RWTH AACHEN UNIVERSITY

UP Valencia, DSIIC, Master Course 2024:

Advanced Machine Learning

Lectures April 2024:

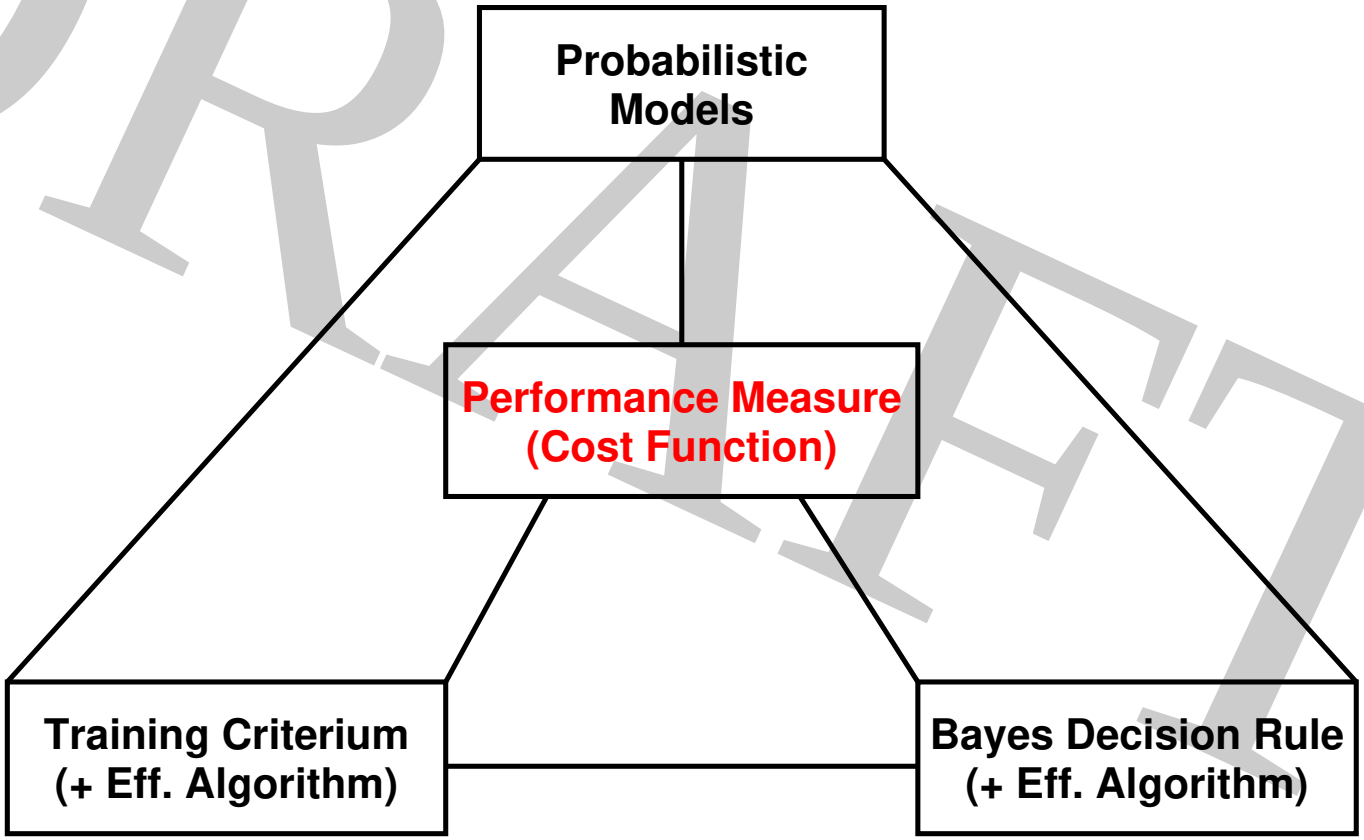
Machine Learning and Bayes Decision Theory: A Unifying View

Hermann Ney

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

HLT

central role of performance measure or classification error in sequence processing for ASR, MT and other NLP tasks:



These Lectures

related: classes at UPV in January 2024:
– model structures

these lectures: emphasis on:
- prinicples of general ML
- sequence processing in general
- role of (language model) prior in sequence modelling
- training criteria
- optimality/justification of training criteria
- relation to Bayes decision rule
- mathematical justification with details

why are we doing what we are doing?

– relation to classification error and Bayes decision rule
– interdependencies of concepts
– not mainstream
– rarely addressed in papers and textbooks

Outline

1 Introduction 11
1.1 Sequence Processing . . . 11
1.2 Preview . . . 26

2 Machine Learning: A Tentative Synopsis 34
2.1 Principles . . . 34
2.2 Training Data and Empirical Distribution . . . 42
2.3 Training Criteria . . . 46
2.3.1 Squared Error . . . 48
2.3.2 Cross-Entropy . . . 57
2.3.3 Binary Cross-Entropy . . . 62
2.4 From Single Symbols to Symbol Sequence Processing . . . 71
2.4.1 RNN: Recurrent Neural Networks . . . 73
2.4.2 Synchronized Strings and Tagging . . . 92
2.4.3 Cross-Attention and Transformer . . . 96
2.4.4 Finite-State Transducer: HMM, CTC, RNN-T . . . 101
2.5 Sequence Processing: Conclusions . . . 105

3 Sequences: Modelling and Training 110
3.1 Principle: From Single Symbols to Strings . . . 110
3.2 Cross-Entropy at Sequence Level . . . 117
3.3 From Generative to Discriminative Models . . . 121
3.4 Separation of Prior . . . 127
3.5 Model with Explicit LM . . . 137
3.6 Revisit Separation of Prior LM . . . 141

4 Bayes Decision Rule: Principle 143
4.1 Decision Rules and Loss Function . . . 144

4.2 Empirical Distribution . . . 150
4.3 Bayes Decision Rule . . . 154
4.4 Conclusions . . . 159

5 Bayes Decision Rule: Loss Function 160
5.1 Atomic Output: Single Symbol . . . 161
5.2 Structured Output: String with Synchronization . . . 164
5.3 Structured Output: String with no Synchronization . . . 171
5.3.1 Pathological Example: Bayes String with Zero Probability . . . 172
5.3.2 Basic Inequality for Metric Loss Function . . . 178
5.3.3 Rewrite Posterior Loss . . . 183
5.3.4 Edit Distance and Symbol Posterior Probability . . . 187
5.3.5 Summary . . . 193
5.3.6 DRAFT: Redefine Loss: Frame Error vs. Edit Distance . . . 194

6 Classification Error and Training Criterion 198
6.1 Introduction . . . 198
6.2 Probability Models and Decision Rule . . . 206
6.3 Classification Errors: Two Types . . . 210
6.4 Training Criteria and Bounds . . . 215
6.4.1 Squared Error . . . 216
6.4.2 Binary Divergence . . . 220
6.4.3 Kullback-Leibler Divergence . . . 222
6.5 Training Criteria Revisited: Squared Error vs. Cross-Entropy . . . 224

7 Computer Simulations and Tight Error Bounds 235
7.1 Principle . . . 235
7.2 Three Tight Bounds . . . 235
7.3 Statistics and Information Theory . . . 235

7.4 Bounds with Constraints A_* 235

8 Refinement: Classification Error Bounds 236
8.1 From Discriminative to Generative Training 236
8.2 Hamming Distance: Synchronized Strings 242
8.3 General Loss Function 247
8.4 Conclusion 256

9 DRAFT: Summary and Conclusions 257

10 References 261

Similar Lectures

adapted from:

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

slides: UP Valencia, classes 2017-2021

- **master course "Advanced Machine Learning"**

slides: summer schools on *Deep Learning*

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

Lectures: Concept

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

  [as opposed to image (object, face, ...) recognition]
- sequence-to-sequence processing
  – as opposed to isolated events/images
  – sequence context is important
- models of sequence probabilities (ASR, HTR, MT):
  – synchronization: FST (HMM, CTC, transducer) and attention
  – language model: captures the context of output sequence
- justification of approaches:
  – Bayes decision rule and statistical framework
  – study training criteria

Lectures: Concept

my presentation:

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

problems:

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

Lectures: Concept

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

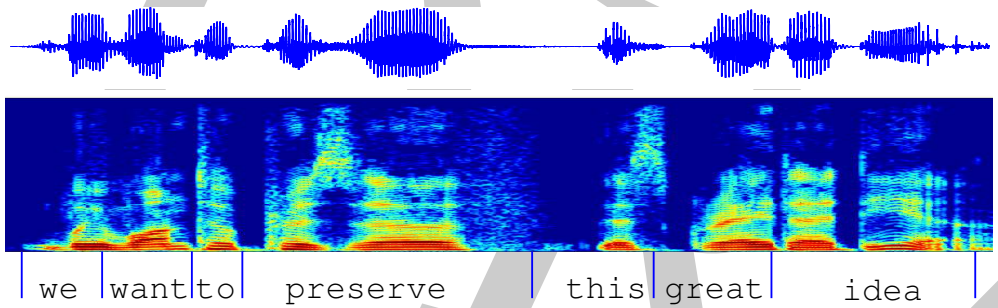
1.1 Sequence Processing

outline:

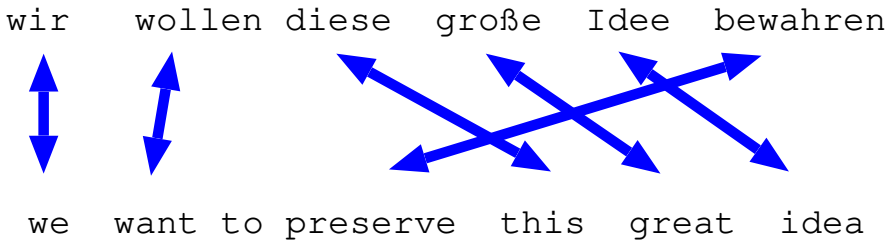
– terminology: speech vs. language
– speech and language technology
– seq-to-seq processing

Speech & Language Technology: Sequence-to-Sequence Processing

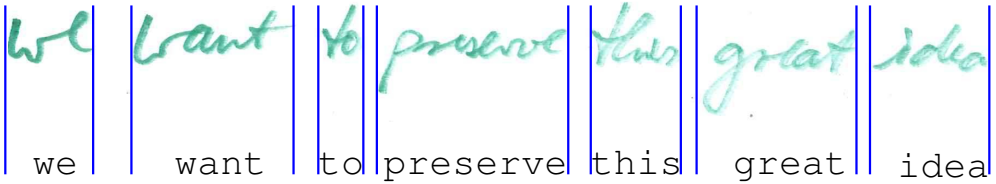
Automatic Speech Recognition (ASR) (speech signal processing)



Machine Translation (MT) (symbol or text processing)



Handwritten Text Recognition (HTR) (text image processing)



common characteristics:

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

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

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; speech and language technology**
- **specific well-defined tasks in HLT:**
  - **automatic speech recognition (ASR)**
  - **text image recognition (printed and handwritten text) (HTR)**
  - **machine translation (MT) (of language and speech)**

characteristic properties of ASR: :

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

Overview and History: HLT and AI

- AI: artificial intelligence:
  a computer that performs human tasks like ASR, MT and other HLT tasks
  (prototypical tasks of AI)
- 1970-2010: rule-based approaches for AI (mainly text/symbolic)
  facts and formal derivations/proofs: like PROLOG programming
- (wide-range) success of artificial neural nets around 2012/2013: deep learning
  today: AI = data-driven AI = deep learning

today's terminology: generative AI (vs. formal AI/NLP)
- generative NLP tasks: input and output in natural language
  (form of natural language varies: spoken, handwritten, digital text)
- formal NLP tasks: output requires a specific formalism

common goals of both fields:

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

differences in practice:

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

Large-Scale Projects 1984-2020

- SPICOS 1984-1989: speech recognition und understanding
  - conditions: 1000 words, continuous speech, speaker dependent
  - funded by German BMBF: Siemens, Philips, German universities
- Verbmobil 1993-2000: funded by German BMBF
  - domain: appointment scheduling, recognition and translation, German-English, limited vocabulary (8.000 words)
  - large project: 10 million DM per year, about 25 partners
  - partners: Daimler, Philips, Siemens, DFKI, KIT Karlsruhe, RWTH, U Stuttgart, ...
- TC-STAR 2004-2007: funded by EU
  - domain: recognition and translation of speeches given in EU parliament
  - task: speech translation: ASR + MT (+TTS)
  - challenge: MT robust wrt ASR errors $\rightarrow$ data-driven methods
  - first research prototype for unlimited domain and real-life data
    - fully automatic, not real time
    - without deep learning!
  - partners: KIT Karlsruhe, RWTH, CNRS Paris, UPC Barcelona, IBM-US Research, ...
- GALE 2005-2011: funded by US DARPA
  - recognition, translation and understanding for Chinese and Arabic
  - largest project ever on HLT: 40 million USD per year, about 30 partners
  - US partners: BBN, IBM, SRI, CMU, Stanford U, Columbia U, UW, USCLA, ...
  - EU partners: CNRS Paris, U Cambridge, RWTH

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

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

ASR and MT: Real-Life Applications and Tasks

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

hardware products:

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

History: ASR and ANNs

very short history:

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

ASR: first research 1975–1980

ASR is sequence-to-sequence processing:

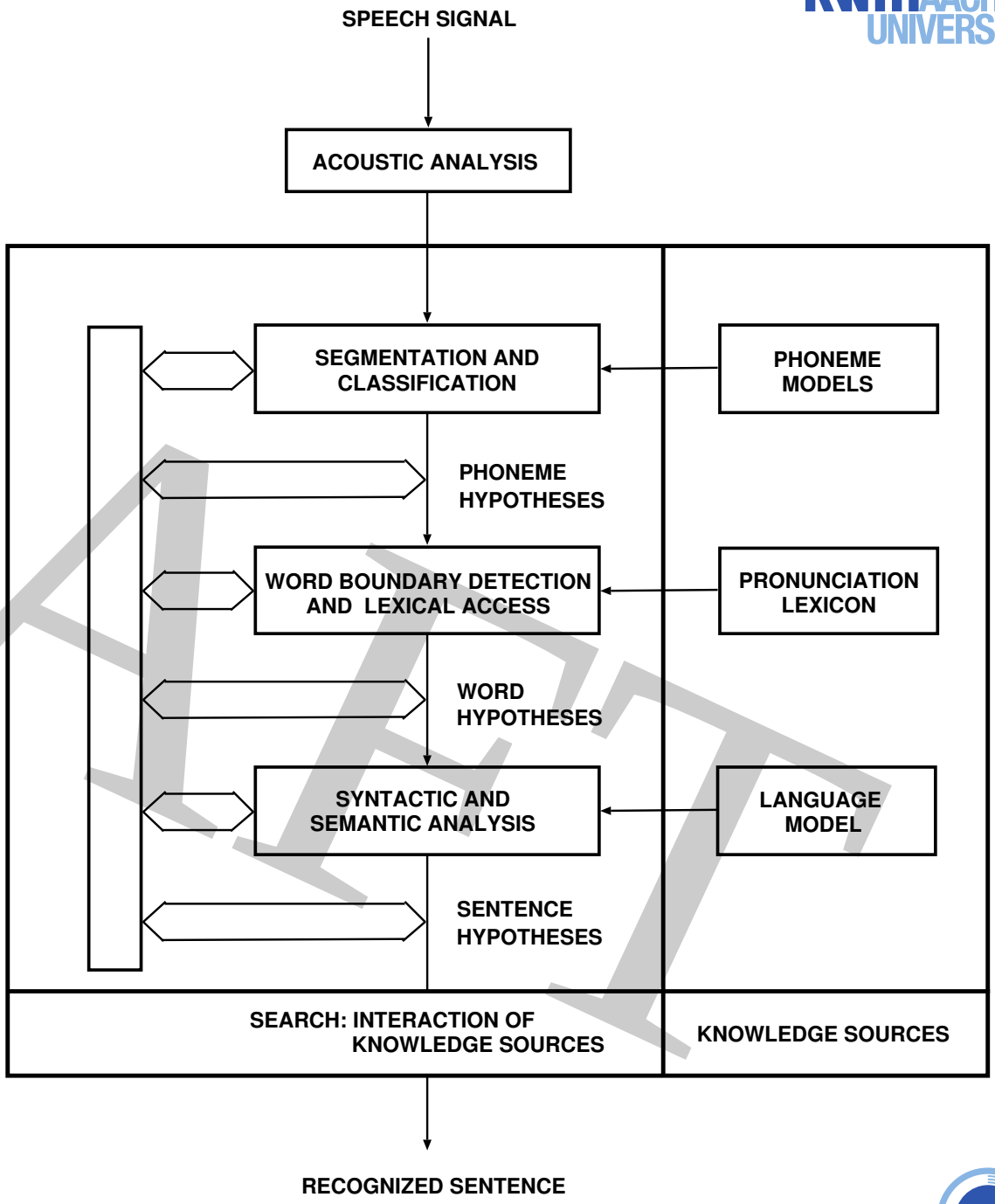
- sequence of 10-ms acoustic vectors
- sequence of sounds/phonemes
- sequence of letters
- sequence of words

problems:

- ambiguities at all levels
- interdependencies of decisions

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

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

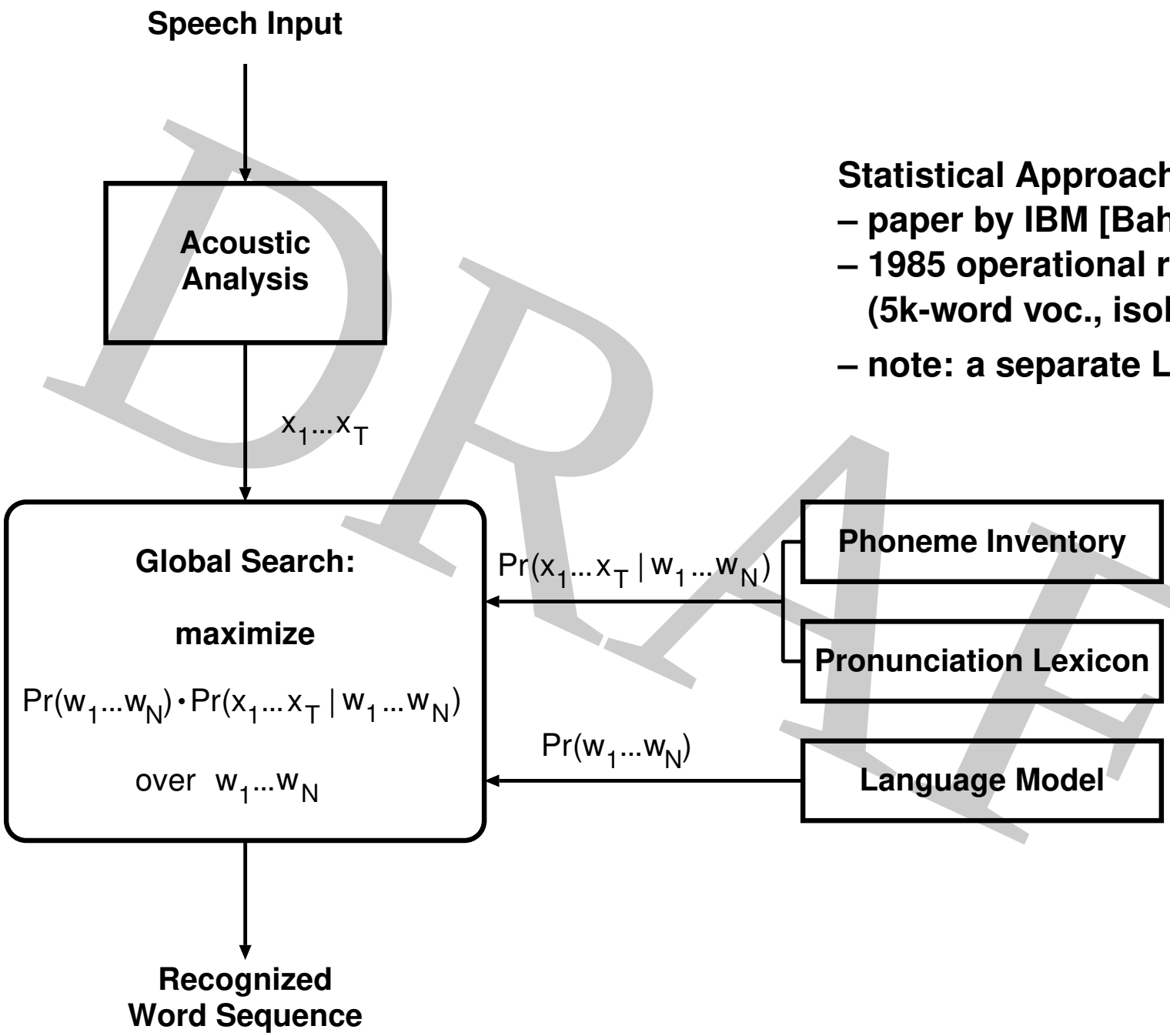


- modelling: probability distributions/data-driven approaches with

  10-msec vectors: $x_1^T = x_1 ... x_t ... x_T \quad x_t \in \mathbb{R}^D$

  word string: $w_1^N = w_1 ... w_n ... w_N$

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



Statistical Approach to ASR

– **paper by IBM [Bahl & Jelinek$^+$ 83]**

– **1985 operational research system: *Tangora* (5k-word voc., isolated words, speaker dep.)**

– **note: a separate LM**

ASR History: Operational Research Systems

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

alternative concepts (with less success):

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

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

ASR: ANN in Acoustic Modelling

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

Neural ASR: Tandem vs. Hybrid HMM

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

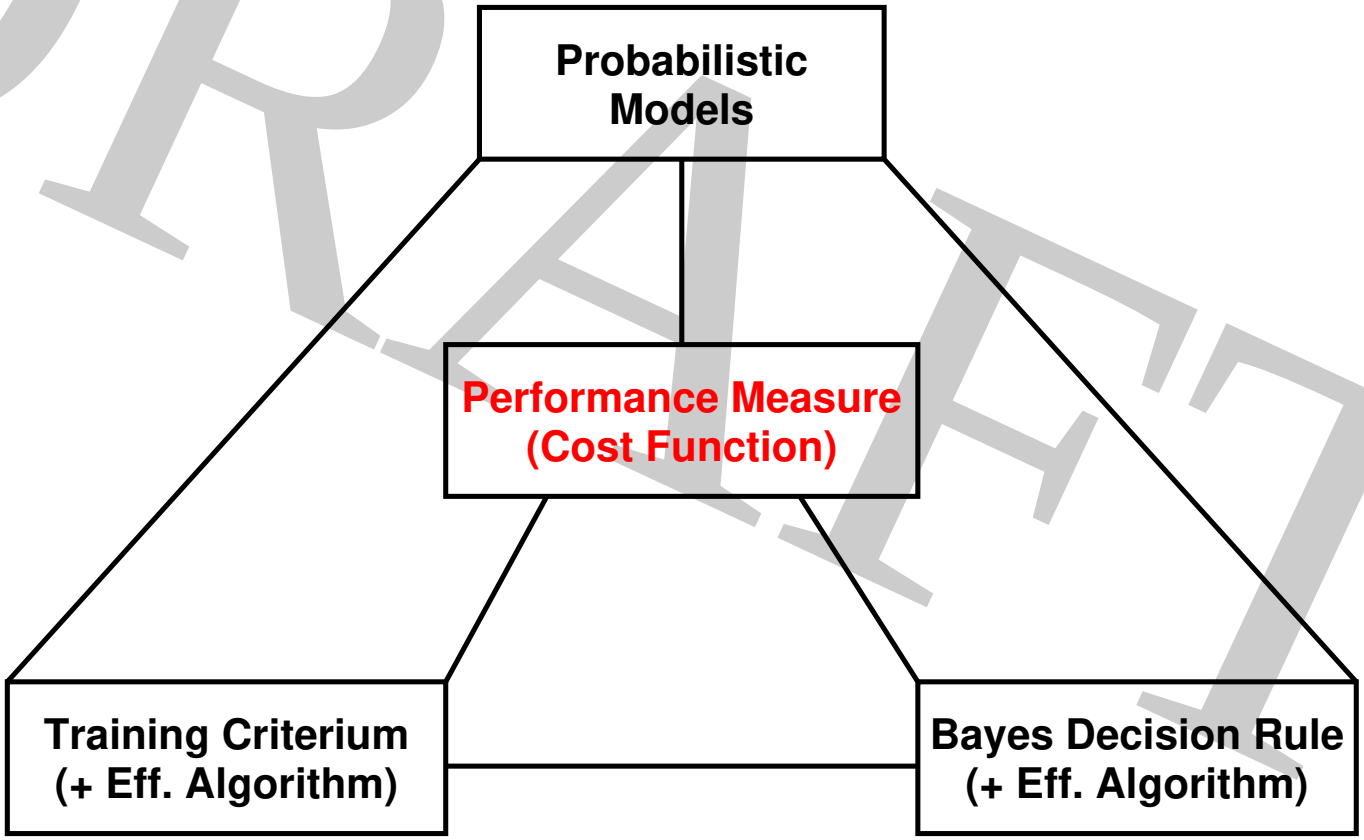
- 2009 [Graves 09]: CTC - good results on LSTM RNN for handwriting task
- 2010 [Dahl & Ranzato$^+$ 10]: improvement in phone recognition on TIMIT
- 2011 [Seide & Li$^+$ 11, Dahl & Yu$^+$ 12]: Microsoft Research
  - fully-fledged hybrid HMM
  - 30% rel. WER reduction on Switchboard 300h
- since 2012: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

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

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

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

central role of performance measure or classification error in sequence processing for ASR, MT and other NLP tasks:



principles [textbook by Duda & Hart 1973, p. 11-16]:

- **two strings: input $x_1^T := x_1 ... x_t ... x_T$ and output $c_1^N := c_1 ... c_n ... c_N$ 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 $p(c_1^N)$**

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

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

System Components

an sequence processing (ASR, MT, HLT) system has several components:

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

view: various ways of combinining these components

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

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

HLT Systems: Components

principal components:

- **performance measure, error measure, cost function:**
  - **how to judge the quality of the system output**
  - **examples: ASR: edit distance; MT: TER or BLEU**

  (later: relation to associated decision/generation rule)
- **probabilistic models (with a suitable structure)**
  for capturing the dependencies within and between input and output strings:
  - **given synchronization:**
    Markov chain, CRF, (LSTM) RNN, ...
  - **input/output synchronization:**
    generative/hybrid HMM, CTC, transducer, attention (incl. transformer, ...
  - **separation of acoustic and language models vs. *end-to-end* concepts**

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

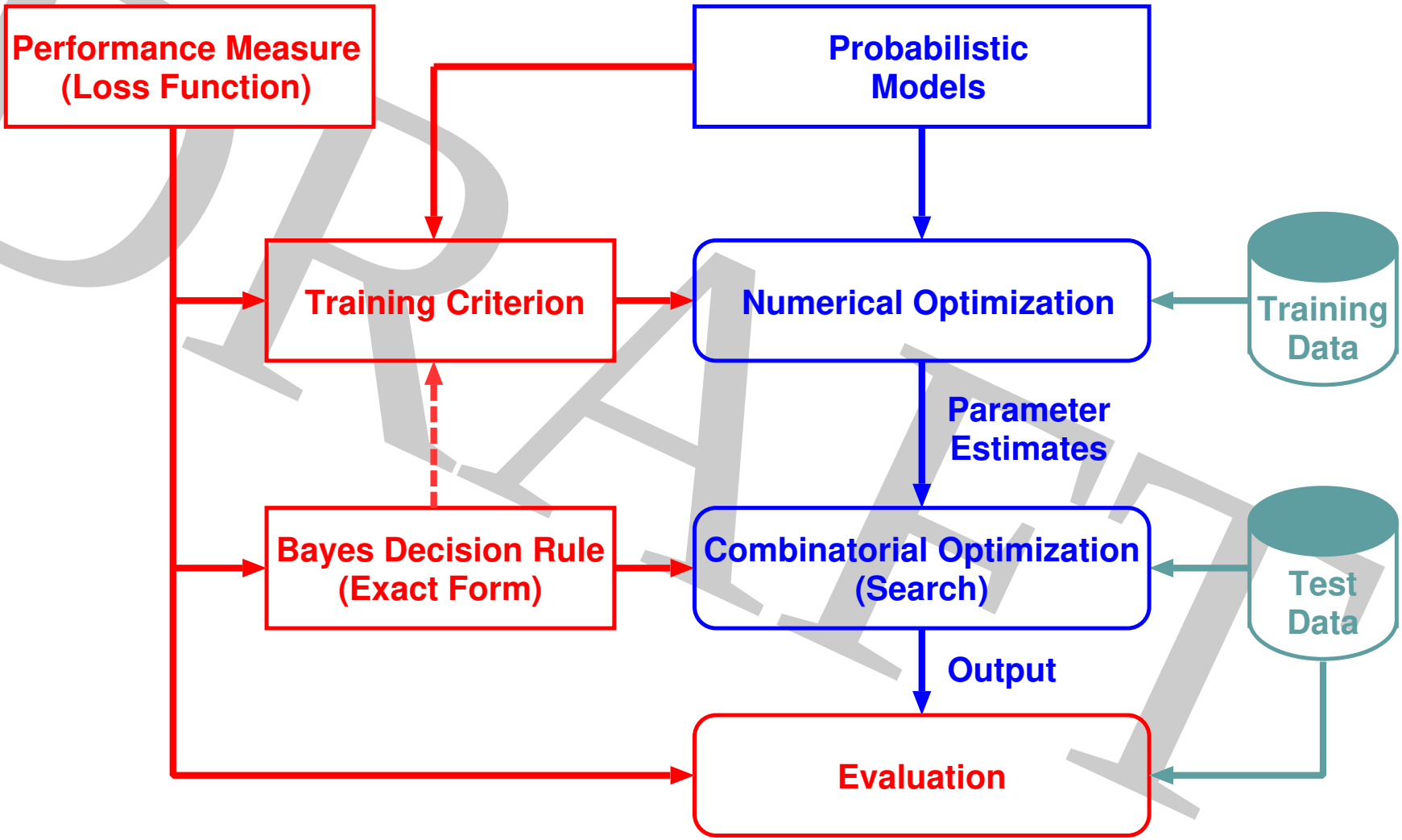
  examples of training criteria:
  – maximum likelihood
  – cross-entropy
  – sum criterion (CTC) and EM algorithm
  – sequences discriminative training (and epxected Bayes risk)

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

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

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

Architecture for Sequence Processing



Open Questions

– why do we use Bayes decision rule for ASR ?
– what is the relation of the ANN framework with Bayes decison rule?
– what is the role of softmax output layer in ANNs ?
– what is the relation of training criteria with Bayes decison rule/classification error ?
– what about using the exact loss function in Bayes decision rule?
– what is the relation between training criteria and end-to-end modelling ?
– why should we use an explicit language model in HLT tasks like ASR and MT?

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'):**

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

RWTH AACHEN UNIVERSITY

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)?

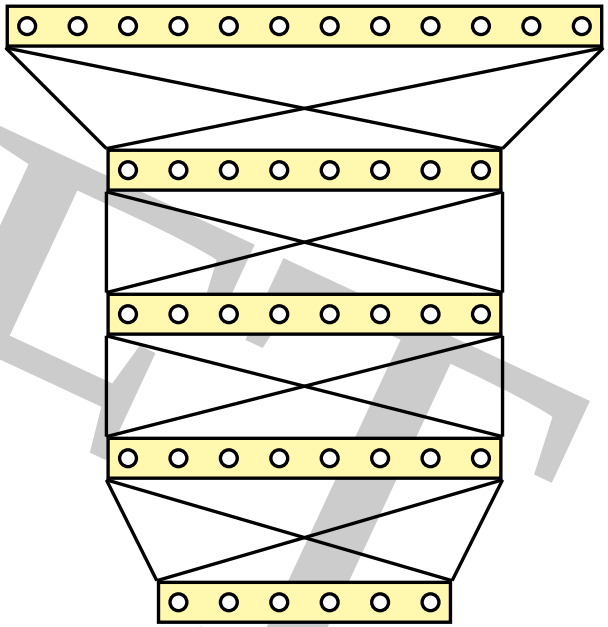
HLT

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

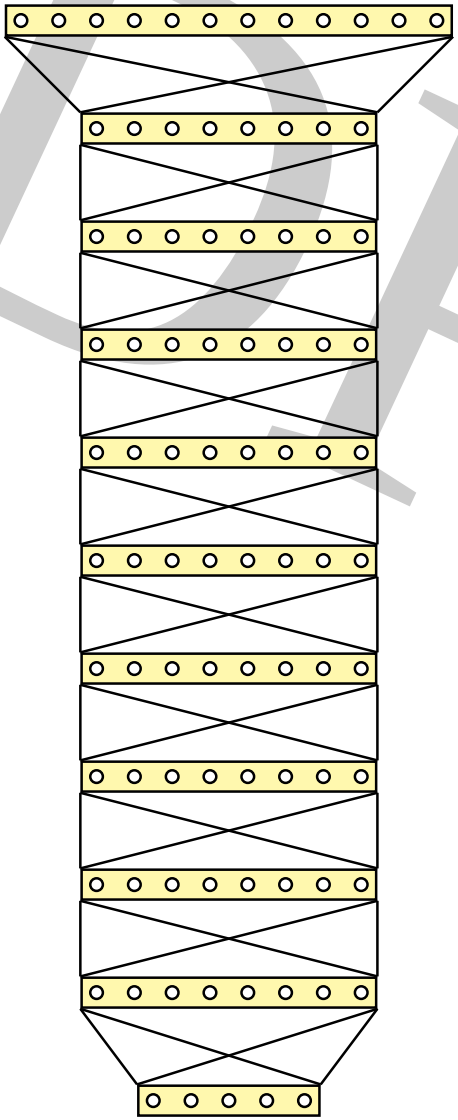
typical ANN structure:
MLP: feedforward multi-layer perceptron
with input, hidden and output layers

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

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



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



question: what is different now after 30 years?

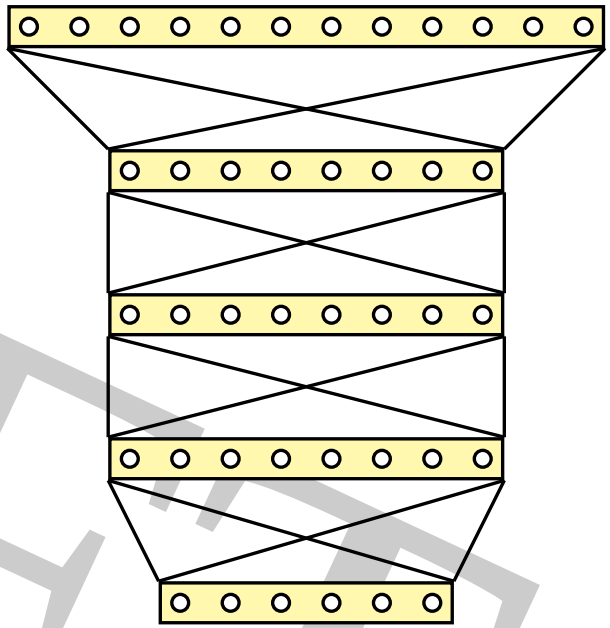
answer: we have learned how to (better) handle a complex mathematical optimization problem:

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

Activation Functions for ANN

examples of activation functions:

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

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

- **hyperbolic tangent:**

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

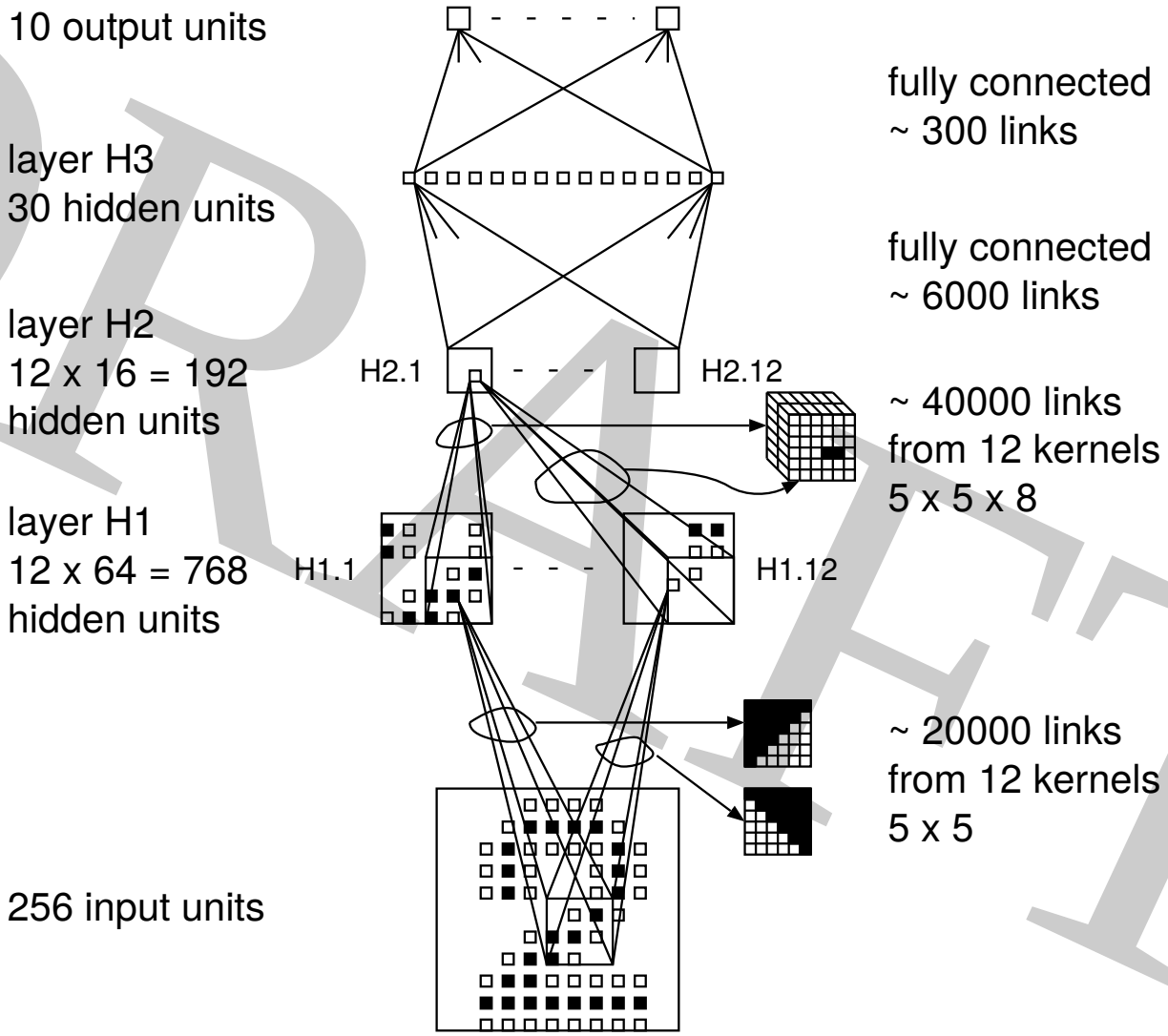
  - **in principle: no difference between $\sigma(\cdot)$ and $\tanh(\cdot)$**
  - **in practice: difference due to side effects**

- **rectifying linear unit (ReLU): $u \rightarrow r(u) = \max\{0, u\}$**
  so far: not useful in symbolic processing (?)

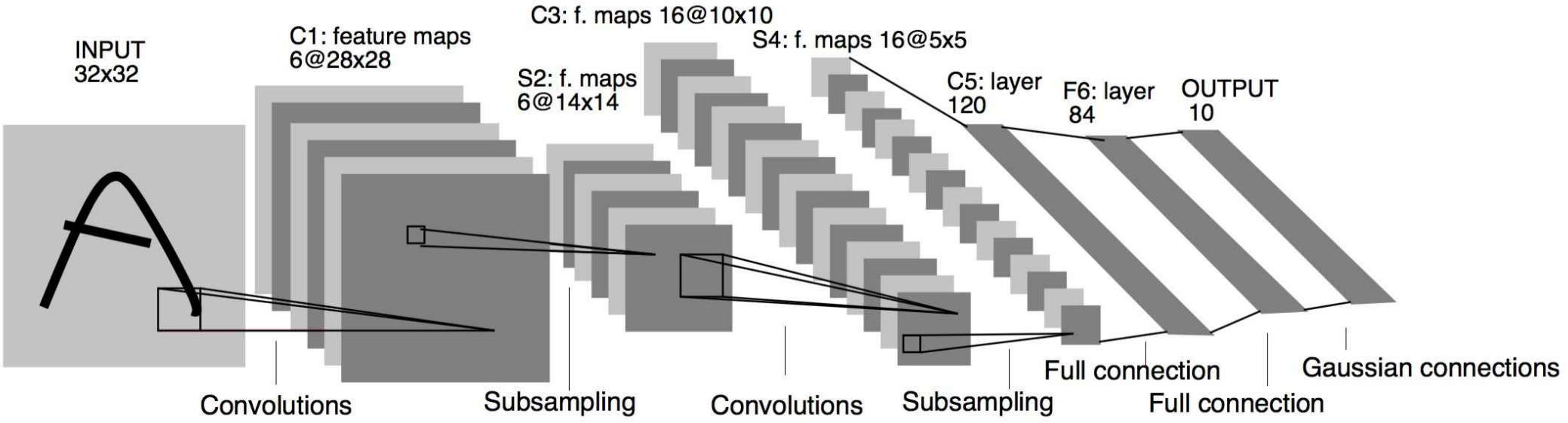
- **softmax function: generates normalized output (for probability distribution)**
  for each node c of the layer under consideration (typically: output layer):

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

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



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



2.2 Training Data and Empirical Distribution

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

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

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

with $\delta(\cdot, \cdot)$ being Kronecker delta (or Dirac delta function for $x \in \mathbb{R}^D$)

interpretations: histogram (or big table), counting, kernel density

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

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

note: the empirical distribution summarizes EVERYTHING about the training data

Types of Empirical Distributions

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

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

Empirical Distribution: Continuous-Valued Observations

- **continuous-valued observations:** $x \in \mathbb{R}^D$
  D**-dim. vector space**
- **replace Kronecker delta by Dirac's delta function**
  $$\delta(x - x_n) \quad \text{for } x, x_n \in \mathbb{R}^D$$
- **approximation by kernel function**
  $$h(u), \quad u = x - x_n$$
  that is unimodel (a single optimum)
  and concentrated around $u = 0$

  typical example: Gaussian (density) distribution
- **approach: known as *kernel density estimation***

Empirical Distribution: Remarks on the Concept

advantage of using empirical distribution:

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

- conceptual view of decision rule:

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

2.3 Training Criteria

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

ANN outputs and associated training criteria:

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

preview of important results:

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

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

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

Three Types of ANN Outputs

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

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

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

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

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

note: the decision output does not change:

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

2.3.1 Squared Error

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:**

$$\begin{array}{ll} 1 & \text{for correct class } c = c_n \\ 0 & \text{for all rival classes } c \neq c_n \end{array}$$

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:

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

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 $F(\vartheta|x)$:

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

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:

$$\sum_c p_c \sum_{c'} [q_{c'} - \delta_{c'c}]^2 = \sum_c p_c \sum_{c'} \left[q_{c'}^2 - 2\, q_{c'}\, \delta_{c'c} + \delta_{c'c}^2\right]$$

$$= \text{... (carry out sums over } c \text{ and } c'\text{)}$$

$$= \sum_c q_c^2 - 2 \sum_c p_c\, q_c + 1 = \sum_c \left[q_c^2 - 2\, p_c\, q_c\right] + 1$$

$$= \sum_c [q_c - p_c]^2 + 1 - \sum_c p_c^2$$

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

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

$$= \sum_c p_c \cdot [1 - p_c] = \sum_c p_c \sum_{c' \neq c} p_{c'}$$

Training Criterion: Squared Error

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!

Remark: Parametric vs. Non-Parametric Approaches

distinguish two types of modelling approaches:

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

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

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

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

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

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

Squared Error Criterion: Remarks

remarks:

– two types of squared error:
  a) error between model and true distribution
  b) error between model and ideal targets
  difference between the two types: Gini criterion/index

– Gini criterion/index: measures the smoothness/concentration of a distribution

– other measure: (Shannon) entropy

– Gini index and (Shannon) entropy:
  are used for the design of decision trees (CART)

decision trees or CART (classification and regression trees)

we apply the mimimum 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:

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

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

- **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.

Squared Error Criterion: Quadratic Normalization

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:

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

Squared Error Criterion: Quadratic Normalization

for quadratic normalization, we have shown:

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

training criterion for parameter ϑ :

$$\hat{\vartheta} = \operatorname*{argmin}_\vartheta F(\vartheta) = \operatorname*{argmax}_\vartheta \left\{ \sum_{c,x} pr(c,x)\, p_\vartheta(c,x) \right\}$$
$$= \operatorname*{argmax}_\vartheta \left\{ \sum_n p_\vartheta(c_n, x_n) \right\}$$

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

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

result: a re-normalized class posterior

2.3.2 Cross-Entropy

more appropriate term: logarithm of class posterior probability

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

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

note: different notation to express normalization

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

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

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

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

Training Criterion: Cross-Entropy

criterion is equivalent to (negative) cross-entropy:

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

training criterion for parameter ϑ :

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

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

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

note: normalized output is required!

Cross-Entropy: Proof

consider the equivalent maximization problem:

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

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

terminology from information theory
(warning: misleading/wrong interpretations in internet blogs):

$$\text{(self) entropy:} \quad -\sum_c pr(c|x) \log pr(c|x)$$

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

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

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

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

consider two distributions p_c, q_c over a random variable c

divergence inequality:

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

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:

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

$$\hat{q}_c = p_c$$

Alternative Training Criterion: Pathological Estimate (Dropping the Logarithm)

- usual assumption: normalized model $q(c|x)$ with cross-entropy training:

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

- tentative training criterion:
  drop the logarithm in cross-entropy criterion:

$$\begin{aligned} F(\{q(c|x)\}) &= 1/N \cdot \sum_n q(c_n|x_n) = \sum_x pr(x) \sum_c pr(c|x)\, q(c|x) \\ \hat{q}(c|x) &:= \underset{\{q(c|x)\}}{\operatorname{argmax}} \Big\{ F(\{q(c|x)\}) \Big\} \\ &= \delta(c, c_*(x)) = \begin{cases} 1 & \text{if } c = c_*(x) = \operatorname{argmax}_{c'}\, pr(c'|x) \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

- result: peaky model,
  i. e. a pathological estimate, which is not useful.

2.3.3 Binary Cross-Entropy

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:

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

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

Binary Cross-Entropy: Proof

consider the equivalent maximization problem:

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

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

Training Criteria: Summary

important result:

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

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

remarks:

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

Preview: Training Criterion and Classification Error

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

relation between error rate and training criteria?

- **we need a strict distinction:**
  – error rate for the true distribution: Bayes classification error E_*
  – error rate for the learned distribution: model classification error E_ϑ (model distribution with parameters ϑ)
- **we will prove later [Ney 03]:**
  each of the three training criteria gives a tight upper bound to the squared difference between these two error rates

  example: Kullback-Leibler divergence for cross-entropy criterion

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

Empirical Distribution: Generalization

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

- **central role: empirical distribution,**
  - **it is non-zero only for the observed pairs (x_n, c_n) in the corpus**
  - **for unseen pairs (x,c): $pr(x,c) = 0$**
  - **for unseen observations x: $pr(x) = 0$**
- **assumption for future experiments:**
  approach should work for all inputs x
- **the training corpus can cover only a very very tiny fraction of all possible inputs**
  consider the example of speech recognition:
  1 sec of audio = 100 10-ms frames, each frame with a vector $x \in \mathbb{R}^{D=50}$:
  number of possible input strings: $\left(\left(2^8\right)^{50}\right)^{100} = 2^{8 \cdot 50 \cdot 100} = 2^{40.000} \cong 10^{12.000}$

concept of generalization:

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

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

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

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

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

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

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

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

distinguish strictly:

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

gradient search (incl. backpropagation):

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

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

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

note: there have been hundreds of variants

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

Conclusions: Three Training Criteria

three training criteria:

– each of them: class posterior probability
– role of class prior (later: LM)
– if data are very noisy: $\hat{p}_\theta(c|x) = pr(c)$
note: bias in softmax for ANNs

preview:
– each of them is related to the classification error
– from single events to sequences:
cross-entropy has advantages

- sequence/string processing:
  – input sequence
  – output sequence
- model of posterior string probability:

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

  we are looking at the models as a whole;
  interior details not so important
- various concepts for decomposition into more basic structures:
  – decomposition into prior LM and AM
  – factorization over output positions:

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

**tentative overview of practical realizations,
nowadays ANN and deep learning structures:**

- **RNN: elemanary concept:
how to get from single symbols to symbol sequences**
- **cross-attention architecture:
synchronization or time alignment**
- **transformer:
self-attention and cross-attention**
- **finite state transducer or HMM:
hidden variable: state sequence for synchronization**

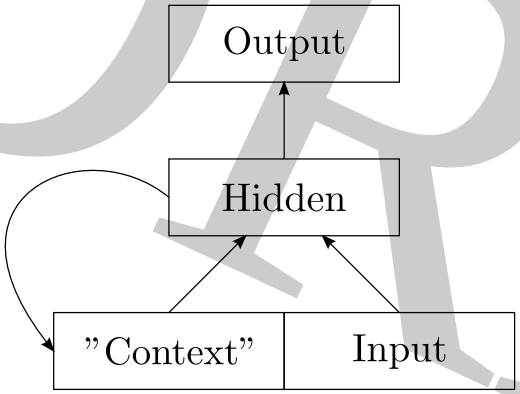
2.4.1 RNN: Recurrent Neural Networks

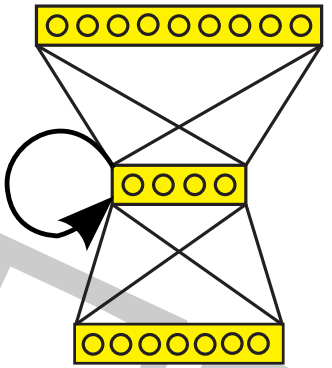
- assumption:
  synchronized input and output positions
- RNN: recurrent neural network
  maybe with LSTM or other extensions
- question: what variants of RNNs?
  and what do they model and learn?

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

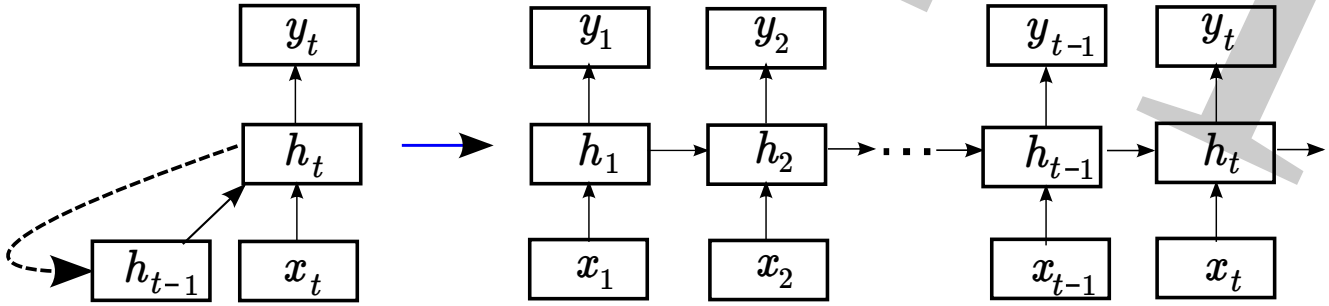
observations x_1^T:	x_1	x_2	...	x_{t-1}	x_t	x_{t+1}	...	x_{T-1}	x_T
	|	|	|	|	|	|	|	|	|
class labels c_1^T:	c_1	c_2	...	c_{t-1}	c_t	c_{t+1}	...	c_{T-1}	c_T

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



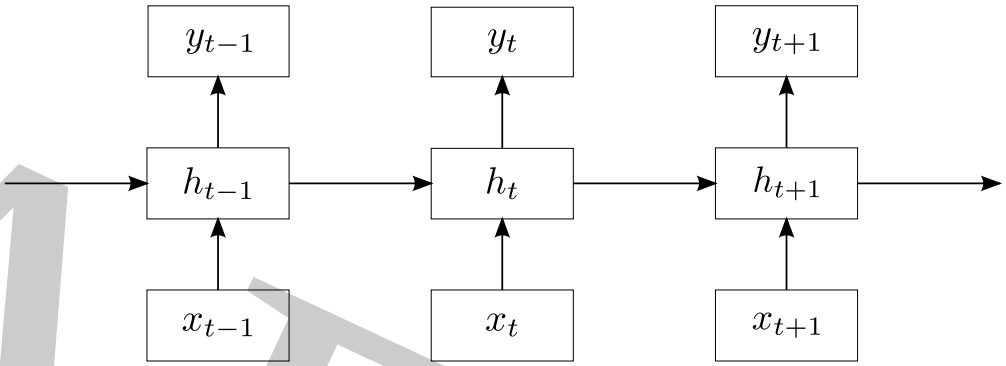


equivalent interpretation: unfolding RNN over time:
feedforward network with a special DEEP architecture.



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

- **output vector:** $y_t(c) = p_\vartheta(c_t|x_1^t) = S(Vh_t)$
  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



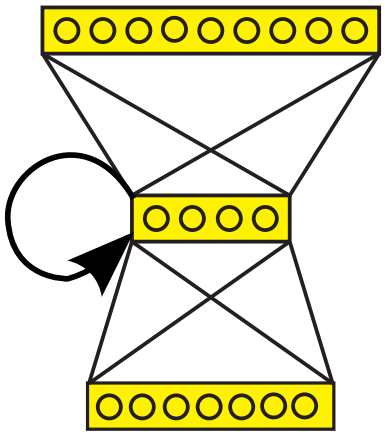
interpretation: conditional probability of RNN:

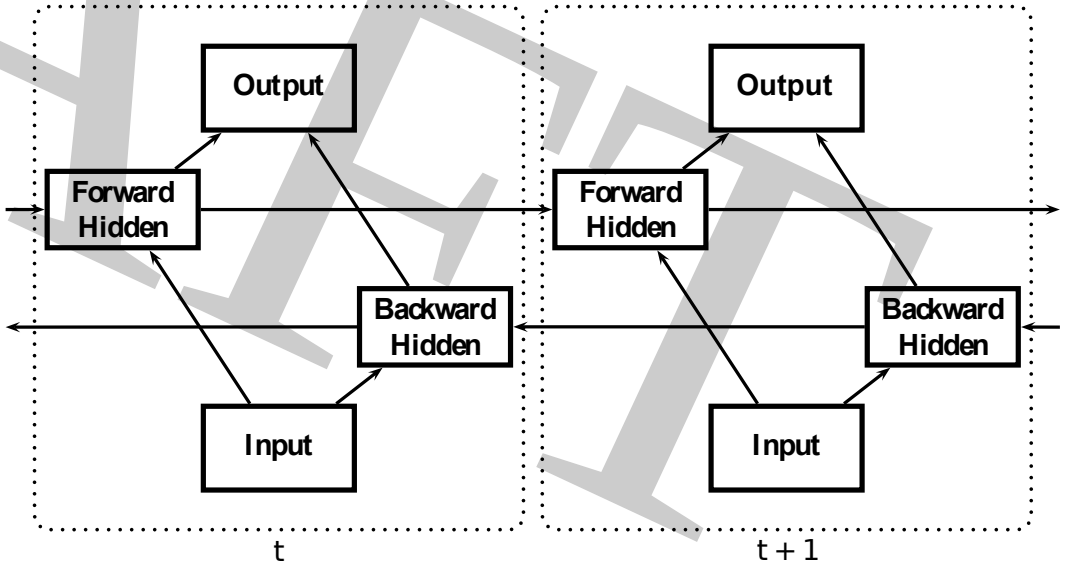
$$\begin{aligned} h_t &= h_t(h_{t-1}, x_t) = ... = h_t(x_1^t) \\ p_t(c|h_t) &= p_t(c|x_1^t) \end{aligned}$$

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

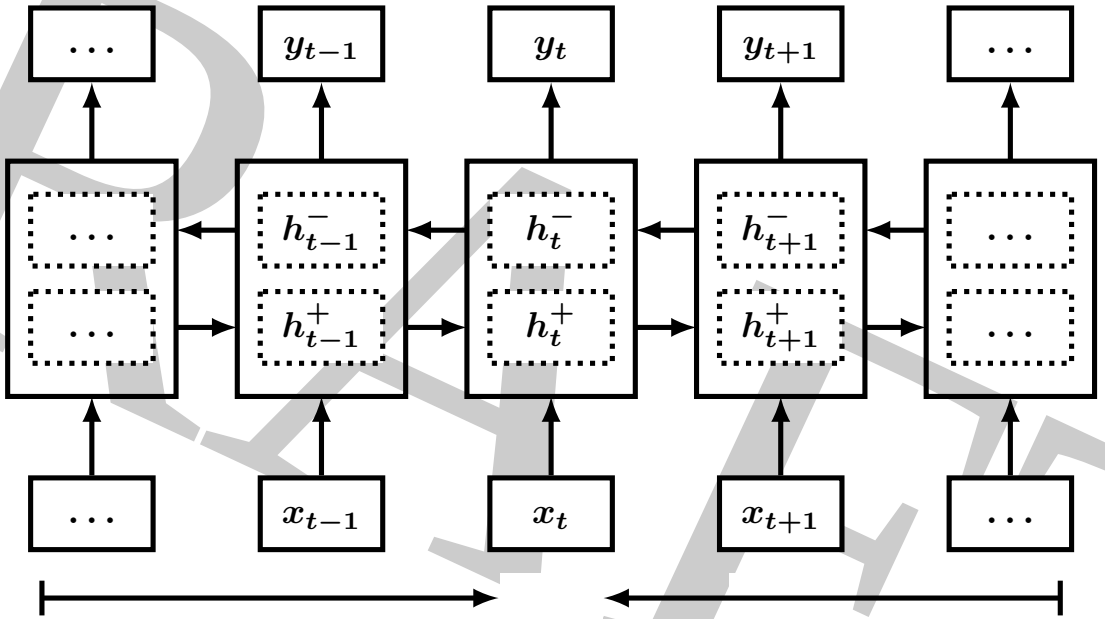
Look-Ahead: Bidirectional RNN
[Schuster & Paliwal 97]

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



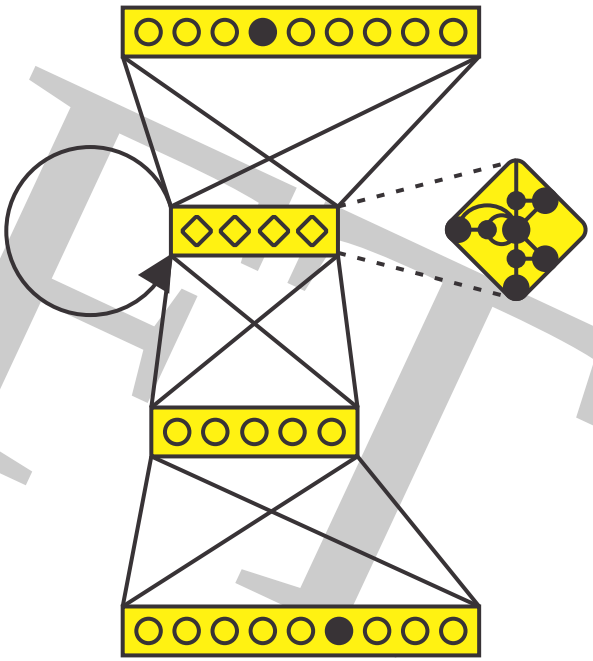


Internal Structure: Separate Forward and Backward Hidden Layers



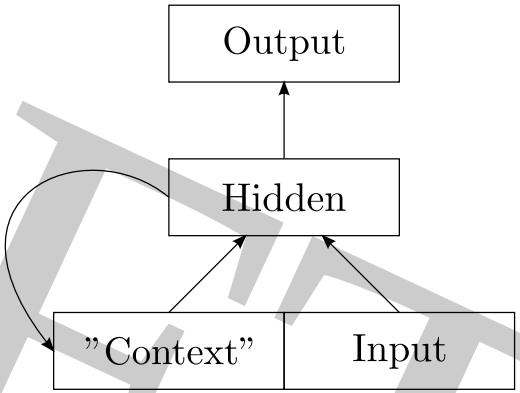
extension of (simple) RNN by
LSTM: long short-term memory
[Hochreiter & Schmidhuber 97, Gers & Schraudolph$^+$ 02]

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



Recurrent Neural Network (RNN): Probabilistic Interpretation

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



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

input: sequence of observations: $x_1^T = x_1 ... x_t ... x_T$

output: sequence of class labels: $c_1^T = c_1 ... c_t ... c_T$

consider the posterior probabilty of the output string:

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

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

and more variants ...

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

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

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

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

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

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

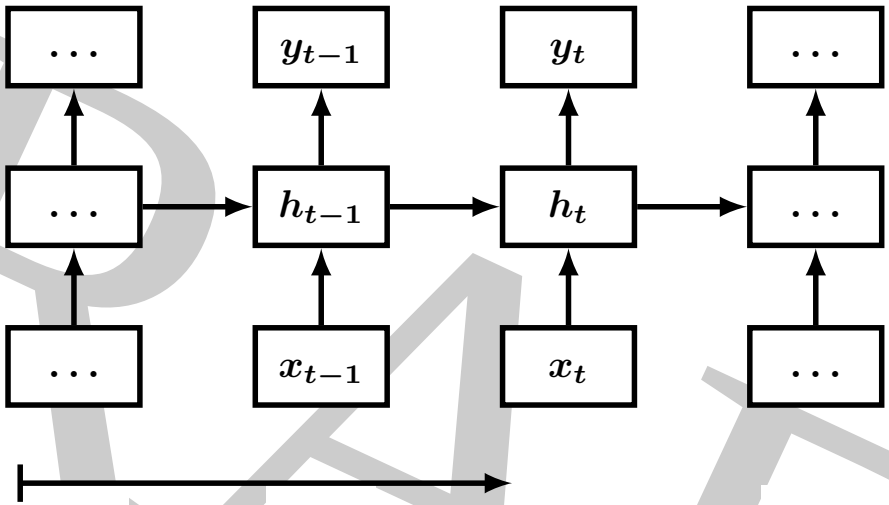
observations x_1^T:	x_1	x_2	...	x_{t-1}	x_t	$-$	...	$-$	$-$
	|	|	|	|	|	|	|	|	|
class labels c_1^T:	c_1	c_2	...	c_{t-1}	$\boxed{c_t}$	$-$	...	$-$	$-$

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

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

note: this is the most general case.

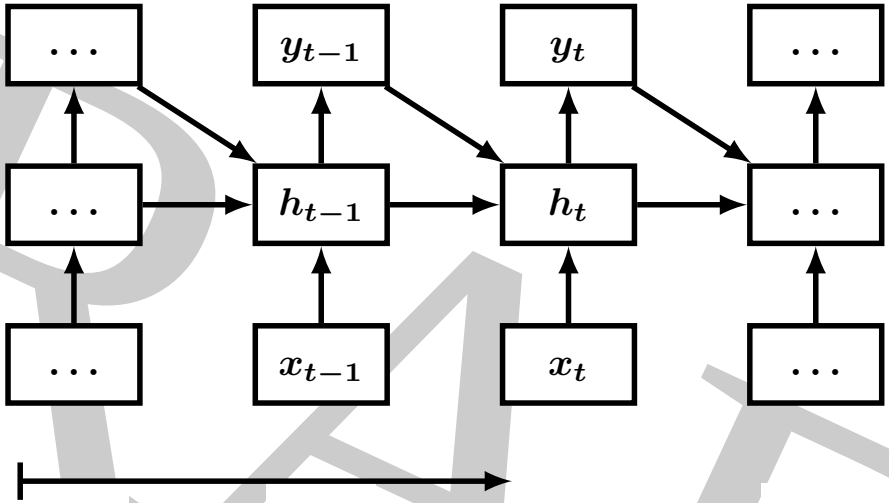
RNN: Variant 1
uni-directional, no feedback of output labels



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

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

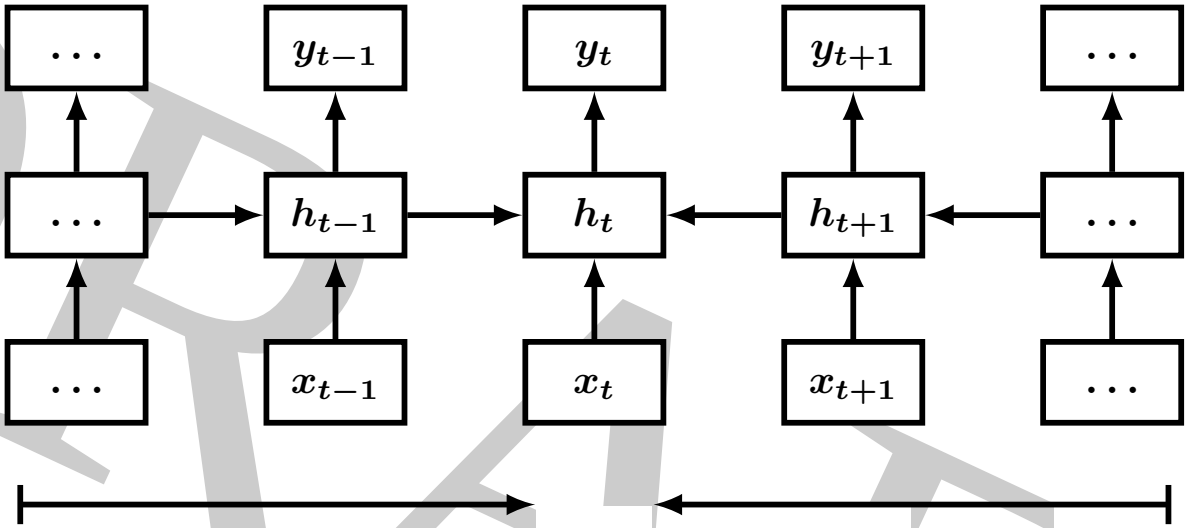
RNN: Variant 2
uni-directional, with feedback of output labels



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

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

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

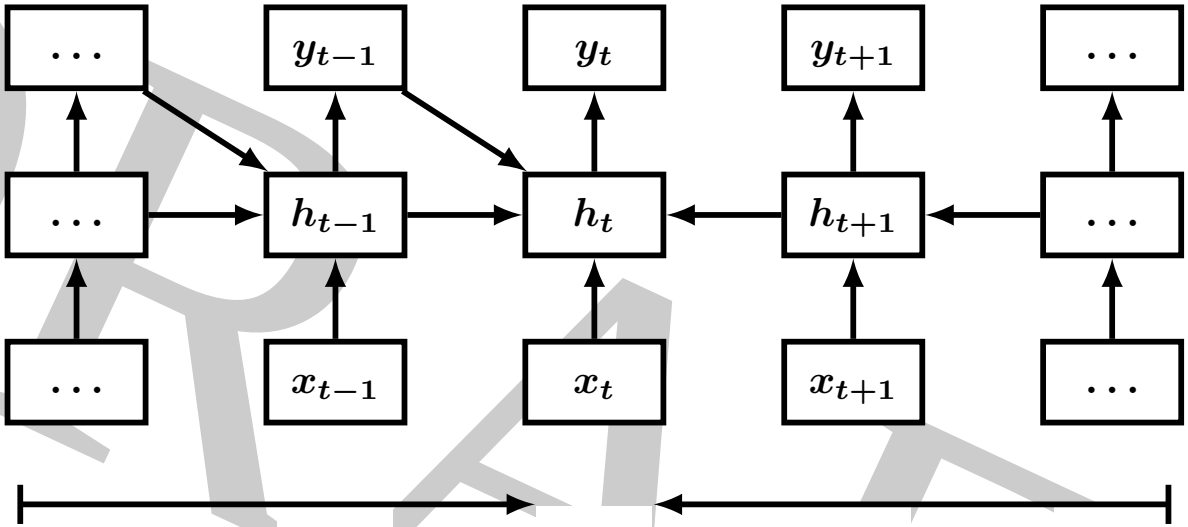


RNN output: marginal probability in position t with full (non-causal) input x_1^T

$$h_t = \ldots = h_t(x_1^T)$$
$$y_t(c) = p_t(c|h_t) = p_t(c|x_1^T)$$

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

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

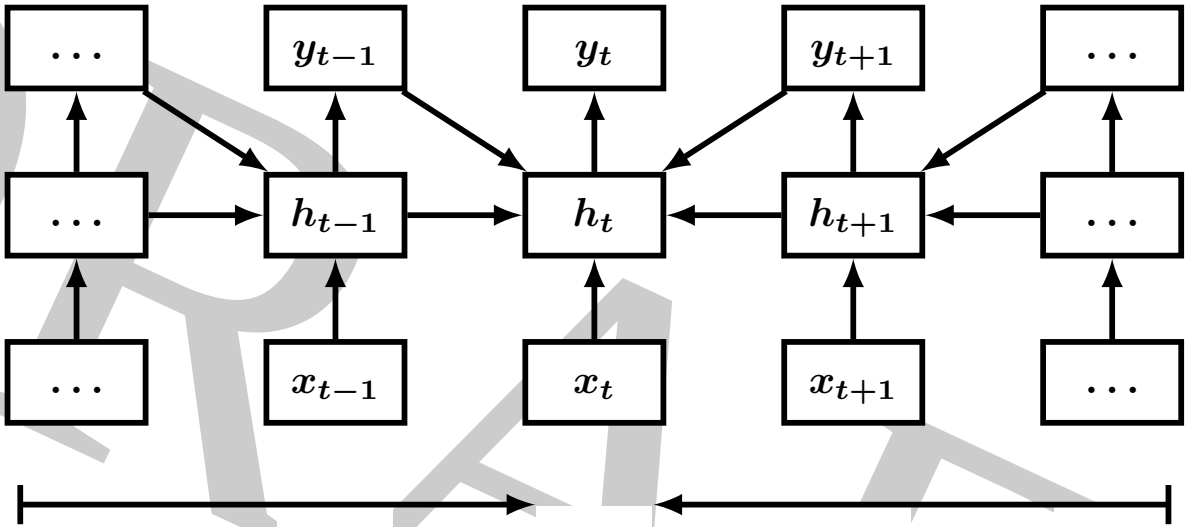


RNN output: conditional probability in position t with non-causal input x_1^T

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

note: most general case for factorization!

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



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

$$h_t = ... = h_t(c|c_1^T \setminus c_t, x_1^T)$$
$$y_t(c) = p_t(c|h_t) = p_t(c|c_1^T \setminus c_t, x_1^T)$$

note: this output cannot be used for factorization!

Overview of RNN Outputs

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

remarks:
– extension: several hidden layers (e.g. 4, 5 or more)
– most important variants (e.g. for RNN transducer and RNN attention):
  o variant 3 for input sequence
  o variant 2 for output sequence

certain types of RNNs are widely used:

- bidirect. RNN without label feedback:

$$p_t(c|h_t(x_1^T))$$

  encoder $h_t(x_1^T)$ of the input (speech) sequence:
  – hybrid HMM and relate models
  – RNN transducer
  – RNN attention

- bidirect. RNN with left-to-right label feedback:

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

  decoder on the output (label) sequence
  (with some additional input from the input sequence)
  – RNN transducer
  – RNN attention
  underlying concept: model the LM dependencies in the output

how to train the parameters ϑ of an RNN?

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

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

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

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

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

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

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:
  - general discriminant functions: [Patterson & Womack 66]
  - ANNs for hybrid HMMs in ASR: [Bourlard & Wellekens 89, ?]

- experimental success and deep learning:
  - LeNet for digit image recognition: Le Cun 1989
  - RNN for ASR: Robinson 1994
  - systematic experimental success:
    only in 2009-2011 (handwriting, speech, image)
  - real improvements over competing methods
    (like Gaussian, generative HMM, SVM, log-linear models, ...):

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

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

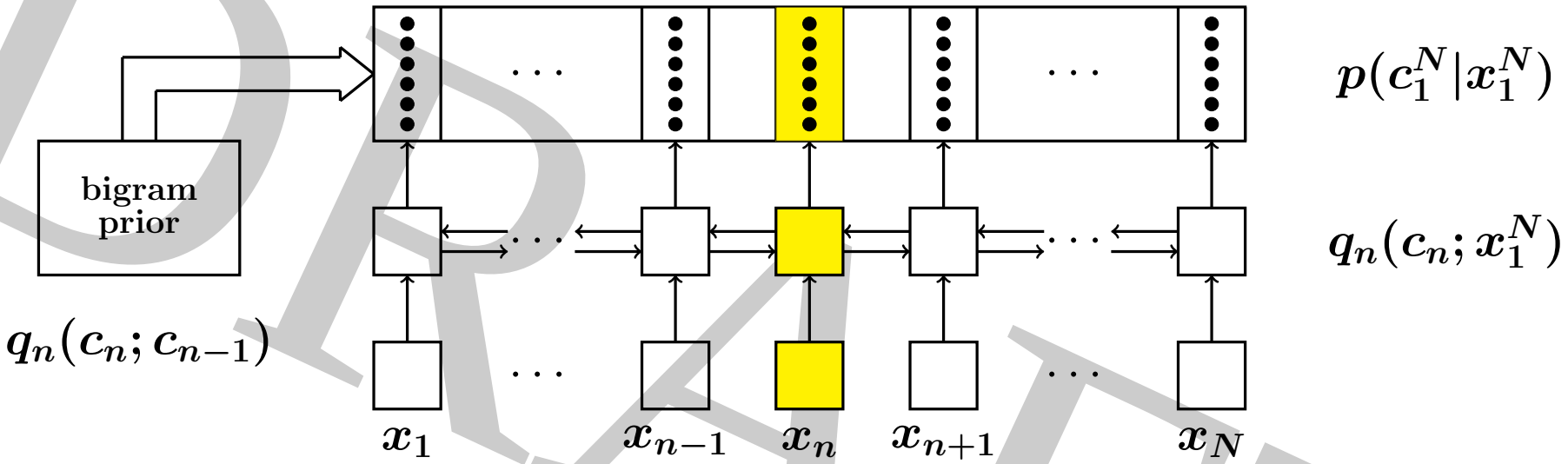
2.4.2 Synchronized Strings and Tagging

synchronized strings: typical task
– POS tagging: POS = parts of speech = word categories

given synchronization:

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

POS Tagging: RNN based CRFs



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

remark:
this approach does NOT start with conditional probabilities!

Example of Synchronized Strings: RNN-CRF

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:

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

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

compare with re-normalized generative model with NORMALIZED components:

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

popular system: BERT [Devlin et al., Google 2018]

BERT= Bidirectional Encoder Representations from Transformers

2.4.3 Cross-Attention and Transformer

sequence pair $[x_1^T, c_1^N]$ with no synchronization:

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

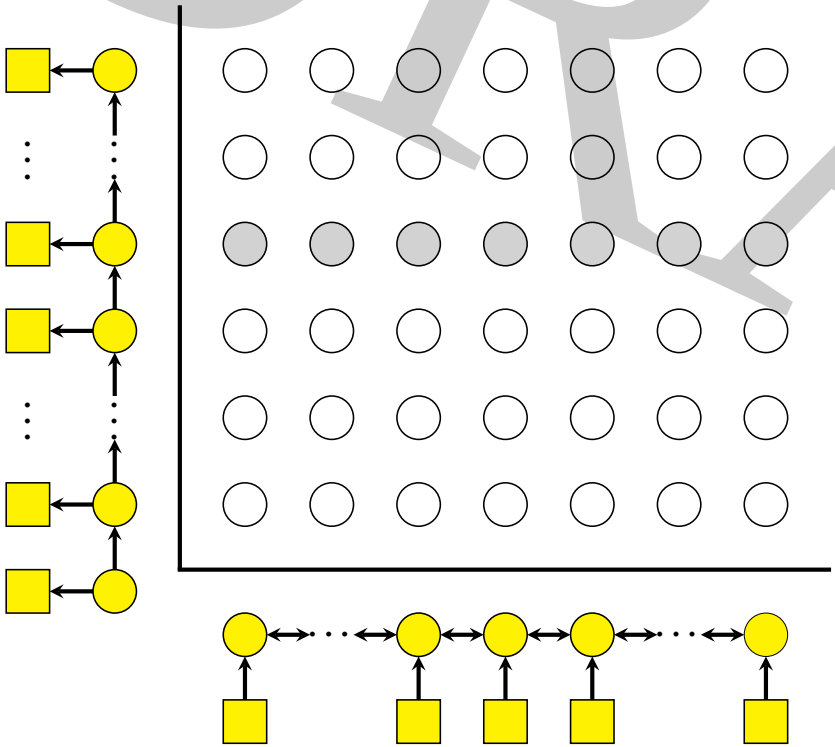
synchronisation: two-dimensional problem:

– find associations/alignments between input and output sequence

– preprocessing (representations) for both input and output:

o input: 'encoder': bi-directional sequence model

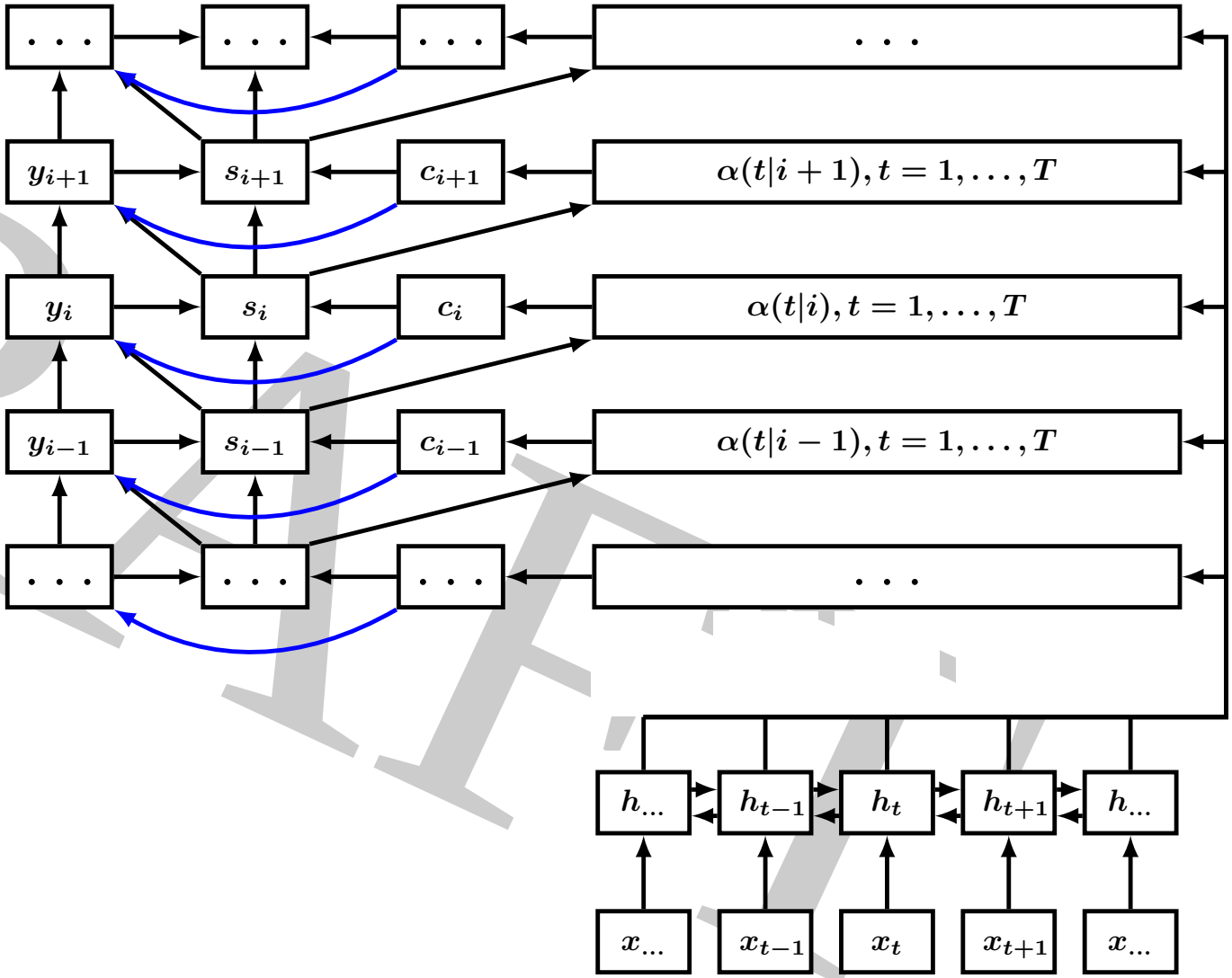
o output: 'decoder': uni-directional sequence model



today's most succesful approaches:

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

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



Transformer Approach: Architecture

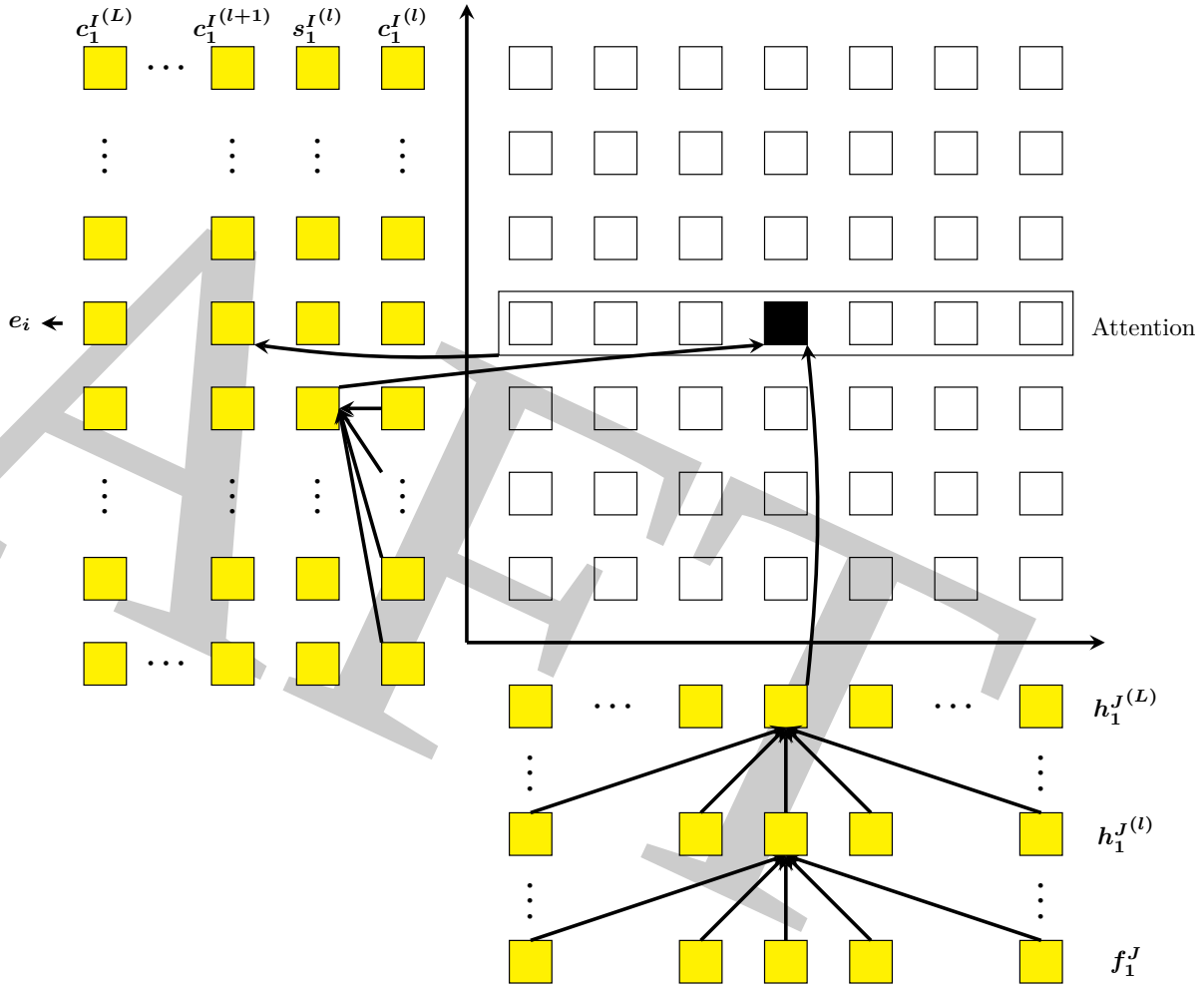
key quantities:

- **self-attention on source side:**
  - **word representation h_j with**
  - **self-attention weights $\alpha(j|j')$**
- **target side:**
  - **state vectors s_i with**
  - **self-attention weights $\alpha(i|i')$**
  - **context vectors c_i with**
  - **cross-attention weights $\alpha(j|i)$**

in addition:

– everywhere: several layers $l = 1...L$

– multi-head for cross-attention
$n = 1...N$



missing synchronization:

input vectors:	x_1	x_2	...	...	x_{t-1}	x_t	x_{t+1}	...	...	x_{T-1}	x_T
		?			?		?		?		
output symbols:	a_1	a_2	...		a_{i-1}	a_i	a_{i+1}	...	a_{I-1}	a_I	

important result: factorization (despite missing synchronization!)

$$p(a_1^I|x_1^T) = \prod_i p(a_i|a_0^{i-1}, x_1^T) = \prod_i p(a_i|a_{i-1}, s_{i-1}, x_1^T)$$

with some state vector s_i

2.4.4 Finite-State Transducer: HMM, CTC, RNN-T

terminology:
– FST: finite-state transducers (or machines/models)
– HMM: hidden Markov models
in both cases: (tacit) assumption of first-order dependencies
related terminology in computer science: probabilistic regular grammar

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

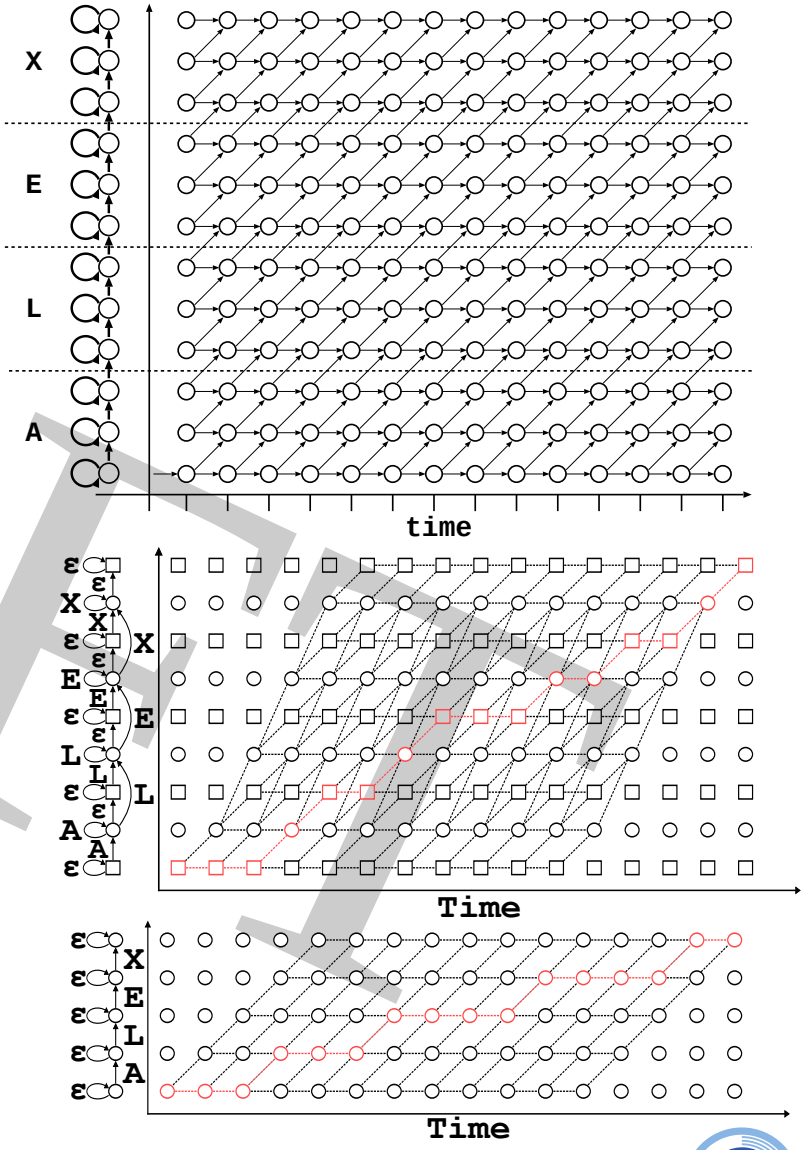
Finite State Machines:
Hybrid HMM, Direct HMM, (RNN-) Transducer

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

main differences in structures:

- **generative model:**
  classical model of 1975–2000
- **discriminative model:**
  started around 1990
  – hybrid HMM
  – CTC (with blank symbol ϵ
  and specific transition constraints)
  – RNN transducer: similar to CTC
  with 'integrated LM' for output labels
  – direct or posterior HMM (RWTH team)

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



Basic Units for ASR and MT

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

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

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

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

alignment path $t \to n = s_t$:
mapping from time t (in x_1^T) to symbol positions n (in c_1^N)

model: sum over hidden alignments s_1^T:

$$p(c_1^N|x_1^T) = \sum_{s_1^T} p(s_1^T, c_1^N|x_1^T)$$

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

with labels y_t at each time position t

note: factorization into cond.prob. $p(c_n|c_{n-1}, x_1^T)$ not easy
(in contrast to cross-attention models!)

2.5 Sequence Processing: Conclusions

interpretation of sequences:
– structured output unlike single events
– the system output can be decomposed into symbols with positions

two types of sequence processing tasks:
– with synchronization
– no synchronization

Types of Posterior Probability Models

various types of events for which we compute a posterior probablity:

- **single event pair $[x, c]$**

$$p(c|x)$$

- **sequence pair $[x_1^N, c_1^N]$:**
  symbol at position n in sequence context:

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

- **full sequence of symbols:**

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

- full sequence of symbols:

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

- sequence $[x_1^N, c_1^N]$: symbol at position n WITHOUT context:

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

types of implementation:
– used suitable RNN structure producing this output directly
– carry out sum explicitly (maybe hard!)

- sequence model with hidden alignments s_1^T (FST, HMM):

$$p(c_1^N|x_1^T) = \sum_{s_1^T} p(c_1^N, s_1^T|x_1^T) = \sum_{s_1^T} \prod_t p(s_t, y_t = c_{s_t}|s_{t-1}, x_1^T)$$

remarks:
– all these models result in cross-entropy training
– the concept of cross-entrpoy training is NOT limited to the output of an ANN

Visualization: Strings with/without Synchronization (Input-Output Relation)

given synchronization:

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

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

missing synchronization:

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

$$p(c_1^N|x_1^T) = \prod_n p(c_n|c_0^{n-1}, x_1^T) \qquad p_n(c|x_1^T) \;?\cong? \sum_{c_1^N: c_n = c} p(c_1^N|\hat{c}_1^N, x_1^T)$$

using a seed string $\hat{c}_1^N = \hat{c}_1^N(x_1^T)$ (e. g. transcribed reference string)

Visualization: Strings with/without Synchronization
Error Counting

synchronization between input/output carries over to
synchronization between recognized string and correct string

given synchronization:

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

loss function: Hamming distance

missing synchronization between c_1^N and $\tilde{c}_1^M$:

correct string:	$\tilde{c}_1$	$\tilde{c}_2$	...	...	$\tilde{c}_{m-1}$	$\tilde{c}_m$	$\tilde{c}_{m+1}$	...	...	$\tilde{c}_{M-1}$	$\tilde{c}_M$
			?		?		?		?		
hypothesized string:		c_1	c_2	...	c_{n-1}	c_n	c_{n+1}	...	c_{N-1}	c_N	

loss function: WER (= edit distance) in ASR and TER in MT

3.1 Principle: From Single Symbols to Strings

preview:
– how to apply cross-entropy training to sequences ?
– specific model: $p\left(c_n \middle| c_{n-k}^{n-1}, x_{n-m}^{n+m}\right)$
– what exactly does it learn?

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

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

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

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

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

- illustration:

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

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

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

typical problems:

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

Sequences: Notation

- **single events: basic joint event**

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

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

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

- **sequences: basic joint event = sequence pair:**

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

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

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

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

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

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

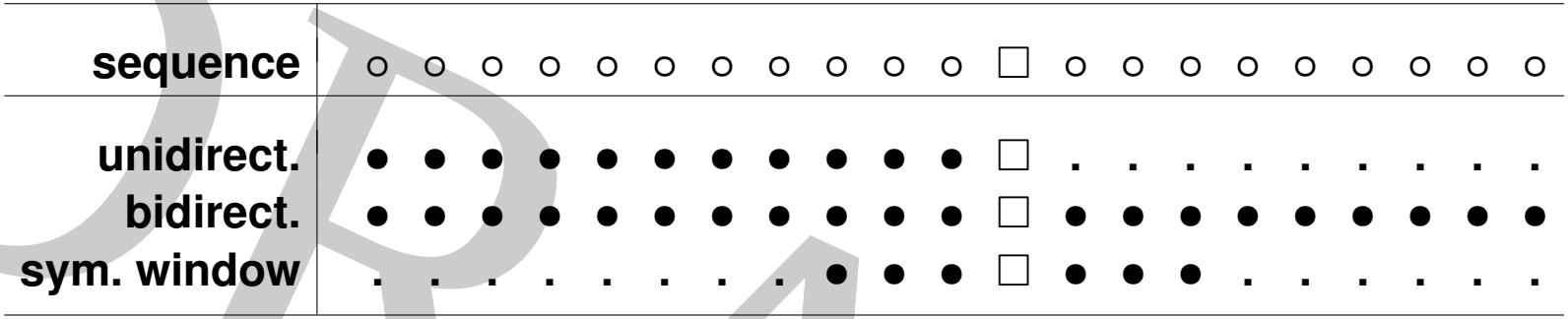
$$p(c_1^N|x_1^N) = \prod_{n=1}^{N} p(c_n|c_0^{n-1}, x_1^N)$$

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

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

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

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



associated distributions (maybe difficult to compute!):

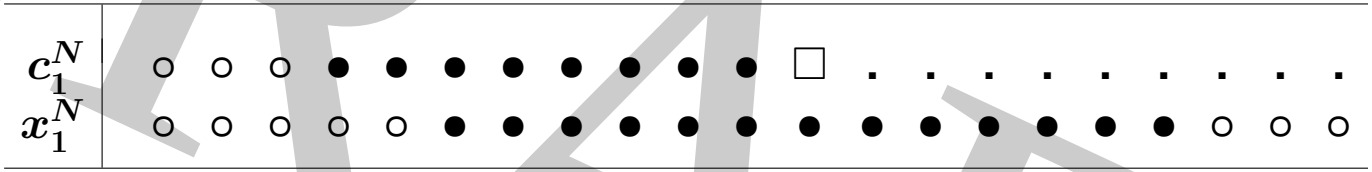
- **full sequence:** $p(c_1^N|x_1^N) = \prod_n p(c_n|c_0^{n-1}, x_1^N)$
- **unidirectional or full left context:** $p(c_n|c_0^{n-1}, x_1^N)$
- **bidirectional or full context:** $p(c_n|c_1^{n-1}, c_{n+1}^N, x_1^N)$
- **symmetric window:** $p(c_n|c_{n-k}^{n-1}, c_{n+1}^{n+k}, x_1^N)$
  special case $k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

input sequence x_1^N is handled in a similar way

general model to be studied:

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

illustration: context windows around a position n:



remarks: context windows

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

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

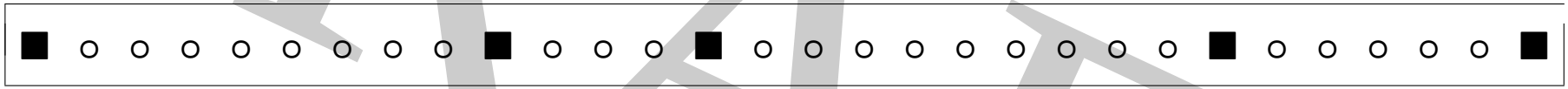
– interpretation: decomposition of input/output sequence
into overlapping blocks $[c_{n-k}^n,\ x_{n-m}^{n+m}]$

3.2 Cross-Entropy at Sequence Level

model with limited contexts:

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

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



cross-entropy criterion:

$$\begin{aligned} F(\{p\}) &= \sum_s \log p\big([c_s]_1^N \big| [x_s]_1^N\big) = \sum_s \log \prod_n p\big([c_{sn}\big|[c_s]_{n-k}^{n-1}, [x_s]_1^N\big) \\ &= \sum_s \sum_n \log p(c_{sn}|[c_s]_0^{n-1}, [x_s]_1^N) \\ &\quad \textbf{(assume model with limited contexts)} \\ &= \sum_s \sum_n \log\ p\big(c_{sn}\big|[c_s]_{n-k}^{n-1}, [x_s]_{n-m}^{n+m}\big) \end{aligned}$$

model with limited context (for input and output):

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

training criterion: cross-entropy:

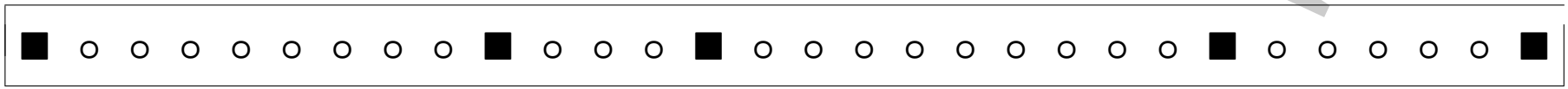
- **set of sequence pairs $s = 1, .., S$:** $[x_s, c_s]_{n=1}^{N_s}$

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

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

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

note the formal similarity!



fairly general model $p(c_n|c_{n-k}^{n-1}, x_{n-m}^{n+m})$**:**

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

we consider a simplified model $p(c_n|c_{n-1}, x_n)$**:**

$$F(p) = 1/N \cdot \sum_{n=1}^{N} \log p(c_n|c_{n-1}, x_n)$$

joint events: $[c', c, x] \equiv [c_{n-1}, c_n, x_n]$

with empirical distribution: $pr(c', c, x)$

$$= \sum_{c',c,x} pr(c', c, x) \log p(c|c', x)$$

$$= \sum_{c',x} pr(c', \cdot, x) \sum_{c} pr(c|c', x) \log p(c|c', x)$$

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

remarks:

– optimization can be carried out for each pair (c', x)

– result: similar to the single symbol case

we consider the general model $p\left(c_n \middle| c_{n-k}^{n-1},\, x_{n-m}^{n+m}\right)$:

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

solution:

$$\hat{p}\left(c_n \middle| c_{n-k}^{n-1},\, x_{n-m}^{n+m}\right) = pr\left(c_n \middle| c_{n-k}^{n-1},\, x_{n-m}^{n+m}\right)$$

note: n is only a formal position index

warning: ideal assumptions:

history c_{n-k}^{n-1}: implemented with word embeddings

3.3 From Generative to Discriminative Models

approach:
– history: generative models
– log-linear models and pseudo-posterior
– global re-normalization
– local re-normalization

concept: we introduce a pseudo posterior distribution,
i.e. the posterior without a prior:
– use a uniform prior
– works for empirical and model distribution

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

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

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

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

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

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

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

Posterior and Pseudo Posterior: Towards Log-Linear Modelling

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

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

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

we re-write the posterior:

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

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

Posterior and Pseudo Posterior: Distributions for Limiting Cases

limit $\alpha \to 0$:

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

limit $\alpha \to \infty$:

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

Pseudo Posterior: Sequences

joint distribution specifically for sequences:

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

we re-write the posterior:

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

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

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

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

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

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

Pseudo Posterior: Direct Model

direct factoriztion of posterior probability:

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

re-interpretation:

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

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

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

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

and all but one product terms will cancel !

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

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

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

3.4 Separation of Prior

prior for sequences: called LM

approaches:
– double softmax
– inverse prior

Remove the Class Prior: Inverse Prior Weighting

- **standard cross-entropy training:**
  the model always includer the class prior of the training data
- **countermeasures:**
  - **inverse prior weighting**
  - **model with separated prior**

Zero-Order Model: Inverse Prior Weighting

RWTH AACHEN UNIVERSITY

consider sequence modelling:

$$p(c_1^N|x_1^N) = \prod_n p(c_n|x_n)$$

zero-order model: no dependence on predecessor symbols c_0^{n-1}

$$p(c_n|x_n) \equiv p(c|x)$$

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

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

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

principle:
– take care of class prior $pr(c_n)$ in training data
– method: assign weights ($= 1/pr(c_n)$) to each sample position n

HLT

re-consider standard training:

$$\begin{aligned} F\Big(\{p_\theta(c|x)\}\Big) &= 1/N \cdot \sum_n \log\, p_\theta(c_n|x_n) \\ &= \sum_{x,c} pr(c,x) \log\, p_\theta(c|x) \end{aligned}$$

identity: $pr(c,x) = pr(\cdot,x) \cdot pr(c|x)$

$$\begin{aligned} &= \sum_x pr(\cdot,x) \cdot \sum_c pr(c|x) \log\, p_\theta(c|x) \\ \hat{p}_{\hat{\theta}}(c|x) &= pr(c|x) \equiv \frac{pr(c) \cdot pr(x|c)}{\sum_{\tilde{c}} pr(\tilde{c}) \cdot pr(x|\tilde{c})} \end{aligned}$$

inverse prior weighting:

$$\begin{aligned} F\Big(\{p_\theta(c|x)\}\Big) &= 1/N \cdot \sum_n \frac{1}{pr(c_n)} \cdot \log\, p_\theta(c_n|x_n) \\ &= \sum_{x,c} \frac{pr(x,c)}{pr(c)} \cdot \log\, p_\theta(c|x) \\ &= \sum_{x,c} pr(x|c) \cdot \log\, p_\theta(c|x) \\ &= \sum_x \tilde{pr}(x) \sum_c \tilde{pr}(c|x) \cdot \log\, p_\theta(c|x) \end{aligned}$$

solution: .... (and switch from zero-order to first-order model)

first-order model: ANN model with limited context:

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

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

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

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

principle:
– take care of class prior $pr(c_n, c_{n-1})$ in training data
– method: assign weights ($= 1/pr(c_n, c_{n-1})$) to each sample position n

re-consider standard training:

$$
\begin{aligned}
F\Big(\{p_\theta(c|x,c')\}\Big) &= 1/N \cdot \sum_n \log\, p_\theta(c_n|x_n, c_{n-1}) \\
&= \sum_{x,c,c'} pr(c', c, x) \log\, p_\theta(c|x, c') \\
&\quad \textbf{identity:} \quad pr(c', c, x) = pr(c', \cdot, x) \cdot pr(c|x, c') \\
&= \sum_{x,c'} pr(c', \cdot, x) \cdot \sum_c pr(c|x, c') \log\, p_\theta(c|x, c') \\
\hat{p}_{\hat{\theta}}(c|x, c') &= pr(c|x, c') \equiv \frac{pr(c|c') \cdot pr(x|c, c')}{\sum_{\tilde{c}} pr(\tilde{c}|c') \cdot pr(x|\tilde{c}, c')}
\end{aligned}
$$

inverse prior weighting:

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

inverse prior weighting:

$$F\Big(\{p_\theta(c|x,c')\}\Big) = 1/N \cdot \sum_n \frac{1}{pr(c_n, c_{n-1})} \cdot \log\, p_\theta(c_n|x_n, c_{n-1})$$

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

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

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

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

result: solution for fully saturated model = pseudo posterior

First-Order Model: Inverse Prior Weighting
Variant: Conditional Probability

inverse prior weighting:

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

Remove the Class Prior: Separated Prior

introduce specific structure into observation model:

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

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

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

$$\frac{pr(c|c') \cdot q_\theta(c|x,c')}{\sum_{c''} pr(c''|c') \cdot q_\theta(c''|x,c')} \stackrel{!}{=} \frac{pr(c|c') \cdot pr(x|c,c')}{\sum_{c''} pr(c''|c') \cdot pr(x|c'',c')} = \frac{pr(c|c') \cdot \frac{pr(x|c,c')}{Q(x,c')}}{\sum_{c''} pr(c''|c') \cdot \frac{pr(x|c'',c')}{Q(x,c')}}$$

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

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

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

Separated Prior: Implementation

two approaches to separating the prior:

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

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

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

- numerical approach: define ANN structure with double softmax:

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

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

3.5 Model with Explicit LM

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

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

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

- **training criterion for string pairs** $[X_r, W_r],\ r = 1, ..., R$**:**

$$\max_{\vartheta} \left\{ \sum_r \log\ p_\vartheta(W_r|X_r) \right\}$$

- **composed model of language model and acoustic model:**

$$p_\vartheta(W|X) = \frac{q_\vartheta^\alpha(W) \cdot q_\vartheta^\beta(W|X)}{\sum_{\tilde{W}} q_\vartheta^\alpha(\tilde{W}) \cdot q_\vartheta^\beta(\tilde{W}|X)}$$

- **consider cross-entropy training criterion:**

$$\begin{aligned}
\hat{\vartheta} &= \arg\max_{\vartheta} \left\{ \sum_r \log\ p_\vartheta(W_r|X_r) \right\} \\
&= \arg\max_{\vartheta} \left\{ \alpha \cdot \sum_r \log\ q_\vartheta(W_r) + \beta \cdot \sum_r \log\ q_\vartheta(W_r|X_r) \right. \\
&\qquad \left. - \sum_r \log \sum_{\tilde{W}} q_\vartheta^\alpha(\tilde{W}) \cdot q^\beta(\tilde{W}|X_r) \right\}
\end{aligned}$$

note: weak dependence on ϑ due to sum over $\tilde{W}$

$$\cong \arg\max_{\vartheta} \left\{ \alpha \cdot \sum_r \log\ q_\vartheta(W_r) + \beta \cdot \sum_r \log\ q_\vartheta(W_r|X_r) \right\}$$

- **resulting approximation:**

$$\hat{\vartheta} = \arg\max_{\vartheta} \left\{ \sum_r \log\, p_\vartheta(W_r|X_r) \right\}$$

$$\cong \arg\max_{\vartheta} \left\{ \alpha \cdot \sum_r \log\, q_\vartheta(W_r) \right\} + \beta \cdot \sum_r \log\, q_\vartheta(W_r|X_r) \right\}$$

- **independent parameters for language model and aoustic model:**

$$\vartheta := \{\lambda, \mu\}$$

$$\hat{\lambda} \cong \arg\max_{\lambda} \left\{ \alpha \cdot \sum_r \log\, q_\lambda(W_r) \right\} = \arg\max_{\lambda} \left\{ \sum_r \log\, q_\lambda(W_r) \right\}$$

$$\hat{\mu} \cong \arg\max_{\mu} \left\{ \beta \cdot \sum_r \log\, q_\mu(W_r|X_r) \right\} = \arg\max_{\mu} \left\{ \sum_r \log\, q_\mu(W_r|X_r) \right\}$$

result of approximation: decoupled training:
– wellknown method for acoustic model
– wellknown method for language model

Language Model Training: Perplexity

- **result: training criterion for language model with strings $W_r = [w_1 ... w_i ... w_{I_r}] = w_1^{I_r}$:**

$$\hat{\lambda} = \arg\max_{\lambda} \left\{ \sum_r \log\, q_\lambda(w_1^{I_r}) \right\} = \arg\max_{\lambda} \left\{ \sum_r \sum_{i=1}^{I_r} \log\, q_\lambda(w_i | w_0^{i-1}) \right\}$$

remark: independence of audio data!
pure text data can be used!

- **interpretation: superstring = concatenation of all strings $W_r, r = 1..., R$:**

$$w_1^N := W_1^R = [w_1 ... w_i ... w_{I_r}]_{r=1}^{r=R}$$

note: special handling (extra symbols) for string boundaries

- **normalized criterion = inverse geometric average: perplexity PP**

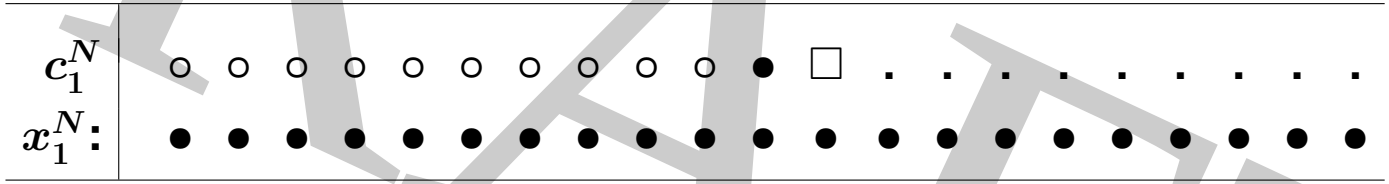
$$\log PP := \log\, 1 \Big/ \sqrt[N]{q_\vartheta(w_1^N)} = -1/N \cdot \sum_{n=1}^{N} \log\, q_\vartheta(w_n | w_0^{n-1})$$

terminology:
– maximum likelihood (ASR community)
– cross-entropy (ANN community)

3.6 Revisit Separation of Prior LM

problem: acoustic model will be masked by LM!

consider the model $q(c_n|c_{n-1}, x_1^N)$:



problem formulation:
– the model assumes bigram dependence for c_1^N
– the true dependence might require a larger context
– result (to be shown): separation of the LM in the model is hard

Baseline: Implicit Learning of Language Model
Specific Case: Full Sentence Context

we assum an LM $pr(c_1^N)$ and an observation model $pr(x_1^N|c_1^N) = \prod_n pr(x_n|c_n)$

we rewrite (include c_0 as sentence start symbol):

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

$$pr(c_{n-1}^n, x_1^N) = \sum_{c_1^N \setminus c_{n-1}^n} pr(c_1^N, x_1^N) = \sum_{c_1^N \setminus c_{n-1}^n} pr(c_1^N) \cdot pr(x_1^N|c_1^N)$$

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

important conclusions:
– the bigram dependence in the emprirical posterior distribution
  is the result of a long-range LM dependence.
– thus the corresponding model $q(c|c', x_1^N)$ learned by cross-entropy training is:

$$\hat{q}(c|c', x_1^N) = pr(c|c', x_1^N)$$

– the exact compensation of the LM prior is hard in practice!

4 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?

4.1 Decision Rules and Loss Function

terminology and notation:

- **input x: observation:**
  - **single event: set of measurements, observation vector**
  - **sequence/string of observations: $x = x_1^T := x_1 ... x_t ... x_T$**
- **output c: class symbol(s):**
  - **single symbol: no internal structure, i.e. atomic**
  - **sequence/string of events: $c = c_1^N := c_1 ... c_n ... c_N$**

  speech recognition: words $w = w_1^N := w_1 ... w_n ... w_N$

- **functional dependence: notational variants:**

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

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

Loss Function and Performance

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: $x := x_1 ... x_t ... x_T,\ x_t \in \mathbb{R}^D$)
  – output sequence: $c := c_1 ... c_n ... c_N$ 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:

$$(x_k, c_k), \; k = 1, ..., K$$

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

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

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:

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

Loss Functions

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

Loss Functions for Symbol Strings

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)

Preview: Loss Functions and Decision Rules

decision rules for general loss $L[c, \tilde{c}]$:

$$\text{model:} \quad x \rightarrow c_\vartheta(x) = \arg\min_{\tilde{c}} \{L_\vartheta[\tilde{c}|x]\} = \arg\min_{\tilde{c}} \{\sum_c p_\vartheta(c|x)\, L[\tilde{c}, c]\}$$

$$\text{Bayes:} \quad x \rightarrow c_*(x) = \arg\min_{\tilde{c}} \{L_*[\tilde{c}|x]\} = \arg\min_{\tilde{c}} \{\sum_c pr(c|x)\, L[\tilde{c}, c]\}$$

expectation of loss using empirical distribution:

$$\text{model:} \quad L_\vartheta = \sum_x pr(x) \sum_c pr(c|x)\, L[c_\vartheta(x), c]$$

$$\text{Bayes:} \quad L_* = \sum_x pr(x) \sum_c pr(c|x)\, L[c_*(x), c] = \sum_x pr(x)\, \min_{\tilde{c}} \{\sum_c pr(c|x)\, L[\tilde{c}, c]\}$$

decision rule for error count = 0/1 loss: $L[c, \tilde{c}] = 1 - \delta(c, \tilde{c})$:

$$\text{model:} \quad x \rightarrow c_\vartheta(x) = \arg\max_{\tilde{c}} \{p_\vartheta(\tilde{c}|x)\}$$

$$\text{Bayes:} \quad x \rightarrow c_*(x) = \arg\max_{\tilde{c}} \{pr(\tilde{c}|x)\}$$

error rate = expectation of 0/1 loss using empirical distribution:

$$\text{model:} \quad E_\vartheta = L_\vartheta = \sum_x pr(x)\, pr(c \neq c_\vartheta(x)|x) = 1 - \sum_x pr(x)\, pr(c = c_\vartheta(x)|x)$$

$$\text{Bayes:} \quad E_* = L_{**} = \sum_x pr(x)\, pr(c \neq c_*(x)|x) = 1 - \sum_x pr(x)\, \max_c \{pr(c|x)\}$$

4.2 Empirical Distribution

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

Decision Rule and Empirical Distribution: Illustration

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:

$$(x_k, c_k), \; k = 1, ..., K$$

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.

Illustration: Expectation and Empirical Distribution

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 $pr(c) > 0$:

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

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

$$L[d] = \frac{1}{K} \cdot \sum_{k=1}^{K} L[c_k, c_d(x_k)] = \sum_{x,c} pr(x,c)\, L[c, c_d(x)] = \sum_{x} pr(x) \underbrace{\sum_{c} pr(c|x)\, L[c, c_d(x)]}_{=: L[d|x]}$$

- for each x, we minimize over all decision rules d:

$$\min_d L[d|x] = \min_d \Big\{ \sum_c pr(c|x)\, L[c, c_d(x)] \Big\}$$
$$= \min_{\tilde{c}_x} \Big\{ \sum_c pr(c|x)\, L[c, \tilde{c}_x] \Big\} = \min_{\tilde{c}} \Big\{ \sum_c pr(c|x)\, L[c, \tilde{c}] \Big\}$$

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

- we have arrived at the Bayes decision rule:

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

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 $p(c|x)$**

Error Count for Atomic Outputs: 0/1 Loss Function

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:

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

remarks:

– result: MAP decision rule (MAP = maximum a-posteriori)

– threshold effect: $pr(c|x) \overset{?}{\geq} 0.5$

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:

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

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 distrribution 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**

4.4 Conclusions

theoretical 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?**

5 Bayes Decision Rule: Loss Function

general form of Bayes decision rule:

$$x \rightarrow c_*(x) \;=\; \arg\min_{c} \Big\{ \sum_{\tilde{c}} \; p(\tilde{c}|x) \; L[\tilde{c}, c] \Big\}$$

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

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

5.1 Atomic Output: Single Symbol

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:

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

often referred to as MAP rule:

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

Atomic Output: Weighted Errors

weighted classification errors:

- **typical task: output = single class symbol**
- **we assign weights to the errors**
- **assumption: weights α_c depend on hypothesized class c:**

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

Bayes decision rule:

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

- **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:

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

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

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

approaches:

- **local normalization:**

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

most succesful approach: RNN with bidirectional input

- **global normalization (linear chain CRF):**

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

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

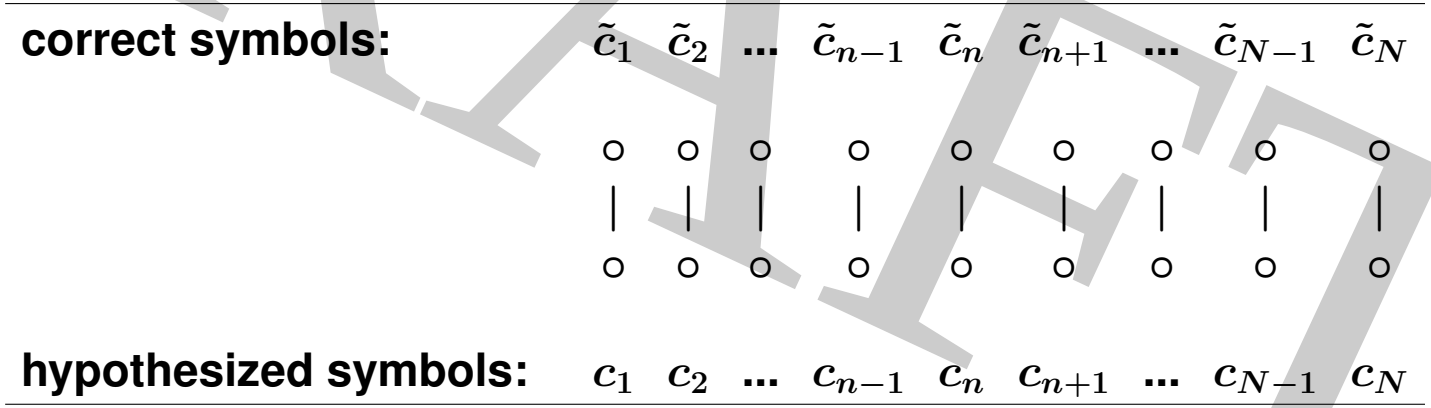
with an ***arbitray*** **non-negative model** $Q_n(c_n; c_0^{n-1}, x_1^N)$

most succesful 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 $n = 1, ..., N$:

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

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



important assumption:
the positions $n = 1, ..., N$ (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 $x = x_1^N$:

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

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 $n = 1, ..., N$, and the same goes for the minimization procedure:

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

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

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

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

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

minimum symbol error:

$$x \rightarrow \hat{c}_n(x) = \arg\max_{c_n}\{p_n(c_n|x)\}$$

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

minimum string error:

$$x \rightarrow \hat{c}_1^N(x) = \arg\max_{c_1^N}\{p(c_1^N|x)\}$$

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

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.

Decision Rules: String vs. Symbol Errors: Experiments

- 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

5.3 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]

5.3.1 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: $\{a, b, d, e, f, ..., u, v, w, ...\}$

– start with abd **and generate three more strings by subsitutions**

– 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:

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

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

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

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

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

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

Pathological Example 2 (with Insertion Errors)

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 minimizatinig 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 $c = c_i, \; i = 1, 2, 3$:**

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

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 $c = \tilde{c} \neq c_i, i = 0, 1, 2, 3$:**

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

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**

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

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

identity/reflexivity: $L[\hat{c},\hat{c}] = 0$
triangle inequality: $L[\hat{c},c] - L[\tilde{c},c] \leq L[\hat{c},\tilde{c}]$

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

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

Important Conclusion: 50% Condition for MAP Rule

For any pair of classes $\hat{c}$ and $\tilde{c}$, we have shown:

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

analysis:

- **assumption: $p(\hat{c}|x) \geq 0.5$:**

  hence $[1 - 2\,p(\hat{c}|x)] \; L[\hat{c}, \tilde{c}] \leq 0$ and

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

- **conclusion: consider the MAP rule:**

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

  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:

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

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

exercise: work out the proofs!

Application to ASR: WER = edit distance

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 $\rightarrow$ 10.6
all cases	100	4.0 $\rightarrow$ 3.9

experimental conditions:

- a trained model $p(c|x)$ is used rather than true distribution
- use of N-best lists: $N = 10\,000$
- use of scaling exponent for language model

experimental result: 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	57	34.2
D: experiment: $c_0(x) \neq c_*(x)$	21	14.3 $\rightarrow$ 13.8	32	43.5 $\rightarrow$ 42.6
all cases	100	9.6 $\rightarrow$ 9.5	100	36.6 $\rightarrow$ 36.3

conclusion: disappointing improvements ...

5.3.3 Rewrite Posterior Loss

- consider a metric loss function: $L[\tilde{c}, c] = 0, 1, 2, ...$
  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 = duration/40 ms

- goal of rewriting:
  to illustrate why the MAP rule nearly always provides
  a good approximation to the correct Bayes decision rule.

Rewrite Posterior Loss

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

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

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

Illustration: Number of Strings with Loss = 1

RWTH AACHEN UNIVERSITY

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: $C \cdot (2\,N+1)$ 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 subsitution:	$\hat{c}_1$	$\hat{c}_2$	...	$\hat{c}_{n-1}$	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	$\hat{c}_{n+1}$	...	$\hat{c}_{N-1}$	$\hat{c}_N$

HLT

Interpretation

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:

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

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 (i. e. substitutions only): $(C-1)\cdot N$ strings
  – edit distance: $C\cdot(2N+1)$ strings

5.3.4 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	$H[\hat{c}^\epsilon, c^\epsilon]$
$\hat{c}^\epsilon$	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 $\hat{\hat{c}}_n^\epsilon$ in positions $n = 1, ..., N_\epsilon$:

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

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

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

exercise: verify/correct/improve example

iterative procedure with local convergence:

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

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

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

- **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: $\hat{\hat{c}}_n^\epsilon := \text{argmax}_{c_n^\epsilon} \left\{ p_n(c_n^\epsilon | \hat{c}, x) \right\}$**

  remove ϵ symbols and use this string as a new seed string

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

$$
\begin{aligned}
L[\hat{c}|x] &= \sum_c p(c|x) \cdot \min_A \{L_A[\hat{c}, c]\} = \sum_c p(c|x) \cdot \sum_{n=1}^{N_\epsilon} [1 - \delta(\hat{c}_n^\epsilon, c_n^\epsilon)] \\
&= \sum_{n=1}^{N_\epsilon} \sum_c p(c|x) \cdot [1 - \delta(\hat{c}_n^\epsilon, c_n^\epsilon)] \\
&= \sum_{n=1}^{N_\epsilon} [1 - \sum_{c: c_n^\epsilon = \hat{c}_n^\epsilon} p(c|x)] \\
&= \sum_{n=1}^{N_\epsilon} [1 - p_n(\hat{c}_n^\epsilon|\hat{c}, x)] \\
&\geq \sum_{n=1}^{N_\epsilon} [1 - \max_{c_n^\epsilon} p_n(c_n^\epsilon|\hat{c}, x)] \\
&\quad \textbf{define:} \quad \hat{\hat{c}}_n^\epsilon := \operatorname*{argmax}_{c_n^\epsilon} \{p_n(c_n^\epsilon|\hat{c}, x)\} \\
&= \textbf{...(equations in backward direction; details to be worked out)} \\
&\geq L[\hat{\hat{c}}|x]
\end{aligned}
$$

Edit Distance and Hamming Distance: Remarks

- how to find an improvement over the seed string?

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

  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.

Symbol Posterior Probability: ASR

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%

5.3.5 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

5.3.6 DRAFT: Redefine Loss: Frame Error vs. Edit Distance

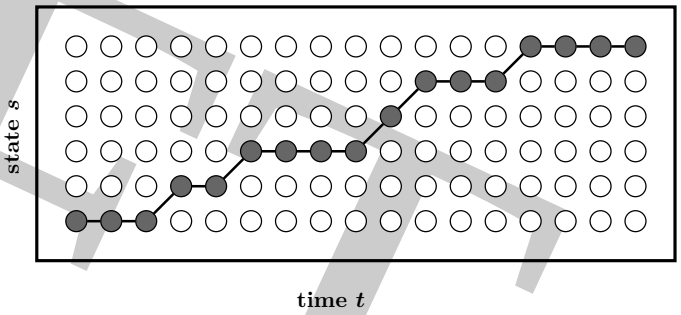
we re-consider ASR and the output string:

- 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):

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

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



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:**

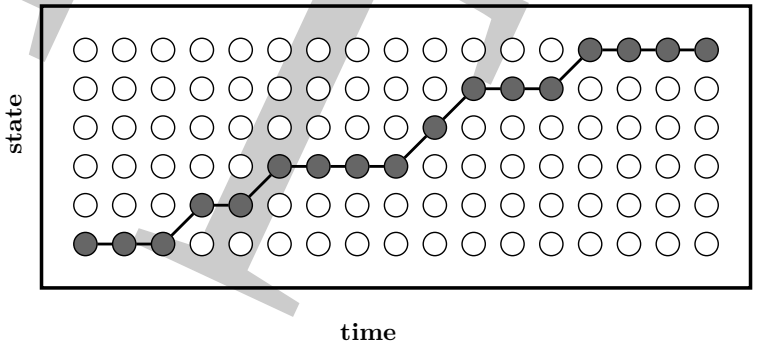
$$L[y_1^T|x_1^T] := \sum_{\tilde{y}_1^T} p(\tilde{y}_1^T|x_1^T) \cdot \sum_t [1 - \delta(y_t, \tilde{y}_t)] = \ldots = \sum_t \left[1 - p_t(y_t|x_1^T)\right]$$

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

- **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 compopnents (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):

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

6 Classification Error and Training Criterion

6.1 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:**
  o the model distribution is different from the true distribution
  o how does this mismactch effect the optimality of the Bayes decision rule?
  o can we train model for optimum performance/minimum classification error?

methodology:
– purely theoretical/mathematical
– bounds on classification error

central role of performance measure or classification error:

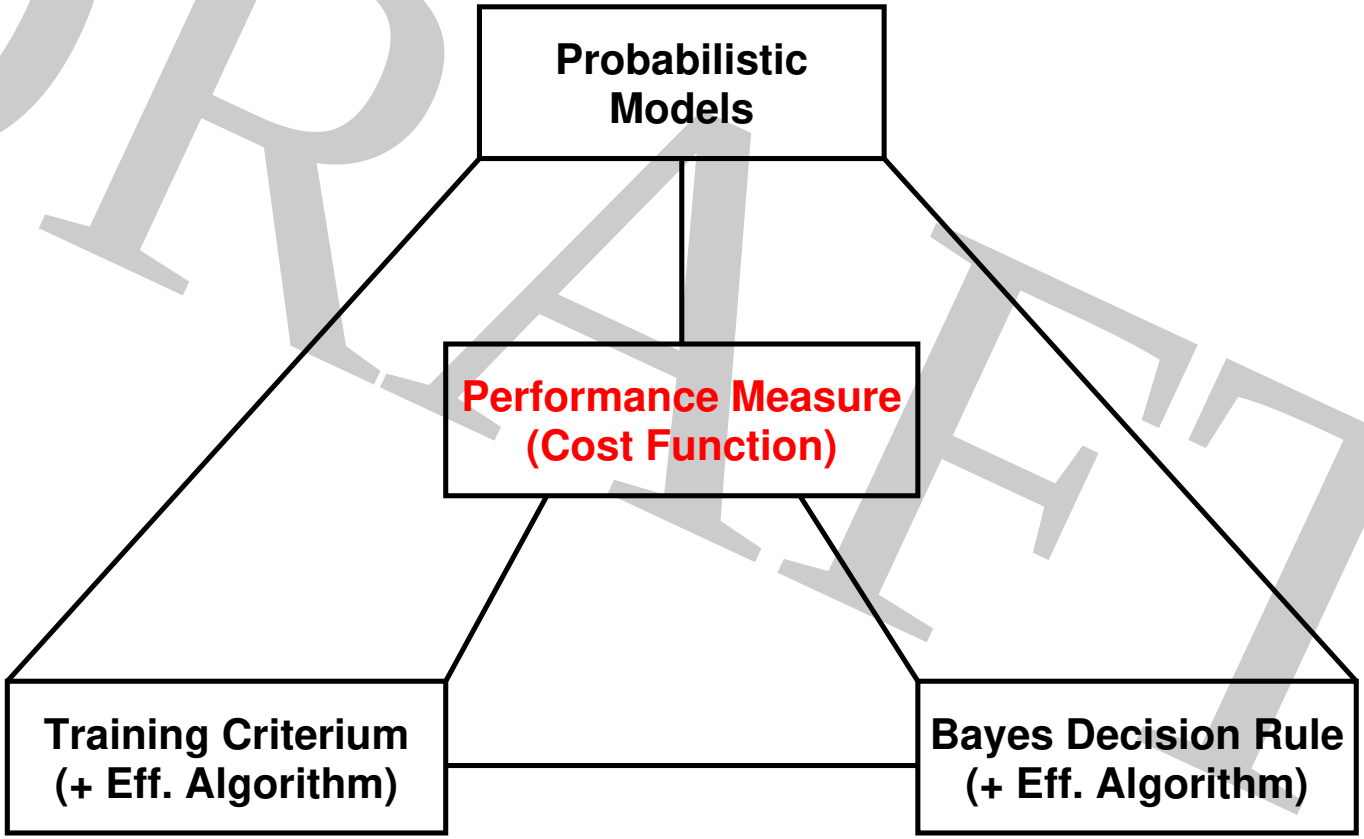
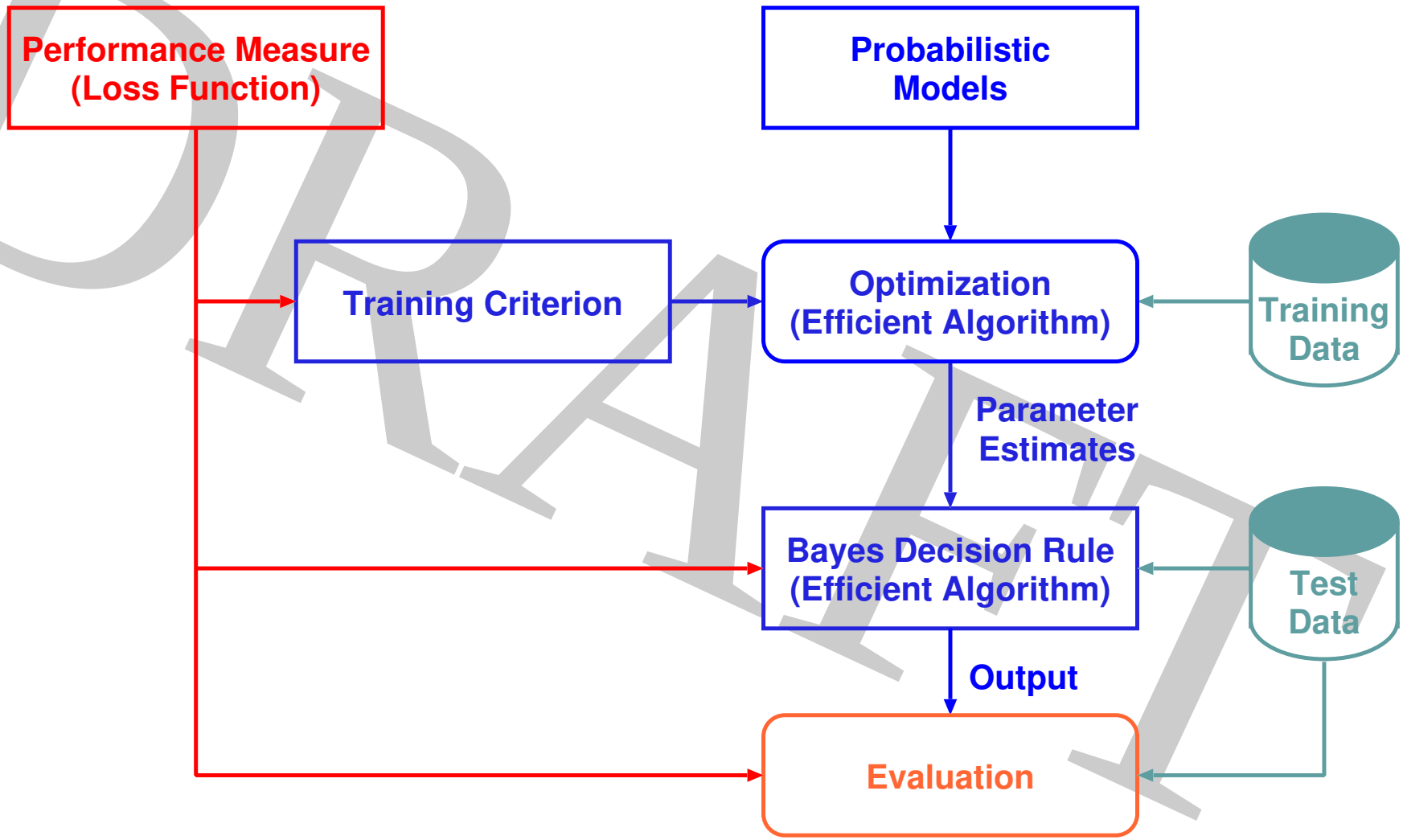


Illustration of the Statistical Approach



emphasis 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**

Remotely Related Work

- bounds to Bayes classification error [Fukunaga 72, ?]:
  – 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 $p_\vartheta(c|x)$
  - question: do we loose the optimality by these mismatch conditions?
- training criteria:
  - upper bounds for the difference between model and Bayes classification error
  - derived from the above upper bounds
  - relation between different training criteria
- questions:
  - do we need the true distribution for optimal performance? (necessary condition?)
  - do we need probability models at all? (as opposed to discriminant functions)

interpretation: a mathematically correct justification of the probabilistic approach

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:**

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

Example: Three Types of Model Outputs

example:

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

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

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

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

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

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

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

this example: the decision output does not change:

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

examples of typical model distributions:

- artificial neural net (or discriminant function)
- non-parametric model for discrete x
- maximum entropy or log-linear model
- decision tree (CART) approach
- classical approach (generative model, noisy-channel model):

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

  with class priors $p_\vartheta(c)$ and class-conditional model $p_\vartheta(x|c)$,
  e.g. single Gaussian distribution or Gaussian mixture

- Hidden Markov Model (in the classical approach)
  for an observation string $x = x_1^T$

idea of 'true' Bayesian modelling:

- **model: $p(y|\lambda)$ with $y = (x, c)$ and unknown parameter λ**
- **labelled training data: $Y = \{(x_n, c_n) : n = 1, ..., N\}$**
- **prior distribution for λ**

formalism: compute predictive distribution

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

theoretical advantage (?): no training problem

in practice: $p_\vartheta(y|Y)$
with other-type of unknown parameters ϑ:

- **what prior distribution $p(y|\lambda)$**
  and what hyperparameters?
- **what type of model $p(y|\lambda)$?**
  (Bayesian term: model selection)

6.3 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**

Classification Errors: Two Types

decision rules:

$$\text{model:} \quad c_\vartheta(x) = \arg\max_c \{p_\vartheta(c,x)\}$$
$$\text{Bayes:} \quad c_*(x) = \arg\max_c \{pr(c|x)\}$$

classification errors:

$$\text{model:} \quad E_\vartheta = \sum_x pr(x) \sum_c pr(c \neq c_\vartheta(x)|x) = \sum_x pr(x) \left[1 - pr(c_\vartheta(x)|x)\right]$$

$$\text{Bayes:} \quad E_* = \sum_x pr(x) \sum_c pr(c \neq c_*(x)|x) = \sum_x pr(x) \left[1 - pr(c_*(x)|x)\right]$$
$$= \sum_x pr(x) \left[1 - \max_c pr(c|x)\right]$$

consider the difference in classification error:

$$E_\vartheta - E_* = \sum_x pr(x) \left[pr(c_*(x)|x) - pr(c_\vartheta(x)|x)\right]$$
$$= \sum_x pr(x) \left[E_\vartheta(x) - E_*(x)\right]$$

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

consider difference of local classification errors in point x:

$$
\begin{aligned}
E_\vartheta(x) - E_*(x) &= pr(c_*(x)|x) - pr(c_\vartheta(x)|x) \\
&\quad \text{use: } p_\vartheta(c,x) \le p_\vartheta(c_\vartheta(x),x) \\
&\le pr(c_*(x)|x) - pr(c_\vartheta(x)|x) + p_\vartheta(c_\vartheta(x),x) - p_\vartheta(c_*(x),x) \\
&= \left[pr(c_*(x)|x) - p_\vartheta(c_*(x),x)\right] + \left[p_\vartheta(c_\vartheta(x),x) - pr(c_\vartheta(x)|x)\right] \\
&\quad \text{apply: } u \le |u| \\
&\le \left|pr(c_*(x)|x) - p_\vartheta(c_*(x),x)\right| + \left|pr(c_\vartheta(x)|x) - p_\vartheta(c_\vartheta(x),x)\right| \\
&\le \sum_c \left|pr(c|x) - p_\vartheta(c,x)\right|
\end{aligned}
$$

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

Local Bounds using l_r Norms

starting point is the inequality:

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

- **l_1 bound:**

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

- **l_∞ (or maximum) bound:**

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

- **l_2 bound (follows from l_∞) (warning: can be improved!):**

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

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

if $p_\vartheta(c, x) \to pr(c|x)$ **then** **Bound** $\to 0$

and $E_\vartheta(x) \to E_*(x)$

Improved l_2 Bound

the l_2 bound can be improved [Yang 2023, RWTH, pers.com.]:

$$
\begin{aligned}
0 \leq E_\vartheta(x) - E_*(x) &:= pr(c_*(x)|x) - pr(c_\vartheta(x)|x) \\
&\leq \sum_{c:\ c=c_*(x),c_\vartheta(x)} \left|pr(c|x) - p_\vartheta(c,x)\right| \\
&\leq \sqrt{2 \cdot \sum_{c:\ c=c_*(x),c_\vartheta(x)} \left|pr(c|x) - p_\vartheta(c,x)\right|^2} \\
&\leq \sqrt{2 \cdot \sum_{c} \left|pr(c|x) - p_\vartheta(c,x)\right|^2}
\end{aligned}
$$

using the inequality for $a, b \geq 0$: $\quad a + b \leq \sqrt{2 \cdot (a^2 + b^2)}$

(special case of *QM-AM-GM-HM Inequalities*, see Wikipedia)

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

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

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 $pr(x)$:

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

- squaring and using the variance inequality:

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

l_2 bound: similar procedure

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

6.4.1 Squared Error

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

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

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

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

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

practical relevance: we switched from true distribution to ideal targets

summary of re-writing:

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

training criterion for minimum classification error: minimize the upper bound

only the left term of the bound depends on ϑ:

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

which is the well-known squared error criterion.

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

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

Squared Error Criterion: Quadratic Normalization

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):

$$\hat{p}_{\hat{\vartheta}}(c,x) = \underset{\{p_\vartheta(c,x)\}}{\operatorname{argmin}} F(\vartheta) = \frac{pr(c|x)}{\sqrt{\sum_{c'} pr^2(c'|x)}}$$

$$\min_{\{p_\vartheta(c,x)\}} F(\vartheta) \stackrel{?}{=} \sum_x pr(x) \cdot \left(1 - \sqrt{\sum\nolimits_c pr^2(c|x)}\right)^2$$

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:

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

we consider the squared error criterion:

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

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

$$\begin{aligned} \hat{q}_{\hat{\vartheta}}(c,x) &= \operatorname*{argmin}_{\{q_\vartheta(c,x)\}} F(\vartheta) = pr^2(c|x) \\ &\min_{\{p_\vartheta(c,x)\}} F(\vartheta) = 0 \end{aligned}$$

practical training criterion: ?

no, not different from conventional squared error criterion

6.4.2 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:

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

summary of re-writing:

$$\left(E_\vartheta - E_*\right)^2 \leq \sum_{x,c} pr(x,c) \sum_{c'} \log \frac{1 - |\delta(c',c) - pr(c'|x)|}{1 - |\delta(c',c) - p_\vartheta(c',x)|}$$

training criterion for minimum classification error: minimize the upper bound

only the denominator of the argument of the logarithm depends on ϑ:

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

which is the so-called binary cross-entropy criterion [Solla & Levin$^+$ 88].

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

$$\begin{aligned} \hat{\vartheta} &:= \arg\max_\vartheta \{F(\vartheta)\} \\ &= \arg\max_\vartheta \left\{ \sum_n \left(\log\, p_\vartheta(c_n, x_n) + \sum_{c \neq c_n} \log\left[1 - p_\vartheta(c, x_n)\right]\right)\right\} \end{aligned}$$

6.4.3 Kullback-Leibler Divergence

Pinsker inequality for two distributions (normalized!) p_c and q_c [Fedotov & Harremoes$^+$ 03], [Cover & Thomas 91, pp. 300]:

$$\Big(\sum_c |p_c - q_c|\Big)^2 \leq 2 \cdot \sum_c p_c \log \frac{p_c}{q_c}$$

right-hand side: the Kullback-Leibler divergence (or relative entropy)

re-write l_1 bound (using variance inequality):

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

summary of re-writing:

$$\Big(E_\vartheta - E_*\Big)^2 \leq 2 \cdot \sum_x pr(x) \sum_c pr(c|x) \left[\log pr(c|x) - \log p_\vartheta(c|x)\right]$$

training criterion for minimum classification error: minimize the upper bound

only the right term depends on ϑ:

$$F(\vartheta) := \sum_x \sum_c pr(x,c) \log p_\vartheta(c|x)$$

which is the so-called cross-entropy or log-probability criterion.

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

$$\begin{aligned}\hat{\vartheta} &:= \arg\max_\vartheta \{F(\vartheta)\} \\ &= \arg\max_\vartheta \Big\{ \sum_n \log p_\vartheta(c_n|x_n) \Big\}\end{aligned}$$

6.5 Training Criteria Revisited: Squared Error vs. Cross-Entropy

assumption: normalized model:

$$p_\vartheta(c|x) \geq 0, \qquad \sum_c p_\vartheta(c|x) = 1$$

squared error criterion for normalized (!!) model:

$$F(\vartheta) = 1/N \cdot \sum_{n=1}^{N} \sum_c \left[p_\vartheta(c|x_n) - \delta(c, c_n)\right]^2$$

approach:

- **for each sample (c_n, x_n), we interpret the numeric squared error**
  $$\sum_c \left[p_\vartheta(c|x_n) - \delta(c, c_n)\right]^2$$
  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 $c \neq c_n$:

$$\sum_c \left[p_\vartheta(c|x_n) - \delta(c, c_n)\right]^2 \leq \left[p_\vartheta(c_n|x_n) - 1\right]^2 + \left[p_\vartheta(c_n|x_n) - 1\right]^2 = 2 \cdot \left[p_\vartheta(c_n|x_n) - 1\right]^2$$

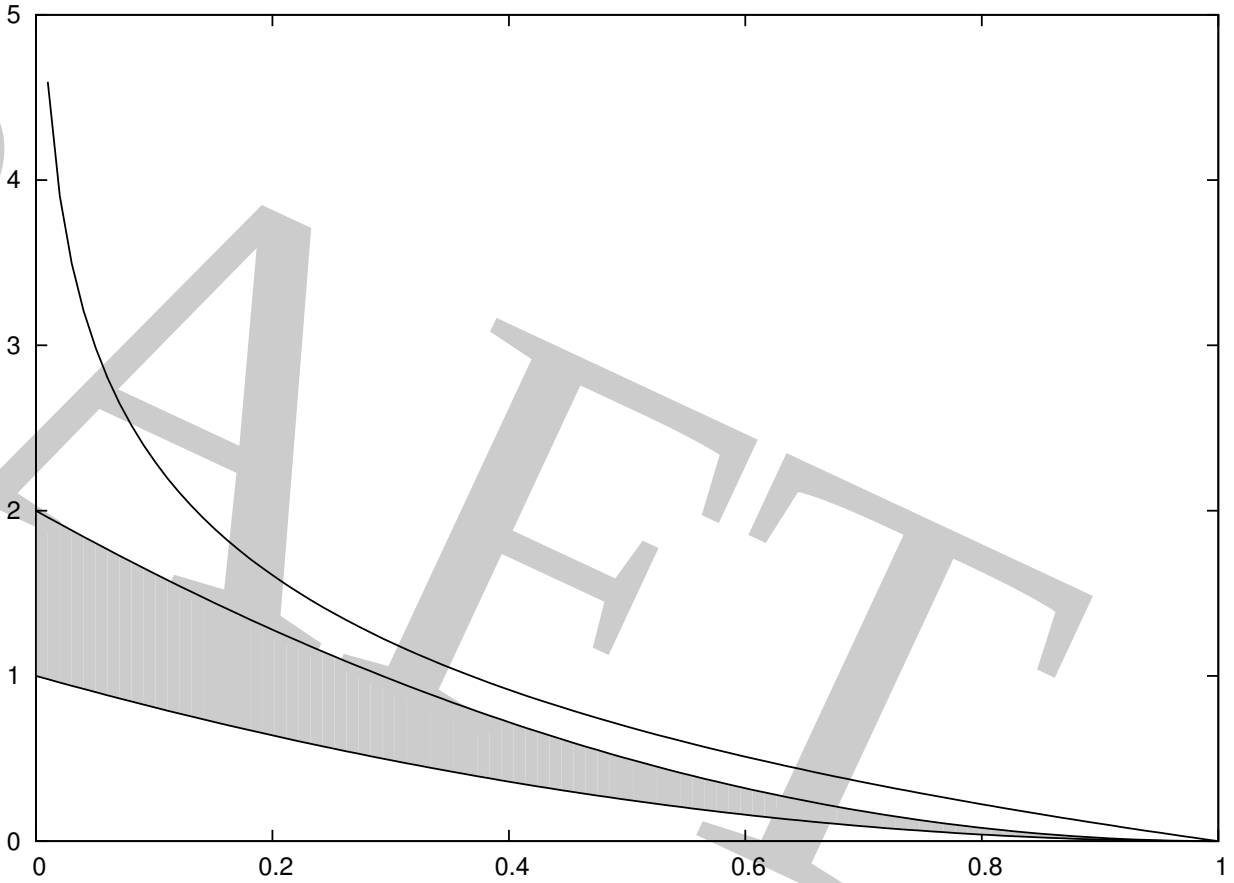
- lower bound: is attained when the remaining probability mass $[1 - p_\vartheta(c_n|x_n)]$ is uniformly distributed over all rival classes $c \neq c_n$:

$$\begin{aligned}\sum_c \left[p_\vartheta(c|x_n) - \delta(c, c_n)\right]^2 &\geq \left[p_\vartheta(c_n|x_n) - 1\right]^2 + [C-1] \cdot \left(\frac{p_\vartheta(c_n|x_n) - 1}{C-1}\right)^2 \\ &= \frac{C}{C-1} \cdot \left[p_\vartheta(c_n|x_n) - 1\right]^2 \\ &\geq \left[p_\vartheta(c_n|x_n) - 1\right]^2\end{aligned}$$

- result: factor 2 between upper and lower bound

plot:

- **horizontal axis:** $p_\vartheta(c_n|x_n)$
- **vertical axis:**
  a) negative logarithm
  b) squared error:
  lower and upper bound



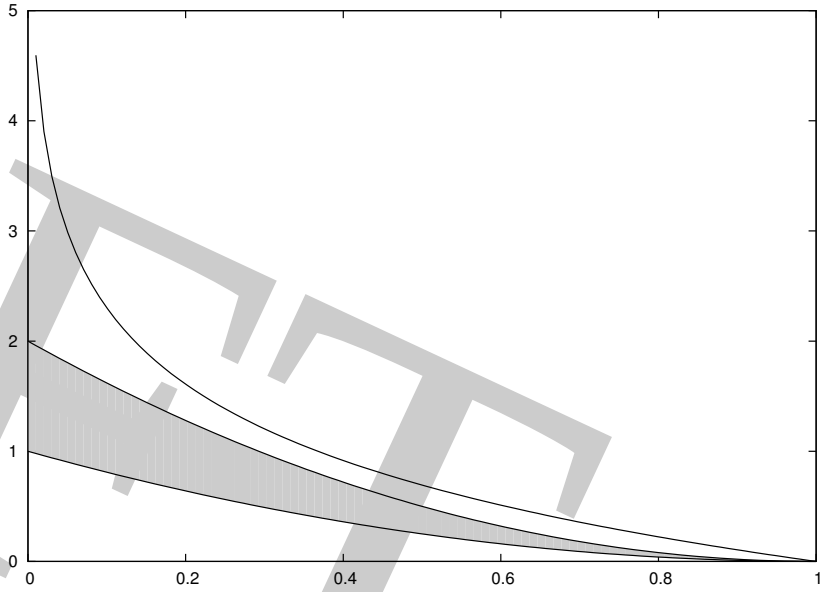
Training Criterion: Squared Error vs. Logarithm
Comparison of Functional Dependence

observations for squared error:

- **bounded over full range $0 \leq q(c_n|x_n) \leq 1$:**
- **factor of 2 between upper and lower bound**

observations for negative logarithm for threshold q_0 (e. g. $q_0 = 0.2$):

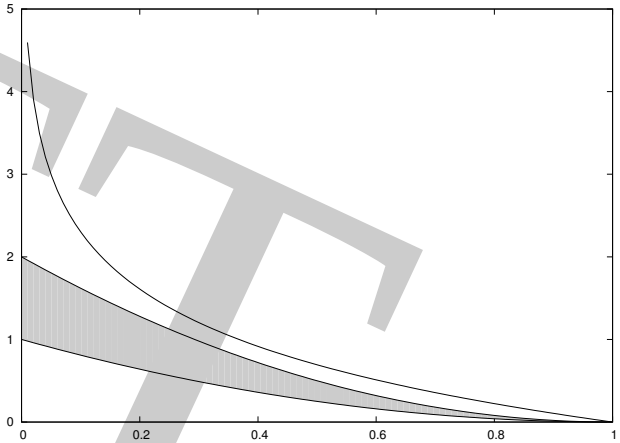
- **interval $0 \leq q(c_n|x_n) \leq q_0$:**
  - **no bound for $q(c_n|x_n) \to 0$**
  - **typical property of logarithm**
- **interval $q_0 \leq q(c_n|x_n) \leq 1$:**
  - **virtually identical shape like the upper bound of squared error**
  - **maybe surprising result**



Training Criterion: Squared Error vs. Logarithm
Comparison with MCE: minimum classification error

remarks:

- **squared error:**
  - **local score: exact value depends on *all* scores $q(c|x_n)$**
  - **for $q(c_n|x_n) \rightarrow 1$, the local score approaches 0**
  - **for $q(c_n|x_n) \rightarrow 0$, the local score is between 1 and 2**
  - **thus the local score CAN be interpreted as a 'smoothed' count of the classification error**
- **negative logarithm:**
  - **for $q(c_n|x_n) \rightarrow 1$, the local score approaches 0**
  - **for $q(c_n|x_n) \rightarrow 0$, the local score in not bounded**
  - **thus the score CANNOT be interpreted as a 'smoothed' count of the classification error**



- **suggested remedy (?):**
  - **MCE = minium classification error:**
  - **minimize the smoothed empirical error count (rate)**

for a correct error-count analysis,
we should consider the complete upper bound to the classification error
(rather than the practical training criterion):

$$\sum_c [p_c - q_c]^2 = \sum_c p_c \underbrace{\sum_{c'} [q_{c'} - \delta_{c'c}]^2}_{\text{practical training: squared error}} - \left(1 - \sum_c p_c^2\right)$$

$$\sum_c p_c \log \frac{p_c}{q_c} = \sum_c p_c \underbrace{[-\log q_c]}_{\text{practical training: cross-entropy}} + \sum_c p_c \log p_c$$

conclusions:

- squared error:
  - bounds for both practical criterion and classification error bound
- cross-entropy:
  - no upper bound for both practical criterion and classification error bound
  - consideration: problem only matters for *pathological* models $\{q_c\}$,
    unlike solutions close to minimum !

Squared Error and Cross-Entropy: Exercise in Chebyshev Approximation

exercise in mathematical optimization:

- **cubic approximation to the natural logarithm with $u \equiv q(c_n|x_n) < 1$:**

$$\begin{aligned} -\log u &= -[u-1] + \frac{1}{2}[u-1]^2 - \frac{1}{3}[u-1]^3 + \ldots \\ &\geq -[u-1] + \frac{1}{2}[u-1]^2 - \frac{1}{3}[u-1]^3 \end{aligned}$$

- **consider a quadratic approximation to the negative 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)**

Conclusions

summary:

- **goal: to study the model based classification error**
  - **explicit distinction to Bayes classification error**
  - **tight bounds on model based classification error**
- **three resulting bounds:**
  - **squared error: unconstrained model outputs**
  - **binary divergence: constrained model outputs**
  - **Kullback-Leibler 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?

Error Bounds: Summary

atomic decisions (or at string level):

- we have studied the model based classification error
  – explicit distinction between Bayes and model error
  – tight bounds on the squared difference $(E_\vartheta - E_*)^2$
- 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

7 Computer Simulations and Tight Error Bounds

7.1 Principle

7.2 Three Tight Bounds

7.3 Statistics and Information Theory

7.4 Bounds with Constraints A_*

8.1 From Discriminative to Generative Training

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

ASR (and other NLP) systems:
generative approach with an explicit language model

generative approach:

$$\begin{aligned} \textbf{true distribution:} \quad & pr(c,x) = pr(c) \cdot pr(x|c) \\ \textbf{model distribution:} \quad & q(c,x) = q(c) \cdot q(x|c) \end{aligned}$$

with
– class prior or language distribution: $q(c)$ **and** $pr(c)$
– observation distribution: $q(x|c)$ **and** $pr(x|c)$
note: change in notation: model $q(c,x)$ **rather than** $p_\vartheta(c,x)$

we compute the 'class posterior' model by re-normalization:

$$q(x) = \sum_c q(c,x)$$

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

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: $q(c|x) = q(c,x)/q(x)$

$$= \underbrace{\sum_{x} pr(x) \log \frac{q(x)}{pr(x)}}_{\leq 0} + \underbrace{\sum_{x,c} pr(c,x) \log \frac{pr(c,x)}{q(c,x)}}_{\geq 0}$$

$$\leq \sum_{x,c} pr(c,x) \log \frac{pr(c,x)}{q(c,x)}$$

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

note: $q(x)$ **depends on** $q(c,x)$**:** $q(x) = \sum_c q(c,x)$

summary of re-writing the (Kullback-Leibler) divergence bound:

$$\frac{1}{2} \cdot \Delta E_q^2 \leq \sum_{x,c} pr(c,x) \log \frac{pr(c|x)}{q(c|x)}$$

$$\leq \sum_{x,c} pr(c,x) \log \frac{pr(c,x)}{q(c,x)} = \sum_{c} pr(c) \log \frac{pr(c)}{q(c)} + \sum_{x,c} pr(c,x) \log \frac{pr(x|c)}{q(x|c)}$$

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)**

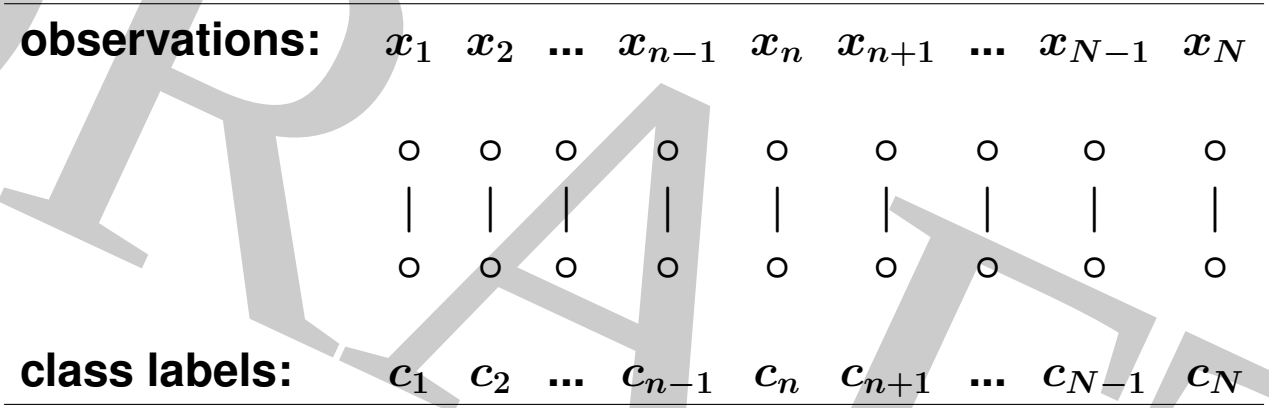
generative models: from cross-entropy to maximum likelihood:

- **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

8.2 Hamming Distance: Synchronized Strings

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



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: ΔE_n^q**

useful tool (later): log-sum inequality for non-negative numbers a_k and b_k

$$\left(\sum_k a_k\right) \log \frac{\sum_k a_k}{\sum_k b_k} \leq \sum_k a_k \log \frac{a_k}{b_k}$$

misnomer: better term = expected loss/risk;
correct terminology: expected loss L_q of model q over all (training) data:

$$
\begin{aligned}
E_q \equiv L_q &= \sum_{c_1^N, x_1^N} pr(c_1^N, x_1^N)\, L\big[[c^q]_1^N(x_1^N),\, c_1^N\big] \\
&\qquad \text{loss function:} \quad L[\tilde{c}_1^N, c_1^N] := \sum_n [1 - \delta(\tilde{c}_n, c_n)] \\
&\qquad \text{decision rule:} \quad x_1^N \to c_n^q(x_1^N) = \operatorname{argmax}_c \; q_n(c|x_1^N) \\
&= \sum_{c_1^N, x_1^N} pr(c_1^N, x_1^N) \sum_n \big[1 - \delta\big(c_n^q(x_1^N),\, c_n\big)\big] \\
&= \sum_n \Big(1 - \sum_{c_1^N, x_1^N} pr(c_1^N, x_1^N) \delta\big(c_n^q(x_1^N),\, c_n\big)\Big) \\
&= \sum_n \Big(1 - \sum_{c_n, x_1^N} pr(c_n, x_1^N) \delta\big(c_n^q(x_1^N),\, c_n\big)\Big) \\
&= \sum_n \Big(1 - \sum_{x_1^N} pr\big(c_n = c_n^q(x_1^N),\, x_1^N\big)\Big) \\
&= \sum_n \Big(1 - \sum_{x_1^N} pr(x_1^N)\, pr\big(c_n = c_n^q(x_1^N) \big| x_1^N\big)\Big)
\end{aligned}
$$

interpretation: this result (as it must be):

corresponds to 0/1 loss in position $n = 1, ..., N$ for an observation x_1^N

Kullback-Leibler Bound: From Marginal to Joint Distribution

re-write Kullback-Leibler bound for (c, x_1^N) in position n:

$$\frac{1}{2} \cdot \left[\Delta E_n^q\right]^2 \leq \sum_{x_1^N} pr(x_1^N) \sum_c pr_n(c|x_1^N) \log \frac{pr_n(c|x_1^N)}{q_n(c|x_1^N)}$$

$$= \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N: c_n = c} pr(c_1^N|x_1^N) \log \frac{\sum_{c_1^N: c_n = c} pr(c_1^N|x_1^N)}{\sum_{c_1^N: c_n = c} q(c_1^N|x_1^N)}$$

log-sum inequality: $\left(\sum_k a_k\right) \log \frac{\sum_k a_k}{\sum_k b_k} \leq \sum_k a_k \log \frac{a_k}{b_k}$

$$\leq \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N: c_n = c} pr(c_1^N|x_1^N) \log \frac{pr(c_1^N|x_1^N)}{q(c_1^N|x_1^N)}$$

$$= \sum_{x_1^N} pr(x_1^N) \sum_{c_1^N} pr(c_1^N|x_1^N) \log \frac{pr(c_1^N|x_1^N)}{q(c_1^N|x_1^N)}$$

important results:
– training at string level improves symbol level, too!
– training at symbol level is always better than at string level

l_1 Bound: From Marginal to Joint Distribution

starting point: l_1 bound:

$$\begin{aligned}
\Delta E_n^q &\leq \sum_{x_1^N} pr(x_1^N) \sum_c \left| pr_n(c|x_1^N) - q_n(c|x_1^N \right| \\
&= \sum_{x_1^N} pr(x_1^N) \sum_c \Big| \sum_{c_1^N: c_n = c} \left[pr(c_1^N|x_1^N) - q(c_1^N|x_1^N \right] \Big| \\
&\leq \sum_{x_1^N} pr(x_1^N) \sum_c \sum_{c_1^N: c_n = c} \left| \left[pr(c_1^N|x_1^N) - q(c_1^N|x_1^N \right] \right| \\
&\leq \sum_{x_1^N} pr(x_1^N) \sum_{c_1^N} \left| pr(c_1^N|x_1^N) - q(c_1^N|x_1^N \right|
\end{aligned}$$

results:

– we have extended the l_1 bound
from the marginal to the joint distribution

– for position-dep. WER in synchronized strings:
the three tight bounds apply at the string level

8.3 General Loss Function

remarks:
– model-based expected Bayes risk
– metric loss functions
– early stage
– experimental resuls: yes, difficult to intepret
(Povey: exected WER, Hain: state-level minimmum Bayes risk)

decision rules:

$$\textbf{model:} \quad c_\vartheta(x) = \arg\min_{\tilde{c}} \Big\{ \sum_c p_\vartheta(c|x)\, L[\tilde{c}, c] \Big\}$$

$$\textbf{Bayes:} \quad c_*(x) = \arg\min_{\tilde{c}} \Big\{ \sum_c pr(c|x)\, L[\tilde{c}, c] \Big\}$$

expectation of loss using empirical distribution:

$$\textbf{model:} \quad L_\vartheta = \sum_x pr(x) \sum_c pr(c|x)\, L[c_\vartheta(x), c]$$

$$\textbf{Bayes:} \quad L_* = \sum_x pr(x) \sum_c pr(c|x)\, L[c_*(x), c] = \sum_x pr(x) \min_{\tilde{c}} \Big\{ \sum_c pr(c|x)\, L[\tilde{c}, c] \Big\}$$

consider the difference in expected losses:

$$\begin{aligned} L_\vartheta - L_* &= \sum_x pr(x) \sum_c pr(c|x) \big(L[c_\vartheta(x), c] - L[c_\vartheta(x), c] \big) \\ &= \sum_x pr(x) \big[L_\vartheta(x) - L_*(x) \big] \end{aligned}$$

using the local losses $L_\vartheta(x)$ and $L_*(x)$ in point x

consider the difference in local losses:

$$
\begin{aligned}
L_\vartheta(x) - L_*(x) &= \sum_c pr(c|x) \left(L[c_\vartheta(x), c] - L[c_*(x), c] \right) \\
&\leq \sum_c pr(c|x) \left(L[c_\vartheta(x), c] - L[c_*(x), c] \right) \\
&\quad + \sum_c p_\vartheta(c|x) \left(L[c_*(x), c] - L[c_\vartheta(x), c] \right) \\
&= \sum_c \left[pr(c|x) - p_\vartheta(c|x) \right] \left(L[c_\vartheta(x), c] - L[c_*(x), c] \right) \\
&\leq \sum_c \left| pr(c|x) - p_\vartheta(c|x) \right| \left| L[c_\vartheta(x), c] - L[c_*(x), c] \right|
\end{aligned}
$$

assumption: triangle inequality

$$
\begin{aligned}
&\leq L[c_\vartheta(x), c_*(x)] \sum_c \left| pr(c|x) - p_\vartheta(c|x) \right| \\
&\leq I_x \cdot \sum_c \left| pr(c|x) - p_\vartheta(c|x) \right|
\end{aligned}
$$

with I_x: maximum number of output symbols for input string x

Three Loss Functions

three specific loss functions:

- **0/1 loss:**
- **Hamming loss for synchronized strings**

$$L[\tilde{c}, c] := ... \leq N$$

with N denoting the string length (as given by observation x)

- **edit distance for strings:**

$$L[\tilde{c}, c] := \min_{A}\{L_A[\tilde{c}, c]\} = ...$$

A: alignment with sub/ins/del operations

$$\leq \max\{|\tilde{c}|, |c|\}$$

with $|c|$ denoting the length of string c.

Expected Loss for General Loss Function

(true) Bayes decision rule:

$$x \rightarrow c_*(x) = \underset{c}{\operatorname{argmin}} \left\{ L_*[c|x] \right\} = \underset{c}{\operatorname{argmin}} \left\{ \sum_{\tilde{c}} pr(\tilde{c}|x)\, L[c, \tilde{c}] \right\}$$

***pseudo* Bayes decision rule:**

$$x \rightarrow c_q(x) = \underset{c}{\operatorname{argmin}} \left\{ L_q[c|x]) \right\} = \underset{c}{\operatorname{argmin}} \left\{ \sum_{\tilde{c}} q(\tilde{c}|x)\, L[c, \tilde{c}] \right\}$$

misnomer in literature:

***Bayes* risk/loss refers to the use of the exact loss function (e. g. edit distance) rather than true distribution.**

difference of expected losses
between the two decison rules on true distribution $pr(c, x)$:

$$
\begin{aligned}
0 \leq L_q - L_* &= \sum_x pr(x) \underbrace{\sum_{\tilde{c}} pr(\tilde{c}|x) \cdot \Big(L[c_q(x), \tilde{c}] - L[c_*(x), \tilde{c}] \Big)}_{\geq 0} \\
&\leq \sum_x pr(x) \sum_{\tilde{c}} pr(\tilde{c}|x) \cdot \Big(L[c_q(x), \tilde{c}] - L[c_*(x), \tilde{c}] \Big) \\
&\quad + \sum_x pr(x) \underbrace{\sum_{\tilde{c}} q(\tilde{c}|x) \cdot \Big(L[c_*(x), \tilde{c}] - L[c_q(x), \tilde{c}] \Big)}_{\geq 0} \\
&= \sum_x pr(x) \sum_{\tilde{c}} \Big(pr(\tilde{c}|x) - q(\tilde{c}|x) \Big) \cdot \Big(L[c_q(x), \tilde{c}] - L[c_*(x), \tilde{c}] \Big)
\end{aligned}
$$

note how the model-based decision rule for expected loss is used for this inequality!

Metric Loss Function: l_1 Bound

we re-write:

$$
\begin{aligned}
0 \leq\ L_q - L_*\ le\ & \sum_x pr(x) \sum_{\tilde{c}} \Big(pr(\tilde{c}|x) - q(\tilde{c}|x) \Big) \cdot \Big(L[c_q(x), \tilde{c}] - L[c_*(x), \tilde{c}] \Big) \\
& \leq \sum_x pr(x) \sum_{\tilde{c}} \Big| pr(\tilde{c}|x) - q(\tilde{c}|x) \Big| \cdot \underbrace{\Big| L[c_q(x), \tilde{c}] - L[c_*(x), \tilde{c}] \Big|}_{\text{use triangle inequality}} \\
& \leq \sum_x pr(x) \sum_{\tilde{c}} \Big| pr(\tilde{c}|x) - q(\tilde{c}|x) \Big| \cdot L[c_q(x), c_*(x)] \\
& = \sum_x pr(x)\, L[c_q(x), c_*(x)] \sum_{\tilde{c}} \Big| pr(\tilde{c}|x) - q(\tilde{c}|x) \Big|
\end{aligned}
$$

three specific loss functions:

– 0/1 loss:

$$L[c_q(x), c_*(x)] \leq 1$$

– Hamming distance for strings of length N:

$$L[c_q(x), c_*(x)] \leq N$$

– edit distance for strings with maximum length N_{max}:

$$L[c_q(x), c_*(x)] \leq N_{max}$$

Metric Loss Function: l_1 Bound

expected (Bayes) risk and s-MBR

see extended Hamming distance: Section: 5.3.4 Edit Distance and Symbol Posterior Probability

Metric Loss Function: l_1 Bound

important conclusions:

- **from the l_1 bound, we can derive the usual three bounds:**
  – squared error
  – binary divergence
  – Kullback-Leibler divergence
  at the level of the string distributions.
- **are there tighter (useful) bounds?**
  (as we have derived for the Hamming distance, see previous section)
- **discrepancy:**
  different from the heuristic concept of *expected Bayes risk*

8.4 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)

9 DRAFT: 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:

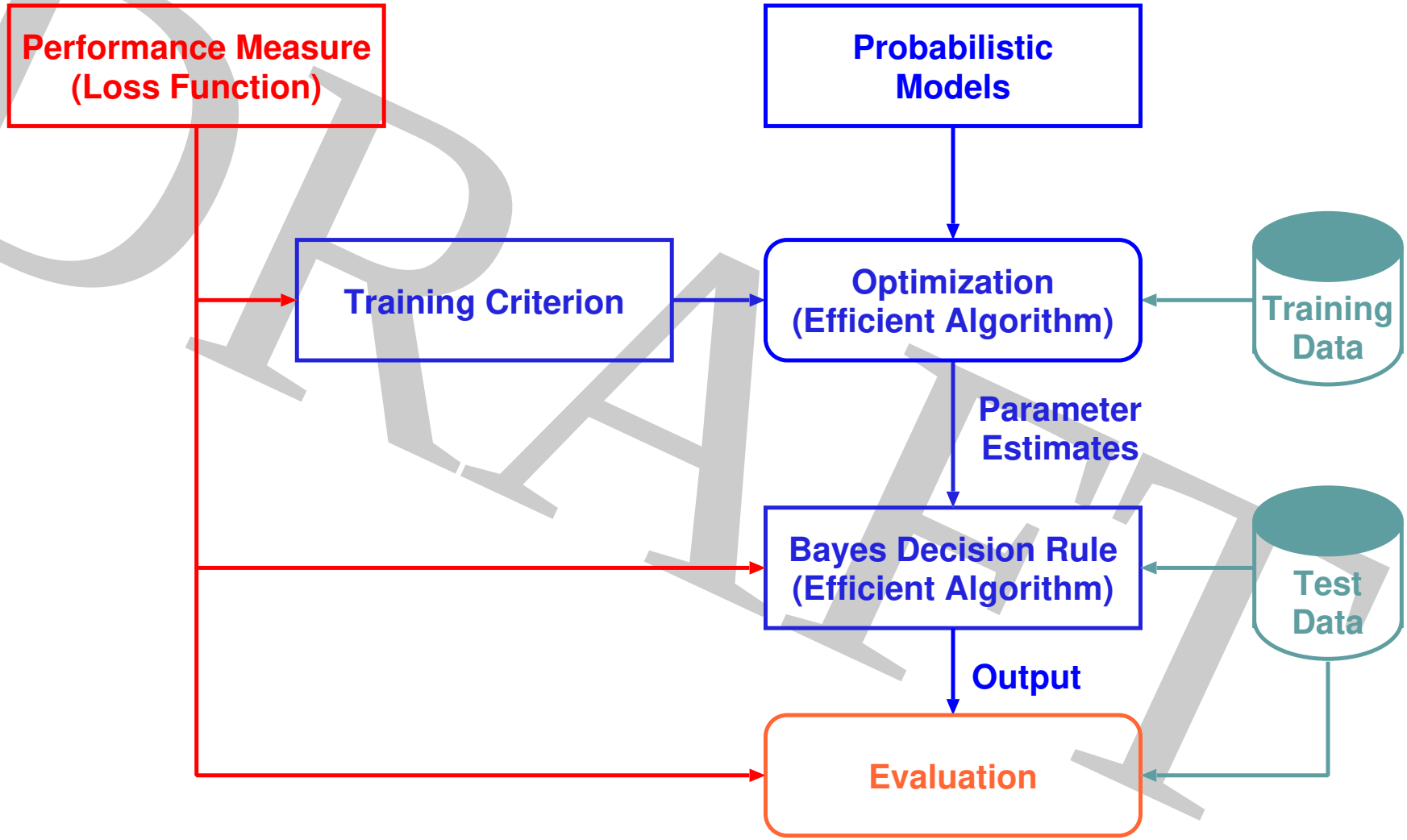
– there has been life before ANNs and deep learning:
  Bayes decision rule, Gaussian modelling, HMMs, discriminative training, ...
– ANNs helped to improve the performance dramatically

Conclusions: ANNs in Speech and Language Technology

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?**

Data Driven Approach to Speech and Language Tasks



Conclusions

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 succesful?**
- **...**

10 References

[Baevski & Schneider+ 20] A. Baevski, S. Schneider, M. Auli: VQ-Wav2Vec: Self-Supervised Learning of Discrete Speech Representations. Facebook AI Research, Menlo Park, CA, arxiv, 16-Feb-2021.

[Bahdanau & Cho+ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural machine translation by jointly learning to align and translate. Int. Conf. on Learning and Representation (ICLR), San Diego, CA, May 2015.

[Bahl & Jelinek+ 83] L. R. Bahl, F. Jelinek, R. L. Mercer: A Maximum Likelihood Approach to Continuous Speech Recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 5, pp. 179-190, March 1983.

[Bahl & Brown+ 86] L. R. Bahl, P. F. Brown, P. V. de Souza, R. L. Mercer: Maximum mutual information estimation of hidden Markov parameters for speech recognition. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Tokyo, pp.49-52, April 1986.

[Beck & Schlüter+ 15] E. Beck, R. Schlüter, H. Ney: Error Bounds for Context Reduction and Feature Omission, Interspeech, Dresden, Germany, Sep. 2015.

[Bang & Cahyawijaya+ 23] Y. Bang, S. Cahyawijaya, N. Lee et al.: A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. HKUST, arxiv, 28-Feb-2023.

[Bengio & Ducharme+ 00] Y. Bengio, R. Ducharme, P. Vincent: A neural probabilistic language model. Advances in Neural Information Processing Systems (NIPS), pp. 933-938, Denver, CO, USA, Nov. 2000.

[Bishop 95a] C. M. Bishop: Neural Networks for Pattern Recognition. Oxford University Press, 1995.

[Bishop 95b] C. M. Bishop: Training with Noise is Equivalent to Tikhonov Regularization. Neural Computation, 7(1), pp. 108-116, Jan. 1995.

[Botros & Irie+ 15] R. Botros, K. Irie, M. Sundermeyer, H. Ney: On Efficient Training of Word Classes and Their Application to Recurrent Neural Network Language Models. Interspeech, pp.1443-1447, Dresden, Germany, Sep. 2015.

[Bourlard & Wellekens 87] H. Bourlard, C. J. Wellekens: Multilayer perceptrons and automatic speech recognition. First Int. Conf. on Neural Networks, pp. 407-416, San Diego, CA, 1987.

[Bourlard & Wellekens 89] H. Bourlard, C. J. Wellekens: 'Links between Markov Models and Multilayer Perceptrons', in D.S. Touretzky (ed.): "Advances in Neural Information Processing Systems I", Morgan Kaufmann Pub., San Mateo, CA, pp.502-507, 1989.

[Bridle 82] J. S. Bridle, M. D. Brown, R. M. Chamberlain: An Algorithm for Connected Word Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Paris, pp. 899-902, May 1982.

[Bridle 89] J. S. Bridle: Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition, in F. Fogelman-Soulie, J. Herault (eds.): 'Neuro-computing: Algorithms, Architectures and Applications', NATO ASI Series in Systems and Computer Science, Springer, New York, 1989.

[Bridle & Dodd 91] J. S. Bridle, L. Dodd: An Alphanet Approach To Optimising Input Transformations for Continuous Speech Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Toronot, pp. 277-280, April 1991.

[Brown & Della Pietra$^+$ 93] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, R. L. Mercer: Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, Vol. 19.2, pp. 263-311, June 1993.

[Brown & Mann$^+$ 22] T. R. Brown, B. Mann, N. Ryder et al.: Language Models are Few-Shot Learners. OpenAI (GPT-3), arxiv, 22-Jul-2022.

[Collobert & Weston 08] R. Collobert, J. Weston: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. Int. Conference on Machine Learning (ICML), 2008.

[Collobert & Weston$^+$ 11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa: Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research, 2011.

[Castano & Vidal$^+$ 93] M.A. Castano, E. Vidal, F. Casacuberta: Inference of stochastic regular languages through simple recurrent networks. IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives, pp. 16/1-6, Colchester, UK, April 1993.

[Castano & Casacuberta 97] M. Castano, F. Casacuberta: A connectionist approach to machine translation. European Conf. on Speech Communication and Technology (Eurospeech), pp. 91–94, Rhodes, Greece, Sep. 1997.

[Castano & Casacuberta+ 97] M. Castano, F. Casacuberta, E. Vidal: Machine translation using neural networks and finite-state models. Int. Conf. on Theoretical and Methodological Issues in Machine Translation (TMI), pp. 160-167, Santa Fe, NM, USA, July 1997.

[Chien & Lu 15] : Jen-Tzung Chien, Tsai-Wei Lu: Deep Recurrent RegularizatioN Neural Network for Speech Recognition. ICASSP 2015, pp. 4560-4564.

[Cover & Thomas 91] T. M. Cover, J. A. Thomas: Elements of Information Theory. John Wiley & Sons, New York, NY, 1991.

[Dahl & Ranzato+ 10] G. E. Dahl, M. Ranzato, A. Mohamed, G. E. Hinton: Phone recognition with the mean-covariance restricted Boltzmann machine. Advances in Neural Information Processing Systems (NIPS) 23, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, Eds. Cambridge, MA, MIT Press, 2010, pp. 469-477.

[Dahl & Yu+ 12] G. E. Dahl, D. Yu, L. Deng, A. Acero: Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. IEEE Tran. on Audio, Speech and Language Processing, Vol. 20, No. 1, pp. 30-42, Jan. 2012.

[Dehak & Kenny+ 11] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, P. Ouellet: Front-End Factor Analysis for Speaker Verification IEEE Trans. on audio, speech, and language processing, pp. 788-798, Vol. 19, No. 4, May 2011.

[Devlin & Zbib+ 14] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, J. Makhoul: Fast and Robust Neural Network Joint Models for Statistical Machine Translation. Annual Meeting of the ACL, pp. 1370–1380, Baltimore, MA, June 2014.

[Devroye & Györfi+ 96] L. Devroye, J. Györfi, G. Lugosi: A Probabilistic Theory of Pattern Recognition. Springer, New York, 1996.

[Doetsch & Hannemann+ 17] P. Doetsch , M. Hannemann, R. Schlüer, H. Ney: Inverted Alignments for End-to-End Automatic Speech Recognition. IEEE Journal of selected topics in Signal Processing, Vol. 11, No. 8, pp. 1265-1273, Dec. 2017.

[Duda & Hart 73] R. O. Duda, P. E. Hart: Pattern Classification and Scene Analysis. Wiley, Hoboken, 1973.

[Fedotov & Harremoes+ 03] A. A. Fedotov, P. Harremoes, F. Topsoe: Refinements of Pinsker's Inequality. Some Inequalities for Information Divergence and Related Measures of Discrimination. IEEE Trans. on Information Theory, Vol. 49, No. 6, pp. 1491-1498, June 2003.

[Forcada & Carrasco 05] M. L. Forcada, R. C. Carrasco: Learning the initial state of a second-order recurrent neural network during regular language inference. Neural Computation, Vol. 7, No. 5, pp. 923-930, Sep. 2005.

[Fontaine & Ris+ 97] V. Fontaine, C. Ris, J.-M. Boite: Nonlinear discriminant analysis for improved speech recognition. Eurospeech, Rhodes, Greece, Sep. 1997.

[Fritsch & Finke+ 97] J. Fritsch, M. Finke, A. Waibel: Adaptively Growing Hierarchical Mixtures of Experts. NIPS, Advances in Neural Information Processing Systems 9, MIT Press, pp. 459-465, 1997.

[Fukunaga 72] K. Fukunaga: Introduction to Statistical Pattern Recognition. Academic Press, New York, 1972.

[Gemello & Manai+ 06] R. Gemello, F. Mana, S. Scanzio, P. Lafac, R. De Mori: Adaptation of Hybrid ANN/HMM Models Using Linear Hidden Transformations and Conservative Training. IEEE Int. Conf. on Acoustics Speech and Signal Processing Proceedings, Toulouse, 2006.

[Gers & Schmidhuber+ 00] F. A. Gers, J. Schmidhuber, F. Cummin: Learning to forget: Continual prediction with LSTM. Neural computation, Vol 12, No. 10, pp. 2451-2471, 2000.

[Gers & Schraudolph+ 02] F. A. Gers, N. N. Schraudolph, J. Schmidhuber: Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, Vol. 3, pp. 115-143, 2002.

[Goldberg 17] Y. Goldberg: Neural Network Methods in Natural Language Processing. Morgan & Claypool Publishers, 2017.

[Graves 12] A. Graves: Sequence Transduction with Recurrent Neural Networks. U of Toronto, Canada, arxiv, 12-Nov-2012.

[Graves & Fernandez+ 06] A. Graves, S. Fernandez, F Gomez, J. Schmidhuber: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. Int. Conf. on Machine Learning, Pittsburgh, PA, pp. 369-376, 2006.

[Graves & Schmidhuber 09] A. Graves, J. Schmidhuber: Offline handwriting recognition with multidimensional recurrent neural networks. NIPS 2009.

[Grezl & Fousek 08] F. Grezl, P. Fousek: Optimizing bottle-neck features for LVCSR. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 4729-4732, Las Vegas, NV, March 2008.

[Grosicki & El Abed 09] E. Grosicki, H. El Abed: ICDAR 2009 Handwriting Recognition Competition. Int. Conf. on Document Analysis and Recognition (ICDAR) 2009, Barcelona, pp. 139-1402, July 2009.

[Haffner 93] P. Haffner: Connectionist Speech Recognition with a Global MMI Algorithm. 3rd Europ. Conf. on Speech Communication and Technology (Eurospeech'93), Berlin, Germany, Sep. 1993.

[Hampshire & Pearlmutter 90] J. B. Hampshire, B. Pearlmutter: Equivalence Proofs for Multilayer Perceptron Classifiers and the Bayesian Discriminant Function. In D. S. Touretzky, J. L. Hinton, T. J. Sejnowksi, G. E. Hinton (eds.): *Proceedings of the 1990 Connectionist Summer School*, pp. 159-172, San Mateo, CA, Morgan Kaufmann.

[Heigold & Macherey 05+] G. Heigold, W. Macherey, R. Schlüter, H. Ney: Minimum Exact Word Error Training. IEEE ASRU workshop, pp. 186-190, San Juan, Puerto Rico, Nov. 2005.

[Heigold & Schlüter 12+] G. Heigold, R. Schlüter, H. Ney, S. Wiesler: Discriminative Training for Automatic Speech Recognition: Modeling, Criteria, Optimization, Implementation, and Performance. IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 58-69, Nov. 2012.

[Hermansky & Ellis+ 00] H. Hermansky, D. W. Ellis, S. Sharma: Tandem connectionist feature extraction for conventional HMM systems. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 1635-1638, Istanbul, Turkey, June 2000.

[Hinton & Osindero+ 06] G. E. Hinton, S. Osindero, Y. Teh: A fast learning algorithm for deep belief nets. Neural Computation, Vol. 18, No. 7, pp. 1527-1554, July 2006.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long short-term memory. Neural Computation, Vol. 9, No. 8, pp. 1735–1780, Nov. 1997.

[Ivakhnenko 71] A. G. .Ivakhnenko: Polynomial theory of complex systems. IEEE Transactions on Systems, Man and Cybernetics, Vol. 1, No. 4, pp. 364-378, Oct. 1971.

[Jelinek & Mercer+ 77] F. Jelinek, R. L. Mercer, L. R. Bahl: Perplexity – a measure of the difficulty of speech recognition tasks. Journal of the Acoustical Society of America, 1977.

[Kaltenbrenner & Blunsom 13] N. Kalchbrenner, P. Blunsom: Recurrent continuous translation models. EMNLP 2013.

[Juang & Chou+ 97] B.-H. Juang, W. Chou, C.-H. Lee: Minimum Classification Error Rate Methods for Speech Recognition. IEEE Transactions on Speech and Audio Processing, Vol. 5, No. 3, pp. 257-265, May 1997.

[Juang & Katagiri 92] B.-H. Juang, S. Katagiri: Discriminative Learning for Minimum Error Classification. IEEE Transactions on Signal Processing, Vol. 40, No. 12, pp. 3043-3054, Dec. 1992.

[Klakow & Peters 02] D. Klakow, J. Peters: Testing the correlation of word error rate and perplexity. Speech Communication, pp. 19–28, 2002.

[Koehn & Och+ 03] P. Koehn, F. J. Och, D. Marcu: Statistical Phrase-Based Translation. HLT-NAACL 2003, pp. 48-54, Edmonton, Canada, May-June 2003.

[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 & Corrado$^+$ 13] T. Mikolov, G. Corrado, K. Chen, J. Dean: Efficient Estimation of Word Representations in Vector Space. Google, arxiv, 07-Sep-2013.

[Mikolov & Karafiat$^+$ 10] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, S. Khudanpur: Recurrent neural network based language model. Interspeech, pp. 1045-1048, Makuhari, Chiba, Japan, Sep. 2010.

[Mohamed & Dahl$^+$ 09] A. Mohamed, G. Dahl, G. Hinton: Deep belief networks for phone recognition. NIPS Workshop Deep Learning for Speech Recognition and Related Applications, 2009.

[Morgan & Bourlard 90] N. Morgan, H. Bourlard: Continuous speech recognition using multilayer perceptrons with hidden Markov models. ICASSP 1990, pp. 413-416, Albuquerque, NM, 1990.

[Nakamura & Shikano 89] M. Nakamura, K. Shikano: A Study of English Word Category Prediction Based on Neural Networks. ICASSP 89, p. 731-734, Glasgow, UK, May 1989.

[Neco & Forcada 97] R. P. Neco, M. L. Forcada: Asynchronous translations with recurrent neural nets. IEEE Int. Conf. on Neural Networks, pp. 2535-2540, June 1997.

[Ney 03] H. Ney: On the Relationship between Classification Error Bounds and Training Criteria in Statistical Pattern Recognition. First Iberian Conf. on Pattern Recognition and Image Analysis, Puerto de Andratx, Spain, Springer LNCS Vol. 2652, pp. 636-645, June 2003.

[Ney 84] H. Ney: The Use of a One–Stage Dynamic Programming Algorithm for Connected Word Recognition. IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-32, No. 2, pp. 263-271, April 1984.

[Ney & Haeb-Umbach$^+$ 92] H. Ney, R. Haeb-Umbach, B.-H. Tran, M. Oerder: Improvements in Beam Search for 10000-Word Continuous Speech Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, San Francisco, CA, pp. 13-16, March 1992.

[Normandin & Cardin+ 94] Y. Normandin, R. Cardin, R. De Mori: High-Performance Connected Digit Recognition Using Maximum Mutual Information Estimation. IEEE Trans. on Speech and Audio Processing, vol. 2, no. 2, pp. 299-311, April 1994.

[Ouyang & Wu+ 22] L. Ouyang, J. Wu, X. Jiang et al.: Training language models to follow instructions with human feedback. OpenAI, arxiv, 04-Mar-2022.

[Och & Ney 03] F. J. Och, H. Ney: A Systematic Comparison of Various Alignment Models. *Computational Linguistics,* Vol. 29, No. 1, pp. 19-51, March 2003.

[Och & Ney 04] F. J. Och, H. Ney: The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, Vol. 30, No. 4, pp. 417-449, Dec. 2004.

[Och & Tillmann+ 99] F. J. Och, C. Tillmann, H. Ney: Improved Alignment Models for Statistical Machine Translation. Joint ACL/SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora, College Park, MD, pp. 20-28, June 1999.

[Patterson & Womack 66] J. D. Patterson, B. F. Womack: An Adaptive Pattern Classification Scheme. IEEE Trans. on Systems, Science and Cybernetics, Vol.SSC-2, pp.62-67, Aug. 1966.

[Pereyra & Tucker+ 17] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, G. Hinton: Regularizing Neural Networks bY Penalizing Confident Output Distributions. arxiv, 23-Jan-2017, ICLR 2017.

[Povey & Woodland 02] D. Povey, P.C. Woodland: Minimum phone error and I-smoothing for improved discriminative training. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 105–108, Orlando, FL, May 2002.

[Printz & Olsen 02] H. Printz, P. A. Olsen: Theory and practice of acoustic confusability. Computer Speech and Language, pp. 131–164, Jan. 2002.

[Radford & Wu+ 18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever: Language Models are Unsupervised Multitask Learners. OpenAI (GPT-2), preprint, 2018.

[Radford & Narasimhan+ 19] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever: Improving Language Understanding by Generative Pre-Training. OpenAI (GPT-1), preprint, 2019.

[Raissi & Beck+ 20] T. Raissi, E. Beck, R. Schlüter, H. Ney: Context-Dependent Acoustic Modeling without Explicit Phone Clustering arxiv, 2020.

[Raissi & Beck+ 21] T. Raissi, E. Beck, R. Schlüter, H. Ney: Towards Consistent Hybrid HMM Acoustic Modeling. arxiv, 2021.

[Raissi & Beck+ 22] T. Raissi, E. Beck, R. Schlüter, H. Ney: Improving Factored Hybrid HMM Acoustic Modeling without State Tying. arxiv, 2022.

[Robinson 94] A. J. Robinson: An Application of Recurrent Nets to Phone Probability Estimation. IEEE Trans. on Neural Networks, Vol. 5, No. 2, pp. 298-305, March 1994.

[Sainath & Weiss+ 16] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, M. Bacchiani: Factored Spatial and Spectral Multichannel Raw Waveform CLDNNs, Proc. ICASSP, 2016.

[Saon & Tüske+ 2021] G. Saon, Z. Tüske, D. Bolanos, B. Kingsbury: Advancing RNN Transducer Technology for Speech Recognition. IBM Research AI, Yorktown Heights, USA, arxiv, 17-Mar-2021.

[Sakoe & Chiba 71] H. Sakoe, S. Chiba: A Dynamic Programming Approach to Continuous Speech Recognition. Proc. 7th Int. Congr. on Acoustics, Budapest, Hungary, Paper 20 C 13, pp. 65-68, August 1971.

[Sak & Shannon+ 17] H. Sak, M. Shannon, K. Rao, F. Beaufays: Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping. Interspeech, Stockholm, Sweden, pp. 1298-1302, Aug. 2017.

[Schlüter & Beck+ 19] R. Schlüter, E. Beck, H. Ney: Upper and Lower Tight Error Bounds for Feature Omission with an Extension to Context Reduction. IEEE Trans. Pattern Anal. Mach. Intell., Vol. 41, No. 2, pp. 502-514, 2019.

[Schlüter & Nussbaum+ 11] R. Schlüter, M. Nussbaum-Thom, H. Ney: On the Relationship between Bayes Risk and Word Error Rate in ASR. IEEE Trans. on Audio, Speech, and Language Processing, vol. 19, no. 5, p. 1103-1112, July 2011.

[Schlüter & Nussbaum+ 12] R. Schlüter, M. Nussbaum-Thom, H. Ney: Does the Cost Function Matter in Bayes Decision Rule? IEEE Trans. PAMI, No. 2, pp. 292–301, Feb. 2012.

[Schlüter & Nussbaum-Thom+ 13] R. Schlüter, M. Nußbaum-Thom, E. Beck, T. Alkhouli, H. Ney: Novel Tight Classification Error Bounds under Mismatch Conditions based on f-Divergence. IEEE Information Theory Workshop, pp. 432–436, Sevilla, Spain, Sep. 2013.

[Schlüter & Scharrenbach+ 05] R. Schlüter, T. Scharrenbach, V. Steinbiss, H. Ney: Bayes Risk Minimization using Metric Loss Functions Interspeech, pages 1449-1452, Lisboa, Portugal, Sep. 2005.

[Schmidhuber 14] Deep Learning in Neural Networks: An Overview. 88 pages with 53 pages of references, arXiv:1404.7828v4, 08-Oct-2014.

[Schuster & Paliwal 97] M. Schuster, K. K. Paliwal: Bidirectional Recurrent Neural Networks. IEEE Trans. on Signal Processing, Vol. 45, No. 11, pp. 2673-2681, Nov. 1997.

[Schwenk 07] H. Schwenk: Continuous space language models. Computer Speech and Language, Vol. 21, No. 3, pp. 492–518, July 2007.

[Schwenk 12] H. Schwenk: Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. 24th Int. Conf. on Computational Linguistics (COLING), Mumbai, India, pp. 1071–1080, Dec. 2012.

[Schwenk & Costa-jussa+ 07] H. Schwenk , M. R. Costa-jussa, J. A. R. Fonollosa: Smooth bilingual n-gram translation. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 430–438, Prague, June 2007.

[Schwenk & Déchelotte+ 06] H. Schwenk, D. Déchelotte, J. L. Gauvain: Continuous Space Language Models for Statistical Machine Translation. COLING/ACL 2006, pp. 723–730, Sydney, Australia July 2006.

[Schwenk & Gauvain 02] H. Schwenk, J.-L. Gauvain: Connectionist language modeling for large vocabulary continuous speech recognition. pp. 765-768, ICASSP 2002.

[Seide & Li+ 11] F. Seide, G. Li, D. Yu: Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. Interspeech, pp. 437-440, Florence, Italy, Aug. 2011.

[Solla & Levin+ 88] S. A. Solla, E. Levin, M. Fleisher: Accelerated Learning in Layered Neural Networks. Complex Systems, Vol.2, pp. 625-639, 1988.

[Soltan & Ananthakrishnan+ 22] S. Soltan, S. Ananthakrishnan, J. FitzGerald et al.: AlexaTM 20B: Few-Shot Learning Using a Large-Scale Multilingual Seq2seq Model. Amazon, arxiv, 03-Aug-2022.

[Stolcke & Grezl+ 06] A. Stolcke, F. Grezl, M.-Y. Hwang, X. Lei, N. Morgan, D. Vergyri: Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Toulouse, France, May 2006.

[Sundermeyer & Alkhouli+ 14] M. Sundermeyer, T. Alkhouli, J. Wuebker, H. Ney: Translation Modeling with Bidirectional Recurrent Neural Networks. Conf. on Empirical Methods in Natural Language Processing (EMNLP), pp. 14–25, Doha, Qatar, Oct. 2014.

[Sundermeyer & Ney+ 15] M. Sundermeyer, H. Ney, R. Schlüter: From feedforward to recurrent LSTM neural networks for language modeling. IEEE/ACM Trans. on Audio, Speech, and Language Processing, Vol. 23, No. 3, pp. 13–25, March 2015.

[Sundermeyer & Schlüter+ 12] M. Sundermeyer, R. Schlüter, H. Ney: LSTM neural networks for language modeling. Interspeech, pp. 194–197, Portland, OR, USA, Sep. 2012.

[Sutskever & Vinyals+ 14] I. Sutskever, O. Vinyals, Q. V. Le: Sequence to Sequence Learning with Neural Networks. Google, arxiv, 14-Dec-2014.

[Tikhonov & Arsenin 77] A. N. Tikhonov, V. Y. Arsenin: Solutions of Ill-Posed Problems. Washington DC, Winston 1977.

[Tüske & Plahl+ 11] Z. Tüske, C. Plahl, R. Schlüter: A study on speaker normalized MLP features in LVCSR. Interspeech, pp. 1089-1092, Florence, Italy, Aug. 2011.

[Tüske & Golik+ 14] Z. Tüske, P. Golik, R. Schlúter, H. Ney: Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR. Interspeech, ISCA best student paper award, pp. 890-894, Singapore, Sep. 2014.

[Utgoff & Stracuzzi 02] P. E. Utgoff, D. J. Stracuzzi: Many-layered learning. Neural Computation, Vol. 14, No. 10, pp. 2497-2539, Oct. 2002.

[Valente & Vepa+ 07] F. Valente, J. Vepa, C. Plahl, C. Gollan, H. Hermansky, R. Schlüter: Hierarchical Neural Networks Feature Extraction for LVCSR system. Interspeech, pp. 42-45, Antwerp, Belgium, Aug. 2007.

[Vapnik 98] Vapnik: Statistical Learning Theory. Addison-Wesley, 1998.

[Variani & Sainath+ 16] E. Variani, T. N. Sainath, I. Shafran, M. Bacchiani: Complex Linear Projection (CLP): A Discriminative Approach to Joint Feature Extraction and Acoustic Modeling. Interspeech 2016, San Francisco, CA, pp. 808-812, Sep. 2016.

[Vaswani & Shazeer+ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser: Attention Is All You Need. Google, arxiv, 06-Dec-2017.

[Vaswani & Zhao+ 13] A. Vaswani, Y. Zhao, V. Fossum, D. Chiang: Decoding with Large-Scale Neural Language Models Improves Translation. Conf. on Empirical Methods in Natural Language Processing (EMNLP, pp. 1387–1392, Seattle, Washington, USA, Oct. 2013.

[Velichko & Zagoruyko 70] V. M. Velichko, N. G. Zagoruyko: Automatic Recognition of 200 Words. Int. Journal Man-Machine Studies, Vol. 2, pp. 223-234, June 1970.

[Vintsyuk 68] T. K. Vintsyuk: Speech Discrimination by Dynamic Programming. Kibernetika (Cybernetics), Vol. 4, No. 1, pp. 81-88, Jan.-Feb. 1968.

[Vintsyuk 71] T. K. Vintsyuk: Elementwise Recognition of Continuous Speech Composed of Words from a Specified Dictionary. Kibernetika (Cybernetics), Vol. 7, pp. 133-143, March-April 1971.

[Vogel & Ney+ 96] S. Vogel, H. Ney, C. Tillmann: HMM-based word alignment in statistical translation. Int. Conf. on Computational Linguistics (COLING), pp. 836-841, Copenhagen, Denmark, Aug. 1996.

[Waibel & Hanazawa+ 88] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. L. Lang: Phoneme Recognition: Neural Networks vs. Hidden Markov Models. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), New York, NY, pp.107-110, April 1988.

[Wang & Alkhouli+ 17] W. Wang, T. Alkhouli, D. Zhu, H. Ney: Hybrid Neural Network Alignment and Lexicon Model in Direct HMM for Statistical Machine Translation. Annual Meeting ACL, pp. 125-131, Vancouver, Canada, Aug. 2017.

[Wang & Zhu+ 18] W. Wang, D. Zhu, T. Alkhouli, Z. Gan, H. Ney: Neural Hidden Markov Model for Machine Translation. Annual Meeting ACL, Melbourne, Australia, July 2018.

[Xu & Povey+ 10] H. Xu, D. Povey, L. Mangu, J. Zhu: Minimum Bayes Risk Decoding and System Combination Based on a Recursion for Edit Distance. Computer Speech and Language, Sep. 2010.

[Zens & Och+ 02] R. Zens, F. J. Och, H. Ney: Phrase-Based Statistical Machine Translation. 25th Annual German Conf. on AI, pp. 18–32, LNAI, Springer 2002.

[Zhou & Berger+ 2021] W. Zhou, S. Berger, R. Schlüter, H. Ney: Phoneme Based Neural Transducer for Large Vocabulary Speech Recognition. ICASSP, Toronto, June 2021.

[Zhou & 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.

[Zhou & Zeyer+ 2021] W. Zhou, A. Zeyer, A. Merboldt, R .Schlüter, H. Ney: Equivalence of Segmental and Neural Transducer Modeling: A Proof of Concept. Interspeech, pp. 2891-2895, Graz, 2021.

END

Master Course: Advanced ML
UP Valencia, April 2024