Introduction to Machine Learning

with **Python**

Faculty of Mathematics and Computer Science, University of Bucharest

and

Sparktech Software

Academic Year 2018/2019, 1st Semester

Team

Adrian Cosma
adrian.cosma@sparktech.ro

Andrei Manea
andrei.manea@sparktech.ro

Antonio Bărbălău
antonio.barbalau@sparktech.ro

ml.lecture@sparktech.ro

Andrei Iușan
andrei.iusan@sparktech.ro

Ioana Toma
ioana.toma@sparktech.ro

Andrei Sbârcea
andrei.sbarcea@sparktech.ro

Sparktech Software

www.sparktech.ro

Objectives of this course

- Provide you with an **intuitive understanding** of fundamental Machine Learning notions and algorithms.
    - Sometimes the *idea* behind an algorithm is more important than the algorithm itself.
- At the same time, provide you with a clear **mathematical foundation** for them.
    - Understanding the *inner workings* of an algorithm allows you to truly take advantage of what it can do.
- Allow you to **experiment hands-on** with the notions discussed in the lecture.
    - Using *Python* with *NumPy*, *matplotlib*, *scikit-learn* and *TensorFlow*.
    - Make *connections* between theoretical and practical aspects.

Administrative

Lecture

- Wednesday, 16:00 – 18:00
- 3rd floor, Țițeica Amphitheatre
- Slides and materials: https://www.dropbox.com/sh/udukklpadhd53oq/AAANVc7WL8GD_Xd-dWBs6aYSa?dl=0

Evaluation

Lab attendance	1 point
3 homework assignments	6 points
Written exam	3 points

Labs

- Wednesday, 18:00 – 20:00

Room \ Time	18:00 – 19:00	19:00 – 20:00
L-321	Group 1	Group 2
L-309	Opt. 3rd year	Group 3

What is Machine Learning?

Machine Learning is everywhere

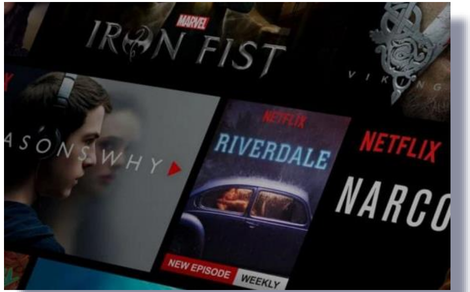
Game Playing

Self-driving Cars

Machine Translation

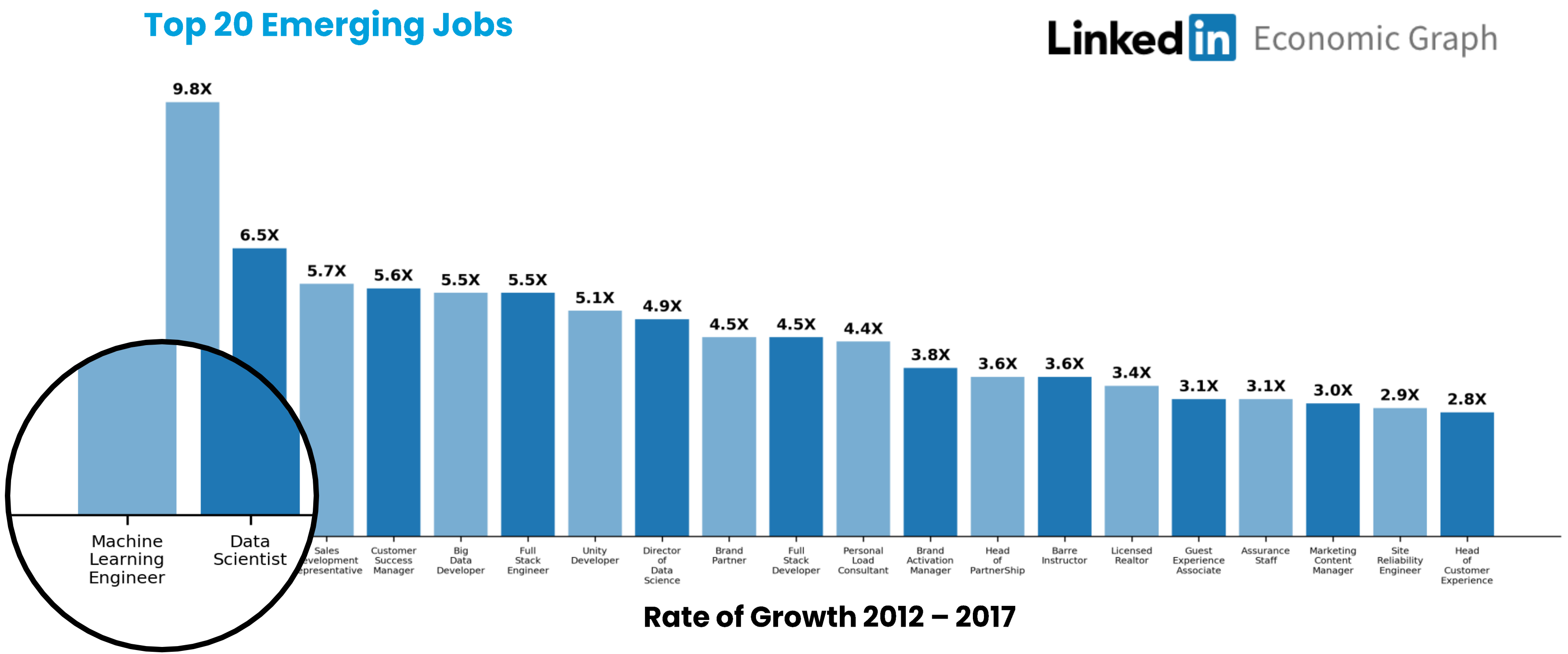

Intelligent Assistants


Style Transfer



Recommendation Engines

Machine Learning is everywhere



What is Machine Learning?

- Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

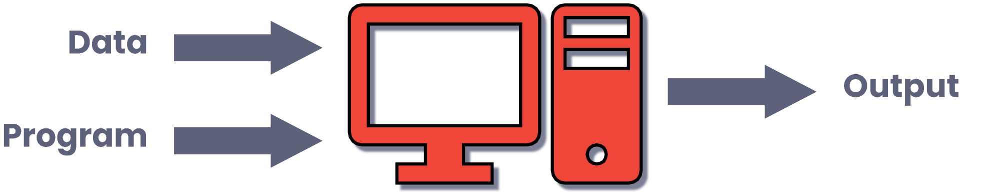
Arthur Samuel, 1959

- A computer program is said to learn from experience *E* with respect to some task *T* and some performance measure *P*, if its performance on *T*, as measured by *P*, improves with experience *E*.

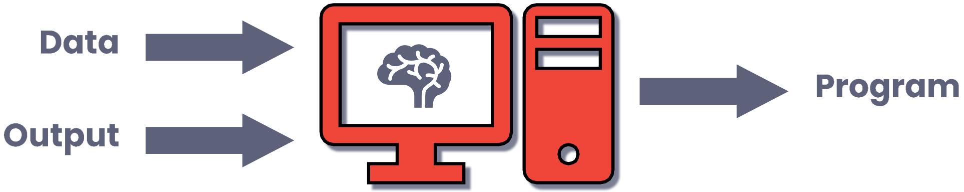
Tom Mitchell, 1997

What is Machine Learning?

Traditional Programming

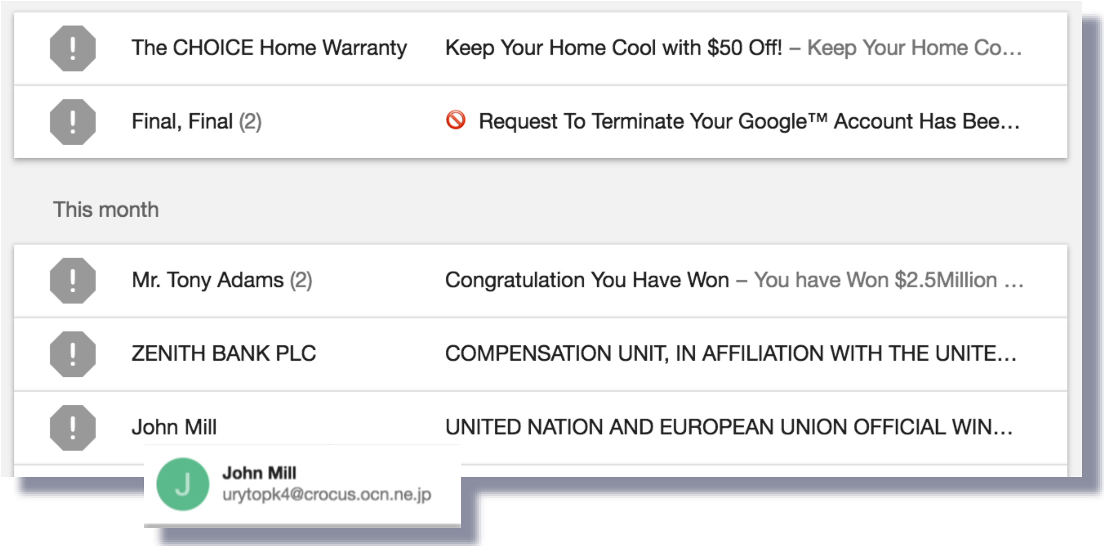


Machine Learning



When to use Machine Learning?

- How would you write a spam filter without ML?



When to use Machine Learning?

- How would you write a spam filter without ML?
  - Tons of “if this and this, but not this, then that” rules.
  - Very hard to maintain.

When to use Machine Learning?

- How would you write a program which detects cars in an image without ML?

When to use Machine Learning?

- How would you write a program which detects cars in an image without ML?
    - ¯_(ツ)_/¯

When to use Machine Learning?

- Problems for which traditional solutions require *lots of hand-tuning* or long lists or rules, which are hard to maintain:
    - E.g. Spam Detection, Machine Translation

- "*Unprogrammable*" tasks: Complex problems for which using a traditional programming approach is virtually impossible:
    - E.g. Object Detection, Speech Recognition

- Revealing insights and *unsuspected correlations* from large amounts of data.

Machine Learning Terminology

Terminology

- We are going to use a **dataset** with information about two types of fruit.

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

Terminology

- A **label** (or **target**) is what we are trying to predict.

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

Terminology

- A **label** (or **target**) is what we are trying to predict.
  - If it is discrete, it is also called a **class** and the process is called *classification*.
  - If it is continuous, the process is called *regression*.

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

Terminology

- A **feature** (or **attribute**) is “an individual measurable property of a phenomenon being observed”.

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

Terminology

- A **feature** (or **attribute**) is "an individual measurable property of a phenomenon being observed".
- All features of a data point form a **feature vector.**

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

Terminology

- A **feature** (or **attribute**) is “an individual measurable property of a phenomenon being observed”.
- All features of a data point form a **feature vector.**
- Most ML algorithms work with numerical values, so there are ways of converting categorical attributes to numbers.

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

Terminology

- An **example** (or **sample**) is a particular instance of data (a data point).
- It may or may not include a *label* (labeled vs. unlabeled data).

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

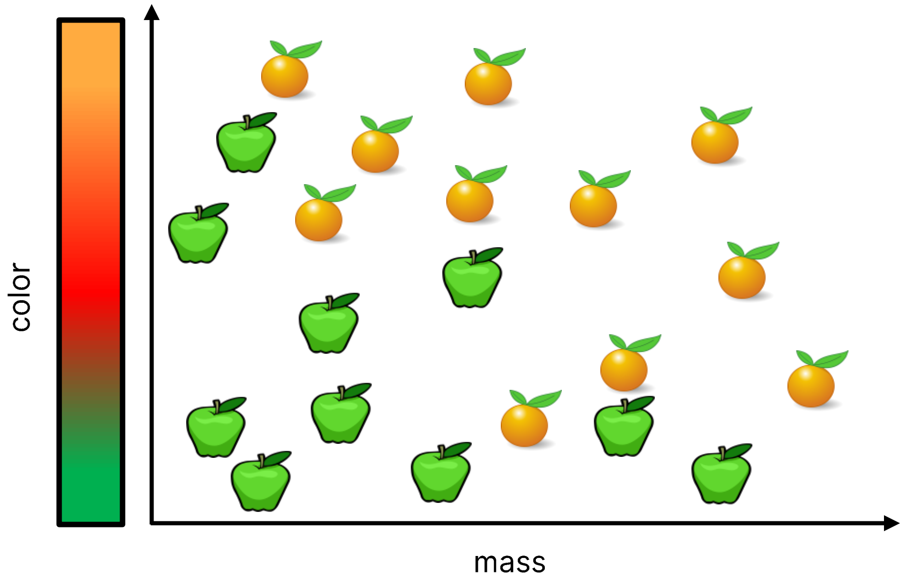
Terminology

- An **example** (or **sample**) is a particular instance of data (a data point).
- It may or may not include a *label* (labeled vs. unlabeled data).
- Either comes directly as a feature vector, or the feature vector is computed by selecting and transforming certain characteristics through **feature engineering**.

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
…				
117	Red	Rough	3.4	**Orange**

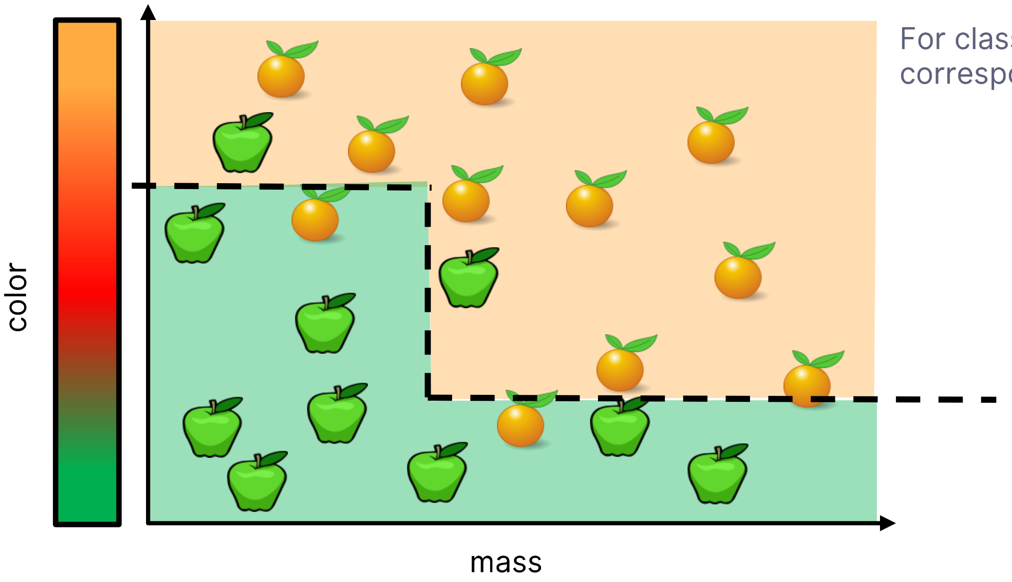
Terminology

- A **model** (or **hypothesis**) is an *established* relationship between features and labels.



Terminology

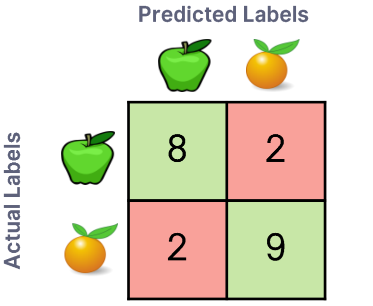
- A **model** (or **hypothesis**) is an *established* relationship between features and labels.
- Establishing the relationship based on a set of data points is called **training** (or **fitting**) the model.



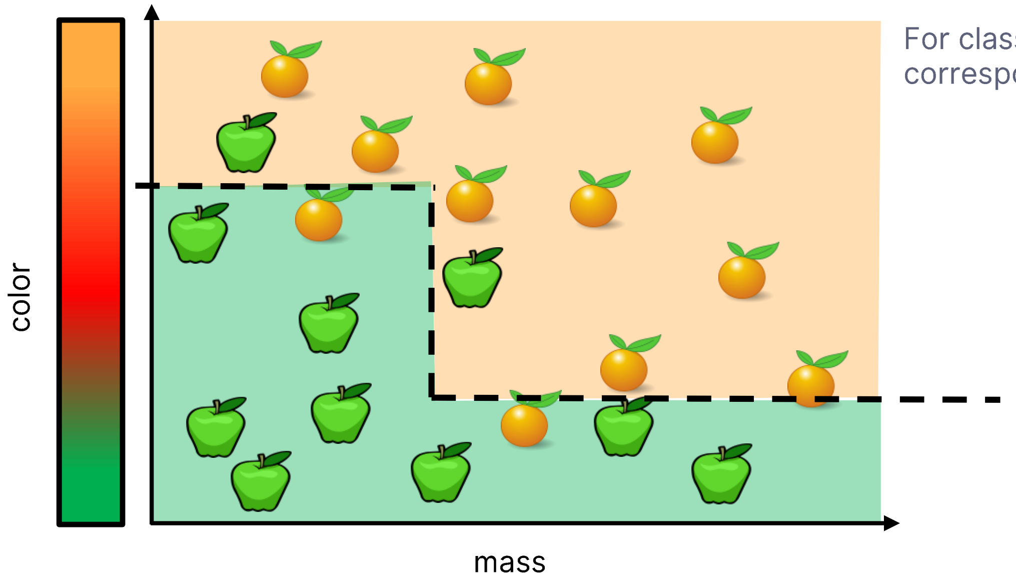
For classification, each model has a corresponding **decision boundary.**

Terminology

- A **model** (or **hypothesis**) is an *established* relationship between features and labels.
- Establishing the relationship based on a set of data points is called **training** (or **fitting**) the model.



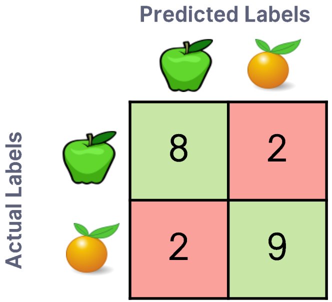
Confusion Matrix



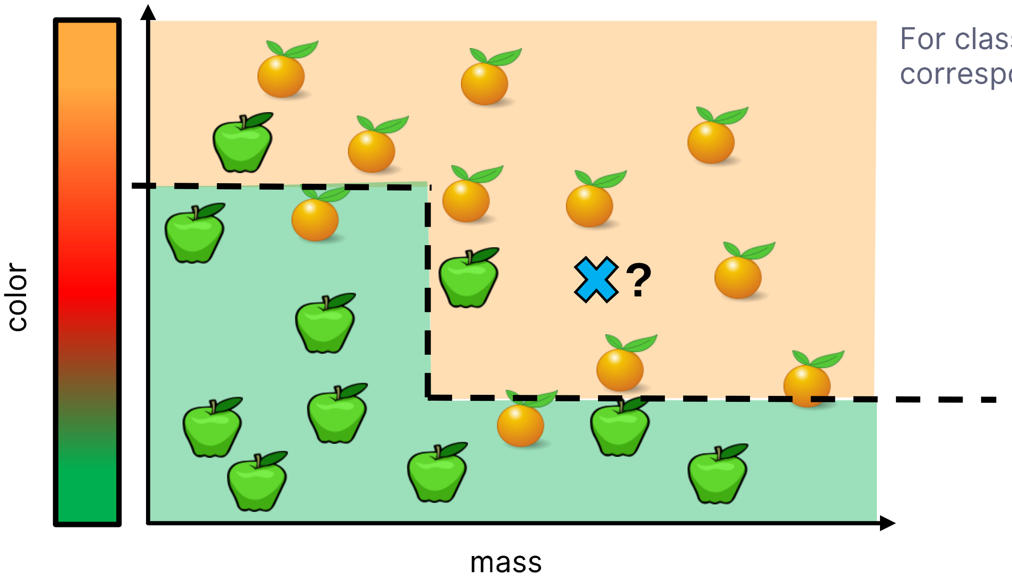
For classification, each model has a corresponding **decision boundary.**

Terminology

- A **model** (or **hypothesis**) is an *established* relationship between features and labels.
- Establishing the relationship based on a set of data points is called **training** (or **fitting**) the model.
- During **inference**, the trained model is used to make predictions on previously unseen points.



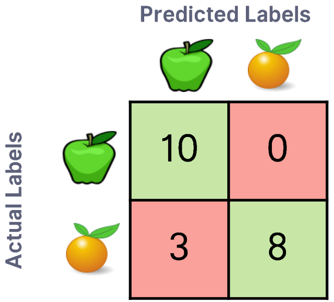
Confusion Matrix



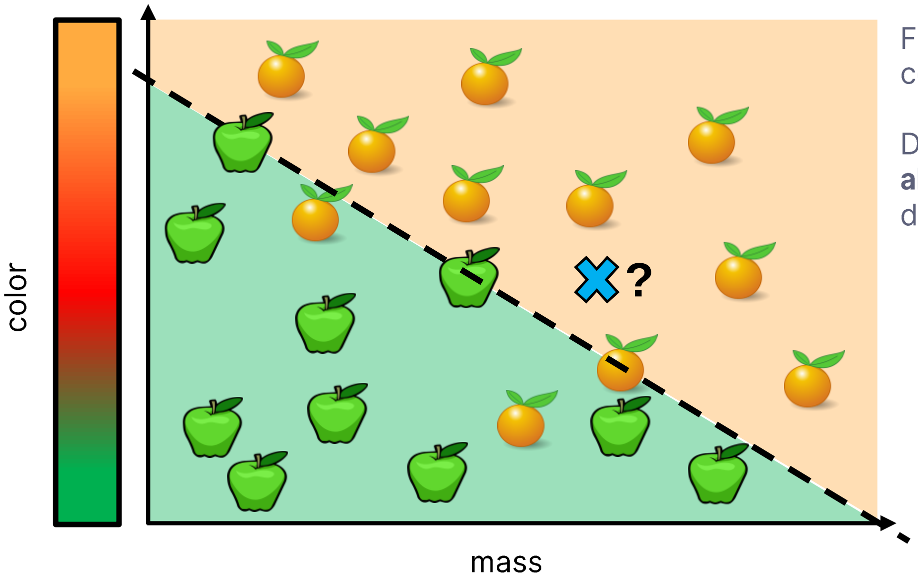
For classification, each model has a corresponding **decision boundary.**

Terminology

- A **model** (or **hypothesis**) is an *established* relationship between features and labels.
- Establishing the relationship based on a set of data points is called **training** (or **fitting**) the model.
- During **inference**, the trained model is used to make predictions on previously unseen points.



Confusion Matrix

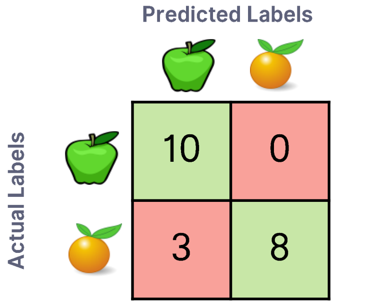


For classification, each model has a corresponding **decision boundary.**

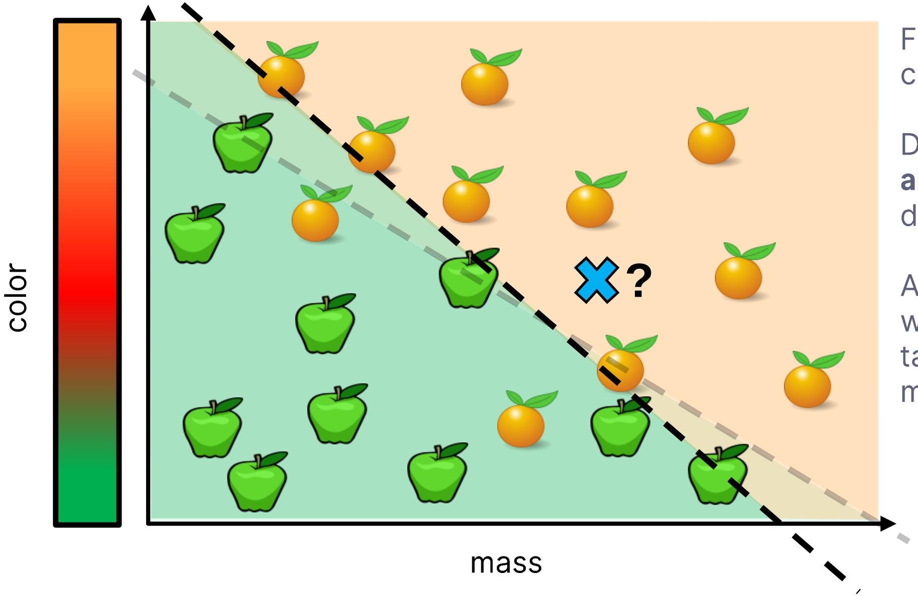
Different **algorithms** have different **allowed hypotheses** and therefore different decision boundary shapes.

Terminology

- A **model** (or **hypothesis**) is an *established* relationship between features and labels.
- Establishing the relationship based on a set of data points is called **training** (or **fitting**) the model.
- During **inference**, the trained model is used to make predictions on previously unseen points.



Confusion Matrix



For classification, each model has a corresponding **decision boundary.**

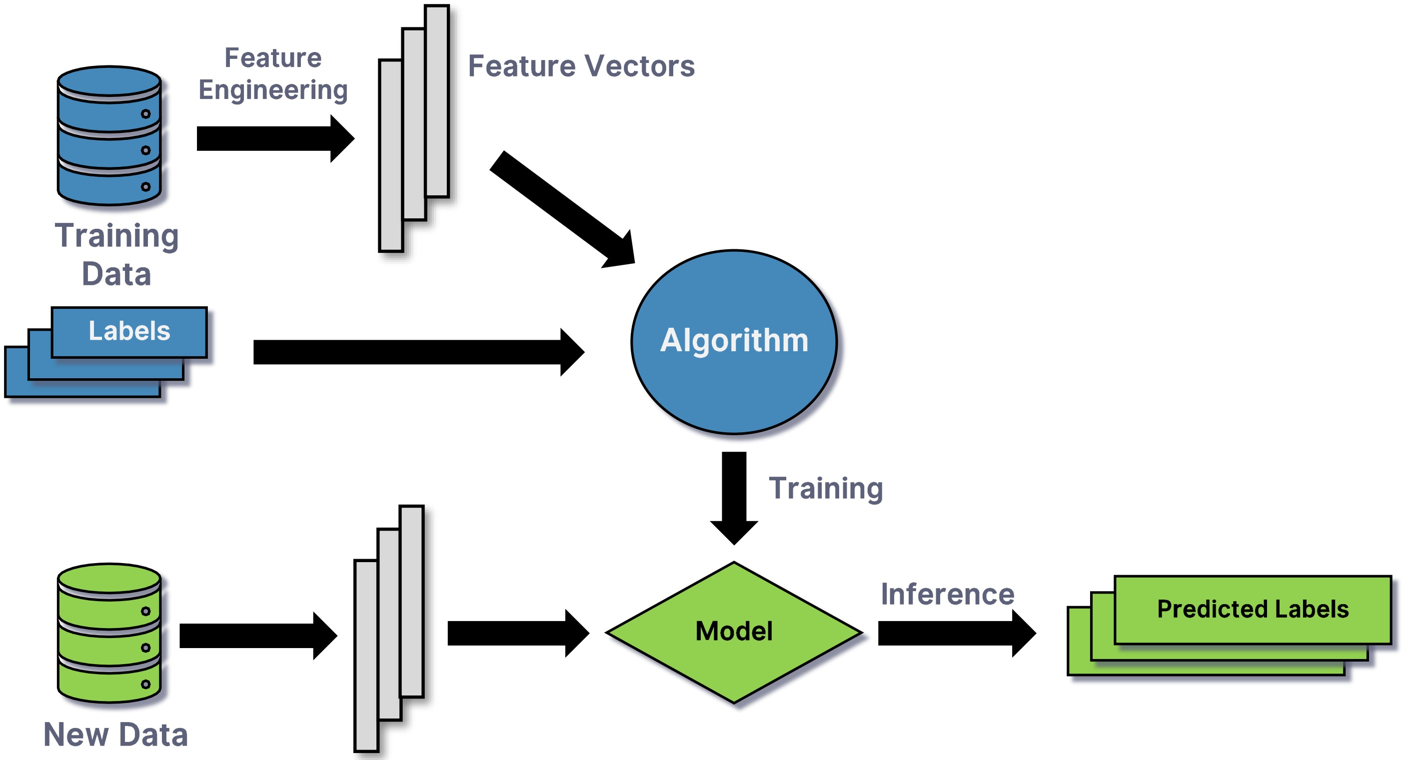
Different **algorithms** have different **allowed hypotheses** and therefore different decision boundary shapes.

Algorithms have **hyperparameters** which control how the learning takes place, affecting the resulting model.

Terminology Recap

- **Label** (or **target**) → What we are trying to *predict*.
- **Feature** (or **attribute**) → Measurable characteristic of a **sample** (data point).
  - All features form a **feature vector**.
- **Model** (or **hypothesis**) → Relationship between features and labels.
- **Training** (or **fitting**) → Establishing the relationship based on a set of data points.
- **Inference** → Making predictions on previously unseen points.
- **Algorithm** → Defines a concrete way of doing training.
  - Has constrains on the set of **allowed hypotheses**, some by design, some by the use of **hyperparameters**

Typical Machine Learning flow



Types of Learning

Supervised Learning

- There is a **label** which we are trying to predict
    - To do this we need labeled samples.

Supervised Learning

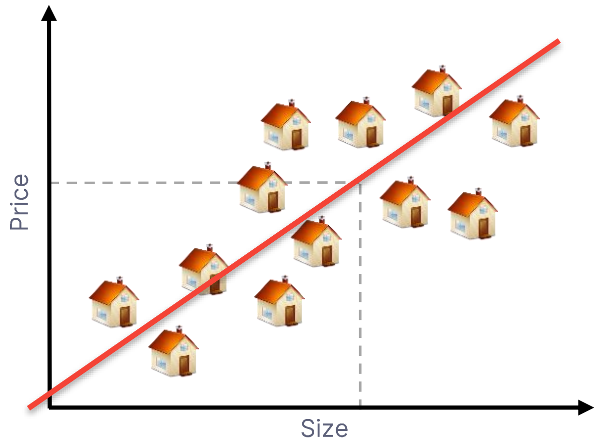
- There is a **label** which we are trying to predict
  - To do this we need labeled samples.

Regression – Label is continuous
e.g. Predicting house price given size

Supervised Learning

- There is a **label** which we are trying to predict
  - To do this we need labeled samples.

Regression – Label is continuous
e.g. Predicting house price given size

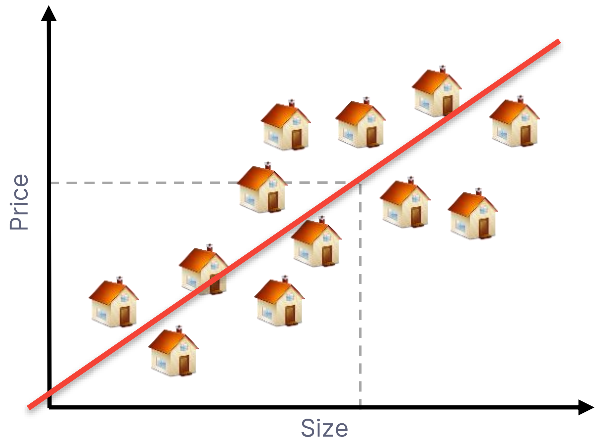


- Linear Regression
- KNN Regression
- Regression Trees

Supervised Learning

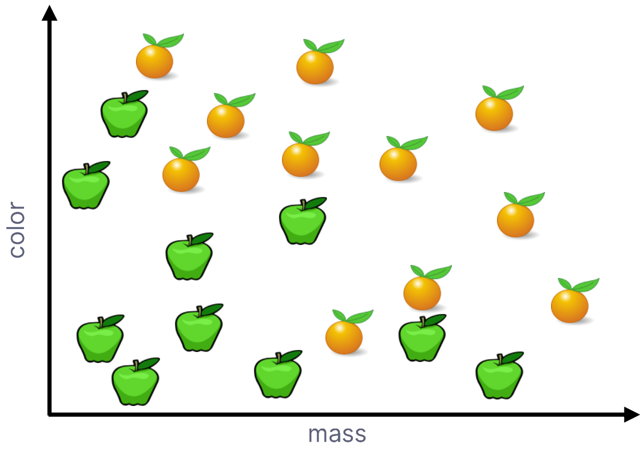
- There is a **label** which we are trying to predict
  - To do this we need labeled samples.

Regression – Label is continuous
e.g. Predicting house price given size



- Linear Regression
- KNN Regression
- Regression Trees

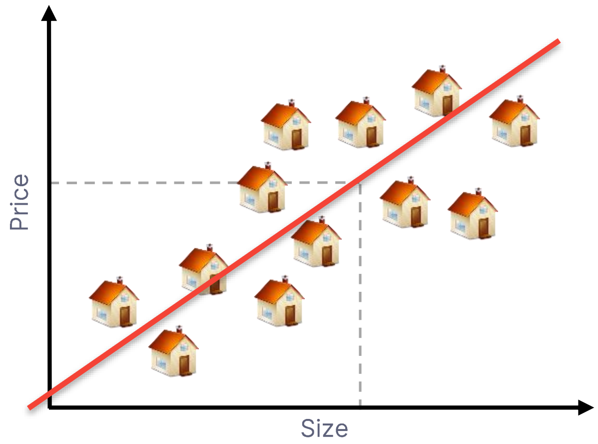
Classification – Label is discrete
e.g. Predicting fruit type given weight and color



Supervised Learning

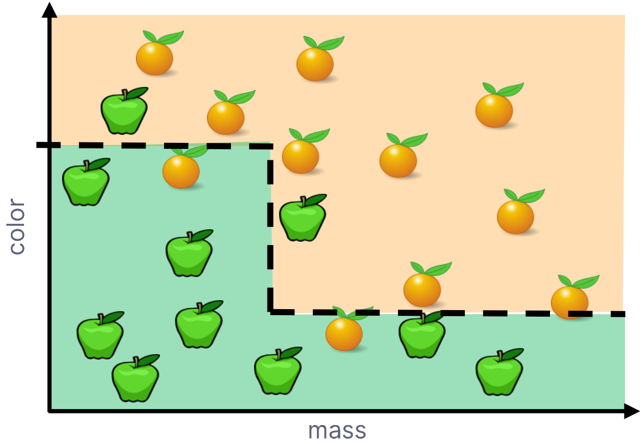
- There is a **label** which we are trying to predict
  - To do this we need labeled samples.

Regression – Label is continuous
e.g. Predicting house price given size



- Linear Regression
- KNN Regression
- Regression Trees

Classification – Label is discrete
e.g. Predicting fruit type given weight and color



- Logistic Regression
- KNN
- Decision Trees
- SVMs

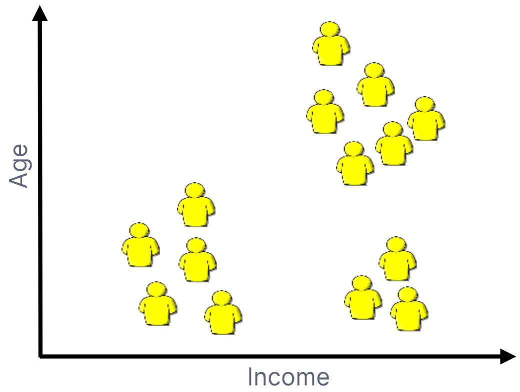
Unsupervised Learning

- There is no expected label, we are trying to **discover structure** in the data

Unsupervised Learning

- There is no expected label, we are trying to **discover structure** in the data

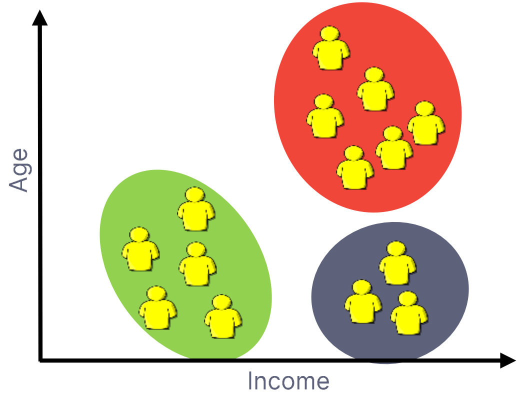
Clustering
Grouping *similar* items



Unsupervised Learning

- There is no expected label, we are trying to **discover structure** in the data

Clustering
Grouping *similar* items

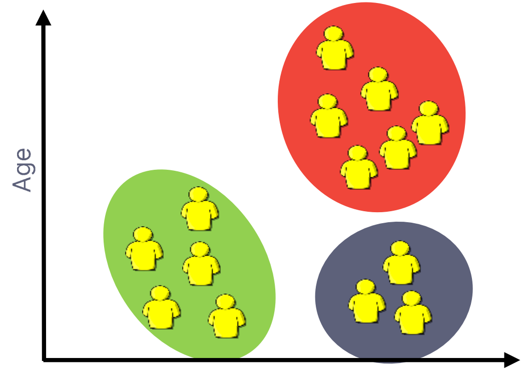


- K-means
- DBSCAN
- Hierarchical Clustering

Unsupervised Learning

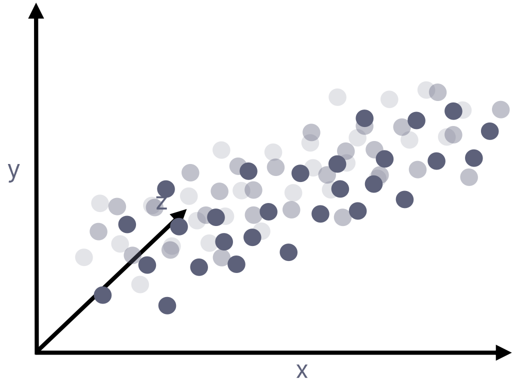
- There is no expected label, we are trying to **discover structure** in the data

Clustering
Grouping *similar* items



- K-means
- DBSCAN
- Hierarchical Clustering

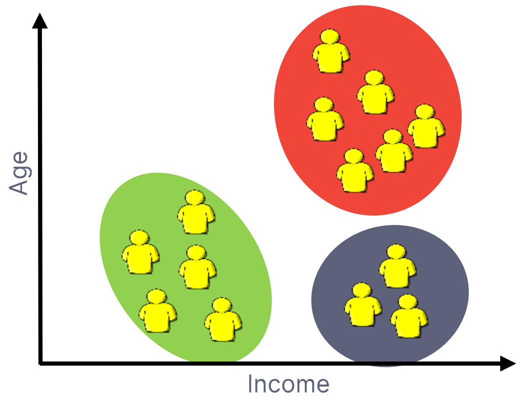
Dimensionality Reduction
Reducing the number of features



Unsupervised Learning

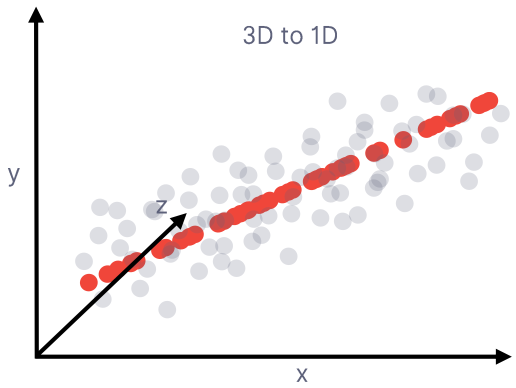
- There is no expected label, we are trying to **discover structure** in the data

Clustering
Grouping *similar* items



- K-means
- DBSCAN
- Hierarchical Clustering

Dimensionality Reduction
Reducing the number of features

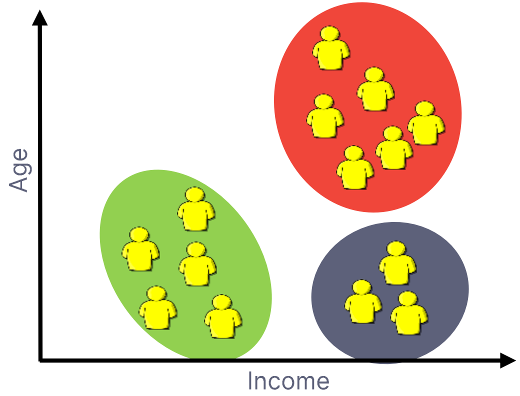


- Principal Component Analysis (PCA)
- T-SNE
- Self-organizing Maps (SOMs)

Unsupervised Learning

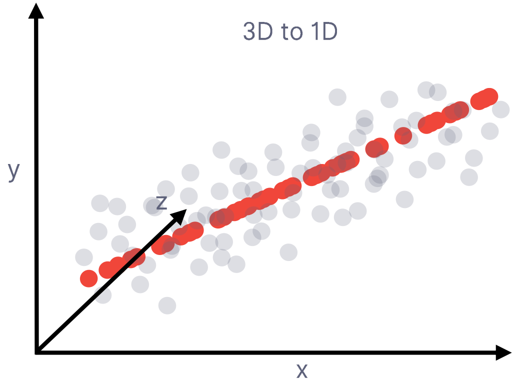
- There is no expected label, we are trying to **discover structure** in the data

Clustering
Grouping *similar* items



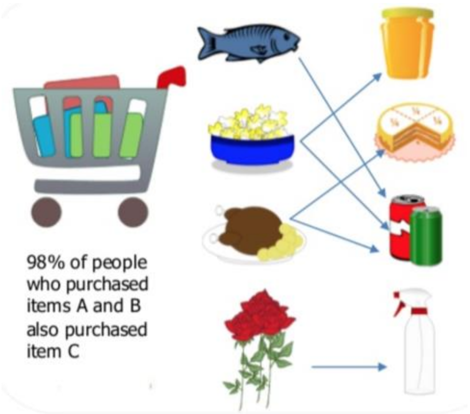
- K-means
- DBSCAN
- Hierarchical Clustering

Dimensionality Reduction
Reducing the number of features



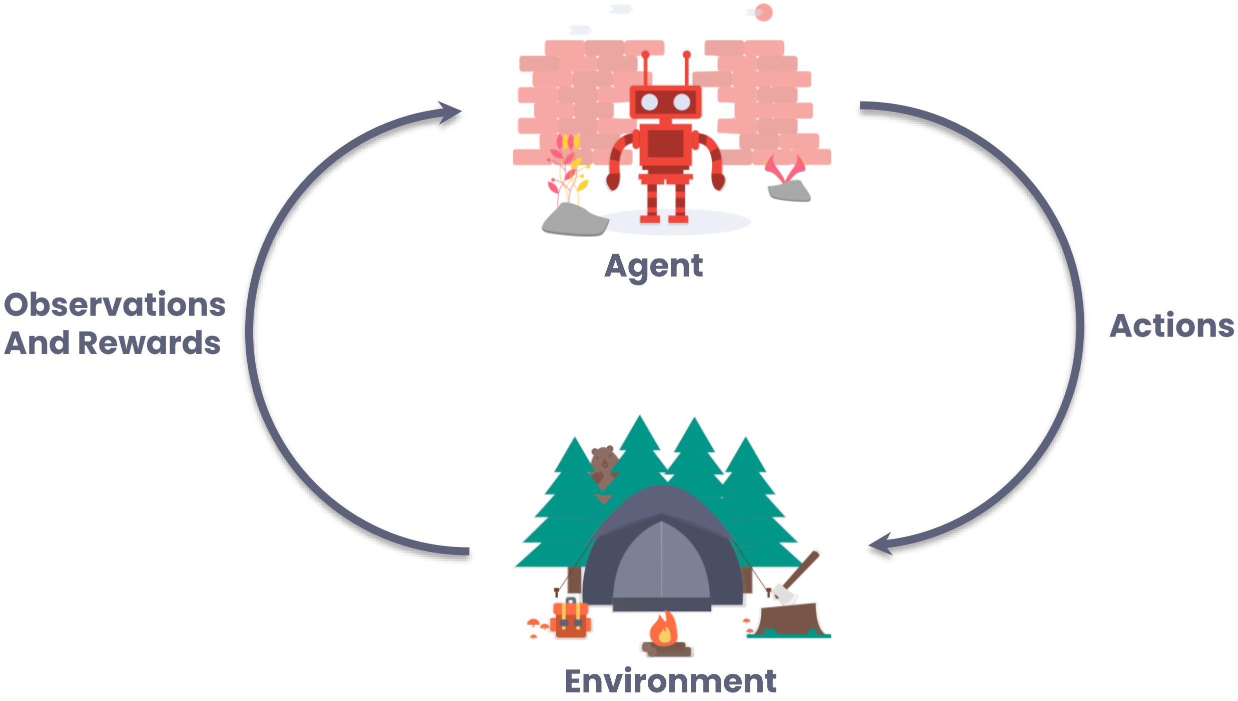
- Principal Component Analysis (PCA)
- T-SNE
- Self-organizing Maps (SOMs)

Association Rule Mining
Finding frequent patterns
and correlations



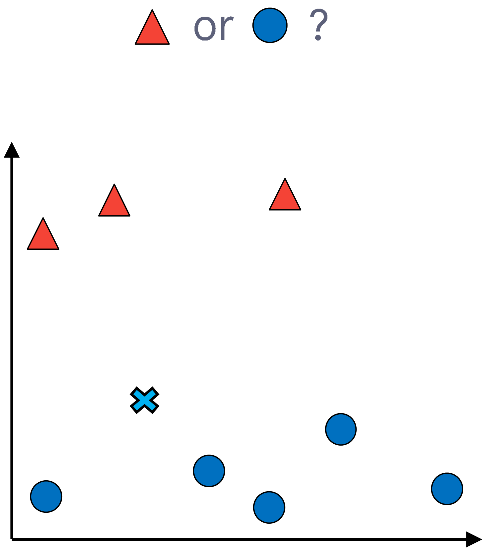
Reinforcement Learning

- There is no label, only **rewards** (or penalties) for taking **actions**



Semi-supervised Learning

- Only a few labeled examples.

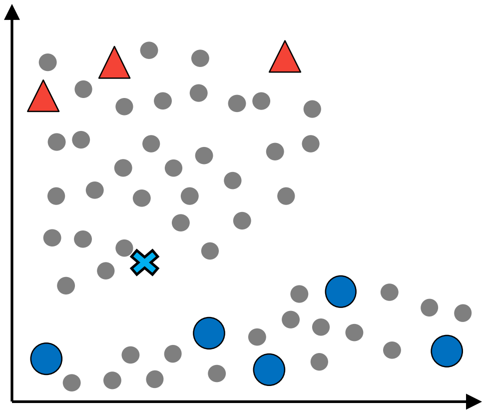


Semi-supervised Learning

- Only a few labeled examples.
- Lots of **unlabeled** examples.
    - Very common situation in practice

or ?

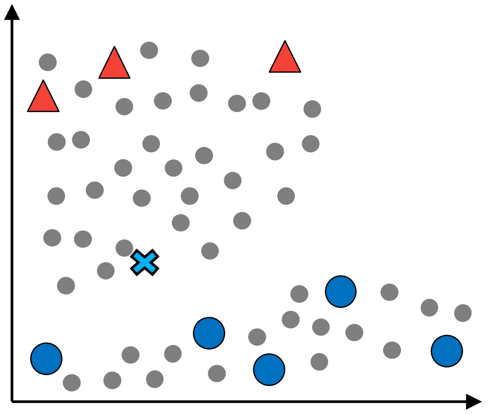
What about now?



Semi-supervised Learning

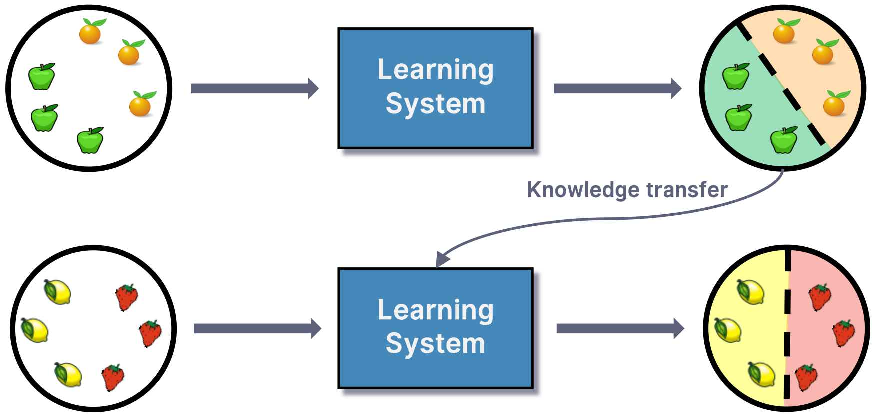
- Only a few labeled examples.
- Lots of **unlabeled** examples.
    - Very common situation in practice
- The unlabeled data can help improve supervised algorithms.
- Semi-supervised techniques:
    - Label Propagation
    - Co-training
    - GANs
    - Word2Vec

▲ or ● ?
What about now?



Transfer Learning

- Use the knowledge gained from solving one problem to solve another problem.
  - Usually used when there was much more data (or training time) available for the original problem.
- Very common in Deep Learning.
  - e.g. Using a model which was pre-trained on a large dataset



Types of Learning Recap

- **Supervised Learning** → There is a label
    - Label is continuous → **Regression**
    - Label is discrete → **Classification**
- **Unsupervised Learning** → Discovering structure in the data.
    - Grouping similar items → **Clustering**
    - Reducing number of features → **Dimensionality Reduction**
    - Frequent patterns → **Association Rule Mining**
- **Reinforcement Learning** → There is no label, only rewards (or penalties) for taking actions
- *Semi-supervised Learning* → Some labeled, lots of unlabeled data
- *Transfer Learning* → Use model trained for one task to speed up learning for another task

Performance Evaluation Terminology

Evaluating a model

- A model is only powerful if it performs well on data which it has not seen before.

Evaluating a model

- A model is only powerful if it performs well on data which it has not seen before.
  - This means that it doesn't just remember the training data, but it has the capacity to **generalize**
  - **Overfitting** means the model performs well on training data, but it fails to generalize
  - **Underfitting** means the model performs badly on both training and unseen data

Evaluating a model

- A model is only powerful if it performs well on data which it has not seen before.
  - This means that it doesn't just remember the training data, but it has the capacity to **generalize**
  - **Overfitting** means the model performs well on training data, but it fails to generalize
  - **Underfitting** means the model performs badly on both training and unseen data

- The **true error** is error on all the possible data points.
- The **empirical error** is the error on a finite set of samples.
  - Sometimes called the **test error**, or **sample error**, or **generalization error**

Evaluating a model

- A model is only powerful if it performs well on data which it has not seen before.
  - This means that it doesn't just remember the training data, but it has the capacity to **generalize**
  - **Overfitting** means the model performs well on training data, but it fails to generalize
  - **Underfitting** means the model performs badly on both training and unseen data

- The **true error** is error on all the possible data points.
- The **empirical error** is the error on a finite set of samples.
  - Sometimes called the **test error**, or **sample error**, or **generalization error**

- We want to minimize the true error, but it is impossible to measure.
- So we make sure that the empirical error is a *good estimate* of the true error.
  - How?

How do we avoid overfitting?

- By not computing empirical error on the same data which was used for training:
    - Hold out data for testing.

How do we avoid overfitting?

- By not computing empirical error on the same data which was used for training:
    - Hold out data for testing.
- By not choosing model hyperparameters to only fit the test set:
    - Hold out data for validation.
    - Use *cross-validation*

How do we avoid overfitting?

- By not computing empirical error on the same data which was used for training:
  - Hold out data for testing.
- By not choosing model hyperparameters to only fit the test set:
  - Hold out data for validation.
  - Use *cross-validation*
- By preferring simple models over complex models:
  - Occam's Razor principle.
  - Penalizing model complexity is called **regularization.**

How do we avoid overfitting?

- By not computing empirical error on the same data which was used for training:
  - Hold out data for testing.
- By not choosing model hyperparameters to only fit the test set:
  - Hold out data for validation.
  - Use *cross-validation*
- By preferring simple models over complex models:
  - Occam's Razor principle.
  - Penalizing model complexity is called **regularization.**
- By making sure data points are i.i.d. *(independent and identically distributed*):
  - i.i.d. means there is no bias when selecting the training set

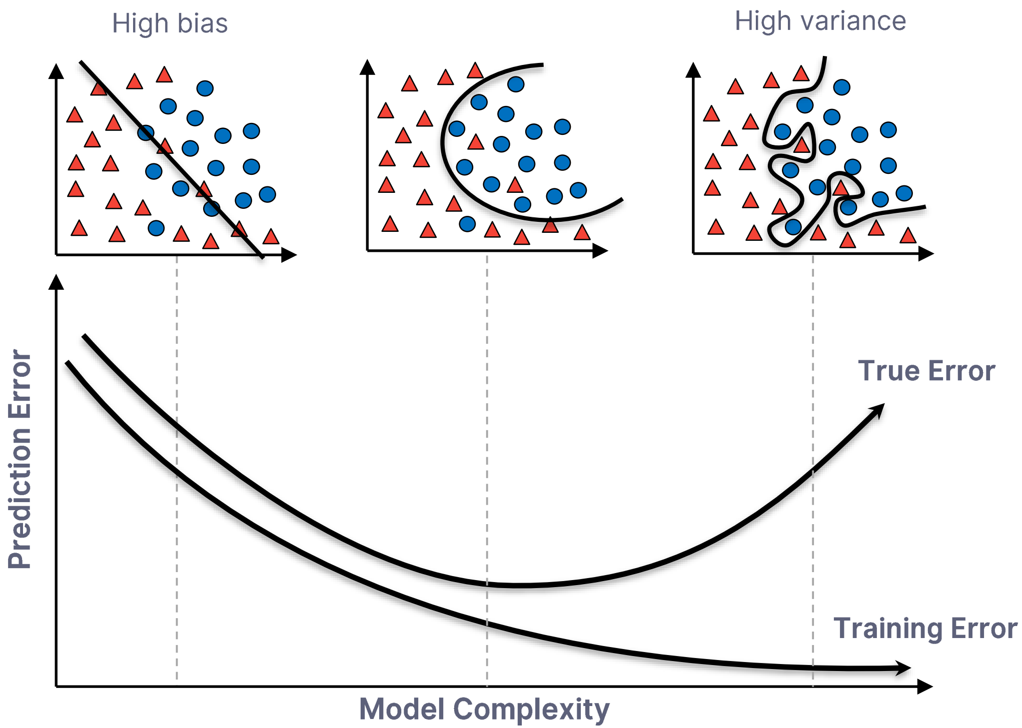
    → every point is selected independently and from the same distribution
  - This assumption is very often violated in practice!

How do we avoid overfitting?

- By not computing empirical error on the same data which was used for training:
  - Hold out data for testing.
- By not choosing model hyperparameters to only fit the test set:
  - Hold out data for validation.
  - Use *cross-validation*
- By preferring simple models over complex models:
  - Occam's Razor principle.
  - Penalizing model complexity is called **regularization.**
- By making sure data points are i.i.d. *(independent and identically distributed*):
  - i.i.d. means there is no bias when selecting the training set

    → every point is selected independently and from the same distribution
  - This assumption is very often violated in practice!
- By getting more data. ☺

Underfitting vs. Overfitting



"Everything should be made simple as possible, but not simpler"
Albert Einstein

I.I.D. Assumption

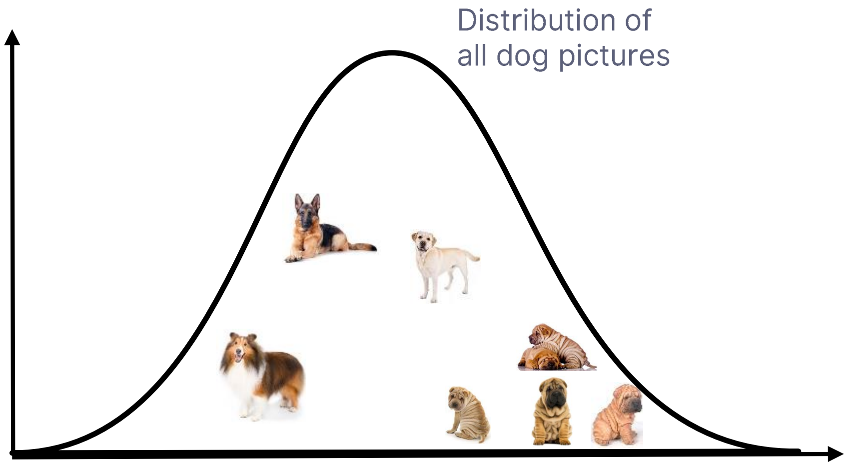
- Most algorithms assume data points are *independent and identically distributed.*

I.I.D. Assumption

- Most algorithms assume data points are *independent and identically distributed.*
- Let's say that we want to train a model to recognize pictures of dogs:
    - We give it this training set: ( ,  ,  , )

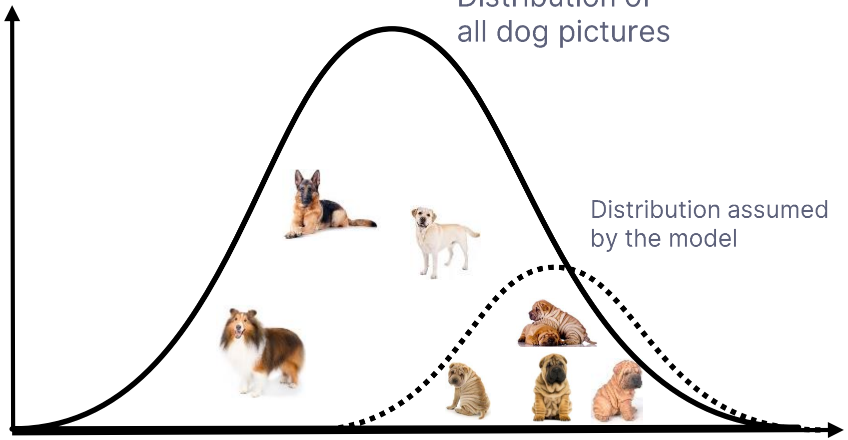
I.I.D. Assumption

- Most algorithms assume data points are *independent and identically distributed.*
- Let’s say that we want to train a model to recognize pictures of dogs:
  - We give it this training set: ( ,  ,  , )
- The examples are not i.i.d.



I.I.D. Assumption

- Most algorithms assume data points are *independent and identically distributed.*
- Let's say that we want to train a model to recognize pictures of dogs:
    - We give it this training set: 
- The examples are not i.i.d.
    - It likely that the model will make mistakes on pictures from different parts of the distribution



Mathematical Frame for Machine Learning

Learning a function

- We assume we have:
  - An instance space X, with a fixed (but unknown) distribution D_x
  - A target space Y and a function $f: X \to Y$

Learning a function

- We assume we have:
  - An instance space X, with a fixed (but unknown) distribution D_x
  - A target space Y and a function $f: X \to Y$
- We are given:
  - A set of labeled examples $\mathrm{E} \subseteq X \times Y = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}$

    such that: $\forall e = (x, y) \in E \Rightarrow f(x) = y$

    $x \sim D_x$ (x is drawn i.i.d. from D_x)
  - A set of allowed hypothesis $\mathcal{H}$

Learning a function

- We assume we have:
  - An instance space X, with a fixed (but unknown) distribution D_x
  - A target space Y and a function $f: X \to Y$
- We are given:
  - A set of labeled examples $\mathrm{E} \subseteq X \times Y = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}$
    such that: $\forall e = (x, y) \in E \Rightarrow f(x) = y$
    $x \sim D_x$ (x is drawn i.i.d. from D_x)
  - A set of allowed hypothesis $\mathcal{H}$

Sometimes there is no function f, only a distribution $\mathrm{D_{XY}}$ with

$$(\mathrm{x}, \mathrm{y}) \sim \mathrm{D_{XY}}$$

(x has a probability of having label y)

Learning a function

- We assume we have:
  - An instance space X, with a fixed (but unknown) distribution D_x
  - A target space Y and a function $f: X \to Y$
- We are given:
  - A set of labeled examples $\mathrm{E} \subseteq X \times Y = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}$

    such that: $\forall e = (x, y) \in E \Rightarrow f(x) = y$

    $x \sim D_x$ (x is drawn i.i.d. from D_x)
  - A set of allowed hypothesis $\mathcal{H}$
- We need to find:
  - A hypothesis $h \in \mathcal{H}$ s.t. $error_{D_x}(h) \stackrel{\text{def}}{=} \mathbb{E}_{\mathrm{D_x}}\left[\mathcal{L}\big(f(x), h(x)\big)\right]$ is minimal.
  - D_x is unknown, so we compute $error_S(h) \stackrel{\text{def}}{=} \frac{1}{|S|} \sum_{x \in S} \mathcal{L}\big(f(x), h(x)\big)$
    - where $S \subset X$ is a finite set (also i.i.d. from D_x)

Sometimes there is no function f, only a distribution $\mathrm{D_{XY}}$ with

$$(\mathrm{x}, \mathrm{y}) \sim \mathrm{D_{XY}}$$

(x has a probability of having label y)

$\mathcal{L}$ is the loss on a single example x

$\mathrm{error}_{D_x}(h)$ is the expected error over D_x

(i.e. the true error of h)

$\mathrm{error}_S(h)$ is the average error on set S

Learning a function

- The loss function should reflect the nature of the problem:
  - Classification:
    - Y is finite (usually small, sometimes binary)
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$ 0-1 loss
    - $\text{error}_{D_x}(h) = P_{x\sim D_x}\big(f(x) \neq h(x)\big)$

Learning a function

- The loss function should reflect the nature of the problem:
  - Classification:
    - Y is finite (usually small, sometimes binary)
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$ 0-1 loss
    - $\text{error}_{D_x}(h) = P_{x \sim D_x}\big(f(x) \neq h(x)\big)$
  - Regression:
    - Y is continuous
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$ Squared loss

Learning a function

- The loss function should reflect the nature of the problem:
  - Classification:
    - Y is finite (usually small, sometimes binary)
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$
    - $\text{error}_{D_x}(h) = P_{x \sim D_x}\big(f(x) \neq h(x)\big)$
  - Regression:
    - Y is continuous
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$

0-1 loss

Squared loss

$\hat{y}$ ("y hat") is the notation we'll use for predicted label

Learning a function

- The loss function should reflect the nature of the problem:
  - Classification:
    - Y is finite (usually small, sometimes binary)
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$

      0-1 loss
    - $\text{error}_{D_x}(h) = P_{x \sim D_x}\big(f(x) \neq h(x)\big)$

      $\hat{y}$ ("y hat") is the notation we'll use for predicted label
  - Regression:
    - Y is continuous
    - $\mathcal{L}\big(f(x), h(x)\big) = \mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$

      Squared loss
  - These are very common loss functions, but many others are used as well.

Keywords

Machine Learning

Label | Feature | Feature Vector | Data Point

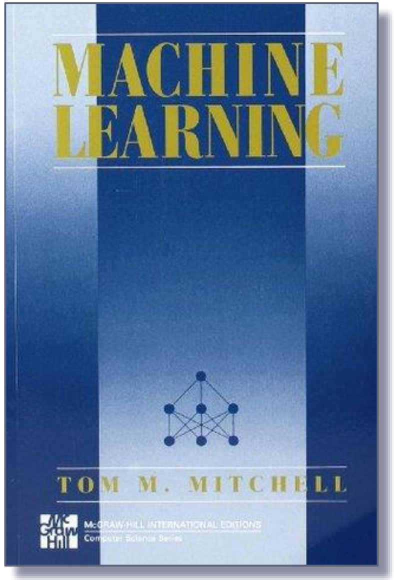
Training | Fitting | Inference | Model | Hypothesis | Algorithm | Hyperparameter

Supervised | Classification | Regression | Unsupervised | Clustering

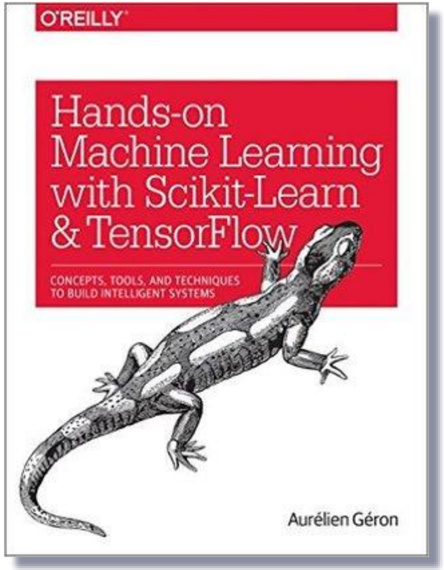
Reinforcement | Semi-supervised | Transfer Learning

Overfitting | Generalization | Occam’s Razor | Loss | True Error | Empirical Error | I.I.D.

Bibliography



Machine Learning,
Tom Mitchell



Hands-On Machine Learning with Scikit-Learn and TensorFlow,
Aurélien Géron

scikit-learn Tutorials

Google Machine Learning Crash Course

TensorFlow Tutorials