Natural Language Processing: Part II Overview of Natural Language Processing (L90): ACS Lecture 9

Ann Copestake

Computer Laboratory
University of Cambridge

October 2017

Distributional semantics and deep learning: outline

Neural networks in pictures

word2vec

Visualization of NNs

Some general comments on deep learning for NLP

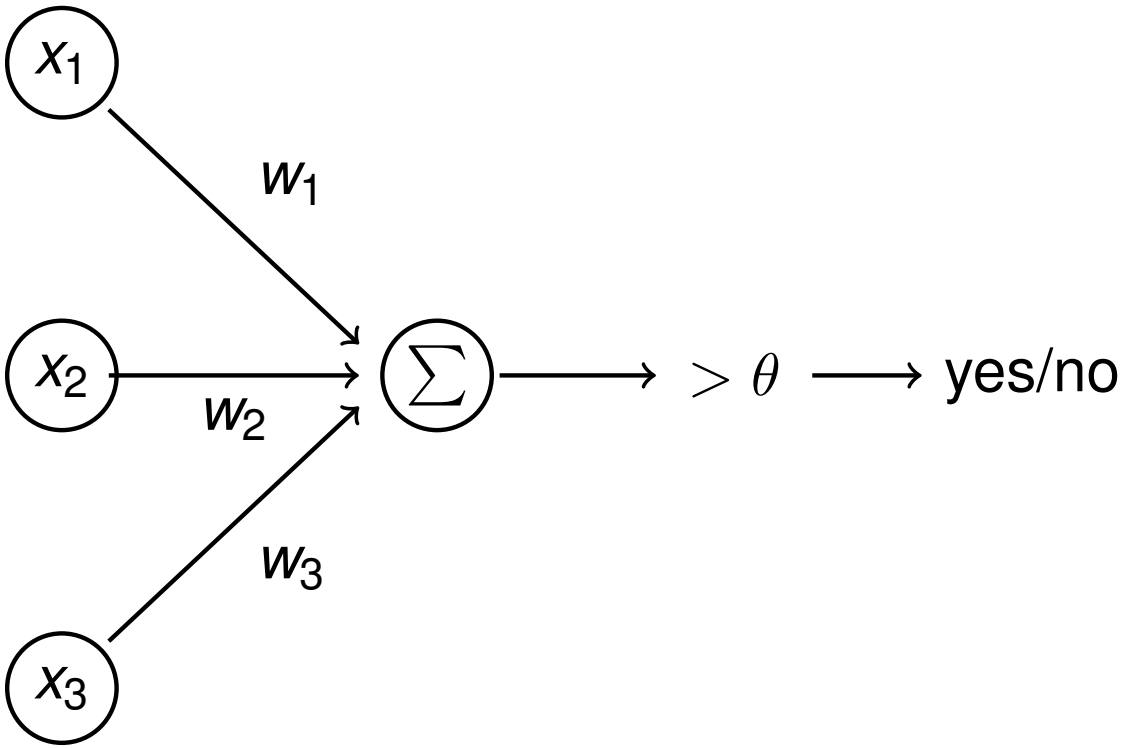
VQA has been moved to lecture 12 (insufficient time today)

Some slides adapted from Aurelie Herbelot.

Outline.

Perceptron

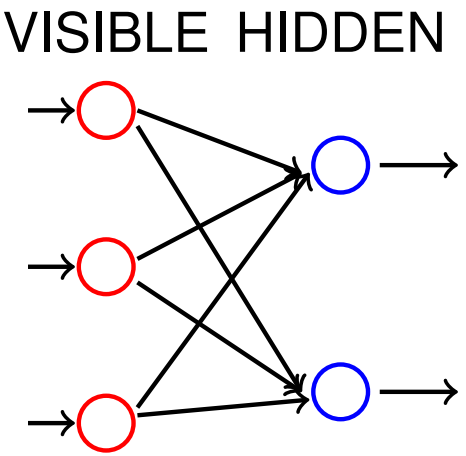
- Early model (1962): no hidden layers, just a linear classifier, summation output.



Dot product of an input vector $\vec{x}$ and a weight vector $\vec{w}$, compared to a threshold θ

Restricted Boltzmann Machines

- Boltzmann machine: hidden layer, arbitrary interconnections between units. Not effectively trainable.
- Restricted Boltzmann Machine (RBM): one input and one hidden layer, no intra-layer links.

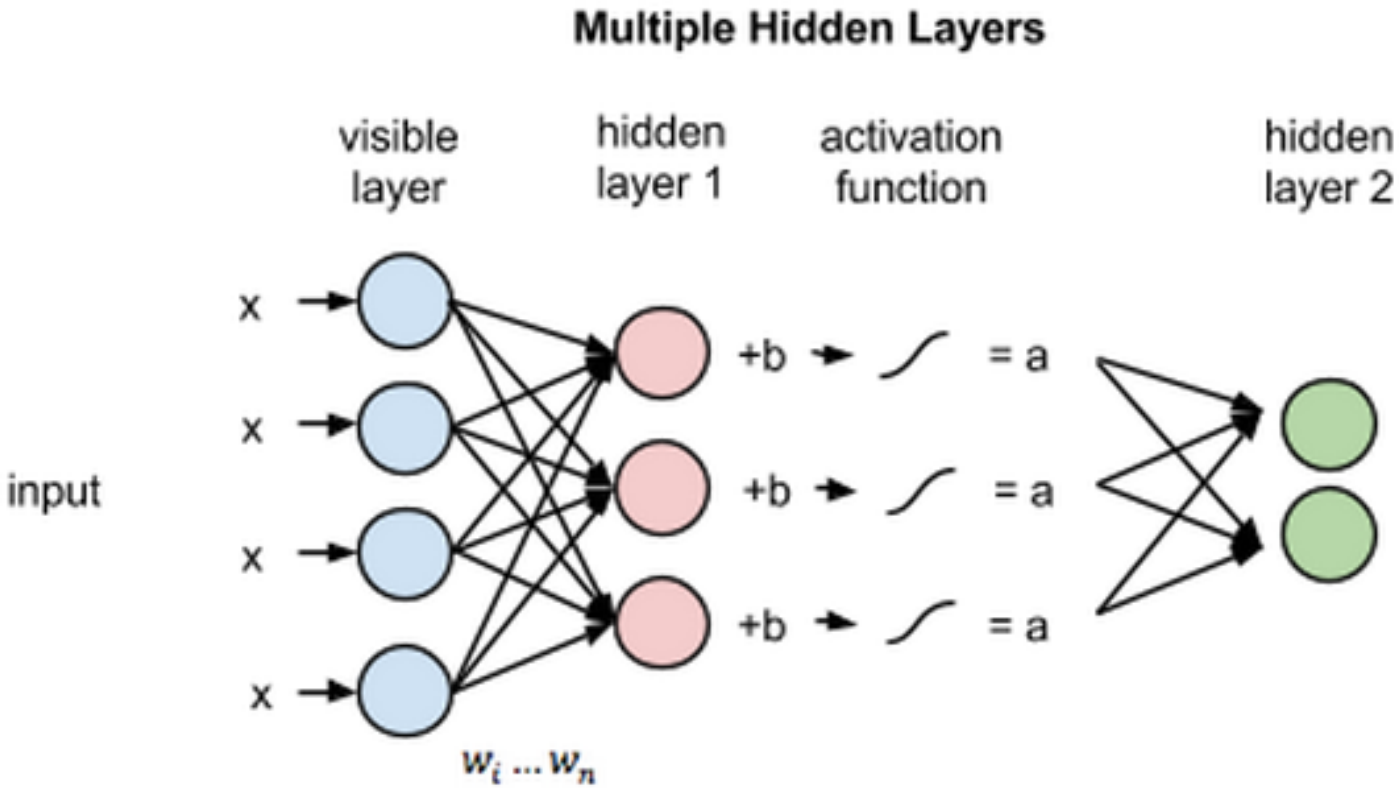


$w_1, \ldots w_6$ b (bias)

Restricted Boltzmann Machines

- Hidden layer (note one hidden layer can model arbitrary function, but not necessarily trainable).
- RBM layers allow for efficient implementation: weights can be described by a matrix, fast computation.
- One popular deep learning architecture is a combination of RBMs, so the output from one RBM is the input to the next.
- RBMs can be trained separately and then fine-tuned in combination.
- The layers allow for efficient implementations and successive approximations to concepts.

Combining RBMs: deep learning

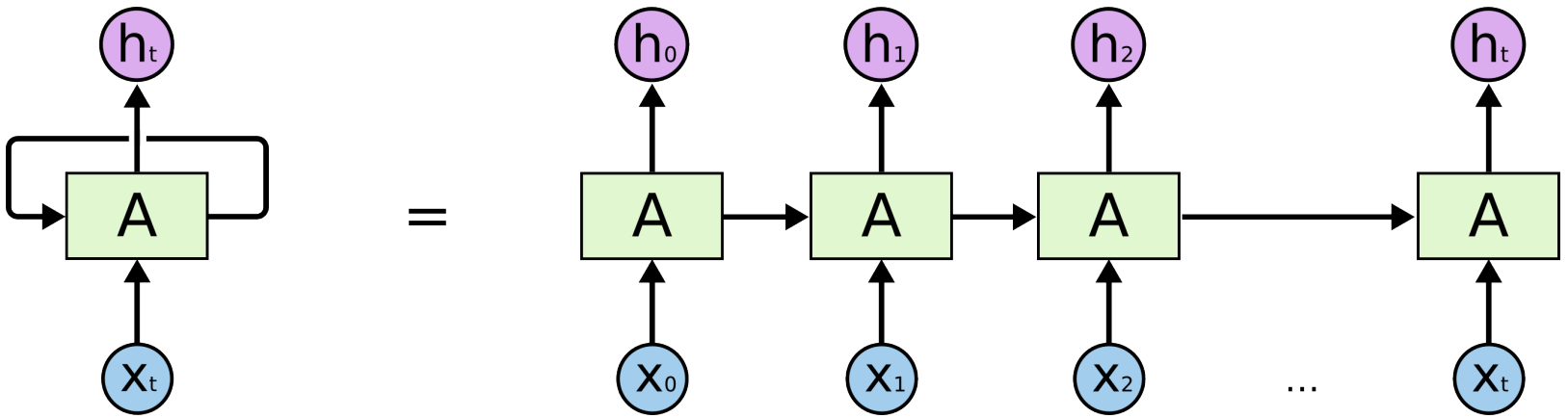


https://deeplearning4j.org/restrictedboltzmannmachine

Sequences

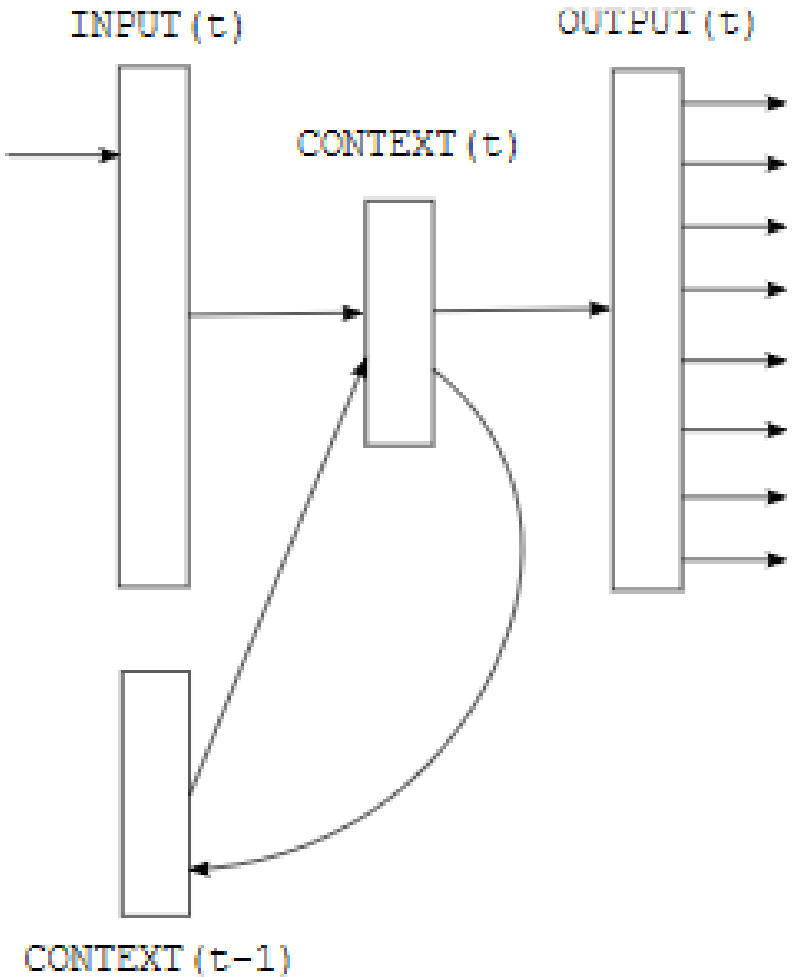
- Combined RBMs etc, cannot handle sequence information well (can pass them sequences encoded as vectors, but input vectors are fixed length).
- So different architecture needed for sequences and most language and speech problems.
- RNN: Recurrent neural network.
- Long short term memory (LSTM): development of RNN, more effective for (some?) language applications.

Recurrent Neural Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

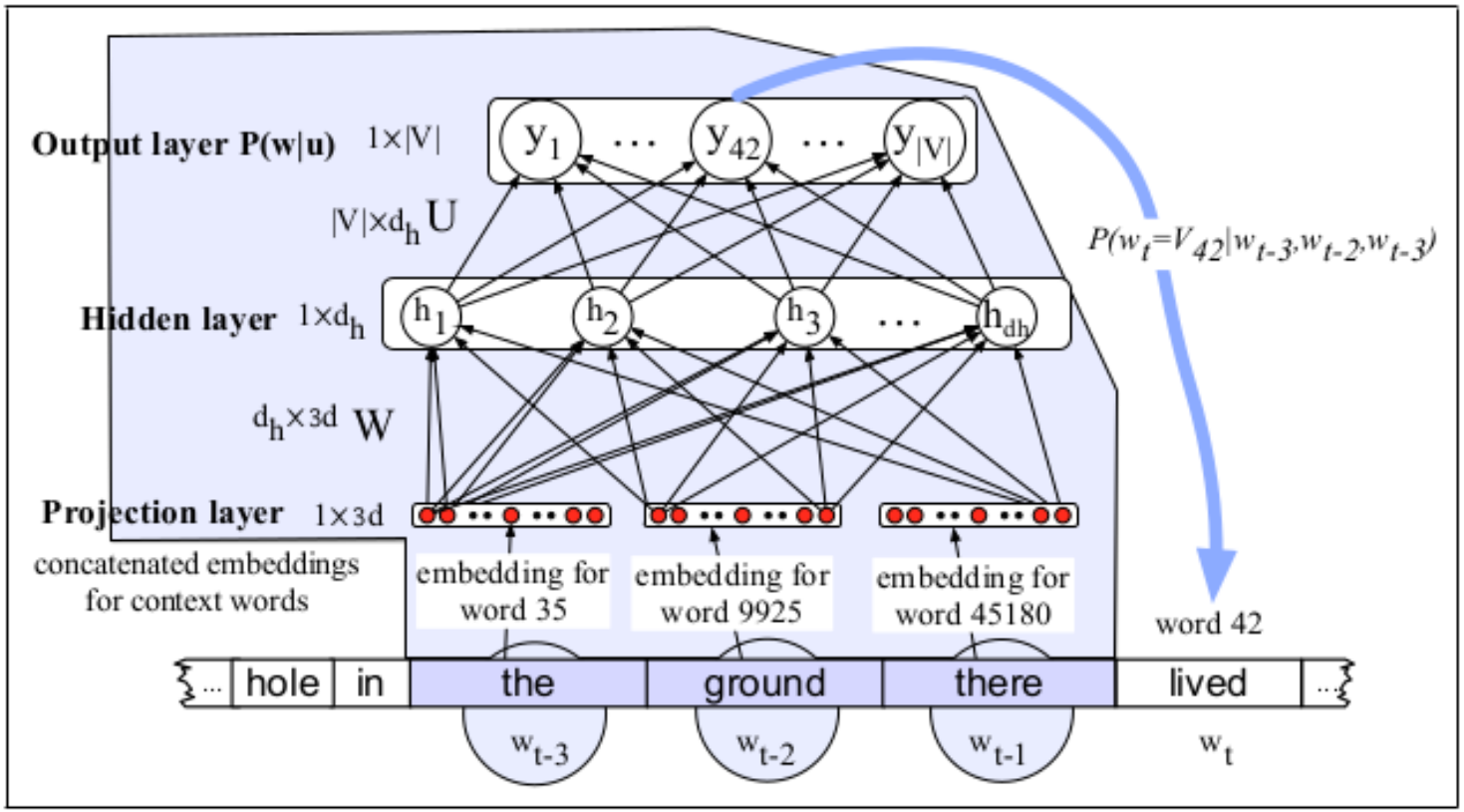
RNN language model: Mikolov et al, 2010



RNN as a language model

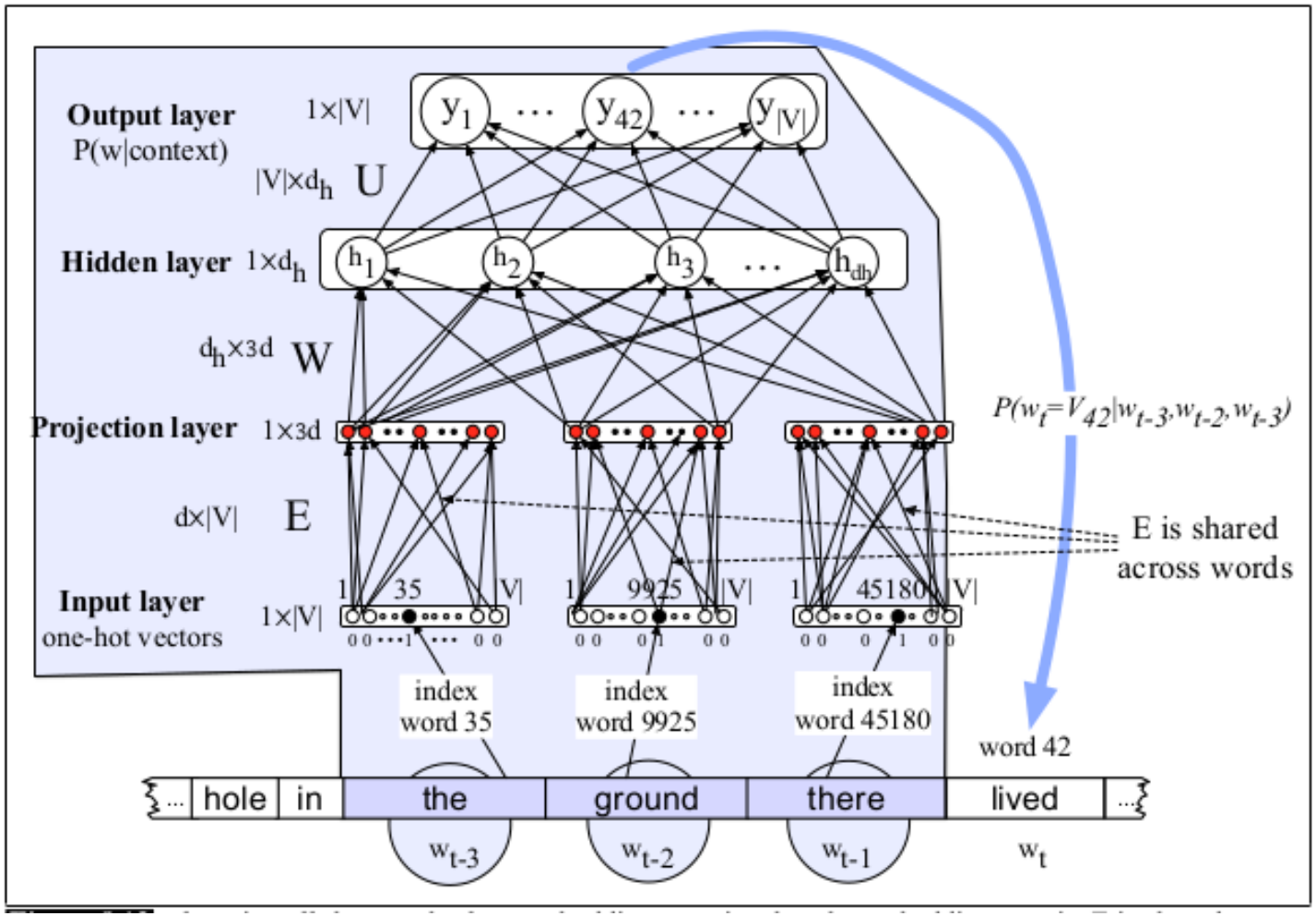
- Trained on a very large corpus to predict the next word.
- Input vector: vector for word at t concatenated to vector which is output from context layer at $t-1$.
- one-hot vector: one dimension per word (i.e., index)
- input embeddings: distributional model (with dimensionality reduction)
- embeddings: may be externally created (from another corpus) or learned for specific application.

External embeddings for prediction



Jurafsky and Martin, third edition `web.stanford.edu/~jurafsky/slp3/`

Learned embeddings for prediction



Jurafsky and Martin, third edition `web.stanford.edu/~jurafsky/slp3/`

Other neural language models

- LSTMs etc capture long-term dependencies:

  She shook her head.
  She decided she did not want any more tea, so shook her head when the waiter reappeared.
- not the same as long distance dependency in linguistics
- LSTMs now standard for speech, but lots of experimentation for other language applications.

Multimodal architectures

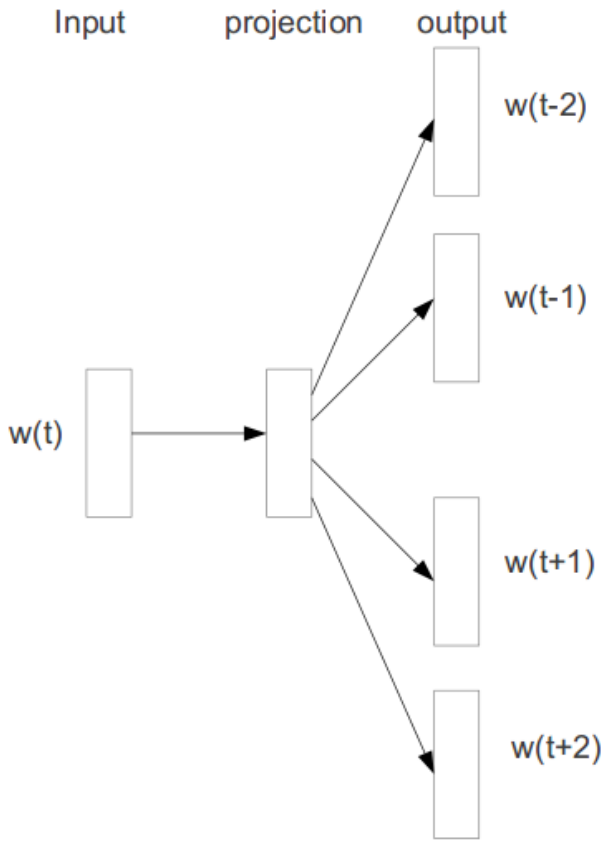
- Input to a NN is just a vector: we can combine vectors from different sources.
- e.g., features from a CNN for visual recognition concatenated with word embeddings.
- multimodal systems: captioning, visual question answering (VQA).
- Will be discussed further in lecture 12 . . .

Outline.

Embeddings

- embeddings: distributional models with dimensionality reduction, based on prediction
- word2vec: as originally described (Mikolov et al 2013), a NN model using a two-layer network (i.e., not deep!) to perform dimensionality reduction.
- two possible architectures:
  - given some context words, predict the target (CBOW)
  - given a target word, predict the contexts (Skip-gram)
- Very computationally efficient, good all-round model (good hyperparameters already selected).

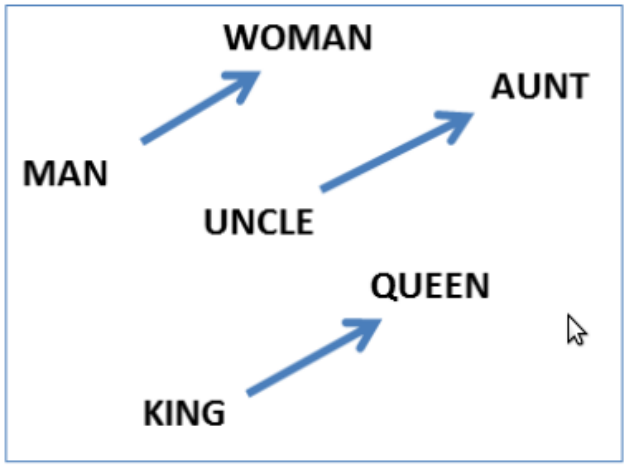
The Skip-gram model

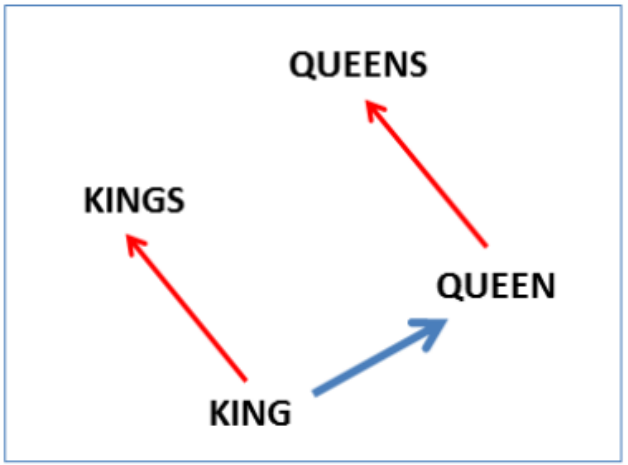


Features of word2vec representations

- A representation is learnt at the reduced dimensionality straightaway: we are outputting vectors of a chosen dimensionality (parameter of the system).
- Usually, a few hundred dimensions: dense vectors.
- The dimensions are not interpretable: it is impossible to look into ‘characteristic contexts’.
- For many tasks, word2vec (skip-gram) outperforms standard count-based vectors.
- But mainly due to the hyperparameters and these can be emulated in standard count models (see Levy et al).

What Word2Vec is famous for





BUT . . . see Levy et al and Levy and Goldberg for discussion

The actual components of word2vec

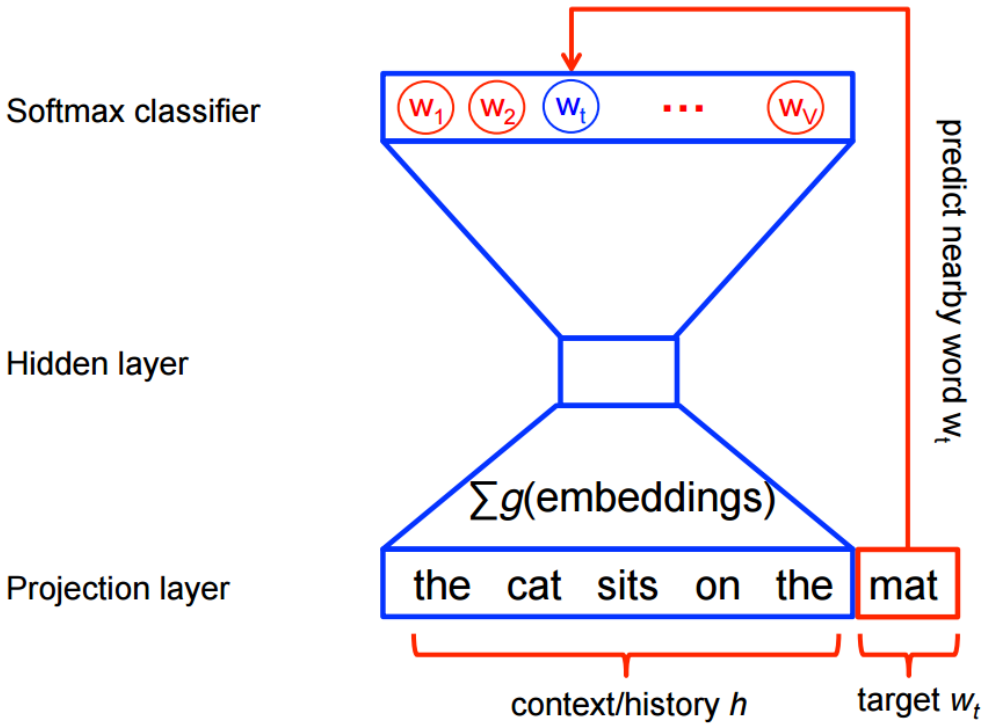
- A vocabulary. (Which words do I have in my corpus?)
- A table of word probabilities.
- Negative sampling: tell the network what *not* to predict.
- Subsampling: don't look at all words and all contexts.

Negative sampling

Instead of doing full softmax (final stage in a NN model to get probabilities, very expensive), word2vec is trained using logistic regression to discriminate between real and fake words:

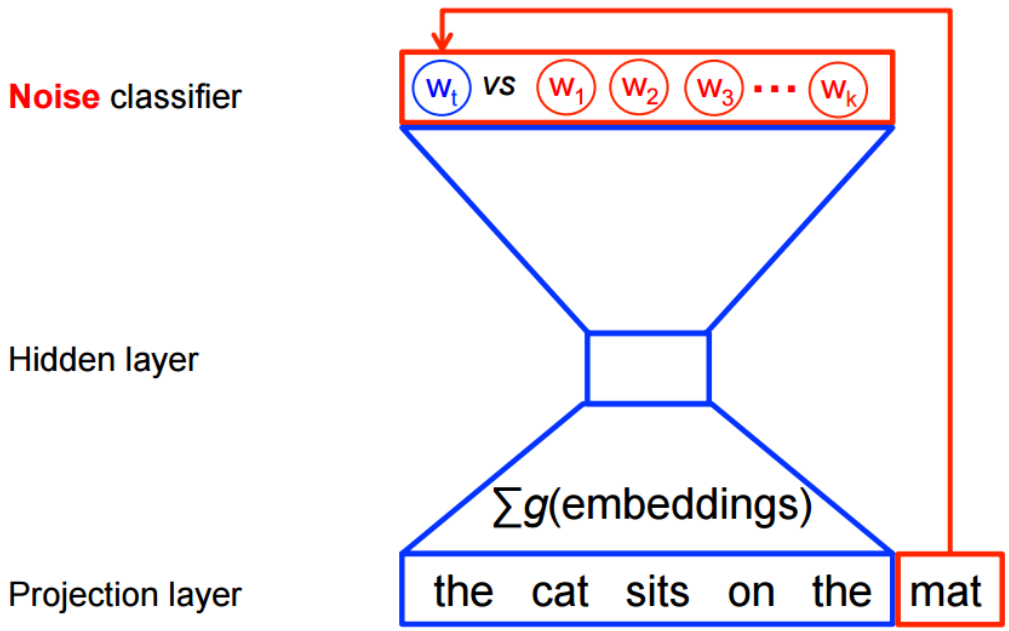
- Whenever considering a word-context pair, also give the network some contexts which are not the actual observed word.
- Sample from the vocabulary. The probability to sample something more frequent in the corpus is higher.
- The number of negative samples will affect results.

Softmax (CBOW)



https://www.tensorflow.org/versions/r0.11/tutorials/word2vec/index.html

Negative sampling (CBOW)



https://www.tensorflow.org/versions/r0.11/tutorials/word2vec/index.html

Subsampling

- Instead of considering all words in the sentence, transform it by randomly removing words from it:
  considering all sentence transform randomly words
- The subsampling function makes it more likely to remove a frequent word.
- Note that word2vec does not use a stop list.
- Note that subsampling affects the window size around the target (i.e., means word2vec window size is not fixed).
- Also: weights of elements in context window vary.

Using word2vec

- predefined vectors or create your own
- can be used as input to NN model
- many researchers use the gensim Python library
  `https://radimrehurek.com/gensim/`
- Emerson and Copestake (2016) find significantly better performance on some tests using parsed data
- Levy et al's papers are very helpful in clarifying word2vec behaviour
- Bayesian version: Barkan (2016)

  `https://arxiv.org/ftp/arxiv/papers/1603/1603.06571.pdf`

doc2vec: Le and Mikolov (2014)

- Learn a vector to represent a 'document': sentence, paragraph, short document.
- skip-gram trained by predicting context word vectors given an input word, distributed bag of words (dbow) trained by predicting context words given a document vector.
- order of document words ignored, but also dmpv, analogous to cbow: sensitive to document word order
- Options:
  1. start with random word vector initialization
  2. run skip-gram first
  3. use pretrained embeddings (Lau and Baldwin, 2016)

doc2vec: Le and Mikolov (2014)

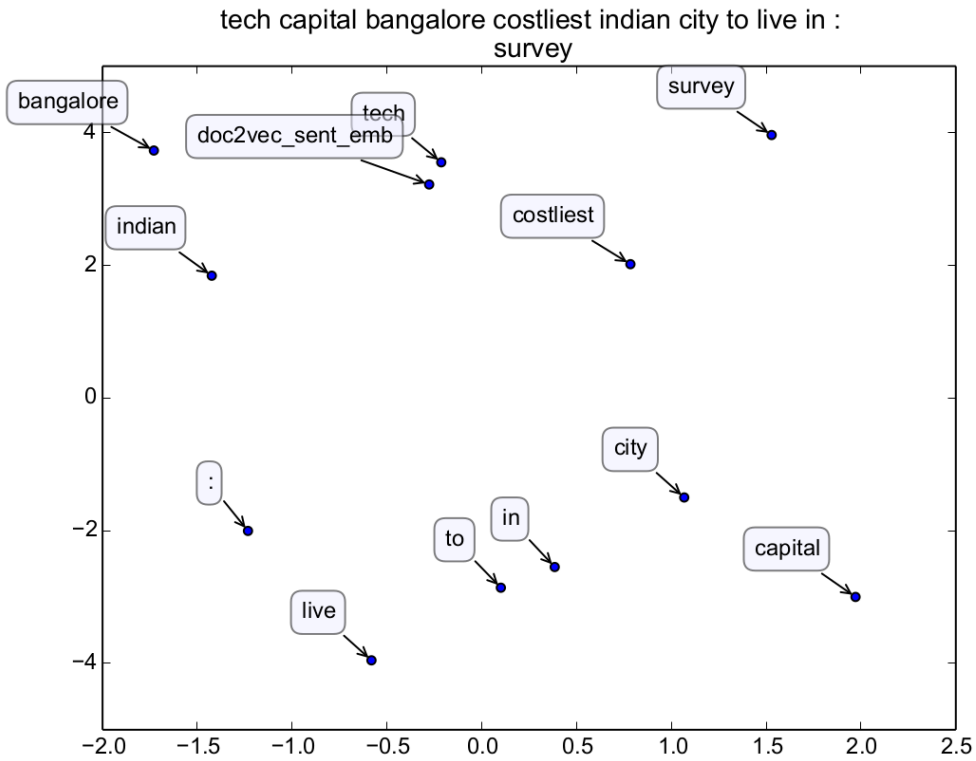
- Learned document vector effective for various tasks, including sentiment analysis.
- Lots and lots of possible parameters.
- Some initial difficulty in reproducing results, but Lau and Baldwin (2016) have a careful investigation of doc2vec, demonstraing its effectiveness.

Outline.

Finding out what NNs are really doing

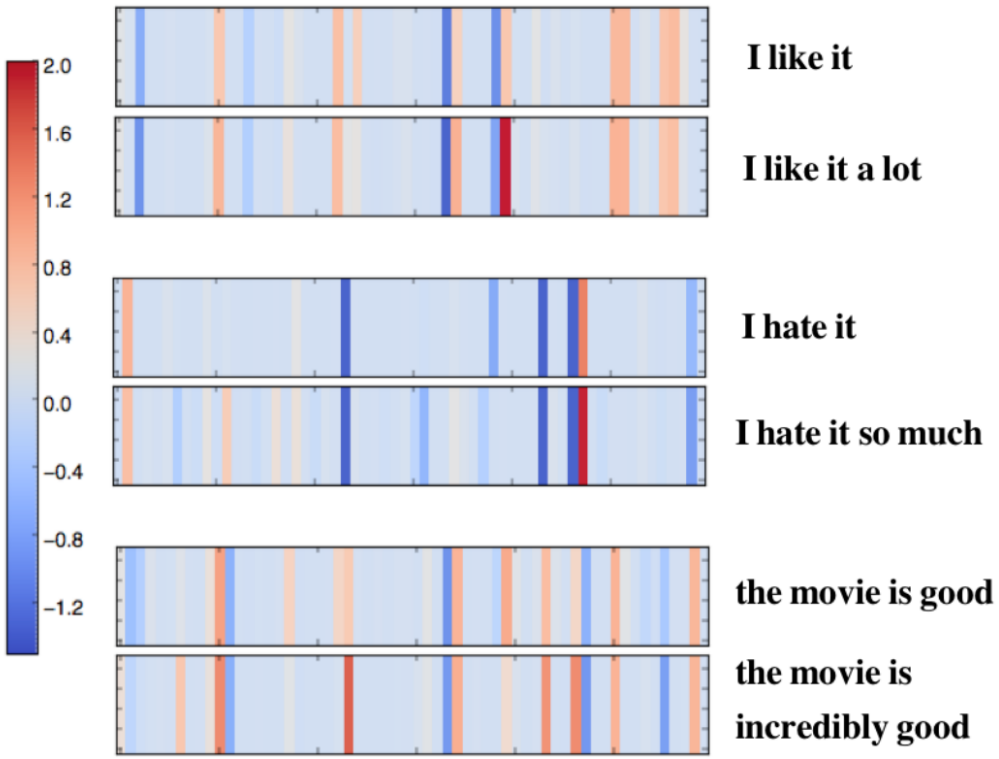
- Careful investigation of models (sometimes including going through code), describing as non-neural models (Omer Levy, word2vec).
- Building proper baselines (e.g., Zhou et al, 2015 for VQA).
- Selected and targeted experimentation (examples in lecture 12).
- Visualization.

t-SNE example: Lau and Baldwin (2016



arxiv.org/abs/1607.05368

Heatmap example: Li et al (2015)



arxiv.org/abs/1506.01066

Outline.

Deep learning: positives

- Really important change in state-of-the-art for some applications: e.g., language models for speech.
- Multi-modal experiments are now much more feasible.
- Models are learning structure without hand-crafting of features.
- Structure learned for one task (e.g., prediction) applicable to others with limited training data.
- Lots of toolkits etc
- Huge space of new models, far more research going on in NLP, far more industrial research . . .

Deep Learning: negatives

- Models are made as powerful as possible to the point they are“barely possible to train or use” (`http://www.deeplearningbook.org` 16.7).
- Tuning hyperparameters is a matter of much experimentation.
- Statistical validity of results often questionable.
- Many myths, massive hype and almost no publication of negative results: but there are some NLP tasks where deep learning is not giving much improvement in results.
- Weird results: e.g., ‘33rpm’ normalized to ‘thirty two revolutions per minute’

  `https://arxiv.org/ftp/arxiv/papers/1611/1611.00068.pdf`
- Adversarial examples (lecture 12, maybe).

New methodology required for NLP?

- Perspective here is applied machine learning . . .
- Methodological issues are fundamental to deep learning: e.g., subtle biases in training data will be picked up.
- Old tasks and old data possibly no longer appropriate.
- The lack of predefined interpretation of the latent variables is what makes the models more flexible/powerful . . .
- but the models are usually not interpretable by humans after training: serious practical and ethical issues.