

SCC361: Artificial Intelligence

Week 2: Features and Feature Extraction

Dr Alistair Baron

School of Computing and Communications, Lancaster University

Office: InfoLab21 C34 | Email: a.baron@lancaster.ac.uk | [Office Hours: Mon. & Wed. PM](#)

Slides from Dr. Bryan M. Williams



Lecture Plan

Weeks 1-2 (AB):

- Introduction to Artificial Intelligence and Machine Learning
- **Features in Machine Learning and Feature Extraction**

Weeks 3-6 (BW):

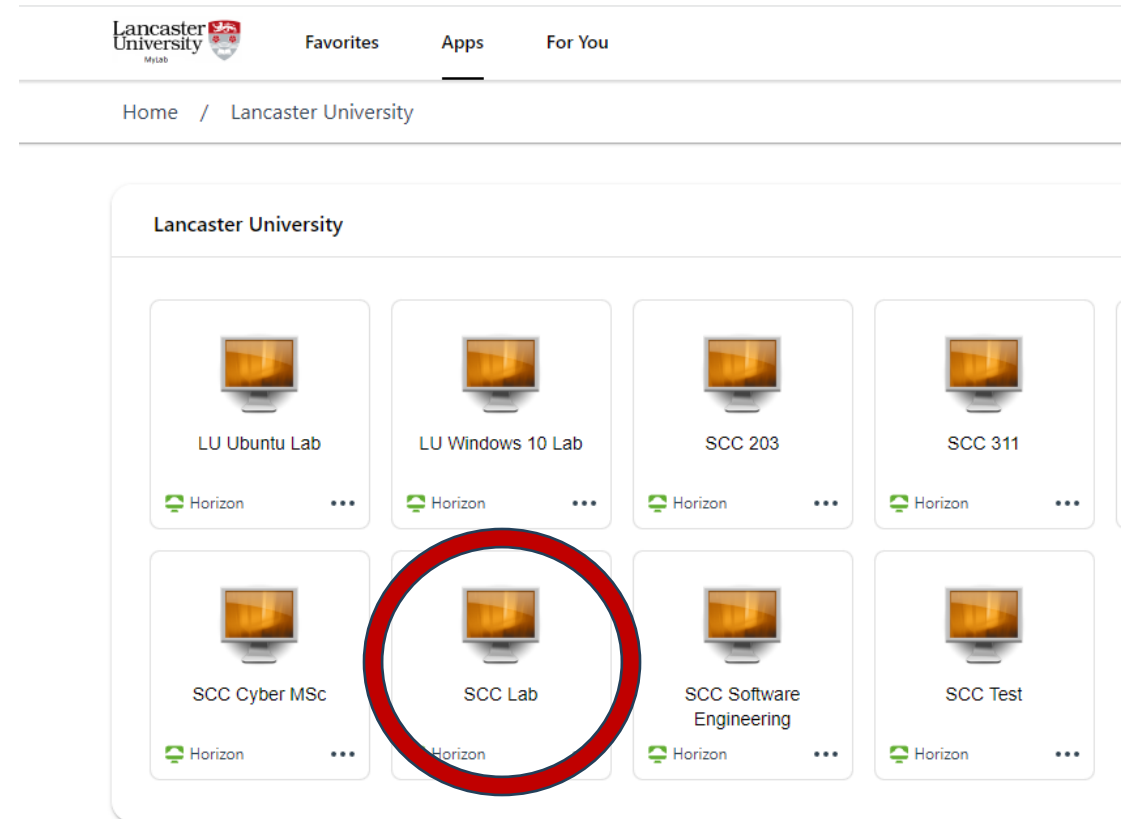
- Clustering and Classification
- Decision Trees
- Statistical approaches: Naïve Bayes
- Genetic Algorithms

Weeks 7-10 (HR):

- Artificial Neural Networks: MLP
- Current research in AI (AB & BW)
- Introduction to Deep Neural Networks
- Introduction to Convolutional Neural Networks
- Segmentation and Generative AI

Lab Sessions

- Purpose
 - Familiarity with MATLAB
 - Common AI and CV tasks
 - Smoothing, Pooling, Blurring, Edge Detection
- Solutions available on Moodle
 - .m / .mlx file / .pdf
- SCC Lab on mylab.lancaster.ac.uk
- Details for own install:
<https://portal.lancaster.ac.uk/ask/digital/software/directory/matlab/>



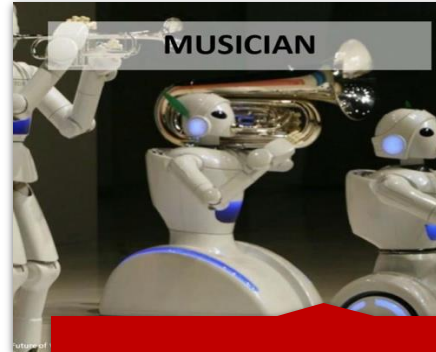
Week 1 Key Takeaways - Context



Real Life



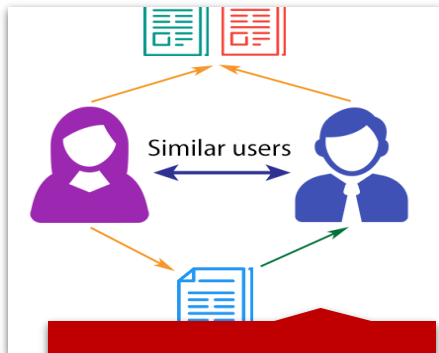
Movies



Music



Agriculture



NLP



Medicine



Security



Forensic
Investigation

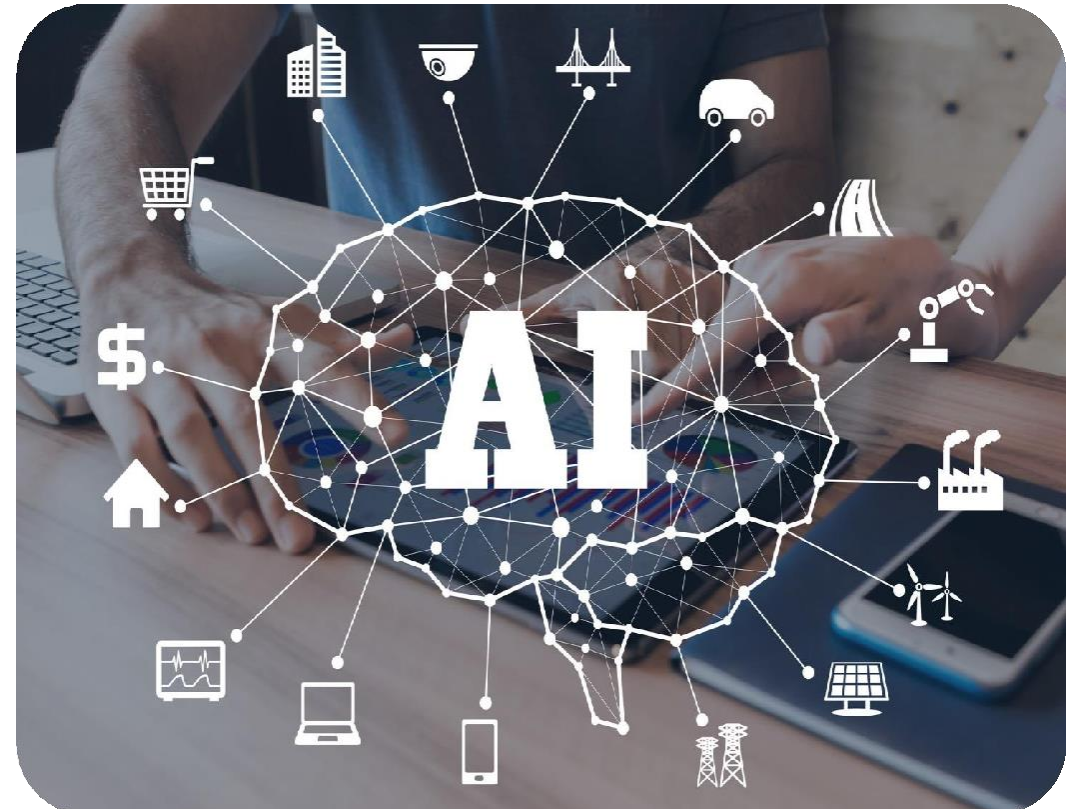
Definition of AI

A **computational agent** is an **agent** (something that **acts** within an environment) whose decisions and actions can be explained in terms of computation.

Artificial intelligence, or **AI** is the field that studies the *synthesis and analysis of **computational agents that act rationally***

AI Summary

- Artificial Intelligence: An overview
 - Application, history, foundations of AI
- Definition of AI
 - Rational-agent approach
- Goals of AI
- AI and the Society
 - Benefits
 - Risk and Challenges
 - Ethical Issues



Machine Learning

- Overview of Machine Learning
 - AI and ML, Definitions of ML, How to learn
- Types of Machine Learning
 - Supervised, unsupervised, semi-supervised
- Supervised Learning
 - Classification and regression
- Unsupervised Learning
 - Clustering and association



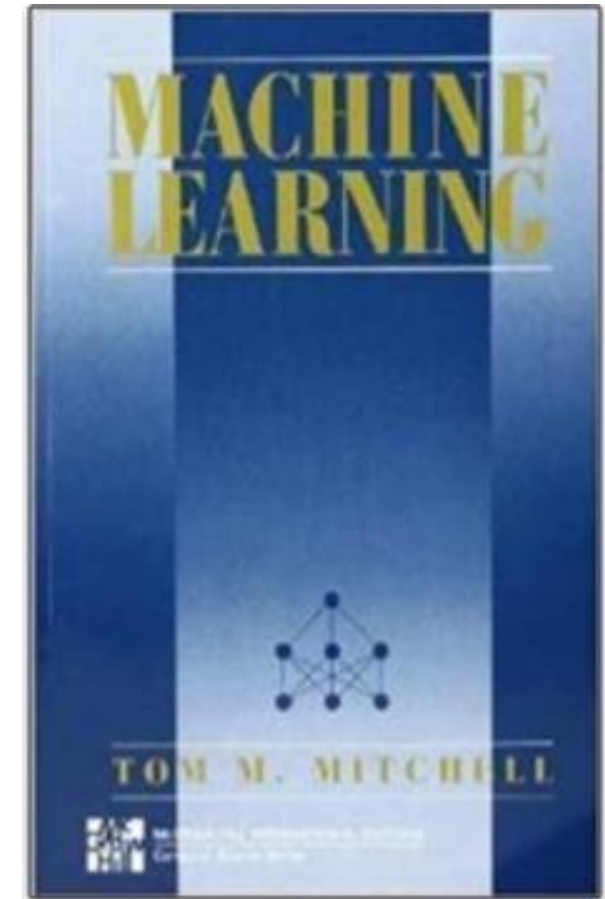
What is Machine Learning?

A popular definition:

*“A computer is said to **learn** from experience **E** with respect to task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improved with experience **E**”*

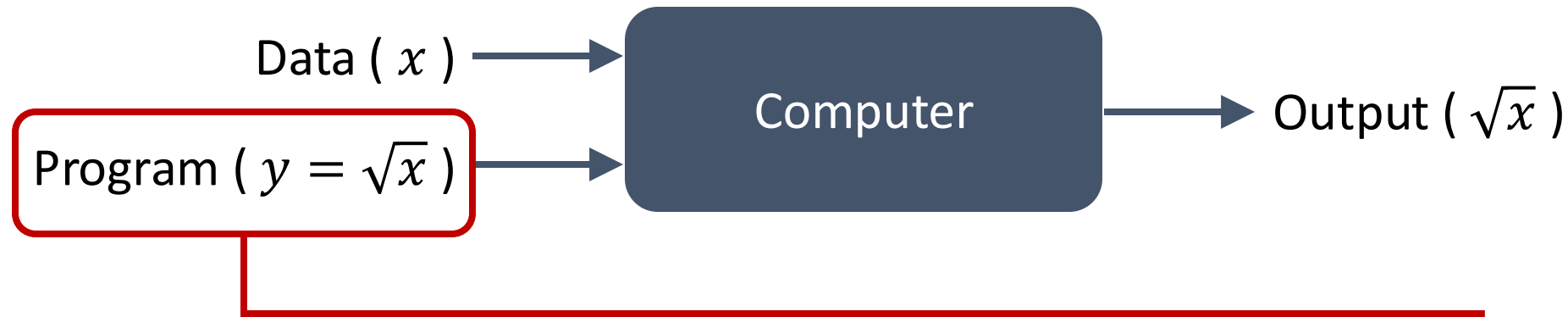
– Tom Mitchell (1997)

- Again, the key is learning from experience
- Not explicitly programmed

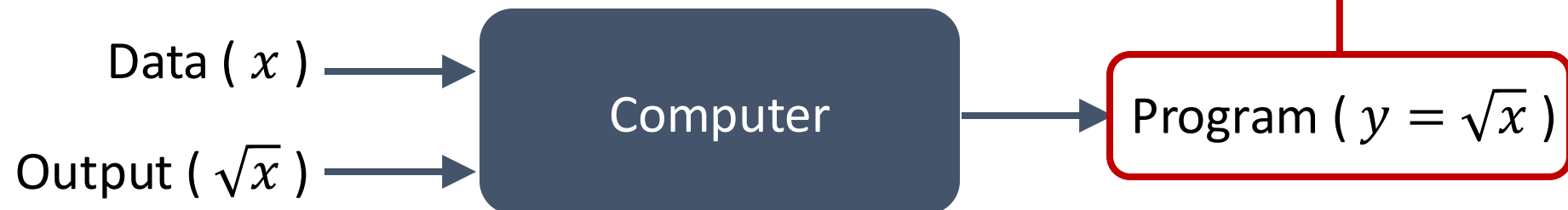


What is Machine Learning?

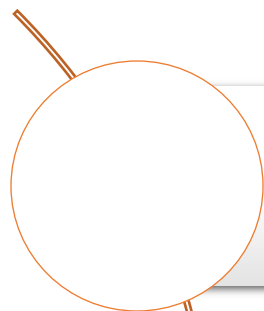
- Consider the function $y = f(x)$ (e.g. $f(x) = \sqrt{x}$)
- Traditional Programming (Software 1.0)



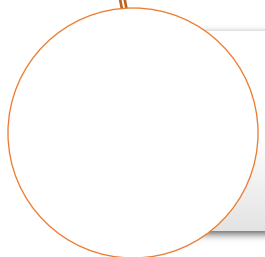
- Machine Learning (Software 2.0)



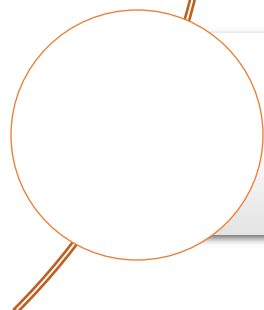
This Week



Machine Learning



Features in Machine Learning



Feature Extraction in Images and Text

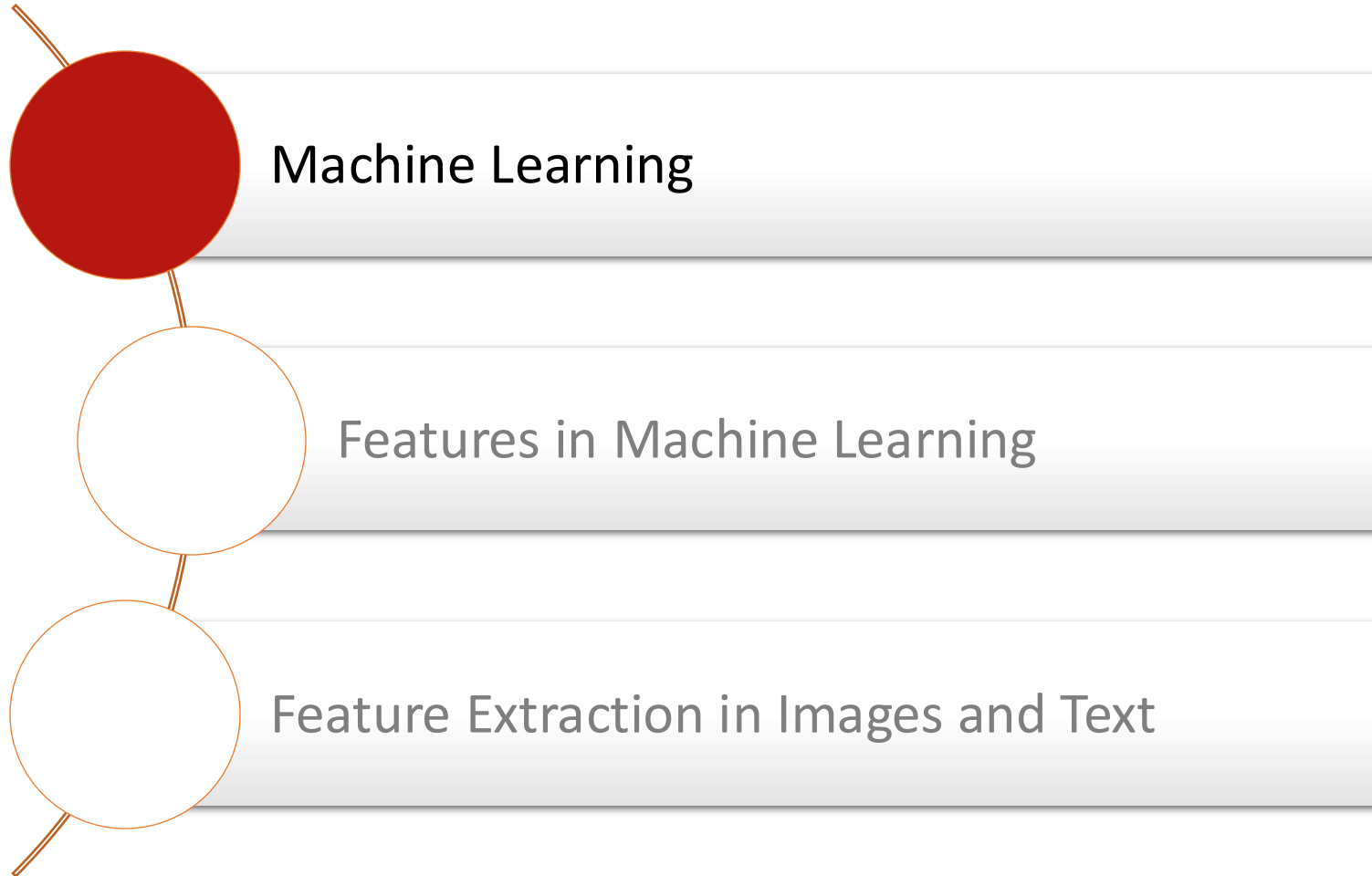


Expected Learning Outcomes

By the end of this week, **we will be able to:**

- Distinguish supervised and unsupervised machine learning.
- Understand what features and labels are.
- Define feature extraction.
- Understand the distinction between manual and automatic approaches to feature extraction.
- Understand processes of feature extraction in images and text.

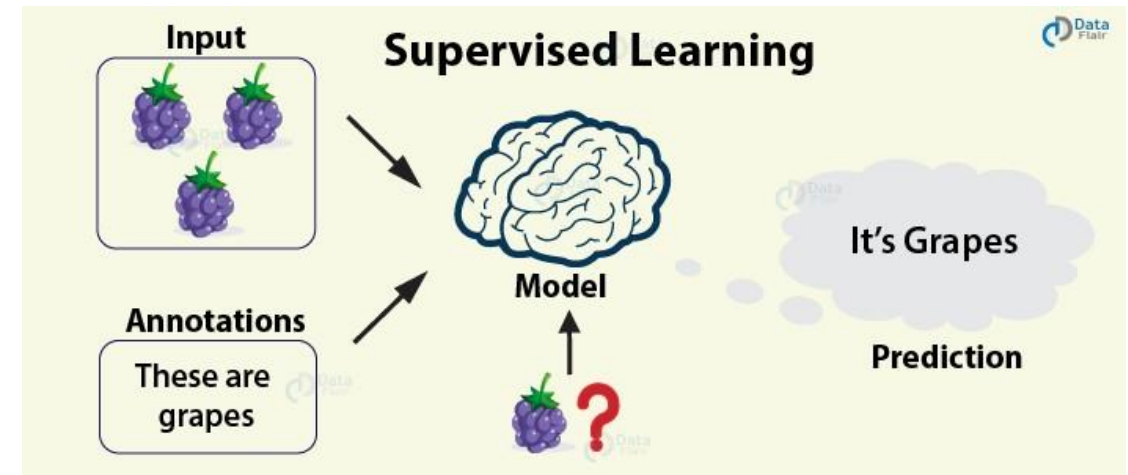
Now: Machine learning...



Types of Machine Learning

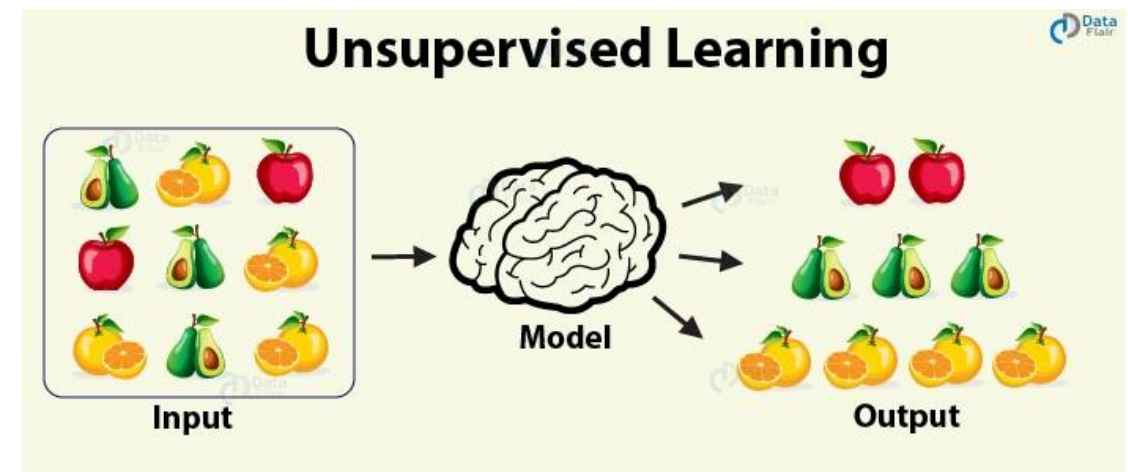
Supervised Learning

- Classification
- Regression



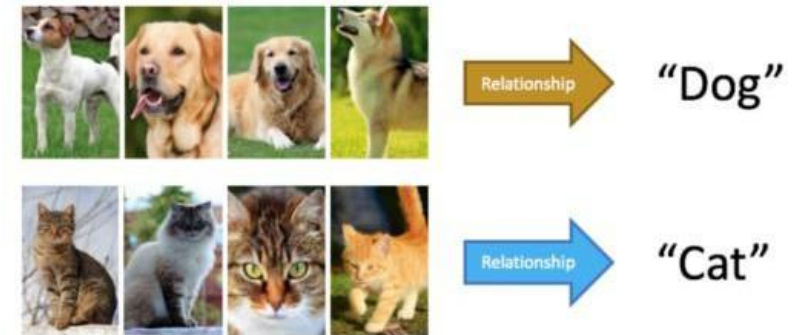
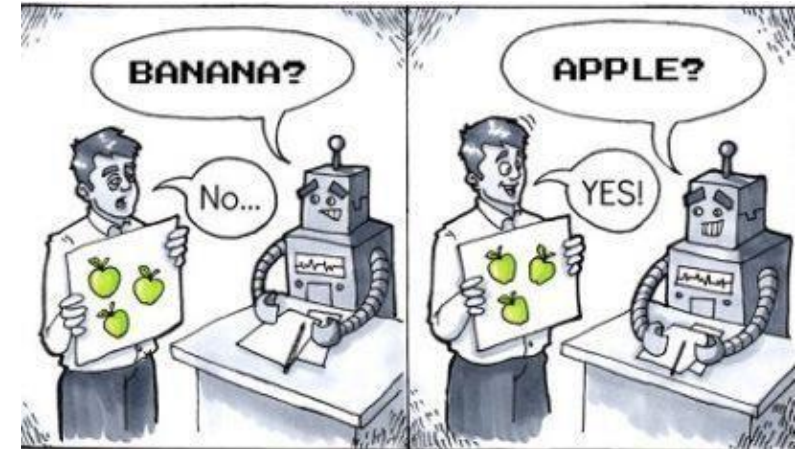
Unsupervised Learning

- Clustering
- Association

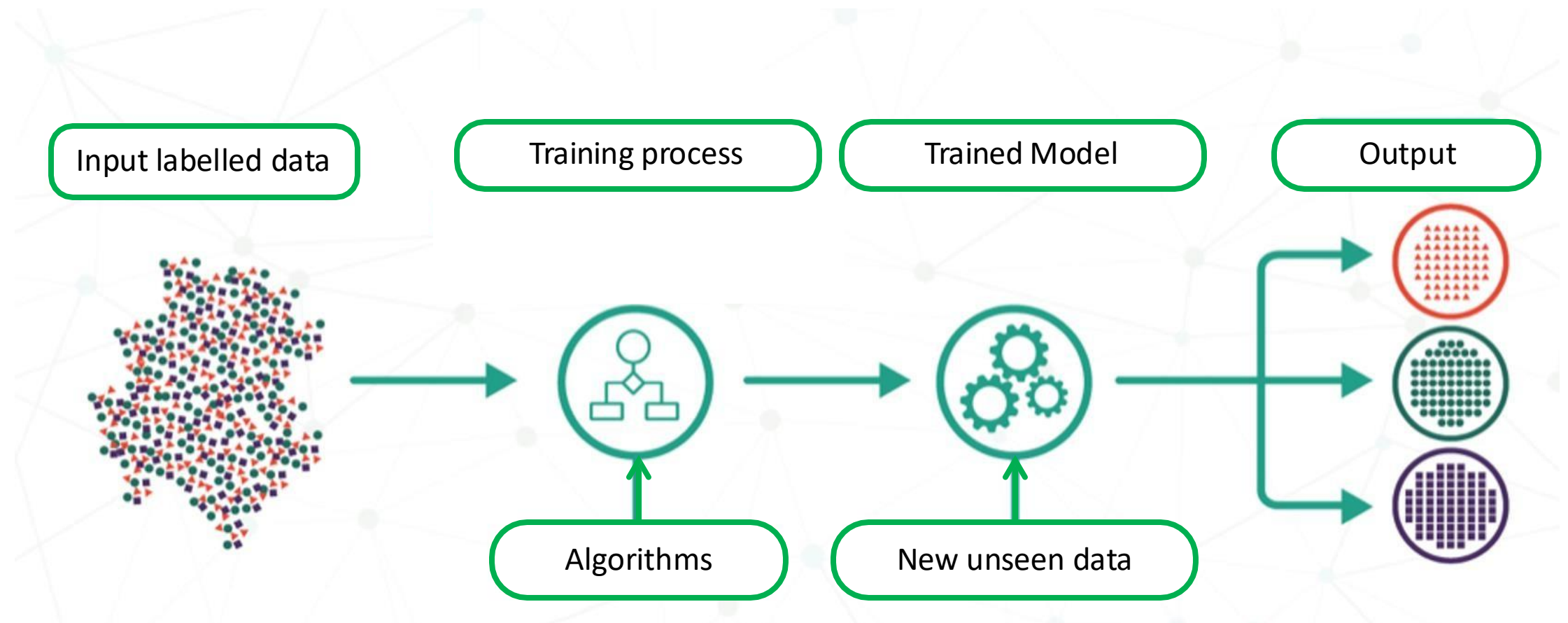


Supervised Learning

- The algorithm learns to map an **input** to a **particular output**.
- Instances of data are presented along with their **correctly labelled** output.
- Similar to a **teacher-student** scenario.
- The algorithm learns from **experience** to predict new unseen data.
- Two broad categories:
 - Regression
 - Classification

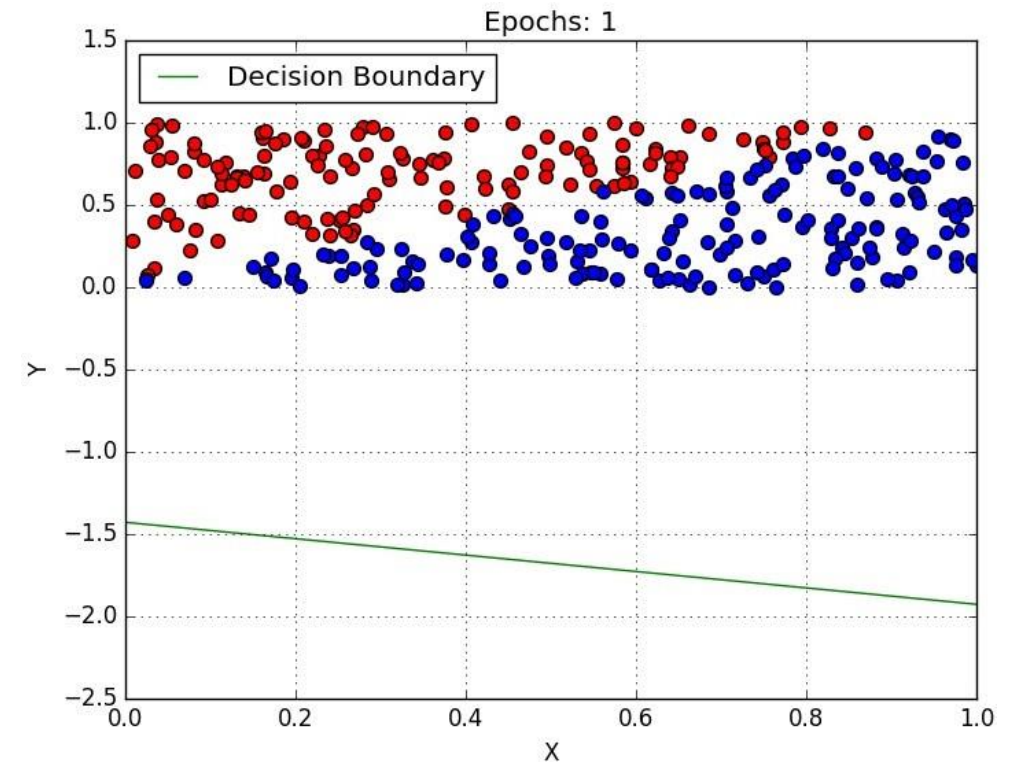
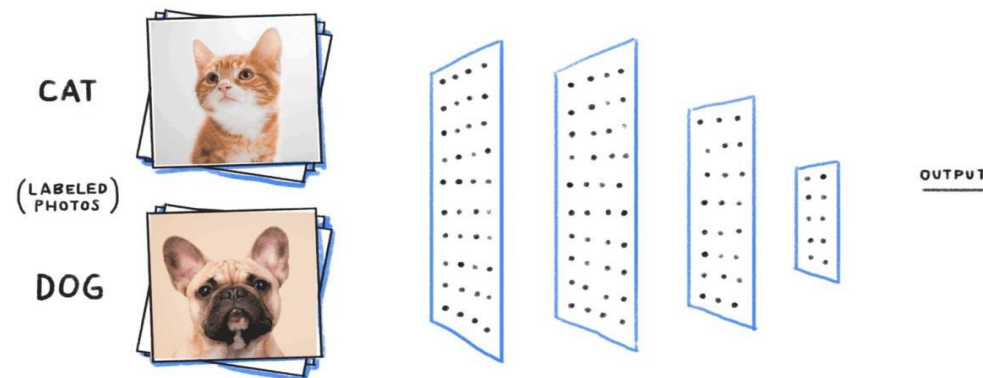


Supervised Learning



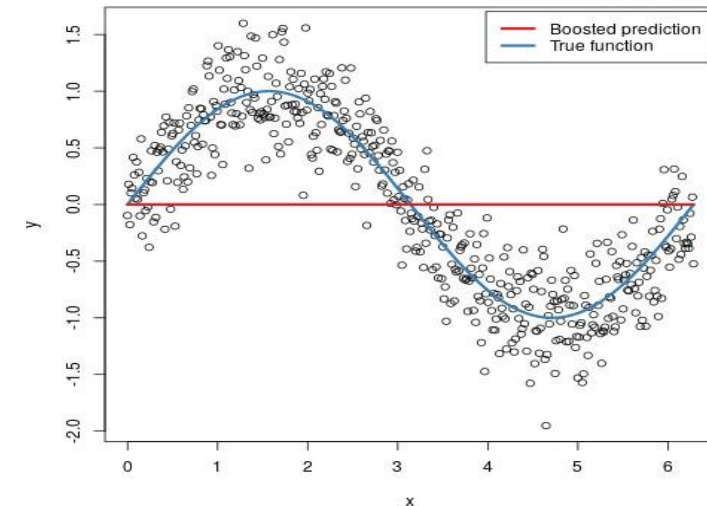
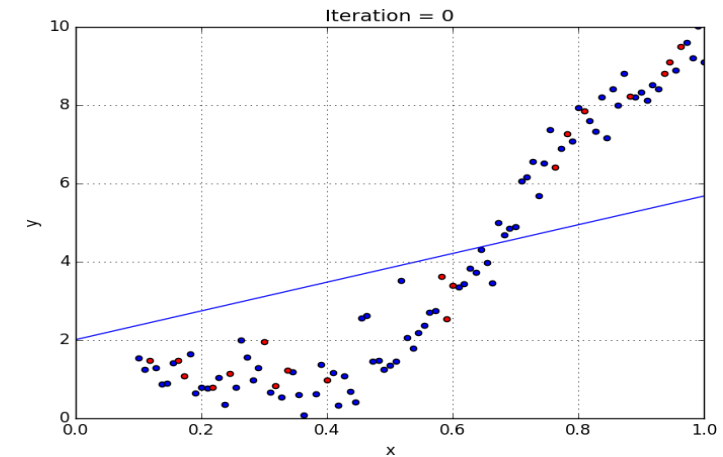
Classification

- Learns from labelled data (supervised)
- Predicts a **category** or a **class**
 - Cats|Dogs
 - Spam|Ham
 - Cancer|Not Cancer
- Attempts to separate the data into specific categories (or classes or labels)



Regression

- Learns from labelled data (supervised)
- Predicts a **continuous-valued output**
 - height, price, duration etc.
- Consider a function $y = f(x)$
 - we want our model to predict y_i given x_i
 - x_i not seen during training
- Typically fits some linear or quadratic curve of the data plot
- Linear or logistic regression algorithms are often used

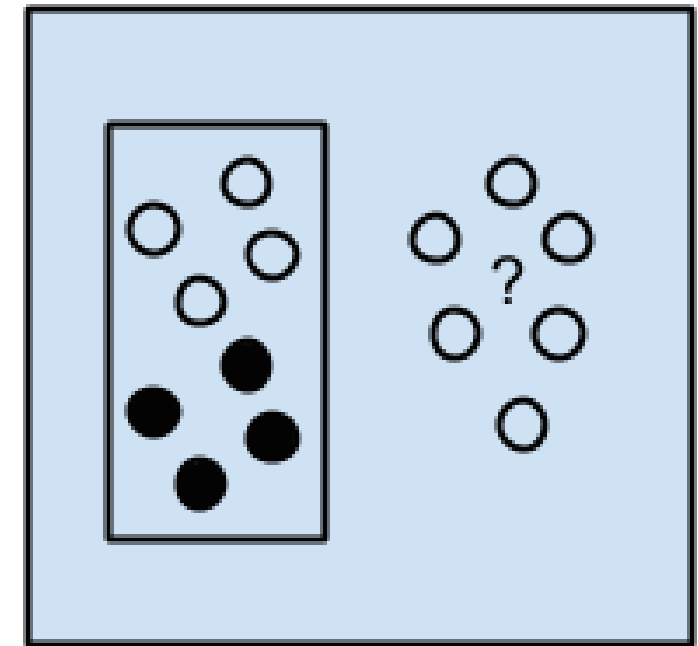


Quiz: Classification vs Regression

- If we wish to learn models to address the following
 1. Predict how many students will enrol in this module in the next 3 years given the past enrolment data
 2. Predict whether a student will pass the module given previous years records
- How should we proceed
 - a. Both are regression problems
 - b. Both are classification problems
 - c. Problem 1 is regression while Problem 2 is classification
 - d. Problem 2 is regression while Problem 1 is classification

Supervised Learning Algorithms

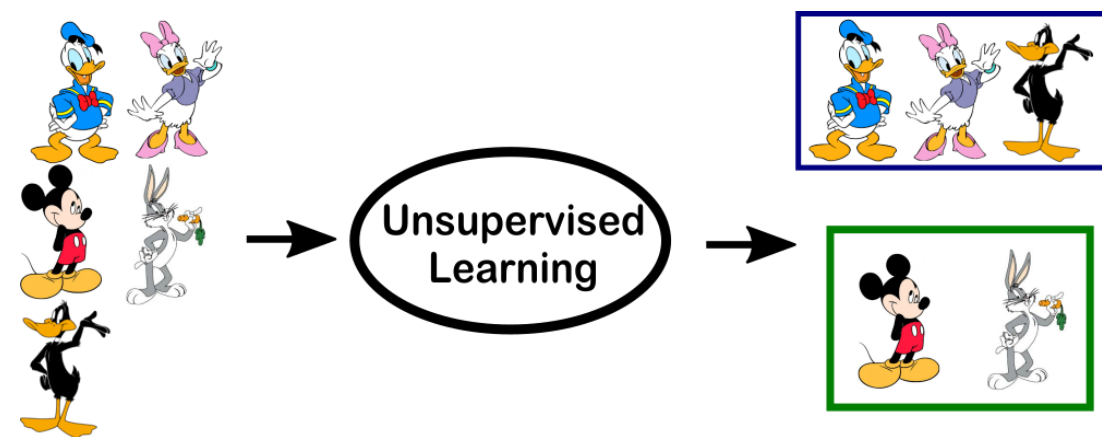
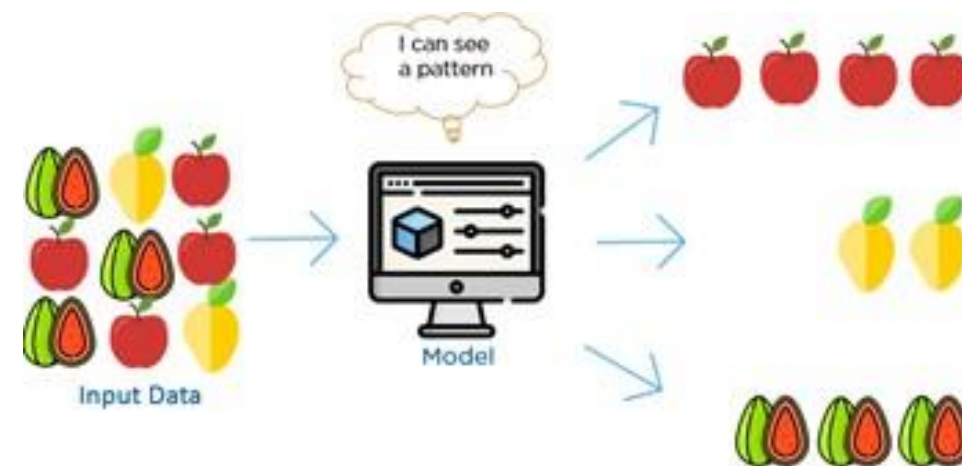
- Input data = training data
 - with labels e.g. spam/ham or stock price at t
- In training
 - the model makes a prediction and is corrected if the prediction is wrong
- Training process continues until a desired accuracy is achieved
- Problem types: Classification and Regression
- Example algorithms:
 - Logistic Regression, Back Propagation Neural Network, SVM



Supervised Learning
Algorithms

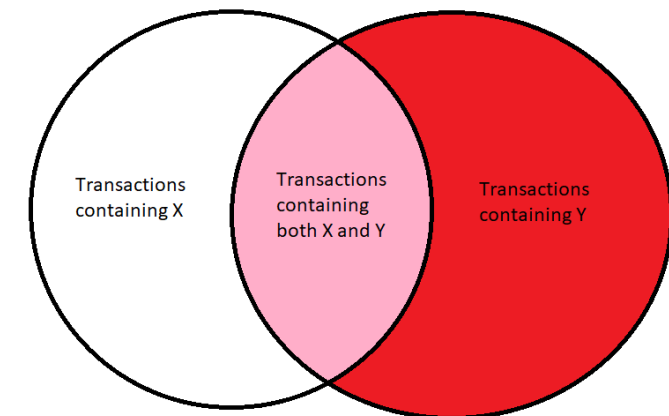
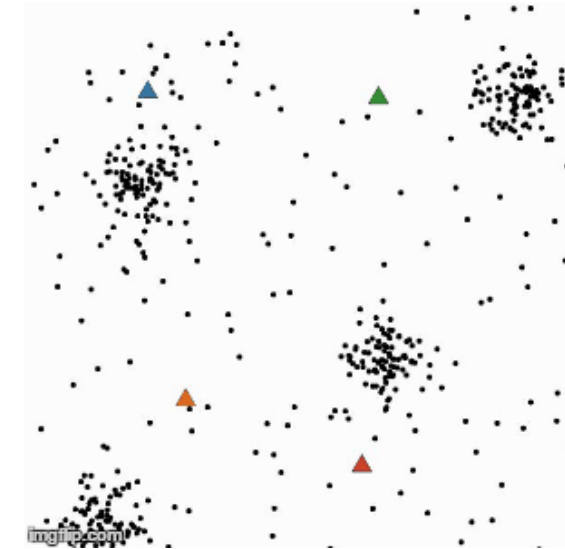
Unsupervised Learning

- Remember the function $y = f(x)$
- With unsupervised learning, only the input data, x , is available
- There are no corresponding labels (classes or categories) i.e. no output variable, y
- Aims at modelling the underlying structure of the data
- Two main categories:
 - Clustering
 - Association



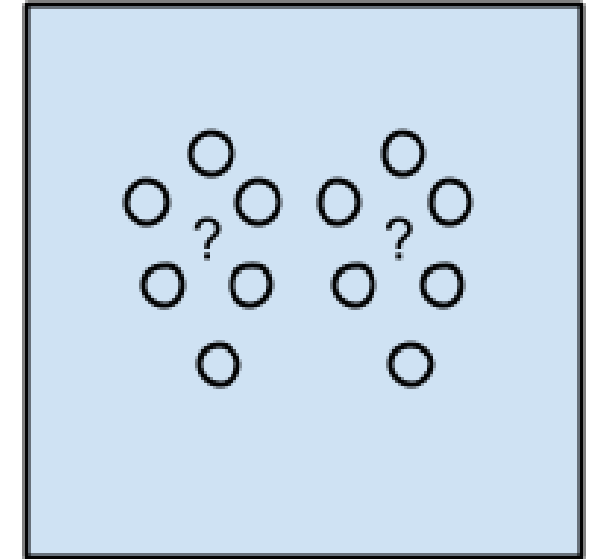
Clustering and Association

- In a **clustering** problem, we want to discover the inherent groupings in the data:
 - e.g. grouping customers by purchasing behaviour.
- In an **association** rule learning problem, we want to discover rules that describe large portions of your data
 - e.g. people that buy X also tend to buy Y



Unsupervised Learning Algorithms

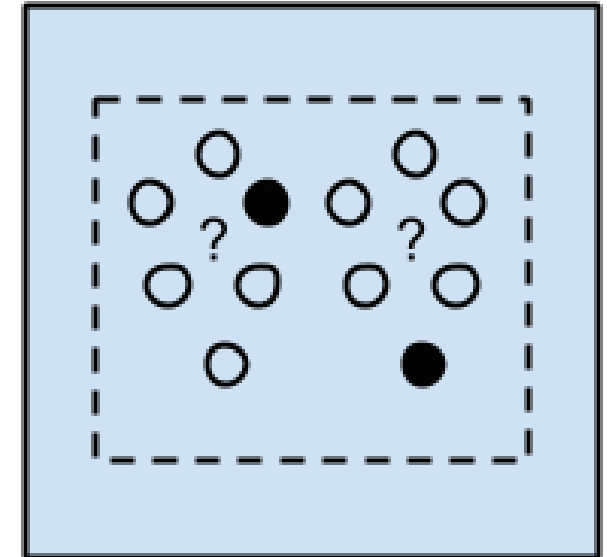
- Input data is not labelled
 - Output not known
- In training
 - Deduces structures present in the input data
 - Extracting general rules, reducing redundancy or organise data by similarity
- Problem types: clustering, dimensionality reduction, and association rule learning
- Example algorithms:
 - K-Means algorithm, PCA, Apriori algorithm



Unsupervised Learning
Algorithms

Semi-supervised Learning

- Semi-supervised learning approach refers to:
 - when we have a large amount of input data (X) but **only some** of the data is labelled (Y)
 - e.g. a photo archive where only some of the images are labelled (e.g. *dog*, *cat*), and the majority are unlabelled.
- Many real world problems adopt this method
 - It can be expensive or time-consuming to label data
 - A hybrid design often helps to bridge the gaps
- Algorithms:
 - A flexible combination of supervised and unsupervised algorithms

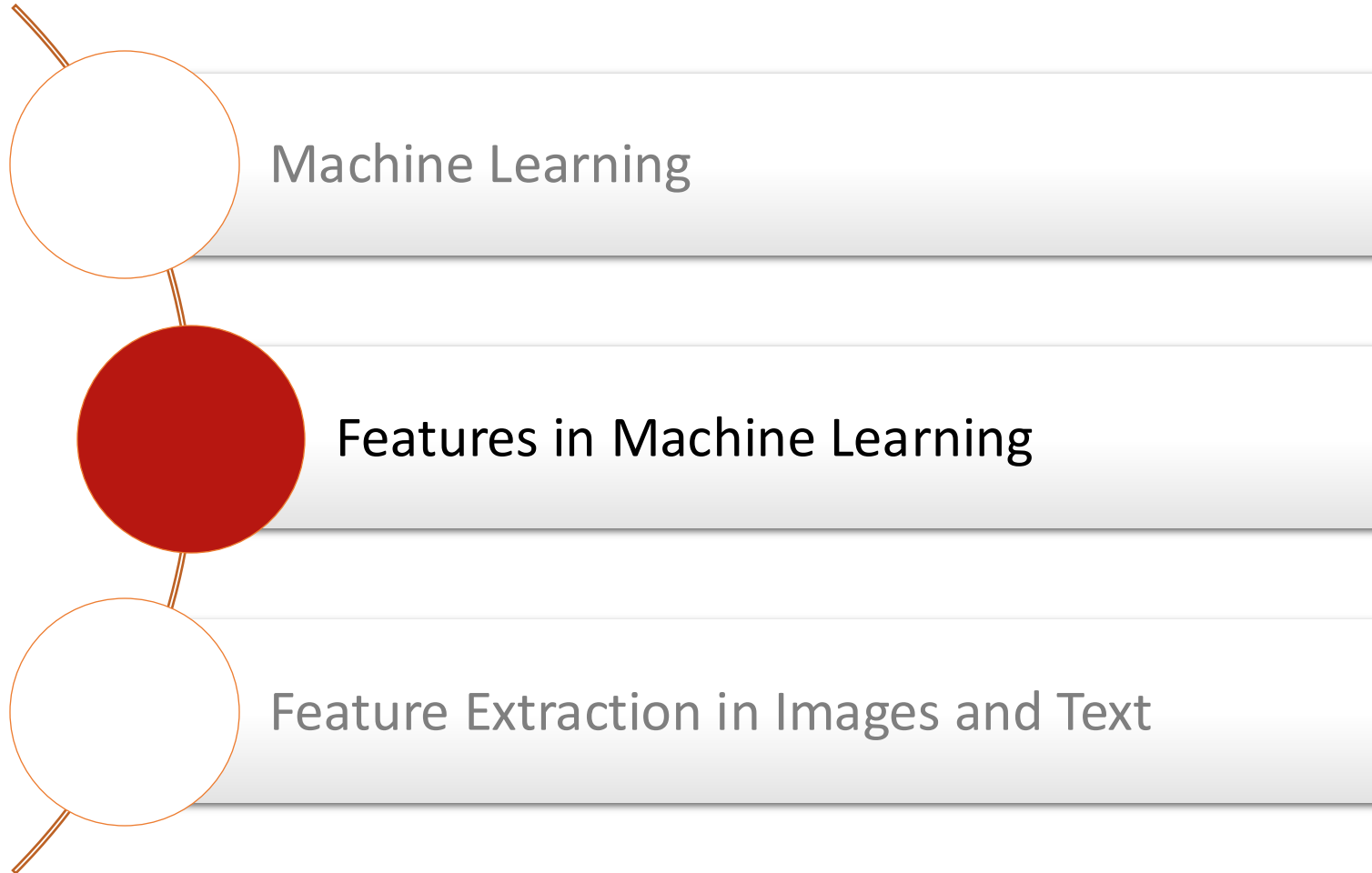


Semi-supervised Learning Algorithms

Machine Learning Summary

- Overview of Machine Learning
 - AI and ML, Definitions of ML, How to learn
- Types of Machine Learning
 - Supervised, unsupervised, semi-supervised
- Supervised Learning
 - Classification and regression
- Unsupervised Learning
 - Clustering and association

Next: Features in Machine Learning



Features in Machine Learning

- Machine learning models are dependent on data
- Machine learning algorithms are designed to understand numbers
- How can real-world data be interpreted as numbers?
 - e.g. Images, video, text



Features and Labels

Feature:

A **feature** is an individual measurable property or characteristic of a phenomenon being observed.

A set of features represent the information you can extract from data.

Label:

The label is the tag you wish to assign to a set of features



Animals

Features and Labels – text as layers

The cat sat on the mat.

the,cat,sat,on,the,mat

AT,NN1,VVD,II,AT,NN1

t,h,e, ,c,a,t, ,s,a,t, ,o,n, ,t,[..]

^t,th,he,e , c,ca,at,t , o,on,n ,[..]

^^t,^th,the,he ,e c, ca,cat,at ,[..]

(input)

(words)

(POS)

(char unigram)

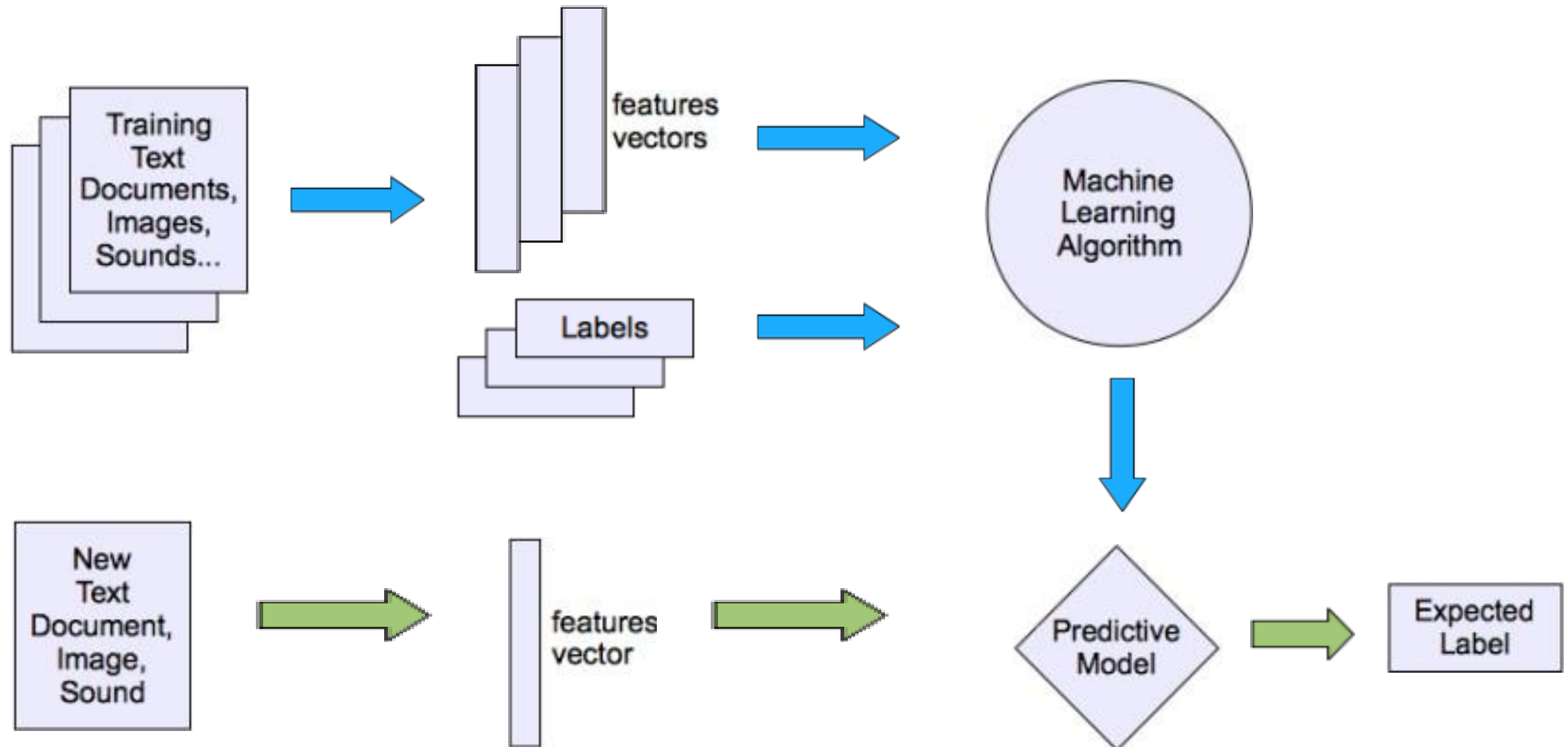
(char bigram)

(char trigram)

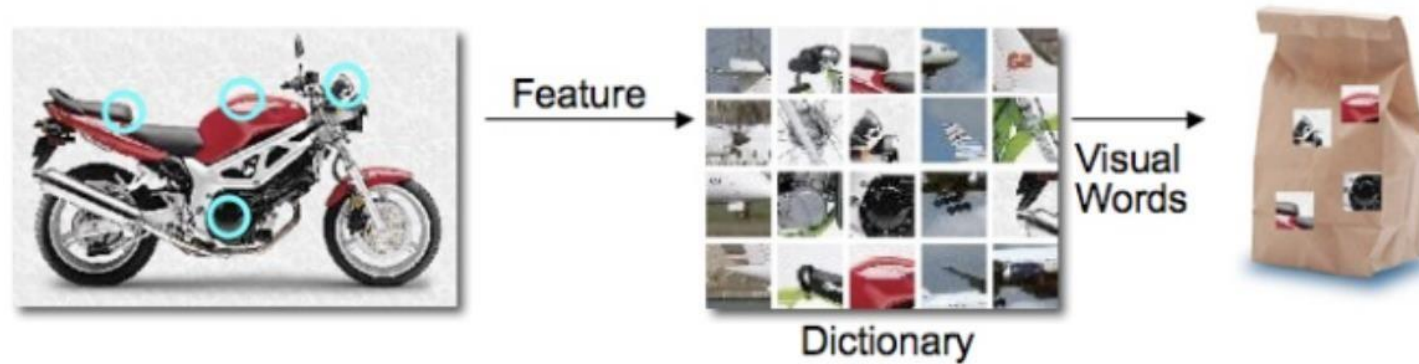
- Word n-grams
- Word lengths
- Sentence length
- Syllables per word
- Topics
- ...



Features for ML



Feature Extraction



- A set of methods that map input features to new output features
- Any technique that transform raw data into features that can be used as input to a learning algorithm

Feature Extraction



- A set of methods that map input features to new output features
- Any technique that transform raw data into features that can be used as input to a learning algorithm
- The process of transforming **raw data** into **numerical features** that can be processed while **preserving** the information in the original data set.

Why Feature Extraction?

- Learning algorithms (and computers) prefer data in numeric format



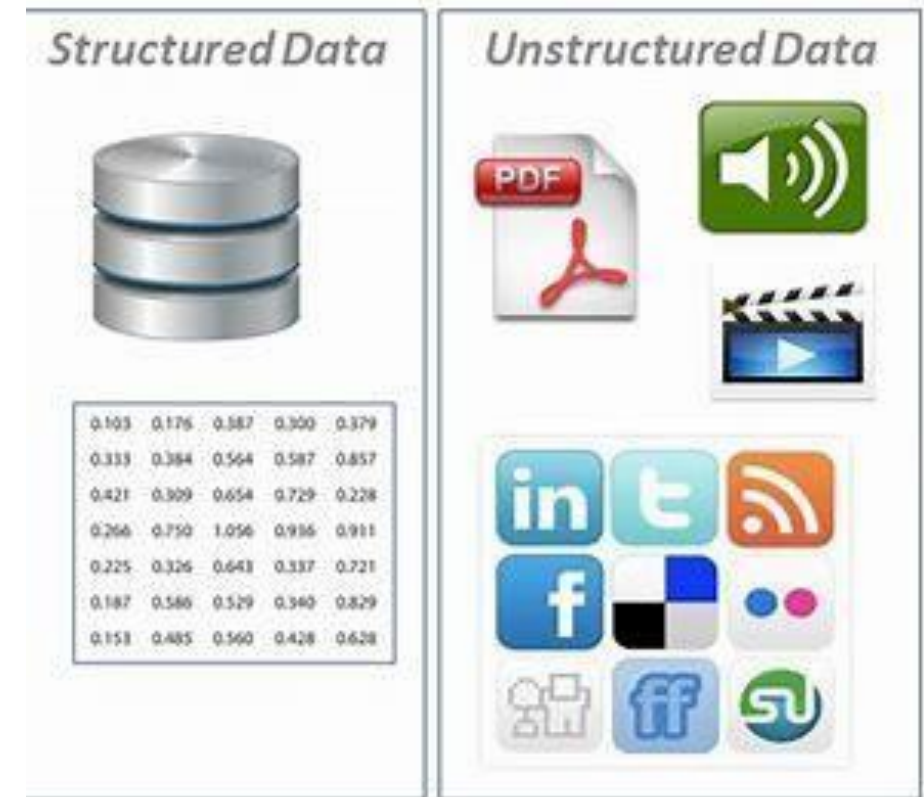
Why Feature Extraction?

- Learning algorithms (and computers) prefer data in numeric format
- Real-world data are often generated in non-numeric format:
 - text, image, audio, video, etc.



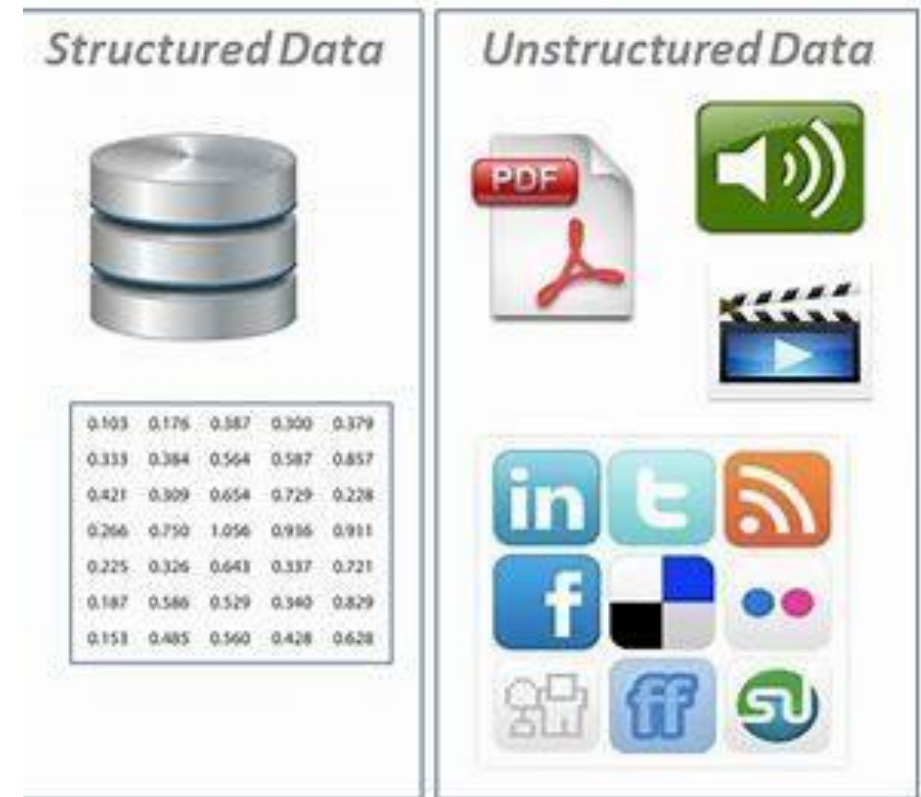
Why Feature Extraction?

- Learning algorithms (and computers) prefer data in numeric format
- Real-world data are often generated in non-numeric format:
 - text, image, audio, video, etc.
- Real-world data are mostly unstructured



Why Feature Extraction?

- Learning algorithms (and computers) prefer data in numeric format
- Real-world data are often generated in non-numeric format:
 - text, image, audio, video, etc.
- Real-world data are mostly unstructured
- Increases learning accuracy by extracting the most significant features



Examples of Feature Extraction

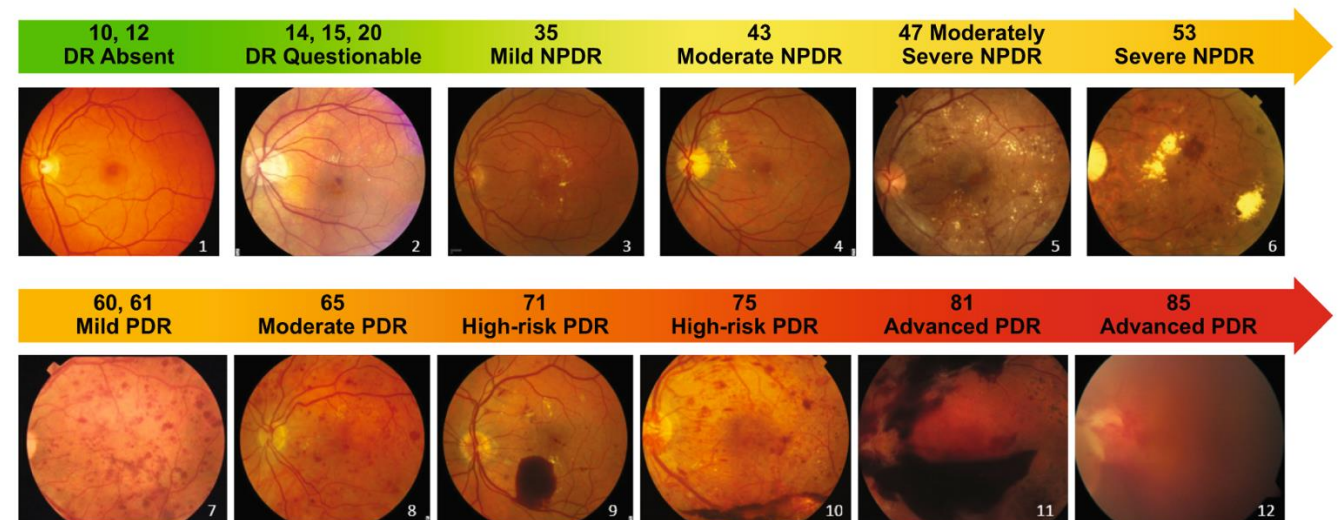
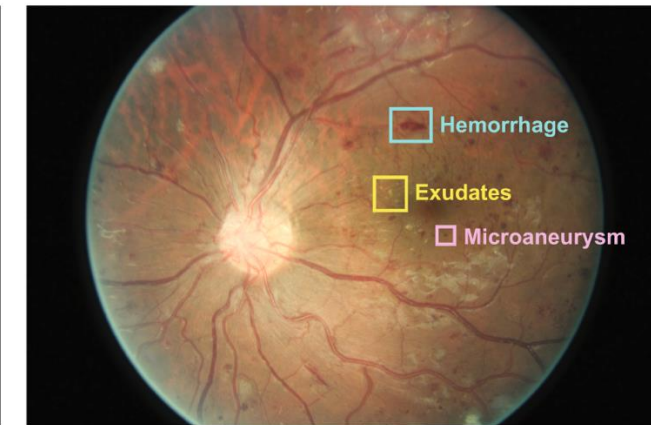
What are the features and labels?

- Image type?
- Patient Diagnosis?
- Image Diagnosis?
- Pathology?
- Anatomy?

Patient without DR



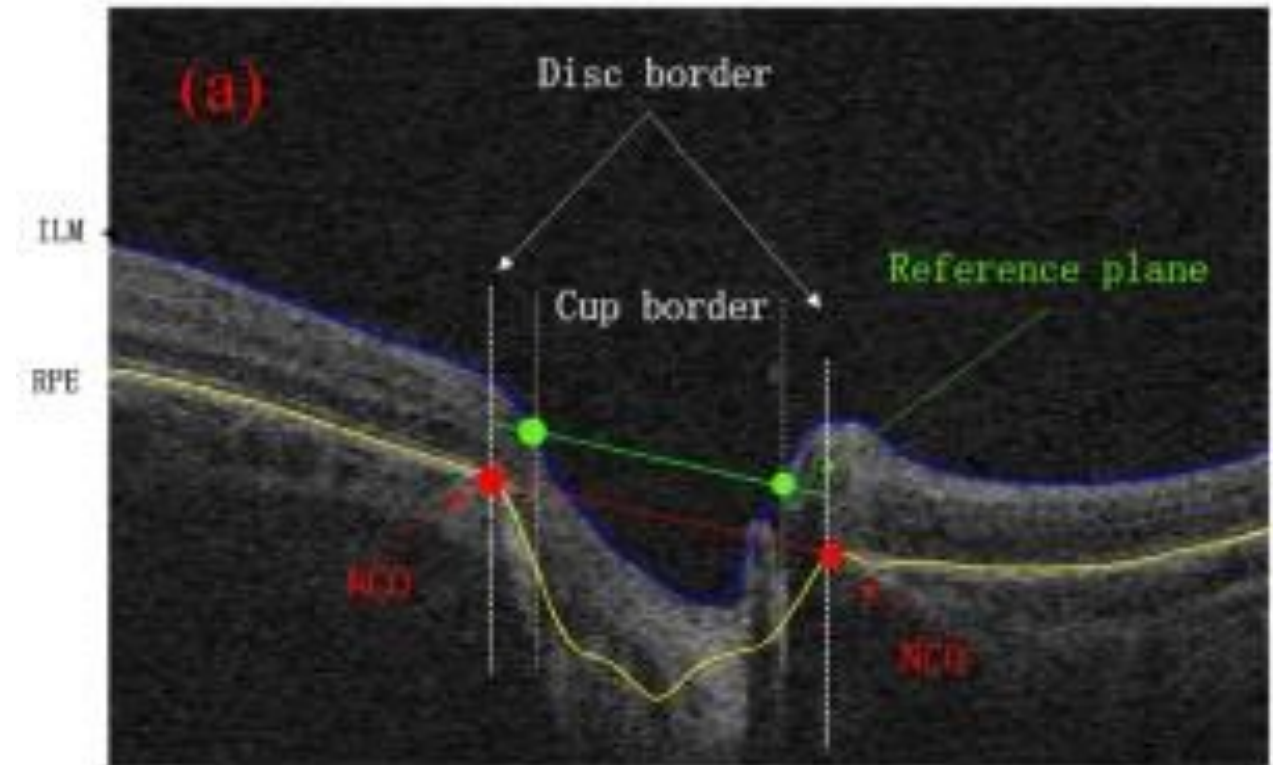
Patient with DR



Examples of Feature Extraction

What type of label do you need?

- Dimension?
- Data type?
- Number of labels?



Examples of Feature Extraction

What is the label?

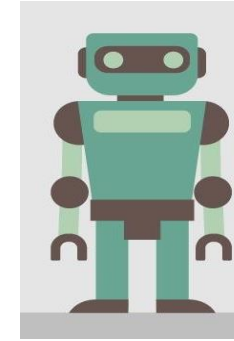
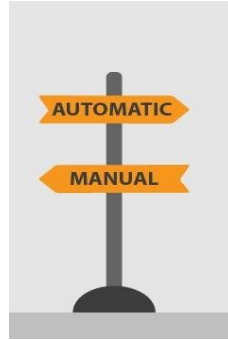


Approaches to Feature Extraction



Manual:

- Manually identifying and describing relevant features for a given problem and implementing a way to extract those features.
- Mostly requires experience or domain knowledge
- e.g. computing the mean of a window in signal processing



Automated:

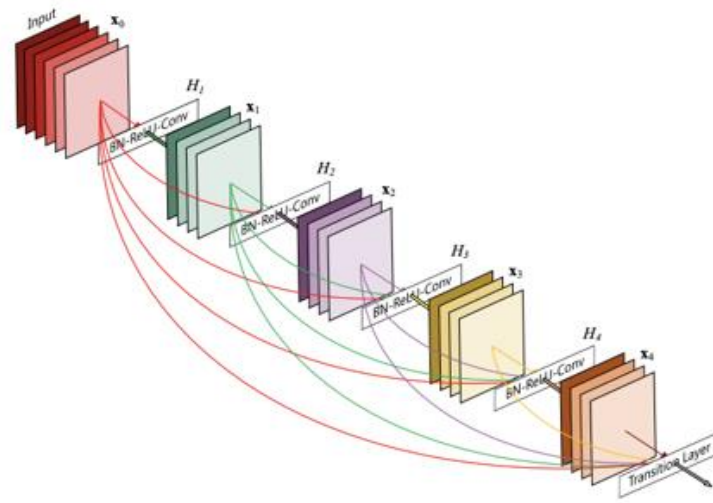
- uses specialized algorithms to extract features automatically from signals or images without the need for human intervention
- More efficient for raw data transformation
- e.g. Wavelet scattering

Feature Vector Example

How can we compare these images?

Look for meaningful attributes?

High-dimensional encoding?

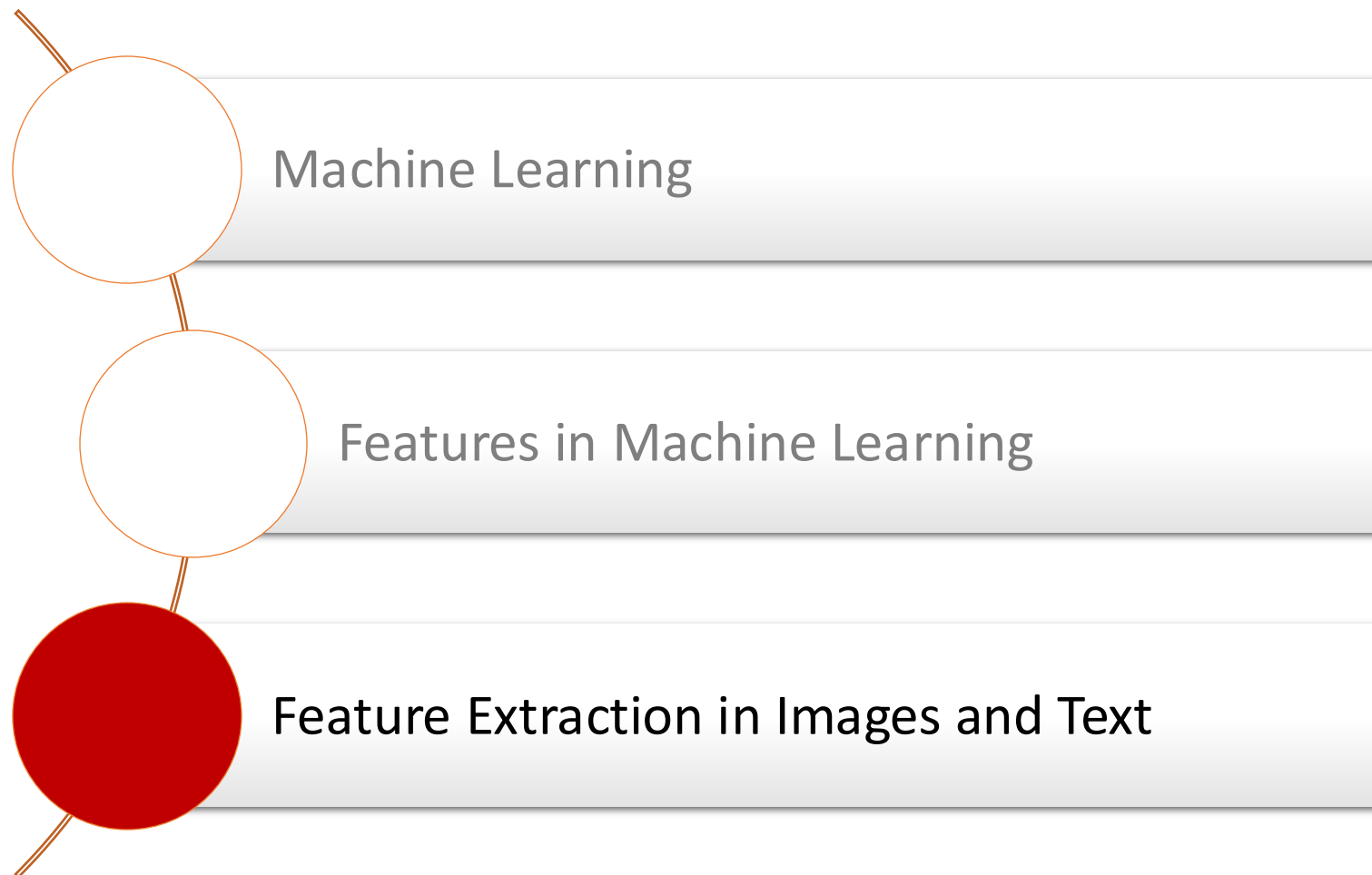


Definition

A **feature** is an individual measurable property or characteristic of a phenomenon.

A **feature vector** represents numeric or symbolic characteristics, called features, of an object in a mathematical analysable way

Up next: Feature Extraction



Feature Extraction from Text

- How do we extract features from text?
- Text data = a sequence of words
- Words are made up of characters
- Characters are represented with ASCII or Unicode
- The machine does not really make any meaning from that
- **So, how do we encode meaning representations from text?**



A	B	C
⋮	⋮	⋮
65	66	67
cat	sat	on
⋮	⋮	⋮
?	?	?

1-Hot Vector

- **Vocabulary**, $V = \{the, cat, sat, on, mat\}$
- Each **word** is assigned a **vector** of size $|V|$:
 - 1: for corresponding word in V
 - 0: everywhere else
- **Problems:**
 - Very sparse vectors
 - No notion of similarity
 - $|V|$ can be 10s or 100s of thousands
 - Word not in V ?
 - Fixed length representation

word	Features (vocabulary)
the	[1 0 0 0 0]
cat	[0 1 0 0 0]
sat	[0 0 1 0 0]
on	[0 0 0 1 0]
the	[1 0 0 0 0]
mat	[0 0 0 0 1]

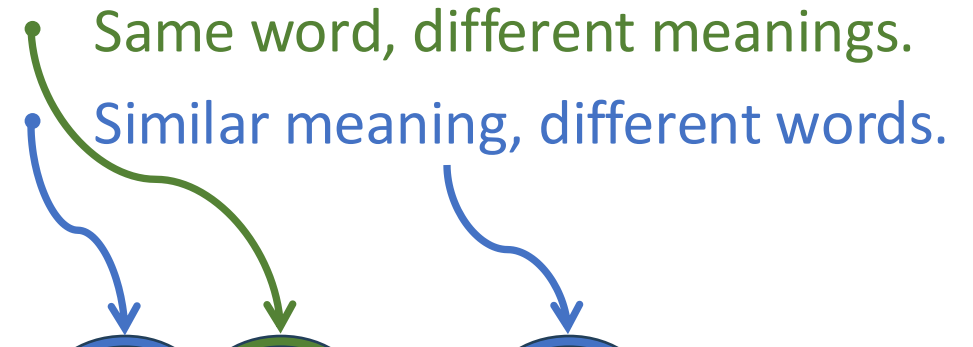
the cat sat on the mat: [1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1]

Bag of Words (BoW)

- Each **document** represented as an unordered collection of words.
- Document: a unit of text to be analysed
 - Sentence, paragraph, email, book, etc.
- BoW: The **frequency** of words in **vocabulary** in each document.

- **Problems:**

- Order of words not preserved.
- Counts not normalised.
- Common words dominate
- Same word, different meanings.
- Similar meaning, different words.



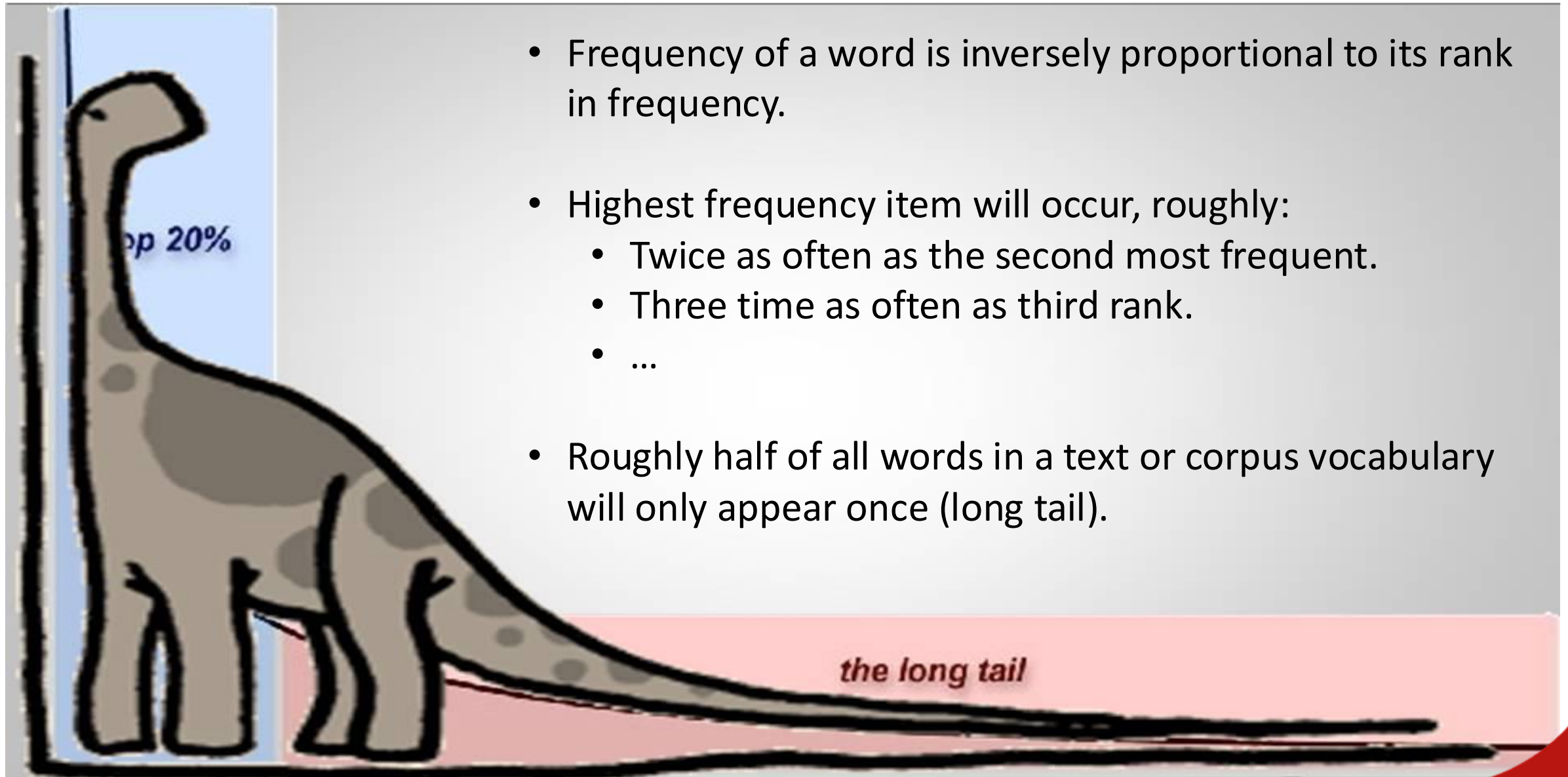
document	the	boy	liked	book	i	like	flight	will
the boy liked the book	2	1	1	1	0	0	0	0
i like the flight	1	0	0	0	1	1	1	0
the boy will book the flight	2	1	0	1	0	0	1	1

Bag of Words (BoW) – n-grams

- To preserve some order, **count n-grams**
 - word pairs (bigrams), or triples (trigrams).
- **Problems:**
 - Even more features
 - More sparsity

document	the boy	boy liked	liked the	the book	i like	like the	the flight	...
the boy liked the book	1	1	1	1	0	0	0	...
i like the flight	0	0	0	0	1	1	1	...
the boy will book the flight	1	0	0	0	0	0	1	...

Zipfian distribution



Reducing number of features

- Test normalisation:
 - US, usa, U.S.A. → USA
 - like, likes, liked → like
 - Stemming, lemmatisation, spelling normalisation, casing
- Remove high frequency words?
 - Stop words: articles, conjunctions, prepositions (e.g. and, a, the)
- Remove low frequency words?
 - Typos, rare words
- Medium frequency words, that distinguish documents:
 - The features we want

Term Frequency (TF)

- Term frequency $tf(t, d)$
 - Frequency of term (or n-gram) t in document d
- Variants:
 - Raw frequency: $f_{t,d}$ $f_{boy,d1} = 1$ (MATLAB default)
 - Binary $f_{t,d} > 0 : 1, else: 0$ 1
 - Relative frequency $f_{t,d} / \sum_{t' \in d} f_{t',d}$ 1/5
 - Log normalization $\log(1 + f_{t,d})$ $\log(1 + 1)$

document	the	boy	liked	book	i	like	flight	will
the boy liked the book	2	1	1	1	0	0	0	0
i like the flight	1	0	0	0	1	1	1	0
the boy will book the flight	2	1	0	1	0	0	1	1

Inverse Document Frequency (IDF)

- $N = |\mathbf{D}|$ = total number of **documents**
- $|\{\mathbf{d} \in \mathbf{D} : \mathbf{t} \in \mathbf{d}\}|$ = number of documents where term appears
- $idf(\mathbf{t}, \mathbf{D}) = \log \frac{N+1}{|\{\mathbf{d} \in \mathbf{D} : \mathbf{t} \in \mathbf{d}\}|+1}$
- $idf(\text{boy}, \mathbf{D}) = \log \frac{3+1}{2+1}$

Note MATLAB default *idf*:
 $\ln N / |\{\mathbf{d} \in \mathbf{D} : \mathbf{t} \in \mathbf{d}\}|$

document	the	boy	liked	book	i	like	flight	will
the boy liked the book	2	1	1	1	0	0	0	0
i like the flight	1	0	0	0	1	1	1	0
the boy will book the flight	2	1	0	1	0	0	1	1

TF-IDF

- $tfidf_{t,d \in D} = tf_{t,d} \times idf_{t,D}$
- $tfidf_{boy,d1 \in D} = 1 \times \log(1 + 1)$
- IDF measures the spread of the term across the document collection.
- Higher ***tfidf*** means word is more frequent in document and appears at all in less documents.
- Low ***tfidf*** means either word is rare in document or appears in most documents.

document	the	boy	liked	book	i	like	flight	will
the boy liked the book	2	1	1	1	0	0	0	0
i like the flight	1	0	0	0	1	1	1	0
the boy will book the flight	2	1	0	1	0	0	1	1

Remaining challenges

- **Vocabulary:**
 - Needs careful management of size, which impacts document representation, sparsity, and feature space.
- **Sparsity:**
 - A lot of elements will be 0s
 - Harder to model, less efficient
- **Meaning**
 - Discarding word order ignores context, and in turn the meaning of the words.

Possible solutions

- Further finetuning:
 - Tokenisation, stopwords removal, text normalisation.
- Applying **dimensionality reduction**
 - e.g. Principle Components Analysis
- Learning low-dimensional **word embeddings**, e.g.
 - Word2Vec
 - GloVe
 - fastText

Meaning in Context

Foundational assumption

- You shall know a word by the company it keeps
 - Firth J.R. (1957)
- You gain a lot by representing a word in terms of its neighbours
- One of the most successful ideas in text processing

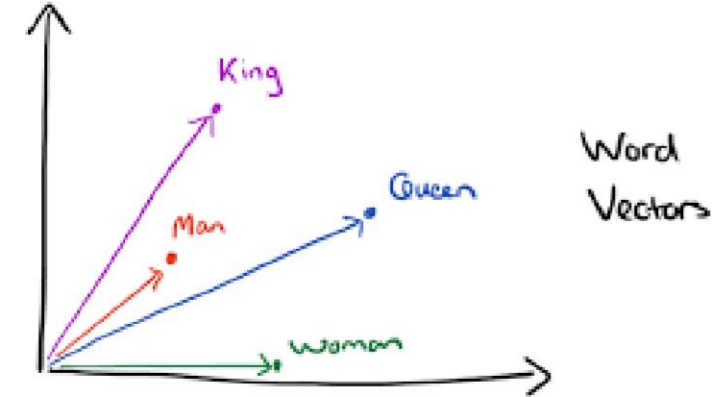
Vector representation

- aims at capturing the similarities between words



Word Embedding Models

- Dense vector representation of words, **learned** from a very large corpus of text.
- A **word** is represented by a low-dimensional fixed-size vector.
- Similar words have similar vectors.
- Can do “word maths” with vectors.
- Can be used as input to machine learning.



$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

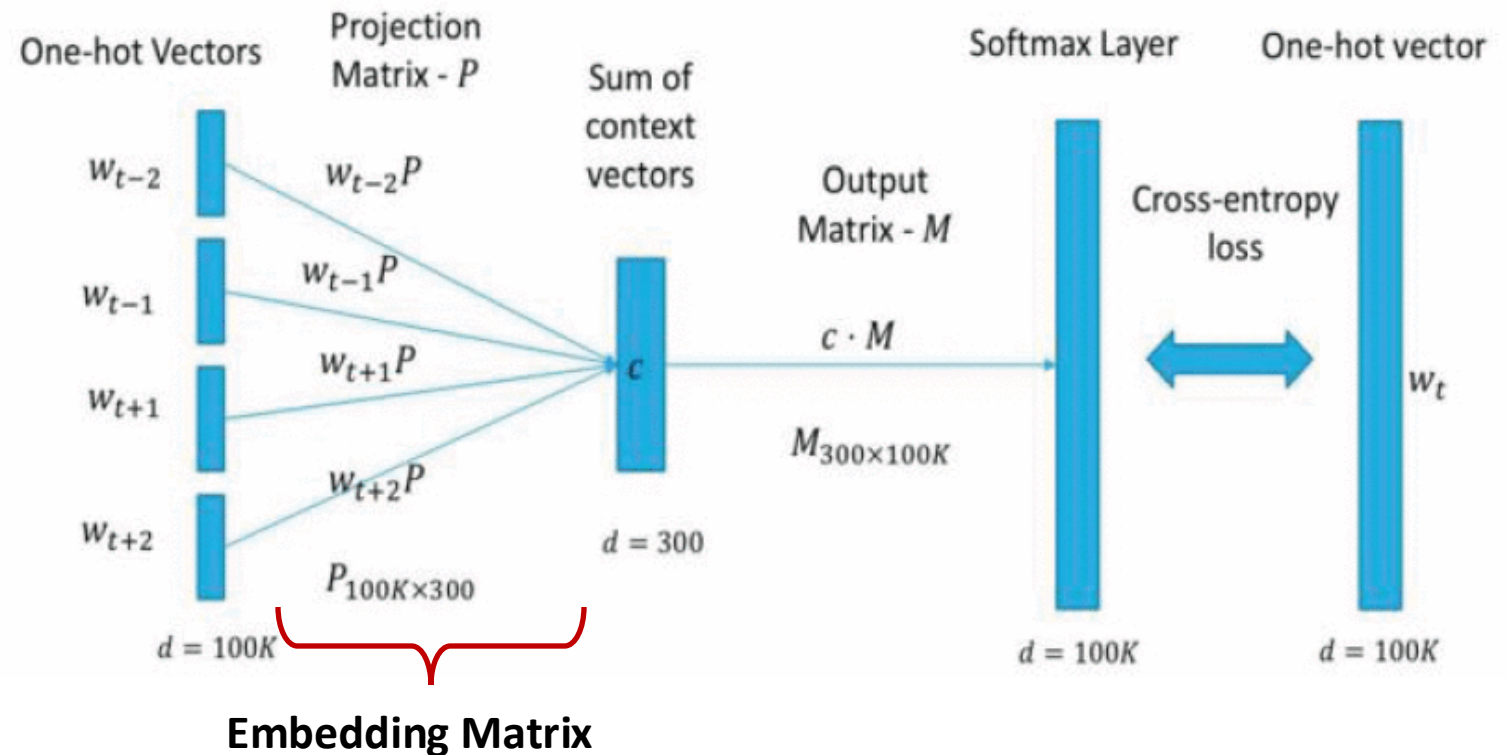
$$\text{good} \approx \text{great}$$

$$\text{Germany} \leftrightarrow \text{Berlin} \approx \text{Iran} \leftrightarrow \text{Tehran}$$

Word2Vec - CBOW

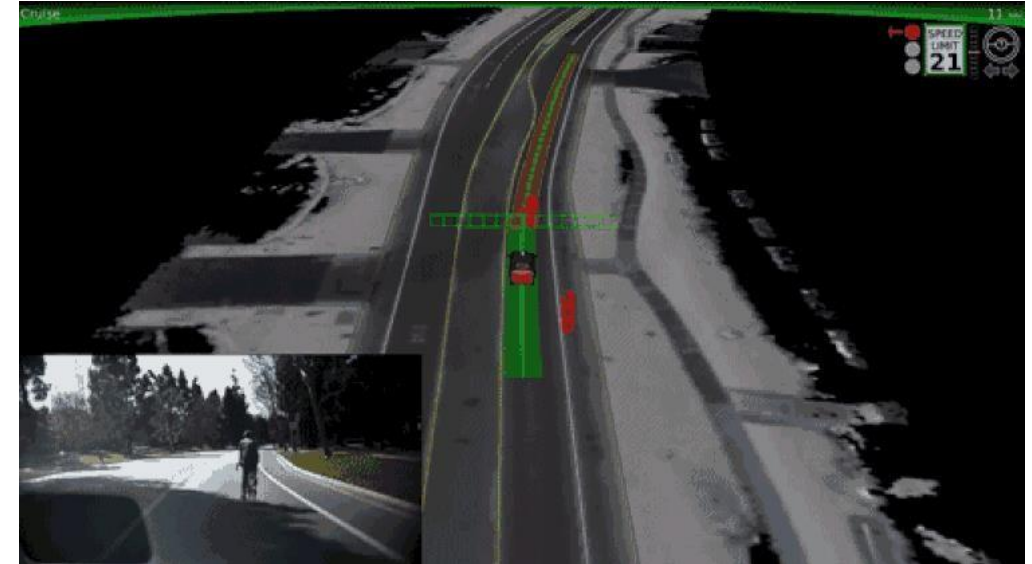
- Continuous Bag of Words (CBOW) Neural Network model, trained on running text (**Experience**), with target word masked, and surrounding context as extracted features, to predict target word (**Task**), and tested against masked word (**Performance**).
- Back propagation updates weights in layer of neural network. Final weights (here 300) are extracted as our Word Embedding vectors for each word in the defined vocabulary (here 100k).

<i>the</i>	<i>boy</i>	<i>liked</i>	<i>the</i>	<i>book</i>
w_{t-2}	w_{t-1}	w_t	w_{t+1}	w_{t+3}



Feature Extraction from Images

- Detects and represents the interesting parts of an image as a compact feature vector
- A critical step in image processing and computer vision
- Transforms pictorial image to numeric data representation
- These representations can be used as inputs to classification algorithms

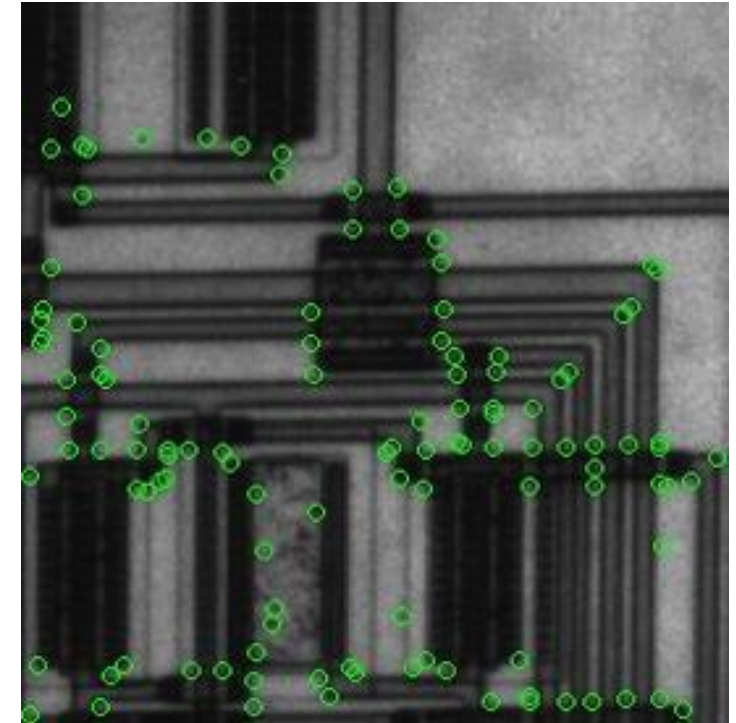


Example: Comparing Images



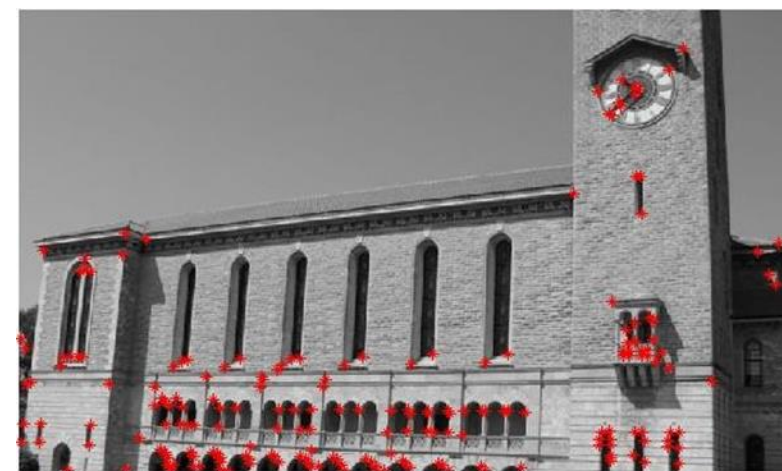
What counts as a “good” feature?

- Repeatable
 - Should be detectable at the same location in different images despite changes in viewpoint and illumination
- Saliency
 - Descriptive
 - Same points in different images should have similar features
- Compactness
 - Affect speed of matching
 - Fewer and smaller features can be better

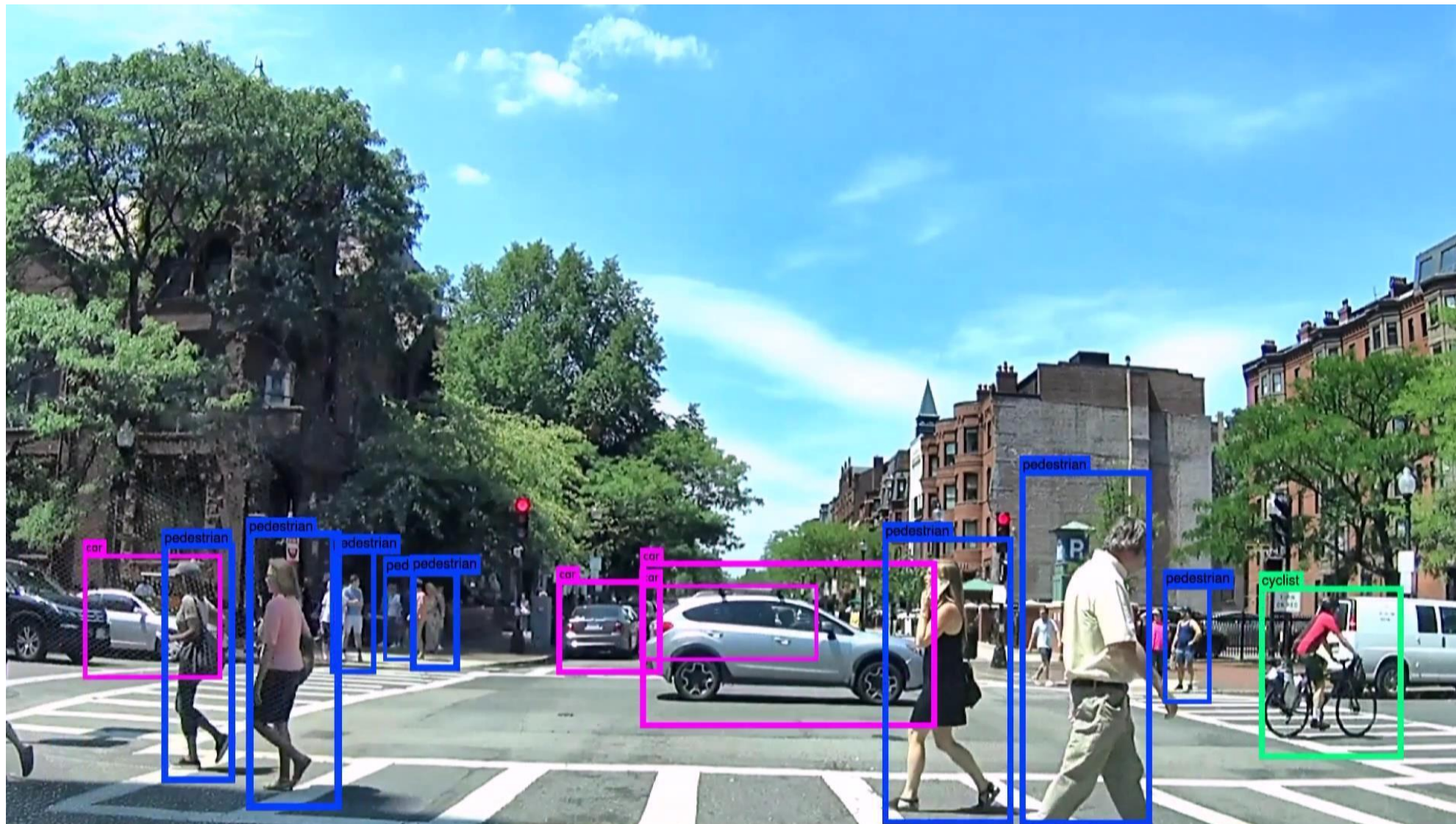


Steps to Image Feature Extraction

- There are two key steps:
- **Feature detection:**
 - Where to extract features from
 - Not all locations on the image are good
- **Feature Extraction:**
 - Orientation of the extraction window
 - What to encode in the feature



Encoding the world around us



How do we interpret an image

- An image is an array of pixels
- Pixel values can be extracted as feature vector representation for images
- It is important to know which **pixels** and which **interpretation** are relevant for the task

- 8-bit Image:

$$\mathbf{u} \in \{0, 1, \dots, 255\}^{m \times n \times p}$$

- Double Format:

$$\mathbf{u} \in [0, 1]^{m \times n \times p}$$

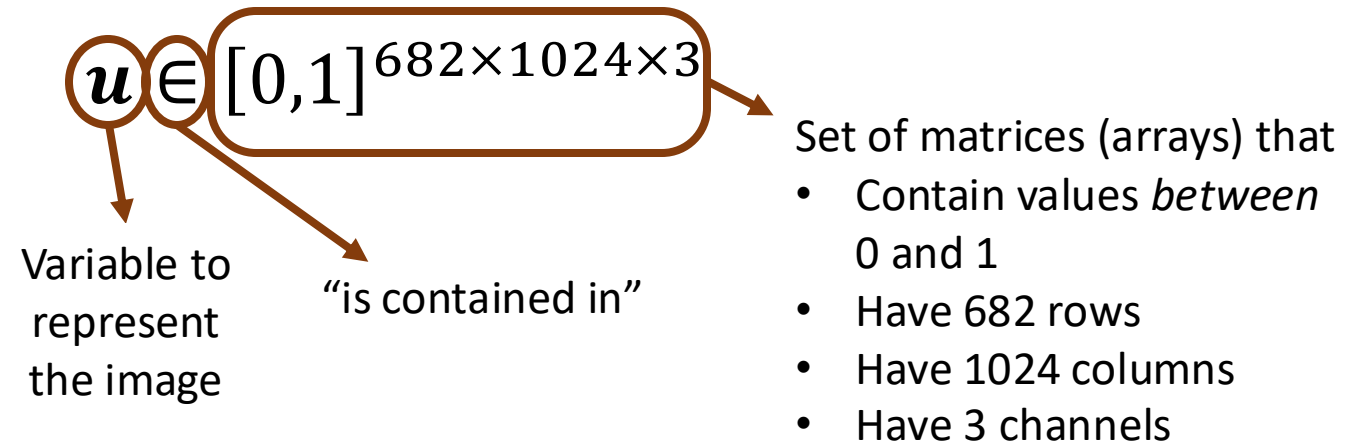


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	168	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	103	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	168	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	103	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

How do we interpret an image

- An image is an array of pixels
- Pixel values can be extracted as feature vector representation for images
- It is important to know which **pixels** and which **interpretation** are relevant for the task



"This is a 682x1024x3 image with values between 0 and 1 and we'll call it u "

How do we interpret an image

- An image is an array of pixels
- Pixel values can be extracted as feature vector representation for images
- It is important to know which **pixels** and which **interpretation** are relevant for the task



$$\mathbf{u} \in [0,1]^{682 \times 1024 \times 3}$$

$$\mathbf{u}^r(\Omega) = \begin{bmatrix} 0.55 & 0.72 & 0.44 \\ 0.23 & 0.56 & 0.23 \\ 0.67 & 0.52 & 0.93 \end{bmatrix}$$

$$\mathbf{u}^g(\Omega) = \begin{bmatrix} 0.45 & 0.23 & 0.76 \\ 0.76 & 0.34 & 0.76 \\ 0.94 & 0.98 & 0.93 \end{bmatrix}$$

$$\mathbf{u}^b(\Omega) = \begin{bmatrix} 0.56 & 0.57 & 0.34 \\ 0.12 & 0.57 & 0.75 \\ 0.56 & 0.76 & 0.45 \end{bmatrix}$$

How do we interpret an image

- An image is an array of pixels
- Pixel values can be extracted as feature vector representation for images
- It is important to know which **pixels** and which **interpretation** are relevant for the task



$$\mathbf{u} \in [0,1]^{682 \times 1024 \times 3}$$

$$\mathbf{u}^r(\Omega) = \begin{bmatrix} 0.55 & 0.72 & 0.44 \\ 0.23 & 0.56 & 0.23 \\ 0.67 & 0.52 & 0.93 \end{bmatrix}$$

$$\mathbf{u}^g(\Omega) = \begin{bmatrix} 0.45 & 0.23 & 0.76 \\ 0.76 & 0.34 & 0.76 \\ 0.94 & 0.98 & 0.93 \end{bmatrix}$$

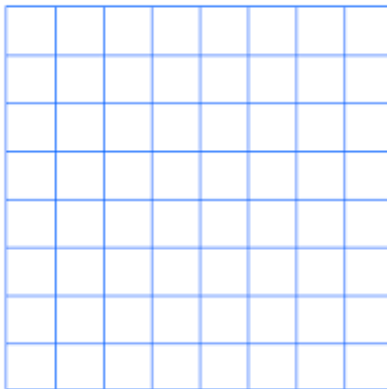
$$\mathbf{u}^b(\Omega) = \begin{bmatrix} 0.56 & 0.57 & 0.34 \\ 0.12 & 0.57 & 0.75 \\ 0.56 & 0.76 & 0.45 \end{bmatrix}$$

How do we interpret an image

Discrete Space vs Continuous Space

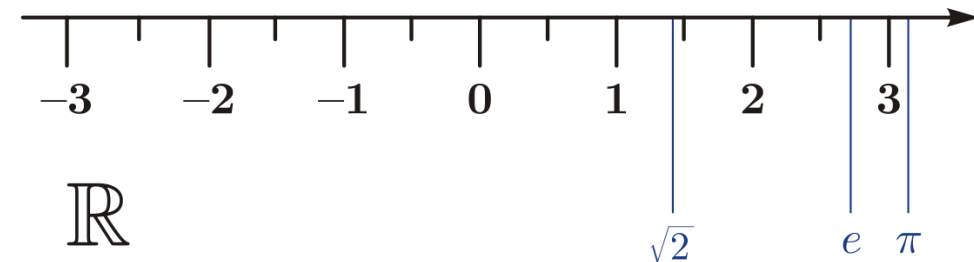
Discrete:

- Think of as a grid
- Limited Set of Values
- Practice



Continuous Space

- Think of the Real Line
- Infinite Set of Values
- Theory



How do we interpret an image

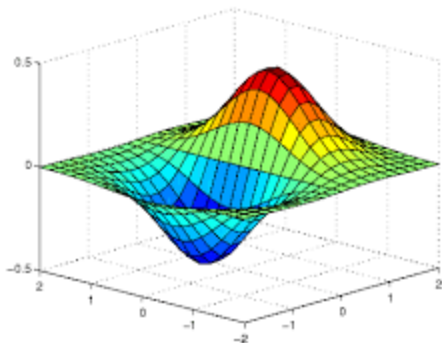
To develop computer vision theory:

We view an image as a function:

$$u: \underbrace{\mathbb{Z} \times \mathbb{Z}}_{\text{pixel rows and columns}} \rightarrow \underbrace{\mathbb{R}}_{\text{intensity values}}$$

We view a video as a function:

$$u: \underbrace{\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}}_{\text{pixel rows and columns, frame}} \rightarrow \underbrace{\mathbb{R}}_{\text{intensity values}}$$



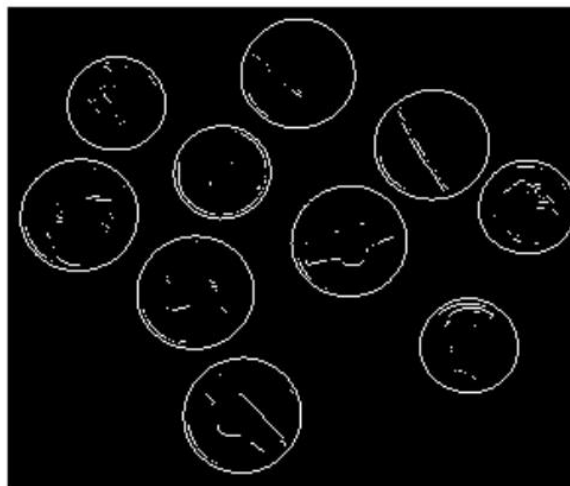
Try loading an image “im” in Matlab and typing
`mesh(im(:,: , 1))`

Edge Detection

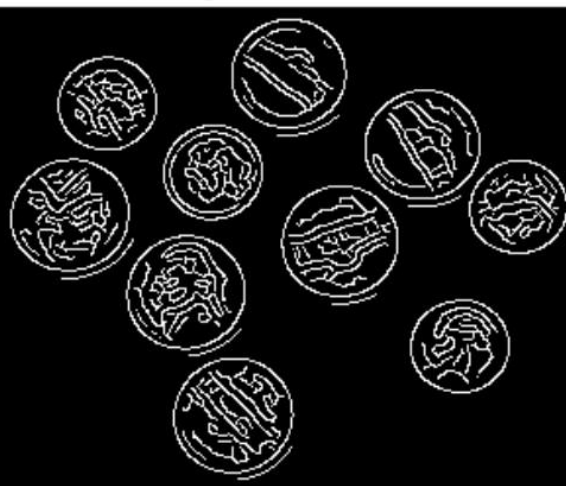
- An edge is a curve that follows a path of rapid change in image intensity
- Edge detection is used to identify the edges in an image
- The most well-known edge-detection method is Canny filter – based on gradients



Sobel Filter

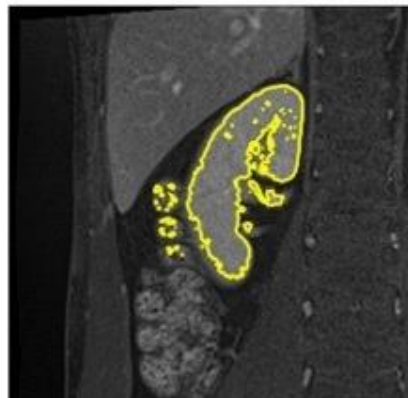
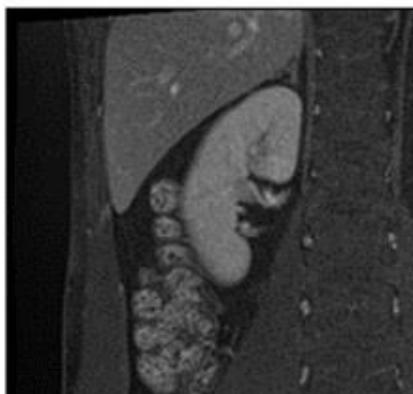


Canny Filter



Edge Detection

- An edge is a curve that follows a path of rapid change in image intensity
- Edge detection is used to identify the edges in an image
- The most well-known edge-detection method is Canny filter – based on gradients
- But more complex approaches do exist



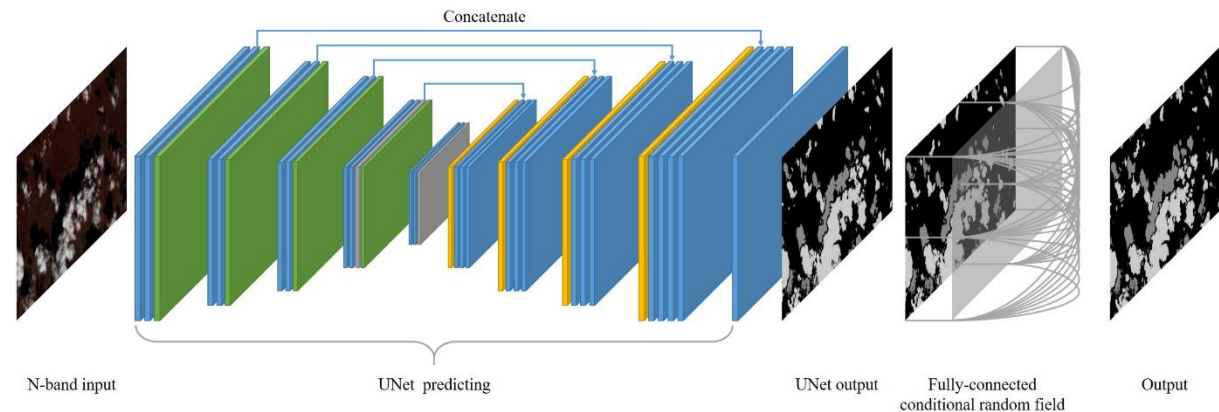
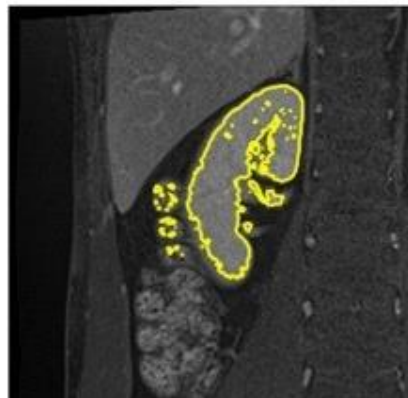
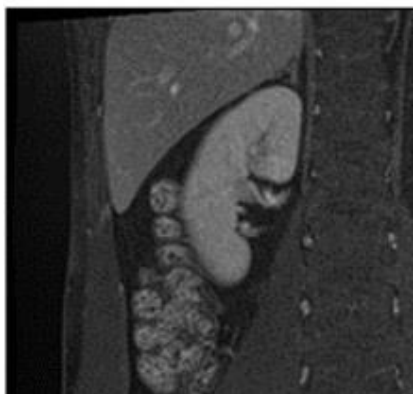
This is a research example for interest. You do **not** need to know this for this module:

$$\min_{\Omega} \int_{\Omega} |f(x) - m(\Omega)|^2 dx + \int_{\Omega^c} |f(x) - m(\Omega^c)|^2 dx + \text{Perim}(\Omega)$$

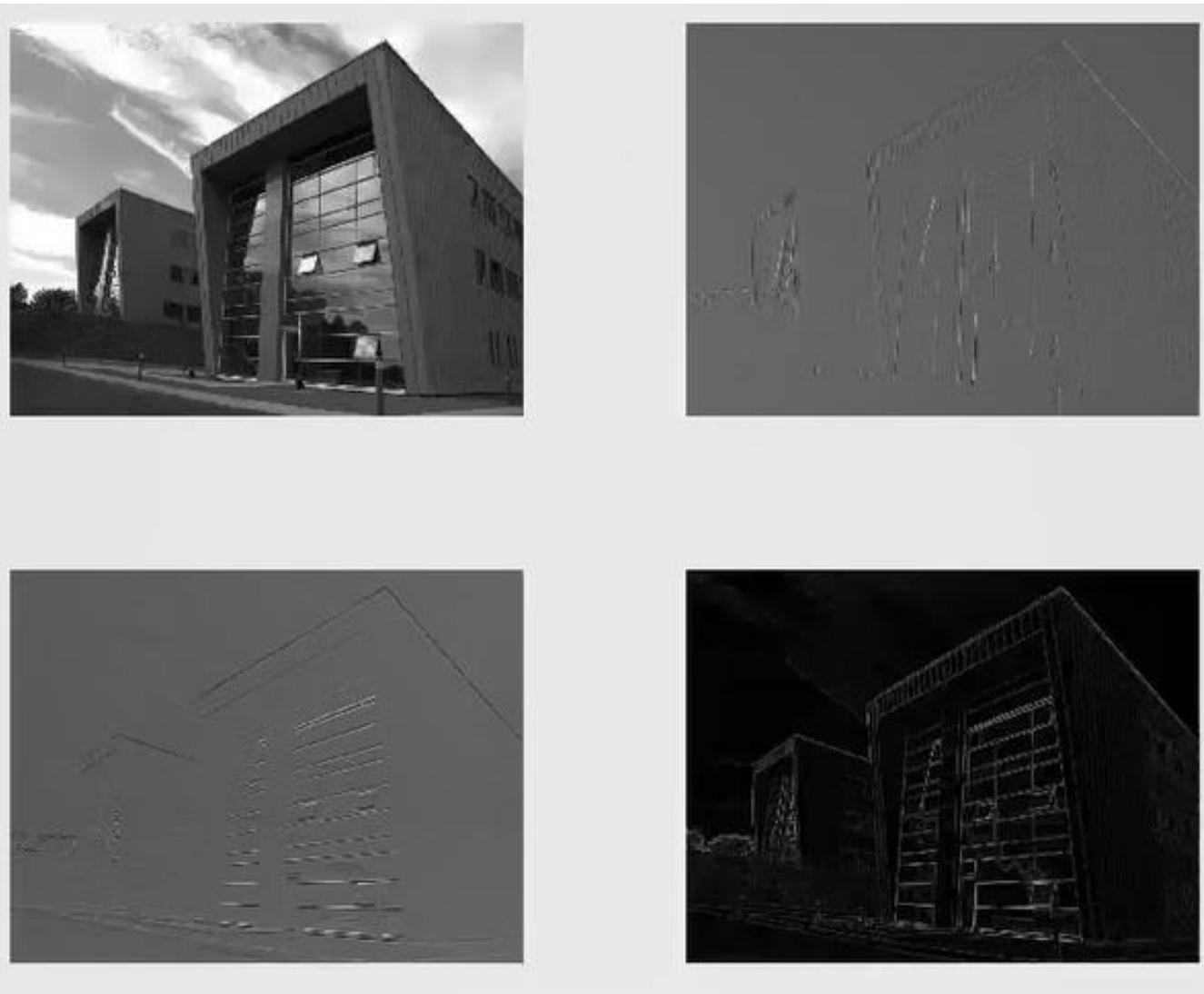
$$\text{where } m(\Omega) \stackrel{\text{def.}}{=} \frac{1}{|\Omega|} \int_{\Omega} f(x) dx$$

Edge Detection

- An edge is a curve that follows a path of rapid change in image intensity
- Edge detection is used to identify the edges in an image
- The most well-known edge-detection method is Canny filter – based on gradients
- But more complex approaches do exist



Meaning of Edge Detection Approaches?



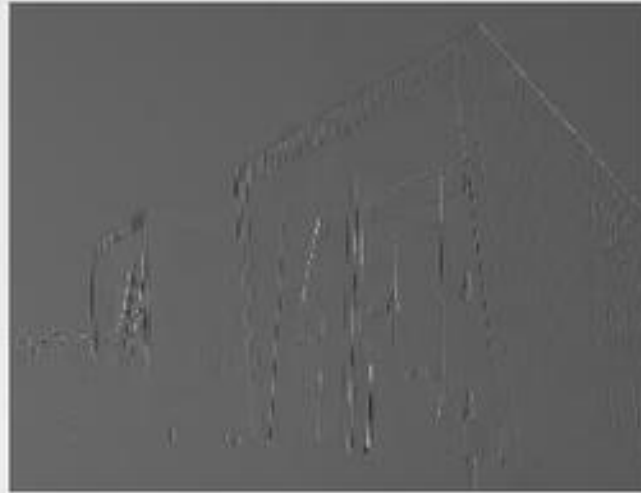
Edge Detection

Sobel Filter

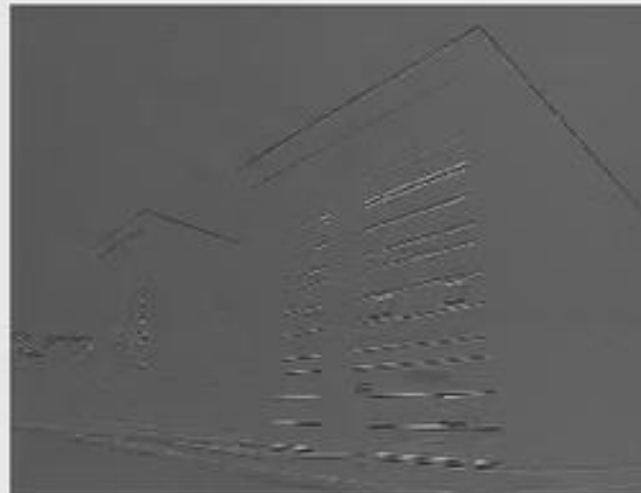
A



$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A$$



$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$



$$G = \sqrt{G_x^2 + G_y^2}$$



Convolution in Practice

$$\begin{array}{ccccc} & \mathbf{u} & & & \\ \hline & 8 & 8 & 2 & 1 & 1 \\ \hline & 7 & 8 & 2 & 1 & 0 \\ \hline & 10 & 10 & 3 & 2 & 1 \\ \hline & 9 & 10 & 1 & 1 & 0 \\ \hline & 9 & 9 & 2 & 2 & 0 \\ \hline \end{array} \quad * \quad \begin{array}{ccc} & \mathbf{k} & \\ \hline & +1 & 0 & -1 \\ \hline & +2 & 0 & -2 \\ \hline & +1 & 0 & -1 \\ \hline \end{array} \quad = \quad \begin{array}{ccccc} & & \mathbf{u * k} & & \\ \hline & & & & \\ \hline & & 23 & 29 & 7 \\ \hline & & 27 & 32 & 7 \\ \hline & & 30 & 33 & 6 \\ \hline & & & & \\ \hline \end{array}$$

Convolution in Practice

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

k

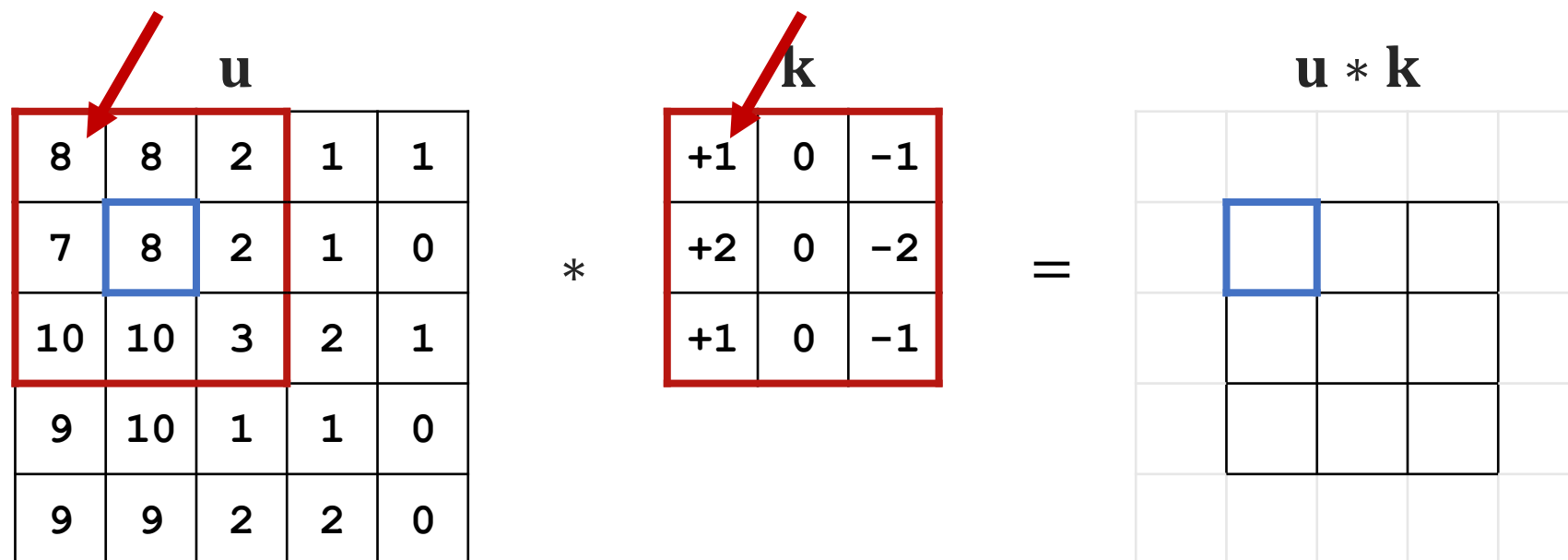
+1	0	-1
+2	0	-2
+1	0	-1

$*$

u * k

	23			

Convolution in Practice

 8×1


Convolution in Practice

$$8 \times 1 + 8 \times 0$$

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

k

+1	0	-1
+2	0	-2
+1	0	-1


=

u * k

Convolution in Practice

$$8 \times 1 + 8 \times 0 + 2 \times -1$$


u



8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

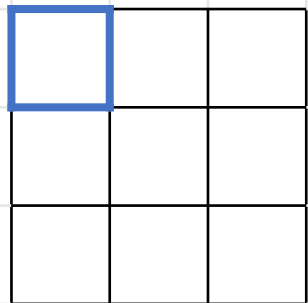
k



+1	0	-1
+2	0	-2
+1	0	-1

=

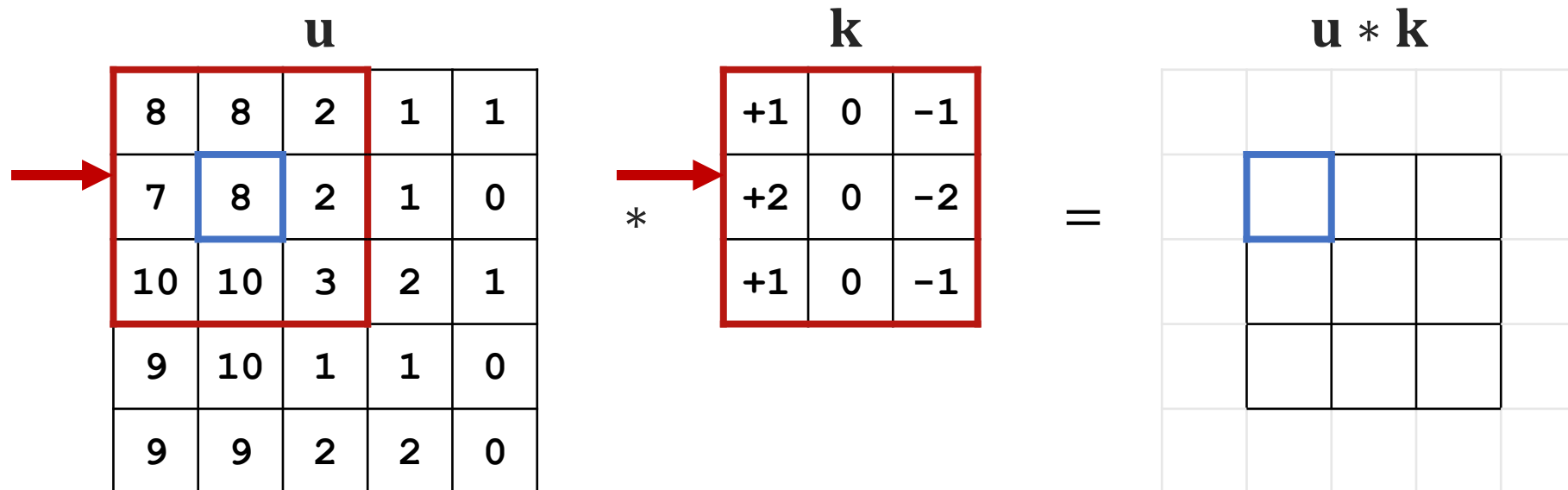
u * k



Convolution in Practice

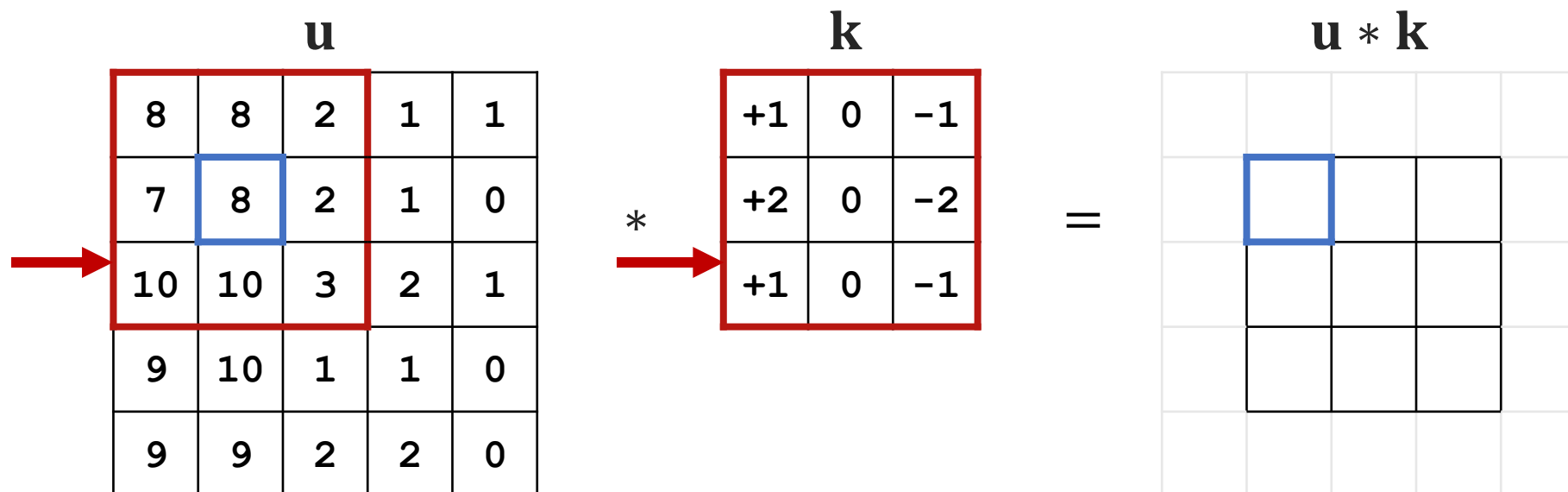
$$8 \times 1 + 8 \times 0 + 2 \times -1 +$$

$$7 \times 2 + 8 \times 0 + 2 \times -2$$



Convolution in Practice

$$\begin{aligned}
&8 \times 1 + 8 \times 0 + 2 \times -1 + \\
&7 \times 2 + 8 \times 0 + 2 \times -2 + \\
&10 \times 1 + 10 \times 0 + 3 \times -1
\end{aligned}$$



Convolution in Practice

$$\begin{aligned}
 &8 + 0 - 2 + \\
 &14 + 0 - 4 + \\
 &10 + 0 - 3 \\
 &= 23
 \end{aligned}$$

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

k

+1	0	-1
+2	0	-2
+1	0	-1

=

u * k

	23			

Convolution in Practice

$$\begin{aligned}
&8 \times 1 + 2 \times 0 + 1 \times -1 + \\
&8 \times 2 + 2 \times 0 + 1 \times -2 + \\
&10 \times 1 + 3 \times 0 + 2 \times -1 \\
&= \mathbf{29}
\end{aligned}$$

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

k

+1	0	-1
+2	0	-2
+1	0	-1

=

u * k

	23	29		

Convolution in Practice

$$\begin{aligned}
&2 \times 1 + 1 \times 0 + 1 \times -1 + \\
&2 \times 2 + 1 \times 0 + 0 \times -2 + \\
&3 \times 1 + 2 \times 0 + 1 \times -1 \\
&= 7
\end{aligned}$$

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

k

+1	0	-1
+2	0	-2
+1	0	-1

=

u * k

	23	29	7	

Convolution in Practice

$$\begin{aligned}
&7 \times 1 + 8 \times 0 + 2 \times -1 + \\
&10 \times 2 + 10 \times 0 + 3 \times -2 + \\
&9 \times 1 + 10 \times 0 + 1 \times -1 \\
&= \mathbf{27}
\end{aligned}$$

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

k

+1	0	-1
+2	0	-2
+1	0	-1

=

u * k

	23	29	7	
	27			

and so on...

Convolution in Practice

$$\begin{aligned}
 &3 \times 1 + 2 \times 0 + 1 \times -1 + \\
 &1 \times 2 + 1 \times 0 + 0 \times -2 + \\
 &2 \times 1 + 2 \times 0 + 0 \times -1 \\
 &= 6
 \end{aligned}$$

u

8	8	2	1	1
7	8	2	1	0
10	10	3	2	1
9	10	1	1	0
9	9	2	2	0

*

k

+1	0	-1
+2	0	-2
+1	0	-1

=

u * k

	23	29	7	
	27	32	7	
	30	33	6	

Note: actually, the values shown in **u * k** would be negative, as convolution rotates the kernel (**k**) by 180°

Edge Detection

Sobel Filter

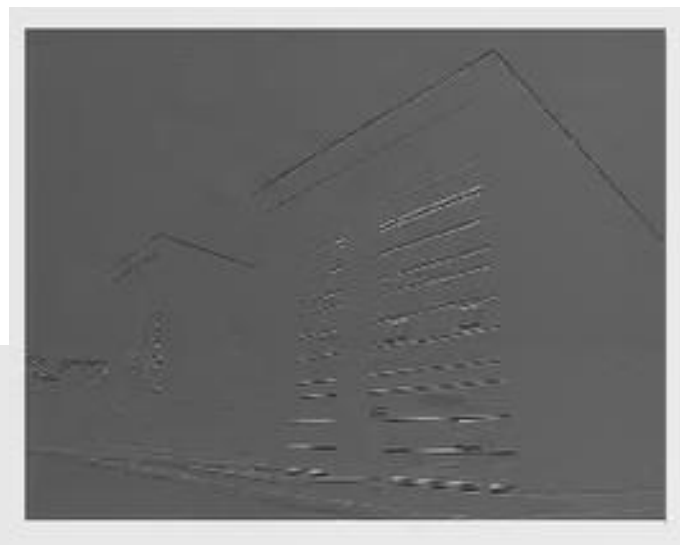
A



$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A$$



$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$



$$G = \sqrt{G_x^2 + G_y^2}$$



Edge Detection – in continuous space...

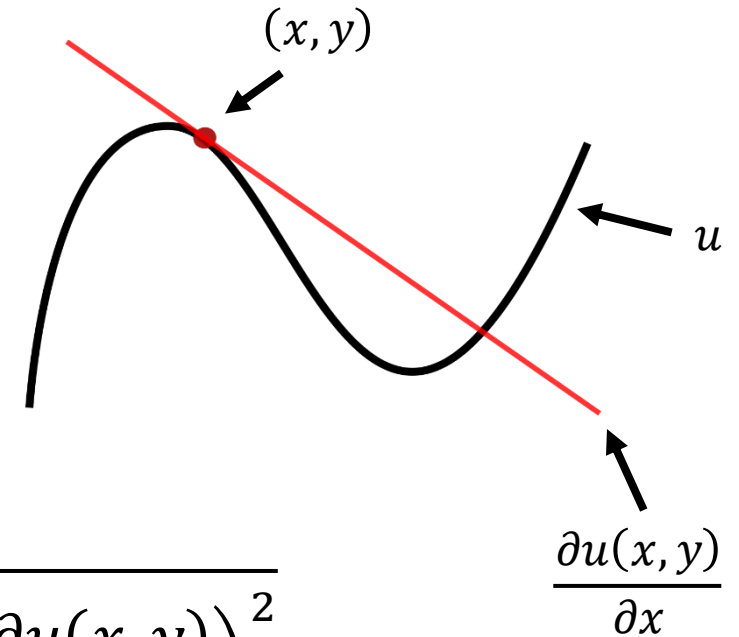
Differentiation

1st and 2nd derivative wrt x: $\frac{\partial u(x,y)}{\partial x}, \frac{\partial^2 u(x,y)}{\partial x^2}$

1st and 2nd derivative wrt y: $\frac{\partial u(x,y)}{\partial y}, \frac{\partial^2 u(x,y)}{\partial y^2}$

Edge Detector:

$$E(x,y) = \sqrt{\left(\frac{\partial^2}{\partial x^2} \frac{\partial u(x,y)}{\partial y}\right)^2 + \left(\frac{\partial^2}{\partial y^2} \frac{\partial u(x,y)}{\partial x}\right)^2}$$



Achieve matrix representation through discretisation with finite differences

Edge Detection – finite differences

Finite Differences

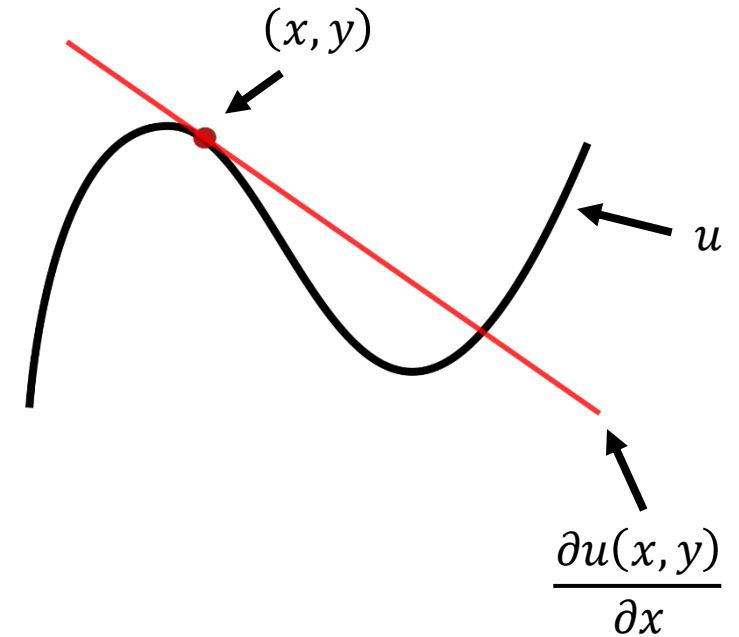
1st order wrt x:
$$\frac{\partial u(x, y)}{\partial x} = \lim_{h \rightarrow 0} \frac{u(x + h, y) - u(x, y)}{h}$$

Set $h = 1$:
$$\left. \frac{\partial u(x, y)}{\partial x} \right|_{h=1} = u(x + 1, y) - u(x, y)$$

In the discrete case, we have

$$\text{diff}(\mathbf{u}) = \mathbf{u}[i + 1, j] - \mathbf{u}[i, j]$$

What about the second derivative? **HINT:** apply 1st order twice



Edge Detection – implementation

2nd Order:

$$\text{diff}(\text{diff}(\mathbf{u})) = \mathbf{1} \cdot \mathbf{u}[i + \mathbf{1}, j] - 2 \cdot \mathbf{u}[i, j] + \mathbf{1} \cdot \mathbf{u}[i - \mathbf{1}, j]$$

Edge Detector:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

Edge Detection

Key take home message:

	Theory:	Implementation:
1 st order derivative wrt x	$\frac{\partial u(x, y)}{\partial x}$	$u[i + 1, j] - u[i, j]$
1 st order derivative wrt y	$\frac{\partial u(x, y)}{\partial y}$	$u[i, j + 1] - u[i, j]$
2 nd order derivative wrt x	$\frac{\partial^2 u(x, y)}{\partial x^2}$	$u[i - 1, j] - 2u[i, j] + u[i + 1, j]$
2 nd order derivative wrt y	$\frac{\partial^2 u(x, y)}{\partial y^2}$	$u[i, j - 1] - 2u[i, j] + u[i, j + 1]$

Coming up... Decision Making

