Introduction: Machine Learning

Lecture Series „Machine Learning“

Niels Landwehr

Research Group „Data Science“
Institute of Computer Science
University of Hildesheim

Agenda For Lecture Today

- Organization and format of the lecture
- Motivation: why machine learning?
- Basic concepts in machine learning

Agenda For Lecture Today

- Organization and format of the lecture
- Motivation: why machine learning?
- Basic concepts in machine learning

Welcome

- Lecture „Maschinelles Lernen“/„Machine Learning" given by „Data Science“ group this year

Headed by Prof. Dr. Niels Landwehr

Research and teaching in machine learning and its applications

Contact:
<landwehr@uni-hildesheim.de>
Building J, Room 103 on main campus

- Lecture is alternating between ISMLL and data science groups
- Lecture content is similar, but not identical

Organization: Degree Programmes

- **Lecture „Maschinelles Lernen"/"Machine Learning": 2+2 SWS, 6 CPs**

- Degree programmes in which the lecture appears:
  - Bsc degrees „Informationsmanagement und Informationstechnologie" and „Angewandte Informatik" (mandatory)
  - Bsc degree „Wirtschaftsinformatik" (elective)
  - Msc degree „Data Analytics" (mandatory)
  - Msc degrees „Informationsmanagenement und Informationstechnologie" and „Angewandte Informatik" (elective, if not taken during Bachelor studies)

- Other degrees: as given in degree regulations

- The lecture is a pre-requisitive for many follow-up courses in the area of machine learning offered by ISMLL and Data Science groups

Organization: Lecture

- **Language:** language of lecture and also exercises will generally be English, but ok to ask questions in German

- **Format of lecture:** we will (mostly) use an **inverted-classroom** format:
  – The main lecture content will be in a weekly 90-minute video lecture that is made available every Wednesday on the Learnweb page of the course
  – Additionally, we will meet every Friday at 10:15 in lecture hall H2 for a Q & A session to ask questions and discuss the lecture content
  – In order to follow the lecture, you must watch the weekly video lecture
  – Additionally, it is highly recommended to come to the on-campus Q & A, especially if you have questions about the current video lecture
  – To be able to ask meaningful questions, please watch the video lecture before coming to the Q & A on Friday

Organization: Lecture

- **Motivation for inverted-classroom format:**
  – In inverted-classroom format, we use the in-presence time for interactive discussions rather than just a frontal lecture which tends to not be very interactive
  – Experience is that students often can best ask questions **after** having gone through a lecture in detail: difficult to ask questions "live" during the lecture
  – Having a video lecture is helpful for reviewing the lecture content later and studying for exams

- **Timeline**:
  – Every Wednesday there will be a new video lecture
  – This video lecture will be discussed Friday the same week
  – Today we have an introductory regular lecture as there is nothing to discuss yet
  – First video lecture will be available Wednesday next week (08.11.2023)
  – First Q & A session will be on Friday 10.11.2023

Organization: Tutorials

- **Tutorials**: The lecture will be accompanied by weekly tutorial sessions

- There will be five groups:
  - G1: Monday 8:15-9:45, Room C 2.13, Samelson [Jafar Bakhshaliyev]
  - G2: Monday 8:15-9:45, Room B 1.48, Samelson [Muhammad Sameer Ali Khan]
  - G3: Wednesday 8:15-9:45, Room C 2.13, Samelson [Aseel Alhermi]
  - G4: Wednesday 8:15-9:45, Room B 0.26, Samelson [Can Shenol Berk]
  - G5: Wednesday 14:15-15:45, Room G 009, main campus [Ujjwal]

- You will be assigned to a tutorial group using a poll in the Learnweb:
  - On the learnweb page of the tutorials, you will find a link called „Choose your tutorial group"
  - Click on the link and enter three options for tutorial groups that you would like to attend (ordered by your preferences)
  - **You have to enter your preferences until Monday, 06.11.2023 using that link**
  - On Tuesday 07.11.2023, we publish the assignment of students to tutorial groups

Learnweb Page of Course

- **Learnweb page of course:** https://www.uni-hildesheim.de/learnweb2023/course/view.php?id=2991

- Learnweb page contains much important information about organization of lecture
  - Dates, deadlines, announcements (also per email)
  - Lecture slides and video lecture
  - Weekly exercise sheets

- **Learnweb course for the exercises**: https://www.uni-hildesheim.de/learnweb2023/course/view.php?id=2992

- **Please enroll in the Learnweb courses for the lecture and for the exercises if you want to participate in the lecture and check the Learnweb frequently!**

Organization: Exercise Sheets

- There will be weekly exercises that are made available on the Learnweb page of the course
  - Some exercises are theoretical, some practical programming tasks
  - Exercise sheets are made available every Thursday, and you have to solve them and hand them in until the following Thursday at noon (12:00)
  - Solutions have to be handed in through the exercise Learnweb: we will make separate upload links available for tutorial groups 1, 2, 3, 4, 5.
  - **Please make sure you upload your solution to the right tutorial group (the one you were assigned to) to avoid confusion**

- Exercises will be corrected by tutors and the points you get in the exercises count up to 10% in the final grade you achieve for the course

- **First exercise sheet has just been made available, and is due 09.11.2023 at noon**

Next Steps and Next Meetings

- **Next steps and next meetings**

  Fri 03.11.2023: First introductory lecture (lecture hall H2)

  Fri 03.11.2023: First introductory exercise sheet is made available

  Wed 08.11.2023: First video lecture is made available in the Learnweb

  Thu 09.11.2023: You have to hand in solution to first exercise sheet

  Thu 09.11.2023: Second exercise sheet is made available

  Fri 10.11.2023: First Q & A (lecture hall H2, main campus)

  Mon 13.11.2023/Wed 15.11.2023: First tutorials take place

  Wed 15.11.2023: Second video lecture is made available in the Learnweb

  Thu 16.11.2023: You have to hand in solution to second exercise sheet

  Thu 16.11.2023: Third exercise sheet is made available

  Fri 17.11.2023: Second Q & A (lecture hall H2, main campus)

  Mon 20.11.2023/Wed 22.11.2023: Second tutorials take place

  ...

Questions?

Agenda For Lecture Today

- Organization and format of the lecture
- Motivation: why machine learning?
- Basic concepts in machine learning

Motivation: Computer Vision

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Example image classification**: it is very difficult to manually write a program that can classify images from their pixel representation
  - Image analysis is difficult due to „semantic gap“: pixel values determine semantic content of image, but the mapping from pixels to content is not straightforward
  - Manually writing code that analyzes pixel representation and determines, for example, if a cat is visible in an image is very difficult

In this example image, a human observer will immediately recognize the cat.

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

Computer only „sees“ big grid of numbers (pixel values).
Not easy to go from pixel values to semantic category!

J. Johnson, 2019

Motivation: Natural Language Processing

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Example automatic translation**: it is very difficult to manually write a program that can translate text e.g. from German to English at high quality
  - We know a lot about languages (world-level dictionaries, rules of grammar, etc.), but the complexity and ambiguity of natural language is a huge challenge
  - Manually designed rule-based systems can translate language up to a certain level of quality, but lack far behind human translators

Example for ambiguity in language:

“The dog liked to guard the house and the postman could not make it to the door because he was barking viciously”

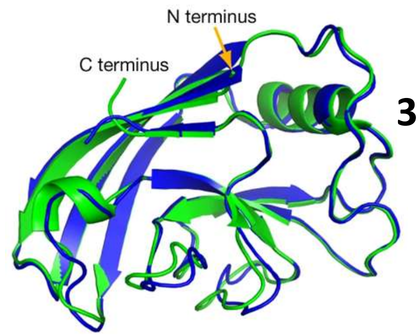
⇨ From grammatical perspective, unclear if the dog is barking or the postman

Motivation: Protein Folding

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Protein folding**: it is very difficult to manually write a program that can predict how a protein will fold in 3D based on its amino acid sequence
  - long standing and important problem in biology: predict the 3D structure of a protein from its amino acid sequence

Amino acid sequence



3D structure

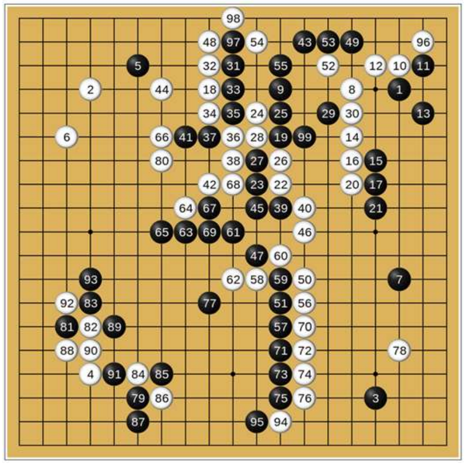
Jumper, J., Evans, R., Pritzel, A. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583–589 (2021).

  - Folding is based on principles of physics, which can be simulated using software
  - However, complexity is so high that predictions based on physical simulations are not always reliable

Motivation: Playing Go

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Game playing („Go“)**: it is very difficult to write a computer program that can beat a professional human player at the game „Go“
  - Even though the rules of Go are simple and can be easily encoded in software, programming a computer to play Go at expert level is very difficult
  - Studied for decades as a difficult benchmark problem for game-tree search
  - Much more difficult than e.g. chess due to high branching factor and difficulty in evaluating board positions (determining which player is ahead)



Game state in „Go“

Machine Learning

- **Machine learning: solve such difficult problems by learning a solution from data, rather than manually programming a solution!**

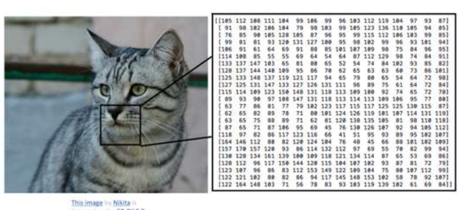
1. Collect **data** about the problem we are trying to solve: for example, images of cats and non-cats, known German-English translations, known protein structures, observed Go games, ...
2. From the observed data, **learn a model** that can solve the problem
3. Model can then be applied to new problem instances to solve them

- Surprisingly powerful paradigm that can be applied to surprisingly many domains
- **Machine learning is the driving force behind the „AI Revolution“**

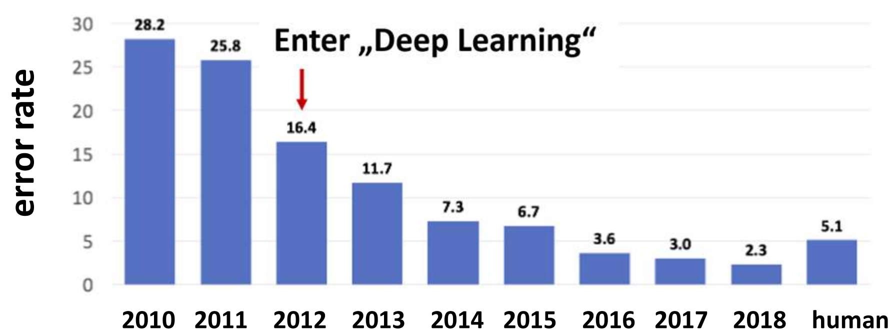
Machine Learning for Computer Vision

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Example machine learning for computer vision**: instead of manually programming a solution, collect example images and learn a classifier
- Modern machine learning has dramatically pushed the state-of-the-art in almost all computer vision problems
  - In some domains, machine-learning approaches reach superhuman performance
  - E.g. ImageNet competition: error rates have fallen from approx. 30% in 2010 to 2.3% in 2017, which is better than human performance



Results ImageNet competition



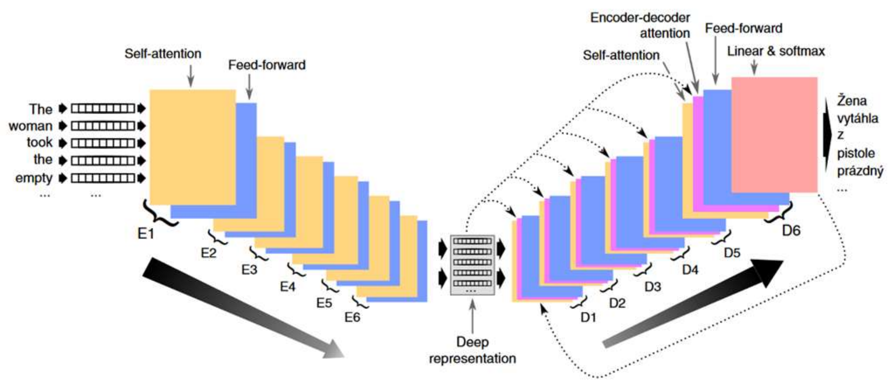
Machine Learning for Natural Language Processing

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Example machine learning for automatic translation**: instead of manually programming a solution, collect example translations and train a machine learning model

Translation model

input sentence (English)



output sentence (Czech)

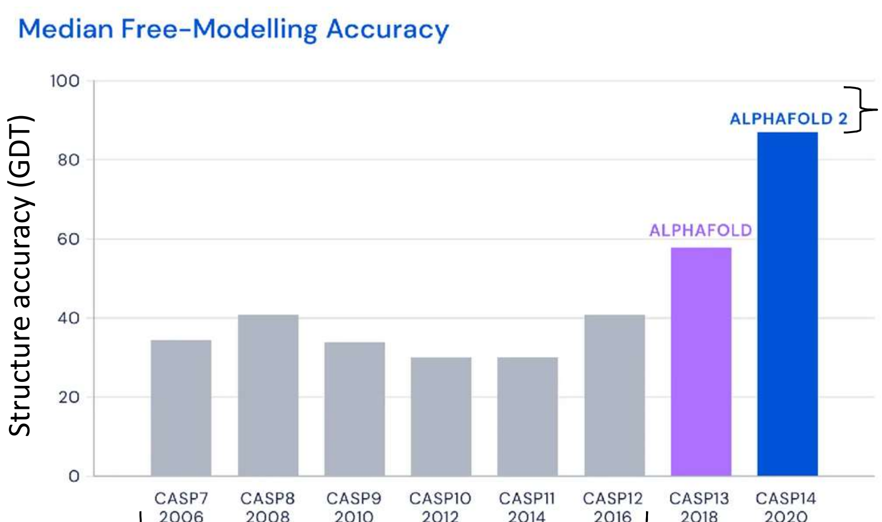
Popel, Martin, et al. "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals." *Nature communications* 11.1 (2020): 1-15.

- Dramatically improved the quality of machine translation compared to earlier systems, in some cases reaching or surpassing translation quality of human professionals

Machine Learning for Protein Folding

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

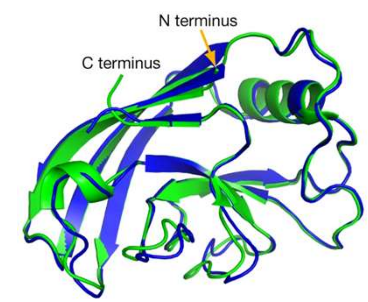
- **Example machine learning for protein fold prediction**: instead of relying on physical simulations, learn to predict structures based on a data set of known structures
  – Again, machine learning approaches have yielded dramatic improvements
  – E.g. „Critical Assessment of Structure Prediction" (CASP) competition: machine-learning system AlphaFold greatly reduces error compared to previous methods



error reduced by a factor of four

for many years, limited progress

Image: DeepMind



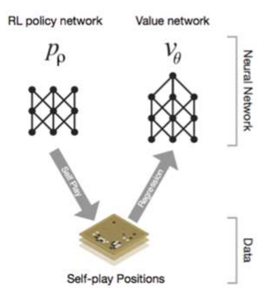
Jumper, J., Evans, R., Pritzel, A. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583–589 (2021).

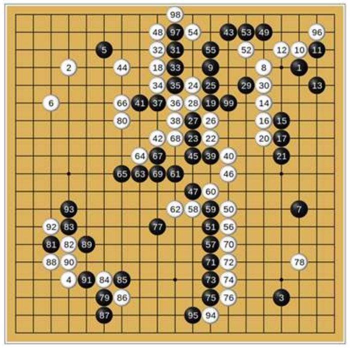
Machine Learning for Playing Go

- **Motivation: how do we solve problems in computer science for which manually programming a solution is difficult?**

- **Example machine learning for game playing („Go“)**: instead of manually programming a Go playing engine, let a computer learn to play Go through playing games against itself („reinforcement learning“)
  – Until relatively recently, it was thought that Go programs that play at the level of professional human players were still decades away
  – Using deep reinforcement learning has quickly lead to Go programs that play far better than any human player

In 2016, in a widely televised match, the machine learning-based system AlphaGo beats Lee Sedol, the best Go player in the world





Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (2016)

One Research Area, Many Names (and Aspects)

- **Data-driven solutions in computer science are a broad field and are studied under different names, which each stress different aspects:**
  - **Machine learning**: focus on the actual step of building models from data, using different kinds of models from decision trees to probababilistic models or deep neural networks
  - **Pattern recognition**: similar meaning as machine learning, often used in engineering, for example for computer vision and speech task
  - **Data mining, big data**: stresses the aspect of large data sets and complicated tasks
  - **Knowledge discovery in data bases (KDD)**: stresses the embedding of machine learning tasks in applications, that is, aspects such as preprocessing and deployment
  - **Data analysis**: historically stresses multivariate regression and unsupervised tasks (see below)
  - **Applied statistics**: stresses underlying statistical models, testing and methodological rigor
  - **Predictive analytics, business analytics, data analytics**: stresses business applications
  - **Data science**: umbrella term encompassing aspects of machine learning, big data, data analysis etc

⇒ No fully standardized terminology, can be a bit confusing...

Agenda For Lecture Today

- Organization and format of the lecture
- Motivation: why machine learning?
- Basic concepts in machine learning

Supervised Learning: Prediction

- In the following, we start to formalize the „learn a model from data“ idea, starting with the most common machine learning setting called **supervised learning**

- In supervised learning, the goal is to make predictions y about some type of objects $\mathbf{x}$:

Input: an **instance** (object) $\mathbf{x}$ that lives in an instance space $\mathcal{X}$

$$\mathbf{x} \mapsto y$$

Output: a **label** or **target** y that lives in a target space $\mathcal{Y}$

- To obtain predictions, we are looking for a **model** f that produces a prediction $f(\mathbf{x}) \in \mathcal{Y}$ for an input instance $\mathbf{x} \in \mathcal{X}$:

$$f : \mathcal{X} \to \mathcal{Y}$$

Input: **instance** $\mathbf{x}$

$$\mathbf{x} \mapsto f(\mathbf{x})$$

Output: prediction $f(\mathbf{x})$

- Remark on notation: within formulas, we will usually use bold notation to indicate vectors or matrices and non-bold for scalar variables

Supervised Learning: Instances

- **How can we represent objects as instances $\mathbf{x}$?**
- In the simplest case, instances are simply represented by vectors, that is, $\mathcal{X} = \mathbb{R}^M$:
- The elements x_m of the vector $\mathbf{x}$ are called **features** or **attributes**

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_M \end{pmatrix}$$

A single element in the vector describes a particular attribute or feature of the object that is represented by the instance

- The dimension M is given by the number of attributes or features that describe an object: can range from <10 to hundreds of thousands

Example Instance Representation: Iris Data Set

- As an example for representing objects as instances, assume we are trying to classify Iris flowers into different classes

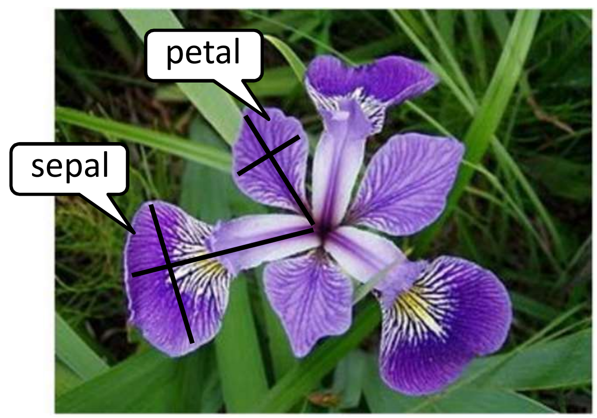
Iris versicolor

Iris setosa

Iris virginia

- A particular flower could be characterized by measuring its petal and sepal width and height, resulting in four attributes that describe an object/instance:

instance representation $$\mathbf{x} = \begin{array}{l} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \end{array} \begin{array}{l} \text{sepal length} \\ \text{sepal width} \\ \text{petal length} \\ \text{petal width} \end{array}$$



„Sepal“ and „petal“ refer to different parts of the Iris flower

Example Instance Representation: Digit Recognition

- As another example for representing objects, consider the problem of recognizing scanned handwritten digits, which are represented as 28 x 28 pixel grayscale images
- These images can be represented by a $28 \cdot 28 = 784$ dimensional vector:

Scanned handwritten digit: 28 x 28 pixel grayscale image

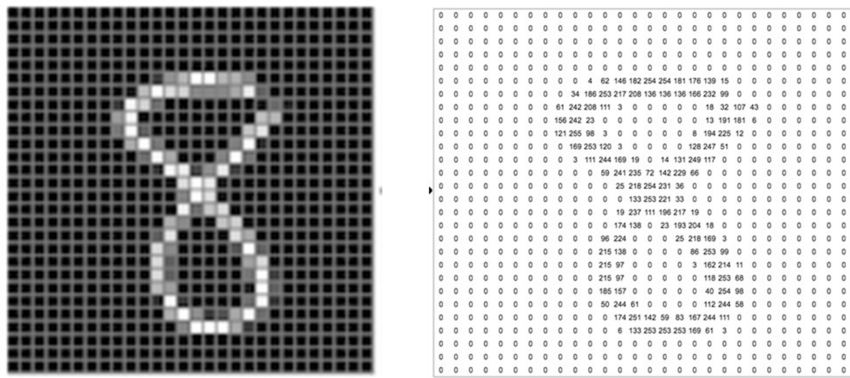


Image information:
$28 \cdot 28 = 784$ pixel values between 0 and 255.

first pixel value

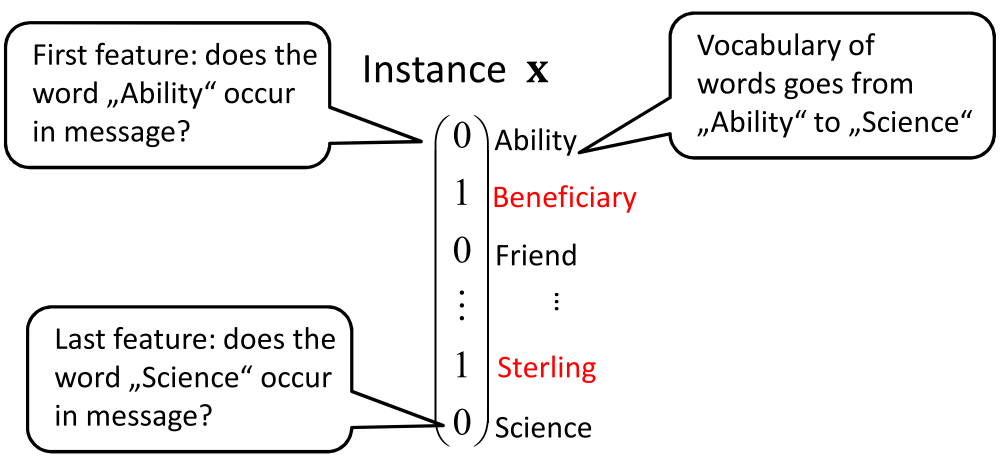
instance representation $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_{784} \end{pmatrix}$

last pixel value

Note: representing images as vectors is not the best idea, better to use a matrix or tensor representation(see e.g. lecture „Advanced Computer Vision“)

Example Instance Representation: Spam Filtering

- As another example for representing objects as instances, assume we are trying to classify email messages into legitimate and spam messages
- An email message can be represented by a binary word occurence vector
  - There is a vocabulary of words that can occur in messages
  - An instance is represented by a vector $\mathbf{x} \in \mathbb{R}^M$, where M is the size of the vocabulary. That is, there is one element in the vector for every word
  - The word-specific features x_m are zero or one depending on whether that word appears in the message or not



Email message

> Dear Beneficiary,
>
> your Email address has been picked online in this years MICROSOFT CONSUMER AWARD as a Winner of One Hundred and Fifty Five Thousand Pounds Sterling…

Supervised Learning: Targets

- We have talked about representing instances $\mathbf{x} \in \mathcal{X}$ (using feature vectors)
- What are possible outputs or targets $y \in \mathcal{Y}$?

- Depending on the outputs $y \in \mathcal{Y}$, different settings can be distinguished

- In **classification**, the output is one of a fixed set of possible classes
  - Formally, the target space is therefore $\mathcal{Y} = \{c_1, \dots, c_T\}$
  - Without loss of generality and to simplify notation, we often assume $\mathcal{Y} = \{1, \dots, T\}$: can simply rename the classes to $1, \dots, T$

- In **regression**, the output is a continuous value (e.g. temperature, price, ...)
  - Formally, the target space is therefore $\mathcal{Y} = \mathbb{R}$

- More complex settings also exist: for example, in **structured output prediction** the target space $\mathcal{Y}$ can contain structured outputs such as sequences, trees, graphs etc.

Learning Models From Data

- Recap: we want to have a model $f : \mathcal{X} \to \mathcal{Y}$ that predicts targets for instances

- **Idea of machine learning**: the model $f : \mathcal{X} \to \mathcal{Y}$ will be inferred from training data
- The **training data** is a set

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

Training instances $\mathbf{x}_n \in \mathcal{X}$: observed objects in training data, for example flowers, images of digits, or emails

Observed labels or targets $y_n \in \mathcal{Y}$ in training data, for example classes of flowers, digits 0...9, or spam/legitimate classifications

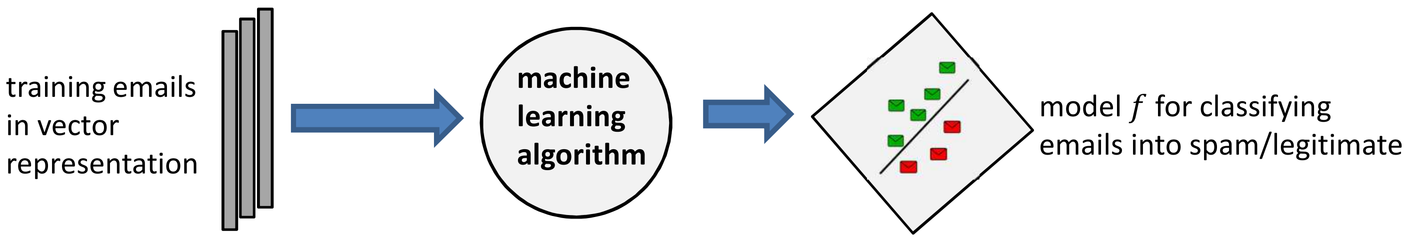
- Idea: from the training data we can indirectly observe the relationship between inputs and targets
  - Are there any patterns in instances that make a certain class more likely?
  - E.g. pixel patterns that indicate cats, or words that indicate spam messages

- **Learning**: constructing a model $f : \mathcal{X} \to \mathcal{Y}$ that extracts these relationships from the data and thereby makes them explicit and applicable to predictions

Learning Algorithms

- The main task in machine learning is to construct the model $f : \mathcal{X} \to \mathcal{Y}$ from data $\mathcal{D}$
- A **learning algorithm** takes as input the training data $\mathcal{D}$ and produces a model f

Example spam filter:



- Often, learning is formulated as an optimization problem:
  - There is a space of possible models f
  - The learning algorithm has to search in this space for a model that captures the $(\mathbf{x}, y)$-relationship observed in training data well
  - Many possible choices for models, settings, and optimization approaches: see following lectures during the semester

Training Data: Assumptions

- For learning to work, we have to assume that there is some reasonably stable relationship between inputs and outputs that can be captured by a model

- Typical assumption: training examples are independently drawn from a (constant) joint distribution over inputs and outputs:

$$(\mathbf{x}_n, y_n) \sim p(\mathbf{x}, y)$$

- Because $p(\mathbf{x}, y) = p(\mathbf{x})p(y \mid \mathbf{x})$, the assumption can be reformulated as

- The instances $\mathbf{x}_n$ are sampled from a probability distribution over instances
- $p(\mathbf{x})$ describes distribution over population of objects
- For example, certain flowers, digits, or email texts are encountered with a certain probability

$$\mathbf{x}_n \sim p(\mathbf{x})$$
$$y_n \sim p(y \mid \mathbf{x}_n)$$

- Given an instance $\mathbf{x}_n$, its label is drawn from a distribution $p(y \mid \mathbf{x}_n)$ that represents the relationship between input and output
- The relationship could be deterministic (probabilities 0 or 1) but this formulation also allows for randomness or noise in data

Assumptions About Data at Application Time

- Crucially (and somewhat optimistically) we assume that at application time, that is, when the trained model is deployed in the real world, it will encounter instances from the same distribution $p(\mathbf{x}, y)$ that the training set has been drawn from

- At application time, the model will encounter novel instances: e.g. novel measurements of Iris flowers, novel images of digits, or novel email texts
  - We assume that such a novel instance is drawn from the same distribution:

$$\mathbf{x}_{new} \sim p(\mathbf{x})$$

  - The model produces a prediction for the novel instance:

$$\hat{y} = f(\mathbf{x}_{new})$$

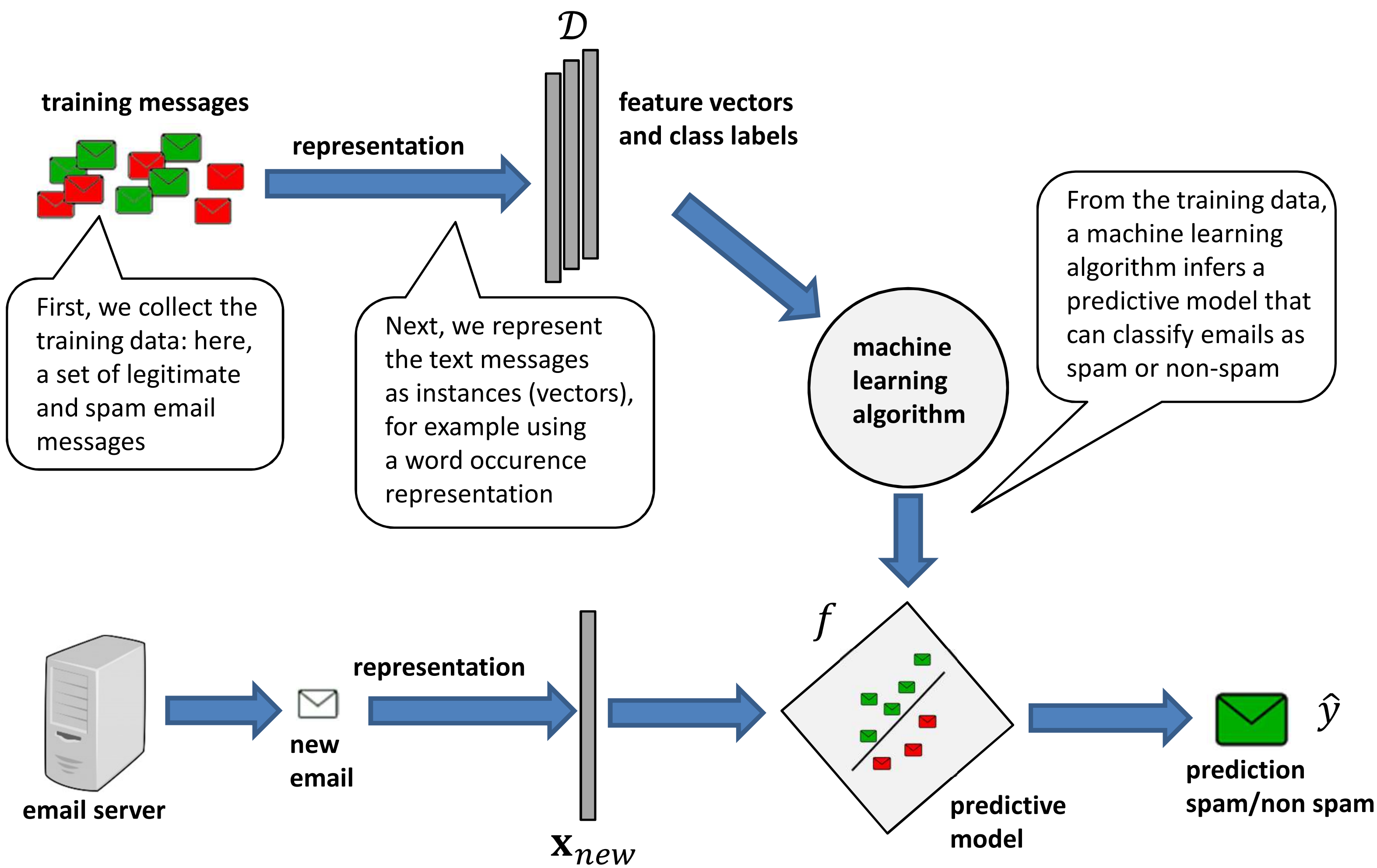
  - There is also a true target for the new instance, which is unknown (for example, the true class of the flower, true digit, or true legitimate/spam status of email):

$$y_{new} \sim p(y \mid \mathbf{x}_{new})$$

  - If learning has worked, we expect that prediction $\hat{y}$ and true target y_{new} are the same or close (more formal treatment later...)

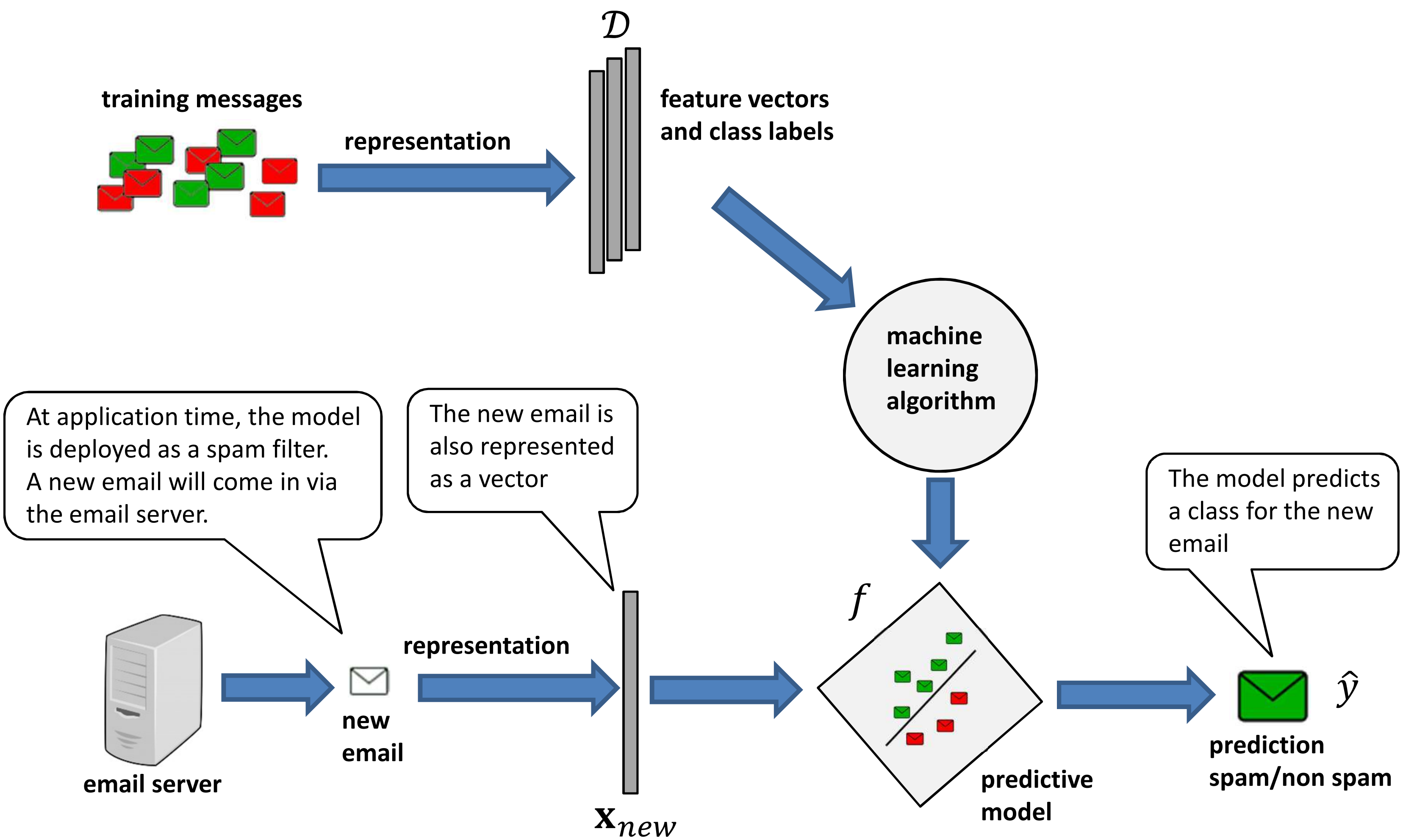
Overall Machine Learning Pipeline

- Typical overall machine learning pipeline, for the example of spam filtering



Overall Machine Learning Pipeline

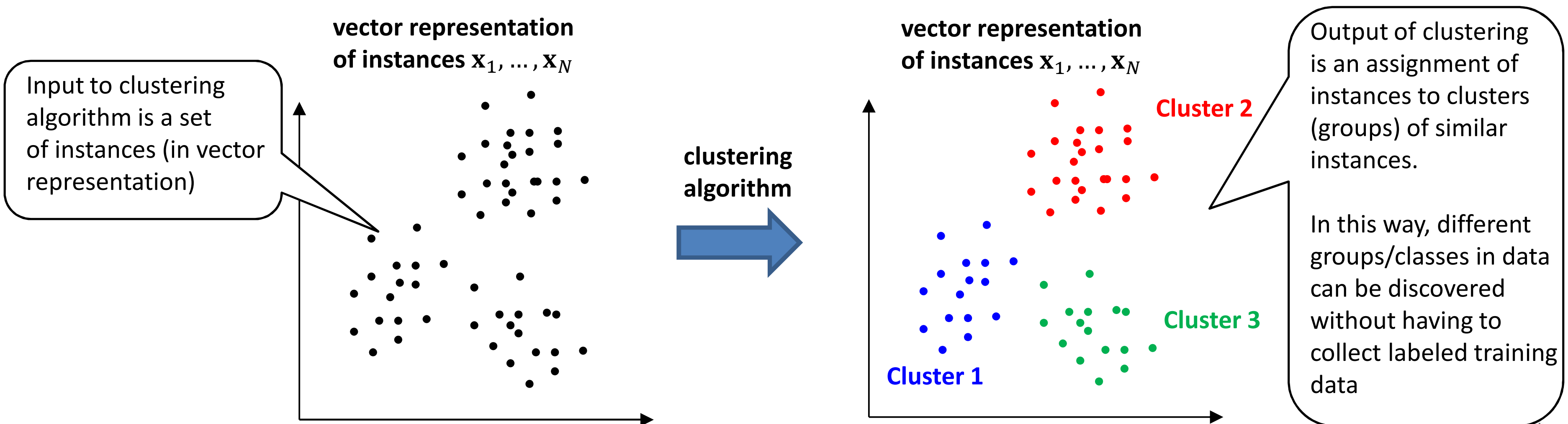
- Typical overall machine learning pipeline, for the example of spam filtering



Unsupervised Learning

- So far we talked about supervised learning, which is charaterized by having input-output pairs of the form $(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_N, y_N)$ as training data

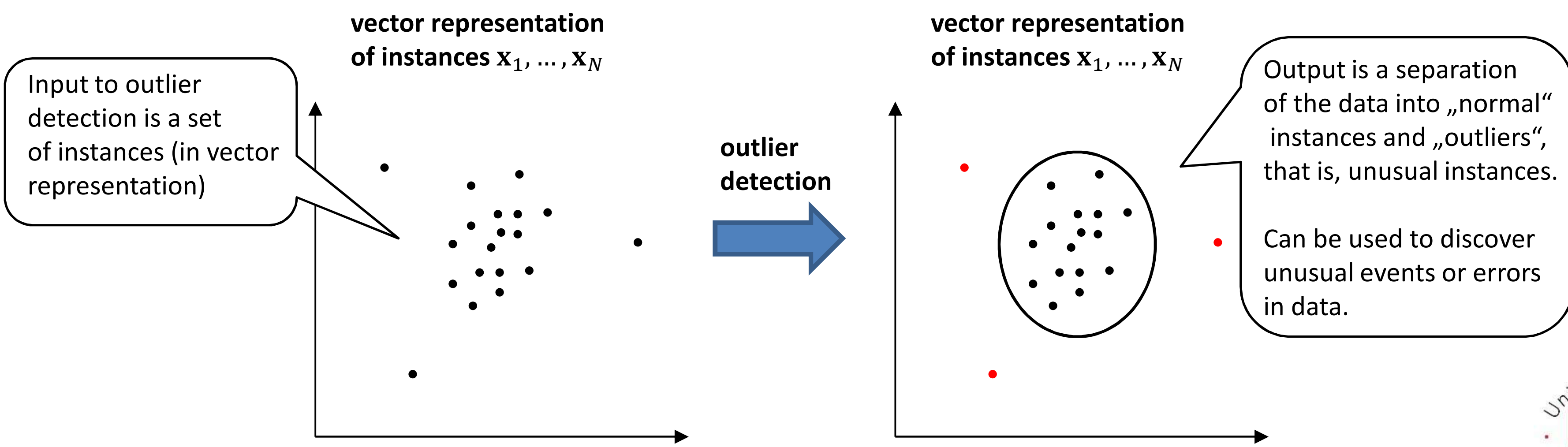
- **Unsupervised learning** is an alternative scenario for machine learning, in which only instances $\mathbf{x}_1,\ldots,\mathbf{x}_N$ but no labels $y_1,\ldots,y_N$ are available
- Within unsupervised learning, different problem settings can be studied, for example:
  – Clustering: find groups of similar instances



Unsupervised Learning

- So far we talked about supervised learning, which is charaterized by having input-output pairs of the form $(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_N, y_N)$ as training data

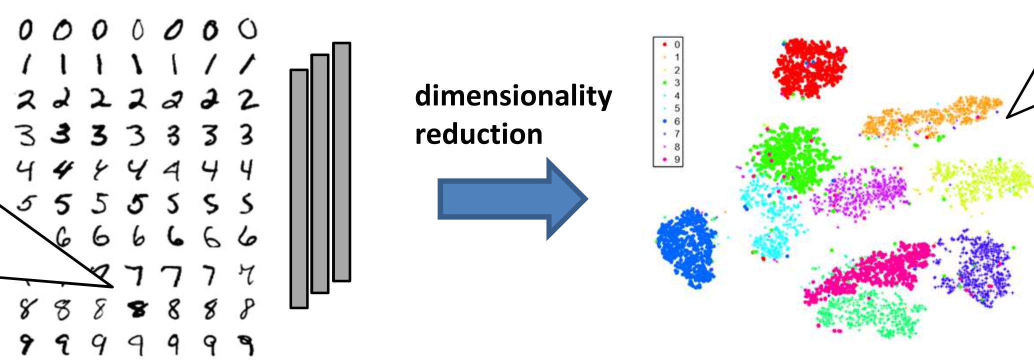
- **Unsupervised learning** is an alternative scenario for machine learning, in which only instances $\mathbf{x}_1,\ldots,\mathbf{x}_N$ but no labels $y_1,\ldots,y_N$ are available
- Within unsupervised learning, different problem settings can be studied, for example:
  – Clustering: find groups of similar instances
  – Outlier detection



Unsupervised Learning

- So far we talked about supervised learning, which is charaterized by having input-output pairs of the form $(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_N, y_N)$ as training data
- **Unsupervised learning** is an alternative scenario for machine learning, in which only instances $\mathbf{x}_1,\ldots,\mathbf{x}_N$ but no labels $y_1,\ldots,y_N$ are available
- Within unsupervised learning, different problem settings can be studied, for example:
  - Clustering: find groups of similar instances
  - Outlier detection
  - Visualization/ dimensionality reduction

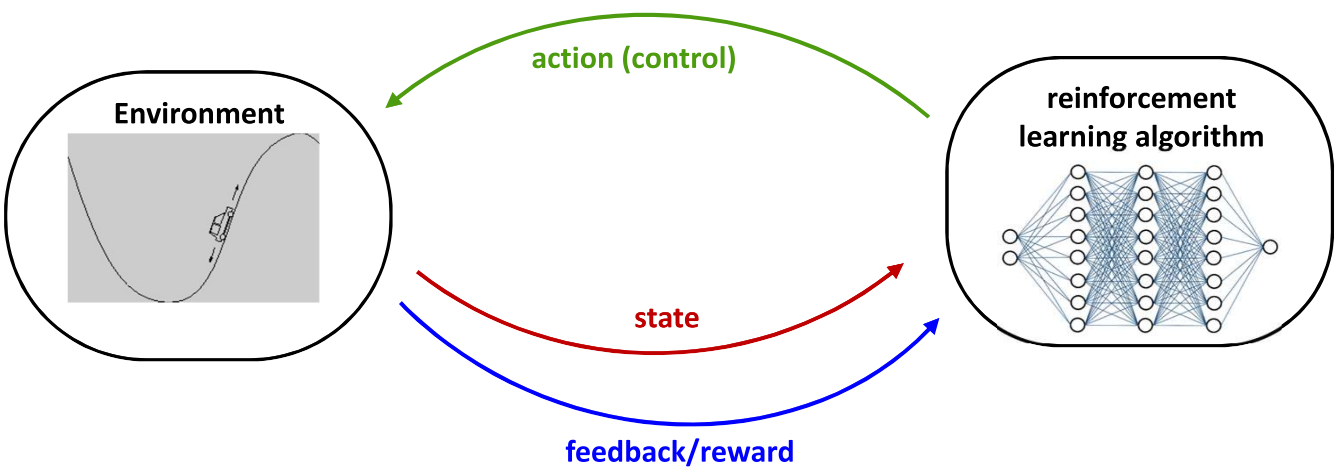
Input to dimensionalty reduction is a set of instances in high-dimensional vector representation, for example images of digits



Output of dimensionality reduction is the same set of instances in a lower-dimensional representation (here: 2D), such that similarities between instances are preserved. Here, digits 0-9 form visible clusters in 2D.

Reinforcement Learning

- Besides supervised and unsupervised settings, another important setting in machine learning is **reinforcement learning**
- In reinforcement learning, the goal is to solve a control problem:
  - An agent (controlled by the reinforcement learning algorithm) can interact with its environment and perceive the state of the environment
  - At any point in time, the agent can take one of several actions
  - The correct action is never shown, however, the agent gets **delayed feedback**: at certain points in time, it gets a positive or negative „reward“ that indicates whether the executed sequence of actions was successful or not

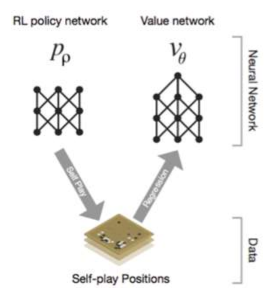


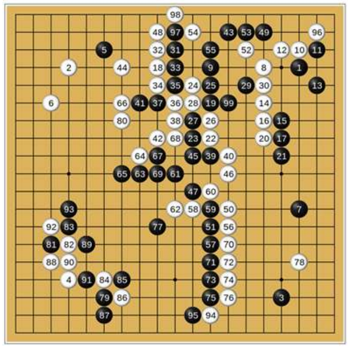
Reinforcement Learning: Example

- Example for reinforcement learning: learn to play the boardgame „Go“ (also see slides above)

- Observable is the current board state
- Possible actions: place a stone in one of the 381 grid cells of the board (minus illegal moves...)
- Reward: +1 if in the end game is won, -1 if game is lost
- Could train by playing against human players, or (more effective) a copy of itself

Leads to superhuman Go playing abilities (in combination with tree search techniques)





Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (2016)

Summary: Machine Learning Problems

- Summary: There are many different problem settings within machine learning

1. Density Estimation, generative models
2. Regression
3. Classification
4. Planning/Optimal Control
5. Clustering
6. Outlier Detection
7. Dimensionality Reduction
8. Association Analysis

Estimating the probability distribution over a set of instances or learning to generate novel instances from this distribution

supervised learning (2–3)

reinforcement learning (4)

unsupervised learning (5–8)

Find relationships in data (patterns, correlations, ...)

Literature

- Some books about machine learning:
  - Gareth James, Daniela Witten, Trevor Hastie, R. Tibshirani (2013): An Introduction to Statistical Learning with Applications in R, Springer.
  - Kevin P. Murphy (2012): Machine Learning, A Probabilistic Perspective, MIT Press.
  - Trevor Hastie, Robert Tibshirani, Jerome Friedman (2009): The Elements of Statistical Learning, Springer. Also available online as PDF at http://www-stat.stanford.edu/~tibs/ElemStatLearn/
  - Christopher M. Bishop (2007): Pattern Recognition and Machine Learning, Springer.
  - Richard O. Duda, Peter E. Hart, David G. Stork (2001): Pattern Classication, Springer.
  - Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016): Deep Learning, MIT Press. Also available online at www.deeplearningbook.org