Machine Learning I

AMIT DHOMNE

Logistics 1: How this course works

- Two (interdependent!) parts to the course, two exams, two lecturers
  - Matthias Bethge: Professor at Uni/CIN/BCCN/MPI,
    Group ‘Computational Vision and Neuroscience’
  - Jakob Macke: Junior Group Leader at BCCN/MPI/CIN
    Group ‘Neural Computation and Behaviour’
- One two-hour lecture a week, followed by one hour of exercises
- Homework-assignments every week
- How much you will learn from the course depends (partially) on you
  - Work through the equations we write down in the lectures on your own, make sure you understand the derivations
  - Take the homework-assignments seriously-- we can not make you do them, but they are an important part of the course (and will help you in preparing for the exam)
  - Take notes-- we will provide a transcript of the lectures, but there might be a delay between the lecture and us updating the transcript
  - Ask questions-- during the exercise sessions, during the lectures, during the tutorials, by email.

Logistics 2: Dates

Supervised Learning

Unsupervised Learning

Weeks	Day	(Provisional) Topic	Lecturer
1	19.10.	***Introduction***	Macke
2	26.10.	***Bayesian Inference***	Macke
3	**31.10.***	***Linear Regression***	**Macke**
4	9.11.	***Gaussian Models***	Macke
5	16.11.	***Linear Classification 1***	Macke
6	23.11.	***Linear Classification 2***	Macke
7	30.11.	***Tutorial***	Macke
8	07.12.	***Written Exam***	Macke
9	14.12.	***Probability Distributions***	Bethge
11	21.12.	***Unsupervised learning***	Bethge
		Vacation	
12	11.01.	***Principal component analysis (PCA)***	Bethge
13	18.01.	***Generalized PCA***	Bethge
14	25.01.	***Clustering & Mixture models***	Bethge
14	01.02.	***Nonlinear feature spaces***	Bethge
17	04.02-15.02	**Exam**	

Logistics 3: Exercises and Exams

- After every lecture there will (usually) be a one-hour exercise session, in this seminar room.
- These sessions will be run by Nicolas Ludolph (nicolas.ludolph@student.uni-tuebingen.de)
- You will receive an exercise-sheet every week, and have to hand it in at the beginning of the subsequent lecture. We will return and discuss your solutions one week later.
- For example, the first exercise will be handed out today (19.10), you have to hand it on 26.10, and it will be discussed in the tutorial on 31.10.
- There be a ‘long’ tutorial on 30.11 which will cover the entire first section of the course.
- Starting from the second session, selected questions be presented by students. Expect to present at least one solution during the course of the course.
- To pass the course, you need to pass **both** exams **and** get 50% of the marks across all sheets. Note that sheets might vary in length and number of achievable marks.

Q: What is machine learning?

What is machine learning?

Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Shapire: Machine learning studies how to automatically learn to make predictions based on past observations.

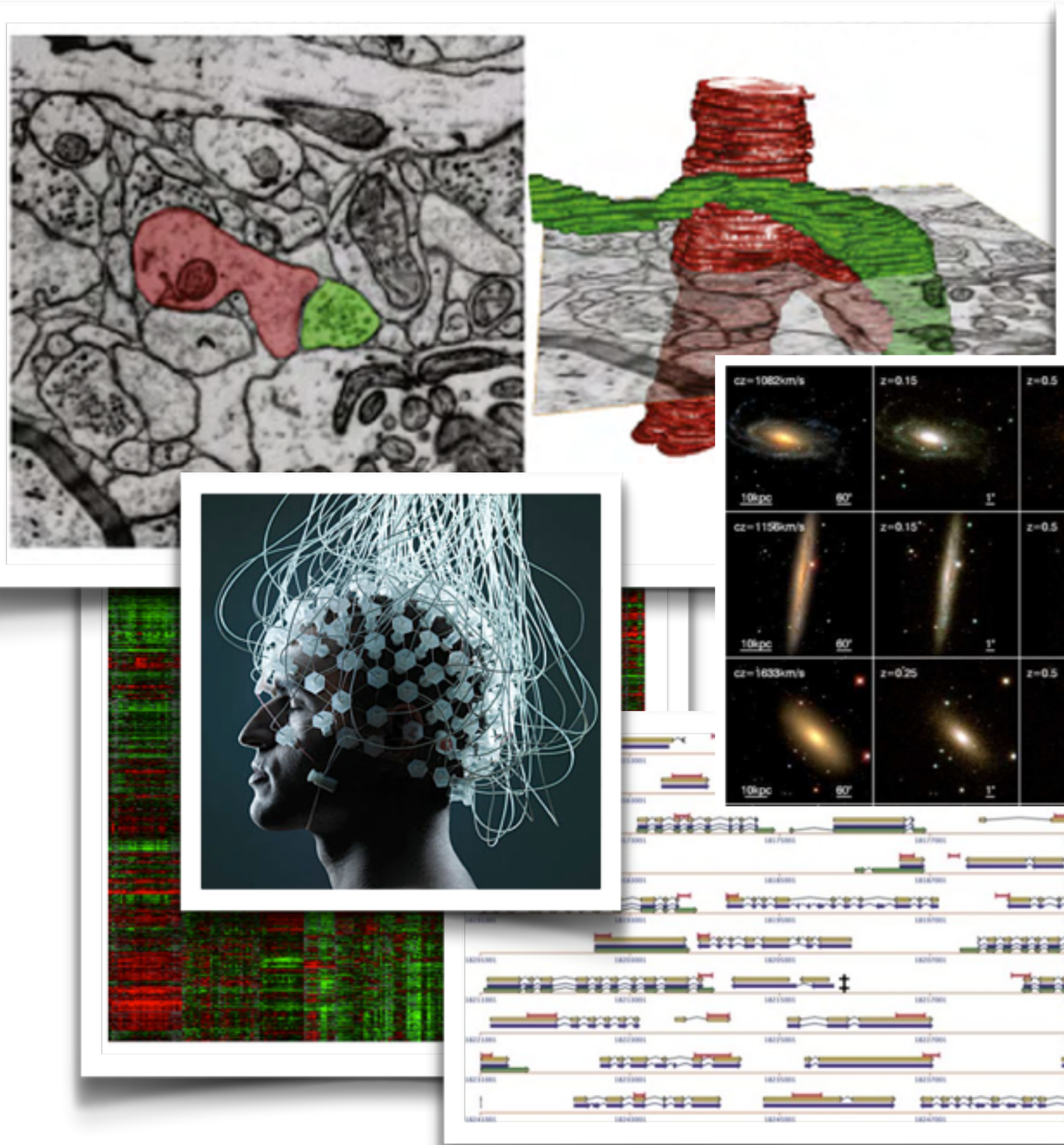
The field of machine learning tries to build and understand systems that can automatically extract information from empirical data in order to improve their performance.

As a scientific discipline, machine learning is an interdisciplinary (and relatively young) field focusing both on theoretical foundations of systems that learn, reason and act as well as on practical applications of these systems.

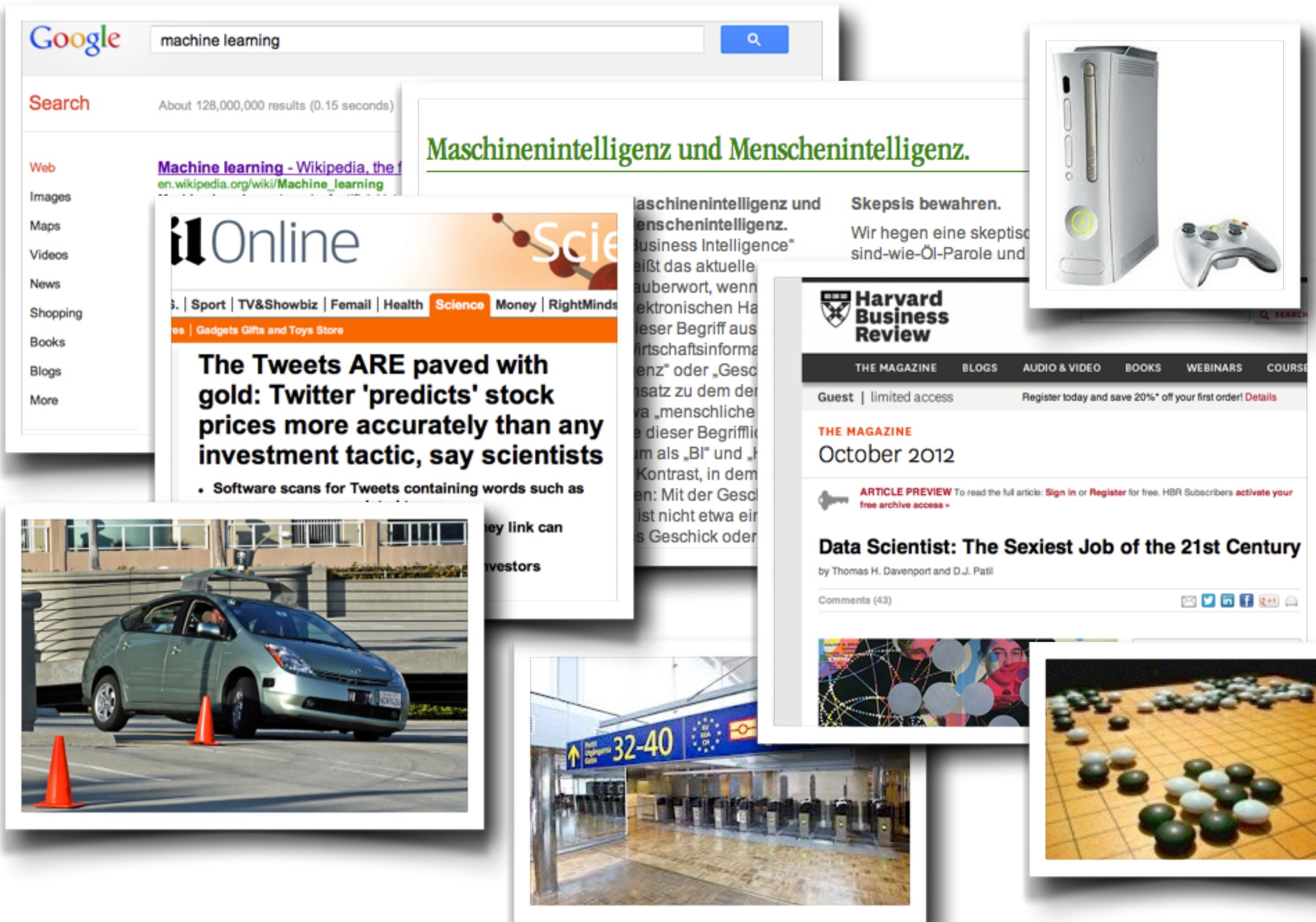
Machine learning draws inspiration and concepts from many scientific fields

- Statistics: Inference from data, probabilistic models, learning theory, ...
- Mathematics: Optimization theory, numerical methods, tools for theory, ...
- Engineering: Signal processing, system identification, robotics, control, information theory, data-mining, ...
- Computer science: Artificial intelligence, computer vision, information retrieval, data-structures, implementations ...
- Economics: decision theory, operations research, econometrics, ...
- Psychology/Cognitive science: Computational linguistics, learning, reinforcement learning, movement control, ...
- Physics: Energy minimization principles, entropy, capacity
- Computational Neuroscience: Neural networks, principles of neural information processing, ...
- Frequently: Information flowing back in from application domains, e.g. tools for bioinformatics getting used in other domains, ...

Machine learning techniques are used in many scientific disciplines ...



... and are becoming more and more pervasive even in everyday life



Q: Why is machine learning important for neuroscience?

Machine learning provides both important statistical tools for neuroscience as well as a conceptual foundations and inspiration

A) Machine learning provides tools for neural data analysis. Examples:

- Use of classification algorithms to ‘decode’ stimuli/mental representations from functional imaging data
- Methods for spike-sorting
- Bayesian inference for psychometric functions
- Predicting spikes from calcium transients
- Receptive field mapping
- Many, many more...

B) Machine learning provides a conceptual foundation for many tasks in neuroscience and related fields. How do sensory systems and organisms learn structure from data, and use this information to make (good or even optimal) decisions? Examples:

- Learning sensory representations from natural stimuli
- Bayesian cue-integration
- Models of reinforcement learning
- Predictive coding
- Probabilistic population
- Models of memory retrieval
- Many, many more...

There are three major types of machine learning: Supervised learning, unsupervised learning and reinforcement learning

Suppose you have some data x_1, x_2, x_3 ...

- **Supervised Learning:** You are also given some desired outputs y_1, y_2, y_3, ..., and your goal is learn a rule/function that you can use to predict y_i from x_i
- **Unsupervised Learning:** Your goal is to build a good 'model' of x that you can use for decision making, interpretation, other learning tasks, visualization, data-compression, science, etc...
- **Reinforcement Learning:** You have the ability to produce actions a_1, a_2, a_3 ..., and you will receive rewards (or punishments) r_1, r_2, r_3 ..., depending on your actions, the state of the environment, as well as your luck. Your goal is to find a strategy that allows you to maximize your rewards in the long term.

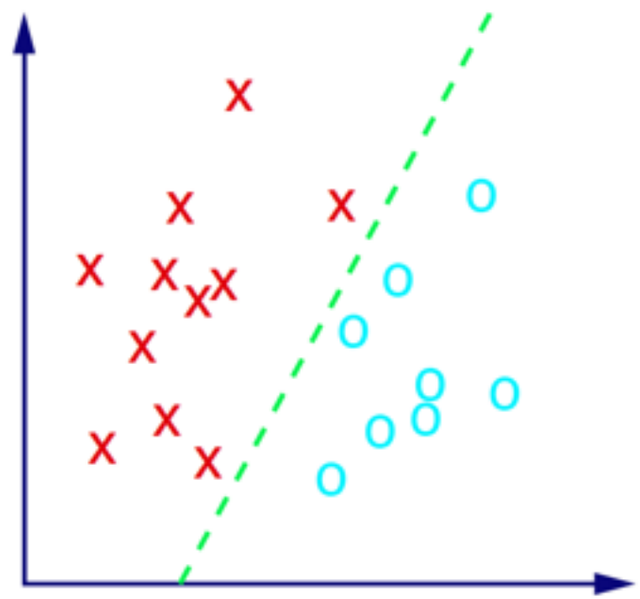
- Of course, this is a bit of an over-simplification, as there exist many variants of these types as well as hybrids ('semi-supervised learning') and approaches which do not quite fit into either of the three.
- We will deal with Supervised Learning (Jakob Macke, Part I) and with Unsupervised Learning (Matthias Bethge, Part II), but not with Reinforcement Learning

Part 1: Supervised Learning
Basic concepts

In supervised learning, the machine (or organism) has to predict unknown outputs given some sensory inputs

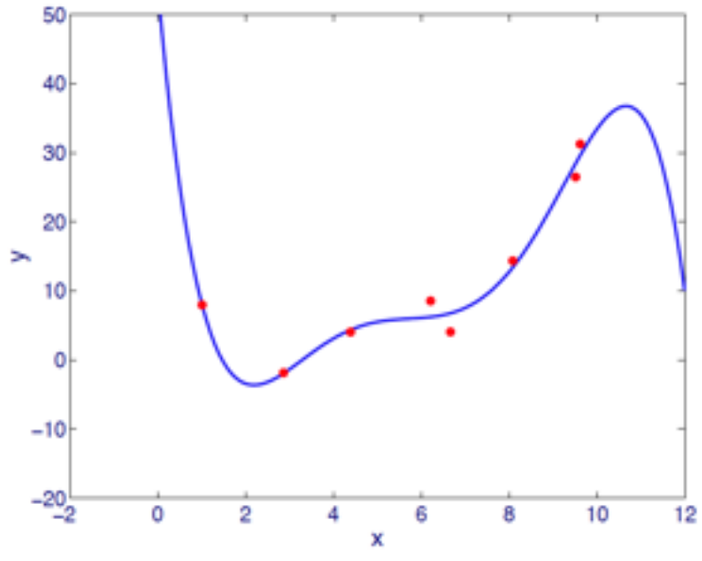
Classification: Predict binary (or categorial) output, e.g:

- Does this example belong to class one or two?
- Will a neuron spike in response to this stimulus?
- Based on this brain-scan, does this patient have a given disease or not?
- Will this customer buy this product or not?



Regression: Predict continuous (or multi-valued) output:

- How many spikes will the neuron fire?
- How quickly will the disease progress?
- How much is this customer willing to pay for this product?



Structured output regression: Anything more complex, e.g:

- What patterns of population activity are evoked by some stimulus?
- Based on some neural measurement, can you reconstruct the image/movie that was shown?

Example: Polynomial regression (Bishop PRML)

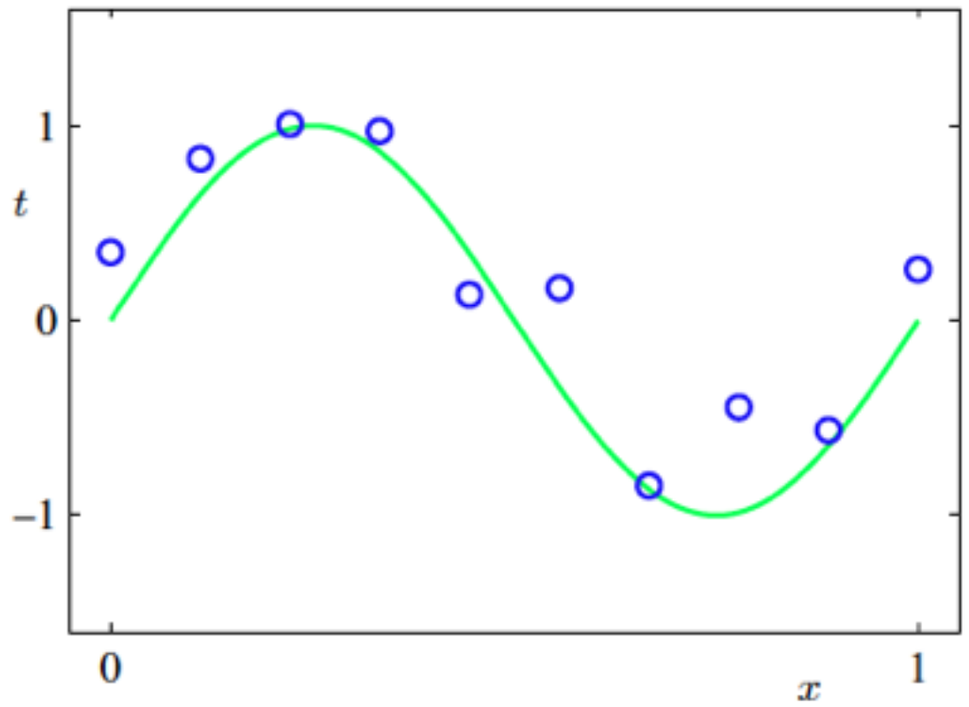


Image from Christopher M Bishop: Pattern Recognition and Machine Learning (PRML)

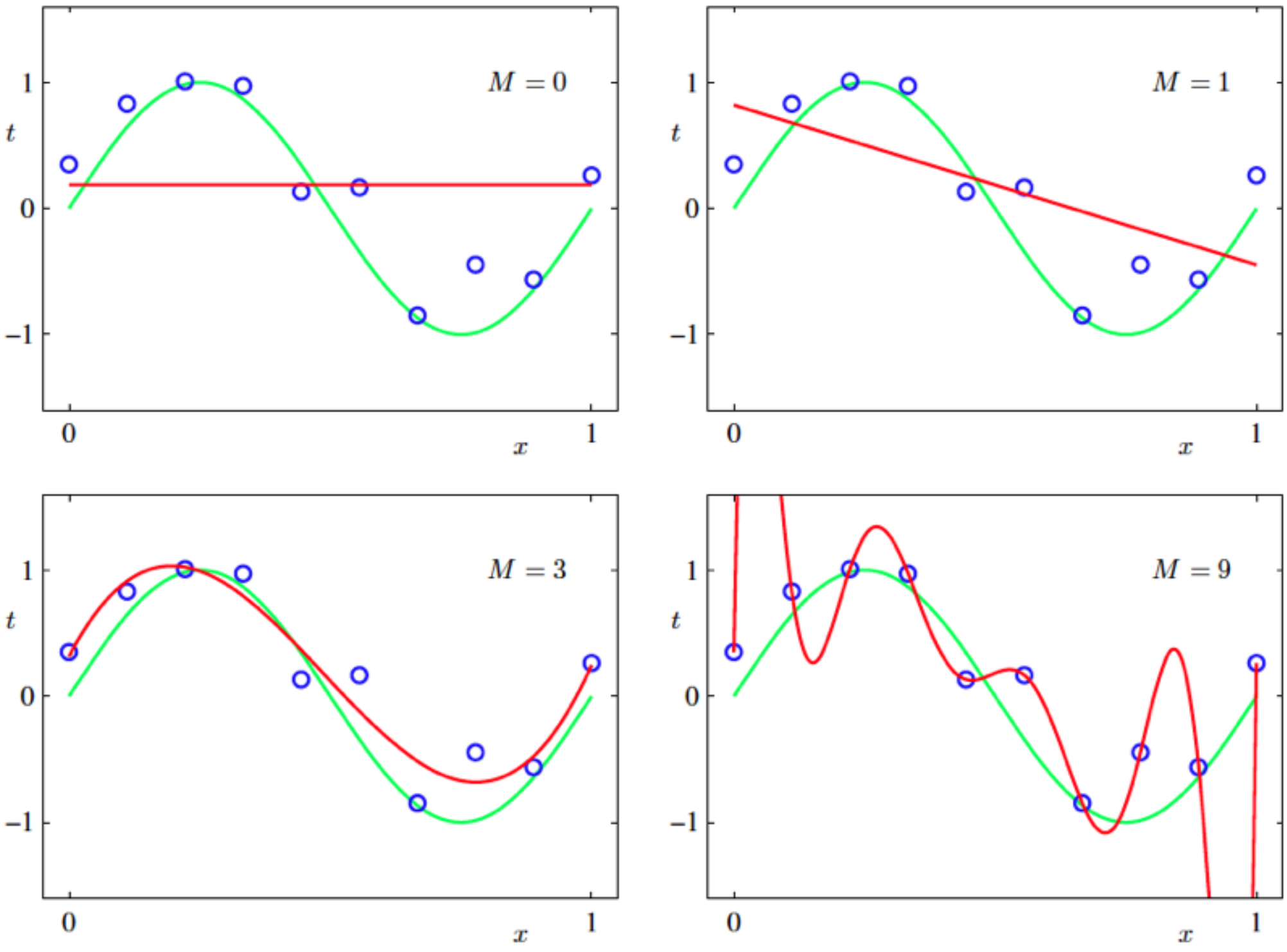
$$y(x, w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

$$E(w) = \sum_{n=1}^{N} (y(x_n, w) - t_n)^2$$

(Derivation of parameters that minimize error on board)

Which polynomial should we use?

Images from Christopher M Bishop: Pattern Recognition and Machine Learning (PRML)



We are interested in generalization ability. *Cross-validation* allows you to quantify how well your algorithm generalizes.

- Split your data (randomly) into non-overlapping ‘training’ and ‘test’ sets
- Fit parameters on ‘training set’
- Test quality of model by testing how well it fits on the test-set
- K-fold cross-validation: Repeat this procedure K times such that every data-point appears exactly once in a test-set.
- Cross-validation gives an estimate of the generalization error.

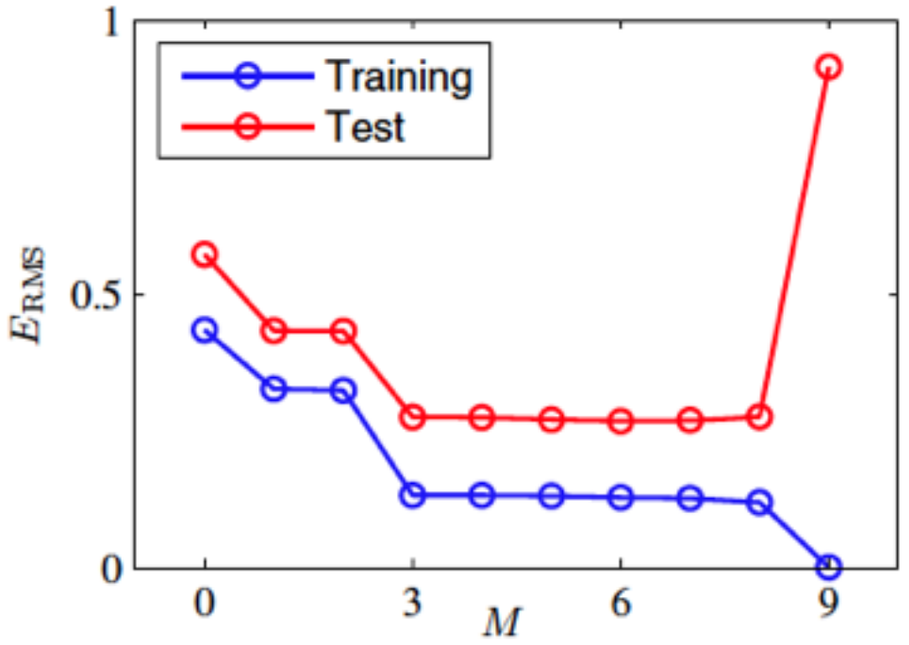


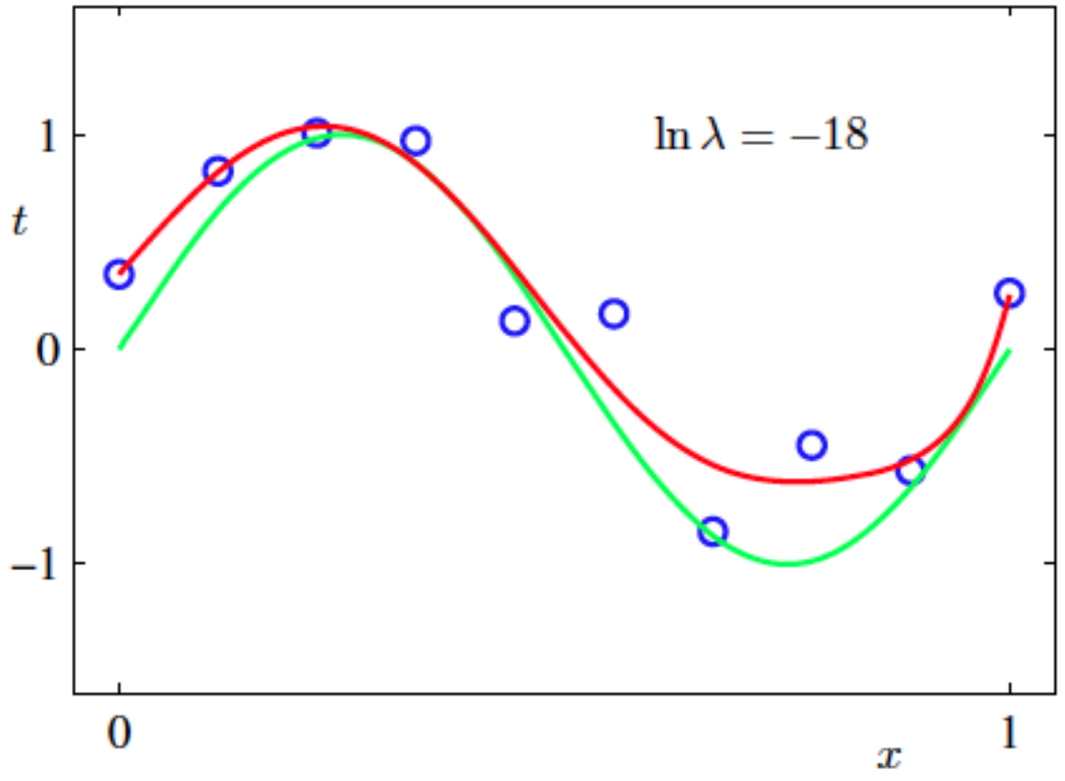
Image from Christopher M Bishop: Pattern Recognition and Machine Learning (PRML)

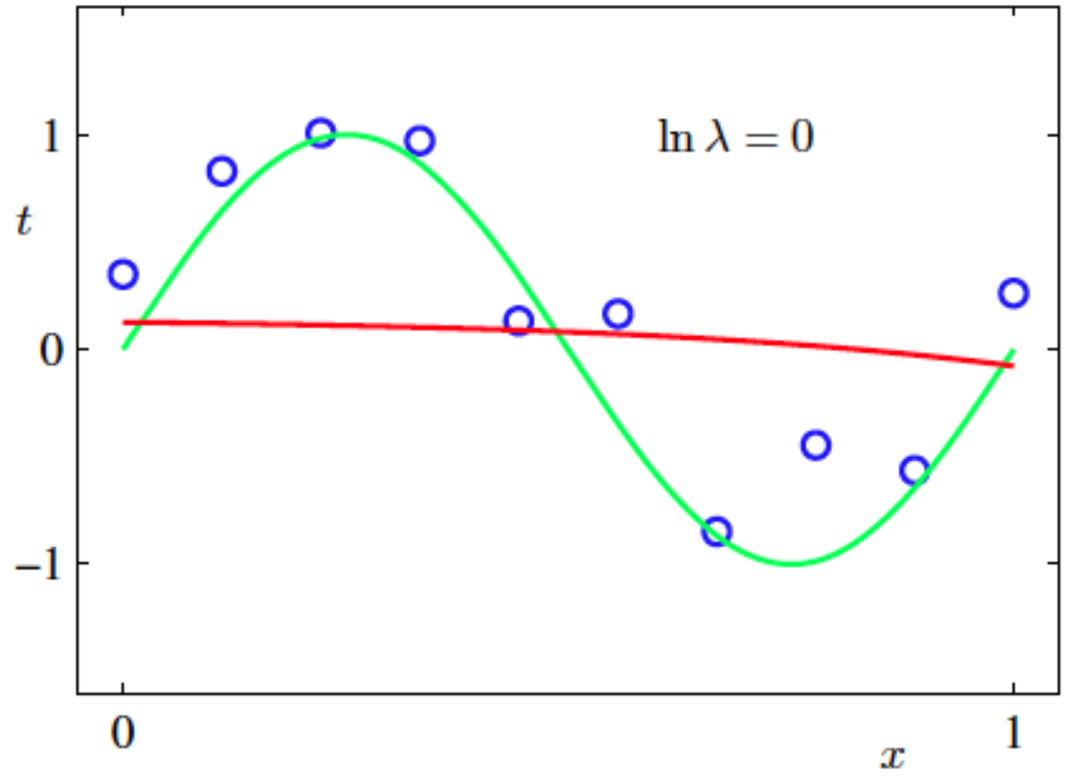
Regularization protects against over-fitting.

- Prefer simple explanations (=models) over complex ones: Occams Razor
- Trade off between model-complexity and goodness-of-fit
- Bayesian approach: Specify a prior distribution over models, and search for a model which both has high probability under the prior AND fits the data well

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Images from Christopher M Bishop: Pattern Recognition and Machine Learning (PRML)





In this course, we follow (and promote) a very *statistical* (and mostly Bayesian) approach to machine learning

- Many approaches in machine learning can be motivated from a statistical perspective.
- We will follow a statistical/probabilistic approach, but not be ‘hard core’ Bayesian
- Advantages of a probabilistic approach:
  - Unified theoretical and conceptual framework
  - Principled methods, e.g. for setting free parameters
  - Make inferences about missing inputs
  - Ability to sample from model
  - Scientific reasoning: Model comparison, hypothesis testing
  - Connections with neural coding and cognitive models
- Disadvantages: mostly computational in nature
  - Exact solutions are often intractable, and an approximate solution to a great problem can be worse than a simpler approach with an exact solution
  - ‘Non Bayesian’ algorithms are often faster and more efficient, especially on big data sets.
- Famous ‘non probabilistic’ algorithms: e.g. Support Vector Machines, Convolutional Neural Networks