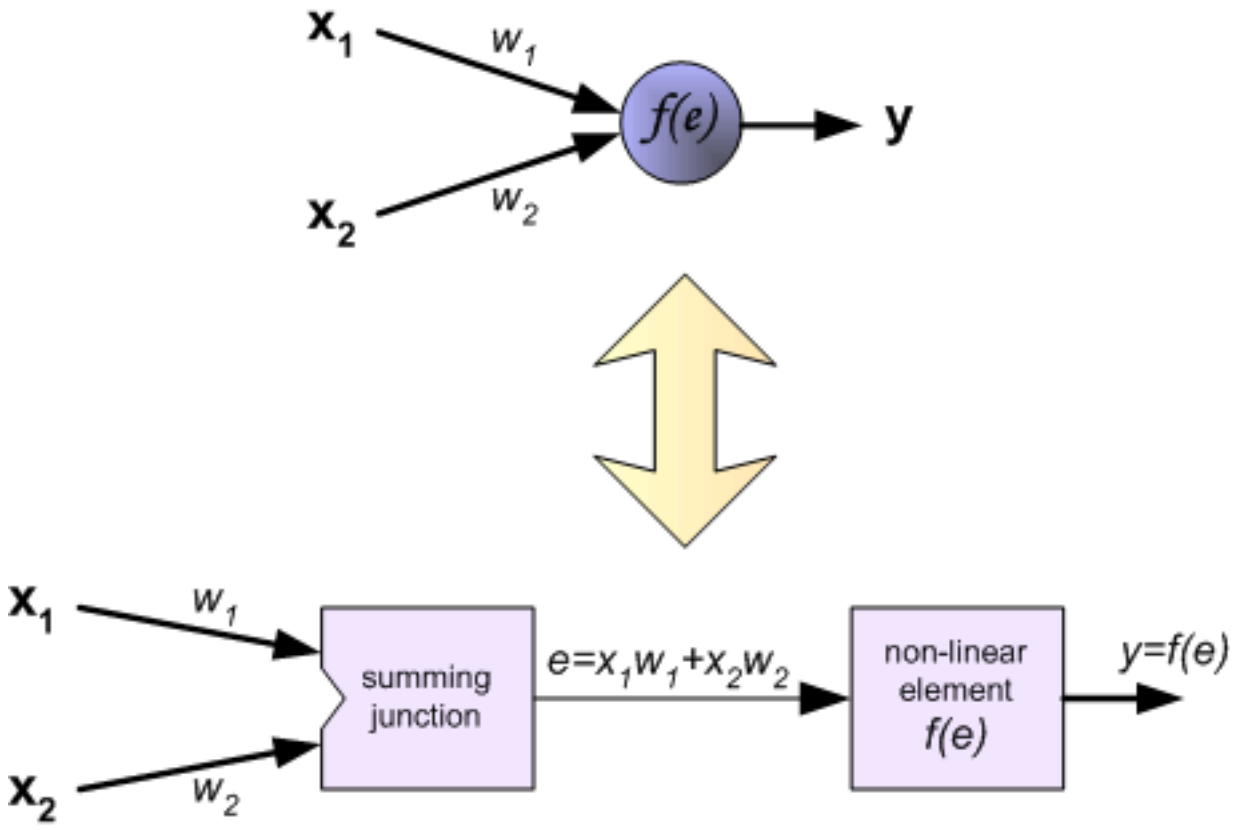
An Introduction to Neural Network based Language Modeling

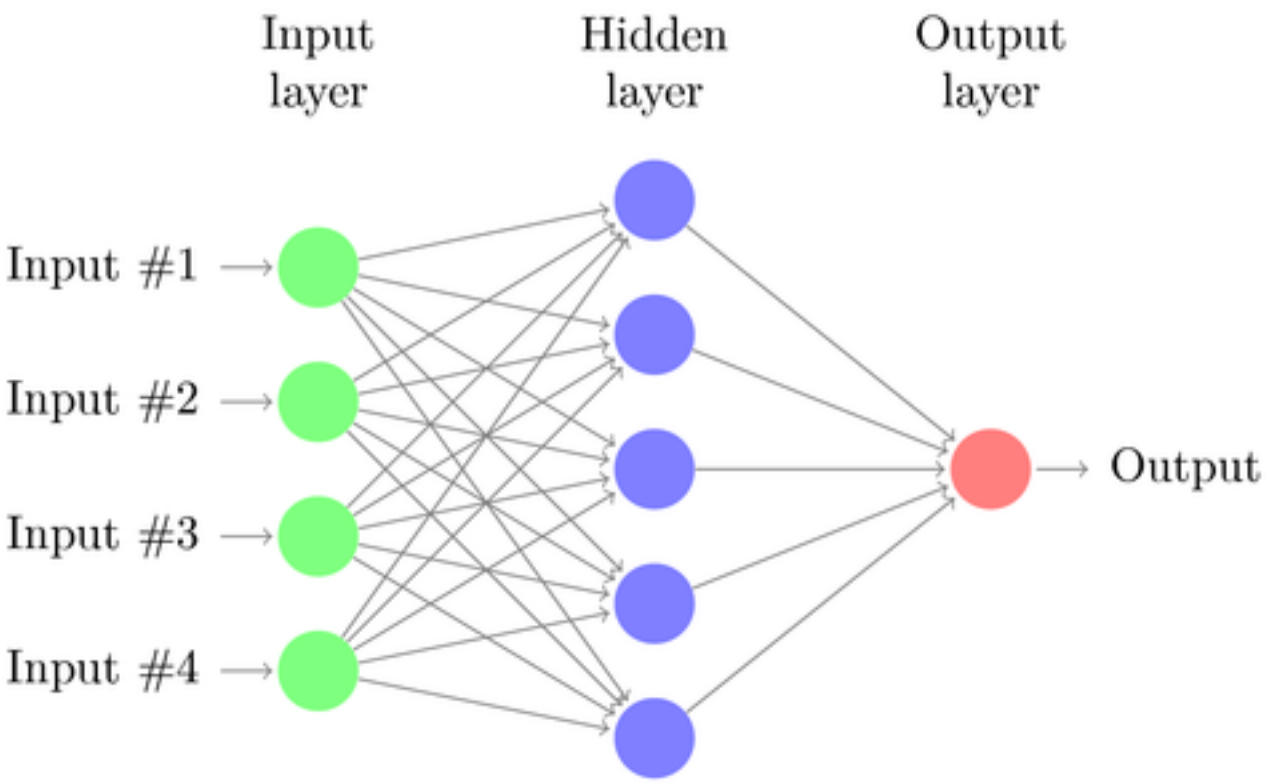
Manaal Faruqui

clab

Perceptron

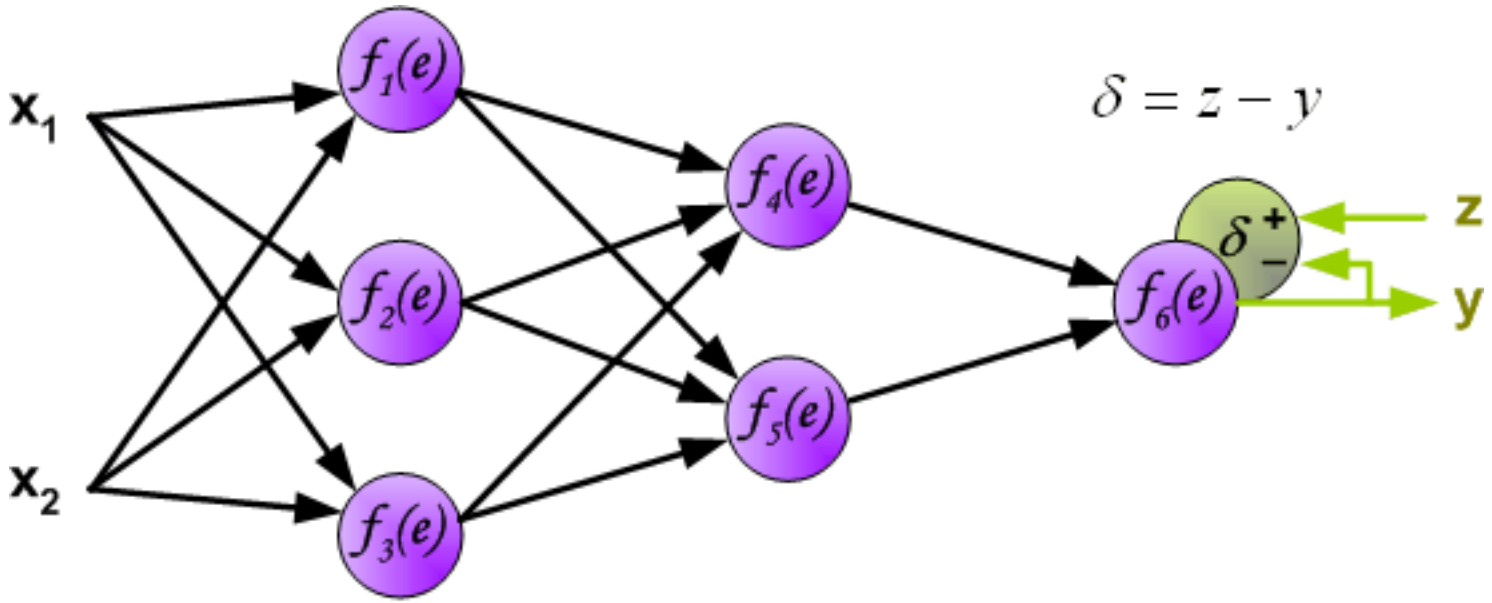

x1
w1
f(e)
y
x2
w2
x1
w1
summing
junction
e=x1w1+x2w2
non-linear
element
f(e)
y=f(e)
x2
w2

Neural Network



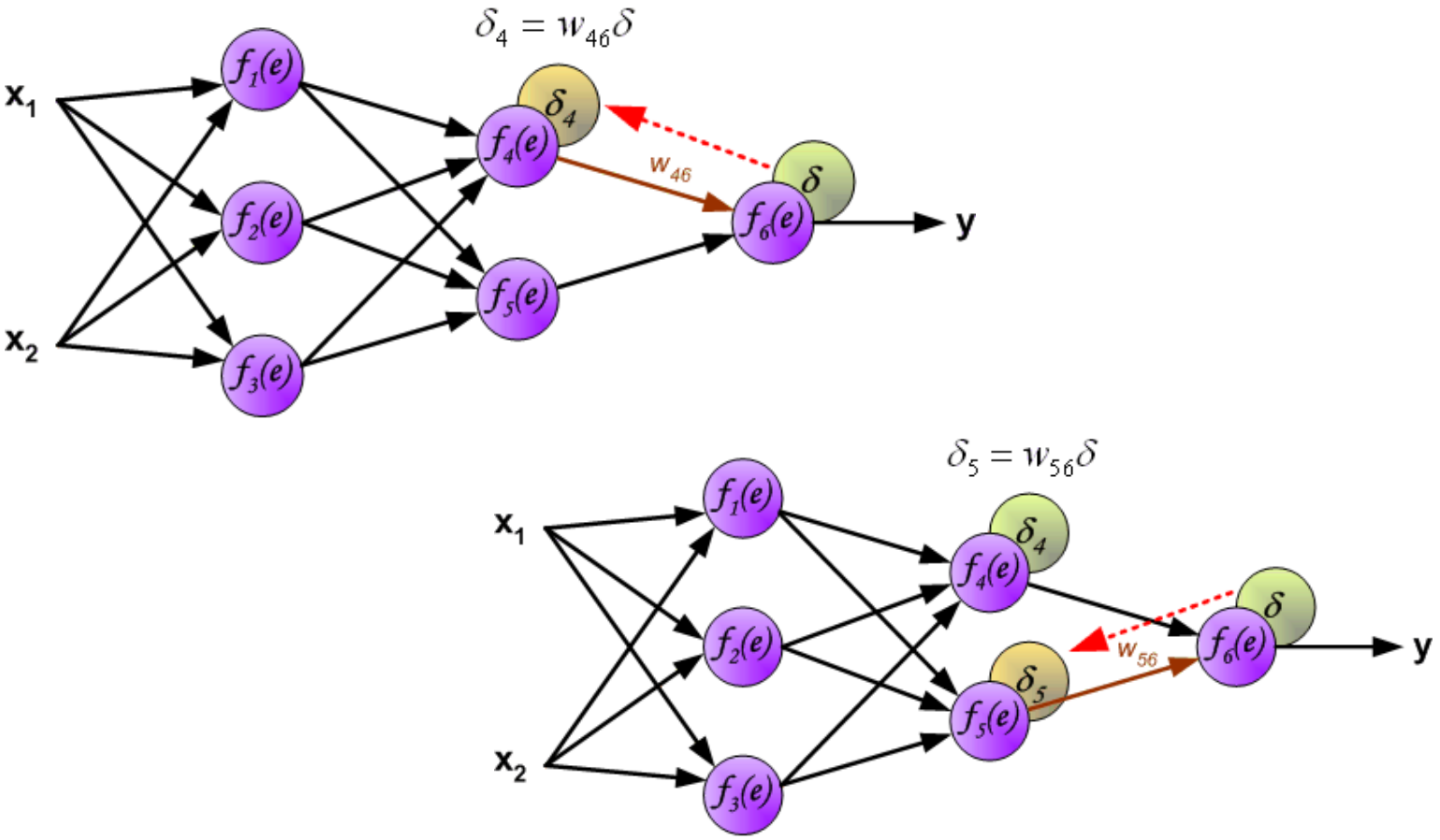
Learning: Backpropagation

Calculate error at the output



Images: http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

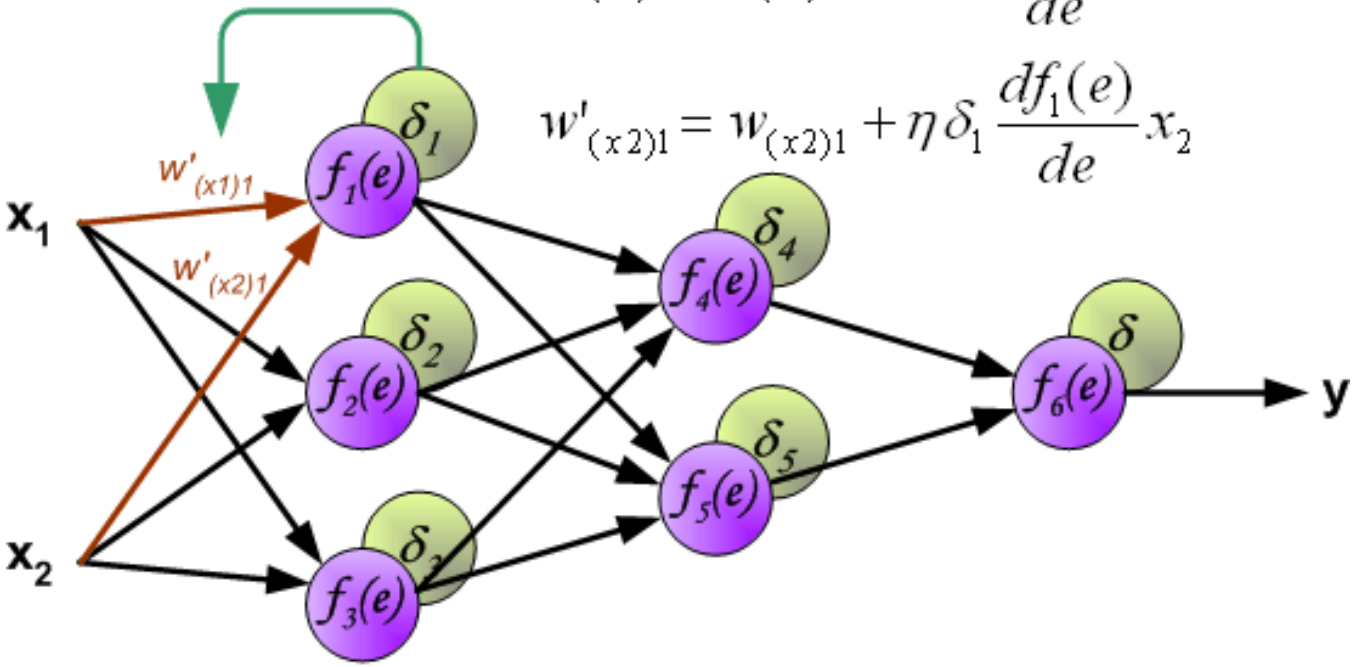
Learning: Backpropagation



Learning: Backpropagation

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

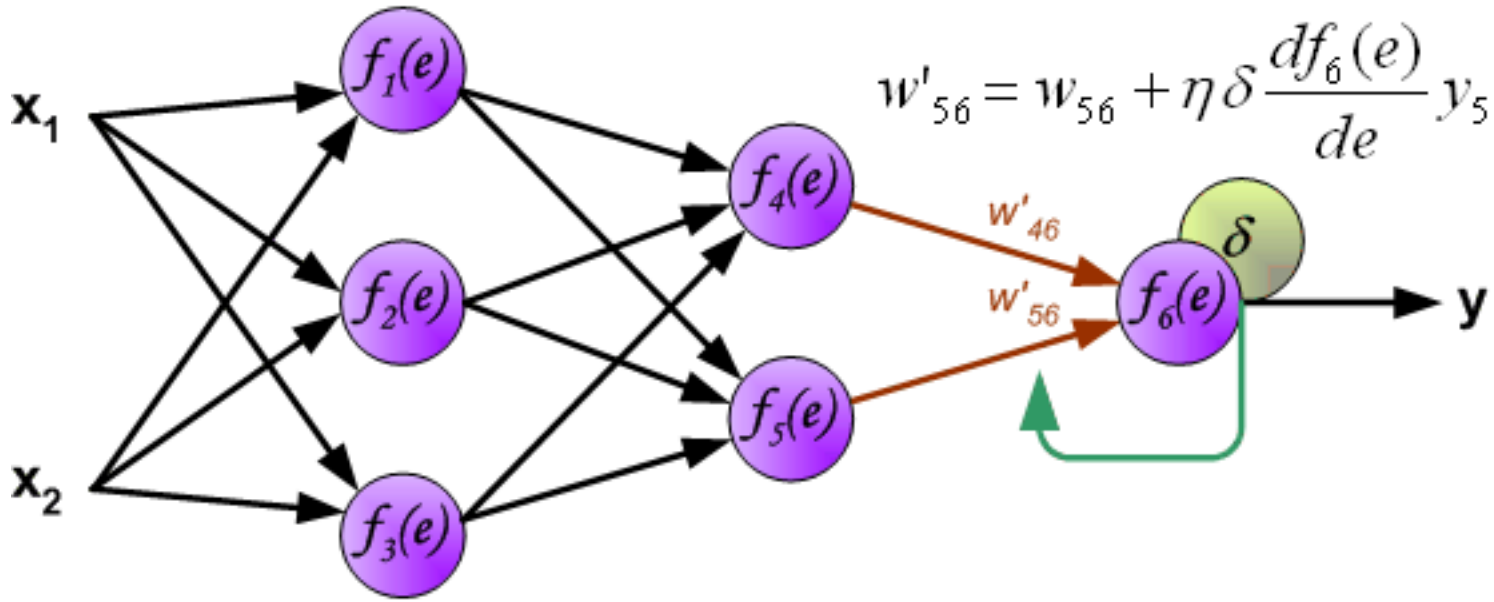
$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



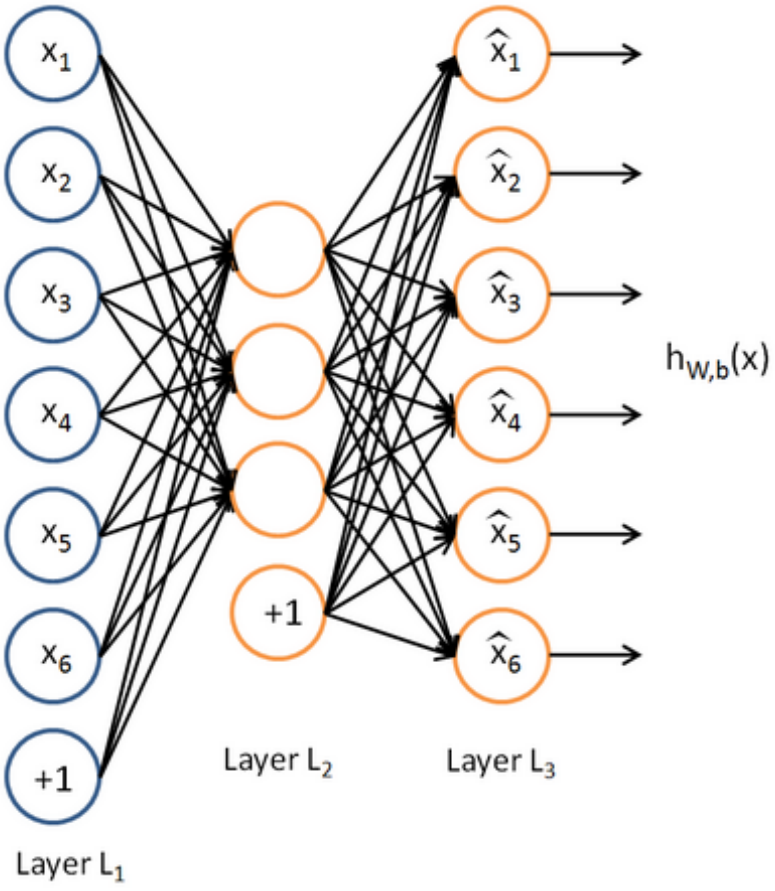
Learning: Backpropagation

$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$



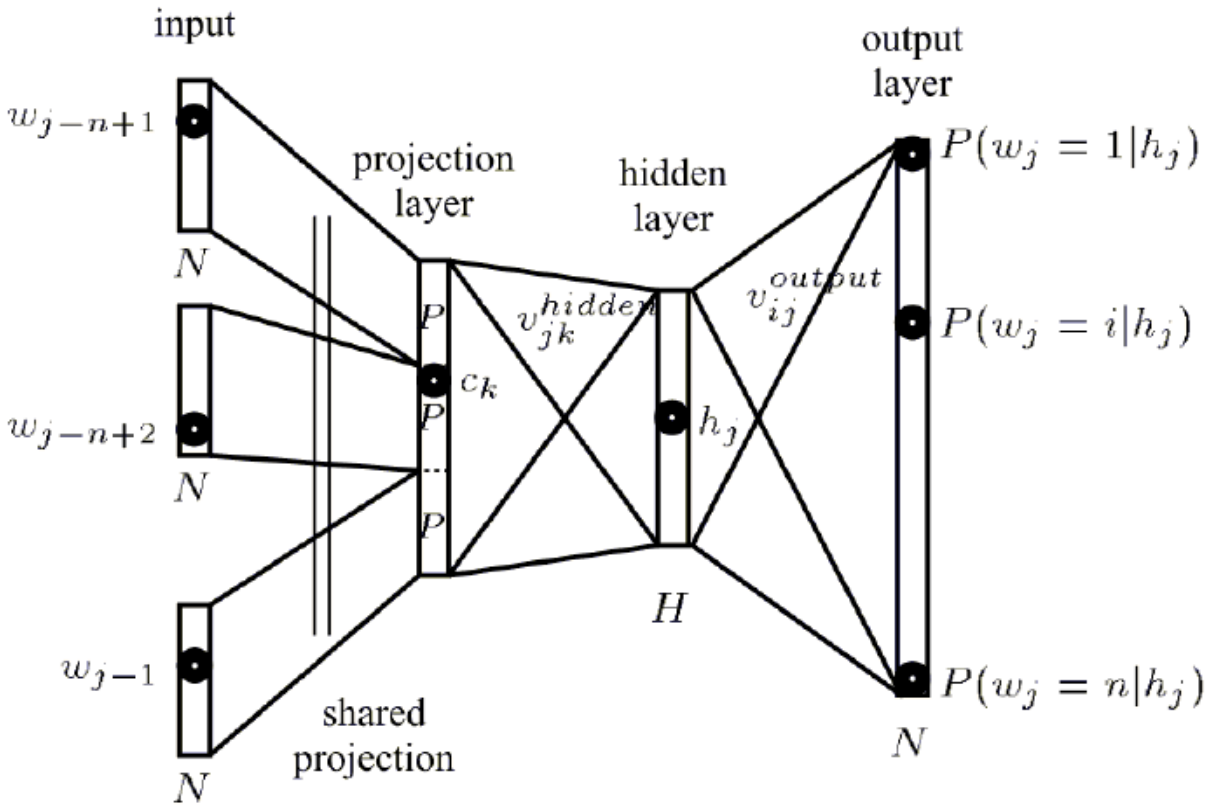
Auto-encoders



- Can be used for finding out a lower-dimensional representation of word vectors.
- Additional sparsity constraints on the can be enforced on the hidden layer

Image: http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Sparsity

Language modeling using NN

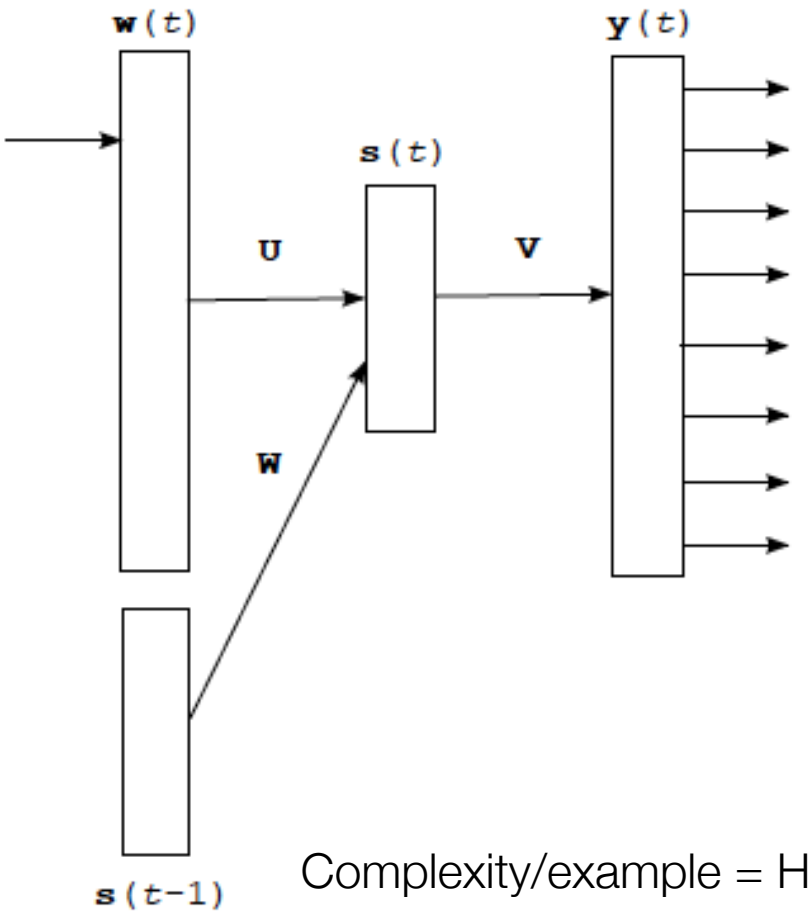


- N-previous words are encoded using 1-of-V coding
- Words are projected by a linear operation on the projection layer
- Softmax function is used at the output layer to ensure that 0 <= p <= 1
- Weights learnt using backpropagation

Complexity/example = N * D + N * D * H + H * V

Image Courtesy: Tomas Mikolov

Language modeling using Recurrent NN

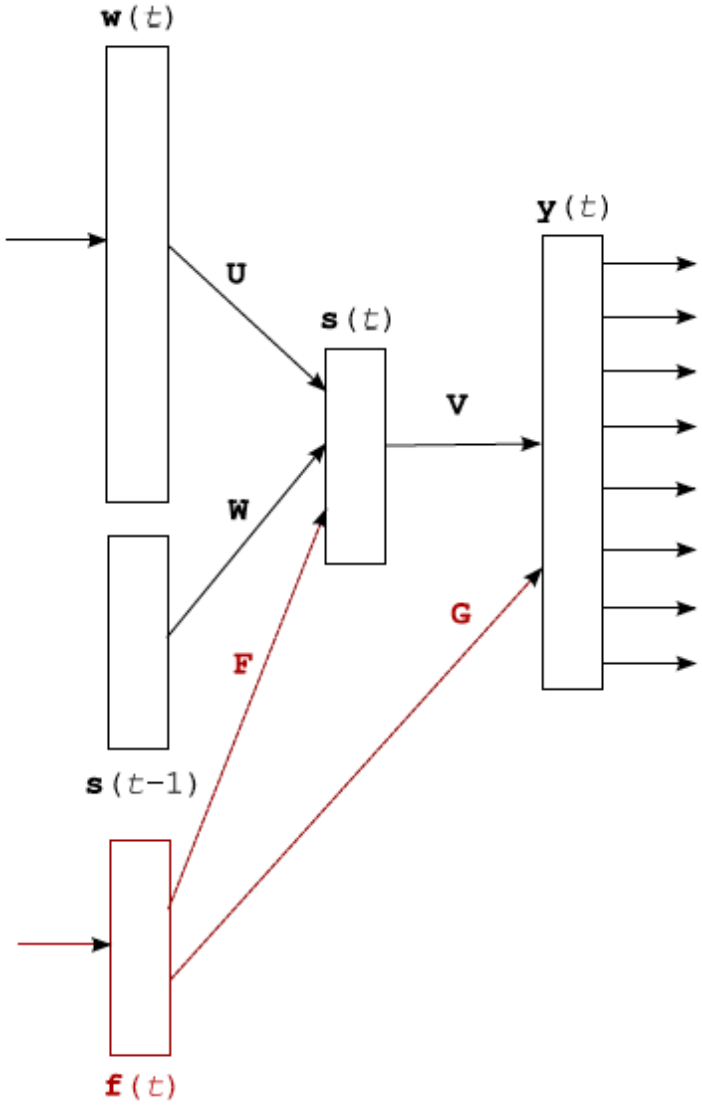


- No need to specify the context length
- No projection layer
- Hidden layer of the previous layer connects to the hidden layer of the next word
- Some kind of a short term memory which has information about the history
- Without the recurrent term this would reduce to ?

Complexity/example = H * H + H * V

Image Courtesy: Tomas Mikolov

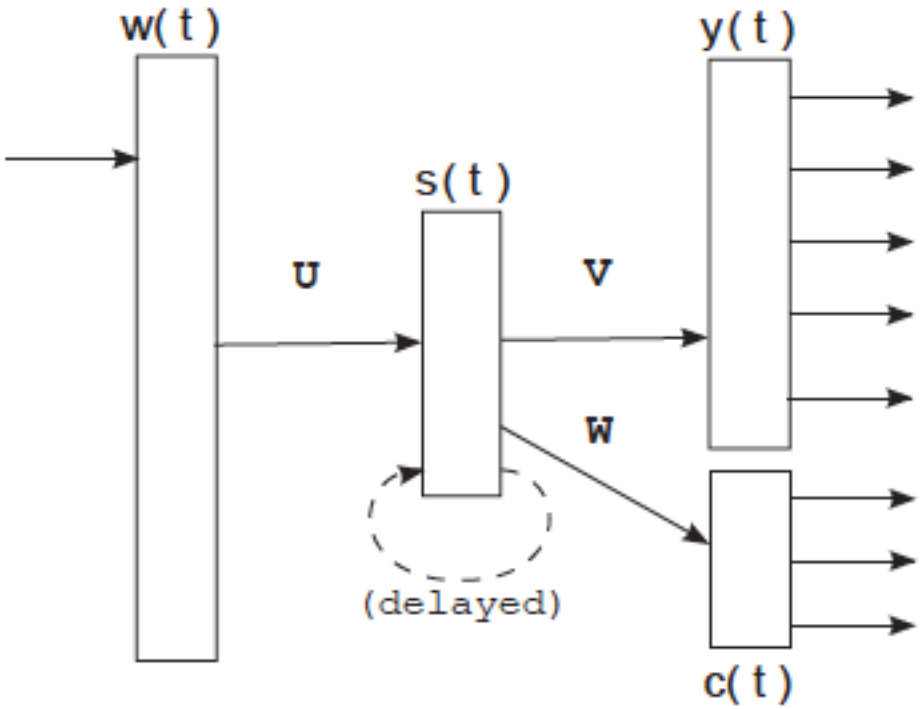
Extensions in Recurrent NNLM



- Adding more smart features like morphological information, POS information etc.

Image Courtesy: Tomas Mikolov

Extensions in Recurrent NNLM



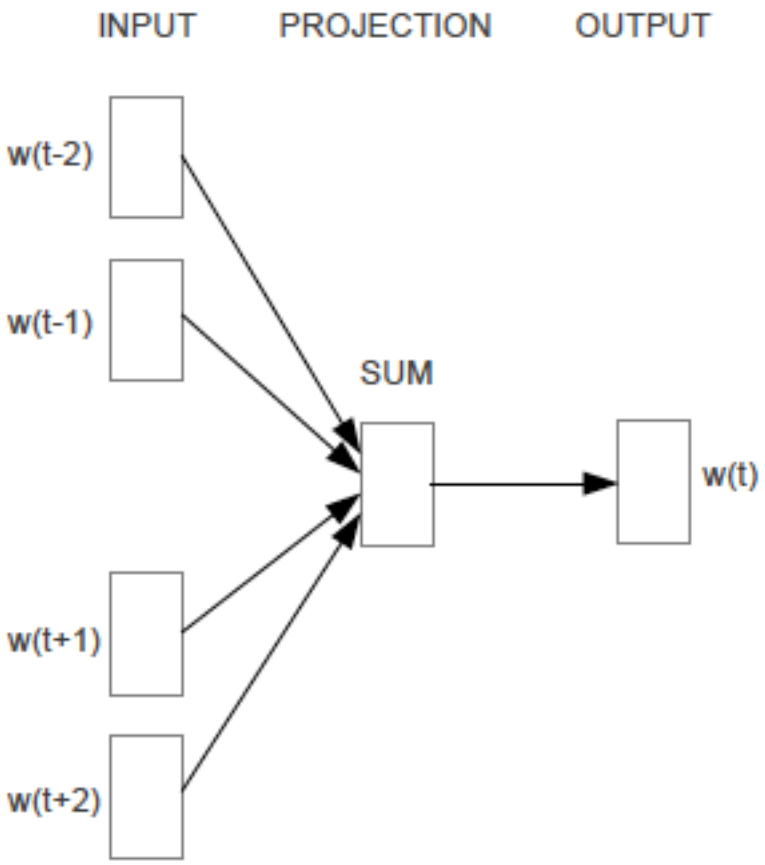
- Factorization of the output layer into classes

$$P(w_i|history) = P(c_i|\mathbf{s}(t))P(w_i|c_i, \mathbf{s}(t))$$

???

Image Courtesy: Tomas Mikolov

New Log-Linear models being trained by Google



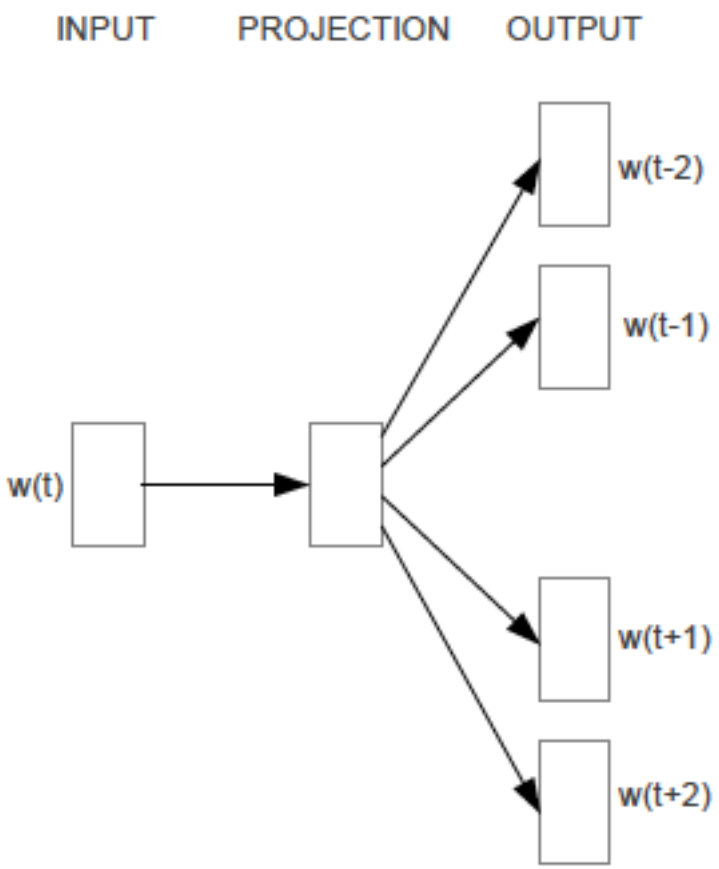
Continuous Bag-of-words model

- No hidden layer
- Predicting a given word given its past and future context

1. Projecting the contextual word vectors (averaging)
2. Running a log-linear classifier on the averaged vector to get the resultant word

Image Courtesy: Tomas Mikolov

New Log-Linear models being trained by Google



Skip-gram model

- No hidden layer

1. Each word is used as an input to a log-linear classifier to predict contextual words

2. Increasing the range improves the prediction but is costlier

Image Courtesy: Tomas Mikolov

Resources

Implementation of Recurrent Neural Network Language Model

- http://research.microsoft.com/en-us/projects/rnn/
- Takes approx. one hour for one iteration
- For 1 sentences (24 m words) approx. 15 iterations = 15 hours

Faster models being developed at Google to be released soon

Joseph Turian's imlpementation of the NNLM

- https://github.com/turian/neural-language-model

Kaldi: http://sourceforge.net/projects/kaldi/

Restricted Boltzmann Machines (to be covered by Chris)

- https://code.google.com/p/visual-rbm/