Neural Network Language Models

Steve Renals

Automatic Speech Recognition— ASR Lecture 12
2 March 2015

Neural networks for speech recognition

- Introduction to Neural Networks
- Training feed-forward networks
- Hybrid neural network / HM M acoustic models
- Neural network features – Tandem, posteriorgrams
- Deep neural network acoustic models
- Neural network language models

Neural networks for speech recognition

- Introduction to Neural Networks
- Training feed-forward networks
- Hybrid neural network / HMM acoustic models
- Neural network features – Tandem, posteriorgrams
- Deep neural network acoustic models
- **Neural network language models**

n-gram language modelling

- The problem: estimate the probability of a sequence of T words, $P(w_1, w_2, \dots, w_T) = P(w_1^T)$
- Decompose as conditional probabilities

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t \mid w_1^{t-1})$$

- n-gram approximation: only consider $(n-1)$ words of context:

$$P(w_t \mid w_1^{t-1}) \sim P(w_t \mid w_{t-(n-1)}^{t-1})$$

- Many possible word sequences — consider vocab size $|V| = 100\,000$ with a 4-gram
    - $100\,000^4$ possible 4-grams, i.e. 10^{20} parameters
- Most n-grams not in training data — zero-probability problem
- Smooth n-gram model with models with smaller context size (interpolation)
- State of the art — modified Kneser-Ney smoothing

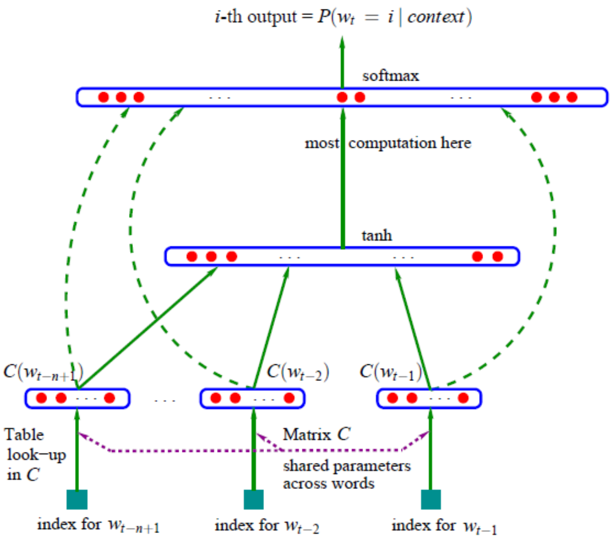
Problems with n-grams

1. *Curse of dimensionality* — model size (number of parameters) increases exponentially with context size
2. Probability estimation in a high-dimensional discrete smooth — not smooth, small changes in discrete context may result in large changes in probability estimate
3. Does not take word similarity into account

Distributed representation for language modelling

- Each word is associated with a learned *distributed representation* (feature vector)
- Use a neural network to estimate the conditional probability of the next word given the the distributed representations of the context words
- Learn the distributed representations and the weights of the conditional probability estimate jointly by maximising the log likelihood of the training data
- Similar words (distributionally) will have similar feature vectors — small change in feature vector will result in small change in probability estimate (since the NN is a smooth function)

Neural Probabilistic Language Model



Bengio et al (2006)

Neural Probabilistic Language Model

- Train using stochastic gradient ascent to maximise log likelihood
- Number of free parameters (weights) scales
  - Linearly with vocabulary size
  - Linearly with context size
- Can be (linearly) interpolated with n-gram model
- Perplexity results on AP News (14M words training). $|V| = 18k$

model	**n**	**perplexity**
NPLM(100,60)	6	109
n-gram (KN)	3	127
n-gram (KN)	4	119
n-gram (KN)	5	117

NPLM — Shortlists

- Majority of the weights (hence majority of the computation) is in the output layer
- Reduce computation by only including the s most frequent words at the output — the *shortlist* (S) (full vocabulary still used for context)
- Use an n-gram model to estimate probabilities of words not in the shortlist
- Neural network thus redistributes probability for the words in the shortlist

$$P_S(h_t) = \sum_{w \in S} P(w|h_t)$$

$$P(w_t|h_t) = \begin{cases} P_{NN}(w_t|h_t) P_S(h_t) & \text{if} w_t \in S \\ P_{KN}(w_t|h_t) & \text{else} \end{cases}$$

- In a $|V| = 50k$ task a 1024 word shortlist covers 89% of 4-grams, 4096 words covers 97%

NPLM — ASR results

Speech recognition results on Switchboard

7M / 12M / 27M words in domain data.

500M words background data (broadcast news)

Vocab size $|V| = 51k$, Shortlist size $|S| = 12k$

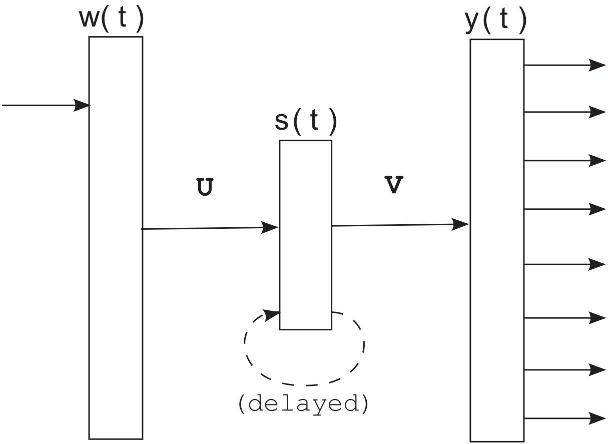
WER/%

in-domain words	7M	12M	27M
KN (in-domain)	25.3	23.0	20.0
NN (in-domain)	24.5	22.2	19.1
KN (+b/g)	24.1	22.3	19.3
NN (+b/g)	23.7	21.8	18.9

Recurrent Neural Network (RNN) LM

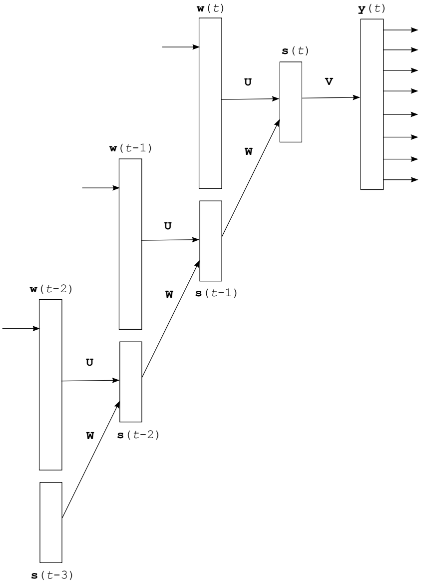
- Rather than fixed input context, *recurrently connected* hidden units provide memory
- Model learns "how to remember" from the data
- Recurrent hidden layer allows clustering of variable length histories

RNN LM

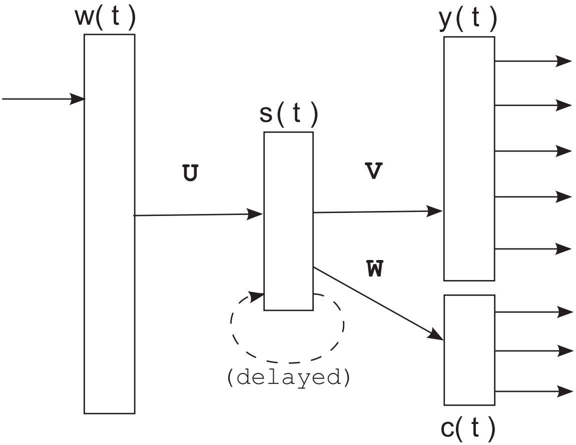


Mikolov (2011)

RNN training: back-propagation through time



Factorised RNN LM



Perplexity Results

Table 2. *Comparison of different neural network architectures on Penn Corpus (1M words) and Switchboard (4M words).*

	Penn Corpus		Switchboard	
Model	NN	NN+KN	NN	NN+KN
KN5 (baseline)	-	141	-	92.9
feedforward NN	141	118	85.1	77.5
RNN trained by BP	137	113	81.3	75.4
RNN trained by BPTT	123	106	77.5	72.5

Reading

- Y Bengio et al (2006), Neural probabilistic language models (sections 6.1, 6.2, 6.3, 6.7, 6.8), Studies in Fuzziness and Soft Computing Volume 194, Springer, chapter 6.
- T Mikolov et al (2011), Extensions of recurrent neural network language model, Proc IEEE ICASSP–2011