UVA CS 4501: Machine Learning

Lecture 14: Neural Network / Deep Learning

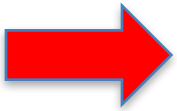
Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Where are we ? ➔
Five major sections of this course

- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

Today



- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN

Many classification models invented since late 80's

- Neural networks
- Boosting
- Support Vector Machine
- Random Forest
- ……

A study comparing Classifiers

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine

Proceedings of the 23rd International Conference on Machine Learning (ICML `06).

A study comparing Classifiers

➔ 11 binary classification problems / 8 metrics

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

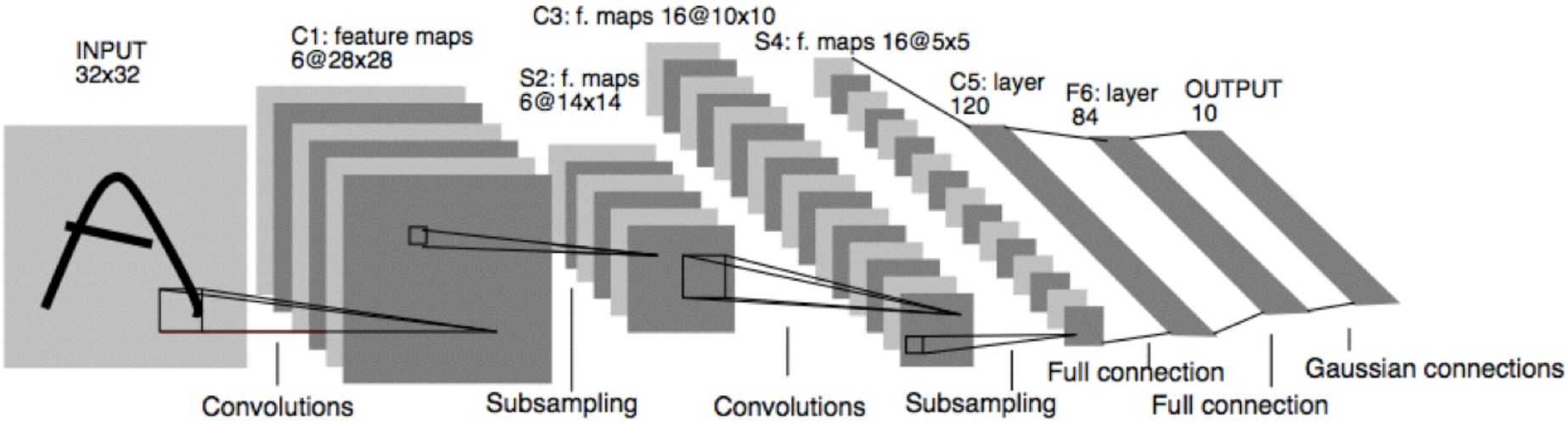
MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774

Proceedings of the 23rd International Conference on Machine Learning (ICML `06).

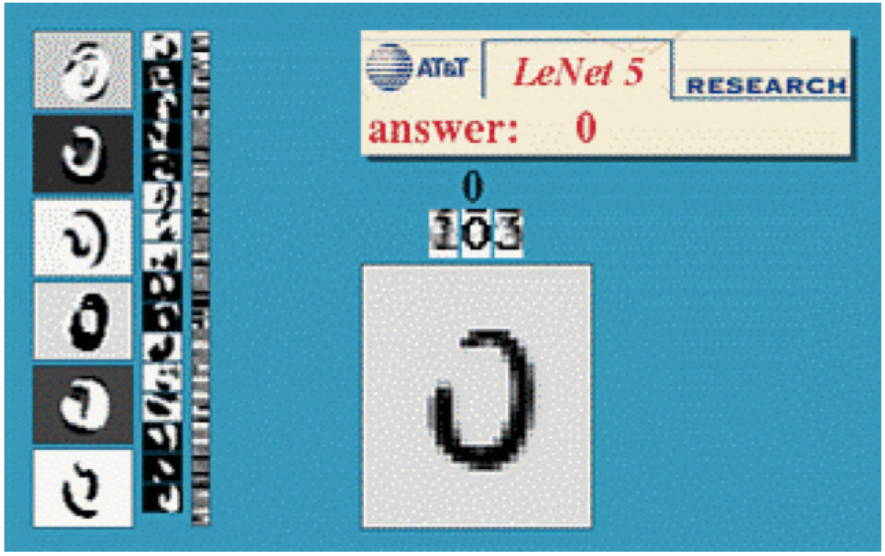
Deep Learning in the 90's

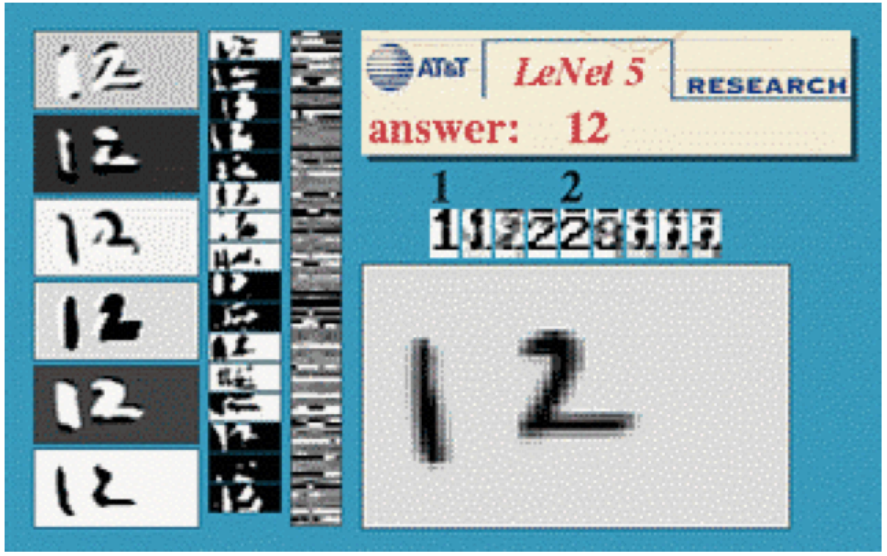
- Prof. Yann LeCun invented Convolutional Neural Networks

- First NN successfully trained with many layers



“LeNet” Early success at OCR

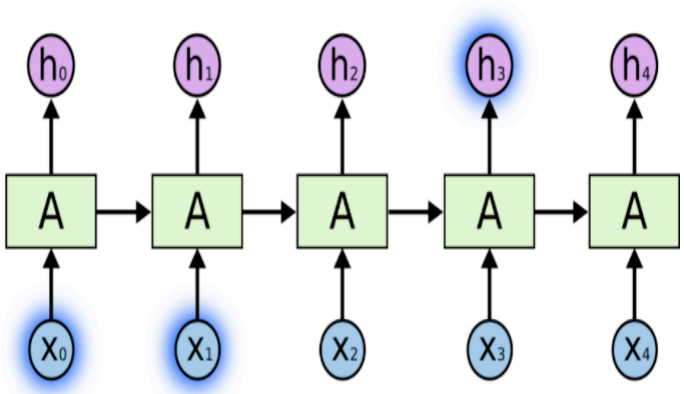


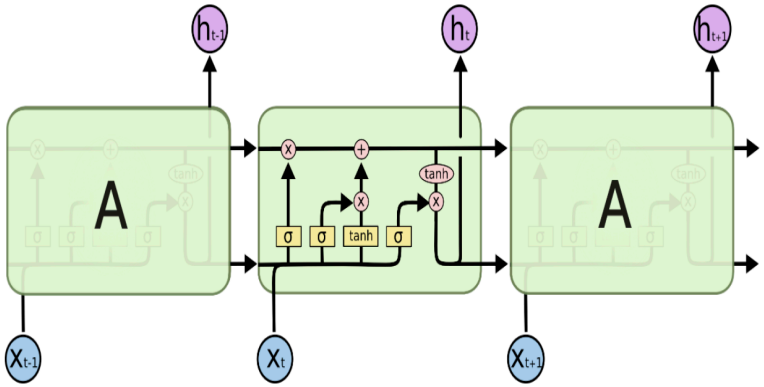


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Deep Learning in the 90's

- Prof. Schmidhuber invented "Long short-term memory" – Recurrent NN (LSTM-RNN) model in 1997





The repeating module in an LSTM contains four interacting layers.

Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780.

Image Credits from Christopl

Between ~2000 to ~2011 Machine Learning Field Interest

- Learning with Structures ! + Convex Formulation!
  - Kernel learning
  - Transfer Learning
  - Semi-supervised
  - Manifold Learning
  - Sparse Learning
  - Structured input-output learning …
  - Graphical model

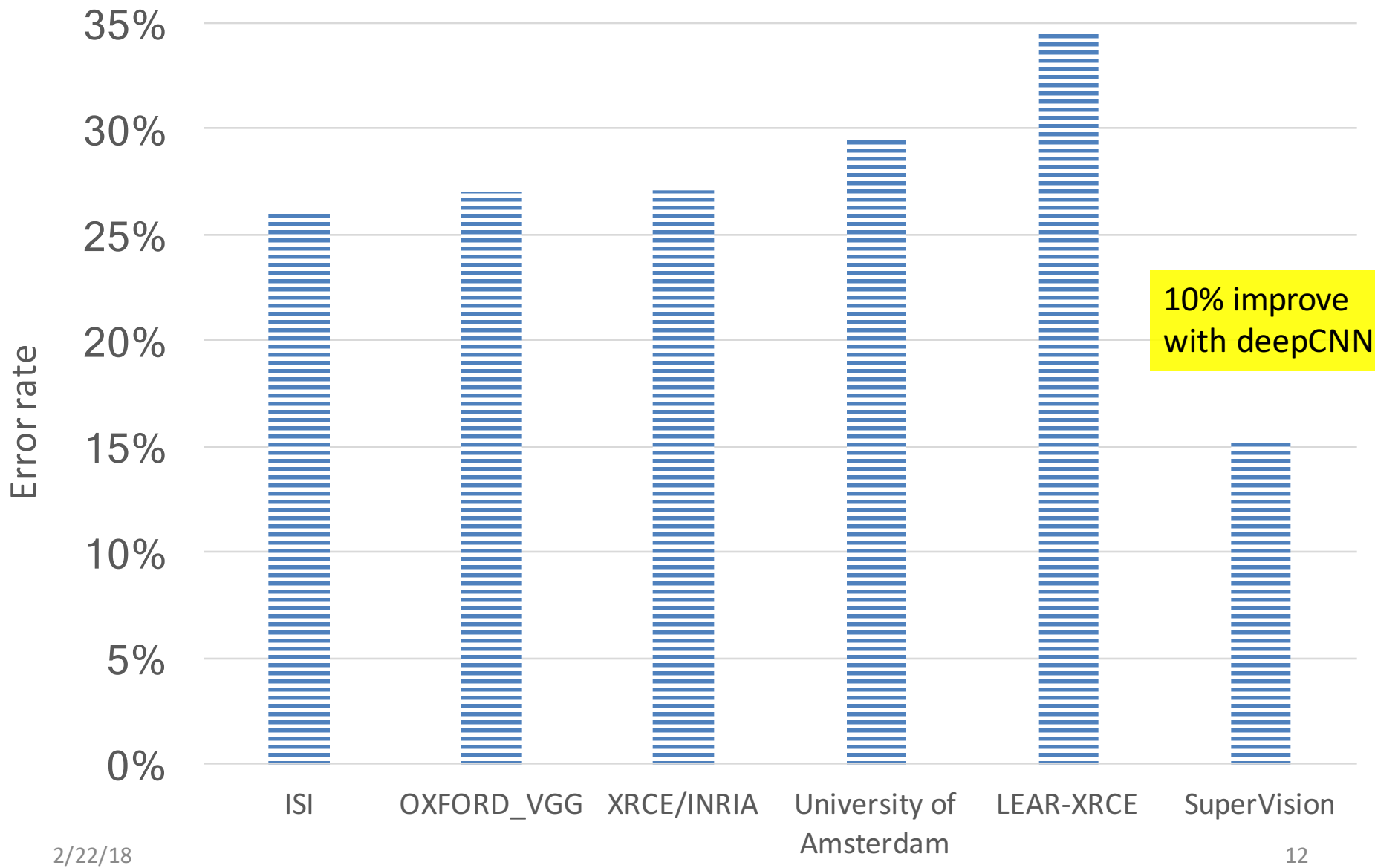
"Winter of Neural Networks" Since 90's ! to ~2011

- Non-convex

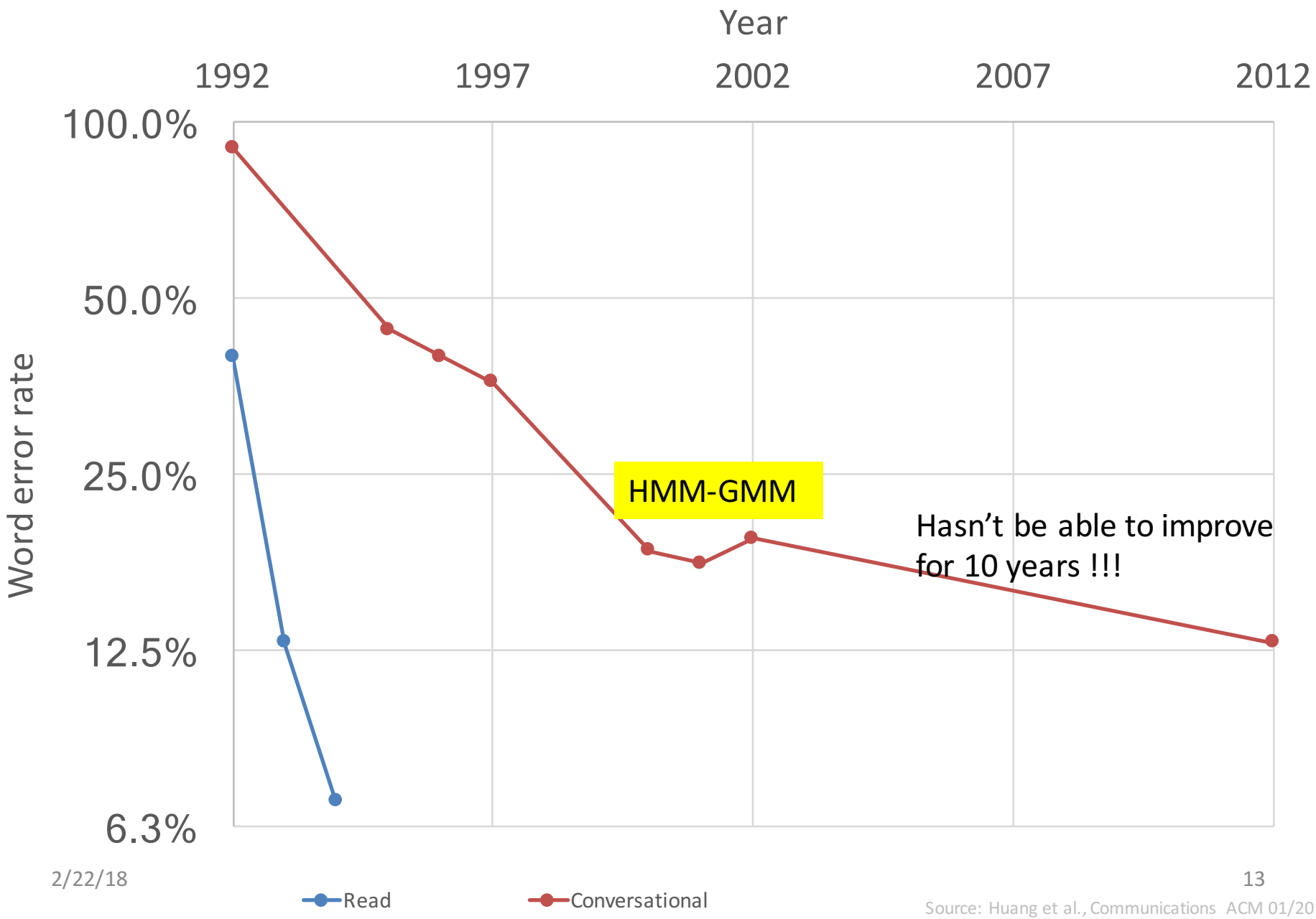
- Need a lot of tricks to play with
  - How many layers ?
  - How many hidden units per layer ?
  - What topology among layers ? …….

- Hard to perform theoretical analysis

Large-Scale Visual Recognition Challenge 2012



Speech Recognition



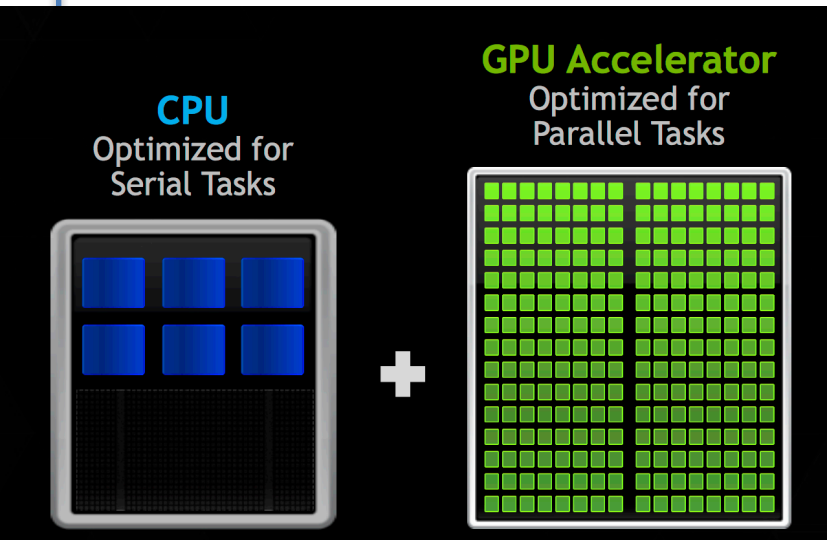
Source: Huang et al., Communications ACM 01/2014

Reason: Plenty of (Labeled) Data

- Text: trillions of words of English + other languages
- Visual: billions of images and videos
- Audio: thousands of hours of speech per day
- User activity: queries, user page clicks, map requests, etc,
- Knowledge graph: billions of labeled relational triplets
- ……

Dr. Jeff Dean's talk

Reason: Advanced Computer Architecture that fits DNNs



http://www.nvidia.com/content/events/geoInt2015/LBrown_DL.pdf

	Neural Networks	GPUs
Inherently Parallel	✓	✓
Matrix Operations	✓	✓
FLOPS	✓	✓

GPUs deliver --
- *same or better prediction accuracy*
- *faster results*
- *smaller footprint*
- *lower power*

Today

- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN

MIT Technology Review

10 Breakthrough Technologies 2013

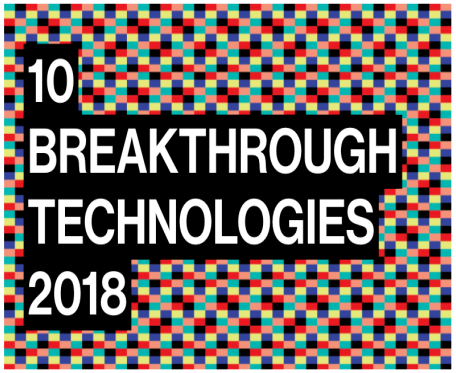
Think of the most frustrating, intractable, or simply annoying problems you can imagine. Now think about what technology is doing to fix them. That's what we did in coming up with our annual list of 10 Breakthrough Technologies. We're looking for technologies that we believe will expand the scope of human possibilities.

Deep Learning

10 Breakthrough Technologies 2017

These technologies all have staying power. They will affect the economy and our politics, improve medicine, or influence our culture. Some are unfolding now; others will take a decade or more to develop. But you should know about all of them right now.

Deep Reinforcement Learning



Generative Adversarial Network (GAN)

WHY BREAKTHROUGH ?

How can we build more intelligent computer / machine ?

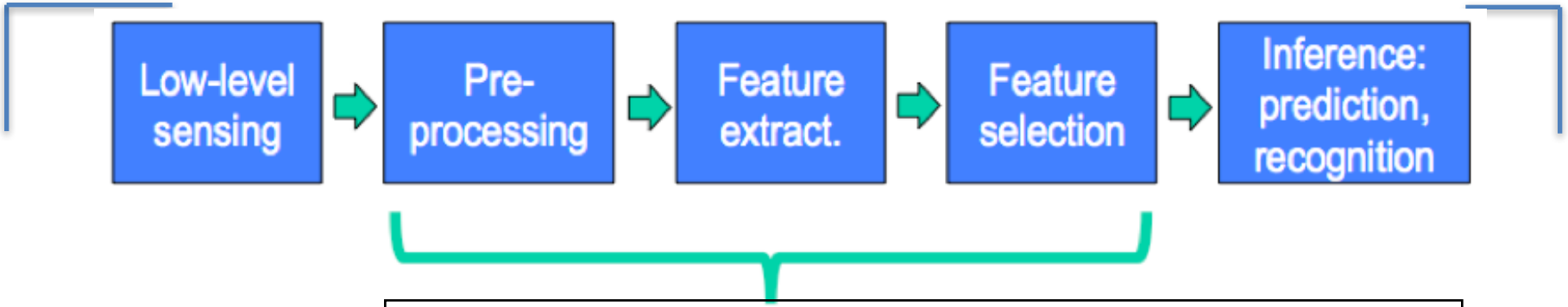
- Able to
  - **Perceive the world**
  - **Understand the world**
  - **Interact with the world**

- This needs
  - Basic speech capabilities
  - Basic vision capabilities
  - Language understanding
  - User behavior / emotion understanding
  - Being able to think / reason …

DNNs help us build more intelligent computers **(Extra)**

- Perceive the world,
  - e.g., objective recognition, speech recognition, …
- Understand the world,
  - e.g., machine translation, text semantic understanding
- Interact with the world,
  - e.g., AlphaGo, AlphaZero, self-driving cars, …
- Being able to think / reason,
  - e.g., learn to code programs, learn to search deepNN, …
- Being able to imagine / to make analogy,
  - e.g., learn to draw with styles, ……

Deep Learning Way: Learning features / Representation from data



Feature Engineering

- ✓ Most critical for accuracy
- ✓ Account for most of the computation for testing
- ✓ Most time-consuming in development cycle
- ✓ Often hand-craft and task dependent in practice

Feature Learning

- ✓ Easily adaptable to new similar tasks
- ✓ Layerwise representation
- ✓ Layer-by-layer unsupervised training
- ✓ Layer-by-layer supervised training

To perceive the world: Application I: Objective Recognition / Image Labeling

72%, 2010

74%, 2011

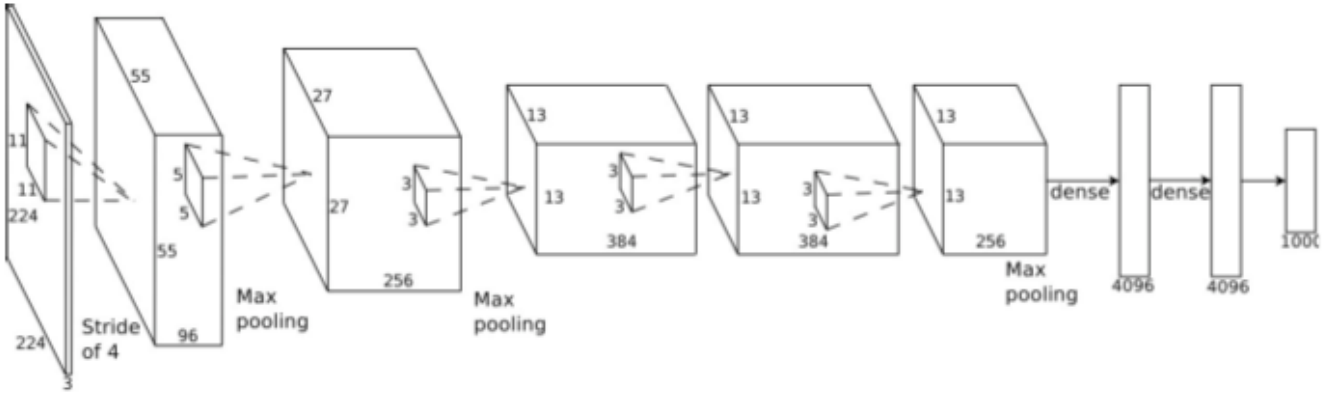
85%, 2012

"Very large-scale" ImageNet competition

(training on 1.2 million images [X]

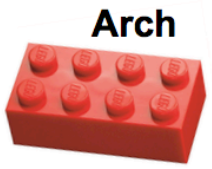
vs.1000 different word labels [Y])

To perceive the world: Application I: Objective Recognition / Image Labeling

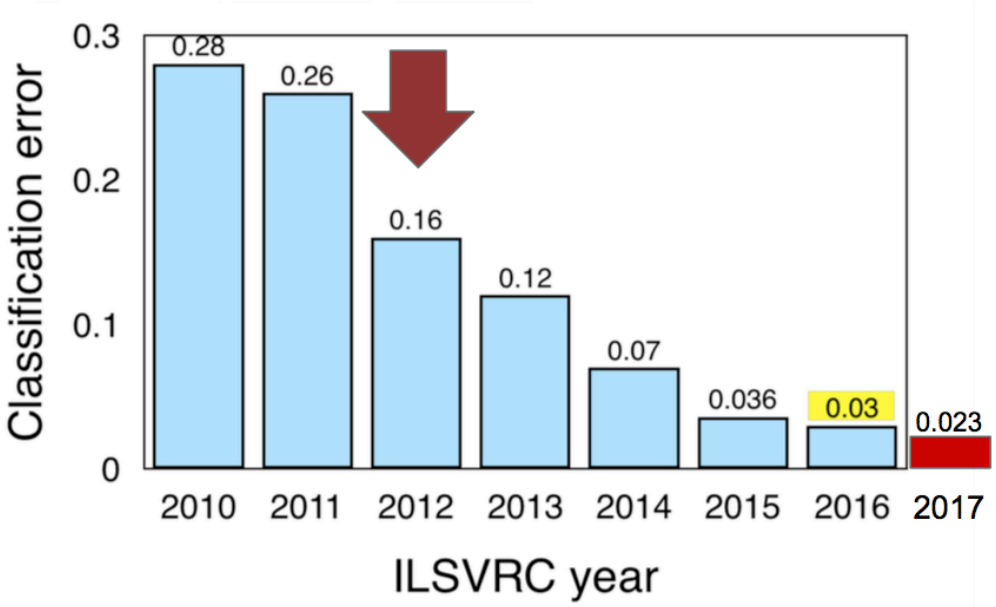


Deep Convolution Neural Network (CNN) and variants have won (as best systems) on "very large-scale" ImageNet competition 2012-2017

ImageNet Challenge

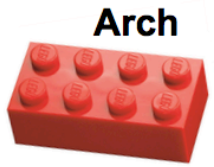


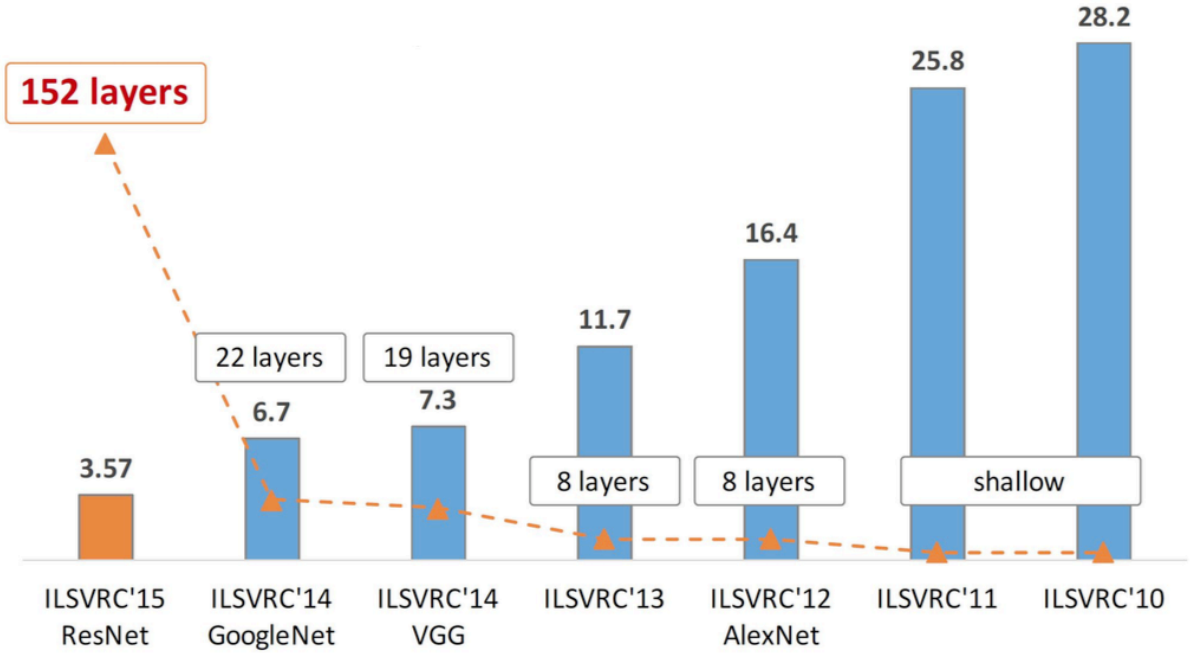
- 2010-11: hand-crafted computer vision pipelines
- 2012-2016: ConvNets
  - 2012: AlexNet
    - major deep learning success
  - 2013: ZFNet
    - improvements over AlexNet
  - 2014
    - VGGNet: deeper, simpler
    - InceptionNet: deeper, faster
  - 2015
    - ResNet: even deeper
  - 2016
    - ensembled networks
  - 2017
    - Squeeze and Excitation Network



Adapt from From NIPS 2017 DL Trend Tutorial

Revolution of Depth





Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

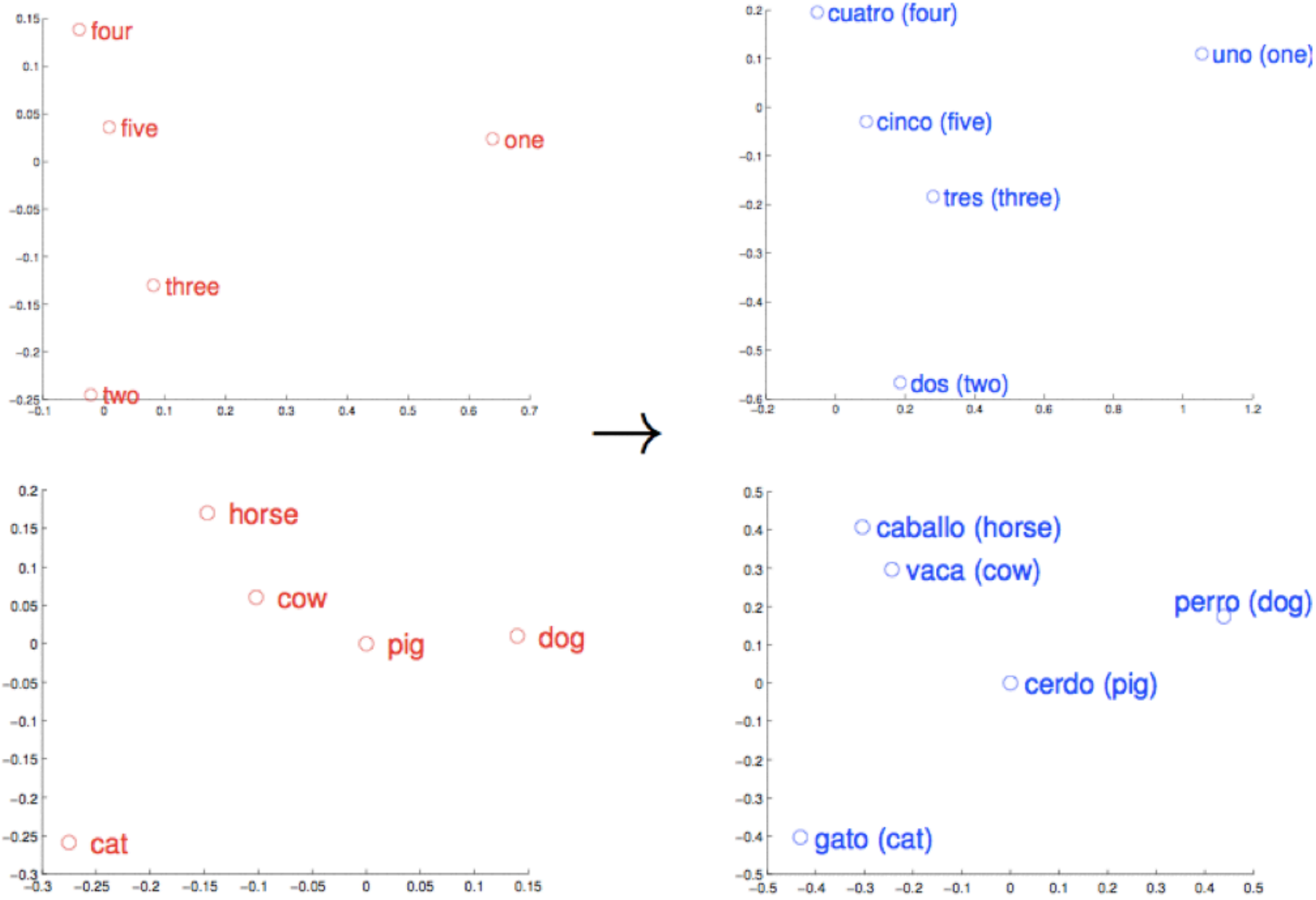
Adapt from From NIPS 2017 DL Trend Tutorial

To understand the world: Application II: Semantic Understanding e.g. Word embedding / Neural Language Models

- Learn to embed each word into a vector of real values
  - Semantically similar words have closer embedding representations

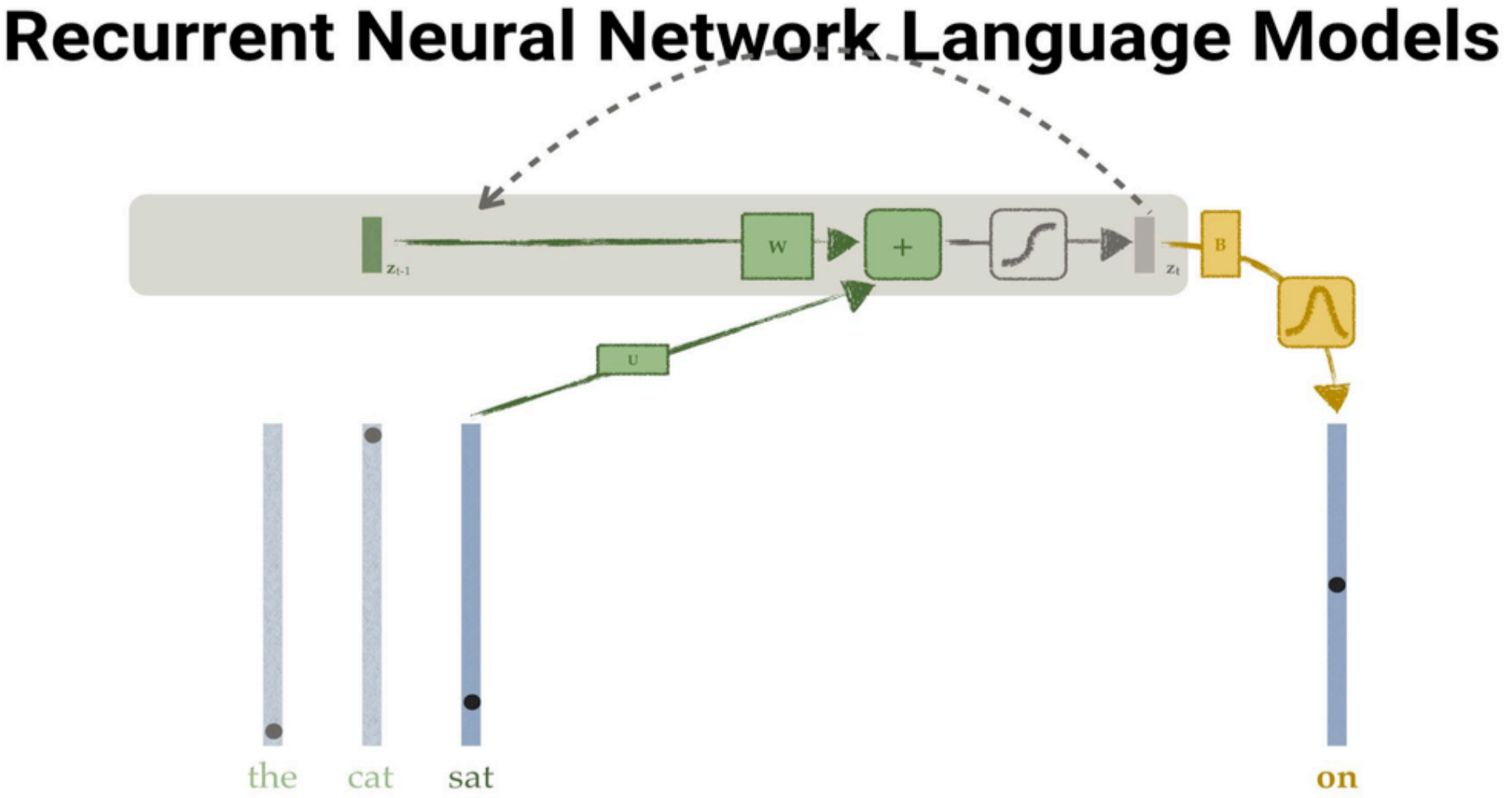
- Progress in 2013/14
  - Arithmetic operations for semantic /syntactic word relationships
    - [king] - [male] + [female] ~= [queen]
    - [Berlin] - [Germany] + [France] ~= [Paris]
    - [eating] - [eat] + [fly] ~= [flying]

Dr. Jeff Dean's talk



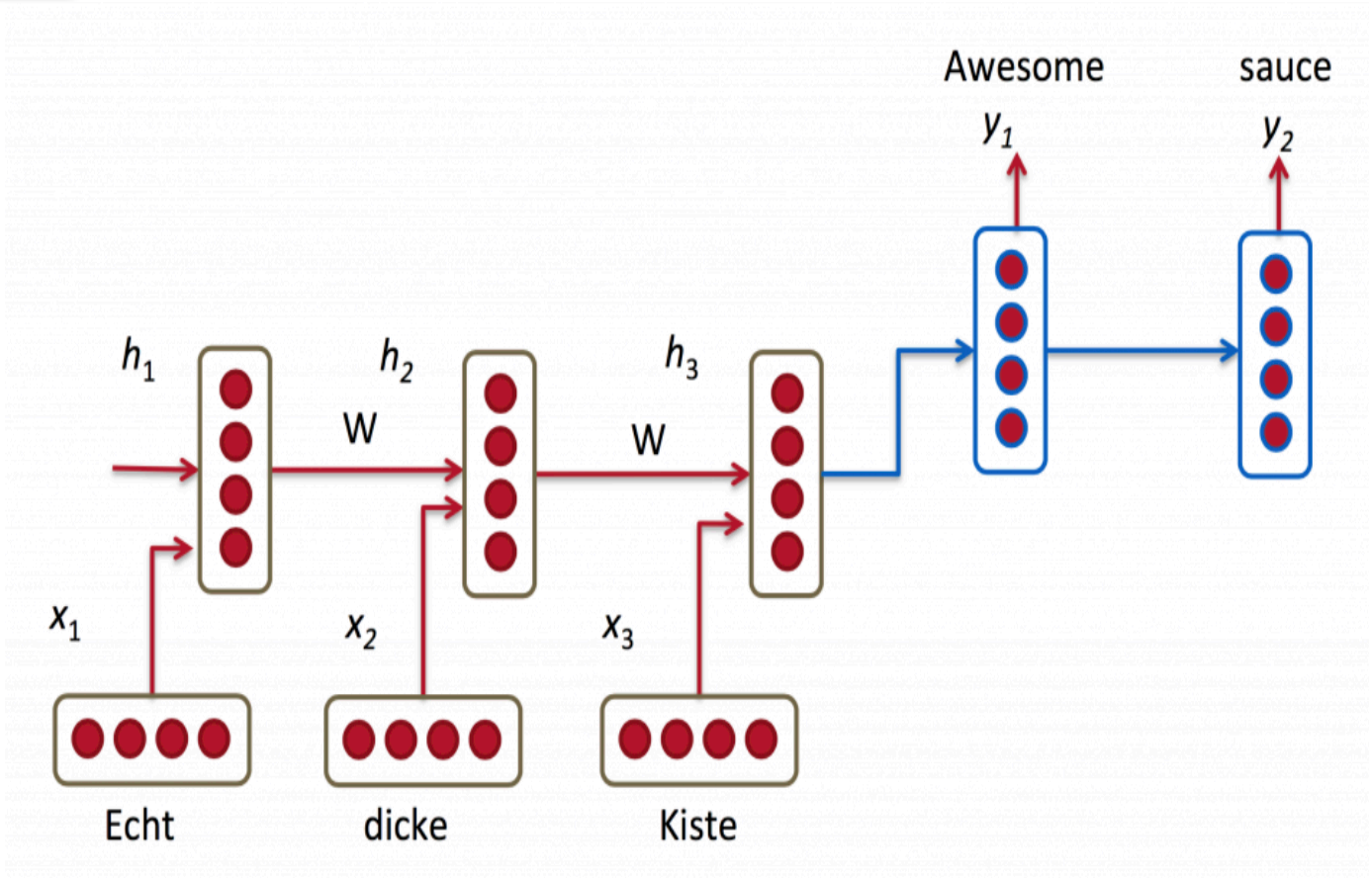
source: Exploiting Similarities among Languages for MT

e.g.



http://cs231n.stanford.edu/slides/

e.g. For machine translation with 2RNN



To interact with the world: Application III: Deep Learning to Play Games

- DeepMind:
  - Learning to Play & win dozens of Atari games
  - new Deep Reinforcement Learning algorithms
  - AlphaGo / AlphaZero

Olivier Grisel's talk

Able to Reason: Application IV:

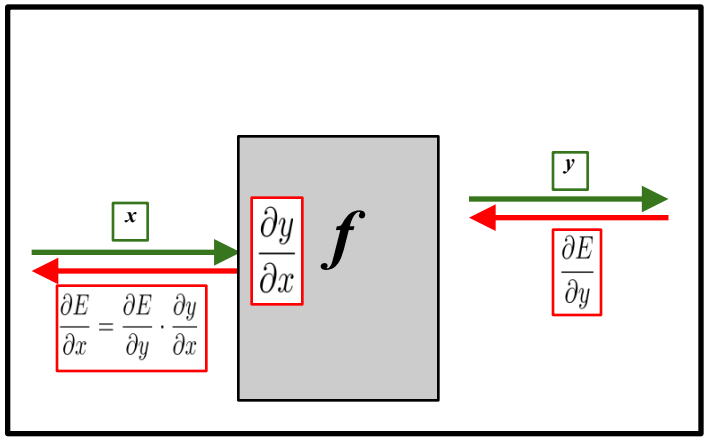
Deep Learning to Execute and Program

Neural Turing Machines

- Google DeepMind, October 2014 (very new)
- Neural Network coupled to external memory (tape)
- Analogue to a Turing Machine but differentiable
- Can be used to learn to simple programs from example input / output pairs

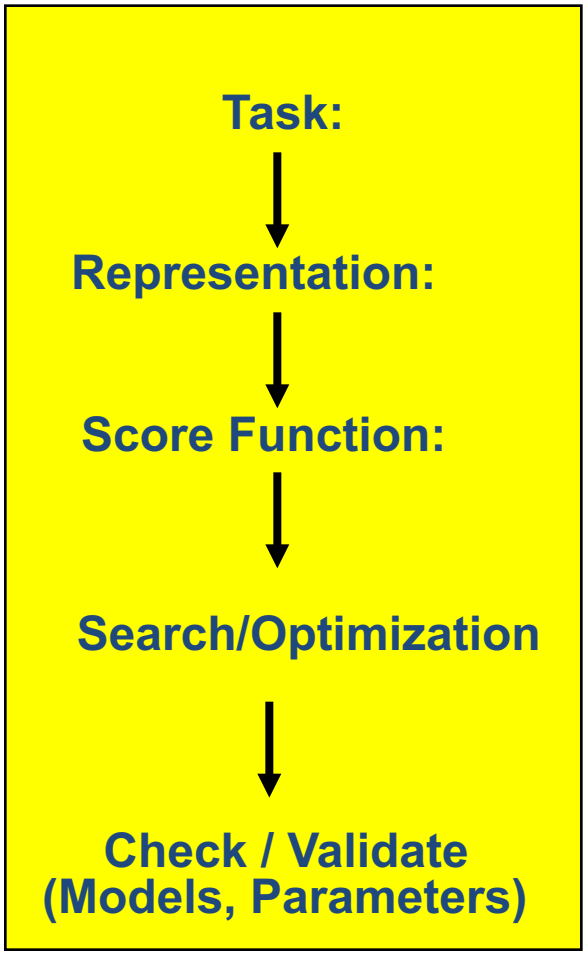
Olivier Grisel's talk

Building Deep Neural Nets



http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Machine (Deep) Learning in a Nutshell



DESIGN Deep NN: Five Aspects (Extra)

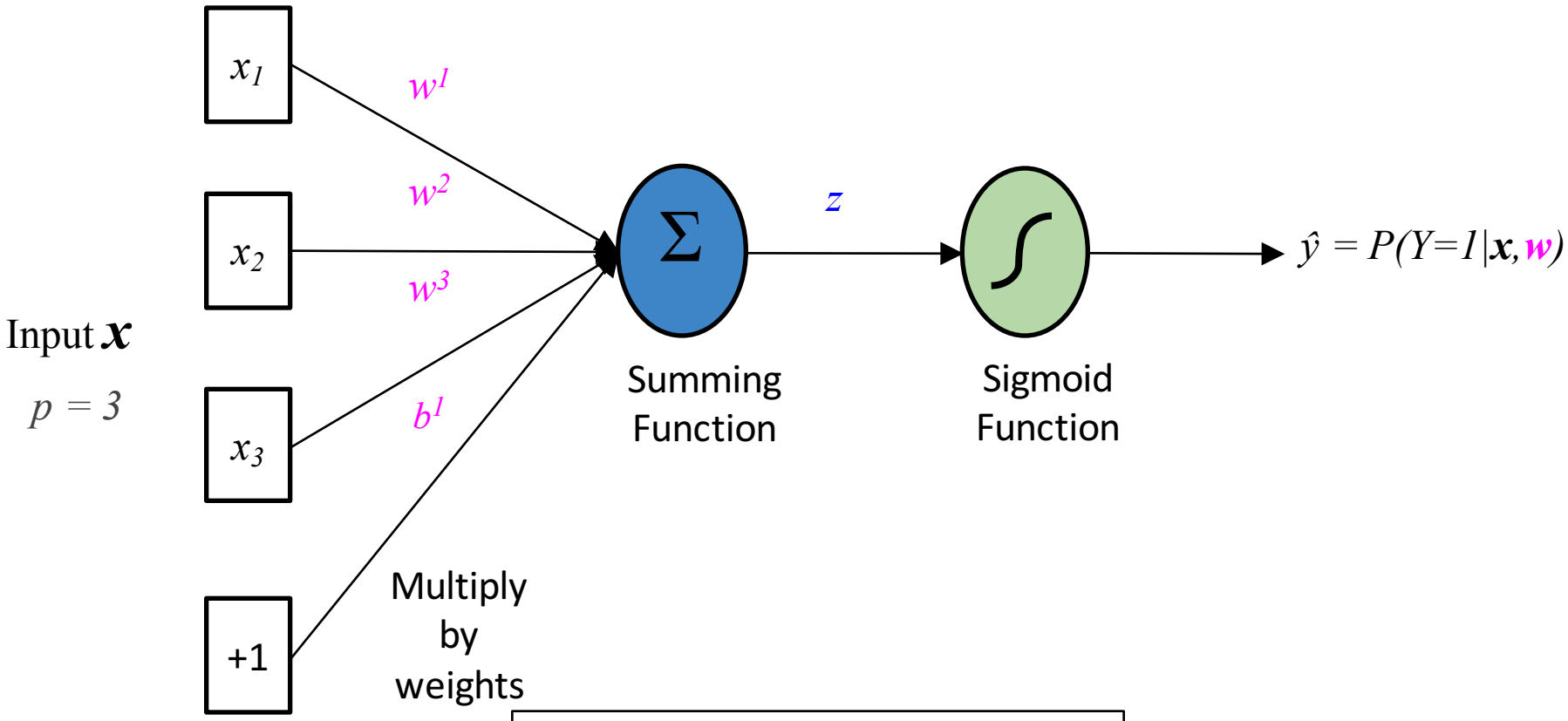
- Tasks:
  - Discriminative classification / Generative / Reinforce / Reasoning …
- Formulate Input / Output:
  - Data representation
- Architecture Design:
  - Network Topology, Network Parameters
- Training / Searching / Learning
  - With new losses / with new optimization tricks
  - New formulation of learning
  - Scaling up with GPU, Scaling up with distributed optimization , e.g. Asynchronous SGD
- Validation / Trust / Test / Understand …

Today

- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN

One "Neuron": Expanded Logistic Regression

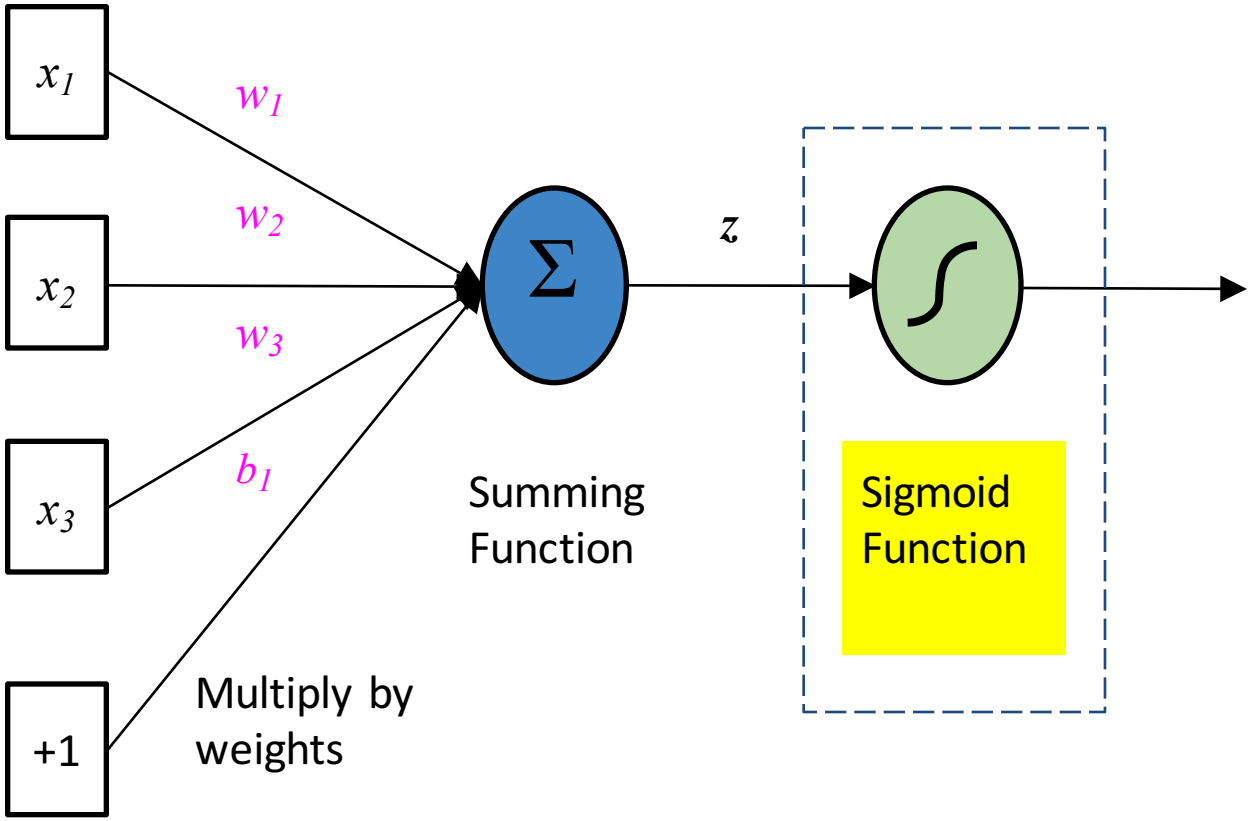


$$z = \boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b}$$

$$y = sigmoid(z) = \frac{e^z}{1 + e^z}$$

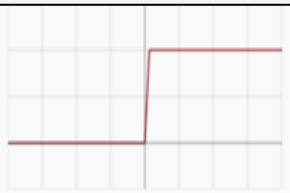
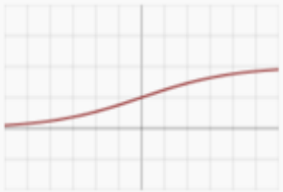
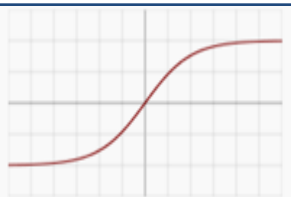
Nonlinearity Functions

(i.e. transfer or activation functions)



Nonlinearity Functions

(aka transfer or activation functions)

Name	Plot	Equation	Derivative (w.r.t x)
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectifier (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$

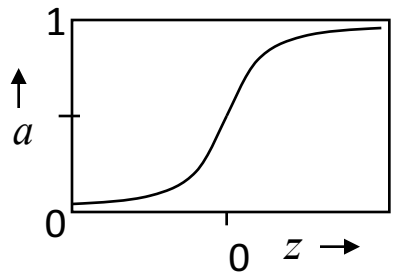
usually works best in practice

https://en.wikipedia.org/wiki/Activation_function#Comparison_of_activation_functions

Transfer / Activation functions

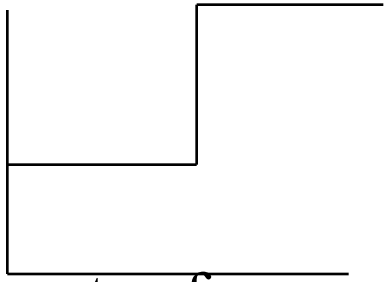
- Common ones include:
  - Threshold / Step function:
    - f(v) = 1 if v > c, else -1
  - Sigmoid (s shape func):
    - E.g. logistic func: $f(v) = 1/(1 + e^{-v})$, Range [0, 1]
  - Hyperbolic Tanh :
    - $f(v) = (e^{v} - e^{-v})/(e^{v} + e^{-v})$, Range [-1,1]

sigmoid func



$$a = \frac{1}{1+e^{-z}}$$

- Desirable properties:
  - Monotonic, Nonlinear, Bounded
  - Easily calculated derivative

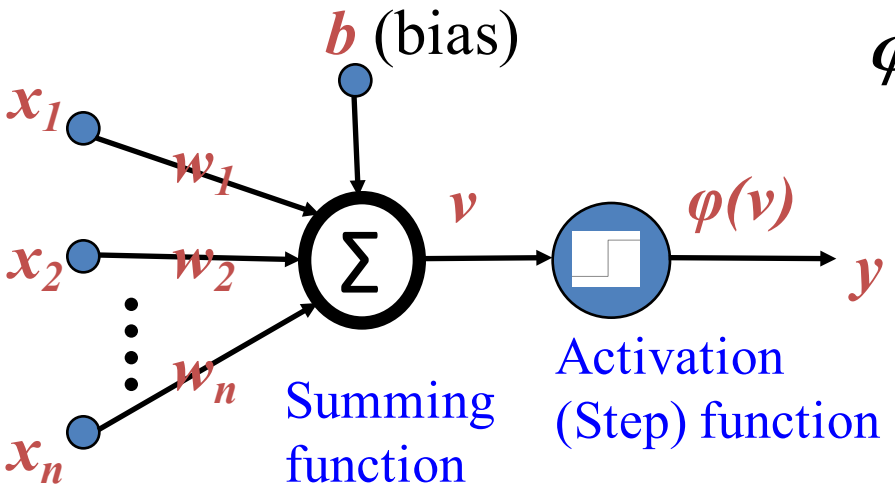


step func

Perceptron: Another 1-Neuron Unit

(Special form of single layer feed forward)

- The perceptron was first proposed by Rosenblatt (1958) is a simple neuron that is used to classify its input into one of two categories.
- A perceptron uses a **step function** that returns +1 if weighted sum of its input large or equal to 0, and -1 otherwise



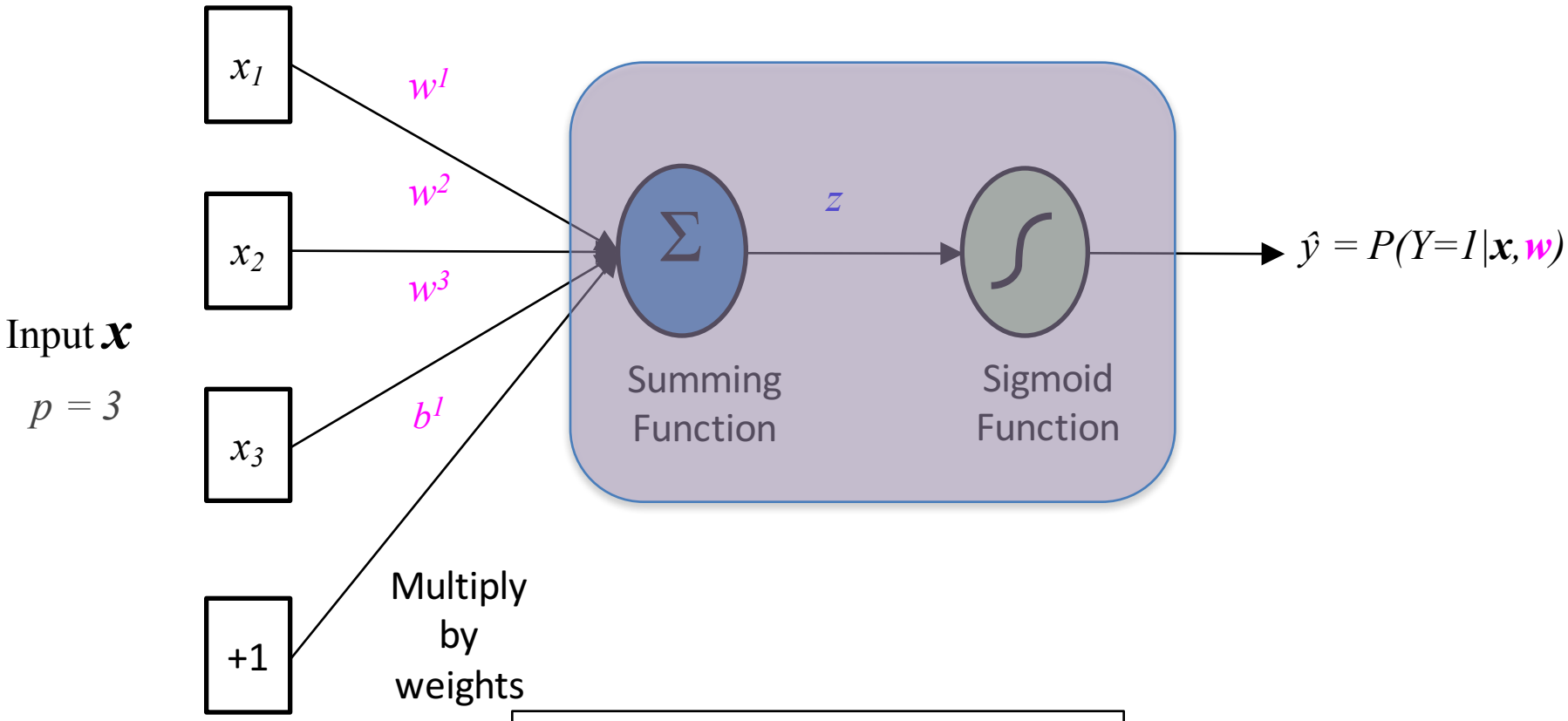
$$\varphi(v) = \begin{cases} +1 \text{ if } v \geq 0 \\ -1 \text{ if } v < 0 \end{cases}$$

Today

- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN

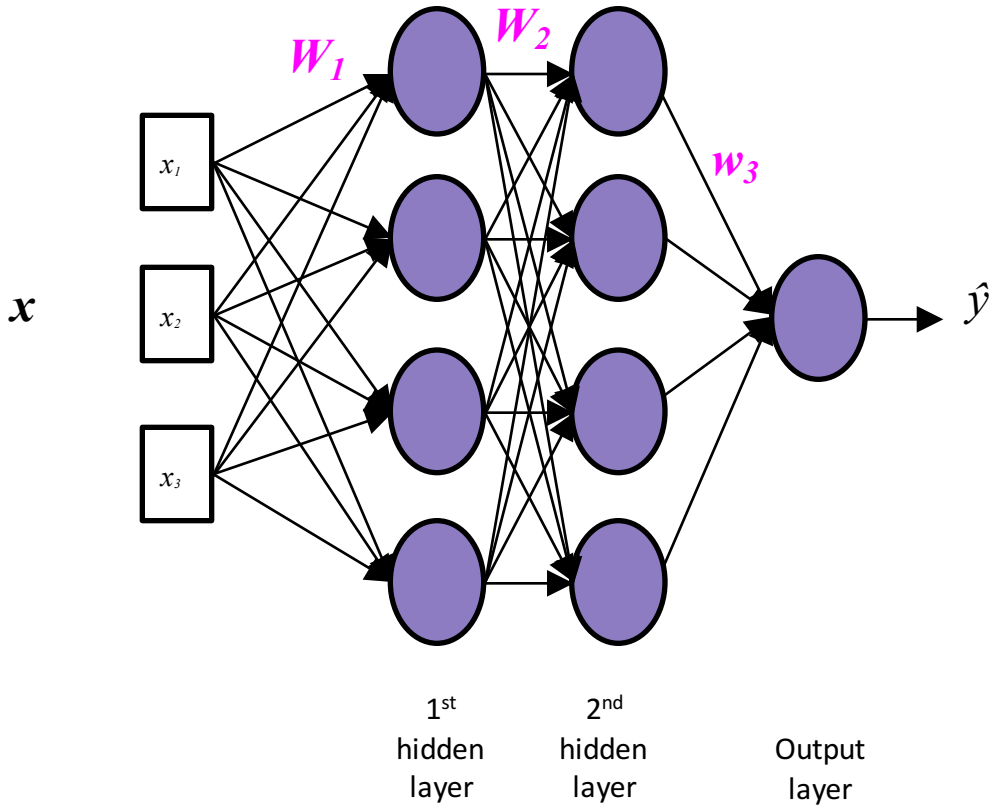
One "Neuron": Expanded Logistic Regression



$$z = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b}$$

$$y = sigmoid(z) = \frac{e^z}{1 + e^z}$$

Multi-Layer Perceptron Neural Network (MLP)



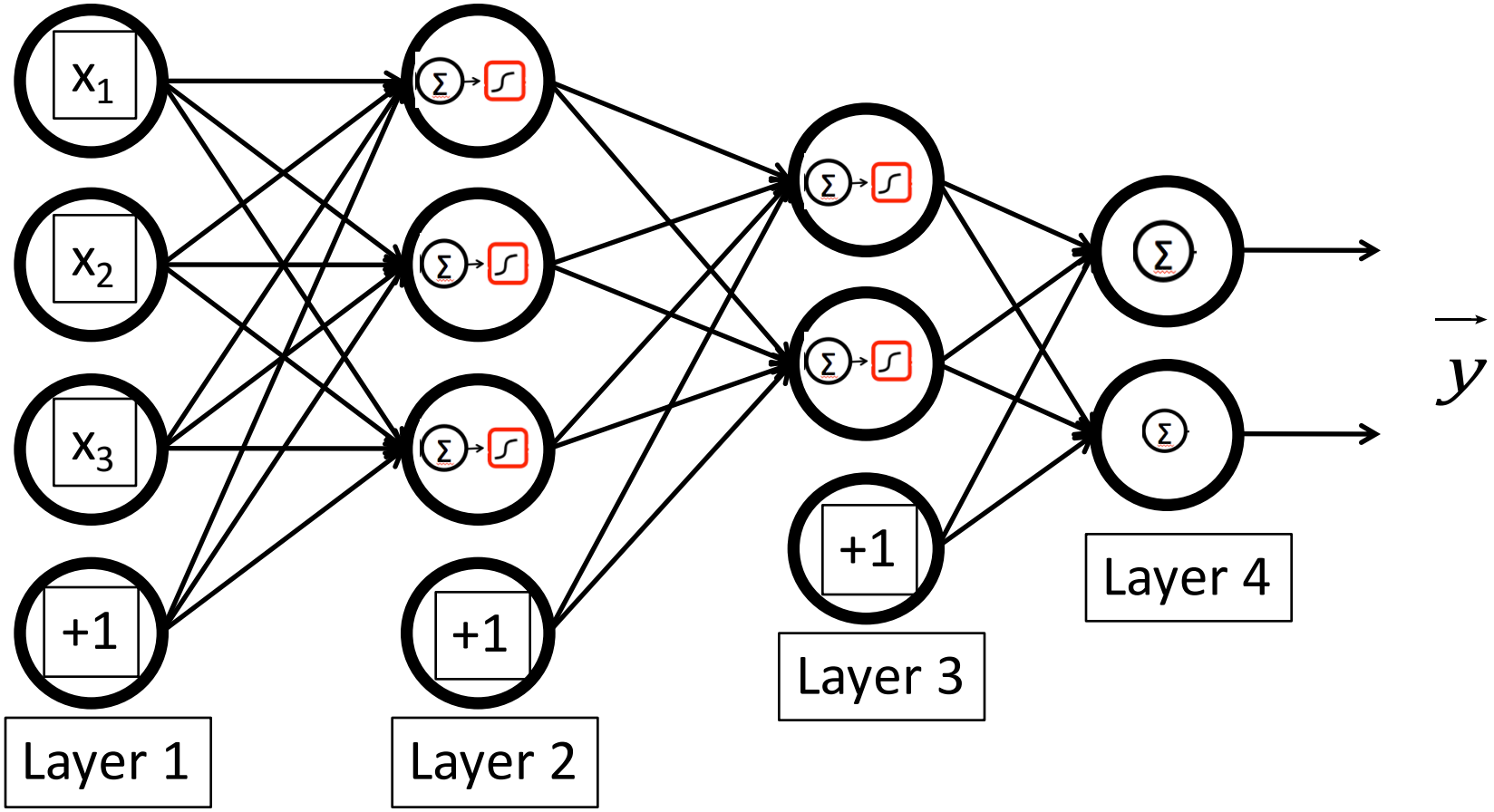
Today

- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

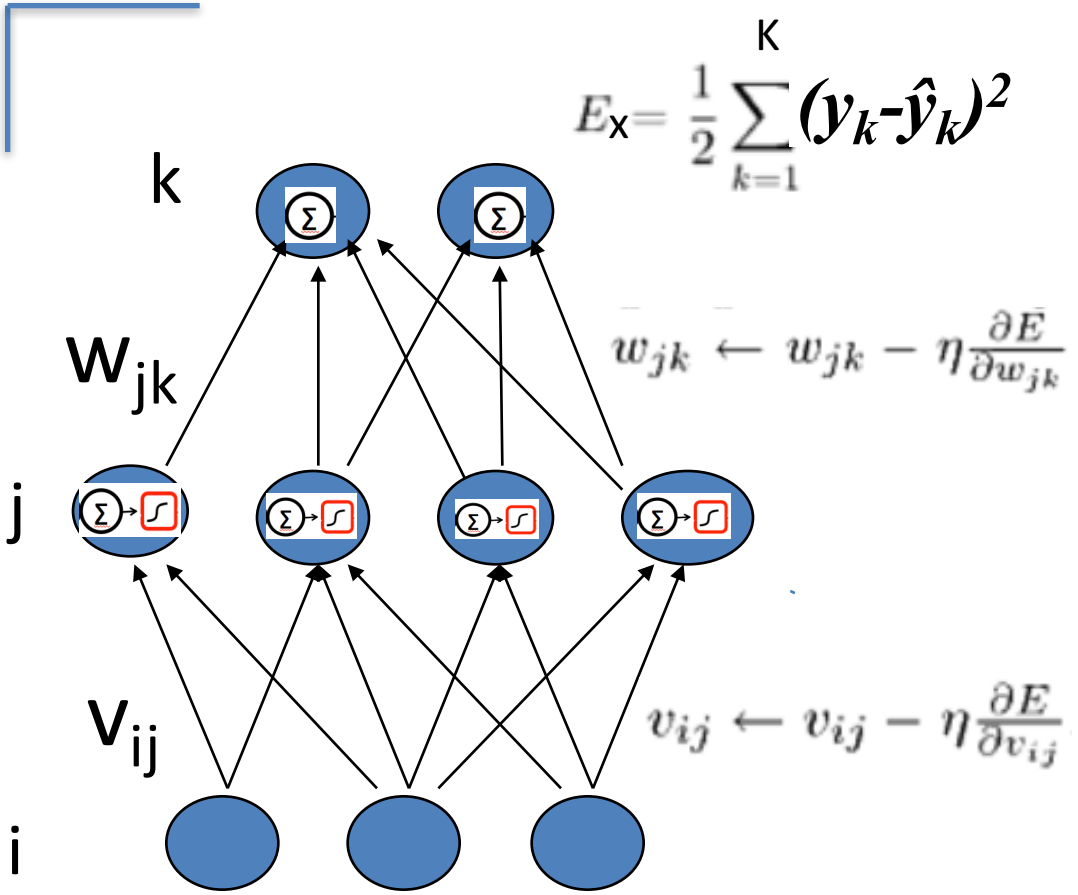
- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - loss, e.g., multi-class classification, softmax layer
  - More about training NN

Multi-Layer Perceptron (MLP) for Regression

Example: 4 layer network with 2 output units:



When for Regression



$$E_x = \frac{1}{2}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

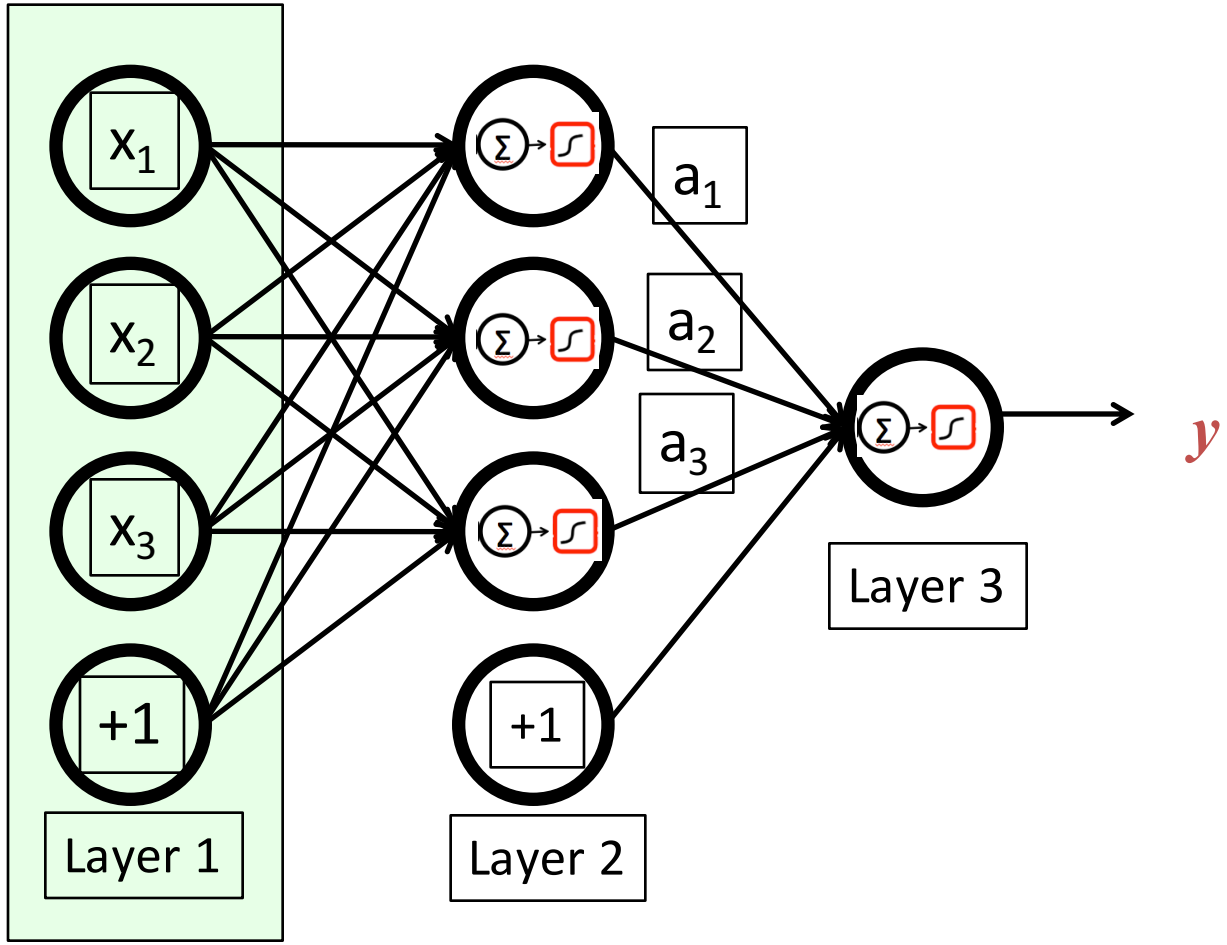
$$w_{jk} \leftarrow w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

$$v_{ij} \leftarrow v_{ij} - \eta \frac{\partial E}{\partial v_{ij}}$$

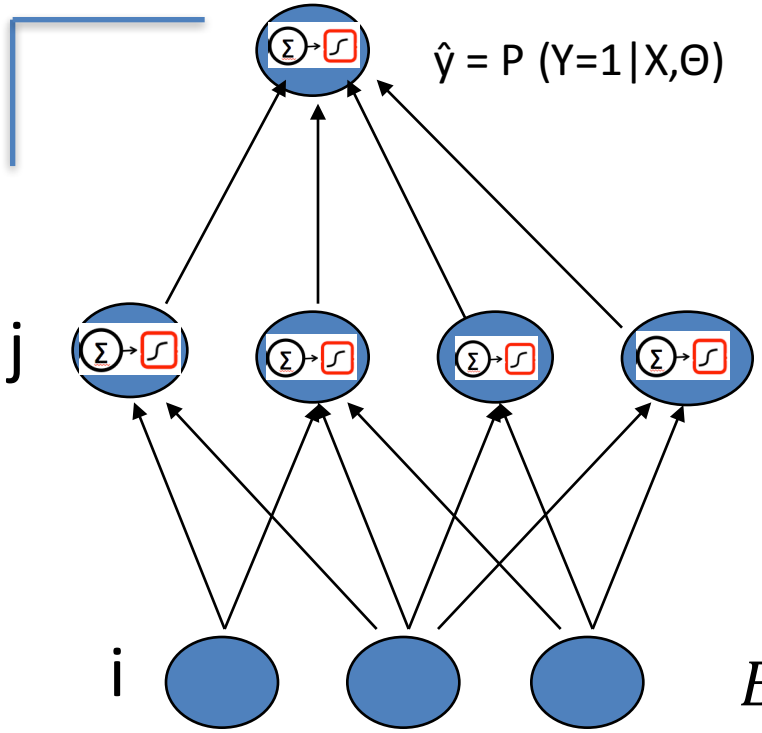
- Training NN in order to minimize the network total sum squared error (SSE).

Multi-Layer Perceptron (MLP) for Binary Classification

String a lot of logistic units together. Example: 3 layer network:



When for binary classification (output layer with 1 neuron for binary output)



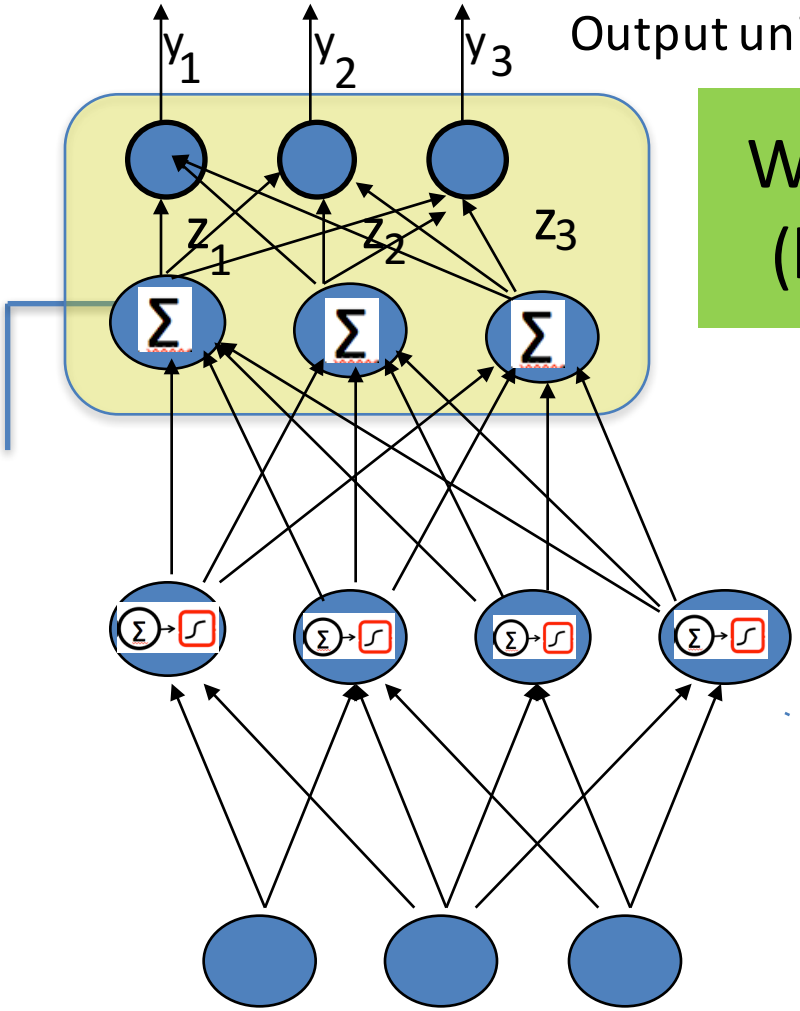
For Bernoulli distribution,

$$p(y=1\,|\,x)^{y}(1-p)^{1-y}$$

$$E_x(\theta) = Loss_x(\theta) = -\log \Pr(Y=y\,|\,X=x)$$

$$= -\{y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\}$$

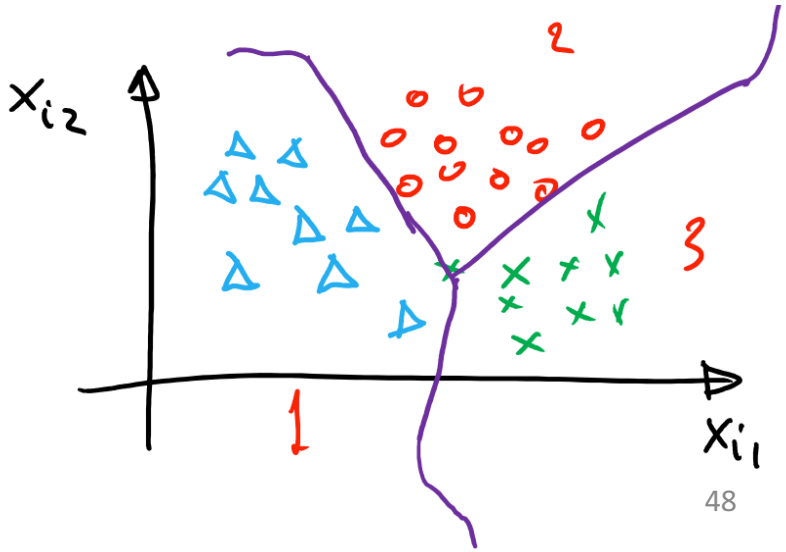
Cross-entropy loss function, OR named as "deviance", OR negative log-likelihood



When for multi-class classification (last output layer: softmax layer)

When multi-class output, last layer is softmax output layer ➔ a Multinoulli logistic regression unit

y_{i1}	y_{i2}	y_{i3}	
0	1	0	class 2
1	0	0	class 1
0	0	1	class 3



Review: Multi-class variable representation

- Multi-class variable ➔

An indicator basis vector representation

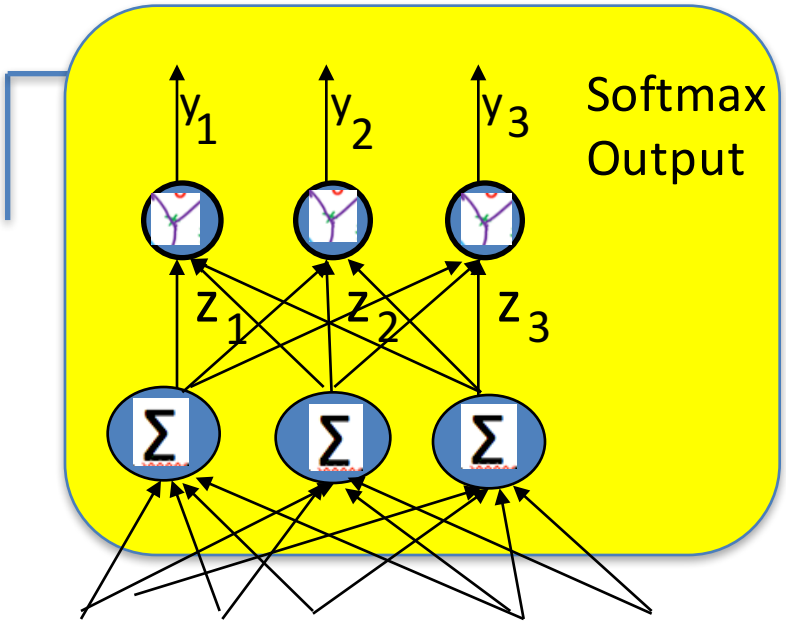
  - If output variable G has K classes, there will be K indicator variable y_i

g	y_1	y_2	y_3	y_4
3	0	0	1	0
1	1	0	0	0
2	0	1	0	0
4	0	0	0	1
1	1	0	0	0

$z_1(x),\ z_2(x), z_3(x),\ z_4(x)$

- How to classify to multi-class ?

Strategy : Use "softmax" layer function for multi-class classification



$$Pr(G = k \mid X = x) = Pr(Y_k = 1 \mid X = x)$$

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

"Softmax" function.
Normalizing function which converts each class output to a probability.

$$\frac{\partial y_i}{\partial z_i} = y_i (1 - y_i)$$

Use "softmax" layer function for multi-class classification

The natural cost function is the negative log probability of the right answer

➔ Cross entropy loss function :

"0" for all except true class

$$E_x(true\vec{y}, \hat{\vec{y}}) = -\sum_{j=1....K} truey_j \ln \hat{y}_j = -\sum_j truey_j \ln p(y_j = 1|x)$$

$$\frac{\partial E}{\partial z_i} = \sum_{j=1....K} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_i} = \hat{y}_i - truey_i$$

Error calculated from Output vs. true

The steepness of function E exactly balances the flatness of the softmax

Logistic: a special case of softmax for two classes

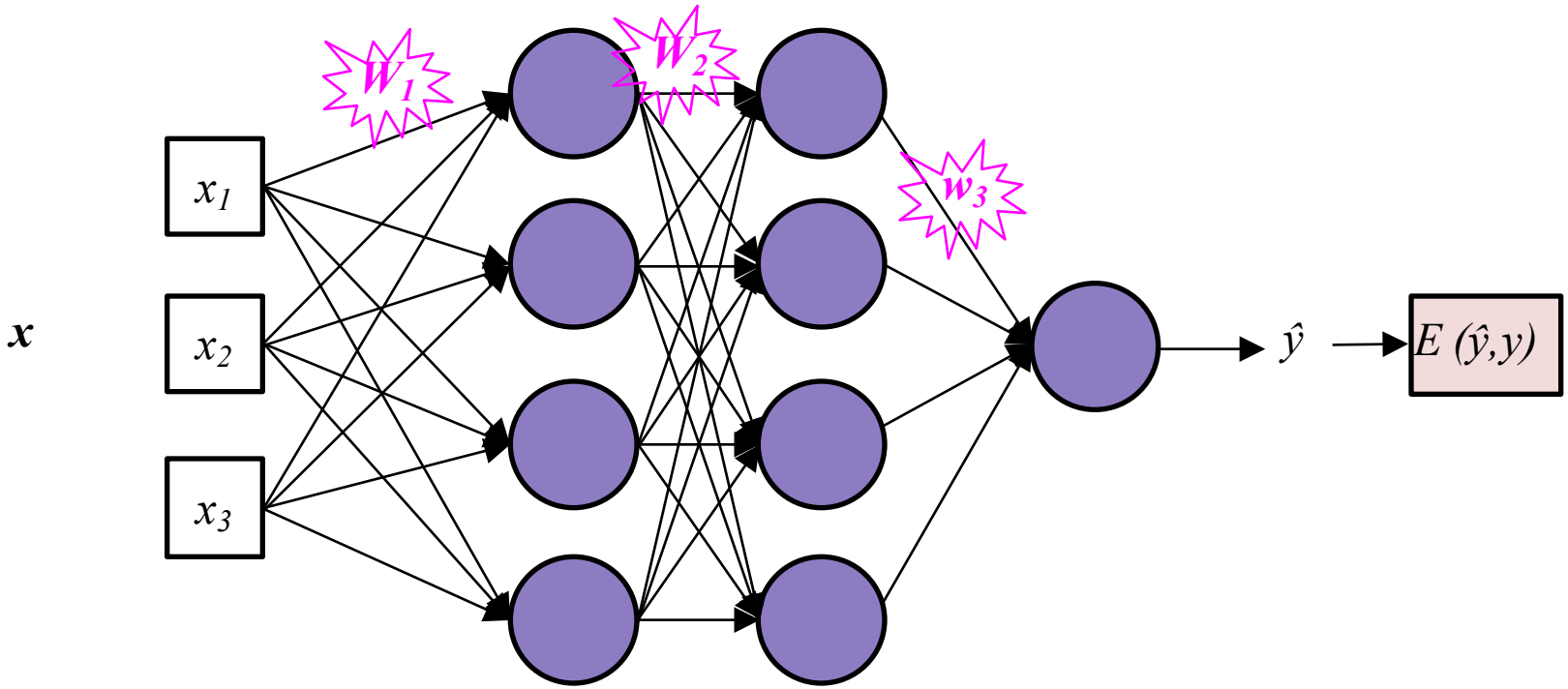
$$y_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_0}} = \frac{1}{1 + e^{-(z_1 - z_0)}}$$

- So the logistic is just a special case that avoids using redundant parameters:
  - Adding the same constant to both z1 and z0 has no effect.
  - The over-parameterization of the softmax is because the probabilities must add to 1.

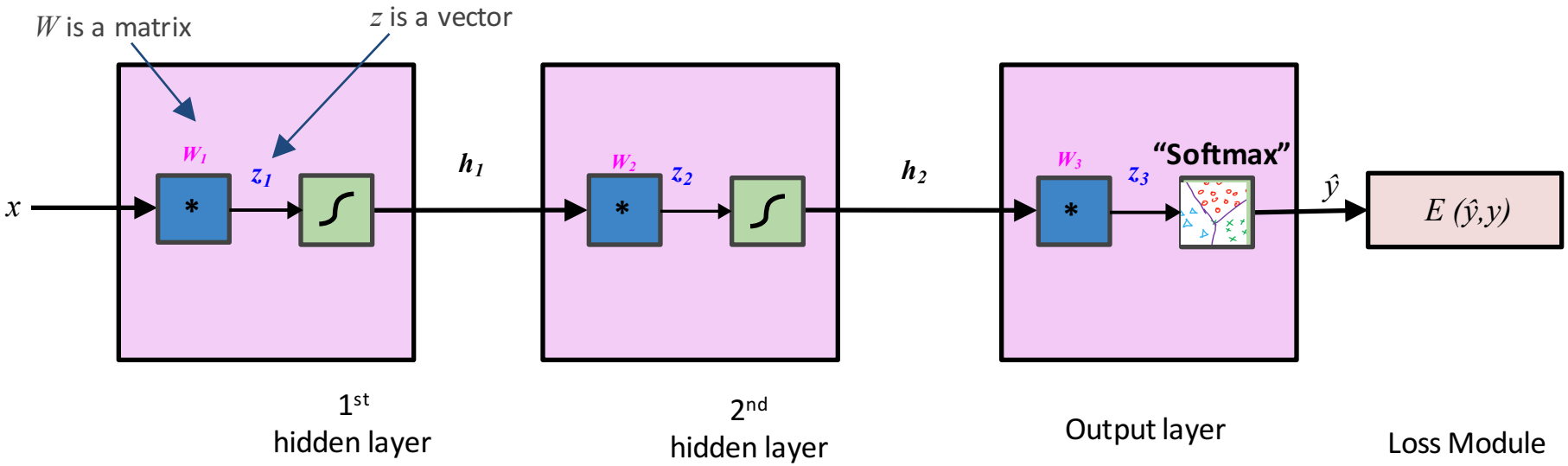
Today

- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

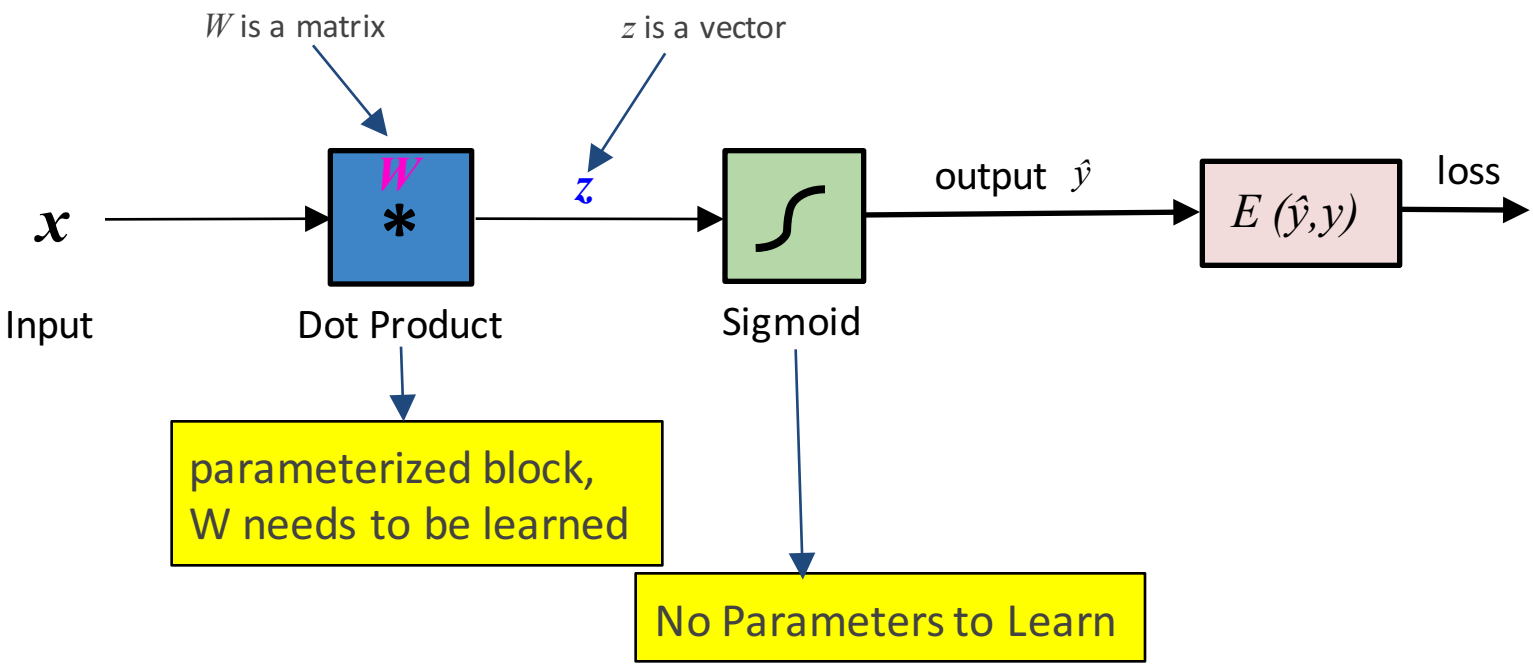
- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN


x
x1
x2
x3
w1
w2
w3
ŷ
E (ŷ,y)

“Block View” of multi-class NN



“Block View” of Logistic Regression



Review: Stochastic GD ➔

- For LR: linear regression, We have the following descent rule:

$$\theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \Big|_t$$

- ➔ For neural network, we have the delta rule

$$\Delta \mathrm{w} = -\eta \frac{\partial E}{\partial W^t}$$

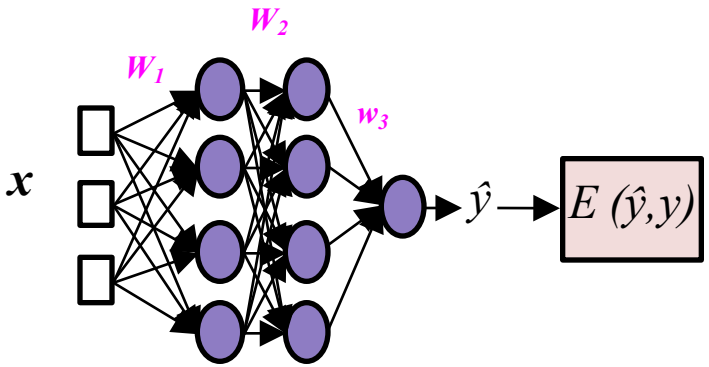
$$W^{t+1} = W^t - \eta \frac{\partial E}{\partial W^t} = W^t + \Delta w$$

Training Neural Networks by **Backpropagation** - to jointly optimize all parameters

How do we learn the optimal weights W_L for our task??

- **Stochastic Gradient descent:**

$$W_L^{t+1} = W_L^t - \eta \frac{\partial E_x}{W_L^t}$$



But how do we get gradients of lower layers?

- **Backpropagation!**
  - Repeated application of chain rule of calculus
  - Locally minimize the objective
  - Requires all "blocks" of the network to be differentiable

LeCun et. al. *Efficient Backpropagation*. 1998

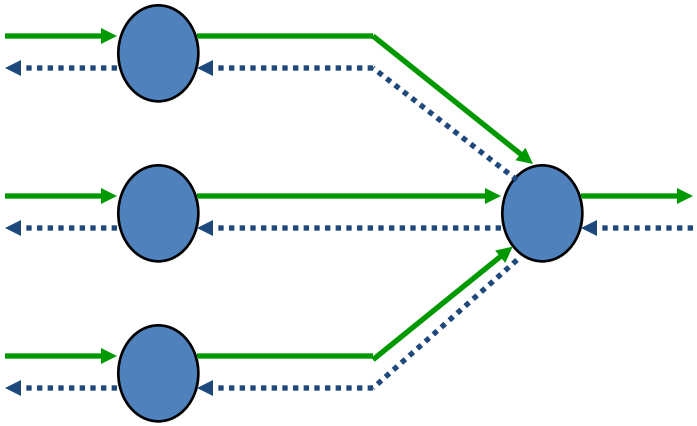
Backpropagation

- 1. Initialize network with random weights
- 2. For all training cases (examples):
  - **a.** Present training inputs to network and calculate output of each layer, and final layer
  - **b.** For all layers (starting with the output layer, back to input layer):
    - i. Compare network output with correct output (error function)
    - ii. Adapt weights in current layer

$$W^{t+1} = W^{t} - \eta \frac{\partial E}{\partial W^{t}}$$

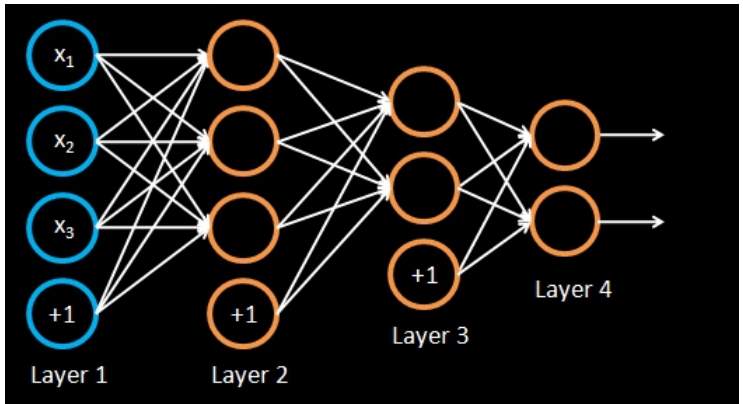
Backpropagation

- Back-propagation training algorithm

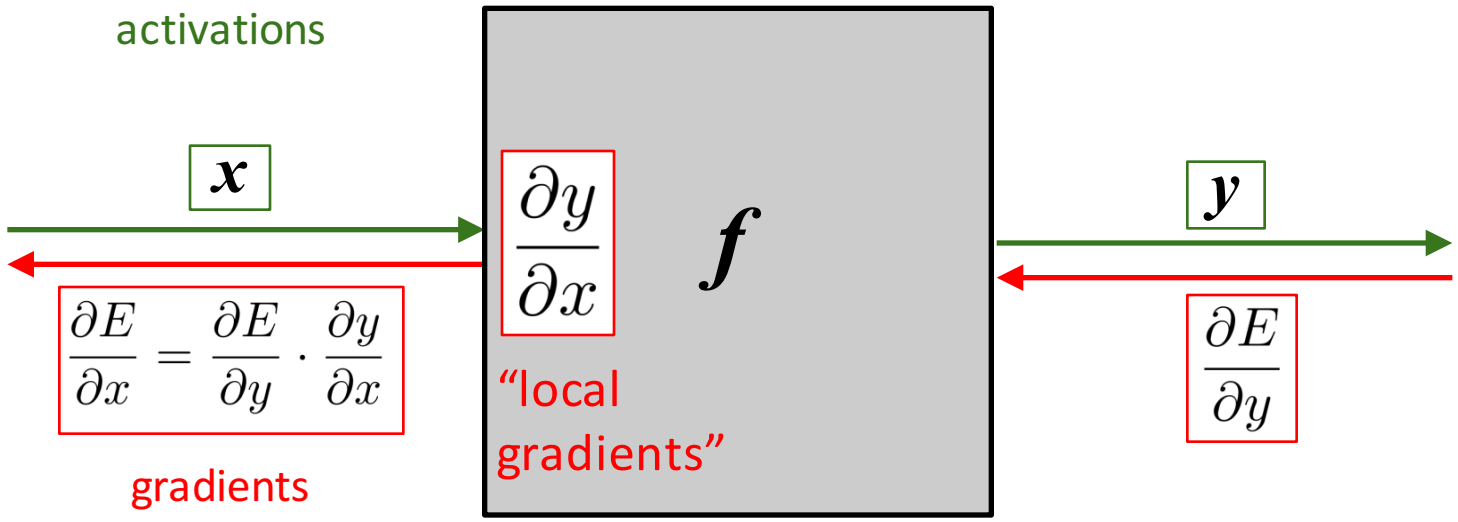


Network activation
Forward Step

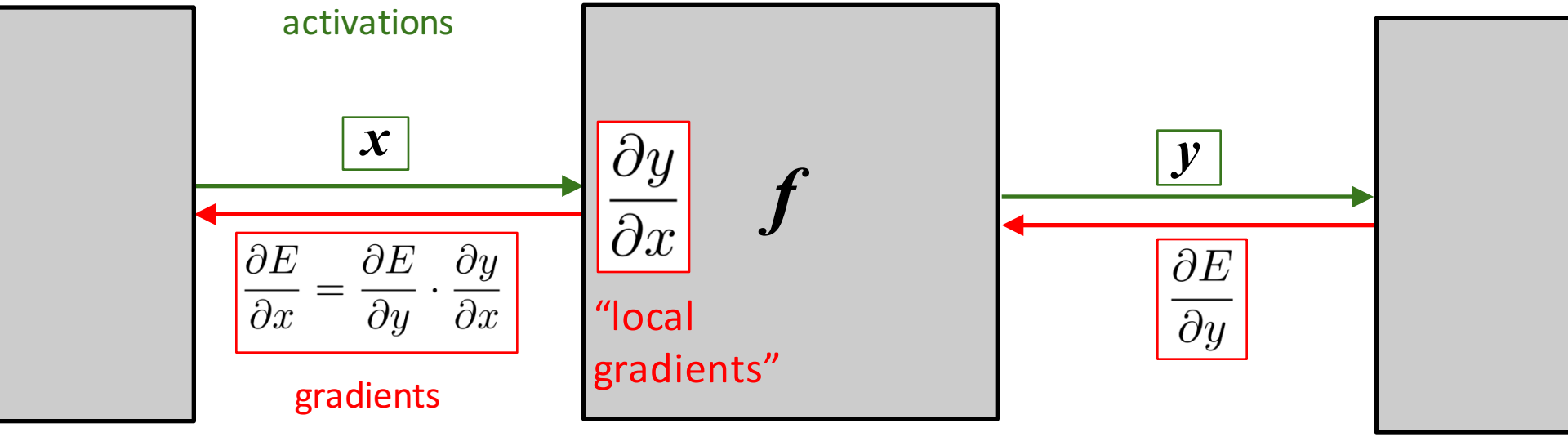
Error propagation
Backward Step



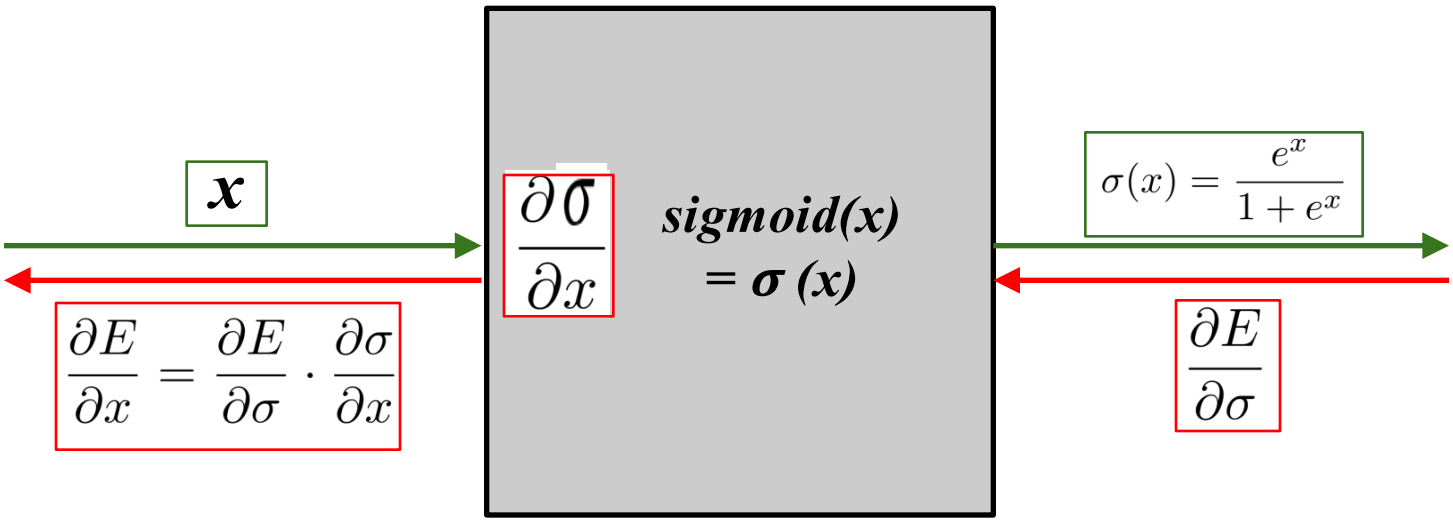
“Local-ness” of Backpropagation



“Local-ness” of Backpropagation

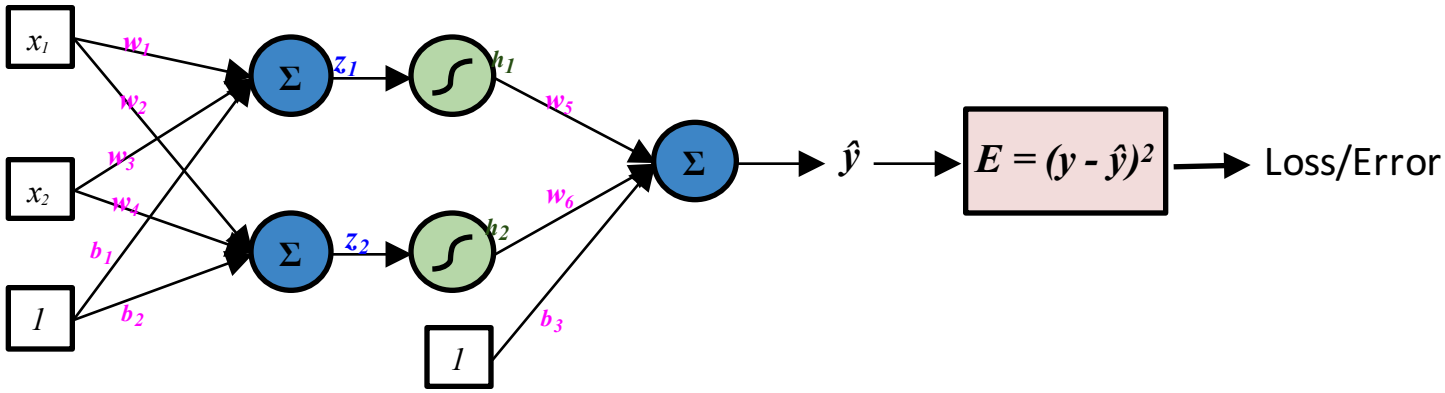


Example: Sigmoid Block



$$\frac{\partial \sigma}{\partial x} = \sigma(1-\sigma)$$

Backpropagation Example



f_1	$z_1 = x_1 w_1 + x_2 w_3 + b_1$ $z_2 = x_1 w_2 + x_2 w_4 + b_2$
f_2	$h_1 = \frac{exp(z_1)}{1 + exp(z_1)}$ $h_2 = \frac{exp(z_2)}{1 + exp(z_2)}$
f_3	$\hat{y} = h_1 w_5 + h_2 w_6 + b_3$
f_4	$E = (y - \hat{y})^2$

$$argmax\_w \quad \{ f_4 (f_3 (f_2 (f_1 ( )))) \}$$

$$\frac{\partial E}{\partial w_5} =$$

$$\frac{\partial E}{\partial w_1} =$$

argmax_w $\{ f_4 (f_3 (f_2 (f_1 (\ )))) \}$

f_1	$z_1 = x_1 w_1 + x_2 w_3 + b_1$ $z_2 = x_1 w_2 + x_2 w_4 + b_2$
f_2	$h_1 = \frac{exp(z_1)}{1 + exp(z_1)}$ $h_2 = \frac{exp(z_2)}{1 + exp(z_2)}$
f_3	$\hat{y} = h_1 w_5 + h_2 w_6 + b_3$
f_4	$E = (y - \hat{y})^2$

x	y	$\partial y / \partial x$
x_1, x_2	z_1, z_2	$\frac{\partial z_1}{\partial x_1} = w_1$
z_1, z_2	h_1, h_2	$\frac{\partial h_1}{\partial z_1} = h_1 (1 - h_1)$
h_1, h_2	$\hat{y}$	$\partial \hat{y} / \partial h_1 = w_5$
$\hat{y}$	loss E	$\partial E / \partial \hat{y} = -2(y - \hat{y})$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial w_1} = \frac{\partial f_4}{\partial f_3} \frac{\partial f_3}{\partial f_2} \frac{\partial f_2}{\partial f_1} \frac{\partial f_1}{\partial w_1}$$

$$= -2(y - \hat{y}) \frac{\partial (h_1 w_5 + h_2 w_6 + b_3)}{\partial w_1}$$

$$= -2(y - \hat{y}) \left(w_5 \frac{\partial h_1}{\partial w_1} + w_6 \frac{\partial h_2}{\partial w_1} \right)$$

$$= -2(y - \hat{y}) w_5 \frac{\partial h_1}{\partial w_1} = -2(y - \hat{y}) w_5 \frac{\partial h_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

$$= \underbrace{-2(y - \hat{y})}_{f_4 \text{ local}} \underbrace{w_5}_{f_3 \text{ local}} \underbrace{h_1 (1 - h_1)}_{f_2 \text{ local}} x_1 \frac{\partial f_1}{\partial w_1}$$

How to choose learning rate:
Review: Linear Regression with Stochastic

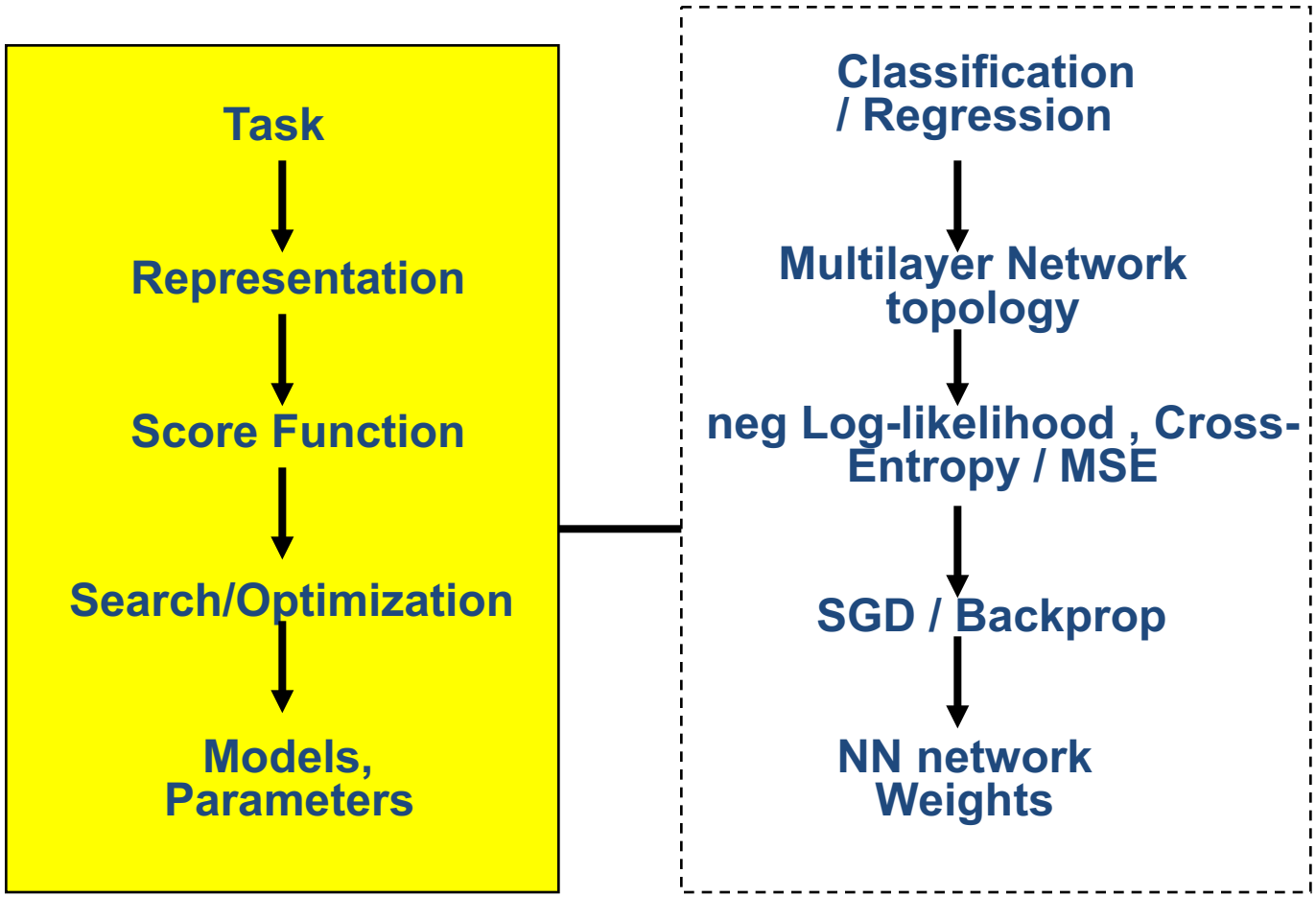
In Linear Regression training with SGD, How do we choose the learning rate α?

- Tuning set, or
- Cross validation, or
- Small value for slow, conservative learning

For DeepNN, how to choose the learning rate ?

- Many other great tools, e.g., Adam
- https://arxiv.org/abs/1609.04747

Basic Neural Network



Today Recap

- Deep Learning
  - History
  - Why breakthrough ?
  - Recent trends (Extra Lecture this Friday!)

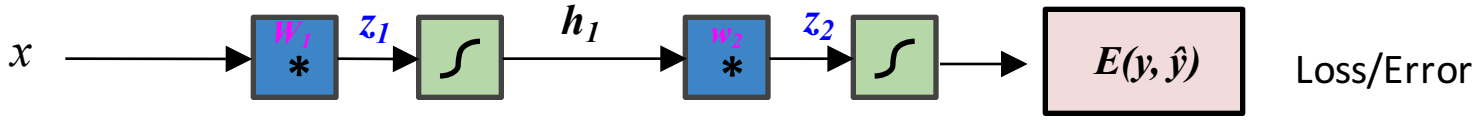
- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN

References

- Dr. Yann Lecun's deep learning tutorials
- Dr. Li Deng's ICML 2014 Deep Learning Tutorial
- Dr. Kai Yu's deep learning tutorial
- Dr. Rob Fergus' deep learning tutorial
- Prof. Nando de Freitas' slides
- Olivier Grisel's talk at Paris Data Geeks / Open World Forum
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.

Another Example of Backpropagation

(binary classification example)



$$E = \text{loss} = -y \ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

Gradient Descent to Minimize loss:

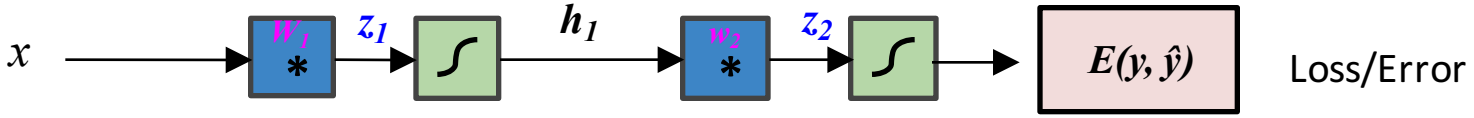
$$\mathbf{w}_2(t+1) = \mathbf{w}_2(t) - \eta \frac{\partial E}{\partial \mathbf{w}_2(t)}$$

$$W_1(t+1) = W_1(t) - \eta \frac{\partial E}{\partial W_1(t)}$$

Need to find these!

Backpropagation

(binary classification example)



$$E = f_4 = -y \ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = f_3 = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = f_3 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

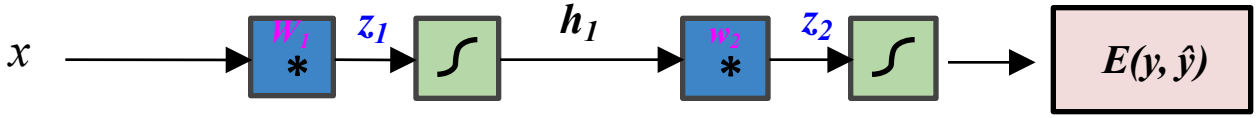
$$\boldsymbol{h}_1 = f_2 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = f_1 = W_1^T \boldsymbol{x}$$

$$E = f_4(f_3(f_2(f_1(x))))$$

Backpropagation

(binary classification example)



$$E = f_4 = -y \ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = f_3 = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = f_3 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

$$\boldsymbol{h}_1 = f_2 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = f_1 = W_1^T \boldsymbol{x}$$

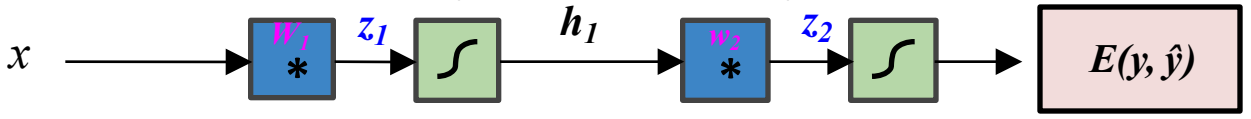
$$\frac{\partial E}{\partial \mathbf{w}_2} = ??$$

$$\frac{\partial E}{\partial \mathbf{W}_1} = ??$$

$$E = f_4(f_3(f_2(f_1(x))))$$

Backpropagation

(binary classification example)



Loss/Error

$$E = f_4 = -y \ln(\hat{y}) - (1-y)\ln(1-\hat{y})$$

$$\hat{y} = f_3 = \frac{e^{z_2}}{1+e^{z_2}}$$

$$z_2 = f_3 = \boldsymbol{w}_2^T \boldsymbol{h}_1$$

$$\boldsymbol{h}_1 = f_2 = \frac{e^{z_1}}{1+e^{z_1}}$$

$$\boldsymbol{z}_1 = f_1 = W_1^T \boldsymbol{x}$$

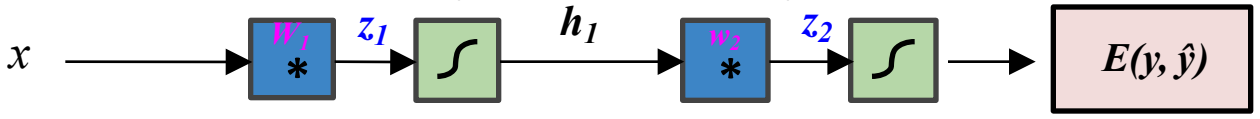
$$\frac{\partial E}{\partial \mathbf{w}_2} = ??$$

$$\frac{\partial E}{\partial \mathbf{W}_1} = ??$$

$$E = f_4(f_3(f_2(f_1(x))))$$ → Exploit the chain rule!

Backpropagation

(binary classification example)

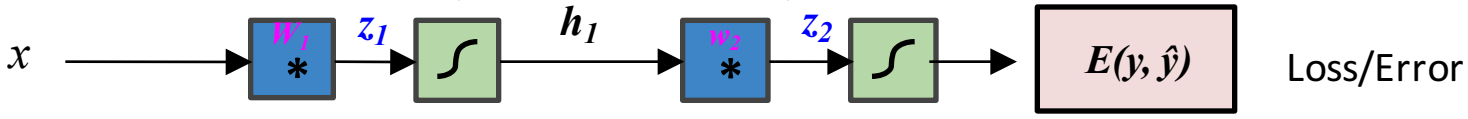


$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$\frac{\partial E}{\partial \boldsymbol{w}_2} =$$

Backpropagation

(binary classification example)



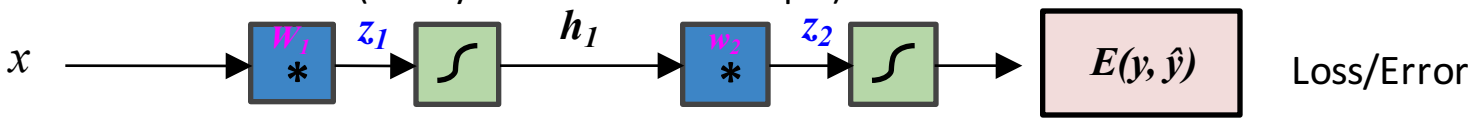
$$
\begin{aligned}
E &= -y\ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

chain rule

$$
\frac{\partial E}{\partial \boldsymbol{w}_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2}
$$

Backpropagation

(binary classification example)

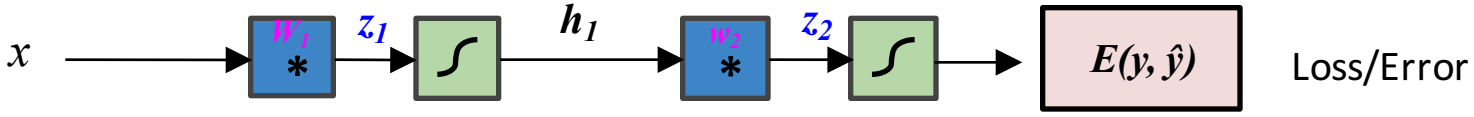


$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}_2} &= \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \right) \cdot
\end{aligned}
$$

Backpropagation

(binary classification example)

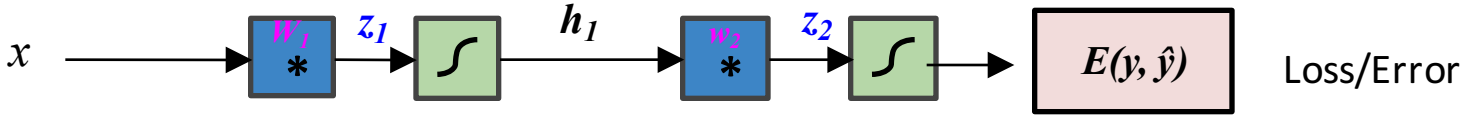


$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}_2} &= \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \right) \cdot \left(\frac{e^{z_2}}{1+e^{z_2}} \left(1 - \frac{e^{z_2}}{1+e^{z_2}} \right) \right) \cdot
\end{aligned}
$$

Backpropagation

(binary classification example)

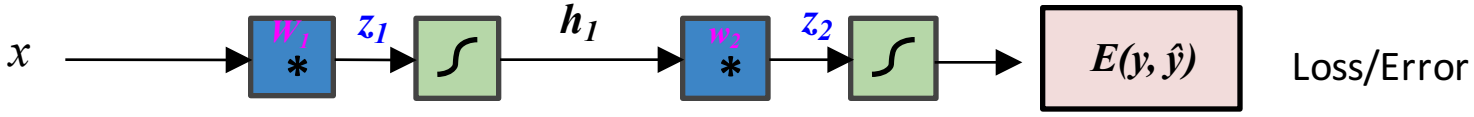


$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}_2} &= \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \right) \cdot \left(\frac{e^{z_2}}{1+e^{z_2}} \left(1 - \frac{e^{z_2}}{1+e^{z_2}} \right) \right) \cdot (\boldsymbol{h}_1)
\end{aligned}
$$

Backpropagation

(binary classification example)

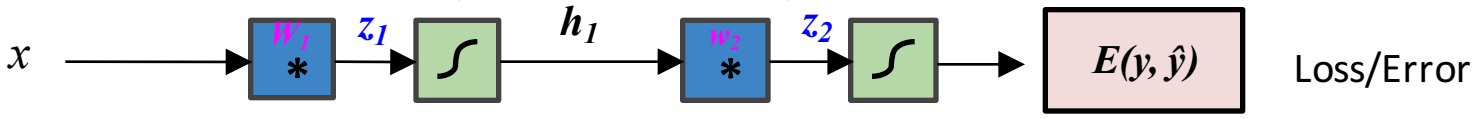


$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \cdot \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \cdot \boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}_2} &= \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{w}_2} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot \left(\frac{e^{z_2}}{1+e^{z_2}}\left(1-\frac{e^{z_2}}{1+e^{z_2}}\right)\right) \cdot (\boldsymbol{h}_1) \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}\right) \cdot (\hat{y}(1-\hat{y})) \cdot (\boldsymbol{h}_1)
\end{aligned}
$$

Backpropagation

(binary classification example)

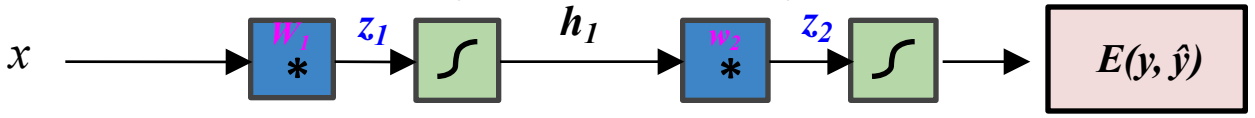


$$
\begin{aligned}
E &= -\,y \ln(\hat{y}) \\
&\quad -\,(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$\frac{\partial E}{\partial \boldsymbol{W}_1} =$$

Backpropagation

(binary classification example)

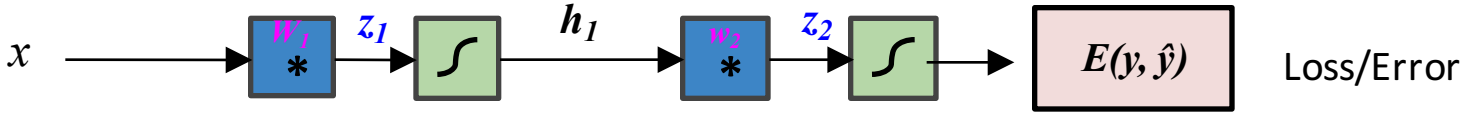


$$
\begin{aligned}
E &= -y\ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$
\frac{\partial E}{\partial \boldsymbol{W}_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial W_1}
$$

Backpropagation

(binary classification example)

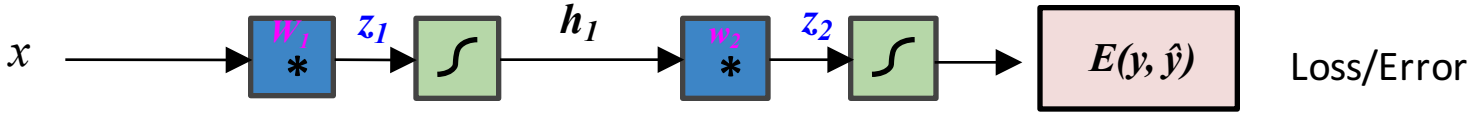


$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{W}_1} &= \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial W_1} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \right) \cdot (\hat{y}(1-\hat{y})) \cdot (\boldsymbol{w}) \cdot (\boldsymbol{h}_1(1-\boldsymbol{h}_1)) \cdot (x)
\end{aligned}
$$

Backpropagation

(binary classification example)



$$
\begin{aligned}
E &= -y \ln(\hat{y}) \\
&\quad -(1-y)\ln(1-\hat{y}) \\
\hat{y} &= \frac{e^{z_2}}{1+e^{z_2}} \\
z_2 &= \boldsymbol{w}_2^T \boldsymbol{h}_1 \\
\boldsymbol{h}_1 &= \frac{e^{z_1}}{1+e^{z_1}} \\
\boldsymbol{z}_1 &= W_1^T \boldsymbol{x}
\end{aligned}
$$

already computed

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{W}_1} &= \boxed{\frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2}} \cdot \frac{\partial z_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial W_1} \\
&= \left(\frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \right) \cdot (\hat{y}(1-\hat{y})) \cdot (\boldsymbol{w}) \cdot (\boldsymbol{h}_1(1-\boldsymbol{h}_1)) \cdot (x)
\end{aligned}
$$