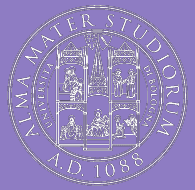


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

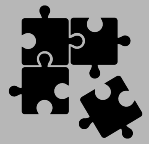
Intelligenza Artificiale

An introduction to Machine Learning

Ivan Heibi
Dipartimento di Filologia Classica e Italianistica (FICLIT)
Ivan.heibi2@unibo.it

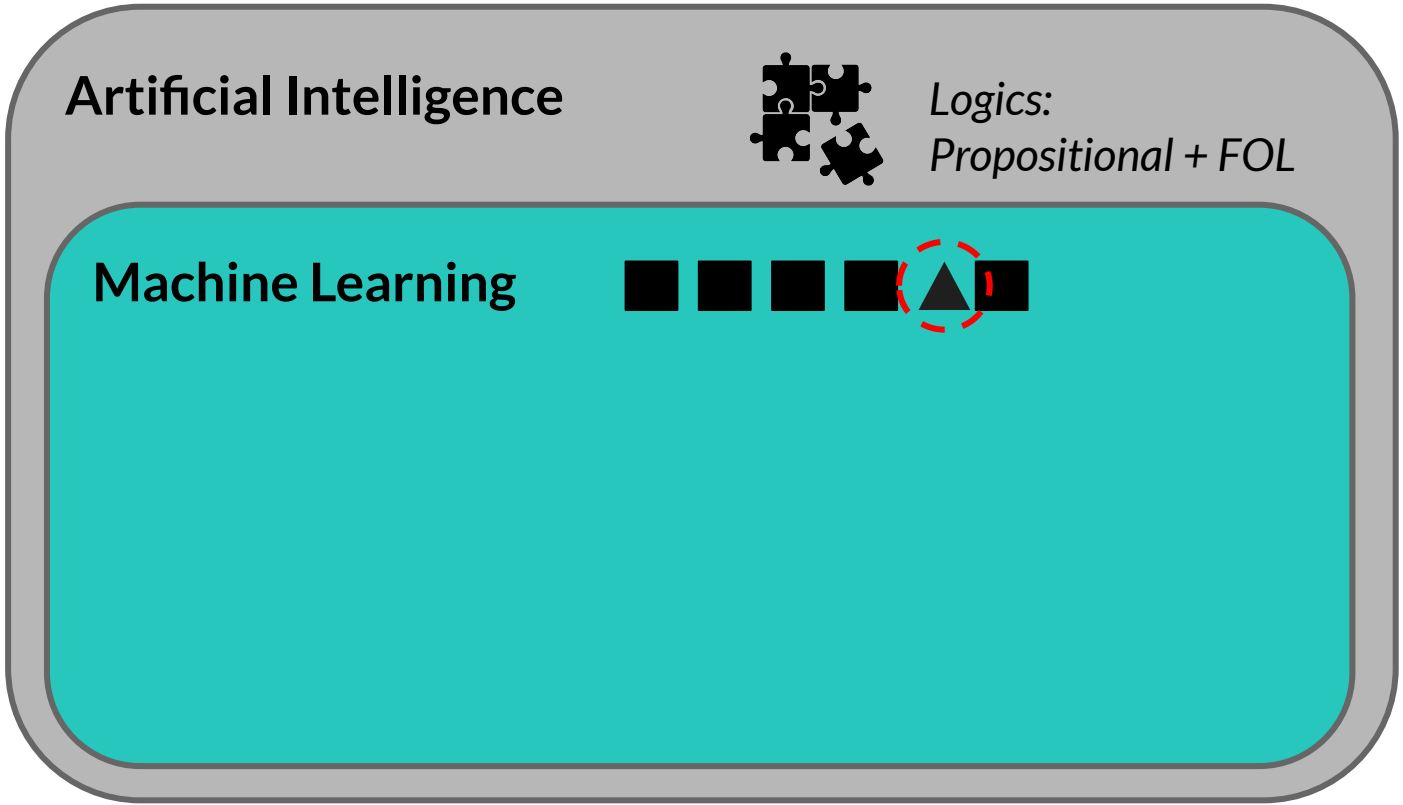
Artificial Intelligence an Overview

Artificial Intelligence

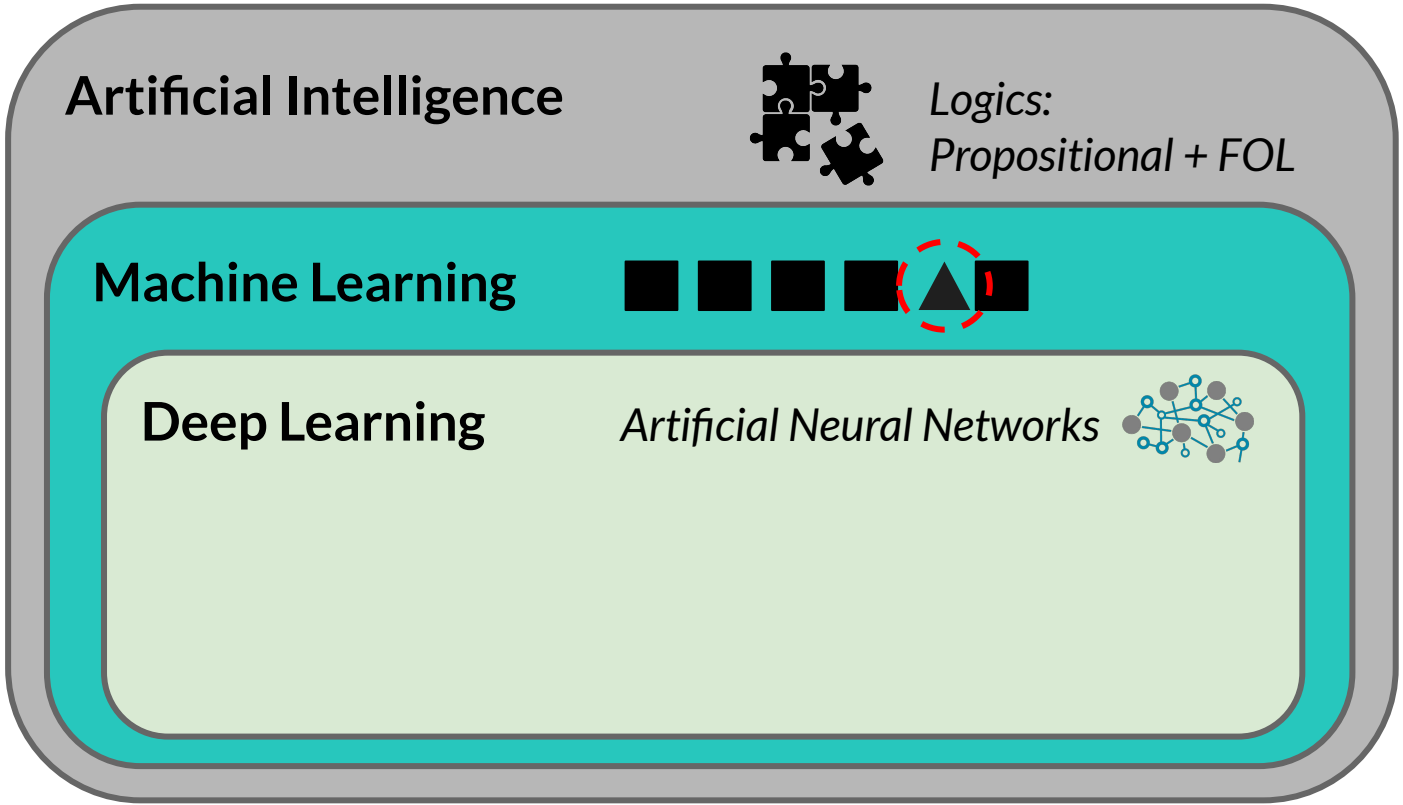


Logics:
Propositional + FOL

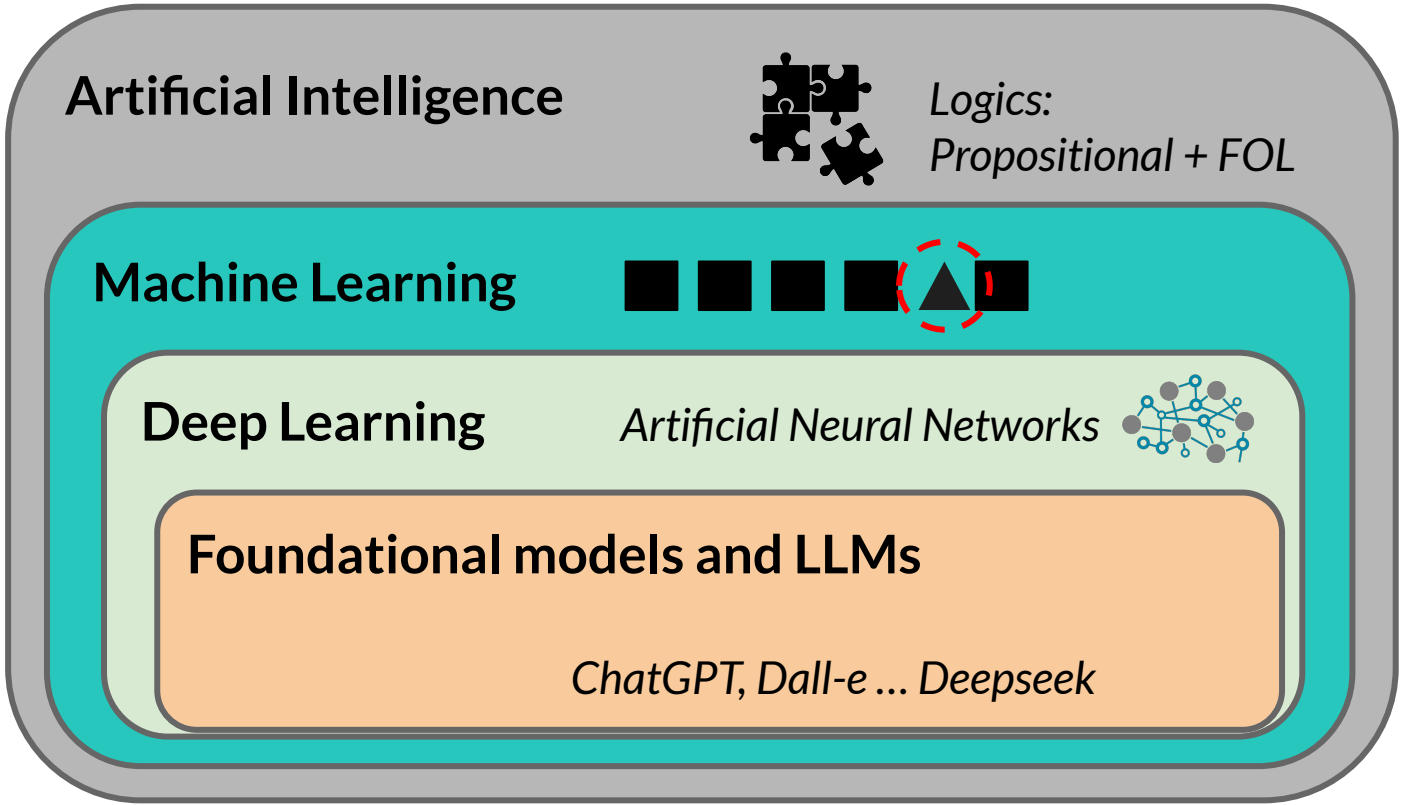
Artificial Intelligence an Overview



Artificial Intelligence an Overview

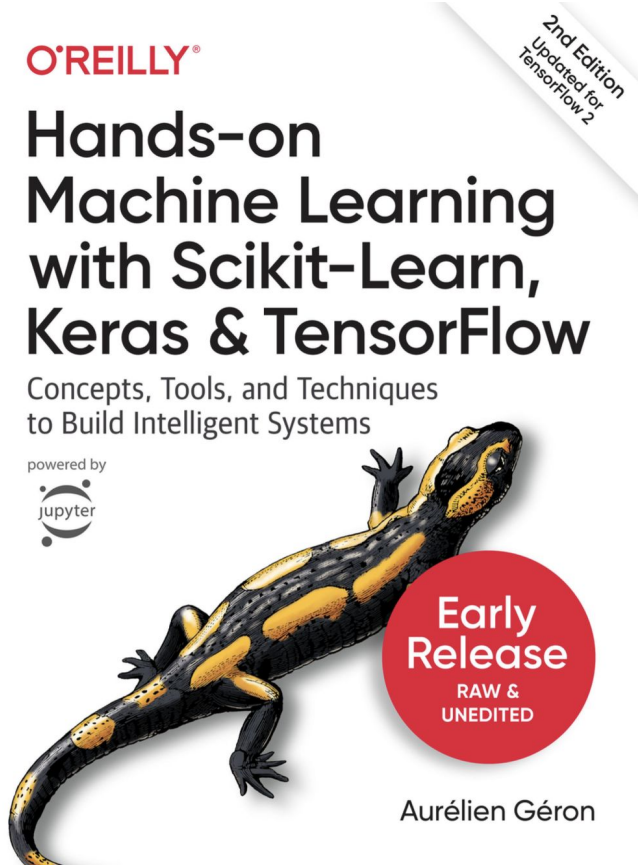


Artificial Intelligence an Overview



"AI, Machine Learning, Deep Learning and Generative AI Explained", ***IBM Technology,*** https://tinyurl.com/4csecc34 ***(youtube video)***

Bibliography



What is Machine Learning?

"A 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

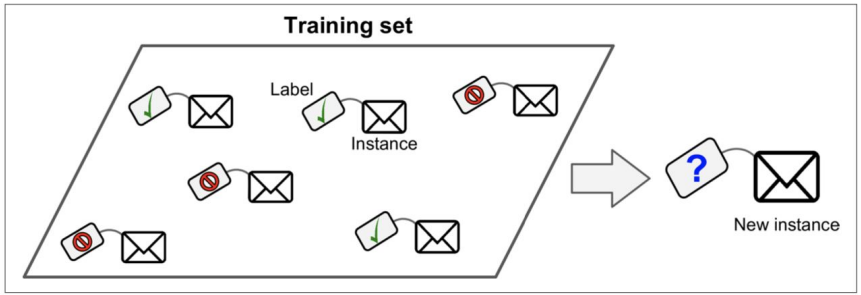


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

T = SPAM Classification
E = Labelled emails (Training Set)
P = Ratio of correctly classified emails

Why would we want an agent to learn?

If the design of the agent can be improved, why wouldn't the designers just program in that improvement to begin with?

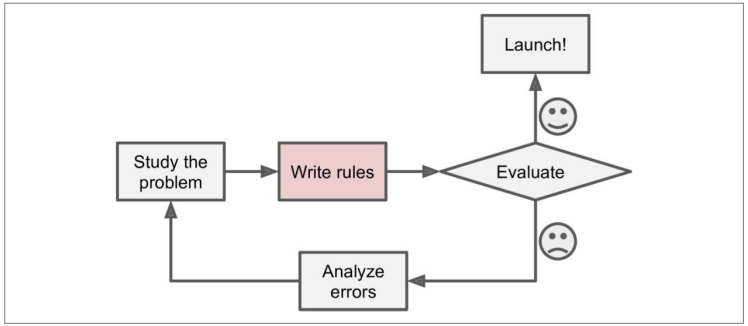


Figure 1-1. The traditional approach

Reasons:

1. The designer cannot anticipate all possible situations that the agent might find itself in.
2. The designer cannot anticipate all changes over time (agent should be able to adapt its behaviour)
3. Sometimes human programmers have no idea how to program a solution themselves!

Why would we want an agent to learn?

If the design of the agent can be improved, why wouldn't the designers just program in that improvement to begin with?

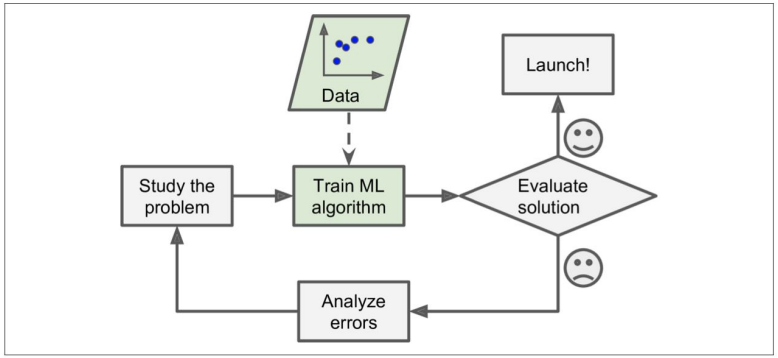


Figure 1-2. Machine Learning approach

SPAM keywords: "4U", "credit card", "free", "amazing"

Reasons:

1. **The designer cannot anticipate all possible situations that the agent might find itself in.**
2. The designer cannot anticipate all changes over time (agent should be able to adapt its behaviour)
3. Sometimes human programmers have no idea how to program a solution themselves!

Why would we want an agent to learn?

If the design of the agent can be improved, why wouldn't the designers just program in that improvement to begin with?

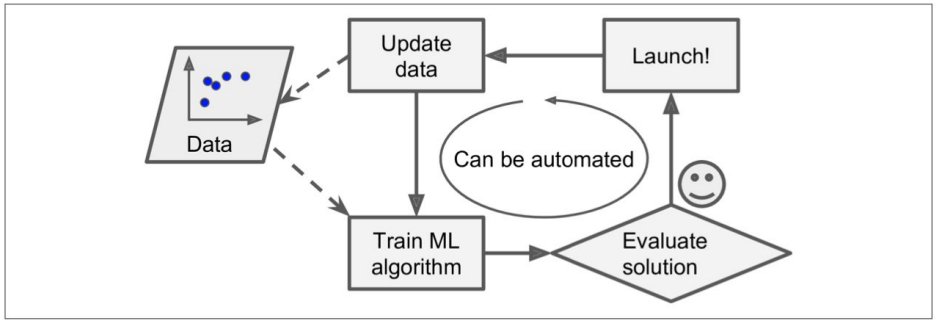


Figure 1-3. Automatically adapting to change

SPAM keywords:
"4U"-> "For U",
"credit card", "free",
"amazing"

Reasons:

1. The designer cannot anticipate all possible situations that the agent might find itself in.
2. **The designer cannot anticipate all changes over time (agent should be able to adapt its behaviour)**
3. Sometimes human programmers have no idea how to program a solution themselves!

Why would we want an agent to learn?

If the design of the agent can be improved, why wouldn't the designers just program in that improvement to begin with?

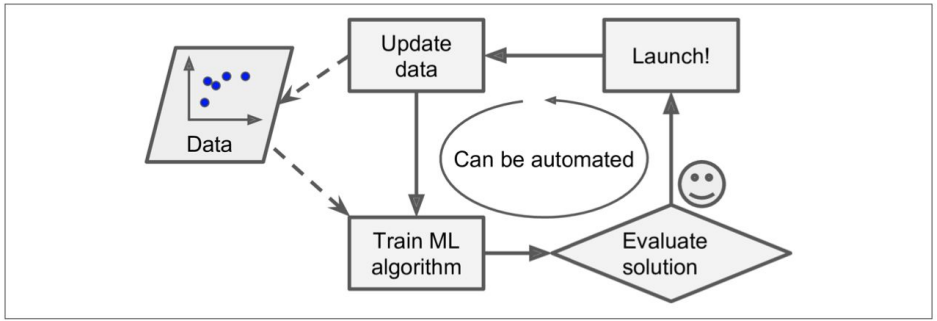


Figure 1-3. Automatically adapting to change

SPAM keywords:
"4U"-> "For U",
"credit card", "free",
"amazing"

Reasons:

1. The designer cannot anticipate all possible situations that the agent might find itself in.
2. The designer cannot anticipate all changes over time (agent should be able to adapt its behaviour)
3. **Sometimes human programmers have no idea how to program a solution themselves!**
   Example: Speech Recognition

Machine Learning can help humans learn

ML algorithms can be inspected to see what they have learned.

For instance, once the spam filter has been trained on enough spam, it can easily be inspected to reveal the list of words and combinations of words that it believes are the best predictors of spam.

Applying ML techniques to dig into large amounts of data can help discover patterns that were not immediately apparent. This is called **data mining**.

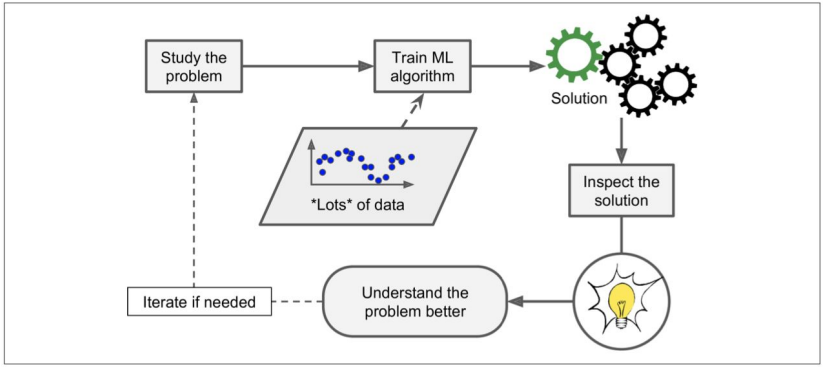


Figure 1-4. Machine Learning can help humans learn

Types of Machine Learning Systems

There are different types of Machine Learning systems.

It is useful to classify them in broad categories based on:

- **Whether or not they are trained with human supervision**
  supervised | *unsupervised* | *semi-supervised* | *Reinforcement Learning*
- **Whether or not they can learn incrementally on the fly**
  online learning | *batch learning*
- **Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do**
  instance-based | *model-based*

Supervised/Unsupervised learning

Machine Learning systems can be classified according to the amount and type of supervision they get during training.

We can distinguish four main categories:

- Supervised learning
- Unsupervised learning
- Semisupervised learning
- Reinforcement Learning

Supervised learning: classification

In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels.

A typical supervised learning task is **classification**.

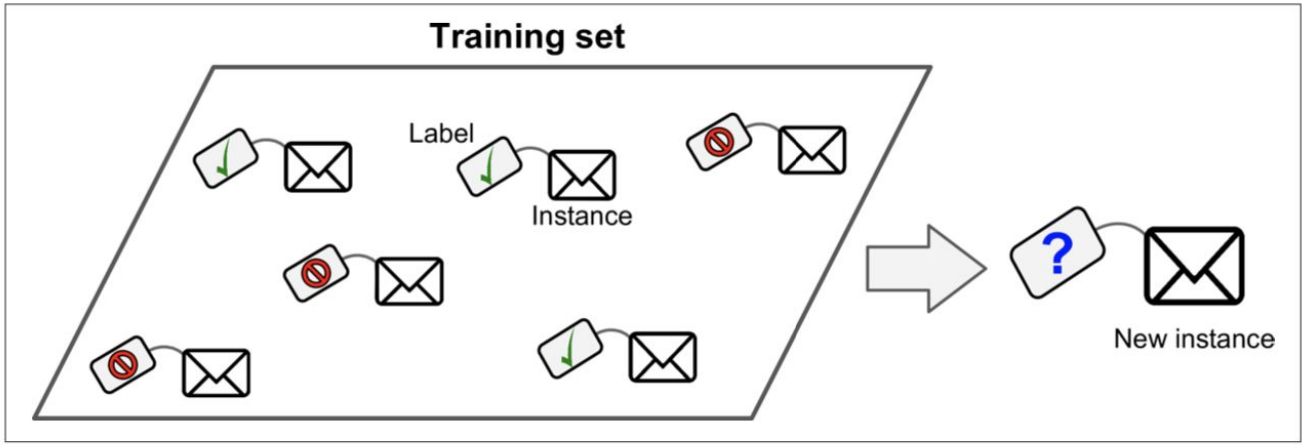


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

Supervised learning: regression

Another typical task is to predict a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc.) called predictors.

This sort of task is called **regression**.

To train the system, you need to give it many examples of cars, including both their predictors and their labels (i.e., their prices).

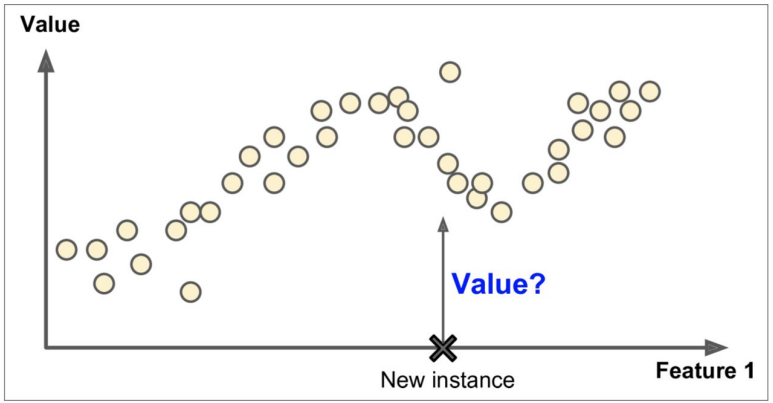


Figure 1-6. Regression

Unsupervised learning

In unsupervised learning the **training data is unlabeled**

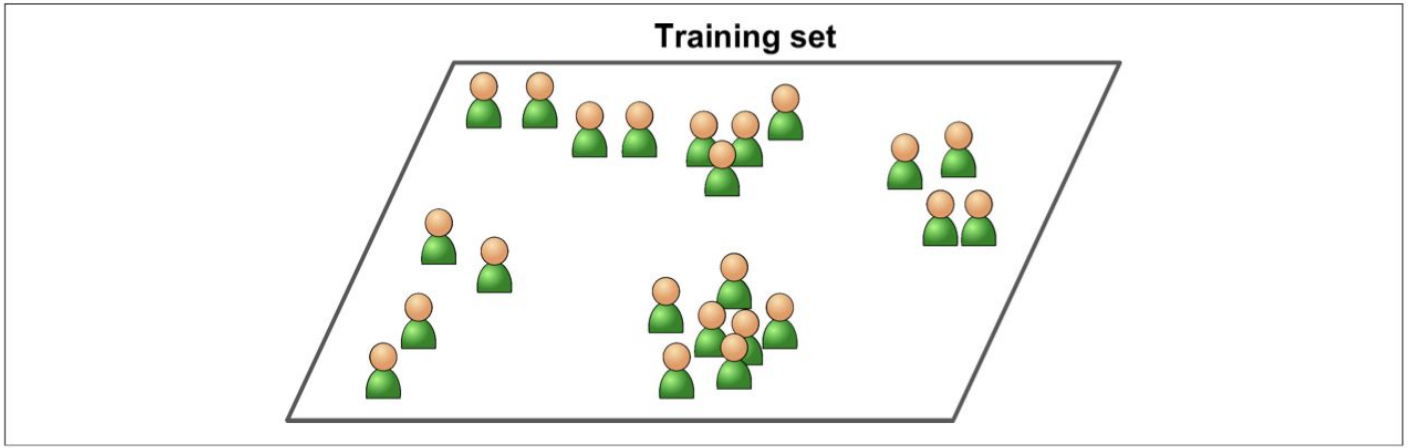


Figure 1-7. An unlabeled training set for unsupervised learning

Unsupervised learning

In unsupervised learning the training data is unlabeled

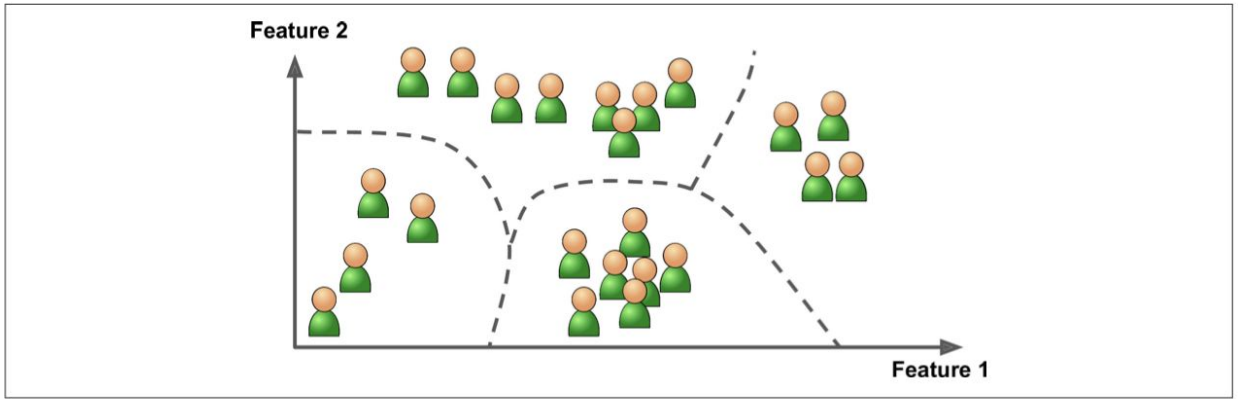


Figure 1-8. Clustering

Other tasks in unsupervised learning:

- **dimensionality reduction** (aiming at simplifying the data without losing too much information)
- **visualisation**,
- **anomaly detection** (automatic detection of outliers)

Unsupervised learning: pros and cons

[+] unlabeled data is often more abundant and easier to obtain than labeled data

[+] It can discover hidden patterns and structures in data that may not be obvious to humans.

[+] It can be used for data preprocessing tasks (e.g., cleaning and dimensionality reduction) which can improve the performance of other algorithms.

[-] It can be difficult to interpret the results of the algorithm, since we don't know exactly what it has learned.

[-] It can be difficult to evaluate the performance of the algorithm, since we don't have labels for the output data.

Semi-supervised learning

Some algorithms can deal with partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data. This is called **semi-supervised learning**.

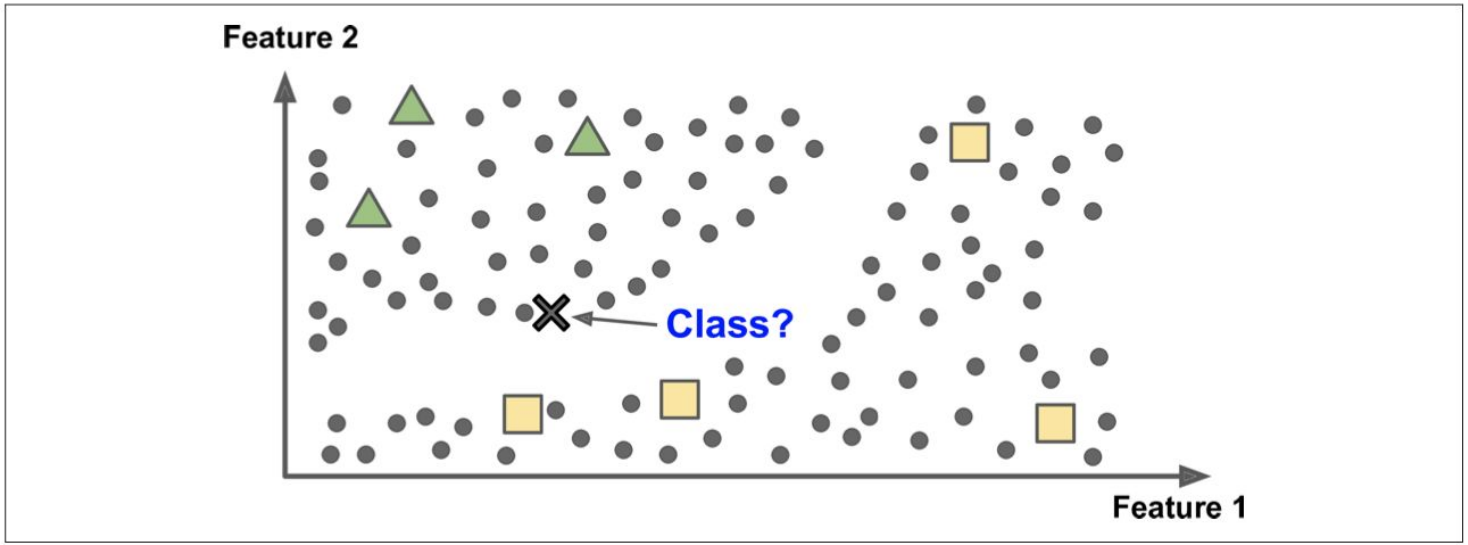


Figure 1-11. Semisupervised learning

Self assessment

https://forms.gle/pwLiBpE5zAdmXjcB9

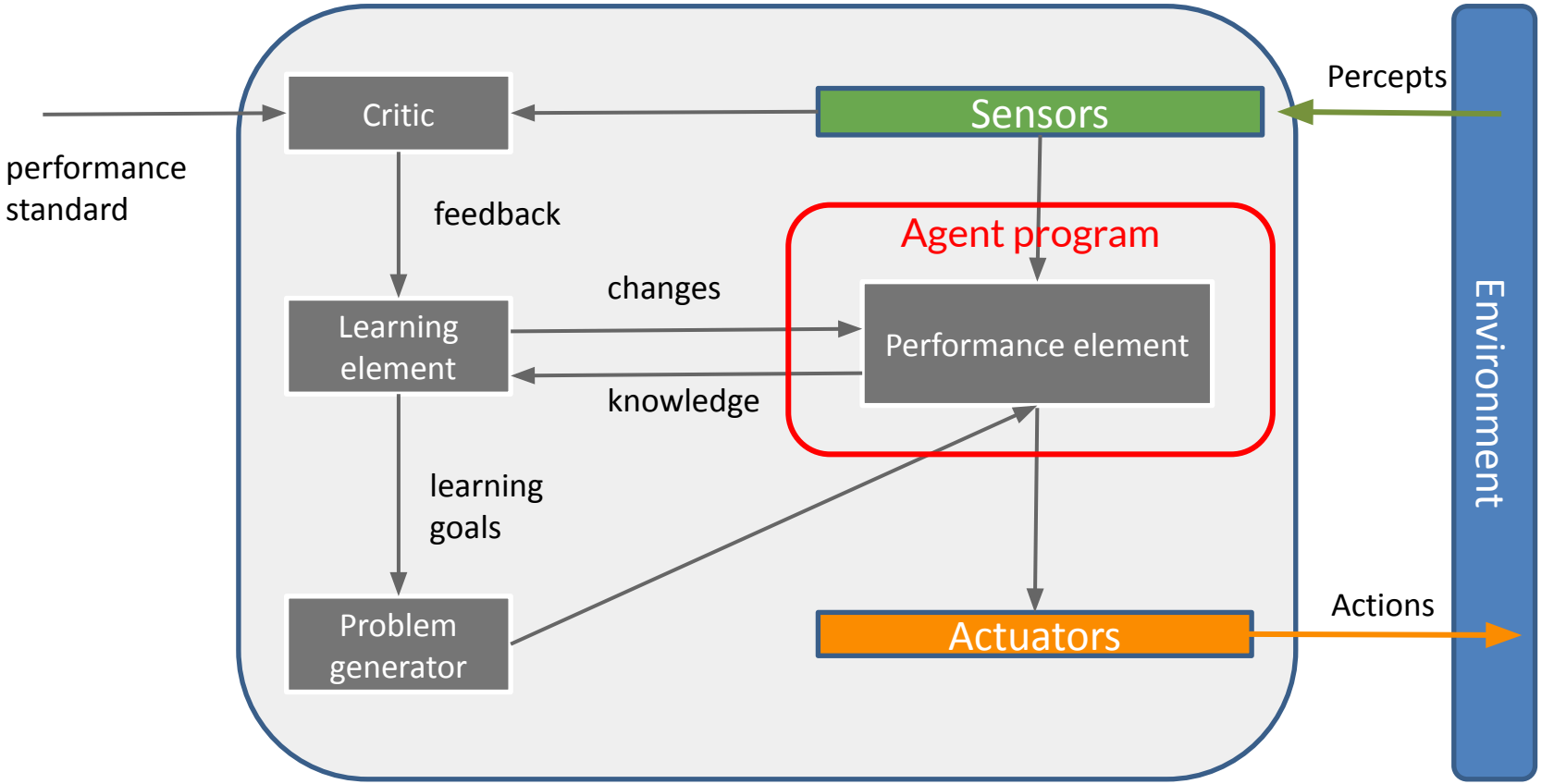
Reinforcement learning

A learning agent in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards.

It must then learn by itself what is the best strategy, called a **policy**, to get the most reward over time.

A policy defines what action the agent should choose when it is in a given situation.

Reinforcement learning: learning agent



Reinforcement learning

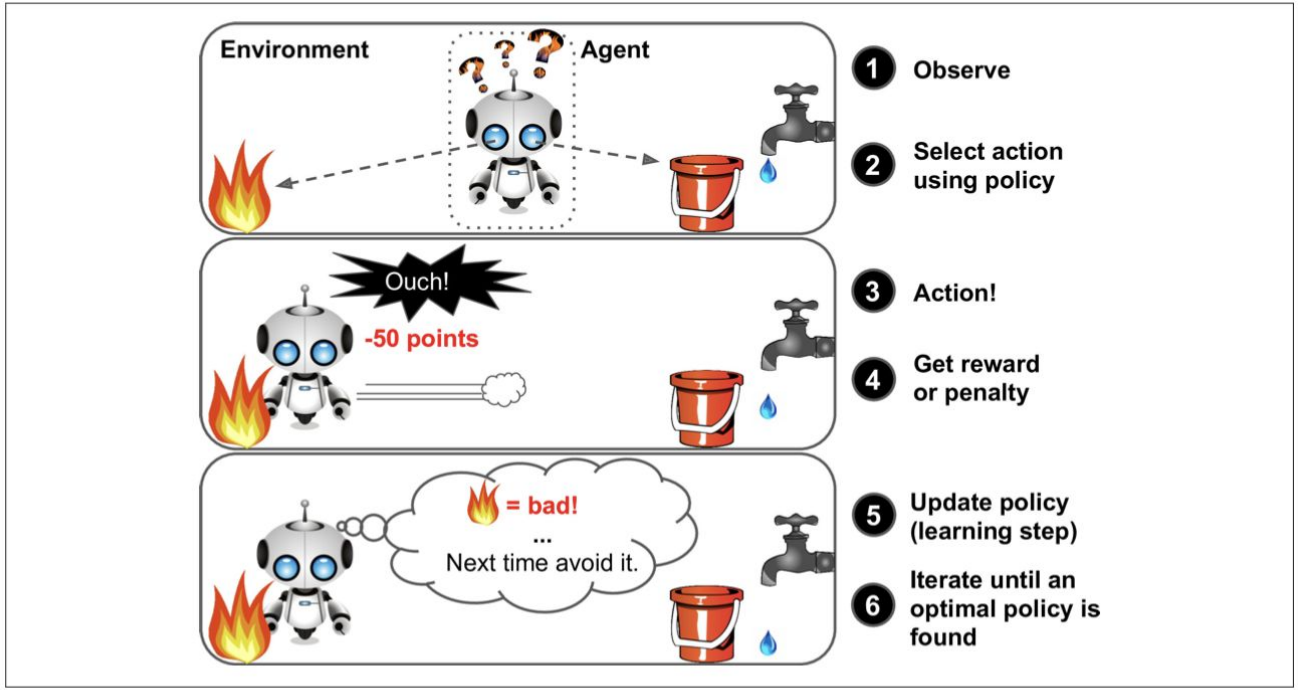


Figure 1-12. Reinforcement Learning

Reinforcement learning demo

http://projects.rajivshah.com/rldemo/

Batch and Online learning

Another criterion used to classify Machine Learning systems is whether or not the system can learn incrementally from a stream of incoming data.

We can distinguish two kinds of approaches:

- Batch learning
- Online learning

Batch learning

In Batch learning (also called offline learning) the system must be trained with all the data available.

It usually takes a lot of time.

If new data comes, the system needs to be retrained.

Online learning

In online learning, you train the system incrementally by feeding it data instances sequentially, either individually or by small groups called mini-batches.

Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.

Online learning is a valuable solution in case of: limited computational resources, huge training dataset (which may not fit in the system's memory.

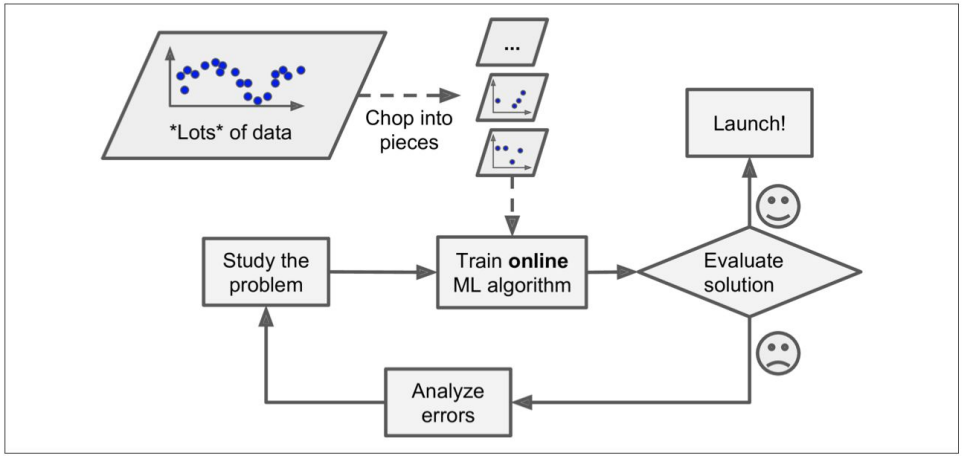


Figure 1-14. Using online learning to handle huge datasets

Instance-based/model-based learning

One more way to categorize Machine Learning systems is by how they generalize.

Most Machine Learning tasks are about making predictions.
> This means that given a number of training examples, the system needs to be able to generalize to examples it has never seen before.

Having a good performance measure on the training data is good, but insufficient; the true goal is to perform well on new instances.

There are two main approaches to generalization:

- instance-based learning
- model-based learning.

Instance-based learning

A trivial criterion to classify would be just flagging all emails that are **identical** to emails that have already been flagged by users.

A more general criterion would require **measuring the similarity** between emails and tagging the new email as the most similar one.

A similarity metric would be counting the number of words two emails have in common.

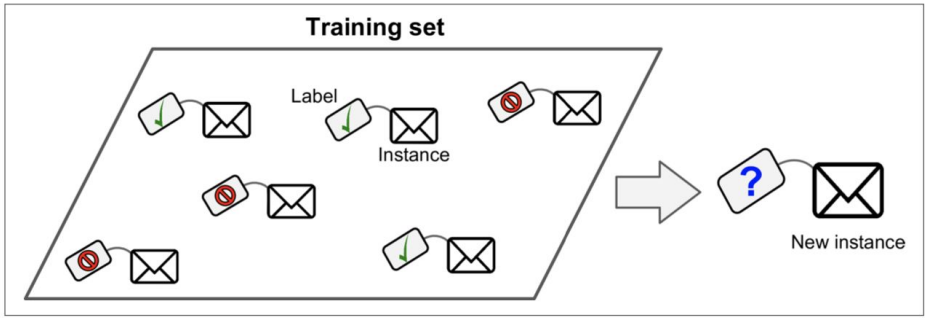


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

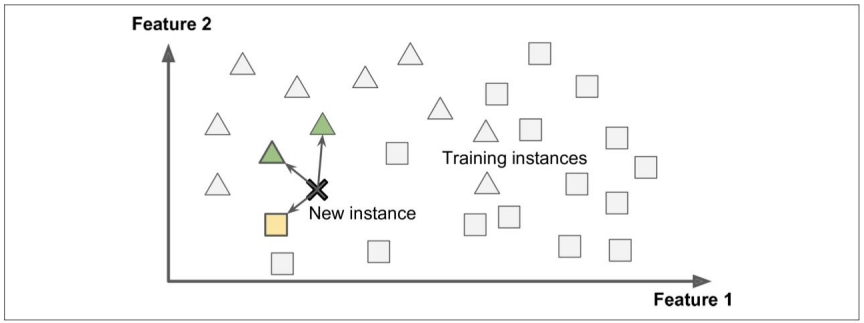


Figure 1-15. Instance-based learning

Model-based learning

In model-based learning a model is used to make prediction.

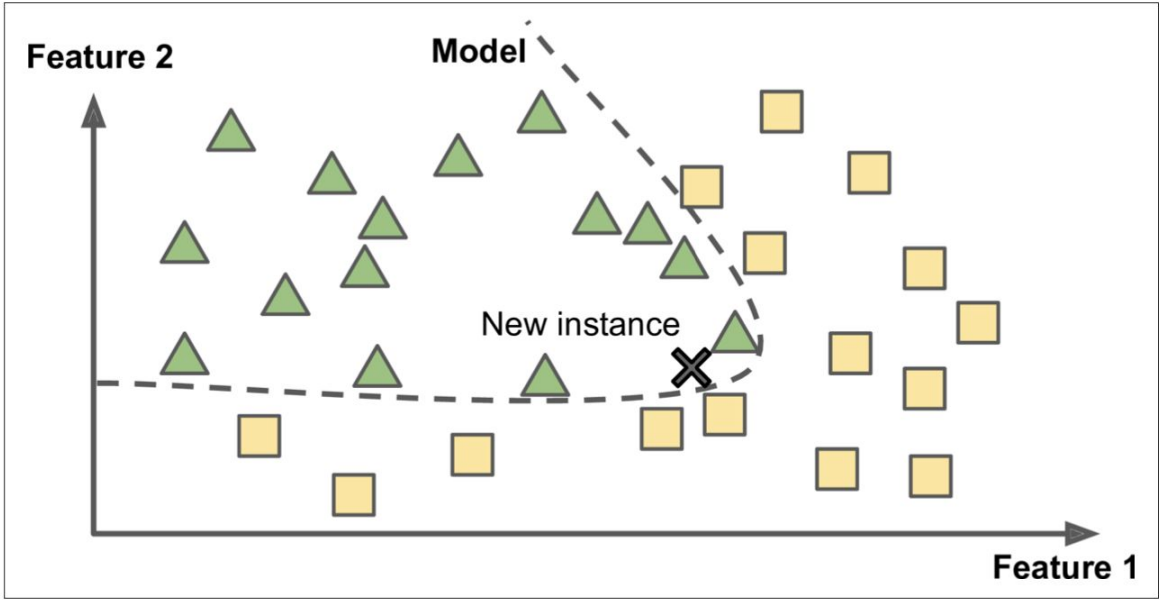


Figure 1-16. Model-based learning

Model-based learning

In model-based learning a model is used to make prediction.

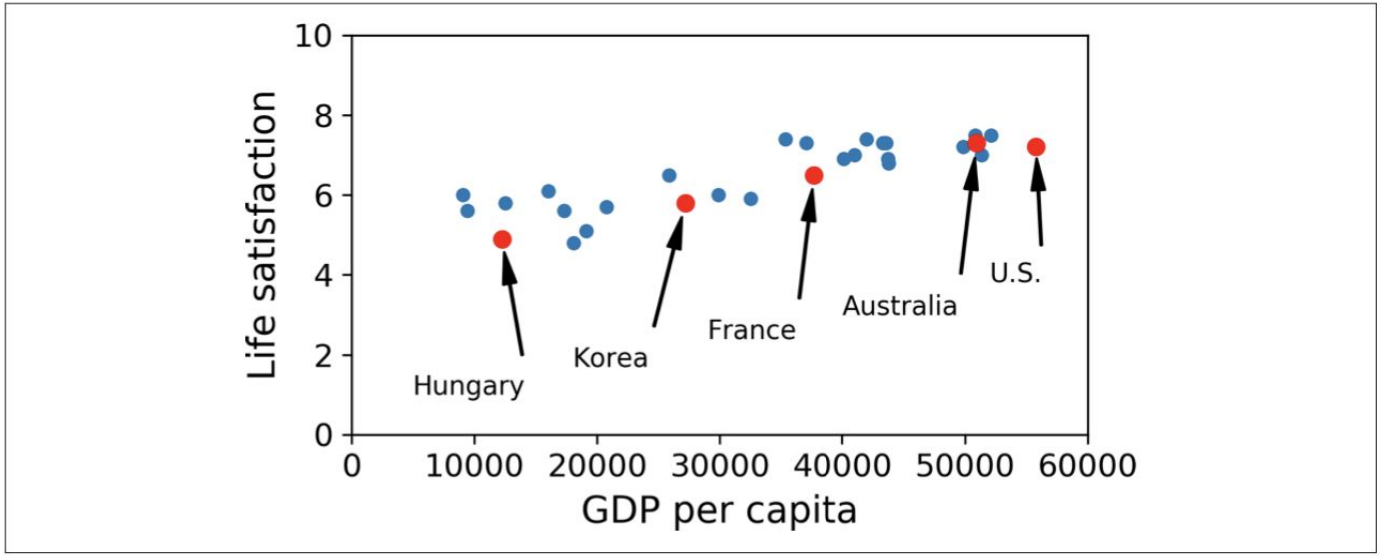


Figure 1-17. Do you see a trend here?

Model-based learning

$$life\_satisfaction = \theta_0 + \theta_1 * GDP\_per\_capita$$

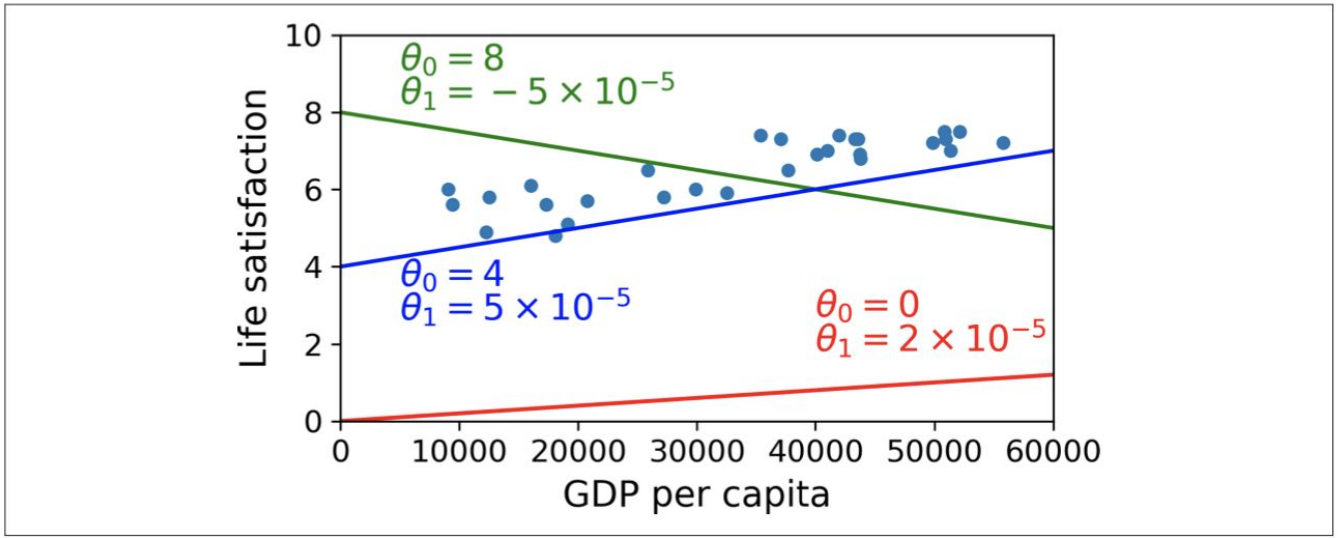


Figure 1-18. A few possible linear models

A fitness (cost) function measures how good(bad) the model is.
A fitness (cost) function has to be defined to define parameters (i.e. theta).

Model-based learning

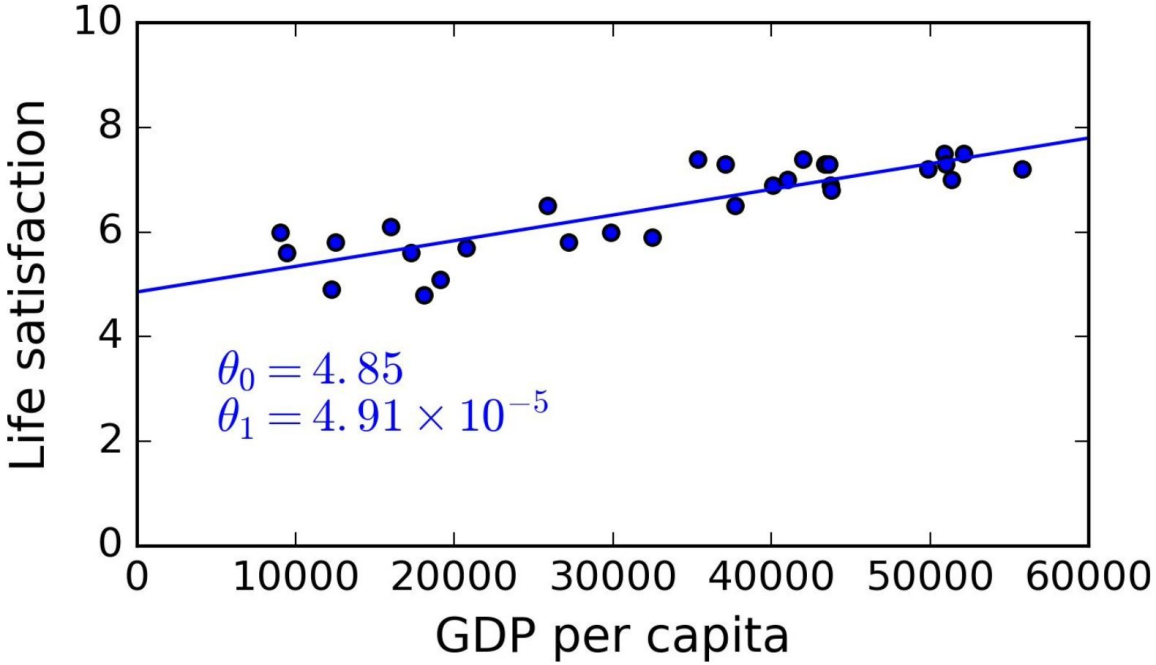


Figure 1-19. The linear model that fits the training data best

https://github.com/ageron/handson-ml/blob/master/04_training_linear_models.ipynb

Challenges

Model-based learning

The main Machine Learning task is to select a learning algorithm and train it on some data.

Therefore, there are two things that can go wrong:

- The data is wrong for your learning problem
- The algorithm selected is wrong for your learning problem

Training data: insufficient quantity

Most of Machine Learning algorithms take a lot of data to work properly.

Even for very simple problems you typically need thousands of examples, and for complex problems such as image or speech recognition, you may need millions of examples.

Contrarily, a toddler is able to learn what an apple is from a very examples.

In a very famous paper Banko and Brill [1] showed that very different Machine Learning algorithms, including fairly simple ones, performed almost identically well on a complex problem of WSD once they were given enough data.

[1] Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL '01). Association for Computational Linguistics, USA, 26–33. https://doi.org/10.3115/1073012.1073017

Training data: nonrepresentative

It is crucial that your training data be representative of the new cases you want to generalize to

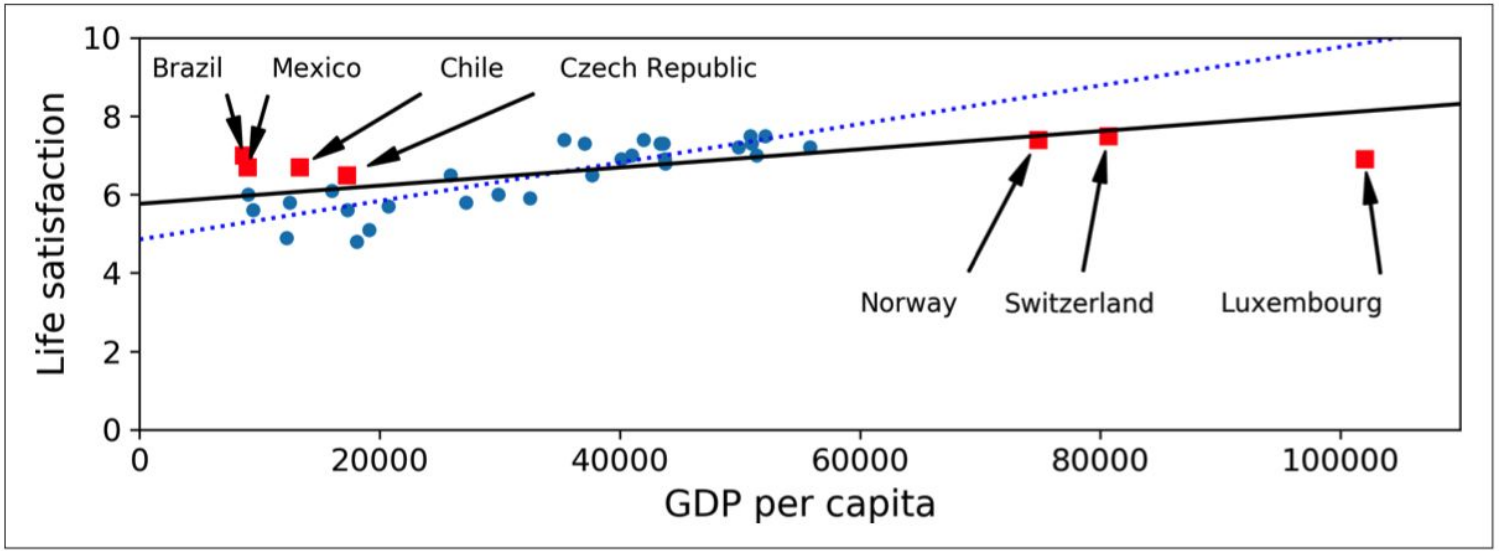


Figure 1-21. A more representative training sample

Training data: quality and irrelevant features

If your training data is full of errors, outliers, and noise (e.g., due to poor-quality measurements), it will make it harder for the system to detect the underlying patterns, so your system is less likely to perform well.

Your system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones.

A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called feature engineering, involves:

- Feature selection: selecting the most useful features to train on among existing features.
- Feature extraction: combining existing features to produce a more useful one (as we saw earlier, dimensionality reduction algorithms can help).
- Creating new features by gathering new data

Algorithm challenges: Overfitting the training data

Overfitting when a model performs well on the training data, but it does not generalize well on new data.

Underfitting (the opposite of overfitting) it occurs when a model is too simple to learn the underlying structure of the data.

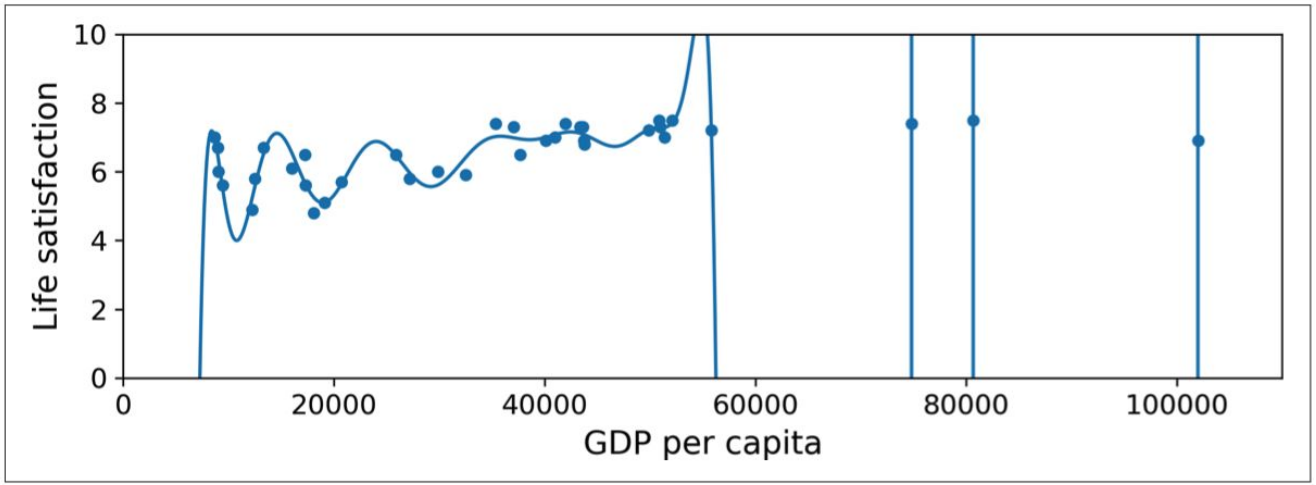


Figure 1-22. Overfitting the training data

Machine learning bias

Machine learning **bias** is a phenomenon that occurs when an algorithm produces results that are systemically prejudiced due to erroneous assumptions in the machine learning process.

Machine learning bias generally stems from problems introduced by the individuals who design and/or train the machine learning systems.

These individuals could either:

- create algorithms that reflect unintended cognitive biases or real-life prejudices.
- Or the individuals could introduce biases because they use incomplete, faulty or prejudicial data sets to train and/or validate the machine learning systems.

Testing and validating

The only way to know how well a model will generalize to new cases is to actually try it out on new cases.

This is done by splitting the available data into: **training** set and **test** set.

It is customary to train algorithms on the 80% of data, and testing them on the remaining 20%.

Hyperparameter tuning

A hyperparameter is a parameter whose value is used to control the learning process.

Different model training algorithms require different hyperparameters, some simple algorithms require none.

Tuning hyperparameters may improve performance over training data.

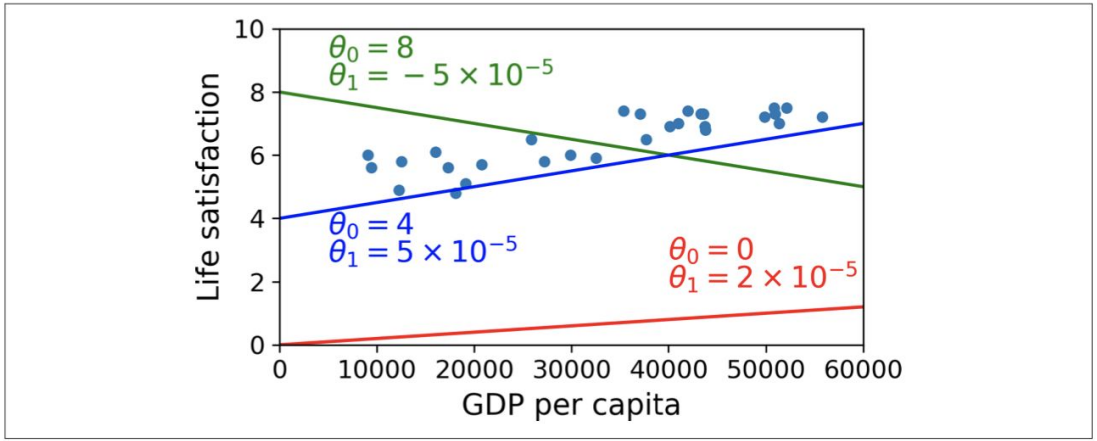


Figure 1-18. A few possible linear models

Classification

Classification is the problem of identifying which of a set of categories an observation/instance/example belongs to.

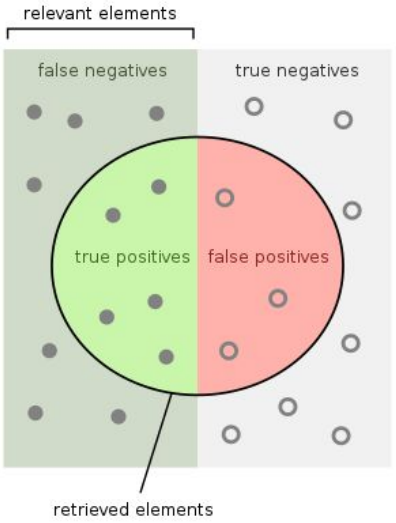
For example, assigning a given email to the "spam" or "non-spam" class.

Evaluating a classifier: precision and recall

Precision and Recall are performance metrics that apply to data retrieved from a collection.

Consider a text search scenario over a corpus of documents, e.g. searching the web. Suppose you enter a text string, e.g. “cats with attitude”. The search engine returns a number of documents (i.e. the web pages). You typically have that:

- Only a fraction of the pages are relevant to your search (**true positive**).
- The search engine does not return other relevant pages (**false negative**).



Precision is the fraction of relevant documents among the retrieved ones.
Recall is the fraction of relevant documents that were retrieved.

$$\text{recall} = \frac{TP}{TP + FN} \qquad \text{precision} = \frac{TP}{TP + FP}$$

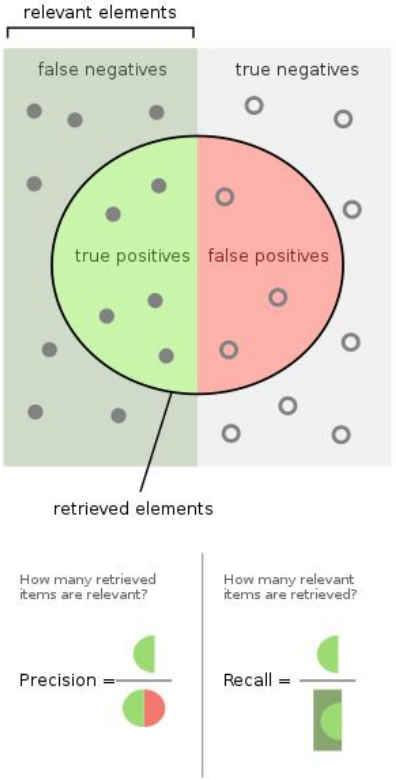
Evaluating a classifier: F-Score

The **F-score** is a measure of a test's accuracy.
It is calculated from the precision and recall of the test.

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}.$$

The F1 score is the harmonic mean of the precision and recall. It thus symmetrically represents both precision and recall in one metric.

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}.$$



Evaluating a classifier: Precision/recall trade off

The F1 score favours classifiers that have similar precision and recall.

This is not always what you want: in some contexts, you mostly care about precision, and in other contexts you really care about recall, e.g.:

Detecting safe videos for kids (you would probably favour precision)
Detecting shoplifters (you would probably favour recall)

Unfortunately, increasing precision reduces recall

precision and recall: example

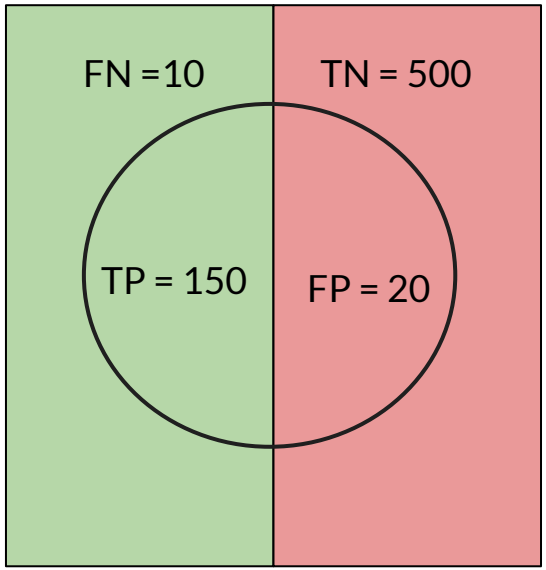
A binary classification problem to identifying whether an email is spam (positive class) or non-spam (negative class).

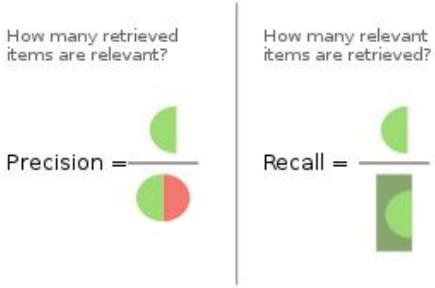
True Positive (TP) = 150
- Reality: spam
- ML model predicted: spam

False Positive (FP) = 20
- Reality: non-spam
- ML model predicted: spam

False Negative (FN) = 10
- Reality: spam
- ML model predicted: non-spam

True Negative (TN) = 500
- Reality: non-spam
- ML model predicted: non-spam





Precision
= 150/170
= 0.882

Recall
= 150/160
= 0.938

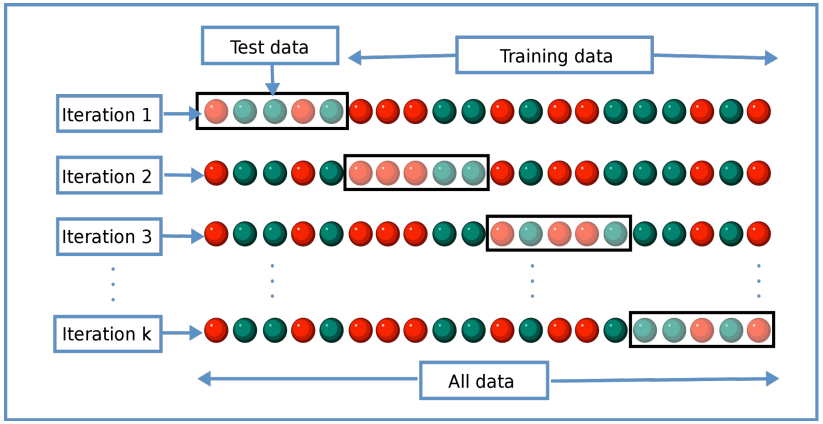
$$F_1 = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}.$$

$$= \frac{2 * 150}{2 * 150 + 20 + 10}$$

$$= 0.909$$

Evaluating a classifier: K-fold cross-validation

K-fold cross-validation means splitting the training set into K-folds (e.g. three), then making predictions and evaluating them on each fold using a model trained on the remaining folds.

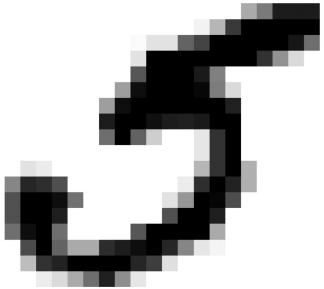


1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Classification task: MNIST dataset

Figure 3-1. A few digits from the MNIST dataset

- 70,000 images
- Each image is 28x28 pixels
- Therefore, we can model images with 784 features (one per pixel).
- Each feature is an integer representing the intensity of the pixel, ranging from 0 (white) to 255 (black)



Binary classification: MNIST dataset

Instead to train the algorithm to identify all the digits (i.e. 0-9), let's simplify the problem in the identification of the digit 5.

In this case, we have a **binary classification**, since the algorithm has to distinguish between two classes: images picturing the digit 5, and images that do represent picture the digit 5.

Binary classification: MNIST dataset

Binary classification: MNIST dataset

Shuffle examples!

Binary classification: MNIST dataset

Collect new examples and balance classes via undersampling/oversampling of the training and test set.

Confusion matrix

The general idea is to count the number of times instances of class A are classified as class B. For example, it allows you to know the number of times the classifier confused images of 5s with other digits.

Each row in a confusion matrix represents an actual class, while each column represents a predicted class.

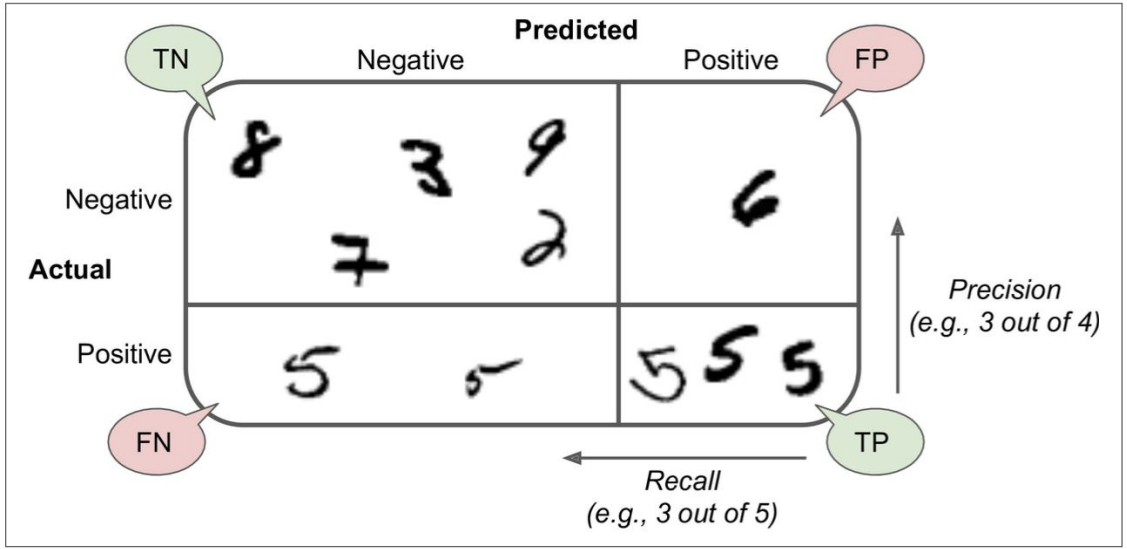


Figure 3-2. An illustrated confusion matrix

Multiclass classification

Multiclass classifiers can distinguish between **more than two classes.**

Some algorithms (such as Random Forest classifiers or naive Bayes classifiers) are capable of handling multiple classes directly.

Others (such as Support Vector Machine classifiers or Linear classifiers) are strictly binary classifiers.

However, there are various strategies that you can use to perform multiclass classification using multiple binary classifiers.

The most common are: **One-versus-All (OvA) and One-versus-One (OvO)**.

Multiclass classification: One-versus-All

Suppose you want to train a classifier to distinguish all the digits, i.e. 0 to 9.

One way to create such a system is to train 10 binary classifiers, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).

Then when you want to classify an image, you get the decision score from each classifier for that image and you select the class whose classifier outputs the highest score.

This is called the one-versus-all (OvA) strategy (also called one-versus-the-rest).

Multiclass classification: One-versus-One

Another strategy is to train a binary classifier for every pair of digits: one to distinguish 0s and 1s, another to distinguish 0s and 2s, another for 1s and 2s, and so on.

This is called the one-versus-one (OvO) strategy.

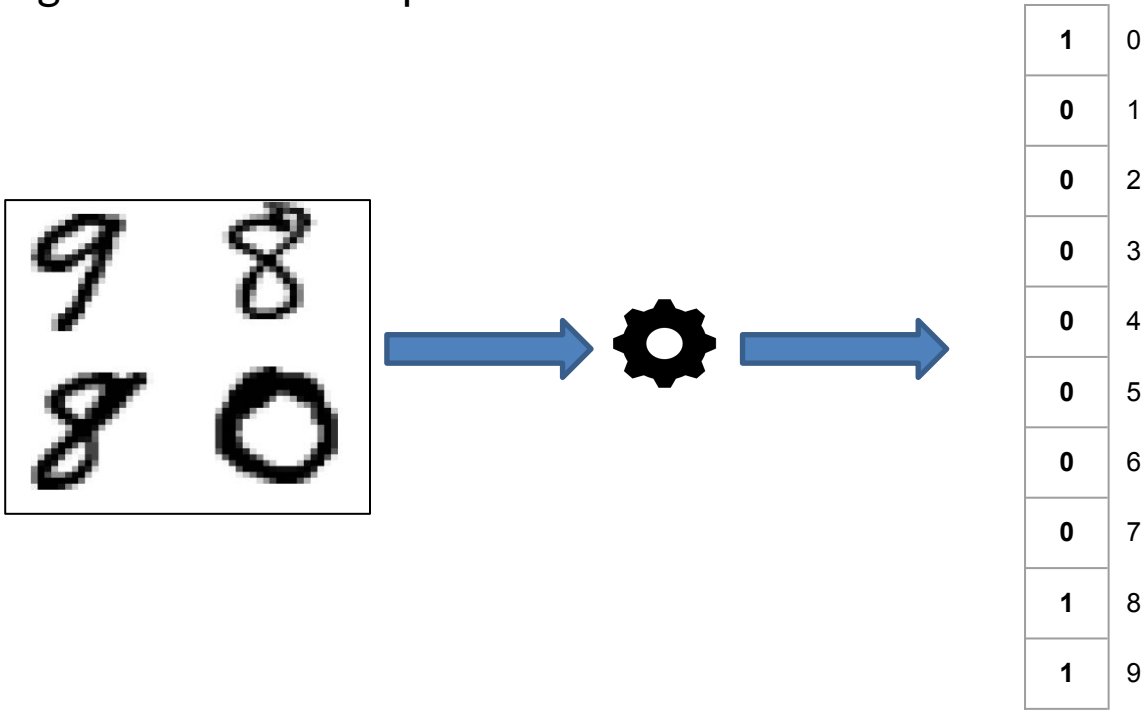
If there are N classes, you need to train $N \times (N - 1) / 2$ classifiers.

Some algorithms (such as Support Vector Machine classifiers) scale poorly with the size of the training set, so for these algorithms, OvO is preferred since it is faster to train many classifiers on small training sets than training a few classifiers on large training sets.

For most binary classification algorithms, however, OvA is preferred.

Multilabel classification

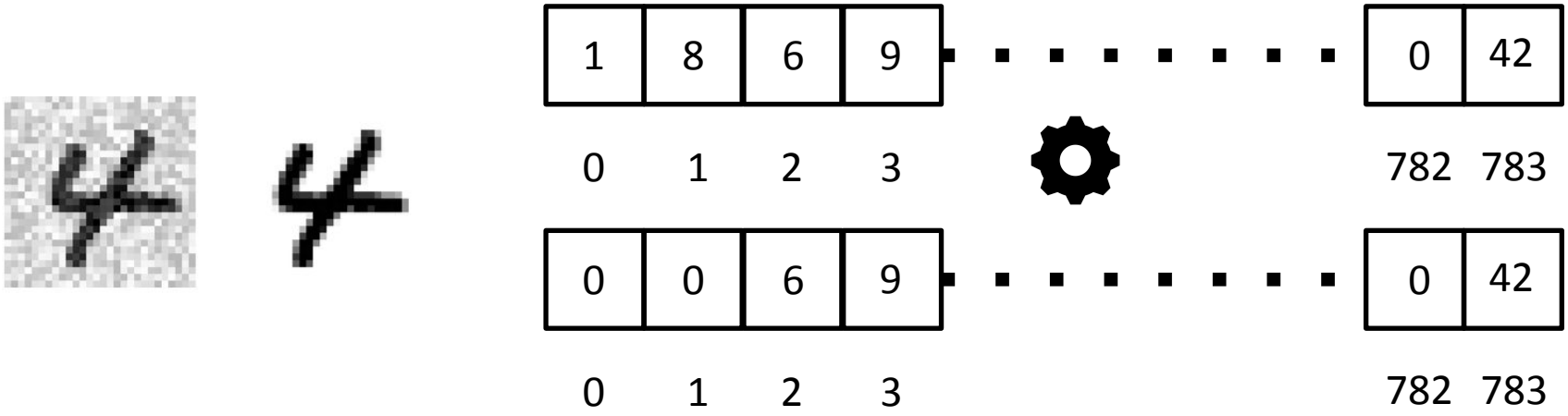
Multilabel classification is a variant of the classification problem where multiple labels (classes) can be assigned to each example.



Multioutput classification

Multioutput classification is a generalization of the multilabel classification, where each label can be multiclass (i.e., it can have more than two possible values).

For example, consider a system that removes noise from images. It will take as input a noisy digit image, and it will output a clean digit image, represented as an array of pixel intensities. Notice that the classifier's output is multilabel (one label per pixel) and each label can have multiple values (pixel intensity ranges from 0 to 255). It is thus an example of a multioutput classification system.



Unsupervised learning

Wouldn’t it be great if the algorithm could just exploit the unlabeled data without needing human intervention?

Clustering

In this presentation, we dive into a specific unsupervised learning task: clustering.

Clustering is the task of identifying similar instances and assigning them to clusters, i.e., groups of similar instances.

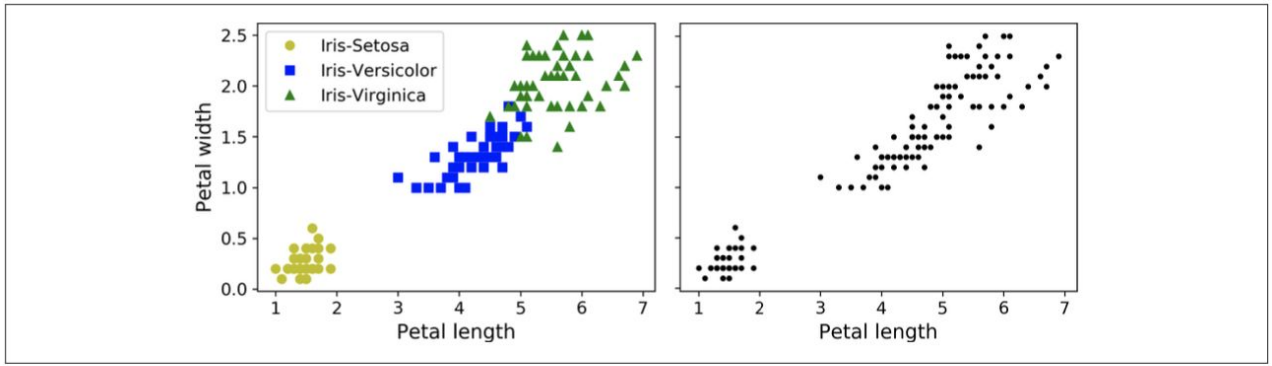


Figure 9-1. Classification (left) versus clustering (right)

Examples of applications: customer segmentation, data analysis, dimensionality reduction, anomaly detection, propagating labels for a partially annotated dataset, search engine

K-means

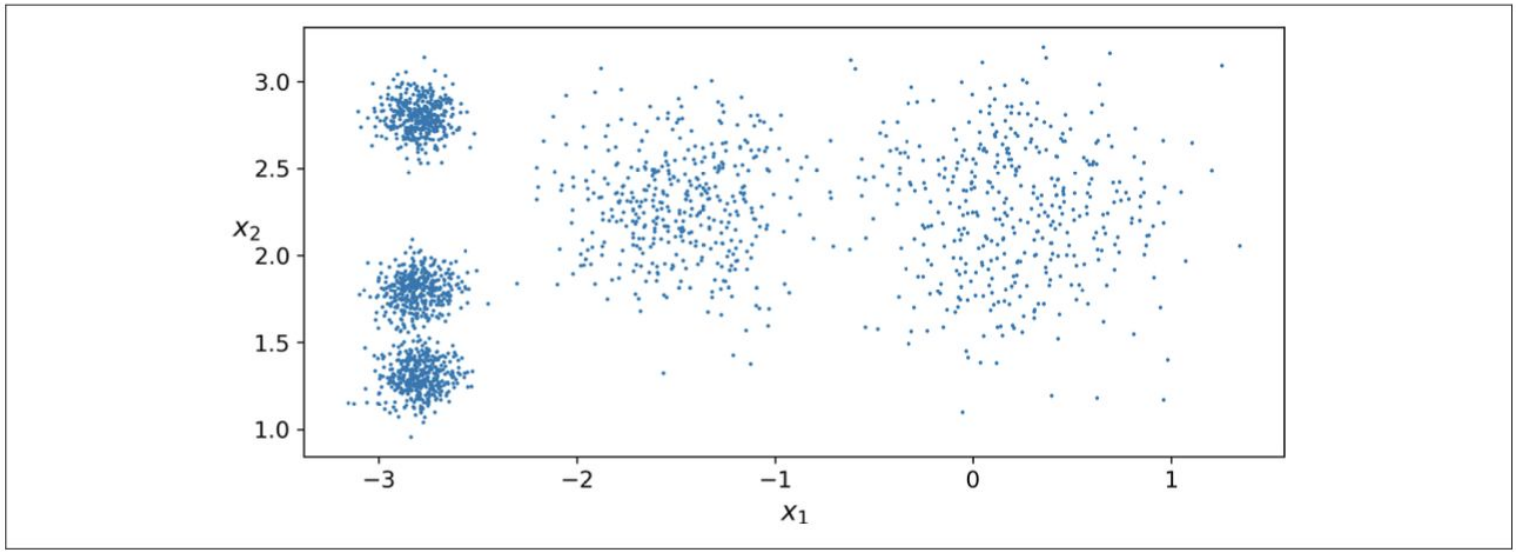


Figure 9-2. An unlabeled dataset composed of five blobs of instances

K-means

Suppose you were given the centroids: you could easily label all the instances in the dataset by assigning each of them to the cluster whose centroid is closest.

Conversely, if you were given all the instance labels, you could easily locate all the centroids by computing the mean of the instances for each cluster.

But, typically, you are given neither the labels nor the centroids.

- Start by placing the centroids randomly (e.g., by picking k instances at random and using their locations as centroids).
- Assign instances to the closest centroid
- Compute new centroids
- Continue until the centroids stop moving

Online demo: https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

K-means

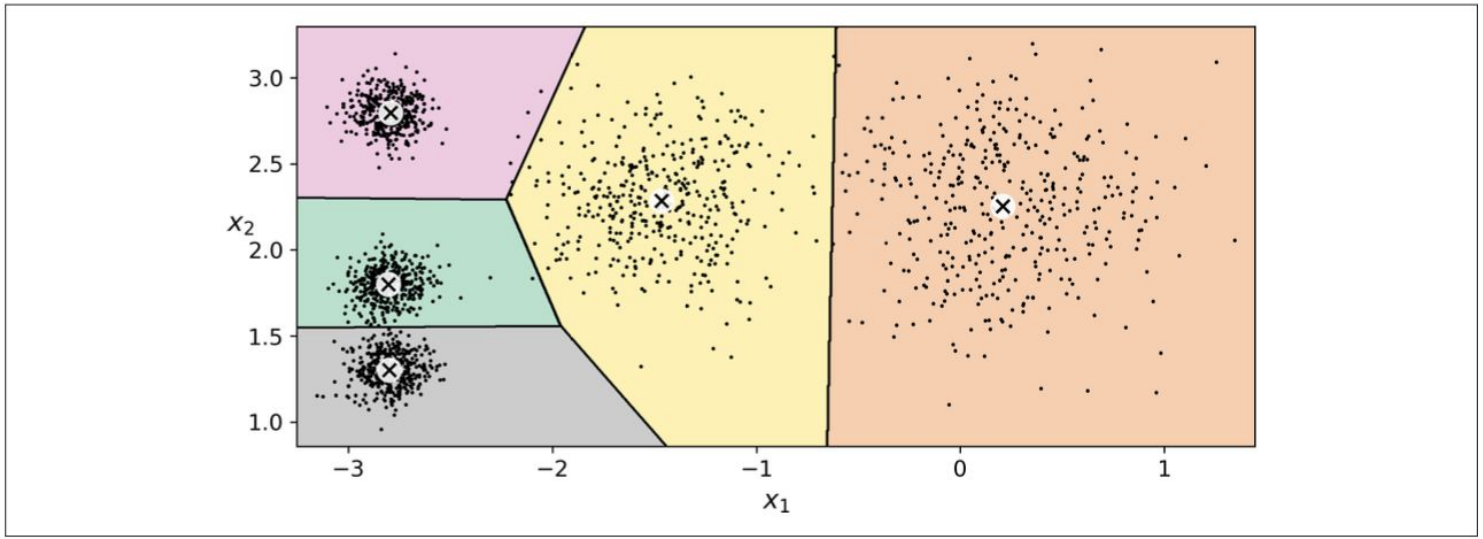
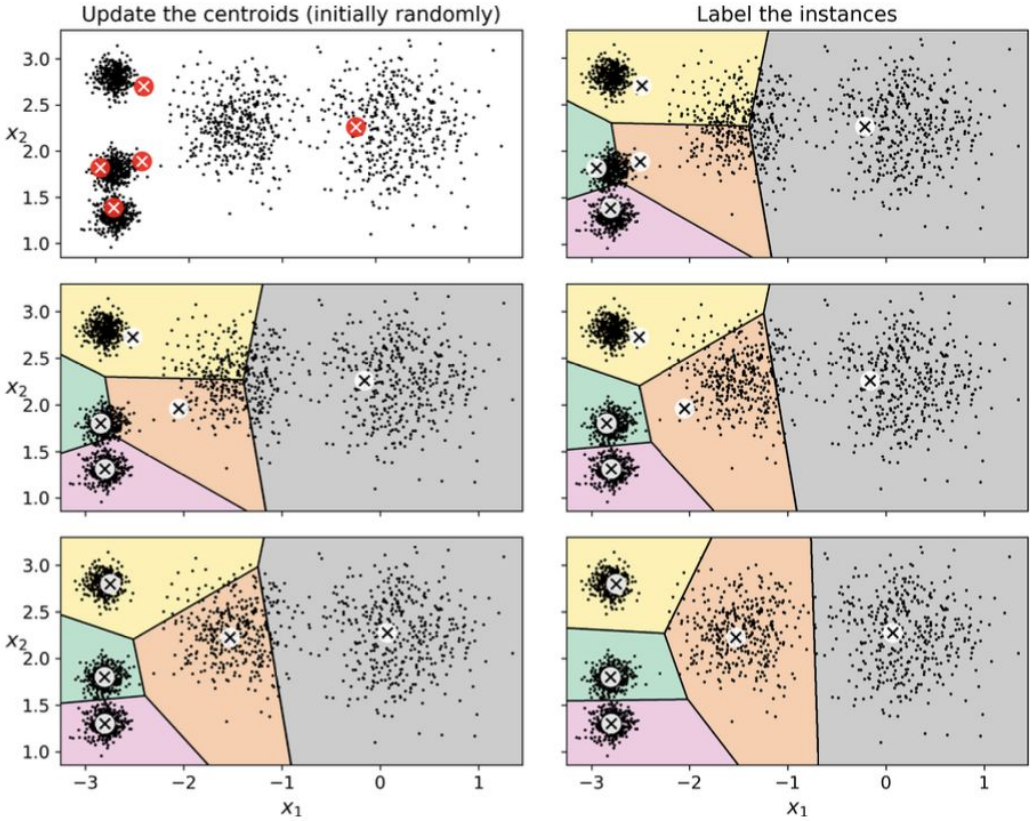


Figure 9-3. K-Means decision boundaries (Voronoi tessellation)

K-means


Update the centroids (initially randomly)
Label the instances
x_2
x_1
3.0
2.5
2.0
1.5
1.0
−3
−2
−1
0
1

K-Means Clustering Demo

https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

K-means

The algorithm is guaranteed to converge in a finite number of steps (usually quite small), it will not oscillate forever.

Unfortunately, although the algorithm is guaranteed to converge, it may not converge to the right solution (i.e., it may converge to a local optimum): this depends on the centroid initialization.

A simple solution to the problem could be running the algorithm multiple times with different random initializations and keeping the best solution.

How exactly does it know which solution is the best?

The performance metric used is called inertia which is the mean squared distance between each instance and its closest centroid.

K-Means finding the optimal k

So far, we have set the number of clusters k to 5 because it was obvious by looking at the data that this is the correct number of clusters. But in general, it will not be so easy to know how to set k, and the result might be quite bad if you set it to the wrong value.

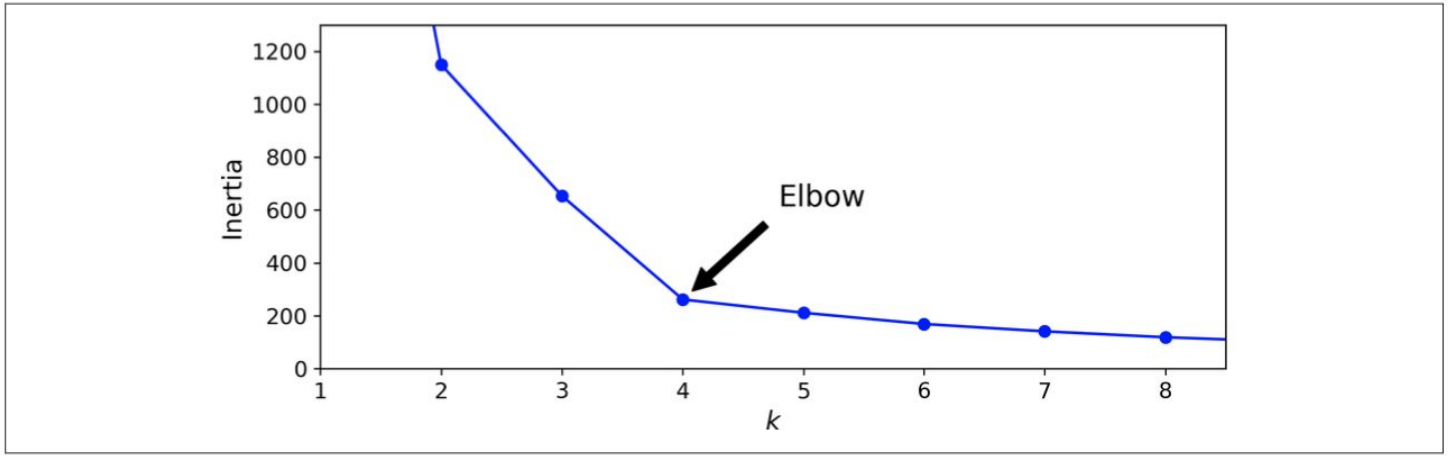


Figure 9-8. Selecting the number of clusters k using the "elbow rule"