_\vartheta(c)$ afterwards, not as part of the optimization procedure.**
- **no difference for ANN models with full degrees of freedom.**

6 FSM-HMM for Synchronization

overview:

- sequences (attention, LM)

- sequence discrim. training

- FST/HMM with sum and max

Sequences: General Model Structures

outline:

– isolated events (no context): discriminative vs. generative modelling
softmax = posterior of a generative model: prior + Gaussian model
baseline DNN = Gaussian posterior + feature extraction

following subsections:
– model structures for seq-to-seq processing:
generative model, log-linear model (CRF), direct model
– first-order models for sequence synchronization:
HMM, CTC, transducer
– training criteria revisited
– attention mechanism for synchronization

From Single Events to Strings: Model Structures

- **string processing: pair of [input,output] strings (or sequences):**

$$[x_1^T, c_1^N] \qquad x_1^T := x_1 ... x_t ... x_T \qquad c_1^N = c_1 ... c_n ... c_N$$

- **starting point for strings: posterior probability (sloppy notation)**

$$p_\vartheta(N, c_1^N | x_1^T) \;\equiv\; p_\vartheta(c_1^N | x_1^T) = \prod_n p(c_n | c_0^{n-1}, x_1^T)$$

with sentence end symbol S: $\quad c_N = S \rightarrow p(c_n | c_0^{n-1}, x_1^T) = 0 \quad n \geq N$

with unknown synchronization and unknown length N of output sequnece c_1^N

- **we are facing three problems in string modelling:**
  - **structural assumptions for string dependencies**
  - **normalization requirement**
  - **synchronization mechanism between input and output string**

- **concepts for normalization:**
  - **factorization followed by local re-normalization**
  - **global re-normalization followed by a factorization**

unifying view:
HMMs, hybrid HMMs, CTC, (RNN-)Transducer

general concepts:
- **formal grammars: regular**
- **finite-state machine with transitions**
- **add output to input**
- **add probabilisties/weights**

area speech & language:
- **HMMs,**
- **combination with ANNs:**
  Bourlard 89, P. Haffner 93, T. Robinson 94, CTC 2006, RNN-T 2011
- **transducer (Vidal)**

related issues: efficieny of operations and learning

6.1 FSM-HMM Formalism

here: only ASR, not MT (non-monotonicity, see Wang ACI 2018)

First-Order Models for ASR sequence synchronization:
such as hybrid HMM, CTC, (RNN-)Transducer, ...
– common mathematical framework: first-order dependences
– differences in details: labels, transitions, etc.

starting point: log-linear model at sequence level:

$$p_\vartheta(c_1^N|x_1^T) := \frac{q_\vartheta^\alpha(c_1^N) \cdot q_\vartheta^\beta(c_1^N|x_1^T)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} q_\vartheta^\alpha(\tilde{c}_1^{\tilde{N}}) \cdot q_\vartheta^\beta(\tilde{c}_1^{\tilde{N}}|x_1^T)}$$

notation for ASR: words consist of smaller units:
– sequence of segments/phonemes/letters: $a_1^S = a_1...a_s...a_S$
– whole word sequence: $W = a_1^S$

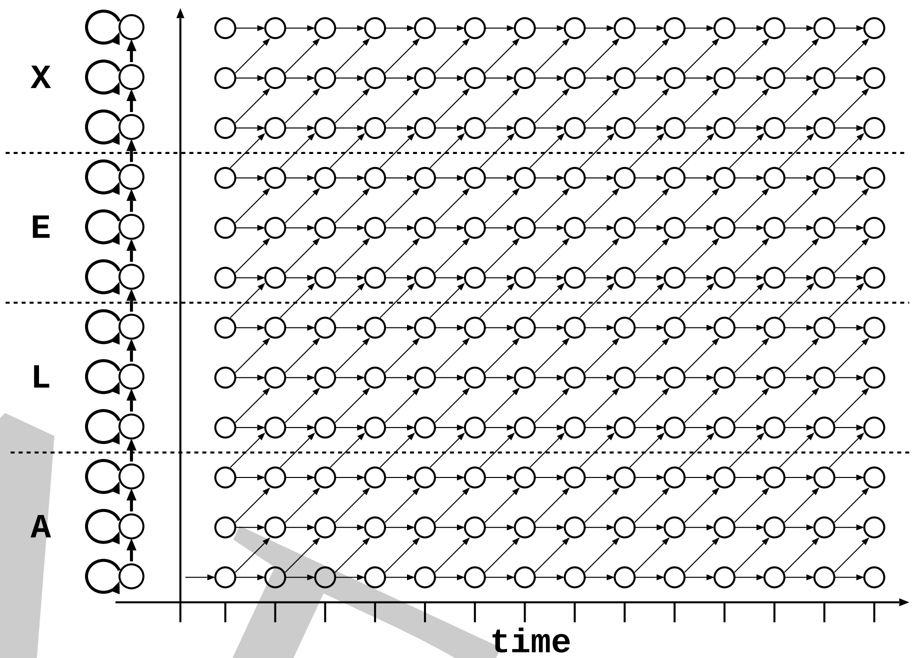
$$p_\vartheta(W|x_1^T) := \frac{q_\vartheta^\alpha(W) \cdot q_\vartheta^\beta(W = a_1^S|x_1^T)}{\sum_{\tilde{W}} q_\vartheta^\alpha(W) \cdot q_\vartheta^\beta(\tilde{W} = \tilde{a}_1^S|x_1^T)}$$

– next step: explicit structure for $q_\vartheta^\beta(W = a_1^S|x_1^T)$

First-Order Models: Hidden Markov Model (HMM)

– sequence of acoustic vectors:
$X = x_1^T = x_1 ... x_t ... x_T$ over time t

– sequence of states $s = 1, ..., S$
$s_1^T = s_1 ... s_t ... s_T$ over time t
with associated state labels:
$a_1^S = a_1 ... a_s ... a_S$
$= W$: word sequence



objective of HMM: time alignment
= synchronization between input and output

- classical HMM: generative model for x_1^T:

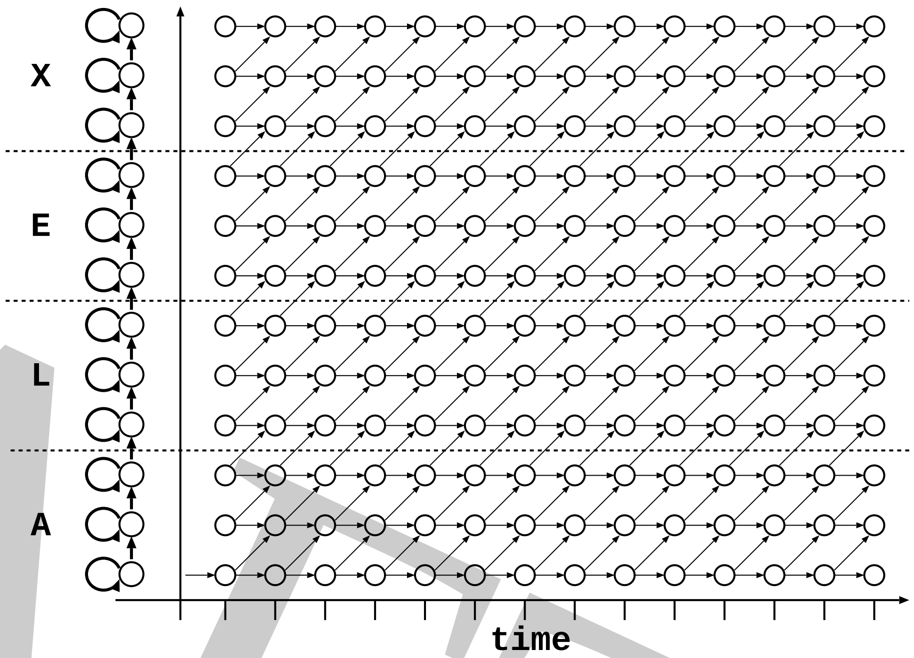
$$q_\vartheta(x_1^T | W = a_1^S) = \sum_{s_1^T} \prod_t q(s_t | s_{t-1}, W, \vartheta) \cdot q_t(x_t | a_{s=s_t}, \vartheta)$$

- hybrid HMM: model of label posterior sequence a_1^S:

$$q_\vartheta(W = a_1^S | x_1^T) = \sum_{s_1^T} \prod_t q(s_t | s_{t-1}, W, \vartheta) \cdot q_t(a_{s=s_t} | x_1^T, \vartheta)$$

machine learning point-of-view: it is easier to model $q_t(a_s | x_1^T, \vartheta)$ than $q_t(x_t | a_s, \vartheta)$
HMM: [Bourlard & Wellekens 89], CTC: [Graves & Fernandez$^+$ 06], RNN-T: [Graves 12]

Label Posterior Probability



key quantity: frame label posterior at time t over labels $a = a_s$ for state/segment s:

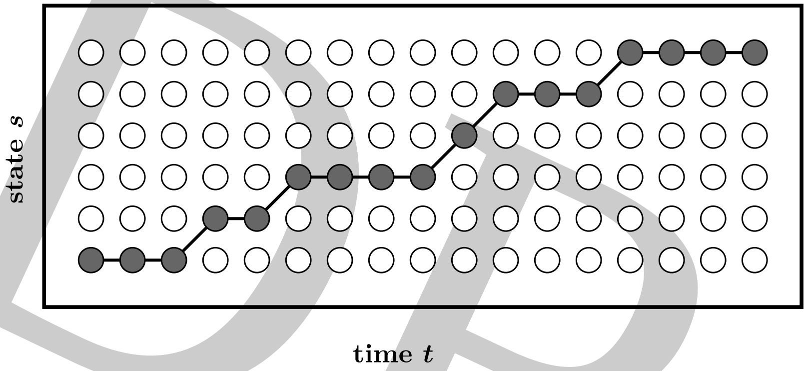
$$q_t(a|x_1^T) \equiv q_t(a|x_1^T, \vartheta)$$

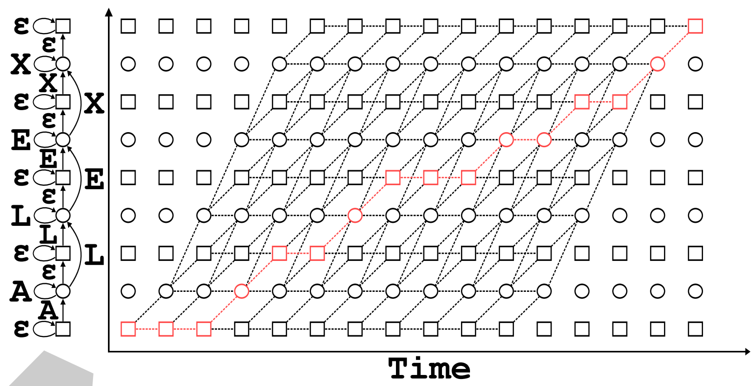
ANN modelling:

- **MLP with window around t:** $q_t(a|x_{t-\delta}^{t+\delta})$
- **bi-direct. (LSTM) RNN: full context x_1^T**
- **transformer/conformer structures: full context x_1^T**

illustration:

acoustic vectors:	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T
	|	|	|	|	|	|	|	|	|
label posteriors:	—	—	...	—	$\boxed{q_t(a\|x_1^T)}$	—	...	—	—





HMM with no skips: segmentation

- **output labels: segments**
- **introduce a special symbol:**
  ϵ: **empty symbol, garbage or blank symbol, don't care symbol**
  also used for silence portions
- **choice (within sequence constraints given by a_1^S):**
  frame labels for each segment s with label a_s:
  – one or more label a_s
  – and the rest will be ϵ

From HMM to CTC

- **result:**
  – structural constraints on transitions
  – no transition probabilities
- **posterior distribution over frame labels $y_t = a_s$:**

$$q_\theta(y_t = a_s | x_1^T) \qquad \text{with} \quad \sum_{y_t \in \{a\} \cup \epsilon} q_\theta(y_t | x_1^T) = 1$$

interpretation (?): due to special symbol ϵ, transition probabilities are not needed

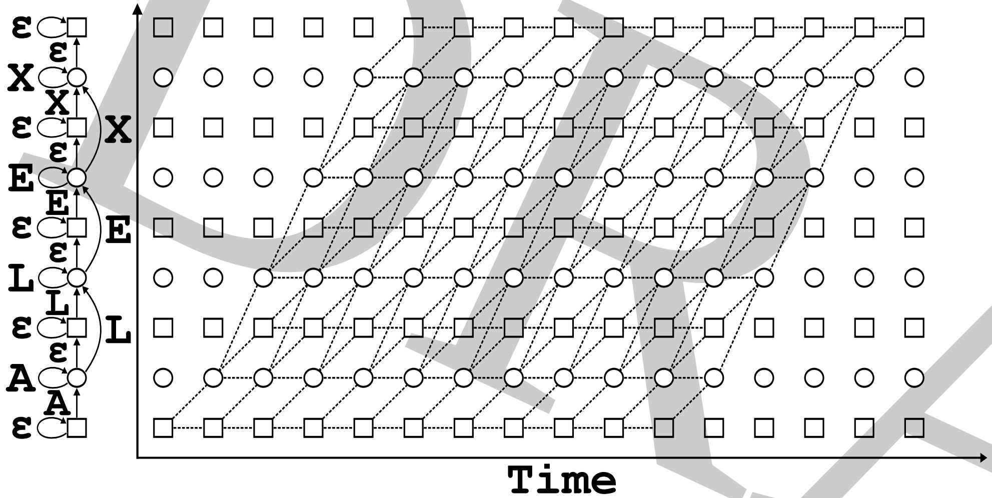
- **CTC model:**

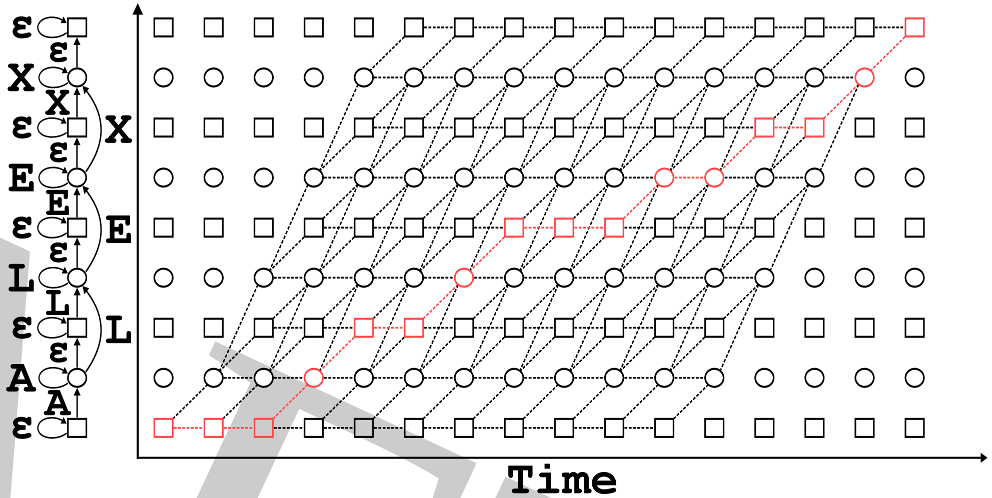
$$q_\vartheta(W = a_1^S | x_1^T) = \sum_{s_1^T} \prod_t q_\vartheta(y_t = a_{s_t} | x_1^T)$$

and structural constraints on transitions $s_{t-1} \rightarrow s_t$

- **compare with general hybrid HMM:**

$$q_\vartheta(W = a_1^S | x_1^T) = \sum_{s_1^T} \prod_t q_\vartheta(s_t | s_{t-1}, W) \cdot q_\vartheta(y_t = a_{s_t} | x_1^T)$$





analysis (mathematical and experimental):

the optimum solution will have the tendency to produce a maximum number of ϵ

from segment labels to frame labels:

- **sequence a_1^S of segment labels for $S = 3$:**

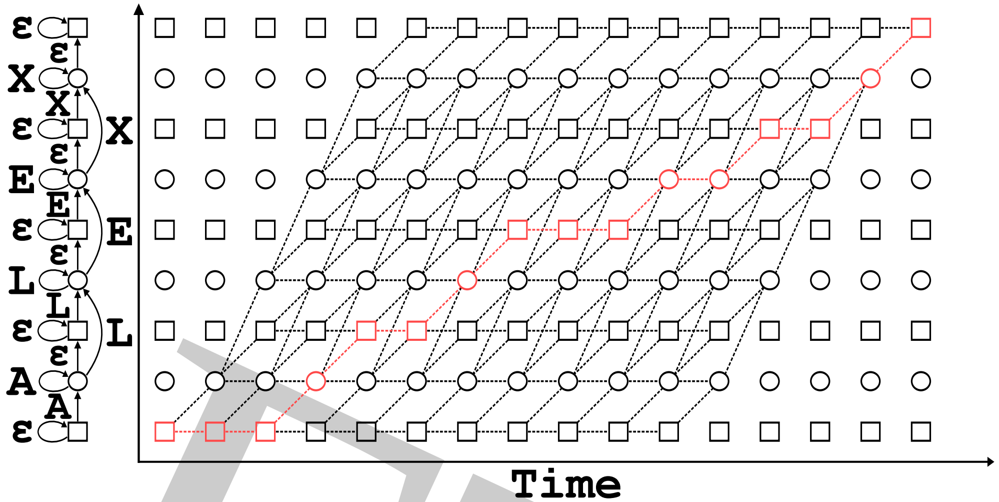
$$A \; B \; C$$

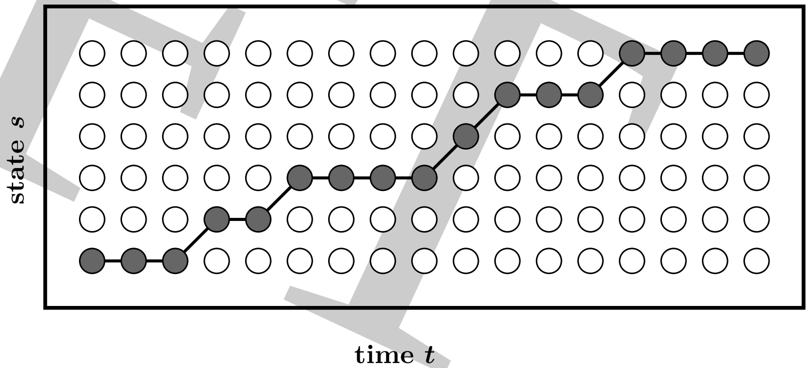
operation: repeat a symbol or insert ϵ

- **some possible sequences $y_1^T \in [a_1^S]$ of frame labels for $T = 9$:**

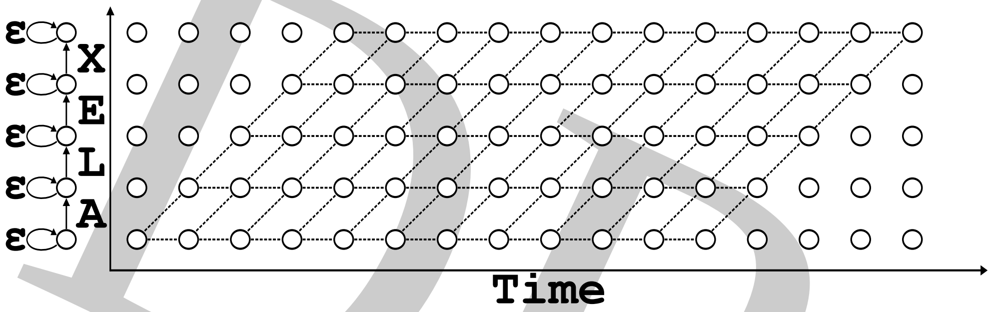
$$\begin{array}{ccccccccc}
A & A & A & B & B & B & B & C & C \\
A & \epsilon & \epsilon & B & \epsilon & \epsilon & \epsilon & C & \epsilon \\
\epsilon & \epsilon & A & \epsilon & B & C & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & A & \epsilon & B & \epsilon & \epsilon & C & \epsilon \\
\epsilon & \epsilon & A & \epsilon & B & B & \epsilon & C & \epsilon
\end{array}$$

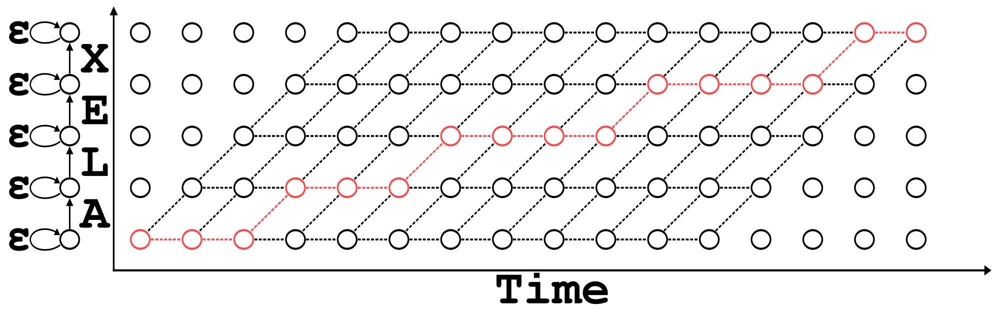
- **analysis (mathematical and experimental):**
  the optimum solution will have the tendency to produce a maximum number of ϵ





RNN-T: RNN Transducer





from CTC to RNN-T: define model:

– each true symbols (i. e. other than ϵ) occurs exactly once:
  assigned to forward transitions only

– dependence on output context:

  full context: $q_\vartheta(y_t = a_s | a_0^{s-1}, x_1^T)$

  limited context: k predecessor symbols: $q_\vartheta(y_t = a_s | a_{s-k-1}^{s-1}, x_1^T)$

  typical assumption here: context = 1 or 0

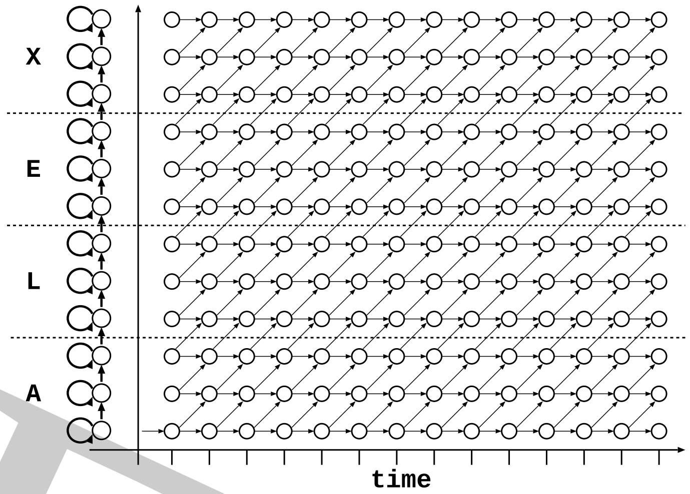
terminology:

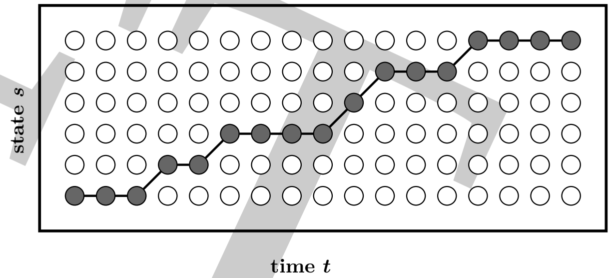
– general case: RNN-T: could have vertical transitions [Graves 12]

– RNN Aligner (RNN-A): no vertical transitions [Sak & Shannon^{+} 17]

– transducer: without RNN (on output side)

Revisit the Mathematical Problem: From Frame Labels to Segment Labels





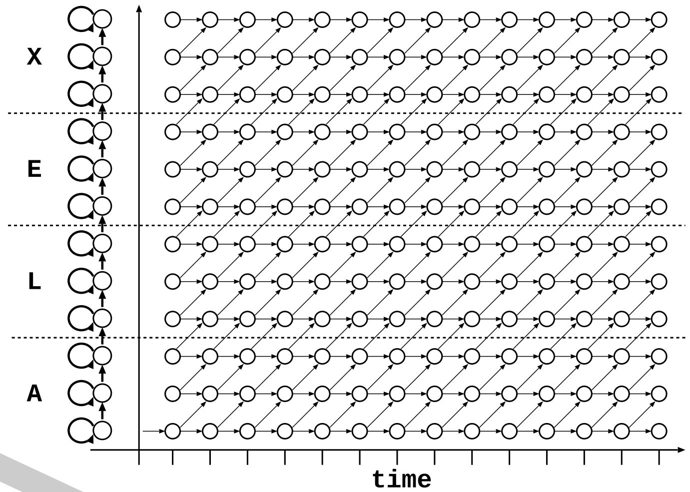
- consider sequence of frame labels over $t = 1, ..., T$:
  (allophonic/CART labels, phonemes, letters,...):
  $$y_1^T = y_1 ... y_t ... y_T$$
  specific condition in ASR:
  phoneme = 90 msec = 9 · 10-msec frames

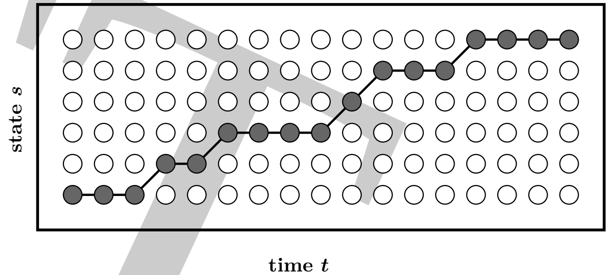
- goal: convert frame sequence to segment sequence:
  $$y_1^T = aaabbcccc \;\rightarrow\; a_1^S = a_1^S(y_1^T) = ?$$
  definition: segment := repeated frame labels

- example: sequence of frame labels:
  $$y_1^T = aaabbcccc \;\rightarrow a_1^S = ?$$
  possible associated segment sequences a_1^S:
  $$abc \quad aabc \quad abcc$$

- conclusions:
  – a posterior model $q(y_1^T|x_1^T)$ alone is not sufficient for specifying the problem
  – in addition: we need some segmentation information
  – formalism: segmentation path s_1^T with $t \rightarrow s = s_t$:
  $$(s_1^T, y_1^T) \;\rightarrow\; a_1^S \quad \text{with: } a_{s_t} = y_t$$

From Frame Labels to Segment Labels: Posterior HMM – Formalism





- **three levels of sequences:**

$$x_1^T \rightarrow (s_1^T, y_1^T) \rightarrow a_1^S$$

- **segment sequence posterior probability: include path s_1^T as a hidden variable:**

$$q(a_1^S|x_1^T) \;:=\; \sum_{s_1^T} q\big(s_1^T, [y_t = a_{s_t}]_{t=1}^T \big| x_1^T\big)$$
$$= \sum_{s_1^T} q\big(s_1^T, a_1^S \big| x_1^T\big)$$

formalism: HMM or FSM (finite-state machine)

- **general concept of first-order finite FSM/HMM: assume first-order dependence on s_1^T and factorize over frame times t and segment positions s:**

$$q(s_1^T, a_1^S|x_1^T) \;=\; \prod_t q\big(s_t, y_t = a_{s_t} \big| s_{t-1}, ..., x_1^T\big) = \prod_s \prod_{t:s=s_t} q\big(s_t, y_t = a_s \big| s_{t-1}, ..., x_1^T\big)$$

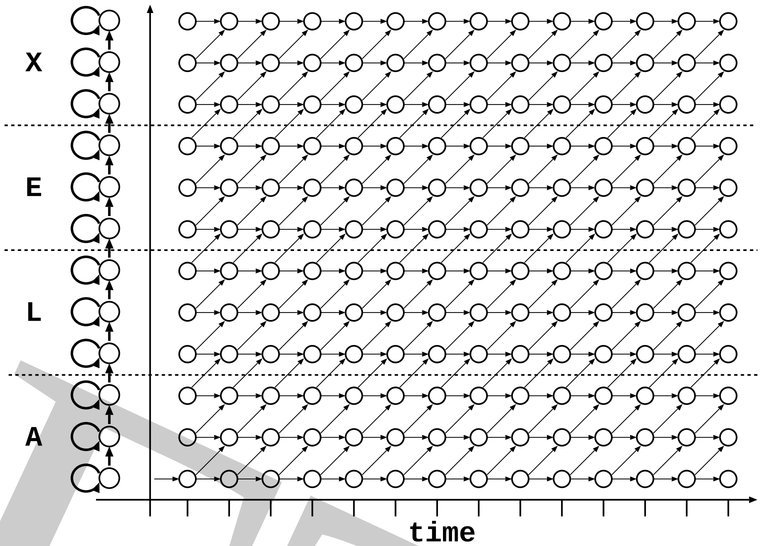
Posterior HMM (without ϵ)

consider segment/state s with label a_s:

- **basic step: leaving $[t, s]$:**
  transition $[t, s = s_t] \;\rightarrow\; [t+1, s_{t+1} = s_t + \delta_t]$:
  – first: generate label $y_t = a_s$
  – then: decide $\delta_t = 0/1$: *stay in / leave segment s?*

- **all frames but last: $q(y_t = a_s, \delta_t = 0|x_1^T)$**
  with $\delta_t = 0$: stay in segment

- **last frame: $q(y_t = a_s, \delta_t = 1|x_1^T)$**
  with $\delta_t = 1$: leave segment

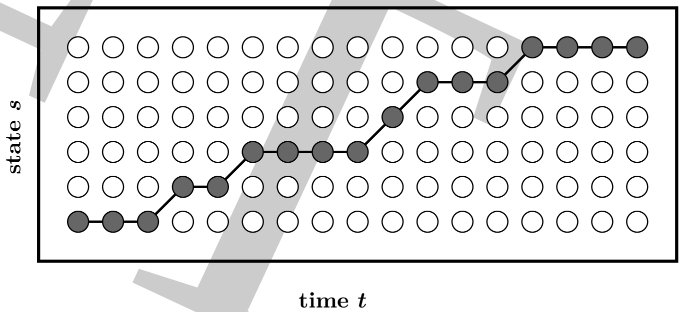


re-write posterior HMM probability:

$$q(a_1^S|x_1^T) \;=\; \sum_{s_1^T} q(a_1^S, s_1^T|x_1^T) = \sum_{s_1^T} \prod_t q(y_t = a_{s_t}, s_{t+1}|s_t, x_1^T)$$

define transition variable: $\quad \delta_t := s_{t+1} - s_t$

$$= \sum_{s_1^T} \prod_t q(y_t = a_{s_t}, \delta_t|x_1^T) = \sum_{s_1^T} \prod_s \prod_{t:s=s_t} q(y_t = a_s, \delta_t|x_1^T)$$



specific assumptions:

$$\begin{aligned} q(y_t = a_s, \delta|x_1^T) \;&\equiv\; q(y_t = a_s|x_1^T) \cdot q(\delta_t|a_s, x_1^T) \\ &=\; \underbrace{q(y_t = a_s|x_1^T)}_{\text{label prob.}} \cdot \underbrace{q(\delta_t|a_s)}_{\text{trans. prob.}} \end{aligned}$$

related papers by RWTH: [Raissi & Beck$^+$ 20/21/22 arxiv]
[Zhou & Berger$^+$ 2021], [Zhou & Zeyer$^+$ 2021]

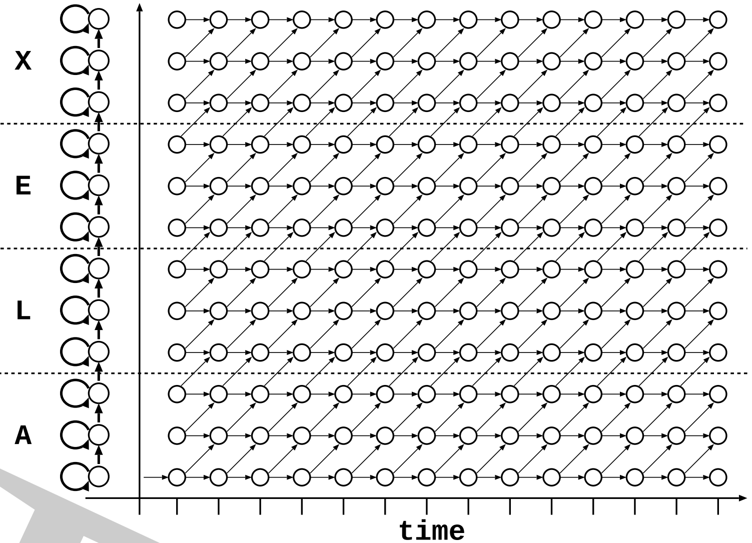
Transducer: Posterior HMM with ϵ

consider segment/state s with label a_s:

- **basic step: arriving in $[t, s]$:**
  transition $[t-1, s_{t-1}] \;\rightarrow\; [t, s = s_t = s_{t-1} + \delta_t]$:
  – first: decide $\delta_t = 0/1$: *start new segment: yes/no?*
  – then: generate label $y_t = a_s$

- **first frame: $q(\delta_t = 1, y_t = a_s | x_1^T)$**
  with $\delta_t = 1$: start new segment

- **all frames but first: $q(\delta_t = 0, y_t = a_s | x_1^T)$**
  with $\delta_t = 0$: stay in segment

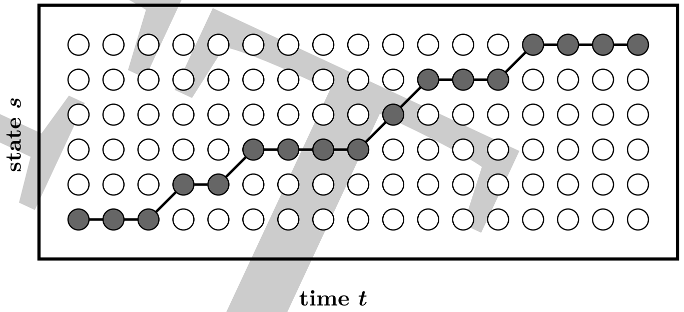


re-write posterior HMM probability:

$$q(a_1^S | x_1^T) \;=\; \sum_{s_1^T} \prod_t q(s_1^T, a_1^S | x_1^T) = \sum_{s_1^T} \prod_t q(s_t, y_t = a_{s_t} | s_{t-1}, x_1^T)$$

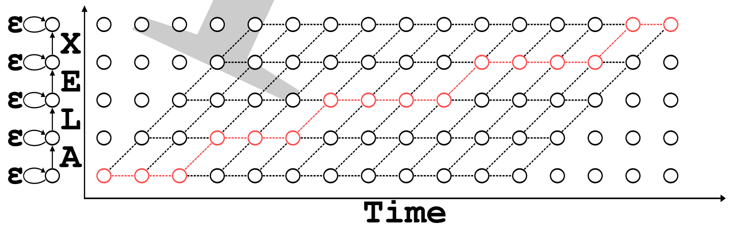
define transition variable: $\quad \delta_t := s_t - s_{t-1}$

$$= \sum_{s_1^T} \prod_t q(\delta_t, y_t = a_{s_t} | x_1^T) = \sum_{s_1^T} \prod_{s=1}^{S} \prod_{t: s = s_t} q(\delta_t, y_t = a_s | x_1^T)$$



for JOINT event $(\delta_t, y_t = a_s)$, define NEW set of labels:
– transition $\delta_t = 1$: use true symbol a_s
– transition $\delta_t = 0$: use blank symbol ϵ

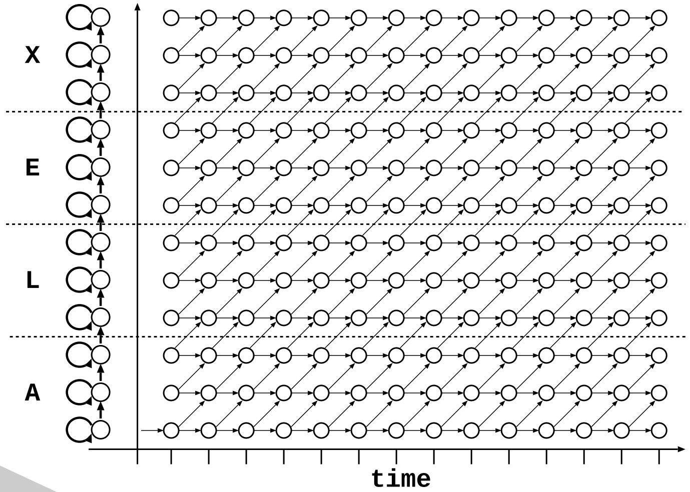
with normalization: $\displaystyle \sum_{y_t \in \{a_s\} \cup \epsilon} q(y_t | x_1^T) = 1$



Posterior HMM: From Hybrid HMM to CTC to RNN-T

direct re-writing of posterior HMM probability:

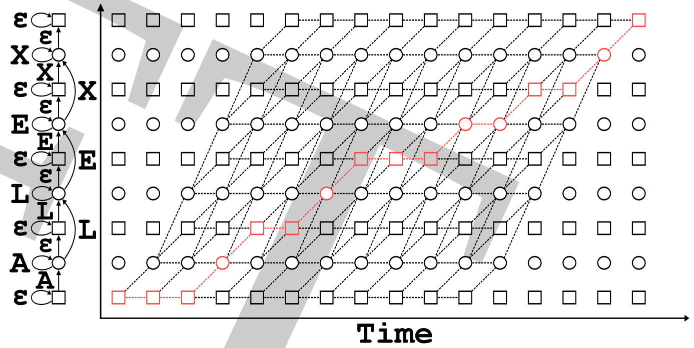
$$\begin{aligned} q_\vartheta(a_1^S|x_1^T) &= \sum_{s_1^T} q_\vartheta(s_1^T, a_1^S|x_1^T) \\ &= \sum_{s_1^T} \prod_t q_\vartheta(s_{t+1}, y_t = a_{s_t}|s_t, a_{s_t-1}, x_1^T) \\ &= \sum_{s_1^T} \prod_t q_\vartheta(s_{t+1}|s_t, a_{s_t}) \cdot q_\vartheta(y_t = a_{s_t}|a_{s_t-1}, x_1^T) \end{aligned}$$



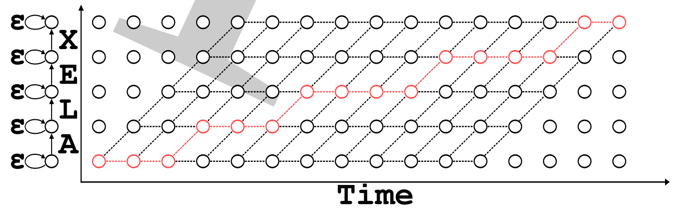
papers by RWTH: [Raissi & Beck+ 20/21/22 arxiv]
[Zhou & Berger+ 2021], [Zhou & Zeyer+ 2021]

posterior HMM with ϵ symbol: CTC and transducer (RNN-T/RNN-A)
[Graves & Fernandez+ 06, Graves 12, Sak & Shannon+ 17]:
– remove transition probabilities
and add special symbol: blank or ϵ:

$$\sum_{y_t \in \{a_s\} \cup \epsilon} q_\vartheta(y_t|a_{s'}, x_1^T) = 1$$



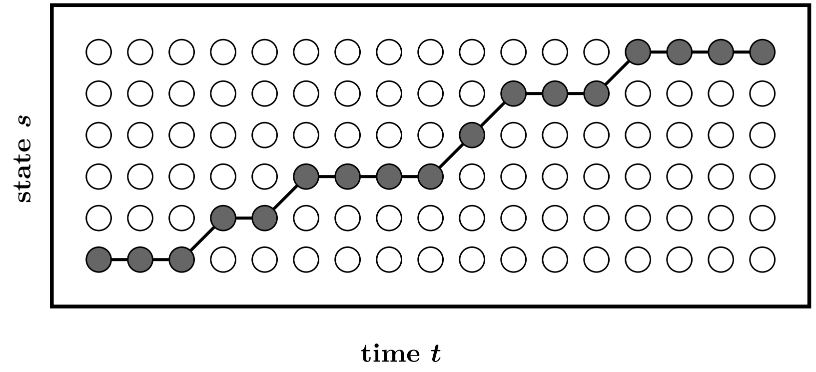
– interpretation as probability of symbol repetition
and segmental model [Zhou & Zeyer+ 2021]
– transducer variant: limited LM context [Zhou & Berger+ 2021]

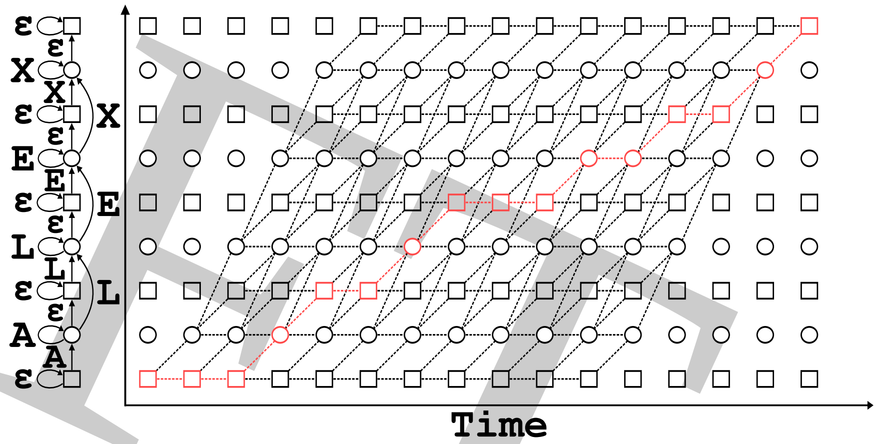


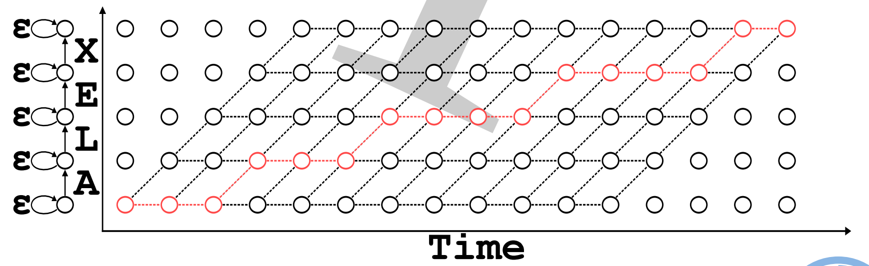
unifying principles
for posterior HMM, CTC and transducer with no internal LM:
– hidden variable: alignment path
– sum criterion (or best path) along with EM-style training
– acoustic encoder to be included

principal considerations:
– with ϵ/blank or transition prob.
– use of frame label priors
– duration constraints
– acoustic context dependence of labels: monophone, triphone, CART labels
– LM context in output generation: recursive, limited, none

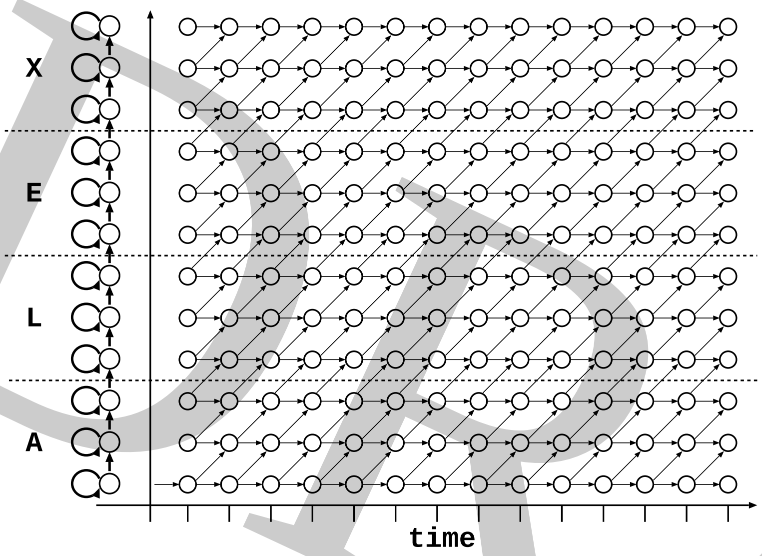
practical tricks (maybe important):
– chunking
– spec-augment
– label smoothing
– extended training criteria: encoder loss, focal loss
– sub-sampling (e.g. 10→30→60 msec)
– ...

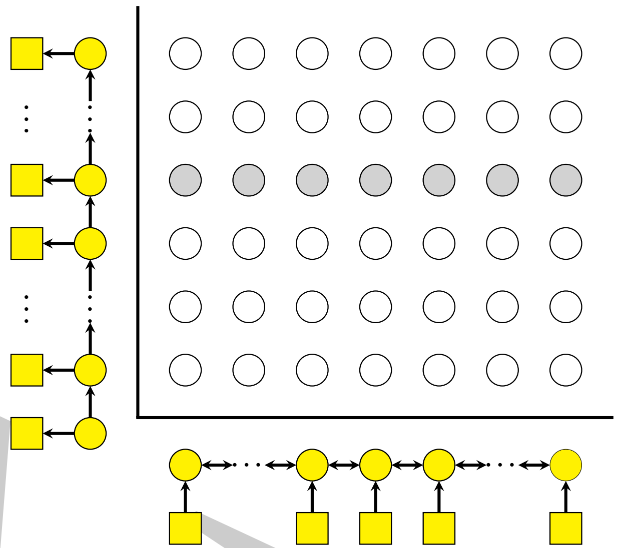






Frame Label Posterior Probability





key quantity:
frame label posterior at time t
over labels $a = a_s$ for state/segment s:

$$q_t(a_s|x_1^T) \equiv q(y_t = a_s|h_t(x_1^T))$$

with frame labels $y_t,\ t = 1, ..., T$

acoustic encoder / feature extraction:
– deep MLP with window around t: $x_{t-\delta}^{t+\delta}$
– bi-direct. (LSTM) RNN: full context x_1^T
– transformer and conformer

note: huge progress 1990-2020

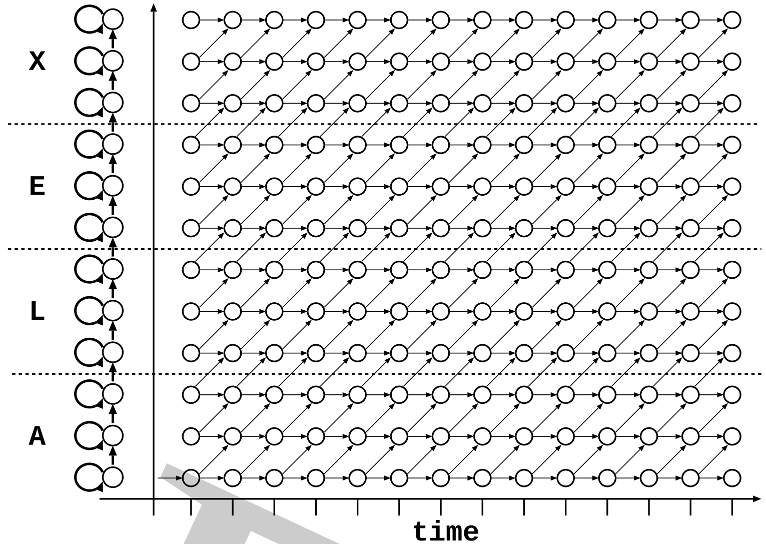
label posteriors	$\circ$	$\circ$	...	$\circ$	$q(a\|h_t)$	$\circ$	...	$\circ$	$\circ$
features	h_1	h_2	...	h_{t-1}	h_t	h_{t+1}	...	h_{T-1}	h_T
acoustic vectors	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T

Jaca: HMM-FST

Preview: From Generative HMM to RNN-Transducer

steps:
– change direction of HMM
– output symbols: phonemes, letters, letter BPEs
– introduce blanks/ϵ
– drop transition probabilities
– use bid. RNN for acoustic encoder
– use unid. RNN for output dependencies
– training: extend EM algorithm by backpropagation

Hidden Markov Model (HMM): Classical vs. Hybrid HMM



– **sequence of acoustic vectors:**
$X = x_1^T = x_1 ... x_t ... x_T$ **over time** $t = 1, ..., T$

– **sequence of states/segments** $s = 1, ..., S$
$s_1^T = s_1 ... s_t ... s_T$ **over time** t
with phonetic/graphemic labels:
$a_1^S = a_1 ... a_s ... a_S$
$= W$**: word sequence**

- **classical HMM: generative model for input sequence** x_1^T**:**

$$p(x_1^T | W = a_1^S) = \sum_{s_1^T} \prod_t p(s_{t+1}|s_t, a_{s_t}) \cdot p(x_t | a_{s=s_t})$$

- **hybrid HMM: discriminative model for output sequence** a_1^S**:**
[Bourlard & Wellekens 89] machine learning point-of-view:
it is much(!) better to model $p(a_s|x_t)$ **than** $p(x_t|a_s)$ **:**

$$p(x_t|a_s) = q(a_s|x_t) \cdot p(x_t) \Big/ q(a_s) \qquad \text{(note: approximative relation!)}$$

$$p(W = a_1^S | x_1^T) = \sum_{s_1^T} \prod_t p(s_{t+1}|s_t, a_{s_t}) \cdot p(a_{s=s_t}|x_t)$$

Direct or Posterior HMM: Re-consider Problem

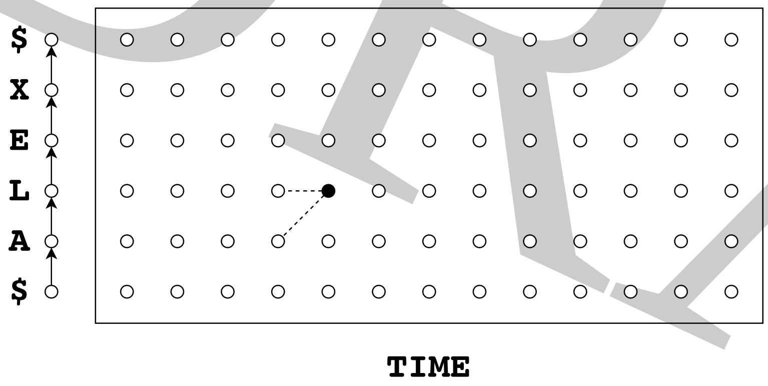
distinguish: frame labels vs. segment labels
for each frame label: stay or next segment.

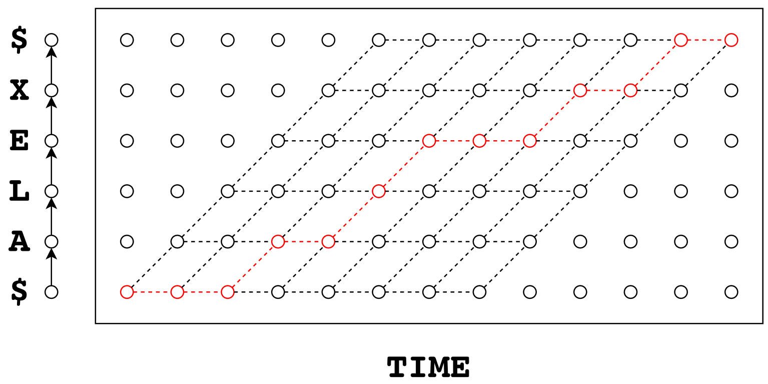
:

Direct or Posterior HMM (Variants: CTC, Transducer, ...)
((view: how to reach $[t,\ s=s_t]$?)

three sequences over time:

$$\begin{aligned} x_1^T &= x_1, ..., x_t, ..., x_T \\ s_1^T &= s_1, ..., s_t, ..., s_T \\ y_1^T &= y_1, ..., y_t, ..., y_T \end{aligned}$$





path consists of transitions reaching $[t,\ s=s_t]$:
first transition δ_t and then label y_t:

$$[t-1,\ s_{t-1}] \rightarrow [t,\ s=s_t=s_{t-1}+\delta_t] \qquad \delta_t \in \{0,1\}$$

JOINT event of δ_t and frame label y_t:

$$[\delta_t, y_t]: \quad p\big([\delta_t,\ y_t]\big| ..., x_1^T\big)$$

link to state s with label $a_s \in a_1^S$:

$$[\delta_t, y_t]: \quad p\big([\delta_t,\ y_t=a_s]\big| ..., x_1^T\big)$$

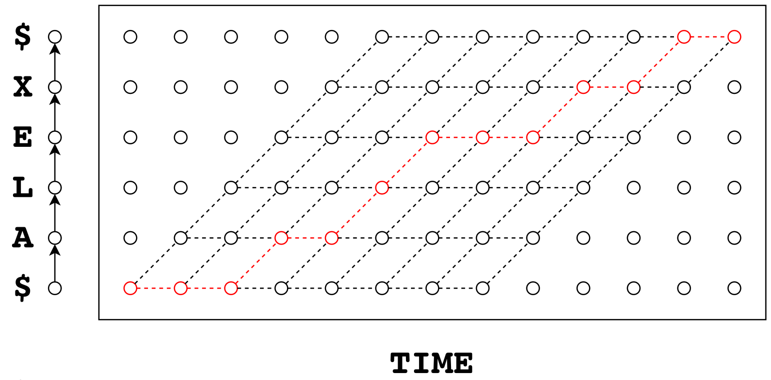
first-order dependence in a_1^S:

$$[\delta_t, y_t]: \quad p\big([\delta_t,\ y_t=a_s]\big| a_{s-1}, ..., x_1^T\big)$$

remarks:
- **for full context, replace a_{s-1} by a_0^{s-1}**
- **alternative view: how to leave $[t,\ s=s_t]$?**
  first label y_t and then transition δ_t:
  $$p\big([y_t=a_s,\ \delta_t]\big| a_{s-1}, x_1^T\big)$$

Mathematical Formalism: Direct or Posterior HMM for $p(a_1^S|x_1^T)$ (view: how to reach $[t,\ s = s_t]$?)

formal derivation of full model:



$$p(a_1^S|x_1^T) = \sum_{s_1^T} p(a_1^S, s_1^T|x_1^T)$$

finite-state model: factorization over t:
first-order model in s_1^T and a_1^S

$$= \sum_{s_1^T} \prod_t p\big([s_t, y_t = a_{s_t}\big| s_{t-1}, a_{s_{t-1}}, x_1^T\big)$$

difference in state/segment indices: $\delta_t := s_t - s_{t-1}$

$$= \sum_{s_1^T} \prod_t p\big([\delta_t, y_t = a_{s_t}]\big| a_{s_{t-1}}, x_1^T\big)$$

explicit segmental interpretation:

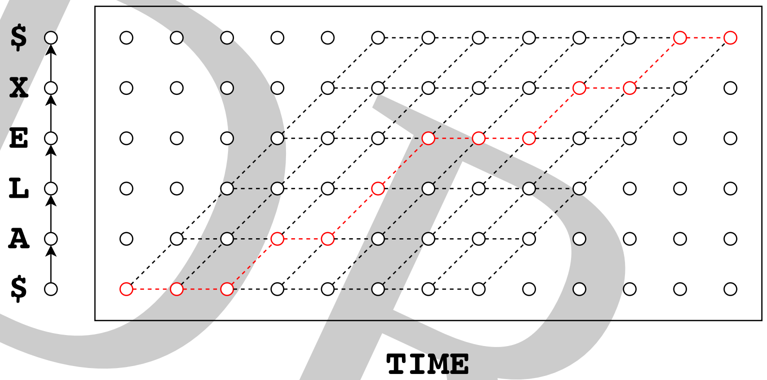
$$= \sum_{s_1^T} \prod_s \prod_{t:\, s_t = s} p\big([\delta_t, y_t = a_s]\big| a_{s-1}, x_1^T\big)$$

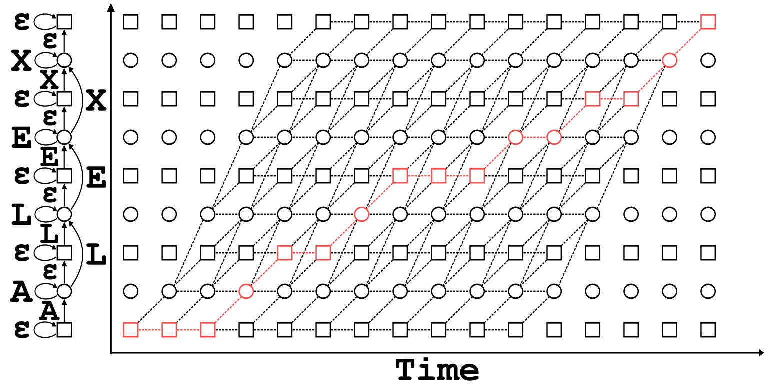
acoustic encoder : $h_t = h_t(x_1^T)$

$$= \sum_{s_1^T} \prod_s \prod_{t:\, s_t = s} p\big([\delta_t, y_t = a_s]\big| a_{s-1}, h_t(x_1^T)\big)$$

frames t within segment s:
– **first frame:** $\delta_t = 1$
– **other frames:** $\delta_t = 0$

Direct HMM and Variants: CTC, [RNN-] Transducer, Blank/ϵ Models





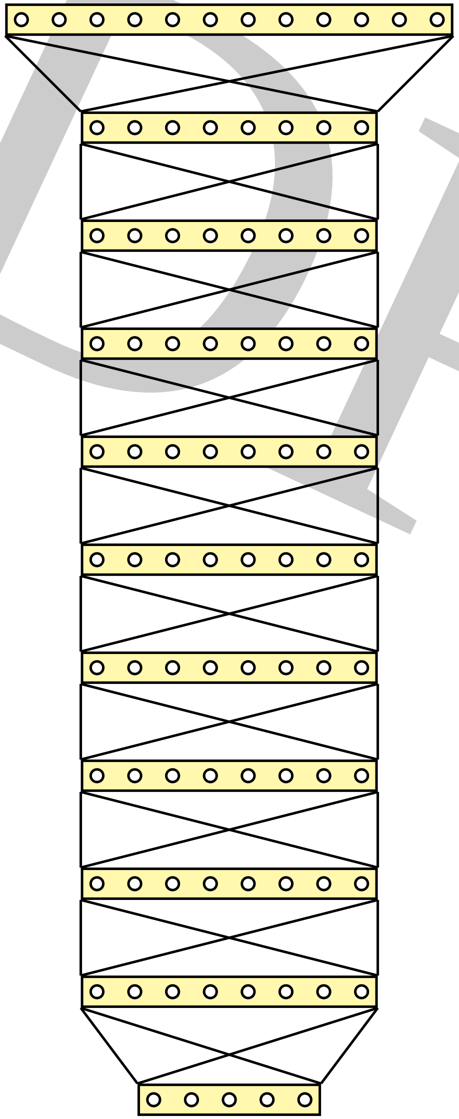
direct HMM: without and without blanks/ϵ

question: how to model the joint event $[\delta_t,\, y_t = a_s]]$ in $p\big([\delta_t, y_t = a_s]\big|a_{s-1}, x_1^T\big)$?
here: no separation of transition and label probabilities !

- direct HMM (no blanks/ϵ):
  keep the original joint alphabet for the ANN output nodes:
  $\Big\{[\delta_t \in \{0,1\},\, y_t = a_s]\Big\}$ = 2 x (segment label alphabet) + silence label

- transducer: with blanks/ϵ:
  simplify the alphabet of joint events $[\delta_t,\, y_t = a_s]$:
  $[\delta_t = 1, y_t = a_s] \;:= a_s \qquad\qquad [\delta_t = 0, y_t = a_s] \;:= \epsilon$
  resulting alphabet: 1x (segment label alphabet) + ϵ (also for silence)

Artificial Neural Networks (ANN) and Deep Learning:



question: what is different now after 30 years?

answer: we have learned how to (better) handle a complex numerical optimization problem:

- more powerful hardware (e. g. GPUs)
- empirical recipies for optimization: practical experience and heuristics, e.g. layer-by-layer pretraining
- result: we are able to handle more complex architectures (deep MLP, RNN, attention, transformer, etc.)

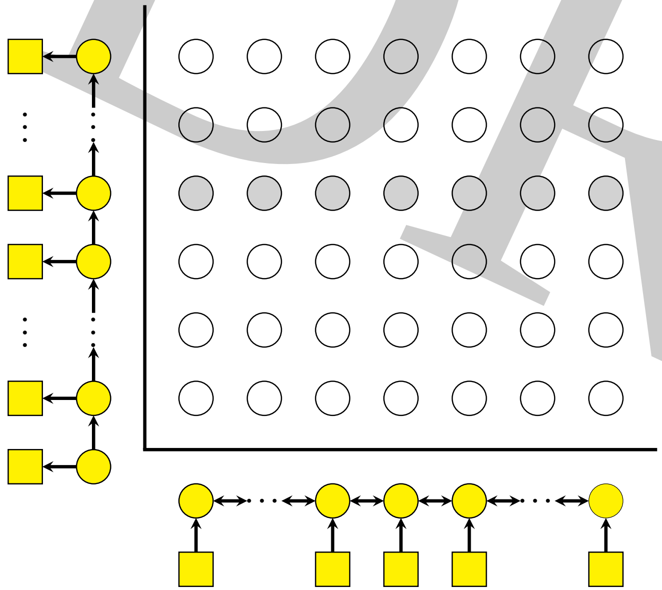
my interpretation: 2022's most advanced ASR systems:

$=$ sophisticated feature extraction/representation

$+$ softmax (= Gaussian posterior)

Input-Output Alignment: Attention and Transducer

common properties:
– input: acoustic encoder: representation/state vectors $h_t = h_t(x_1^T), t = 1, ..., T$
– output: (phoneme) labels $a_s,\ s = 1, ..., S$ with/without integrated language model



• (cross-) attention: direct factorization:

$$p(a_1^S|x_1^T) \;=\; \prod_s p(a_s|a_0^{s-1}, x_1^T) = \prod_s p(a_s|a_{s-1}, r_{s-1}, c_s)$$

$$c_s \;:=\; \sum_t p(t|a_0^{s-1}, x_1^T) \cdot h_t$$

with context vector c_s and output state vector r_s

criticism for ASR: lack of strict monotonicity
and localization

• finite-state transducer (direct HMM, CTC, RNN-T, ...):
introduce hidden paths and then factorize:

$$p(a_1^S|x_1^T) \;=\; \sum_{s_1^T} p\Big(s_1^T, a_1^S|h_1^T(x_1^T)\Big)$$

$$= \sum_{s_1^T} \prod_t p\Big(s_{t+1}, y_t = a_{s_t}\Big| s_t, a_0^{s_t-1}, h_1^T(x_1^T)\Big)$$

details: RWTH papers at ICASSP and Interspeech

representation/state vectors h_t:
– deep MLP: finite window
– RNN and LSTM-RNN
– self-attention (transformer)

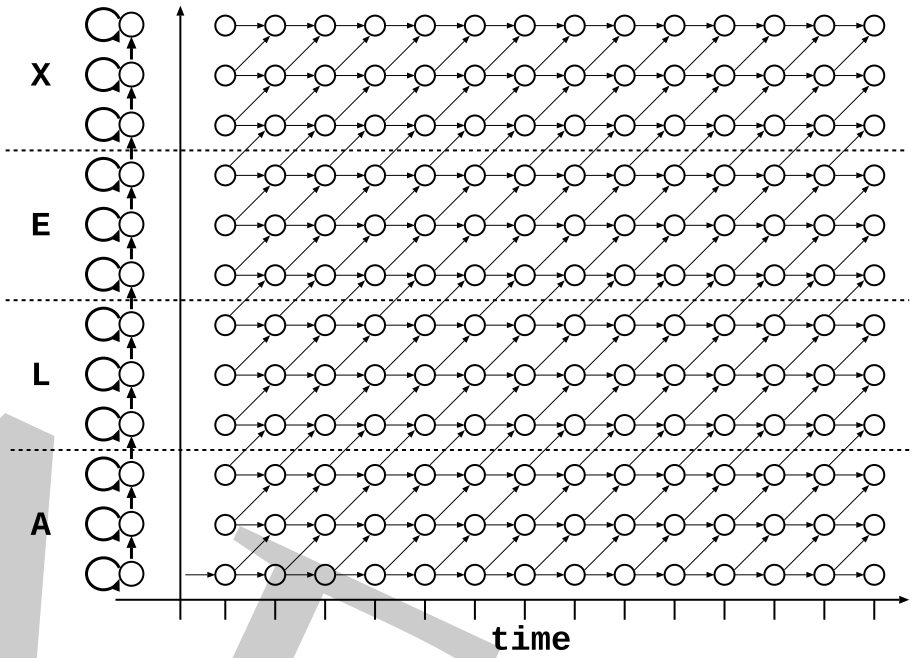
similar: output string

6.2 Training: Sum Criterion

Training of First-Order Models: Baseline Approach

- **HMM structure: generative, hybrid, CTC, RNN-T, ...**
  important property: first-order dependence
- **most interesting application of the EM algorithm**
- **today's importance of EM algorithm:**
  – is less important than before
  – is superseded by gradient search (backpropagation)

First-Order Models: Unified Notation and Approach



– sequence of acoustic vectors:
$x_1^T = x_1 ... x_t ... x_T$ over time t

– sequence of states $s = 1, ..., S$
$s_1^T = s_1 ... s_t ... s_T$ over time t
with associated state labels:
$a_1^S = a_1 ... a_s ... a_S$
$= W$: word sequence

hybrid HMM (similar for CTC and RNN-T):

- full notation with parameters λ:

$$q_\lambda(W = a_1^S | x_1^T) = \sum_{s_1^T} \prod_t q(s_t | s_{t-1}, W, \lambda) \cdot q_t(a_{s=s_t} | x_1^T, \lambda)$$

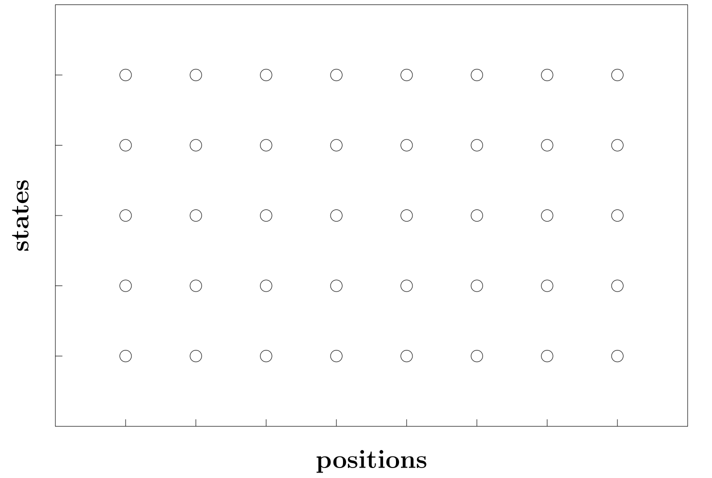
- simplified notation (for DP, training and EM algorithm):

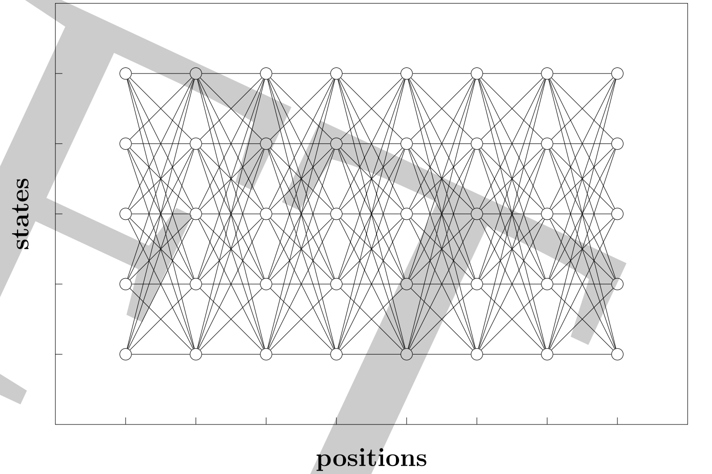
$$\lambda \rightarrow \sum_{s_1^T} \prod_t q_t(s_t, s_{t-1}, x_1^T; \lambda)$$

note the notation: output string $W = a_1^S$ is dropped

- **sequence over time (= positions):**
  $x = x_1^T = x_1 ... x_t .... x_T$
- **model with parametes λ:**
  - **states** $s = 1, ..., S$
  - **state transitions** $(t-1, s') \to (t, s)$ **with local score (no normalization!)**

$$q_t(s', s, x; \lambda) \;\geq\; 0$$





- **global score: sum over all state sequences s_1^T:**

$$f(\lambda, x) := \sum_{s_1^T} \prod_{t=1}^{T} q_t(s_{t-1}, s_t, x; \lambda)$$

- **functional dependence:**

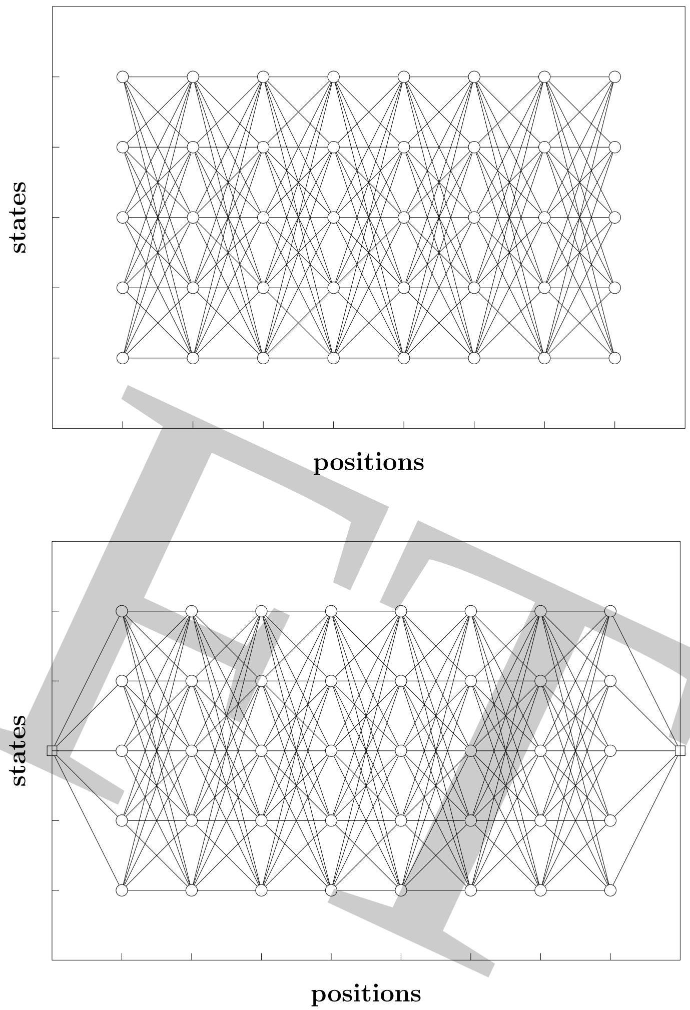
$$(x, \lambda) \;\to\; \{q_t(s', s, x; \lambda) : s', s, t\} \;\to\; f(\lambda, x) := \sum_{s_1^T} \prod_{t=1}^{T} q_t(s_{t-1}, s_t, x; \lambda)$$

Mathematical Specification: Constraints on Transitions

most general case: $q_t(s_{t-1}, s_t, x; \lambda)$
with full dependence on transitions $s_{t-1} \rightarrow s_t$

specific models CTC and RNN-T:
– first-order for model structure/transitions
– zero-order for scores $q_t(s_t, x_1^T; \lambda)$

boundary conditions:
– beginning: $t = 0 : \ s_{t=0} =?$
– end: $t = T$
additional specifications required ...
(but mostly omitted)

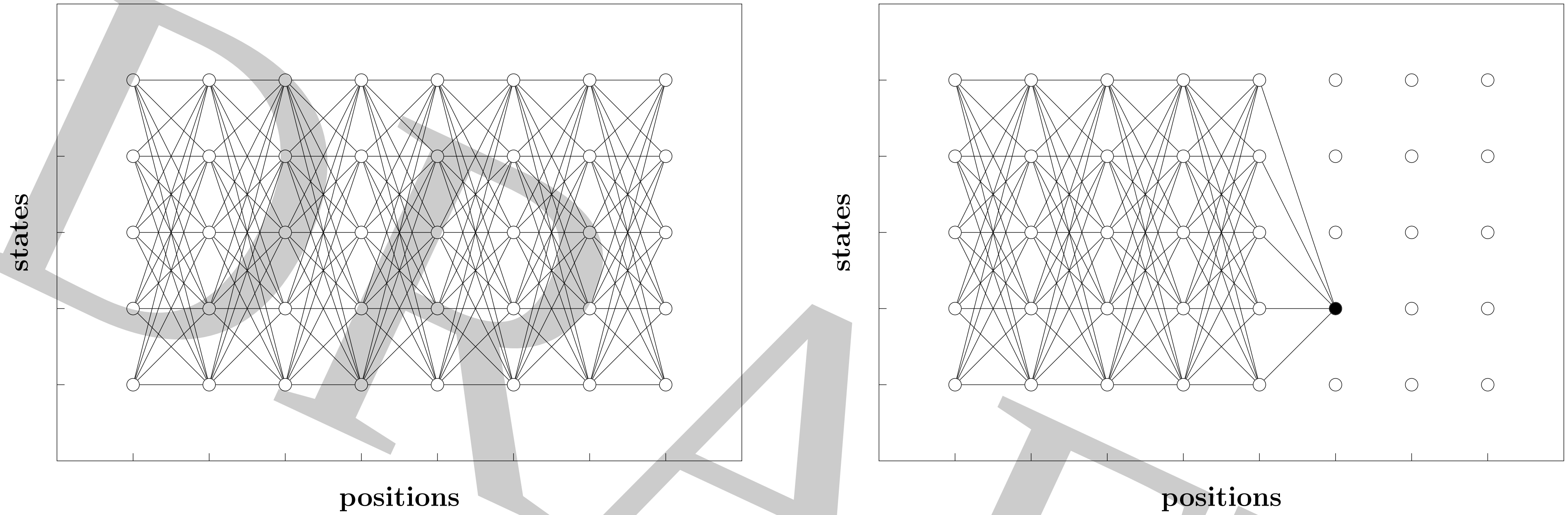


First-Order Models: Unified Concept

three problems:

- **scoring: efficient calculation of the sum needed in training and recognition**
- **training using sum criterion: EM algorithm and direct gradient/backpropagation**
- **training using sum criterion with first-order dependencies: EM algorithm and direct gradient/backpropagation**

comparison: direct gradient vs. EM algorithm



remarks:

– two variants: sum or maximum of all paths (best path, Viterbi)

$$\sum_{s_1^T} \prod_{t=1}^{T} q_t(s_{t-1}, s_t, x; \lambda) \qquad \textbf{vs.} \qquad \max_{s_1^T} \left\{ \prod_{t=1}^{T} q_t(s_{t-1}, s_t, x; \lambda) \right\}$$

– both variants (sum and maximum) are used in recognition and training

Scoring: Dynamic Programming

problem: efficient calculation of global score:

$$\sum_{s_1^T} \prod_{\tau=1}^{T} q_\tau(s_{\tau-1}, s_\tau, x)$$

define auxiliary function for sub-problem:

$$Q_t(s;x) \ := \sum_{s_1^t : s_t = s} \prod_{\tau=1}^{t} q_\tau(s_{\tau-1}, s_\tau, x)$$

DP recursion:

$$Q_t(s;x) \ = \sum_{s'} Q_{t-1}(s', x) \cdot q_t(s', s, x)$$

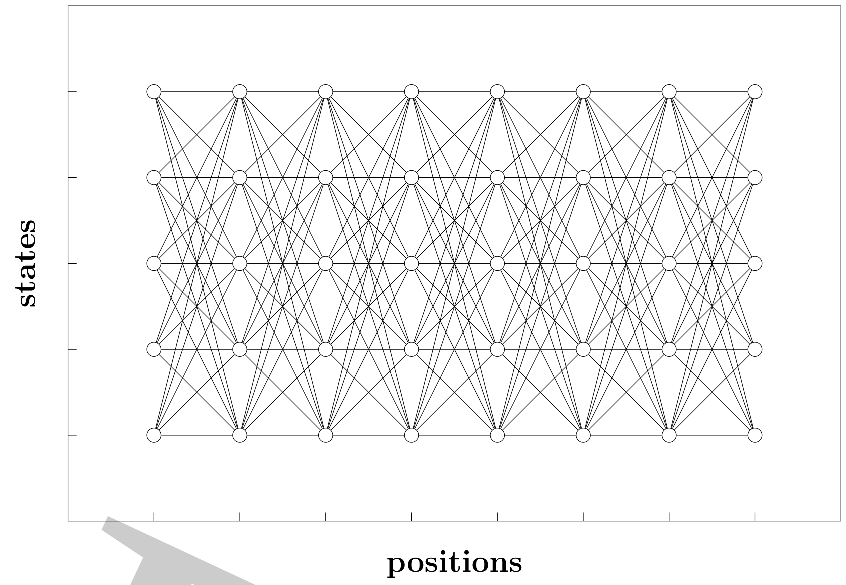
straightforward proof:

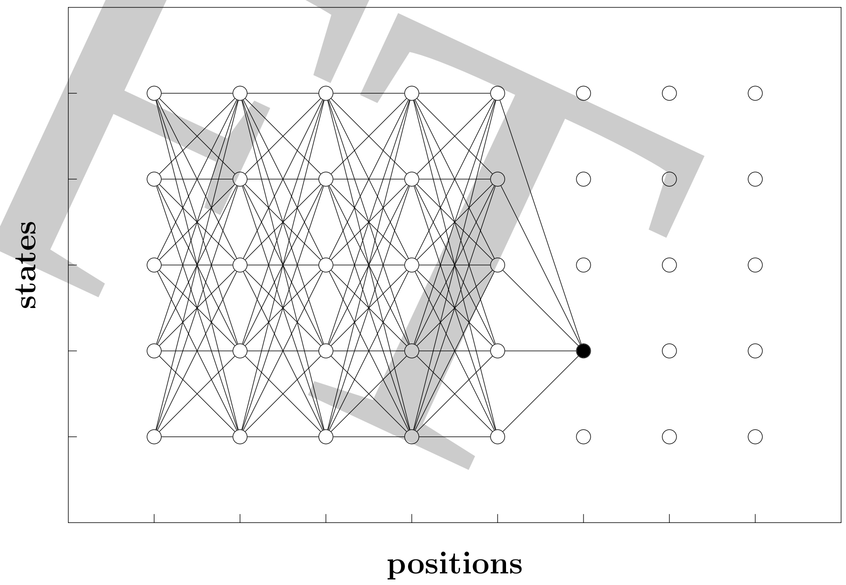
exploit 'additivity'of the problem

complexity:

- **basic unit: local score $q_t(s', s, x)$**
- **general case: $T \cdot S^2$ operations**
- **constrained transitions: $T \cdot S \cdot \Delta S$ operations**

number of paths: S^T





Dynamic Programming: Best Path

from sum to maximum:

$$\max_{s_1^T} \left\{ \prod_{\tau=1}^{T} q_\tau(s_{\tau-1}, s_\tau, x) \right\}$$

we use the same concept and define auxiliary function for sub-problem:

$$Q_t(s;x) \;:=\; \max_{s_1^t : s_t = s} \left\{ \prod_{\tau=1}^{t} q_\tau(s_{\tau-1}, s_\tau, x) \right\}$$

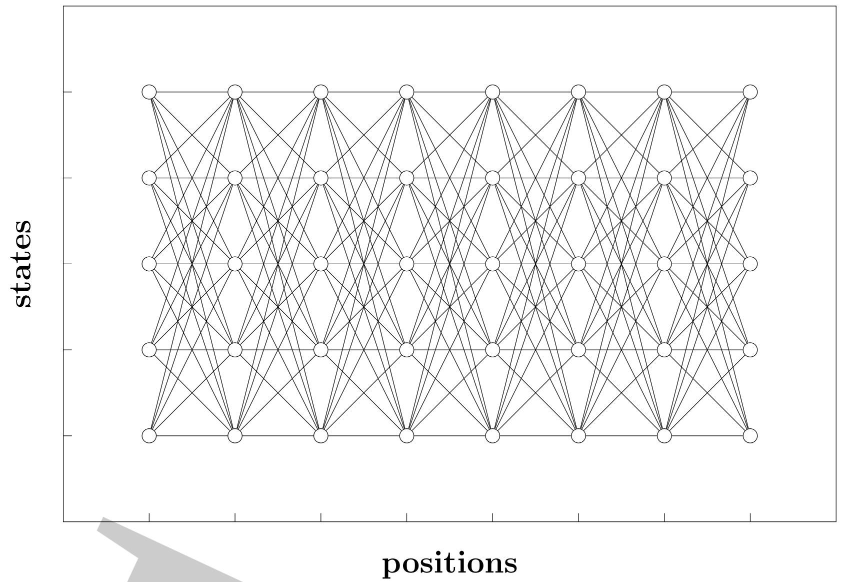
DP recursion:

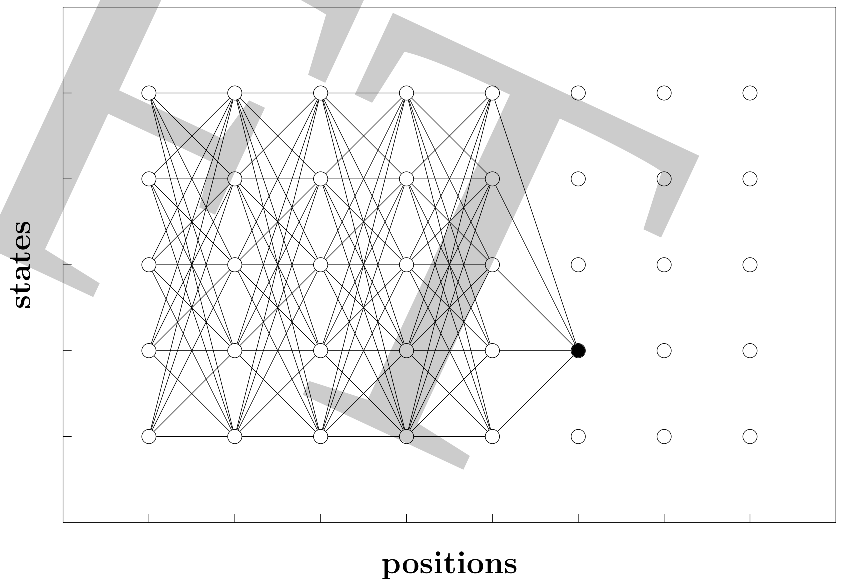
$$Q_t(s;x) \;=\; \max_{s'} \left\{ Q_{t-1}(s',x) \cdot q_t(s',s,x) \right\}$$
$$B_t(s;x) \;=\; \arg\max_{s'} \left\{ Q_{t-1}(s',x) \cdot q_t(s',s,x) \right\}$$

additional auxiliary quantity: backpointers $B_t(s;x)$:

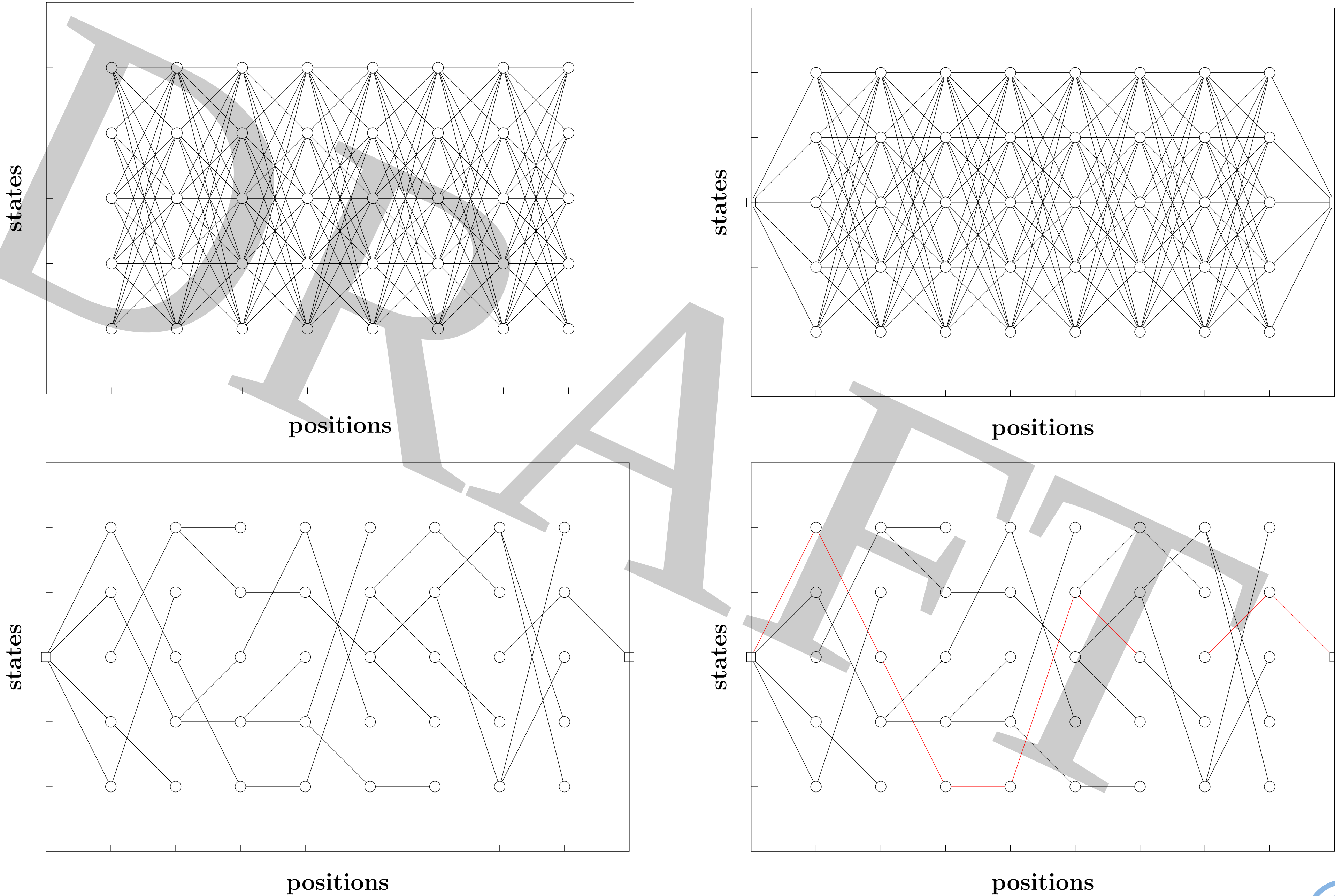
– structure: "parallel" to the scores $Q_t(s;x)$

– useful for recovering the best path





Dynamic Programming: Backpointers and Best Path



specification of optimization problem:

- **training input sequences $x_n,\ n = 1, ...N$, each $x_n = [x_1...x_t...x_{T_n}]$ with length T_n (note the dependene on n)**
- **first-order model: function to be optimized:**

$$\lambda \to F(\lambda) \ := \ \sum_{n=1}^{N} \log \sum_{s_1^{T_n}} \prod_{t=1}^{T_n} q_t(s_{t-1}, s_t, x_n; \lambda) \ = \ \sum_{n=1}^{N} \log \sum_{s_1^{T_n}} \ \tilde{q}(s_1^{T_n}, x_n; \lambda)$$

interpretation: mixture model or sum criterion:

$$\lambda \to F(\lambda) \ := \sum_{n=1}^{N} \log \sum_{y} q(y, x_n; \lambda)$$

- **presentation of EM algorithm:**
  - **first step: introduce approach for mixture model with hidden variable y**
  - **second step: identify the state sequence s_1^T with the hidden variable of a mixture model and exploit the first-order structure of the model**

Training Strategy: Two Approaches

- **classical EM framework:**
  - **applied to sum criterion with no closed-form solutions**
  - **result: gradient search within EM framework**
- **backpropagation:**
  - **directly compute gradient of sum criterion**

remarks presentation of EM algorithm (beyond literature):
– model with no normalization
– proof by using log-sum inequality
result: shortest proof ever (?)
– natural by-product: *posterior weights*
– extension beyond EM algorithm with closed-form solutions:
gradient search as part of EM algorithm

consider function difference between new $\tilde{\lambda}$ and old parameters λ:

$$F(\tilde{\lambda}) - F(\lambda) = \sum_n \log \frac{\sum_y q(y, x_n; \tilde{\lambda})}{\sum_y q(y, x_n; \lambda)}$$

log-sum inequality for non-negative numbers a_k and b_k (extension of divergence inequality):

$$\log \frac{\sum_k a_k}{\sum_k b_k} \geq \sum_k \frac{b_k}{\left(\sum_{k'} b_{k'}\right)} \log \frac{a_k}{b_k}$$

$$\geq \sum_n \sum_y w(y|x_n, \lambda) \cdot \log \frac{q(y, x_n; \tilde{\lambda})}{q(y, x_n; \lambda)}$$

with weights: $$w(y|x, \lambda) = \frac{q(y, x; \lambda)}{\sum_{y'} q(y', x; \lambda)}$$

– important result: normalized posterior weights (without assuming probabilistic models!)
– standard proof: divergence inequality for normalized models

Training: Classical EM Algorithm

- **we have proved the baseline EM inequality:**

$$F(\tilde{\lambda}) - F(\lambda) \;\geq\; \sum_n \sum_y w(y|x_n,\lambda) \cdot \log \frac{q(y,x_n;\tilde{\lambda})}{q(y,x_n;\lambda)}$$

with weights: $$w(y|x,\lambda) \;=\; \frac{q(y,x;\lambda)}{\sum_{y'} q(y',x;\lambda)}$$

$$= \; Q(\lambda,\tilde{\lambda}) - Q(\lambda,\lambda)$$

with: $$Q(\lambda,\tilde{\lambda}) \;:=\; \sum_n \sum_y w(y|x_n,\lambda) \cdot \log\, q(y,x_n;\tilde{\lambda})$$

terminology: EM operations applied to auxiliary function $Q(\lambda,\tilde{\lambda})$:
– E: expectation using posterior weights $w(y|x,\lambda)$
– M: maximization of $\tilde{\lambda} \to Q(\lambda,\tilde{\lambda})$ in lieu of $\tilde{\lambda} \to F(\tilde{\lambda})$

- **iterative optimization: two alternating steps:**
  – compute posterior weights $w(y|x,\lambda)$ using present value of λ
  – maximize $Q(\lambda,\tilde{\lambda})$ over $\tilde{\lambda}$:

$$\hat{\lambda} \;=\; \operatorname*{argmax}_{\tilde{\lambda}} Q(\lambda,\tilde{\lambda}) \;=\; \operatorname*{argmax}_{\tilde{\lambda}} \Big\{ \sum_n \sum_y w(y|x_n,\lambda) \cdot \frac{\partial}{\partial\tilde{\lambda}} \log\, q(y,x_n;\tilde{\lambda}) \Big\}$$

– closed-form solution in each iteration (in classical set-up)
– guaranteeed (local) convergence

maximization of $Q(\lambda, \tilde{\lambda})$ function:

$$\underset{\tilde{\lambda}}{\operatorname{argmax}}\, Q(\lambda, \tilde{\lambda}) \;=\; \underset{\tilde{\lambda}}{\operatorname{argmax}} \left\{ \sum_n \sum_y w(y|x_n, \lambda) \cdot \log\, q(y, x_n; \tilde{\lambda}) \right\}$$

two variants:

- **classical variant: closed-form solution**
  – possible only for simple models
  – advantage: no heuristics and guaranteed convergence
  typical example: Gaussian mixture (first-order: Gaussian HMM)
- **extended variant: no closed-form solution**
  remedy: gradient search inside EM algorithm

$$\frac{\partial}{\partial \tilde{\lambda}} Q(\lambda, \tilde{\lambda}) \;=\; \sum_n \sum_y w(y|x_n, \lambda) \cdot \frac{\partial}{\partial \tilde{\lambda}} \log\, q(y, x_n; \tilde{\lambda})$$

result: backpropagation for

$\tilde{\lambda} \rightarrow Q(\lambda, \tilde{\lambda})$ **rather than** $\lambda \rightarrow F(\lambda)$

next topic:

backpropagation applied to $\lambda \rightarrow F(\lambda)$ directly

definitions: function to be optimized and posterior weights:

$$F(\lambda) := \sum_n \log \sum_y q(y, x_n; \lambda) \qquad w(y|x_n, \lambda) := \frac{q(y, x_n; \lambda)}{\sum_{y'} q(y', x_n; \lambda)}$$

direct calculation of gradient:

$$\begin{aligned}
\frac{\partial}{\partial\lambda} F(\lambda) &= \sum_n \frac{\partial}{\partial\lambda} \log \sum_y q(y, x_n; \lambda) \\
&= \sum_n \sum_y \frac{1}{\sum_{y'} q(y', x_n; \lambda)} \cdot \frac{\partial}{\partial\lambda} q(y, x_n; \lambda)
\end{aligned}$$

use: $\displaystyle \frac{\partial}{\partial\lambda} f(\lambda) = f(\lambda) \cdot \frac{\partial}{\partial\lambda} \log f(\lambda)$

$$\begin{aligned}
&= \sum_n \sum_y \frac{q(y, x_n; \lambda)}{\sum_{y'} q(y', x_n; \lambda)} \cdot \frac{\partial}{\partial\lambda} \log q(y, x_n; \lambda) \\
&= \sum_n \sum_y w(y|x_n, \lambda) \cdot \frac{\partial}{\partial\lambda} \log q(y, x_n; \lambda)
\end{aligned}$$

functional dependence:

$$F(\lambda) := \sum_n \log \sum_y q(y, x_n; \lambda) \qquad w(y|x_n, \lambda) := \frac{q(y, x_n; \lambda)}{\sum_{y'} q(y', x_n; \lambda)}$$

- **backpropagation:**

$$\frac{\partial}{\partial \lambda} F(\lambda) = \sum_n \sum_y w(y|x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q(y, x_n; \lambda)$$

- **gradient inside EM algorithm:**

$$\tilde{\lambda} := \operatorname*{argmax}_{\tilde{\lambda}} \left\{ Q(\lambda, \tilde{\lambda}) \right\}$$

$$\frac{\partial}{\partial \tilde{\lambda}} Q(\lambda, \tilde{\lambda}) = \sum_n \sum_y w(y|x_n, \lambda) \cdot \frac{\partial}{\partial \tilde{\lambda}} \log q(y, x_n; \tilde{\lambda})$$

difference: different update strategies for posterior weights $w(y|x_n, \lambda)$

Summary: Training for Sum Criterion

- **structure of optimization problem:**
  – sum over hidden variable:
  – no explicit probabilistic structure or interpretation
- **remark: existence of (local) optimum:**
  has to be verified independenly (well defined optimization problem?)
- **gradient with and without EM algorithm:**
  – (normalized) posterior weights
  – resulting from the optimization structure
- **difference with and without EM algorithm:**
  update strategy for posterior weights
- **most important structure:**
  first-order models like hybrid HMM, CTC and RNN-T

general approach for both EM and backpropagation algorithms:

– **interpret the complele state sequence s_1^T as a hidden variable in sum criterion**
– **use mixture model as a starting point**
– **term re-writing: exploit the first-order structure of the model**

transfer the mixture results to first-order model with $y \equiv s_1^T$:

$$F(\lambda) := \sum_n \log \sum_{s_1^{T_n}} \prod_t q_t(s_{t-1}, s_t, x_n; \lambda)$$

$$\frac{\partial Q(\lambda, \tilde{\lambda})}{\partial \tilde{\lambda}} = \sum_n \sum_{s_1^{T_n}} w(s_1^{T_n} | x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log \prod_t q_t(s_{t-1}, s_t, x_n; \tilde{\lambda})$$

$$w(s_1^{T_n} | x_n, \lambda) := \frac{\prod_t q_t(s_{t-1}, s_t, x_n; \lambda)}{\sum_{\tilde{s}_1^{T_n}} \prod_t q_t(\tilde{s}_{t-1}, \tilde{s}_t, x_n; \lambda)}$$

$$= \sum_n \sum_t \sum_{s_1^{T_n}: s'=s_{t-1}, s=s_t} w(s_1^{T_n} | x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s', s, x_n; \tilde{\lambda})$$

$$= \sum_n \sum_t \sum_{s',s} w_t(s', s | x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s', s, x_n; \tilde{\lambda})$$

with posterior weights $w_t(s', s | x_n, \lambda)$ for input sequence x_n:

$$w_t(s', s | x_n, \lambda) := \sum_{s_1^{T_n}: s'=s_{t-1}, s_t=s} w(s_1^{T_n} | x_n, \lambda) = \frac{\sum_{s_1^{T_n}: s'=s_{t-1}, s_t=s} \prod_t q_t(s_{t-1}, s_t, x_n; \lambda)}{\sum_{\tilde{s}_1^{T_n}} \prod_t q_t(\tilde{s}_{t-1}, \tilde{s}_t, x_n; \lambda)}$$

transfer the results from mixture model to first-order model:

$$F(\lambda) \; := \sum_n \log \sum_{s_1^{T_n}} \prod_t q_t(s_{t-1}, s_t, x_n; \lambda)$$

$$\frac{\partial}{\partial \lambda} F(\lambda) \; = \sum_n \sum_{s_1^{T_n}} w(s_1^{T_n}|x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log \prod_t q_t(s_{t-1}, s_t, x_n; \lambda)$$

$$w(s_1^{T_n}|x_n, \lambda) := \frac{\prod_t q_t(s_{t-1}, s_t, x_n; \lambda)}{\sum_{\tilde{s}_1^{T_n}} \prod_t q_t(\tilde{s}_{t-1}, \tilde{s}_t, x_n; \lambda)}$$

$$= \sum_n \sum_t \sum_{s_1^{T_n}: s'=s_{t-1}, s=s_t} w(s_1^{T_n}|x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s', s, x_n; \lambda)$$

$$= \sum_n \sum_t \sum_{s', s} w_t(s', s|x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s', s, x_n; \lambda)$$

with posterior weights $w_t(s', s|x_n, \lambda)$ **as in classical EM algorithm for first-order models:**

$$w_t(s', s|x_n, \lambda) := \sum_{s_1^{T_n}: s'=s_{t-1}, s_t=s} w(s_1^{T_n}|x_n, \lambda)$$

functional dependence:

$$\lambda \;\rightarrow\; F(\lambda) \;:=\; \sum_n \log \sum_{s_1^{T_n}} \prod_t q_t(s_{t-1}, s_t, x_n; \lambda)$$

- **backpropagation: = gradient of original function $F(\lambda)$:**

$$\frac{\partial}{\partial \lambda} F(\lambda) \;=\; \sum_n \sum_t \sum_{s',s} w_t(s', s|x_n, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s', s, x_n; \lambda)$$

- **gradient inside EM algorithm: = gradient of auxiliary function $Q(\cdot, \cdot)$:**

$$\tilde{\lambda} \;:=\; \operatorname*{argmax}_{\tilde{\lambda}} \left\{ Q(\lambda, \tilde{\lambda}) \right\}$$

$$\frac{\partial}{\partial \tilde{\lambda}} Q(\lambda, \tilde{\lambda}) \;=\; \sum_n \sum_t \sum_{s',s} w_t(s', s|x_n, \lambda) \cdot \frac{\partial}{\partial \tilde{\lambda}} \log q_t(s', s, x_n; \tilde{\lambda})$$

remarks:
– both cases: the form looks like an optimization without the sum
– both cases: weighted form of gradient of the local model
– difference: update strategies for posterior weights $w_t(s', s|x_n, \lambda)$

EM Training: Forward-Backward Algorithm

posterior weight for input sequence $x = x_1^T$:

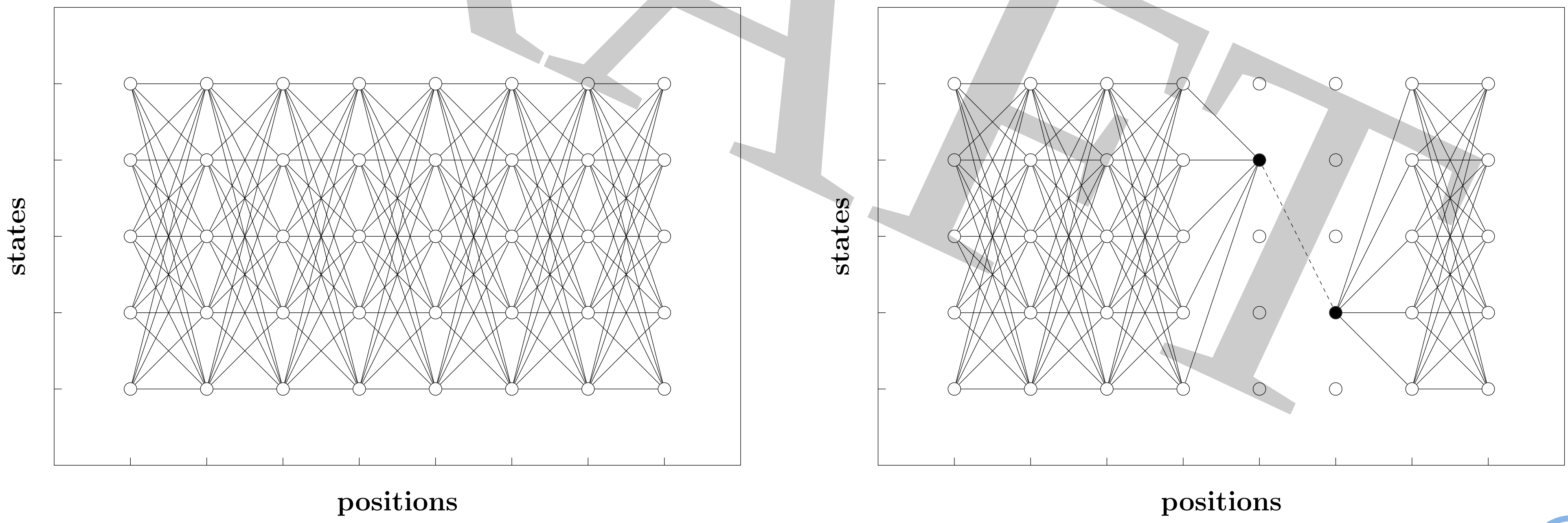
$$w_t(s', s|x, \lambda) := \sum_{s_1^T : s'=s_{t-1}, s_t=s} w(s_1^T|x, \lambda) = \frac{\sum_{s_1^T : s'=s_{t-1}, s_t=s} \prod_t q_t(s_{t-1}, s_t, x; \lambda)}{\sum_{\tilde{s}_1^T} \prod_t q_t(\tilde{s}_{t-1}, \tilde{s}_t, x; \lambda)}$$

forward–backward algorithm for computing the posterior weights:

– forward dynamic programming: $t = 1, 2, ..., T$

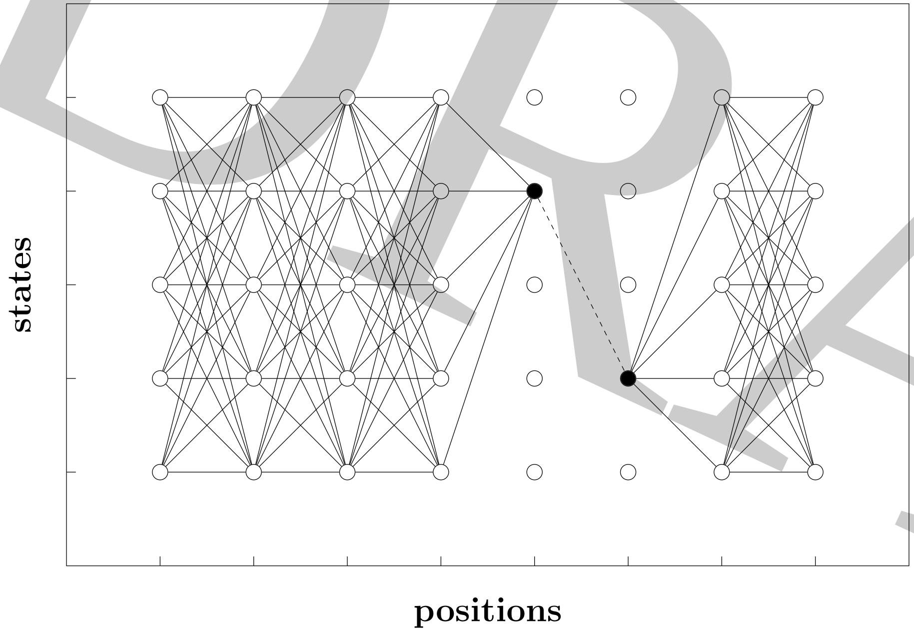
– backward dynamic programming: $t = T, T-1, ..., 1$

denominator **numerator for (s', s, t)**

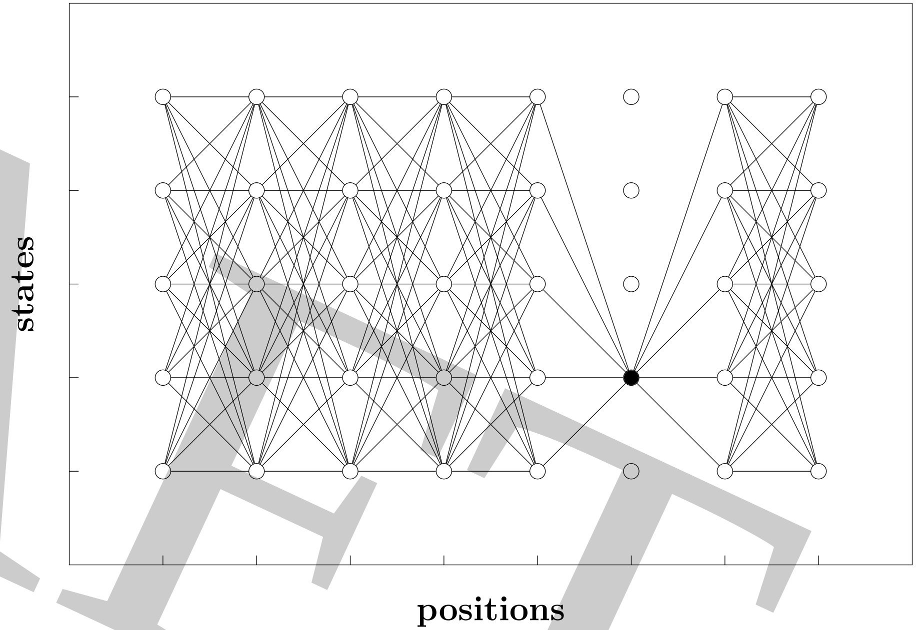


two types of posterior weights:

for transitions: $w_t(s', s|x, \lambda)$



for states: $w_t(s|x, \lambda) := \sum_{s'} w_t(s', s|x, \lambda)$



posterior weights: terminology in literature:
occupation probabilities (gamma)

EM Algorithm and Gradient Search:

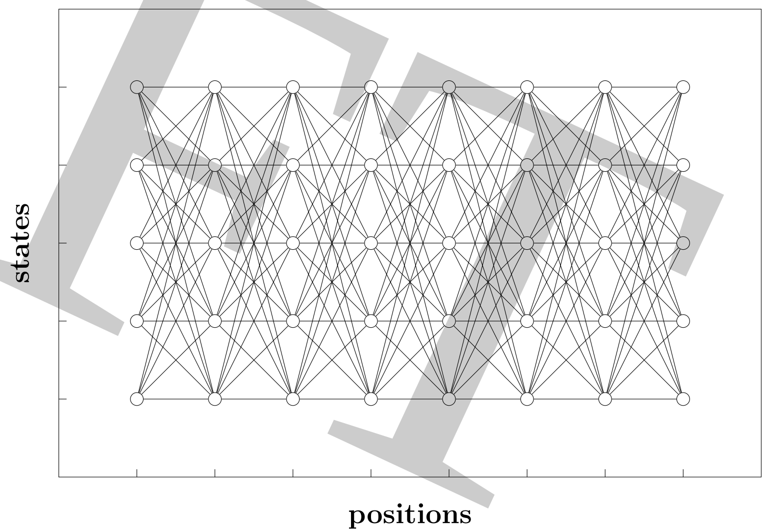
- classical EM algorithm:
  - optimization problem with explicit probability models with hidden random variables
    typical problem: maximum likelihood estimation for generative models
  - iterative procedure
  - within each iteration: closed-form solution
- optimization problems considered here:
  - general functions to be optimized (beyond the elementary ANN outputs):
    function with sum over 'hidden' variables
  - finite-state models for ASR and MT
- specific aspects covered:
  - revisiting the classical EM algorithm
  - EM algorithm with no closed-form solutions
  - relation between EM algorithm and gradient search
  - generalizing the EM algorithm:
    general optimization method beyond explicit probabilistic models

question: how does backpropagation handle the DP recursion?

application of recursion:
– produces forward–backward algorithm:
(exercise: work out the details)
– backpropagation is supported by automatic differentiation tools
(e.g. Tensorflow and PyTorch; computational graph)

approach:

- trellis: MLP-like structure
from left to right with layers $t = 1, .., T$
- forward pass:
DP recursion from left to right
- backward pass: apply chain rule
to DP recursion from right to left



note: no explicit concept of posterior weigths

relation between MLP backpropagation and EM algorithm for HMM:

- **J. Bridle:** ***Alpha-nets: a recurrent neural network architecture with a hidden Markov model interpretation.*** **Speech Communication, pp. 83-92, Vol. 9, 1990.**

  Abstract: ***... A hidden Markov model isolated word recogniser using full likelihood scoring for each word model can be treated as a recurrent 'neural' network ... The back-propagation has exactly the same form as the backward pass of the Baum-Welch (EM) algorithm for maximum-likelihood HMM training ...***

- **J. Bridle, L. Dodd:** ***An Alpha-net approach to optimising input transformations for continuous speech recognition.*** **ICASSP 1991, pp. 277-280.**

  Section 2: ***... When the outputs of the network have the form of a probability distribution (i.e. $Q_w > 0, \sum_w Q_w = 1$), we favour using a relative entropy-based score ...***

  interpretation: 'discriminative training for isolated words'

History: Automatic Differentation

- R. E. Wengert: *A simple automatic derivative evaluation program*. Comm. ACM, Vol. 7, No. 8, pp. 463-464, 1964.
- G. Kedem: *Automatic differentiation of computer programs*. ACM Transactions on Mathematical Software (TOMS), Vol. 6, No. 2, pp. 150-165, June 1980.
- L. B. Rall: *Automatic Differentiation - Techniques and Applications*. Springer Lecture Notes in Computer Science, Vol. 120, 1981.
- A. Griewank: *On Automatic Differentiation*. Argonne National Lab., report, 28 pages, Nov. 1988.
- C. Bischof, A. Carle, G. Corliss, A. Griewank, P. Hovland: *ADIFOR: generating derivative codes from Fortran programs*. Scientific Programming, Vol. 1, No. 1, pp. 11-29, 1991.
- A. Griewank: *An mathematical view of automatic differentation* (overview). Acta Numerica, pp. 1-78, 2003.

remarks:
– automatic differentiation: history independent of backpropagation and machine learning
– key component: chain rule of differentiation (as in backpropagation)

– more details: see Excursion: *Automatic Differentiation in Trellis DP*

computation of gradient:

- **function to be optimized for a set of training sequences $x_n, n = 1, ...N$:**

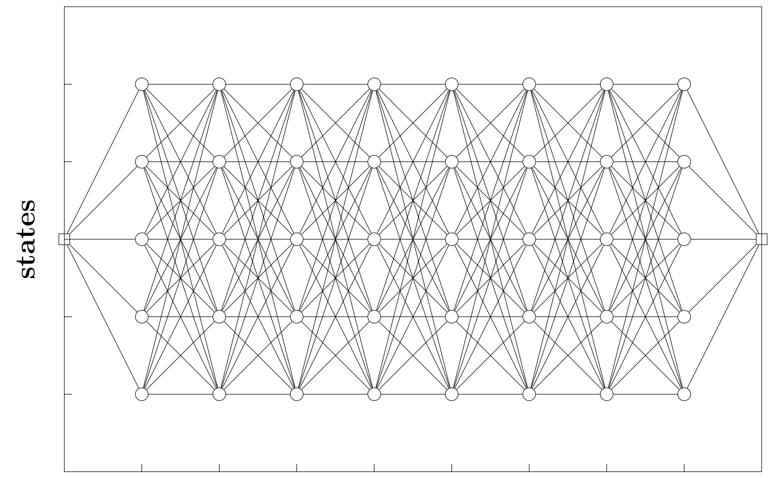
$$\lambda \;\rightarrow\; F(\lambda) \;:=\; \sum_n \log f(x_n, \lambda) = \sum_n \log \sum_{s_1^{T_n}} \prod_{t=1}^{T_n} q_t(s_{t-1}, s_t, x_n; \lambda)$$

- **form of gradient:**
  it is sufficient to consider a single term in the sum over n:

$$\lambda \;\rightarrow\; f(x_n, \lambda) = \sum_{s_1^{T_n}} \prod_{t=1}^{T_n} q_t(s_{t-1}, s_t, x_n; \lambda)$$

- **simplified notation: we drop the explicit symbol for the training sequence x_n in both $f(x_n; \lambda)$ and $q_t(s', s, x_n; \lambda)$ and assume a sequence length T:**

$$\lambda \;\rightarrow\; f(\lambda) = \sum_{s_1^{T}} \prod_{t=1}^{T} q_t(s_{t-1}, s_t; \lambda)$$



forward pass using DP recursion over $t = 1, 2, ..., T$:

$$Q_t(s_t;\lambda) = \sum_{s_{t-1}} Q_{t-1}(s_{t-1};\lambda) \cdot q_t(s_{t-1}, s_t;\lambda)$$

$$t = T: \quad f(\lambda) = \sum_{s_T} Q_T(s_T;\lambda)$$

backward pass using chain rule over $t = T, T-1, ..., 1$:

$$\frac{\partial}{\partial\lambda}\log f(\lambda) = \frac{1}{f(\lambda)} \cdot \sum_{s_T} \frac{\partial}{\partial\lambda} Q_T(s_T;\lambda) = \frac{1}{f(\lambda)} \cdot \sum_{s_T}\sum_{s_{T-1}} \frac{\partial}{\partial\lambda}\Big(\underbrace{Q_{T-1}(s_{T-1};\lambda)}_{\texttt{backward rec.}} \cdot q_T(s_{T-1}, s_T;\lambda)\Big)$$

$$= \frac{1}{f(\lambda)} \cdot \sum_{s_T}\sum_{s_{T-1}} Q_{T-1}(s_{T-1};\lambda) \cdot \frac{\partial\, q_T(s_{T-1}, s_T;\lambda)}{\partial\lambda}$$

$$+ \frac{1}{f(\lambda)} \cdot \sum_{s_T}\sum_{s_{T-1}} q_T(s_{T-1}, s_T;\lambda) \sum_{s_{T-2}} \frac{\partial}{\partial\lambda}\Big(\underbrace{Q_{T-2}(s_{T-2};\lambda)}_{\texttt{backward rec.}} \cdot q_{T-1}(s_{T-2}, s_{T-1};\lambda)\Big)$$

$$= \ldots$$

result: OK for numerical calculations,
but awkward for analytic calculations

continue evaluating the backward recursion:

$$\frac{\partial}{\partial\lambda}\log f(\lambda) =$$

$$= \frac{1}{f(\lambda)}\cdot\sum_{s_T}\sum_{s_{T-1}}\frac{\partial}{\partial\lambda}\Big(\underbrace{Q_{T-1}(s_{T-1};\lambda)}_{\texttt{backward rec.}}\cdot q_T(s_{T-1},s_T;\lambda)\Big)$$

$$= \frac{1}{f(\lambda)}\cdot\sum_{s_T}\sum_{s_{T-1}} Q_{T-1}(s_{T-1};\lambda)\cdot\frac{\partial\, q_T(s_{T-1},s_T;\lambda)}{\partial\lambda}$$

$$+\frac{1}{f(\lambda)}\cdot\sum_{s_T}\sum_{s_{T-1}} q_T(s_{T-1},s_T;\lambda)\sum_{s_{T-2}}\frac{\partial}{\partial\lambda}\Big(\underbrace{Q_{T-2}(s_{T-2};\lambda)}_{\texttt{backward rec.}}\cdot q_{T-1}(s_{T-2},s_{T-1};\lambda)\Big)$$

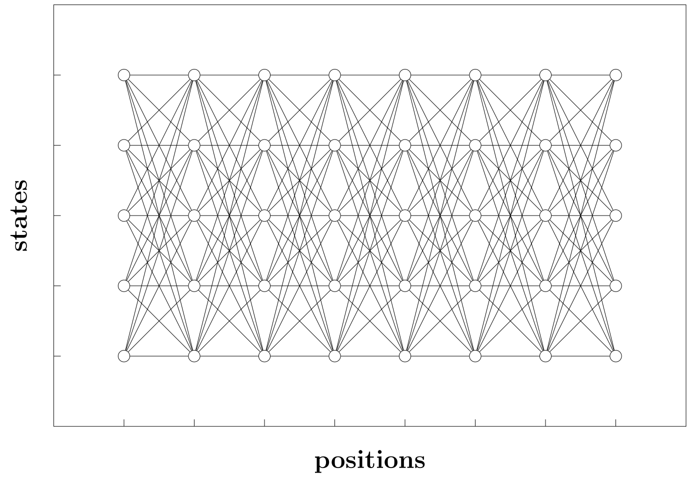
$$= \frac{1}{f(\lambda)}\cdot\sum_{s_T}\sum_{s_{T-1}} Q_{T-1}(s_{T-1};\lambda)\cdot\frac{\partial\, q_T(s_{T-1},s_T;\lambda)}{\partial\lambda}$$

$$+\frac{1}{f(\lambda)}\cdot\sum_{s_T}\sum_{s_{T-1}}\sum_{S_{T-2}} q_T(s_{T-1},s_T;\lambda)\; Q_{T-2}(s_{T-2};\lambda)\cdot\frac{\partial\, q_{T-1}(s_{T-2},s_{T-1};\lambda)}{\partial\lambda}$$

$$+\frac{1}{f(\lambda)}\cdot\sum_{s_T}\sum_{s_{T-1}}\sum_{s_{T-2}} q_T(s_{T-1},s_T;\lambda)\; q_{T-1}(s_{T-2},s_{T-1}T;\lambda)\sum_{S_{T-3}}\frac{\partial}{\partial\lambda}\Big(\underbrace{Q_{T-3}(s_{T-3};\lambda)}_{\texttt{backward rec.}}\cdot q_{T-2}(s_{T-3},s_{T-2};\lambda)\Big)$$

$$= \ldots$$

interpretation: can be re-written in terms of forward–backward probabilities



- goal: show equivalence analytically between
  – EM-style backpropagation
  – backpropagation applied to DP recursion

- we consider the dependence of the global score $f(\lambda)$ on the local score of each arc (t, s', s):

$$\lambda \;\rightarrow\; \{q_t(s', s; \lambda) : s', s, t\} \;\rightarrow\; f(\lambda) := \sum_{s_1^T} \prod_{t=1}^{T} q_t(s_{t-1}, s_t; \lambda)$$

- we consider the global score $f(\lambda)$ and compute the derivative:

$$\frac{\partial}{\partial\lambda} \log\, f(\lambda) \;=\; \frac{1}{f(\lambda)} \cdot \frac{\partial f(\lambda)}{\partial\lambda} \;=\; \frac{1}{f(\lambda)} \cdot \sum_t \sum_{s',s} \frac{\partial f(\lambda)}{\partial q_t(s', s; \lambda)} \cdot \frac{\partial q_t(s', s; \lambda)}{\partial\lambda}$$

$$= \frac{1}{f(\lambda)} \cdot \sum_t \sum_{s',s} \frac{\partial f(\lambda)}{\partial q_t(s', s; \lambda)} \cdot q_t(s', s; \lambda) \cdot \frac{\partial \log\, q_t(s', s; \lambda)}{\partial\lambda}$$

$$\frac{\partial f(\lambda)}{\partial q_\tau(s', s; \lambda)} \;=\; \frac{\partial}{\partial q_\tau(s', s; \lambda)} \sum_{s_T} Q_T(s_T, \lambda) \;=\; \sum_{s_T} \frac{\partial Q_T(s_T; \lambda)}{\partial q_\tau(s', s; \lambda)}$$

Backpropagation and Trellis Dynamic Programming: Equivalence with EM-Style Backpropagation

- **backward pass: starting point at $t = T$:**

$$\frac{\partial f(\lambda)}{\partial q_\tau(s', s; \lambda)} = \frac{\partial}{\partial q_\tau(s', s; \lambda)} \sum_{s_T} Q_T(s_T, \lambda) = \sum_{s_T} \frac{\partial Q_T(s_T; \lambda)}{\partial q_\tau(s', s; \lambda)}$$

- **we need the derivatives of the DP auxiliary function $Q_t(s_t; \lambda)$ wrt $q_\tau(s', s; \lambda)$ in the forward DP recursion over $t = 1, ..., T$:**

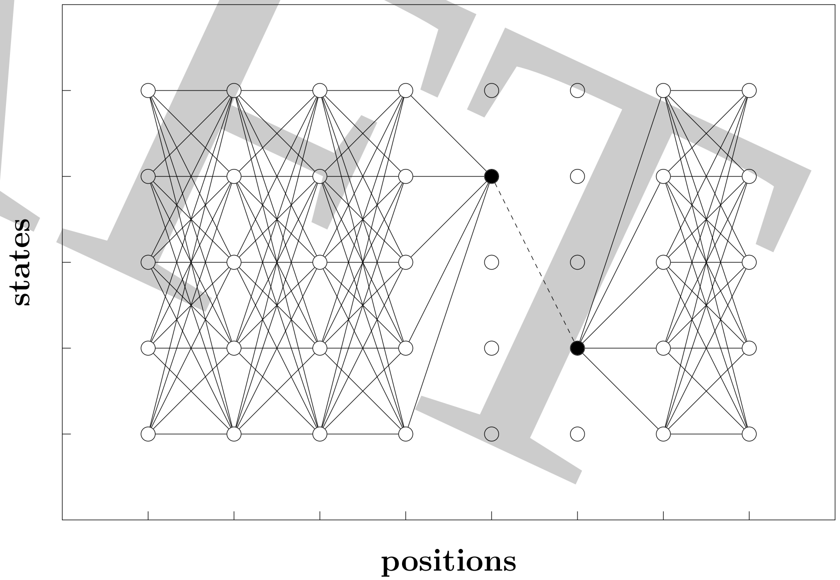
$$Q_t(s_t; \lambda) = \sum_{s_{t-1}} Q_{t-1}(s_{t-1}; \lambda) \cdot q_t(s_{t-1}, s_t; \lambda)$$

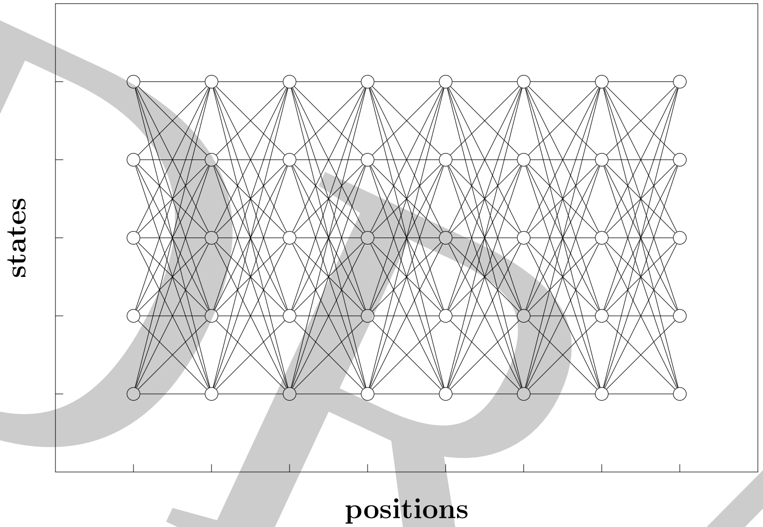
- **to exploit the DP recursion in backpropagation, we fix a grid point (t, s_t) with DP score $Q_t(s_t; \lambda)$ and an arc (τ, s', s) with model $q_\tau(s', s; \lambda)$:**

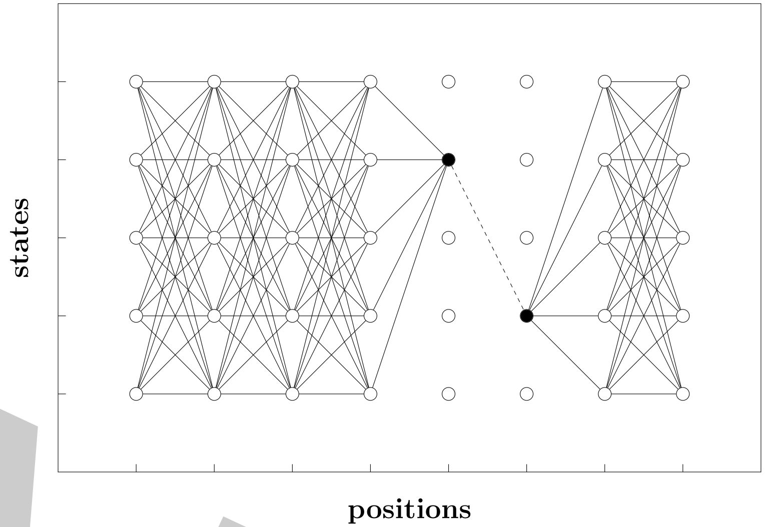
$$q_\tau(s', s; \lambda) \rightarrow Q_t(s_t; \lambda)$$

$$\frac{\partial Q_t(s_t; \lambda)}{\partial q_\tau(s', s; \lambda)} = \;?$$

note the five-tuple of indices: $(t, s_t;\; \tau, s', s)$







DP recursion and its derivative:

$$Q_t(s_t;\lambda) = \sum_{s_{t-1}} q_t(s_{t-1},s_t;\lambda)\cdot Q_{t-1}(s_{t-1};\lambda)$$

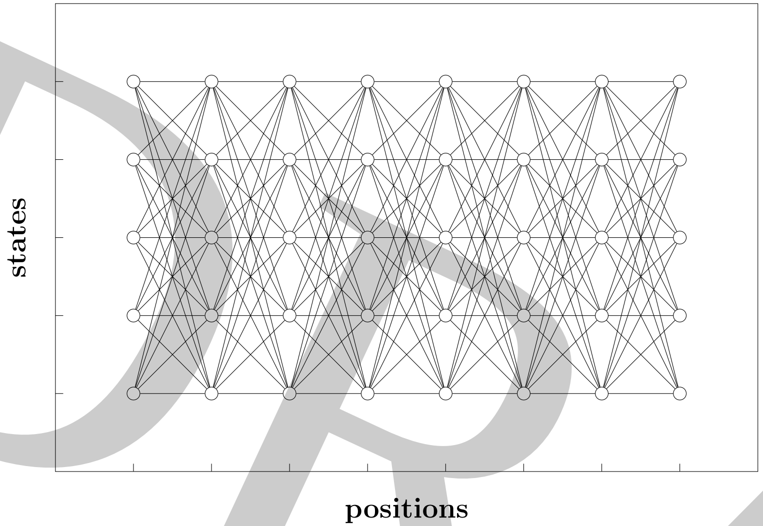
$$\frac{\partial Q_t(s_t;\lambda)}{\partial q_\tau(s',s;\lambda)} = \sum_{s_{t-1}} \Big(q_t(s_{t-1},s_t;\lambda)\cdot \frac{\partial Q_{t-1}(s_{t-1};\lambda)}{\partial q_\tau(s',s;\lambda)} + \frac{\partial q_t(s_{t-1},s_t;\lambda)}{\partial q_\tau(s',s;\lambda)}\cdot Q_{t-1}(s_{t-1};\lambda)\Big)$$

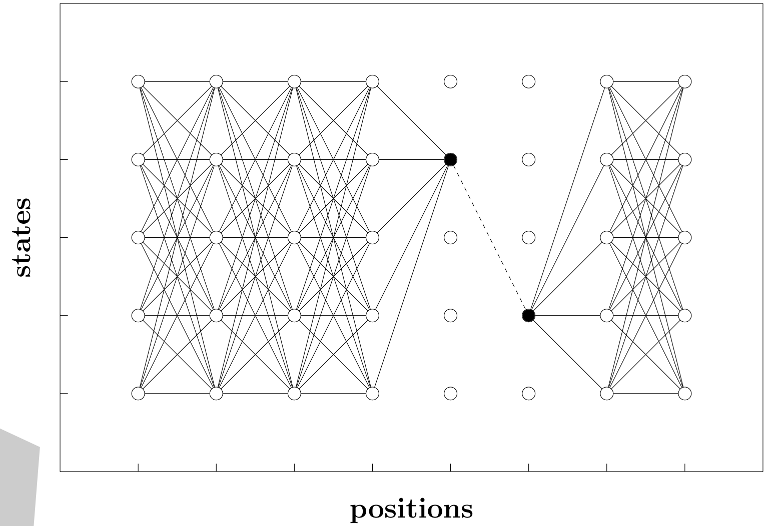
analysis of derivative:

– **case** $t<\tau$: **no dependence of** $Q_t(s_t;\lambda)$ **on** $q_\tau(s',s;\lambda)$

– **term** $\partial q_t(s_{t-1},s_t;\lambda)/\partial q_\tau(s',s;\lambda)$: **different from zero only for** $\tau=t$

– **term** $\partial Q_{t-1}(s_{t-1};\lambda)/\partial q_\tau(s',s;\lambda)$: **causes the recursion in the derivatives**





- **case $\tau < t$: recursion over $t = T, T-1, ..., \tau+1$**

$$\frac{\partial Q_t(s_t;\lambda)}{\partial q_\tau(s',s;\lambda)} = \sum_{s_{t-1}} q_t(s_{t-1},s_t;\lambda) \cdot \frac{\partial Q_{t-1}(s_{t-1};\lambda)}{\partial q_\tau(s',s;\lambda)}$$

- **case $\tau = t$:**

$$\begin{aligned}\frac{\partial Q_\tau(s_\tau;\lambda)}{\partial q_\tau(s',s;\lambda)} &= \sum_{s_{\tau-1}} \frac{\partial q_\tau(s_{\tau-1},s_\tau;\lambda)}{\partial q_\tau(s',s;\lambda)} \cdot Q_{\tau-1}(s_{\tau-1};\lambda) \\ &= \delta(s_\tau,s) \cdot Q_{\tau-1}(s';\lambda)\end{aligned}$$

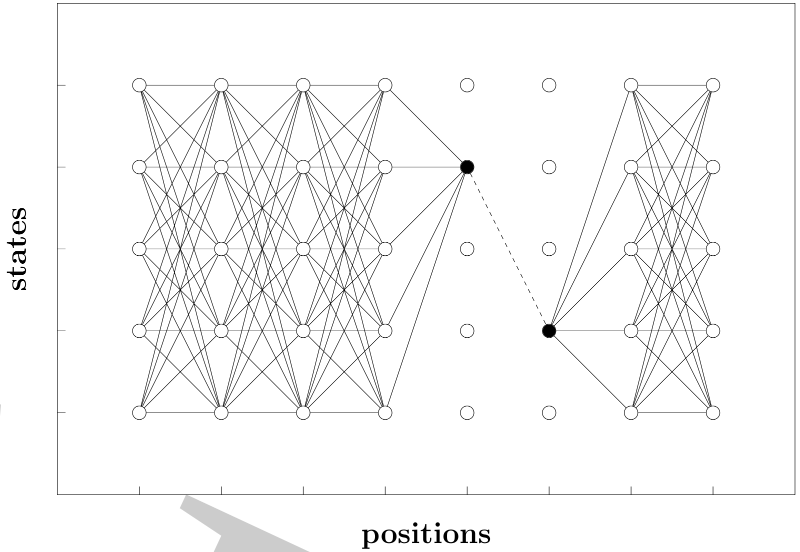
- **case $\tau > t$:**

$$\frac{\partial Q_t(s_t;\lambda)}{\partial q_\tau(s',s;\lambda)} = 0$$

backward recursion from $t = T, T-1, ..., \tau$:

$$\frac{\partial f(\lambda)}{\partial q_\tau(s', s; \lambda)} = \sum_{s_T} \frac{\partial Q_T(s_T; \lambda)}{\partial q_\tau(s', s; \lambda)} =$$

$$= \sum_{s_T} \sum_{s_{T-1}} q_T(s_{T-1}, s_T; \lambda) \cdot \frac{\partial Q_{T-1}(s_{T-1}; \lambda)}{\partial q_\tau(s', s; \lambda)}$$



$$= \sum_{s_T} \sum_{s_{T-1}} q_T(s_{T-1}, s_T; \lambda) \sum_{s_{T-2}} q_{T-1}(s_{T-2}, s_{T-1}; \lambda) \cdot \frac{\partial Q_{T-2}(s_{T-2}; \lambda)}{\partial q_\tau(s', s; \lambda)}$$

$$= ...$$

$$= \sum_{s_T} \sum_{s_{T-1}} q_T(s_{T-1}, s_T; \lambda) \sum_{s_{T-2}} q_{T-1}(s_{T-2}, s_{T-1}; \lambda) \, ... \sum_{s_{\tau+1}} ... \sum_{s_\tau} q_\tau(s_\tau, s_{\tau+1}; \lambda) \cdot \frac{\partial Q_\tau(s_\tau; \lambda)}{\partial q_\tau(s', s; \lambda)}$$

$$= \sum_{s_T} \sum_{s_{T-1}} q_T(s_{T-1}, s_T; \lambda) \sum_{s_{T-2}} q_{T-1}(s_{T-2}, s_{T-1}; \lambda) \, ... \sum_{s_{\tau+1}} ... \sum_{s_\tau} q_\tau(s_\tau, s_{\tau+1}; \lambda) \cdot \delta(s_\tau, s) \cdot Q_{\tau-1}(s'; \lambda)$$

$$= \sum_{s_T} \sum_{s_{T-1}} q_T(s_{T-1}, s_T; \lambda) \sum_{s_{T-2}} q_{T-1}(s_{T-2}, s_{T-1}; \lambda) \, ... \sum_{s_{\tau+1}} ... \; q_\tau(s, s_{\tau+1}; \lambda) \cdot Q_{\tau-1}(s'; \lambda)$$

$$= Q_{\tau-1}(s'; \lambda) \cdot \sum_{s_\tau^T : s_\tau = s} \prod_{t=\tau+1}^{T} q_t(s_{t-1}, s_t; \lambda) = \textbf{(literature)} \quad \alpha(t-1, s') \cdot \beta(t, s)$$

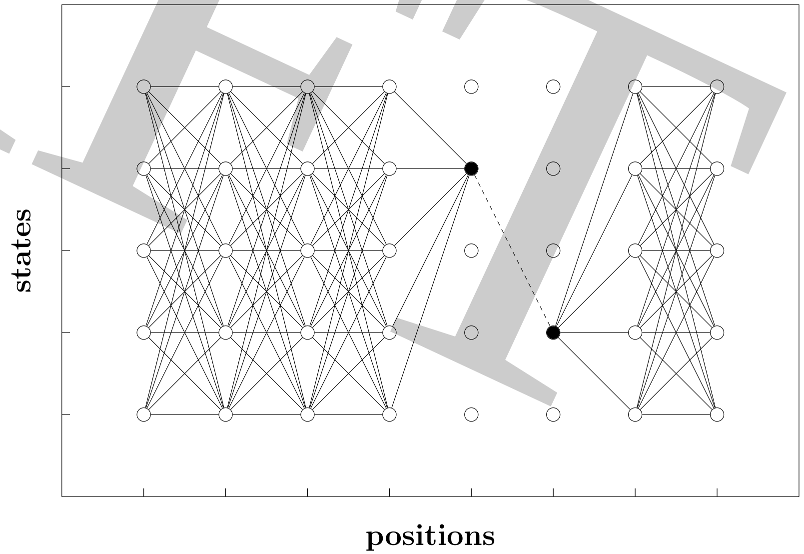
we have shown:

$$\frac{\partial f(\lambda)}{\partial q_t(s',s;\lambda)} = \underbrace{Q_{t-1}(s';\lambda)}_{\textit{forward pass: } \alpha(t-1,s')} \cdot \underbrace{\sum_{s_t^T : s_t = s} \prod_{\tau=t+1}^{T} q_\tau(s_{\tau-1}, s_\tau; \lambda)}_{\textit{backward pass: } \beta(t,s)}$$

$$\frac{\partial}{\partial \lambda} \log f(\lambda) = \frac{1}{f(\lambda)} \cdot \sum_t \sum_{s',s} \frac{\partial f(\lambda)}{\partial q_t(s',s;\lambda)} \cdot q_t(s',s;\lambda) \cdot \frac{\partial \log q_t(s',s;\lambda)}{\partial \lambda}$$

$$= \sum_t \sum_{s',s} w_t(s',s|\lambda) \cdot \frac{\partial \log q_t(s',s;\lambda)}{\partial \lambda}$$

interpretation:
– normalization by $f(\lambda)$
– equivalent to forward–backward probabilities



7 Cross-Attention for Synchronization

cross-attention approach:

- **originally introduced for MT:**
  D. Bahdanau, K. Cho, Y. Bengio:
  Neural machine translation by jointly learning to align and translate.
  ICLR, 2015.
- **turned out to be very succesful also for ASR**
- **the method has not changed much since 2015,**
  some extensions like for ASR: additional CTC loss in training
- **most important extension:**
  transformer approach [Vaswani & Shazeer^{+} 17].

7.1 Principle using RNN

ASR: from speech signal x_1^T to word sequence a_1^I
(other output: letters or BPE units)

key concepts for modelling posterior probability $p(a_1^I|x_1^J = T)$

- **direct approach: use unidirectional RNN over target positions $i = 1, ..., I$ with internal state vector s_i:**

$$p(a_1^I|x_1^T) = \prod_i p(a_i|a_0^{i-1}, x_1^J) = \prod_i p(a_i|a_{i-1}, s_{i-1}, x_1^T)$$

  interpretation: extended language model for target word sequence

- **additional component: attention mechanism for localization**

$$p(a_i|a_{i-1}, s_{i-1}, x_1^T) = p(a_i|a_{i-1}, s_{i-1}, c_i)$$

  with a context vector: $c_i := C(s_{i-1}, x_1^T)$

word embeddings and representations:

- **word embedding for word sequence:**
  – word symbol: a_i
  – word vector: $\tilde{a}_i = R(a_i)$ with the embedding (matrix) R
- **representation vector h_t for speech signal using a bidirectional RNN:** $h_t = H_t(x_1^T)$

final model and its terminology:

$$p(a_1^I|x_1^T) = \prod_i p(a_i|\tilde{a}_{i-1}, s_{i-1}, c_i) \qquad c = C(s_{i-1}, h_1^T(x_1^T))$$

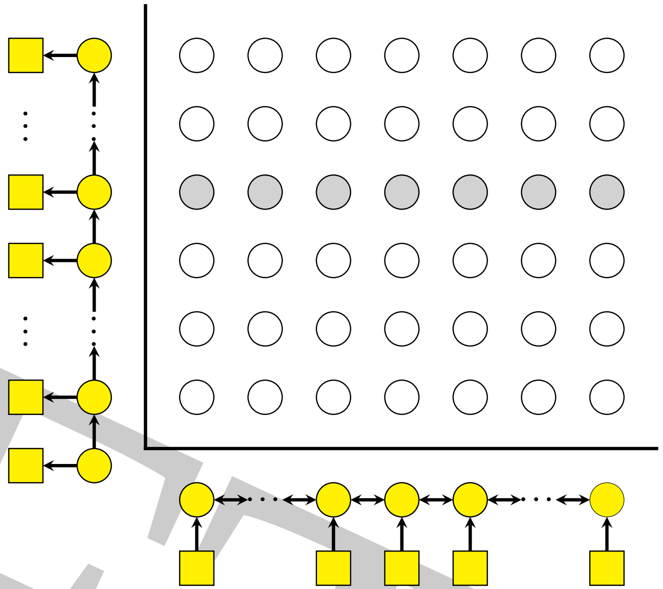
– **encoder:** $h_t = H_t(x_1^T)$
– **decoder: (full) output model** $p(a_i|...)$

Attention-based ASR
[Bahdanau & Cho+ 15]

approach:

- **input: bidirectional RNN over source positions j:**
$$(x_1^T, t) \rightarrow h_t = H_t(x_1^T)$$

- **output: unidirectional RNN over target positions i:**
$$y_i = Y(y_{i-1}, s_{i-1}, c_i)$$
conventional notation:
$$p(a_i|\tilde{a}_{i-1}, s_{i-1}, c_i)$$
with RNN state vector $s_i = S(s_{i-1}, \tilde{e}_i, c_i)$
and context vector $c_i = C(s_{i-1}, h_1^T)$



- **context vector c_i: weighted average of input vector representations:**

$$c_i = \sum_t \alpha(t|i, s_{i-1}, h_1^T) \cdot h_t$$
$$\alpha(t|i, s_{i-1}, h_1^J) = \frac{\exp(A[s_{i-1}, h_t])}{\sum_{t'} \exp(A[s_{i-1}, h_{t'}])}$$

with the normalized attention weights $\alpha(t|i, s_{i-1}, h_1^T)$
and real-valued attention scores $A[s_{i-1}, h_t]$

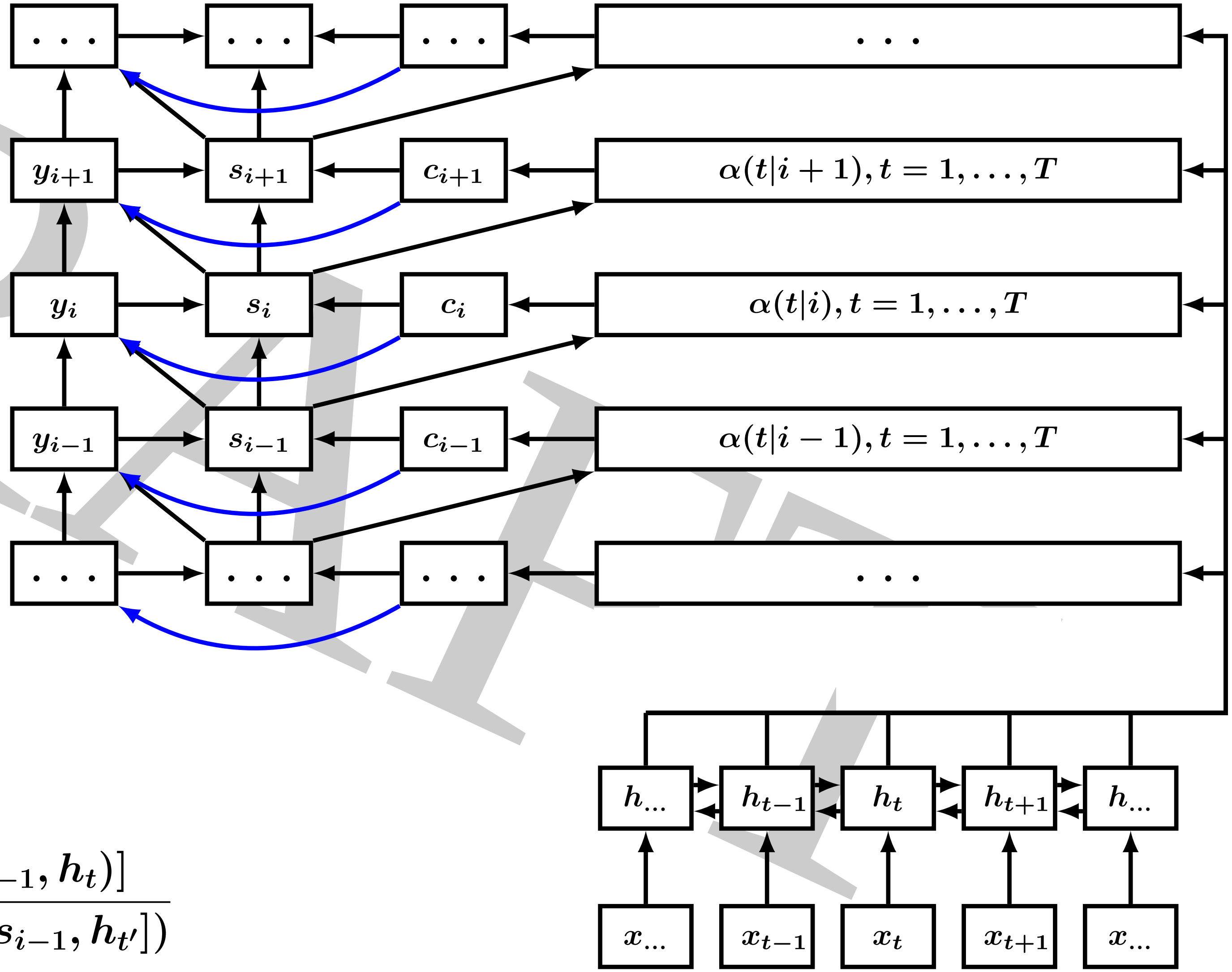
Attention-based ASR:

$$x_1^T \rightarrow a_1^I$$

principle:

- **input: source sequence:**
  $(x_1^T, t) \rightarrow h_t = H_t(x_1^T)$
- **output distribution:**
  $y_i \equiv p_i(a|\tilde{a}_{i-1}, s_{i-1}, c_i)$
  notation in ANN style:
  $y_i = Y(y_{i-1}, s_{i-1}, c_i)$
- **state vector of target RNN:**
  $s_i = S(s_{i-1}, y_i, c_i)$
- **weighted context vector:**
  $c_i = \sum_t \alpha(t|i, s_{i-1}, h_1^T) \cdot h_t$
- **attention weights:**

$$\alpha(t|i, s_{i-1}, h_1^T) = \frac{\exp(A[s_{i-1}, h_t)]}{\sum_{t'} \exp(A[s_{i-1}, h_{t'}])}$$



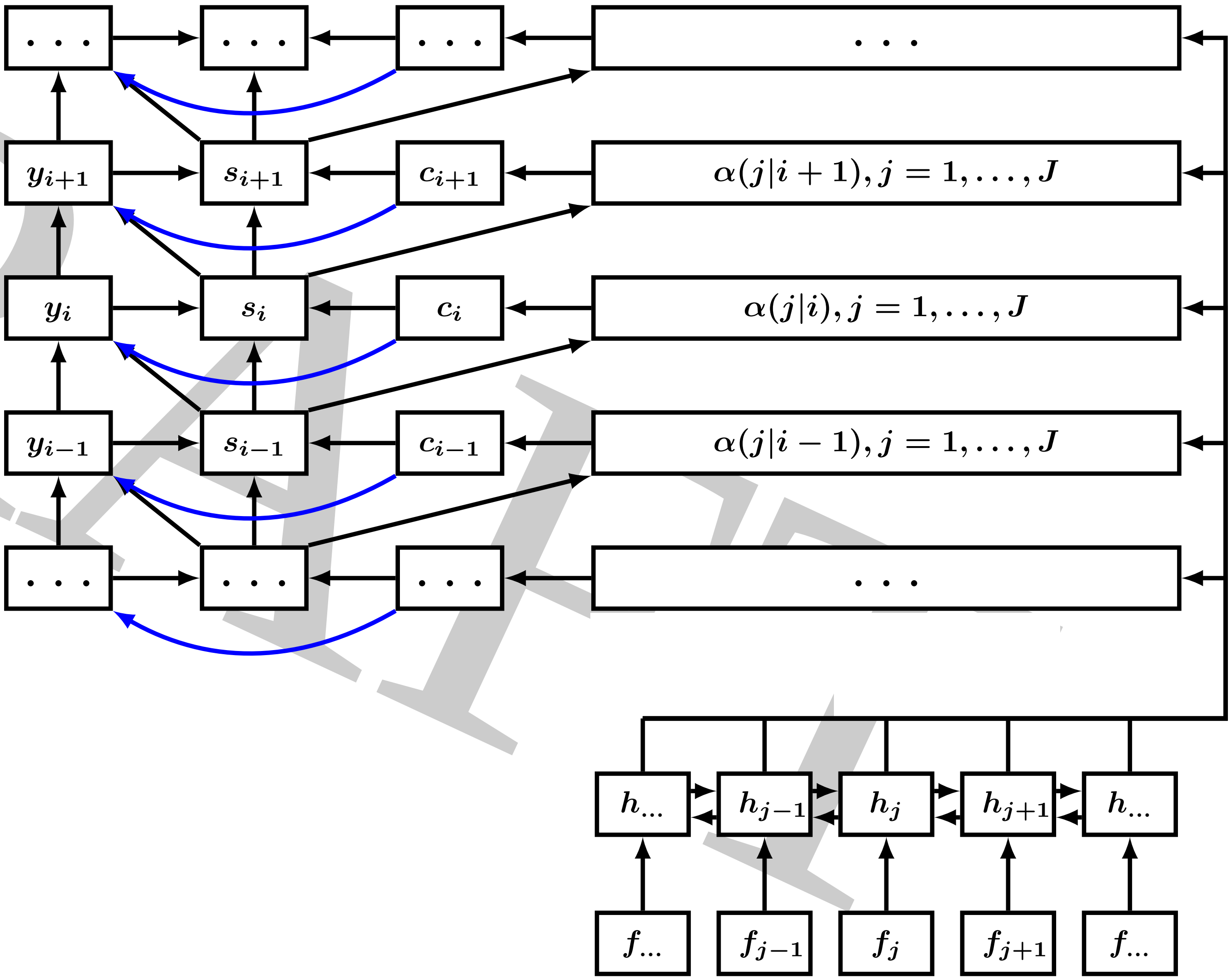
Attention-based ASR: Sequential Order of Operations

preparations:

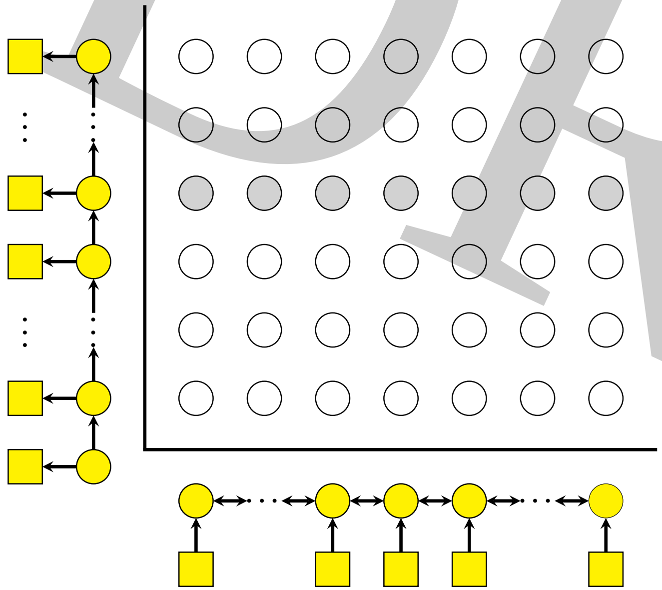
- **input preprocessing:**
  $x_1^T \rightarrow h_t = H_t(x_1^T)$
- **available at position $i-1$:**
  $\tilde{a}_{i-1} \equiv y_{i-1},\ s_{i-1},\ c_{i-1}$

sequence of operations for position i:

1. **attention weights:**
   $\alpha(t|i, s_{i-1}, h_1^T) = ...$
2. **context vector:**
   $c_i = \sum_t \alpha(t|i, s_{i-1}, h_1^T) \cdot h_t$
3. **output distribution:**
   $y_i = Y(y_{i-1}, s_{i-1}, c_i)$
4. **state vector:**
   $s_i = S(s_{i-1}, y_i, c_i)$



common properties:
– input: acoustic encoder: representation/state vectors $h_t = h_t(x_1^T), t = 1, ..., T$
– output: (phoneme) labels $a_s,\ s = 1, ..., S$ with/without integrated language model



- (cross-) attention: direct factorization:

$$p(a_1^S|x_1^T) = \prod_s p(a_s|a_0^{s-1}, x_1^T) = \prod_s p(a_s|a_{s-1}, r_{s-1}, c_s)$$

$$c_s := \sum_t p(t|a_0^{s-1}, x_1^T) \cdot h_t$$

with context vector c_s and output state vector r_s

criticism for ASR: lack of strict monotonicity and localization

- finite-state transducer (direct HMM, CTC, RNN-T, ...): introduce hidden paths and then factorize:

$$p(a_1^S|x_1^T) = \sum_{s_1^T} p\Big(s_1^T, a_1^S|h_1^T(x_1^T)\Big)$$
$$= \sum_{s_1^T} \prod_t p\Big(s_{t+1}, y_t = a_{s_t}\Big| s_t, a_0^{s_t-1}, h_1^T(x_1^T)\Big)$$

details: RWTH papers at ICASSP and Interspeech

representation/state vectors h_t:
– deep MLP: finite window
– RNN and LSTM-RNN
– self-attention (transformer)

similar: output string

History: Models for Synchronization

concept of first-order models:

- generative HMM: around 1975
- MMI criterion for HMM: Bahl et al. 1986
  now called: sequence discriminative training
- hybrid HMM: Bourlard & Wellekens 1989
  emission model: use ANN output
- unified concept for hybrid HMM: Haffner 1993
  label-sequence posterior probabilities, sum criterion and training
- hybrid HMM with phoneme RNN models: Robinson 1994
  competitive results on WSJ
- CTC: empty symbol: Graves 2006
- RNN-T (transducer): extension of CTC: Graves 2012
- RNN-A (aligner): special case of RNN-T: Sak et al. 2017
- direct HMM: Raissi et al. 2021 at RWTH
  phoneme labels (instead of CART) and training from scratch

other concept: attention mechanism 2015

change of notation: from ASR to MT

– **ASR:** $x_1^T \rightarrow a_1^I$
– **MT:** $f_1^J \rightarrow e_1^I$

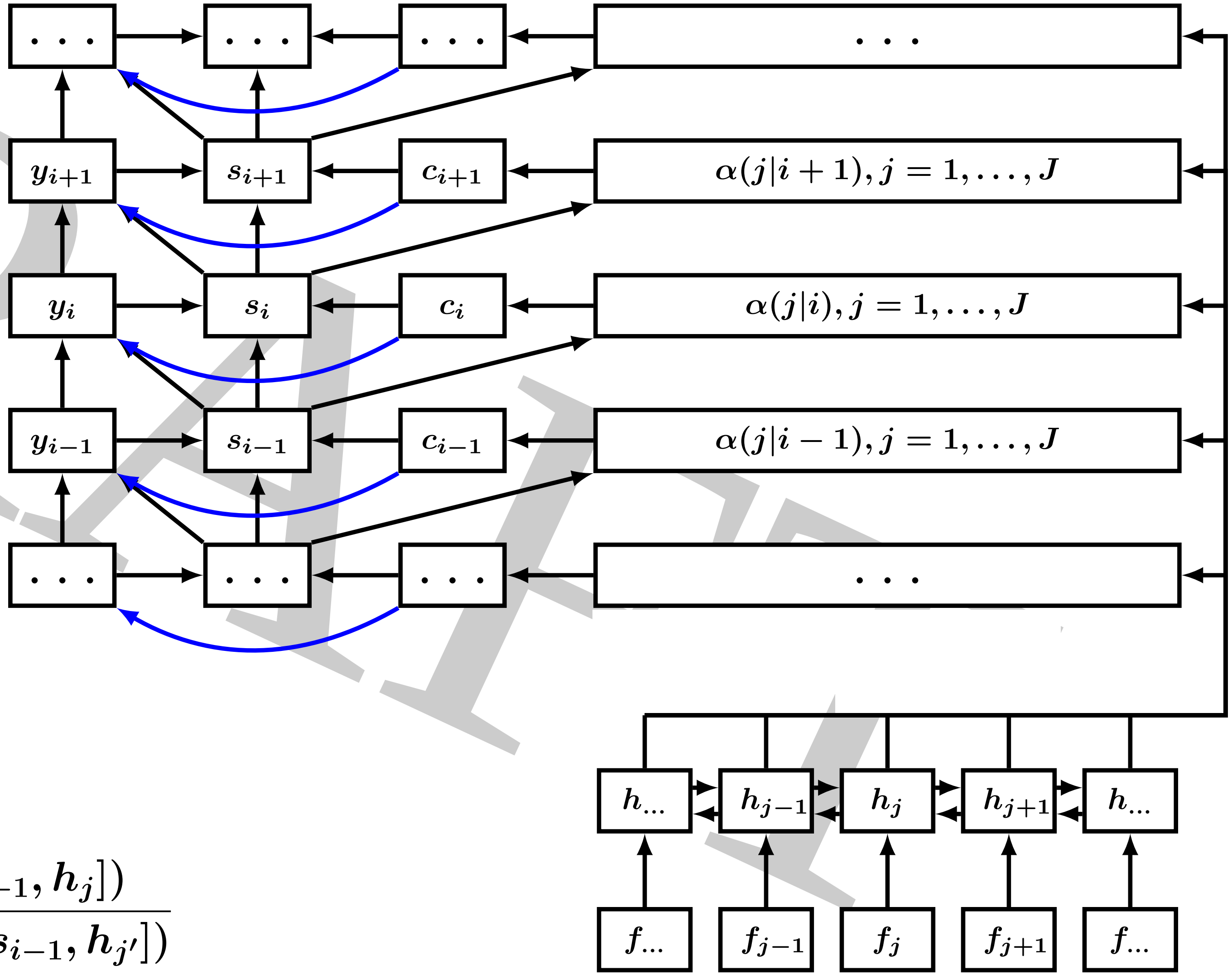
Attention-based Neural MT
[Bahdanau & Cho+ 15]

principle:

- **input: source sequence:**
  $(f_1^J, j) \rightarrow h_j = H_j(f_1^J)$
- **output distribution:**
  $y_i \equiv p_i(e|\tilde{e}_{i-1}, s_{i-1}, c_i)$
  notation in ANN style:
  $y_i = Y(y_{i-1}, s_{i-1}, c_i)$

- **state vector of target RNN:**
  $s_i = S(s_{i-1}, y_i, c_i)$
- **weighted context vector:**
  $c_i = \sum_j \alpha(j|i, s_{i-1}, h_1^J) \cdot h_j$
- **attention weights:**

$$\alpha(j|i, s_{i-1}, h_1^J) = \frac{\exp(A[s_{i-1}, h_j])}{\sum_{j'} \exp(A[s_{i-1}, h_{j'}])}$$



Attention Weights
Feedforward ANN vs. Dot Product

re-consider attention weights:

$$\alpha(j|i, s_{i-1}, h_1^J) = \frac{\exp(A[s_{i-1}, h_j])}{\sum_{j'} \exp(A[s_{i-1}, h_{j'}])}$$

two approaches to modelling attention scores $A[s_{i-1}, h_j]$:

- **additive variant: feedforward (FF) ANN:**

  $$A[s_{i-1}, h_j] := v^T \cdot \tanh(S s_{i-1} + H h_j)$$

  with matrices S and H and vector v
  basic implementation: one FF layer + softmax

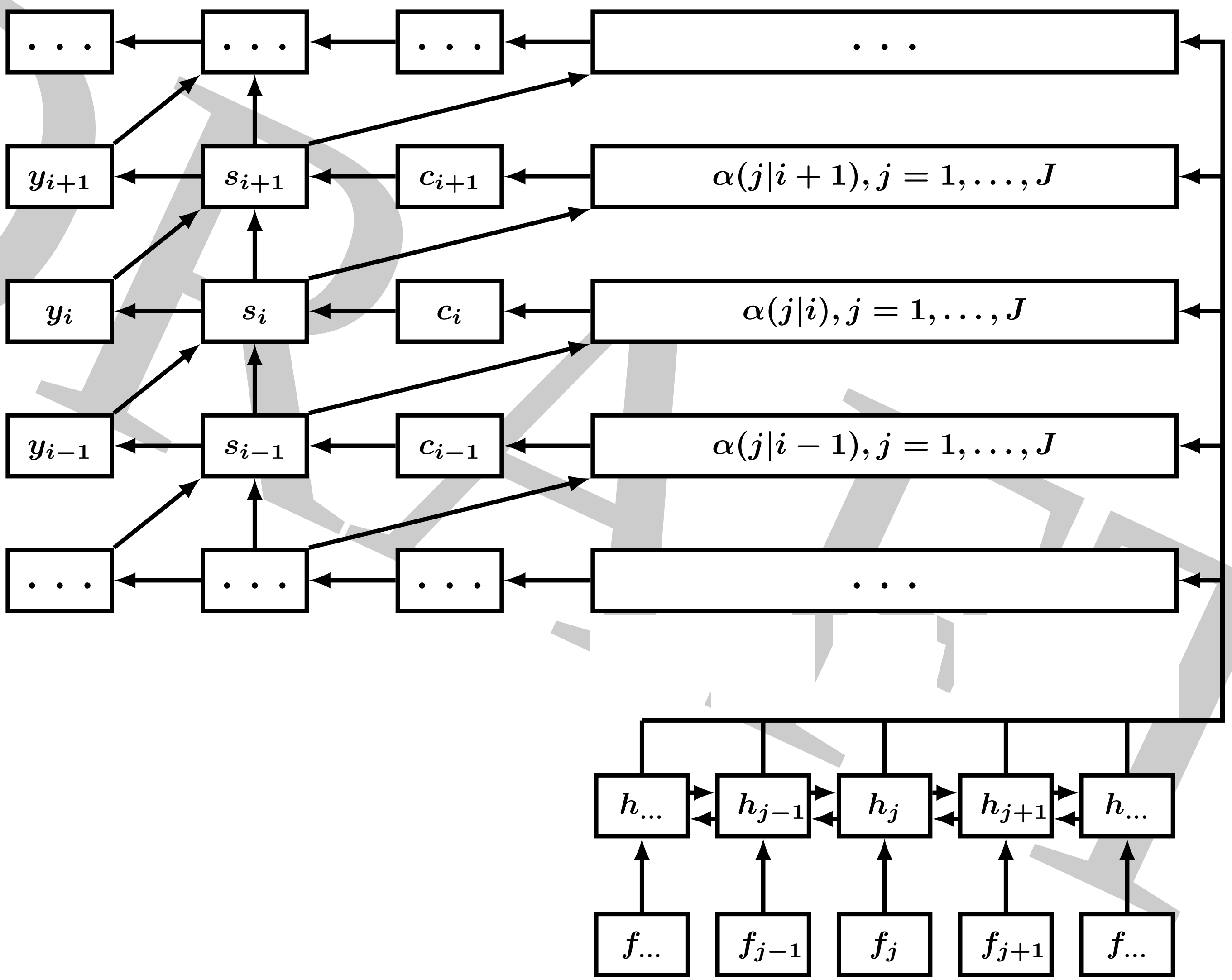
- **multiplicative variant: (generalized) dot product between vectors:**

  $$A[s_{i-1}, h_j] := s_{i-1}^T \cdot W \cdot h_j$$

  with a attention matrix W

experimental result: not much difference (for MT)

Attention-based Neural MT: Variant
[Bahdanau & Cho+ 15]



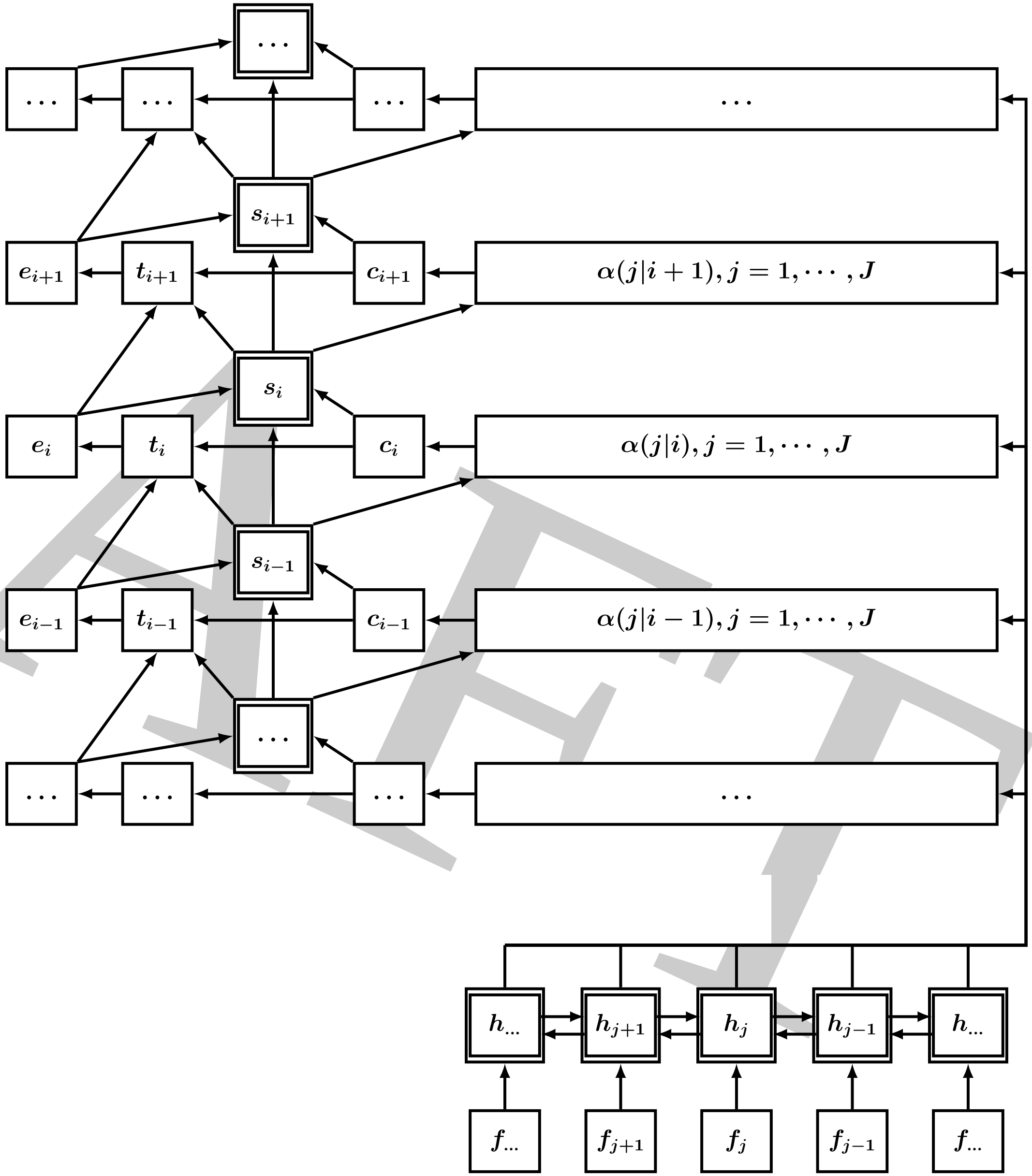
Attention-based Neural MT: Refinements [Bahdanau & Cho+ 15]

refinement:

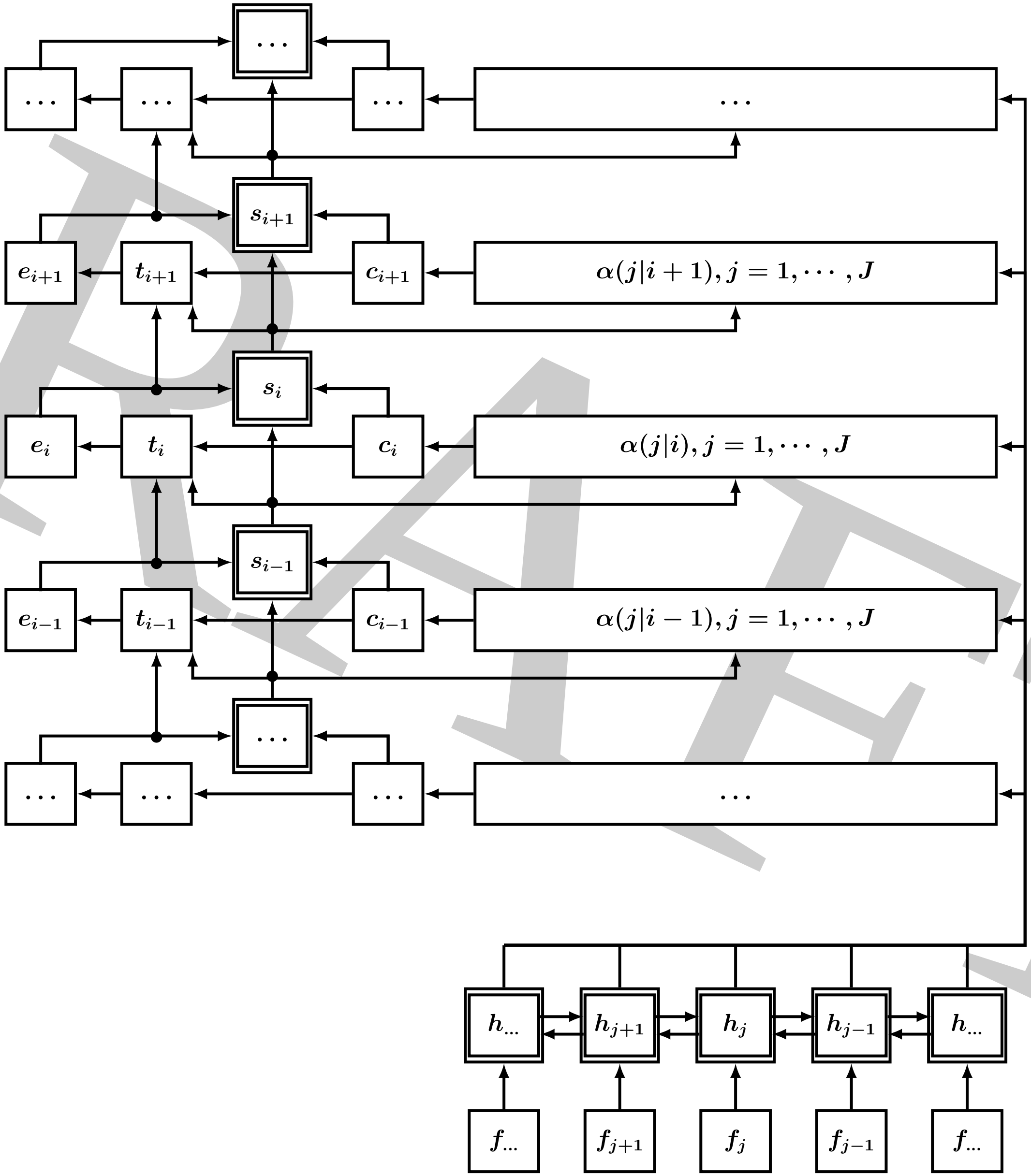
$$y_i = Y(y_{i-1}, s_{i-1}, c_i)$$

is replaced by:

$$t_i = Y(y_{i-1}, s_{i-1}, c_i)$$
$$y_i = Y(t_i)$$



Attention-based Neural MT: Refinements [Bahdanau & Cho$^+$ 15]



Summary: ASR and MT using Attention

full approach requires three ingredients:

- probabilistic model structure:
  how to compute the score $p(e_1^I|f_1^J)$ for a given pair (e_1^I, f_1^J)?
- training criterion:
  cross-entropy, typcally for ASR with additioncal CTC loss
  (which favors monotonicty)

  model with (million of) parameters θ
  along with (numerical optimization) strategy (standard backpropagation)
- search or generation: Bayes decision rule:
  open question: How to organize the search?

Unknown Target String: Length Normalization

starting point: Bayes decision rule:

$$f_1^J \to \hat{e}_1^{\hat{I}}(f_1^J) \;=\; \operatorname*{argmax}_{I, e_1^I : e_I = \$} p(e_1^I | f_1^J) \;=\; \operatorname*{argmax}_{I, e_1^I : e_I = \$} \left\{ \prod_i p(e_i | e_0^{i-1}, f_1^J) \right\}$$

with sentence end symbol \$

problem:
– in practice: approximative models only
– probability scores $p(e_1^I | f_1^J)$ decrease with increasing target length I

practical remedy: length normalization:

$$f_1^J \to \hat{e}_1^{\hat{I}}(f_1^J) \;=\; \operatorname*{argmax}_{I, e_1^I : e_I = \$} \sqrt[I]{p(e_1^I | f_1^J)} \;=\; \operatorname*{argmax}_{I, e_1^I : e_I = \$} \left\{ \sqrt[I]{\prod_i p(e_i | e_0^{i-1}, f_1^J)} \right\}$$

experiments: remedy works sufficiently well

Unknown Target String: Search

auxilary quantity for each unknown partial string e_1^i:

$$Q(i; e_1^i) := \prod_{i'=1}^{i} p(e_{i'} | e_0^{i'-1}, f_1^J)$$

search: extending partial hypothesis from e_1^{i-1} to e_1^i:

$$Q(i; e_1^i) \;=\; p(e_i | e_0^{i-1}, f_1^J) \cdot Q(i-1, ; e_1^{i-1})$$

final result:

$$p(e_1^I | f_1^J) \;=\; Q(I; e_1^I)$$

search organization:
search tree of partial hypotheses e_1^i,
synchronous with target positions i

Unknown Target String: Illustration of Beam Search and Tree Organization

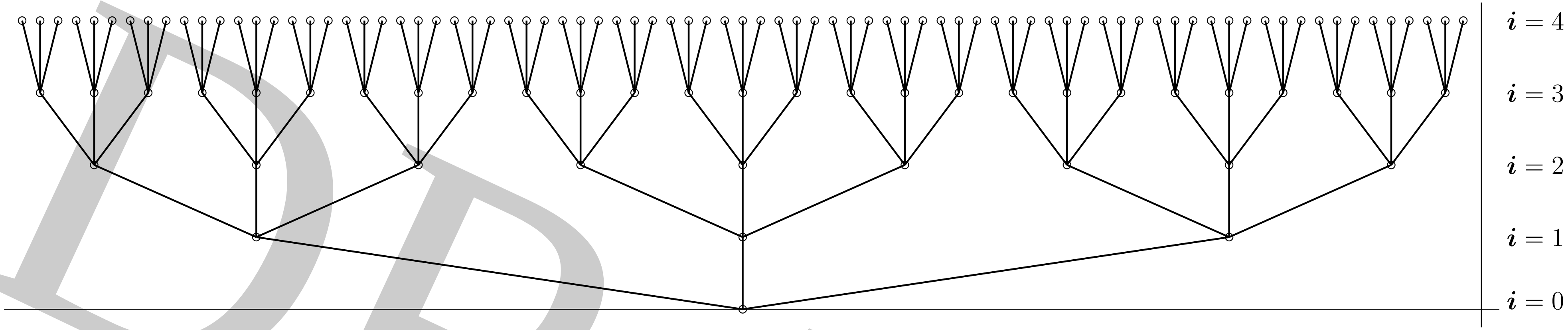
organization:
– tree organization of partial hypotheses e_1^i
– a beam (= set) of partial hypotheses e_1^i with scores $p(e_i|e_0^{i-1}, f_1^J)$
– approximation: beam search using tree representation

$$Q_0(i) := \max_{e_1^i} Q(i; e_1^i)$$

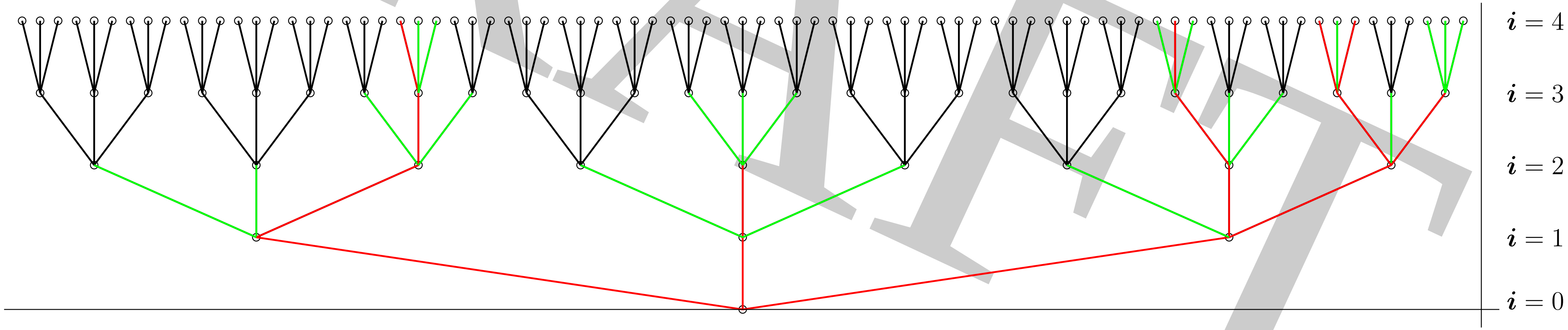
retain hypotheses e_1^i 'close' to $Q_0(i)$

remarks:
– no concept of fertility or coverage constraint
– no recombination (as in other cases of beam search)

Tree Search: Illustration for a Three-Word Vocabulary



full search: keep all hypotheses



beam search (with beam width = 4):
– black arcs: possible arcs
– green arcs: extended and discarded
– red arcs: extended and preserved

7.2 Self-Attention and Transformer

from attention approach to
Google's transformer approach [Vaswani & Shazeer^{+} 17]:
– the attention mechanism between source and target sentence is preserved
– many additional modifications, e. g. self-attention

several concepts:

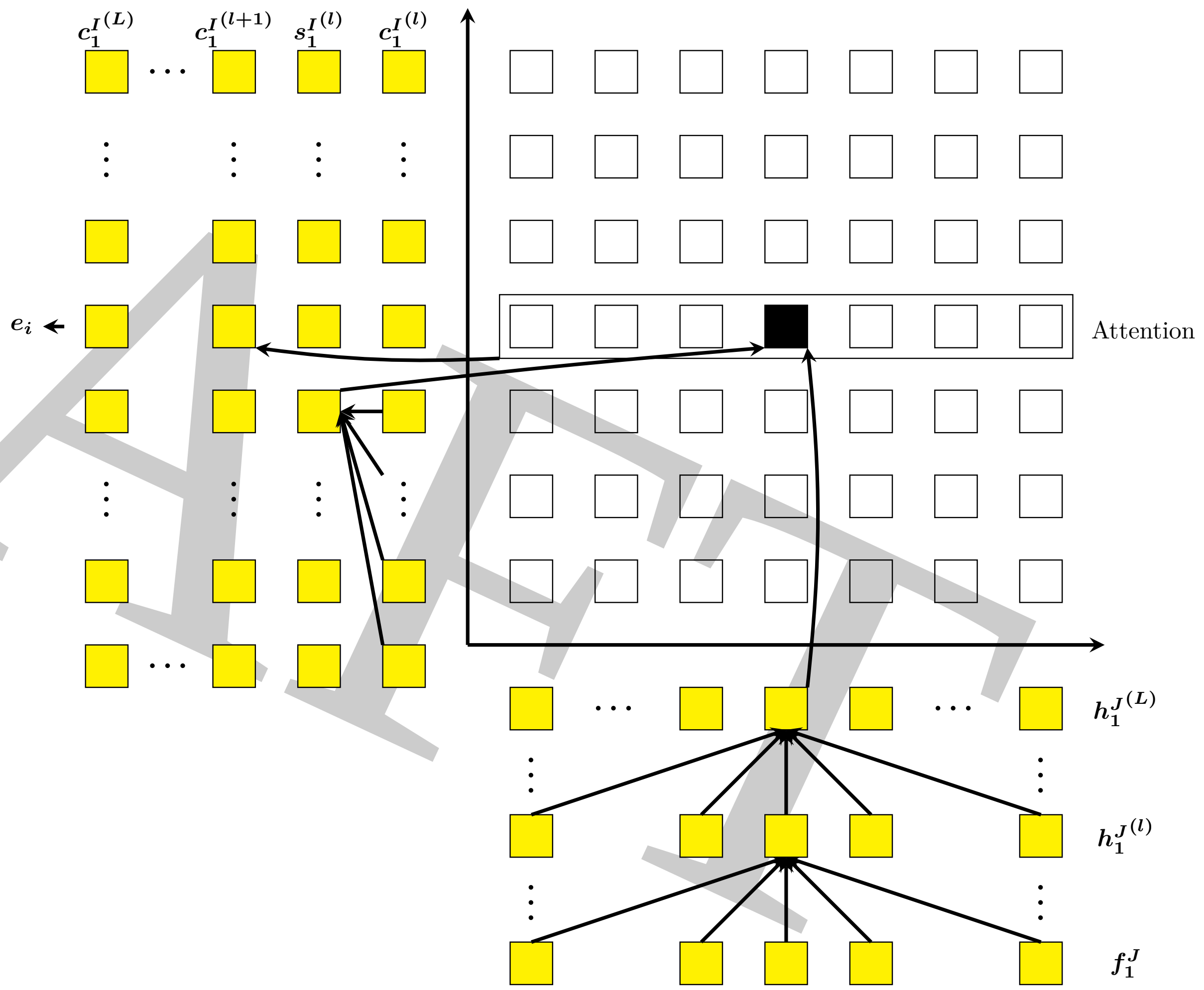
- ***self-attention:*** **also called** ***intra-sentence attention*****:**
  – ***direct*** **interaction between 'all' pairs of positions**
  – replaces the RNN mechanism
  – source side: all pairs (j', j) (bidirectional)
  – target side: pairs (i', i) with $i' < i$ (uni-directional)
  additional extension: position encoding/embedding
- ***multi-head attention:***
  use of several attention scores $A_n[\cdot, \cdot], n = 1, ..., N$
- ***stacking of attention:***
  several layers (as in RNNs)

Transformer Approach: Architecture

key quantities:

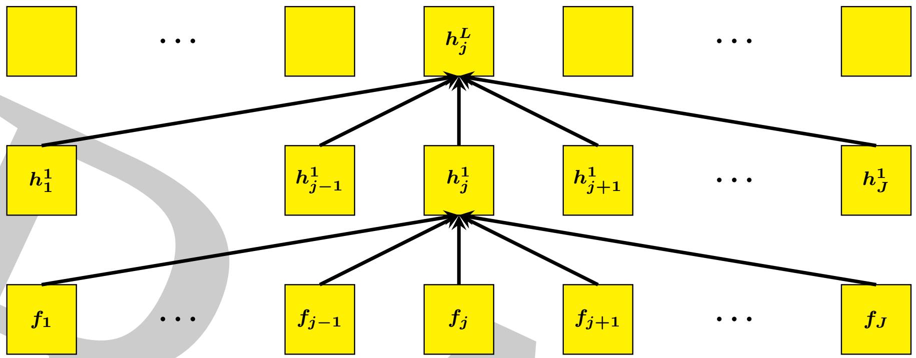
- **self-attention on source side:**
  – **word representation h_j with**
  – **self-attention weights $\alpha(j|j')$**
- **target side:**
  – **state vectors s_i with**
  – **self-attention weights $\alpha(i|i')$**

  – **context vectors c_i with**
  – **cross-attention weights $\alpha(j|i)$**



in addition:
– **everywhere: several layers $l = 1...L$**
– **multi-head for cross-attention $n = 1...N$**

Transformer Approach
Self-Attention: Source Sentence



mechanism:

– **several layers** $l = 1, .., L$ **with representation vectors** $h_j^{(l)}$
initializiation by embedding: $h_j^{(l=0)} := \tilde{f}_j$

– **self-attention weights** $\alpha^{(l+1)}(j'|j)$ **with attention scores** $A[\cdot, \cdot]$**:**

$$\alpha^{(l+1)}(j'|j) = \texttt{softmax}(A[h_j^{(l)}, h_{j'}^{(l)}])$$

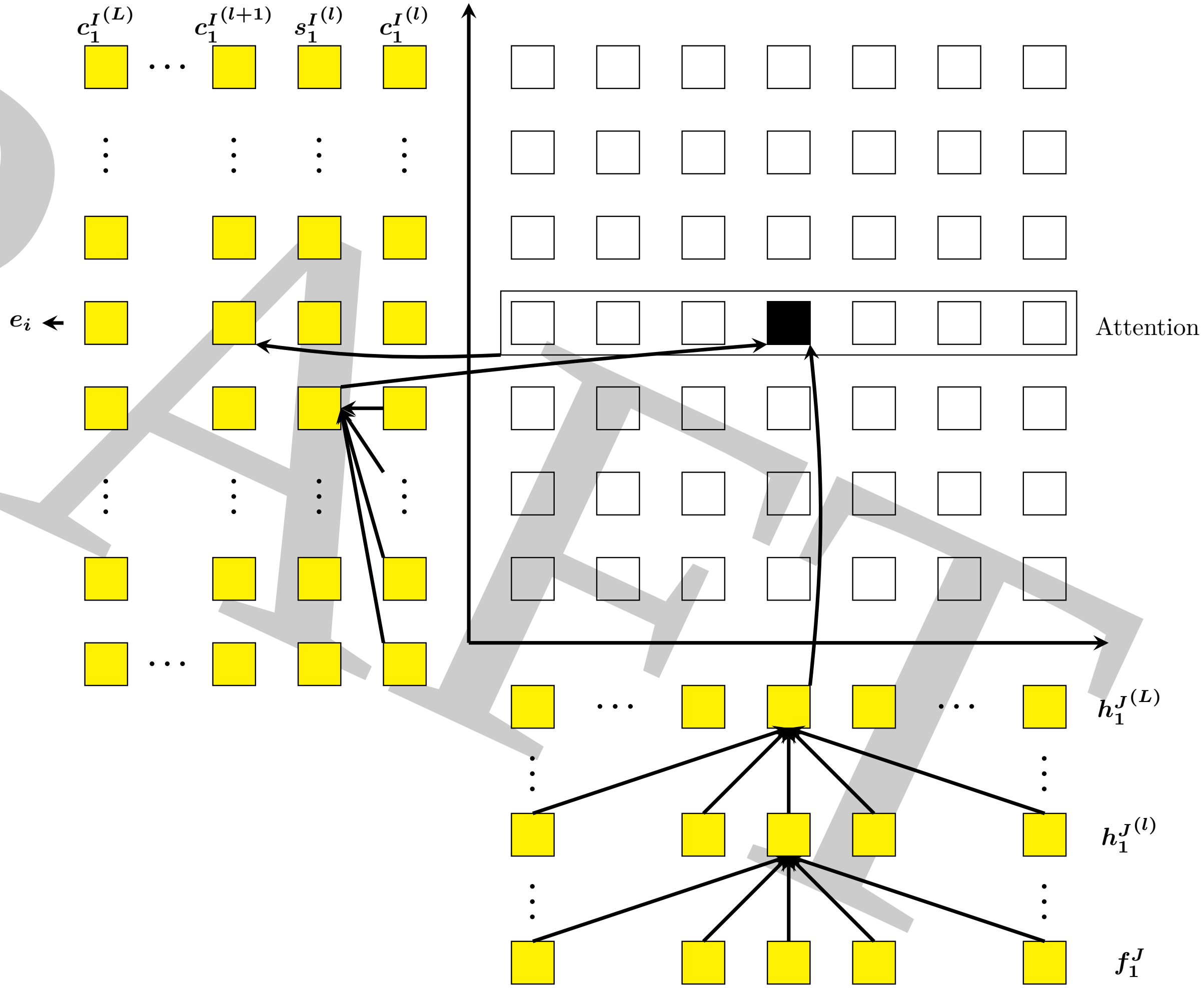
– **representation vector** $h_j^{(l+1)}$**:**

$$h_j^{(l+1)} = \sum_{j'=1}^{J} \alpha^{(l+1)}(j'|j) \cdot h_{j'}^{(l)}$$

Transformer Approach: Target Sentence

several layers over target positions $i = 1, ..., I$:

- **state vectors s_i (self-attention)**
- **context vectors c_i (multi-head attention)**



- **self-attention between target positions:**
  - **self-attention weights $\alpha^{(l+1)}(i'|i-1)$:**

$$\boldsymbol{\alpha}^{(l+1)}(i'|i-1) = \texttt{softmax}(A[c_{i-1}^{(l)}, c_{i'-1}^{(l)}])$$

  - **state vectors $s_{i-1}^{(l+1)}$ on target side:**

$$s_{i-1}^{(l+1)} = \sum_{i'=1}^{i-1} \alpha^{(l+1)}(i'|i-1) \cdot c_{i'-1}^{(l)}$$

  - **with the context vector c_i^l**

- **cross-attention between source and target positions**
  - **cross-attention weights $\alpha^{(l+1)}(j|i)$:**

$$\alpha^{(l+1)}(j|i) = \texttt{softmax}\big(A[s_{i-1}^{(l)}, h_j^{(L)}]\big)$$

  - **context vectors $c_i^{(l+1)}$:**

$$c_i^{(l+1)} = \sum_{j=1}^{J} \alpha^{(l+1)}(j|i) \cdot h_j^{(L)} \qquad \textbf{with} \quad c_i^{(l=0)} := \tilde{e}_i$$

Transformer Approach: Output Generation

output generation:

– ReLU activation:

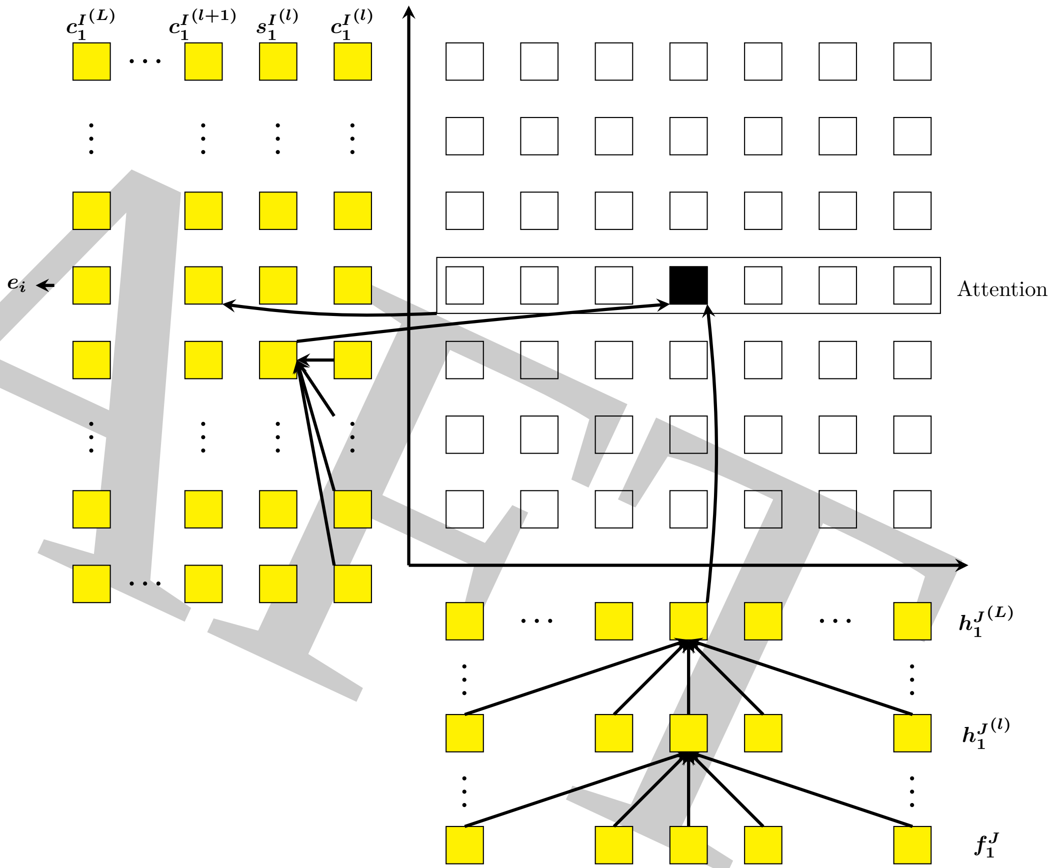
$$t_i = ReLU(C_i^{(L)})$$

– softmax output:

$$y_i \equiv p_i(e|e_1^{i-1}, f_1^J) = \texttt{softmax}(t_i)$$

– for comparison in original attention approach:

$$\begin{aligned} y_i &\equiv p_i(e|e_1^{i-1}, f_1^J) \\ &= \texttt{softmax}(\tilde{e}_{i-1}, s_{i-1}, c_i) \end{aligned}$$



Transformer Approach: Multi-Head Attention

- principle: for each head index $n = 1, ...N$:
  – attention scores $A_n(\cdot, \cdot)$
  – attention weights $\alpha_n(\cdot, \cdot)$
  for self-attentions of source and target strings and for cross-attention

- stacked vectors for:
  – source state vectors: $H_j^{(l)} := [h_{j,1}^{(l)}, \cdots, h_{j,n}^{(l)} \cdots, h_{j,N}^{(l)}]$
  – target state vectors: $S_i^{(l)} := [s_{i,1}^{(l)}, \cdots, s_{i,n}^{(l)} \cdots, s_{i,N}^{(l)}]$
  – context vectors: $C_i^{(l)} := [c_{i,1}^{(l)}, \cdots, c_{i,n}^{(l)} \cdots, c_{i,N}^{(l)}]$

- example: cross-attention between source and target positions
  – cross-attention weights $\alpha_n^{(l+1)}(j|i)$:

$$\alpha_n^{(l+1)}(j|i) = \texttt{softmax}\big(A_n[s_{i-1}^{(l)}, h_j^L]\big)$$

  – context vectors $c_{i,n}^{(l+1)}$:

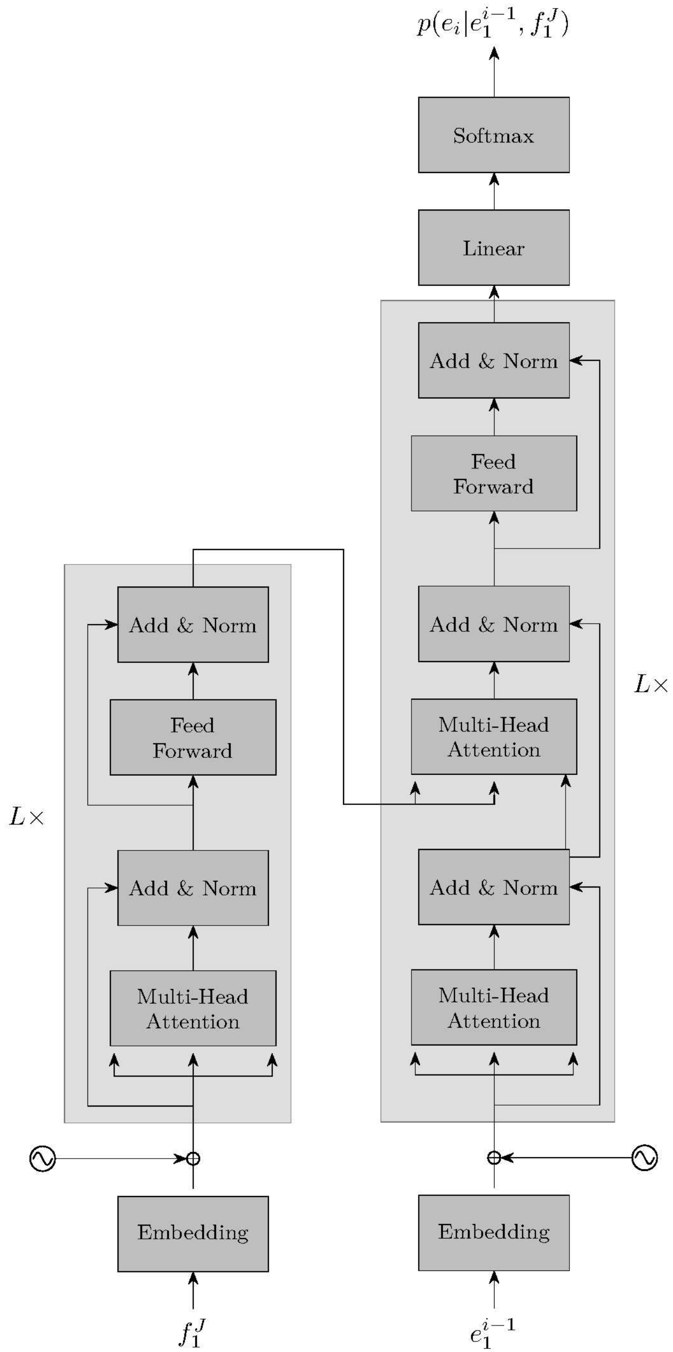
$$c_{i,n}^{(l+1)} = \sum_{j=1}^{J} \alpha_n^{(l+1)}(j|i) \cdot h_j^{(L)} \qquad \text{with} \quad c_{i,n}^{(l=0)} := \tilde{e}_i$$

Transformer Approach: Multi-Layer Architecture

terminology in Google paper for attention mechanism:

definitions:
key/query: arguments in $A_n[\cdot,\cdot]$
value: state/context vectors

(key, value, query):
– self-attention: (h_j, h_j, h_j)
– self-attention: (c_i, c_i, c_i)
– cross-attention: (h_j^L, h_j^L, s_i)

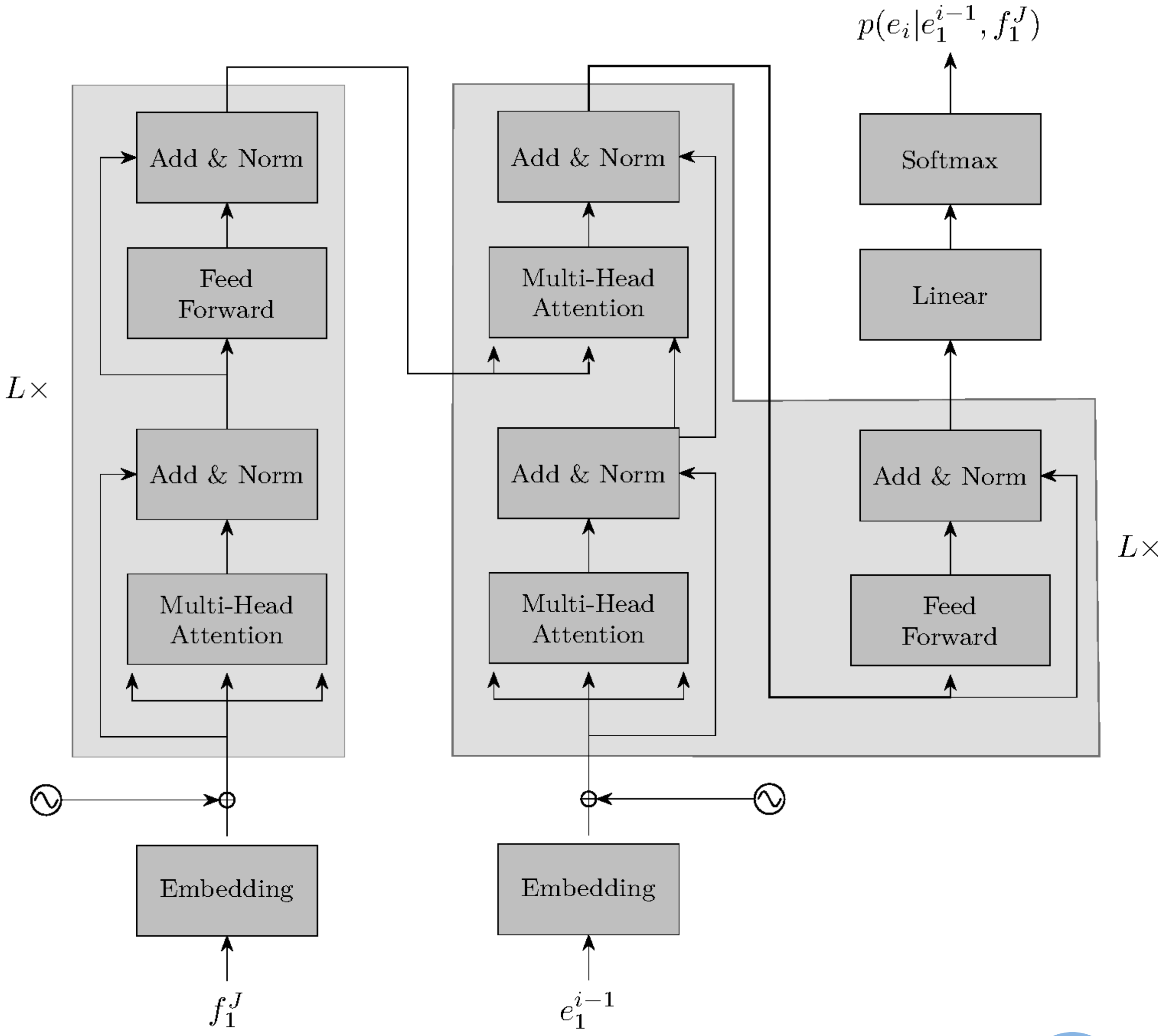


Transformer Approach: Multi-Layer Architecture

terminology in Google paper for attention mechanism:

definitions:
key/query: arguments in $A_n[\cdot,\cdot]$
value: state/context vectors

(key, value, query):
- **self-attention: (h_j, h_j, h_j)**
- **self-attention: (c_i, c_i, c_i)**
- **cross-attention: (h_j^L, h_j^L, s_i)**



Transformer Approach: Conclusions

transformer approach:

- **allows *direct* pairwise interactions between word position pairs for both source and target string**
- **present hardware structure:**
  supports efficient implementation of transformer model:
  – matrix-vector products
  – parallelization
- **experimental results:**
  – significant improvement over baseline attention approach: +3% BLEU
  – time: much faster
  – general remark: BLEU favors n-gram/phrase models over attention models

8 Training Revisited: Effect of Language Model

preview: even a simple model learns an implicit LM!

problem: acoustic model will be masked by LM!

figure: illustration of two windows around n:

c_1^N	○	●	●	●	●	○	○	○	○	○	●	□	○	○	.	.	.	.	.	.	.
x_1^N:	●	●	●	●	●	●	●	●	●	●	●	□	●	●	●	●	●	●	●	●	●

we consider an arbitrary position n and its (large) context $[n-M, n+M]$ (sufficiently large for position n):

- **starting point: large full context:** $pr\left(c_{n-M}^{n+M}, x_{n-M}^{n+M}\right)$
- **selecty two sizes (k, m) for context windows and summing out:**

$$pr\left(c_{n-k}^{n+k}, x_{n-m}^{n+m}\right) = \sum_{...} pr\left(c_{n-M}^{n+M}, x_{n-M}^{n+M}\right)$$

- **note: full-sentence context with boundaries: $[1, N]$ is a special case.**

- **explicit LM and factorization:**

$$pr\left(c_{n-k}^{n+k},\, x_{n-m}^{n+m}\right) = pr\left(c_{n-k}^{n+k}\right) \cdot pr\left(x_{n-m}^{n+m} \,\middle|\, c_{n-k}^{n+k}\right)$$

the only assumption: correct desciption of reality (i.e. required context length!)

- **we *assume* an ANN model with bigram LM dependence and cross-entropy training:**

$$\hat{q}\left(c_n \,\middle|\, c_{n-1},\, x_{n-m}^{n+m}\right) = pr\left(c_n \,\middle|\, c_{n-1},\, x_{n-m}^{n+m}\right) = \frac{pr\left(c_{n-1}, c_n\,,\, x_{n-m}^{n+m}\right)}{\sum_{\tilde{c}_n} pr\left(c_{n-1}, \tilde{c}_n\,,\, x_{n-m}^{n+m}\right)}$$

with $pr\left(c_{n-1}, c_n\,,\, x_{n-m}^{n+m}\right) = \displaystyle\sum_{\tilde{c}_{n-k}^{n+k}:\, \tilde{c}_{n-1}=c_{n-1},\, \tilde{c}_n=c_n} pr\left(\tilde{c}_{n-k}^{n+k}\right) \cdot pr\left(x_{n-m}^{n+m} \,\middle|\, \tilde{c}_{n-k}^{n+k}\right)$

conclusion: the true LM dependence is masked by the model!
resulting problem for the model: change of domain, i.e. LM context

remarks:

- a true distribution with with 1:1 correspondence for $k = m$:

$$pr\left(x_{n-m}^{n+m} \,\middle|\, c_{n-m}^{n+m}\right) = \prod_{i=n-m}^{n+m} pr(x_i|c_i)$$

  conclusion: even for this extremely simple case, we see an effect of the LM!

- question: why not model bigram dependence directly by summing out?

$$pr(c_{n-1}^{n}, x_{n-m}^{n+m}) = pr(c_{n-1}^{n}) \cdot pr(x_{n-m}^{n+m}|c_{n-1}^{n})$$

  reason: hidden complexity in $pr(x_{n-m}^{n+m}|c_{n-1}^{n})$

- renormalize and pseudo posterior:
- conclusion for choosing the ANN model:
  we should use the appropriate context length k for the LM:

$$q\left(c_n \,\middle|\, c_{n-k}^{n-1}, x_{n-m}^{n+m}\right) := \frac{\sum\limits_{c_{n+1}^{n+k}} q(c_{n-k}^{n+k}) \cdot \prod\limits_{i=n-k}^{n+k} q\left(c_i \,\middle|\, c_{i-1}, x_{n-m}^{n+m}\right)}{\sum_{\tilde{c}_n} \ldots}$$

  principle: correct ??
  exercise: correct index error in index $i = n - k$

full sentence context (in lieu of moving context window)
along with elementary 1:1 observation model $pr(x_i|c_i)$
(include c_0 **as sentence start symbol):**

$$pr(c_1^N, x_1^N) = pr(c_1^N) \cdot pr(x_1^N|c_1^N) = pr(c_1^N) \cdot \prod_{i=1}^{N} pr(x_i|c_i)$$

$$pr(c_{n-1}^n, x_1^N) = \sum_{c_1^N \setminus c_{n-1}^n} pr(c_1^N, x_1^N) = \sum_{c_1^N \setminus c_{n-1}^n} pr(c_1^N) \cdot \prod_{i=1}^{N} pr(x_i|c_i)$$

$$pr(c_n|c_{n-1}, x_1^N) = \frac{pr\big(c_{n-1}, c_n\,,\, x_1^N\big)}{\sum_{\tilde{c}_n} pr\big(c_{n-1}, \tilde{c}_n\,,\, x_1^N\big)}$$

important conclusion: the bigram dependence in the emprirical posterior distribution is the result of a long-range LM dependence.

consider four cases:
– unigram with full sentence x_1^N
– unigram with window x_{n-m}^{n+m}: no simplification
– unigram with window and local model: yes simplification
– bigram with window and local model

simplified model and situation (notation for stationary modelling):

$$q(c_0, x_{0-m}^{0+m}$$

question: only unigram LM? — NO! No!
– the model learns the UNIGRAM posterior distribution:

$$pr(c_0|x_{0-m}^{0+m}$$

– the unigram posterior is the result of the full-context LM
– assume the large symmetric window for input and output:

$$\begin{aligned} pr(c_0, x_{0-m}^{0+m}) &:= \sum_{c_{0-M}^{0+M} \backslash c_0} pr(c_{0-M}^{0+M}) \cdot pr(x_{0-M}^{0+M}|c_{0-m}^{0+M}) \\ &:= \sum_{c_{0-M}^{0+M} \backslash c_0} pr(c_{0-M}^{0+M}) \cdot \prod_i pr(x_i|c_i) \end{aligned}$$

true distribution with LARGE window around position n:

$$pr\left(c_{n-M}^{n+M}, x_{n-M}^{n+M}\right) = pr\left(c_{n-M}^{n+M}\right) \cdot pr\left((x_{n-M}^{n+M} | c_{n-M}^{n+M}\right)$$

$$pr_n(c, x_1^N) = \sum_{c_1^N : c = c_n} pr(c_1^N) \cdot pr(x_1^N | c_1^N)$$

impossible to compute analytically

even for very local true dependence:

$$pr(x_1^N | c_1^N) = \prod_n pr(x_n | c_n)$$

$$pr_n(c | x_1^N) = \frac{pr_n(c, x_1^N)}{\sum_{\tilde{c}} pr_n(\tilde{c}, x_1^N)}$$

effect of cross-entropy learning:

– full sequence model:

$$\hat{q}(c_n | x_{n-m}^{n+m}) = \ldots$$

– limited window x_{n-m}^{n+m}:

$$\hat{q}(c_n | x_{n-m}^{n+m}) = \ldots$$

$$pr_n(c', c, x_{n-m}^{n+m})$$

true distribution with full dependencies:

$$pr(c_1^N, x_1^N) \;=\; pr(c_1^N) \cdot pr(x_1^N | c_1^N)$$

$$pr_n(c, x_1^N) \;=\; \sum_{c_1^N : c = c_n} pr(c_1^N) \cdot pr(x_1^N | c_1^N)$$

impossible to compute analytically

even for very local true dependence:

$$pr(x_1^N | c_1^N) = \prod_n pr(x_n | c_n)$$

$$pr_n(c | x_1^N) \;=\; \frac{pr_n(c, x_1^N)}{\sum_{\tilde{c}} pr_n(\tilde{c}, x_1^N)}$$

effect of cross-entropy learning:

– full sequence model:

$$\hat{q}(c_n | x_{n-m}^{n+m}) = \ldots$$

– limited window x_{n-m}^{n+m}:

$$\hat{q}(c_n | x_{n-m}^{n+m}) = \ldots$$

$$pr_n(c', c, x_{n-m}^{n+m}) := \sum_{c_{n-m}^{n+m} : c' = c_{n-1}, c = c_n} pr(c_{n-m}^{n+m}) \cdot \prod_{i=n-m}^{n+m} pr(x_i | c_i)$$

– notation problem for $c', c)$! x_1^N is given and different!
– ANN model $q(c_n|x_1^N)$ with no (explicit) LM dependence
– result of training:

$$\hat{q}(c_n|x_1^N) \;=\; pr(c_n|x_1^N) \;=\; \frac{\sum_{c_1^N:c=c_n} pr(c_1^N) \cdot pr_n(x_1^N|c_1^N)}{\sum_{\tilde{c}} \sum_{c_1^N:\tilde{c}=c_n} pr(c_1^N) \cdot pr_n(x_1^N|c_1^N)}$$

– effect: ANN is affected by LM prior!

Cross-Entropy Criterion: Implicit Learning of LM

re-consider derivation:
dependencies: limited window in lieu of full sentence
special model:

$$q(c_{n-1}, c_n, x_{n-m}^{n+m}) = ???$$

replace sentence context by window context:

$$pr(c_1^N, x_1^N) \qquad \textbf{vs.} \quad pr\left(c_{n-k}^{n+k}, x_{n-m}^{n+m}\right)$$

8.2 Separation of LM and Renormalization

baseline acoustic models (with synchronization):
– FSM-HMM
– cross-attention

concepts for normalization:
– factorization followed by local re-normalization
– global re-normalization followed by a factorization

overview:
– a large variety of models and combinations is possible
– here: selection of three most general concepts

concepts:
– separation of language models
– move from generative models to log-linear models
– factorization as a product of conditional probabilities

Why do we want to separate the language model?

language model: $p_\vartheta(c_1^N)$ or $q_\vartheta(c_1^N)$:= prior of output string c_1^N

justification: training conditions in ASR (? similar in HTR and MT):

– annotated audio data, i. e. pairs of $[x_1^T, c_1^N]$:
1000 hours of audio = 10 Mio words

– text (=output) data, i. e. text strings c_1^N only:
100 Mio words

resulting difference: a factor of 10 for training data

String-to-String Modelling: Three Concepts - Overview

notation ϑ: full set of parameters (maybe 500 Mio!)
for both language model and observation model (typically: no sharing):

- **generative model at sequence level (obsolete?):**

$$p_\vartheta(c_1^N|x_1^T) := \frac{p_\vartheta(x_1^T, c_1^N)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} p_\vartheta(x_1^T, \tilde{c}_1^{\tilde{N}})} = \frac{p_\vartheta(c_1^N) \cdot p_\vartheta(x_1^T|c_1^N)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} p_\vartheta(\tilde{c}_1^{\tilde{N}}) \cdot p_\vartheta(x_1^T|\tilde{c}_1^{\tilde{N}})}$$

  with the concept of a JOINT model $p_\vartheta(x_1^N, c_1^N)$
  (which imitates the empirical distribution!)

- **log-linear model at sequence level (additional parameters α and β)**
  (with re-normalization; CRF=conditional random field):

$$p_\vartheta(c_1^N|x_1^T) := \frac{q_\vartheta^\alpha(c_1^N) \cdot q_\vartheta^\beta(c_1^N|x_1^T)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} q_\vartheta^\alpha(\tilde{c}_1^{\tilde{N}}) \cdot q_\vartheta^\beta(\tilde{c}_1^{\tilde{N}}|x_1^T)} = \frac{\prod_n \; q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1} x_1^T)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} \prod_n \; q_\vartheta^\alpha(\tilde{c}_n|\tilde{c}_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|\tilde{c}_0^{n-1}, x_1^T)}$$

  with independent positive (normalized) models $q_\vartheta(c_1^N)$ and $q_\vartheta(c_1^N|x_1^T)$

- **direct model using factorization: = log-linear model at symbol level:**

$$p_\vartheta(c_1^N|x_1^T) = \prod_n p_\vartheta(c_n|c_0^{n-1}, x_1^T) \;=\; \prod_n \frac{q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\sum_{\tilde{c}_n} q_\vartheta^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_0^{n-1}, x_1^T)}$$

  advantage: easy re-normalization unlike sequence level

generative and generalized model:

- **re-normalization at string level:**
  – requires a sum over all strings
  – difficult problem with specific approximations/simplifications
- **Bayes decison rule:**
  sum over all strings in denominator is NOT required!
- **acoustic model (hybrid HMM, CTC, transducer/RNN-T, ...):**
  – often limited context, e. g.: $q_\vartheta(c_n|c_0^{n-1}, x_1^T) = q_\vartheta(c_n|c_{n-1}, x_1^T)$
  maybe left and right context: $q_\vartheta(c_n|c_{n-1}, c_{n+1}, x_1^T)$
  – note neural attention: different (?)
- **training criterion: cross-entropy at sequence level**
  – denominator IS required
  – case of generative model:
  approximation: drop denominator $\rightarrow$ maximum likelihood for strings
  – case of log-linear model: see later

log-linear modelling:

- **sentence-level (or global) re-normalization:**

$$p_\vartheta(c_1^N|x_1^T) := \frac{\prod_n \; q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} \prod_n \; q_\vartheta^\alpha(\tilde{c}_n|\tilde{c}_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|\tilde{c}_0^{n-1}, x_1^T)}$$

Bayes decision rule:

$$\begin{aligned} x_1^T \to \hat{c}_1^{\hat{N}}(x_1^T) &= \operatorname*{argmax}_{N,c_1^N} \left\{ p_\vartheta(c_1^N|x_1^T) \right\} = \operatorname*{argmax}_{N,c_1^N} \left\{ \log \; p_\vartheta(c_1^N|x_1^T) \right\} \\ &= \operatorname*{argmax}_{N,c_1^N} \left\{ \sum_n \left[\alpha \cdot \log q_\vartheta(c_n|c_0^{n-1}) + \beta \cdot \log q_\vartheta(c_n|c_0^{n-1}, x_1^T) \right] - const(c_1^N) \right\} \\ &= \operatorname*{argmax}_{N,c_1^N} \left\{ \sum_n \left[\alpha \cdot \log q_\vartheta(c_n|c_0^{n-1}) + \beta \cdot \log q_\vartheta(c_n|c_0^{n-1}, x_1^T) \right] \right\} \end{aligned}$$

warning: looks like LOCAL combination, but is based on GLOBAL re-normalization

- **symbol-level (or local) re-normalization:**

$$p_\vartheta(c_1^N|x_1^T) := \prod_n \frac{q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\sum_{\tilde{c}_n} q_\vartheta^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_0^{n-1}, x_1^T)} = \frac{\prod_n q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\prod_n \sum_{\tilde{c}_n} q_\vartheta^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_0^{n-1}, x_1^T)}$$

note: (dis)similarity to global re-normalization

- **sentence-level (or global) re-normalization:**

$$p_\vartheta(c_1^N|x_1^T) := \frac{\prod_n \; q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} \prod_n \; q_\vartheta^\alpha(\tilde{c}_n|\tilde{c}_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|\tilde{c}_0^{n-1}, x_1^T)}$$

- **symbol-level (or local) re-normalization:**

$$p_\vartheta(c_1^N|x_1^T) \;:=\; \frac{\prod_n q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\prod_n \sum_{\tilde{c}_n} q_\vartheta^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_0^{n-1}, x_1^T)} = \frac{\prod_n \; q_\vartheta^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_0^{n-1}, x_1^T)}{\sum_{\tilde{N},\tilde{c}_1^{\tilde{N}}} \prod_n \; q_\vartheta^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_0^{n-1}, x_1^T)}$$

remark: analyze notation and compare with global re-normalization

Bayes decision rule:

$$\begin{aligned}
x_1^T \to \hat{c}_1^{\hat{N}}(x_1^T) \;&=\; \underset{N,c_1^N}{\text{argmax}} \left\{ p_\vartheta(c_1^N|x_1^T) \right\} = \underset{N,c_1^N}{\text{argmax}} \left\{ \sum_n \log \; p_\vartheta(c_n|c_0^{n-1}, x_1^T) \right\} \\
&=\; \underset{N,c_1^N}{\text{argmax}} \left\{ \sum_n \Big[\alpha \cdot \log q_\vartheta(c_n|c_0^{n-1}) + \beta \cdot \log q_\vartheta(c_n|c_0^{n-1}, x_1^T) \right. \\
&\qquad\qquad \left. - \log \sum_{\tilde{c}_n} q_\vartheta^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_0^{n-1}, x_1^T) \Big] \right\} \\
&\overset{?}{\cong}\; \underset{N,c_1^N}{\text{argmax}} \left\{ \sum_n \Big[\alpha \cdot \log q_\vartheta(c_n|c_0^{n-1}) + \beta \cdot \log q_\vartheta(c_n|c_0^{n-1}, x_1^T) \Big] \right\}
\end{aligned}$$

end-to-end: many meanings:
– no pronunciation lexicon
– no separate Language model
– a single training criterion
- ...

Effect of Language Model and End-to-End Concepts

(full) sequence posterior probability $p_\vartheta(W|X)$
with output $W = a_1^S$ **and input** $X = x_1^T$ **using a log-linear model:**

$$p_\vartheta(W|X) \;=\; \frac{q_\vartheta^\alpha(W) \cdot q_\vartheta^\beta(W|X)}{\sum_{\tilde{W}} q_\vartheta^\alpha(\tilde{W}) \cdot q_\vartheta^\beta(\tilde{W}|X)}$$

with the two basic model components (and independent sets of parameters ϑ**):**

- **language model (LM)** $q_\vartheta(W)$
  – prior that scores the syntactic-semantic adequacy of a word sequence
  – idea: can be learned from text data only (e. g. 100 million words)
  – no manual annotation required!
- **acoustic model (AM):** $q_\vartheta(W|X)$
  – learned from manually transcribed audio data (e. g. 1000 hours = 10 million words)
  – first-order models (hybrid HMM, CTC, RNN-T): $q_\vartheta(W|X)$

training criterion for acoustic model: cross-entropy of composed model at sequence level:

$$\max_\vartheta \left\{ \sum_r \log \, p_\vartheta(W_r | X_r) \right\}$$

with training data: (audio,text) pairs $[X_r, W_r], \; r = 1, ..., R$

RWTH AACHEN UNIVERSITY

mathematical analysis of exact training criterion:

- **composed model with fixed parameters for language model:**

$$p_\vartheta(W|X) \;=\; \frac{q^\alpha(W)\cdot q_\vartheta^\beta(W|X)}{\sum_{\tilde{W}} q^\alpha(\tilde{W})\cdot q_\vartheta^\beta(\tilde{W}|X)}$$

$$q_\vartheta(W=a_1^S|X) \;=\; \sum_{s_1^T}\prod_t q(s_t|s_{t-1},W,\vartheta)\cdot q_t(a_{s=s_t}|X,\vartheta)$$

- **sequence discriminative training = cross-entropy training of composed model:**

$$\hat{\vartheta} \;=\; \arg\max_\vartheta\Big\{\sum_r \log\, p_\vartheta(W_r|X_r)\Big\}$$

- **approximation: ignore sum in denominator:**

$$\begin{aligned}\hat{\vartheta} \;&\cong\; \arg\max_\vartheta\Big\{\sum_r \log\,\Big(q^\alpha(W_r)\cdot q_\vartheta^\beta(W_r|X_r)\Big)\Big\}\\ &=\; \arg\max_\vartheta\Big\{\sum_r \log\, q^\alpha(W_r) + \sum_r \log\, q_\vartheta^\beta(W_r|X_r)\Big\}\\ &=\; \arg\max_\vartheta\Big\{\sum_r \log\, q_\vartheta(W_r|X_r)\Big\}\end{aligned}$$

resulting criterion: cross-entropy of acoustic model
note: NO effect of language model in this approximation

Training in ASR: Acoustic and Language Models

exact training criterion for data $[X_r, W_r],\ r = 1, ..., R$:

$$\max_{\vartheta} \Big\{ \sum_r \log p_\vartheta(W_r|X_r) \Big\} \qquad \textbf{with} \qquad p_\vartheta(W|X) \;=\; \frac{q^\alpha(W) \cdot q_\vartheta^\beta(W|X)}{\sum_{\tilde{W}} q^\alpha(\tilde{W}) \cdot q_\vartheta^\beta(\tilde{W}|X)}$$

exact training criterion: hard optimization problem due to denominator

- **baseline training: ignore denominator**
  – first-order models (hybrid HMM, CTC, RNN-T): $q_\vartheta(W|X)$
  – variants: full sum or best path (frame-wise CE, Viterbi)
  – result: no effect of language model (LM) on the training of the acoustic model (AM)
- **complete criterion: *sequence discriminative training* (ASR: MMI)**
  better approximation: approximate sum in denominator
  – use word hypothesis lattice
  – use simplifed language model (e. g. phoneme fourgram LM; lattice-free MMI)
  overall effect: LM affects training of AM!
- **history (mathematical optimization problem!):**
  Bahl et al./IBM 1986, Normandin 1991, Valtchev 1996,
  Povey 2002 and 2016, Heigold 2005 and 2012

End-to-End Concept

reconsider training criterion for (audio,text) pairs $[X_r, W_r],\ r = 1, ..., R$:

$$\max_{\vartheta} \left\{ \sum_r \log\, p_\vartheta(W_r|X_r) \right\} \qquad\qquad p_\vartheta(W|X) := \frac{q^\alpha(W) \cdot q_\vartheta^\beta(W|X)}{\sum_{\tilde{W}} q(\tilde{W}) \cdot q_\vartheta^\beta(\tilde{W}|X)}$$

different variants of ***sequence discriminative training*** in ASR:
– string error (see above)
– symbol error in string context, e. g. edit distance: phonemes, letters, words
various concepts: Povey's minimum word/phoneme error rate,
state-level minimum Bayes risk (sMBR), ...

terminology: What does *end-to-end* mean?

- training criterion: one global criterion for optimum performance, independent of model structure
- optimization strategy of the criterion: one monolithic strategy? e. g. gradient search and backpropagation
- monolithic structure of a model:
  – maybe simplicity/elegance of programming?
  – what about adequacy/performance?

End-to-End Concept: Dichotomy between Acoustic Model and Language Model

remarks:

- **terminology: virtually meaningless**
- **all "good" systems are trained in a *end-to-end* style:
  minimze a criterion that reflects WER (or performance)**
- **practical condition virtually everywhere in ASR:
  difference between amounts of transcribed audio and text data:
  – transcribed audio: 1000 hours = 10 million words
  – text (from press, books, internet,...): 100 million words and more
  (maybe) similar: HTR and MT**
- **an optimal system should have an architecture
  that can benefit from the two types of training data for
  *acoustic model and language model***

consider full model for a string pair $[X = x_1^T, W = a_1^S]$:

$$p_\vartheta(W|X) = \frac{q^\alpha(W) \cdot q_\vartheta^\beta(W|X)}{\sum_{\tilde{W}} q^\alpha(\tilde{W}) \cdot q_\vartheta^\beta(\tilde{W}|X)}$$

$$q_\vartheta(W = a_1^S|X) = \sum_{s_1^T} \prod_t q_\vartheta(s_t, a_{s=s_t}|s_{t-1}, X)$$

training criteria (for a 'single' sentence):

- **full model $p_\vartheta(W|X)$, i. with LM: *sequence discriminative training* (mathematical problem: denominator with sum over label sequences):**
  – symbol level in sequence context: Povey's WER, sMBR, ...
  – sequence level (MMI): $\vartheta \to \log\, p_\vartheta(W|X)$
- **only acoustic model $q_\vartheta(W = a_1^S|X)$, i. e. without LM and re-normalization:**
  – sum over all paths:
  ***sum criterion*: $\vartheta \to \log\; q_\vartheta(W = a_1^S|X) = \log \sum_{s_1^T} \prod_t q_\vartheta(s_t, a_{s=s_t}|s_{t-1}, X)$**
  – single best path:
  ***frame-wise cross entropy*: $\vartheta \to \log\; \max_{s_1^T} \prod_t q_\vartheta(s_t, a_{s=s_t}|s_{t-1}, X)$**

conclusions:
– ultimately: (virtually) all criteria have the form of a cross-entropy
– more specification is needed: type of model and of its random variables

9 Fully Fledged Systems
(along with Experimental Results)

outline:

– ASR: some experimental results
– language models in ASR
– MT (machine translation): from signals to symbols

9.1 ASR: Acoustic Modelling

ANN approaches in ASR:

- **1988 [Waibel & Hanazawa$^+$ 88]:**
  phoneme recognition using time-delay neural networks (convolutional NNs!)
- **1989 [Bridle 89]:**
  softmax operation ('Gaussian posterior') for normalization of ANN outputs
- **1989 [Bourlard & Wellekens 89]:**
  - **for squared error criterion, ANN outputs can be interpreted as class posterior probabilities (rediscovered: [Patterson & Womack 66])**
  - **hybrid type of HMM:**
    emission probabilities are replaced by ANN ouputs, i. e. label posteriors
- **1991 [Bridle & Dodd 91] backpropagation for HMM discriminative training at word level**
- **1993 [Haffner 93]: sum over label-sequence posterior probabilities in hybrid HMMs (*sequence discriminative training*)**
- **1994 [Robinson 94]: recurrent neural network in hybrid HMM**
  - **competitive results on WSJ task**
  - **his work remained a singularity in ASR**
- **...**

hybrid HMMs until 2008:

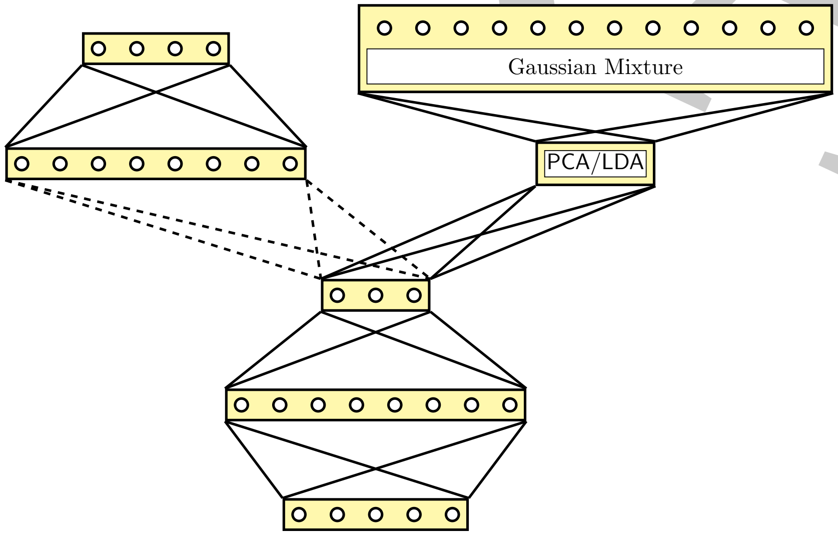
ANNs were never superior to Gaussian mixture models

procedures advocated by ASR community (and NIST/DARPA):

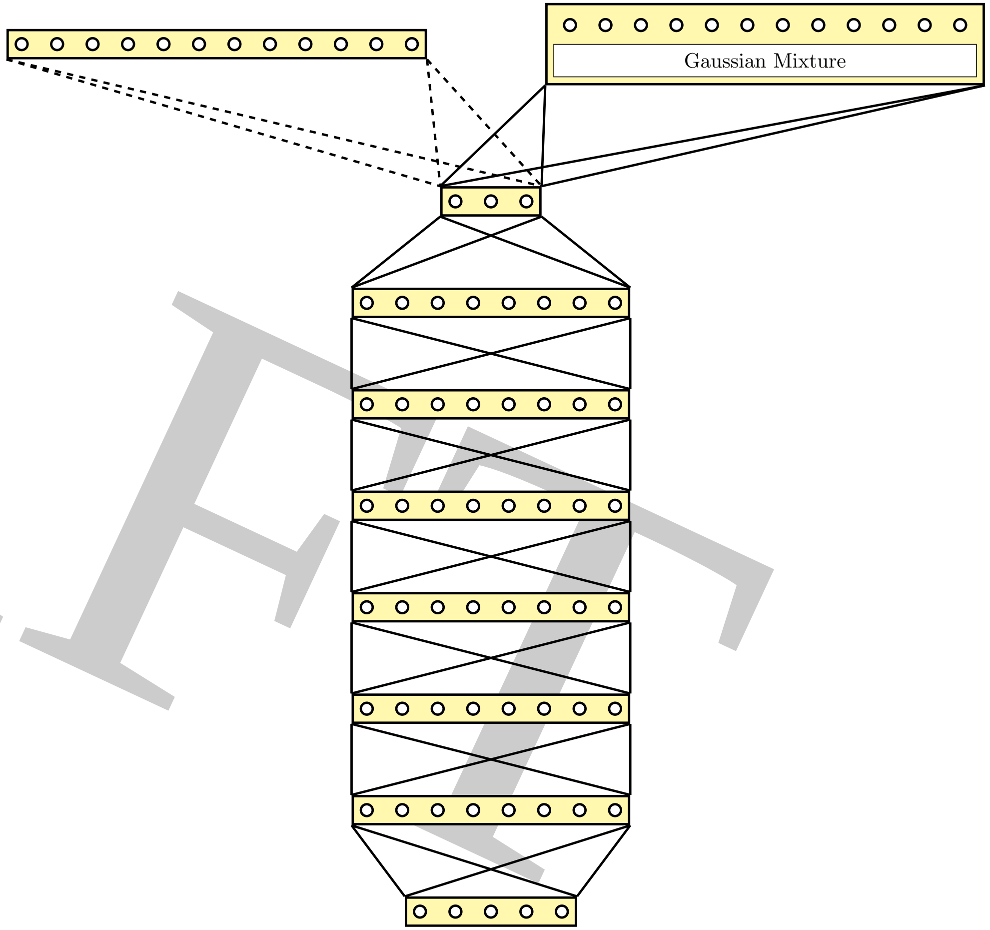
- **well defined performance metric in ASR:**
  – edit distance: substitutions, deletions, insertions
  – no phonetic transcription required!
- **shared public data bases:**
  – for testing: must be the same (to make results comparable)
  – for training: depends on goals
- **evaluation campaigns (public and project internal):**
  – well-known phenomenon in pattern recognition: risk of *training on the testing data*
  – remedy: change testing data over time
- **public data/tasks for ASR:**
  – TI digit strings, TIMIT, RM-1k, WSJ-5k, NAB-20k, Switchboard, ...
  – AMI meeting corpus, TED-LIUM corpus, Librispeech task, ...
- **evaluation concepts adapted in other fields:**
  – handwriting recognition
  – object recognition
  – machine translation

- **tandem approach: two parts:**
  MLP for feature extraction + generative HMM

  [Fontaine & Ris$^+$ 97, Hermansky & Ellis$^+$ 00]
- **extensions, e. g. bottleneck concept**
  [Stolcke & Grezl$^+$ 06, Grezl & Fousek 08],
  [Valente & Vepa$^+$ 07, Tüske & Plahl$^+$ 11]



RWTH's Tandem Structure
[Tüske & Plahl$^+$ 11]



tandem approaches (i. e. DNN-based explicit feature extraction)

- 2000 [Hermansky & Ellis^{+} 00]: multiple layers of processing by combining Gaussian model and ANN for ASR
- 2006 [Stolcke & Grezl^{+} 06]: cross-domain and cross-language portability
- 2007 [Valente & Vepa^{+} 07]: 8% WER reduction on LVCSR
- 2011 [Tüske & Plahl^{+} 11]: 22% WER reduction on LVCSR

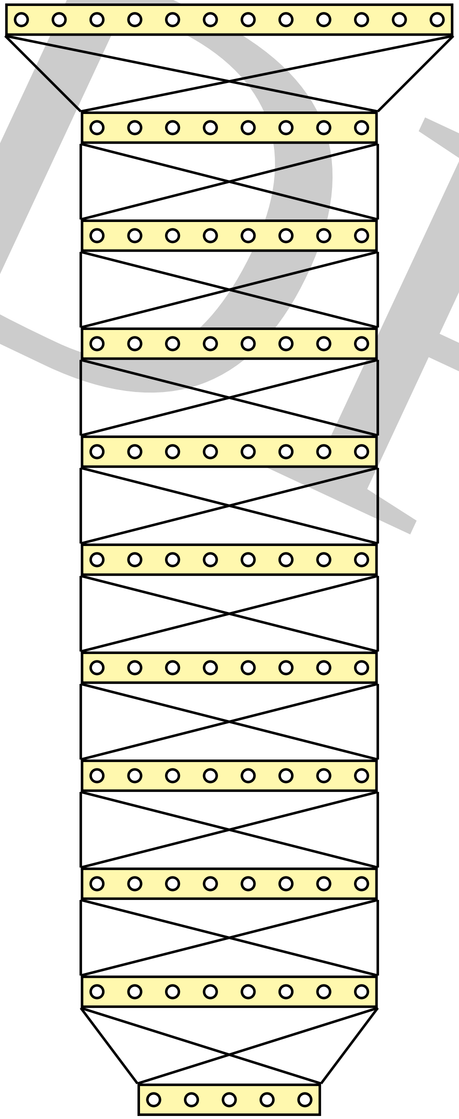
hybrid approaches:

- 2008 [Graves 08]: CTC - good results on LSTM RNN for handwriting task
- 2010 [Dahl & Ranzato^{+} 10]: improvement in phone recognition on TIMIT
- 2011 [Seide & Li^{+} 11, Dahl & Yu^{+} 12]: Microsoft Research
  – fully-fledged hybrid approach
  – 30% WER reduction on Switchboard 300h
- since 2012: other teams confirmed reductions of WER by 20% to 30%

comparison: hybrid vs. tandem approach:

– hybrid approach: more monolithic and compact
– the same structure in training and testing
– widely used nowadays

Artificial Neural Networks (ANN) and Deep Learning:

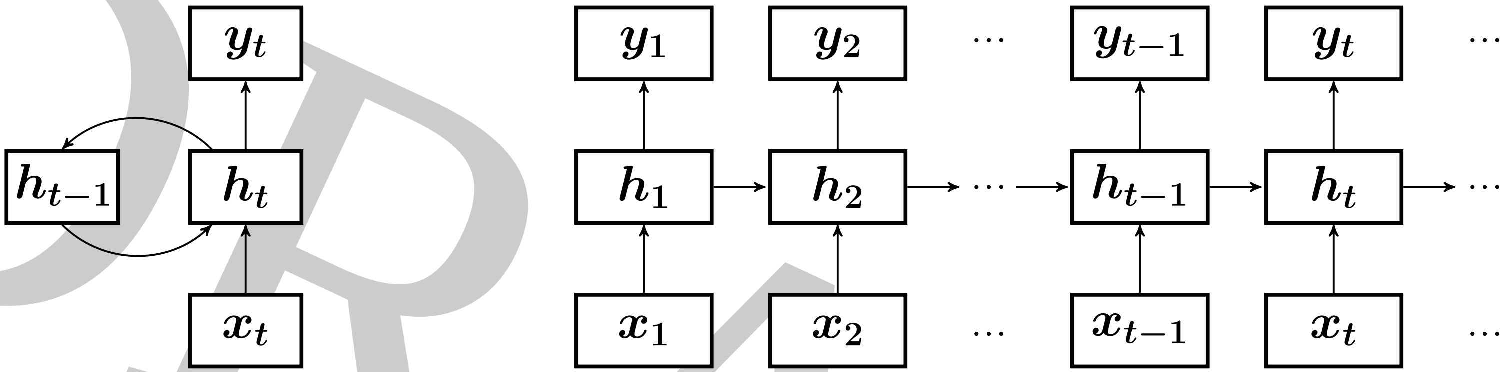


question: what is different now after 30 years?

answer: we have learned how to (better) handle
a complex mathematical optimization problem:

- more powerful hardware
  (e. g. GPUs)
- empirical recipies for optimization:
  practical experience and heuristics,
  e.g. layer-by-layer pretraining
- result: we are able to handle more
  complex architectures
  (deep MLP, RNN, LSTM-RNN, transformer, conformer, etc.)

ASR: sequence-to-sequence processing



from simple ANN to RNN:
– introduce a memory (or context) component to keep track of history
– result: two types of input at time t: memory h_{t-1} and observation x_t

extensions:

- **bidirectional structure [Schuster & Paliwal 97]**
- **LSTM: long short-term memory [Hochreiter & Schmidhuber 97, Gers & Schraudolph$^+$ 02]**

Systematic Experiments on QUAERO English Eval 2013 (Tüske et al. RWTH 2017)

QUAERO task: broadcast news/conversations, podcasts, TED lectures

Word error rates [%] on QUAERO English Eval 2013 (note: acoustic input features were optimized for acoustic model):

Acoustic Model: hybrid HMM			Language Model	
Model		Criterion	Count PP=131.1	Count+ANN PP=92.0
Gaussian Mixtures		max.lik.	20.7	
		seq.disc. training	19.2	16.1
Neural Net	FF MLP	frame-wise CE	11.6	
		seq.disc. training	10.7	9.0
	LSTM RNN	frame-wise CE	10.6	
		seq.disc. training	9.8	8.2

observations:
- **improvements by acoustic ANNs: 50% relative**
- **improvement by language model ANN: 15% relative**
- **total improvements by deep learning: 60% relative (from 19.2% to 8.2%)**

Tasks: Switchboard and Call Home

- conversational speech: telephone speech, narrow band;
  challenging task: initial WER: 60% (and higher) on Switchboard

- training data for acoustic model: Switchboard corpus
  - about 300 hours of speech
  - about 2400 two-sided recordings with an average of 200 seconds
  - 543 speakers
- test set Hub5'00
  - 20 telephone recordings form Switchboard studies (SWB)
  - 20 telephone conversations from Call-Home US English Speech (CHM)
  - total: 3.5 hours of speech
- training data for language model
  - vocabulary size fixed to 30k
  - Switchboard corpus: 2.9M running words
  - Fisher corpus: 21M running words

ASR: Results (April 2019)

baseline models:
– language model: 4-gram count model
– acoustic model: hybrid HMM with CART (allophonic) labels:
LSTM bi-RNN with frame-wise cross-entropy training

– speaker/channel adaptation: i-vector [Dehak & Kenny$^+$ 11]
– affine transformation [Gemello & Manai$^+$ 06, Miao & Metze 15]

word error rates [%]:

adaptation	methods	SWB	CHM	average
no	baseline approach	9.7	19.1	14.4
	+ seq. discr. training (sMBR)	9.6	18.3	13.9
	+ LSTM-RNN language model	7.7	15.8	11.7
yes (i-vector)	baseline approach	9.0	18.0	13.5
	+ seq. discr. training (sMBR)	8.4	17.2	12.8
	+ LSTM-RNN language model	6.8	15.1	10.9
+ adaptation by affine transformation		6.7	13.5	10.2

overall improvements over baseline:
– 33% relative reduction in WER
– by seq. discr. training, LSTM-RNN language model and adaptation

Best Results on Call Home (CHM) and Switchboard (SWB)
(best word error rates [%] reported)

team	CHM	SWB	training data, remarks
Johns Hopkins U 2017	18.1	9.0	300h, no ANN-LM, single model, data perturbation
Microsoft 2017	17.7	8.2	300h, ResNet, with ANN-LM
ITMO U 2016	16.0	7.8	300h, with ANN-LM, model comb., data perturbation
Google 2019/arXiv	14.1	6.8	300h, attention models
RWTH U 2017	15.7	8.2	300h, with ANN-LM, model comb.
RWTH U 2019/arXiv	13.5	6.7	300h, single models, adaptation
Microsoft 2017	12.0	6.2	2000h, model comb.
IBM 2017	10.0	5.5	2000h, model comb.
Capio 2017	9.1	5.0	2000h, model comb.

ASR: Librispeech Task
(Vassil Panayotov & Daniel Povey)

RWTH AACHEN UNIVERSITY

speech data: read audiobooks from the LibriVox project
with training data:
– acoustic model: 960 hrs of speech
– language model: 800 million words

word error rates[%]:

team	approach	dev 1st half	dev 2nd half	test 1st half	test 2nd half
Irie, Zeyer et al. RWTH (Interspeech 2019)	attention with BPE units, 'no' LM	4.3	12.9	4.4	13.5
	+ LSTM-RNN LM	3.0	9.1	3.5	10.0
	+ transformer LM	2.9	8.8	3.1	9.8
Lüscher, Beck et al. RWTH (Interspeech 2019)	hybrid HMM, CART, 4g LM	4.3	10.0	4.8	10.7
	+ seq. disc. training	3.7	8.7	4.2	9.3
	+ LSTM-RNN LM	2.4	5.8	2.8	6.2
	+ transformer LM	2.3	5.2	2.7	5.7

Zeghidour et al., FB 2018	gated CNN with letters/words	3.2	10.1	3.4	11.2
Irie et al., Google 2019	attention with WPM units	3.3	10.3	3.6	10.3
Park et al., Google 2019	attention ... data augmentation	-	-	2.5	5.8

results on phoneme/grapheme RNN-transducer (RNN-T):
IBM research [Saon & Tüske$^+$ 2021] and RWTH [Zhou & Berger$^+$ 2021]

table and results from [Saon & Tüske$^+$ 2021]
on Switchboard (SWB) and Call-Home (CHM):

authors	team	approach		WER[%]	
		acoust.model	lang.model	SWB	CHM
Saon & Tüske$^+$ 2021	IBM	RNN-T	LSTM-RNN	6.3	13.1
Tüske & Saon$^+$ 2020	IBM	attention	LSTM-RNN	6.4	12.5
Park & Chan$^+$ 2019	Google	attention	LSTM-RNN	6.8	14.1
Hadiani & Sameti$^+$ 2018	JHU	LF-MMI	RNN	7.5	14.6
Irie & Zeyer$^+$ 2019	RWTH	hybrid HMM	transformer	6.7	12.9

more results on Italian and Spanish (conversational telephone speech)

conclusions based on [Saon & Tüske$^+$ 2021, Zhou & Berger$^+$ 2021]:
RNN-T has similar performance like hybrid HMM

9.2 LM: Language Modelling in ASR

Bayes decision rule for generating word sequence w_1^N from speech signal x_1^T (assuming a log-linear model and dropping the denominator):

$$x_1^T \rightarrow \hat{w}_1^{\hat{N}}(x_1^T) \;=\; \operatorname*{argmax}_{N,w_1^N} \left\{ q_\vartheta^\alpha(w_1^N) \cdot q^\beta(w_1^N|x_1^T) \right\}$$

language model: the prior probability $q_\vartheta(w_1^N)$ and its parameters ϑ

observations about the language model $q_\vartheta(w_1^N)$:
– it can be learned from text only (unlabeled data!), e. g. from 100-1000 Mio words
– it can improve performance dramatically

question:

How to measure the quality of an LM (without a recognition experiment)?

considerations:

- **use prior $q_\vartheta(w_1^N)$ in Bayes decision rule,**
  but it depends on the single sentence and its length
- **define a sufficiently large test corpus**
  by concatenating all test sentences to a LONG super sentence
  (use special symbols for sentence end and unknown word)
- **apply the LM probability to this super sentence of N words**
  and perform normalization:
  – geometric average of probability per word by computing N-th root
  – invert average probability into perplexity: = average *effective* vocabulary size

formal definition of perplexity PP:

$$PP \;:=\; \Big(q_\vartheta(w_1^N)\Big)^{-1/N} \;=\; \Big(\prod_{n=1}^{N} q_\vartheta(w_n|w_0^{n-1})\Big)^{-1/N}$$

$$\log PP \;=\; -\frac{1}{N} \cdot \sum_{n=1}^{N} \log\, q_\vartheta(w_n|w_0^{n-1})$$

with artificial start symbol w_0

prior probability $q_\vartheta(w_1^N)$ of any sentence $w_1^N = w_1 ... w_n ... w_N$:

$$q_\vartheta(w_1^N) \;=\; \prod_{n=1}^{N} q_\vartheta(w_n|w_1^{n-1}) = \prod_{n=1}^{N} q_\vartheta(w_n|w_{n-2}, w_{n-1})$$

simplified dependence: word trigram language model

disambiguation of homophones (IBM 1985):

- **homophones: two, too, to**

  Twenty-two people are too many to be put in this room.

- **homophones: write, Wright, right**

  Please write to Mrs. Wright right away.

- **limited history: Markov chain of order k:**
  limit the dependence on the full history w_0^{n-1} to the immediate k predecessor words:

$$q_\vartheta(w_n|w_0^{n-1}) \ := \ q_\vartheta(w_n|w_{n-k}^{n-1})$$

  two modelling concepts:
  - **event counts (e. g. word fourgrams, trigrams, bigrams, unigrams) and smoothing**
  - **FF-MLP with word embeddings, i. e. a mapping from word symbols to vectors**

- **unlimited history: RNN and extensions**

natural training criterion for a corpus w_1^N: minimum perplexity

$$\max_\vartheta \left\{ \sum_{n=1}^{N} \log \ q_\vartheta(w_n|w_0^{n-1}) \right\}$$

– **equivalent to cross-entropy training (or maximum likelihood)**
– **resulting estimates: relative frequencies based on event counts**

- **training criterion for string pairs $[X_r, W_r],\ r = 1, ..., R$:**

$$\max_{\vartheta} \Big\{ \sum_r \log\ p_\vartheta(W_r|X_r) \Big\}$$

- **composed model of language model and acoustic model:**

$$p_\vartheta(W|X) \ = \ \frac{q_\vartheta^\alpha(W) \cdot q^\beta(W|X)}{\sum_{\tilde{W}} q_\vartheta^\alpha(\tilde{W}) \cdot q^\beta(\tilde{W}|X)}$$

for a fixed acoustic model (i. e. no free parameters)

- **approximation: separate $q_\vartheta^\alpha(W_r)$ and ignore remaining term:**

$$\begin{aligned}
\hat{\vartheta} \ &= \ \arg\max_{\vartheta} \Big\{ \sum_r \log\ p_\vartheta(W_r|X_r) \Big\} \\
&= \ \arg\max_{\vartheta} \Big\{ \alpha \cdot \sum_r \log\ q_\vartheta(W_r) + \sum_r \log\ \frac{q^\beta(W|X)}{\sum_{\tilde{W}} q_\vartheta^\alpha(\tilde{W}) \cdot q^\beta(\tilde{W}|X)} \Big\} \\
&\cong \ \arg\max_{\vartheta} \Big\{ \sum_r \log\ q_\vartheta(W_r) \Big\}
\end{aligned}$$

result of approximation: acoustic model cancels!

Language Modelling: Perplexity

- **result: training criterion for language model with strings $W_r = [w_1...w_i...w_{I_r}] = w_1^{I_r}$:**

$$\hat{\vartheta} \;=\; \arg\max_{\vartheta} \Big\{ \sum_r \log\; q_\vartheta(w_1^{I_r}) \Big\} \;=\; \arg\max_{\vartheta} \Big\{ \sum_r \sum_{i=1}^{I_r} \log\; q_\vartheta(w_i|w_0^{i-1}) \Big\}$$

- **interpretation: superstring = concatenation of all strings $W_r, r = 1..., R$:**

$$w_1^N := W_1^R = [w_1...w_i...w_{I_r}]_{r=1}^{r=R}$$

note: special handling (extra symbols) for string boundaries

- **normalized criterion = inverse geometric average: perplexity PP**

$$\log PP \;:=\; \log\; 1 \Big/ \sqrt[N]{q_\vartheta(w_1^N)} \;=\; -1/N \cdot \sum_{n=1}^{N} \log\; q_\vartheta(w_n|w_0^{n-1})$$

terminology:
– maximum likelihood (ASR community)
– cross-entropy (ANN community)

goal of language modelling: compute the prior $q_\vartheta(w_1^N)$ of a word sequence w_1^N
– how plausible is this word sequence w_1^N (independently of observation x_1^T!) ?
– measure of language model quality: perplexity PP (= geometric average)
interpretation: effective vocabulary size as seen by ASR decoder/search

$$\log PP \;:=\; \log\, 1 \Big/ \sqrt[N]{q_\vartheta(w_1^N)} \;=\; -1/N \cdot \sum_{n=1}^{N} \log\, q_\vartheta(w_n | w_0^{n-1})$$

perplexity PP on test data (QUAERO)
(Sundermeyer et al.; RWTH 2012, 2015):

approach	PP
baseline: count model	163.7
10-gram MLP	136.5
RNN	125.2
LSTM-RNN	107.8
10-gram MLP with 2 layers	130.9
LSTM-RNN with 2 layers	100.5

interpretation: prediction task:
based on history w_0^{n-1}, predict $p(w_n|...)$

approaches:
– use full history: RNN or LSTM
– truncate history: $\rightarrow$ k-gram MLP

important result: improvement of PP by 40%

Interpolated Language Models: Perplexity and WER

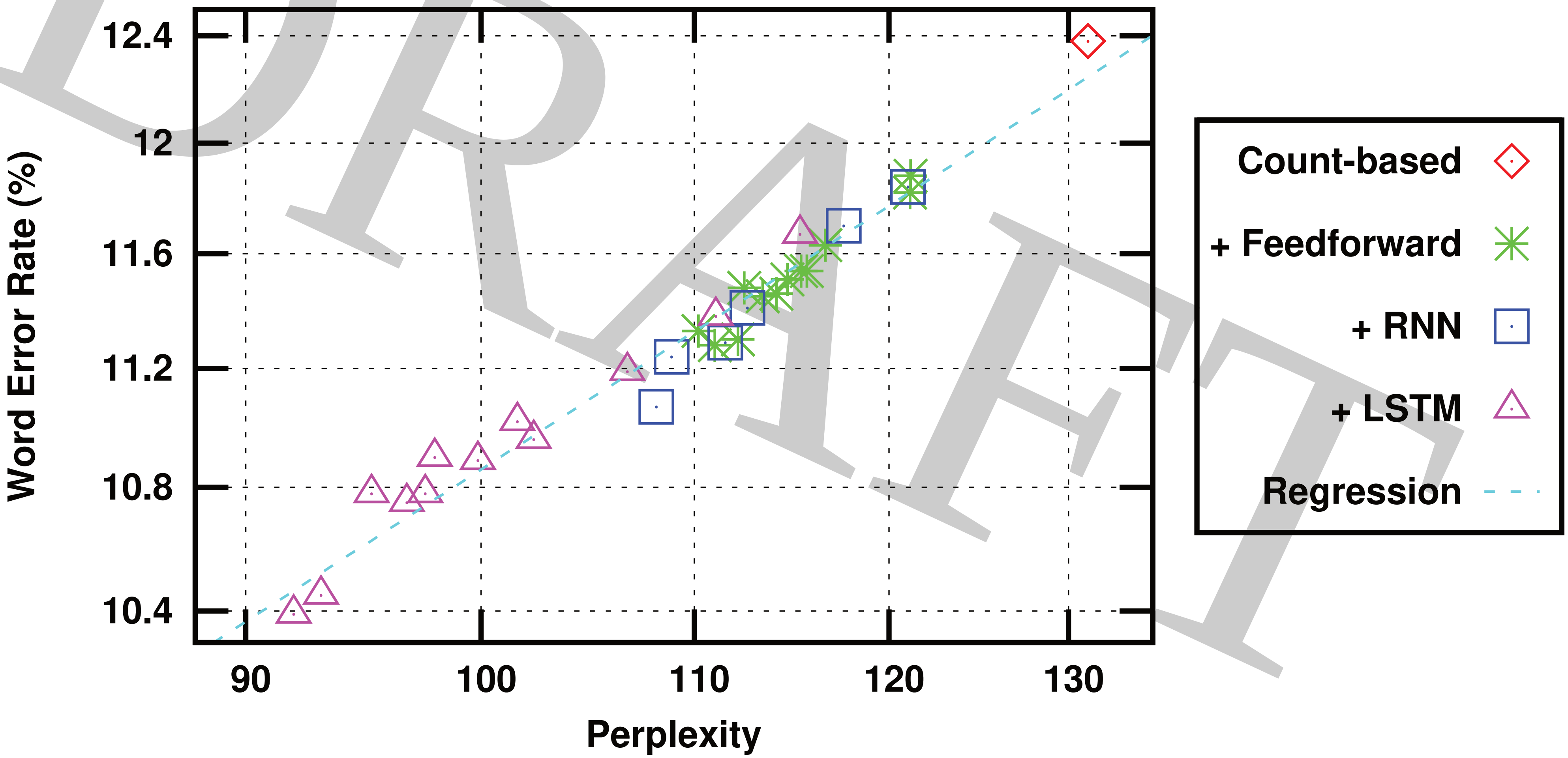
- more details and refinements:
  - use of word classes for softmax in output layer
  - unlimited history of RNN: requires re-design of ASR search
- in practice:
  interpolation of TWO models: count model (3B words) + ANN model (60M words)
- perplexity and word error rate on test data

models	PP	WER[%]
count model	131.2	12.4
+ 10-gram MLP	112.5	11.5
+ Recurrent NN	108.1	11.1
+ LSTM-RNN	96.7	10.8
+ 10-gram MLP with 2 layers	110.2	11.3
+ LSTM-RNN with 2 layers	92.0	10.4

- improvements achieved:
  - perplexity: 30% reduction: from 131 to 92
  - WER: 15% reduction: from 12.4% to 10.4%

Plot: Perplexity vs. Word Error Rate

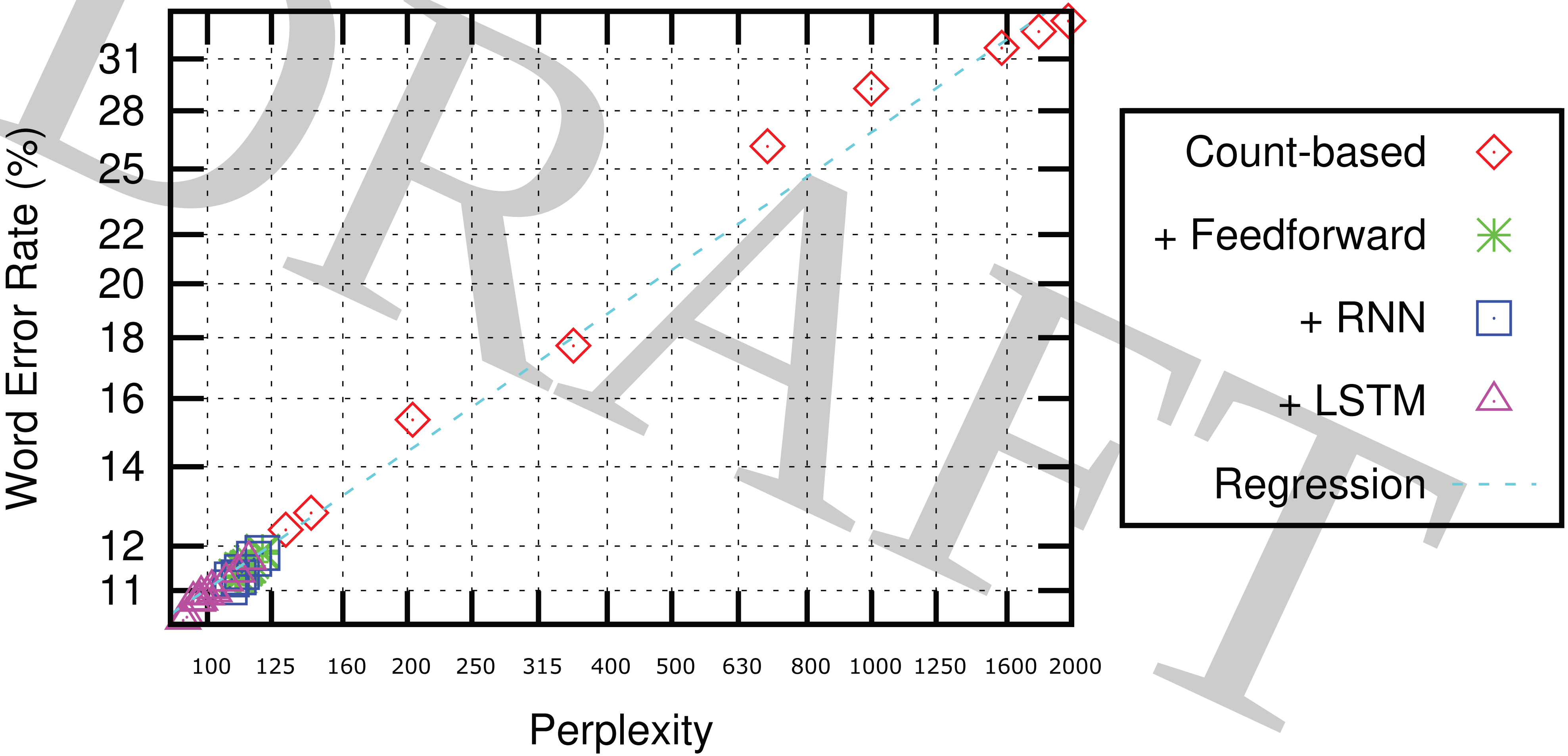
empirical law: $WER = \alpha \cdot PP^{\beta}$ **with** $\beta \in [0.3, 0.5]$
[Makhoul & Schwartz 94, Klakow & Peters 02]



Extended Range: Perplexity vs. Word Error Rate

empirical law: $WER = \alpha \cdot PP^{\beta}$

open question: theoretical justification?



note: Google paper at ICASSP-23: LLM for ASR

encryption method: homophonic ciphers:
each plaintext letter is mapped to one or several ciphertext symbols.
compare with spoken language:
a *homophone* (= pronunciation) has several different writings.

encrypted texts: two examples:
Beale ciphers (Virginia/US 1820/85) and Zodiac killer ciphers (Bay Area/US 1968/9)
– Beale cipher 2: sequence of 762 numbers with 182 distinct numbers
– Zodiac killer 408-cipher: sequence of 408 ’artificial’ symbols with distinct 54 symbols

(sort of) perfect decipherment:

- letter-based language model (of general English) is used to score all possible substitution possibilities
- combinatorial search problem: beam search
- paper at EMNLP 2014: M. Nuhn, J. Schamper, H. Ney: *Improved Decipherment of Homophonic Ciphers.*
- article in *Mental Floss*, 04-Jun-2018:
  `https://www.mentalfloss.com/article/540277/beale-ciphers-buried-treasure`

9.3 LLM: Large Language Models

important result of ANN research:

value of word/symbol representation using ANNs

History:

- **1989 [Nakamura & Shikano 89]:**
  English word category prediction based on neural networks.
- **1993 [Castano & Vidal$^+$ 93]:**
  Inference of stochastic regular languages through simple recurrent networks
- **2000 [Bengio & Ducharme$^+$ 00]:**
  A neural probabilistic language model
- **2002 [Schwenk & Gauvain 02, Schwenk 07]: Continuous space language models**
- **2010 [Mikolov & Karafiat$^+$ 10]:**
  Recurrent neural network based language model
- **2012 RWTH Aachen [Sundermeyer & Schlüter$^+$ 12]:**
  LSTM recurrent neural networks for language modeling
- **2017 [Vaswani & Shazeer$^+$ 17]: transformer architecture (originally for MT)**
- **since 2019 beyond ASR: multi-lingual, multi-task, many parameters (200 billion!)**
  (GPT, Whisper, LaMDA, OPT, Bloom, ChatGPT, ...):
  – GPT: general pretrained transformer
  – LLM: large-scale language models

Word Representations/Embeddings

important component in ANN-based LMs (contrast: count-based LM):
– word/symbol representations/embeddings: vectors in high-dim. space
– in addition to ANN structures (MLP, RNN, LSTM-RNN, transformer, ...)

word representations used without ANN context
(personal communication, Eduardo Lleida, 13-Nov-2023):

- **1971 Salton: information retrieval using term-document matrix**
- **1993 Schütze & Peterson: co-occurrence of two words**
- **2004 Bellegarda: Latent Semantic Modelling for Speech Recognition**
- **2013 Hofmann: Probabilistic Latent Semantic Analysis**

power of LMs and word representations (spirit of *distributional semantics*):
1954 Harris: Words are similar if they appear in similar contexts.
1957 Firth: You shall know a word by the company it keeps.

- papers by [Collobert & Weston 08, Collobert & Weston$^+$ 11]:
  2008: *A Unified Architecture for NLP: Deep Neural Networks with Multitask Learning.*
  2011: *NLP (almost) from Scratch.*

  use of word vectors for *formal* NLP tasks:
  POS/NER tagging, syntactic analysis, semantic role labeling, text classif., ...

- word vectors: (semantic) interpretations and calculations
  examples of relations between word vectors [Mikolov & Corrado$^+$ 13]:

$$Germany - Berlin \cong France - Paris$$
$$king - queen \cong man - woman$$

- 2013/2014: use LM concept for MT [Kaltenbrenner & Blunsom 13, Sutskever & Vinyals$^+$ 14]

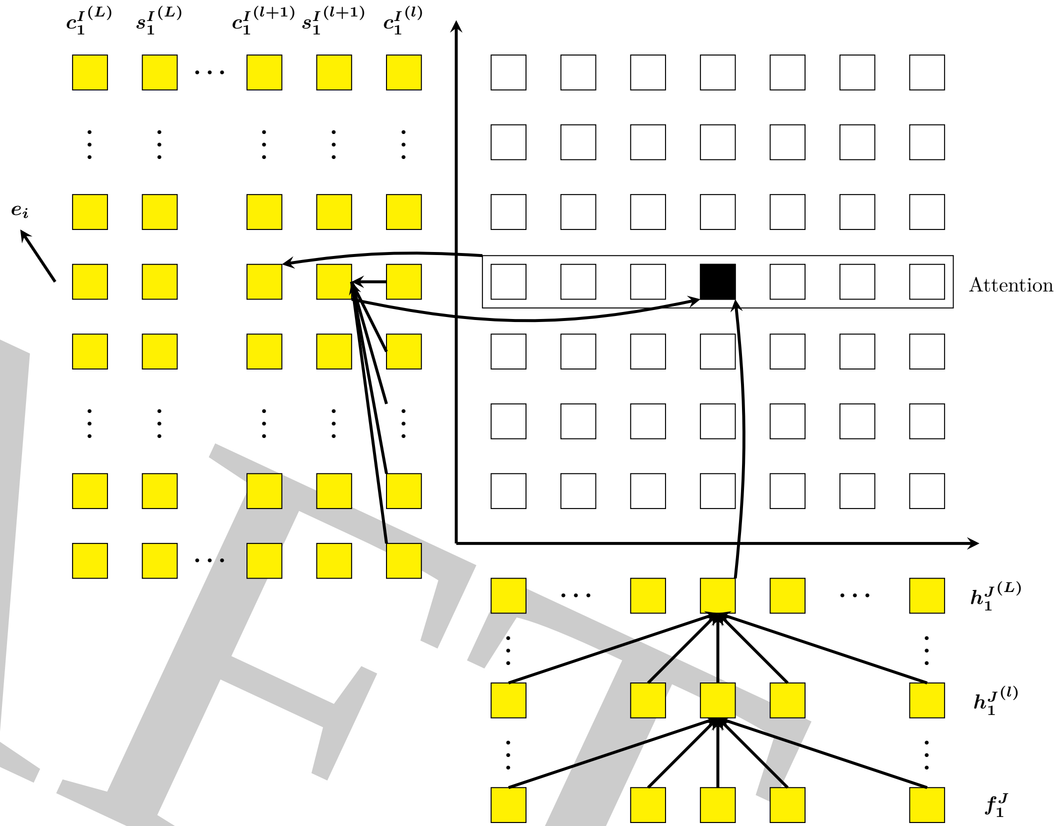
- since 2019: LLMs (large-scale LMs) based on GPT architecture:
  – G: generative: generate text (as opposed to formal NLP tasks)
  – P: pre-trained: based on text without *any* annotation
  – T: transformer: ANN structure for sequence-to-sequence processing

  LLM implies: more data, more parameters (200 Bio), multi-lingual, multi-task, ...

Sequence-to-Sequence Processing: Transformer Approach (Google [Vaswani & Shazeer+ 17])

designed for a 'two-dim.' problem
with input and output sequences:

- keep the *cross-attention* between output and input as in RNN attention [Bahdanau & Cho+ 15]
- for input and output sequence: replace RNN structure by *self-attention*, i. e. pair-wise associations



2020 OpenAI: transformer GPT-3:
– 96 layers, each with 12.288 nodes
– 96 attention heads
in total: 175 Bio parameters

consider MT to be a 1-dim. LM problem:
[input, output] sequences $\rightarrow$ single stream
2013 [Kaltenbrenner & Blunsom 13]
2014 [Sutskever & Vinyals+ 14]
today: GPT successful for many NLP tasks
(generative tasks, beyond MT)

Refining LLMs: *InstructGPT*

InstructGPT **introduced by OpenAI, arxiv, 04-Mar-2022:**
Training language models to follow instructions with human feedback.

three levels of training:

- **pre-training or unsupervised training (using log perplexity):**
  - **training mode: raw text with no annotation**
  - **operation mode (surprising result !):**
    type of task (*prompt*): can be specified in plain language
    e. g. Q&A, summarization, story generation, *dialog!*, ...
    e. g. *multilingual* LLM: translation
    full system operation is described by a triplet (in plain language!):
    triplet $:=$ [prompt, input, output]
    (typically used in so-called *few-shot learning/conditioning*)
- **supervised fine-tuning:**
  - **training data: based on (many) triplets of the above type**
  - **training criterion: (log) perplexity**
    all triplets are interpreted as a *single* sequence of text
- **human feedback and reinforcement learning:**
  - **starting point: system is used to generate the outputs for [prompt, input] pairs**
  - **human evaluation and ranking for LLM-generated outputs**
  - **reinforcement learning based on human scores**

LLM and GPT: Typical Tasks

every-day NLP tasks with plain text for input and output:

- **text summarization:**
  input: ***full text***
  output: ***text summary***

- **story generation:**
  input: ***key words***
  output: ***full text***

- **machine translation (with bilingual training data):**
  input: ***sentence in source language***
  output: ***sentence in target language***

- **conversational dialog (with many turns):**
  input: ***customer query/command***
  output: ***system response***

remarkable property (in contrast to formal NLP tasks):
everything is expressed in terms of plain every-day language:
– system input: formulated by the user
– type of task (*prompt/instruction*): specified by the user
– generated output: smooth fluent language
(primary goal which a language model is designed for)

- large-scale language model (LLM) called *chatGPT*:
  - API introduced on 30-Nov-2022 by OpenAI
  - function: human-like conversational (text) dialog *(unlimited domain)*
  - CEO S. Altman: "costs are eye-watering"
  - operational loss in 2022: 540 Mio USD (416 on computing, 89 on staff)
- OpenAI's technology behind *chatGPT*:
  - baseline architecture *GPT: generative pre-trained transformer*
  - *GPT-3*: with 1.3 to 175 Bio parameters,
    trained on 300 Bio (subword) tokens (cut-off date: June 2020)
  - *InstructGPT* (sibling to *ChatGPT*): refinement with human feedback
- other types of dialog systems:
  - limited-domain, task-oriented dialog
  - explicit dialog strategy: manually designed and coded

  specific systems: *voice command and control*
  - Amazon's Alexa (loss in 2022: 10 Bio USD - 12 000 employees)
  - Apple's Siri
  - Google's (Digital) Assistant

Some LLMs (until 2022)

- OpenAI:
  - 2018 GPT-1: 0,12 Bio
  - 2019 GPT-2: 1,5 Bio
  - 2020 GPT-3: 175 Bio (train: 300 Bio)
  - 2022 *InstructGPT* and *ChatGPT*
- Google:
  - 2018 BERT: 3,3 Bio (train: 300 Bio, 40 epochs)
  - 2019 T5: 11 Bio (train: 1000 Bio)
  - 2020 Meena (for dialog): 2,6 Bio (train: 61 Bio)
  - 2022 LaMDA: 137 Bio (train: 2810 Bio)
  - 2022 PaLM: 540 Bio (train: 780 Bio)
- more LLMs:
  - 2019 BART / Meta: 0,33 Bio (train: 55 Bio, 40 epochs)
  - 2019 Megatron / Nvidia: 3,9 Bio (train: 366 Bio)
  - 2020 DialoGPT / Microsoft: 0,76 Bio (train: 10 Bio)
  - 2022 OPT / Meta: 175 Bio (train: 180 Bio)
- years 2021-2022: more than 50 LLMs
  recent European activities:
  - BLOOM / BigScience: 176 Bio (train: 366 Bio)
  - Luminous / Aleph Alpha (OpenGPT-X): 70 Bio (train: 588 Bio)
  - HPLT (EU project): major EU languages

9.4 MT: Machine Translation

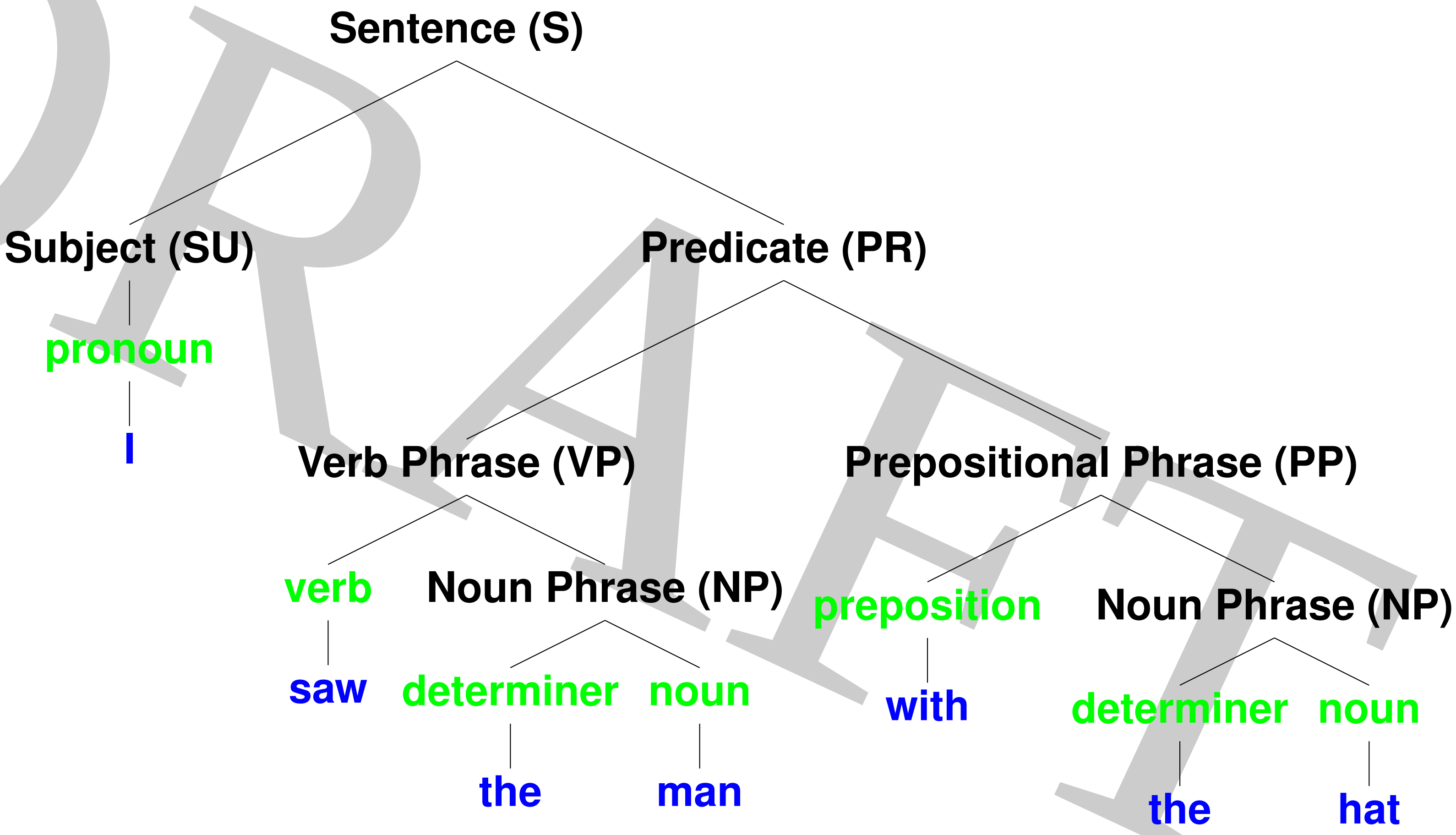
important conceptual aspect:
– from signal/subsymbolic processing to symbolic processing
– similar to LM in ASR
(but in ASR LM is only an auxiliary component!)

statistical approaches were controversial in MT (and other NLP tasks):

- 1969 Chomsky:
  ... the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term.
- result: strict dichotomy until 1995-2005:
  – speech = spoken language: signals, subsymbolic, machine learning
  – language = written text: symbols, grammars, rule-based AI
- until 2000: mainstream approach was rule-based
  – result: huge human effort required in practice
  – problems: coverage and consistency of rules
- 1989-93: IBM Research: statistical approach to MT
  1994: key people (Mercer, Brown) left for a hedge fund
- 1996-2002 RWTH: improvements beyond IBM's approach:
  phrase-based approach and log-linear modelling
- around 2004: from singularity to mainstream in MT
  F. Och (and more RWTH PhD students) joined Google
  2008: service *Google Translate*
- 2015: neural MT: attention mechanism [Bahdanau & Cho$^+$ 15]

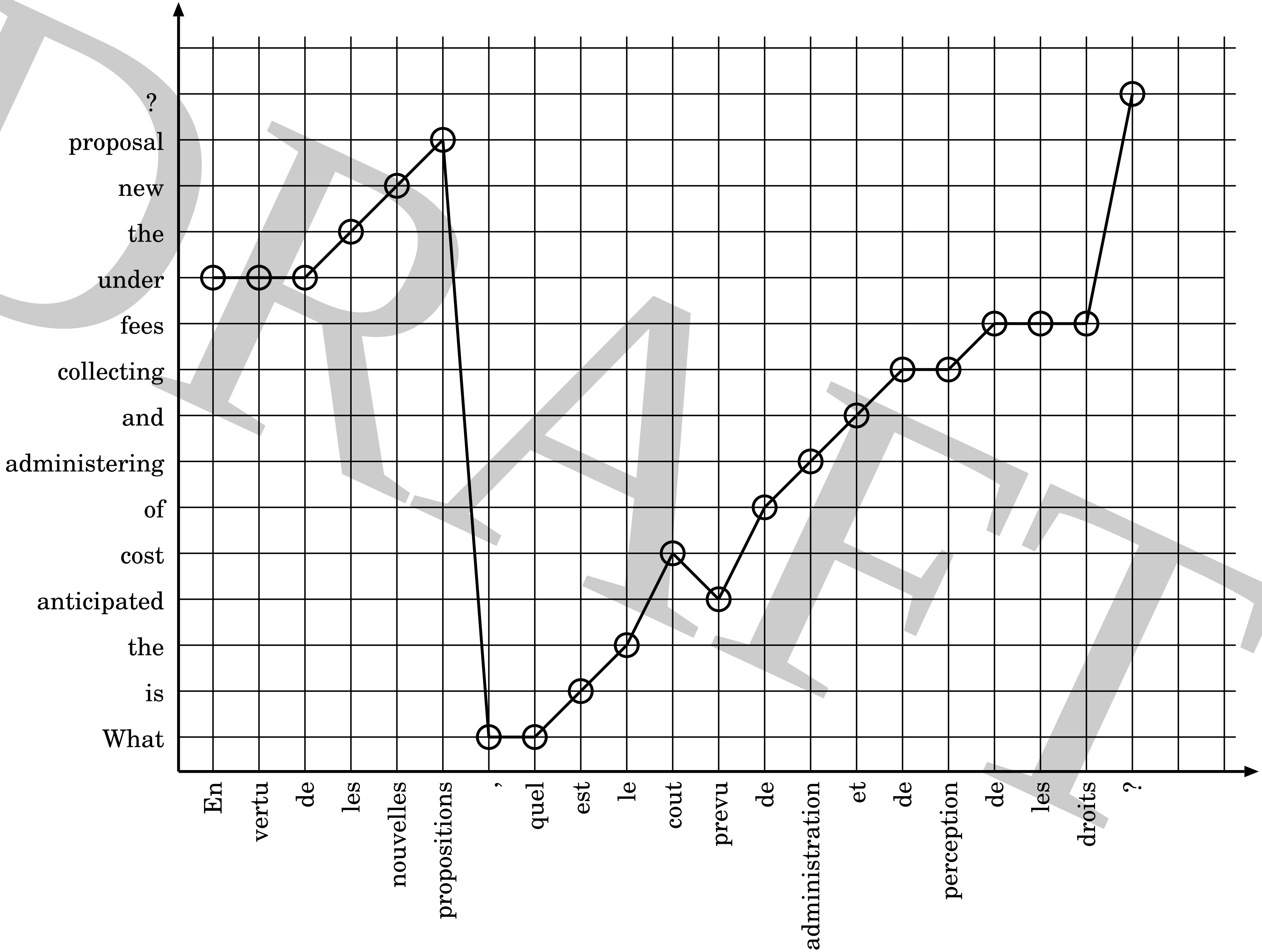
Rule-based Artficial Intelligence
Example: Grammar Rules and Syntactic Structure

- **principle:**

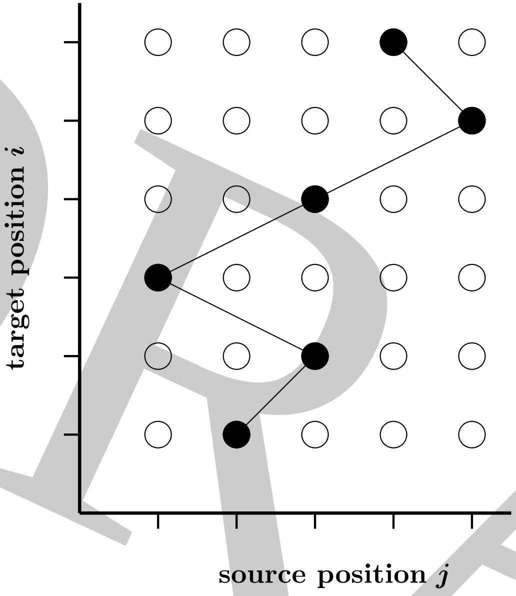


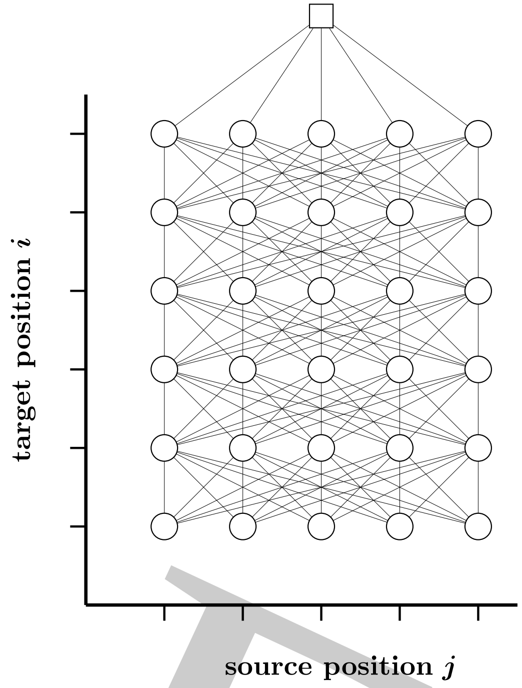
- **extensions along many dimensions**

Word Alignments (based on HMM)
(learned automatically; Canadian Parliament)



Machine Translation: Direct HMM



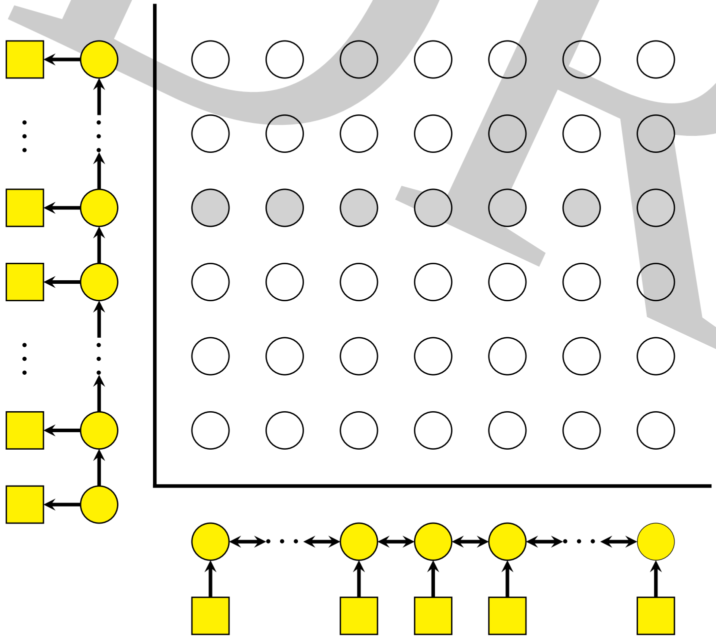


- translation: from source sentence $f_1^J = f_1 ... f_j ... f_J$ to target sentence $e_1^I = e_1 ... e_i ... e_I$
- alignment direction: from target to source: $i \rightarrow j = b_i$
- first-order hidden alignments and factorization:

$$p(e_1^I|f_1^J) = \sum_{b_1^I} p(b_1^I, e_1^I|f_1^J) = \sum_{b_1^I} \prod_i p(b_i, e_i|b_{i-1}, e_0^{i-1}, f_1^J)$$

- resulting model: exploit first-order structure (or zero-order)
  training: backpropagation within EM algorithm

common properties in both approaches:
– bi-directional LSTM RNN over input words f_j, $j = 1, ..., J$
– uni-directional LSTM RNN over output words e_i, $i = 1, ..., I$



- **direct HMM (finite-state model):**
  summing over probability models:

$$p(e_1^I|f_1^J) \;=\; \sum_{b_1^I} \prod_i p(e_i, b_i|b_{i-1}, e_0^{i-1}, f_1^J)$$

special case: zero-order HMM:

$$p(e_1^I|f_1^J) \;=\; \prod_i p(e_i|e_0^{i-1}, f_1^J) = \prod_i \sum_j p(e_i, j|e_0^{i-1}, f_1^J)$$
$$= \prod_i \sum_j p(j|e_0^{i-1}, f_1^J) \cdot p(e_i|j, e_0^{i-1}, f_1^J)$$

- **attention mechanism: averaging over internal RNN representations h_j:**

$$p(e_i|e_0^{i-1}, f_1^J) \;=\; p(e_i|e_{i-1}, s_{i-1}, c_i)$$
$$\textbf{with} \quad c_i \;=\; \sum_j p(j|e_0^{i-1}, f_1^J) \cdot h_j(f_1^J)$$

(Unpublished) Experimental Results

- WMT task: German $\rightarrow$ English:
  – training data: 6M sentence pairs = (137M, 144M) words
  – test data: (about) 3k sentence pairs = (64k, 67k) words

- WMT task: Chinese $\rightarrow$ English:
  – training data: 14M sentence pairs = (920M Chinese letters, 364M English words)
  – test data: (about) 2k sentence pairs = (153k Chinese letters, 71k English words)

- performance measures:
  – BLEU [%]: accuracy measure: "the higher, the better"
  – TER [%]: error measure: "the lower, the better"

- basic units for implementation:
  – BPE (*byte pair encoding*) units rather than full-form words
  – alphabet size: about 40k

- RWTH papers (with preliminary results):
  [Wang & Alkhouli$^+$ 17, Wang & Zhu$^+$ 18]

Neural Hidden Markov Model

- **LSTM-RNN based representations for input and output: 4 layers of encoder and 1 layer of decoder**
- **independent models of alignment and lexicon (no parameter sharing as in attention approach)**

HMM	German→English						Chinese→English					
	#Par	PPL	test2017		test2018		#Par	PPL	dev2017		test2017	
			BLEU	TER	BLEU	TER			BLEU	TER	BLEU	TER
zero-order	129M	5.29	30.9	57.4	37.4	48.9	125M	8.12	20.1	65.1	20.7	64.2
first-order	136M	4.64	31.6	56.5	38.7	48.4	138M	7.63	20.1	64.0	22.0	63.2

Comparison: Best Results

best results	German→English						Chinese→English					
	#Par	PPL	test2017		test2018		#Par	PPL	dev2017		test2017	
			BLEU	TER	BLEU	TER			BLEU	TER	BLEU	TER
LSTM-RNN att.	162M	4.57	32.1	56.3	38.8	48.1	156M	6.60	21.4	63.6	22.9	62.0
self-attention	101M	3.90	33.4	55.3	40.4	46.8	97M	5.34	21.8	62.9	23.5	60.1
neural HMM	179M	5.24	31.9	56.6	38.3	48.3	174M	7.41	20.8	63.2	22.4	61.4

conclusions about neural HMM:
– (nearly) competitive with LSTM-RNN attention approach
– some performance gap to self-attention approach (= Google's transformer)

– room for improvement of neural HMM (in MT and ASR, too!)

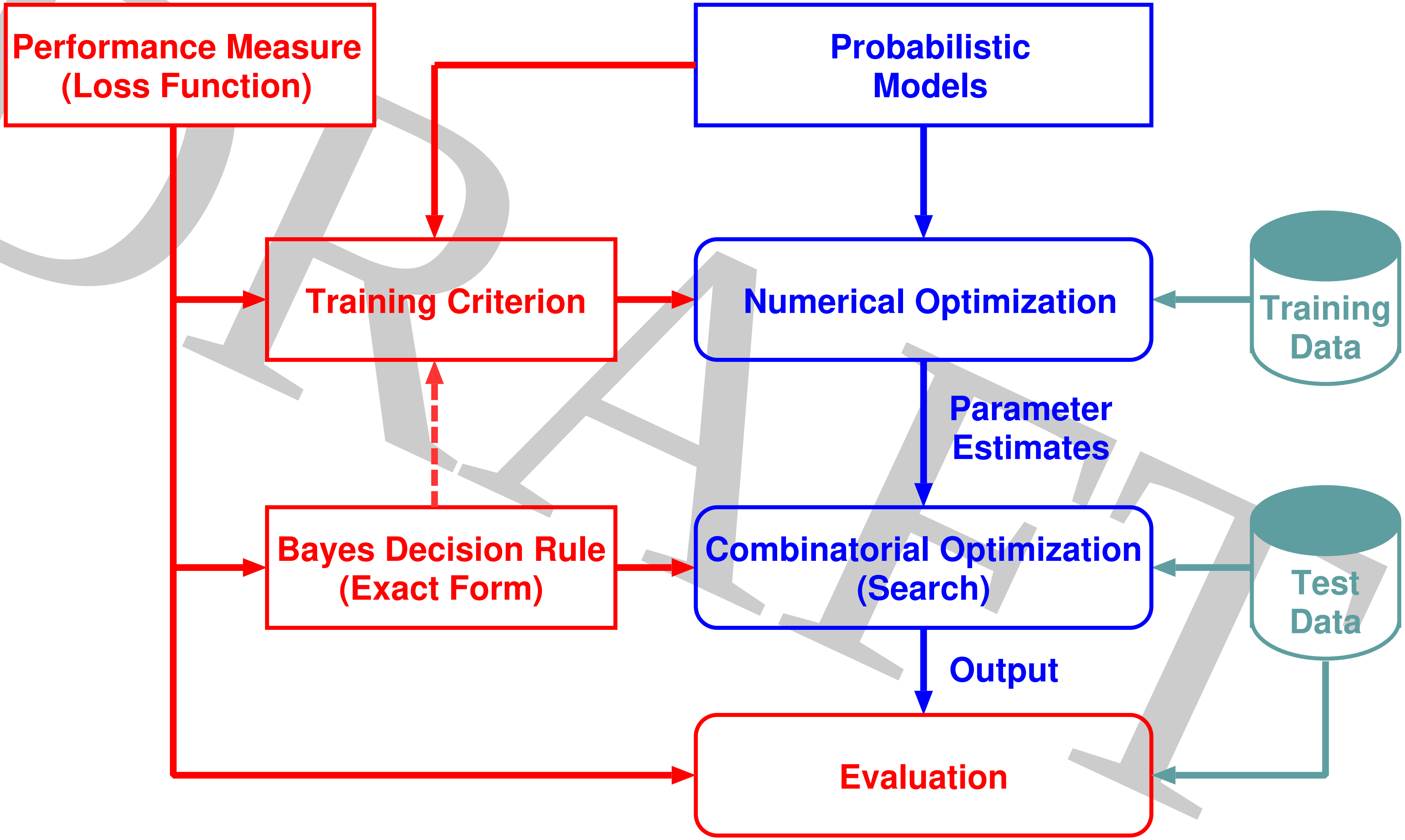
10 Summary and Conclusions

- **Bayes decision theory and statistical approach:**
  principal ingredients
  – choice of performance measure: errors at sequence, word, phoneme, frame level
  – probabilistic models at these levels and the interaction between these levels
  – training criterion along with an optimization algorithm
  – Bayes decision rule: search/decoder with an efficient implementation

- **deep learning:**
  – defines one family of probabilistic models within statistical approach
  – baseline structure: matrix-vector product + nonlinearities
  – yes, resulted in significant improvements

- **history of machine learning and statistical classification:**
  – there has been and will be life outside deep learning
  – we must get not only the principles, but also the details right

RWTH AACHEN UNIVERSITY



Challenges

- challenges for general machine learning:
  - mathematical optimization with huge complexity:
    we need a theoretical framework for practical aspects of gradient search
  - can we find ANNs with more explicit probabilistic structures?
  - novel structures beyond matrix-vector product + nonlinearties?
- challenges in ASR:
  - to continue the general improvements (ongoing for 40 years!)
  - task: informal colloquial speech (meetings)
  - robustness wrt acoustic conditions and language context
    (improved adaptation ?)
- unsupervised training for ASR:
  machine learning with (virtually) no labeled data?
- features for ASR beyond spectral analysis/Fourier transform:
  - recent work [Tüske & Golik 2014, Sainath et al. 2015, Baevski & Schneider^{+} 2020]
  - real and consistent improvements over spectral analysis?
- architecture for speech translation:
  challenge: handle three types of training data

Tasks in Human Language Technology:
Speech-to-Text Translation

SPEECH SIGNAL IN SOURCE LANGUAGE

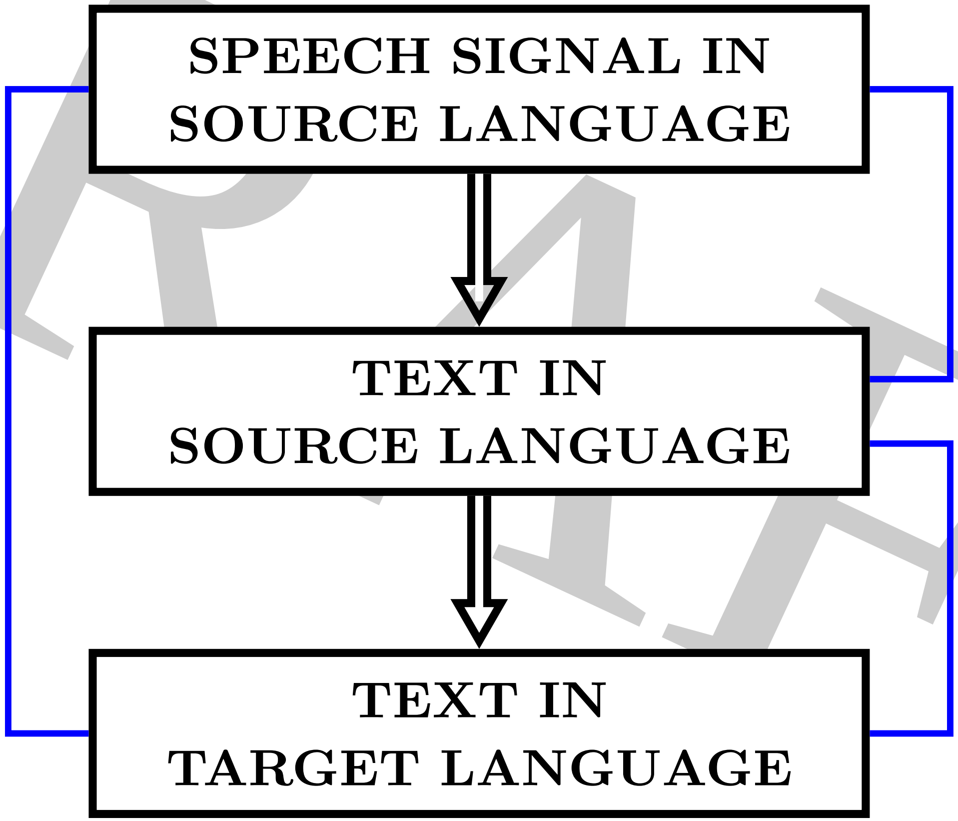
$\Downarrow$

TEXT IN SOURCE LANGUAGE

$\Downarrow$

TEXT IN TARGET LANGUAGE

Tasks in Human Language Technology: Speech-to-Text Translation



Conclusions (Jaca)

40 years of building operational systems for HLT:

- **success of data-driven vs. handcrafted rule-based approaches**
- **misconception: things started 40 years ago, not in 2013!**
- **persistent evolution of data-driven concepts:**
  - **signal-processing NLP: ASR and HTR**
  - **text-processing NLP:**
    - **language models for ASR (+ HTR + MT)**
    - **machine translation (MT)**
    - **large language models for NLU, e. g. Q&A, dialog management, ...**
- **statistical decision theory:**
  unifying framework for data-driven approach and machine learning:
  - **distinguish ingredients:**
    loss function, prob.model, training criterion along with numerical optimization
  - **includes as a special case: ANNs and deep learning**
  - **most useful framework after 40 years of NLP**

- **large-scale language models:**
  - **primary design goal: to generate smooth fluent text**
  - **approach: data, but no manual design or coding**
  - **dialog management: learned by data-driven approach (unlike manually designed dialog strategies)**
  - **(hopeful) by-product: semantic correctness ?**
- **LLMs are part of data-driven machine learning:**
  - **more data, more complex models, more computation**
  - **1989 R. Mercer/IBM: *There is no data like more data.***
- **specific success ('revolution'):**
  - **symbol embeddings/vectors in contrast to symbol count statistics along with operations in high-dim. vector space:**
  - **useful for areas beyond NLP? general concept for categorical statistics?**

Success of LLMs: Why?

where does the success/hype of LLMs come from?

- power of transformer architecture
  (and computer hardware!)
- huge amount of training data:
  – no annotation required!
  – straightforward training criterion: perplexity
- instruction/prompt along with input and output:
  everything in every-day language (unlike *formal* NLP tasks)
- in particular: success for dialog tasks:
  no explicit dialog strategy!

- unclear:
  relevance of supervised fine-tuning and reinforcement learning

What about the Future?

future: what time horizon: 3, 5, 10, 20 years?
e. g. difficult prediction: ANN around 1990

short-term horizon: low-hanging fruits
more data, more complex models, more parameters, more computation

long-term horizon: scientific challenges:
beyond more data, we need better mathematical frameworks:

- back-propagation search:
  beyond trial and error: better theory of numerical optimization
- present ANN structures
  – deep MLP, RNN, LSTM, self-embedding, transducer, transformer,...:
  – lack of principal mathematical justification:
    why are some structures better for modelling and learning?
- beyond ANN structures:
  – what about going beyond the present structures (matrix-vector product + nonlinearity)?
  – there is plenty of (data-driven) life outside and beyond deep learning!
    (but yes, it will be complex mathematical models)

What about the Future? (ctd)

- **word/symbol embeddings in symbolic processing (NLP):**
  – most important concept in lieu of count-based statistics
  – widely underrated in statistics of categorical data (and general NLP ?)

- **open research directions: beyond *supervised* machine learning:**
  strictly *unsupervised* machine learning,
  i. e. absolutely no parallel (input,output) pairs

END
BACK-UP SLIDES

Tasks in Human Language Technology: Speech-to-Speech Translation

SPEECH SIGNAL IN SOURCE LANGUAGE

$\Downarrow$

TEXT IN SOURCE LANGUAGE

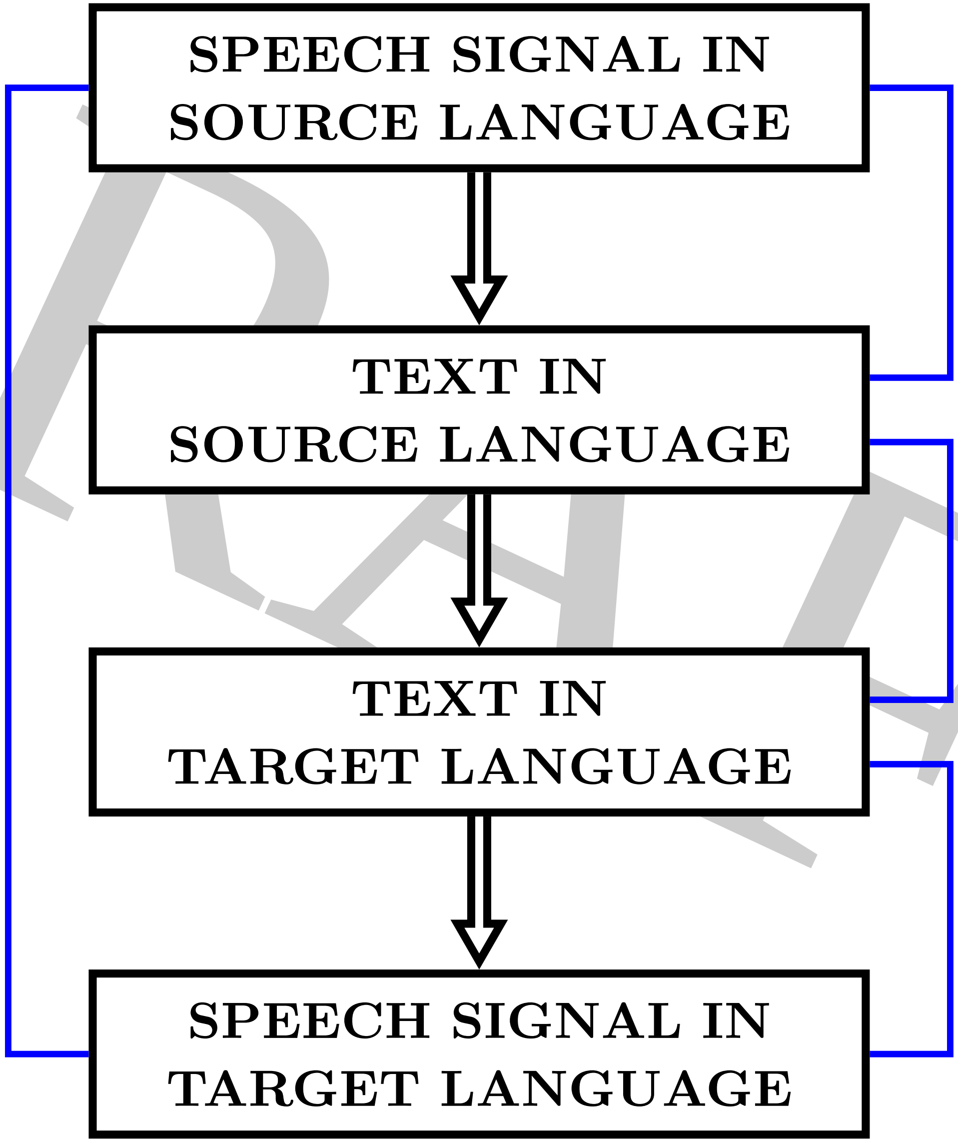
$\Downarrow$

TEXT IN TARGET LANGUAGE

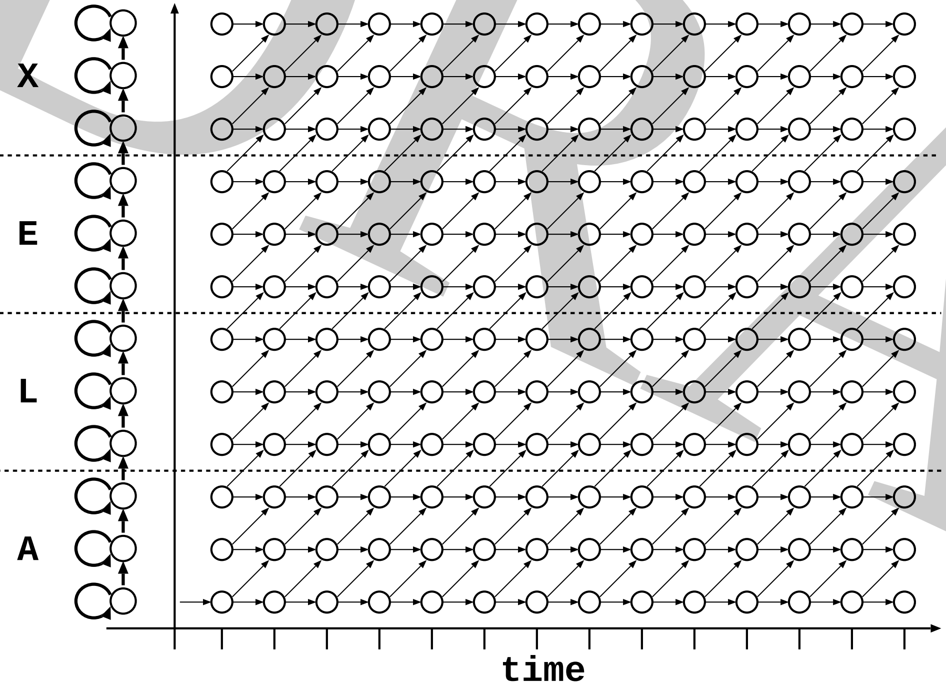
$\Downarrow$

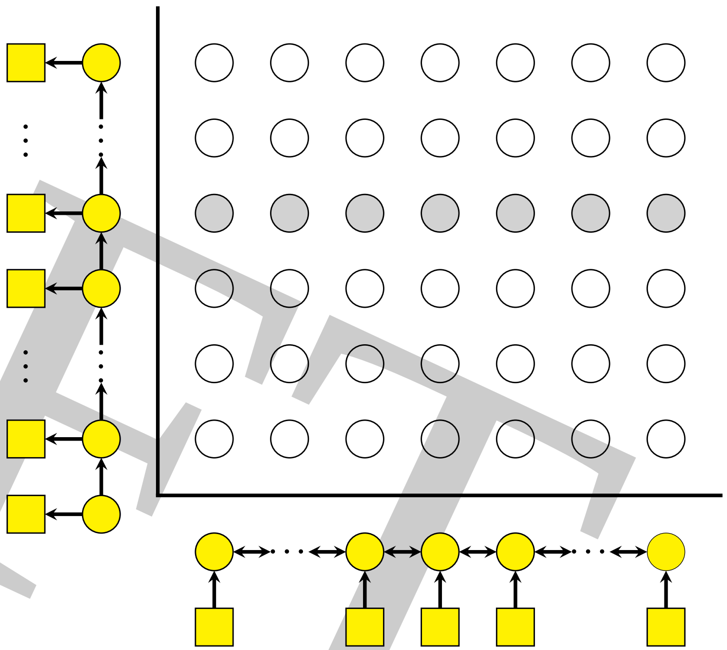
SPEECH SIGNAL IN TARGET LANGUAGE

Tasks in Human Language Technology: Speech-to-Speech Translation



Sequence-to-Sequence Processing: Direct HMM and Attention Model





History:

- **1989 [Nakamura & Shikano 89]:**
  English word category prediction based on neural networks.
- **1993 [Castano & Vidal^{+} 93]:**
  Inference of stochastic regular languages through simple recurrent networks
- **2000 [Bengio & Ducharme^{+} 00]:**
  A neural probabilistic language model
- **2007 [Schwenk 07]: Continuous space language models**
  2007 [Schwenk & Costa-jussa^{+} 07]: Smooth bilingual n-gram translation (!)
- **2010 [Mikolov & Karafiat^{+} 10]:**
  Recurrent neural network based language model
- **2012 RWTH Aachen [Sundermeyer & Schlüter^{+} 12]:**
  LSTM recurrent neural networks for language modeling

today: ANNs in language show competitive results.

History of NN based approaches to MT:

- **1997 [Neco & Forcada 97]:**
  asynchronous translations with recurrent neural nets
- **1997 [Castano & Casacuberta 97, Castano & Casacuberta^{+} 97]:**
  machine translation using neural networks and finite-state models
- **2007 [Schwenk & Costa-jussa^{+} 07]:**
  smooth bilingual n-gram translation
- **2012 [Le & Allauzen^{+} 12, Schwenk 12]:**
  continuous space translation models with neural networks
- **2014 [Devlin & Zbib^{+} 14]:**
  fast and robust neural networks for SMT
- **2014 [Sundermeyer & Alkhouli^{+} 14]:**
  recurrent bi-directional LSTM RNN for SMT
- **2015 [Bahdanau & Cho^{+} 15]:**
  joint learning to align and translate

11 APPENDIX: Formal Proofs: Bayes Decision Rule

11.1 Metric Loss Function: 50%-Rule

general form of (pseudo) Bayes decision rule:

$$x \to c_*(x) \;=\; \arg\min_c \{L[c|x]\}$$

$$\text{with} \quad L[c|x] := \sum_{\tilde{c}} p(\tilde{c}|x)\; L[\tilde{c}, c]$$

with $p(c|x)$ being either the true distribution $pr(c|x)$ or a normalized model $p_\vartheta(c|x)$
(note: purely mathematical statements about equivalence)

Basic Inequality for Metric Loss Function

expected loss:

$$L[c|x] := \sum_{\tilde{c}} p(\tilde{c}|x)\; L[\tilde{c}, c]$$

consider difference in expected loss
for two classes $\hat{c}$ and $\tilde{c}$ with an observation x:

$$\begin{aligned}
L[\hat{c}|x] - L[\tilde{c}|x] &= \sum_{c} p(c|x)\; \big(L[c,\hat{c}] - L[c,\tilde{c}]\big) \\
&= p(\hat{c}|x)\; \big(L[\hat{c},\hat{c}] - L[\hat{c},\tilde{c}]\big) \;+\; \sum_{c \neq \hat{c}} p(c|x)\; \big(L[\hat{c},c] - L[\tilde{c},c]\big)
\end{aligned}$$

identity/reflexivity: $L[\hat{c},\hat{c}] = 0$
triangle inequality: $L[\hat{c},c] - L[\tilde{c},c] \leq L[\hat{c},\tilde{c}]$

$$\begin{aligned}
&\leq\; -p(\hat{c}|x)\; L[\hat{c},\tilde{c}] \;+\; [1 - p(\hat{c}|x)]\; L[\hat{c},\tilde{c}] \\
&=\; [1 - 2\,p(\hat{c}|x)]\; L[\hat{c},\tilde{c}]
\end{aligned}$$

$$L[\hat{c}|x] \;\leq\; L[\tilde{c}|x] \;+\; [1 - 2\,p(\hat{c}|x)]\; L[\hat{c},\tilde{c}]$$

For any pair of classes $\hat{c}$ and $\tilde{c}$, we have shown:

$$L[\hat{c}|x \leq L[\tilde{c}|x] \; + \; [1 - 2\, p(\hat{c}|x)] \; L[\hat{c}, \tilde{c}]$$

analysis:

- assumption: $p(\hat{c}|x) \geq 0.5$:

  hence $[1 - 2\, p(\hat{c}|x)] \; L[\hat{c}, \tilde{c}] \leq 0$ and

  $$L[\hat{c}|x] \leq L[\tilde{c}|x] \quad \text{for all } \tilde{c}$$

- conclusion: consider the decision rule for 0/1 loss function:

  $$x \to \hat{c}_x = \operatorname*{argmax}_c p(c|x)$$

  if $p(\hat{c}_x|x) > 0.5$, then this string is the minimizing string in the the Bayes decision rule with any metric loss function more details in literature: [Schlüter & Scharrenbach$^+$ 05]

experiments on 5k-WSJ recognitions (740 sentences = 12137 words)
with 0/1 loss rule $c_0(x)$ and edit distance rule $c_*(x)$:

case distinction	sentences [%]	WER [%]
A: theory: 50% threshold	54	1.1
B: theory: extended 50% threshold	8	3.6
C: experiment: $c_0(x) = c_*(x)$	31	6.6
D: experiment: $c_0(x) \neq c_*(x)$	7	11.8 $\rightarrow$ 10.6
all cases	100	4.0 $\rightarrow$ 3.9

experimental conditions:

- a trained model $p(c|x)$ is used rather than true distribution
- use of N-best lists: $N = 10\,000$
- use of scaling exponent for language model

experimental result: 0/1 and edit distance rules differ only for high error rates!

11.2 Pseudo Bayes Decision Rule: Mismatch Conditions and Training

Bayes decision rule and mismatch conditions:

- **optimality of Bayes decision rule:**
  proved only under the condition that we use the *true* distribution

- **real world:**
  the true distribution is replaced by a *model* distribution,
  whose parameters are learned from data.

- **mismatch condition:**
  o the model distribution is different from the true distribution
  o how does this mismactch effect the optimality of the Bayes decision rule?
  o can we train model for optimum performance/minimum classification error?

methodology:
– purely theoretical/mathematical
– bounds on classification error

central role of performance measure or classification error:

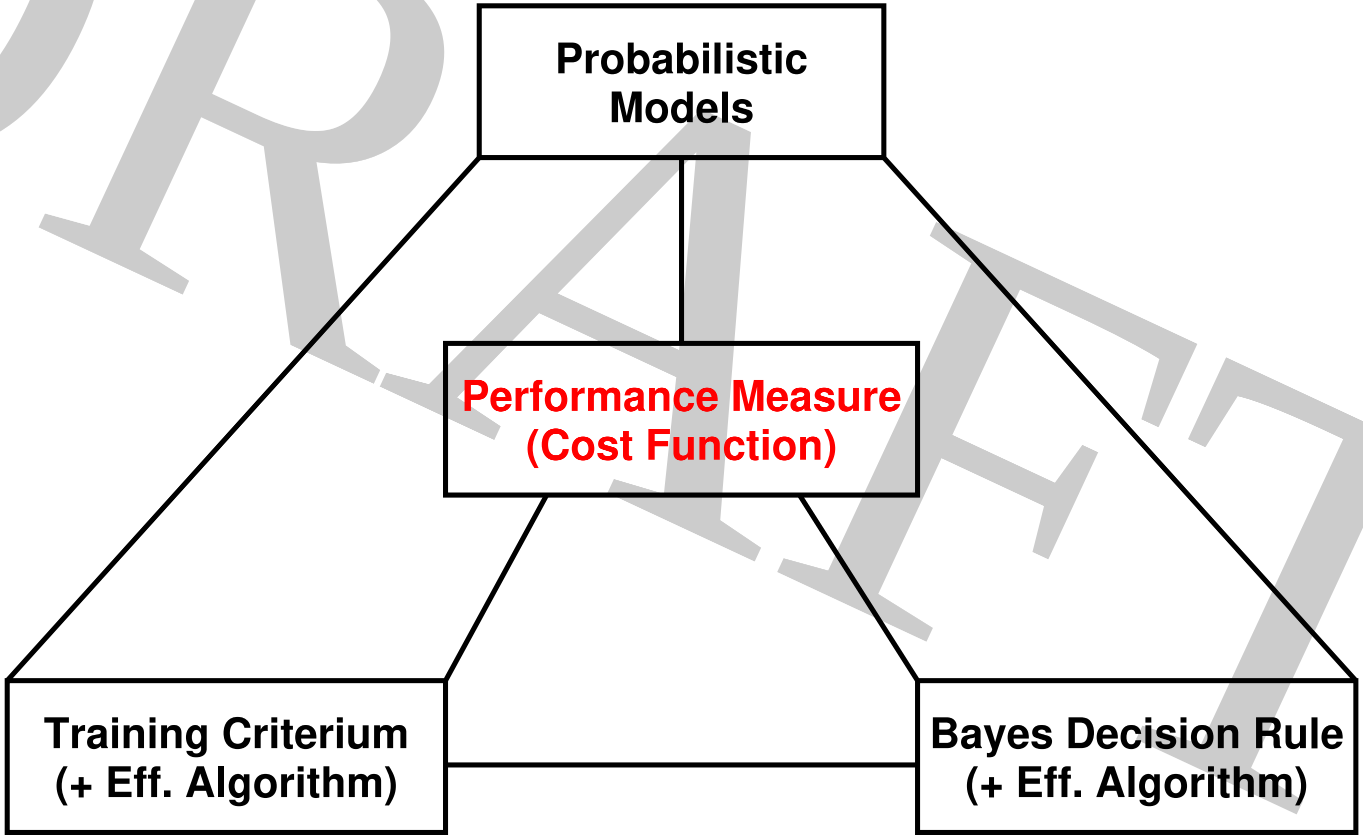
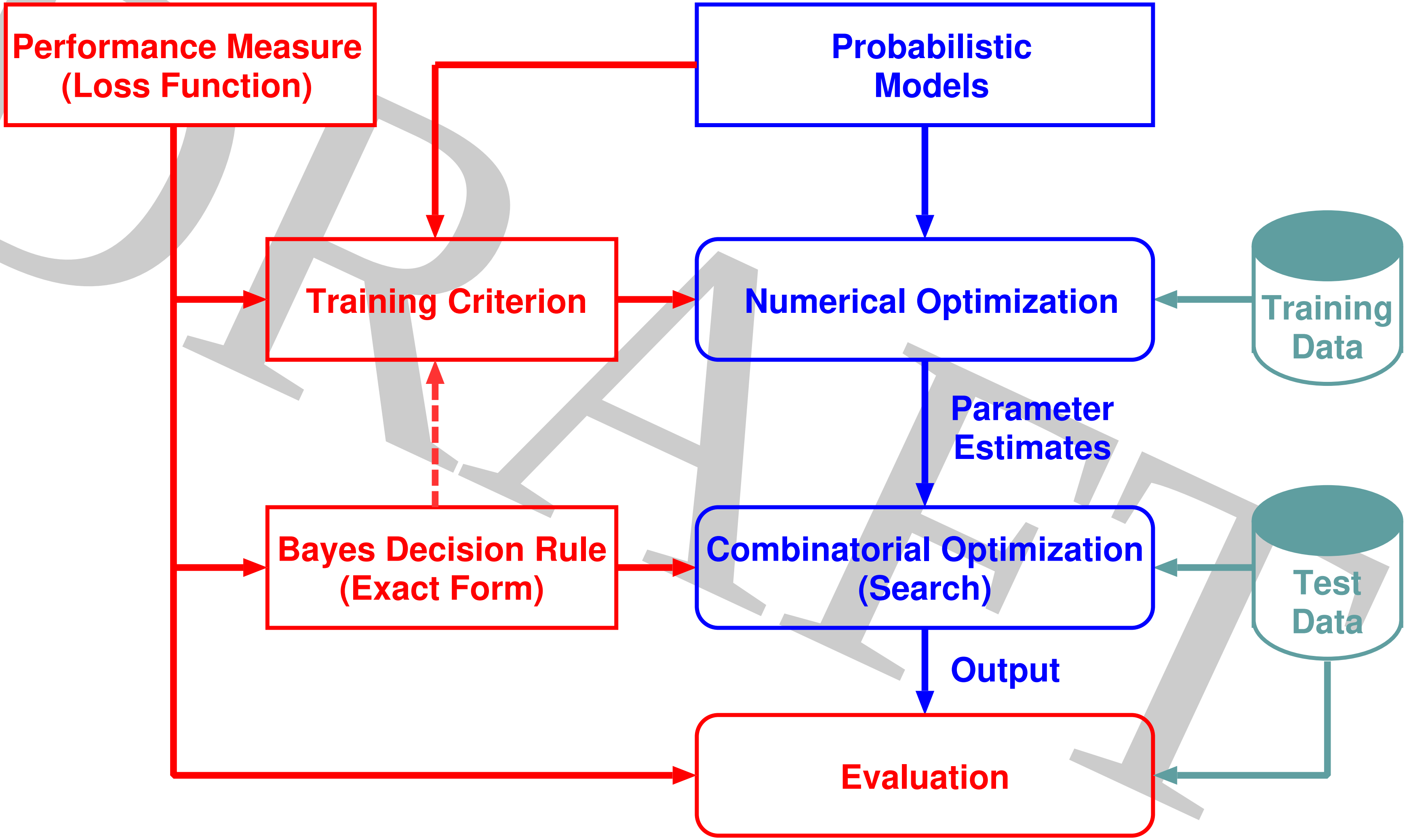


Illustration of the Statistical Approach



emphasis here:
– recognition performance: classification error for atomic outputs,
  i. e. single symbols or symbol strings as a whole
– relationship between classification error and training criteria

three contributions:

- we study the *model-based* classification error
  as opposed to the Bayes classification error
- we derive upper bounds for the *model-based* classification error
  more exact: the difference between *model-based* and Bayes classification errors
- we show that these bounds result in three practical training criteria
  (squared error, cross-entropy and binary cross-entropy):
  widely used, but relation to error rate is not known

Remotely Related Work

- bounds to Bayes classification error [Fukunaga 72, Duda & Hart 73]:
  – widely known in statistical classification (incl. information theory)
  – not useful in this context
- bounds to *model-based* classification error [Devroye & Györfi^{+} 96]:
  limited to two-class recognition tasks
- Vapnik's framework of empirical risk minimization [Vapnik 98]:
  emphasis on statistical variability across training samples
- specific area of ASR: [Printz & Olsen 01; Klakow & Peters 02; ...]

machine learning/pattern recognition:
most important goal: minimum classification error and nothing else

- **present situation: hodge-podge of TRAINING CRITERIA:**
  - **maximum likelihood and Bayes estimation**
  - **conditional likelihood**
  - **maximum mutual information (information theory)**
  - **cross entropy**
  - **minimum squared error**
  - **empirical (smoothed) error rate**
  - **decision trees (CART): Gini index and (Shannon) entropy**
  - **error-related criteria for specific tasks:**
    - **linear discriminant analysis (e.g. 'volume' of scatter matrices)**
    - **feature selection and extraction**
    - **optimum margin for class separation**
  - **...**

- **missing: consistent framework for the links to classification error**
  - **what is the relation among these criteria?**
  - **when should what criterion be used?**
  - **what is the relation with the classification error?**
- **why should such an analysis be helpful?**
  – today's systems are very complex
  – many shortcuts/approximations whose effects are hard to understand
  – mathematical model and analysis: clear concepts, no side effects

topics of this lecture:
conditions and aspects related to *optimal performance* according to Bayes decision rule

- **Bayes decision rule: exact form:**
  e.g. sentence/string error vs. word error in ASR
- **unknwon true distribution:**
  - **mismatch conditions: true distribution $pr(c|x)$ vs. model distribution $p_\vartheta(c|x)$**
  - **question: do we loose the optimality by these mismatch conditions?**
- **training criteria:**
  - **upper bounds for the difference between model and Bayes classification error**
  - **derived from the above upper bounds**
  - **relation between different training criteria**
- **questions:**
  - **do we need the true distribution for optimal performance? (necessary condition?)**
  - **do we need probability models at all? (as opposed to discriminant functions)**

interpretation: a mathematically correct justification of the probabilistic approach

Decision Rule under Mismatch Conditions

model-based decision rule $c_\vartheta(\cdot)$:

$$x \;\to\; c_\vartheta(x) := \arg\max_c \Big\{ p_\vartheta(c,x) \Big\}$$

For the error bounds, we will distinguish three types of model outputs (as for the training criterion of ANNs):

- **unconstrained output:**

$$p_\vartheta(c,x) \in \mathbb{R}$$

- **constrained output:**

$$p_\vartheta(c,x) \in [0,1]$$

- **normalized output:**

$$p_\vartheta(c|x) \in [0,1] \qquad \text{and} \qquad \sum_c p_\vartheta(c|x) = 1$$

Example: Three Types of Model Outputs

example:

- **unconstrained output: scoring function for $x \in \mathbb{R}^D$**

$$g_\vartheta(c, x) \;=\; \alpha_c + \lambda_c^T x + x \Lambda_c x^T$$

note: quadratic in x and linear in parameters $\vartheta = \{\alpha_c \in \mathbb{R}, \lambda_c \in \mathbb{R}^D, \Lambda_c \in \mathbb{R}^{D \cdot D}\}$

- **constrained output: logistic regression (ANN: sigmoid function):**

$$p_\vartheta(c, x) \;=\; \frac{1}{1 + \exp\big[-g_\vartheta(c, x)\big]}$$

- **normalized output: log-linear model (ANN: softmax):**

$$p_\vartheta(c|x) \;=\; \frac{\exp\big[g_\vartheta(c, x)\big]}{\sum_{c'} \exp\big[g_\vartheta(c', x)\big]}$$

this example: the decision output does not change:

$$x \to c_\vartheta(x) = \operatorname*{argmax}_c g_\vartheta(c, x) = \operatorname*{argmax}_c p_\vartheta(c, x) = \operatorname*{argmax}_c p_\vartheta(c|x)$$

examples of typical model distributions:

- **artificial neural net (or discriminant function)**
- **non-parametric model for discrete x**
- **maximum entropy or log-linear model**
- **decision tree (CART) approach**
- **classical approach (generative model, noisy-channel model):**

$$p_\vartheta(c|x) = \frac{p_\vartheta(c) \cdot p_\vartheta(x|c)}{\sum_{c'} p_\vartheta(c') \cdot p_\vartheta(x|c')}$$

  with class priors $p_\vartheta(c)$ and class-conditional model $p_\vartheta(x|c)$, e.g. single Gaussian distribution or Gaussian mixture

- **Hidden Markov Model (in the classical approach) for an observation string $x = x_1^T$**

idea of 'true' Bayesian modelling:

- **model:** $p(y|\lambda)$ **with** $y = (x, c)$ **and unknown parameter** λ
- **labelled training data:** $Y = \{(x_n, c_n) : n = 1, ..., N\}$
- **prior distribution for** λ

formalism: compute predictive distribution

$$p(y|Y) \;=\; \int d\lambda \; p(y|\lambda) \, p(\lambda|Y) \qquad p(\lambda|Y) \;=\; \frac{p(\lambda) \, p(Y|\lambda)}{\int d\lambda \; p(\lambda) \, p(Y|\lambda)}$$

theoretical advantage (?): no training problem

in practice: $p_\vartheta(y|Y)$
with other-type of unknown parameters ϑ**:**

- **what prior distribution** $p(y|\lambda)$
  and what hyperparameters?
- **what type of model** $p(y|\lambda)$ **?**
  (Bayesian term: model selection)

key questions about (classification) error:

- how does the mismatch between the true distribution $pr(c, x)$ and the model distribution $p_\vartheta(c, x)$ affect our approach?
- if the model $p_\vartheta(c, x)$ approximates the true distribution $pr(c, x)$: does the model error approximate the Bayes error? If yes, how tight?

partially published in 2003 [Ney 03]

this approach:

- upper and tight bound on the difference between model and Bayes classification error
- bound is a smooth convex function of the model
- bound can be re-written as a practical discriminative training criterion

Classification Errors: Two Types

decision rules:

$$\begin{aligned} \textbf{model:} \quad c_\vartheta(x) &= \arg\max_c \{p_\vartheta(c,x)\} \\ \textbf{Bayes:} \quad c_*(x) &= \arg\max_c \{pr(c|x)\} \end{aligned}$$

classification errors:

$$\begin{aligned} \textbf{model:} \quad E_\vartheta &= \sum_x pr(x) \sum_c pr(c \neq c_\vartheta(x)|x) &= \sum_x pr(x) \Big[1 - pr(c_\vartheta(x)|x)\Big] \\ \textbf{Bayes:} \quad E_* &= \sum_x pr(x) \sum_c pr(c \neq c_*(x)|x) &= \sum_x pr(x) \Big[1 - pr(c_*(x)|x)\Big] \\ & &= \sum_x pr(x) \Big[1 - \max_c pr(c|x)\Big] \end{aligned}$$

consider the difference in classification error:

$$\begin{aligned} E_\vartheta - E_* &= \sum_x pr(x) \Big[pr(c_*(x)|x) - pr(c_\vartheta(x)|x)\Big] \\ &= \sum_x pr(x) \Big[E_\vartheta(x) - E_*(x)\Big] \end{aligned}$$

using the local classification errors $E_\vartheta(x)$ and $E_*(x)$ in point x

consider difference of local classification errors in point x:

$$
\begin{aligned}
E_\vartheta(x) - E_*(x) &= pr(c_*(x)|x) - pr(c_\vartheta(x)|x) \\
& \qquad \textbf{use: } \; p_\vartheta(c,x) \le p_\vartheta(c_\vartheta(x),x) \\
&\le pr(c_*(x)|x) - pr(c_\vartheta(x)|x) + p_\vartheta(c_\vartheta(x),x) - p_\vartheta(c_*(x),x) \\
&= \Big[pr(c_*(x)|x) - p_\vartheta(c_*(x),x)\Big] + \Big[p_\vartheta(c_\vartheta(x),x) - pr(c_\vartheta(x)|x)\Big] \\
& \qquad \textbf{apply: } u \le |u| \\
&\le \Big|pr(c_*(x)|x) - p_\vartheta(c_*(x),x)\Big| + \Big|pr(c_\vartheta(x)|x) - p_\vartheta(c_\vartheta(x),x)\Big| \\
&\le \sum_c \Big|pr(c|x) - p_\vartheta(c,x)\Big|
\end{aligned}
$$

last step: all remaining classes are included to arrive at l_1 norm

l_1 **bound:**

$$\boldsymbol{E_\vartheta(x) - E_*(x) \;\leq\; \sum_c |pr(c|x) - p_\vartheta(c,x)|}$$

l_∞ **(or maximum) bound:**

$$\boldsymbol{E_\vartheta(x) - E_*(x) \;\leq\; 2 \cdot \max_c \left\{ |pr(c|x) - p_\vartheta(c,x)| \right\}}$$

l_2 **bound (follows from** l_∞**):**

$$\boldsymbol{E_\vartheta(x) - E_*(x) \;\leq\; 2 \cdot \sqrt{\sum_c \left[pr(c|x) - p_\vartheta(c,x)\right]^2}}$$

note: these local bounds for x are TIGHT in the following sense:

if	$p_\vartheta(c,x) \to pr(c|x)$	**then**	**Bound** $\to 0$
		and	$E_\vartheta(x) \to E_*(x)$

l_1 **bound: consider the difference between the local error rates:**

$$E_\vartheta(x) - E_*(x) \;\leq\; \sum_c \Big| pr(c|x) - p_\vartheta(c,x) \Big|$$

two operations for moving from local to global bounds:

- **summation over x using $pr(x)$:**

$$E_\vartheta - E_* \;\leq\; \sum_x pr(x) \sum_c \Big| pr(c|x) - p_\vartheta(c,x) \Big|$$

- **squaring and using the variance inequality:**

$$\Big(E_\vartheta - E_*\Big)^2 \;\leq\; \Big(\sum_x pr(x) \sum_c \Big| pr(c|x) - p_\vartheta(c,x) \Big|\Big)^2 \;\leq\; \sum_x pr(x) \Big(\sum_c \Big| pr(c|x) - p_\vartheta(c,x) \Big|\Big)^2$$

l_2 **bound: similar procedure**

result: all global bounds apply to the SQUARE of the difference in classification error

identity for squared error (normalized p_c, arbitrary q_c!) [Patterson & Womack 66, Bourlard & Wellekens 89]:

$$\sum_c [p_c - q_c]^2 \;=\; \sum_c p_c \sum_{c'} [q_{c'} - \delta_{c'c}]^2 \;-\; \Big(1 - \sum_c p_c^2\Big)$$

re-write the l_2 bound (using variance inequality):

$$E_\vartheta(x) - E_*(x) \;\leq\; 2 \cdot \sqrt{\sum_c \big[pr(c|x) - p_\vartheta(c,x)\big]^2}$$

$$\begin{aligned}
\big(E_\vartheta - E_*\big)^2 \;&\leq\; 4 \sum_x pr(x) \sum_c \big[pr(c|x) - p_\vartheta(c,x)\big]^2 \\
&=\; 4 \sum_x pr(x) \Big(\sum_c pr(c|x) \sum_{c'} \big[p_\vartheta(c',x) - \delta(c',c)\big]^2 \;-\; \big[1 - \sum_c pr^2(c|x)\big] \Big) \\
&=\; 4 \Big(\sum_{x,c} pr(x,c) \sum_{c'} \big[p_\vartheta(c',x) - \delta(c',c)\big]^2 - \sum_x pr(x) \big[1 - \sum_c pr^2(c|x)\big] \Big)
\end{aligned}$$

practical relevance: we switched from true distribution to ideal targets

summary of re-writing:

$$\big(E_\vartheta - E_*\big)^2 \;\le\; 4\,\Big(\sum_{x,c} pr(x,c) \sum_{c'} \big[p_\vartheta(c',x) - \delta(c',c)\big]^2 - \sum_{x} pr(x)\big[1 - \sum_{c} pr^2(c|x)\big]\Big)$$

training criterion for minimum classification error: minimize the upper bound

only the left term of the bound depends on ϑ:

$$F(\vartheta) \;:=\; \sum_{x,c} pr(c,x) \sum_{c'} \big[p_\vartheta(c',x) - \delta(c',c)\big]^2$$

which is the well-known squared error criterion.

For training data $(x_n, c_n), n = 1, ..., N$, we use the empirical average:

$$\begin{aligned}
\hat{\vartheta} \;&:=\; \arg\min_{\vartheta}\{F(\vartheta)\} \\
&=\; \arg\min_{\vartheta}\Big\{ \sum_{n=1}^{N} \sum_{c} \big[p_\vartheta(c,x_n) - \delta(c,c_n)\big]^2
\end{aligned}$$

inequality [Cover & Thomas 91, pp. 300] for $0 \leq p_c, q_c \leq 1$:

$$2 \cdot [p_c - q_c]^2 \;\leq\; p_c \log \frac{p_c}{q_c} \;+\; (1 - p_c) \log \frac{1 - p_c}{1 - q_c}$$

right-hand side: binary divergence or binary relative entropy

re-write the squared error bound:

$$\begin{aligned}
\Big(E_\vartheta - E_*\Big)^2 \;&\leq\; 4 \sum_x pr(x) \sum_c \Big[pr(c|x) - p_\vartheta(c,x)\Big]^2 \\
&\leq\; 2 \sum_x pr(x) \sum_c \Big(pr(c|x) \;\log \frac{pr(c|x}{p_\vartheta(c,x)} + [1 - pr(c|x)] \;\log \frac{1 - pr(c|x)}{1 - p_\vartheta(c,x)}\Big) \\
&=\; 2 \sum_x pr(x) \sum_c pr(c|x) \sum_{c'} \log \frac{1 - |\delta(c',c) - pr(c'|x)|}{1 - |\delta(c',c) - p_\vartheta(c',x)|} \\
&=\; 2 \sum_{x,c} pr(x,c) \sum_{c'} \log \frac{1 - |\delta(c',c) - pr(c'|x)|}{1 - |\delta(c',c) - p_\vartheta(c',x)|}
\end{aligned}$$

summary of re-writing:

$$\left(E_\vartheta - E_*\right)^2 \;\leq\; 2 \sum_{x,c} pr(x,c) \sum_{c'} \log \frac{1 - |\delta(c',c) - pr(c'|x)|}{1 - |\delta(c',c) - p_\vartheta(c',x)|}$$

training criterion for minimum classification error: minimize the upper bound

only the denominator of the argument of the logarithm depends on ϑ:

$$\begin{aligned} F(\vartheta) &= \sum_{x,c} pr(x,c) \sum_{c'} \log\left[1 - |\delta(c',c) - p_\vartheta(c',x)|\right] \\ &= \sum_{x,c} pr(x,c) \Big(\log p_\vartheta(c,x) + \sum_{c' \neq c} \log\left[1 - p_\vartheta(c',x)|\right] \Big) \end{aligned}$$

which is the so-called binary cross-entropy criterion [Solla & Levin[+] 88].

For training data $(x_n, c_n), n = 1, ..., N$, we use the empirical average:

$$\begin{aligned} \hat{\vartheta} &:= \arg\max_{\vartheta} \{F(\vartheta)\} \\ &= \arg\max_{\vartheta} \Big\{ \sum_n \Big(\log p_\vartheta(c_n, x_n) + \sum_{c \neq c_n} \log\left[1 - p_\vartheta(c, x_n)\right]\Big) \Big\} \end{aligned}$$

Pinsker inequality for two distributions (normalized!) p_c and q_c
[Fedotov & Harremoes$^+$ 03], [Cover & Thomas 91, pp. 300]:

$$\Big(\sum_c |p_c - q_c| \Big)^2 \; \le \; 2 \cdot \sum_c p_c \, \log \frac{p_c}{q_c}$$

right-hand side: the Kullback-Leibler divergence (or relative entropy)

re-write l_1 bound (using variance inequality):

$$\begin{aligned}
E_\vartheta(x) - E_*(x) \; &\le \; \sum_c |pr(c|x) - p_\vartheta(c|x)| \\
\Big(E_\vartheta - E_* \Big)^2 \; &\le \; \sum_x pr(x) \Big(\sum_c |pr(c|x) - p_\vartheta(c|x)| \Big)^2 \\
&\le \; 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \, \log \frac{pr(c|x)}{p_\vartheta(c|x)} \\
&= \; 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \Big[\log pr(c|x) - \log p_\vartheta(c|x) \Big]
\end{aligned}$$

summary of re-writing:

$$\Big(E_\vartheta - E_*\Big)^2 \le 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \Big[\log pr(c|x) - \log p_\vartheta(c|x)\Big]$$

training criterion for minimum classification error: minimize the upper bound

only the right term depends on ϑ:

$$F(\vartheta) := \sum_x \sum_c pr(x,c) \log p_\vartheta(c|x)$$

which is the so-called cross-entropy or log-probability criterion.

For training data $(x_n, c_n), n = 1, ..., N$, we use the empirical average:

$$\begin{aligned} \hat{\vartheta} &:= \arg\max_\vartheta \{F(\vartheta)\} \\ &= \arg\max_\vartheta \Big\{ \sum_n \log p_\vartheta(c_n|x_n) \Big\} \end{aligned}$$

criterion: log class posterior probability:

- **links to information theory:**
  maximum mutual information and equivocation
- **criterion used in *discriminative training*, e.g. log-linear modelling:**

$$\max_{\vartheta} \left\{ \sum_{n=1}^{N} \log \, p_{\vartheta}(c_n|x_n) \right\}$$

as opposed to conventional maximum likelihood:

$$\max_{\vartheta} \left\{ \sum_{n=1}^{N} \log \, p_{\vartheta}(x_n|c_n) \right\}$$

- **solution for fully saturated model (with counts $N(c, x)$):**

$$\hat{p}_{\vartheta}(c|x) = pr(c|x) = N(c, x)/N(\cdot, x)$$

also used for decision trees (CART): (Shannon) entropy criterion

Conclusions

summary:

- goal: to study the model-based classification error
  – explicit distinction to Bayes classification error
  – tight bounds on model-based classification error
- three resulting bounds:
  – squared error: unconstrained model outputs
  – binary divergence: constrained model outputs
  – Kullback-Leibler distance: normalized model outputs

  associated training criteria are well known,
  but their relation to classification error was unknown

- applications:
  – discriminative training, neural nets and log-linear modelling
  – discriminative training in speech recognition (HMM)
  – splitting criteria in CART
  – ...

open issues:

- linear bounds in classification error E_ϑ:
  are there tight linear bounds rather than quadratic ones?
- estimation problem:
  statistical fluctuations from training sample to test sample?
- compound decisions, i.e. decisions in context:
  e.g. in speech recognition and natural language processing:
  how to separate the contribution of each system component (knowledge source) ?

Classification Error: Overview of Issues

questions and issues:

- **classical bounds on Bayes classification error**
- **mismatch condition:**
  - general case: true distribution and model
  - special case: effect of class prior model only?
- **Bayes-Bayes condition:**
  - two true distributions
  - what is the difference of the classification errors?
- **omitted/additional features:**
  how much does the classification error change?
- **classification in context:**
  - typical situation: STRING of class symbols
  - issue: string error vs. symbol error
  - how much does the context help to reduce the classification error?

Error Bounds: Summary

atomic decisions (or at string level):

- we have studied the model-based classification error
  – explicit distinction between Bayes and model error
  – tight bounds on the squared difference
- three resulting bounds and associated training criteria:
  – squared error: unconstrained model outputs
  – binary divergence: constrained model outputs
  – Kullback-Leibler divergence: normalized model outputs
- associated training criteria are well known,
  but their relation to classification error was unknown

open issues:
– how to go from discriminative training to generative training
– how to go from single symbols to symbol strings

11.3 Smoothed Error Count: MCE

MCE: minimum classification error

- **concept: learn the parameters of the model in such a way that the empirical error rate on the training data is minimized**
- **optimization criterion:**
  requires a smoothing of the classification error count:
  – weak rival class: error count $\rightarrow$ 0
  – strong rival class: error count $\rightarrow$ 1
- **optimization strategy: gradient search**
- **history:**
  – general principle well known in machine learning since 1970
  – used in ASR: [Juang & Katagiri 92, Juang & Chou^{+} 97]
- **practice:**
  – not widely used
  – replaced by methods called *expected loss* or *minimum Bayes risk*

consider a training sample (x_n, c_n) using non-negative model outputs $p_\vartheta(c, x)$:

$$\begin{aligned}
p_\vartheta(c_n, x_n) &\overset{?}{>} \max_{c \neq c_n} \{p_\vartheta(c, x_n)\} \\
p_\vartheta(c_n, x_n) &\overset{?}{>} \Big(\sum_{c \neq c_n} p_\vartheta^r(c, x_n)\Big)^{1/r} \\
p_\vartheta^r(c_n, x_n) &\overset{?}{>} \sum_{c \neq c_n} p_\vartheta^r(c, x_n) \\
\frac{p_\vartheta^r(c_n, x_n)}{\sum_{c \neq c_n} p_\vartheta^r(c, x_n)} &\overset{?}{>} 1
\end{aligned}$$

using this final comparison, we smooth the counts using an (inverted) sigmoid function:

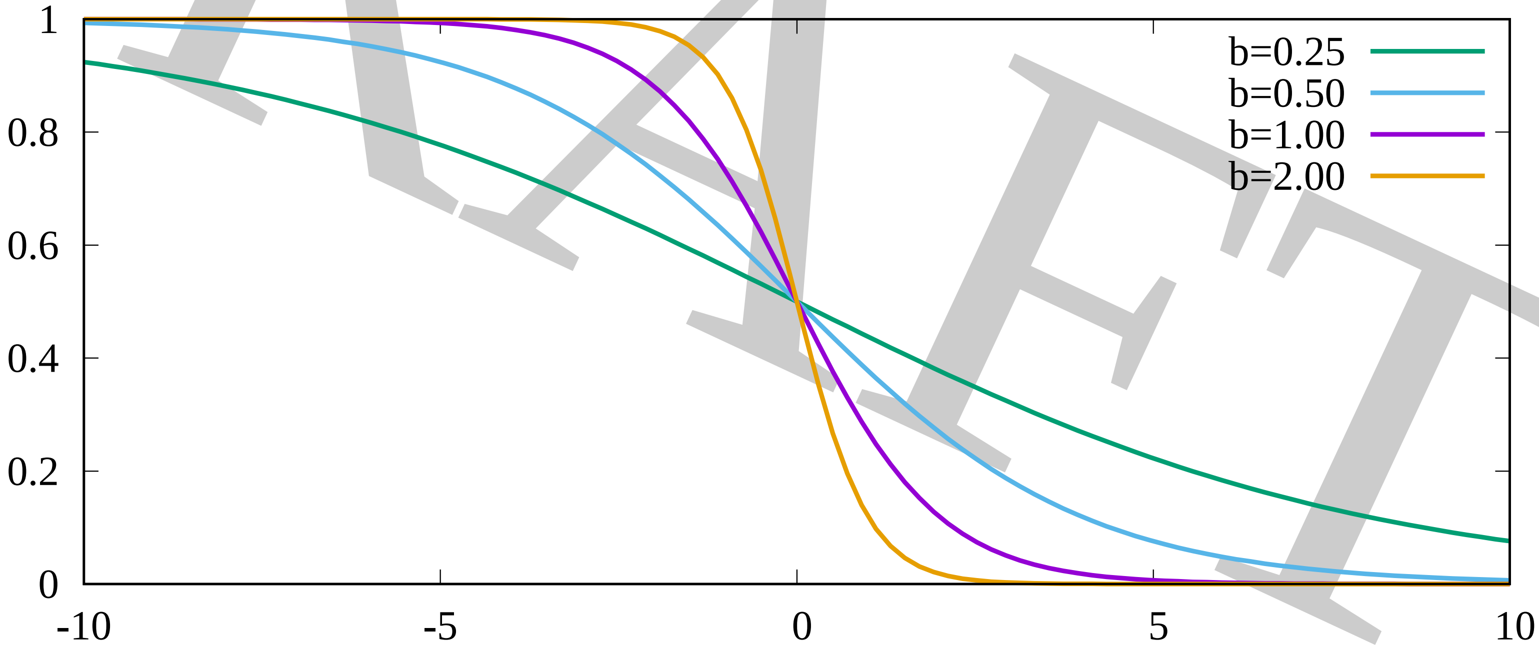
$$F(\vartheta) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{1 + \Big(\frac{p_\vartheta^r(c_n, x_n)}{\sum_{c \neq c_n} p_\vartheta^r(c, x_n)}\Big)^b}$$

– parameter $r > 1$: controls the sharpness of the model distribution $p_\vartheta(c, x)$
– parameter $b > 0$: controls the smoothness of the counting function $h[\cdot]$:

$$u \to h[u] := 1/[1 + \exp(b \cdot \log\, u)] = 1/[1 + u^b]$$

Plots: Smoothing the Classification Error Count

$$\log\, u \to h[u] := \frac{1}{1 + \exp(b \cdot \log\, u)} \qquad \textbf{with} \qquad u := \frac{p_\vartheta^r(c_n, x_n)}{\sum_{c \neq c_n} p_\vartheta^r(c, x_n)}$$



considerations:

- MCE: minimum classification error [Juang & Katagiri 92, Juang & Chou^{+} 97]
  – define smoothed classification error count and use it as training criterion
  – characteristic: explicit use of rival classes
- alternative approach
  [Hampshire & Pearlmutter 90], [Bishop 95a, pp. 245-247]:
  – question: *what is the optimal smoothing function?*
    optimality criterion: smoothed error count
  – in addition: we want to get back the class posterior distribution
    as optimal model
- this presentation: related, but slightly different concept:
  – define the smoothed error count as a function of
    the unknown smoothing function
  – analyze the resulting optimization problem and introduce necessary constraints:
    result = logarithmic scoring

(*pseudo*) Bayes decision rule using model $p_\vartheta(c,x)$:

$$x \to c_\vartheta(x) \;=\; \arg\max_c \left\{ p_\vartheta(c,x) \right\}$$

consider a training sample (x_n, c_n) using model outputs $p_\vartheta(c,x) \geq 0$:

$$p_\vartheta(c_n,x_n) \;\overset{?}{>}\; \max_{c \neq c_n} \Big\{ p_\vartheta(c,x_n) \Big\}$$

$$p_\vartheta(c_n,x_n) \;\overset{?}{>}\; \Big(\sum_{c \neq c_n} p_\vartheta^r(c,x_n) \Big)^{1/r}$$

with smoothing exponent r

$$p_\vartheta^r(c_n,x_n) \;\overset{?}{>}\; \sum_{c \neq c_n} p_\vartheta^r(c,x_n)$$

$$2\, p_\vartheta^r(c_n,x_n) \;\overset{?}{>}\; \sum_{c} p_\vartheta^r(c,x_n)$$

$$\frac{p_\vartheta^r(c_n,x_n)}{\sum_c p_\vartheta^r(c,x_n)} \;\overset{?}{>}\; \frac{1}{2}$$

result: rival classes are absorbed by the re-normalization and disappear!

Smoothed Error/Accuracy Count: Another Concept

- **define a new model with normalization**
  (note notation: symbol $q(c|x)$ includes parameters ϑ and r):

$$q(c|x) \;:=\; \frac{p_\vartheta^r(c,x)}{\sum_{c'} p_\vartheta^r(c',x)}$$

- **inverse error count = accuracy count:**
  count the correct decisions on training data $(x_n, c_n), n = 1, ..., N$
  (with the usual definition of the empirical distribution $pr(x,c)$)
  using a smooth monotonic counting function h (e.g. sigmoid function):

$$h : u \in \mathbb{R} \;\;\rightarrow\;\; h[u] \in \mathbb{R}$$

$$\begin{aligned} F(h,q) \;&=\; \frac{1}{N} \cdot \sum_{n=1}^{N} h\big[q(c_n|x_n)\big] \\ &=\; \sum_{x,c} pr(x,c) \cdot h\big[q(c|x)\big] \;=\; \sum_x pr(x) \sum_c pr(c|x) \cdot h\big[q(c|x)\big] \end{aligned}$$

- **question: what is an "optimal" count function $h[\cdot]$?**
  answer: analyze optimization problem and consider useful constraints

- **result: smoothed count with unknown count function $h[\cdot]$:**

$$F(h,q) \;:=\; \sum_x pr(x) \underbrace{\sum_c pr(c|x) \cdot h\big[q(c|x)\big]}_{:=\, F(h,q;x)}$$

- **result: criterion to be optimized over function $h[\cdot]$ in any point x:**

$$\begin{aligned} F(h,q;x) &= \sum_c pr(c|x) \cdot \log \exp\big(h\big[q(c|x)\big]\big) \\ &= \sum_c pr(c|x) \, \log \, pr(c|x) \;+\; \underbrace{\sum_c pr(c|x) \cdot \log \frac{\exp\big(h\big[q(c|x)\big]\big)}{pr(c|x)}}_{:=\, \Delta F(h,q;x) \,\le\, \log \sum_c \exp\left(h\left[q(c|x)\right]\right)} \end{aligned}$$

where the inequality is based on the log-sum inequality for $a_k, b_k > 0$:

$$\sum_k \frac{b_k}{\left(\sum_{k'} b_{k'}\right)} \, \log \frac{a_k}{b_k} \;\le\; \log \frac{\sum_k a_k}{\sum_k b_k}$$

$$\sum_{k'} b_{k'} = 1 : \qquad \sum_k b_k \, \log \frac{a_k}{b_k} \;\le\; \log \sum_k a_k$$

- upper bound (remember: accuracy count):

$$\Delta F(h,q;x) := \sum_c pr(c|x) \cdot \log \frac{\exp\left(h\left[q(c|x)\right]\right)}{pr(c|x)} \;\leq\; \log \sum_c \exp\left(h\left[q(c|x)\right]\right)$$

choose logarithm for $h(u)$: $\quad h(u) \;:=\; \log\, u$

- resulting effects:
  - upper bound is independent of x and of the model $\{q(c|x)\}$
  - thus we avoid the dependence of counting function $h(u)$ on model $\{q(c|x)\}$

$$\sum_c \exp\left(h\left[q(c|x)\right]\right) \;=\; \sum_c \exp\left(\log\,\left[q(c|x)\right]\right) \;=\; \sum_c q(c|x) = 1$$

$$\Delta F(h,q;x) \;=\; \sum_c pr(c|x) \cdot \log \frac{q(c|x)}{pr(c|x)} \;\leq\; \log\, 1 = 0$$

- optimal model $q(c|x)$ on training data with $pr(c|x)$:
  - criterion: maximimize the accuracy count over unknown model $q(c|x)$
  - optimal solution (using divergence inequality):

$$\hat{q}(c|x) \;=\; pr(c|x)$$

  (note: the upper bound is attained then)

11.4 Kullback-Leibler Bound: Refinements and Strings

- **maximum likelihood training:**
  clear result: cross-entropy must always be better
- **structured output: from single symbols to strings**
  – strings with synchronization: yes
  – strings without synchronization: no simple proof

ASR (and other NLP) systems:
generative approach with an explicit language model

generative approach:

$$\begin{aligned} \text{true distribution:} \quad pr(c,x) &= pr(c) \cdot pr(x|c) \\ \text{model distribution:} \quad q(c,x) &= q(c) \cdot q(x|c) \end{aligned}$$

with
– class prior or language distribution: $q(c)$ and $pr(c)$
– observation distribution: $q(x|c)$ and $pr(x|c)$
note: change in notation: model $q(c,x)$ rather than $p_\vartheta(c,x)$

we compute the 'class posterior' model by re-normalization:

$$\begin{aligned} q(x) &= \sum_c q(c,x) \\ q(c|x) &= \frac{q(c,x)}{q(x)} \end{aligned}$$

re-write (Kullback-Leibler) divergence bound:

$$\frac{1}{2} \cdot [E_* - E_q]^2 \leq \sum_{x,c} pr(c,x) \log \frac{pr(c|x)}{q(c|x)}$$

re-write $q(c|x)$: $\quad q(c|x) = q(c,x)/q(x) \quad q(x) := \sum_c q(c,x)$

and $pr(c|x)$ **similarly**

$$= \sum_{x,c} pr(c,x) \log \frac{pr(c,x)/pr(x)}{q(c,x)/q(x)}$$

$$= \underbrace{\sum_{x} pr(x) \log \frac{q(x)}{pr(x)}}_{\leq 0} + \underbrace{\sum_{x,c} pr(c,x) \log \frac{pr(c,x)}{q(c,x)}}_{\geq 0}$$

$$\leq \sum_{x,c} pr(c,x) \log \frac{pr(c,x)}{q(c,x)}$$

$$= \sum_{c} pr(c) \log \frac{pr(c)}{q(c)} + \sum_{x,c} pr(c,x) \log \frac{pr(x|c)}{q(x|c)}$$

summary of re-writing the (Kullback-Leibler) divergence bound:

$$\frac{1}{2}\cdot\left[E_* - E_q\right]^2 \le \sum_{x,c} pr(c,x)\,\log\frac{pr(c|x)}{q(c|x)}$$

$$\le \sum_{x,c} pr(c,x)\,\log\frac{pr(c,x)}{q(c,x)} = \sum_{c} pr(c)\,\log\frac{pr(c)}{q(c)} + \sum_{x,c} pr(c,x)\,\log\frac{pr(x|c)}{q(x|c)}$$

interpretations:

– first line: divergence of the posterior distributions $\rightarrow$ cross-entropy training

– second line: divergence of joint distribution $\rightarrow$ maximum-likelihood training

observations:

- **absolute minimum of upper bound: zero is attained if**
  – discriminative model: $\hat{q}(c|x) = pr(c|x)$
  – generative model: $\hat{q}(c,x) = pr(c,x)$
- **if absolute minimum is NOT attained:**
  – discriminative bound is better than generative bound
  – discriminative training (cross-entropy) is always better (in terms of classification error) than generative training (maximum likelihood)

generative models:

- **training criterion: joint probability (in most general case)**
  = maximum likelihood of prior model $p(c)$ and of class-conditional model $p(x|c)$
- **provides also an upper bound to the squared difference of the classification error**
- **but: is always worse than the discriminative criterion**

11.5 Symbol Strings: Models of Posterior Probability

From Single Symbols to Symbol Strings

model with 1:1 correspondence between class labels c_1^N and observations x_1^N (string length N is known):

observations:	x_1	x_2	...	x_{n-1}	x_n	x_{n+1}	...	x_{N-1}	x_N
	o	o	o	o	o	o	o	o	o
	|	|	|	|	|	|	|	|	|
	o	o	o	o	o	o	o	o	o
class labels:	c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N

typical problems:

– POS tagging (POS: parts of speech)

– frame labelling in ASR (incl. pronunciation and language models!)

– recognition problems with no problems of boundary detection: isolated words, printed character recognition, ...

we fix a position n in the string:

- we compute the marginal probability in position n
  from the model of string posterior probability $q(c_1^N|x_1^N)$:

$$q_n(c|x_1^N) = \sum_{c_1^N : c_n = c} q(c_1^N|x_1^N)$$

  similarly for the empirical distribution: $pr_n(c|x_1^N)$

- difference in classification error in position n: $\Delta E_{q,n}$

useful: log-sum inequality for non-negative numbers a_k and b_k

$$\left(\sum_k a_k\right) \log \frac{\sum_k a_k}{\sum_k b_k} \leq \sum_k a_k \log \frac{a_k}{b_k}$$

re-write Kullback-Leibler bound for (c, x_1^N) in position n:

$$\frac{1}{2} \cdot \Delta E_{q,n}^2 \leq \sum_{x_1^N} pr(x_1^N) \sum_c pr_n(c|x_1^N) \log \frac{pr_n(c|x_1^N)}{q_n(c|x_1^N)}$$

$$= \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N : c_n = c} pr(c_1^N|x_1^N) \log \frac{\sum_{c_1^N : c_n = c} pr(c_1^N|x_1^N)}{\sum_{c_1^N : c_n = c} q(c_1^N|x_1^N)}$$

(use log-sum inequality)

$$\leq \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N : c_n = c} pr(c_1^N|x_1^N) \log \frac{pr(c_1^N|x_1^N)}{q(c_1^N|x_1^N)}$$

$$= \sum_{x_1^N} pr(x_1^N) \sum_{c_1^N} pr(c_1^N|x_1^N) \log \frac{pr(c_1^N|x_1^N)}{q(c_1^N|x_1^N)}$$

important results:

– training at string level improves symbol level, too!

– training at symbol level is always better than at string level

12 References

[Baevski & Schneider+ 20] A. Baevski, S. Schneider, M. Auli: VQ-Wav2Vec: Self-Supervised Learning of Discrete Speech Representations. Facebook AI Research, Menlo Park, CA, arxiv, 16-Feb-2021.

[Bahdanau & Cho+ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural machine translation by jointly learning to align and translate. Int. Conf. on Learning and Representation (ICLR), San Diego, CA, May 2015.

[Bahl & Jelinek+ 83] L. R. Bahl, F. Jelinek, R. L. Mercer: A Maximum Likelihood Approach to Continuous Speech Recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 5, pp. 179-190, March 1983.

[Bahl & Brown+ 86] L. R. Bahl, P. F. Brown, P. V. de Souza, R. L. Mercer: Maximum mutual information estimation of hidden Markov parameters for speech recognition. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Tokyo, pp.49-52, April 1986.

[Beck & Schlüter+ 15] E. Beck, R. Schlüter, H. Ney: Error Bounds for Context Reduction and Feature Omission, Interspeech, Dresden, Germany, Sep. 2015.

[Bang & Cahyawijaya+ 23] Y. Bang, S. Cahyawijaya, N. Lee et al.: A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. HKUST, arxiv, 28-Feb-2023.

[Bengio & Ducharme+ 00] Y. Bengio, R. Ducharme, P. Vincent: A neural probabilistic language model. Advances in Neural Information Processing Systems (NIPS), pp. 933-938, Denver, CO, USA, Nov. 2000.

[Bishop 95a] C. M. Bishop: Neural Networks for Pattern Recognition. Oxford University Press, 1995.

[Bishop 95b] C. M. Bishop: Training with Noise is Equivalent to Tikhonov Regularization. Neural Computation, 7(1), pp. 108-116, Jan. 1995.

[Botros & Irie+ 15] R. Botros, K. Irie, M. Sundermeyer, H. Ney: On Efficient Training of Word Classes and Their Application to Recurrent Neural Network Language Models. Interspeech, pp.1443-1447, Dresden, Germany, Sep. 2015.

[Bourlard & Wellekens 87] H. Bourlard, C. J. Wellekens: Multilayer perceptrons and automatic speech recognition. First Int. Conf. on Neural Networks, pp. 407-416, San Diego, CA, 1987.

[Bourlard & Wellekens 89] H. Bourlard, C. J. Wellekens: 'Links between Markov Models and Multilayer Perceptrons', in D.S. Touretzky (ed.): "Advances in Neural Information Processing Systems I", Morgan Kaufmann Pub., San Mateo, CA, pp.502-507, 1989.

[Bridle 82] J. S. Bridle, M. D. Brown, R. M. Chamberlain: An Algorithm for Connected Word Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Paris, pp. 899-902, May 1982.

[Bridle 89] J. S. Bridle: Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition, in F. Fogelman-Soulie, J. Herault (eds.): 'Neuro-computing: Algorithms, Architectures and Applications', NATO ASI Series in Systems and Computer Science, Springer, New York, 1989.

[Bridle & Dodd 91] J. S. Bridle, L. Dodd: An Alphanet Approach To Optimising Input Transformations for Continuous Speech Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Toronot, pp. 277-280, April 1991.

[Brown & Della Pietra$^+$ 93] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, R. L. Mercer: Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, Vol. 19.2, pp. 263-311, June 1993.

[Brown & Mann$^+$ 22] T. R. Brown, B. Mann, N. Ryder et al.: Language Models are Few-Shot Learners. OpenAI (GPT-3), arxiv, 22-Jul-2022.

[Collobert & Weston 08] R. Collobert, J. Weston: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. Int. Conference on Machine Learning (ICML), 2008.

[Collobert & Weston$^+$ 11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa: Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research, 2011.

[Castano & Vidal$^+$ 93] M.A. Castano, E. Vidal, F. Casacuberta: Inference of stochastic regular languages through simple recurrent networks. IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives, pp. 16/1-6, Colchester, UK, April 1993.

[Castano & Casacuberta 97] M. Castano, F. Casacuberta: A connectionist approach to machine translation. European Conf. on Speech Communication and Technology (Eurospeech), pp. 91–94, Rhodes, Greece, Sep. 1997.

[Castano & Casacuberta+ 97] M. Castano, F. Casacuberta, E. Vidal: Machine translation using neural networks and finite-state models. Int. Conf. on Theoretical and Methodological Issues in Machine Translation (TMI), pp. 160-167, Santa Fe, NM, USA, July 1997.

[Chien & Lu 15] : Jen-Tzung Chien, Tsai-Wei Lu: Deep Recurrent RegularizatioN Neural Network for Speech Recognition. ICASSP 2015, pp. 4560-4564.

[Cover & Thomas 91] T. M. Cover, J. A. Thomas: Elements of Information Theory. John Wiley & Sons, New York, NY, 1991.

[Dahl & Ranzato+ 10] G. E. Dahl, M. Ranzato, A. Mohamed, G. E. Hinton: Phone recognition with the mean-covariance restricted Boltzmann machine. Advances in Neural Information Processing Systems (NIPS) 23, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, Eds. Cambridge, MA, MIT Press, 2010, pp. 469-477.

[Dahl & Yu+ 12] G. E. Dahl, D. Yu, L. Deng, A. Acero: Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. IEEE Tran. on Audio, Speech and Language Processing, Vol. 20, No. 1, pp. 30-42, Jan. 2012.

[Dehak & Kenny+ 11] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, P. Ouellet: Front-End Factor Analysis for Speaker Verification IEEE Trans. on audio, speech, and language processing, pp. 788-798, Vol. 19, No. 4, May 2011.

[Devlin & Zbib+ 14] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, J. Makhoul: Fast and Robust Neural Network Joint Models for Statistical Machine Translation. Annual Meeting of the ACL, pp. 1370–1380, Baltimore, MA, June 2014.

[Devroye & Györfi+ 96] L. Devroye, J. Györfi, G. Lugosi: A Probabilistic Theory of Pattern Recognition. Springer, New York, 1996.

[Doetsch & Hannemann+ 17] P. Doetsch , M. Hannemann, R. Schlüer, H. Ney: Inverted Alignments for End-to-End Automatic Speech Recognition. IEEE Journal of selected topics in Signal Processing, Vol. 11, No. 8, pp. 1265-1273, Dec. 2017.

[Duda & Hart 73] R. O. Duda, P. E. Hart: Pattern Classification and Scene Analysis. Wiley, Hoboken, 1973.

[Fedotov & Harremoes+ 03] A. A. Fedotov, P. Harremoes, F. Topsoe: Refinements of Pinsker's Inequality. Some Inequalities for Information Divergence and Related Measures of Discrimination. IEEE Trans. on Information Theory, Vol. 49, No. 6, pp. 1491-1498, June 2003.

[Forcada & Carrasco 05] M. L. Forcada, R. C. Carrasco: Learning the initial state of a second-order recurrent neural network during regular language inference. Neural Computation, Vol. 7, No. 5, pp. 923-930, Sep. 2005.

[Fontaine & Ris+ 97] V. Fontaine, C. Ris, J.-M. Boite: Nonlinear discriminant analysis for improved speech recognition. Eurospeech, Rhodes, Greece, Sep. 1997.

[Fritsch & Finke+ 97] J. Fritsch, M. Finke, A. Waibel: Adaptively Growing Hierarchical Mixtures of Experts. NIPS, Advances in Neural Information Processing Systems 9, MIT Press, pp. 459-465, 1997.

[Fukunaga 72] K. Fukunaga: Introduction to Statistical Pattern Recognition. Academic Press, New York, 1972.

[Gemello & Manai+ 06] R. Gemello, F. Mana, S. Scanzio, P. Lafac, R. De Mori: Adaptation of Hybrid ANN/HMM Models Using Linear Hidden Transformations and Conservative Training. IEEE Int. Conf. on Acoustics Speech and Signal Processing Proceedings, Toulouse, 2006.

[Gers & Schmidhuber+ 00] F. A. Gers, J. Schmidhuber, F. Cummin: Learning to forget: Continual prediction with LSTM. Neural computation, Vol 12, No. 10, pp. 2451-2471, 2000.

[Gers & Schraudolph+ 02] F. A. Gers, N. N. Schraudolph, J. Schmidhuber: Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, Vol. 3, pp. 115-143, 2002.

[Graves 12] A. Graves: Sequence Transduction with Recurrent Neural Networks. U of Toronto, Canada, arxiv, 12-Nov-2012.

[Graves & Fernandez+ 06] A. Graves, S. Fernandez, F Gomez, J. Schmidhuber: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. Int. Conf. on Machine Learning, Pittsburgh, PA, pp. 369-376, 2006.

[Graves & Schmidhuber 09] A. Graves, J. Schmidhuber: Offline handwriting recognition with multidimensional recurrent neural networks. NIPS 2009.

[Grezl & Fousek 08] F. Grezl, P. Fousek: Optimizing bottle-neck features for LVCSR. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 4729-4732, Las Vegas, NV, March 2008.

[Grosicki & El Abed 09] E. Grosicki, H. El Abed: ICDAR 2009 Handwriting Recognition Competition. Int. Conf. on Document Analysis and Recognition (ICDAR) 2009, Barcelona, pp. 139-1402, July 2009.

[Haffner 93] P. Haffner: Connectionist Speech Recognition with a Global MMI Algorithm. 3rd Europ. Conf. on Speech Communication and Technology (Eurospeech'93), Berlin, Germany, Sep. 1993.

[Hampshire & Pearlmutter 90] J. B. Hampshire, B. Pearlmutter: Equivalence Proofs for Multilayer Perceptron Classifiers and the Bayesian Discriminant Function. In D. S. Touretzky, J. L. Hinton, T. J. Sejnowksi, G. E. Hinton (eds.): *Proceedings of the 1990 Connectionist Summer School*, pp. 159-172, San Mateo, CA, Morgan Kaufmann.

[Heigold & Macherey 05+] G. Heigold, W. Macherey, R. Schlüter, H. Ney: Minimum Exact Word Error Training. IEEE ASRU workshop, pp. 186-190, San Juan, Puerto Rico, Nov. 2005.

[Heigold & Schlüter 12+] G. Heigold, R. Schlüter, H. Ney, S. Wiesler: Discriminative Training for Automatic Speech Recognition: Modeling, Criteria, Optimization, Implementation, and Performance. IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 58-69, Nov. 2012.

[Hermansky & Ellis+ 00] H. Hermansky, D. W. Ellis, S. Sharma: Tandem connectionist feature extraction for conventional HMM systems. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 1635-1638, Istanbul, Turkey, June 2000.

[Hinton & Osindero+ 06] G. E. Hinton, S. Osindero, Y. Teh: A fast learning algorithm for deep belief nets. Neural Computation, Vol. 18, No. 7, pp. 1527-1554, July 2006.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long short-term memory. Neural Computation, Vol. 9, No. 8, pp. 1735–1780, Nov. 1997.

[Ivakhnenko 71] A. G. .Ivakhnenko: Polynomial theory of complex systems. IEEE Transactions on Systems, Man and Cybernetics, Vol. 1, No. 4, pp. 364-378, Oct. 1971.

[Jelinek & Mercer$^+$ 77] F. Jelinek, R. L. Mercer, L. R. Bahl: Perplexity – a measure of the difficulty of speech recognition tasks. Journal of the Acoustical Society of America, 1977.

[Kaltenbrenner & Blunsom 13] N. Kalchbrenner, P. Blunsom: Recurrent continuous translation models. EMNLP 2013.

[Juang & Chou$^+$ 97] B.-H. Juang, W. Chou, C.-H. Lee: Minimum Classification Error Rate Methods for Speech Recognition. IEEE Transactions on Speech and Audio Processing, Vol. 5, No. 3, pp. 257-265, May 1997.

[Juang & Katagiri 92] B.-H. Juang, S. Katagiri: Discriminative Learning for Minimum Error Classification. IEEE Transactions on Signal Processing, Vol. 40, No. 12, pp. 3043-3054, Dec. 1992.

[Klakow & Peters 02] D. Klakow, J. Peters: Testing the correlation of word error rate and perplexity. Speech Communication, pp. 19–28, 2002.

[Koehn & Och$^+$ 03] P. Koehn, F. J. Och, D. Marcu: Statistical Phrase-Based Translation. HLT-NAACL 2003, pp. 48-54, Edmonton, Canada, May-June 2003.

[Le & Allauzen$^+$ 12] H.S. Le, A. Allauzen, F. Yvon: Continuous space translation models with neural networks. NAACL-HLT 2012, pp. 39-48, Montreal, QC, Canada, June 2002.

[LeCun & Bengio$^+$ 94] Y. LeCun, Y. Bengio: Word-level training of a handwritten word recognizer based on convolutional neural networks. Int. Conf. on Pattern Recognition, Jerusalem, Israel, pp. 88-92, Oct. 1994.

[Makhoul & Schwartz 94] J. Makhoul, R Schwartz: State of the Art in Continuous Speech Recognition. Chapter 14, pp. 165-198, in D. B. Roe, J. G. Wilpon (Editors): Voice Communication Between Humans and Machines. National Academy of Sciences, 1994.

[Miao & Metze 15] Y. Miao. F Metze: On speaker adaptation of long short-term memory recurrent neural networks. Interspeech, Dresden, Germany, 2015.

[Mikolov & Corrado+ 13] T. Mikolov, G. Corrado, K. Chen, J. Dean: Efficient Estimation of Word Representations in Vector Space. Google, arxiv, 07-Sep-2013.

[Mikolov & Karafiat+ 10] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, S. Khudanpur: Recurrent neural network based language model. Interspeech, pp. 1045-1048, Makuhari, Chiba, Japan, Sep. 2010.

[Mohamed & Dahl+ 09] A. Mohamed, G. Dahl, G. Hinton: Deep belief networks for phone recognition. NIPS Workshop Deep Learning for Speech Recognition and Related Applications, 2009.

[Morgan & Bourlard 90] N. Morgan, H. Bourlard: Continuous speech recognition using multilayer perceptrons with hidden Markov models. ICASSP 1990, pp. 413-416, Albuquerque, NM, 1990.

[Nakamura & Shikano 89] M. Nakamura, K. Shikano: A Study of English Word Category Prediction Based on Neural Networks. ICASSP 89, p. 731-734, Glasgow, UK, May 1989.

[Neco & Forcada 97] R. P. Neco, M. L. Forcada: Asynchronous translations with recurrent neural nets. IEEE Int. Conf. on Neural Networks, pp. 2535-2540, June 1997.

[Ney 03] H. Ney: On the Relationship between Classification Error Bounds and Training Criteria in Statistical Pattern Recognition. First Iberian Conf. on Pattern Recognition and Image Analysis, Puerto de Andratx, Spain, Springer LNCS Vol. 2652, pp. 636-645, June 2003.

[Ney 84] H. Ney: The Use of a One–Stage Dynamic Programming Algorithm for Connected Word Recognition. IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-32, No. 2, pp. 263-271, April 1984.

[Ney & Haeb-Umbach+ 92] H. Ney, R. Haeb-Umbach, B.-H. Tran, M. Oerder: Improvements in Beam Search for 10000-Word Continuous Speech Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, San Francisco, CA, pp. 13-16, March 1992.

[Normandin & Cardin+ 94] Y. Normandin, R. Cardin, R. De Mori: High-Performance Connected Digit Recognition Using Maximum Mutual Information Estimation. IEEE Trans. on Speech and Audio Processing, vol. 2, no. 2, pp. 299-311, April 1994.

[Ouyang & Wu+ 22] L. Ouyang, J. Wu, X. Jiang et al.: Training language models to follow instructions with human feedback. OpenAI, arxiv, 04-Mar-2022.

[Och & Ney 03] F. J. Och, H. Ney: A Systematic Comparison of Various Alignment Models. *Computational Linguistics,* Vol. 29, No. 1, pp. 19-51, March 2003.

[Och & Ney 04] F. J. Och, H. Ney: The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, Vol. 30, No. 4, pp. 417-449, Dec. 2004.

[Och & Tillmann+ 99] F. J. Och, C. Tillmann, H. Ney: Improved Alignment Models for Statistical Machine Translation. Joint ACL/SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora, College Park, MD, pp. 20-28, June 1999.

[Patterson & Womack 66] J. D. Patterson, B. F. Womack: An Adaptive Pattern Classification Scheme. IEEE Trans. on Systems, Science and Cybernetics, Vol.SSC-2, pp.62-67, Aug. 1966.

[Pereyra & Tucker+ 17] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, G. Hinton: Regularizing Neural Networks bY Penalizing Confident Output Distributions. arxiv, 23-Jan-2017, ICLR 2017.

[Povey & Woodland 02] D. Povey, P.C. Woodland: Minimum phone error and I-smoothing for improved discriminative training. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 105–108, Orlando, FL, May 2002.

[Printz & Olsen 02] H. Printz, P. A. Olsen: Theory and practice of acoustic confusability. Computer Speech and Language, pp. 131–164, Jan. 2002.

[Radford & Wu+ 18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever: Language Models are Unsupervised Multitask Learners. OpenAI (GPT-2), preprint, 2018.

[Radford & Narasimhan+ 19] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever: Improving Language Understanding by Generative Pre-Training. OpenAI (GPT-1), preprint, 2019.

[Raissi & Beck+ 20] T. Raissi, E. Beck, R. Schlüter, H. Ney: Context-Dependent Acoustic Modeling without Explicit Phone Clustering arxiv, 2020.

[Raissi & Beck$^+$ 21] T. Raissi, E. Beck, R. Schlüter, H. Ney: Towards Consistent Hybrid HMM Acoustic Modeling. arxiv, 2021.

[Raissi & Beck$^+$ 22] T. Raissi, E. Beck, R. Schlüter, H. Ney: Improving Factored Hybrid HMM Acoustic Modeling without State Tying. arxiv, 2022.

[Robinson 94] A. J. Robinson: An Application of Recurrent Nets to Phone Probability Estimation. IEEE Trans. on Neural Networks, Vol. 5, No. 2, pp. 298-305, March 1994.

[Sainath & Weiss$^+$ 16] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, M. Bacchiani: Factored Spatial and Spectral Multichannel Raw Waveform CLDNNs, Proc. ICASSP, 2016.

[Saon & Tüske$^+$ 2021] G. Saon, Z. Tüske, D. Bolanos, B. Kingsbury: Advancing RNN Transducer Technology for Speech Recognition. IBM Research AI, Yorktown Heights, USA, arxiv, 17-Mar-2021.

[Sakoe & Chiba 71] H. Sakoe, S. Chiba: A Dynamic Programming Approach to Continuous Speech Recognition. Proc. 7th Int. Congr. on Acoustics, Budapest, Hungary, Paper 20 C 13, pp. 65-68, August 1971.

[Sak & Shannon$^+$ 17] H. Sak, M. Shannon, K. Rao, F. Beaufays: Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping. Interspeech, Stockholm, Sweden, pp. 1298-1302, Aug. 2017.

[Schlüter & Beck$^+$ 19] R. Schlter, E. Beck, H. Ney: Upper and Lower Tight Error Bounds for Feature Omission with an Extension to Context Reduction. IEEE Trans. Pattern Anal. Mach. Intell., Vol. 41, No. 2, pp. 502-514, 2019.

[Schlüter & Nussbaum$^+$ 11] R. Schlüter, M. Nussbaum-Thom, H. Ney: On the Relationship between Bayes Risk and Word Error Rate in ASR. IEEE Trans. on Audio, Speech, and Language Processing, vol. 19, no. 5, p. 1103-1112, July 2011.

[Schlüter & Nussbaum$^+$ 12] R. Schlüter, M. Nussbaum-Thom, H. Ney: Does the Cost Function Matter in Bayes Decision Rule? IEEE Trans. PAMI, No. 2, pp. 292–301, Feb. 2012.

[Schlüter & Nussbaum-Thom$^+$ 13] R. Schlüter, M. Nußbaum-Thom, E. Beck, T. Alkhouli, H. Ney: Novel Tight Classification Error Bounds under Mismatch Conditions based on f-Divergence. IEEE Information Theory Workshop, pp. 432–436, Sevilla, Spain, Sep. 2013.

[Schlüter & Scharrenbach+ 05] R. Schlüter, T. Scharrenbach, V. Steinbiss, H. Ney: Bayes Risk Minimization using Metric Loss Functions Interspeech, pages 1449-1452, Lisboa, Portugal, Sep. 2005.

[Schmidhuber 14] Deep Learning in Neural Networks: An Overview. 88 pages with 53 pages of references, arXiv:1404.7828v4, 08-Oct-2014.

[Schuster & Paliwal 97] M. Schuster, K. K. Paliwal: Bidirectional Recurrent Neural Networks. IEEE Trans. on Signal Processing, Vol. 45, No. 11, pp. 2673-2681, Nov. 1997.

[Schwenk 07] H. Schwenk: Continuous space language models. Computer Speech and Language, Vol. 21, No. 3, pp. 492–518, July 2007.

[Schwenk 12] H. Schwenk: Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. 24th Int. Conf. on Computational Linguistics (COLING), Mumbai, India, pp. 1071–1080, Dec. 2012.

[Schwenk & Costa-jussa+ 07] H. Schwenk , M. R. Costa-jussa, J. A. R. Fonollosa: Smooth bilingual n-gram translation. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 430–438, Prague, June 2007.

[Schwenk & Déchelotte+ 06] H. Schwenk, D. Déchelotte, J. L. Gauvain: Continuous Space Language Models for Statistical Machine Translation. COLING/ACL 2006, pp. 723–730, Sydney, Australia July 2006.

[Schwenk & Gauvain 02] H. Schwenk, J.-L. Gauvain: Connectionist language modeling for large vocabulary continuous speech recognition. pp. 765-768, ICASSP 2002.

[Seide & Li+ 11] F. Seide, G. Li, D. Yu: Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. Interspeech, pp. 437-440, Florence, Italy, Aug. 2011.

[Solla & Levin+ 88] S. A. Solla, E. Levin, M. Fleisher: Accelerated Learning in Layered Neural Networks. Complex Systems, Vol.2, pp. 625-639, 1988.

[Soltan & Ananthakrishnan+ 22] S. Soltan, S. Ananthakrishnan, J. FitzGerald et al.: AlexaTM 20B: Few-Shot Learning Using a Large-Scale Multilingual Seq2seq Model. Amazon, arxiv, 03-Aug-2022.

[Stolcke & Grezl+ 06] A. Stolcke, F. Grezl, M.-Y. Hwang, X. Lei, N. Morgan, D. Vergyri: Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Toulouse, France, May 2006.

[Sundermeyer & Alkhouli+ 14] M. Sundermeyer, T. Alkhouli, J. Wuebker, H. Ney: Translation Modeling with Bidirectional Recurrent Neural Networks. Conf. on Empirical Methods in Natural Language Processing (EMNLP), pp. 14–25, Doha, Qatar, Oct. 2014.

[Sundermeyer & Ney+ 15] M. Sundermeyer, H. Ney, R. Schlüter: From feedforward to recurrent LSTM neural networks for language modeling. IEEE/ACM Trans. on Audio, Speech, and Language Processing, Vol. 23, No. 3, pp. 13–25, March 2015.

[Sundermeyer & Schlüter+ 12] M. Sundermeyer, R. Schlüter, H. Ney: LSTM neural networks for language modeling. Interspeech, pp. 194–197, Portland, OR, USA, Sep. 2012.

[Sutskever & Vinyals+ 14] I. Sutskever, O. Vinyals, Q. V. Le: Sequence to Sequence Learning with Neural Networks. Google, arxiv, 14-Dec-2014.

[Tikhonov & Arsenin 77] A. N. Tikhonov, V. Y. Arsenin: Solutions of Ill-Posed Problems. Washington DC, Winston 1977.

[Tüske & Plahl+ 11] Z. Tüske, C. Plahl, R. Schlüter: A study on speaker normalized MLP features in LVCSR. Interspeech, pp. 1089-1092, Florence, Italy, Aug. 2011.

[Tüske & Golik+ 14] Z. Tüske, P. Golik, R. Schlúter, H. Ney: Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR. Interspeech, ISCA best student paper award, pp. 890-894, Singapore, Sep. 2014.

[Utgoff & Stracuzzi 02] P. E. Utgoff, D. J. Stracuzzi: Many-layered learning. Neural Computation, Vol. 14, No. 10, pp. 2497-2539, Oct. 2002.

[Valente & Vepa+ 07] F. Valente, J. Vepa, C. Plahl, C. Gollan, H. Hermansky, R. Schlüter: Hierarchical Neural Networks Feature Extraction for LVCSR system. Interspeech, pp. 42-45, Antwerp, Belgium, Aug. 2007.

[Vapnik 98] Vapnik: Statistical Learning Theory. Addison-Wesley, 1998.

[Variani & Sainath+ 16] E. Variani, T. N. Sainath, I. Shafran, M. Bacchiani: Complex Linear Projection (CLP): A Discriminative Approach to Joint Feature Extraction and Acoustic Modeling. Interspeech 2016, San Francisco, CA, pp. 808-812, Sep. 2016.

[Vaswani & Shazeer+ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser: Attention Is All You Need. Google, arxiv, 06-Dec-2017.

[Vaswani & Zhao+ 13] A. Vaswani, Y. Zhao, V. Fossum, D. Chiang: Decoding with Large-Scale Neural Language Models Improves Translation. Conf. on Empirical Methods in Natural Language Processing (EMNLP, pp. 1387–1392, Seattle, Washington, USA, Oct. 2013.

[Velichko & Zagoruyko 70] V. M. Velichko, N. G. Zagoruyko: Automatic Recognition of 200 Words. Int. Journal Man-Machine Studies, Vol. 2, pp. 223-234, June 1970.

[Vintsyuk 68] T. K. Vintsyuk: Speech Discrimination by Dynamic Programming. Kibernetika (Cybernetics), Vol. 4, No. 1, pp. 81-88, Jan.-Feb. 1968.

[Vintsyuk 71] T. K. Vintsyuk: Elementwise Recognition of Continuous Speech Composed of Words from a Specified Dictionary. Kibernetika (Cybernetics), Vol. 7, pp. 133-143, March-April 1971.

[Vogel & Ney+ 96] S. Vogel, H. Ney, C. Tillmann: HMM-based word alignment in statistical translation. Int. Conf. on Computational Linguistics (COLING), pp. 836-841, Copenhagen, Denmark, Aug. 1996.

[Waibel & Hanazawa+ 88] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. L. Lang: Phoneme Recognition: Neural Networks vs. Hidden Markov Models. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), New York, NY, pp.107-110, April 1988.

[Wang & Alkhouli+ 17] W. Wang, T. Alkhouli, D. Zhu, H. Ney: Hybrid Neural Network Alignment and Lexicon Model in Direct HMM for Statistical Machine Translation. Annual Meeting ACL, pp. 125-131, Vancouver, Canada, Aug. 2017.

[Wang & Zhu+ 18] W. Wang, D. Zhu, T. Alkhouli, Z. Gan, H. Ney: Neural Hidden Markov Model for Machine Translation. Annual Meeting ACL, Melbourne, Australia, July 2018.

[Xu & Povey+ 10] H. Xu, D. Povey, L. Mangu, J. Zhu: Minimum Bayes Risk Decoding and System Combination Based on a Recursion for Edit Distance. Computer Speech and Language, Sep. 2010.

[Zens & Och+ 02] R. Zens, F. J. Och, H. Ney: Phrase-Based Statistical Machine Translation. 25th Annual German Conf. on AI, pp. 18–32, LNAI, Springer 2002.

[Zhou & Berger+ 2021] W. Zhou, S. Berger, R. Schlüter, H. Ney: Phoneme Based Neural Transducer for Large Vocabulary Speech Recognition. ICASSP, Toronto, June 2021.

[Zhou & Zeyer+ 2021] W. Zhou, A. Zeyer, A. Merboldt, R .Schlüter, H. Ney: Equivalence of Segmental and Neural Transducer Modeling: A Proof of Concept. Interspeech, pp. 2891-2895, Graz, 2021.

END

Master Course, ASR
UP Valencia, Jan. 2024