

What is Interaction + AI / ML / DL

AI-based HRI

2024 Fall Semester

AI + HRI

Robots?



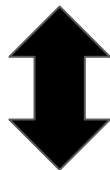
Human has a curiosity about how to understand and simulate human beings.

It would be in our nature to desire making intelligent mobile artifact, robots.



Robots: Definitions

“A reprogrammable, multi-functional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions for performance of variety of tasks” (Russell & Norvig, 1995)



“An automatic device that performs functions normally ascribed to humans or a machine in the form of a human” (Merriam Webster’s collegiate dictionary, 1993)

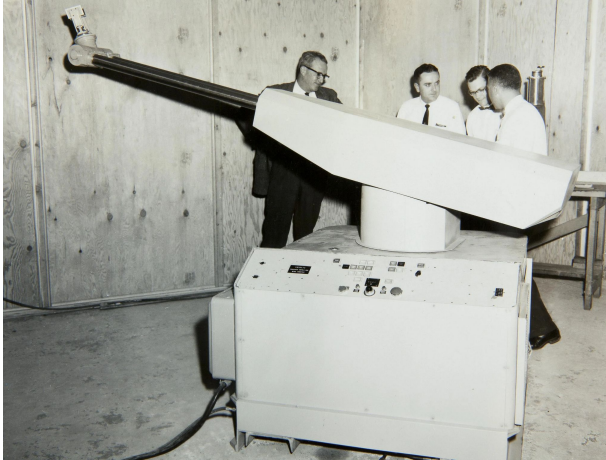
Even if detailed definition might differ, people will agree on below:

Robots inhabit same physical space as people do, and manipulate human artifacts.

Robots: Category by U.N.

1. Industrial Robots

- Having a fuzzy boundary with non-robotic manufacturing device.
- Usually has a multiple actuated elements, arranged in changes (arm).
- Starts from early of 1960 (Unimate), and revolution happens in the early of 1970 (Nissan).
- Interact less directly with human.
 - Of course, there is a research field focusing on industrial robot configuring and programming easier.



Robots: Category by U.N.

2. Professional Service Robots

- Having a professional goals, outside industrial settings.
 - Hospital, Surgical robots.
 - Robots in chemical or biological labs, operating in proper speed and precision.
- Work in environments inaccessible to people:
 - Cleaning nuclear waste, navigate abandoned mines.



Robots: Category by U.N.

3. Personal Service Robot

- Assist or entertain people in domestic or recreational activities.
 - Domestic: Vacuum cleaners, lawnmowers, receptionist, assistants.
 - Recreational: Toy robot, Soccer playing robot.
- People with less knowledge with robot also can operate.
- Finding effective means of interaction is more crucial than in industrial or professional service robots.
- Having an autonomy can be more difficult than other expensive robots.
 - Autonomy: robot's ability to accommodate variations in its environment.



What is Human-Robot Interaction (HRI)?

[Definition and Category by Michael A. Goodrich and Alan C. Schultz]

A field of study for understanding, designing, and evaluating robotic systems for use by or with humans.

Interaction is a communication between robots and humans, which has several forms with respect to proximity.

1. Remote Interaction (so-called tele-operation)

- The human and the robot are not co-located and are separated spatially.
- Or even temporally (i.e., Mars Rovers separated from earth both in space and time)

2. Proximate Interaction

- The humans and robots are co-located.

HRI Applications can be categorized based on requirements:

Does it require mobility?

Does it require physical manipulation?

Does it require social interaction?

What is Human-Robot Interaction (HRI)?

[Category by K. Dautenhahn]

1. Robot-centred HRI

- Assume a robot as a creature, an autonomous entity with its own goal, motivation and emotions.
- For robot, interaction with people is for “fulfilling its internal needs”

2. Human-centred HRI

- Focus on how a robot can fulfill its task while making it acceptable and comfortable to humans.
- Study how people would react and interpret robot’s behavior or appearance
 - Regardless of its robot’s architecture or inside cognitive processes.
- How to design robot behavior or appearance, how to measure the quality of interaction, avoiding ‘uncanny valley’, could be relevant challenges to be solved.

3. Robot Cognition-centred HRI

- Emphasizes the robot as an intelligent system.
- A robot should make decisions on its own, solves problems in a particular application domain.

What is Human-Robot Interaction (HRI)?

[Category by S. Thrun]

1. Indirect Interaction

- Happens when a human operates a robot with a command.
 - Robot just receive the command, and execute.
- Not bi-directional.
- Professional service robots are expect to be indirect.

2. Direct Interaction

- Robot acts on its own motivation, such that human and robot are in “equal footing”.
 - Robot can also initiate the interaction.
- Personal service robots are expect to be direct.
 - Except for vacuum cleaners.
- Interfaces such as screens, speech module will be required + gesture/gaze recognition

Open Question: Is “Social Robot” mandatory?

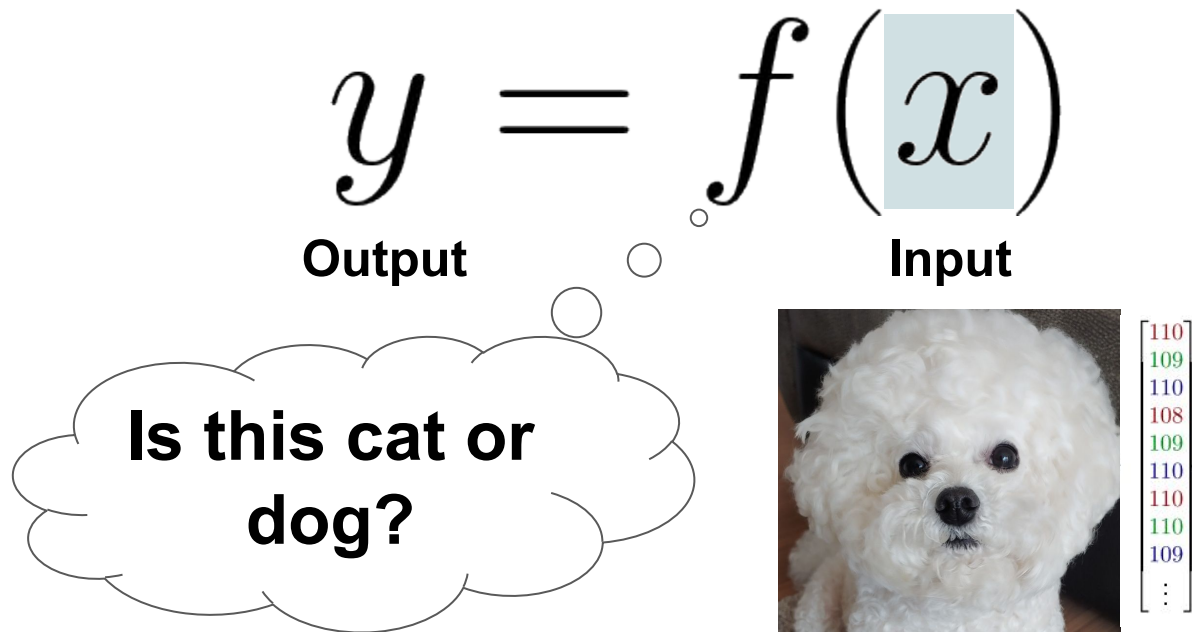
- With sufficient abilities and natural interface for people, it is expected that human will use their instinctive and culturally developed subconscious techniques for communicating with other people, to communicate with robots.
- Reeves & Nass (1996) shows that humans tend to treat computers in certain ways as people, applying social rules and heuristics from the domain of people to the domain of machines.
- However, it is not generally accepted that robot’s social skills are more than a necessary ‘add-on’, to make robot more ‘attractive’ to human interacting with it.
- Research on intelligent robots usually focuses on giving robots planning, reasoning, navigation, manipulation, and other skill-related knowledges in “non-social” environment.
 - Later, they add ‘social skills’ and other aspects of social cognition.

References in this chapter

- [1] Brooks, Rodney A., et al. "Technologies for human/humanoid natural interactions." *The 2nd International Symposium in HUmAnoid RObots (HURO'99), Tokyo, Japan*. 1999.
- [2] Dautenhahn, Kerstin. "Socially intelligent robots: dimensions of human–robot interaction." *Philosophical transactions of the royal society B: Biological sciences* 362.1480 (2007): 679-704.
- [3] Thrun, Sebastian. "Toward a framework for human-robot interaction." *Human–Computer Interaction* 19.1-2 (2004): 9-24.
- [4] Goodrich, Michael A., and Alan C. Schultz. "Human–robot interaction: a survey." *Foundations and Trends® in Human–Computer Interaction* 1.3 (2008): 203-275.

AI + HRI

AI as a “function”

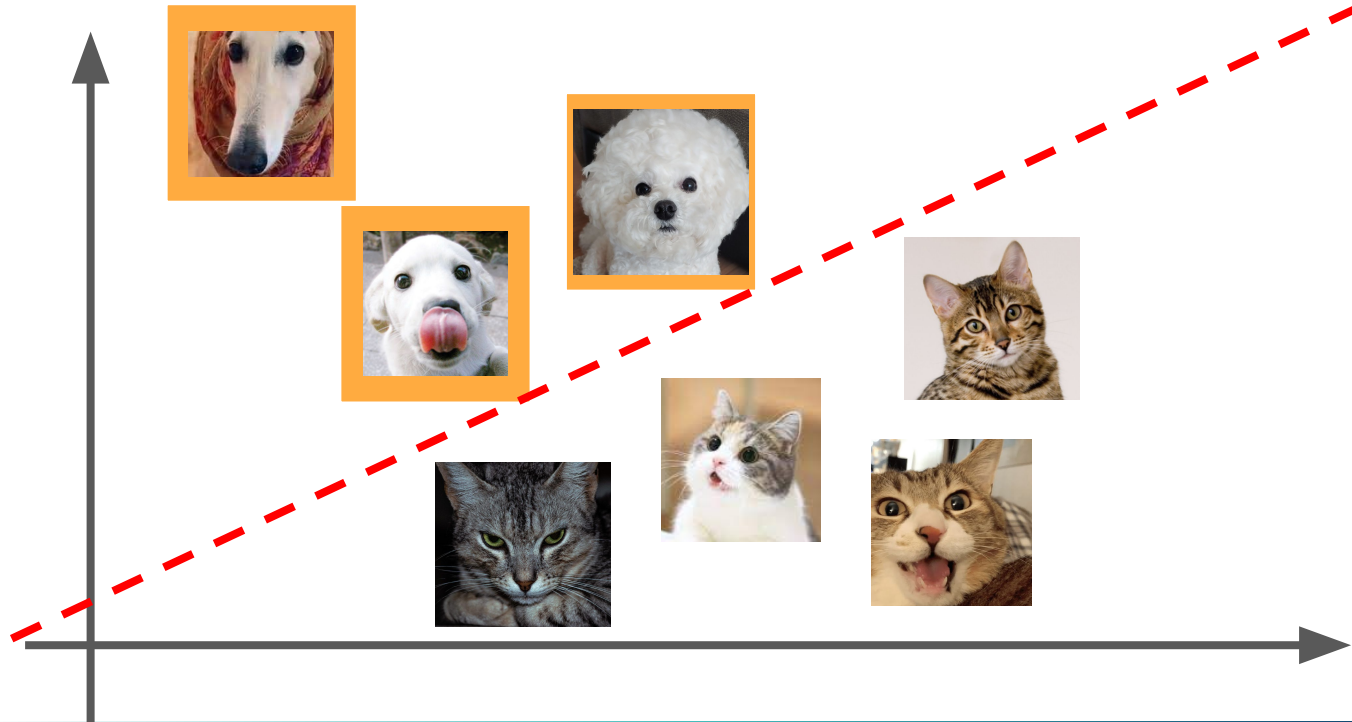


AI as a “function”



AI as a “function”

$$y = f(x) = wx + b$$

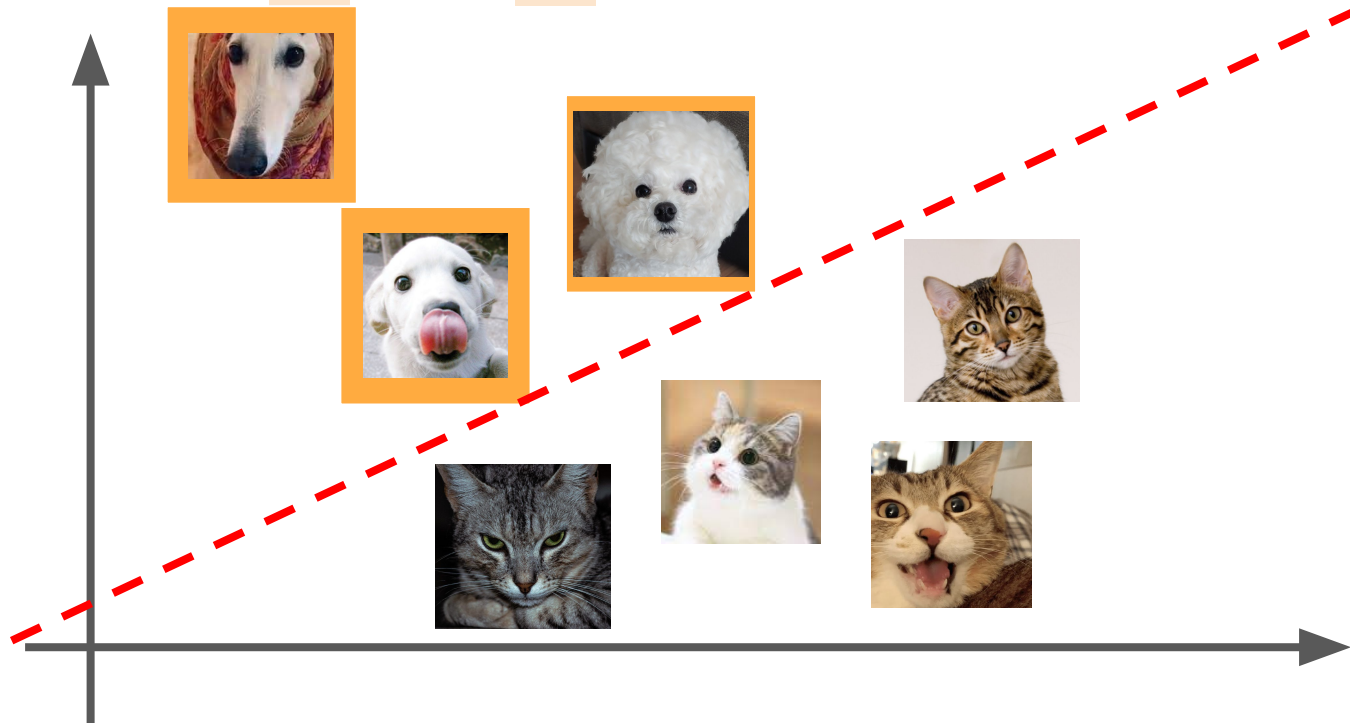


AI as a “function”

Trainable Parameters

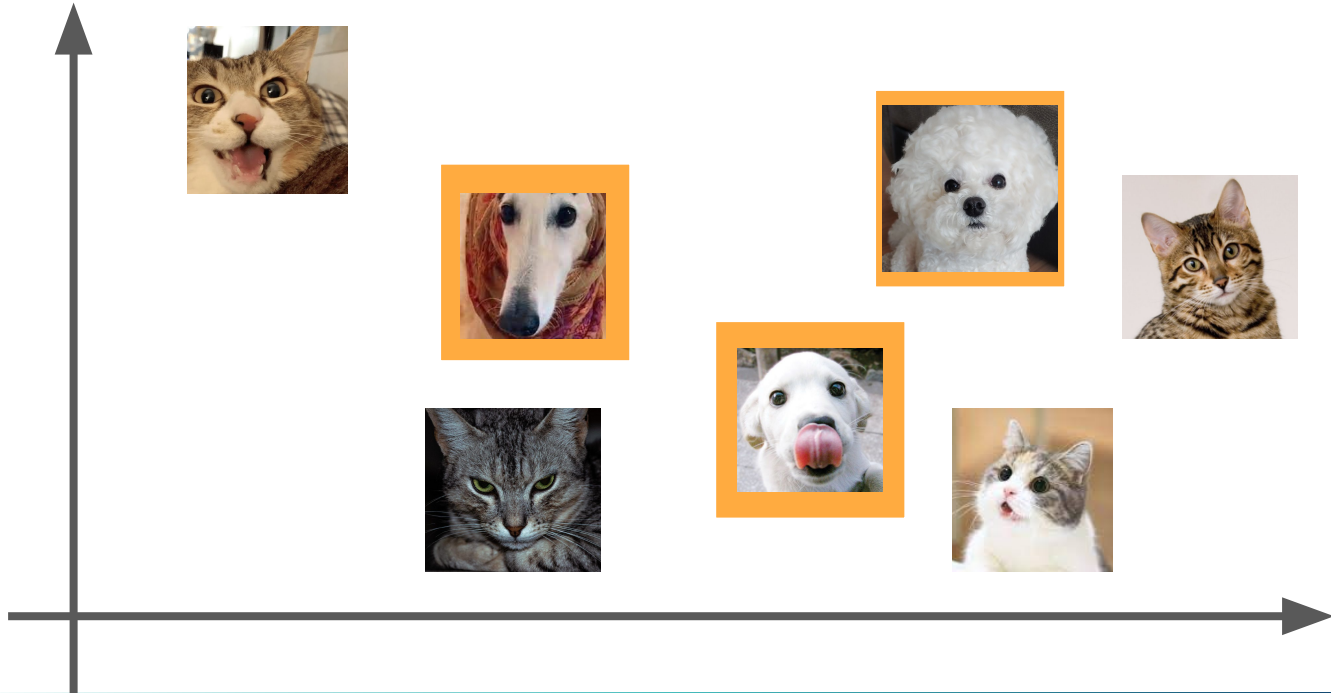
Objective is to find “optimal” parameter

$$y = f(x) = wx + b$$



AI as a “function”

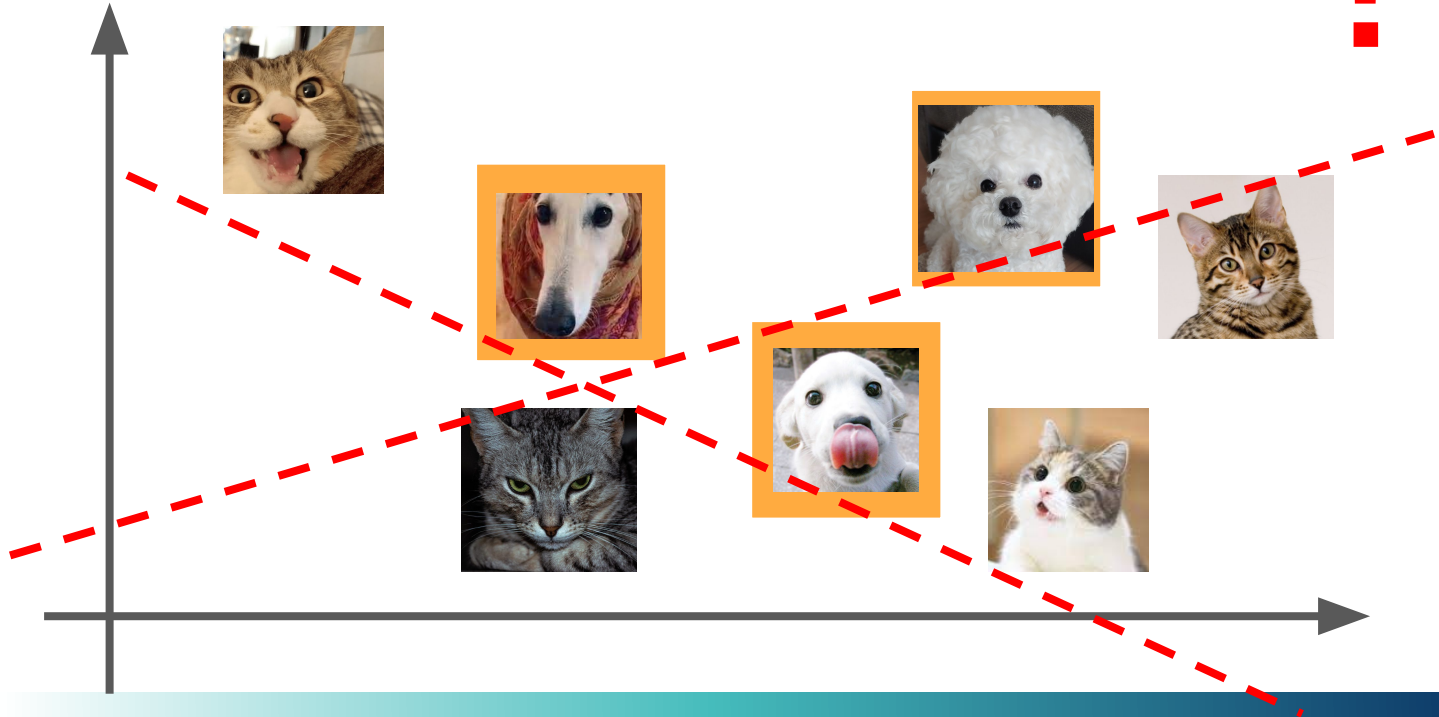
$$y = f(x) = wx + b$$



AI as a “function”

$$y = f(x) = wx + b$$

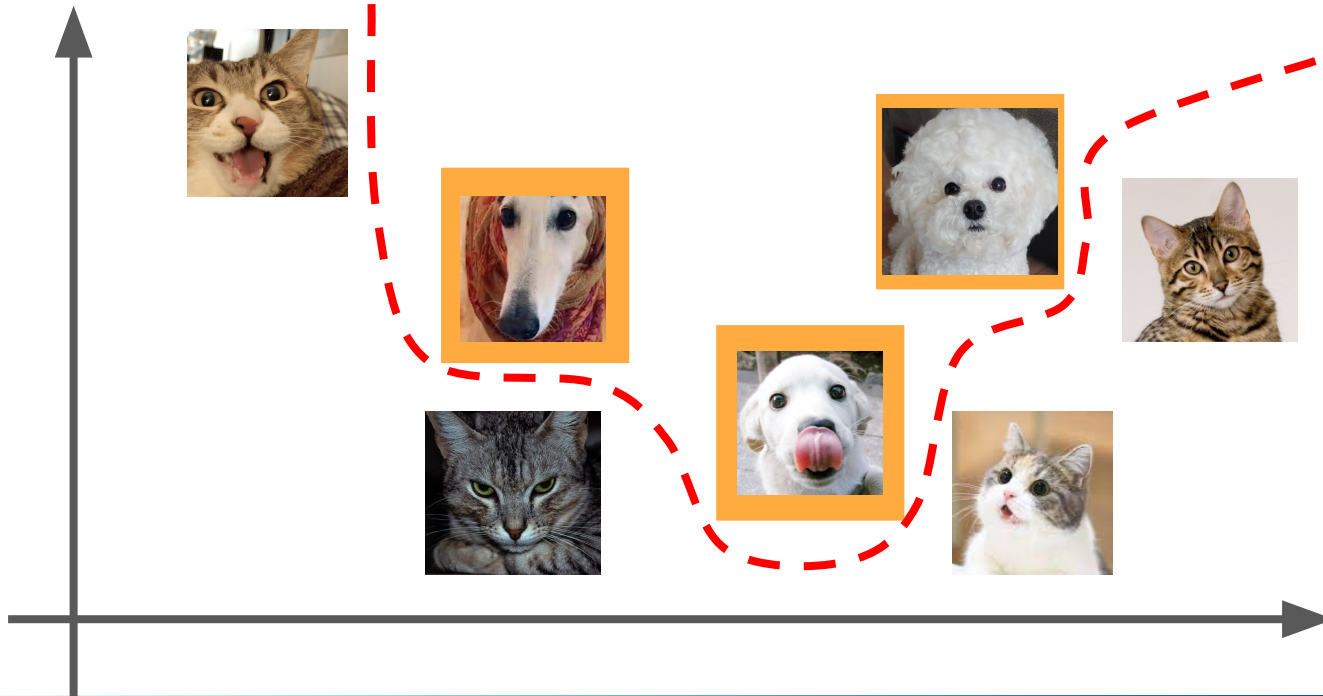
?



AI as a “function”

Engineer can design the relationship between input, output, and parameter

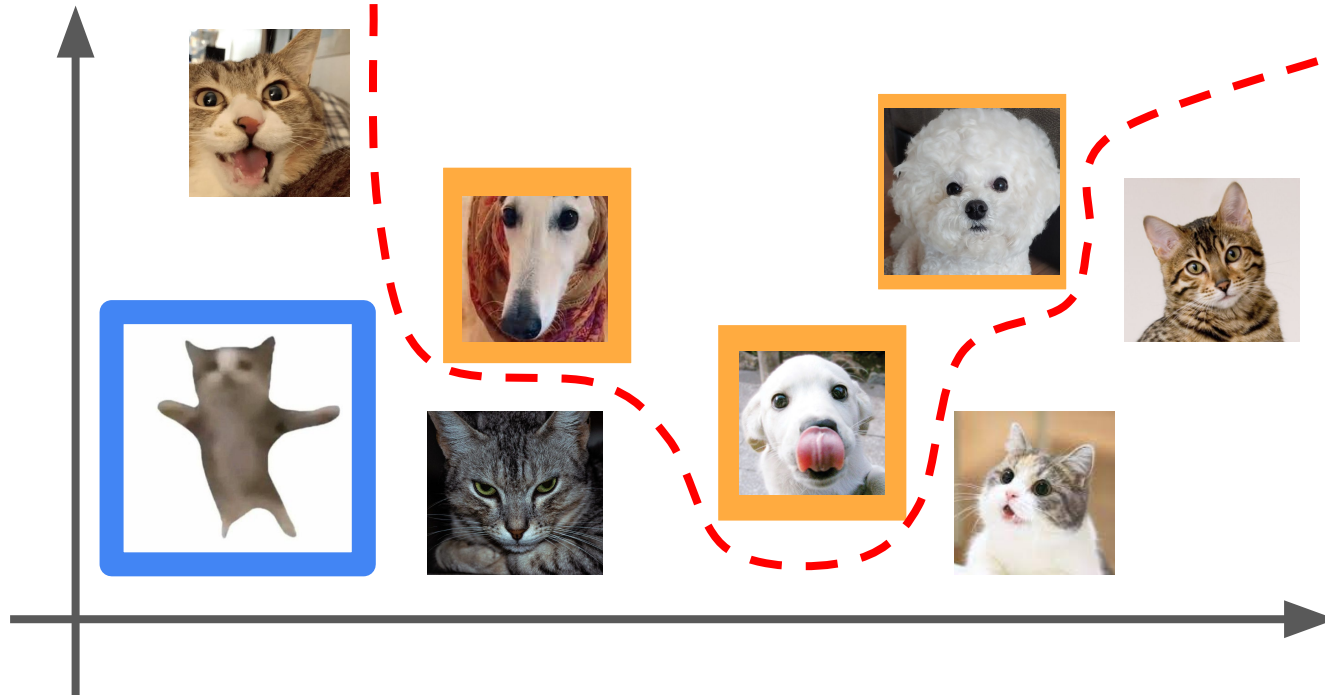
$$y = f(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + \dots$$



AI as a “function”

Engineer can design the relationship between input, output, and parameter

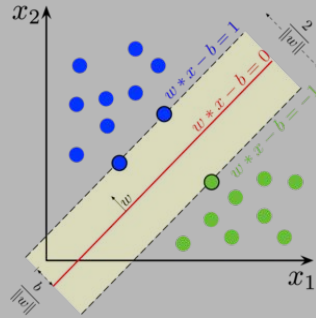
$$y = f(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + \dots$$



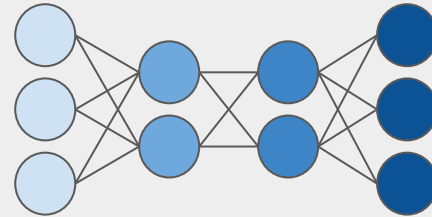
Artificial Intelligence & Deep Learning

Artificial Intelligence

Machine Learning (SVM, Random Forest..)

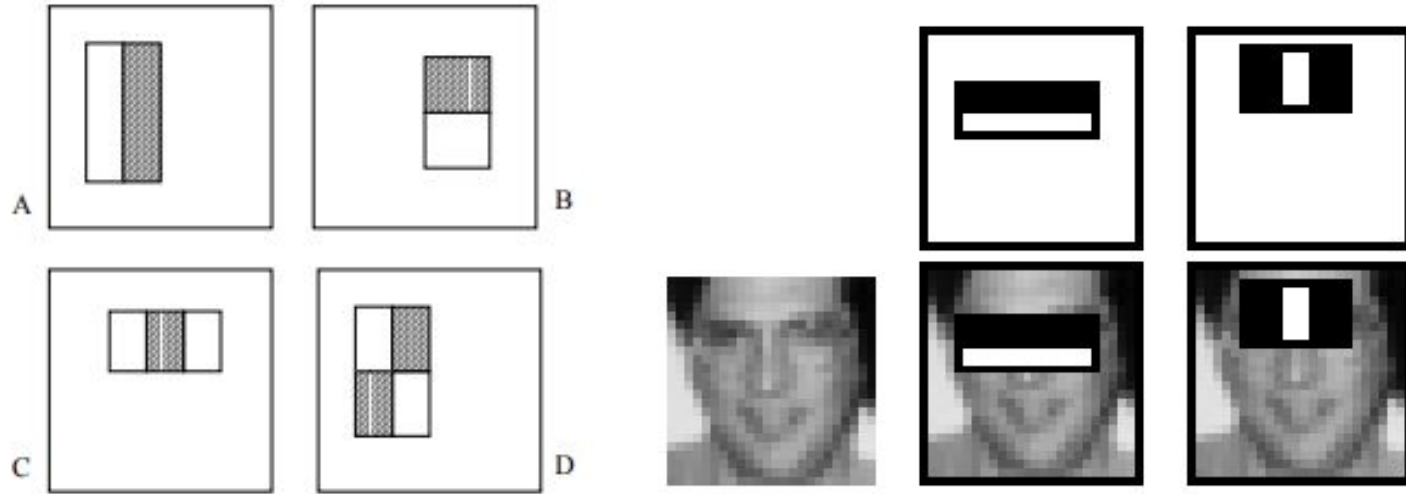


Deep Learning (MLP, CNN, RNN, ...)



What makes Deep Learning Special?

- Example: Feature Engineering for Face Detection



Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. IEEE, 2001.

What makes Deep Learning Special?

- Example: Face detection result, on about 23 years ago.

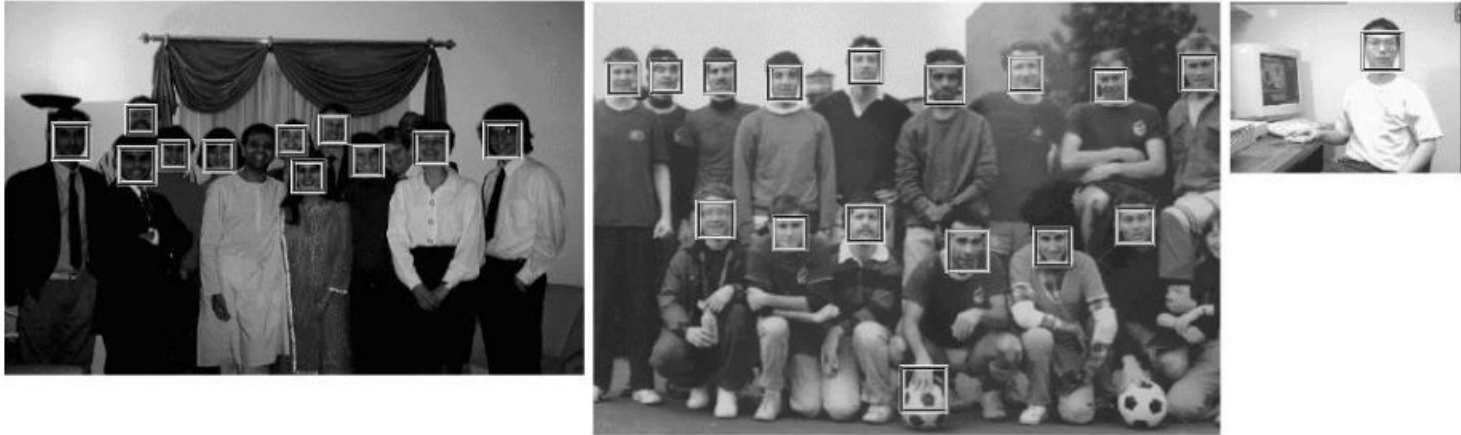


Figure 7: Output of our face detector on a number of test images from the MIT+CMU test set.

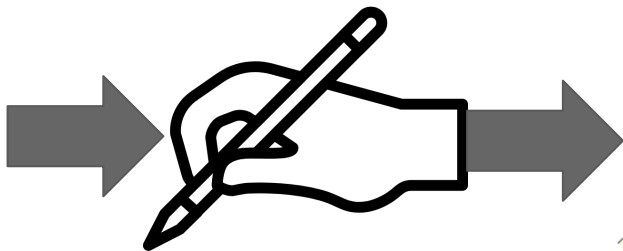
Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. IEEE, 2001.

What makes Deep Learning Special?

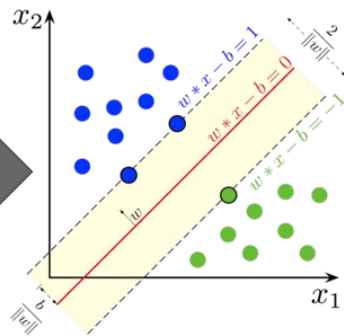
- From Feature Engineering to Feature Learning



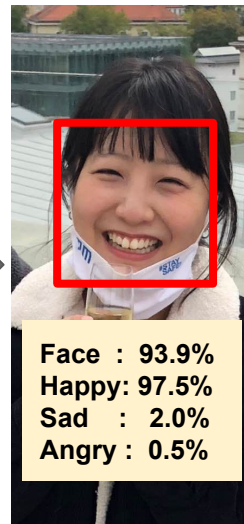
Input Data



Hand-Crafted Features



Trained Classifier



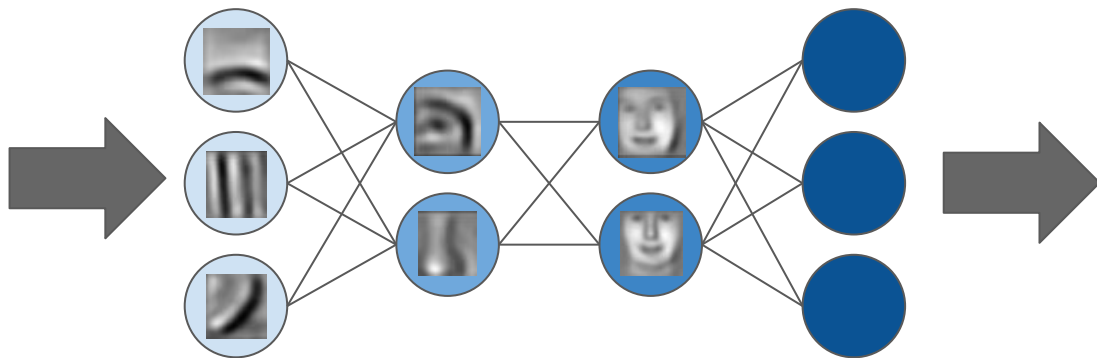
Get Results

What makes Deep Learning Special?

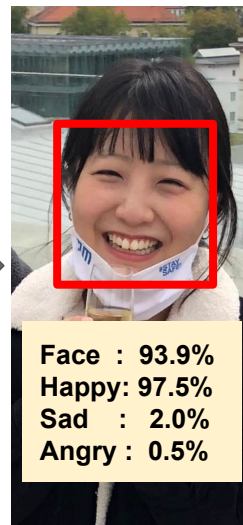
- From Feature Engineering to Feature Learning



Input Data



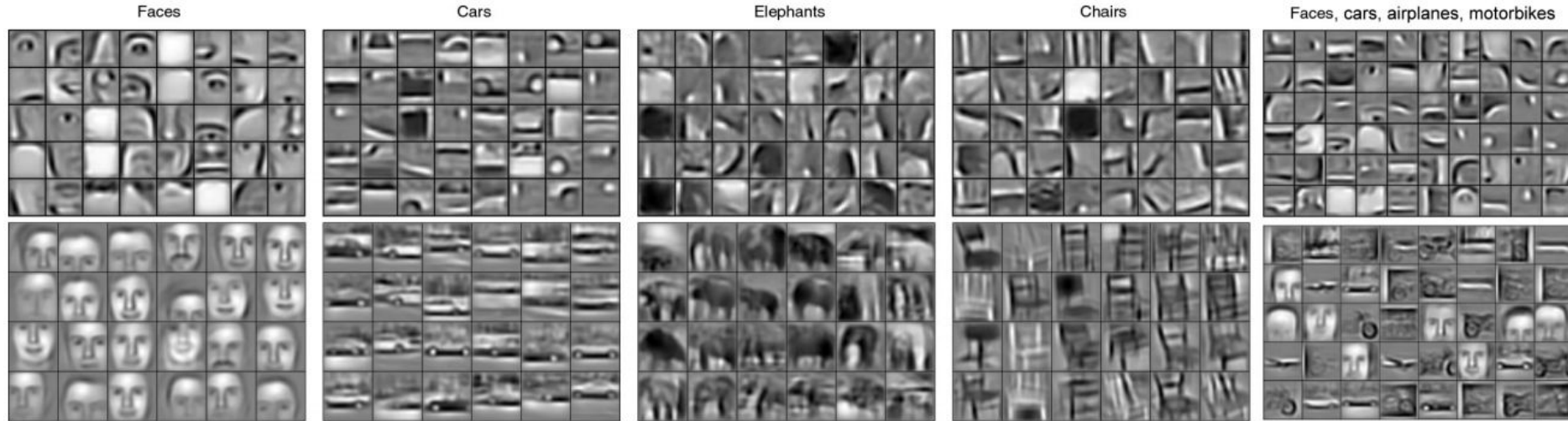
Trained Neural Network Classifier with Learned Features



Get Results

What makes Deep Learning Special?

- Example of Learned Features



Lee, Honglak, et al. "Unsupervised learning of hierarchical representations with convolutional deep belief networks." *Communications of the ACM* 54.10 (2011): 95-103.















APA

What makes Deep Learning Special?

Face Detection on PASCAL Face

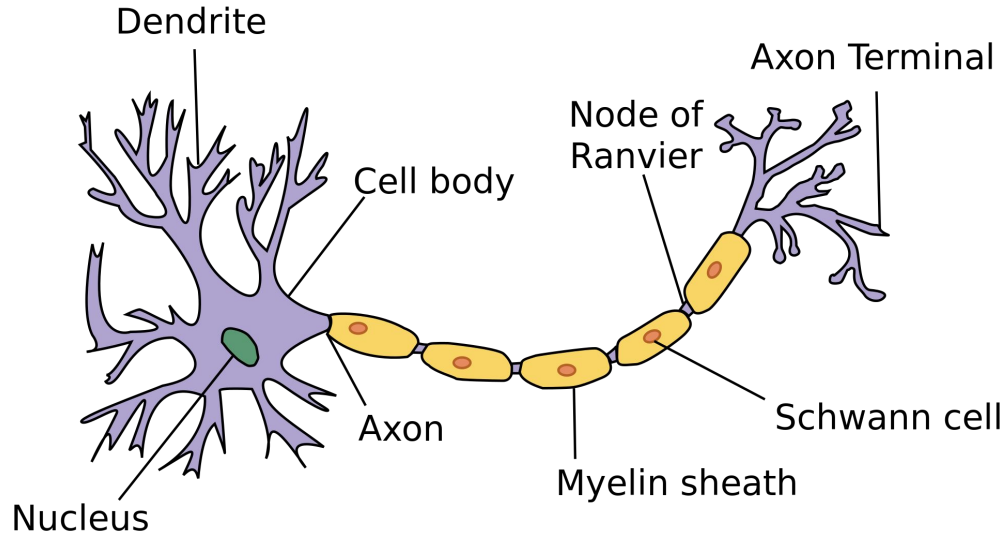


What makes Deep Learning Special?

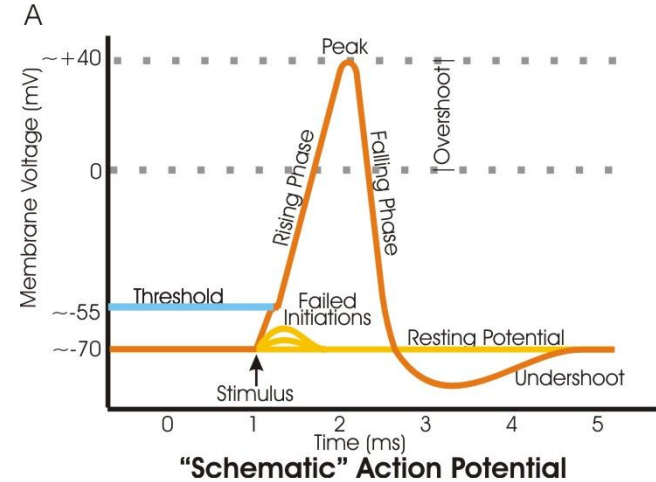
RANK	MODEL	AP 	PAPER	CODE	RESULT	YEAR
1	SRN	0.9909	Selective Refinement Network for High Performance Face Detection			2018
2	Anchor-based	0.990	Robust Face Detection via Learning Small Faces on Hard Images			2018
3	S3FD	0.9849	S³FD: Single Shot Scale-invariant Face Detector			2017
4	FaceBoxes	0.9630	FaceBoxes: A CPU Real-time Face Detector with High Accuracy			2017
5	HyperFace-ResNet	0.9620	HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition			2016
6	STN	0.9410	Supervised Transformer Network for Efficient Face Detection			2016
7	DPM	0.9029	A Fast and Accurate Unconstrained Face Detector			2014

Artificial “Neural” Network

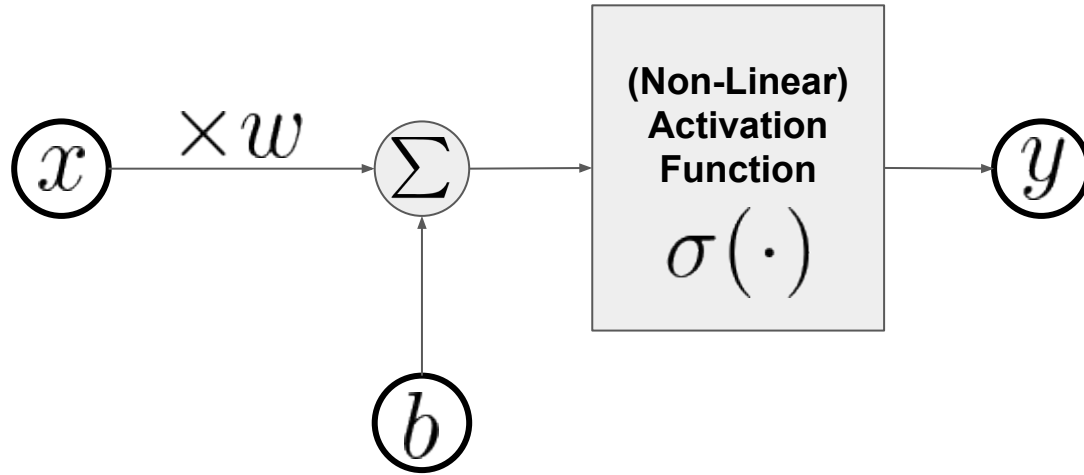
How human’s neuron works?



Images from wikipedia



Artificial Neural Networks - Perceptrons



$x \in \mathbb{R}$: Input

$y \in \mathbb{R}$: Output

$w \in \mathbb{R}$: Weight

$b \in \mathbb{R}$: Bias

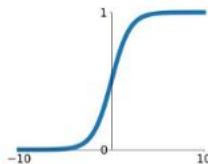
$$y = \sigma(wx + b)$$

Activation Functions

What are activation functions? : Introducing non-linearity!

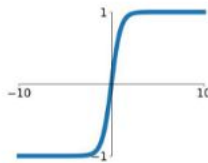
Sigmoid

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



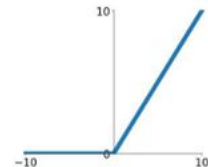
tanh

$$\tanh(x)$$



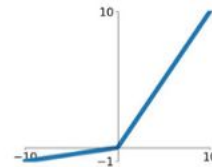
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

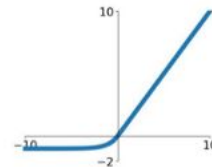
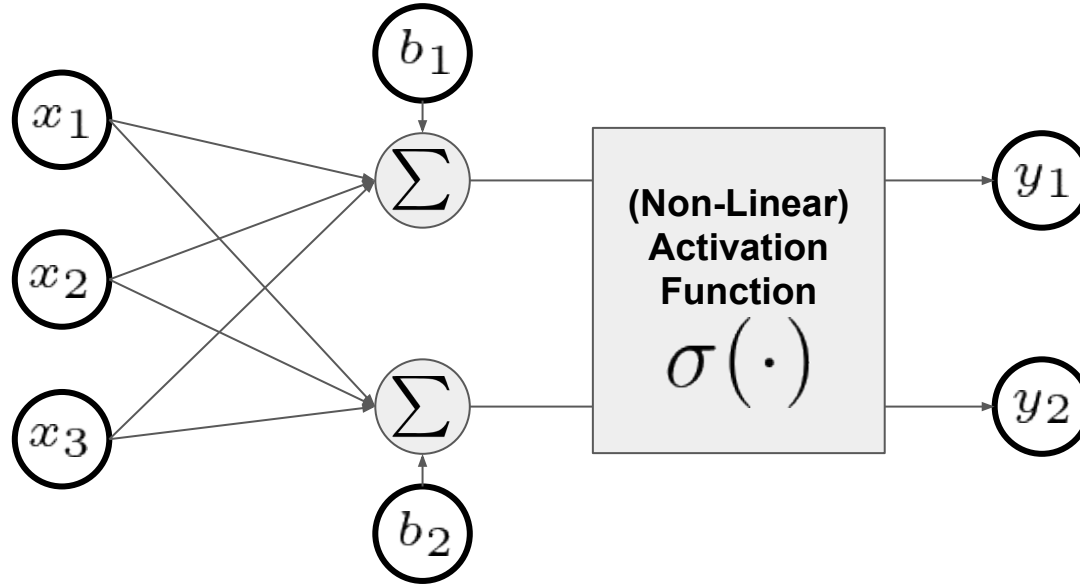


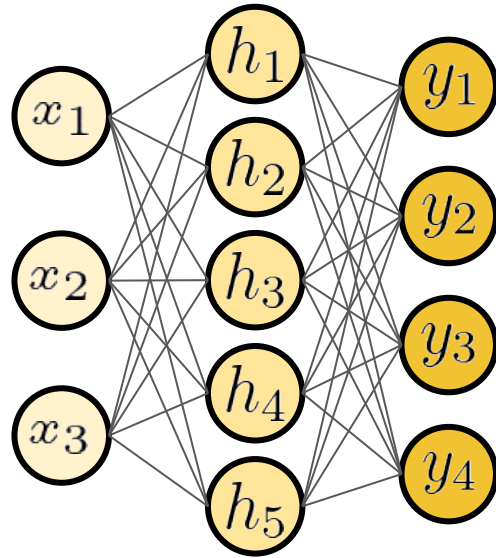
Image credit : <https://mc.ai/complete-guide-of-activation-functions/>

Artificial Neural Networks - Perceptrons

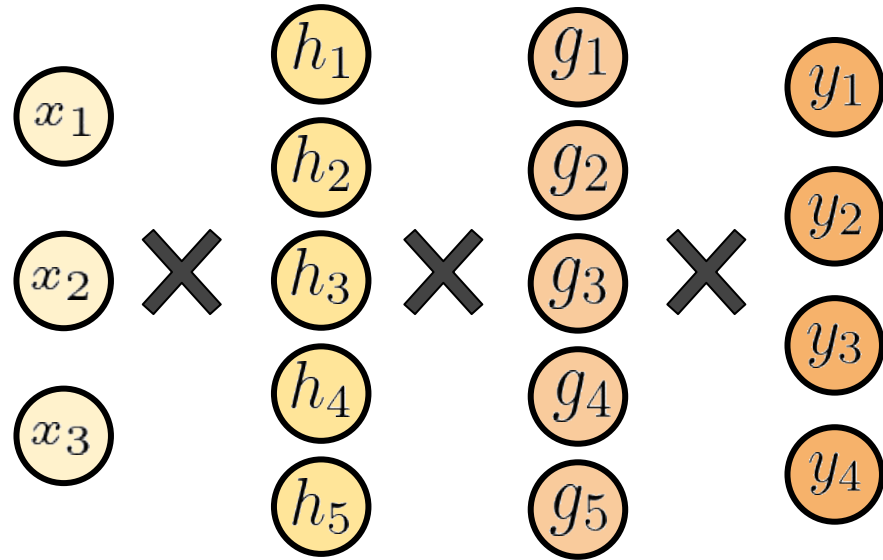
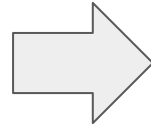


Artificial Neural Networks - Perceptrons

And stack more layers....!



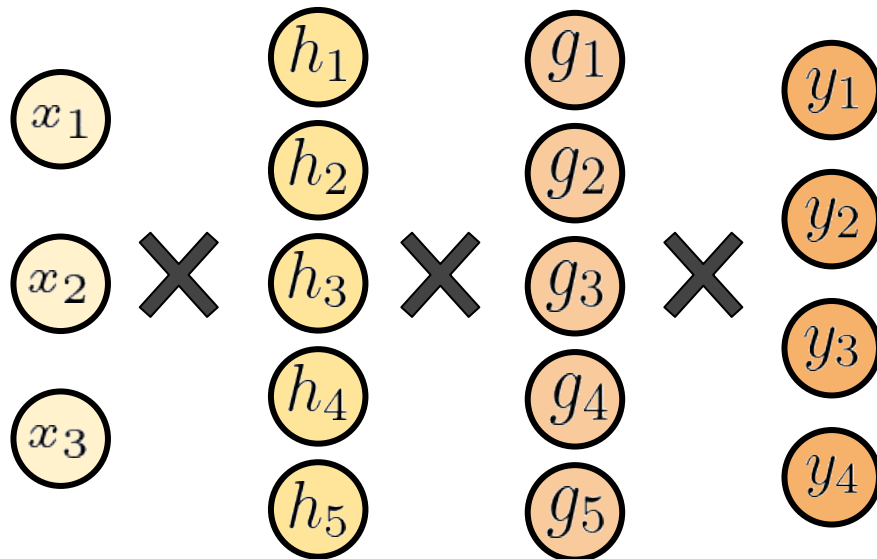
“Hidden Layer”



“Two Hidden Layers”

Multi Layered Perceptrons (MLPs)

Stack layers to construct more complex function!

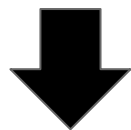


$$y = \mathbf{W}^{(3)} \sigma(\mathbf{W}^{(2)} \sigma(\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$

Multi Layered Perceptrons (MLPs)

What would happen if we do not have non-linear activation function?

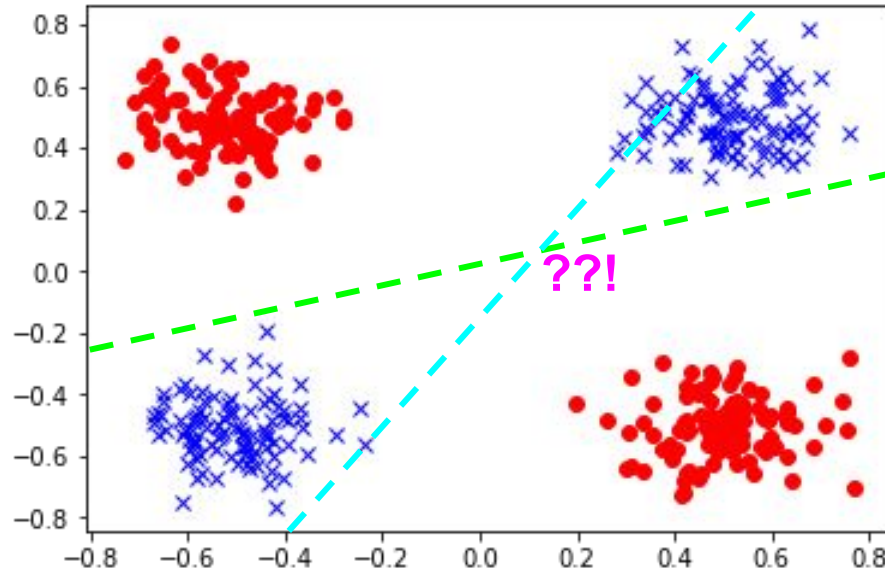
$$y = \mathbf{W}^{(3)} \sigma(\mathbf{W}^{(2)} \sigma(\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$



$$\begin{aligned} y &= \mathbf{W}^{(3)} (\mathbf{W}^{(2)} (\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)} \\ &= \mathbf{W}^{(3)} \mathbf{W}^{(2)} \mathbf{W}^{(1)} \mathbf{x} + (\mathbf{W}^{(3)} \mathbf{W}^{(2)} \mathbf{b}^{(1)} + \mathbf{W}^{(3)} \mathbf{b}^{(2)} + \mathbf{b}^{(3)}) \\ &= \mathbf{W}^{(new)} \mathbf{x} + \mathbf{b}^{(new)} \end{aligned}$$

Deep Feed Forward Network

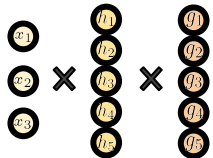
What would happen if we do not have non-linear activation function?



<https://towardsdatascience.com/visualizing-the-non-linearity-of-neural-networks-c55b2a14ad7a>

Deep Feed Forward Network

The point is to learn the (non-linear) representation !

$$y = f^*(\mathbf{x}) \xrightarrow{\text{approximate}} y = f(\mathbf{x}; \theta, \mathbf{w}) = \underbrace{\phi(\mathbf{x}; \theta)}_{\text{Hidden layer!}}^\top \mathbf{w}$$


The diagram illustrates the internal structure of the hidden layer. It shows three vertical columns of nodes. The first column contains three input nodes labeled x_1, x_2, x_3 . The second column contains five hidden nodes labeled h_1, h_2, h_3, h_4, h_5 . The third column contains five output nodes labeled q_1, q_2, q_3, q_4, q_5 . Multiplication symbols (\times) are placed between the first and second columns, and between the second and third columns, indicating the dot product operation between the input vector and the hidden layer weights, and between the hidden layer output and the output weights.

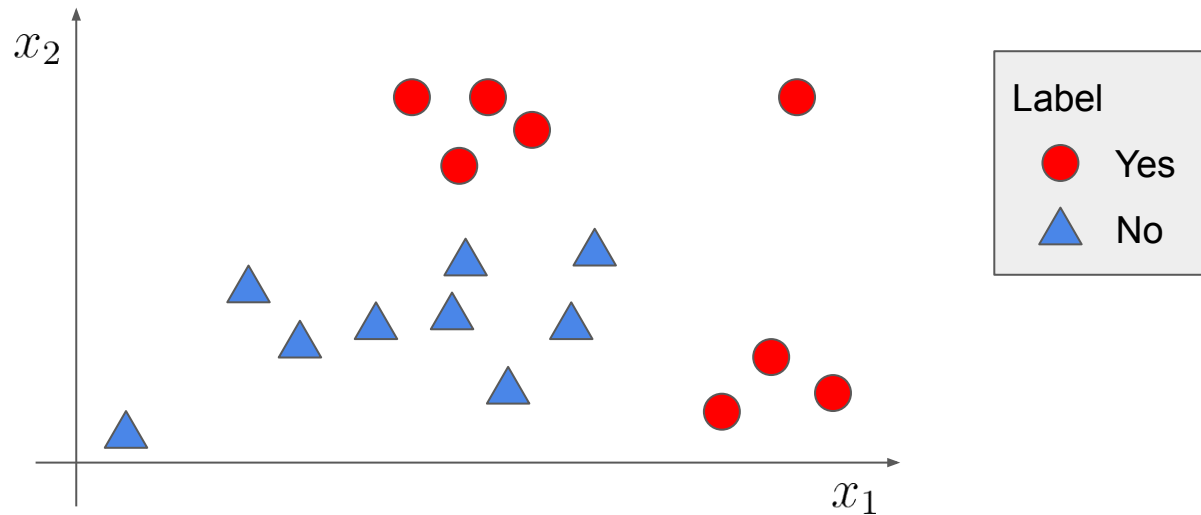
What engineers do:

- Find the right general function family $\phi(\cdot)$, that can parameterize the representation of input \mathbf{x} with θ .
- Use the optimization algorithm to find θ that can build a good representation.

Example Problem

Q. Am I addicted to Instagram? (classification)

- Inputs: x_1 : Spent times in Instagram, x_2 : Number of uploaded Stories per day

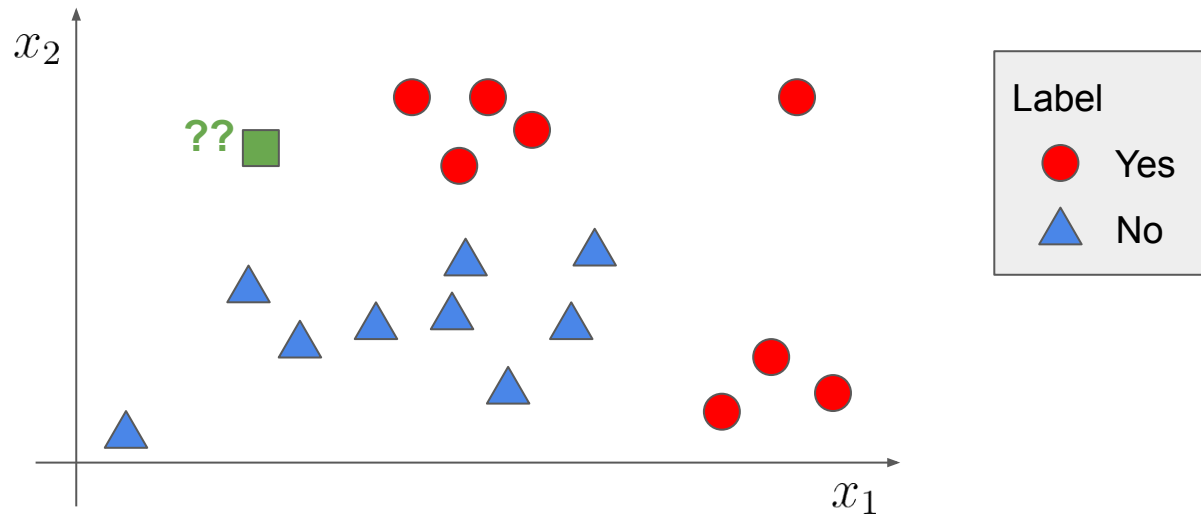


(Inspired by slides from http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf)

Example Problem

Q. Am I addicted to Instagram? (classification)

- Inputs: x_1 : Spent times in Instagram, x_2 : Number of uploaded Stories per day

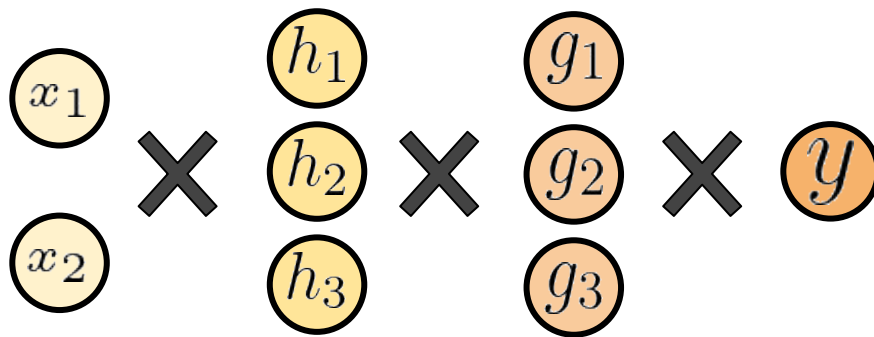


(Inspired by slides from http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf)

Example Problem

Q. Am I addicted to Instagram? (classification)

- Inputs: x_1 : Spent times in Instagram, x_2 : Number of uploaded Stories per day



Outputs: y : A probability of being addicted.

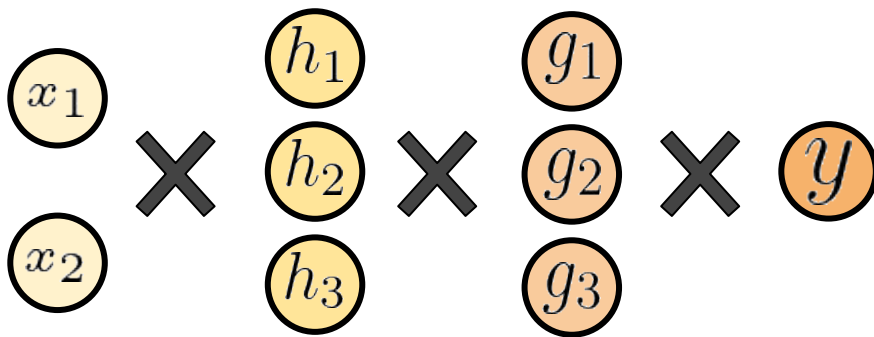
- If $y \geq 0.5$: consider as addiction
- Otherwise, consider as non-addiction.

(Inspired by slides from http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf)

Example Problem

Q. Am I addicted to Instagram? (classification)

- Inputs: x_1 : Spent times in Instagram, x_2 : Number of uploaded Stories per day



Outputs: y : A probability of being addicted.

- If $y \geq 0.5$: consider as addiction
- Otherwise, consider as non-addiction.

** If predicted and actual outputs are too different from each other in general, what can we say about it?

: the parameter needs to be trained !

:or it is overfitted :(

: How can we measure this 'cost' of

'being incorrect'?

(Inspired by slides from http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf)

Loss : the cost from incorrect predictions

Training data : $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1\dots N}$,

Neural network model : $\hat{\mathbf{y}}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$

Loss for i-th data : $\mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$  **How do we usually define this?**

Empirical loss : $\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

Loss : the cost from incorrect predictions

In Classification Problem, how outputs are defined?

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_{N_c} \end{bmatrix}, \text{ where } N_c \text{ is the number of classes, } \hat{y}_i \text{ is a predicted probability of } \mathbf{x} \text{ being } i\text{-th class.}$$

$$\text{Loss for } i\text{-th data : } \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}) = \mathcal{L}(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) = - \sum_{c=1}^{N_c} y_c \log \hat{y}_c \implies \text{Cross-Entropy Loss}$$

Loss : the cost from incorrect predictions

In Classification Problem, what if there are only two labels (True / False)?

$$\mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}) = \mathcal{L}(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) = -[y_0^{(i)} \log \hat{y}_0^{(i)} + (1 - y_0^{(i)}) \log(1 - \hat{y}_0^{(i)})]$$

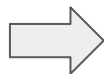
→ Binary Cross-Entropy Loss

Loss : the cost from incorrect predictions

How about regression problem?

\hat{y} = Estimated continuous real value

Loss for i-th data : $\mathcal{L}(f(\mathbf{x}^{(i)}; \theta), y^{(i)}) = \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$



Mean squared error loss

Now what should we do? → Find a set of parameters that can minimize the loss!

...or L1-norm loss or L2-norm loss for vector regression.

Optimization: Find parameters minimizing the Loss

$$\theta^* = \arg \min_{\theta} \left(\frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}) \right)$$

- A neural network based model is not linear.
- The loss function would not be a convex.
- We cannot get the analytic solution for θ^* .

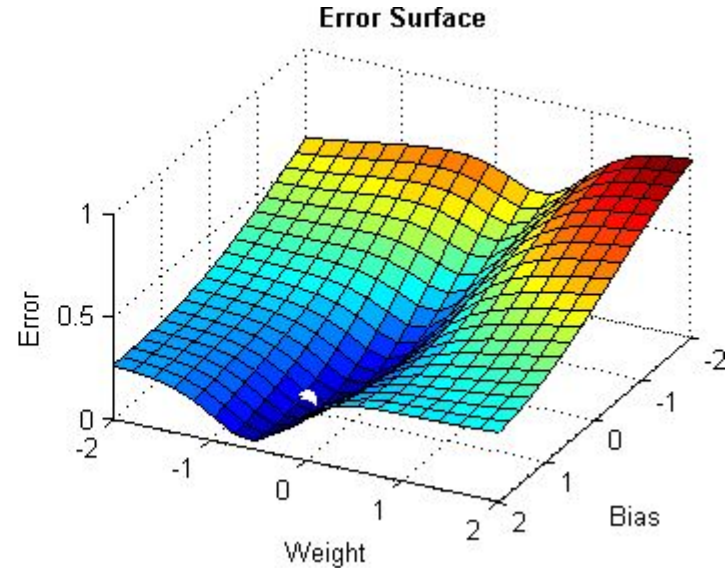
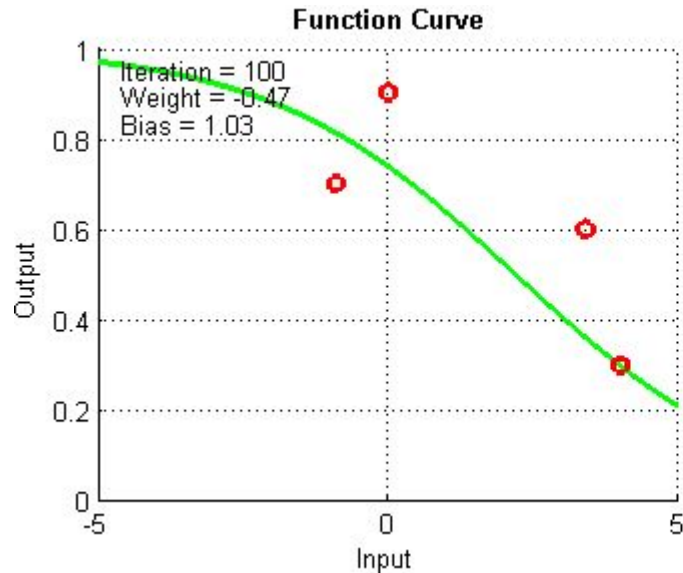
Consider a set of parameters as a n-dimensional vector s.t. $\vec{\theta} \in \mathbb{R}^n$

Can we find $\Delta \vec{\theta}$, a direction to change $\vec{\theta} \rightarrow \vec{\theta} + \Delta \vec{\theta}$ to decrease the loss?

\Rightarrow Opposite direction of gradient is the solution!

Optimization: Gradient Descent

Iteratively optimize parameters based on the training dataset



<https://towardsdatascience.com/improving-vanilla-gradient-descent-f9d91031ab1d>


Optimization: Gradient Descent

Recall : Empirical loss :
$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$$


1. Initialize a set of parameters θ in a random way
2. Loop below until $\mathcal{J}(\theta)$ converges :
 - a) Compute $\nabla_{\theta} \mathcal{J}(\theta)$
 - b) Update $\theta \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{J}(\theta)$
3. Return θ

Optimization: Gradient Descent

Recall : Empirical loss :
$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$$

1. Initialize a set of parameters θ in a random way
2. Loop below until $\mathcal{J}(\theta)$ converges :
 - a) Compute $\nabla_{\theta} \mathcal{J}(\theta)$  **This would be heavy-and-expensive computation!**
 - b) Update $\theta \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{J}(\theta)$
3. Return θ

Optimization: Stochastic Gradient Descent

1. Initialize a set of parameters θ in a random way
1. Loop below until the convergence:
 - a) Pick one data point $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$
 - b) Compute $\nabla_{\theta} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$
 - c) Update $\theta \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$  This can be very noisy :(
2. Return θ

Optimization: Mini-Batch Gradient Descent

1. Initialize a set of parameters θ in a random way
1. Loop below until the convergence:
 - a) Pick a batch $B = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1 \dots N_B}$ of N_B data points.
 - b) Compute $\frac{1}{N_B} \sum_{i=1}^{N_B} \nabla_{\theta} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$
 - c) Update $\theta \leftarrow \theta - \gamma \frac{1}{N_B} \sum_{i=1}^{N_B} \nabla_{\theta} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$
2. Return θ

Optimization: Mini-Batch Gradient Descent

1. Initialize a set of parameters θ in a random way
1. Loop below until the convergence:
 - a) Pick a batch $B = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1 \dots N_B}$ of N_B data points.

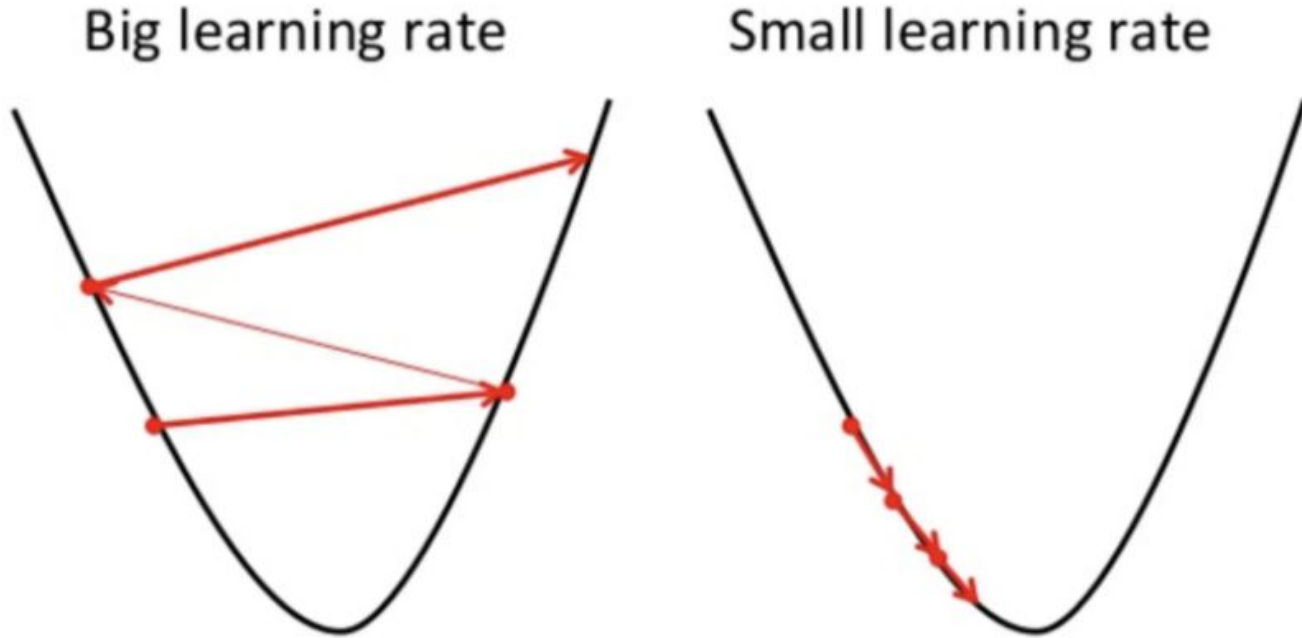
- b) Compute $\frac{1}{N_B} \sum_{i=1}^{N_B} \nabla_{\theta} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

- c) Update $\theta \leftarrow \theta - \gamma \frac{1}{N_B} \sum_{i=1}^{N_B} \nabla_{\theta} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

2. Return θ

How do we choose this value (learning rate)?

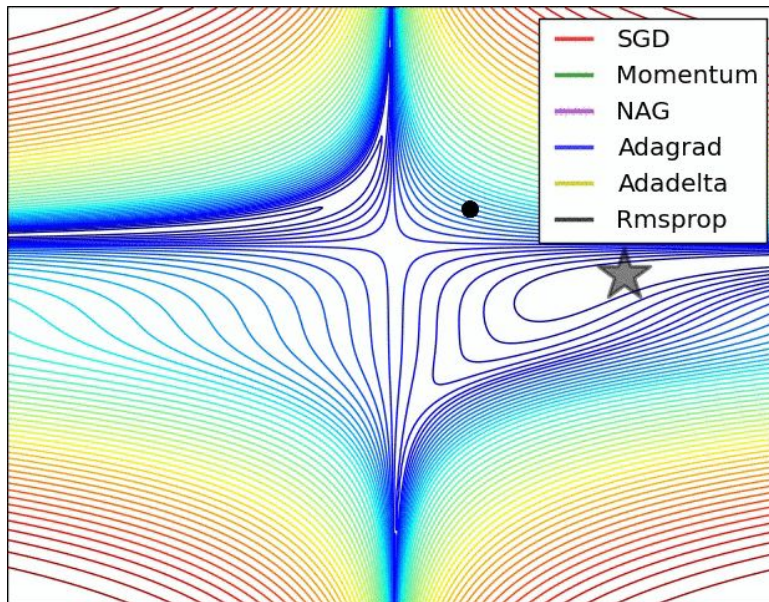
Optimization: minimizing the loss



[https://www.khanacademy.org/a/optimization-101](#)

Optimization: minimizing the loss

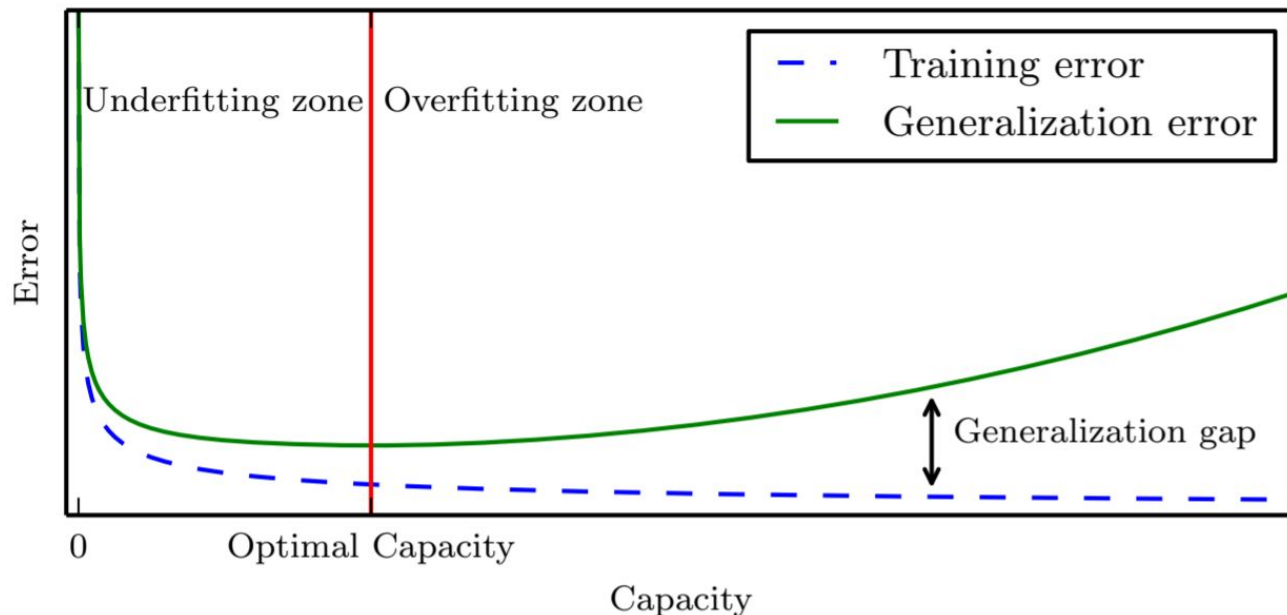
Nowadays we use adaptive learning rate, based on methods like...



<https://cs231n.github.io/neural-networks-3/>

Optimization: minimizing the loss

Can convergence of loss in training data guarantee the generalization performance?



<https://www.deeplearningbook.org/>

Possible to-do list for engineers?

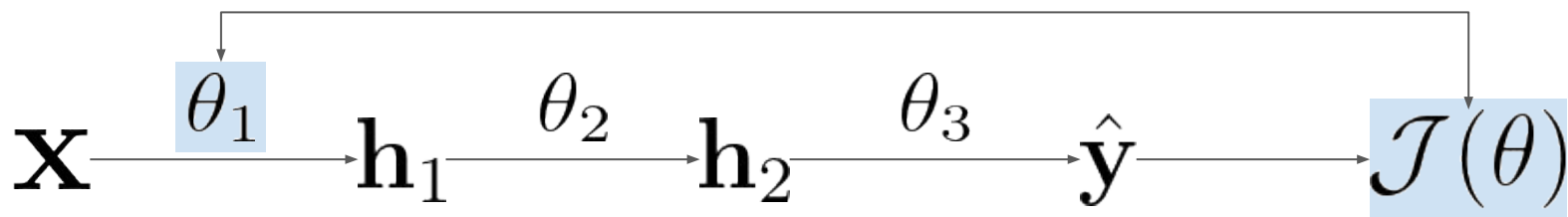
- Decide what would be Input and Output
- Design the neural network architecture, a structure of model.
- Choose which loss function to use
- Choose which optimizer to use
- Choose (initial) learning rate, batch size, model size
- ...and regularly check the validation error during the training!
 - This would be dealt in next lecture in detail.

Optimization - Gradient Descent

Recall : Empirical loss : $\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

1. Initialize a set of parameters θ in a random way
2. Loop below until $\mathcal{J}(\theta)$ converges :
 - a) Compute $\nabla_{\theta} \mathcal{J}(\theta)$ \leftarrow How do we get this?
 - b) Update $\theta \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{J}(\theta)$
3. Return θ

Back-Propagation



$$\nabla_{\theta} \mathcal{J}(\theta) = \begin{bmatrix} \frac{\partial \mathcal{J}}{\partial \theta_1} \\ \frac{\partial \mathcal{J}}{\partial \theta_2} \\ \frac{\partial \mathcal{J}}{\partial \theta_3} \end{bmatrix} \quad ??$$

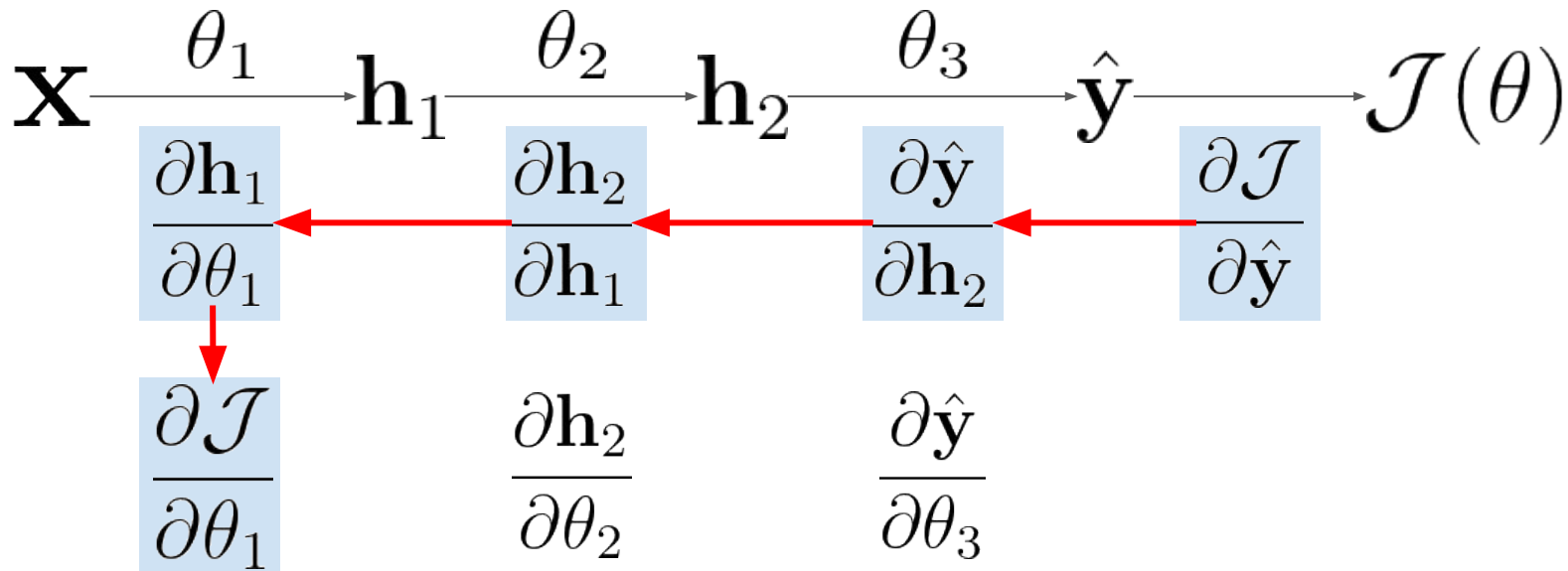
Back-Propagation

Things that are easy to get :

$$\begin{array}{ccccccc} \mathbf{X} & \xrightarrow{\theta_1} & \mathbf{h}_1 & \xrightarrow{\theta_2} & \mathbf{h}_2 & \xrightarrow{\theta_3} & \hat{\mathbf{y}} \xrightarrow{\quad} \mathcal{J}(\theta) \\ & \frac{\partial \mathbf{h}_1}{\partial \theta_1} & & \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} & & \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{h}_2} & \frac{\partial \mathcal{J}}{\partial \hat{\mathbf{y}}} \\ & & & & & & \\ & & & \frac{\partial \mathbf{h}_2}{\partial \theta_2} & & \frac{\partial \hat{\mathbf{y}}}{\partial \theta_3} & \end{array}$$

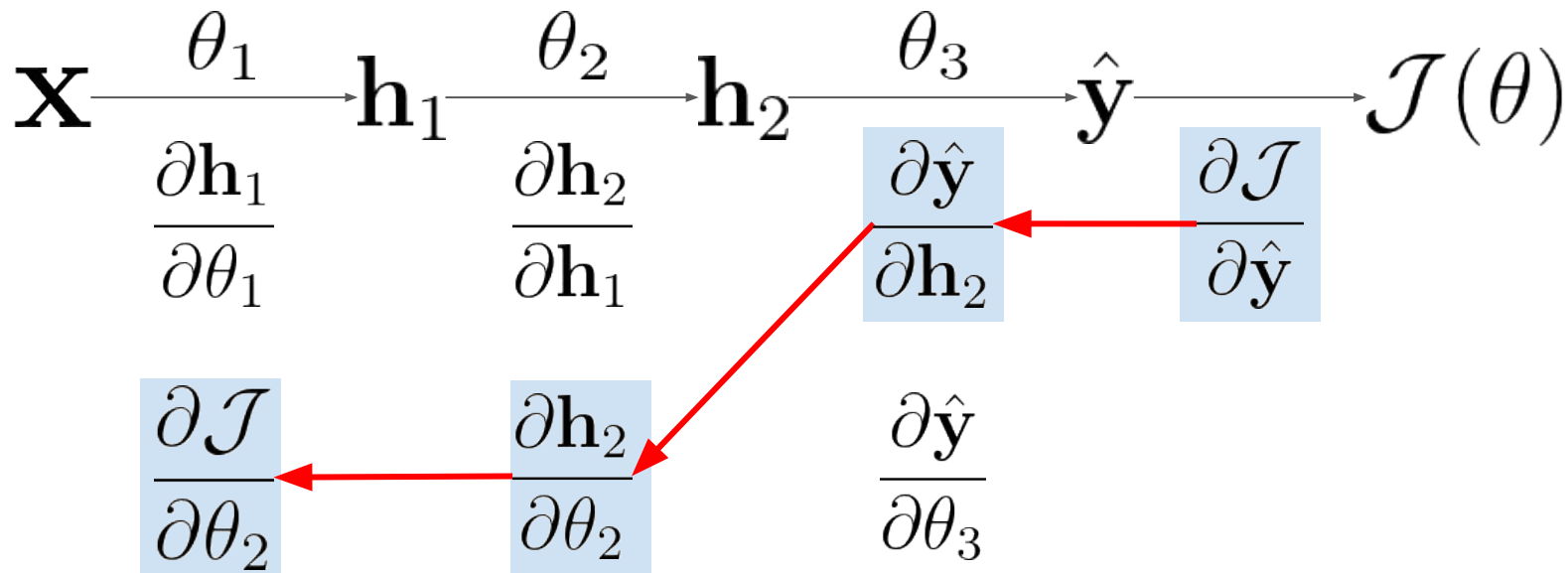
“Back”-Propagation

Use the Chain Rule!



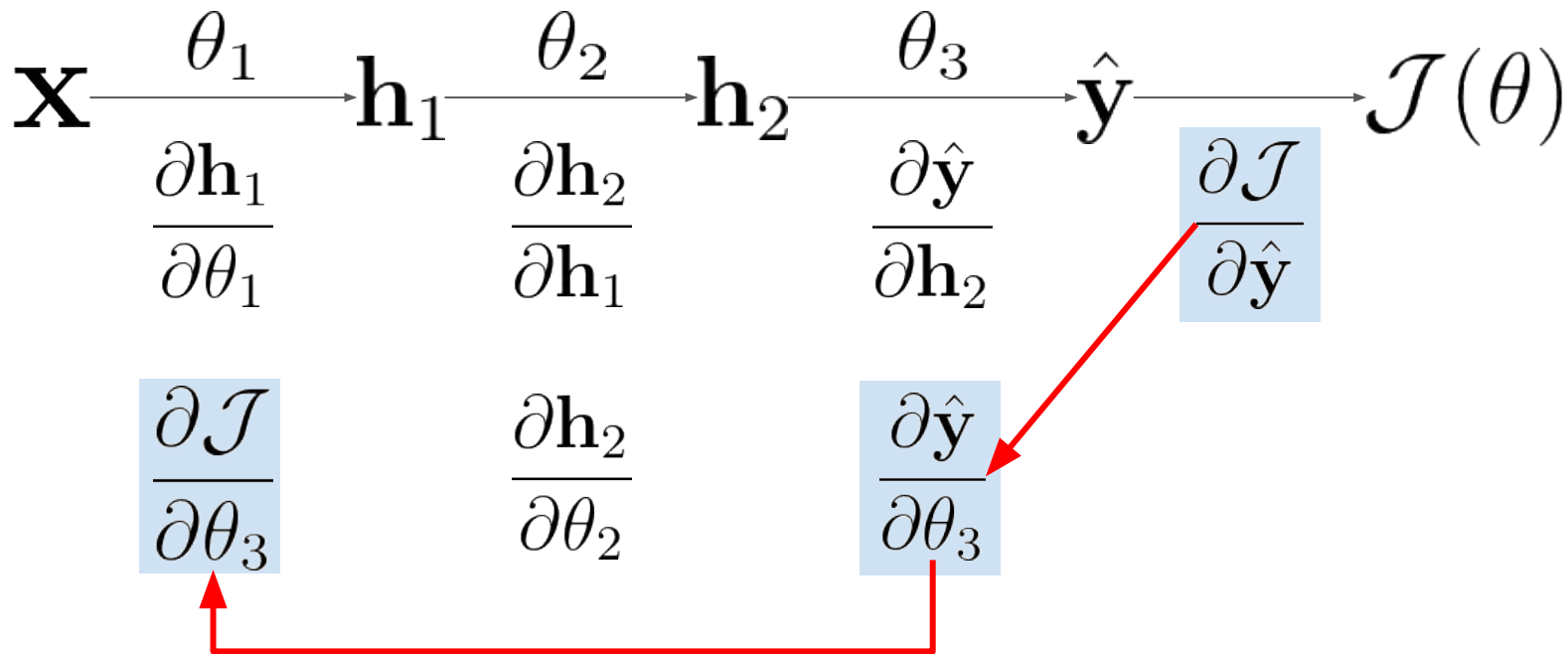
“Back”-Propagation

Use the Chain Rule!



“Back”-Propagation

Use the Chain Rule!

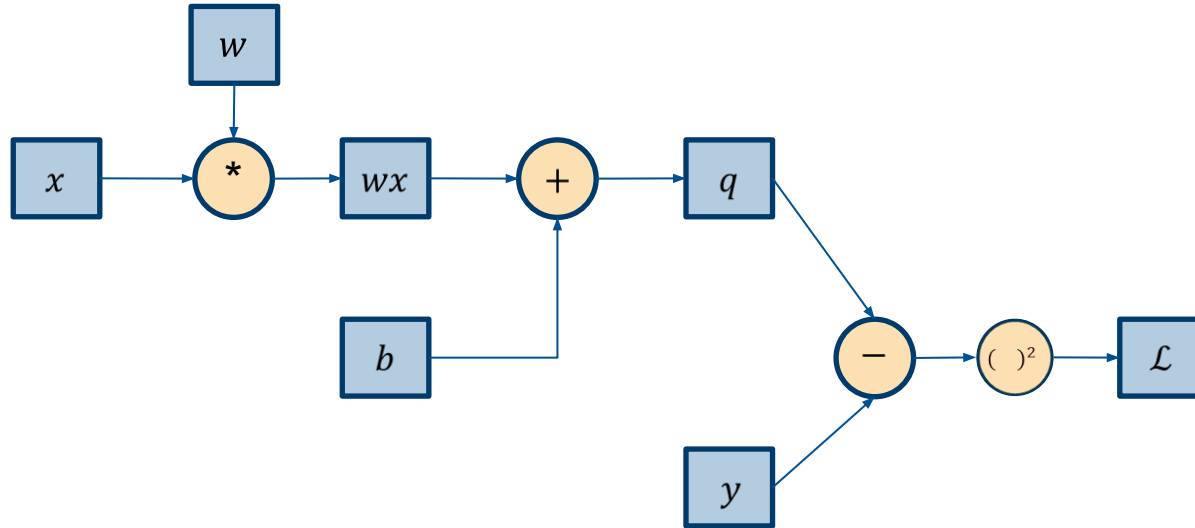


“Back”-Propagation - more examples

$$\mathcal{L} = (y - (wx + b))^2, \text{ and let } q = wx + b$$
$$q = wx + b \quad \frac{\partial q}{\partial w} = x, \quad \frac{\partial q}{\partial b} = 1$$
$$\mathcal{L} = (y - q)^2 \quad \frac{\partial \mathcal{L}}{\partial q} = -2(y - q)$$

For gradient descent,
What we want is...

$$\frac{\partial \mathcal{L}}{\partial w} \text{ and } \frac{\partial \mathcal{L}}{\partial b}$$



“Back”-Propagation - more examples

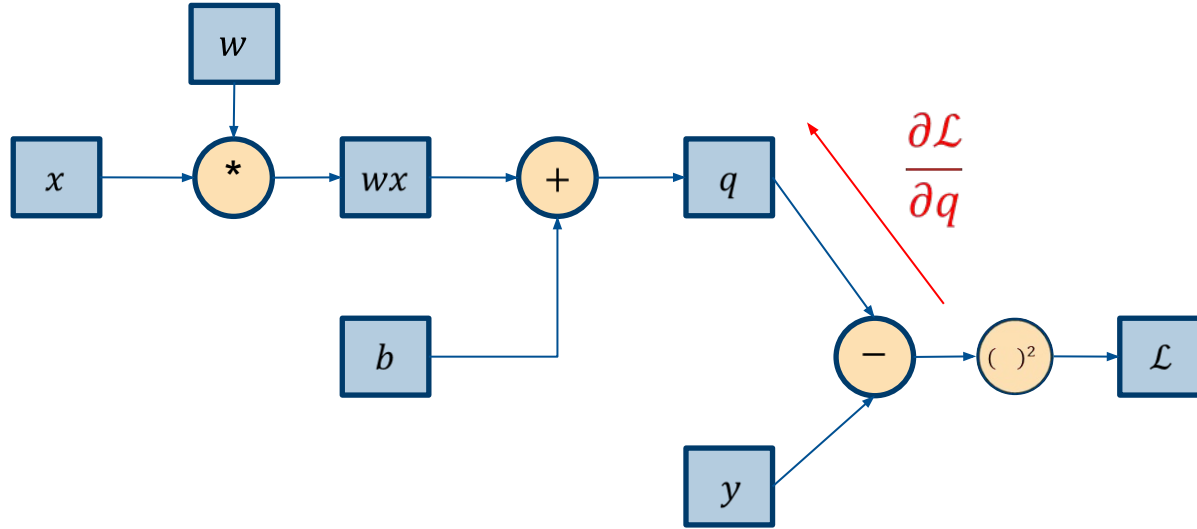
$$\mathcal{L} = (y - (wx + b))^2, \text{ and let } q = wx + b$$

$$q = wx + b \quad \frac{\partial q}{\partial w} = x, \quad \frac{\partial q}{\partial b} = 1$$

$$\mathcal{L} = (y - q)^2 \quad \frac{\partial \mathcal{L}}{\partial q} = -2(y - q)$$

For gradient descent,
What we want is...

$$\frac{\partial \mathcal{L}}{\partial w} \text{ and } \frac{\partial \mathcal{L}}{\partial b}$$



“Back”-Propagation - more examples

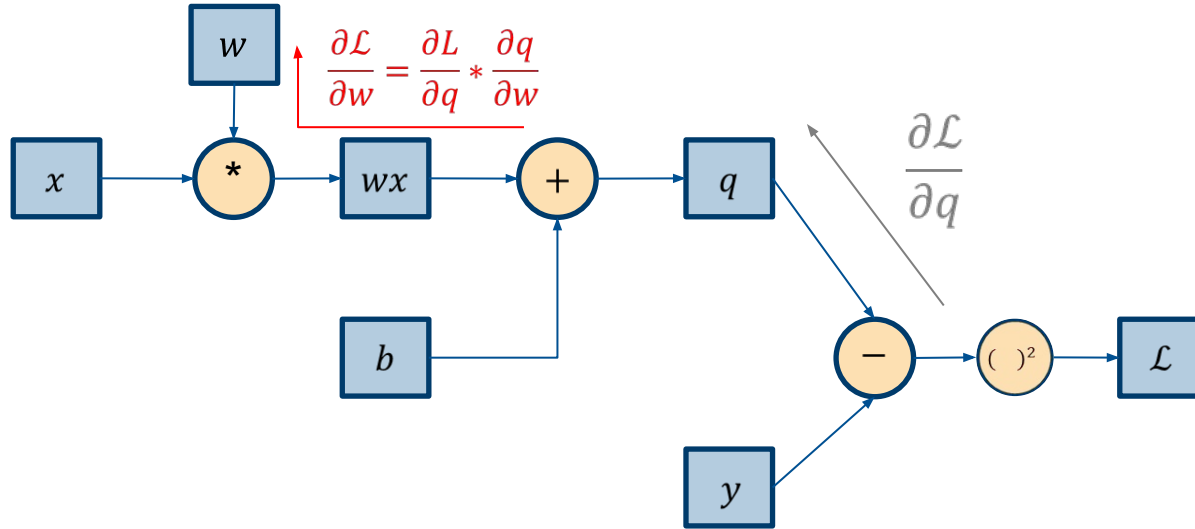
$$\mathcal{L} = (y - (wx + b))^2, \text{ and let } q = wx + b$$

$$q = wx + b \quad \frac{\partial q}{\partial w} = x, \quad \frac{\partial q}{\partial b} = 1$$

$$\mathcal{L} = (y - q)^2 \quad \frac{\partial \mathcal{L}}{\partial q} = -2(y - q)$$

For gradient descent,
What we want is...

$$\frac{\partial \mathcal{L}}{\partial w} \text{ and } \frac{\partial \mathcal{L}}{\partial b}$$

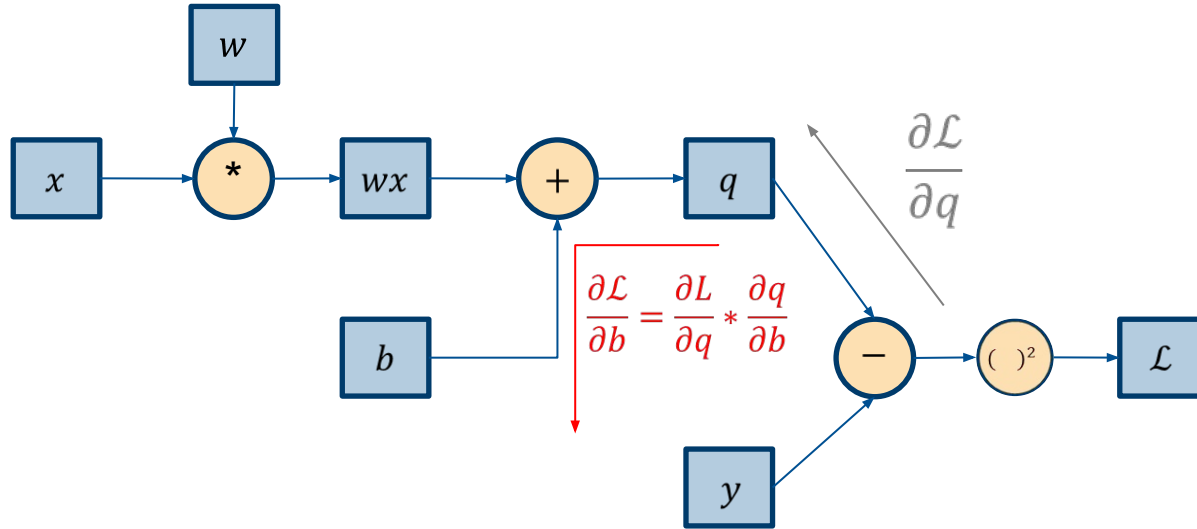


“Back”-Propagation - more examples

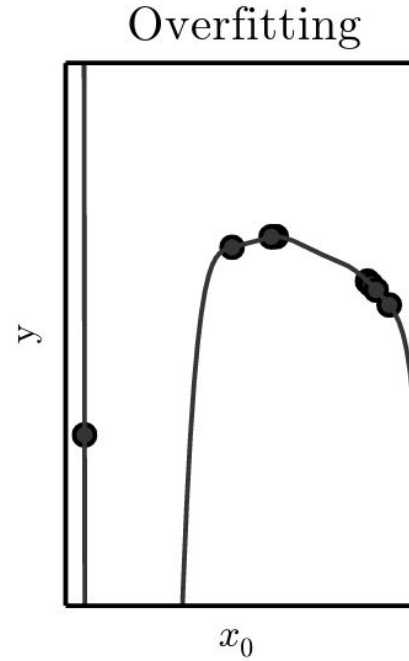
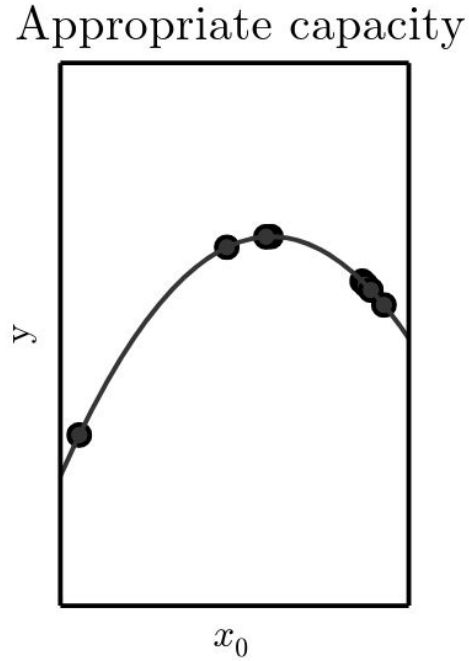
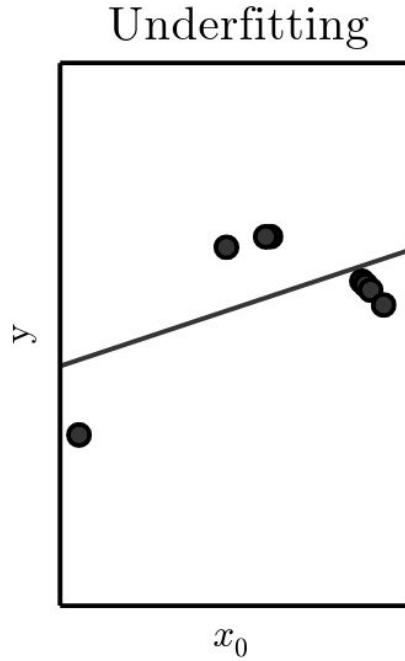
$$\mathcal{L} = (y - (wx + b))^2, \text{ and let } q = wx + b$$
$$q = wx + b \quad \frac{\partial q}{\partial w} = x, \quad \frac{\partial q}{\partial b} = 1$$
$$\mathcal{L} = (y - q)^2 \quad \frac{\partial \mathcal{L}}{\partial q} = -2(y - q)$$

For gradient descent,
What we want is...

$$\frac{\partial \mathcal{L}}{\partial w} \text{ and } \frac{\partial \mathcal{L}}{\partial b}$$



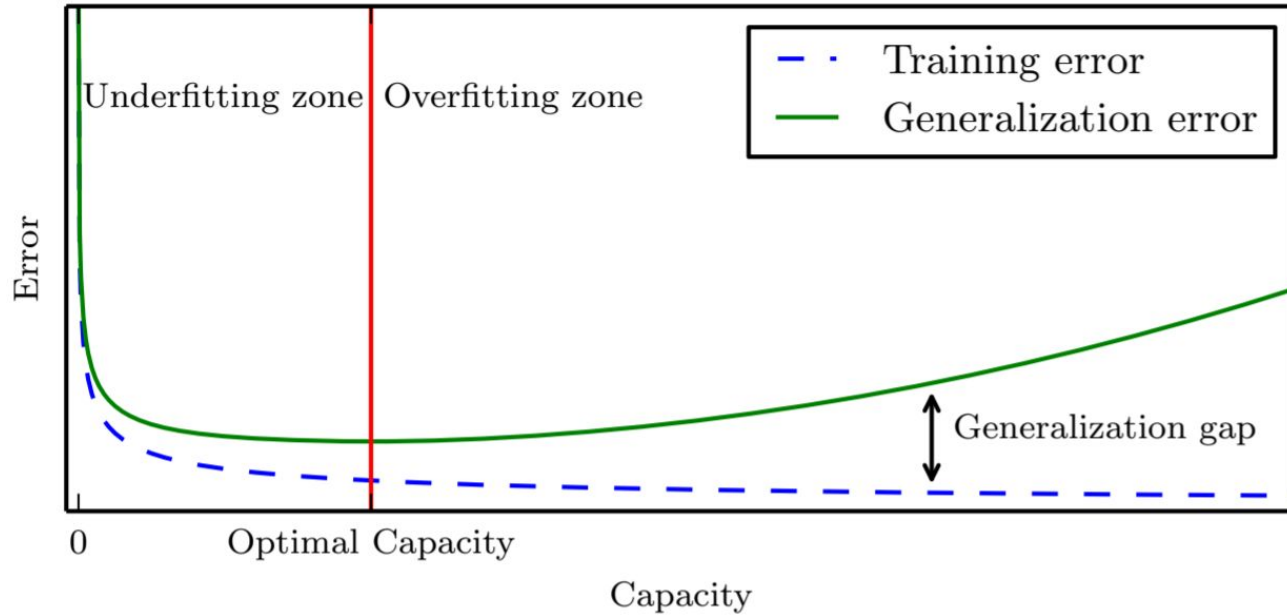
Overfitting



<https://www.deeplearningbook.org/>

Overfitting

Can convergence of loss in training data guarantee the generalization performance?



<https://www.deeplearningbook.org/>

Regularization

“Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”

(GoodFellow 2016)

- L1/L2 regularization
- Dropout
- Data Augmentation
- ... and more things that you might face while reading papers

L1/L2 Regularization

$$\mathcal{L}(\mathbf{W}) + \lambda \sum_{w \in \mathbf{W}} \|w\|_1$$

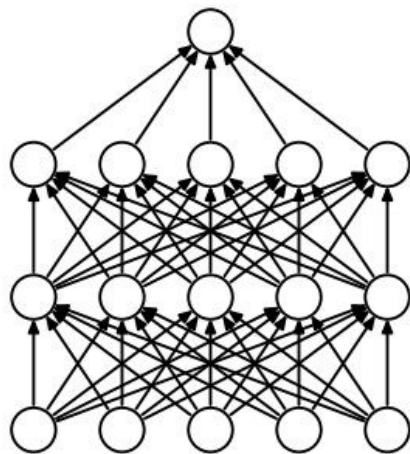
$$\mathcal{L}(\mathbf{W}) + \lambda \sum_{w \in \mathbf{W}} \|w\|_2$$



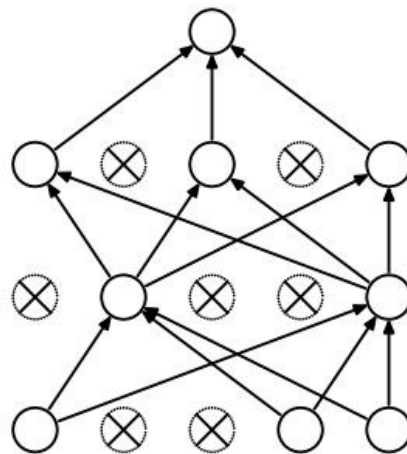
Penalize to the complexity! λ : hyperparameter

Dropout

- Activate the neuron with probability of p : hyperparameter
- “Sampling the Neural Network”



(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.
<https://cs231n.github.io/neural-networks-2>

Data Augmentation



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

Chen, Ting et al. "A simple framework for contrastive learning of visual representations" arXiv preprint arXiv:2002.05719(2020).

Any Questions?