Machine Learning: What is?

A First Step to Practical Machine Learning

Sirakorn Lamyai

Student, Kasetsart U.

October 30, 2018

Acknowledgements

Resources on this slides are mainly adapted and rearranged from

- W. Jitkrittum: Machine Learning Fundamentals I
- K. Muandet: Machine Learning Fundamentals II
- Google's Machine Learning Crash Course

Before we start...

Make sure these are installed on your computer.

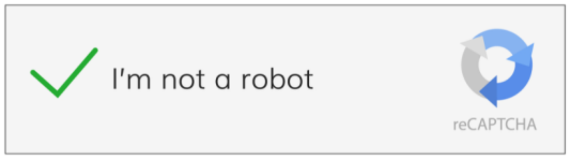
- Python 3.6
- NumPy, Scipy, Matplotlib, Scikit-learn, MLxtend:
  Run `pip install numpy scipy matplotlib sklearn mlxtend`
- MNIST loader
  `pip install git+https://github.com/datapythonista/mnist.git`

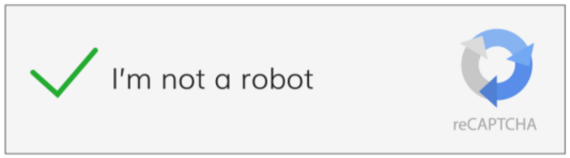
Outline

Introduction to Machine Learning

What is Machine Learning?

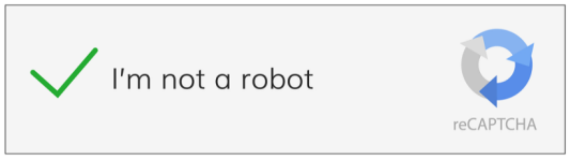
What is Machine Learning?



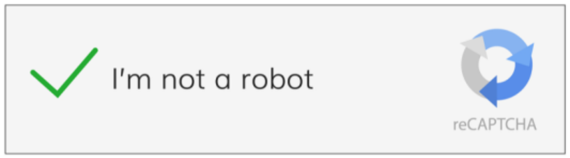


- This is Recaptcha.

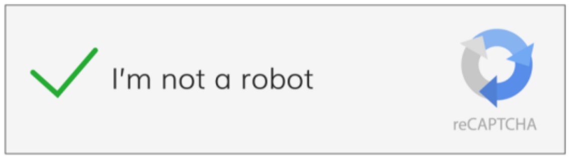
What is Machine Learning?



- This is Recaptcha.
  - Recaptcha helps stop millions of spam a day.



- This is Recaptcha.
  - Recaptcha helps stop millions of spam a day.
  - In some old days, we have to type Captcha texts to distinguish ourself from bots.



- This is Recaptcha.
  - Recaptcha helps stop millions of spam a day.
  - In some old days, we have to type Captcha texts to distinguish ourself from bots.
  - How is it possible that with a single click, an automated system can distinguish bots from humans?

Machine **Learning**

Machine **Learning**

= Improves performance on a specific task.

Machine Learning Problems

Types of Machine Learning problems

Types of Machine Learning problems

1. Supervised learning

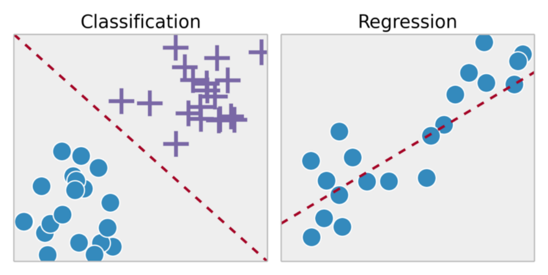
Types of Machine Learning problems

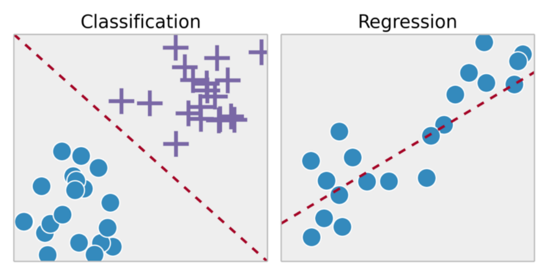
1. Supervised learning
2. Unsupervised learning

Types of Machine Learning problems

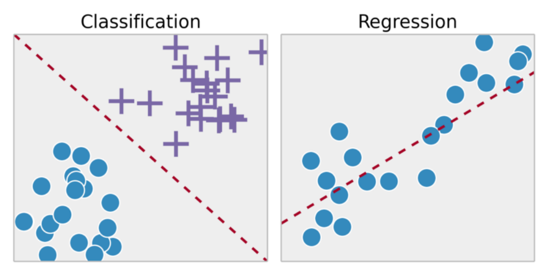
1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

Supervised learning

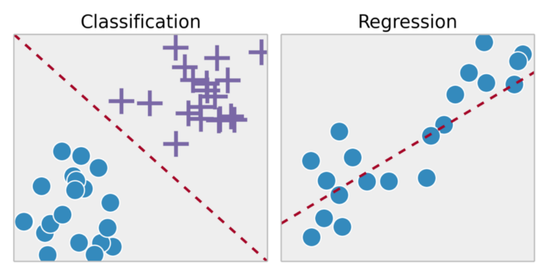




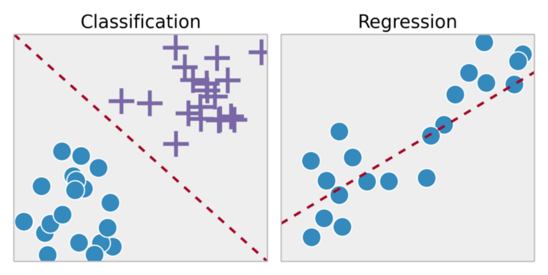
- Given a **training set** for the data, find a **model** to **generalise** well to **unseen** data.



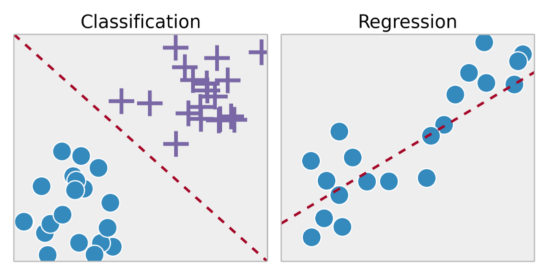
- Given a **training set** for the data, find a **model** to **generalise** well to **unseen** data.
- Two main supervised learning problems



- Given a **training set** for the data, find a **model** to **generalise** well to **unseen** data.
- Two main supervised learning problems
  - Classification: On the discrete data

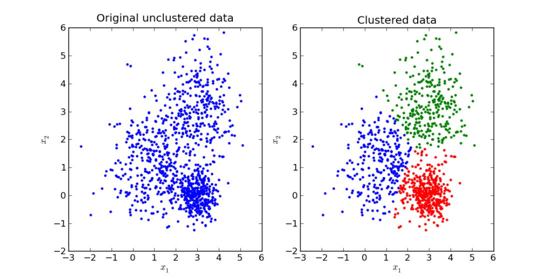


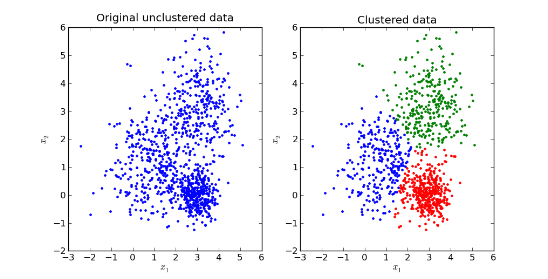
- Given a **training set** for the data, find a **model** to **generalise** well to **unseen** data.
- Two main supervised learning problems
  - Classification: On the discrete data
  - Regression: On the continuous data



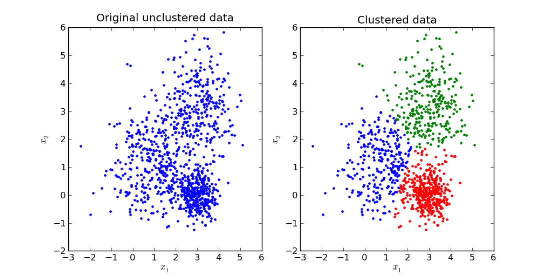
- Given a **training set** for the data, find a **model** to **generalise** well to **unseen** data.
- Two main supervised learning problems
  - Classification: On the discrete data
  - Regression: On the continuous data
- Example problems: Spam E-mail detection, Facial recognition

Unsupervised learning



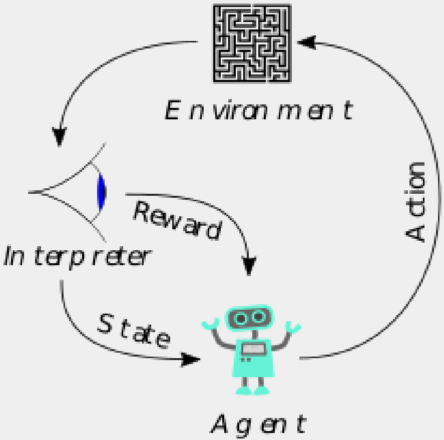


- Discover **hidden** structure in **non-labelled** data.



- Discover **hidden** structure in **non-labelled** data.
- Example: Clustering, Generative models

Reinforcement learning



Model

- A result of the combination between...

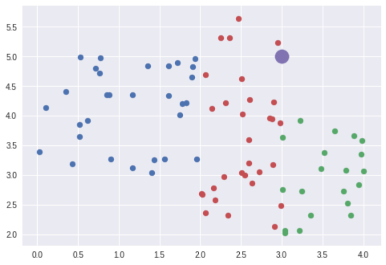
- A result of the combination between...
  - a **method** to recognise the data, and

Model

- A result of the combination between...
  - a **method** to recognise the data, and
  - **sample datas** for such the method

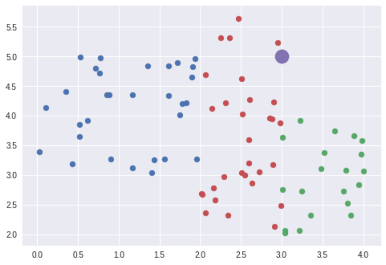
Model

- A result of the combination between...
  - a **method** to recognise the data, and
  - **sample datas** for such the method



Data

- A result of the combination between...
  - a **method** to recognise the data, and
  - **sample datas** for such the method



Determine which group should the purple dot be in (red/green/blue) by **checking the colour of its nearest dot.**

Data

Method

The Goal of Machine Learning

Generalise

Generalise

- We wants our model to know how does the **data pattern** looks like

Generalise

- We wants our model to know how does the **data pattern** looks like
- Our model should not "adhere" to one set of data, but instead knows the pattern of all the data.

Generalise

- We wants our model to know how does the **data pattern** looks like
- Our model should not "adhere" to one set of data, but instead knows the pattern of all the data.
- Therefore, we want our model to **generalise** over any set of data, given the small portion of data that we used to teach the model.

Beginning with our first model

- We're going to write our **first own** machine learning algorithm called ***k*-Nearest Neighbour** (k-NN)

- We're going to write our **first own** machine learning algorithm called ***k*-Nearest Neighbour** (*k*-NN)
  - *k*-NN is known to be very simple, with its concept as

- We're going to write our **first own** machine learning algorithm called ***k*-Nearest Neighbour** (*k*-NN)
  - *k*-NN is known to be very simple, with its concept as

_k_-NN algorithm

To classify label of a data point, get *k* nearest data points to the data point, and select the major label among those data points.

Machine Learning Process

Choosing the parameter for k-NN algorithm

What is the bad way to choose k?

Choosing the parameter for k-NN algorithm

What is the bad way to choose k?

- What if we choose k = # of all points?

Choosing the parameter for k-NN algorithm

What is the bad way to choose k?

- What if we choose k = # of all points?
  - What will happen if our dataset's got 3 labels of A, B, C with 10, 20, and 30 data points of each?

What is the bad way to choose k?

- What if we choose k = # of all points?
  - What will happen if our dataset's got 3 labels of A, B, C with 10, 20, and 30 data points of each?
  - Answer: Our model will always answer the labels with the highest data point count.

Choosing the parameter for k-NN algorithm

What is the bad way to choose k?

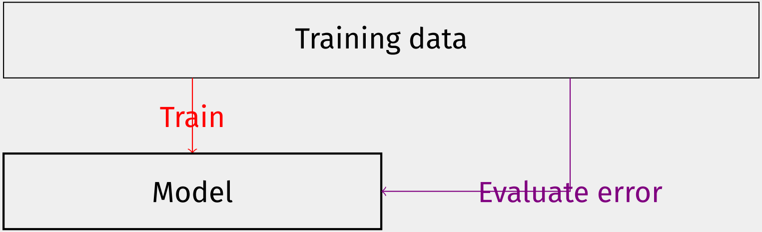
- What if we choose k = # of all points?
  - What will happen if our dataset's got 3 labels of A, B, C with 10, 20, and 30 data points of each?
  - Answer: Our model will always answer the labels with the highest data point count.
- What if we choose $k = 1$?

Choosing the parameter for k-NN algorithm

What is the bad way to choose k?

- What if we choose k = # of all points?
  - What will happen if our dataset's got 3 labels of A, B, C with 10, 20, and 30 data points of each?
  - Answer: Our model will always answer the labels with the highest data point count.
- What if we choose $k = 1$?
  - Let's try!

Training Error



Loss function

$$L(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & y \neq \hat{y} \end{cases}$$

Error

$$\sigma = \frac{1}{N} \sum_{i=0}^{N} L(y_i, \hat{y}_i)$$

What went wrong with our intuition?

What went wrong with our intuition?

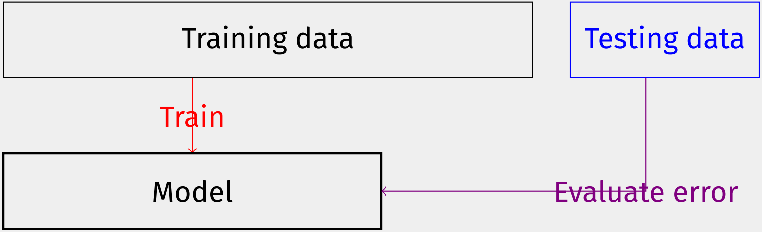
- Evaluating error with a data points that our model had already seen is bad.

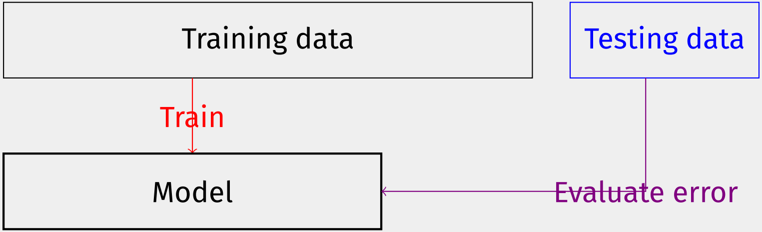
- Evaluating error with a data points that our model had already seen is bad.
- Why? Because our model already knows the answer to that data point! It could just simply answer by looking at the "answer key"

What went wrong with our intuition?

- Evaluating error with a data points that our model had already seen is bad.
- Why? Because our model already knows the answer to that data point! It could just simply answer by looking at the "answer key"
- So how should we evaluate our model?

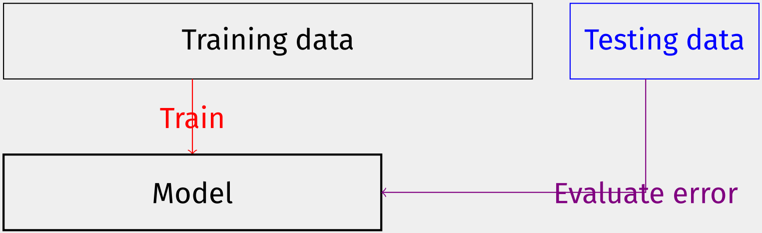
Training and Testing set





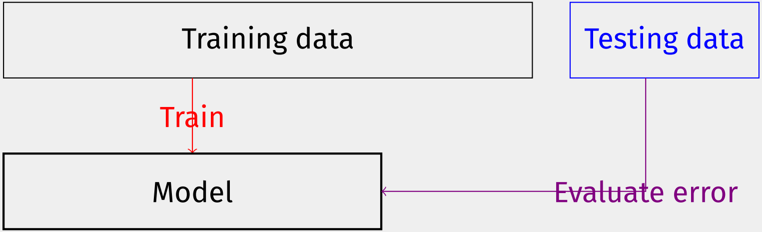
- We separate our dataset into 2 parts: the **training set** and **testing set**

Training and Testing set

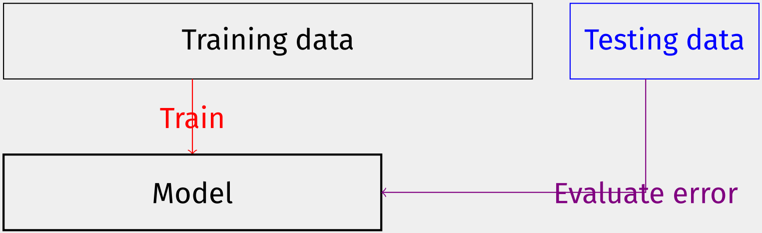


- We separate our dataset into 2 parts: the **training set** and **testing set**
  - Our model sees the correct label of the training set, but not the testing set.

Training and Testing set



- We separate our dataset into 2 parts: the **training set** and **testing set**
  - Our model sees the correct label of the training set, but not the testing set.
  - We all know the correct label of both the training and testing set.

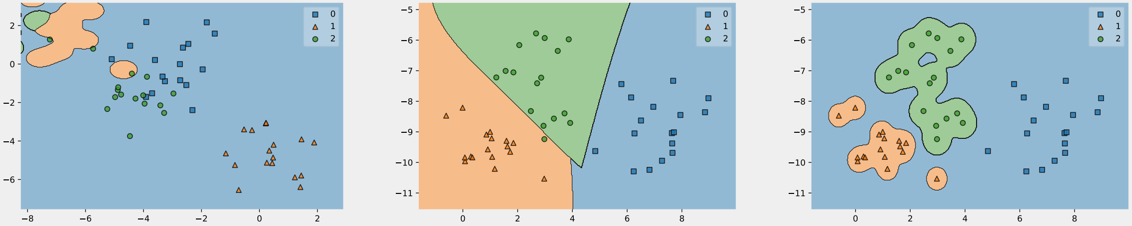


- We separate our dataset into 2 parts: the **training set** and **testing set**
  - Our model sees the correct label of the training set, but not the testing set.
  - We all know the correct label of both the training and testing set.
  - Teach our model with the training set, see how it performs with the testing set.

What will happen if...

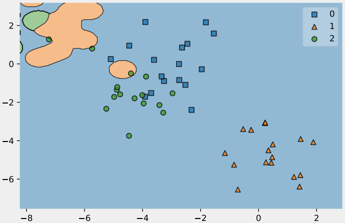
- our k is too small?
- our k is too large?

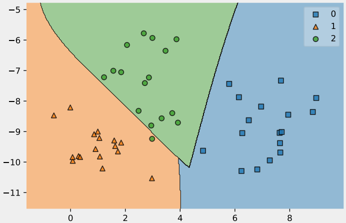
Which decision region is good?

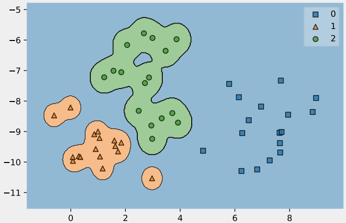


OVERFITTING AND UNDERFITTING

Which decision region is good?

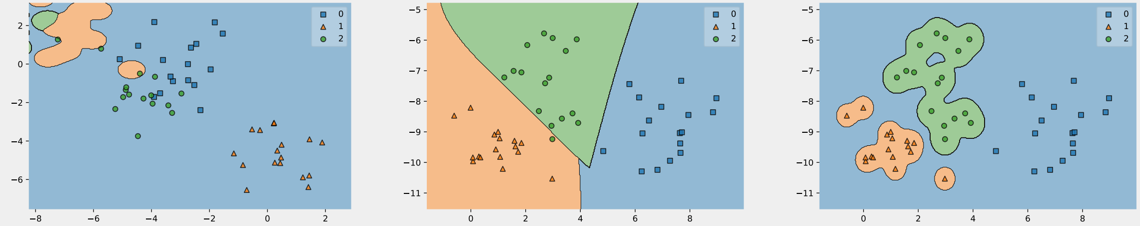






Underfit: The model fails to recognise data pattern

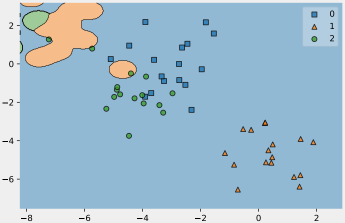
Which decision region is good?



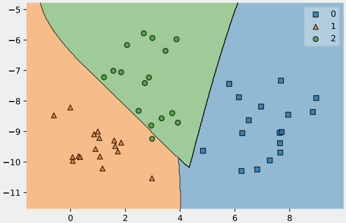
Underfit: The model fails to recognise data pattern

Overfit: The model **remembers** data pattern instead of generalising.

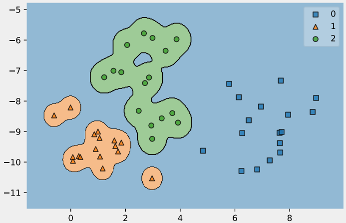
Which decision region is good?



Underfit: The model fails to recognise data pattern



Good fit: The model recognises data pattern **generally**



Overfit: The model **remembers** data pattern instead of generalising.

Good model must **generalise**

It's not only k that we can adjust.

It's not only *k* that we can adjust.

- Actually, the key point in *k*-NN algorithm is choosing *k* points with the least **distant**.

Parameter Optimisation

It's not only k that we can adjust.

- Actually, the key point in k-NN algorithm is choosing k points with the least **distant**.
- What is **distant**?

Norm for k-NN algorithm

Norm for k-NN algorithm

Norm

In linear algebra, a **norm** is a function that assigns a strictly positive length or size to each vector in a vector space - except for the zero vector, which is assigned a length of zero.

Norm

In linear algebra, a **norm** is a function that assigns a strictly positive length or size to each vector in a vector space - except for the zero vector, which is assigned a length of zero.

Given $\vec{x}$ as an N-dimension vector of $\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$

Norm

In linear algebra, a **norm** is a function that assigns a strictly positive length or size to each vector in a vector space - except for the zero vector, which is assigned a length of zero.

Given $\vec{x}$ as an N-dimension vector of $\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$

- l_1 Norm: $|x|_1 = \sum_{i=0}^{N} |x_i|$ (Manhattan)

Norm for k-NN algorithm

Norm

In linear algebra, a **norm** is a function that assigns a strictly positive length or size to each vector in a vector space - except for the zero vector, which is assigned a length of zero.

Given $\vec{x}$ as an N-dimension vector of $\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$

- l_1 Norm: $|x|_1 = \sum_{i=0}^{N} |x_i|$ (Manhattan)
- l_2 Norm: $|x|_2 = \sqrt{\sum_{i=0}^{N} x_i^2}$ (Euclidian)

Norm for k-NN algorithm

Norm

In linear algebra, a **norm** is a function that assigns a strictly positive length or size to each vector in a vector space - except for the zero vector, which is assigned a length of zero.

Given $\vec{x}$ as an N-dimension vector of $\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$

- l_1 Norm: $|x|_1 = \sum_{i=0}^{N} |x_i|$ (Manhattan)
- l_2 Norm: $|x|_2 = \sqrt{\sum_{i=0}^{N} x_i^2}$ (Euclidian)
- l_p Norm: $|x|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$ (Minkowski)

Algorithms for Machine Learning Classification Problem

k-NN is a very simple intuition for machine learning algorithms. However, there exists more algorithm that performs well to other problems.

k-NN is a very simple intuition for machine learning algorithms. However, there exists more algorithm that performs well to other problems.
Example algorithms:

k-NN is a very simple intuition for machine learning algorithms. However, there exists more algorithm that performs well to other problems.
Example algorithms:

- Naïve Bayes

Algorithms for Machine Learning Classification Problem

k-NN is a very simple intuition for machine learning algorithms. However, there exists more algorithm that performs well to other problems.
Example algorithms:

- Naïve Bayes
- SVM

k-NN is a very simple intuition for machine learning algorithms. However, there exists more algorithm that performs well to other problems.
Example algorithms:

- Naïve Bayes
- SVM
- Decision Tree

Algorithms for Machine Learning Classification Problem

k-NN is a very simple intuition for machine learning algorithms. However, there exists more algorithm that performs well to other problems.
Example algorithms:

- Naïve Bayes
- SVM
- Decision Tree
- Logistic Regression

	Gender	Hair
1	M	Long
2	M	Short
3	F	Long
4	F	Long
5	F	Short

Bayes Theorem

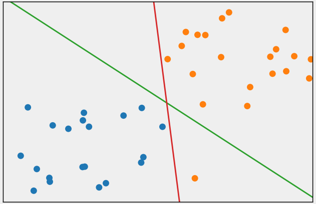
$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A) \times P(A)}{P(B)}$$

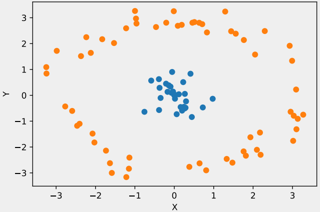
Can we *guess* the gender from hair's length?

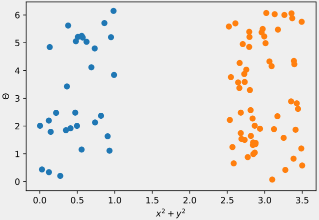
- $P(\text{Male}|\text{Long hair}) = \frac{1}{3}$
- $P(\text{Female}|\text{Long hair}) = \frac{2}{3}$

Therefore, we guess that the long-haired person is more likely to be a female.

Support Vector Machines (SVM)

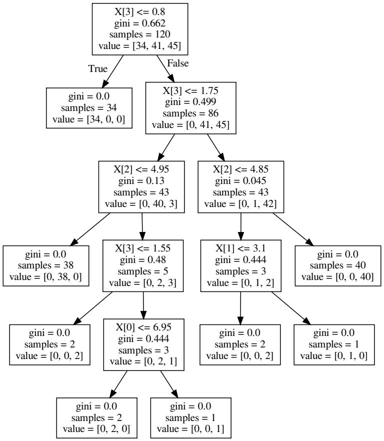






- Goal: to draw a line to separate groups of data
- Ideal good line: maximising the distant between the line and classes of data points
- What if the data is not linearly separable? **Kernel tricks**

Decision Tree



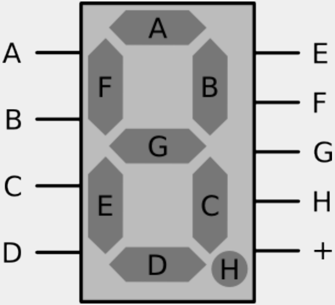
- Creating an if-else conditions automatically
- Nested conditions with a parameter to determine how does the separating of the "tree" performs.

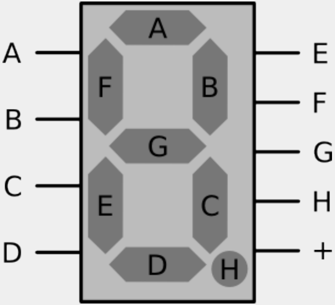
Figure: xkcd - Compiling

Figure: xkcd - Compiling (shamelessly modified)

Problems for Machine Learning

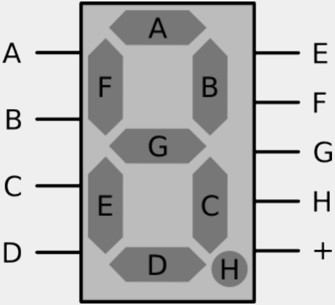
7-Segment display



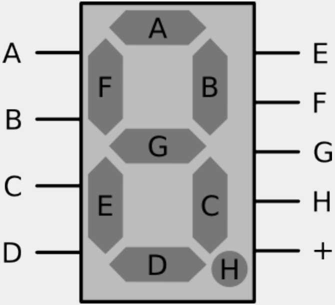


- This is a 7-segment display.

7-Segment display



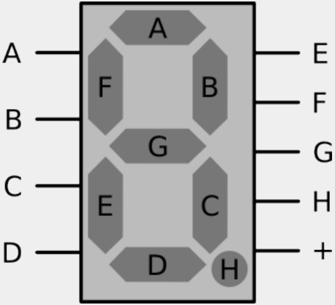
- This is a 7-segment display.
- It consists of a bulb labelled from A-G that could form a number.



- This is a 7-segment display.
- It consists of a bulb labelled from A-G that could form a number.

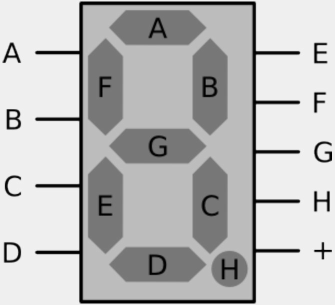
Problem

When the list of the bulb that went on were given, can we determine the number?



Problem

When the list of the bulb that went on were given, can we determine the number?

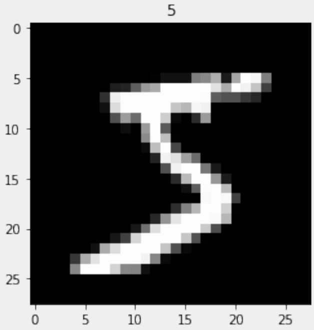


Problem

When the list of the bulb that went on were given, can we determine the number?

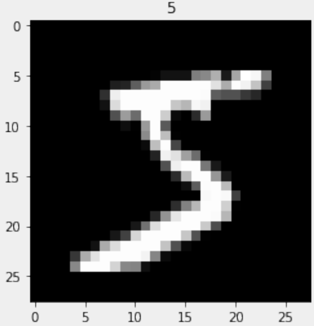
Not only yes, but *easily* yes!

```
if led_on == (b, c):
    return 1
elif led_on == (a, b, g, e, d):
    return 2
...
```



Problem

When the image of the handwriting were given, can we determine the number?



Problem

When the image of the handwriting were given, can we determine the number?

With an **explicit algorithm**? Obviously no! There are too many ways of drawing the number!

MNIST Dataset

MNIST Dataset

- 28*28 pixel images of handwritten numbers (0-9)

- 28*28 pixel images of handwritten numbers (0-9)
  - Later to be viewed as a vector of 784 dimensions

MNIST Dataset

- 28*28 pixel images of handwritten numbers (0-9)
  - Later to be viewed as a vector of 784 dimensions
- 60,000 training images

MNIST Dataset

- 28*28 pixel images of handwritten numbers (0-9)
  - Later to be viewed as a vector of 784 dimensions
- 60,000 training images
- 10,000 testing images

- Training: Pretty fast, no calculations on training phase
- Testing: **thinking**
  - 60,000 data points to calculate the distant + 10,000 data points to test
  - = 600,000,000 calculations to be made
    (this excludes sorting, of which is a $\mathcal{O}(n)$ process)
  - = **(relatively) slow**
- Good results with k-NN were achieved.
  (k-NN w/ non-linear deformation (P2DHMDM), preprocessed with shiftable edges, results into 0.52% error rate.)

- Training: Slow as hell.

  Trust me, I've tried.
- Good results with SVM were achieved.
  (Virtual SVM, deg-9 poly, 2-pixel jittered with deskewing preprocessing results into 0.56% error rate.)

Neural Networks

Artificial Neural Networks (ANN)

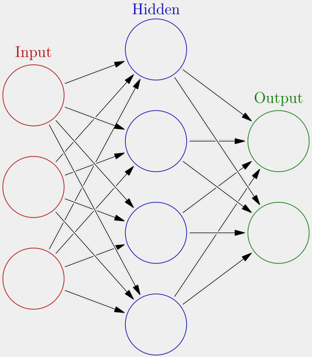


Figure: Neural network (Courtesy: Glosser.ca from Wikimedia Commons)

This seems complex, right? We'll get start a little by little...

Neurons

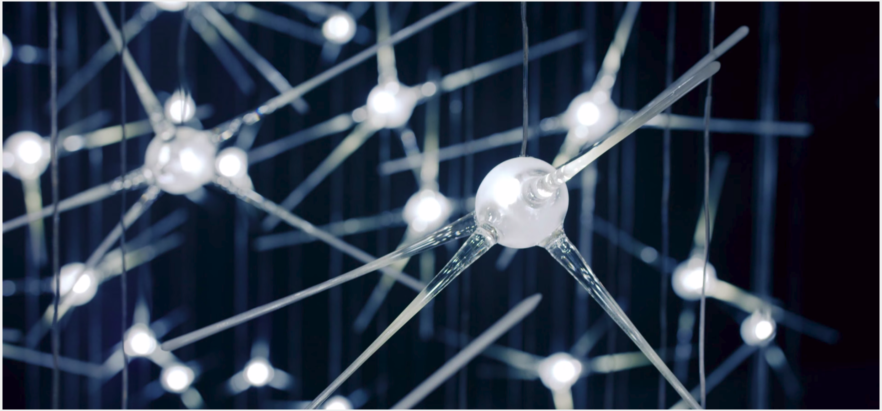


Figure: "Neurons" chandelier installed at Princh Mahidol Hall, Nakhon Pathom, Thailand (Courtesy: LASVIT’s promotion video)

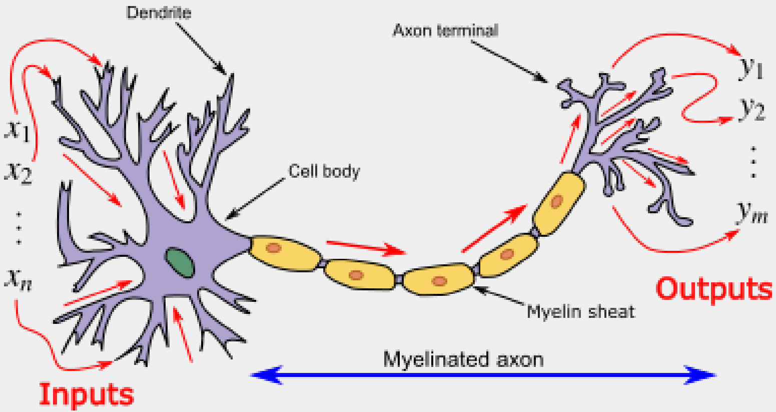
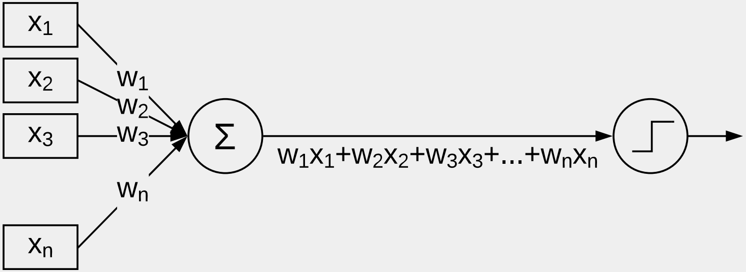
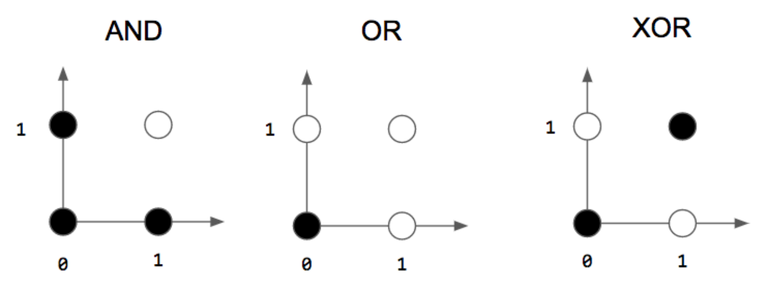


Figure: Neuron (Courtesy: Egm4313.s12 from Wikimedia Commons)



- **Inputs** consisting of *n* inputs from x_1, x_2 ... to x_n.
- **Weights** of each inputs, namely w_1, w_2, ..., w_n
- **Summation** of all the weighted inputs $\Sigma = w_1x_1 + w_2x_2 + \ldots + w_nx_n$
- **Activation function** in either the form of $\Sigma > k$ or $\Sigma < k(nonlinear)$

Logic gates with Perceptron



AND gate

- $w_1 = 1$
- $w_2 = 1$
- $\Sigma = 1x_1 + 1x_2$
- $f : \Sigma \geq 2$

OR gate

- $w_1 = 1$
- $w_2 = 1$
- $\Sigma = 1x_1 + 1x_2$
- $f : \Sigma \geq 1$

XOR gate

Why can't XOR gate be created using a perceptron?

Linearly Separable Problem

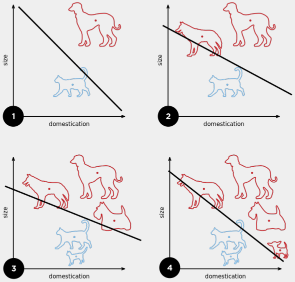


Figure: Linearly Separable Problem (Courtesy: Elizabeth Goodspeed from Wikimedia Commons)

- Now our problem is that the perceptron is a **linear classifier**, that means it could only separate datas that is linearly separable.
- Real-world problems are not that easy to separate
- How can we solve this problem?

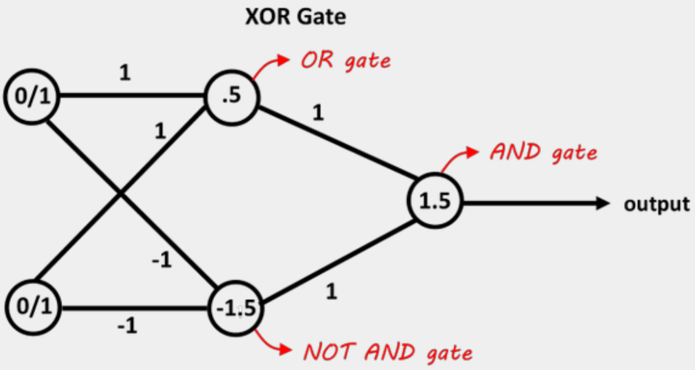


Figure: XOR gate with Perceptrons connected together (Courtesy: Parth Udawant)

How are the weights of each dendrites (inputs) are automatically adjusted?

How are the weights of each dendrites (inputs) are automatically adjusted?

- Through a **backpropagation** Process

How are the weights of each dendrites (inputs) are automatically adjusted?

- Through a **backpropagation** Process
    - Initiate weights randomly

How are the weights of each dendrites (inputs) are automatically adjusted?

- Through a **backpropagation** Process
  - Initiate weights randomly
  - Calculate the error over the training set

How are the weights of each dendrites (inputs) are automatically adjusted?

- Through a **backpropagation** Process
  - Initiate weights randomly
  - Calculate the error over the training set
  - Attempts to adjust weights little by little to minimise loss

How are the weights of each dendrites (inputs) are automatically adjusted?

- Through a **backpropagation** Process
  - Initiate weights randomly
  - Calculate the error over the training set
  - Attempts to adjust weights little by little to minimise loss

    (Seriously, one day it will converge. There exists a mathematical proof)

Your next step into Machine Learning

Courses in KU-CPE to be taken

- Engineering Mathematics I
- Discrete Mathematics and Linear Algebra
- Aritificial Intelligence/Machine Learning

MOOCs

For practical use, go ahead with...

- Google's Machine Learning Crash Course
- Udemy's Machine Learning (UD120)

Thanks!

https://twitter.com/public_srakrn
https://www.facebook.com/srakrn
sirakorn.l@ku.th