*Introduction of*
*Machine Learning Security*

Lecture 01

# Overview of Machine Learning and Its Security

**Patrick Chan**
patrickchan@ieee.org

## Agenda

- Machine Learning Age
- Examples of Machine Learning Security
- Type of Attacks
- Adversary's Characteristics
- Refresher on Machine Learning



## Machine Learning Age

- It will be great if a machine can learn by itself



## Machine Learning Age

- Before, AI & ML mainly can be found in fictions or Hollywood Movies

Artificial Intelligence

Avengers: Age of Ultrons

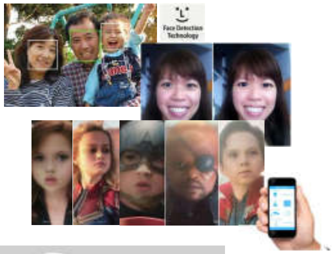
The Terminator

Alita: Battle Angel

iRobot

## Machine Learning Age

- AI & ML is everywhere now!






## AI Impact

AlphaGo
(2017)


DeepMind

AlphaGo Zero
without using data from human games,
and stronger than any previous version

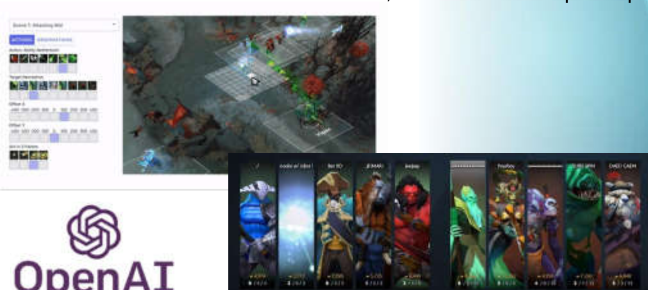
https://en.wikipedia.org/wiki/AlphaGo
https://deepmind.com/



## AI Impact

OpenAI Five (2018)
Dota 2 Bot

Defeat the professional team twice
99.4% win in 42,729 matches with public players



OpenAI

https://en.wikipedia.org/wiki/OpenAI_Five



## AI Impact

**IBM: Project Debater (2019)**

**"We should subsidize preschool."**
- Project Debater (Agree)
- Harish Natarajan (Disagree)

15 mins Preparation
4 mins Opening statement
4 mins Rebuttal
2 mins Summary

58%: Project Debater better enriched their knowledge about the topic compared to Harish's 20%

| Poll | Agree | Disagree | Undecided |
|---|---|---|---|
| Before | 79% | 13% | 8% |
| After | 62% (-17%) | 30% (+17%) | 8% |

https://www.research.ibm.com/artificial-intelligence/project-debater/live/

## OpenAI: ChatGPT (2022)

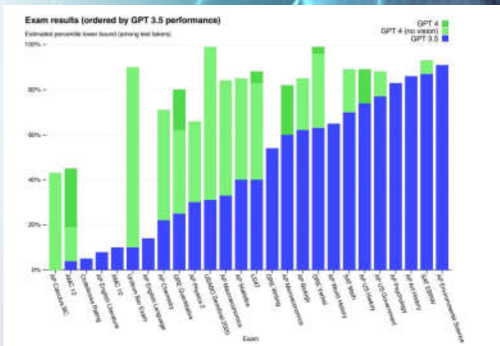
- Chat with images, voice and create images
- Understanding: Summary, extraction, expansion
- Translation
- Programming

Large Language Model

Replace the equivalent of 300 million full-time jobs

"ChatGPT is scary good, we are not far from dangerously strong AI.“ by Elon Musk

OpenAI




https://openai.com/chatgpt





## OpenAI: Sora (2024)

- Create realistic and imaginative scenes from text instructions

A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.

OpenAI

https://openai.com/sora





https://openai.com/sora





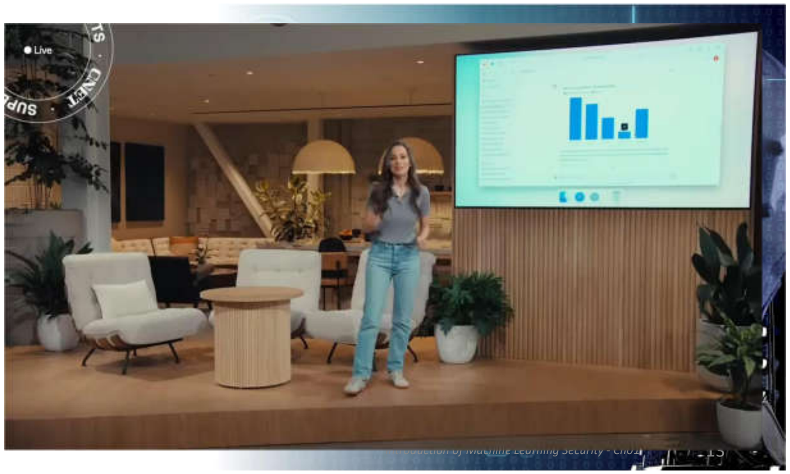
Historical footage of California during the gold rush

A close up view of a glass sphere that has a zen garden within it. There is a small dwarf in the sphere who is raking the zen garden and creating patterns in the sand.

https://openai.com/sora

# AI Impact

**OpenAI ChatGPT 4o**



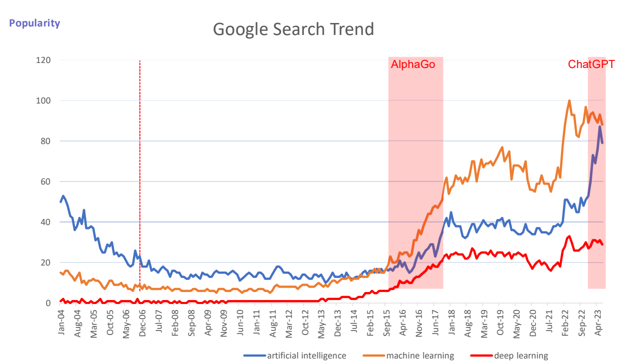
https://openai.com/sora

# AI Impact

https://openai.com/sora

# Machine Learning Age

- Due to the great success of Deep Learning, Machine Learning becomes more popular



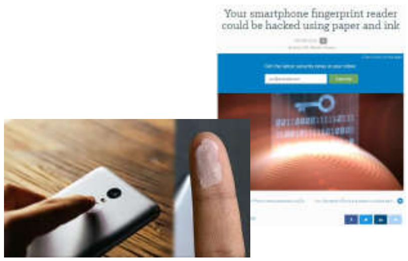



# Machine Learning Age

- Everything looks good?!?

# Machine Learning: Security

- **Person Identification in Mobile**
  - Fingerprint
  - Face (RGB, Depth, Inferred)



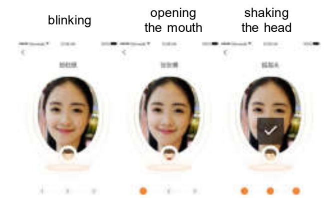






# Machine Learning: Security

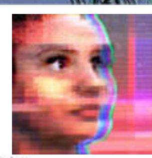
- Action videos showing movements such as shaking heads, blinking, and opening the mouth by using High-Definition Headshots (March 2021, Shanghai )
- Fool the liveness detection of person identification






# Machine Learning: Security

- **Tay** is a chatter bot released by Microsoft via Twitter in 2016
- Learn from interacting with human users of Twitter
- 16 hours after releasing, Tay was shut down due to her abusive and offensive messages






# Machine Learning: Security

- Hack Google Maps?

## Machine Learning: Security

- Can we mislead **Tesla**?



## Machine Learning: Security

- Can we mislead **Tesla**?





## Machine Learning: Security

- Can we mislead **Tesla**?

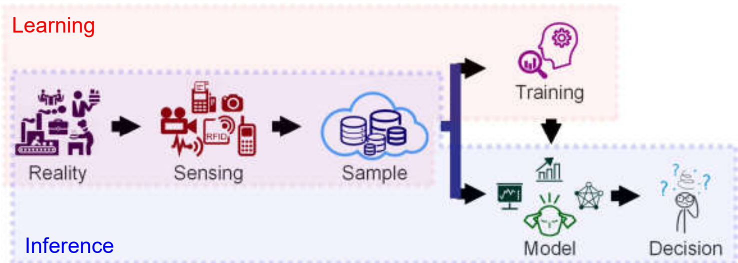
https://keenlab.tencent.com/zh/2019/03/29/Tencent-Keen-Security-Lab-Experimental-Security-Research-of-Tesla-Autopilot/



## Machine Learning: Security

- Security issues of Machine Learning techniques have not been investigated deeply before applying them to the real world
- A machine learning system can be fooled much easier than one might imagine

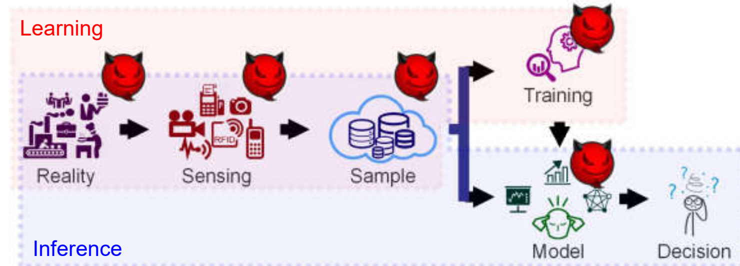
- **Machine Learning**
  Algorithm is improved automatically by using data
- Two phases: Learning + Inference








- An **adversary** may exist at anywhere to mislead a model
  - Especially in a security-related application

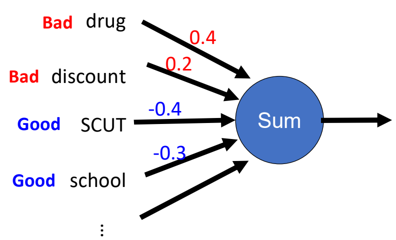


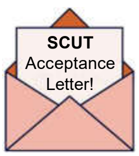





# Example

- **Junk Mail Filter**
  Classify if an email is a junk mail
  - Positive: Junk Mail (Spam)
  - Negative: Legitimate Mail (Ham)
- A linear Classifier with Boolean features indicating whether a word is present
  - **Bad Word**
    positive weight
  - **Good Word**
    negative weight




| | | |
|---|---|---|
| SCUT | -0.4 | Good Word |
| Acceptance | 0.1 | Bad Word |
| Letter | N/A | Natural Word |

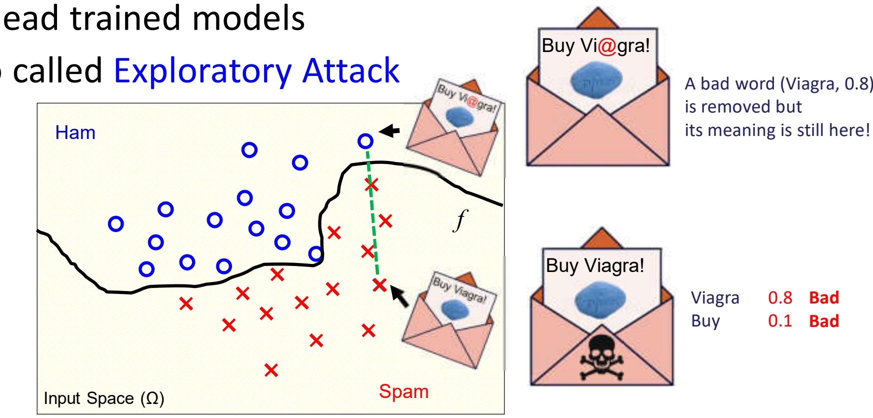
**Total = -0.3. It is Legitimate!**



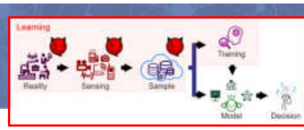


# Evasion Attack

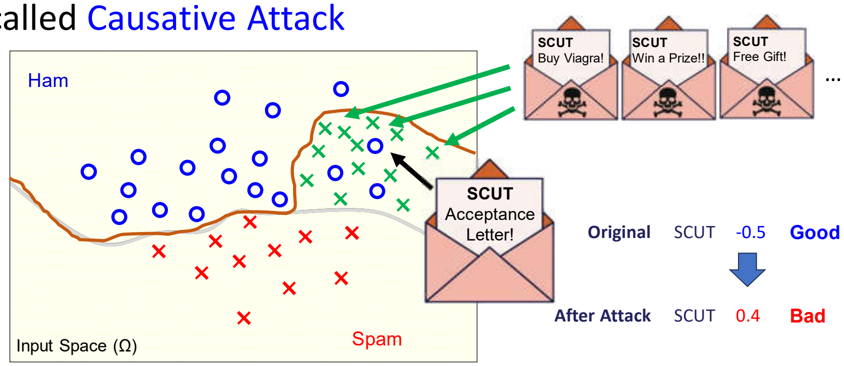
- Mislead trained models
- Also called Exploratory Attack



# Poisoning Attack




- Affect a training process
- Also called Causative Attack

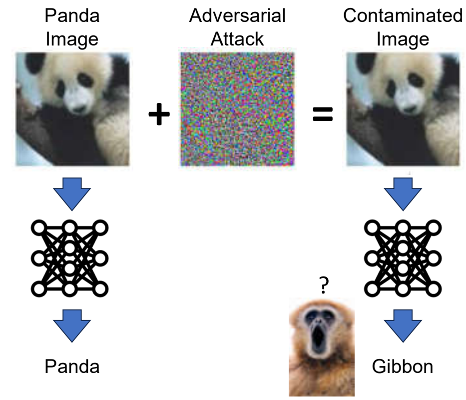







# Example: Classification

- Adversarial Attack is commonly discussed in classification problems
  - Personal Identification (Face, fingerprint…)
  - Object Identification (Sign…)







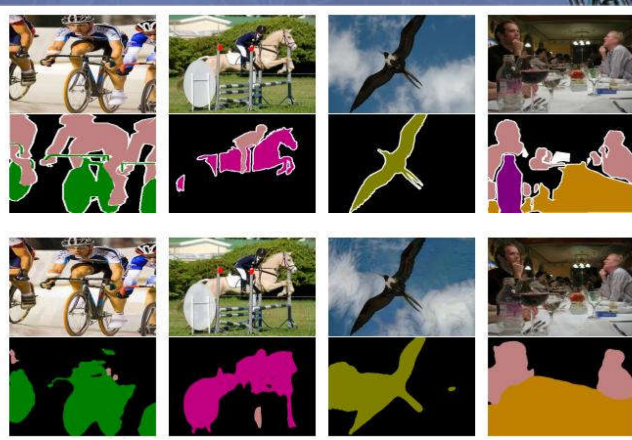

# Example: Segmentation

**Original Images**

Objects can be identified

**Contaminated Images**

Objects cannot be identified



*Gu, J., Zhao, H., Tresp, V., & Torr, P. H. (2022). Segpgd: An effective and efficient adversarial attack for evaluating and boosting segmentation robustness. In European Conference on Computer Vision.*





# Example: Recommender System

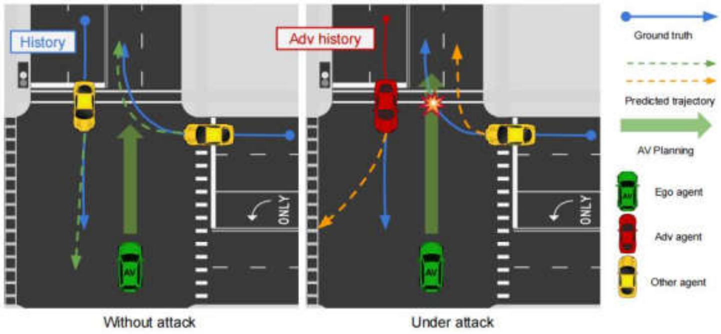
- Retailers want products rank at the top to increase the sales
  - Aim to manipulate rankings by injection fake user profiles
    - Push Attack: recommend more
    - Nuke Attack: recommend less

| | Item1 | Item2 | Item3 | Item4 | Item5 | Item k | |
|---|---|---|---|---|---|---|---|
| Alice | 5 | 3 | / | 4 | / | ? | |
| User1 | 3 | / | 2 | 3 | 5 | 2 | sim(Alice, User1) =1 |
| User2 | / | 3 | 4 | 3 | 3 | 1 | sim(Alice, User2) =0.87 |
| User3 | 3 | / | / | 2 | / | 4 | sim(Alice, User3) =0 |
| Fake1 | 5 | 3 | / | / | / | 5 | sim(Alice, Fake1) =0.96 |
| Fake2 | 5 | / | 2 | 4 | / | 5 | sim(Alice, Fake2) =0.92 |
| Fake3 | / | 3 | / | 4 | / | 5 | sim(Alice, Fake2) =0.99 |

*M Si, Q Li (2020)Shilling attacks against collaborative recommender systems: a review. In: Artificial Intelligence Review, 53(1), 291-319*

# Example: Autonomous Driving

- Mislead the predicted trajectories by slightly adjusting the history trajectory of one car




Cao, Y., Xiao, C., Anandkumar, A., Xu, D., & Pavone, M. (2022). Advdo: Realistic adversarial attacks for trajectory prediction. In European Conference on Computer Vision.





# Comparison

- Methods dealing with outliers and noise may not work in adversarial environment

| Outlier | Adversarial Attack |
| --- | --- |
| • Model Independent<br>• Very different from normal | • Design based on model<br>• May camouflage as normal samples |

| Stochastic Noise | Adversarial Attack |
| --- | --- |
| • Model Independent<br>• Follow a distribution<br>• Slightly different from normal | • Design based on model<br>• Can be in any shape<br>• A few attack samples may significantly downgrade performance |





# Why Vulnerable?

## 1. Aim of Machine Learning

- A ML system typically aims to maximize performance, i.e. accuracy & efficiency
- Security is usually neglected





# Why Vulnerable?

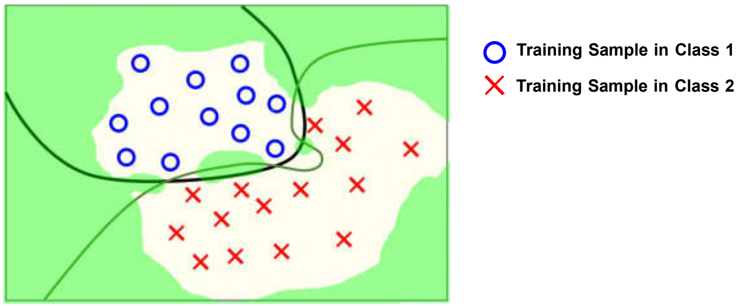
## 2. Machine Learning Assumptions

- Samples are independent and identically distributed (i.i.d.)
- Training and test samples follow the same (similar) distributions
- Implication:
  - 100% trust in the samples
  - Not consider a change of distribution
  - Samples are independent of a model
- All are violated by adversarial attacks

## Why Vulnerable?

**3. Uncertain situations**

- Samples are limited but the space is infinite






## Is Your ML System Safe?

- How can we know whether it is safe?
  - Try to attack it! Identify vulnerabilities
- Then, Improve its robustness

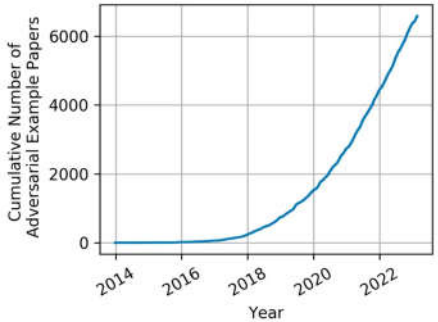
- Arms race between adversary and defender

B. Biggio, G. Fumera, F. Roli(2014) Security evaluation of pattern classifiers under attack. In: IEEE transactions on knowledge and data engineering



## Adversarial Learning

- **Adversarial Learning**
  Study on machine learning in adversarial environments in which decisions of models will be misled




Nicholas Carlini, "A Complete List of All (arXiv) Adversarial Example Papers"



## Adversary's Aspect


Adversary's Goal

Adversary's Capability

Adversary's Knowledge

Barreno et al., Can Machine Learning Be Secure? ASIACCS '06

# Adversary's Goal

- Cause security violation
  - An adversary forces a ML system to
    - **Learn wrong** things
    - **Do wrong** things
    - **Reveal wrong** things

Integrity

Mis-operate on some situations but do not compromise normal ones

Availability

Compromise normal system operation

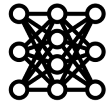
Confidentiality/Privacy

Reveal confidential information

Barreno et al., Can Machine Learning Be Secure? ASIACCS '06



# Adversary's Knowledge



**Machine Learning System**

Trained Parameters | Learning Parameters | Model | Model's Output | Training Sample | Feature Space

**Perfect Knowledge**
**White-box**
**ALL** knowledge

**Limited Knowledge**
**Grey-box**
**Partial** knowledge

**No Knowledge**
**Black-box**
**"No"** knowledge
(Feature Space may be obtained)

B. Biggio, G. Fumera, F. Roli, IEEE TKDE 2014



# Adversary's Capability

- Adversary should **not be omnipotent**
  - Messages of an email should be delivered to human
  - Malware must able to be executed and generate some damages
- **Concealment** should be considered
  - Contaminated samples should be similar to the clean ones

- **Constrains**
  - Number of manipulated samples
  - Number of manipulated features
  - Maximum amount of modifications on a feature



# Adversary's Capability

| Attack Type | Training Phase | Inference Phase | Manipulation |
|---|---|---|---|
| Evasion Attack | No | Yes | Feature |
| Poisoning Attack | Yes | No (maybe) | Feature / Label / Model |

## Attack Types

| | | Attacker's Goal | | |
|---|---|---|---|---|
| | | **Integrity** Mis-operate on some situations but do not compromise normal ones | **Availability** Compromise normal system operation | **Privacy / Confidentiality** Reveal confidential information |
| **Attacker's Capability** | **Test data** | Evasion (Adversarial Attack) | Sponge Attacks | Model Stealing Training Set Recovery |
| | **Training data** | Integrity Poisoning e.g. Targeted Poisoning Attack, Backdoor Attack | Indiscriminate Poisoning Attack, e.g. DoS | / |

Biggio & Roli, Wild Patterns, PR 2018 https://arxiv.org/abs/1712.03141



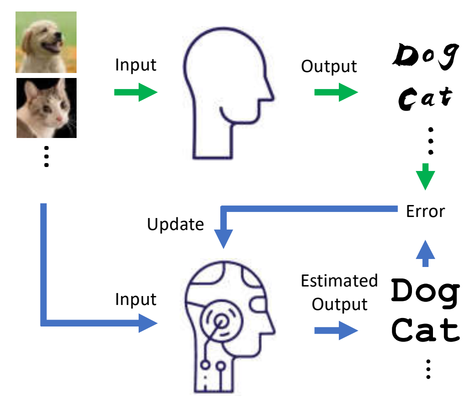
## Course Syllabus

- Ch01 Overview
- Ch02 Evasion Attacks & Countermeasures
- Ch03 Poisoning Attacks & Countermeasures
- Ch04 Privacy Attacks & Countermeasures
  Physical Attacks
  Non-Security Applications
  Conclusion




## Machine Learning

- Machine Learning can be treated as Function Approximation



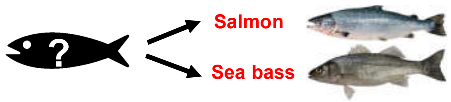
# What is Learning?

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

**Task T**
Separate Salmon and Sea Bass

**Performance P**
Accuracy on identification

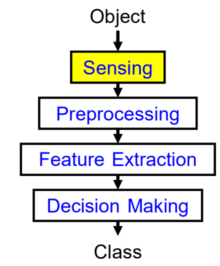
**Experience E**
Caught Salmon and Sea Bass






Machine Learning Procedure
# Sensing

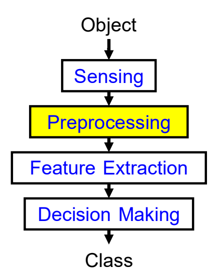
- **Digitize** the object to the format which can be handled by machines
- Example
  - Type of Device
    Camera? Depth Camera? Infra-red? Ultrasound? Movement Sense? Combination?
  - Setting of Device
    Number? Angle? Overlap shooting range?
  - Background
    Lighting? Background simplicity?






Machine Learning Procedure
# Preprocessing

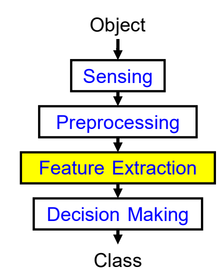
- **Refine** the data
- Example
  - Lighting conditions
  - Position of fish
  - Angle of fish
  - Noise
  - Blurriness
  - Segmentation (remove object from background)





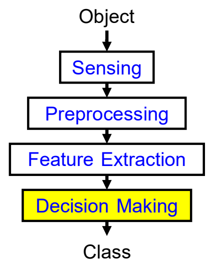

Machine Learning Procedure
# Feature Extraction

- Decide which **information** is able to distinguish classes
- Example
  - Length, width, weight, number and shape of fins, tail shape, etc.
- Rely on technical background and common sense
  - Experts may help



Machine Learning Procedure
## Decision Making

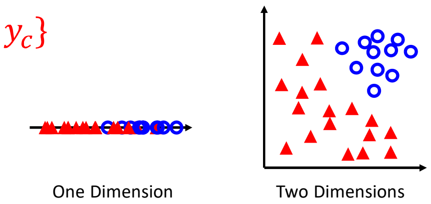
- Decision Type:
  - Class (Classification)
  - Value (Regression, Value Prediction)
  - Rank (Ranking)
  - Action (Reinforcement Learning)
  - Region (Segmentation)
- Many machine learning techniques are available






## Classification

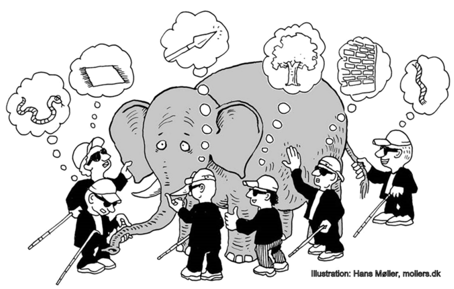
- **Classification** is mainly focused in this course
  - An important ad popular application of machine learning
  - Aim to assign a sample to a class
  - Sample = Feature Vector : $\boldsymbol{x} = [x_1, x_2, \ldots, x_d] \in X$
    - d : feature number
  - Class : $y \in Y, Y = \{y_1, y_2, \ldots, y_c\}$
    - c : class number






## Classification: Formulation

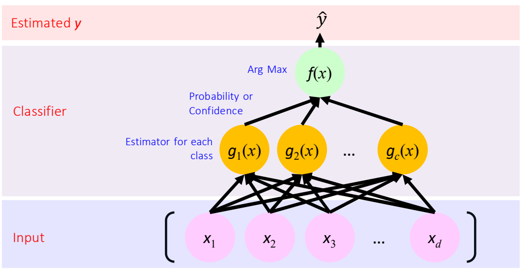
- How to formulate a classification problem $X \rightarrow Y$?
  - Input sample X is a real vector
  - Class Y is discrete
  - Not convenient to calculate, e.g. 1 + 2 + 3 = Class 1?






## Classification: Formulation

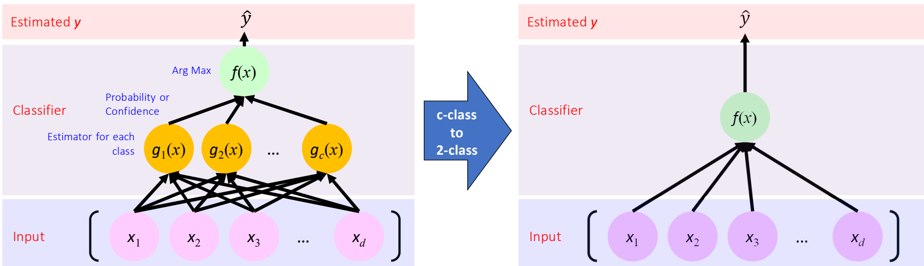
- Probability Estimation of x belongs to a class
  - Contains a set of discriminant functions $g_i(\boldsymbol{x})$, $i = 1, \ldots, c$ indicates how likely $\boldsymbol{x}$ belongs to $y_i$
  - $\boldsymbol{x}$ is assigned to class $y_i$ if $g_i(x)$ is max for $i = 1 \ldots c$



# Classification: Formulation

- A two-class problem is a special case
  - Only one function is required

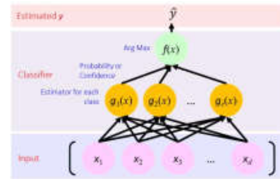
$$g_1(x) > g_2(x),\ x \text{ belongs to class 1}$$
$$g_1(x) - g_2(x) > 0$$
$$f(x) > 0$$






# Classification: Formulation

**Multi-Class Problem**



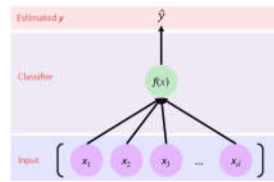
Original Dataset

| $x$ | $y$ |
|---|---|
| 23 | 1 |
| 42 | 2 |
| 52 | 3 |
| 12 | 4 |

| $g_1(x)$ | | $g_2(x)$ | | $g_3(x)$ | | $g_4(x)$ | |
|---|---|---|---|---|---|---|---|
| $x$ | $y^{(1)}$ | $x$ | $y^{(2)}$ | $x$ | $y^{(3)}$ | $x$ | $y^{(4)}$ |
| 23 | 1 | 23 | 0 | 23 | 0 | 23 | 0 |
| 42 | 0 | 42 | 1 | 42 | 0 | 42 | 1 |
| 52 | 0 | 52 | 0 | 52 | 1 | 52 | 0 |
| 12 | 0 | 12 | 0 | 12 | 0 | 12 | 0 |

$$Loss = \sum_{i=1}^{c}\left(g_i(\boldsymbol{x}) - y^{(i)}\right)^2$$

**Two-Class Problem**



Original Dataset

| $x$ | $y$ |
|---|---|
| 23 | 1 |
| 42 | 1 |
| 52 | 2 |
| 12 | 2 |

$f(x)$

| $x$ | $y$ |
|---|---|
| 23 | 1 |
| 42 | 1 |
| 52 | -1 |
| 12 | -1 |

$$Loss = (f(\boldsymbol{x}) - y)^2$$



# Classification: Formulation

- Can a multi-class problem also be formulated like this?

Original Dataset

| $x$ | $y$ |
|---|---|
| 23 | 1 |
| 42 | 2 |
| 52 | 3 |
| 12 | 4 |

$g(x)$

| $x$ | $y$ |
|---|---|
| 23 | 1 |
| 42 | 2 |
| 52 | 3 |
| 12 | 4 |

$$Loss = (g(\boldsymbol{x}) - y)^2$$

$$f(x) = \begin{cases} y_1 & g(x) < 1.5 \\ y_2 & 1.5 \le g(x) < 2.5 \\ y_3 & 2.5 \le g(x) < 3.5 \\ y_4 & 3.5 \le g(x) \end{cases}$$

Original Dataset

| $x$ | $y$ |
|---|---|
| 23 | 1 |
| 42 | 2 |
| 52 | 3 |
| 12 | 4 |

| $g_1(x)$ | | $g_2(x)$ | | $g_3(x)$ | | $g_4(x)$ | |
|---|---|---|---|---|---|---|---|
| $x$ | $y^{(1)}$ | $x$ | $y^{(2)}$ | $x$ | $y^{(3)}$ | $x$ | $y^{(4)}$ |
| 23 | 1 | 23 | 0 | 23 | 0 | 23 | 0 |
| 42 | 0 | 42 | 1 | 42 | 0 | 42 | 1 |
| 52 | 0 | 52 | 0 | 52 | 1 | 52 | 0 |
| 12 | 0 | 12 | 0 | 12 | 0 | 12 | 0 |

$$Loss = \sum_{i=1}^{c}\left(g_i(\boldsymbol{x}) - y^{(i)}\right)^2$$

$$f(x) = y_i, \text{ where } i = \underset{j}{argmax}\, g_j(x)$$

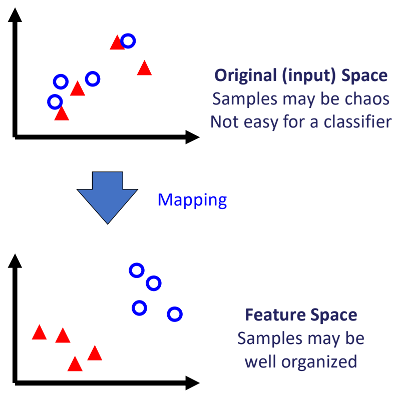


# Classification: Loss Function

- Aim to minimize **the loss function**
  - Less loss means better performance
- Different levels of description
  - Loss function on a sample $\quad L = (f(x) - y)^2$
  - Loss function including explicit w on a sample $\quad L(\boldsymbol{w}) = (f_{\boldsymbol{w}}(x) - y)^2 \quad$ w denotes the parameters
  - Loss function including explicit w on n samples (usually mean training set) $\quad L(\boldsymbol{w}) = \sum_{i=1}^{n}(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$

# Mapping

- Practically, a classification problem is complicated
- Not easily to train a complicated classifier with good performance
- Map samples to a high-dimensional space, which may separate classes better than the original space



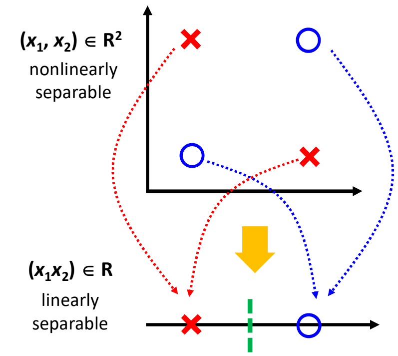



# Mapping

- XOR Example

| $x_1$ | $x_2$ | y |
|---|---|---|
| 1 | 1 | 1 |
| -1 | 1 | -1 |
| 1 | -1 | -1 |
| -1 | -1 | 1 |

| $x_1$ | $x_2$ | $x_1x_2$ | y |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| -1 | 1 | -1 | -1 |
| 1 | -1 | -1 | -1 |
| -1 | -1 | 1 | 1 |



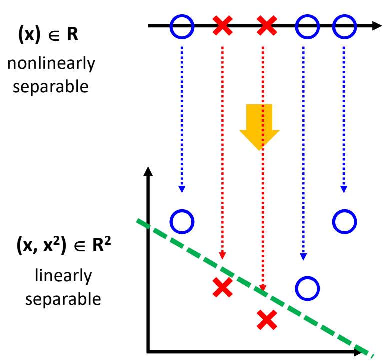



# Mapping

- Another Example

| x | y |
|---|---|
| -2 | 1 |
| -1 | -1 |
| 0 | -1 |
| 1 | 1 |
| 2 | 1 |

| x | x | $x^2$ | y |
|---|---|---|---|
| -2 | -2 | 4 | 1 |
| -1 | -1 | 1 | -1 |
| 0 | 0 | 0 | -1 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 4 | 1 |






Classifier

# SVM: Linearly Separable

- Support Vector Machine (SVM)
- Problem can be formulated as Quadratic Optimization Problem and solve for $w$ and $b$

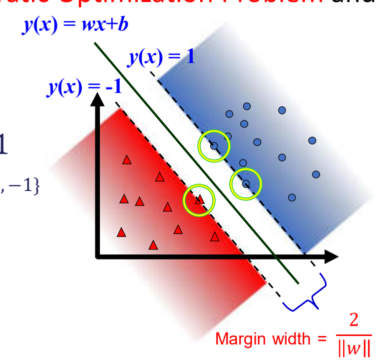
Margin Width

$$\underset{w,\, b}{\text{minimize}} \quad \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1$$

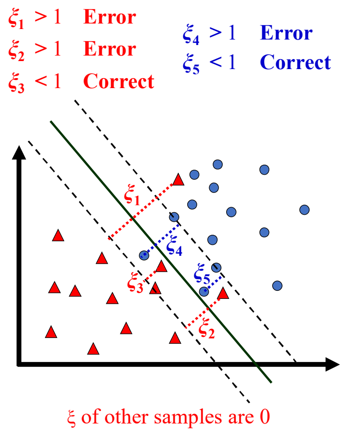
$$\text{where} \quad i = 1 \ldots n \quad \text{and} \quad y = \{1, -1\}$$

All samples should be behind the margin



Classifier

## SVM: Non-Linearly Separable




- **Slack Variable ($\xi$)** is added as a punishment to allow a sample in / far away from the margin
- Optimization:

Margin Width　　Punishment

$$\underset{w,\,b}{\text{Minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i \quad i = 1 \dots N$$

$$\xi_i \geq 0$$

Punishment allow a sample not behind the margin

where
$C$ : tradeoff parameter between error and margin

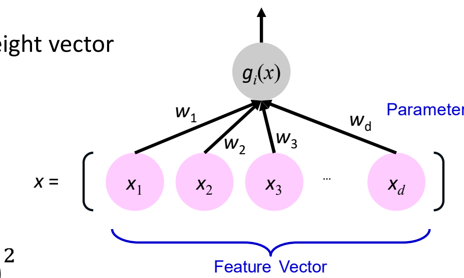


Classifier

## Linear Discriminant Function

- LDF: a linear combination of $x$

$$g(x) = \sum_{i=1}^{d} w_i x_i \qquad w : \text{is the weight vector}$$




- How to train g(x)?
  - Minimize

$$L(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(g_w\left(x^{(i)}\right) - y^{(i)}\right)^2$$



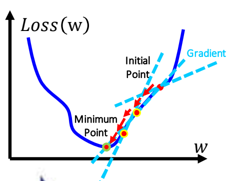
Classifier

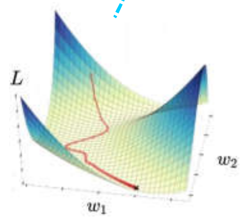
## Gradient Descent

- When $h_w$ is differentiable, gradient descent can be used to minimize the Loss Function
- Influence on $L(\mathrm{w})$ by changing $w$ slightly

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha\frac{\partial L(\boldsymbol{w}^{(t)})}{\partial \boldsymbol{w}}$$

  - $\alpha$ : the learning rate
  - $\boldsymbol{w}^{(t)}$ : the parameters at the time t





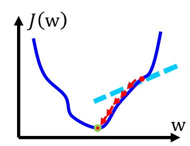

Classifier

## Gradient Descent

- **Algorithm**
  - Start with an arbitrarily chosen weight $\mathrm{w}^{(1)}$
  - Let t = 0
  - Loop
    - t = t + 1
    - Compute gradient vector $\partial Loss(\mathrm{w}^{(t)})/\partial \mathrm{w}$
    - Next value $\mathrm{w}^{(t+1)}$ determined by moving some distance from $\mathrm{w}^{(t)}$ in the direction of the steepest descent

$$\mathrm{w}^{(t+1)} = \mathrm{w}^{(t)} - \alpha\frac{\partial Loss(\mathrm{w}^{(t)})}{\partial \mathrm{w}}$$

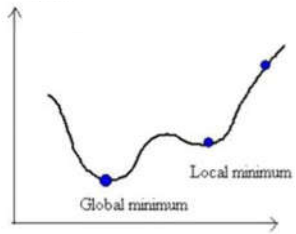
      - i.e., along the negative of the gradient
  - Until Finish Training (Control by number of updates or size of $\partial Loss(\mathrm{w}^{(t)})/\partial \mathrm{w}$)



Classifier

# Gradient Descent

- Related Issues:
  - Size of Learning Rate ($\alpha$)
    - Too small, convergence is needlessly slow
    - Too large, the correction process will overshoot and cannot even diverge
  - Sub-optimal Solution
    - Trapped by local minimum

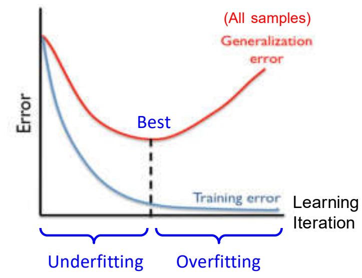





Classifier

# Gradient Descent

- What is the objective of a classifier?
  - Classify training samples accurately?
    - Training Error (Empirical Error) ($R_{emp}$)
      - Error of the training samples, computable
      - Training Objective
  - Classify unseen samples accurately?
    - Generalization Error ($R_{gen}$)
      - Non-computable, estimate only
      - Ultimate Objective
- Training and ultimate objectives are correlated but different

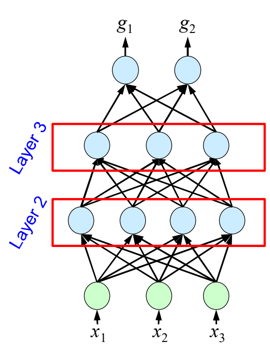





Classifier

# Multi-Layer Perceptron

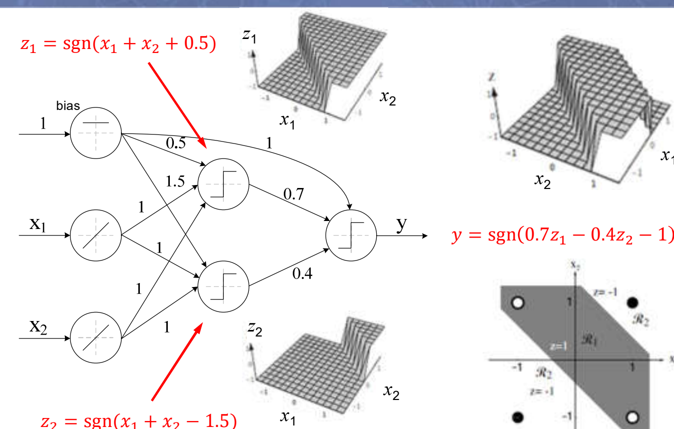
- Multi-Layer Perceptron
  - Neurons are arranged in layers
  - A neuron is connected to all neurons in next layer
    - Fully-connected
    - Feedforward
  - Neurons may have different activation functions or no activation function






Classifier: Multi-Layer Perceptron

# XOR Example

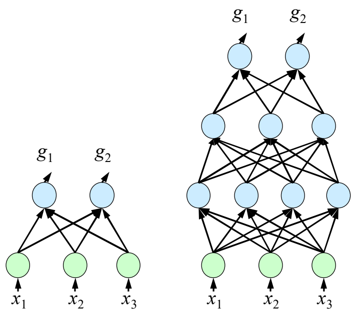


# Backpropagation

- How to determine the weight?
  - Gradient Descent

$$w^{(k+1)} = w^{(k)} + \alpha \frac{\partial J(w^{(k)})}{\partial w}$$

  - $\alpha$: the learning rate
  - How to calculate $\partial J(w)/\partial w$ for each $w$?



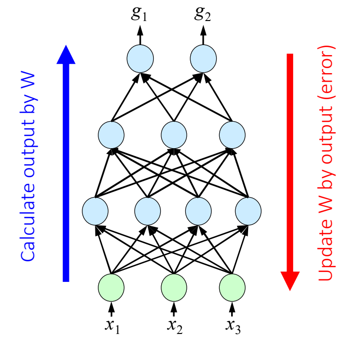

LDF MLP





# Backpropagation

- Backpropagation
  - Calculation of the derivative flows backwards through the network








# Backpropagation

- Recall, Chain rule

$$f(x) = sin(\cos(x^2))$$

$$\frac{\partial f(x)}{\partial x} = \frac{\partial sin(\cos(x^2))}{\partial x}$$

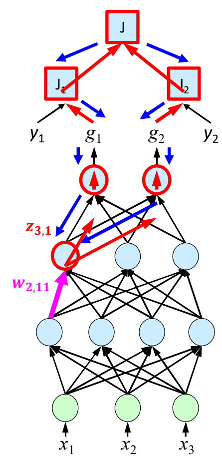
$$= \frac{\partial sin(\cos(x^2))}{\partial \cos(x^2)} \frac{\partial \cos(x^2)}{\partial x}$$

$$= \frac{\partial sin(\cos(x^2))}{\partial \cos(x^2)} \frac{\partial \cos(x^2)}{\partial x^2} \frac{\partial x^2}{\partial x}$$





# Example

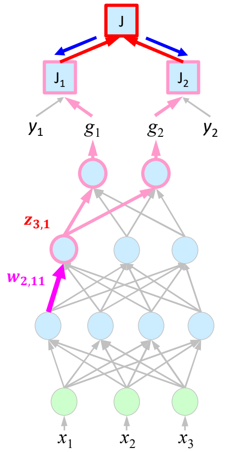



- Which paths to the output are affected by $w_{2,11}$?
- Error on each output should be considered

$$J\left(w^{(k)}\right) = J_1 + J_2$$

- Backprop from $J$ to $w_{2,11}$

# Example




$$\frac{\partial J(w^{(k)})}{\partial w_{2,11}}$$

$$\frac{\partial J(w^{(k)})}{\partial w_{2,11}}$$

$$= \frac{\partial (J_1 + J_2)}{\partial w_{2,11}}$$

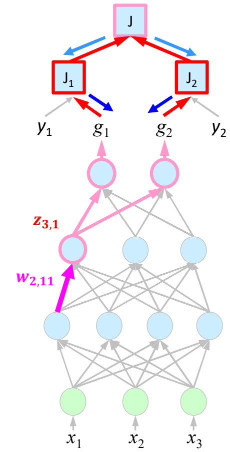
$$= \sum_{i=1}^{2} \frac{\partial J_i}{\partial w_{2,11}}$$

$\boxed{J(w^{(k)}) = J_1 + J_2}$





# Example




$$\frac{\partial J(w^{(k)})}{\partial w_{2,11}} = \sum_{i=1}^{2} \frac{\partial J_i}{\partial w_{2,11}}$$

$$\frac{\partial J_i}{\partial w_{2,11}}$$

$$= \frac{\partial \frac{1}{2}(y_i - g_i)^2}{\partial w_{2,11}}$$

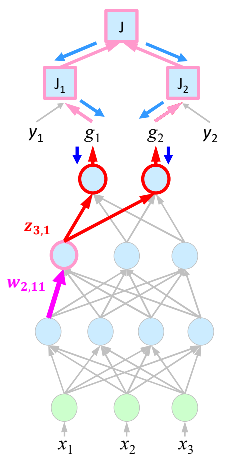
$$= -(y_i - g_i)\frac{\partial g_i}{\partial w_{2,11}}$$

$\boxed{J_i = \frac{1}{2}(y_i - g_i)^2}$





# Example




$$\frac{\partial J(w^{(k)})}{\partial w_{2,11}} = -\sum_{i=1}^{2} (y_i - g_i)\frac{\partial g_i}{\partial w_{2,11}}$$

$$\frac{\partial g_i}{\partial w_{2,11}}$$

$$= \frac{\partial a\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right)}{\partial w_{2,11}}$$

$\boxed{g_i = a\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right)}$

$$= \frac{\partial a\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right)}{\partial\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right)} \frac{\partial\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right)}{\partial w_{2,11}}$$

$\boxed{\text{Let } a'(x) = \frac{\partial a(x)}{\partial (x)}}$

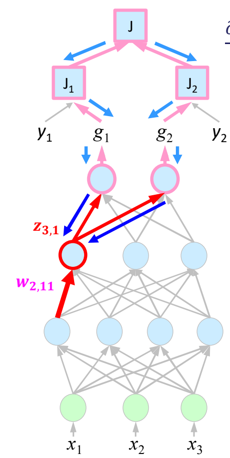
$$= a'\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right) \frac{\partial\left(z_{3,1}w_{3,1i} + z_{3,2}w_{3,2i} + z_{3,3}w_{3,3i}\right)}{\partial w_{2,11}}$$

$$= a'\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right) w_{3,1i}\frac{\partial z_{3,1}}{\partial w_{2,11}}$$





# Example




$$\frac{\partial J(w^{(k)})}{\partial w_{2,11}} = -\sum_{i=1}^{2} \left( (y_i - g_i) a'\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right) w_{3,1i}\frac{\partial z_{3,1}}{\partial w_{2,11}} \right)$$

$$\frac{\partial z_{3,1}}{\partial w_{2,11}}$$

$$= \frac{\partial a\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right)}{\partial w_{2,11}}$$

$\boxed{z_{3,1} = a\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right)}$

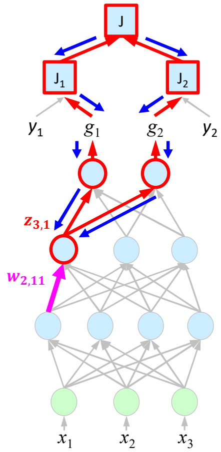
$$= \frac{\partial a\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right)}{\partial\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right)} \frac{\partial\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right)}{\partial w_{2,11}}$$

$\boxed{a'(x) = \frac{\partial a(x)}{\partial (x)}}$

$$= a'\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right) \frac{\partial \begin{pmatrix} z_{2,1}w_{2,11} + z_{2,2}w_{2,21} + \\ z_{2,3}w_{2,31} + z_{2,4}w_{2,41} \end{pmatrix}}{\partial w_{2,11}}$$

$$= a'\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right) z_{2,1}$$

# Example




$$\frac{\partial J(w^{(k)})}{\partial w_{2,11}} = -\sum_{i=1}^{2}\left(\begin{array}{c}(y_i - g_i)a'\left(\sum_{j=1}^{3} z_{3,j}\, w_{3,ji}\right) w_{3,1i}\\ a'\left(\sum_{k=1}^{4} z_{2,k}\, w_{2,k1}\right) z_{2,1}\end{array}\right)$$

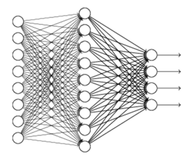
$$\boxed{a'(x) = \frac{\partial a(x)}{\partial(x)}}$$

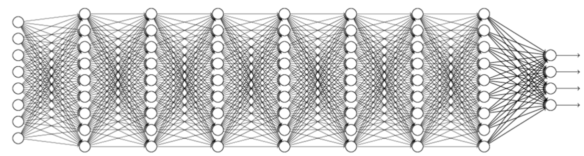




# What is Deep Learning?

- Branch of Machine Learning
- Commonly refer to **a neural network with multiple layers** (deep architecture)



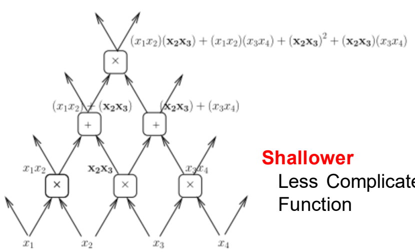






# Why Deep Learning?

- Our brain is a very deep architecture
- A deep architecture can represent more complicated function than a shallow one



**Deeper**
More Complicated Function

**Shallower**
Less Complicated Function

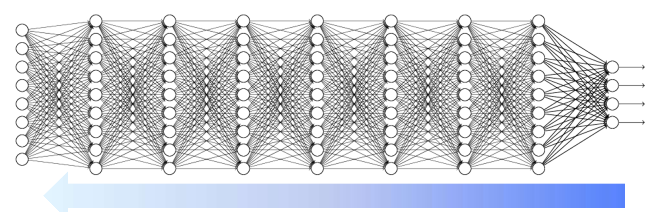
*Yoshua Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), 2009*





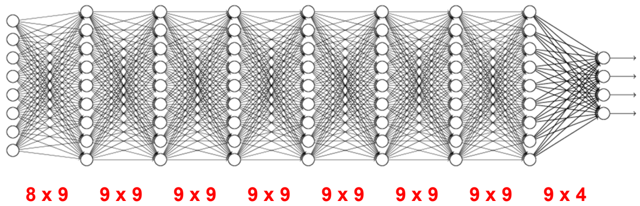
# What's New?

- DNN is less accurate than shallow one by using traditional backpropagation
  - Backpropagation loses its power in deep architecture
  - Vanishing gradient problem



## What's New?

- DNN is less accurate than shallow one by using traditional backpropagation
  - Optimization is very complex
  - Too many parameters in deep architecture








## What's New?

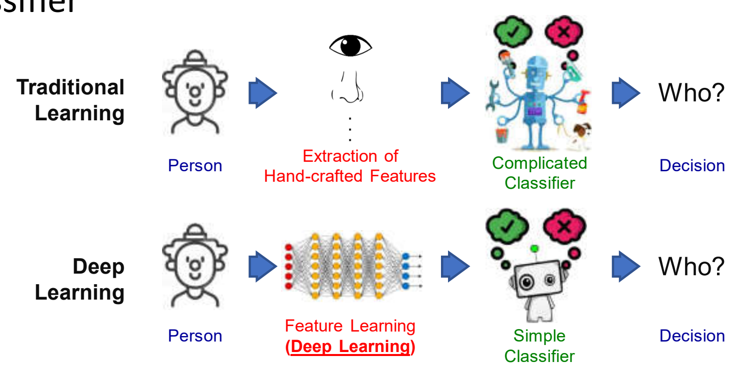
- Vanishing Gradient Problem
  - **Divide and Conquer**
    - Stacked training
    - E.g. Stacked Autoencoder (SA)
  - **Reduce Parameters**
    - Too many parameters
    - E.g. Convolutional Neural Network (CNN)





## Feature Learning

- Deep Learning focuses on feature learning but not a classifier

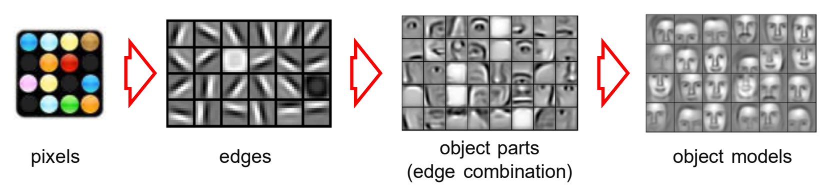







## Feature Learning

- Features extracted by deep learning



## Other Machine Learning Types

- **Regression**
  - A statistical modeling technique used to predict continuous variables based on the relationship between independent and dependent variables.
- **Multi-label classification** (**Tagging**)
  - A classification problem where an instance can be assigned multiple labels simultaneously, allowing for more flexible and nuanced categorization.
- **Recommendation**
  - A system or algorithm that suggests items, products, or content to users based on their preferences, behaviors, or similarities to other users.
- **Reinforcement Learning**
  - A branch of machine learning where an agent learns to make decisions or take actions in an environment to maximize a reward signal, often through trial and error.



## Classifier Comparison

- For a classification problem, given
  - Dataset D
  - Classifiers A and B
- How can we measure which classifier, A or B, is better for D?



## Classifier Comparison

- Method
  - Randomly separate D into **training** and **test sets**
  - Use **Training Set** to train A and B
  - Use **Test Set** to evaluate the performances of trained A and B
  - Select the better performing classifier
- **Is it ok?**
  - The winner may just be lucky in performing better for that particular test set.
  - No guarantee for different test sets



## Classifier Comparison

- The bias of test set should be reduced
- Two re-sampling techniques
  - Independent Run
  - Cross-Validation

Classifier Comparison

# Independent Run

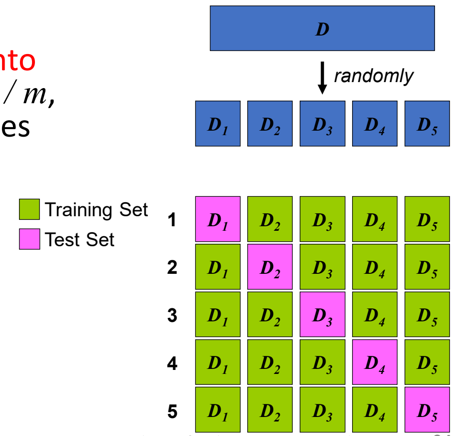
- Statistical method
- Also called Bootstrap and Jackknifing
- Repeat the experiment "n" times independently
  - Repeat $n$ times
    - $i$ is the number of running time
    - Randomly separate $D$ into Training Set$_i$ and Test Set$_i$
    - Use Training Set$_i$ to train $A_i$ and $B_i$
    - Use Test Set$_i$ to evaluate the trained $A_i$ and $B_i$
  - Select the classifier with higher average accuracy



Classifier Comparison

# Cross-Validation

- M-fold Cross-Validation
- Dataset D is randomly divided into $m$ disjoint sets $D_i$ of equal size $n / m$, where $n$ is the number of samples in dataset
- Repeat $m$ times
  - Trained by $D_j$
  - Evaluated by all $D_i$ except $D_j$
- Select the classifier with higher average accuracy






Machine Learning

# Terminology

- **Instance / Sample**
  Observations from an application
- **Feature / Attribute**
  Property or characteristic of a sample
- **Dimensionality**
  The number of features



Machine Learning

# Terminology

- **Training Set**
  A set of samples used to train a model
- **Test Set**
  A set of samples used to evaluate the performance of the trained model.
  Usually separate from the training set.
- **Unseen Samples**
  Any samples not in training set

Machine Learning
# Terminology

- **Training Error**
  Error on training samples
- **Test Error**
  Error on test samples
- **Generalization Error**
  The ability of a model to perform well on unseen samples

  In some discussion,
  Test Error = Generalization Error



Machine Learning
# Terminology

- **Objective Function / Error Function / Loss Function**

  A mathematical function used to quantify error made by a model, closely related to the objective

  Can be more than error on samples, may include any other concepts

  E.g. complexity of a model