Deep Learning

Lecture 2: Machine Learning Basics I

National University of Singapore

Instructor: Joey Zhou

What is machine learning?

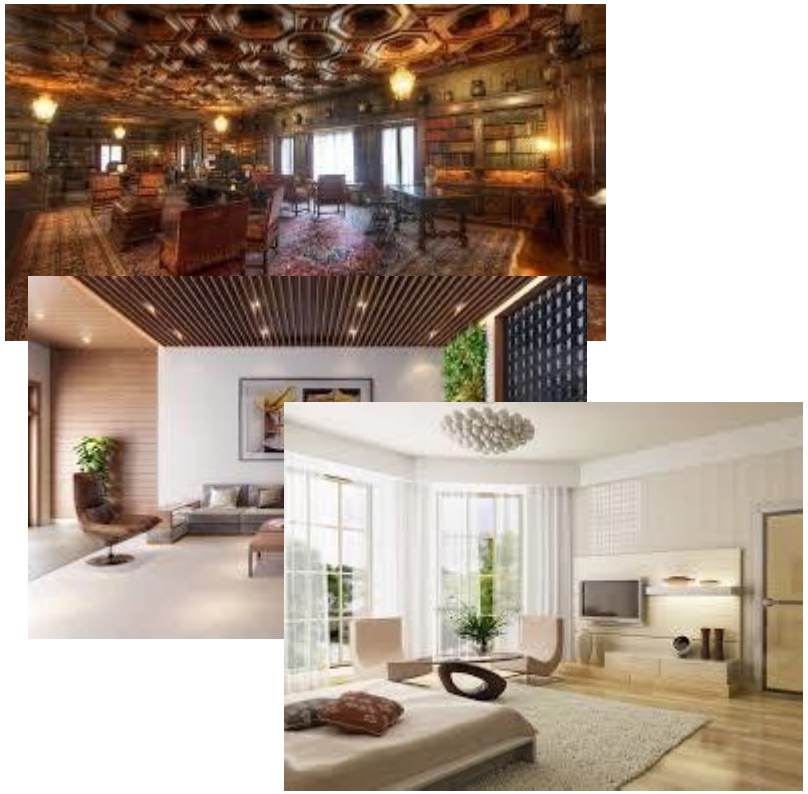
- "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T as measured by P, improves with experience E."

------ *Machine Learning*, Tom Mitchell, 1997

Example 1: image classification

Task: determine if the image is indoor or outdoor
Performance measure: probability of misclassification

Example 1: image classification



Experience/Data:
images with labels

Indoor

outdoor

Example 1: image classification

- A few terminologies
  - Training data: the images given for learning
  - Test data: the images to be classified
  - Binary classification: classify into two classes

Example 1: image classification (multi-class)

ImageNet figure borrowed from vision.standford.edu

Example 2: clustering images

Task: partition the images into 2 groups
Performance: similarities within groups
Data: a set of images

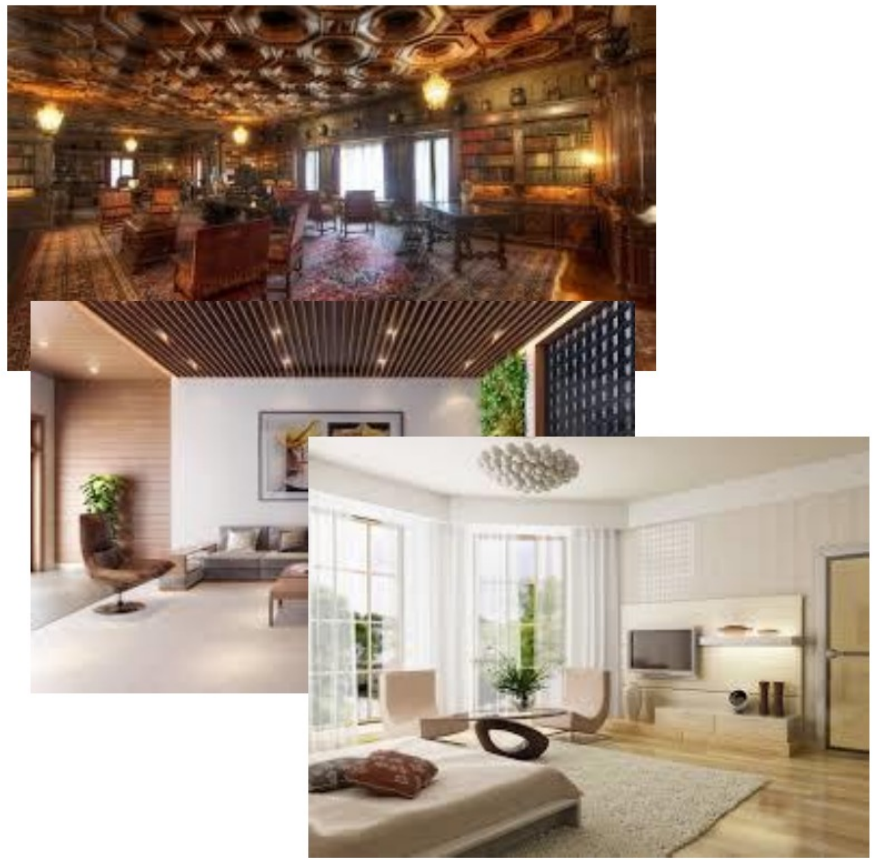
Example 2: clustering images

- A few terminologies
  - Unlabeled data vs labeled data
  - Supervised learning vs unsupervised learning

Machine learning 1-2-3

- Collect data and extract features
- Build model: choose hypothesis class $\mathcal{H}$ and loss function l
- Optimization: minimize the empirical loss

Example 1: image classification

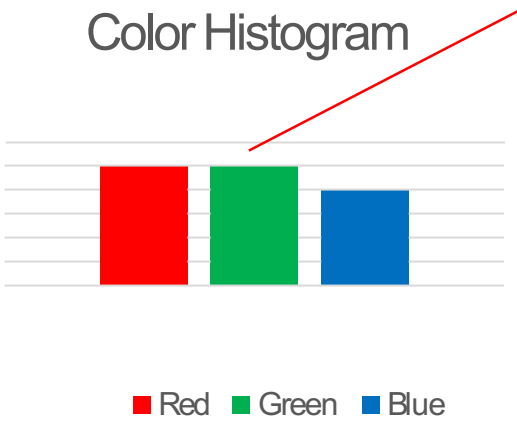


Indoor

outdoor

Hand-craft features

Extract features



Feature vector: x_i

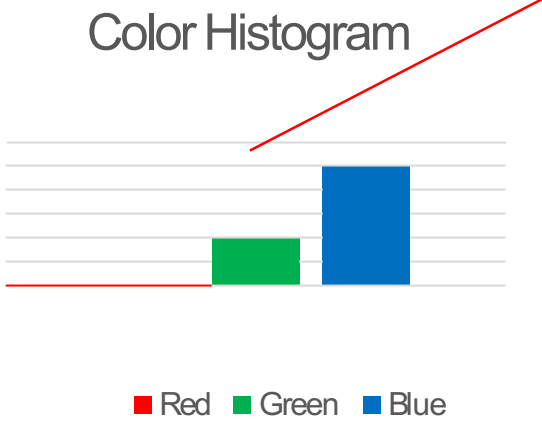
Indoor

0

Label: y_i

Hand-craft features

Extract features



Feature vector: x_j

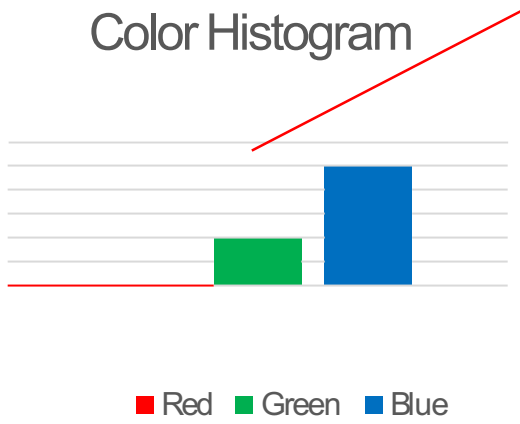
outdoor

1

Label: y_j

Hand-craft features

Extract features



Feature vector: x_j

indoor

1

Label: y_j

What if we face this kind of interior design?

Sheepdog or mop?

Chihuahua or muffin?

Data Representation

Many **AI** tasks can be solved by designing "**right set of features to extract**" for the task to be provided to a simple machine learning algorithm.

Example: "*Write a 'program' to detect cars in image*"

Cars have *wheels* … use "presence of a *wheels* " as a feature? BUT difficult to describe what a *wheel* looks like in terms of **PIXEL** values. *Wheel* has a simple geometric shape BUT its image may be complicated by shadows, sun glaring off metal parts etc.

SOLUTION: use **ML** to **DISCOVER** not only the **MAPPING FROM REPRESENTATION** to output but also the representation itself. .. this approach is known as **REPRESENTATION LEARNING**.

Learned representations often result in better performance than obtainable with "**hand-designed**" representations.. also enable **AI** systems to rapidly adapt to new tasks, with minimal human intervention.

a "**representation learning algorithm**" can discover a good set of features for a simple task in minutes or for a complex task in hours to months. Manually designing features for a complex task requires a great deal of human time & effort.

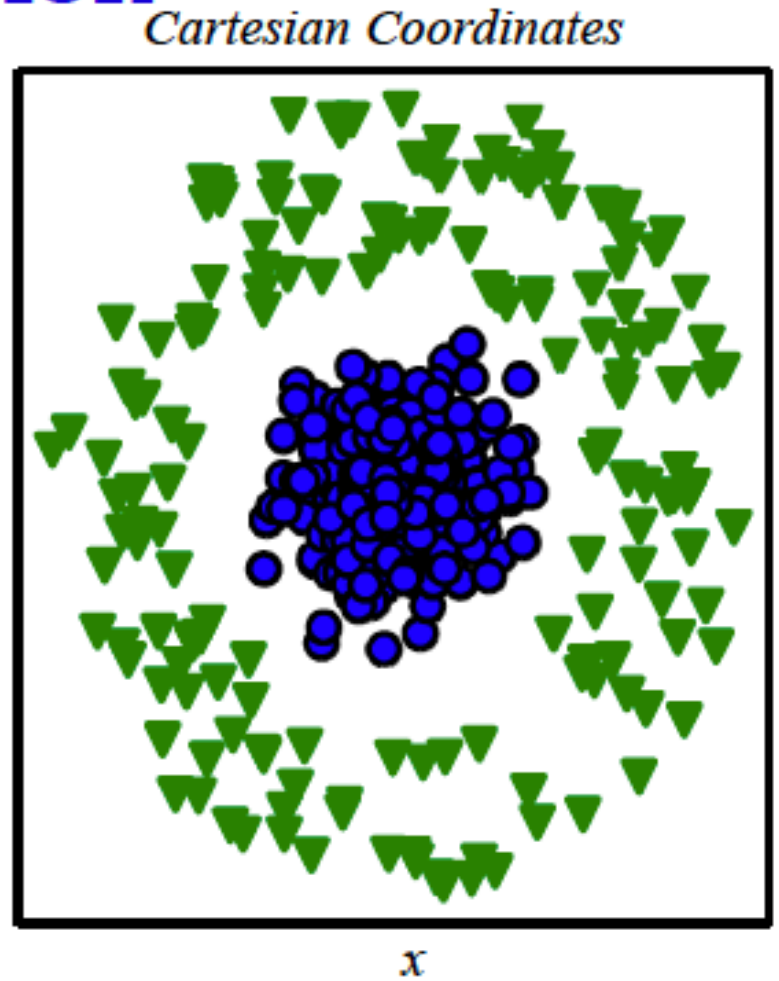
Data Representation

Machine learning algorithms depend heavily on the representation **of the data** they are given:

- **Deep Learning** concerned with **learning** how to represent data in a **Machine Learning (ML)** system

Example:

- scatter plot of two categories of data in 2D space .. y
- problem made easy by changing the way data is represented!



Data Representation

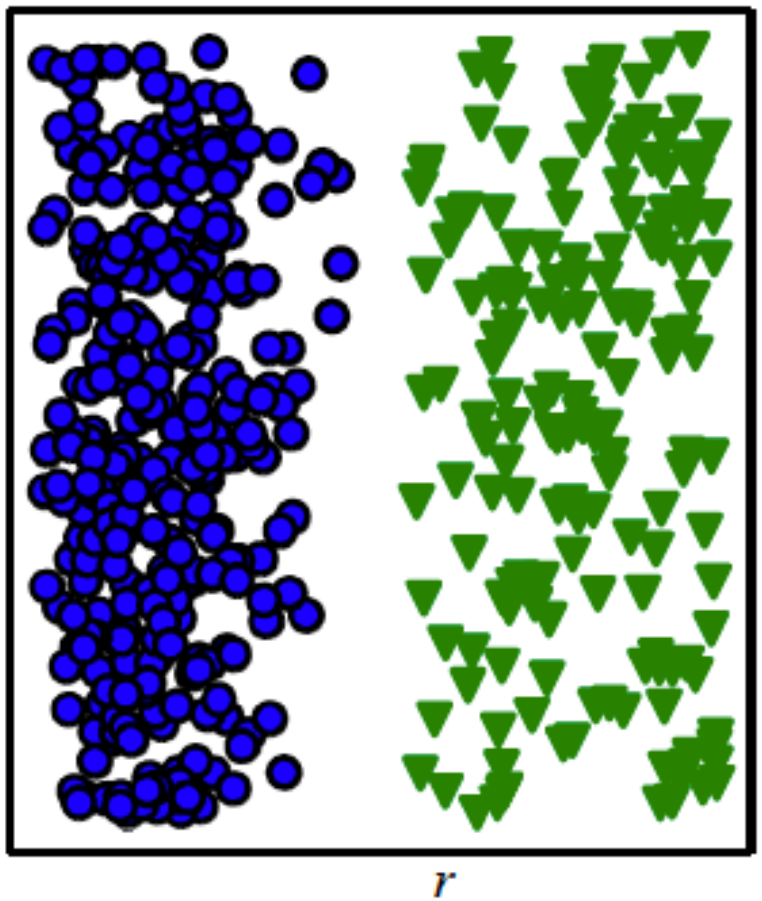
Machine learning algorithms depend heavily on the **representation** **of the data** they are given:

- **Deep Learning** concerned with **learning** how to represent data in a **Machine Learning** (**ML**) system

Example:

- scatter plot of two categories of data in 2D space ..
- problem made easy by changing the way data is **represented**!

ML has been applied successfully to challenging problems .. through **changing** **representation** **of data**...



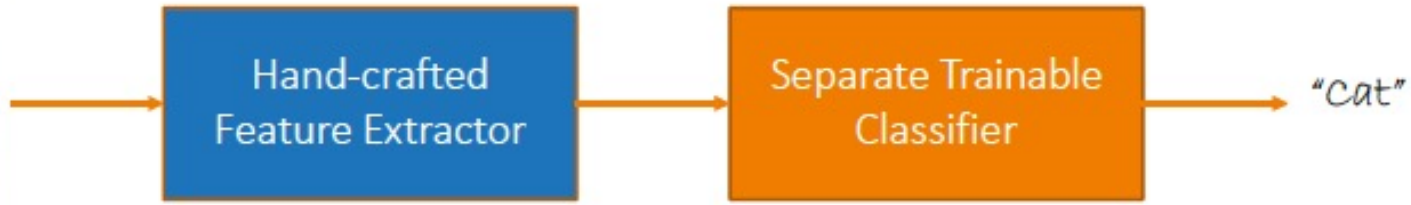
Wait…

- Why handcraft the feature vectors x, y?
  - Can use prior knowledge to design suitable features

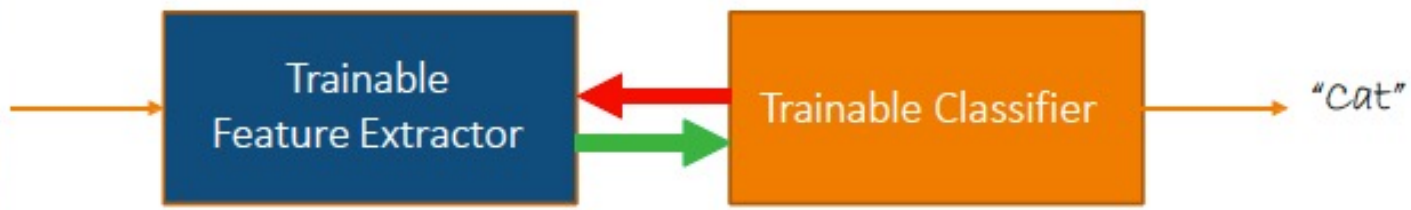
- Can computer learn the features on the raw images?
  - Learn features directly on the raw images: Representation Learning
  - Deep Learning $\subseteq$ Representation Learning $\subseteq$ Machine Learning $\subseteq$ Artificial Intelligence

Learning Representations & Features

Traditional pattern recognition



End-to-end learning → *Features are also learned from data*



Learning .. Features

Manually designed (*hand-crafted*) **features:**

usually take much time to come up with, implement & validate

often incomplete; not easily known if they are optimal for the task

Learned features

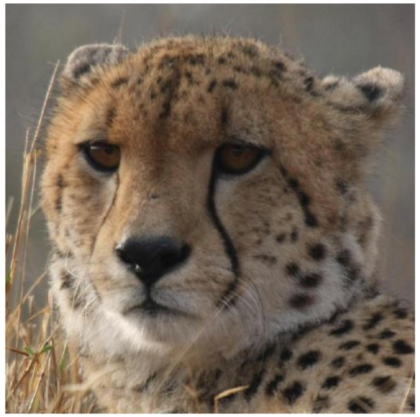
easy to adapt

compact and specific to the task at hand

relatively easy and fast to optimize

Over-confidence Issue

Training data



Prediction: dog
Probability: 0.98

Prediction: dog
Probability: 0.95

The model used was Resnet-18, which yields ~99% accuracy on the validation set. Only using this for evaluation would have us believe it's an amazing model, but that's not why we're here. Below is the image of blogger and a dog, where apparently the blogger is more dog than this actual dog. Even worse, it's 98% confident that he is a dog, so he will be a dog even if we were only considering predictions with over 95% confidence.

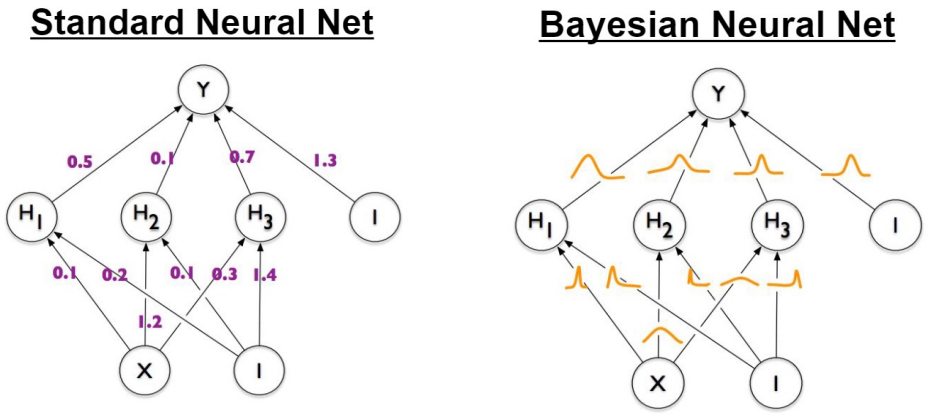
So how to solve this issue?

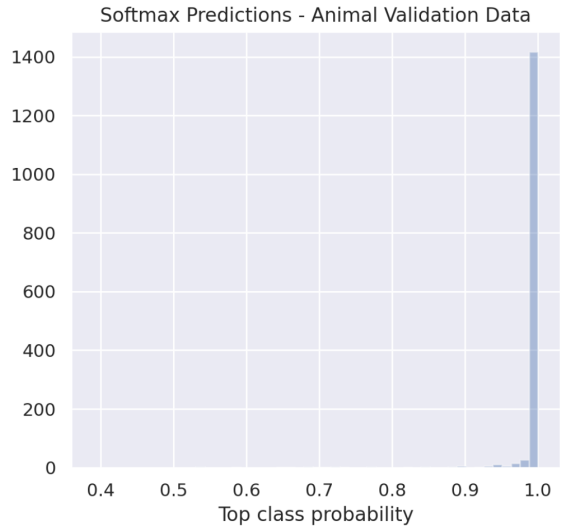
https://jramkiss.github.io/2020/07/29/overconfident-nn/

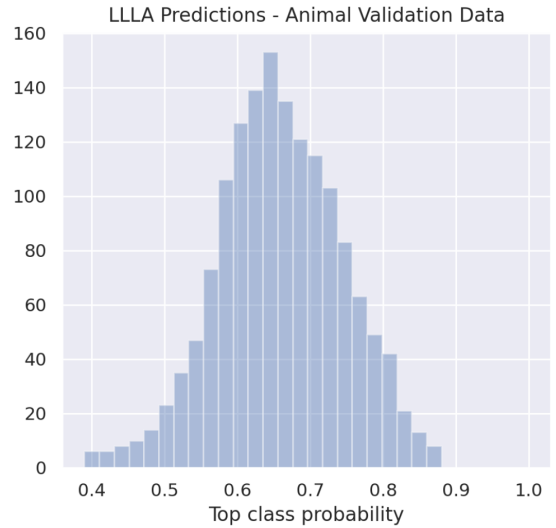
Possible Solutions

1. Collect more data on the human and build up human classifier.

2. Bayesian methods are great for quantifying uncertainty.



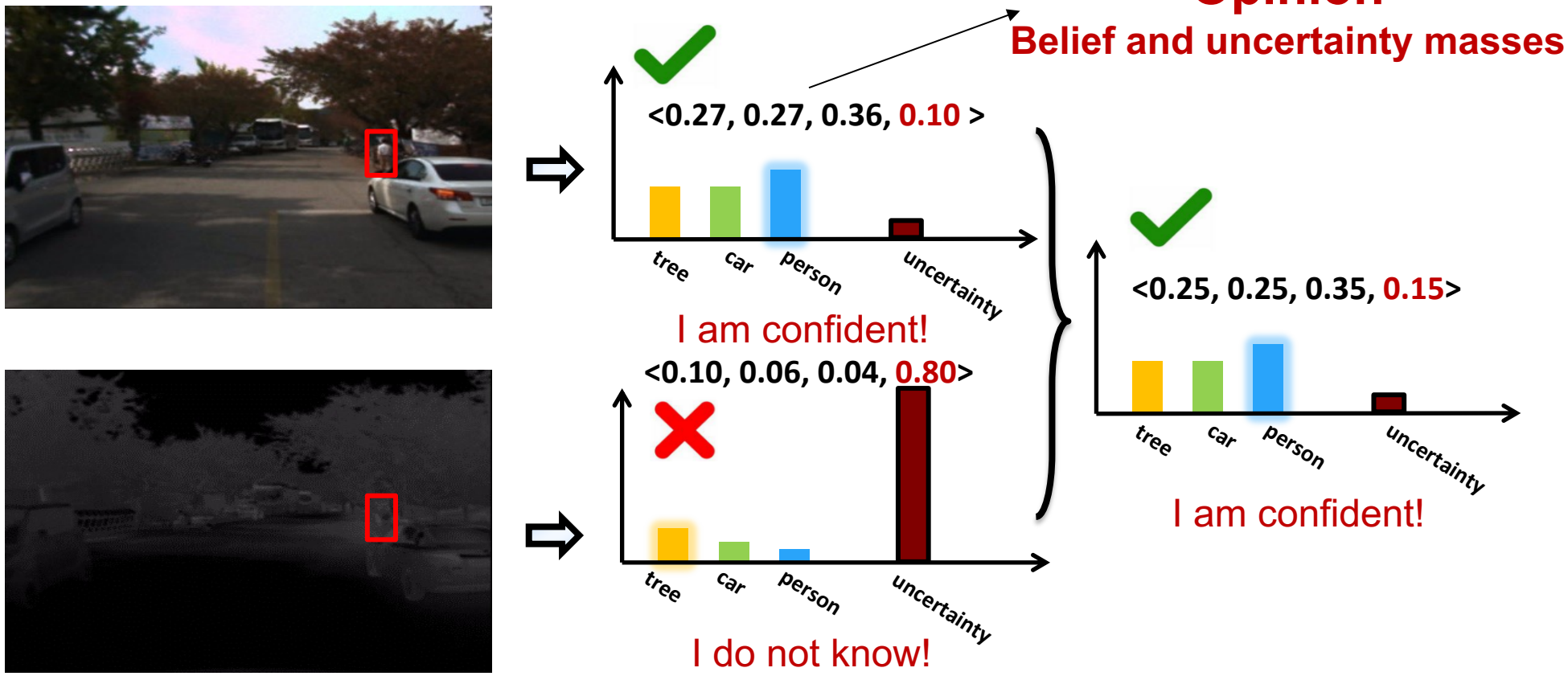




Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Network, ICML 2020.

3. Trustworthy Machine Learning

Predictive uncertainty is directly modeled.

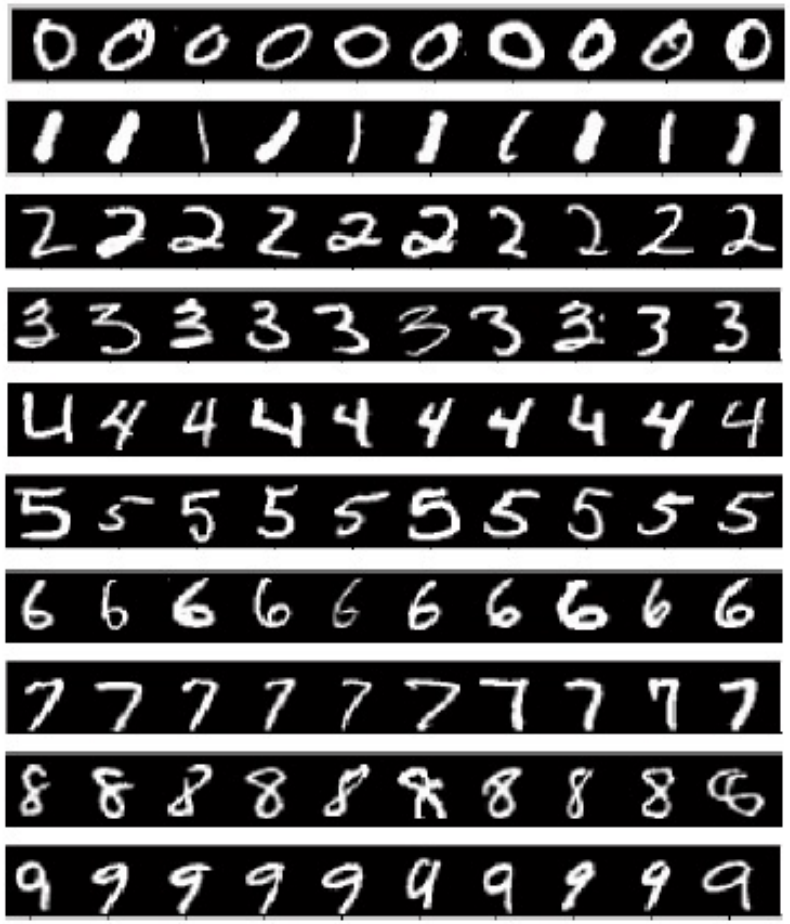


Audun Jøsang. Subjective Logic: A formalism for reasoning under uncertainty. Springer, 2018.
Trusted Multi-View Classification, In ICLR 2021.

Machine learning 1-2-3

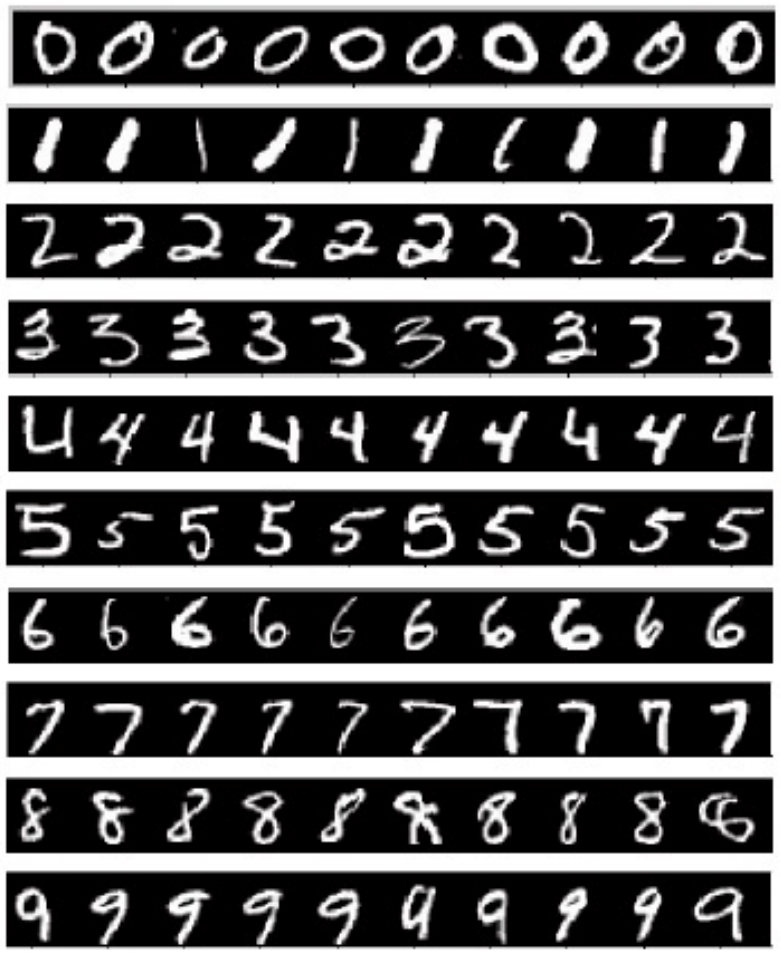
- Collect data and extract features
- Build model: choose hypothesis class $\mathcal{H}$ and loss function l
- Optimization: minimize the empirical loss

Handwritten digits



Problem:
How to recognize the handwritten digits by computer?

Handwritten Digits Recognition



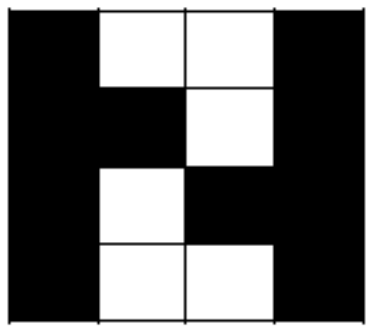
digitizing

representation

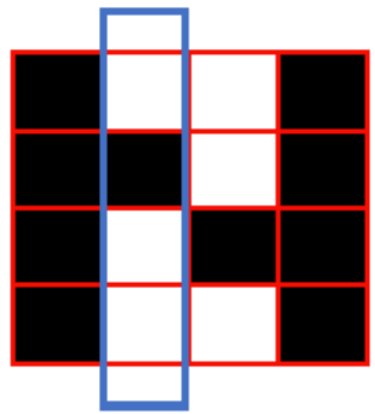
0
1
2
3
4
5
6
7
8
9

Handwritten Digits Recognition

digitizing



4 × 4



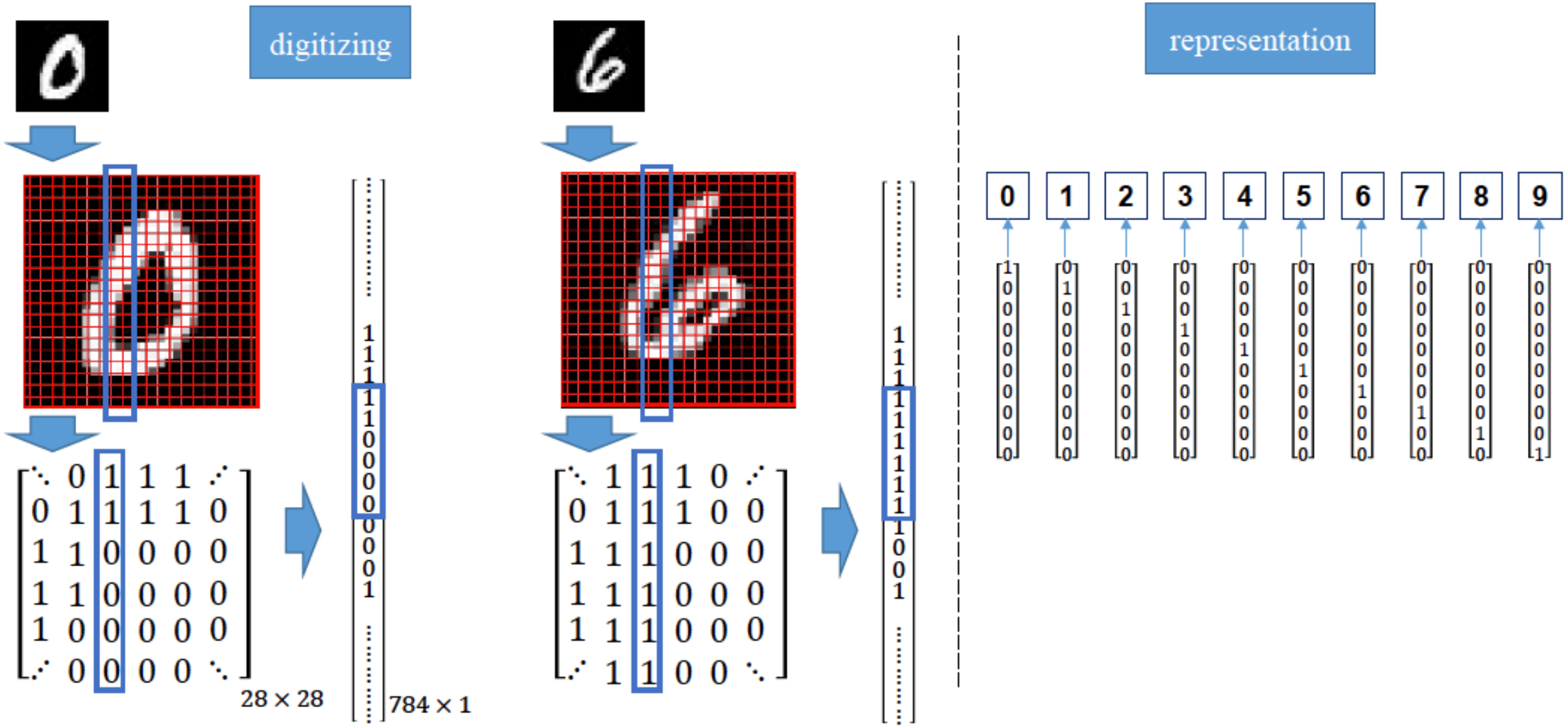
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

4 × 4

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

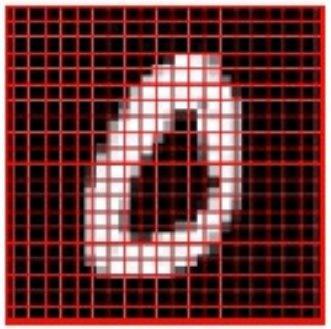
16 × 1

Handwritten Digits Recognition



Handwritten Digits Recognition

pixels



28 × 28

matrix

$$\begin{bmatrix} \ddots & 0 & 1 & 1 & 1 & \iddots \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \iddots & 0 & 0 & 0 & 0 & \ddots \end{bmatrix}$$

28 × 28

vector

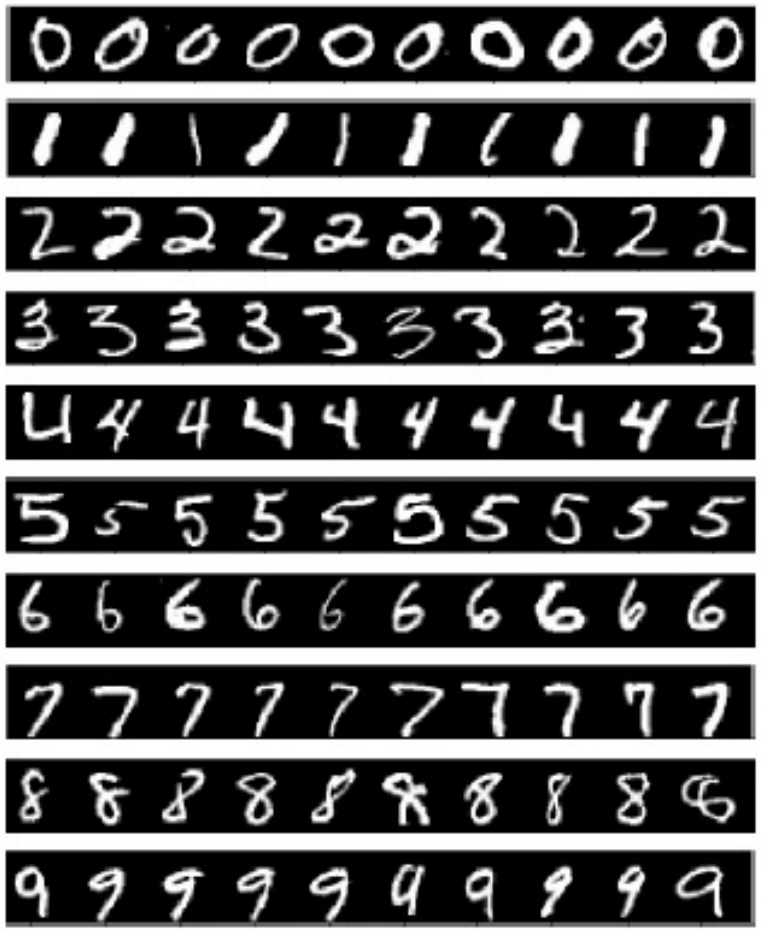
$$x = \begin{bmatrix} \vdots \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

784 × 1

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

0

Handwritten Digits Recognition



$$F: x \rightarrow y$$

x

y

0
1
2
3
4
5
6
7
8
9

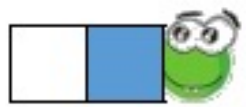
Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$
- Find $y = f(x)$ using training data
- s.t. f correct on test data

What kind of functions?

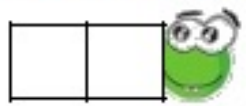
An example: XOR-worms problem

Doted worms



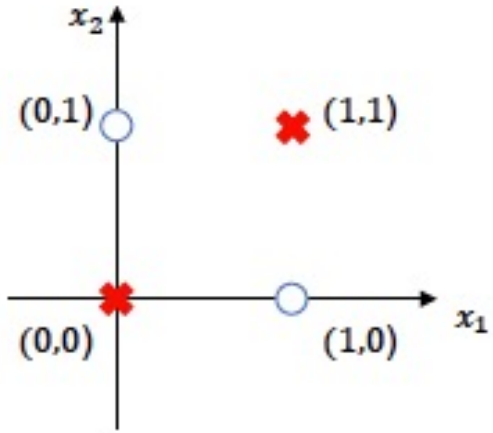
1

Smooth worms



0

An example: XOR-worms problem



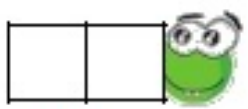
Doted worms

$\begin{bmatrix}0\\1\end{bmatrix}$

$\begin{bmatrix}1\\0\end{bmatrix}$

1

Smooth worms

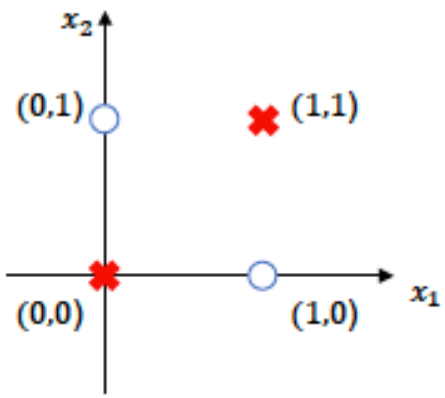


$\begin{bmatrix}0\\0\end{bmatrix}$

$\begin{bmatrix}1\\1\end{bmatrix}$

0

An example: XOR problem



$$\begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1\\0\end{bmatrix} \rightarrow$$

$$\begin{bmatrix}0\\0\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix} \rightarrow$$

x

$F: x \rightarrow y$

$\rightarrow$ 1

$\rightarrow$ 0

y

An example: XOR problem

$\begin{bmatrix}0\\1\end{bmatrix}$ $\begin{bmatrix}1\\0\end{bmatrix}$

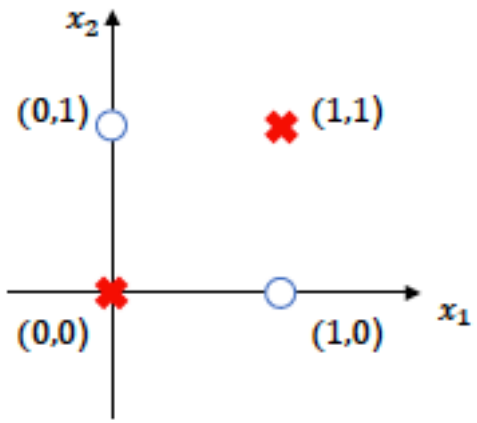
$\begin{bmatrix}0\\0\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$

$$F\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = f[f(2x_1 + 2x_2 - 1) + f(-x_1 - x_2 + 1.5) - 1.5]$$

$$f(s) = \begin{cases}1, & s \geq 0\\ 0, & otherwise\end{cases}$$

→ 1

→ 0



$$F\left(\begin{bmatrix}0\\1\end{bmatrix}\right) = f[f(2\cdot 0 + 2\cdot 1 - 1) + f(-0 - 1 + 1.5) - 1.5] = 1$$

$$F\left(\begin{bmatrix}1\\1\end{bmatrix}\right) = f[f(2\cdot 1 + 2\cdot 1 - 1) + f(-1 - 1 + 1.5) - 1.5] = 0$$

Problem:
How to construct the mapping F?

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. f correct on test data

Hypothesis class

Handwritten Digits Recognition

digitizing

$F: x \rightarrow y$

representation

0
1
2
3
4
5
6
7
8
9

Problem:
How to construct the mapping F?

Handwritten Digits Recognition

$x =$

$$\begin{bmatrix} \vdots \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

28x28

Problem:
How to construct the mapping F from 784 dimensional space to 10 dimensional space?
It is almost impossible manually!

$$F: x \rightarrow y$$

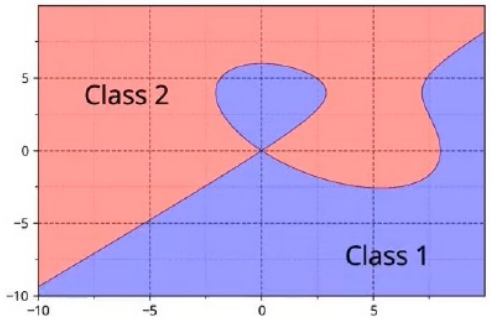
$y =$

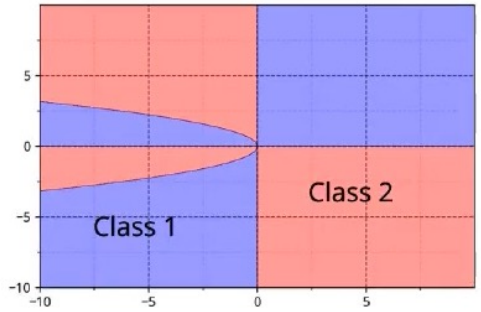
$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

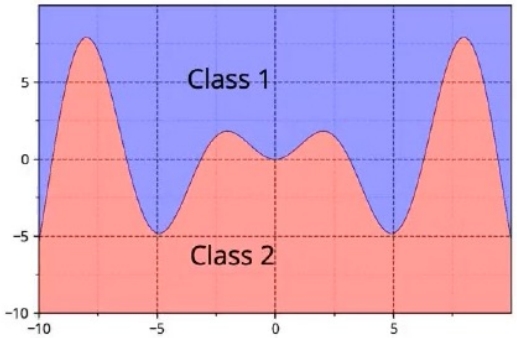
Universal approximation theorem

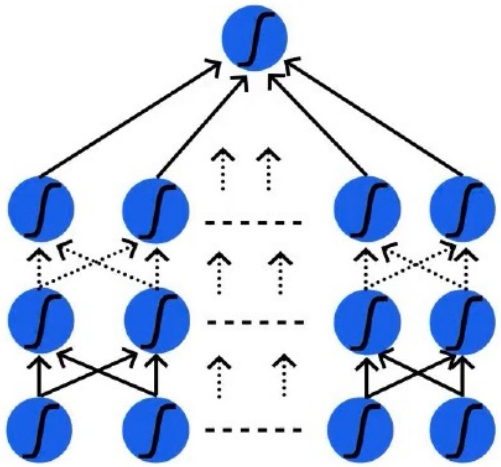
“a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units”

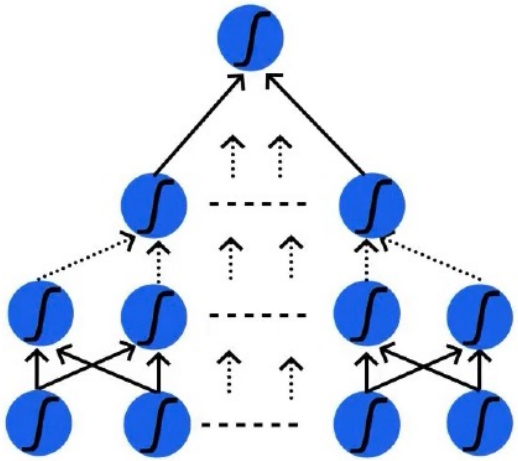
Hornik, 1991

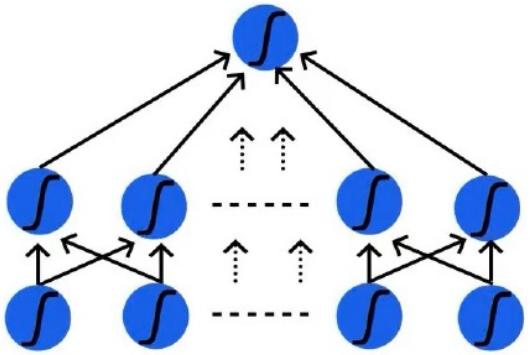




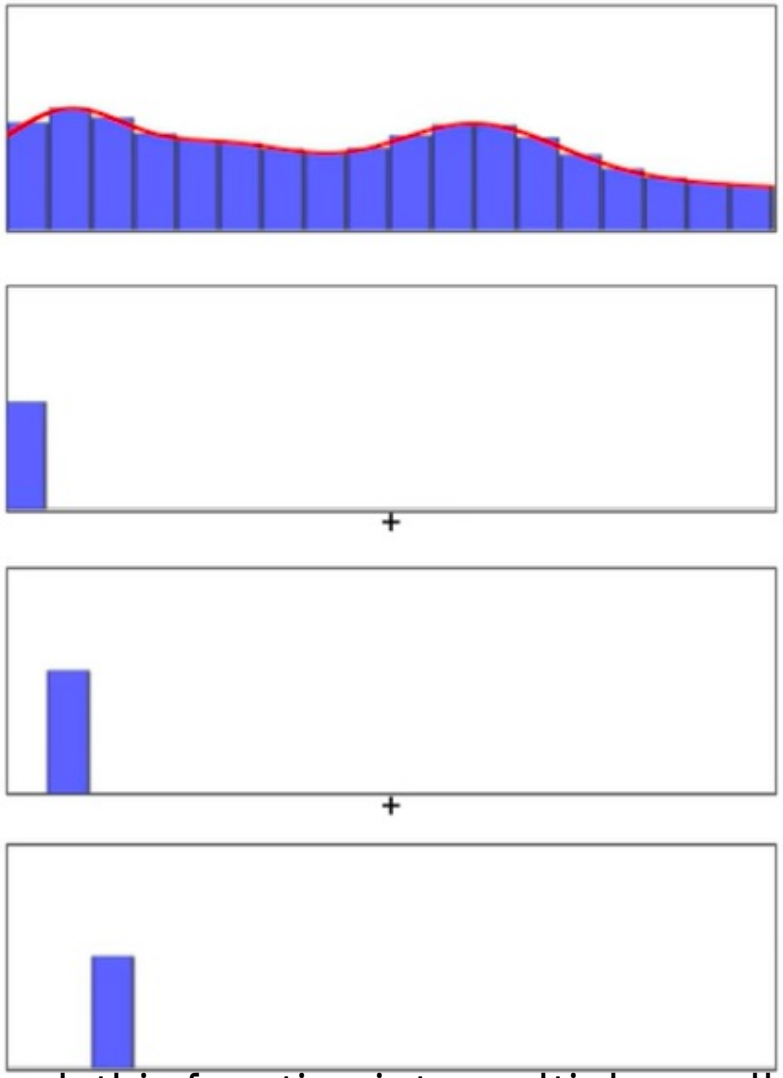


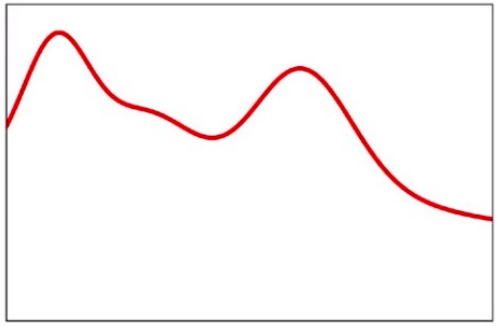


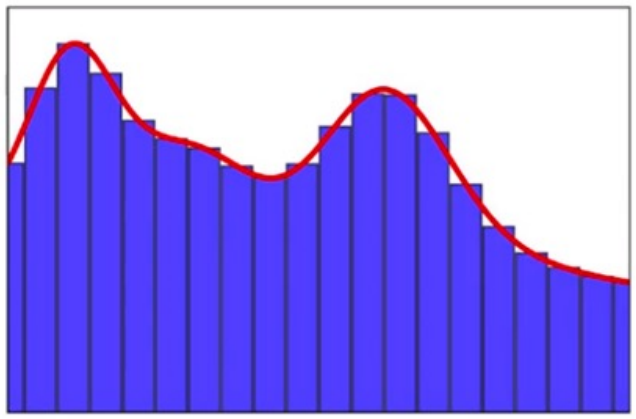


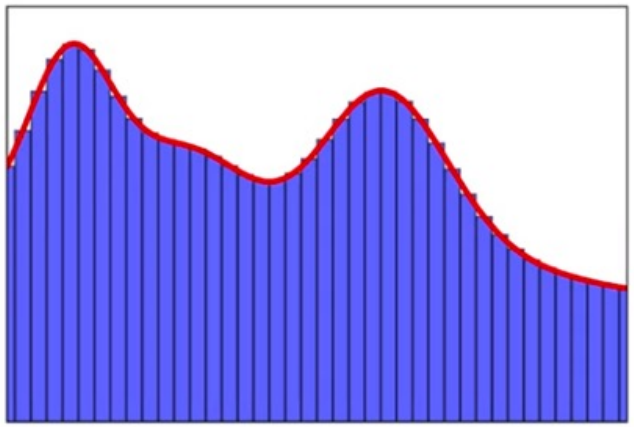


Illustrative Proof of Universal Approximation Theorem







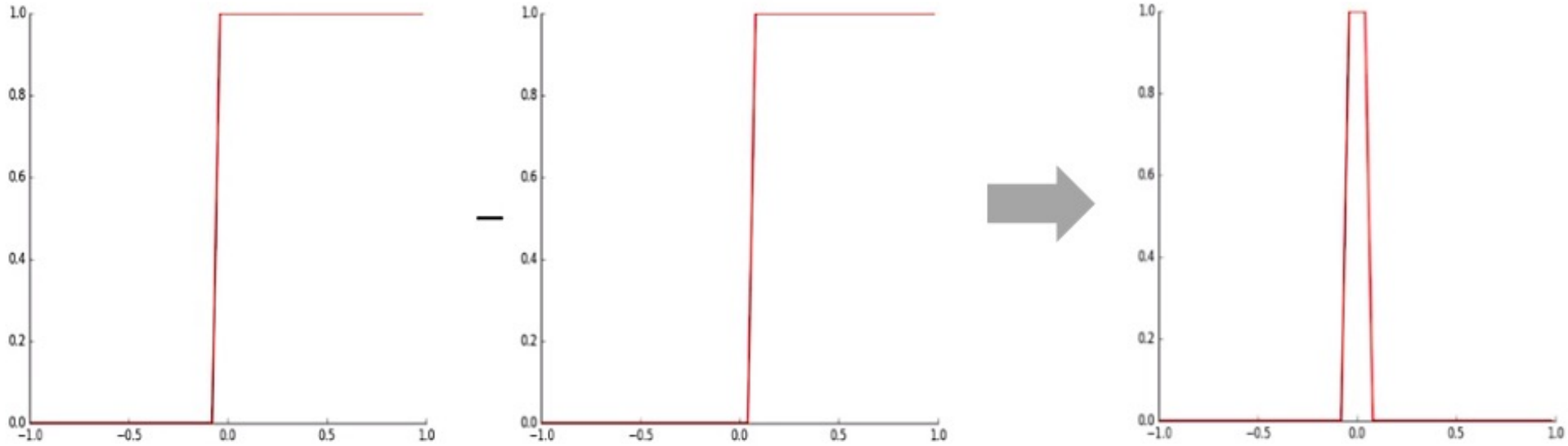


The more functions that I choose in this method the better will be my approximation.

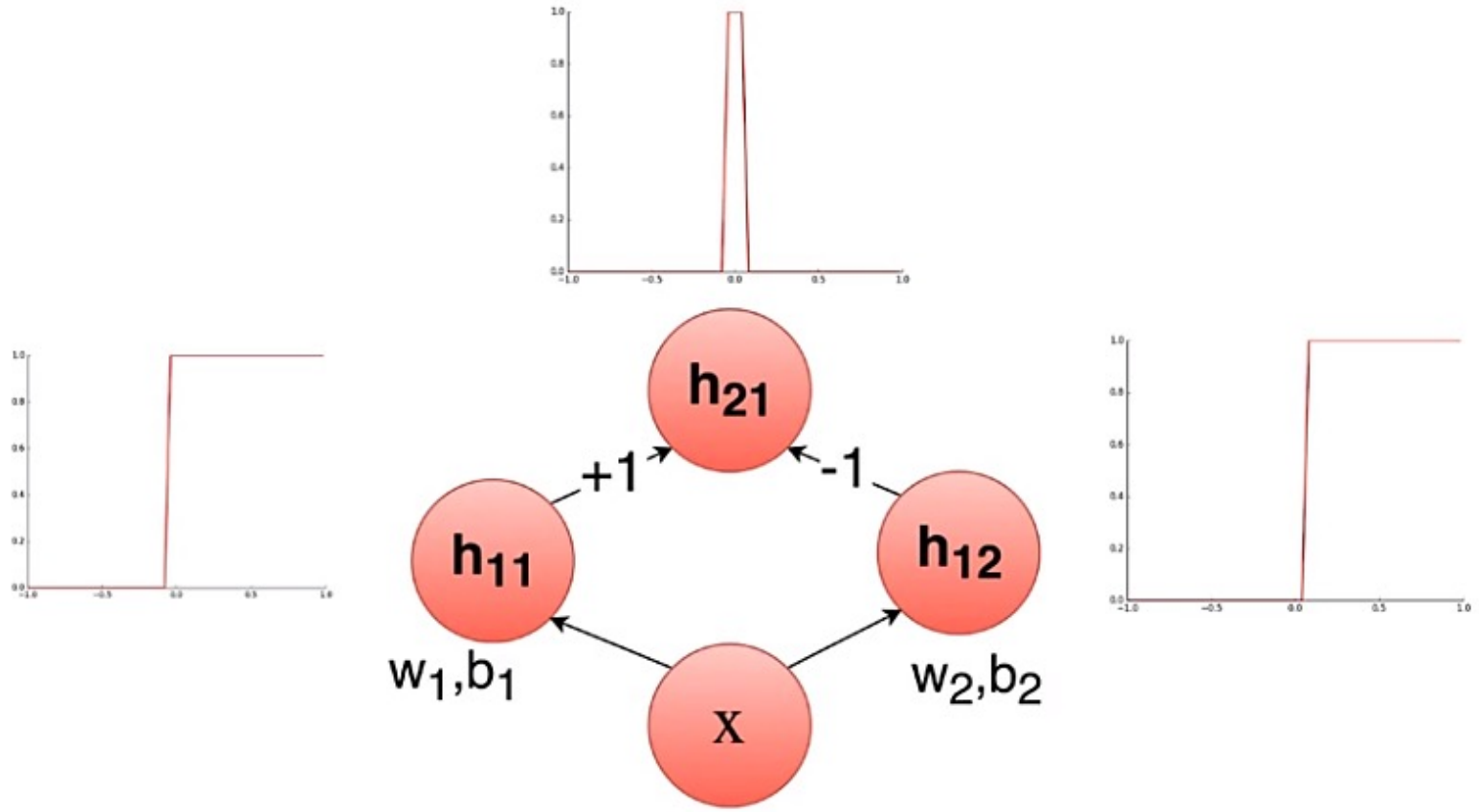
break this function into multiple smaller parts so that each part is represented by a simpler function.

How do we come up with these rectangles/towers and how it will tie back to the sigmoid neuron?

Let’s take two sigmoid functions having a very steep slope and notice that they have a different place at which they peak. The left sigmoid peaks just before zero and right sigmoid peaks just after zero. If I just subtract these two functions the net effect is going to be a tower (rectangular output).



If we can get the series of these towers, then we can approximate any true function between input and output.



if we have an input **x** and it is passed through the two sigmoid neurons and the output from these two neurons are combined in another neuron with weights +1 and -1 i.e... same is as subtracting these two outputs, then we will get our tower. Now you can see that we have our building block ready which is a connection of three sigmoid neurons. If we can construct many such building blocks and add all of them up, we can approximate any complex true relationship between input and output. This is called the ***representational power of deep neural networks***, to approximate any kind of relationship between input and output.

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. f correct on test data

Connection between training data and test data?

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. f correct on test data i.i.d. from distribution D

They have the same distribution

i.i.d.: independently identically distributed

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$ i.i.d. from distribution D
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. f correct on test data i.i.d. from distribution D

What kind of performance measure?

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$ i.i.d. from distribution D
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Various loss functions

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

- Examples of loss functions:
  - 0-1 loss: $l(f, x, y) = \mathbb{I}[f(x) \neq y]$ and $L(f) = \Pr[f(x) \neq y]$
  - l_2 loss: $l(f, x, y) = [f(x) - y]^2$ and $L(f) = \mathbb{E}[f(x) - y]^2$

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$ i.i.d. from distribution D
- Find $y = f(x) \in \mathcal{H}$ using training data
- s.t. the expected loss is small

$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

How to use?

Math formulation

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $y = f(x) \in \mathcal{H}$ that minimizes $\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} l(f, x_i, y_i)$
- s.t. the expected loss is small

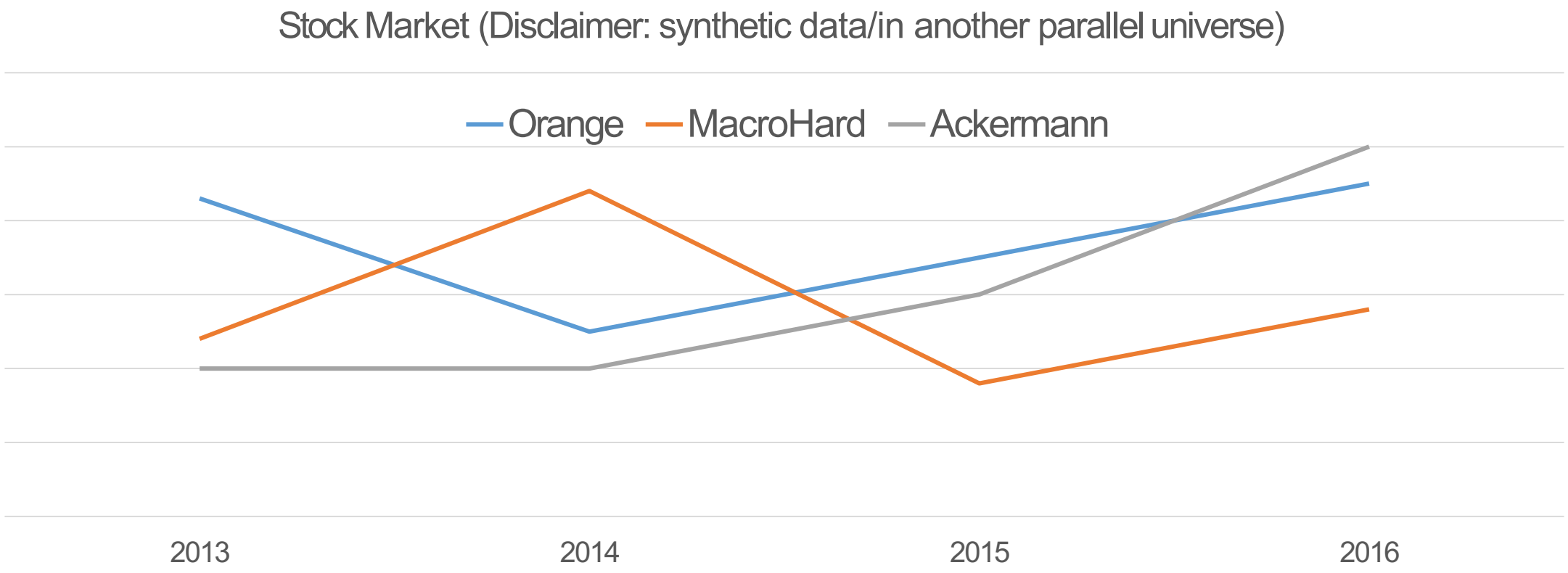
$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Empirical loss

Wait…

- Does Machine Learning-1-2-3 include all approaches?
  - Include many but not all
  - Our current focus will be MachineLearning-1-2-3

Example: Stock Market Prediction



Sliding window over time: serve as input x; non-i.i.d.

Linear regression

Linear regression

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $f_w(x) = w^T x$ that minimizes $\hat{L}(f_w) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$

l_2 loss; also called mean square error

Hypothesis class $\mathcal{H}$

Linear regression: optimization

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $f_w(x) = w^T x$ that minimizes $\hat{L}(f_w) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$

- Let X be a matrix whose i-th row is x_i^T, y be the vector $(y_1, \ldots, y_n)^T$

$$\hat{L}(f_w) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 = \frac{1}{n}\|Xw - y\|_2^2$$

Linear regression: optimization

- Set the gradient to 0 to get the minimizer

$$\nabla_w L(f_w) = \nabla_w \frac{1}{n} \|Xw - y\|_2^2 = 0$$

$$\nabla_w [(Xw - y)^T (Xw - y)] = 0$$

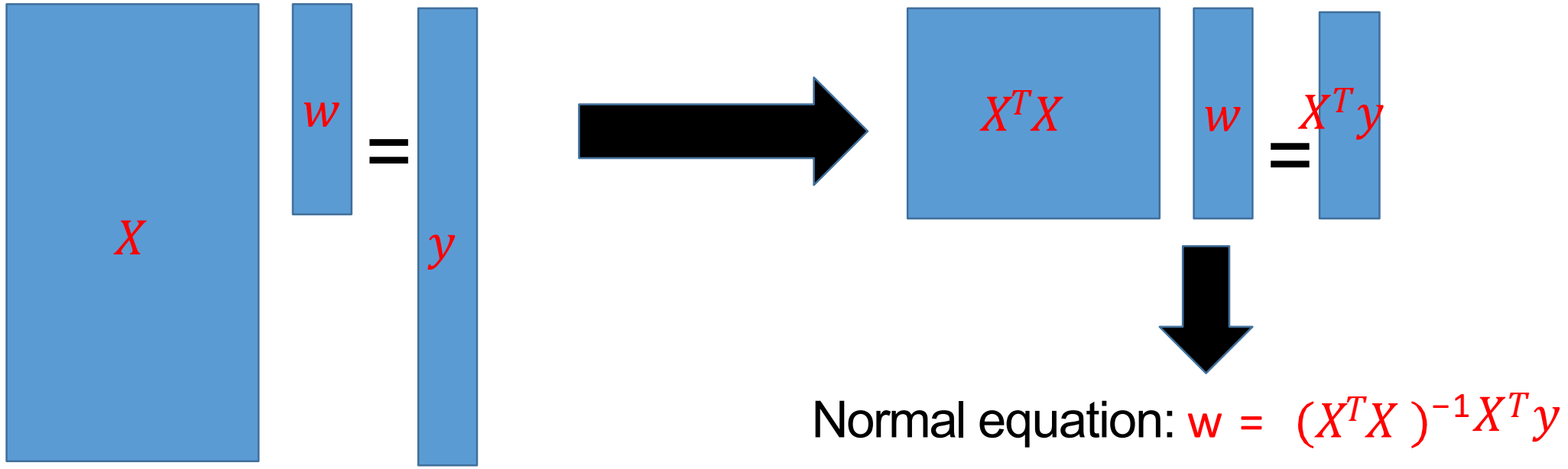
$$\nabla_w [w^T X^T X w - 2 w^T X^T y + y^T y] = 0$$

$$2X^T X w - 2X^T y = 0$$

$$w = (X^T X)^{-1} X^T y$$

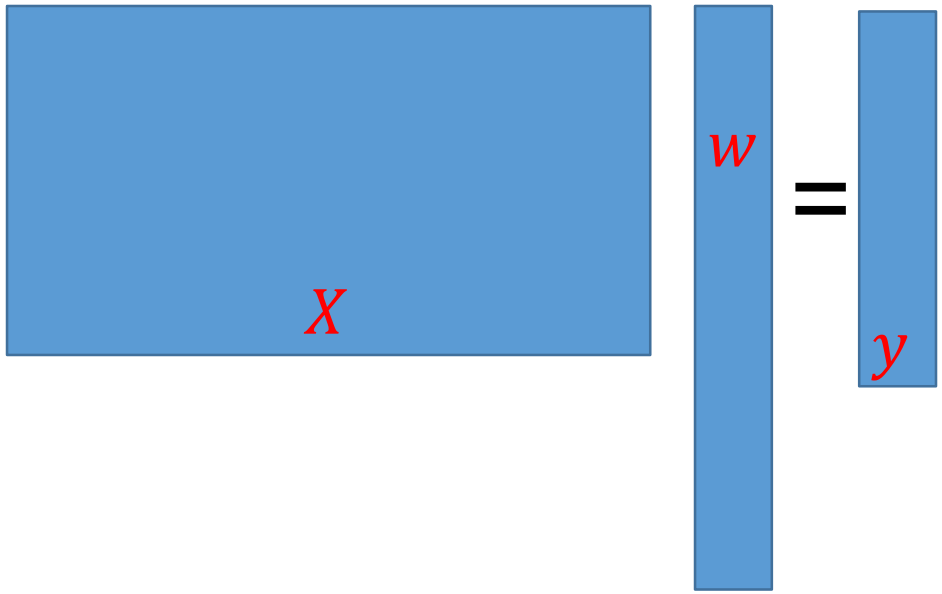
Linear regression: optimization

- Algebraic view of the minimizer
  - If X is invertible, just solve $Xw = y$ and get $w = X^{-1}y$
  - But typically X is a tall matrix



Normal equation: w = $(X^T X)^{-1} X^T y$

How about under-determined system?



(iii) Under-determined system: $m < d$ in $\mathbf{Xw} = \mathbf{y}$, $\mathbf{X} \in R^{m \times d}$

(i.e., there are more unknowns than equations => infinite number of solutions)

If the **right-inverse** of $\mathbf{X}$ exists such that $\mathbf{X}\mathbf{X}^{\dagger} = \mathbf{I}$, then the d-vector $\mathbf{w} = \mathbf{X}^{\dagger}\mathbf{y}$ (one of the infinite cases) satisfies the equation $\mathbf{Xw} = \mathbf{y}$, i.e.,

$$\mathbf{X}\mathbf{X}^{\dagger}\mathbf{y} = \mathbf{y}$$
$$\Rightarrow \quad \mathbf{y} = \mathbf{y}$$

Definition: a matrix **B** that satisfies$\mathbf{AB} = \mathbf{I}$ (identity matrix) is called a **right-inverse** of **A**. (Note: **A** is *m-by-d* and **B** is *d-by-m*).

Note: The right-inverse can be computed as $\mathbf{X}^{\dagger} = \mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T})^{-1}$given $\mathbf{X}\mathbf{X}^{T}$ is invertible.

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (Chp11.1-11.2, 11.5)

Derivation:

Under-determined system: $m < d$ in $\mathbf{Xw} = \mathbf{y}$, $\mathbf{X} \in R^{m \times d}$

(i.e., there are more unknowns than equations => infinite number of solutions => but a unique solution is yet possible by constraining the search using $\mathbf{w} = \mathbf{X}^T\mathbf{a}$!)

If $\mathbf{X}\mathbf{X}^T$ is invertible, let $\mathbf{w} = \mathbf{X}^T\mathbf{a}$, then

$$\mathbf{X}\mathbf{X}^T\mathbf{a} = \mathbf{y}$$

$$\Rightarrow \quad \hat{\mathbf{a}} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$$

$$\Rightarrow \hat{\mathbf{w}} = \mathbf{X}^T\hat{\mathbf{a}} = \underbrace{\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}}_{\mathbf{X}^\dagger}\mathbf{y}$$

right-inverse

Linear regression with bias

Bias term

- Given training data $\{(x_i, y_i): 1 \leq i \leq n \}$ i.i.d. from distribution D
- Find $f_{w,b}(x) = w^T x + b$ to minimize the loss

- Reduce to the case without bias:
  - Let $w' = [w; b], x' = [x; 1]$
  - Then $f_{w,b}(x) = w^T x + b = (w')^T (x')$

Why l_2 loss

- Why not choose another loss
  - l_1 loss, hinge loss, exponential loss, …
- Empirical: easy to optimize
  - For linear case: $\mathrm{w} = (X^T X)^{-1} X^T y$
- Theoretical: a way to encode prior knowledge

Questions:

- What kind of prior knowledge?
- Principal way to derive loss?

Maximum likelihood Estimation

Maximum likelihood Estimation (MLE)

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(x, y): \theta \in \Theta\}$ be a family of distributions indexed by θ

- Would like to pick θ so that $P_\theta(x, y)$ fits the data well

Maximum likelihood Estimation (MLE)

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(x, y): \theta \in \Theta\}$ be a family of distributions indexed by θ

- "fitness" of θ to one data point (x_i, y_i)

$$\text{likelihood}(\theta; x_i, y_i) := P_\theta(x_i, y_i)$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(x, y): \theta \in \Theta\}$ be a family of distributions indexed by θ

- "fitness" of θ to i.i.d. data points $\{(x_i, y_i)\}$

$$\text{likelihood}(\theta; \{x_i, y_i\}) \coloneqq P_\theta(\{x_i, y_i\}) = \prod_i P_\theta(x_i, y_i)$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(x, y): \theta \in \Theta\}$ be a family of distributions indexed by θ

- MLE: maximize "fitness" of θ to i.i.d. data points $\{(x_i, y_i)\}$

$$\theta_{ML} = \text{argmax}_{\theta \in \Theta} \log[\prod_i P_\theta(x_i, y_i)]$$

$$\theta_{ML} = \text{argmax}_{\theta \in \Theta} \sum_i \log[P_\theta(x_i, y_i)]$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(x, y): \theta \in \Theta\}$ be a family of distributions indexed by θ

- MLE: negative log-likelihood loss

$$\theta_{ML} = \text{argmax}_{\theta \in \Theta} \sum_i \log(P_\theta(x_i, y_i))$$

$$l(P_\theta, x_i, y_i) = -\log(P_\theta(x_i, y_i))$$

$$\hat{L}(P_\theta) = -\sum_i \log(P_\theta(x_i, y_i))$$

MLE: conditional log-likelihood

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(y|x): \theta \in \Theta\}$ be a family of distributions indexed by θ

Only care about predicting y from x; do not care about p(x)

- MLE: negative conditional log-likelihood loss

$$\theta_{ML} = \text{argmax}_{\theta \in \Theta} \sum_i \log(P_\theta(y_i|x_i))$$

$$l(P_\theta, x_i, y_i) = -\log(P_\theta(y_i|x_i))$$

$$\hat{L}(P_\theta) = -\sum_i \log(P_\theta(y_i|x_i))$$

MLE: conditional log-likelihood

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Let $\{P_\theta(y|x): \theta \in \Theta\}$ be a family of distributions indexed by θ

P(y|x): discriminative;
P(x,y): generative

- MLE: negative conditional log-likelihood loss

$$\theta_{ML} = \text{argmax}_{\theta \in \Theta} \sum_i \log(P_\theta(y_i|x_i))$$

$$l(P_\theta, x_i, y_i) = -\log(P_\theta(y_i|x_i))$$

$$\hat{L}(P_\theta) = -\sum_i \log(P_\theta(y_i|x_i))$$

Example: l_2 loss

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $f_\theta(x)$ that minimizes $\hat{L}(f_\theta) = \frac{1}{n}\sum_{i=1}^{n}(f_\theta(x_i) - y_i)^2$

Example: l_2 loss

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $f_\theta(x)$ that minimizes $\hat{L}(f_\theta) = \frac{1}{n}\sum_{i=1}^{n}(f_\theta(x_i) - y_i)^2$

l_2 loss: Normal + MLE

- Define $P_\theta(y|x) = \text{Normal}(y; f_\theta(x), \sigma^2)$
- $\log(P_\theta(y_i|x_i)) = \frac{-1}{2\sigma^2}(f_\theta(x_i) - y_i)^2 - \log(\sigma) - \frac{1}{2}\log(2\pi)$
- $\theta_{ML} = \text{argmin}_{\theta\in\Theta}\frac{1}{n}\sum_{i=1}^{n}(f_\theta(x_i) - y_i)^2$

$$\Pr(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^{\mathsf{T}}\boldsymbol{w})^2\right\}$$