Machine Learning

Ingredients of Machine Learning

Learning Objectives

Upon successful completion of this course, you should be able to:

- recognize situations where machine learning methods could be applied

Learning Objectives

Upon successful completion of this course, you should be able to:

- recognize situations where machine learning methods could be applied
- for a given task recognize which subfield of machine learning it belongs to

Learning Objectives

Upon successful completion of this course, you should be able to:

- recognize situations where machine learning methods could be applied
- for a given task recognize which subfield of machine learning it belongs to
- describe the principles behind the main supervised learning algorithms and explain the meaning of assumptions made by these algorithms for a particular application field

Learning Objectives

Upon successful completion of this course, you should be able to:

- recognize situations where machine learning methods could be applied
- for a given task recognize which subfield of machine learning it belongs to
- describe the principles behind the main supervised learning algorithms and explain the meaning of assumptions made by these algorithms for a particular application field
- in practice implement a solution using supervised machine learning algorithms

Learning Objectives

Upon successful completion of this course, you should be able to:

- recognize situations where machine learning methods could be applied
- for a given task recognize which subfield of machine learning it belongs to
- describe the principles behind the main supervised learning algorithms and explain the meaning of assumptions made by these algorithms for a particular application field
- in practice implement a solution using supervised machine learning algorithms
- **write the scripts of ML algorithms from scratch**

Course Format and Some Rules

- Lectures provide the theoretical knowledge

Course Format and Some Rules

- Lectures provide the theoretical knowledge
- Practice sessions support understanding of the lectures and provide practical knowledge

Course Format and Some Rules

- Lectures provide the theoretical knowledge
- Practice sessions support understanding of the lectures and provide practical knowledge
- We will use **Google Colab** platform to write Python programs for the practice sessions

Course Format and Some Rules

- Lectures provide the theoretical knowledge
- Practice sessions support understanding of the lectures and provide practical knowledge
- We will use **Google Colab** platform to write Python programs for the practice sessions
- Before every practice session you will have a homework assignment to complete

Course Materials

- All course materials will be available on **Google Drive**

Course Materials

- All course materials will be available on **Google Drive**
- Including some textbooks that I use for my lectures

Course Materials

- All course materials will be available on **Google Drive**
- Including some textbooks that I use for my lectures
  - Peter Flach, *"The Art and Science of Algorithms that Make Sense of Data"*, 2012

Course Materials

- All course materials will be available on **Google Drive**
- Including some textbooks that I use for my lectures
  - Peter Flach, *"The Art and Science of Algorithms that Make Sense of Data"*, 2012
  - Kevin Murphy, *"Machine Learning: A Probabilistic Perspective"*, 2012

Course Materials

- All course materials will be available on **Google Drive**
- Including some textbooks that I use for my lectures
  - Peter Flach, *"The Art and Science of Algorithms that Make Sense of Data"*, 2012
  - Kevin Murphy, *"Machine Learning: A Probabilistic Perspective"*, 2012
  - Trevor Hastie and others, *"The Elements of Statistical Learning"*, 2008

Course Materials

- All course materials will be available on **Google Drive**
- Including some textbooks that I use for my lectures
  - Peter Flach, *"The Art and Science of Algorithms that Make Sense of Data"*, 2012
  - Kevin Murphy, *"Machine Learning: A Probabilistic Perspective"*, 2012
  - Trevor Hastie and others, *"The Elements of Statistical Learning"*, 2008
  - Christopher Bishop, *"Pattern Recognition and Machine Learning"*, 2006

Mathematical Derivations

- This course is not about memorizing existing algorithms

Mathematical Derivations

- This course is not about memorizing existing algorithms
- Therefore, instead of showing you the algorithm right away, I often derive it from first principles

Mathematical Derivations

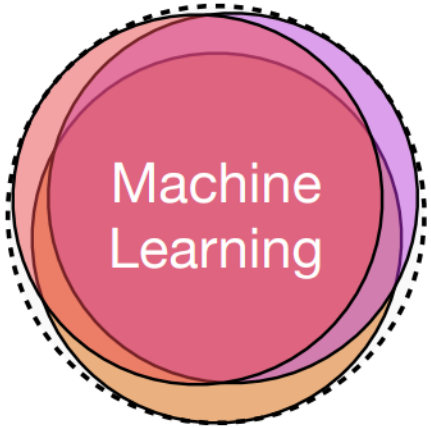
- This course is not about memorizing existing algorithms
- Therefore, instead of showing you the algorithm right away, I often derive it from first principles
- You will see mathematical derivations quite often and we need to make sure that each one of you understands them

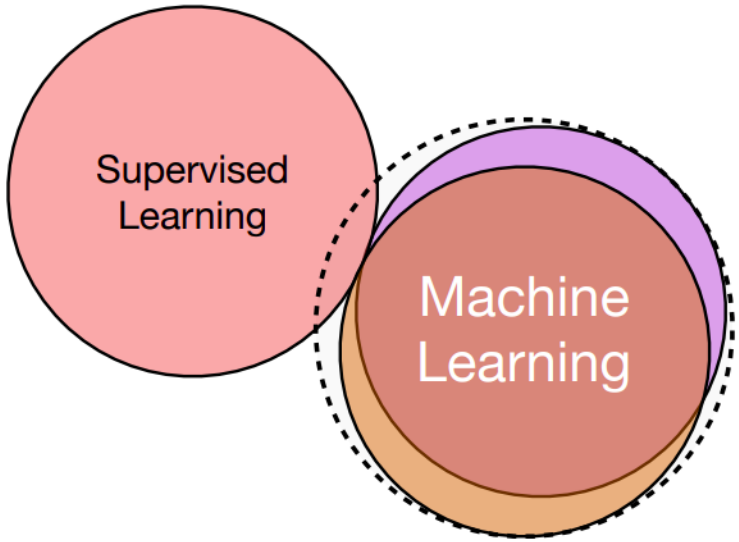
Mathematical Derivations

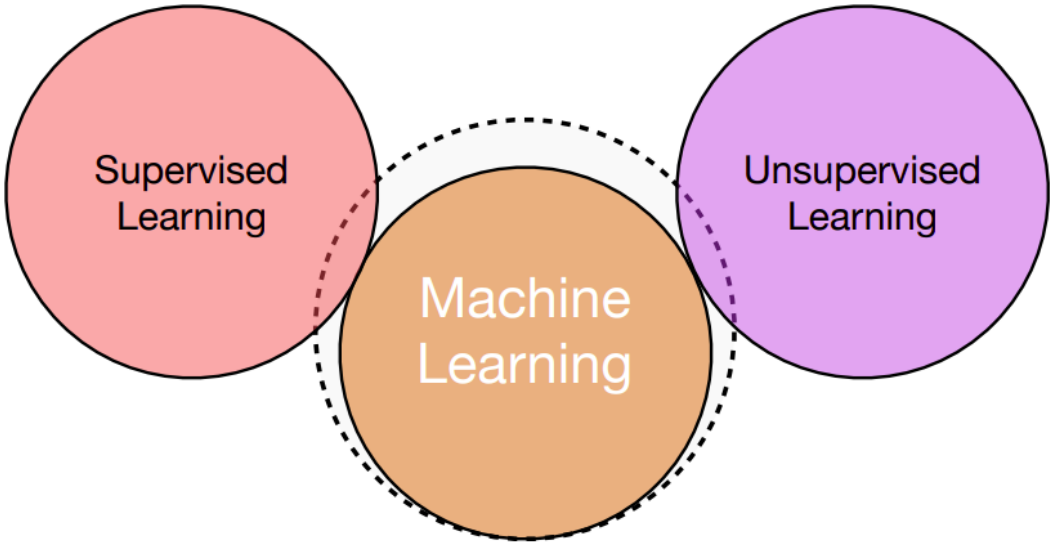
- This course is not about memorizing existing algorithms
- Therefore, instead of showing you the algorithm right away, I often derive it from first principles
- You will see mathematical derivations quite often and we need to make sure that each one of you understands them
- To fulfill the previous point **you need to ask questions**

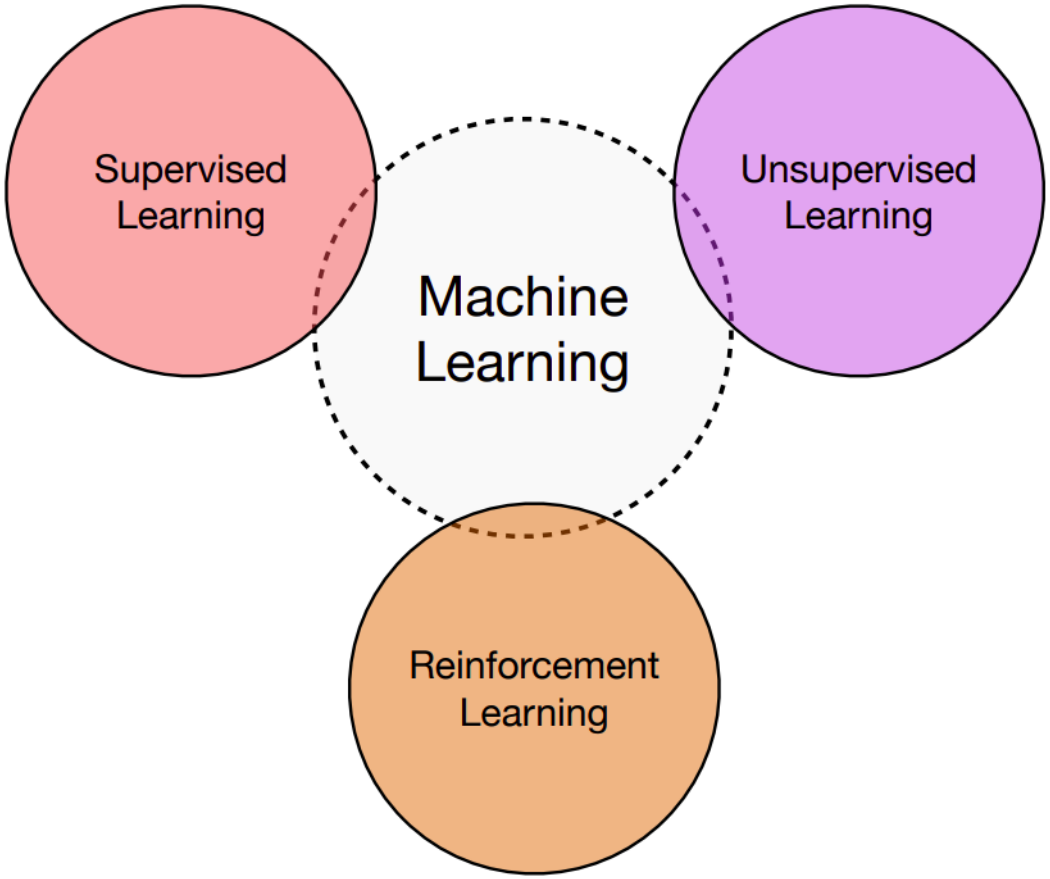
Topics of today's lecture

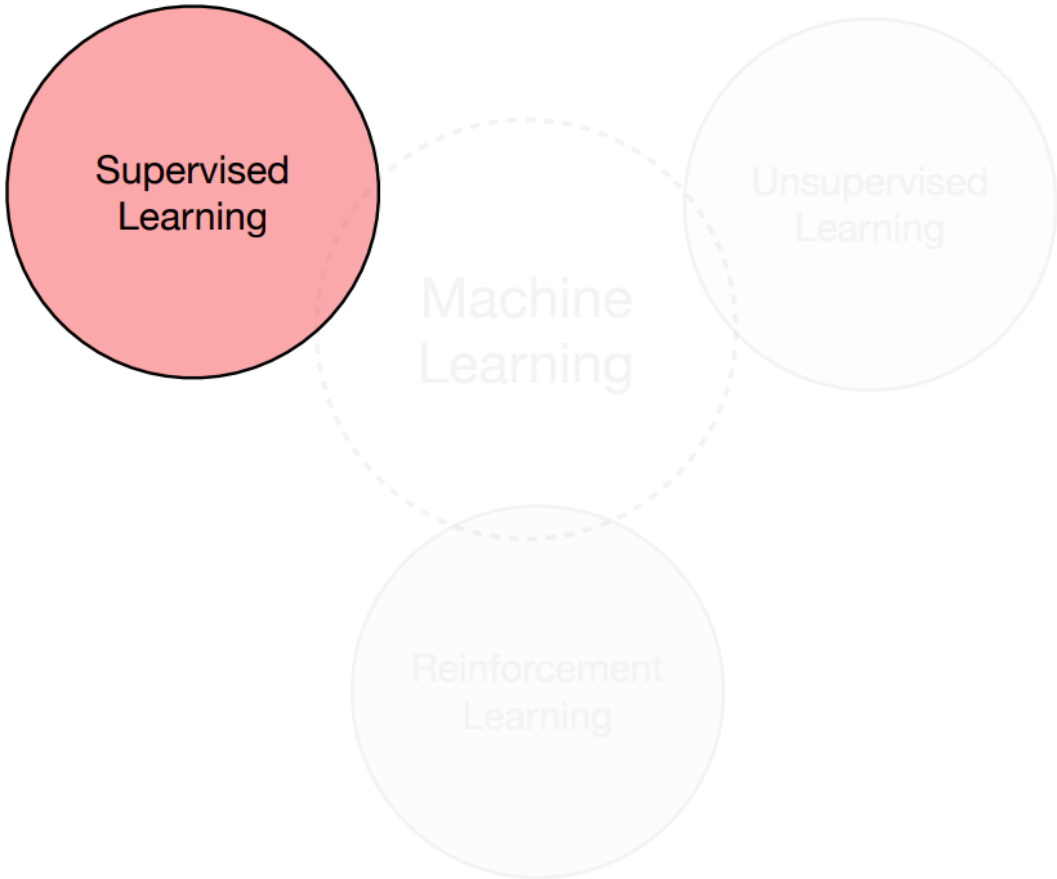
- Ingredients of Machine Learning
- Classification Basics
- Basic Linear Classifier

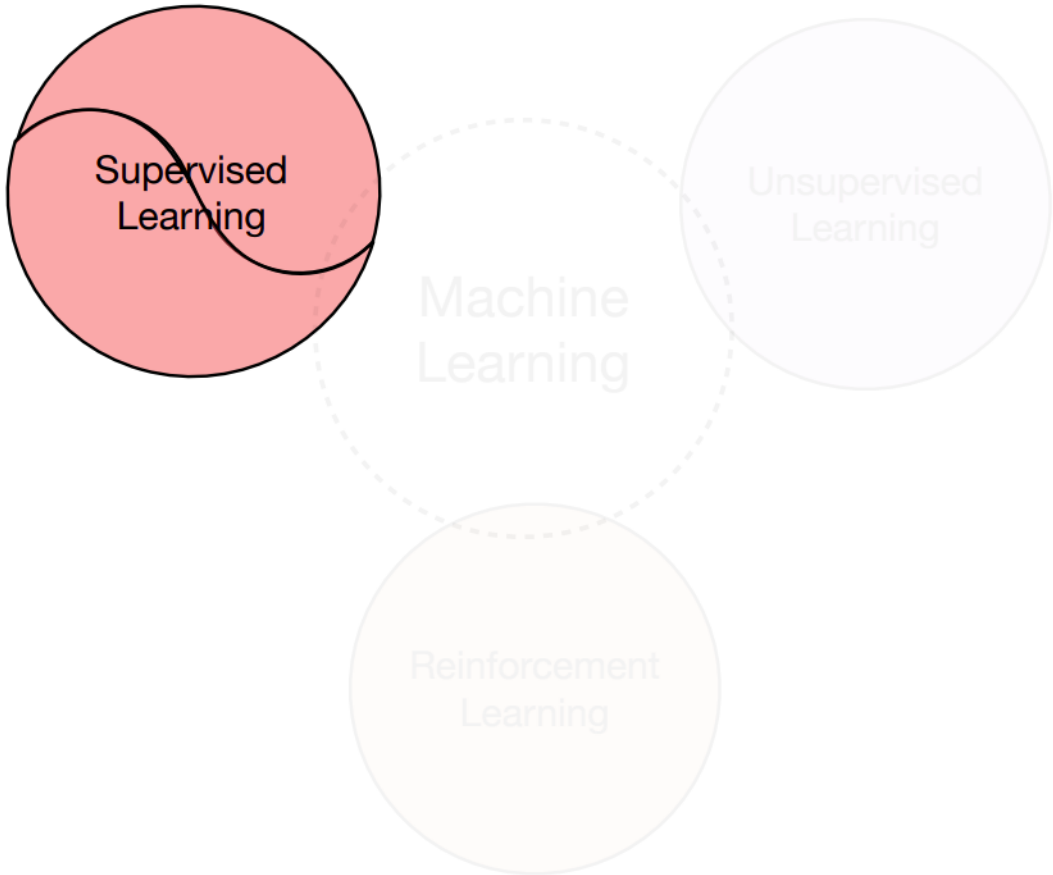

Machine
Learning

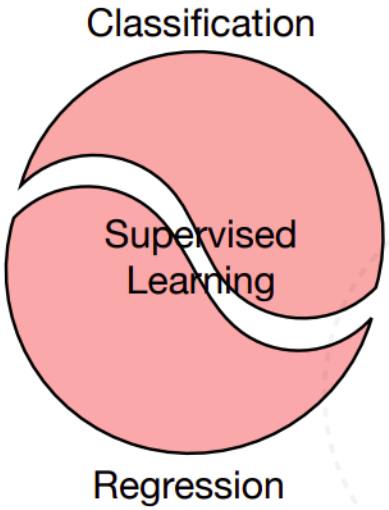

Supervised
Learning
Machine
Learning

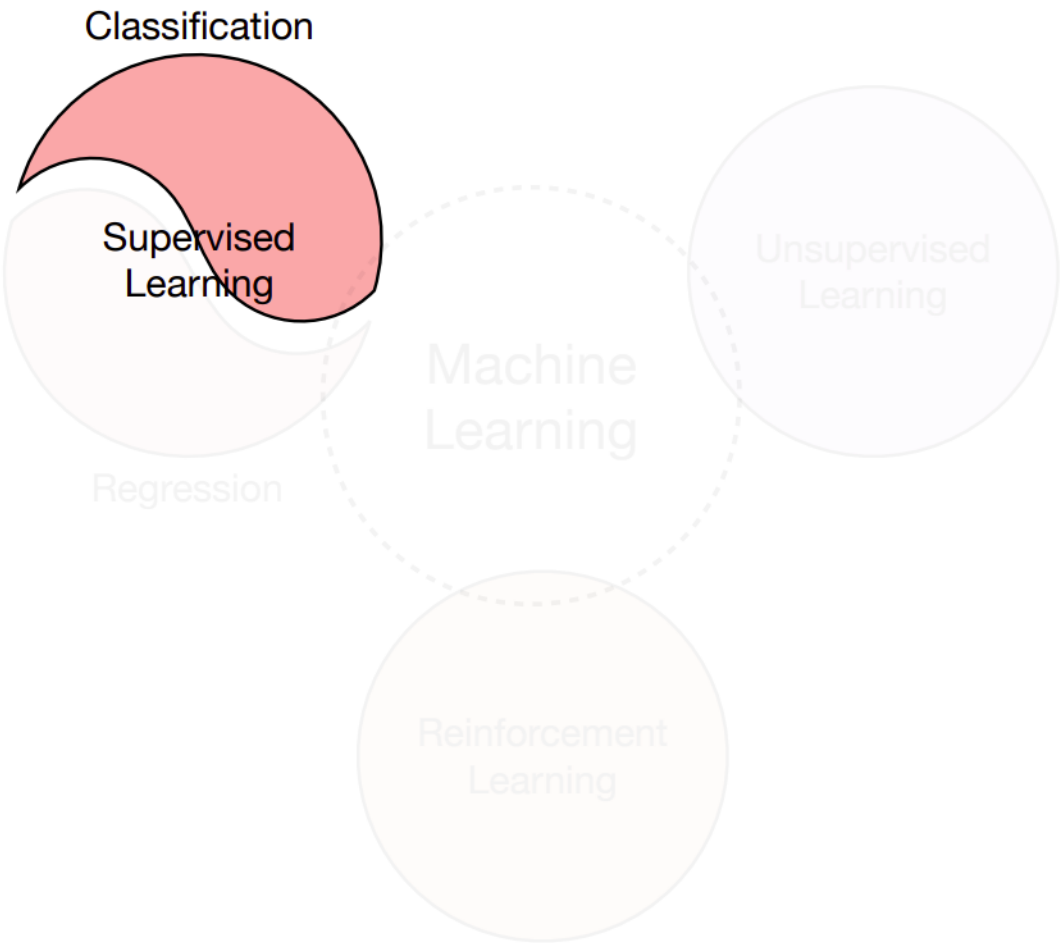

Supervised
Learning
Machine
Learning
Unsupervised
Learning

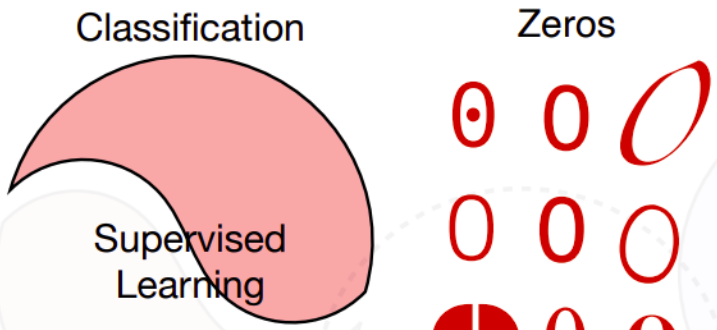

Supervised Learning
Unsupervised Learning
Machine Learning
Reinforcement Learning


Supervised Learning
Unsupervised Learning
Machine Learning
Reinforcement Learning

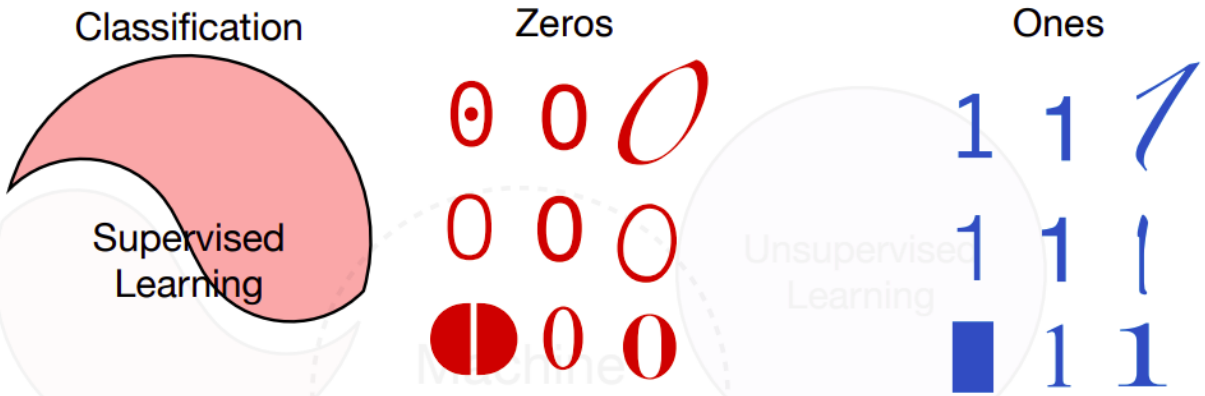

Supervised
Learning
Unsupervised
Learning
Machine
Learning
Reinforcement
Learning

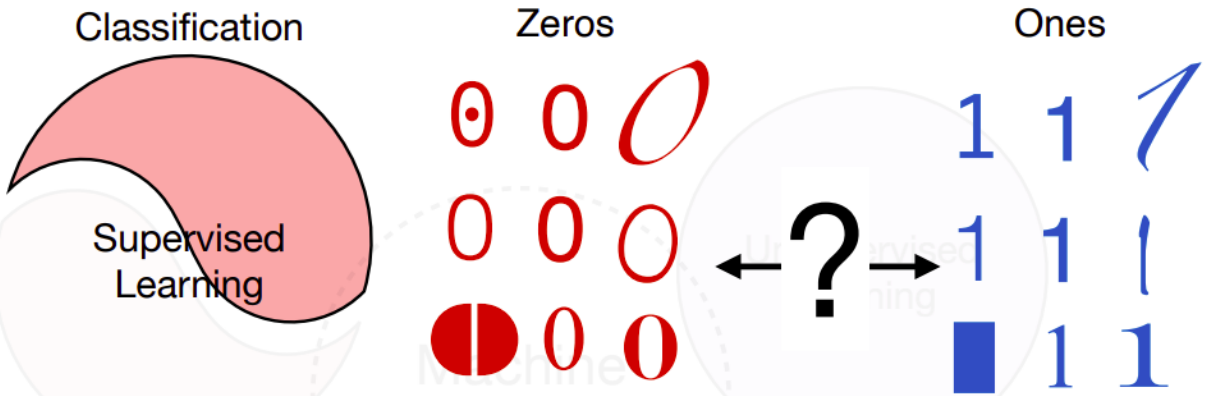

Classification
Supervised
Learning
Regression


Classification
Supervised
Learning
Machine
Learning
Regression
Unsupervised
Learning
Reinforcement
Learning

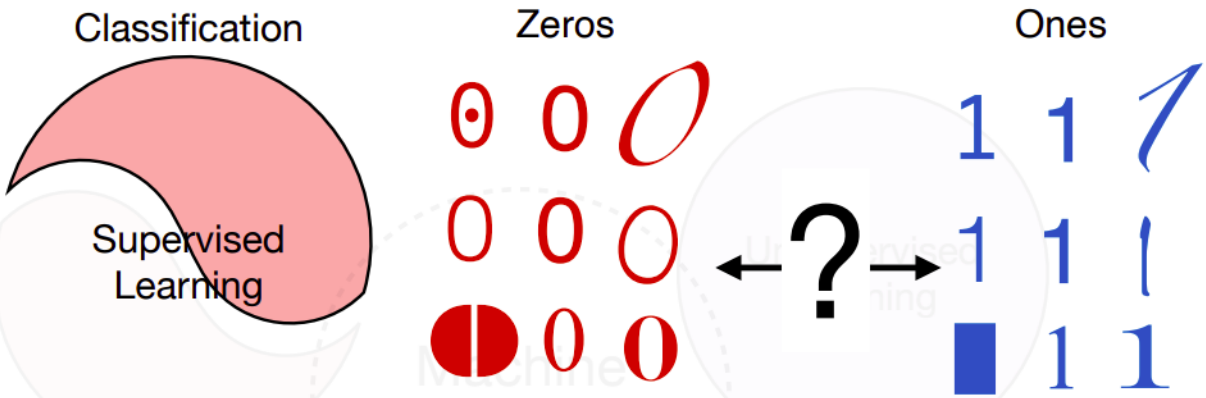

Classification
Supervised
Learning

Zeros

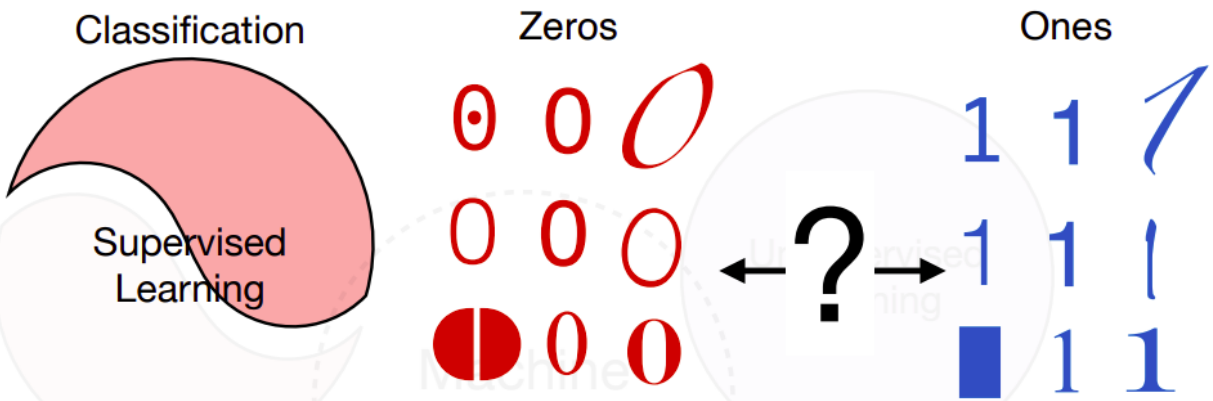

Classification
Supervised Learning
Zeros
Ones


Classification
Supervised Learning
Zeros
Ones
?

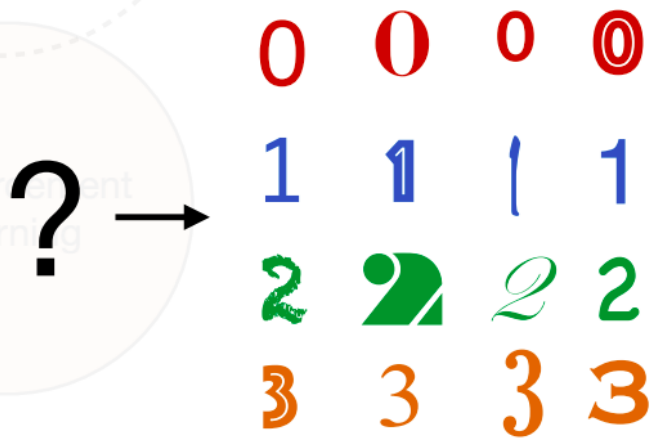
Binary Classification

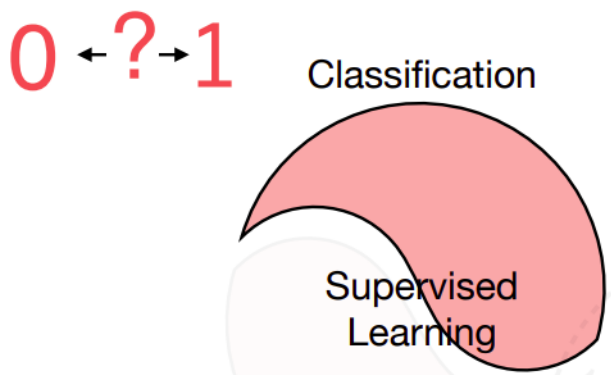


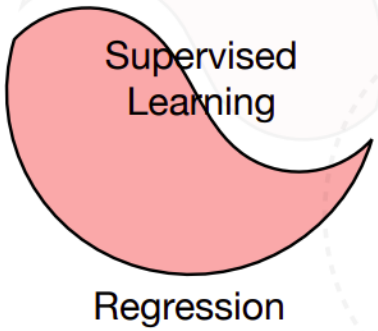
Binary Classification

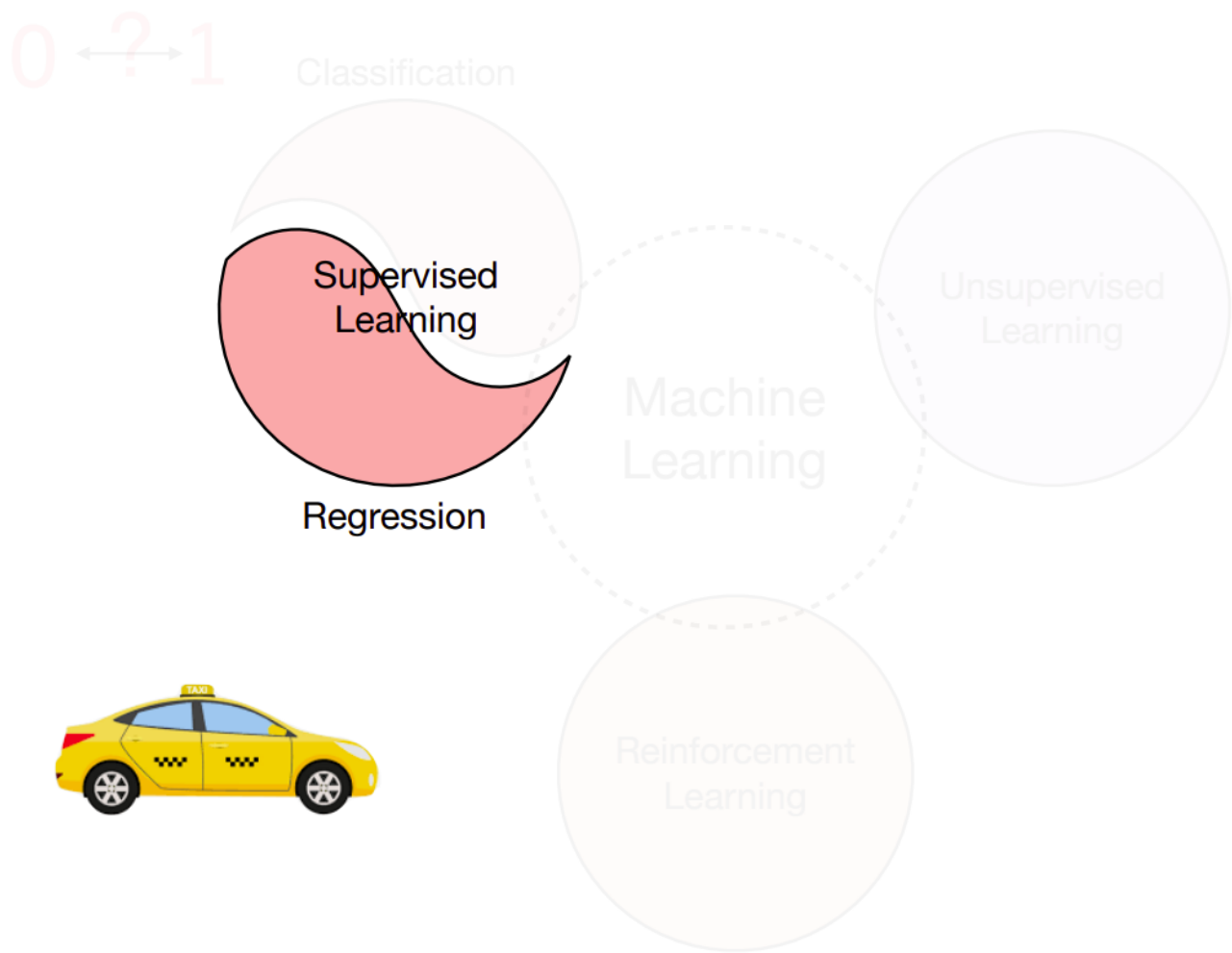


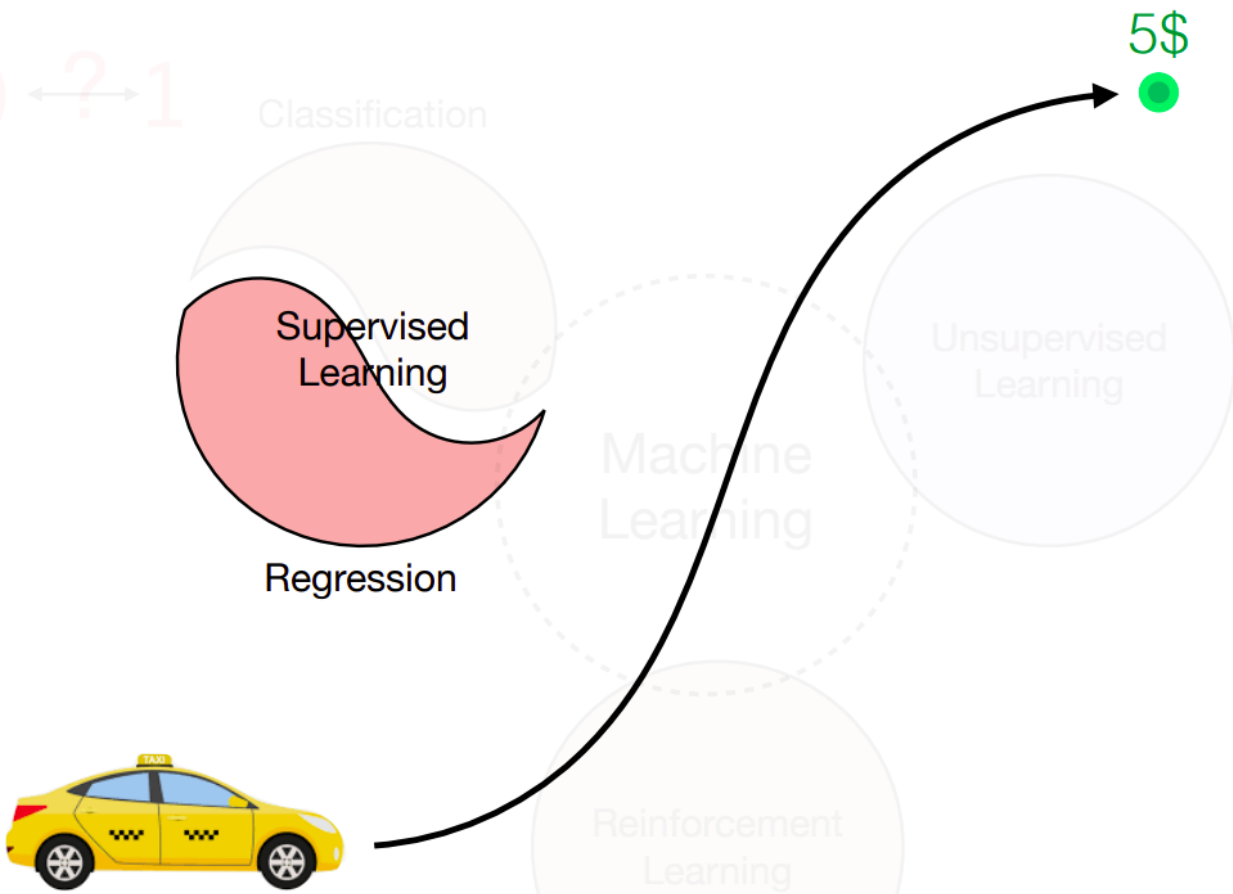
Multiclass Classification

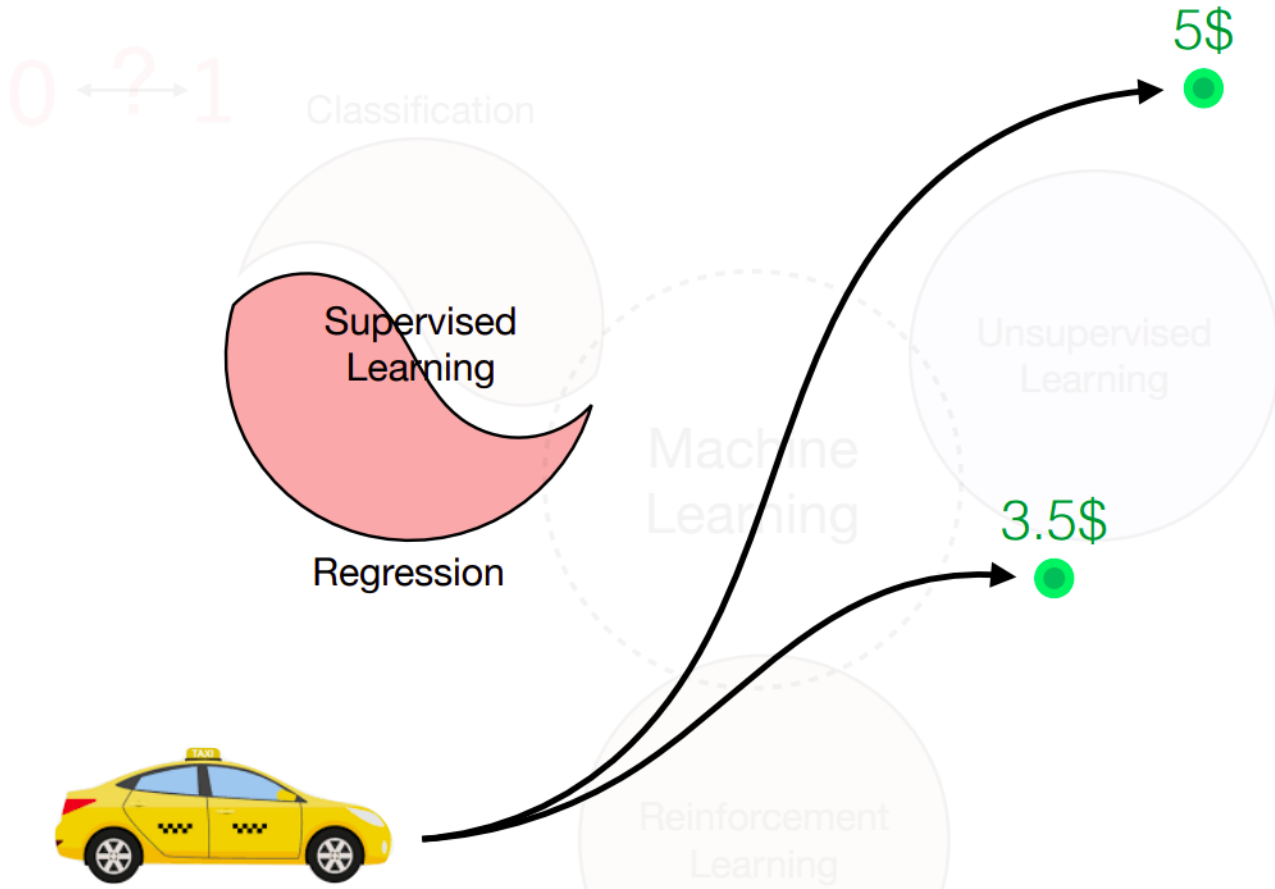


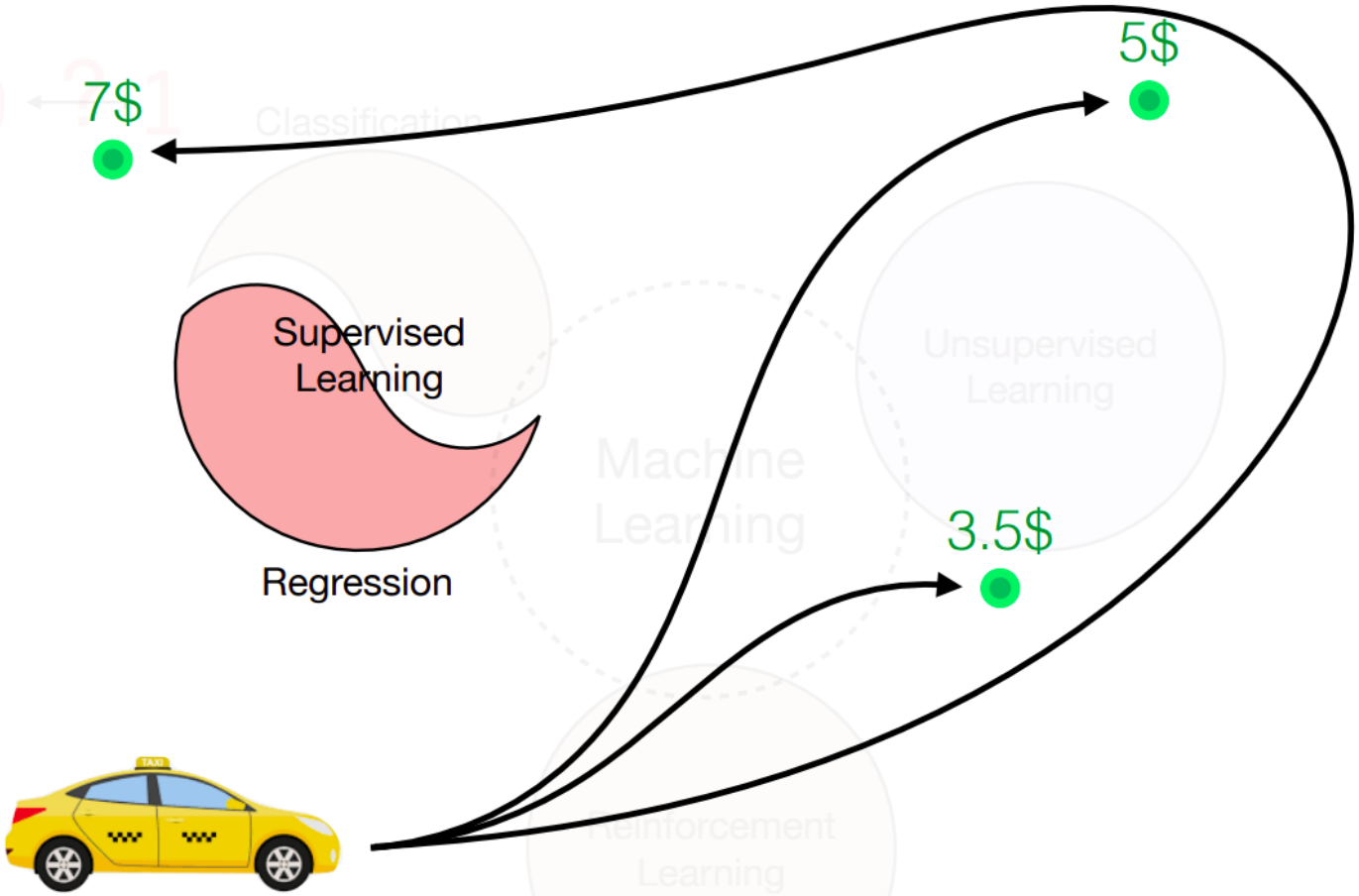

0 ← ? → 1
Classification
Supervised Learning

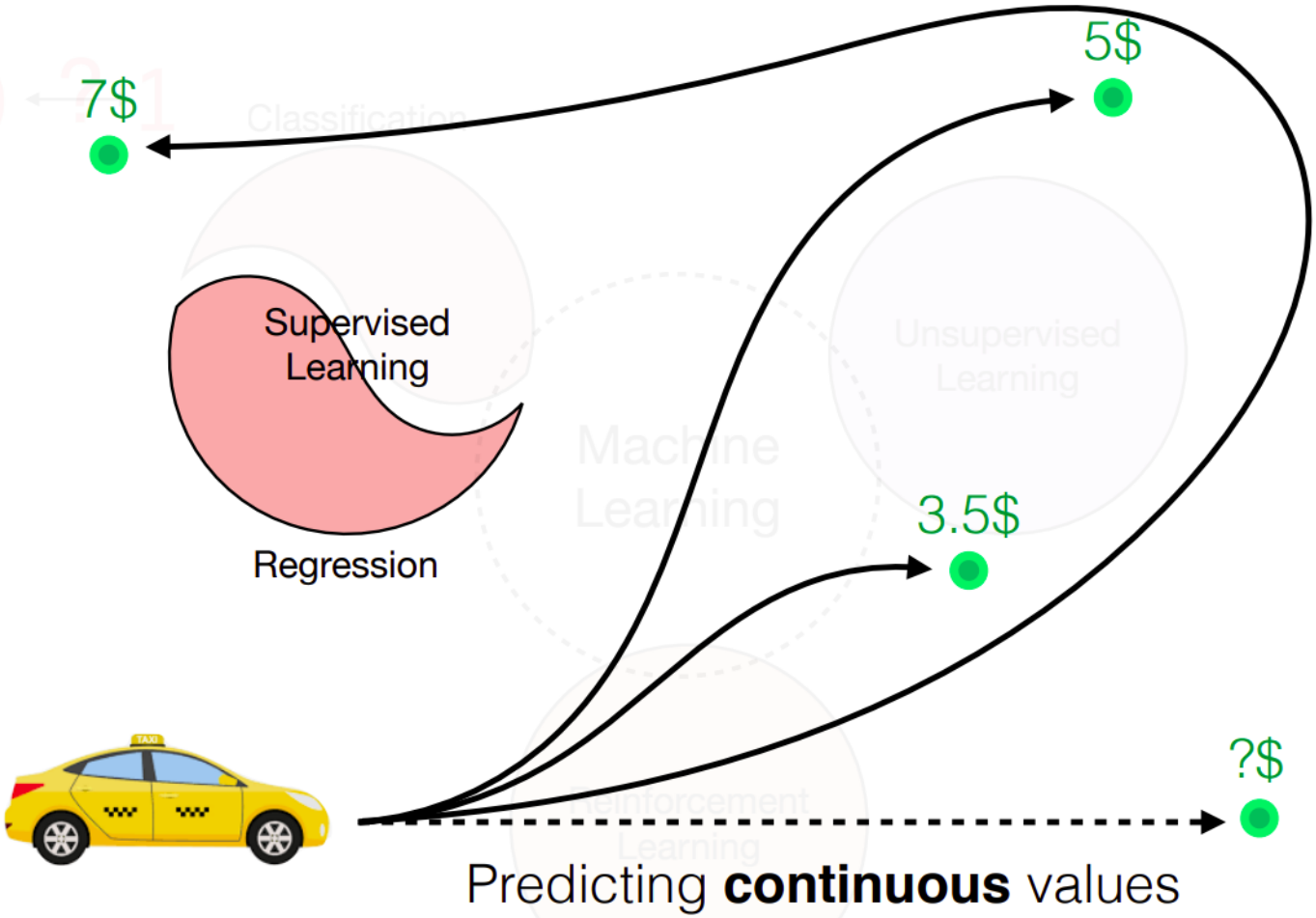

Supervised
Learning
Regression

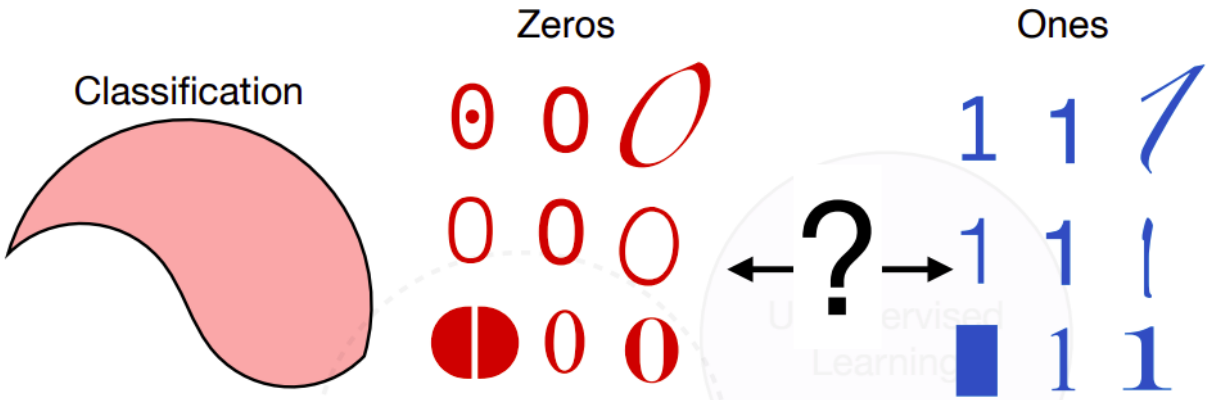

0 ←?→ 1
Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning


5$
Supervised
Learning
Regression

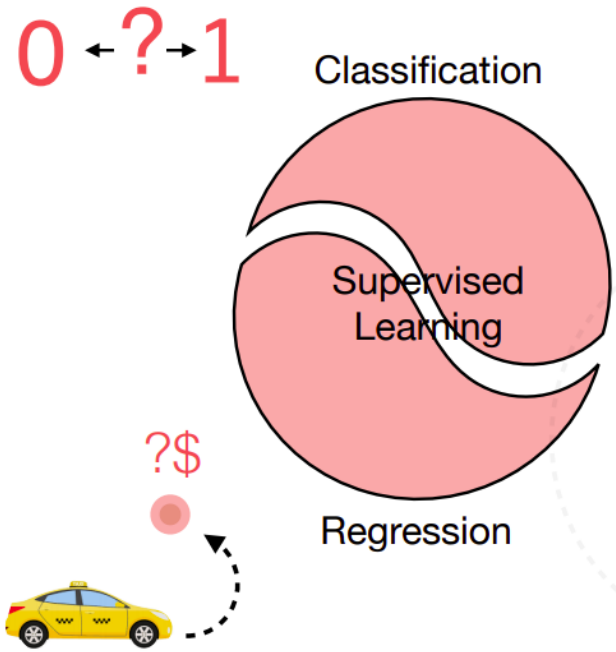

5$
Supervised
Learning
3.5$
Regression

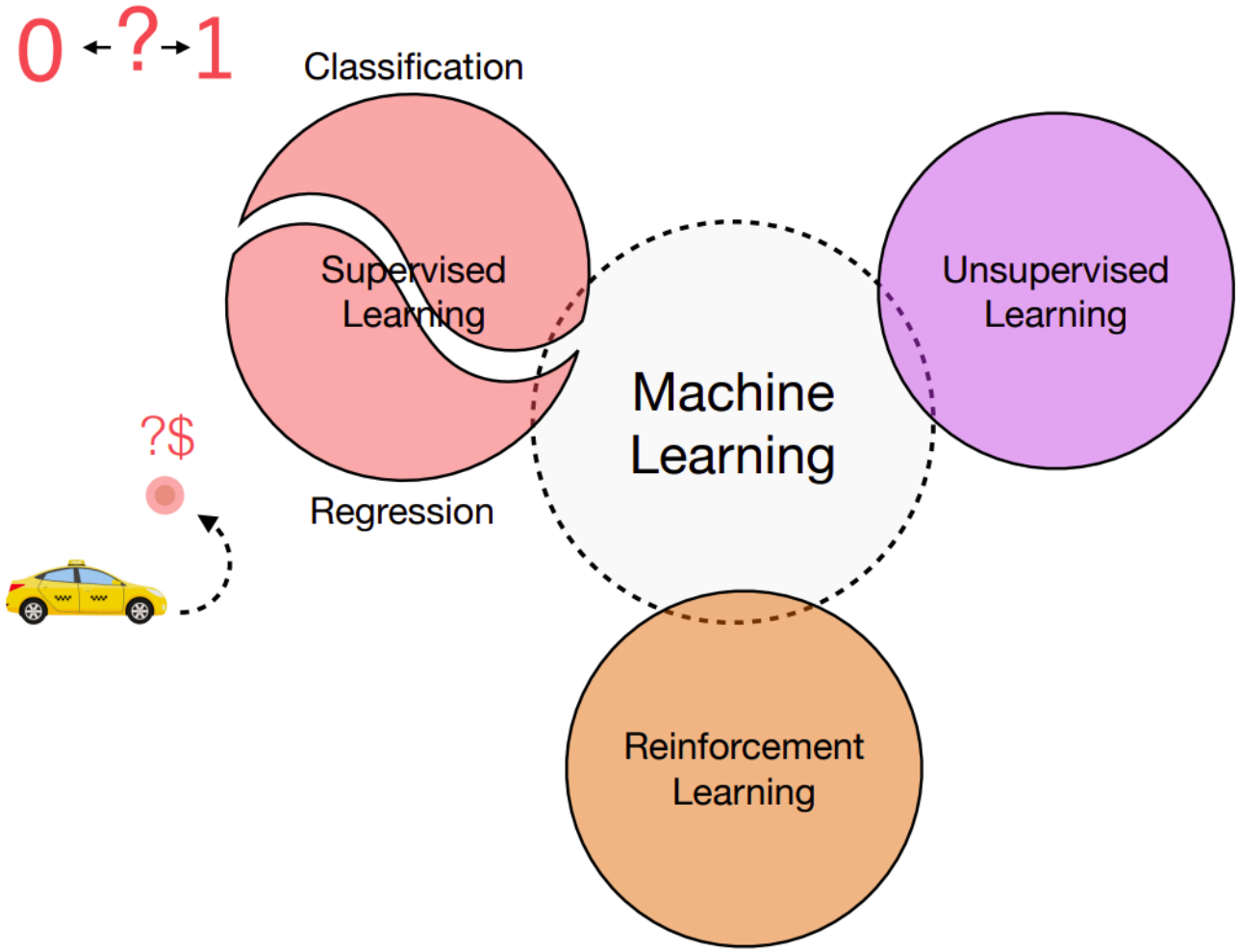

7$
5$
3.5$
Supervised
Learning
Regression

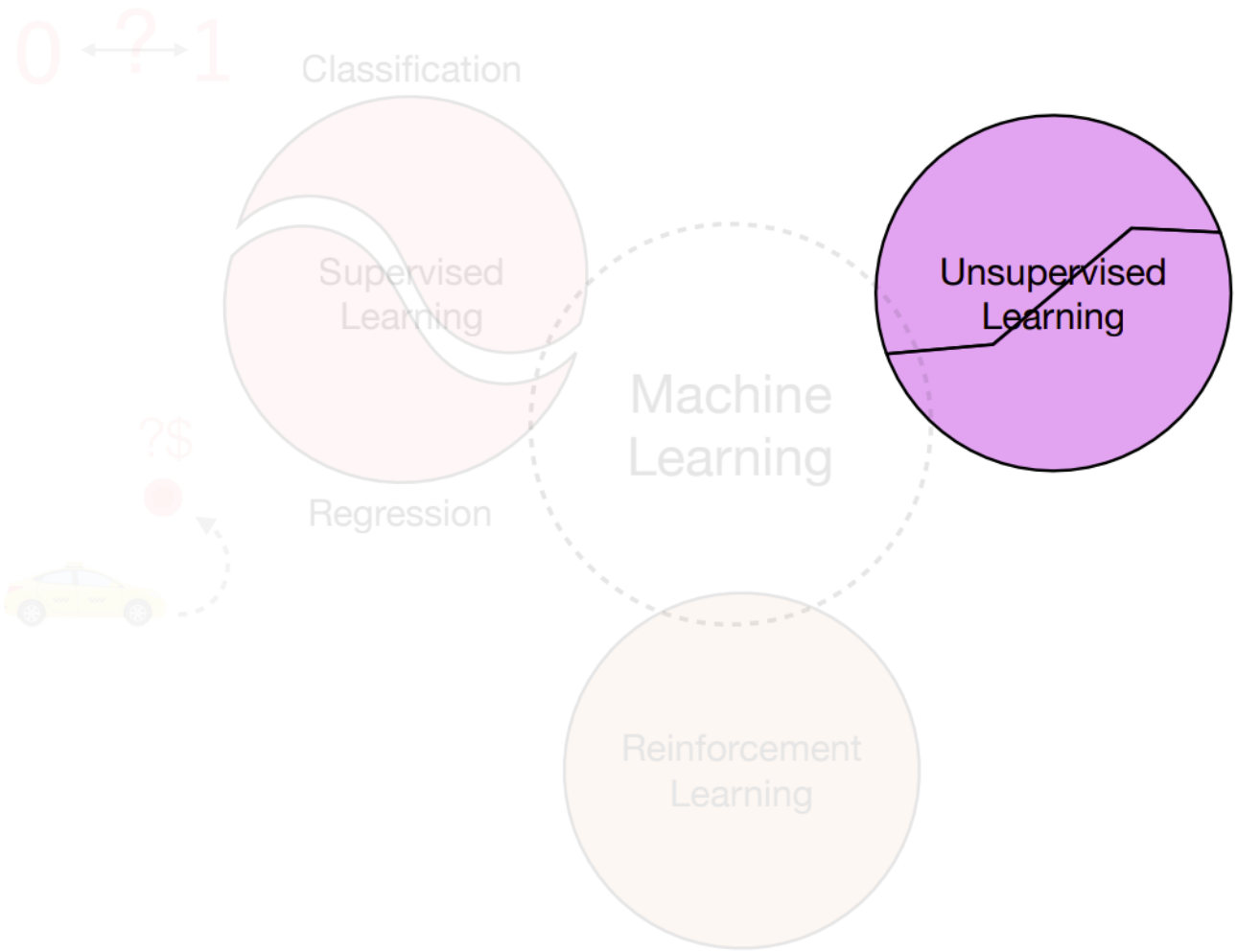

7$
5$
Supervised
Learning
Regression
3.5$
?$
Predicting **continuous** values

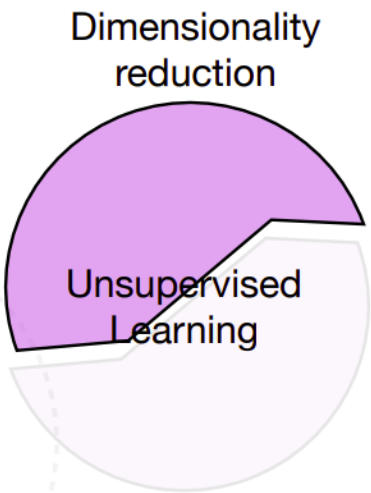


What is the main similarity/difference between these two classes of **supervised learning**?

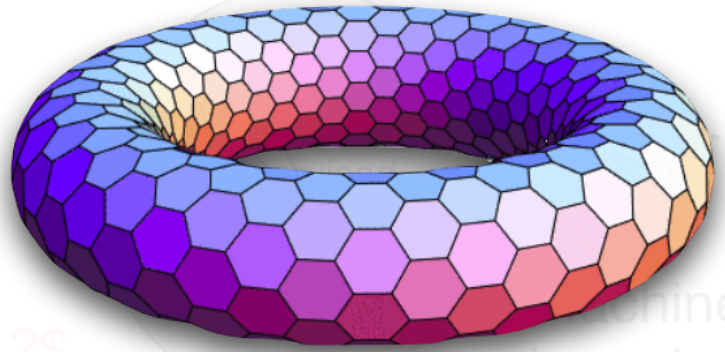

0 ? 1
Classification
Supervised Learning
?$
Regression


0 ← ? → 1
Classification
Supervised Learning
Regression
?$
Machine Learning
Unsupervised Learning
Reinforcement Learning

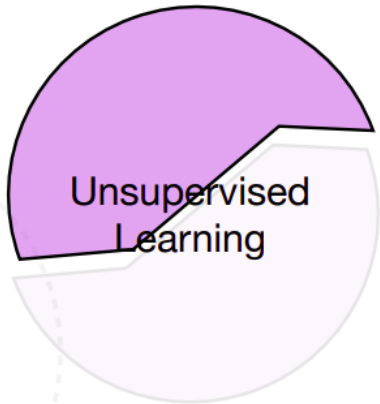

0 ? 1
Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
?$
Reinforcement Learning


Dimensionality
reduction
Unsupervised
Learning

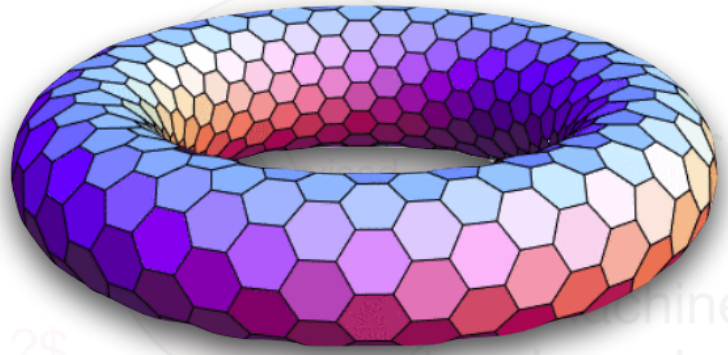
High Dimensional Space



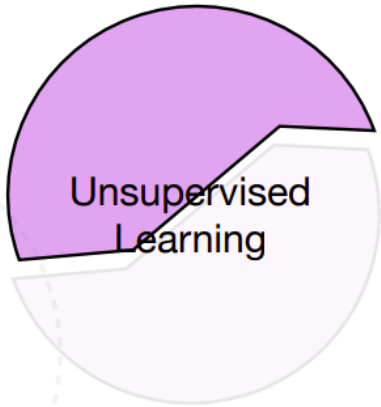
Dimensionality reduction

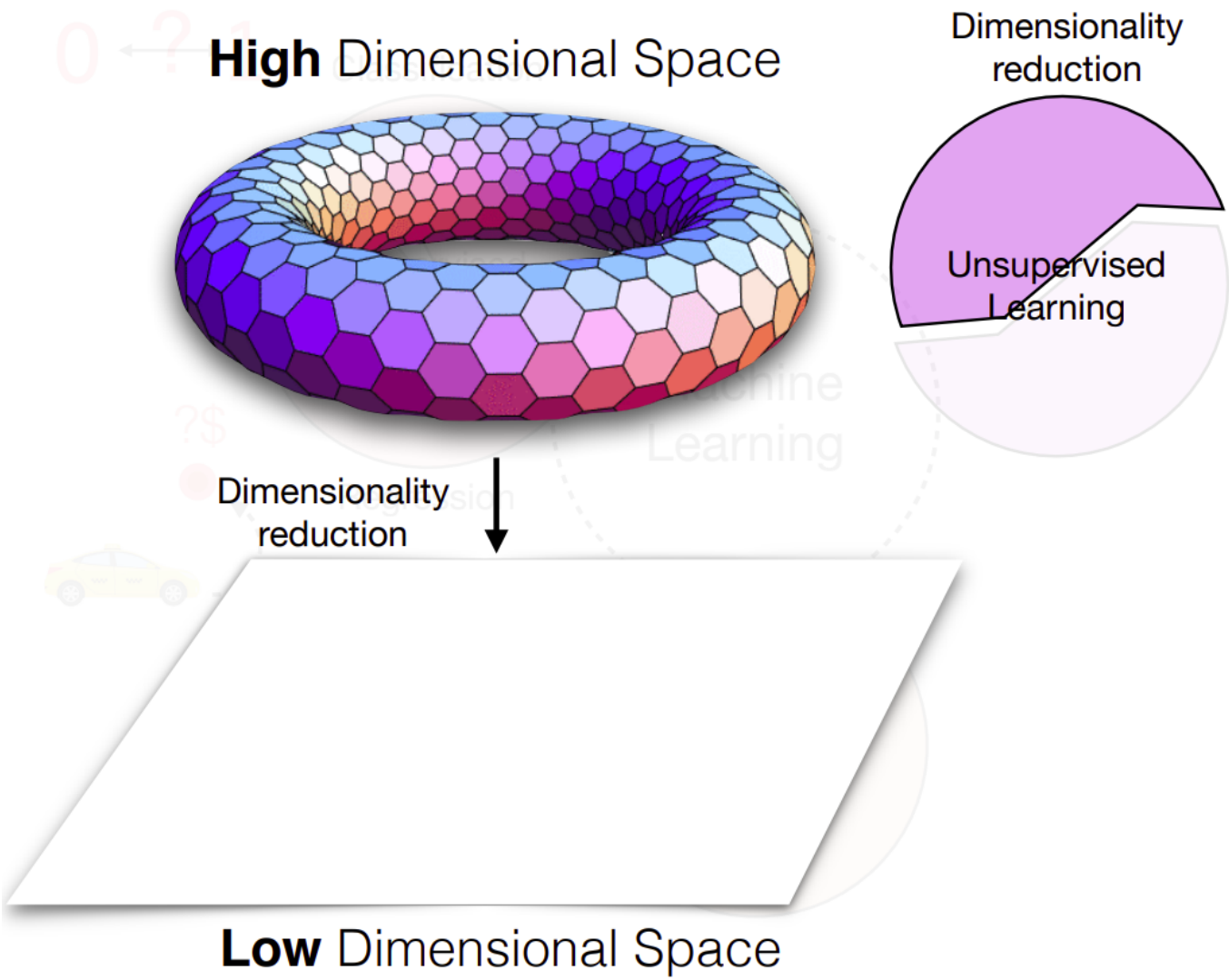


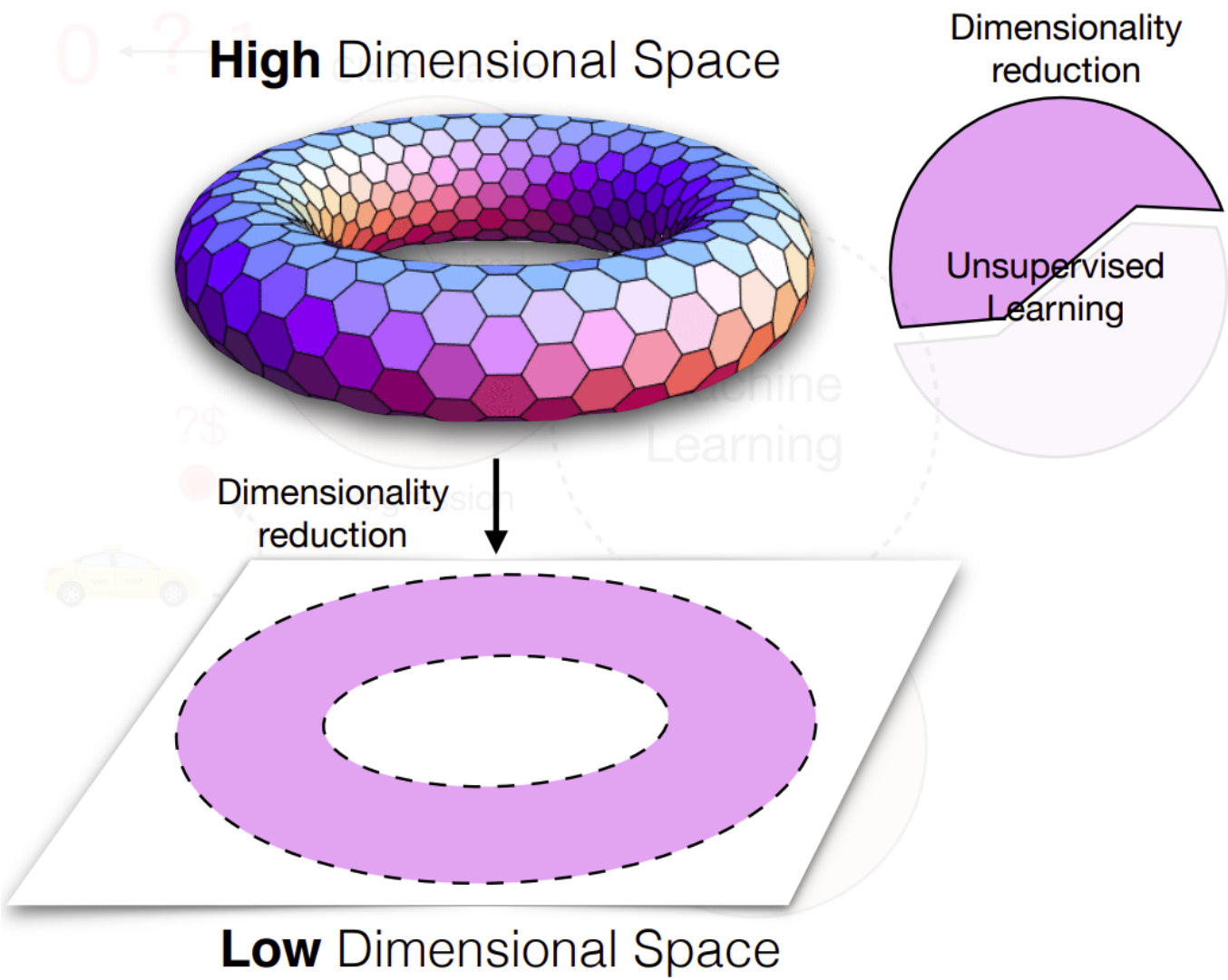
High Dimensional Space

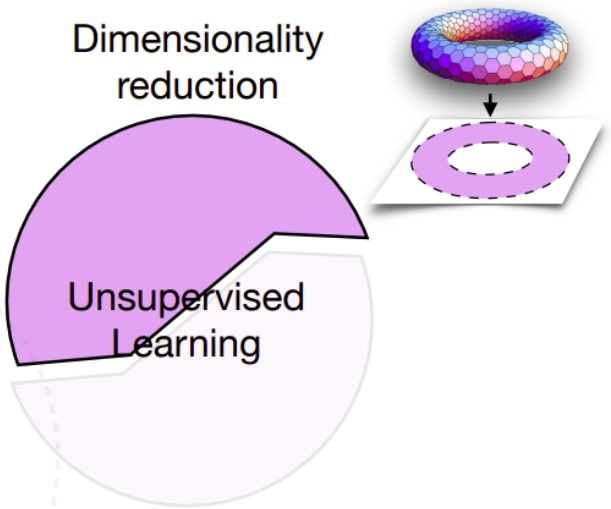


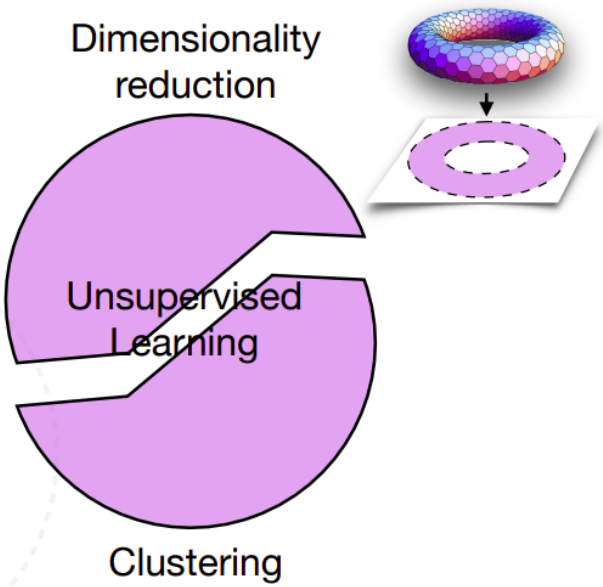
Dimensionality reduction

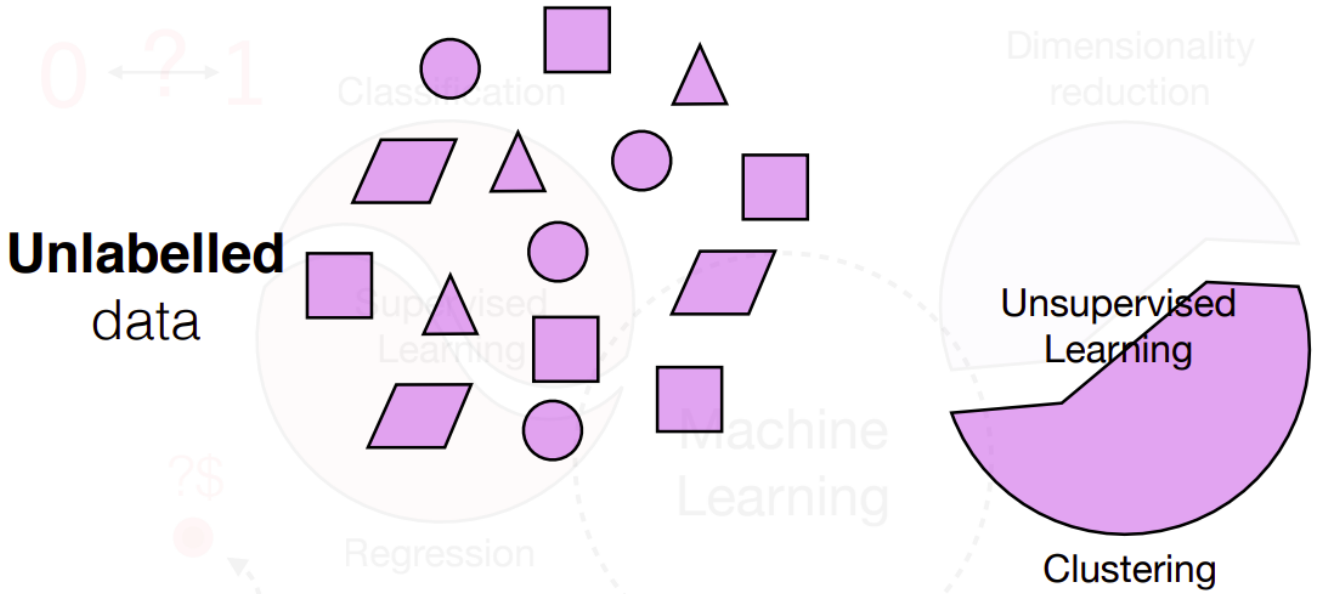


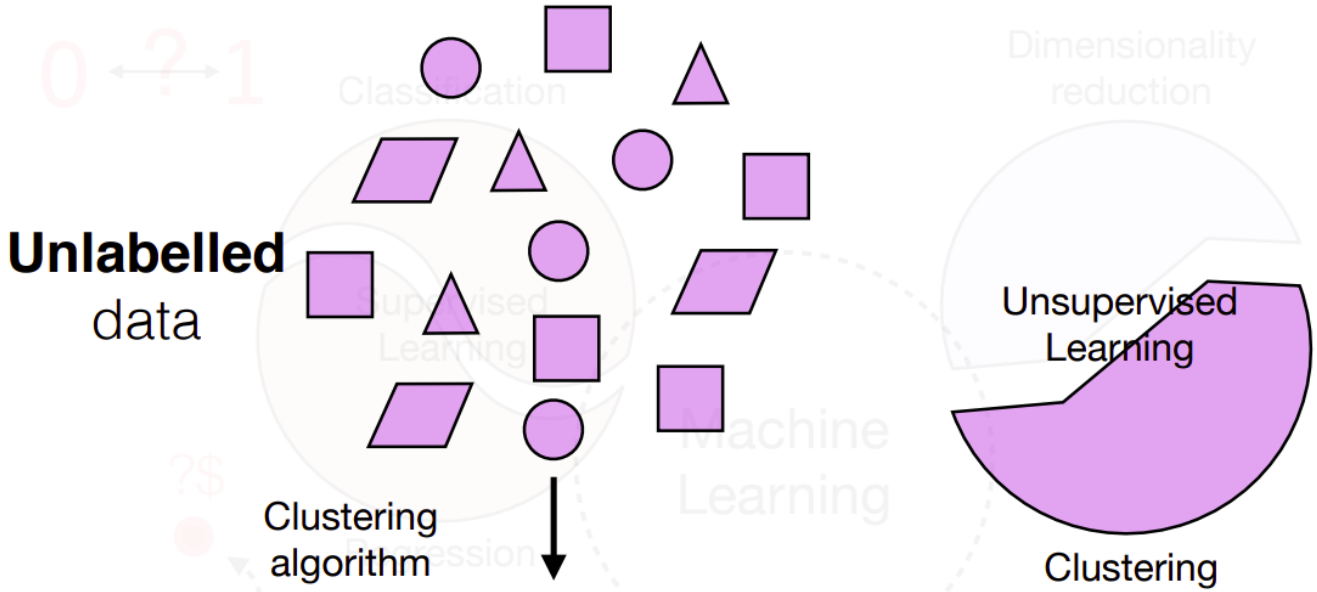

High Dimensional Space
Dimensionality reduction
Unsupervised Learning
Dimensionality reduction
Low Dimensional Space

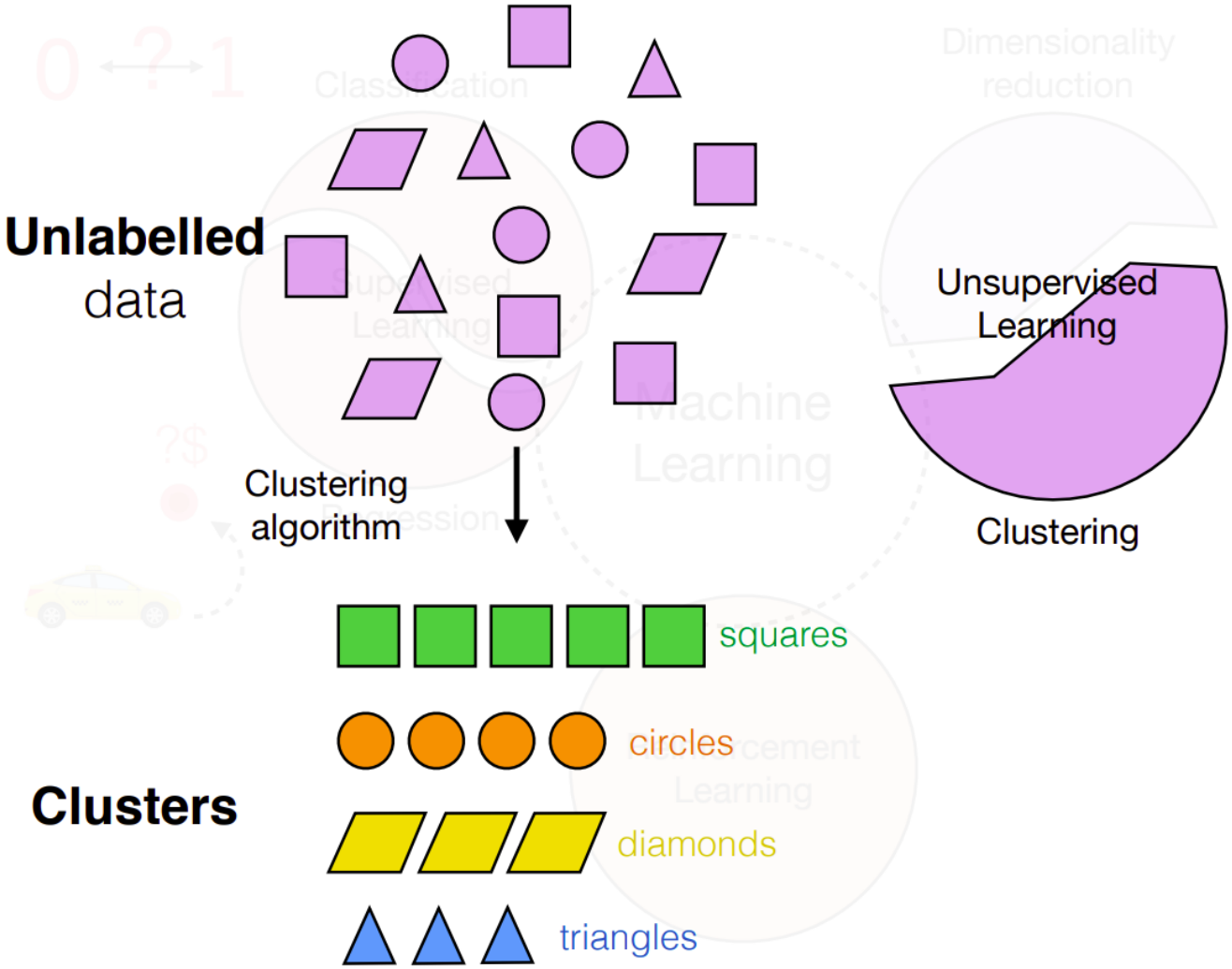

High Dimensional Space
Dimensionality reduction
Unsupervised Learning
Dimensionality reduction
Low Dimensional Space

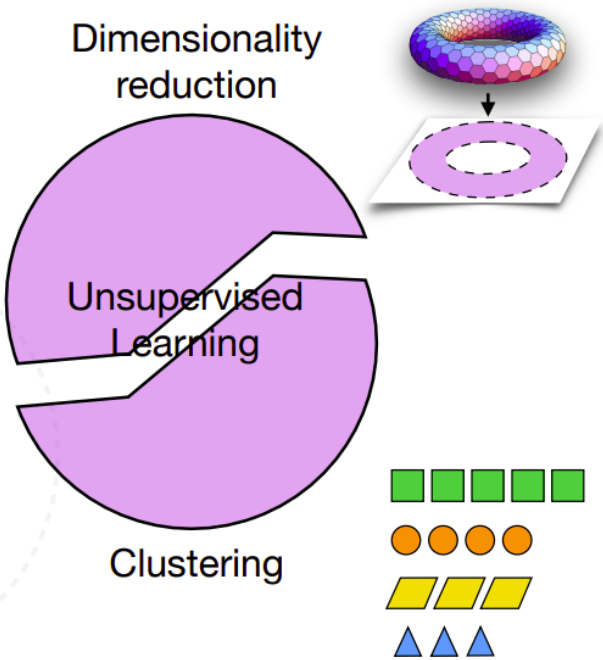

Dimensionality
reduction
Unsupervised
Learning

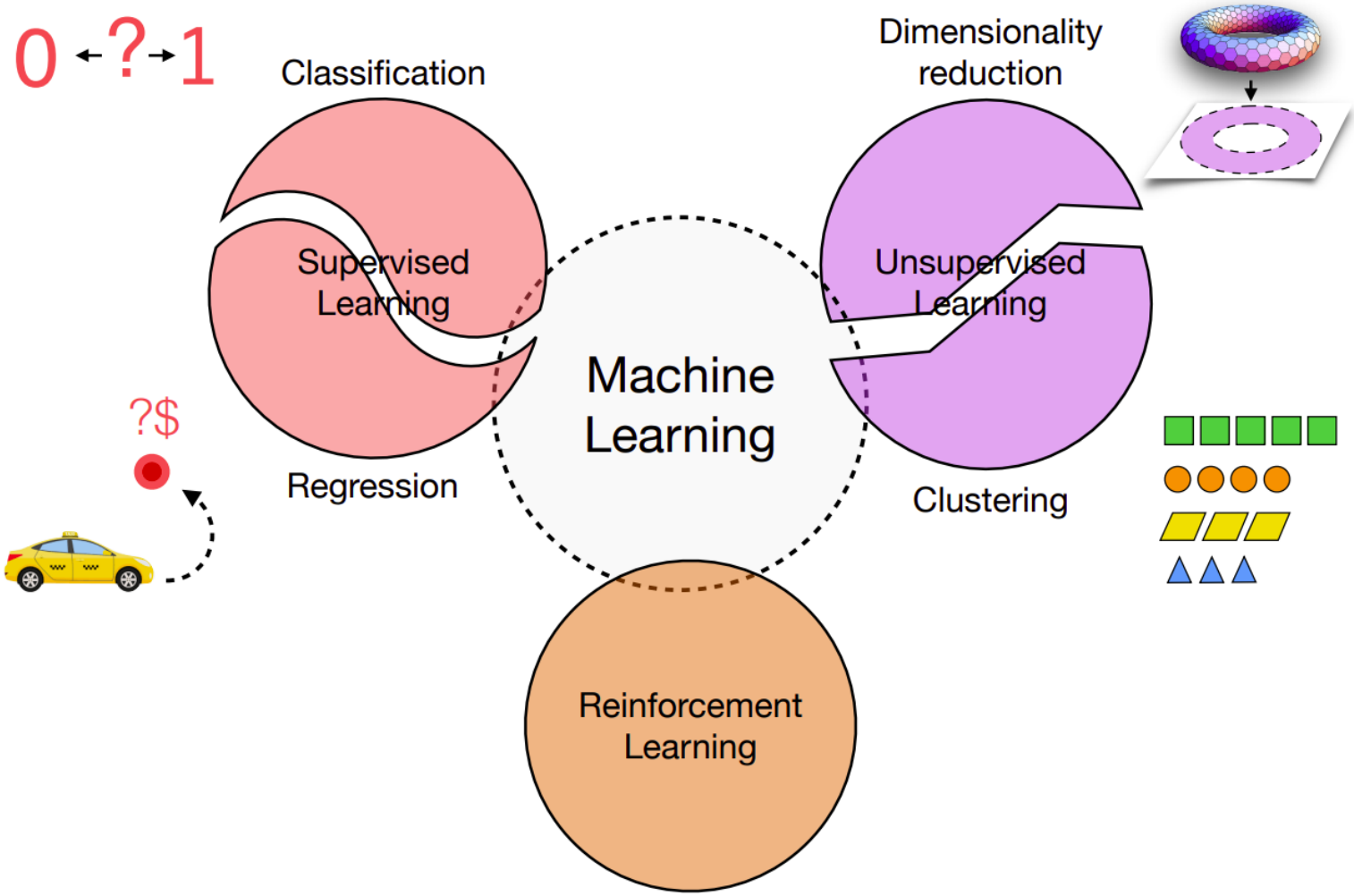

Dimensionality
reduction
Unsupervised
Learning
Clustering

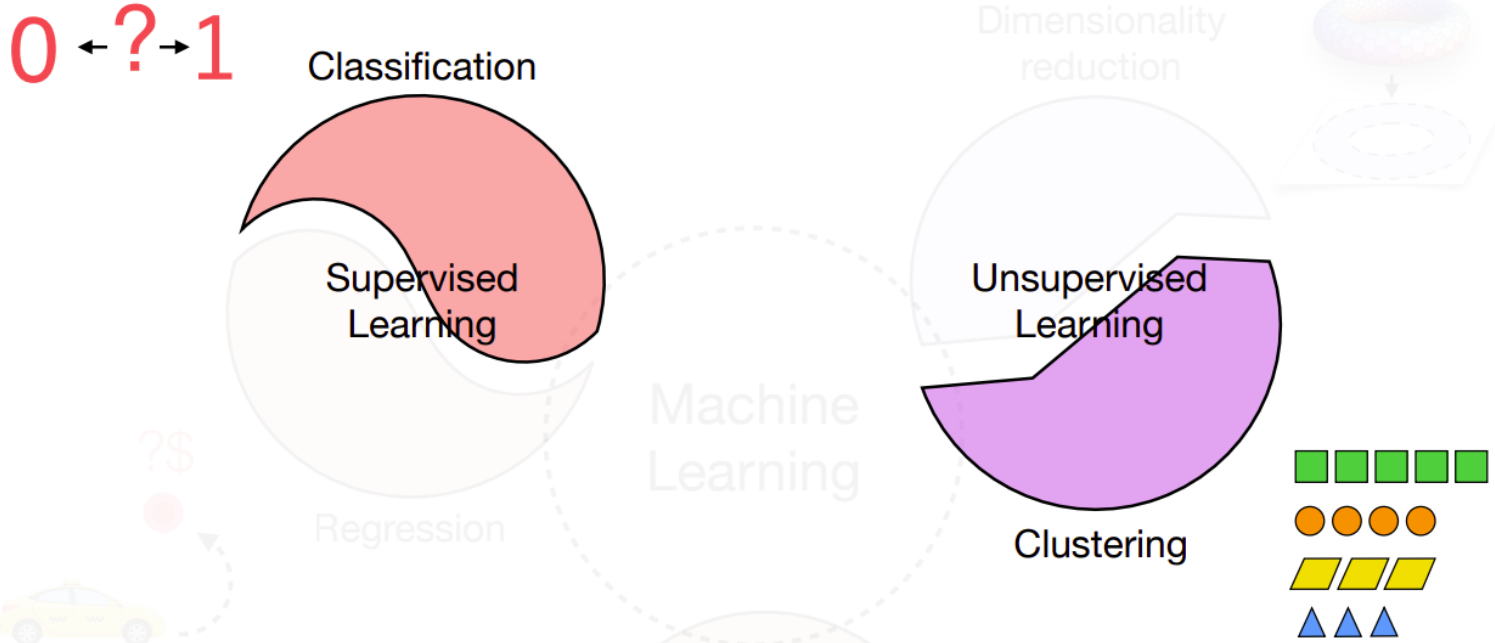

Unlabelled
data
Unsupervised
Learning
Clustering


Unlabelled
data
Clustering
algorithm
Unsupervised
Learning
Clustering

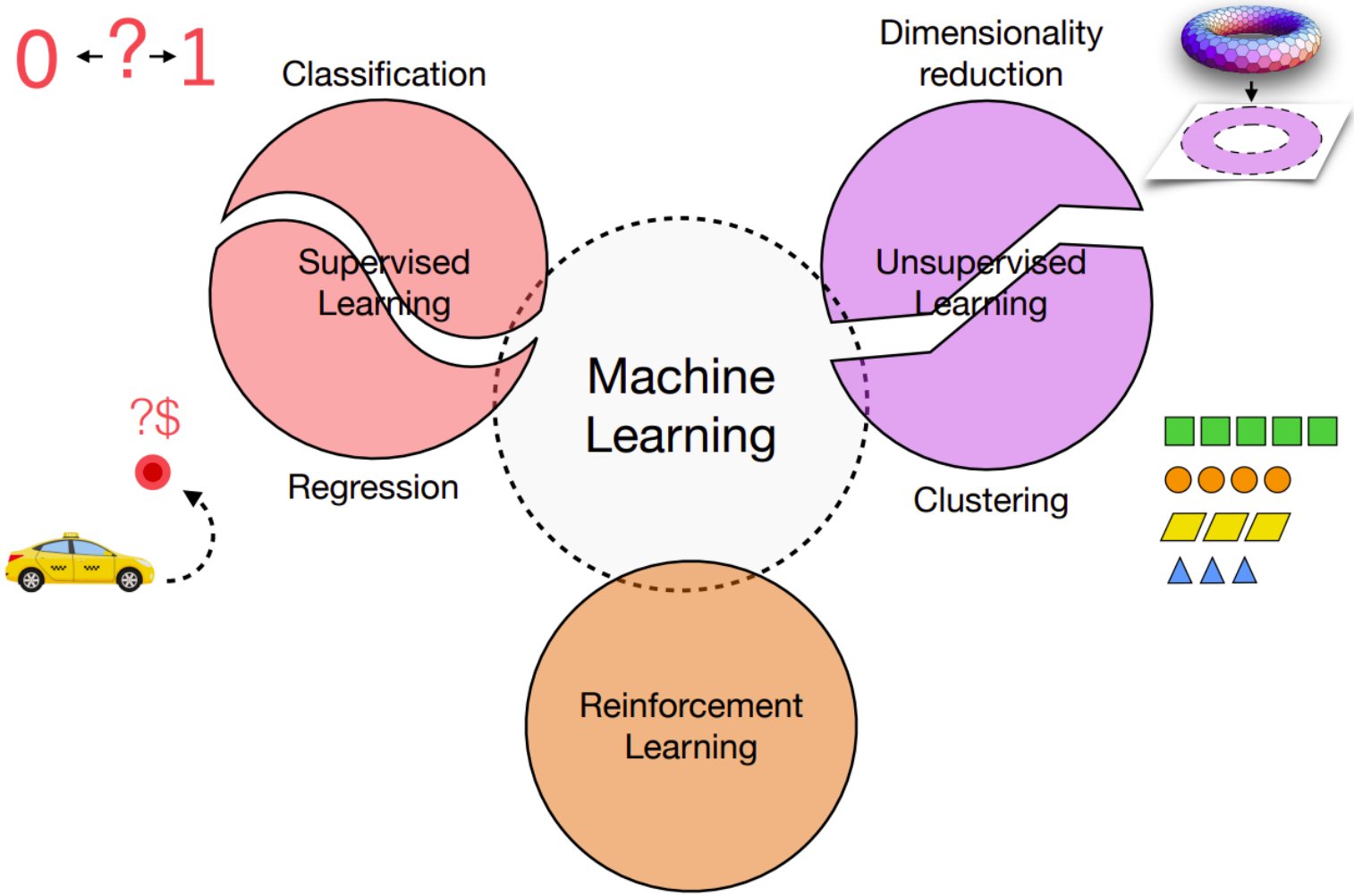

Unlabelled
data
Clustering
algorithm
Clusters
squares
circles
diamonds
triangles
Unsupervised
Learning
Clustering

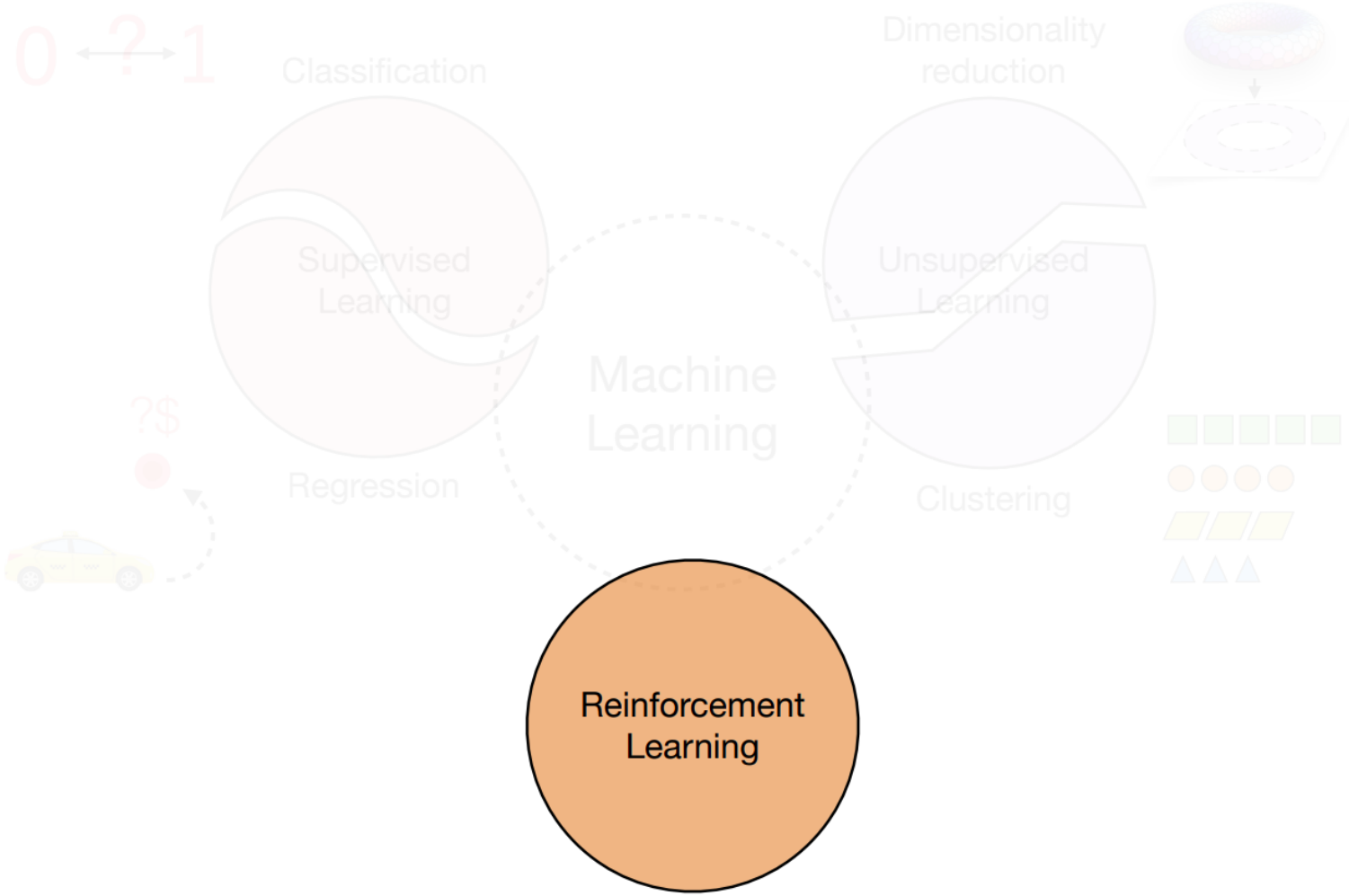

Dimensionality
reduction
Unsupervised
Learning
Clustering

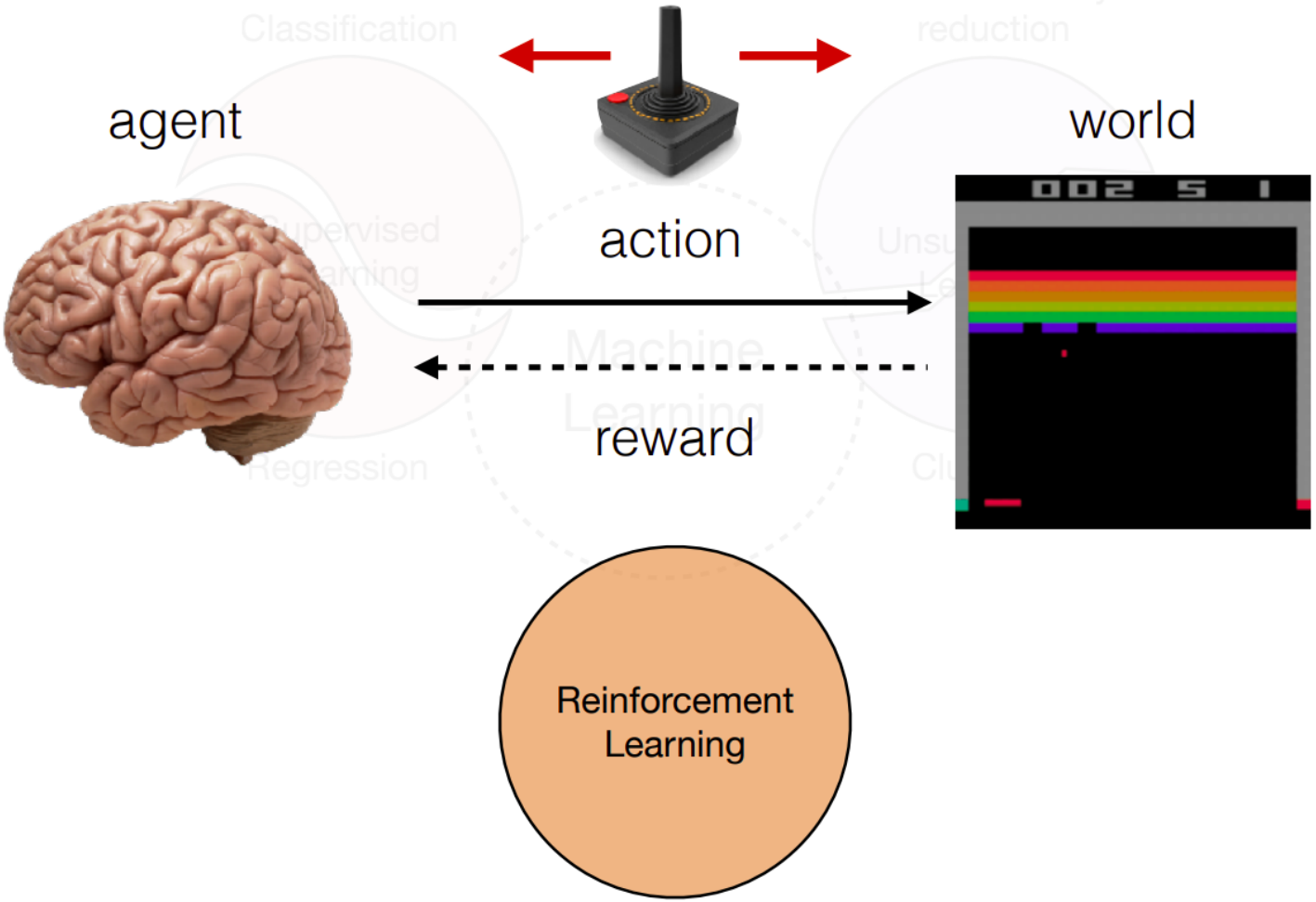

0 ← ? → 1
Classification
Supervised Learning
Regression
Dimensionality reduction
Unsupervised Learning
Clustering
Machine Learning
Reinforcement Learning
?$

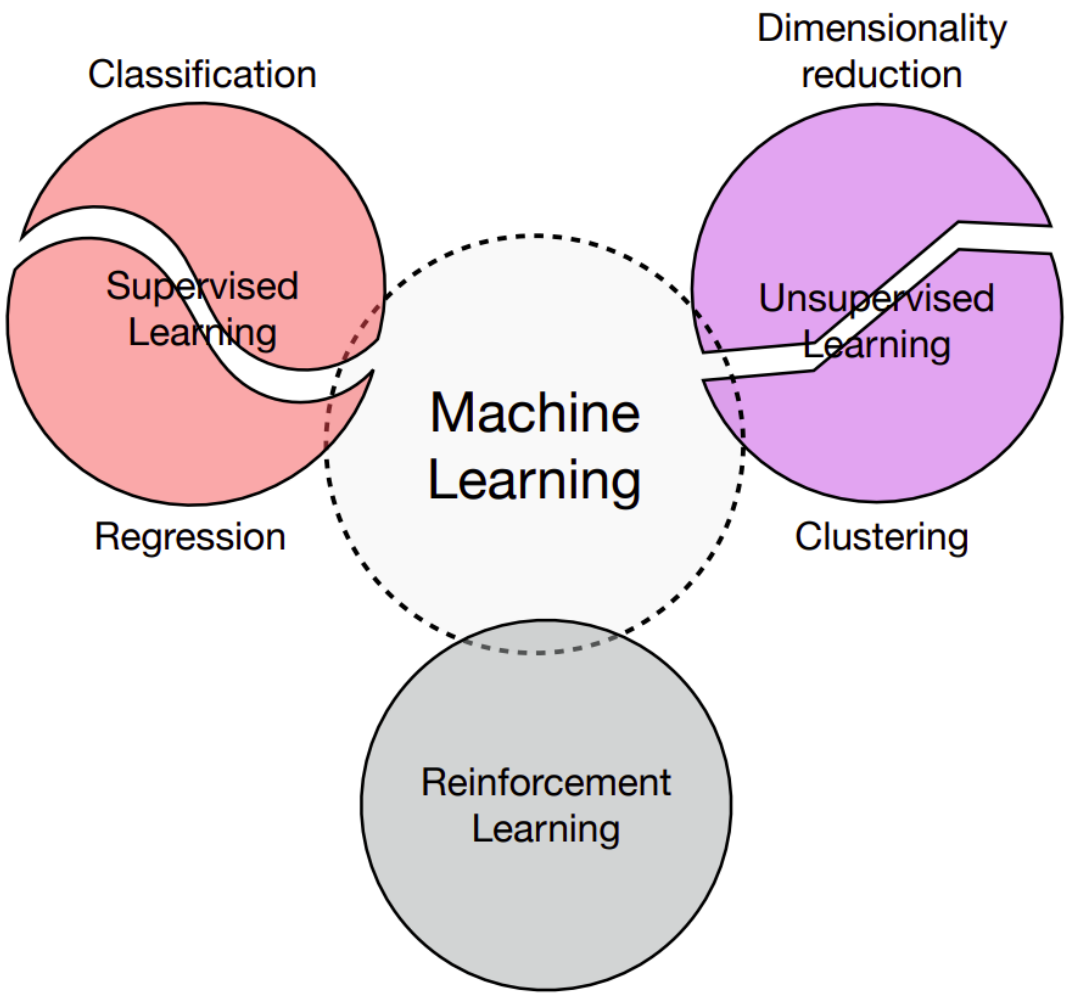


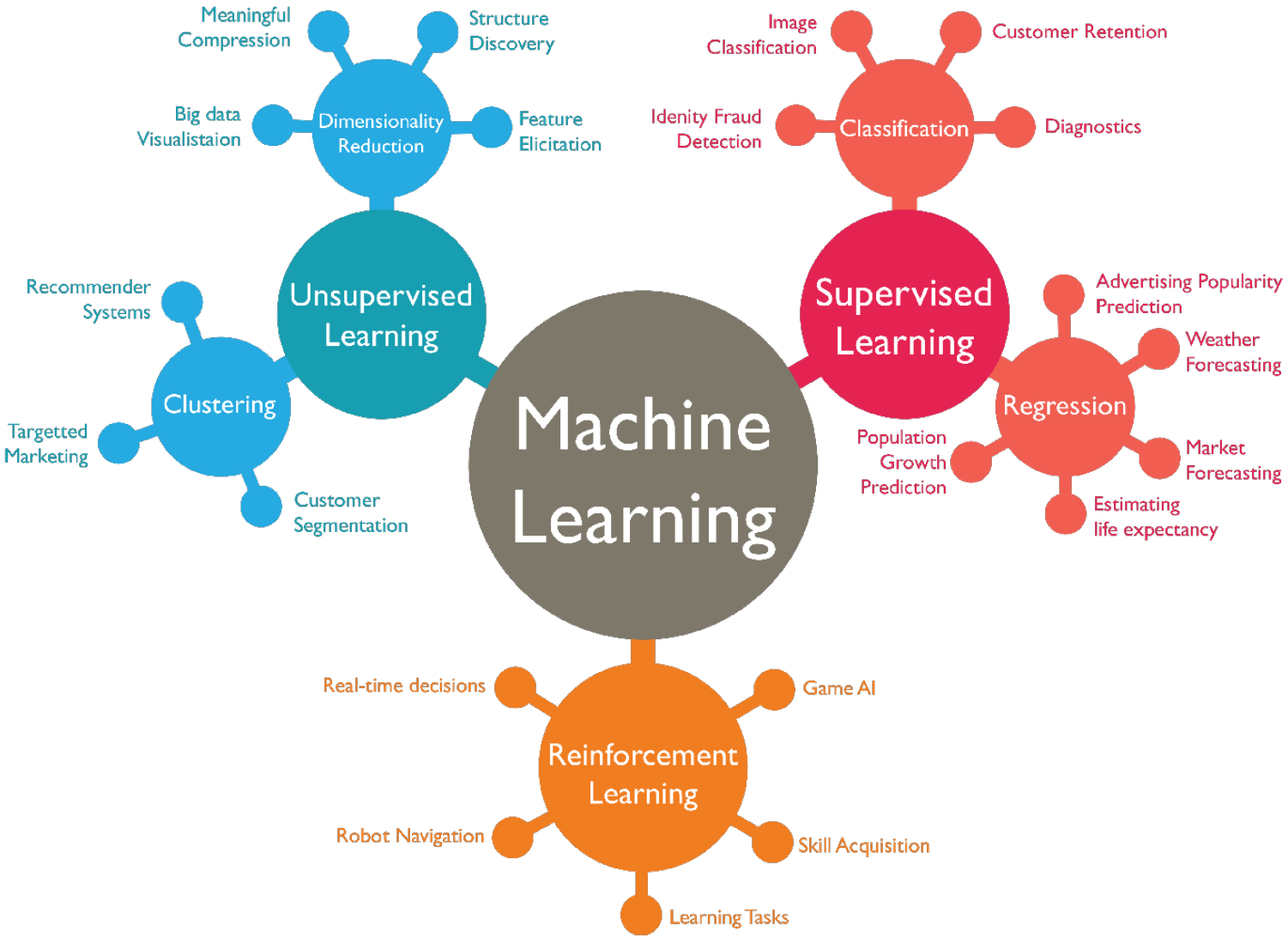
What is the main similarity/difference between **classification** and **clustering**?


0 ← ? → 1
Classification
Supervised Learning
Regression
Dimensionality reduction
Unsupervised Learning
Clustering
Machine Learning
?$
Reinforcement Learning


0 ? 1
Classification
Dimensionality reduction
Supervised Learning
Unsupervised Learning
Machine Learning
?$
Regression
Clustering
Reinforcement Learning


agent
world
action
reward
Reinforcement Learning


Classification
Dimensionality reduction
Supervised Learning
Unsupervised Learning
Machine Learning
Regression
Clustering
Reinforcement Learning


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Estimating life expectancy
Population Growth Prediction
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Course Syllabus

- K-Nearest Neighbours

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees
- Ensemble Methods

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees
- Ensemble Methods
- Evaluation and scoring of classifiers

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees
- Ensemble Methods
- Evaluation and scoring of classifiers
- Linear and Logistic Regression

Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees
- Ensemble Methods
- Evaluation and scoring of classifiers
- Linear and Logistic Regression
- Regression Trees and Gradient Boosting

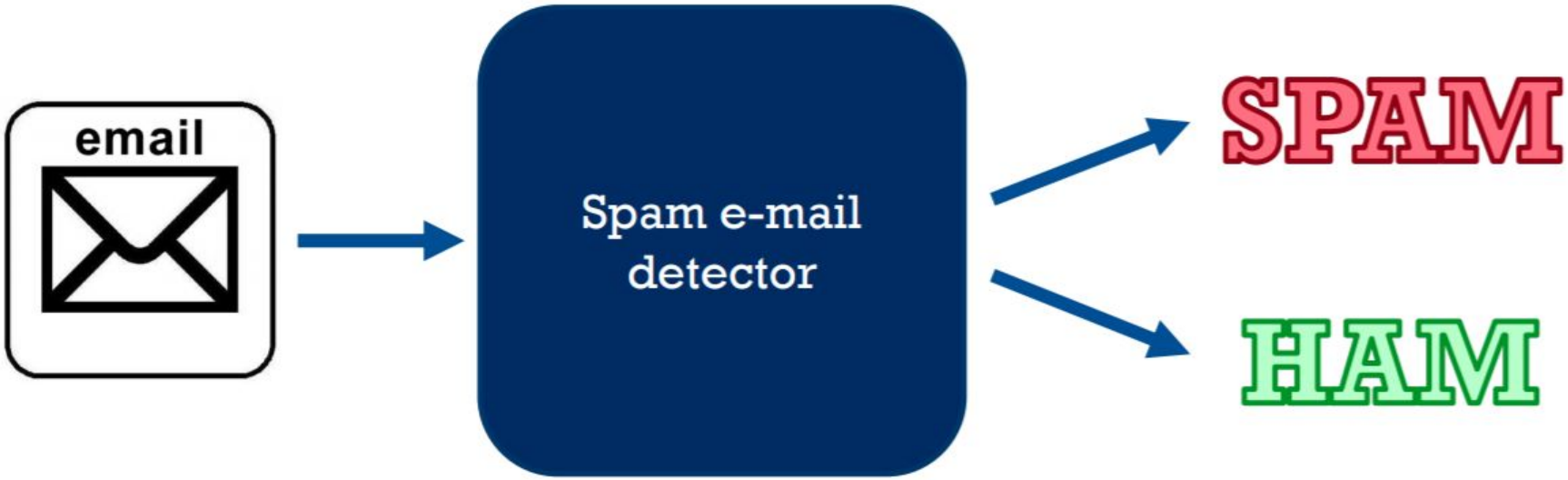
Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees
- Ensemble Methods
- Evaluation and scoring of classifiers
- Linear and Logistic Regression
- Regression Trees and Gradient Boosting
- K-Means and Hierarchical Clustering

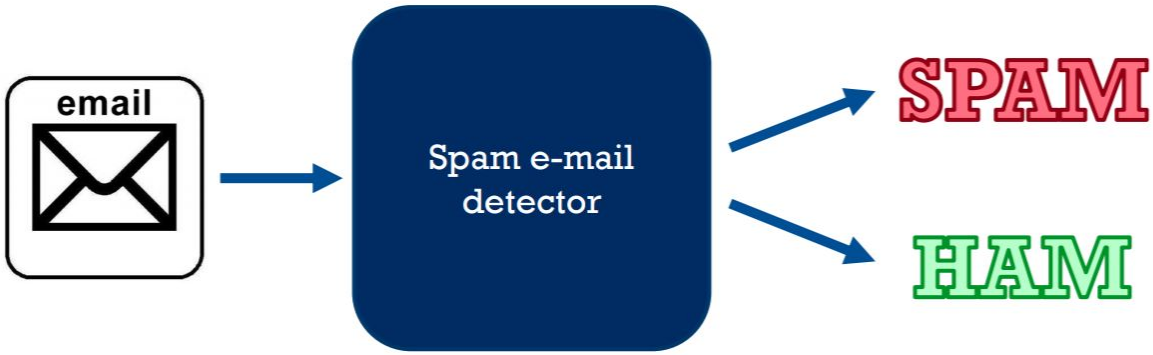
Course Syllabus

- K-Nearest Neighbours
- Naive Bayes
- Linear & Quadratic Discriminant Analysis
- Support Vector Machines and Kernel Methods
- Decision Trees
- Ensemble Methods
- Evaluation and scoring of classifiers
- Linear and Logistic Regression
- Regression Trees and Gradient Boosting
- K-Means and Hierarchical Clustering
- Principal Component Analysis

Spam e-mail detector



Machine Learning Terminology



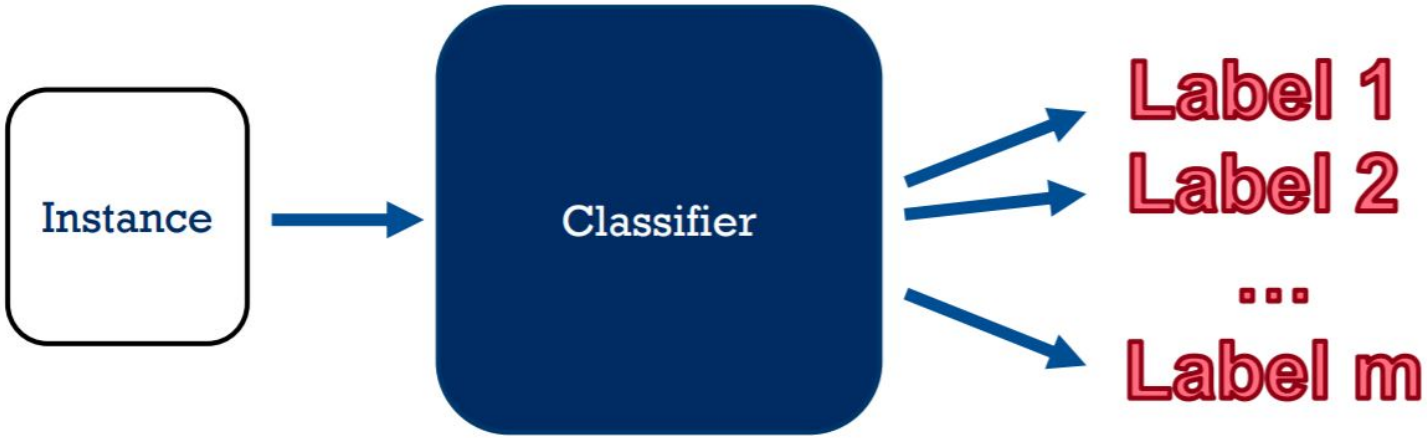
Machine Learning Terminology

- Binary classifier (spam e-mail detector)
  - Inputs an instance (e-mail)
  - Predicts positive or negative label (spam/ham)

Examples of binary classification

- Instance = photo, Label = cat / dog
- Instance = audio, Label = guitar / ukulele
- Instance = text, Label = English / Armenian

Classification



Here m is the number of different labels:

- Binary classifier: $m = 2$
- Multi-class classifier: $m \geq 2$

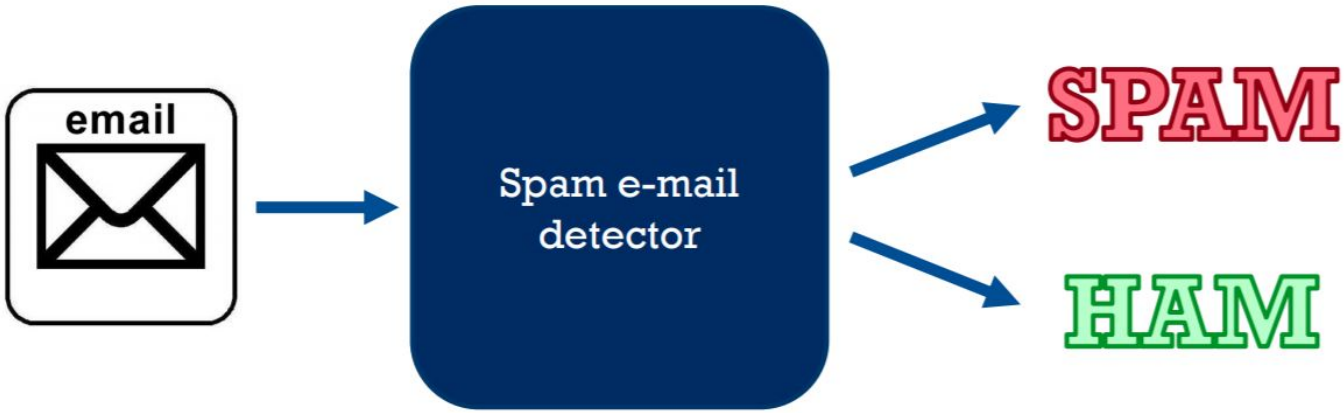
Mathematical notation of classification

- Notations can very in the literature
- $\mathbb{X}$ - input space (set of all possible instances)
- $\mathbb{Y}$ - output space (all possible labels)
- $f : \mathbb{X} \to \mathbb{Y}$ - any such function is a classifier
- $\mathbf{x} \in \mathbb{X}$ - instance
- $y \in \mathbb{Y}$ - actual / true label of instance $\mathbf{x}$
- $\hat{y} = f(\mathbf{x})$- predicted label of instance $\mathbf{x}$

Actual / true label

- In classification we assume that every instance has an actual / true label y
- A classifier is perfect if it is always correct, i.e. it predicts the actual label: $\hat{y} = y$
- Usually classifiers make some errors: $\hat{y} \neq y$

How to build a spam detector?



- Let us think of how to build a spam detector
- Without machine learning for now

Building a spam detector

- What is a typical spam e-mail?

Building a spam detector

- What is a typical spam e-mail?
  - Advertising something that we don't want

Building a spam detector

- What is a typical spam e-mail?
  - Advertising something that we don't want
  - Trying to steal our money

Building a spam detector

- What is a typical spam e-mail?
  - Advertising something that we don't want
  - Trying to steal our money
  - Trying to infect our computer with a virus
  - ...

Building a spam detector

- What is a typical spam e-mail?
  - Advertising something that we don't want
  - Trying to steal our money
  - Trying to infect our computer with a virus
  - ...
- What helps us to discriminate between spam and ham?

Building a spam detector

- What is a typical spam e-mail?
  - Advertising something that we don't want
  - Trying to steal our money
  - Trying to infect our computer with a virus
  - ...
- What helps us to discriminate between spam and ham?
- We need to find features that occur more often in spam than in ham (or vice versa)

100 binary features for spam detection

1. Does the e-mail contain an inline gif-image?
   (yes / no)

100 binary features for spam detection

1. Does the e-mail contain an inline gif-image?
   (yes / no)
2. Does the e-mail contain HTML with more images than text?
   (yes / no)

100 binary features for spam detection

1. Does the e-mail contain an inline gif-image?
   (yes / no)
2. Does the e-mail contain HTML with more images than text?
   (yes / no)
3. Does the sender's hostname contain a long non-vowel sequence?
   (yes / no)

   ...

100 binary features for spam detection

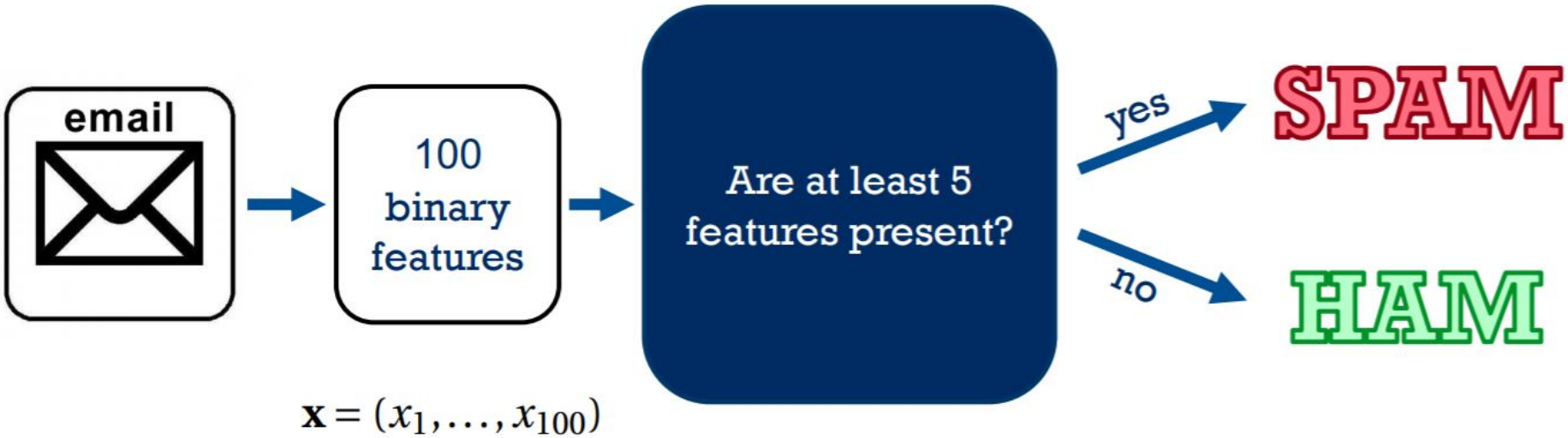
1. Does the e-mail contain an inline gif-image?
   (yes / no)
2. Does the e-mail contain HTML with more images than text?
   (yes / no)
3. Does the sender's hostname contain a long non-vowel sequence?
   (yes / no)

   ...

Our first spam detector

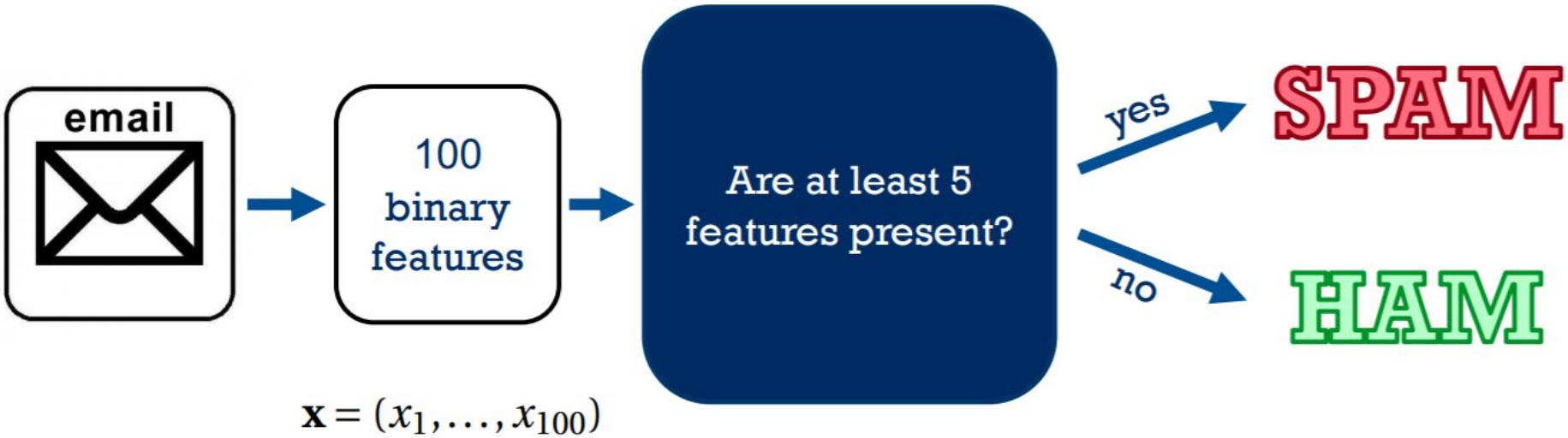
Classify as spam whenever at least 5 features are present.

Building a spam detector



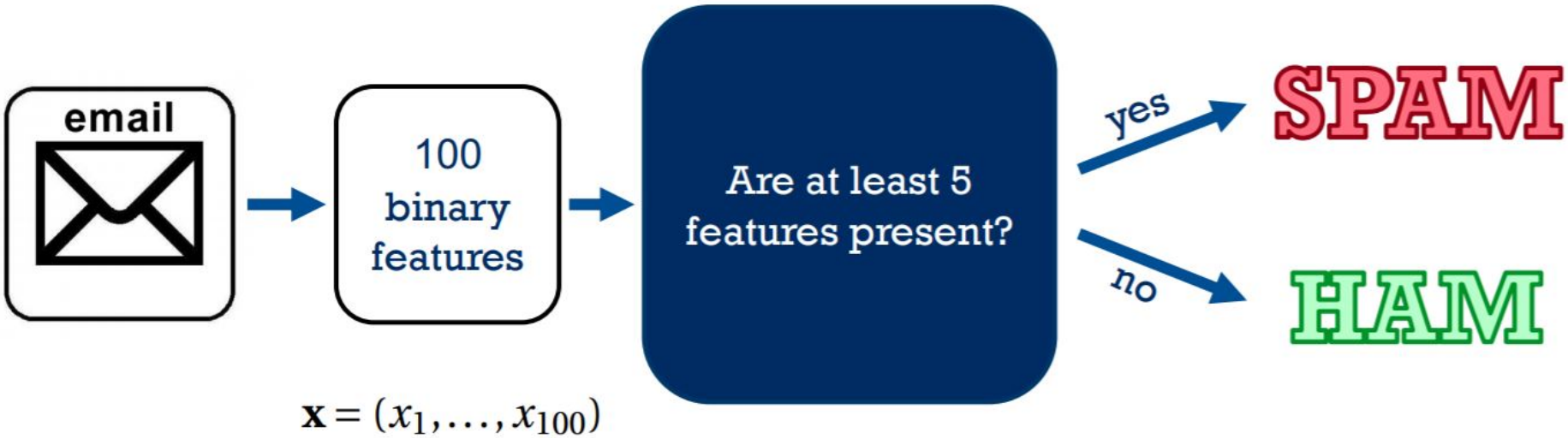
- We will use the following encoding yes=1 and no=0

Building a spam detector



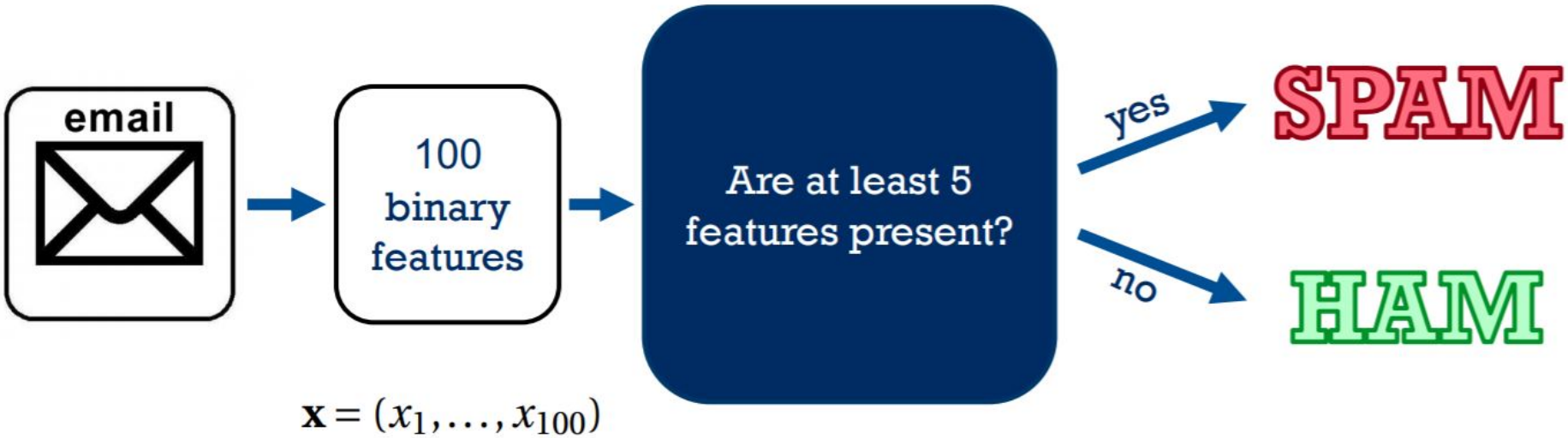
- We will use the following encoding yes=1 and no=0
- The instance will be for example: $\mathbf{x} = (0, 0, 1, 0, \ldots)$

Building a spam detector



- We will use the following encoding yes=1 and no=0
- The instance will be for example: $\mathbf{x} = (0, 0, 1, 0, \ldots)$
- Classifier: $\hat{y} = f(\mathbf{x}) = \begin{cases} \text{spam, if } x_1 + x_2 + \ldots + x_{100} \geq 5 \\ \text{ham, otherwise} \end{cases}$

Building a spam detector



- We will use the following encoding yes=1 and no=0
- The instance will be for example: $\mathbf{x} = (0, 0, 1, 0, \ldots)$
- Classifier: $\hat{y} = f(\mathbf{x}) = \begin{cases} \text{spam, if } x_1 + x_2 + \ldots + x_{100} \geq 5 \\ \text{ham, otherwise} \end{cases}$
- What if some features are more important than others?

Making some features more important

- Let's make
  - x_1 three times more important than it was
  - x_2 two times more important than it was
- Our new classifier:

$$\hat{y} = f(\mathbf{x}) = \begin{cases} \text{spam, if } w_1 x_1 + w_2 x_2 + \ldots + w_{100} x_{100} \geq 5 \\ \text{ham, otherwise,} \end{cases}$$

  where $w_1 = 3, w_2 = 2, w_3 = w_4 = \ldots = w_{100} = 1$
- This particular classifier belongs to the family of **linear classifiers**

Linear classifier

Definition

A binary classifier is called a **linear classifier** if it predicts one class whenever

$$\sum_{i=1}^{m} w_i x_i \geq t$$

and predicts the other class otherwise. Here $w_i \in \mathbb{R}$ for $i = 1, \ldots, m$

Apache SpamAssassin [1]

- Anti-spam platform giving system administrators a filter to classify email and block spam
- Initial release: Apr 2001
- Latest release: Jan 2020
- Still in active use
- Uses a linear classifier on top of constructed binary features

[1] `https://en.wikipedia.org/wiki/Apache_SpamAssassin`

How does SpamAssassin work?

- Takes an e-mail
- Runs several tests on it
- Calculates a score based on the weights of each test
- If the score is 5 or more then classifies as spam, otherwise as ham

How to choose good weight values?

- Take a big set of labelled e-mails (known as the **training set** in machine learning)
- Make sure that your linear classifier makes as few mistakes as possible on this set
  - That is, it predicts the correct label (spam/ham) for as many e-mails as possible
  - Ideally, correct on all e-mails
- We could try through a huge set of possible weight combinations and find the best combination

Example

- We run 2 tests (x_1, x_2)
- We have 4 training e-mails
  - 1 spam (label: Spam=1)
  - 3 ham (label: Spam=0)

E-mail	x_1	x_2	Spam?
1	1	1	1
2	0	0	0
3	1	0	0
4	0	1	0

Example

- We run 2 tests (x_1, x_2)
- We have 4 training e-mails
  - 1 spam (label: Spam=1)
  - 3 ham (label: Spam=0)

E-mail	x_1	x_2	Spam?
1	1	1	1
2	0	0	0
3	1	0	0
4	0	1	0

- Which weights would be good with threshold 5?

Which weights would be good?

- Let's denote weights by w_1, w_2

E-mail	x_1	x_2	Spam?	$w_1x_1 + w_2x_2$
1	1	1	1	$w_1 + w_2$
2	0	0	0	0
3	1	0	0	w_1
4	0	1	0	w_2

- To get correct labelling we need:
  - $w_1 + w_2 \geq 5$
  - $w_1, w_2 < 5$

Which weights would be good?

- Let's denote weights by w_1, w_2

E-mail	x_1	x_2	Spam?	$w_1x_1 + w_2x_2$
1	1	1	1	$w_1 + w_2$
2	0	0	0	0
3	1	0	0	w_1
4	0	1	0	w_2

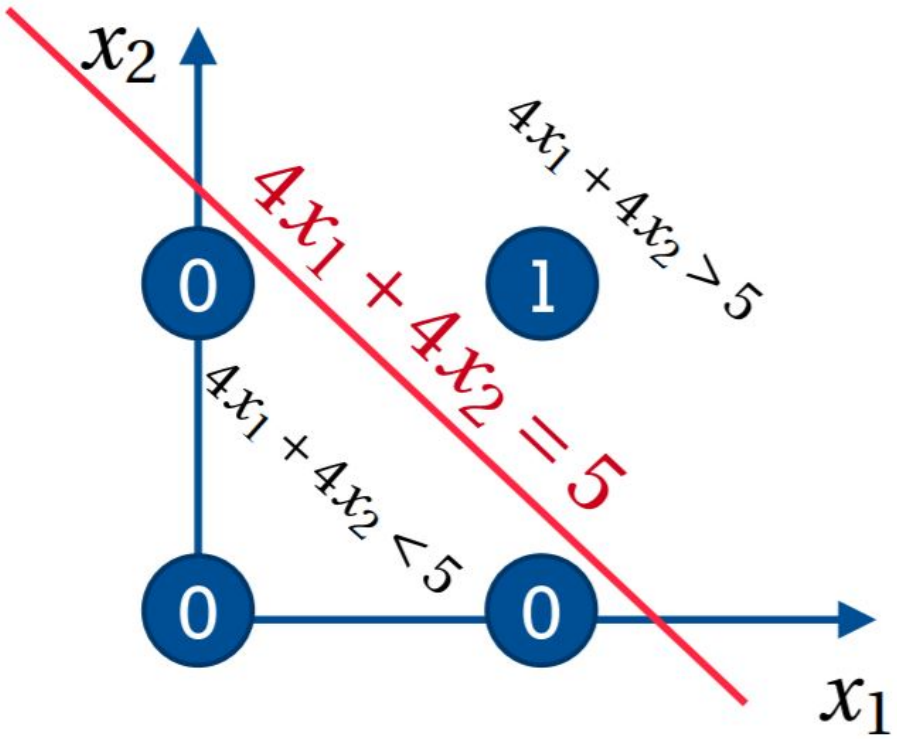
- To get correct labelling we need:
  - $w_1 + w_2 \geq 5$
  - $w_1, w_2 < 5$
- For example: $w_1 = w_2 = 4$

Linear classifier example

E-mail	x_1	x_2	Spam?
1	1	1	1
2	0	0	0
3	1	0	0
4	0	1	0

1 = spam

0 = ham



Training a spam detector

- Suppose now we have 100 features and more e-mails

Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?

Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?
- An expert studies a huge collection of emails, somehow learned a good set of weights. Now everyone can use it.

Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?
- An expert studies a huge collection of emails, somehow learned a good set of weights. Now everyone can use it.
- Why is this not a good solution?

Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?
- An expert studies a huge collection of emails, somehow learned a good set of weights. Now everyone can use it.
- Why is this not a good solution?
- One person's spam is another person's ham!

Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?
- An expert studies a huge collection of emails, somehow learned a good set of weights. Now everyone can use it.
- Why is this not a good solution?
- One person's spam is another person's ham!
- We need to personalize!

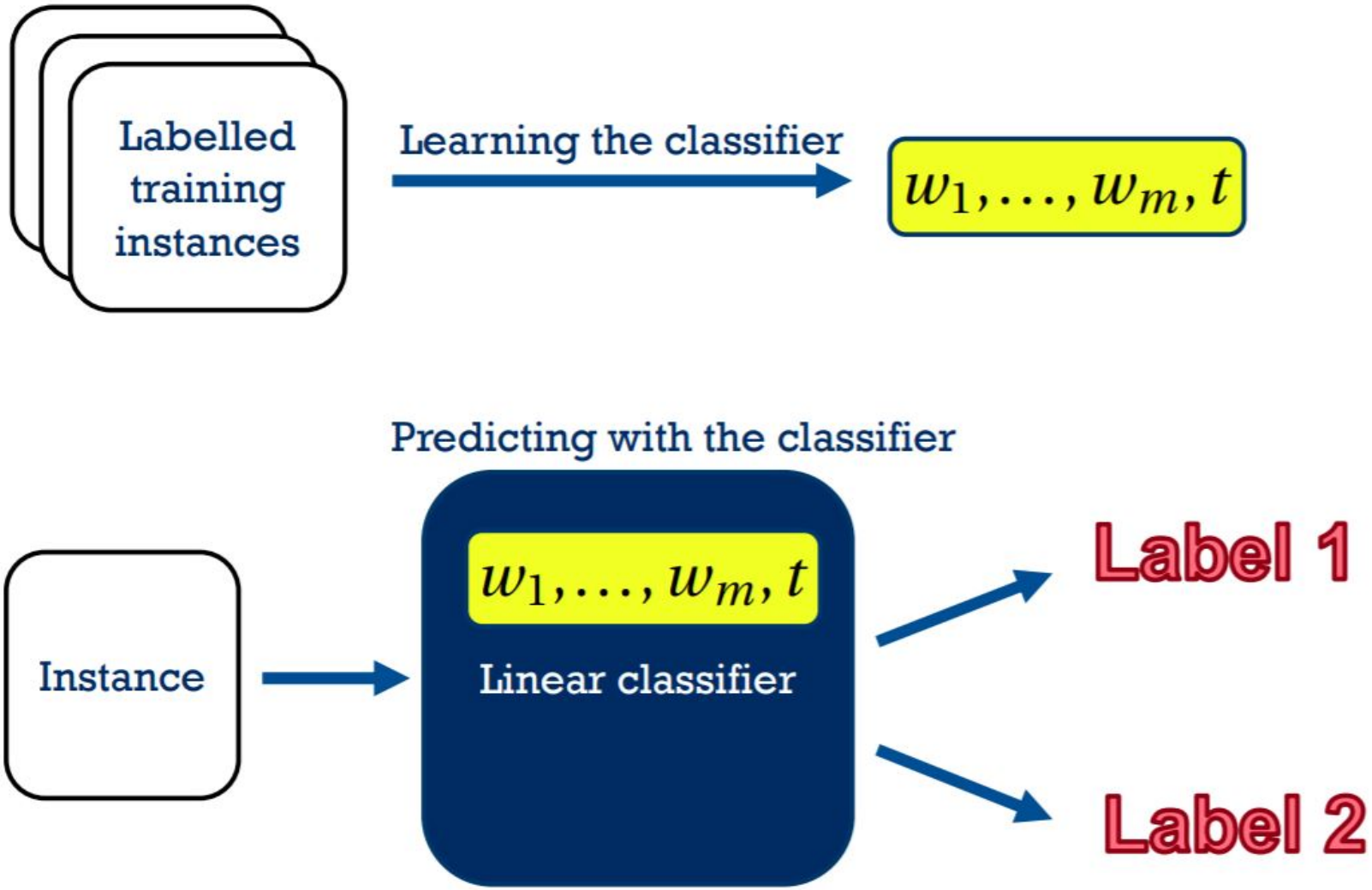
Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?
- An expert studies a huge collection of emails, somehow learned a good set of weights. Now everyone can use it.
- Why is this not a good solution?
- One person's spam is another person's ham!
- We need to personalize!
- Different training datasets for everyone

Training a spam detector

- Suppose now we have 100 features and more e-mails
- How can we build a good spam detector?
- An expert studies a huge collection of emails, somehow learned a good set of weights. Now everyone can use it.
- Why is this not a good solution?
- One person's spam is another person's ham!
- We need to personalize!
- Different training datasets for everyone
- Therefore, new set of weights is required for everyone! The process needs to be automated.

Learning a linear classifier and predicting with it

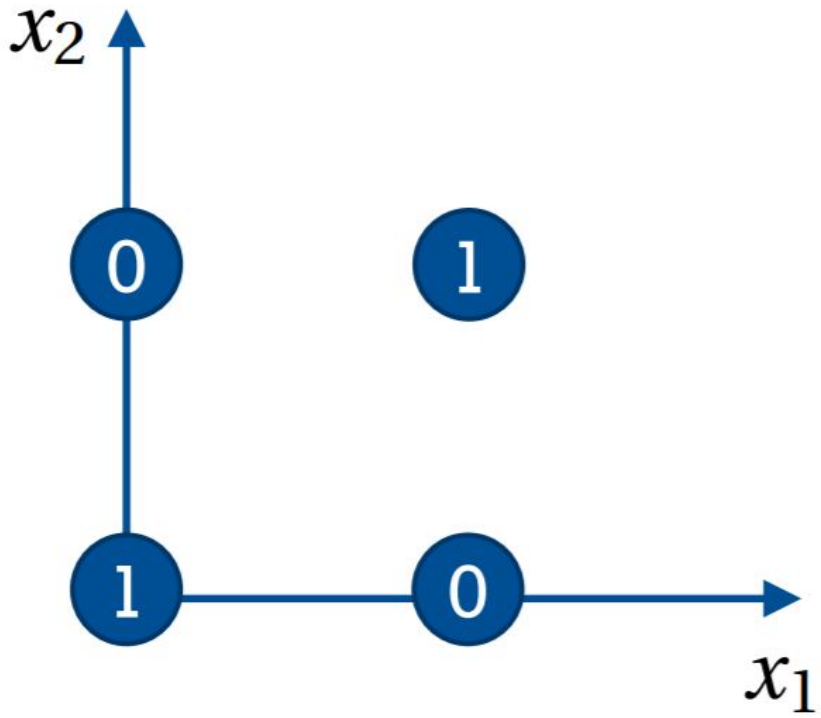


Learning classifier example

E-mail	x_1	x_2	Spam?
1	1	1	1
2	0	0	1
3	1	0	0
4	0	1	0

1 = spam

0 = ham



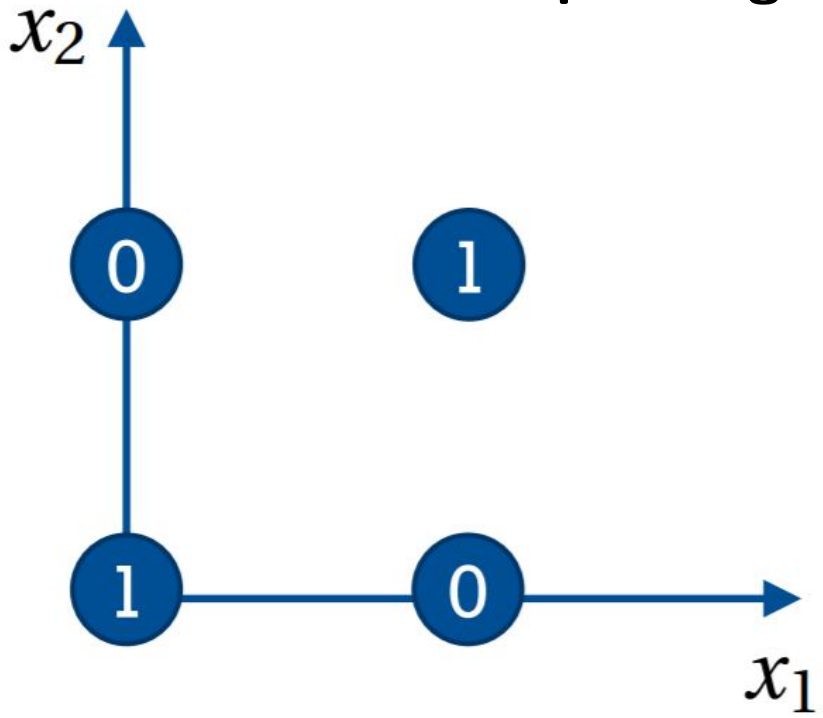
Learning classifier example

E-mail	x_1	x_2	Spam?
1	1	1	1
2	0	0	1
3	1	0	0
4	0	1	0

1 = spam

0 = ham

Seems like we cannot draw a separating line for these two classes.



No linear classifier fits training data?

- Another way of saying it is that *the classes are not linearly separable*
- Possible reasons
  - The data is noisy
  - Linear classifiers are not expressive enough for the given task
- Options
  - Choose a linear classifier which fits as many data as possible
  - Choose a more expressive model type

Finding the weights

- How to find a good set of weights automatically?

Finding the weights

- How to find a good set of weights automatically?
- Trial-and-error takes too long

Finding the weights

- How to find a good set of weights automatically?
- Trial-and-error takes too long
- Let's take one step back and think what is needed intuitively

Finding the weights

- How to find a good set of weights automatically?
- Trial-and-error takes too long
- Let's take one step back and think what is needed intuitively
- We want a classifier that predicts spam if e-mail has many features that the spam e-mails have (i.e. is similar!) and predicts ham if e-mail has many features that the ham e-mails have

Finding the weights

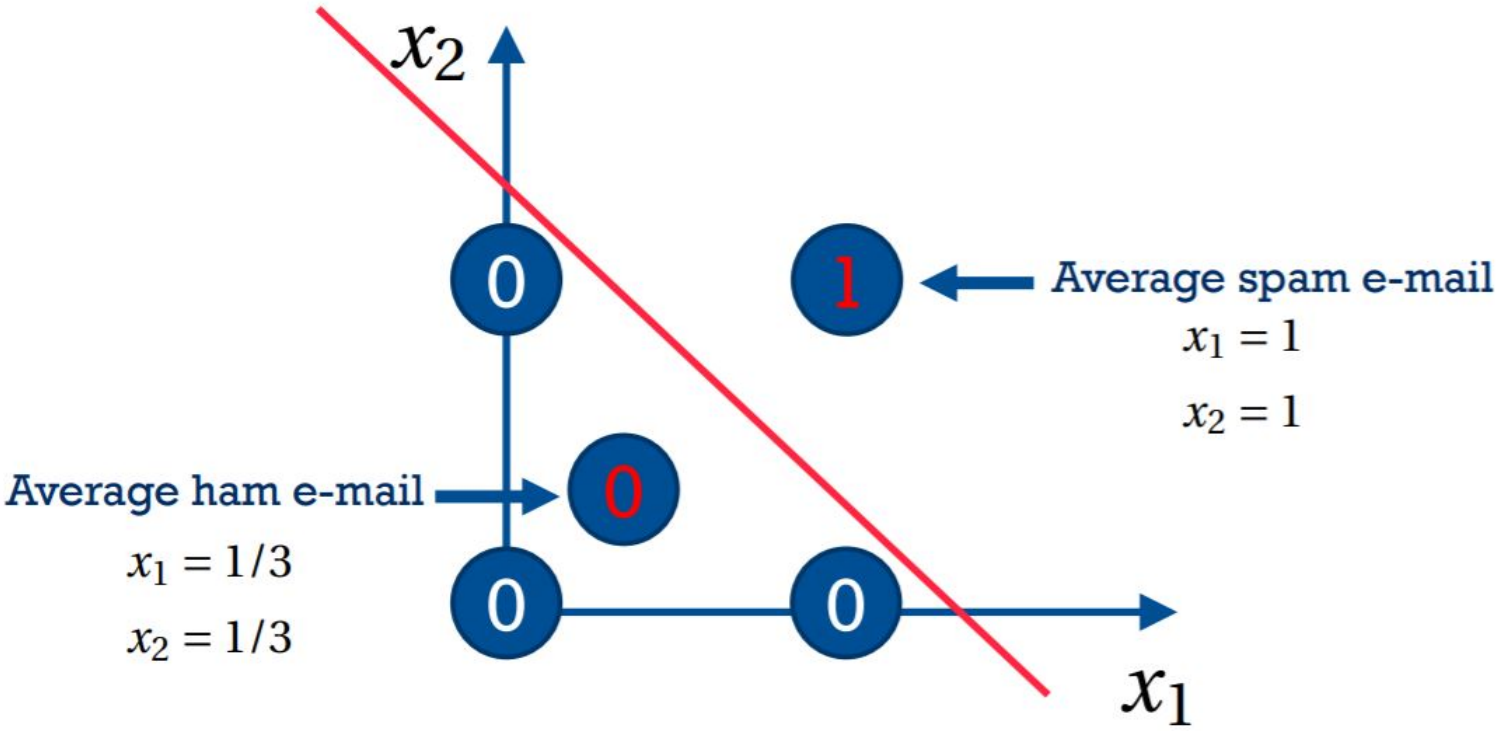
- How to find a good set of weights automatically?
- Trial-and-error takes too long
- Let's take one step back and think what is needed intuitively
- We want a classifier that predicts spam if e-mail has many features that the spam e-mails have (i.e. is similar!) and predicts ham if e-mail has many features that the ham e-mails have
- Let's compare against average spam and average ham e-mail!

Basic linear classifier

E-mail	x_1	x_2	Spam?
1	1	1	1
2	0	0	0
3	1	0	0
4	0	1	0

1 = spam

0 = ham



How to calculate the weights?

- Suppose $\mathbf{x} \in \mathbb{R}^m$, where m is number of features in our training dataset (Tr)

How to calculate the weights?

- Suppose $\mathbf{x} \in \mathbb{R}^m$, where m is number of features in our training dataset (Tr)
- Let's denote by $\mathbf{p}, \mathbf{n}$ - center of positives/negatives and $\mathbf{x}$ - data instance

$$\mathbf{p} = \frac{1}{|Tr^{\oplus}|} \sum_{\mathbf{x} \in Tr^{\oplus}} \mathbf{x}; \quad \mathbf{n} = \frac{1}{|Tr^{\ominus}|} \sum_{\mathbf{x} \in Tr^{\ominus}} \mathbf{x}$$

How to calculate the weights?

- Suppose $\mathbf{x} \in \mathbb{R}^m$, where m is number of features in our training dataset (Tr)
- Let's denote by $\mathbf{p}, \mathbf{n}$ - center of positives/negatives and $\mathbf{x}$ - data instance

$$\mathbf{p} = \frac{1}{|Tr^{\oplus}|} \sum_{\mathbf{x} \in Tr^{\oplus}} \mathbf{x}; \quad \mathbf{n} = \frac{1}{|Tr^{\ominus}|} \sum_{\mathbf{x} \in Tr^{\ominus}} \mathbf{x}$$

- We predict positive if $\mathbf{x}$ is closer to $\mathbf{p}$ than $\mathbf{n}$

How to calculate the weights?

- Suppose $\mathbf{x} \in \mathbb{R}^m$, where m is number of features in our training dataset (Tr)
- Let's denote by $\mathbf{p}, \mathbf{n}$ - center of positives/negatives and $\mathbf{x}$ - data instance

$$\mathbf{p} = \frac{1}{|Tr^{\oplus}|} \sum_{\mathbf{x} \in Tr^{\oplus}} \mathbf{x}; \quad \mathbf{n} = \frac{1}{|Tr^{\ominus}|} \sum_{\mathbf{x} \in Tr^{\ominus}} \mathbf{x}$$

- We predict positive if $\mathbf{x}$ is closer to $\mathbf{p}$ than $\mathbf{n}$
- As we have Euclidean space then distance from $\mathbf{x}$ to $\mathbf{p}$ is

$$\|\mathbf{x} - \mathbf{p}\| = \sqrt{(\mathbf{x} - \mathbf{p})^T (\mathbf{x} - \mathbf{p})}$$

How to calculate the weights?

- Suppose $\mathbf{x} \in \mathbb{R}^m$, where m is number of features in our training dataset (Tr)
- Let's denote by $\mathbf{p}, \mathbf{n}$ - center of positives/negatives and $\mathbf{x}$ - data instance

$$\mathbf{p} = \frac{1}{|Tr^{\oplus}|} \sum_{\mathbf{x} \in Tr^{\oplus}} \mathbf{x}; \quad \mathbf{n} = \frac{1}{|Tr^{\ominus}|} \sum_{\mathbf{x} \in Tr^{\ominus}} \mathbf{x}$$

- We predict positive if $\mathbf{x}$ is closer to $\mathbf{p}$ than $\mathbf{n}$
- As we have Euclidean space then distance from $\mathbf{x}$ to $\mathbf{p}$ is

$$\|\mathbf{x} - \mathbf{p}\| = \sqrt{(\mathbf{x} - \mathbf{p})^T (\mathbf{x} - \mathbf{p})}$$

- Thus, we will predict positive if $\mathbf{x}$ is closer to $\mathbf{p}$ than $\mathbf{n}$:

$$\|\mathbf{x} - \mathbf{n}\| \geq \|\mathbf{x} - \mathbf{p}\|$$

Derivation of weights

$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$

Derivation of weights

$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$

$$(\mathbf{x}-\mathbf{n})\cdot(\mathbf{x}-\mathbf{n}) \geq (\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p})$$

Derivation of weights

$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$

$$(\mathbf{x}-\mathbf{n})\cdot(\mathbf{x}-\mathbf{n}) \geq (\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p})$$

$$\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{n} + \mathbf{n}\cdot\mathbf{n} \geq \mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{p} + \mathbf{p}\cdot\mathbf{p}$$

Derivation of weights

$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$

$$(\mathbf{x}-\mathbf{n})\cdot(\mathbf{x}-\mathbf{n}) \geq (\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p})$$

$$\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{n} + \mathbf{n}\cdot\mathbf{n} \geq \mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{p} + \mathbf{p}\cdot\mathbf{p}$$

$$2\mathbf{x}\cdot\mathbf{p} - 2\mathbf{x}\cdot\mathbf{n} \geq \mathbf{p}\cdot\mathbf{p} - \mathbf{n}\cdot\mathbf{n}$$

Derivation of weights

$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$

$$(\mathbf{x}-\mathbf{n})\cdot(\mathbf{x}-\mathbf{n}) \geq (\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p})$$

$$\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{n} + \mathbf{n}\cdot\mathbf{n} \geq \mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{p} + \mathbf{p}\cdot\mathbf{p}$$

$$2\mathbf{x}\cdot\mathbf{p} - 2\mathbf{x}\cdot\mathbf{n} \geq \mathbf{p}\cdot\mathbf{p} - \mathbf{n}\cdot\mathbf{n}$$

$$(\mathbf{p}-\mathbf{n})\cdot\mathbf{x} \geq (\mathbf{p}-\mathbf{n})\cdot(\mathbf{p}+\mathbf{n})/2$$

Derivation of weights

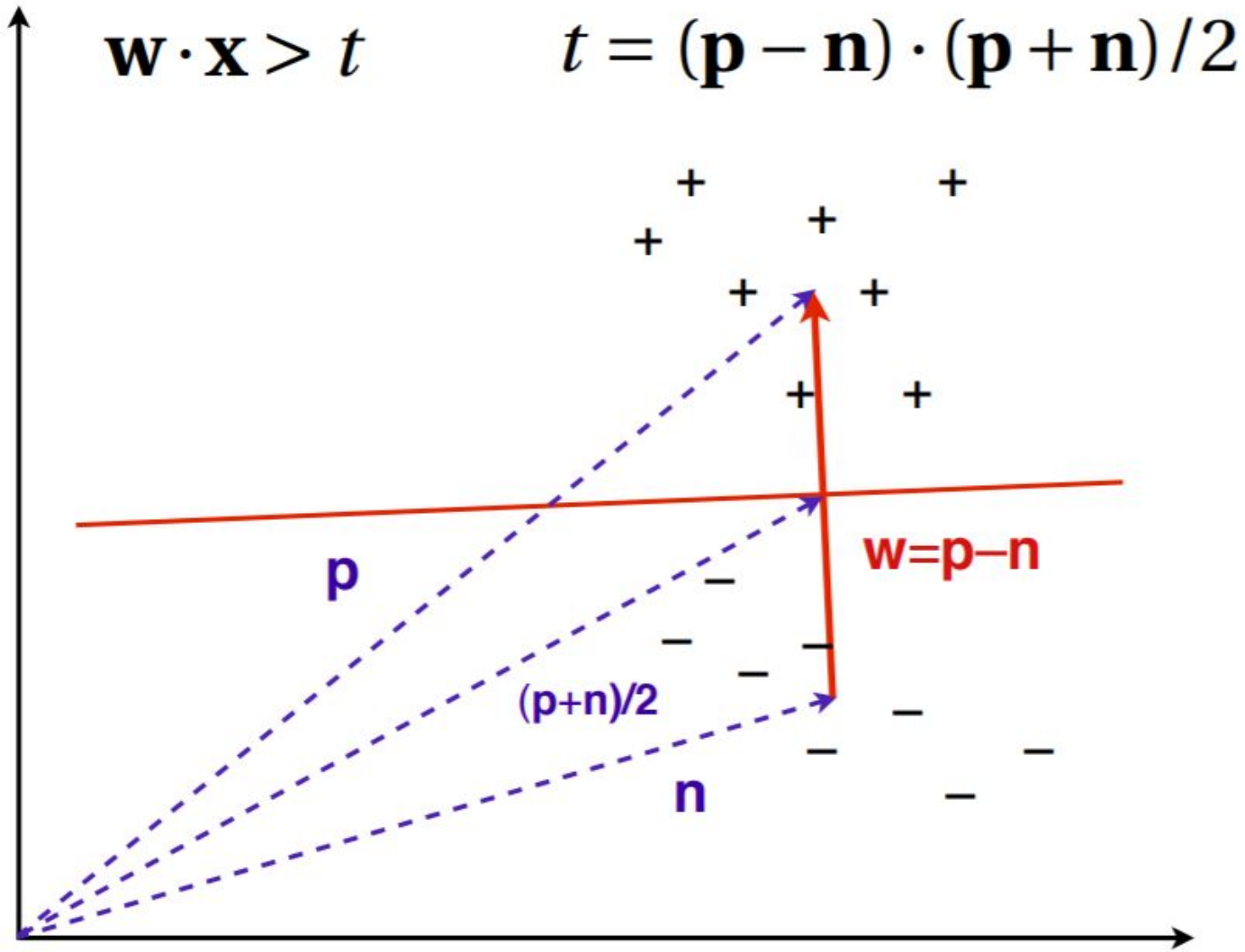
$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$
$$(\mathbf{x}-\mathbf{n})\cdot(\mathbf{x}-\mathbf{n}) \geq (\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p})$$
$$\mathbf{x}\cdot\mathbf{x}-2\mathbf{x}\cdot\mathbf{n}+\mathbf{n}\cdot\mathbf{n} \geq \mathbf{x}\cdot\mathbf{x}-2\mathbf{x}\cdot\mathbf{p}+\mathbf{p}\cdot\mathbf{p}$$
$$2\mathbf{x}\cdot\mathbf{p}-2\mathbf{x}\cdot\mathbf{n} \geq \mathbf{p}\cdot\mathbf{p}-\mathbf{n}\cdot\mathbf{n}$$
$$(\mathbf{p}-\mathbf{n})\cdot\mathbf{x} \geq (\mathbf{p}-\mathbf{n})\cdot(\mathbf{p}+\mathbf{n})/2$$
$$\mathbf{w}\cdot\mathbf{x} \geq t,$$

Derivation of weights

$$\sqrt{(\mathbf{x}-\mathbf{n})^T(\mathbf{x}-\mathbf{n})} \geq \sqrt{(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p})}$$
$$(\mathbf{x}-\mathbf{n})\cdot(\mathbf{x}-\mathbf{n}) \geq (\mathbf{x}-\mathbf{p})\cdot(\mathbf{x}-\mathbf{p})$$
$$\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{n} + \mathbf{n}\cdot\mathbf{n} \geq \mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{p} + \mathbf{p}\cdot\mathbf{p}$$
$$2\mathbf{x}\cdot\mathbf{p} - 2\mathbf{x}\cdot\mathbf{n} \geq \mathbf{p}\cdot\mathbf{p} - \mathbf{n}\cdot\mathbf{n}$$
$$(\mathbf{p}-\mathbf{n})\cdot\mathbf{x} \geq (\mathbf{p}-\mathbf{n})\cdot(\mathbf{p}+\mathbf{n})/2$$
$$\mathbf{w}\cdot\mathbf{x} \geq t,$$

where $\mathbf{w} = \mathbf{p}-\mathbf{n}$, $t = (\mathbf{p}-\mathbf{n})\cdot(\mathbf{p}+\mathbf{n})/2 = (\|\mathbf{p}\|^2 - \|\mathbf{n}\|^2)/2$

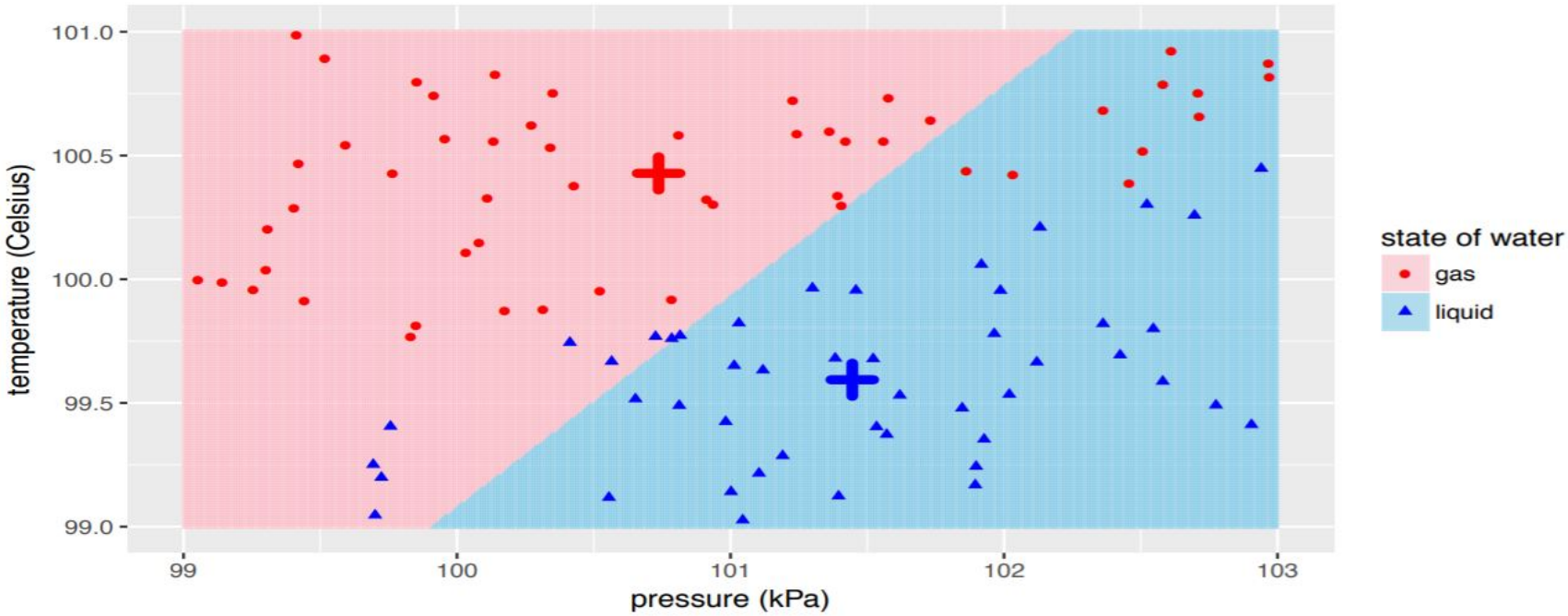
Basic linear classifier



Summary: Basic linear classifier algorithm

- Binary classification method, not widespread
- Learning the model (also called **fitting**):
  - Calculate the centre of mass of positive (training) instances and of negative instances
- Applying the model (also called **predicting**):
  - for a test instance $\mathbf{x}$ predict positive if $\mathbf{w}^T\mathbf{x} > t$
  - otherwise predict negative
  - here $\mathbf{w} = \mathbf{p} - \mathbf{n}$, $t = (\mathbf{p} - \mathbf{n}) \cdot (\mathbf{p} + \mathbf{n})/2$

Problem with basic linear classifier



Main problem

Even though a perfect linear classifier seems to exist, it is not found by the basic linear classifier.

In the future lectures, we will learn about linear classifiers that don't have this type of failure in cases when the label classes are linearly separable.

What have we learned today?

- ✓ Ingredients of Machine Learning
- ✓ Classification Basics
- ✓ Basic Linear Classifier