Artificial Neural Networks or Deep Learning

Instructor: Ruixuan Wang
wangruix5@mail.sysu.edu.cn

School of Data and Computer Science
Sun Yat-Sen University

28 February, 2019

About this course

Objectives

- basic concepts and model structures
- various advanced models and their applications
- ability to read and summarize research papers
- skills and tricks to design and train models
- ability to analyze, implement, and solve real applications

Syllabus

Weeks 1-3	Basics of deep learning
Weeks 4-7	Convolutional neural networks (CNN)
Weeks 8-10	Generative neural networks (e.g., GAN)
Weeks 11-13	Recurrent neural networks (RNN)
Weeks 14-18	Research trends: advanced topics

About this course

Objectives

- basic concepts and model structures
- various advanced models and their applications
- ability to read and summarize research papers
- skills and tricks to design and train models
- ability to analyze, implement, and solve real applications

Syllabus

Weeks 1-3	Basics of deep learning
Weeks 4-7	Convolutional neural networks (CNN)
Weeks 8-10	Generative neural networks (e.g., GAN)
Weeks 11-13	Recurrent neural networks (RNN)
Weeks 14-18	Research trends: advanced topics

About this course

Prerequisite

- Math (linear algebra, calculus), English (reading, writing)
- Interested in AI: ready to spend more time than expected!

Grading

Homework (40%)	• 2-3 members per team • paper summary: week 4, 6, 8, 10, 12 • highlight (100 words) + summary (1000)
Project (50%)	• participate real challenges/contests • report: ≤ 8 pages, ≥ 5 models/methods
Others (10+5%)	• attendance; questions, best summaries, etc.

About this course

Prerequisite

- Math (linear algebra, calculus), English (reading, writing)
- Interested in AI: ready to spend more time than expected!

Grading

Homework (40%)	• 2-3 members per team • paper summary: week 4, 6, 8, 10, 12 • highlight (100 words) + summary (1000)
Project (50%)	• participate real challenges/contests • report: ≤ 8 pages, ≥ 5 models/methods
Others (10+5%)	• attendance; questions, best summaries, etc.

About this course (cont')

Grading (Lab course)

Lab (50%)	• learn and finish assignment in class • show and briefly describe results
Project (40%)	• yourself as a team • choose a 2018/2019 good paper • implement the model/method therein • evaluate the model/method as in paper • report and code • score related to difficulty level
Others (10+5%)	• attendance; questions, lab submissions, etc.

About this course

Scan only if you decided to take this course

Applications of deep learning

Deep learning became well-known only since 2012, but...

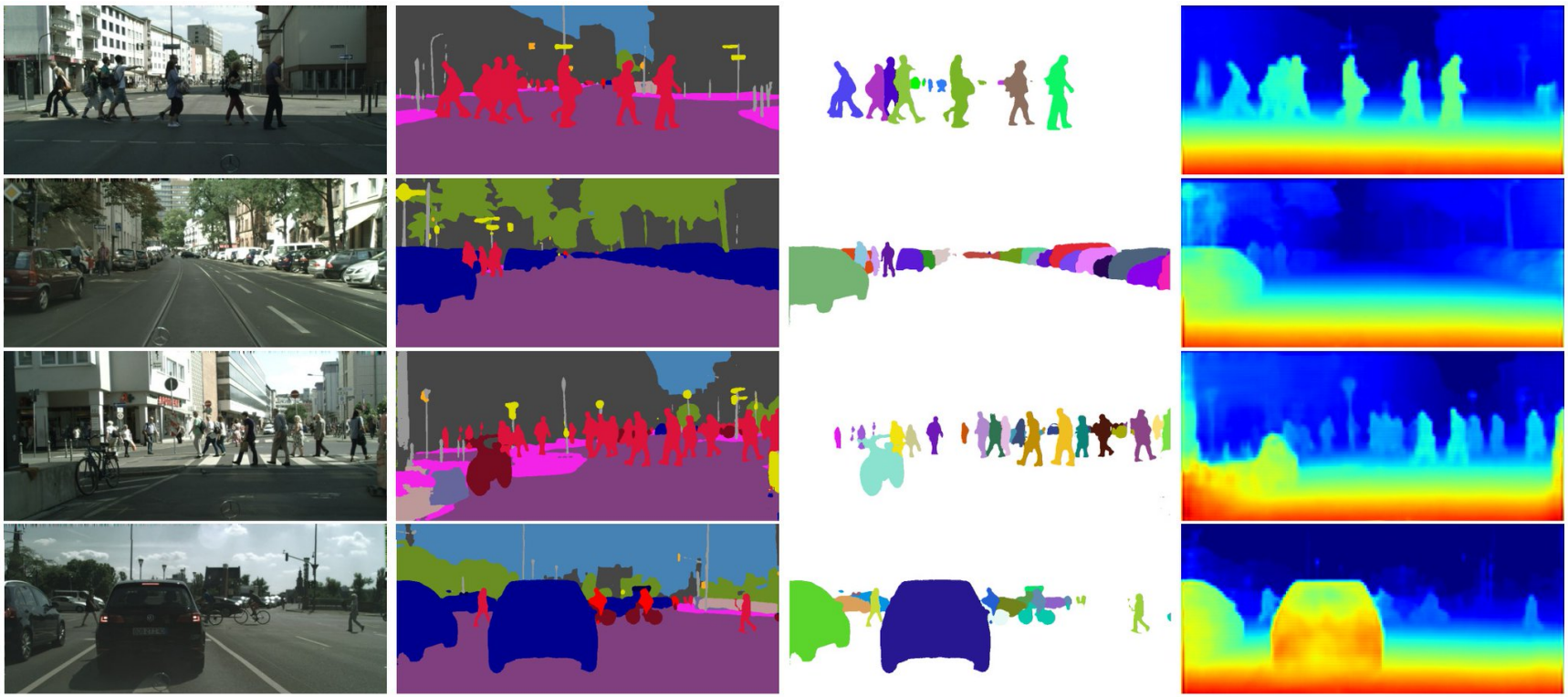
Applications of deep learning: pay-with-face

Face identification: classification, metric learning, retrieval

Applications of deep learning: image recognition

ImageNet challenge: 1000-class image classification

Applications of deep learning: self-driving

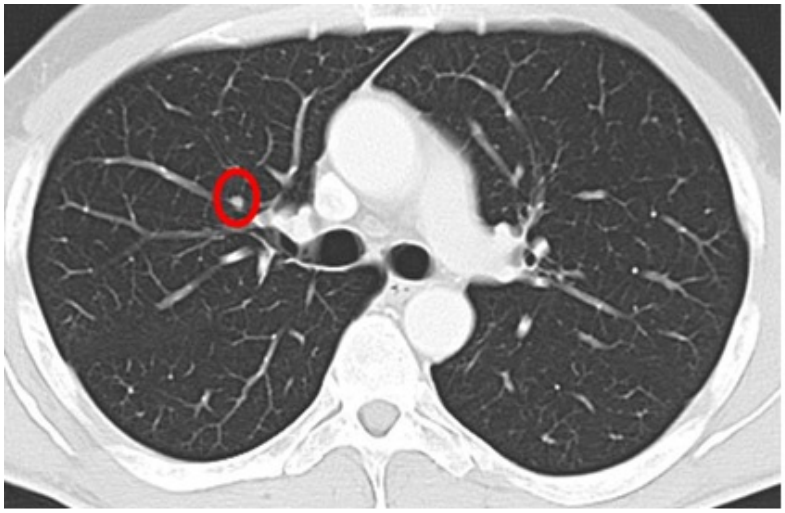


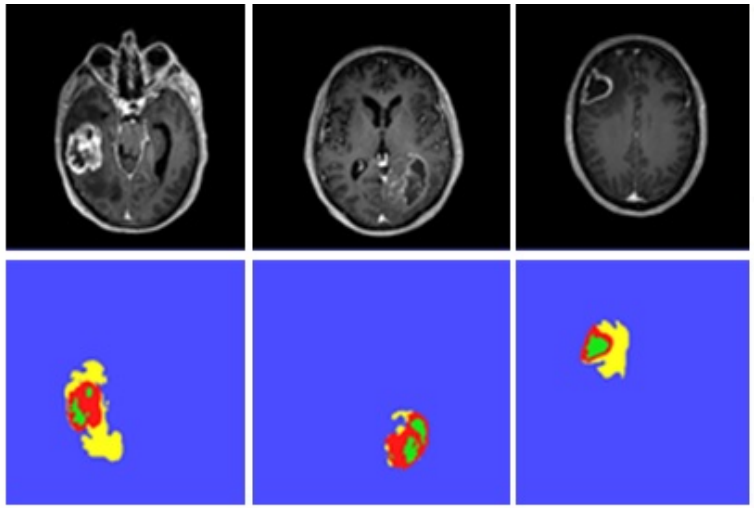
(a) Input image (b) Segmentation output (c) Instance output (d) Depth output

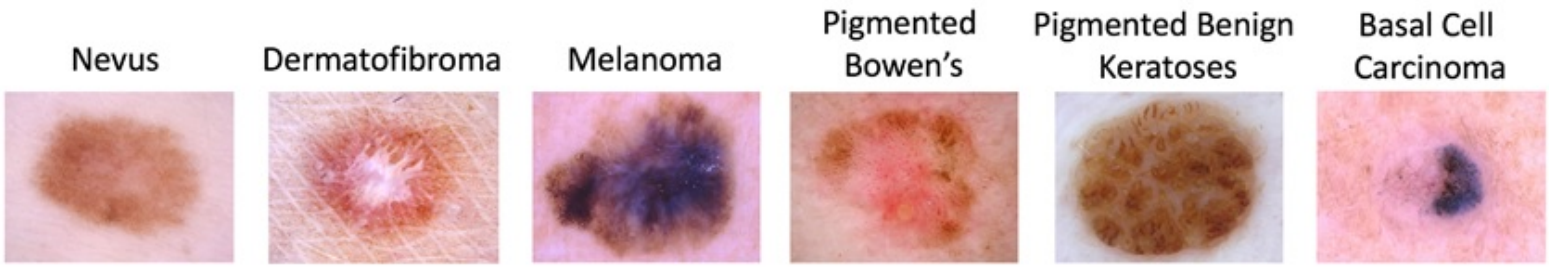
Scene understanding: semantic & instance segmentation, classification; detection of objects+humans+events, steering+acceleration

picture from https://alexgkendall.com

Applications of deep learning: healthcare

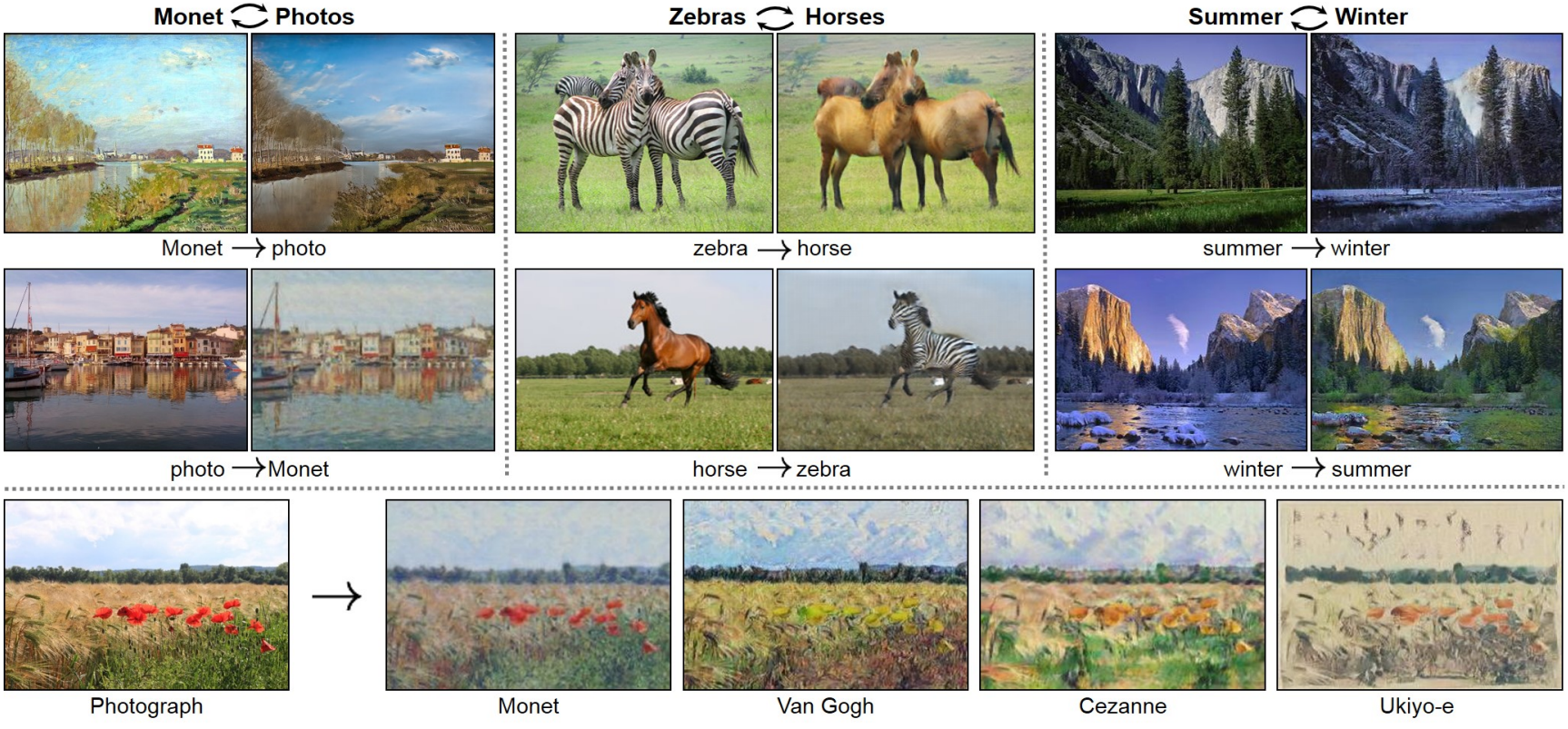






Medical image analysis: detection of lesions, segmentation of lesion regions, diagnosis

Applications of deep learning: image translation



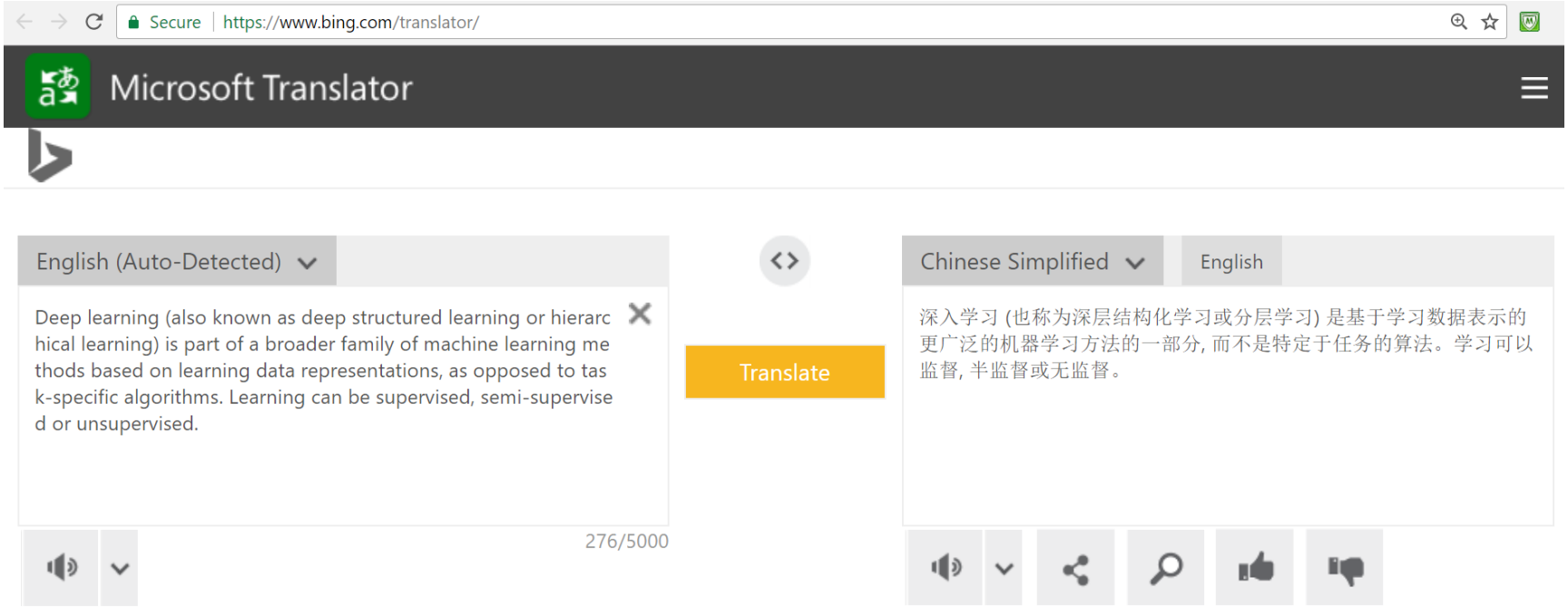
Style transfer: from one style to another

picture from https://github.com/junyanz/CycleGAN

Applications of deep learning: speech recognition

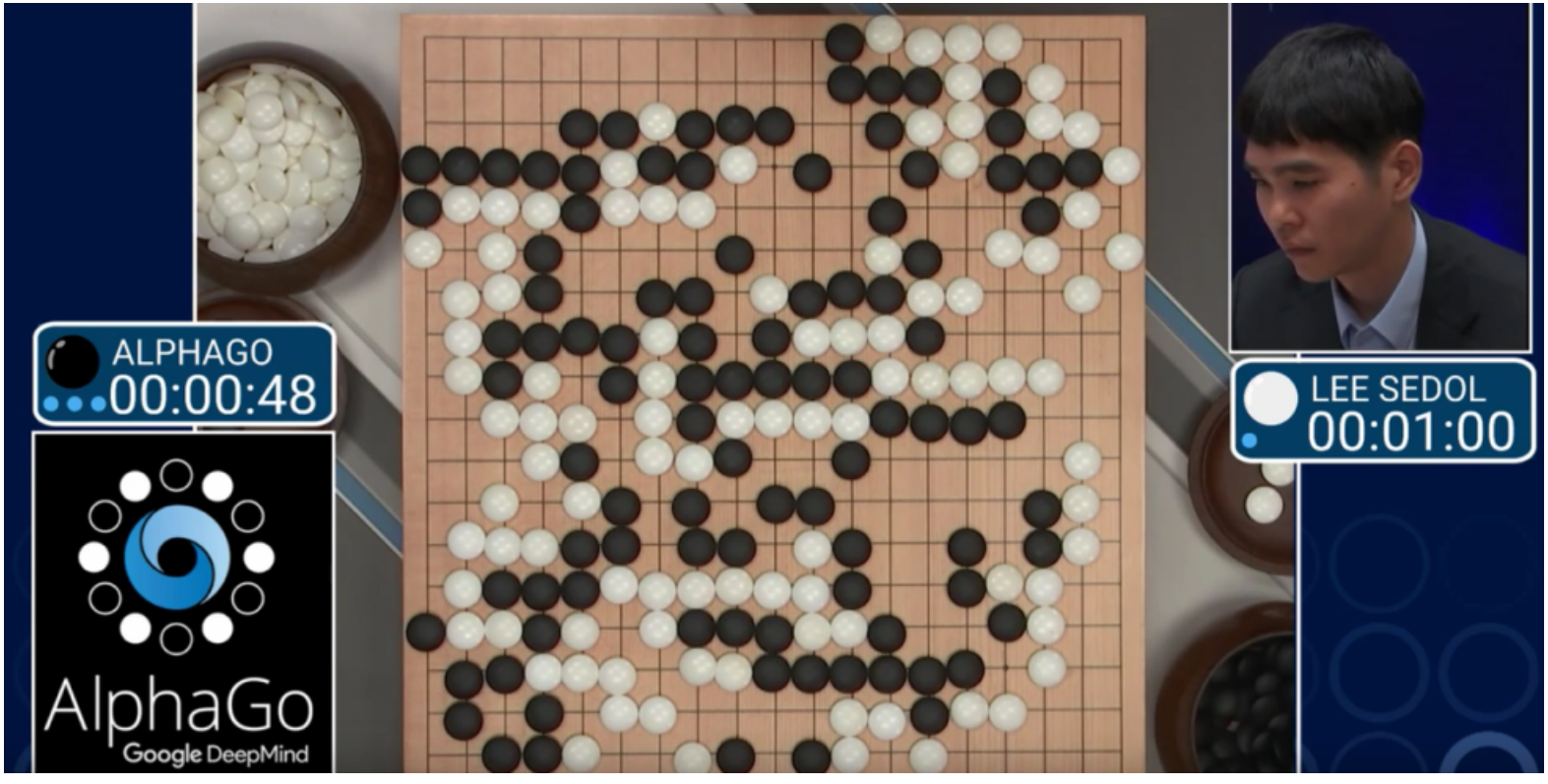
Classification: from voice to text

Applications of deep learning: language translation



Encoder-decoder: from one to another language

Applications of deep learning: AlphaGo



Reinforcement learning: learn rules/policies from experience

Applications of deep learning: a lot more

- Robot and chat-bot
- Surveillance
- Finance
- Agriculture
- Education
- ...

Problem solving

How to solve the above problems?

A linear regression problem: an example

A simple problem

Develop a system to predict a person's weight based on his or her height.

To solve this problem

1. **Formulation**: suppose a person's weight y is linearly correlated with his or her height x, i.e., $y = wx + b$
2. **Data**: collect a number of persons' weight and height: $D = \{(x_i, y_i)|i = 1 \dots N\}$
3. **Optimization**: use D to find best parameters $\theta^* = \{w^*, b^*\}$

A linear regression problem: an example

A simple problem

Develop a system to predict a person's weight based on his or her height.

To solve this problem

1. **Formulation**: suppose a person's weight y is linearly correlated with his or her height x, i.e., $y = wx + b$
2. **Data**: collect a number of persons' weight and height: $D = \{(x_i, y_i)|i = 1 \dots N\}$
3. **Optimization**: use D to find best parameters $\theta^* = \{w^*, b^*\}$

A linear regression problem: an example

A simple problem

Develop a system to predict a person's weight based on his or her height.

To solve this problem

1. **Formulation**: suppose a person's weight y is linearly correlated with his or her height x, i.e., $y = wx + b$
2. **Data**: collect a number of persons' weight and height: $D = \{(x_i, y_i)|i = 1 \dots N\}$
3. **Optimization**: use D to find best parameters $\theta^* = \{w^*, b^*\}$

A linear regression problem: an example

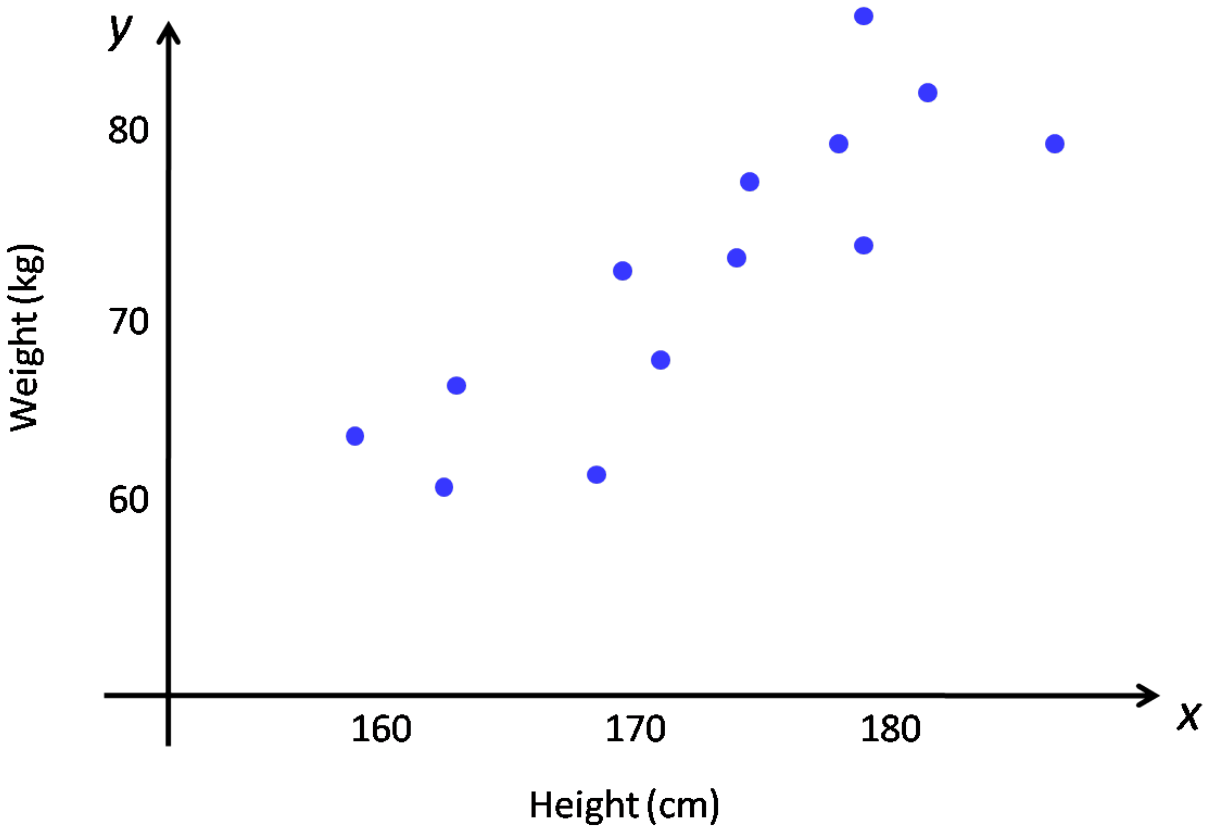
A simple problem

Develop a system to predict a person's weight based on his or her height.

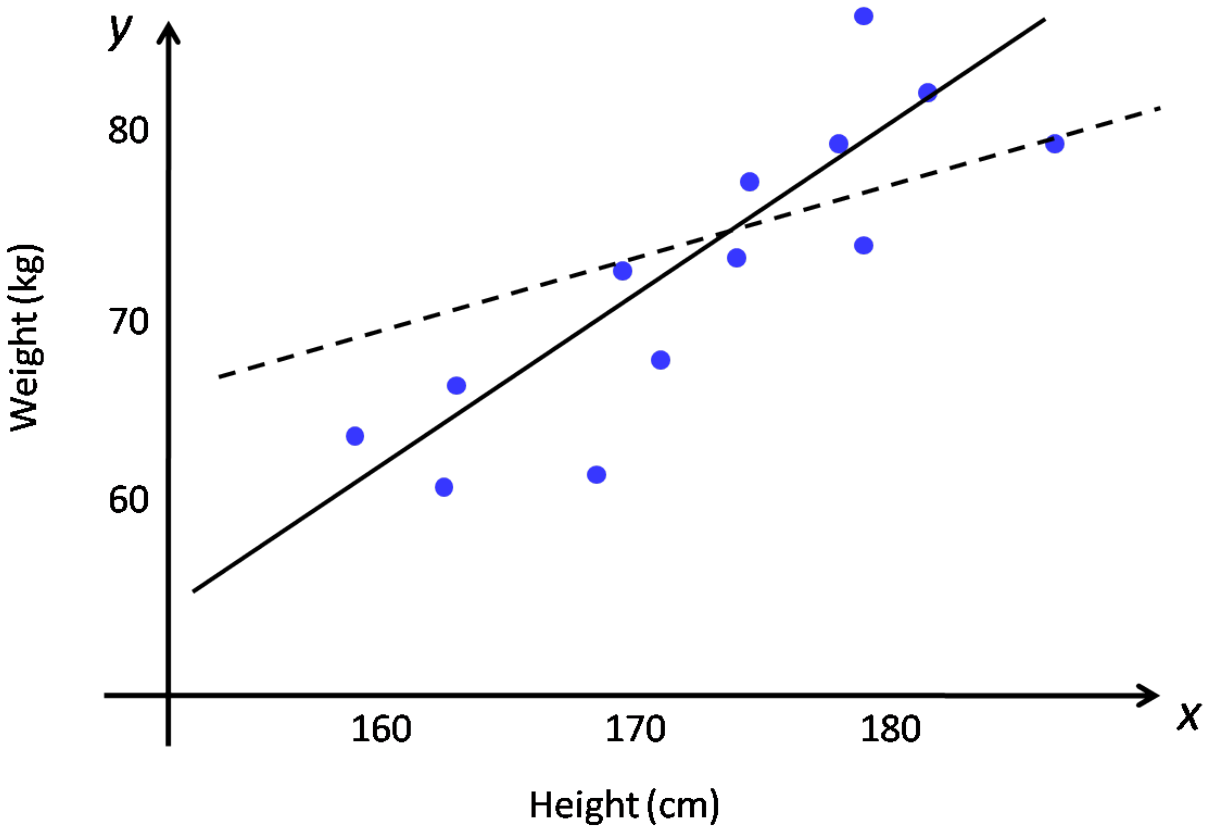
To solve this problem

1. **Formulation**: suppose a person's weight y is linearly correlated with his or her height x, i.e., $y = wx + b$
2. **Data**: collect a number of persons' weight and height: $D = \{(x_i, y_i)|i = 1 \dots N\}$
3. **Optimization**: use D to find best parameters $\boldsymbol{\theta}^* = \{w^*, b^*\}$

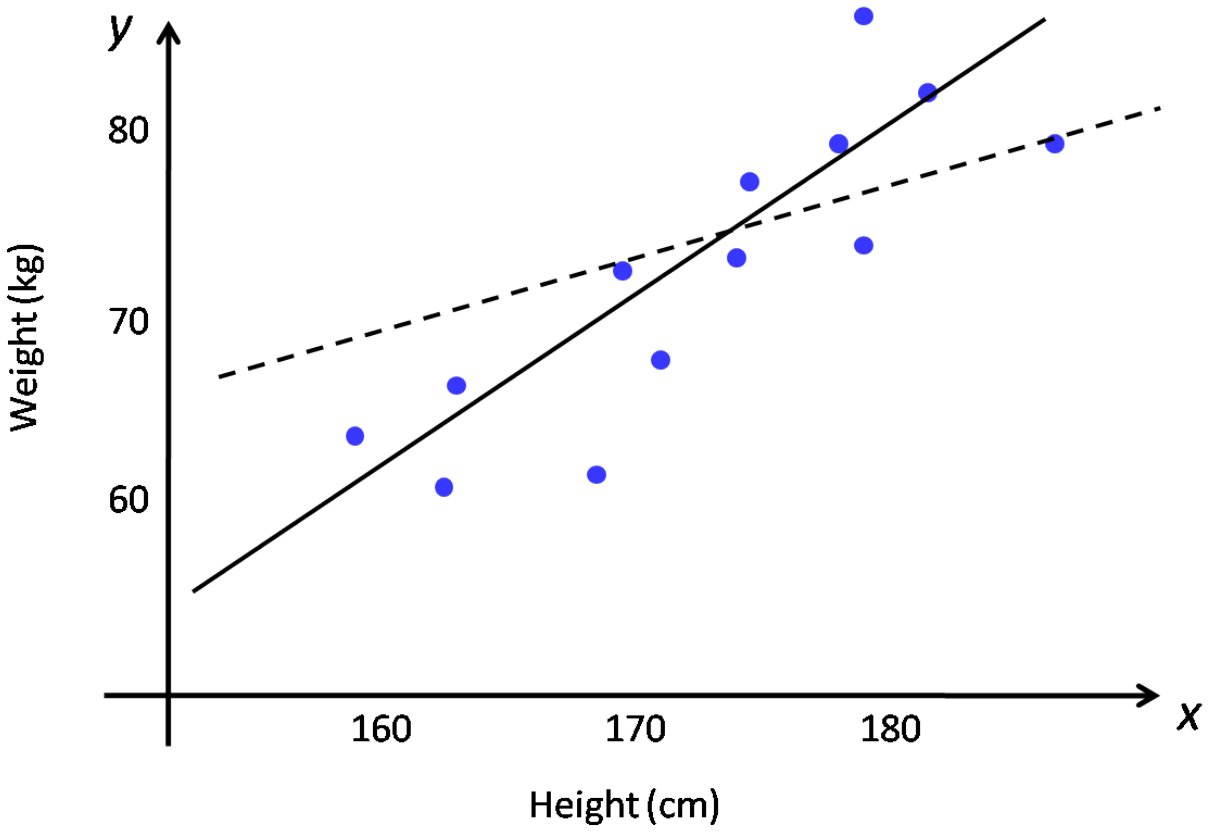
Linear regression



Linear regression (cont')

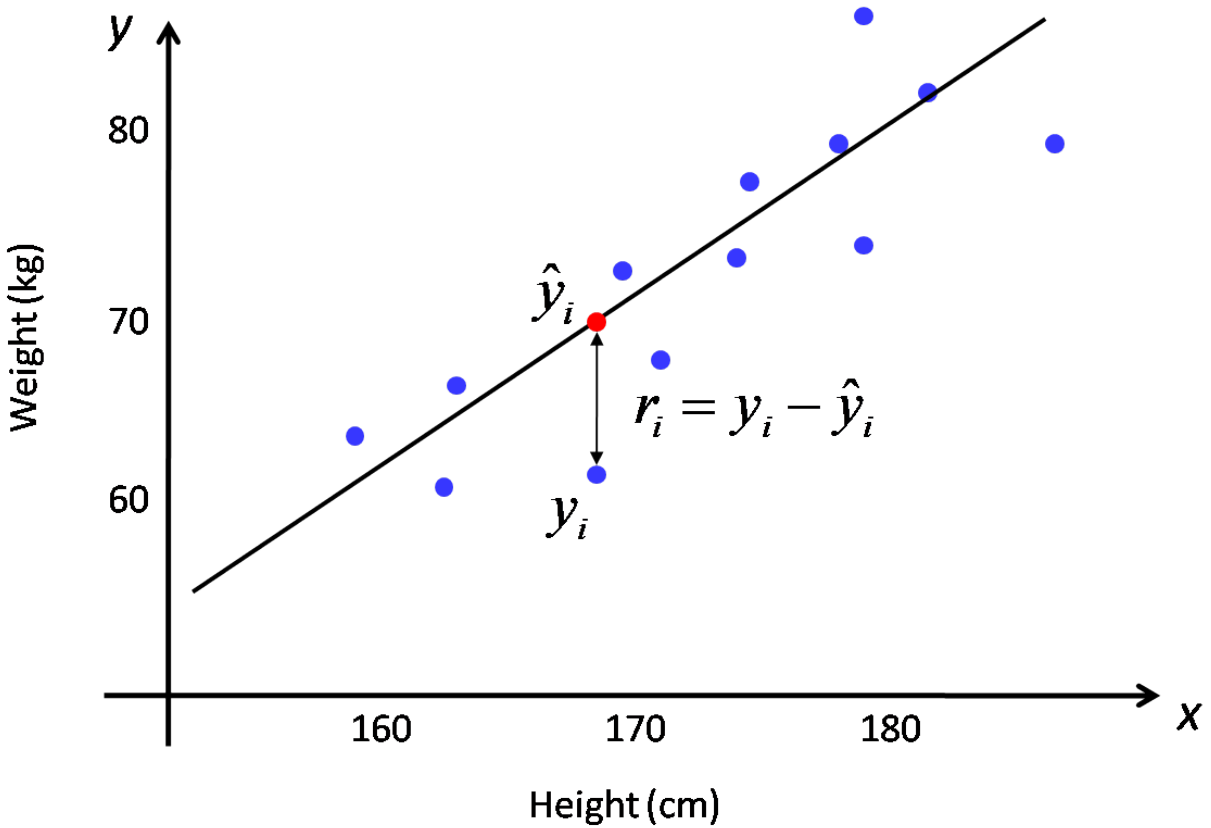


Linear regression (cont')

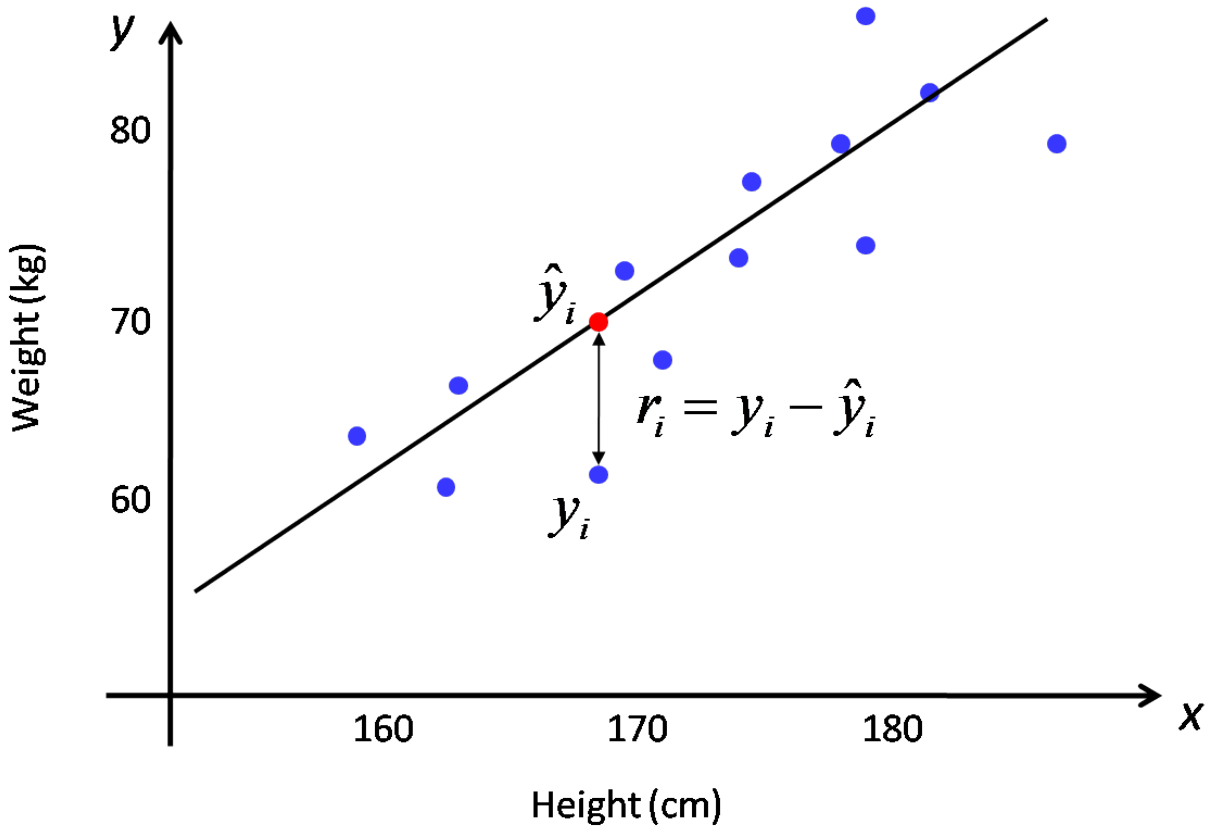


Rule: predicted body weights should be as close to observed body weights as possible.

Linear regression (cont')



Linear regression (cont')



Rule rephrased: Minimization of sum of squared difference r_i^2

Linear regression (cont')

Minimization of sum of squared difference r_i^2

$$\begin{aligned}\min_{\boldsymbol{\theta}} \quad L(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N} r_i^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)_i^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i - b)_i^2\end{aligned}$$

- $\boldsymbol{\theta} = \{w, b\}$ model parameters; L($\boldsymbol{\theta}$) loss function
- Solving a problem = optimizaiton of a loss function with the help of *training data* to find the best model parameter!

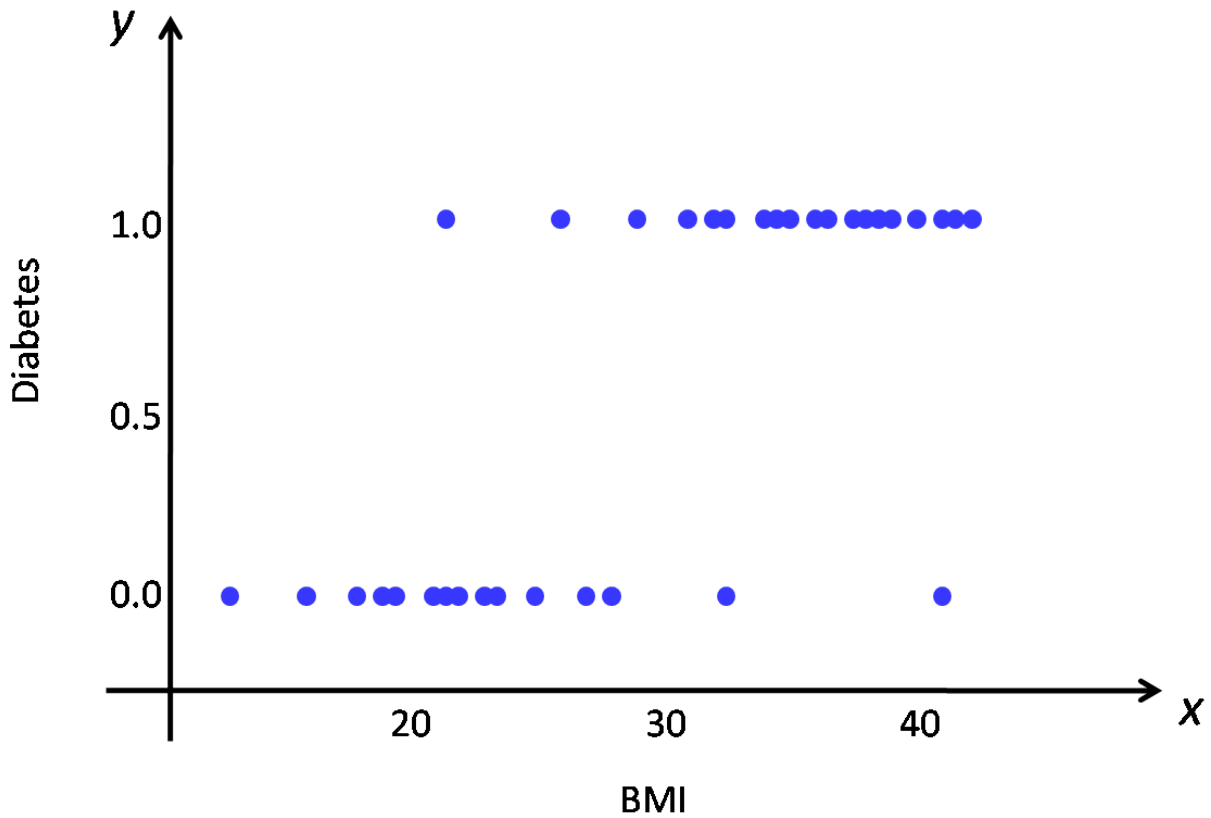
Linear regression (cont')

Minimization of sum of squared difference r_i^2

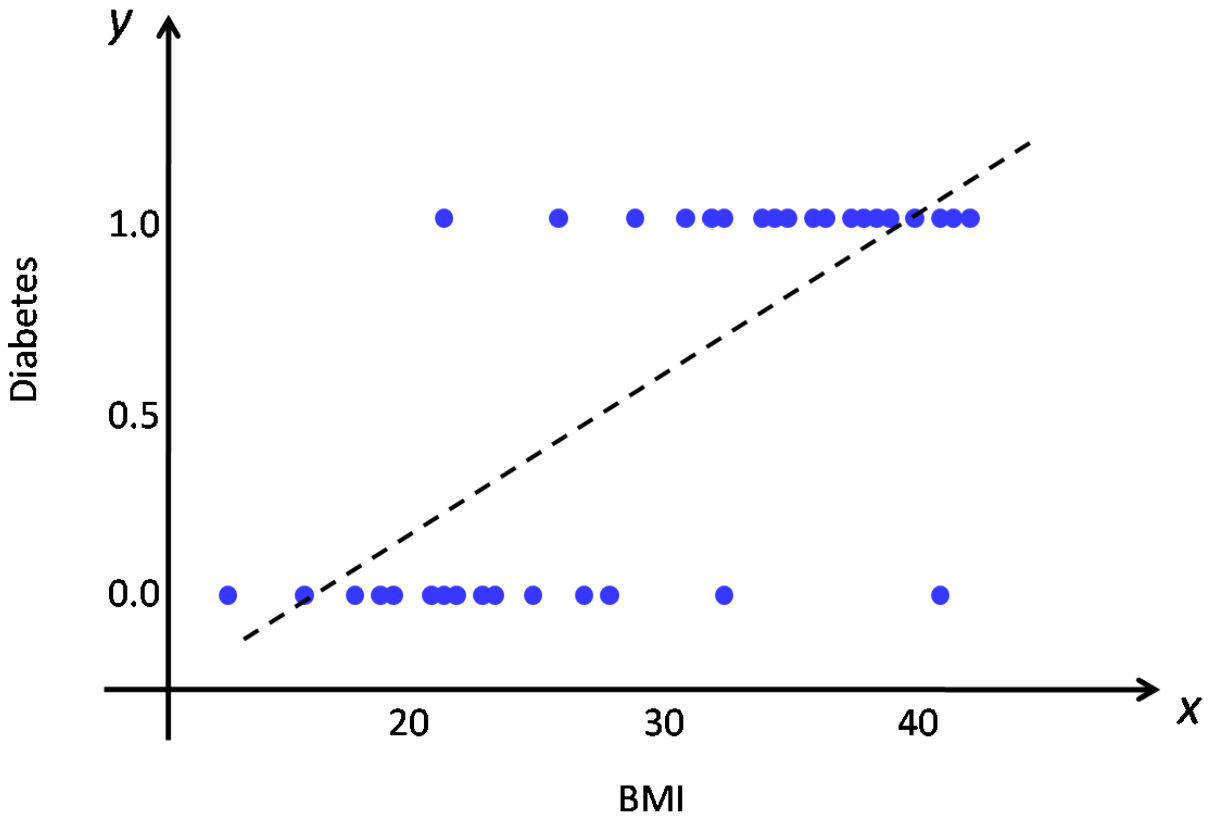
$$\begin{aligned}\min_{\boldsymbol{\theta}} \quad L(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N} r_i^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)_i^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i - b)_i^2\end{aligned}$$

- $\boldsymbol{\theta} = \{w, b\}$ model parameters; L($\boldsymbol{\theta}$) loss function
- Solving a problem = optimizaiton of a loss function with the help of *training data* to find the best model parameter!

What if predicted variable is binary?

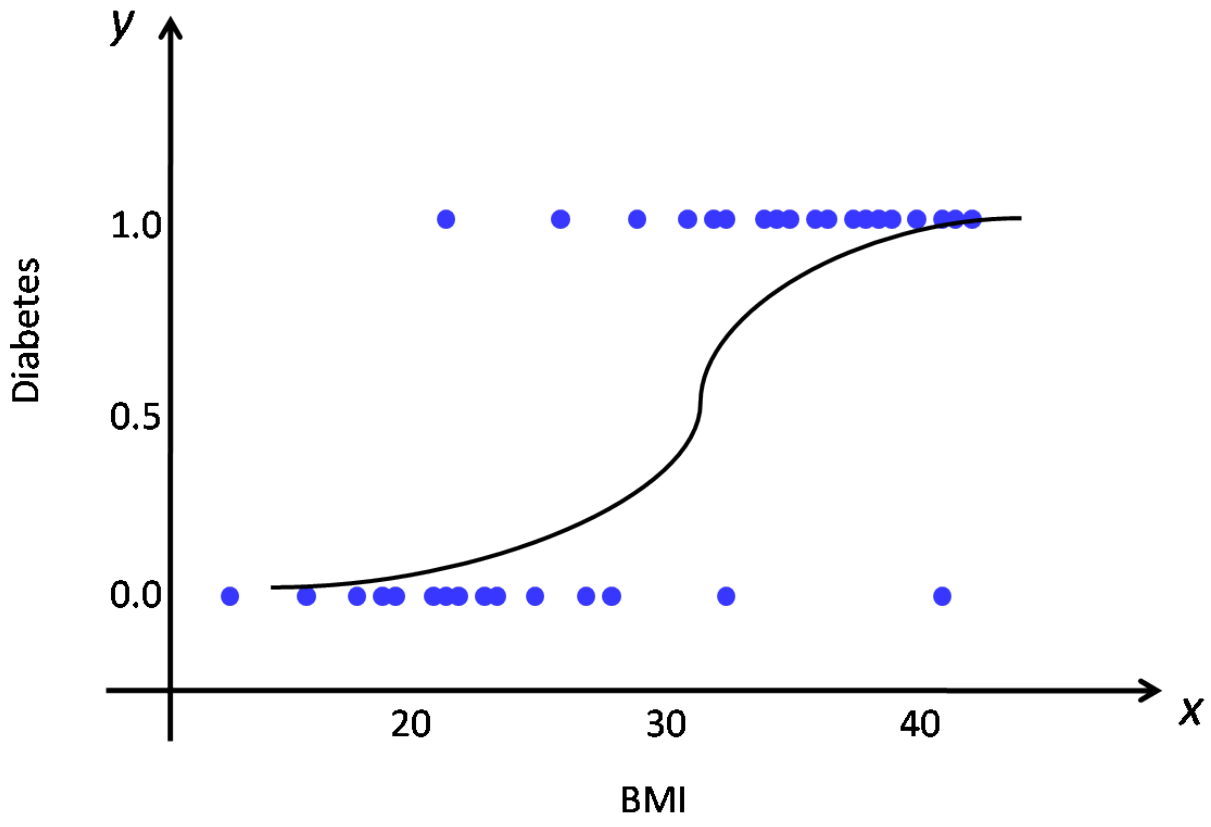


What if predicted variable is binary? (cont')



A linear model is a bad choice!

What if predicted variable is binary? (cont')



A logistic regression (sigmoid) model is often used: $y = \frac{1}{1+e^{-(wx+b)}}$

Further: what if predicted variable is categorical?

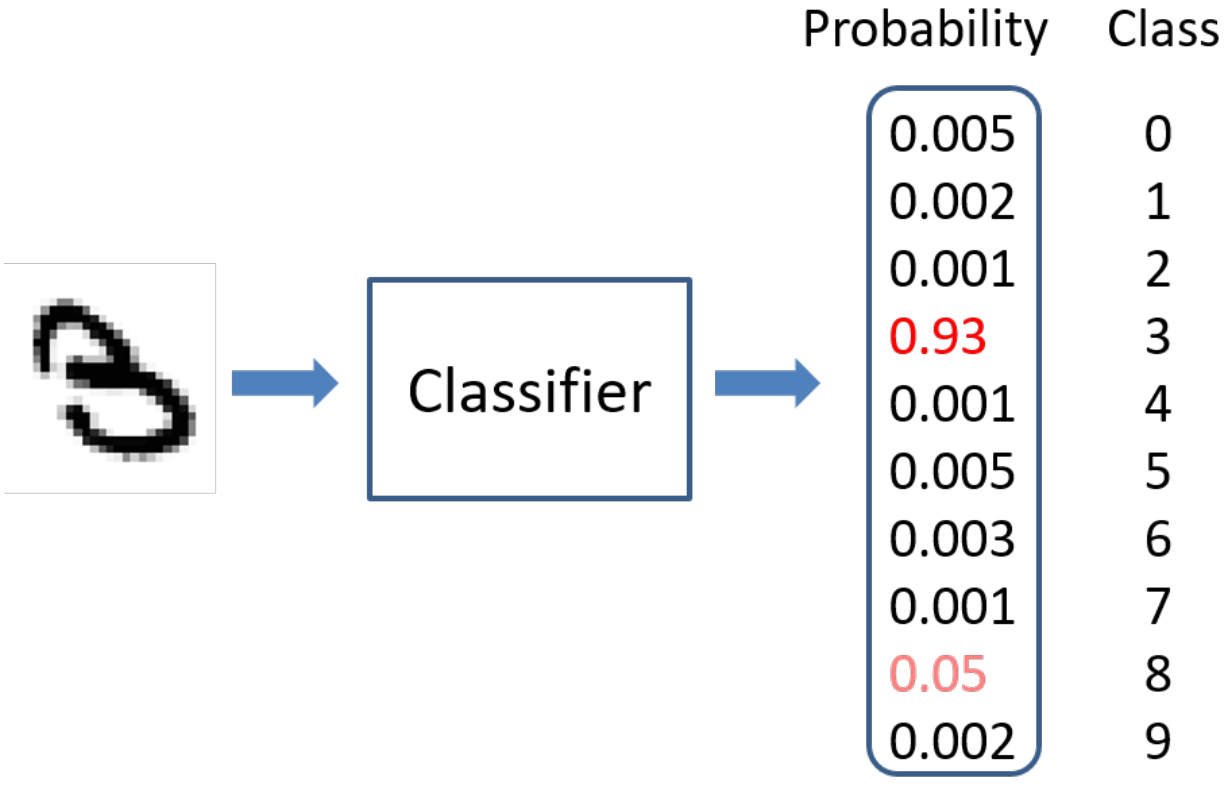
Classification

- Given an input data (e.g., an image, a video clip, recorded voice sequence, a document, etc.), predict the category or class label of the data.
- Number of categories: two or more

Classifier

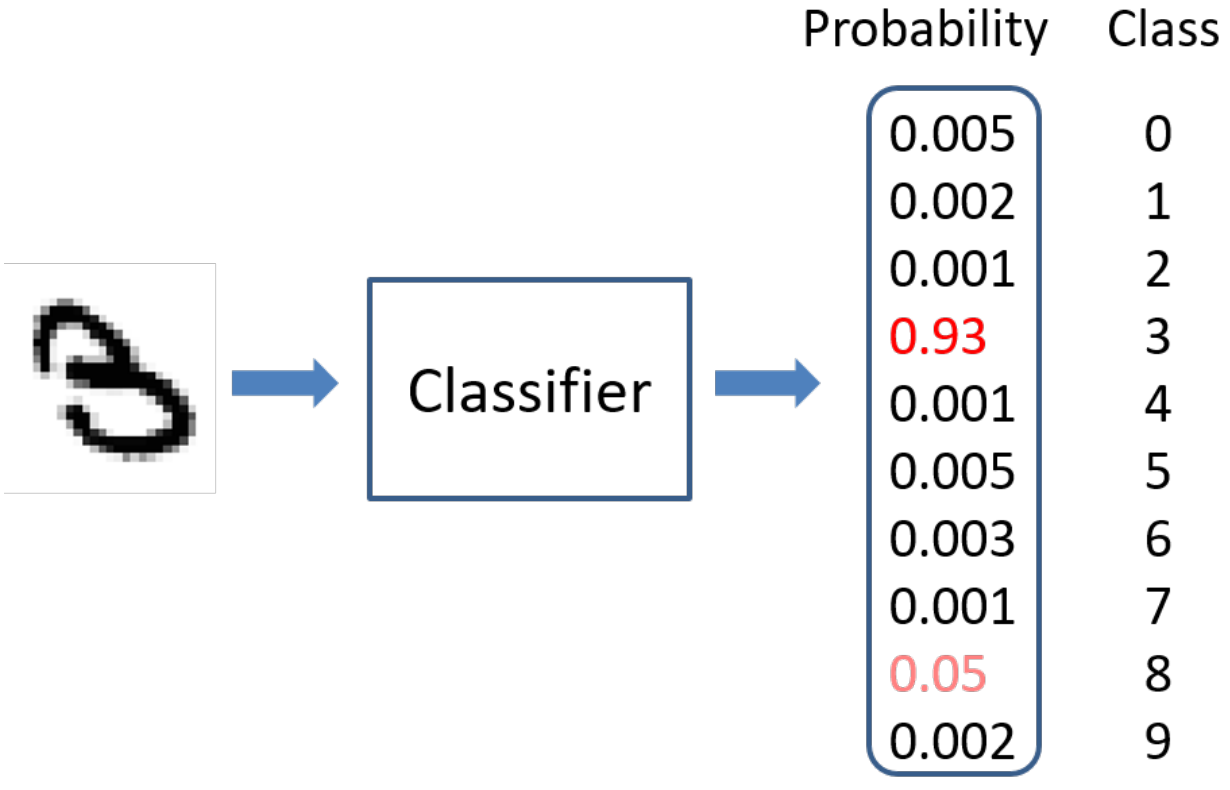
- Given an input data, output the probability of the data belonging to each class.
- So, classifier output is often a probability distribution, i.e., not a single value but a set of values.

What if predicted variable is categorical? (cont')



How to make sure the classifier output is a probability distribution?

What if predicted variable is categorical? (cont')



How to make sure the classifier output is a probability distribution?

What if predicted variable is categorical? (cont')

Softmax function

- Transformation of a K-dimensional vector $\mathbf{z}$ to another K-dimensional vector $\sigma(\mathbf{z})$ in the range [0,1].
- $\sigma(\mathbf{z})$ is a probability distribution: sum up to 1.
- Generalization of the logistic function.
- Also called *normalized exponential function*.

$$\sigma : \quad \mathbb{R}^K \to [0,1]^K$$

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

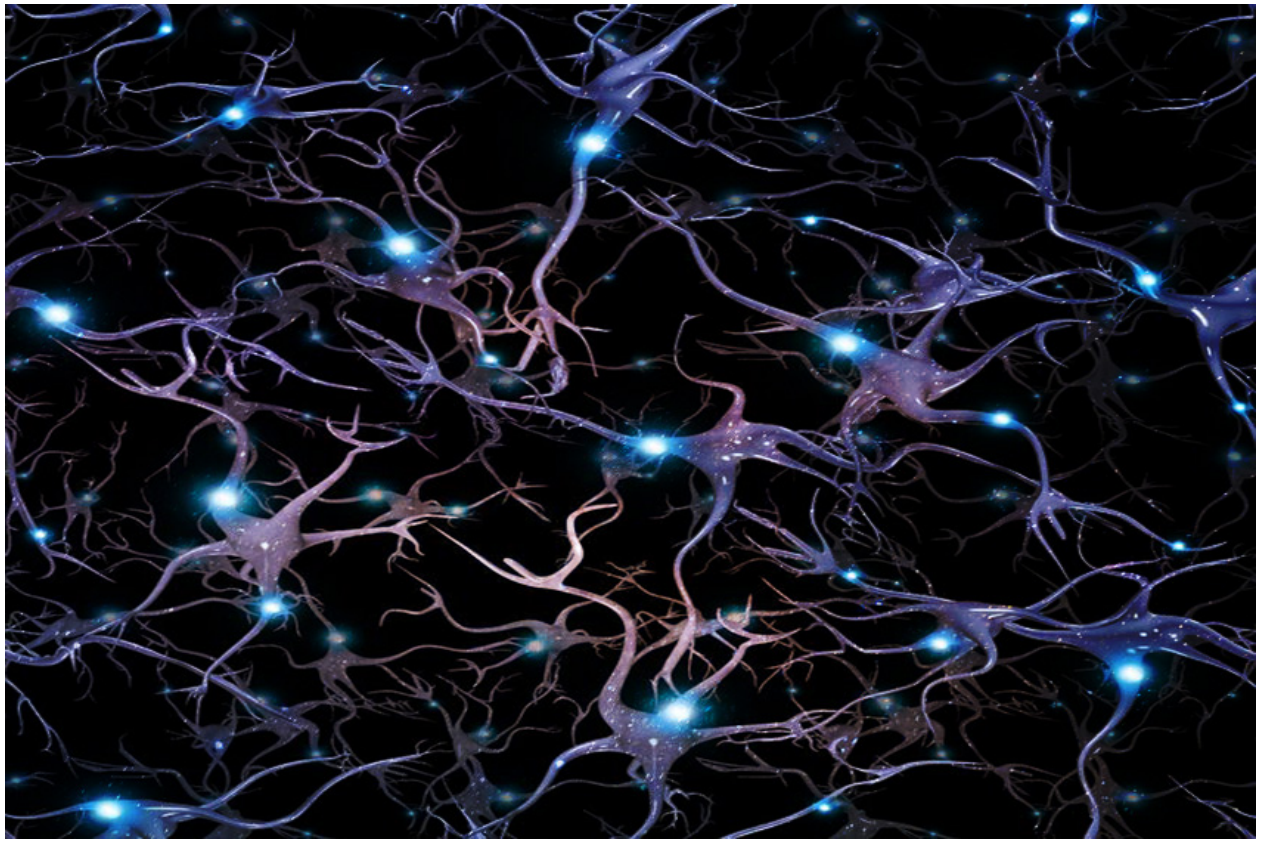
To obtain a classifier

- How to design a classifier model $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$?
- How to design and minimize a loss function $L(\boldsymbol{\theta})$?

To obtain a classifier

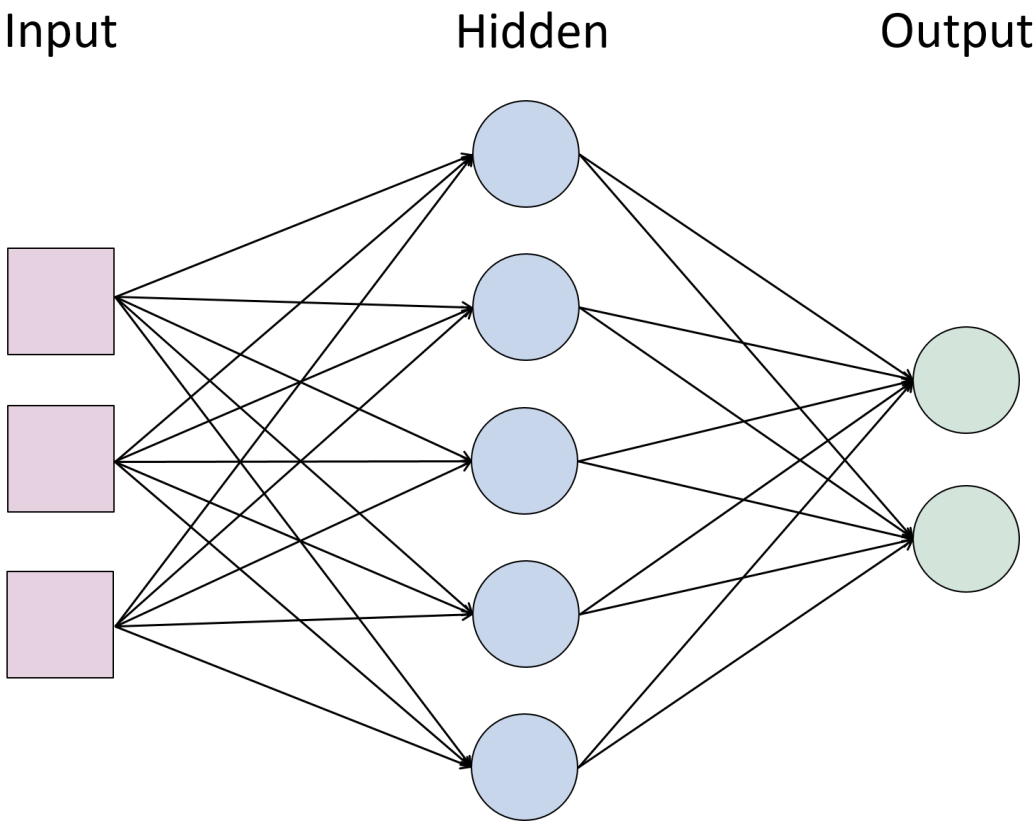
- How to design a classifier model $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$?
  $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$ could be an artificial neural network!

Brain neural network



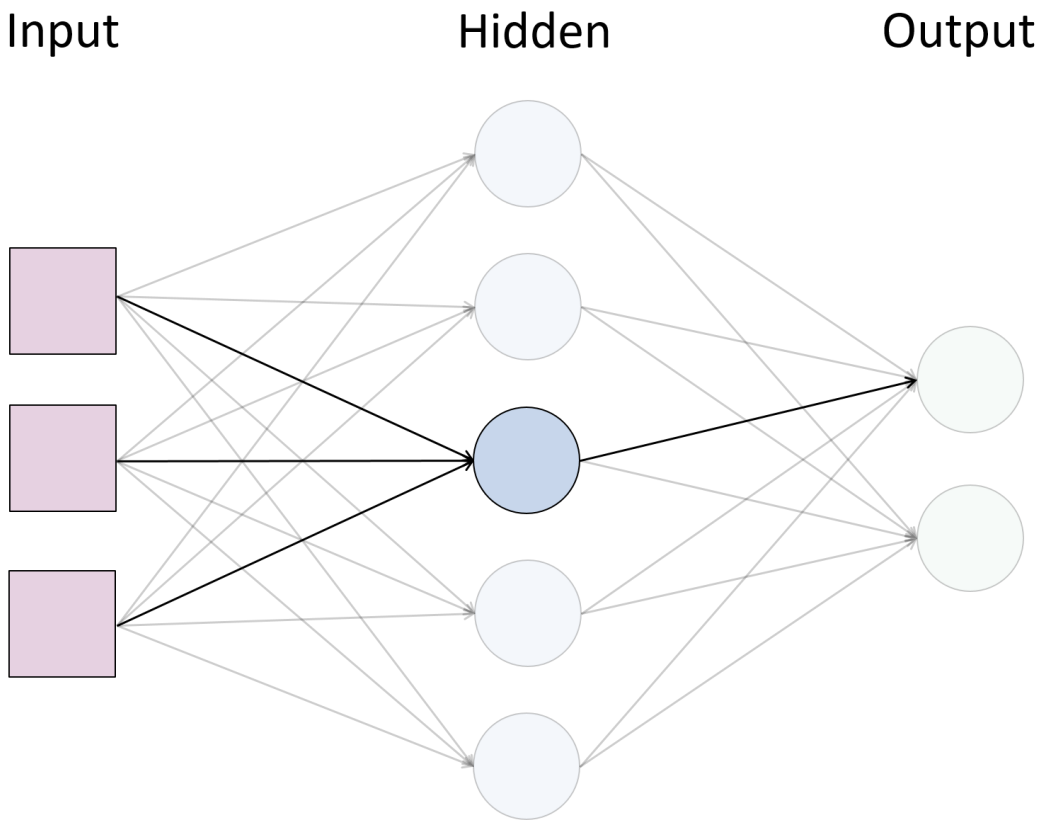
A neuron (i.e., brain cell) receives signals from many neurons, and can be activated and then send signal to many other neurons.

A simple artificial neural network

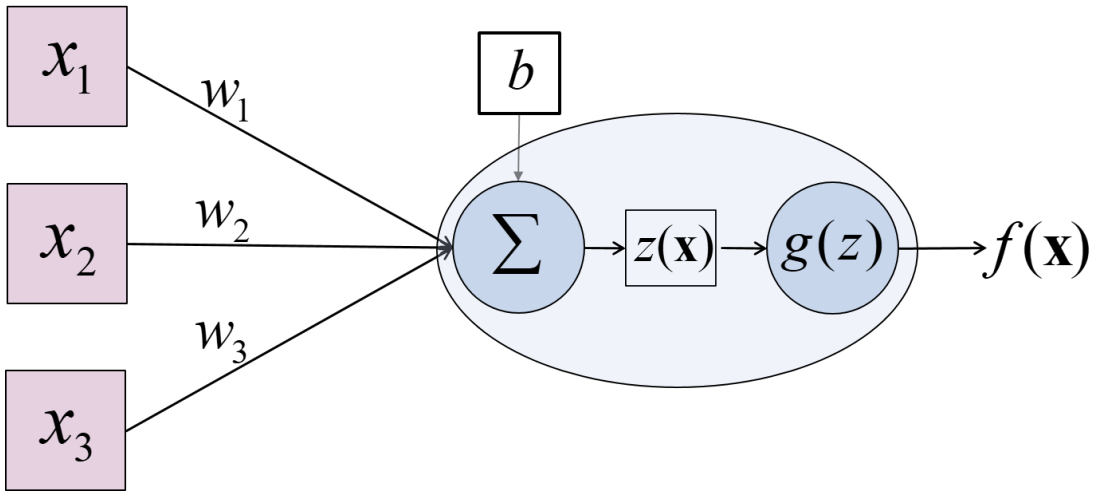


A neuron (i.e., brain cell) receives signals from many neurons, and can be activated and then send signal to many other neurons.

A single neuron



A single neuron (cont')

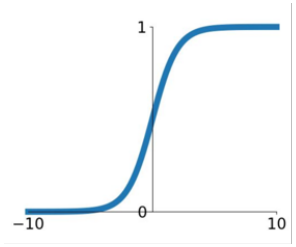


$$z(\mathbf{x}) = \sum_i w_i x_i + b = \mathbf{w}^T \mathbf{x} + b$$

$$f(\mathbf{x}) = g(z) = g(\mathbf{w}^T \mathbf{x} + b)$$

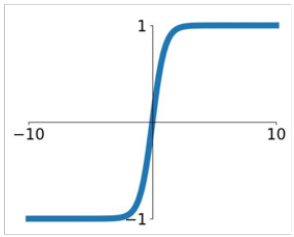
- $\mathbf{x}$ input, $f(\mathbf{x})$ output, $\mathbf{w}$ weights, b bias
- $z(\mathbf{x})$ pre-activation, g activation function
- Model parameters $\boldsymbol{\theta} = \{\mathbf{w}, b\}$

Activation functions



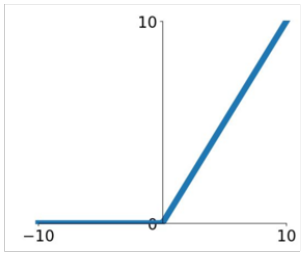
Sigmoid

$$g(z) = \frac{1}{1+e^{-z}}$$



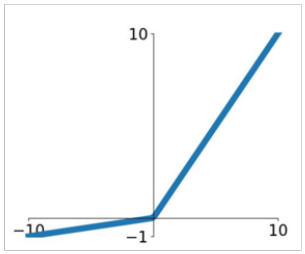
tanh

$$\tanh(z)$$



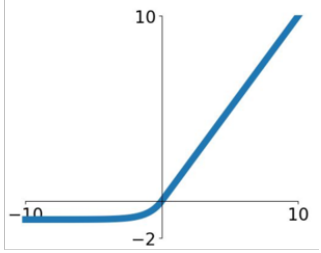
ReLU

$$\max(0, z)$$



Leaky ReLU

$$\max(\alpha z, z)$$



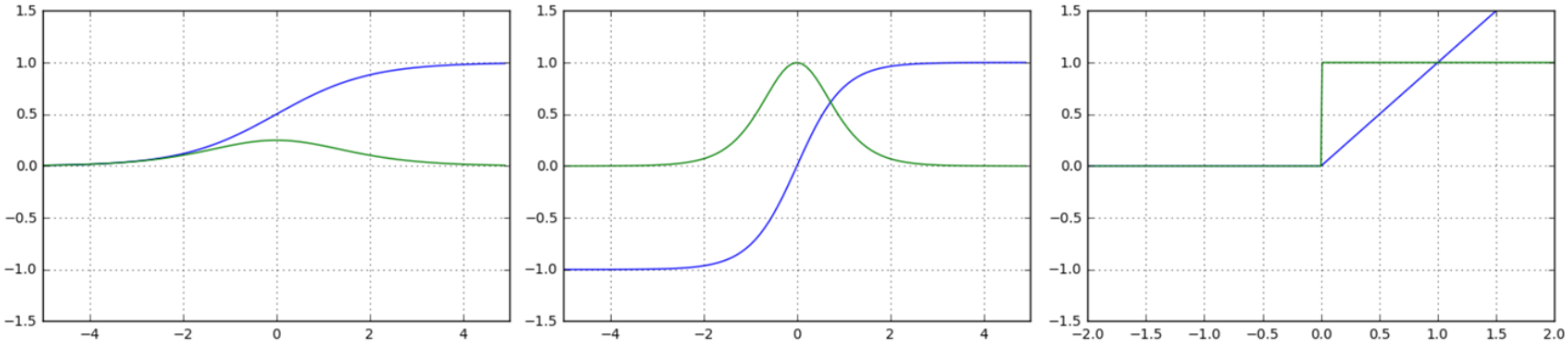
ELU

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

- ReLU: rectified linear unit; ELU: exponential linear unit
- ReLU and its variants Leaky ReLU and ELU are popular now

Activation functions (cont')

Why ReLU is more popular than Sigmoid and tanh?

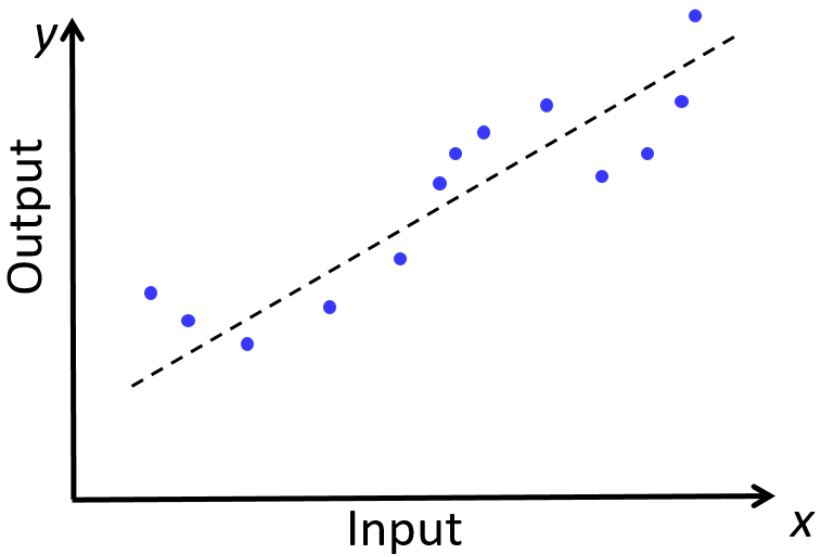


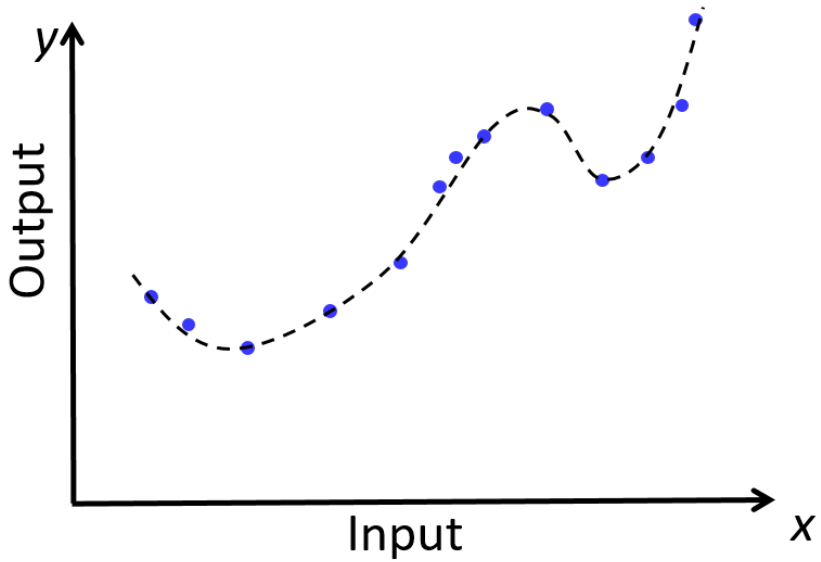
- Blue: activation function; Green: derivative of activation
- Sigmoid & tanh: derivatives become close to zero for larger positive or negative input
- ReLU: derivative is constant (i.e., 1.0) for all positive input, therefore can reduce 'gradient vanishing' issue.

Activation functions (cont')

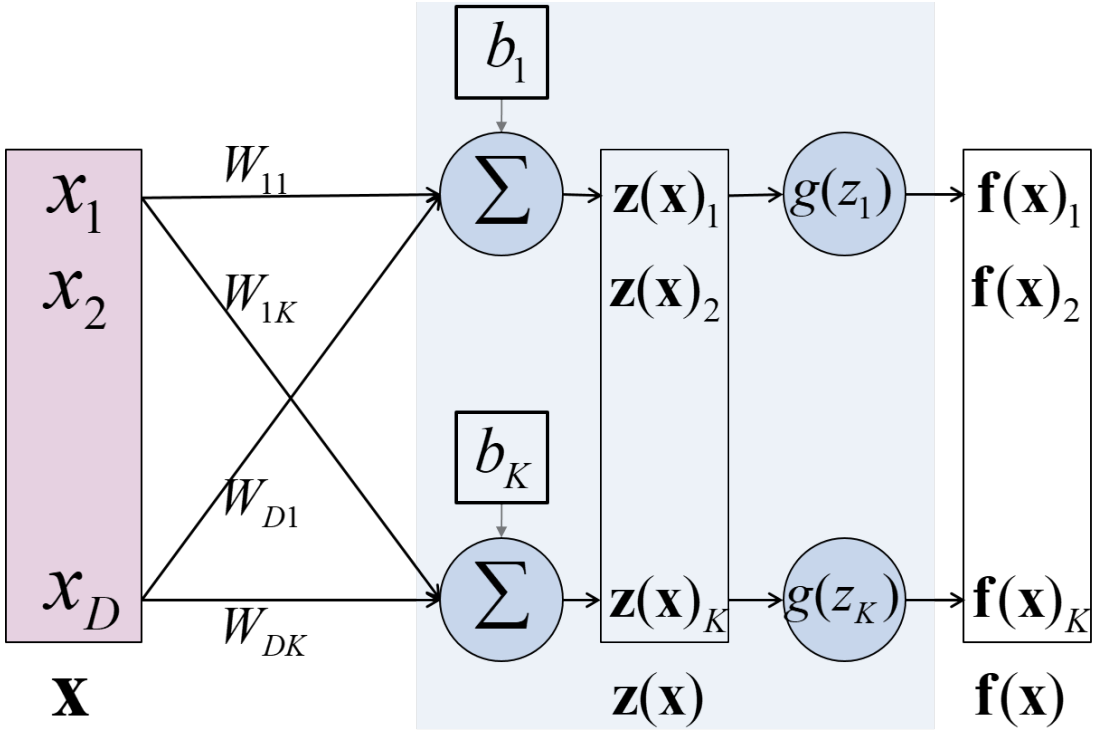
Why are activation functions nonlinear?

- Make the network become a nonlinear function
- Nonlinear function can capture complicated relationships between input and output





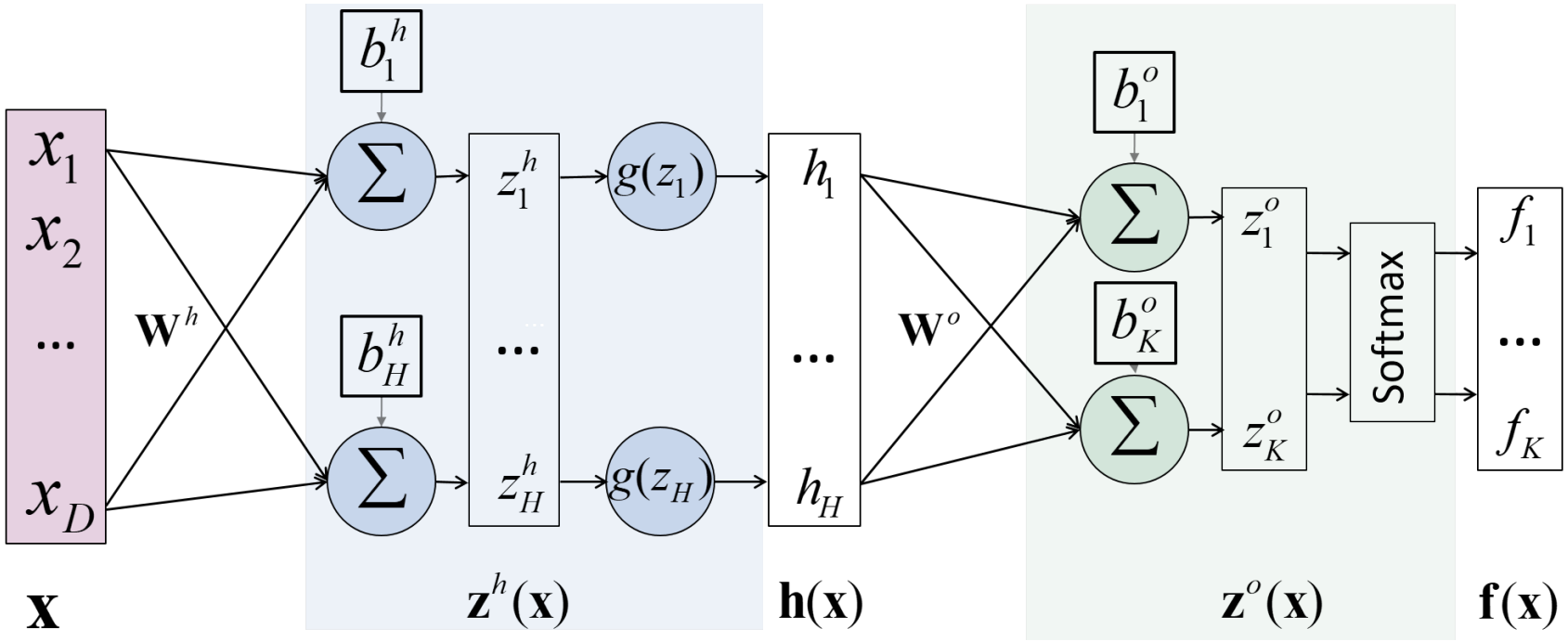
From single neuron to layer of neurons



$$\mathbf{f}(\mathbf{x}) \quad = \quad g(\mathbf{z}(\mathbf{x})) = g(\mathbf{W}\mathbf{x} + \mathbf{b})$$

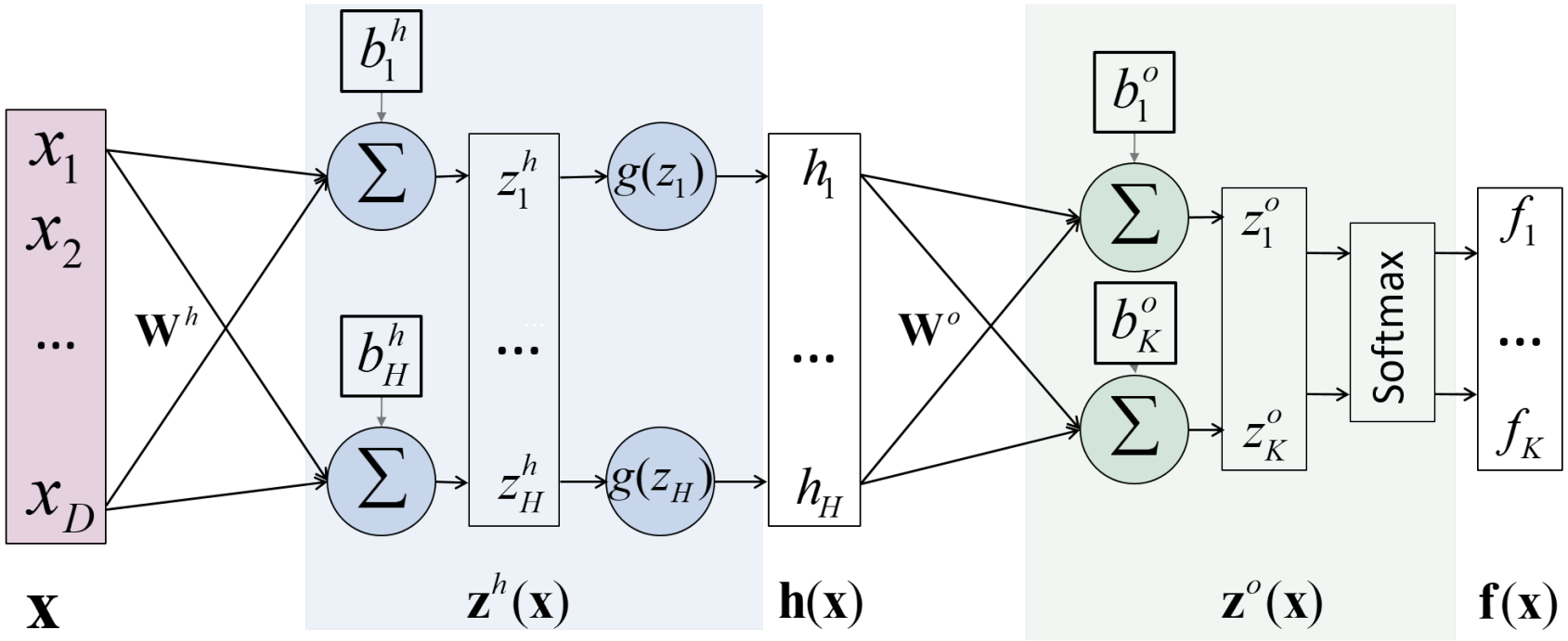
- Model parameters $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$

From one-layer to two-layer network



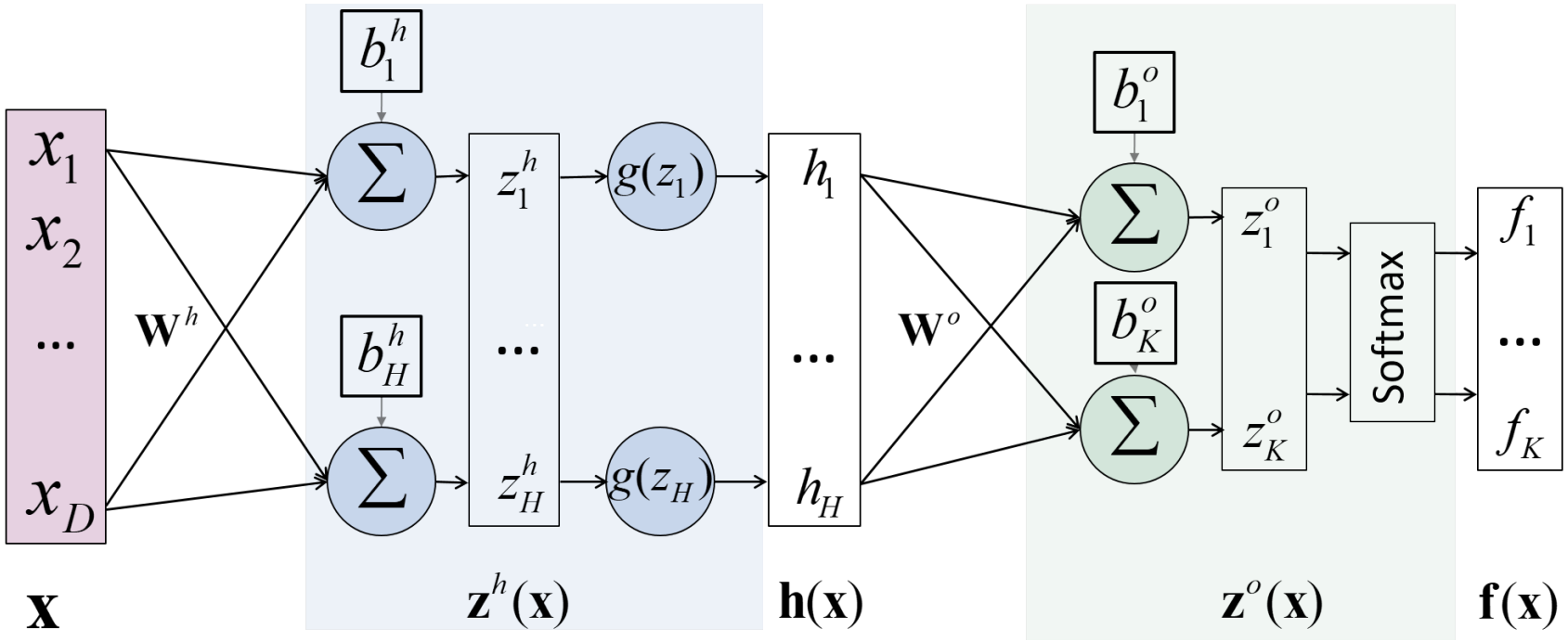
- First layer output $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- Last layer output $\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{z}^o(\mathbf{x})) = \sigma(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$
- Output of 1^{st} layer $\mathbf{h}(\mathbf{x})$ is input to 2^{nd} layer
- Model parameters $\theta = \{\mathbf{W}^h, \mathbf{b}^h, \mathbf{W}^o, \mathbf{b}^o\}$

From one-layer to two-layer network



- First layer output $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- Last layer output $\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{z}^o(\mathbf{x})) = \sigma(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$
- Output of 1^{st} layer $\mathbf{h}(\mathbf{x})$ is input to 2^{nd} layer
- Model parameters $\theta = \{\mathbf{W}^h, \mathbf{b}^h, \mathbf{W}^o, \mathbf{b}^o\}$

From one-layer to two-layer network



- First layer output $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- Last layer output $\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{z}^o(\mathbf{x})) = \sigma(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$
- Output of 1^{st} layer $\mathbf{h}(\mathbf{x})$ is input to 2^{nd} layer
- Model parameters $\boldsymbol{\theta} = \{\mathbf{W}^h, \mathbf{b}^h, \mathbf{W}^o, \mathbf{b}^o\}$

After we designed the classifier using a network...

How do we teach (i.e., train) the classifier?

Classifier training

- Find optimal parameters $\boldsymbol{\theta}^*$ of a classifier $\mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$
- Rule: given input $\mathbf{x}$, classifier output $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$ should be as close to the *ideal output* as possible.

Classifier training

- Find optimal parameters $\boldsymbol{\theta}^*$ of a classifier $\mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$
- Rule: given input $\mathbf{x}$, classifier output $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$ should be as close to the *ideal output* as possible.

Classifier

Probability	Ideal output	Class
0.005	0	0
0.002	0	1
0.001	0	2
0.93	1	3
0.001	0	4
0.005	0	5
0.003	0	6
0.001	0	7
0.05	0	8
0.002	0	9

Cross entropy loss

- Classifier output and ideal output are two probability distributions.
- How to measure difference between two distributions? **Cross entropy!**
- Given input $\mathbf{x}_i$
  Classifier output $\hat{\mathbf{y}}_i = \mathbf{f}(\mathbf{x}_i; \boldsymbol{\theta}) = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iK})$
  Ideal output $\mathbf{y}_i = (y_{i1}, \ldots, y_{iK}) = (0, \ldots, 1, \ldots, 0)$

$$
\begin{aligned}
l(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i; \boldsymbol{\theta})) &= \sum_{k=1}^{K} y_{ik} \log \frac{1}{\hat{y}_{ik}} \\
&= -\sum_{k=1}^{K} y_{ik} \log \hat{y}_{ik}
\end{aligned}
$$

Cross entropy loss to train a classifier

- Collect a set of training data $D = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \ldots, N\}$
- Rule: find best parameters with minimum cross entropy loss.

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad L(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^{N} l(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i; \boldsymbol{\theta})) \\ &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \hat{y}_{ik} \end{aligned}$$

Note: cross entropy loss = negative log likelihood!

Minimizing loss function: gradient descent

Gradient descent

An iterative process:

1. Start from random parameters $\boldsymbol{\theta}_0$
2. For the $(n+1)^{th}$ iteration, compute gradient $\nabla L(\boldsymbol{\theta}_n)$
3. Update parameters $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - r\nabla L(\boldsymbol{\theta}_n)$

Suppose $\boldsymbol{\theta}_n = (\theta_{n1}, \theta_{n2}, \ldots, \theta_{nD},)^T$, gradient of loss function $L(\theta)$ at $\boldsymbol{\theta}_n$ is defined as $\nabla L(\boldsymbol{\theta}_n) = (\frac{\partial L}{\partial \theta_{n1}}, \frac{\partial L}{\partial \theta_{n2}}, \ldots, \frac{\partial L}{\partial \theta_{nD}})^T$.

r is learning rate, a hyper-parameter to be tuned.

Minimizing loss function: gradient descent

Gradient descent

An iterative process:

1. Start from random parameters $\boldsymbol{\theta}_0$
2. For the $(n+1)^{th}$ iteration, compute gradient $\nabla L(\boldsymbol{\theta}_n)$
3. Update parameters $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - r\nabla L(\boldsymbol{\theta}_n)$

Suppose $\boldsymbol{\theta}_n = (\theta_{n1}, \theta_{n2}, \ldots, \theta_{nD},)^T$, gradient of loss function $L(\theta)$ at $\boldsymbol{\theta}_n$ is defined as $\nabla L(\boldsymbol{\theta}_n) = (\frac{\partial L}{\partial \theta_{n1}}, \frac{\partial L}{\partial \theta_{n2}}, \ldots, \frac{\partial L}{\partial \theta_{nD}})^T$.

r is learning rate, a hyper-parameter to be tuned.

Minimizing loss function: gradient descent (cont')

Why can gradient descent find local minimum?

If learning rate r is small enough, then $L(\boldsymbol{\theta}_{n+1}) \leq L(\boldsymbol{\theta}_n)$.

Proof.

1. Talor series $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$
2. Replace $f(x)$ by $L(\boldsymbol{\theta}_{n+1})$, replace a by $\boldsymbol{\theta}_n$, we get

$$\begin{aligned} L(\boldsymbol{\theta}_{n+1}) &= L(\boldsymbol{\theta}_n) + \nabla L(\boldsymbol{\theta}_n)^{\mathrm{T}}(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) + \\ & \quad (\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n)^{\mathrm{T}} \mathbf{H}(\boldsymbol{\theta}_n)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) + \dots \end{aligned}$$

3. Since $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - r\nabla L(\boldsymbol{\theta}_n)$, then

$$\begin{aligned} L(\boldsymbol{\theta}_{n+1}) &= L(\boldsymbol{\theta}_n) - r\|\nabla L(\boldsymbol{\theta}_n)\|^2 + \mathcal{O}(r^2\|\nabla L(\boldsymbol{\theta}_n)\|^2) \\ &\leq L(\boldsymbol{\theta}_n) \end{aligned}$$

Minimizing loss function: gradient descent (cont')

Why can gradient descent find local minimum?

If learning rate r is small enough, then $L(\boldsymbol{\theta}_{n+1}) \leq L(\boldsymbol{\theta}_n)$.

Proof.

1. Talor series $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots$
2. Replace $f(x)$ by $L(\boldsymbol{\theta}_{n+1})$, replace a by $\boldsymbol{\theta}_n$, we get

$$\begin{aligned} L(\boldsymbol{\theta}_{n+1}) &= L(\boldsymbol{\theta}_n) + \nabla L(\boldsymbol{\theta}_n)^{\mathsf{T}}(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) + \\ & \quad (\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n)^{\mathsf{T}}\mathbf{H}(\boldsymbol{\theta}_n)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) + \ldots \end{aligned}$$

3. Since $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - r\nabla L(\boldsymbol{\theta}_n)$, then

$$\begin{aligned} L(\boldsymbol{\theta}_{n+1}) &= L(\boldsymbol{\theta}_n) - r\|\nabla L(\boldsymbol{\theta}_n)\|^2 + \mathcal{O}(r^2\|\nabla L(\boldsymbol{\theta}_n)\|^2) \\ &\leq L(\boldsymbol{\theta}_n) \end{aligned}$$

Minimizing loss function: gradient descent (cont')

Why can gradient descent find local minimum?

If learning rate r is small enough, then $L(\boldsymbol{\theta}_{n+1}) \leq L(\boldsymbol{\theta}_n)$.

Proof.

1. Talor series $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots$
2. Replace $f(x)$ by $L(\boldsymbol{\theta}_{n+1})$, replace a by $\boldsymbol{\theta}_n$, we get

$$\begin{aligned} L(\boldsymbol{\theta}_{n+1}) &= L(\boldsymbol{\theta}_n) + \nabla L(\boldsymbol{\theta}_n)^{\mathsf{T}}(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) + \\ & \quad (\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n)^{\mathsf{T}}\mathbf{H}(\boldsymbol{\theta}_n)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) + \ldots \end{aligned}$$

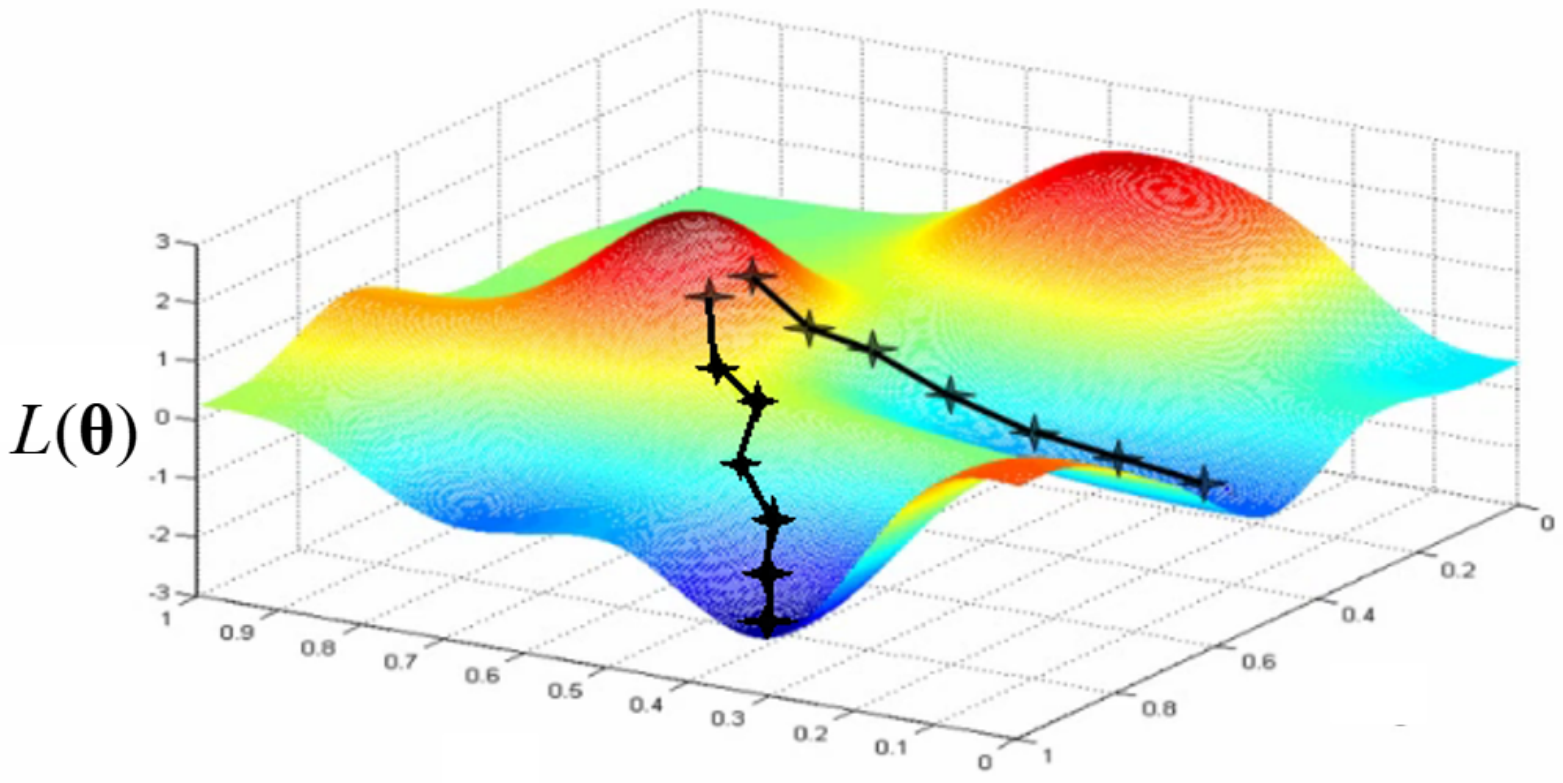
3. Since $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - r\nabla L(\boldsymbol{\theta}_n)$, then

$$\begin{aligned} L(\boldsymbol{\theta}_{n+1}) &= L(\boldsymbol{\theta}_n) - r\|\nabla L(\boldsymbol{\theta}_n)\|^2 + \mathcal{O}(r^2\|\nabla L(\boldsymbol{\theta}_n)\|^2) \\ &\leq L(\boldsymbol{\theta}_n) \end{aligned}$$

Gradient descent tells how to go downhill efficiently!

picture from Stanford CS231n

Gradient descent can help find local minimum!



Summary

- Applications of deep learning spread into most domains
- How to solve a problem? Formulation, data, optimization!
- Artificial neural network has a layer-by-layer structure
- Each layer consists of a series of simple math functions.
- Gradient descent can be used to train a neural network.

Further reading: Chapter 1, in textbook "Deep learning", http://www.deeplearningbook.org/

Summary

- Applications of deep learning spread into most domains
- How to solve a problem? Formulation, data, optimization!
- Artificial neural network has a layer-by-layer structure
- Each layer consists of a series of simple math functions.
- Gradient descent can be used to train a neural network.

Further reading: Chapter 1, in textbook "Deep learning", http://www.deeplearningbook.org/

How to summarize a paper?

For technical papers:

- what is the problem to be tackled?
- what is the difficulty to solve the problem?
- how is the problem solved in the paper?
- what is the novelty in the paper, particularly in methodology?
- what are the significant experimental results?
- what are the advantages and disadvantages of the method/solution proposed in the paper?
- how can the proposed method be further developed?