Learning and deep neural networks

Dr. Jun Wang

Computer Science, UCL

Content

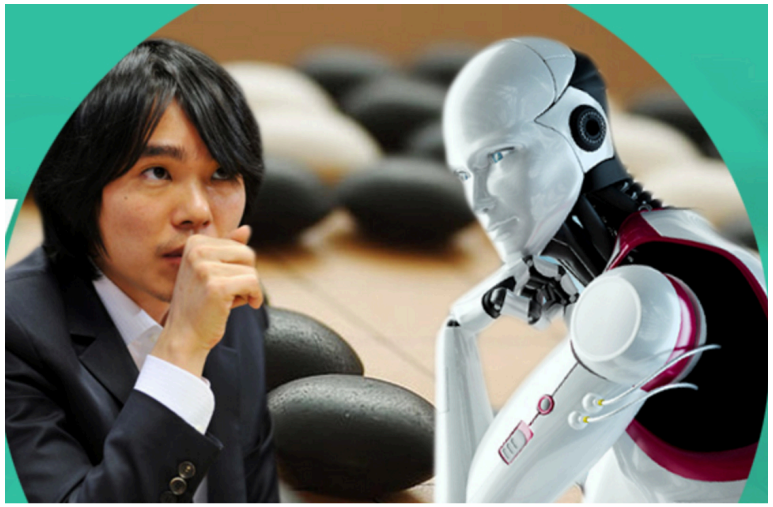
- Lecture 1: Multiagent AI and basic game theory
- Lecture 2: Potential games, and extensive form and repeated games
- Lecture 3: Solving (“Learning”) Nash Equilibria
- Lecture 4: Bayesian Games, auction theory
- Lecture 5: Learning and deep neural networks
- Lecture 6: Single-agent Learning (1)
- Lecture 7: Single-agent Learning (2)
- Lecture 8: Multi-agent Learning (1)
- Lecture 9: Multi-agent Learning (2)
- Lecture 10: Multi-agent Learning (3)

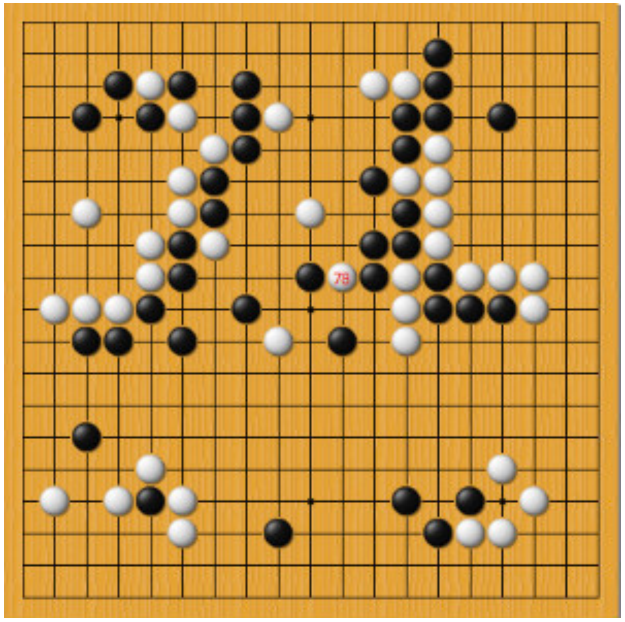
Subject to change

learning in single agent

- A typical AI concerns the learning performed by an individual agent
- In that setting, the goal is to design an agent that learns to function successfully in an environment that is unknown and potentially also changes as the agent is learning
  - Learning to predict click-through rate by logistic regression

AlphaGo vs. the world's 'Go' champion





Rating List

s, check the History page. There is also a History of top ladies.

Rank	Name	♂ ♀	Flag	Elo
1	Ke Jie	♂		3621
2	Park Jungwhan	♂		3569
3	Iyama Yuta	♂		3546
4	Google AlphaGo			3533
5	Lee Sedol	♂		3521
6	Shi Yue	♂		3509
7	Park Yeonghun	♂		3509
8	Kim Jiseok	♂		3504
9	Mi Yuting	♂		3501
10	Zhou Ruiyang	♂		3498
11	Kang Dongyun	♂		3498
12	Tang Weixing	♂		3479
13	Lian Xiao	♂		3475
14	Chen Yaoye	♂		3472
15	Gu Zihao	♂		3468
16	Gu Li	♂		3455
17	Huang Yunsong	♂		3452
18	Jiang Weijie	♂		3448

http://www.goratings.org/

Coulom, Rémi. "Whole-history rating: A bayesian rating system for players of time-varying strength." Computers and games. Springer Berlin Heidelberg, 2008. 113-124.

Deep Learning helps Speech Translation

Scientists See Promise in Deep-Learning Programs

By JOHN MARKOFF NOV. 23, 2012

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese. Hao Zhang/The New York Times

Skype's real-time translator now speaks French and German

By Tom Warren on June 18, 2015 05:12 am @tomwarren

COMMENTS

THE LATEST HEADLINES

Netflix is doubling its number of original scripted shows next year

Google enables Safe Browsing by default on Chrome for Android

Japan's Akatsuki spacecraft is finally in orbit around Venus

Kickstarter projects have 'reasonable' 9 percent failure rate, study finds

We should mock IBM's terrible hair dryer

Milestones on the path to Skype Translator

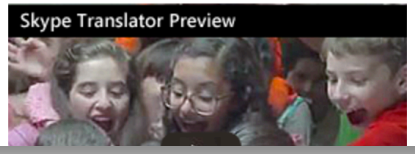
2014

December 15: Microsoft releases a preview version of Skype Translator for English and Spanish audiences.

- Read more on the Skype blog

November 12: Elementary school students in Tacoma, Washington, and Mexico City participate in the first Skype Mystery Call that uses a test version of Skype Translator.

Video: Skype Mystery Call



Skype users to get real-time language translating tool

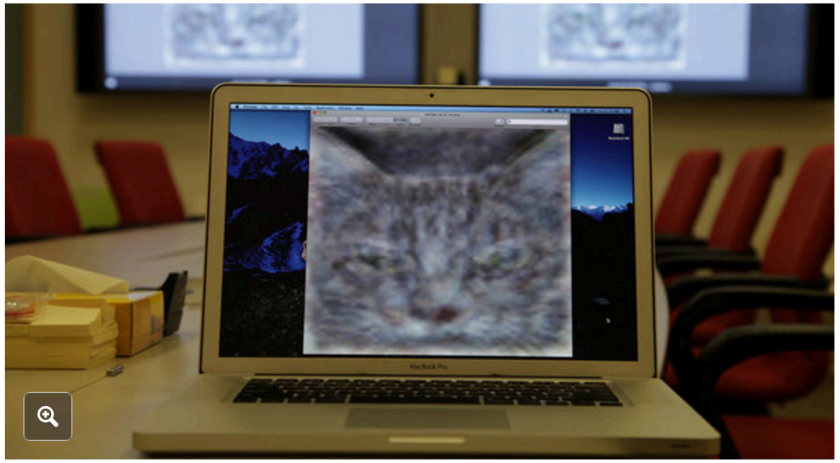
http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-intelligence.html
http://research.microsoft.com/en-us/about/speech-to-speech-milestones.aspx

Deep Learning helps Computer Vision

TECHNOLOGY

How Many Computers to Identify a Cat? 16,000

By JOHN MARKOFF JUNE 25, 2012



An image of a cat that a neural network taught itself to recognize. Jim Wilson/The New York Times

MOUNTAIN VIEW, Calif. — Inside Google's secretive X laboratory, known for inventing self-driving cars and augmented reality glasses, a small group of researchers began working several years ago on a simulation of the human brain.

There Google scientists created one of the largest neural networks for machine learning by connecting 16,000 computer processors, which they turned loose on the Internet to learn on its own.

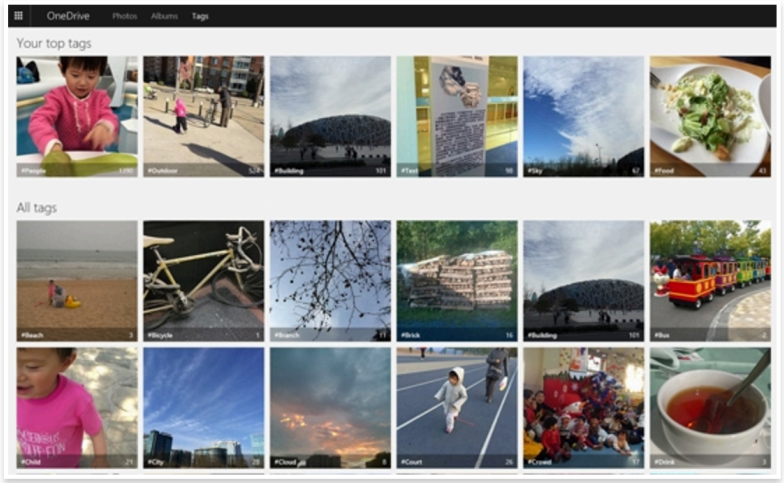
Presented with 10 million digital images found in YouTube videos, what did Google's brain do? What millions of humans do with YouTube: looked for cats.

Microsoft Researchers' Algorithm Sets ImageNet Challenge Milestone

Inside Microsoft Research 10 Feb 2015 6:00 AM 1

Like 91 Tweet

Posted by Richard Eckel



The race among computer scientists to build the world's most accurate **computer vision** system is more of a marathon than a sprint.

The race's new leader is a team of Microsoft researchers in Beijing, which this week published a paper in which they noted their computer vision system based on deep **convolutional neural networks** (CNNs) had for the first time eclipsed the abilities of people to classify objects defined in the ImageNet 1000 challenge.

http://blogs.technet.com/b/inside_microsoft_research/archive/2015/02/10/microsoft-researchers-algorithm-sets-imagenet-challenge-milestone.aspx

Leadertable (ImageNet image classification)

2015 ResNet (ILSVRC'15) 3.57

Year	Codename	Error (percent)	99.9% Conf Int
2014	**GoogLeNet**	**6.66**	**6.40 - 6.92**
2014	VGG	7.32	7.05 - 7.60
2014	MSRA	8.06	7.78 - 8.34
2014	AHoward	8.11	7.83 - 8.39
2014	DeeperVision	9.51	9.21 - 9.82
2013	Clarifai†	11.20	10.87 - 11.53
2014	CASIAWS†	11.36	11.03 - 11.69
2014	Trimps†	11.46	11.13 - 11.80
2014	Adobe†	11.58	11.25 - 11.91
2013	**Clarifai**	**11.74**	**11.41 - 12.08**
2013	NUS	12.95	12.60 - 13.30
2013	ZF	13.51	13.14 - 13.87
2013	AHoward	13.55	13.20 - 13.91
2013	OverFeat	14.18	13.83 - 14.54
2014	Orange†	14.80	14.43 - 15.17
2012	SuperVision†	15.32	14.94 - 15.69
2012	**SuperVision**	**16.42**	**16.04 - 16.80**
2012	ISI	26.17	25.71 - 26.65
2012	VGG	26.98	26.53 - 27.43
2012	XRCE	27.06	26.60 - 27.52
2012	UvA	29.58	29.09 - 30.04

Microsoft ResNet, a 152 layers network

GoogLeNet, 22 layers network

U. of Toronto, SuperVision, a 7 layers network

Unofficial human error is around *5.1%* on a subset

Why human error still? When labeling, human raters judged whether it belongs to a class (binary classification); the challenge is a 1000-class classification problem.

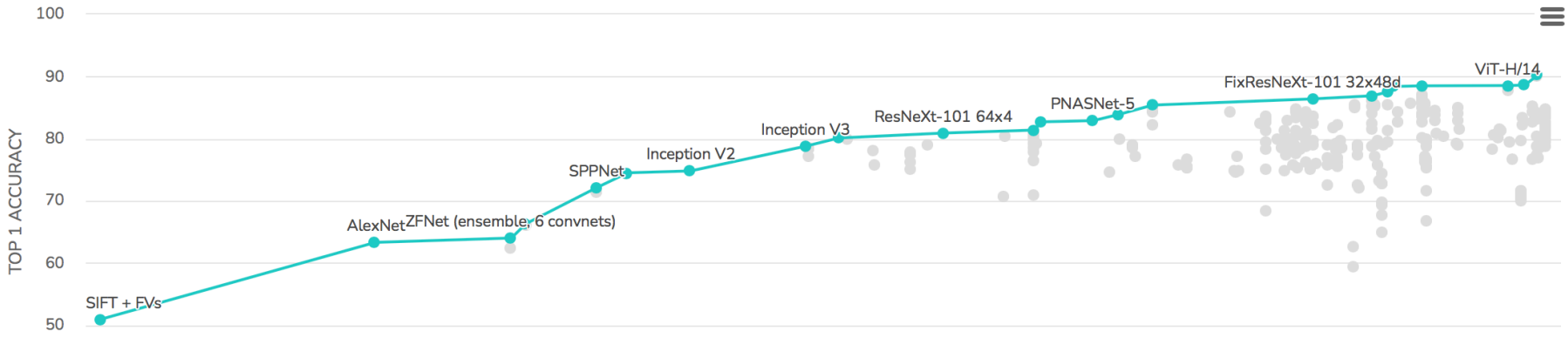
http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/)

Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

Top-1 accuracy as of 2021

Image Classification on ImageNet

Leaderboard Dataset

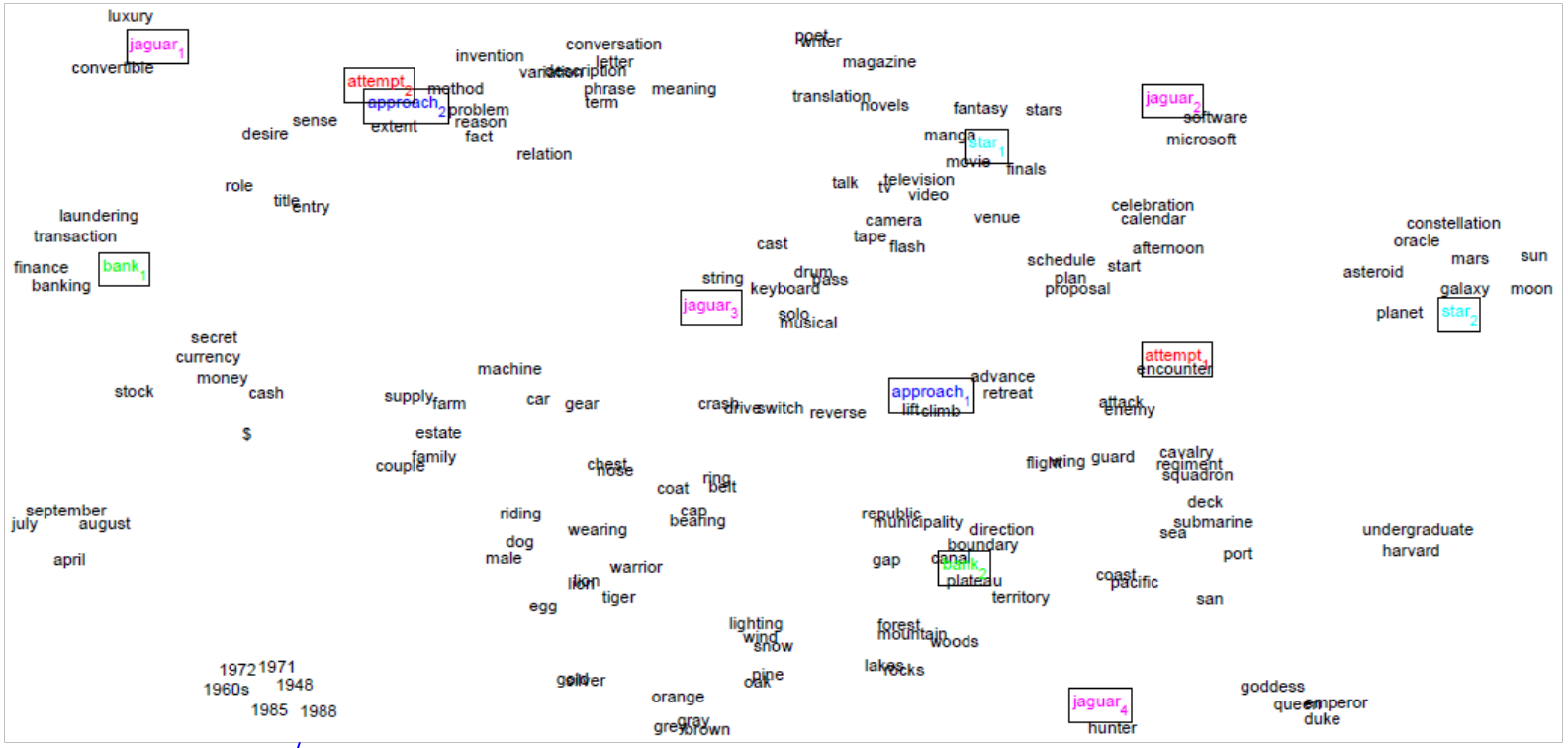


RANK	MODEL	TOP 1 ACCURACY	TOP 5 ACCURACY	NUMBER OF PARAMS	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	**Meta Pseudo Labels** (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels			2021
2	**Meta Pseudo Labels** (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels			2021
3	**EfficientNet-L2-475** (SAM)	88.61%		480M	✓	Sharpness-Aware Minimization for Efficiently Improving Generalization			2020
4	**ViT-H/14**	88.55%		632M	✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020

https://paperswithcode.com/sota/image-classification-on-imagenet

Deep Learning helps Natural Language Processing

From bag of words to distributed representation of words



Expression	*Nearest tokens*
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

https://code.google.com/p/word2vec/

SuperGlue

SuperGLUE, a new benchmark with a new set of more difficult language understanding tasks, improved resources, and a new public leaderboard.

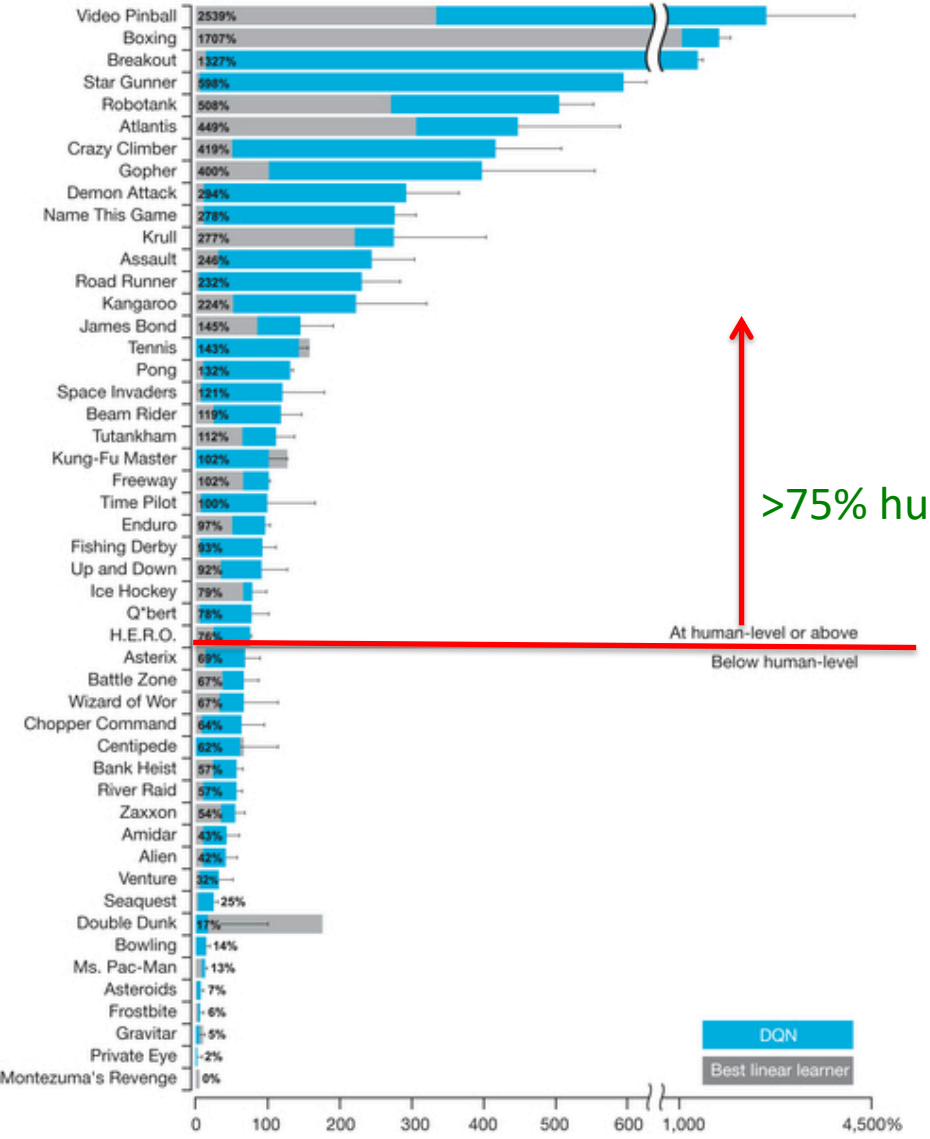
- Transformers
- BERT
- GPT-2

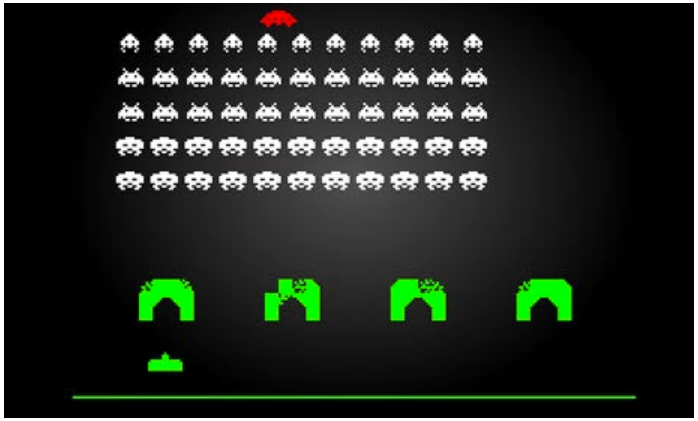
Leaderboard Version: **2.0**

Rank	Name	Model	URL	Score	BoolQ
1	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.3	90.4
2	Zirui Wang	T5 + Meena, Single Model (Meena Team - Google Brain)		90.2	91.3
3	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8	89.0
4	T5 Team - Google	T5		89.3	91.2
5	Huawei Noah's Ark Lab	NEZHA-Plus		86.7	87.8
6	Alibaba PAI&ICBU	PAI Albert		86.1	88.1
7	Tencent Jarvis Lab	RoBERTa (ensemble)		85.9	88.2
8	Zhuiyi Technology	RoBERTa-mtl-adv		85.7	87.1
9	Infosys : DAWN : AI Research	RoBERTa-iCETS		85.0	86.2
10	Facebook AI	RoBERTa		84.6	87.1
11	Anuar Sharafudinov	AILabs Team Transformers		77.5	88.1

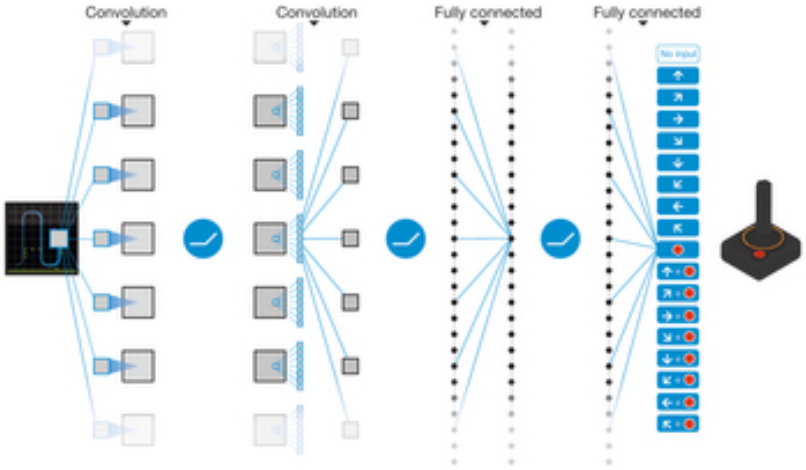
Wang, Alex, et al. "Superglue: A stickier benchmark for general-purpose language understanding systems." *arXiv preprint arXiv:1905.00537* (2019).

Deep Learning helps Control





>75% human level

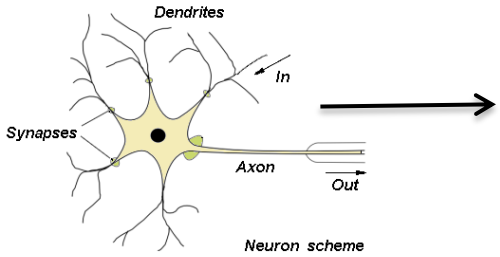


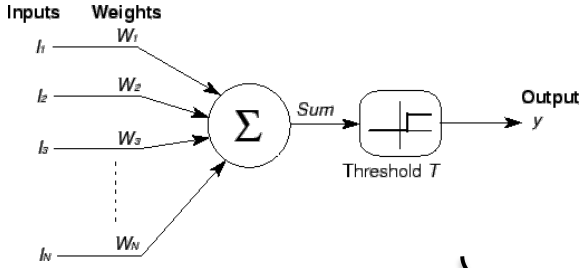
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html#videos

What is Deep Learning?

- "deep learning" has its roots in "neural networks"

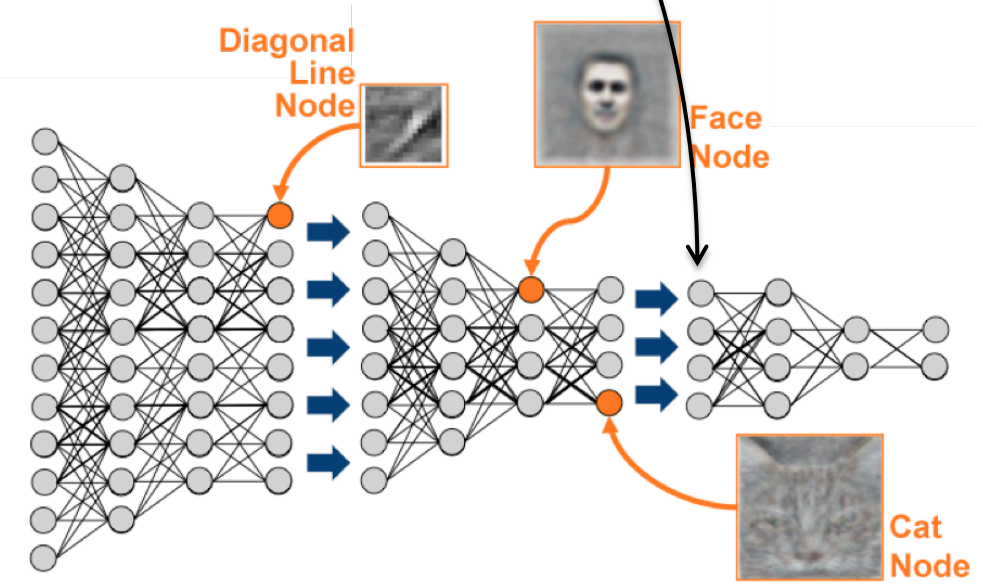
A biological neuron





A computerised artificial neuron

- deep learning creates many layers (deep) of neurons,attempting to learn structured representation of big data, layer by layer



Le, Quoc V. "Building high-level features using large scale unsupervised learning." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

Brief History

- The First wave
  - 1943 McCulloch and Pitts proposed the McCulloch-Pitts neuron model
  - 1958 Rosenblatt introduced the simple single layer networks now called Perceptrons.
  - 1969 Minsky and Papert's book Perceptrons demonstrated the limitation of single layer perceptrons, and almost the whole field went into hibernation.
- The Second wave
  - 1986 The Back-Propagation learning algorithm for Multi-Layer Perceptrons was rediscovered and the whole field took off again.
- The Third wave
  - 2006 Deep (neural networks) Learning gains popularity and
  - 2012 made significant break-through in many applications.

Summay

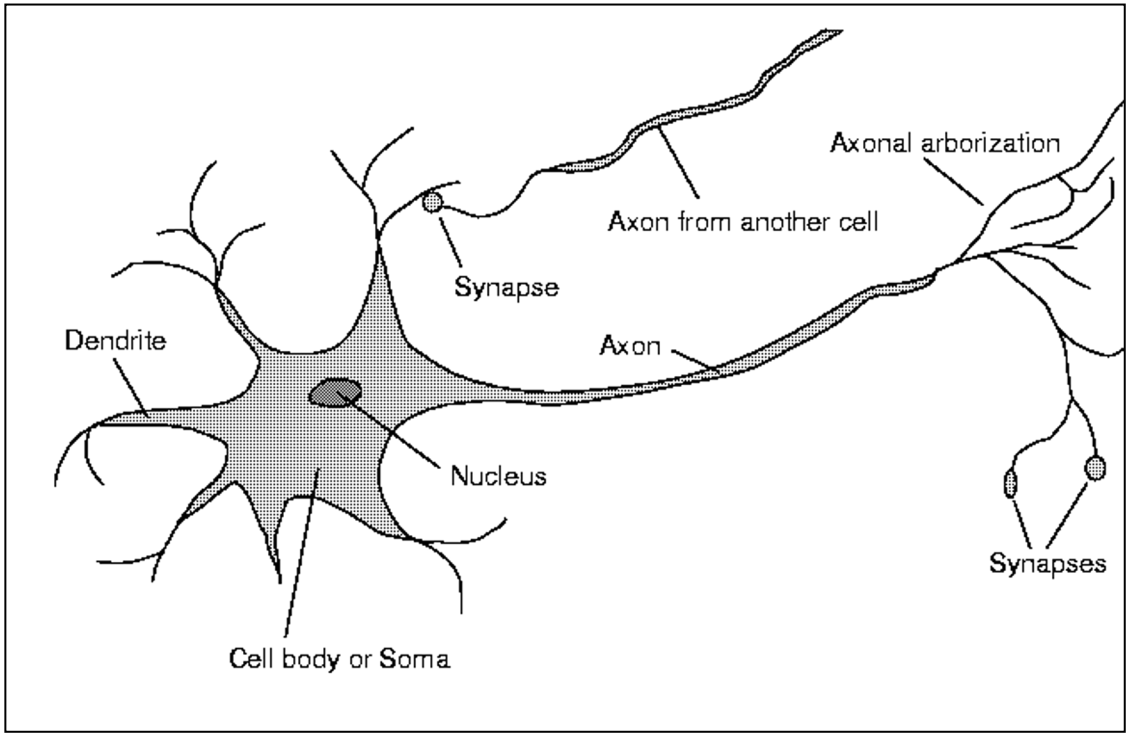
- Neural networks basics
  - its backpropagation learning algorithm
- Word embedding
  - distributed representation of discrete features (tokens)
  - applications：video/item recommendation, click-through rate estimation
- Go deeper in layers
- Convolutional neural networks (CNN)
  - address feature dependency
  - Applications: text classification
- Recurrent neural networks (RNN)
  - address time-dependency
  - applications：language models and multi-modal modelling
- Web search revisited
- Deep reinforcement learning (only focus on MCTS)
  - have the ability of real-time control

Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

Early Artificial Neurons (1943-1969)

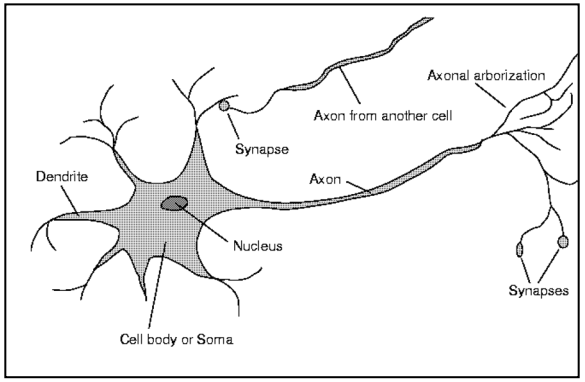
- A biological neuron

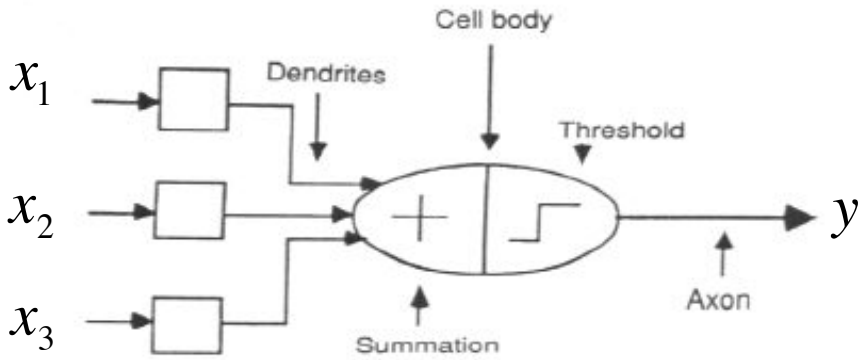


- In the human brain, a typical neuron collects signals from others through a host of fine structures called ***dendrites***
- The neuron sends out spikes of electrical activity through a long, thin stand known as an ***axon***, which splits into thousands of branches
- At the end of each branch, a structure called a ***synapse*** converts the activity from the axon into electrical effects that inhibit or excite activity in the connected neurons.

Early Artificial Neurons (1943-1969)

- McCulloch- Pitts neuron





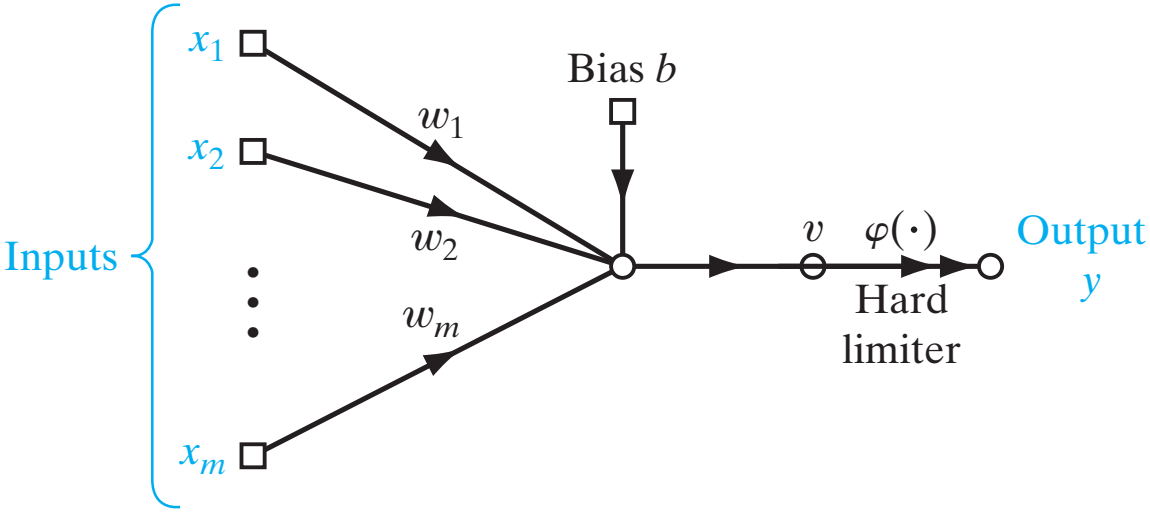
$$y = f\left(\sum_m w_m x_m - b\right) \qquad f(x) = \begin{cases} 1 & if\ x \geq 0; \\ 0 & otherwise \end{cases}$$

- McCulloch and Pitts [1943] proposed a simple model of a neuron as computing machine

- The artificial neuron computes a weighted sum of its inputs from other neurons, and outputs a one or a zero according to whether the sum is above or below a certain threshold

Early Artificial Neurons (1943-1969)

- Rosenblatt's single layer perceptron



Activation function:

$$\varphi(x) = \begin{cases} 1 & if\ x \geq 0; \\ -1 & otherwise \end{cases}$$

$$y = \varphi(\sum_m w_m x_m - b)$$

- Rosenblatt [1958] further proposed the *perceptron* as the first model for learning with a teacher (i.e., supervised learning)

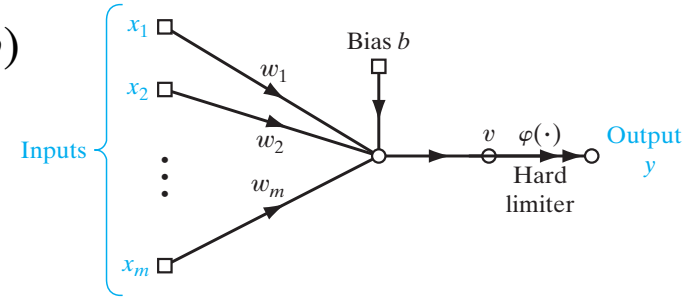
Focused on how to find appropriate weights w_m for two-class classification task (y = 1: class one and y = -1: class two)

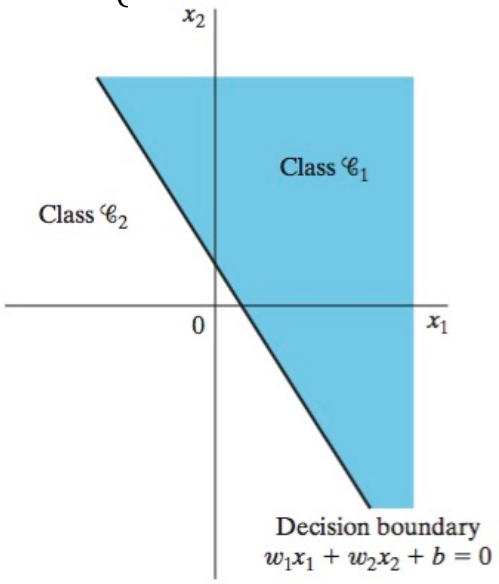
Early Artificial Neurons (1943-1969)

- Rosenblatt's single layer perceptron

$$y = \varphi(\sum_m w_m x_m + b)$$

$$\varphi(x) = \begin{cases} 1 & if\ x \geq 0; \\ -1 & otherwise \end{cases}$$





- Proved the convergence of a *learning algorithm* if two classes said to be *linearly separable* (i.e., patterns that lie on opposite sides of a hyperplane)
- Many people hope that such machines could be a basis for artificial intelligence

$$y^{(t)} = \varphi(\sum_m w_m^{(t)} x_m^{(t)} + b^{(t)}),\ t = \{1,...,T\}$$

$d^{(t)}$ is training label and η is a parameter (the learning rate)

$$\Delta w_m^{(t)} = \eta(d^{(t)} - y^{(t)}) x_m^{(t)}$$

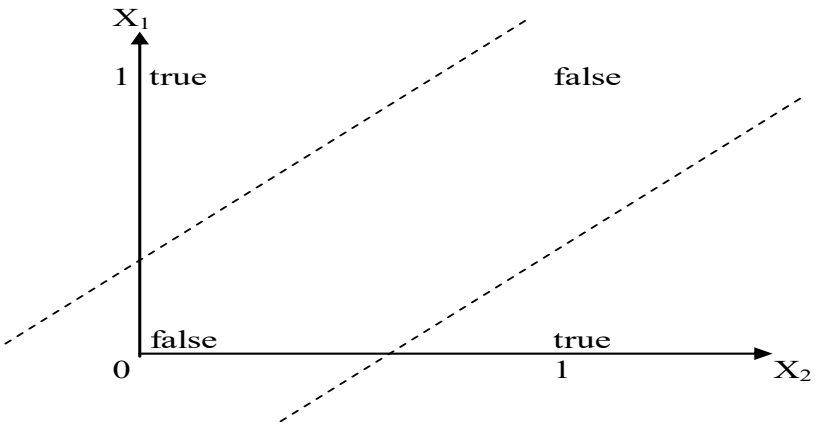
$$w_m^{(t+1)} = w_m^{(t+1)} + \Delta w_m^{(t)}$$

Haykin S S, Haykin S S, Haykin S S, et al. Neural networks and learning machines[M]. Upper Saddle River: Pearson Education, 2009.

Early Artificial Neurons (1943-1969)

- The XOR problem

Input x		Output y
X_1	X_2	X_1 XOR X_2
0	0	0
0	1	1
1	0	1
1	1	0

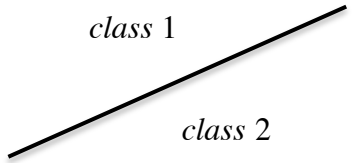


XOR is non linearly separable: These two classes (true and false) cannot be separated using a line.

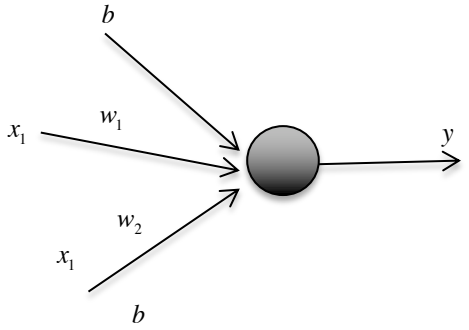
- However, Minsky and Papert [1969] shows that some rather elementary computations, such as *XOR* problem, could not be done by Rosenblatt's one-layer perceptron
- However Rosenblatt believed the limitations could be overcome if more layers of units to be added, but no learning algorithm known to obtain the weights yet
- Due to the lack of learning algorithms people left the neural network paradigm for almost 20 years

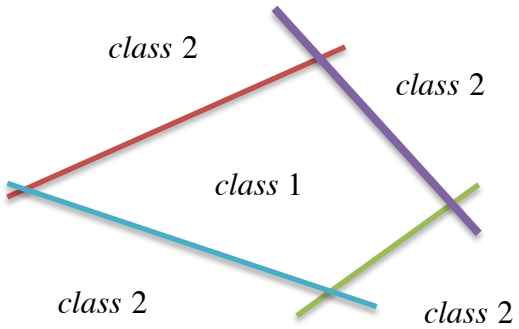
Hidden layers and the backpropagation (1986~)

- Adding hidden layer(s) (internal presentation) allows to learn a mapping that is not constrained by "*linearly separable*"

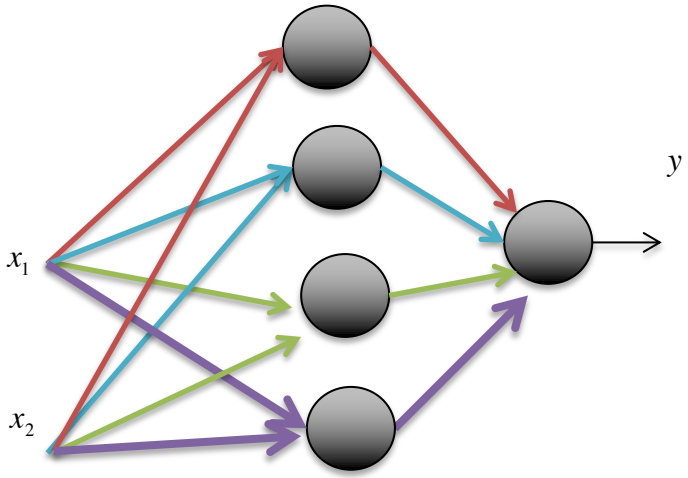


decision boundary: $x_1 w_1 + x_2 w_2 + b = 0$





Each hidden node realises one of the lines bounding the convex region

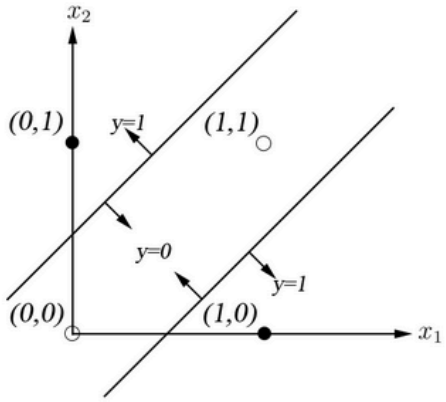


Hidden layers and the backpropagation (1986~)

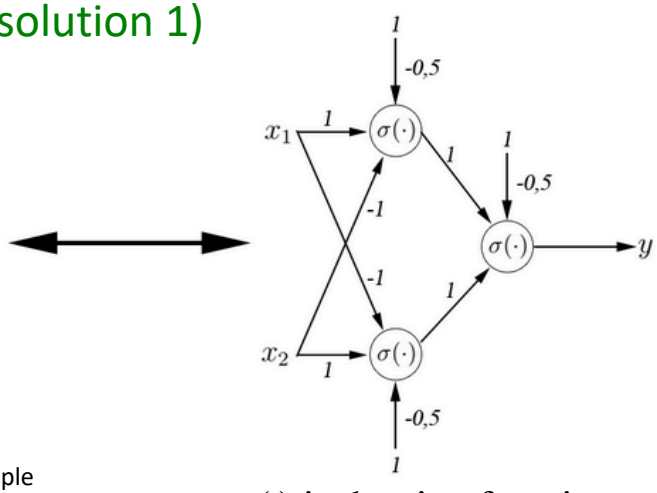
- But the solution is quite often not unique

Input x		Output y
X_1	X_2	X_1 XOR X_2
0	0	0
0	1	1
1	0	1
1	1	0

(solution 1)

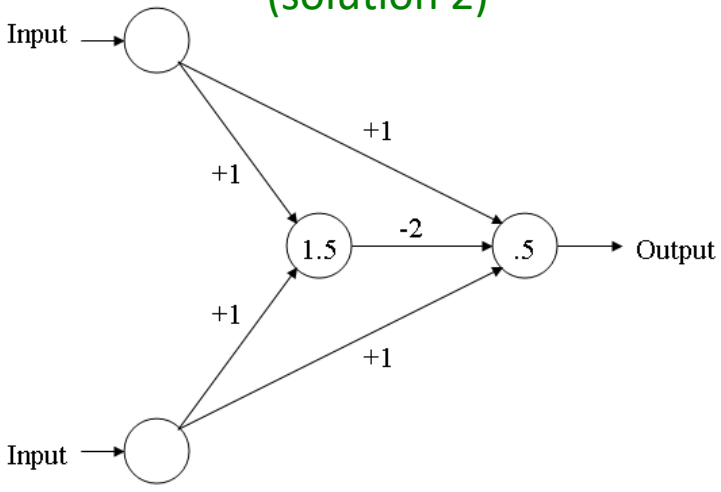


two lines are necessary to divide the sample space accordingly



$\varphi(\cdot)$ is the sign function

(solution 2)



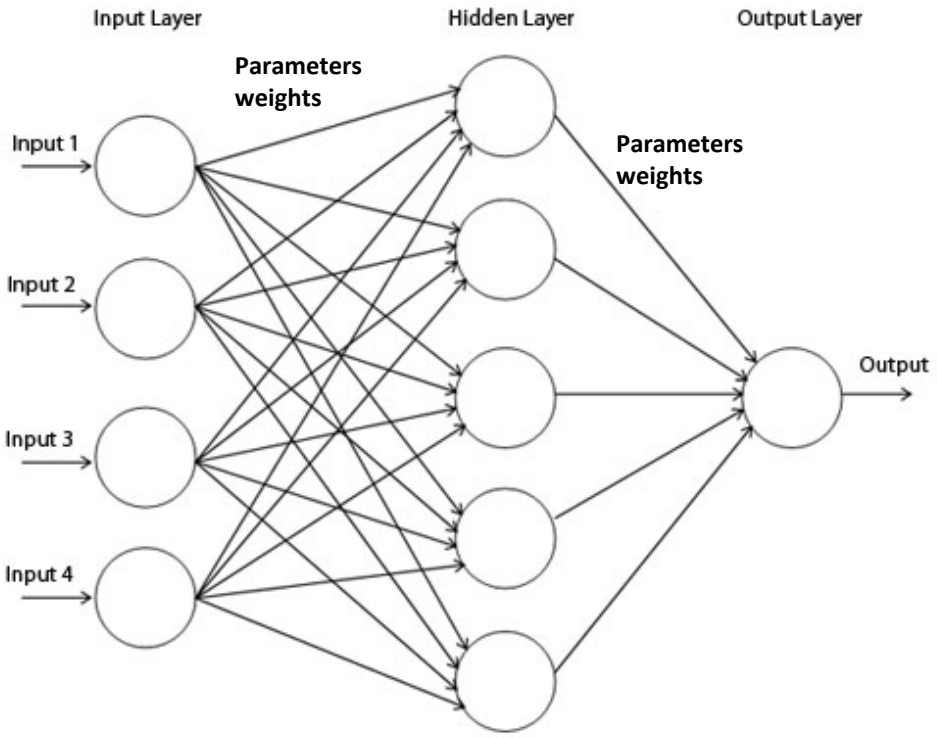
The number in the circle is a threshold

http://recognize-speech.com/basics/introduction-to-artificial-neural-networks

http://www.cs.stir.ac.uk/research/publications/techreps/pdf/TR148.pdf

Hidden layers and the backpropagation (1986~)

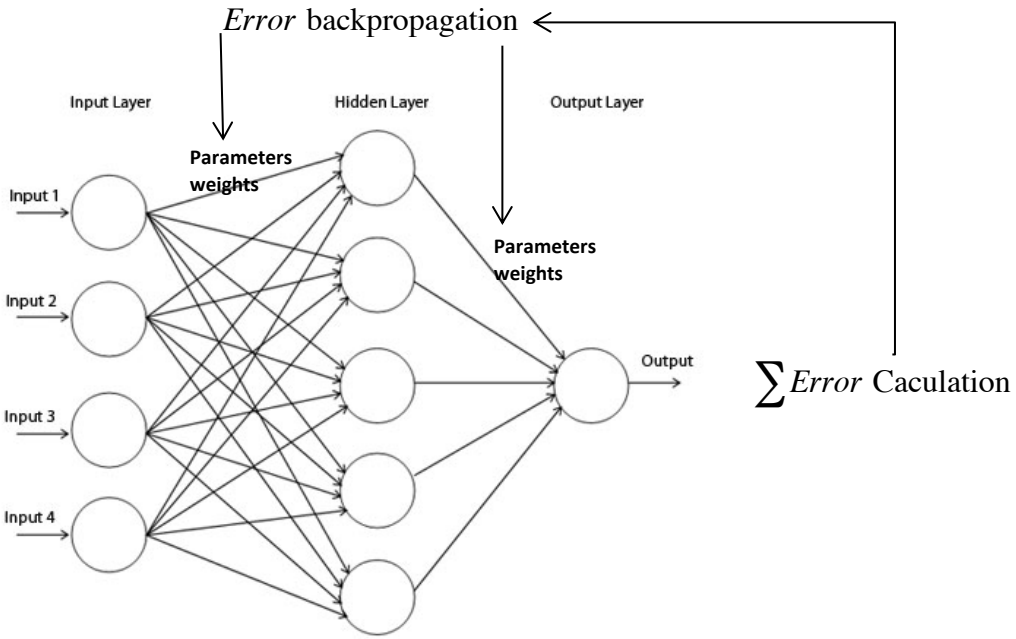
- Feedforward: massages moves forward from the input nodes, through the hidden nodes (if any), and to the output nodes. There are no cycles or loops in the network



Two-layer feedforward neural network

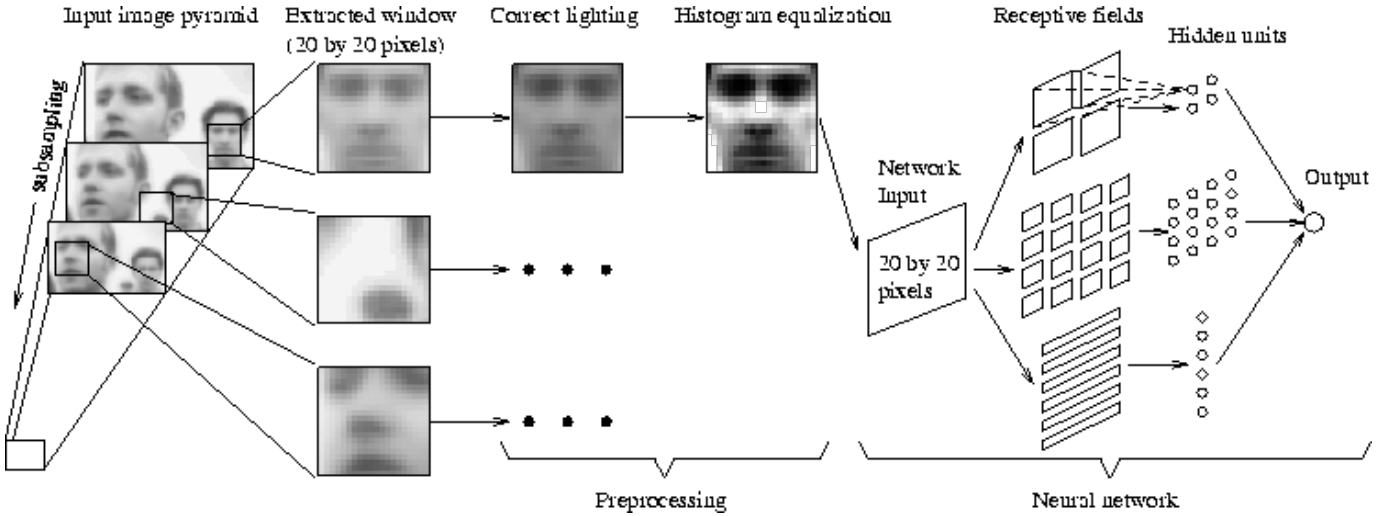
Hidden layers and the backpropagation (1986~)

- One of the efficient algorithms for multi-layer neural networks is the *Backpropagation* algorithm
- It was re-introduced in 1986 and Neural Networks regained the popularity

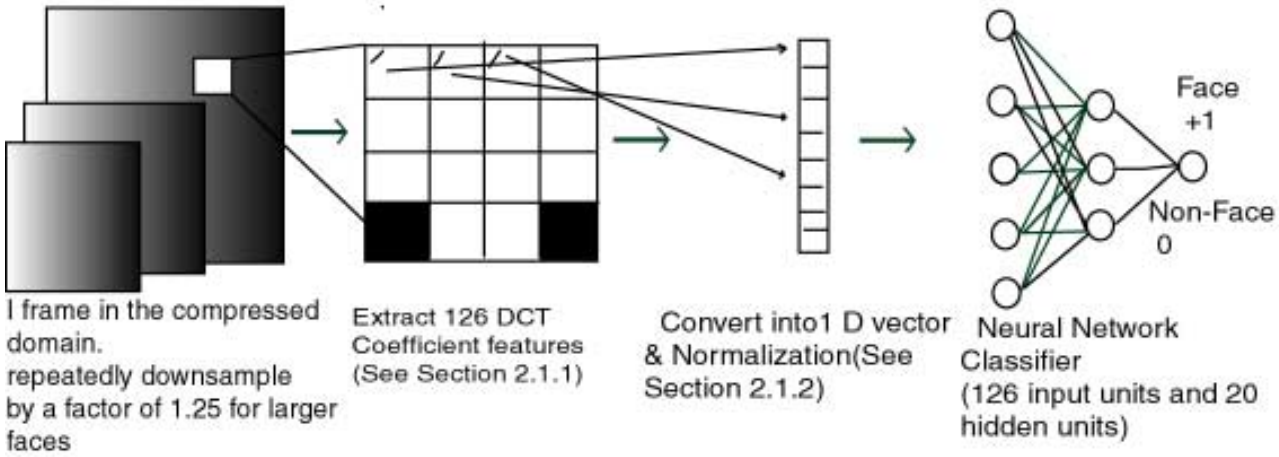


Note: *backpropagation* appears to be found by Werbos [1974]; and then independently rediscovered around 1985 by Rumelhart, Hinton, and Williams [1986] and by Parker [1985]

Example Application: Face Detection

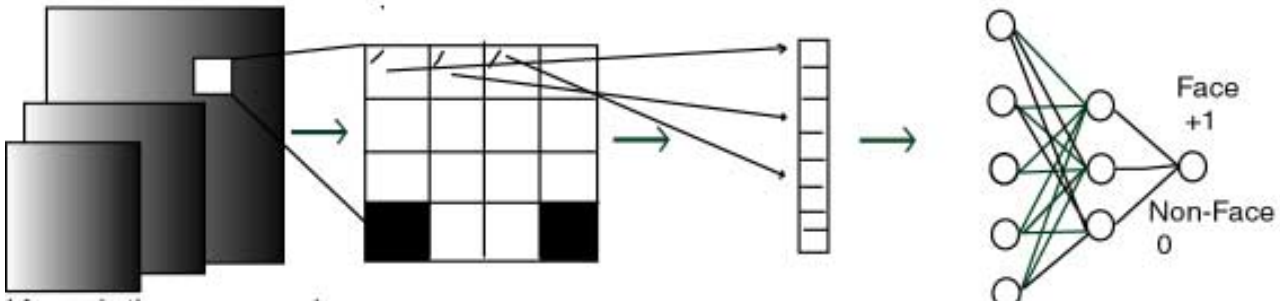


Rowley, Henry A., Shumeet Baluja, and Takeo Kanade. "Neural network-based face detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on 20.1 (1998): 23-38.



Wang J, Kankanhalli M S, Mulhem P, et al. Face detection using DCT coefficients in MPEG video[C]//Proceedings of the International Workshop on Advanced Imaging Technology (IWAIT 2002). 2002: 66-70.

Example Application: Face Detection

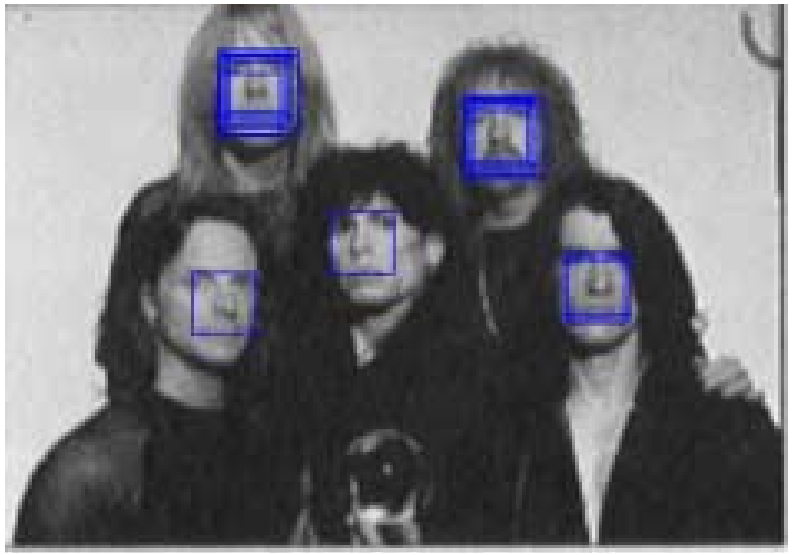


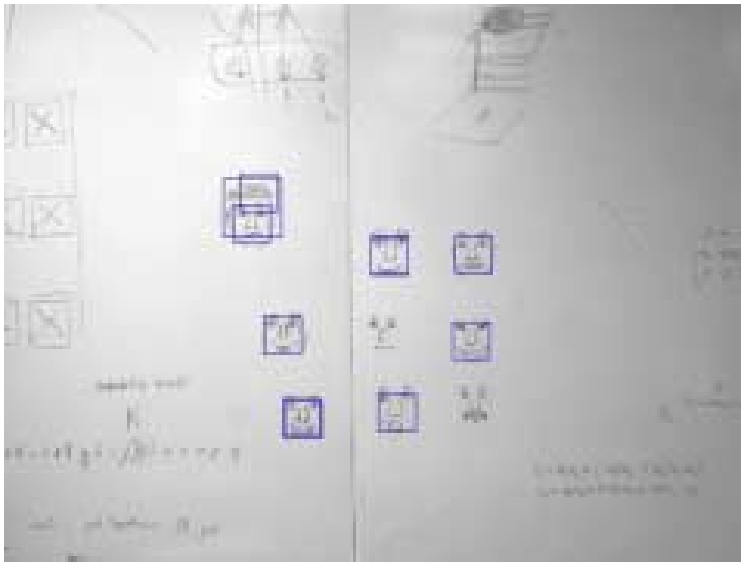
I frame in the compressed domain.
repeatedly downsample by a factor of 1.25 for larger faces

Extract 126 DCT Coefficient features (See Section 2.1.1)

Convert into1 D vector & Normalization(See Section 2.1.2)

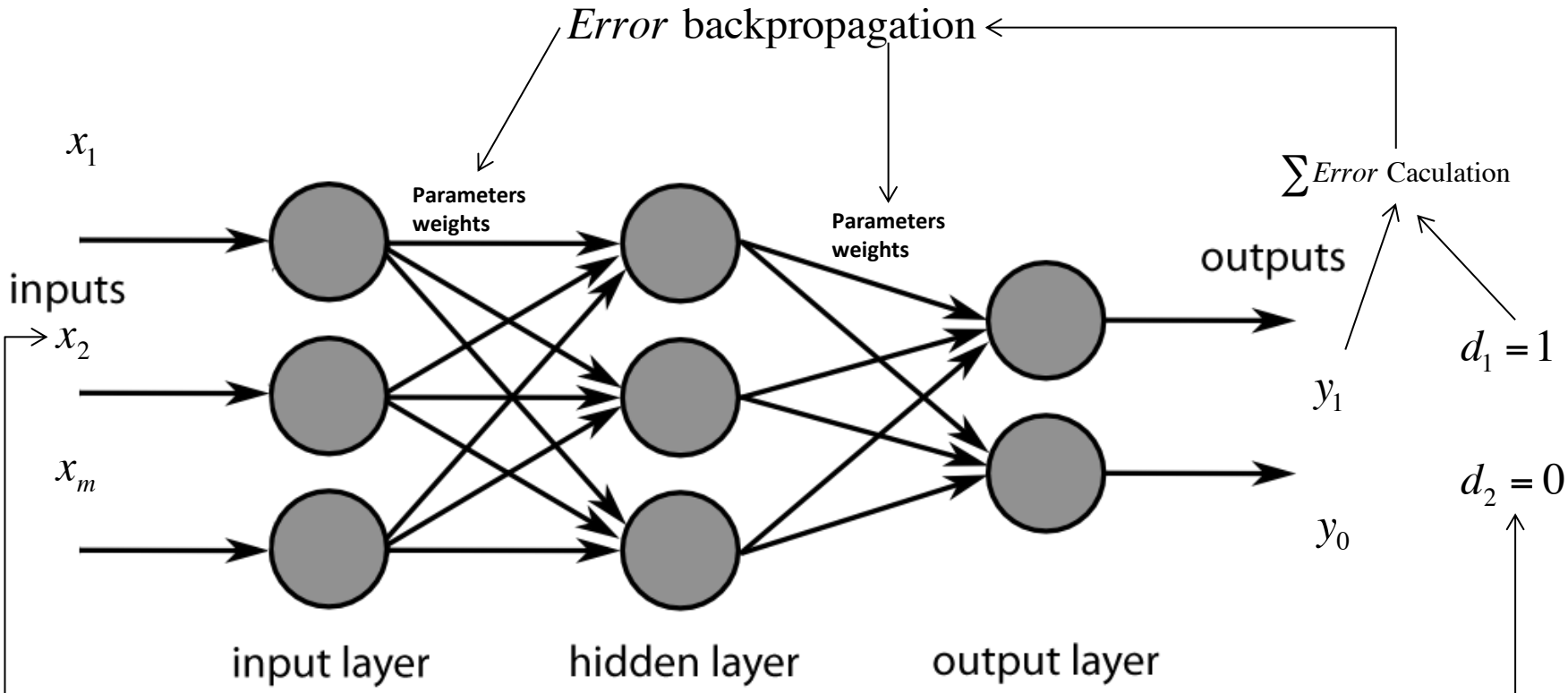
Neural Network Classifier (126 input units and 20 hidden units)





Wang J, Kankanhalli M S, Mulhem P, et al. Face detection using DCT coefficients in MPEG video[C]//Proceedings of the International Workshop on Advanced Imaging Technology (IWAIT 2002). 2002: 66-70.

how it works (training)

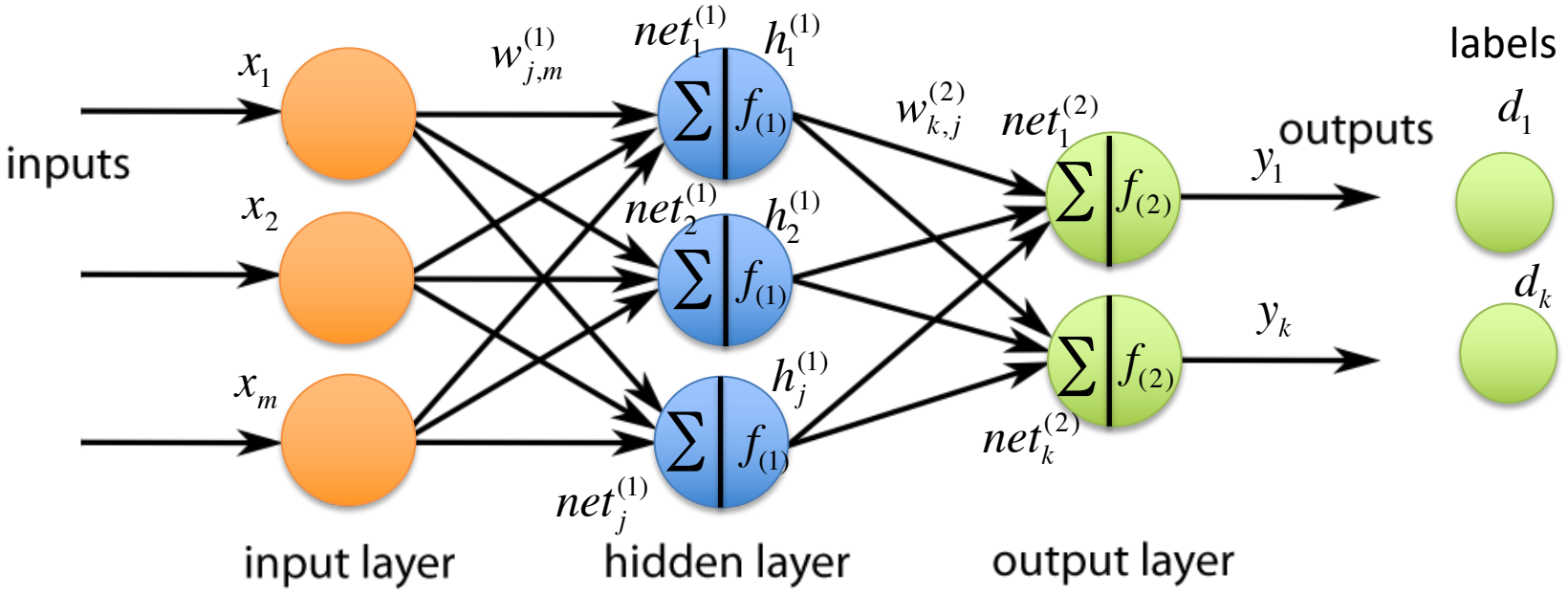


label = *Face*

label = *no face*

Training instances…

When make a prediction



Two-layer feedforward neural network

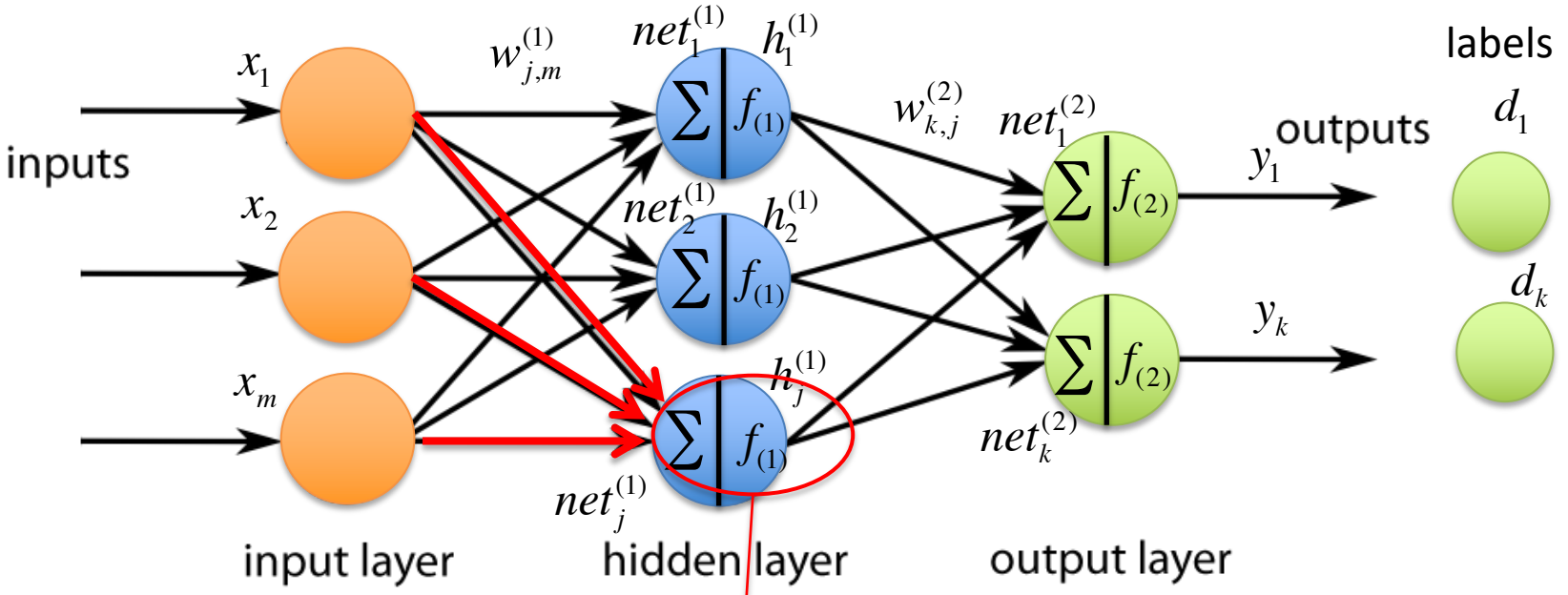
Feed-forward/prediction:

$$x = (x_1, \ldots, x_m) \xrightarrow{h^1_j = f_1(net^{(1)}_j) = f_1(\sum_m w^{(1)}_{j,m} x_m)} h^1_j \xrightarrow{y_k = f_2(net^{(1)}_k) = f_2(\sum_j w^{(2)}_{k,j} h^{(1)}_j)} y_k$$

where $net^{(1)}_j \equiv \sum_m w^{(1)}_{j,m} x_m$

$net^{(2)}_k \equiv \sum_j w^{(2)}_{k,j} h^{(1)}_j$

When make a prediction



Two-layer feedforward neural network

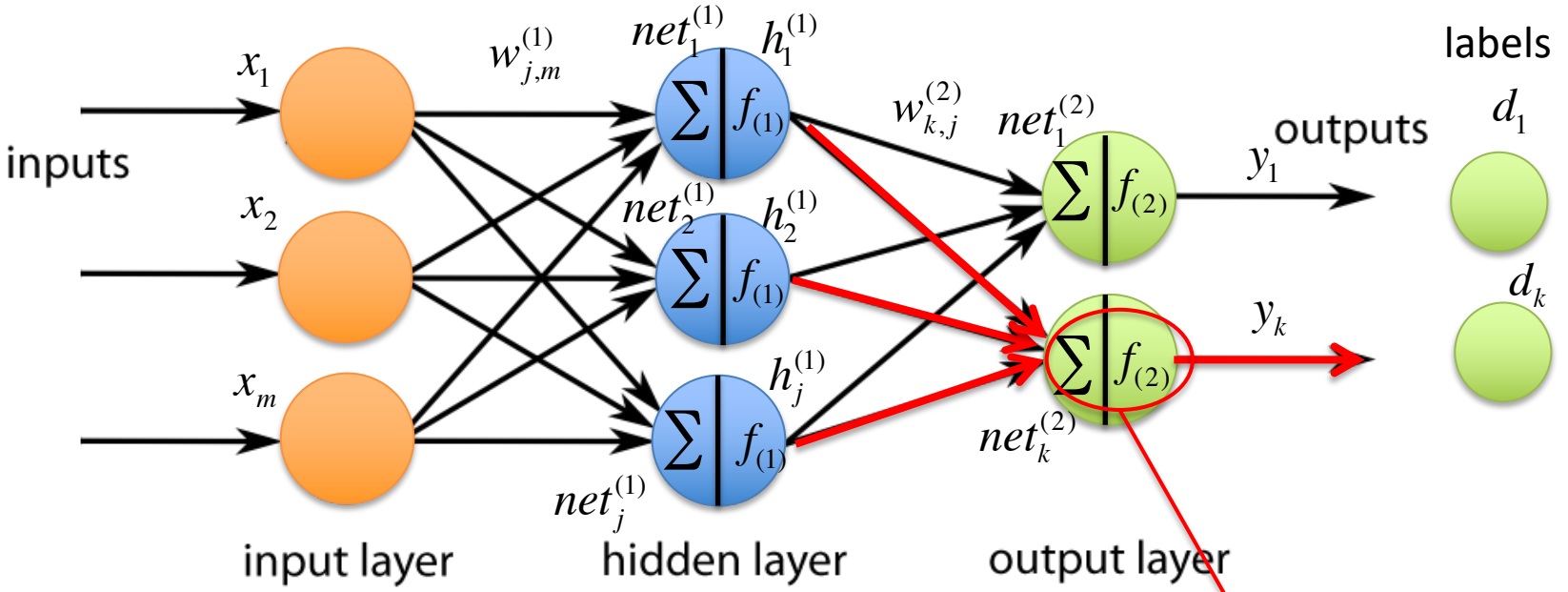
Feed-forward/prediction:

$$x = (x_1, \ldots, x_m) \rightarrow h^1_j = f_1(net^{(1)}_j) = f_1(\sum_m w^{(1)}_{j,m} x_m) \rightarrow y_k = f_2(net^{(1)}_k) = f_2(\sum_j w^{(2)}_{k,j} h^{(1)}_j)$$

where $net^{(1)}_j \equiv \sum_m w^{(1)}_{j,m} x_m$

$net^{(2)}_k \equiv \sum_j w^{(2)}_{k,j} h^{(1)}_j$

When make a prediction



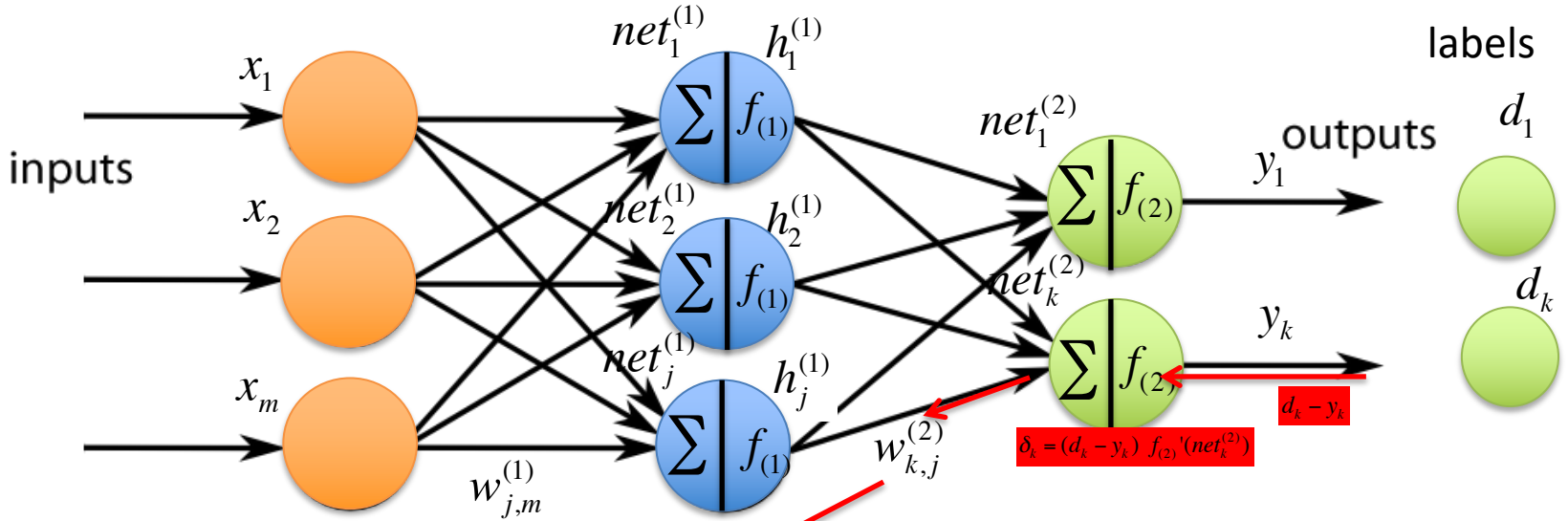
Two-layer feedforward neural network

Feed-forward/prediction: $h^1_j = f_1(net^{(1)}_j) = f_1(\sum_m w^{(1)}_{j,m} x_m)$ $\quad y_k = f_2(net^{(1)}_k) = f_2(\sum_j w^{(2)}_{k,j} h^{(1)}_j)$

$x = (x_1, \ldots, x_m) \longrightarrow h^1_j \longrightarrow y_k$

where $net^{(1)}_j \equiv \sum_m w^{(1)}_{j,m} x_m$ $\quad net^{(2)}_k \equiv \sum_j w^{(2)}_{k,j} h^{(1)}_j$

When backprop/learn the parameters



Two layer feedforward neural network

Notation:

$$net_j^{(1)} \equiv \sum_m w_{j,m}^{(1)} x_m \qquad net_k^{(2)} \equiv \sum_j w_{k,j}^{(2)} h_j^{(1)} \qquad \eta : \text{the given learning rate}$$

Backprop to learn the parameters:

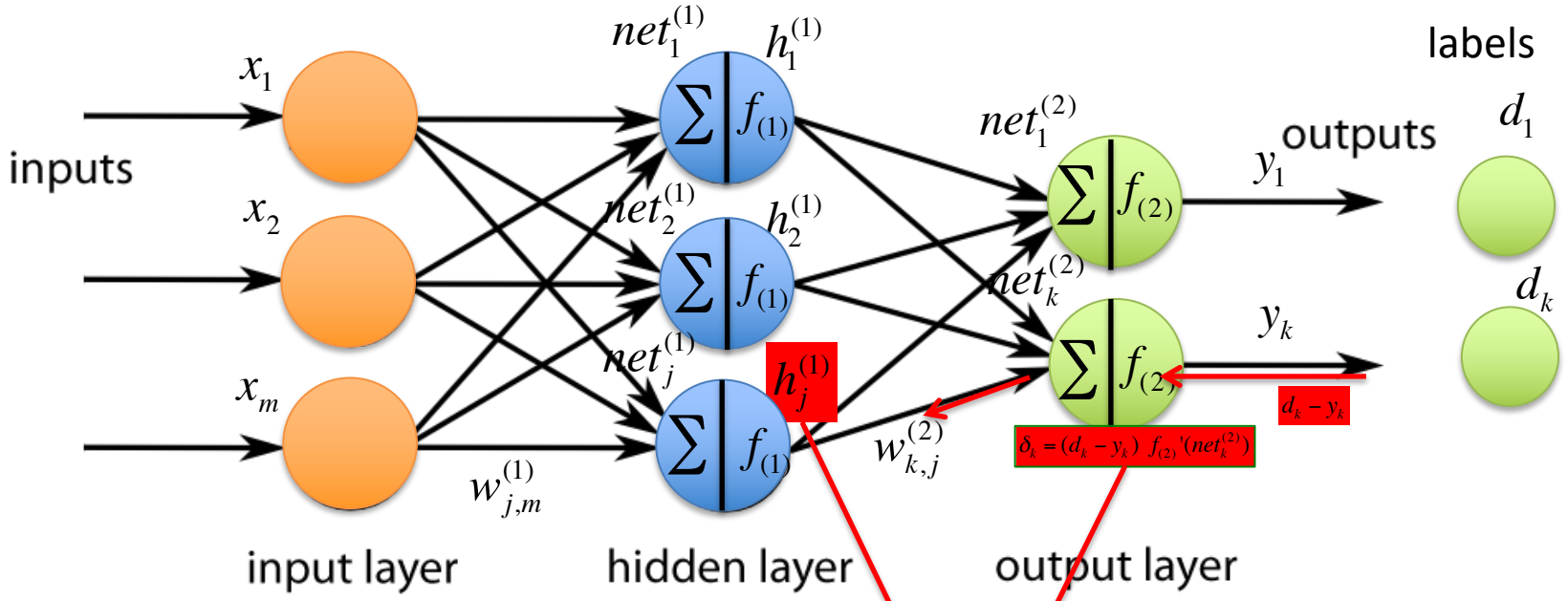
output neuron:

$$w_{k,j}^{(2)} = w_{k,j}^{(2)} + \Delta w_{k,j}^{(2)} \qquad \Delta w_{k,j}^{(2)} = \eta Error_k Output_j = \eta \delta_k h_j^{(1)} \qquad E(W) = \frac{1}{2}\sum_k (y_k - d_k)^2$$

$$\Delta w_{k,j}^{(2)} \equiv -\eta \frac{\partial E(W)}{\partial w_{k,j}^{(2)}} = -\eta (y_k - d_k)\ \frac{\partial y_k}{\partial net_k^{(2)}} \frac{\partial net_k^{(2)}}{\partial w_{k,j}^{(2)}} = \eta (d_k - y_k)\ f_{(2)}'(net_k^{(2)}) h_j^{(1)} = \eta \delta_k h_j^{(1)}$$

$$Error : \delta_k \equiv (d_k - y_k)\ f_{(2)}'(net_k^{(2)})$$

When backprop/learn the parameters



Two layer feedforward neural network

Notation:

$$net_j^{(1)} \equiv \sum_m w_{j,m}^{(1)} x_m \qquad net_k^{(1)} \equiv \sum_j w_{k,j}^{(2)} h_j^{(1)}$$

η : the given learning rate

Backprop to learn the parameters:

output neuron:

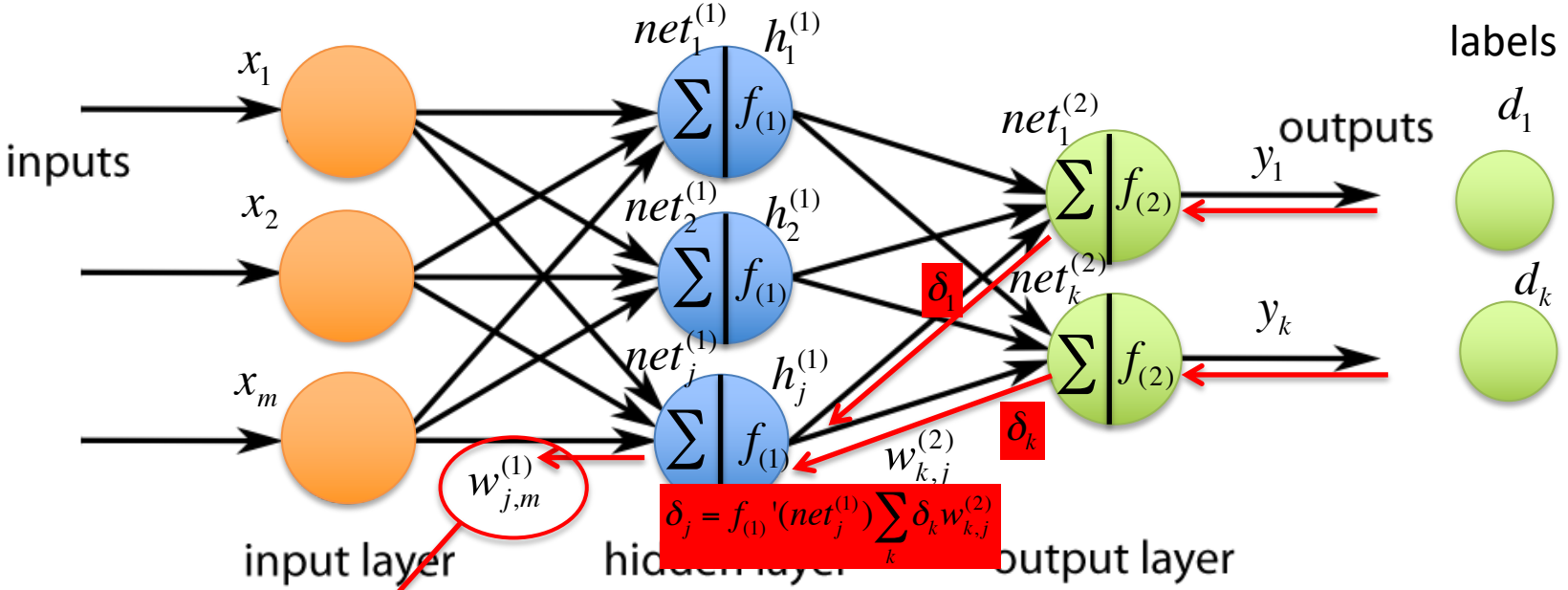
$$w_{k,j}^{(2)} = w_{k,j}^{(2)} + \Delta w_{k,j}^{(2)} \qquad \Delta w_{k,j}^{(2)} = \eta Error_k Output_j = \eta \delta_k h_j^{(1)}$$

$$E(W) = \frac{1}{2}\sum_k (y_k - d_k)^2$$

$$\Delta w_{k,j}^{(2)} \equiv -\eta \frac{\partial E(W)}{\partial w_{k,j}^{(2)}} = -\eta (y_k - d_k)\ \frac{\partial y_k}{\partial net_k^{(2)}} \frac{\partial net_k^{(2)}}{\partial w_{k,j}^{(2)}} = \eta (d_k - y_k)\ f_{(2)}'(net_k^{(2)}) h_j^{(1)} = \eta \delta_k h_j^{(1)}$$

$$Error: \delta_k \equiv (d_k - y_k)\ f_{(2)}'(net_k^{(2)})$$

Multi-layer feedforward neural network



Two layer feedforward neural network

Notation:

$$net_j^{(1)} \equiv \sum_m w_{j,m}^{(1)} x_m \qquad net_k^{(1)} \equiv \sum_j w_{k,j}^{(2)} h_j^{(1)}$$

η : the given learning rate

Backprop to learn the parameters:

hidden neuron:

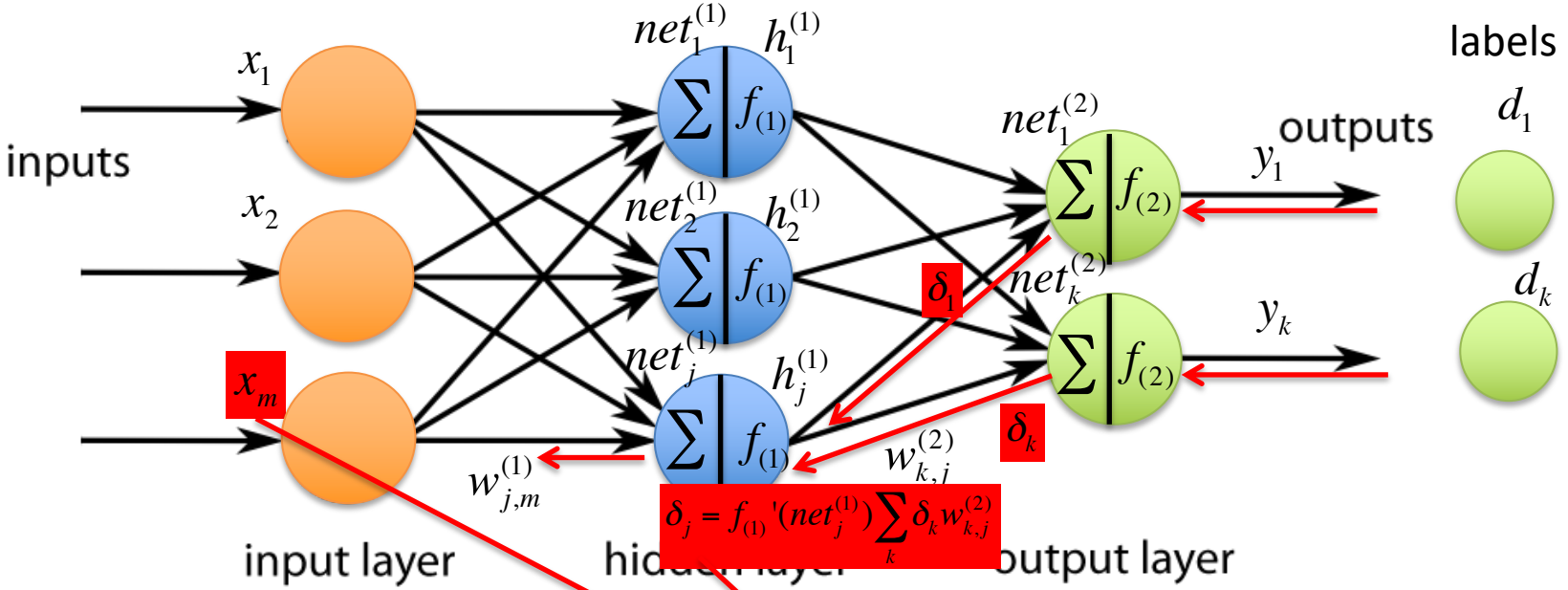
$$w_{j,m}^{(1)} = \Delta w_{j,m}^{(1)} + \Delta w_{j,m}^{(1)} \qquad \Delta w_{j,m}^{(1)} = \eta Error_j Output_m = \eta \delta_j x_m$$

$$E(W) = \frac{1}{2} \sum_k (y_k - d_k)^2$$

$$\Delta w_{j,m}^{(1)} = -\eta \frac{\partial E(W)}{\partial w_{j,m}^{(1)}} = -\eta \frac{\partial E(W)}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial w_{j,m}^{(1)}} = \eta [\sum_k (d_k - y_k)\ f_{(2)}'(net_k^{(1)}) w_{k,j}^{(2)}][x_m f_{(1)}'(net_j^{(1)})] = \eta \delta_j x_m$$

$$Error: \delta_j \equiv f_{(1)}'(net_j^{(1)}) \sum_k (d_k - y_k)\ f_{(2)}'(net_k^{(2)}) w_{k,j}^{(2)}] = f_{(1)}'(net_j^{(1)}) \sum_k \delta_k w_{k,j}^{(2)}$$

Multi-layer feedforward neural network



Two layer feedforward neural network

Notation:

$$net_j^{(1)} \equiv \sum_m w_{j,m}^{(1)} x_m \qquad net_k^{(1)} \equiv \sum_j w_{k,j}^{(2)} h_j^{(1)}$$

η : the given learning rate

Backprop to learn the parameters:

hidden neuron:

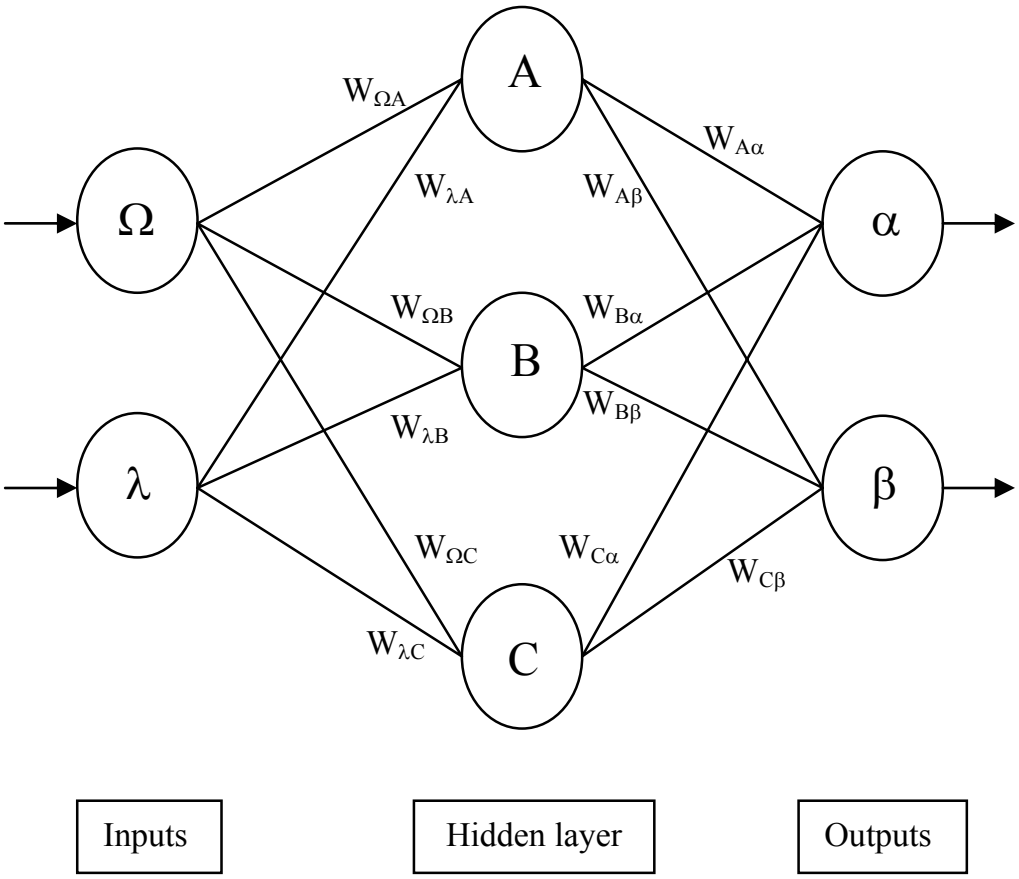
$$w_{j,m}^{(1)} = \Delta w_{j,m}^{(1)} + \Delta w_{j,m}^{(1)} \qquad \Delta w_{j,m}^{(1)} = \eta Error_j Output_m = \eta \delta_j x_m$$

$$E(W) = \frac{1}{2} \sum_k (y_k - d_k)^2$$

$$\Delta w_{j,m}^{(1)} = -\eta \frac{\partial E(W)}{\partial w_{j,m}^{(1)}} = -\eta \frac{\partial E(W)}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial w_{j,m}^{(1)}} = \eta [\sum_k (d_k - y_k)\ f_{(2)}'(net_k^{(1)}) w_{k,j}^{(2)}][x_m f_{(1)}'(net_j^{(1)})] = \eta \delta_j x_m$$

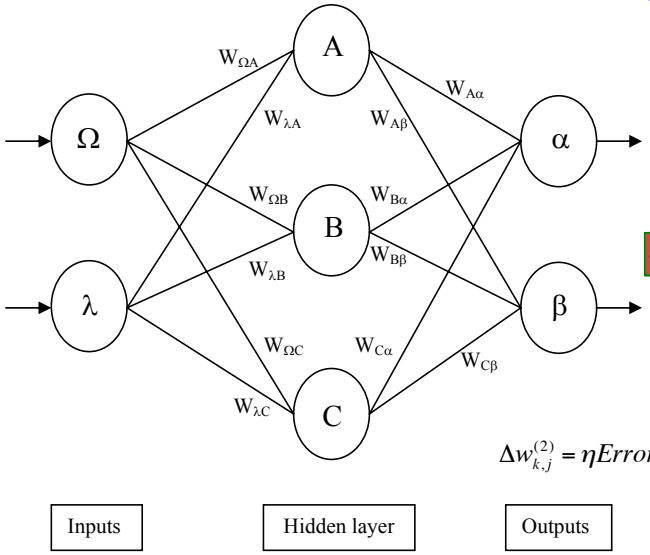
$$Error: \delta_j \equiv f_{(1)}'(net_j^{(1)}) \sum_k (d_k - y_k)\ f_{(2)}'(net_k^{(2)}) w_{k,j}^{(2)}] = f_{(1)}'(net_j^{(1)}) \sum_k \delta_k w_{k,j}^{(2)}$$

An example for Backprop



https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf

An example for Backprop



1. Calculate errors of output neurons

$$\delta_k = (d_k - y_k)\, f_{(2)}'(net_k^{(2)})$$

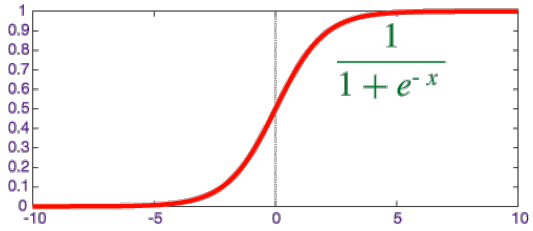
$$\delta_\alpha = out_\alpha\,(1 - out_\alpha)\,(Target_\alpha - out_\alpha)$$
$$\delta_\beta = out_\beta\,(1 - out_\beta)\,(Target_\beta - out_\beta)$$

2. Change output layer weights

$$\Delta w_{k,j}^{(2)} = \eta Error_k Output_j = \eta \delta_k h_j^{(1)}$$

$$W^+_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\, out_A \qquad W^+_{A\beta} = W_{A\beta} + \eta\delta_\beta\, out_A$$
$$W^+_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\, out_B \qquad W^+_{B\beta} = W_{B\beta} + \eta\delta_\beta\, out_B$$
$$W^+_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\, out_C \qquad W^+_{C\beta} = W_{C\beta} + \eta\delta_\beta\, out_C$$

Consider sigmoid activation function $f_{Sigmoid}(x) = \dfrac{1}{1+e^{-x}}$



3. Calculate (back-propagate) hidden layer errors

$$\delta_j = f_{(1)}'(net_j^{(1)}) \sum_k \delta_k w_{k,j}^{(2)}$$

$$\delta_A = out_A\,(1 - out_A)\,(\delta_\alpha W_{A\alpha} + \delta_\beta W_{A\beta})$$
$$\delta_B = out_B\,(1 - out_B)\,(\delta_\alpha W_{B\alpha} + \delta_\beta W_{B\beta})$$
$$\delta_C = out_C\,(1 - out_C)\,(\delta_\alpha W_{C\alpha} + \delta_\beta W_{C\beta})$$

4. Change hidden layer weights

$$\Delta w_{j,m}^{(1)} = \eta Error_j Output_m = \eta \delta_j x_m$$

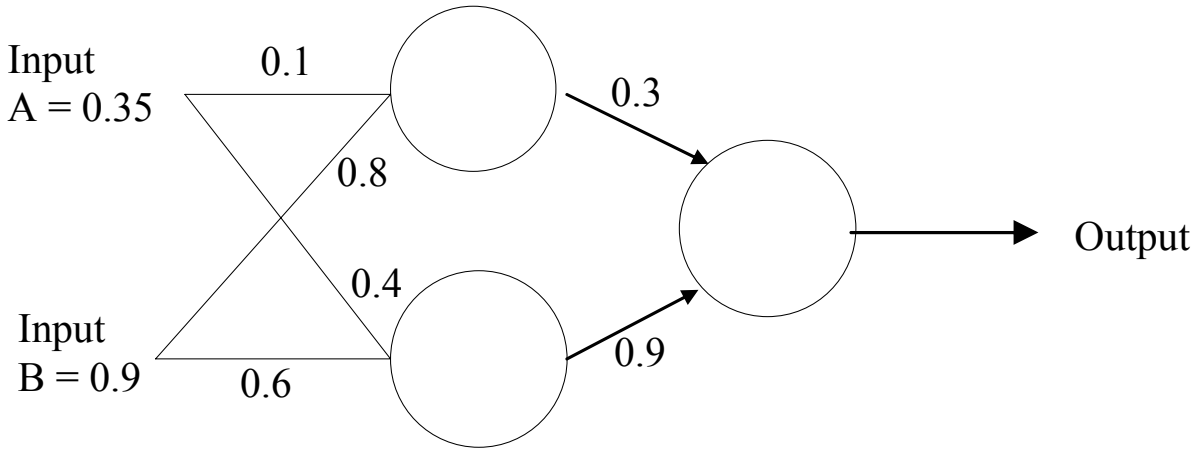
$$W^+_{\lambda A} = W_{\lambda A} + \eta\delta_A\, in_\lambda \qquad W^+_{\Omega A} = W^+_{\Omega A} + \eta\delta_A\, in_\Omega$$
$$W^+_{\lambda B} = W_{\lambda B} + \eta\delta_B\, in_\lambda \qquad W^+_{\Omega B} = W^+_{\Omega B} + \eta\delta_B\, in_\Omega$$
$$W^+_{\lambda C} = W_{\lambda C} + \eta\delta_C\, in_\lambda \qquad W^+_{\Omega C} = W^+_{\Omega C} + \eta\delta_C\, in_\Omega$$

$$f'_{Sigmoid}(x) = f_{Sigmoid}(x)(1 - f_{Sigmoid}(x))$$

https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf

Let us do some calculation

Consider the simple network below:



Assume that the neurons have a Sigmoid activation function and

(i) Perform a forward pass on the network.
(ii) Perform a reverse pass (training) once (target = 0.5).
(iii) Perform a further forward pass and comment on the result.

https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf

Let us do some calculation

Answer:
(i)
Input to top neuron = (0.35x0.1)+(0.9x0.8)=0.755. Out = 0.68.
Input to bottom neuron = (0.9x0.6)+(0.35x0.4) = 0.68. Out = 0.6637.
Input to final neuron = (0.3x0.68)+(0.9x0.6637) = 0.80133. Out = 0.69.

(ii)
Output error $\delta=(t-o)(1-o)o = (0.5-0.69)(1-0.69)0.69 = -0.0406$.

New weights for output layer
$w1^+ = w1+(\delta \times input) = 0.3 + (-0.0406x0.68) = 0.272392$.
$w2^+ = w2+(\delta \times input) = 0.9 + (-0.0406x0.6637) = 0.87305$.

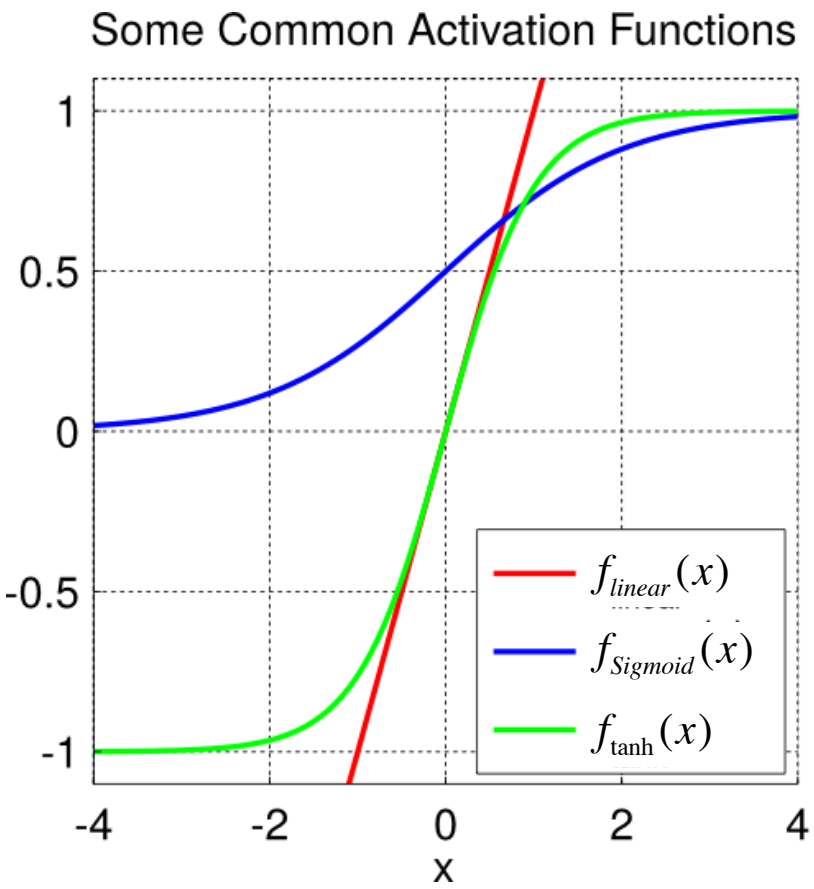
Errors for hidden layers:
$\delta1 = \delta \times w1 = -0.0406 \times 0.272392 \times (1-o)o = -2.406x10^{-3}$
$\delta2 = \delta \times w2 = -0.0406 \times 0.87305 \times (1-o)o = -7.916x10^{-3}$

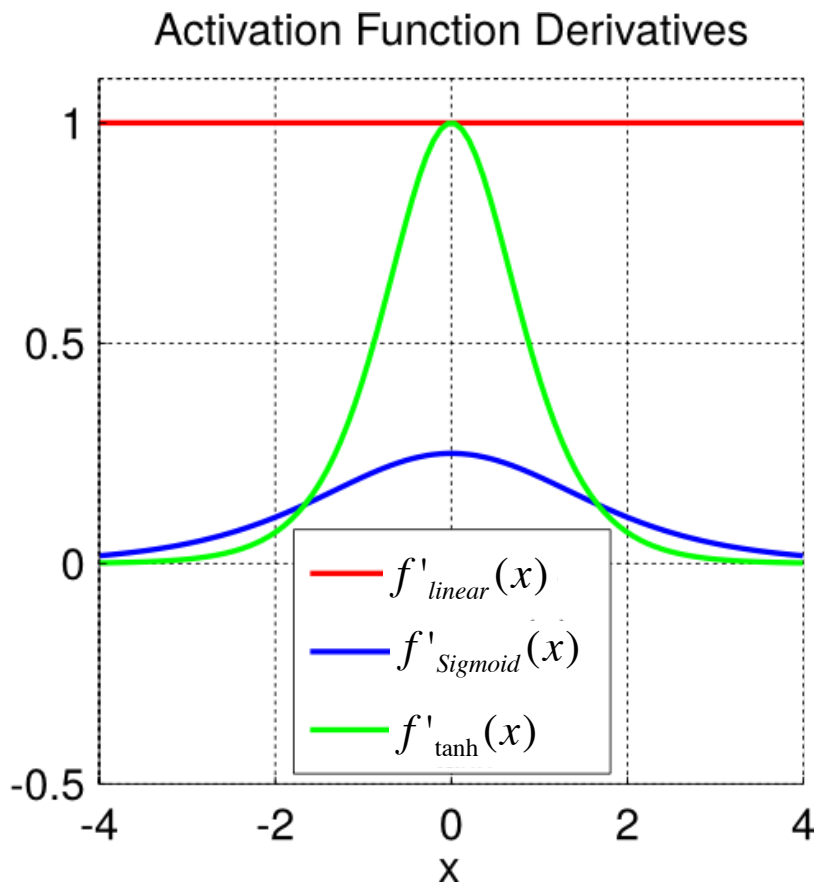
New hidden layer weights:
$w3^+=0.1 + (-2.406 \times 10^{-3} \times 0.35) = 0.09916$.
$w4^+ = 0.8 + (-2.406 \times 10^{-3} \times 0.9) = 0.7978$.
$w5^+ = 0.4 + (-7.916 \times 10^{-3} \times 0.35) = 0.3972$.
$w6^+ = 0.6 + (-7.916 \times 10^{-3} \times 0.9) = 0.5928$.

(iii)
Old error was -0.19. New error is -0.18205. Therefore error has reduced.

https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf

Active functions





https://theclevermachine.wordpress.com/tag/tanh-function/

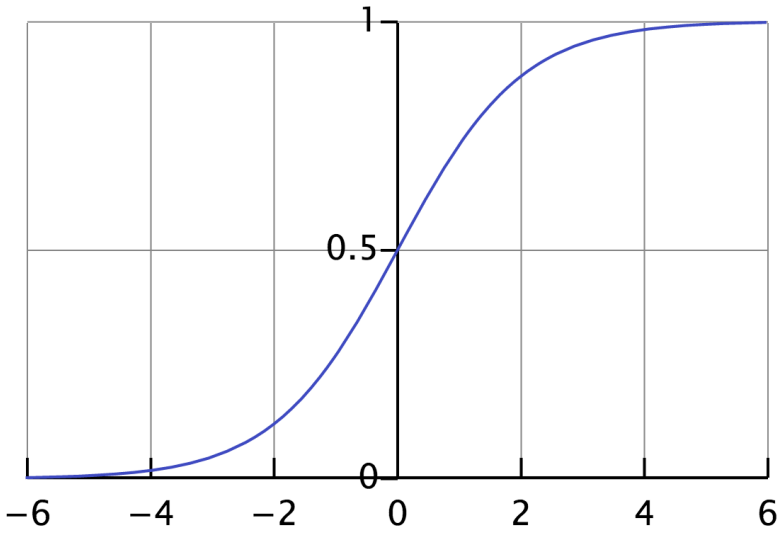
Activation functions

- Logistic Sigmoid:

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

Its derivative:

$$f'_{Sigmoid}(x) = f_{Sigmoid}(x)(1 - f_{Sigmoid}(x))$$

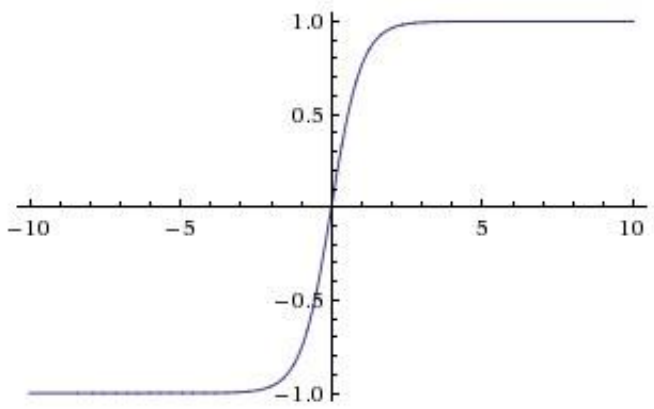


- Output range [0,1]
- Motivated by biological neurons and can be interpreted as the probability of an artificial neuron “firing” given its inputs
- However, saturated neurons make gradients vanished **(why?)**

Activation functions

- Tanh function

$$f_{\tanh}(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



Its gradient:

$$f_{\tanh}'(x) = 1 - f_{\tanh}'(x)$$

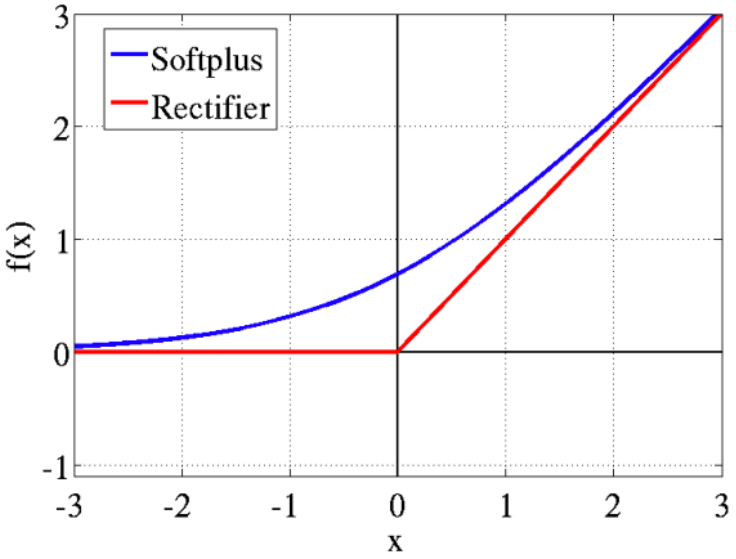
- Output range [-1,1]
- Thus strongly negative inputs to the tanh will map to negative outputs.
- Only zero-valued inputs are mapped to near-zero outputs
- These properties make the network less likely to get “stuck” during training

https://theclevermachine.wordpress.com/tag/tanh-function/

Active Functions

- ReLU (rectified linear unit)

$$f_{\mathrm{Re}LU}(x) = \max(0,x)$$



http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/40811.pdf

The derivative:

for x>0, f'=1, and x<0, g'=0.

- Another version is Noise ReLU:

$$f_{noisy\,\mathrm{Re}LU}(x) = Max(0, x + N(0, \delta(x)))$$

- ReLU can be approximated by softplus function

$$f_{softplus}(x) = \log(1 + e^{x})$$

- ReLU gradient doesn't vanish as we increase x
- It can be used to model positive number
- It is fast as no need for computing the exponential function
- It eliminates the necessity to have a "*pretraining*" phase

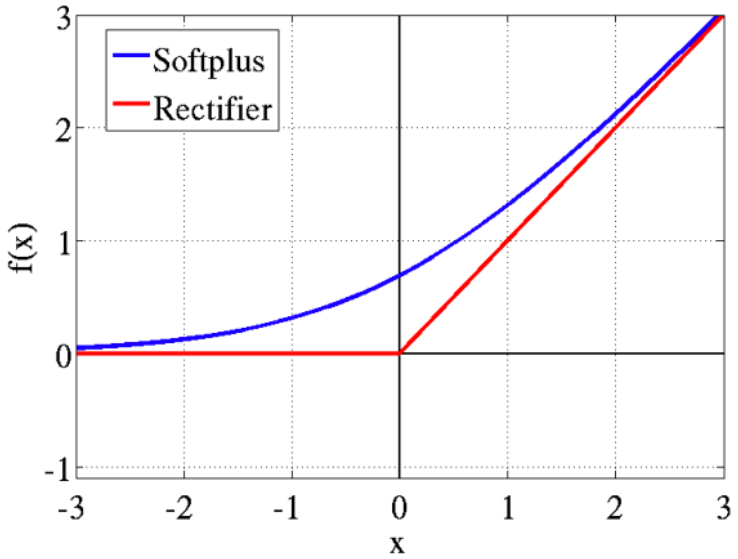
Active Functions

- ReLU (rectified linear unit)

$$f_{ReLU}(x) = \max(0, x)$$

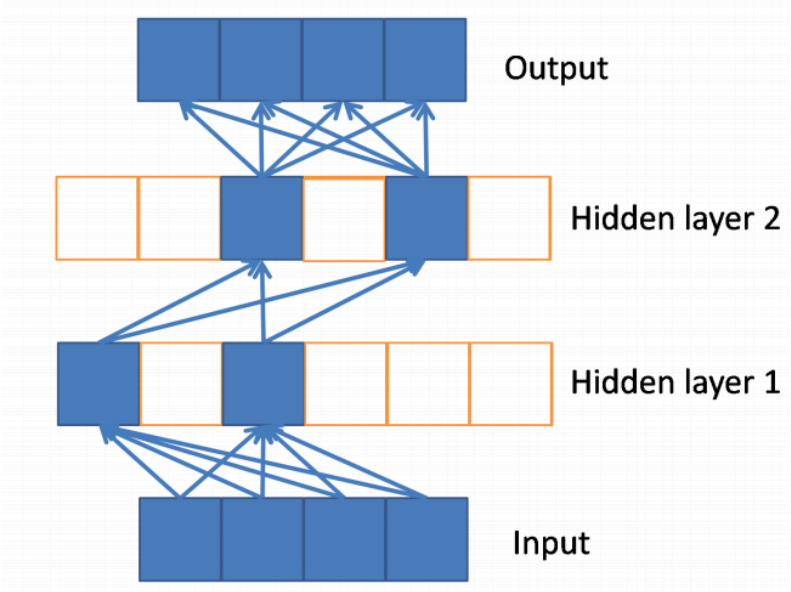
ReLU can be approximated by softplus function

$$f_{softplus}(x) = \log(1 + e^{x})$$



Additional active functions:
Leaky ReLU, Exponential LU, Maxout etc

- The only non-linearity comes from the path selection with individual neurons being active or not
- It allows sparse representations:
  - for a given input only a subset of neurons are active



Sparse propagation of activations and gradients

http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

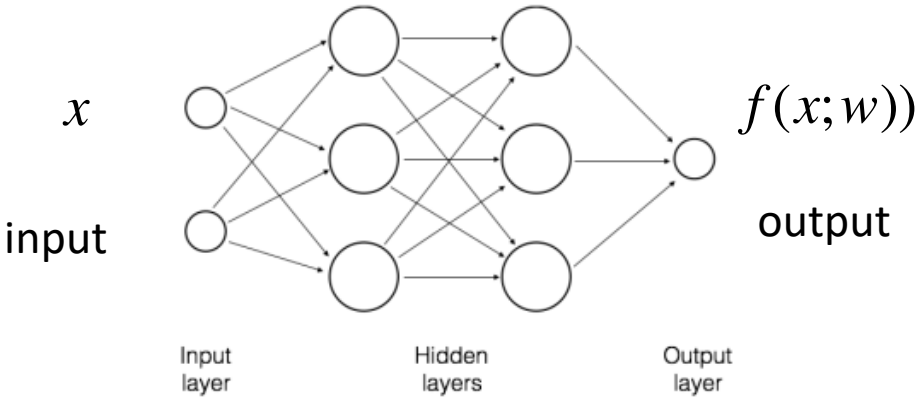
Error/Loss function

- Recall stochastic gradient descent
  - update from a randomly picked example (but in practice do a batch update)

$$w = w - \eta \frac{\partial E(w)}{\partial w}$$

- Squared error loss for one binary output:

$$E(w) = \frac{1}{2}(d_{label} - f(x;w))^2$$



Error/Loss function

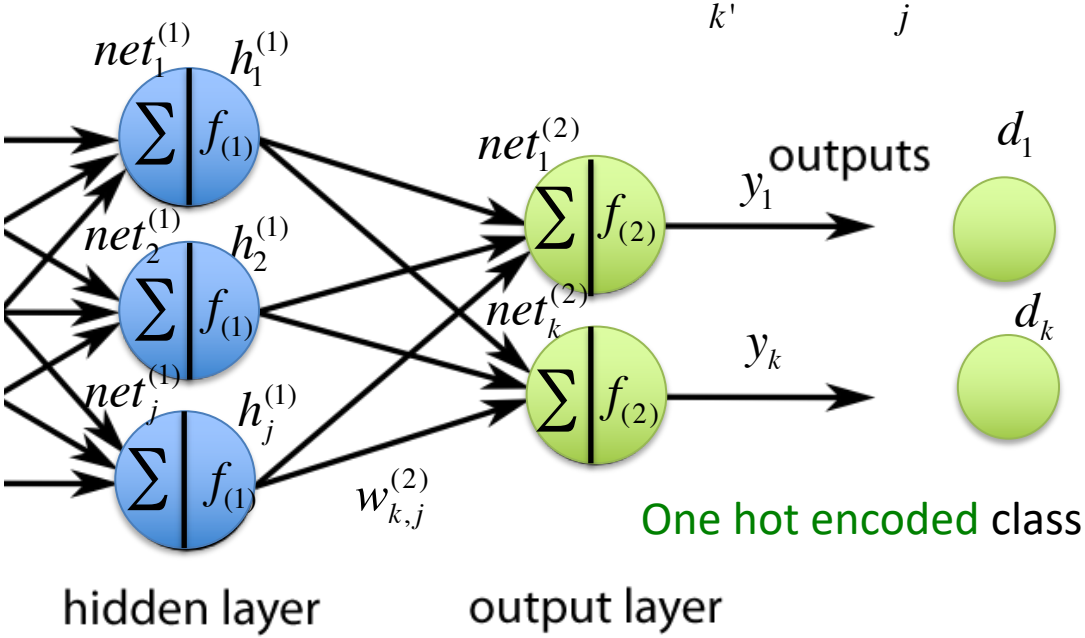
- Softmax (cross-entropy loss) for multiple classes:

(Class labels follow multinomial distribution)

$$E(w) = -\sum_k (d_k \log y_k + (1 - d_k) \log(1 - y_k))$$

where

$$y_k = \frac{\exp(\sum_j w_{k,j}^{(2)} h_j^{(1)})}{\sum_{k'} \exp(\sum_j w_{k',j}^{(2)} h_j^{(1)})}$$



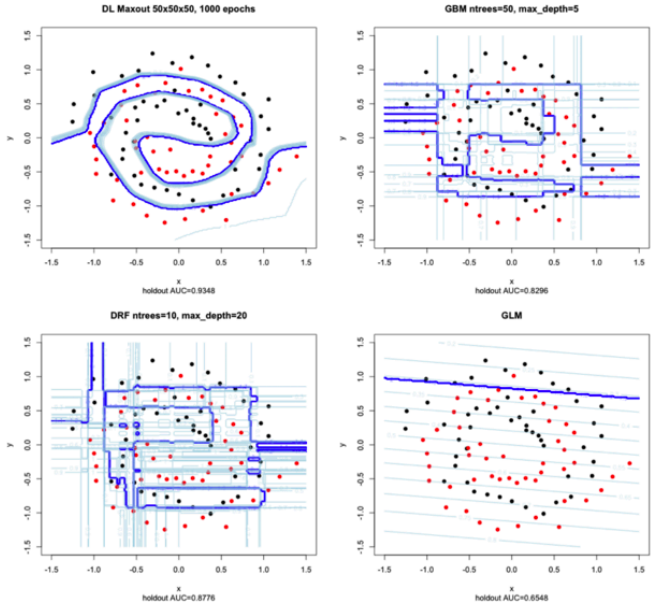
Backpropagation: conclusion

Advantages

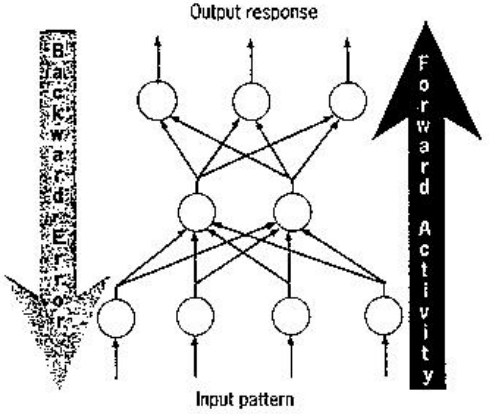
- Multi layer Perceptron can be trained by the back propagation algorithm to perform any mapping between the input and the output.

What is wrong with backpropagation?

- It requires labeled training data.
  Most data is unlabeled.
- The learning time does not scale well
  It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.



http://www.infoworld.com/article/3003315/big-data/deep-learning-a-brief-guide-for-practical-problem-solvers.html



A backpropagation network trains with a two-step procedure. The activity from the input pattern flows forward through the network, and the error signal flows backward to adjust the weights.

Hornik, Kurt. "Approximation capabilities of multilayer feedforward networks." Neural networks 4.2 (1991): 251-257.

Slide based on Geoffrey Hinton's

Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

Word Embedding

- Motivation: representing words with low-dimensional real-valued vectors, utilising them as input to deep learning methods, vs one-hot vectors
- Word2Vec
  - Input: *words and their contexts in documents*
  - Output: *embedding vectors of words*
- Assumption: similar words occur in similar contexts
- Advantage: compact representations (usually 100~ dimensions)

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. InAdvances in neural information processing systems 2013 (pp. 3111-3119).

Word embedding: CBOW

- From bag of word to word embedding
  - Use a a real- valued vector in R^m to represent a word (concept)

  v(‘‘cat")=(0.2, -0.4, 0.7, ...)

  v(‘‘mat")=(0.0, 0.6, -0.1, ...)
- Continuous bag of word (CBOW) model (word2vec)
  - Input/output words *x/y* are one-hot encoded
  - Hidden layer is shared for all input words

***N*-dim Vector representation of a word**

Hidden nodes:

$$\mathbf{h} = \frac{1}{C}\mathbf{W}\cdot(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_C) = \frac{1}{C}\cdot(\mathbf{v}_{w_1}+\mathbf{v}_{w_2}+\cdots+\mathbf{v}_{w_C})$$

The cross-entropy loss:

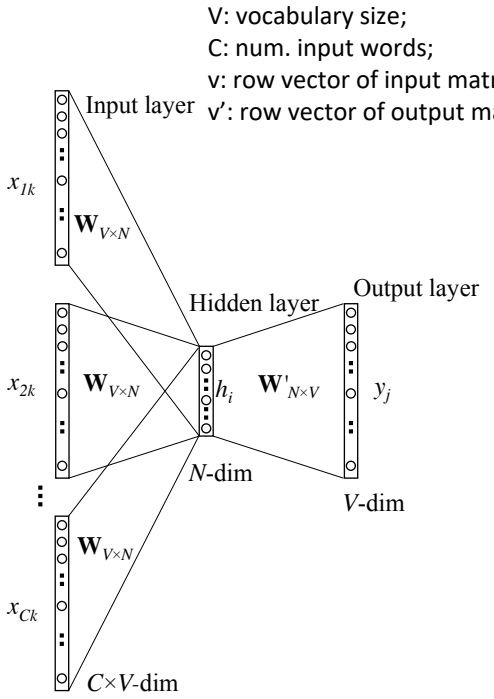
$$E = -\log p(w_O|w_{I,1},\cdots,w_{I,C}) = -\mathbf{v}'_{w_O}{}^T\cdot\mathbf{h}+\log\sum_{j'=1}^{V}\exp(\mathbf{v}'_{w_j}{}^T\cdot\mathbf{h})$$

The gradient updates:

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta\cdot e_j\cdot\mathbf{h} \quad \text{for } j=1,2,\cdots,V.$$

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C}\cdot\eta\cdot\text{EH} \quad \text{for } c=1,2,\cdots,C.$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V}\frac{\partial E}{\partial u_j}\cdot\frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j\cdot w'_{ij} := \text{EH}_i$$



Continuous bag of word (CBOW) model

Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).

Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

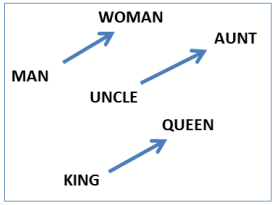
Remarkable properties from Word embedding

- Simple algebraic operations with the word vectors

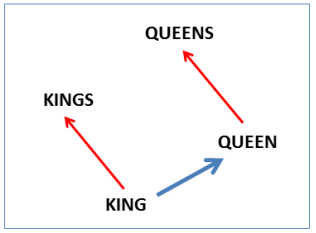
Using X = v(”biggest”) − v(”big”) + v(”small”) as query and searching for the nearest word based on cosine distance results in v(”smallest”)

v(‘‘woman")–v(‘‘man") ≃ v(‘‘aunt")–v(‘‘uncle")

v(‘‘woman")–v(‘‘man") ≃ v(‘‘queen")–v(‘‘king")



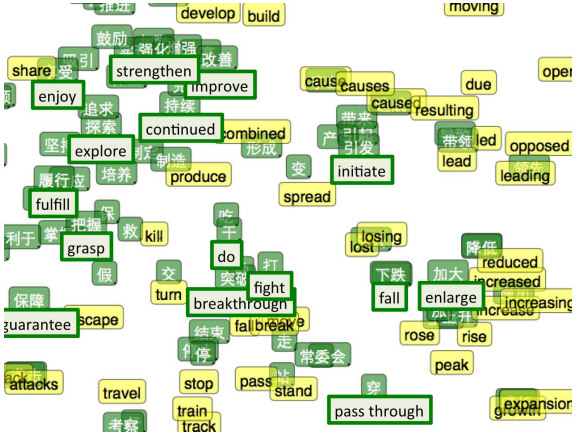
Vector offsets for gender relation



The singular/plural relation for two words

Word the relationship is defined by subtracting two word vectors, and the result is added to another word. Thus for example, **Paris - France + Italy = Rome**.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

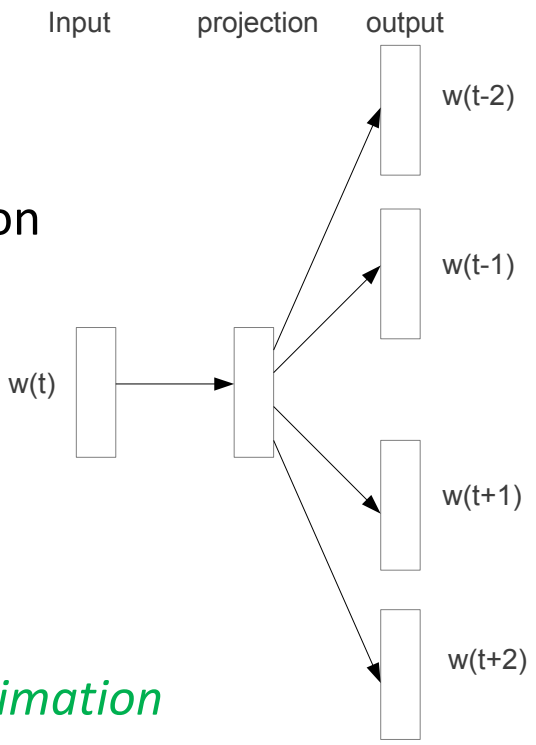


Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations." HLT-NAACL. 2013.
Zou, Will Y., et al. "Bilingual Word Embeddings for Phrase-Based Machine Translation." EMNLP. 2013.

Word Embedding: Skip-Gram with Negative Sampling

- The training objective is to learn word vector representations of w_t that are good at predicting the nearby words w_{t+j}

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$



- Skip-gram defines $p(w_{t+j}|w_t)$ using the *softmax* function ($w_I = w_t$; $w_O = w_{t+j}$)

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$

- A potential problem is the sum in the denominator is computationally expensive
- Solution: replace $p(w_O|w_I)$ with *Noise Contrastive Estimation* (NCE)

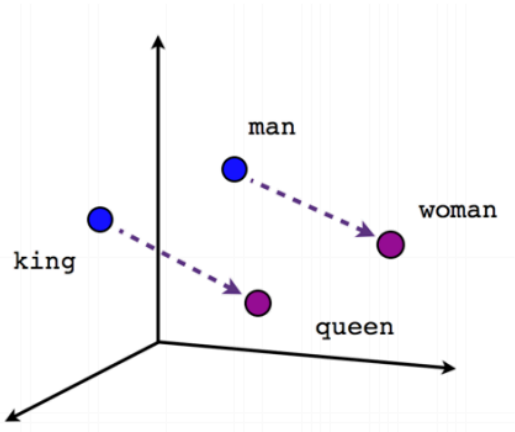
$$\log\sigma({v'_{w_O}}^{\top} v_{w_I}) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma(-{v'_{w_i}}^{\top} v_{w_I})\right]$$

a good model should be able to differentiate data from noise $P_n(w)$

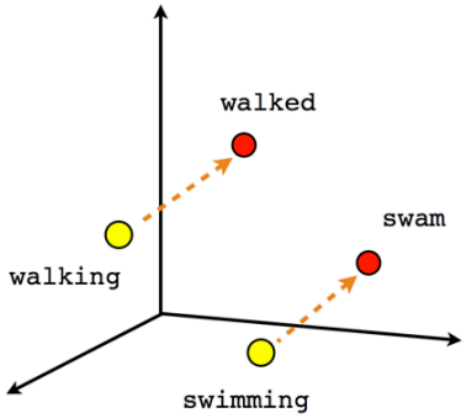
Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. InAdvances in neural information processing systems 2013 (pp. 3111-3119).

Word Embedding

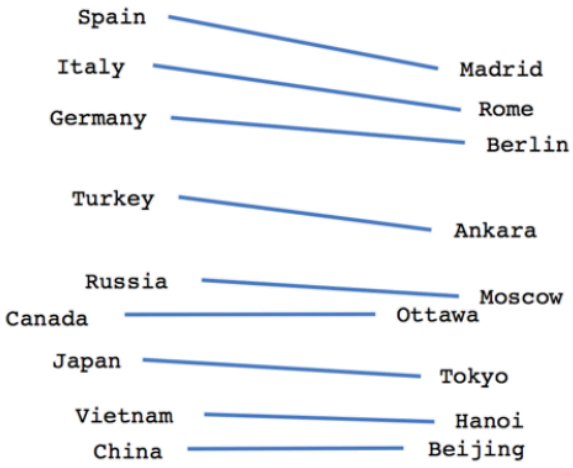
- We can visualize the learned vectors by projecting them down to 2 dimensions



Male-Female



Verb tense



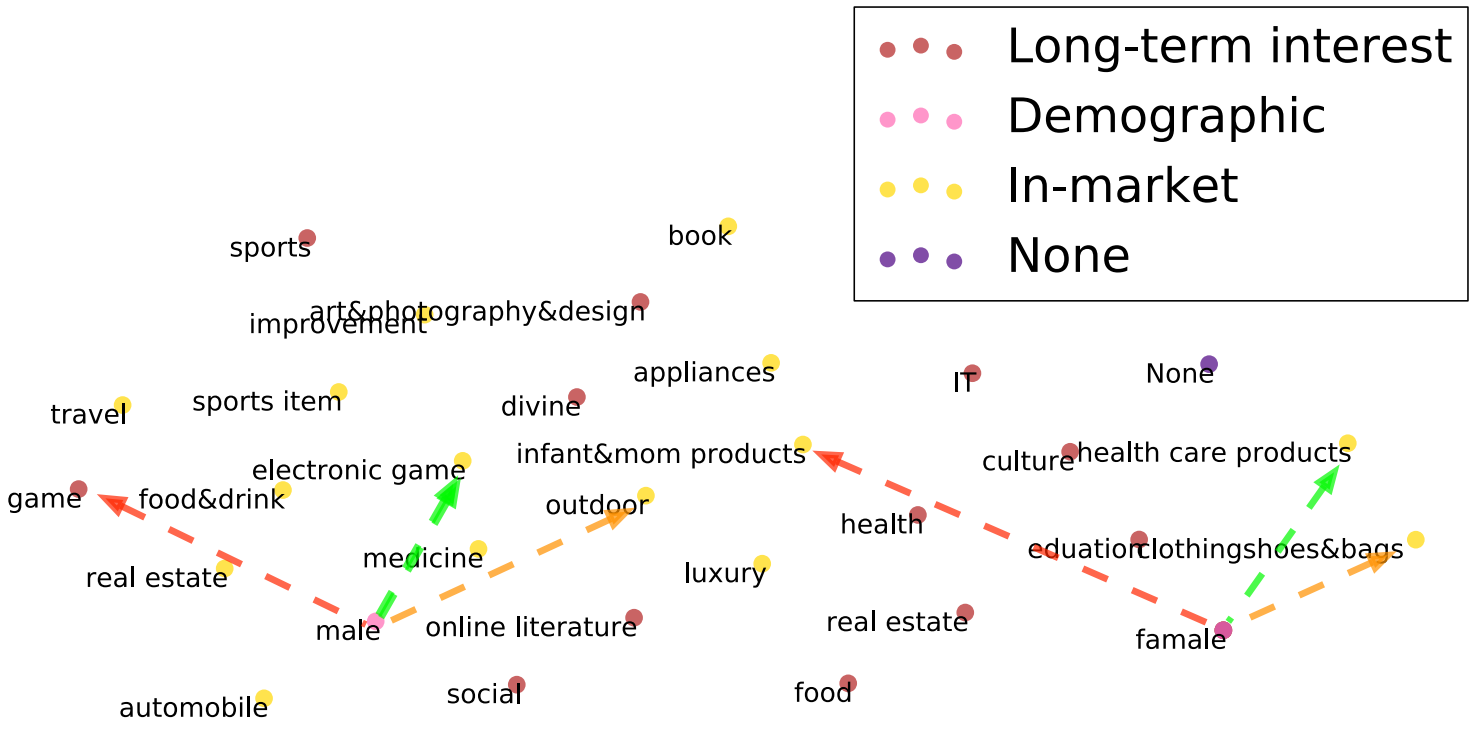
Country-Capital

- They capture some general *semantic information* about words and their relationships to one another

https://www.tensorflow.org/tutorials/word2vec

Word Embedding

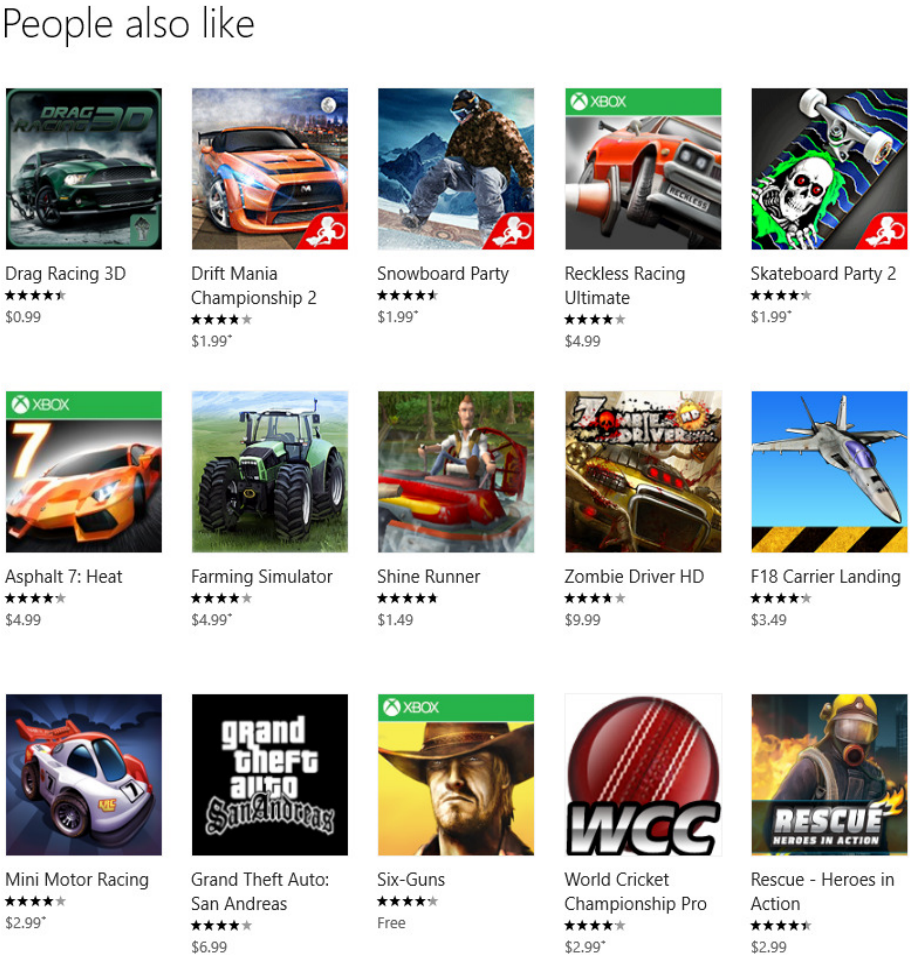
- Test on user interesting tag data in advertising



Ying Wen et al. Cat2Vec: Distributed Representation of Multi-field Categorical Data, under submission, 2016

Item2Vec for recommendation

- Recommendations in Windows 10 Store based on similar items to *Need For Speed*
- Item-based Collaborative Filtering can be cast in the same framework of neural word embedding
- Solution: We can create a joint item pair as (context-item, item) if they *co-occur* in a user profile



Barkan, Oren, and Noam Koenigstein. "Item2vec: neural item embedding for collaborative filtering." *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016.

Recommendation results

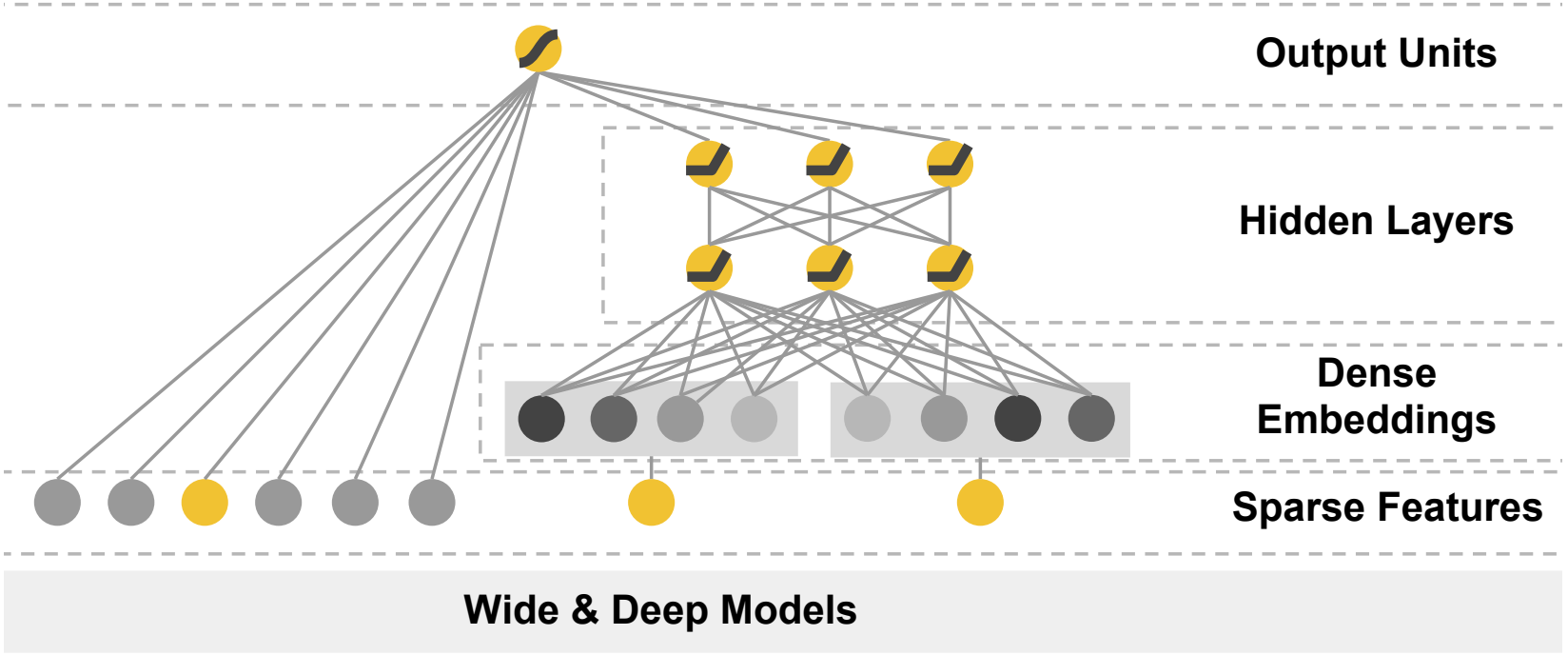
- ITEM2VEC AND SVD FOR SELECTED ITEMS FROM THE STORE DATASET

Seed item	item2vec – Top 4 recommendations	SVD – Top 4 recommendations
LEGO Emmet	LEGO Bad Cop, LEGO Simpsons: Bart, LEGO Ninjago, LEGO Scooby-Doo	Minecraft Foam, Disney Toy Box, Minecraft (Xbox One), Terraria (Xbox One)
Minecraft Lanyard	Minecraft Diamond Earrings, Minecraft Periodic Table, Minecraft Crafting Table, Minecraft Enderman Plush	Rabbids Invasion, Mortal Kombat, Minecraft Periodic Table
GoPro LCD Touch BacPac	GoPro Anti-Fog Inserts, GoPro The Frame Mount, GoPro Floaty Backdoor, GoPro 3-Way	Titanfall (Xbox One), GoPro The Frame Mount, Call of Duty (PC), Evolve (PC)
Surface Pro 4 Type Cover	UAG Surface Pro 4 Case, Zip Sleeve for Surface, Surface 65W Power Supply, Surface Pro 4 Screen Protection	Farming Simulator (PC), Dell 17 Gaming laptop, Bose Wireless Headphones, UAG Surface Pro 4 Case
Disney Baymax	Disney Maleficent, Disney Hiro, Disney Stich, Disney Marvel Super Heroes	Disney Stich, Mega Bloks Halo UNSC Firebase, LEGO Simpsons: Bart, Mega Bloks Halo UNSC Gungoose
Windows Server 2012 R2	Windows Server Remote Desktop Services 1-User, Exchange Server 5-Client, Windows Server 5-User Client Access, Exchange Server 5-User Client Access	NBA Live (Xbox One) – 600 points Download Code, Windows 10 Home, Mega Bloks Halo Covenant Drone Outbreak, Mega Bloks Halo UNSC Vulture Gunship

Barkan, Oren, and Noam Koenigstein. "Item2vec: neural item embedding for collaborative filtering." *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016.

Wide & Deep Learning for Recommender Systems

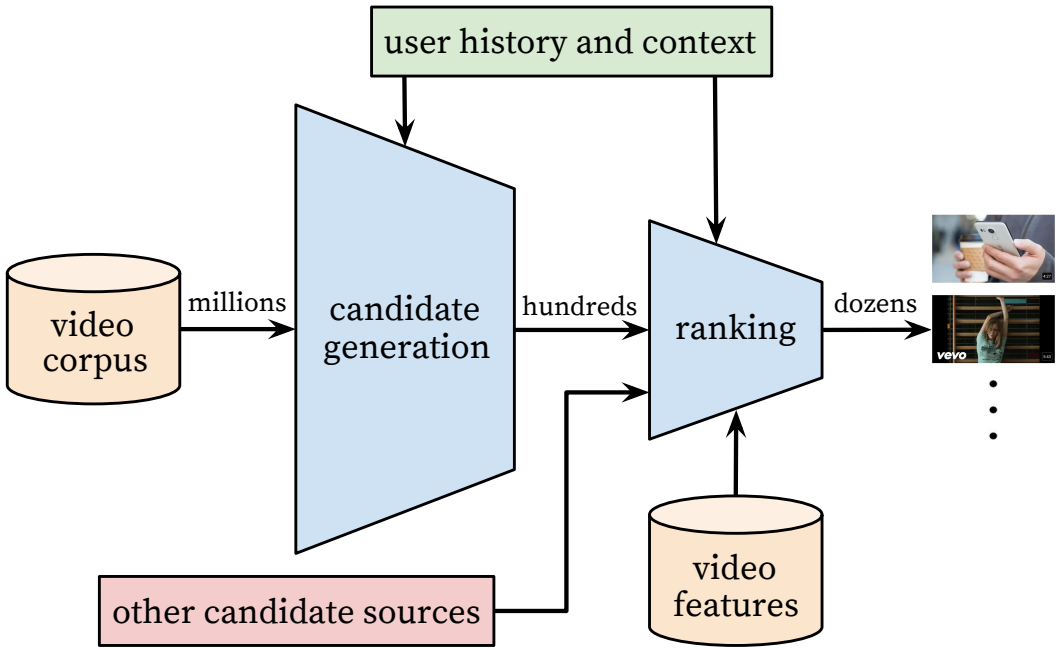
- Jointly trained wide linear models and deep neural networks

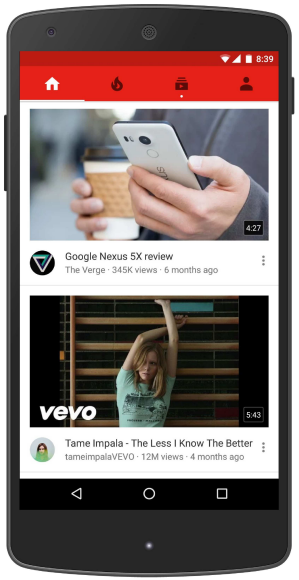


Cheng, Heng-Tze, et al. "Wide & deep learning for recommender systems." *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016.

Video Recommendation

- Two-step architecture in Youtube

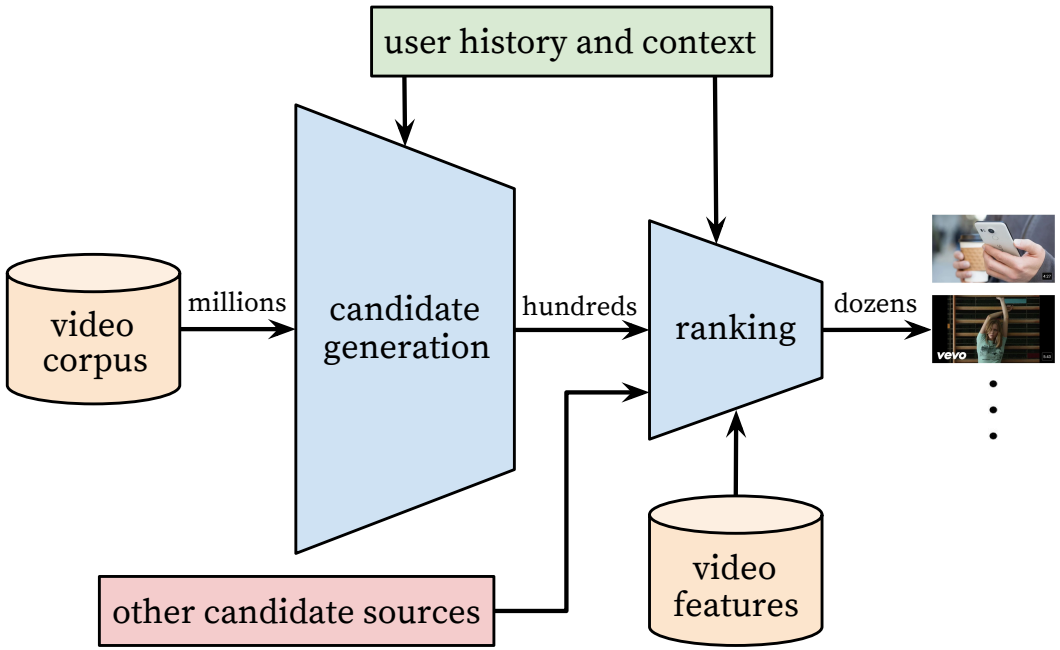


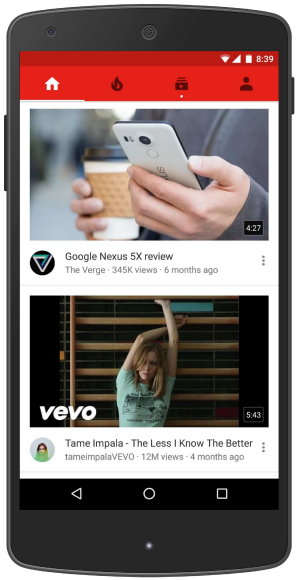


Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016.

Video Recommendation

- Two-step architecture in Youtube

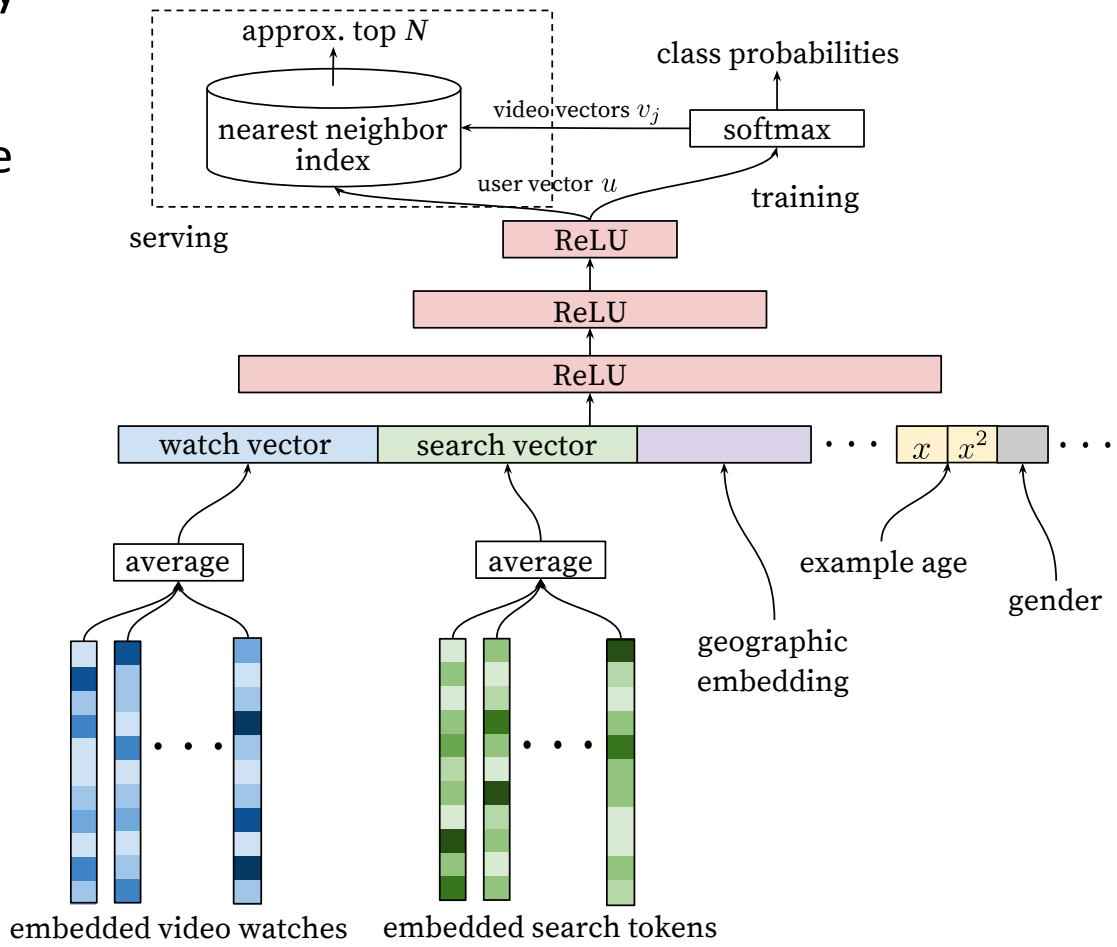




Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016.

Candidate video generation

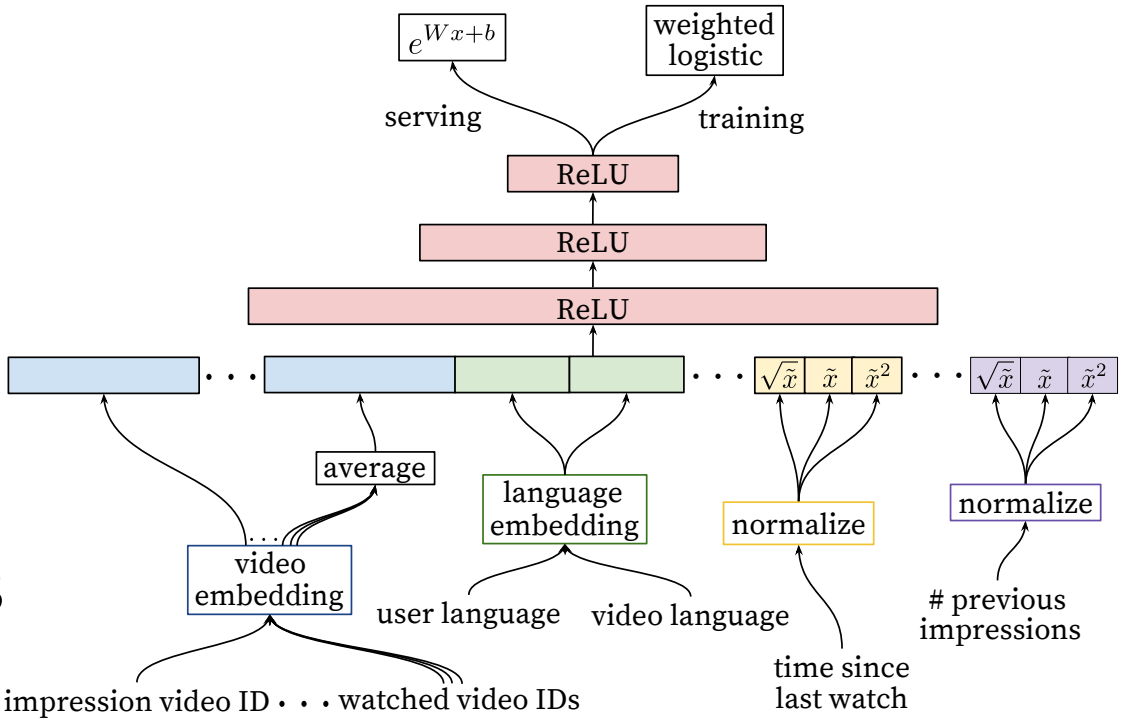
- User profile is represented by a bag of sparse video IDs
- Embeddings of those watched/searched videos are averaged before concatenation
- All hidden layers are fully connected.
- In training, a cross-entropy loss is minimized with gradient descent on the output of the sampled softmax.
- At serving, an approximate nearest neighbor lookup is performed to generate hundreds of candidate video recommendations.



Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016.

Video ranking

- Hundreds of features are fed into the network.
- Embedded categorical features
- Powers of normalized continuous features
- All layers are fully connected.

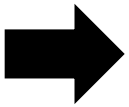


Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016.

User response estimation problem

- Click-through rate estimation as an example

- Date: 20160320
- Hour: 14
- Weekday: 7
- IP: 119.163.222.*
- Region: England
- City: London
- Country: UK
- Ad Exchange: Google
- Domain: yahoo.co.uk
- URL: http://www.yahoo.co.uk/abc/xyz.html
- OS: Windows
- Browser: Chrome
- Ad size: 300*250
- Ad ID: a1890
- User tags: Sports, Electronics



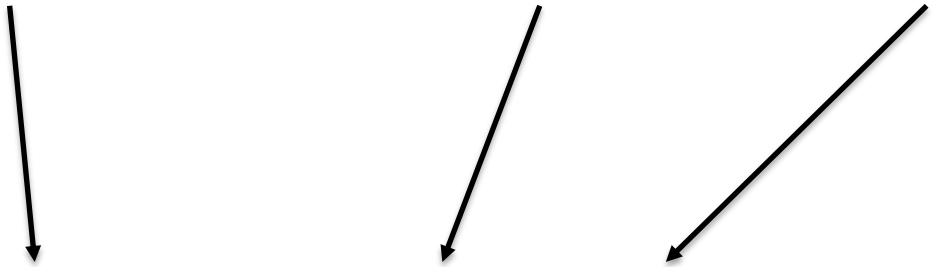
Click (1) or not (0)?

Predicted CTR (0.15)

Feature Representation

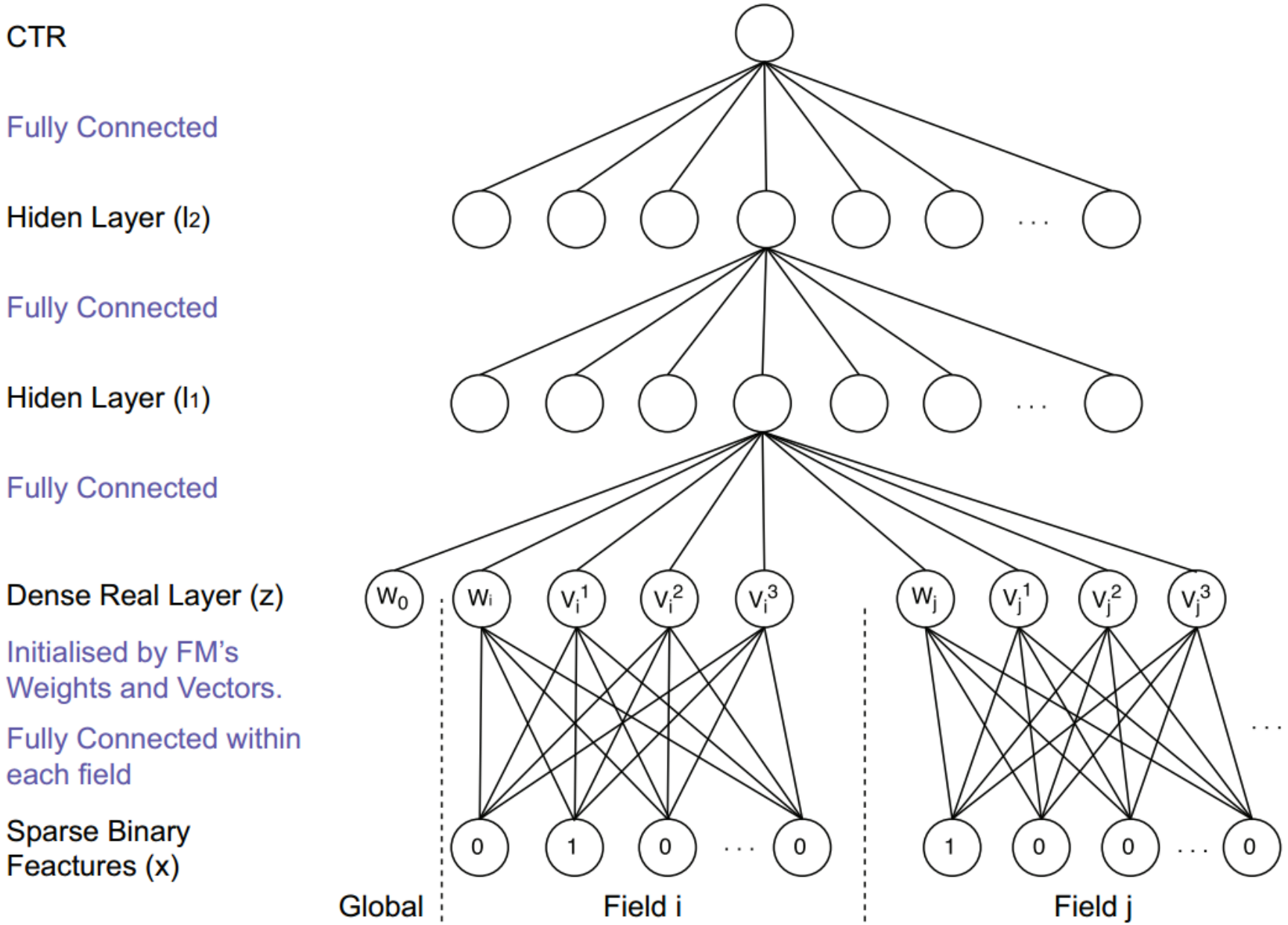
- Binary one-hot encoding of categorical data

x=[Weekday=Wednesday, Gender=Male, City=London]



x=[0,0,1,0,0,0,0 0,1 0,0,1,0…0]

High dimensional sparse binary feature vector



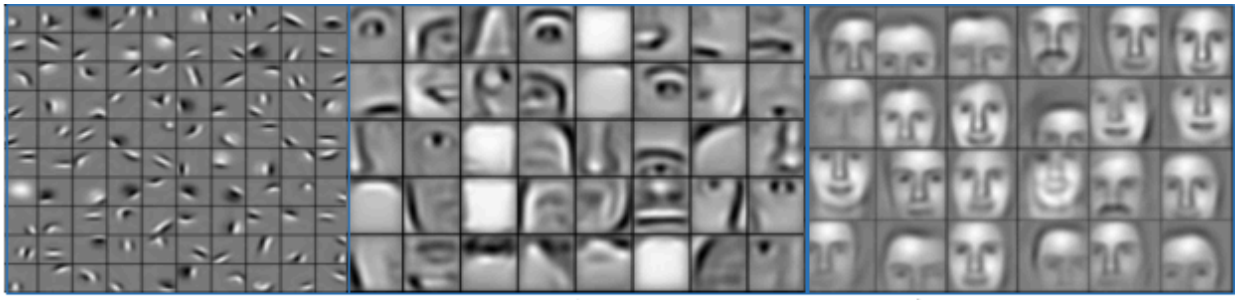
[Zhang et al. Deep Learning over Multi-field Categorical Data – A Case Study on User Response Prediction. ECIR 16]

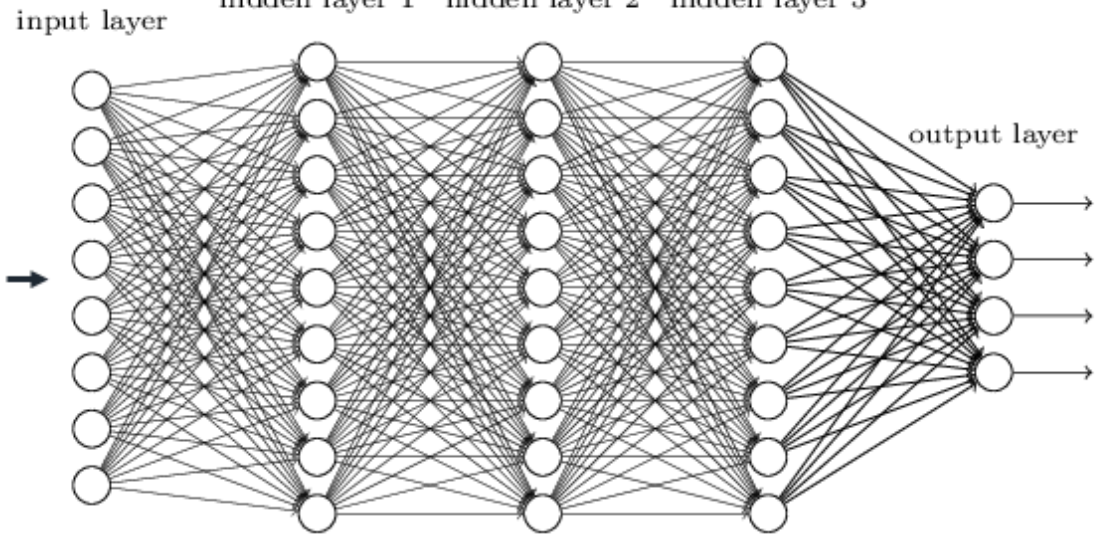
Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

What more than one hidden layer?

Deep neural networks learn hierarchical feature representations





http://www.rsipvision.com/exploring-deep-learning/

Why goes deep now?

- Big data everywhere
- Demands for:
  - Unsupervised learning: most of them are unlabeled
  - End-to-end: trainable feature solution
    - time consuming for hand-crafted features
- Improved computational power:
  - Widely used GPUs for computing
  - Distributed computing (mapreduce & spark)

Deep learning problem

- Difficult to train
  - Many many Parameters
- Two solutions:
  - Regularised by prior knowledge (since 1989)
    - Convolutional Neural Networks (CNN)
  - New method of pre-training using unsupervised methods (since 2006)
    - Stacked auto-encoders
    - Stacked Restricted Boltzmann Machines

New methods for unsupervised pre-training

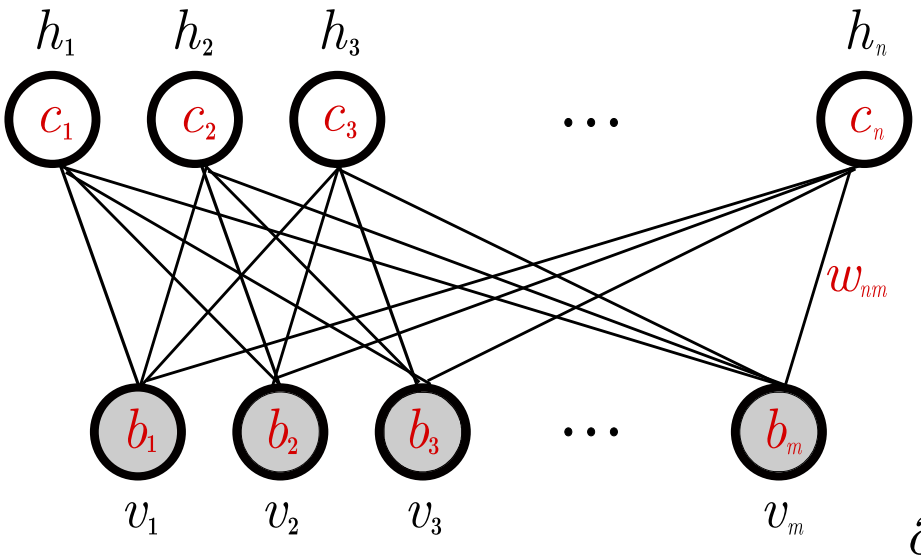
- Stacked Restricted Boltzmann Machines
  - Hinton, G. E, Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527-1554.
  - Hinton, G. E. and Salakhutdinov, R. R, Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.
- Stacked Autoencoders
  - Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks, Advances in Neural Information Processing Systems 19

Restricted Boltzmann Machines (RBM)

- A RBM is a bipartite (two disjointed set) undirected graph and used as probabilistic generative models
  - contains *m* visible units $V = (V_1, \ldots, V_m)$ to represent observable data and
  - n hidden units $H = (H_1, \ldots, H_n)$ to capture their dependencies
- Assume the random variables (V , H) take values (v, h) ∈ $\{0, 1\}^{m+n}$

Joint distribution: $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$, where the energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$



No connections within a layer (restricted) leads to

$$p(H_i = 1 \mid \boldsymbol{v}) = \text{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$ sig(): sigmoid

$$p(V_j = 1 \mid \boldsymbol{h}) = \text{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

We learn the parameters w_{ij} *by maximising a log-likelihood given example* $\boldsymbol{v}$*:*

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

<*>$_{data}$: expectations under the data empirical distribution

<*>$_{model}$: expectations under the model

http://image.diku.dk/igel/paper/AItRBM-proof.pdf

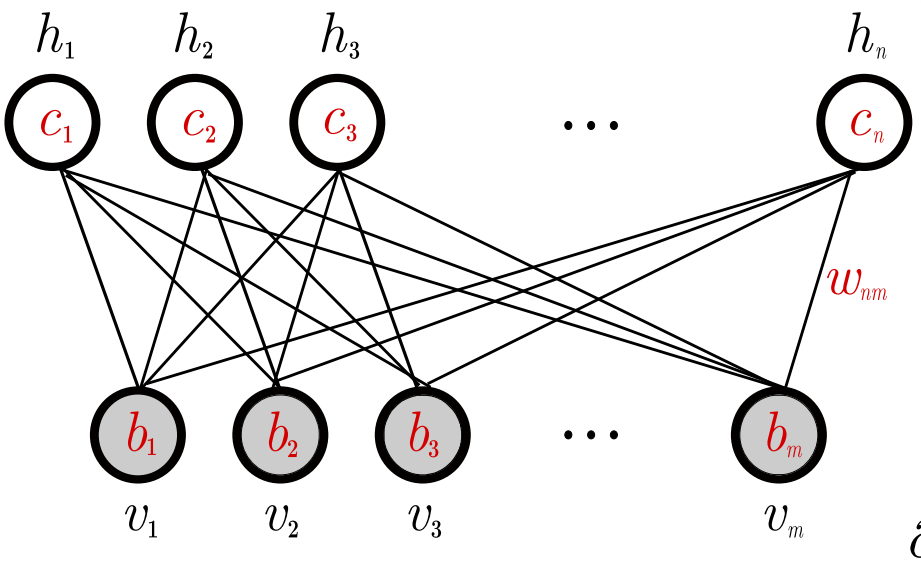
Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

Restricted Boltzmann Machines (RBM)

- A RBM is a bipartite (two disjointed set) undirected graph and used as probabilistic generative models
  - contains *m* visible units $V = (V_1, \ldots, V_m)$ to represent observable data and
  - n hidden units $H = (H_1, \ldots, H_n)$ to capture their dependencies
- Assume the random variables (V , H) take values (v, h) ∈ $\{0, 1\}^{m+n}$

Joint distribution: $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$, where the energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$



No connections within a layer (restricted) leads to

$$p(H_i = 1 \mid \boldsymbol{v}) = \text{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$ sig(): sigmoid

$$p(V_j = 1 \mid \boldsymbol{h}) = \text{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

We learn the parameters w_{ij} *by maximising a log-likelihood given example* $\boldsymbol{v}$*:*

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

<*>$_{data}$: expectations under the data empirical distribution

<*>$_{model}$: expectations under the model

http://image.diku.dk/igel/paper/AItRBM-proof.pdf

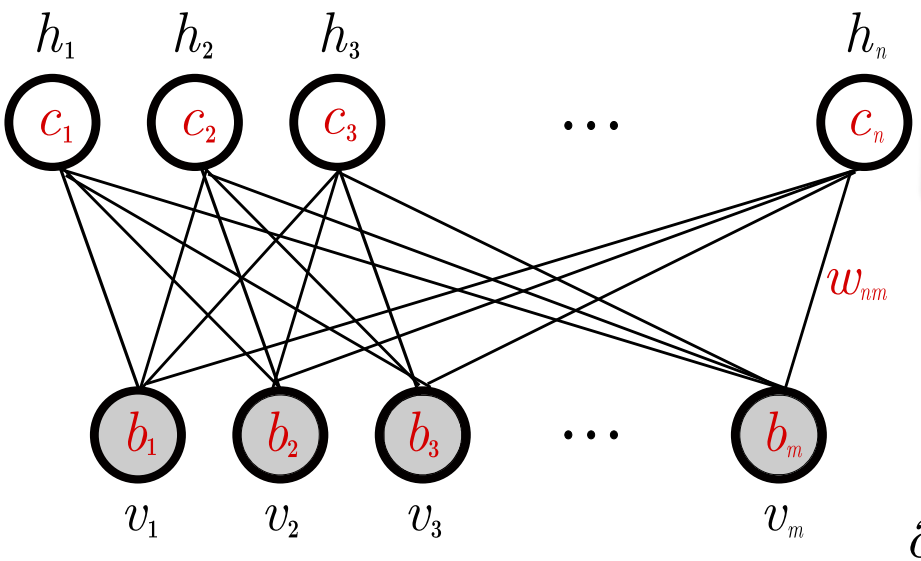
Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

Restricted Boltzmann Machines (RBM)

- A RBM is a bipartite (two disjointed set) undirected graph and used as probabilistic generative models
  - contains *m* visible units $V = (V_1, \ldots, V_m)$ to represent observable data and
  - n hidden units $H = (H_1, \ldots, H_n)$ to capture their dependencies
- Assume the random variables (V , H) take values (v, h) ∈ $\{0, 1\}^{m+n}$

Joint distribution: $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$, where the energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$



No connections within a layer (restricted) leads to

$$p(H_i = 1 \mid \boldsymbol{v}) = \text{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$ sig(): sigmoid

$$p(V_j = 1 \mid \boldsymbol{h}) = \text{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

We learn the parameters w_{ij} *by maximising a log-likelihood given example* $\mathbf{v}$*:*

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

<*>$_{data}$: expectations under the data empirical distribution

<*>$_{model}$: expectations under the model

http://image.diku.dk/igel/paper/AItRBM-proof.pdf

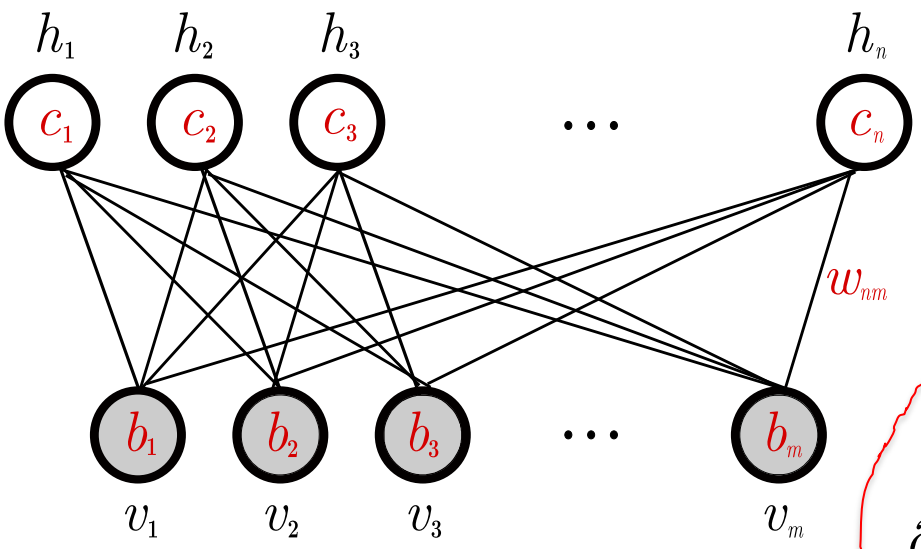
Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

Restricted Boltzmann Machines (RBM)

- A RBM is a bipartite (two disjointed set) undirected graph and used as probabilistic generative models
  - contains *m* visible units $V = (V_1, \ldots, V_m)$ to represent observable data and
  - n hidden units $H = (H_1, \ldots, H_n)$ to capture their dependencies
- Assume the random variables (V , H) take values (v, h) ∈ $\{0, 1\}^{m+n}$

Joint distribution: $p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$, where the energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$



No connections within a layer (restricted) leads to

$$p(H_i = 1 \mid \boldsymbol{v}) = \text{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$ sig(): sigmoid

$$p(V_j = 1 \mid \boldsymbol{h}) = \text{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

We learn the parameters w_{ij} *by maximising a log-likelihood given example* **v**:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

<*>$_{data}$: expectations under the data empirical distribution

<*>$_{model}$: expectations under the model

Can be approximated by Contrastive Divergence

http://image.diku.dk/igel/paper/AItRBM-proof.pdf

Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

RBM for recommender systems

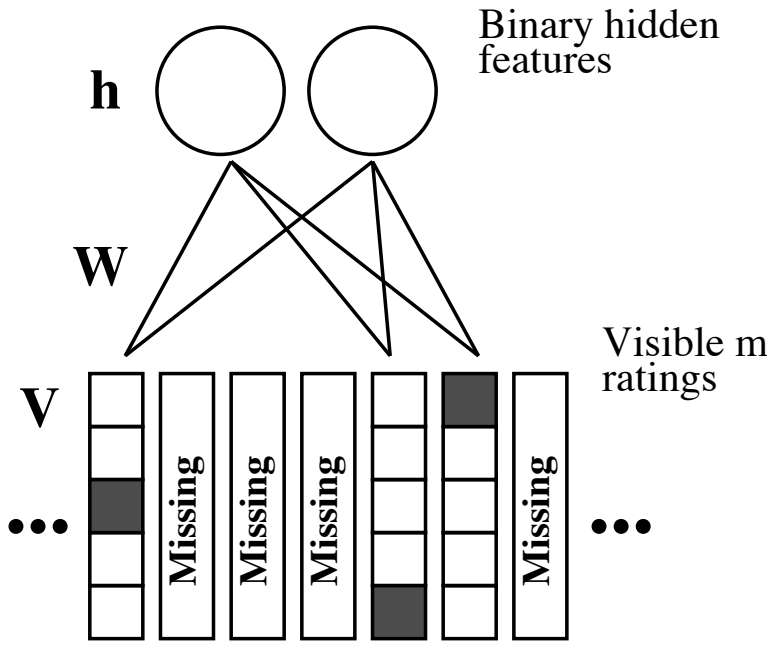
- From hidden layer **h** to the visible units

$$p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{l=1}^{K} \exp\left(b_i^l + \sum_{j=1}^{F} h_j W_{ij}^l\right)}$$

- From the visible units to the hidden layer

$$p(h_j = 1|\mathbf{V}) = \sigma\left(b_j + \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k W_{ij}^k\right)$$

- Learning is done by Contrastive Divergence
- Prediction is taken by the expected value among K=5 rating scale

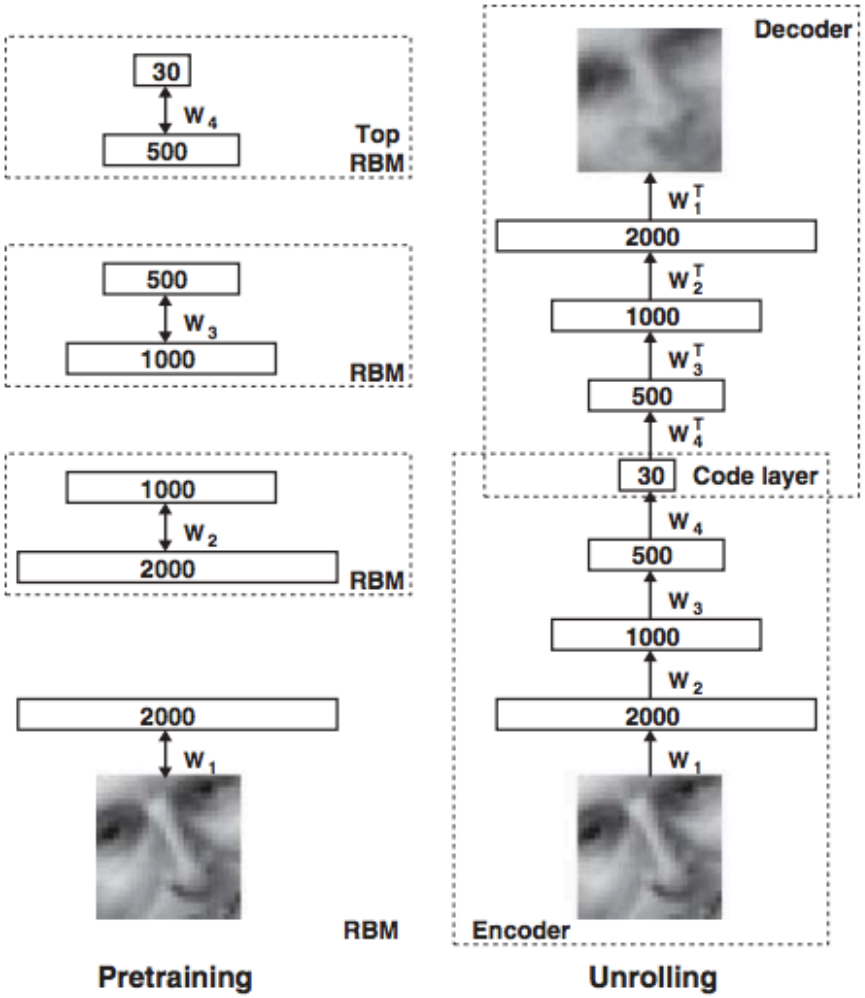


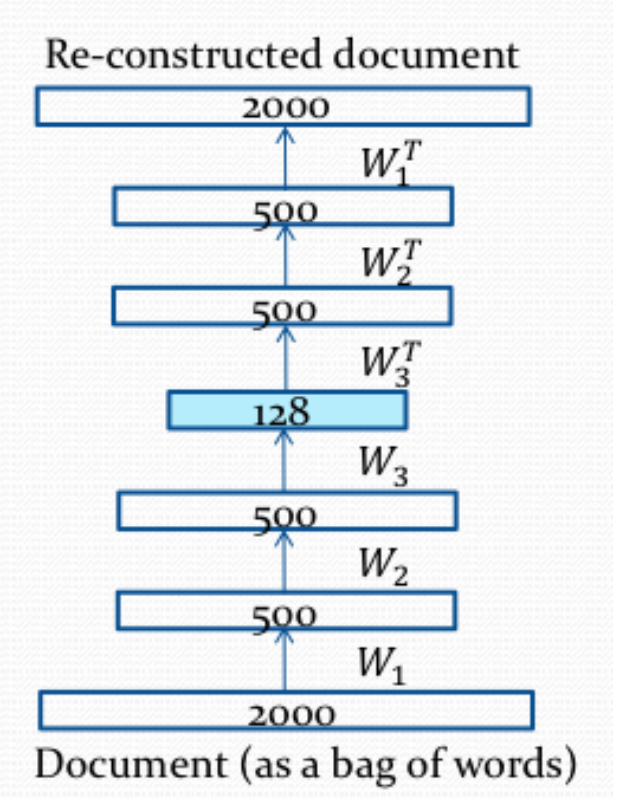
Notation: W^k_{ij} is a symmetric interaction parameter between feature j and rating k of movie i, b^k_i is the bias of rating k for visible movie movie i, and b_j is the bias of feature j.

For each user, the RBM only includes softmax units for the movies that user has rated.

Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." *Proceedings of the 24th international conference on Machine learning*. ACM, 2007.

Stacked RBM





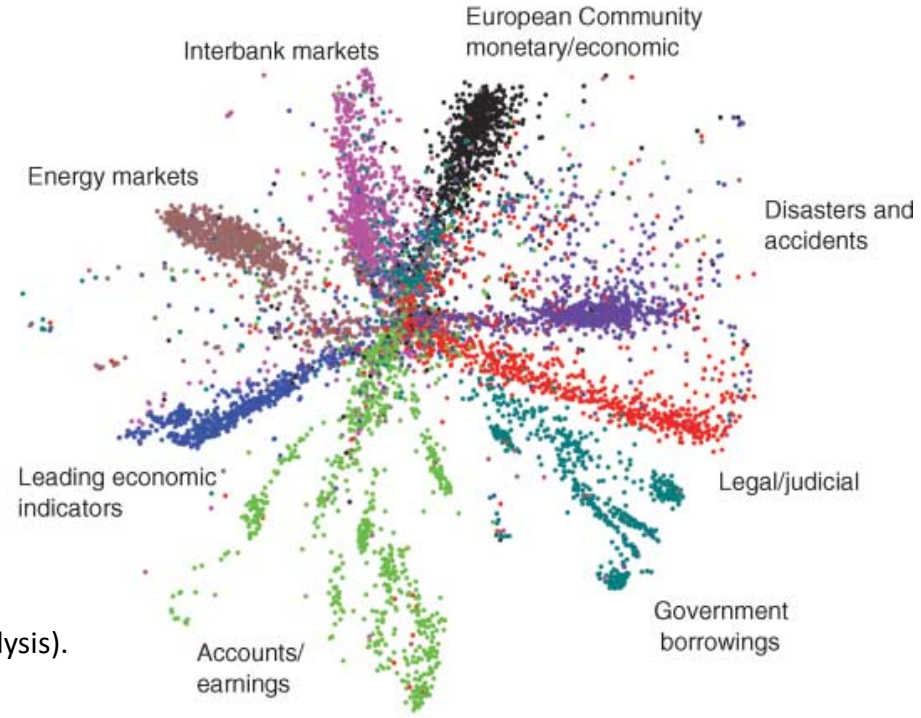
- The learned feature activations of one RBM are used as the ‘‘data’’ for training the next RBM in the stack.
- After the pretraining, the RBMs are ‘‘unrolled’’ to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

Stacked RBM

- Unsupervised learning results

The codes produced by two-dimensional LSA (Latent Semantic Analysis).



The codes produced by a 2000- 500-250-125-2 Stacked RBM

- The learned feature activations of one RBM are used as the ‘‘data’’ for training the next RBM in the stack.
- After the pretraining, the RBMs are ‘‘unrolled’’ to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

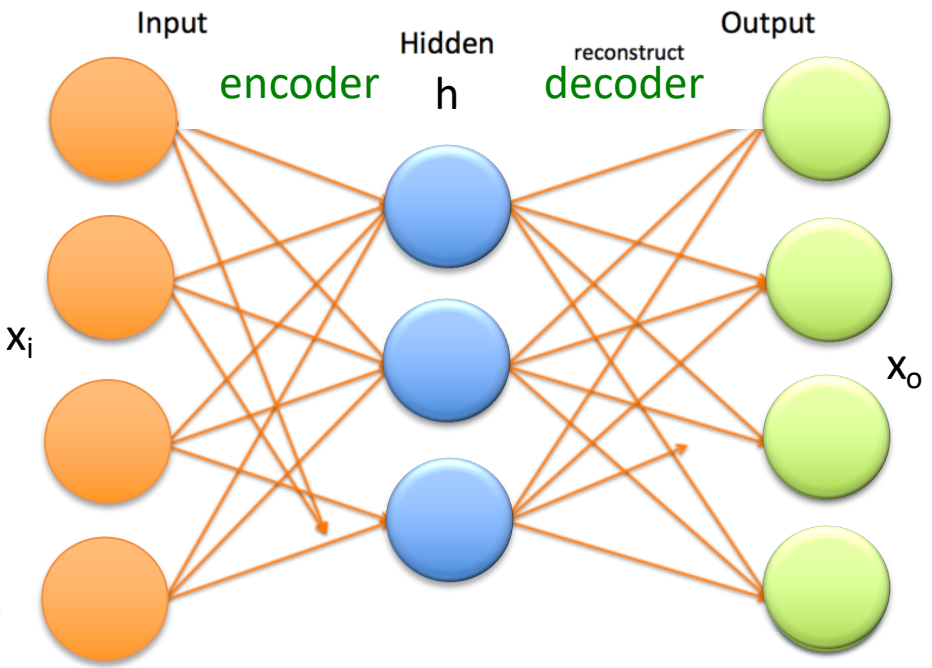
Auto-encoder

- Multi-layer neural network with target output and input:
  - Reconstruction x_o = decoder(encoder(input x_i))
  - Minimizing reconstruction error $(x_i - x_o)^2$

$$h = f(W_e^T x_i + b_e)$$

$$x_o = f(W_d^T h + b_d)$$

Setting : Tied weights $W_e^T = W_d$, f : sigmoid



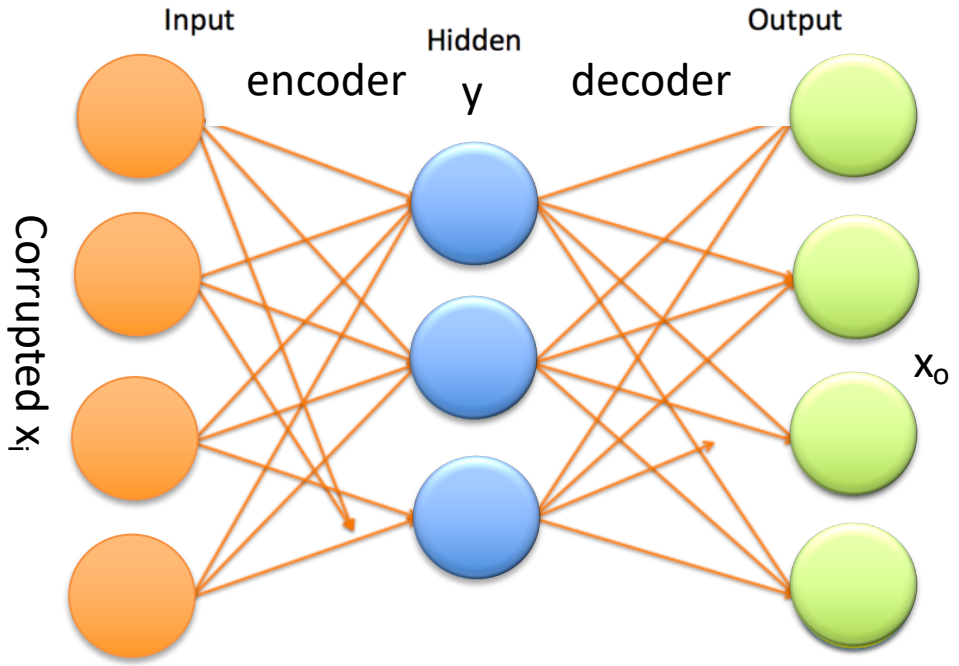
- The goal is to learn $\boldsymbol{W_e}$, b_e, b_d such that the average reconstruction error is minimised
- If f is chosen to be linear, auto-encoder is equivalent to PCA; if non-linear, it captures multi-modal aspects of the input distribution

http://deeplearning.net/tutorial/dA.html

Denoising auto-encoder

- A stochastic version of the auto-encoder.
- We train the autoencoder to reconstruct the input from a *corrupted version* of it , so that
  - force the hidden layer to discover more robust features and
  - prevent it from simply learning the identity

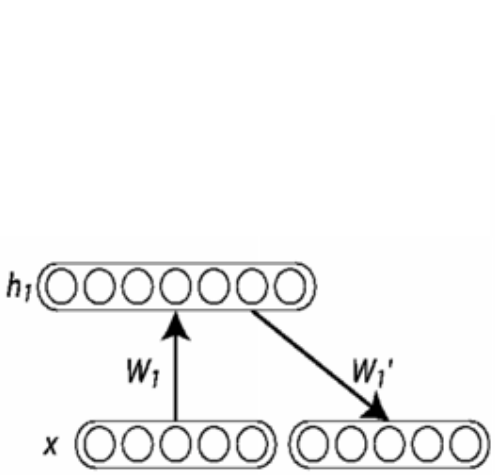
- **The stochastic corruption process** : randomly sets some of the inputs (as many as half of them) to zero
- The goal is to predict the missing values from the uncorrupted (i.e., non-missing) values



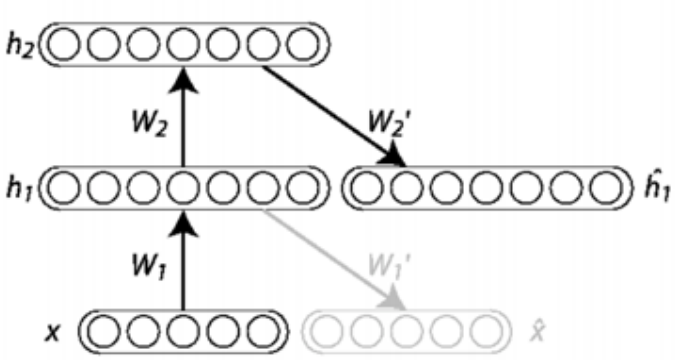
http://deeplearning.net/tutorial/dA.html

Stacked Denoising Auto-encoder

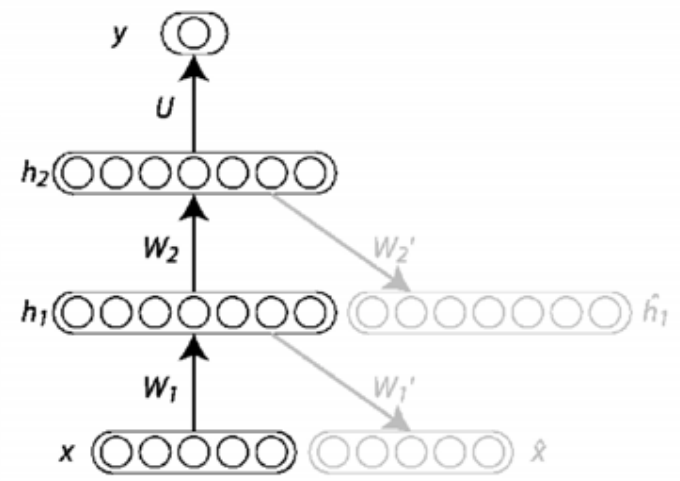
Unsupervised greedy layer-wise training procedure



(1) The first layer pre-training



(2) The second layer pre-training



(3) The while network fine-tuning by label

Parameter matrix *W'* is transpose of *W*

Bengio Y, Lamblin P, Popovici D, et al. Greedy layer-wise training of deep networks[J]. Advances in neural information processing systems, 2007, 19: 153.

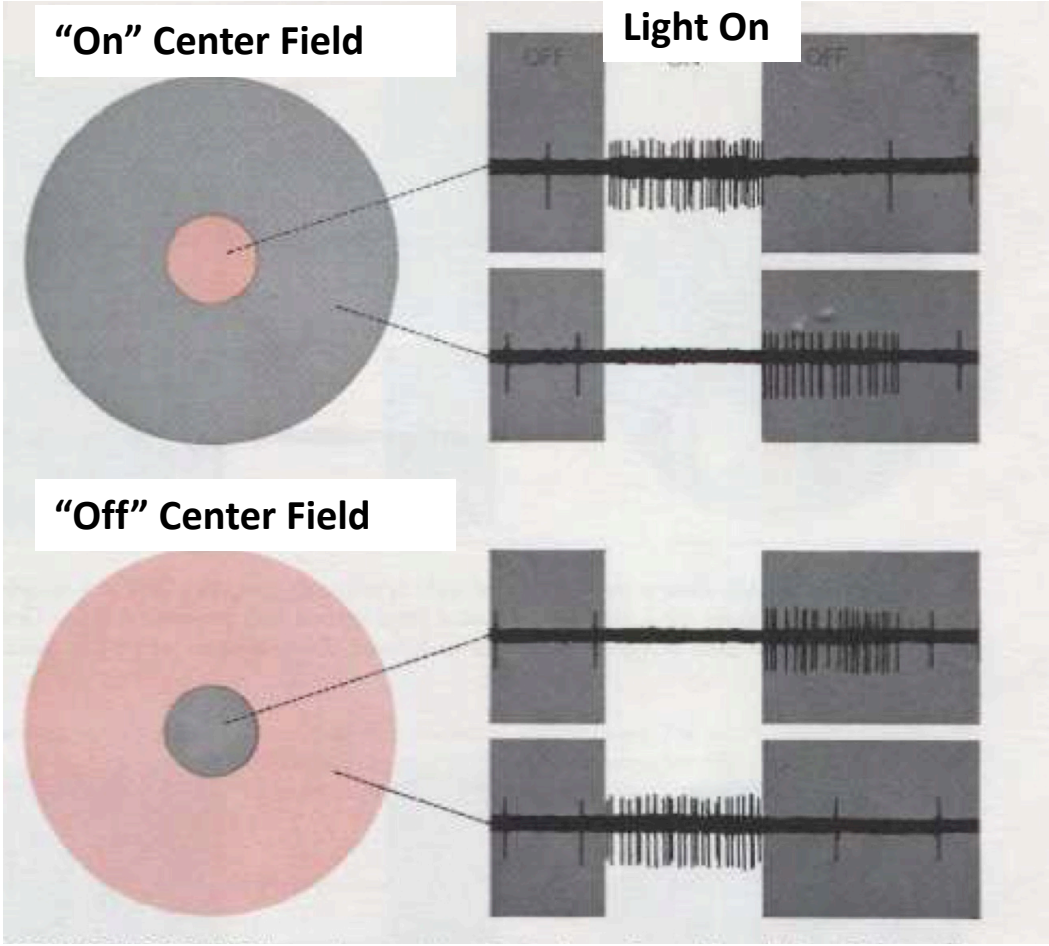
Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

Convolutional neural networks: Receptive field

- Receptive field: Neurons in the retina respond to light stimulus in restricted regions of the visual field

- Animal experiments on receptive fields of two retinal ganglion cells
  - Fields are circular areas of the retina
  - The cell (upper part) responds when the center is illuminated and the surround is darkened.
  - The cell (lower part) responds when the center is darkened and the surround is illuminated.
  - Both cells give on- and off-responses when both center and surround are illuminated, but neither response is as strong as when only center or surround is illuminated



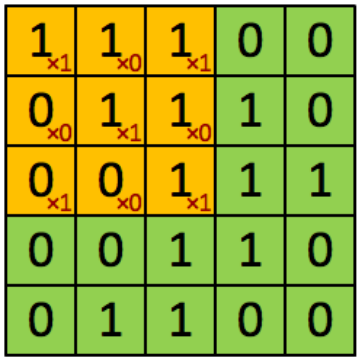
Contributed by Hubel and Wiesel for the studies of the visual system of a cat

Hubel D.H. : The Visual Cortex of the Brain Sci Amer 209:54-62, 1963

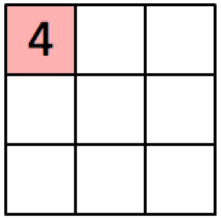
Convolutional neural networks

- Sparse connectivity by local correlation
  - Filter: the input of a hidden unit in layer *m* are from a subset of units in layer *m-1* that have spatially connected receptive fields
- Shared weights
  - each filter is replicated across the entire visual field. These replicated units share the same weights and form a feature map.

2-d case (subscripts are weights)

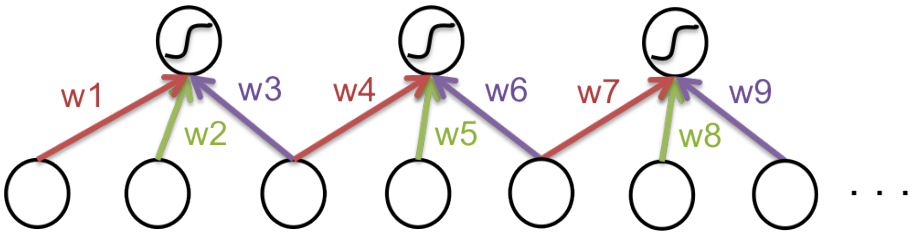


***m*-1 layer**



***one filter at m* layer**

1-d case

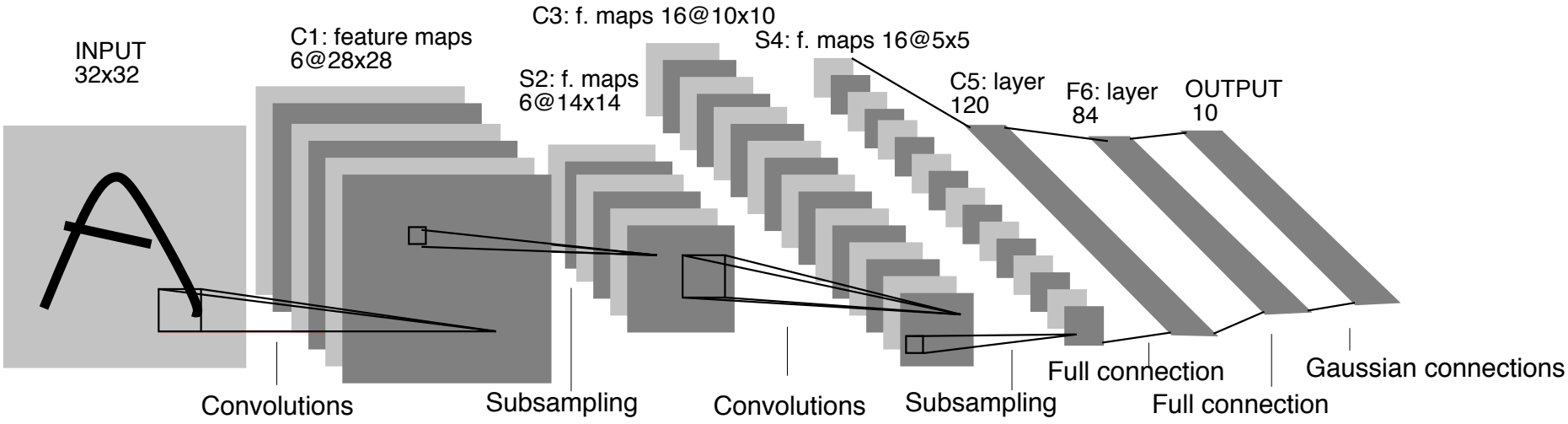


edges that have the same color have the same weight

http://deeplearning.net/tutorial/lenet.html

LeNet 5

- LeNet 5 consists of 7 layers, all of which contain trainable weights;
- End-to end solution: they are used to replace hand crafted feature extraction

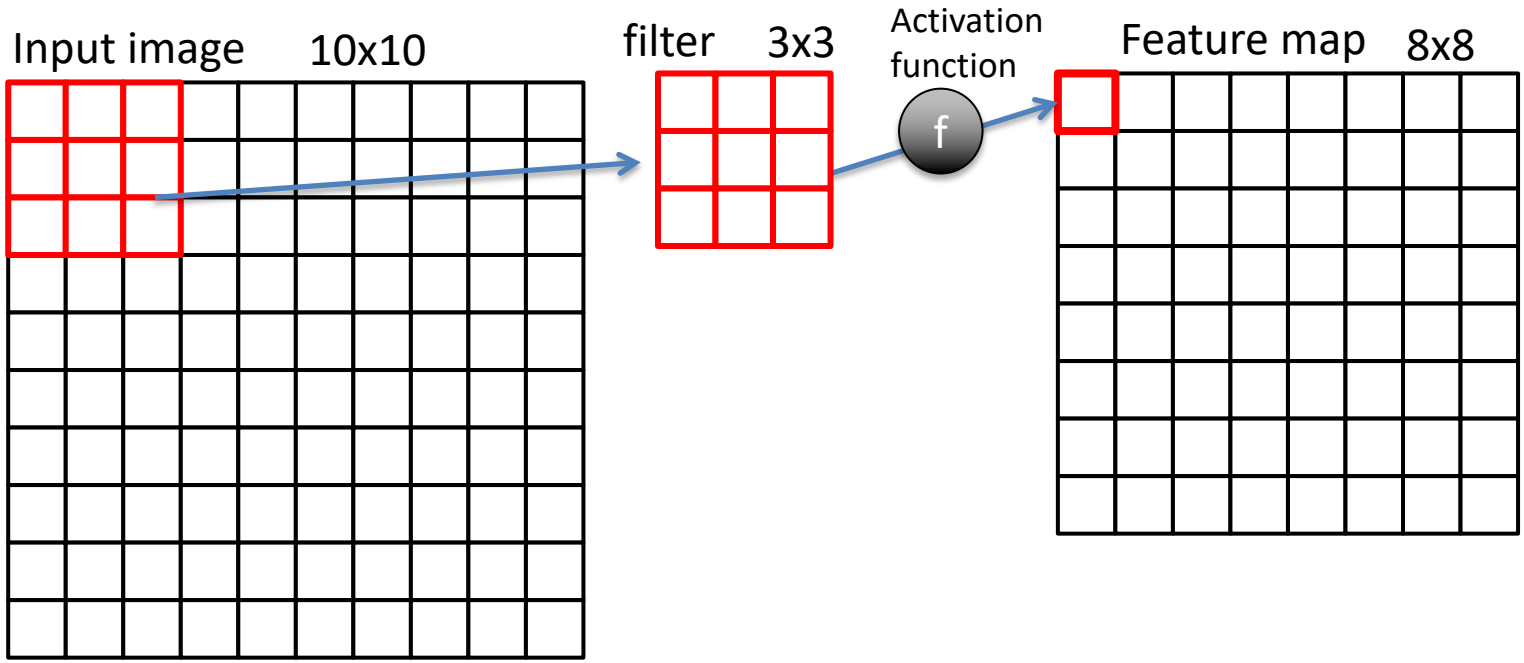


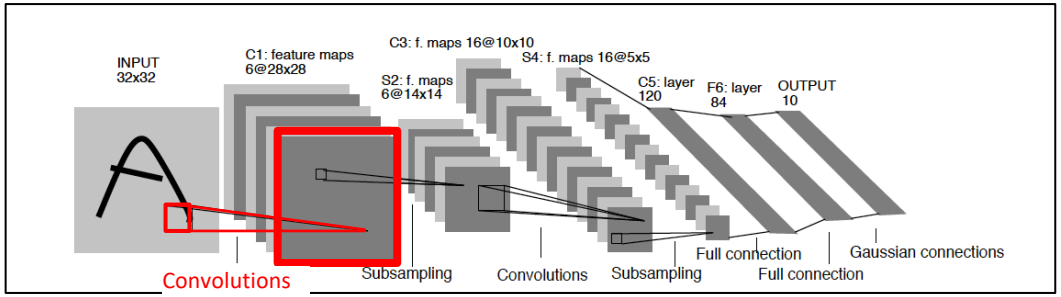
- Cx: convolution layers with 5x5 filter
- Sx: subsampling (pooling) layers
- Fx: fully-connected layers

LeCun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989, 1(4): 541-551.

LeNet 5: Convolution Layer

- Example: a 10x10 input image with a 3x3 filter result in a 8x8 output image

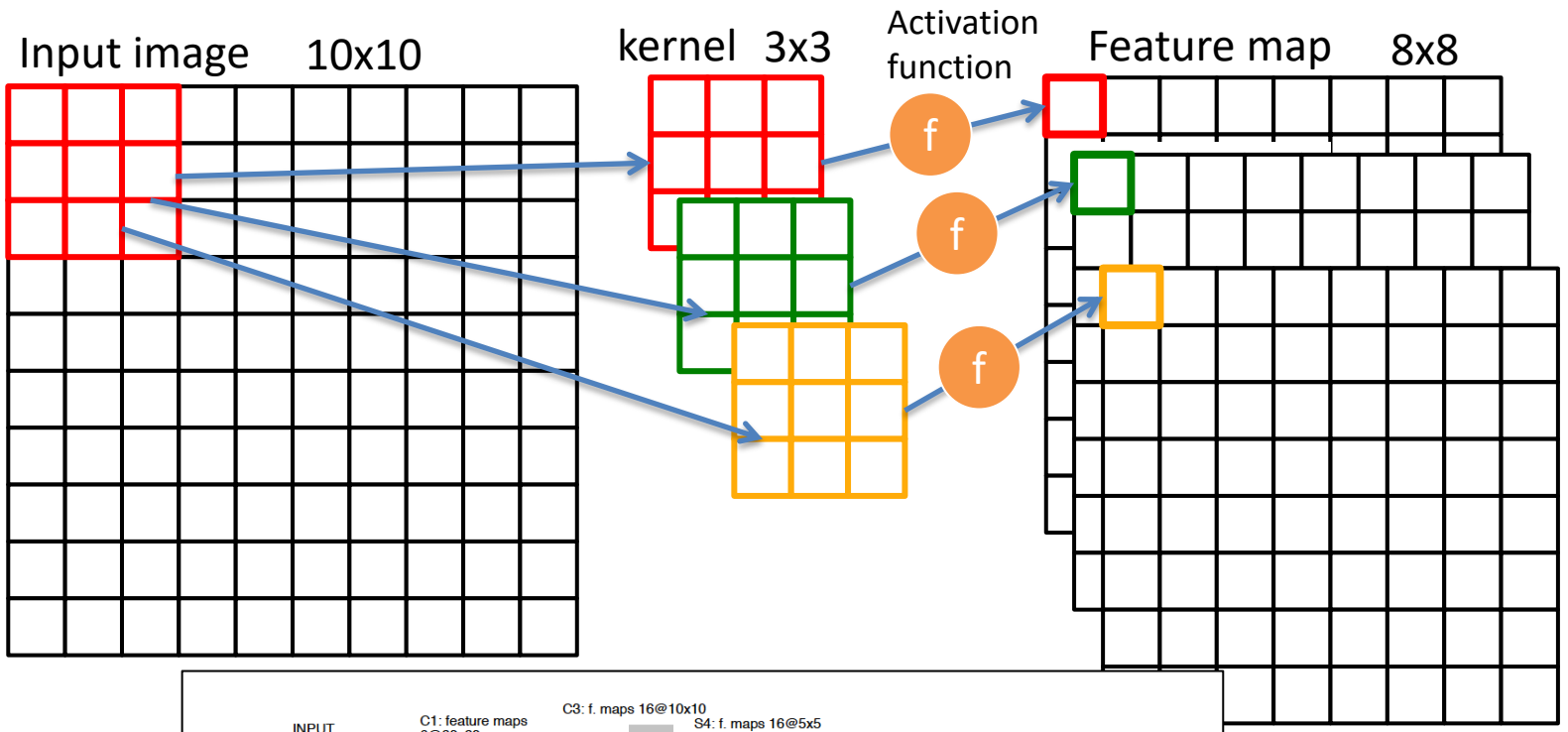


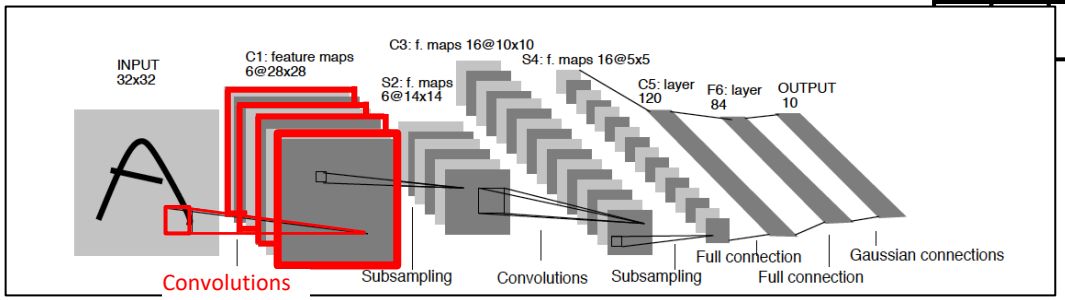


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

LeNet 5: Convolution Layer

- Example: a 10x10 input image with a 3x3 filter result in a 8x8 output image
- 3 different filters (weights are different) lead to 3 8x8 out images.





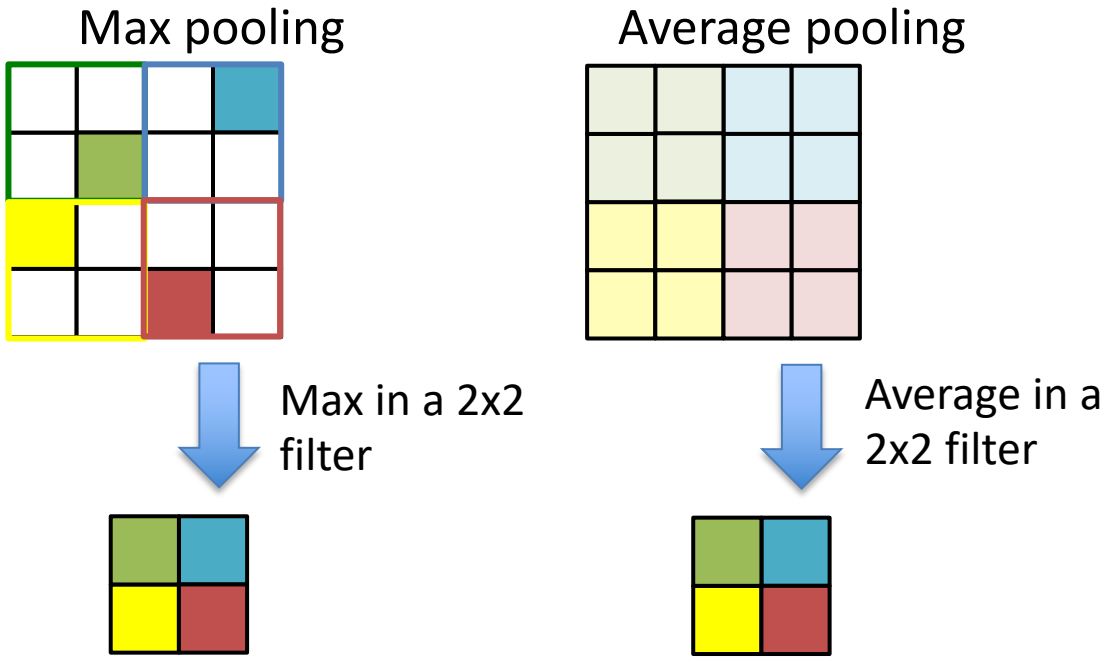
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

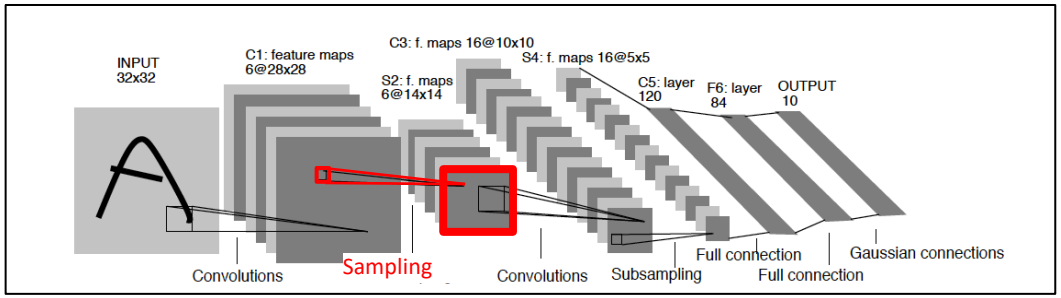
LeNet 5: (Sampling) Pooling Layer

- Pooling: partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum or average value.

Max pooling

- reduces computation and
- is a way of taking the most responsive node of the given interest region,
- but may result in loss of accurate spatial information





Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

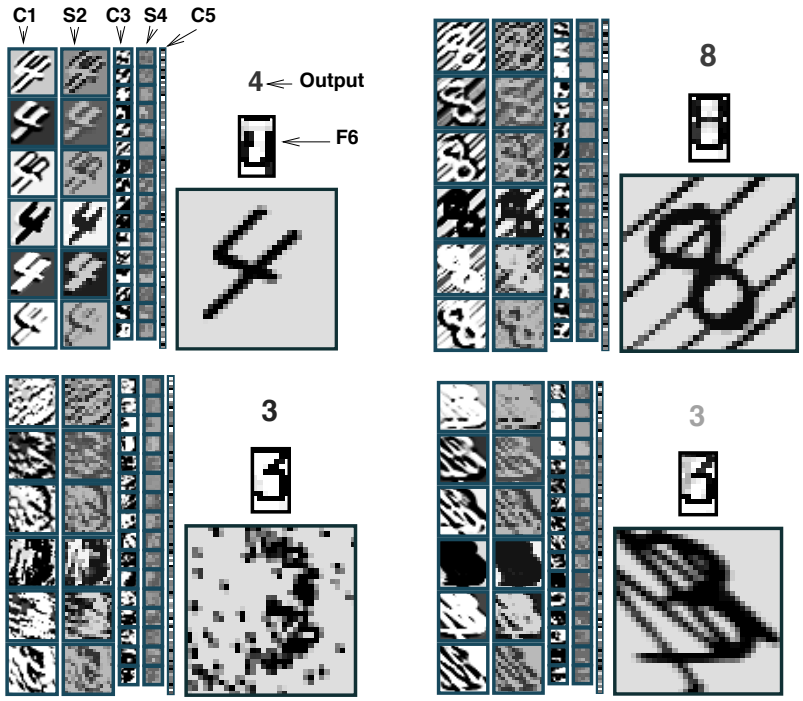
LeNet 5 results

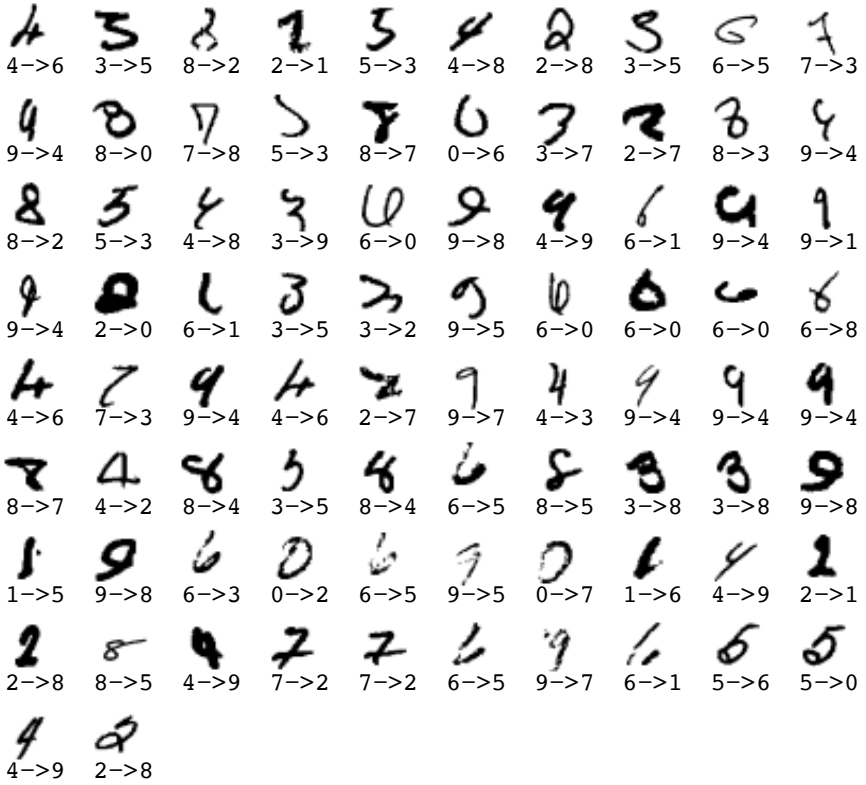
- MNIST (handwritten digits) Dataset:

  http://yann.lecun.com/exdb/mnist/

  – 60k training and 10k test examples

- Test error rate 0.95%



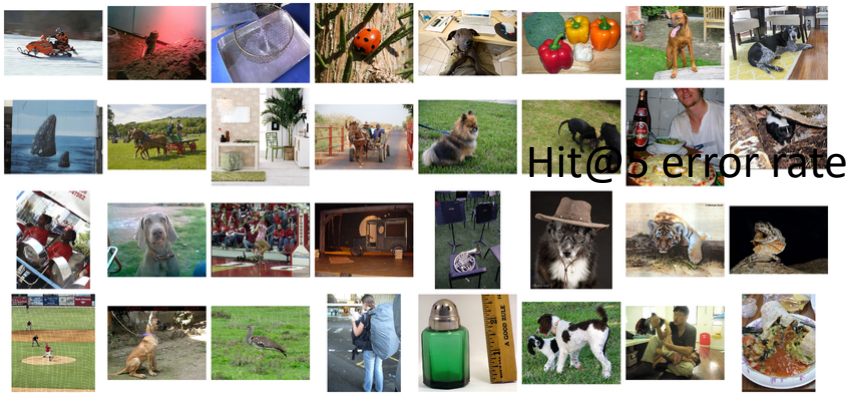


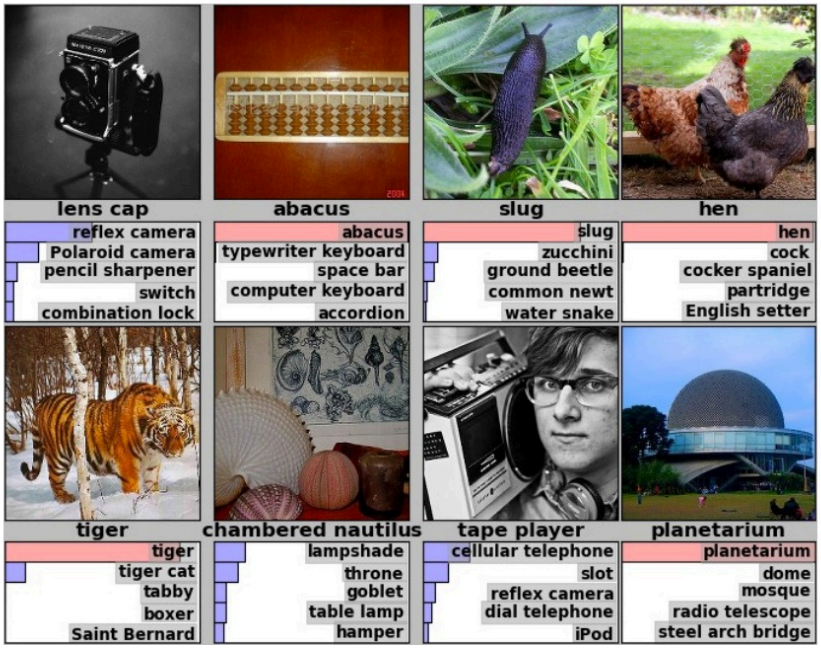
Total only 82 errors from LeNet-5. correct answer left and right is the machine answer.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

From MNIST to ImageNet

- ImageNet
  - Over 15M labeled high resolution images
  - Roughly 22K categories
  - Collected from web and labeled by Amazon Mechanical Turk
- The Image/scene classification challenge
  - Image/scene classification
  - Metric: Hit@5 error rate - make 5 guesses about the image label





http://cognitiveseo.com/blog/6511/will-google-read-rank-images-near-future/

Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

Leadertable (ImageNet image classification)

2015 ResNet (ILSVRC'15) 3.57

Year	Codename	Error (percent)	99.9% Conf Int
2014	**GoogLeNet**	**6.66**	**6.40 - 6.92**
2014	VGG	7.32	7.05 - 7.60
2014	MSRA	8.06	7.78 - 8.34
2014	AHoward	8.11	7.83 - 8.39
2014	DeeperVision	9.51	9.21 - 9.82
2013	Clarifai†	11.20	10.87 - 11.53
2014	CASIAWS†	11.36	11.03 - 11.69
2014	Trimps†	11.46	11.13 - 11.80
2014	Adobe†	11.58	11.25 - 11.91
2013	**Clarifai**	**11.74**	**11.41 - 12.08**
2013	NUS	12.95	12.60 - 13.30
2013	ZF	13.51	13.14 - 13.87
2013	AHoward	13.55	13.20 - 13.91
2013	OverFeat	14.18	13.83 - 14.54
2014	Orange†	14.80	14.43 - 15.17
2012	SuperVision†	15.32	14.94 - 15.69
2012	**SuperVision**	**16.42**	**16.04 - 16.80**
2012	ISI	26.17	25.71 - 26.65
2012	VGG	26.98	26.53 - 27.43
2012	XRCE	27.06	26.60 - 27.52
2012	UvA	29.58	29.09 - 30.04

Microsoft ResNet, a 152 layers network

GoogLeNet, 22 layers network

U. of Toronto, SuperVision, a 7 layers network

Unofficial human error is around *5.1%* on a subset

Why human error still? When labeling, human raters judged whether it belongs to a class (binary classification); the challenge is a 1000-class classification problem.

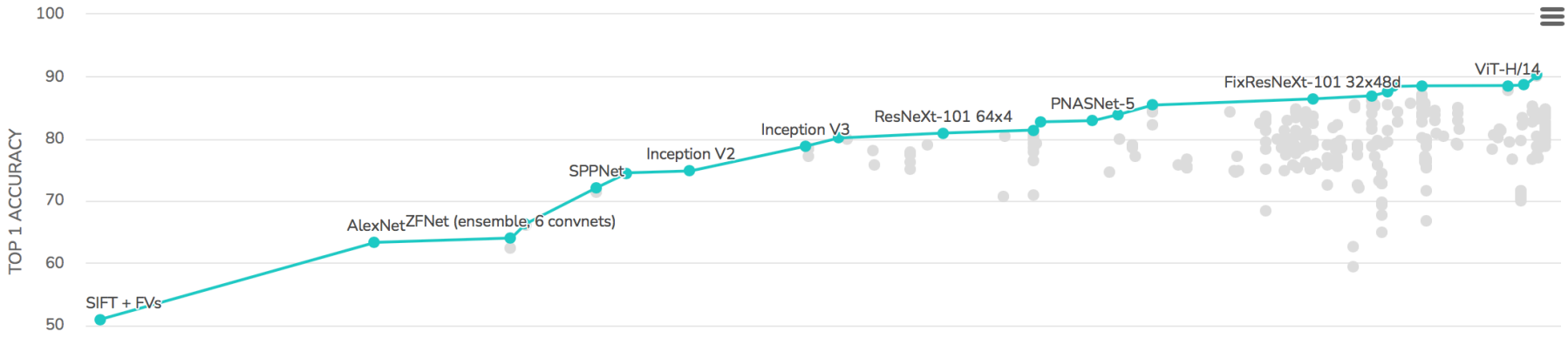
http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/)

Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

Top-1 accuracy as of 2021

Image Classification on ImageNet

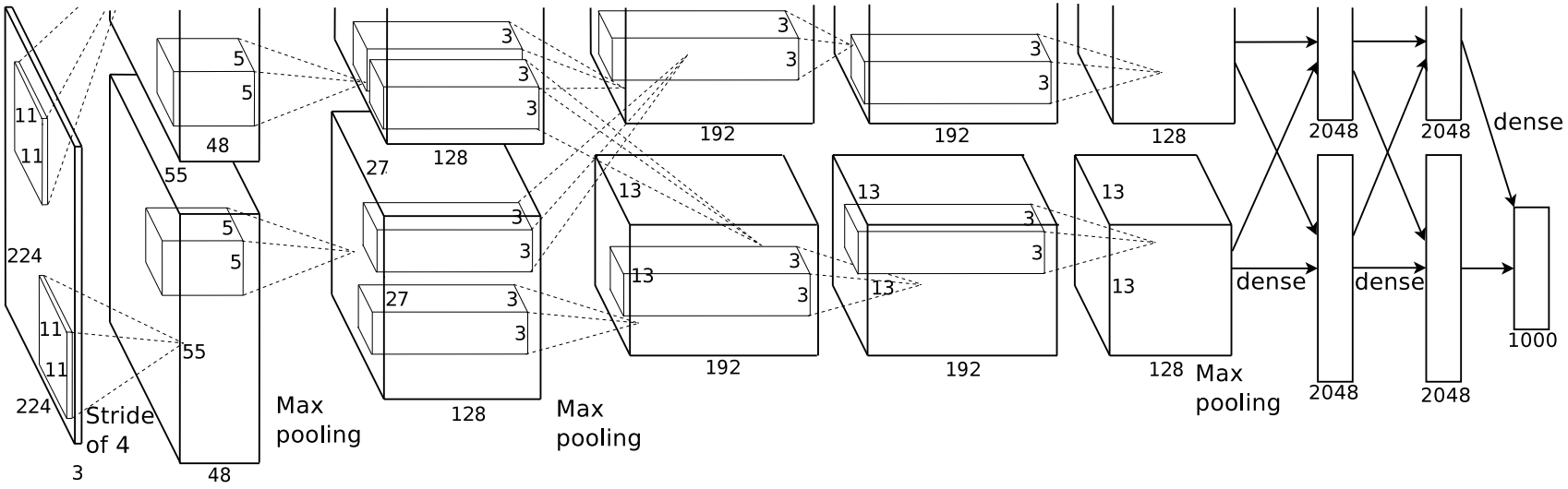
Leaderboard Dataset



RANK	MODEL	TOP 1 ACCURACY	TOP 5 ACCURACY	NUMBER OF PARAMS	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	**Meta Pseudo Labels** (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels			2021
2	**Meta Pseudo Labels** (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels			2021
3	**EfficientNet-L2-475** (SAM)	88.61%		480M	✓	Sharpness-Aware Minimization for Efficiently Improving Generalization			2020
4	**ViT-H/14**	88.55%		632M	✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020

https://paperswithcode.com/sota/image-classification-on-imagenet

SuperVision

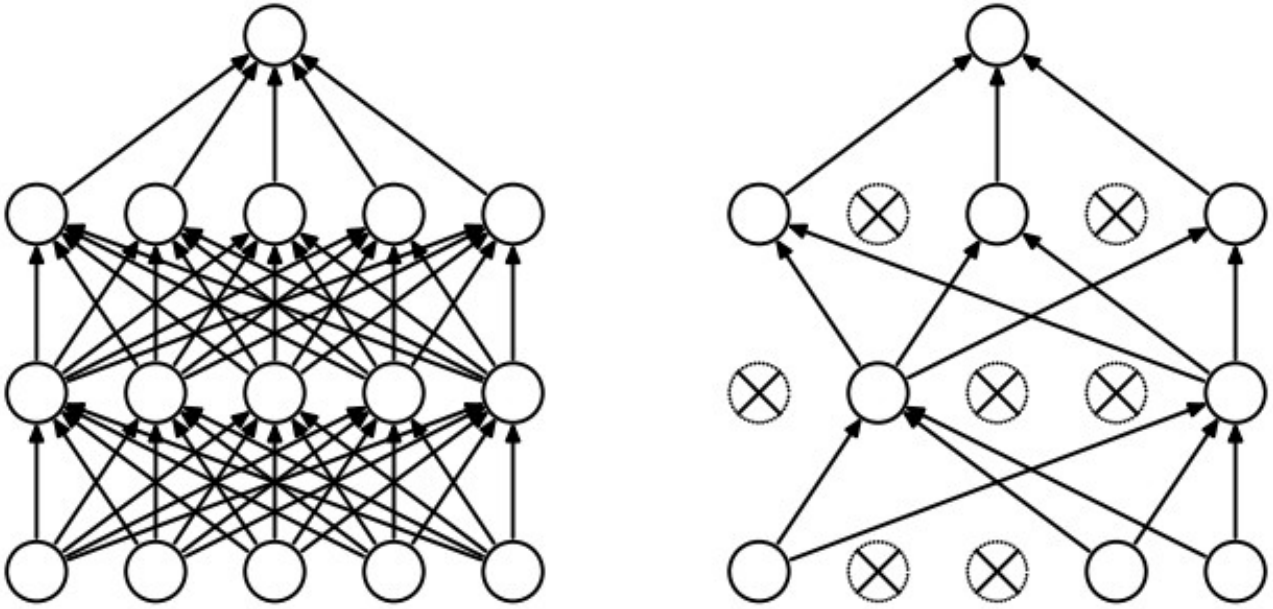


- Rooted from leNet-5
- Rectified Linear Units, overlapping pooling, *dropout* trick
- Randomly extracted 224x224 patches from 256x256 images for more training data

Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems. 2012: 1097-1105.

dropout

- Dropout randomly ‘drops’ units from a layer on each training step, creating ‘sub-architectures’ within the model.
- It can be viewed as a type of sampling of a smaller network within a larger network
- Prevent Neural Networks from Overfitting



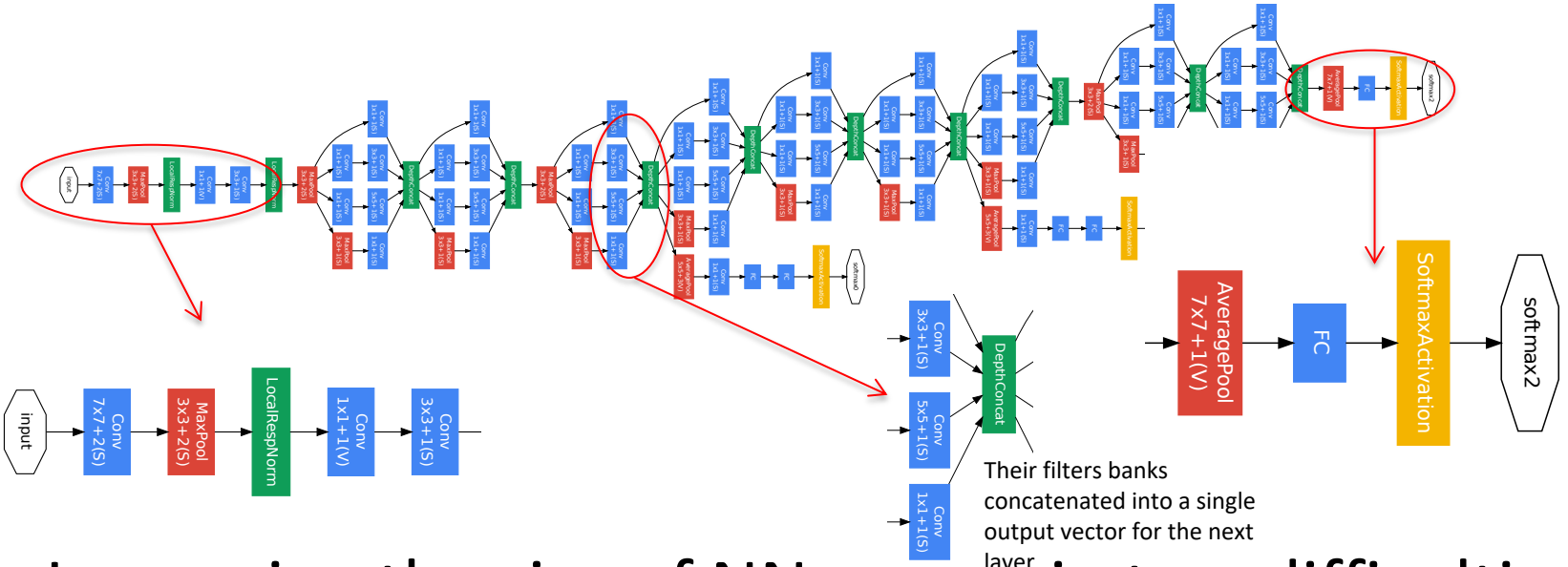
(a) Standard Neural Net

(b) After applying dropout.

Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

Go deeper with CNN

- GoogLeNet, a 22 layers deep network

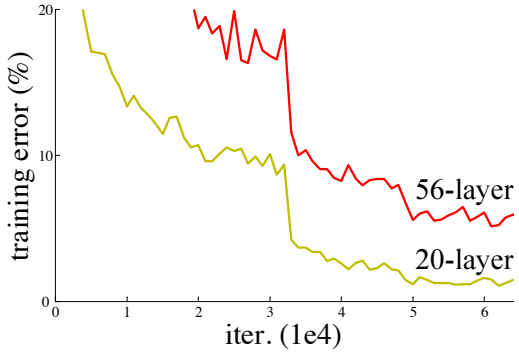


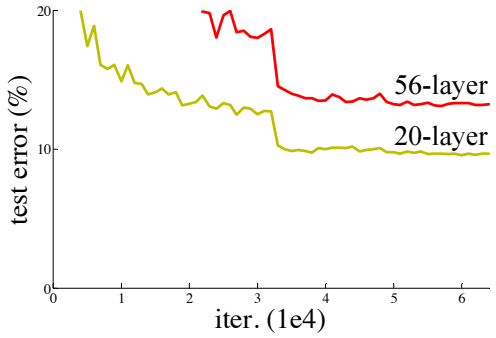
- Increasing the size of NN results in two difficulties
  - 1) overfitting and 2) high computation cost
- Solution: a sparse deep neural network

https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet

ResNet: Deep Residual Learning

- Overfitting prevents a neural network to go deeper
- A Thought experiment:
  - We have a shallower architecture (e.g. logistic regression)
  - Add more layers onto it to construct its deeper counterpart
  - So a deeper model should produce no higher training error than its shallower counterpart, if
    - the added layers are identity mapping, and the other layers are copied from the learned shallower model.
- However, it is reported that adding more layers to a very deep model leads to higher training error



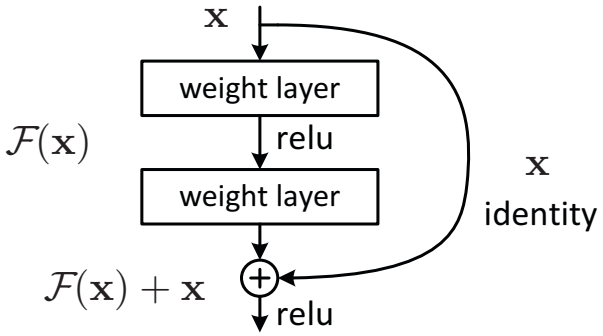


He, Kaiming, et al. "Deep Residual Learning for Image Recognition." arXiv preprint arXiv:1512.03385 (2015).

https://github.com/KaimingHe/deep-residual-networks

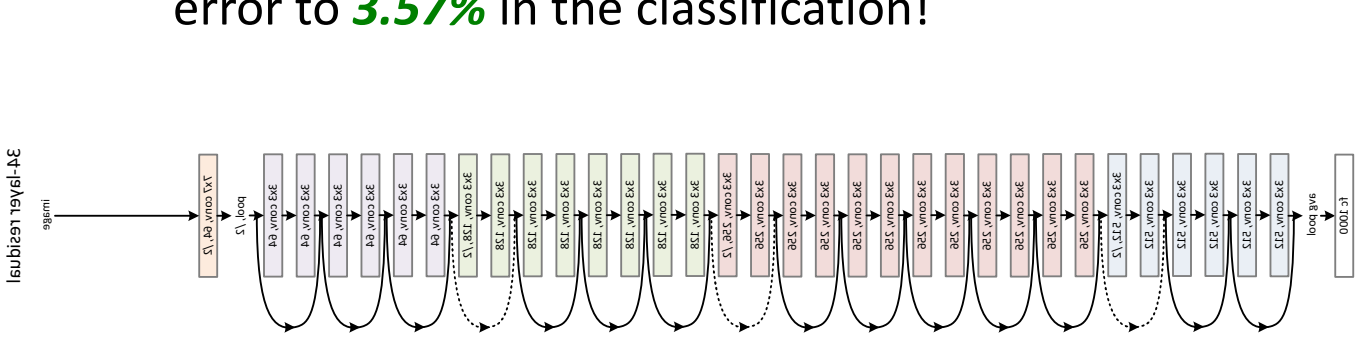
ResNet: Deep Residual Learning

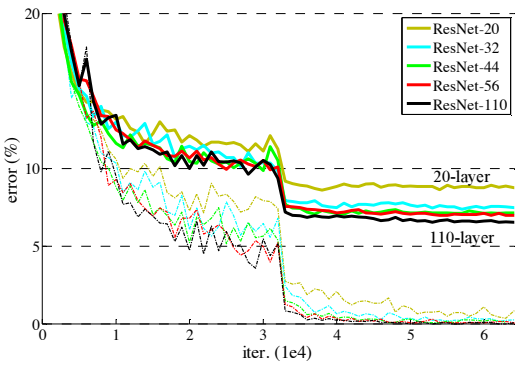
- Solution: feedforward neural networks with "shortcut connections"



1. the desired underlying mapping as *H(x)*.
2. In stead of fitting *H(x)*, a stack of nonlinear layers fit another mapping of *F(x) := H(x)−x*.
3. The original mapping is recast into *F(x)+x*
4. During training, it would be easy to push the residual (*F(x)*) to zero (create an identity mapping) by a stack of nonlinear layers.

- This leads to incredible ***152-layer*** deep convolution network, driving the error to ***3.57%*** in the classification!

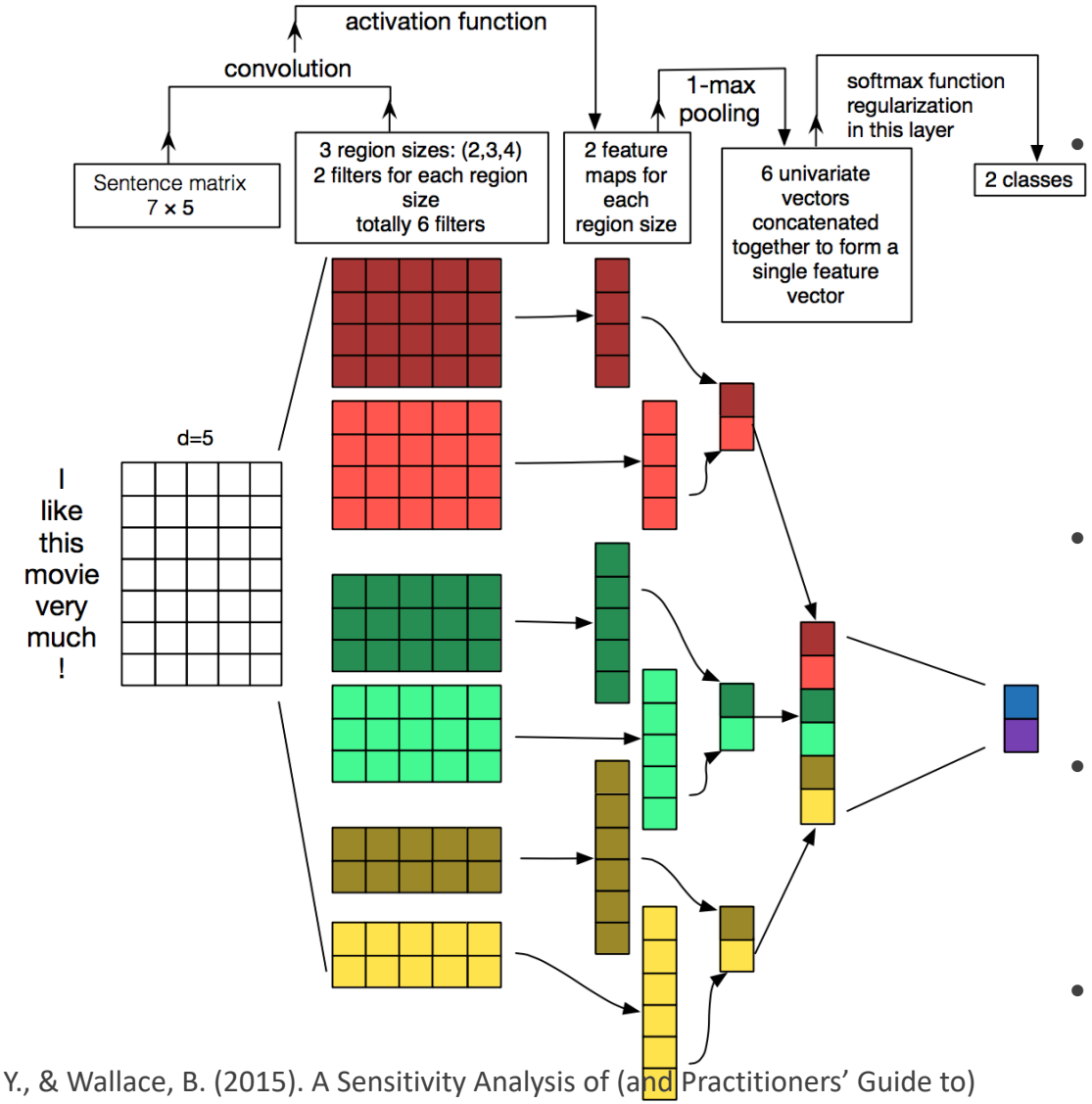




He, Kaiming, et al. "Deep Residual Learning for Image Recognition." arXiv preprint arXiv:1512.03385 (2015).

https://github.com/KaimingHe/deep-residual-networks

CNN for text classification

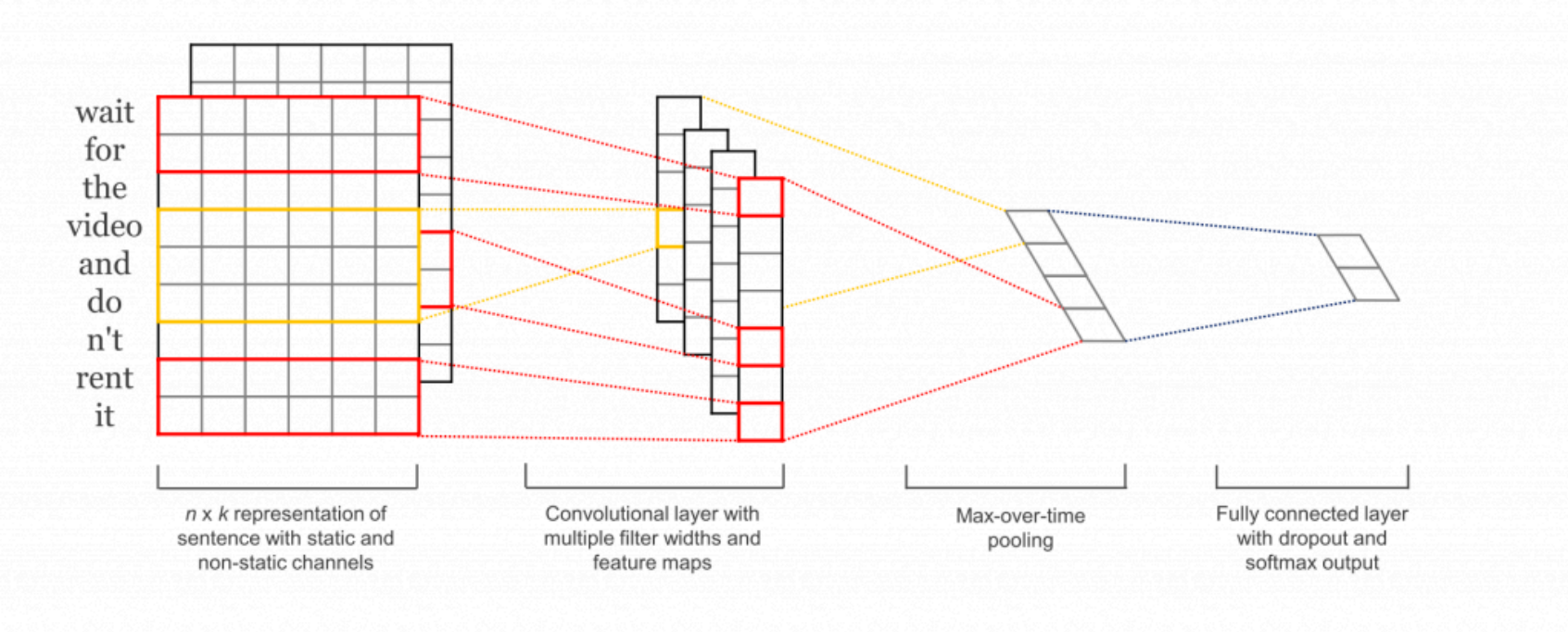


- Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps.
- Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and
- these 6 features are concatenated to form a feature vector for the penultimate layer.
- The final softmax layer then receives this feature vector as input and uses it to classify the sentence
- here we assume binary classification and hence depict two possible output states

Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

CNN for text classification

- CNN architecture on various classification problems such as Sentiment Analysis and Topic Categorization tasks



Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification

Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

Recurrent neural networks

- Why we need another NN model
  - Sequential prediction
  - Time series

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, ..., w_{i-1})$$

- A simple recurrent neural network (RNN)
  - Training can be done by BackPropagation Through Time (BPTT)

x : input vector, o : output vector,
s : hidden state vector,
U : layer 1 param. matrix,
V : layer 2 param. matrix,
f : tanh or ReLU

$o = f(s\mathbf{V})$

$s = f(x\mathbf{U})$

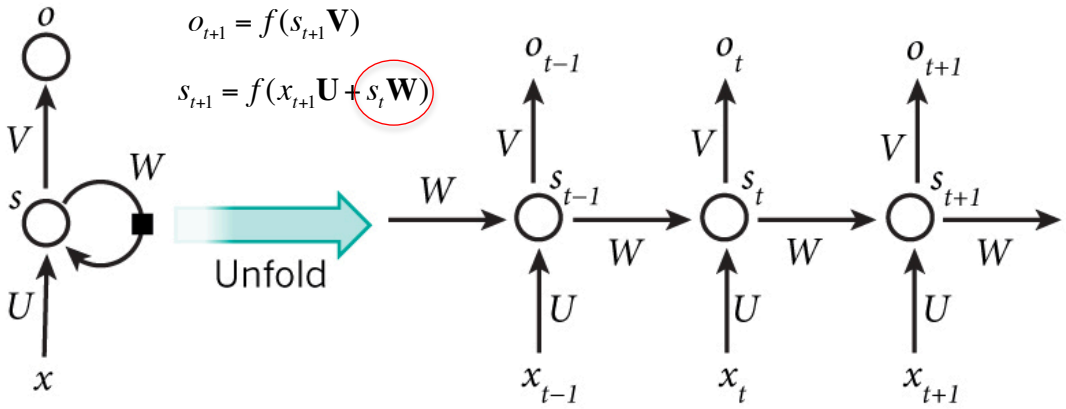
Two-layer feedforward network

Add time-dependency of the hidden state *s*

W : State transition param. matrix

$o_{t+1} = f(s_{t+1}\mathbf{V})$

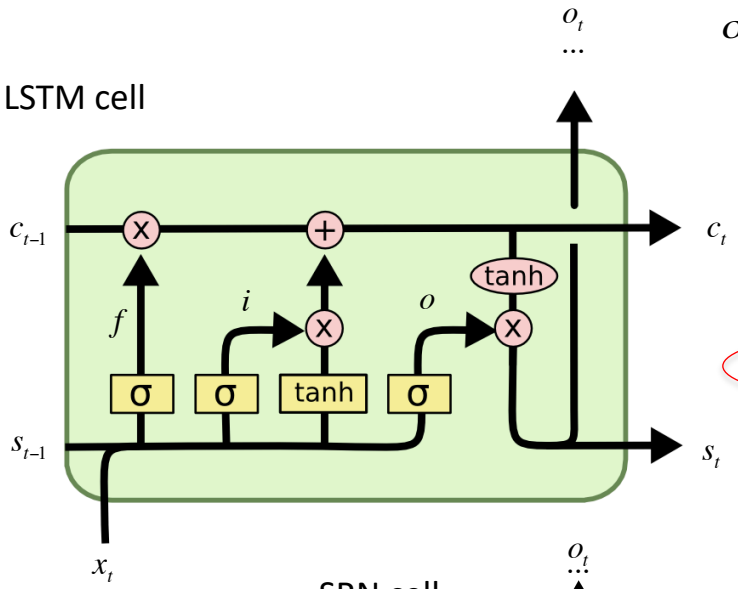
$s_{t+1} = f(x_{t+1}\mathbf{U} + s_t\mathbf{W})$

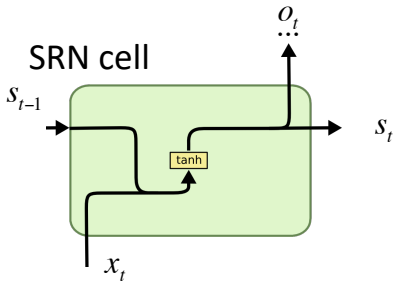


http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

LSTMs (Long Short Term Memory networks)

- Standard Recurrent Network (SRN) has a problem of learning long-term dependency (due to *vanishing gradients of saturated nodes*)
- a LSTM provides another way to compute a hidden state





σ : sigmoid (control signal between 0 and 1); o: elementwise multiplication

$$i = \sigma(x_t U^i + s_{t-1} W^i)$$ input gate

$$f = \sigma(x_t U^f + s_{t-1} W^f)$$ forget gate

$$o = \sigma(x_t U^o + s_{t-1} W^o)$$ output gate

$$g = tanh(x_t U^g + s_{t-1} W^g)$$ "candidate" hidden state:

$$c_t = c_{t-1} \circ f + g \circ i$$ Cell internal memory

$$s_t = \tanh(c_t) \circ o$$ Hidden state

$$s_t = \tanh(x_t \mathbf{U} + s_{t-1} \mathbf{W})$$

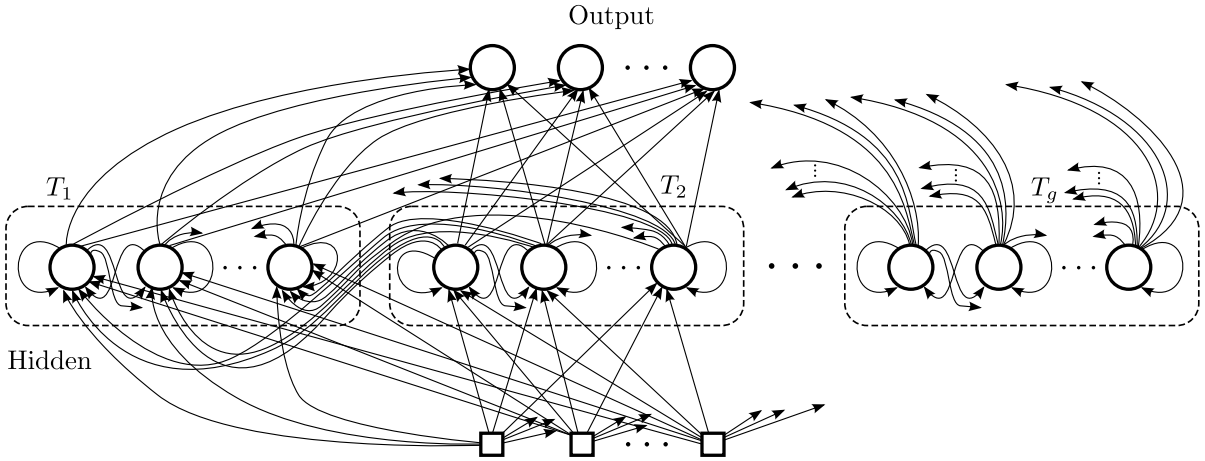
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

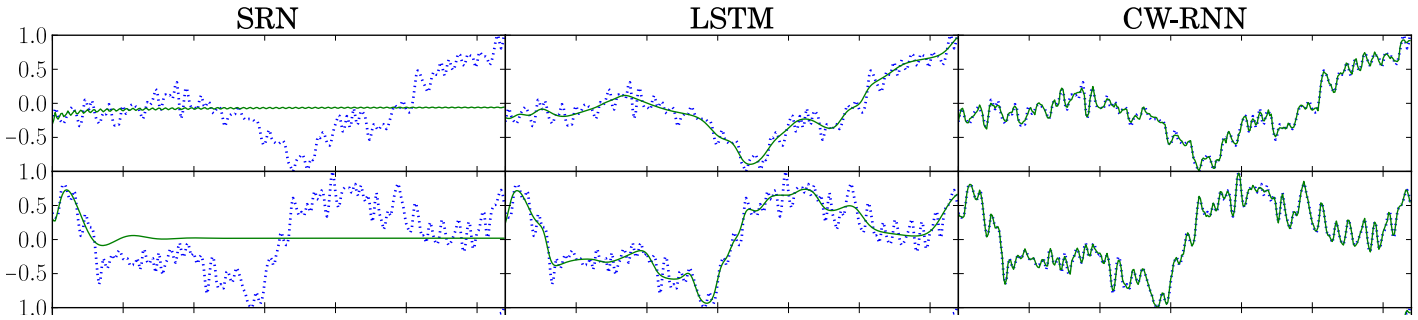
http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

RNN applications: time series

- ClockWork-RNN is similar to a simple RNN with an input, output and hidden layer
- Difference lies in
  - The hidden layer is partitioned into g modules each with its own clock rate
  - Neurons in faster module are connected to neurons in a slower module





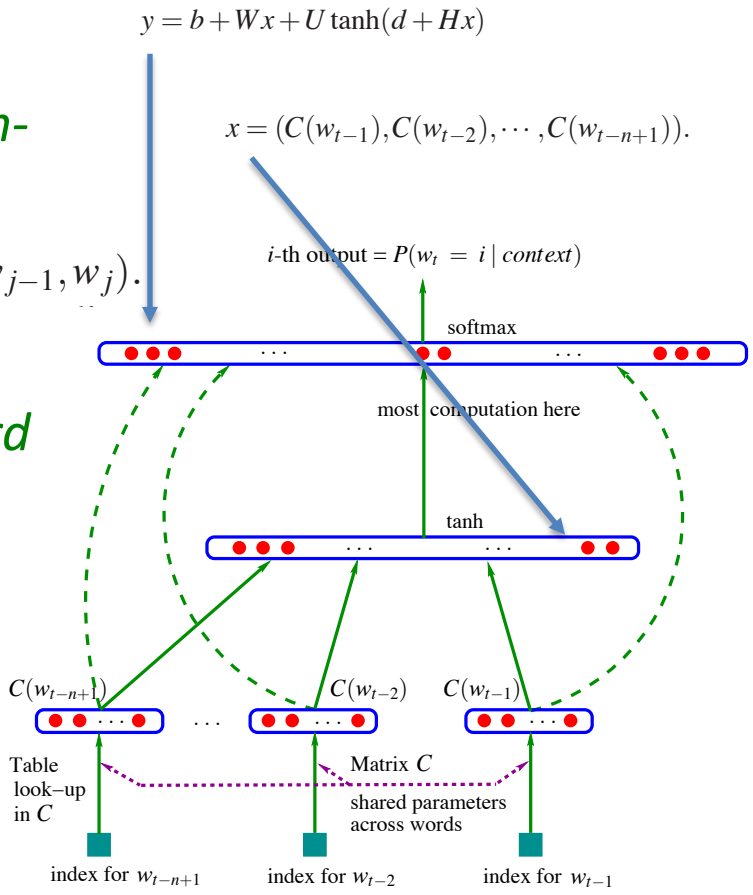
Koutnik, Jan, et al. "A clockwork rnn." arXiv preprint arXiv:1402.3511 (2014).

Neural Language models

- *n-gram* model
  - Construct conditional probabilities for the next word, given combinations of the last *n-1* words (*contexts*)

$$\hat{P}(w_t|w_1^{t-1}) \approx \hat{P}(w_t|w_{t-n+1}^{t-1}) \quad \text{where} \quad w_i^j = (w_i, w_{i+1}, \cdots, w_{j-1}, w_j).$$

- Neural language model
  - associate with each word a *distributed word feature vector* for word embedding,
  - express the joint *probability function* of word sequences using those vectors, and
  - learn simultaneously the *word feature vectors* and the parameters of that *probability function*.



Bengio, Yoshua, et al. "Neural probabilistic language models." Innovations in Machine Learning. Springer Berlin Heidelberg, 2006. 137-186.

RNN based Language models

- The limitation of the feedforward network approach:
  - it has to fix the length context
- Recurrent network solves the issue
  - by keeping a (hidden) context and updating over time

x(t) is the input vector:
It is formed by **concatenating** vector *w(t)* representing current word, and hidden state s at time t − 1. w(t) is one hot encoder of a word

$$x(t) = [w(t), s(t-1)]$$

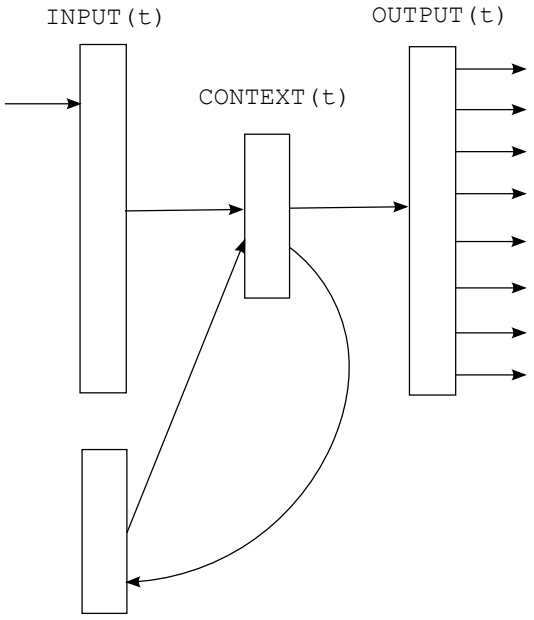
s(t) is state of the network (the hidden layer):

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right)$$

output is denoted as y(t):

$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right)$$

Sigmoid for hidden layer $f(z) = \frac{1}{1+e^{-z}}$ **Softmax for output layer** $g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$



Elman's RNN LM

Mikolov, Tomas, et al. "Recurrent neural network based language model." INTERSPEECH. Vol. 2. 2010.
Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.

Multi-modal modelling

Automated Image Captioning that combines RNN and CNN

a man riding a skateboard down a street
`logprob: -7.50`

a large jetliner flying through a blue sky
`logprob: -5.66`

a group of stuffed animals sitting on top of a table
`logprob: -8.22`

a close up of a remote control on a table
`logprob: -7.67`

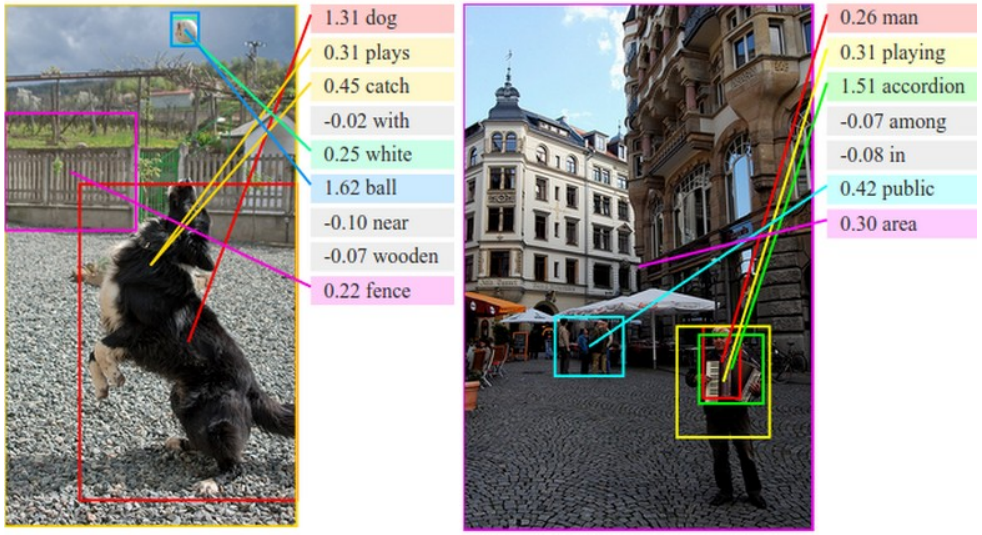
a clock tower with a clock on top of it
`logprob: -8.17`

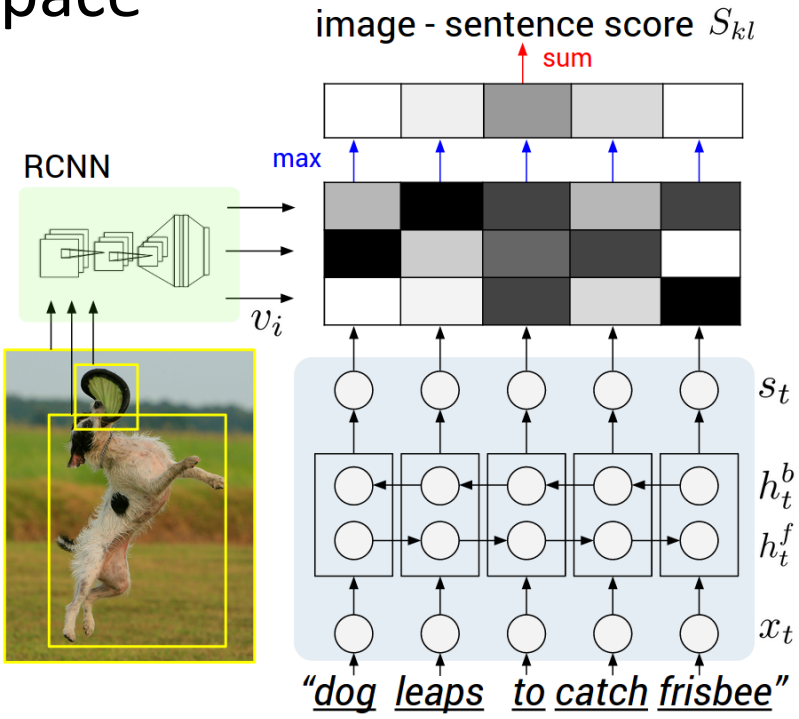
a parking meter is on the side of the road
`logprob: -6.74`

Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

Learning to align visual and language data

- Regional CNN + Bi-directional RNN
  - associates the two modalities through a common, multimodal embedding space



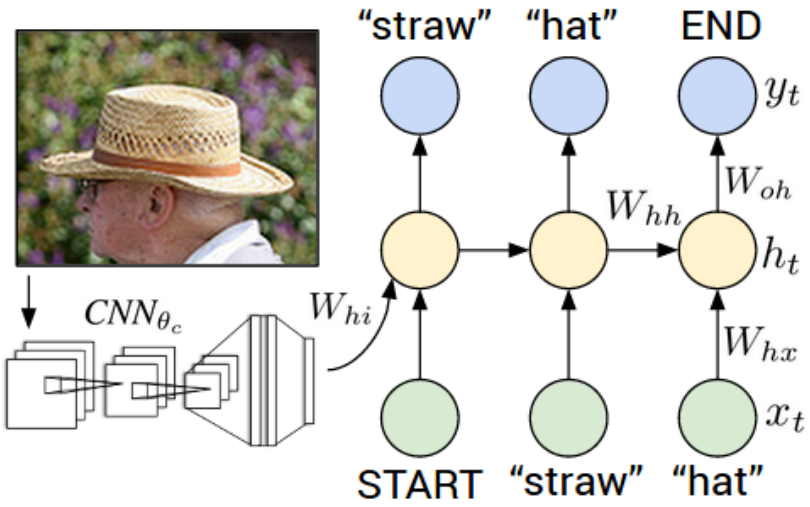


Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

Learning to generate image descriptions

- Trained CNN on images + RNN with sentence
  - The RNN takes a word, the previous context and defines a distribution over the next word
  - The RNN is conditioned on the image information at the first time step
  - START and END are special tokens.

"two young girls are playing with lego toy."



Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

Web Search

- Document:

best home remedies for cold and flu

- Suppose we have queries

Q: cold home remedy

Using stemming to match *remedy* and *remedies*

Q: cold remeedy

Spelling correction: *remeedy-> remedy*

Q: flu treatment

Query expension: *treatment->remedy*

Q: how to deal with stuffy nose

???

Best Home Remedies for Cold and Flu

Wind Heat External Pathogens

By: Catherine Browne, L.Ac., MH, Dipl. Ac.

In Chinese medicine, colds and flu's are delineated into several different energetic classifications. Here we will outline the different types of cold and flu viruses that you will likely encounter, and then describe the best home remedies for these specific patterns that you can use to treat the cold or influenza virus.

Cold and Flu Basics

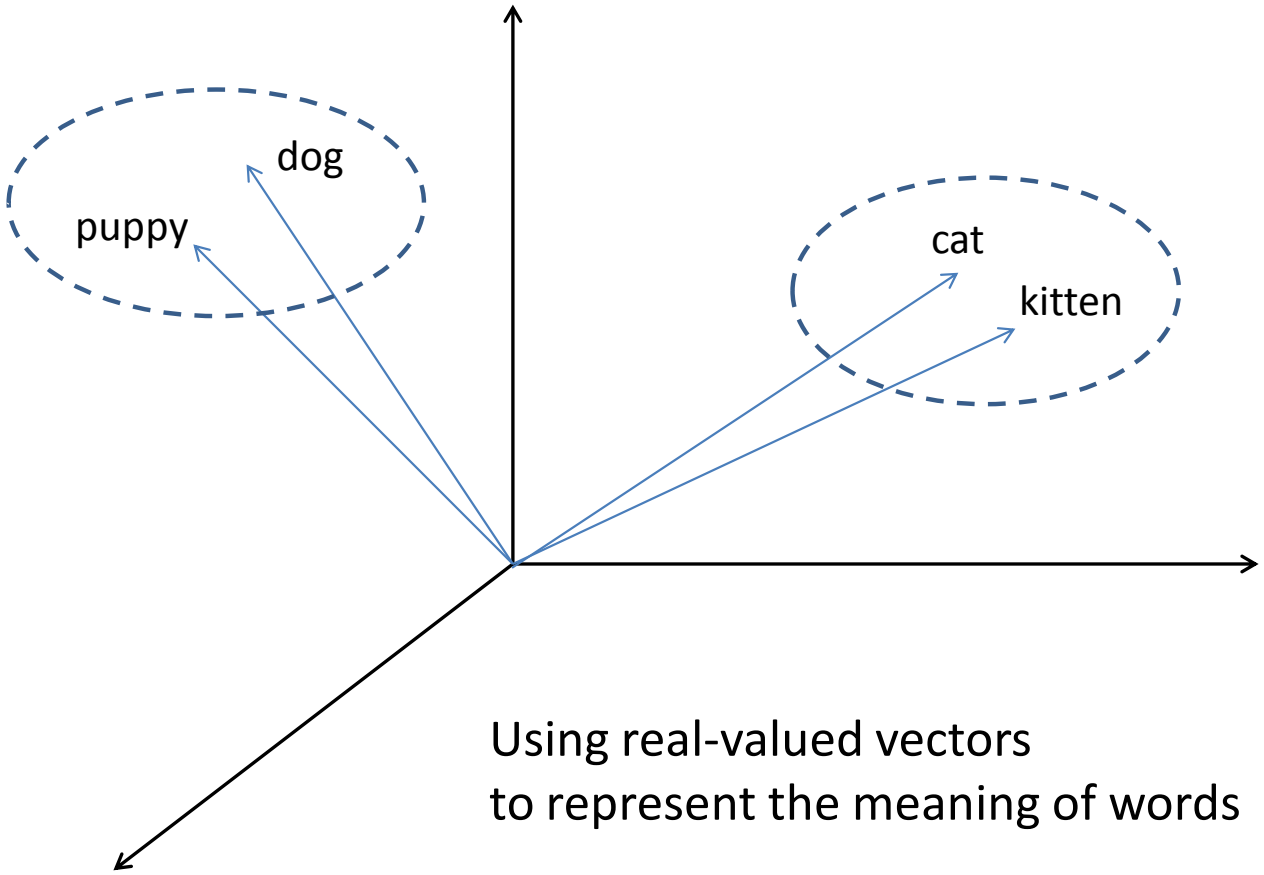
The basic pathogenic influences are:

- **Wind**
- **Cold**
- **Heat**
- **Damp**

Wind

Theoretically, wind enters the body through the back of the neck area or nose carrying the pathogen. It first attacks the Lung system (including the sinuses) because the Lung organ system is the most external Yin organ, a thus the most vulnerable to an external invasion. External Wind invasion is marked by acute conditions with a sudden onset of symptoms.

Distributed representation of words

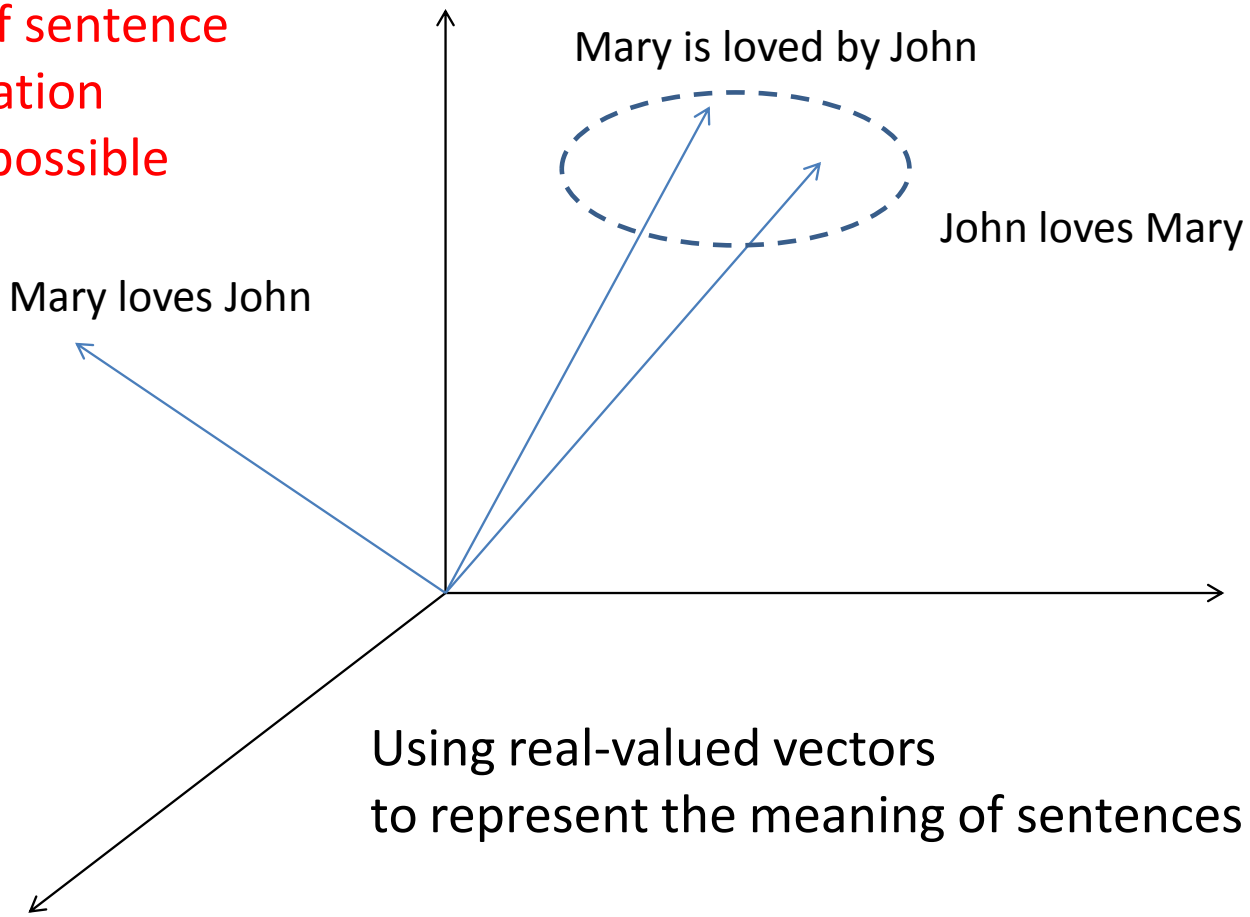


Using real-valued vectors
to represent the meaning of words

Hang Li, Deep Learning for Information Retrieval, SIGIR16

Distributed representation of sentences and documents

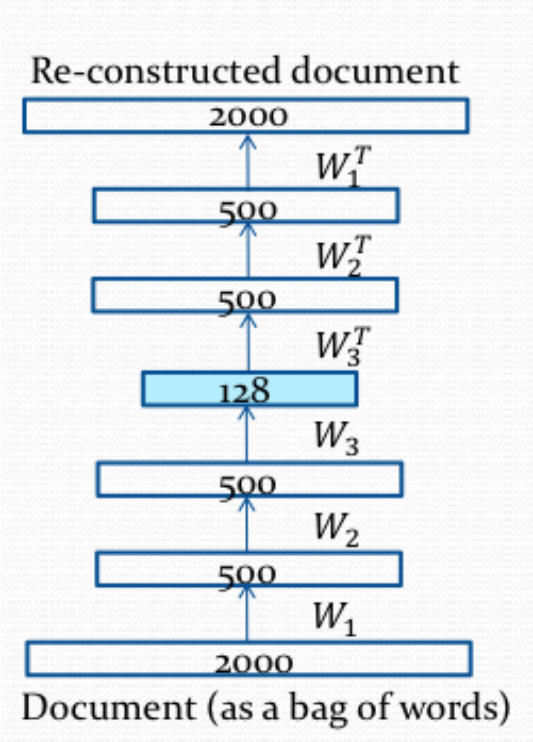
Breakthrough：
learning of sentence representation becomes possible

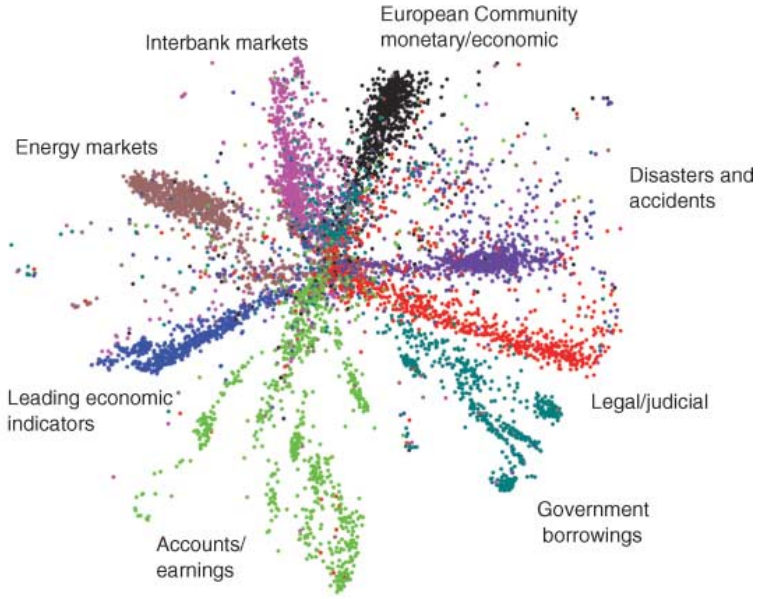


Hang Li, Deep Learning for Information Retrieval, SIGIR16

How about deep autoencoder for semantic space?

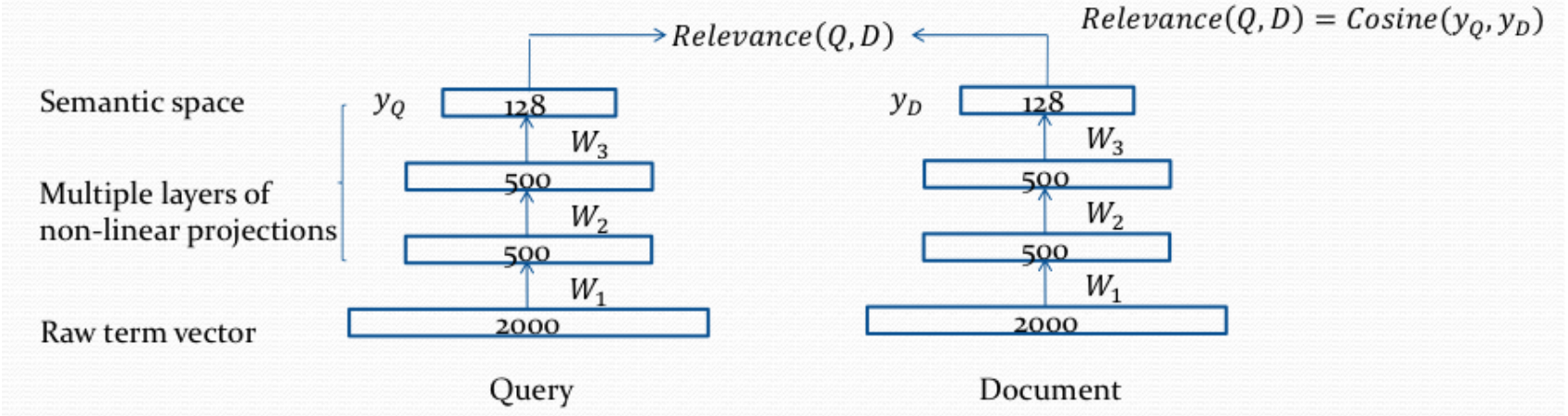
- Recall we can use auto-encoder to build a sematic space





Problem with deep auto-encoder

- Project both query and document to a common semantic space
- Measure the relevance of Q and D in that space directly



- The problem though:
  - Model is trained by reconstructing the document, not for relevance measure
  - Model size increases rapidly along the vocabulary size

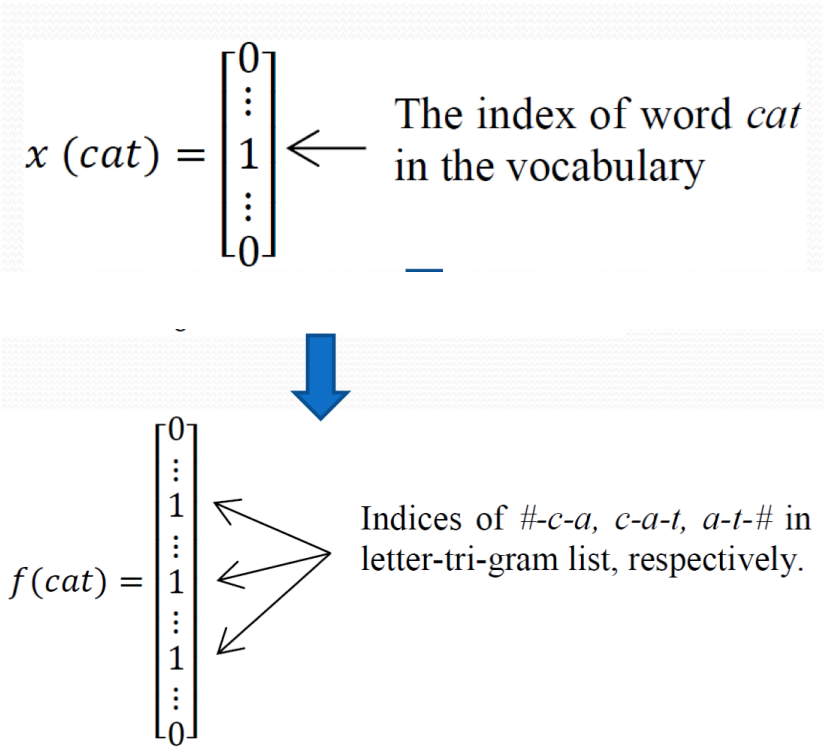
DSSM (Deep Structured Semantic Models) for learning semantic embedding

- Build word/phrase, or sentence-level semantic vector representation
- The idea: trained by a similarity-driven objective
  - projecting semantically similar phrases to vectors close to each other
  - projecting semantically different phrases to vectors far apart
- Using the *tri-letter* based word hashing for scalable word representation
- Using the *deep neural net* to extract high-level semantic representations
- Using the *click signal* to guide the learning

Huang PS, He X, Gao J, Deng L, Acero A, Heck L. Learning deep structured semantic models for web search using clickthrough data. InProceedings of the 22nd ACM international conference on Conference on information & knowledge management 2013 Oct 27 (pp. 2333-2338). ACM

Letter-trigram Representation

- Control the dimensionality of the input space e.g., cat → #cat# → #-c-a, c-a-t, a-t-#
  - Only ~50K letter-trigrams in English;
  - no out-of-vocabulary (OOV) issue
- Capture sub-word semantics (e.g., prefix & suffix)
  - Words with small typos have similar raw representations
- Collision: different words with same letter-trigram representation?

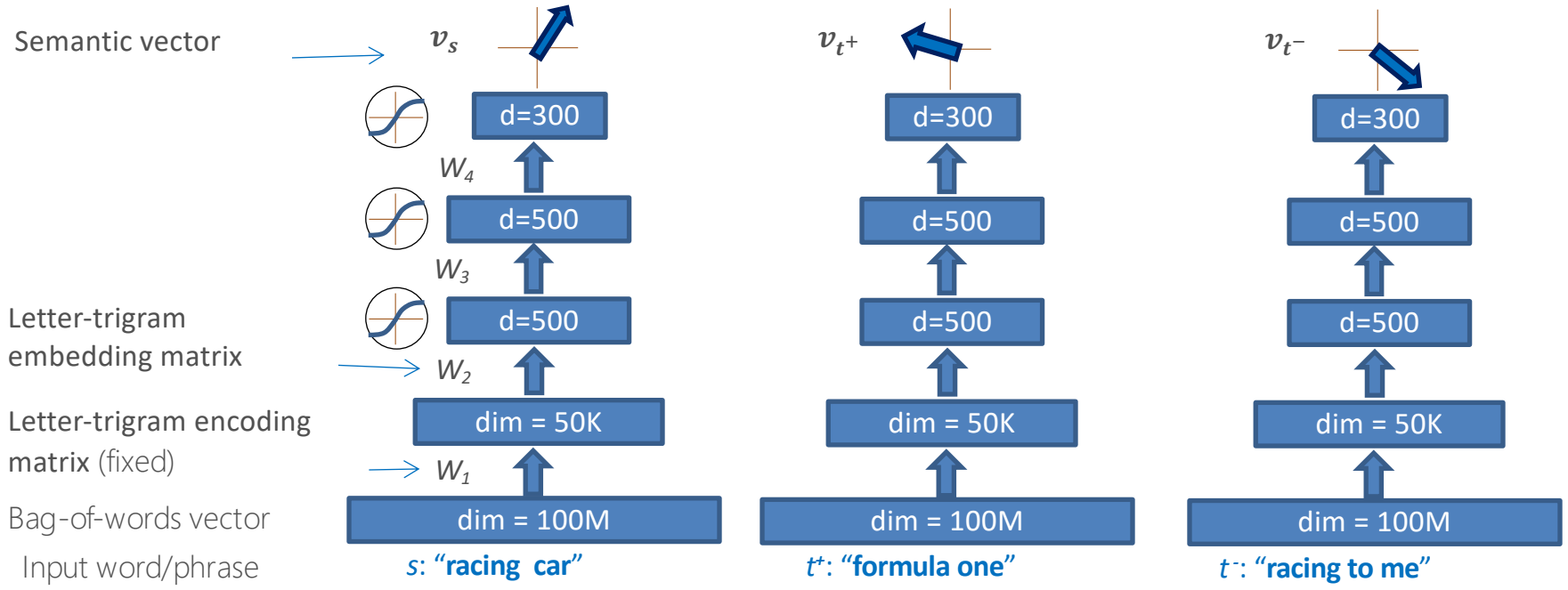


Vocabulary size	# of unique letter-trigrams	# of Collisions	Collision rate
40K	10,306	2	0.0050%
500K	30,621	22	0.0044%
5M	49,292	179	0.0036%

DSSM for learning semantic embedding

Initialization:

Neural networks are initialized with random weights

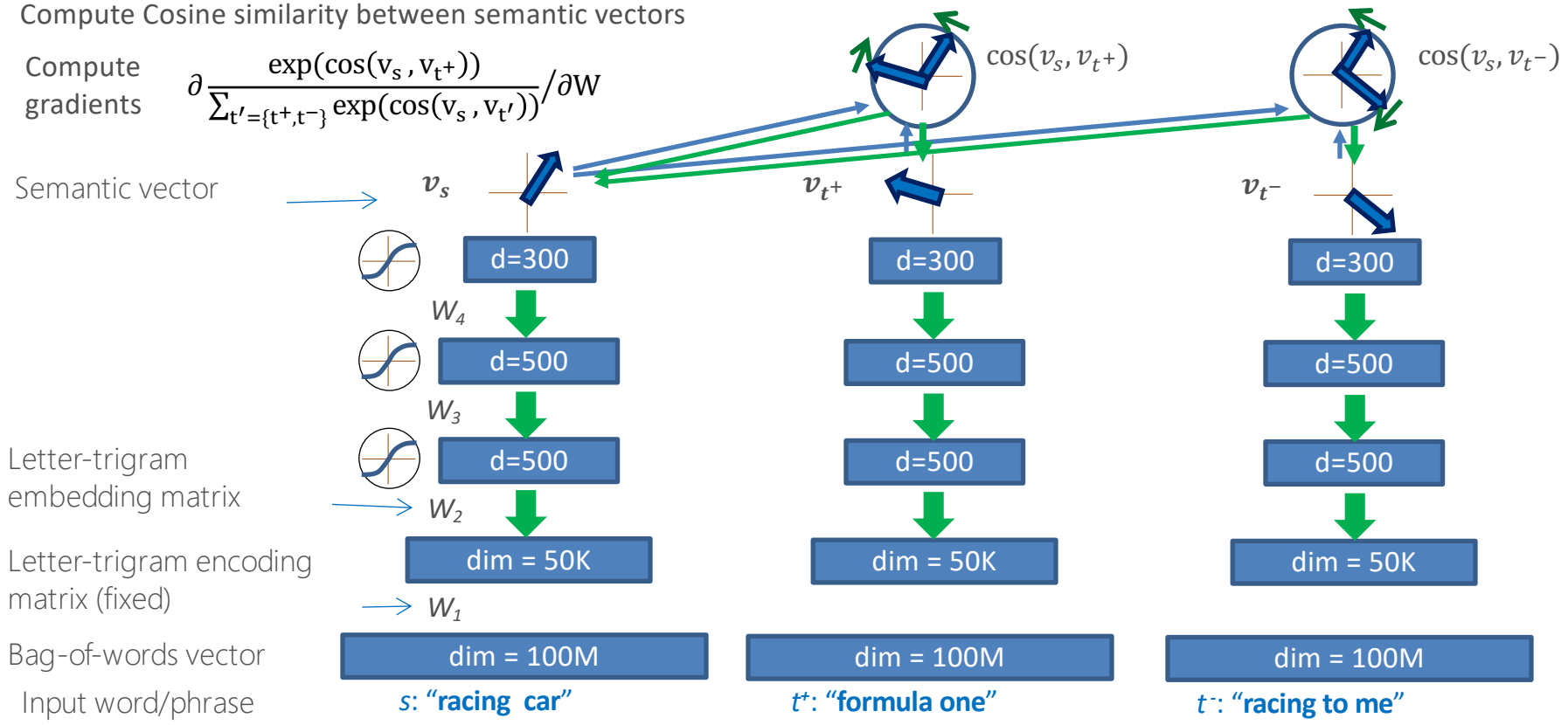


DSSM for learning semantic embedding

Training:

Compute Cosine similarity between semantic vectors

Compute gradients $\partial \frac{\exp(\cos(v_s, v_{t^+}))}{\sum_{t'=\{t^+,t^-\}} \exp(\cos(v_s, v_{t'}))} / \partial W$

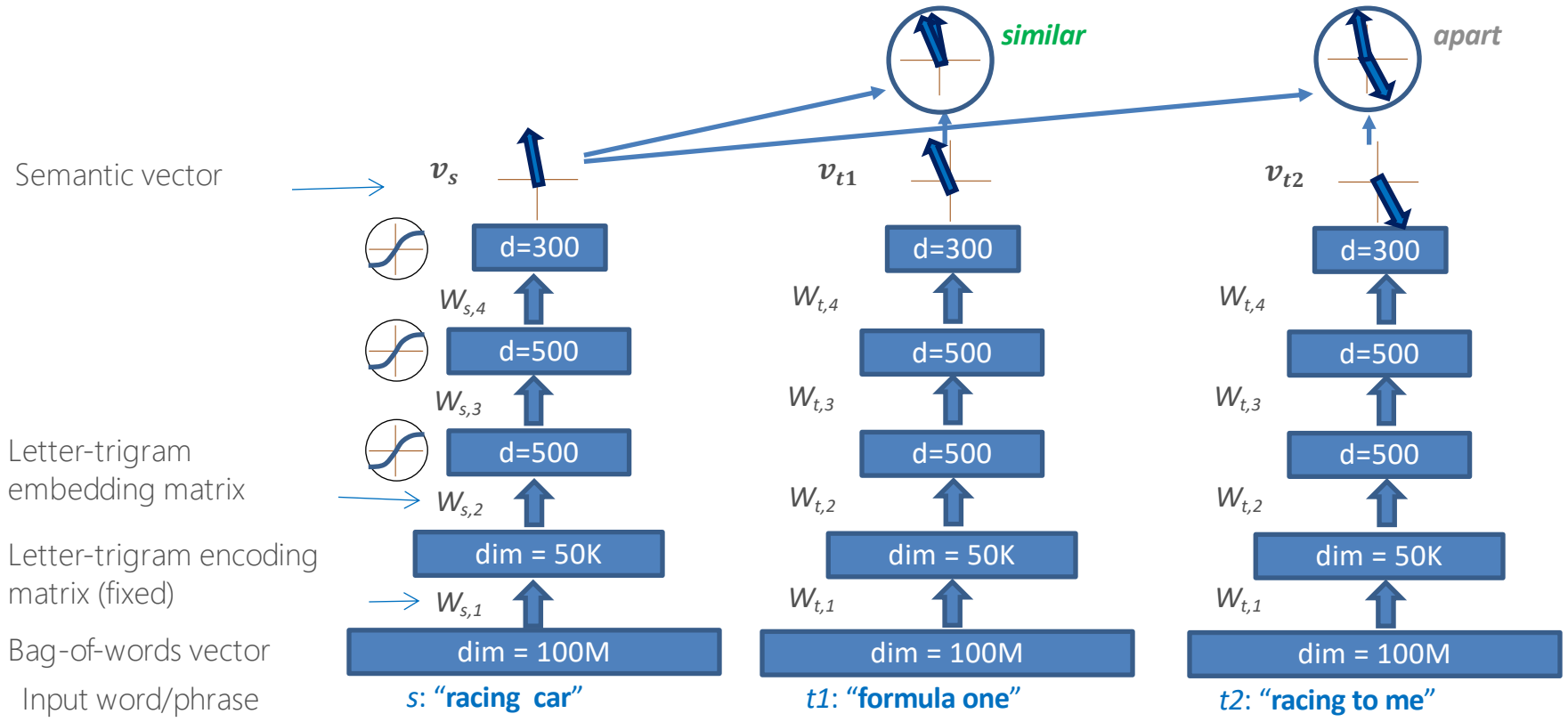


Consider a query s and two docs t^+ and t^-

- Assume t^+ is more relevant than t^- with respect to s
- $\cos(t^+, t^-)$ is the cosine similarity of t^+ and t^- in semantic space, mapped by DSSM parameters

DSSM for learning semantic embedding

Runtime:



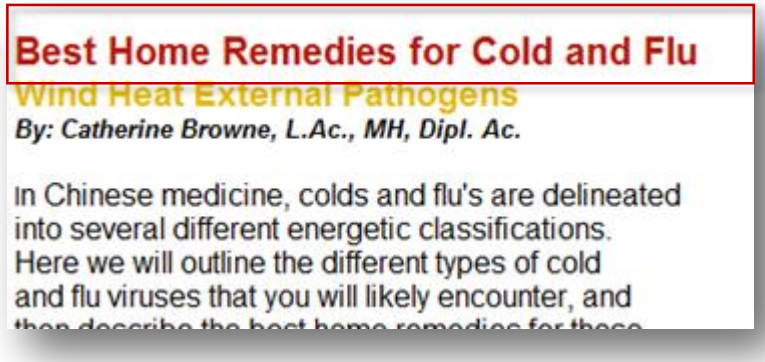
Huang PS, He X, Gao J, Deng L, Acero A, Heck L. Learning deep structured semantic models for web search using clickthrough data. InProceedings of the 22nd ACM international conference on Conference on information & knowledge management 2013 Oct 27 (pp. 2333-2338). ACM

Mining labeled (X, Y) pair from search logs

how to deal with stuffy nose?

stuffy nose treatment

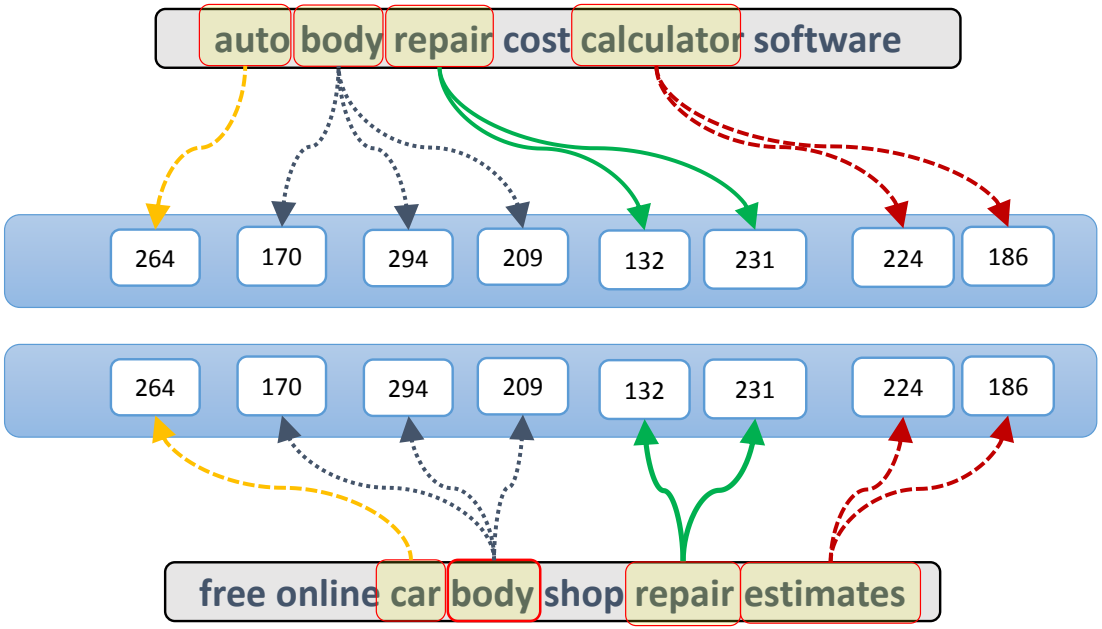
cold home remedies



QUERY (Q)	Title (T)
how to deal with stuffy nose	best home remedies for cold and flu
stuffy nose treatment	best home remedies for cold and flu
cold home remedies	best home remedies for cold and flu
… …	… …
go israel	forums goisrael community
skate at wholesale at pr	wholesale skates southeastern skate supply
breastfeeding nursing blister baby	clogged milk ducts babycenter
thank you teacher song	lyrics for teaching educational children s music
immigration canada lacolle	cbsa office detailed information

Huang PS, He X, Gao J, Deng L, Acero A, Heck L. Learning deep structured semantic models for web search using clickthrough data. InProceedings of the 22nd ACM international conference on Conference on information & knowledge management 2013 Oct 27 (pp. 2333-2338). ACM

Semantic matching of query and document



Most active neurons at the **max-pooling layers** of the query and document nets, respectively

Many applications of DSSM :

Learning semantic similarity between *X* and *Y*

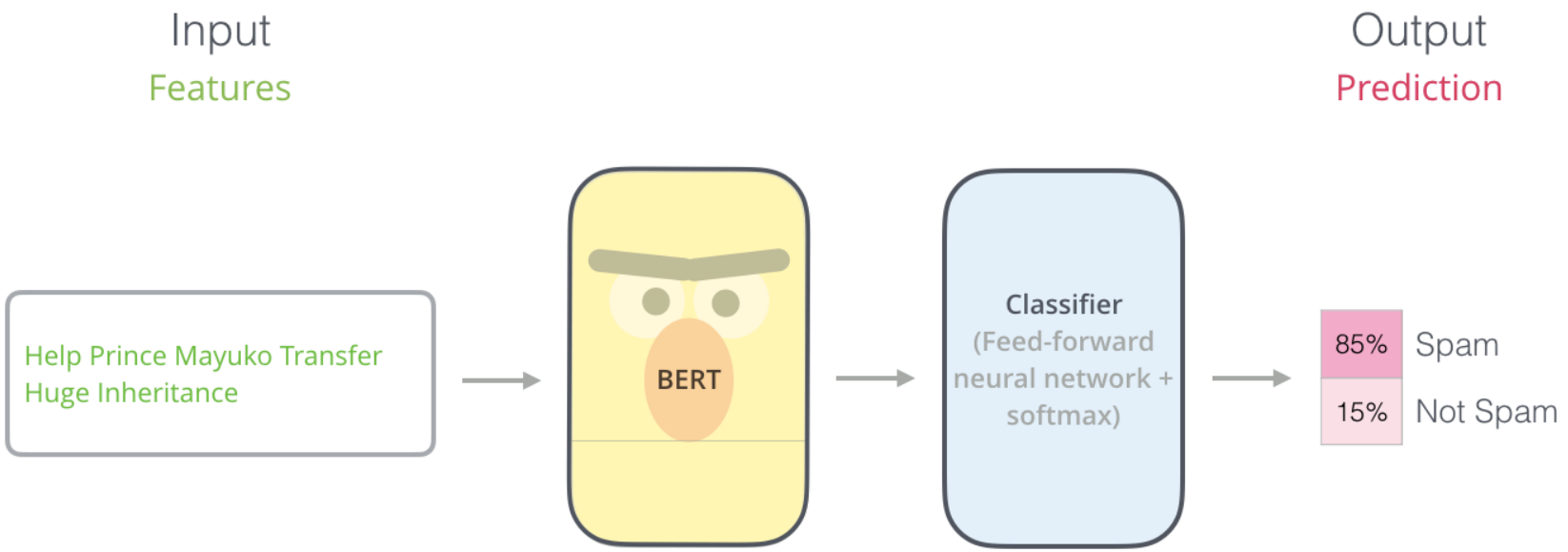
Tasks	*Source X*	*Target Y*
Word semantic embedding	*context*	*word*
Web search	*search query*	*web documents*
Query intent detection	*Search query*	*Use intent*
Question answering	*pattern / mention* (in NL)	*relation / entity* (in KB)
Machine translation	*sentence in language a*	*translated sentences in language b*
Query auto-suggestion	*Search query*	*Suggested query*
Query auto-completion	*Partial search query*	*Completed query*
Apps recommendation	*User profile*	*recommended Apps*
Distillation of survey feedbacks	*Feedbacks in text*	*Relevant feedbacks*
Automatic image captioning	*image*	*text caption*
Image retrieval	*text query*	*images*
Natural user interface	*command (text / speech / gesture)*	*actions*
Ads selection	*search query*	*ad keywords*
Ads click prediction	*search query*	*ad documents*
Email analysis: people prediction	*Email content*	*Recipients, senders*
Email search	*Search query*	*Email content*
Email declutering	*Email contents*	*Email contents in similar threads*
Knowledge-base construction	*entity from source*	*entity fitting desired relationship*
Contextual entity search	*key phrase / context*	*entity / its corresponding page*
Automatic highlighting	*documents in reading*	*key phrases to be highlighted*
Text summarization	*long text*	*summarized short text*

Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

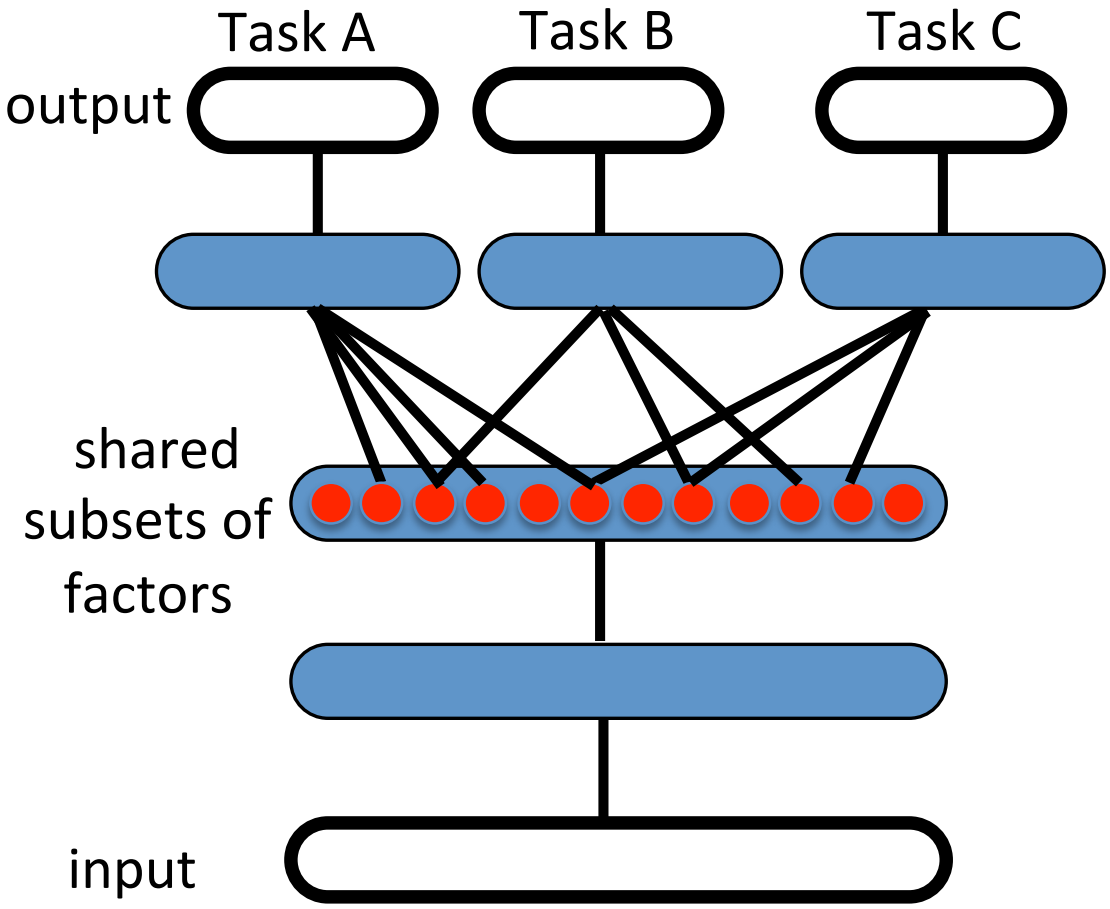
Language models + downstream tasks

- Sentence classifications



https://jalammar.github.io/illustrated-bert/

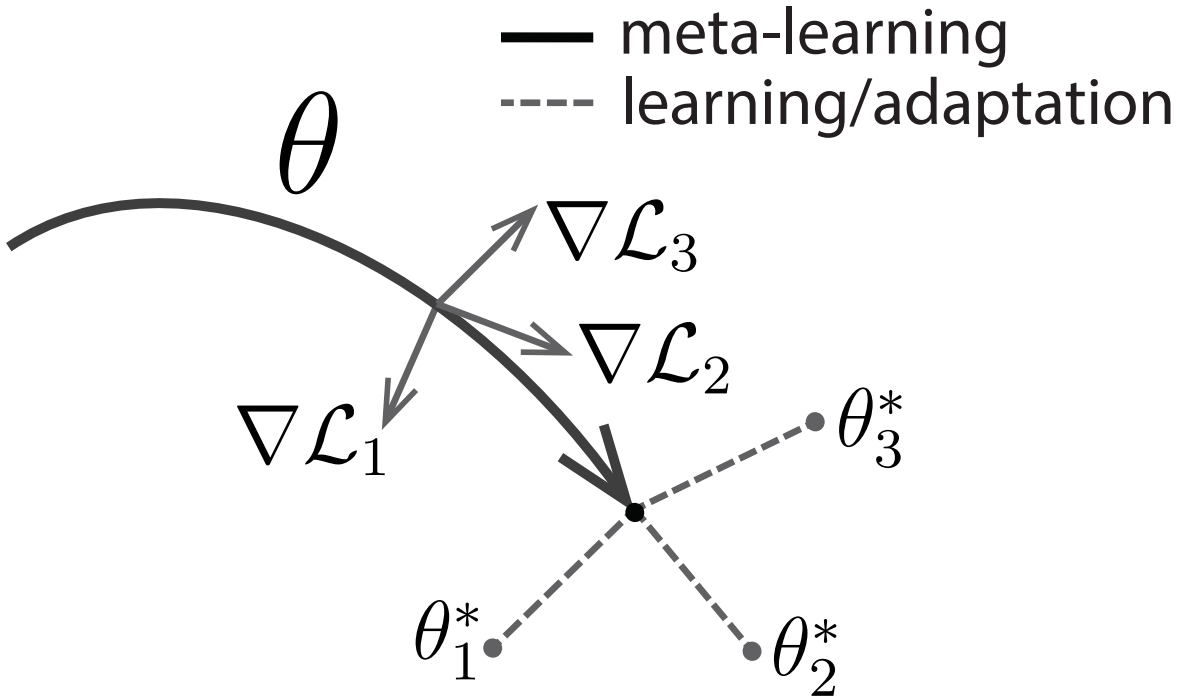
Representation learning



Bengio, Yoshua, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013): 1798-1828.

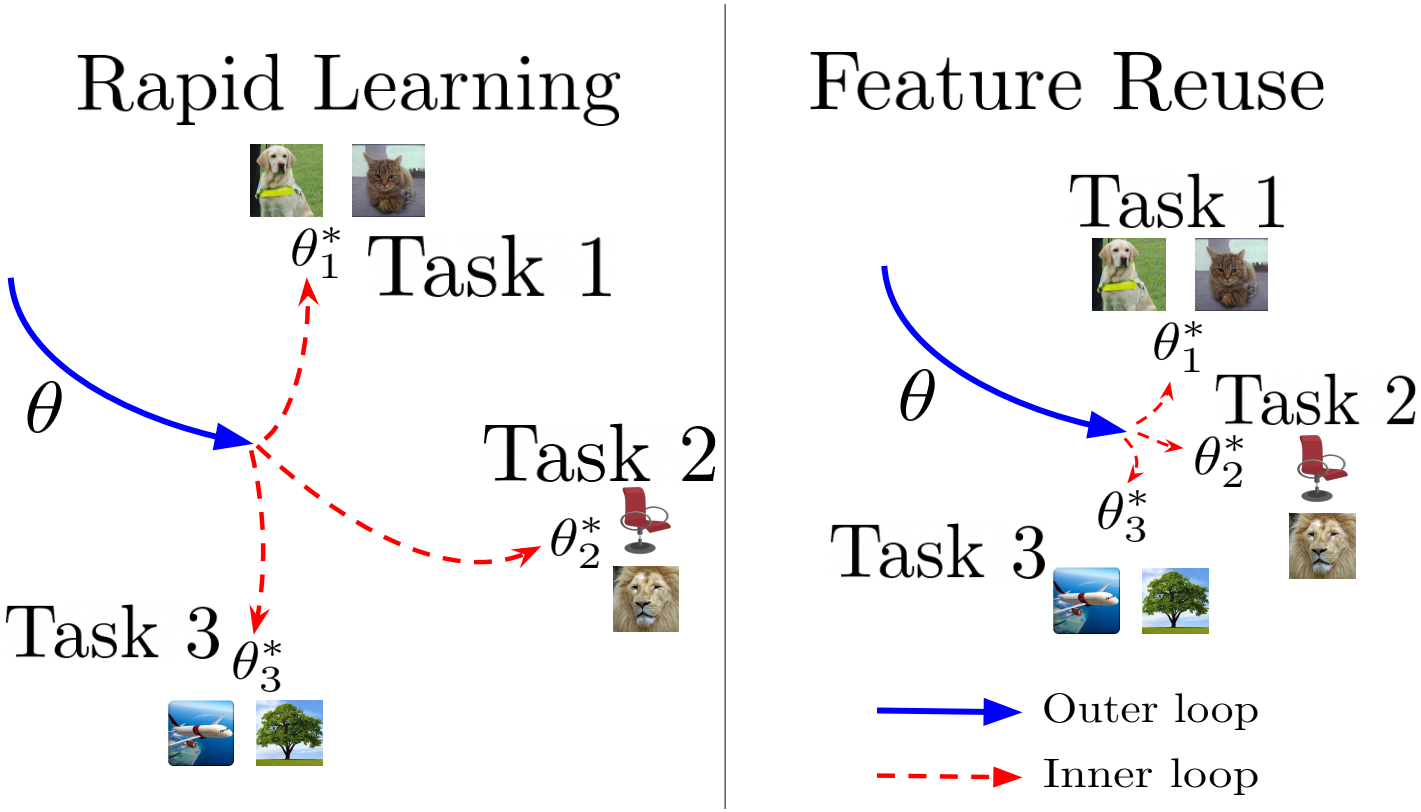
Model-agnostic meta-learning

- The idea is to learn an initialisation of NN parameters and can be quickly adaptive to new tasks with few training examples



Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." *International Conference on Machine Learning*. PMLR, 2017

RAPID LEARNING OR FEATURE REUSE?

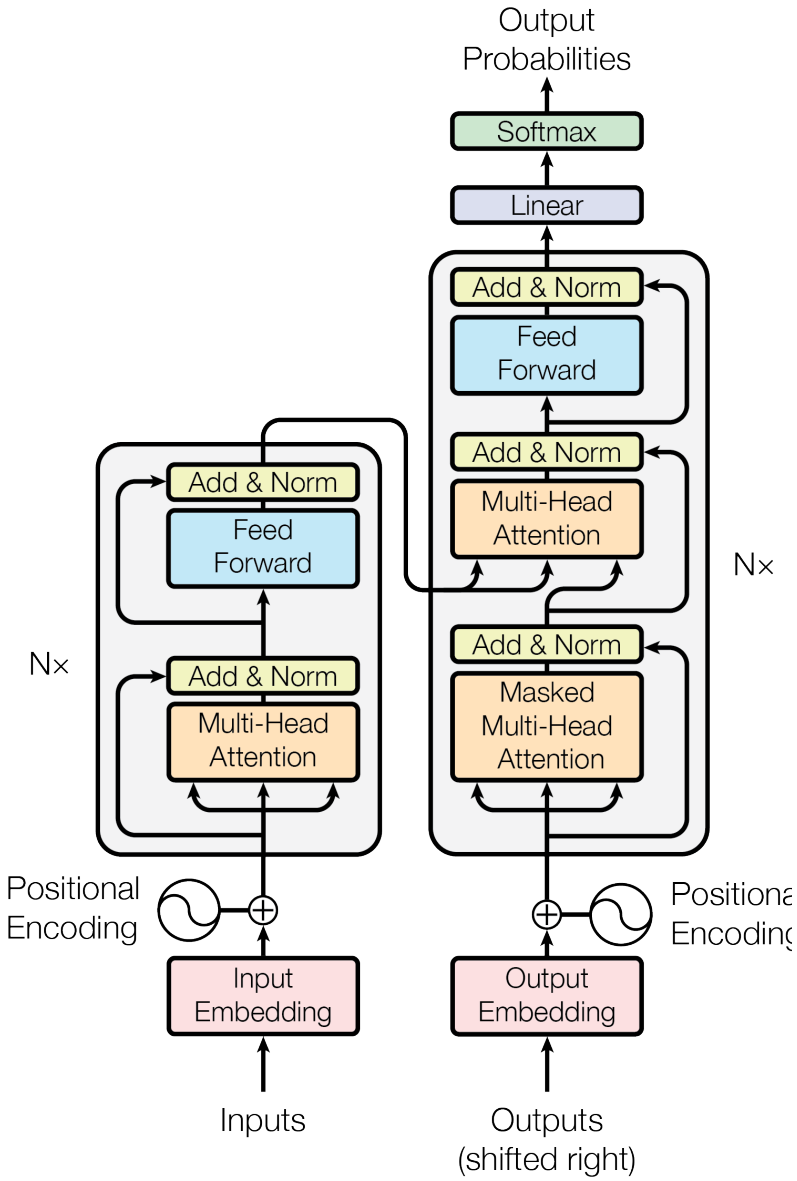


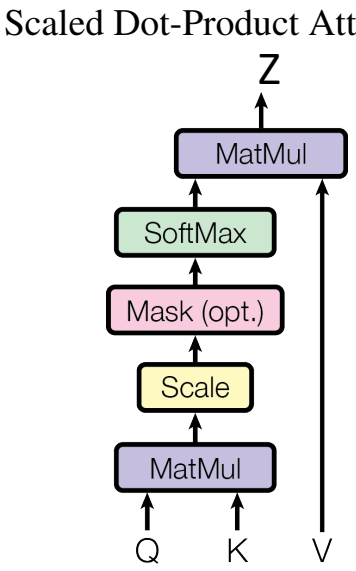
In Rapid Learning, outer loop training leads to a parameter setting that is well-conditioned for fast learning, and inner loop updates result in significant task specialization.

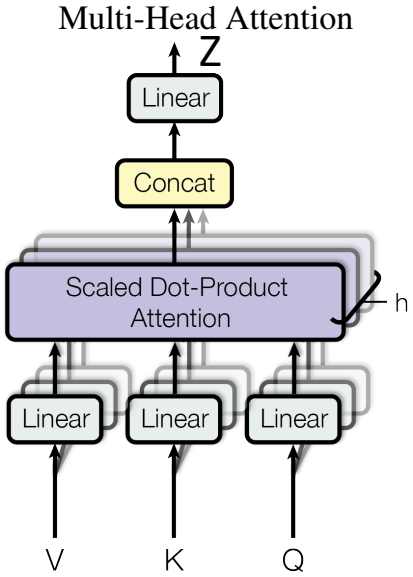
In Feature Reuse, the outer loop leads to parameter values corresponding to reusable features (representation), from which the parameters do not move significantly in the inner loop.

Raghu, Aniruddh, et al. "Rapid learning or feature reuse? towards understanding the effectiveness of maml." *arXiv preprint arXiv:1909.09157* (2019).

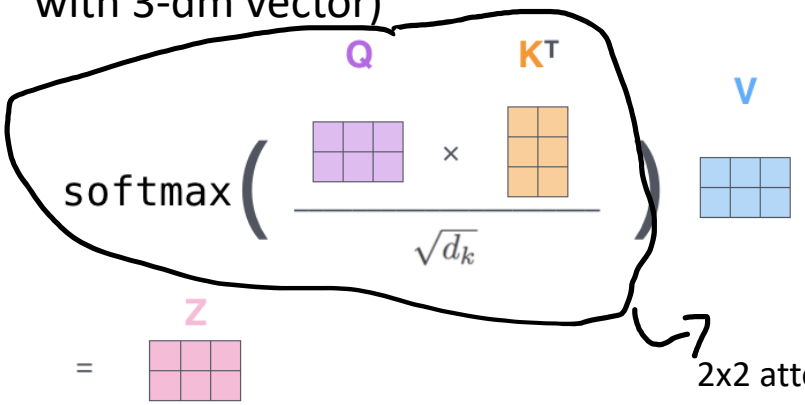
Transformers







Self-attention in matrix form (two words with 3-dm vector)

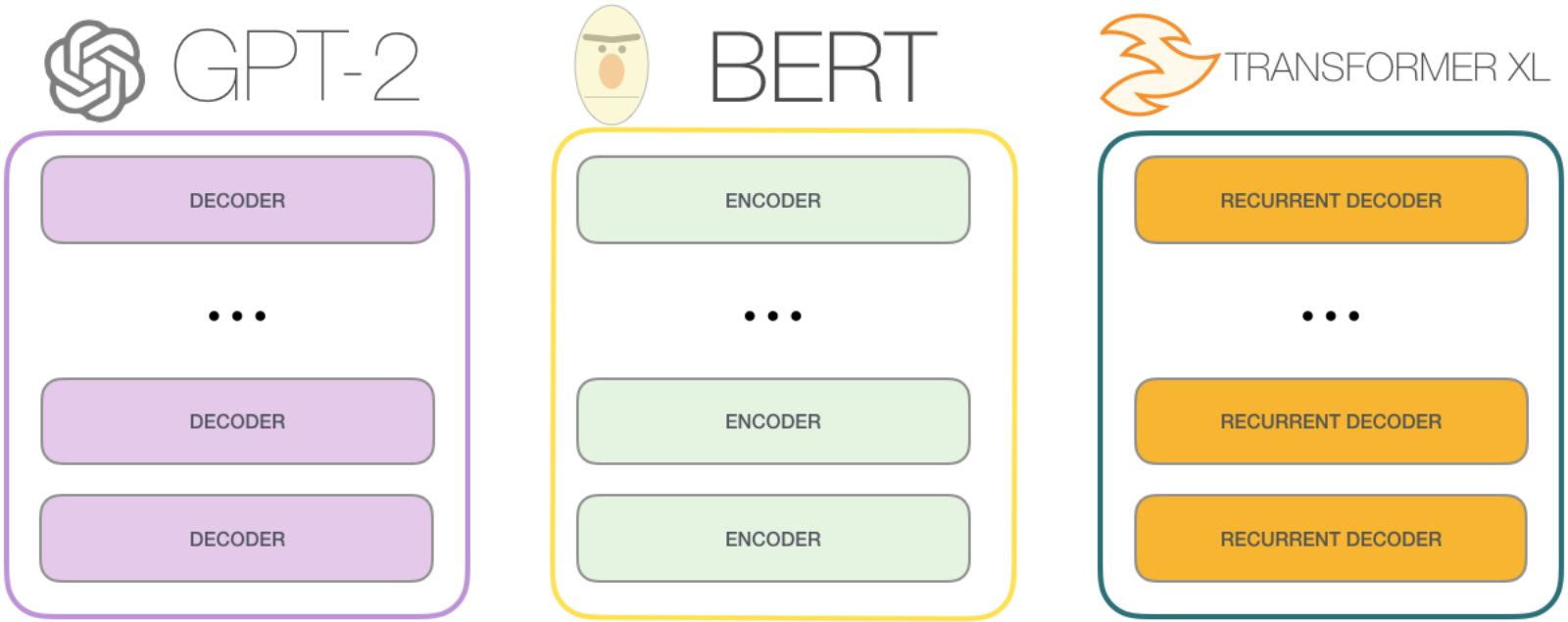


2x2 attention as weights for weighted sum of V

Vaswani, Ashish, et al. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

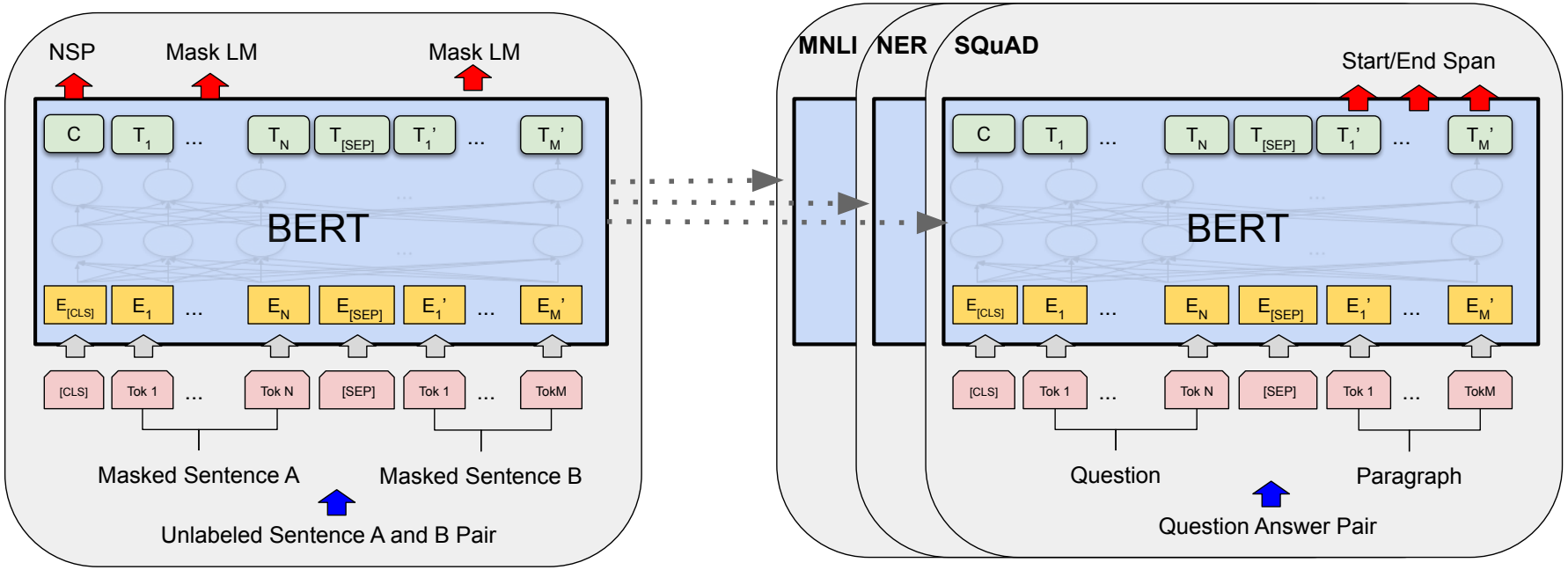
https://jalammar.github.io/illustrated-transformer/

Pre-trained models



https://jalammar.github.io/illustrated-gpt2/

BERT (Bidirectional Encoder Representations from Transformers)



Pre-training

Using the loss from predicting marked tokens (Mask LM) and just it is the corrected next sentence (NSP)

Fine-Tuning

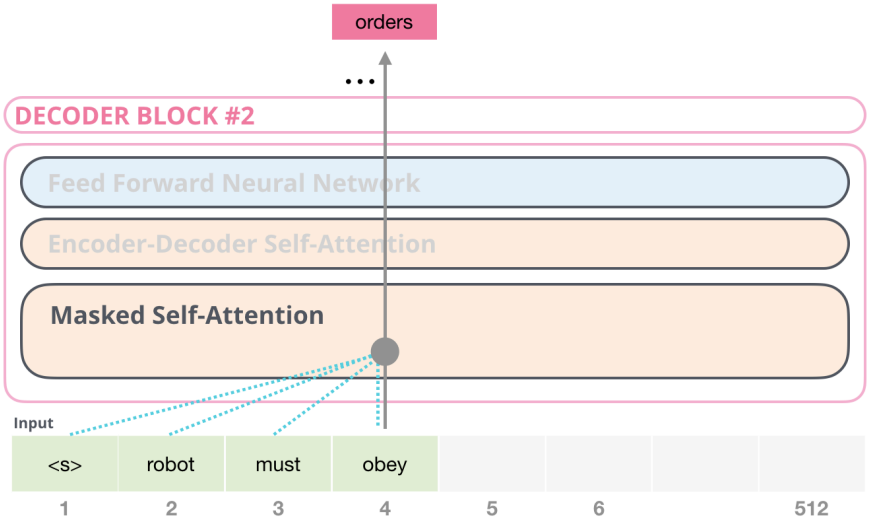
All kind of language related downstream tasks

Apart from output layers, the same architectures are used in both pre-training and fine-tuning. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

GPT (Generative Pre-trained Transformer) 3

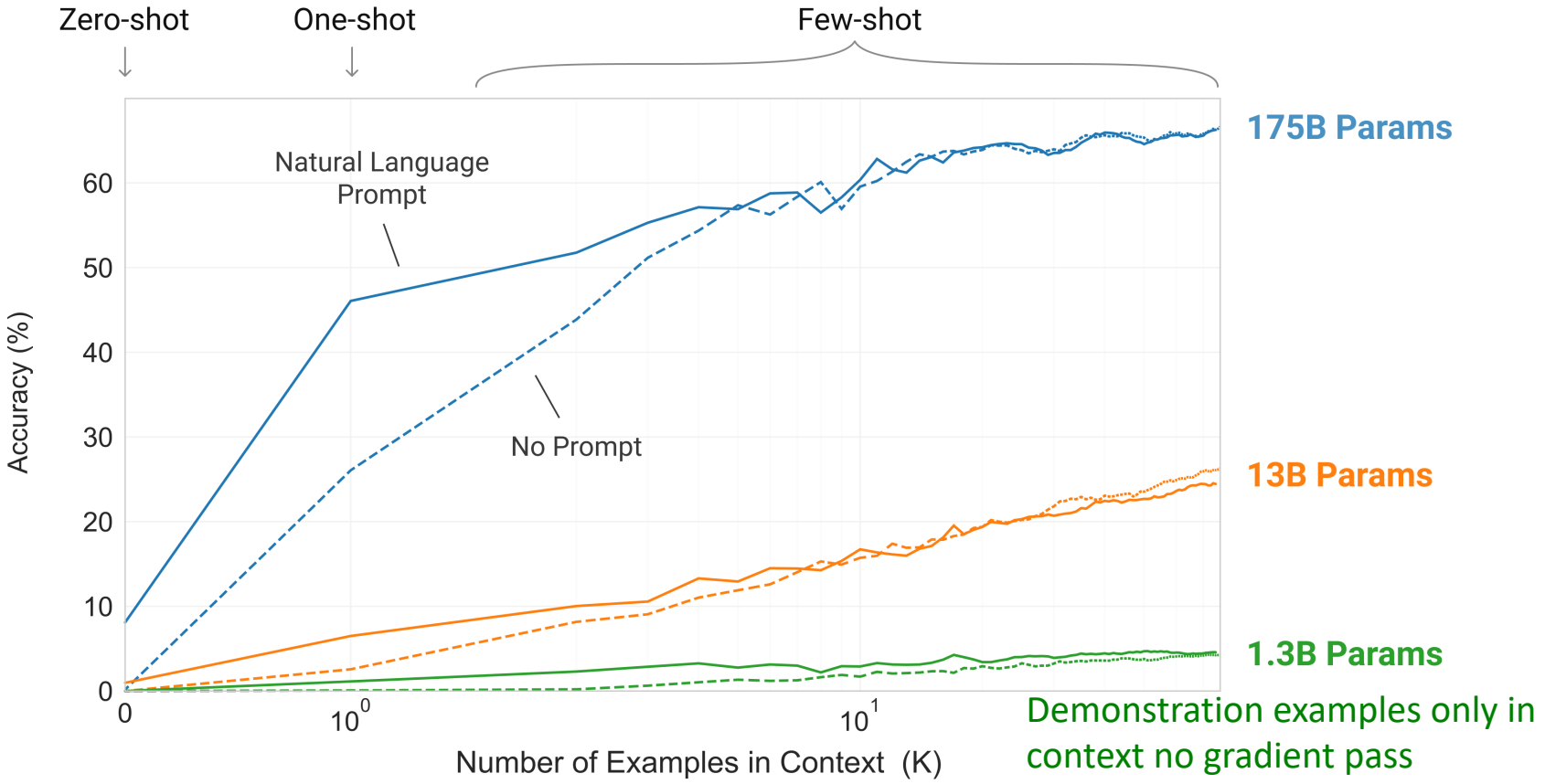
- a large-scale transformer-decoder based autoregressive language model
- which is trained on 175 billion parameters and is 10x more than previous models.



Brown, Tom B., et al. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020).

https://jalammar.github.io/illustrated-gpt2/

GPT (Generative Pre-trained Transformer) 3



Context learning : remove random symbols from a word, both with and without a natural language task description

large models demonstrate improved ability to learn a task from contextual information.

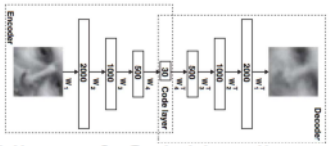
Brown, Tom B., et al. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020).

https://jalammar.github.io/illustrated-gpt2/

Self-supervised learning

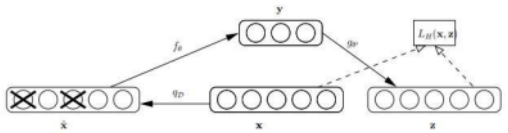
- For computer vision, how to build up pre-training tasks and their labels?

Autoencoders



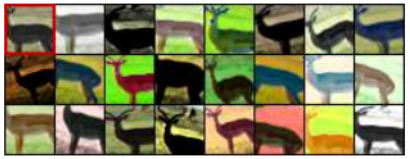
Hinton & Salakhutdinov. Science 2006.

Denoising Autoencoders



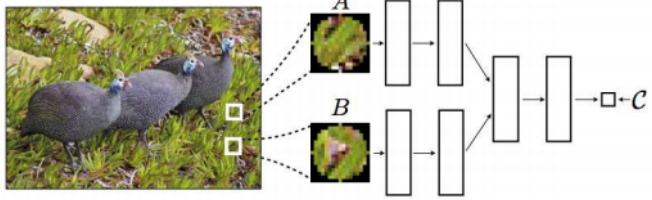
Vincent *et al.* ICML 2008.

Exemplar networks



Dosovitskiy *et al.*, NIPS 2014

Co-Occurrence



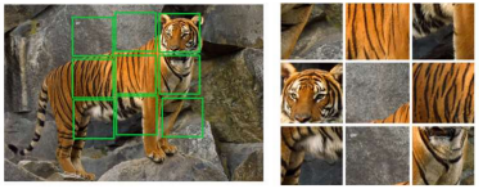
Isola *et al.* ICLR Workshop 2016.

Egomotion

the problem of predicting camera transformation from image pairs

Agrawal *et al.* ICCV 2015 Jayaraman *et al.* ICCV 2015

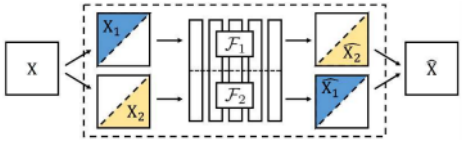
Context



Noroozi et al 2016

Pathak *et al.* CVPR 2016

Split-brain auto-encoders

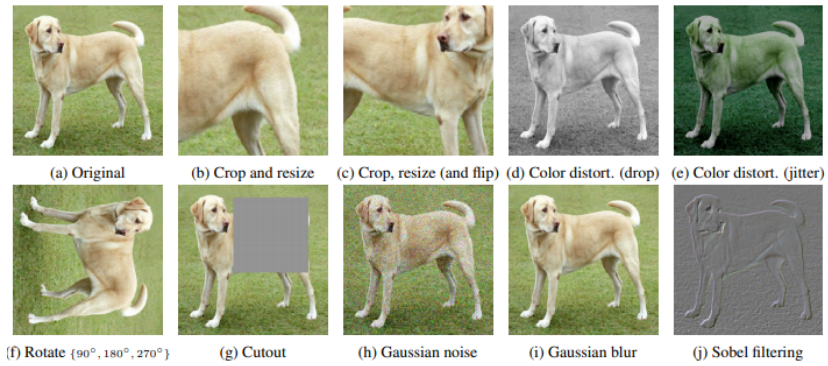


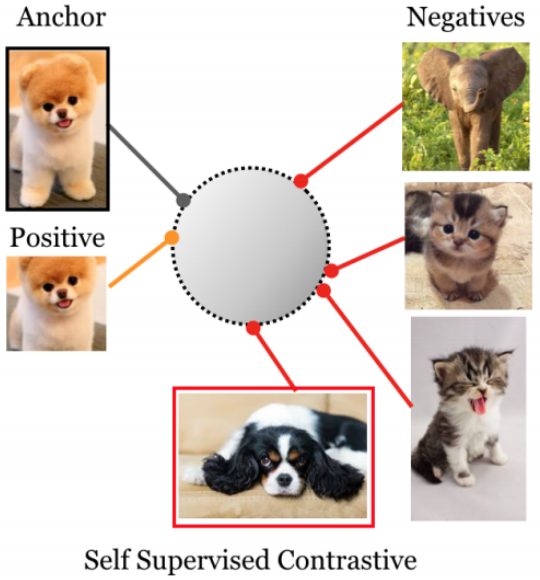
Zhang *et al.* CVPR 2017

https://project.inria.fr/paiss/files/2018/07/zisserman-self-supervised.pdf

Representation by Contrastive Learning

- Ideas:
  - Some unlabeled images, data augmentation methods, want to train a representation network
  - Treat augumented images from the same image as positive pairs, from different images as negative pairs
    - Positive pairs have similar representations
    - Negative pairs have different representations

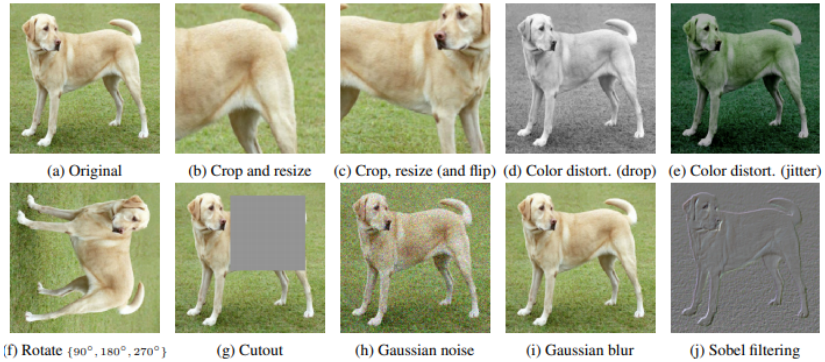


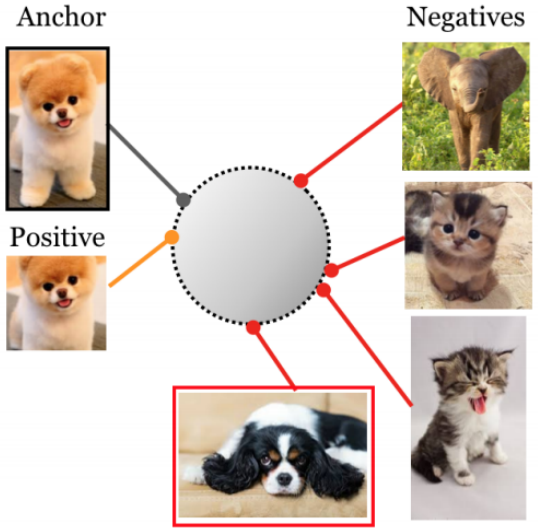


https://arxiv.org/pdf/2004.11362.pdf;
https://arxiv.org/pdf/2002.05709.pdf

Representation by Contrastive Learning

- Ideas:
  - Some unlabeled images, data augmentation methods, want to train a representation network
  - Treat augumented images from the same image as positive pairs, from different images as negative pairs
    - Positive pairs have similar representations
    - Negative pairs have different representations





Self Supervised Contrastive

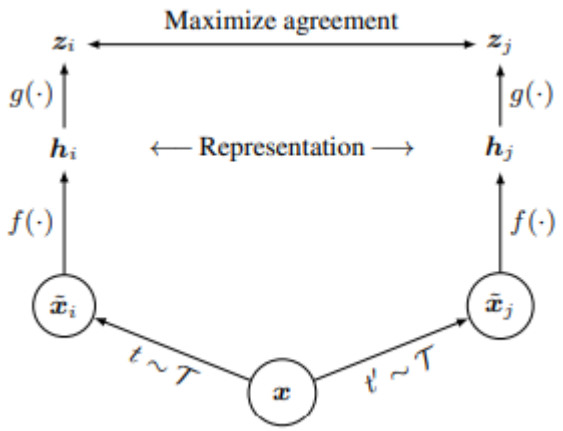
$$I(X;Y) \geq \mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\log\frac{e^{f(x_i,y_i)}}{\frac{1}{K}\sum_{j=1}^{K}e^{f(x_i,y_j)}}\right] \triangleq I_{\mathrm{NCE}}(X;Y),$$

https://arxiv.org/pdf/2004.11362.pdf;
https://arxiv.org/pdf/2002.05709.pdf

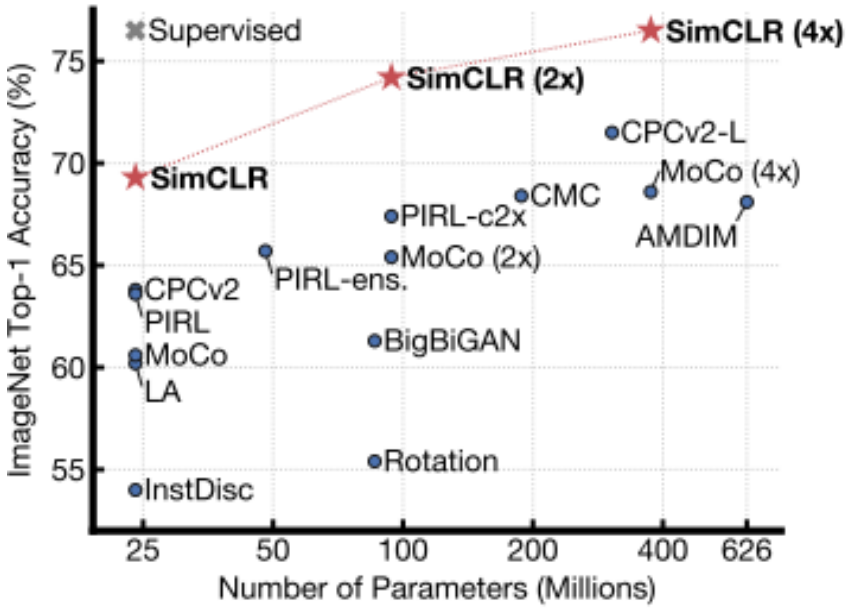
Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." *arXiv preprint arXiv:1807.03748* (2018).

Representation by Contrastive Learning

- Two separate data augmentation operators are sampled from the same family of augmentations



$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$



ImageNet Top-1 accuracy of linear classifiers trained on representations

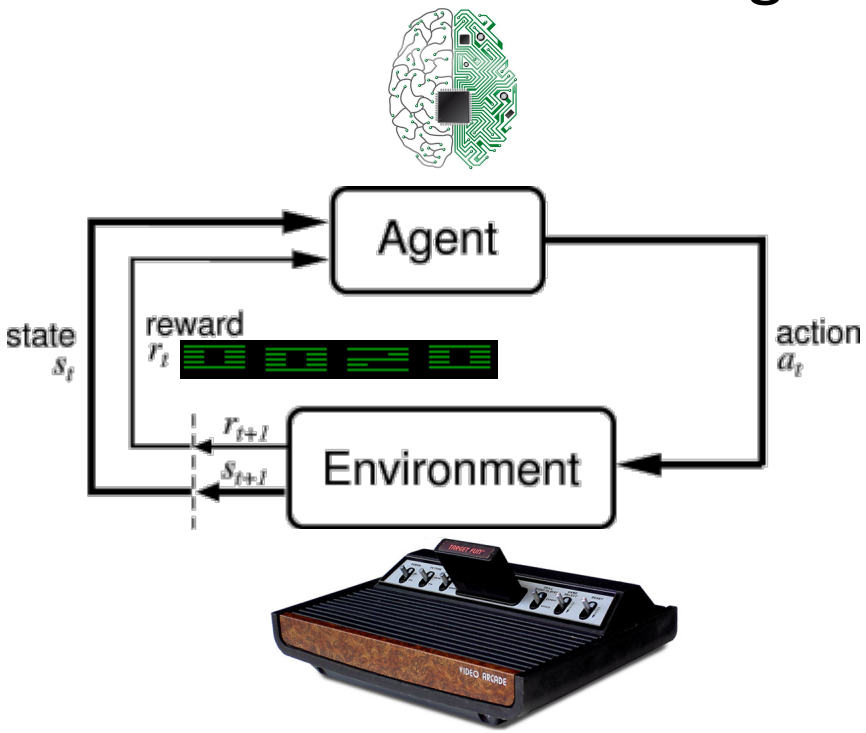
Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020

Summay

- Neural networks basics
- Word embedding
- Go deeper in layers
- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)
- Web search revisited
- Representation learning
- Deep reinforcement learning

Reinforcement learning

- Computerised agent: Learning what to do
  - How to map situations (states) to actions so as to maximise a numerical reward signal



Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 1998.

Reinforcement learning

- {state *s*, action *a*, reward *r*} are made in stages

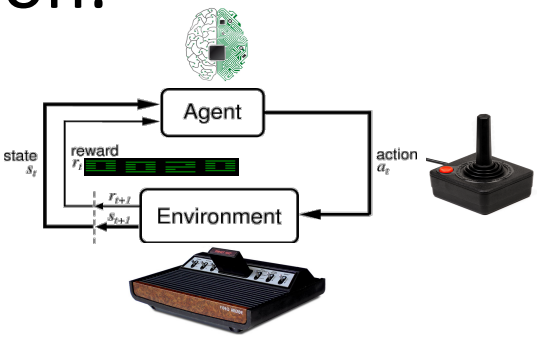
$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

The outcome of each decision is not fully predictable but can be observed before the next decision is made

- The objective is
  - to maximize a numerical measure of total reward

$$R = r_1 + r_2 + r_3 + \ldots + r_n$$

- Decisions cannot be viewed in isolation:
  - need to balance desire for immediate reward with possibility of high future reward



Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 1998.

Q-learning

- For game, the future *discounted* reward at time *t*

$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} \ldots,$ where γ is discount factor (aka prob. of terminating)

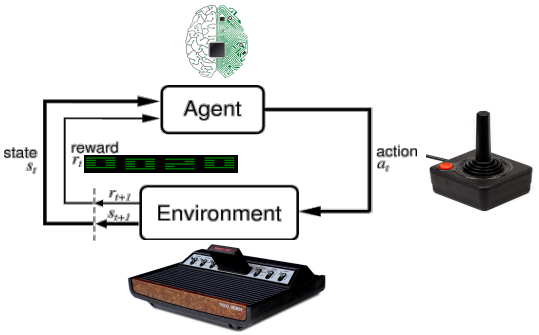
- We define the optimal action-value function

$Q^*(s,a) = \max_\pi [r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \ldots \mid s_t = s, a_t = a, \pi],$ where $\pi = p(a \mid s)$ is a policy

- *Bellman* equation

$Q^*(s,a) = E_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q^*(s,a) \mid s_t = s, a_t = a],$

where ε is sample distribution from an emulator



- So iteratively update Q* as above

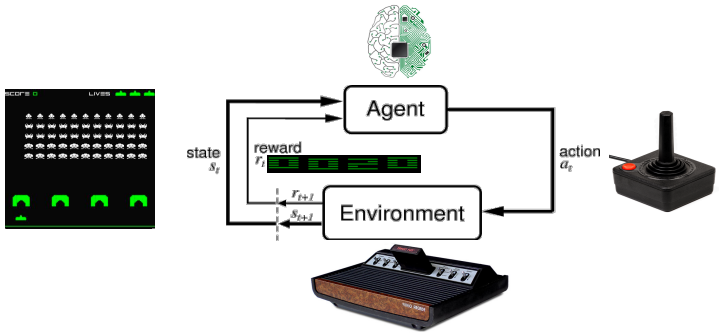
and $a_t^* = \max_a Q^*(s_t, a_t)$

Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 1998.

A simplified Q-learning algorithm

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

Maintain a Q table and keep updating it



http://www.nervanasys.com/demystifying-deep-reinforcement-learning/

Deep Q-learning

- However, in many cases, it is not practical as
  - *Action x State* space is very large
  - lacking generalisation
- *Q* function is commonly approximated by a function, either linear or non-linear

$$Q(s,a;\theta) \approx Q^*(s,a) \quad \text{where } \theta \text{ is model parameter}$$

- It can be trained by minimising a sequence of loss function

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

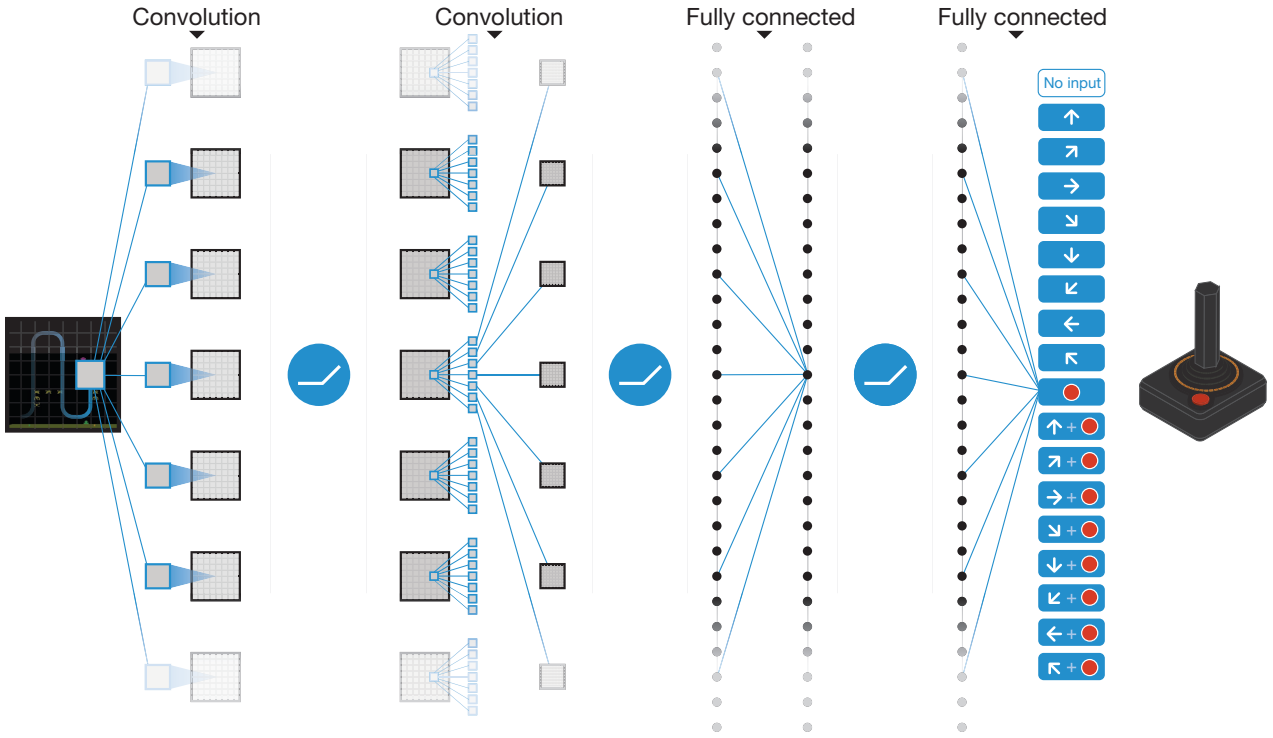
where $U(D)$ is sample distribution from a pool of stored samples and θ_i^- only updates every C steps

target

Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

Deep Q-learning

Deep Q-network (DQN): Input is 4 recent images (84x84) as state s and output is the actions of the game control (respond to its Q value)



The beauty of this is that it learns to play Atari games just from the raw pixel inputs.

The First convolution: 32 filters of 8 x 8 with stride 4
The Second convolution: 64 filters of 3 x 3 with stride 1
The third convolution: 64 filters of 4 x 4 with stride 2

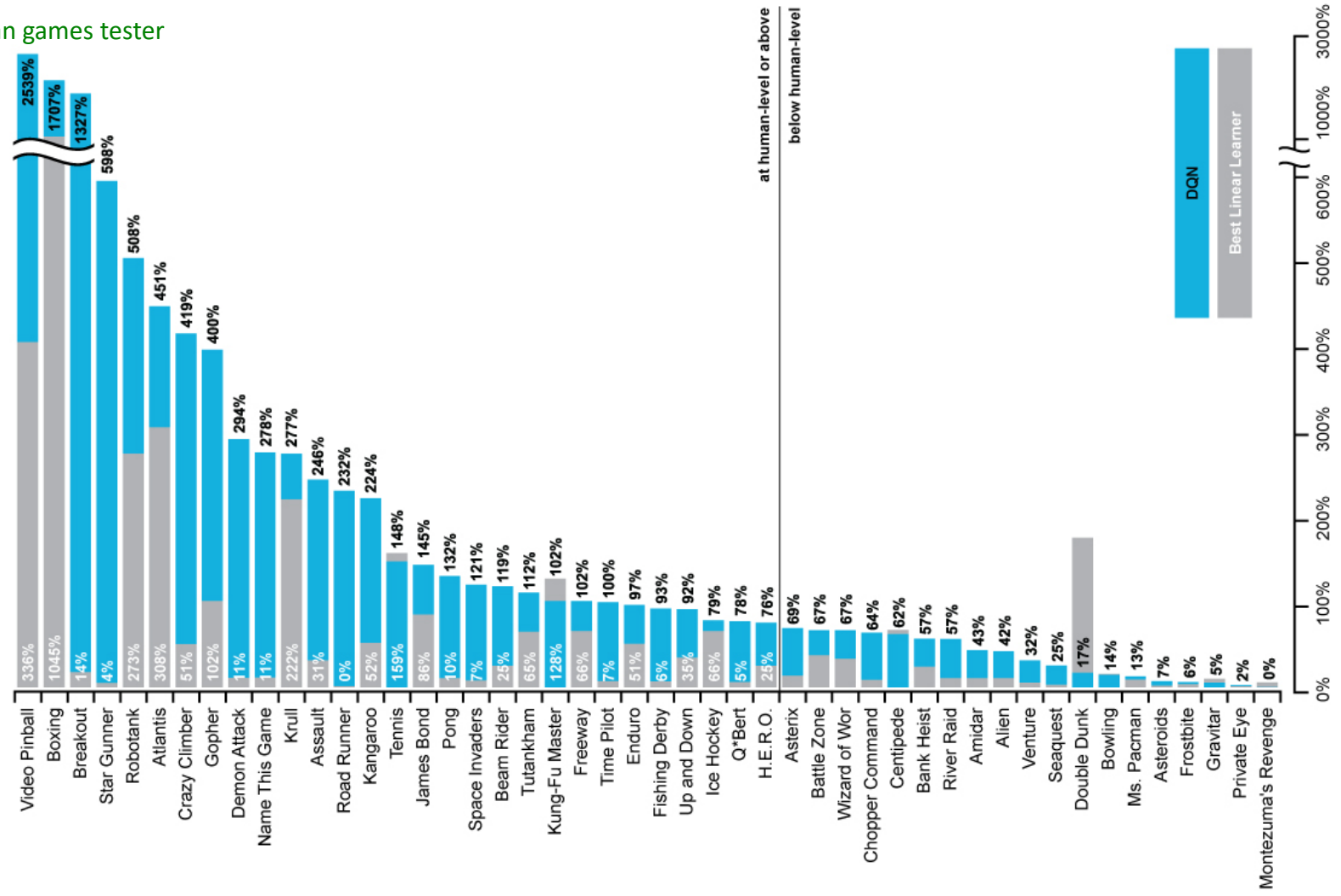
Active function: rectifier nonlinearity

Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.

DQN results on atari games

The performance of DQN is normalized with respect to a professional human games tester



Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

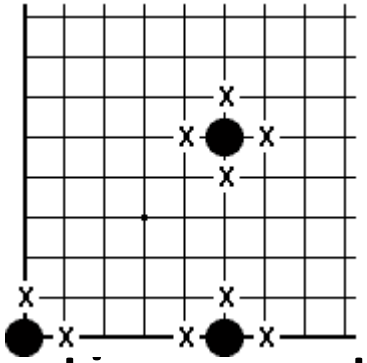
The Go Game

- Go (Weiqi) is an ancient game originated from China, with a history over 3000 years

The board: 19 x 19 lines, making 361 intersections (`points').
The move: starting from Black, stones are placed on points in alternation until the end
The goal: control the max. territory

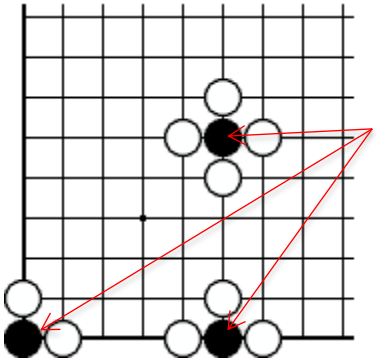
The Go Game: liberties

- Liberties: the unoccupied intersections (or points) that are horizontally or vertically adjacent to the stone



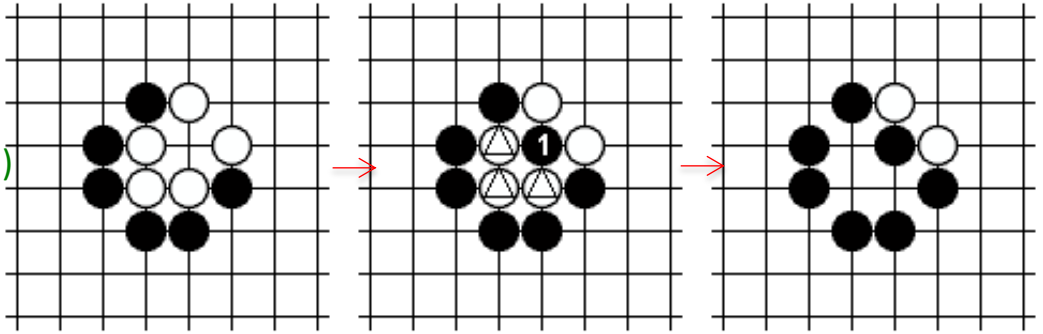
Liberties are marked as X

- Stones without liberties must be removed



Need to be removed (thus captured)

Black captures 3 stones from White



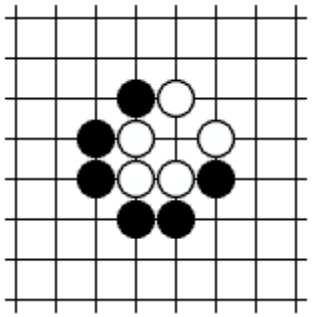
One of the great challenges of AI

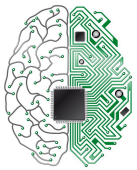
- It is far more complex than chess
  - Chess has 35^{80} possible moves
  - Go has 250^{150} possible moves
- Computers match or surpass top humans in *chess*, *backgammon*, *poker*, even *Jeopardy*
- But not Go game as 2016

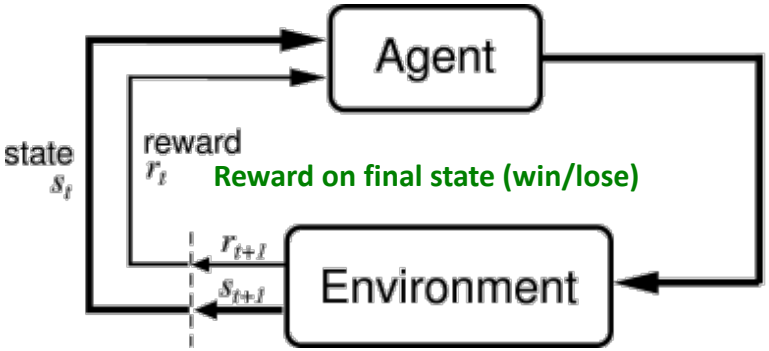
https://jedionston.files.wordpress.com/2015/02/go-game-storm-trooper.jpg

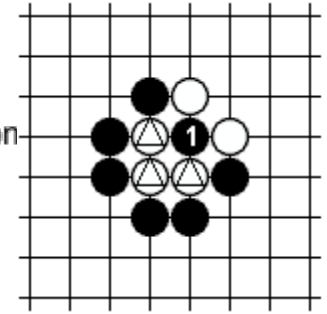
AlphaGo

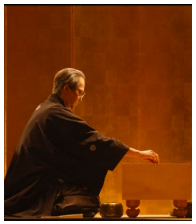
- Yet another example of deep reinforcement learning









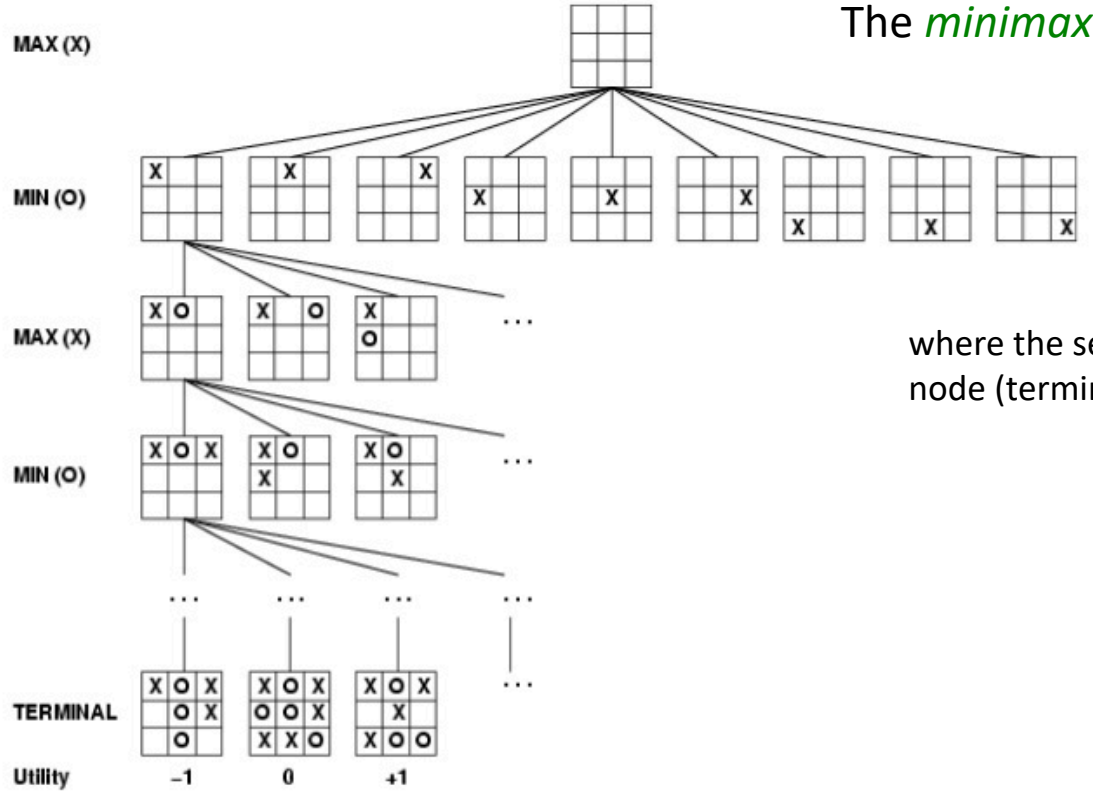


- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Known transition model: $p(s, a, s')$
- Reward on final states: win or lose

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

Minimax search

- Tic-tac-toe zero-sum two-player game



The *minimax* algorithm is recursively as

$$a_i^* = \text{argmax}_{a_i \in N_t} (\min_{a_{-i} \in N_{t+1}} u(a_{-i}, a_i))$$

a_i : action from you at step i

a_{-i} : action from opponent at step i

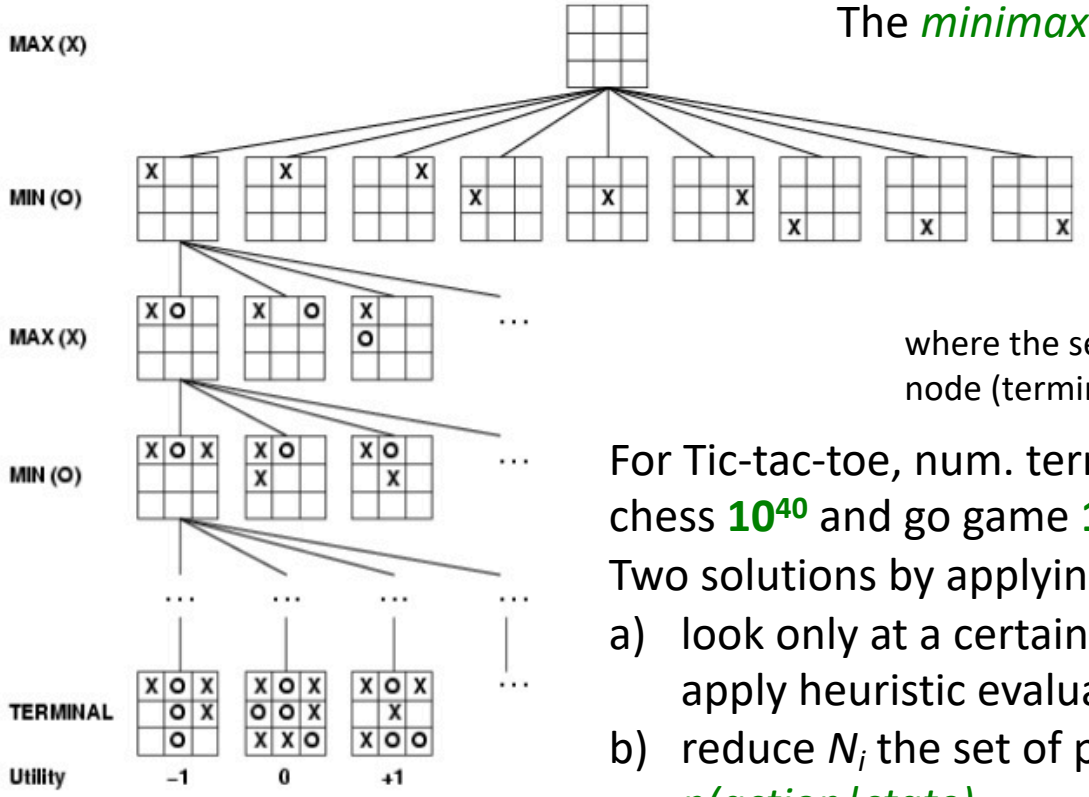
where the set N_{t+1} are the successors of node N_t. If N_t is a leaf node (terminal node), it is value of the node (for you)

Neumann, L. J., & Morgenstern, O. (1947). Theory of games and economic behavior (Vol. 60). Princeton: Princeton university press.

Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25 (1995): 27.

Minimax search

- Tic-tac-toe zero-sum two-player game



The *minimax* algorithm is recursively as

$$a_i^* = \text{argmax}_{a_i \in N_t} (\min_{a_{-i} \in N_{t+1}} u(a_{-i}, a_i))$$

a_i : action from you at step i

a_{-i} : action from opponent at step i

where the set N_{t+1} are the successors of node N_t. If N_t is a leaf node (terminal node), it is value of the node (for you)

For Tic-tac-toe, num. terminal nodes 9! = 362,880, but for chess **10^{40}** and go game **10^{170}**!

Two solutions by applying prior knowledge:

a) look only at a certain number of moves ahead (piles) and apply heuristic evaluation function *u*, e.g. IBM deep blue

b) reduce N_i the set of possible move (each round), i.e., *p(action|state)*

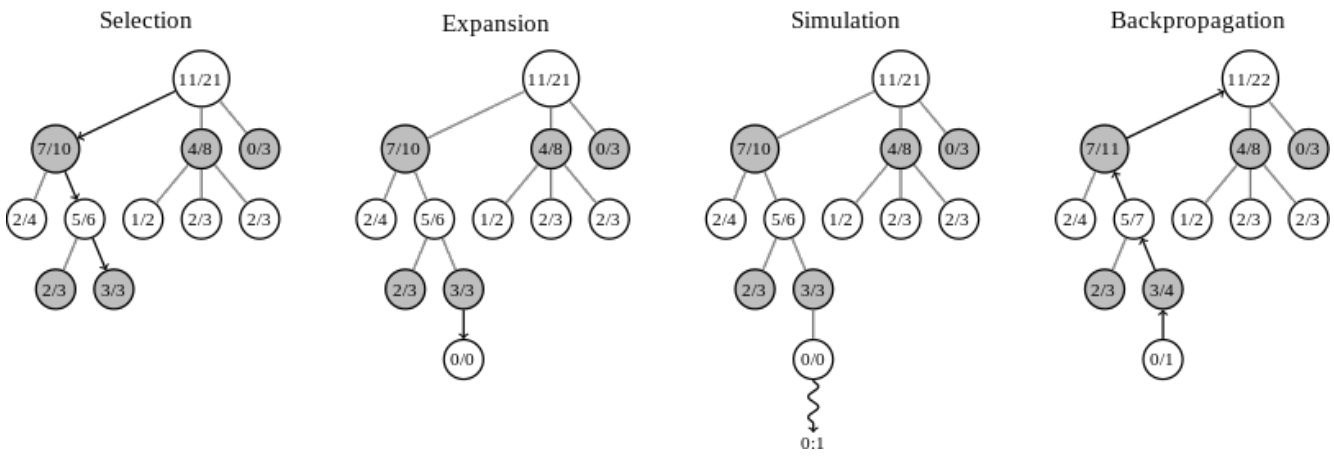
c) Random sampling *by* Monte-Carlo simulation

Neumann, L. J., & Morgenstern, O. (1947). Theory of games and economic behavior (Vol. 60). Princeton: Princeton university press.

Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25 (1995): 27.

Monte-Carlo tree search (MCTS)

Each tree node stores the number of won w(a) /played playouts n(a)



Repeatedly running the four steps until timeout

Then the move with the most simulations made is selected rather than the move with the highest average win rate

- Coined 2006, MCTR in real-time expands search tree by simulation and is further extended using the bandit solution for exploration

$$a_i^* = \text{argmax}_{a_i}(w(a) + \sqrt{\frac{2\ln(n)}{n(a)}}),$$

$w(a)$: num win for a

n: num. visits; n(a): num. visits for a

- In 2008, *MoGo* achieved dan (master) level in 9×9 go

Browne, Cameron B., et al. "A survey of monte carlo tree search methods." Computational Intelligence and AI in Games, IEEE Transactions on 4.1 (2012): 1-43.

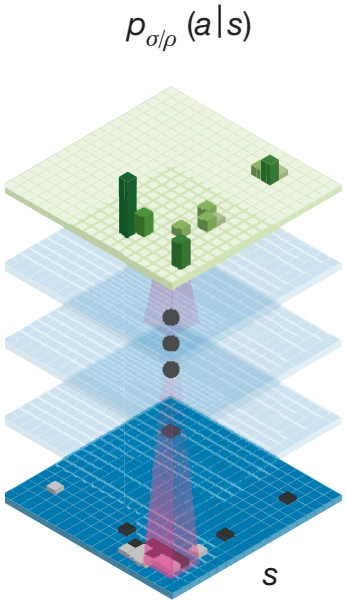
Kocsis, Levente, and Csaba Szepesvári. "Bandit based monte-carlo planning." Machine Learning: ECML 2006. Springer Berlin Heidelberg, 2006. 282-293.

Coulom, Rémi. "Efficient selectivity and backup operators in Monte-Carlo tree search." Computers and games. Springer Berlin Heidelberg, 2006. 72-83.

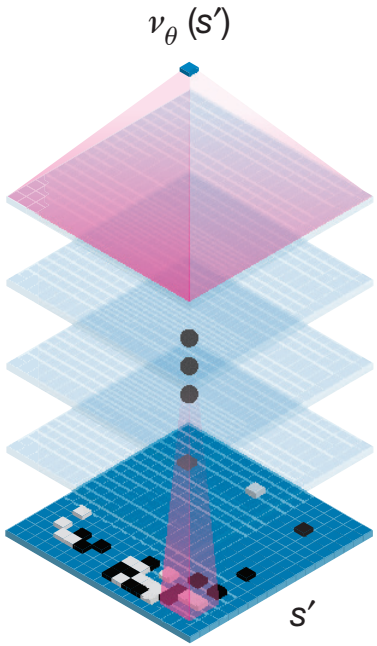
AlphaGo

- Employ two deep convolutional neural networks

Policy network



Value network



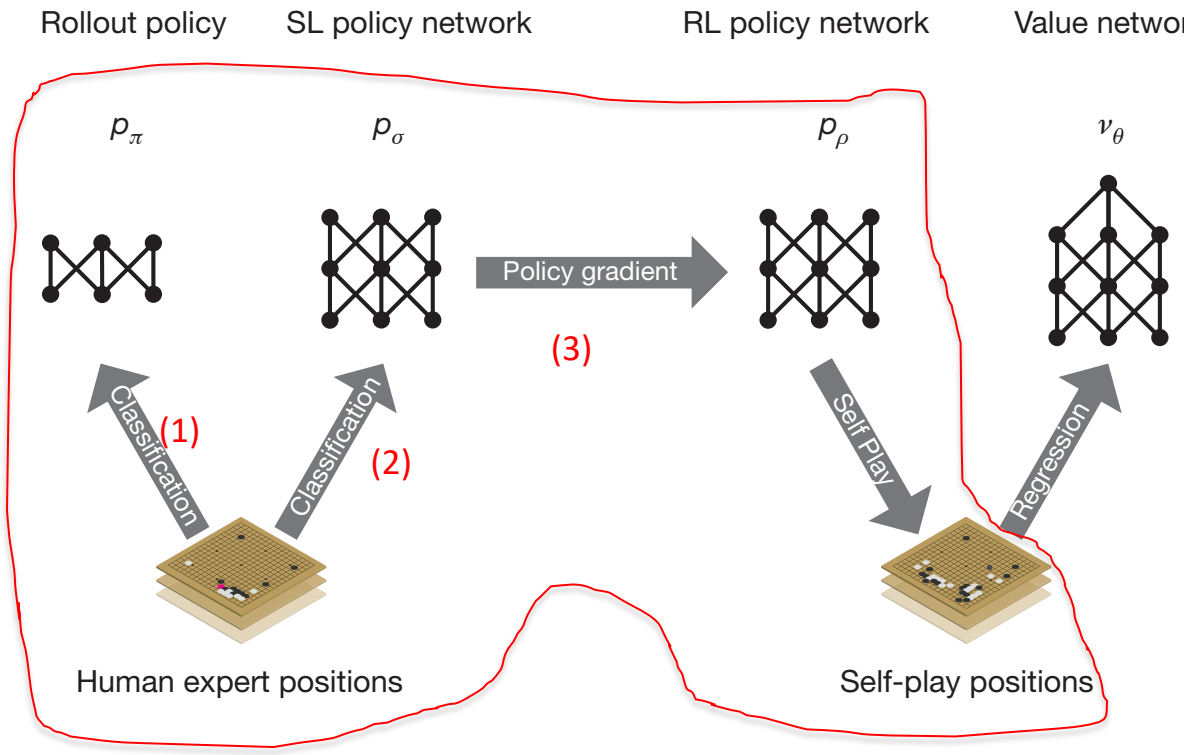
Policy network predicts p(action = a| state = s) of human moves

Value network predicts the expected terminal reward of a given state s

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

AlphaGo

- Training for policy network predicting human moves



Supervised trained (SL) from 30m positions from the KGS Go Server

(1) $p_\pi(a \mid s)$: fast prediction with linear softmax

(2) $p_\delta(a \mid s)$: more accurate prediction with 12-layer CNN. superversied learned from

(3) $p_\delta(a \mid s)$: further tuned by policy gradient from the terminal reward of self-play.

Gradient update

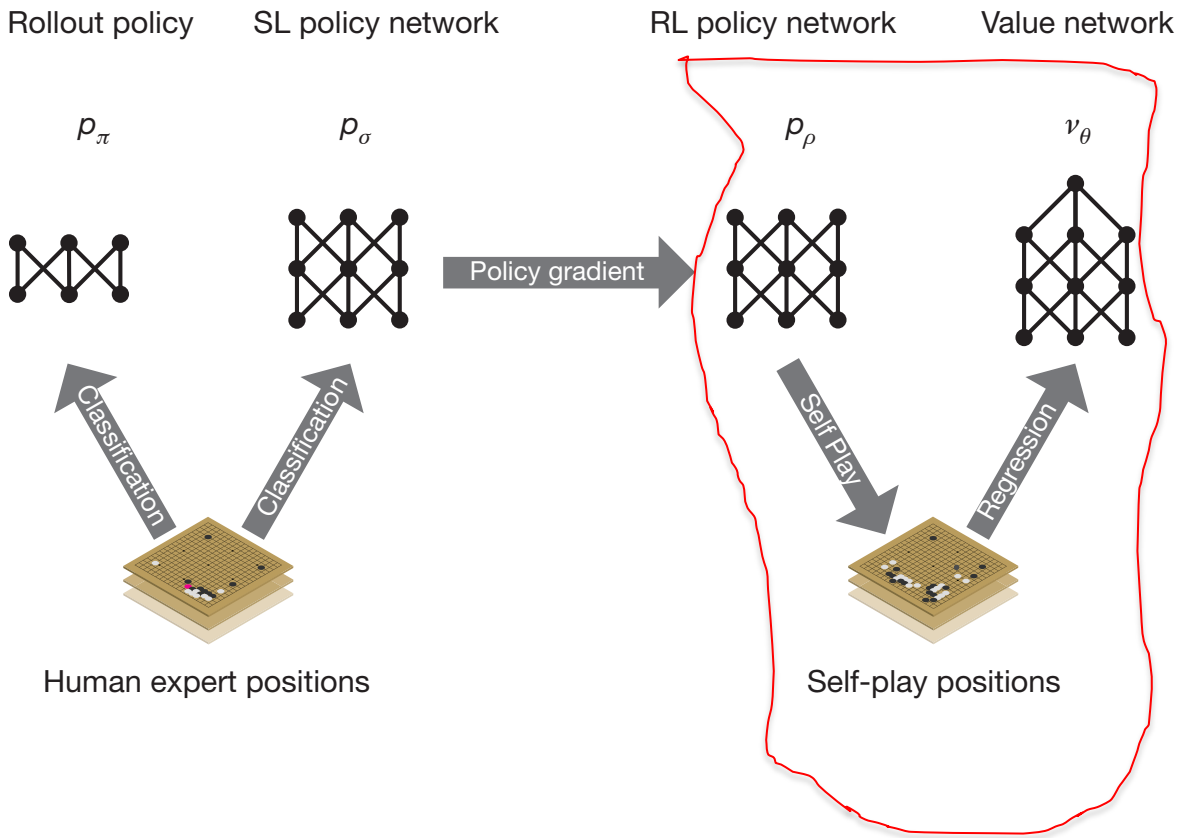
$$\Delta\rho \propto \frac{\partial \log p(a_t \mid s_t)}{\partial \rho} z_t$$

z_t : terminal reward

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

AlphaGo

- Training for value network predicting terminal reward



Selfplay p_p randomly to train value network

$v^\rho(s)$: position evaluation

$v^\rho(s) = \mathrm{E}[z \mid s_t = s, a_t ... T \sim \rho]$

z_t : terminal reward

Gradient update:

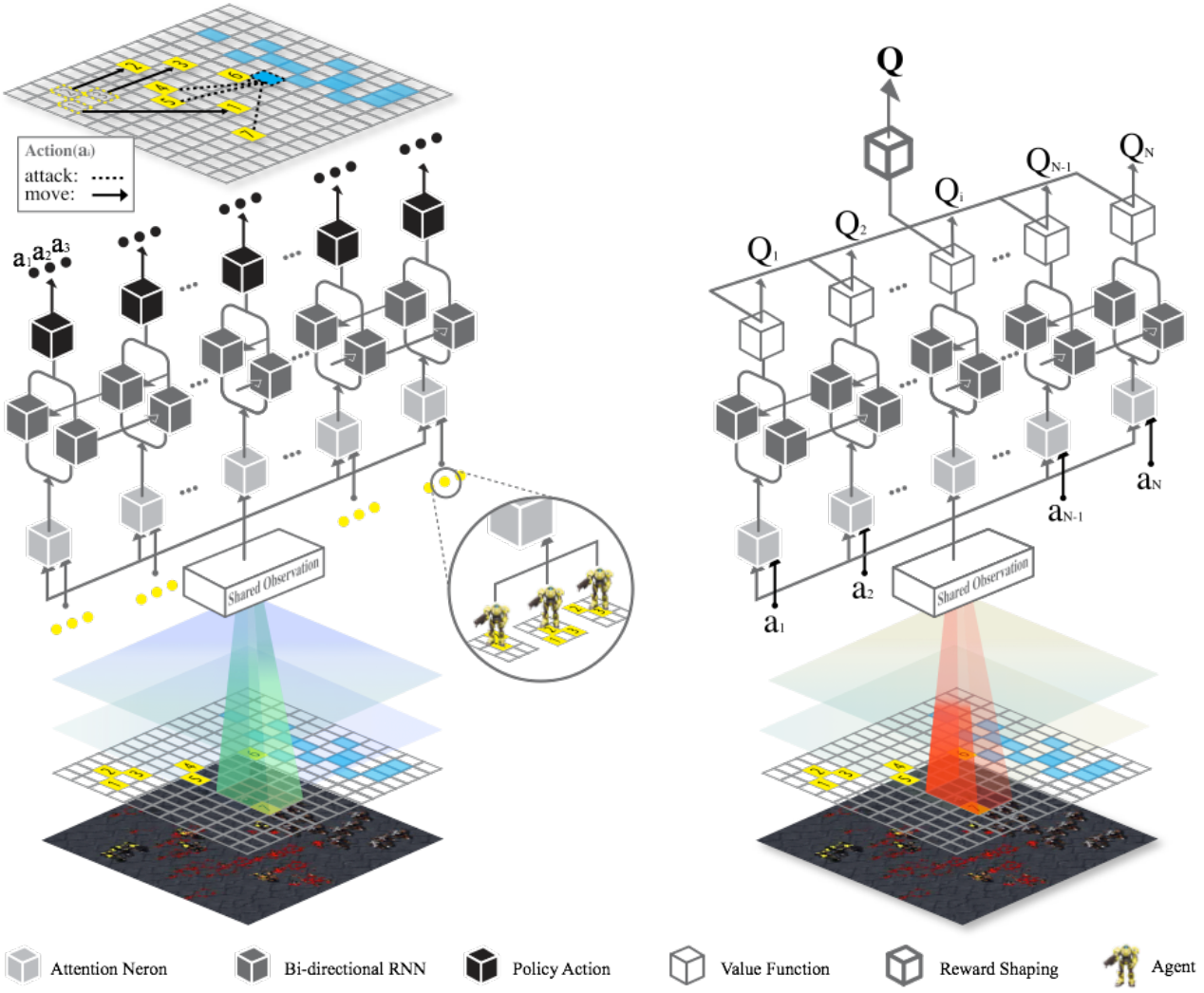
$$\Delta\theta \propto \frac{\partial \log v_\theta(s)}{\partial \theta}(z - v(s))$$

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

AI plays StarCraft

- One of the most difficult games for computers
- At least 10^{1685} possible states (for reference, the game of Go has about 10^{170} states)!
- Multiagent reinforcement learning: how large-scale multiple AI agents could learn human-level collaborations, or competitions, from their experiences?

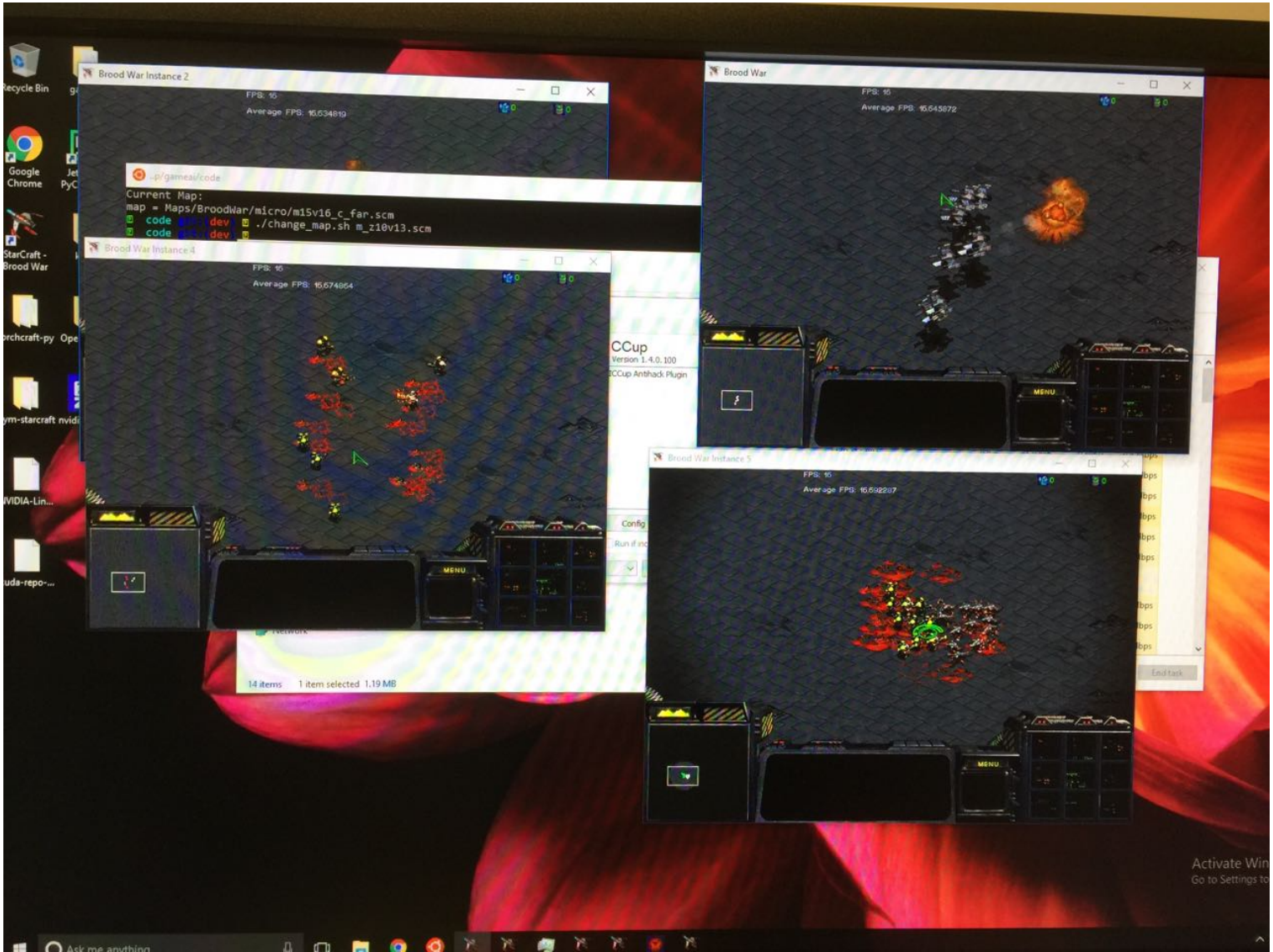
Bidirectional-Coordinated nets (BiCNet)



(a) Multiagent policy networks with grouping (b) Multiagent Q networks with reward shaping

Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, Jun Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, 2017

Unsupervised training without human demonstration and labelled data



Coordinated moves without collision

Combat 3 Marines (ours) vs. 1 Super Zergling (enemy)

(a) Early stage of training (b) Early stage of training (c) Well-trained (d) Well-trained

- The first two (a) and (b) illustrate that the collision happens when the agents are close by during the early stage of the training;
- the last two (c) and (d) illustrate coordinated moves over the well-trained agents

Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, Jun Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, 2017

“Hit and Run” tactics

combat 3 Marines (ours) vs. 1 Zealot (enemy)

(a) time step 1: run when attacked

(b) time step 2: fight back when safe

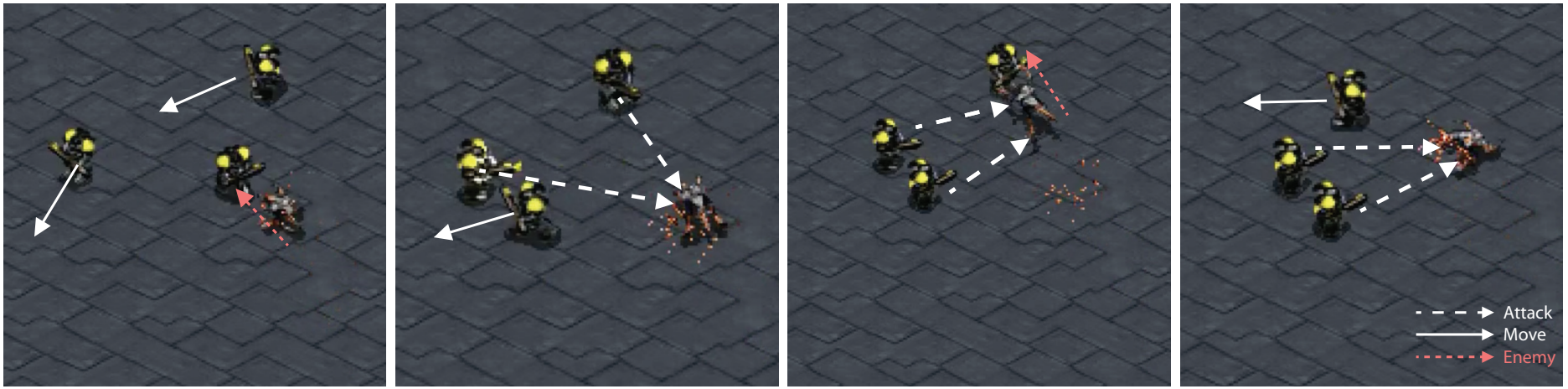
(c) time step 3: run again

(d) time step 4: fight back again

Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, Jun Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, 2017

Coordinated moves without collision

Combat 3 Marines (ours) vs. 1 Zergling (enemy)

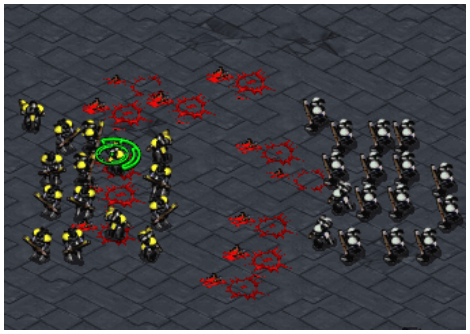


(a) time step 1 (b) time step 2 (c) time step 3 (d) time step 4

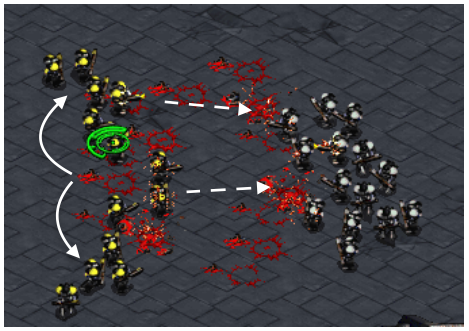
Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, Jun Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, 2017

Focus fire

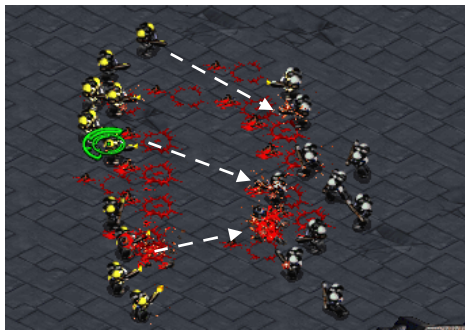
combat 15 Marines (ours) vs. 16 Marines (enemy)



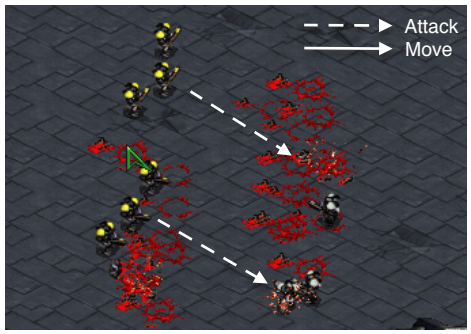
(a) time step 1



(b) time step 2



(c) time step 3

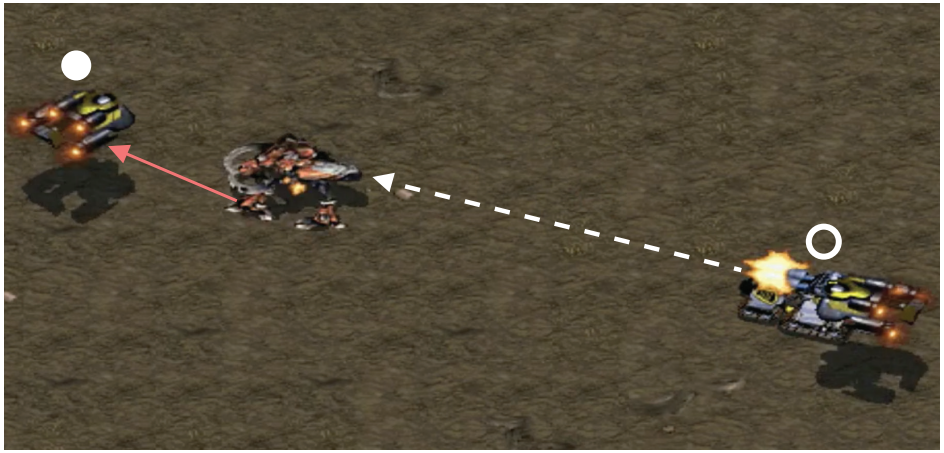


(d) time step 4

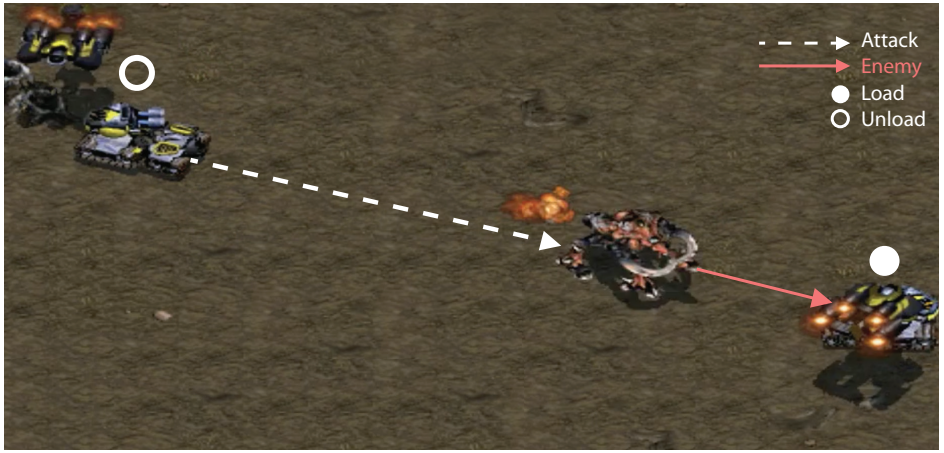
Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, Jun Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, 2017

Coordinated heterogeneous agents

combat 2 Dropships and 2 tanks vs. 1 Ultralisk



(a) time step 1



(b) time step 2

Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, Jun Wang, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, 2017

AI playing StarCraft demo

in collaboration with Alibaba group

Reference Books

- Haykin S S, Haykin S S, Haykin S S, et al. Neural networks and learning machines[M]. Upper Saddle River: Pearson Education, 2009. *very comprehensive*
- Hertz, John, Anders Krogh, and Richard G. Palmer. Introduction to the theory of neural computation. Vol. 1. Basic Books, 1991. *old classic (know the history)*
- Bishop, Christopher M. Neural networks for pattern recognition. Oxford university press, 1995. *narrowly focused, but indepth and with Bayesian view*
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep learning" MIT Press in preparation 2006. *from one of the three top research group on deep learning* http://www.deeplearningbook.org/
- Nikhil Buduma, Fundamentals of Deep Learning - Designing Next-Generation Machine Intelligence Algorithms, 2015. *not ready yet*
- Deng, Li, and Dong Yu. "Deep learning: Methods and applications." Foundations and Trends in Signal Processing 7.3–4 (2014): 197-387 . *Packed with recent papers not recommended if you are a starter.*

  http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf
- Michael nielsen, Neural Networks and Deep Learning

  http://neuralnetworksanddeeplearning.com/index.html *Easy read*

Early Neural networks

- Marvin L. Minsky. 1967. Computation: Finite and Infinite Machines. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65.6 (1958): 386.
- WERBOS, P. "Beyond regression: new fools for prediction and analysis in the behavioral sciences." PhD thesis, Harvard University (1974).
- Parker, 1985, Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT.
- Rumelhart D E, Hinton G E, Williams R J. Learning internal representations by error propagation[R]. CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1985.
- Hornik, Kurt. "Approximation capabilities of multilayer feedforward networks." Neural networks 4.2 (1991): 251-257.

Layer-Local Unsupervised Learning

Restricted Boltzmann machines

- Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.
- Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.
- Fischer, Asja, and Christian Igel. "An introduction to restricted Boltzmann machines." Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer Berlin Heidelberg, 2012. 14-36.
- Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

Autoencoders

- Vincent, H. Larochelle Y. Bengio and P.A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML‘08), pages 1096 - 1103, ACM, 2008.
- Bengio, Yoshua, et al. "Greedy layer-wise training of deep networks." Advances in neural information processing systems 19 (2007): 153.

Understanding and Why questions

- Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." International conference on artificial intelligence and statistics. 2010.
- Erhan, Dumitru, et al. "Why does unsupervised pre-training help deep learning?." The Journal of Machine Learning Research 11 (2010): 625-660.
- Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks[J]. arXiv preprint arXiv:1211.5063, 2012.
- Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.
- Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning[J]. arXiv preprint arXiv:1506.02142, 2015.
- Karpathy, Andrej, Justin Johnson, and Fei-Fei Li. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078 (2015).

Recurrent Neural Networks (RNN)

- Werbos, Paul J. "Backpropagation through time: what it does and how to do it." Proceedings of the IEEE 78.10 (1990): 1550-1560.
- Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.
- Boden M. A guide to recurrent neural networks and backpropagation[J]. The Dallas project, SICS technical report, 2002.
- Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.
- Mikolov, Tomas, et al. "Recurrent neural network based language model." INTERSPEECH. Vol. 2. 2010.
- Koutnik, Jan, et al. "A clockwork rnn." arXiv preprint arXiv:1402.3511 (2014).
- Grégoire Mesnil, Xiaodong He, Li Deng and Yoshua Bengio. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. Interspeech, 2013.

Convolutional Neural Networks (CNN)

- Hubel D.H. : The Visual Cortex of the Brain Sci Amer 209:54-62, 1963
- LeCun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989, 1(4): 541-551.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998
- Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.
- Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems. 2012: 1097-1105.
- Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
- He, Kaiming, et al. "Deep Residual Learning for Image Recognition." arXiv preprint arXiv:1512.03385 (2015).
- Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
- Maddison, Chris J., et al. "Move evaluation in go using deep convolutional neural networks." arXiv preprint arXiv:1412.6564 (2014).

Word2Vec and Language Models

- Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).
- Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
- Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations." HLT-NAACL. 2013.
- Zou, Will Y., et al. "Bilingual Word Embeddings for Phrase-Based Machine Translation." EMNLP. 2013.
- Bengio, Yoshua, et al. "Neural probabilistic language models." Innovations in Machine Learning. Springer Berlin Heidelberg, 2006. 137-186.

Deep Reinforcement Learning and Game

- Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 1998.
- Neumann, L. J., & Morgenstern, O. (1947). Theory of games and economic behavior (Vol. 60). Princeton: Princeton university press.
- Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
- Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
- Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.
- Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25 (1995): 27.
- Browne, Cameron B., et al. "A survey of monte carlo tree search methods." Computational Intelligence and AI in Games, IEEE Transactions on 4.1 (2012): 1-43.
- Kocsis, Levente, and Csaba Szepesvári. "Bandit based monte-carlo planning." Machine Learning: ECML 2006. Springer Berlin Heidelberg, 2006. 282-293.
- Coulom, Rémi. "Efficient selectivity and backup operators in Monte-Carlo tree search." Computers and games. Springer Berlin Heidelberg, 2006. 72-83.