Machine Learning
(IT3190E)

Quang Nhat NGUYEN

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology
School of Information and Communication Technology

Academic year 2020-2021

The course's content:

- **Introduction**
  - **Machine learning**
  - **Successful applications of ML in practice**
  - **Software frameworks and tools**
- Performance evaluation of ML system
- Supervised learning
- Unsupervised learning
- Ensemble learning
- Reinforcement learning

Introduction of Machine learning

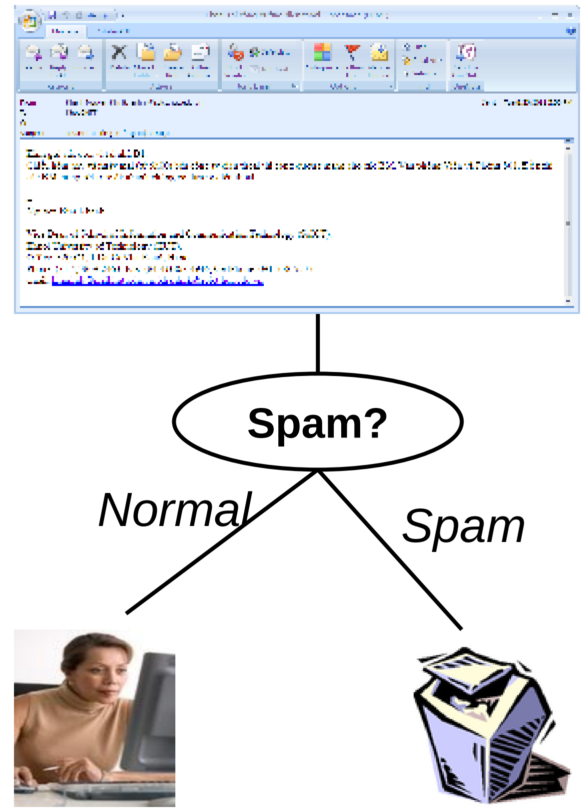
- Machine Learning (ML) is a traditional and very active field of Artificial Intelligence (AI)
- Some examples of definition of ML
  - → A process by that a system improves its performance `[Simon, 1983]`
  - → A process by that a computer program improves its performance in a task through experience `[Mitchell, 1997]`
  - → A programming of computers to improve a performance criterion based on past sample data or experience `[Alpaydin, 2004]`
- Representation of a ML problem `[Mitchell, 1997]`

  ML = Improvement of a task's efficiency through experience
  - A task ***T***
  - For the evaluation criteria of performance ***P***
  - By using some experience ***E***

Example of ML problem (1)

Email spam filtering:

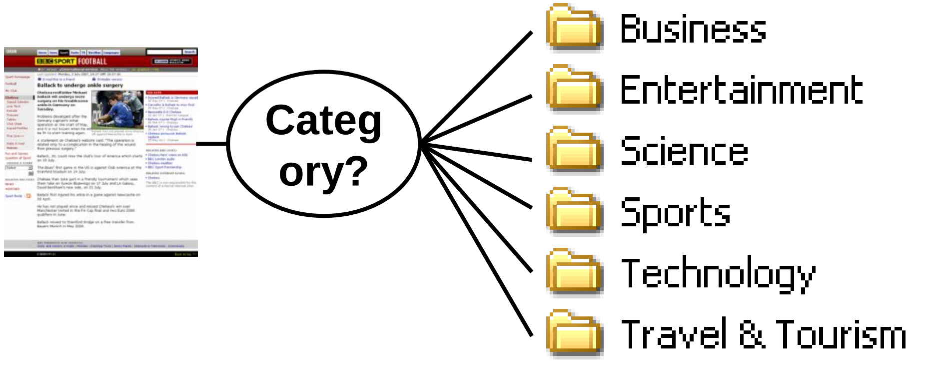
- ***T***: To predict (i.e., to filter) spam emails
- ***P***: % of correctly classified (i.e., predicted) incoming emails
- ***E***: A set of sample emails, where each email is represented by a set of attributes (e.g., a set of keywords) and its corresponding label (i.e., normal or spam)



Example of ML problem (2)

Web page categorization (classification):

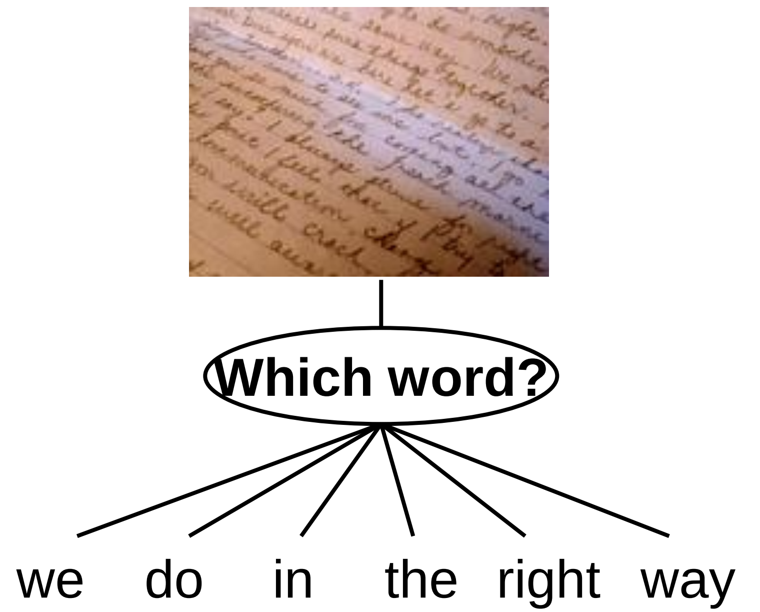
- ***T***: To categorize Web pages in predefined categories
- ***P***: % of correctly categorized Web pages
- ***E***: A set of Web pages, and each one associates with a category



Example of ML problem (3)

Handwritten characters recognition

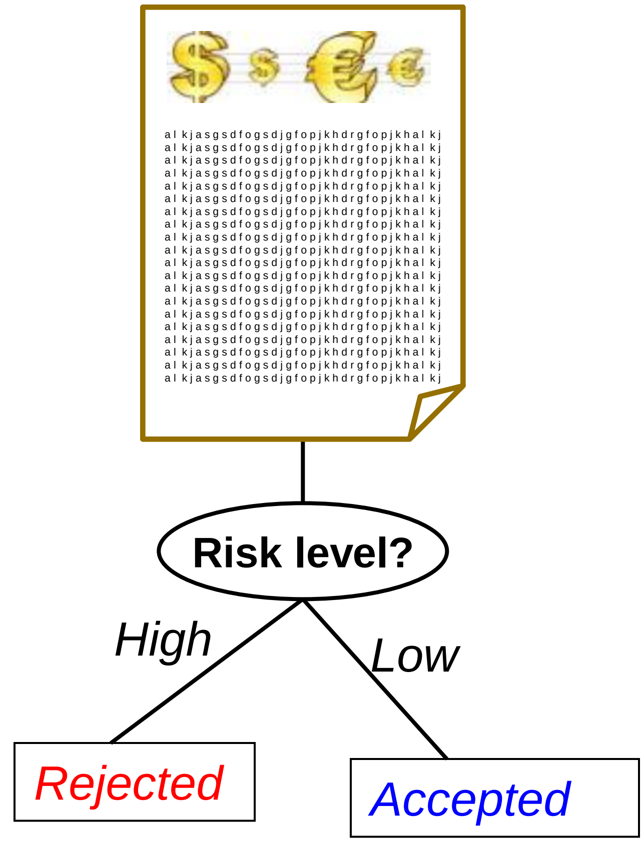
- ***T***: To recognize the words that appear in a captured image of a handwritten document
- ***P***: % of correctly recognized words
- ***E***: A set of captured images of handwritten words, where each image associates with a word's label (ID)



Example of ML problem (4)

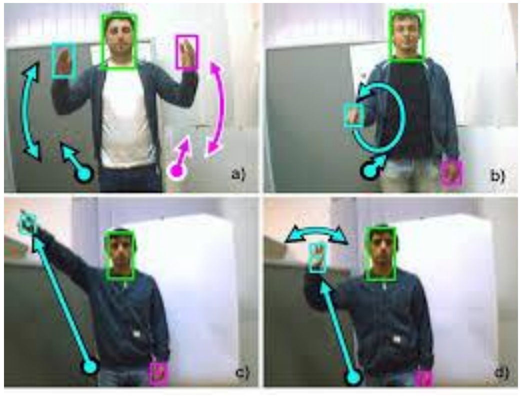
Risk estimation of loan application:

- ***T***: To estimate the level (e.g., high or low) of risk of a loan application
- ***P***: % of correctly estimated high-level-risk loan applications (i.e., those do not return the loans, or returns in a long delay)
- ***E***: A set of loan applications, where each loan application is represented by a set of attributes and a risk level value (high/low)



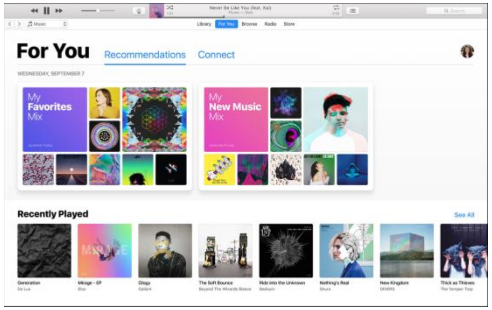
Successful applications of ML in practice (1)

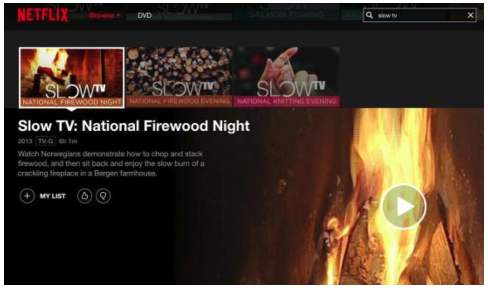
- Human-machine communication
  - Voice, Gesture, Language understanding, …



Successful applications of ML in practice (2)

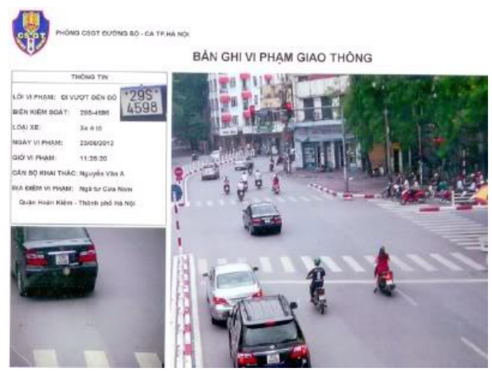
- Entertainment
  - Music, Movies, Games, News, Social networks, …





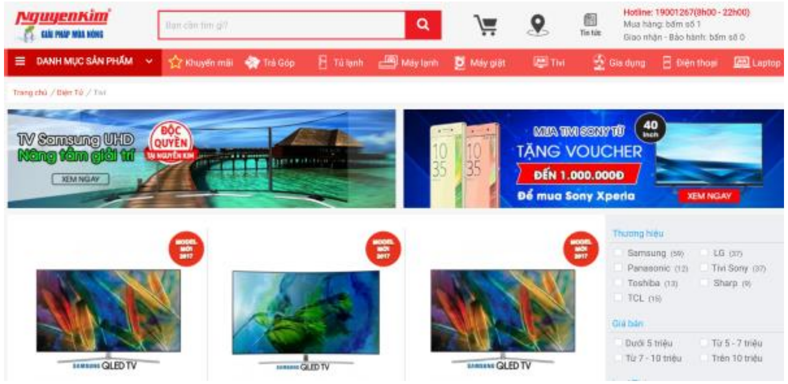
Successful applications of ML in practice (3)

- Transportation
  - Automatic car, Traffic surveillance, Car ride demand estimation, …



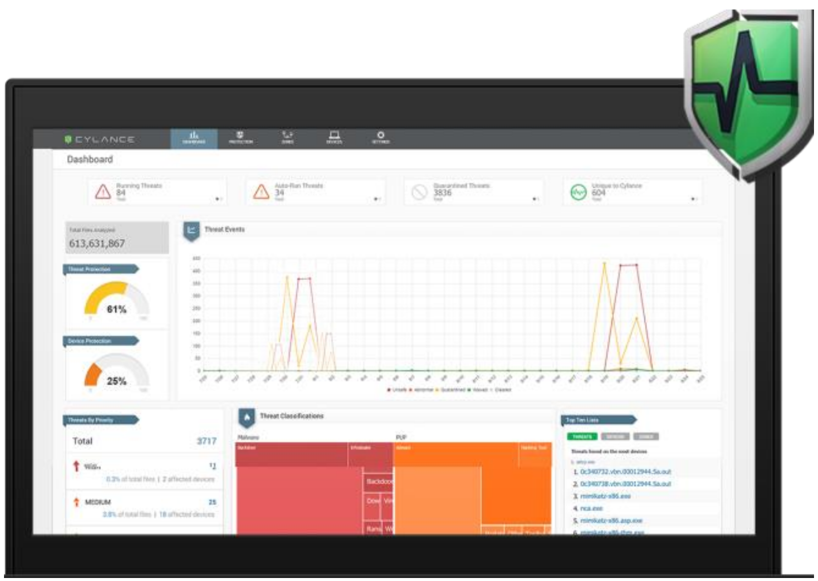
Successful applications of ML in practice (4)

- E-commerce
  - Recommendation of products and services, Customer need prediction, Promotion campaigns, …



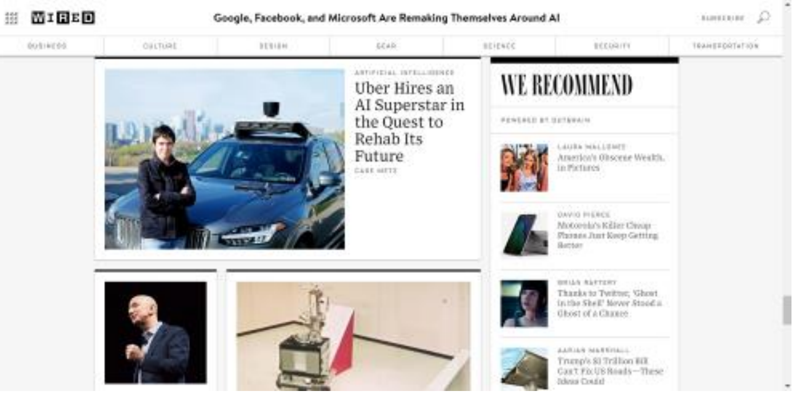
Successful applications of ML in practice (5)

- System security
  - Computer virus detection, Network intrusion detection, Spam email filtering,…

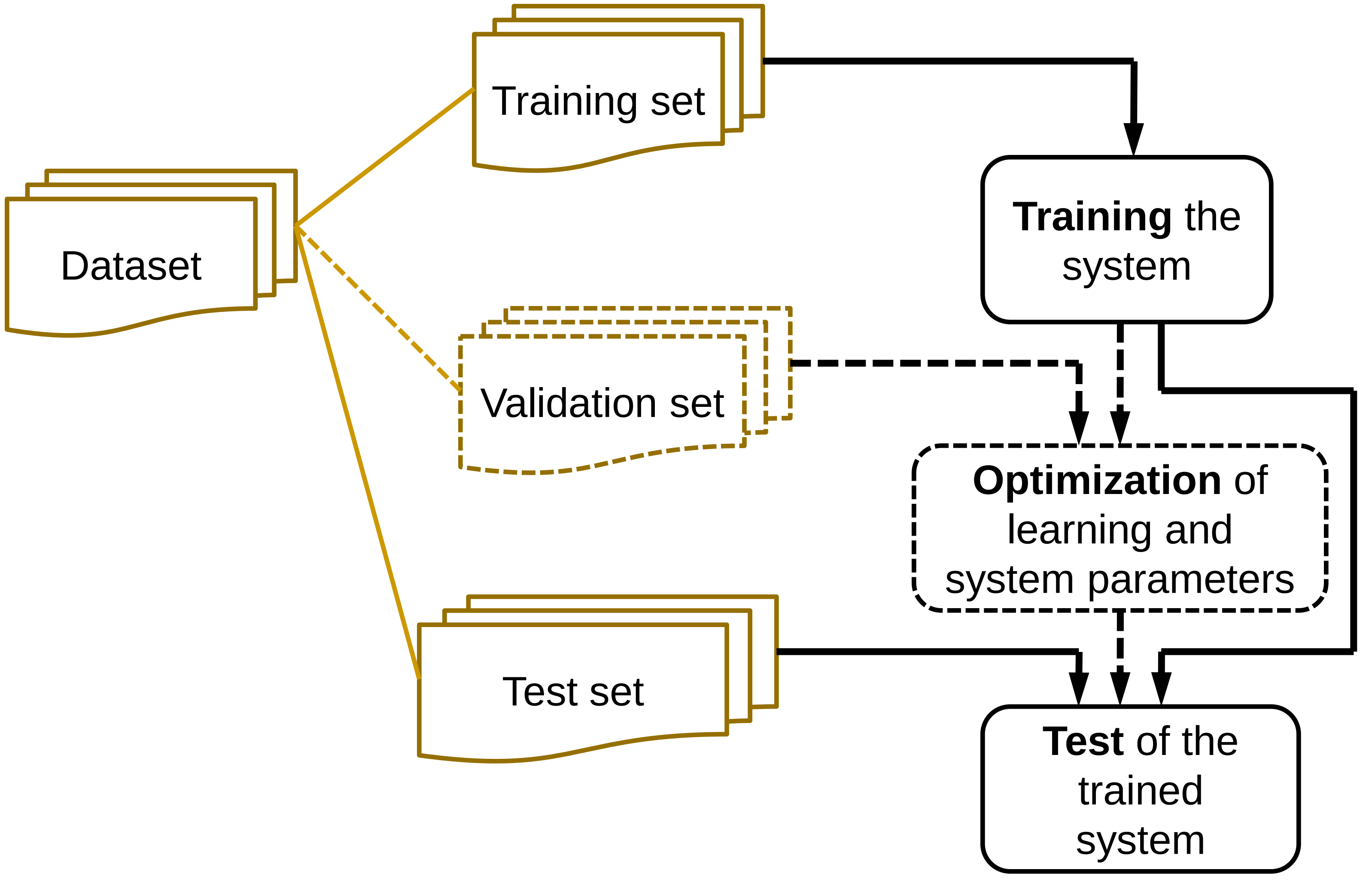


Successful applications of ML in practice (6)

- Marketing and advertisement



Machine learning process



Main elements of ML problem (1)

- Training (learning) examples
  - The training feedback is included in training examples or indirectly provided (e.g., from the working environment)
  - They are supervised or unsupervised training examples
  - The training examples should be compatible with (i.e., representative for) the future test examples
- The target function to be learned
  - F: $X \rightarrow \{0,1\}$
  - F: $X \rightarrow$ A set of class labels
  - F: $X \rightarrow R^{+}$ (i.e., a domain of positive real values)
  - …

Main elements of ML problem (2)

- Representation of the target function to be learned
  - A polynomial function
  - A set of rules
  - A decision tree
  - An artificial neural network
  - …
- ML algorithm that can learn *approximately* the target function
  - Regression-based
  - Rule induction
  - Decision tree learning (e.g., ID3 or C4.5)
  - Back-propagation
  - …

Challenges in ML (1)

- Learning algorithm
  - Which learning algorithms can learn approximately a given target function?
  - Under which conditions, a selected learning algorithm converges (approximately) the target function?
  - For a specific application problem and a specific example (object) representation, which learning algorithm performs best?

Challenges in ML (2)

- Training examples
  - How many training examples are enough for the training?
  - How does the size of the training set (i.e., the number of training examples) affect the accuracy of the learned target function?
  - How do error (noise) and/or missing-value examples affect the accuracy?

Challenges in ML(3)

- Learning process
  - What is the best ways of use order of training examples?
  - How does the order of using training examples vary the complexity of the ML problem?
  - How does the application problem-specific knowledge (apart from the training examples) contribute to the machine learning process?

Challenges in ML (4)

- Learning capability
  - Which target function the system should learn?
    - Representation of the target function: Representation capability (e.g., linear / non-linear function) vs. Complexity of the learning algorithm and learning process
  - The theorical limits for the learning capability of learning algorithms?
  - The system's capability of generalization from the training examples?
    - **Under-fitting** problem
    - **Over-fitting** problem
  - The system's capability of self-adapting its internal architectural representation?
    - To improve the system's capability of representation and learning of the target function

Challenges in ML (5)

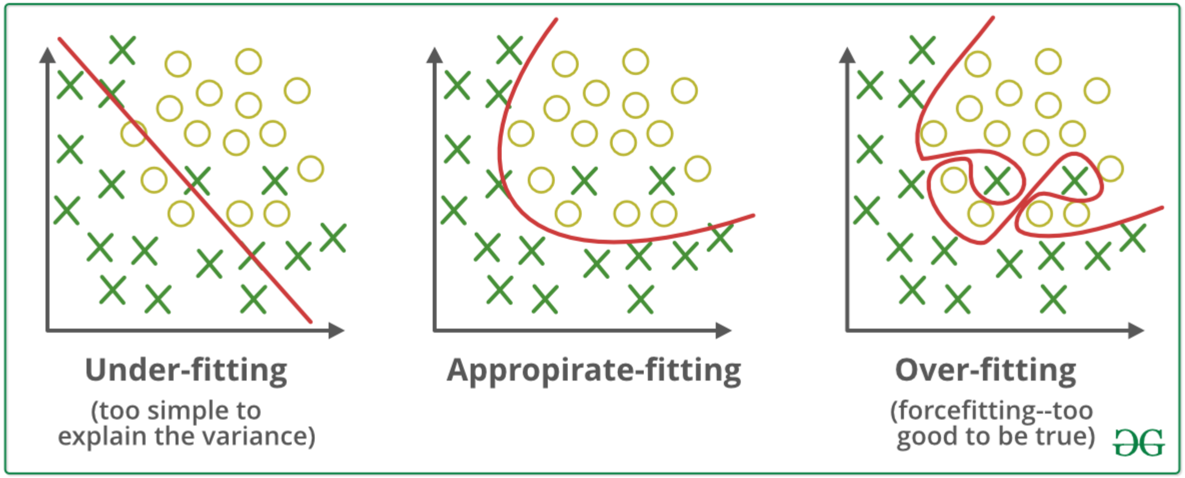
- **WHEN** should a trained model be re-trained?
  - The trained model has performed well on the past examples
  - But at a certain time, the trained model performs <u>significantly poor</u> on the newly coming examples
- **HOW** should a trained model be re-trained?
  - To adapt to the newly coming examples

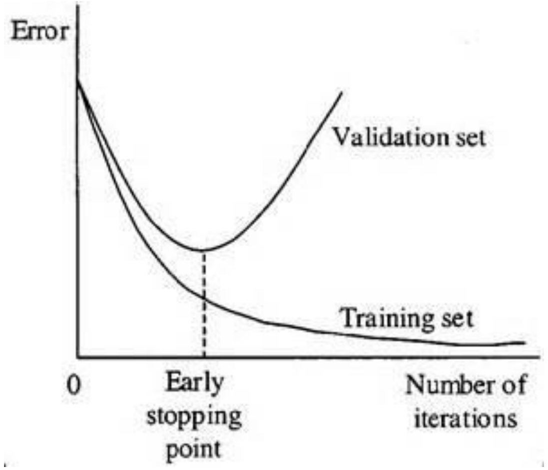
Generalization capability (1)

- Generalization shows the ability of the model to still achieve high accuracy <u>for future (unseen) data</u>
  - <u>Note</u>: We cannot use any test examples during model selection/training!
  - Use the **validation set** (often extracted from (as a small part of) the original training set) to serve as unseen data in the model training/selection
    - <u>Assumption</u>: The data characteristics are similar between the validation and test sets!

Generalization capability (2)

- 2 common (and should be avoided!) problems of generalization:
  - **Under-fitting**: Achieve low accuracy on all the training, validation and test sets
    - Often make false conclusions (i.e., the "*high bias*" characteristic)
  - **Over-fitting**: Achieve high accuracy on the training set, but low accuracy on the validation and test sets
    - Tend to make different conclusions for the same (or rather similar) examples (i.e., the "*high variance*" characteristic)





(https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf)

Problem of over-fit learning (1)

- A learned target function *h* is considered **over-fit** to a specific training set if there exists another target function *h'* such that:
  - *h'* produces lower accuracy than *h* for the training set, but
  - *h'* produces higher accuracy than *h* for the whole dataset (including also those examples that are evaluated after the training process)

Problem of over-fit learning (2)

- Assume that `D` is the whole dataset, and `D_train` the training set
- Assume that $Err_D(h)$ is the error caused by the target function `h` on `D`, and $Err_{D\_train}(h)$ is the error caused by the target function `h` on `D_train`
- The target function `h` is over-fit to `D_train` if there exists another target function `h'`:
    - $Err_{D\_train}(h) < Err_{D\_train}(h')$, and
    - $Err_D(h) > Err_D(h')$

Problem of over-fit learning (3)

- The problem of over-fit learning is often caused by:
  - Errors (noises) in the training set (i.e., by a collection/construction of the training set)
  - The number of training examples is too small, or not representative for the overall distribution of all the examples of the learning problem
  - The accuracy is too high/ideal (~100%) for the training set – The training process converges at a target function that is ideal/perfect for the training examples (<u>but not good for future/unseen examples</u>)

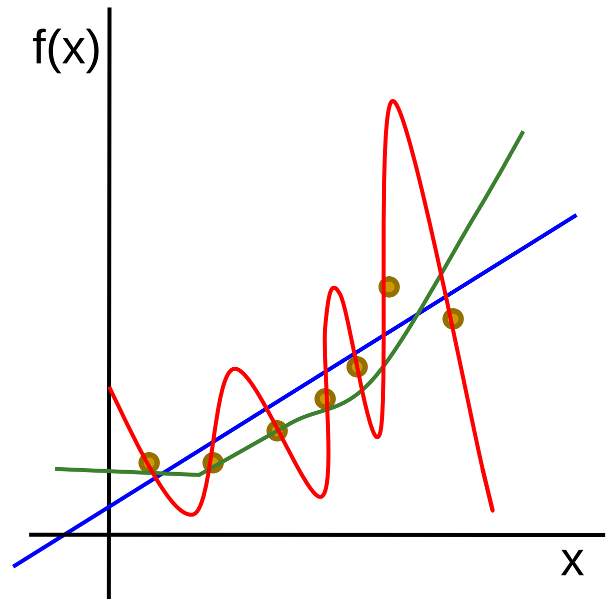
Problem of over-fit learning (4)

- Amongst those target functions learned, which one best generalizes from the training examples?

  <u>Important Note:</u> The goal of machine learning is to achieve <u>high accuracy in prediction for future examples</u>, not for the training ones

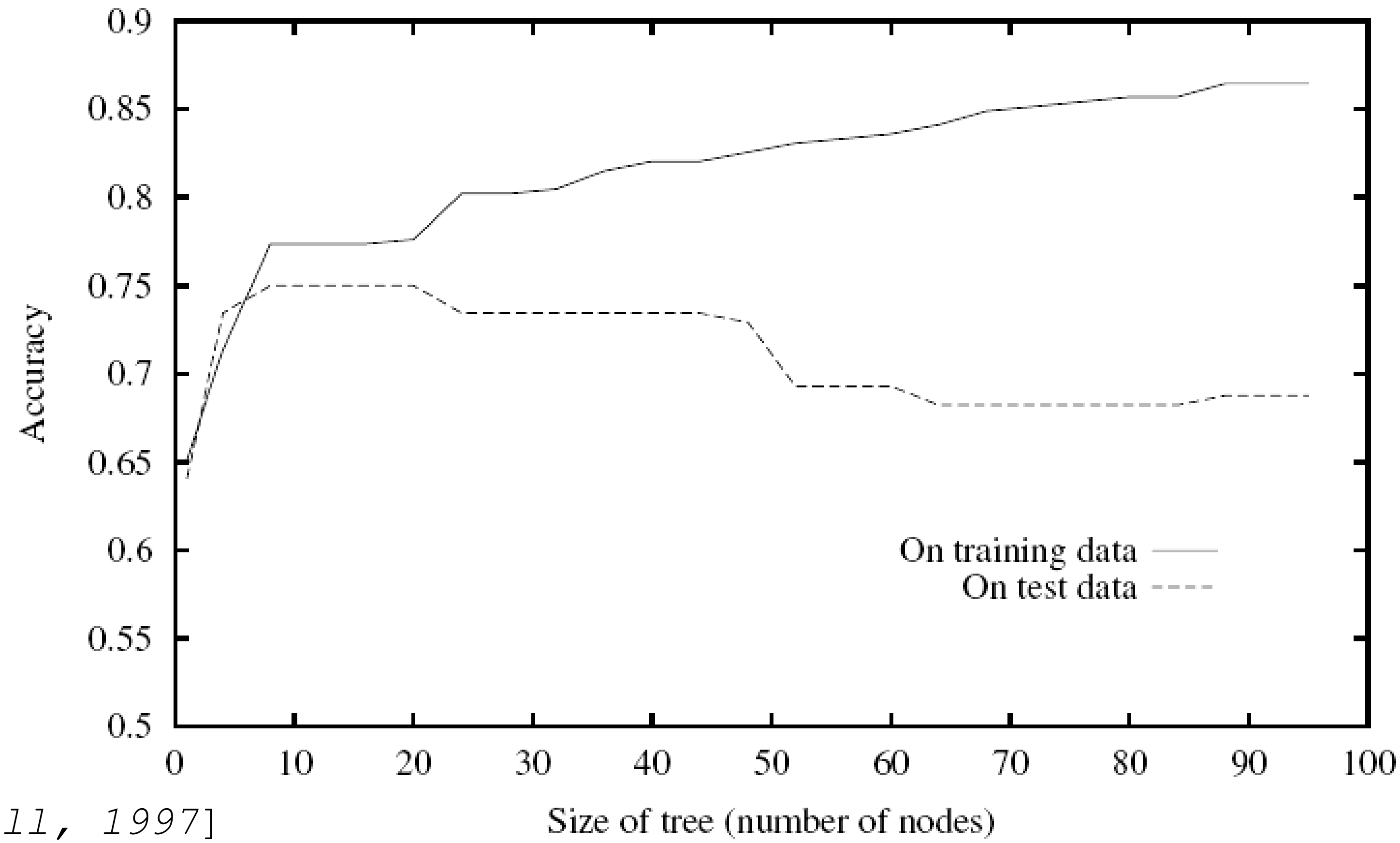
- **Occam's razor**: To select the <u>simplest suitable</u> target function (<u>not necessarily perfect</u>) for the training examples
  - → A better generalization
  - → Easier for explanation/interpretation
  - → Lower in computing cost

Which target function *f(x)* achieves a highest accuracy for <u>future</u> examples?



Example of over-fit learning

Continuing the Decision Tree learning process <u>decreases the accuracy on the test set</u> though <u>increases the accuracy on the training set</u>



[*Mitchell, 1997*]

Frameworks and tools for ML (1)

- **TensorFlow** (www.tensorflow.org)
  - OS: Linux, Mac OS, Windows, Android
  - Programming language: Python, C++, Java
- **Caffe** (caffe.berkeleyvision.org)
  - OS: Linux, Mac OS, Windows
  - Programming language: Python, Matlab
- **Caffe2** (caffe2.ai), **PyTorch** (pytorch.org)
  - On March, 2018, Caffe2 and PyTorch is merged into a single platform
  - OS: Linux, Mac OS, Windows, iOS, Android, Raspbian
  - Programming language: C++, Python
- **Keras** (keras.io)
  - OS: Linux, Mac OS, Windows
  - Programming language: Python
- **Theano** (deeplearning.net/software/Theano)
  - OS: Linux, Mac OS, Windows
  - Programming language: Python

Frameworks and tools for ML (2)

- **CNTK** (www.microsoft.com/en-us/research/product/ cognitive-toolkit/)
  - OS: Windows, Linux
  - Programming language: Python, C++, C#
- **Deeplearning4j** (deeplearning4j.org)
  - OS: Linux, Mac OS, Windows, Android
  - Programming language: Java, Scala, Clojure, Python
- **Apache Mahout** (mahout.apache.org)
  - OS: Any OSs with JVM installed
  - Programming language: Java, Scala
- **MLlib** of Apache Spark (https://spark.apache.org/mllib/)
  - OS: Any OSs with JVM installed
  - Programming language: Java, Python, Scala, R
- **Weka** (http://www.cs.waikato.ac.nz/ml/weka/)
  - OS: Any OSs with JVM installed
  - Programming language: Java

Online courses

- **Statistics-101** (provided by IBM)

  https://cognitiveclass.ai/courses/statistics-101
- **Machine Learning with Python** (provided by IBM)

  https://cognitiveclass.ai/courses/machine-learning-with-python
- **Machine Learning Foundations: A Case Study Approach** (provided by University of Washington)

  https://www.coursera.org/learn/ml-foundations
- **Machine Learning** (provided by Stanford University)

  https://www.coursera.org/learn/machine-learning
- **Predictive Analytics and Data Mining** (provided by University of Illinois at Urbana-Champaign)

  https://www.coursera.org/learn/predictive-analytics-data-mining
- **Data Mining Specialization** (provided by University of Illinois at Urbana-Champaign)

  https://www.coursera.org/specializations/data-mining

References

- E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.
- T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- H. A. Simon. *Why Should Machines Learn?* In R. S. Michalski, J. Carbonell, and T. M. Mitchell (Eds.): Machine learning: An artificial intelligence approach, chapter 2, pp. 25-38. Morgan Kaufmann, 1983.