

3rd International Summer School on Deep Learning 2019
22-26 July 2019, Warsaw, Poland

**Speech Recognition and Machine Translation: From Statistical
Decision Theory to Machine Learning and Deep Neural Networks**

Hermann Ney

Lehrstuhl Informatik 6 (i6)
Human Language Technology and Pattern Recognition
RWTH Aachen University, Aachen, Germany



preview: characteristic properties of this lecture

- application of ANNs to real-life tasks:
 - speech recognition, machine translation, ...
 - important aspect: string-to-string conversion (context information)
- ANNs have a long history:
 - started around 30 years ago for speech and language technology
 - real success only since 2011
- ANNs are part of the statistical (= data driven) approach:
 - define one family of statistical models (concatenation of dot products and nonlinearities)
 - share many properties with other statistical methods
 - need to be embedded in Bayes decision rule etc.
 - comparison with conventional (non-ANN) approaches

very short summary:

- there has been life before ANNs and deep learning:
Bayes decision rule, Gaussian modelling, HMMs, discriminative training, ...
- ANNs helped to improve the performance dramatically
- there will be life after ANNs and deep learning

re-consider today's buzz words in deep learning:

- sequence-to-sequence modelling/systems/training
- end-to-end modelling/systems/training
- discriminative modelling/training
- ...

we will show:

- well known concepts in machine learning and pattern recognition that have been re-invented
- today: these concepts result in improved performance due to deep learning
- properties of these concepts: are they useful? why?

tasks: speech recognition and machine translation

Outline

1	Introduction	8
1.1	Overview of these Lectures	8
1.2	Overview and History: ASR and HLT	12
2	Neural Networks and Deep Learning	36
2.1	Principles	36
2.2	Training Data and Empirical Distribution	44
2.3	Training Criteria and Probabilistic Interpretation	47
2.4	Recurrent Neural Networks for Sequence Processing	70
2.4.1	Principle	70
2.4.2	Extension: LSTM RNN	75
2.4.3	Interpretation of RNN Outputs	82
2.4.4	Backpropagation for Baseline RNN	92
2.5	Historical Review: ANN and Deep Learning	96
3	Bayes Decision Rule: Principle	97
3.1	Decision Rules and Loss Function	98
3.2	Empirical Distribution	103
3.3	Bayes Decision Rule	107
3.4	Conclusions	112
4	Bayes Decision Rule: Loss Function	113
4.1	Atomic Output	114
4.2	Structured Output: String with Synchronization	117
4.3	Structured Output: String with no Synchronization	124
4.3.1	Pathological Example: Bayes String with Zero Probability	125
4.3.2	Basic Inequality for Metric Loss Function	131
4.3.3	Rewrite Posterior Loss	136

4.3.4	Edit Distance and Symbol Posterior Probability	140
4.3.5	Summary	146
4.3.6	Excursion for ASR (draft): Frame Error vs. Edit Distance	147
5	Single Events: Modelling and Training Criteria	151
5.1	Generative Modelling	153
5.2	Direct and Discriminative Modelling	159
5.2.1	Principle: Class Posterior Probability	160
5.2.2	Example: Gaussian Posterior	164
5.2.3	Softmax Revisited	170
5.3	Posterior Distribution and Log-Linear Modelling	171
5.4	Summary	177
6	Structured Output: String-to-String Processing	178
6.1	Overview: Three Approaches to String Posterior Probability	181
6.2	Generative Models	186
6.3	Generalized Models (CRF)	192
6.4	Direct Models	200
6.5	Sequence Discriminative Training Revisited	214
6.6	Conclusion	218
7	Training Criterion: Bayes Decision Rule and Mismatch Conditions	219
7.1	Introduction	219
7.2	Probability Models and Decision Rule	227
7.3	Classification Errors: Two Types	231
7.4	From Global Bounds to Training Criteria	235
7.4.1	Unconstrained Output: Squared Error Bound	236
7.4.2	Constrained Output: Binary Divergence	240
7.4.3	Normalized Output: Kullback-Leibler Divergence Bound	242

7.5 Refinements for Kullback-Leibler Bound	250
7.5.1 From Discriminative to Generative Training	251
7.5.2 From Single Symbols to Symbol Strings	255
7.6 Conclusion	258
8 Training Criteria:	
Error Counting and Heuristic Approaches	261
8.1 Error Counting: Minimum Classification Error	262
8.2 Error Counting and Squared Error	265
8.3 Error Counting and Consistency Requirement	274
8.3.1 Principle	275
8.3.2 Power Approximation to Logarithm	279
8.3.3 Trial: Relaxed Consistency Requirement	284
8.4 Model-Based Expected Loss	286
8.4.1 Principle	288
8.4.2 Calculation of Expected Loss	292
8.4.3 Empirical Averages for Three Types of Losses	301
8.4.4 Training Criteria: Analysis and Comparison	311
8.5 Conclusions	318
9 Deep Learning for Acoustic Modelling	320
9.1 System Architecture for ASR	321
9.1.1 Statistical Approach	321
9.1.2 Hidden Markov Models (HMM)	326
9.1.3 Generative vs. Discriminative Training	333
9.2 History: ASR and ANNs	336
9.3 From HMM to Hybrid HMM and CTC	344
9.4 EM Algorithm and Gradient Search	351
9.4.1 Principle: EM Algorithm and Gradient Search	352

9.4.2 Hybrid HMM and EM Algorithm	358
9.5 Experimental Results	373
10 Deep Learning for Language Modelling	386
10.1 Why Language Modelling?	386
10.2 Word Vectors and Word Embeddings	392
10.3 Neural Networks for LM	407
10.4 Experimental Results	415
11 Deep Learning for Machine Translation	429
11.1 Overview of Translation Approaches	429
11.2 Neural MT using Attention Mechanism	443
11.3 Neural MT using Transformer Approach	458
11.4 Neural MT using Direct HMM	468
11.5 Experimental Results	483
12 Summary and Conclusions	497
13 References	501

1 Introduction

1.1 Overview of these Lectures

Advanced Topics: Machine Learning and Human Language Technology

framework for these lectures:

- machine learning (or pattern recognition):
specific application to human language technology tasks:
speech recognition, handwriting recognition, machine translation, ...
- specific area of machine learning:
sequence-to-sequence recognition/processing
- most successful paradigm today:
artificial neural networks and deep learning
(but only one example of data-driven/statistical methods)

topics/questions covered by these lectures:

- dominant method: ANNs
 - what is the relation to the probabilistic framework?
 - what exactly are the outputs of ANNs?
- performance point-of-view:
 - how to achieve optimal performance (= minimum classification error)?
 - (theoretical) answer: Bayes decision rule
- Bayes decision rule:
 - does the exact form of the performance measure matter?
 - can it be simplified without hurting performance?
- classical modelling approaches outside ANNs:
 - what is their relation to ANNs?
 - what are the associated training criteria?
 - emphasis on sequence/string processing
- mismatch conditions in using Bayes decision rule:
 - mismatch between true vs. model distribution
 - how can we preserve the optimality of the decision rule?
- sequence-to-sequence processing

...

- application of ANNs to real-life tasks:
 - speech recognition, machine translation, ...
 - important aspect: string-to-string conversion (context information)
- ANNs have a long history:
 - started around 30 years ago for speech and language technology
 - worldwide real success started only around 2011
- ANNs are part of the statistical (= data driven) approach:
 - define one family of statistical models (concatenation of dot products and nonlinearities)
 - share many properties with other statistical methods
 - need to be embedded in Bayes decision rule etc.
 - comparison with conventional (non-ANN) approaches

very short summary:

- there has been life before ANNs and deep learning:
Bayes decision rule, Gaussian modelling, HMMs, discriminative training, ...
- ANNs helped to improve the performance dramatically
- there will be life after ANNs and deep learning

re-consider today's buzz words in deep learning:

- sequence-to-sequence modelling/systems/training
- end-to-end modelling/systems/training
- discriminative modelling/training
- ...

we will show:

- well known concepts in machine learning and pattern recognition that have been re-invented
- but today it is true:
due to deep learning, these concepts were improved dramatically
- we consider the properties of these concepts:
are they useful? why?

1.2 Overview and History: ASR and HLT

terminology:

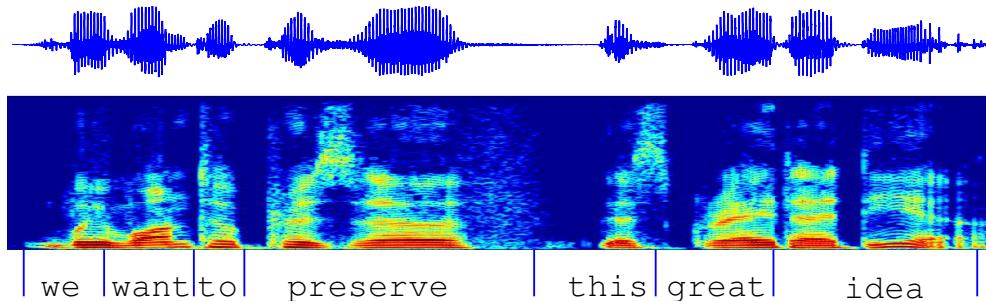
- speech: acoustic signal, spoken language
- language: text, sequence of characters, written language
- scientific disciplines: speech vs. language
 - NLP: natural language processing (in the strict sense): written language only
 - HLT: human language technology: spoken AND written language
- specific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

characteristic properties of these tasks:

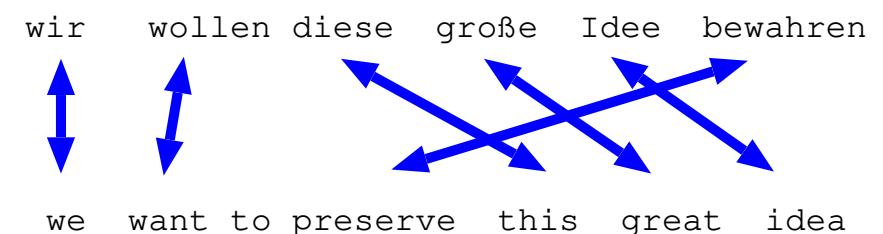
- well-defined 'classification' tasks:
 - due to 5000-year history of (written!) language
 - well-defined classes: letters or words of the language
- easy task for humans
(at least in their native language!)
- hard task for computers
(as the last 50 years have shown!)



Automatic Speech Recognition



Machine Translation



Handwriting Recognition



tasks for machine learning:

- automatic speech recognition
- handwriting recognition
- machine translation
- sign language (gesture) recognition
- semantic tagging (NLU)
- ...

typical situation:

input string → output string

tasks:

- **speech recognition:**
speech signal → string of words/letters
- **recognition of image text (printed and written characters):**
text image → string of words/letters
- **machine translation:**
string of source words → string of target words/letters

common property:

output string = string of words/letters in a natural language

terminology:

- **compound decision theory**
- **contextual pattern recognition**
- **structured output**

**elementary pattern classification
and machine learning:
single class index
without any structure**



participation in large-scale joint projects:

- **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 (8000 words)
 - large project: 10 million DM per year, about 25 partners
 - German partners: Daimler, Philips, Siemens, DFKI, KIT, RWTH, U Stuttgart, ...
- **commercial products by Philips:**
 - since 1989: 30k-word continuous speech recognition for text dictation
1993 commercial product for medical dictation
 - since 1993: spoken dialog systems via telephone, e.g. trainable information
1997 deployments in Switzerland and in the Netherlands

- **TC-STAR 2004-2007:** funded by EU
 - recognition and translation of speeches given in EU parliament
 - first research system for SPEECH TRANSLATION on real-life data
 - partners: KIT Karlsruhe, RWTH Aachen, FBK Trento, 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 Aachen
- **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 recognition
 - French partners (23): Thomson, France Telecom, Bertin, Systran, CNRS, INRIA, universities, ...
 - German Partners (2): KIT, RWTH



- EU project LUNA 2006-2009: spoken dialog system
- EU project SIGNSPEAK 2009-2012: sign language recognition and translation
- 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
- Google focused research award 2018-2020:
 - some fundamental problems in ASR

evaluations and databases: public and project related

- BABEL (IARPA) project
- CHiME evaluation: far-field and noisy speech
- Switchboard database: conversational telephone speech
- IWSLT: Int. Workshop on Spoken Language Translation
- WMT: Workshop on MT (before: NIST MT)

typical RWTH approach:

- we build fully-fledged systems for ASR, MT and HWR
- we work in European (EU, France) and international projects (DARPA, IARPA)
- we participate in international evaluation campaigns (project related and/or public)
- software and implementation:
 - emphasis on development of our own code
 - proprietary software for the full chain (ASR, MT, HWR): modelling, training, decoding
 - sub-package for machine learning using GPUs: RETURNN
 - source code: public and free for academia
 - AppTek: commercial partner
 - comparison with external toolkits and software (e.g. KALDI, MOSES, ANN packages...)
 - note: only a few teams have their own code

typical PhD student at RWTH:

- starts as part-time student and work on master thesis (1-2 years)
- continues with PhD work (4-5 years)
- works on projects, systems and evaluations as a part of the PhD thesis
- many of them stay in HLT field

former PhD students:

- Amazon: 8
- Apple: 6
- AppTek: 5 (+ 14 present PhD students)
- Ebay: 4
- Google: 11
- IBM: 3
- LILT: 3
- Nuance: 6
- ...
- academia: 7

ASR: what is the problem?

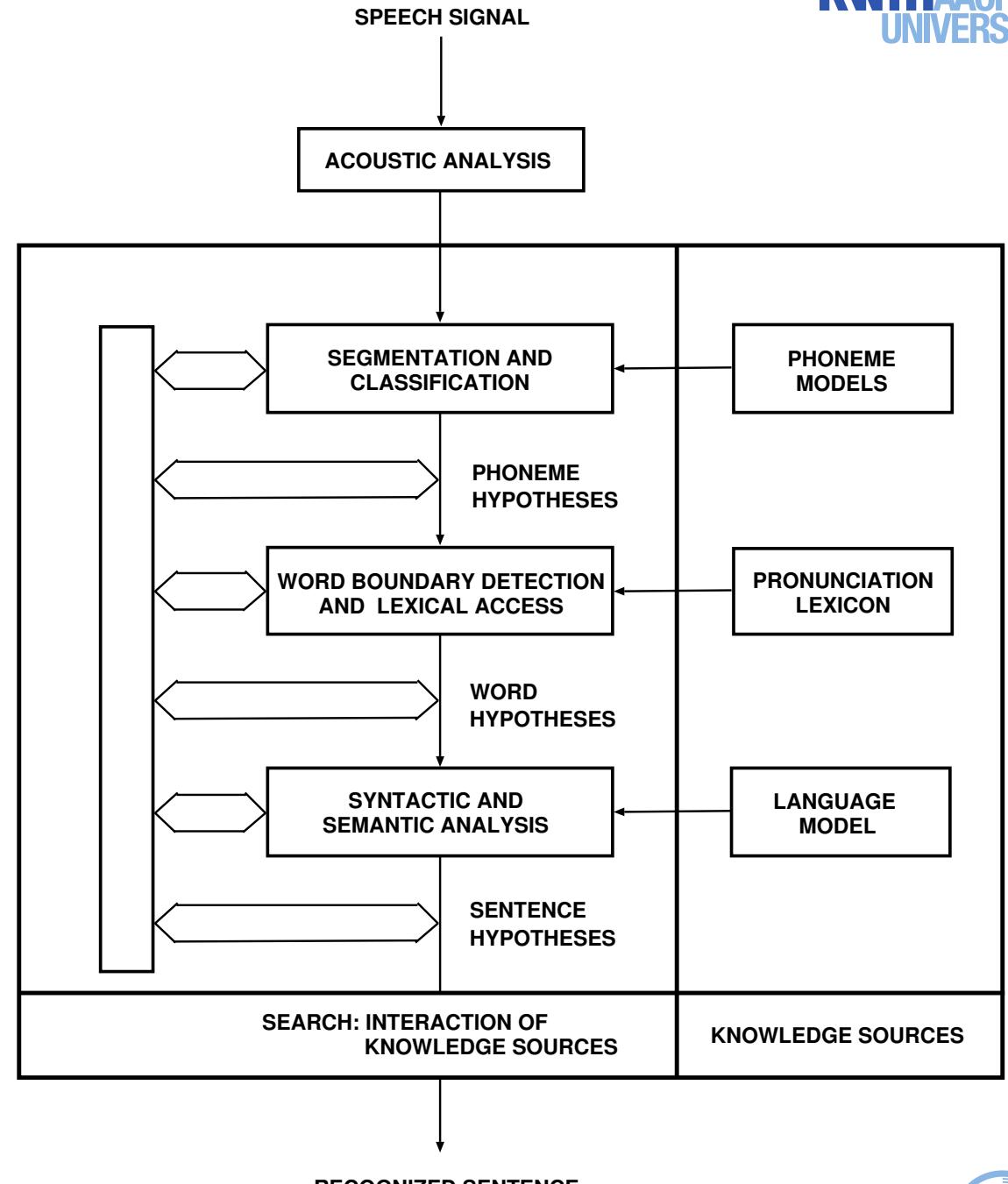
- ambiguities at all levels
- interdependencies of decisions

approach [CMU and IBM 1975]:

- score hypotheses
- Bayes decision rule
- probabilistic modelling and machine learning

important consequence:
the sequence context implies several different levels:

- 10-ms acoustic vectors
- as part of a sound/phoneme
- as part of a word
- as part of a sentence



- **two strings: input $x_1^T := x_1 \dots x_m \dots x_T$ and output $c_1^N := c_1 \dots c_n \dots c_N$ with a probabilistic dependence: $p(c_1^N | x_1^T)$**
- **performance measure or loss (error) function: $L[\tilde{c}_1^{\tilde{N}}, c_1^N]$ between correct output $\tilde{c}_1^{\tilde{N}}$ and generated output c_1^N**
- **Bayes decision rule minimizes expected loss:**

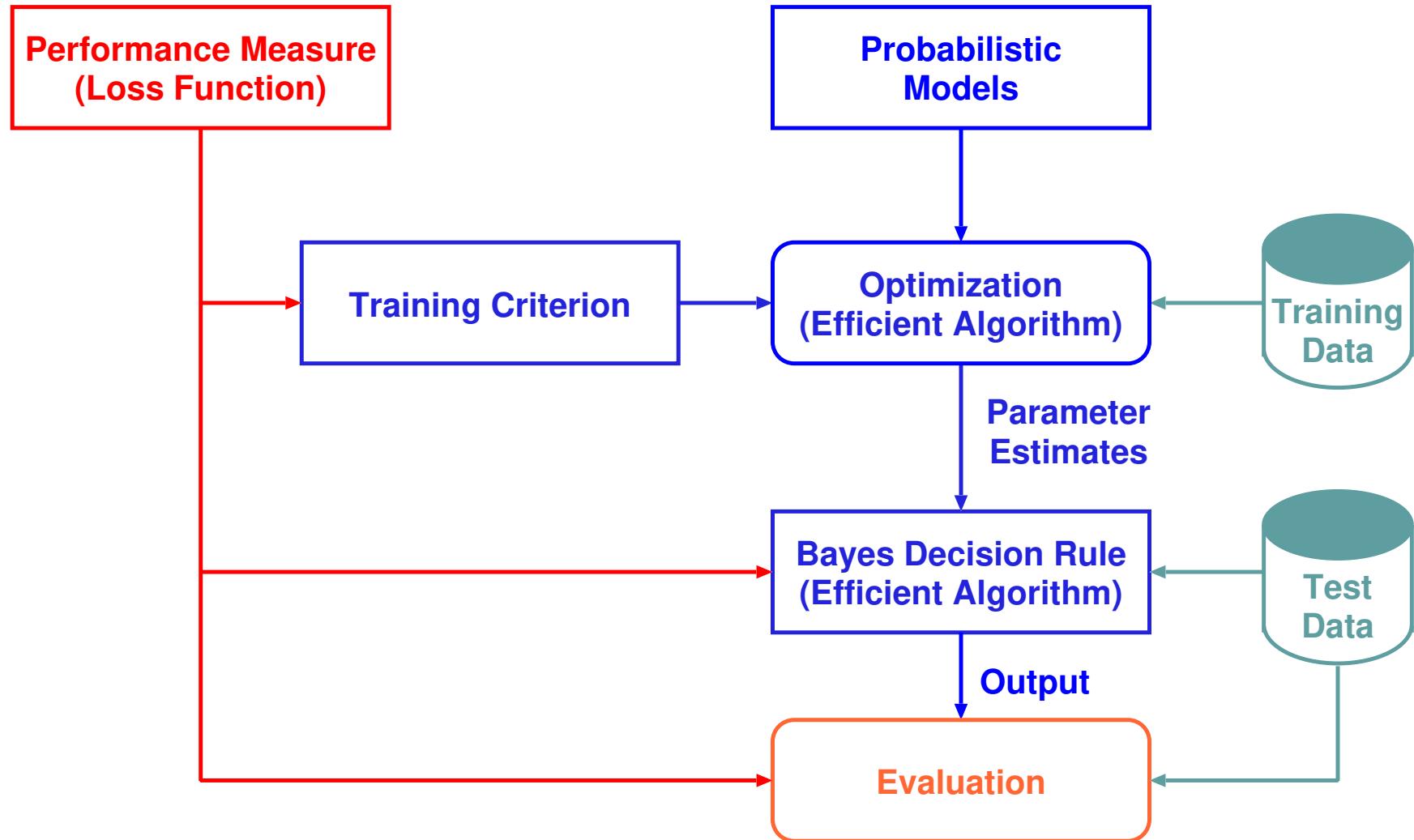
$$x_1^T \rightarrow \hat{c}_1^{\hat{N}}(x_1^T) := \arg \min_{N, c_1^N} \left\{ \sum_{\tilde{N}, \tilde{c}_1^{\tilde{N}}} p(\tilde{c}_1^{\tilde{N}} | x_1^T) \cdot L[\tilde{c}_1^{\tilde{N}}, c_1^N] \right\}$$

simplified rule (minimum string error): $x_1^T \rightarrow \hat{c}_1^{\hat{N}}(x_1^T) := \arg \max_{N, c_1^N} \left\{ p(c_1^N | x_1^T) \right\}$

- **from true to model distribution: separation of language model $p(c_1^N)$**

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

- **advantage: huge amounts of training data without annotation**
- **extension: log-linear modelling**



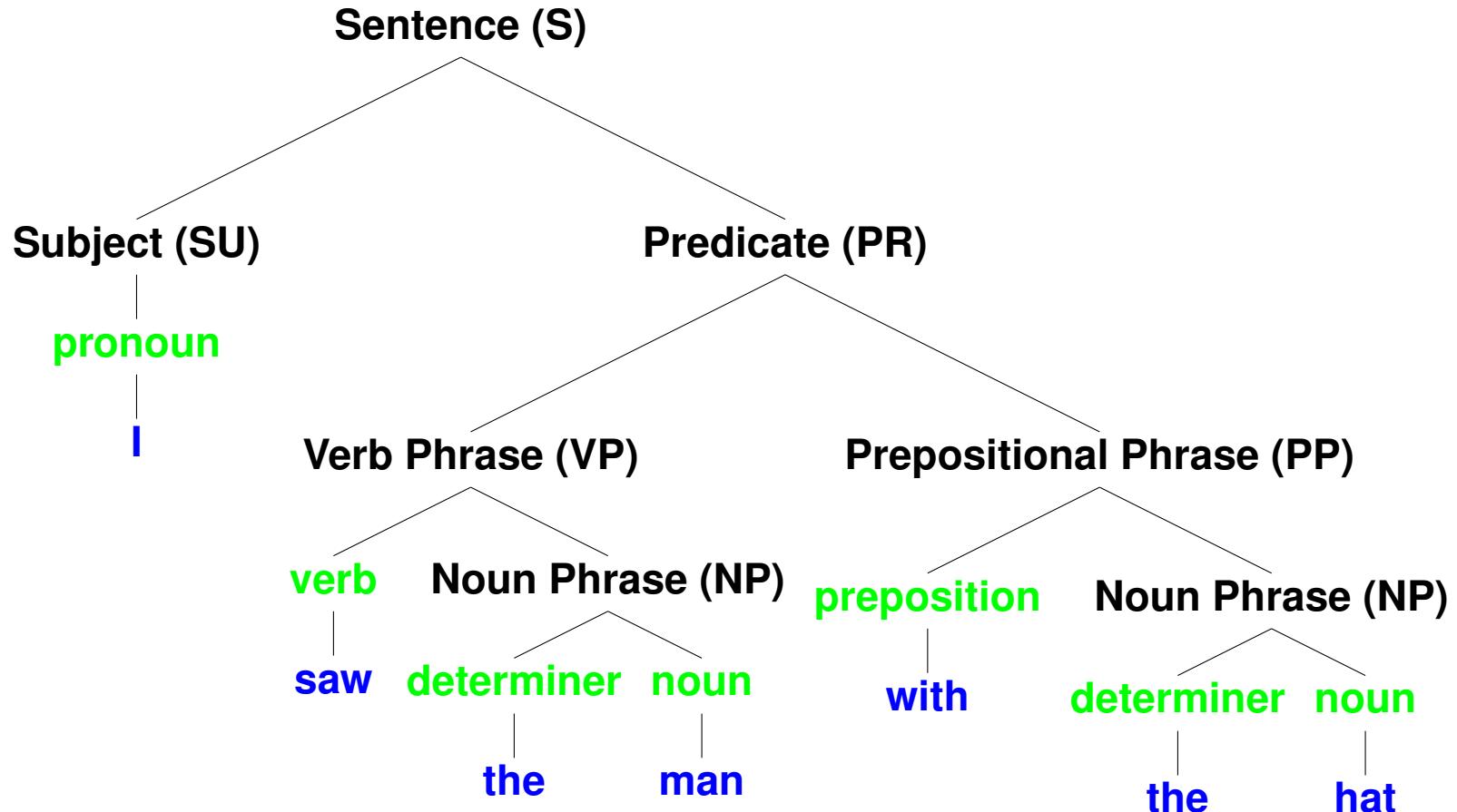
four ingredients:

- **performance measure: error measure (e.g. edit distance)**
we have to decide how to judge the quality of the system output
- **probabilistic models with suitable structures (*machine learning*):**
to capture the dependencies within and between input and output strings
 - elementary observations: Gaussian mixtures, log-linear models, support vector machines (SVM), artificial neural nets (ANN), ...
 - strings: n -gram Markov chains, CRF, Hidden Markov models (HMM), recurrent neural nets (RNN), LSTM RNN, ANN-based models of attention, end-to-end modelling, ...
- **training criterion (*machine learning*):**
to learn the free model parameters from examples
 - ideally should be linked to performance criterion (*end-to-end training*)
 - might result in complex mathematical optimization (efficient algorithms!)
 - extreme situation: number of free parameters vs. observations
- **Bayes decision rule:**
to generate the output word sequence
 - combinatorial problem (efficient algorithms)
 - should exploit structure of models**examples: dynamic programming and beam search, A* and heuristic search, ...**

History Speech Recognition 1975-today

- steady increase of challenges:
 - vocabulary size: 10 digits ... 1000 ... 10.000 ... 500.000 words
 - speaking style: read speech ... colloquial/spontaneous speech
- steady improvement of statistical methods:
HMM, Gaussians and mixtures, statistical trigram language model, adaptation methods, artificial neural nets, ...
- 1985-93: criticism about statistical approach
 - too many parameters and saturation effect
 - ... 'will never work for large vocabularies' ...
- remedy(?) by rule-based approach:
 - language models (text): linguistic grammars and structures
 - phoneme models (speech): acoustic-phonetic expert systems
 - limited success for various reasons:
huge manual effort is required!
problem of coverage and consistency of rules
- evaluations: experimental tests:
 - the same evaluation criterion on the same test data
 - direct comparison of algorithms and systems
- around 2011: second renaissance of ANNs and success of deep learning

- principle:



- extensions along many dimensions

dichotomy until 1990-2000:

- speech: signals → statistics (engineers, industrial labs)
- text: symbols → rules (linguists, universities)

use of statistics has been controversial in text processing
(symbolic processing and computational linguistics):

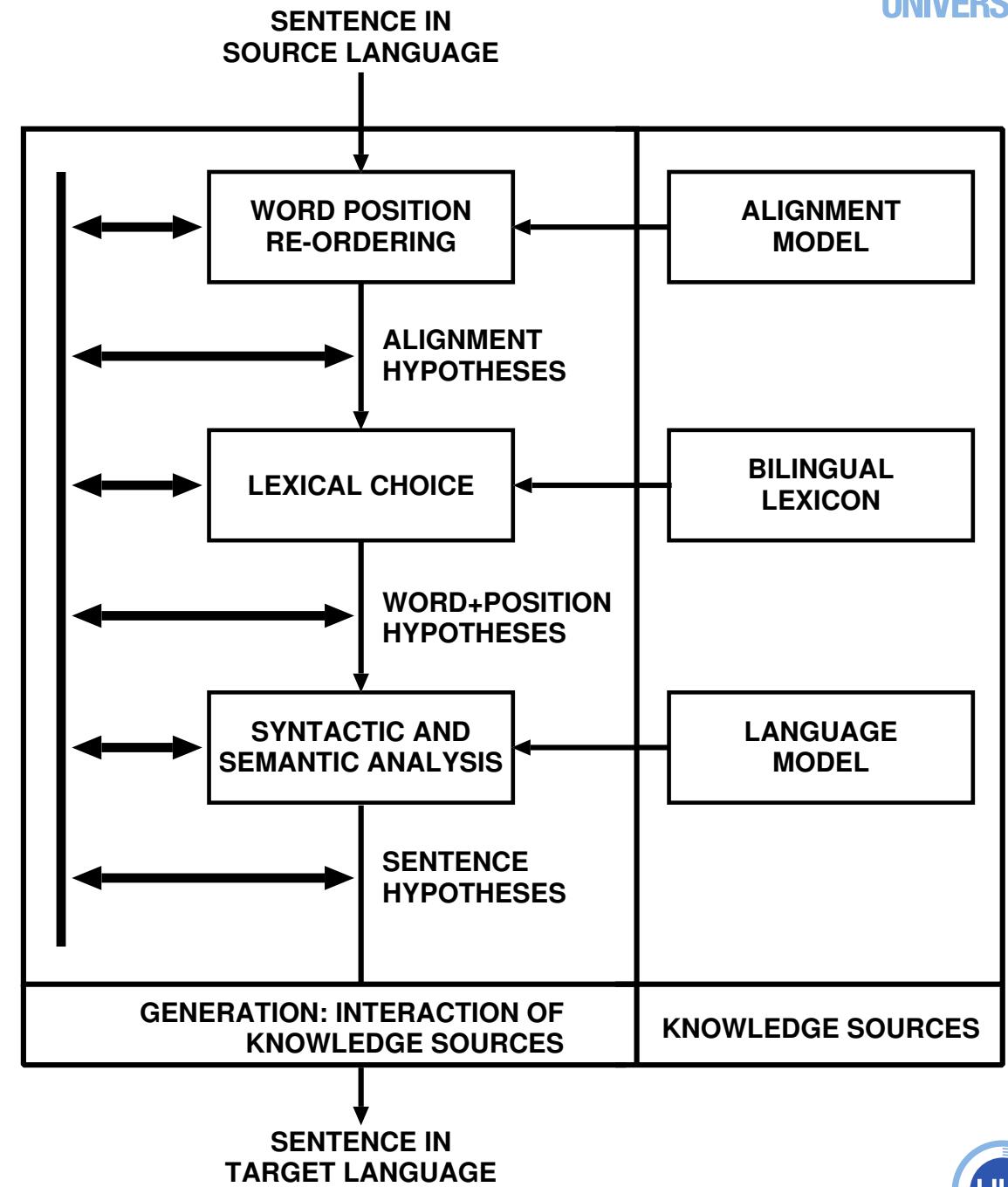
- Chomsky 1969:
... the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term.
- was considered to be true by most experts in (rule-based) human language technology and artificial intelligence
- result until 2000: the rule-based approach dominated artificial intelligence (AI) and natural language processing (NLP)
- (potential) problem of rule-based approach:
 - coverage and consistency of rules
 - result: huge human effort required in practice

history of data-driven approach to MT:

- 1989-93: pioneering work at IBM Research: machine learning approach:
 - learning from bilingual sentence pairs
 - probabilistic models
 - Bayes decision rule
- 1994: key people (R. Mercer, P. Brown) left for *Renaissance Technologies* (hedge fund)
- 1995-2000: only a few teams advocated statistical MT:
RWTH Aachen, UP Valencia, HKUST Hong Kong, CMU Pittsburgh
- improvements beyond IBM's approach:
 - additional alignment model HMM
 - efficient implementation and toolkit GIZA++
 - phrase-based concept
 - using bilingual phrase pairs (learned automatically!)
 - log-linear model combination
- 2002: DARPA project TIDES and comparative evaluations
- 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]

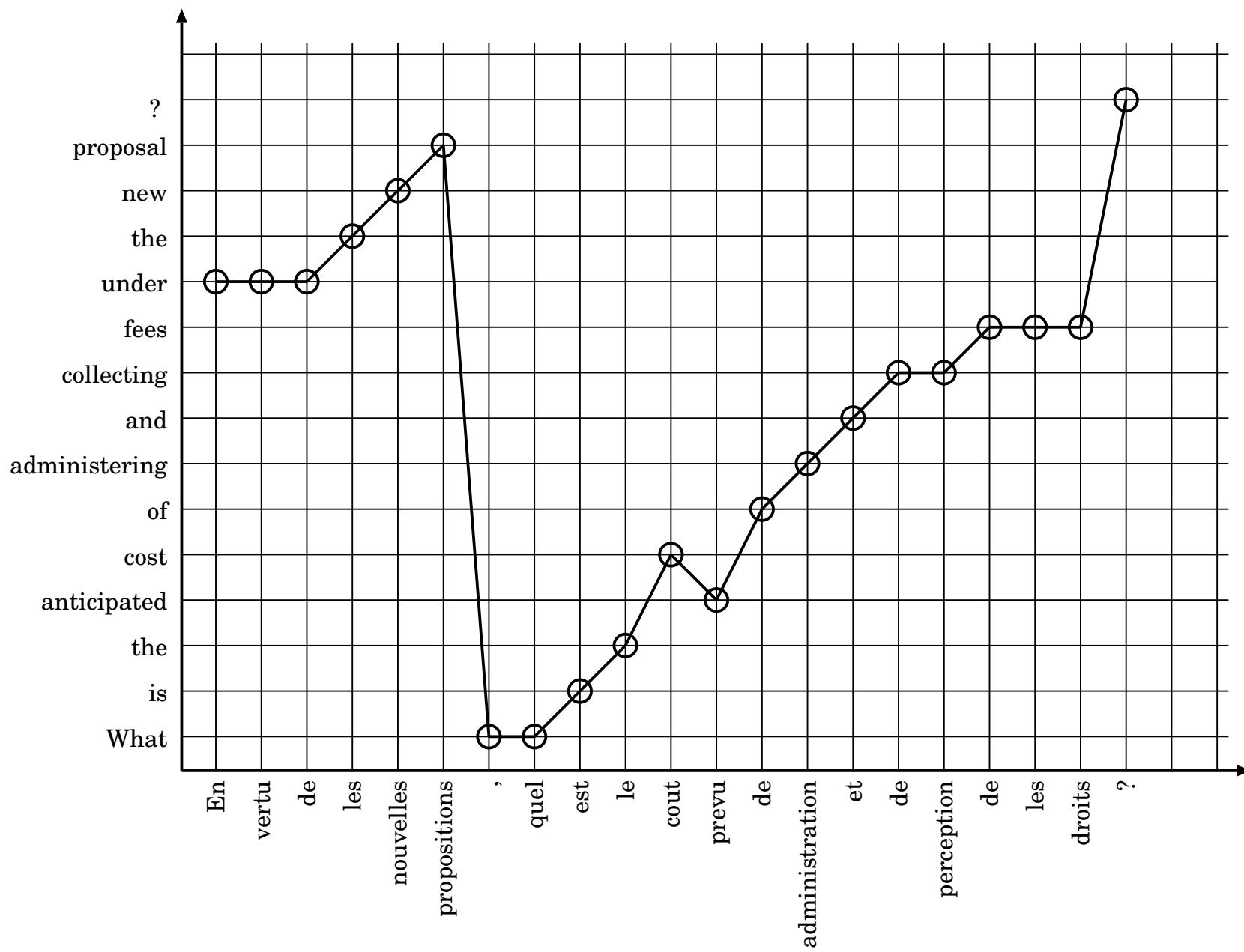
illustration: machine translation

- interaction between three models (or knowledge sources):
 - alignment model $p(A|E)$
 - lexicon model $p(E|F, A)$
 - language model $p(E)$
- handle interdependences, ambiguities and conflicts by Bayes decision rule as for speech recognition



Hidden Markov Models for MT: Word Alignments

(Canadian Parliament; IBM 1993)



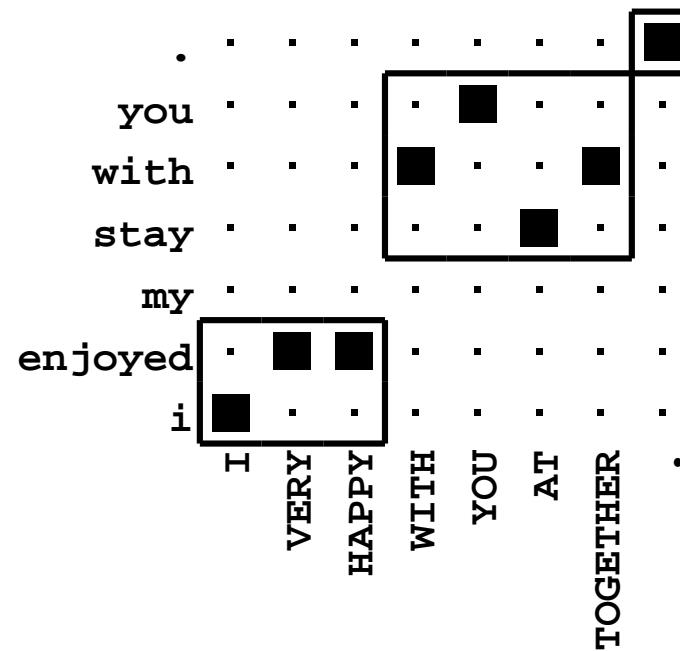
From Single Words to Word Groups (Phrases) (RWTH 1998-2002)

source sentence 我 很 高 兴 和 你 在 一 起 .

gloss notation I VERY HAPPY WITH YOU AT TOGETHER .

target sentence I enjoyed my stay with you .

best alignment for source → target language:



phrase-based approach:

- **training:** extraction
of phrase pairs (= two-dim. 'blocks')
after alignment/lexicon
training
 - **translation process:**
phrases are the smallest units

	source positions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	
--	------------------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	--

- **error measure $L[\tilde{c}, c]$ and Bayes decision rule for optimal performance:**

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

- **probability model:**

assume model $p_\vartheta(\tilde{c}|x)$ to replace true distribution $pr(\tilde{c}|x)$

- **training criterion (+ efficient algorithm):**

learn parameters ϑ from training data

- **pseudo-Bayes decision rule (+ efficient algorithm):**

generate the decision:

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

central role of performance measure or classification error
for ASR (and other NLP Tasks):

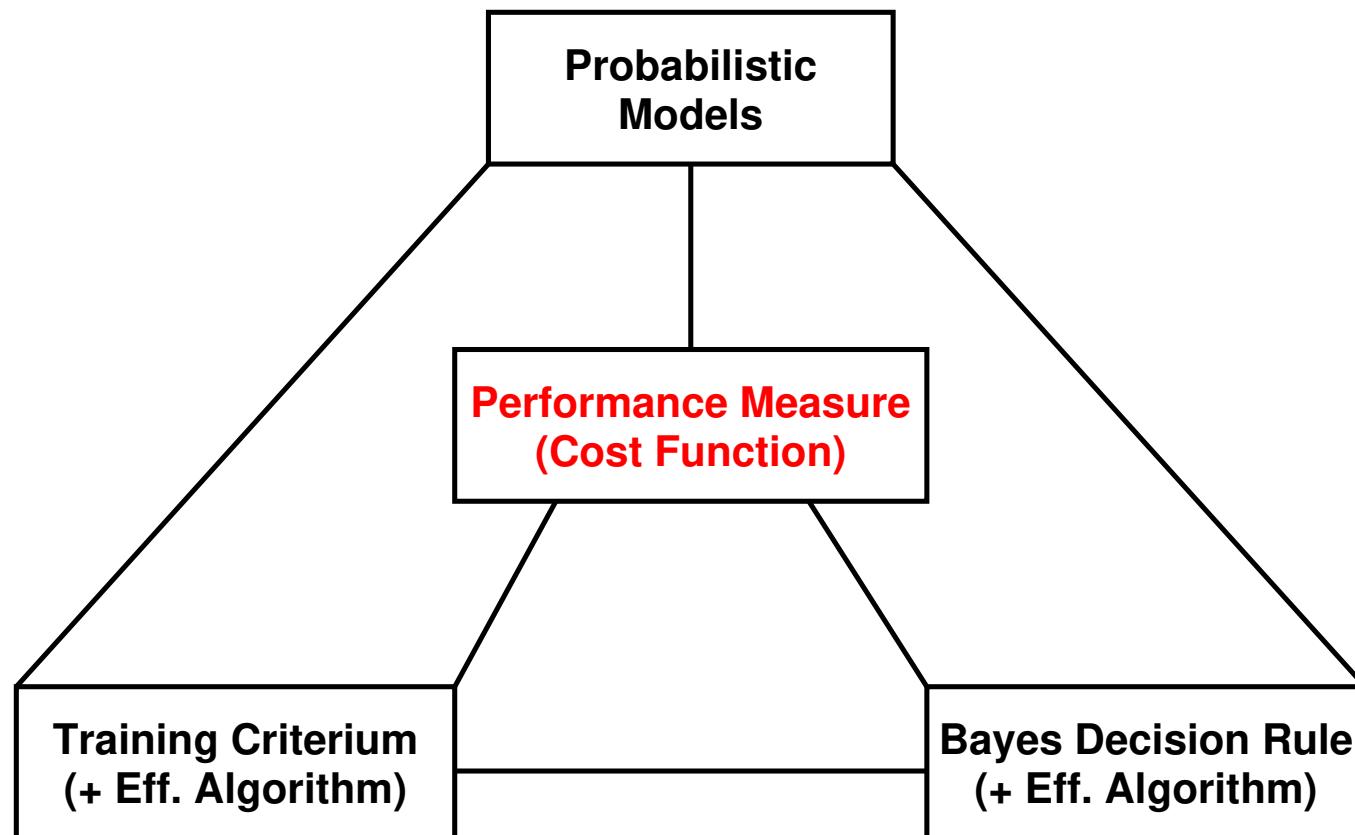
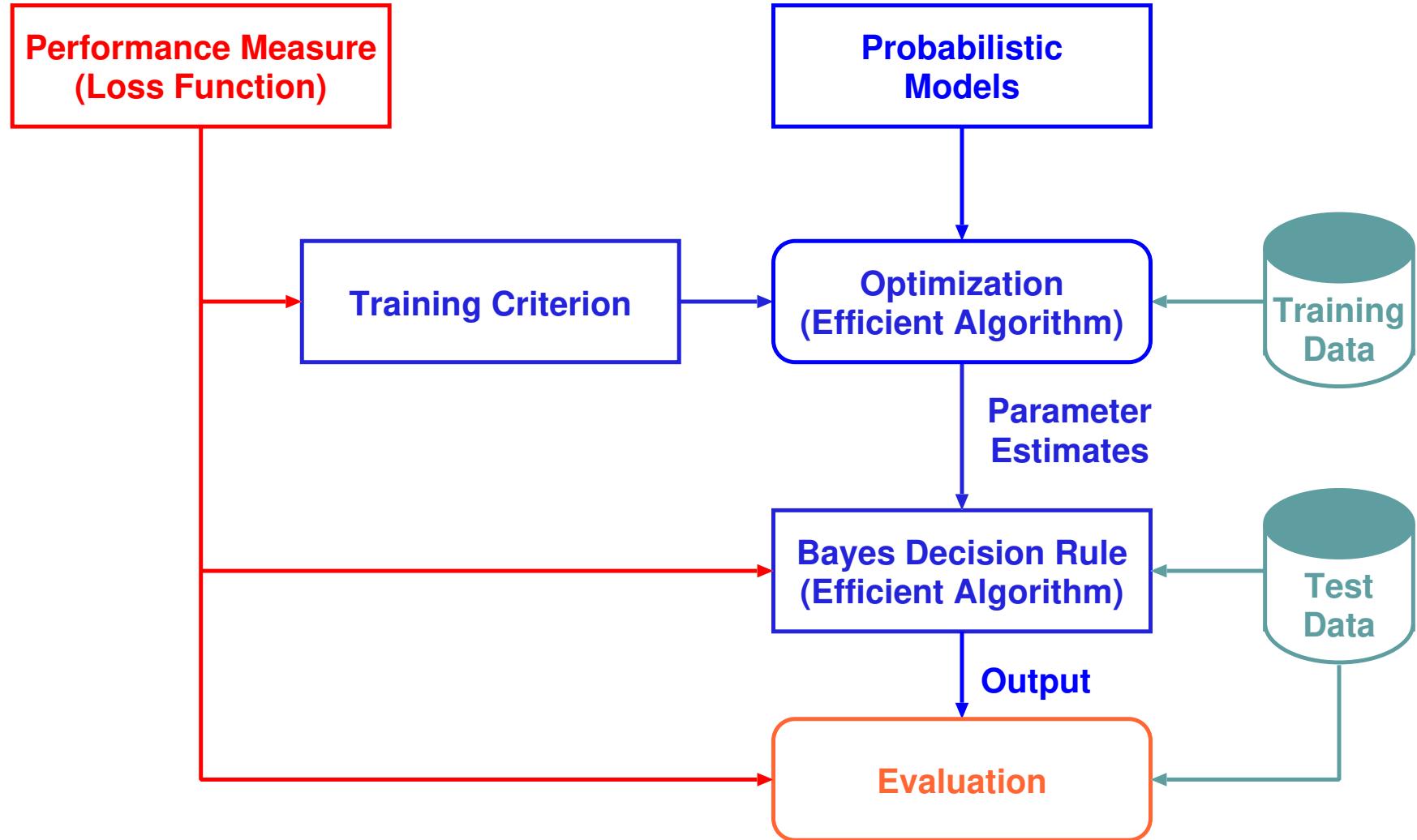


Illustration of the Statistical Approach



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 Bayes' decision rule:

- probability models:
 - 'incorrect' observation x : only an incomplete part or a poor transformation of the true observations is used
 - incorrect models, e.g. $p_\vartheta(c|x)$ or $p_\vartheta(c_1^N|x_1^T)$
- training conditions:
 - poor training criterion
 - not enough training data
 - mismatch conditions between training and test data
- training criterion + efficient algorithm:
 - suboptimal algorithm for training (e.g. gradient descent)
- decision rule:
 - incorrect error measure, e.g. MAP rule in ASR and MT
- decision rule + efficient algorithm:
 - suboptimal search procedure, e.g. beam search or N-best lists

2 Neural Networks and Deep Learning

2.1 Principles

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'):**

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

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

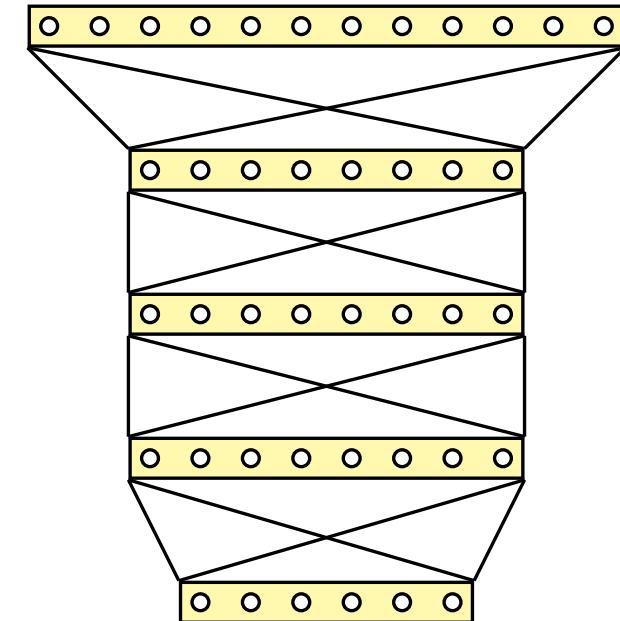
typical ANN structure:

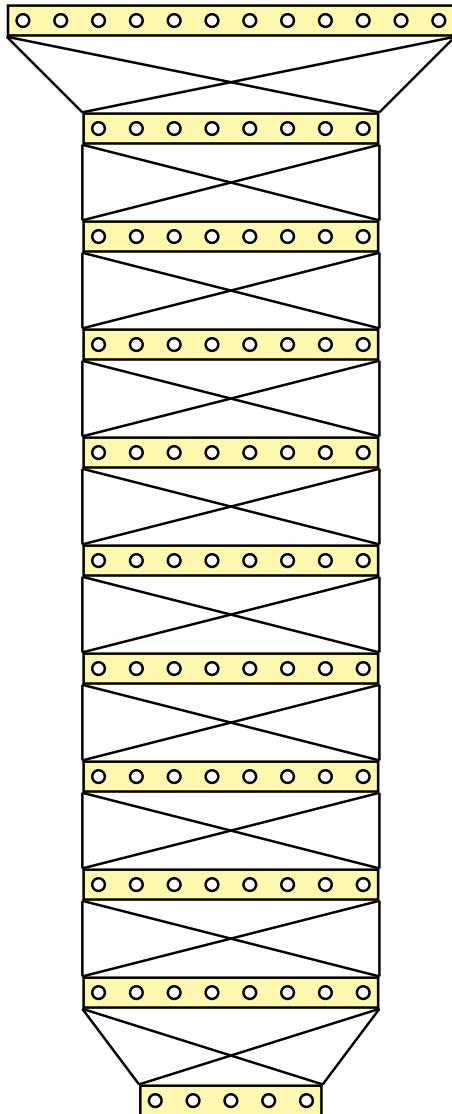
MLP: feedforward multi-layer perceptron
with input, hidden and output layers

theoretical results (?):

one hidden layer should be sufficient (!?)

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





question: what is different now after 30 years?

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

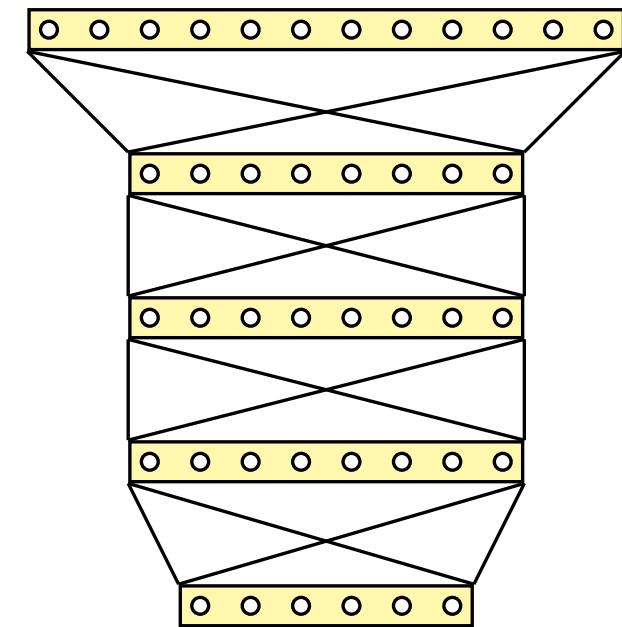
- **more powerful hardware**
(e. g. GPUs)
- **empirical recipes for optimization:**
practical experience and heuristics,
e.g. layer-by-layer pretraining
- **result: we are able to handle more complex architectures**
(deep MLP, RNN, LSTM-RNN, etc.)

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):**

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

- **hyperbolic tangent:**

$$\begin{aligned} u \rightarrow \tanh(u) &= \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)} \quad \in [-1, 1] \\ &= 2\sigma(2u) - 1 \end{aligned}$$

- **in principle: no difference between $\sigma(\cdot)$ and $\tanh(\cdot)$**
- **in practice: difference due to side effects**
- **rectifying linear unit (ReLU):** $u \rightarrow r(u) = \max\{0, u\}$
so far: not useful in symbolic processing (?)
- **softmax function: generates normalized output (for probability distribution)
for each node c of the layer under consideration (typically: output layer):**

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

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

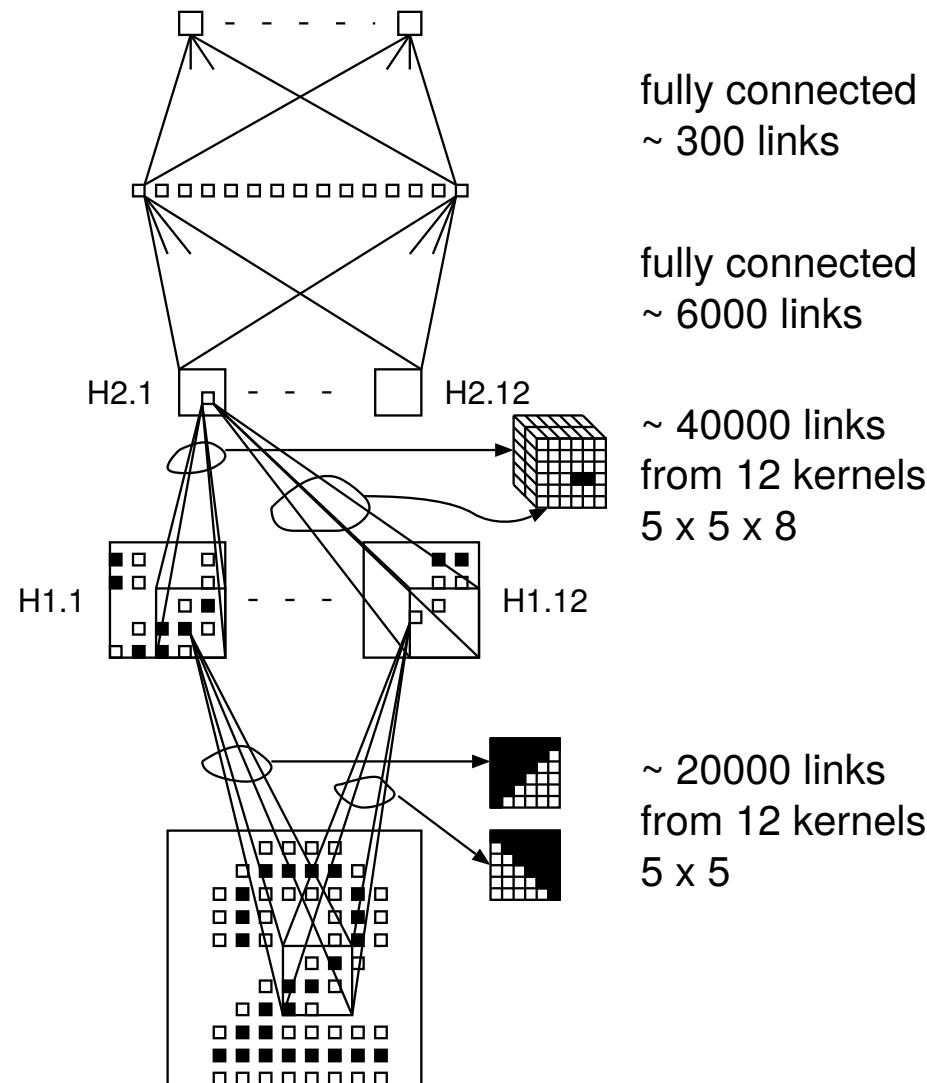
10 output units

layer H3
30 hidden units

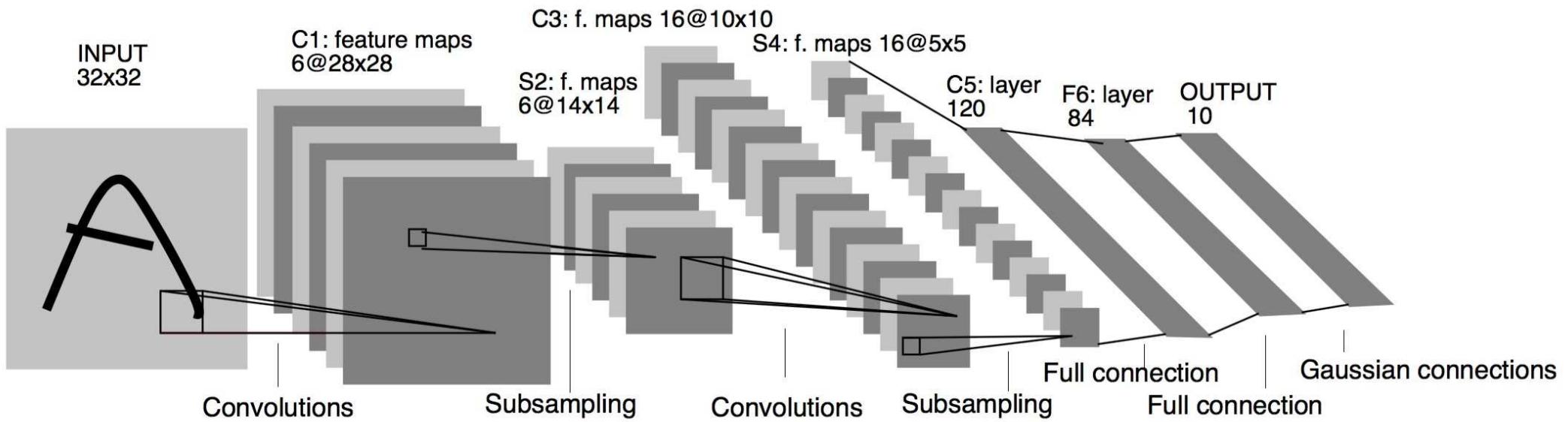
layer H2
 $12 \times 16 = 192$
hidden units

layer H1
 $12 \times 64 = 768$
hidden units

256 input units



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



2.2 Training Data and Empirical Distribution

training data: a (representative) sample of pairs (x, c) :

$$(x_n, c_n), \quad n = 1, \dots, N$$

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

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 $E\{h(x, c)\}$:

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

note: the empirical distribution summarizes EVERYTHING about the training data

- **joint distribution of pairs (x, c) :**

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

- **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 $pr(c) > 0$:**

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

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) $(x_n, c_n), n = 1, \dots, N$
 - to a functional dependence: $(x_n, \hat{c}_n = \hat{c}_{x_n})$
 - using a suitable decision rule: $x \rightarrow \hat{c}_x$

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

preview of important results:

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

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

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 $x \in \mathbb{R}^D$**

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

note: quadratic in x and linear in parameters $\vartheta = \{\alpha_c \in \mathbb{R}, \lambda_c \in \mathbb{R}^D, \Lambda_c \in \mathbb{R}^{D \times D}\}$

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

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

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

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

note: the decision output does not change:

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

approach:

- assume an ANN model $p_\vartheta(c, x) \in \mathbb{R}$ with free parameters ϑ
(or any other discriminant structure) without any (normalization) constraint
- training concept for ideal outputs:

- 1 **for correct class** $c = c_n$
- 0 **for all rival classes** $c \neq c_n$

squared error criterion:

$$\begin{aligned} F(\vartheta) &:= \frac{1}{N} \sum_{n=1}^N \sum_c [p_\vartheta(c, x_n) - \delta(c, c_n)]^2 \\ &= \sum_{c',x} pr(c', x) \sum_c [p_\vartheta(c, x) - \delta(c, c')]^2 \end{aligned}$$

using the empirical distribution of the training data:

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

Squared Error: Proof

We re-write the squared error criterion:

$$\begin{aligned} F(\vartheta) &= \sum_{x,c'} pr(c',x) \sum_c [p_\vartheta(c,x) - \delta(c,c')]^2 \\ &= \sum_x pr(x) \sum_{c'} pr(c'|x) \sum_c [p_\vartheta(c,x) - \delta(c,c')]^2 = \sum_x pr(x) F(\vartheta|x) \end{aligned}$$

using the LOCAL squared error $F(\vartheta|x)$:

$$\begin{aligned} F(\vartheta|x) &:= \sum_c pr(c|x) \sum_{c'} \left[p_\vartheta(c',x) - \delta(c',c) \right]^2 \\ &= \sum_c pr(c|x) \sum_{c'} \left[p_\vartheta^2(c',x) - 2 p_\vartheta(c',x) \delta(c',c) + \delta^2(c',c) \right] \\ &= \sum_c \sum_{c'} \left[pr(c|x) p_\vartheta^2(c',x) - 2 pr(c|x) p_\vartheta(c',x) \delta(c',c) + pr(c|x) \delta^2(c',c) \right] \\ &= \sum_c p_\vartheta^2(c,x) - 2 \sum_c pr(c|x) p_\vartheta(c,x) + 1 \\ &= 1 - \sum_c pr^2(c|x) + \sum_c [pr(c|x) - p_\vartheta(c,x)]^2 \end{aligned}$$

Identity for Squared Error: Minimalistic Proof

**assumption: normalized distribution p_c , arbitrary q_c (can be negative!)
[Patterson & Womack 66, Bourlard & Wellekens 89]**

we re-write the squared error criterion:

$$\begin{aligned} \sum_c p_c \sum_{c'} [q_{c'} - \delta_{c'c}]^2 &= \sum_c p_c \sum_{c'} [q_{c'}^2 - 2 q_{c'} \delta_{c'c} + \delta_{c'c}^2] \\ &= \dots \text{(carry out sums over } c \text{ and } c') \\ &= \sum_c q_c^2 - 2 \sum_c p_c q_c + 1 \\ &= \sum_c [q_c - p_c]^2 + 1 - \sum_c p_c^2 \end{aligned}$$

**note: the absolute minimum of the squared error is the Gini index,
also referred to as *residual error*:**

$$\min_{\{q_c\}} \left\{ \sum_c p_c \sum_{c'} [q_{c'} - \delta_{c'c}]^2 \right\} = 1 - \sum_c p_c^2$$

summary of re-writing the squared error:

$$\begin{aligned}
 F(\vartheta) &:= \frac{1}{N} \sum_{n=1}^N \sum_c [p_\vartheta(c, x_n) - \delta(c, c_n)]^2 \\
 &= \sum_{c',x} pr(c', x) \sum_c [p_\vartheta(c, x) - \delta(c, c')]^2 \\
 &= \sum_x pr(x) \left(1 - \sum_c pr^2(c|x) + \sum_c [pr(c|x) - p_\vartheta(c, x)]^2 \right)
 \end{aligned}$$

training criterion for parameter ϑ :

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

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

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

note: normalization comes for free!

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 probabilities (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.

decision trees or CART (classification and regression trees)

we apply the minimum squared error criterion to CART
and obtain the Gini criterion as impurity function:

- **CART: binary tree structure is used
to define equivalence classes for the input: $x \rightarrow g_x \in \{0, 1\}$**

correct notation:

replace x by g_x in $pr(c|g_x)$ and $p_\vartheta(c, g_x)$

- **'model' learning for each split hypothesis
using minimum squared error training:**

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

$$\min_{\{p_\vartheta(c, x)\}} F(\vartheta) = 1 - \sum_c pr^2(c|x)$$

- **result: the best split is selected according to the Gini criterion,
which is the residual error (= value of the minimum) for the squared error criterion.**

we have shown for a model with sufficient degrees of freedom:

optimum ANN output = class posterior of training data

under the conditions:

- no normalization at all
- normalized model outputs

we consider the case of quadratic normalization

$$\sum_c p_\vartheta^2(c, x) = 1$$

and re-write the squared error criterion:

$$\begin{aligned} F(\vartheta) &= \sum_{c',x} pr(c',x) \sum_c [p_\vartheta(c,x) - \delta(c,c')]^2 \\ &= \sum_x pr(x) \left(1 - \sum_c pr^2(c|x) + \sum_c [pr(c|x) - p_\vartheta(c,x)]^2 \right) \\ &= 2 \cdot \sum_x pr(x) \left(1 - \sum_c pr(c|x) p_\vartheta(c,x) \right) \end{aligned}$$

Squared Error Criterion: Quadratic Normalization

for quadratic normalization, we have shown:

$$\begin{aligned} F(\vartheta) &= 2 \cdot \sum_x pr(x) \left(1 - \sum_c pr(c|x) p_\vartheta(c, x) \right) \\ &= 2 \cdot \left(1 - \sum_{c,x} pr(c, x) p_\vartheta(c, x) \right) \end{aligned}$$

training criterion for parameter ϑ :

$$\begin{aligned} \hat{\vartheta} &= \underset{\vartheta}{\operatorname{argmin}} F(\vartheta) = \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_{c,x} pr(c, x) p_\vartheta(c, x) \right\} \\ &= \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_n p_\vartheta(c_n, x_n) \right\} \end{aligned}$$

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

$$\begin{aligned} \hat{p}_{\hat{\vartheta}}(c, x) &= \underset{\{p_\vartheta(c, x)\}}{\operatorname{argmin}} F(\vartheta) = \frac{pr(c|x)}{\sqrt{\sum_{c'} pr^2(c'|x)}} \\ \min_{\{p_\vartheta(c, x)\}} F(\vartheta) &= 2 \cdot \left(1 - \sum_x pr(x) \sqrt{\sum_c pr^2(c|x)} \right) \end{aligned}$$

result: a re-normalized class posterior

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

$$p_{\vartheta}(c|x) > 0 \quad \sum_c p_{\vartheta}(c|x) = 1.0$$

note: different notation to express normalization

criterion: maximize the logarithm of the model posterior probability $p_{\vartheta}(c_n|x_n)$

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

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

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

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!

Cross-Entropy: Proof

consider the equivalent maximization problem:

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

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 optimization problems)

consider two distributions p_c, q_c over a random variable c

divergence inequality:

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

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

rewrite the divergence inequality:

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

therefore:

$$\begin{aligned} \max_{\{q_c\}} \left\{ \sum_c p_c \log q_c \right\} &= \sum_c p_c \log p_c \\ \hat{q}_c &= p_c \end{aligned}$$

problem with terminology [Solla & Levin⁺ 88]:

related names: relative divergence, relative (cross) entropy, ...

assumption: constrained ANN outputs:

$$0 < p_{\vartheta}(c, x) < 1.0$$

binary cross-entropy criterion:

$$\begin{aligned} F(\vartheta) &:= \frac{1}{N} \sum_n \left(\log p_{\vartheta}(c_n, x_n) + \sum_{c \neq c_n} \log [1 - p_{\vartheta}(c|x_n)] \right) \\ &= \dots \\ &= \sum_x pr(x) \sum_c \left(pr(c|x) \log p_{\vartheta}(c, x) + [1 - pr(c|x)] \log [1 - p_{\vartheta}(c, x)] \right) \\ &= \sum_x pr(x) \sum_c \left(pr(c|x) \sum_{c'} \log [1 - |\delta(c', c) - p_{\vartheta}(c', x)|] \right) \end{aligned}$$

binary problem: class s and its rival classes \bar{c}

consider the equivalent maximization problem:

$$\Delta F(\vartheta) := \sum_x pr(x) \sum_c \left(pr(c|x) \log \frac{p_\vartheta(c,x)}{pr(c|x)} + [1 - pr(c|x)] \log \frac{1 - p_\vartheta(c,x)}{1 - pr(c|x)} \right)$$

terminology: binary divergence between binary versions of true distribution and model distribution:

binary version: class c and its rival classes \bar{c}

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: normalization is for free!

proof: by using binary variant of divergence inequality

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 constrained 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

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

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 observation 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: $\left((2^8)^{50}\right)^{100} = 2^{8 \cdot 50 \cdot 100} = 2^{40.000} \cong 10^{12.000}$

concept of generalization:

- we want to define a model distribution $p_\theta(c|x)$ 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 $p_\theta(c|x)$

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

- regularization of weights: include an additive penalty in the

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

interpretation:

- large weights should be avoided
- Bayesian interpretation: Gaussian prior for weights with zero means
- terminology in backpropagation: weight decay

- Tikhonov regularization [Tikhonov & Arsenin 77]:

- outputs should be smooth in continuous-valued $x_n \in \mathbb{R}^D$
- include a penalty term:

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

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

Training Criteria and Backpropagation

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

Counterexamples: ANN Output \neq Class Posterior

we have shown so far:

$$\text{optimal ANN output} = \text{class posterior}$$

open question: are other optimal ANN outputs possible?

example:

- consider a non-negative model $p_\vartheta(c, x)$ with quadratic normalization:

$$p_\vartheta(c, x) > 0 : \quad \sum_c p_\vartheta^2(c, x) = 1$$

- training criterion: squared error

result:

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

- training criterion: cross-entropy

– verify whether the optimization problem is well defined

– result:

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

2.4 Recurrent Neural Networks for Sequence Processing

2.4.1 Principle

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

problem formulation: from single events to sequences

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

$$(c_t, x_t), t = 1, \dots, T$$

with input vectors x_t and class labels c_t (with known string length T !)

- illustration:

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

model with 1:1 correspondence between class labels c_1^T and observations x_1^T (string length T is known):

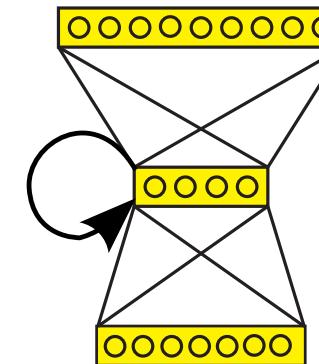
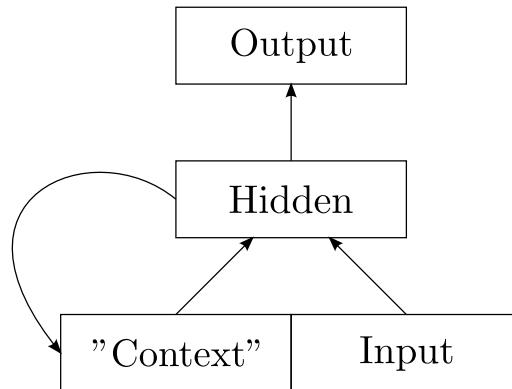
observations x_1^T :	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T
class labels c_1^T :	c_1	c_2	...	c_{t-1}	c_t	c_{t+1}	...	c_{T-1}	c_T

typical problems:

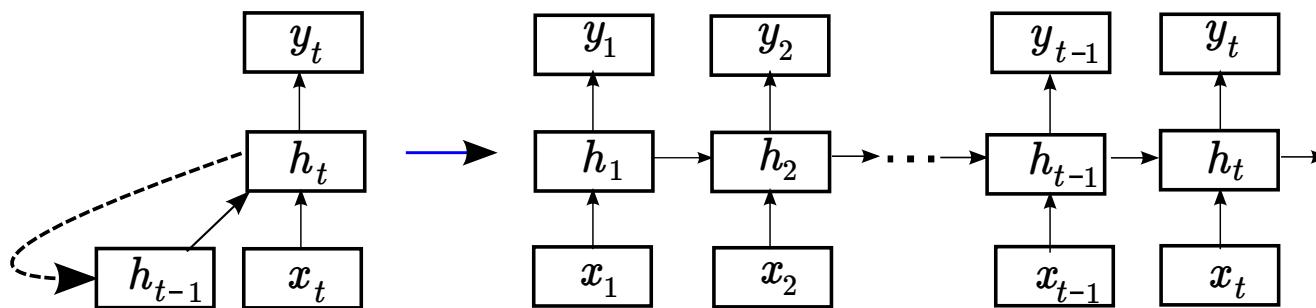
- spelling correction: for letter substitutions only
- POS tagging (POS: parts of speech = word categories) and semantic tagging for NLU
- frame labelling in ASR (incl. pronunciation and language models!) and acoustic scores in hybrid HMMs
- recognition problems with no problems of boundary detection: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

principle:

- 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

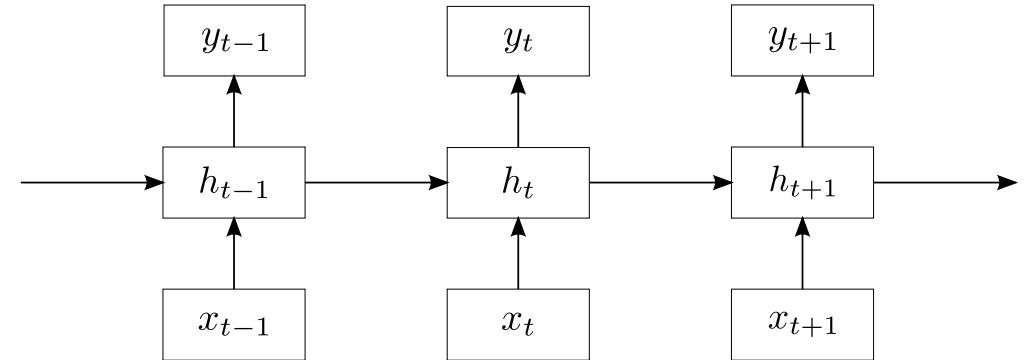


**equivalent interpretation: unfolding RNN over time:
feedforward network with a special DEEP architecture.**



operations from bottom to top:

- **output vector:** $y_t = p_\theta(c_t|x_1^t) = S(Vh_t)$
with output matrix V
- **hidden layer:** $h_t = \sigma(Ux_t + Wh_{t-1})$
with hidden layer matrix U
and recurrence matrix W
- **input vector:** x_t



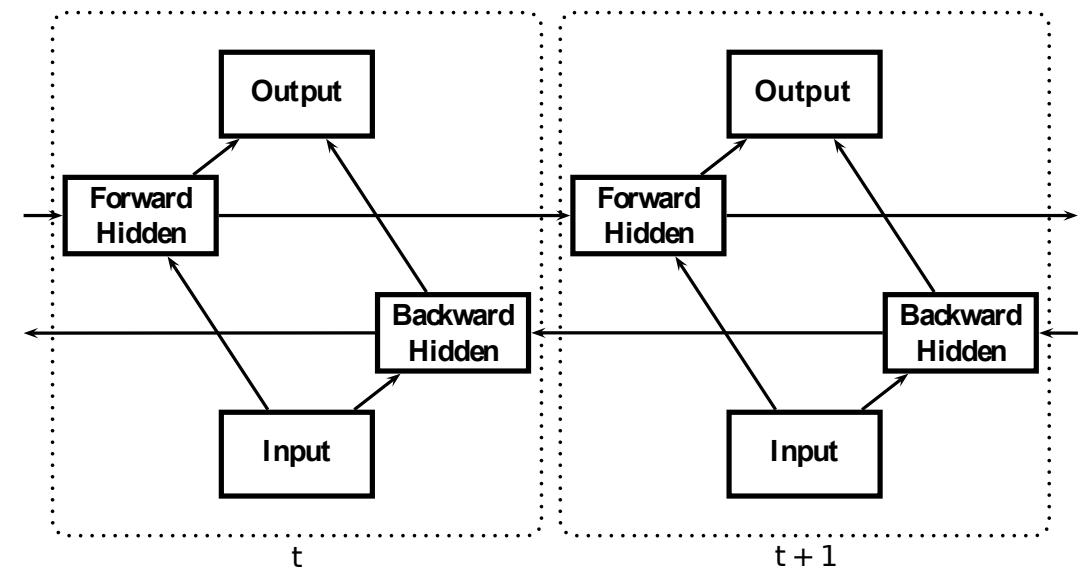
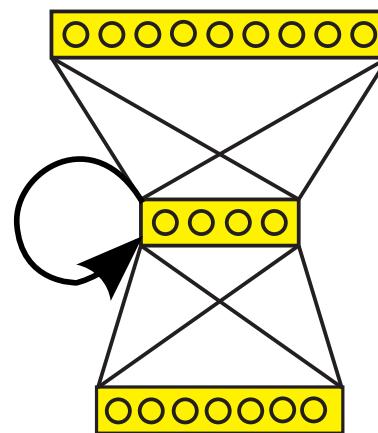
interpretation: conditional probability of RNN:

$$p(c_t|x_1^t) = p(c_t|h_{t-1}, x_t) = p(c_t|h_t)$$

with memory states h_t that group all partial sequences x_1^t into equivalence classes

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



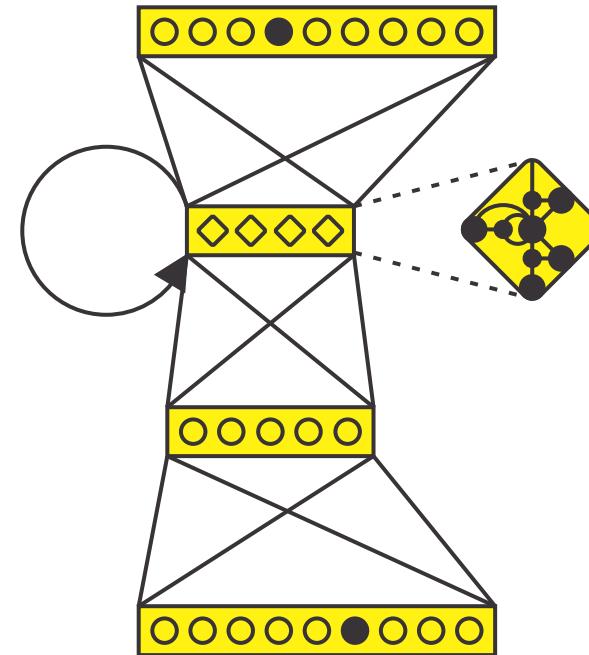
2.4.2 Extension: LSTM RNN

extension of (simple) RNN by

LSTM: long short-term memory

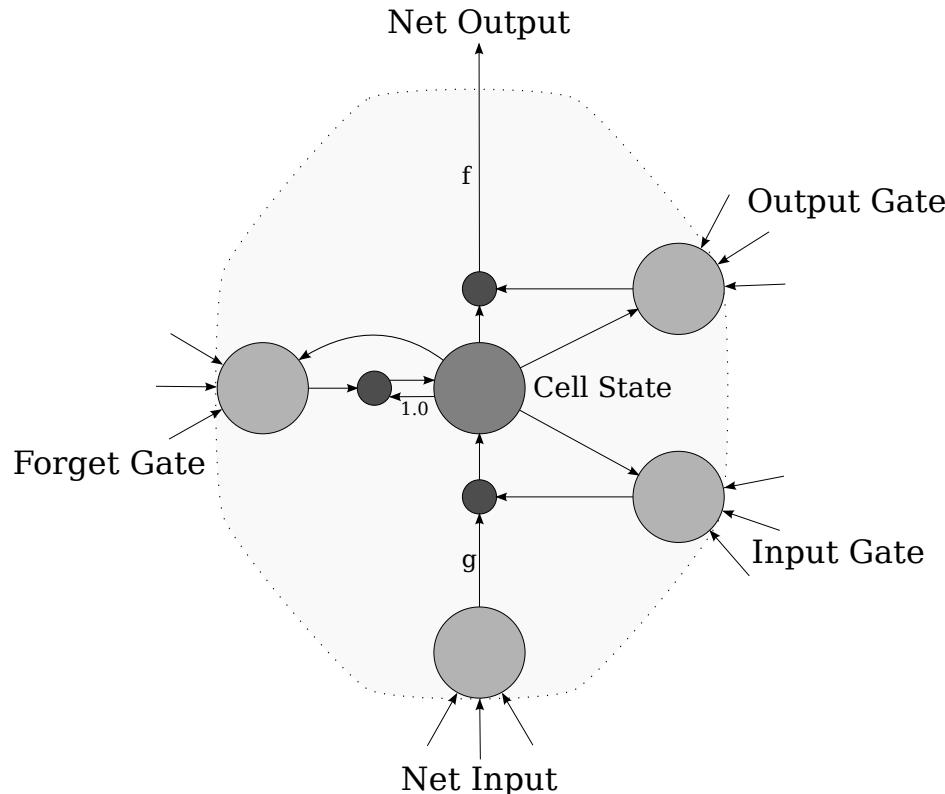
[Hochreiter & Schmidhuber 97, Gers & Schraudolph⁺ 02]

- 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



Long Short-Term Memory (LSTM) RNN

[Hochreiter & Schmidhuber 97, Gers & Schraudolph⁺ 02]



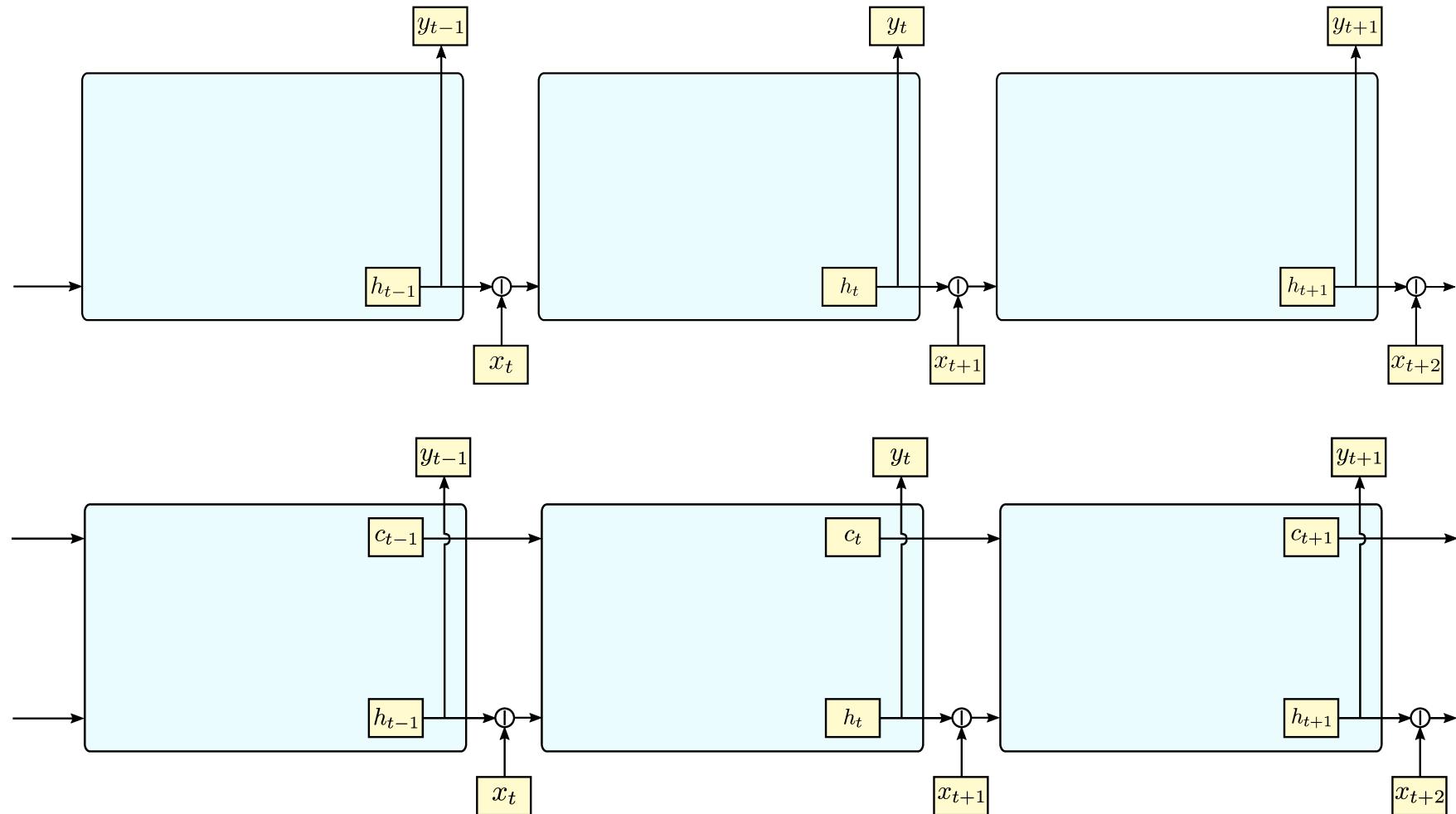
LSTM approach:

- separate (memory) cell state c_t from RNN net output vector h_t
- overall LSTM operations involve three 'input' vectors at time t : c_{t-1}, h_{t-1}, x_t
- update operations at time t :
cell state: $c_t = c_t(h_{t-1}, c_{t-1}, x_t)$
net output: $h_t = h_t(h_{t-1}, c_{t-1}, x_t)$
output layer: $y_t = y_t(h_t)$ with softmax
- introduce three gates (input, output, forget) to control the information flow

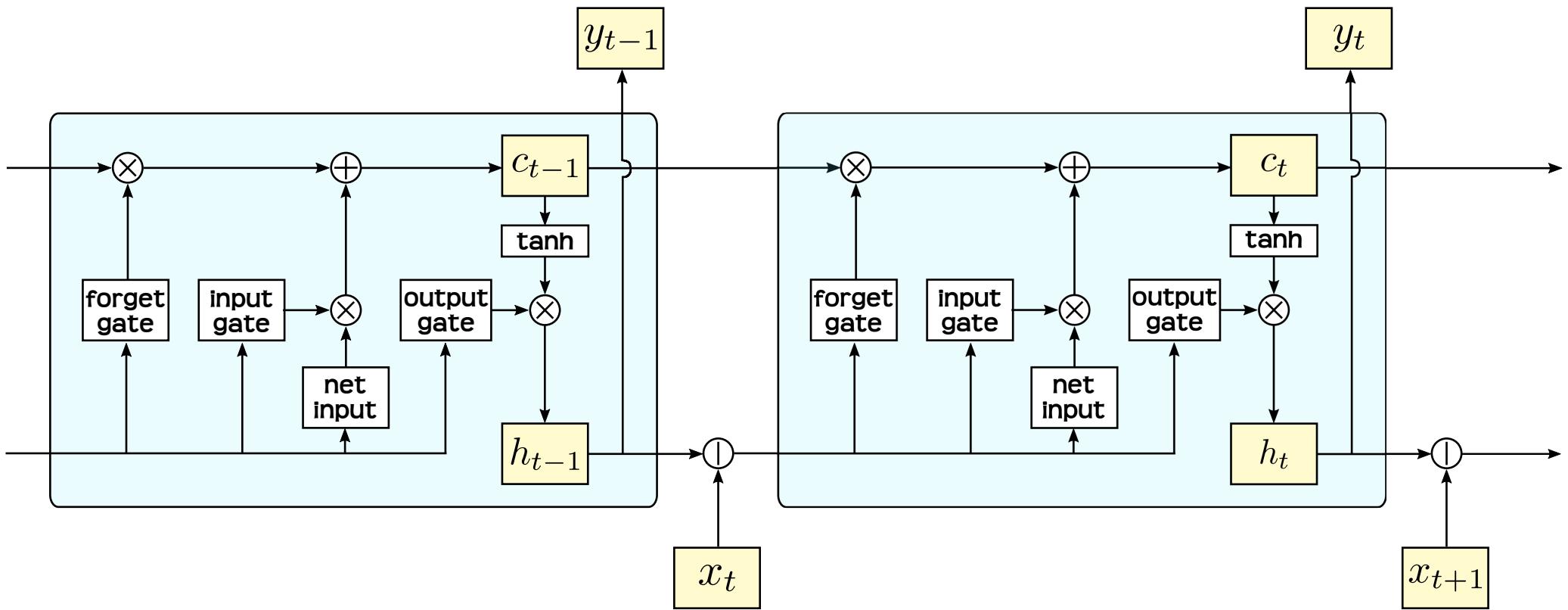
warning about notation: here c_t is NOT a class label!

Recurrent Neural Network (RNN): Extension towards Long Short-Term Memory

add a memory cell vector c_t to hidden state vector h_t :

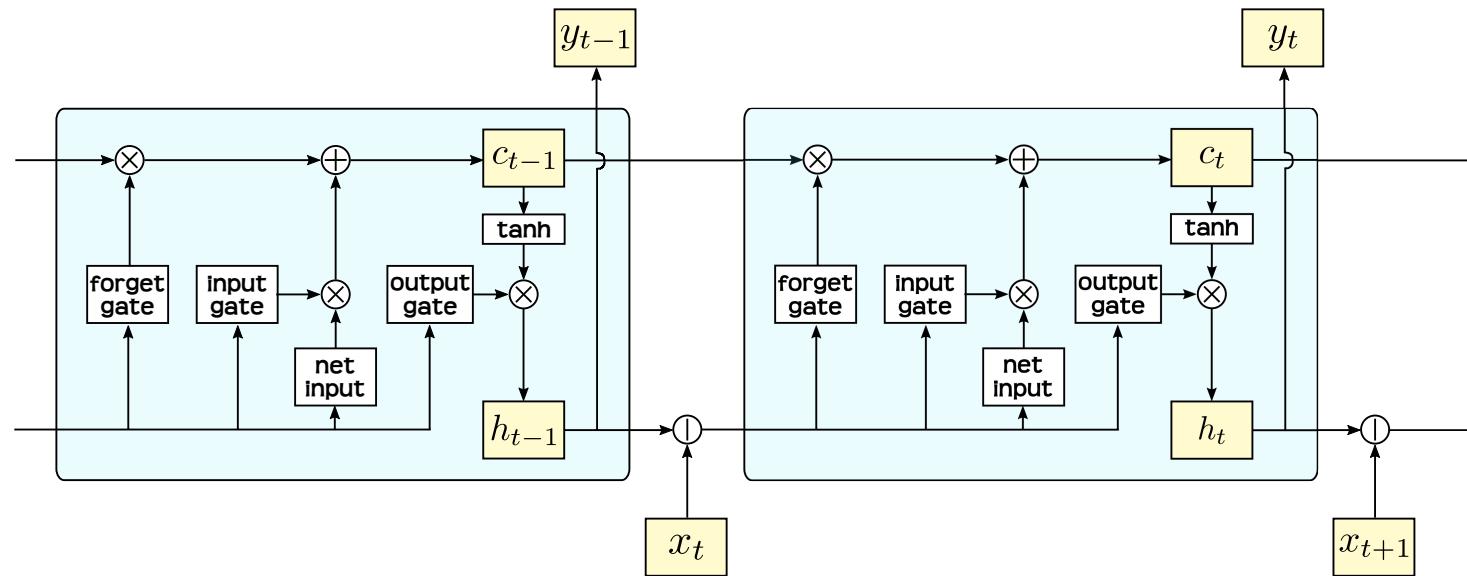


Recurrent Neural Network: Details of Long Short-Term Memory



ingredients:

- separate memory vector c_t in addition to h_t
- use of gates to control information flow
- (additional) effect: make backpropagation more robust



definitions (4 'hidden' matrices and 1 output matrix):

- concatenation of vectors: $\tilde{x}_t := [h_{t-1}, x_t]$
- always use bias/offset in dot product
- gates: component-wise multiplication

operations (without peephole connections):

- **forget gate:** $f_t = \sigma(W_f \tilde{x}_t)$
- **input gate:** $i_t = \sigma(W_i \tilde{x}_t)$
- **compute net input:** $n_t = \tanh(W_n \tilde{x}_t)$ and **update:** $c_t = f_t \odot c_{t-1} + i_t \odot n_t$
- **output gate:** $o_t = \sigma(W_o \tilde{x}_t)$ and **update:** $h_t = o_t \odot \tanh(c_t)$
- **LSTM output:** $y_t = \text{softmax}(W_y h_t)$

LSTM Architecture (obsolete)

implementation:

- three vectors (over time t): c_t, s_t, x_t
- gates (or switches): implemented as sigmoid function $\sigma(\cdot)$
- full matrices ($A_2, R; A_i, R_i, A_f, R_f, A_o, R_o$) and diagonal matrices (W_i, W_f, W_o)
- usual matrix and vector operations and element-wise multiplication \odot

- Net Input (like update formula of simple RNN):

$$z_t = \tanh(A_2x_t + Rs_{t-1})$$

- Should this Net Input z_t access the Cell State c_t ?

Input Gate: $i_t = \sigma(A_i x_t + R_i s_{t-1} + W_i c_{t-1})$

- Should the Cell State c_{t-1} be forgotten?

Forget Gate: $f_t = \sigma(A_f x_t + R_f s_{t-1} + W_f c_{t-1})$

- Based on i_t and f_t , update the Cell State c_t :

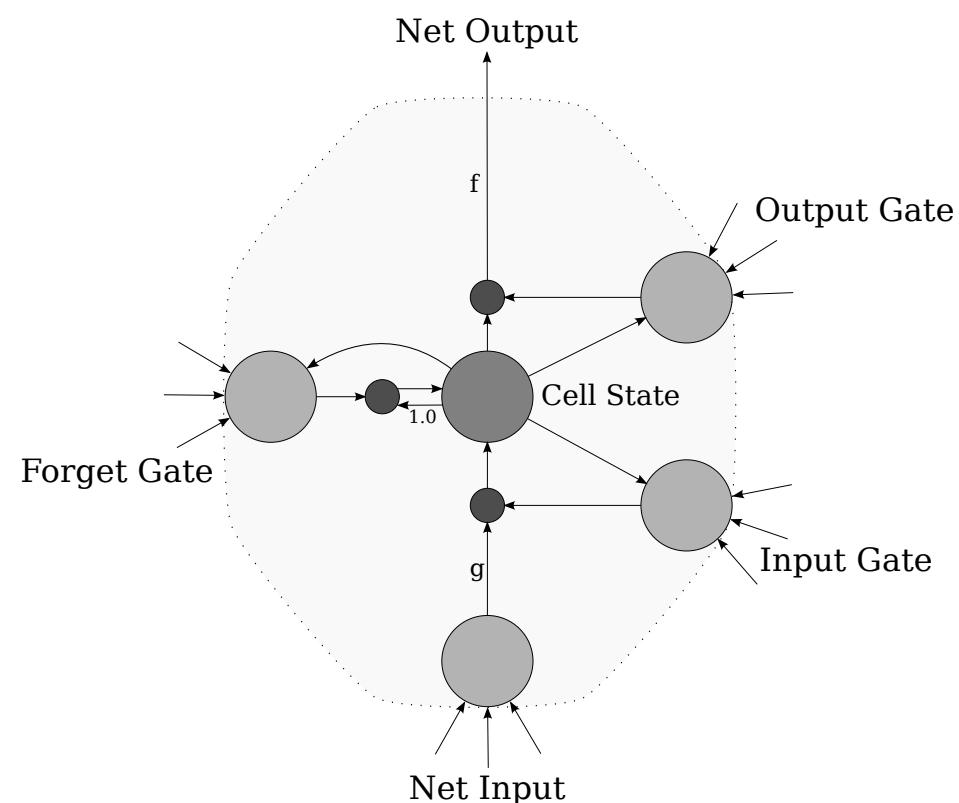
$$c_t = f_t \odot c_{t-1} + i_t \odot z_t$$

- Should this update c_t be outputted?

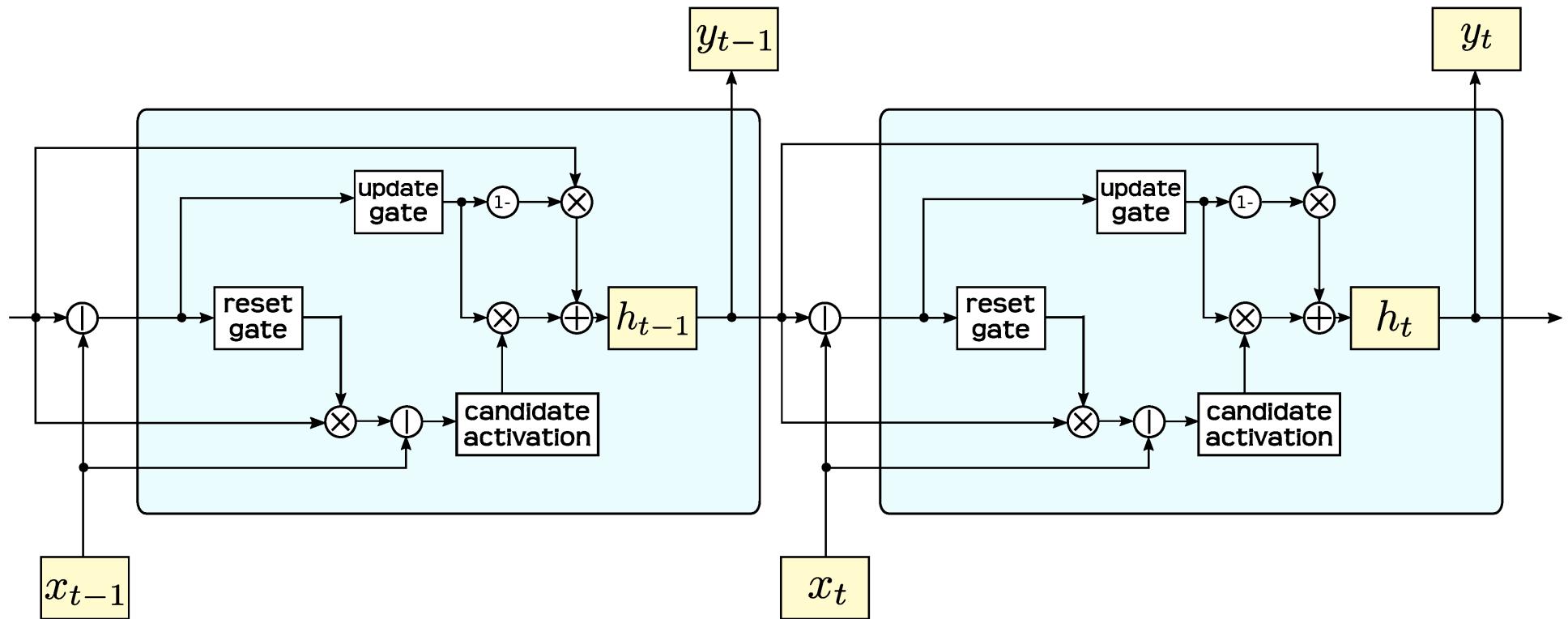
Output Gate: $o_t = \sigma(A_o x_t + R_o s_{t-1} + W_o c_t)$

- Based on o_t , compute the Net Output:

$$s_t = o_t \odot \tanh(c_t)$$



Recurrent Neural Network: Alternative to LSTM: GRU = Gated Recurrence Units



2.4.3 Interpretation of RNN Outputs

note: RNN includes LSTM RNN as a special case

two sequences over time $t = 1, \dots, T$:

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

consider the posterior probability of the output string:

$$\text{factorization over time } t: \quad p(c_1^T | x_1^T) = \prod_{t=1}^T p(c_t | c_0^{t-1}, x_1^T)$$

$$\text{marginalization for time } t: \quad \sum_{c_1^T: c_t=c} p(c_1^T | x_1^T) = p_t(c | x_1^T)$$

more ...

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

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

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 :	—	—	...	—	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}	c_t	—	...	—	—

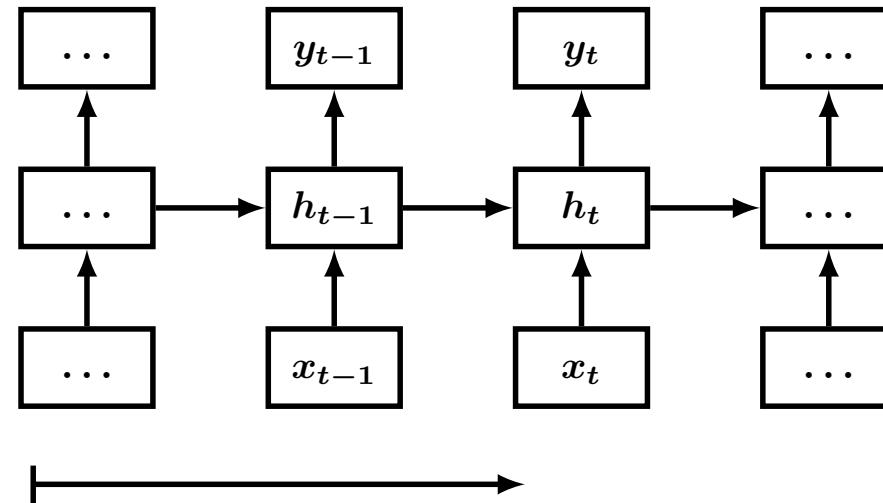
- conditional dependence in c_1^T with look-ahead in x_1^T : $p(c_1^T|x_1^T) = \prod_{t=1}^T p(c_t|c_0^{t-1}, x_1^T)$

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}	c_t	—	...	—	—

note: this is the most general case.

RNN: Variant 1

uni-directional, no feedback of output labels

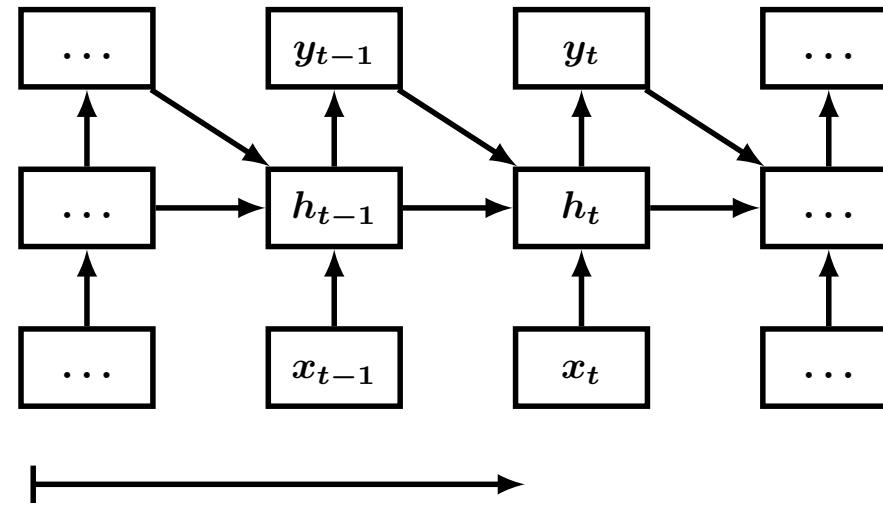


RNN output vector:

$$y_t(c) = p_t(c|x_1^t)$$

RNN: Variant 2

uni-directional, with feedback of output labels

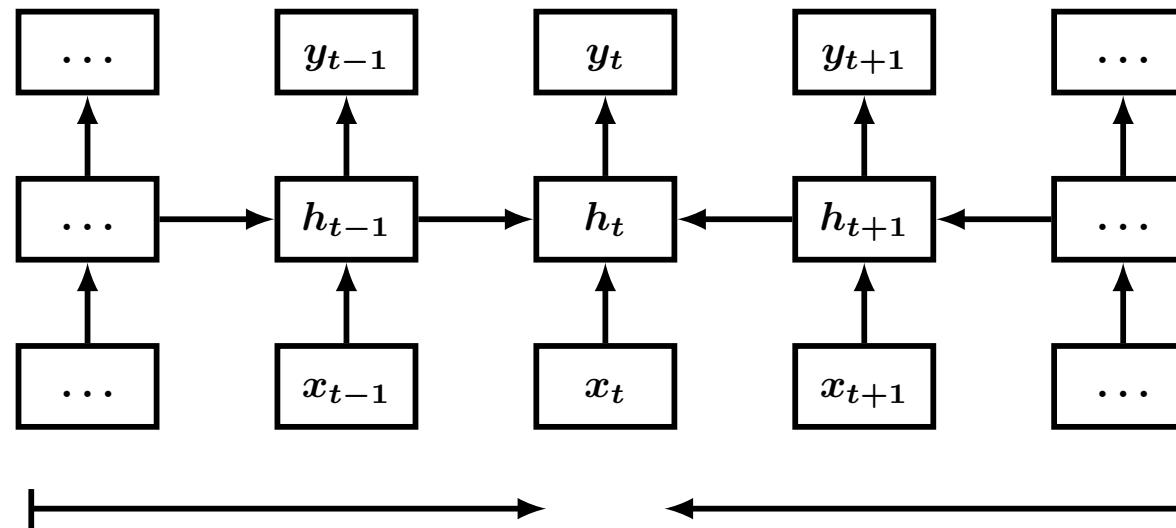


RNN output vector:

$$y_t(c) = p_t(c|c_0^{t-1}, x_1^t)$$

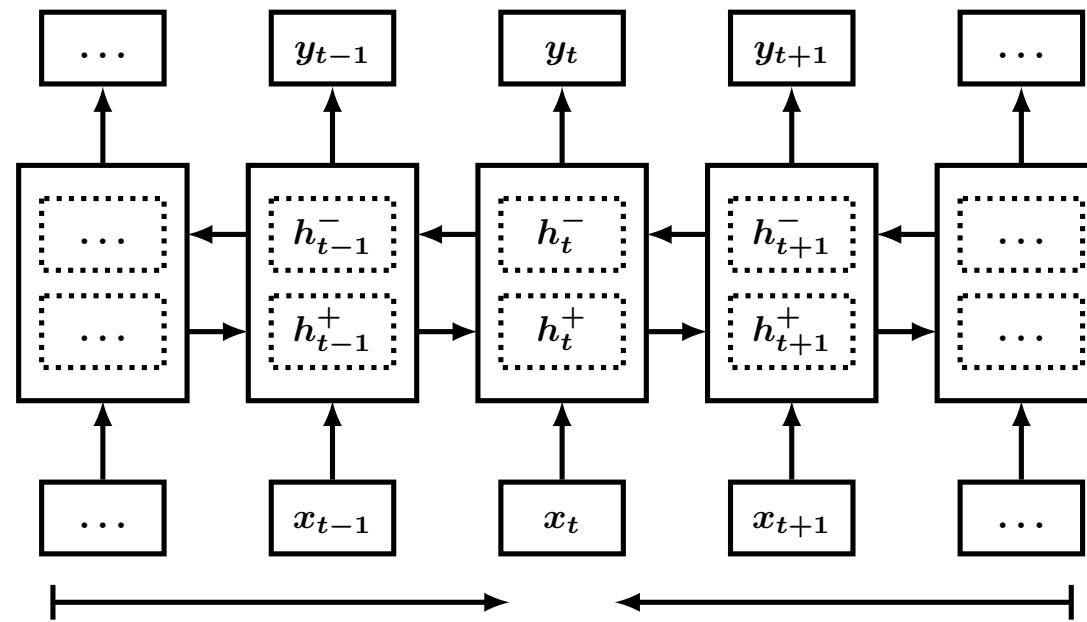
RNN: Variant 3

bi-directional, no feedback of output label



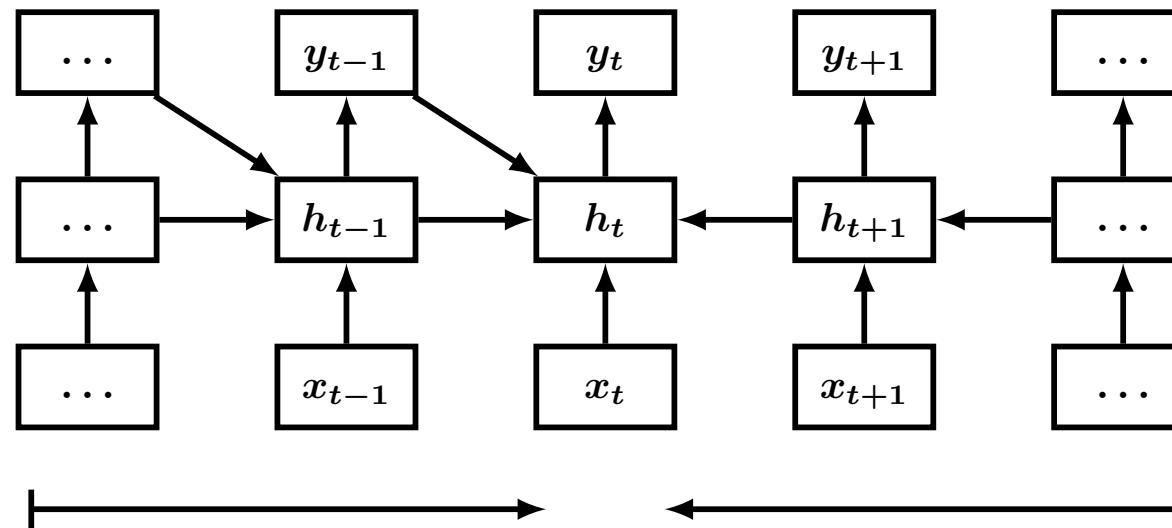
RNN: Variant 3:Forward/Backward Hidden Layer bi-directional, no feedback of output label

Internal Structure: Separate Forward and Backward Hidden Layers



RNN: Variant 4

bi-directional, with uni-directional feedback of output label



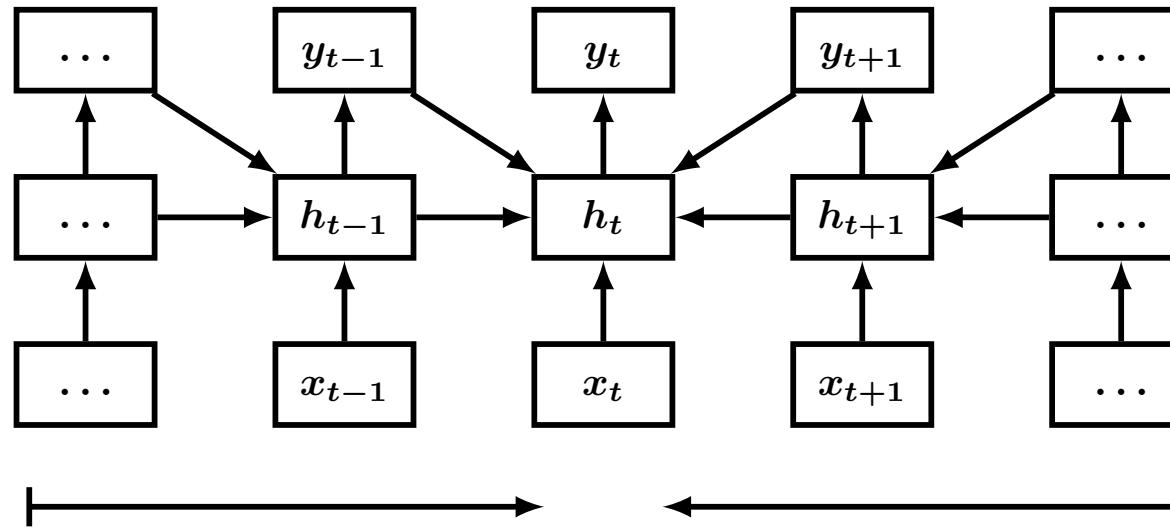
RNN output:

$$y_t(c) = p_t(c|c_0^{t-1}, x_1^T)$$

$$p(c_1^T|x_1^T) = \prod_t p_t(c|c_0^{t-1}, x_1^T)$$

note: most general case for factorization!

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



RNN output:

$$y_t(c) = p_t(c|c_0^{t-1}, c_{t+1}^T, x_1^T)$$

note: this output cannot be used for factorization!

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?

- typical case: posterior probability of an output string for a pair (x_1^T, c_1^T) :

$$p_\vartheta(c_1^T | x_1^T) = \prod_t p_\vartheta(c_t | c_0^{t-1}, x_1^T)$$

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

note: most general case of factorization

- cross-entropy criterion for a sequence of string pairs $[X_n; C_n]$, $n = 1, \dots, N$:

$$[X_n; C_n] = [x_{n1} \dots x_{nt} \dots x_{nT_n}; c_{n1} \dots c_{nt} \dots c_{nT_n}]$$

$$\sum_n \log p_\vartheta(C_n | X_n) = \sum_{n=1}^N \sum_{t=1}^{T_n} \log p_\vartheta(c_{nt} | c_{n0}^{n,t-1}, x_{n1}^{nT_n})$$

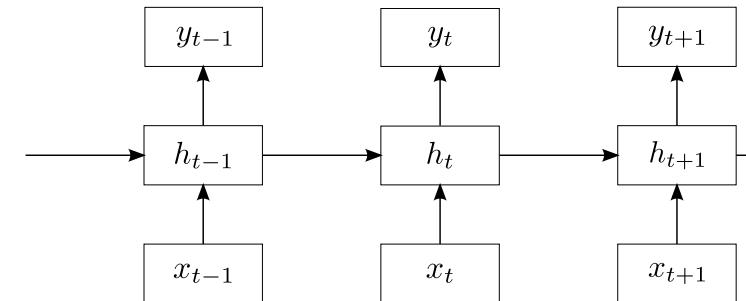
- conclusion for cross-entropy training:
'natural' transition from a single string to a sequence of strings

2.4.4 Backpropagation for Baseline RNN

backpropagation for RNN:

- principle similar to feedforward multi-layer perceptron
- special type of dependence: as a result of recurrence:
parameter sharing (tying) across several layers

terminology: backpropagation through time



consider the training criterion or error function $E = E(y_1^T)$:

$$E := \log p(c_1^T | x_1^T) = \log \prod_t p(c_t | c_0^{t-1}, x_1^t) = \sum_t \log p(c_t | c_0^{t-1}, x_1^t)$$

$$p_t(c | c_0^{t-1}, x_1^t) = y_t = \text{softmax}(Vh_t) \quad \text{with} \quad h_t = \tanh(Ux_t + Wh_{t-1})$$

partial derivative wrt an element w of the recurrence matrix W :

$$\frac{\partial E}{\partial w} = \frac{\partial}{\partial w} \sum_t E_t = \sum_t \frac{\partial E_t}{\partial w}$$

$$\frac{\partial E_t}{\partial w} = \sum_i \frac{\partial E_t}{\partial y_{ti}} \cdot \sum_j \frac{\partial y_{ti}}{\partial h_{tj}} \cdot \frac{\partial h_{tj}}{\partial w}$$

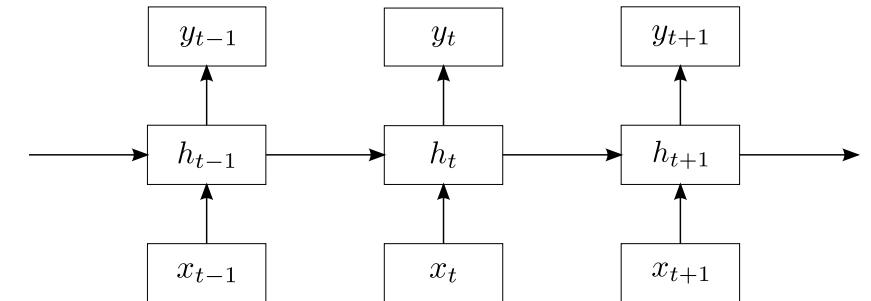
vector notation: $= \frac{\partial E_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial w}$

critical and tricky calculation: $\partial h_t / \partial w$

To calculate the derivative $\partial h_t / \partial w$, we consider and reformulate the recurrent dependence of h_t on w :

$$w \rightarrow h_t(w) = h_t(w, h_{t-1}(w, h_{t-2}(w, h_{t-3}(w, \dots))))$$

$$w \rightarrow H_t(w) = h_t(w, H_{t-1}(w))$$



compute derivative $\partial H_t / \partial w$:

$$\frac{\partial H_t}{\partial w} = \frac{\partial h_t}{\partial w} + \frac{\partial h_t}{\partial H_{t-1}} \cdot \frac{\partial H_{t-1}}{\partial w}$$

$$\frac{\partial H_{t-1}}{\partial w} = \frac{\partial h_{t-1}}{\partial w} + \frac{\partial h_{t-1}}{\partial H_{t-2}} \cdot \frac{\partial H_{t-2}}{\partial w}$$

$$\frac{\partial H_{t-2}}{\partial w} = \frac{\partial h_{t-2}}{\partial w} + \frac{\partial h_{t-2}}{\partial H_{t-3}} \cdot \frac{\partial H_{t-3}}{\partial w}$$

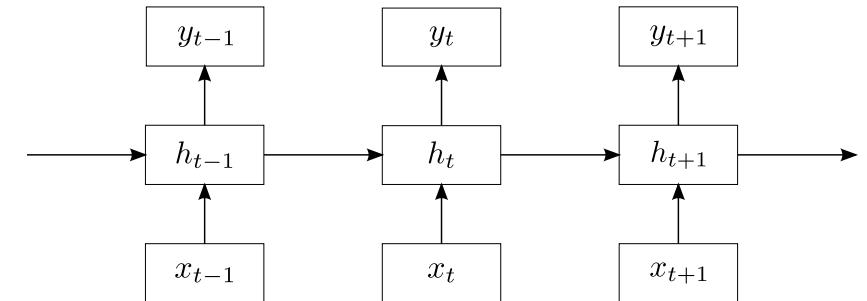
= ...

$$= \sum_{\tau=1}^t \left(\prod_{t'=\tau+1}^t \frac{\partial h_{t'}}{\partial H_{t'-1}} \right) \cdot \frac{\partial h_t}{\partial w}$$

note notation: matrix $\partial h_{t'}/\partial H_{t'-1}$ and vector $\partial h_t/\partial w$

consider dependencies:

$$\begin{aligned} w \rightarrow H_t(w) &= h_t(w, H_{t-1}(w)) \\ &= \tanh(z_t) \\ z_t &= Ux_t + W \cdot H_{t-1} \end{aligned}$$



with component-wise operation $\tanh(z_t)$

derivative of $\tanh(z)$ **for** $z \in \mathbb{R}$: $\frac{\partial}{\partial z} \tanh(z) = 1 - \tanh^2(z)$

compute derivative $\partial h_t / \partial H_{t-1}$:

$$\begin{aligned} \frac{\partial h_t}{\partial H_{t-1}} &= \frac{\partial h_t}{\partial z_t} \cdot \frac{\partial z_t}{\partial H_{t-1}} \\ &= \text{diag}[1 - \tanh^2(z_t)] \cdot W \end{aligned}$$

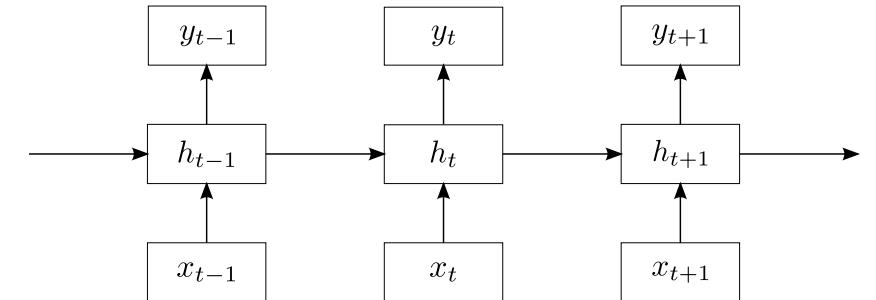
where the 'diagonalization' is caused by the component-wise operation $\tanh(z_t)$

for the entries u of matrix U : similar result

Summary: Backpropagation for RNN

dependencies:

$$\begin{aligned} w \rightarrow H_t(w) &= h_t(w, H_{t-1}(w)) \\ &= \tanh(z_t) \\ z_t &= Ux_t + W \cdot H_{t-1} \end{aligned}$$



overall result in both correct and sloppy notation:

$$\frac{\partial H_t}{\partial w} = \sum_{\tau=1}^t \left(\prod_{t'=\tau+1}^t \frac{\partial h_{t'}}{\partial H_{t'-1}} \right) \cdot \frac{\partial h_t}{\partial w} \quad (\text{correct})$$

$$\frac{\partial h_t}{\partial w} = \sum_{\tau=1}^t \left(\prod_{t'=\tau+1}^t \frac{\partial h_{t'}}{\partial h_{t'-1}} \right) \cdot \frac{\partial h_t}{\partial w} \quad (\text{sloppy})$$

$$\frac{\partial h_t}{\partial H_{t-1}} = \text{diag}[1 - \tanh^2(z_t)] \cdot W$$

2.5 Historical Review: ANN and Deep Learning

- 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:
 - [Patterson & Womack 66]
 - Bourlard and Wellekens 1989; Ney 1991
- experimental success and deep learning:
 - LeNet for digit image recognition: Le Cun 1989
 - RNN for ASR Robinson 1994
 - systematic experimental success: only in 2011 (handwriting, speech, image)
 - real improvements over competing methods
(like Gaussian, HMM, SVM, log-linear models, ...):
2008 Graves/handwriting; 2011 Hinton/TIMIT;
2011 Seide et al./hybrid LVCSR, Tuske et al./tandem LVCSR; 2012 computer vision

3 Bayes Decision Rule: Principle

we have learned the important result:

ANNs learn the class posterior estimate $pr(c|x)$ of the training data

this chapter consider fundamental open questions:

- for atomic outputs/decisions, i. e. unstructured output:
what is the best approach for minimum classification error?
- for structured outputs, i. e. for strings with and without synchronization:
 - what is the right approach?
 - how to measure classification error in case of strings?

terminology and notation:

- **input x : observation:**
 - single event: set of measurements, **observation vector**
 - sequence/string of observations: $x = x_1^T := x_1 \dots x_t \dots x_T$
- **output c : class symbol(s):**
 - single symbol: no internal structure, i.e. **atomic**
 - sequence/string of events: $c = c_1^N := c_1 \dots c_n \dots c_N$
 - speech recognition: words $w = w_1^N := w_1 \dots w_n \dots w_N$
- **functional dependence: notational variants:**

$$\begin{aligned}x &\rightarrow c(x) \\x &\rightarrow c_x \\x &\rightarrow \hat{c}_x \\x &\rightarrow c_x^* \\&\dots\end{aligned}$$

in addition: position index for strings, e.g. c_n or $c_n^*(x)$

starting points:

- ASR and other HLT tasks are very complex problems:
no perfect knowledge of the dependencies in speech and language:
 - different from conventional computer science
 - like a problem in natural sciences (e.g. approximative models in physics)
- perfect solution will be difficult:
 - we accept that the system will make errors
 - but we try to find the best compromise
- fairly general view: strings of random variables
 - input sequence (ASR: sequence over time t : $x := x_1 \dots x_t \dots x_T$, $x_t \in \mathbb{R}^D$)
 - output sequence: $c := c_1 \dots c_n \dots c_N$ of unknown length N
- we need a generation mechanism, i.e. a decision (rule) d :

$$d : x \rightarrow c = c_d(x)$$

- to this purpose, we assume a
 - posterior distribution $pr(c|x)$
 - which can be extremely complex: both arguments are strings!

assumption:

(huge) corpus of input-output (sequence) pairs:

$$(x_k, c_k), \quad k = 1, \dots, K$$

We consider an arbitrary decision rule d ,
i.e. an operational system for ASR or some other HLT task:

$$d : \quad x \rightarrow c = c_d(x)$$

In order to measure the performance of this rule (or system) d ,
we need a performance measure, error measure or loss function:

$$L[c, \tilde{c}]$$

between correct output string $c = c_k$ and the generated output string $\tilde{c} = c_d(x_k)$

We consider a set of decision rules d and the associated
average loss (or risk or performance) $L[d]$ on the whole corpus:

$$L[d] := \frac{1}{K} \cdot \sum_{k=1}^K L[c_k, c_d(x_k)]$$

distinguish two types of decisions for output:

- single symbol c without any internal structure: atomic decision
 - decision without context
 - examples: image object recognition, face recognition
- symbol string $c = c_1^N$ with some structure:
structured output (or compound decision):
 - decision with context
 - examples: text image recognition, automatic speech recognition (ASR), statistical machine translation (SMT)

note:

- combinations are possible: input=string and output=single index
- example: text classification: word string x and single class index c

examples of loss functions:

- ASR: Levenshtein or edit distance:
minimum number of operations: insertions, deletions, substitutions
- SMT: several variants of loss functions:
 - TER (translation error/edit rate):
edit distance + (block) swaps
 - BLEU (bilingual evaluation understudy):
word n -gram accuracy along with brevity penalty
 - count error:
compare counts of each word unigram (or word n -gram)

approach:

- goal: find the best decision rule
- problem: the same input $x = x_k$ might occur several times in the corpus and have DIFFERENT outputs $c = c_k$
- decision rule: must be a function,
i.e. it must always generate the same output.
- approach: define the empirical distribution using the Kronecker delta $\delta(\cdot, \cdot)$:

$$pr(x, c) := \frac{1}{K} \cdot \sum_{x, c} \sum_{k: (x_k, c_k) = (x, c)} 1 = \frac{1}{K} \cdot \sum_{k=1}^K \delta(c, c_k) \cdot \delta(x, x_k)$$

i.e. relative frequencies of pairs (x, c)

- implementation (maybe inefficient): big table or lists
(later: probabilistic models will approximate the empirical distribution)

extension:

from discrete-valued observations to continuous-valued observations x

labelled corpus	
input string (observation)	output string (class)
x_1	c_1
x_2	c_2
x_3	c_3
...	...
...	...
...	...
x_{k-2}	c_{k-2}
x_{k-1}	c_{k-1}
x_k	c_k
x_{k+1}	c_{k+1}
x_{k+2}	c_{k+2}
...	...
...	...
...	...
x_{K-2}	c_{K-2}
x_{K-1}	c_{K-1}
x_K	c_K

labelled corpus is a relation:

$$(x_k, c_k), \ k = 1, \dots, K$$

decision rule d is a function:

$$x \rightarrow c = c_d(x)$$

total loss of rule d on this corpus:

$$\begin{aligned} L[d] &= \frac{1}{K} \cdot \sum_{k=1}^K L[c_k, c_d(x_k)] \\ &= \sum_{x,c} pr(x, c) L[c, c_d(x)] \\ &= \sum_x pr(x) \sum_c pr(c|x) L[c, c_d(x)] \end{aligned}$$

using the empirical distributions $pr(x, c)$ etc.

labelled corpus	
input string (observation)	output string (class)
x_1	c_1
x_2	c_2
x_3	c_3
...	...
...	...
...	...
x_{k-2}	c_{k-2}
x_{k-1}	c_{k-1}
x_k	c_k
x_{k+1}	c_{k+1}
x_{k+2}	c_{k+2}
...	...
...	...
...	...
x_{K-2}	c_{K-2}
x_{K-1}	c_{K-1}
x_K	c_K

definition of empirical distribution:

$$pr(x, c) = \frac{1}{K} \cdot \sum_{k=1}^K \delta(c, c_k) \cdot \delta(x, x_k)$$

equivalence of expectation and empirical average:

$$\begin{aligned} L[d] &= \sum_{x,c} pr(x, c) L[c, c_d(x)] \\ &= \sum_{x,c} \left(\frac{1}{K} \cdot \sum_{k=1}^K \delta(c, c_k) \cdot \delta(x, x_k) \right) L[c, c_d(x)] \\ &= \frac{1}{K} \cdot \sum_{k=1}^K \sum_{x,c} \delta(c, c_k) \cdot \delta(x, x_k) L[c, c_d(x)] \\ &= \frac{1}{K} \cdot \sum_{k=1}^K L[c_k, c_d(x_k)] \end{aligned}$$

- **joint distribution:**

$$pr(x, c) = \frac{1}{K} \cdot \sum_{k=1}^K \delta(c, c_k) \cdot \delta(x, x_k)$$

- **marginal distribution over observations x :**

$$pr(x) = \sum_c pr(x, c) = \frac{1}{K} \cdot \sum_{k=1}^K \delta(x, x_k)$$

- **marginal distribution over classes c :**

$$pr(c) = \sum_x pr(x, c) = \frac{1}{K} \cdot \sum_{k=1}^K \delta(c, c_k)$$

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 $pr(c) > 0$:**

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

3.3 Bayes Decision Rule

- we define the total loss (or risk) $L[d]$:

$$\begin{aligned} L[d] &= \frac{1}{K} \cdot \sum_{k=1}^K L[c_k, c_d(x_k)] = \sum_{x,c} pr(x, c) L[c, c_d(x)] \\ &= \sum_x pr(x) \sum_c pr(c|x) L[c, c_d(x)] \end{aligned}$$

- we minimize over all decision rules d :

$$\begin{aligned} \min_d L[d] &= \sum_x pr(x) \min_d \left\{ \sum_c pr(c|x) L[c, c_d(x)] \right\} \\ &= \sum_x pr(x) \min_{\tilde{c}_x} \left\{ \sum_c pr(c|x) L[c, \tilde{c}_x] \right\} = \sum_x pr(x) \min_{\tilde{c}} \left\{ \sum_c pr(c|x) L[c, \tilde{c}] \right\} \end{aligned}$$

last re-writing: minimization over strings \tilde{c} in lieu of rules d

- we have arrived at the Bayes decision rule:

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

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

- assumption so far: empirical distribution,
 - it was non-zero only for the observed pairs (x_k, c_k) in the corpus
 - for unseen pairs (x, c) : $pr(x, c) = 0$
 - for unseen observation x : $pr(x) = 0$
- assumption for future experiments:
approach should work for all inputs x

concept of generalization:

- we want to define a model distribution $p(c|x)$
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 $p(c|x)$

most simple concept of performance: count the classification errors by using the 0/1 loss function:

$$L[\tilde{c}, c] = 1 - \delta(\tilde{c}, c)$$

distinguish two types of outputs:

- single class symbol: symbol error rate
- strings of class symbols: string error rate

Bayes decision rule for 0/1 loss function:

$$\begin{aligned} x \rightarrow c_*(x) &= \arg \min_c \left\{ \sum_{\tilde{c}} pr(\tilde{c}|x) L[\tilde{c}, c] \right\} \\ &= \arg \min_c \left\{ 1 - \sum_{\tilde{c}} pr(\tilde{c}|x) \delta(\tilde{c}, c) \right\} = \arg \min_c \left\{ 1 - pr(c|x) \right\} \\ &= \arg \max_c \left\{ pr(c|x) \right\} \end{aligned}$$

remarks:

- result: MAP decision rule (MAP = maximum a-posteriori)
- threshold effect: $pr(c|x) \stackrel{?}{\geq} 0.5$

Terminology: MAP Rule vs. Bayes Rule

typical situation in HLT tasks with output strings c :

- cost function in Bayes decision rule:
 - 0/1 loss function at string level
 - MAP rule is used in generation (search/decoding) process
- practical performance measure:
 - errors at symbol level are counted
 - e.g. WER for ASR and TER for SMT
- result: inconsistency between loss function in decision rule and practical performance measure

terminology:

$$\text{MAP string or rule: } x \rightarrow c_0(x) = \arg \max_c \{pr(c|x)\}$$

$$\text{Bayes string or rule: } x \rightarrow c_*(x) = \arg \min_c \left\{ \sum_{\tilde{c}} pr(\tilde{c}|x) L[\tilde{c}, c] \right\}$$



remarks about terminology:

- **correct Bayes decision rule:** what does '**correct**' mean?
 - correct loss function, e. g. edit distance as opposed to 0/1 loss function
 - true distribution as opposed to model distribution
- literature: ***minimum* Bayes decision rule**
 - misleading terminology: Bayes decision rule is always based on the ***minimum***
 - what is meant: correct loss function is used

theoretic analysis: optimality of Bayes decision rule

open questions:

- **what is the relevance of cost function for strings?**
related question: does the details of the cost function matter?
- **modelling problem: how to define a model distribution?**
- **training problem: how to learn a model distribution?**

4 Bayes Decision Rule: Loss Function

general form of Bayes decision rule:

$$x \rightarrow c_*(x) = \arg \min_c \left\{ \sum_{\tilde{c}} p(\tilde{c}|x) L[\tilde{c}, c] \right\}$$

with $p(c|x)$ being either the true distribution $pr(c|x)$ or a normalized model $p_\theta(c|x)$
(note: purely mathematical statements about equivalence)

goal: assumptions about the structure of the outputs and the loss function
so that we get a more explicit form of the Bayes decision rule

three types of outputs and associated loss functions:

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

classification error for atomic output:

- the output is considered as a whole
- classification error count

$$L[\tilde{c}, c] = 1 - \delta(\tilde{c}, c)$$

Bayes decision rule for 0/1 loss function:

$$\begin{aligned} x \rightarrow c_*(x) &= \arg \min_c \left\{ \sum_{\tilde{c}} p(\tilde{c}|x) L[\tilde{c}, c] \right\} \\ &= \arg \min_c \{1 - p(c|x)\} \\ &= \arg \max_c \{p(c|x)\} \end{aligned}$$

often referred to as MAP rule:

select the class with the largest maximum-a-posteriori (MAP) probability

weighted classification errors:

- **typical task: output = single class symbol**
- **errors are assigned weights**
- **assumption: weights α_c depend on hypothesized class c :**

$$L[\tilde{c}, c] = [1 - \delta(\tilde{c}, c)] \cdot \alpha_c$$

Bayes decision rule:

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

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

$$L[\tilde{c}, c] = [1 - \delta(\tilde{c}, c)] \cdot \beta_{\tilde{c}}$$

Bayes decision rule:

$$\begin{aligned} x \rightarrow c_*(x) &= \arg \min_c \left\{ \sum_{\tilde{c}} p(\tilde{c}|x) L[\tilde{c}, c] \right\} \\ &= \arg \min_c \left\{ \sum_{\tilde{c}} p(\tilde{c}|x) \left([1 - \delta(c, \tilde{c})] \cdot \beta_{\tilde{c}} \right) \right\} \\ &= \arg \min_c \left\{ \sum_{\tilde{c}} p(\tilde{c}|x) \cdot \beta_{\tilde{c}} - p(c|x) \cdot \beta_c \right\} \\ &= \arg \max_c \left\{ \beta_c \cdot p(c|x) \right\} \end{aligned}$$

- remark about the weights in both cases:
as a result, the loss function $L[\tilde{c}, c]$ is not symmetric anymore!

4.2 Structured Output: String 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 phonetic (CART) label for each 10-ms acoustic frame
- **part-of-speech (POS) tagging and semantic tagging (NLU):**
assign a syntactic/semantic word category to each (written) word
- **spelling correction (with synchronization!)**

general framework:

decision making in 'context'

input data:	x_1	x_2	\dots	x_{n-1}	x_n	x_{n+1}	\dots	x_{N-1}	x_N
output symbols:	c_1	c_2	\dots	c_{n-1}	c_n	c_{n+1}	\dots	c_{N-1}	c_N

approaches:

- local normalization:

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

most successful approach: RNN with bidirectional input

- global normalization (linear chain CRF):

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

$$Z(x_1^N) := \sum_{c_1^N} \prod_n Q_n(c_n; c_0^{n-1}, x_1^N)$$

with an *arbitrary non-negative model* $Q_n(c_n; c_0^{n-1}, x_1^N)$

most successful approach (?): RNN-CRF, i. e. RNN with GLOBAL re-normalization

loss function for two strings: c_1^N and \tilde{c}_1^N with positions $n = 1, \dots, N$:

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

**in other words: we count the classification errors in each position.
which is referred to as Hamming distance or symbol error in a string.**

correct symbols: $\tilde{c}_1 \quad \tilde{c}_2 \quad \dots \quad \tilde{c}_{n-1} \quad \tilde{c}_n \quad \tilde{c}_{n+1} \quad \dots \quad \tilde{c}_{N-1} \quad \tilde{c}_N$

○	○	○	○	○	○	○	○	○
○	○	○	○	○	○	○	○	○

hypothesized symbols: $c_1 \quad c_2 \quad \dots \quad c_{n-1} \quad c_n \quad c_{n+1} \quad \dots \quad c_{N-1} \quad c_N$

important assumption:

the positions $n = 1, \dots, N$ (including the length N) are well defined by the task.

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

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

with the symbol posterior probability $p_n(c_n|x)$ in position n :

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

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

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

$$\begin{aligned} L[c_1^N | x] &= \sum_n [1 - p_n(c_n | x)] \\ \min_{c_1^N} \{L[c_1^N | x]\} &= \min_{c_1^N} \left\{ \sum_n [1 - p_n(c_n | x)] \right\} \\ &= \sum_n \min_{c_n} [1 - p_n(c_n | x)] \\ &= \sum_n [1 - \max_{c_n} p_n(c_n | x)] \end{aligned}$$

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

$$x \rightarrow \hat{c}_n(x) = \arg \max_{c_n} \left\{ p_{r_n}(c_n | x) \right\}$$

interpretation: looks like decision rule for single symbol in position n

compare the Bayes decision rules for positions $n = 1, \dots, N$:

minimum symbol error:

$$\begin{aligned} x \rightarrow \hat{c}_n(x) &= \arg \max_{c_n} \{p_n(c_n|x)\} \\ &= \arg \max_{c_n} \left\{ \sum_{\tilde{c}_1^N : \tilde{c}_n = c_n} p_n(\tilde{c}_1^N|x) \right\} \end{aligned}$$

minimum string error:

$$\begin{aligned} x \rightarrow \hat{c}_1^N(x) &= \arg \max_{c_1^N} \{p(c_1^N|x)\} \\ x \rightarrow \hat{c}_n(x) &= \arg \max_{c_n} \left\{ \max_{\tilde{c}_1^N : \tilde{c}_n = c_n} p_n(\tilde{c}_1^N|x) \right\} \end{aligned}$$

remarks about the two rules:

- **important conclusion: 50% rule for MAP string:**
if $\max_{c_1^N} p(c_1^N|x) > 0.5$, 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.

- previous work:
 - Merialdo 1993
 - Popovic & Ney 2003
- experimental results:
 - symbol-based rule: experiments not conclusive
 - difficult to draw conclusions due to many assumptions and short-cuts

4.3 Structured Output: String with no Synchronization

**assumption: symbol string with no synchronization,
no fixed positions in symbol string**

specific loss functions:

- **edit distance (ASR)**
- **edit distance with movements of symbol groups (SMT)**

restricted class of loss functions:

property: metric or semi-metric

we consider the edit distance in ASR:

- **most work: experimental studies**
- **not much theoretical work**
- **work by RWTH team:**
[Schlüter & Scharrenbach⁺ 05, Schlüter & Nussbaum⁺ 11, Schlüter & Nussbaum⁺ 12]

4.3.1 Pathological Example: Bayes String with Zero Probability

about pathological examples:

- **purpose: to illustrate the difference between strict mathematics (extreme) models and realistic models**
- **purpose: to understand what properties of models are important for reality**
- **specific example: strings with zero probability are unlikely in practice due to locality effect of models**

Pathological Example 1

Example 1:

- **vocabulary:** $\{a, b, d, e, f, \dots, u, v, w, \dots\}$
- **start with abd and generate three more strings by substitutions**
- **loss function:** edit distance

name	string c	$p(c x)$
c_0	abd	0
c_1	abu	q_1
c_2	aud	q_2
c_3	ubd	q_3
\tilde{c}	...	0

posterior loss of a string c : $L[c|x] = q_1 L[c, c_1] + q_2 L[c, c_2] + q_3 L[c, c_3]$

specific strings: $L[c_0|x] = q_1 + q_2 + q_3 = 1$

$$L[c_1|x] = 2q_2 + 2q_3 = 2(1 - q_1)$$

$$L[c_2|x] = 2q_1 + 2q_3 = 2(1 - q_2)$$

$$L[c_3|x] = 2q_1 + 2q_2 = 2(1 - q_3)$$

$$L[\tilde{c}|x] > 1 \quad \text{for } \tilde{c} \neq c_i, i = 0, 1, 2, 3 \quad (\text{verify})$$

result: Bayes string: $c_0 = abd$ if $q_i < 0.5$ for each i

Example 2:

- vocabulary: $\{a, b, u\}$
- start with ab and generate three more strings by insertions
- loss function: edit distance

name	string c	$p(c x)$
c_0	ab	0
c_1	uab	q_1
c_2	abu	q_2
c_3	aub	q_3
\tilde{c}	...	0

properties:

- apart from c_1, c_2, c_3 , all strings have zero probability: $q_1 + q_2 + q_3 = 1$
- strings c_1, c_2, c_3 : all pairs have the cost: $L[c_i, c_j] = 2$.
- string c_0 : has the cost $L[c_0, c_j] = 1$ for each $c_j = c_1, c_2, c_3$

posterior loss of any string c :

$$\begin{aligned} L[c|x] &= q_1 L[c, c_1] + q_2 L[c, c_2] + q_3 L[c, c_3] \\ &\geq (q_1 + q_2 + q_3) \cdot \min_i L[c, c_i] \\ &= \min_{i=1,2,3} L[c, c_i] \end{aligned}$$

analysis of procedure for finding the Bayes string:

- the above posterior loss consists of two contributions:
the probabilities q_i , $i = 1, 2, 3$ and the edit distances $L[c, c_i]$, $i = 1, 2, 3$
- probabilities q_i :
can be chosen arbitrarily (with normalization constraint!)
- edit distances $L[c, c_i]$:
 - only possible values: non-negative integer
 - for minimizing the posterior loss, only strings c with 'small' edit distances (e.g. 0, 1, 2, ...) are relevant.

posterior loss of a any string c :

$$L[c|x] = q_1 L[c, c_1] + q_2 L[c, c_2] + q_3 L[c, c_3]$$

case distinction: three cases for c :

- **posterior loss for string $c = c_0$:**

$$\begin{aligned} L[c_0|x] &= q_1 L[c_0, c_1] + q_2 L[c_0, c_2] + q_3 L[c_0, c_3] \\ &= q_1 + q_2 + q_3 \\ &= 1 \end{aligned}$$

- **posterior loss for a string $c = c_i, i = 1, 2, 3$:**

$$L[c_i|x] = 2 \cdot (1 - q_i) \quad i = 1, 2, 3$$

proof: as an example onsider $L[c_2|x]$:

$$\begin{aligned} L[c_2|x] &= q_1 L[c_2, c_1] + q_2 L[c_2, c_2] + q_3 L[c_2, c_3] \\ &= q_1 \cdot 2 + q_2 \cdot 0 + q_3 \cdot 2 \\ &= 2 \cdot (q_1 + q_3) \\ &= 2 \cdot (1 - q_2) \end{aligned}$$

- posterior loss for any other string $c = \tilde{c} \neq c_i, i = 0, 1, 2, 3$:

$$\begin{aligned} L[\tilde{c}|x] &= q_1 L[\tilde{c}, c_1] + q_2 L[\tilde{c}, c_2] + q_3 L[\tilde{c}, c_3] \\ &\quad (\text{at least one of the losses } L[\tilde{c}, c_i] \text{ must be 2, say } c_j) \\ &\geq (1 - q_j) \cdot 1 + q_j \cdot 2 \\ &= 1 + q_j \\ &> 1 \end{aligned}$$

remarks:

- critical change in decision: $q_i = 0.5$
- Bayes string: c_0 for $\max_i q_i < 0.5$
 - why? all other strings have edit distance of 1 (or more) to each c_i
 - but: c_0 has zero probability!
- length: not preserved

4.3.2 Basic Inequality for Metric Loss Function

**consider difference in expected loss
for two classes \hat{c} and \tilde{c} with an observation x :**

$$\begin{aligned} L[\hat{c}|x] - L[\tilde{c}|x] &= \sum_c p(c|x) (L[c, \hat{c}] - L[c, \tilde{c}]) \\ &= p(\hat{c}|x) (L[\hat{c}, \hat{c}] - L[\hat{c}, \tilde{c}]) + \sum_{c \neq \hat{c}} p(c|x) (L[\hat{c}, c] - L[\tilde{c}, c]) \end{aligned}$$

identity/reflexivity: $L[\hat{c}, \hat{c}] = 0$

triangle inequality: $L[\hat{c}, c] - L[\tilde{c}, c] \leq L[\hat{c}, \tilde{c}]$

$$\begin{aligned} &\leq -p(\hat{c}|x) L[\hat{c}, \tilde{c}] + [1 - p(\hat{c}|x)] L[\hat{c}, \tilde{c}] \\ &= [1 - 2p(\hat{c}|x)] L[\hat{c}, \tilde{c}] \end{aligned}$$

$$L[\hat{c}|x] \leq L[\tilde{c}|x] + [1 - 2p(\hat{c}|x)] L[\hat{c}, \tilde{c}]$$

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

$$L[\hat{c}|x] \leq L[\tilde{c}|x] \text{ for all } \tilde{c}$$

- conclusion: consider the MAP rule:

$$x \rightarrow \hat{c}_x = \operatorname{argmax}_c p(c|x)$$

if $p(\hat{c}_x|x) > 0.5$, then the MAP rule is equivalent to the Bayes decision rule with any metric loss function

notation: x and y now denote symbol strings: $d(x, y)$ in lieu of $L[c, \tilde{c}]$

A function

$$(x, y) \rightarrow d(x, y) \geq 0$$

is a metric if the following properties hold:

- **reflexivity/identity:** $d(x, y) = 0 \iff x = y$
- **symmetry:** $d(x, y) = d(y, x)$
- **triangle inequality:** $d(x, y) \leq d(x, z) + d(z, y)$

examples of a metric: Euclidean distance, edit distance

For a semi-metric, the reflexivity property is relaxed:

$$d(x, y) = 0 \text{ for } x \neq y$$

example of a semi-metric: count-based error (PER) in SMT
(PER = Position independent Error Rate)

exercise: work out the proofs!

more details in literature: [Schlüter & Scharrenbach⁺ 05]

experiments on 5k-WSJ recognitions (740 sentences = 12137 words)
 with MAP rule $c_0(x)$ and Bayes decision rule $c_*(x)$:

case distinction	sentences [%]	WER [%]
A: theory: 0.5-MAP condition	54	1.1
B: theory: extended MAP condition	8	3.6
C: experiment: $c_0(x) = c_*(x)$	31	6.6
D: experiment: $c_0(x) \neq c_*(x)$	7	11.8 → 10.6
all cases	100	4.0 → 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: MAP and Bayes rules differ only for high error rates!

more results [Schlüter & Nussbaum⁺ 11] on:

- EPPS: European Parliament Plenary Sessions (EU project)
- GALE: broadcast news & conversations (US project)

ASR Task	EPPS Spanish		GALE Arabic	
vocabulary size	52k		255k	
size of N -best lists	10k		10k	
case distinction	sentences	WER [%]	sentences	WER [%]
A: theory: 0.5-MAP condition	29	4.0	8	15.7
B: theory: extended MAP condition	6	5.0	3	13.6
C: experiment: $c_0(x) = c_*(x)$	45	9.0	58	34.2
D: experiment: $c_0(x) \neq c_*(x)$	21	14.3 → 13.8	32	43.5 → 42.6
all cases	100	9.6 → 9.5	101(?)	36.6 → 36.3

conclusion: disappointing improvements ...

4.3.3 Rewrite Posterior Loss

- consider a metric loss function: $L[\tilde{c}, c] = 0, 1, 2, \dots$
typical case for edit distance and other loss functions for strings
- candidate strings \hat{c} for the minimum posterior loss
have a maximum length I_x depending on the input string x .
example: edit distance at letter/phoneme level: $I_x = \text{duration}/40 \text{ ms}$
- goal of rewriting:
to illustrate why the MAP rule nearly always provides
a good approximation to the correct Bayes decision rule.

given: input string x with maximum length I_x for output string

we re-write the posterior loss for any input-output pair (x, \hat{c}) :

$$\begin{aligned}
 L[\hat{c}|x] &= \sum_c p(c|x) L[c, \hat{c}] \\
 &= \sum_{i=1}^{I_x} i \cdot \sum_{c:L[c,\hat{c}]=i} p(c|x) \\
 &= 1 \cdot \sum_{c:L[c,\hat{c}]=1} p(c|x) + \sum_{i=2}^{I_x} i \cdot \sum_{c:L[c,\hat{c}]=i} p(c|x) \\
 &= \sum_{c:L[c,\hat{c}]\geq 1} p(c|x) + \sum_{i=2}^{I_x} (i-1) \cdot \sum_{c:L[c,\hat{c}]=i} p(c|x) \\
 &= 1 - p(\hat{c}|x) + \sum_{i=2}^{I_x} (i-1) \cdot \sum_{c:L[c,\hat{c}]=i} p(c|x)
 \end{aligned}$$

We have re-written the posterior loss:

$$L[\hat{c}|x] = 1 - p(\hat{c}|x) + \sum_{i=2}^{I_x} (i-1) \cdot \sum_{c:L[c,\hat{c}]=i} p(c|x)$$

and can re-formulate the Bayes decision rule:

$$\begin{aligned} x \rightarrow c_*(x) &= \operatorname{argmin}_{\hat{c}} \{ L[\hat{c}|x] \} \\ &= \operatorname{argmax}_{\hat{c}} \left\{ p(\hat{c}|x) - \sum_{i=2}^{I_x} (i-1) \cdot \sum_{c:L[c,\hat{c}]=i} p(c|x) \right\} \end{aligned}$$

observations:

- sum over c : no direct contributions from strings c with loss=1:
if strings with loss=1 have *sufficient* probability mass,
the second term can be ignored for the maximum approximation:

Bayes decision rule = MAP rule

- number of strings with loss = 1 with alphabet size C and length N of string \hat{c}
that can *absorb* much probability mass:
 - Hamming distance (= 1:1): $(C - 1) \cdot N$
 - edit distance: $C \cdot (2N + 1)$ (see next slide)

consider the number of possible strings with loss=1 to the candidate string \hat{c} (with size C of class symbol alphabet):

- by substitution errors: $(C - 1)$ in each of N positions
- by deletion errors: 1 in each of N positions
- by insertion errors: C in each of $(N + 1)$ positions

in total: $C \cdot (2N + 1)$ possible strings

that are likely to absorb most of the remaining probability mass $[1 - p(\hat{c}|x)]$
(justification: locality effect)

candidate string \hat{c}:	\hat{c}_1	\hat{c}_2	...	\hat{c}_{n-1}	\hat{c}_n	\hat{c}_{n+1}	...	\hat{c}_{N-1}	\hat{c}_N	
one substitution:	\hat{c}_1	\hat{c}_2	...	\hat{c}_{n-1}	c_n	\hat{c}_{n+1}	...	\hat{c}_{N-1}	\hat{c}_N	
one deletion:	\hat{c}_1	\hat{c}_2	...	\hat{c}_{n-1}	.	\hat{c}_{n+1}	...	\hat{c}_{N-1}	\hat{c}_N	
one insertion:	\hat{c}_1	\hat{c}_2	...	\hat{c}_{n-1}	\hat{c}_n	c_n	\hat{c}_{n+1}	...	\hat{c}_{N-1}	\hat{c}_N

4.3.4 Edit Distance and Symbol Posterior Probability

[Schlüter & Nussbaum⁺ 11], [Xu, Povey et al. 2010]

compare the two cases for structured output:

- strings with synchronization:
 explicit solution by using the symbol posterior probability
- strings without synchronization:
 - position axis: is not well defined
 - remedy in case of edit distance:
 use a seed string to define an auxiliary position axis

approach:

- starting point: the local loss for observed string x :

$$L[\hat{c}|x] = \sum_c p(c|x) \cdot \min_A \{ L_A[\hat{c}, c] \}$$

- procedure:
 - explicit use of alignments for representing deletions/insertions
 - introduction of ϵ (=empty) symbols for representing deletions/insertions
- definition of normalized position axis
and iterative procedure with local convergence

Example: From Edit Distance to Hamming Distance [Schlüter & Nussbaum⁺ 11]

edit distances $L[\hat{c}, c]$

string c	$p(c x)$	symbols	$L[\hat{c}, c]$
\hat{c}	q_0	abu	0
c_1	q_1	aub	2
c_2	q_2	$adub$	2
c_3	q_3	adb	2

Hamming distances $H[\hat{c}^\epsilon, c^\epsilon]$

string c^ϵ	$p(c x)$	symbols	$H[\hat{c}^\epsilon, c^\epsilon]$
\hat{c}^ϵ	q_0	$a\epsilon bu\epsilon$	0
c_1^ϵ	q_1	$aub\epsilon\epsilon$	2
c_2^ϵ	q_2	$a\epsilon dub$	2
c_3^ϵ	q_3	$adb\epsilon\epsilon$	2

seed string:

initial: $\hat{c} = abu$

after ϵ transformations: $\hat{c}^\epsilon = a\epsilon bu\epsilon$

(potentially) improved string with symbols \hat{c}_n^ϵ in positions $n = 1, \dots, N_\epsilon$:

$$\hat{c}_n^\epsilon := \underset{c_n^\epsilon}{\operatorname{argmax}} \{ p_n(c_n^\epsilon | \hat{c}, x) \}$$

result for $q_i > 0.5$, $i = 1, 2, 3$: $\hat{c}^\epsilon = c_i^\epsilon$

result for ...??...: $\hat{c}^\epsilon \stackrel{?}{=} a\epsilon b\epsilon\epsilon$

exercise: verify/correct/improve example

Edit Distance and Hamming Distance

iterative procedure with local convergence:

- select a seed string \hat{c} , e. g. $\hat{c} = \hat{c}_x = \operatorname{argmax}_c p(c|x)$
- compute pairwise alignments $\hat{A}(\hat{c}, c) := \operatorname{argmin}_A \{L_A[\hat{c}, c]\}$ for all strings c
- define normalized position axis $n = 1, \dots, N_\epsilon$:
 - use ϵ symbols to represent deletions/insertions in alignments and extend all strings c (incl. \hat{c}) by ϵ : $c = [c_1 \dots c_n \dots c_N] \rightarrow c^\epsilon = [c_1^\epsilon \dots c_n^\epsilon \dots c_{N_\epsilon}^\epsilon]$
 - result: equality between edit distance and Hamming distance:

$$\min_A \{L_A[\hat{c}, c]\} = \sum_{n=1}^{N_\epsilon} [1 - \delta(\hat{c}_n^\epsilon, c_n^\epsilon)] \quad \text{for all } c$$

$$\min_A \{L_A[\tilde{c}, c]\} \leq \sum_{n=1}^{N_\epsilon} [1 - \delta(\tilde{c}_n^\epsilon, c_n^\epsilon)] \quad \text{for all pairs } (\tilde{c}, c)$$

- define symbol posterior probabilities for each symbol c_n^ϵ in position n :

$$p_n(c_n^\epsilon | \hat{c}, x) := \sum_{\tilde{c}: \tilde{c}_n^\epsilon = c_n^\epsilon} p(\tilde{c}|x)$$

- compute improved string candidate: $\hat{c}_n^\epsilon := \operatorname{argmax}_{c_n^\epsilon} \{p_n(c_n^\epsilon | \hat{c}, x)\}$
remove ϵ symbols and use this string as a new seed string

start: a seed string \hat{c} , e. g. $\hat{c} = \hat{c}_x = \operatorname{argmax}_c p(c|x)$

$$\begin{aligned}
 L[\hat{c}|x] &= \sum_c p(c|x) \cdot \min_A \{L_A[\hat{c}, c]\} = \sum_c p(c|x) \cdot \sum_{n=1}^{N_\epsilon} [1 - \delta(\hat{c}_n^\epsilon, c_n^\epsilon)] \\
 &= \sum_{n=1}^{N_\epsilon} \sum_c p(c|x) \cdot [1 - \delta(\hat{c}_n^\epsilon, c_n^\epsilon)] \\
 &= \sum_{n=1}^{N_\epsilon} [1 - \sum_{c:c_n^\epsilon=\hat{c}_n^\epsilon} p(c|x)] \\
 &= \sum_{n=1}^{N_\epsilon} [1 - p_n(\hat{c}_n^\epsilon|\hat{c}, x)] \\
 &\geq \sum_{n=1}^{N_\epsilon} [1 - \max_{c_n^\epsilon} p_n(c_n^\epsilon|\hat{c}, x)]
 \end{aligned}$$

define: $\hat{c}_n^\epsilon := \operatorname{argmax}_{c_n^\epsilon} \{p_n(c_n^\epsilon|\hat{c}, x)\}$

= ... (equations in backward direction; details to be worked out)

$\geq L[\hat{c}|x]$

Edit Distance and Hamming Distance: Remarks

- how to find an improvement over the seed string?

$$\sum_{n=1}^{N_\epsilon} [1 - p_n(\hat{c}_n^\epsilon | \hat{c}, x)] \geq \sum_{n=1}^{N_\epsilon} [1 - \max_{c_n^\epsilon} p_n(c_n^\epsilon | \hat{c}, x)]$$

present derivation is based on:

- introducing a normalized position axis
- improvements by symbol substitutions only

open question: what about (explicit) symbol insertions or deletions?

- how to arrive at an efficient implementation?

(so far, we have considered principles)

- conversion to ϵ strings
- symbol posterior probabilities: word lattice, confusion network etc.

application to ASR:

- representation of word strings:
N-best list, word graph/lattice, confusion network, ...
- related work:
 - L. Mangu, E. Brill, A. Stolcke [Eurospeech 1999]
 - public toolkit: SRI language model (A. Stolcke)
- general experimental experience for these and similar approaches:
 - clear improvements only for high error rates: e.g. from 41% to 40%
 - improvements tend to be small
 - consistent with the 50% MAP rule and its extensions
 - conjecture: maybe we are missing a theoretical analysis of MAP rule below 50%

4.3.5 Summary

task: strings with no synchronization

MAP rule vs. exact Bayes decision rule with a metric loss function:

- **condition** $\max_c p(c|x) > 0.5$: **exact equivalence**
- **in other cases: MAP rule is an excellent approximation**
- **improvements beyond MAP rule:**
 - yes, they are possible
 - example for edit distance: position-based symbol posterior probabilities
 - but they tend to be marginal

non-metric loss function (e. g. using weighted error counts):

- different situation that requires another analysis

4.3.6 Excursion for ASR (draft): Frame Error vs. Edit Distance

we re-consider ASR and the output string:

- input: sequence of acoustic observations x_1^T (continuous-valued vectors)
- output: sequence of time-synchronous frame labels y_1^T
 - examples: CART, allophonic labels, phonemes etc
 - advantage: more uniform weighting of input x_1^T

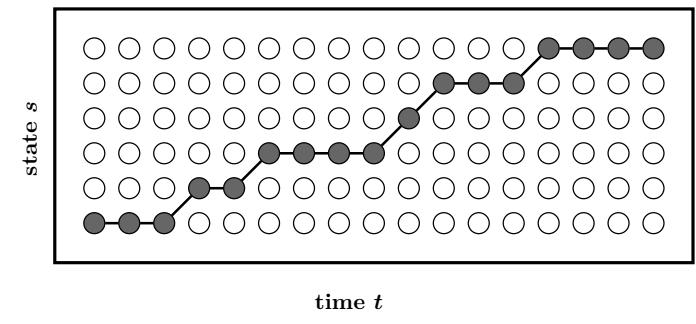
example (note: the same label alphabet for frames and segments):

$$\begin{aligned}\text{string labels: } y_1^T &= \alpha\alpha\alpha\beta\beta\gamma\gamma\gamma\gamma\alpha\delta\delta\delta\beta\beta\beta\beta \\ &= \alpha^3\beta^2\gamma^4\alpha^1\delta^3\beta^4\end{aligned}$$

$$\text{segment labels: } a_1^S = \alpha\beta\gamma\alpha\delta\beta$$

possible loss functions:

- string labels: frame error rate
- segment labels: edit distance
(maybe suboptimal decision rule based on position probabilities)



starting point: correct frame labels are known

- Bayes decision rule for minimum label error at time t :

$$L[y_1^T | x_1^T] := \sum_{\tilde{y}_1^T} p(\tilde{y}_1^T | x_1^T) \cdot \sum_t [1 - \delta(y_t, \tilde{y}_t)] = \dots = \sum_t [1 - p_t(y_t | x_1^T)]$$

$$\min_{y_1^T} L[y_1^T | x_1^T] = \sum_t [1 - \max_{y_t} p_t(y_t | x_1^T)] \quad \text{with} \quad p_t(y_t | x_1^T) := \sum_{\tilde{y}_1^T: y_t = \tilde{y}_t} p(\tilde{y}_1^T | x_1^T)$$

- disadvantages:

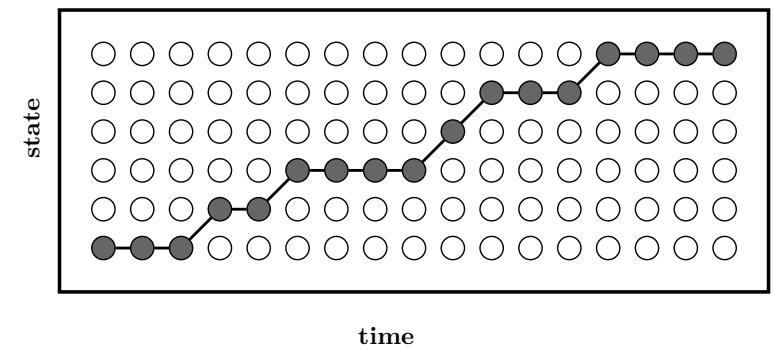
- wrong problem: we are NOT interested in the position details
- evaluation requires manual labelling of frames

- potential(?) remedy: modified loss function

- specify only the segment labels a_1^S ,
- i. e. without explicit duration
- optimize over segmentation path s_1^T

$$L[a_1^S | x_1^T] := \sum_{\tilde{y}_1^T} p(\tilde{y}_1^T | x_1^T) \cdot \min_{s_1^T} \left\{ \sum_t [1 - \delta(a_{s_t}, \tilde{y}_t)] \right\}$$

potential problem: asymmetric loss function



ASR: Frame vs. Segment Labels

remark on position-based symbol posterior probability over frame labels:

$$p_t(y_t|x_1^T) := \sum_{\tilde{y}_1^T : y_t = \tilde{y}_t} p(\tilde{y}_1^T|x_1^T)$$

note: the model $p(y_1^T|x_1^T)$ includes three components (at least!):

- language model,
- pronunciation lexicon,
- acoustic model, e. g. an RNN with LSTM extension

- practical error measure in praxis:
(sort of) edit distance with no deletions in reference string
- real advantage over conventional approach: yes?
- efficient implementation: open problem
- consider upper bound (to reduce computational complexity):

$$\begin{aligned}
 L[a_1^S | x_1^T] &:= \sum_{y_1^T} p(y_1^T | x_1^T) \cdot \min_{s_1^T} \left\{ \sum_t [1 - \delta(y_t, a_{s_t})] \right\} \\
 &\leq \min_{s_1^T} \left\{ \sum_{y_1^T} p(y_1^T | x_1^T) \cdot \sum_t [1 - \delta(y_t, a_{s_t})] \right\} \\
 &=: \min_{s_1^T} L[a_1^S, s_1^T | x_1^T]
 \end{aligned}$$

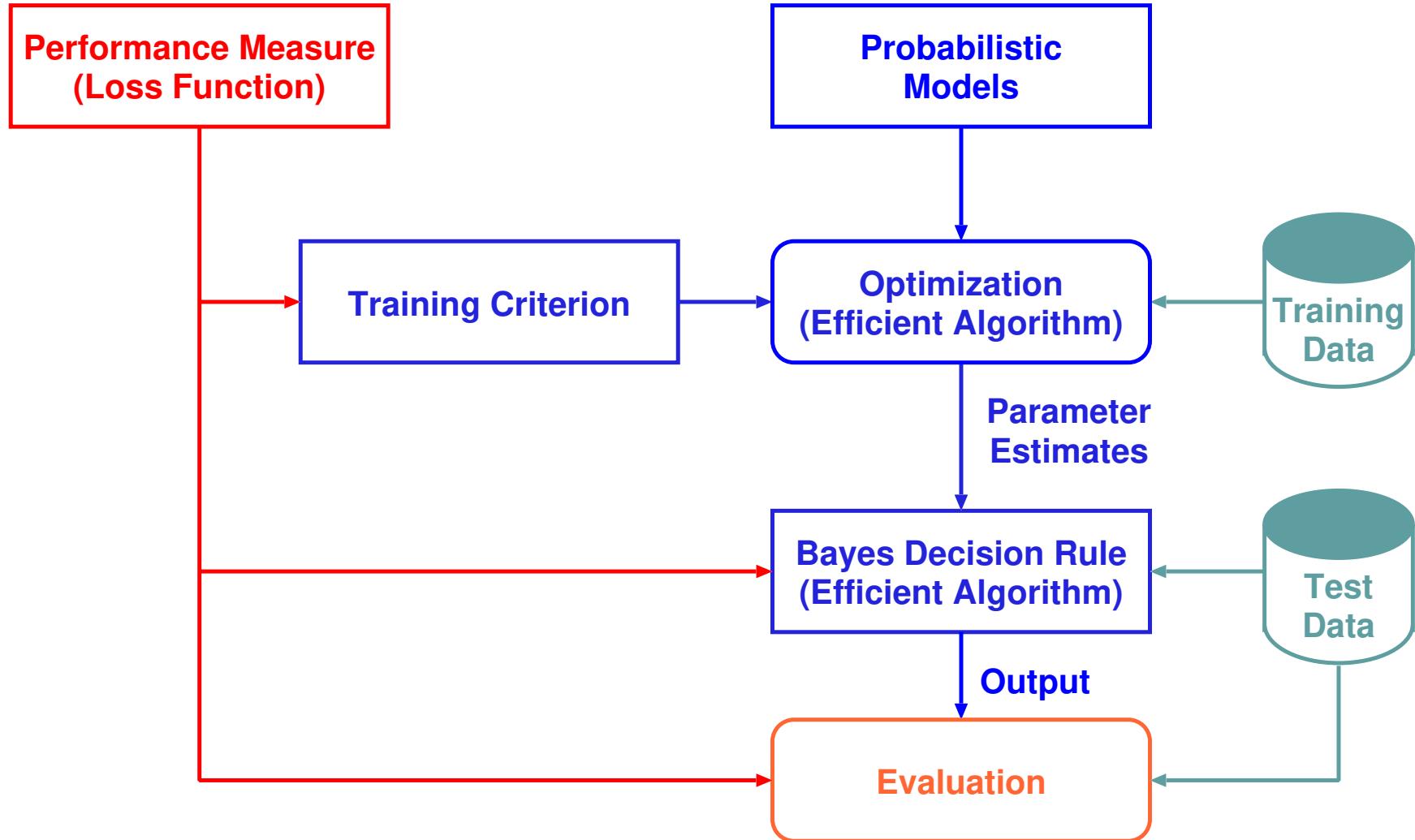
$$\begin{aligned}
 L[a_1^S, s_1^T | x_1^T] &= \sum_t \sum_{y_1^T} p(y_1^T | x_1^T) \cdot [1 - \delta(y_t, a_{s_t})] \\
 &= \sum_t \sum_{y_t} p_t(y_t | x_1^T) \cdot [1 - \delta(y_t, a_{s_t})] \\
 &= \sum_t [1 - p_t(y_t = a_{s_t} | x_1^T)]
 \end{aligned}$$

overview of approaches:

- generative: joint distribution $p(c, x)$
- discriminative: posterior distribution $p(c|x)$
- log-linear

procedure:

- first: single events
- later: extension to strings
- note: in principle, many considerations apply also to strings as a whole,
but details might be complicated (e.g. sum over symbol strings rather than single symbol)



re-write Bayes decision rule for atomic outputs with 0/1 loss function:

$$x \rightarrow c_*(x) = \arg \max_c \{pr(c|x)\} = \arg \max_c \{pr(c, x)\}$$

using the joint probability $pr(c, x)$ of the empirical distribution

approach:

- **Bayes decision rule: use probability model $p_\theta(c, x)$:**

$$x \rightarrow c_\theta(x) = \arg \max_c \{p_\theta(c, x)\}$$

- **problem with terminology:**

- often called Bayes decision rule
 - misnomer: correct term only for the true distribution, not for a model

Generative Model: Derived Distributions (cf. Types of Empirical Distributions)

- joint model of pairs (x, c) :

baseline model: $p_\vartheta(x, c)$

- marginal model of observations x :

$$p_\vartheta(x) = \sum_c p_\vartheta(x, c)$$

- marginal model over classes c :

$$p_\vartheta(c) = \sum_x p_\vartheta(x, c)$$

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

- (class) posterior model for observations x (note: $p_\vartheta(x) > 0$):

$$p_\vartheta(c|x) = p_\vartheta(x, c)/p_\vartheta(x)$$

- class conditional model of classes c (note: $p_\vartheta(c) > 0$):

$$p_\vartheta(x|c) = p_\vartheta(x, c)/p_\vartheta(c)$$

consider a representative sample (maybe huge) with ingredients:

- annotated data: pairs of (discrete) observation x and class index c :

$$(x_r, c_r), r = 1, \dots, R$$

and empirical distribution (= true distribution of this data set) $pr(x, c)$

note: distribution $pr(x, c)$ summarizes the whole data set $(x_r, c_r), r = 1, \dots, R$ (with no free parameters!)

- explicit separation of class prior and observation model:

$$\begin{aligned} pr(c, x) &= pr(c) \cdot pr(x|c) \\ p_\vartheta(c, x) &= p_\vartheta(c) \cdot p_\vartheta(x|c) \end{aligned}$$

this separation is useful:

- for defining the structure of generative models
- for learning the two models from different sets of training data

natural training criterion:

$$\begin{aligned}\hat{\vartheta} &= \operatorname{argmax}_{\vartheta} \left\{ \prod_r p_{\vartheta}(c_r, x_r) \right\} = \operatorname{argmax}_{\vartheta} \left\{ \sum_r \log p_{\vartheta}(c_r, x_r) \right\} \\ &= \operatorname{argmax}_{\vartheta} \left\{ \sum_r \log p_{\vartheta}(c_r) + \sum_r \log p_{\vartheta}(x_r | c_r) \right\}\end{aligned}$$

criterion re-written with empirical distribution:

$$\begin{aligned}\hat{\vartheta} &= \operatorname{argmax}_{\vartheta} \left\{ \sum_{c,x} pr(c, x) \log p_{\vartheta}(c, x) \right\} \\ &= \operatorname{argmax}_{\vartheta} \left\{ \sum_c pr(c) \log p_{\vartheta}(c) + \sum_{c,x} pr(c, x) \log p_{\vartheta}(x | c) \right\}\end{aligned}$$

derivative of log-likelihood function:

$$\sum_r \frac{\partial}{\partial \vartheta} \log p_{\vartheta}(c_r, x_r) \stackrel{!}{=} 0$$

**with closed-form solutions for many elementary models
(like Gaussian, Poisson, multinomial count model etc.)**

explicit separation of parameters for prior and class-conditional model:

- factorize the joint probability model using $\vartheta = (\lambda, \mu)$ with class prior $p_\lambda(c)$ and observation model $p_\mu(x|c)$:

$$p_\vartheta(c, x) = p_{(\lambda, \mu)}(c, x) = p_\lambda(c) \cdot p_\mu(x|c)$$

- maybe: use two independent training sets for prior and observation model:

class prior: $c_i, i = 1, \dots, I$
 $\hat{\lambda} = \operatorname{argmax}_\lambda \left\{ \sum_i \log p_\lambda(c_i) \right\}$

observation model: $[x_r, c_r], r = 1, \dots, R$
 $\hat{\mu} = \operatorname{argmax}_\mu \left\{ \sum_r \log p_\mu(x_r|c_r) \right\}$

**(later) useful in speech recognition at sequence level:
language model vs. acoustic model**

Generative Modelling and Maximum Likelihood Approach: Critical Discussion

advantages: nice mathematical properties

- nice property:
 - simulates the empirical distribution $pr(c, x)$
 - allows a full description of the 'recognition/decision problem'
 - separate learning of prior and observation model
- closed-form solutions for maximum likelihood:
 - prior $p_\lambda(c)$: relative frequencies (count models)
 - Gaussian models $p_\mu(x|c)$: empirical mean vector and covariance matrix (and many other elementary models): global optimum
 - Gaussian mixture and other models with hidden variables:
expectation-maximisation (EM) algorithm with local convergence and closed form solutions in each iteration

disadvantages: does not capture the true learning problem

- it does more than is needed in Bayes decision rule:
 $p_\vartheta(c|x)$ rather than $p_\vartheta(c, x)$
- well known in machine learning (since 1960s):
learning a density, i.e. a distribution over continuous-valued random variables, is hard (which was widely ignored in ASR)



5.2 Direct and Discriminative Modelling

two approaches to discriminative modelling $p(c|x)$:

- direct modelling, e. g. ANN or other discriminant function
- posterior distribution derived from a generative model

we will show:

- there is no fundamental difference between these two approaches
- unifying concept: re-normalization over all classes

5.2.1 Principle: Class Posterior Probability

definition: class posterior probability with parameters ϑ :

observation conditioned distribution over classes: with parameters ϑ

$$p_\vartheta(c|x) \quad \text{with} \quad \sum_c p_\vartheta(c|x) = 1$$

The normalization constrained is achieved by explicit re-normalization.

Distinguish two approaches:

- posterior form of an explicit generative model:

$$p_\vartheta(c|x) = \frac{p_\vartheta(c, x)}{\sum_{\tilde{c}} p_\vartheta(\tilde{c}, x)} \quad \text{with} \quad \sum_{c,x} p_\vartheta(c, x) = 1$$

- discriminative model:

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

with an arbitrary model $q_\vartheta(c, x) \geq 0$:

– normalization of $q_\vartheta(c, x)$ is not required!

note: this normalization is the only difference to a generative model.

– any structure is possible, e.g. log-linear model

– special case of log-linear model: softmax in ANNs

natural training criterion for parameters ϑ :

maximize the class posterior probability of the training data (c_r, x_r) , $r = 1, \dots, R$:

$$\begin{aligned}\hat{\vartheta} &:= \operatorname{argmax}_{\vartheta} \left\{ \prod_r p_{\vartheta}(c_r | x_r) \right\} = \operatorname{argmax}_{\vartheta} \left\{ \sum_r \log p_{\vartheta}(c_r | x_r) \right\} \\ &= \operatorname{argmax}_{\vartheta} \left\{ \sum_{c,x} pr(c, x) \log p_{\vartheta}(c|x) \right\}\end{aligned}$$

terminology: class posterior probability, cross-entropy,
MMI (maximum mutual information, for strings), ...

properties:

- explicitly discriminative in contrast to maximum likelihood estimation
- no closed-form solution
- for certain models/structures: global optimum (i.e. convex maximization problem)

remarks:

- other training criteria (as for ANNs) are possible in principle:
 squared error and binary cross-entropy
- (see later) all three criteria have a direct relation to classification error difference
 between the *model* $p_{\vartheta}(c|x)$ and the true distribution $pr(c|x)$

- re-write the cross-entropy criterion for the posterior distribution of a generative model:

$$\begin{aligned}\sum_r \log p_\vartheta(c_r|x_r) &= \sum_r \log \frac{p_\vartheta(x_r, c_r)}{\sum_c p_\vartheta(x_r, c)} \\ &= \sum_r \log p_\vartheta(x_r, c_r) - \sum_r \log \sum_c p_\vartheta(x_r, c)\end{aligned}$$

- assumption/approximation: weak dependence of the second term on ϑ due to sum

$$\begin{aligned}\hat{\vartheta} &= \operatorname{argmax}_\vartheta \left\{ \sum_r \log p_\vartheta(c_r|x_r) \right\} \cong \operatorname{argmax}_\vartheta \left\{ \sum_r \log p_\vartheta(x_r, c_r) \right\} \\ &= \operatorname{argmax}_\vartheta \left\{ \sum_r \log p_\vartheta(c_r) + \sum_r \log p_\vartheta(x_r|c_r) \right\}\end{aligned}$$

- result: simplified training criterion = maximum likelihood

- remarks:

- works only for correctly normalized models!
- possible advantage: closed-form solutions or similar (like EM algorithm)
(possible initialization for iterative procedures)

Class Posterior Probability: Derivative

posterior distribution by re-normalizing a generative model $p_\vartheta(x, c)$:

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

compute derivative for training data $(c_r, x_r), r = 1, \dots, R$:

$$\begin{aligned} \frac{\partial}{\partial \vartheta} \sum_r \log p_\vartheta(c_r | x_r) &= \sum_r \left[\frac{\partial}{\partial \vartheta} \log p_\vartheta(c_r, x_r) - \frac{\partial}{\partial \vartheta} \log \sum_c p_\vartheta(c, x_r) \right] \\ &= \sum_r \left[\frac{\partial}{\partial \vartheta} \log p_\vartheta(c_r, x_r) - \sum_c p_\vartheta(c|x_r) \frac{\partial}{\partial \vartheta} \log p_\vartheta(c, x_r) \right] \\ &= \sum_r \sum_c [\delta(c, c_r) - p_\vartheta(c|x_r)] \cdot \frac{\partial}{\partial \vartheta} \log p_\vartheta(c, x_r) \end{aligned}$$

remarks:

- interpretation: weighted maximum likelihood estimation
- verify approximate solution: by dropping the derivative of denominator, we have the maximum likelihood derivative
- exact solution: no closed form
- for Gaussian posterior: convex optimization problem (see next)

5.2.2 Example: Gaussian Posterior

joint Gaussian model $p(c, x)$ for (c, c) with $x \in \mathbb{R}^D$:

$$\begin{aligned} p(c, x) &= p(c) \cdot p(x|c) \\ p(x|c) &= \mathcal{N}(x|\mu_c, \Sigma_c) \\ &= \frac{1}{\sqrt{\det(2\pi\Sigma_c)}} \cdot \exp\left(-\frac{1}{2}(x - \mu_c)^t \Sigma_c^{-1} (x - \mu_c)\right) \\ &= \frac{1}{\sqrt{\det(2\pi\Sigma_c)}} \cdot \exp\left(-\frac{1}{2}x^t \Sigma_c^{-1} x + \mu_c^t \Sigma_c^{-1} x - \frac{1}{2}\mu_c^t \Sigma_c^{-1} \mu_c\right) \end{aligned}$$

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

characteristic property of Gaussian model:

generalized (squared) Euclidean distance as argument of exponential function

conventional maximum likelihood estimation:

- $p(c)$: relative frequency
- μ_c : empirical mean vector
- Σ_c : empirical covariance matrix

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

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

with the parameter dependent marginal distribution normalization term $Z_\vartheta(x)$ ($\stackrel{?}{=} p_\vartheta(x)$) and the ('constrained') parameters (see later):

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

remark: matrices Λ_c are defined to be symmetric!

summary of re-writing the Gaussian posterior:

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

important result for Gaussian posterior model:

- (log) quadratic in observations $x \in \mathbb{R}^D$
- (log) linear in parameters $\alpha_c \in \mathbb{R}$, $\lambda_c \in \mathbb{R}^D$, $\Lambda_c \in \mathbb{R}^{DxD}$
- special case of a log-linear model
 - with CONVEX optimization problem: unique solution
- note: the parameters have shift invariances (next slide)

- Gaussian posterior model is invariant under additive transformations:

$$\alpha_c \rightarrow \alpha_c + \alpha_0 \in \mathbb{R}$$

$$\lambda_c \rightarrow \lambda_c + \lambda_0 \in \mathbb{R}^D$$

$$\Lambda_c \rightarrow \Lambda_c + \Lambda_0 \in \mathbb{R}^{D \cdot D}$$

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

- for conversion back to Gaussian model:

exploit these invariances to satisfy the constraints of Gaussian model:

- positivity and normalization of $p(c)$
- positive definiteness of Σ_c
- invertibility of Σ_c

for more details: [IEEE TASLP, Heigold & Ney 12]

- note when going from generative Gaussian model to its posterior form:
parameters of (discriminatively trained) Gaussian models are not unique anymore!
- summary: EXACT equivalence between
 - posterior form of Gaussian model
 - log-linear model with quadratic observations (features)

observations:

- result: log-linear model
 - training criterion: convex optimization
 - general covariance matrix: softmax extended with 'squared' features
 - pooled covariance matrix: EXACT softmax
- more pairs of generative/discriminative models
[IEEE TASLP, Heigold & Ney 12]
- this presentation: hybrid approach in HMMs for ASR
alternative: explicit feature extraction: tandem approach in ASR

pooled covariance matrix $\Sigma_c = \Sigma$, i. e. $\Lambda_c = \Lambda$:

$$\begin{aligned} p_\theta(c|x) &= \frac{\exp(x^t \Lambda_c x + \lambda_c^t x + \alpha_c)}{\sum_{c'} \exp(x^t \Lambda_{c'} x + \lambda_{c'}^t x + \alpha_{c'})} \\ &= \frac{\exp(\lambda_c^t x + \alpha_c)}{\sum_{c'} \exp(\lambda_{c'}^t x + \alpha_{c'})} \end{aligned}$$

important result:

- linear dependence on parameter vectors λ_c and offset α_c
- equivalence: = softmax with weight vectors λ_c and bias α_c

remarks:

- linear dependence: hyperplanes as class boundaries
- compares well with SVMs (support vector machines),
but no public toolkit

5.2.3 Softmax Revisited

conventional view: consider MLP with softmax output:

- **input layer:** raw input vector z
- **hidden layers perform feature extraction,** $i = 1, \dots, I$:

$$x \rightarrow f_i(x) \quad f(x) := [f_1(x), \dots, f_I(x)]$$

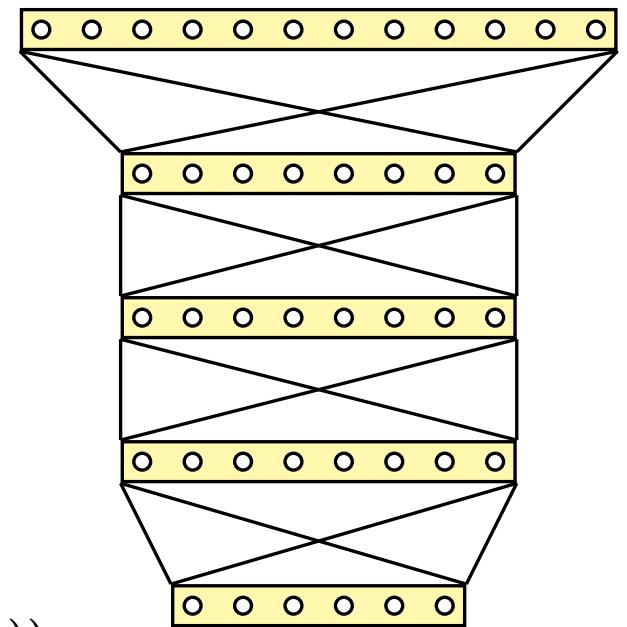
with feature vector $f(x) \in \mathbb{R}^I$ before output layer

note: no dependence on class labels $c = 1, \dots, C$

- **output layer:** probability distribution over classes c

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

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



interpretation of MLP with softmax output:

feature extraction followed by posterior form of a Gaussian model

5.3 Posterior Distribution and Log-Linear Modelling

generalization:

- starting point: dependencies in Gaussian posterior for vector $x = x_1^D \in \mathbb{R}^D$:

$$x \rightarrow y(x) := [1, x_1, \dots, x_d, \dots, x_D, x_1^2, \dots, x_{d_1}x_{d_2}, \dots, x_D^2]$$

$$p(c|x) = \exp(\lambda_c^t y(x)) / \sum_{c'} \exp(\lambda_{c'}^t y(x))$$

with parameter vectors λ_c

- define general feature functions $f_i(x)$ and associated log-linear model:

$$x \rightarrow f_i(x) \in \mathbb{R} \quad i = 1, \dots, I$$

$$p(c|x) = \exp\left(\sum_i \lambda_{ci} f_i(x)\right) / \sum_{c'} \exp\left(\sum_i \lambda_{c'i} f_i(x)\right)$$

with parameters $\lambda_{ci} \in \mathbb{R}$

- include class label c in feature functions $f_i(x, c)$:

$$(x, c) \rightarrow f_i(x, c) \in \mathbb{R} \quad i = 1, \dots, I$$

$$p(c|x) = \exp\left(\sum_i \lambda_i f_i(x, c)\right) / \sum_{c'} \exp\left(\sum_i \lambda_i f_i(x, c')\right)$$

with parameters $\lambda_i \in \mathbb{R}$

most general form of log-linear model:

- **define unconstrained (but suitable!) feature functions**

$$(x, c) \rightarrow f_i(x, c) \in \mathbb{R}$$

- **define associated log-linear model for $x \in \mathbb{R}^D$:**

$$p_{\{\lambda_i\}}(c|x_1^D) = \frac{\exp\left(\sum_i \lambda_i f_i(x, c)\right)}{\sum_{c'} \exp\left(\sum_i \lambda_i f_i(x, c')\right)}$$

with unknown parameter λ_i for each feature function $f_i(\cdot, \cdot)$

- **training: convex optimization problem for data (x_r, c_r) , $r = 1, \dots, R$**

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

conclusions:

- **there is only a single global optimum for this optimization problem**
- **in principle: any gradient search algorithm should work**

definition:

$$p_{\Lambda}(c|x) := \frac{q_{\Lambda}(c, x)}{\sum_{c'} q_{\Lambda}(c', x)}$$

$$\log q_{\Lambda}(c, x) := \sum_i \lambda_i f_i(x, c) \quad \Lambda = \{\lambda_i \in \mathbb{R}\}$$

compute the derivative of the cross-entropy training criterion for data (c_r, x_r) , $r = 1, \dots, R$:

$$\begin{aligned} \frac{\partial}{\partial \lambda_i} \sum_r \log p_{\Lambda}(c_r|x_r) &= \sum_r \sum_c [\delta(c, c_r) - p_{\Lambda}(c|x_r)] \frac{\partial}{\partial \lambda_i} \log q_{\Lambda}(c_r, x_r) \\ &= \sum_r \sum_c [\delta(c, c_r) - p_{\Lambda}(c|x_r)] f_i(c, x_r) \\ &= \sum_r f_i(c_r, x_r) - \sum_r \sum_c p_{\Lambda}(c|x_r) f_i(c, x_r) \\ &\stackrel{!}{=} 0 \end{aligned}$$

remark: log-linear model can be justified as a maximum entropy approach

assumption: training data $(c_r, x_r), r = 1, \dots, R$ with empirical distribution $pr(x)$

principles:

- **define feature functions $f_i(c, x), i = 1, \dots, I$ along with their counts:**

$$N_i := \sum_r f_i(c_r, x_r)$$

- **consider an unknown probability model $p(c|x)$ with feature constraints:**

$$N_i \stackrel{!}{=} \sum_r \sum_c p(c|x_r) f_i(c, x_r) = R \cdot \sum_x pr(x) \sum_c p(c|x) f_i(c, x)$$

- **find the model $p(c|x)$ that maximize the entropy criterion:**

$$\max_{\{p(c|x)\}} \left\{ - \sum_x pr(x) \sum_c p(c|x) \log p(c|x) \right\}$$

in other words: find the most general model that satisfies the feature constraints

- **solution: log-linear model with Lagrange multipliers for the feature constraints**

alternative definition of features, i. e. class-independent features:

$$\log q_{\Lambda}(c, x) := \sum_i \lambda_{ic} f_i(x) \quad \Lambda = \{\lambda_{ic} \in \mathbb{R}\}$$

$$p_{\Lambda}(c|x) := \frac{q_{\Lambda}(c, x)}{\sum_{c'} q_{\Lambda}(c', x)}$$

compute the derivative of the cross-entropy training criterion for data (c_r, x_r) , $r = 1, \dots, R$:

$$\begin{aligned} \frac{\partial}{\partial \lambda_{ic}} \sum_r \log p_{\Lambda}(c_r | x_r) &= \sum_r \left[\frac{\partial}{\partial \lambda_{ic}} \log q_{\Lambda}(c_r, x_r) - \frac{\partial}{\partial \lambda_{ic}} \log \sum_{c'} q_{\Lambda}(c', x_r) \right] \\ &= \sum_r [\delta(c, c_r) - p_{\Lambda}(c|x_r)] f_i(x_r) \\ &\stackrel{!}{=} 0 \end{aligned}$$

log-linear models (with given feature functions):

- unique optimum for class posterior model $p_{\Lambda}(c|x)$
- iterative search required due to lack of closed-form solution
- various strategies:
 - scaling methods: GIS/IIS algorithm
 - gradient methods (backpropagation): baseline, RPROP, L-BFGS, ADAM (?), ...
- potential numeric problems:
 - check convergence: plot training criterion over iteration number
 - check stepsize
- open problem: how to find the feature functions $f_i(x, c)$?
remedy:
 - learn them automatically
 - today's ANNs have exactly this form:
feature extraction + softmax (= log-linear model)
 - convexity is lost if feature extraction is included!

5.4 Summary

conceptual results:

- from generative Gaussian to discriminative/posterior Gaussian
- posterior Gaussian: log-linear model with convex optimization
- general log-linear model with general feature functions
- today's MLP: learned feature function + Gaussian posterior (= softmax)

6 Structured Output: String-to-String Processing

this chapter: from single events to strings

typical situation: strings in ASR, HWR, SMT

conclusions from modelling single events:

- key quantity: class posterior probability
- there was always an explicit normalization
(for generative, log-linear or ANN models)
- training criterion: cross-entropy,
 - what level of posterior probability:
sentence vs. symbol level (in case of synchronization)
 - approximation for generative models: maximum likelihood

we are facing three problems in string modelling:

- normalization requirement
- structural assumptions for strings
- synchronization mechanism between input and output string

preview for structured output, i. e. strings:

- starting point for strings: posterior probability
- concepts for synchronized strings:
 - factorization followed by local re-normalization
 - global re-normalization followed by a factorization
- additional extension for strings without synchronization:
 - alignment mechanisms like HMM or CTC
 - attention mechanism

additional aspects:

- training criterion: is it convex?
example: CRF with log-linear models for POS tagging
- are there 'nice' iterative solutions? like the EM algorithm?
- separation of subproblems, like language models learned from text

remarks about this chapter "string-to-string processing":

- **goal of this chapter: principles of architecture and interaction between components**
- **we will use (two) basic components: language model and observation model**
- **more details of components might be covered elsewhere:**
 - language model
 - HMM: hidden Markov model
 - basic structure and EM algorithm
 - attention mechanism
 - details of backpropagation
 - CRF models at sentence level
 - ...

6.1 Overview: Three Approaches to String Posterior Probability

overview:

- a large variety of models and combinations is possible
- here: selection of three most general concepts

notation for posterior probability:

- with synchronization: $p(c_1^N | x_1^N)$
- without synchronization: $p(c_1^N | x_1^T)$

note: the length N is also unknown (not always correct notation!)

training criterion with training data, i. e. pairs of strings (x_r, c_r) , $r = 1, \dots, R$:

$$\operatorname{argmax}_{\vartheta} \left\{ \sum_r \log p_{\vartheta}(c_r | x_r) \right\}$$

remarks:

- criterion: cross-entropy at sequence level (? symbol level)
- important aspect for all models: how difficult is this optimization problem?

three concepts:

- generative model (based on *joint* probability $p_\vartheta(c_1^N, x_1^T)$):

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

with two separate probabilistic models: the language model $p_\vartheta(c_1^N)$ and the observation model $p_\vartheta(x_1^T | c_1^N)$

- general model with re-normalization: CRF (conditional random field):

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

with arbitrary positive models: $Q_\vartheta(c_1, x_1^T)$, $q_\vartheta(c_1^N)$, $q_\vartheta(c_1^N | x_1^T)$
(maybe with 'posterior' normalization)

- direct model using factorization:

$$p_\vartheta(c_1^N | x_1^T) := \prod_n p_\vartheta(c_n | c_0^{n-1}, x_1^T) = \prod_n \frac{q_\vartheta(c_n; c_0^{n-1}, x_1^T)}{\sum_{c'} q_\vartheta(c'; c_0^{n-1}, x_1^T)}$$

training criterion: fits directly into cross-entropy training of an RNN

orthogonal concept: synchronization mechanism if required

generative and generalized model:

- **re-normalization:**
 - requires a sum over all strings
 - difficult problem with specific approximations/simplifications
- **Bayes decision rule:**
sum in denominator is NOT required!
- **training criterion: cross-entropy at sequence level**
 - denominator IS required
 - case of generative model:
approximation: drop denominator → maximum likelihood for strings

String-to-String Modelling: Design Considerations

considerations and decisions about the model structure:

- language model: do we want to separate it?
- re-normalization: global or local?
- for global re-normalization: generative or log-linear model?
- model components:
should each individual component be normalized?
- for input/output strings without synchronization:
what should be the synchronization mechanism?

there are different types of output labels c_1^N :

- automatic speech recognition:
 - frame labels (synchronous problem!)
 - state/segment labels (e.g. HMM/CART states)
 - phonemes/sounds or characters
 - subword units (e. g. based on BPE = byte pair encoding)
 - full-form words
- machine translation:
 - characters (only in exceptional cases)
 - subword units (e. g. based on BPE = byte pair encoding)
 - full-form words
- POS and semantic tagging (NLU):
 - special case: synchronized input/output
 - output labels refer to whole words (or word groups)
 - output alphabet depends on framework

note:

- there are different types of output labels
- no explicit definition of the type of output labels

6.2 Generative Models

string posterior model:

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

remarks:

- most important tasks: ASR, HWR and SMT
with two independent models: language model and observation model
- simplified training: maximum likelihood, i. e. numerator only
both for language model and observation model
- synchronization/alignment required between x_1^T and c_1^N :
generative Hidden Markov model with first-order hidden alignments s_1^T

$$p_\vartheta(x_1^T | c_1^N) = \sum_{s_1^T} p_\vartheta(x_1^T, s_1^T | c_1^N) = \sum_{s_1^T} \prod_t p_\vartheta(x_t, s_t | s_{t-1}, c_1^N)$$

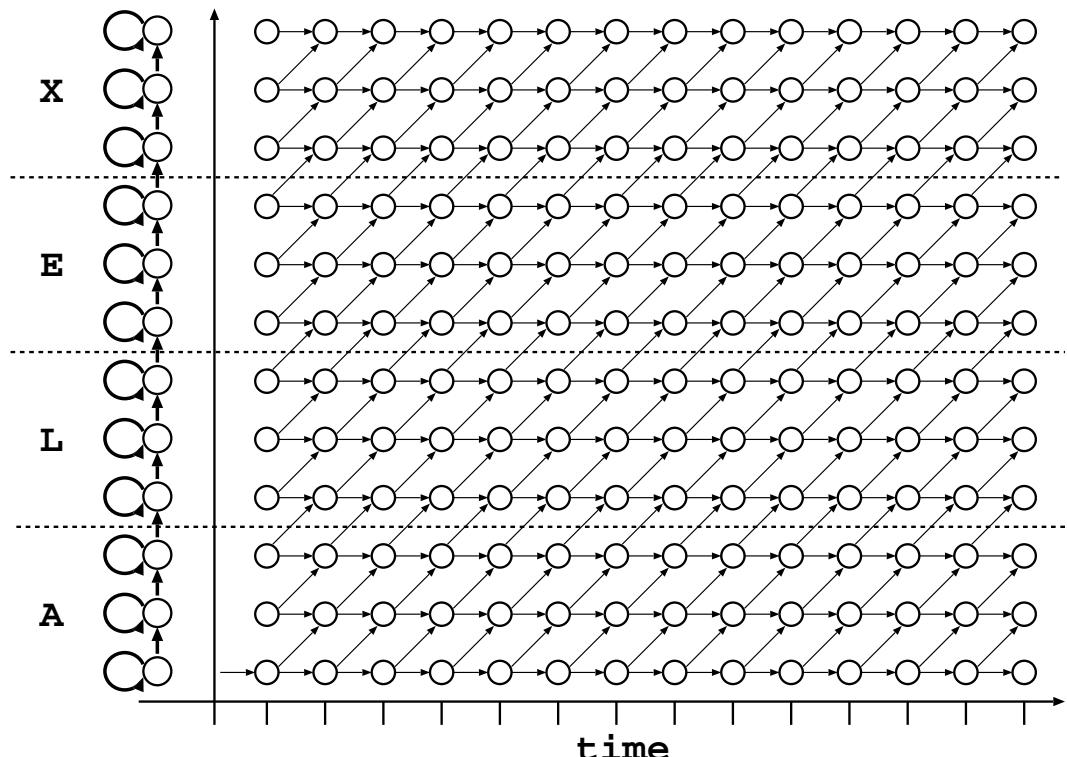
training: maximum likelihood using EM algorithm

- language model: $p_\vartheta(c_1^N) = \prod_n p_\vartheta(c_n | c_0^{n-1})$
 - using an m -gram Markov model or RNN
 - can be learned from text data only
 - independent training: maximum likelihood (= minimum perplexity)

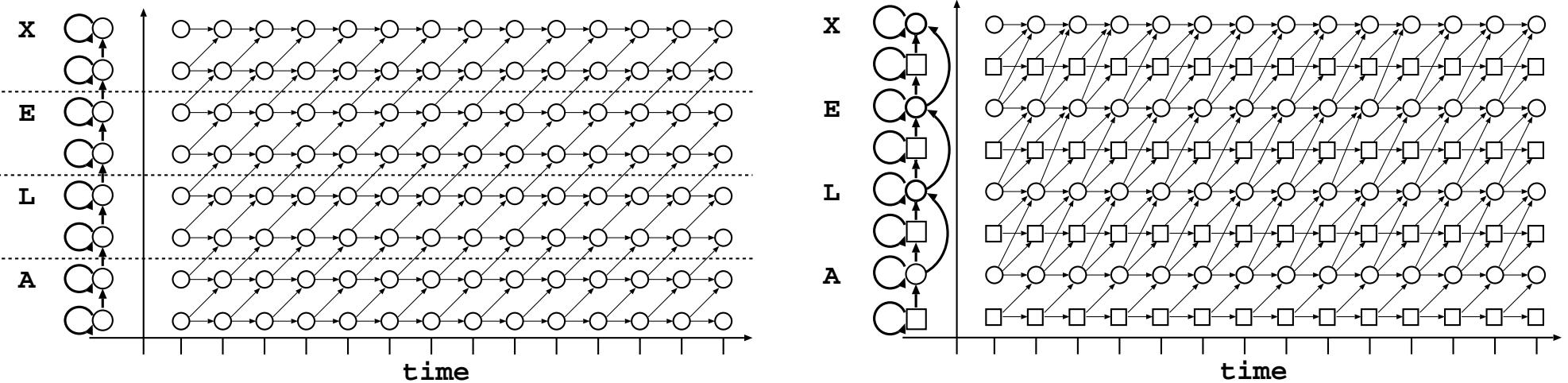
Illustration of HMM

remarks about HMM:

- **HMM along with its trellis**
- **states with labels:**
 $A_1, A_2, A_3, L_1, L_2, L_3, \dots$
- **two forms:**
 - generative form
 - hybrid form(spirit: similar to CTC)
- **efficient calculation of model score (how?)**
- **'natural training algorithm':**
EM – what does it optimize?



Comparison of Topologies: Hybrid Two-State HMM vs. CTC



result:

- CTC and two-state HMM very similar
- difference: transitions/transition probabilities and re-scaling by priors
- number of free parameters: virtually the same
- only output layer is different: C vs. $2 \cdot C$ output nodes

recognition criterion: best path with priors and transition probabilities

typical situation (e.g. in ASR):

- independent parameters for language model (λ) and observation model (μ)
- discriminative training for observation model only,
i. e. language model is kept fixed in discriminative training
- contribution of each training pair (c, x) to cross-entropy criterion
using simplified notation $x = x_1^T$ and $c = c_1^N$:

$$\log p_{\lambda\mu}(c|x) = \log \frac{p_\lambda(c) \cdot p_\mu(x|c)}{\sum_{c'} p_\lambda(c') \cdot p_\mu(x|c')} = \log \frac{p_{\lambda\mu}(c, x)}{p_{\lambda\mu}(x)}$$

effect of language model prior: **sequence discriminative training**
(note: similar for generalized model!)

- for fixed parameters λ , the contribution of each training pair (c, x) to the optimization over μ is equivalent to the term (= mutual information):

$$\operatorname{argmax}_\mu \left\{ \log \frac{p_{\lambda\mu}(c, x)}{p_\lambda(c) \cdot p_{\lambda\mu}(x)} \right\} = \operatorname{argmax}_\mu \left\{ \log \frac{p_{\lambda\mu}(c, x)}{p_{\lambda\mu}(x)} - \log p_\lambda(c) \right\}$$

terminology: **MMI** = *maximum mutual information*

Generative Model: Discriminative Training in ASR (MMI, sequence discriminative training)

implementation for ASR:

- general concept: gradient search
- sum in denominator:
can only be computed by approximations:
 - using word hypothesis lattice (or N-Best lists: inefficient)
 - using a simplified language model $p_\vartheta(c_1^N)$
[Povey, Interspeech 2016: lattice-free MMI]
- Povey's and all approaches:
a large number of heuristic tricks

typical example of synchronized string:

- **specific task: POS and semantic tagging:**
 - input x_1^N : **sequence of words**
 - output c_1^N : **sequence of associated syntactic/semantic categories**
- **explicit form of model (baseline):**

$$p_\vartheta(c_1^N | x_1^N) := \frac{\prod_n p_\vartheta(c_n | c_{n-1}) \cdot p_\vartheta(x_n | c_n)}{\sum_{\tilde{c}_1^N} \prod_n p_\vartheta(\tilde{c}_n | \tilde{c}_{n-1}) \cdot p_\vartheta(x_n | \tilde{c}_n)}$$

with first-order prior model and zero-order membership model

- **training criterion: maximum likelihood**
 - relative frequencies based on counts of events (c_{n-1}, c_n) and (c_n, x_n)
 - maybe smoothing required!
- **training criterion: cross-entropy at sequence level:**
 - exact log-linear model at sequence level (see next section):
convex optimization problem
 - efficient solution due to first-order dependencies (sum in denominator!) and convexity

other example: frame labelling in ASR (never done)

6.3 Generalized Models (CRF)

general model with re-normalization: CRF (conditional random field):

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

with arbitrary positive models: $Q_{\vartheta}(c_1, x_1^T), q_{\vartheta}(c_1^N), q_{\vartheta}(c_1^N | x_1^T)$
(maybe with 'posterior' normalization)

conditional random field (CRF): quite general definition:

- applies to (nearly) any distribution of random variables over strings (1-dim), images (2-dim) or graphs
- explicit global renormalization

general model with re-normalization: CRF (conditional random field):

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

typical CRF structure in HLT:

- strings: one-dimensional
- synchronized situation: $p_\vartheta(c_1^N | x_1^N)$
- Markov chain for output string: e. g. first order:

$$Q(c_1^N, x_1^N) := \prod_n q_n(c_{n-1}, c_n, x_1^N)$$

note: Markov chain allows an efficient calculation of denominator

- log-linear structure in $q_n(c_{n-1}, c_n, x_1^N)$
- training criterion: cross-entropy at sequence level:
= convex optimization problem

experimental facts: two successful independent concepts:

- RNN: recurrent neural network for sequence processing (e.g. LSTM)
- CRF: log-linear model with global re-normalization

combination of RNN and CRF

[Lampl & Ballesteros⁺ 16], [Goldberg 17, section 19.4.2, pp. 229-232]:

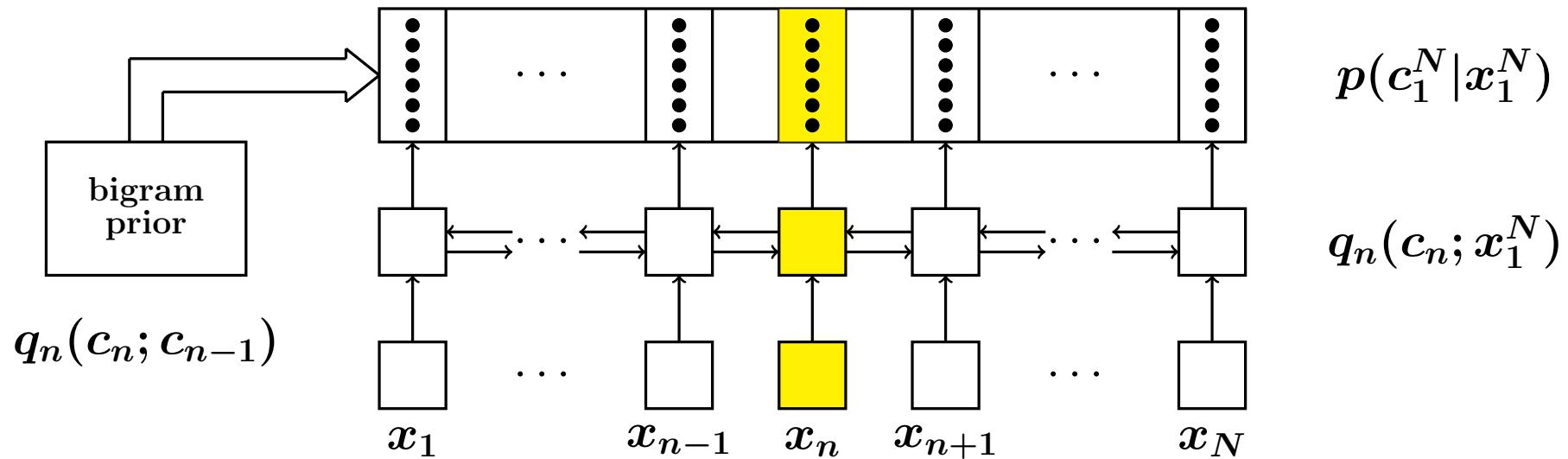
- RNN in bidirectional variant with scores: $q_n(c_n; x_1^N) > 0$
- bigram prior: transition matrix with first-order dependencies: $q(c_{n-1}; c_n) > 0$
- global re-normalization: RNN based CRF:

$$\begin{aligned} p(c_1^N | x_1^N) &:= \frac{\prod_n q^\alpha(c_n; c_{n-1}) \cdot q_n^\beta(c_n; x_1^N)}{\sum_{\tilde{c}_1^N} \prod_n q^\alpha(\tilde{c}_n; \tilde{c}_{n-1}) \cdot q_n^\beta(\tilde{c}_n; x_1^N)} \\ &= \frac{\exp \left(\sum_n [\alpha \cdot \log q(c_n; c_{n-1}) + \beta \cdot \log q_n(c_n; x_1^N)] \right)}{\sum_{\tilde{c}_1^N} \exp \left(\sum_n [\alpha \cdot \log q(\tilde{c}_n; \tilde{c}_{n-1}) + \beta \cdot \log q_n(\tilde{c}_n; x_1^N)] \right)} \end{aligned}$$

compare with re-normalized generative model with NORMALIZED components:

$$p(c_1^N | x_1^N) := \frac{\prod_n p(c_n | c_{n-1}) \cdot p(x_n | c_n)}{\sum_{\tilde{c}_1^N} \prod_n p(\tilde{c}_n | \tilde{c}_{n-1}) \cdot p(x_n | \tilde{c}_n)}$$

Illustration of RNN based CRFs



$$p(c_1^N | x_1^N) = \frac{\exp \left(\sum_n [\alpha \cdot \log q(c_n; c_{n-1}) + \beta \cdot \log q_n(c_n; x_1^N)] \right)}{\sum_{\tilde{c}_1^N} \exp \left(\sum_n [\alpha \cdot \log q(\tilde{c}_n; \tilde{c}_{n-1}) + \beta \cdot \log q_n(\tilde{c}_n; x_1^N)] \right)}$$

- training data: pairs of input-output string pairs

$$(X_r, C_r), \quad r = 1, \dots, R$$

with $X_r = x_{r1} \dots x_{rn} \dots x_{rN_r}$ and $C_r = c_{r1} \dots c_{rn} \dots c_{rN_r}$

- overall model $p_\vartheta(c_1^N | x_1^N)$ has free parameters:
all parameters of RNN and transition matrix (and log-linear parameters)
- training criterion:

$$\operatorname{argmax}_\vartheta \left\{ \sum_r \log p_\vartheta(C_r | X_r) \right\}$$

optimization strategy:

- in principle: backpropagation
- in combination with forward-backward algorithm (sort of dynamic programming)
for computing the denominator (with first-order structure!)

experimental results for named entity tagging [Lampl & Ballesteros⁺ 16]:

- best results and systematic improvements over conventional methods
- main advantage: automatic feature extraction for multi-lingual tasks

components of a log-linear model with exponents α and β :

- language model prior $q(c_n|c_0^{n-1})$
- observation model $q_\vartheta(c_n|c_{n-1}, x_1^N) = q_{n,\vartheta}(c_n|c_{n-1}, x_1^N)$

- **global (= sequence-level) re-normalization (= conditional random field, CRF):**

$$p_\vartheta(c_1^N|x_1^N) = \frac{\prod_n [q^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_{n-1}, x_1^N)]}{\sum_{\tilde{c}_1^N} \prod_n [q^\alpha(\tilde{c}_n|\tilde{c}_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|\tilde{c}_{n-1}, x_1^N)]}$$

- training criterion: sequence-level cross-entropy = sequence discriminative training
- potential problem: sum over all symbol sequences

- **local (= symbol-level) re-normalization (= direct model):**

$$\begin{aligned} p_\vartheta(c_1^N|x_1^N) &= \prod_n p_\vartheta(c_n|c_0^{n-1}, x_1^N) = \prod_n \frac{q^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_{n-1}, x_1^N)}{\sum_{\tilde{c}_n} [q^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_{n-1}, x_1^N)]} \\ &= \frac{\prod_n [q^\alpha(c_n|c_0^{n-1}) \cdot q_\vartheta^\beta(c_n|c_{n-1}, x_1^N)]}{\prod_n \sum_{\tilde{c}_n} [q^\alpha(\tilde{c}_n|c_0^{n-1}) \cdot q_\vartheta^\beta(\tilde{c}_n|c_{n-1}, x_1^N)]} \end{aligned}$$

- training criterion: sequence-level cross-entropy = symbol-level cross-entropy
- potential advantage: easy re-normalization

**question: how can we convert these two types of models?
(conversion might be needed in some contexts)**

- **conversion from local to global model:**

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

implementation: easy (by definition)

- **conversion from global to local model:**

$$p_{\vartheta}(c_1^N | x_1^N) := \frac{\prod_n [q^{\alpha}(c_n | c_0^{n-1}) \cdot q_{\vartheta}^{\beta}(c_n | c_{n-1}, x_1^N)]}{\sum_{\tilde{c}_1^N} \prod_n [q^{\alpha}(\tilde{c}_n | \tilde{c}_0^{n-1}) \cdot q_{\vartheta}^{\beta}(\tilde{c}_n | \tilde{c}_{n-1}, x_1^N)]}$$

$$p_{\vartheta}(c_n | c_0^{n-1}, x_1^N) = ...?$$

implementation:

- **principle: clear**
- **efficiency: depends on structure of global model**

No Synchronization

typical situation:

- HMM for alignment: HCRF = hidden CRF
- training criterion: cross-entropy at sequence level
 - no difference anymore to hybrid HMM with sequence discriminative training
- convexity is lost due to sum over hidden variable
- not widely used (?)

**Quattoni & Wang et al.: Hidden Conditional Random Fields,
IEEE Trans. PAMI, pp. 1848–1852, 2007.**

6.4 Direct Models

direct model using factorization:

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

training criterion: fits directly into cross-entropy training of an RNN

remarks:

- difference to the two previous models:
local normalization as opposed to global normalization:

$$p_{\vartheta}(c_n | c_0^{n-1}, x_1^T) = \frac{q_{\vartheta}(c_n; c_0^{n-1}, x_1^T)}{\sum_{c'} q_{\vartheta}(c'; c_0^{n-1}, x_1^T)}$$

- easy to compute (in training and testing)
- note: cannot be dropped in Bayes decision rule!
- synchronization mechanisms:
 - attention model
 - direct HMM (inverted direction of alignments)

definition of model:

$$p_{\vartheta}(c_1^N | x_1^T) = \prod_n p_{\vartheta}(c_n | c_0^{n-1}, x_1^T) = \prod_n p_{\vartheta}(c_n | \tilde{c}_{n-1}, s_{n-1}, r_n)$$

approach:

- **input: bidirectional RNN over input positions t :** $x_1^T \rightarrow h_t = h_t(x_1^T)$
- **output: unidirectional RNN over output positions n :**

$$p_{\vartheta}(c_n | \tilde{c}_{n-1}, s_{n-1}, r_n)$$

with vector representation \tilde{c}_n of symbol c_n

and RNN state vector $s_n = S(s_{n-1}, \tilde{c}_n, r_n)$

and context vector $r_n = R(s_{n-1}, h_1^T)$

- **context vector r_n : weighted average of input representations h_t :**

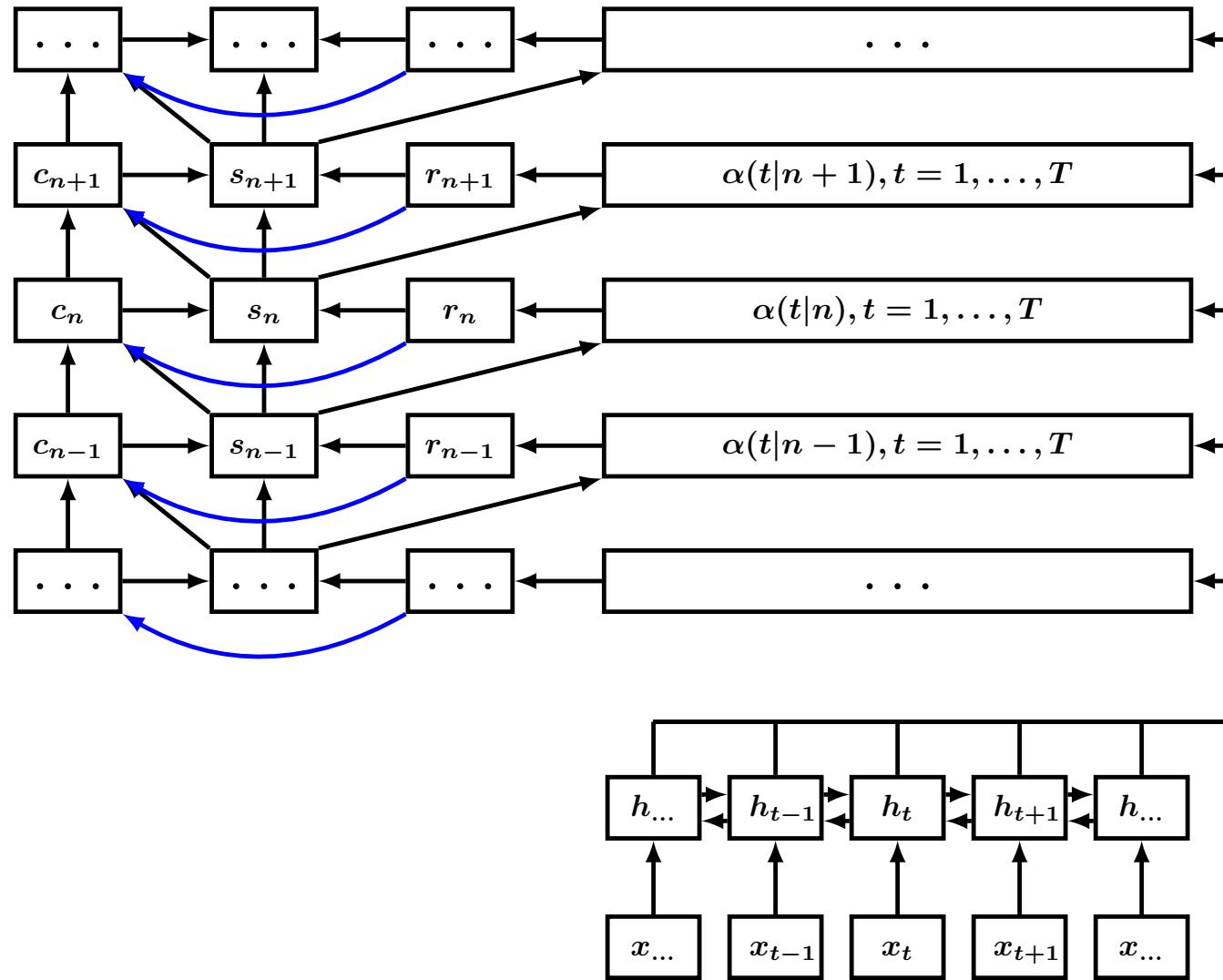
$$r_n = \sum_t \alpha(t|n, s_{n-1}, h_1^T) \cdot h_t \quad \alpha(t|n, s_{n-1}, h_1^T) = \frac{\exp(A[s_{n-1}, h_t])}{\sum_{t'} \exp(A[s_{n-1}, h_{t'}])}$$

with the normalized attention weights $\alpha(t|n, s_{n-1}, h_1^T)$

and real-valued attention scores $A[s_{n-1}, h_t]$, e. g. generalized dot product $s_{n-1}^T \cdot W \cdot h_t$

- **training criterion at sequence level: = sum over input positions**

Direct Model: Attention Mechanism



Direct Model: Attention Mechanism Sequential Order of Operations

preparations:

- **input preprocessing:**

$$x_1^T \rightarrow h_t = H_t(x_1^T)$$

- **available at position ($n - 1$):**

$$c_{n-1}, s_{n-1}, r_{n-1}$$

**sequence of operations
for positions $n = 1, \dots, N$:**

1. **attention weights:**

$$\alpha(t|n, s_{n-1}, h_1^T) = \dots$$

2. **context vector:**

$$r_n = \sum_t \alpha(t|n, s_{n-1}, h_1^T) \cdot h_t$$

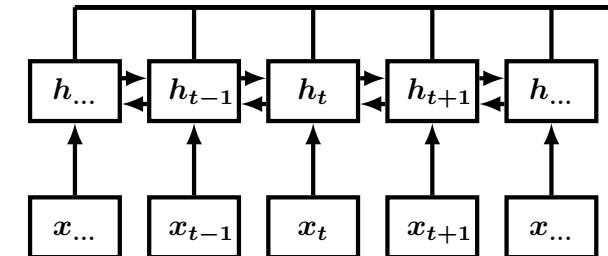
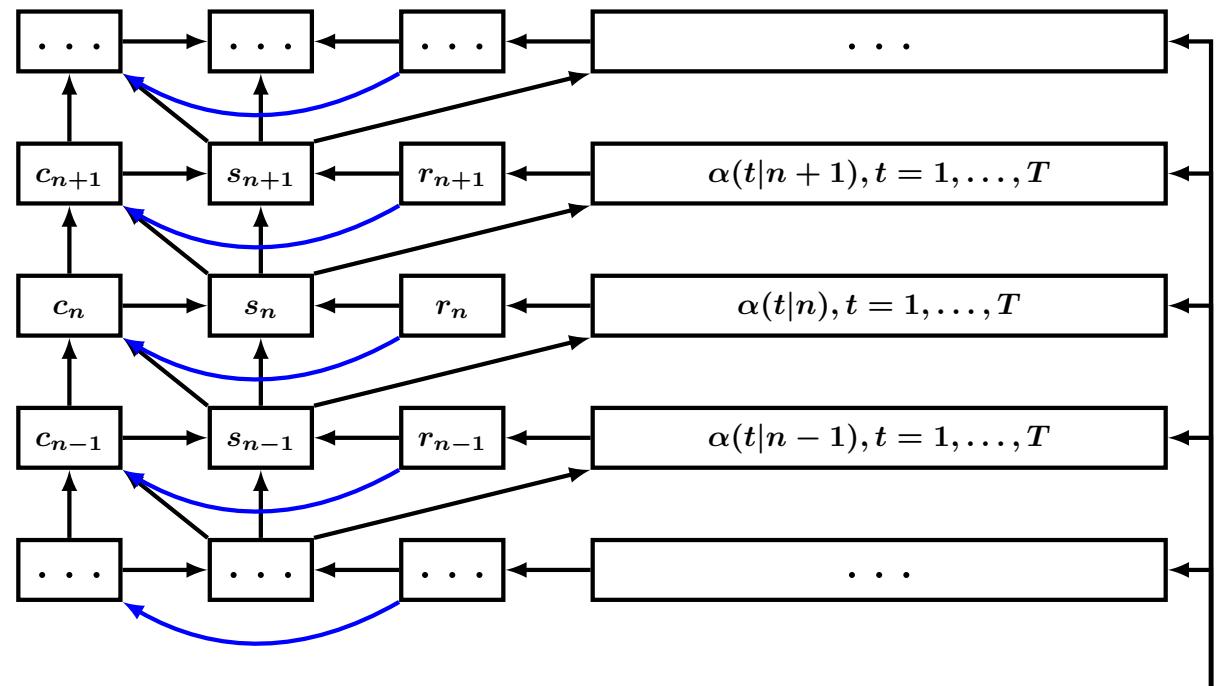
3. **output distribution:**

$$p(c_n|c_{n-1}, s_{n-1}, r_n)$$

4. **state vector:**

$$s_n = S(s_{n-1}, c_n, r_n)$$

complexity: $T \cdot N$



Direct Model: Attention Mechanism Summary of Principles

- **input sequence:**
bidirectional LSTM-RNN
- **output sequence:** sort of language model with
unidirectional LSTM-RNN
- **context vector:**
 - provides the link between input and output sequence
 - is computed as weighted average of representation vectors of input
 - weights: attention mechanism

generative HMM with first-order hidden alignments s_1^T :

$$p_\vartheta(x_1^T | c_1^N) = \sum_{s_1^T} p_\vartheta(x_1^T, s_1^T | c_1^N) = \sum_{s_1^T} \prod_t p_\vartheta(x_t, s_t | s_{t-1}, c_1^N)$$

posterior distribution based on HMM with first-order hidden alignments t_1^N :

$$p_\vartheta(c_1^N | x_1^T) = \sum_{t_1^N} p_\vartheta(c_1^N, t_1^N | x_1^T) = \sum_{t_1^N} \prod_n p_\vartheta(c_n, t_n | t_{n-1}, c_0^{n-1}, x_1^T)$$

remarks:

- similar goal as CTC (= connectionist temporal classification):
generate output symbols c_1^N rather than input string x_1^T
- implementation similar to attention model:
 - input positions: bidirectional RNN
 - output positions: unidirectional RNN
- training: cross-entropy at sequence level:
backpropagation within (sort of) EM framework

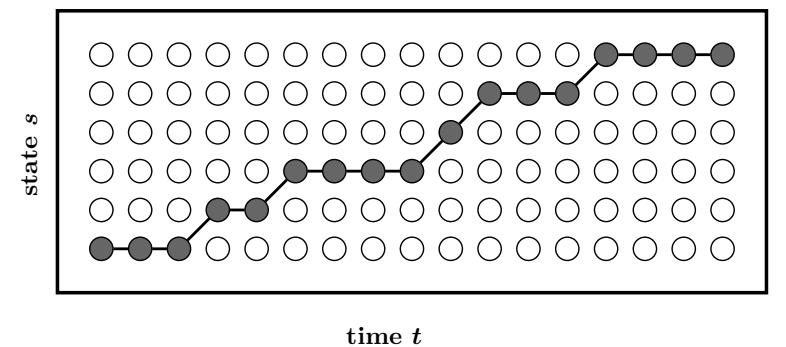
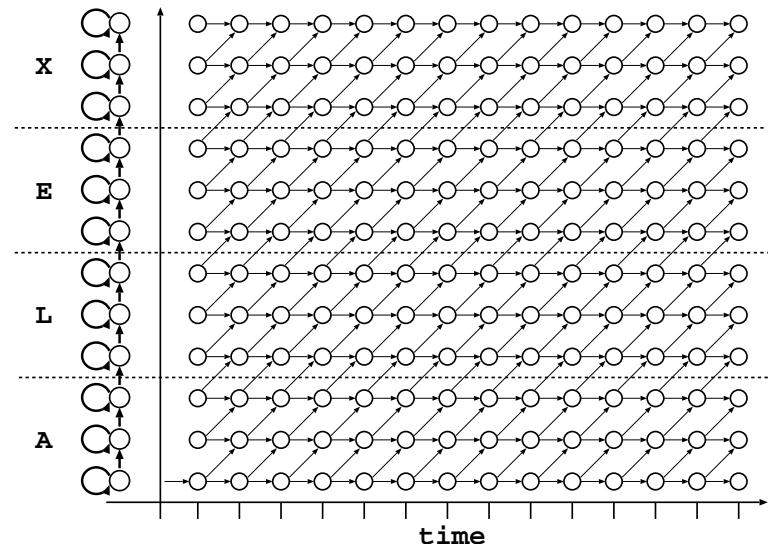
- output labels c_1^N (= state/segment labels) along with hidden (segment) boundaries t_1^n :

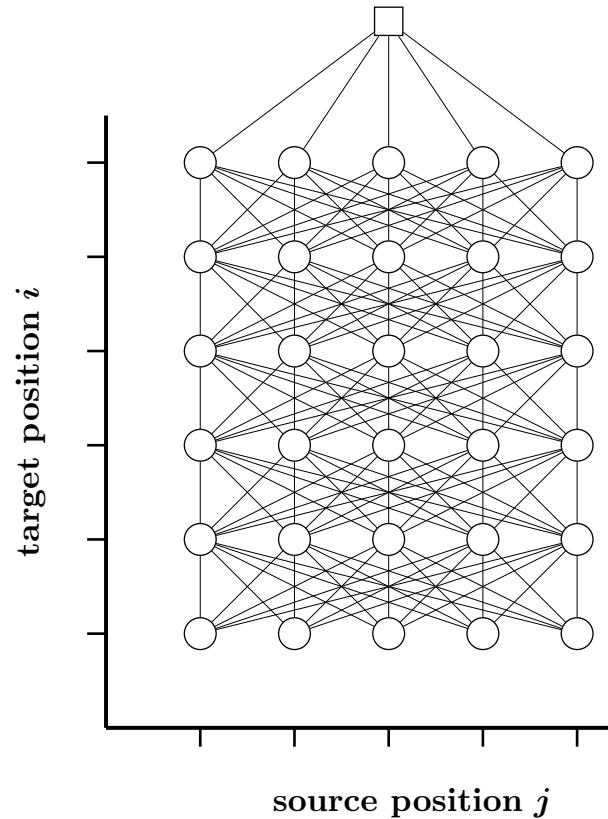
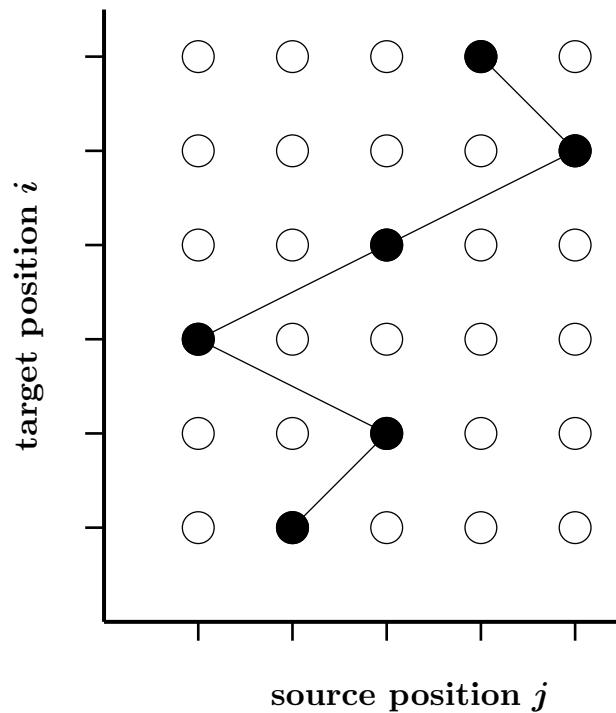
$$\begin{aligned} c_1^N &:= c_1 \dots c_n \dots c_N \\ t_1^n &:= t_1 \dots t_n \dots t_N \end{aligned}$$

- definition: t_n is the end boundary of segment n
- remark: the same label sequence c_1^N has many hidden segmentations t_1^n
- model with hidden segment boundaries t_1^n :

$$\begin{aligned} p_\vartheta(c_1^N | x_1^T) &= \sum_{t_1^n} p_\vartheta(c_1^N, t_1^n | x_1^T) \\ p_\vartheta(c_1^N, t_1^n | x_1^T) &= \prod_n p_\vartheta(c_n, t_n | t_{n-1}, c_0^{n-1}, x_1^T) \end{aligned}$$

with first-order dependence in sequence t_1^n





- **alignment direction:** from target to source
- **sum over alignments (first-order model):**
 - can be represented by trellis
 - can be computed EXACTLY in training and decoding
- **additional advantage:**
convenient for decoding using a bottom-to-top search

direct model: posterior distribution with hidden alignments t_1^N :

- first-order model:

$$p_\vartheta(c_1^N | x_1^T) = \sum_{t_1^N} p_\vartheta(c_1^N, t_1^N | x_1^T) = \sum_{t_1^N} \prod_n p_\vartheta(c_n, t_n | t_{n-1}, c_0^{n-1}, x_1^T)$$

efficient calculation for a given string pair (c_1^N, x_1^T) :

$$\begin{aligned} Q(n, t) &= \sum_{t'} Q(n-1, t') \cdot p_\vartheta(c_n, t | t', c_0^{n-1}, x_1^T) \\ p_\vartheta(c_1^N | x_1^T) &:= Q(N, T) \quad \text{or:} \quad = \sum_t Q(N, t) \end{aligned}$$

complexity: $N \cdot T^2$

- zero-order: allows interchange of sum and product:

$$p_\vartheta(c_1^N | x_1^T) = \sum_{t_1^N} \prod_n p_\vartheta(c_n, t_n | -, c_0^{n-1}, x_1^T) = \prod_n \sum_{t_n} p_\vartheta(c_n, t_n | c_0^{n-1}, x_1^T)$$

result: mixture model over input positions t for each output position n

complexity: $N \cdot T$

most general form of direct first-order HMM:

$$p_{\theta}(c_1^N, t_1^N | x_1^T) = \prod_n p_{\theta}(c_n, t_n | t_{n-1}, c_0^{n-1}, x_1^T)$$

- principle concept: direct hidden alignments with first-order (or zero-order) dependencies
- model score: sum over all alignments
 - efficient calculation by forward algorithm/dynamic programming
 - note: independent of realization details!
- details of realization:
question: how to model dependencies using ANNs?
- training criterion: sequence posterior probability
 - (sort of) EM algorithm with backpropagation inside
 - details: see later chapter on MT
- dependence of implementation on task:
ASR, HWR, MT

Direct HMM using RNNs

most general form of direct first-order HMM:

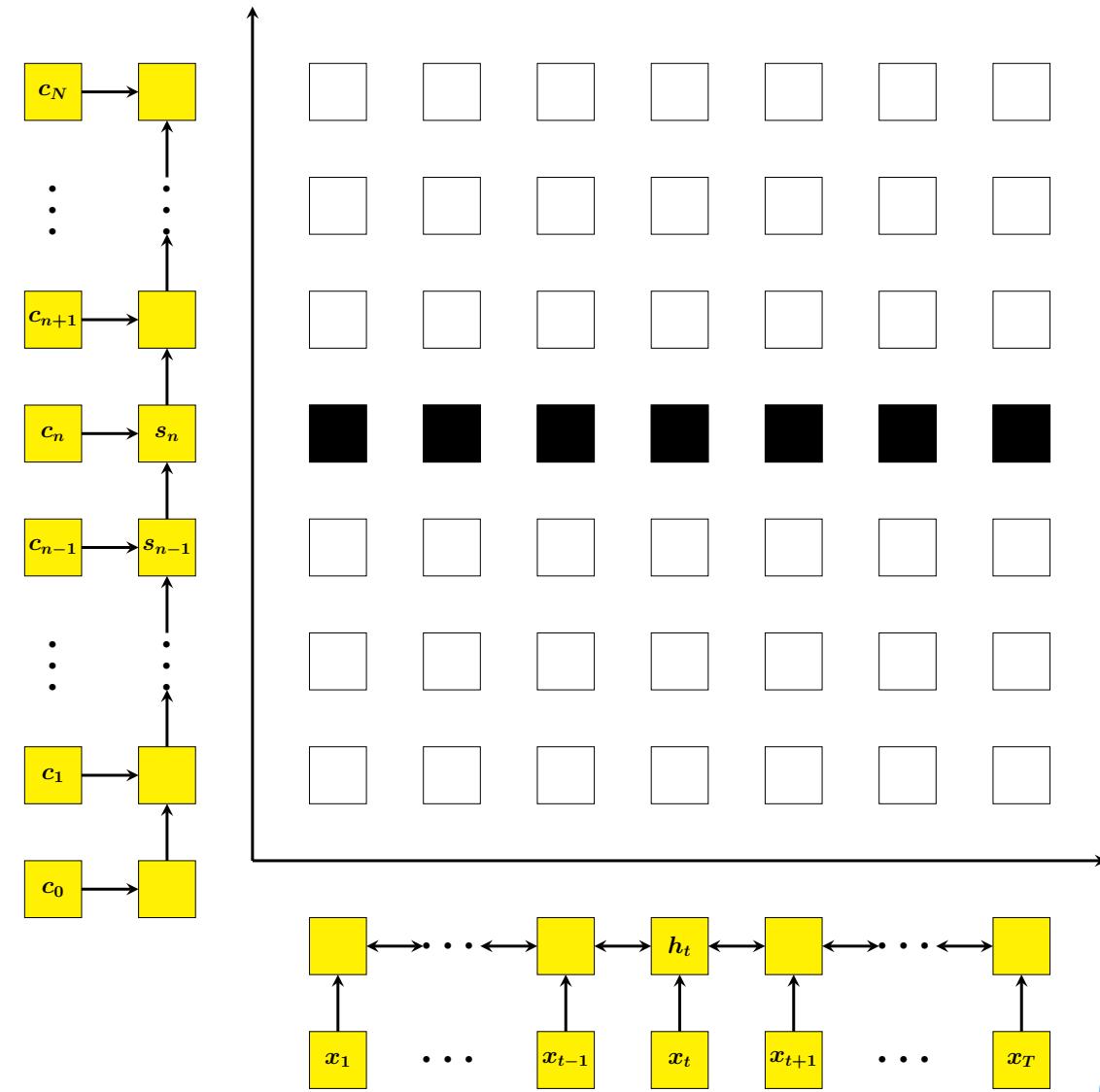
$$p_{\vartheta}(c_n, t_n | t_{n-1}, c_0^{n-1}, x_1^T) = p_{\vartheta}(t_n | t_{n-1}, c_0^{n-1}, x_1^T) \cdot p_{\vartheta}(c_n | t_{n-1}^n, c_0^{n-1}, x_1^T)$$

alignment model using RNNs:

$$\begin{aligned} p_{\vartheta}(t_n | t_{n-1}, c_0^{n-1}, x_1^T) \\ = p_{\vartheta}(t_n | t_{n-1}, c_{n-1}, s_{n-1}, h_1^T) \\ = p_{\vartheta}(t_n | c_{n-1}, s_{n-1}, h_{t_{n-1}}) \end{aligned}$$

lexicon model using RNNs:

$$\begin{aligned} p_{\vartheta}(c_n | t_{n-1}^n, c_0^{n-1}, x_1^T) \\ = p_{\vartheta}(c_n | t_{n-1}^n, c_{n-1}, s_{n-1}, h_1^T) \\ = p_{\vartheta}(c_n | c_{n-1}, s_{n-1}, h_{t_{n-1}}, h_{t_n}) \end{aligned}$$



compare:

- posterior distribution with hidden alignments t_1^N :

$$\begin{aligned}
 p_{\vartheta}(c_1^N | x_1^T) &= \sum_{t_1^N} p_{\vartheta}(c_1^N, t_1^N | x_1^T) = \sum_{t_1^N} \prod_n p_{\vartheta}(c_n, t_n | t_0^{n-1}, c_0^{n-1}, x_1^T) \\
 (\text{zero-order assumption}) &= \prod_n \sum_{t_n} p_{\vartheta}(c_n, t_n | -, c_0^{n-1}, x_1^T) \\
 &= \prod_n \sum_{t_n} [p_{\vartheta}(t_n | c_0^{n-1}, x_1^T) \cdot p_{\vartheta}(c_n | t_n, c_0^{n-1}, x_1^T)] \\
 &= \prod_n \sum_{t_n} [p_{\vartheta}(t_n | n, s_{n-1}, h_1^T) \cdot p_{\vartheta}(c_n | s_{n-1}, c_{n-1}, h_{t_n})}
 \end{aligned}$$

result: mixture model over input positions $t_n = t = 1, \dots, T$

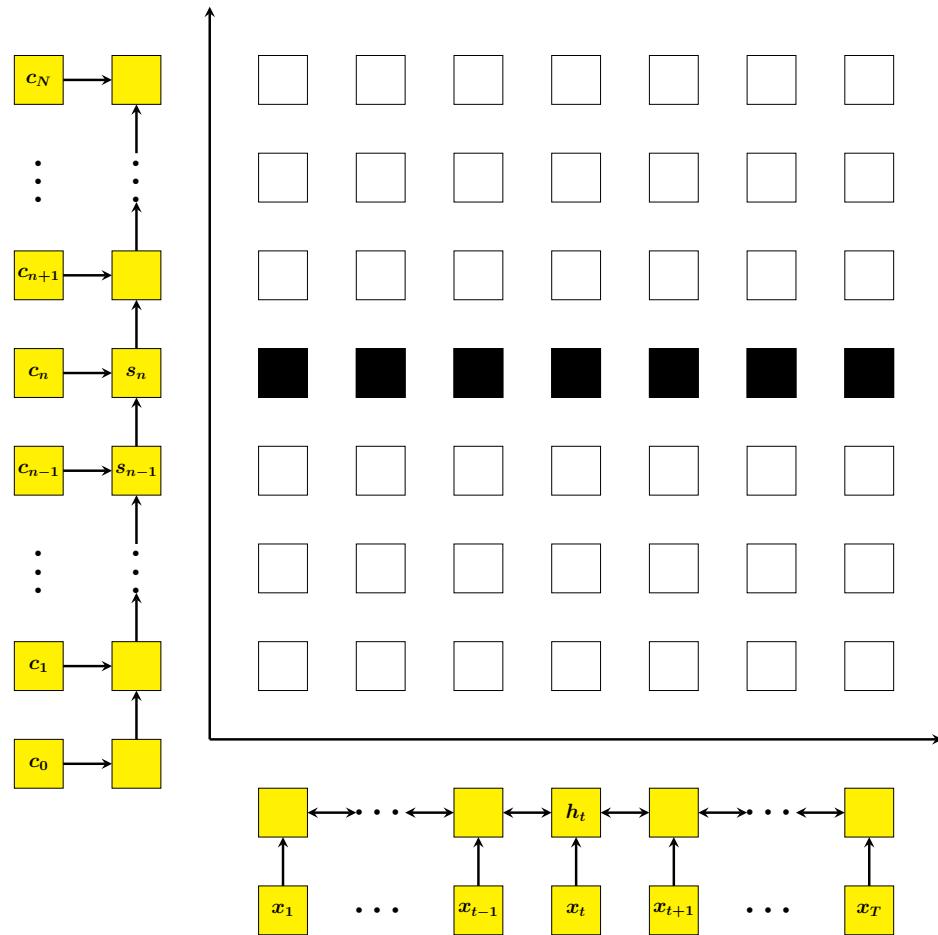
- posterior distribution with attention mechanism:

$$\begin{aligned}
 p_{\vartheta}(c_1^N | x_1^T) &= \prod_n p_{\vartheta}(c_n | \tilde{c}_{n-1}, s_{n-1}, r_n) \\
 r_n &= \sum_t \alpha(t | n, s_{n-1}, h_1^T) \cdot h_t
 \end{aligned}$$

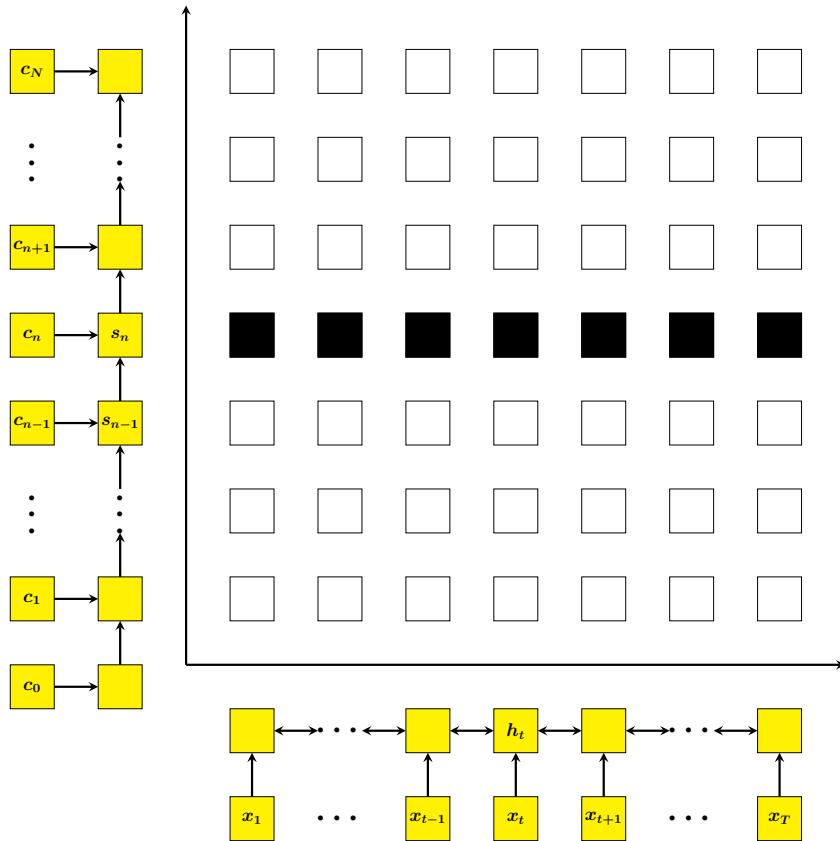
- difference:

summation over input positions inside or outside probability distributions

- **localization modelling:**
 - attention approach
 - HMM approach
- **dependencies:**
 - zero-order
 - first-order
- **present combinations:**
 - attention: zero-order
 - HMM: first-order
- ?? more combinations



Zero-Order HMM vs. Attention Mechanism



- **attention mechanism with attention weights $\alpha(t|n, \dots)$:**

$$p(c_n|c_{n-1}, s_{n-1}, h_1^T) = p(c_n|c_{n-1}, s_{n-1}, r_n)$$

with $r_n = \sum_t \alpha(t|n, s_{n-1}, h_1^T) \cdot h_t$

averaging: over internal representations h_t

- **zero-order HMM: mixture model**

$$p(c_n|c_{n-1}, s_{n-1}, h_1^T) =$$

$$= \sum_t p(t|n, s_{n-1}, h_1^T) \cdot p(c_n|t, c_{n-1}, s_{n-1}, h_1^T)$$

equivalence: $\alpha(t|n, \dots) = p(t|n, s_{n-1}, h_1^T)$
averaging: over probability models

- we consider the synchronized case (for simplicity):

$$p(c_1^N | x_1^N, \vartheta) := \dots$$

using a model with local or global re-normalization

- training in the spirit of *end-to-end*:
 - use final model as a whole for training
 - use gradient search/backpropagation for optimization
- open question: what is the adequate training criterion?

different variants of sequence discriminative training:

- sequence as a whole
- each symbol in the sequence context

details: next slides

- we consider a model $p(c_1^N|x_1^N, \vartheta)$
- two possible (pseudo-) Bayes decision rules (see Section 4.2):

minimum sequence error: $x_1^N \rightarrow \hat{c}_1^N(x_1^N) = \arg \max_{c_1^N} \{p(c_1^N|x_1^N, \vartheta)\}$

$$n = 1, \dots, N : \quad x_1^N \rightarrow \hat{c}_n(x_1^N) = \arg \max_{c_n} \left\{ \max_{\tilde{c}_1^N : \tilde{c}_n = c_n} p_n(\tilde{c}_1^N|x_1^N, \vartheta) \right\}$$

minimum symbol error: $x_1^N \rightarrow \hat{c}_n(x_1^N) = \arg \max_{c_n} \{p_n(c_n|x_1^N, \vartheta)\}$

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

- question: what are the *associated* natural(?) training criteria?

sequence as a whole for a model $p(c_1^N|x_1^N, \vartheta)$:

- (pseudo-) Bayes decision rule for minimum sequence error:

$$x_1^N \rightarrow \hat{c}_1^N(x_1^N) = \operatorname{argmax}_{c_1^N} \{p(c_1^N|x_1^N, \vartheta)\}$$

classification error: sequence error for all sequences

- **training:**

- training data: pairs of (input,output) strings

$$[c_1^N, x_1^N]_r \equiv [c_{r1}^{rN_r}, x_{r1}^{rN_r}], \quad r = 1, \dots, R$$

- training criterion:

$$\max_{\vartheta} \left\{ \sum_{r=1}^R \log p(c_{r1}^{rN_r}|x_{r1}^{rN_r}, \vartheta) \right\} = \max_{\vartheta} \left\{ \sum_{r=1}^R \sum_{n=1}^{N_r} \log p(c_{rn}|c_{r0}^{r,n-1}, x_{r1}^{rN_r}, \vartheta) \right\}$$

- **remarks on training criterion:**

- considered so far in this chapter
- virtually the only criterion used in today's ANN community
(i.e. for direct models with local re-normalization)
- more difficult for models with global re-normalization

individual symbols in the sequence context for a model $p(c_1^N | x_1^N, \vartheta)$:

- (pseudo-) Bayes decision rule for minimum symbol error:

$$n = 1, \dots, N : \quad x_1^N \rightarrow \hat{c}_n(x_1^N) = \operatorname{argmax}_c \{p_n(c|x_1^N, \vartheta)\}$$

classification error: symbol error in all positions and all sequences

- training:

- training data: pairs of (input,output) strings

$$[c_1^N, x_1^N]_r \equiv [c_{r1}^{rN_r}, x_{r1}^{rN_r}], \quad r = 1, \dots, R$$

- training criterion:

$$\max_{\vartheta} \left\{ \sum_{r=1}^R \sum_{n=1}^{N_r} \log p_n(c_{rn}|x_{r1}^{rN_r}, \vartheta) \right\} \quad \text{with} \quad p_n(c|x_1^N, \vartheta) := \sum_{c_1^N : c_n = c} p(c_1^N | x_1^N, \vartheta)$$

- remarks on training criterion:

- position-based symbol probability: efficient calculation?
- virtually unknown in ANN community
- somewhat related to Povey's MWE/MPE and *state-level minimum Bayes risk* [Povey & Woodland 02]
- practical problem/exercise: compute gradient for backpropagation

6.6 Conclusion

theoretical results:

- most general models: re-normalization at sequence level
- optimization problem: very hard
- pure log-linear models: convexity in some cases
- non-convexity caused by RNN structure and structures with hidden variables (like HMM)

work concerning training/optimization problem:

- local models with re-normalization (incl. attention):
 - standard case for ANN community
 - optimization: part of backpropagation
- global models with re-normalization:
intensively considered in ASR and POS tagging (CRF models)
 - history: Bahl et al. 1986, Normandin 1991, Valtchev 1996, Povey 2002 and 2016, ...
 - decomposition into several passes and lots of heuristics
 - outside ASR: only studied in CRF context for POS tagging
- training criterion: symbol error (as opposed to sequence error)
 - studied in ASR
 - virtually unknown outside ASR

7 Training Criterion: Bayes Decision Rule and Mismatch Conditions

7.1 Introduction

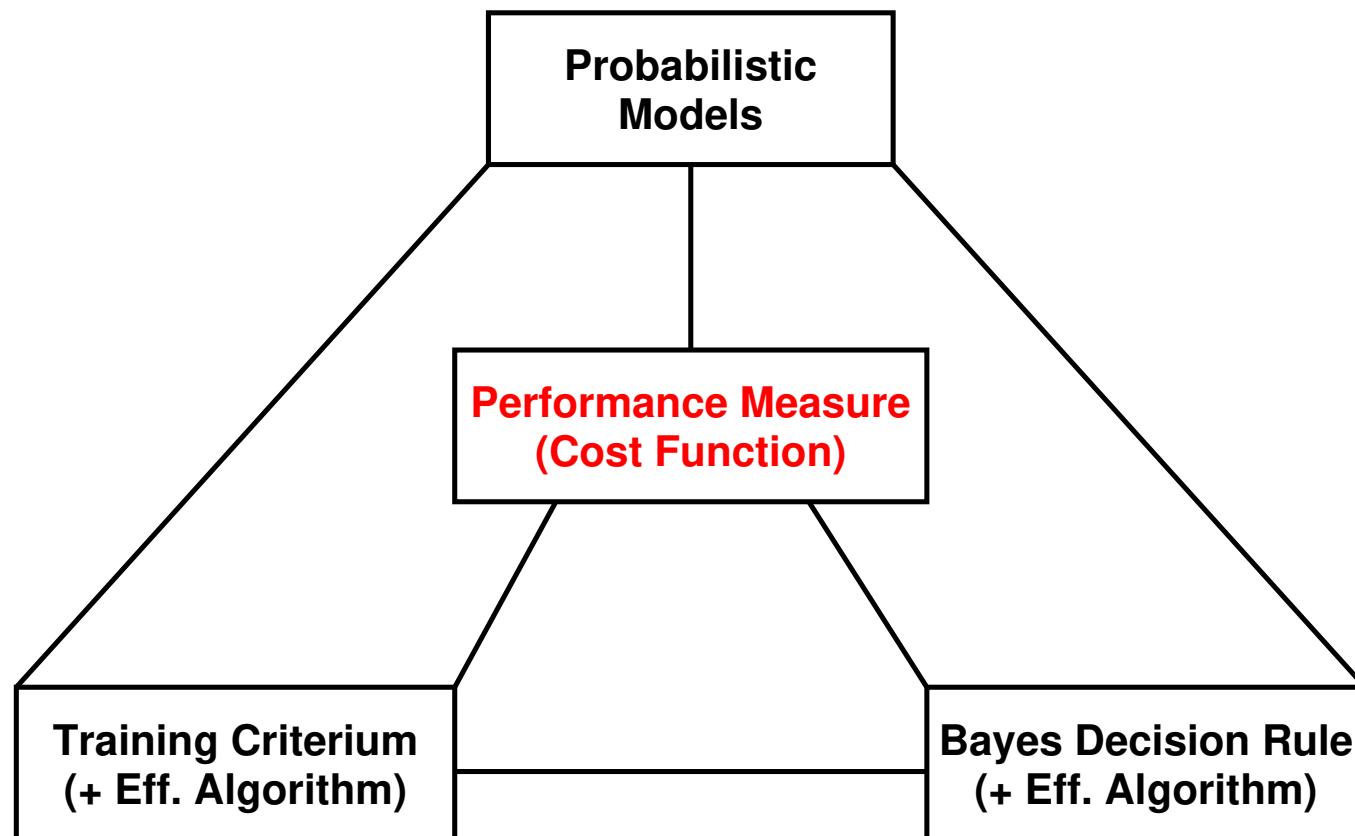
Bayes decision rule and mismatch conditions:

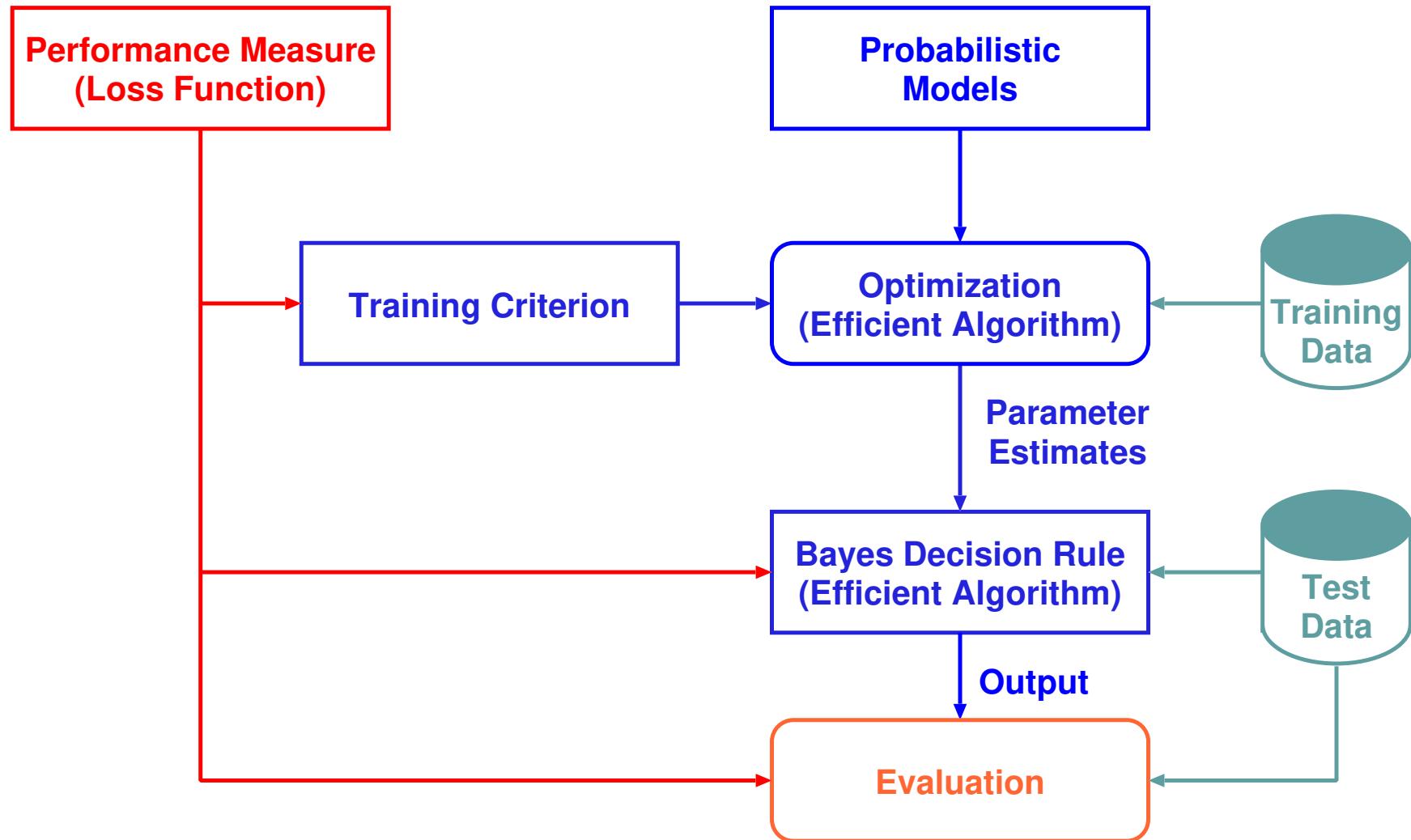
- optimality of Bayes decision rule:
proved only under the condition that we use the *true* distribution
- real world:
the true distribution is replaced by a *model* distribution,
whose parameters are learned from data.
- mismatch condition:
 - the model distribution is different from the true distribution
 - how does this mismatch effect the optimality of the Bayes decision rule?
 - 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:





emphasis in this chapter:

- 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

- bounds to Bayes classification error [Fukunaga 72, Duda & Hart⁺ 01]:
 - 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

conditions and aspects related to *optimal performance* according to Bayes decision rule

- Bayes decision rule: mathematically exact results for 0/1-loss function:
 - atomic case: symbol error
 - string (general case): sequence error
 - string with synchronization: symbol error in sequence context
- unknown true distribution:
 - mismatch conditions: true distribution $pr(c|x)$ vs. model distribution $p_\theta(c|x)$
 - question: do we lose 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

7.2 Probability Models and Decision Rule

model based decision rule $c_\vartheta(\cdot)$:

$$x \rightarrow c_\vartheta(x) := \arg \max_c \left\{ p_\vartheta(c, x) \right\}$$

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:

$$p_\vartheta(c|x) \in [0, 1] \quad \text{and} \quad \sum_c p_\vartheta(c|x) = 1$$

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

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

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

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

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

this example: the decision output does not change:

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

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:** $Y = \{(x_n, c_n) : n = 1, \dots, N\}$
- **prior distribution for λ**

formalism: compute predictive distribution

$$p(y|Y) = \int d\lambda p(y|\lambda) p(\lambda|Y) \quad p(\lambda|Y) = \frac{p(\lambda) p(Y|\lambda)}{\int d\lambda p(\lambda) p(Y|\lambda)}$$

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)**

7.3 Classification Errors: Two Types

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

decision rules:

$$\begin{aligned}\textbf{model: } c_\vartheta(x) &= \arg \max_c \{p_\vartheta(c, x)\} \\ \textbf{Bayes: } c_*(x) &= \arg \max_c \{pr(c|x)\}\end{aligned}$$

classification errors:

$$\begin{aligned}\textbf{model: } E_\vartheta &= \sum_x pr(x) \sum_c pr(c \neq c_\vartheta(x)|x) = \sum_x pr(x) [1 - pr(c_\vartheta(x)|x)] \\ \textbf{Bayes: } E_* &= \sum_x pr(x) \sum_c pr(c \neq c_*(x)|x) = \sum_x pr(x) [1 - pr(c_*(x)|x)] \\ &= \sum_x pr(x) [1 - \max_c pr(c|x)]\end{aligned}$$

consider the difference in classification error:

$$\begin{aligned}E_\vartheta - E_* &= \sum_x pr(x) [pr(c_*(x)|x) - pr(c_\vartheta(x)|x)] \\ &= \sum_x pr(x) [E_\vartheta(x) - E_*(x)]\end{aligned}$$

using the local classification errors $E_\vartheta(x)$ and $E_*(x)$ in point x

consider difference of local classification errors in point x :

$$E_\vartheta(x) - E_*(x) = pr(c_*(x)|x) - pr(c_\vartheta(x)|x)$$

use: $p_\vartheta(c, x) \leq p_\vartheta(c_\vartheta(x), x)$

$$\leq pr(c_*(x)|x) - pr(c_\vartheta(x)|x) + p_\vartheta(c_\vartheta(x), x) - p_\vartheta(c_*(x), x)$$

$$= [pr(c_*(x)|x) - p_\vartheta(c_*(x), x)] + [p_\vartheta(c_\vartheta(x), x) - pr(c_\vartheta(x)|x)]$$

apply: $u \leq |u|$

$$\leq |pr(c_*(x)|x) - p_\vartheta(c_*(x), x)| + |pr(c_\vartheta(x)|x) - p_\vartheta(c_\vartheta(x), x)|$$

$$\leq \sum_c |pr(c|x) - p_\vartheta(c, x)|$$

last step: all remaining classes are included to arrive at l_1 norm

l_1 bound:

$$E_\vartheta(x) - E_*(x) \leq \sum_c |pr(c|x) - p_\vartheta(c, x)|$$

l_∞ (or maximum) bound:

$$E_\vartheta(x) - E_*(x) \leq 2 \cdot \max_c \{ |pr(c|x) - p_\vartheta(c, x)| \}$$

l_2 bound (follows from l_∞):

$$E_\vartheta(x) - E_*(x) \leq 2 \cdot \sqrt{\sum_c [pr(c|x) - p_\vartheta(c, x)]^2}$$

note: these local bounds for x are TIGHT in the following sense:

if $p_\vartheta(c, x) \rightarrow pr(c x)$	then	Bound $\rightarrow 0$
	and	$E_\vartheta(x) \rightarrow E_*(x)$

7.4 From Global Bounds to Training Criteria

l_1 bound: consider the difference between the local error rates:

$$E_\vartheta(x) - E_*(x) \leq \sum_c |pr(c|x) - p_\vartheta(c, x)|$$

two operations for moving from local to global bounds

(specific case: from l_1 bound to Kullback-leibler divergence, see later):

- summation over x using $pr(x)$:

$$E_\vartheta - E_* \leq \sum_x pr(x) \sum_c |pr(c|x) - p_\vartheta(c, x)|$$

- squaring and using the variance inequality:

$$\begin{aligned} (E_\vartheta - E_*)^2 &\leq \left(\sum_x pr(x) \sum_c |pr(c|x) - p_\vartheta(c, x)| \right)^2 \\ &\leq \sum_x pr(x) \left(\sum_c |pr(c|x) - p_\vartheta(c, x)| \right)^2 \end{aligned}$$

l_2 bound: similar procedure

result: all global bounds apply to the SQUARE of the difference in classification error

7.4.1 Unconstrained Output: Squared Error Bound

identity for squared error (normalized p_c , arbitrary q_c !)
 [Patterson & Womack 66, Bourlard & Wellekens 89]:

$$\sum_c [p_c - q_c]^2 = \sum_c p_c \sum_{c'} [q_{c'} - \delta_{c'c}]^2 - \left(1 - \sum_c p_c^2\right)$$

re-write the l_2 bound (using variance inequality):

$$E_\vartheta(x) - E_*(x) \leq 2 \cdot \sqrt{\sum_c [pr(c|x) - p_\vartheta(c,x)]^2}$$

$$\begin{aligned} (E_\vartheta - E_*)^2 &\leq 4 \sum_x pr(x) \sum_c [pr(c|x) - p_\vartheta(c,x)]^2 \\ &= 4 \sum_x pr(x) \left(\sum_c pr(c|x) \sum_{c'} [p_\vartheta(c',x) - \delta(c',c)]^2 - [1 - \sum_c pr^2(c|x)] \right) \\ &= 4 \left(\sum_{x,c} pr(x,c) \sum_{c'} [p_\vartheta(c',x) - \delta(c',c)]^2 - \sum_x pr(x) [1 - \sum_c pr^2(c|x)] \right) \end{aligned}$$

practical relevance: we switched from true distribution to ideal targets

summary of re-writing:

$$(E_\vartheta - E_*)^2 \leq 4 \left(\sum_{x,c} pr(x,c) \sum_{c'} [p_\vartheta(c',x) - \delta(c',c)]^2 - \sum_x pr(x) \left[1 - \sum_c pr^2(c|x) \right] \right)$$

training criterion for minimum classification error: minimize the upper bound

only the left term of the bound depends on ϑ :

$$F(\vartheta) := \sum_{x,c} pr(c,x) \sum_{c'} [p_\vartheta(c',x) - \delta(c',c)]^2$$

which is the well-known squared error criterion.

For training data $(x_n, c_n), n = 1, \dots, N$, we use the empirical average:

$$\begin{aligned} \hat{\vartheta} &:= \arg \min_{\vartheta} \{ F(\vartheta) \} \\ &= \arg \min_{\vartheta} \left\{ \sum_{n=1}^N \sum_c [p_\vartheta(c, x_n) - \delta(c, c_n)]^2 \right\} \end{aligned}$$

we have derived the squared error criterion without using any normalization constraint or any other property of the model output

we consider the case of quadratic normalization

$$\sum_c p_\vartheta^2(c, x) = 1$$

and re-write the squared error criterion:

$$\begin{aligned} F(\vartheta) &= \sum_x pr(x) \sum_c [pr(c|x) - p_\vartheta(c, x)]^2 \\ &= 1 + \sum_x pr(x) \sum_c pr^2(c|x) - 2 \cdot \sum_x pr(x) \sum_c pr(c|x) p_\vartheta(c, x) \end{aligned}$$

global optimum for the ANN $p_\vartheta(c, x)$ as a whole (see Section 2.3):

$$\begin{aligned} \hat{p}_{\hat{\vartheta}}(c, x) &= \underset{\{p_\vartheta(c, x)\}}{\operatorname{argmin}} F(\vartheta) = \frac{pr(c|x)}{\sqrt{\sum_{c'} pr^2(c'|x)}} \\ \min_{\{p_\vartheta(c, x)\}} F(\vartheta) &\stackrel{?}{=} \sum_x pr(x) \cdot \left(1 - \sqrt{\sum_c pr^2(c|x)}\right)^2 \end{aligned}$$

result: a re-normalized class posterior

Squared Error Criterion: Model Substitution (unsuccessful attempt)

we replace the original model $p_\vartheta(c, x) \geq 0$
by a new model with square-root normalization:

$$p_\vartheta(c, x) := \sqrt{q_\vartheta(c, x)} \geq 0 \quad \sum_c \sqrt{q_\vartheta(c, x)} = 1$$

we consider the squared error criterion:

$$\begin{aligned} F(\vartheta) &= \sum_x pr(x) \sum_c \left[pr(c|x) - \sqrt{q_\vartheta(c, x)} \right]^2 \\ &= \sum_x pr(x) \sum_c (pr^2(c|x) + q_\vartheta(c, x)) - 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \sqrt{q_\vartheta(c, x)} \end{aligned}$$

global optimum for the ANN $p_\vartheta(c, x)$ as a whole (see Section 2.3):

$$\begin{aligned} \hat{q}_\vartheta(c, x) &= \operatorname{argmin}_{\{q_\vartheta(c, x)\}} F(\vartheta) = pr^2(c|x) \\ &\min_{\{p_\vartheta(c, x)\}} F(\vartheta) = 0 \end{aligned}$$

practical training criterion: ?
no, not different from conventional squared error criterion

7.4.2 Constrained Output: Binary Divergence

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:

$$\begin{aligned} (E_\vartheta - E_*)^2 &\leq 4 \sum_x pr(x) \sum_c [pr(c|x) - p_\vartheta(c, x)]^2 \\ &\leq 2 \sum_x pr(x) \sum_c \left(pr(c|x) \log \frac{pr(c|x)}{p_\vartheta(c, x)} + [1 - pr(c|x)] \log \frac{1 - pr(c|x)}{1 - p_\vartheta(c, x)} \right) \\ &= 2 \sum_x pr(x) \sum_c pr(c|x) \sum_{c'} \log \frac{1 - |\delta(c', c) - pr(c'|x)|}{1 - |\delta(c', c) - p_\vartheta(c', x)|} \\ &= 2 \sum_{x,c} pr(x, c) \sum_{c'} \log \frac{1 - |\delta(c', c) - pr(c'|x)|}{1 - |\delta(c', c) - p_\vartheta(c', x)|} \end{aligned}$$

Binary Divergence Bound

summary of re-writing:

$$(E_\vartheta - E_*)^2 \leq 2 \sum_{x,c} pr(x,c) \sum_{c'} \log \frac{1 - |\delta(c', c) - pr(c'|x)|}{1 - |\delta(c', c) - p_\vartheta(c', x)|}$$

training criterion for minimum classification error: minimize the upper bound

only the denominator of the argument of the logarithm depends on ϑ :

$$\begin{aligned} F(\vartheta) &= \sum_{x,c} pr(x,c) \sum_{c'} \log [1 - |\delta(c', c) - p_\vartheta(c', x)|] \\ &= \sum_{x,c} pr(x,c) \left(\log p_\vartheta(c, x) + \sum_{c' \neq c} \log [1 - p_\vartheta(c', x)] \right) \end{aligned}$$

which is the so-called binary cross-entropy criterion [Solla & Levin⁺ 88].

For training data $(x_n, c_n), n = 1, \dots, N$, we use the empirical average:

$$\begin{aligned} \hat{\vartheta} &:= \arg \max_{\vartheta} \{F(\vartheta)\} \\ &= \arg \max_{\vartheta} \left\{ \sum_n \left(\log p_\vartheta(c_n, x_n) + \sum_{c \neq c_n} \log [1 - p_\vartheta(c, x_n)] \right) \right\} \end{aligned}$$

7.4.3 Normalized Output: Kullback-Leibler Divergence Bound

Pinsker inequality for two distributions (normalized!) p_c and q_c
 [Fedotov & Harremoes⁺ 03], [Cover & Thomas 91, pp. 300]:

$$\left(\sum_c |p_c - q_c| \right)^2 \leq 2 \cdot \sum_c p_c \log \frac{p_c}{q_c}$$

right-hand side: the Kullback-Leibler divergence (or relative entropy)

re-write l_1 bound (using variance inequality):

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

summary of re-writing:

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

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 $(x_n, c_n), n = 1, \dots, N$, we use the empirical average:

$$\begin{aligned} \hat{\vartheta} &:= \arg \max_{\vartheta} \{ F(\vartheta) \} \\ &= \arg \max_{\vartheta} \left\{ \sum_n \log p_\vartheta(c_n|x_n) \right\} \end{aligned}$$

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

to avoid the logarithm and its singularity for $u = 0$, we use for $0 < r < 1$:

$$\log u \leq \frac{u^r - 1}{r} \quad \log u = \lim_{r \rightarrow 0} \frac{u^r - 1}{r}$$

we re-write the bound:

$$\begin{aligned} (E_\vartheta - E_*)^2 &\leq 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \log \frac{pr(c|x)}{p_\vartheta(c|x)} \\ &\leq \frac{2}{r} \cdot \sum_x pr(x) \sum_c pr(c|x) \left(\frac{pr^r(c|x)}{p_\vartheta^r(c|x)} - 1 \right) \end{aligned}$$

properties:

- tight bound and optimum solution: the same as with logarithm
- practical training criterion: no !!

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 divergence: 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
 - sequence discriminative training
 - splitting criteria in CART
 - ...

open issues:

- **linear bounds in E_θ :**
are there tight linear bounds rather than quadratic ones?
- **estimation problem:**
statistical fluctuations from training sample to test sample?
- **decisions in sequence context:**
e.g. in ASR, MT and NLP:
how to separate the contribution of each system component,
e. g. the language model?

questions and issues:

- mismatch condition (=this chapter):
 - general case: true distribution and model
 - special case: effect of class prior model only?
- what about bounds on Bayes classification error itself?
- Bayes-Bayes condition (in contrast to mismatch conditions):
 - two true distributions for comparing two conditions/systems
 - what is the difference of the classification errors?
- omitted/additional components in feature vectors:
how much does the classification error change?
- context: *sequence-to-sequence* processing
 - issue: string error vs. symbol error
 - how much does the context help to reduce the classification error?

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 $(E_\vartheta - E_*)^2$
- 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

two refinements:

- **from cross-entropy to 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**

7.5.1 From Discriminative to Generative Training

ASR (and other NLP) systems:

generative approach with an explicit language model

generative approach:

$$\text{true distribution: } pr(c, x) = pr(c) \cdot pr(x|c)$$

$$\text{model distribution: } q(c, x) = q(c) \cdot q(x|c)$$

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 $p_\theta(c, x)$

we compute the 'class posterior' model by re-normalization:

$$q(x) = \sum_c q(c, x)$$
$$q(c|x) = \frac{q(c, x)}{q(x)}$$

re-write (Kullback-Leibler) divergence bound:

$$\frac{1}{2} \cdot \Delta E_q^2 \leq \sum_{x,c} pr(c, x) \log \frac{pr(c|x)}{q(c|x)}$$

insert: $q(c|x) = q(c, x)/q(x)$

$$\begin{aligned} &= \underbrace{\sum_x pr(x) \log \frac{q(x)}{pr(x)}}_{\leq 0} + \underbrace{\sum_{x,c} pr(c, x) \log \frac{pr(c, x)}{q(c, x)}}_{\geq 0} \\ &\leq \sum_{x,c} pr(c, x) \log \frac{pr(c, x)}{q(c, x)} \\ &= \sum_c pr(c) \log \frac{pr(c)}{q(c)} + \sum_{x,c} pr(c, x) \log \frac{pr(x|c)}{q(x|c)} \end{aligned}$$

note: $q(x)$ depends on $q(c, x)$: $q(x) = \sum_c q(c, x)$

summary of re-writing the (Kullback-Leibler) divergence bound:

$$\begin{aligned} \frac{1}{2} \cdot \Delta E_q^2 &\leq \sum_{x,c} pr(c, x) \log \frac{pr(c|x)}{q(c|x)} \\ &\leq \sum_{x,c} pr(c, x) \log \frac{pr(c, x)}{q(c, x)} = \sum_c pr(c) \log \frac{pr(c)}{q(c)} + \sum_{x,c} pr(c, x) \log \frac{pr(x|c)}{q(x|c)} \end{aligned}$$

interpretations:

- first line: **divergence of the posterior distributions** → **cross-entropy training**
- second line: **divergence of joint distribution** → **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: from cross-entropy to maximum likelihood:

- **starting point: (Kullback-Leibler) divergence:**
 - we loosen the original bound
 - we move from class posterior probability to joint probability
- **resulting upper bound to the squared difference $(E_\vartheta - E_*)^2$ divergence in terms of joint probability**
- **training criterion: based on joint probability**
terminology: maximum likelihood
- **but: is always worse than the discriminative criterion**
which is based on the class posterior of the generative model

7.5.2 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!)
(after some forced alignment)
- 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 tool: 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 :

$$\begin{aligned}
 \frac{1}{2} \cdot \Delta E_{q,n}^2 &\leq \sum_{x_1^N} pr(x_1^N) \sum_c pr_n(c|x_1^N) \log \frac{pr_n(c|x_1^N)}{q_n(c|x_1^N)} \\
 &= \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N : c_n = c} pr(c_1^N|x_1^N) \log \frac{\sum_{c_1^N : c_n = c} pr(c_1^N|x_1^N)}{\sum_{c_1^N : c_n = c} q(c_1^N|x_1^N)} \\
 &\text{log-sum inequality: } \left(\sum_k a_k \right) \log \frac{\sum_k a_k}{\sum_k b_k} \leq \sum_k a_k \log \frac{a_k}{b_k} \\
 &\leq \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N : c_n = c} pr(c_1^N|x_1^N) \log \frac{pr(c_1^N|x_1^N)}{q(c_1^N|x_1^N)} \\
 &= \sum_{x_1^N} pr(x_1^N) \sum_{c_1^N} pr(c_1^N|x_1^N) \log \frac{pr(c_1^N|x_1^N)}{q(c_1^N|x_1^N)}
 \end{aligned}$$

important results:

- training at string level improves symbol level, too!
- training at symbol level is always better than at string level

7.6 Conclusion

- Bayes decision rule:
we have derived mathematically exact results in terms of upper bounds:
 - squared error criterion for unconstrained models
 - binary divergence for constrained models
 - (Kullback-Leibler) divergence for normalized models
- upper bounds result in practical training criteria:
squared error, binary cross-entropy, cross-entropy
- assumption: count of classification errors:
unifying concept in all cases: 0/1 loss function at suitable level
 - atomic case: single symbols or sequences as a whole
 - sequence with synchronization: symbol error in sequence context
- open question:
how to handle sequences without synchronization?
tentative approaches (??):
 - normalized positions axes for edit distance?
 - other type of inequalities (see next slides)

decision rules:

$$\text{model: } c_\vartheta(x) = \arg \min_{\tilde{c}} \left\{ \sum_c p_\vartheta(c|x) L[\tilde{c}, c] \right\}$$

$$\text{Bayes: } c_*(x) = \arg \min_{\tilde{c}} \left\{ \sum_c pr(c|x) L[\tilde{c}, c] \right\}$$

expectation of loss using empirical distribution:

$$\text{model: } L_\vartheta = \sum_x pr(x) \sum_c pr(c|x) L[c_\vartheta(x), c]$$

$$\text{Bayes: } L_* = \sum_x pr(x) \sum_c pr(c|x) L[c_*(x), c] = \sum_x pr(x) \min_{\tilde{c}} \left\{ \sum_c pr(c|x) L[\tilde{c}, c] \right\}$$

consider the difference in expected losses:

$$\begin{aligned} L_\vartheta - L_* &= \sum_x pr(x) \sum_c pr(c|x) (L[c_\vartheta(x), c] - L[c_*(x), c]) \\ &= \sum_x pr(x) [L_\vartheta(x) - L_*(x)] \end{aligned}$$

using the local losses $L_\vartheta(x)$ and $L_*(x)$ in point x

consider the difference in local losses:

$$\begin{aligned}
 L_\vartheta(x) - L_*(x) &= \sum_c pr(c|x) (L[c_\vartheta(x), c] - L[c_*(x), c]) \\
 &\leq \sum_c pr(c|x) (L[c_\vartheta(x), c] - L[c_*(x), c]) \\
 &\quad + \sum_c p_\vartheta(c|x) (L[c_*(x), c] - L[c_\vartheta(x), c]) \\
 &= \sum_c [pr(c|x) - p_\vartheta(c|x)] (L[c_\vartheta(x), c] - L[c_*(x), c]) \\
 &\leq \sum_c |pr(c|x) - p_\vartheta(c|x)| |L[c_\vartheta(x), c] - L[c_*(x), c]| \\
 &\quad \text{assumption: triangle inequality} \\
 &\leq L[c_\vartheta(x), c_*(x)] \sum_c |pr(c|x) - p_\vartheta(c|x)| \\
 &\leq I_x \cdot \sum_c |pr(c|x) - p_\vartheta(c|x)|
 \end{aligned}$$

with I_x : maximum number of output symbols for input string x

8 Training Criteria: Error Counting and Heuristic Approaches

terminology: what does the attribute *heuristic* mean?

- no explicit reference to Bayes framework
- no reference to optimum performance given by Bayes decision rule

we mainly consider the classification error in various types of classification problems:

- atomic case or single events: symbol error
- string as a whole: sequence error
- string with synchronization: symbol error in sequence context

unifying concept in all cases: 0/1 loss function at suitable level

decision rule using a normalized model $p_\vartheta(c|x)$ for input x and output c :

$$x \rightarrow c_\vartheta := \operatorname{argmax}_c p_\vartheta(c|x)$$

preview: general concepts for training:

- minimize classification error on training data
- smoothed counts: ideal requirements
- squared error criterion: used as smooth count
- most general case: strings with *model based expected loss*

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 → 0
 - strong rival class: error count → 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 $p_\vartheta(c, x)$:

$$p_\vartheta(c_n, x_n) \stackrel{?}{>} \max_{c \neq c_n} \{p_\vartheta(c, x_n)\}$$

$$p_\vartheta(c_n, x_n) \stackrel{?}{>} \left(\sum_{c \neq c_n} p_\vartheta^r(c, x_n) \right)^{1/r}$$

$$p_\vartheta^r(c_n, x_n) \stackrel{?}{>} \sum_{c \neq c_n} p_\vartheta^r(c, x_n)$$

$$\frac{p_\vartheta^r(c_n, x_n)}{\sum_{c \neq c_n} p_\vartheta^r(c, x_n)} \stackrel{?}{>} 1$$

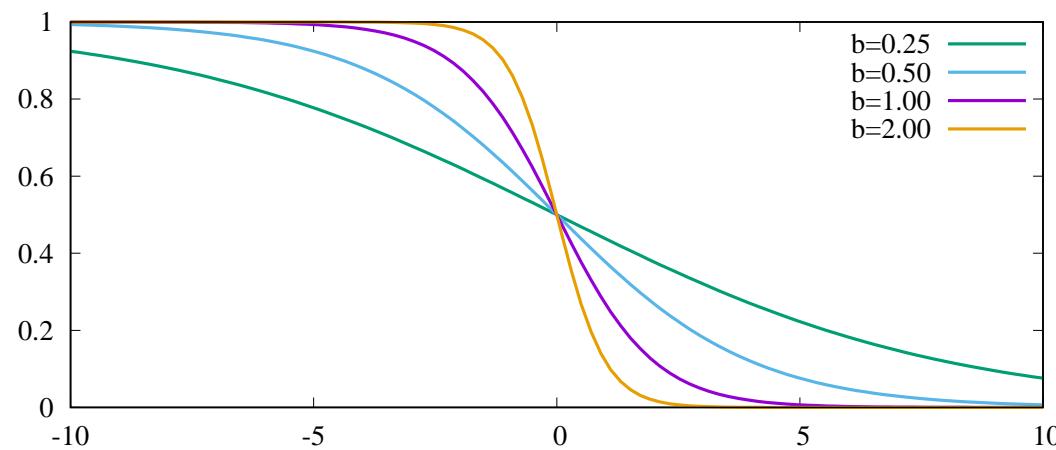
using this final comparison, we smooth the counts using an (inverted) sigmoid function:

$$F(\vartheta) = \frac{1}{N} \sum_{n=1}^N \frac{1}{1 + \left(\frac{p_\vartheta^r(c_n, x_n)}{\sum_{c \neq c_n} p_\vartheta^r(c, x_n)} \right)^\beta}$$

- parameter $r > 1$: controls the sharpness of the model distribution $p_\vartheta(c, x)$
- parameter $\beta > 0$: controls the smoothness of the counting function $h[\cdot]$:

$$u \rightarrow h[u] := 1/[1 + \exp(\beta \cdot \log u)] = 1/[1 + u^\beta]$$

Plots: Smoothing the Classification Error Count



8.2 Error Counting and Squared Error

assumption: normalized model:

$$p_\vartheta(c|x) \geq 0, \quad \sum_c p_\vartheta(c|x) = 1$$

squared error criterion for normalized (!! model:

$$F(\vartheta) = 1/N \cdot \sum_{n=1}^N \sum_c [p_\vartheta(c|x_n) - \delta(c, c_n)]^2$$

approach:

- **for each sample (c_n, x_n) , we interpret the numeric squared error**

$$\sum_c [p_\vartheta(c|x_n) - \delta(c, c_n)]^2$$

as a smoothed count of the classification error of model $p_\vartheta(c|x)$

- **we want to consider the squared error as a function of the posterior probability of the correct class: $p_\vartheta(c_n|x_n)$**
- **the squared error depends on all posterior probabilities of the model and therefore we compute lower and upper bounds**

we consider the contribution of each training sample (c_n, x_n) to the training criterion:

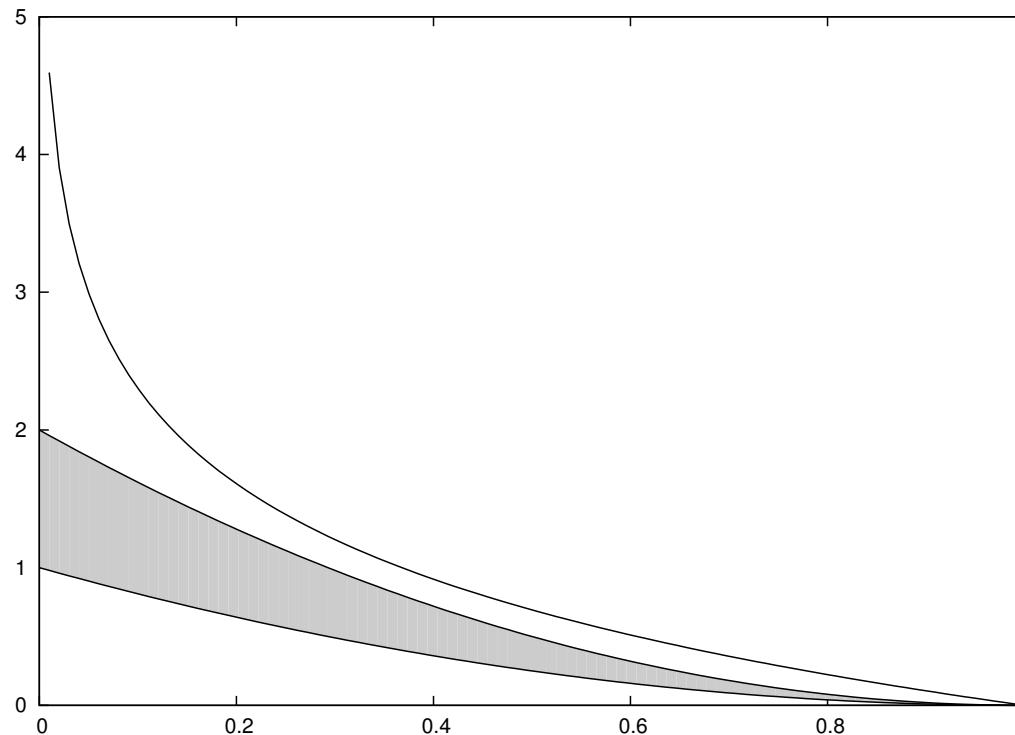
- **upper bound: is attained when the remaining probability mass $[1 - p_\vartheta(c_n|x_n)]$ is put on a single rival class $c \neq c_n$:**

$$\begin{aligned} \sum_c [p_\vartheta(c|x_n) - \delta(c, c_n)]^2 &\leq [p_\vartheta(c_n|x_n) - 1]^2 + [p_\vartheta(c_n|x_n) - 1]^2 \\ &= 2 \cdot [p_\vartheta(c_n|x_n) - 1]^2 \end{aligned}$$

- **lower bound: is attained when the remaining probability mass $[1 - p_\vartheta(c_n|x_n)]$ is uniformly distributed over all rival classes $c \neq c_n$:**

$$\begin{aligned} \sum_c [p_\vartheta(c|x_n) - \delta(c, c_n)]^2 &\geq [p_\vartheta(c_n|x_n) - 1]^2 + [C - 1] \cdot \left(\frac{p_\vartheta(c_n|x_n) - 1}{C - 1}\right)^2 \\ &= \frac{C}{C - 1} \cdot [p_\vartheta(c_n|x_n) - 1]^2 \\ &\geq [p_\vartheta(c_n|x_n) - 1]^2 \end{aligned}$$

- **result: factor 2 between upper and lower bound**

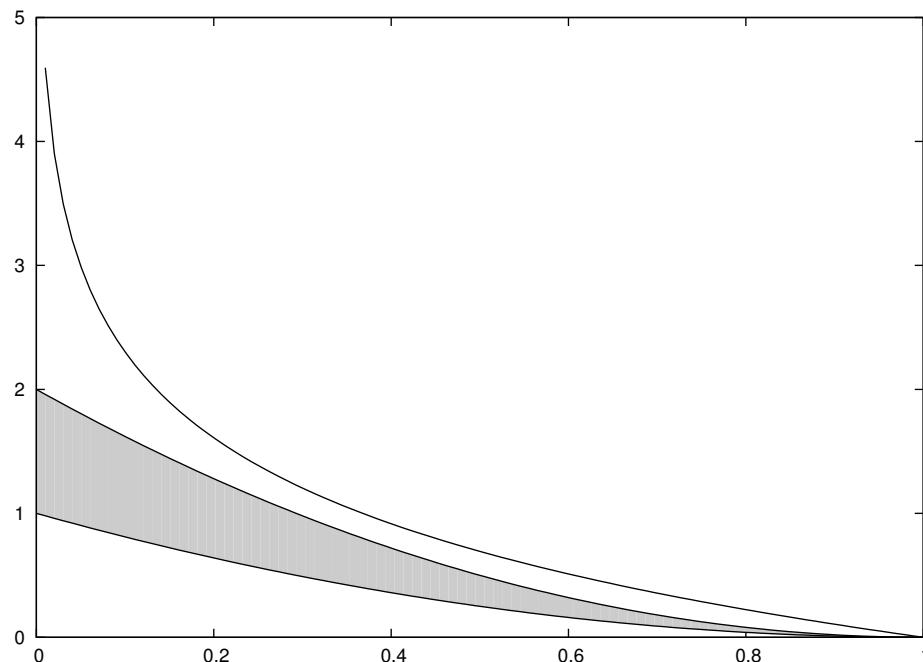


**plots of lower and upper bounds
of the squared error training criterion (along with negative logarithm)
as a function of $p_\vartheta(c_n|x_n)$, i.e. the model probability of the correct class.**

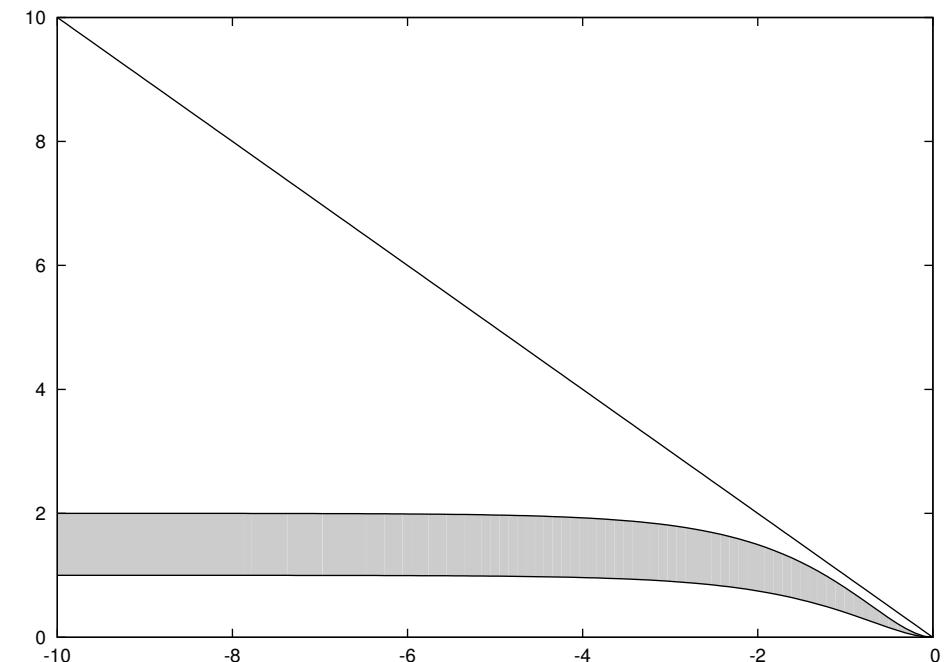
remarks about the squared error:

- squared error as a function of $p_\vartheta(c_n|x_n)$:
lower bound and upper bound
- the training criterion generates a 'smoothed' count of the classification errors:
 - smoothed count is between 1 (no strong rival class) and 2 (one single rival class)
 - smooth transition between totally correct and totally wrong decision
 - exact value of smoothed count: depends on *all* scores $p_\vartheta(c|x_n)$
- optimum solution: $\hat{p}_\vartheta(c|x) = pr(c|x)$
for 'unconstrained' model
- similar to MCE: minimum classification error,
for which however: $\hat{p}_\vartheta(c|x) \neq pr(c|x)$
- rarely used in practice
(reason: not known?)

**plots of squared error and cross-entropy
as a function of $p_\vartheta(c_n|x_n)$, i.e. the model probability of the correct class**



linear axes



logarithmic axes

**comparison of functional dependence
for cross-entropy and upper bound of squared error:**

- **interval** $p_\vartheta(c_n|x_n) < 0.2$:
 - **unlimited difference** for $p_\vartheta(c_n|x_n) \rightarrow 0$
 - **usual situation with logarithmic function**
- **interval** $p_\vartheta(c_n|x_n) \geq 0.2$:
 - **virtually identical shape** of the two plots
 - **maybe surprising result**

Squared Error and Cross-Entropy: Exercise in Chebyshev Approximation

exercise in mathematical optimization:

- **cubic approximation to the natural logarithm with $u \equiv p_\theta(c_n|x_n)$:**

$$\log u = [u - 1] - \frac{1}{2} [u - 1]^2 + \frac{1}{6} [u - 1]^3 - \dots$$

- **consider a quadratic approximation to the logarithm:**

$$\log u \cong \alpha + \beta \cdot [u - 1]^2$$

with parameters: offset α and scaling factor β

- **minimize the maximum of the absolute difference in the interval $[u_0, 1]$:**

$$\min_{\alpha, \beta} \left\{ \max_{u_0 < u < 1} \left| \alpha + \beta \cdot [u - 1]^2 - \log u \right| \right\}$$

- **result: min-max optimization problem (Chebyshev approximation)**

we will show that by using quadratic normalization the squared error criterion can be converted to a counting function.

quadratic normalization:

$$\sum_c p_\vartheta^2(c, x) = 1$$

we re-write the squared error criterion:

$$\begin{aligned} F(\vartheta) &= \sum_{c',x} pr(c',x) \sum_c [p_\vartheta(c,x) - \delta(c,c')]^2 \\ &= \sum_x pr(x) \left(1 - \sum_c pr^2(c|x) + \sum_c [pr(c|x) - p_\vartheta(c,x)]^2 \right) \\ &= 2 \cdot \sum_x pr(x) \left(1 - \sum_c pr(c|x) p_\vartheta(c,x) \right) = 2/N \cdot \sum_n [1 - p_\vartheta(c_n, x_n)] \end{aligned}$$

thus we have a linear counting function $u \rightarrow h[u]$:

$$u = p_\vartheta(c|x) \rightarrow h[u] = 1 - u$$

training criterion for parameter ϑ :

$$\begin{aligned}\hat{\vartheta} &= \underset{\vartheta}{\operatorname{argmin}} F(\vartheta) = \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_{c,x} pr(c, x) p_{\vartheta}(c, x) \right\} \\ &= \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_n p_{\vartheta}(c_n, x_n) \right\}\end{aligned}$$

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

$$\begin{aligned}\hat{p}_{\hat{\vartheta}}(c, x) &= \underset{\{p_{\vartheta}(c, x)\}}{\operatorname{argmin}} F(\vartheta) = \frac{pr(c|x)}{\sqrt{\sum_{c'} pr^2(c'|x)}} \\ \min_{\{p_{\vartheta}(c, x)\}} F(\vartheta) &= 2 \cdot \left(1 - \sum_x pr(x) \sqrt{\sum_c pr^2(c|x)} \right)\end{aligned}$$

result: a re-normalized class posterior

8.3 Error Counting and Consistency Requirement

considerations:

- definition: consistency requirement:
for a model with sufficient degrees of freedom, the optimum solution
is the class posterior distribution of the training data
- property/problem of traditional MCE (minimum classification error)
 - consistency requirement: not satisfied
 - proof: exercise
- tentative approach:
try to find a smoothing function so that the consistency requirement
is satisfied (exactly or approximatively)
- idea of consistency requirement and approach:
 - [Hampshire & Pearlmutter 90], [Bishop 95, pp. 245-247]
 - this presentation: slightly different, e.g. explicit normalization

8.3.1 Principle

consider a training sample (x_n, c_n) using non-negative model outputs $p_\vartheta(c, x)$ (without any normalization constraints!):

$$\begin{aligned} p_\vartheta(c_n, x_n) &\stackrel{?}{>} \max_{c \neq c_n} \left\{ p_\vartheta(c, x_n) \right\} \\ p_\vartheta(c_n, x_n) &\stackrel{?}{>} \left(\sum_{c \neq c_n} p_\vartheta^r(c, x_n) \right)^{1/r} \\ p_\vartheta^r(c_n, x_n) &\stackrel{?}{>} \sum_{c \neq c_n} p_\vartheta^r(c, x_n) \\ 2 p_\vartheta^r(c_n, x_n) &\stackrel{?}{>} \sum_c p_\vartheta^r(c, x_n) \\ \frac{p_\vartheta^r(c_n, x_n)}{\sum_c p_\vartheta^r(c, x_n)} &\stackrel{?}{>} \frac{1}{2} \end{aligned}$$

result: rival classes are absorbed by the re-normalization and disappear!

we define a new model with normalized outputs:

$$q_{\vartheta}(c|x) := \frac{p_{\vartheta}^r(c, x)}{\sum_{c'} p_{\vartheta}^r(c', x)}$$

training data: (c_n, x_n) , $n = 1, \dots,$

with the empirical distributions $pr(x, c), pr(x), pr(c|x)$, e. g.

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

classification error using a smooth counting function $h : u \rightarrow h[u]$:

$$\begin{aligned} F(\vartheta) &= \frac{1}{N} \cdot \sum_{n=1}^N h[q_{\vartheta}(c_n|x_n)] \\ &= \sum_{x,c} pr(x, c) \cdot h[q_{\vartheta}(c|x)] = \sum_x pr(x) \sum_c pr(c|x) \cdot h[q_{\vartheta}(c|x)] \end{aligned}$$

we fix a specific point x and include the normalization constraint:

$$\tilde{F}(\vartheta|x) = \sum_c pr(c|x) \cdot h[q_\vartheta(c|x)] - \lambda \left(\sum_c q_\vartheta(c|x) - 1 \right)$$

we determine the unknown counting function $u \rightarrow h[u]$:

- minimize the smoothed error count: derivative wrt $q_\vartheta(c|x)$
(justification: calculus of variations):

$$pr(c|x) \cdot h'[q_\vartheta(c|x)] - \lambda = 0$$

- consistency requirement:

$$\hat{q}_\vartheta(c|x) \stackrel{!}{=} pr(c|x)$$

- we combine the two equations with $u \equiv q_\vartheta(c|x)$
and obtain the differential equation:

$$u \cdot h'[u] - \lambda = 0$$

with the solution:

$$h[u] = \lambda \cdot \log u$$

set $\lambda := -1$ (justification: ?)

- optimum solution for the counting function $h[u]$ with $u = q_\vartheta(c_n|x_n)$ (equivalent to the cross-entropy criterion):

$$h[u] = -\log u$$

- properties of the solution:

$$\begin{aligned} u \rightarrow 1 : \quad h[u] &= 0 \\ u = e^{-i} : \quad h[u] &= i, \quad i = 1, 2, \dots \\ u \rightarrow 0 : \quad h[u] &\rightarrow +\infty \end{aligned}$$

- surprising result:

- for $u \rightarrow 0$, i. e. a 'severe' classification error,
the counting function $h[u]$ is not bounded.
- explanation:
what matters is each term as a whole in the sum of error counts:
each term: $-u \cdot \log u$, which is bounded for $u \rightarrow 0$

- additional remarks:

- parameter r , i. e. degree of smoothing, does not matter
- classification error: model error = Bayes error
(as it must be due to $q_\vartheta(c|x) = pr(c|x)$)

8.3.2 Power Approximation to Logarithm

training criterion: 'optimum' counting of classification errors:

$$F(\vartheta) := - \sum_n \log p_\vartheta(c_n|x_n)$$

approach:

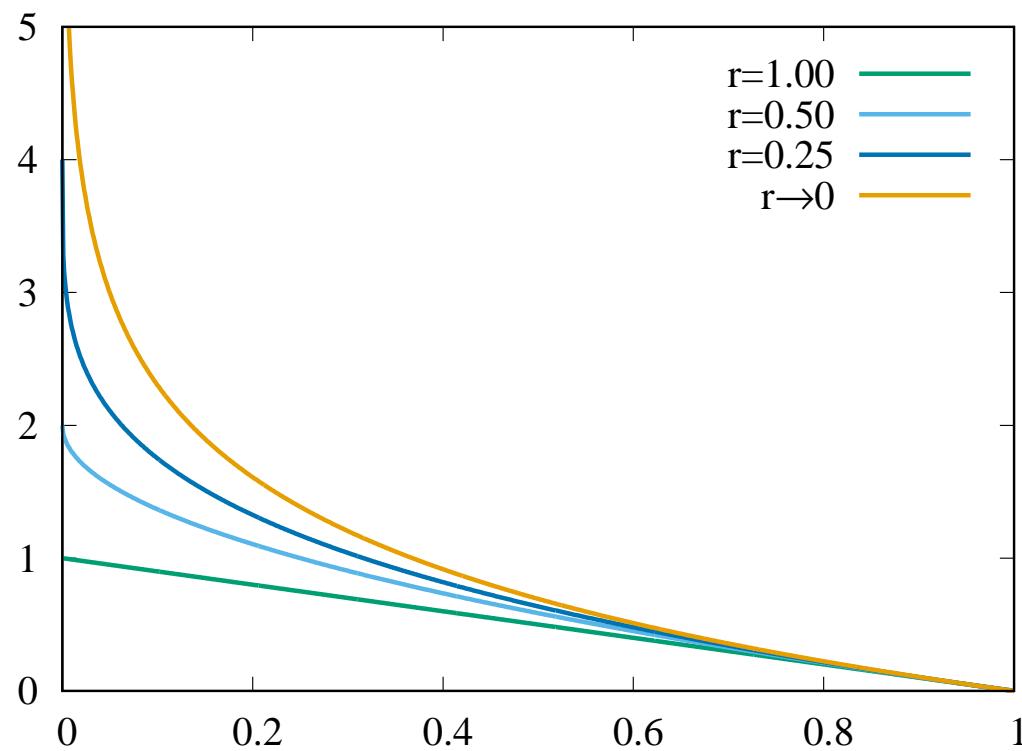
- due to $\log u \rightarrow -\infty$ for $u \rightarrow 0$,
small class posterior probabilities will result in an 'unlimited count'
- remedy: we approximate the counting function $h[u]$;
and define a modified criterion for some r with $0 < r < 1$
(see visualization):

$$\log u = \lim_{r \rightarrow 0} \frac{u^r - 1}{r} \quad \tilde{F}(\vartheta) := -\frac{1}{r} \sum_n [p_\vartheta^r(c_n|x_n) - 1]$$

- result: still a convex optimization problem
as a function of (normalized!) model $p_\vartheta(c|x)$!

Power Approximation to Logarithm

$$\log u = \lim_{r \rightarrow 0} \frac{u^r - 1}{r}$$



training criterion: power approximation to logarithm for $0 < r < 1$:

$$\tilde{F}(\vartheta) = -\frac{1}{r} \sum_n [p_\vartheta^r(c_n|x_n) - 1] = -\frac{1}{r} \sum_x pr(x) \sum_c pr(c|x) [p_\vartheta^r(c|x_n) - 1]$$

**optimum solution for (normalized!) model with sufficient degrees of freedom
(= monotonic transformation of $pr(c|x)$):**

$$\hat{p}_\vartheta(c|x) = \frac{pr^{1/(1-r)}(c|x)}{\sum_{c'} pr^{1/(1-r)}(c'|x)}$$

$$\begin{aligned} r \rightarrow 0 : \quad \hat{p}_\vartheta(c|x) &= pr(c|x) \\ r = 0.1/1.1 : \quad \hat{p}_\vartheta(c|x) &= pr^{1.1}(c|x) / \sum_{c'} pr^{1.1}(c'|x) \\ r = 1/2 : \quad \hat{p}_\vartheta(c|x) &= pr^2(c|x) / \sum_{c'} pr^2(c'|x) \\ r \rightarrow 1 : \quad \hat{p}_\vartheta(c|x) &= \delta(c, c_*(x)) \end{aligned}$$

proof:

- method of Lagrangian multipliers
- Hölder inequality

training criterion: power approximation to logarithm for $0 < r < 1$:

$$\begin{aligned}\tilde{F}(\vartheta) &= -\frac{1}{r} \sum_n [p_\vartheta^r(c_n|x_n) - 1] \\ \frac{\partial}{\partial \vartheta} \frac{u_\vartheta^r - 1}{r} &= u_\vartheta^r \cdot \frac{\partial}{\partial \vartheta} \log u_\vartheta\end{aligned}$$

consider the derivative (for gradient search):

$$\frac{\partial}{\partial \vartheta} \tilde{F}(\vartheta) = \dots = - \sum_n p_\vartheta^r(c_n|x_n) \cdot \frac{\vartheta}{\partial \vartheta} \log p_\vartheta(c_n|x_n)$$

$$\text{for } r \rightarrow 1 : \quad = - \sum_n p_\vartheta(c_n|x_n) \cdot \frac{\vartheta}{\partial \vartheta} \log p_\vartheta(c_n|x_n) = - \sum_n \frac{\vartheta}{\partial \vartheta} p_\vartheta(c_n|x_n)$$

$$\text{for } r \rightarrow 0 : \quad = - \sum_n \frac{\vartheta}{\partial \vartheta} \log p_\vartheta(c_n|x_n)$$

general case $r \in [0, 1]$: there is a weight $p_\vartheta(c_n|x_n)$ for each derivative of the logarithm

Comparison: Squared Error and Power Approximation to Logarithm

- power approximation with $r = 1/2$: model $p_\vartheta(c|x)$:

$$\begin{aligned} F(\vartheta) &= \sum_x pr(x) \sum_c pr(c|x) \cdot [1 - \sqrt{p_\vartheta(c|x)}] & \sum_c p_\vartheta(c|x) = 1 \\ &= 1/N \sum_n [1 - \sqrt{p_\vartheta(c_n|x_n)}] \end{aligned}$$

- squared error criterion with quadratic normalization: model $q_\vartheta(c|x)$:

$$\begin{aligned} F(\vartheta) &= \sum_{c,x} pr(c,x) \sum_{c'} [q_\vartheta(c'|x) - \delta(c',c)]^2 & \sum_c q_\vartheta^2(c|x) = 1 \\ &= \sum_x pr(x) \sum_c pr(c|x) \cdot [1 - q_\vartheta(c|x)] \\ &= 1/N \sum_n [1 - q_\vartheta(c_n|x_n)] \end{aligned}$$

- exact equivalence: $q_\vartheta(c|x) := \sqrt{p_\vartheta(c_n|x_n)}$
 generalization: from square root with $r = 1/2$ to any $r \in [0, 1]$

8.3.3 Trial: Relaxed Consistency Requirement

approach:

- starting point: non-negative model: $p_\vartheta(c|x) \geq 0$
- decision rule is invariant wrt monotonic transformations using a power transformation with $\alpha > 0$:

$$x \rightarrow c_\vartheta(x) = \operatorname{argmax}_\vartheta \{ p_\vartheta^\alpha(c|x) \} = \operatorname{argmax}_\vartheta \{ p_\vartheta(c|x) \}$$

- relaxed consistency requirement using the power transformation (with a re-normalization constant γ_x):

$$p_\vartheta(c|x) \stackrel{!}{=} \gamma_x \cdot pr^\alpha(c|x) \quad \sum_c p_\vartheta^{1/\alpha}(c|x) = 1$$

- unknown counting function $u \rightarrow h[u]$: minimize classification error:

$$\sum_{n=1}^N h h[q_\vartheta(c_n|x_n)] = \sum_{x,c} pr(x,c) \cdot h[q_\vartheta(c|x)] = \sum_x pr(x) \sum_c pr(c|x) \cdot h[p_\vartheta(c|x)]$$

Relaxed Consistency Requirement

we fix a specific point x and include the normalization constraint:

$$F(\vartheta|x) = \sum_c pr(c|x) \cdot h[p_\vartheta(c|x)] - \lambda_x \left(\sum_c p_\vartheta^{1/\alpha}(c|x) - 1 \right)$$

we determine the unknown counting function $u \rightarrow h[u]$:

- minimize the smoothed error count: derivative wrt $p_\vartheta(c|x)$
(justification: calculus of variations):

$$pr(c|x) \cdot h'[p_\vartheta(c|x)] - \lambda_x \cdot 1/\alpha \cdot p_\vartheta^{1/\alpha-1}(c|x) = 0$$

- consistency requirement:

$$p_\vartheta(c|x) \stackrel{!}{=} \gamma_x \cdot pr^\alpha(c|x)$$

- we combine the two equations with $u \equiv p_\vartheta(c|x)$
and obtain the differential equation:

$$\begin{aligned} u^{1/\alpha} \cdot h'[u] - \lambda \cdot u^{1/\alpha-1} &= 0 \\ h[u] &= \lambda \cdot u^{-1} \end{aligned}$$

- solution: $h[u] = -\log u$ (the same solution as before)
(unsuccessful attempt)

this section: we consider the most general case:

- input string x , output string c and model $p_\theta(c|x)$
- strings with and without synchronization
(sequence-to-sequence processing)
- general loss function $L[\tilde{c}, c]$: **structured output**

training method: ***expected loss based on a model***

- misleading terminology in literature: empirical or expected loss or Bayes risk
- not directly related to empirical distribution and its associated empirical loss/risk
- interpretation: extension of sequence discriminative training for a general loss function

remarks about ***model based expected loss*** in pseudo-Bayes decision rule:

key concept in Bayes framework:

- in testing: exact cost is typically simplified to MAP rule
- in training: how to use the exact cost ?

history:

- introduced in ASR: 'minimum Bayes risk'
 - Povey et al. 2002: minimum word error (MPW), minimum phone error (MPE)
 - Hain et al. 2006: state-label minimum Bayes risk (sMBR)
 - extended to MT: training based on expected TER and BLEU
- experimental results, mainly for ASR and edit distance:
most successful variant of sequence discriminative training
- other fields: reinforcement learning (?)

- we keep the decision rule of the Bayes framework using the *model based expected loss* for the minimization:

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

- new concept for training criterion: minimize the *model based expected loss* over ϑ on the training data $(c_n, x_n), n = 1, \dots, N$:

$$\min_\vartheta \left\{ \sum_n \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x_n) L[\tilde{c}, c_n] \right\} = \min_\vartheta \left\{ \sum_n \sum_{\tilde{c} \neq c_n} p_\vartheta(\tilde{c}|x_n) L[\tilde{c}, c_n] \right\}$$

- remarks on training criterion:
 - no explicit use of decision rule (like other approaches)
 - criterion: the same in correct decision rule and training
 - criterion tries to put all probability mass on correct output c_n
(note the metric property: $L[c, c] \equiv 0$)
 - expectation: = smoothing based on model $p_\vartheta(c|x)$
 - optimization: approximations and gradient search
 - possible criticism: ill-posed optimization problem (see later)

consider the the expected loss and decision rule for a training sample $(c, x) = (c_n, x_n)$:

$$L[\vartheta|x, c] := \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c] \quad x \rightarrow c_\vartheta(x) = \arg \min_c \{L[\vartheta|x, c]\}$$

analysis of dependence on generated output $c_\vartheta(x)$:

- condition: $c_\vartheta(x) \neq c$: $L[c_\vartheta(x), c] \neq 0$:

$$L[\vartheta|x, c] = p_\vartheta(c|x) \cdot 0 + p_\vartheta(c_\vartheta(x)|x) \cdot L[c_\vartheta(x), c] + \sum_{\tilde{c} \neq c, c_\vartheta(x)} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c]$$

- condition: $c_\vartheta(x) = c$: $L[c_\vartheta(x), c] = 0$:

$$L[\vartheta|x, c] = p_\vartheta(c_\vartheta(x)|x) \cdot 0 + \sum_{\tilde{c} \neq c} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c]$$

remarks:

- criterion tries to put all probability mass on correct output (due to $\sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) = 1$)
- no explicit dependence on decision output $c_\vartheta(x)$,
- note: expected loss on training data is not a practical measure of performance

- we consider the expected loss for a training sample $(c_n, x_n) = (c, x)$:

$$L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$$

- we re-write the 'baseline' loss $L[c_\vartheta(x), c]$ using the Kronecker delta $\delta(\tilde{c}, c)$

$$L[c_\vartheta(x), c] = \sum_{\tilde{c}} \delta(\tilde{c}, c_\vartheta(x)) L[\tilde{c}, c]$$

- interpretation:

$$p_\vartheta(\tilde{c}|x) \rightarrow \delta(\tilde{c}, c_\vartheta(x)) : \quad L[\vartheta|x, c] \rightarrow L[c_\vartheta(x), c]$$

expected loss $L[\vartheta|x, c]$ is a smooth approximation to $L[c_\vartheta(x), c]$:

- advantage for gradient search: it is differentiable
- upper bound: not necessarily

- **remark:** unlike $L[c_\vartheta(x), c]$,
 $L[\vartheta|x, c]$ does not have a practical interpretation in experiments.

consider the the expected loss and decision rule
for a training sample $(c, x) = (c_n, x_n)$:

$$L[\vartheta|x, c] := \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c] \quad x \rightarrow c_\vartheta(x) = \arg \min_c \{L[\vartheta|x, c]\}$$

consider the loss $L[\hat{c} = c_\vartheta(x), c]$ incurred by the desision rule $x \rightarrow c_\vartheta(x)$::

$$\begin{aligned} L[c_\vartheta(x), c] &\leq L[\tilde{c}, c_\vartheta(x)] + L[\tilde{c}, c] \quad \text{for any } \tilde{c} \\ L[c_\vartheta(x), c] &\leq \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c_\vartheta(x)] + \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c] \\ &\leq 2 \cdot \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \cdot L[\tilde{c}, c] \\ &= 2 \cdot L[\vartheta|x, c] \end{aligned}$$

result: $L[c_\vartheta(x), c] \leq 2 \cdot L[\vartheta|x, c]$

remarks:

- we used the triangle inequality, i. e. metric loss function
- the derivation does not involve the empirical distribution

8.4.2 Calculation of Expected Loss

expected loss:

$$L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$$

- key quantity in Bayes decision theory
- dependence on two arguments:
 - (optimum) string c : used in decision rule,
but usually approximated by MAP rule
 - (optimum) parameter ϑ : used in training
- explicit calculation
 - analytic calculation:
 - exact calculations, e. g. Hamming distance
 - approximations, e. g. edit distance after position normalization
 - efficient implementation: sum over all symbol strings:
 - exploit structure, e. g. n-order Markov chain
 - numeric approximation, e. g. by using symbol hypothesis lattice
 - explicit gradient (for gradient search in training)

Hamming distance for strings $(c, \tilde{c}) \equiv (c_1^I, \tilde{c}_1^I)$
(with synchronization or after artificial synchronization with $I = I_x$):

$$L[c, \tilde{c}] = L[c_1^I, \tilde{c}_1^I] = \sum_i [1 - \delta(c_i, \tilde{c}_i)]$$

- we compute the model based expected loss:

$$\begin{aligned} L[\vartheta|x, c] &= \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c] = \sum_i [1 - p_{\vartheta i}(c_i|x)] \\ p_{\vartheta i}(c_i|x) &= \sum_{\tilde{c}_1^I: c_i = \tilde{c}_i} p_\vartheta(\tilde{c}_1^I|x) \end{aligned}$$

- we use the expected loss to define decision rule:

$$\begin{aligned} x \rightarrow [c_\vartheta]_1^I(x) &= \arg \min_{c_1^I} \{L[\vartheta|x, c_1^I]\} = \arg \max_{c_1^I} \left\{ \sum_i p_{\vartheta i}(c_i|x) \right\} \\ x \rightarrow c_{\vartheta i}(x) &= \arg \max_{c_i} \{p_{\vartheta i}(c_i|x)\} \quad i = 1, \dots, I \end{aligned}$$

terminology: rule for minimum symbol error ('minimum Bayes risk')
(MAP approximation: rule for minimum string error)

- loss of decision rule in point (c, x) :

$$L[\hat{c} = c_\vartheta(x), c] = \sum_i [1 - \delta(c_{\vartheta i}(x), c_i)]$$

expected loss and training criterion for $(c, x) = (c_n, x_n)$:

$$L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$$

type of loss functions: based on counting the classification errors (i. e. no weights):

- **atomic case: symbol error and sequence error**

$$L[\vartheta|x, c] = 1 - p_\vartheta(c|x)$$

- **sequence: symbol error in sequence context with $c \equiv [c_i] \equiv [c_1^I]$:**
 - Hamming distance: sequence with synchronization:

$$L[\vartheta|x, c] = \sum_i [1 - p_{\vartheta i}(c_i|x)]$$

- edit distance: sequence without synchronization (see Section 4.3.4):
 (??) normalize output positions i using correct string c and ϵ symbols:

$$L[\vartheta|x, c] \cong \sum_i [1 - p_{\vartheta i}(c_i^\epsilon|x)]$$

overall result: (position-wise) symbol posterior probability

- will be used extensively in the following

training based on explicit form of expected loss:

$$L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$$

- optimization problem using gradient search
- gradient is required:
 - for closed-form solution (like Hamming distance)
 - for sum approximation for a general loss function

typical structure of model:

- **log-linear combination of models (without exponents):**

$$q_{\vartheta}(c_1^I; x) = q(c_1^I) \cdot \prod_i q_{\vartheta i}(c_i|x) = \prod_i [q(c_i|c_{i-2}^{i-1}) \cdot q_{\vartheta i}(c_i|x)]$$

- **re-normalization:**

$$p_{\vartheta}(c_1^I|x) = q_{\vartheta}(c_1^I, x) / \sum_{\tilde{c}_1^I} q_{\vartheta}(\tilde{c}_1^I, x)$$

- **(position-wise) symbol posterior probability:**

$$p_{\vartheta i}(c_i|x) = \sum_{\tilde{c}_1^I: c_i = \tilde{c}_i} p_{\vartheta}(\tilde{c}_1^I|x)$$

result: 'complicated' derivative

compute the gradient for three criteria:

- **logarithm of sequence posterior probability**
- **logarithm of symbol posterior probability**
- **expected loss with sum and general loss function**

remark: variants of sequence discriminative training

Sequence Posterior Probability: Gradient

sequence posterior probability for training sample $(c, x) = (c_n, x_n)$ with $c \equiv c_1^I$:

$$p_\vartheta(c_1^I|x) = \frac{q_\vartheta(c_1^I, x)}{\sum_{\tilde{c}_1^I} q_\vartheta(\tilde{c}_1^I, x)} \quad q_\vartheta(c_1^I, x) = q(c_1^I) \cdot \prod_i q_{\vartheta i}(c_i|x)$$

derivative of denominator:

$$\begin{aligned} \frac{\partial}{\partial \vartheta} \log \sum_{\tilde{c}_1^I} q_\vartheta(\tilde{c}_1^I, x) &= \sum_{\tilde{c}_1^I} p_\vartheta(\tilde{c}_1^I|x) \cdot \frac{\partial}{\partial \vartheta} \log q_\vartheta(\tilde{c}_1^I, x) \\ &= \sum_{\tilde{c}_1^I} p(\tilde{c}_1^I|x) \cdot \sum_i \frac{\partial}{\partial \vartheta} \log q_{\vartheta i}(\tilde{c}_i|x) = \sum_i \sum_{\tilde{c}_i} p_{\vartheta i}(\tilde{c}_i|x) \cdot \frac{\partial}{\partial \vartheta} \log q_{\vartheta i}(\tilde{c}_i|x) \end{aligned}$$

complete derivative:

$$\begin{aligned} \frac{\partial}{\partial \vartheta} \log p_\vartheta(c_1^I|x) &= \sum_i \left[\frac{\partial}{\partial \vartheta} \log q_{\vartheta i}(c_i|x) - \sum_{\tilde{c}_i} p_{\vartheta i}(\tilde{c}_i|x) \cdot \frac{\partial}{\partial \vartheta} \log q_{\vartheta i}(\tilde{c}_i|x) \right] \\ &= \sum_i \sum_{\tilde{c}_i} [\delta(c_i, \tilde{c}_i) - p_{\vartheta i}(\tilde{c}_i|x)] \cdot \frac{\partial}{\partial \vartheta} \log q_{\vartheta i}(\tilde{c}_i|x) \end{aligned}$$

remark: key role of weights: = symbol posterior probabilities $p_{\vartheta i}(\tilde{c}_i|x)$

symbol posterior probability for training sample $(c, x) = (c_n, x_n)$ with $c \equiv c_1^I$:

$$p_{\vartheta i}(c_i|x) = \frac{\sum_{\tilde{c}_1^I: \tilde{c}_i = c_i} q_{\vartheta}(\tilde{c}_1^I, x)}{\sum_{\tilde{c}_1^I} q_{\vartheta}(\tilde{c}_1^I, x)} \quad q_{\vartheta}(c_1^I, x) = q(c_1^I) \cdot \prod_i q_{\vartheta i}(c_i|x)$$

compute derivative (work out details):

$$\frac{\partial}{\partial \vartheta} \sum_i \log p_{\vartheta i}(c_i|x) = \dots ?$$

efficient implementation?

definition of symbol posterior probability:

$$p_t(a_t|x_1^T) = \sum_{\tilde{a}_1^T : \tilde{a}_t = a_t} p(\tilde{a}_1^T|x_1^T) = \frac{\sum_{\tilde{a}_1^T : \tilde{a}_t = a_t} q(\tilde{a}_1^T, x_1^T)}{\sum_{\tilde{a}_1^T} q(\tilde{a}_1^T, x_1^T)}$$

$$q(a_1^T, x_1^T) = q(a_1^T) \cdot \prod_t q_t(a_t|x_1^T, \vartheta)$$

complete derivative:

$$\begin{aligned} \sum_t \frac{\partial}{\partial \vartheta} \log p_t(a_t|x_1^T) &= \frac{\partial}{\partial \vartheta} \sum_t \left[\log \sum_{\tilde{a}_1^T : \tilde{a}_t = a_t} q(\tilde{a}_1^T, x_1^T) - \log \sum_{\tilde{a}_1^T} q(\tilde{a}_1^T, x_1^T) \right] \\ &= \sum_t \frac{\partial}{\partial \vartheta} \log \sum_{\tilde{a}_1^T : \tilde{a}_t = a_t} q(\tilde{a}_1^T, x_1^T) - T \cdot \frac{\partial}{\partial \vartheta} \log \sum_{\tilde{a}_1^T} q(\tilde{a}_1^T, x_1^T) \end{aligned}$$

expected loss for pair $(c, x) = (c_n, x_n)$:

$$L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$$

baseline model with normalization and its derivative:

$$\begin{aligned} p_\vartheta(c|x) &= q_\vartheta(c, x) / \sum_{\tilde{c}} q_\vartheta(\tilde{c}, x) \\ \frac{\partial}{\partial \vartheta} p_\vartheta(c|x) &= p_\vartheta(c|x) \cdot \frac{\partial}{\partial \vartheta} \log p_\vartheta(c|x) \\ &= p_\vartheta(c|x) \cdot \sum_{c'} [\delta(c', c) - p_\vartheta(c'|x)] \cdot \frac{\partial}{\partial \vartheta} \log q_\vartheta(c', x) \end{aligned}$$

gradient for expected loss:

$$\frac{\partial}{\partial \vartheta} L[\vartheta|x, c] = \dots = \sum_{\tilde{c}} (L[\tilde{c}, c] - L[\vartheta|c, x]) \cdot p_\vartheta(\tilde{c}|x) \cdot \frac{\partial}{\partial \vartheta} \log q_\vartheta(\tilde{c}, x)$$

remark: sum over all strings \tilde{c} :

requires efficient calculation or approximation

references: Heigold 2005, Cortez 2018

8.4.3 Empirical Averages for Three Types of Losses

literature: confusing terminology: *empirical* vs. *expected* loss

we distinguish:

- model distribution $p_\theta(c|x)$:
 - expected loss = **model based expectation of loss**
 - conditioned on x
- empirical or true distribution $pr(c, x)$:
 - empirical loss = expectation based on empirical distribution
 - empirical distribution is derived from training data $(c_n, x_n), n = 1, \dots, N$:

$$pr(c, x) := 1/N \cdot \sum_n \delta(c_n, c) \delta(x_n, x)$$

two variants:

- expectation using $pr(c|x)$ conditioned on x
- expectation using $pr(c, x)$: (global) empirical average

definition of expected loss for model $p_\vartheta(c|x)$ in point (c, x) :

$$L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$$

- empirical average of *model based expected loss*:

$$\begin{aligned} E_\vartheta[L] &= \sum_{c,x} pr(c, x) L[\vartheta|x, c] \\ &= \sum_{c,x} pr(c, x) \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c] = \frac{1}{N} \sum_n \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x_n) L[\tilde{c}, c_n] \end{aligned}$$

note: this average is computed *without* an explicit decision rule
 it is based on *three* expectations over x, c, \tilde{c}

- empirical average of loss incurred by model based decision rule:

$$\begin{aligned} x \rightarrow c_\vartheta(x) &= \arg \min_c \{L[\vartheta|x, c]\} \\ L_\vartheta &= \sum_{c,x} pr(c, x) L[c_\vartheta(x), c] = \frac{1}{N} \sum_n L[c_\vartheta(x_n), c_n] \end{aligned}$$

note: this average requires an explicit decision rule $x \rightarrow \hat{c}(x)$

overview:

- decision rules for model $p_\vartheta(c|x)$ and loss function $L[c, \tilde{c}]$:

$$\begin{aligned}\text{Bayes: } c_*(x) &= \arg \min_{\tilde{c}} \left\{ \sum_c p_r(c|x) L[\tilde{c}, c] \right\} \\ \text{model: } c_\vartheta(x) &= \arg \min_{\tilde{c}} \left\{ \sum_c p_\vartheta(c|x) L[\tilde{c}, c] \right\}\end{aligned}$$

- we consider three types of losses for a training sample $(c_n, x_n) = (c, x)$:

- loss of Bayes decision rule: $L[\hat{c} = c_*(x), c]$
- loss of model based decision rule: $L[\hat{c} = c_\vartheta(x), c]$
- *model based expectation loss over all classes*: $L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$

- expectation based on the empirical distribution $p_r(c, x)$
= empirical average over the training data (c_n, x_n) , $n = 1, \dots, N$:

$$\sum_{c,x} p_r(c, x) L[x, c] = 1/N \cdot \sum_n L[x_n, c_n]$$

where $L[x, c]$ denotes any of the above three losses

model based expected loss on training data:

$$\begin{aligned}
 E_\vartheta[L] &= \sum_x pr(x) \sum_c pr(c|x) \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c] \\
 &= \sum_x pr(x) \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \sum_c pr(c|x) L[\tilde{c}, c] \\
 &\geq \sum_x pr(x) \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \min_{c'} \left\{ \sum_c pr(c|x) L[c', c] \right\} \\
 &= \sum_x pr(x) \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) \sum_c pr(c|x) L[c_*(x), c] \\
 &= \sum_x pr(x) \sum_c pr(c|x) L[c_*(x), c] \\
 &= L_*
 \end{aligned}$$

remarks:

- derivation: requires properties beyond local aspects of (c, x)
- possible criticism: 'Bayes loss is always the minimum'
yes, but expected loss is NOT the loss of a decision rule

Expected Loss vs. Pseudo-Bayes Loss: An Unsuccessful Attempt

pseudo-Bayes loss: = loss of decision rule based on model distribution

model based expected loss on training data:

$$\begin{aligned} E_\vartheta[L] &= \sum_x pr(x) \sum_c pr(c|x) \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c] \\ &\geq \sum_x pr(x) \sum_c pr(c|x) \min_{c'} \left\{ \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c'] \right\} \\ &= \sum_x pr(x) \min_{c'} \left\{ \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c'] \right\} \\ &= \dots \text{ how to continue ?} \end{aligned}$$

remark: not useful?

inequality for the three losses:

$$L_* \leq L_\vartheta \leq 2 \cdot E_\vartheta[L]$$

$$L_* \leq E_\vartheta[L]$$

unknown inequality:

$$L_\vartheta \stackrel{?}{\leq} E_\vartheta[L]$$

open questions:

- are the inequalities tight?
- are there more interesting inequalities?
- training criterion: can the inequalities help?

specific situation:

- sequence with well defined positions
(maybe after position normalization)
- loss function: Hamming distance
- key role: symbol posterior probability in each position

two ways of normalizing the count of the classification errors:

- **per output sequence:**
 - theoretical framework used so far
 - problem: sequences have different lengths
 - rarely used as practical performance measure
- **per output symbol:**
 - usual way of reporting performance
 - better comparison across different databases
 - used beyond Hamming distance: edit distance (ASR) and TER (MT)

input sequence:	x_1	x_2	\dots	x_{i-1}	x_i	x_{i+1}	\dots	x_{I-1}	x_I
correct output sequence:	c_1	c_2	\dots	c_{i-1}	c_i	c_{i+1}	\dots	c_{I-1}	c_I
generated output sequence:	\tilde{c}_1	\tilde{c}_2	\dots	\tilde{c}_{i-1}	\tilde{c}_i	\tilde{c}_{i+1}	\dots	\tilde{c}_{I-1}	\tilde{c}_I

consider the case of synchronization with symbol-wise loss function:

loss per sequence:	$L[c_1^I, \tilde{c}_1^I] = \sum_{i=1}^I L[c_i, \tilde{c}_i] = \sum_{i=1}^I [1 - \delta(c_i, \tilde{c}_i)]$
decision rule per symbol:	$x_1^I \rightarrow c_{\vartheta i}(x_1^I) \quad i = 1, \dots, I$
associated loss per symbol:	$L[c_i, \tilde{c}_i = c_{\vartheta i}(x_1^I)]$

possible extensions beyond Hamming distance

(based on known correct output sequence):

- edit distance (ASR) and TER (MT)
- use correct sequence as 'seed' sequence
(details to be worked out)

empirical loss of model based decision rule:

$$\begin{aligned}
 L_\vartheta &= \sum_{x_1^I} pr(x_1^I) \sum_{c_1^I} pr(c_1^I|x_1^I) \sum_{i=1}^I L[c_{\vartheta i}(x_1^I), c_i] = \sum_{x_1^I} pr(x_1^I) \sum_{i=1}^I \sum_{c_1^I} pr(c_1^I|x_1^I) L[c_{\vartheta i}(x_1^I), c_i] \\
 &= \sum_{x_1^I} pr(x_1^I) \sum_{i=1}^I \sum_{c_i} pr_i(c_i|x_1^I) L[c_{\vartheta i}(x_1^I), c_i]
 \end{aligned}$$

resulting average loss:

- **loss per sequence, i. e. number of symbol errors per sequence**
- **what is needed for better comparison across databases:**
loss per correct output symbol with average length \bar{I} of output sequence:

$$L_\vartheta / \bar{I}$$

- **from Hamming distance to edit distance (in ASR) and TER (in MT):**
 - the same normalization concept works as well
 - two reasons:
 - loss is based on counts of symbol errors
 - correct symbol sequence is known!

decision rules for model $p_\vartheta(c|x)$ and Hamming distance $L[\tilde{c}, \tilde{c}]$ as loss function:

$$\text{Bayes: } c_*(x) = \operatorname{argmin}_{\tilde{c}} \left\{ \sum_c pr(c|x) L[\tilde{c}, c] \right\} = \operatorname{argmin}_{\tilde{c}_1^I} \left\{ \sum_i [1 - pr_i(c_i|x)] \right\}$$

$$\text{model: } c_\vartheta(x) = \operatorname{argmin}_{\tilde{c}} \left\{ \sum_c p_\vartheta(c|x) L[\tilde{c}, c] \right\} = \operatorname{argmin}_{\tilde{c}_1^I} \left\{ \sum_i [1 - p_{\vartheta i}(c_i|x)] \right\}$$

- **loss of Bayes decision rule:** $L[\hat{c} = c_*(x), c]$

$$L_* = \sum_{c_1^I, x} pr(c_1^I, x) \sum_i [1 - \delta(c_{*i}(x), c_i)] = 1/N \cdot \sum_n \sum_i [1 - \delta(c_{*i}(x_n), c_{ni})]$$

- **loss of model based decision rule:** $L[\hat{c} = c_\vartheta(x), c]$

$$L_\vartheta = \sum_{c_1^I, x} pr(c_1^I, x) \sum_i [1 - \delta(c_{\vartheta i}(x), c_i)] = 1/N \cdot \sum_n \sum_i [1 - \delta(c_{\vartheta i}(x_n), c_{ni})]$$

- **model based expectation loss over all classes:** $L[\vartheta|x, c] = \sum_{\tilde{c}} p_\vartheta(\tilde{c}|x) L[\tilde{c}, c]$

$$E_\vartheta[L] = \sum_{c_1^I, x} pr(c_1^I, x) \sum_i [1 - p_{\vartheta i}(c_i|x)] = 1/N \cdot \sum_n \sum_i [1 - p_{\vartheta i}(c_{ni}|x_n)]$$

exercise: verify the previous inequalities for these losses

8.4.4 Training Criteria: Analysis and Comparison

this section: unifying view of training criteria

important aspects:

- what is the relation of the training criterion to the performance?
- type of optimization problem: well defined vs. ill-posed singular solutions?

concepts for training criteria:

- upper bound framework in Bayesian decision theory:
 - tight upper bound on performance
 - can be combined with exact loss function
 - consistency guaranteee: model → true posterior
- smoothed error counts: *minimum classification error (MCE)*
 - pragmatic concept, no optimality guaranteee
 - can be combined with exact loss function
- model based expected loss:
 - identical to exact form of pseudo-Bayes decision rule
 - consistency requirements? upper bound?



Training Criteria: Overview

sequence posterior probability with $(c, x) = (c_n, x_n)$ with $c \equiv c_1^I$:

$$p_\vartheta(c_1^I|x) = \frac{q_\vartheta(c_1^I, x)}{\sum_{\tilde{c}_1^I} q_\vartheta(\tilde{c}_1^I, x)}$$
$$q_\vartheta(c_1^I, x) = q(c_1^I) \cdot \prod_i q_{\vartheta i}(c_i|x)$$

training strategy: a two-step approach:

- training of individual components:
 - observation model (OM): $q_{\vartheta i}(c_i|x)$
(position-wise cross-entropy)
 - language model (LM): $q(c_1^I)$
(minimum perplexity)
- sequence discriminative training: includes the LM when training the OM
 - re-writing of expected loss:
 - sequence as a whole
 - symbol in sequence context
 - approximative sum for expected loss
 - ...
 - functional form of training criterion:
 - logarithmic
 - linear
 - squared error
 - ...

in addition: gradient search strategy

Training Criteria: Synopsis Re-normalization of Loss

- **input-output sequences (x_1^I, c_1^I) with synchronization and Hamming distance as loss function:
closed-form solution using the position-wise symbol posterior distribution:**

$$x = x_1^I \rightarrow c_{\vartheta i}(x_1^I) = \operatorname{argmin}_{c_i} \{ p_{\vartheta i}(c_i | x_1^I) \} = \operatorname{argmin}_{c_i} \left\{ \sum_{\tilde{c}_1^I : c_i = \tilde{c}_i} p_{\vartheta i}(c_i | x_1^I) \right\}$$

- **re-interpretation of loss function: loss per output symbol c_i (or position):**

$$L[c_{\vartheta i}(x_1^I), c_i]$$

- **average loss per output symbol c_i (used on next slides):**

$$\begin{aligned} L_\vartheta &:= \frac{1}{\bar{I}} \cdot \sum_{x_1^I} pr(x_1^I) \sum_{i=1}^I \sum_{c_i} pr_i(c_i | x_1^I) L[c_{\vartheta i}(x_1^I), c_i] \\ \bar{I} &:= \sum_{x_1^I} pr(x_1^I) \cdot I \end{aligned}$$

- **goal of this style of global re-normalization:
all output symbols get equal weights (unlike sequence-specific re-normalization)**

Training Criteria: Synopsis

- upper bound framework: Kullback-Leibler divergence:

$$1/2 \cdot [L_\vartheta - L_*]^2 \leq 1/\bar{I} \cdot \sum_x pr(x) \sum_i \sum_{c_i} pr_i(c_i|x) \log \frac{pr_i(c_i|x)}{p_{\vartheta i}(c_i|x)}$$

optimum solution: $p_{\vartheta i}(c_i|x) \rightarrow pr_i(c_i|x)$ with $L_\vartheta \rightarrow L_*$

- upper bound framework: squared error:

$$1/4 \cdot [L_\vartheta - L_*]^2 \leq 1/\bar{I} \cdot \sum_x pr(x) \sum_i \sum_{c_i} [pr_i(c_i|x) - p_{\vartheta i}(c_i, x)]^2$$

optimum solution: $p_{\vartheta i}(c_i, x) \rightarrow pr_i(c_i|x)$ with $L_\vartheta \rightarrow L_*$

- smoothed error count ($0 < r < 1, r \rightarrow 0$):

$$\begin{aligned} L_\vartheta &\cong 1/\bar{I} \cdot \sum_x pr(x) \sum_i \sum_{c_i} pr_i(c_i|x) \cdot h[p_{\vartheta i}(c_i|x)] \\ &= 1/\bar{I} \cdot \sum_x pr(x) \sum_i \sum_{c_i} pr_i(c_i|x) \cdot [1 - p_{\vartheta i}^r(c_i|x)] \end{aligned}$$

optimum solution: $p_{\vartheta i}(c_i|x) \rightarrow \gamma_x \cdot pr_i^{1/(1-r)}(c_i|x)$ with $L_\vartheta \rightarrow ?$

- model based expectation loss:

$$E_\vartheta[L] = 1/\bar{I} \cdot \sum_x pr(x) \sum_i \sum_{c_i} pr_i(c_i|x) \cdot [1 - p_{\vartheta i}(c_i|x)]$$

optimum solution: $p_{\vartheta i}(c_i|x) \rightarrow \delta(c_i, c_{*i}(x))$ with $E_\vartheta[L] \rightarrow L_*$

Training Criteria: Synopsis

- upper bound framework: Kullback-Leibler divergence:

$$1/2 \cdot [L_\vartheta - L_*]^2 \leq \frac{1}{N \bar{I}} \cdot \sum_n \sum_i \log \frac{pr_i(c_{ni}|x_n)}{p_{\vartheta i}(c_{ni}|x_n)}$$

optimum solution: $p_{\vartheta i}(c_i|x) \rightarrow pr_i(c_i|x)$ with $L_\vartheta \rightarrow L_*$

- upper bound framework: squared error with quadratic normalization: $\sum_c p_{\vartheta i}^2(c_i; x) = 1$

$$1/4 \cdot ([L_\vartheta - L_*]^2 - \Delta_{Gini}) \leq \frac{2}{N \bar{I}} \cdot \sum_n \sum_i [1 - p_{\vartheta i}(c_{ni}; x_n)]$$

optimum solution: $p_{\vartheta i}(c_i; x) \rightarrow \gamma_x \cdot pr_i(c_i|x)$ with $L_\vartheta \rightarrow L_*$ (indirectly)

- smoothed error count ($< r < 1, r \rightarrow 0$):

$$L_\vartheta \cong \frac{1}{N \bar{I}} \cdot \sum_n \sum_i h[p_{\vartheta i}(c_{ni}|x_n)] = \frac{1}{N \bar{I}} \cdot \sum_n \sum_i [1 - p_{\vartheta i}^r(c_{ni}|x_n)]$$

optimum solution: $p_{\vartheta i}(c_i|x) \rightarrow \gamma_x \cdot pr_i^{1/(1-r)}(c_i|x)$ with $L_\vartheta \rightarrow ?$

- model based expectation loss:

$$E_\vartheta[L] = \frac{1}{N \bar{I}} \cdot \sum_n \sum_i [1 - p_{\vartheta i}(c_{ni}|x_n)]$$

optimum solution: $p_{\vartheta i}(c_i|x) \rightarrow \delta(c_i, c_{*i}(x))$ with $E_\vartheta[L] \rightarrow L_*$

remarks on upper bound framework: $[L_\vartheta - L_*]$

- squared performance difference
- from sequences to symbols: derivation has to be re-verified

**possible discrepancy between theoretical concepts and experimental results:
what we measure depends on many details:**

- approximation for sum over sequences:
symbol hypothesis lattice, simplified language model, ...
- exact form of optimization criterion:
e. g. logarithmic vs. linear dependence
- gradient search:
 - o problems with local optima
 - o speed of convergence

Comparison: Cross-Entropy and Expected Loss Logarithmic vs. Direct Symbol Posterior Probability

general loss function for input/output sequences (c, x) with $c = c_1^I$:

- pseudo-Bayes decision rule using expected loss:

$$x \rightarrow \hat{c}_\vartheta(x) = \operatorname{argmax}_{c_1^I} \left\{ \sum_i p_{\vartheta i}(c_i|x) \right\} = \operatorname{argmax}_{c_1^I} \left\{ \sum_i \log p_{\vartheta i}(c_i|x) \right\}$$

result: no difference caused by logarithm

- training criterion for count-based classification errors

(training data: pairs of input-output sequences (c_n, x_n) with $c_n \equiv [c_{n1}^{nI_n}]$):

- model based expected loss:

$$\hat{\vartheta} = \operatorname{argmax}_{\vartheta} \left\{ \sum_n \sum_i p_{\vartheta i}(c_{ni}|x_n) \right\}$$

- cross-entropy: based upon Bayes framework with mismatch conditions:

$$\hat{\vartheta} = \operatorname{argmax}_{\vartheta} \left\{ \sum_n \sum_i \log p_{\vartheta i}(c_{ni}|x_n) \right\}$$

result: evident difference due to logarithm

**training criteria and strategies
(around Hamming distance and extensions):**

- **sequence posterior:**
 - OK for (pseudo) Bayes decision
 - not OK for training → symbol posterior explanation?
- **sequence posterior:**
provides an upper bound, but symbol posterior is better (see Section 7.5.2, p. 261)
- **criterion vs. gradient search:**
what is more relevant in practice?
- **symbol posterior: which type of symbols?**
 - CART labels vs. phonemes/letters vs. words?
 - should we modify the *practical* evaluation metric?
- **frame-level criterion (see section 4.3.6)**

ideas for research topics (mainly ASR):

- which level of symbols?
state labels vs. phonemes vs. ...
- symbol posterior probability:
 - logarithmic vs. linear dependence
 - cross-entropy vs. squared error criterion
- approximation to sum:
symbol hypothesis lattice vs. simplified language model
- gradient search:
explicit control of weights (=symbol posterior)
- symbol positions (for edit distance):
 - input sequence positions vs. output sequence positions
 - related issue: *direct* or *segmental* HMM (p. 210)
- direct sequence posterior modelling (Section 6.4, p. 204):
how to compute symbol posterior efficiently?
- extensions beyond ASR:
 - natural language understanding: syntactic/semantic tagging (CRF)
 - machine translation: symbol posterior for TER (?)

characteristic properties of ASR:

- **several levels:**
10-ms frames, phonemes/sounds, words, sentences
- **synchronization/alignment:** key problem in ASR
best solution today:
 - hybrid HMM (or simplification: CTC)
 - novel concept: attention mechanism
- **training criterion:**
 - cross-entropy at frame level
 - cross-entropy at sentence level (MMI):
sequence discriminative training

9.1 System Architecture for ASR

9.1.1 Statistical Approach

starting points:

- very complex problem:
no perfect knowledge of the dependencies in speech and language:
 - different from conventional computer science
 - like a problem in natural sciences (e.g. approximative models in physics)
- perfect solution will be difficult:
 - we accept that the system will make errors
 - but we try to find the best compromise
- fairly general view:
 - input sequence (ASR: sequence over time t : $X := x_1 \dots x_t \dots x_T$)
 - output sequence: $W := w_1 \dots w_n \dots w_N$ of unknown length N
- we need a generation mechanism:

$$X \rightarrow W = \hat{W}(X)$$

- to this purpose, we assume a
 - posterior distribution $pr(W|X)$
 - which can be extremely complex: both arguments are strings!

performance measure or loss function (e. g. edit or Levenshtein distance) between true output sequence W and hypothesized output sequence \tilde{W} :

$$L[W, \tilde{W}]$$

Bayes decision rule minimizes expected loss:

$$X \rightarrow W = \hat{W}(X) := \arg \min_{\tilde{W}} \left\{ \sum_W pr(W|X) \cdot L[W, \tilde{W}] \right\}$$

Under these two conditions:

$$\begin{aligned} L[W, \tilde{W}] : & \quad \text{satisfies triangle inequality} \\ \max_W \{pr(W|X)\} > 0.5 \end{aligned}$$

we have [Schlüter & Nussbaum⁺ 12]:

$$X \rightarrow W = \hat{W}(X) := \arg \max_W \left\{ pr(W|X) \right\}$$

Since [Bahl & Jelinek⁺ 83], this simplified Bayes decision rule is widely used for speech recognition, handwriting recognition, machine translation, ...

Bayes decision rule:

$$X \rightarrow W = \hat{W}(X) := \arg \min_{\tilde{W}} \left\{ \sum_W pr(W|X) \cdot L[W, \tilde{W}] \right\}$$

practical considerations:

- **unknown distribution $pr(W|X)$:**
remedy: replace true $pr(W|X)$ by a model $p(W|X)$ and learn its free parameters from a HUGE set of examples
- **important problem:**
 - compositional modelling for $p(W|X)$ is needed since W and X are strings
 - units smaller than the whole sequence are needed
(e.g. phrases/word groups, words, letters)
- **principal modelling concepts:**
 - direct approach by factorizing over $W = w_1^N$: $p(w_1^N|X) = \prod_n p(w_n|w_0^{n-1}, X)$
 - separate language model and use generative model $p(W)$:

$$p(W|X) = p(W) \cdot p(X|W) / \sum_{W'} [p(W') \cdot p(X|W')]$$

extension: log-linear modelling

training: approximation by maximum likelihood

Problem in Bayes decision rule:

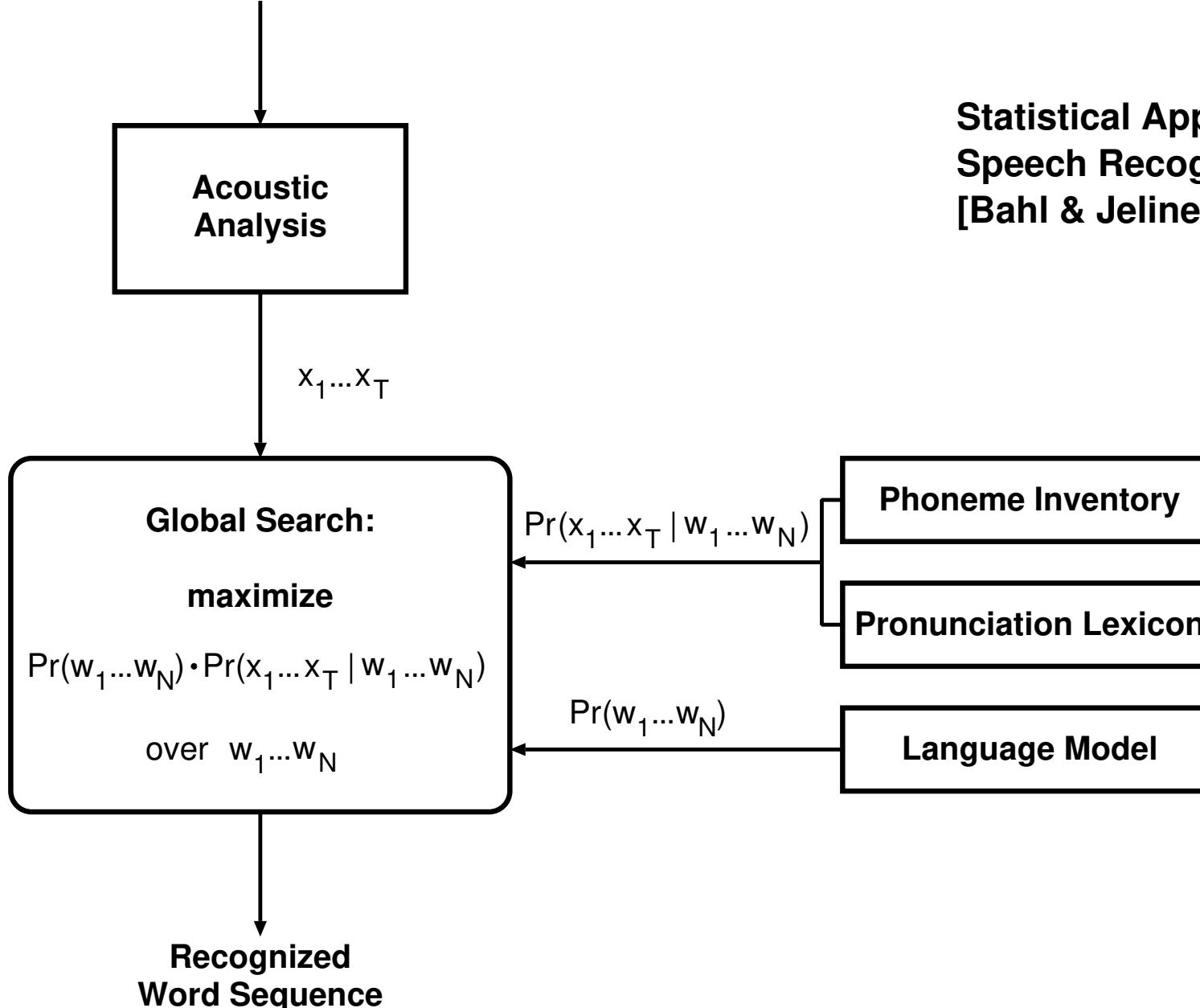
- true posterior distribution: unknown
- to replace it, assume suitable model distributions with free parameters:

$$p(W|X) = \frac{p(W) \cdot p(X|W)}{\sum_{\tilde{W}} p(\tilde{W}) \cdot p(X|\tilde{W})}$$

- generative model: language model $p(W)$ and acoustic model $p(X|W)$

structure of spoken language implies a hierarchy of levels:

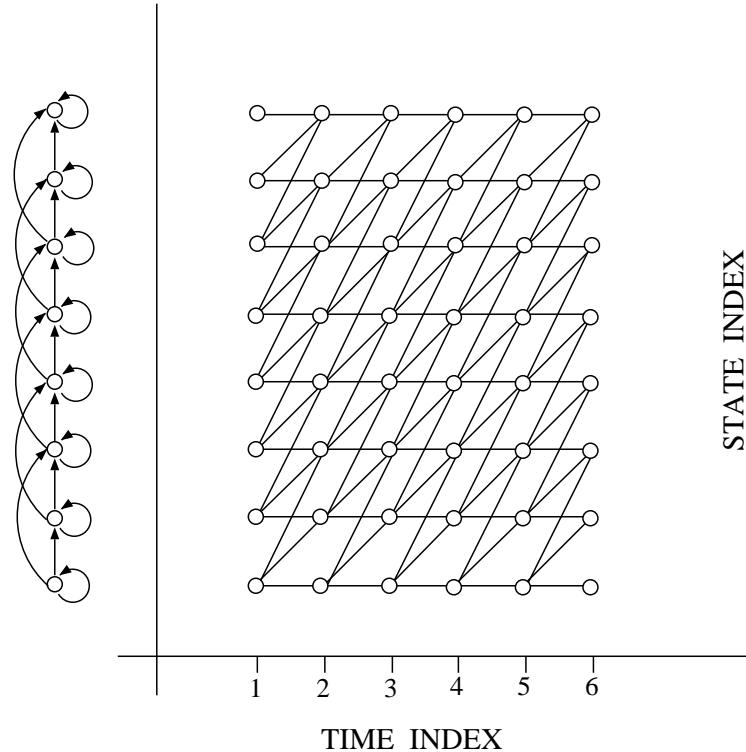
- sentence: concatenation of words $W = w_1 \dots w_n \dots w_N$
modelled by (statistical) language model $p(W)$
- words: concatenation of phonemes (sounds)
modelled by (conventional) pronunciation dictionary
- phonemes (sounds): time signal or waveform
 - acoustic (spectral) analysis every 10 msec
 - result: sequence of observed acoustic vectors (10-ms events) $X = x_1 \dots x_t \dots x_T$
 - fundamental problem in modelling $p(X|W)$: variability in speaking rate



Statistical Approach to Automatic
Speech Recognition (ASR)
[Bahl & Jelinek⁺ 83]

9.1.2 Hidden Markov Models (HMM)

- fundamental problem in ASR:
non-linear time alignment
- Hidden Markov Model:
 - linear chain of states $s = 1, \dots, S$
 - transitions: forward, loop and skip
 - first-order dependences
- trellis:
 - unfold HMM over time $t = 1, \dots, T$
 - path: state sequence $s_1^T = s_1 \dots s_t \dots s_T$
 - observations: $x_1^T = x_1 \dots x_t \dots x_T$

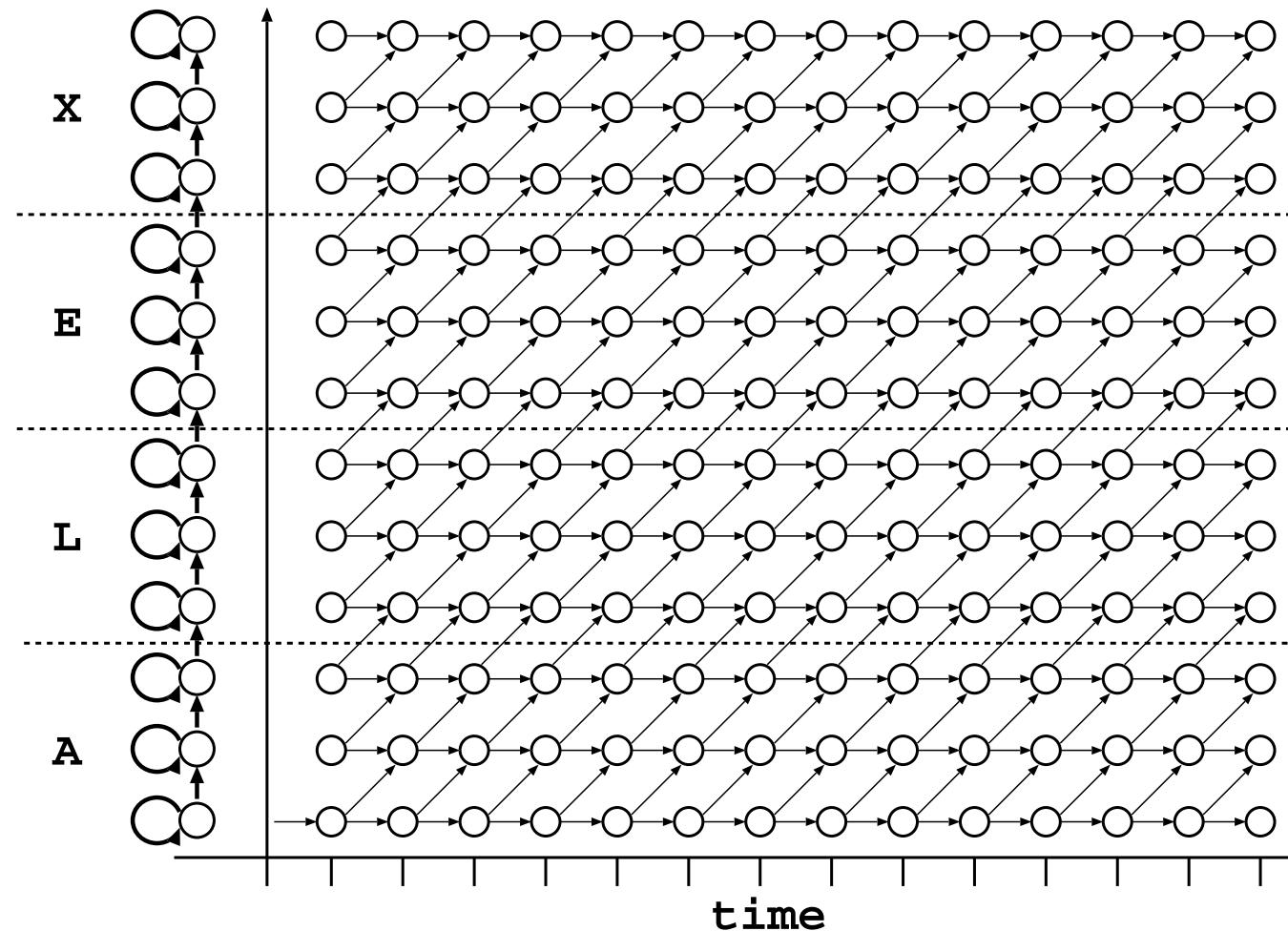


general view:

- two sequences without synchronization: acoustic vectors and states (with labels)
- HMM: mechanism that takes care of the synchronization (=alignment) problem

HMM of a word sequence:

- an HMM for each sound/phoneme (or generalized triphone, CART)
- concatenation of phoneme HMMs according to pronunciation lexicon



The acoustic model $p(X|W)$ provides the link between sentence hypothesis W and observations sequence $X = x_1^T = x_1 \dots x_t \dots x_T$:

- acoustic probability $p(x_1^T|W)$ using hidden state sequences s_1^T :

$$p(x_1^T|W) = \sum_{s_1^T} p(x_1^T, s_1^T|W) = \sum_{s_1^T} \prod_t [p(s_t|s_{t-1}, W) \cdot p(x_t|s_t, W)]$$

- two types of distributions:
 - transition probability $p(s|s', W)$: not important
 - emission probability $p(x_t|s, W)$: key quantity
realized by GMM: Gaussian mixtures models (trained by EM algorithm)
- phonetic labels (allophones, sub-phones): $(s, W) \rightarrow \alpha = \alpha_{sW}$

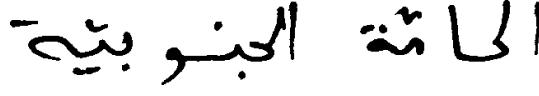
$$p(x_t|s, W) = p(x_t|\alpha_{sW})$$

typical approach: phoneme models in triphone context:
decision trees (CART) for finding equivalence classes

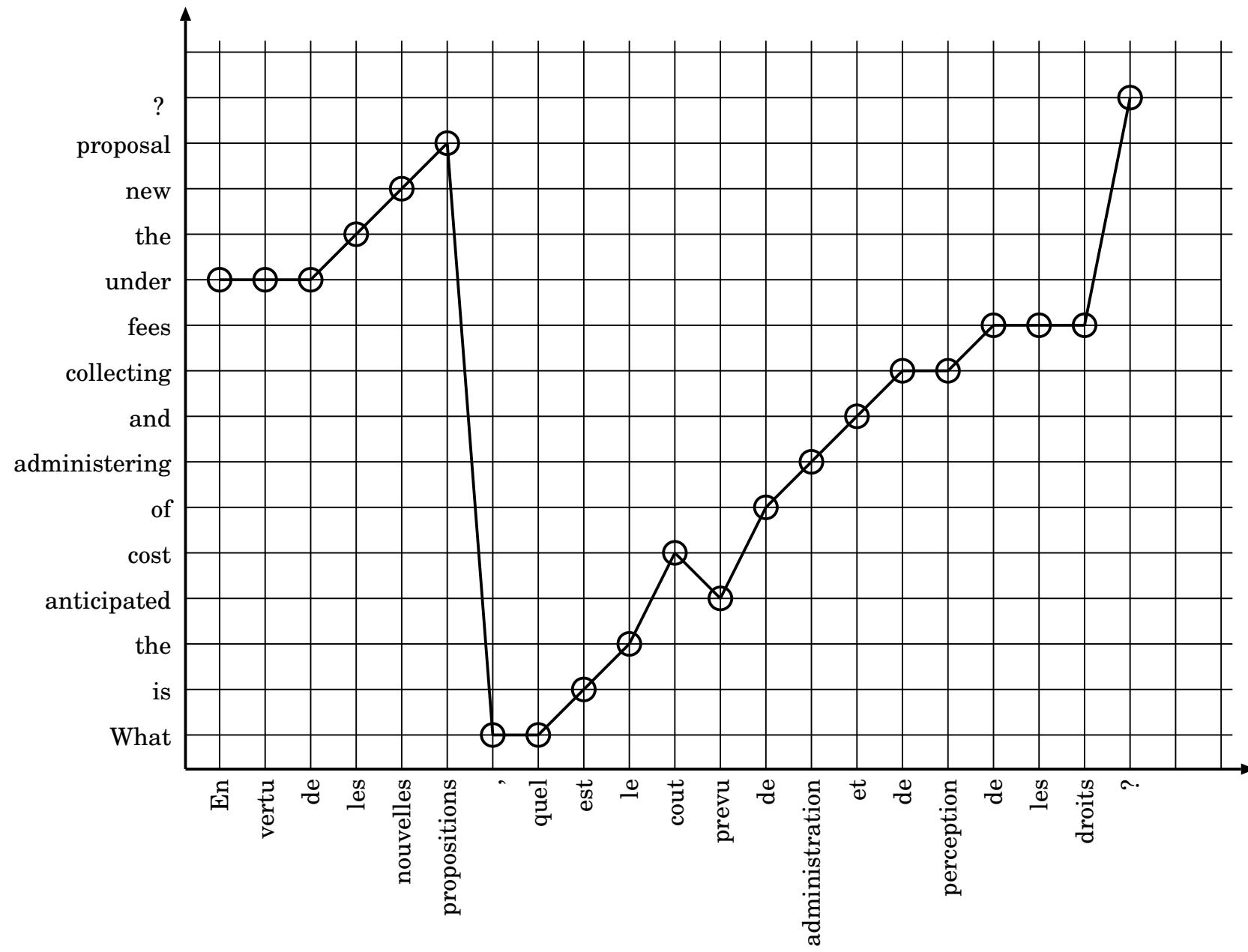
- refinements:
 - augmented feature vector: context window around position t
 - subsequent LDA (linear discriminant analysis)

image text recognition:

- define vertical slots over horizontal axis
- result: image signal = (quasi) one-dim. structure like speech signal

Language	Database	Example
French	RIMES	
Arabic	IfN/ENIT	
English	IAM	

From Speech Recognition to Machine Translation



Algorithms for HMMs

How to compute $p(x_1^T | W) = \sum_{s_1^T} p(x_1^T, s_1^T | W)$ efficiently for a pair (x_1^T, W) ?

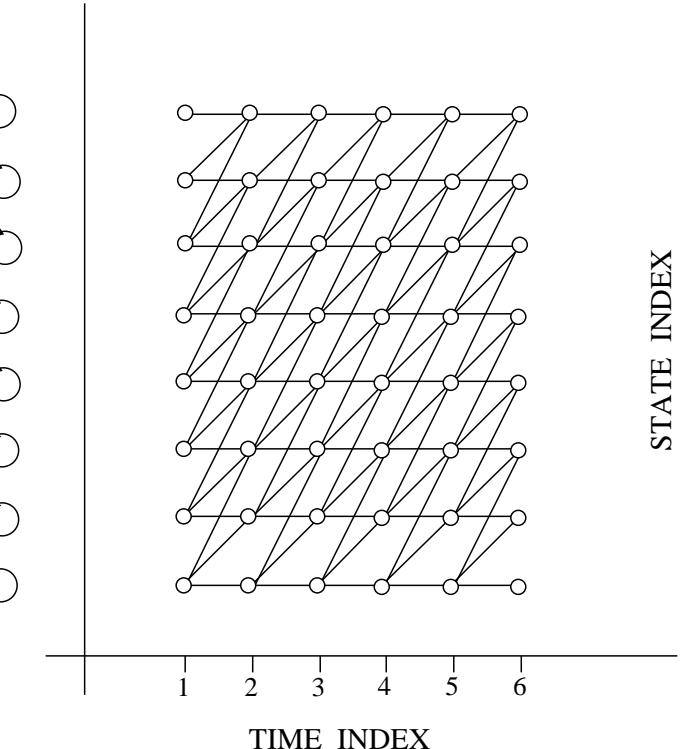
solution:

exploit first-order dependence of HMM structure:

- define two-dimensional array of grid points (s, t)
- define auxiliary quantity $Q(s, t)$
- evaluate recurrence equation of DP:

$$Q(s, t) = p(x_t | s, W) \cdot \sum_{s'} [p(s | s', W) \cdot Q(s', t - 1)]$$

terminology: forward recurrence/algorithm



How to compute $\max_{s_1^T} \{p(x_1^T, s_1^T | W)\}$ efficiently for a pair (x_1^T, W) ?

solution: similar to sum problem:

$$Q(s, t) = p(x_t | s, W) \cdot \max_{s'} [p(s | s', W) \cdot Q(s', t - 1)]$$

terminology: best path, maximum approximation, Viterbi algorithm

How to perform maximum likelihood estimation for the unknown parameters θ ?

$$\max_{\theta} \left\{ \sum_r \log p_{\theta}(x_1^{T_r} | W_r) \right\} = \max_{\theta} \left\{ \sum_r \log \sum_{s_1^{T_r}} p_{\theta}(s_1^{T_r}, x_1^{T_r} | W_r) \right\}$$

for a set of training sentences $(X_r, W_r), r = 1, \dots, R$

two solutions:

- expectation-maximization (EM) algorithm
 - iterative procedure with guaranteed local convergence
 - allows closed form solution in each iteration step
 - requires forward-backward recurrence (Baum-Welch algorithm)
 - offers a direct statistical interpretation in terms of state posterior probabilities
- simplification: maximum (or Viterbi) approximation
use only the single best path

practice for a traditional HMM:

- emission model: Gaussian mixtures
- implementation requires approximations:
beam search, maximum (or Viterbi) approximation,...

9.1.3 Generative vs. Discriminative Training

starting point:

- models $p_\theta(W)$ and $p_\theta(X|W)$ with unknown parameters θ
- training data: set of (audio, sentence) pairs $(X_r, W_r), r = 1, \dots, R$

training criteria in ASR:

- generative model: maximum likelihood (along with EM/Viterbi algorithm):

$$F(\theta) = \sum_r \log p_\theta(W_r, X_r) = \sum_r \log p_\theta(W_r) + \sum_r \log p_\theta(X_r|W_r)$$

nice property: decomposition into two separate problems:

- language model $p_\theta(W)$: without annotation!
- acoustic model $p_\theta(X|W)$: with annotation!
- sentence posterior probability (MMI = maximum mutual information)
[Bahl & Brown⁺ 86], [1991 Normandin]:

$$F(\theta) = \sum_r \log p_\theta(W_r|X_r) \quad p_\theta(W|X_r) := \frac{p_\theta(W) p_\theta(X_r|W)}{\sum_{W'} p_\theta(W') p_\theta(X_r|W')}$$

today's terminology: sequence discriminative training

- (C.-H. Lee's) MCE: minimum classification error
(old concept in pattern recognition) with smoothing parameter β

$$F(\theta) = \sum_r \frac{1}{1 + \left(\frac{p_\theta(X_r, W_r)}{\sum_{W \neq W_r} p_\theta(X_r, W)} \right)^{2\beta}}$$

- [Povey & Woodland 02] MWE/MPE: minimum word/phoneme error (= expected 'accuracy'):

$$F(\theta) = \sum_r \sum_W p_\theta(W|X_r) \cdot A(W, W_r)$$

with the accuracy $A(W, W_r)$ of hypothesis W for correct sentence W_r :
= count of correct frame labels (e.g. phoneme: 1 out of 50)

remarks:

- initialization by maximum likelihood
- complex optimization problem: sum over all sentences in denominator
- approximation: word lattice, many shortcuts, ...
- experiments: relative improvement by 5-10% over maximum likelihood

baseline training of HMM:

- maximum likelihood by EM (expectation/maximization) algorithm
- looks like the ultimate and perfect solution

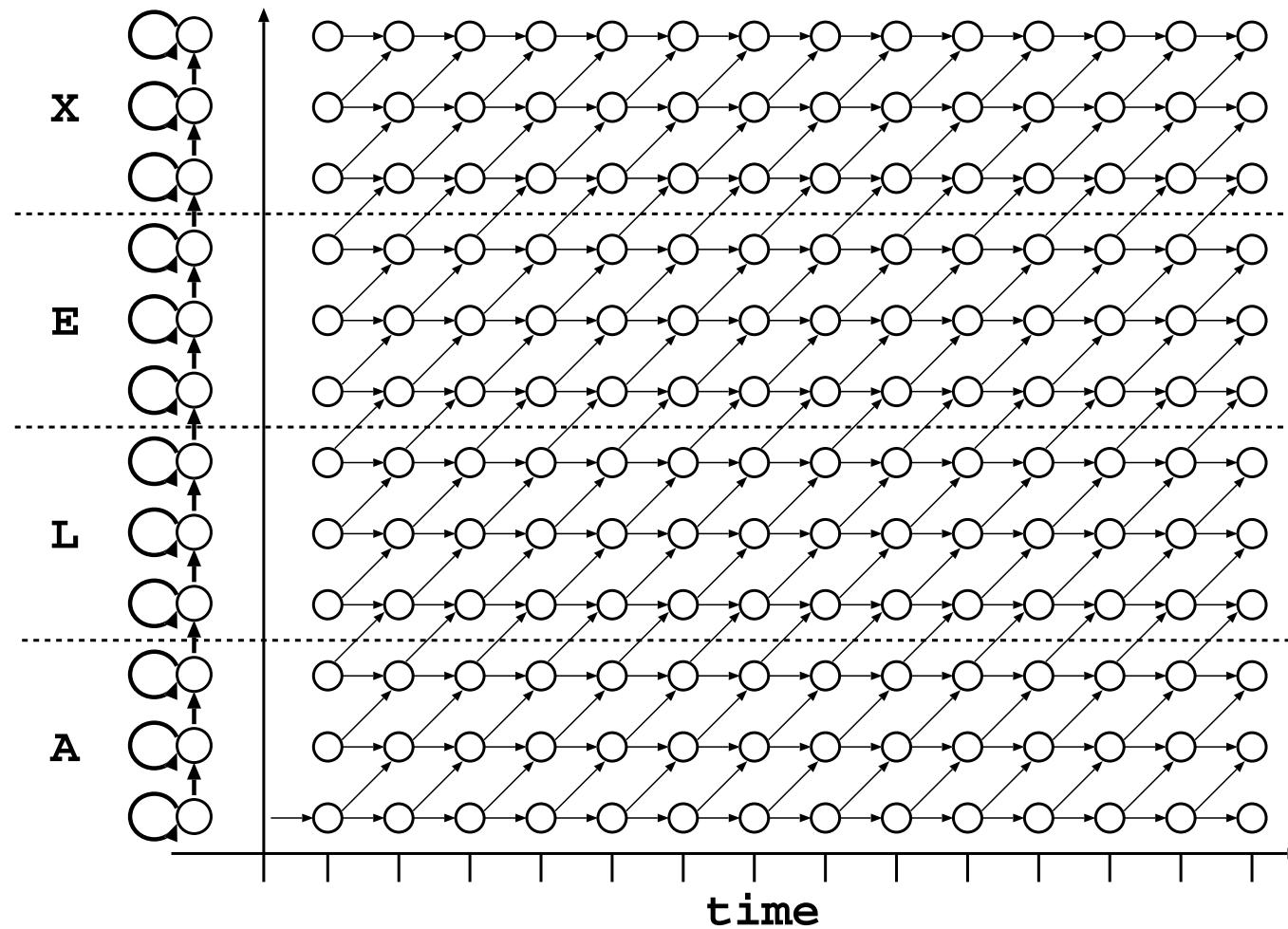
positive properties:

- FULL generative model: $p_\theta(W, X) = p_\theta(W) \cdot p_\theta(X|W)$
along with HMM for $p_\theta(X|W)$: describes the problem completely
- natural training criterion:
 - maximum likelihood, i.e. $\max_\theta \left\{ \sum_r \log p_\theta(W_r, X_r) \right\}$
 - virtually closed form solutions by EM algorithm
 - nice from the mathematical point of view

negative properties:

- EM or maximum likelihood criterion
 - solves a problem that is more complex than required, i.e. $p_\theta(W, X)$ vs. $p_\theta(W|X)$
 - VERY hard from the estimation (learning) point of view
- well-known in classical pattern recognition, but ignored/overlooked in ASR:
density estimation, i.e. learning $p_\theta(X|W)$ or $p_\theta(x_t|\alpha)$, is much harder than
classification, i.e. learning $p_\theta(W|X)$ or $p_\theta(\alpha|x_t)$
- hybrid HMM (since 1989): model $p_\theta(\alpha|x_t)$ by an ANN

alignment problem: HMM or related structure (CTC)



consider modelling the acoustic vector x_t in an HMM:

- re-write the emission probability for label α and acoustic vector x_t
(strictly speaking: an approximation only):

$$p(x_t|\alpha) = p(x_t) \cdot \frac{p(\alpha|x_t)}{p(\alpha)}$$

- prior probability $p(\alpha)$: estimated as relative frequencies
- for recognition purposes: the term $p(x_t)$ can be dropped
- result: model the label posterior probability by an ANN:

$$x_t \rightarrow p(\alpha|x_t)$$

rather than the state emission distribution $p(x_t|\alpha)$

- justification:
 - easier learning problem: labels $\alpha = 1, \dots, 5000$ vs. vectors $x_t \in \mathbb{R}^{D=40}$
 - well-known result in pattern recognition (but ignored in ASR!)

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 HMM:
we replace the emission probabilities by ANN ouputs
- 1993 [Haffner 93]: sum over label-sequence posterior probabilities in hybrid HMMs (CTC like approach)
- 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:

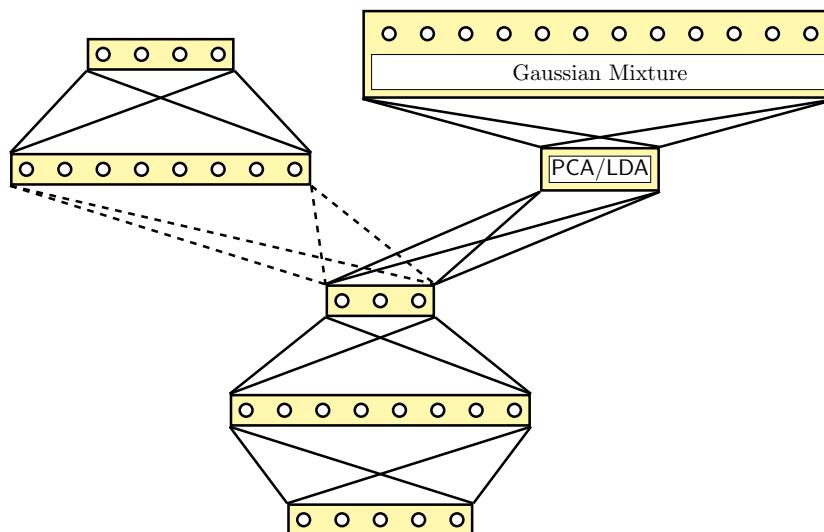
ANNs were never superior to Gaussian mixture models

related work before 2000:

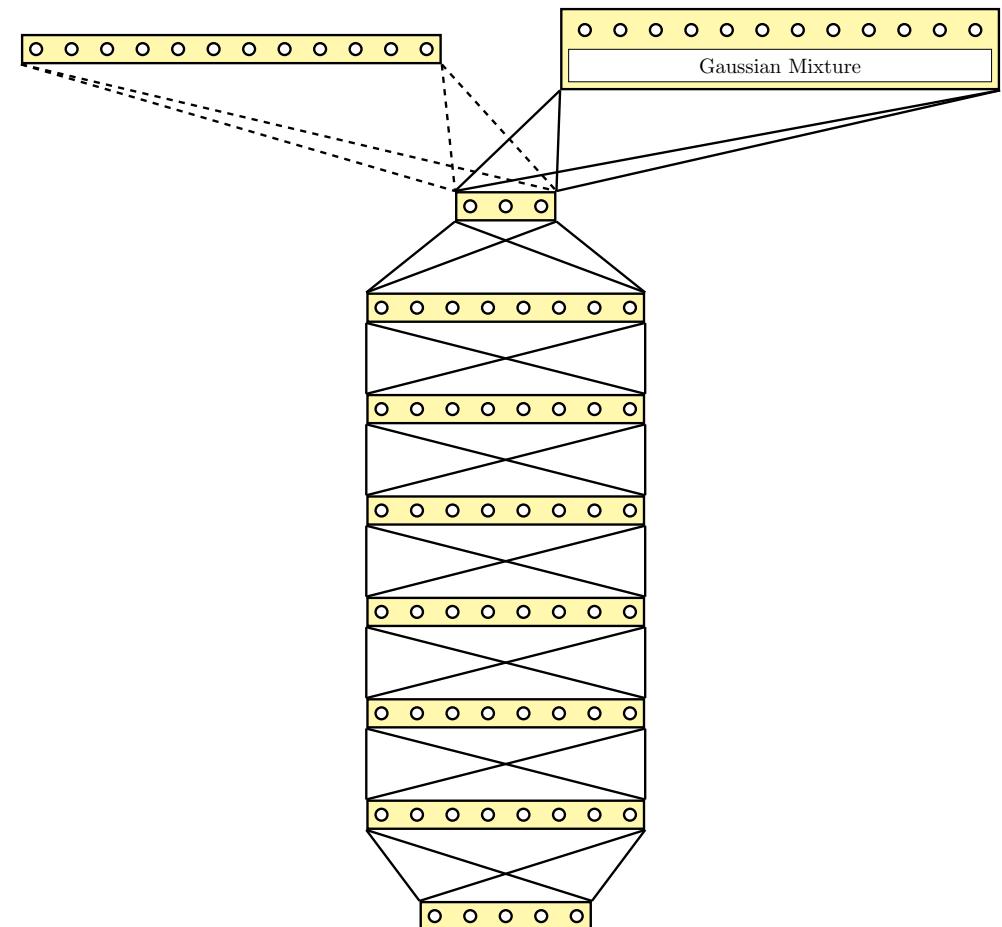
- 1994 [LeCun & Bengio⁺ 94]:
convolutional neural networks (CNNs) for image recognition
- 1997 A. Waibel's team [Fritsch & Finke⁺ 97]:
hierarchical mixtures of experts
- 1997 [Hochreiter & Schmidhuber 97]:
long short-term memory (LSTM) neural computation
with extensions [Gers & Schraudolph⁺ 02]
- 1997 [Schuster & Paliwal 97]:
RNN: bi-directional extension

approach:

- tandem: use MLP for feature extraction in a 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:

- 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

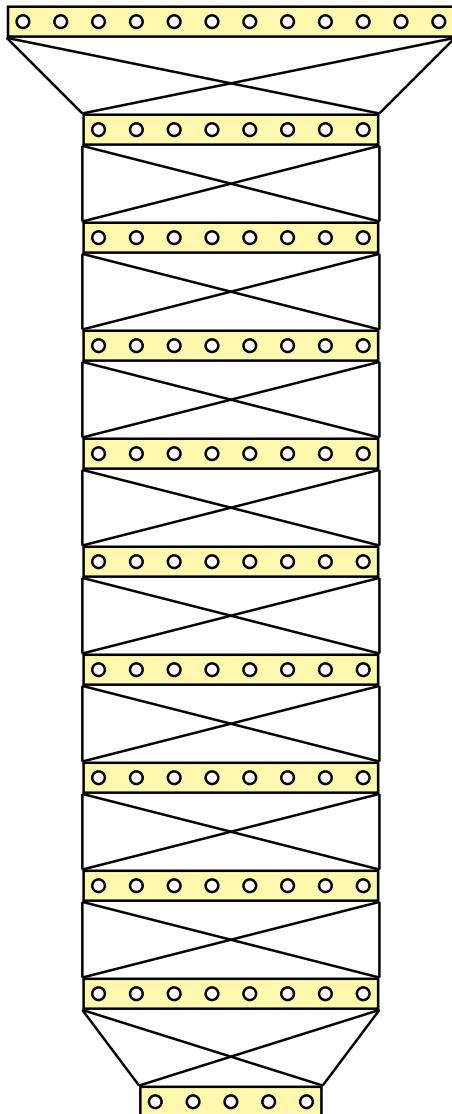
hybrid approaches:

- 2008 [Graves 08]: good results on LSTM RNN for handwriting task (in CTC context)
- 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



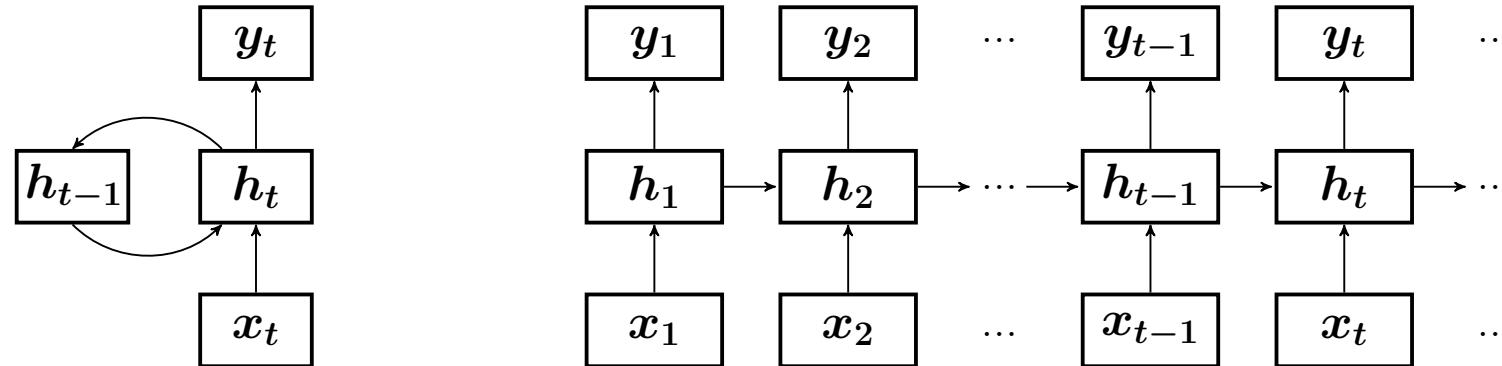


question: what is different now after 30 years?

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

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

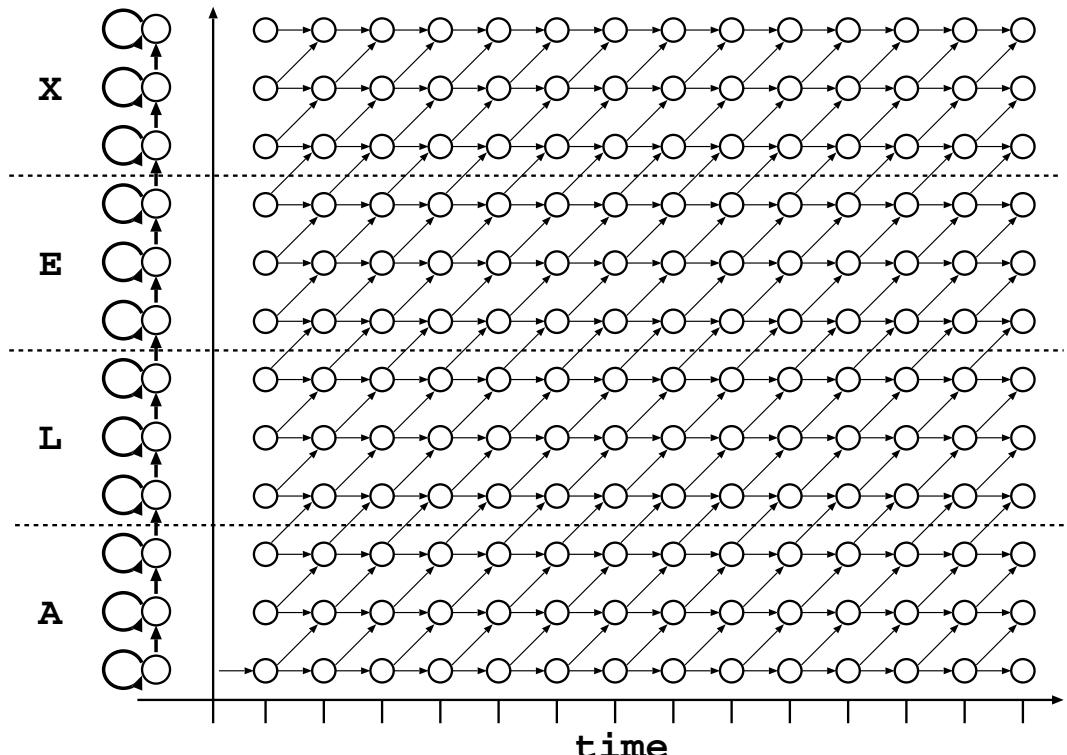
- bidirectional structure [Schuster & Paliwal 97]
- LSTM: long short-term memory
[Hochreiter & Schmidhuber 97, Gers & Schraudolph⁺ 02]

9.3 From HMM to Hybrid HMM and CTC

for each class symbol (sound or character),
define HMM:

- sequence of states (e.g. three) with label state posterior probability
- set of transitions with transition probability

main purpose: time alignment



- traditional HMM: acoustic model of observations $x_1^T = x_1 \dots x_t \dots x_T$ over time $t = 1, \dots, T$ for word string W with hidden state sequence $s_1^T = s_1 \dots s_t \dots s_T$:

$$p(x_1^T | W) = \sum_{s_1^T} p(s_1^T, x_1^T | W) = \sum_{s_1^T} \prod_t [p(s_t | s_{t-1}, W) \cdot p(x_t | s_t, W)]$$

with first-order transition model $p(s_t | s_{t-1}, W)$ and emission model $p(x_t | s_t, W)$

- emission model depends only on annotations (= labels) $A(s_t, W)$ for each state s_t (e. g. CART phones, allophones, sub-phones) of word string W :

$$(s_t, W) \rightarrow a_t = A_{W s_t} \quad \text{and} \quad p(x_t | s_t, W) = p(x_t | a_t)$$

- hybrid approach: re-write the acoustic model $p(x_1^T | W)$ with label sequence a_1^T using label priors $p(a_t)$ and observation marginals $p(x_t)$:

$$p(x_t | a_t) = p(a_t | x_t) \cdot p(x_t) / p(a_t)$$

$$p(x_1^T | W) = \sum_{a_1^T} \prod_t [p(a_t | a_{t-1}, W) \cdot p(a_t | x_t) \cdot p(x_t) / p(a_t)]$$

after replacing state sequence s_1^T of word string W by its label sequence a_1^T

slight abuse of notation in dropping dependence on W :

$$p(x_1^T | W) = \sum_{s_1^T} \prod_t [p(s_t | s_{t-1}, W) \cdot p(A_{W s_t} | x_t) \cdot p(x_t) / p(A_{W s_t})]$$

- posterior probability of word string W for hybrid approach:

$$p(W|x_1^T) := \frac{p(W) \cdot p(x_1^T|W)}{\sum_{\tilde{W}} p(\tilde{W}) \cdot p(x_1^T|\tilde{W})} = \frac{p(W) \cdot \sum_{a_1^T} \prod_t [p(a_t|a_{t-1}, W) \cdot p(a_t|x_t)/p_t(a_t)]}{\sum_{\tilde{W}} p(\tilde{W}) \cdot \sum_{a_1^T} \prod_t [p(a_t|a_{t-1}, \tilde{W}) \cdot p(a_t|x_t)/p_t(a_t)]}$$

remarks:

- the marginal probabilities $p(x_t)$ have cancelled!
- extension: replace single vector x_t by a context window: $p(a_t|x_{t-\delta}^{t+\delta})$
or the 'whole' input: $p_t(a_t|x_1^T)$ (using LSTM-RNN)
- log-linear framework: use exponent (scaling factor) for each of the four models

- recognition using Bayes decision rule with maximum approximation:

$$\begin{aligned} x_1^T \rightarrow \hat{W}(x_1^T) &= \operatorname{argmax}_W \left\{ p(W|x_1^T) \right\} = \operatorname{argmax}_W \left\{ p(W) \cdot p(x_1^T|W) \right\} \\ &\approx \operatorname{argmax}_W \left\{ p(W) \cdot \max_{a_1^T} \left\{ \prod_t [p(a_t|a_{t-1}, W) \cdot p_t(a_t|x_1^T)/p_t(a_t)] \right\} \right\} \end{aligned}$$

search strategy: time-synchronous dynamic programming
with extensions like beam search, look-ahead, etc.

Training Strategies

posterior probability of word string W for hybrid approach:

$$p(W|x_1^T) = \frac{p(W) \cdot \sum_{a_1^T} \prod_t [p(a_t|a_{t-1}, W) \cdot p_t(a_t|x_1^T)/p_t(a_t)]}{\sum_{\tilde{W}} p(\tilde{W}) \cdot \sum_{a_1^T} \prod_t [p(a_t|a_{t-1}, \tilde{W}) \cdot p_t(a_t|x_1^T)/p_t(a_t)]}$$

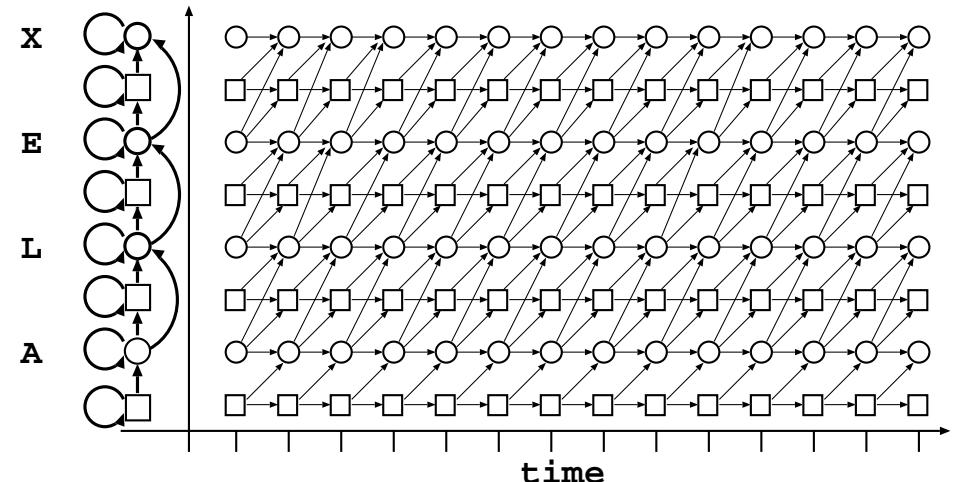
training strategies:

- discriminative at frame level: *frame-wise cross-entropy*
 - justification: ignore denominator and replace state sum by maximum
 - required: single best path for each training sentence
 - re-alignments during backprop learning: yes ... occasionally ... no
 - result: simple implementation due to decoupling of best path and backprop
- discriminative at (word/label) sequence level: *sequence-wise cross-entropy*,
sequence discriminative training, (sequence) MMI:
 - keep the denominator and include the language model of words (or labels) $p(W)$
 - various types of *sequence discriminative training* criteria:
 - MMI: posterior probability only
 - Povey's MPE applied to phone labels: 1 out of 50
 - Povey's MPE applied to CART labels: 1 out of 5000
 - result: difficult optimization problem → many approximations:
single best path, lattice with/without re-computation, ...

modified structure of an hybrid HMM:

- topology: two states only
- tie first state across all symbols:
white space (or garbage) label
that can be skipped
- drop transition probabilities
- drop prior probabilities

neural net: LSTM RNN



suggested training criterion for CTC:

SUM over label sequences a_1^T (of word string W) for an observation sequence x_1^T :

$$\max_{\vartheta} \left\{ \log \sum_{a_1^T} \prod_t p_t(a_t | x_1^T; \vartheta) \right\}$$

interpretation:

- usual choice of sum vs. maximum
- for hybrid HMM: [Haffner 93]

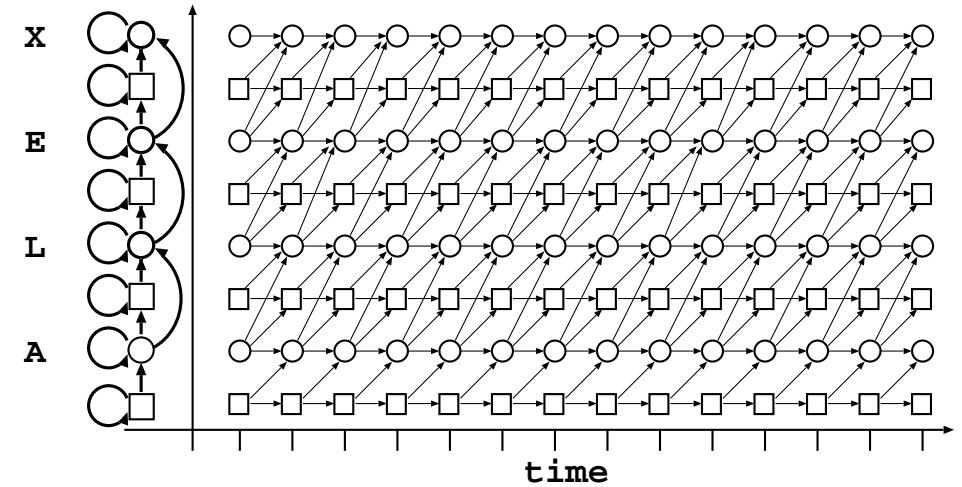
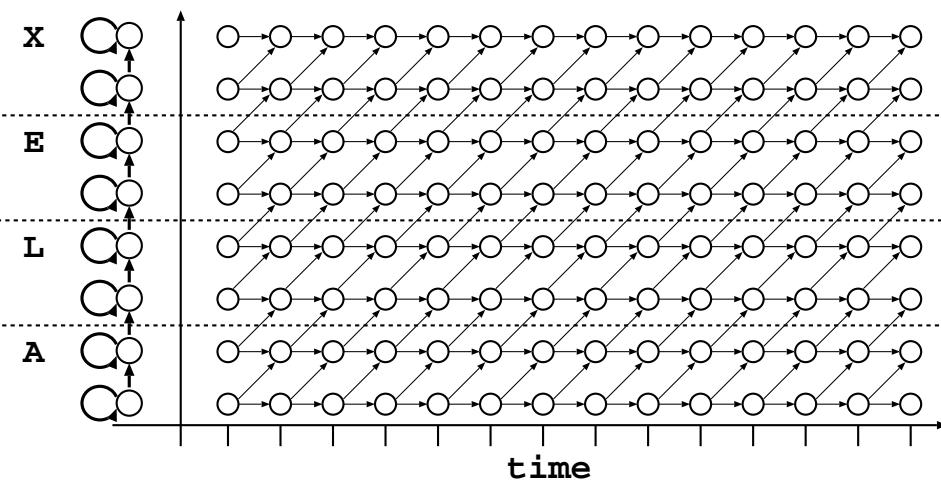
re-visit sequence discriminative training criterion
(sequence MMI, Povey's MWE and MPE, CTC and hybrid HMM):

$$p(W|x_1^T) = \frac{p(W) \cdot \sum_{a_1^T} \prod_t [p(a_t|a_{t-1}, W) \cdot p_t(a_t|x_1^T)/p_t(a_t)]}{\sum_{\tilde{W}} p(\tilde{W}) \cdot \sum_{a_1^T} \prod_t [p(a_t|a_{t-1}, \tilde{W}) \cdot p_t(a_t|x_1^T))/p_t(a_t)]}$$

implementation/approximation strategies:

- **drop denominator: what about sum over label sequences in numerator?**
 - keep it: → CTC or related criterion
terminology: *end-to-end training*: misnomer since no LM is used!
 - replace it by maximum: → framewise training with cross-entropy;
nice decoupling of best path problem and backpropagation
- **keep denominator: what about sum over label sequences in numerator and denominator?**
 - keep it and approximate word sequence sum:
by either simplified LM or dense lattice
 - replace it (in N+D) by maximum and approximate word sequence sum:
by either simplified LM or dense lattice

Comparison of Topologies: Two-State Hybrid HMM vs. CTC



result:

- CTC and two-state HMM very similar
- difference: transitions/transition probabilities and re-scaling by priors
- number of free parameters: virtually the same
- only output layer is different: C vs. $2 \cdot C$ output nodes

recognition criterion: best path with priors and transition probabilities

- **classical EM algorithm:**
 - optimization problem with explicit probability models with hidden random variables
typical problem: maximum likelihood estimation
 - 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
 - neural HMM (and CTC models) for speech recognition
 - neural HMM for translation
- **specific aspects to be 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

problem specification:

- general form of optimization problem:

consider a function with non-negative components $q(y, x_n; \lambda)$

for observations $x_1 \dots x_n \dots x_N$ with a hidden variable y and parameter λ :

$$F(\lambda) := \sum_n \log \sum_y q(y, x_n; \lambda)$$

- remarks:

- no normalization constraints
- gradient search in lieu of closed-form solutions

- compare with classical EM algorithm:

- explicit probabilistic models with hidden random variables
- closed-form solutions within each iteration

Inequality for EM Algorithm: Classical Proof based on Divergence Inequality

consider function difference between new and old parameters:

$$\begin{aligned}
 F(\hat{\lambda}) - F(\lambda) &= \sum_n \log \frac{\sum_y q(y, x_n; \hat{\lambda})}{\sum_y q(y, x_n; \lambda)} \\
 &= \sum_n \sum_y w(y|x_n, \lambda) \cdot \log \frac{\sum_{y'} q(y', x_n; \hat{\lambda})}{\sum_{y'} q(y', x_n; \lambda)} \\
 \text{define: } w(y|x, \lambda) &= \frac{q(y, x; \lambda)}{\sum_{y'} q(y', x; \lambda)} \\
 &= \sum_n \sum_y w(y|x_n, \lambda) \cdot \log \frac{q(y, x_n; \hat{\lambda}) \cdot w(y|x_n, \lambda)}{q(y, x_n; \lambda) \cdot w(y|x_n, \hat{\lambda})} \\
 &= \sum_n \sum_y w(y|x_n, \lambda) \cdot \log \frac{q(y, x_n; \hat{\lambda})}{q(y, x_n; \lambda)} + \sum_n \sum_y w(y|x_n, \lambda) \cdot \log \frac{w(y|x_n, \lambda)}{w(y|x_n, \hat{\lambda})}
 \end{aligned}$$

second sum: apply divergence inequality

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

important result: normalized posterior weights (without assuming probabilistic models!)

Inequality for EM Algorithm: Novel Proof based on Log-Sum Inequality

novelties:

- EM beyond probability distributions
- direct proof using log-sum inequality (shortest proof ever?)

consider function difference between new and old parameters:

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

use log-sum inequality for non-negative numbers a_k and b_k :

$$\begin{aligned} \log \frac{\sum_k a_k}{\sum_k b_k} &\geq \sum_k \frac{b_k}{(\sum_{k'} b_{k'})} \log \frac{a_k}{b_k} \\ &\geq \sum_n \sum_y w(y|x_n, \lambda) \cdot \log \frac{q(y, x_n; \hat{\lambda})}{q(y, x_n; \lambda)} \end{aligned}$$

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

- we have proved the baseline inequality:

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

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

$$= Q(\lambda, \hat{\lambda}) - Q(\lambda, \lambda)$$

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

terminology: EM operations applied to auxiliary function $Q(\lambda, \hat{\lambda})$:

- E: expectation using posterior weights $w(y|x, \lambda)$
- M: maximization of $\hat{\lambda} \rightarrow Q(\lambda, \hat{\lambda})$ in lieu of $\hat{\lambda} \rightarrow F(\hat{\lambda})$

- iterative optimization: two alternating steps:

- compute posterior weights $w(y|x, \lambda)$ using present value of λ
- maximize $Q(\lambda, \hat{\lambda})$ over $\hat{\lambda}$ using gradient search:

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

note: classical EM algorithm has closed-form solutions

definitions: function to be optimized and posterior weights:

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

direct calculation of gradient:

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

compare with gradient in EM algorithm:

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

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

- **structure of optimization problem:** sum over hidden variable:
no explicit probabilistic structure
- **remark: existence of (local) optimum:**
has to be verified independently (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:**
hybrid HMM (incl. CTC) with first-order dependences

9.4.2 Hybrid HMM and EM Algorithm

- **HMM structure:** generative, hybrid, CTC, ...
important property: first-order dependencies
- most interesting application of the EM algorithm
- today's importance of EM algorithm:
 - is less important than before
 - is superseded by gradient search (backpropagation)

Hybrid HMM

- sequence of acoustic vectors:

$$x_1^T = x_1 \dots x_t \dots x_T \text{ over time } t$$

- sequence of states $s = 1, \dots, S$

$$s_1^T = s_1 \dots s_t \dots s_T \text{ over time } t$$

with associated state labels:

$$a_1^S = a_1 \dots a_s \dots a_S$$

= W : word sequence

objective of HMM: time alignment

= synchronization between input and output

- classical HMM: generative model for x_1^T :

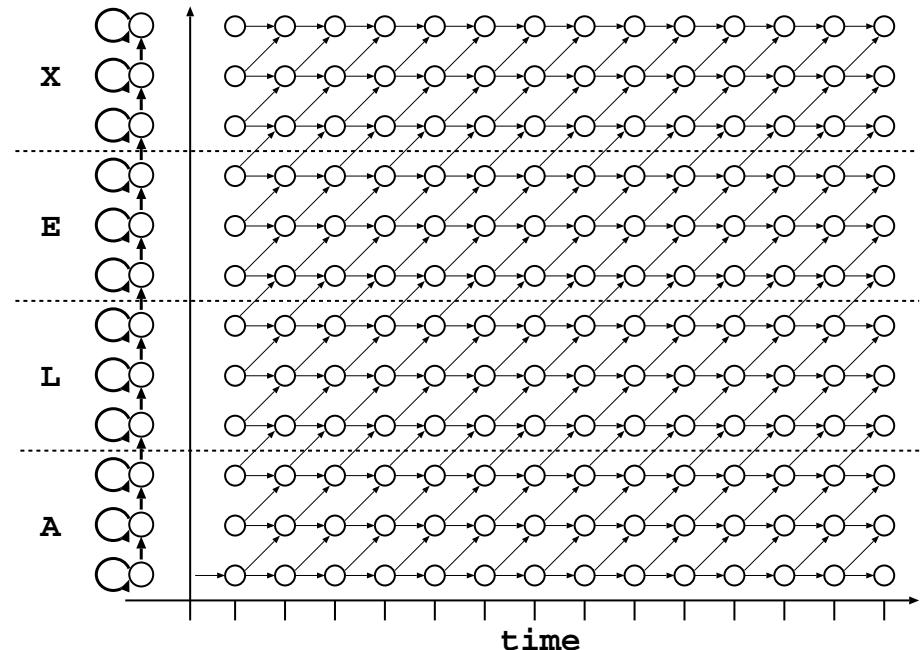
$$q(x_1^T | W = a_1^S) = \sum_{s_1^T} \prod_t q(s_t | s_{t-1}, W) \cdot q_t(x_t | a_{s=s_t})$$

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

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

justification: it is easier to model $q_t(a_s | x_1^T)$ than $q_t(x_t | a_s)$

[Bourlard & Wellekens 89], CTC: [Graves & Fernandez⁺ 06]



- **sequence of acoustic vectors:**

$$x_1^T = x_1 \dots x_t \dots x_T \text{ over time } t$$

- **sequence of states** $s = 1, \dots, S$

$$s_1^T = s_1 \dots s_t \dots s_T \text{ over time } t$$

- with associated state labels:**

$$a_1^S = a_1 \dots a_s \dots a_S$$

- $= W$: word sequence**

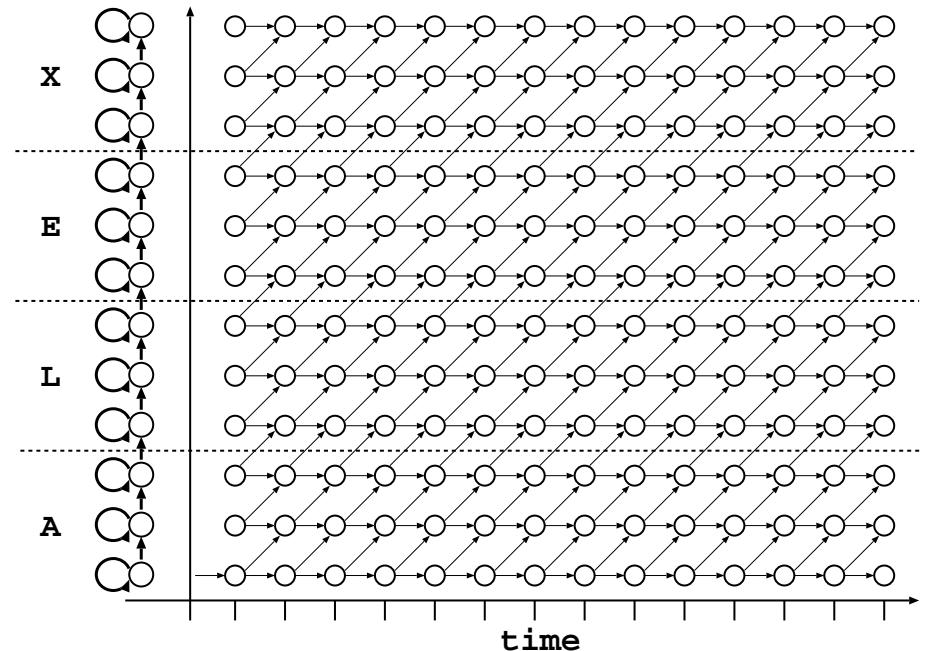
hybrid HMM:

- **full notation with parameters λ :**

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

- **simiplified notation (for the EM algorithm):**

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



specification of optimization problem:

- **single training sample with input sequence $x \equiv x_1^T = x_1 \dots x_t \dots x_T$ (without explicit symbols for output label sequence):**

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

with model components $q_t(s_t, s_{t-1}, x_1^T; \lambda) \geq 0$

- **training input sequences x_n , $n = 1, \dots, N$, each x_n with length T_n (needed for the alignment path $s_1^{T_n}$ with $T = T_n$)**
- **function to be optimized in comparison with general case:**

first-order model: $\lambda \rightarrow F(\lambda) := \sum_{n=1}^N \log \sum_{s_1^{T_n}} \prod_{t=1}^{T_n} q_t(s_{t-1}, s_t, x_n; \lambda)$

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

- **calculation of gradient:
will exploit the first-order dependence of the optimization problem**

results for general form of gradient:

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

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

apply the above results to first-order HMM:

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

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

with posterior weights $w_t(s', s|x_n, \lambda)$ as in classical HMM-EM algorithm:

$$w_t(s', s|x_n, \lambda) := \sum_{s_1^{T_n}:s'=s_{t-1},s_t=s} w(s_1^{T_n}|x_n, \lambda)$$

compare:

- gradient search without EM algorithm
(derivation: see previous slide):

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

- gradient search with EM algorithm
(derivation: similar to previous slide):

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

difference: update strategies for posterior weights

applications and experiments:

- generative HMM
- hybrid HMM
- segmental (direct) HMM
- neural HMM for machine translation

question:

for what type of training criterion can the EM algorithm be used?

- discriminative or generative?
- sequence discriminative: sequence as a whole?
- sequence discriminative: symbol in sequence context?

answer:

it depends on the mathematical structure of the optimization problem

Alternative Derivation: EM Algorithm and Gradient Search

- **simplification (w.l.o.g): one very long sequence** $x_1^T = x_1 \dots x_t \dots x_T$
- **model (without normalization!) for each state** $s = 1, \dots, S$:

$$q_t(s, x_1^T | \lambda) \geq 0$$

- **overall score: sum over all state sequences Y**

$$F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda)$$

functional dependence:

$$\lambda \rightarrow \{q_t(s, x_1^T | \lambda) : s, t\} \rightarrow F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda)$$

Direct Calculation of Gradient: Remark on Notation

simplification (or abuse) of notation:

$$\begin{aligned} F(\lambda) &:= \sum_{s_1^T} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda) \\ &= \prod_{t=1}^T \sum_{s_t} q_t(s_t, x_1^T | \lambda) = \prod_{t=1}^T \sum_s q_t(s, x_1^T | \lambda) \end{aligned}$$

i. e. there is no real first-order dependence

correct notation:

- constraints on set of sequences \mathcal{S} :

$$F(\lambda) := \sum_{s_1^T \in \mathcal{S}} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda)$$

- full first-order dependence:

$$F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_{t-1}, s_t, x_1^T | \lambda)$$

definitions: function and posterior weights:

$$F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda) \quad w_t(s|x_1^T, \lambda) := \sum_{s_1^T: s_t=s} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda) / F(\lambda)$$

derivative:

$$\begin{aligned} \frac{\partial}{\partial \lambda} \log F(\lambda) &= \frac{1}{F(\lambda)} \cdot \frac{\partial}{\partial \lambda} F(\lambda) = \frac{1}{F(\lambda)} \cdot \sum_{s_1^T} \frac{\partial}{\partial \lambda} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda) \\ &= \frac{1}{F(\lambda)} \cdot \sum_{s_1^T} \left(\prod_{\tilde{t}=1}^T q_{\tilde{t}}(s_{\tilde{t}}, x_1^T | \lambda) \right) \cdot \frac{\partial}{\partial \lambda} \log \prod_{t=1}^T q_t(s_t, x_1^T | \lambda) \\ &= \frac{1}{F(\lambda)} \cdot \sum_t \sum_{s_1^T} \left(\prod_{\tilde{t}=1}^T q_{\tilde{t}}(s_{\tilde{t}}, x_1^T | \lambda) \right) \cdot \frac{\partial}{\partial \lambda} \log q_t(s_t, x_1^T | \lambda) \\ &= \frac{1}{F(\lambda)} \cdot \sum_t \sum_s \left(\sum_{s_1^T: s_t=s} \prod_{\tilde{t}=1}^T q_{\tilde{t}}(s_{\tilde{t}}, x_1^T | \lambda) \right) \cdot \frac{\partial}{\partial \lambda} \log q_t(s, x_1^T | \lambda) \\ &= \sum_s \sum_t w_t(s|x_1^T, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s, x_1^T | \lambda) \end{aligned}$$

with posterior weights $w_t(s|x_1^T, \lambda)$, i. e. posterior distribution over states s

functional dependence:

$$\lambda \rightarrow \{q_t(s, x_1^T | \lambda) : s, t\} \rightarrow F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_t, x_1^T | \lambda)$$

derivative:

$$\begin{aligned} \frac{\partial}{\partial \lambda} \log F(\lambda) &= \frac{1}{F(\lambda)} \cdot \frac{\partial}{\partial \lambda} F(\lambda) \\ &= \frac{1}{F(\lambda)} \cdot \sum_s \sum_t \frac{\partial F(\lambda)}{\partial q_t(s, x_1^T | \lambda)} \cdot \frac{\partial q_t(s, x_1^T | \lambda)}{\partial \lambda} \\ &= \frac{1}{F(\lambda)} \cdot \sum_s \sum_t \frac{\partial F(\lambda)}{\partial q_t(s, x_1^T | \lambda)} \cdot q_t(s, x_1^T | \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s, x_1^T | \lambda) \\ &= \dots \text{ (see next slide)} \\ &= \sum_s \sum_t w_t(s | x_1^T, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s, x_1^T | \lambda) \end{aligned}$$

with posterior weights $w_t(s | x_1^T, \lambda)$, i. e. posterior distribution over states s

consider the contributions for a specific time τ in the overall score:

$$\begin{aligned} F(\lambda) &:= \sum_{s_1^T} \prod_t q_t(s_t, x_1^T | \lambda) \\ &= \sum_{\sigma} \sum_{s_1^T : s_\tau = \sigma} \prod_t q_t(s_t, x_1^T | \lambda) = \sum_{\sigma} \sum_{s_1^T : s_\tau = \sigma} q_\tau(\sigma, x_1^T | \lambda) \cdot \prod_{t \neq \tau} q_t(s_t, x_1^T | \lambda) \end{aligned}$$

$$\frac{\partial F(\lambda)}{\partial q_\tau(\sigma, x_1^T | \lambda)} = \sum_{s_1^T : s_\tau = \sigma} \prod_{t \neq \tau} q_t(s_t, x_1^T | \lambda)$$

$$\begin{aligned} w_\tau(\sigma | x_1^T, \lambda) &:= \frac{1}{F(\lambda)} \cdot \frac{\partial F(\lambda)}{\partial q_\tau(\sigma, x_1^T | \lambda)} \cdot q_\tau(\sigma, x_1^T | \lambda) \\ &= \frac{\sum_{s_1^T : s_\tau = \sigma} \prod_t q_t(s_t, x_1^T | \lambda)}{\sum_{s_1^T} \prod_t q_t(s_t, x_1^T | \lambda)} \end{aligned}$$

result: normalized distribution $w_t(s | x_1^T, \lambda)$ over states s
(= posterior state occupation probability in traditional HMM)

define the auxiliary function $Q(\lambda, \hat{\lambda})$ by splitting the dependence on parameter λ between weights and models:

$$Q(\lambda, \hat{\lambda}) := \sum_s \sum_t w_t(s|x_1^T, \lambda) \cdot \log q_t(s, x_1^T | \hat{\lambda})$$

$$\frac{\partial}{\partial \hat{\lambda}} Q(\lambda, \hat{\lambda}) = \sum_s \sum_t w_t(s|x_1^T, \lambda) \cdot \frac{\partial}{\partial \hat{\lambda}} \log q_t(s, x_1^T | \hat{\lambda})$$

compare with baseline gradient:

$$\frac{\partial}{\partial \lambda} \log F(\lambda) = \sum_s \sum_t w_t(s|x_1^T, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s, x_1^T | \lambda)$$

proof: re-visit the classical proof of EM algorithm:

$$F(\hat{\lambda}) - F(\lambda) \geq Q(\lambda, \hat{\lambda}) - Q(\lambda, \lambda)$$

- consider a model $q_t(s_{t-1}, s_t | x_1^T)$ with explicit first-order dependencies:

$$F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_{t-1}, s_t, x_1^T | \lambda)$$

- functional dependence:

$$\lambda \rightarrow \{q_t(s', s, x_1^T | \lambda) : s', s, t\} \rightarrow F(\lambda) := \sum_{s_1^T} \prod_{t=1}^T q_t(s_{t-1}, s_t, x_1^T | \lambda)$$

- gradient:

$$\frac{\partial}{\partial \lambda} \log F(\lambda) = \dots = \sum_{s', s} \sum_t w_t(s', s | x_1^T, \lambda) \cdot \frac{\partial}{\partial \lambda} \log q_t(s', s, x_1^T | \lambda)$$

$$w_\tau(s', s | x_1^T, \lambda) := \frac{\sum_{s_1^T: s_{\tau-1}=s', s_\tau=s} \prod_t q_t(s_{t-1}, s_t, x_1^T | \lambda)}{\sum_{s_1^T} \prod_t q_t(s_{t-1}, s_t, x_1^T | \lambda)}$$

with the posterior weights $w_\tau(s', s | x_1^T, \lambda)$ that can be interpreted as a normalized distribution over state transitions $s' \rightarrow s$ at time t
(as in traditional EM for HMM)

9.5 Experimental Results

QUAERO task and database:

- Training data for acoustic model
 - 250 hours of speech
 - Broadcast news and conversation (e.g. BBC, CNN), podcast, TED talks
 - about 400 recordings
- Test sets:
 - Eval11 and Eval13
 - Broadcast news and conversation, podcast
 - 3 hours each
- Training data for language model
 - 3.1B running words
 - Lectures, blogs, news, web-crawl
 - Resources available in open challenges: IWSLT, WMT
 - * IWSLT: International Workshop on Spoken Language Translation
 - * WMT: Workshop/Conference on Machine Translation

experimental conditions:

- **QUAERO task: English broadcast news and conversations (evaluation campaign 2011)**
- **training data: two conditions: 50 and 250 hours**
- **test data: dev and eval sets, each 3 hours**
- **language model: vocabulary size of 150k (OOV: 0.4%) and perplexity of 130**

baseline acoustic model:

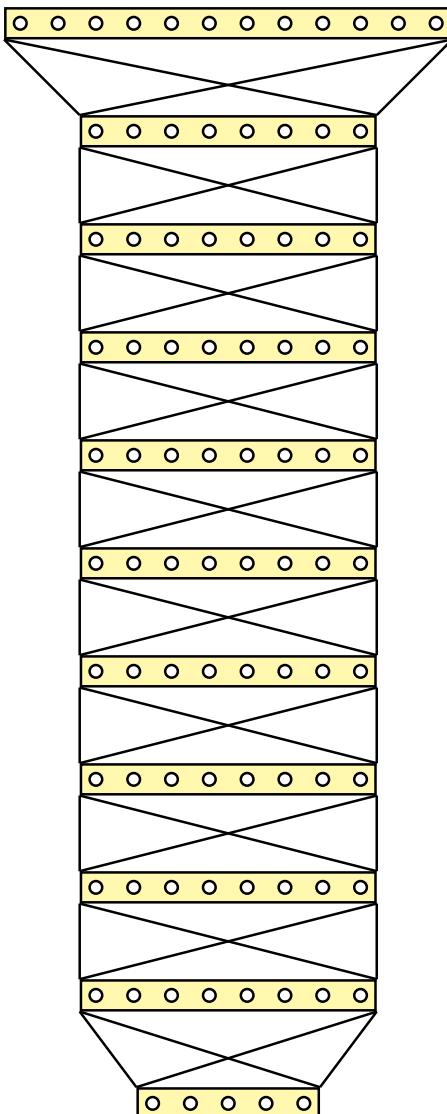
- **feature vector: 16 MFCC (mel frequency cepstral coefficients)**
- **augmented feature vector: $9 \cdot 16$**
- **high-performance baseline system:
Gaussian mixtures with pooled diagonal covariance matrix:**
 - reduction by LDA to 45-dimensional vector
 - 4501 CART labels
 - 680k densities
 - **total number of free parameters: $680k \cdot (45 + 1) = 31.3M$**

word error rates [%]:

Training Criterion	50h		250h	
	dev	eval	dev	eval
Maximum likelihood	24.4	31.6	22.1	28.6
Cross-entropy at frame level	23.9	30.9	22.1	28.6
Cross-entropy at sentence level	24.1	31.2	21.7	28.1
Minimum phone error	23.6	30.2	20.4	26.2

remarks:

- best improvement over maximum likelihood:
5-10% relative by MPE (Minimum Phone Error)
- comparative evaluations in QUAERO:
competitive results with LIMSI Paris and KIT Karlsruhe



**word error rates [%] for 50-h training corpus
as a function of number of hidden layers:**

structure of MLP:

- **input dimension:**
493 (window + derivatives)
- **2000 nodes per hidden layer**
- **nonlinearity: sigmoid**
- **number of parameters:**
6-layer MLP contains 30.0M parameters:

$$493 \cdot 2000 + 5 \cdot 2000^2 + 2000 \cdot 4501$$

$$= 30.0M$$

improvement over best GMM: 20% relative

hidden layers	WER [%]	
	dev	eval
1	24.5	31.3
2	22.0	28.3
3	20.5	26.7
4	19.8	26.1
5	20.1	26.0
6	19.6	25.4
7	19.7	25.5
8	19.6	25.7
9	19.3	25.3
best GMM	23.6	30.2

Acoustic Modelling and Deep Learning: Training Strategy

typical procedure for training:

- initialization: Gaussian models
using Viterbi framework and maximum likelihood
- keep alignment paths fixed
and train triphone CART equivalence classes
- training CART models using Gaussian mixtures
using Viterbi framework and maximum likelihood
- keep alignment paths fixed and
train ANNs at frame level using cross-entropy
- ***sequence discriminative training:***
include language model and consider sequence posterior probability

word error rates [%] for 50-h QUAERO training corpus for different LSTM-RNN variants

- **baseline: best MLP:**

- input: 50 Gammatone features
- 9 hidden layers
- ReLU
- training criterion: cross-entropy

- **LSTM-RNN structure:**

- input: 50 Gammatone features
- training criterion: cross-entropy
- bidirectional with several hidden layers
- 500 nodes per hidden layer
- training on a single GPU

- **improvements:**

- 13% relative over MLP
- 40% relative over GMM

LSTM structure	#params	time / epoch	WER [%]	
			dev	eval
1 layer	6.7M	0:28h	17.6	22.7
2 layer	12.7M	1:00h	14.6	18.8
3 layer	18.7M	1:11h	14.0	18.4
4 layer	24.7M	1:33h	13.5	17.7
5 layer	30.7M	1:48h	13.6	17.7
6 layer	36.7M	2:10h	13.5	17.5
7 layer	42.7M	2:36h	13.8	18.0
8 layer	48.7M	3:14h	14.2	18.4
best MLP (9x2000)	42.7M	0:35h	15.3	20.3
best GMM	31.3M	–	23.6	30.2

missing experiments: discriminative sequence training for MLP and LSTM-RNN

hybrid approach:

replace emission probability of an hidden Markov model by ANN output

three types of emission models in HMMs:

- GMM: Gaussian mixture model
- MLP: deep multi-layer perceptron
- LSTM RNN: recurrent neural network with long short-term memory

experimental results for QUAERO English 2011:

approach	layers	WER[%]
conventional: best GMM	–	30.2
hybrid: best MLP	9	20.3
hybrid: best LSTM RNN	6	17.5

remarks:

- comparative evaluations in QUAERO 2011:
competitive results with LIMSI Paris and KIT Karlsruhe
- best improvement over Gaussian mixture models
by 40% relative using an LSTM RNN

Systematic Experiments on QUAERO English Eval 2013 (Tueske 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		Language Model	
Model	Criterion	Count	Count+ANN
		PP=131.1	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
	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: 60% relative

Tasks: Switchboard and Call Home

- conversational speech: telephone speech, narrow band;
challenging task: initial WER: 60% 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 from Switchboard studies (SWB)
 - 20 telephone conversations from Call-Home US English Speech
 - 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

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.

training data:

- audio : 960 hrs
- text: 800 million words

word error rates[%]:

team	approach	dev		test	
		1st half	2nd half	1st half	2nd half
Irie, Zeyer et al. RWTH 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 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

- there has been life before ANN and deep learning:
Bayes decision rule, Gaussian modelling, HMMs, discriminative training, ...
- acoustic modelling:
 - best improvements in WER by ANN over Gaussian models: around 50% relative
 - attention modelling: competitive (?)
- language modelling (more details: next class):
best improvements in WER by ANN : around 5-10% relative
- ANNs form *just one of many approaches*
in machine learning and pattern recognition

10.1 Why Language Modelling?

definition of language model (LM):

$p(w_1^N)$: (prior) probability of the word sequence $w_1^N := w_1 \dots w_n \dots w_N$

reason for language modelling: Bayes decision rule in ASR (and SMT!):

$$x_1^T \rightarrow \hat{w}_1^N(x_1^T) = \operatorname{argmax}_{N, w_1^N} \left\{ p(w_1^N) \cdot p(x_1^T | w_1^N) \right\}$$

observations about the language model $p(w_1^N)$:

- it can be learned from text only (unlabeled data!)
- it can improve performance dramatically

question:

How to measure the quality of an LM (without a recognition experiment)?

considerations:

- use prior $p(w_1^N)$ 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:

$$PP := \left(p(w_1^N) \right)^{-1/N} = \left(\prod_{n=1}^N p(w_n | w_0^{n-1}) \right)^{-1/N}$$

$$\log PP = -\frac{1}{N} \cdot \sum_{n=1}^N \log p(w_n | w_0^{n-1})$$

with artificial start symbol w_0

prior probability $p(w_1^N)$ of any sentence $w_1^N = w_1 \dots w_n \dots w_N$:

$$p(w_1^N) = \prod_{n=1}^N p(w_n | w_1^{n-1}) = \prod_{n=1}^N p(w_n | w_{n-2}, w_{n-1})$$

disambiguation of homophones (e.g. by a word trigram model):

- Homophones: **two, to, too**

Twenty-**two** people are **too** many **to** be put in this room.

- Homophones: **right, write, Wright**

Please **write** to Mrs. **Wright** **right** away.

conventional approach:

- assume Markov chain of order k :
limit the dependence on the full history w_0^{n-1} to the immediate k predecessor words:

$$p(w_n | w_0^{n-1}) := p_\vartheta(w_n | w_{n-k}^{n-1})$$

terminology: $(k + 1)$ -gram, e.g. fourgram, trigram, bigram, unigram

- free parameters ϑ to be learned from training data:
conditional probabilities $p_\vartheta(w_n | w_{n-k}^{n-1})$ for the $(k + 1)$ -gram events
- natural training criterion for a corpus w_1^N : minimum perplexity

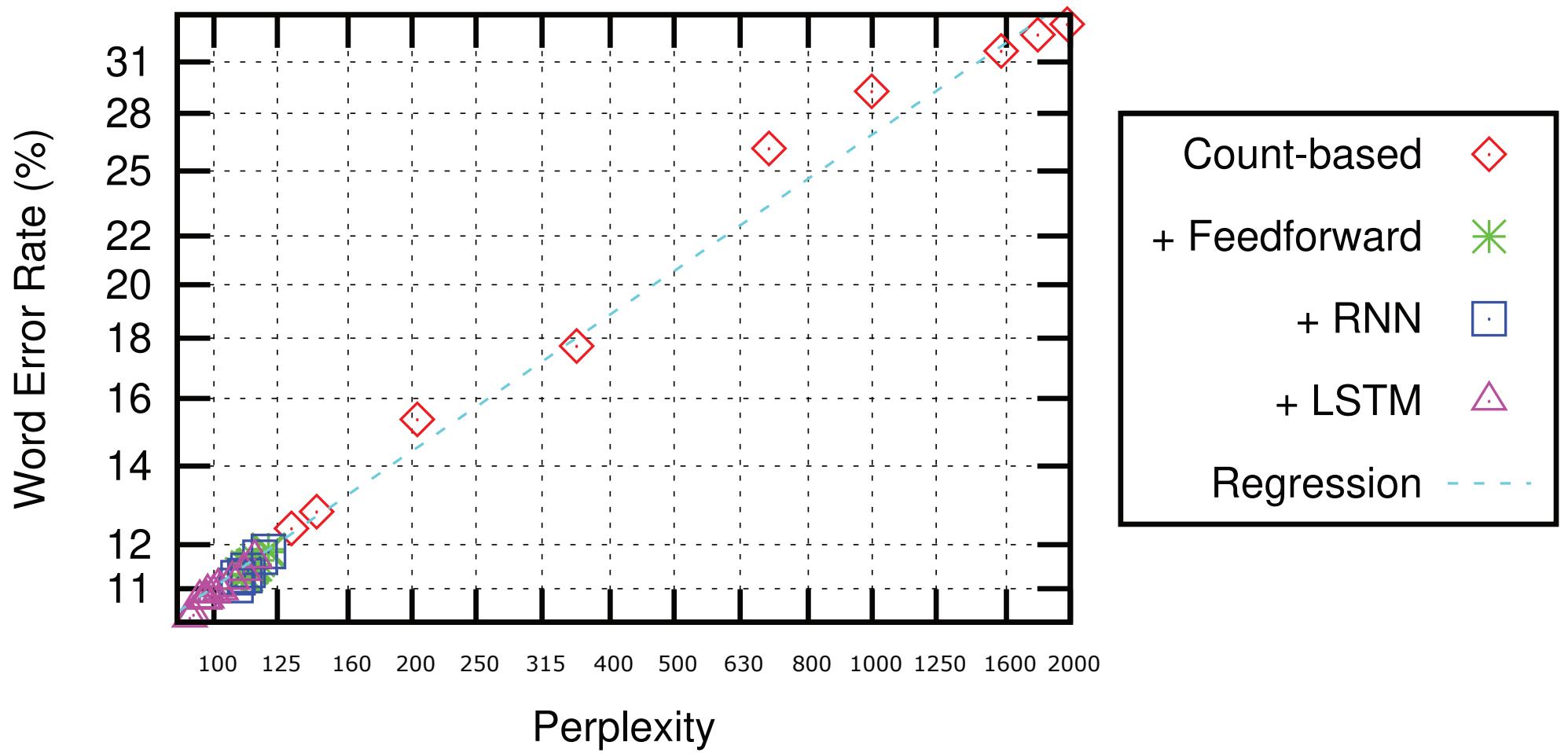
$$\max_{\vartheta} \left\{ \sum_{n=1}^N \log p_\vartheta(w_n | w_{n-k}^{n-1}) \right\}$$

- equivalent to cross-entropy training (or maximum likelihood)
- resulting estimates: relative frequencies based on event counts

- problem: most of the events are never seen in training data
 - example: vocabulary of $100k = 10^5$ words results in 10^{15} possible trigrams
 - result: virtually all event counts are zero
- remedy: interpolation/combination of LMs of various orders k ,
e.g. fivegrams, fourgram, trigram, bigram and unigram events
- various strategies
(Turing-Good estimation, Katz smoothing, Kneser-Ney smoothing, ...):
 - models: linear discounting, absolute discounting, back-off, ...
 - estimation: cross-validation or leave-one-out
 - concept of generalized marginal distributions, e.g going from trigrams to bigrams
- most strategies implemented in LM toolkit by SRI

Preview: Word Error Rate vs. Perplexity

empirical law: $WER = \alpha \cdot PP^\beta$



word representation: from discrete symbols to vectors:

- **motivation:**

**replace symbolic event (atomic entities)
by high-dimensional continuous-valued vector**

- **advantages:**

- **vector calculations and learning/estimation**
- **better generalization**

history:

- **1971 Salton: information retrieval using term-document matrix**
- **1993 Schütze & Peterson: co-occurrence of two words**
- **2006 Bengio, Schwenk et al.: ANN based language modelling**
- **2013 Mikolov et al.: specific approaches to language modelling:
skip bigrams, bag-of words**

applications in NLP:

- **document classification (similar: information retrieval):**
co-occurrence of a document and a word
- **language modelling:**
 - two adjacent words
 - two words co-occurring within a window (or document)
- **machine translation:**
co-occurrence of source-target word in a source-target sentence pair
- ...

Word Vectors: Asymmetric Version (Unpublished Work)

example: bigram language model $p(w|v)$

asymmetric case: different mappings for words and predecessor words:

- **assumptions:**

each word w is represented by a normalized vector r_w and a prior $\alpha_w \geq 0$:

$$w \rightarrow r_w \in \mathbb{R}^D \quad \|r_w\|^2 = 1 \quad \alpha_w \geq 0$$

each predecessor v is represented by a normalized vector s_v and a prior $\beta_v \geq 0$:

$$v \rightarrow s_v \in \mathbb{R}^D \quad \|s_v\|^2 = 1 \quad \beta_v \geq 0$$

- **model of conditional bigram probability $p(w|v)$:**

$$\begin{aligned} p(w|v) &= 1/Z'_v \cdot \alpha_w \cdot \beta_v \cdot \exp(-1/2 \cdot \|r_w - s_v\|^2) \\ &= 1/Z_v \cdot \alpha_w \cdot \exp(r_w^T \cdot s_v) \end{aligned}$$

$$Z_v := Z_v(\{\alpha_w, r_w\}, s_v) = \sum_w \alpha_w \cdot \exp(r_w^T \cdot s_v)$$

remarks:

- predecessor prior β_v cancels
- normalization of vectors r_w and s_v will be essential for SVD solution

Word Vectors: Asymmetric Version

- consider the probability of a (very long) word sequence $w_1^N = w_1, \dots, w_n, \dots, w_N$:

$$\begin{aligned} \log \prod_n p(w_n | w_{n-1}) &= \sum_n \log p(w_n | w_{n-1}) = \sum_{v,w} N(v, w) \log p(w|v) \\ &= \sum_w N(\cdot, w) \log \alpha_w + \sum_{v,w} N(v, w) s_v^T \cdot r_w - \sum_v N(v, \cdot) \log Z_v \end{aligned}$$

with asymmetric counts $N(v, w)$ of word bigrams (v, w)

- assuming a weak dependence of $\sum_v N(v, \cdot) \log Z_v$ on $\{\alpha_w, r_w, s_v\}$, we have the (approximate) maximum likelihood criterion:

$$\operatorname{argmax}_{\{\alpha_w, r_w, s_v\}} \left\{ \sum_w N(\cdot, w) \log \alpha_w + \sum_{v,w} N(v, w) s_v^T \cdot r_w \right\}$$

assuming a normalized prior $\sum_w \alpha_w = 1$, we have the solutions:

$$\alpha_w = N(\cdot, w) / N(\cdot, \cdot)$$

r_w, s_v : vectors from singular value decomposition (SVD) of $N(v, w)$

note: in this approximation, the SVD vectors do not depend on the prior α_w !

Word Vectors: Symmetric Version (Unpublished Work)

**example: probability $p(v, w)$ of a (distant) word pair (v, w) ,
i. e. co-occurrence of two words v and w within a window**

- **assumptions:**

each word w (or v) is represented by a normalized vector r_w and a prior $\alpha_w \geq 0$:

$$w \rightarrow r_w \in \mathbb{R}^D \quad \|r_w\|^2 = 1 \quad \alpha_w \geq 0$$

- **probability model $p(v, w)$ of word co-occurrence:**

$$\begin{aligned} p(v, w) &= 1/Z' \cdot \alpha_v \cdot \alpha_w \cdot \exp(-1/2 \cdot \|r_w - r_v\|^2) \\ &= 1/Z \cdot \alpha_v \cdot \alpha_w \cdot \exp(r_w^T \cdot r_v) \\ Z := Z(\{\alpha_w, r_w\}) &= \sum_{v,w} \alpha_v \cdot \alpha_w \cdot \exp(r_w^T \cdot r_v) \end{aligned}$$

remark:

normalization of vector r_w will be essential for eigenvector solution

- consider a (very long) sequence of symmetric pairs $(v_n, w_n), n = 1, \dots, N$:

$$\begin{aligned} \log \prod_n p(v_n, w_n) &= \sum_n \log p(v_n, w_n) = \sum_{v,w} N(v, w) \log p(v, w) \\ &= 2 \sum_w N(w) \log \alpha_w + \sum_{v,w} N(v, w) r_v^T \cdot r_w - N \log Z \end{aligned}$$

with symmetric counts $N(v, w)$ of word pairs (v, w)

- assuming a weak dependence of Z on $\{\alpha_w, r_w\}$,
we have the (approximate) maximum probability criterion:

$$\operatorname{argmax}_{\{\alpha_w, r_w\}} \left\{ 2 \sum_w N(w) \log \alpha_w + \sum_{v,w} N(v, w) r_v^T \cdot r_w \right\}$$

assuming a normalized prior $\sum_w \alpha_w = 1$, we have the solutions:

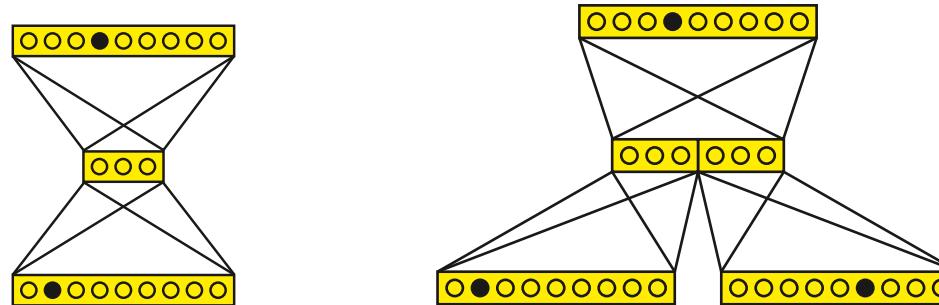
$$\alpha_w = N(w)/N$$

r_w : eigenvectors of $N(v, w)$

note: in this approximation, the eigenvectors r_w do not depend on the prior α_w !

From Word Vectors to Neural Net Language Models: with No Hidden Layers

no hidden layer of weights:



approach for bigram language model:

- output layer: a score $p(w|u)$ for each word w
with vector r_w and representation vector s_v of predecessor word:
we drop the normalization of the word vectors r_w :

$$\begin{aligned} p(w|v) &= 1/Z'_v \cdot \alpha_w \beta_v \cdot \exp(-1/2 \cdot \|r_w - s_v\|^2) \\ &= 1/Z_v \cdot \exp(r_w^T \cdot s_v - 1/2 \cdot \|r_w\|^2 + \log \alpha_w) \\ &= 1/Z_v \cdot \exp(r_w^T \cdot s_v + \lambda_w) \end{aligned}$$

in other words: dot product with bias term and softmax operation

- input layer:
 - input word $v = 1, \dots, V$: **1-of-V encoding**: $\tilde{v} := [0, 0, \dots, 0, 1, 0, \dots, 0]$
 - word vectors $s_v \in \mathbb{R}^D$ form columns of a projection (representation) matrix $S \in \mathbb{R}^{D \cdot V}$

$$S : v \rightarrow s_v = S \cdot \tilde{v}$$

matrix-vector product:

$$\begin{aligned} S : \mathcal{W} &\rightarrow \mathbb{R}^D \\ v &\rightarrow s_v = S \cdot \tilde{v} \end{aligned}$$

with dimensions ($V := |\mathcal{W}|$): $D \ll V, S \in \mathbb{R}^{D \times V}$

detailed notation:

$$\begin{bmatrix} s_{1,v} \\ s_{2,v} \\ \vdots \\ s_{D,v} \end{bmatrix} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,v-1} & s_{1,v} & s_{1,v+1} & \cdots & s_{1,V} \\ s_{2,1} & \cdots & s_{2,v-1} & s_{2,v} & s_{2,v+1} & \cdots & s_{2,V} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_{D,1} & \cdots & s_{D,v-1} & s_{D,v} & s_{D,v+1} & \cdots & s_{D,V} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- projection of word v : select column v of projection matrix S
- might already include the sigmoid operation

- so far no ANNs: closed-form solutions, but only for approximative training criterion
- using ANNs: exact training criterion with numerical solutions (no closed form)

approach:

- output layer: a score $p(w|u)$ for each word w
with vector r_w and representation vector u of preceding (hidden) layer:
we drop the normalization of the word vectors r_w :

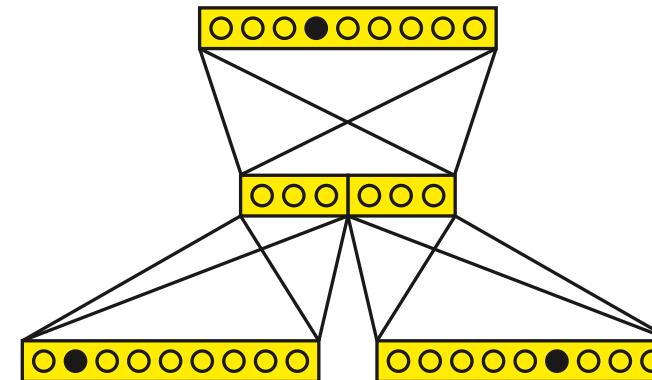
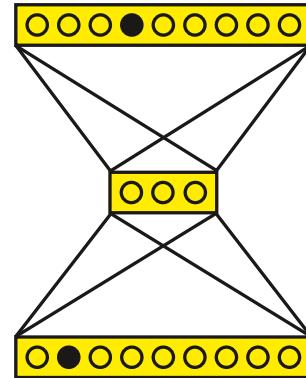
$$\begin{aligned} p(w|u) &= 1/Z'_u \cdot \alpha_w \beta_u \cdot \exp(-1/2 \cdot \|r_w - u\|^2) \\ &= 1/Z_u \cdot \exp(r_w^T \cdot u - 1/2 \cdot \|r_w\|^2 + \log \alpha_w) \end{aligned}$$

in other words: dot product with bias term and softmax operation

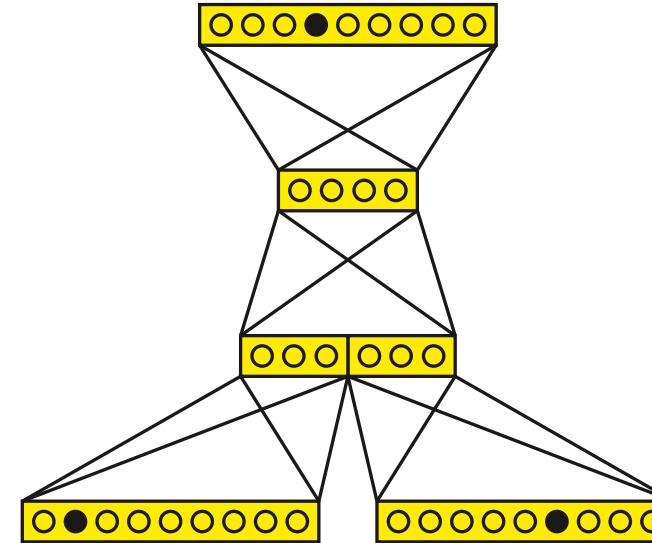
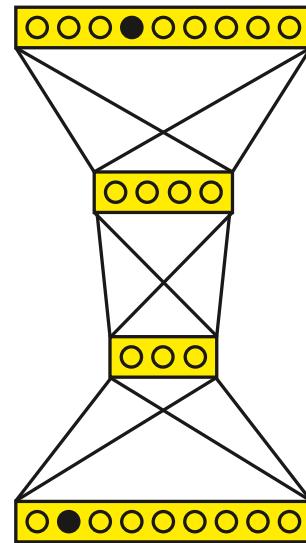
- several hidden layers with dot product operations:
bias term and nonlinearity (sigmoid)
- input layer:
 - input word $v = 1, \dots, V$: 1-of-V encoding: $\tilde{v} := [0, 0, \dots, 0, 1, 0, \dots, 0]$
 - word vectors $s_v \in \mathbb{R}^D$ form columns of a projection (representation) matrix $S \in \mathbb{R}^{D \times V}$

$$S : v \rightarrow s_v = S \cdot \tilde{v}$$

no hidden layer of weights:



one hidden layer of weights:



- normalization constraint:

$$r_w = \|\mathbf{r}_w\| \cdot \frac{\mathbf{r}_w}{\|\mathbf{r}_w\|} = \lambda_w \cdot \tilde{\mathbf{r}}_w \quad \text{with} \quad \lambda_w > 0$$

- output layer: a score $p(w|u)$ for each word w
with vector r_w and representation vector u ('features') of preceding (hidden) layer:

$$p(w|u) = 1/Z_u \cdot \exp(r_w^T \cdot u - 1/2 \cdot \|\mathbf{r}_w\|^2 + \log \alpha_w) = 1/Z_u \cdot \exp(\lambda_w \tilde{\mathbf{r}}_w^T \cdot u)$$

with bias term included in dot product (abusing notation)

- consider M -gram language model $p(w|h)$ with history $h = v_1^M = v_1 \dots v_m \dots v_M$ and associated feature vectors $u_1^M = u_1 \dots u_m \dots u_M$:

$$p(w|v_1^M) = 1/Z \cdot \exp\left(\sum_m \lambda_m \tilde{\mathbf{r}}_w^T \cdot u_m\right) = 1/Z \cdot \exp\left(\tilde{\mathbf{r}}_w^T \cdot \sum_m \lambda_m u_m\right)$$

- applications: ?
 - weighted decay coefficients λ_m
 - different projection matrices for $m = 1, \dots, M$

definition (information theory): mutual information of a pair of discrete random variables $(x, y) \in (X, Y)$ with (true) joint probability $p(x, y)$:

$$\begin{aligned} I(X, Y) &= \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x) \cdot p(y)} \\ &= \sum_{x,y} p(x, y) \log p(x, y) - \sum_x p(x) \log p(x) - \sum_y p(y) \log p(y) \\ &\geq 0 \end{aligned}$$

related concepts: entropy, cross-entropy and Kullback-Leibler divergence

(tentative) analogy of

- *local* mutual information between random variable pair (x, y)
- and the squared Euclidean distance of the vector representations (r_x, r_y) :

$$\begin{aligned} \log \frac{p(x, y)}{p(x) \cdot p(y)} &= \log p(x, y) - \log p(x) - \log p(y) \\ -\|r_x - r_y\|^2 &= 2 r_x^T \cdot r_y - \|r_x\|^2 - \|r_y\|^2 \end{aligned}$$

motivation and justification:

information beyond statistically independent variables x and y

Relation to Mutual Information

starting point: analogy with Euclidean distance:

$$\log \frac{p(x, y)}{p(x) \cdot p(y)} = \log p(x, y) - \log p(x) - \log p(y)$$

$$-1/2 \|r_x - r_y\|^2 = r_x^T \cdot r_y - 1/2 \|r_x\|^2 - 1/2 \|r_y\|^2$$

tentative approach including normalization factor Z :

$$p(x, y) = 1/Z \cdot p(x) p(y) \cdot \exp(-1/2 \|r_x - r_y\|^2)$$

$$Z := \sum_{x,y} p(x) p(y) \cdot \exp(-1/2 \|r_x - r_y\|^2)$$

$$\log \frac{p(x, y)}{p(x) \cdot p(y)} = -1/2 \|r_x - r_y\|^2 - \log Z$$

note the complications:

- additive term in comparison with Euclidean distance
- marginal constraints for consistent modelling, e.g. for $p(y) = \sum_x p(x, y)$:

$$Z = \sum_x p(x) \cdot \exp(-1/2 \|r_x - r_y\|^2) \quad \forall y$$

difficult to handle and thus often dropped in practice

starting point: analogy with Euclidean distance:

$$\frac{\mathbf{r}_x^T \cdot \mathbf{r}_y}{\|\mathbf{r}_x\| \cdot \|\mathbf{r}_y\|} = 1 - \frac{1}{2} \left\| \frac{\mathbf{r}_x}{\|\mathbf{r}_x\|} - \frac{\mathbf{r}_y}{\|\mathbf{r}_y\|} \right\|^2$$

$$\log \frac{\mathbf{r}_x^T \cdot \mathbf{r}_y}{\|\mathbf{r}_x\| \cdot \|\mathbf{r}_y\|} \approx -\frac{1}{2} \left\| \frac{\mathbf{r}_x}{\|\mathbf{r}_x\|} - \frac{\mathbf{r}_y}{\|\mathbf{r}_y\|} \right\|^2$$

and re-normalization

we try the approach:

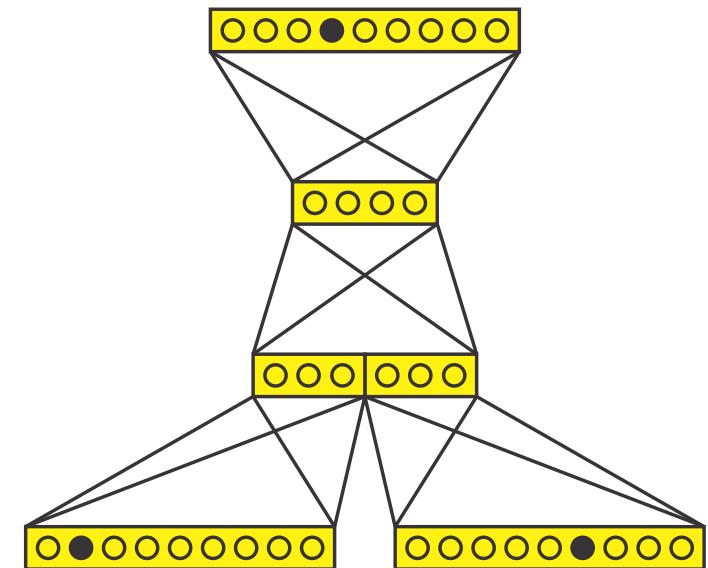
$$\log \frac{p(x, y)}{p(x) \cdot p(y)} = -\frac{1}{2} \left\| \frac{\mathbf{r}_x}{\|\mathbf{r}_x\|} - \frac{\mathbf{r}_y}{\|\mathbf{r}_y\|} \right\|^2 - \log Z$$

- mutual information: nice as a motivation
- possible additional constraint: normalization of vectors
result: dot product as in ANNs
- actual model in practice:
exponential model with Euclidean distance or dot product
 - output: softmax operation
 - input: projection matrix
 - exact training criterion (log of posterior probability)
 - numerical optimization, e. g. backpropagation
- ...

10.3 Neural Networks for LM

overview: approaches to modelling $p(w_n|w_0^{n-1})$:

- non ANN: count models (Markov chain):
 - limited history w_0^{n-1} of k predecessor words
 - smooth relative frequencies
- feedforward multi-layer perceptron (FF-MLP):
 - limited history too
 - use predecessor words as input to MLP
- recurrent neural networks (RNN):
advantage: unlimited history



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.

usual approach for vocabulary representation:

- **input layer:** each node stands for a word w of the vocabulary.
 - interpretation: vector $\tilde{w} \in \mathbb{R}^V$
 - terminology: 1-of- V encoding, one-hot representation

example (vocabulary in alphabetic order):

$$\text{vocabulary} = \{\text{are}, \text{doing}, \text{how}, \text{old}, \text{you}\}$$

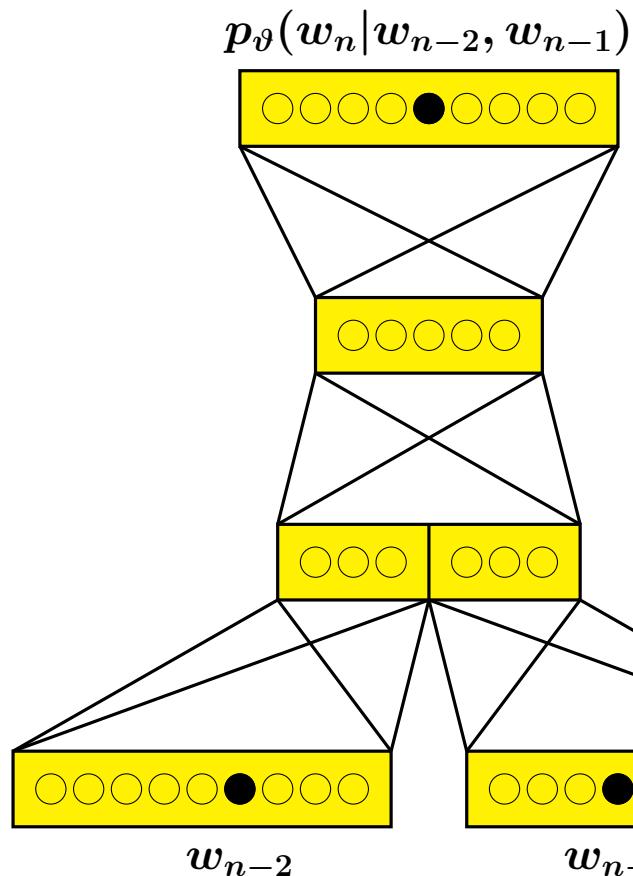
$$\text{how} = [\quad 0, \quad 0, \quad 1, \quad 0, \quad 0 \quad]$$

$$\text{you} = [\quad 0, \quad 0, \quad 0, \quad 0, \quad 1 \quad]$$

$$\text{doing} = [\quad 0, \quad 1, \quad 0, \quad 0, \quad 0 \quad]$$

- **output layer:**
 - each node stands for a word w of the vocabulary
 - softmax normalization results in conditional probability: $p_\vartheta(w|\cdot)$

note: multiple notation conventions: $w_1^N = w_1 \dots w_n \dots w_N$ and $w_1^T = w_1 \dots w_t \dots w_T$



- Softmax activation function is applied to obtain the word posterior probability $p_\theta(w_n | w_{n-2}, w_{n-1})$
- ↑
- Sigmoid activation function is applied element-wise to get hidden layer activation
- ↑
- 1-of- V encoded predecessor words w_{n-2} and w_{n-1} are multiplied by a weight matrix and then concatenated to form the projection layer activation

conditional probability $p_\vartheta(w_t|w_{t-k}^{t-1})$ of the $(k+1)$ -gram language model:

- **Input: k predecessor words $w_{t-k}^{t-1} := w_{t-k}, \dots, w_{t-2}, w_{t-1}$ with 1-of- V coding:**

$$\tilde{w}_{t-k}, \dots, \tilde{w}_{t-2}, \tilde{w}_{t-1} \in \mathbb{R}^V$$

- **Projection layer with matrix A_1 :**

- idea: dimension reduction $\mathbb{R}^V \rightarrow \mathbb{R}^d$ (e.g. from $V = 150k$ to $d = 600$)
- a linear operation (matrix multiplication) without sigmoid activation
- shared across all predecessor words w_{t-k}^{t-1}

$$z_t = [A_1 \tilde{w}_{t-k}, \dots, A_1 \tilde{w}_{t-2}, A_1 \tilde{w}_{t-1}]$$

- **(Second) hidden layer with matrix A_2 :**

$$h_t = \sigma(A_2 z_t) \quad \text{with the sigmoid function } \sigma(\cdot)$$

- **Output layer with matrix A_3 and output vector $y_t \in \mathbb{R}^V$:**

$$\begin{aligned} y_t = p_\vartheta(w_t|w_{t-k}^{t-1}) &= S(A_3 h_t) \quad \text{with the softmax function } S(\cdot) \\ &= S(A_3 \sigma(A_2 [A_1 \tilde{w}_{t-k}, \dots, A_1 \tilde{w}_{t-2}, A_1 \tilde{w}_{t-1}])) \end{aligned}$$

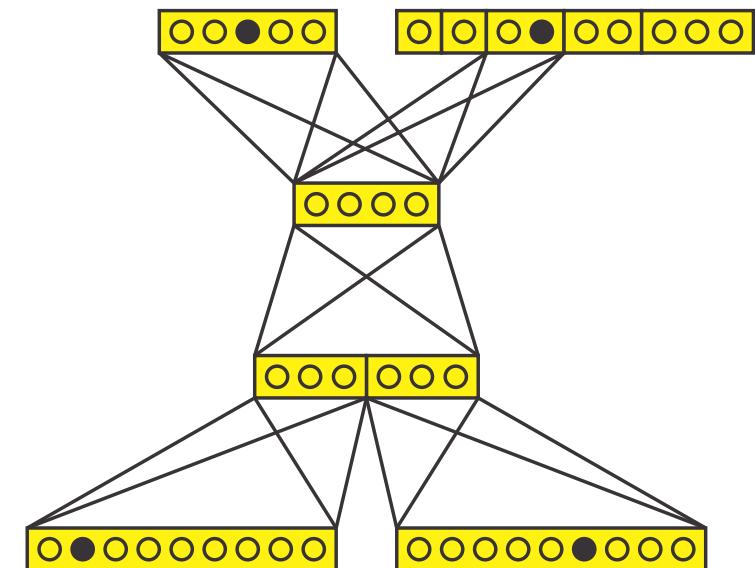
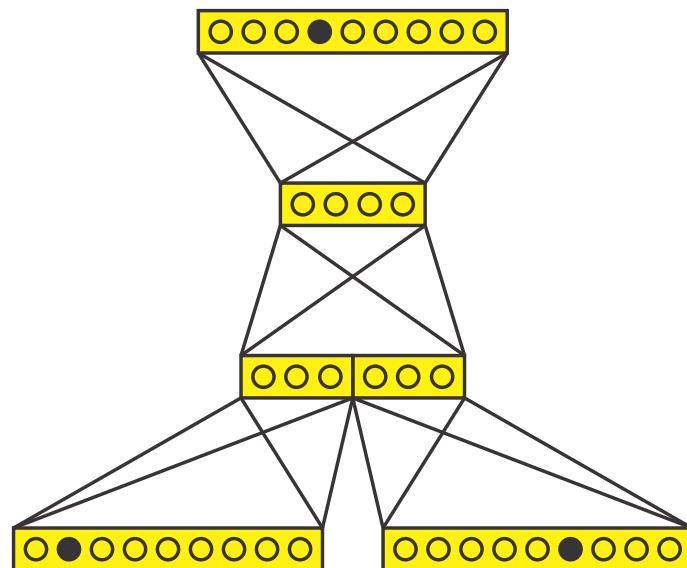
Why word classes?

- softmax operation is computationally expensive (sum over full vocabulary)
- remedy: word classes that are automatically trained

conditional language model probability $p(w_t|w_{n-k}^{n-1})$ for each history w_{n-k}^{n-1} :

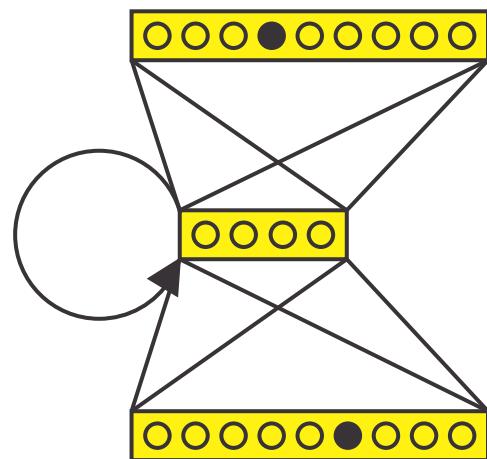
$$p_\theta(w_n|w_{n-k}^{n-1}) = p_\theta(g_n|w_{n-k}^{n-1}) \cdot p_\theta(w_n|g_n, w_{n-k}^{n-1})$$

using a unique word class $g_n = g(w_n)$ for each word w_n

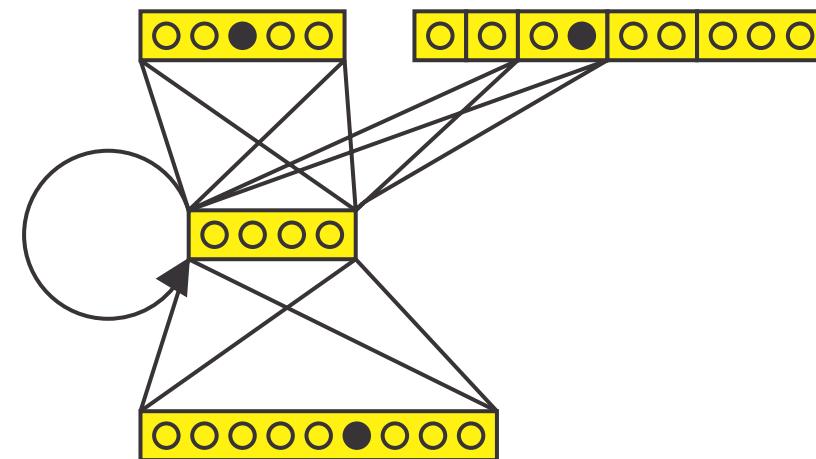


note: projection layer is not shown.

output layer with no word classes



output layer with word classes



extension: LSTM

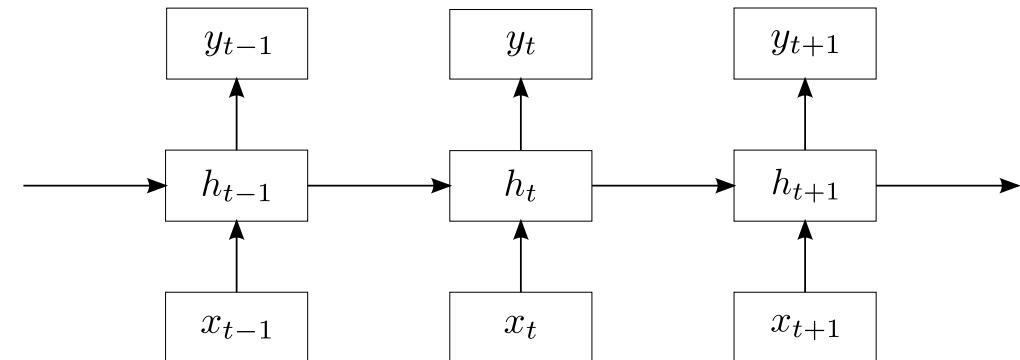
remember: RNN = ANN with memory for sequence processing

input to RNN for position/time t :

(single) predecessor word w_{t-1} using 1-of- V coding: \tilde{w}_{t-1}

operations from bottom to top:

- **output vector:** $y_t = p_\vartheta(w_t | w_0^{t-1}) = S(A_3 h_t)$
- **second hidden layer:** $h_t = \sigma(A_2 z_t + R h_{t-1})$
- **projection layer:** $z_t = A_1 \tilde{w}_{t-1}$
- **for LM prediction:** $x_t = y_{t-1}$



interpretation: conditional probability of RNN LM:

$$p_\vartheta(w_t | w_0^{t-1}) = p_\vartheta(w_t | h_{t-1}, w_{t-1}) = p_\vartheta(w_t | h_t)$$

with memory states h_t that group all histories w_1^{t-1} into equivalence classes

- results on QUAERO English:
 - vocabulary size: 128k words
 - training text: 50M words
 - development corpus: 39k words
 - evaluation corpus: 35k words
- structure of MLP:
 - projection layer: 2700 nodes (300 per predecessor word)
 - hidden layer: 600 nodes
 - size of MLP is dominated by input and output layers:
$$128k \cdot 300 + 600 \cdot 128k = 115M$$
- perplexity (PP) on development data

Approach	PP
4-gram count model	163.7
10-gram MLP	136.5
10-gram MLP with 2 layers	130.9

structure of recurrent neural nets (RNN):

- projection and hidden layer: each 600 nodes
- size of RNN is dominated by input and output layers (like for MLP):

$$128k \cdot 600 + 600 \cdot 128k = 154M$$

Perplexity PP of four approaches on dev data:

Approach	PP
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

observation: (huge) improvement by 40%

Training times (without GPUs!) for training corpus of 50 Million words:

Models	PP	CPU Time (Order)
Count model	163.7	30 min
MLP	136.5	1 week
LSTM-RNN	107.8	3 weeks

- problem: high computation times
- remedy: two types of language models:
 - count model: trained on a huge corpus: 3.1 Billion words
 - ANN models: trained on a small corpus: 50 Million words
- resulting language model used in ASR experiments:
linear interpolation of TWO models

The number of parameters for a LSTM-RNN LM with

- a vocabulary of 128k words
- projection layer of size 600
- 2 LSTM-RNN layers of size 600 each
- 1000 word classes to factorize the output layer

is computed as follows:

$$\begin{aligned} & 128.000 \times 600 && \text{projection layer} \\ & + 2 \times 600 \times 600 \times 8 && \text{2 LSTM layers (8 matrices per layer for each: 2 per gates} \times 3 + 2 \text{ for input)} \\ & + 600 \times 1000 && \text{output layer: class prediction part} \\ & + 600 \times 128.000 && \text{output layer: word prediction part} \\ & = 159.960.000 \\ & = 160M && (\text{bias and peephole weights can be neglected here}) \end{aligned}$$

while the training data consist of $50M$ running words !

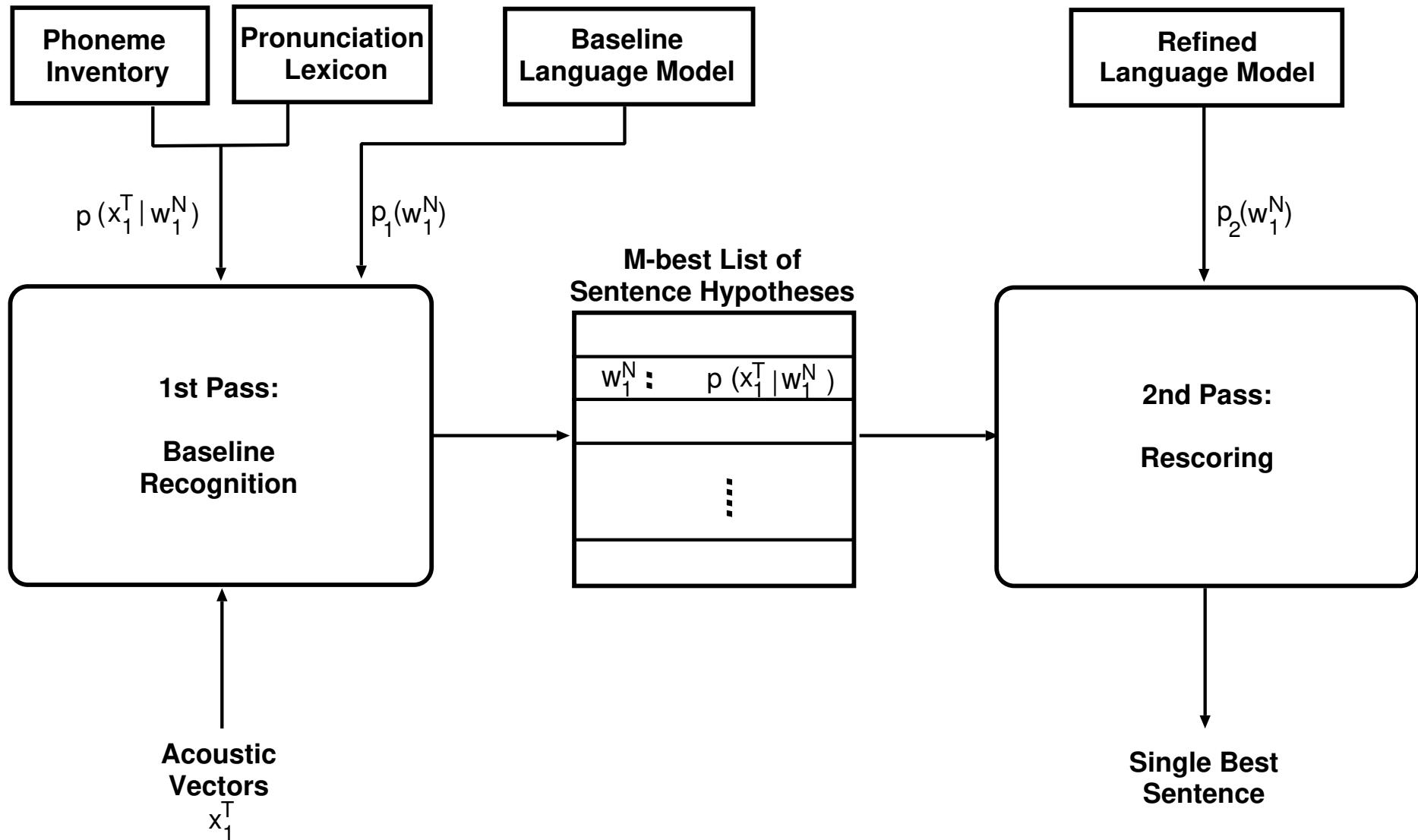
important result: there is NO data reduction!

- linear interpolation of TWO models: count model + ANN model
- recognition experiments:
due to unlimited history, RNN LMs require re-design of ASR search
- 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

- experimental result:
 - significant improvements by ANN language models
 - best improvement in perplexity: 30% reduction (from 131 to 92)
 - empirical observation:
power law between perplexity and WER (cube to square root)

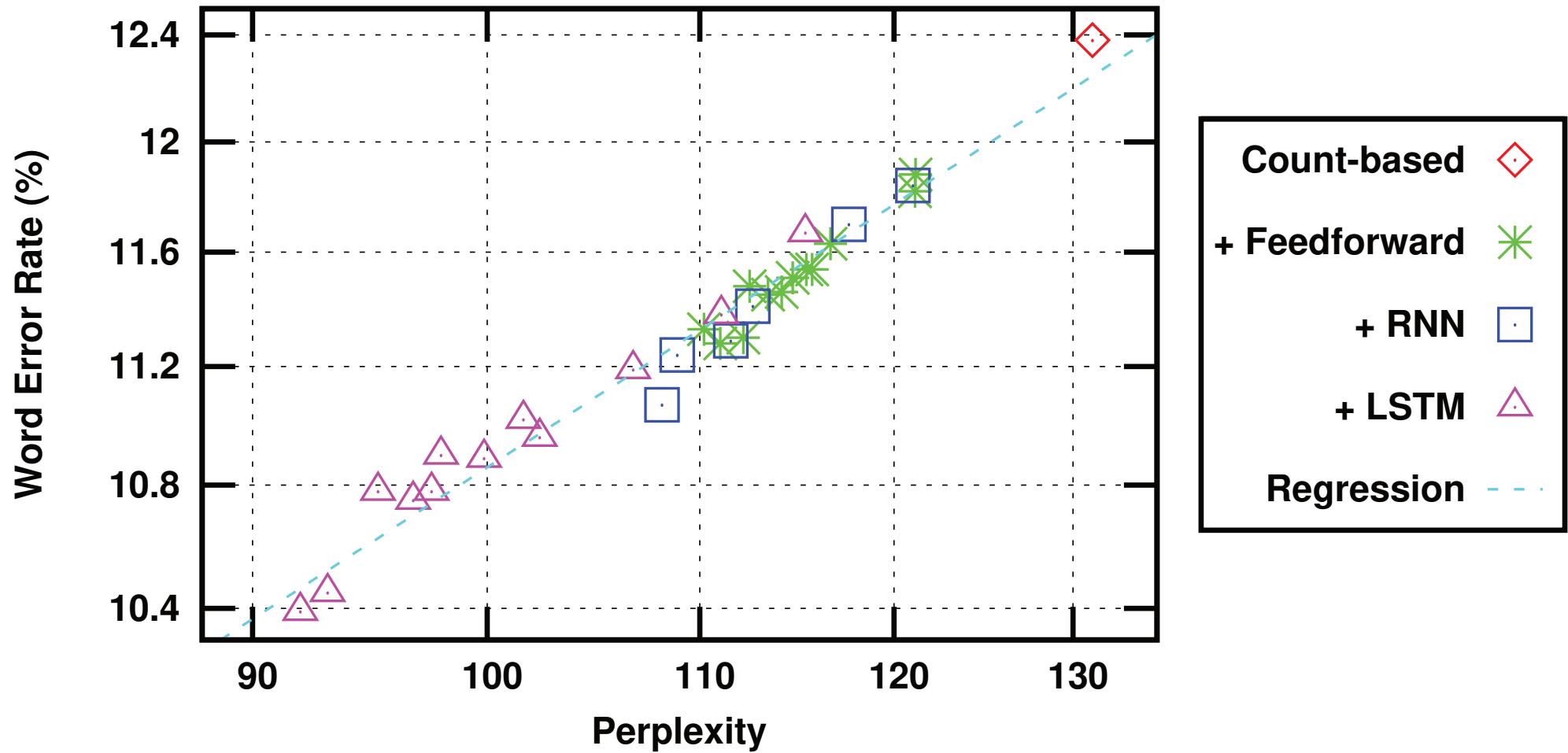
Re-design of ASR Search: Two-Pass Strategy (Rescoring: N -Best List or Word Graph)



Plot: Perplexity vs. Word Error Rate

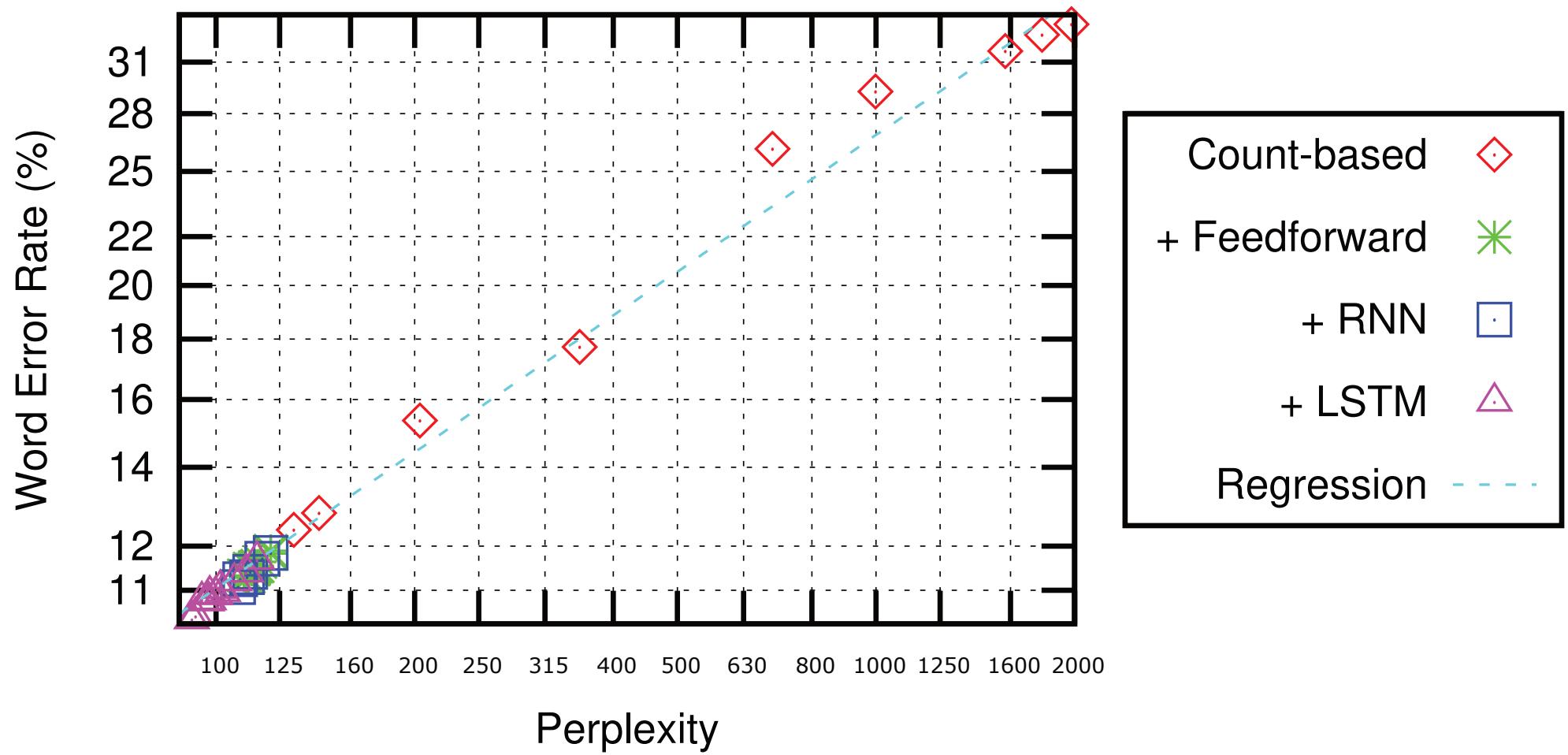
empirical law: $WER = \alpha \cdot PP^\beta$

[Makhoul & Schwartz 94, Klakow & Peters 02]



Extended Range: Perplexity vs. Word Error Rate

empirical law: $WER = \alpha \cdot P P^\beta$



- **design of language models based on neural networks**
- **three major architectures:**
feedforward MLP, recurrent NN, long short-term memory RNN
 - result in significant improvements over conventional models
 - computationally expensive (in training)
- **combination of count model and NNLM currently gives the best results**
(tradeoff between CPU time and perplexity)
- **software available at:**
www-i6.informatik.rwth-aachen.de/web/Software/rwthlm.php

Problem: Normalization in Output Layer

CPU problem: computing probabilities for the full vocabulary is very expensive.

Short List:

define a short-list of most frequent words (e.g. 10k):

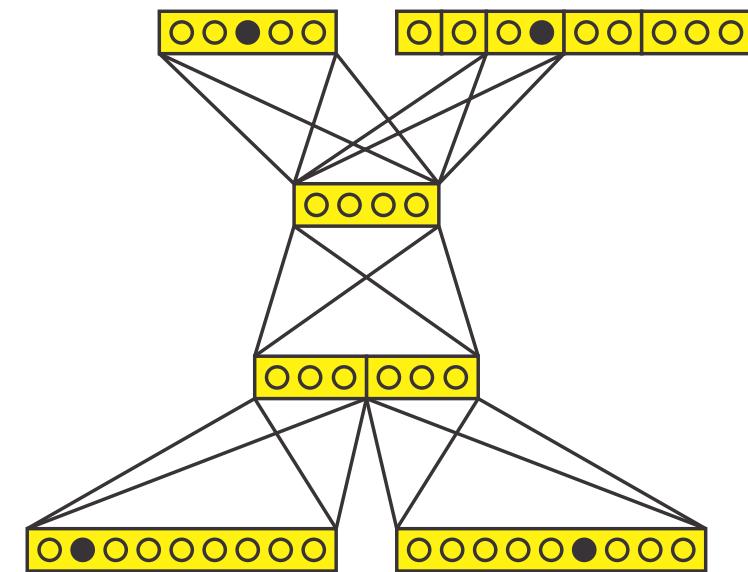
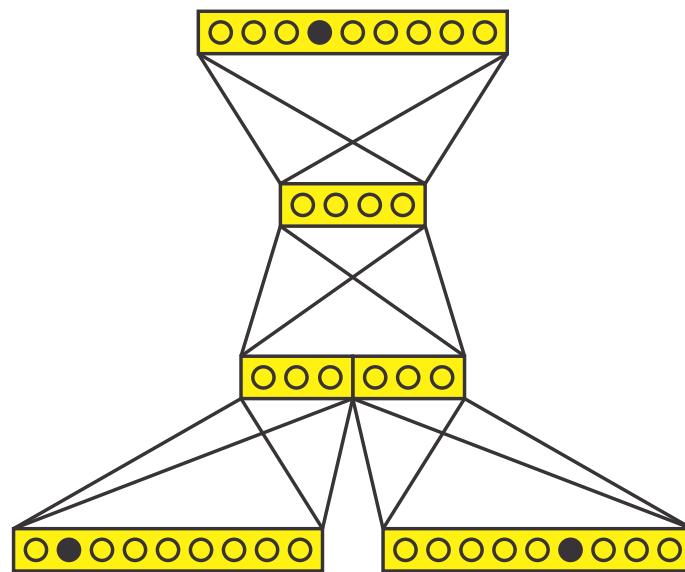
- probabilities of words in short-list are directly calculated by network
- probabilities of remaining words are obtained from back-off LM

Word Classes:

- cluster all vocabulary words into (unique) word classes using perplexity criterion and exchange algorithm [Martin & Ney 96] [Botros & Irie⁺ 15]
mapping $g(\cdot) : w \rightarrow g = g(w)$
- the output layer is split into two parts:
 - class prediction: $p_\vartheta(g_n | w_{n-k}^{n-1})$ with $g_n := g(w_n)$
 - word prediction: $p_\vartheta(w_n | g_n, w_{n-k}^{n-1})$
- complete language model:

$$p_\vartheta(w_n | w_{n-k}^{n-1}) = p_\vartheta(g_n | w_{n-k}^{n-1}) \cdot p_\vartheta(w_n | g_n, w_{n-k}^{n-1})$$

LM MLP without and with Word Classes



Output Normalization: Comparison of Approaches

experimental conditions for Penn Treebank Corpus (WSJ newspaper):

- LSTM RNN + count model
- vocabulary size: 10k words
- number of running words: 930k
- number of classes: 1000
- short list size: 8k
- number of nodes (optimized for batch size of 8): 200 to 300

Approach	PP LSTM	PP LSTM + count	Training Time[h]
full output	118.7	102.8	4.5
classes	119.3	102.4	1.0
short list	–	110.4	2.5

conclusion for vocabulary of 10k words;

no loss due to word classes, but reduction of CPU time by factor of 4

Output Normalization: Larger Corpus with 50M words

experimental conditions for QUAERO:

- 10-gram MLP (2 layers) and count model,
trained on 50M and 3B running words, respectively
- total vocabulary size: 150k words (128k words for NNLM, renormalized for interpolation)
- number of classes: 1000
- number of nodes: 300 per word for input (total of 2700) and 600 for each hidden layers

Approach	PP MLP	PP MLP + count	Training Time
full output	130.6	110.8	1 week GPU \approx 10 weeks CPU
classes	130.9	110.2	1 week CPU

conclusion for vocabulary of 150k words;
no loss due to word classes, but reduction of CPU time by factor of 10

Practicalities of NN LM training (implementation and software):

- sigmoid function
- no regularization, no momentum term, no drop-out (so far!)
- no pretraining (so far!)
- vocabulary reduction: remove singletons, or keep most frequent words
- random initialization of weights: Gaussian of mean 0, variance 0.01
- training criterion: cross-entropy (perplexity)
- stopping: cross-validation, perplexity on a development text
- initial learning rate: typically between $1 \cdot e^{-3}$ and $1 \cdot e^{-2}$
- learning rate: halved when the dev perplexity is worse than the best of previous epochs
- use of minibatches: 4 to 64
- low level implementation in C++
- GPUs: sometimes yes, sometimes no

public toolkit with source code: RWTH, Informatik 6 (i6), software

11.1 Overview of Translation Approaches

- **formalism: string-to-string conversion**
 $\text{source sentence} \rightarrow \text{target sentence}$
- **localization:**
 - not all source words interact with all target words
 - extreme case: bilingual pairs of (source, target) words
 - terminology: (probabilistic) alignment model
- **classical approach [Brown & Della Pietra⁺ 93, Vogel & Ney⁺ 96]:**
consider the alignment to be a hidden variable (as in HMM for ASR)
- **alternative (2015 Bengio's team in Montreal): attention model**
[Bahdanau & Cho⁺ 15]

translate source sentence F into a target sentence E :

$$F \rightarrow \hat{E}(F)$$

use Bayes decision rule (in the MAP approximation):

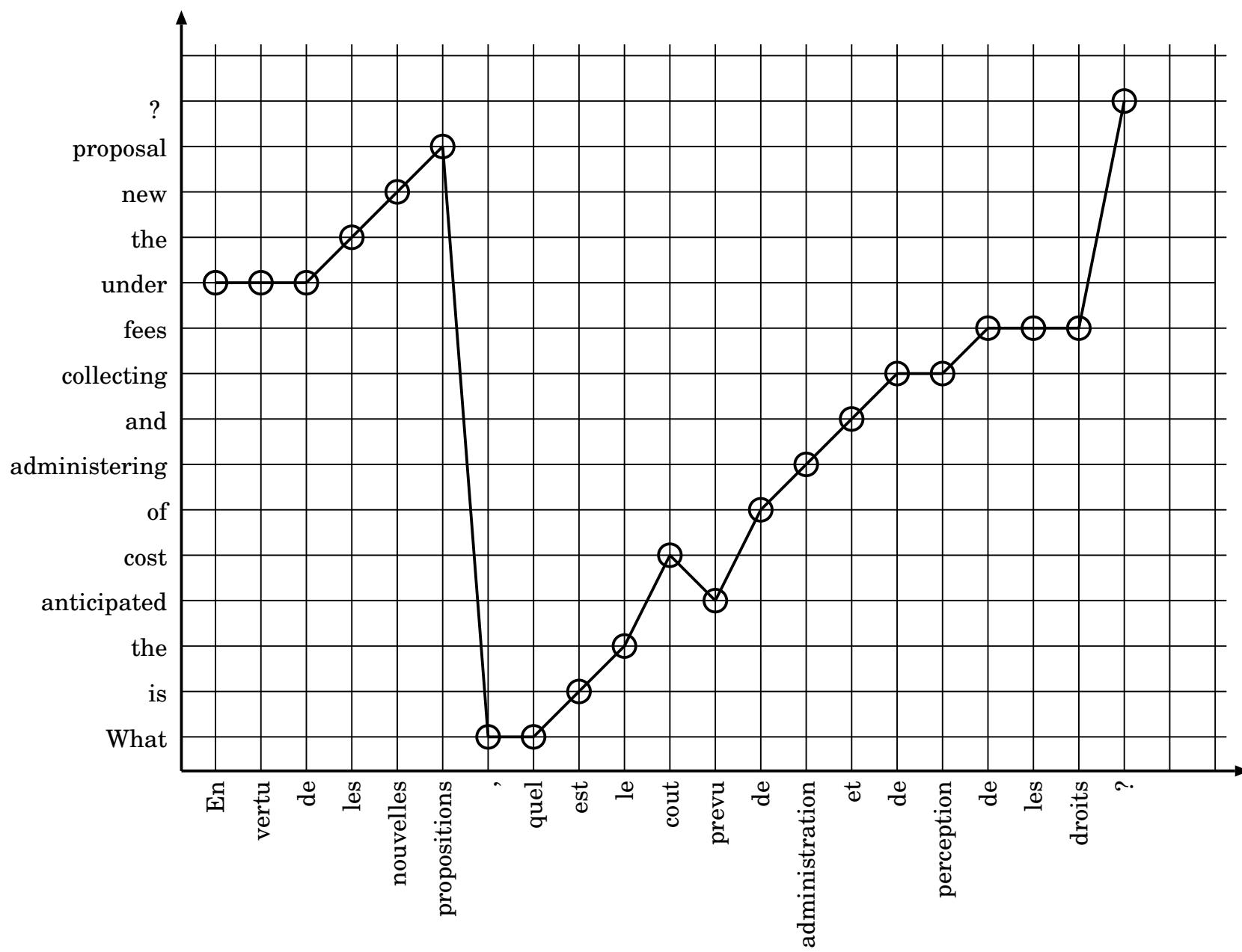
$$F \rightarrow \hat{E}(F) = \operatorname{argmax}_E \{pr(E|F)\} = \operatorname{argmax}_E \{pr(E, F)\}$$

generate approach: approximate the joint probability $pr(E, F)$ by a combination of language model, alignment model (with hidden alignments A) and lexicon model:

$$\begin{aligned} F \rightarrow \hat{E}(F) &= \operatorname{argmax}_E \{p(E, F)\} \\ &= \operatorname{argmax}_E \{p(E) \cdot p(F|E)\} \\ &= \operatorname{argmax}_E \{p(E) \cdot \sum_A p(A|E) p(F|A, E)\} \\ &\cong \operatorname{argmax}_E \{p(E) \cdot \max_A \{p(A|E) p(F|A, E)\}\} \end{aligned}$$

extension: more models → log-linear model combination

Hidden Markov Models for MT: Word Alignments (Canadian Parliament; IBM 1993)



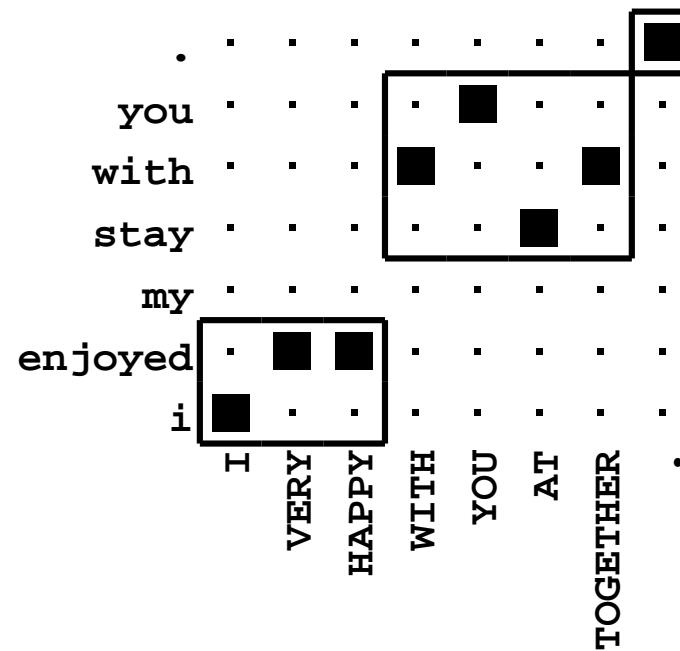
From Single Words to Word Groups (Phrases) (RWTH 1998-2002)

source sentence 我 很 高 兴 和 你 在 一 起 .

gloss notation I VERY HAPPY WITH YOU AT TOGETHER .

target sentence I enjoyed my stay with you .

best alignment for source → target language:

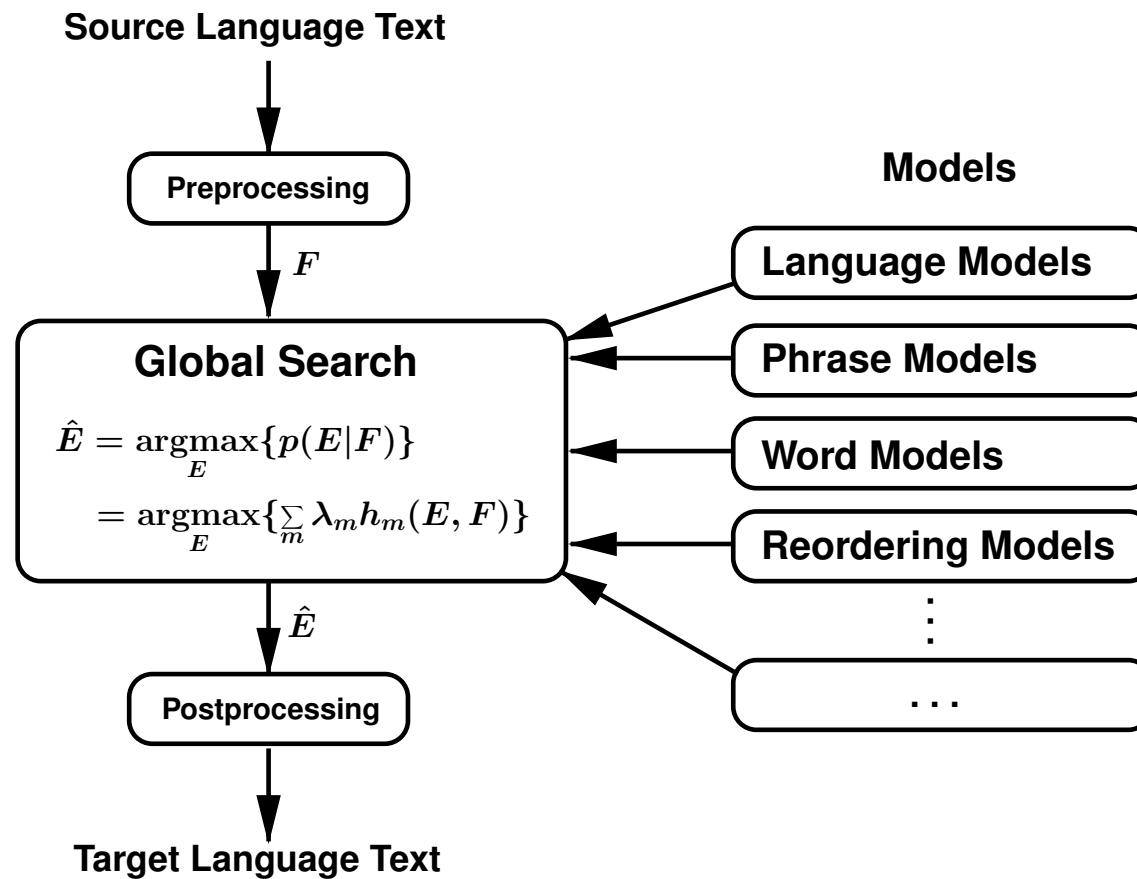


phrase-based approach:

- **training:** extraction
of phrase pairs (= two-dim. 'blocks')
after alignment/lexicon
training
 - **translation process:**
phrases are the smallest units

Phrase Models and Log-Linear Scoring (RWTH 2002)

combination of various types of dependencies $h_m(E, F), m = 1, \dots, M$
using log-linear framework: $p(E|F) = 1/Z_{[\lambda_1^M]}(F) \cdot \exp(\sum_m \lambda_m h_m(E, F))$



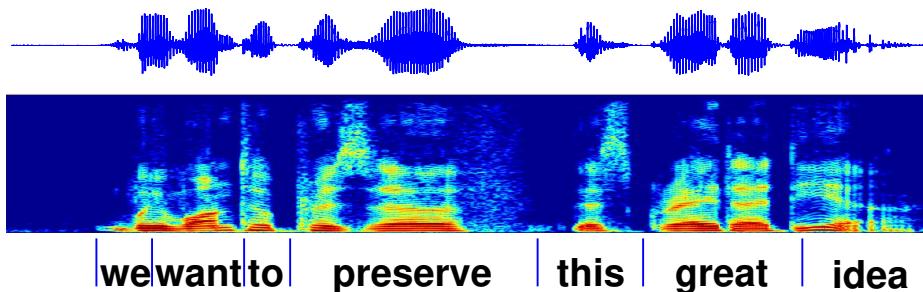
traditional phrase-based MT:

- count-based models and discrete events
- explicit alignment and lexicon model
- hierarchy of models in training: IBM-1 to IBM-5 and HMM
- selection of phrase pairs
- log-linear combination of 'independent' models
- decoding: maximum approximation using coverage constraint

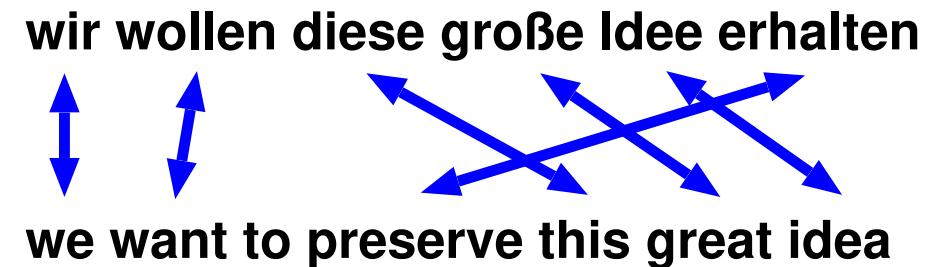
direct modelling for MT, e. g. attention-based approach:

- word embeddings using word vectors
- no explicit alignment model, but attention probabilities $\alpha(j|i)$
- no coverage constraint
- monolithic model and end-to-end training

Speech Recognition



Machine Translation



Text Image Recognition



tasks for machine learning:

- automatic speech recognition (ASR)
- text image recognition
- machine translation (MT)
- sign language (gesture) recognition

- **formalism: sequence-to-sequence conversion**

input sequence F → output sentence E

- **statistical decision theory for optimum performance:**

Bayes decision rule:

$$F \rightarrow \hat{E}(F) = \underset{E}{\operatorname{argmax}}\{p(E|F)\}$$

- **localization effect in sequence pairs:**

- not all input data/symbols interact with all output symbols
- traditional SMT: pairs of (input, output) symbols

HMM and IBM models: alignment and lexicon models

- **NN approaches to localization:**

- attention mechanism
- direct HMM

question: Why are NNs good?

- NOT: because the neural structures are good as such
- BUT: they allow good estimates of the class posterior probabilities needed in Bayes decision rule

most important properties:

- for usual training criteria (squared error, cross-entropy, binary cross-entropy), NN outputs are class posterior probabilities of the training data.
- from single events to sequences:
 - sequence-to-sequence processing by recurrent NN
 - sequence probability: based on chain rule of factorization
- softmax output [Bridle 89]:
output vector is normalized and can be interpreted as posterior distribution
- softmax output of recurrent NN:
use factorization for $E = e_1^I = e_1 \dots e_i \dots e_I$:

$$p(E|F) = p(e_1^I|F) = \prod_i p(e_i|e_0^{i-1}, F)$$

with position-wise posterior probabilities $p(e_i|e_0^{i-1}, F)$

- from subsymbolic (= signal) to symbolic (= text) processing:
 - language modelling, MT and other NLP tasks
 - huge progress by word embeddings
 - = continuous-valued vector representation of words
 - discrete events (words) x, y with vector embeddings $r_x, r_y \in \mathbb{R}^D$: consider the ('co-occurrence') of events x and y :

$$-\log \frac{p(x, y)}{p(x) \cdot p(y)} \cong \|r_x - r_y\|^2 = \|r_x\|^2 + \|r_y\|^2 - 2 r_x^T \cdot r_y$$

interpretation: dot product models the interaction between x and y
(= key operation in ANNs)

- result: joint probability $p(x, y)$ or conditional probability $p(y|x)$ can be computed WITHOUT seeing the pair (x, y) in training
- alleviates the smoothing problem
- smoothing was originally not a NN concept, but fits nicely into the NN framework

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 & Alkhouri⁺ 14]:
recurrent bi-directional LSTM-RNN for SMT
- 2015 [Bahdanau & Cho⁺ 15]: **first competitive ANN stand-alone system for MT:**
jointly learning to align and translate
- 2017 [Vaswani & Shazeer⁺ 17]:
 - Google's transformer approach
 - "attention is all you need"

distinguish two approaches to modelling $p(E|F)$:

- traditional approach (in ASR and MT):

- separate language model $p(E)$ and observation model $p(F|E)$
- consider posterior probability:

$$p(E|F) = \frac{p(E) \cdot p(F|E)}{\sum_{\tilde{E}} p(\tilde{E}) \cdot p(F|\tilde{E})}$$

- discriminative approach (MMI in ASR):

- use $p(E|F)$ as training criterion

- generative approach: ignore denominator in training
and use maximum likelihood in lieu of MMI

- direct approach (discriminative):

- start with posterior probability and use factorization for $E = e_1^I = e_1 \dots e_i \dots e_I$:

$$p(E|F) = p(e_1^I|F) = \prod_i p(e_i|e_0^{i-1}, F)$$

- need for a localization mechanism:

- attention model or alignment model (as in HMM)

differences to conventional phrase-based MT:

- **direct modelling:**
no rewriting using language model prior
- **one single model:**
contrast: log-linear model combination in phrase-based MT
- **one single training criterion:**
contrast: phrase-based MT with training, phrase extraction etc.
- **terminology: 'end-to-end'**

model: posterior probability $p(e_1^I | f_1^J)$ with source sentence f_1^J and target sentence e_1^I

key concepts:

- **direct approach: use unidirectional RNN over target positions $i = 1, \dots, I$ with internal state vector s_i :**

$$p(e_1^I | f_1^J) = \prod_i p(e_i | e_0^{i-1}, f_1^J) = \prod_i p(e_i | e_{i-1}, s_{i-1}, f_1^J)$$

interpretation: extended language model for target word sequence

- **additional component: attention mechanism for localization**

$$p(e_i | e_{i-1}, s_{i-1}, f_1^J) = p(e_i | e_{i-1}, s_{i-1}, c_i)$$

with a context vector: $c_i := C(s_{i-1}, f_1^J)$

word embeddings and representations:

- word embedding for target sequence:
 - word symbol: e_i
 - word vector: $\tilde{e}_i = R_e(e_i)$ with the embedding (matrix) R_e
- word embedding for source sequence:
 - word symbol: f_j
 - word vector: $\tilde{f}_j = R_f(f_j)$ with the embedding (matrix) R_f
- word representation h_j for source sequence using a bidirectional RNN: $h_j = H_j(f_1^J)$

warning:

- concept: clear distinction between f_j, \tilde{f}_j, h_j
- notation and terminology: not necessarily consistent

approach:

- **input: bidirectional RNN over source positions** $j: f_1^J \rightarrow h_j = H_j(f_1^J)$
- **output: unidirectional RNN over target positions** $i:$

$$y_i = Y(y_{i-1}, s_{i-1}, c_i)$$

conventional notation:

$$p(e_i | \tilde{e}_{i-1}, s_{i-1}, c_i)$$

with RNN state vector $s_i = S(s_{i-1}, \tilde{e}_i, c_i)$ **and context vector** $c_i = C(s_{i-1}, h_1^J)$

- **context vector** c_i : **weighted average of source word representations**:

$$c_i = \sum_j \alpha(j|i, s_{i-1}, h_1^J) \cdot h_j \quad \alpha(j|i, s_{i-1}, h_1^J) = \frac{\exp(A[s_{i-1}, h_j])}{\sum_{j'} \exp(A[s_{i-1}, h_{j'}])}$$

with the normalized attention weights $\alpha(j|i, s_{i-1}, h_1^J)$
and real-valued attention scores $A[s_{i-1}, h_j]$

principle:

- **input: source sequence:**

$$f_1^J \rightarrow h_j = H_j(f_1^J)$$

- **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:**

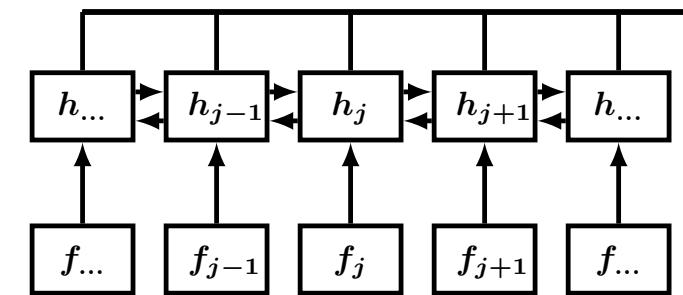
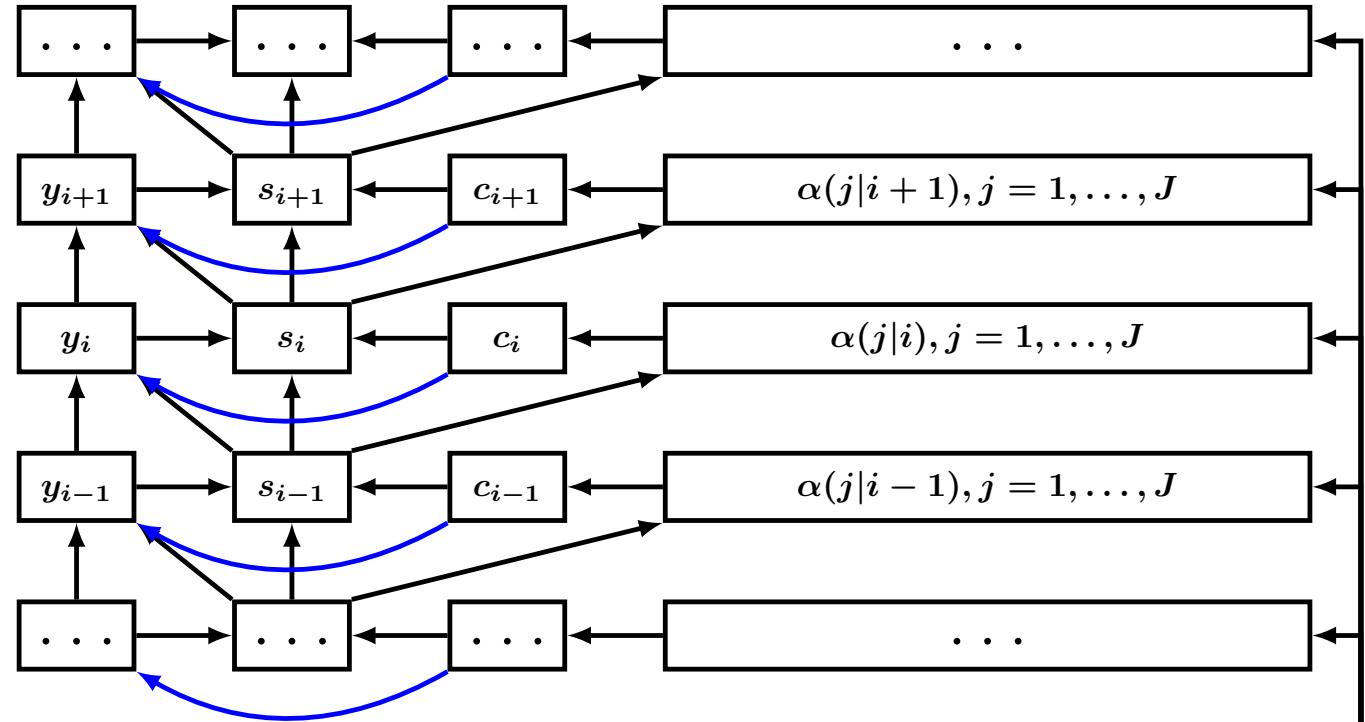
$$s_i = S(s_{i-1}, y_i, c_i)$$

- **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-based Neural MT: Sequential Order of Operations

preparations:

- **input preprocessing:**

$$f_1^J \rightarrow h_j = H_j(f_1^J)$$

- **available at position $i - 1$:**

$$\tilde{e}_{i-1} \equiv y_{i-1}, s_{i-1}, c_{i-1}$$

**sequence of operations
for position i :**

1. **attention weights:**

$$\alpha(j|i, s_{i-1}, h_1^J) = \dots$$

2. **context vector:**

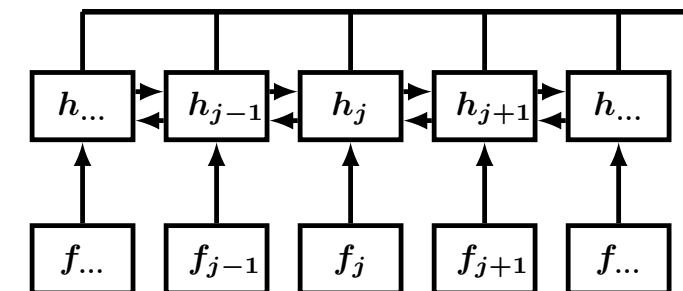
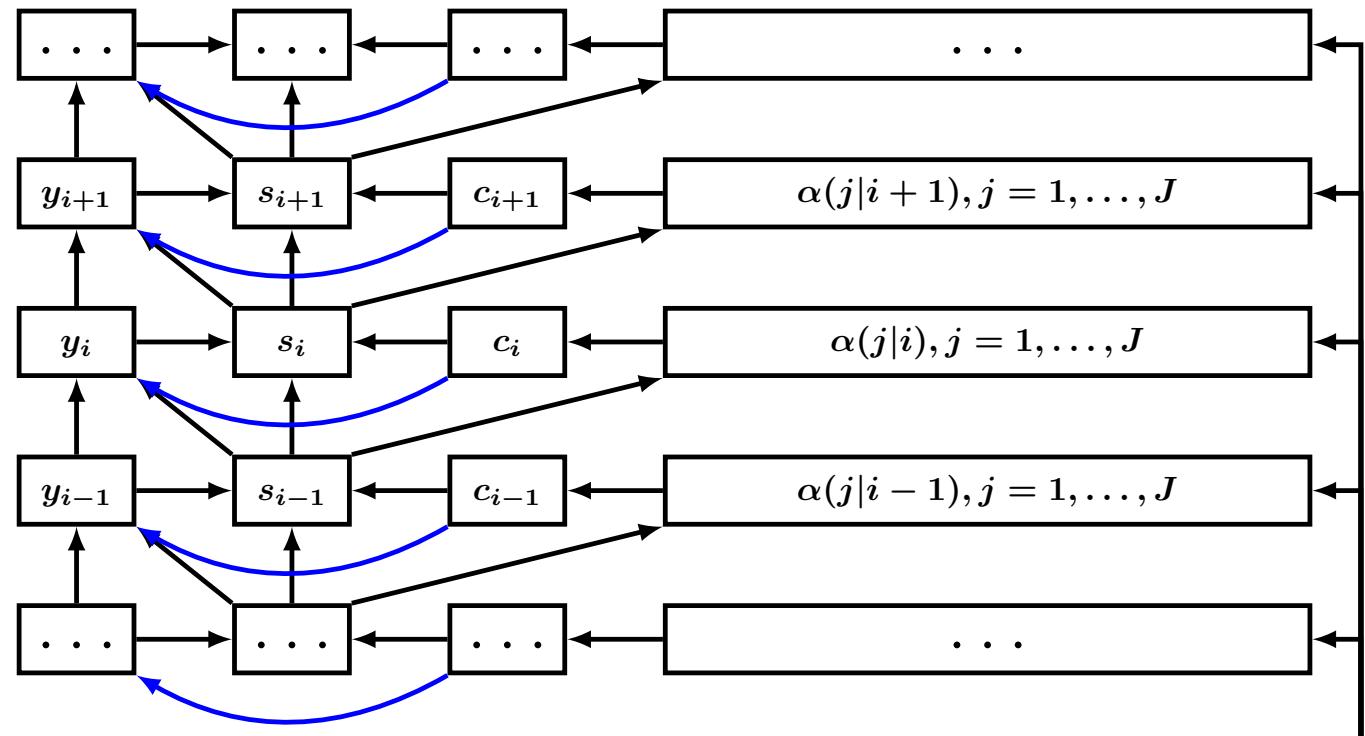
$$c_i = \sum_j \alpha(j|i, s_{i-1}, h_1^J) \cdot h_j$$

3. **output distribution:**

$$y_i = Y(y_{i-1}, s_{i-1}, c_i)$$

4. **state vector:**

$$s_i = S(s_{i-1}, y_i, c_i)$$



Attention Weights FF 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:

$$A[s_{i-1}, h_j] := v^T \cdot \tanh(Ss_{i-1} + Hh_j)$$

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

see examples of attention weights [Bahdanau & Cho⁺ 15]:

SMT_RNN_BahdanauChoBengio_ICLR_2015.pdf

different definitions of attention/alignment weights:

- attention formalism:

$$p(j|i, e_1^{i-1}, f_1^J) = \alpha(j|i, s_{i-1}, h_1^J)$$

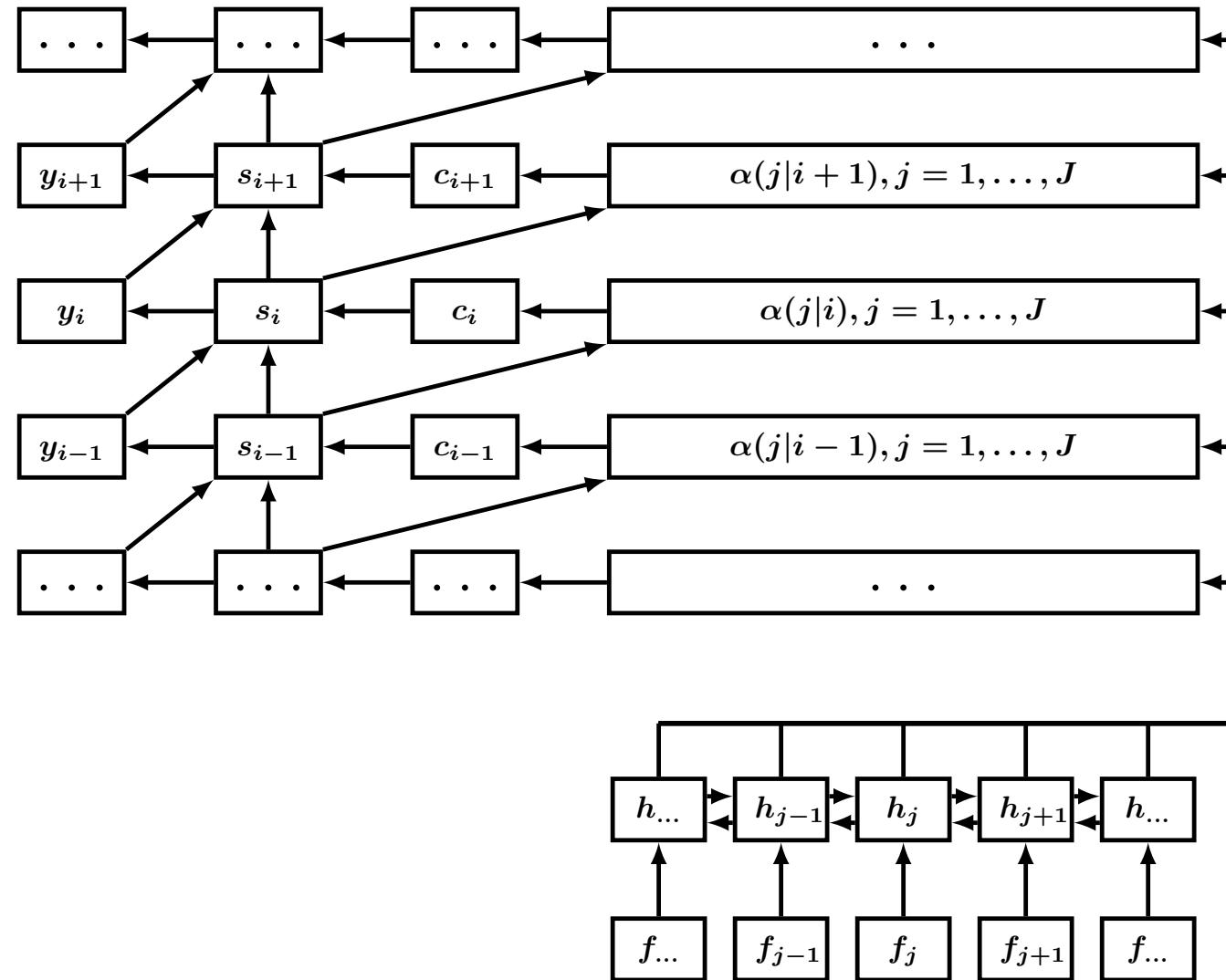
- alignments in direct HMM formalism:

distinguish three types of occupation probabilities:

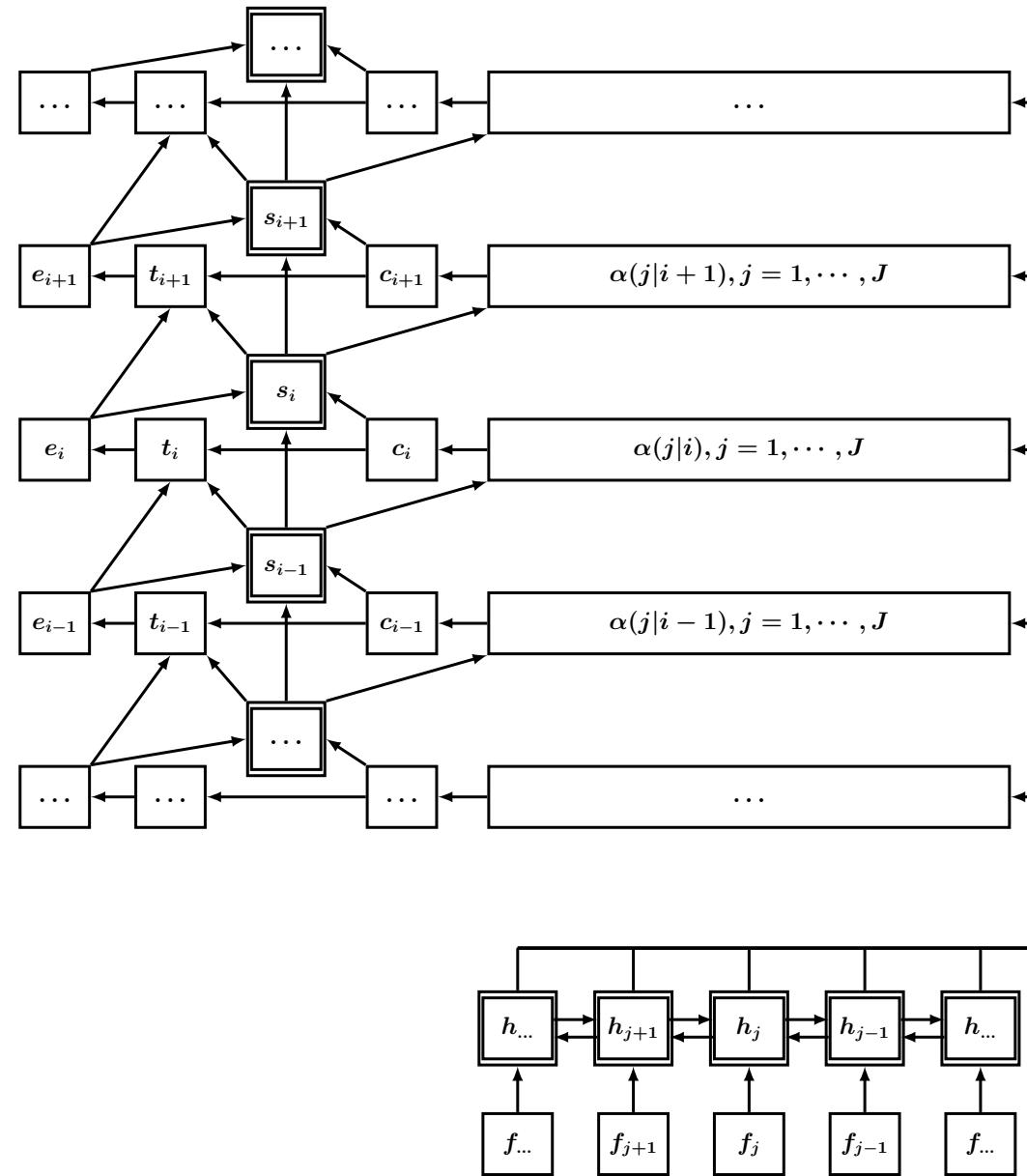
$$\begin{aligned} p(j|i, e_1^{i-1}, f_1^J) &= \dots \\ p(j|i, e_1^i, f_1^J) &= \dots \\ p(j|i, e_1^I, f_1^J) &= \dots \end{aligned}$$

exercise: algorithm for computing these probabilities

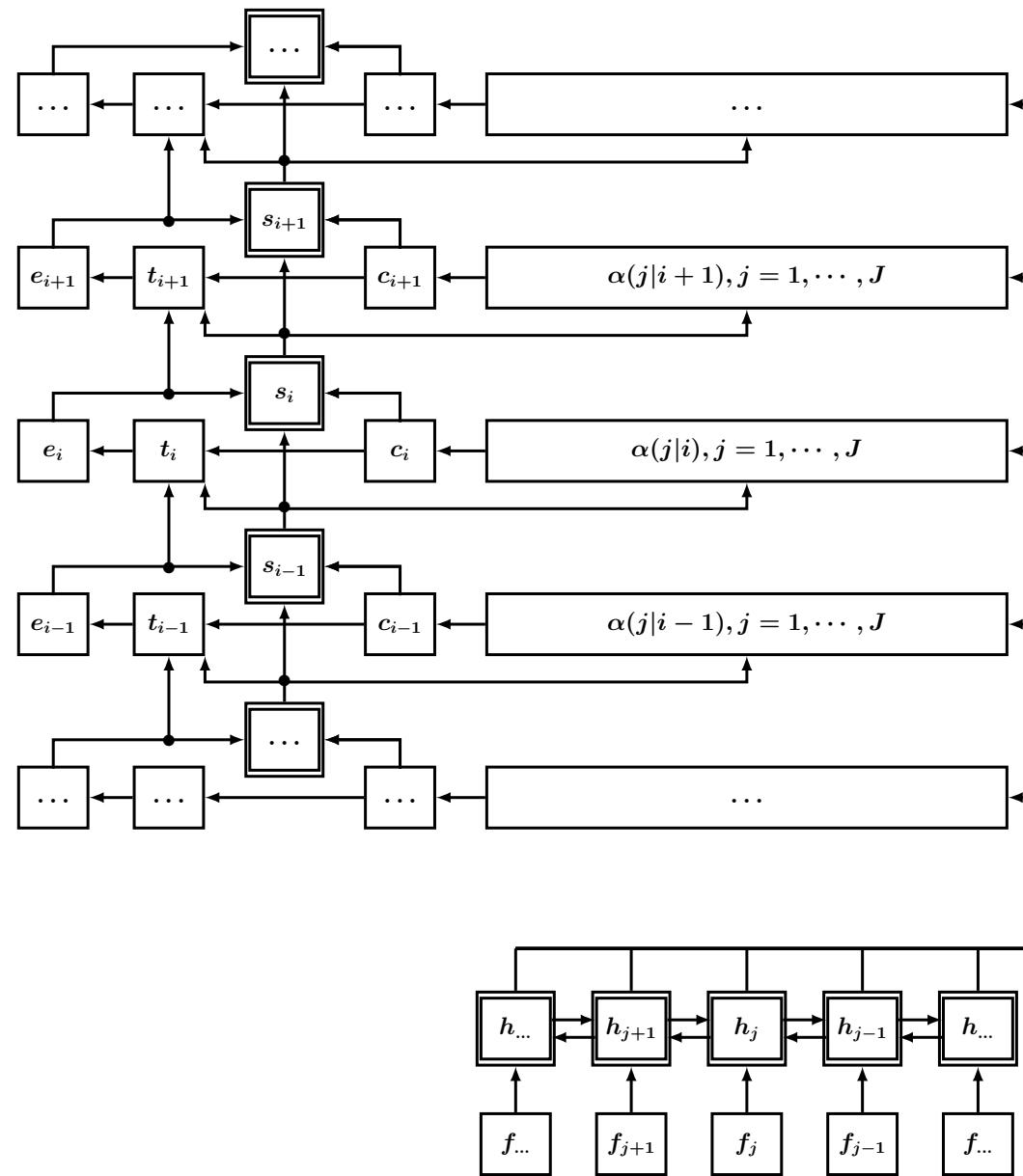
Attention-based Neural MT: Variant [Bahdanau & Cho⁺ 15]



Attention-based Neural MT: Refinements [Bahdanau & Cho⁺ 15]



Attention-based Neural MT: Refinements [Bahdanau & Cho⁺ 15]



full approach requires three ingredients:

- **modelling:**
how to compute the score $p(e_1^I | f_1^J)$ for a given pair (e_1^I, f_1^J) ?
- **training criterion:**
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?

starting point: Bayes decision rule:

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} p(e_1^I | f_1^J) = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} \left\{ \prod_i p(e_i | e_0^{i-1}, f_1^J) \right\}$$

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:

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} \sqrt[I]{p(e_1^I | f_1^J)} = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} \left\{ \sqrt[I]{\prod_i p(e_i | e_0^{i-1}, f_1^J)} \right\}$$

experiments: remedy works sufficiently well

auxiliary 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

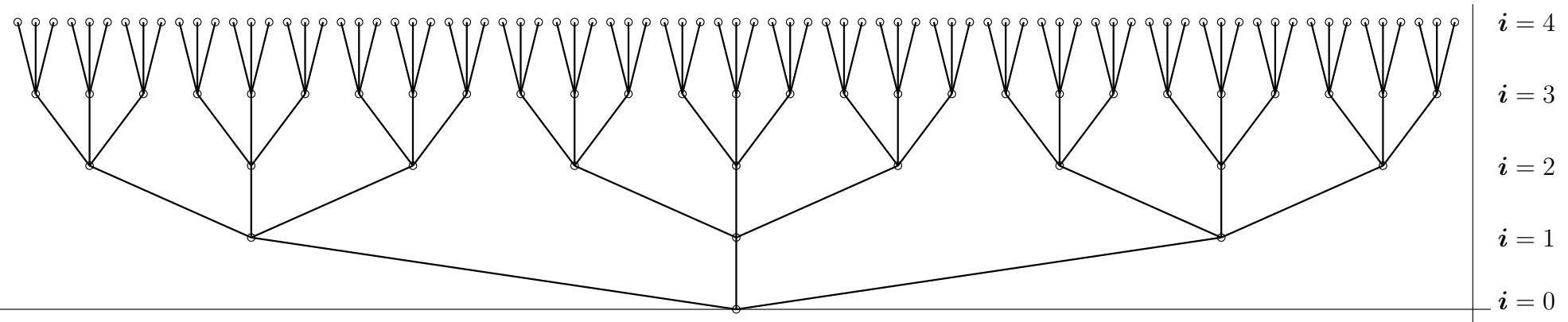
$$Q_0(i) := \max_{e_1^i} Q(i; e_1^i)$$

retain hypotheses e_1^i 'close' to $Q_0(i)$

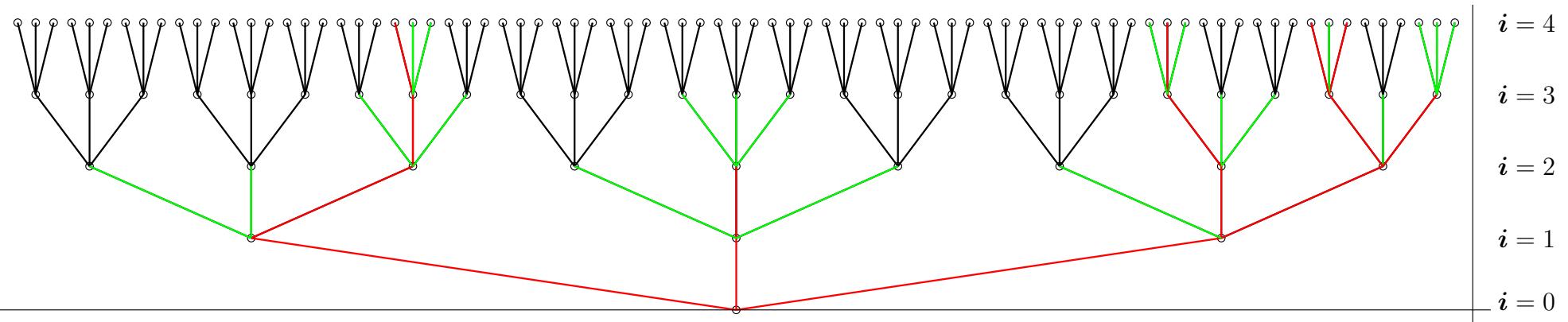
remarks:

- 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

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 $A_n[\cdot, \cdot], n = 1, \dots, N$
- ***stacking of attention***:
several layers (as in RNNs)

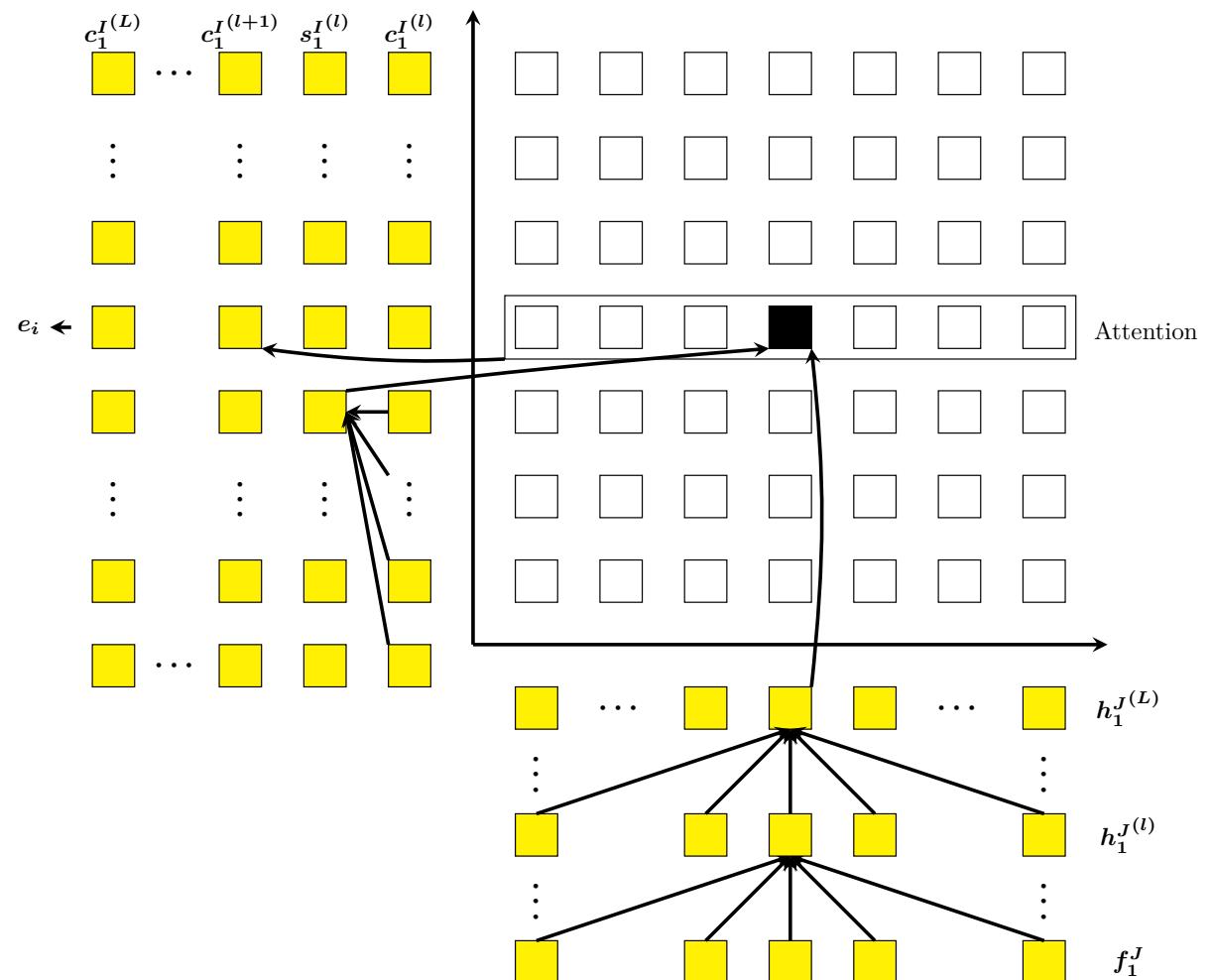
Transformer Approach: Architecture

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 $\alpha(j|i)$

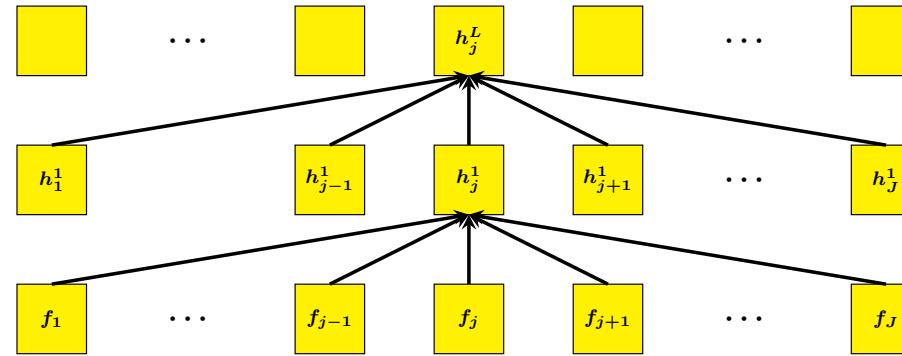
in addition:

- **everywhere: several layers** $l = 1 \dots L$
- **multi-head for cross-attention**
- $n = 1 \dots N$



Transformer Approach

Self-Attention: Source Sentence



mechanism:

- **several layers** $l = 1, \dots, L$ **with representation vectors** $h_j^{(l)}$
initialization by embedding: $h_j^{(l=0)} := \tilde{f}_j$
- **self-attention weights** $\alpha^{(l+1)}(j'|j)$ **with attention scores** $A[\cdot, \cdot]$:

$$\alpha^{(l+1)}(j'|j) = \text{softmax}(A[h_j^{(l)}, h_{j'}^{(l)}])$$

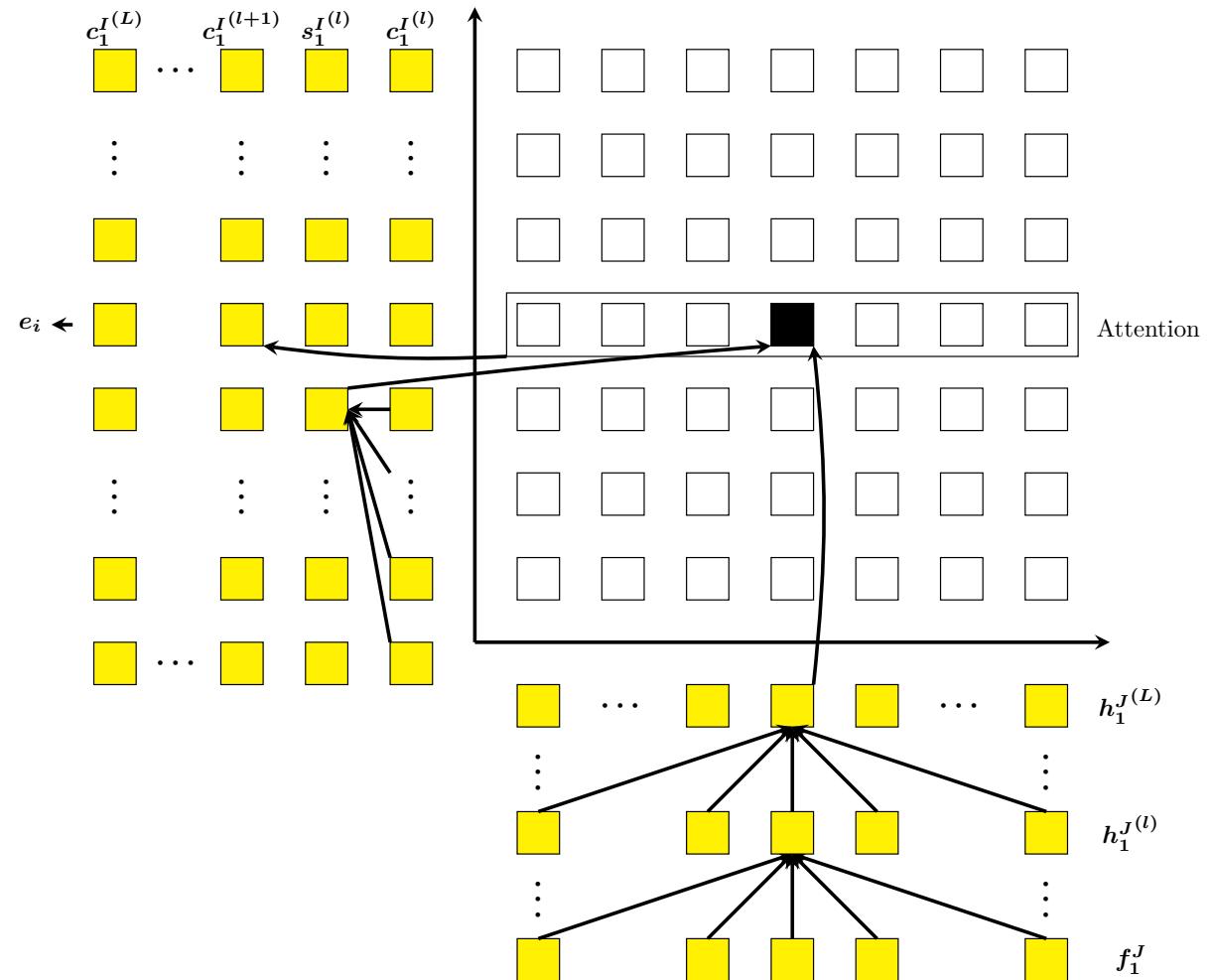
- **representation vector** $h_j^{(l+1)}$:

$$h_j^{(l+1)} = \sum_{j'=1}^J \alpha^{(l+1)}(j'|j) \cdot h_{j'}^{(l)}$$

Transformer Approach: Target Sentence

**several layers over
target positions $i = 1, \dots, I$:**

- **state vectors s_i
(self-attention)**
- **context vectors c_i
(multi-head attention)**



Transformer Approach: Target Sentence

- **self-attention between target positions:**

- self-attention weights $\alpha^{(l+1)}(i'|i - 1)$:

$$\alpha^{(l+1)}(i'|i - 1) = \text{softmax}(A[c_{i-1}^{(l)}, c_{i'-1}^{(l)}])$$

- 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 $\alpha^{(l+1)}(j|i)$:

$$\alpha^{(l+1)}(j|i) = \text{softmax}(A[s_{i-1}^{(l)}, h_j^{(L)}])$$

- context vectors $c_i^{(l+1)}$:

$$c_i^{(l+1)} = \sum_{j=1}^J \alpha^{(l+1)}(j|i) \cdot h_j^{(L)} \quad \text{with} \quad c_i^{(l=0)} := \tilde{e}_i$$

Transformer Approach: Output Generation

output generation:

- ReLU activation:

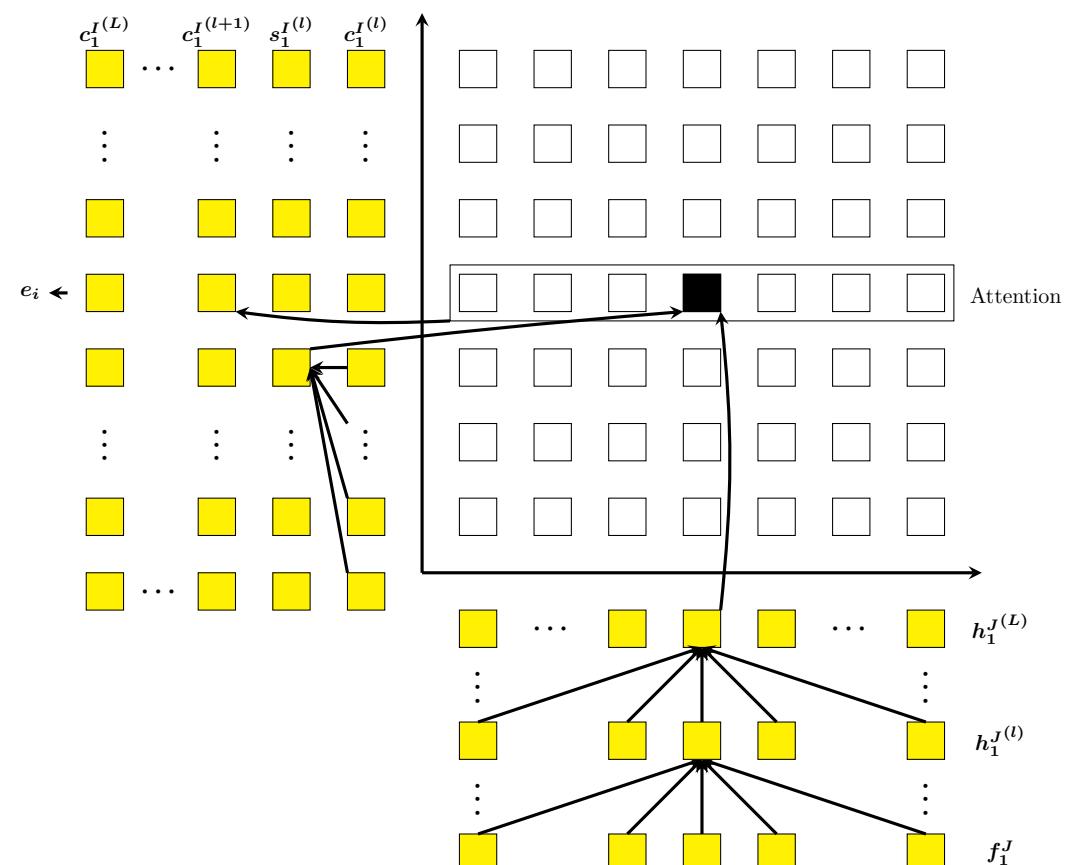
$$t_i = \text{ReLU}(C_i^{(L)})$$

- softmax output:

$$y_i \equiv p_i(e|e_1^{i-1}, f_1^J) = \text{softmax}(t_i)$$

- for comparison in original
attention approach:

$$\begin{aligned} y_i &\equiv p_i(e|e_1^{i-1}, f_1^J) \\ &= \text{softmax}(\tilde{e}_{i-1}, s_{i-1}, c_i) \end{aligned}$$



Transformer Approach: Multi-Head Attention

- principle: for each head index $n = 1, \dots, N$:
 - 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: $H_j^{(l)} := [h_{j,1}^{(l)}, \dots, h_{j,n}^{(l)}, \dots, h_{j,N}^{(l)}]$
 - target state vectors: $S_i^{(l)} := [s_{i,1}^{(l)}, \dots, s_{i,n}^{(l)}, \dots, s_{i,N}^{(l)}]$
 - context vectors: $C_i^{(l)} := [c_{i,1}^{(l)}, \dots, c_{i,n}^{(l)}, \dots, c_{i,N}^{(l)}]$
- example: cross-attention between source and target positions
 - cross-attention weights $\alpha_n^{(l+1)}(j|i)$:

$$\alpha_n^{(l+1)}(j|i) = \text{softmax}(A_n[s_{i-1}^{(l)}, h_j^L])$$

- context vectors $c_{i,n}^{(l+1)}$:

$$c_{i,n}^{(l+1)} = \sum_{j=1}^J \alpha_n^{(l+1)}(j|i) \cdot h_j^{(L)} \quad \text{with} \quad c_{i,n}^{(l=0)} := \tilde{e}_i$$

Transformer Approach: Multi-Layer Architecture

terminology in Google paper
for attention mechanism:

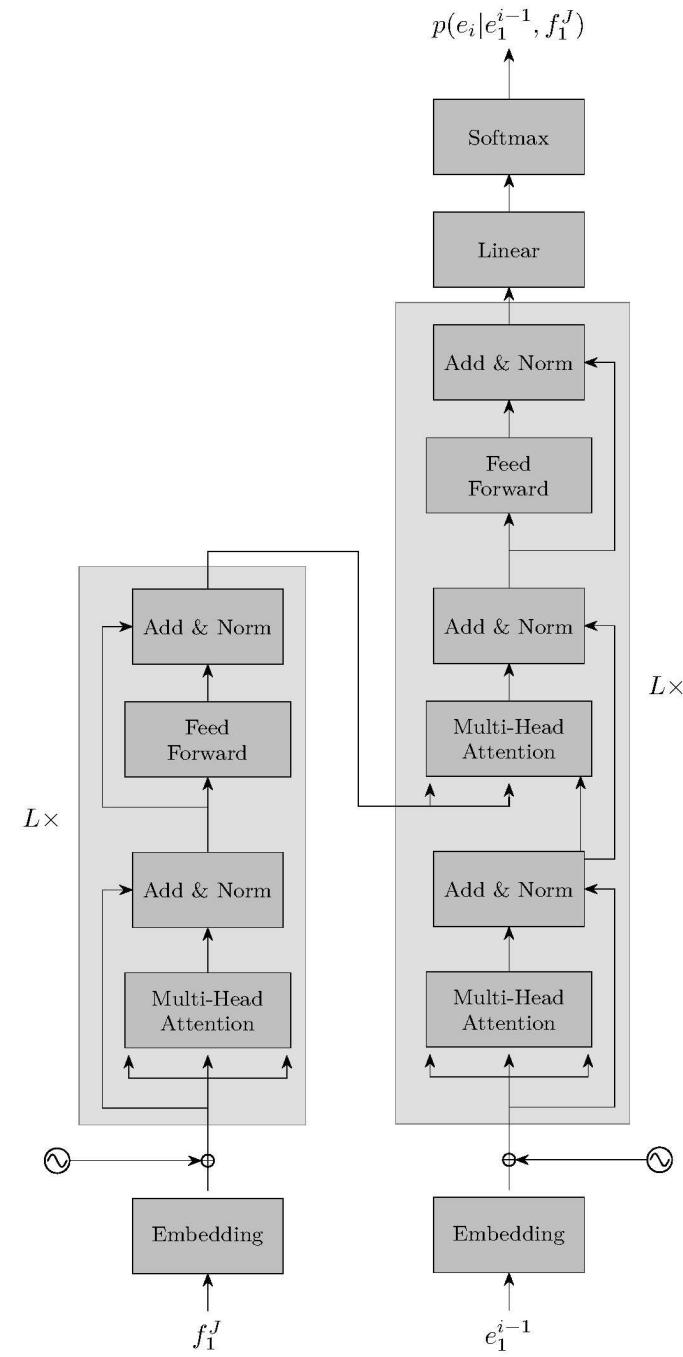
definitions:

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 $A_n[\cdot, \cdot]$

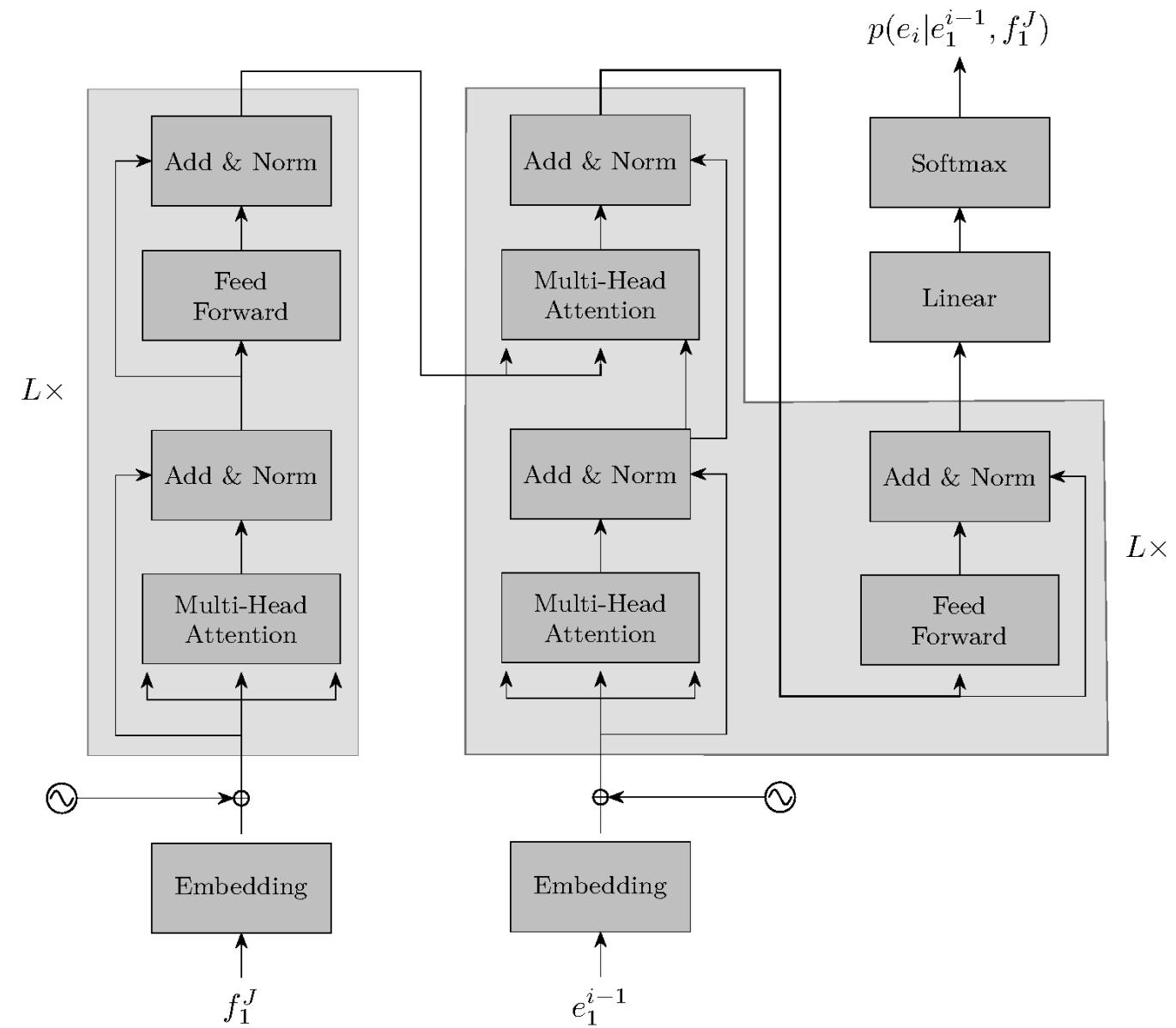
value: state/context vectors

(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)



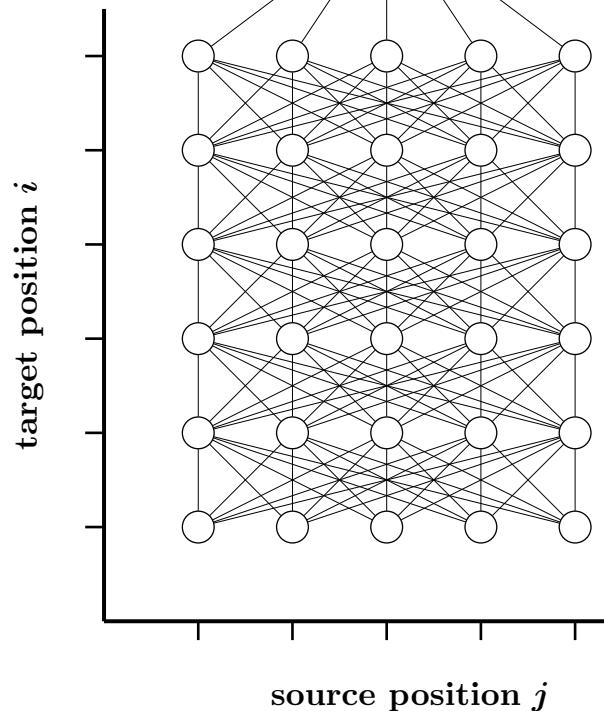
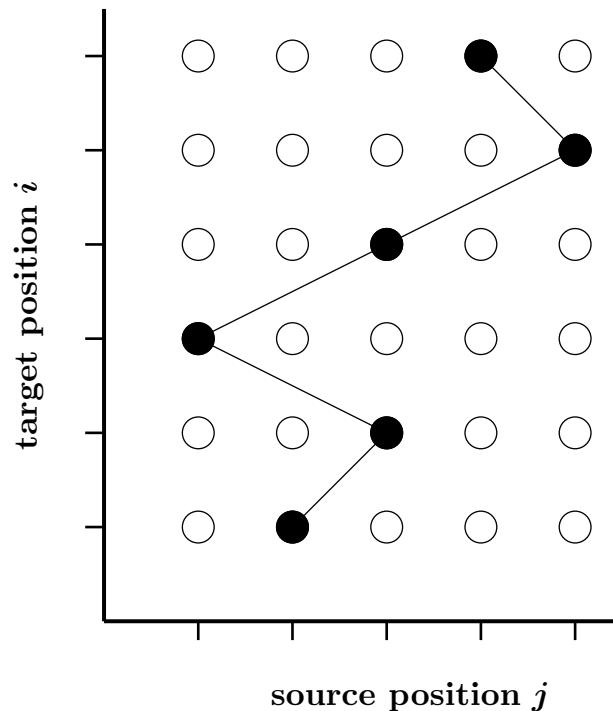
transformer approach:

- 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

motivation:

- direct factorization as in attention modelling
- alignment path:
 - from target to source
 - using neural networks for modelling alignment and lexicon dependencies
- sum over all alignments:
can be carried out exactly

Alignment-based HMM



- **alignment direction:** from target to source
- **sum over alignments (first-order model):**
 - can be represented by trellis
 - can be computed EXACTLY in training and decoding
- **additional advantage:**
convenient for decoding using a bottom-to-top search

[RWTH: EMNLP 14, EMNLP 15, WMT 15, ACL 16, WMT 16, WMT 17, ACL 17]

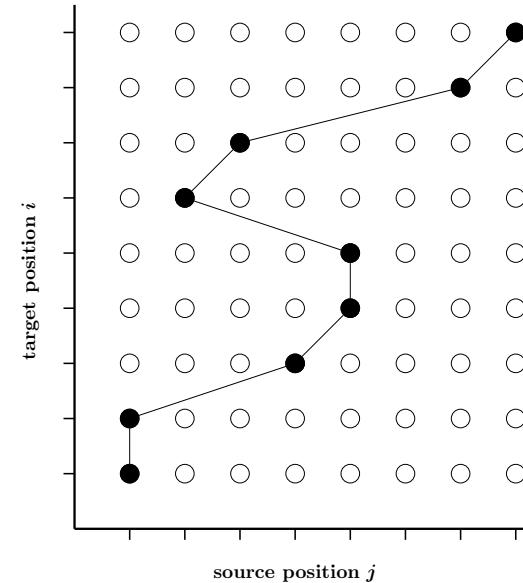
translation:

from source sentence $f_1^J = f_1 \dots f_j \dots f_J$
to target sentence $e_1^I = e_1 \dots e_i \dots e_I$

alignments from target i to source j :

$$i \rightarrow j = b_i$$

reasons: better for modelling and decoding



we re-write the translation probability $p(e_1^I | f_1^J)$ with first-order structure

[Vogel & Ney⁺ 96]:

$$\begin{aligned} p(e_1^I | f_1^J) &= \sum_{b_1^I} p(b_1^I, e_1^I | f_1^J) = \sum_{b_1^I} \prod_i p(b_i, e_i | b_0^{i-1}, e_0^{i-1}, f_1^J) = \dots \\ &= \sum_{b_1^I} \prod_i p(b_i, e_i | b_{i-1}, e_0^{i-1}, f_1^J) \end{aligned}$$

with extended lexicon/alignment model $p(b_i, e_i | \dots)$

novel approach:

$$\begin{aligned} p(e_1^I | f_1^J) &= \sum_{b_1^I} \prod_i p(b_i, e_i | b_{i-1}, e_0^{i-1}, f_1^J) \\ &= \sum_{b_1^I} \prod_i [p(b_i | b_{i-1}, e_0^{i-1}, f_1^J) \cdot p(e_i | b_{i-1}^i, e_0^{i-1}, f_1^J)] \end{aligned}$$

with neural networks (to be defined)
for the alignment model $p(b_i | \dots)$ and lexicon model $p(e_i | \dots)$

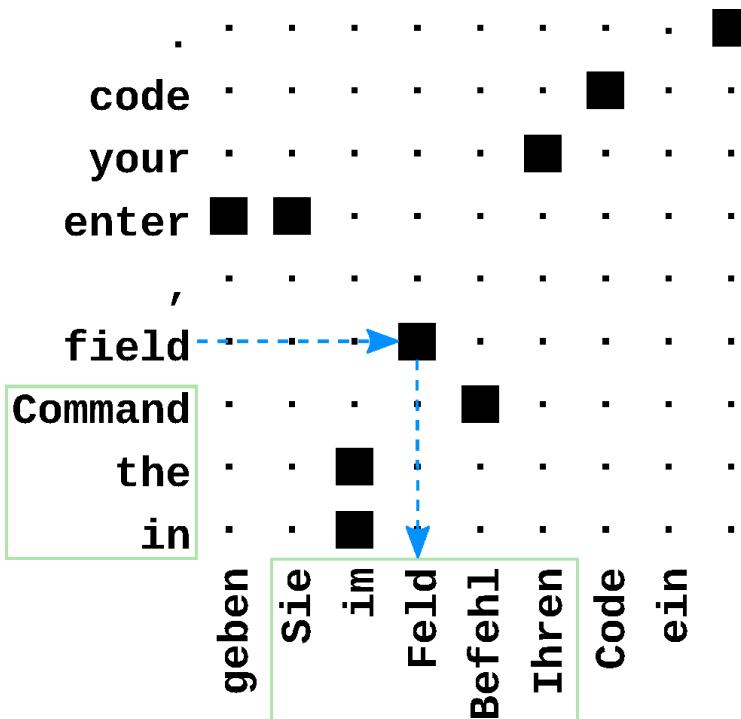
traditional approach using simplified dependencies [Vogel & Ney⁺ 96]:

$$p(e_1^I | f_1^J) = \sum_{b_1^I} \prod_i [p(b_i | b_{i-1}, I, J) \cdot p(e_i | f_{b_i})]$$

with count-based tables (as in IBM models)
for the alignment model $p(b_i | b_{i-1}, I, J)$ and the lexicon model $p(e_i | f_{b_i})$

posterior probability:

$$\begin{aligned}
 p(e_1^I | f_1^J) &= \sum_{b_1^I} \prod_i p(b_i, e_i | b_{i-1}, e_0^{i-1}, f_1^J) \\
 &= \sum_{b_1^I} \prod_i [p(b_i | b_{i-1}, e_0^{i-1}, f_1^J) \cdot p(e_i | b_{i-1}^i, e_0^{i-1}, f_1^J)]
 \end{aligned}$$



lexicon model:

$$p(e_i | f_{b_{i-2}}^{b_i+2}, e_{i-3}^{i-1})$$

alignment model:

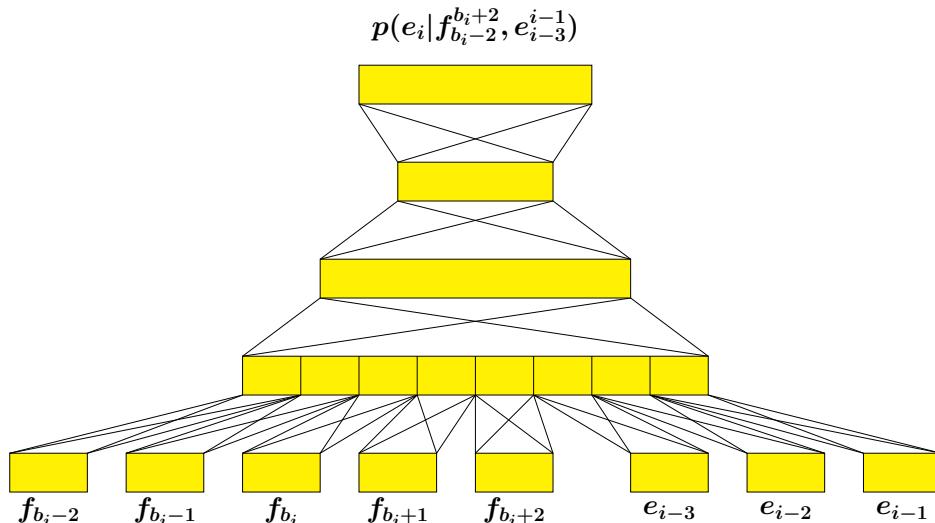
$$p(\Delta_i | f_{b_{i-1}-2}^{b_{i-1}+2}, e_{i-3}^{i-1})$$

with $\Delta_i := b_i - b_{i-1}$

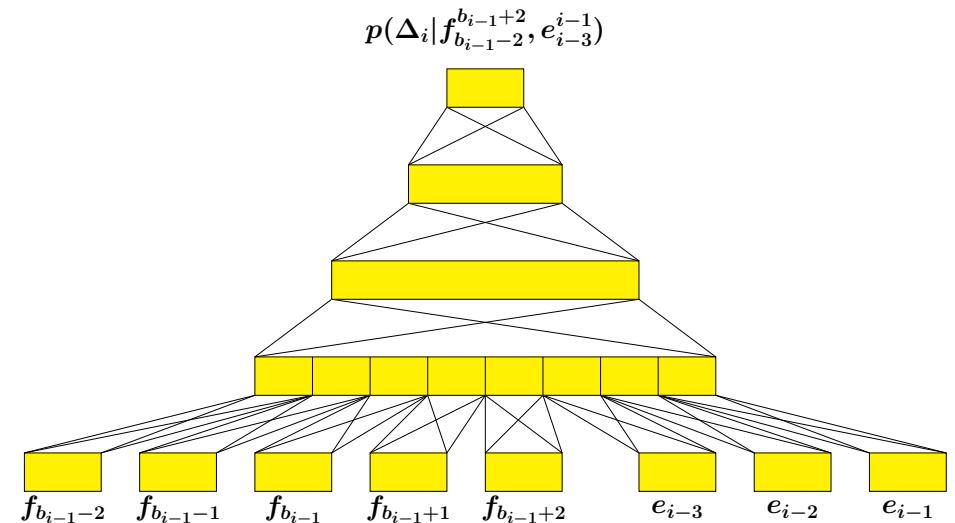
FF NN: Structure

- **NN structure:** feedforward NN for both models:

- input: hot encoding $1 : V$
- projection layer
- several hidden layers
- output layer: softmax

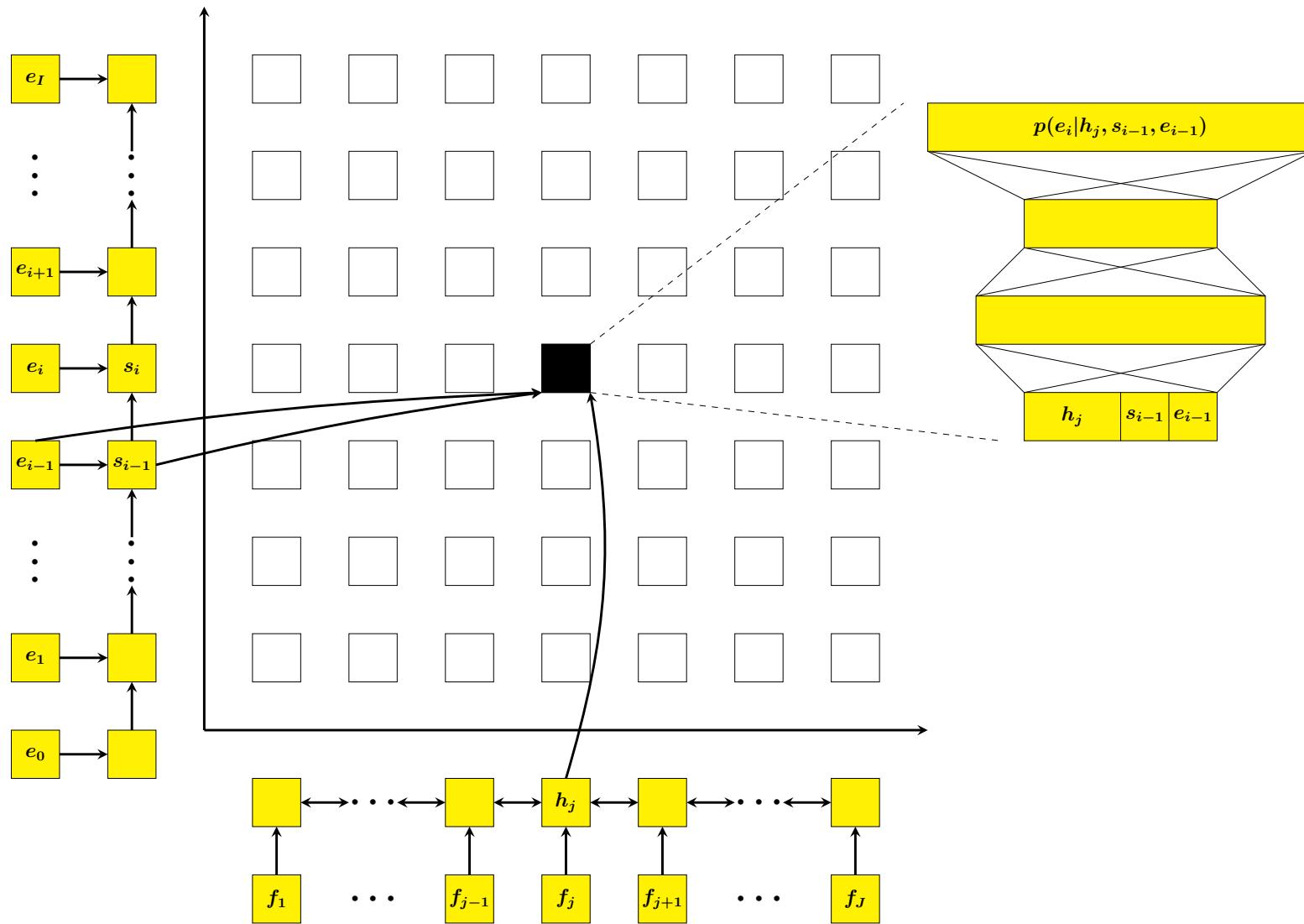


lexicon model $p(e_i | f_{b_{i-2}}^{b_i+2}, e_{i-3}^{i-1})$
(limited size of output by BPE)



alignment model $p(\Delta_i | f_{b_{i-1-2}}^{b_{i-1}+2}, e_{i-3}^{i-1})$
with $\Delta_i := b_i - b_{i-1}$

LSTM RNN: Structure



modelling assumptions:

- **lexicon model:**
 - drop dependence on b_{i-1}
 - **source-side bidirectional LSTM-RNN embedding:**

$$h_j = H_j(f_1^J)$$

- **target-side unidirectional LSTM-RNN embedding:**

$$s_i = S(s_{i-1}, e_i, f_1^J)$$

result: $p(e_i | h_j, s_{i-1})$

- **alignment model: similar to lexicon model**
 - **in addition: dependence on predecessor source word:** $h_{j'}$, where $j' = b_{i-1}$
 - **use relative position** $\Delta_i = b_i - b_{i-1}$

result: $p(\Delta_i | h_{j'}, s_{i-1})$

Computational Aspects (in contrast to modelling aspects)

- known target string:
efficient calculation of the sum over all alignments:

$$p(e_1^I | f_1^J) = \sum_{b_1^I} \prod_i p(b_i, e_i | b_{i-1}, e_0^{i-1}, f_1^J)$$

exact solution: dynamic programming or Baum algorithm

- unknown target string:
search (decoder) over all possible target strings

$$\max_{I, e_1^I} p(e_1^I | f_1^J) = \max_{I, e_1^I} \left\{ \sum_{b_1^I} \prod_i p(b_i, e_i | b_{i-1}, e_0^{i-1}, f_1^J) \right\}$$

note: exact sum over alignments

target-synchronous beam search for target string

- training criterion: log of sentence posterior probabilities
(cross-entropy, MMI in ASR) over training sentence pairs $(E_r, F_r), r = 1, \dots, R$:

$$\operatorname{argmax}_{\theta} \left\{ \sum_r \log p_{\theta}(E_r | F_r) \right\}$$

solution: (sort of) EM algorithm + backpropagation

most general first order model:

$$p(j = b_i, e_i | j' = b_{i-1}, e_0^{i-1}, f_1^J)$$

**DP recursion for auxiliary quantity $Q(i, j)$
for a KNOWN target string e_1^I :**

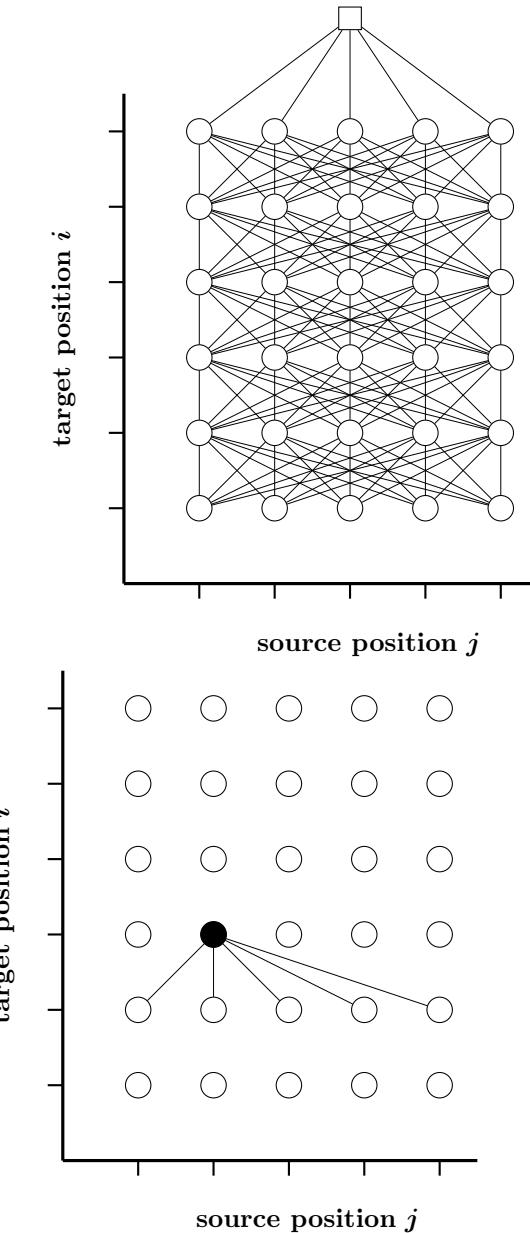
$$Q(i, j) = \sum_{j'} [p(j, e_i | j', e_0^{i-1}, f_1^J) \cdot Q(i - 1, j')]$$

final result: $p(e_1^I | f_1^J) = \sum_j Q(I, j)$

overall complexity: $J^2 \cdot I$

maximum approximation in DP recursion:

$$Q(i, j) = \max_{j'} \{p(j, e_i | j', e_0^{i-1}, f_1^J) \cdot Q(i - 1, j')\}$$



auxiliary quantity for each unknown partial string e_1^i :

$$Q(i, j; e_1^i)$$

search: extending partial hypothesis from e_1^{i-1} to e_1^i :

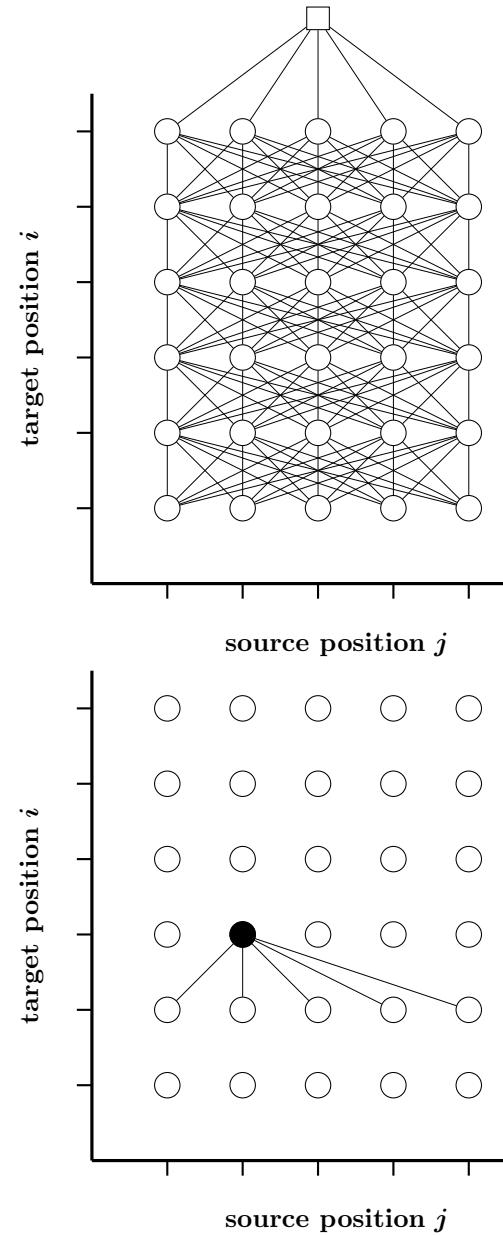
$$Q(i, j; e_1^i) = \sum_{j'} [p(j, e_i | j', e_0^{i-1}, f_1^J) \cdot Q(i - 1, j'; e_1^{i-1})]$$

final result:

$$p(e_1^I | f_1^J) = \sum_j Q(I, j; e_1^I)$$

resulting complexity:

- alignments only: polynomial (by DP)
- language model (LM): exponential for n -gram and RNN LMs
- approximation: beam search using tree representation of partial target string hypotheses



Unknown Target String: Length Normalization (as for Attention Model)

starting point: Bayes decision rule:

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} p(e_1^I | f_1^J) = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} \left\{ \prod_i p(e_i | e_0^{i-1}, f_1^J) \right\}$$

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:

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} \sqrt[I]{p(e_1^I | f_1^J)} = \underset{I, e_1^I : e_I = \$}{\operatorname{argmax}} \left\{ \sqrt[I]{\prod_i p(e_i | e_0^{i-1}, f_1^J)} \right\}$$

experiments: remedy works sufficiently well

sequence of source-target sentence pairs for training:

$$(F_r, E_r), r = 1, \dots, R$$

**training criterion: log of sentence posterior probabilities
(cross-entropy, MMI in ASR):**

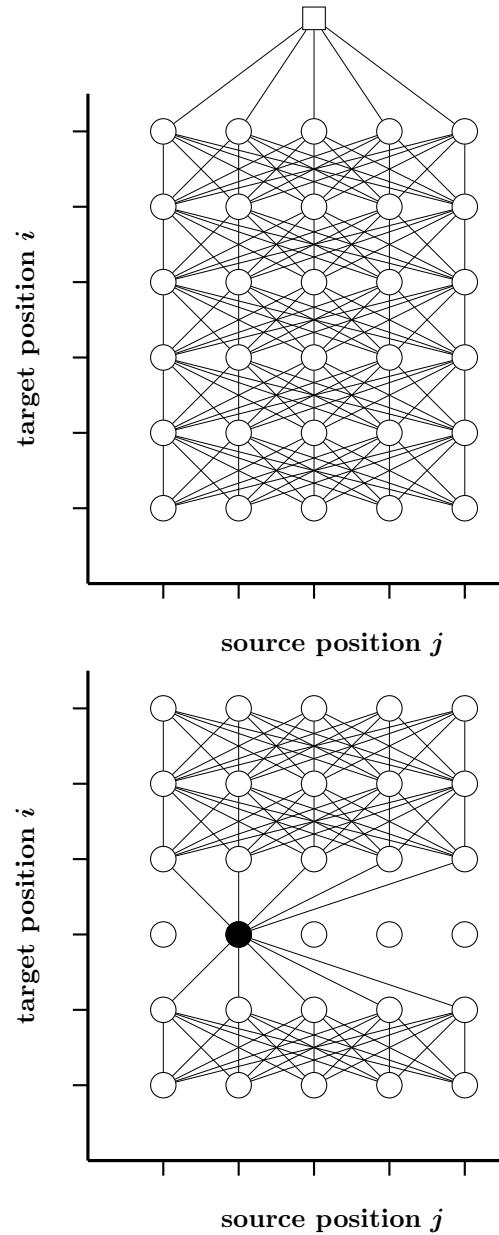
$$\operatorname{argmax}_{\theta} \left\{ \sum_r \log p_{\theta}(E_r | F_r) \right\}$$

derivative for a single sentence pair $(E, F) = (e_1^I, f_1^J)$:

$$\begin{aligned} \frac{\partial}{\partial \theta} \log p_{\theta}(E|F) &= \dots \\ &= \sum_{j',j} \sum_i p_i(j', j | f_1^J, e_1^I; \theta) \cdot \frac{\partial}{\partial \theta} \log p(j, e_i | j', e_0^{i-1}, f_1^J; \theta) \end{aligned}$$

**with the HMM posterior weights $p_i(j', j | f_1^J, e_1^I; \theta)$
(as in traditional EM algorithm for HMM)**

**illustration (bottom): forward-backward algorithm
for computing HMM weights $p_i(j | f_1^J, e_1^I; \theta)$**



training strategy:

- **initialization:**

**IBM-1 model is used to compute the HMM posterior weights
(guarantee of global optimum)**

- **backpropagation in EM framework:**

- 1) **compute:**

- the posterior HMM weights
- the local gradients (backpropagation)

- 2) **update NN weights**

related work [Schwenk 12, Le & Allauzen⁺ 12, Devlin & Zbib⁺ 14]:

- FF NN as additional model in phrase-based approach
- alignments were fixed (GIZA++)

related work by Dyer et al. 2016:

- stand-alone NN system
- source-to-target alignments
- rescoring only

related work by RWTH:

- maximum approximation:
additional heuristics beyond 1:1 word alignments
- neural models: Alkhouri et al. (EMNLP, WMT)
- event counts: Guta et al. (EMNLP, WMT)
- training: starts with GIZA++ alignments

WMT 2017 German→English corpus statistics

WMT 2017		German	English
train	Sentences	4.2M	
	Running Words	107M	109M
	Vocabulary	813K	773K
newtest2015 (dev set)	Sentences	2169	
	Running Words	53K	49K
	Unique words	8022	6706
newtest2016	Sentences	2999	
	Running Words	74K	68K
	Unique words	9253	7506
newtest2017	Sentences	3004	
	Running Words	73K	68K
	Unique words	9088	7378

byte pair encoding (20K subword units)
– no OOV words

WMT 2017 German→English translation task

WMT G→E	Test A (newstest2016)		Test B (newstest2017)		# free parameters
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	
phrase-based system	31.4	49.4	27.1	53.7	30M
alignment-based FF NN	30.2	49.6	26.3	53.5	30M
+ source side RNN	30.8	48.8	26.8	53.1	40M
+ bag-of-words	30.8	48.7	26.8	53.0	44M
+ predecessor source word	30.7	48.7	26.8	52.8	44M
+ target side RNN	31.2	48.5	27.2	52.6	45M
+ weight averaging	31.4	48.3	27.4	52.3	45M
attention-based NMT*	32.1	47.9	27.9	52.3	77M

* RWTH variant:

- theano blocks, LSTM-RNN, 2 encoder layers, 1 decoder layer
- with weight averaging
- without ensembling and without synthetic data

BOLT Chinese→English corpus statistics

BOLT		Chinese	English
train	Sentences	4.1M	
	Running Words	78M	86M
	Vocabulary	384K	817K
dev	Sentences	1845	
	Running Words	38K	47K
	Unique words	5824	5715
test1	Sentences	1844	
	Running Words	39K	49K
	Unique words	5916	5646
test2	Sentences	1124	
	Running Words	24K	31K
	Unique words	4398	4120

byte pair encoding (20K subword units)
– no OOV words

BOLT Chinese→English translation task

BOLT C→E	Test A (DEV12)		Test B (P1R6)		# free parameters
	BLEU[%]	TER[%]	BLEU[%]	TER[%]	
phrase-based system	17.9	68.3	17.1	67.4	31M
alignment-based FF NN	17.2	68.5	17.0	67.8	33M
+ source side RNN	18.0	67.9	17.6	67.4	43M
+ target side RNN	18.4	67.4	18.0	67.1	48M
+ weight averaging	18.5	67.2	18.3	67.0	48M
attention-based NMT*	19.7	66.3	19.3	66.2	78M

* RWTH variant:

- theano blocks, LSTM RNN cell, 2 encoder layers, 1 decoder layer
- with weight averaging
- without ensembling and without synthetic data

operations:

- **RNN on source side:**

$$f_1^J \rightarrow h_j = H_j(f_1^J)$$

- **alignment weights:**

$$\alpha(j|i) = A(s_{i-1}, h_j)$$

- **weighted context vector:**

$$c_i = \sum_j \alpha(j|i) \cdot h_j$$

- **output vector y_i :**

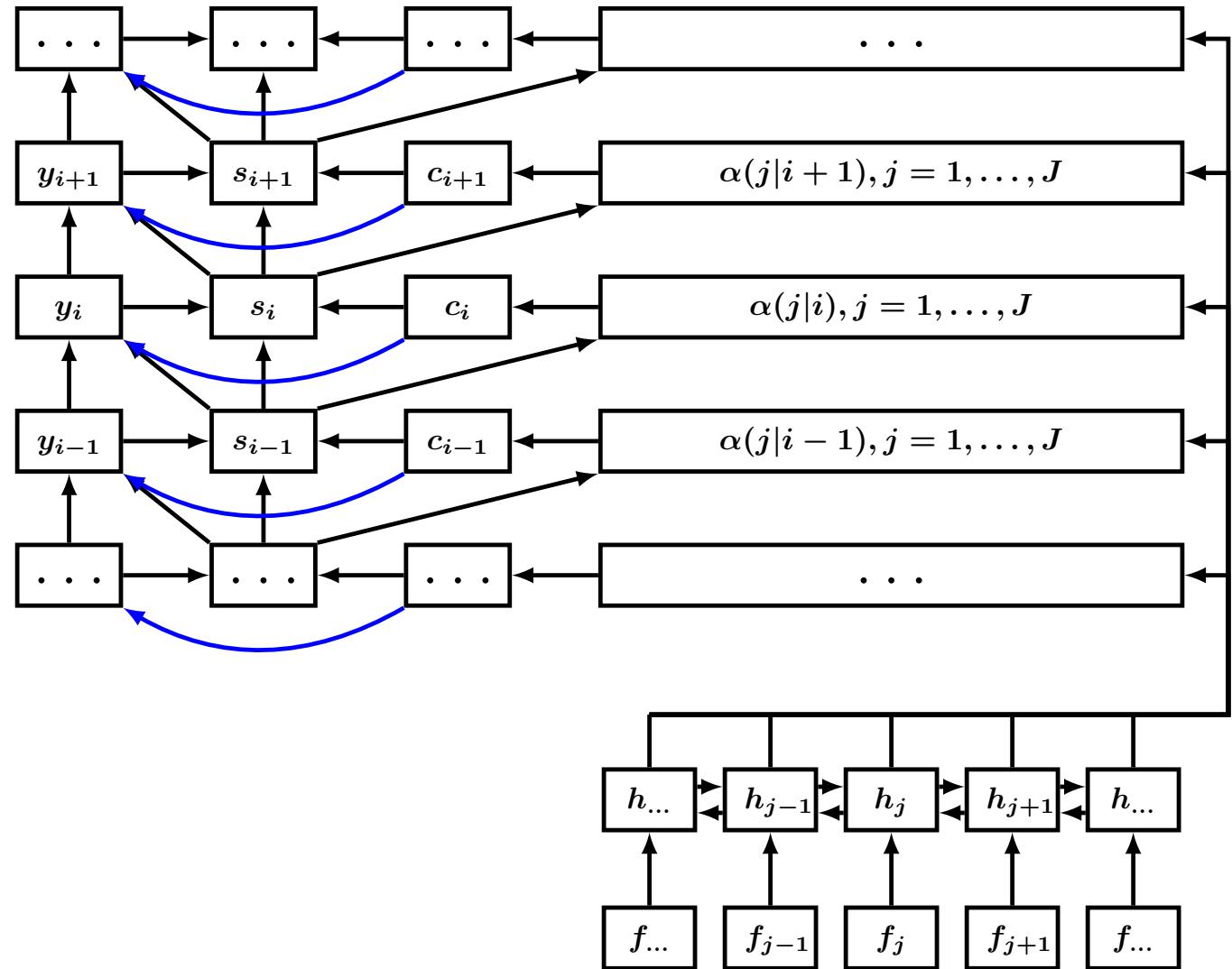
$$y_i = Y(y_{i-1}, s_{i-1}, c_i)$$

conventional notation:

$$y_i \equiv p_i(e|e_0^{i-1}, h_1^J)$$

- **state vector of target RNN:**

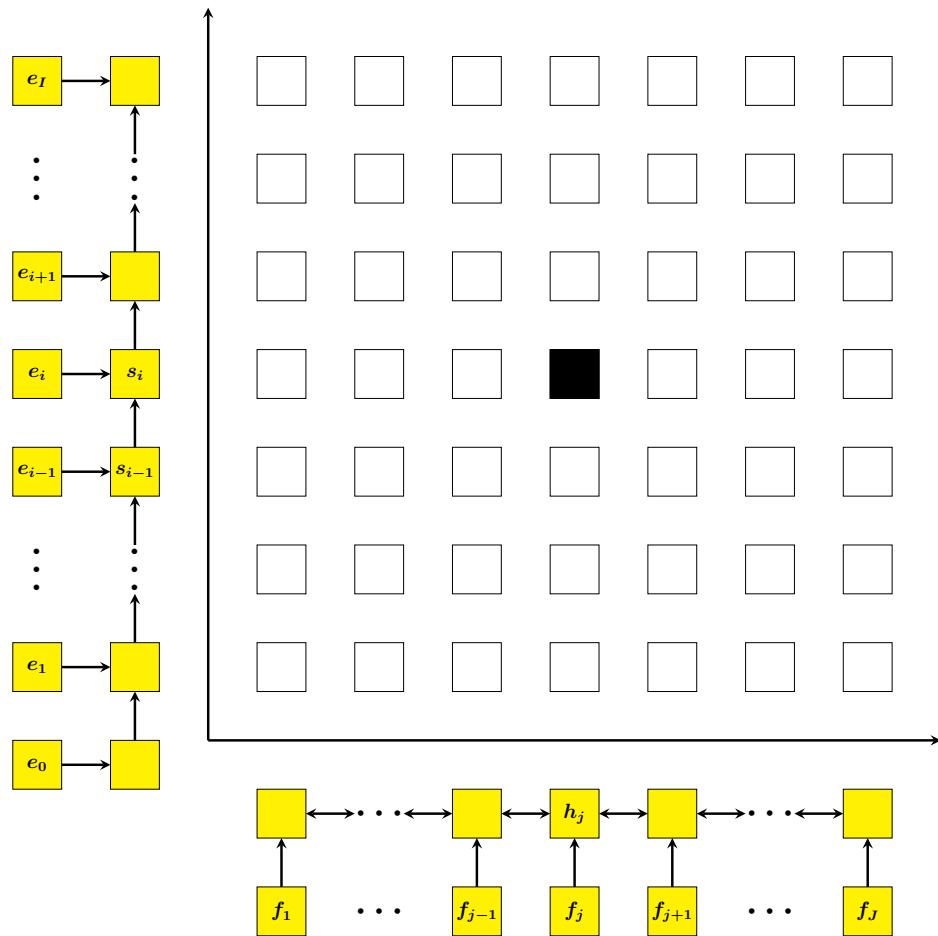
$$s_i = S(s_{i-1}, y_i, c_i)$$



RNNs with LSTM/GRU extensions on source and target sides

comparison:

- **alignment approach:**
 - probabilities $p(e_i, b_i | \dots)$ of alignment and lexicon models
 - true 2D recurrence over i and j
- **attention approach:**
 - attention weights provide the link between i and j :
$$c_i = \sum_j \alpha(j|i) \cdot h_j$$
 - two decoupled 1D recurrences



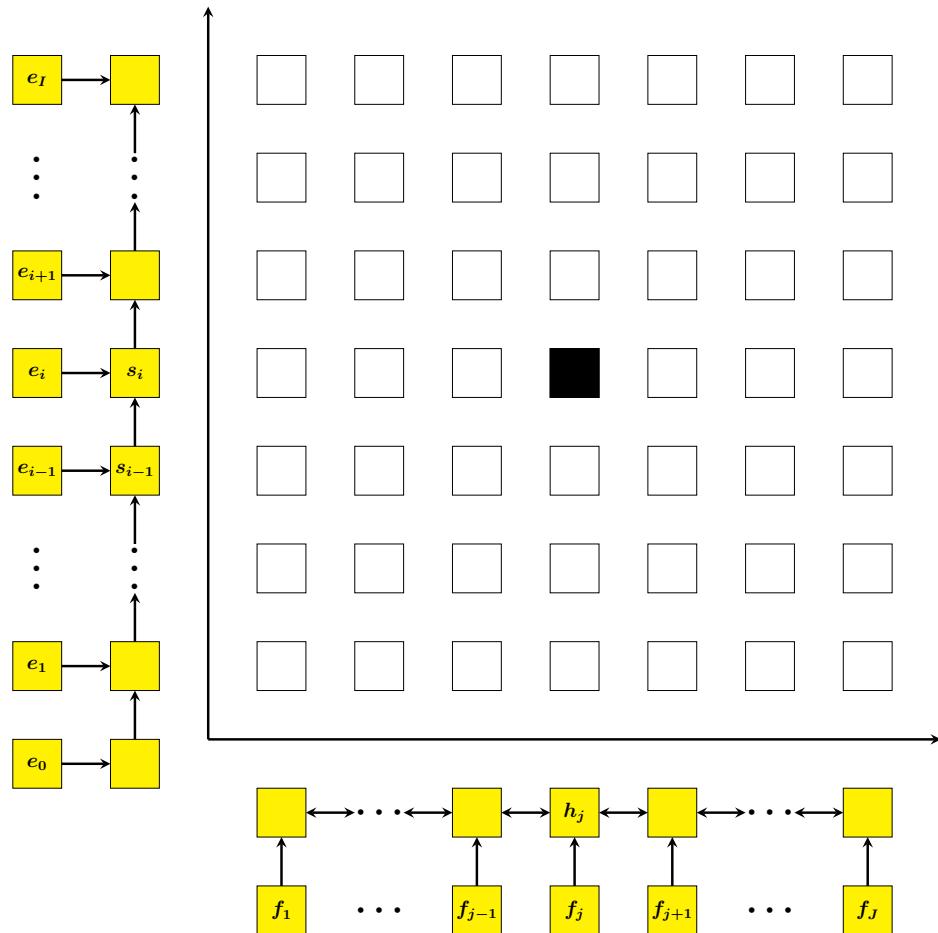
Alignment-based Neural MT:

- novel concept for MT: alignments + NN
 - similar to hybrid HMM in ASR
 - important difference: alignment direction
- ongoing work:
 - stand-alone decoder and end-to-end system
 - LSTM-RNN on source and target sides
 - dependencies: single word, window context, full sentence
- future directions:
 - optimization of all dependence types
 - use of self-attention to replace RNN structures
- theoretical analysis:
relation between alignment and attention models (in MT and ASR)

similar approach in ASR using alignments from state → time:

**Doetsch et al., IEEE Trans. on Signal Processing,
special issue on end-to-end approaches, Sep. 17.**

- **localization modelling:**
 - attention approach
 - HMM approach
- **dependencies:**
 - zero-order
 - first-order
- **present combinations:**
 - attention: zero-order
 - HMM: first-order
- ?? more combinations



Attention-based Neural MT: Comparison with Zero-Order Alignment Model

modelling steps:

- factorization into conditional probabilities:

$$\begin{aligned} p(e_1^I | f_1^J) &= \prod_i p(e_i | e_0^{i-1}, f_1^J) \\ &= \prod_i p(e_i | i, e_{i-1}, s_{i-1}, f_1^J) \end{aligned}$$

with an RNN implementation and state vector s_i

- zero-order alignment model (like IBM-2):

$$\begin{aligned} p(e_i | e_{i-1}, s_{i-1}, f_1^J) &= \sum_j p(j, e_i | i, e_{i-1}, s_{i-1}, f_1^J) \\ &= \sum_j p(j | i, e_{i-1}, s_{i-1}, f_1^J) \cdot p(e_i | e_{i-1}, s_{i-1}, j, f_1^J) \\ &= \dots \end{aligned}$$

with (zero-order) alignment $p(j | \dots)$ and lexicon probabilities $p(e_i | \dots)$

- LSTM RNN representation for h_1^J :

$$h_j = H_j(f_1^J)$$

compare the two approaches with source-side RNN representations $h_j = H_j(f_1^J)$:

- zero-order alignment model (like IBM-2):

$$p(e_i|e_{i-1}, s_{i-1}, h_1^J) = \sum_j p(j|i, e_{i-1}, s_{i-1}, h_j) \cdot p(e_i|e_{i-1}, s_{i-1}, h_j)$$

weighted average: level of probability models

- attention-based approach:

$$p(e_i|e_{i-1}, s_{i-1}, h_1^J) = p(e_i|e_{i-1}, s_{i-1}, c_i)$$

with $c_i = \sum_j \alpha(j|i) \cdot h_j$ and $\alpha(j|i) = A(s_{i-1}, h_j)$

weighted average: level of internal representations h_j

- comparison:
 - main difference is the type of averaging
 - ongoing experiments

Summary: Neural MT

- three concepts for localization in string-to-string processing:
 - attention model
 - transformer approach (= extension of attention model)
 - alignment model in HMM
- characteristic property:
end-to-end modelling and training
- (deep) neural networks: new age of ML ?
 - only *one* example of probabilistic models
(i. e. matrix-vector product + nonlinearity)
 - more powerful than other methods (at present!)

room for improvements along various dimensions:

- stand-alone decoder
- include LSTM-RNN on source side
- include bag-of-words model for window and/or full sentence
- include predecessor position $j' = b_{i-1}$ in lexicon model
- include exponents (in training/testing)
- sharing parameters between lexicon and alignment models (?)
- relation to attention-based neural MT (?)

Conclusions:

An Alignment-based Neural Network Approach to Machine Translation:

- (deep) neural networks: new age of ML ?
 - only *one* example of probabilistic models
 - more powerful than other methods
- explicit alignment model for MT:
 - alignment from target to source:
some practical advantages
 - neural networks for alignment and lexicon models
 - holistic view: spirit of *end-to-end*
- some first promising results,
room for improvements

- deep neural networks: new age of ML ?
only *one* example of probabilistic models
- spirit of *end-to-end* design: holistic view
 - decision process: Bayes decision rule
 - training procedure: open questions
- two questions in *end-to-end* training:
 - suitable training criterion: link to performance criterion
 - numerical optimization strategy
- *end-to-end training* in ASR and SMT: *sequence discriminative training*
bottleneck: optimization strategy
- specific future challenges for ANNs in general:
 - complex mathematical models that are hard to analyze
 - question: can we find suitable mathematical approximations with more explicit descriptions of the dependencies and level interactions and of the performance criterion?
- characters vs. whole words:
combination of both concepts

12 Summary and Conclusions

summary: characteristic properties of this lecture

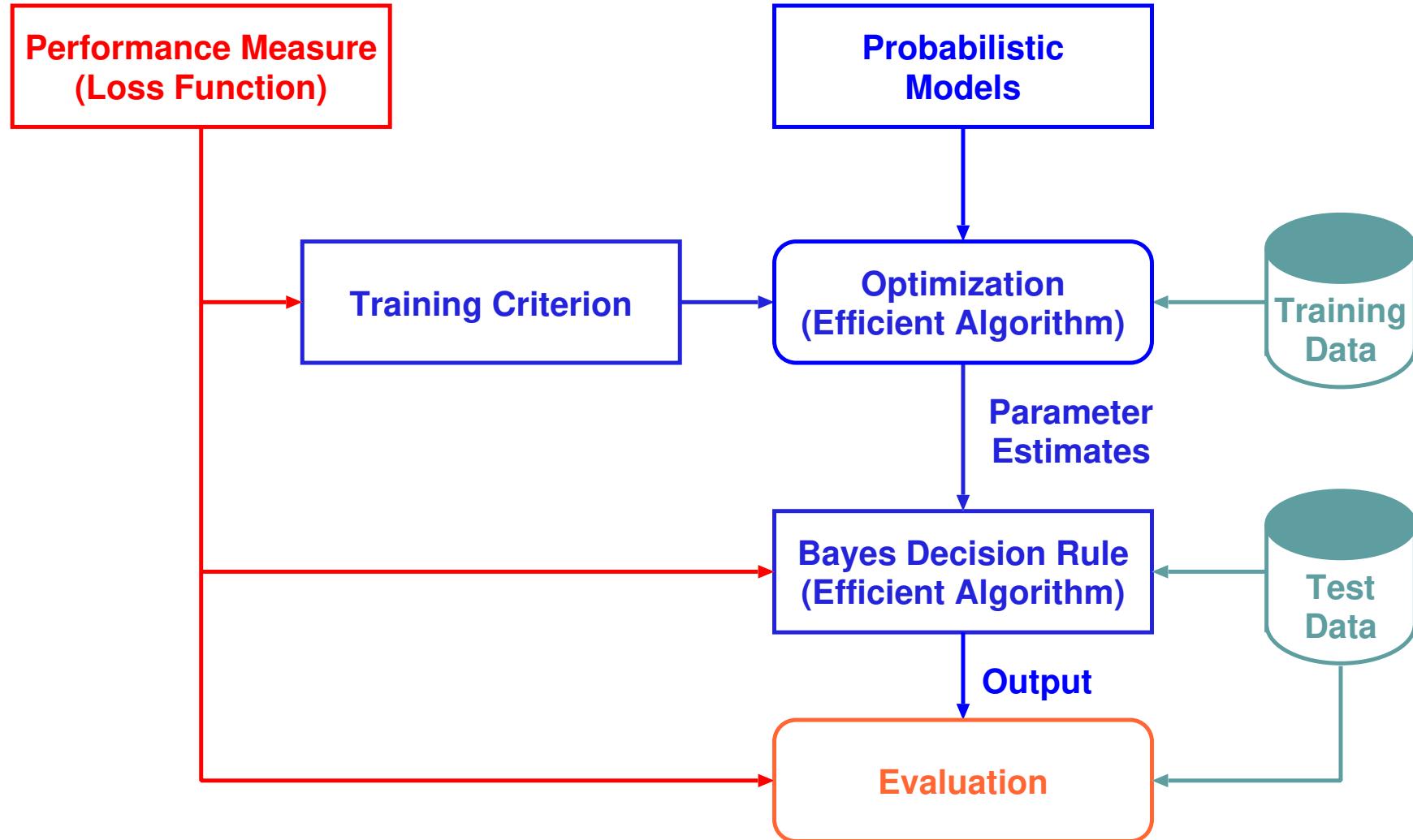
- **application of ANNs to real-life tasks (as opposed to toy tasks):**
 - speech recognition, machine translation, ...
 - important aspect: string-to-string conversion (context information)
- **ANNs have a long history:**
 - started around 30 years ago for speech and language technology
 - real success only since 2011
- **ANNs are part of the statistical approach:**
 - define one family of statistical models (concatenation of dot product and nonlinearities)
 - share many properties with other statistical methods
 - need to be embedded in Bayes decision rule etc.
 - comparison with conventional (non-ANN) approaches

very short summary:

- there has been life before ANNs and deep learning:
Bayes decision rule, Gaussian modelling, HMMs, discriminative training, ...
- ANNs helped to improve the performance dramatically

ASR (and SMT and many other NLP tasks):
mapping from input string to output string

- **statistical approach (inc. ANNs): four key ingredients**
 - choice of performance measure: errors at string, 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 along with an efficient implementation
- **about recent work on artificial neural nets:**
 - they result in significant improvements
 - they provide one more type of probabilistic models
 - they are PART of the statistical approach
- **specific future challenges for statistical approach (incl. ANNs) in general:**
 - complex mathematical model that is difficult to analyze
 - questions: can we find suitable mathematical approximations with more explicit descriptions of the dependencies and level interactions and of the performance criterion (error rate)?
- **specific challenges for ANNs:**
 - can the HMM-based alignment mechanism be replaced?
 - can we find ANNs with more explicit probabilistic structures?



questions and interpretations:

- **Do the ANN discover dependencies that we cannot model explicitly?**
- **Is it a better way of smoothing that makes the ANN better?**
- **Is it the use of crossvalidation that makes ANN successful?**
- ...

13 References

- [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.
- [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 95] 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 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 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.

- [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.
- [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.
- [Cortes & Kuznetsov⁺ 18] C. Cortes, V. Kuznetsov, M. Mohri, D. Storcheu, S. Yang: Efficient Gradient Computation for Structured Output Learning with Rational and Tropical Losses. *NIPS 2018*.
- [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. 69-477.
- [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.
- [Duda & Hart⁺ 01] R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification. 2nd ed., J. Wiley & Sons, New York, NY, 2001.
- [Fedotov & Harremoes⁺ 03] A. A. Fedotov, P. Harremoes, F. Topsøe: 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.
- [Gibson & Hain 06] M. Gibson, T. Hain: Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition. *Interspeech*, pp. 2406-2409, Sep. 2006.
- [Goldberg 17] Y. Goldberg: Neural Network Methods in Natural Language Processing. Morgan & Claypool Publishers, 2017.
- [Graves & Fernandez⁺ 06] A. Graves, S. Fernández, F. Gomez, J. Schmidhuber: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. *Int. Conf. on Machine Learning*, Pittsburgh, USA, 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.
- [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.
- [Heigold & Macherey⁰⁵] 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 & Schlüter¹²] 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. A58-69, 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.
- [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.
- [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.
- [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.
- [Lampl & Ballesteros⁺ 16] G. Lampl, M. Ballesteros, S. Subramania, K. Kawakam, C. Dyer: Neural Architectures for Named Entity Recognition. *Proc. NAACL-HLT 2016*, pp. 260-270, San Diego, CA, June 2016.
- [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 & Karafiat⁺ 10] T. Mikolov, M. Karafiat, L. Burget, J. Černocký, S. Khudanpur: Recurrent neural network based language model. Interspeech, pp. 1045-1048, Makuhari, Chiba, Japan, Sep. 2010.
- bibitem [Mohamed & Dahl⁺ 09] MohamedDahl09 A. Mohamed, G. Dahl, G. Hinton: Deep belief networks for phone recognition. NIPS Workshop Deep Learning for Speech Recognition and Related Applications, 2009.
- [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.
- [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.
- [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.
- [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.
- [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. Sejnowski, *Neural Networks: Theory and Applications*, pp. 111-126, Morgan Kaufmann, San Mateo, CA, 1990.

Hinton (eds.): *Proceedings of the 1990 Connectionist Summer School*, pp. 159-172, San Mateo, CA, Morgan Kaufmann.

[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.

[Plank & Søgaard⁺ 16] B. Plank, A. Søgaard, Y. Goldberg: Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss, Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 412–418, Berlin/Germany, Aug. 2016.

[Printz & Olsen 02] H. Printz, P. A. Olsen: Theory and practice of acoustic confusability. Computer Speech and Language, pp. 131–164, Jan. 2002.

[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.

[Schlüter & Nussbaum⁺ 11] R. Schlüter, M. Nussbaum-Thom, H. Ney: On the Relationship between Bayes Risk and Word Error Rate in ASR. IEEE Transactions on Audio, Speech, and Language Processing, Vol. 19, No. 55, pages 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. Alkhouri, 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. European Conf. on Speech Communication and Technology, Interspeech, pp. 1449-1452, Lisbon, Portugal, Sep. 2005
(for corrections, see paper version on RWTH i6 webpage).

- [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.
- [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.
- [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 & Alkhouri⁺ 14] M. Sundermeyer, T. Alkhouri, 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.
- [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, August 2011.
- [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.
- [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, WA, Oct. 2013.
- [Vaswani & Shazeer⁺ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser: Attention Is All You Need. (Google’s transformer approach to MT) archive, 06-Dec-2017.
- [Vesely & Ghoshal⁺ 13] K. Vesely, A. Ghoshal, L. Burget, D. Povey: Sequence-discriminative training of deep neural networks. *Interspeech* 2013.
- [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.
- [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.
- [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.
- [Zhou & Stolcke⁺ 04] Q. Zhu, A. Stolcke, B. Y. Chen, N. Morgan: Incorporating tandem/HATs MLP features into SRI's conversational speech recognition system. Proc. DARPA Rich Transcription Workshop, 2004.

END

DeepLearn Warsaw Poland
24/25-Jul-2019