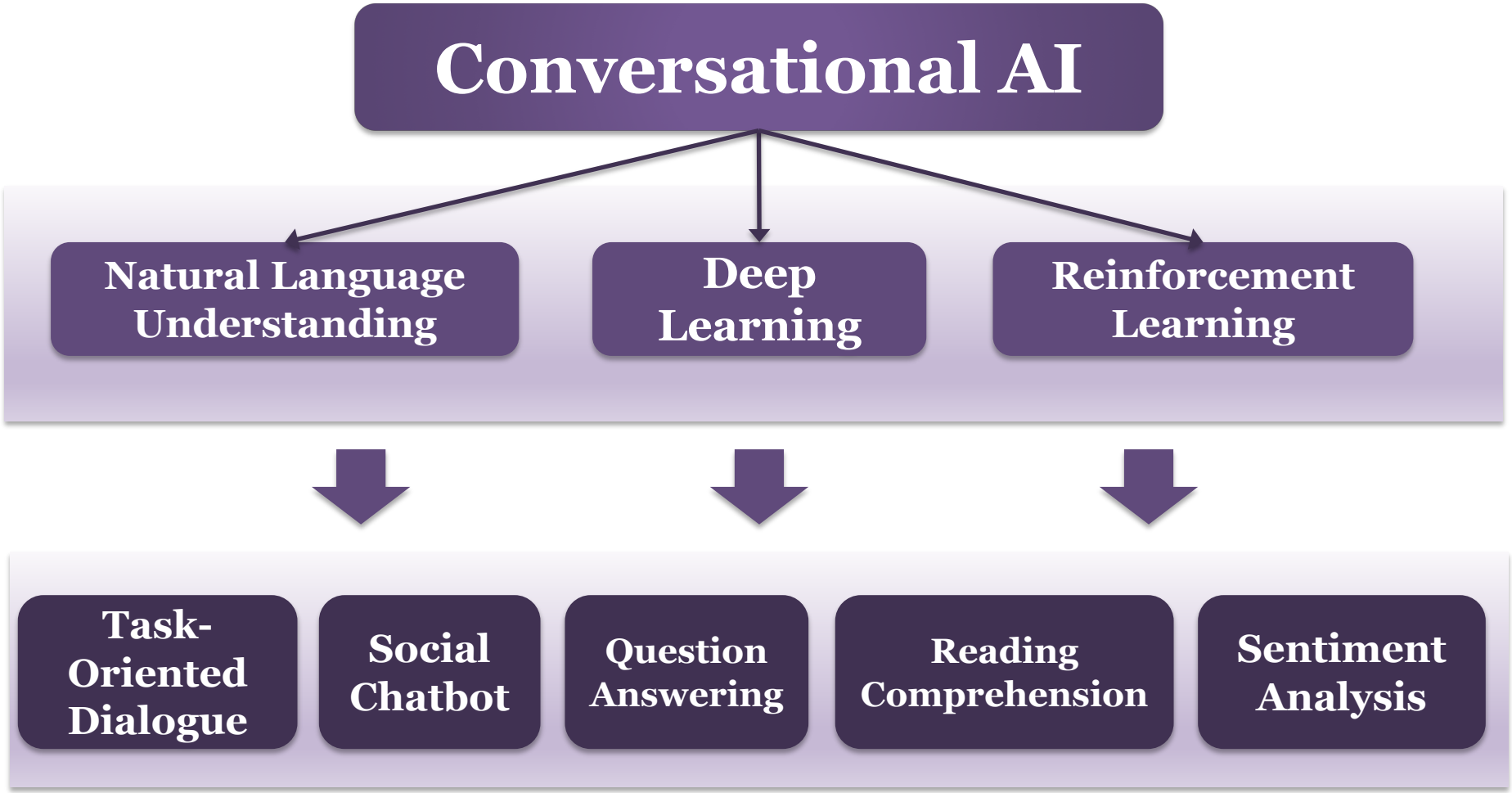
Tsinghua Deep Learning Summer School
2018/7/23

Introduction to Deep Learning

Minlie Huang
aihuang@tsinghua.edu.cn
http://coai.cs.tsinghua.edu.cn/hml

Research Picture of the CoAI Group

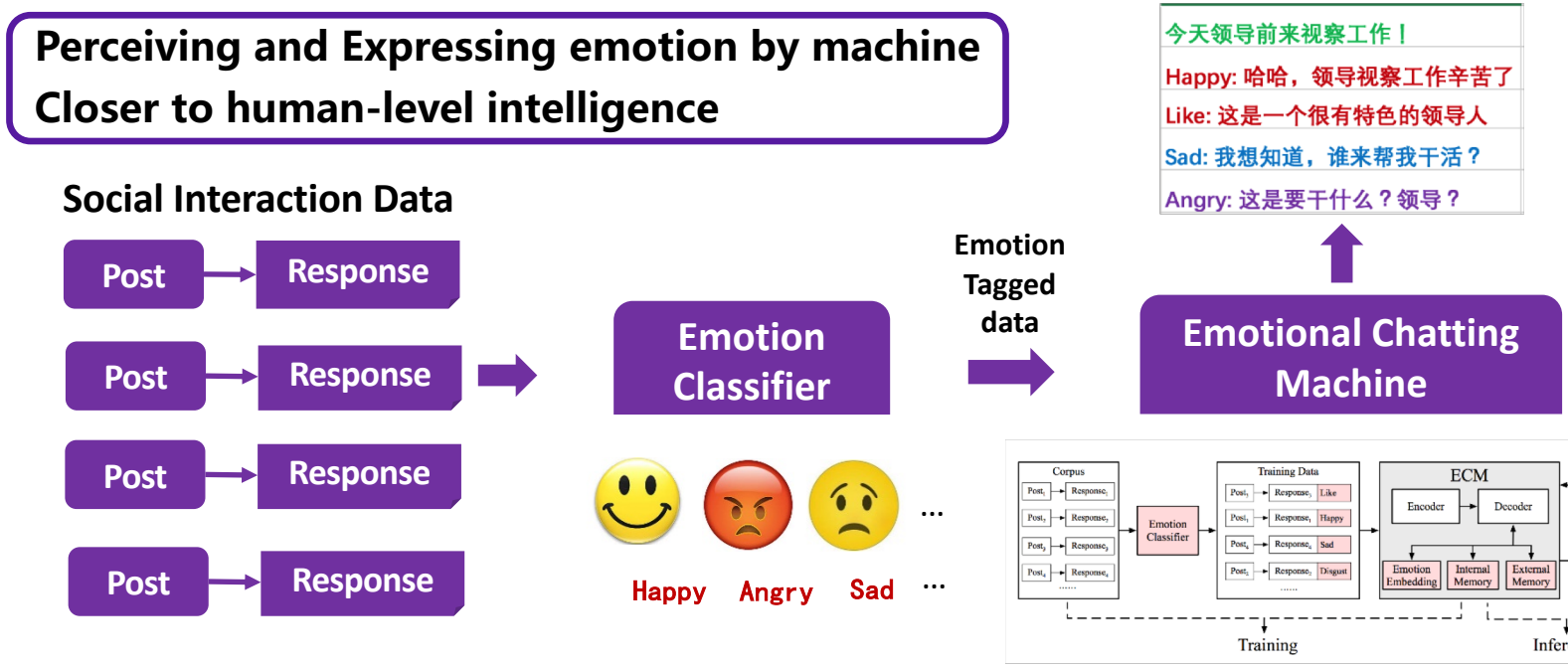


Major Research Directions

- **Sentiment & Emotion Understanding**
  - EMNLP 2016, ACL 2017, TOIS 2017, WWW 2018/2019
- **Language Generation**
  - ACL 2018, AAAI 2018/2019, IJCAI 2018/2019, WWW 2019
- **Dialog Systems**
  - ACL 2018, SIGIR 2018/2019, AAAI 2018/2019, IJCAI 2018/2019
- **Reinforcement Learning in NLP & Search**
  - WWW 2018/2019, AAAI 2018/2019, IJCAI 2018/2019

Emotional Chatting Machine

Perceiving and Expressing emotion by machine
Closer to human-level intelligence

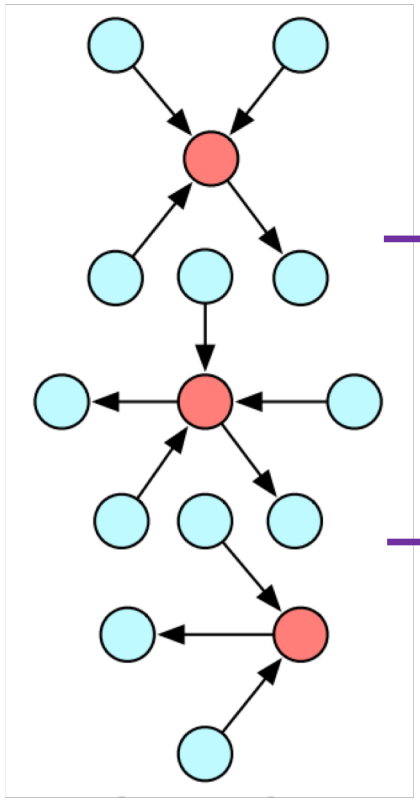


Our work was reported by **MIT Technology Review, the Guardian, Cankao News, Xinhua News Agency etc.**

Prof Björn Schuller: "an important step" towards personal assistants that could read the emotional undercurrent of a conversation and respond with something akin to empathy.

•Hao Zhou, Minlie Huang, Xiaoyan Zhu, Bing Liu. Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory. **AAAI 2018.**

Commonsense-aware Chatbot



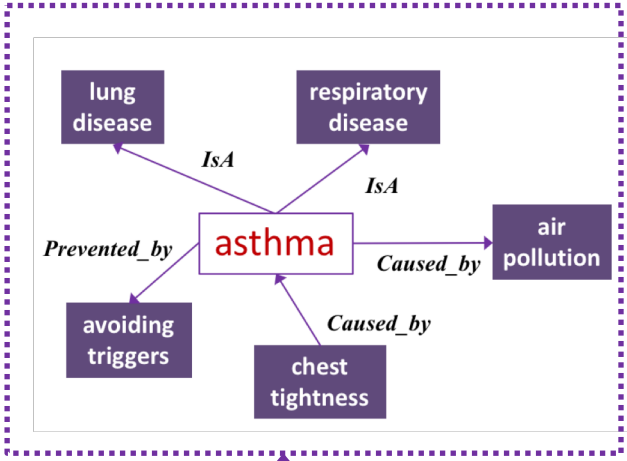
Output: It is good for you to avoid **triggers**.

Graph Attention

Knowledge Aware Decoder

Graph Embedding

Knowledge Aware Encoder



Input: I have **asthma** since three years old.

Commonsense Knowledge Aware Conversation Generation with Graph Attention. **IJCAI-ECAI 2018 distinguished paper** (3470 submissions; 710 accepted)

Outline

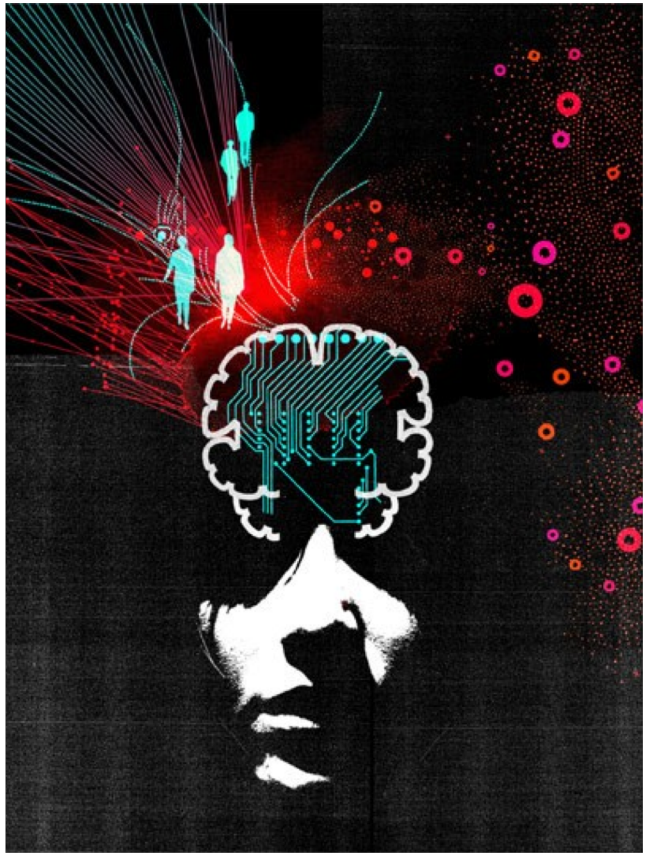
- Overview of deep learning
- Math basics
- Machine learning basics

MIT Technology Review

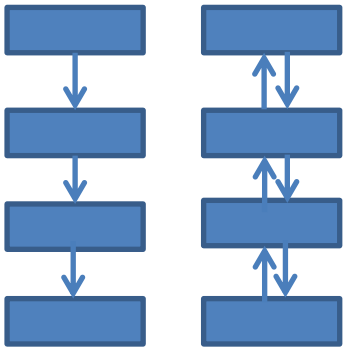
10 Breakthrough Technologies 2013

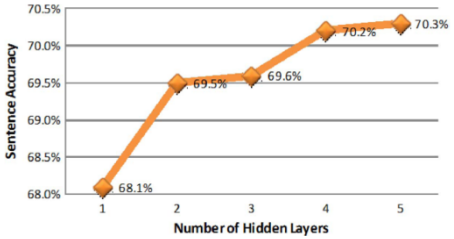
Deep Learning

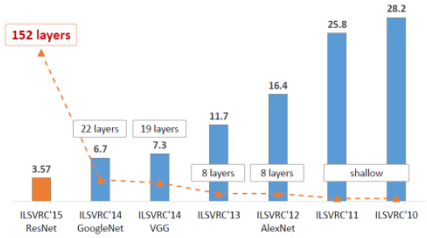
With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.



What is deep learning and why is it so popular?





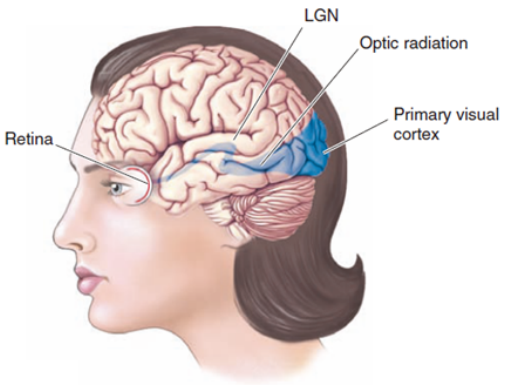


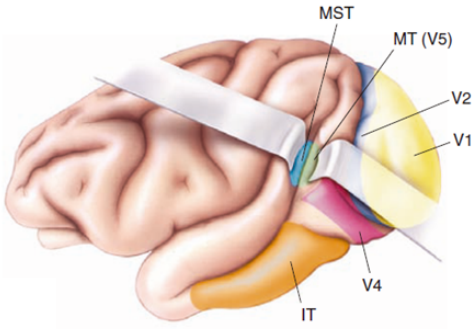
Why go deep?

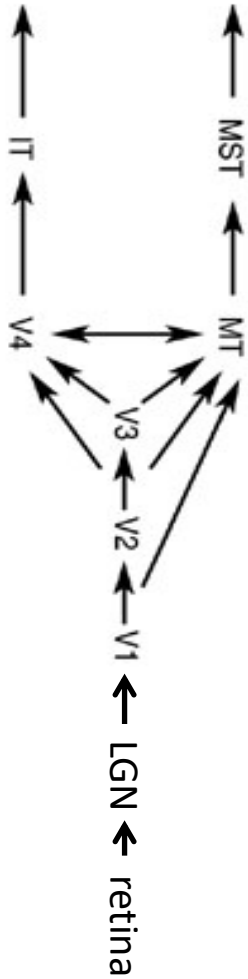
- Data are often high-dimensional.
- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.
- Insufficient depth can require more computational elements than architectures whose depth matches the task.
- Deep nets provide simpler but more descriptive model of many problems.

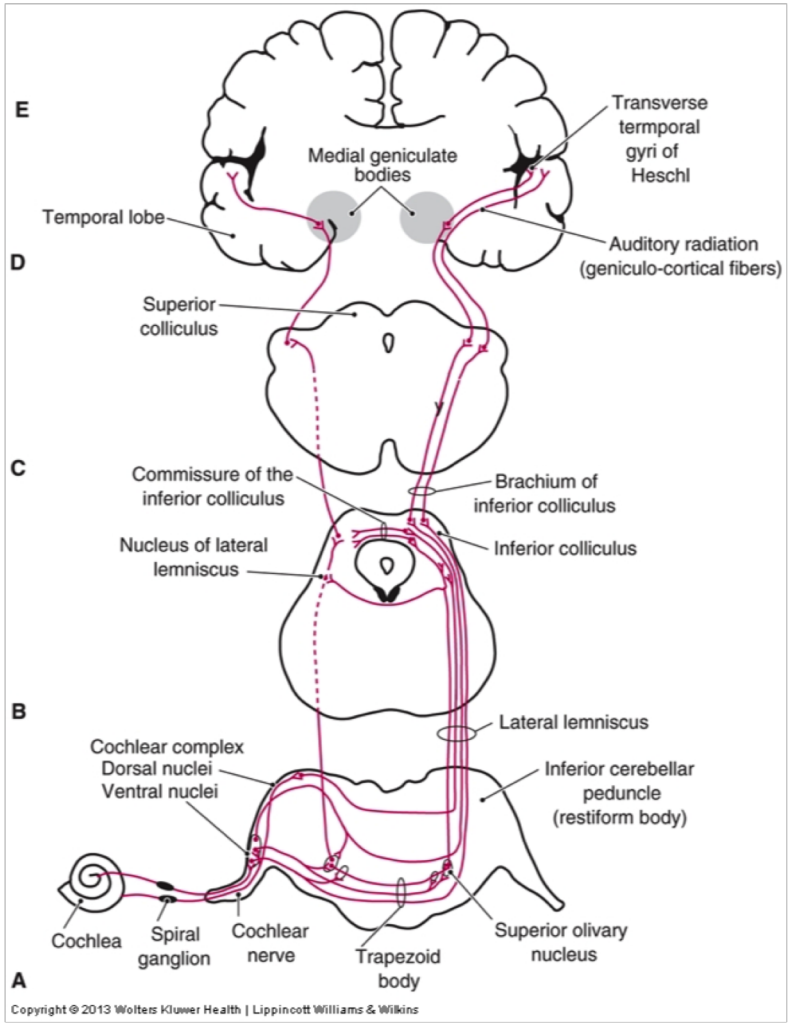
-By Geoffery Hinton

Hierarchical structures in the brain

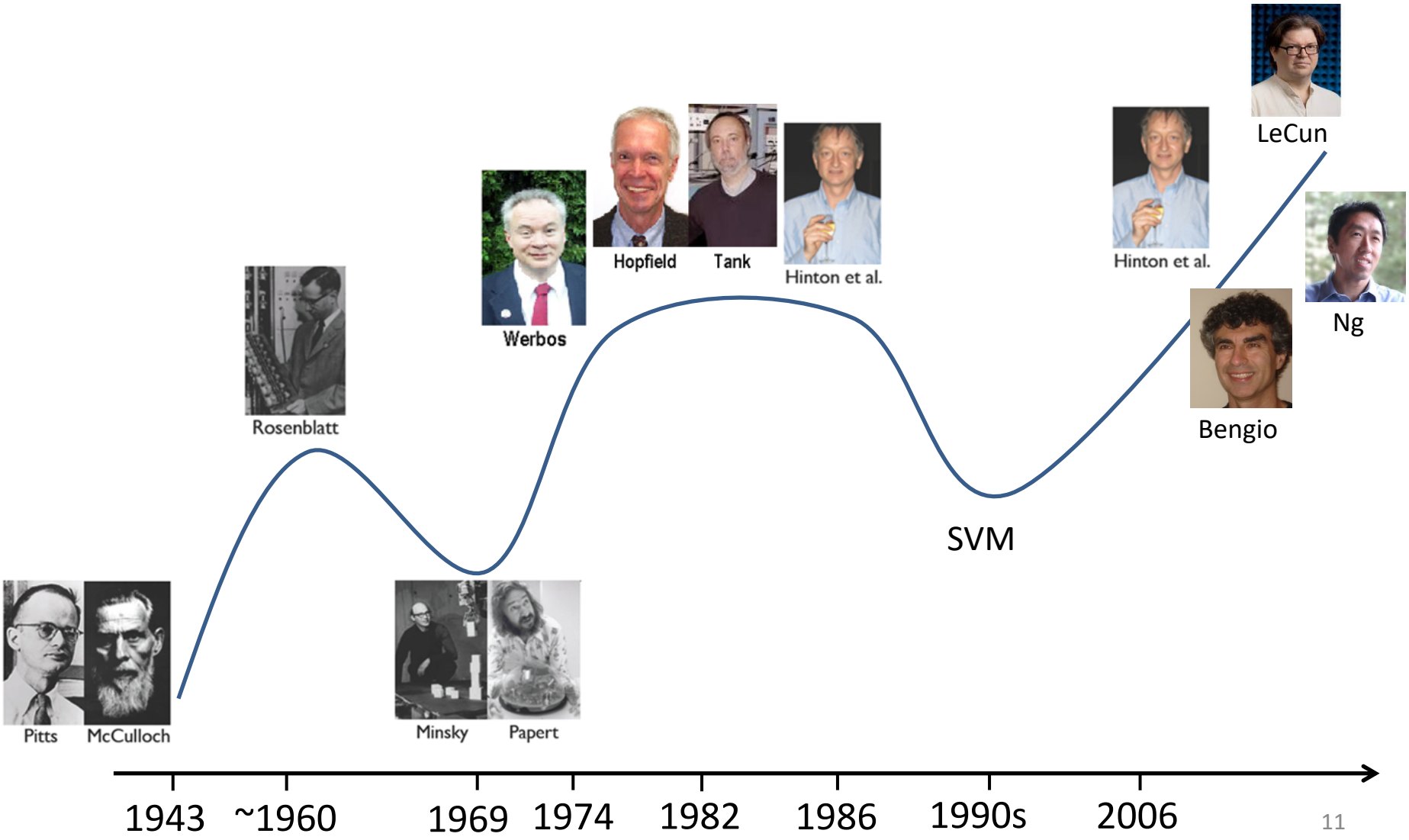




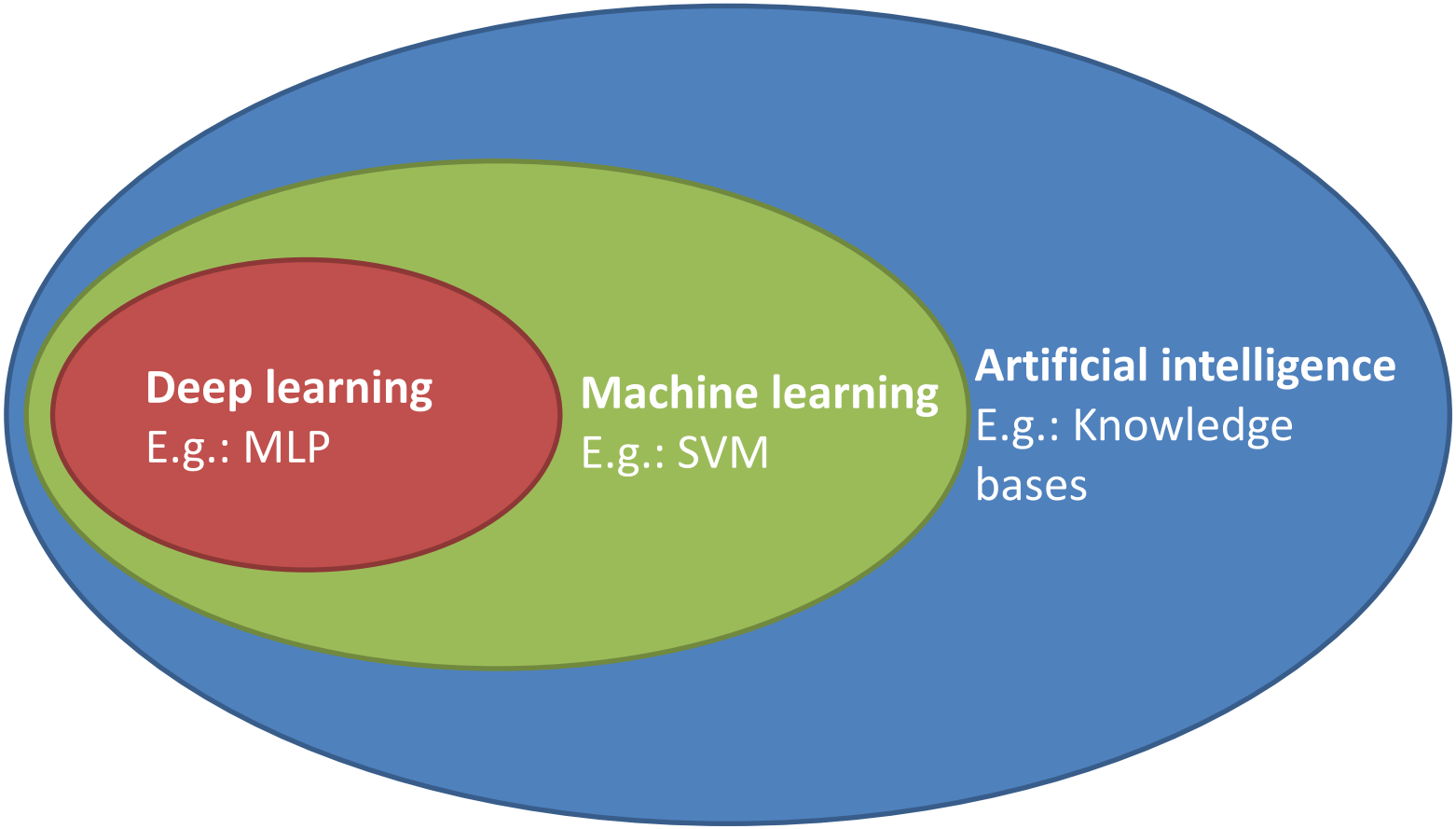




History of deep learning



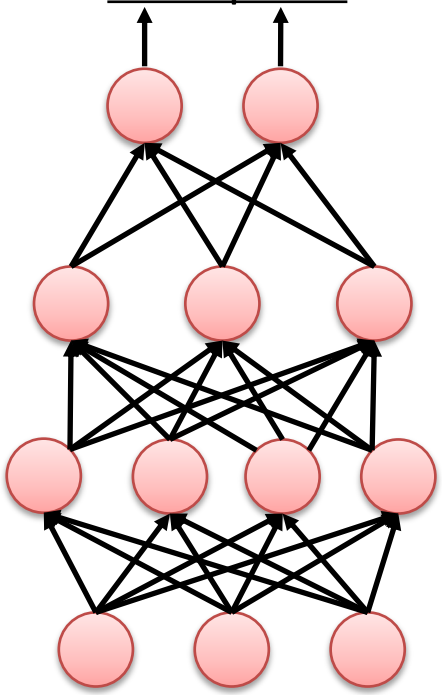
Deep learning, machine learning and AI



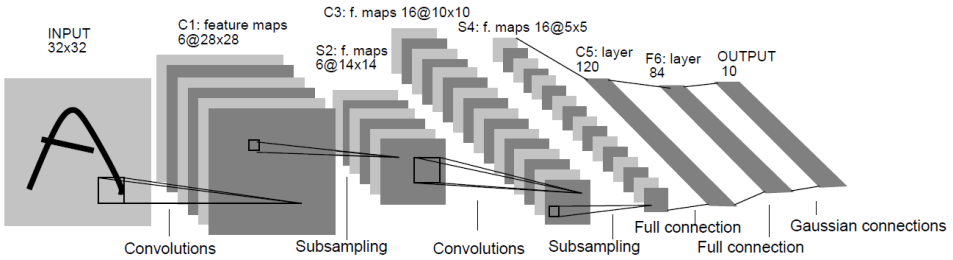
Contents to be covered in this summer school

- Various deep learning models

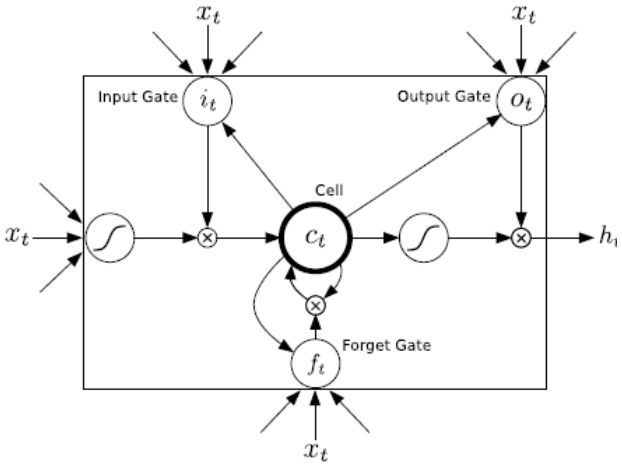
Multilayer Perceptron



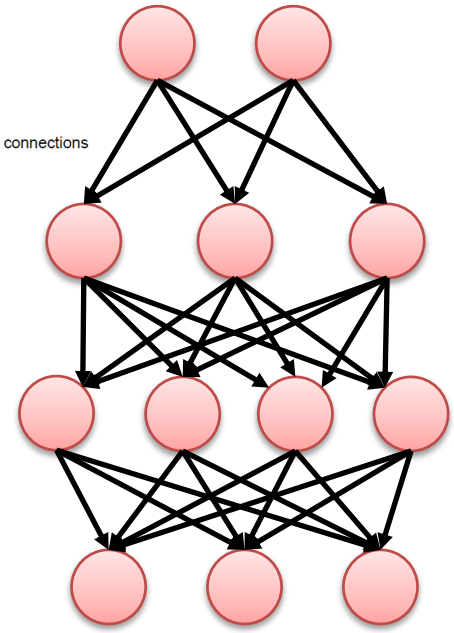
Convolutional neural network



Recurrent neural network



Deep belief network



Contents to be covered in this summer school

- Various applications
  - Image classification (Xiaolin Hu & Wenguang Chen)
  - Social multimedia analysis (Peng Cui)
  - Natural language processing (Minlie Huang)
  - Knowledge graph (Zhiyuan Liu)
  - Affective Computing (Jia Jia)
  - Robot sensing (Huaping Liu)
  - Deep Generative Model & Deep Reinforcement Learning (Jun Zhu)

Outline

- Overview of deep learning
- Math basics
- Machine learning basics
- A brief intro to Python

Math basics

LINEAR ALGEBRA

Math Concepts

- Scalar
  - A single number, often denoted by a lower case letter without boldface, e.g., a, b, x
- Vector
  - An array of numbers, often denoted by a lowercase letter with boldface, e.g., $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{x}$

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

- Matrix
  - A 2D array of numbers, often denoted by an uppercase letter with boldface, e.g., $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{X}$

$$\boldsymbol{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}$$

- Tensor
  - An n-D array of numbers, often denoted like this: e.g., $\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}$

$$\mathbf{A} = \left(\begin{pmatrix} A_{1,1,1} & A_{1,2,1} \\ A_{2,1,1} & A_{2,2,1} \end{pmatrix}, \begin{pmatrix} A_{1,1,2} & A_{1,2,2} \\ A_{2,1,2} & A_{2,2,2} \end{pmatrix} \right)$$

Simple operations

- Matrix transpose: $\boldsymbol{A}^\top$

$$\boldsymbol{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \qquad \boldsymbol{A}^\top = \begin{pmatrix} A_{1,1} & A_{2,1} \\ A_{1,2} & A_{2,2} \end{pmatrix}$$

- A vector can be viewed as a special matrix

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \qquad \boldsymbol{a}^\top = (a_1, a_2, a_3)$$

- Matrix product: if $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, then $\boldsymbol{C} = \boldsymbol{AB}$ with shape $m \times p$ and $C_{i,j} = \sum_k A_{i,k} B_{k,j}$
- Elementwise product (Hadamard product): $\boldsymbol{C} = \boldsymbol{A} \odot \boldsymbol{B}$ where the 3 matrices are of the same shape and $C_{i,j} = A_{i,j} B_{i,j}$

Simple operations

- Properties
  - $\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{A}\boldsymbol{B}+\boldsymbol{A}\boldsymbol{C}$
  - $\boldsymbol{A}(\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{A}\boldsymbol{B})\boldsymbol{C}$
  - $\boldsymbol{A}\boldsymbol{B} \neq \boldsymbol{B}\boldsymbol{A}$ but $\boldsymbol{x}^\top\boldsymbol{y} = \boldsymbol{y}^\top\boldsymbol{x}$
  - $(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top\boldsymbol{A}^\top$
- Identity matrix: $\boldsymbol{I} \in \mathbb{R}^{n\times n}$
  - For $x \in \mathbb{R}^n$, we have $\boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$
  - For $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, we have $\boldsymbol{I}\boldsymbol{A} = \boldsymbol{A}$

$$\boldsymbol{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

- Matrix inverse of a square matrix $\boldsymbol{A}$ is a matrix denoted by $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$

Linear equations

- A system of linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^{m}$ and $\boldsymbol{x} \in \mathbb{R}^{n}$, or equivalently

$$A_{1,1}x_1 + A_{1,2}x_2 + \cdots + A_{1,n}x_n = b_1$$
$$A_{2,1}x_1 + A_{2,2}x_2 + \cdots + A_{2,n}x_n = b_2$$
$$\cdots$$
$$A_{m,1}x_1 + A_{m,2}x_2 + \cdots + A_{m,n}x_n = b_m$$

  - If $m = n$ and $\boldsymbol{A}^{-1}$ exits, then there is exactly one solution $\boldsymbol{x}^*$ to the above equations
  - If $m < n$, there are infinite number of solutions — underdetermined
  - If $m > n$, the solution may not exists! — overdetermined

Norms

- Norm is used to measure the "size" of a vector or matrix
- The L_p norm of a vector is defined as

$$\left|\left|\boldsymbol{x}\right|\right|_p = \left(\sum_i |x_i|^p\right)^{1/p}$$

for $p \in \mathbb{R}, p > 1$

  - The L_2 norm is known as the Euclidean norm, and $\left|\left|\boldsymbol{x}\right|\right|_2 = \sqrt{\boldsymbol{x}^\top \boldsymbol{x}}$
  - The L_1 norm is simplified to $\left|\left|\boldsymbol{x}\right|\right|_1 = \sum_i |x_i|$
  - The L_∞ norm is simplified to $\left|\left|\boldsymbol{x}\right|\right|_\infty = \max_i |x_i|$

Norms

- A norm is any function f that satisfies
  - $f(\boldsymbol{x}) \geq 0$
  - $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0$
  - $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(x) + f(y)$ (the triangle inequality)
  - $f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x}), \forall \alpha \in \mathbb{R}$
- Frobenius norm of a matrix

$$||\boldsymbol{A}||_F = \sqrt{\sum_{i,j} A_{i,j}^2}$$

Math basics

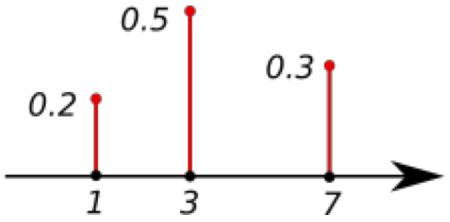
PROBABILITY THEORY

Random variable

- A random variable is a variable that can take different values randomly
  - Denote the random variable by x and its two possible values by x_1 and x_2
  - For vectors, we write the random variable as $\mathbf{x}$ and one of its values as $\boldsymbol{x}$
  - Discrete versus continuous

Probability distribution

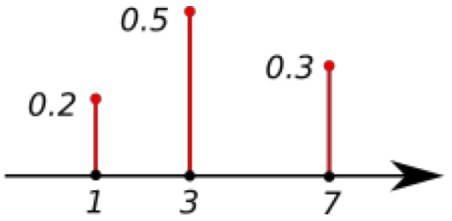
A probability distribution is a distribution of how likely a random variable or a set of random variables is to take on each of its possible states

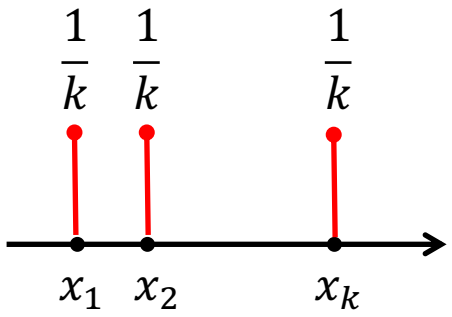


- A probability distribution over discrete variables may be described using a probability mass function (PMF)
  - The prob that $\mathrm{x} = x$ is denoted as $P(x)$ or $P(\mathrm{x} = x)$
  - $\mathrm{x} \sim P(\mathrm{x})$ specify which distribution x follows
- Joint probability
  - $P(\mathrm{x} = x, \mathrm{y} = y)$ or $P(x, y)$ denotes the prob that $\mathrm{x} = x$ and $\mathrm{y} = y$ simultaneously

Probability mass function

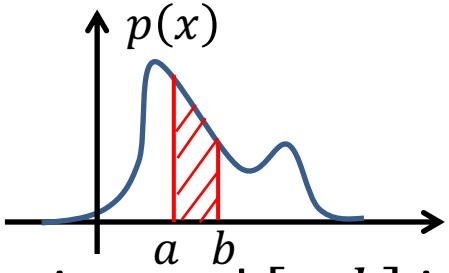
- To be a PMF of a random variable x, a function P must satisfy:
  - The domain of P must be the set of all possible states of x
  - $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$
  - $\sum_{x \in \mathrm{x}} P(x) = 1$
- Uniform distribution
  - Consider a single discrete random variable x with k different states
  - $P(\mathrm{x} = x_i) = \frac{1}{k}, \forall\, i$





Probability density function

- A probability distribution over continuous variables may be described using a probability density function (PDF)
- To be a PDF, a function p must satisfy the following properties
  - The domain of p must be the set of all possible states of x
  - $\forall x \in \mathrm{x}, p(x) \geq 0$
  - $\int p(x)dx = 1$



Can $p(x) > 1$?

- The prob that x lies in the interval $[a, b]$ is given by $\int_a^b p(x)dx$
- Note $p(x)$ does not give the prob of a specific state directly

Marginal probability

Suppose we know the prob distribution over a set of variables. The prob distribution over just a subset of them is known as the marginal prob distribution

- Let $P(\mathrm{x}, \mathrm{y})$ denote the prob distribution of discrete random variables x and y, then

$$P(\mathrm{x} = \mathrm{a}) = \sum_{b} P(\mathrm{x} = a, \mathrm{y} = b)$$

- Let $p(\mathrm{x}, \mathrm{y})$ denote the PDF of continuous random variables x and y, then

$$p(x) = \int P(x, y) dy$$

Bayes' rule

Thomas Bayes (1702~1761)

Prior probability of x

Posterior probability of x

Probability of y given x; likelihood

$$P(\mathrm{x}|\mathrm{y}) = \frac{P(\mathrm{x})P(\mathrm{y}|\mathrm{x})}{P(\mathrm{y})}$$

Prior probability of y

Conditional probability

The conditional probability is the probability of some event, given that some other event has happened

- The conditional prob that $\mathrm{y} = b$ given $\mathrm{x} = a$ is denoted by $P(\mathrm{y} = b|\mathrm{x} = a)$, which can be calculated as
$$P(\mathrm{y} = b|\mathrm{x} = a) = P(\mathrm{y} = b, \mathrm{x} = a)/P(\mathrm{x} = a)$$
- The chain rule
$$P\left(\mathrm{x}^{(1)}, \dots, \mathrm{x}^{(n)}\right) = P\left(\mathrm{x}^{(1)}\right)\Pi_{i=2}^{n} P\left(\mathrm{x}^{(i)}\middle|\mathrm{x}^{(1)}, \dots, \mathrm{x}^{(i-1)}\right)$$
- Exercise: With this definition, is the following correct?
$$P(\mathrm{a}, \mathrm{b}, \mathrm{c}) = P(\mathrm{a}|\mathrm{b}, \mathrm{c})P(\mathrm{b}|\mathrm{c})P(\mathrm{c})$$

Independence and conditional independence

Two random variables x and y are independent if their joint prob distribution can be expressed as a product of two factors, one involving x and one involving y :

Denoted by $\mathrm{x} \perp \mathrm{y}$

$$\forall x \in \mathrm{x}, y \in \mathrm{y},$$
$$p(\mathrm{x} = a, \mathrm{y} = b) = p(\mathrm{x} = a)p(\mathrm{y} = b)$$

Two random variables x and y are conditionally independent given a random variable z if the conditional prob distribution over x and y factorizes in this way:

Denoted by $\mathrm{x} \perp \mathrm{y}|\mathrm{z}$

$$\forall a \in \mathrm{x}, b \in \mathrm{y}, c \in \mathrm{z},$$
$$p(\mathrm{x} = a, \mathrm{y} = b|\mathrm{z} = c) = p(\mathrm{x} = a|\mathrm{z} = c)p(\mathrm{y} = b|\mathrm{z} = c)$$

Expectation

The expectation, or expected value, of some function $f(x)$ w.r.t. a prob distribution $P(\mathrm{x})$ is the average value that f takes on when x is drawn from P

- For discrete variables

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \sum_x P(x)f(x)$$

- For continuous variables

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \int P(x)f(x)dx$$

- If the identity of the distribution is clear, we may write $\mathbb{E}_{\mathrm{x}}[f(x)]$
- Expectation is linear: if α and β do not depend on x, then

$$\mathbb{E}_{\mathrm{x}}[\alpha f(x) + \beta g(x)] = \alpha\mathbb{E}_{\mathrm{x}}[f(x)] + \beta\mathbb{E}_{\mathrm{x}}[g(x)]$$

Variance and covariance

- The variance of a function $f(x)$

$$\mathrm{Var}\big(f(x)\big) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$$

  - It measures how much the value of the function f varies when x is sampled from its distribution

- The covariance of two functions $f(x)$ and $g(x)$

$$\mathrm{Cov}(f(x), g(y)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])]$$

  - It measures how much two values are linearly related to each other, as well as the scale of these variables
  - High absolute values of the covariance mean that the values change very much and are both far from their respective means

- The covariance matrix of a random vector $\boldsymbol{x} \in \mathbb{R}^n$ is an $n \times n$ matrix

$$\mathrm{Cov}(\mathbf{x})_{i,j} = \mathrm{Cov}(\mathrm{x}_i, \mathrm{x}_j)$$

Common prob distributions

- Bernoulli distribution: over a single binary random variable

$$P(\mathrm{x} = 1) = \phi, P(\mathrm{x} = 0) = 1 - \phi$$
$$P(\mathrm{x} = x) = \phi^{x}(1 - \phi)^{1-x}$$
$$\mathbb{E}_{\mathrm{x}}[\mathrm{x}] = \phi, \mathrm{Var}(\mathrm{x}) = \phi(1 - \phi)$$

- Multinoulli or categorical distribution: over a single discrete variable with k diff states where k is finite

$$P(\mathrm{x} = i|\boldsymbol{p}) = p_i$$

  - where $\boldsymbol{p} \in [0,1]^k$ and $\sum_{i=1}^{k} p_i = 1$

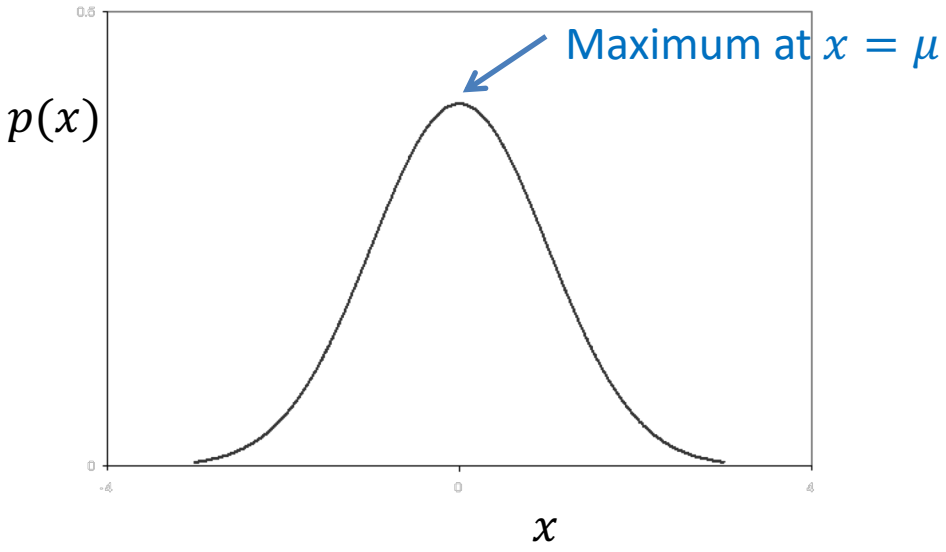
Common prob distributions

- Gaussian distribution or normal distribution: over a continuous variable

$$\mathcal{N}(x;\mu,\sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  - Mean: $\mathbb{E}[\mathrm{x}] = \mu$
  - Variance: $\mathrm{Var}[\mathrm{x}] = \sigma^2$
  - Standard deviation: σ

The *central limit theorem* shows that the sum of many independent variables is approximately normally distributed



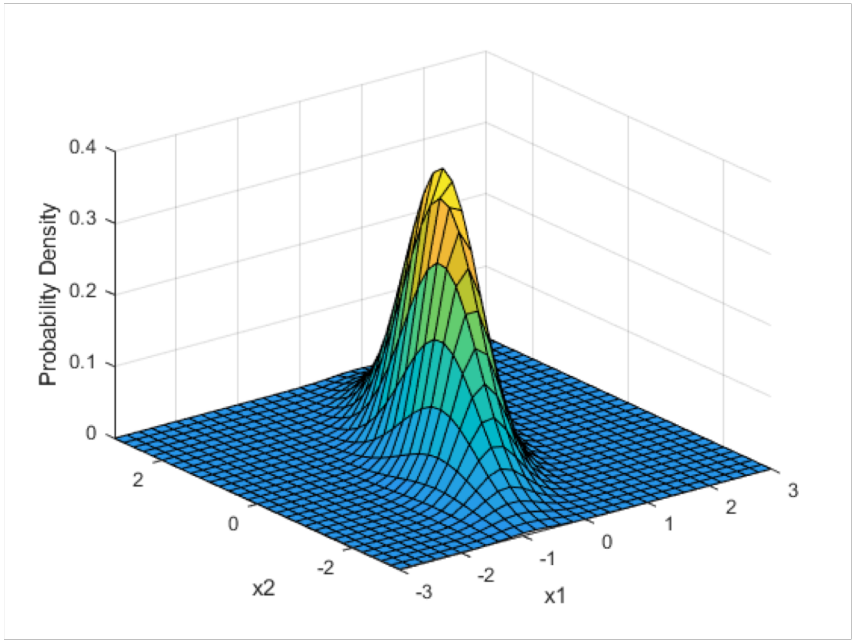
Common prob distributions

- Multivariate normal distribution: over a continuous vector

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\sigma^2}) = \sqrt{\frac{1}{(2\pi)^2 \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma^{-1}} (\boldsymbol{x}-\boldsymbol{\mu})\right)$$

  - Mean: $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$
  - Covariance: $\mathrm{Cov}[\mathbf{x}] = \boldsymbol{\Sigma}$

Isotropic Gaussian distribution: $\boldsymbol{\Sigma} = \alpha \boldsymbol{I}$ where α is a scalar



Math basics

OPTIMIZATION

Gradient-based optimization

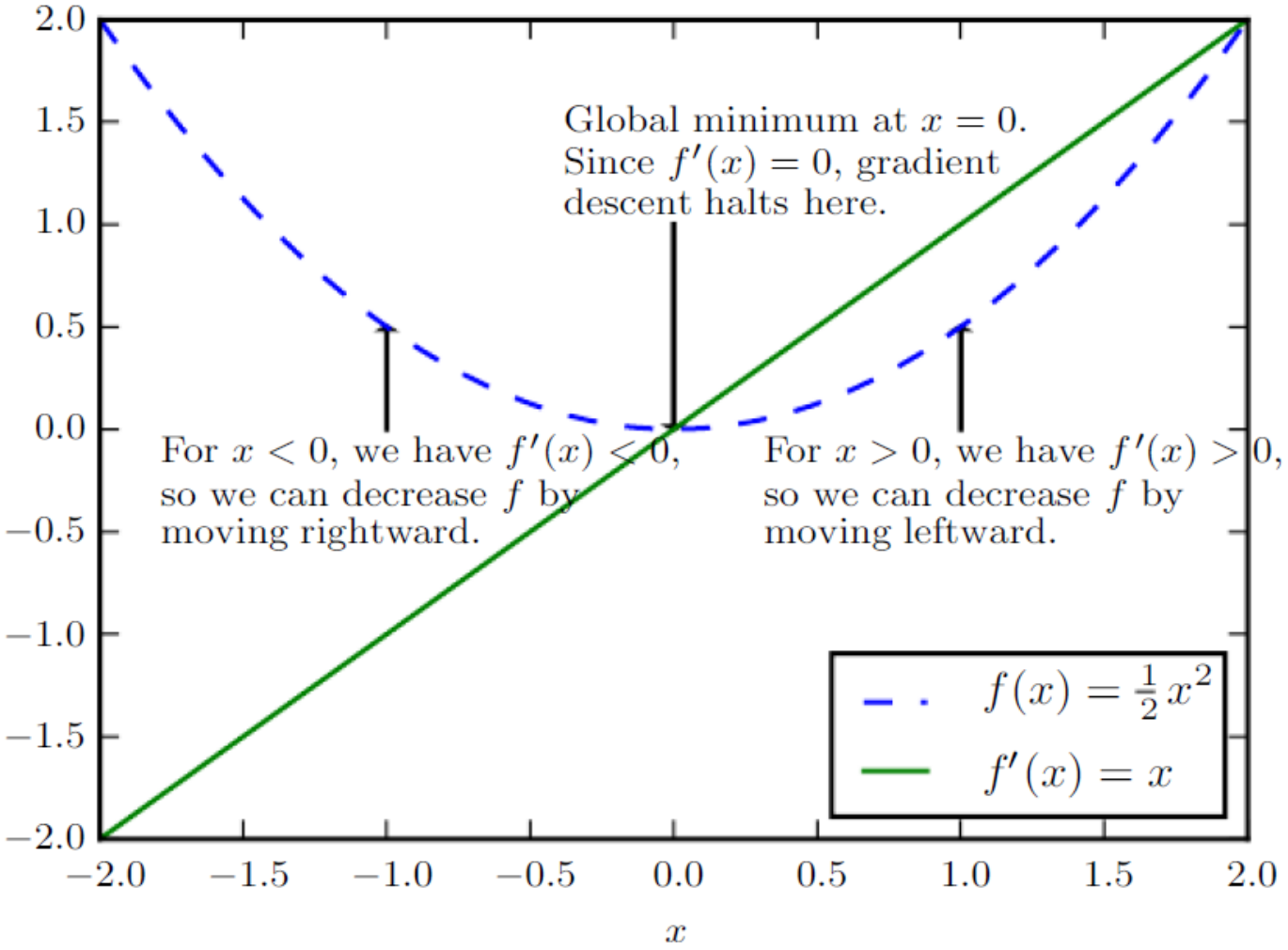
- The function we want to minimize or maximize is called objective function
- When we are minimizing it, we may also call it the cost function, loss function, or error function
- The derivative of a function $y = f(x)$, denoted by $f'(x)$ or $\frac{dy}{dx}$, gives the slope, or gradient, of f at the point x
- Gradient descent

$$x' = x - \eta f'(x)$$

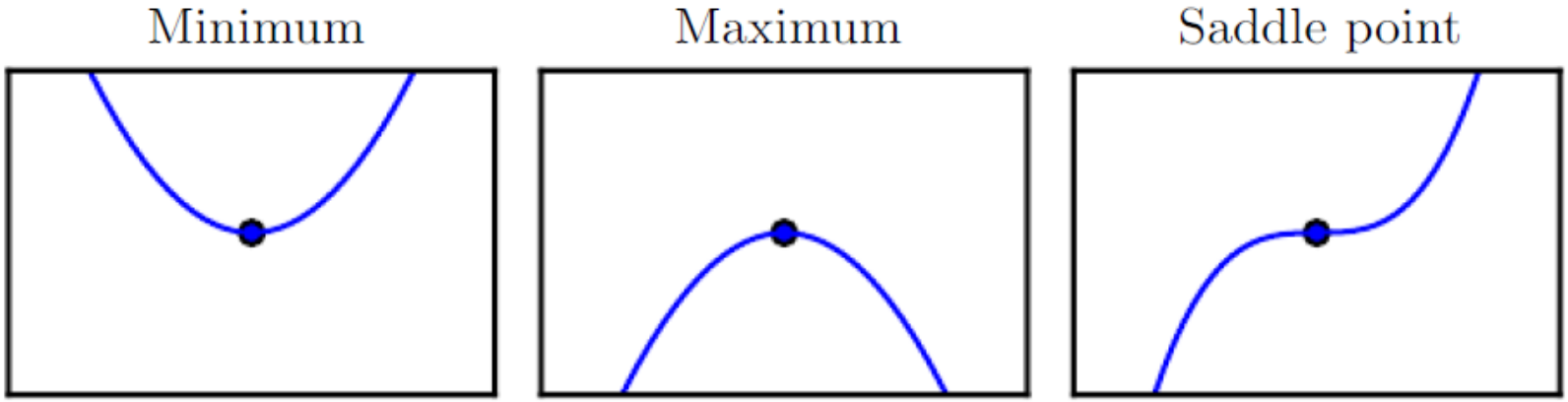
where $\eta > 0$ is the learning rate

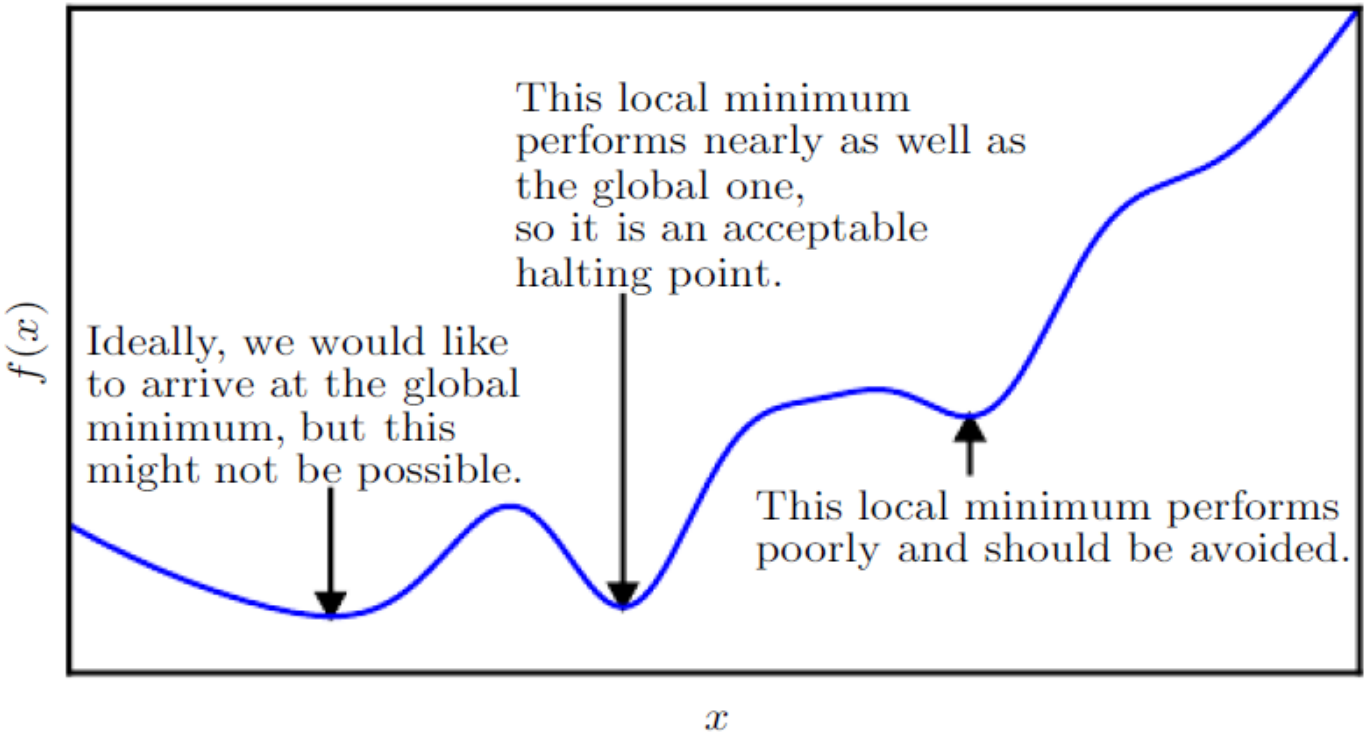
Gradient descent

$$x' = x - \eta f'(x)$$



Critical points





Gradient decent for multivariate functions

- For a function of a single variable $y = f(x)$, the gradient decent method is

$$x' = x - \eta f'(x)$$

- For a function $y = f(\boldsymbol{x})$, the partial derivative is denoted by $\partial f / \partial x_i$
- The gradient decent method becomes

$$\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

where $\eta > 0$ and $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \dots \\ \partial f / \partial x_n \end{pmatrix}$

Summary so far

- Linear algebra
  - Math objects: Scalars, vectors, matrices, tensors
  - Simple operations: matrix transpose, inverse, product
  - Norms: L_p norm
- Probability theory
  - Random variables: discrete, continuous
  - Prob distribution: PMF and PDF
  - Marginal probability
  - Conditional probability
  - Independence and conditional independence
  - Expectation, variance and covariance
  - Common prob distributions
  - Bayes' rule
- Optimization
  - Gradient descent
  - Critical points

Outline

- Overview of deep learning
- Math basics
- Machine learning basics

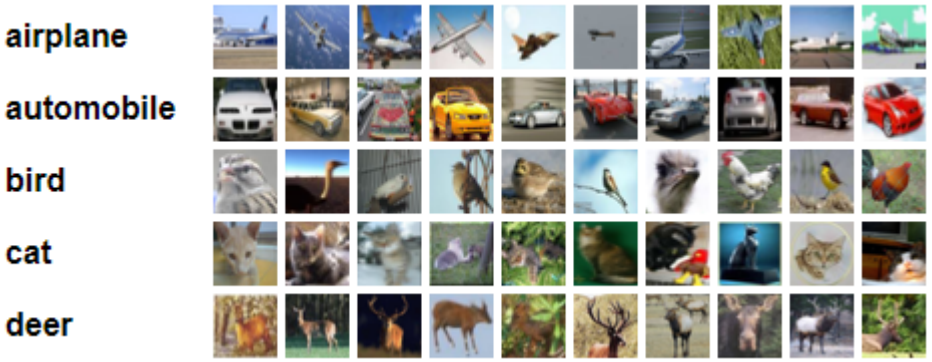
Learning algorithms

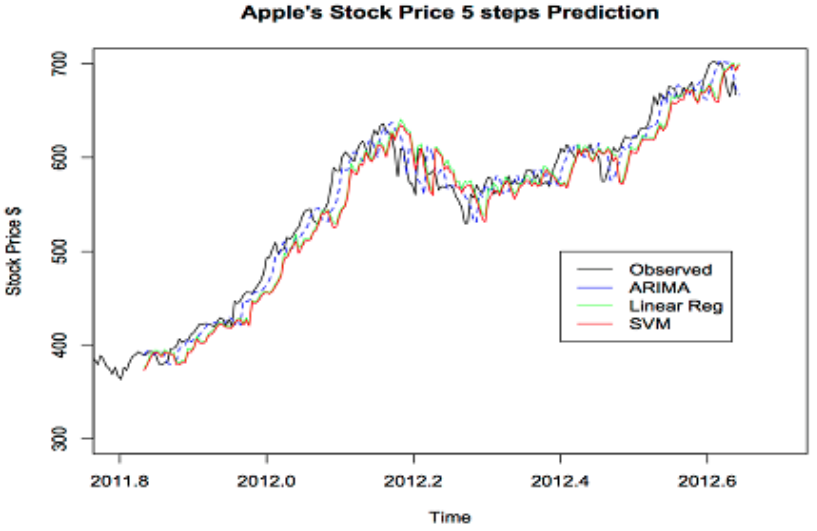
"A computer program is said to learn from experience E w.r.t. some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." ---Tom Mitchell, 1997

- Machine learning (ML) tasks are usually described in terms of how the ML system should process an example
- An example is a collection of features that have been quantitatively measured from some object or event
  - Features of a bucket: color, diameter, height, material, etc
  - Features of an animal: size, shape, number of legs, , etc

The tasks T

- Classification
  - Suppose there are k categories. Find a function $f: \mathbb{R}^n \to \{1, \ldots, k\}$





- Regression
  - Find a function $f: \mathbb{R}^n \to \mathbb{R}^m$, and m is often 1

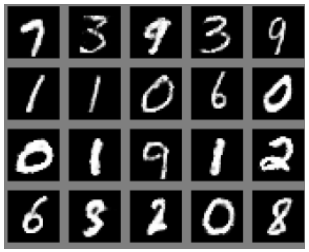
Regression results might be converted to classification results, if the regression curves have special forms

The tasks T

- Synthesis and sampling

dataset

Synthesized using GAN



- Denoising

The tasks T

- Transcription

Input

Output

Deep learning is cool

- Machine translation

The tasks, T

- Structured output
- Anomaly detection
- Synthesis and sampling
- Imputation of missing values
- Density estimation
- Etc.

The performance measure, P

- A performance measure is required to quantitatively evaluate the performance of a ML system
- Usually this measure P is specific to the task T being carried out by the system
  - Classification and transcription: accuracy or error rate
  - Regression and denoising: distance between the ground-truth and prediction
  - Synthesis, machine translation: difficult and sometimes need human evaluation
- What we are more interested in is the performance measure on a test set of data that is separated from the data used for training the system

The experience, E

- ML algorithms including deep learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process
- The algorithms experience a dataset, which is a collection of many examples or data points denoted by $\boldsymbol{x}$
  - We can view examples as samples of a random variable $\mathbf{x}$
- Unsupervised learning algorithms
  - experience a dataset containing many features, then learn useful properties or structures of this dataset
- Supervised learning algorithms
  - experience a dataset containing many features, but each example is also associated with a label or target denoted by $\boldsymbol{y}$

learn $p(\mathbf{x})$

learn $p(\mathbf{y}|\mathbf{x})$

Example: linear regression

x_i: feature
w_i: weight

- Task T: to predict y from x by outputting $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$
- Performance P: mean squared error of the model on the test with m test samples $\{(\boldsymbol{x}_i, y_i)\}^{\text{test}}$

$$\text{MSE}_{\text{test}} = \frac{1}{m}\sum_i (\hat{y}_i - y)^{\text{test}}$$

- Experience E: minimize the MSE on the training set of q samples $\{(\boldsymbol{x}_i, y_i)\}^{\text{train}}$

$$\text{MSE}_{\text{train}} = \frac{1}{q}\sum_i (\hat{y}_i - y)^{\text{train}}$$

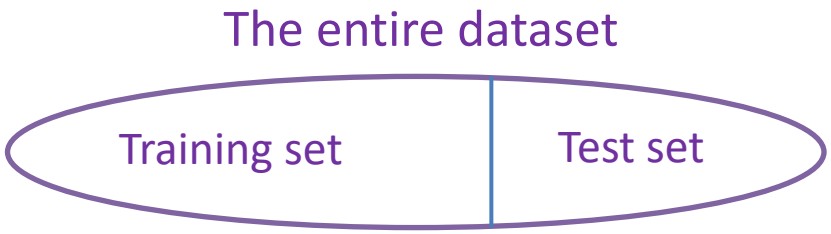
  - Denote $\{(\boldsymbol{x}_i, y_i)\}^{\text{train}}$ collectively by $\left(\boldsymbol{X}^{\text{train}}, \boldsymbol{y}^{\text{train}}\right)$, then

$$\nabla_w \text{MSE}_{\text{train}} = \nabla_w \frac{1}{q}\left|\left|\hat{\boldsymbol{y}}^{\text{train}} - \boldsymbol{y}^{\text{train}}\right|\right|_2^2 = 0$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{\text{train}\top}\boldsymbol{X}^{\text{train}}\right)^{-1}\boldsymbol{X}^{\text{train}\top}\boldsymbol{y}^{\text{train}}$$

Capacity, overfitting and underfitting

- A ML algorithm must perform well on new, previously unseen inputs—not just on which it was trained
  - This ability is called generalization



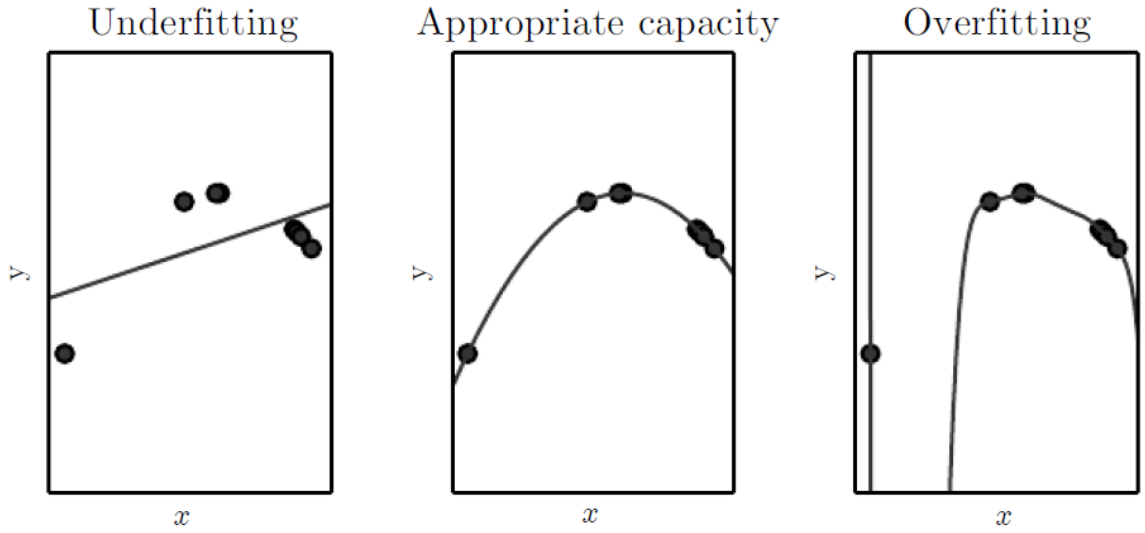
- Smaller training error → higher model capacity
  - If the training error is too large, the model is underfitting the training set
- Smaller test error or generalization error → higher generalization ability
  - If the training error is very small but the test error is very large, the model is overfitting the training set

Example: polynomial regression

- Consider a regression problem in which the input x and output y are both scalars. Find a function $f: \mathbb{R} \to \mathbb{R}$ to fit the data
  - $f(x) = b + wx$
  - $f(x) = b + w_1 x + w_2 x^2$
  - $f(x) = b + \sum_{i=1}^{9} w_i x^i$

MSE training:

$$\min_w \frac{1}{N} \sum_{n=1}^{N} \left|\left| f\left(x^{(n)}\right) - y^{(n)} \right|\right|_2^2$$



General principles

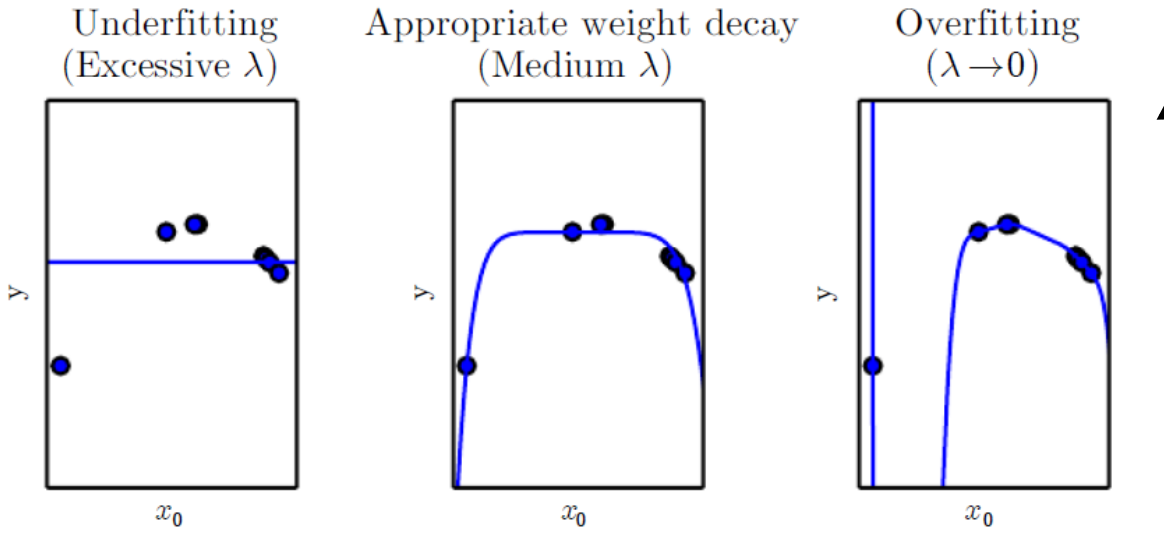
- Increase the model capacity
  - Make the training error small
- Increase the generalization ability
  - Make the gap between training error and test error small

Regularization

- To carry out a specific task, we often build a set of preferences into the learning algorithm, which is embodied by a regularizer Ω
- E.g., for polynomial regression, the total cost function becomes

$$J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^{\top}\boldsymbol{w}$$

where $\lambda > 0$ is a constant.



A high-degree polynomial regression example

- Here $\Omega(\boldsymbol{w}) = \boldsymbol{w}^{\top}\boldsymbol{w}$
- There are many regularizers

Regularization

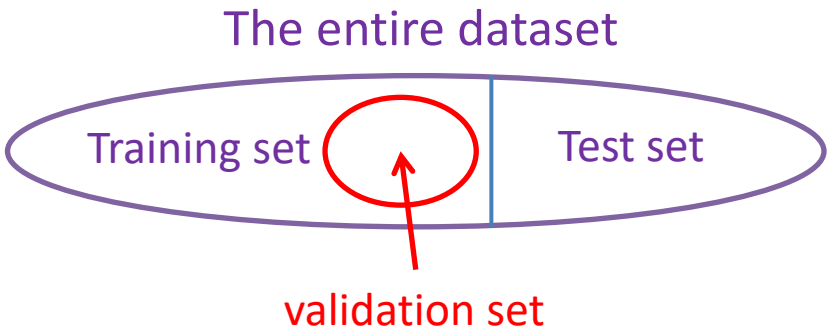
Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

Hyperparameters

- Many machine learning algorithms have two sets of parameters:
  - Hyperparameters: control the algorithm's behavior and are not adapted by the algorithm itself. They often determines the capacity of the model
  - Learnable parameters ("learnable" is often omitted): can be learned from data
- The polynomial regression algorithm $J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^{\top} \boldsymbol{w}$
  - Hyperparameters: λ
  - Learnable parameters: $\boldsymbol{w}$
- A neural network
  - Hyperparameters: the number of layers, the number of neurons per layer, etc.
  - Learnable parameters: weights and bias in each layer

Validation sets

- How to choose the hyperparameters considering that we cannot see the test set?
  - Set them such that the training error is as small as possible?
- We need another set on which the model is not trained on
  - Make the error on this set as small as possible
  - This is called the validation set
- How do we obtain a validation set?



Maximum likelihood estimation

Problem definition

- Given a set of m examples $\mathbb{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ drawn independently from the true but unknown data-generating distribution $p_{\text{data}}(\mathbf{x})$
- Find a prob distribution $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ to approximate $p_{\text{data}}(\mathbf{x})$
- The task is to find optimal $\boldsymbol{\theta}$

- The maximum likelihood estimator for $\boldsymbol{\theta}$ is defined as

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\text{model}}(\mathbb{X}; \boldsymbol{\theta}) = \arg\max_{\theta} \Pi_{i=1}^{m} p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

- We usually use

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$$

Log-likelihood

  - where $\hat{p}_{\text{data}}$ is the empirical distribution

Conditional log-likelihood

- Estimate a conditional probability $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ in order to predict $\mathbf{y}$ given $\mathbf{x}$
  - E.g. For classification, $\mathbf{y}$ is a (discrete) random variable representing label of an input $\mathbf{x}$
- If $\boldsymbol{X}$ represents all inputs and $\boldsymbol{Y}$ all observed targets, then the conditional maximum likelihood estimator is

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} P_{\mathrm{model}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\theta})$$

- If the examples are assumed to be i.i.d., then this can be decomposed into

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P_{\mathrm{model}}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

Stochastic gradient decent

- Minimizing the cost function over the entire training set is computationally expensive
- We often decompose the training set into smaller subsets or minibatches and optimize the cost function defined over individual minibatches $(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)})$ and take the average

$$J(\boldsymbol{\theta}) = \frac{1}{m'} \sum_{i=1}^{m'} L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta})$$

$$\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta})$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \boldsymbol{g}$$

- A total of m' minibatches
- The batchsize ranges from 1 to a few hundreds

Summary

- Overview of deep learning
- Math basics
  - Linear algebra
  - Probability theory
  - Optimization
- Machine learning basics
  - Learning algorithms: Task, performance measure, experience
  - Capacity, Overfitting and Underfitting
  - Hyperparameters and Validation Sets
  - Maximum Likelihood Estimation
  - Stochastic gradient decent