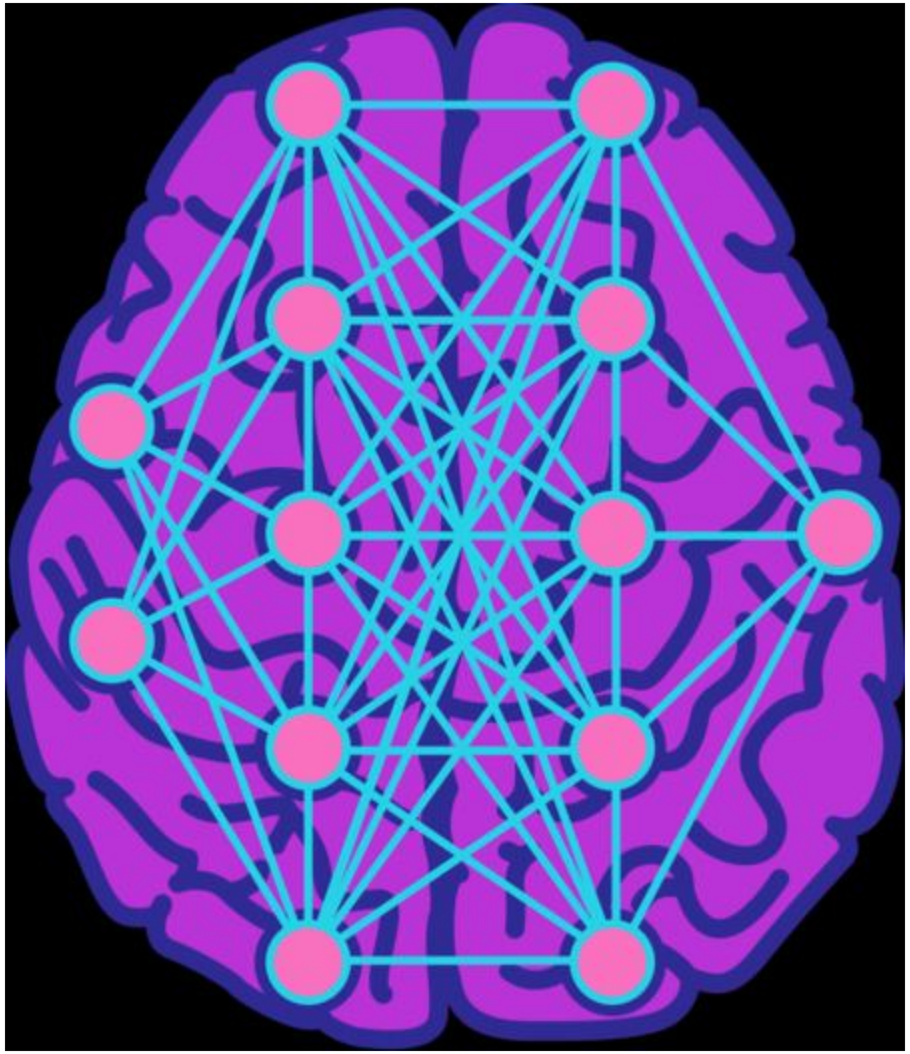
Neural Networks for NLP

Yan Song

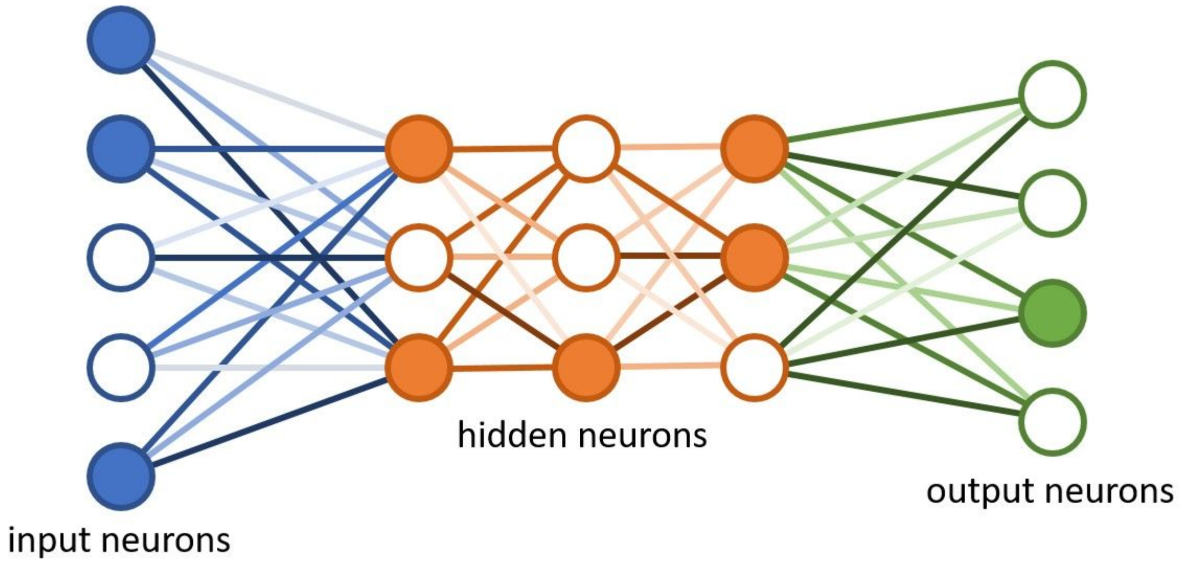
Outline

- Basic Concepts
- MLP
- RNN (LSTM)
- CNN
- Seq2Seq
- Attention
- Optimizers
- Implementation with Keras



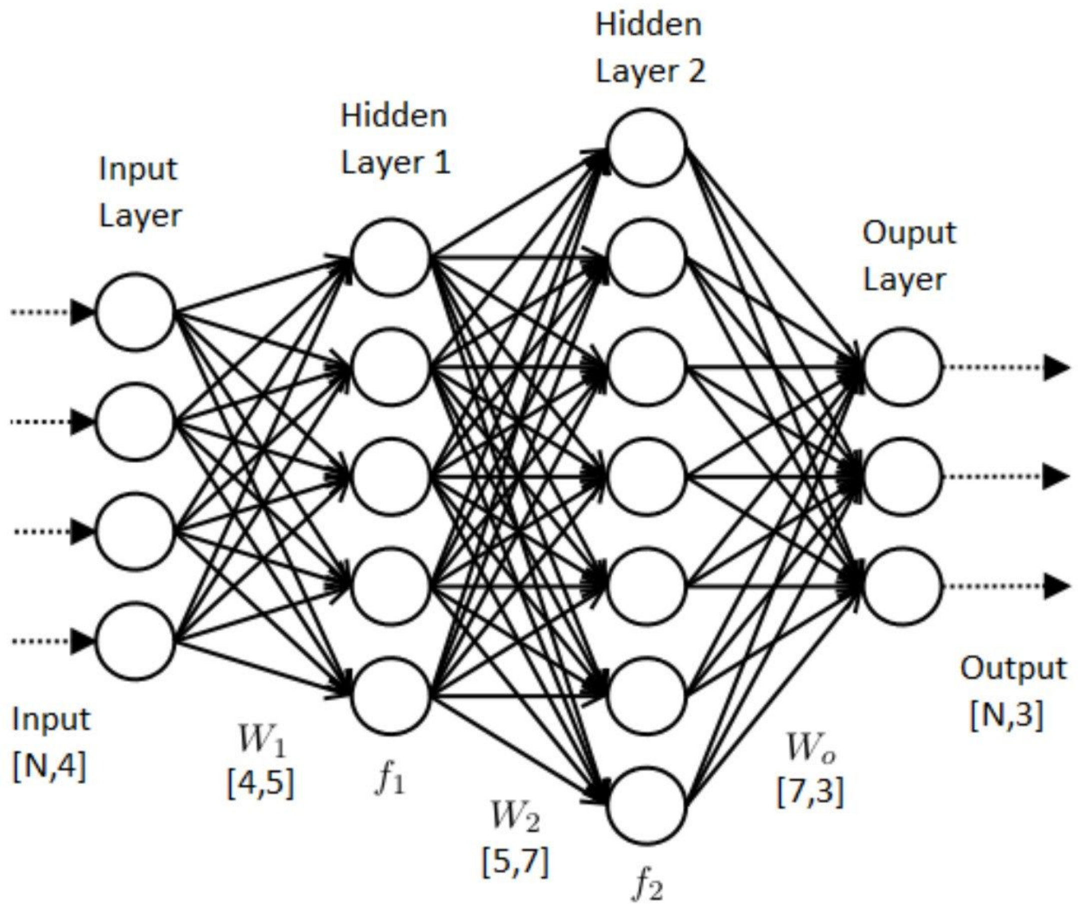
Basic Concepts

- Multi-node graph structure
- Multiple intermediate layers (Deep learning)
- Connections are weighted (Parameters to be learned)



Basic Concepts

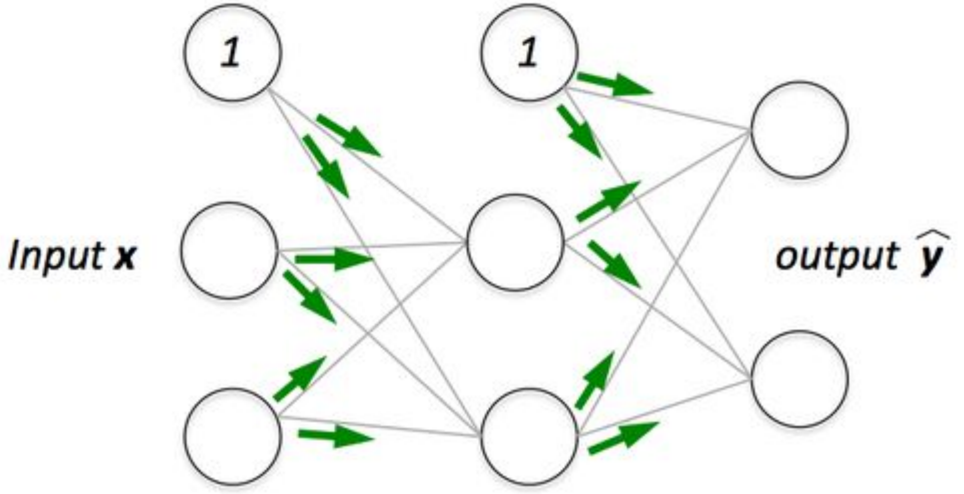
- Parameters are organized in matrices
- Matrices are stacked
- Convert complicated problems into matrix computation
- Automatic extracting salient input information
- Easy to scale up and down

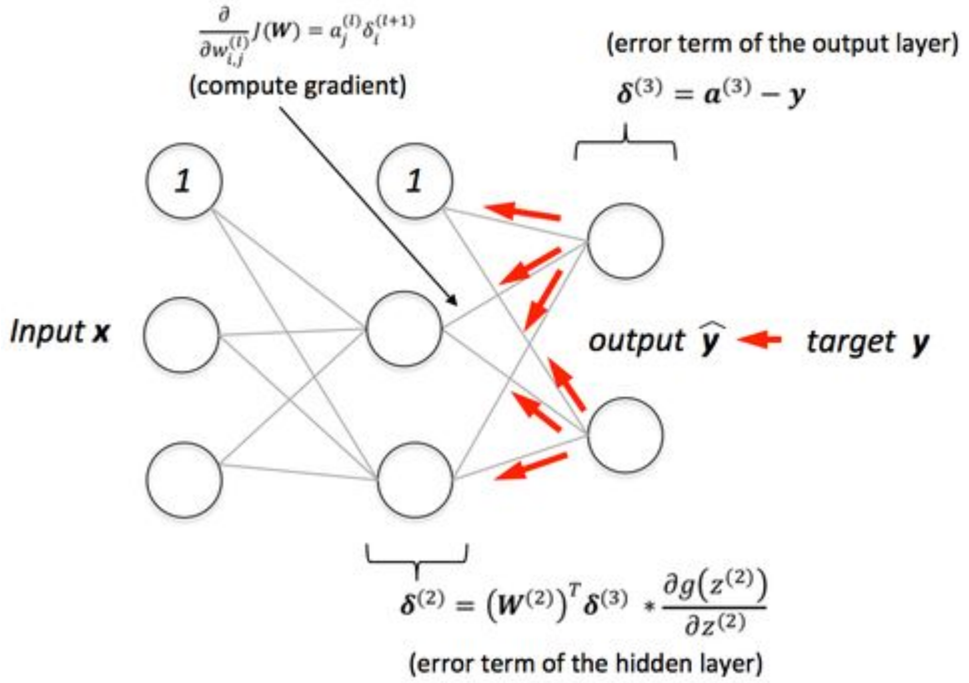


Basic Concepts

- Back-propagation
  - Key to neural model learning
  - Can be done layer-wisely

- Important for representation learning
  - Updating input layer
  - Adjustable during learning





Basic Concepts

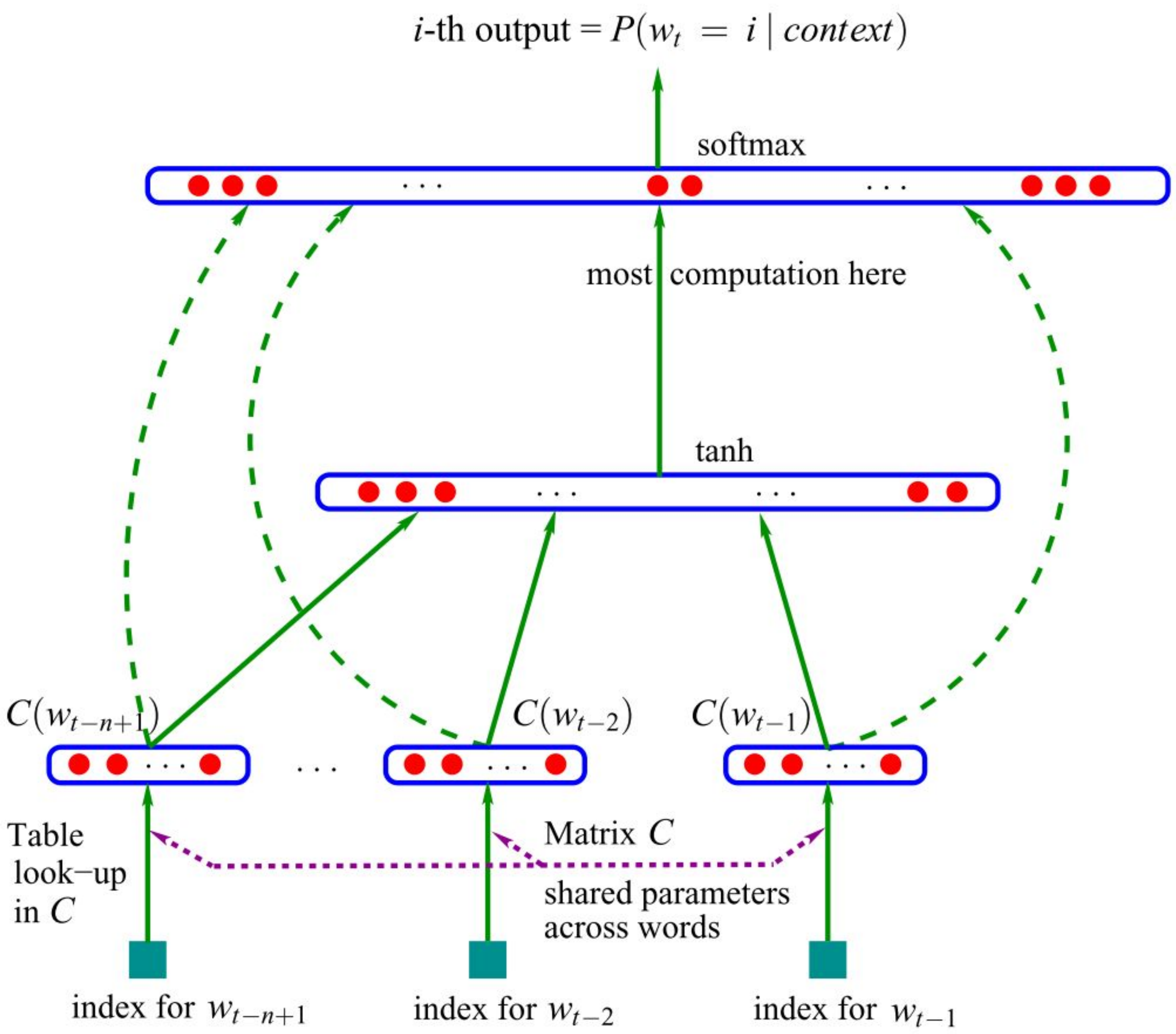
- Important factors for representation learning
  - Different layers can be utilized, esp. the input and output layer
  - The represeatation power depends on the layer depth
    - Input layer normally for words
    - Ouptut layer for sentences
    - …
  - Representation capability can be different from the network learning objective

Basic Concepts

- Normal NLP tasks
  - Word Segmentation
  - POS Tagging
  - Text Classification
  - Sentiment Analysis
  - Machine Translation
  - ...
- How many types of model required?
  - Classification
  - Sequence Labeling

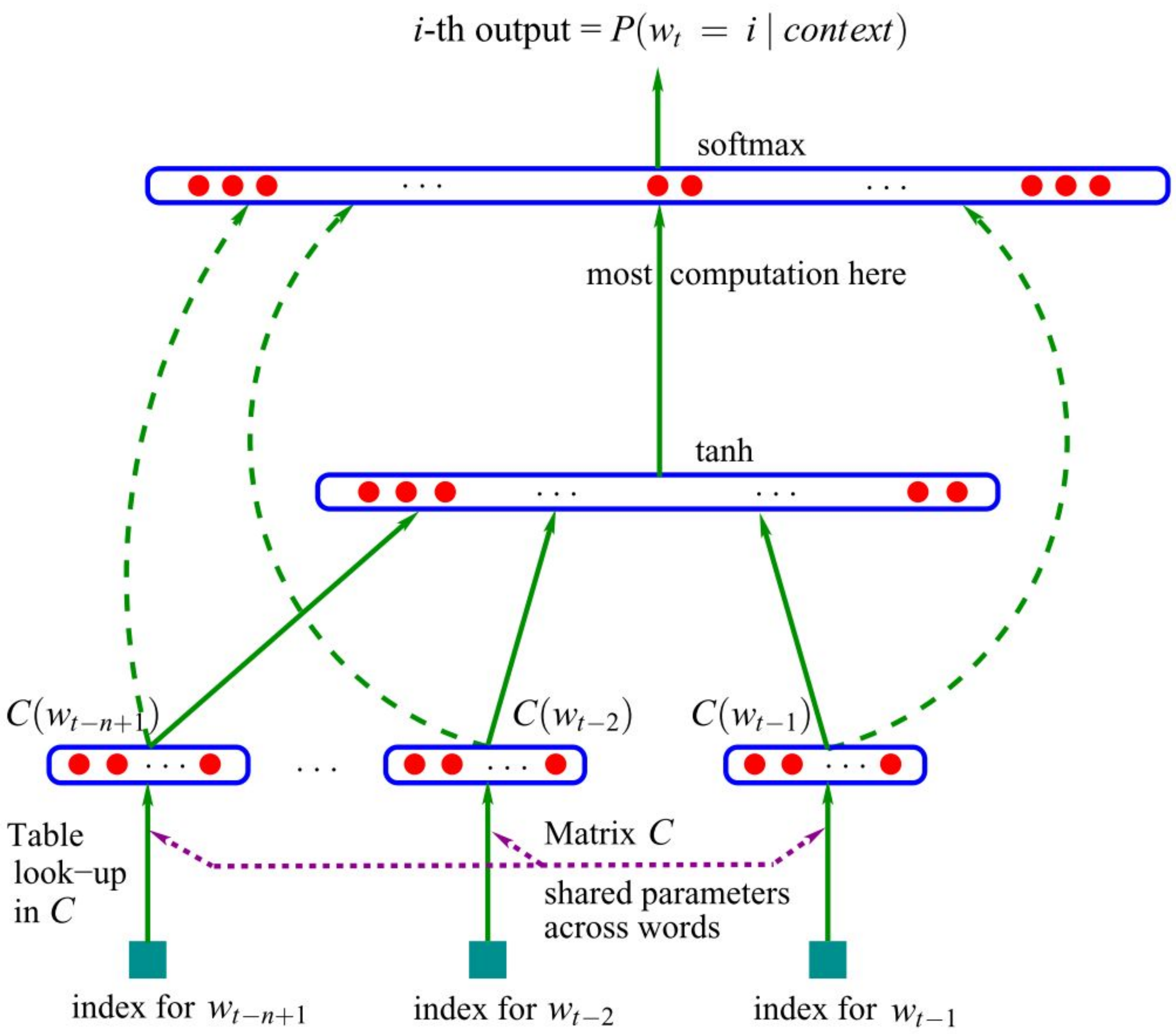
MLP

- Multi-layer Perceptron
- Layer-wise information flow
- +/- layer activation
- Can be trained in a parallel way



MLP

- For classification-style tasks
  - Language modeling
  - Document categorization
  - Sentiment analysis
  - ...
- The structure for word2vec
  - Context-to-target prediction
  - Efficient output layer design

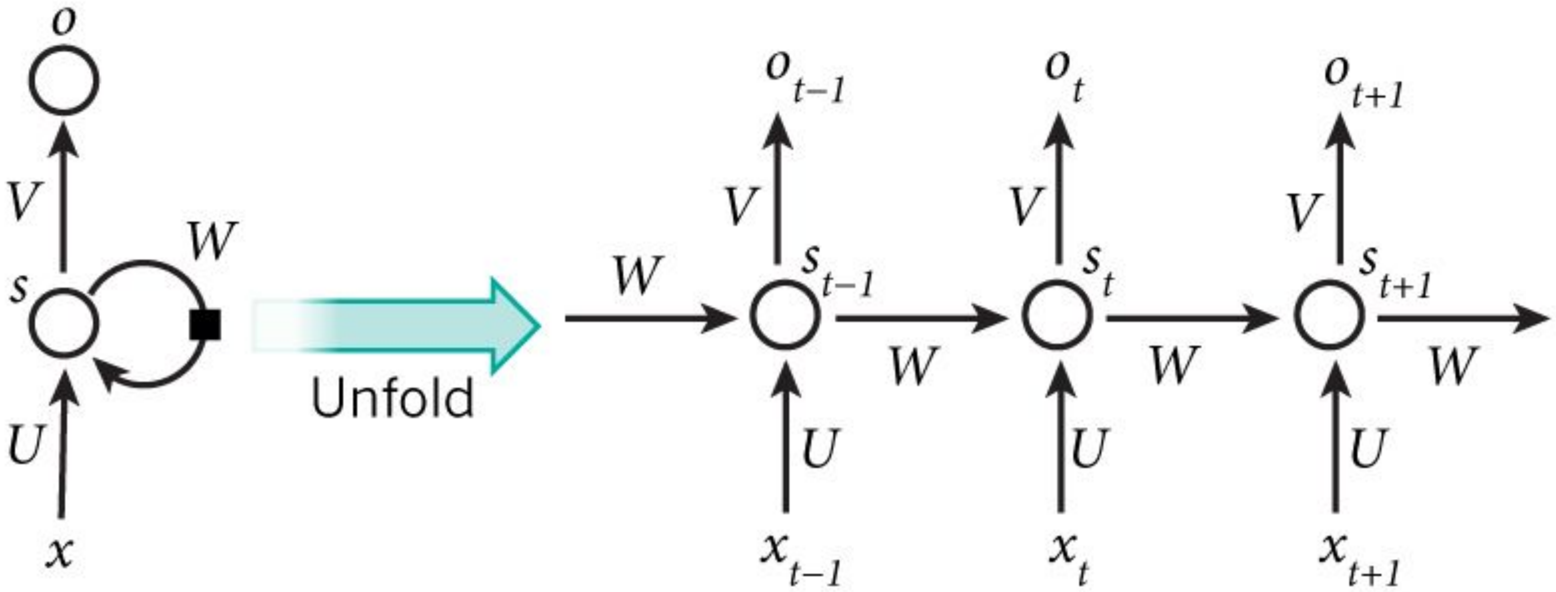


MLP

- Advantage and Disadvantage
  - Easy structure
  - Efficient in parallel training
  - Cannot handle langauge structures
  - Require a vocabulary-size input dimension
- Some tricks (esp. for NLP)
  - For most tasks, use 1-2 hidden layers
  - Use linear activation among layers
  - Best use as an ensemble model

RNN

- Recurrent model
- The node is an MLP
- Captures structure
- Stateful



- Problems
  - Gradient vanishing
  - Not easy to be paralleled

Why RNN?

The trophy would not fit in the brown suitcase because it was too **big**.

Trophy

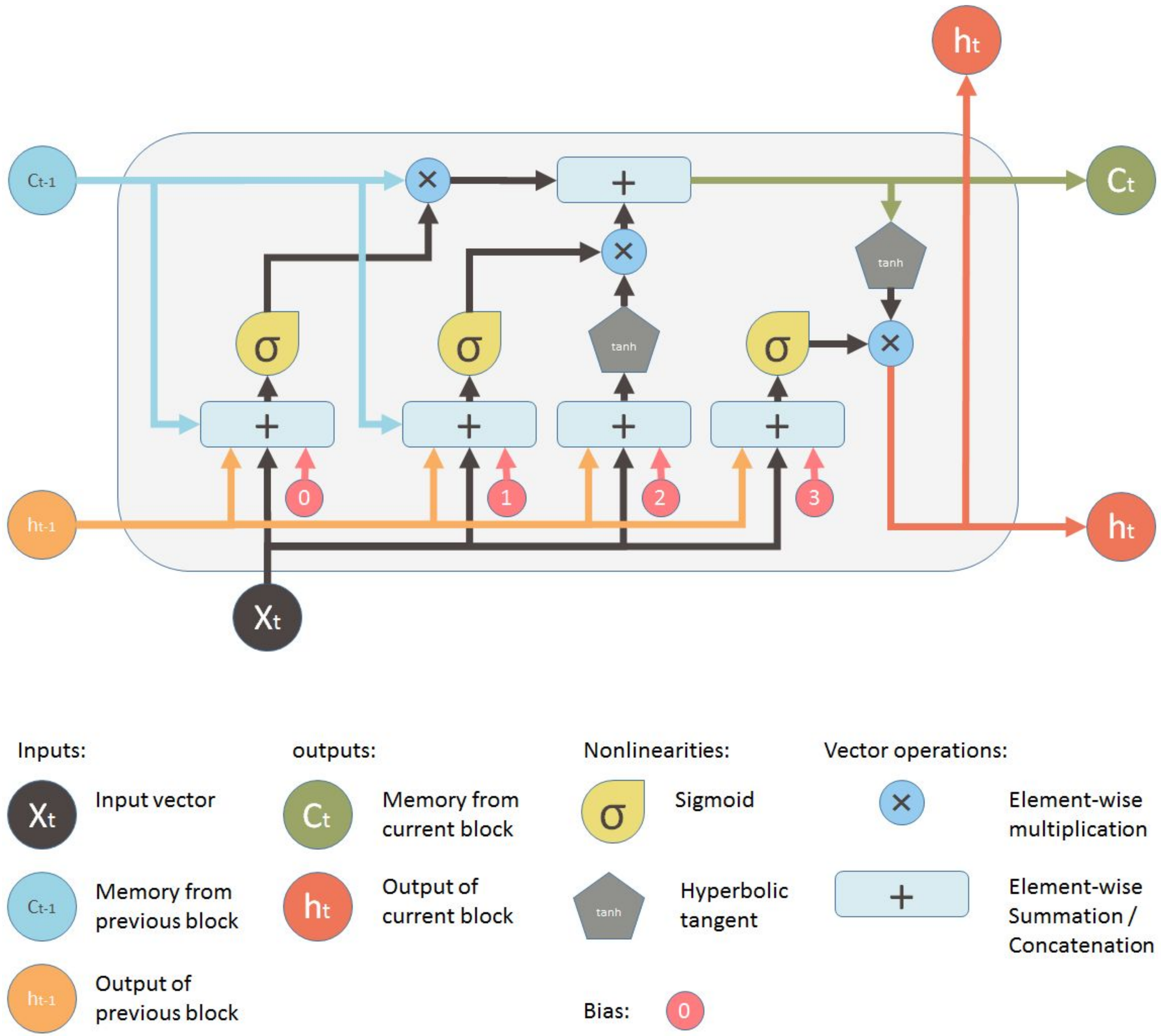
The trophy would not fit in the brown suitcase because it was too **small**.

Suitcase

- Language is in a time-sequence form
- Being able to model structure information (MLP cannot)
- Remembers some information step-wise

LSTM

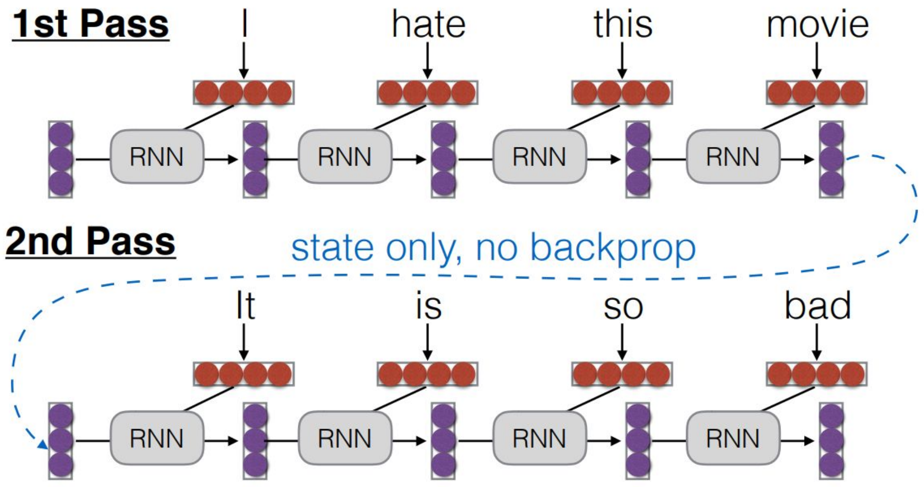
- Long short-term
- Additive connections between time steps
- Gates to control information flow



LSTM

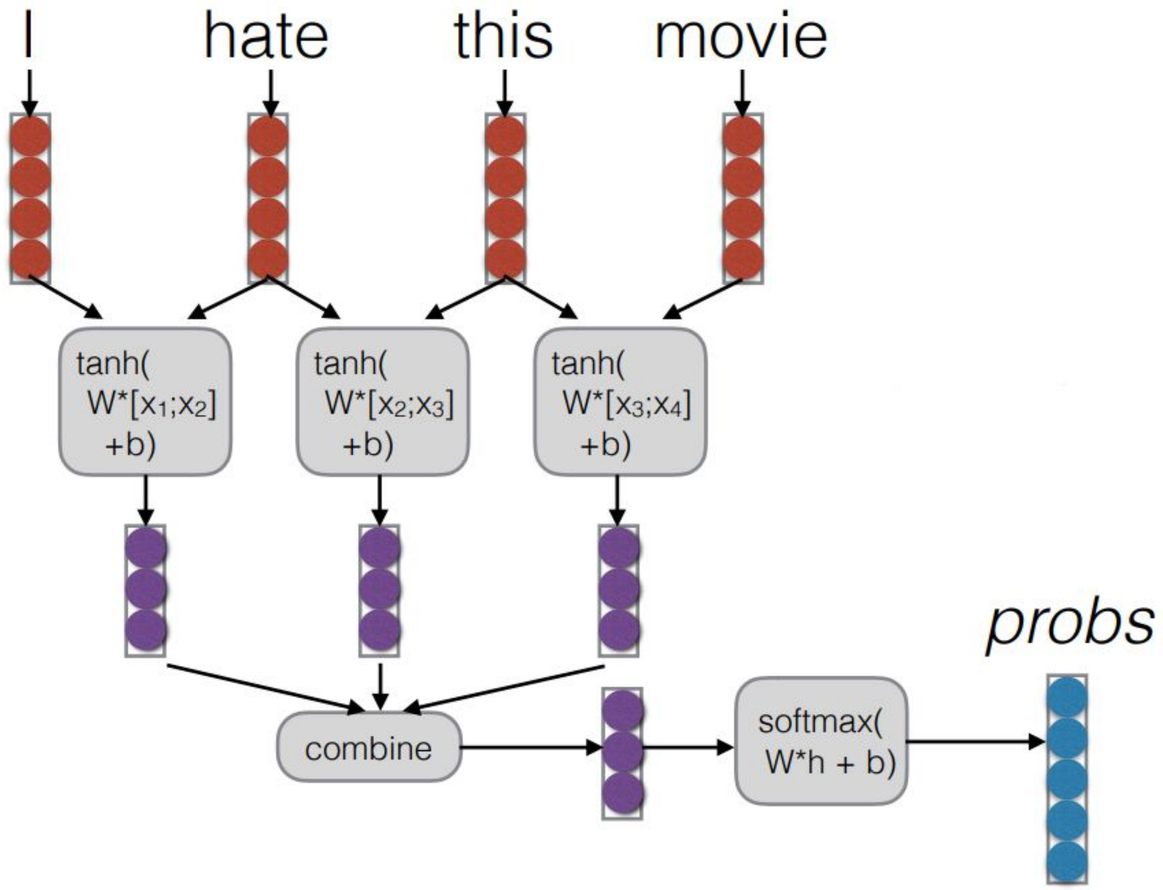
- Advantages
  - No gradient vanishing (Gradient go through "+" rather than "×")
  - Long distance dependency
  - Modeling structures (like RNN)

- Tricks
  - Always try RMSProp, not Adam
  - Use batch size to control training
  - Always use Dropout
  - How to handle long sequences? ----------->



CNN

- (Normally) 1-D CNN for NLP
- Enhanced bag-of-word model
- Capture local dependencies
  - Window size matters
- Efficient than recursive models
  - Can be paralleled
- More flexible possibilities in designing particular structures
  - More controllable parameters without structure change

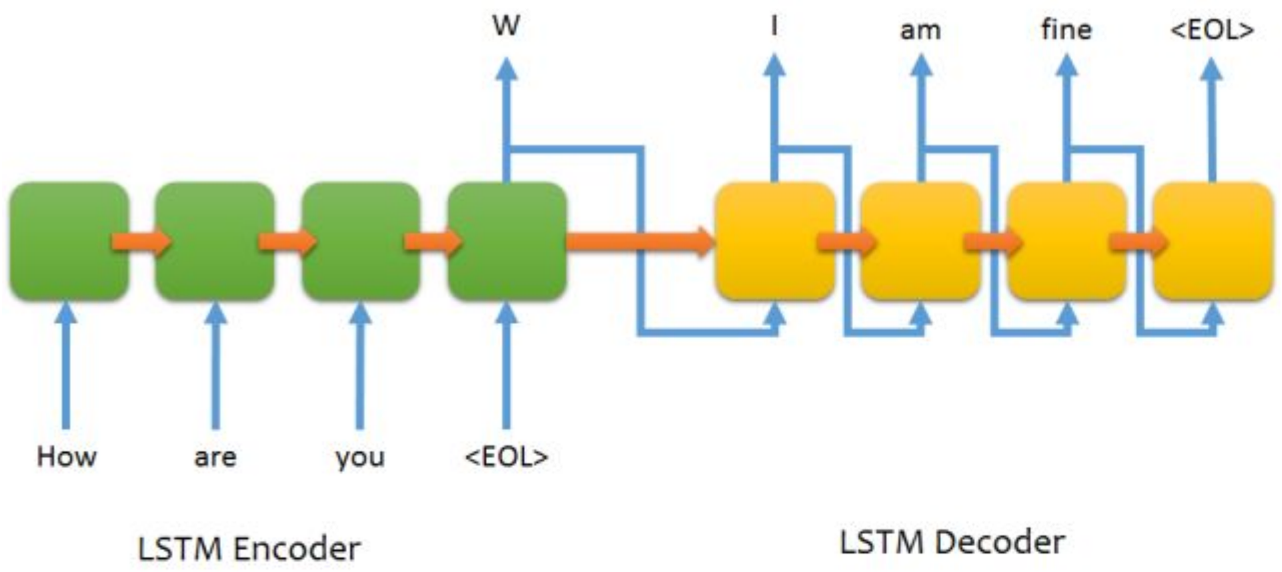


CNN

- Pooling
  - Max pooling: a very important feature appears at somewhere
  - Averaged pooling: frequent patterns of features appear in the entire range
  - k-Max pooling: an important feature has more than one time appearance
  - Dynamic pooling: an important feature may appear with arbitrary values

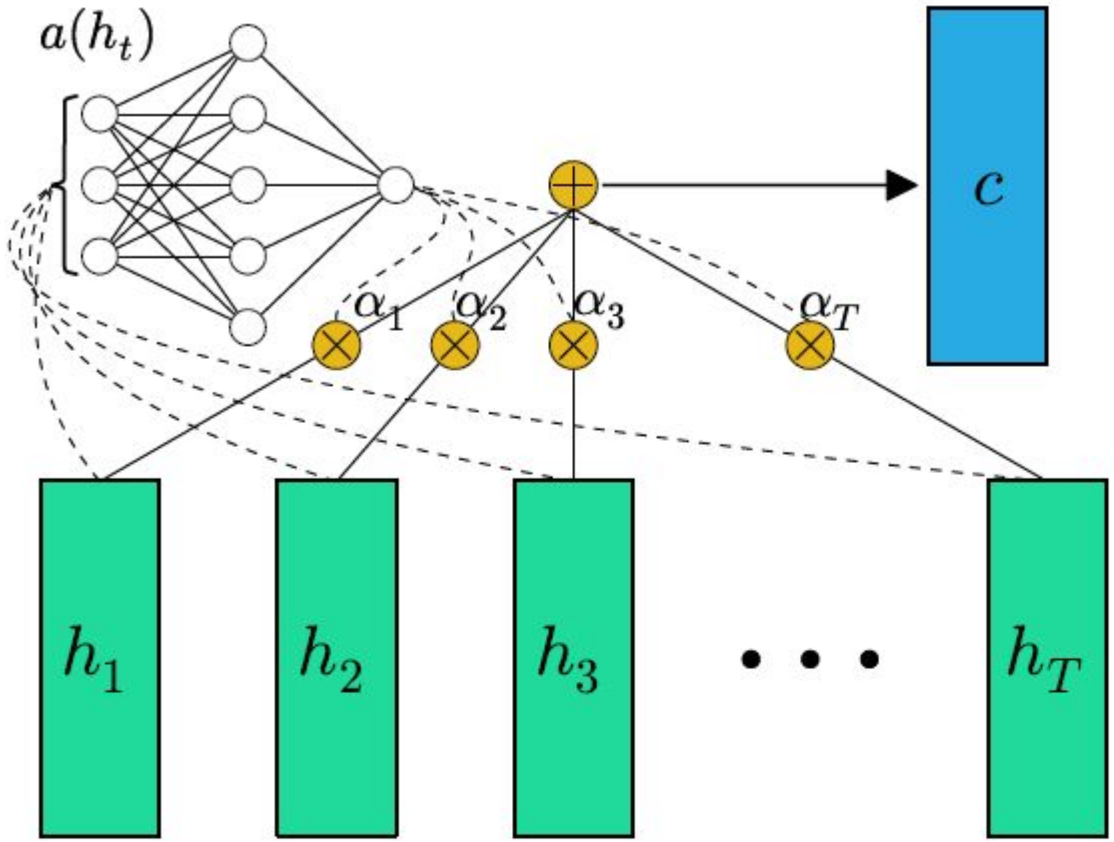
Seq2seq

- Sequence-to-sequence
- RNN-based structure
  - Can CNN do it?
- Decoder is a dynamic process (output + state)
- Why it works?
  - Thanks to recursive process
  - Generalization ability of neural networks
  - Representation of the input (encoder) is very powerful



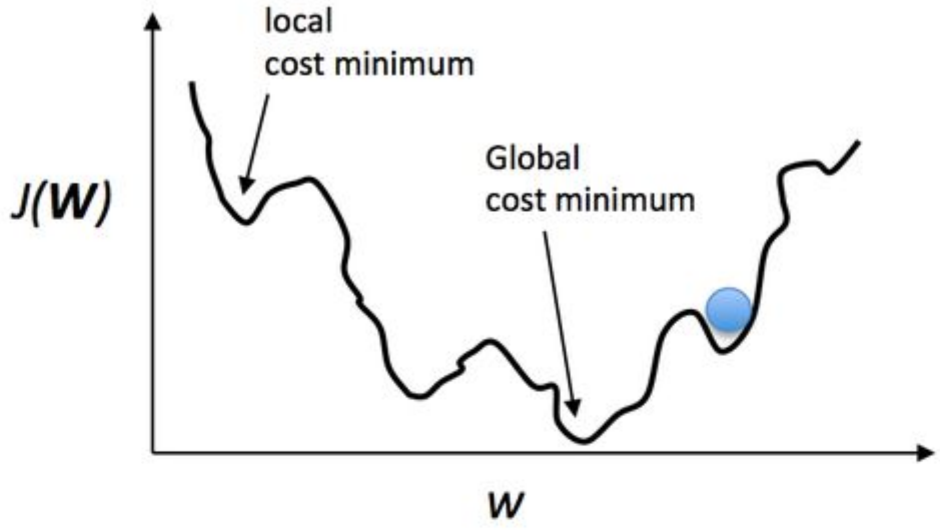
Attention

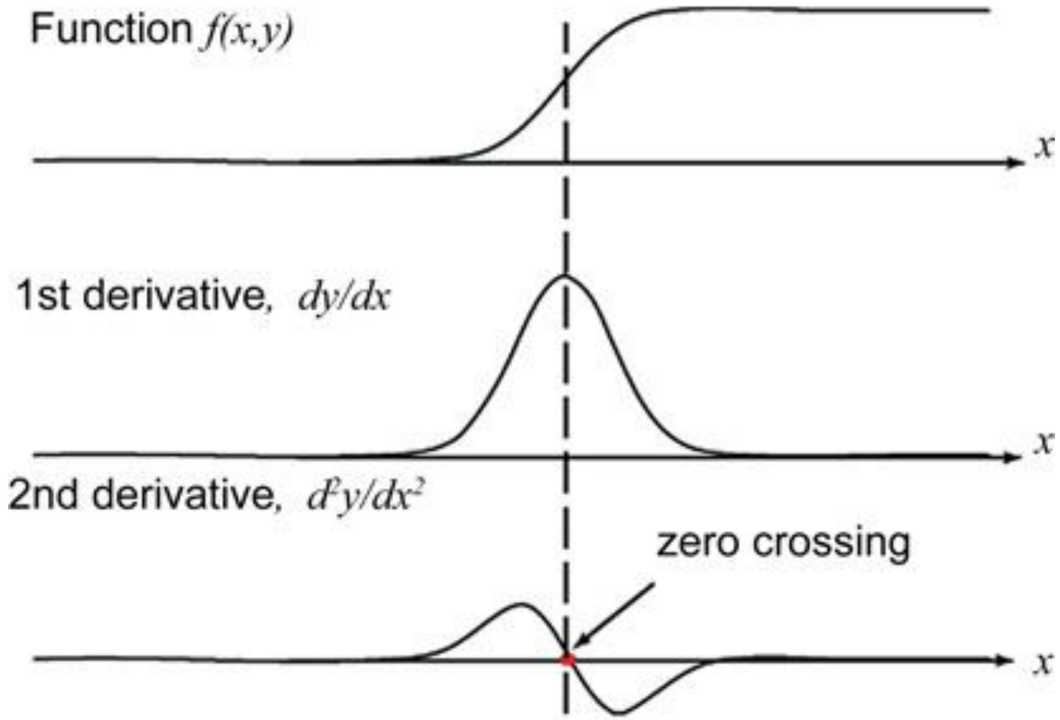
- Feed-forward attention
- Why we need attention
  - Break the equal contribution of each input (state)
  - Do not want to change the original network
  - Automatically learn the salient feature
- Tricks
  - Layer normalization
  - Control learning rate



Optimizers

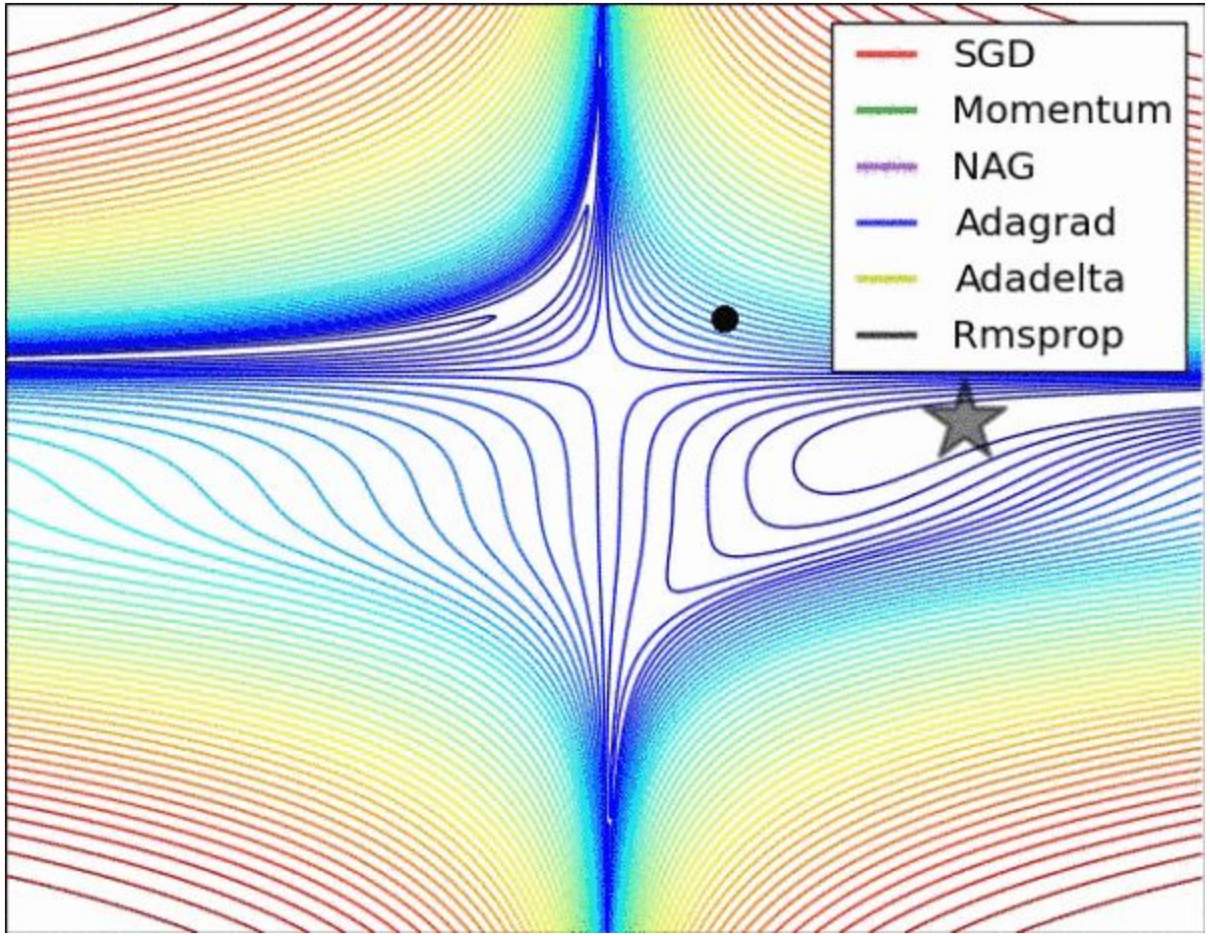
- SGD
- 1st Order Momentum
  - SGD-M
  - NAG
- 2nd Order Momentum
  - AdaGrad
  - AdaDelta
  - RMSProp
  - Adam
  - Nadam





Optimizers

- Visualization
- SGD
  - slow
  - easy to be entrapped
- RMSProp
  - momentum
  - adjustable delta (in a window)
- Adam
  - averaged momentum
  - bias correction



Examples in Codes

- Go to Keras :-)

Hw3

- Using Keras to train a classifier for ATIS intent classification
- Due at Monday noon
- Define your own model
- Using the provided Training and Test data
- File format, each line has three parts
  - words tokenized + EOS
  - tab delimiter
  - tags of each word and the intent

Hw3

- Try two input embeddings
  - The GloVe one you trained in Hw2
  - Onehot through an Embedding layer
- Submit a model.py and a readme.txt
  - model.py: the code file (should follow the course policy, compile and run)
  - readme.txt: describe what model structure and hyper-parameters you use and the result analysis on different input embeddings
- Send the two files to me via Email or Canvas mail