Named Entity Recognition (NER) with Deep Learning

Shobeir Fakhraei
University of Southern California

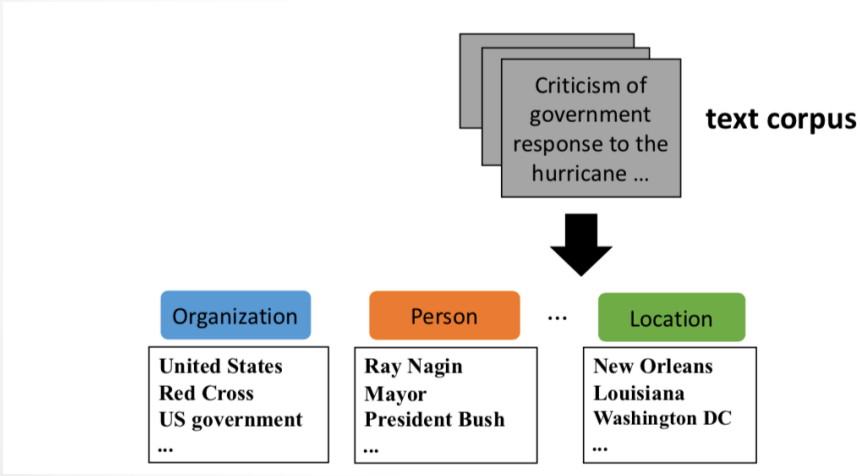
Slides based on material provided by many people!

Where are we?

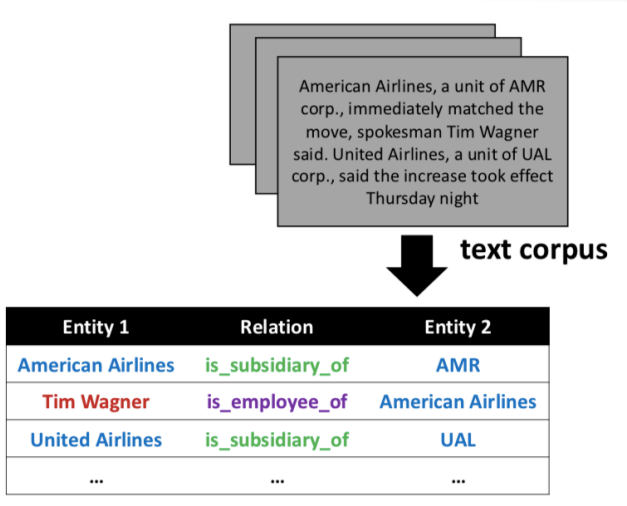
- Information Extraction
  - Semi-structured: Web, HTML
  - **Unstructured: Text, Adds, Tweets, …**
- Entity Linkage, Data Cleaning, Normalization
- Logical Data Integration
  - Mediators, Query Rewriting
  - Warehouse, Logical Data Exchange
- Automatic Source Modeling/Learning Schema Mappings
- Semantic Web
  - RDF, SPARQL, OWL, Linked Data
- Advanced Topics
  - Geospatial Data Integration, Knowledge Graphs

Landscape of IE Tasks (1):
Types of Extractions

Named Entity Recognition



Relationship Extraction



NER Approaches

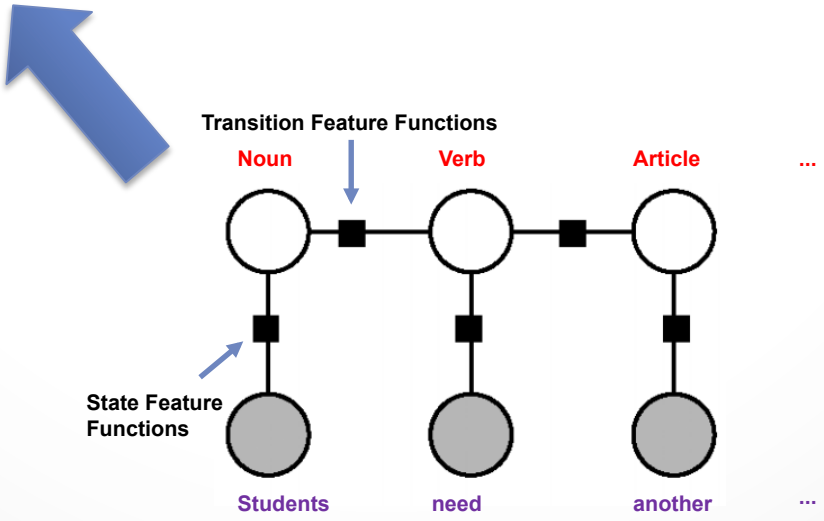
1. Rule/Pattern-Based
2. Standard Classifiers
   - KNN, decision tree, naïve Bayes, SVM, ...
3. Sequence Models
   - HMMs, CRFs, LSTM-CRF

NER Approaches

1. Rule/Pattern-Based
2. Standard Classifiers

   KNN, decision tree, naïve Bayes, SVM, ...
3. Sequence Models

   HMMs, CRFs, LSTM-CRF

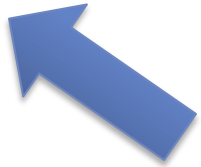


NER Approaches

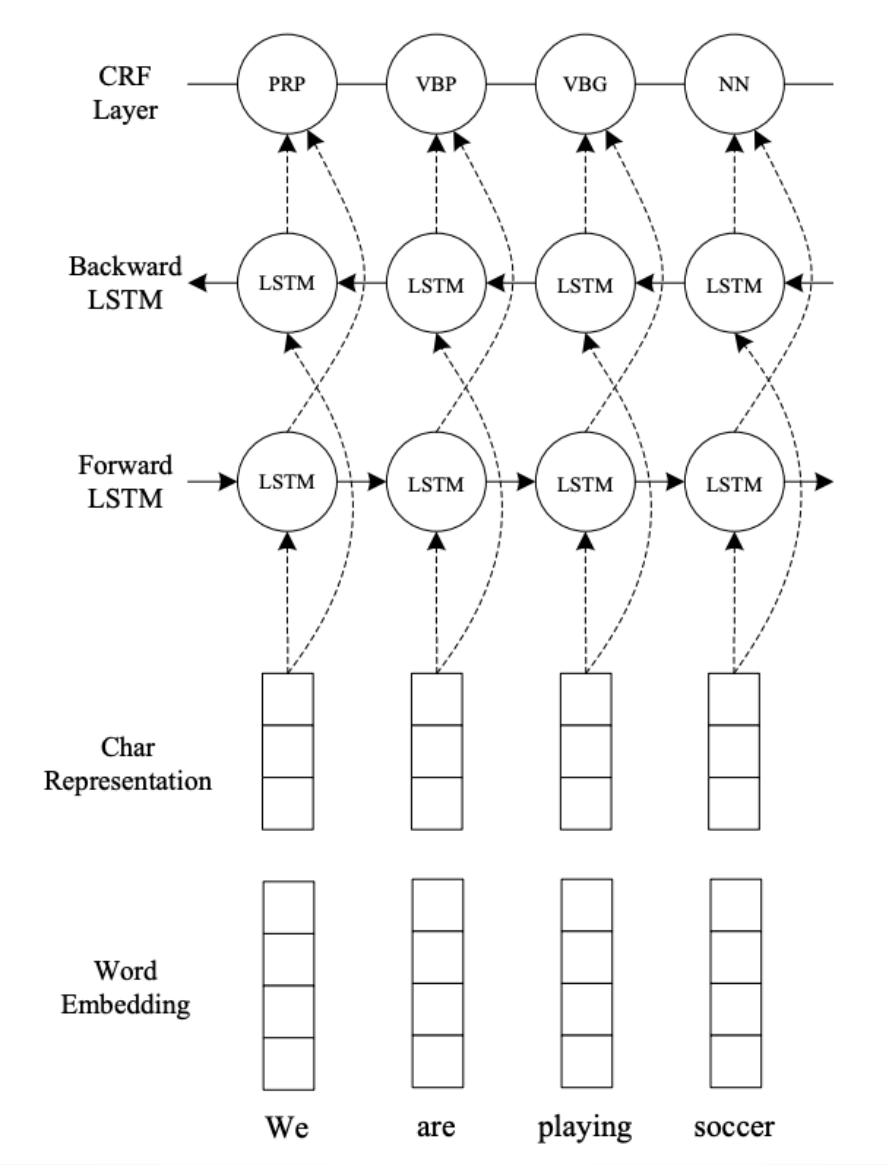
1. Rule/Pattern-Based
2. Standard Classifiers

   KNN, decision tree, naïve Bayes, SVM, ...
3. Sequence Models

   HMMs, CRFs, LSTM-CRF



Bi-directional LSTM-CNN-CRF

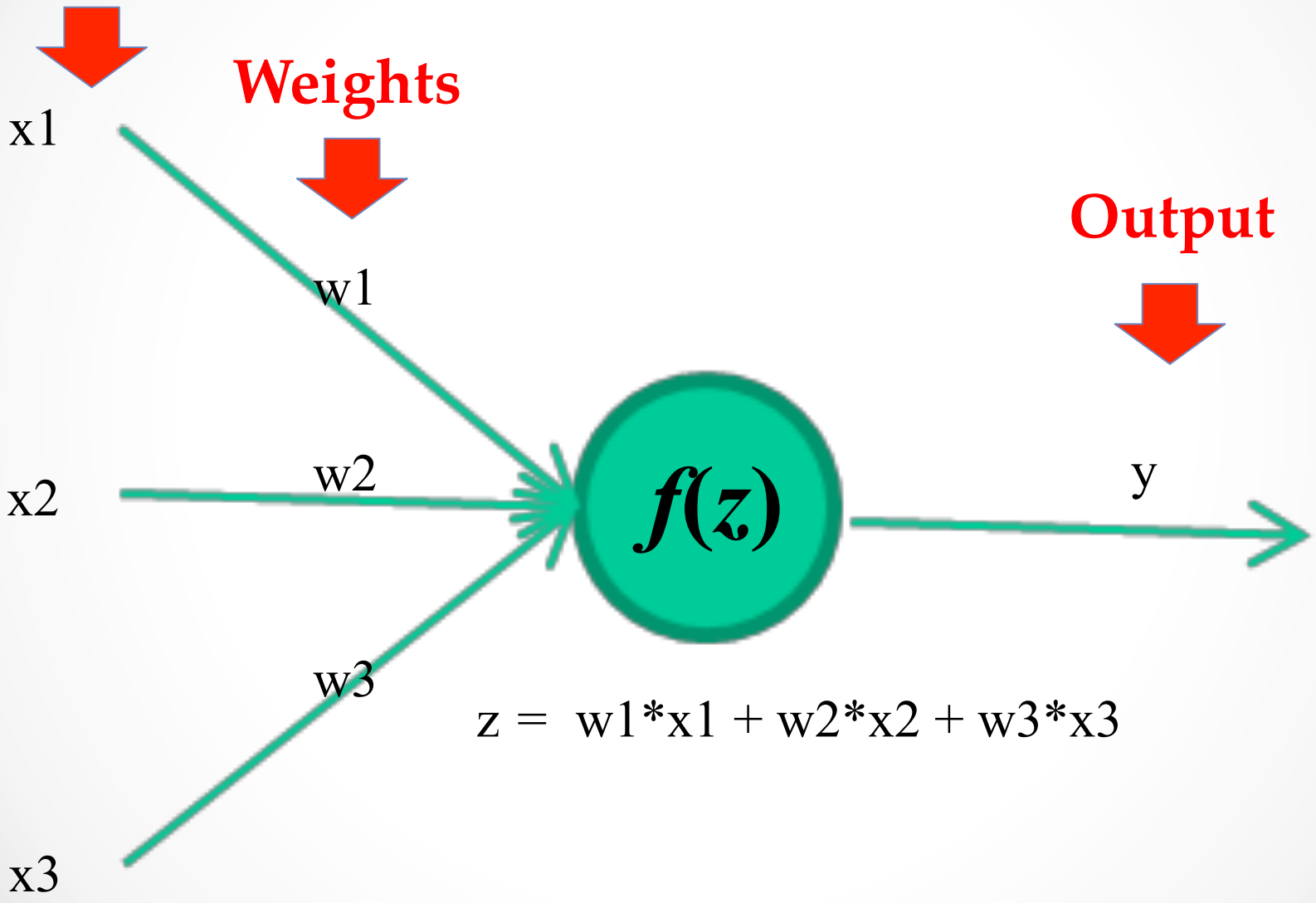


Outline

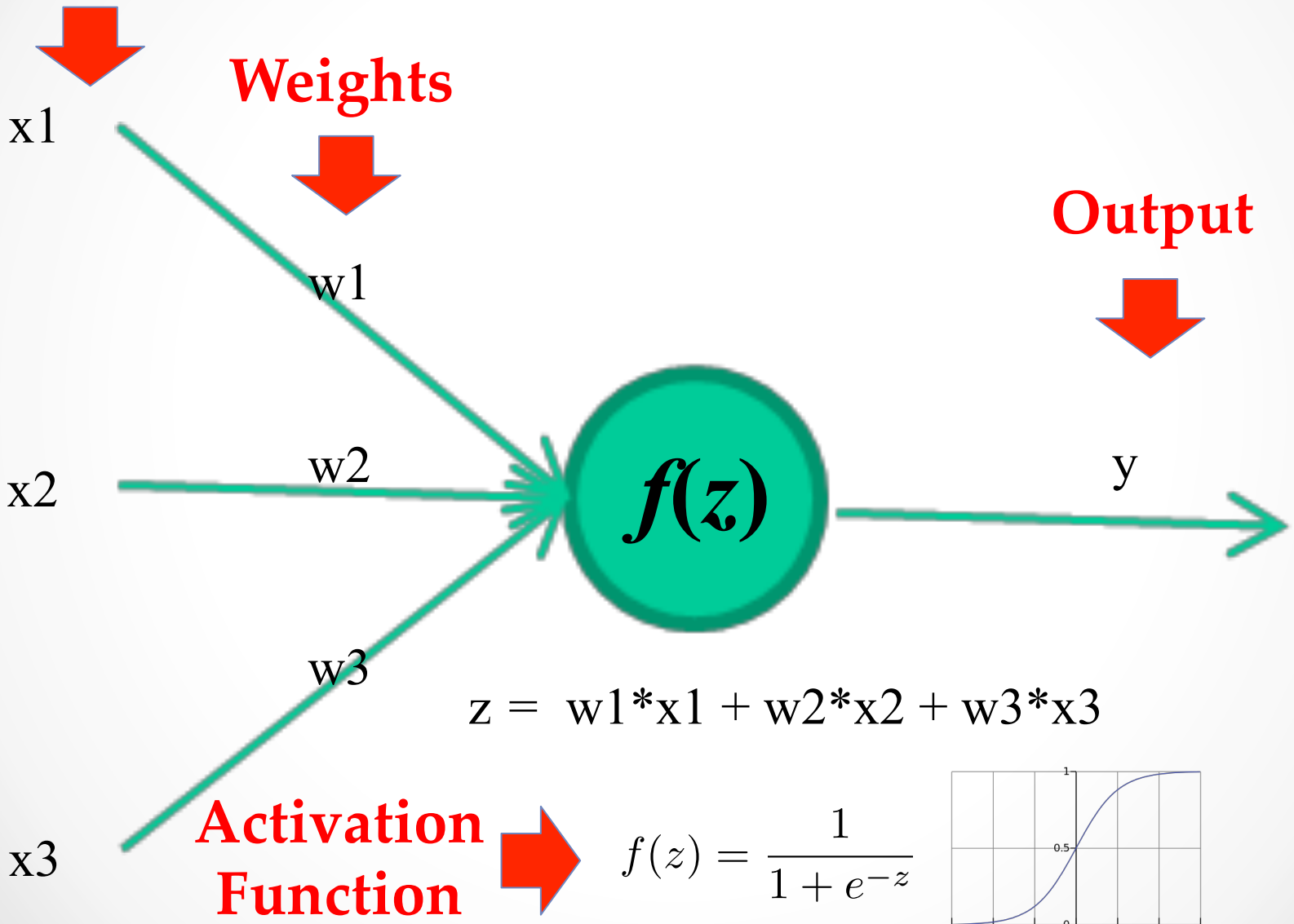
- Deep Learning Background
  - Activation functions, Back propagation, Dropout, CNN
- Sequence Models
  - Recurrent Neural Networks (RNN)
  - Long Short Term Memory (LSTM)
- Word and Character Embedding
  - Word2Vec, GloV
- NER Models with Deep Learning
  - CNN-BiLSTM-CRF

Background: Neural Networks

A neuron is just function!



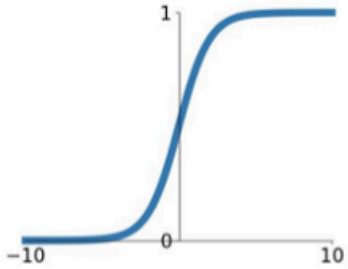
A neuron is just function!



Activation Functions

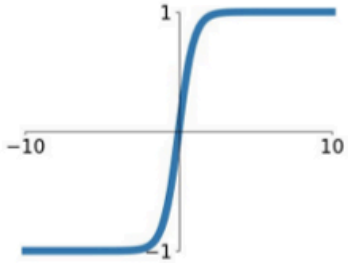
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



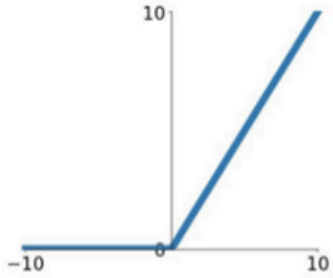
tanh

$$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

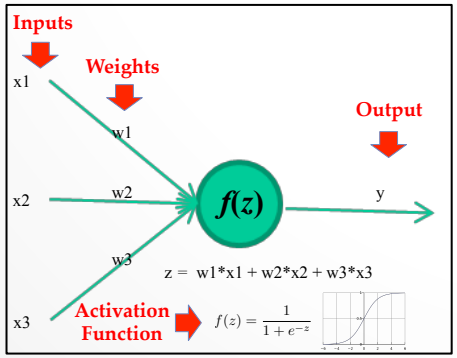


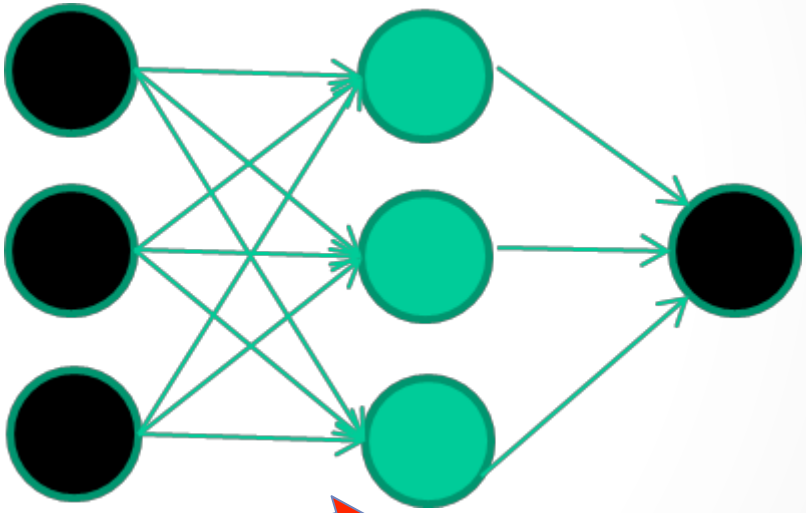
ReLU

$$\max(0, x)$$



Artificial Neural Network



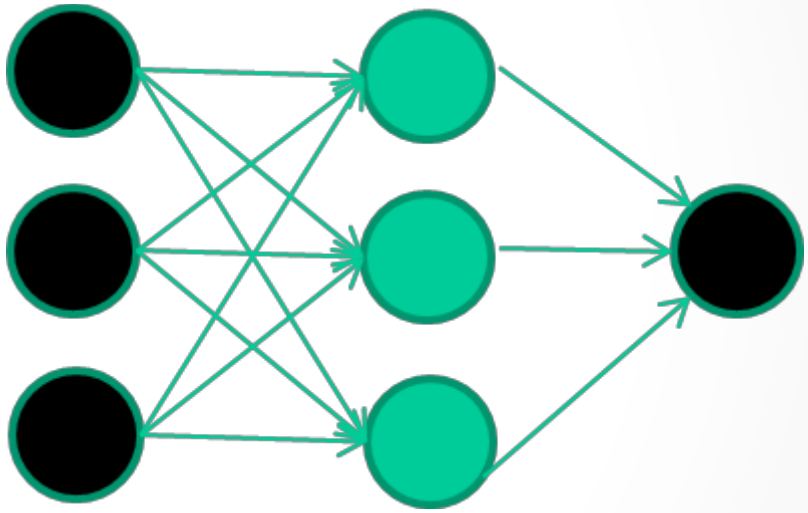


Training ANN

A dataset

Fields			*class*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

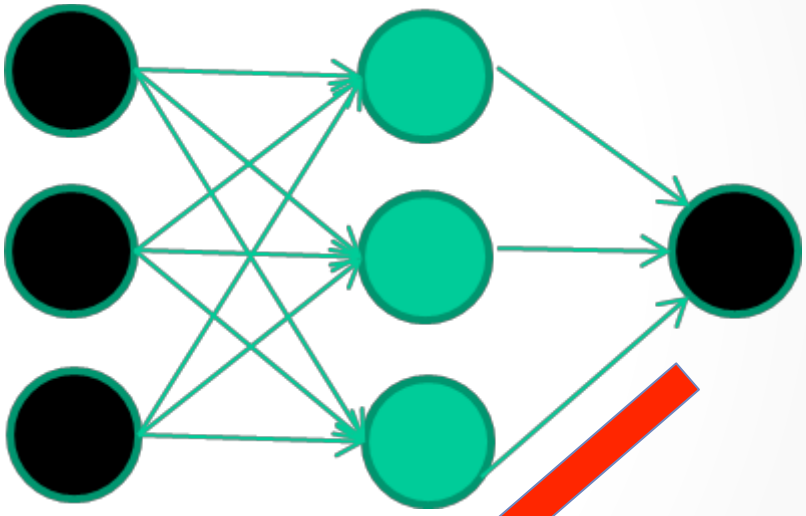


Training ANN

A dataset

Fields			*class*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …



Cost Function

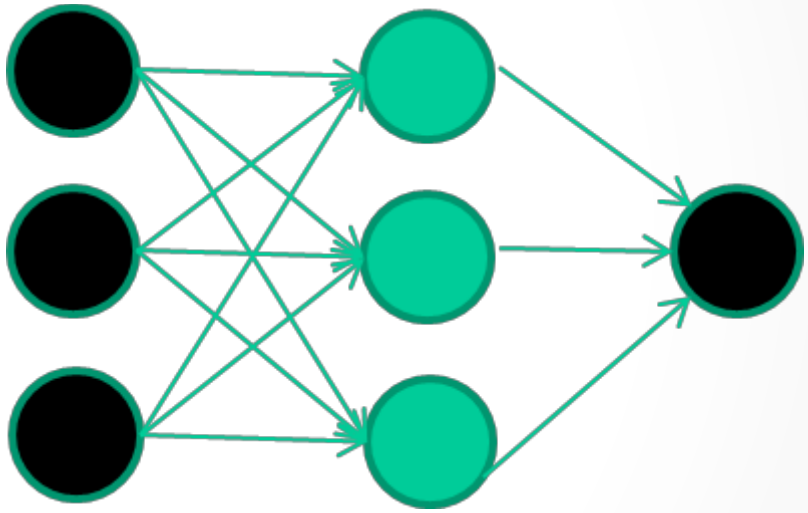
$$C = \frac{1}{n}\sum (y_t - y_p)^2$$

Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Initialise with random weights

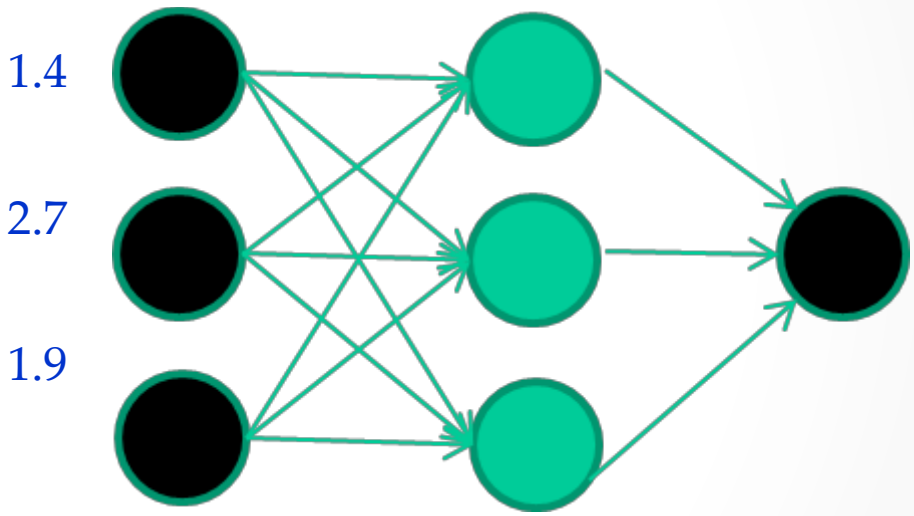


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Input a training instance

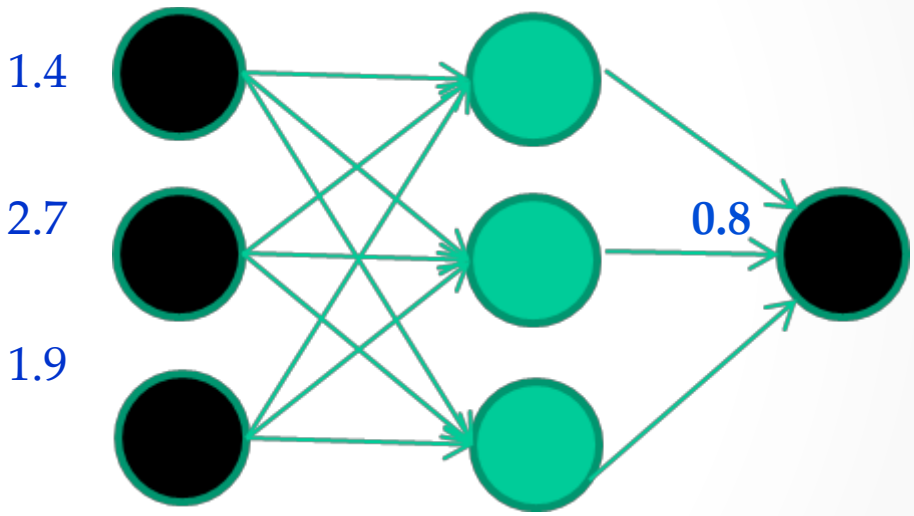


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output

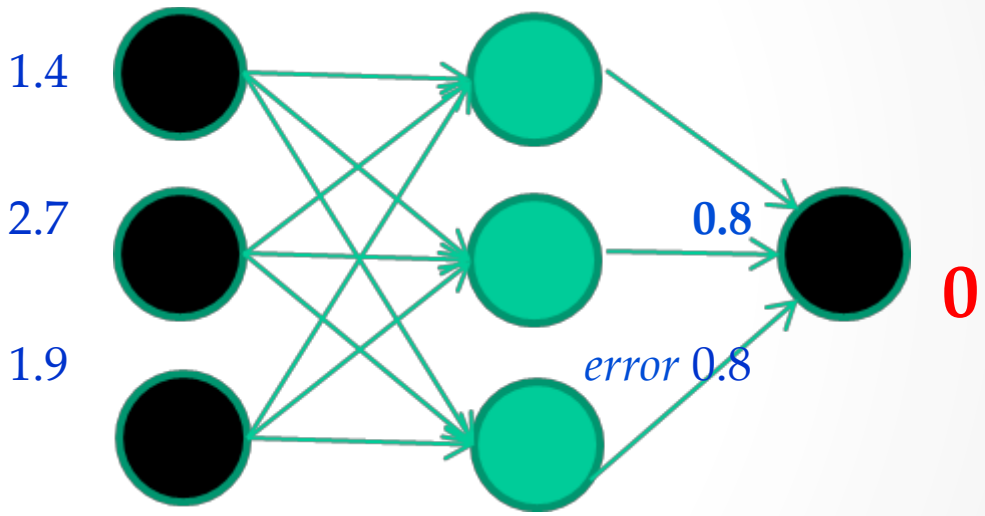


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output

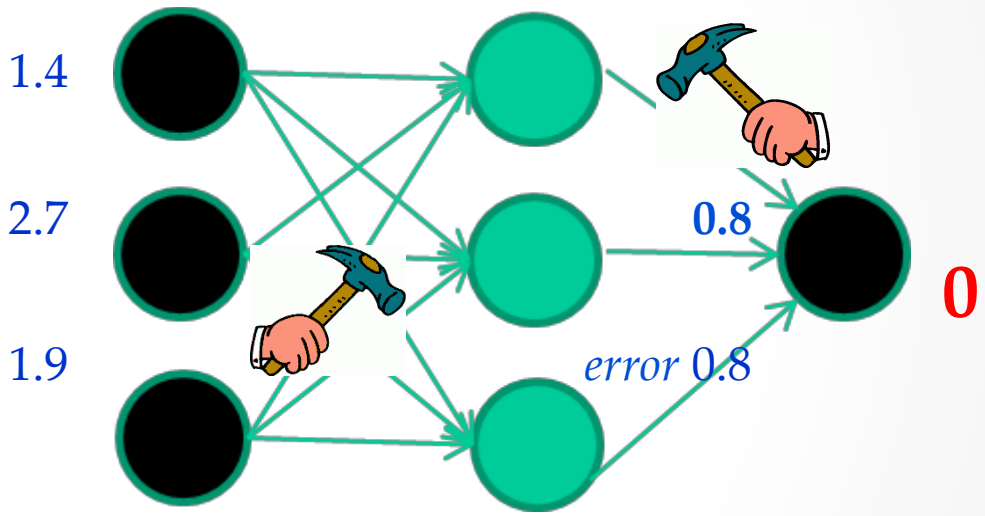


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

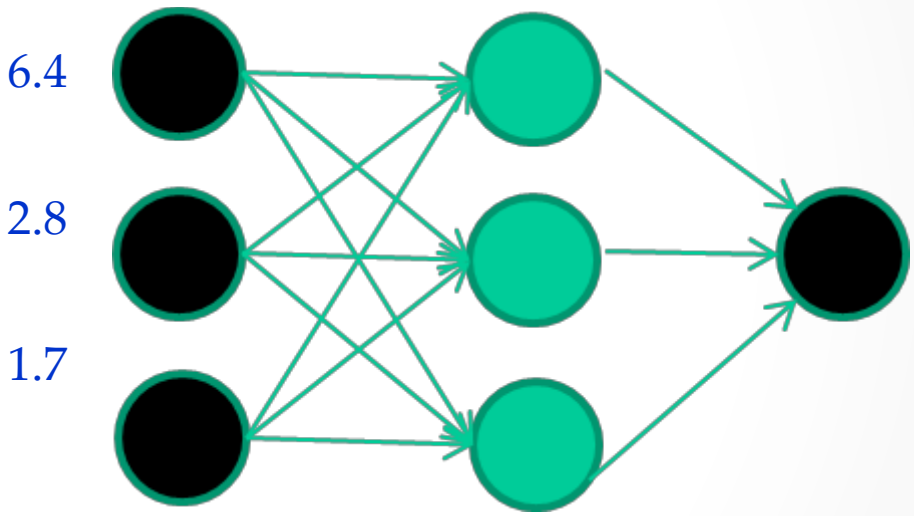


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Input a training example

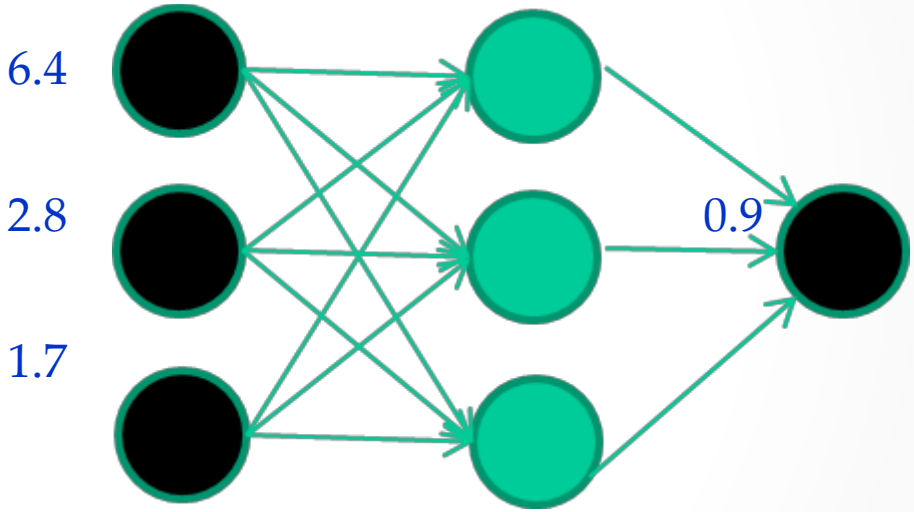


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output



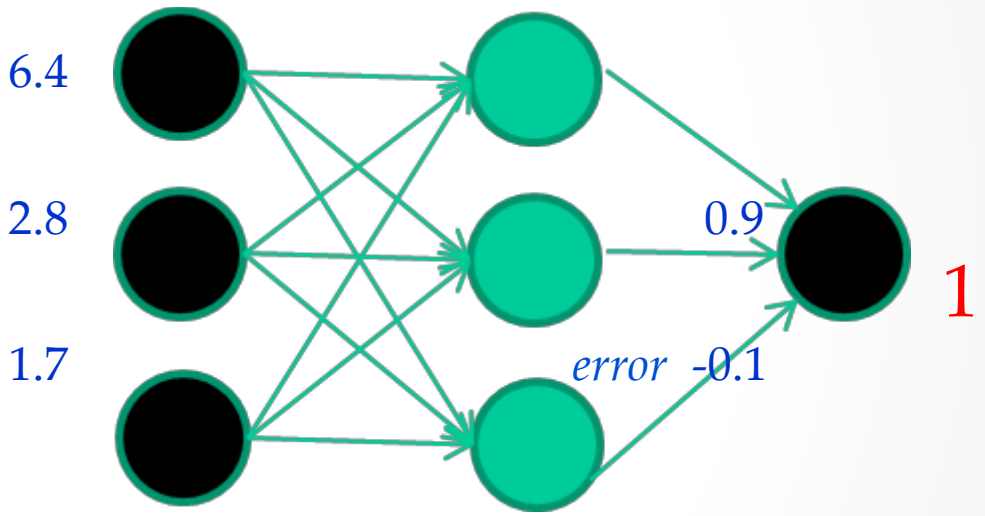
Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

Compare with target output

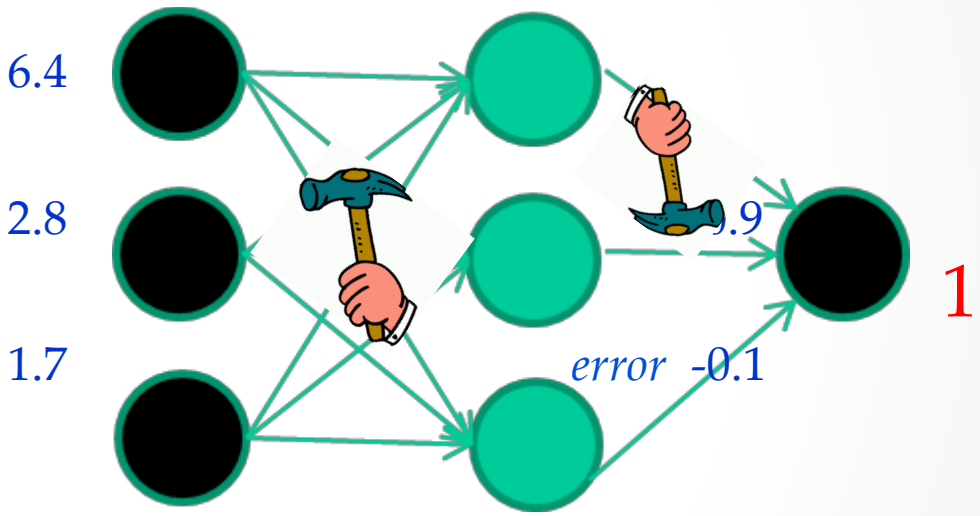


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

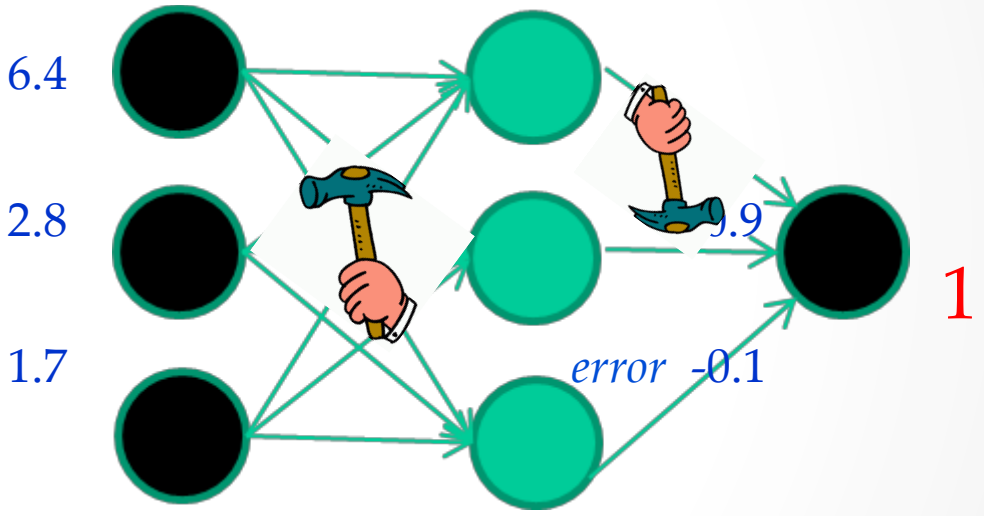


Training ANN

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

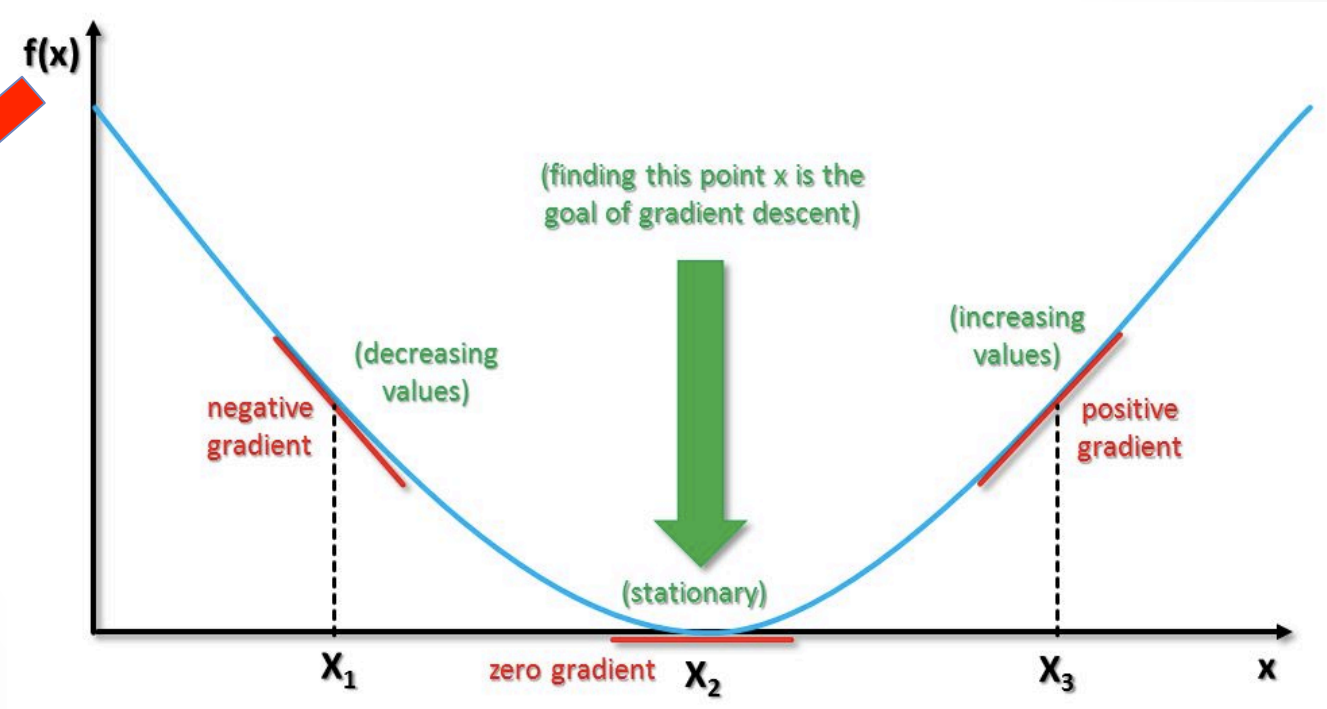
And so on ….



Repeat this thousands of times – each time taking a random training instance, and making slight weight adjustments

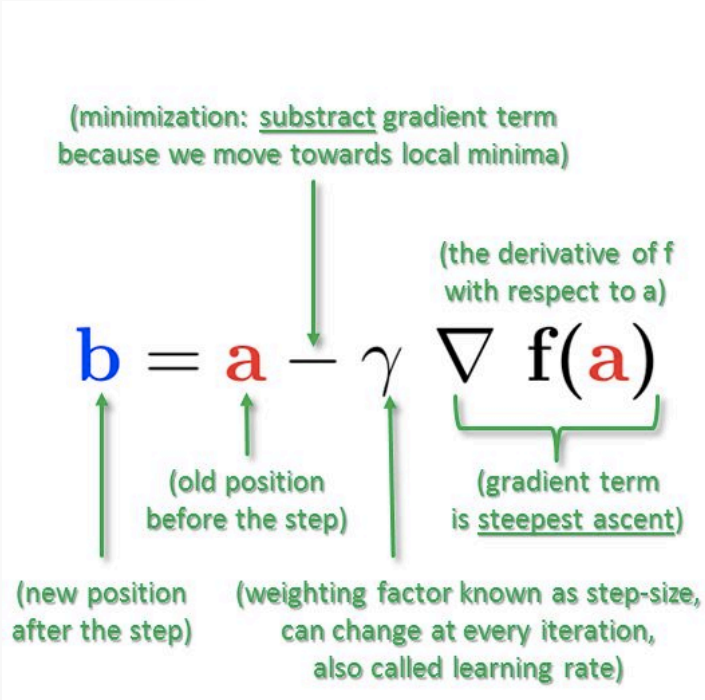
Gradient Descent

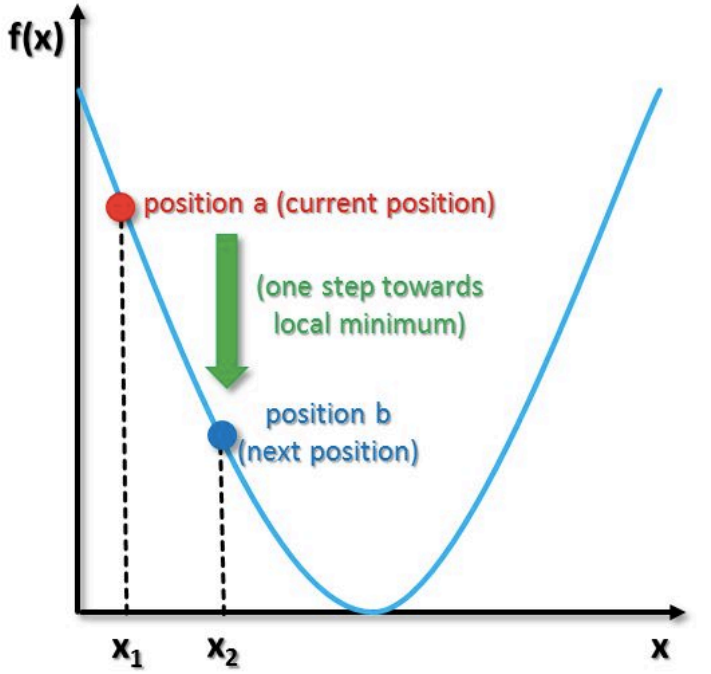
$$C = \frac{1}{n}\sum (y_t - y_p)^2$$



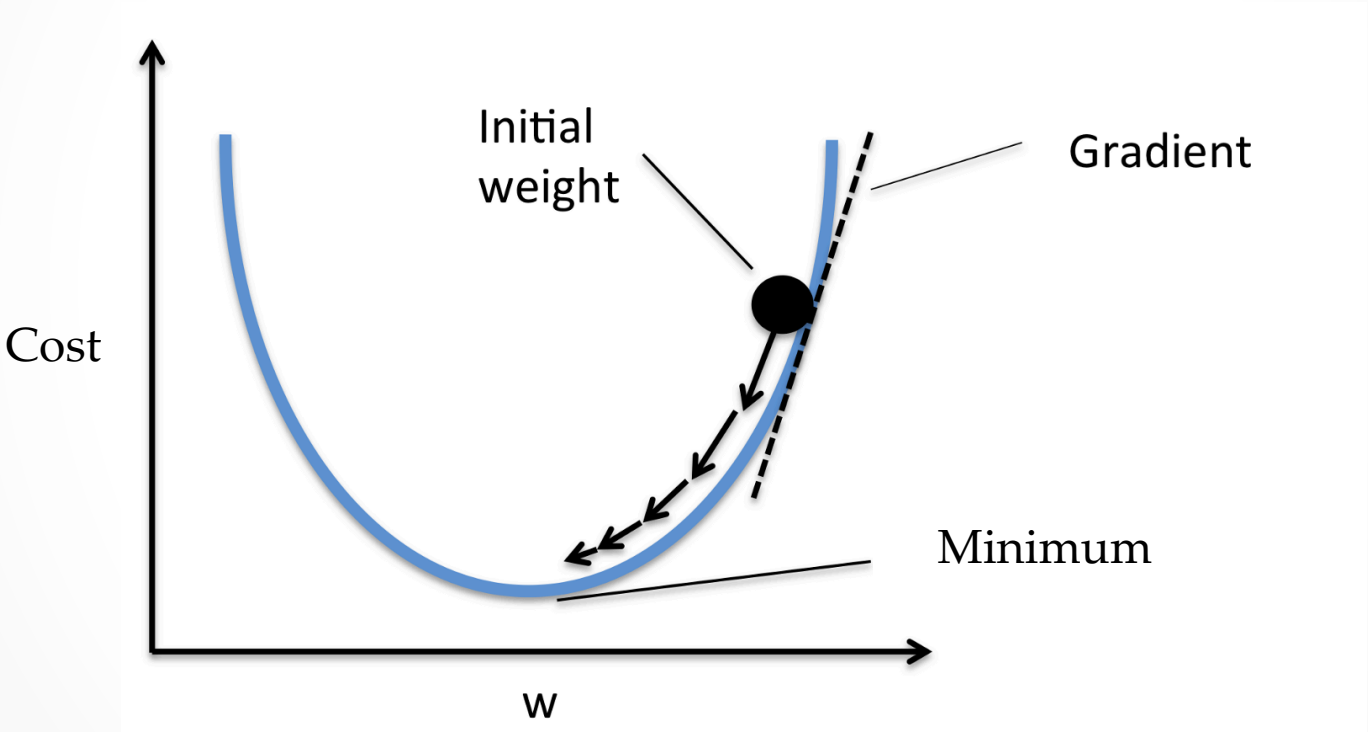
Weights (w)

Gradient Descent

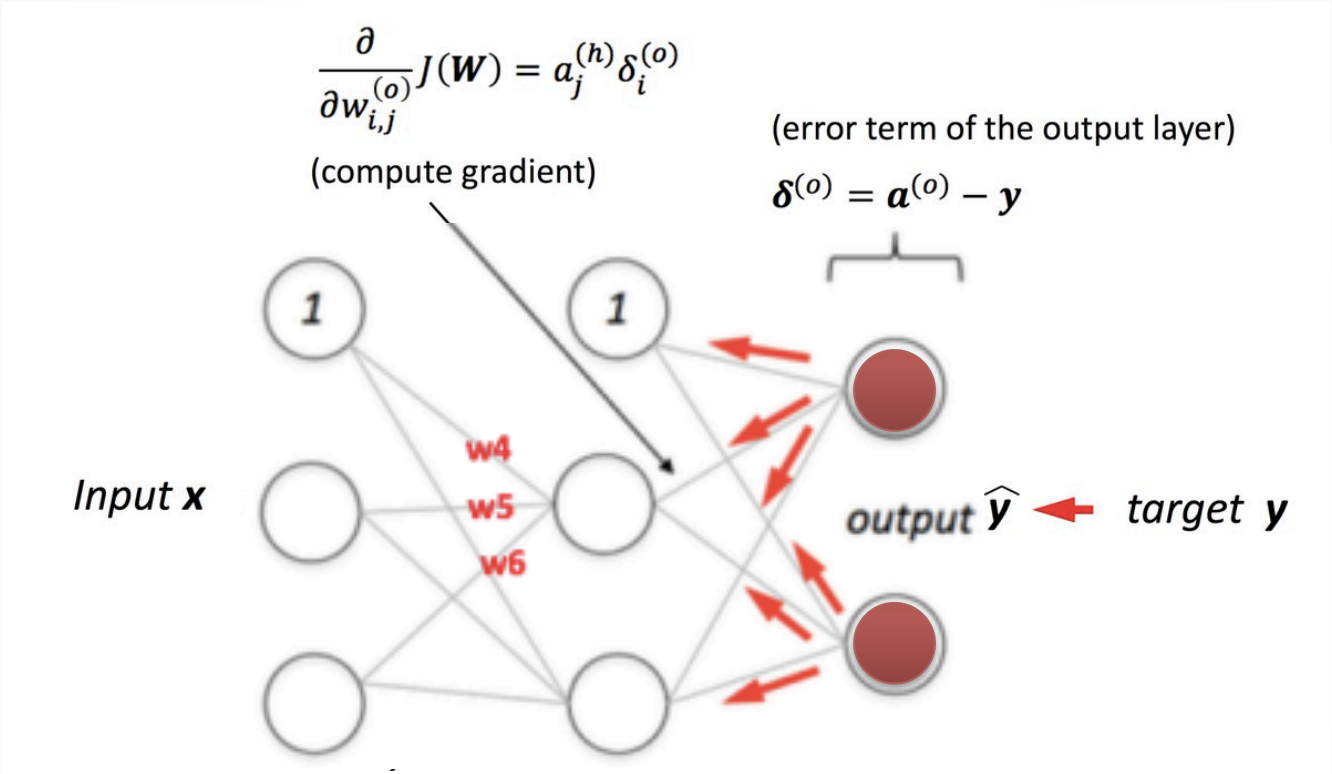




Gradient Descent

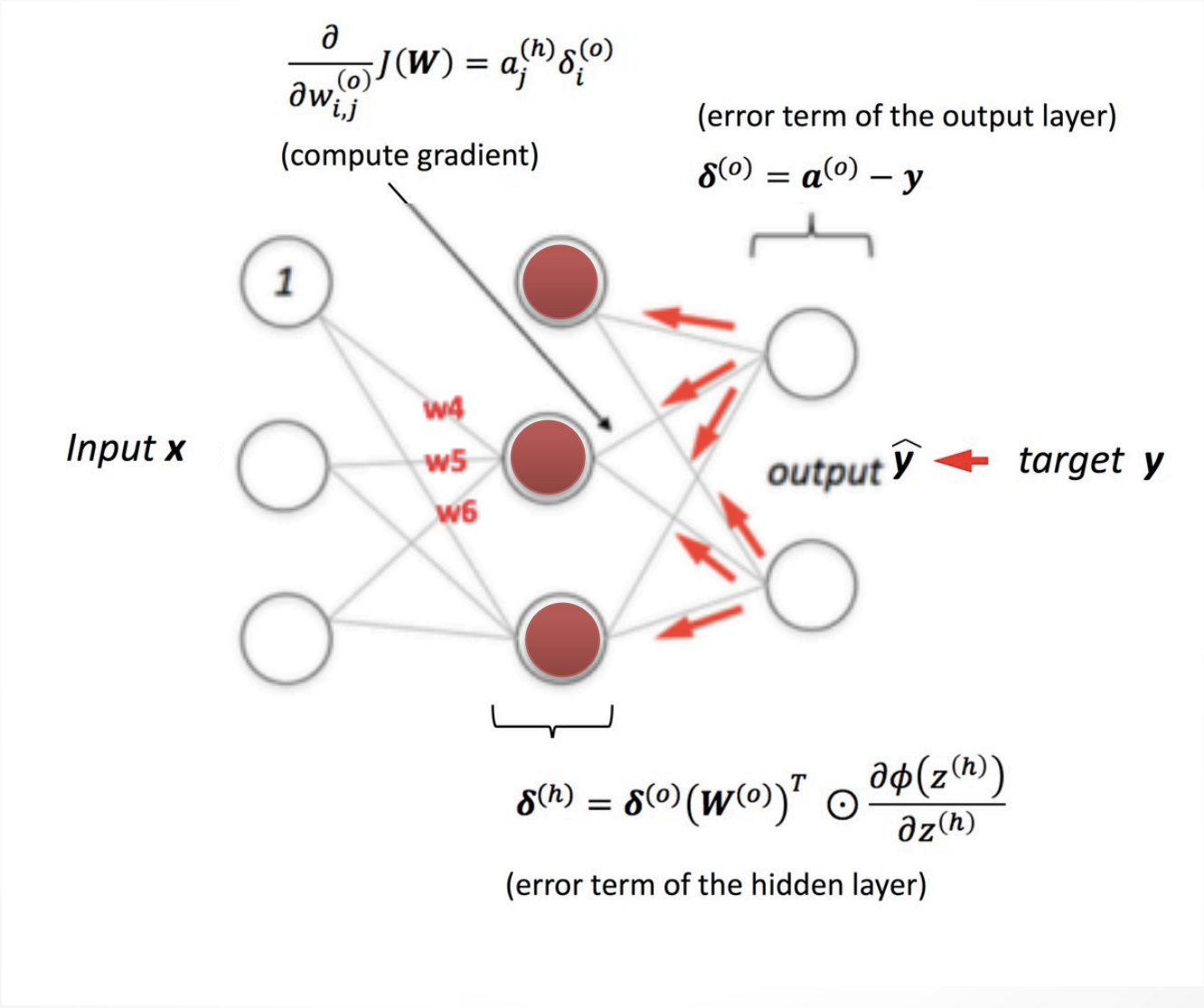


Back Propagation

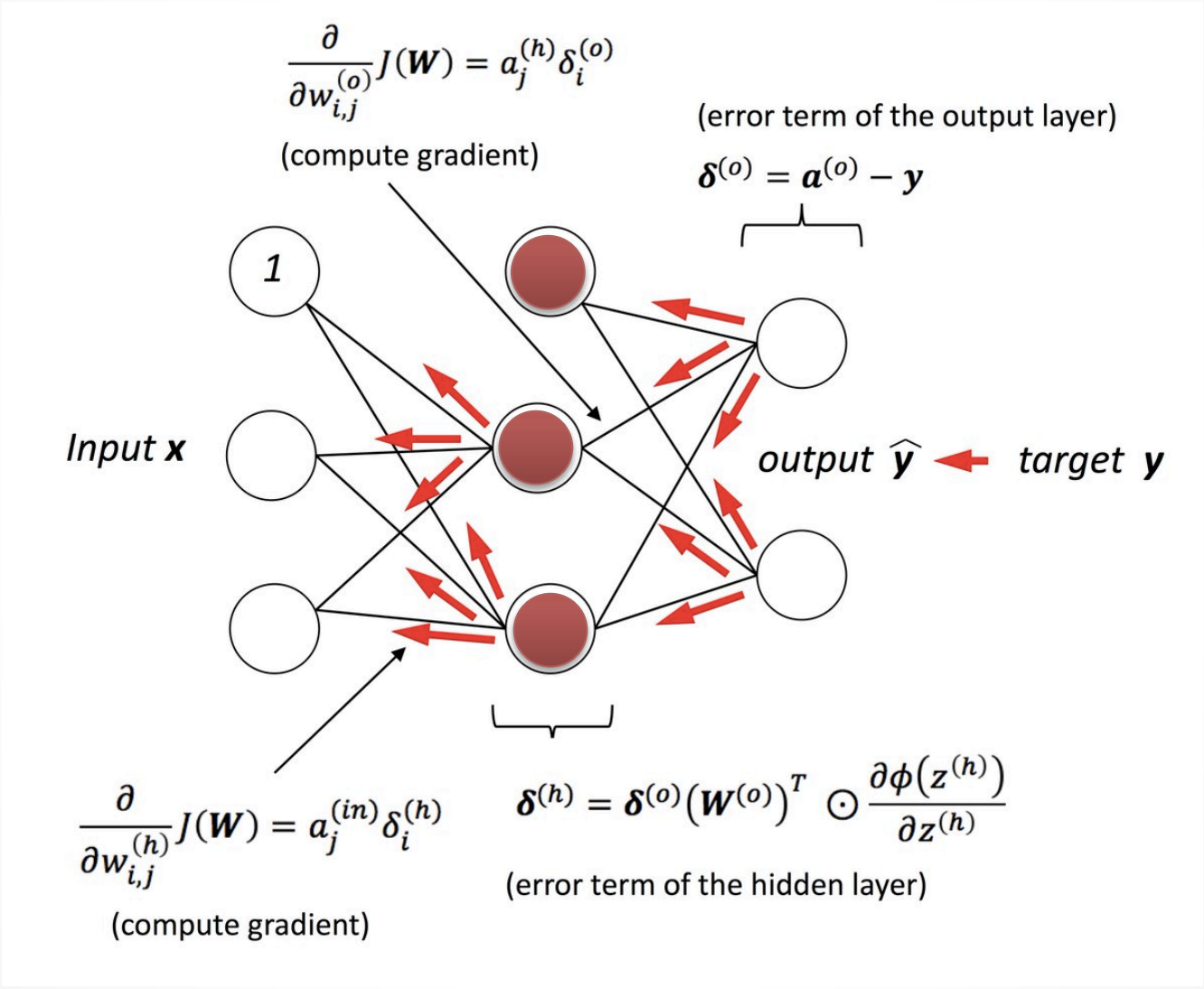


Demo:
https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/

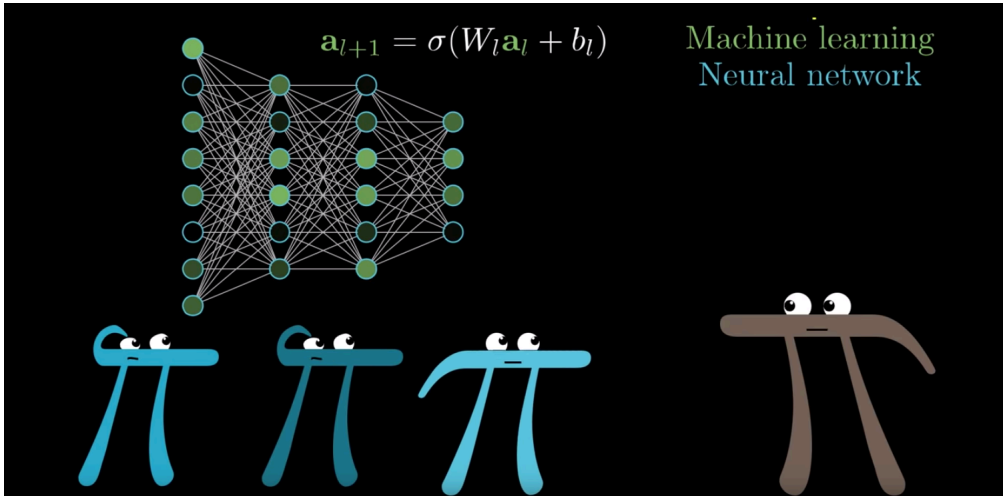
Back Propagation



Back Propagation



Watch the video series!



3BLUE1BROWN Deep Learning Video Series:
https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

One Perspective of Neural Networks History

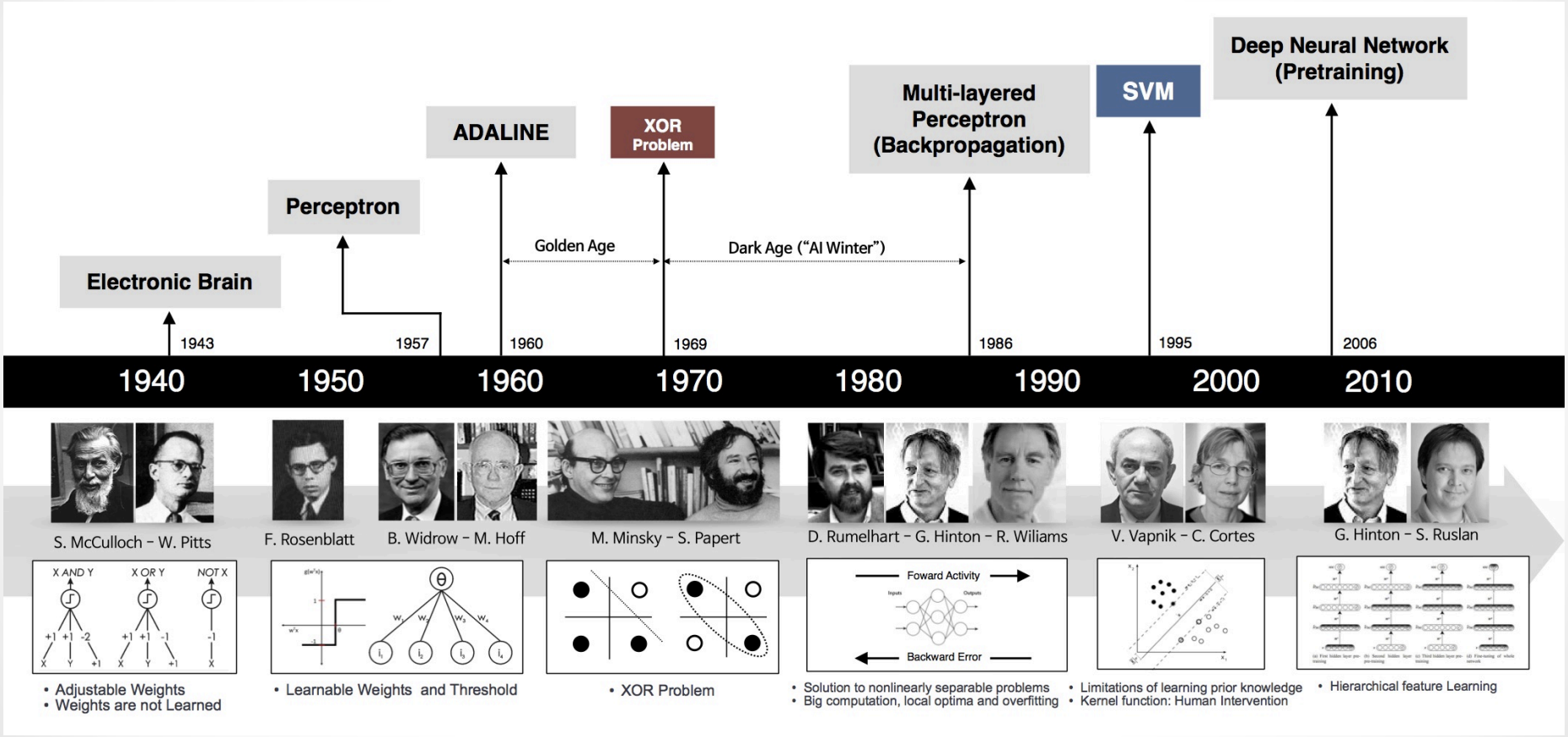


Figure by Andew Beam

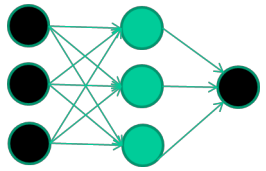
What's New?

Multilayer neural networks have been around for many years. What's actually new?

What's New?

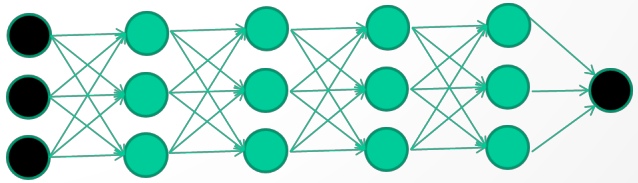
Multilayer neural networks have been around for many years. What's actually new?

we have always had good algorithms for learning the weights in networks with 1 hidden layer

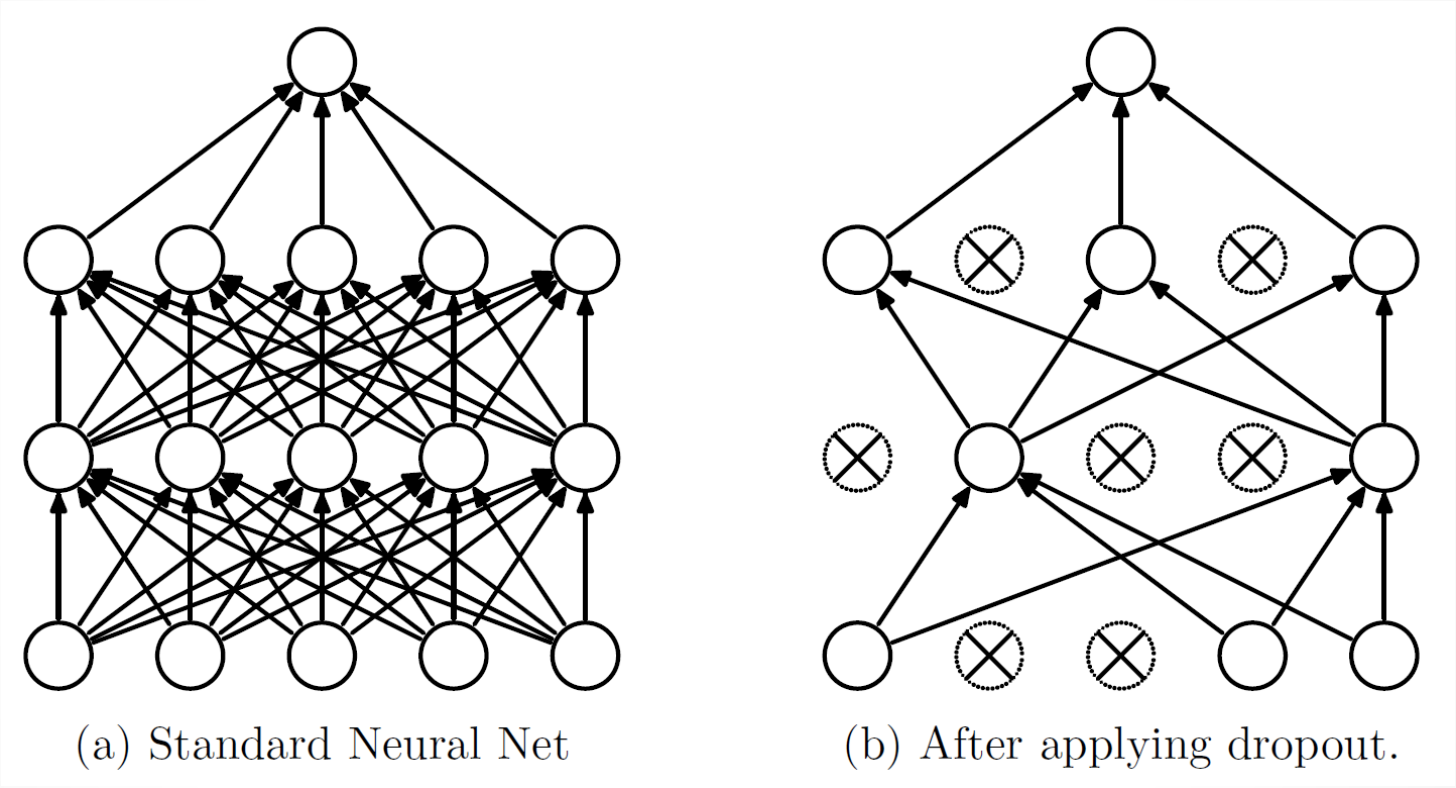


but these algorithms are not good at learning the weights for networks with more hidden layers

what's new is: algorithms for training many-layer (deep) networks



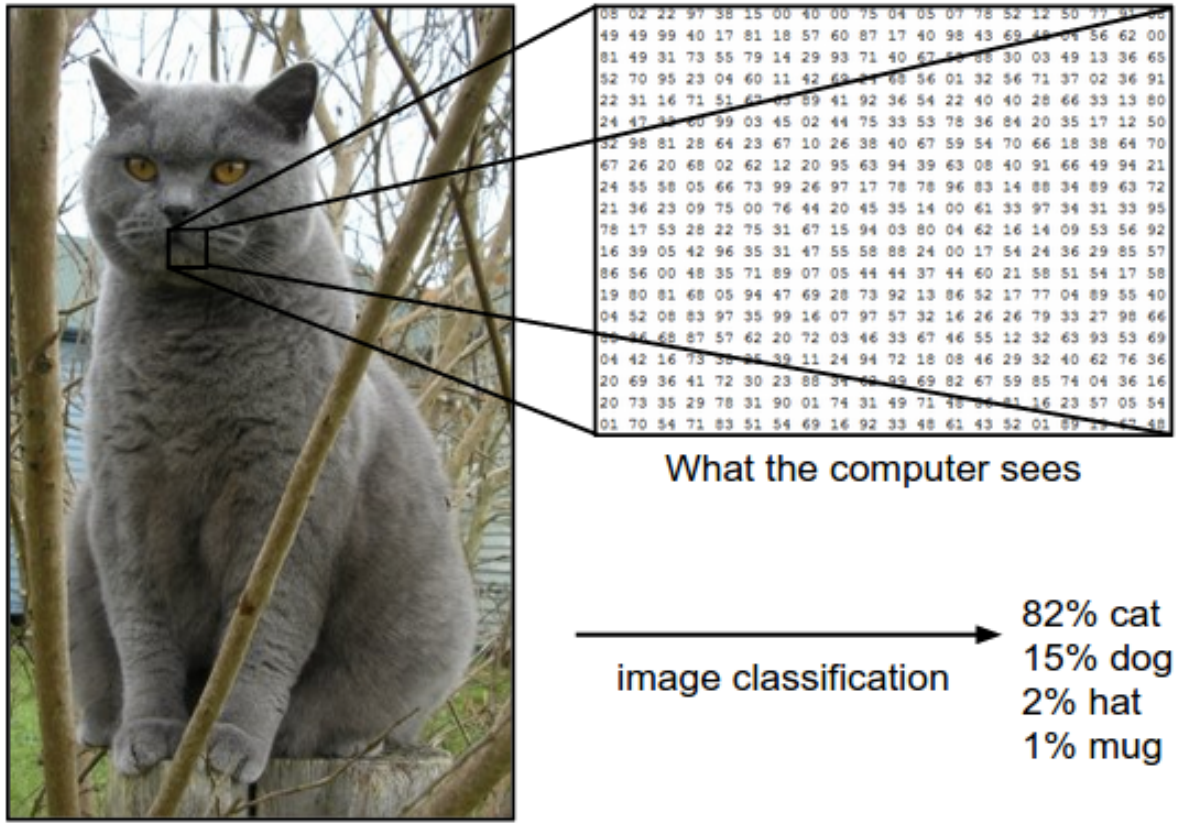
Dropout to avoid over-fitting



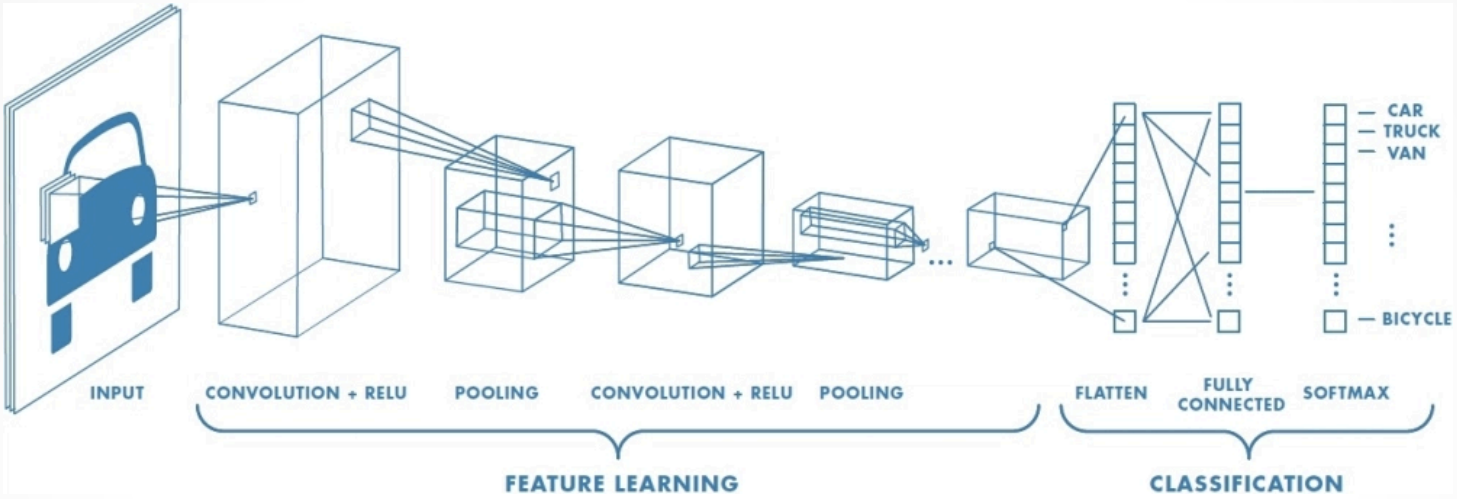
(a) Standard Neural Net

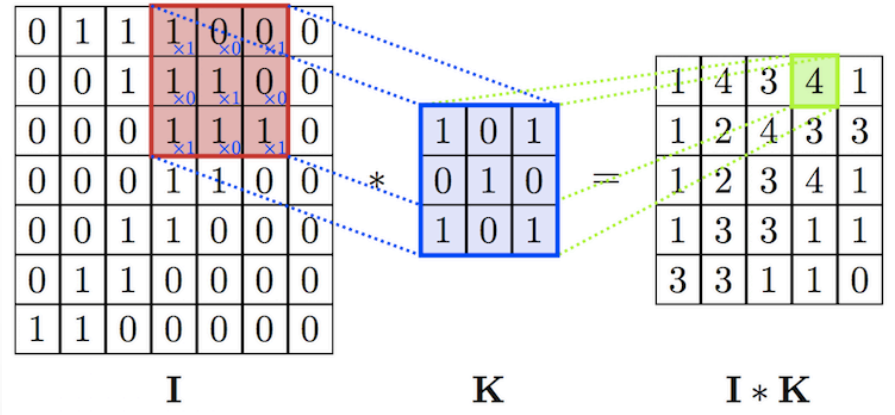
(b) After applying dropout.

Convolutional Neural Networks

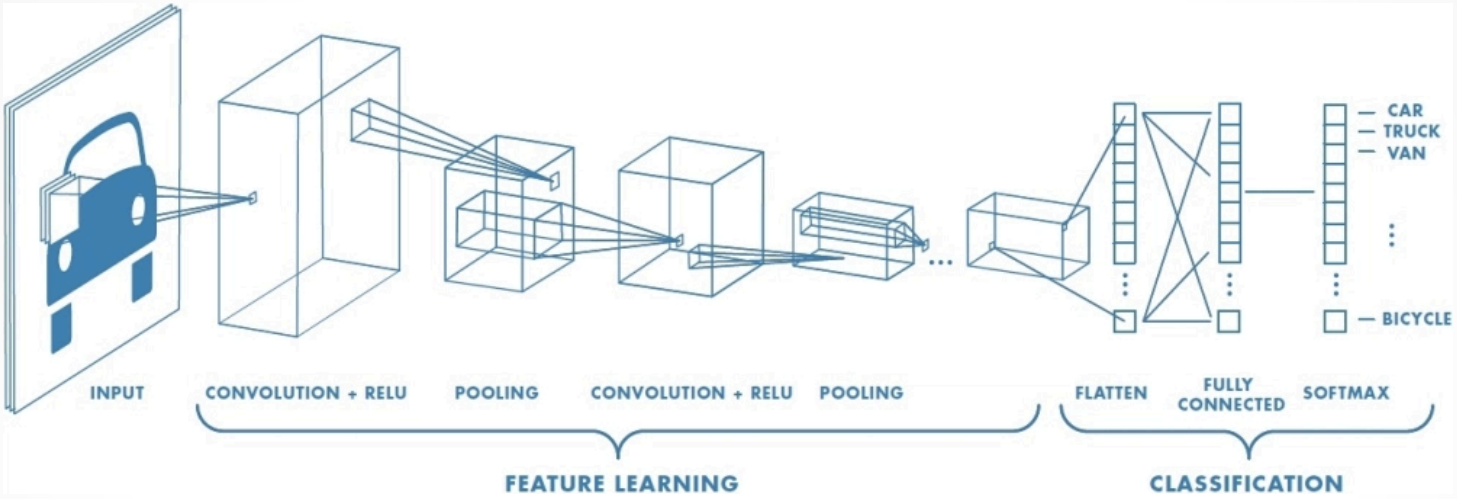


Convolutional Neural Networks





Convolutional Neural Networks

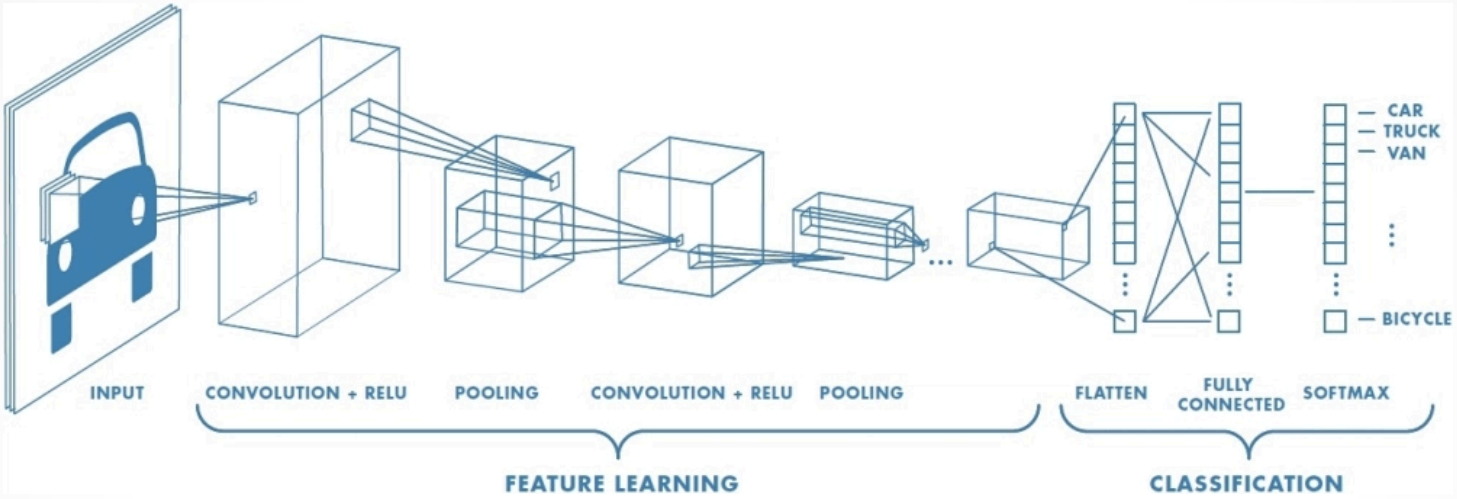


12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

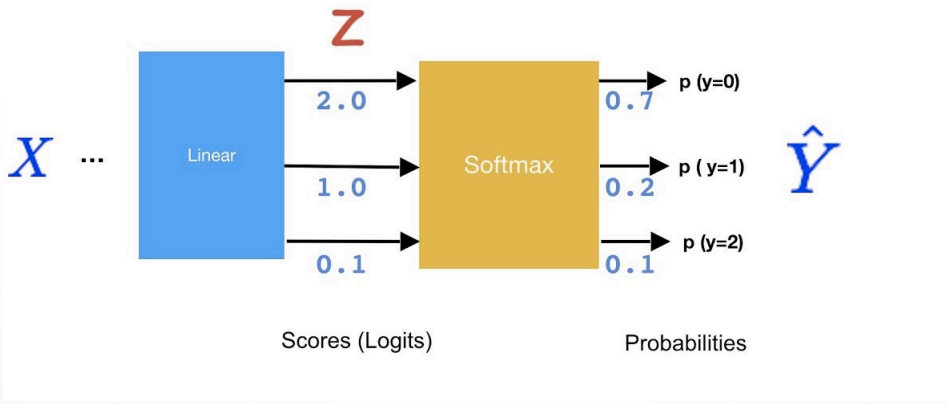
2×2 Max-Pool

20	30
112	37

Convolutional Neural Networks

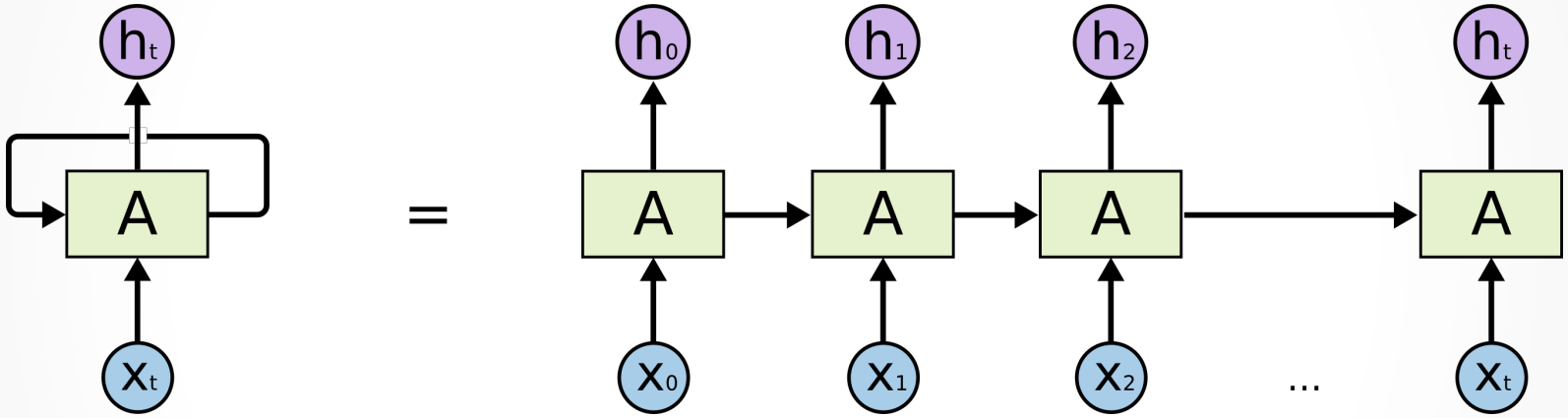


Meet Softmax $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.



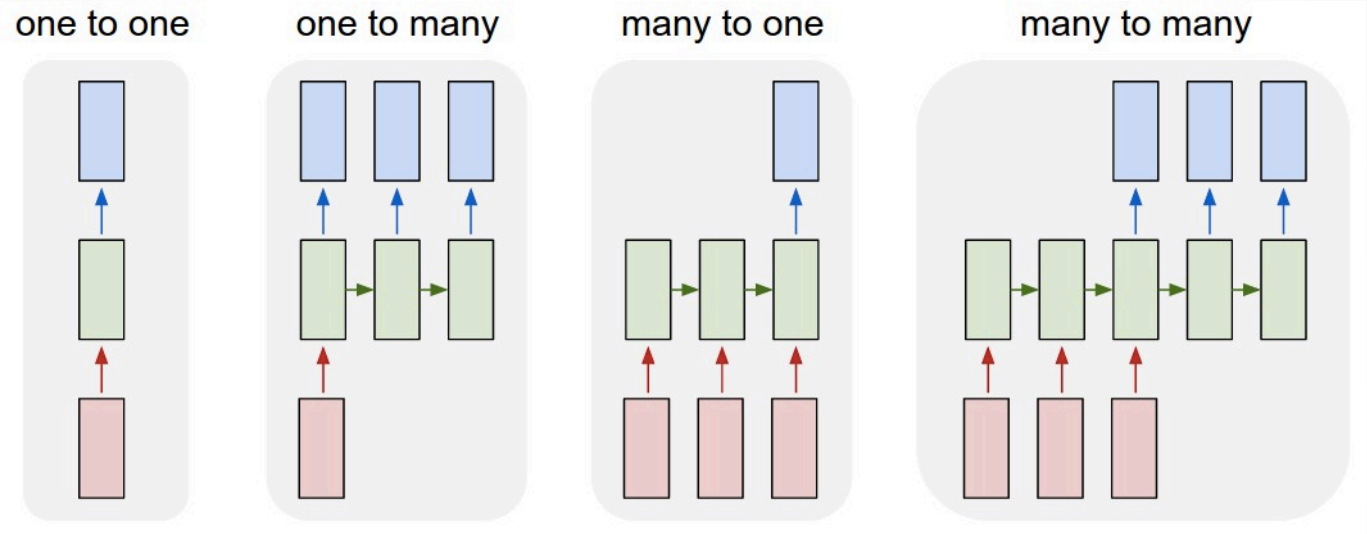
RNN and LSTM

Recurrent Neural Network (RNN)



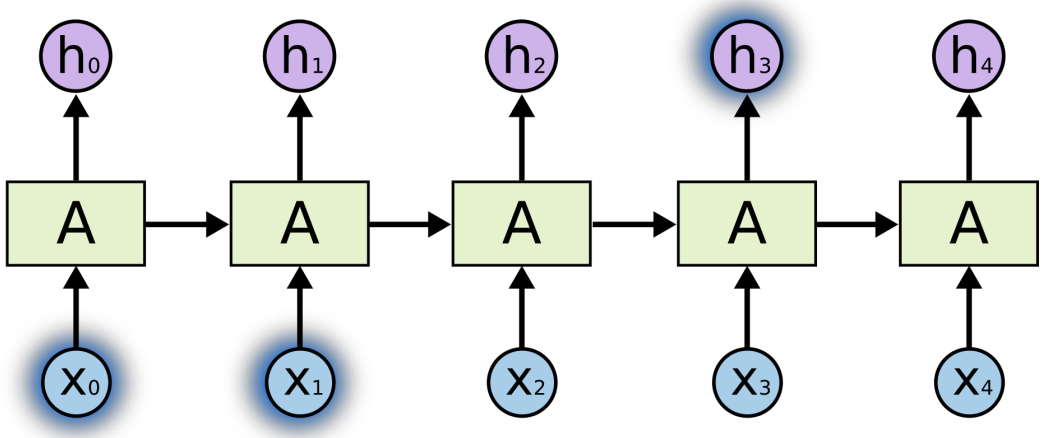
Humans don't start their thinking from scratch every time.
Why should machines?

Applications of RNN



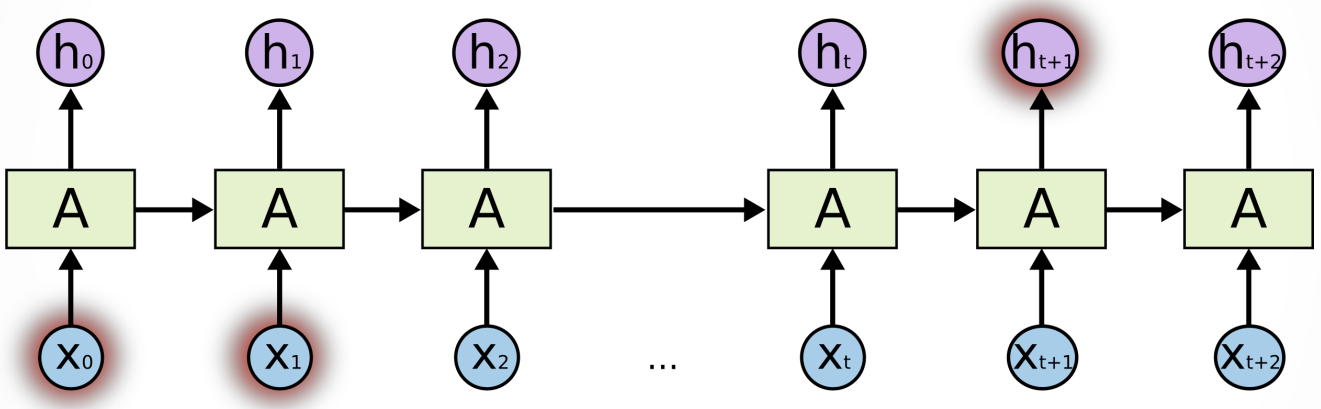
(1) Fixed-sized input to fixed-sized output
(e.g. image classification)
(2) Sequence output
(e.g. image captioning takes an image and outputs a sentence of words)
(3) Sequence input
(e.g. sentiment analysis)
(4) Sequence input and sequence output
(e.g. Machine Translation: a sentence in English to a sentence in French)

RNNs are good for short dependencies



The Clouds are in the *sky*

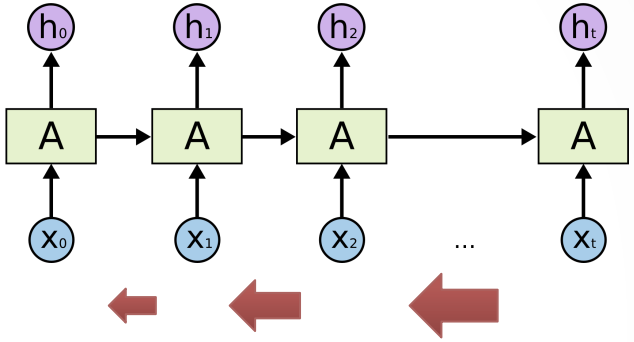
RNNs are problematic with longer dependencies



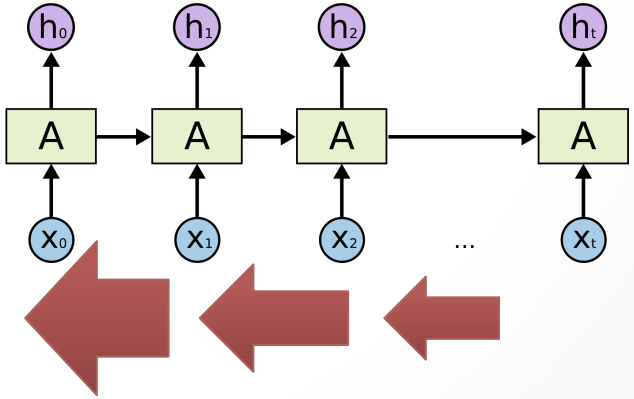
I grew up in France … I speak fluent *French*

Problems with RNNs

Vanishing Gradients



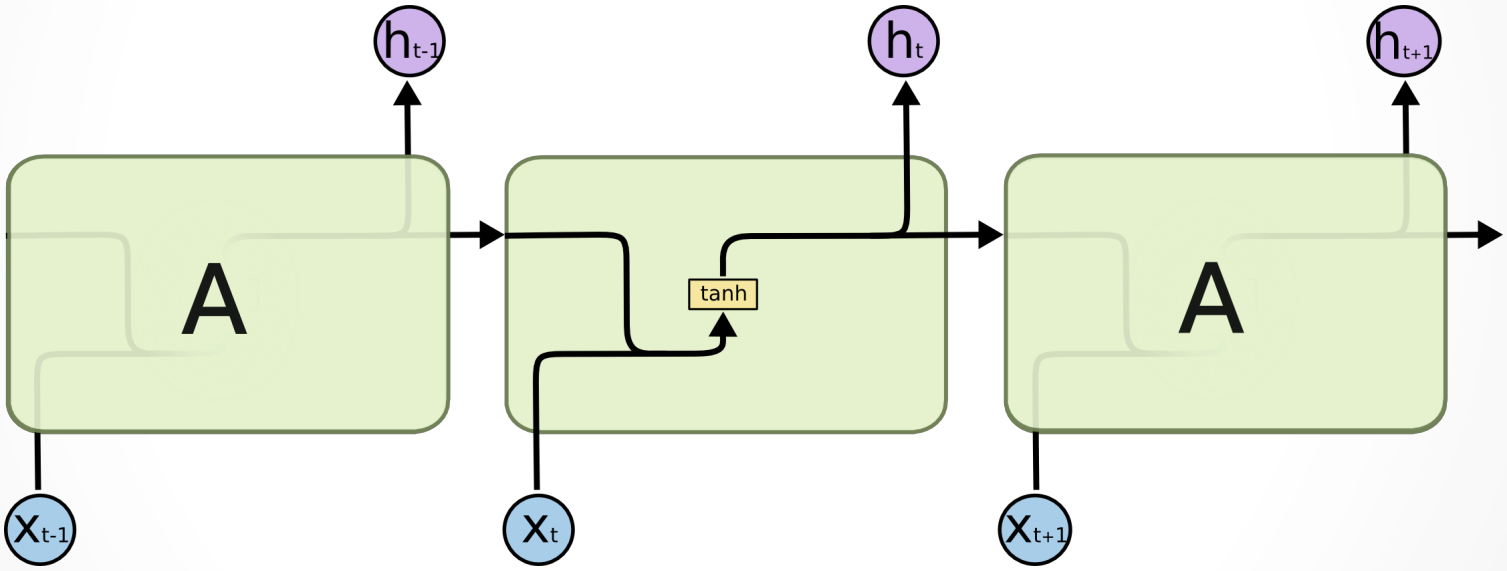
Exploding Gradients



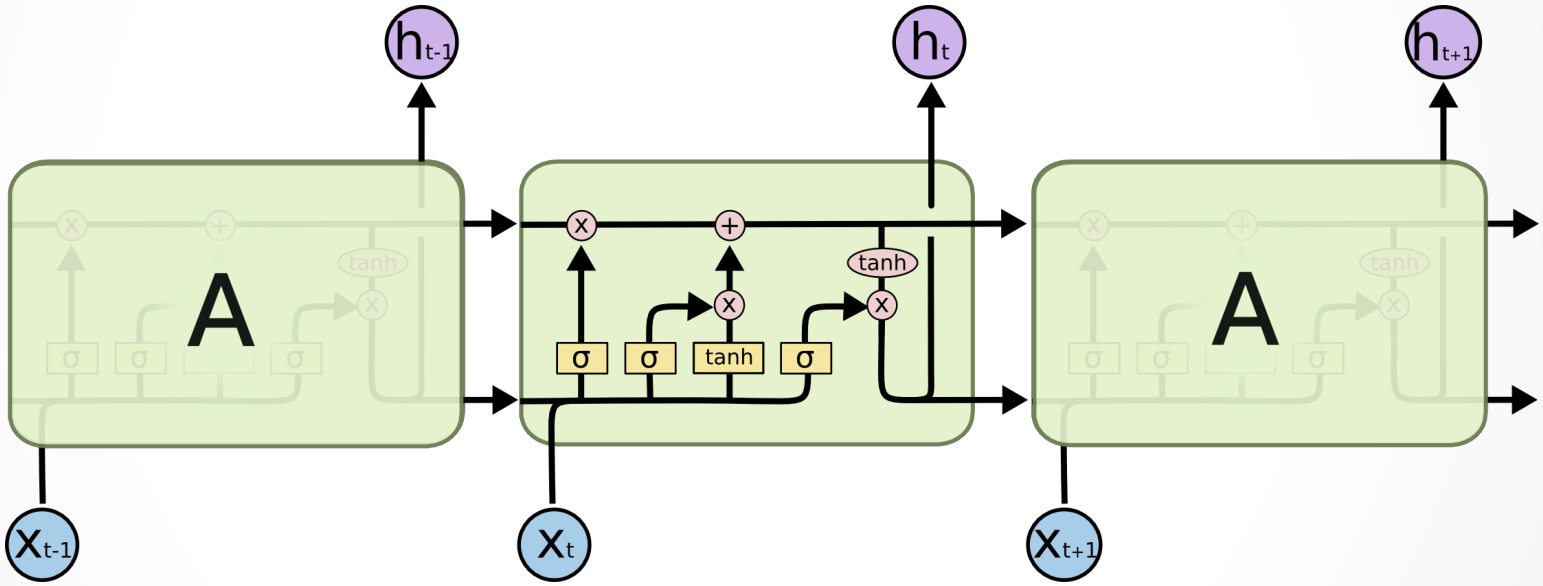
The Solution?

Long Short Term Memory networks (LSTM)

Inside RNN units

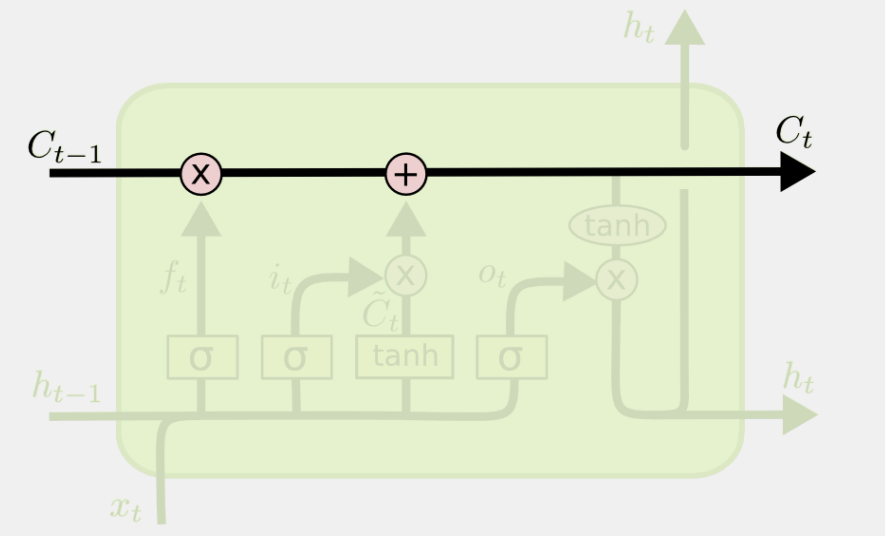


Inside LSTM units

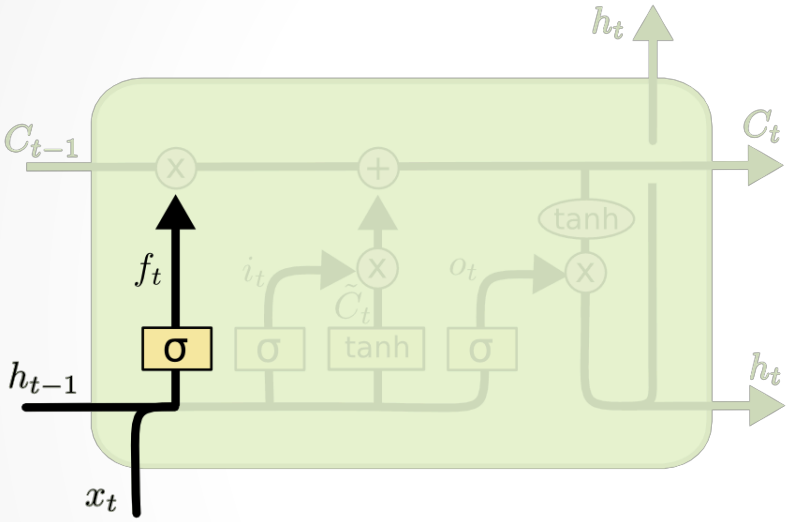


Gates to control adding and removing information

Cell State



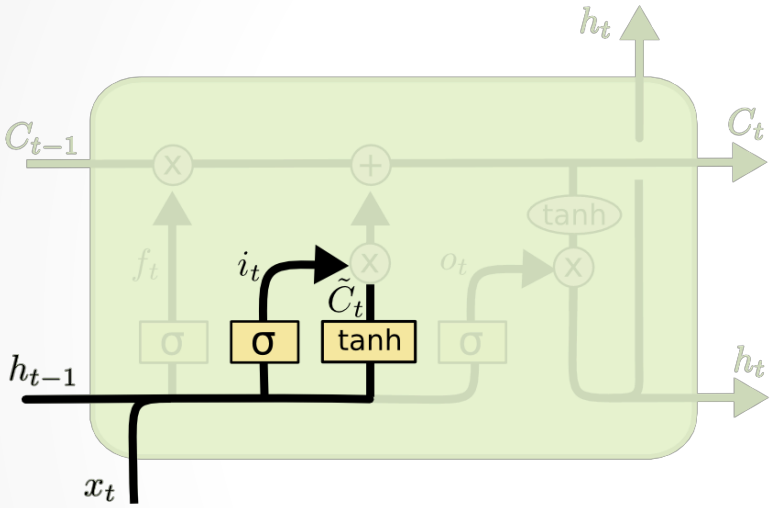
Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

e.g.,
The cell state might include the gender of the present subject, so that the correct pronouns can be used.
When we see a new subject, we want to forget the gender of the old subject.

Input Gate Layer

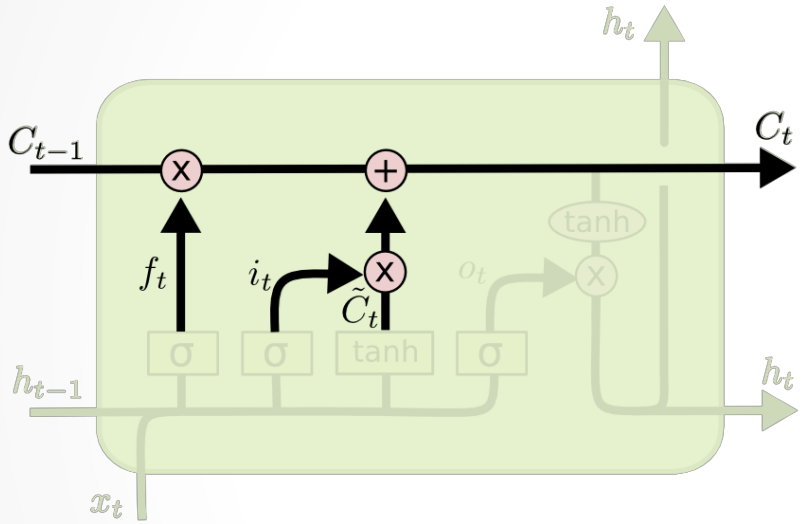


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

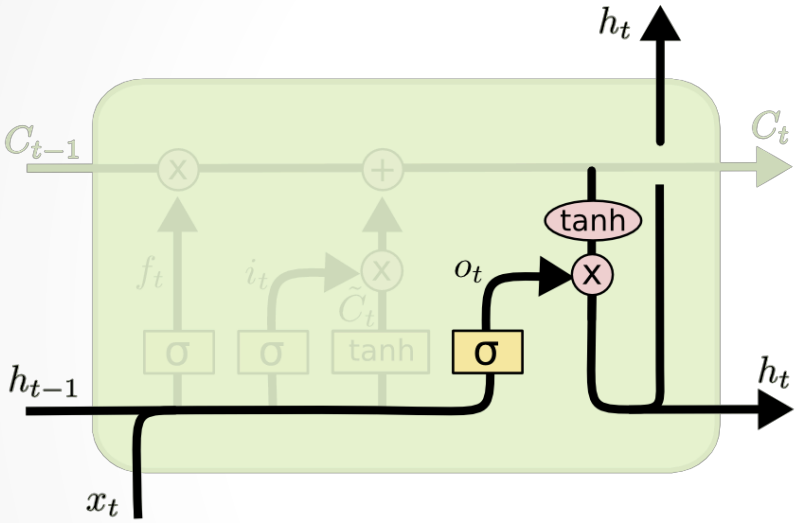
e.g.,
we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.

Applying the updates



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output gate



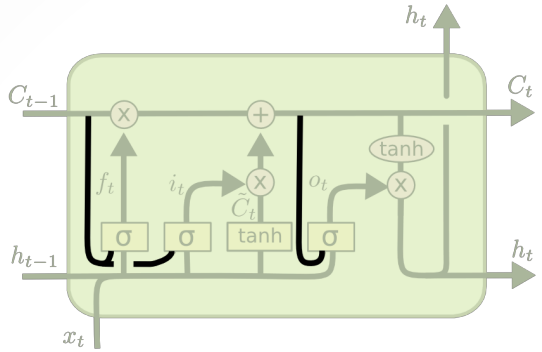
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

e.g.,
The network just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next.
For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.

Variants on Long Short Term Memory

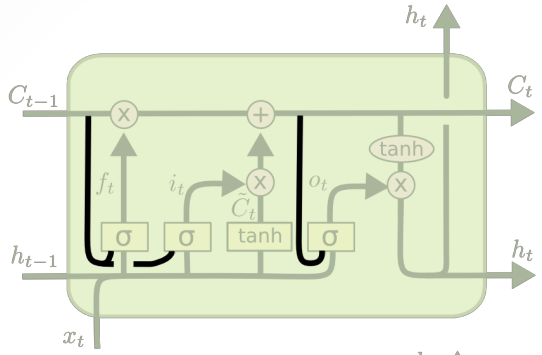
Peephole connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

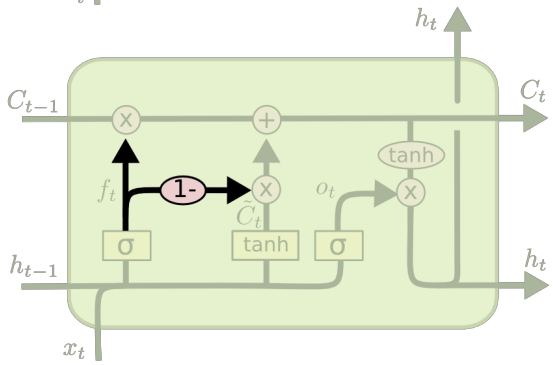
Variants on Long Short Term Memory

Peephole connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

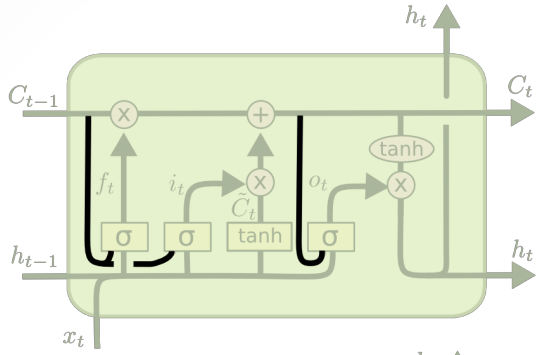
Coupled forget and input gates



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

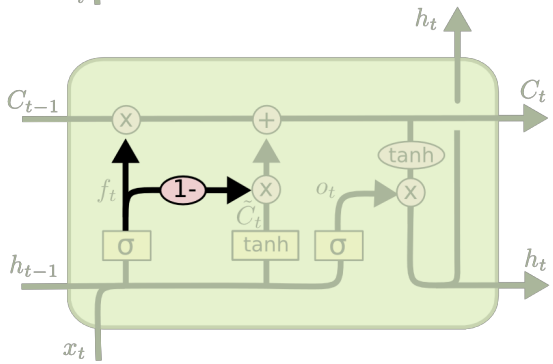
Variants on Long Short Term Memory

Peephole connections



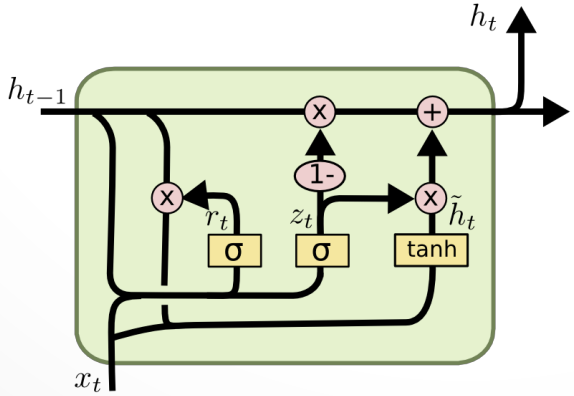
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Coupled forget and input gates



$$C_t = f_t * C_{t-1} + \boldsymbol{(1 - f_t)} * \tilde{C}_t$$

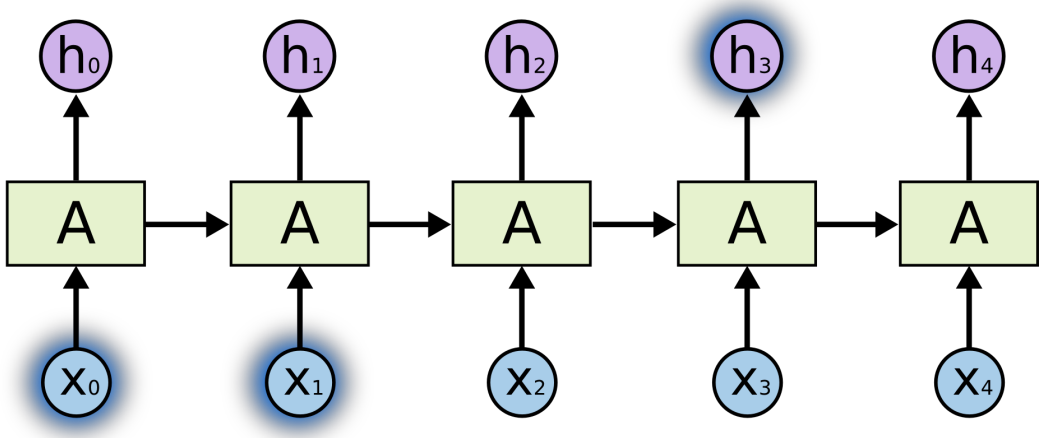
Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Word Embedding

Our inputs are strings!



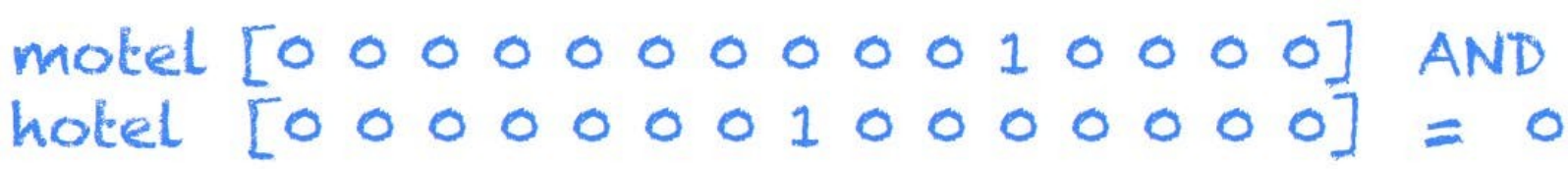
The Clouds are in the *sky*

How to Represent a Word?

- Challenge
  - Discrete structure
- Simple representation
  - One-hot representation: a vector with one 1 and a lot of zeroes
  - E.g., Motel =

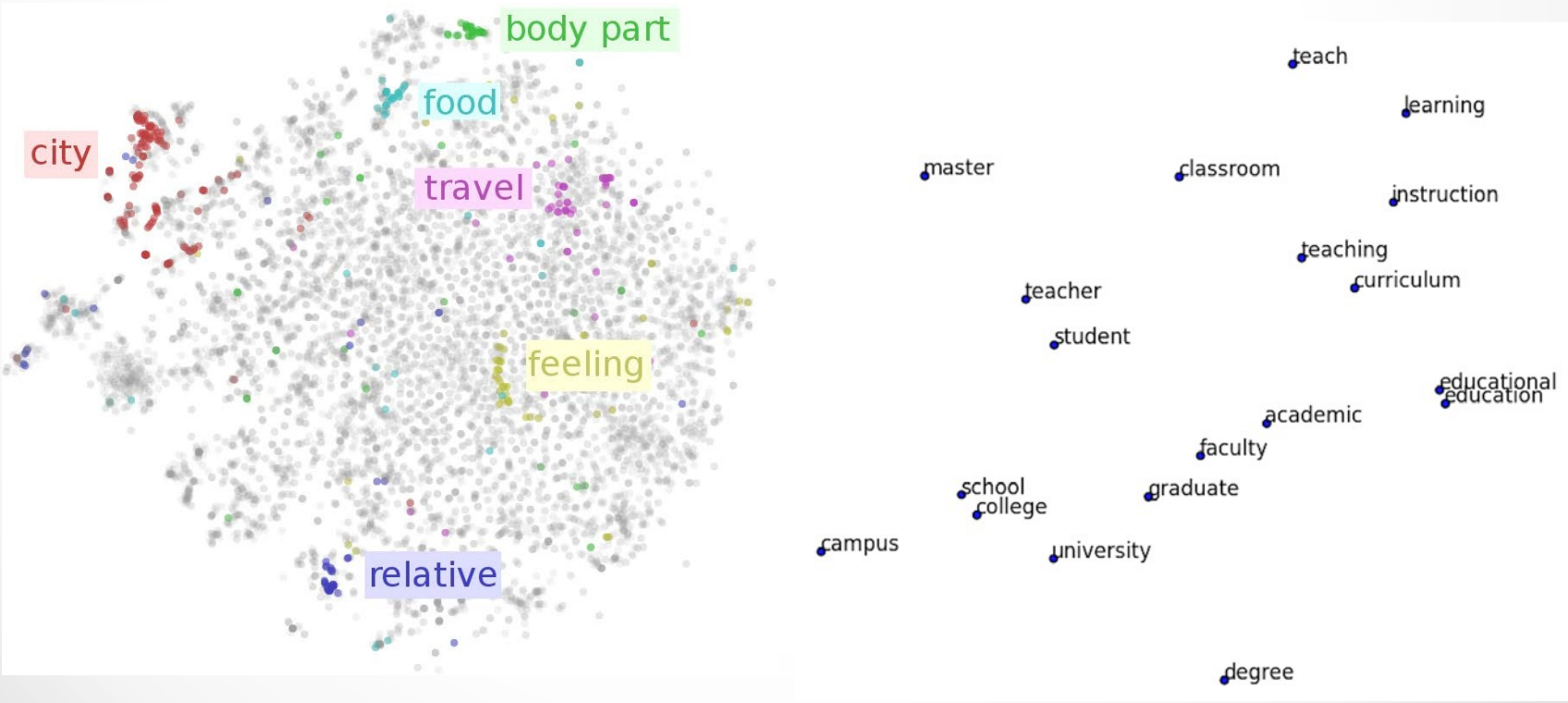
One-Hot Representation

- High dimensionality
  - E.g., for Google news, 13M words
- Sparse
  - Only 1 non-zero value
- It has no semantics
  - E.g.,



Word Embedding

- Low dimensional vector representation of every word
  - E.g., motel = [1.3, -1.4] and hotel = [1.2, -1.5]



How to Learn Such Embeddings?

- Using context information!

...he curtains open and the moon shining in on the barely...

...ars and the cold , close moon " . And neither of the w...

...rough the night with the moon shining so brightly , it...

...made in the light of the moon . It all boils down , wr...

...surely under a crescent moon , thrilled by ice-white...

...sun , the seasons of the moon ? Home , alone , Jay pla...

...m is dazzling snow , the moon has risen full and cold...

...un and the temple of the moon , driving out of the hug...

...in the dark and now the moon rises , full and amber a...

...bird on the shape of the moon over the trees in front...

Word2Vec

- Proposed by Mikolov et al. at Google in 2013
- The most popular word embedding models
- Fast
  - "an optimized single-machine implementation can train on more than 100 billion words in one day"
- Two main architectures:
  - Continuous bag-of-words (CBOW)
  - Skip-gram

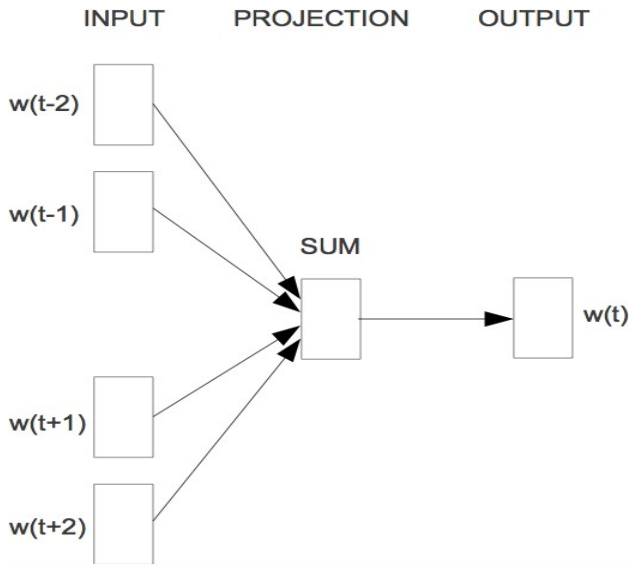
Main Idea of Word2Vec

- Consider a local window of a target word

- CBOW:
  predict the target words given the neighbors
- Skip-gram:
  predict neighbors given the target words

CBOW

- Predicting target using neighbors

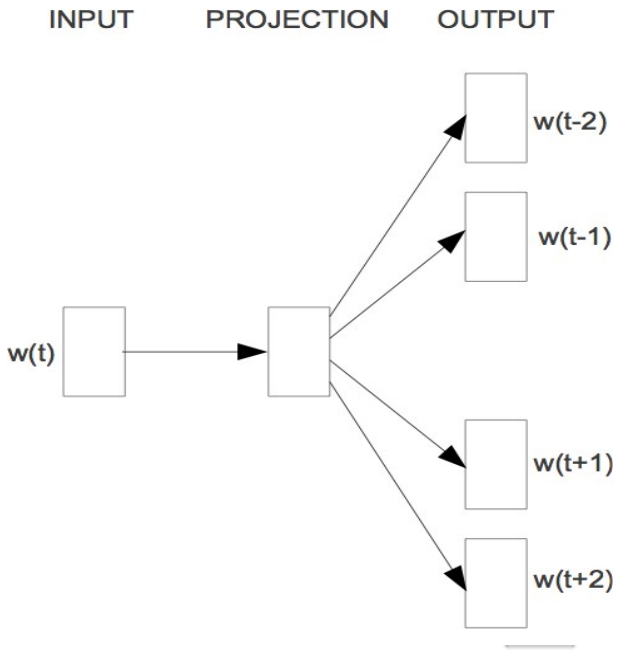


$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} logp(w_t|w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n})$$

More details can be found in:
http://www.1-4-5.net/~dmm/ml/how_does_word2vec_work.pdf

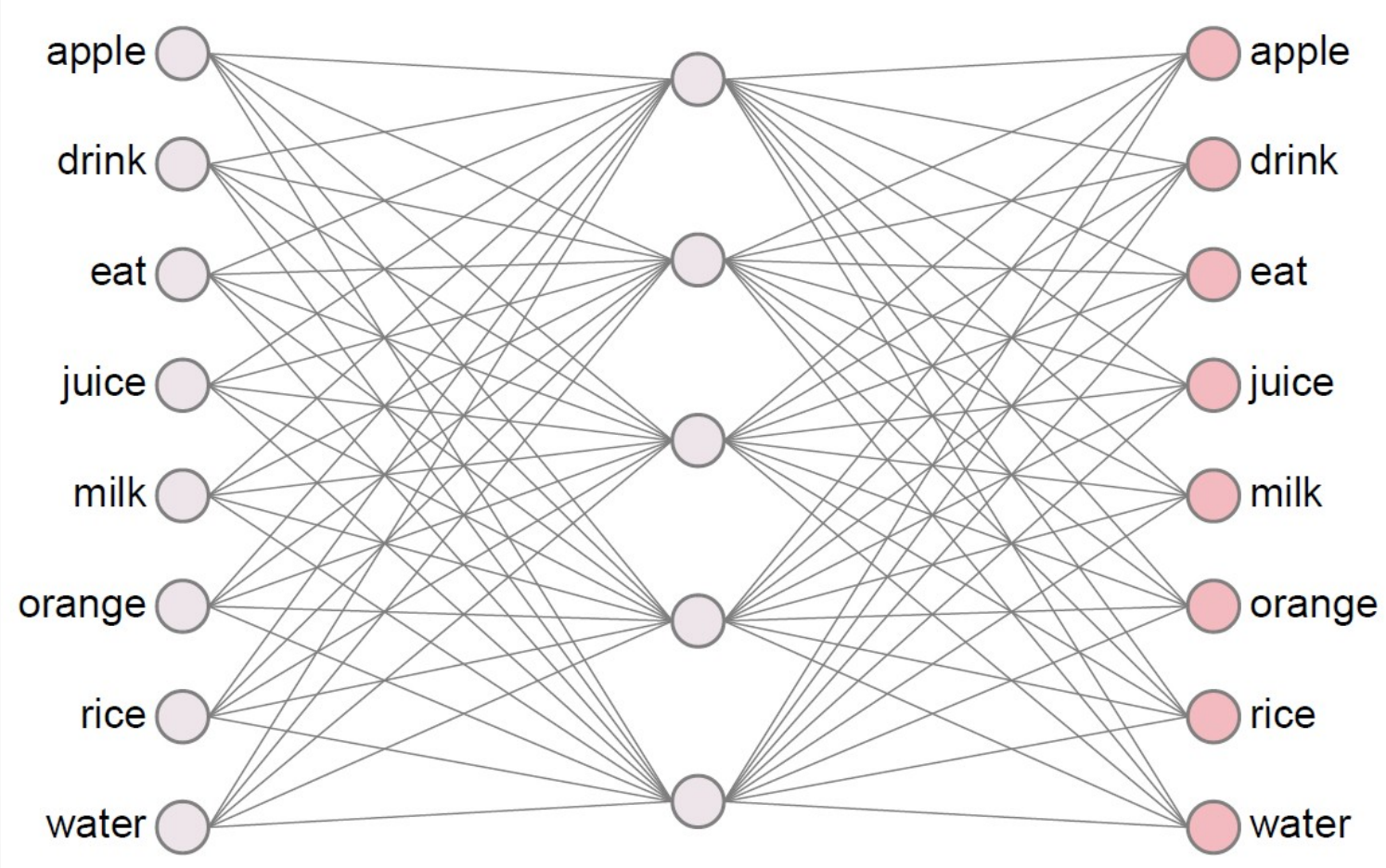
Skip-Gram

- Predicting neighbors using target



$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\le j\le n, j\neq 0} logp(w_{t+j}|w_t)$$

A Neural Network Point of View



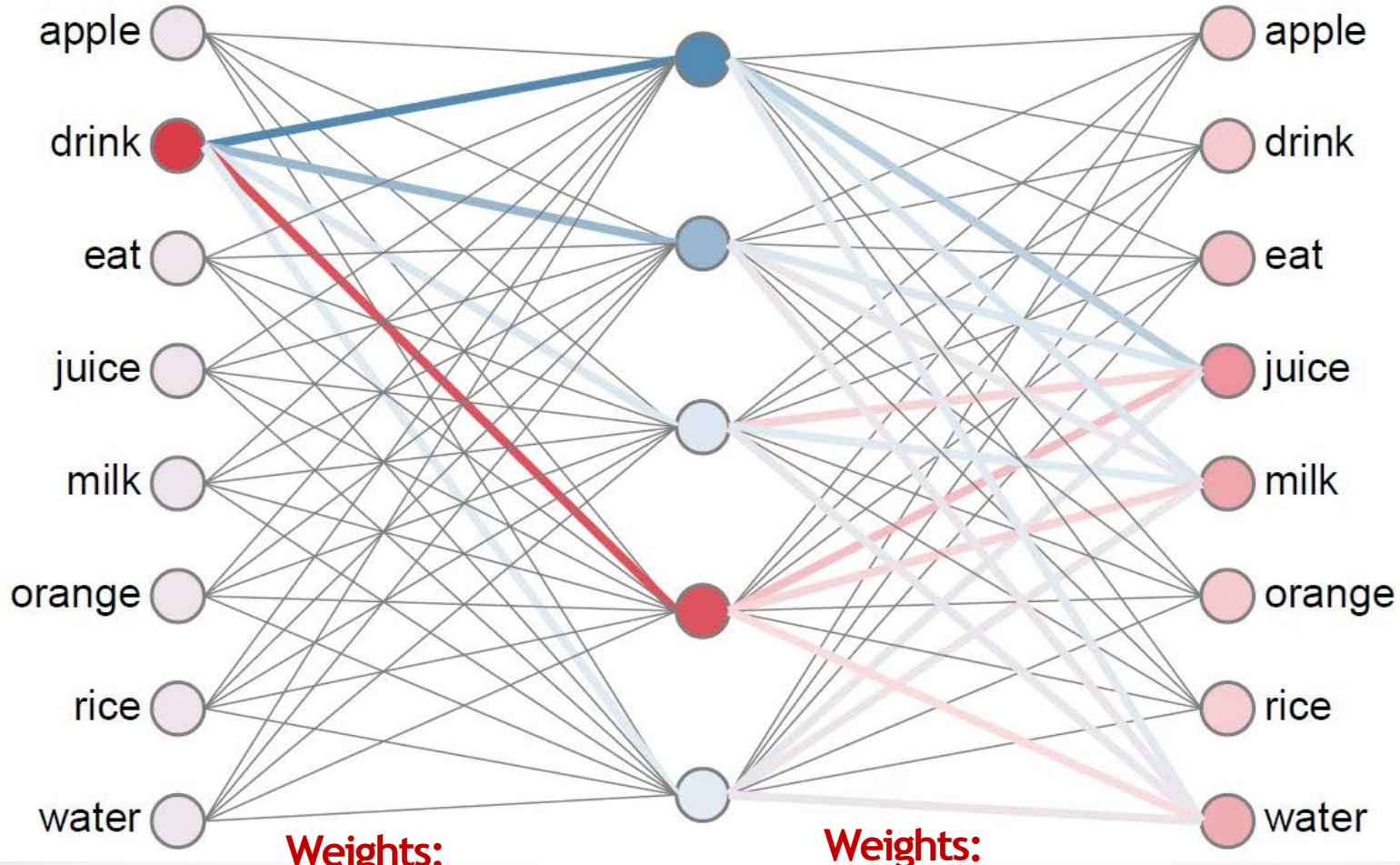
Input Layer:
one-hot vector

Hidden Layer:
Linear (Identity)

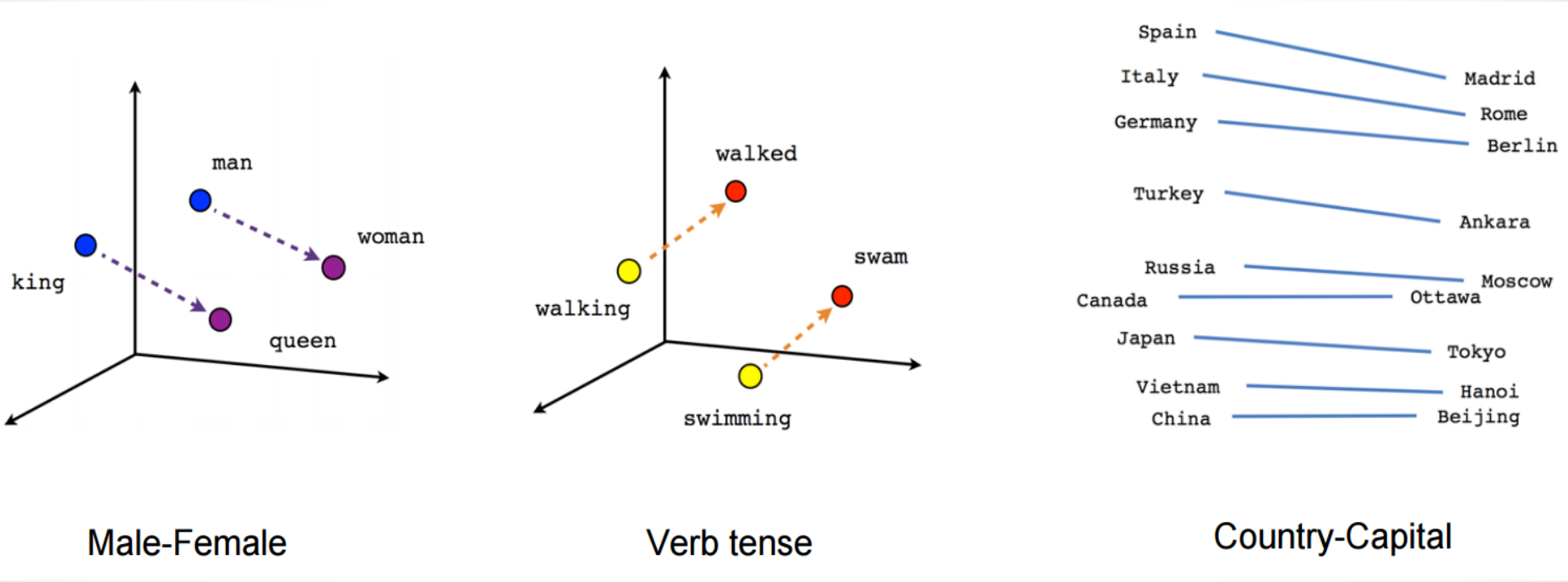
Output Layer:
softmax

Demo

- https://ronxin.github.io/wevi/



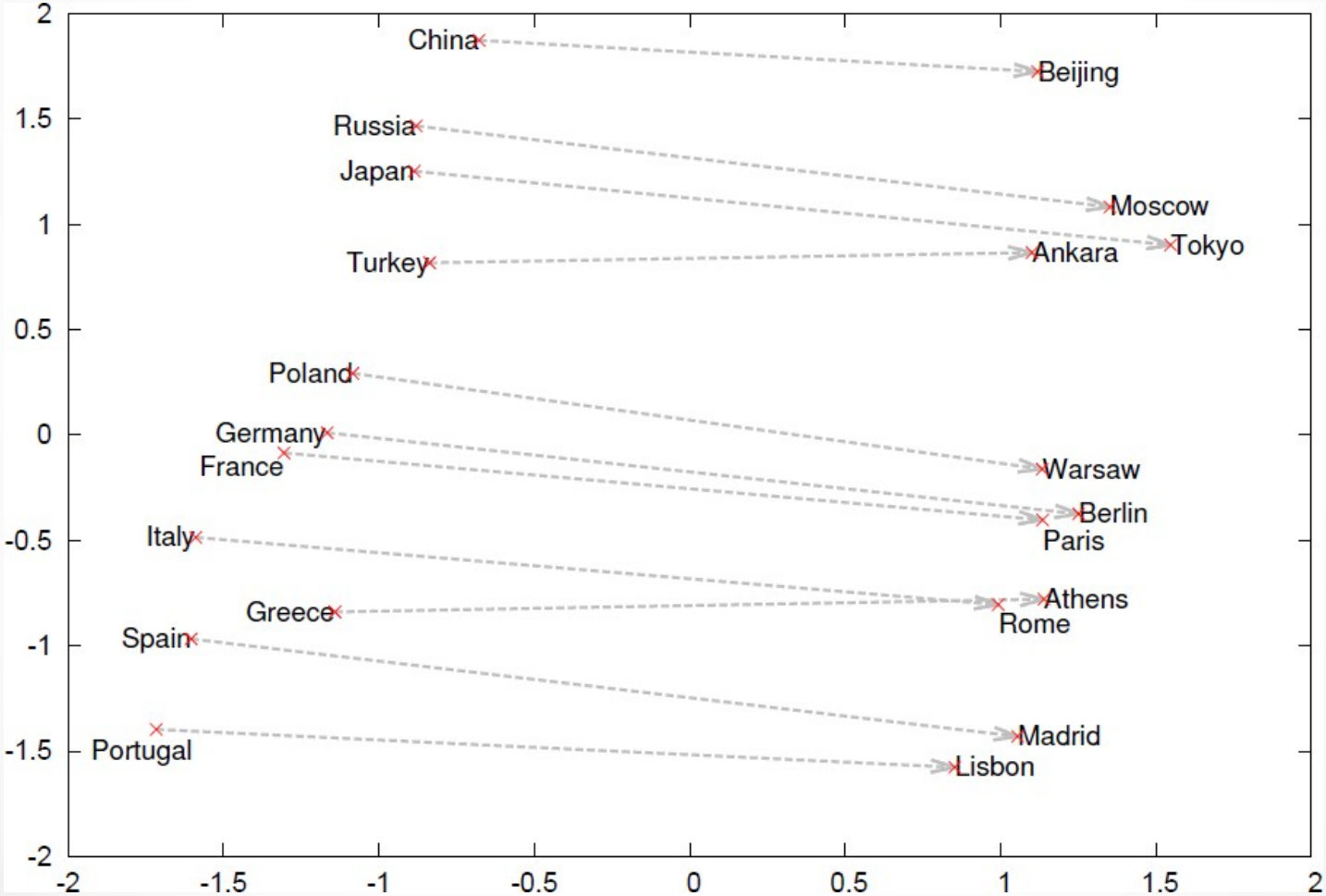
Examples



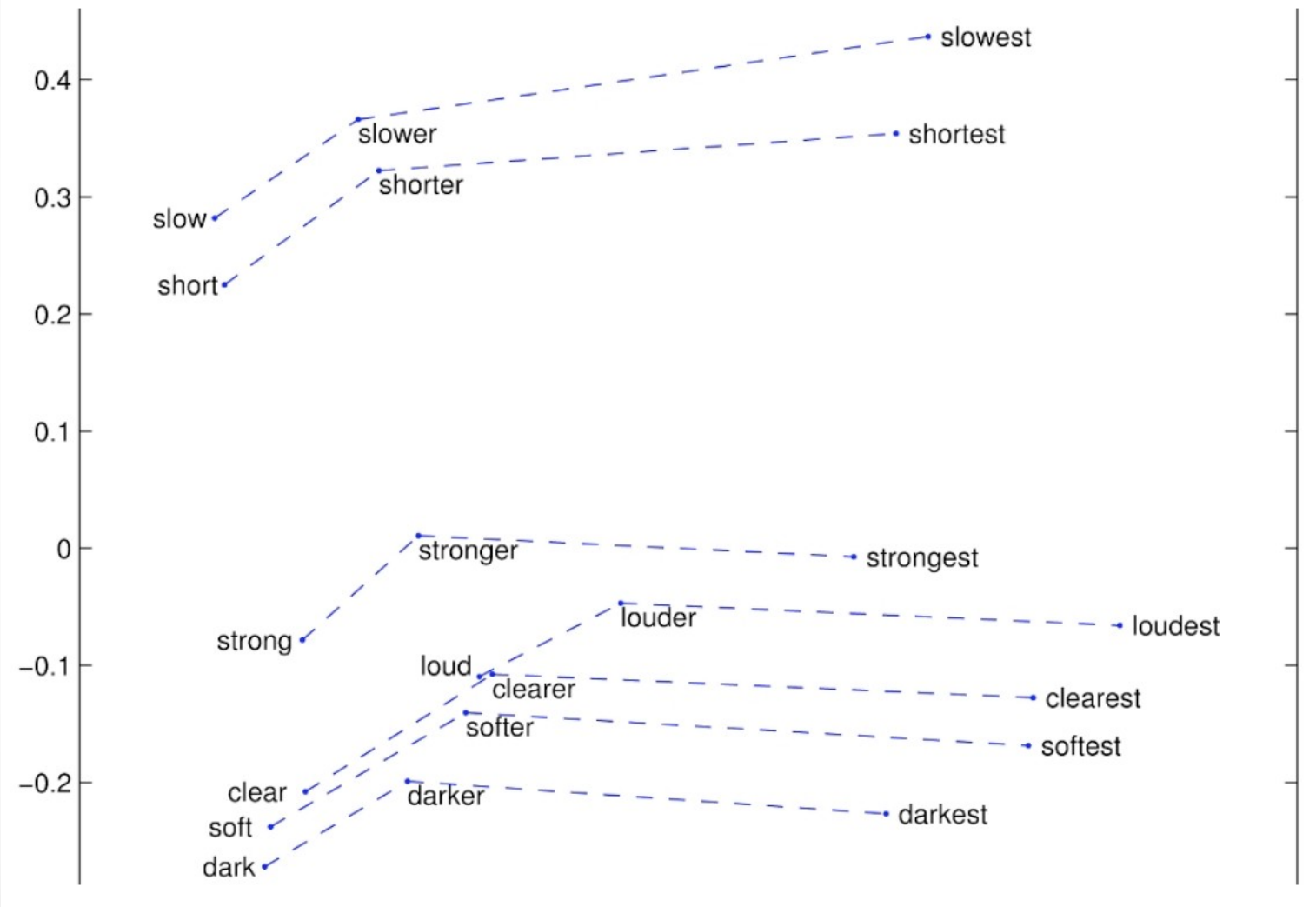
vector[Queen] = vector[King] - vector[Man] + vector[Woman]

A Potential Application

- Relationship detection



Some Interesting Results: Superlatives



Word2Vec

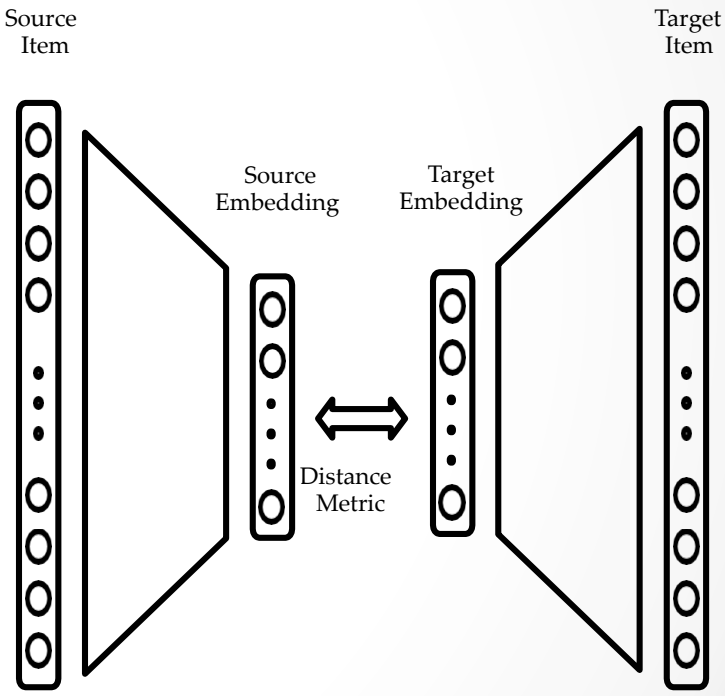
Start with a paired items dataset

[source, target]

Train a neural network

Bottleneck layer gives you a dense vector representation

E.g., word2vec



Pennington, Socher, and Manning. "Glove: Global Vectors for Word Representation." *EMNLP* (2014).

GloVe
(Global Vectors for Word Representation)

Start with a paired items dataset

[source, target]

Make a Source x Target matrix

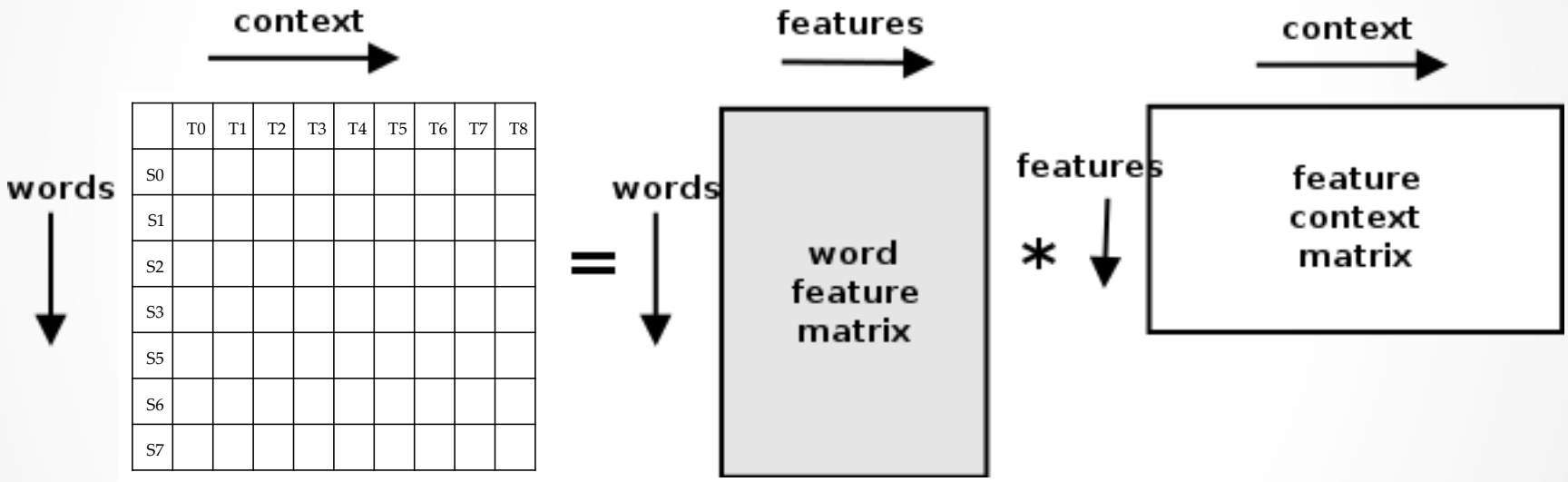
Factorizing the matrix gives you a dense vector representation

E.g., LSA, GloVe

	T0	T1	T2	T3	T4	T5	T6	T7	T8
S0									
S1									
S2									
S3									
S5									
S6									
S7									

Pennington, Socher, and Manning. "Glove: Global Vectors for Word Representation." *EMNLP* (2014).

GloVe
(Global Vectors for Word Representation)



Pennington, Socher, and Manning. "Glove: Global Vectors for Word Representation." *EMNLP* (2014).

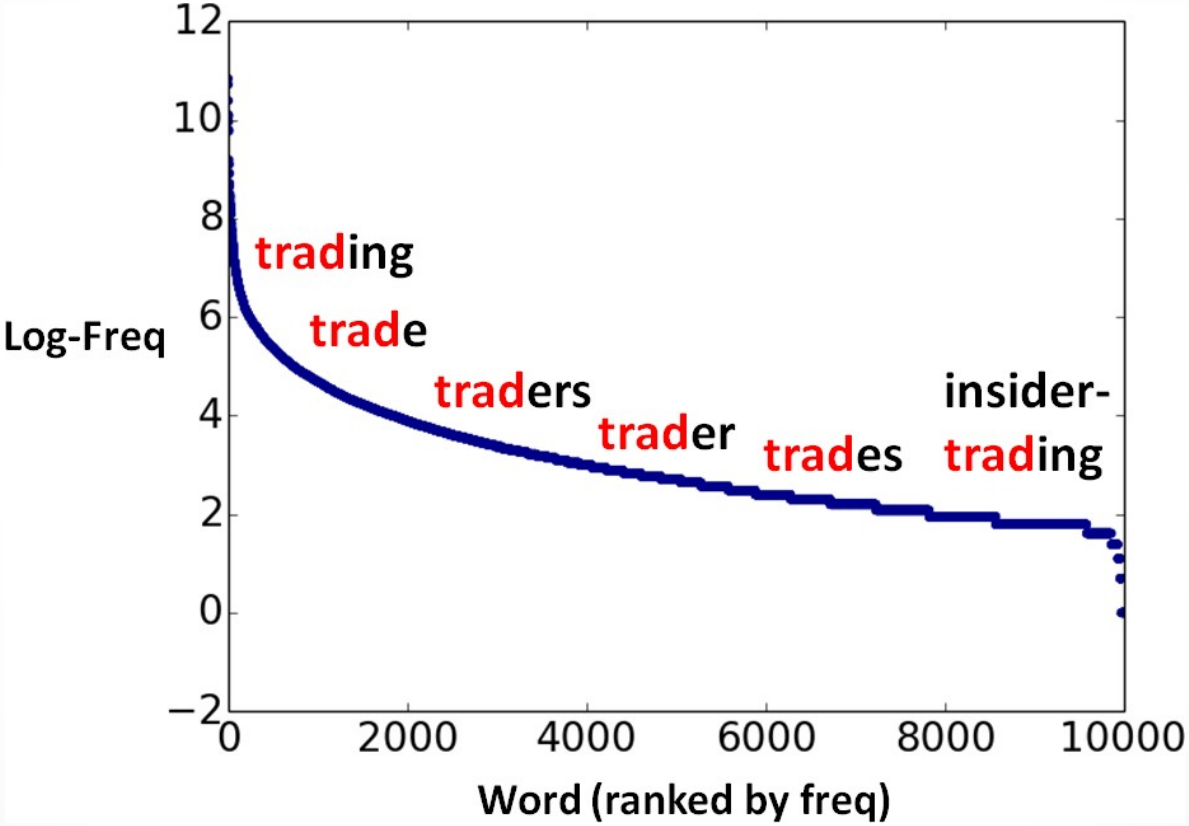
Some examples of text embeddings

	Embedding for	Source Item	Target Item	Learning Model
Latent Semantic Analysis Deerwester et. al. (1990)	Single word	Word (one-hot)	Document (one-hot)	Matrix factorization
Word2vec Mikolov et. al. (2013)	Single Word	Word (one-hot)	Neighboring Word (one-hot)	Neural Network (Shallow)
Glove Pennington et. al. (2014)	Single Word	Word (one-hot)	Neighboring Word (one-hot)	Matrix factorization
Semantic Hashing (auto-encoder) Salakhutdinov and Hinton (2007)	Multi-word text	Document (bag-of-words)	Same as source (bag-of-words)	Neural Network (Deep)
DSSM Huang et. al. (2013), Shen et. al. (2014)	Multi-word text	Query text (bag-of-trigrams)	Document title (bag-of-trigrams)	Neural Network (Deep)
Session DSSM Mitra (2015)	Multi-word text	Query text (bag-of-trigrams)	Next query in session (bag-of-trigrams)	Neural Network (Deep)
Language Model DSSM Mitra and Craswell (2015)	Multi-word text	Query prefix (bag-of-trigrams)	Query suffix (bag-of-trigrams)	Neural Network (Deep)

Character Embedding

An Issue with word embedding

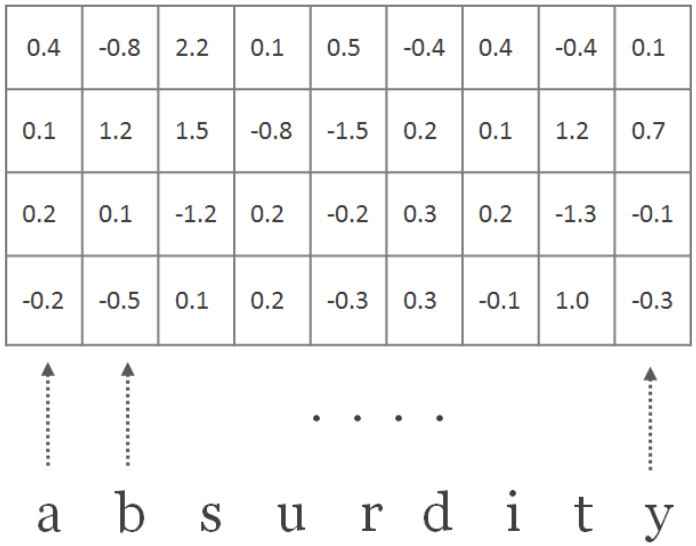
Separate embeddings for “trading”, “trade”, “trades”, etc.



One Solution: Pre-pocess using morphological segmenter

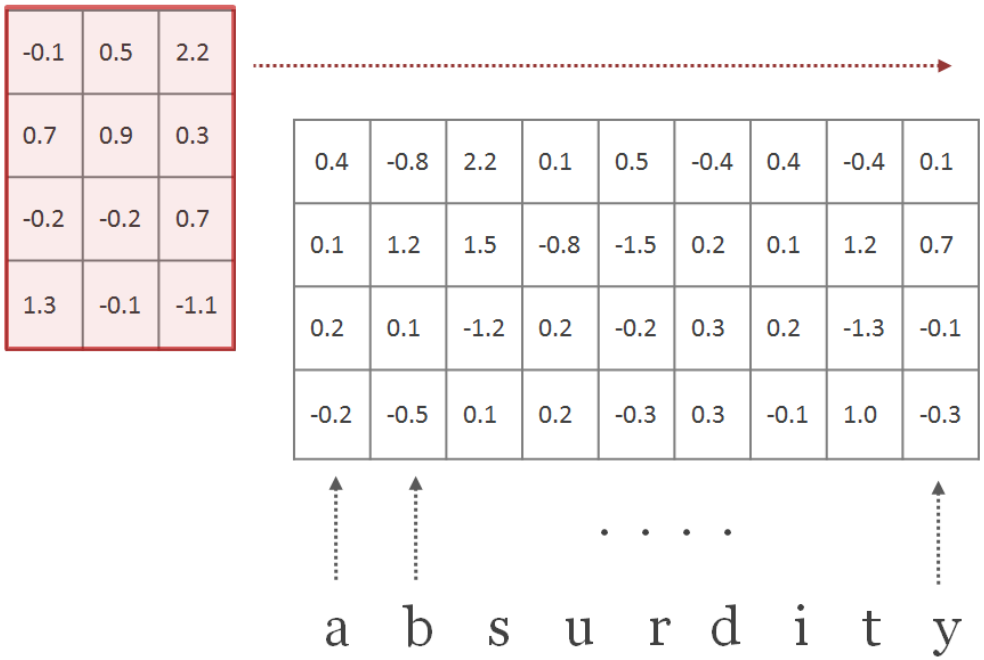
CNN - Character Embedding

$\mathbf{C} \in R^{d \times l}$: Representation of *absurdity*



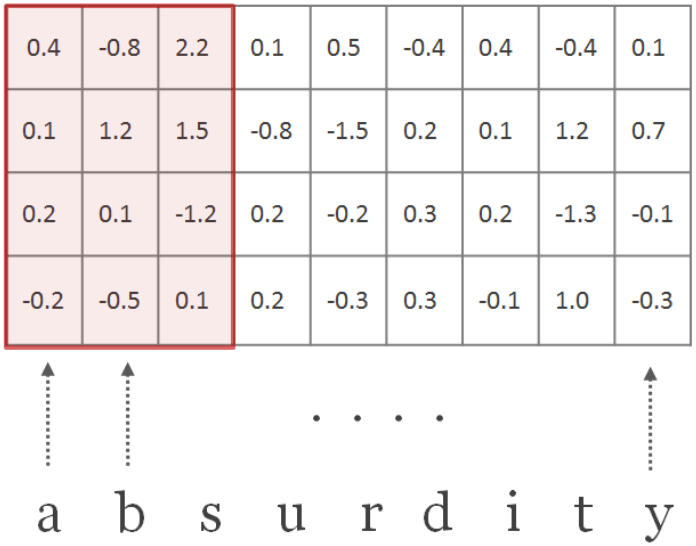
CNN - Character Embedding

$\mathbf{H} \in R^{d \times w}$: Convolutional filter matrix of width $w = 3$



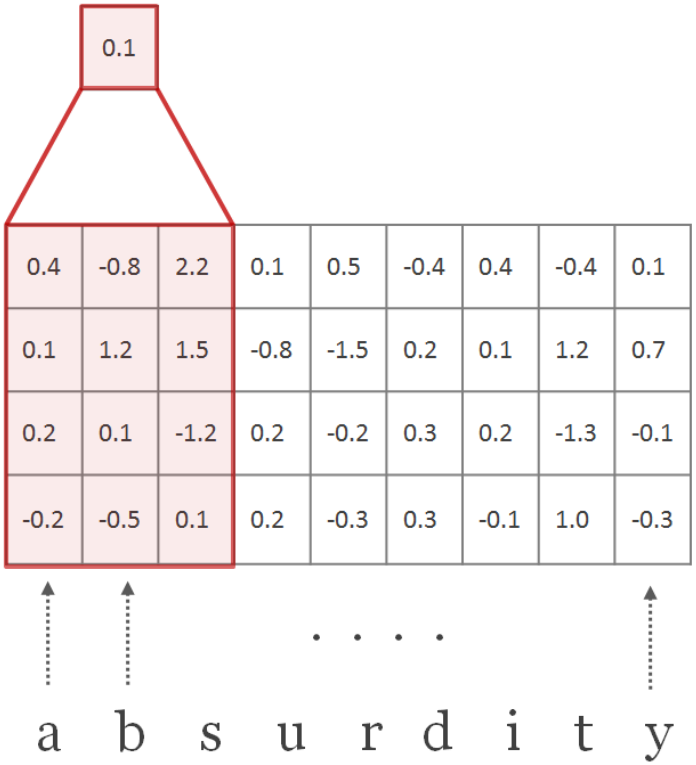
CNN - Character Embedding

$$\mathbf{f}[1] = (\mathbf{C}[*, 1:3], \mathbf{H})$$



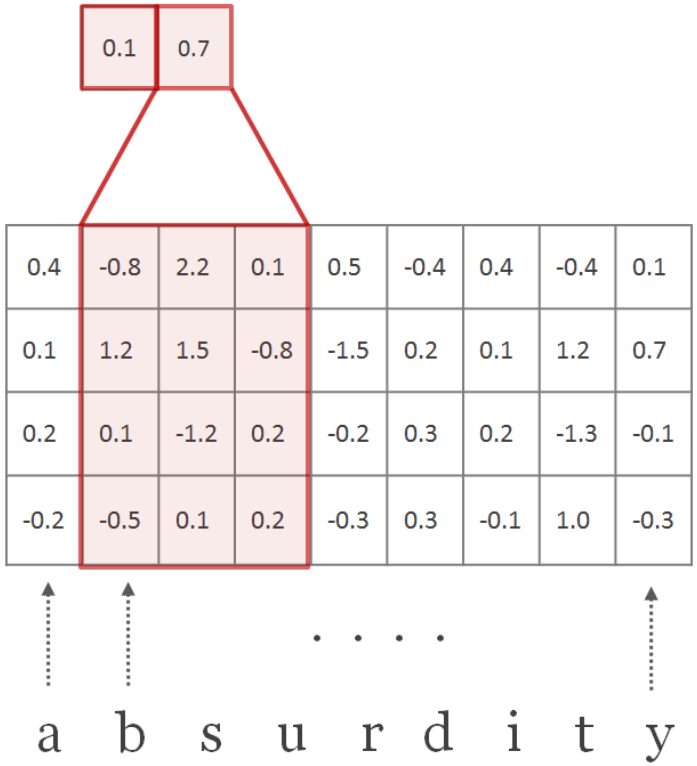
CNN - Character Embedding

$$\mathbf{f}[1] = (\mathbf{C}[*, 1:3], \mathbf{H})$$



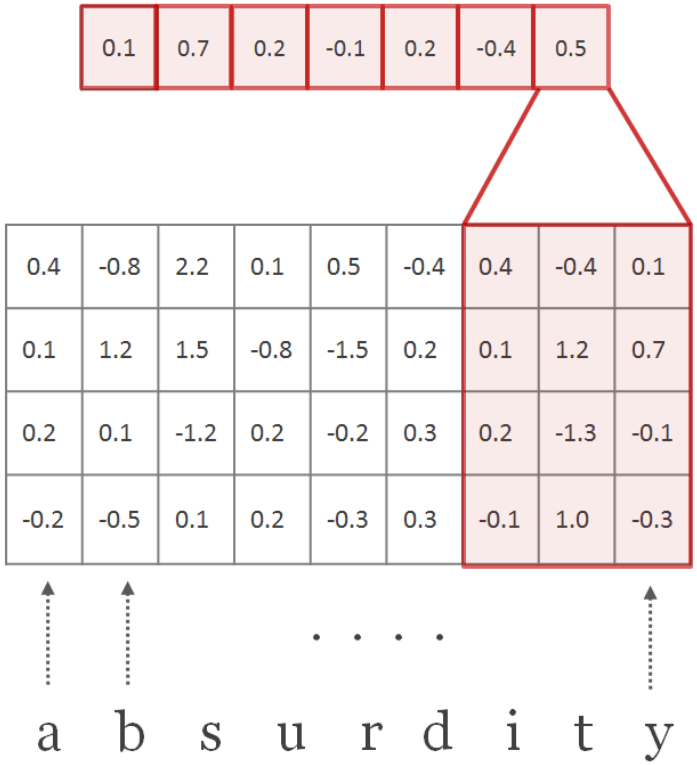
CNN - Character Embedding

$$\mathbf{f}[2] = (\mathbf{C}[*, 2:4], \mathbf{H})$$



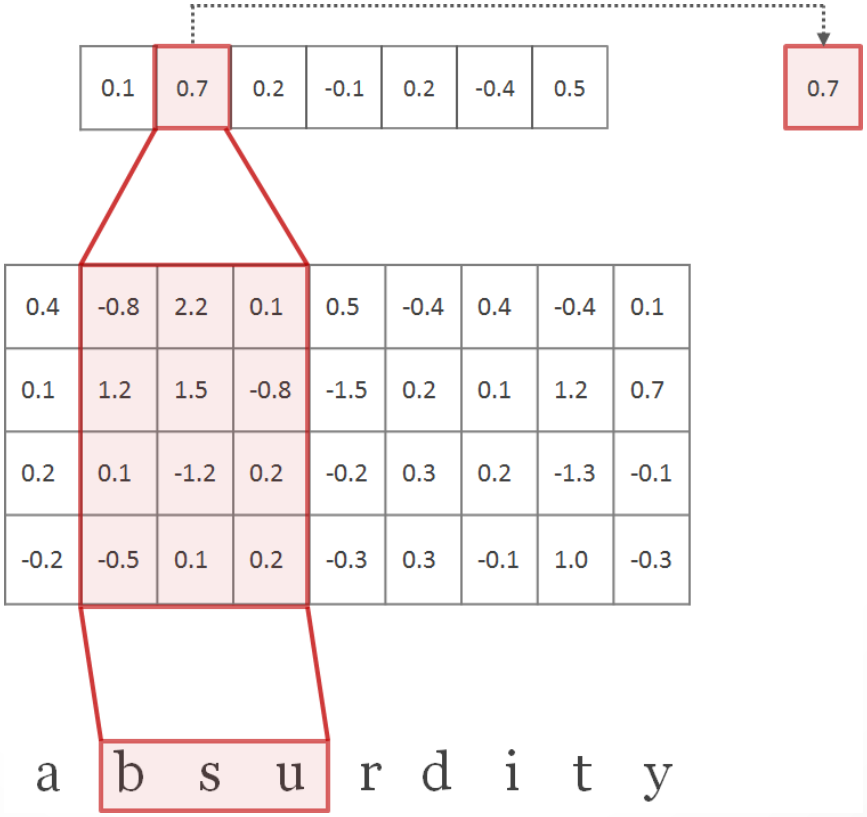
CNN - Character Embedding

$$\mathbf{f}[T-2] = (\mathbf{C}[*, T-2 : T], \mathbf{H})$$



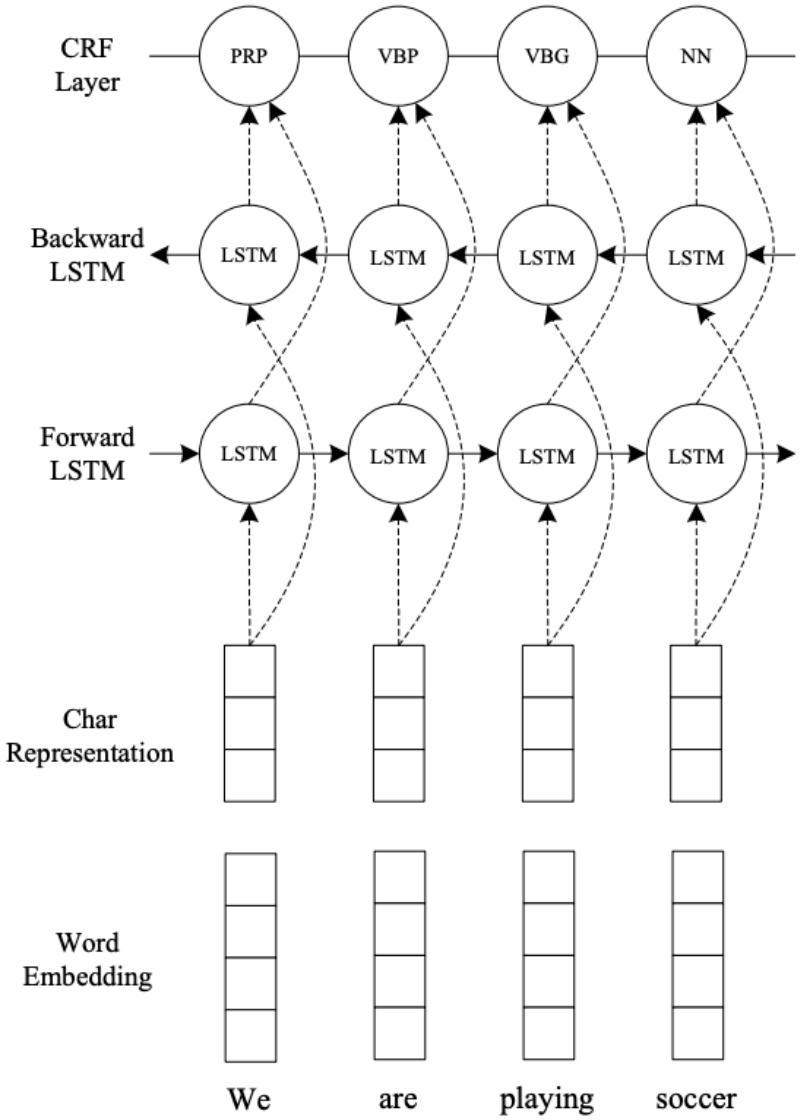
CNN - Character Embedding

Each filter picks out a character *n*-gram

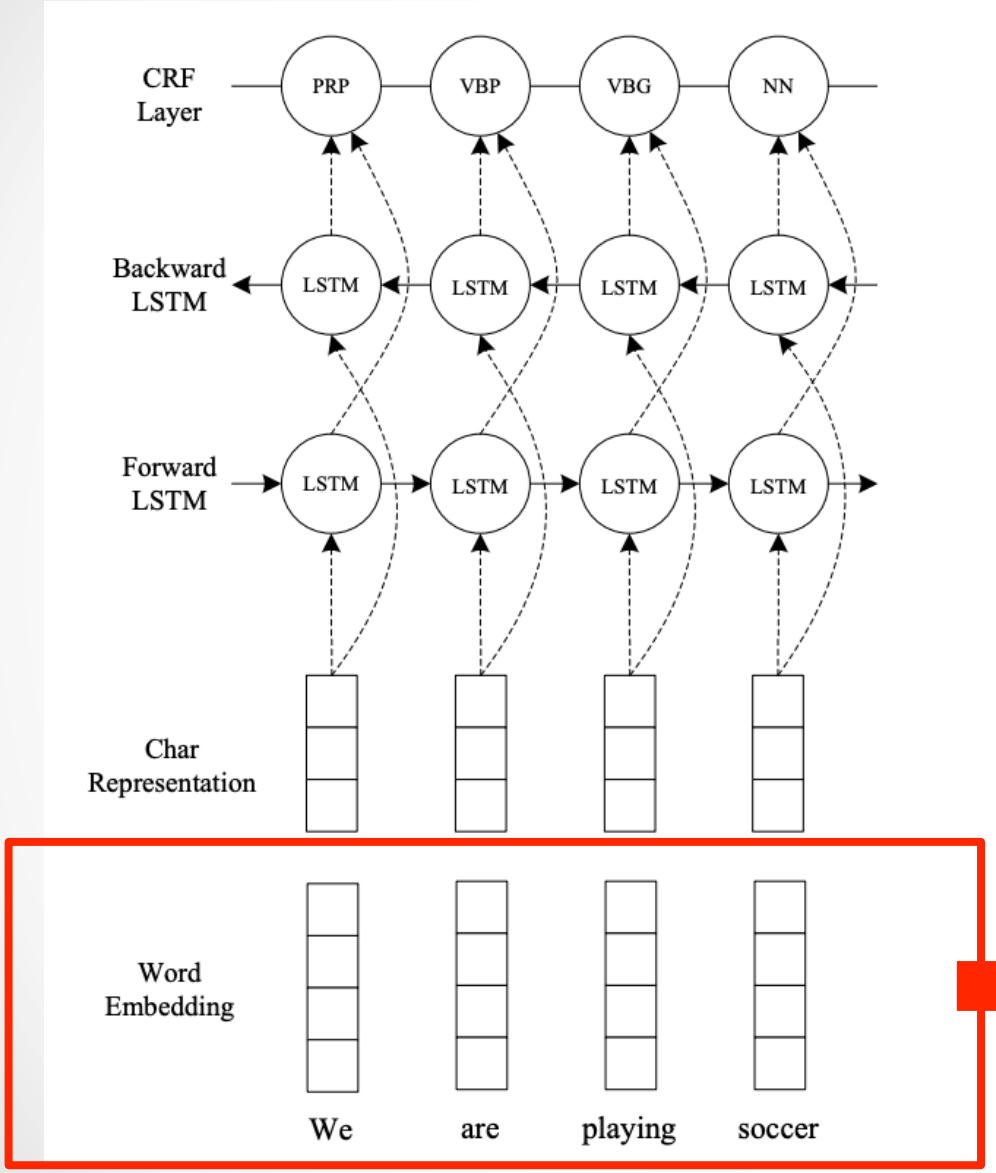


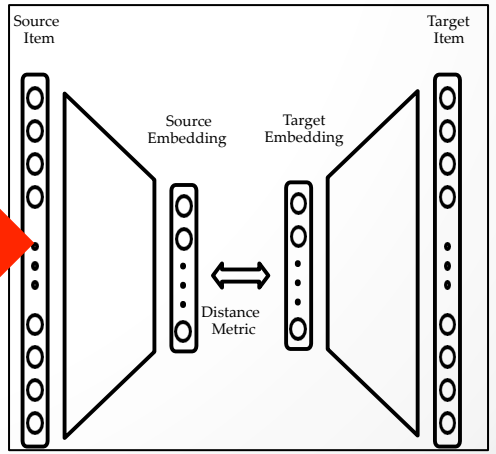
A Deep Learning Model for NER

Bi-directional LSTM-CNNs-CRF

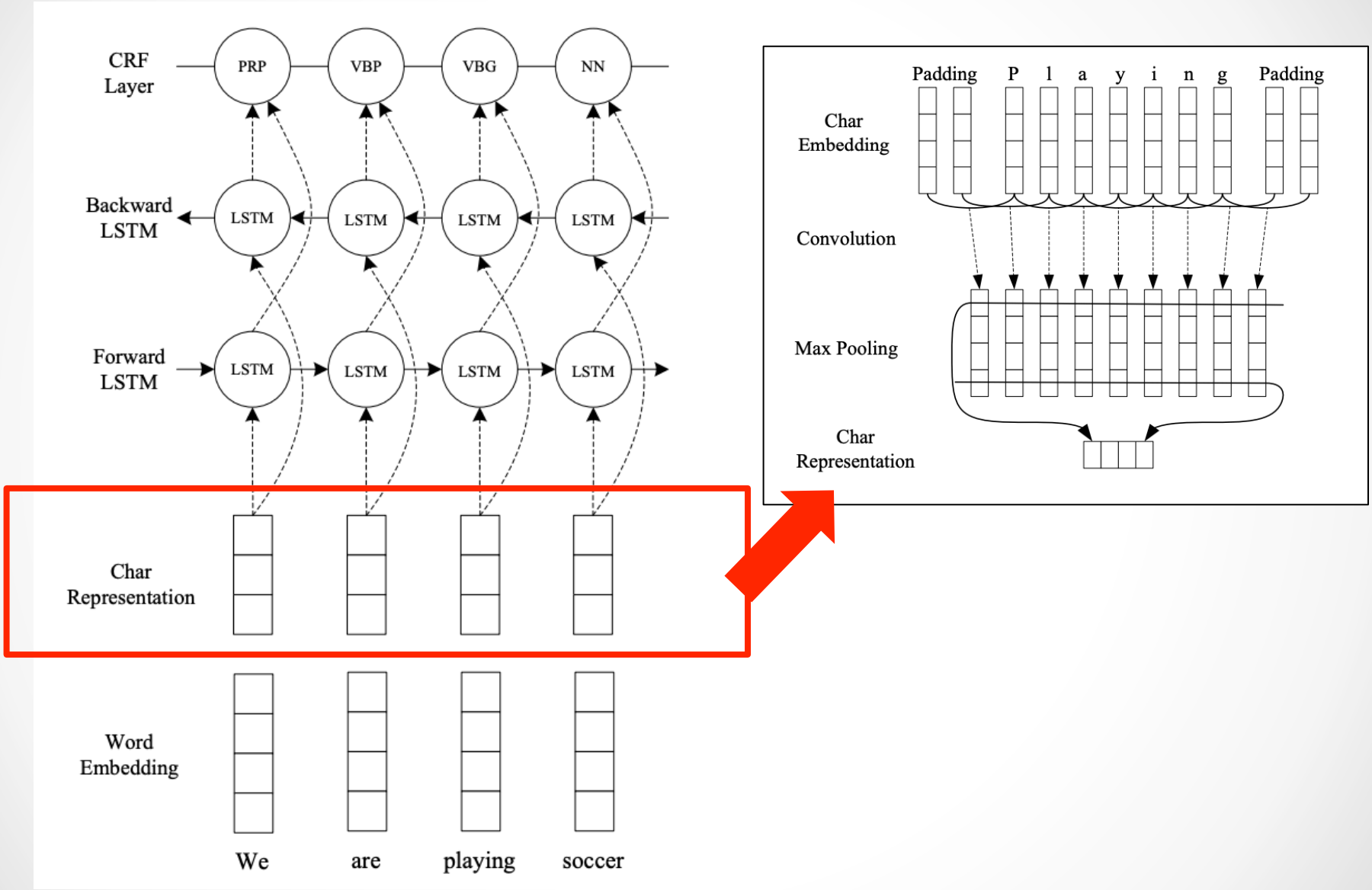


Bi-directional LSTM-CNNs-CRF

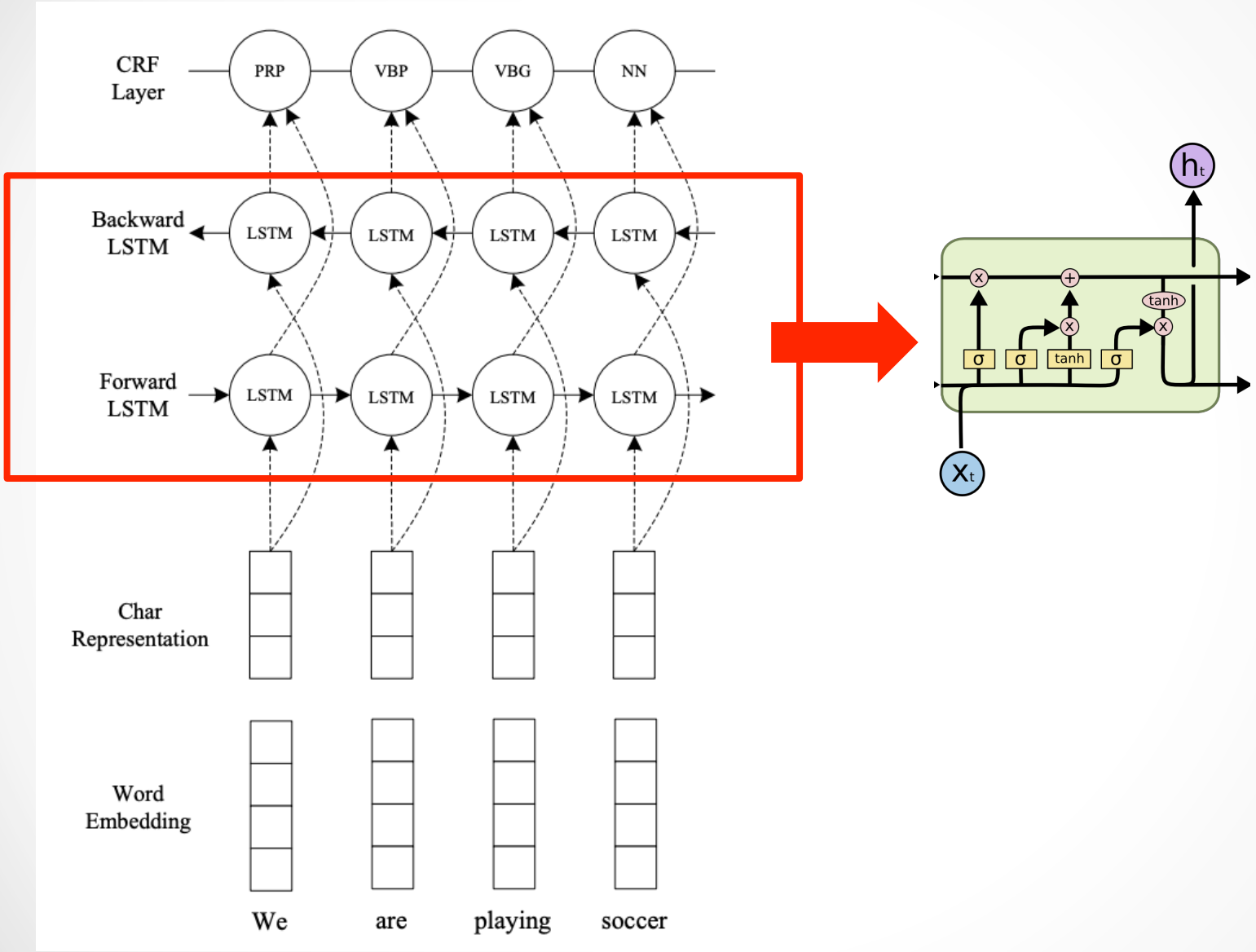




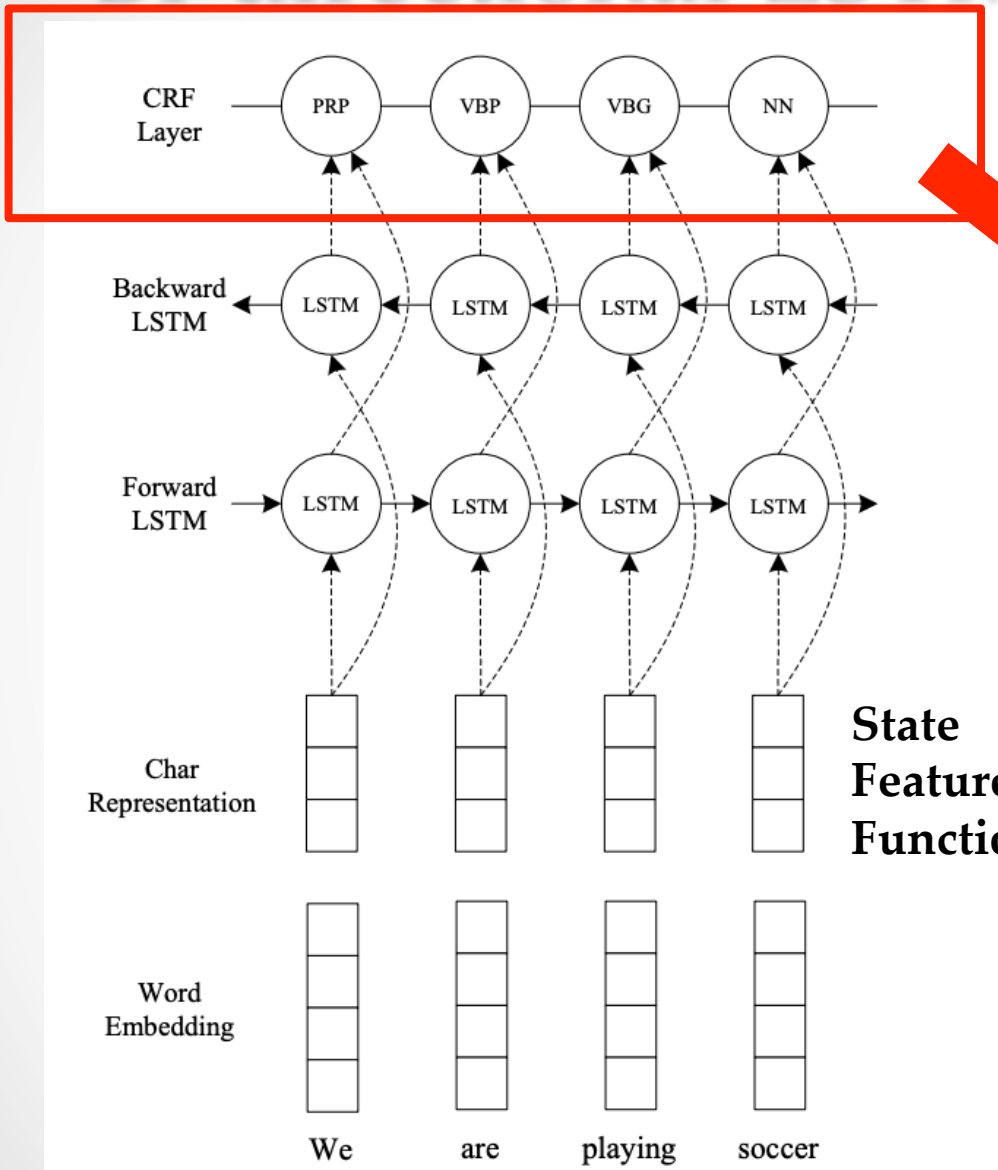
Bi-directional LSTM-CNNs-CRF

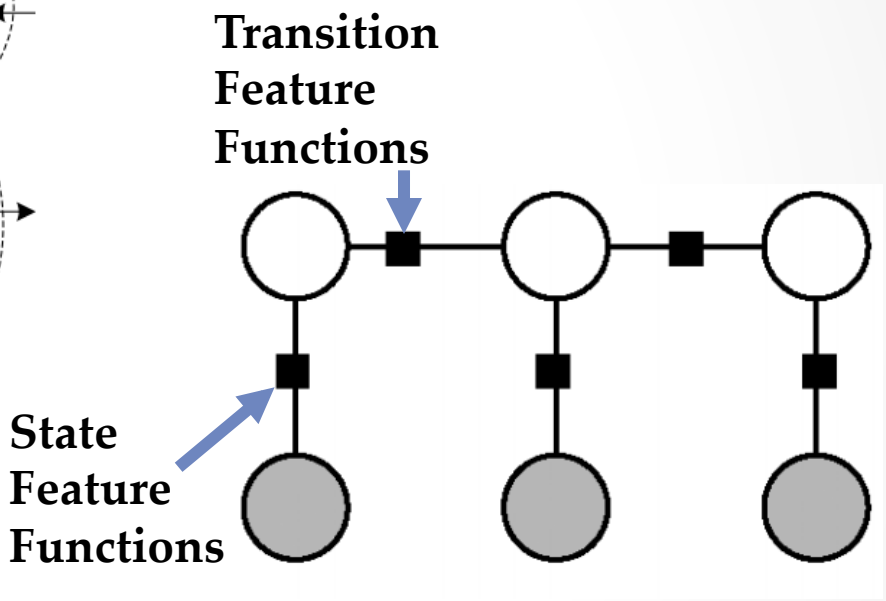


Bi-directional LSTM-CNNs-CRF



Bi-directional LSTM-CNNs-CRF





Performance Result

Model	NER		
	Test		
	Prec.	Recall	F1
BRNN	87.05	83.88	85.44

CoNLL 2003 (PERSON, LOCATION, ORGANIZATION, and MISC)

Performance Result

Model	NER		
	Test		
	Prec.	Recall	F1
BRNN	87.05	83.88	85.44
BLSTM	87.77	86.23	87.00

CoNLL 2003 (PERSON, LOCATION, ORGANIZATION, and MISC)

Performance Result

Model	NER		
	Test		
	Prec.	Recall	F1
BRNN	87.05	83.88	85.44
BLSTM	87.77	86.23	87.00
BLSTM-CNN	88.53	90.21	89.36

CoNLL 2003 (PERSON, LOCATION, ORGANIZATION, and MISC)

Performance Result

Model	NER		
	Test		
	Prec.	Recall	F1
BRNN	87.05	83.88	85.44
BLSTM	87.77	86.23	87.00
BLSTM-CNN	88.53	90.21	89.36
BRNN-CNN-CRF	91.35	91.06	91.21

CoNLL 2003 (PERSON, LOCATION, ORGANIZATION, and MISC)