MMI 727 - DEEP LEARNING: METHODS AND APPLICATIONS

Week 2: Introduction to Machine Learning

Alptekin Temizel
atemizel@metu.edu.tr

Cihan Öngün
congun@metu.edu.tr

Outline

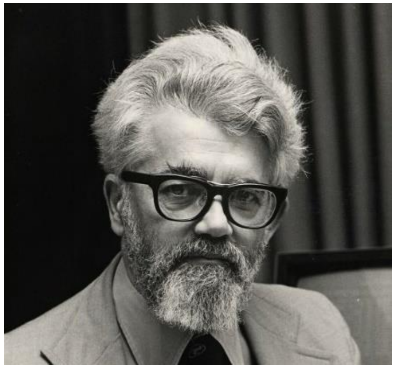
- History of AI
- What is Machine Learning
- Types and Applications of Machine Learning
- Training and Test
- Measuring Success
- Overfitting , Bias-Variance
- Gradient descent
- Image Classification Example
- Introduction to Deep Learning

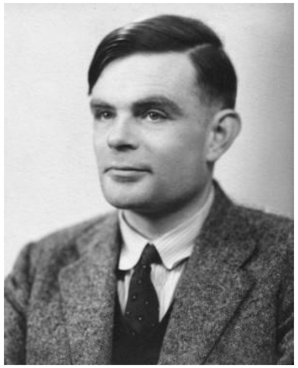
History of AI

- First formal publications are in 1940s
- Alan Turing
- Increasing popularity after success during WW2
- Mostly boolean logic, if-else, decision trees etc.
- The field of AI research was born at a workshop at Dartmouth College in 1956

- Herbert Simon: «machines will be capable, within twenty years, of doing any work a man can do» (1965)
- Marvin Minsky : «within a generation ... the problem of creating 'artificial intelligence' will substantially be solved» (1967)
- Turing predicted that Turing Test will be passed in 50 years (1950)

- Winter of AI around 1980s





What is AI

- AI is the field devoted to building persons
- AI is the effort to automate intellectual tasks normally performed by humans.
- AI is the field devoted to building artifacts capable of displaying, in controlled, well-understood environments, and over sustained periods of time, behaviors that we consider to be intelligent, or more generally, behaviors that we take to be at the heart of what it is to have a mind
- Machine learning (ML) is the study of computer algorithms that improve automatically through experience without being explicitly programmed to do so.
- Deep learning (DL) is part of a broader family of machine learning methods based on artificial neural networks.

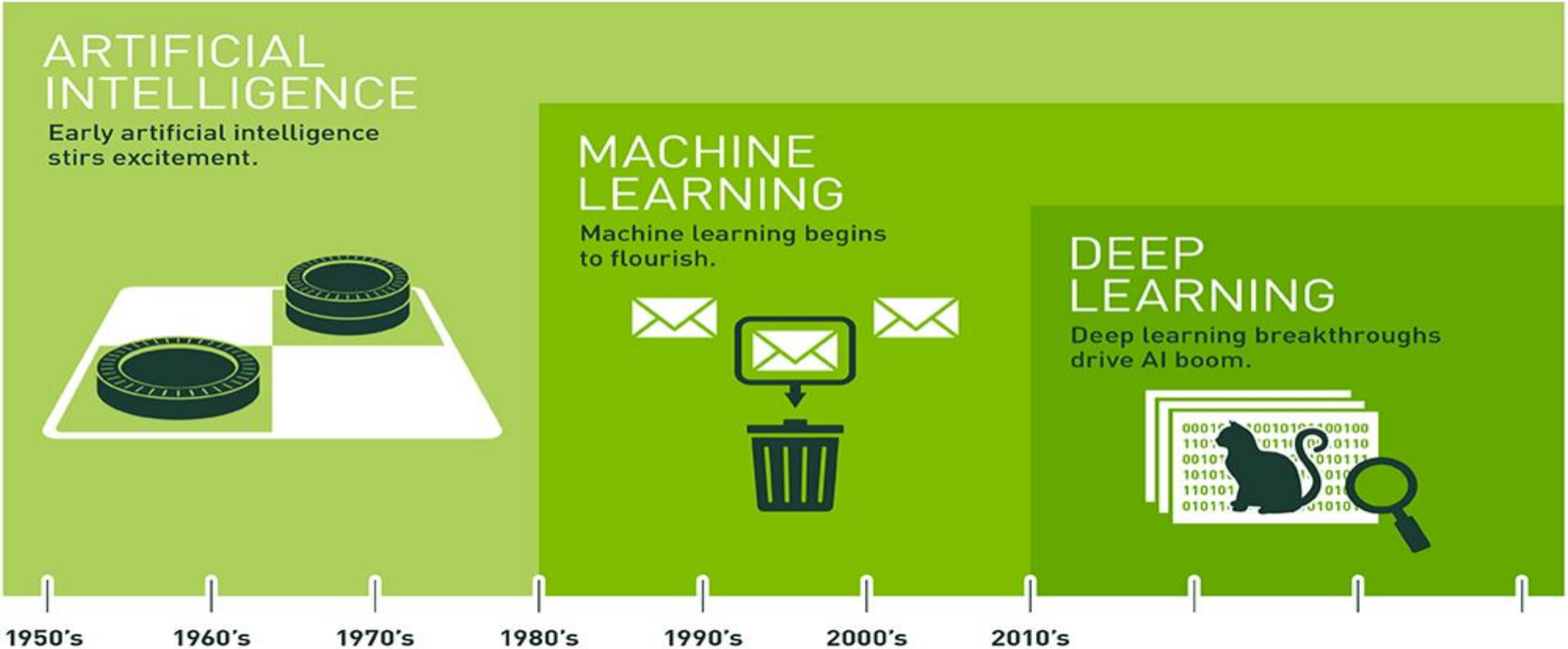
What is AI

- Artificial Intelligence
  - If-Else
  - Boolean Logic
  - Symbolic Operations
  - …

  - Machine Learning
    - K-means
    - SVM
    - Linear Regression
    - …

    - Deep Learning
      - Fully Connected Neural Networks
      - Convolutional Neural Networks
      - Recurrent Neural Networks
      - …

AI – Machine Learning



Turing Award 2018

Learning

- Predicting new actions from past experiences

- There is no need to “learn” to calculate payroll

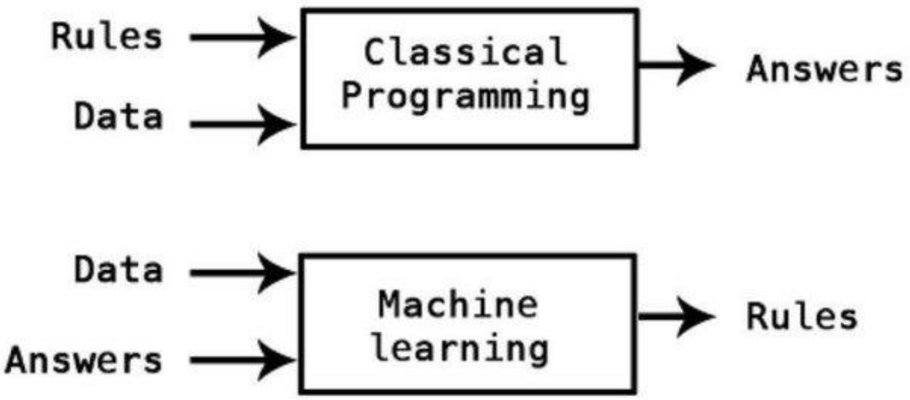
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.

- Build a model that is a good and useful approximation to the data.

What is Machine Learning

- Machine Learning is the ability to teach a computer without explicitly programming it
- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
    - Solve the optimization problem
    - Representing and evaluating the model for inference



Applications of Machine Learning

- Handwriting Recognition
  - convert written letters into digital letters
- Language Translation
  - translate spoken and or written languages (e.g. Google Translate)
- Speech Recognition
  - convert voice snippets to text (e.g. Siri, Cortana, and Alexa)
- Image Classification
  - label images with appropriate categories (e.g. Google Photos)
- Autonomous Driving
  - enable cars to drive

Data Mining

- Retail: Market basket analysis, Customer relationship management (CRM)
- Finance: Credit scoring, fraud detection
- Manufacturing: Control, robotics, troubleshooting
- Medicine: Medical diagnosis
- Telecommunications: Spam filters, intrusion detection
- Bioinformatics: Motifs, alignment
- Web mining: Search engines
- ...

Features in Machine Learning

- Features are the observations that are used to form predictions
  - For image classification, the pixels are the features
  - For voice recognition, the pitch and volume of the sound samples are the features
  - For autonomous cars, data from the cameras, range sensors, and GPS are features

- Extracting relevant features is important for building a model
  - Time of day is an irrelevant feature when classifying images
  - Time of day is relevant when classifying emails because SPAM often occurs at night

- Common Types of Features in Robotics
  - Pixels (RGB data)
  - Depth data (sonar, laser rangefinders)
  - Movement (encoder values)
  - Orientation or Acceleration (Gyroscope, Accelerometer, Compass)

Types of Machine Learning

- **Supervised Learning**
  - Training data is labeled
  - Goal is to correctly label new data

- **Unsupervised Learning**
  - Training data is unlabeled
  - Goal is to categorize the observations

- **Reinforcement Learning**
  - Training data is unlabeled
  - System receives feedback for its actions
  - Goal is to perform better actions

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

→ Cat

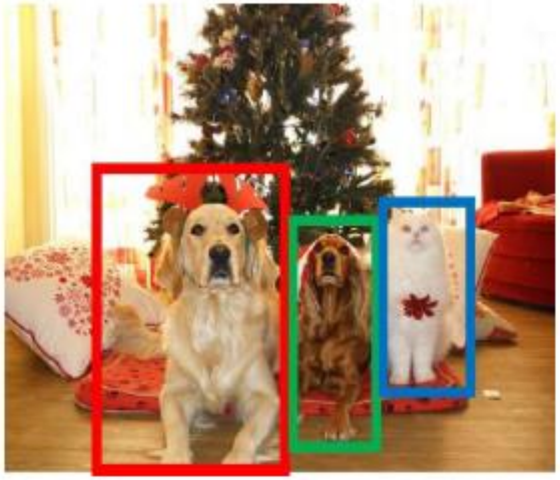
Classification

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



DOG, DOG, CAT

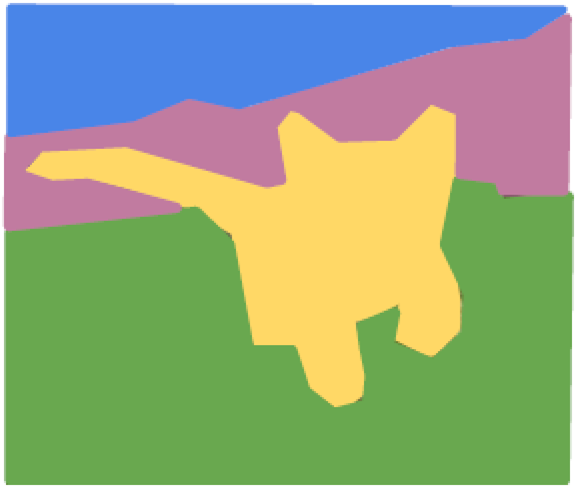
Object Detection

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



TREE, SKY

Semantic Segmentation

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

A cat sitting on a suitcase on the floor

Image captioning

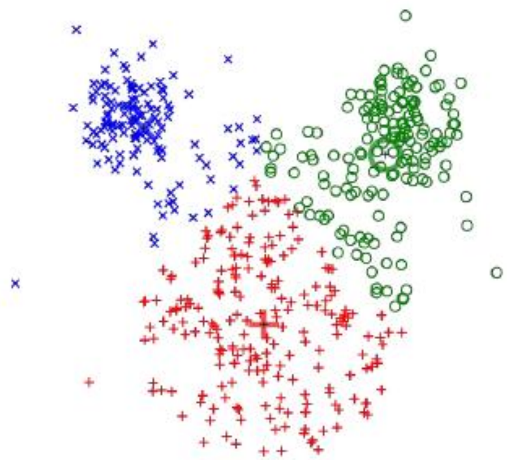
Caption generated using neuraltalk2
Image is CC0 Public domain.

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



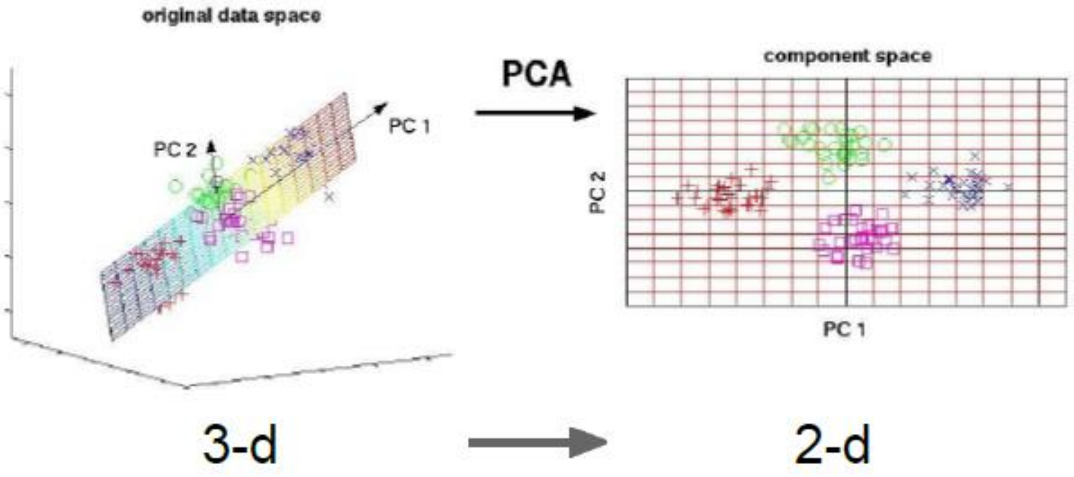
K-means clustering

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



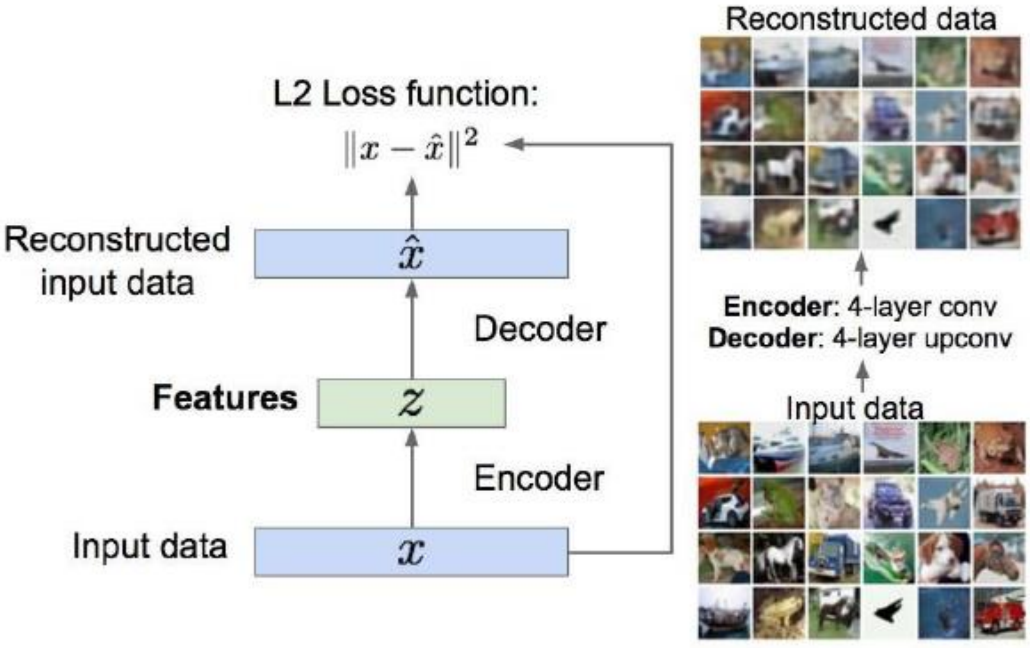
Principal Component Analysis
(Dimensionality reduction)

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Autoencoders
(Feature learning)

Unsupervised Learning

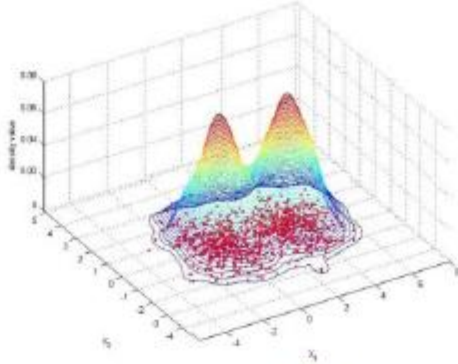
Data: x
Just data, no labels!

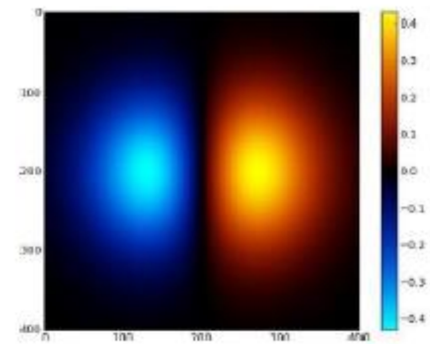
Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation





2-d density estimation

2-d density images left and right are CC0 public domain

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

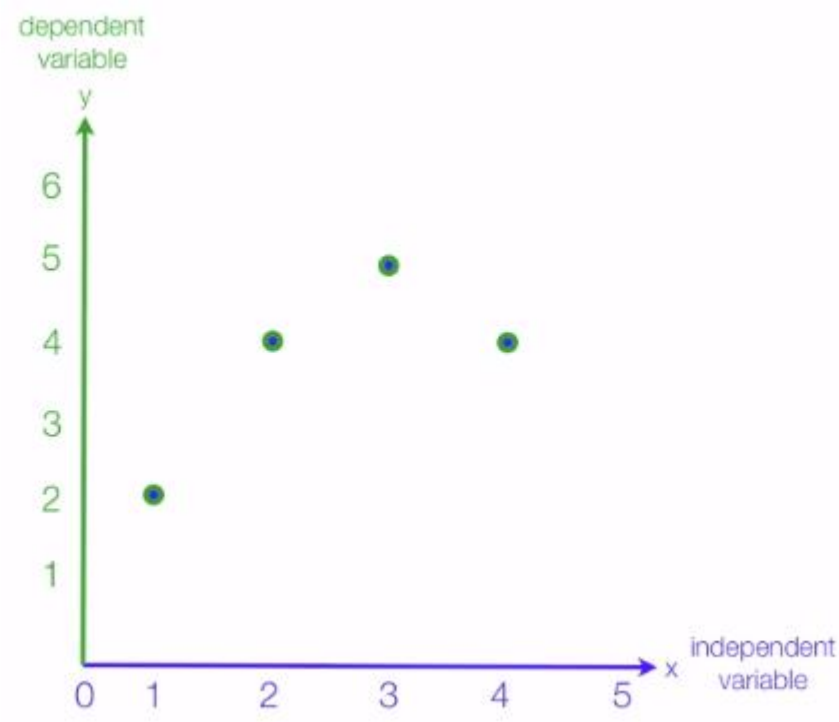
Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised Learning Algorithms

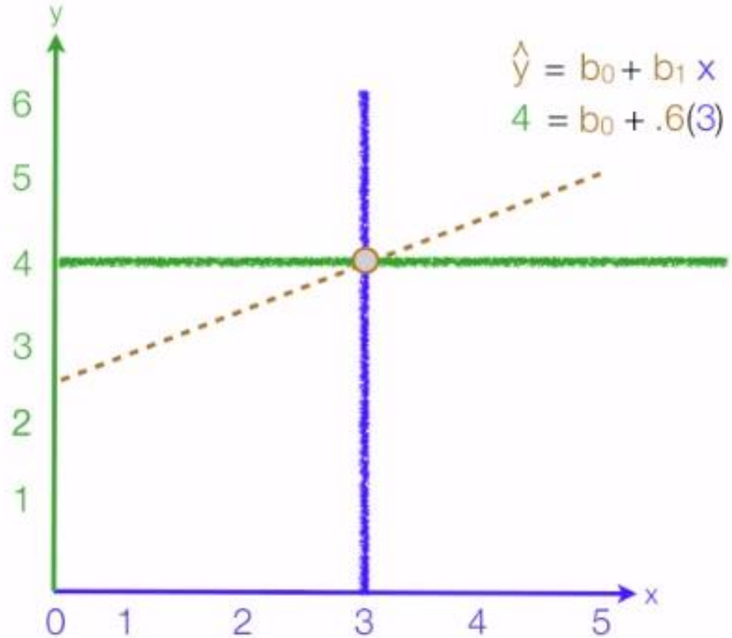
- Linear Regression
- Decision Trees
- Support Vector Machines
- K-Nearest Neighbor
- …

Linear Regression



independent variable	dependent variable
x	y
1	2
2	4
3	5
4	4
5	5

Linear Regression



$\hat{y} = b_0 + b_1 x$

$4 = b_0 + .6(3)$

$b_0 = 2.2$

$b_1 = .6$

$\hat{y} = 2.2 + .6x$

	x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
	1	2	-2	-2	4	4
	2	4	-1	0	1	0
	3	5	0	1	0	0
	4	4	1	0	1	0
	5	5	2	1	4	2
					10	6
mean	3	4				

$4 = b_0 + .6(3)$

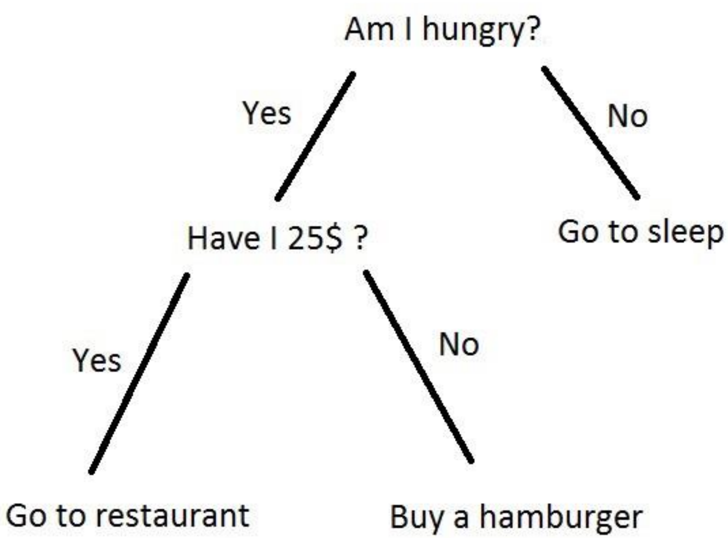
$4 = b_0 + \cancel{1.8}$

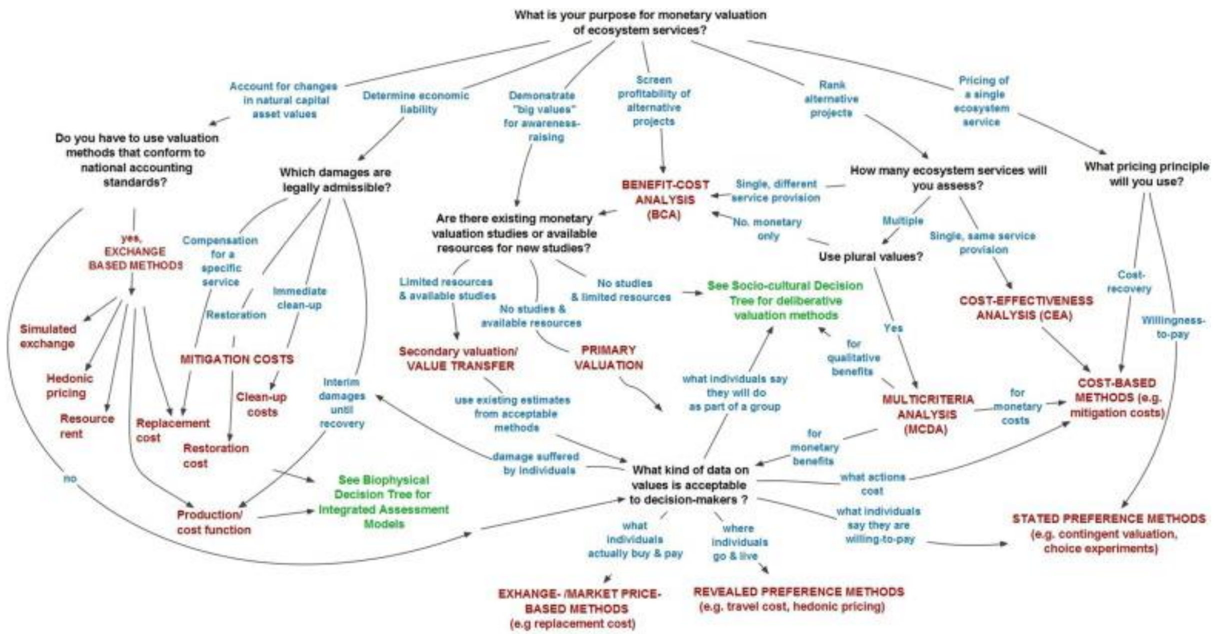
$-1.8 \qquad -\cancel{1.8}$

$2.2 = b_0$

$$b_1 = \frac{6}{10} = .6 = \frac{\Sigma (x - \bar{x})(y - \bar{y})}{\Sigma (x - \bar{x})^2}$$

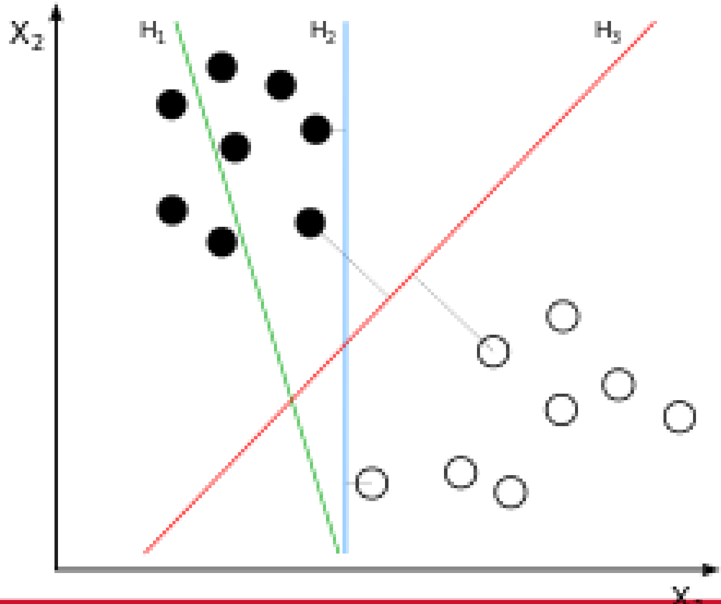
Decision Tree

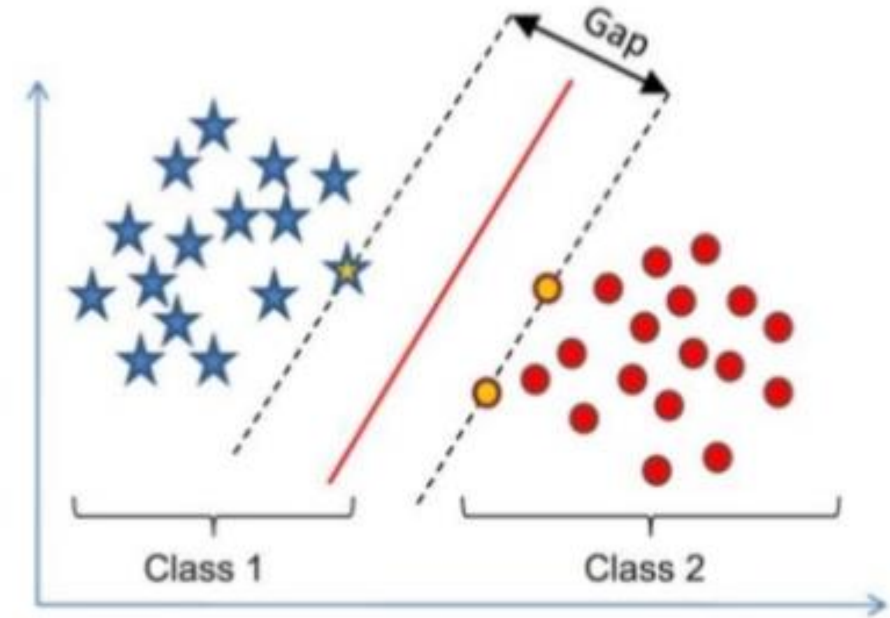




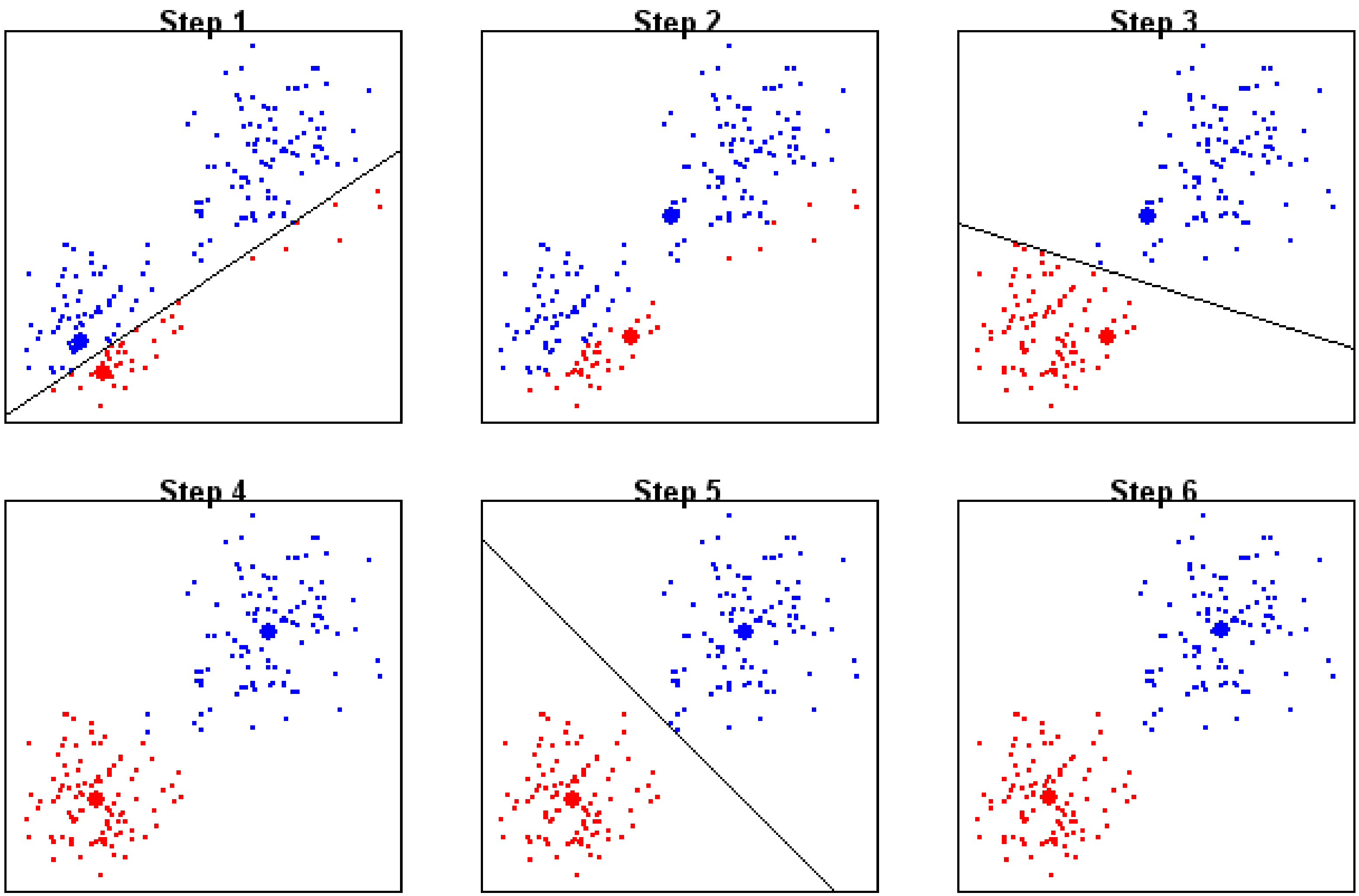
Support Vector Machine

Find a linear decision surface that can separate classes and has the largest distance between border points

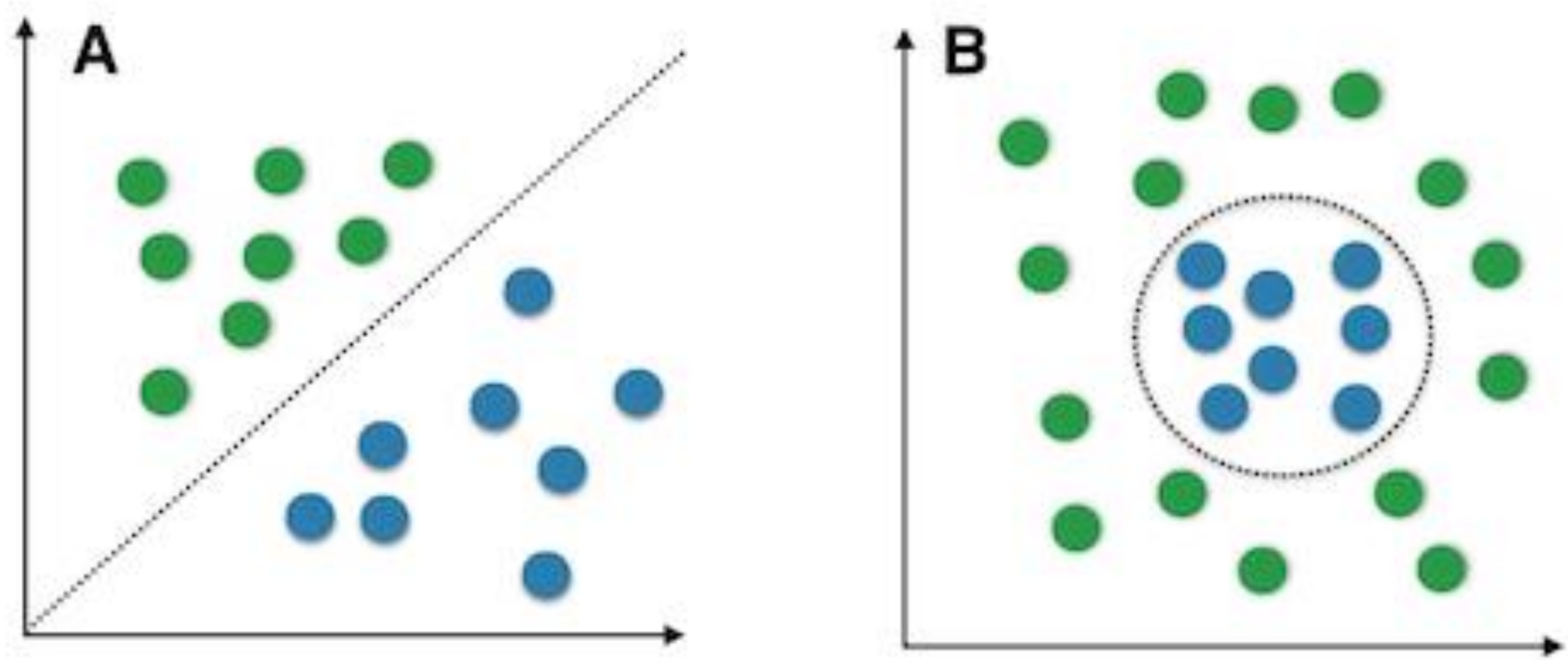




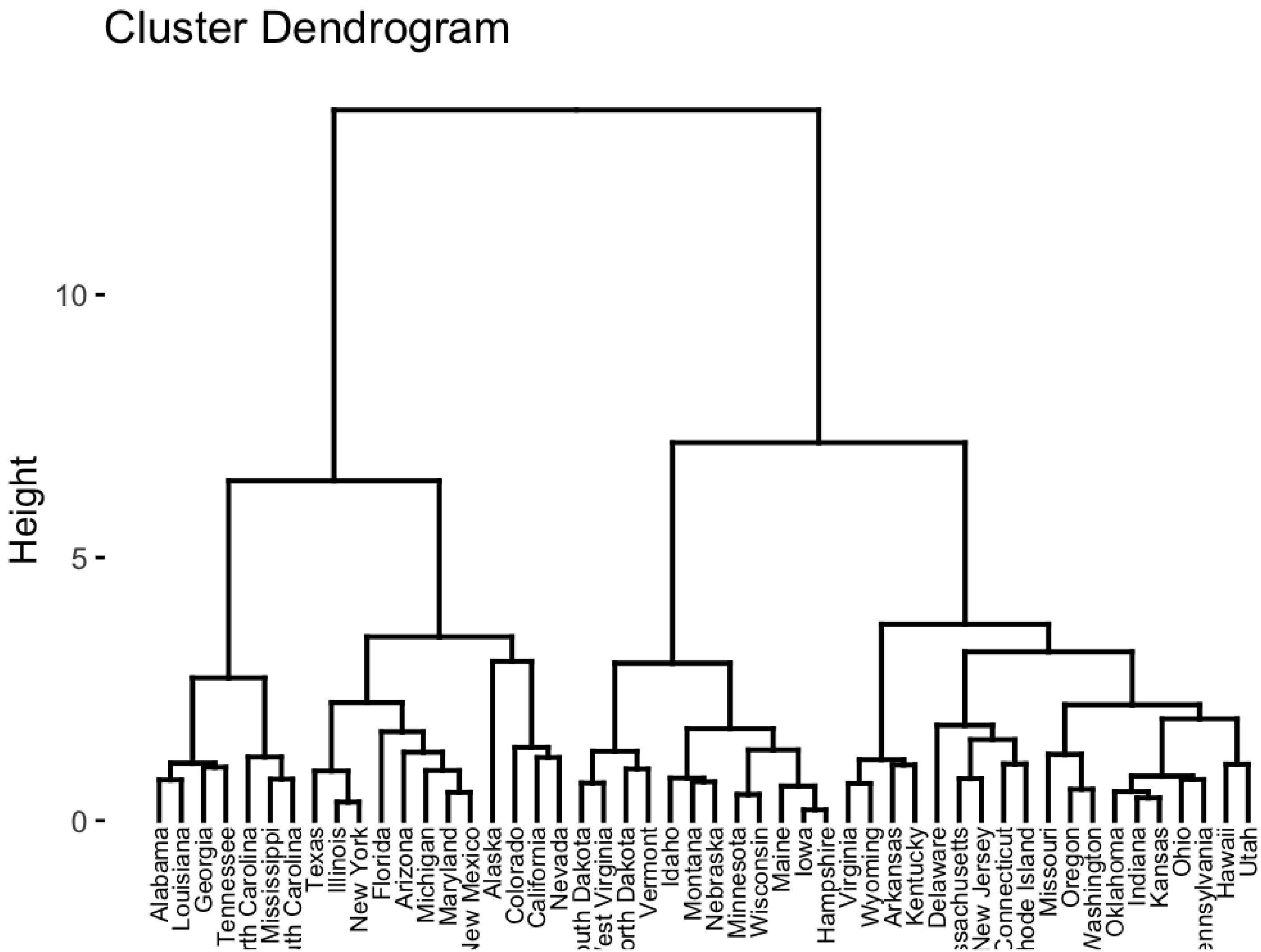
K-means (Unsupervised)



Linear – nonlinear problems

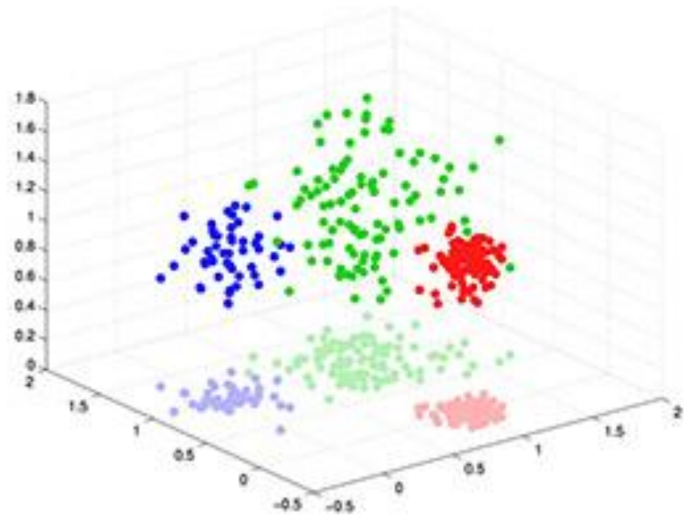


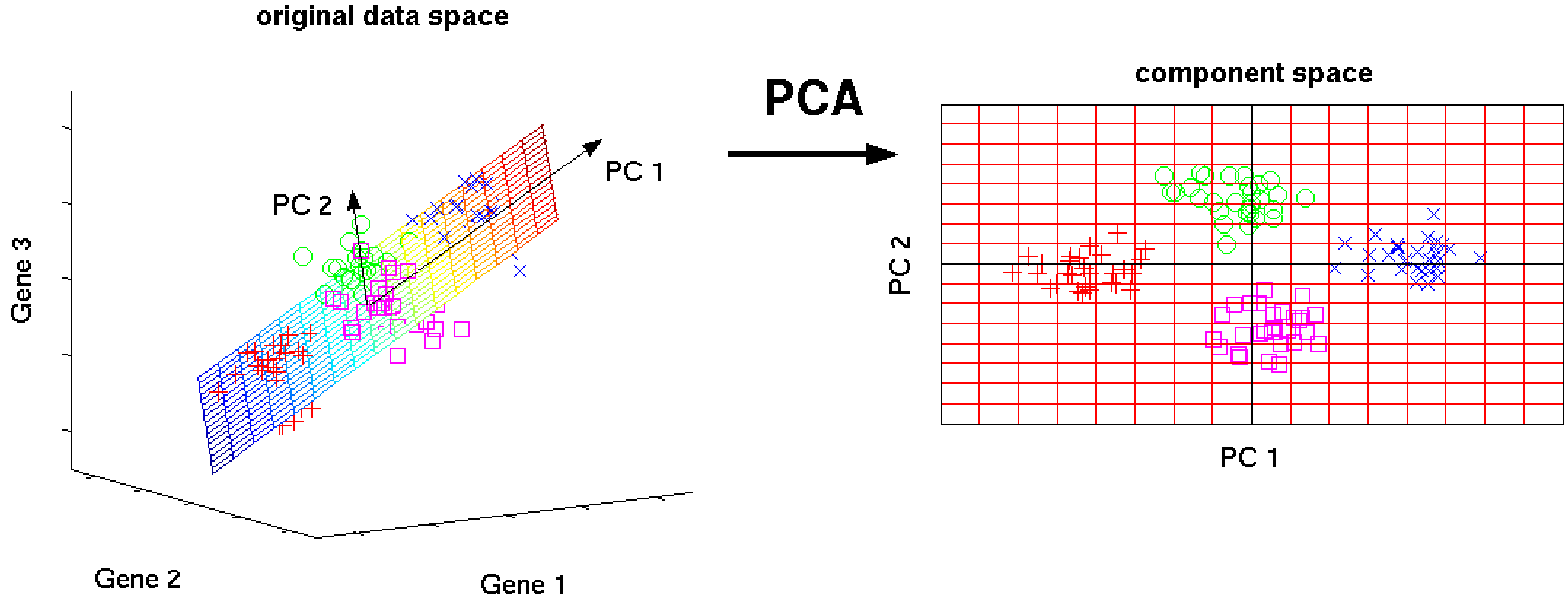
Agglomerative Clustering



Reducing Dimensionality

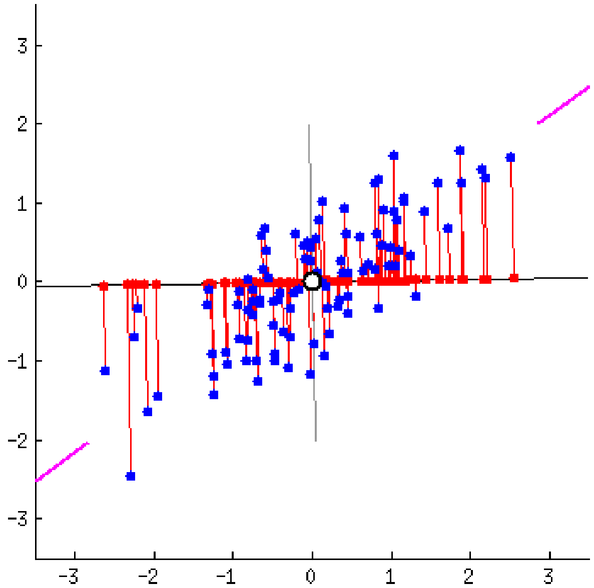
Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.



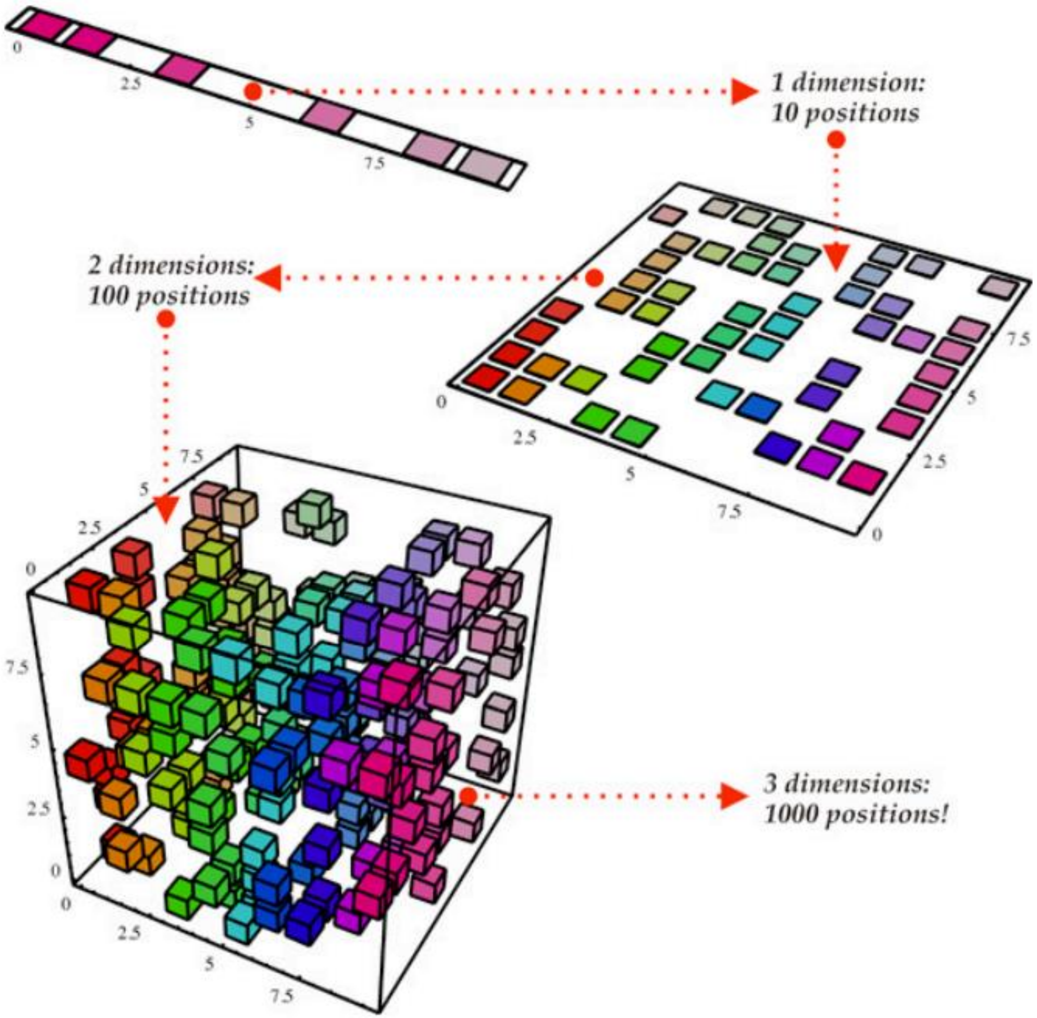


Reducing Dimensionality

- Reduces time complexity: Less computation
- Reduces space complexity: Less parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions



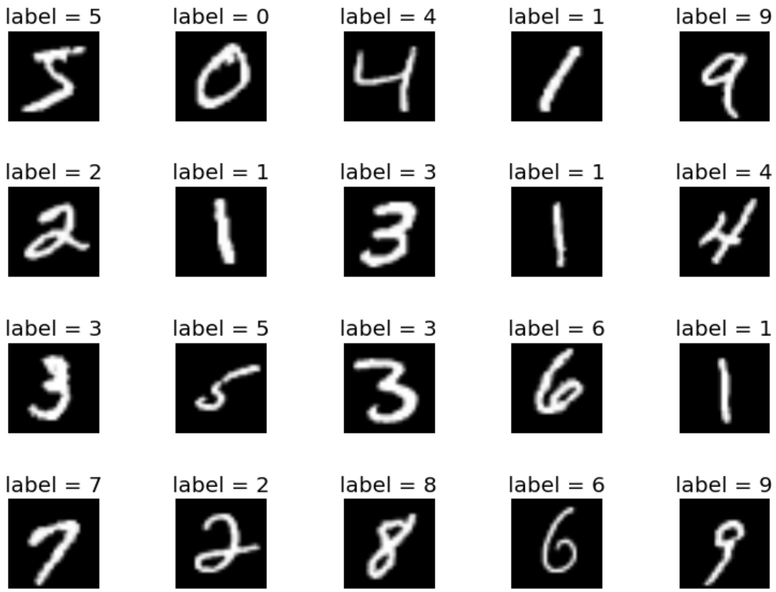
Reducing Dimensionality



Dataset

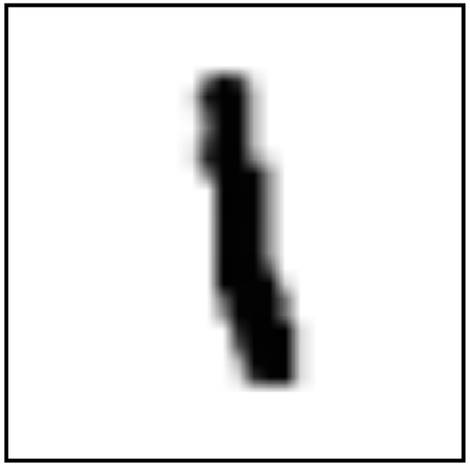
- A collection of data?
- Semantically grouped, large number of samples with given attributes
- Image, 3D model, audio, video, network, signal, behavior, usage, finance etc.

Dataset



MNIST dataset

- Handwritten digits
- 60000 training + 10000 test samples
- All labeled
- 28x28 resolution



$$\approx \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

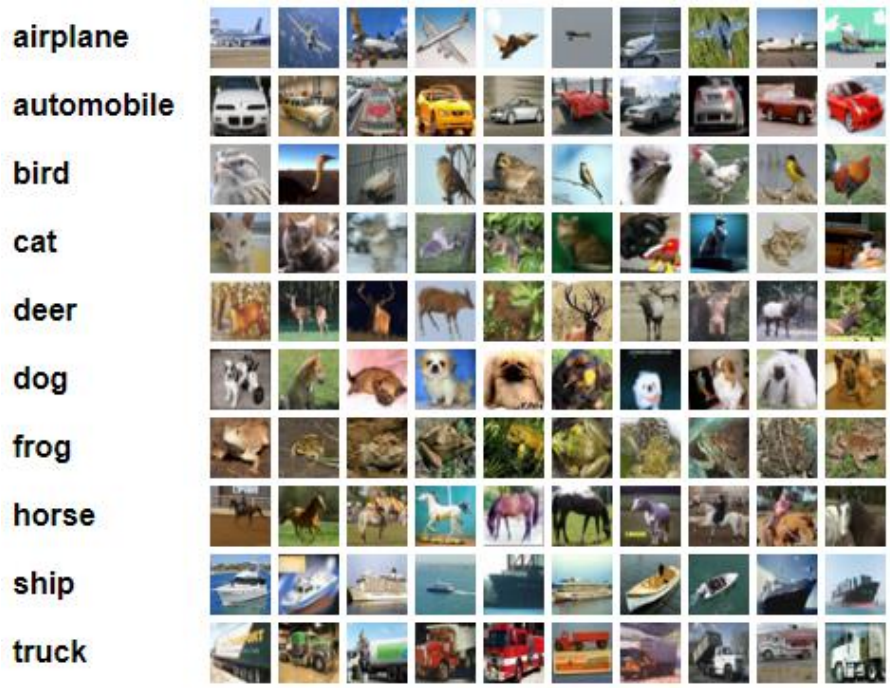
Dataset

CIFAR10

- Color images
- 10 classes
- 50000 training + 10000 test samples
- Labeled
- 32x32 resolution

CIFAR100

- Color images
- 100 classes
- 500 training + 100 test samples per class
- Labeled
- 32x32 resolution



Dataset

ImageNet

- 1.200.000 images
- 1000 categories (subcategories)

COCO

- 330.000 images (1.5 million object instances)
- 80 object categories (5 captions per image)
- Object detection, segmentation, and captioning

CelebA

- 200.000 celebrity faces
- 40 attributes

ShapeNet

- 51000 3D models

- Kaggle
- Google Dataset Search
- Make your own dataset!

Dataset

Training set

- Used for learning, that is to fit the parameters (weights, biases etc.)

Test set

- Used only to assess the performance (accuracy, success etc.) of a fully specified model

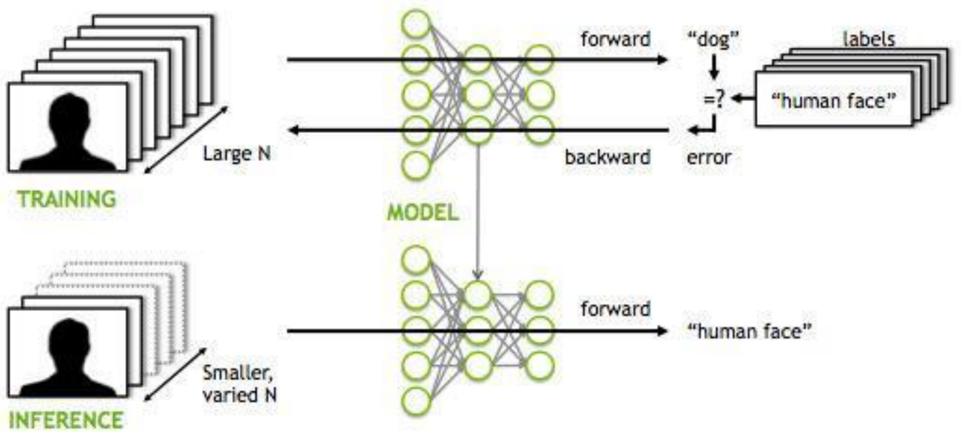
Validation set

- Used to tune the hyperparameters (architecture, layers, units etc.)
- Prevent overfitting

Training	Validation	Test
%75	%25	Separate
%50	%25	%25
%70	%20	%10

ML/DL development pipeline

- Select/Implement a suitable model for your data/task
- Train the model with the data
  - Forward pass to get predictions
  - Calculate loss/error between predictions and true labels
  - Backward pass to update model parameters/weights to minimize the loss
- Test your model with unseen data to evaluate the model performance
- Deploy the trained model to end device/user



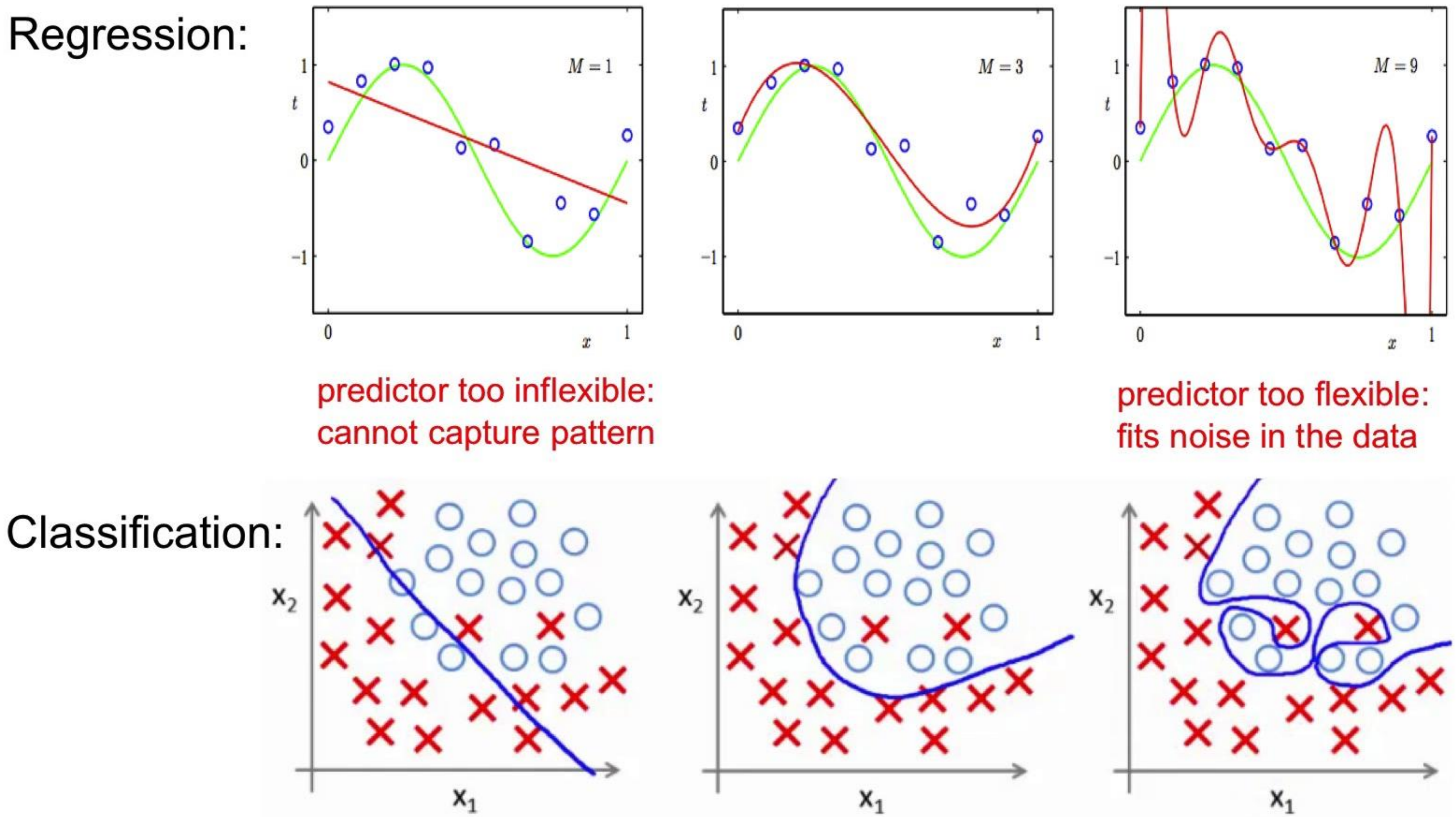
Overfitting

- The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably
- Model performs well on training data but poorly on test data
- Poor generalization

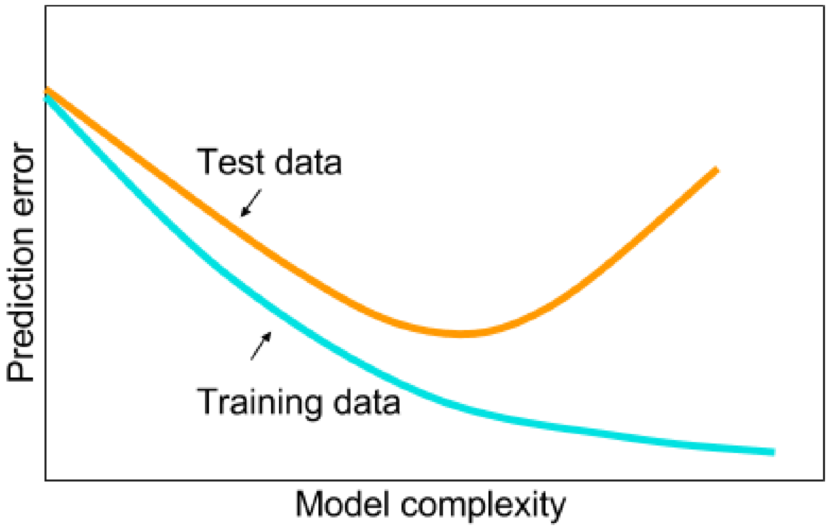
Underfitting

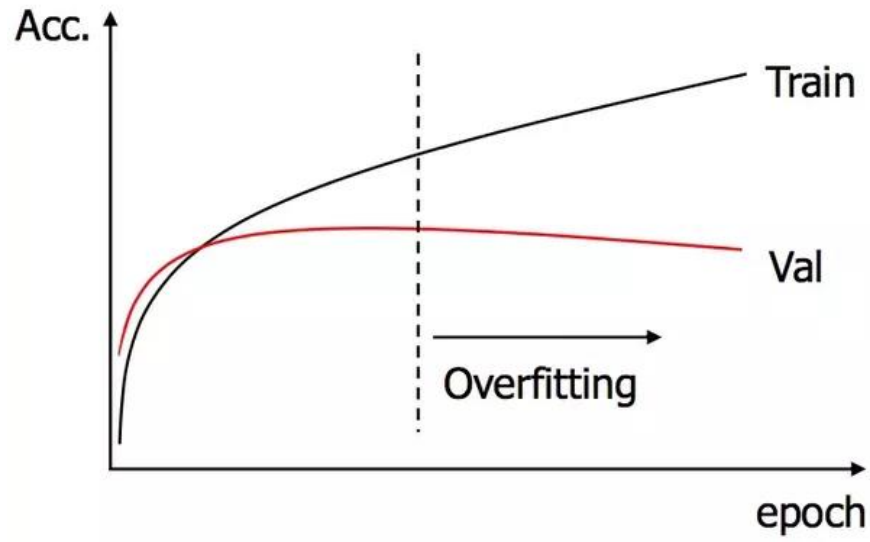
- Model cannot capture the underlying structure of the data
- Performs poorly on training data
- Not enough learning

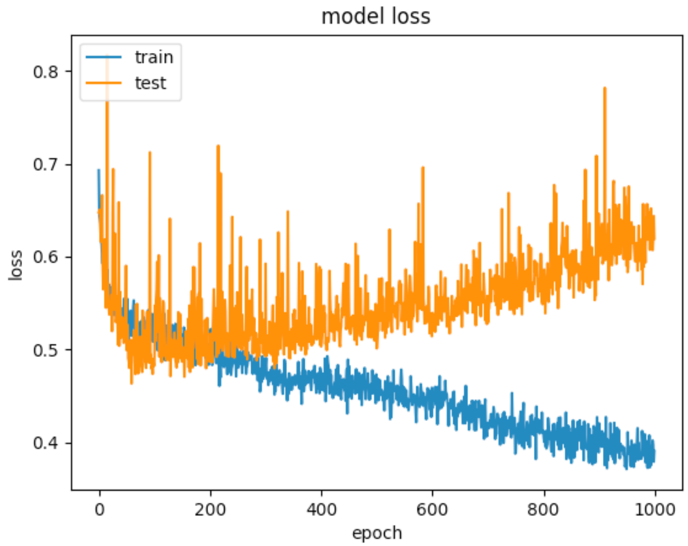
Overfitting



Overfitting







Measuring success

- True Positive: Correctly identified as relevant
- True Negative: Correctly identified as not relevant
- False Positive: Incorrectly labeled as relevant
- False Negative: Incorrectly labeled as not relevant

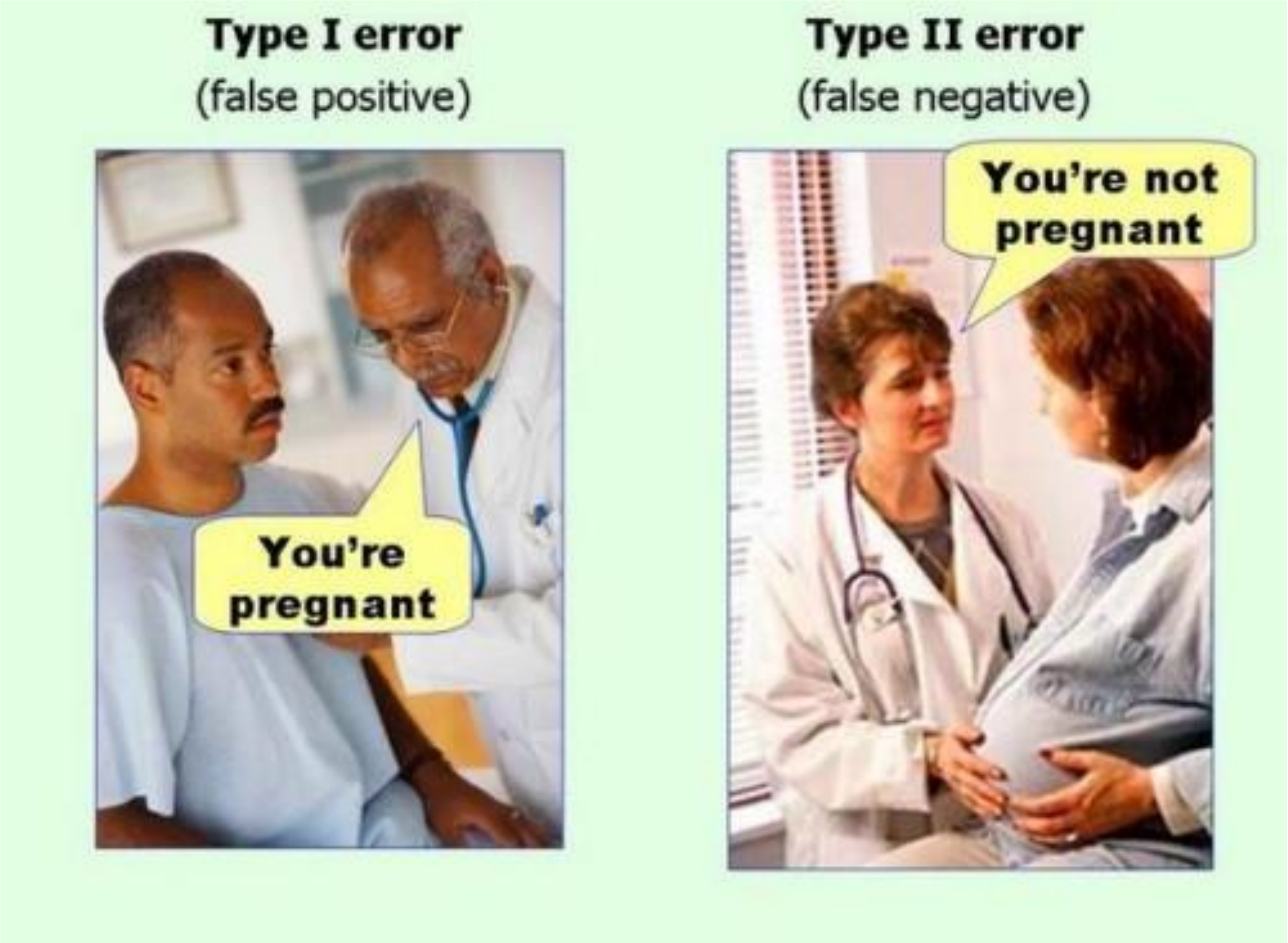
Prediction:

Image:

True Positive | **True** Negative | False Negative | False Positive

Images from the STL-10 dataset

Measuring success



Measuring success

Bias

- expected difference between model's prediction and truth

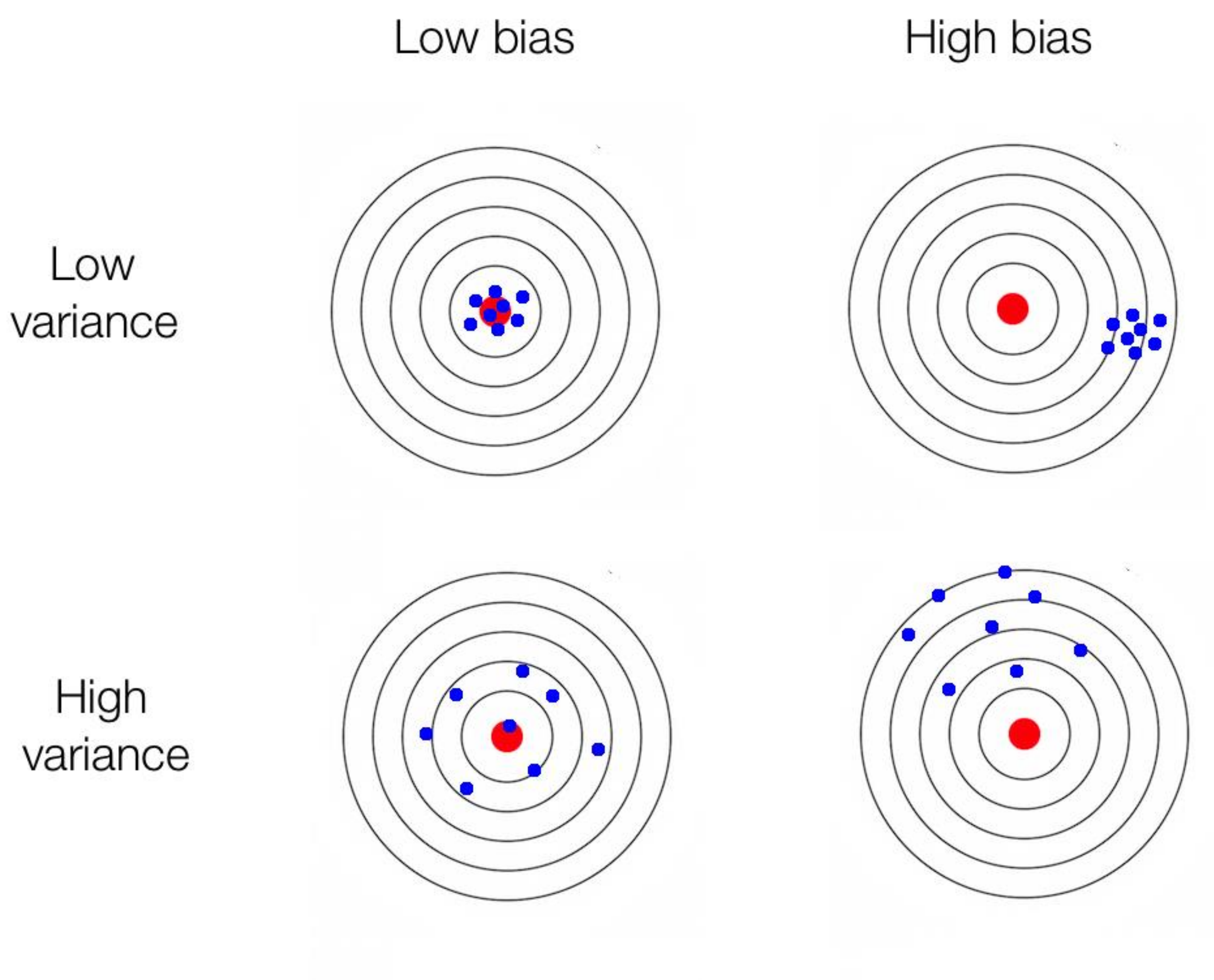
Variance

- how much the model differs among training sets

Model Scenarios

- High Bias: Model makes inaccurate predictions on training data
- High Variance: Model does not generalize to new datasets
- Low Bias: Model makes accurate predictions on training data
- Low Variance: Model generalizes to new datasets

Measuring success



Measuring success

Accuracy

- Percentage of correct labels
- Accuracy = (# true positives + # true negatives) / (# of samples)

Precision

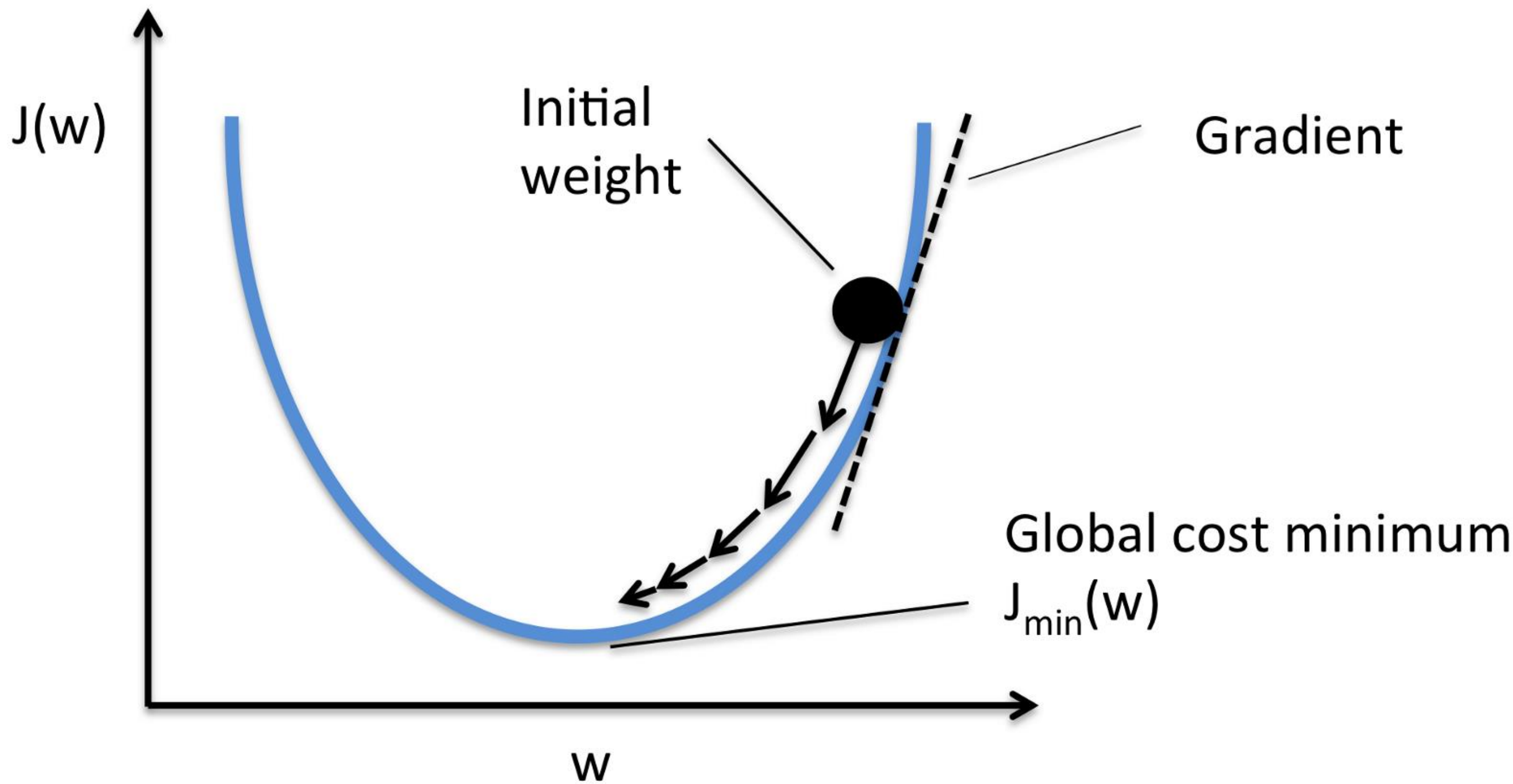
- Percentage of positive labels that are correct
- Precision = (# true positives) / (# true positives + # false positives)

Recall

- Percentage of positive examples that are correctly labeled
- Recall = (# true positives) / (# true positives + # false negatives)

Gradient Descent

Find minimum point of a function

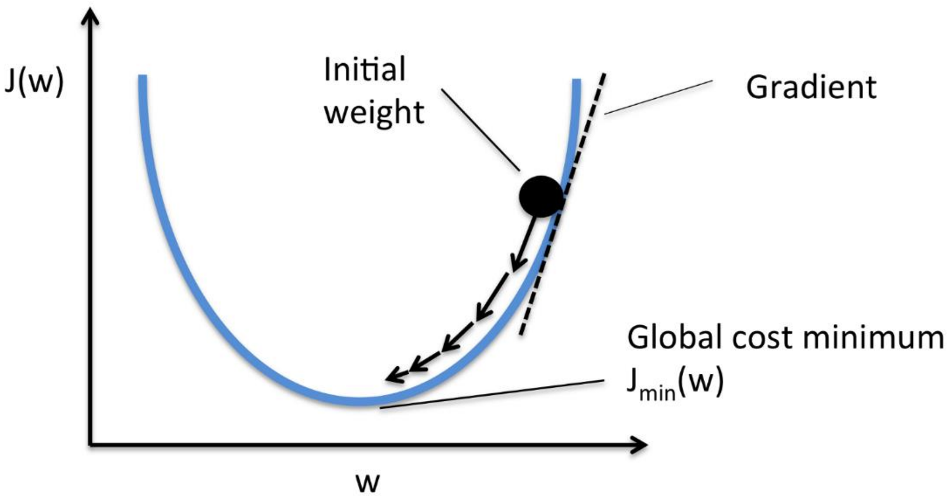


Gradient Descent

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

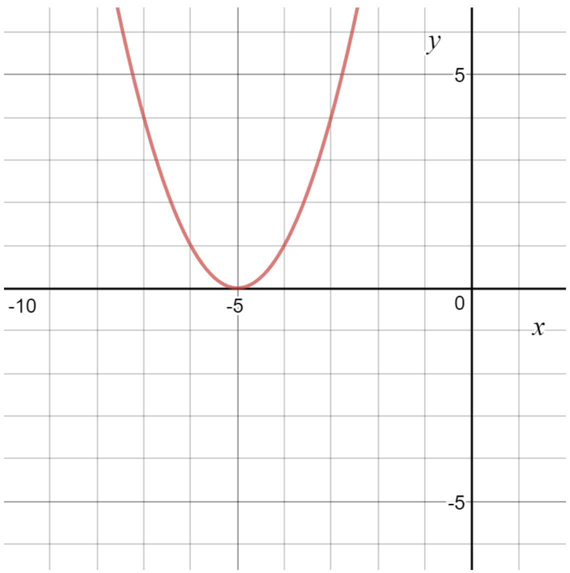
}



$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent

Question : Find the local minima of the function $y=(x+5)^2$ starting from the point $x=3$



Initialize Parameters :

$X_0 = 3$

Learning rate = 0.01

$$\frac{dy}{dx} = \frac{d}{dx}(x+5)^2 = 2 * (x+5)$$

Iteration 1 :

$$X_1 = X_0 - (learning\ rate) * (\frac{dy}{dx})$$

$$X_1 = 3 - (0.01) * \big(2 * (3+5)\big) = 2.84$$

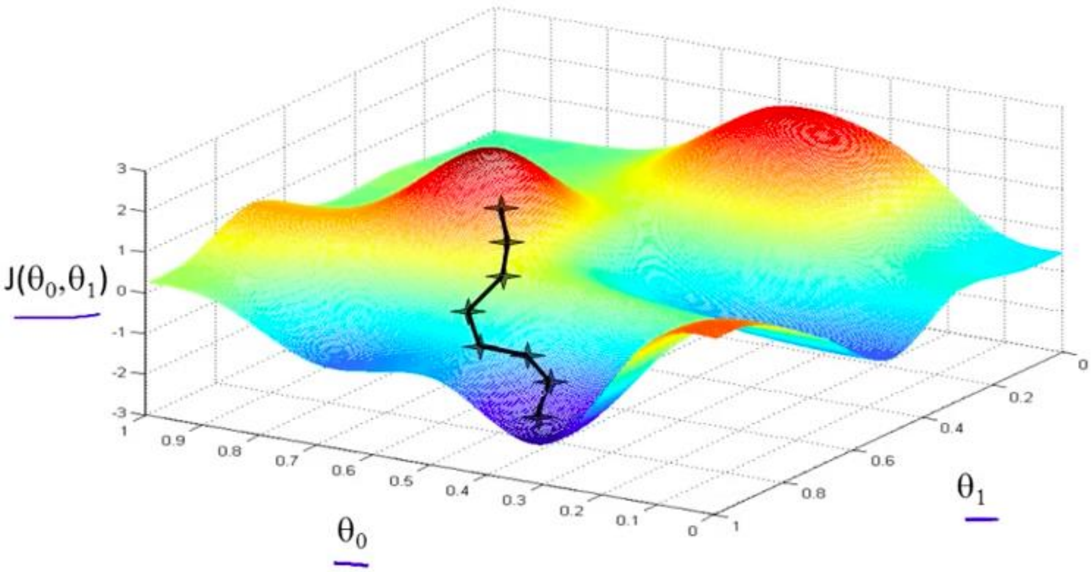
Iteration 2 :

$$X_2 = X_1 - (learning\ rate) * (\frac{dy}{dx})$$

$$X_2 = 2.84 - (0.01) * \big(2 * (2.84+5)\big) = 2.6832$$

https://towardsdatascience.com/implement-gradient-descent-in-python-9b93ed7108d1

Gradient Descent



repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

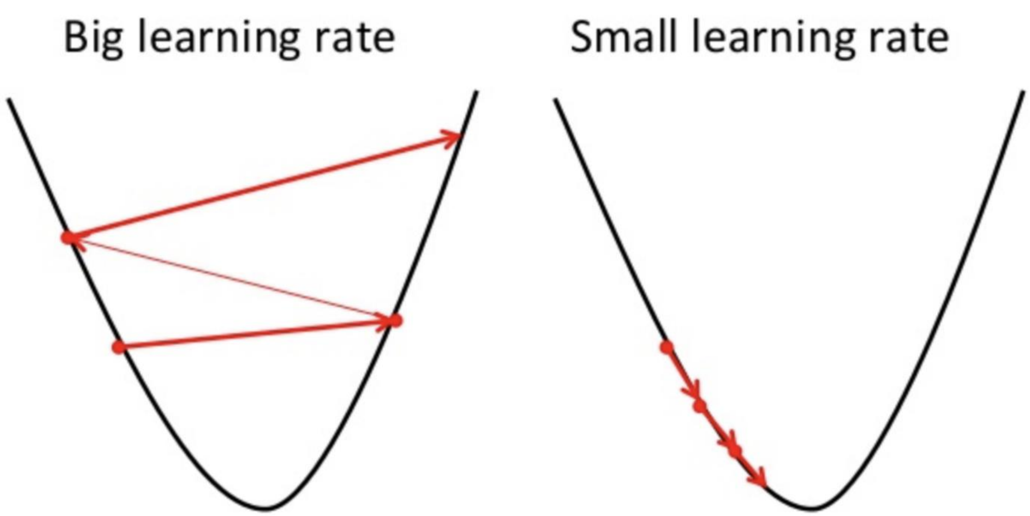
Gradient Descent

Big learning rate

- Never converges or diverges

Small learning rate

- Too long time to converge



Gradient Descent

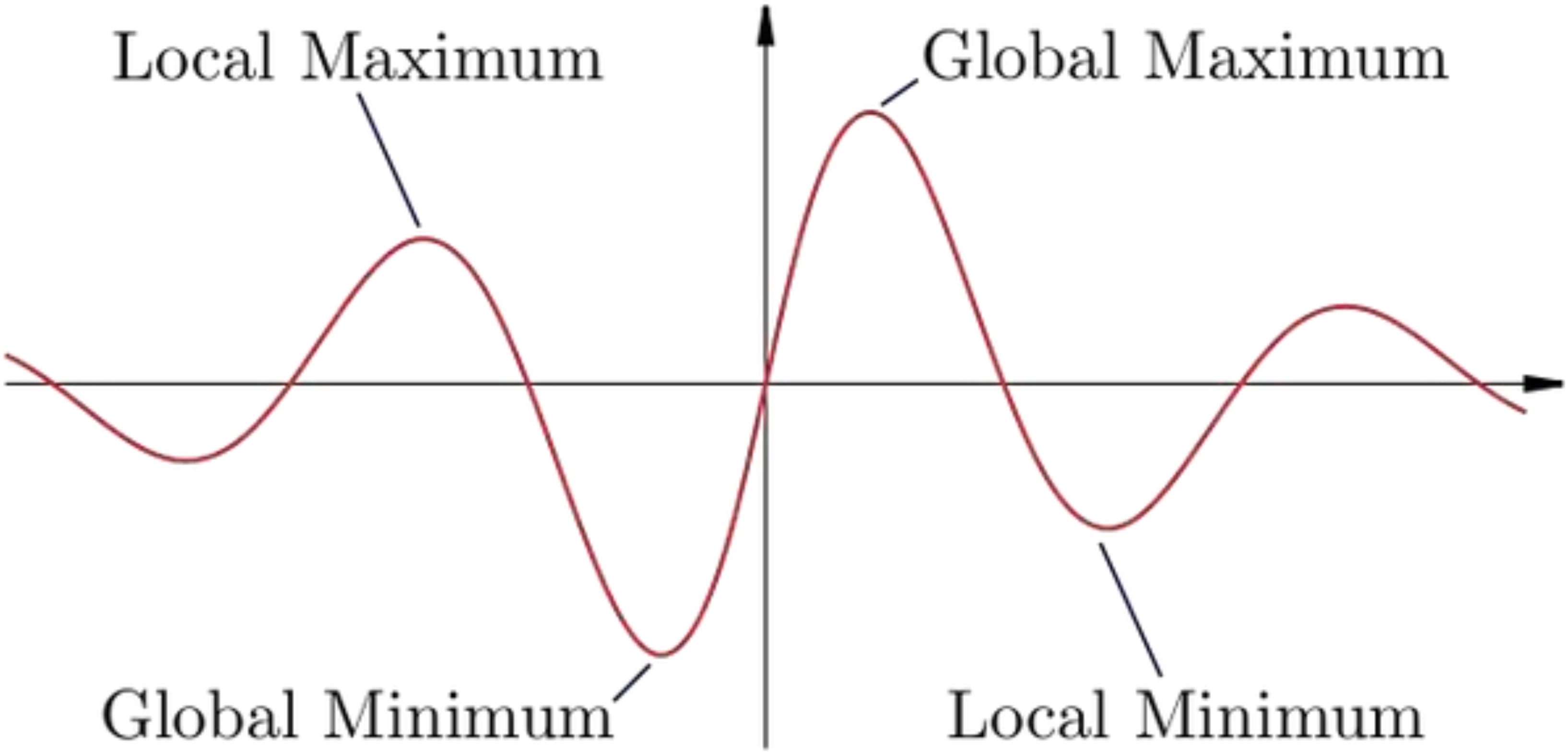


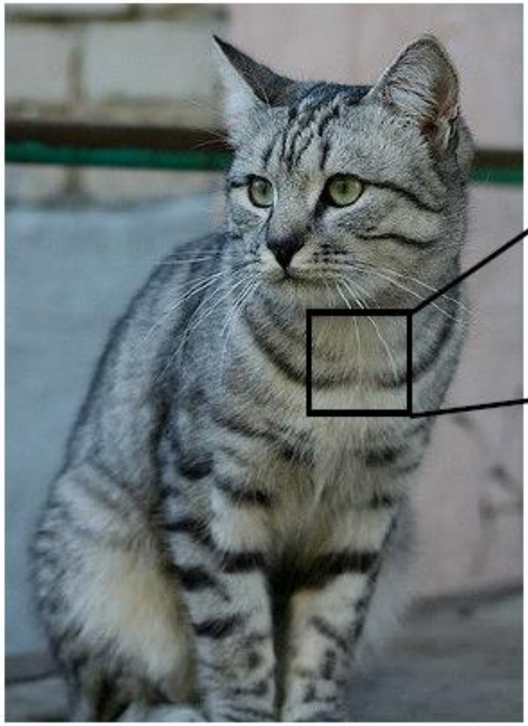
Image Classification

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Image Classification

The Problem: Semantic Gap




```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

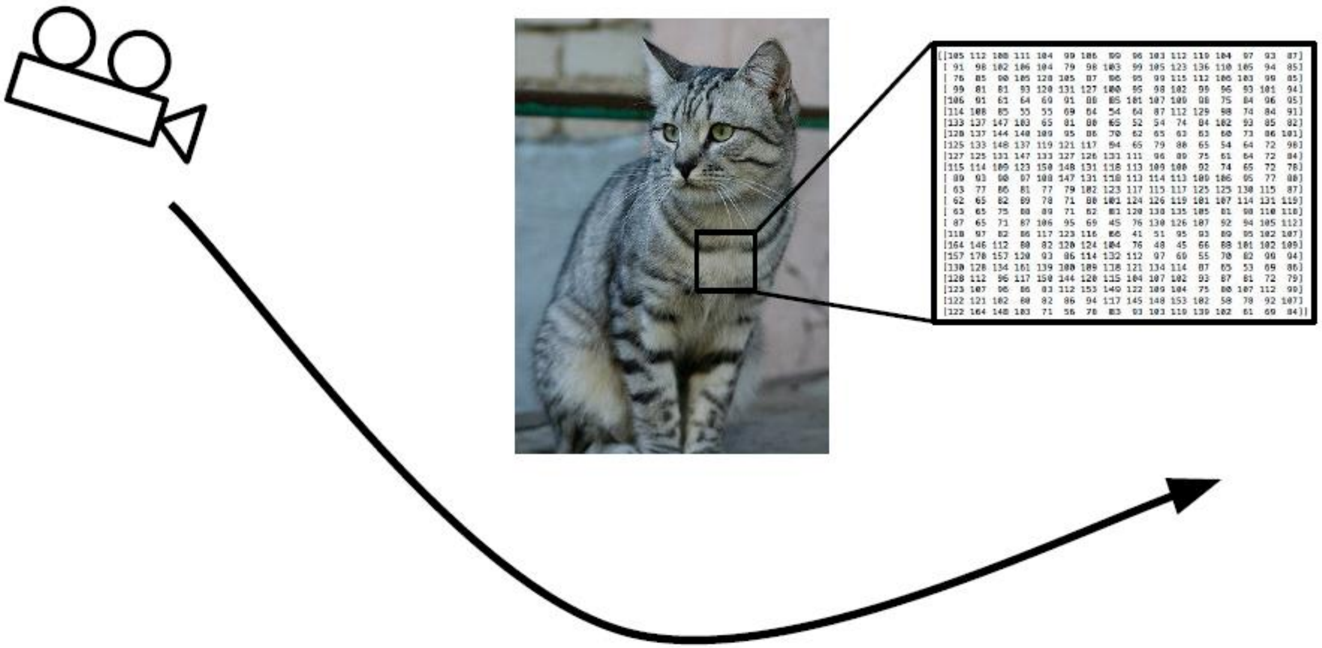
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Image Classification

Challenges: Viewpoint variation



All pixels change when the camera moves!

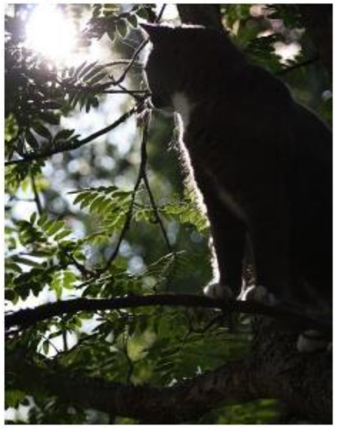
Image Classification

Challenges: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

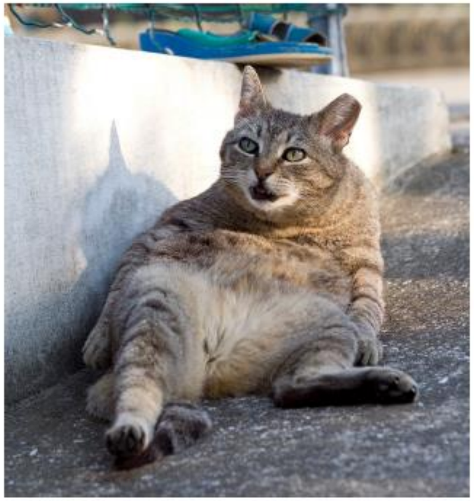
This image is CC0 1.0 public domain



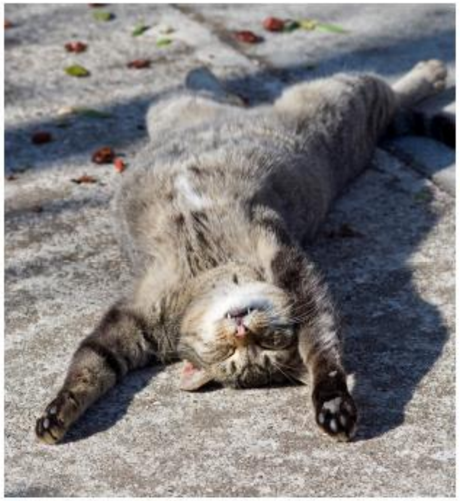
This image is CC0 1.0 public domain

Image Classification

Challenges: Deformation

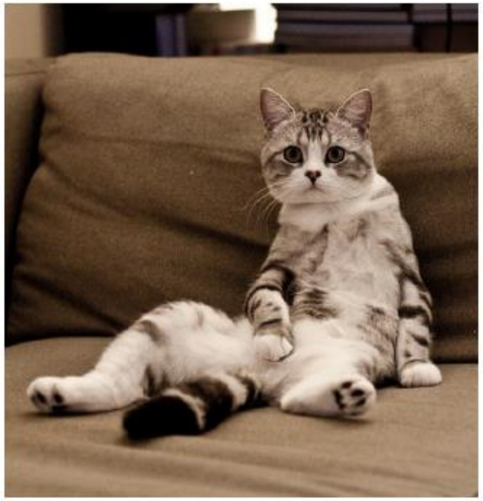


This image by Umberto Salvagnin is licensed under CC-BY 2.0



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0



This image by Tom Thai is licensed under CC-BY 2.0

Image Classification

Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Image Classification

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Image Classification

Challenges: Intraclass variation

Image Classification

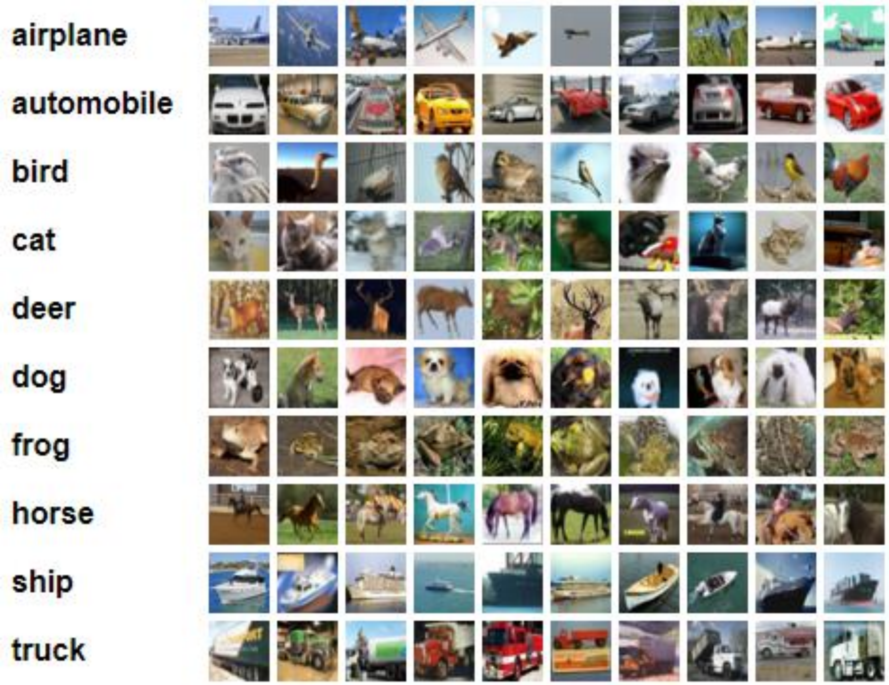
- No obvious way to hardcode
- If-else is not enough

Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels

2. Use Machine Learning to train a classifier

3. Evaluate the classifier on new images



Nearest Neighbor Classifier

Nearest Neighbor Classifier

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

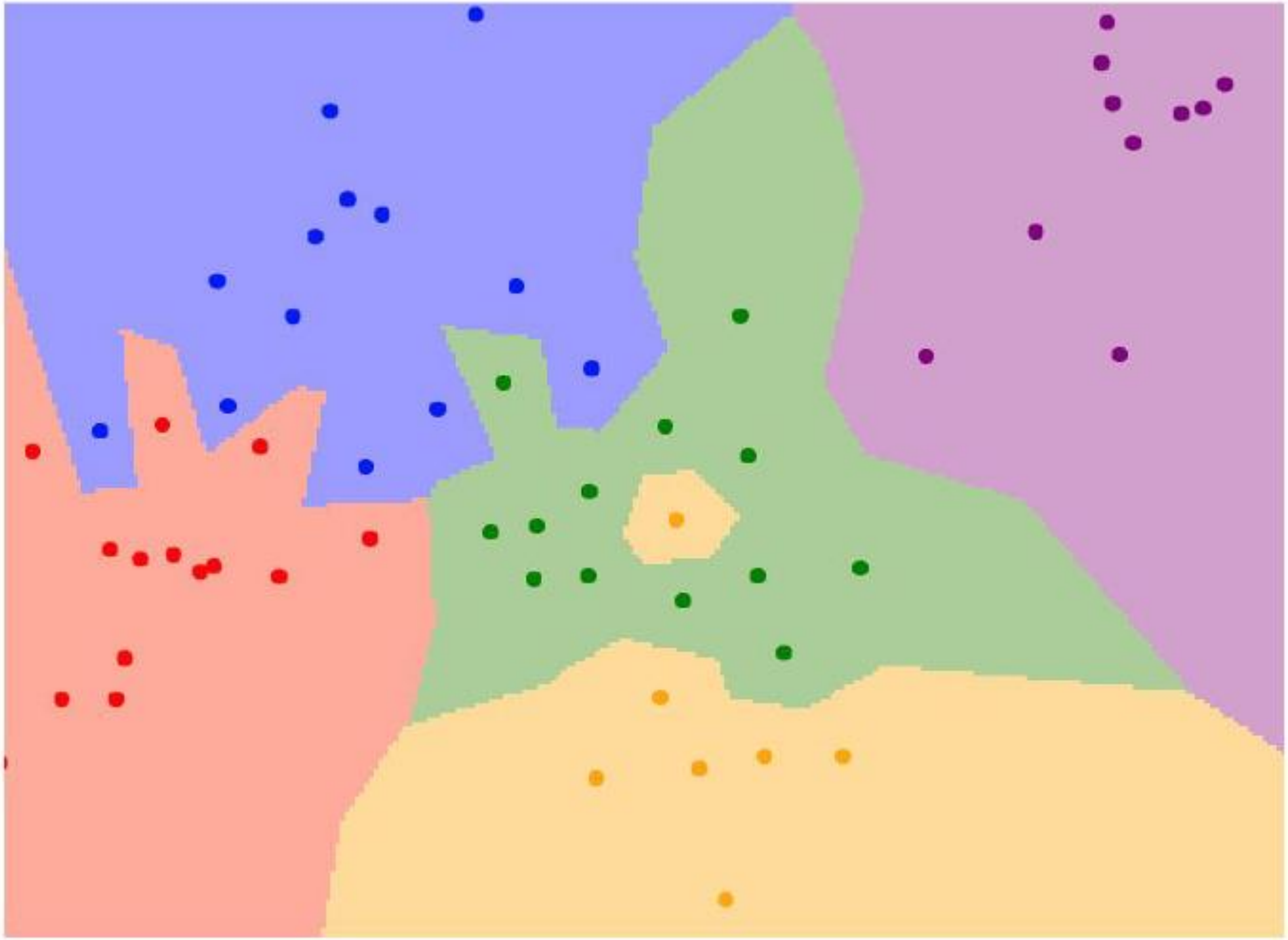
=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

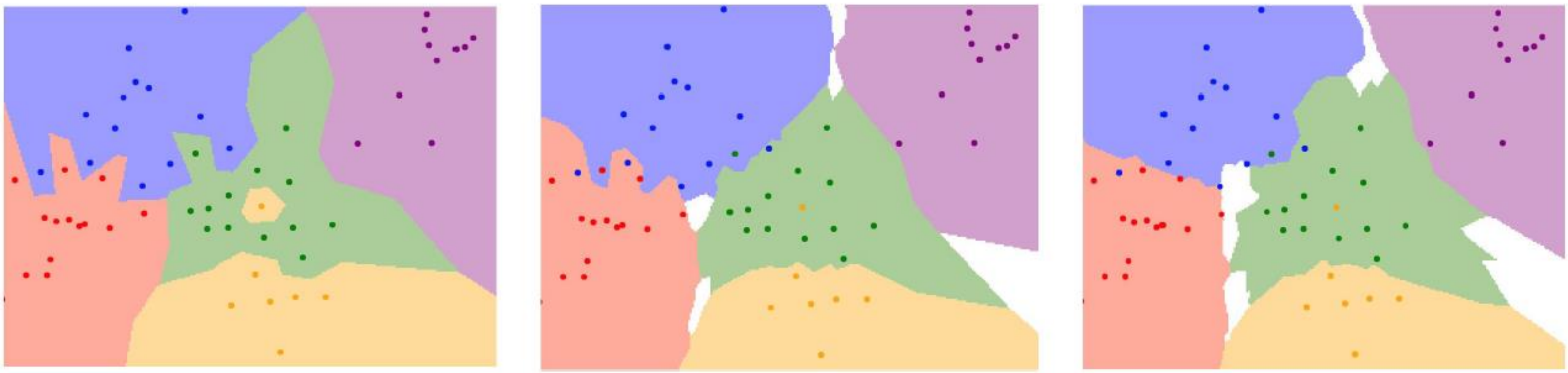
$\xrightarrow{\text{add}}$ 456

Nearest Neighbor Classifier



K-Nearest Neighbors Classifier

Instead of copying label from nearest neighbor, take **majority vote** from K closest points

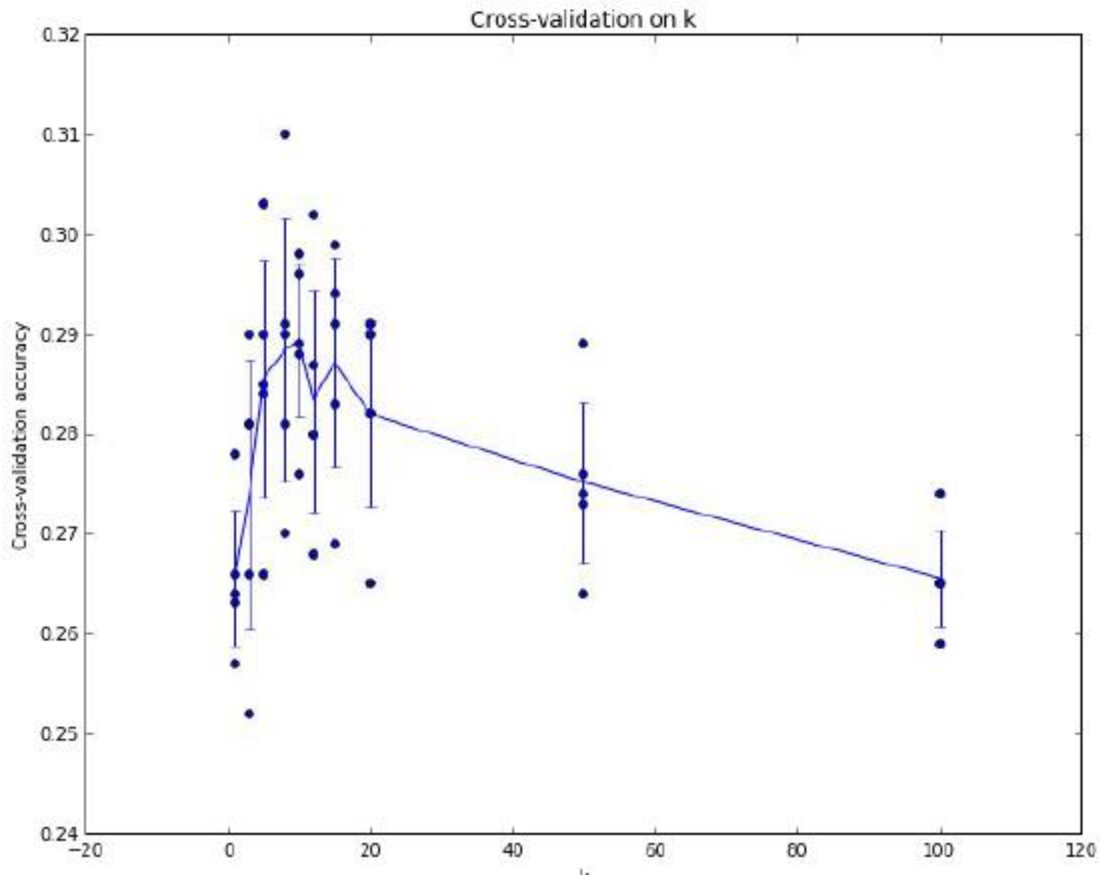


K = 1

K = 3

K = 5

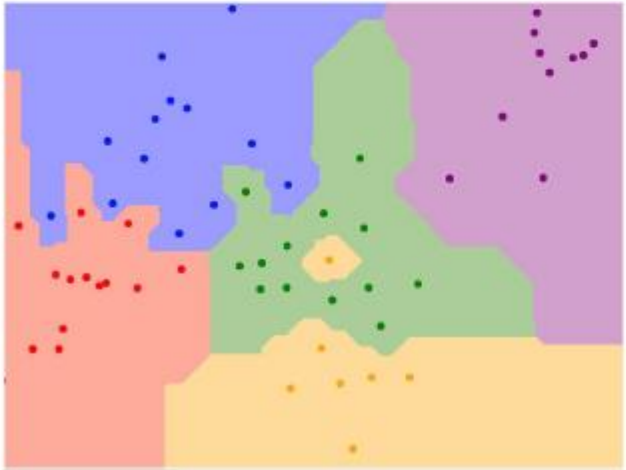
K-Nearest Neighbors Classifier



K-Nearest Neighbors Classifier

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

K = 1

Image Classification with K-Nearest Neighbors Classifier

Summary

- Aim is to predict the classes of images in the test set

- We started with a **training set** using images and labels

- We used **K-Nearest Neighbours** to make predictions

- **K value** and **distance metrics** are **hyperparameters**
- We found the **best hyperparameters** using **validation set**

- We tested the system using **test set** to calculate success (**accuracy**) of the model

Linear Classifier

$$f(x,W) = Wx$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

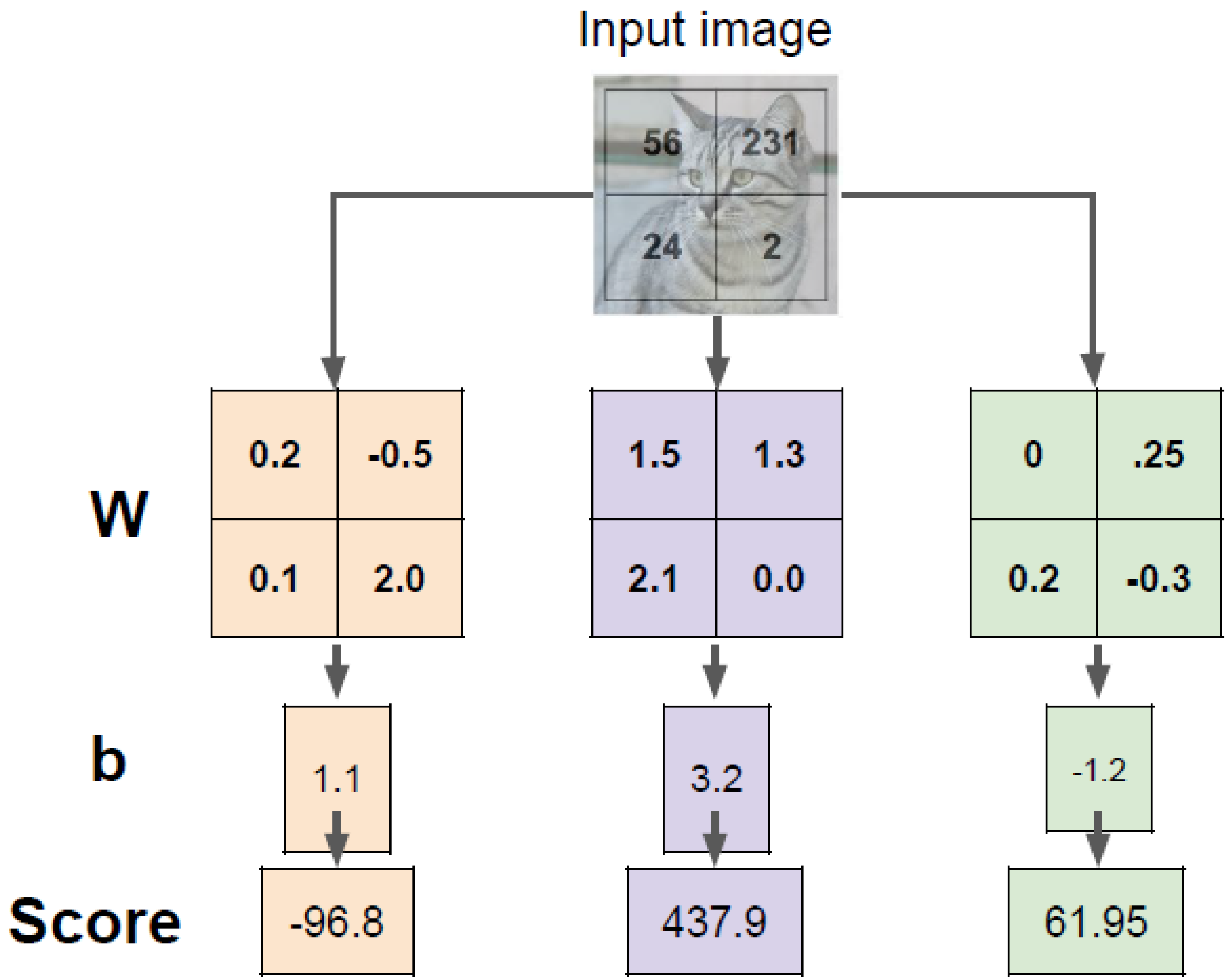
$\longrightarrow$ f(**x**,**W**) $\longrightarrow$ **10** numbers giving class scores

$\uparrow$

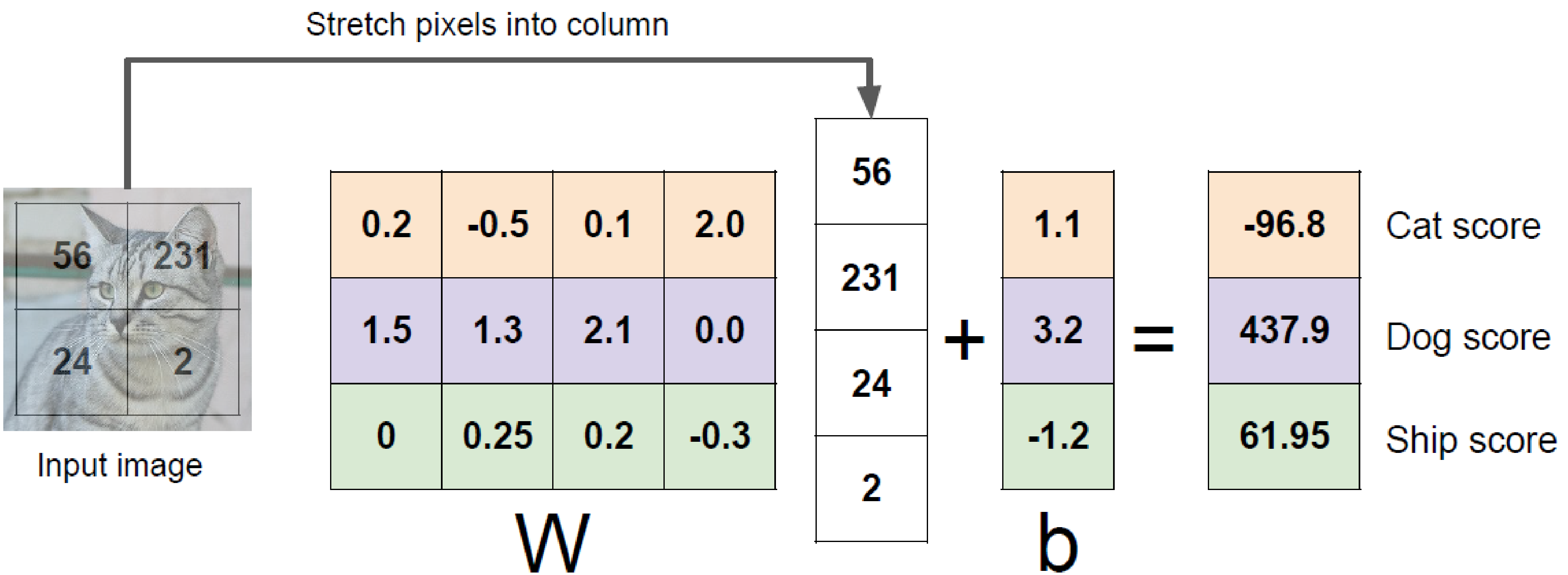
W

parameters
or weights

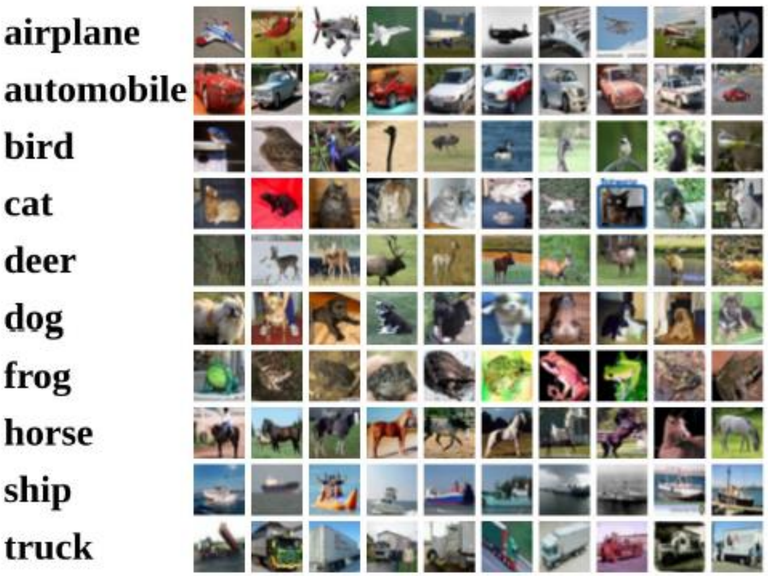
Linear Classifier

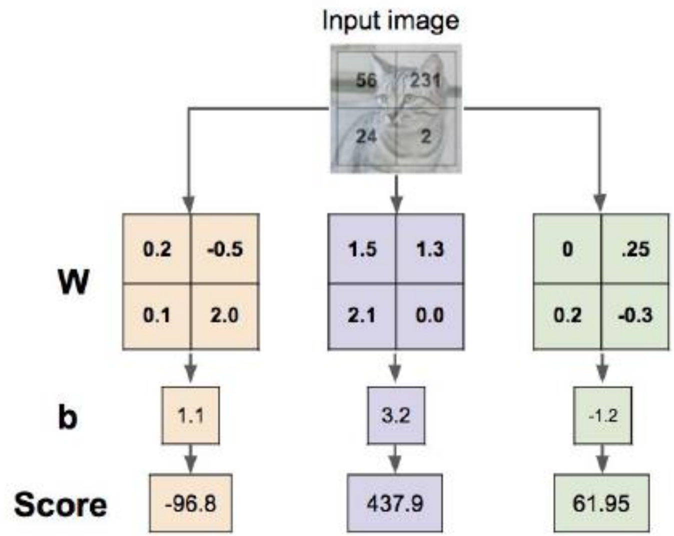


Linear Classifier

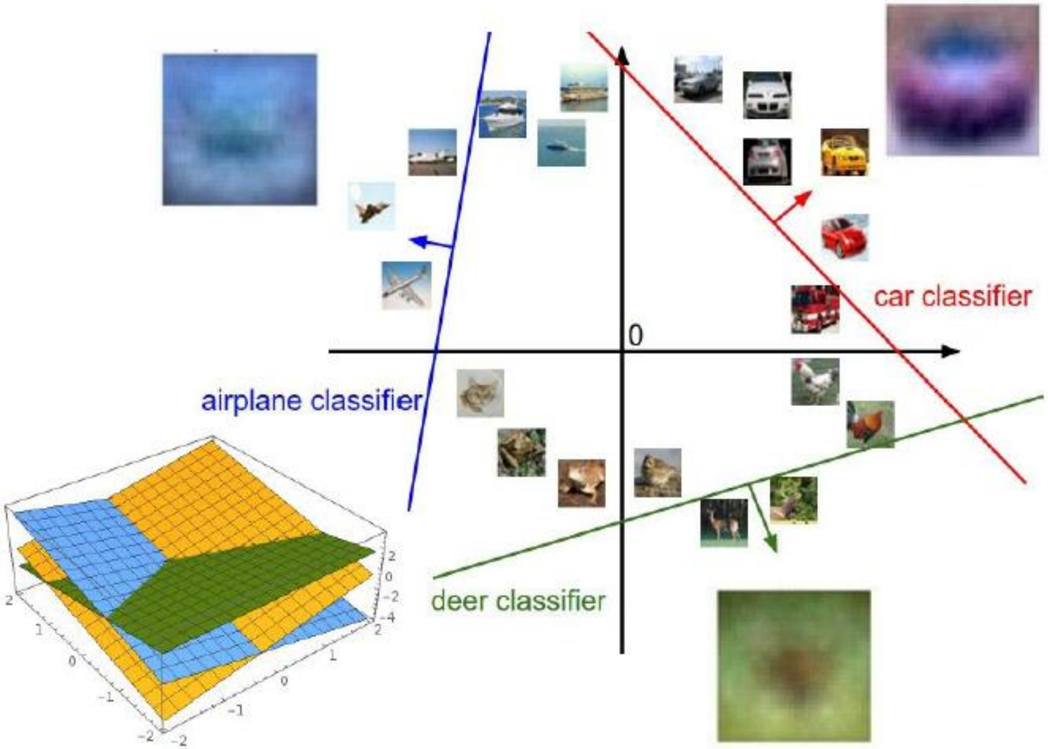


Linear Classifier





Linear Classifier



Plot created using Wolfram Cloud

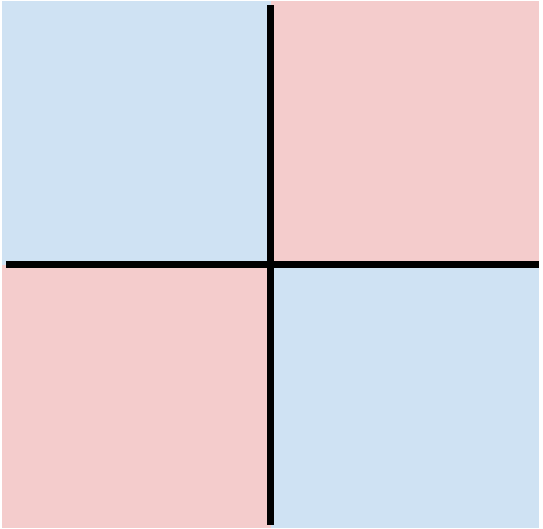
$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

Nonlinear examples

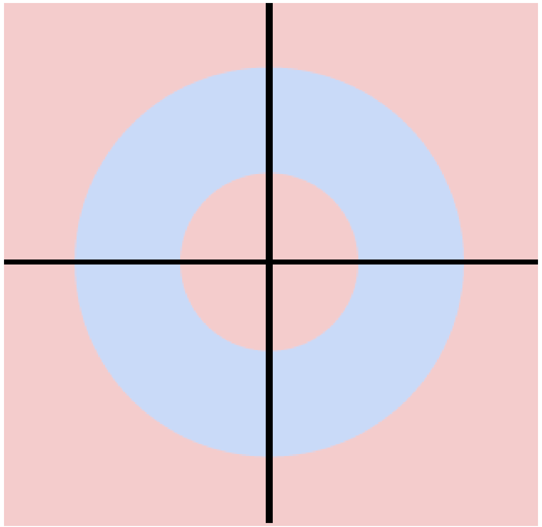
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



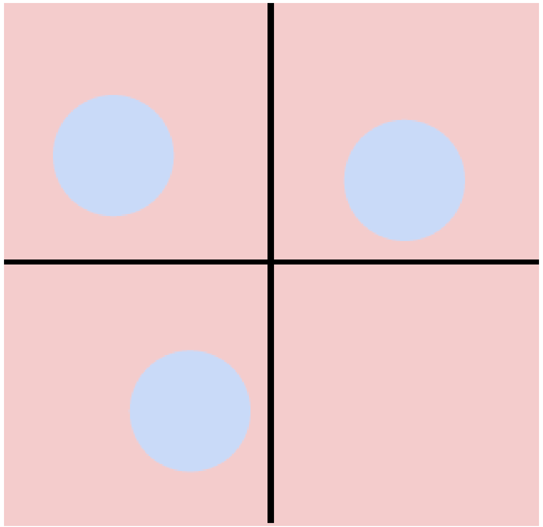
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else

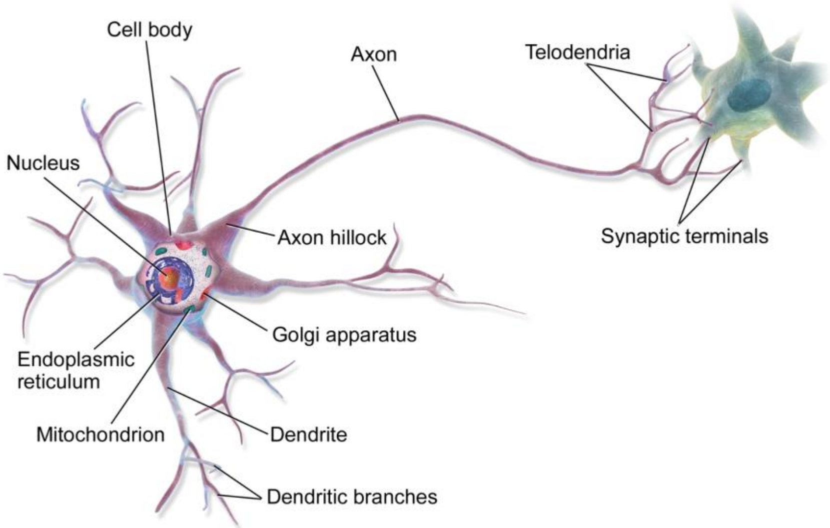


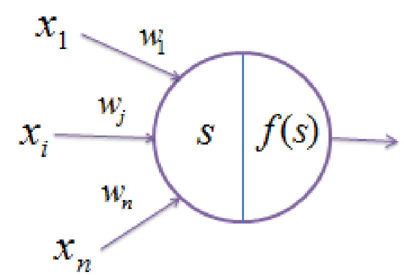
Class 1:
Three modes

Class 2:
Everything else



Introduction to Deep Learning



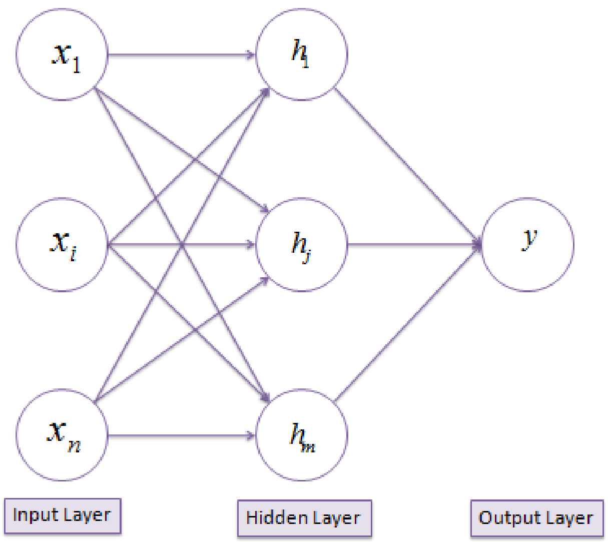


Summation

$$s = \sum w \cdot x$$

Transformation

$$f(s) = \frac{1}{1+e^{-s}}$$



References

- Cambridge Handbook of Artificial Intelligence
- Deep Learning Book - Ian Goodfellow, Yoshua Bengio and Aaron Courville
- CS231n: Convolutional Neural Networks for Visual Recognition (Stanford University)