MPhil in Machine Learning and Machine Intelligence

Module MLMI2: Speech Recognition

L15: Neural Network Language Models

Phil Woodland
pcw@eng.cam.ac.uk

Michaelmas 2021

Cambridge University Engineering Department

Introduction

This lecture discusses the use of **Neural Network Language Models** in speech recognition.

Such language models can also have a range of other applications in natural language processing & understanding etc.

In particular we will discuss:

- Feed-forward Neural Network Language Models
- Recurrent Neural Network Language Models
- Efficient application during Recognition
- Advanced Training Criteria
- Increased context: utterance and conversation level
- Transformer language models
- Pre-trained transformer language models

Note that NNLMs can be used with a **hybrid** approach and are sometimes also incorporated into **end-to-end** neural network speech recognition approaches.

Motivation

As with N-gram LMs for speech recognition, the purpose is to assign probabilities to different word sequences.

The overall aim is to improve on N-gram models which treat all words as separate entities modelled in a **discrete** space.

Here the aim is to

- Model **words and word sequences** in a **continuous space** i.e. **embeddings**
- Better capture **relationships between words**
- Improve data usage (better performance for the same amount of training data)
- Improve **generalisation** to new word sequences
- Allow **longer-term modelling** of context than N-grams

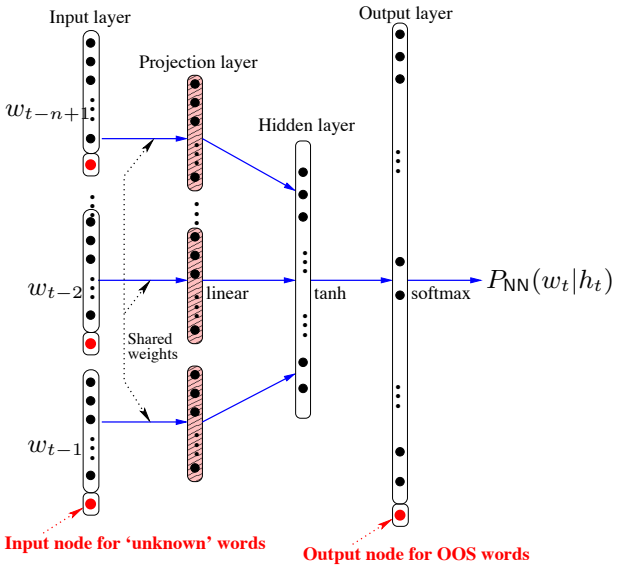
Issues that will need to be addressed include

1. Application in Decoding.
   - If a **word-prediction probability** can be computed from the usual sentence decomposition, NNLM can be used in standard search
   - If whole sentence probability cannot be computed left-to-right, then apply in N-best rescoring
2. Training. Need efficient GPU training to be able to train on large corpora

In both cases, LM probabilities need to be computed efficiently.

Feedforward Neural Network Language Models

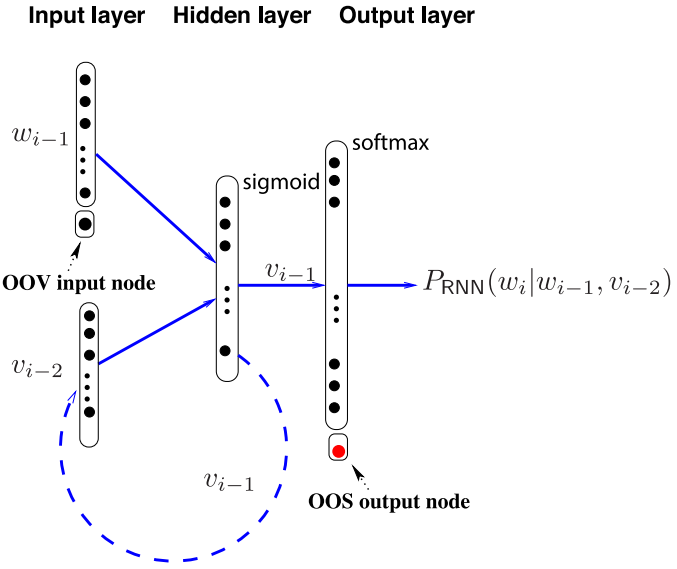
- Feed-forward neural network structure
- Continuous form of N-gram model
- Inputs to the network are $n-1$ history words in input vocabulary V_{in}
- Between input and output layers, two hidden layers for projection & non-linear probability estimation
- Input uses 1-of-K coding, projection layer learns an embedding
- Output uses a softmax over the output vocabulary: predicts next word
- Have a separate input for unknown-words



- Model a "shortlist" of output vocab
  - Reduce cost of computing output layer and soft-max
- Can interpolate with an N-gram - possibly trained on a much larger corpus
  - Can use N-gram for out-of-shortlist words
- Can use in conventional lattice rescoring (since N-gram history context needed)

Recurrent Neural Network Language Models

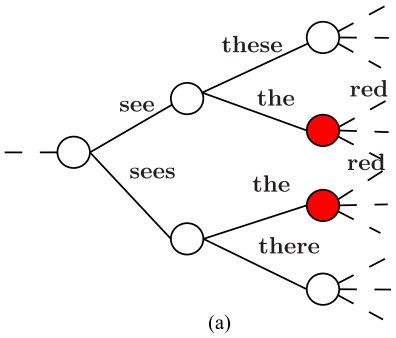
- Input uses 1-of-K coding
- Feed in word-by-word
- Learns an embedding
- Recurrent links on hidden layer
- Output uses a softmax over the output vocabulary
- Have a separate input for unknown-words
- Model a "shortlist" of output vocab
- Trained by back-propagation through time



- Can have a separate class and word outputs
  - Compute $P(c_i|\boldsymbol{v}_{i-1})$ and $P(w_i|c_i, \boldsymbol{v}_{i-1})$ — multiply together to get $P(w_i|\boldsymbol{v}_{i-1})$
  - Each word belongs to only one class
  - Separate output layers for each word class reduce cost of output layer computations
  - however would like to avoid this approximation (& better for use with GPUs)

Applying Recurrent Neural Network Language Models

- Similar to FFNNLMs, combine with an N-gram possibly built on much larger corpus
- **Non-Markovian** nature of RNNLMs mean that apply on complete utterance hypotheses
  - Use N-best rescoring
  - Can tree-structure N-best list into a **prefix-tree** for efficient application
  - Note that items in red are not merged as they would be in a lattice



- Better if can approximate to use lattice rescoring
  - Need to merge/cluster histories so that scales linearly with utterance length
  - N-gram approximation for histories: merge if differences beyond previous $n-1$
  - Directly cluster history vectors based on a distance measure
- Approximate lattice rescoring (or even direct decoding using N-gram approx.) widely used
- RNNLMs provide **significant reductions** in perplexity and WER over std N-grams and FF-NNLMs
- Can extend to LSTM-based language models which improve performance further

Advanced Training Techniques for RNNLMs

- Recognition RNNLM computation dominated by the large output layer and softmax normalisation term (requires all outputs to be calculated - even if not needed for search)
- Only need to compute normalisation term if significant **variance** between calculations / histories
- RNNLMs trained to optimise cross-entropy
  (i.e. maximum likelihood of output sequence given input)
- Can add a **variance regularisation** term in training
  - Add extra term to loss function to minimise variance of softmax denominator
- Still have high computation in training
- Alternative is to use **Noise Contrastive Estimation**
  - Include a small number (e.g. 10) of noise words during training of each predicted word sampled differently from a unigram distribution for each occurrence
  - Model becomes self-normalising during both training and test
  - Allows models to be trained on GPUs on large datasets e.g. a billion word tokens (in 2015!)

Adapting RNNLM models

One advantage of NNLMs (& NN models in general) is that can input data in a flexible manner.

For acoustic NN models, various adaptation approaches are possible including

- **augmenting the input** with a speaker-dependent vector (e.g. i-vector)

For NNLM adaptation, similar approaches can be used.

- Augment the input with a **topic or genre dependent vector** estimated over the "document"
- Use the same topic vector for all utterances in a current "document" / story etc.
- For ASR this implies that the topic vector is estimated from a first pass decoding or other side-information

For instance, for the MGB challenge with a wide range of programme types and styles

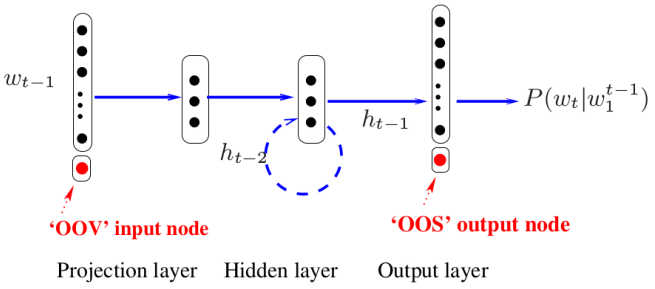
- RNNLM topic adaptation using Latent Dirichlet allocation (LDA)
- 30 dimensional episode level topic posterior vectors that augmented the input
- Used in both training and decoding
- Estimation of vectors robust to ASR errors in first pass decoding

Increasing LM context

There are multiple ways to increase LM context. First consider normal within-utterance case.

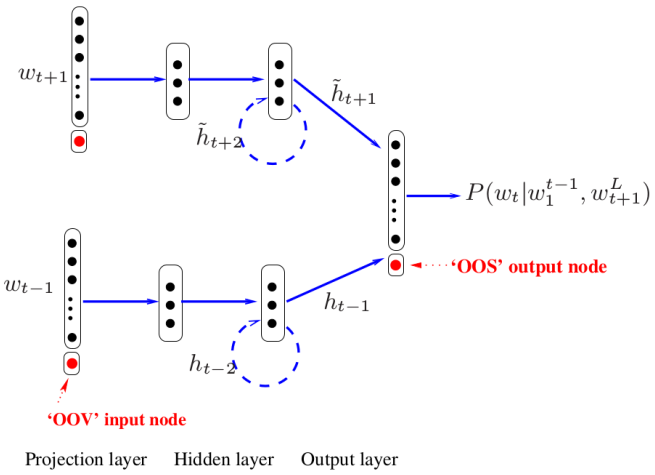
For the uni-directional RNNLMs considered previously

- No future context
- Can apply (approx) in lattice rescoring



If use a bi-directional RNNLM

- get access to future context
- not standard L-R sentence probability decomposition
- need extra normalisation term
- N-best (not lattice) rescoring
- improved error rates



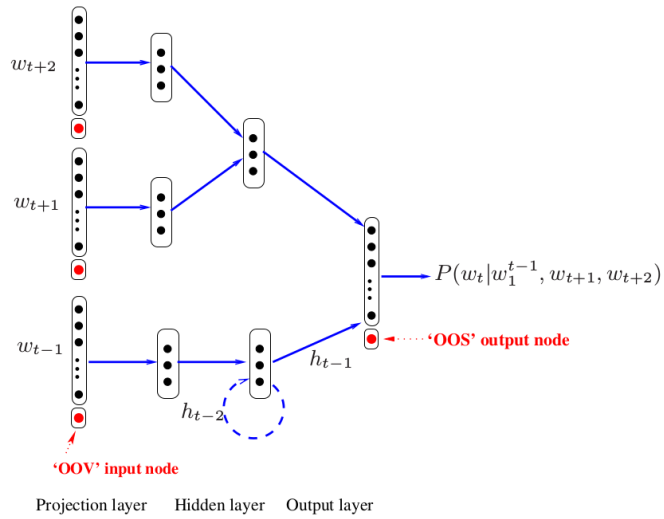
Increased context: Succeeding Word RNNLM

Having access to future context for the current utterance in a bi-directional RNNLM

- reduced the word error rate in rescoring
- **but** cannot apply to lattice rescoring.

Succeeding Word RNNLM

- Uses a **fixed future context**.
- Two words in future used in figure
- Most of WER reduction of bi-directional RNNLMs in N-best rescoring
- Can apply (approx) in lattice rescoring

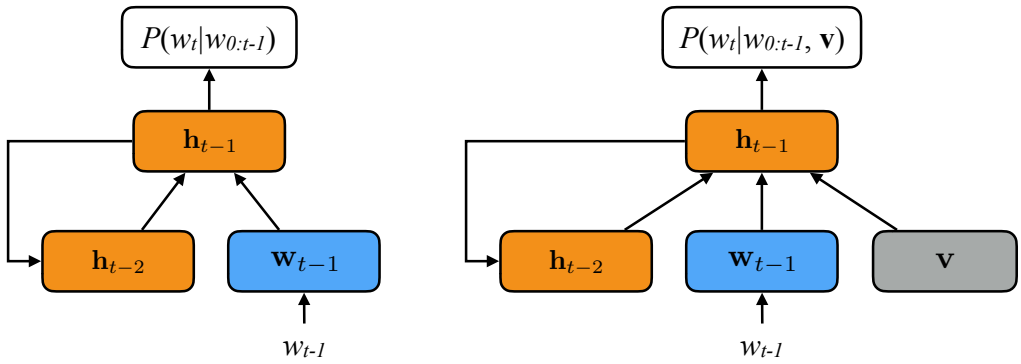


Conversational LM context

Simplest way to include information from previous utterances is to use the final RNN hidden state from the preceding utterance to initialise the RNN hidden state of the current utterance.

More effective is use of a cross-utterance vector $\boldsymbol{v}$ using the same broad idea as used in RNNLM topic adaptation

- vector $\boldsymbol{v}$ may come from RNN-based encoding of previous and future utterances.
- an **attention mechanism** can be applied to extract a better context representation.

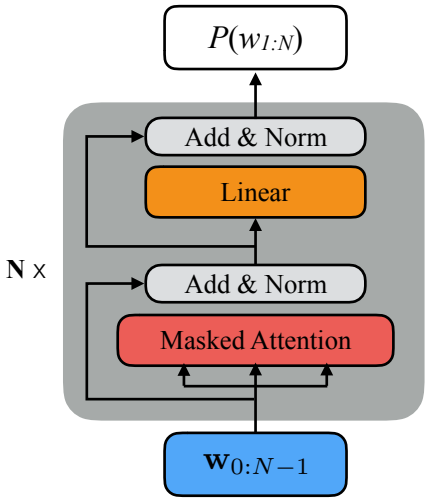


Above the left-hand figure is a standard RNNLM, and the right-hand figure has an additional input $\boldsymbol{v}$ representing the context.

Transformer Language Models

The transformer structure (covered in 4F10) has been applied to language modelling, both for language models for ASR and for **language model pre-training**.

- For language models, normally the **decoder** part of a complete transformer model is used (see figure)
- No recurrent connections in standard Transformer
- N represents the number of Transformer layer blocks.
- Input is the word embedding sequence with added positional encoding
- Output is the prediction of each word given the previous word history
- A **lower-triangular mask** is used for the attention so information from current and future ground-truth words will not flow into the probability prediction of the current word.
- Transformer based language models currently give the lowest perplexity / error rates



Transformer structures can also be used in cross-utterance language models where there are dependencies from the previous utterance used.

Language Model Pre-Training

Various types of structures, typically trained on very large text corpora.

Many of these have been based on transformer architecture because

- transformers scale well on GPUs since no recurrent connections
- very large models on huge amount of text data possible

Two main families:

1. GPT type. Auto-regressive prediction e.g. GPT, GPT2 , GPT3, XLNet
   - Transformer decoder with lower-triangular mask as the language model
   - Standard language model auto-regressive decomposition of sentence probability
   - Std PPL is valid for the full sequence
   - Scales to very large models (GPT 3 has 175 billion parameters!)
2. Partial sequence prediction: e.g. BERT, RoBERTa and other variations of BERT etc.
   - Masking a number of words in sequence: use the masked sequence as input
   - Model predicts masked words given the other un-masked words
   - Prediction is simultaneous: doesn't use auto-regressive decomposition
   - Information from past and future can be used
   - Perplexity can only be measured for the predicted partial sequence

These models can typically be fine-tuned on a some task data and can perform very well by **transfer learning** representations from a large amount of data.

- GPT style models can be used directly for ASR rescoring - work well especially when fine-tuned and can be combined with other types of neural network LM

Example Recognition Results

Recognition results for LMs trained from scratch & fine tuning of pre-trained models

- LMs trained/fine-tuned on Switchboard + Fisher training data (27MW)
- WERs given on Switchboard/Call Home eval 2000 set
- Acoustic models trained on Switchboard 300h training data (TDNN LF-MMI)
- WER given when rescoring 100-best lists

From scratch models:

- FF: 768 units, 2 layers. 256 embedding
- LSTM 2048 units, 2 layers. 256 embed
- Transformer 24 decoder blocks, each block 768 units. 256 embedding
- word level tokens
- $\oplus$ denotes weighted log prob combination

Model	Swbd	CH
4-gram	8.6	17.0
FNN LM	7.9	15.8
LSTM LM	6.7	13.7
Transformer LM	6.6	13.7
F $\oplus$ L $\oplus$ T NNLM	6.5	13.5

Fine tuning pre-trained models

- GPT: 12 blocks
- GPT-2 24 blocks, 10$\times$ pre-training data
- word piece models
- increased computation (esp GPT-2)

Model	Swbd	CH
GPT	6.3	13.1
GPT-2	6.2	12.8
GPT $\oplus$ GPT-2	6.1	12.7

Summary

Described different types of **neural network language model**

- feedforward N-gram type models
- recurrent models (non-Markovian)
- For non-Markovian models in decoding use N-best rescoring or approx. lattice search
  - Can use N-gram approximation in lattices
- Efficient decoding/training avoid softmax denominator e.g. via variance regularisation or noise contrastive training
- NNLM adaptation by additional input vector
- Increasing LM context
  - bidirectional models
  - succeeding word models allow lattice rescoring
  - cross-utterance models condition of other parts of a conversation
- Transformer LMs
  - current state of the art with fairly large data sets
  - language model pre-training on huge data sets, task-specific fine tuning effective

The use of NNLMs of various types has made a very large difference in LM performance for ASR, with significant WER reductions.

NNLM References (partial list)

- Y. Bengio, R. Ducharme, P. Vincent, & C. Jauvin, “A neural probabilistic language model”,Journal of Machine Learning Research, vol. 3, pp. 1137-1155, 2003.
- X. Chen, T. Tan, X. Liu, P. Lanchantin, M. Wan, M.J.F. Gales & P.C. Woodland, “Recurrent neural network language model adaptation for multi-genre broadcast speech recognition”, Proc. Interspeech, 2015
- X. Chen, X. Liu, Y. Wang, M. Gales & P. Woodland, “Efficient training and evaluation of recurrent neural network language models for automatic speech recognition,” IEEE/ACM Trans. ASLP, 2016
- X. Chen, X. Liu, Y. Wang, A. Ragni, J. Wong & M. Gales, “Exploiting Future Word Contexts in Neural Network Language Models for Speech Recognition”, IEEE/ACM Trans ASLP, 2019
- X. Liu, X. Chen, Y. Wang, M. Gales, & P. Woodland, “Two efficient lattice rescoring methods using recurrent neural network language models,” IEEE/ACM Trans. ASLP, vol. 24, pp. 1438-1449, 2016.
- T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, & S. Khudanpur, “Recurrent neural network based language model”,Proc.Interspeech, 2010.
- J. Park, X. Liu, M. Gales & P.C, Woodland, “Improved neural network based language modelling and adaptation”, Proc. Interspeech, 2010
- H. Schwenk, “Continuous space language models”, Computer Speech & Language, vol. 21, no. 3, pp. 492–518, 2007.
- G. Sun, C. Zhang & P. C. Woodland, “Cross-Utterance Language Models with Acoustic Error Sampling”, arXiv:2009.01008, 2020
- G. Sun, C. Zhang & P. C. Woodland “Transformer Language Models with LSTM-based Cross-utterance Information Representation”, Proc. ICASSP, 2021.
- X. Zheng, C. Zhang & P.C. Woodland “Adapting GPT, GPT-2 and BERT Language Models for Speech Recognition”, Proc. ASRU, 2021.