Machine Learning

Outline:

- Introduction to machine learning
- Unsupervised learning
- Supervised learning

Co-funded by the European Union

What is Machine Learning about?

- To enable machines to *learn and adapt* without explicitly programming them
- Our only frame of reference for learning is from biology
  - ...but brains are hideously complex, the result of ages of evolution
- Like much of AI, Machine Learning mainly takes an **engineering approach**[1]
  - Remember, humanity didn't master flight by just imitating birds! ☺

1. Although there is occasional biological inspiration

Co-funded by the European Union

Theoretical Foundations

Mathematical foundations borrowing from several areas

Statistics (theories of how to **learn from data**)

Optimization (how to **solve** such learning problems)

Computer Science (efficient **algorithms** for this)

This intro lecture will focus more on **intuitions** than mathematical details

ML also **overlaps** with multiple areas of engineering, e.g.

Computer vision

Natural language processing (e.g. machine translation)

Robotics, signal processing and control theory

...but traditionally differs by focusing more on **data-driven** models and AI

Co-funded by the European Union

Why Machine Learning

Difficulty in **manually programming** agents for every possible situation

The world is ever **changing**, if an agent cannot adapt, it will fail

Many argue learning is required for Artificial **General** Intelligence (AGI)

We are still far from human-level general learning ability…

…but the algorithms we have so far have shown themselves to be useful in a wide range of applications!

Using just data, recent "deep learning" approaches can come *near* human performance on many problems, but *near* may not always be sufficient

Co-funded by the European Union

When Is Machine Learning Useful Today?

- While **not as data-efficient** as human learning, once an AI is “good enough”, it can be cheaply duplicated
- Computers work **24/7** and you can usually **scale** throughput by piling on more of them

Software Agents (Apps and web services)

- Companies collect ever more data and processing power is cheap (“*Big data*”)
- Can **let an AI learn how to improve business**, e.g. smarter product recommendations, search engine results, ad serving, to decision support
- Can **sell services that traditionally required human work**, e.g. translation, image categorization, mail filtering, content generation…?

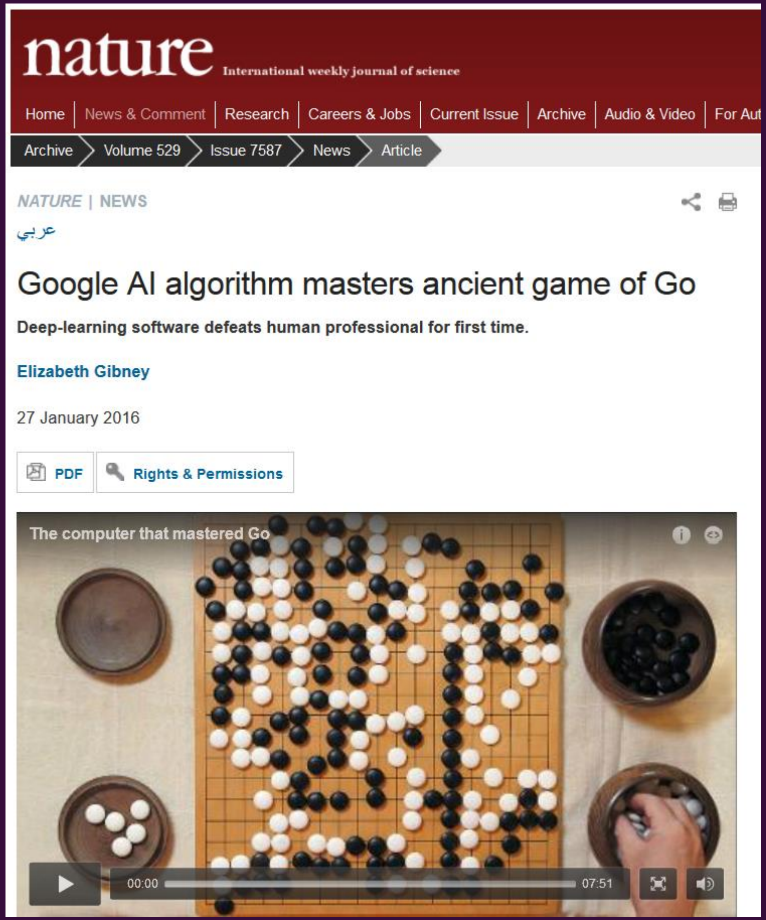
Hardware Agents (Robotics)

- Although data is more extensive, many capabilities that humans take for granted like **locomotion, grasping, recognizing objects**, **speech** have turned out to be **difficult to manually construct rules** for.

Co-funded by
the European Union

Example – Google Deepmind’s Go Agent

However, in **narrow applications** machine learning can rival or beat human performance.


nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Aut

Archive > Volume 529 > Issue 7587 > News > Article

NATURE | NEWS

عربي

Google AI algorithm masters ancient game of Go

Deep-learning software defeats human professional for first time.

Elizabeth Gibney

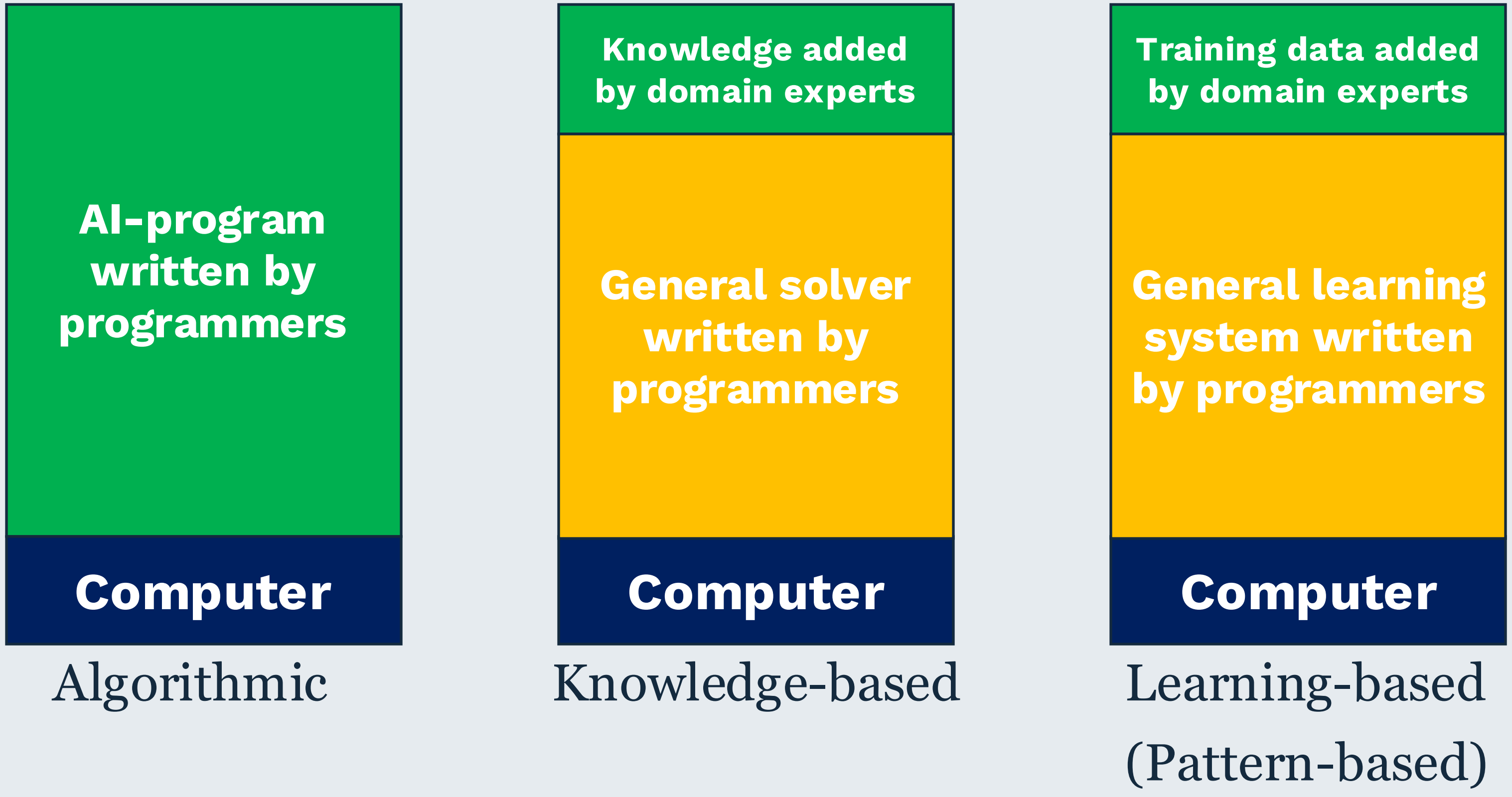
27 January 2016

PDF | Rights & Permissions

The computer that mastered Go

00:00 07:51

Algorithmic, Knowledge-Based and Learning-Based AI

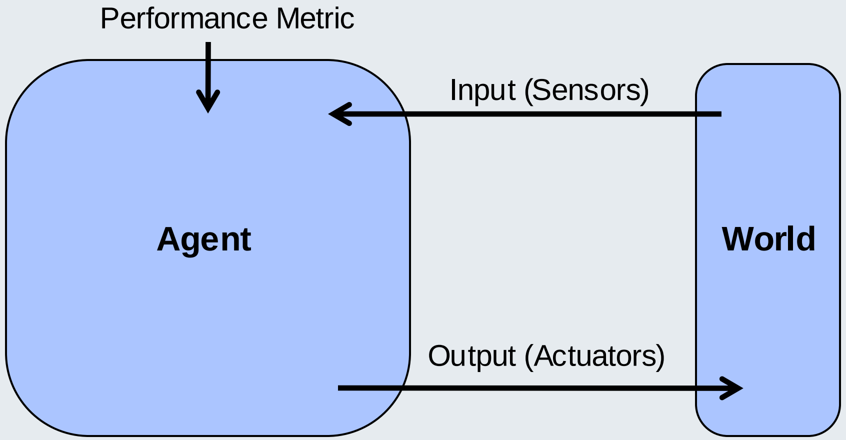


Co-funded by the European Union

To Define Machine Learning

Given a task, mathematically encoded via some performance metric, a machine can improve its performance by learning from experience (data)

From the agent perspective:

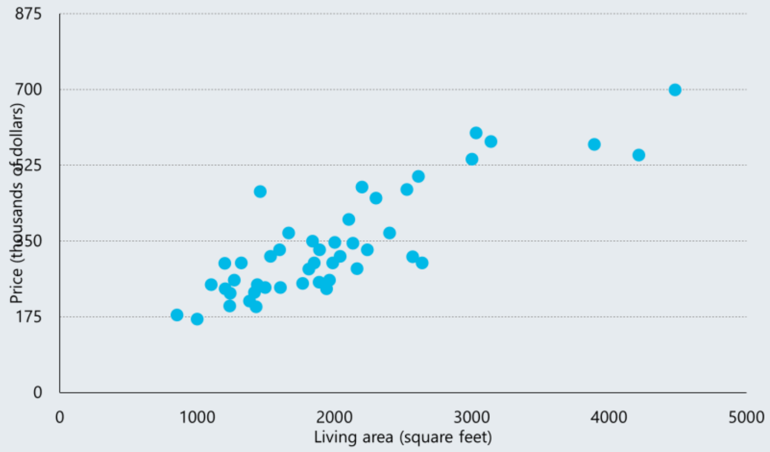


Co-funded by the European Union

To Define Machine Learning

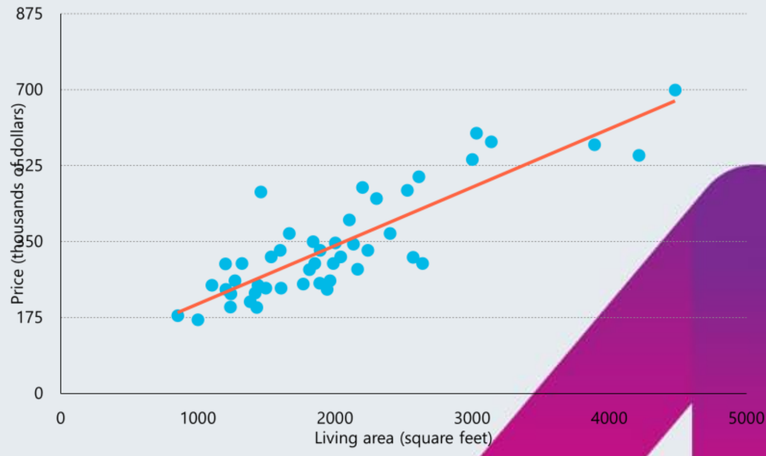
- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
- Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
    - Experience E is Watching you label emails as spam or not spam.
    - Task T is Classifying emails as spam or not spam.
    - Performance P is The number (or fraction) of emails correctly classified as spam/not spam.

Machine Learning



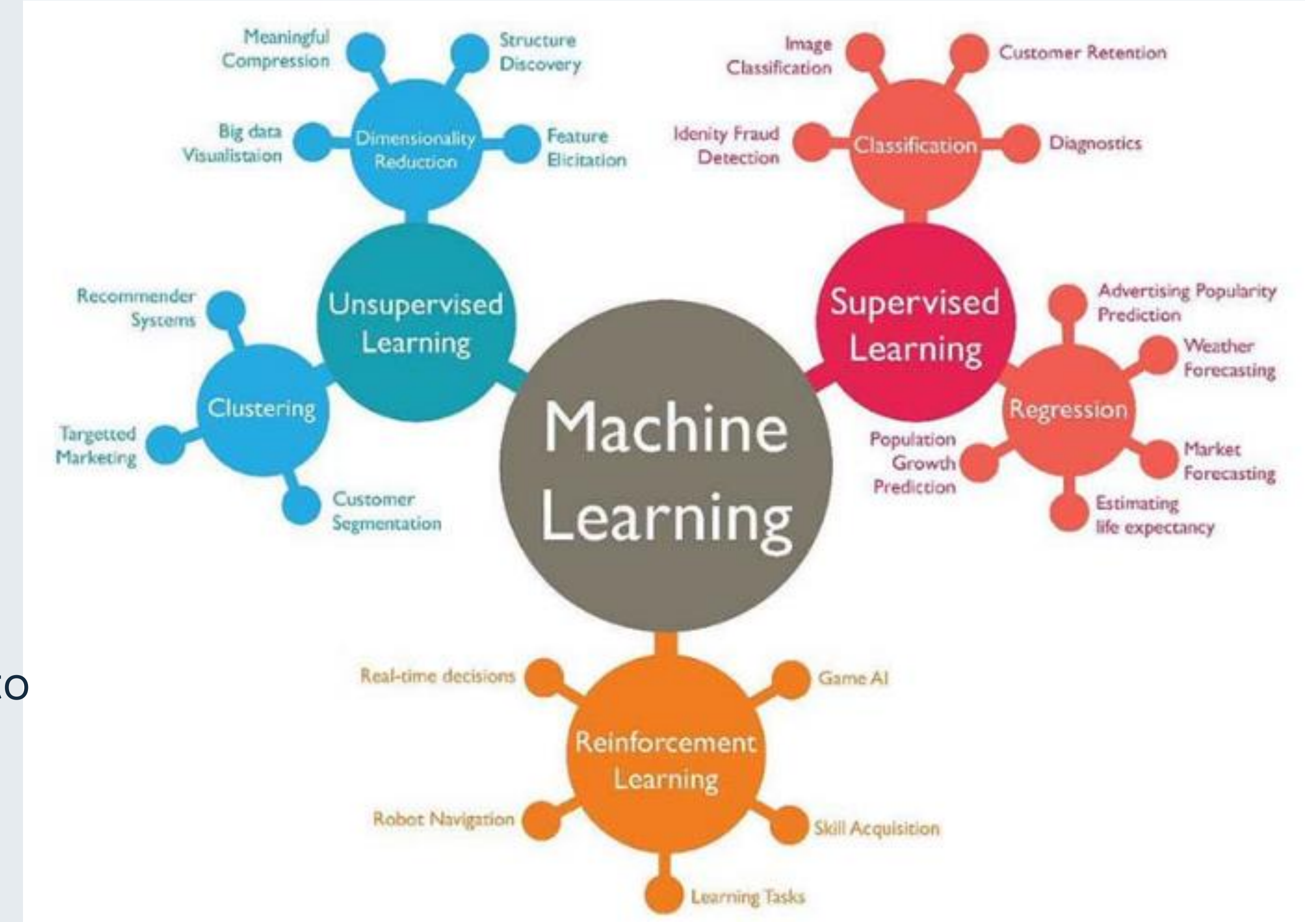
Linear regression

$$\hat{y} = \boldsymbol{x\theta}$$



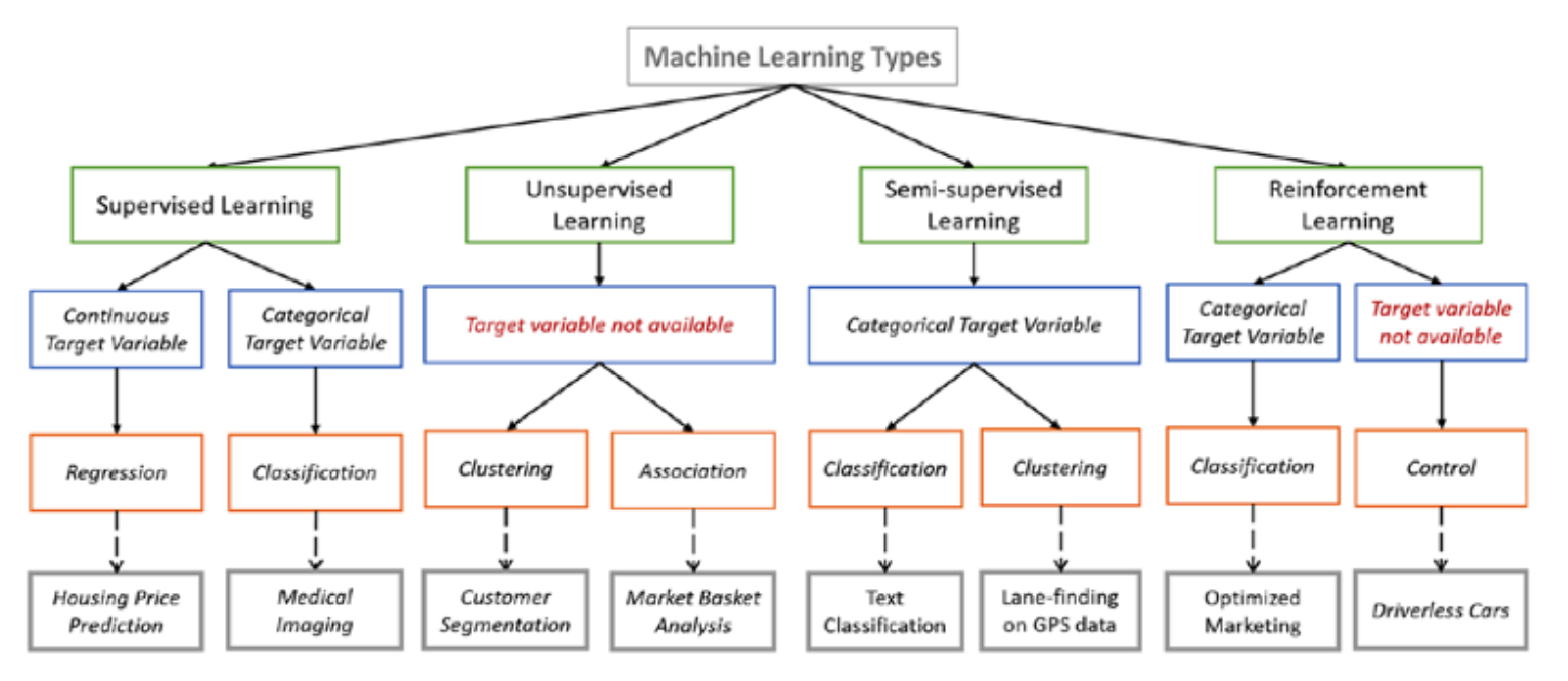
The Three Main Types of Machine Learning

- Supervised learning
  - Given input-output examples f(X)=Y, learn the function f().
- Unsupervised learning
  - Given input examples, find patterns such as clusters
- Reinforcement learning
  - Select and execute an action, get feedback, update policy (what action to do in which state).

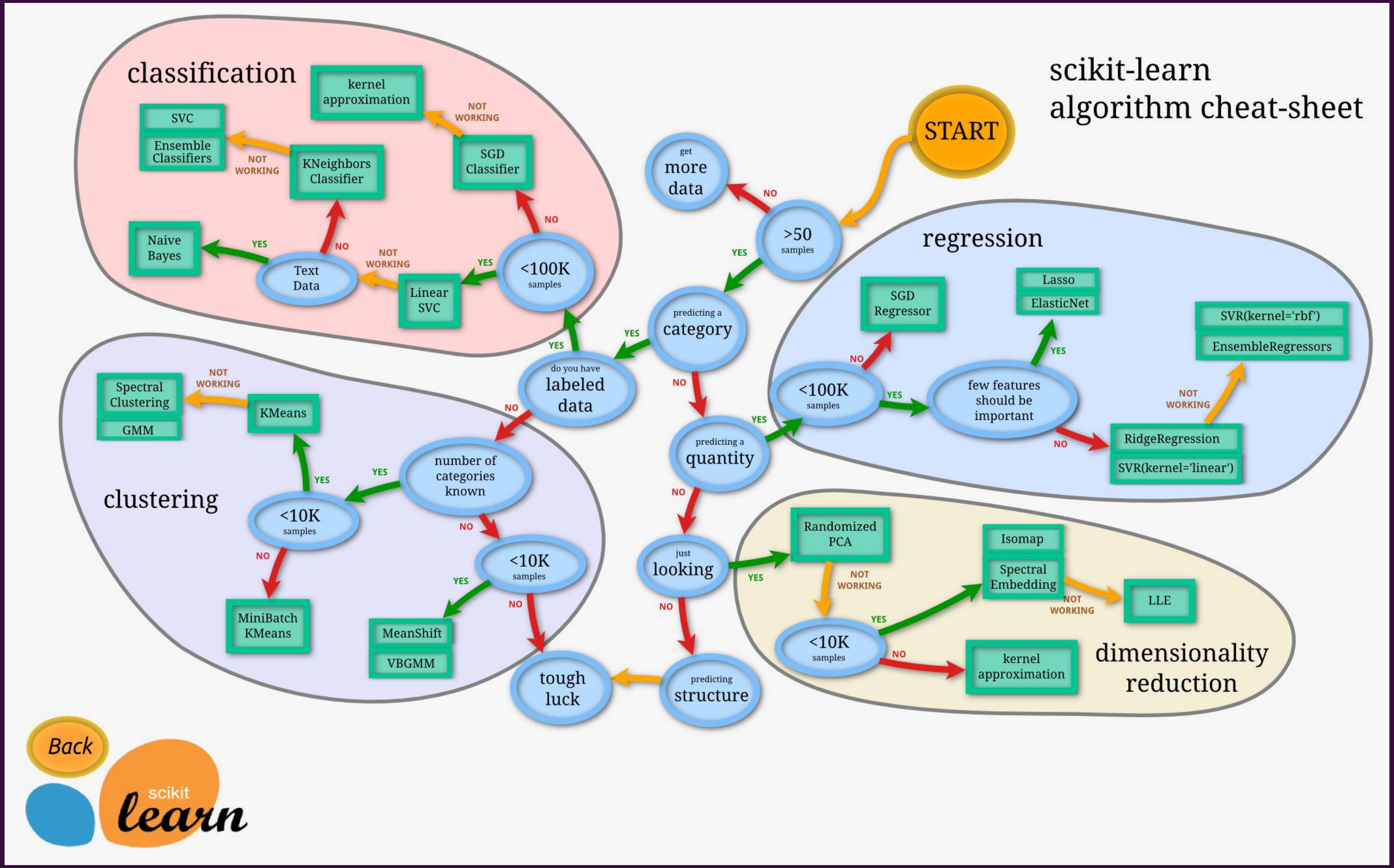


Co-funded by the European Union

https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/



Co-funded by the European Union



Digital Business
Evolving your digital future

Co-funded by the European Union

Supervised Learning at a Glance

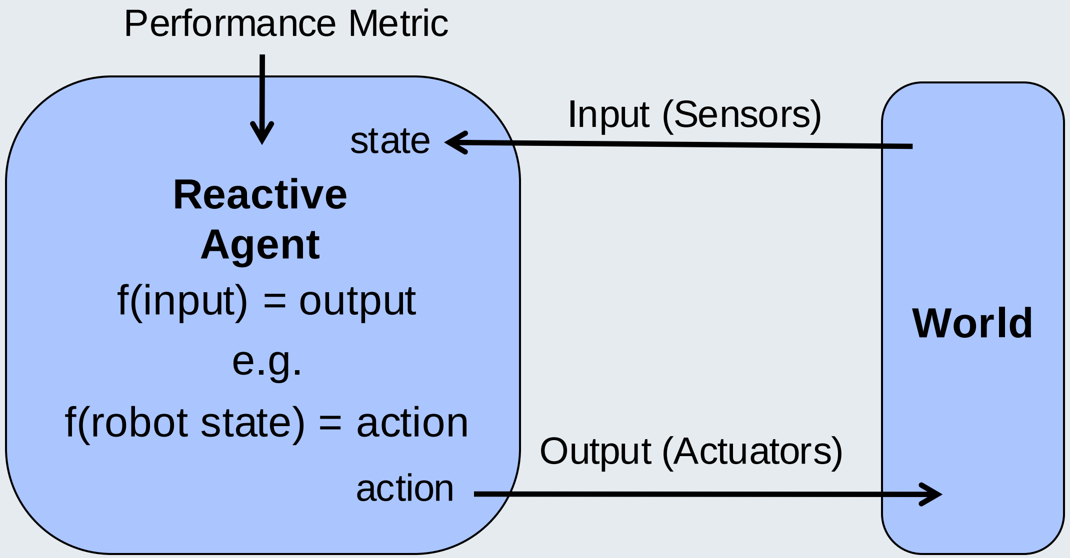
In **supervised learning**

- Agent has to learn from **examples of *correct*** behavior
- Formally, **learn an unknown function f*(x) = y*** given examples of (x, y)
- Performance metric: **Loss** (difference) between learned function and correct examples
- Typically classified into:
    - Regression: Predict continuous valued output
    - Classification: Discrete valued output

Co-funded by
the European Union

Supervised Learning – Agent Perspective

Representation from agent perspective:



...but it can also be used as a component in other architectures

Co-funded by the European Union

Reinforcement Learning at a Glance

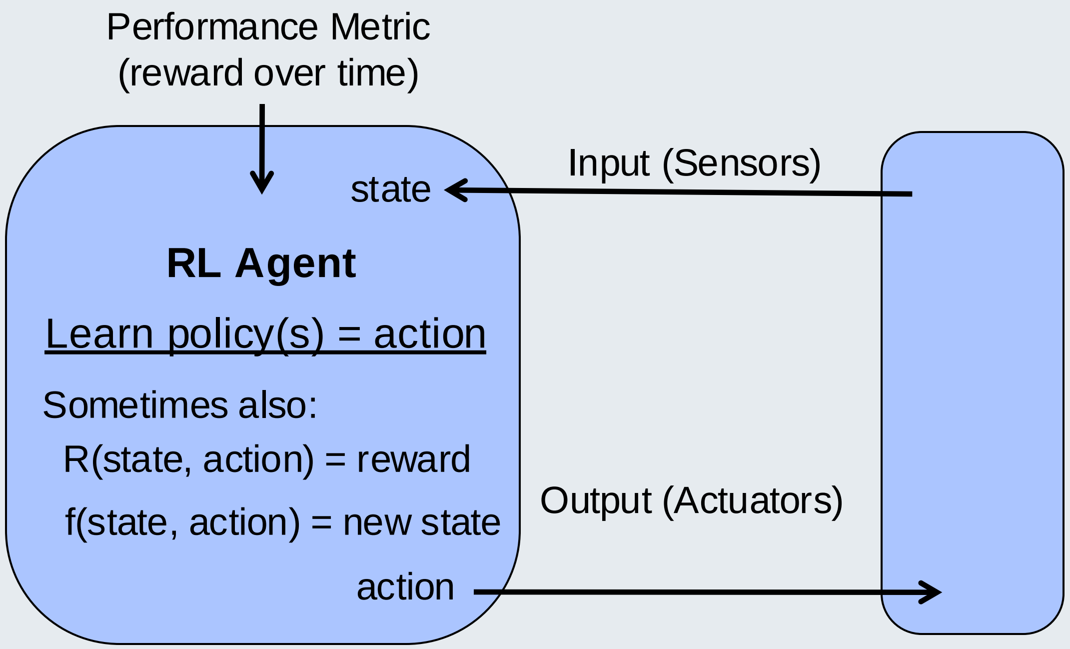
In **reinforcement learning**

- World may have **state** (e.g. position in maze) and be **unknown** (how does an action change the state)
- In each step the agent is only given **current state** and **reward** instead of examples of correct behavior
- Performance metric is sum of **rewards over time**
- Combines learning with a **planning** problem
  - Agent has to **plan a sequence of actions** for good performance
- The agent can **even learn on its own** if the reward signal can be mathematically defined

Reinforcement Learning at a Glance II

RL is based on a utility (reward) maximizing agent framework

- Agent learns *policy* (plan function) to maximize reward over time
- Either learn intermediate models of the effect of actions (next state, reward) from state s, or use *model-free* approaches



Co-funded by the European Union

Real world examples – Robot Behavior, Game Playing (AlphaGo…)

Supervised vs. Reinforcement Learning for Robot Behavior

- Learning to flip pancakes, ”supervised” and reinforcement learning (reward not shown).

Unsupervised Learning at a Glance

In **unsupervised learning**

- Neither a correct answer/output, nor a reward is given
- Task is to **find some structure** in the data
- Performance metric is some **reconstruction** error of patterns compared to the input data distribution

Examples:

- **Clustering** – When the data distribution is confined to lie in a small number of “clusters” we can find these and use them instead of the original representation, e.g. bigger recommender system (news, ads, etc.)
- **Dimensionality Reduction** – Finding a suitable lower dimensional representation while preserving as much information as possible, e.g. image/video compression

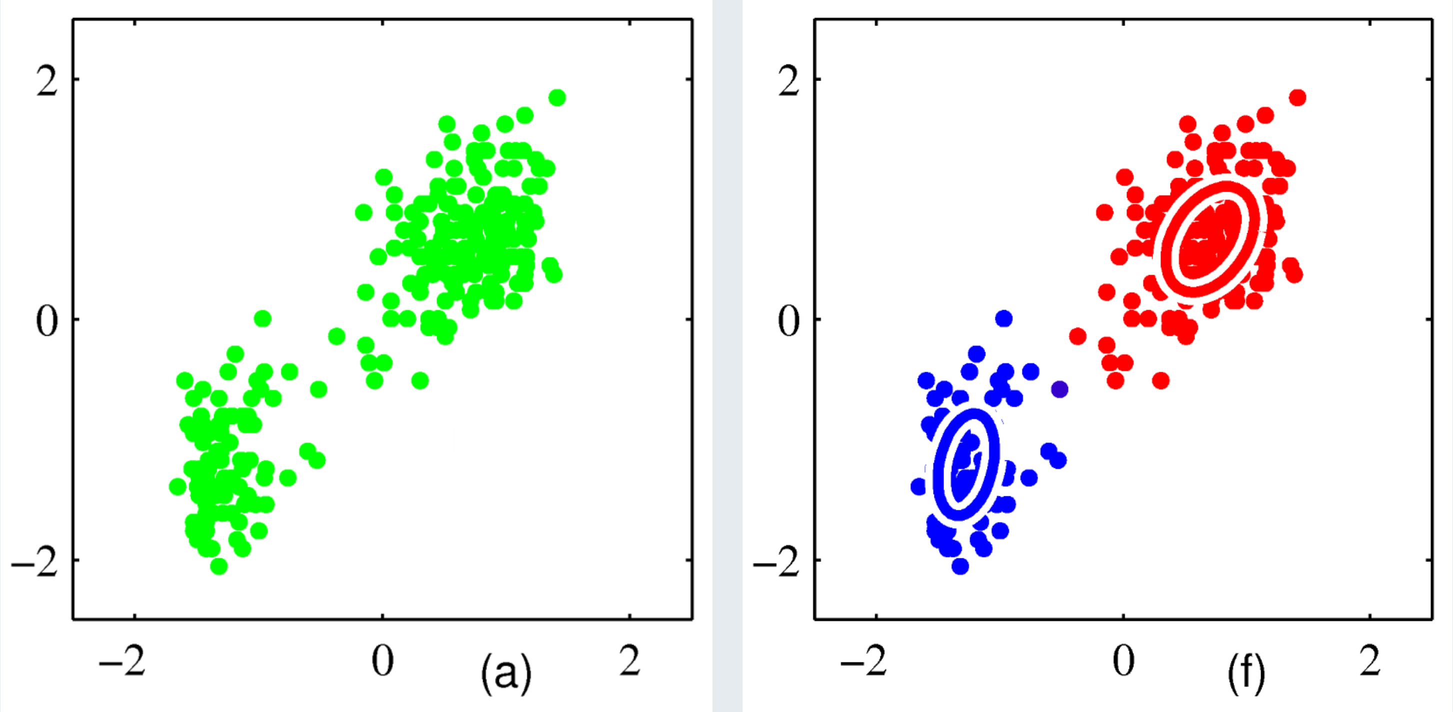
Recent trend: Found structure can be used to **generate new data (content)!**

Co-funded by
the European Union

Unsupervised Learning at a glance II

- Not directly applicable to the agent perspective as there is no clear way to encode a goal or behavior
- However, the techniques can be useful as a **preprocessing step** in other learning approaches
  - If fewer dimensions or a few clusters can accurately describe the data, big **computational wins** can be made
- It is also frequently used for **visualization** as smaller representations are easier to visualize on a computer screen
- To keep this brief, we will not go into any further detail on unsupervised learning

Unsupervised Learning Example: Clustering – Continuous Data



Two-dimensional continuous input

(Bishop, 2006)

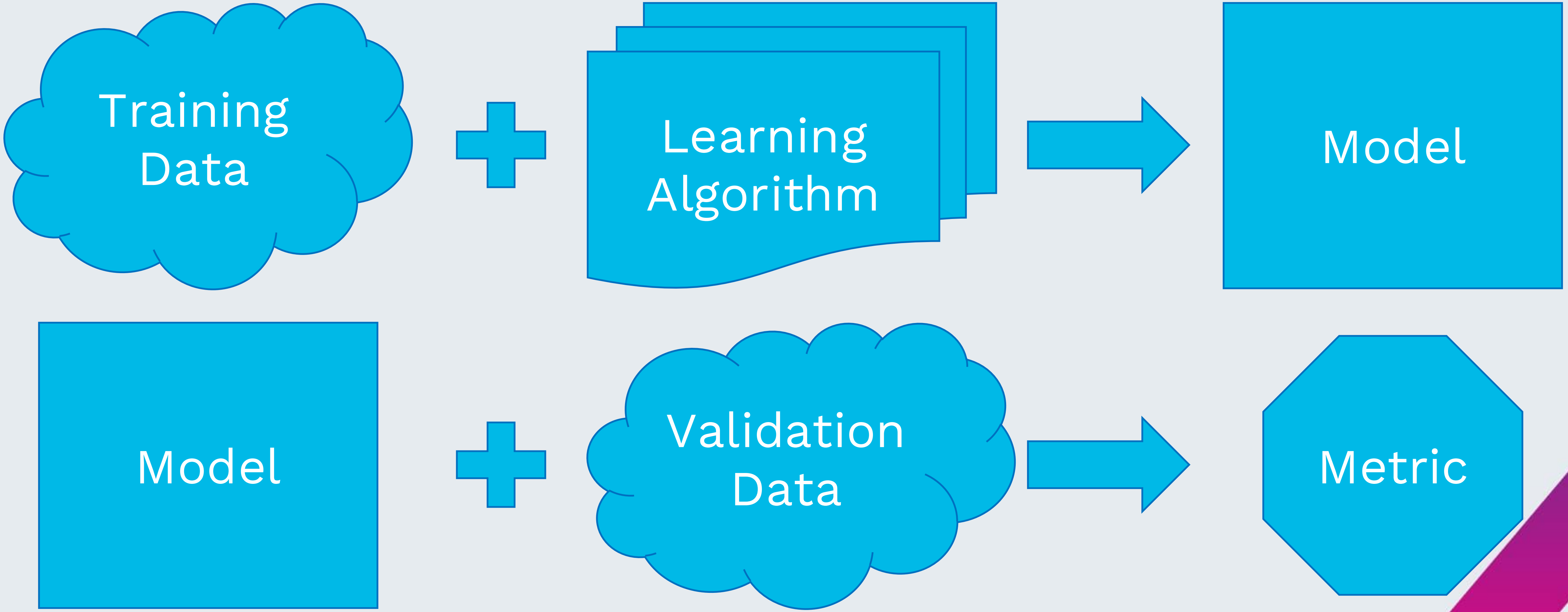
Co-funded by the European Union

Unsupervised example

- Original faces were down sampled to save space but still remain majority features.

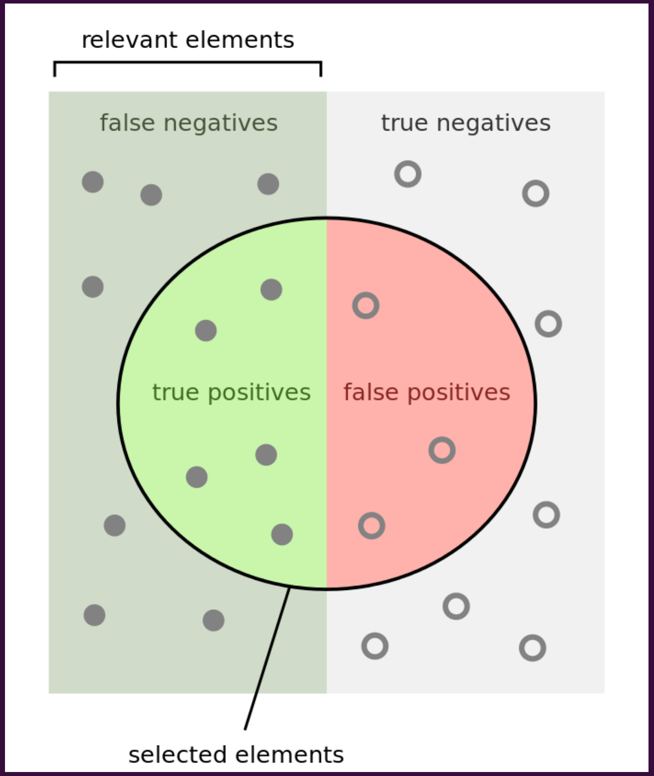
Co-funded by the European Union

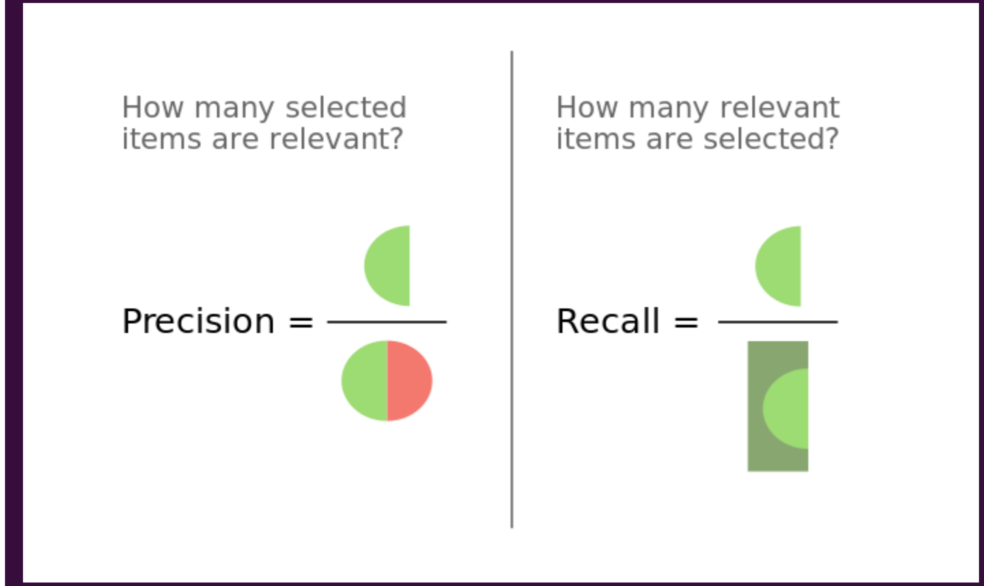
Training, Validation, and Test Data



Co-funded by the European Union

Precision and Recall

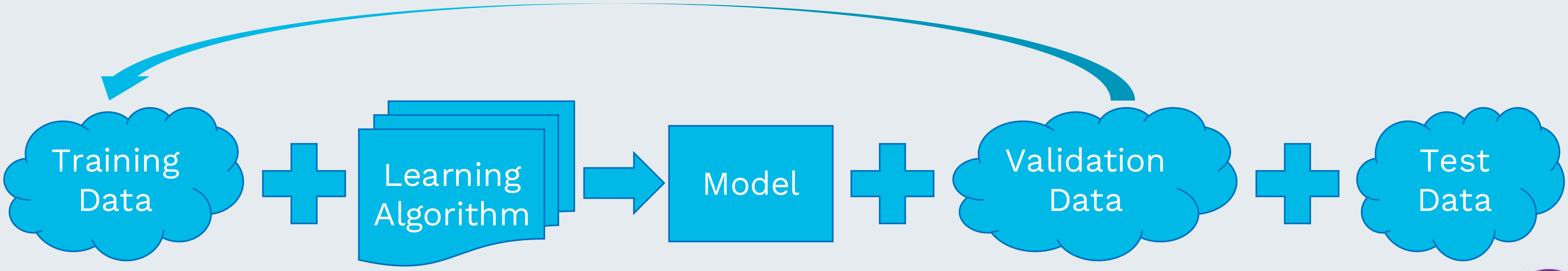




https://en.wikipedia.org/wiki/Precision_and_recall

Co-funded by the European Union

Machine Learning Process



Co-funded by the European Union

Training error and generalization error

We train a model by minimizing its error on the training data.

optimization

The training error is different from the generalization error – the expected value of the error on previously unseen inputs.

We can estimate the generalization error of a model by measuring its test error – the error on a held-out test set.

held-out = not seen during training

Co-funded by the European Union

How can we hope to perform well on the test set?

- Assumption 1: The examples in the training set and the test set are mutually independent.
- Assumption 2: The examples in the training set and the test set are identically distributed.
  - sampled from the same data generating distribution
- Under these assumptions, the expected test error is greater than or equal to the expected training error.

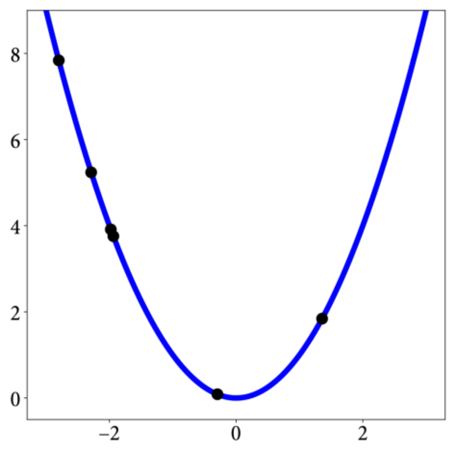
Co-funded by
the European Union

Underfitting and overfitting

- Underfitting
  - The model is unable to obtain a sufficiently low error on the training set. The model is not expressive enough.
- Overfitting
  - The gap between the training error and the test error is too large. The model is over-optimized for the training data.
  - memorises noise

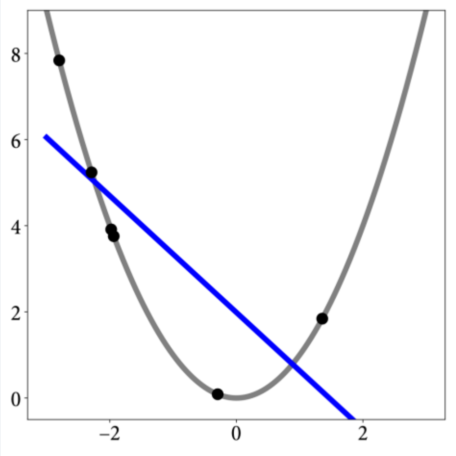
Co-funded by
the European Union

Underfitting and overfitting



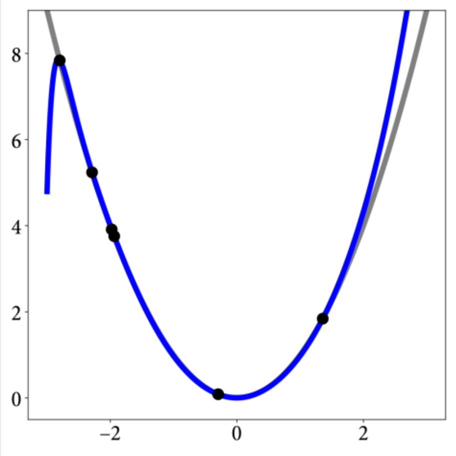
appropriate

polynomial of
degree 2



underfitting

polynomial of
degree 1

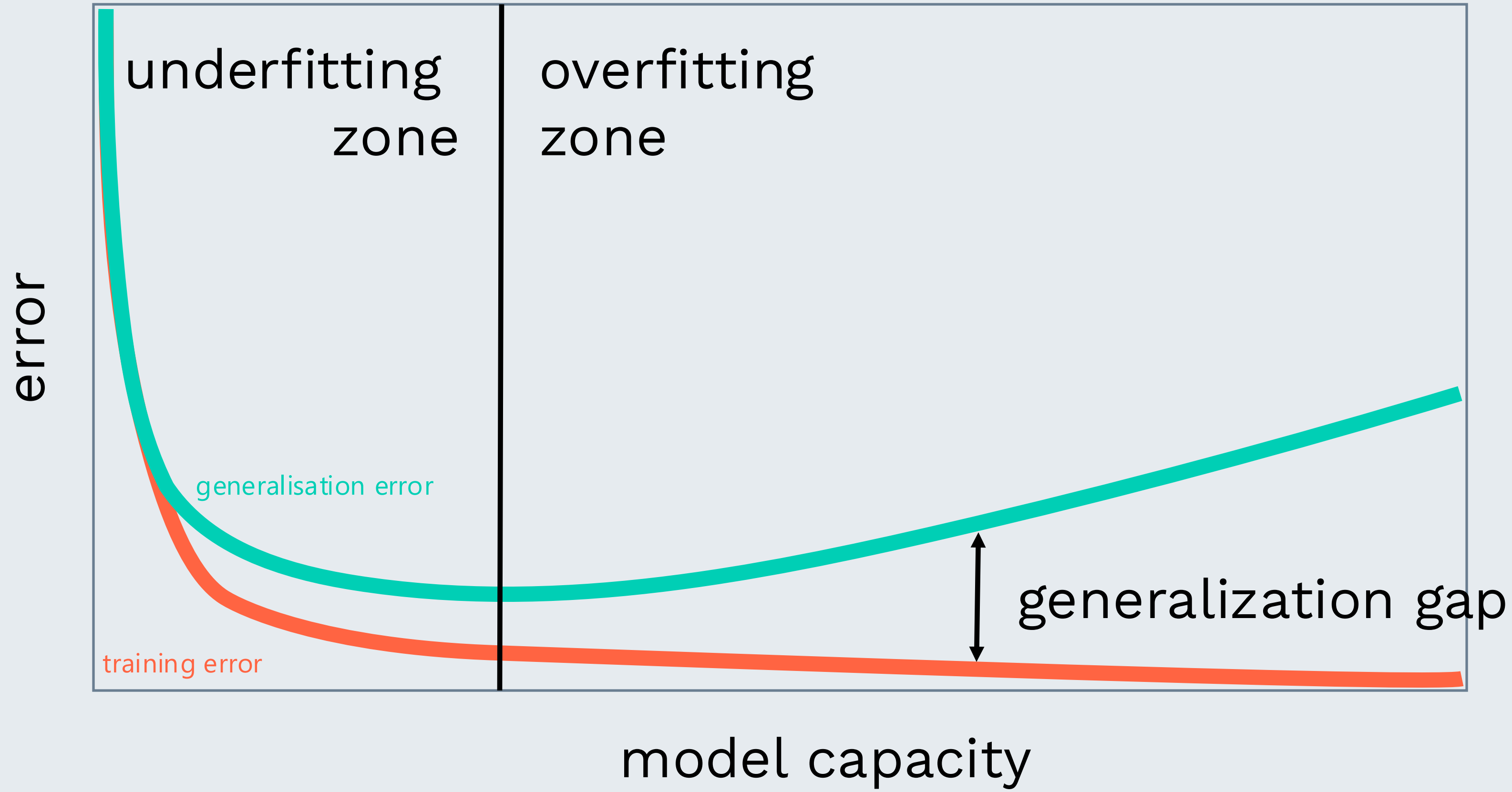


overfitting

polynomial of
degree 30

Co-funded by
the European Union

Relationship between model capacity and error



Source: GBC Figure 5.3

‘No free lunch’ theorems

Digital Business 4
Evolving your digital future

Averaged over all possible data-generating distributions, every learning algorithm has the same generalisation error.

Wolpert (1996)

This means that there is no universal learning algorithm or absolute best learning algorithm.

We need to make assumptions about the kinds of data-generating distributions we encounter in practice.

Co-funded by the European Union

Hyperparameters and validation sets

A setting of a machine learning algorithm that is not adapted by the algorithm itself is called a hyperparameter.

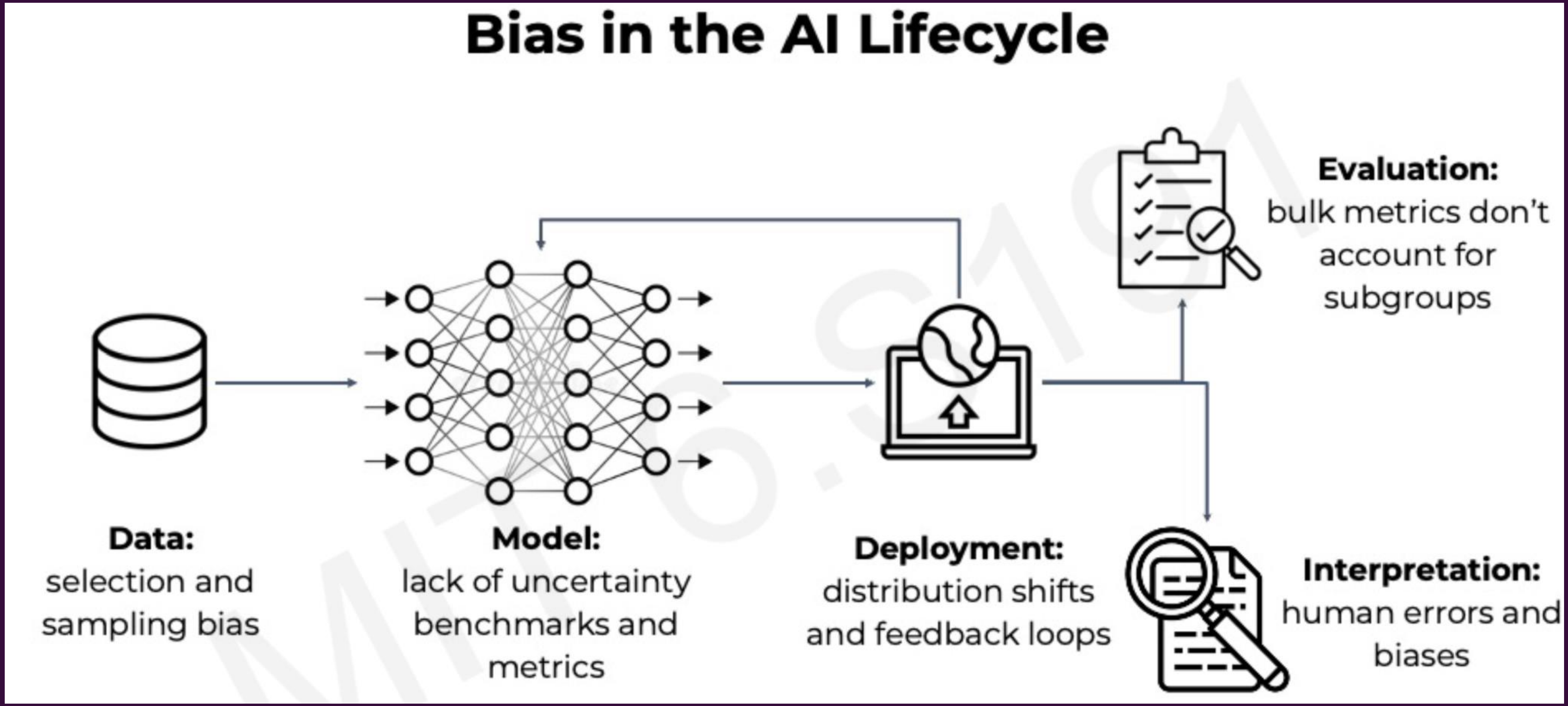
typical example: learning rate

Some settings need to be hyperparameters because adapting them during training would lead to overfitting.

such as parameters related to the model’s capacity

To tune hyperparameters, we need a separate validation set, or need to use cross-validation.

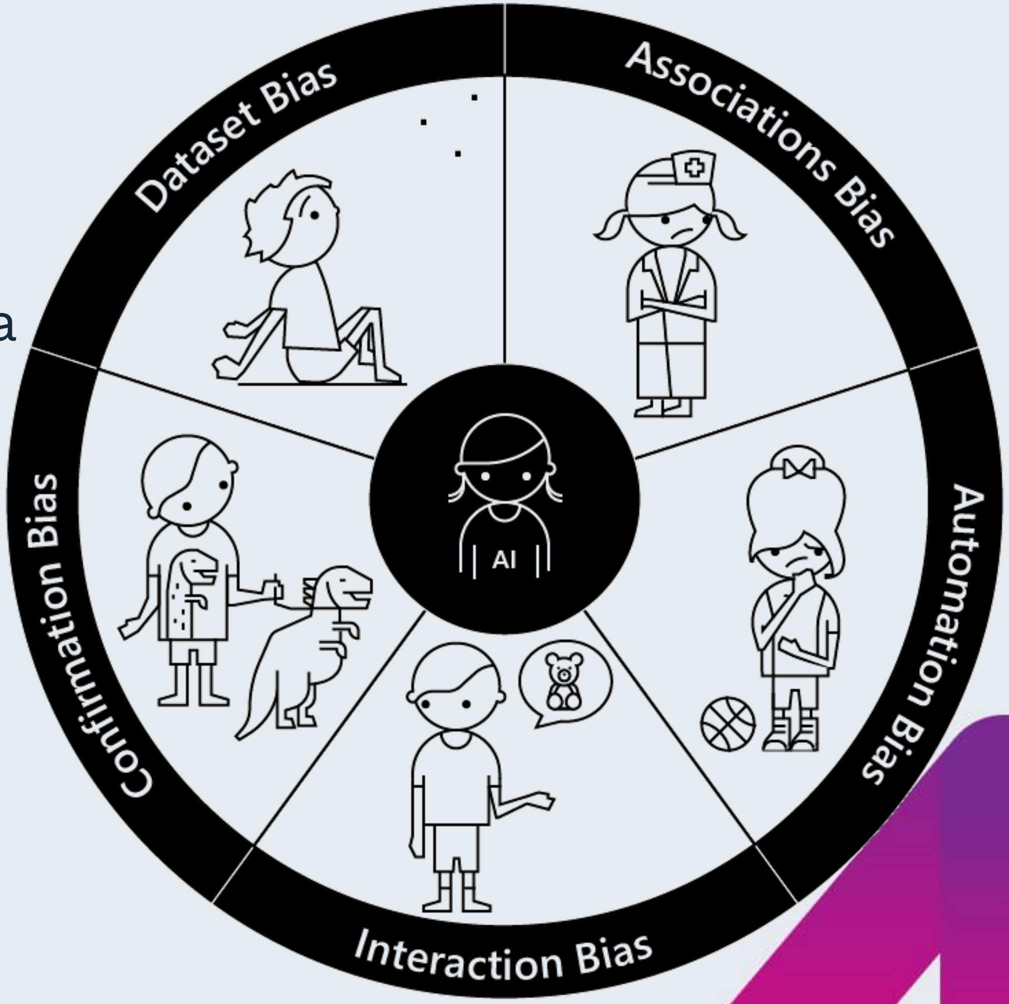
Co-funded by
the European Union



Co-funded by the European Union

Bias

- **Dataset bias** – When the data used to train machine learning models doesn't represent the diversity of the customer base.
- **Association bias** – When the data used to train a model reinforces and multiplies a cultural bias.
- **Automation bias** – When automated decisions override social and cultural considerations.
- **Interaction bias** – When humans tamper with AI and create biased results.
- **Confirmation bias** – When oversimplified personalization makes biased assumptions for a group or an individual.



Co-funded by the European Union

https://medium.com/microsoft-design/how-to-recognize-exclusion-in-ai-ec2d6d89f850

Supervised Learning

Co-funded by
the European Union

Outline of Supervised Learning

Today we will focus on **Supervised Learning**

Definition

Fundamentals: Features, Models, Loss (or cost) Functions, Training

Linear Models

Neural Networks

- Trend: Deep Learning (more in next ML lecture)

Pitfalls and Limitations

Co-funded by the European Union

Formalizing Supervised Learning

Remember, in Supervised Learning:

- Given tuples of ***training data*** consisting of ($\mathbf{x}$,y) pairs
- The objective is to learn to **predict** the **output** y' for a new input **x'**

Formalized as **searching** for approximation to **unknown function** y = ***f(x)***, given N examples of **x** and y: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$

Two major classes of supervised learning

- **Classification** – Output are **discrete** category labels

Example: Detecting disease, y = "healthy" or "ill"

- **Regression** – Output are **numeric** values

Example: Predicting temperature, y = 15.3 degrees

In either case, input data $\mathbf{x}_i$ could be **vector valued** and **discrete**, **continuous** or **mixed**. Example: $\mathbf{x}_1$ = (12.5, "cat", true).

Co-funded by the European Union

Digital 4 Business
Evolving your digital future

Classical Supervised Learning in Practice

Can be seen as **searching** for an approximation to unknown function y = ***f(x)*** given N examples of **x** and y: $(\mathbf{x}_1, y_1)$, … ,$(\mathbf{x}_n, y_n)$

Want the algorithm to **generalize** from **training** examples to new inputs **x’,** so that y’=f(**x’**) is “close” to the correct answer

1. An **input “feature” vector** $\mathbf{x}_i$ of examples is constructed by **mathematically encoding** relevant problem data
   - Examples of such $(\mathbf{x}_i, y_i)$ make up the **training set**
2. A *model (or hypothesis)* for *f(x)* is selected with some parameters
3. A *loss* function is selected that defines “closeness” to correct answers
4. The model is **trained** on the examples by searching for its **parameters** that minimize loss on the training set (i.e. are “close” to unknown *f(x)*)

Co-funded by the European Union

Feature Vector Construction

Want to learn ***f(x)* =** *y* given N examples of **x** and y: ($\mathbf{x}_1$,y_1), … ,($\mathbf{x}_n$,y_n)

Most standard algorithms work on **real number variables**

- If inputs ***x*** or outputs y contain categorical values like "book" or "car", we need to encode them with numbers
  - With only two classes we get y in {0,1}, called **binary classification**
  - Classification into multiple classes can be reduced to a sequence of binary one-vs-all classifiers
- The variables may also be structured as **text**, **audio**, **image** or **video** data

Finding a suitable feature representation **can be non-trivial**, but there are standard approaches for the common domains

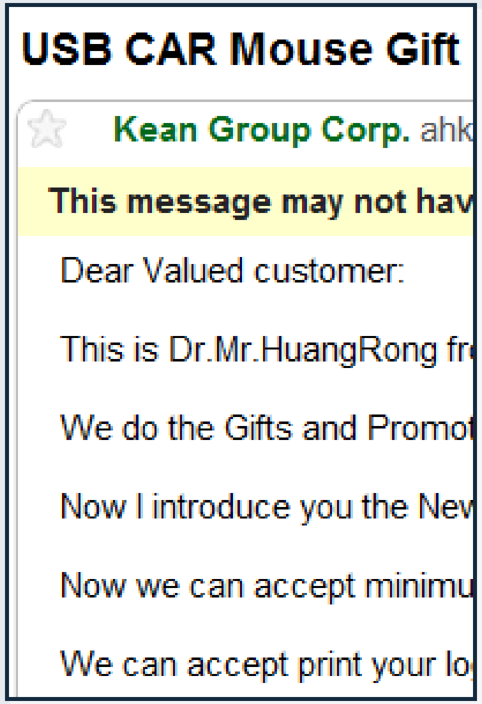
- With sufficient data, features can also be learned (*deep learning*, later…)

Co-funded by the European Union

Feature Vector Example for Text - Bag of Words

One of the early successes of ML was **learning spam filters**

Spam classification example:



Each mail is an input, some mails are flagged as spam or not spam to create training examples.

Bag of Words Feature Vector:

Encode the existence of a fixed set of relevant **key words** in each mail as the **feature vector.**

$\mathbf{x_i}$ = words$_i$ **=**

Feature	Exists?
"Customer"	1 (Yes)
"Dollar"	0 (No)
"Fund"	0
"Accept"	1
"Bank"	0
….	…

y_i = 1 (spam) or 0 (not spam)

Simply learn f(x)=y using suitable classifier!

Co-funded by the European Union

Selecting Models: Linear **Regression Example**

I. Construct a **feature vector $\mathbf{x_i}$** to be used with examples of y_i

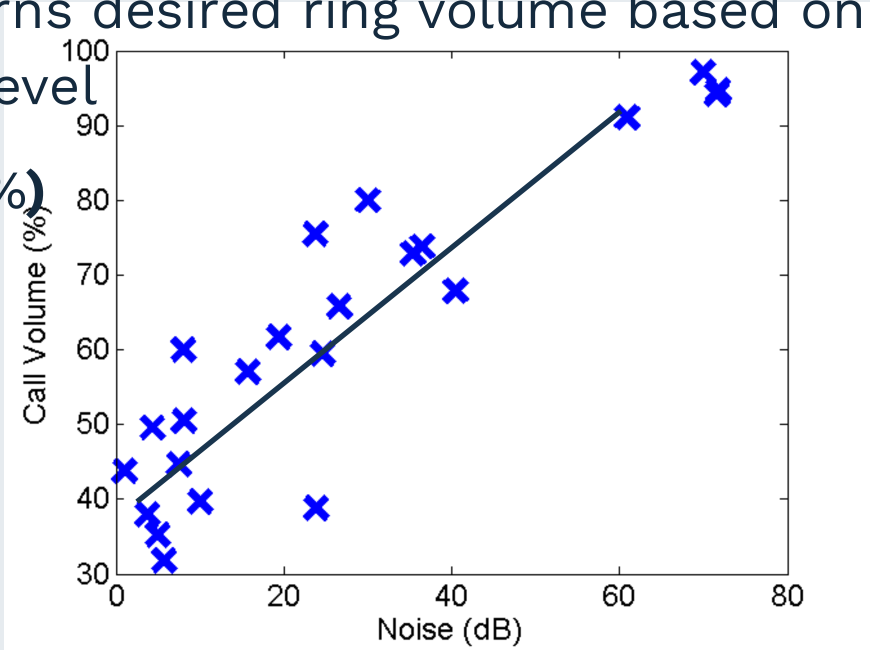
II. Select a model and **train** it on examples (search for a good approximation to the unknown function)

Fictional example: Smartphone app that learns desired ring volume based on examples of volume and background noise level

Feature vector $\mathbf{x_i}$ **=** (Noise dB), y_i = (Volume %)

- Select the familiy of **linear** functions: $y_i = w_1 \cdot x_i + w_0$
- **Train** the algorithm by searching for a line that **fits the data well**

…but how does "training" really work?



Co-funded by the European Union

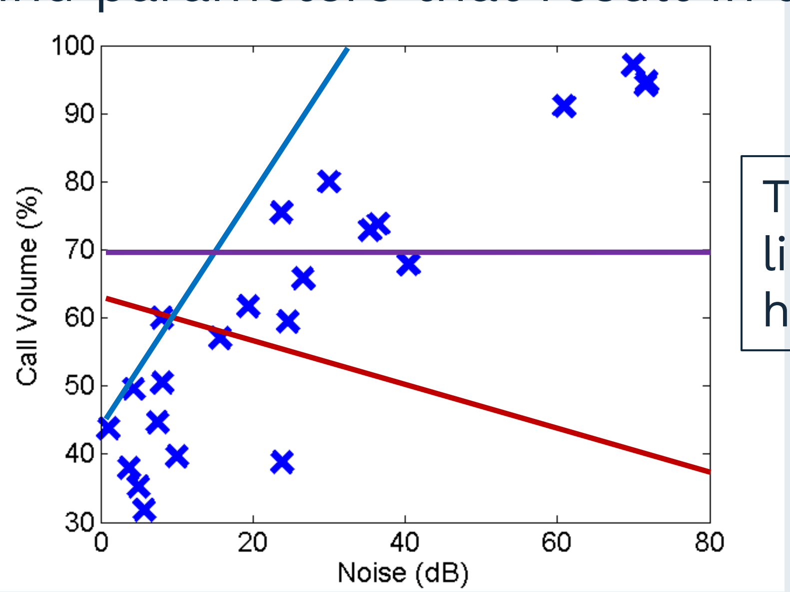
Training a Learning Algorithm

Feature vector $\mathbf{x_i}$ **=** (Noise in dB), outputs y_i = (Volume %)

- Recap: Want to find approximation *h*(x) to the unknown function f(x)
- As an example, let it to be the family of **linear** functions:

$$y_i = w_1 \cdot x_i + w_0$$

- The model $h_{\mathbf{w}}(x)$ has two **parameters:** $\mathbf{w} = (w_1, w_0)$ (line slope and offset)
- How do we find parameters that result in a **good** approximation h?



Three poor linear hypotheses

Co-funded by the European Union

Training a Learning Algorithm – Loss Functions

How do we find parameters **w** that result in a **good** approximation $h_{\mathbf{w}}(x)$?

- Need a performance metric for function approximations of unknown *f(x)*
  - **Loss functions** $L(f(x), h_{\mathbf{w}}(x))$
- Minimize deviation against the N example data points from *f(x)*
  - For **regression** one common choice is a **sum square loss** function:

$$L(f(x), h_{\mathbf{w}}(x)) = (f(x) - h_{\mathbf{w}}(x))^2 = \sum_{i=1}^{N} (y_i - h_{\mathbf{w}}(x_i))^2$$

- Why square loss? Negative difference is as bad as positive
- Search in continuous domains like **w** is known as **optimization**
  - (if unfamiliar, see Ch4.2 Local Search in Continuous Spaces in course book AI: A Modern Approach)

Co-funded by the European Union

Training a Learning Algorithm – Optimization

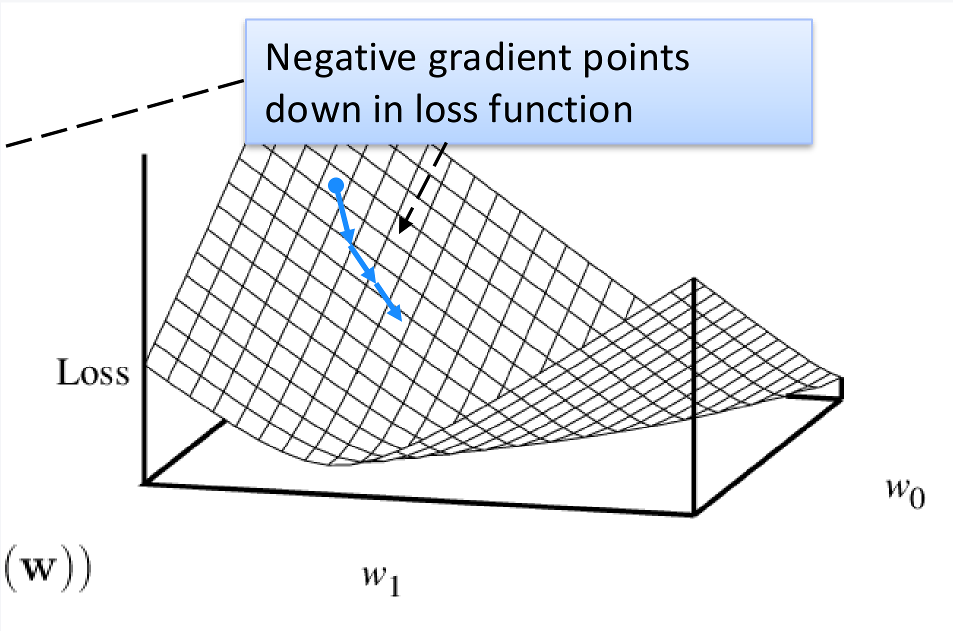
How do we find parameters **w** that minimize the loss?

- Optimization approaches iteratively move in the **direction that decreases** the loss function *L(w)*
- Simple and popular approach: **gradient descent**

w_j

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} L(\mathbf{w})$$

Step size (learning rate)



$$Gradient = \left(\frac{\partial}{\partial w_0} L(\mathbf{w}), \frac{\partial}{\partial w_1} L(\mathbf{w})\right)$$

Co-funded by the European Union

What about categorical outputs (classification)?

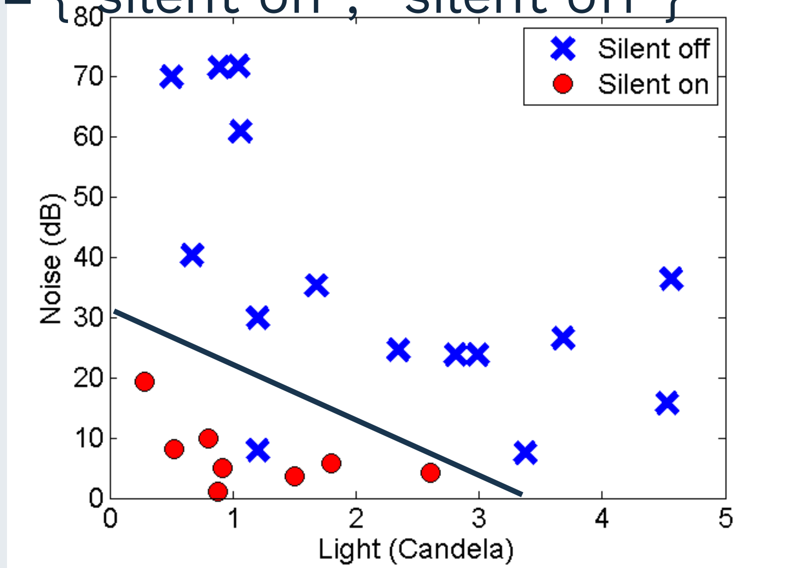
I. Construct a **feature vector x_i** to be used with examples of y_i

II. Select a model and **train** it on examples (search for a good approximation to the unknown function)

Fictional example: Smartphone app that learns if silent mode should be on/off at different levels of **background noise** and **light**

Feature vector **x_i = (Noise, Light level)**, y_i = {“silent on”, “silent off”}

- Again, can select the familiy of **linear** functions. However, now outputs y have to be **transformed** to the interval [0,1]
- Can classify new inputs according to how close output is to 0 or 1.
- For linear models, the decision boundary will still be a straight line.



Co-funded by the European Union

Classifier Training – Loss Functions II

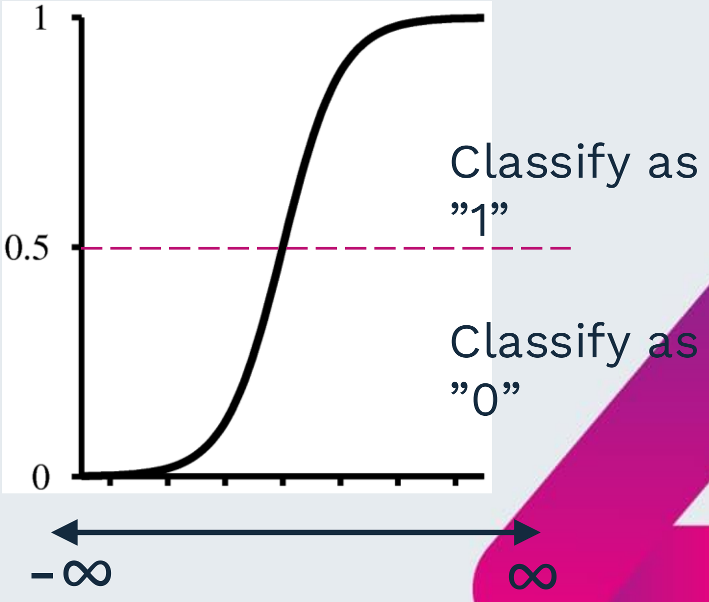
- How to transform standard models to classification?
  - Squared error does not make sense when target output discrete set {0,1}
- Could use custom loss functions for classification
  - Minimize number of missclassifications (unsmooth w.r.t. parameter changes)
  - Maximize information gain (used in decision trees, see book)
  - **However**, requires specialized parameter search methods
- Instead: Make outputs **probabilities [0,1]** by **squashing** predicted **numeric outputs** via sigmoid (”S”)

Sigmoid functions allow us to do use **any** regression model with binary classification by def. Pr(y=”1”|X) = g(model(x))

Where g is **”logistic” sigmoid** :

$$g(x) = \frac{1}{1+e^{-x}}$$

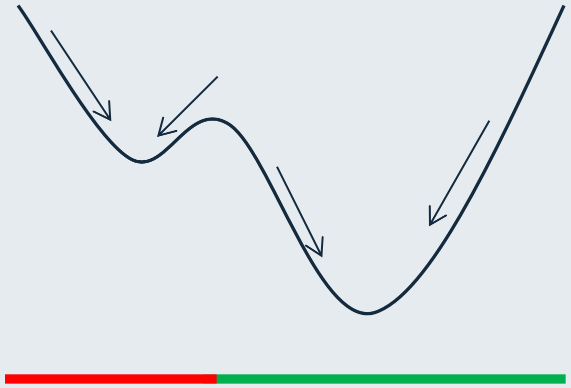
For >2 classes, use **soft-max** (see book)



Training a Learning Algorithm – Limitations

Limitations

- Local optimization of loss is greedy – Gets stuck in *local* minima unless the loss function is **convex** w.r.t. **w**, i.e. there is **only one minima**.
- **Linear models are convex**, however most more advanced models are **vulnerable** to getting stuck in local minima.
- Care should be taken when training such models by using for example **random restarts** and picking the least bad minima.



If we happen to start in red area, optimization will get stuck in a bad local minima!

Co-funded by the European Union

Linear Models in Summary

Advantages

- Linear algorithms are **simple and computationally efficient**
  - For both regression and classification
- Training them is a **convex** optimization problem, i.e. one is guaranteed to find the **best** hypothesis in the space of linear hypothesis
- Can be extended by non-linear feature transformations

Disadvantages

- The hypothesis space is very restricted, it cannot handle non-linear relations well

Still **widely used** in applications

- Recommender Systems – Initial Netflix Cinematch was a linear regression, before their **$1 million** competition to improve it. Rather simple and are appropriate for small systems.
- Often a good place to start…
- At the core of many big internet services. Ad systems at Twitter, Facebook, Google etc…

Co-funded by the European Union

What about models with uncertainty?

Supervised Learning:

Mathematically, can be seen as finding an approximation to an unknown function y = ***f(x)*** given N examples of **x** and y

Two perspectives:

- **Deterministic Models**
  - Search for a suitable function y = h(x)
  - What we have looked at so far, the most common approach
  - Example: In classification something may be either A or B, never in-between, regression gives an exact answer like 15.3
- **Probabilistic Models**
  - Search for a suitable ***probability distribution*** like P(Y|X)
  - When we also want to predict the **uncertainty**
  - Example: P(Y="Healthy"|X) = 0.7 and P(Y="Cancer"|X) = 0.3
  - In a spam filter we might prefer to get a spam too many than to trash that important mail from your boss...

Co-funded by the European Union

Model Selection – Choosing Between Models

Digital 4 Business
Evolving your digital future

- In conclusion, we want to avoid **unnecessarily complex** models
- This is a fairly general concept throughout science and is often referred to as **Ockham's Razor**:

  "*Pluralitas non est ponenda sine necessitate*"

  -Willian of Ockham

  "*Everything should be kept as simple as possible, but no simpler.*"

  -Albert Einstein (paraphrased)

- There are several mathematically principled ways to **penalize** model **complexity** during training, e.g. regularization, which we will not cover here.
- A simple approach is to use a separate **validation set** with examples that are <u>**only**</u> used for evaluating models of different complexity.

Co-funded by the European Union

Model Selection – Hold-out Validation

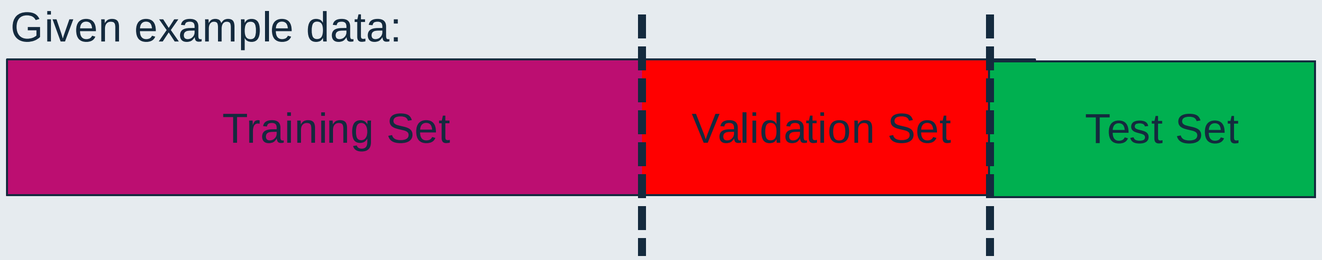
- This is called a **hold-out validation set** as we keep the data away from the training phase
- Measuring performance (loss) on such a validation set is a **better metric of actual generalization** error to unseen examples
- With the validation set we can compare models of **different complexity** to select the one which generalizes best, for model selection.
- Examples could be polynomial models of different order, the number of neurons or layers in an ANN etc.

Given example data:

Co-funded by
the European Union

Measuring Final Generalization Error

- We have seen that having a validation set will lead to a more accurate estimation of generalization error to use for model selection
- However, by **extensively** using the validation set for model selection we can also contaminate it (overfitting model against the data in the validation set)
- To combat this one usually sets aside a separate **test set**
- This test set is **not** used during training or model selection
- It is basically locked away in a safe and only brought out in the end to get a fair estimate of final generalization error



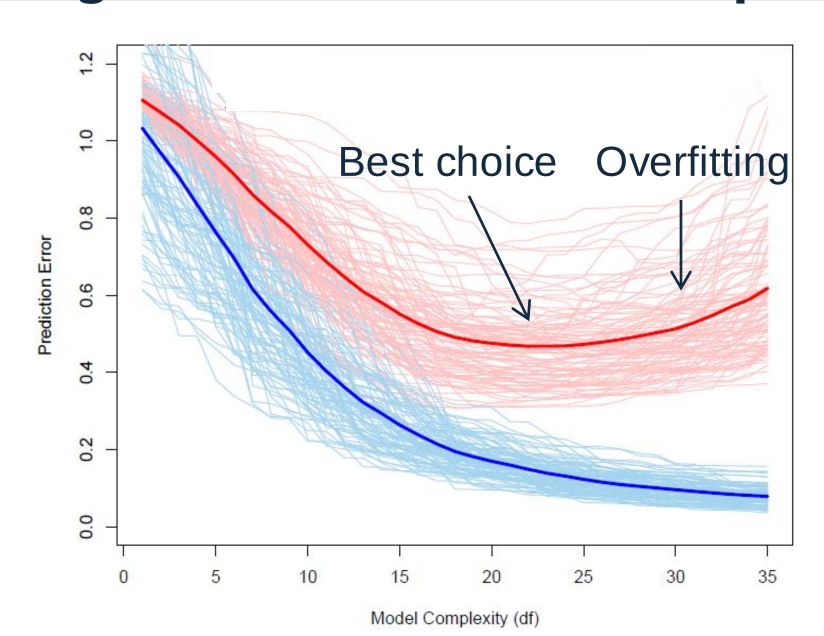
Co-funded by the European Union

Model Selection – Selection Strategy

- As the number of parameters increases, the size of the hypothesis space also increases, allowing a **better fit to training data**
- However, at some point it is **sufficiently flexible** to capture the underlying patterns. Any more will just capture noise, leading to **worse generalization to new examples!**

Example: Prediction error vs. model complexity over many (simulated) data sets. (Hastie et al., 2009)

Red: Validation set (generalization) error
Blue: Training set error



- Do we need to train and test many models of different complexity?
  - Various tricks to avoid this

Co-funded by the European Union

Early Stopping: Model Complexity Trick with Neural Networks

- Training neural networks tends to progress from simple functions to more complex ones
- This comes from initializing the parameter values **w** close to zero
  - Remember, a neuron's output = g(**w***x)
  - Common activation functions g (e.g. sigmoid) are linear around zero
  - This makes the NN effectively "start out" as a linear model
- **Early stopping** NN trick: Can make a model complexity vs. validation loss curve **while training**, stop when validation error starts increasing

Exercise: Back to the NN demo app
- Observe "test loss" plot
- Reset network
- Train again, but keep an eye on test loss
- Try to pause at low test loss
  - Can adjust "learning rate"

Stop training here!

Co-funded by the European Union