Multimedia Retrieval

Chapter 1: Performance Evaluation

Dr. Roger Weber, roger.weber@gmail.com

1.1 Introduction

- In this course, we consider a number of retrieval models, feature extraction algorithms, and search algorithms. At some point, we need to understand the performance of an approach to determine the best way of searching. Often, there is no absolute answer what is the best method; instead the performance of a method varies from application to application:
  - Vector space retrieval was proven to outperform Boolean retrieval (similarly: probabilistic retrieval). Nevertheless, web search engine such as AltaVista (used vector space retrieval) and Inktomi (probabilistic retrieval) could not compete with Google (Boolean retrieval)
  - When searching for similar images (still photos), it is well accepted that color is more important than texture, and texture is more important than shape. In medical imagery, however, the contrary is true: color is often meaningless (X-ray, MRI, CT, ...), texture often plays an important role to detect the type of tissue, but shape is of highest importance (e.g., skin cancer).
  - Machine learning with Deep Neuronal Networks outperforms most other classification methods, but AI comes with high computational costs. Is the additional effort worth the better performance, or is “simpler & faster” just good enough, e.g., plain old web search or meta data search?
- We note that the performance of an approach depends on
  - the collection,
  - the type of queries / learning scenarios,
  - the information needs of users,
  - … and some non-functional constraints (e.g., costs, time, storage)

  With other words: for each retrieval and learning task, a new evaluation is required to determine the best approach. Generalization do work to a certain degree, but not always.

- For retrieval systems, **Precision** and **Recall** are the dominant measures. The core concept is based on un-ordered sets of documents but was extended to include also ranking (top rank is most important) and graded assessment (e.g., values between 0 and 3 with 3 being high relevant).
- Before we can evaluate measures, we need to define a benchmark evaluation approach. The evaluation should be stable with regard to incomplete knowledge of what is relevant / not-relevant for a given query. The measure should consider the context of the use case (student/web search: give me an answer quickly → top result; patent lawyer/property search: give me an exhaustive list of everything that is relevant for my query). The method should be simple to apply and should compare methods objectively with each other.

- The training and evaluation of learning methods depend on the desired task and output. For instance, the assessment of
  - binary classification is very similar to Boolean retrieval (precision, recall)
  - multi-class classification uses so-called confusion matrices to understand for which combinations of classes the algorithm performs good/bad
  - classification with scores and thresholds requires us to determine good thresholds and a metric to compare different methods (given different thresholds)
  - classification with probability distributions is often based on entropy (log-loss)
  - regression tasks (fitting real valued output data) uses mean squared error (MSE)
  - deep learning uses various methods to define what “good” means

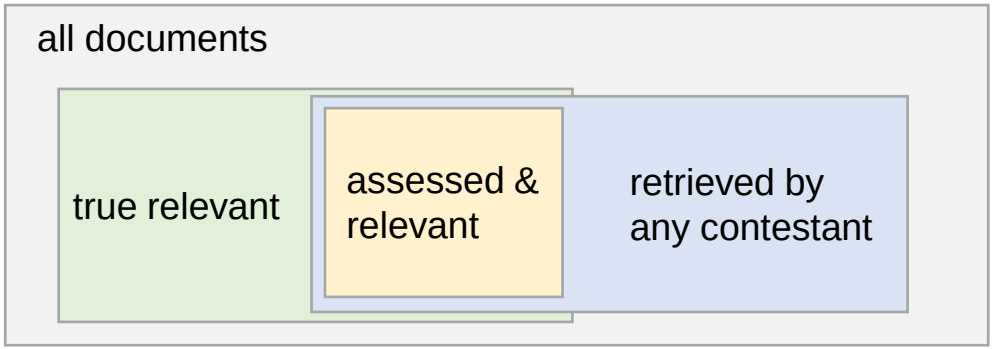
1.2 Defining a Benchmark for Retrieval

- So what makes a good benchmark? First of all, we need a sound collection that provides a diverse set of different documents matching the retrieval scenario. We also need queries (many of them) covering various aspects of the retrieval task, and we need an assessment of all documents against all queries. Finally, we need an evaluation method to capture the essence of “what is good”. In summary, a test collection forming a benchmark consists of
  - fixed set of documents (document can be anything)
  - fixed set of queries (typically split into train, dev-eval, and eval parts)
  - fixed set of assessments (not for all query-document combinations, can be binary or graded)
  - fixed method to evaluate performance (what means good? comparison of two mehtods

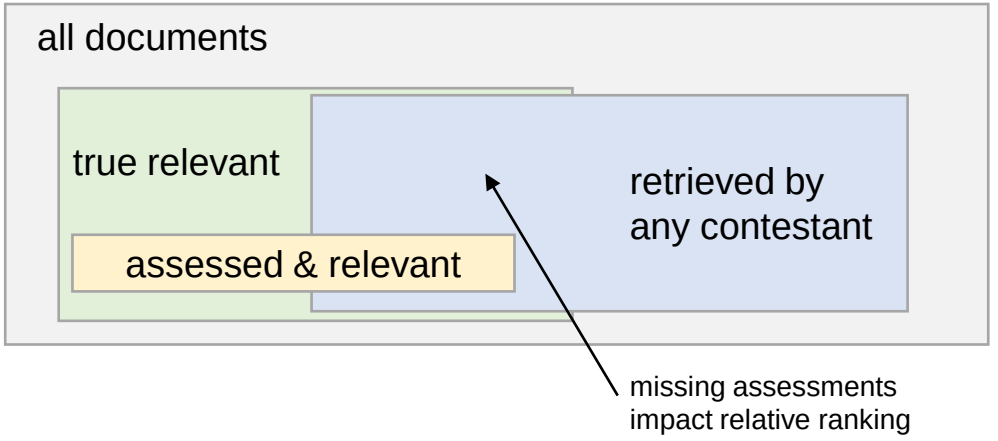
- Sparse vs. dense assessment
  - MS MARCO (Microsoft Machine Reading Comprehension Dataset) is an example for a **sparsely assessed benchmark**: MS selected over 500 thousands of queries from Bing (with high occurrence count to exclude personal data leakage) with millions of documents and passages but only about one relevant document is assessed per query
  - TREC (Text REtrieval Conference) data sets are examples for **densely assessed benchmarks**: much smaller corpus with 50+ queries and hundreds of assessments per query. Part of the benchmarking process works through “crowd-sourcing” of assessments, i.e., the contestants rate the results from the search engines to improve relative ranking

- Relative performance assessments of search engines based on precisions and recall measures (defined by the organizer of the benchmark)

- Dense assessments overcomes the impact of unknown / incomplete data but is not always possible
  - Many conferences organize the assessment as part of the contest
  - All documents retrieved by contestants are “crowd-assessed”
  - Although there are more relevant documents, the relative ranking does not change as no contestant found them
  - Only feasible for small sets of queries (50+) and assessments only updated during the course of the benchmark. Absolute relevance “unknown” and potential bias of contestants impacts ranking



- With very high data and query volumes, we must consider the impact of partial knowledge of “truth”
  - Contestants may return relevant documents that are wrongly considered as “unknown” / “not relevant”
  - Relative ranking between contestants only feasible within subset of assessed documents (but may look different if all relevant documents are considered)
  - Huge impact and bias on what is assessed and considered relevant



1.3 Boolean Retrieval

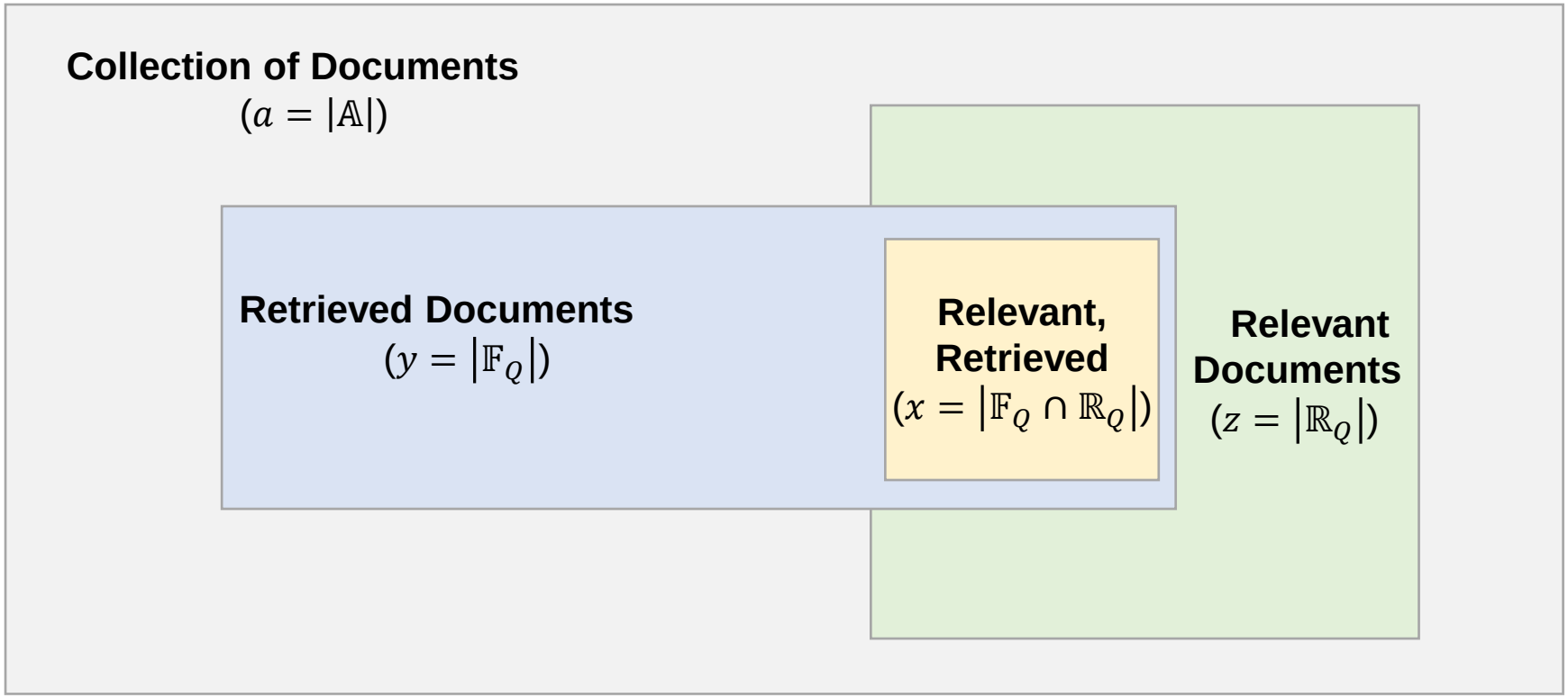
- **Precision** and **recall** are the earliest and still most important measures used for the evaluation of search algorithms. Precision denotes how many of the answers are relevant from a user's perspective. Recall describes the percentage of retrieved and relevant answers over all relevant documents in the collection. They form the key dimensions that covers the user's interests:
  - Precision drives the desire to have relevant document at the top of the list. A good search engine that provides only relevant documents at the top suits many "knowledge" or factual based queries (student, facts checker).
  - Recall supports the desire to overview a vast area of the result space. A good search engine must return as many relevant hits as possible to allow the user to explore and narrow down the area of interest (patent lawyer, search with very vague criteria)
- Notations:

$\mathbb{A}$	Set of all documents
$\mathbb{R}_Q$	Set of relevant documents for a query Q in the collection $\mathbb{A}$
$\mathbb{F}_Q$	Set of documents retrieved by a system for query Q

- Precision p, recall r are defined as follows:

$$p = \frac{\left|\mathbb{F}_Q \cap \mathbb{R}_Q\right|}{\left|\mathbb{F}_Q\right|} \qquad r = \frac{\left|\mathbb{F}_Q \cap \mathbb{R}_Q\right|}{\left|\mathbb{R}_Q\right|} \qquad f = \frac{\left|\mathbb{F}_Q \setminus \mathbb{R}_Q\right|}{\left|\mathbb{A} \setminus \mathbb{R}_Q\right|}$$

- Visualization



Precision: $p = \frac{x}{y}$ Recall: $r = \frac{x}{z}$

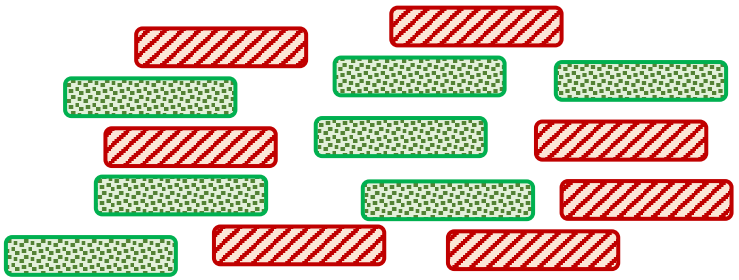
– **F-Measure:** Combines Precision and Recall to a single value. The parameter β determines how more important Recall over Precision shall be. With $\beta = 0$ only Precision counts; with $\beta = \infty$ only Recall counts.

$$F_\beta = \frac{(\beta^2 + 1) \cdot p \cdot r}{\beta^2 \cdot p + r}$$

The larger the F-Measure, the better an algorithm or system works. A typical value is $\beta = 1$. Having a single measure instead of two values simplifies comparisons; β is pushing either precision (need some relevant documents) or recall (need all relevant documents).

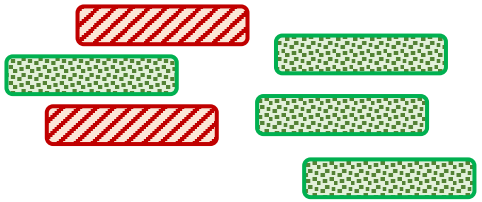
- Example: Comparing two methods (query has a total of 20 relevant documents)

Search Method A



$p = \frac{7}{14} = 50\%$ $r = \frac{7}{20} = 35\%$

Search Method B



$p = \frac{4}{6} = 67\%$ $r = \frac{4}{20} = 20\%$

relevant document non-relevant document

- Usually, we are not just using a single experiment to assess the performance of methods. Rather, we run a series of queries and then compute an “average” precision and recall. Let N be the number of queries, and for each query Q_i, we obtain a set $\mathbb{F}_i$ (retrieved documents for query Q_i) and a set $\mathbb{R}_i$ (relevant documents for query Q_i). For each query, we can compute the precision-recall pair (p_i, r_i). To obtain an average value, two method exist:
  - **Macro Evaluation**: p and r are given as average value over p_i and r_i , respectively:

$$p = \frac{1}{N}\sum_{i=1}^{N} p_i = \frac{1}{N}\sum_{i=1}^{N} \frac{|\mathbb{F}_i \cap \mathbb{R}_i|}{|\mathbb{F}_i|} \qquad r = \frac{1}{N}\sum_{i=1}^{N} r_i = \frac{1}{N}\sum_{i=1}^{N} \frac{|\mathbb{F}_i \cap \mathbb{R}_i|}{|\mathbb{R}_i|}$$

  - **Micro Evaluation**: summing up numerators and denominators leads to:

$$p = \frac{\sum_{i=1}^{N}|\mathbb{F}_i \cap \mathbb{R}_i|}{\sum_{i=1}^{N}|\mathbb{F}_i|} \qquad r = \frac{\sum_{i=1}^{N}|\mathbb{F}_i \cap \mathbb{R}_i|}{\sum_{i=1}^{N}|\mathbb{R}_i|}$$

  The micro evaluation is more stable if the sets $\mathbb{F}_i$ and $\mathbb{R}_i$ vary significantly in size.

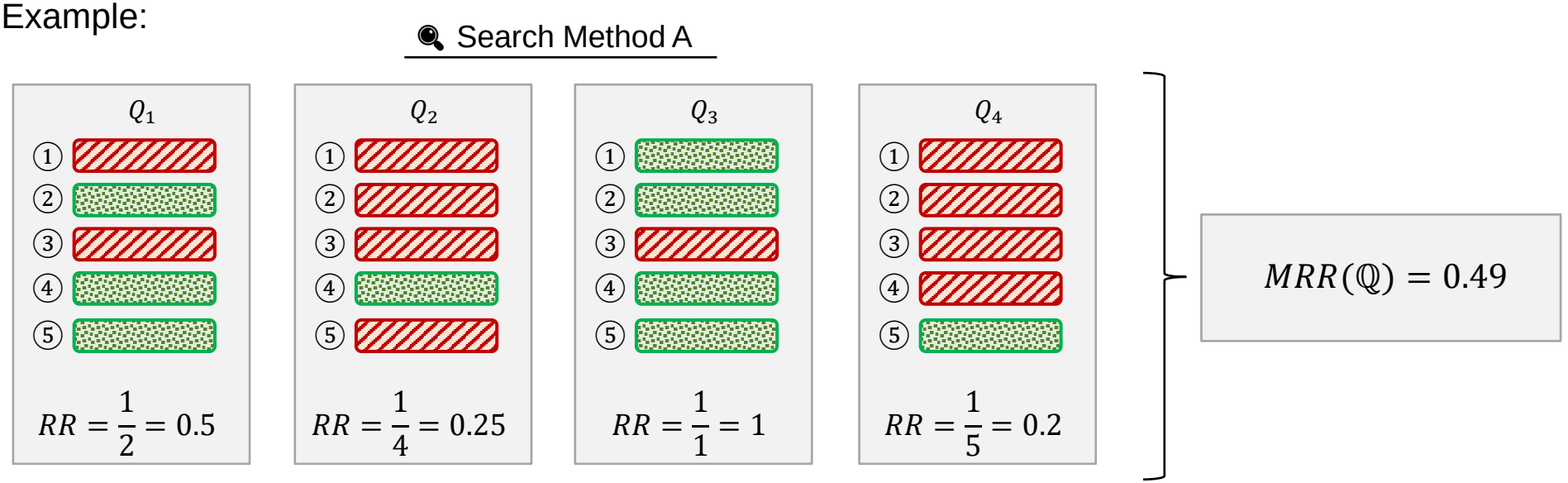
1.4 Retrieval with Ordering

- Precision and recall are easy to understand but they do not take the ranking of documents into account. So how can we take ranking into account?
- The **Mean Reciprocal Rank (MRR)** works with either very sparse assessments (only a few assessments per query available) or with users that only look at top results and stop searching once they found a relevant document. The definition for queries $Q_i \in \mathbb{Q}$ is:

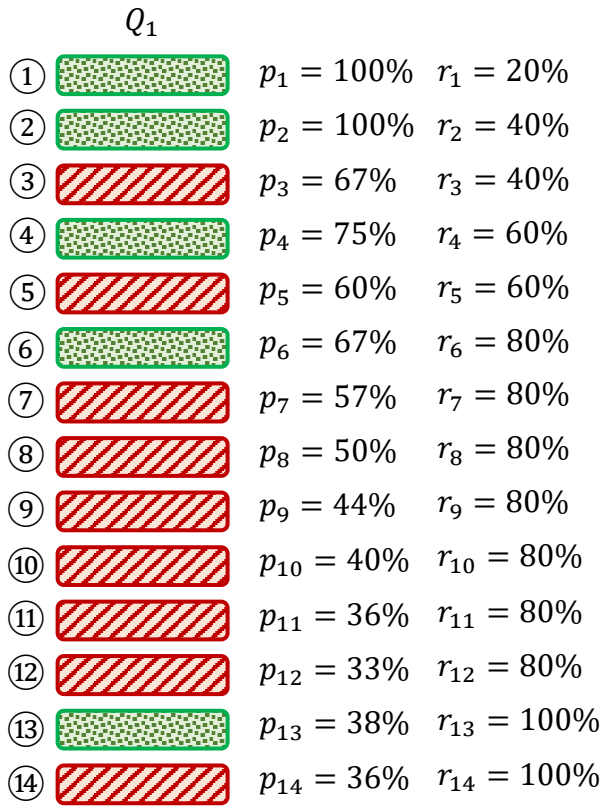
$$MRR(\mathbb{Q}) = \frac{1}{|\mathbb{Q}|} \sum_{Q_i \in \mathbb{Q}} \frac{1}{rank_i}$$

with $rank_i$ being the rank of the first relevant document for query Q_i. The reciprocal rank gives a very strong emphasis on the first position in the ranking and quickly converges to 0.

- Example:



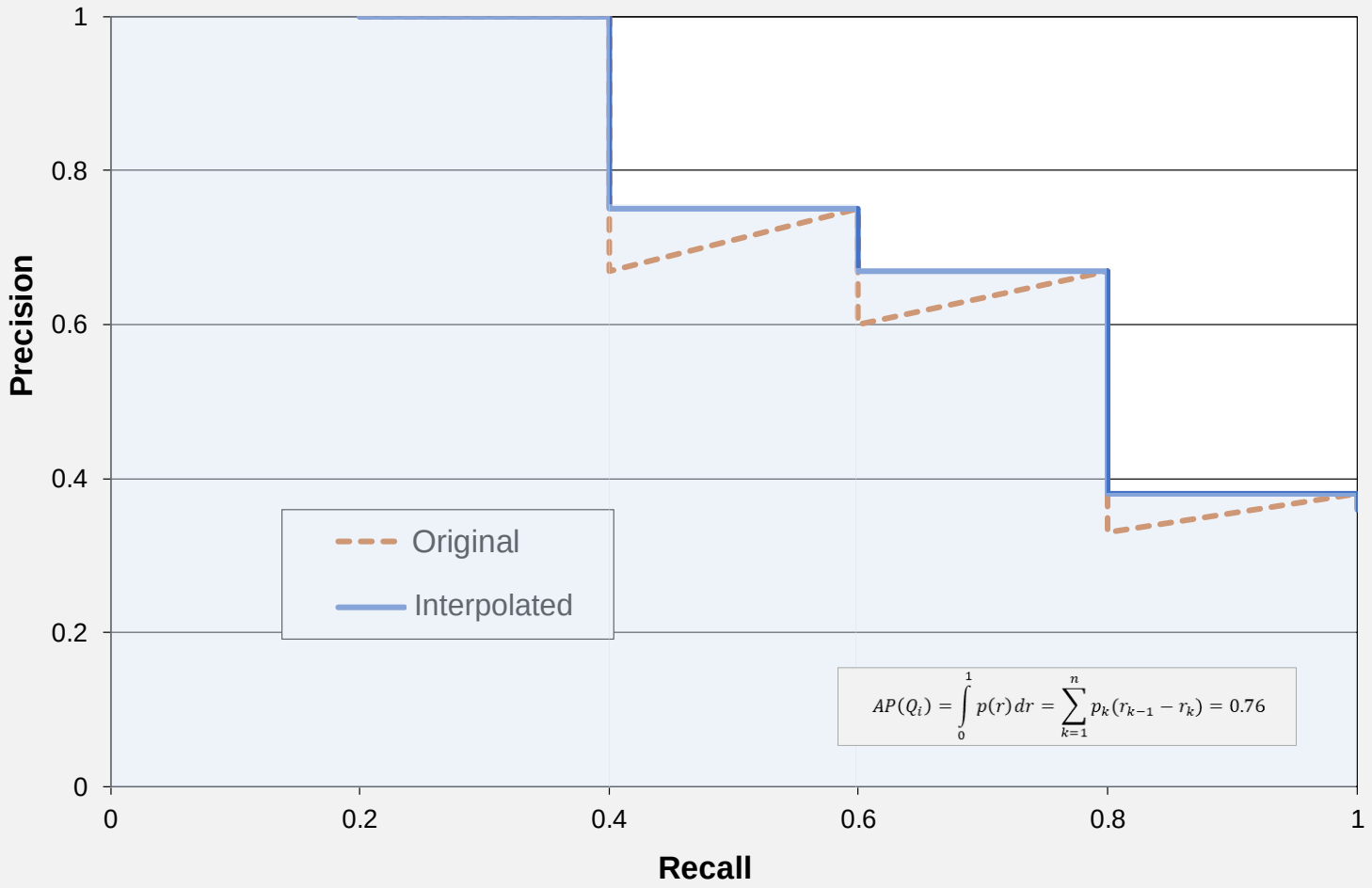
- The **Precision-Recall-Curve** extends the simple precision/recall measures for ranked lists:
  - for each rank, a precision-recall pair (PR-pair) is computed for the results up to this rank
  - average precision (AP) is the area underneath the (interpolated) graph of the PR-pairs
  - mean average precision (MAP) summarizes average precision for a set of queries
- Precision-recall pair at rank k is the precision and the recall value if we only look at the top-k results. As we walk down the ranked list, precision and recall increases with each relevant document, while only precision decreases with each non-relevant document.
- Example (total of 5 relevant documents)



$$AP(Q_i) = \int_0^1 p(r)dr = \sum_{k=1}^{n} p_k(r_k - r_{k-1}) = 0.76$$

$$MAP(\mathbb{Q}) = \frac{1}{|\mathbb{Q}|} \sum_{Q_i \in \mathbb{Q}} AP(Q_i)$$

- Below are the PR pairs of the example in a 2-dimensional plot. Notice that recall values only increase while precision values increase whenever a new relevant document is in the list, and decrease otherwise. To smooth the P-R curve, we interpolate the values to obtain a step curve as depicted below in blue. The area under the blue line is the average precision AP.



- Interpretation of P-R-Curve:
  - Close to $(r = 0, p = 1)$: retrieved documents are relevant but not all relevant documents found. Optimal for queries where just one correct answer is needed, e.g., *"is this mushroom poisonous"*
  - Close to $(r = 1, p = 0)$: all relevant documents retrieved but many retrieved document are non-relevant. High recall is important for queries like *"is there a patent"*
  - $p = 1$ is usually difficult to achieve; $r = 1$ is simple—just return all documents

- Next to the average precision and the mean average precision, further simple measures exist:
  - **Precision at k (P@k):** computes the precision for the top-k results and ignores recall. Typically used in scenarios with thousands/millions of relevant documents exist or if we don't know how many relevant documents exist (recall becomes very small or cannot be computed correctly)
  - **System Efficiency:** prefers an ideal system that returns all relevant and only relevant documents. That is, we prefer both high precision and high recall values. In the precision-recall plot, if the curve of a method A lies closer to the point $(r = 1, p = 1)$ than the curve of a method B, then we consider A to outperform B. Let d be the minimal distance of the precision-recall pairs to $(r = 1, p = 1)$. The system efficiency E is then given as:

$$E = 1 - \frac{d}{\sqrt{2}}$$

  - **R-Precision:** if we favor precision over recall, the R-Precision is a good alternative. It denotes the precision of a method after having retrieved a given percentage of all relevant documents:

$$RP = \max_{p,r} \begin{cases} 0 & if\ r < r_{threshold} \\ p & if\ r \geq r_{threshold} \end{cases}$$

1.5 Machine Learning Basics

- The Machine Learning Problem

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E [Mitchell 1997]

- There is a wide variety of machine learning problems as a combination of what the task is, what experience is provided and how performance is measured.
- Often, real-life examples employ a set of different approaches and combine them to achieve the overall objective. For instance, in credit card fraud, the first component is to learn fraudulent transactions based on past transactions and investigations. This knowledge is used to predict fraud in real-time for new transactions. A second component segments transactions to identify outliers or anomalies that may lead to new types of fraud that have not been identified/learned yet. While the first component is an example for supervised learning where the algorithms get labeled data to learn from, the second component is unsupervised, i.e., we don't know what we are looking for and the algorithm must identify the patterns without any human interaction or feedback.
- Other examples include cascading several methods: for instance, a first step reduces dimensionality and eliminates outliers (unsupervised learning), a second step learns the mapping of reduced features to a set of labels (supervised learning).
- Modern approaches in Deep Learning build excessively deep sequences with neuronal networks to apply multiple different approaches to extents that require vast amounts of compute power to train and then to use the network.

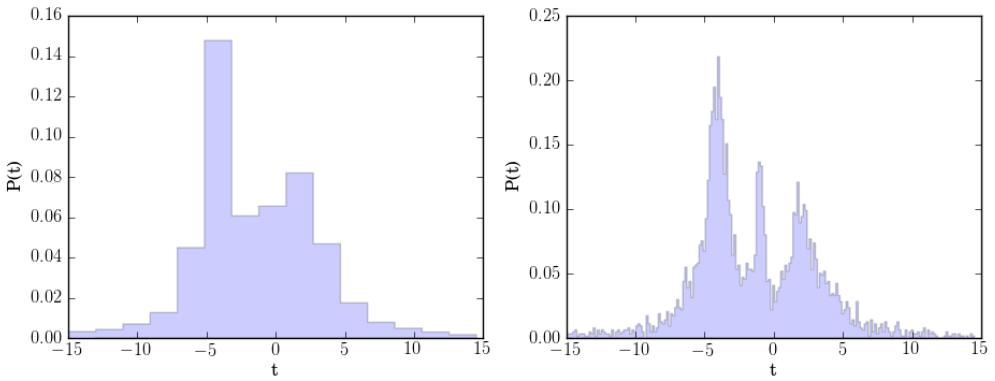
1.5.1 Tasks

- With task, we do not mean the learning process itself. Rather the ability that the machine is supposed to perform. For instance, if we want a car to drive autonomously, then driving is the task. Often, machine learning tasks involve a set of input features that the system needs to process into a "correct" set of output features.
- **Classification** is the task of mapping the input features to a set of K categories. Typically this means to find a function f that maps a M-dimensional vector $\boldsymbol{x}$ to a category represented by a numeric value y, i.e., $y = f(\boldsymbol{x})$ with $f\colon \mathbb{R}^M \rightarrow \{1, \ldots, K\}$. A variant of the classification task requires a probability distribution $P(y)$ over all classes y with $P(y) = 1$ denoting the class y is certain and $P(y) = 0$ denoting the class y is impossible, i.e., $P(y) = f(\boldsymbol{x})$ with $f\colon \mathbb{R}^M \rightarrow [0,1]^K$
  - Applications include object recognition in images, text categorization, spam filtering, handwriting and speech recognition, credit scoring, pattern recognition, and many more

Sample	fixed acidity	volatile acidity	citric acid	pH	alcohol	quality
#1	8.5	0.28	0.56	3.3	10.5	7
#2	8.1	0.56	0.28	3.11	9.3	5
#3	7.4	0.59	0.08	3.38	9	4
#4	7.9	0.32	0.51	3.04	9.2	6
#5	8.9	0.22	0.48	3.39	9.4	6

- **Classification with missing input** is similar to classification with the exception that some input values can be missing. Instead of a single function f, a set of functions is needed to map different subsets of inputs to a category y (or distribution $P(y)$), potentially 2^M functions. A better way is to learn the probability distributions over all relevant features and to marginalize out the missing ones. All tasks have a generalization with missing inputs.

- **Regression** is the task of predicting a numerical value given the input features. The learning algorithm must find a function f that maps an M-dimensional vector $\boldsymbol{x}$ to a numeric value, i.e. $f: \mathbb{R}^M \rightarrow \mathbb{R}$. The difference to classification is the output: instead of a category, a real number is required. Also, regression does not deliver distribution functions over all possible values.
  - Applications: predictions / extrapolations to the future, statistical analysis, algorithmic trading, expected claim (insurance), risk assessment (financial), cost restrictions, budgeting, data mining, pricing (and impact on sales), correlation analysis
- **Clustering** divides a set of inputs into groups. Unlike in classification, the groups (and the number of groups) are not known beforehand and the machine learning algorithm must find them. As the output is not known at training time, this type of task is called “unsupervised” while the ones before are “supervised” (we told the machine in the examples before what outputs we expect).
  - Applications: human genetic clustering, market segmentation (groups of customers), social network analysis (communities), image segmentation, anomaly detection, crime analysis
- **Density estimation (probability mass function estimation)** is the construction of an estimate of an underlying, unknown probability density function given the input features. In the most simple case, the algorithm must learn a function p: $\mathbb{R}^M \rightarrow \mathbb{R}$ where $p(x)$ is interpreted as a probability density function (if $\boldsymbol{x}$ is discrete p is called probability mass function). The most basic form is shown in the example on the right with histogram based density estimation using two different numbers of bins.
  - Applications: age at death for countries, modelling of complex patterns, feature extraction, simplification of models

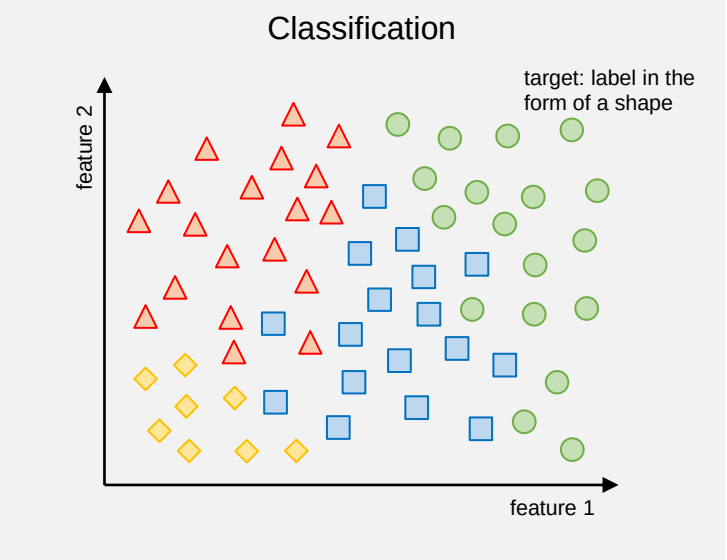


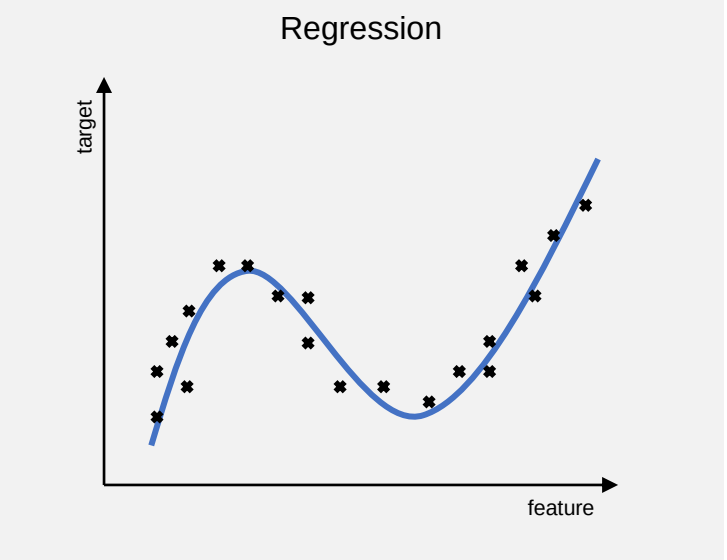
1.5.2 Performance

- To evaluate (and improve) a machine learning algorithm, we need to provide a quantitative measure for the "accuracy" of carrying out the task T. We will look deeper into these methods later in this chapter following this machine learning introduction. A short summary:
  - **Binary classification** (0-1 decisions) uses a **confusion matrix** to assess the performance, and provides numeric summary values to optimize for a desired optimum for the task. Typical measures include precision, accuracy and so on.
  - **Multi-class classification** (one out of a set of classes) requires a generalized **confusion matrix** resulting in a table with pair-wise "confusion". Accuracy still works fine; in addition, we can summarize performance of a single class against all other classes.
  - **Binary classification with scores and thresholds** is a simple extension of the confusion matrix. With increasing threshold values, we obtain a method to optimize the threshold (adjustment of a hyper-parameter), and the Receiver Operating Characteristic Curve (ROC Curve). The area under the ROC curve is a simple method to assess performance.
  - **Multi-class Classification with Probabilities** measures the performance based on the probabilities of the class labels of an object. Typically, this is based on cross-entropy with the log-loss measure being a simpler version of it.
  - With **Regression** tasks, we measure the performance as the **mean squared error (MSE)** between the actual values and the predicted ones.
  - As we will see, machine learning algorithms not only use these measures to evaluate performance but also employ them to find an optimal set of parameters to minimize the error/loss function. In addition, it can also be used to control so-called hyper-parameters (→ learning process).

1.5.3 Experience

- **Supervised Learning** algorithms observe a data set with features and a target for each instance of the data set. The goal is to learn a general rule that maps features to targets and that can be applied to predict the outcome of newly presented data items. The term “supervised” originates from the view that the target is provided by an instructor or teacher. As an example, classification tasks presents for each example, described as a set of feature, a target in the form of a label (or set of labels). The “teacher” instructs the algorithm how the sets of features are correctly mapped to labels and the algorithm should learn the mapping rule.

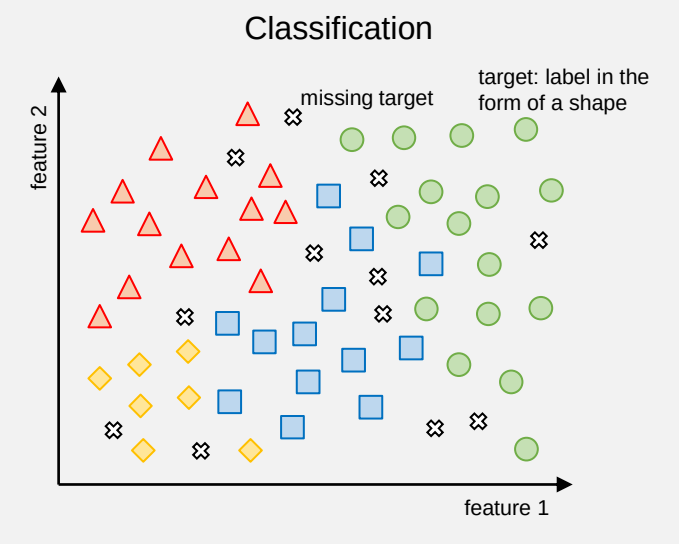




  - The teacher also provides an error measure that allows the machine learning algorithm to assess accuracy during training sessions
  - Even though targets are given, the algorithm must be able to deal with noise in the output values due to human errors (wrong labelling) or sensor errors (defects, distortion)

- **Semi-Supervised Learning** is a special case of supervised learning. The algorithm is presented with features and targets, however, some features or targets are missing (incomplete observation) in the training data. Depending on the task, the algorithm must either complete the missing features or predict targets for newly presented data sets.
  - **Missing targets:** The training set consists of complete features but some objects do not have targets (or labels). Incomplete targets often result if the labeling process is expensive or labor intensive. Consider a data set for credit card fraud detection with billions of transactions. Naturally, credit card firms investigate only a small subset of "suspicious" transactions and label them based on the outcome of an investigation ("fraud", "no fraud"). The vast amount is not labeled. To learn from such data sets, algorithms make one of the following assumptions:
    1) Smoothness: points in close proximity share the same label, i.e., the distribution function is continuous
    2) Cluster: data tends to form clusters and all objects in the same cluster share the same label
    3) Manifold: often, features are high-dimensional but there are only a few labels. Hence, the data is more likely to lie on a low dimensional manifold

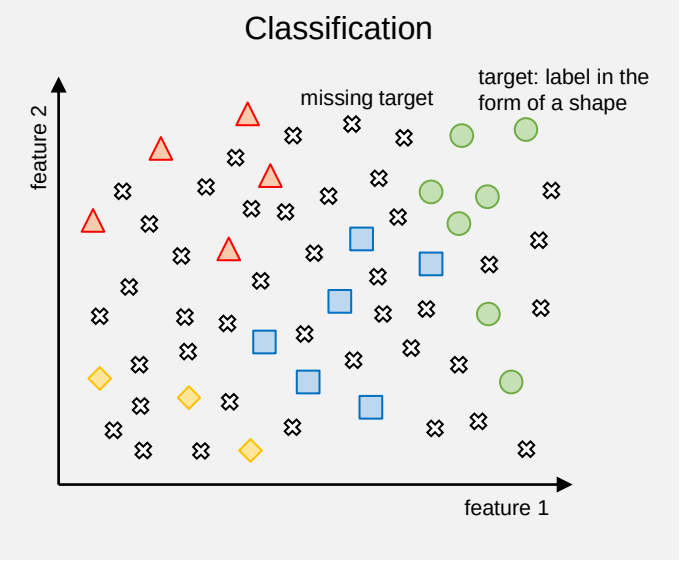
    Semi-supervised learning takes ideas both from supervised learning and from unsupervised learning.



- **Induction:** if only a few labels are missing, a good strategy is to learn the distribution from the labeled data items with a supervised learning method. We can then go back and predict the missing labels. However, this does not work well if lots of objects have no label as the training set is not sufficient to capture the true distribution of labels. Evidently, such training ignores most of the data (information loss).

- **Transduction:** to consider all data points, transductive algorithms identify clusters in the data set and apply the same label to all objects in the cluster. A simple approach is the partitioning transduction:
  1. Start with a single cluster with all objects
  2. While a cluster has two objects with different labels Partition the cluster to resolve the conflict
  3. For all clusters: assign the same label to all objects in the cluster

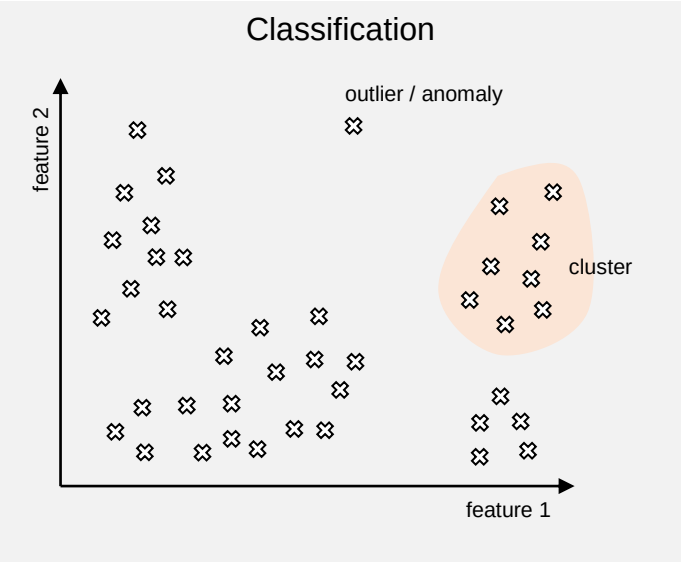
  There are other variants to develop the clusters.



– **Missing features:** The training set has complete targets, but some objects lack some of the features. For newly presented data, potentially with missing features, the algorithm must predict the target. A good example is disease prediction where the target (“healthy”, “has disease”) must be predicted from a set of test results. Laboratory tests are expensive and naturally not all features (test results) are available. Prediction of missing features depends on the AI method:

- Naïve Bayes (more details later in the deck) is a simple technique for building classifiers based on conditional probabilities. Let there be K classes C_k and M features x_i. The best class k^* is then given by $k^* = \underset{k}{\operatorname{argmax}} P(C_k) \prod_i P(x_i|C_k)$. The probabilities $P(C_k)$ and $P(x_i|C_k)$ are learned from the training data (ignoring missing features x_i). To predict the class for a new object with missing features, we simply ignore them in the Naïve Bayes optimization.
- If we have learned the distribution function over all features, we can simply “integrate” or “average” over the missing features, i.e., we assume that the missing features follow the distribution of the training set and we approximate them with an expected value.

- **Unsupervised Learning** algorithms observe a data set without targets and infer a function that captures the inherent structure and/or distribution of the data. In other words, we want to identify interesting facts in the data and derive new knowledge about its structure. In contrast to supervised learning, there is no instructor or teacher that provides targets or assess the performance of the outcome. The algorithm must learn without any guidance.



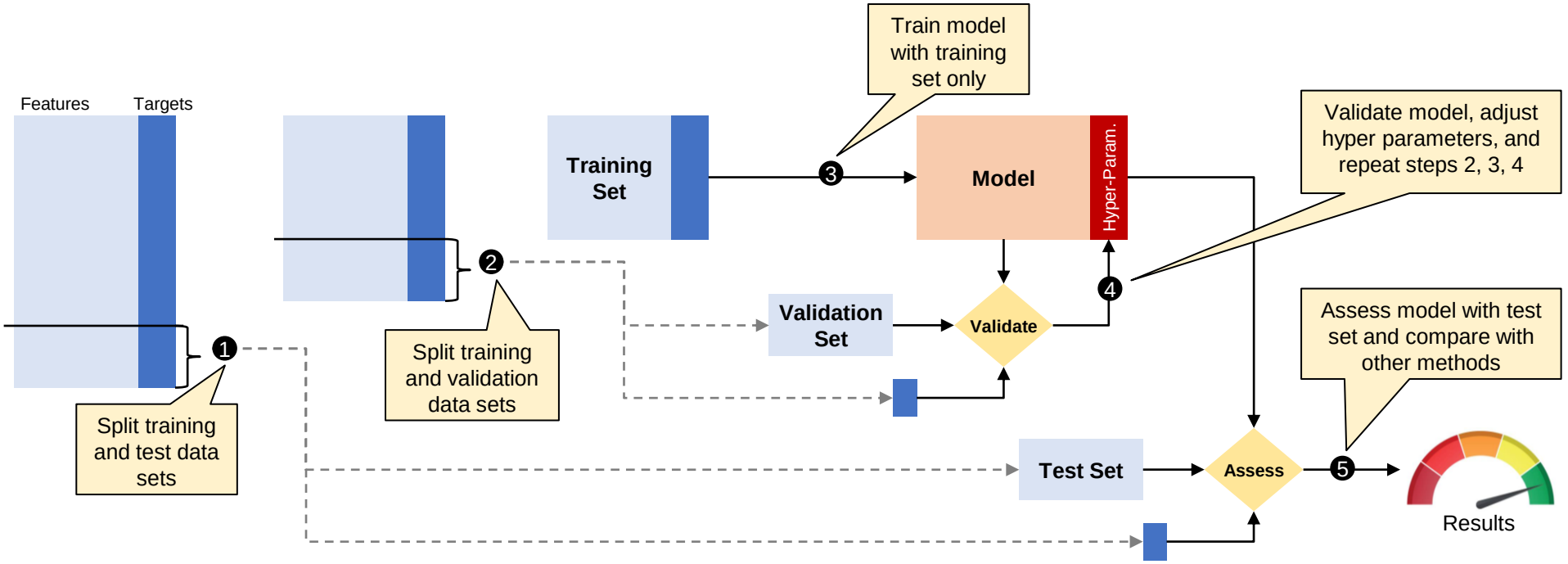
  - Clustering: the most common task for unsupervised learning is to identify groups of objects that “belong” together (with regard to a distance function). The number of clusters is often not known and must be learned too.
  - Outlier/Anomaly detection: the algorithm must learn the “normal” behavior through any means and identify outliers that significantly differ from the other objects. Note that the training data may also contain outliers.
  - Density function: describe the data set through an “appropriate” density function. A simple method is a Gaussian approximation and learning its mean value and variance from the data. More complex methods choose from a set of different distribution functions and optimize to the “best fit”
  - Dimensionality reduction: high-dimensional features often disguise an inherently much simpler characteristic of the data. Principle component analysis extracts “core concepts” along principal directions in the feature space that provide a simpler (but still accurate) view on the data.
  - Self-organizing maps (SOM): a SOM produces a discrete (often 2-dimensional) presentation of the data in a mesh of nodes, thereby mapping high-dimensional data to a low-dimensional view. It uses a competitive learning approach.

1.6 The Machine Learning Process

- Machine learning algorithm learn from data. It is critical that we feed the “right” data into this process for the task that we want it to solve. “Right” is not only referring to good data quality, complete data, but also the extraction of meaningful features. A number of challenges arises in this context:
  - Feature selection, i.e., ability to capture essential information to learn a task
  - Data cleansing, i.e., ability to remove the negative impact of outliers or of noise
  - Normalization, i.e., ability to address correlation between features and to normalize scales
  - Curse of dimensionality, i.e., inability to learn underlying structure due to sparse data space
  - Overfitting, i.e., inability to generalize well from training data to new data sets
  - Underfitting, i.e., inability of the algorithm to capture the true essence of the data structure

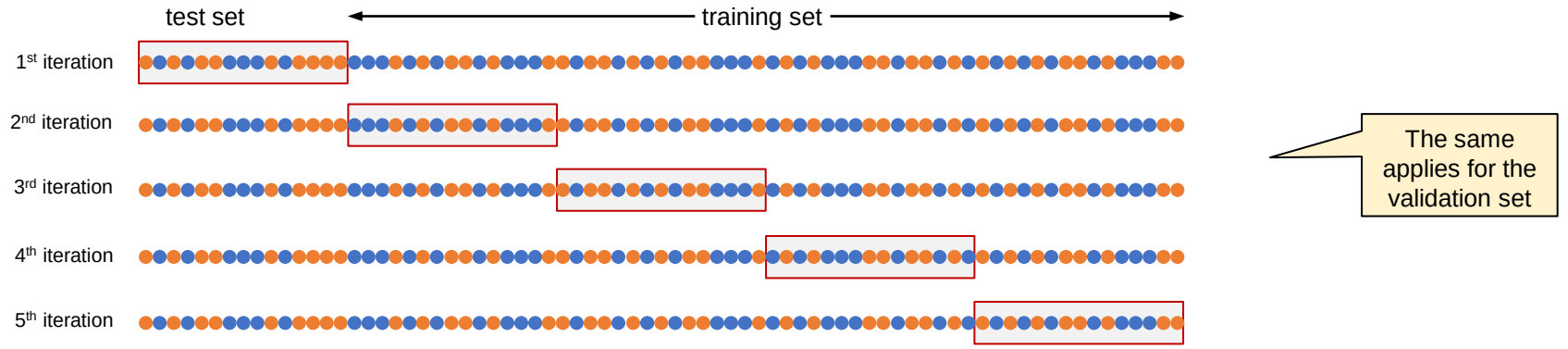
- Data preparation is a 3-step approach which we do not further discuss in this section. With the term “data” we always include features and targets (if they are available)
  1) Select Data
  2) Preprocess Data
  3) Transform Data

- We need to pay attention how we divide the data sets into training sets, validations sets, and test sets. The latter aspects is essential to adjust hyper-parameter of the algorithm including capacity and to measure its ability to correctly generalize. In the following, we focus on the overall learning process and address the above overfitting and underfitting issues.



- To understand how well a machine learning algorithms can generalize to new data sets, it is essential that training sets and test sets are distinct. Otherwise, we can construct a memorizing algorithm that simply stores all features and targets. Assessments of such an algorithm will produce the best possible results, but the algorithm will perform poorly on new data.
- Most algorithms have models with so-called hyper parameters that drive their inherent **capacity** or structure. For example, we can vary the degree of a polynomial regression model to adjust to a larger variety of functions. In a neural network, the capacity is provided by the number of neurons and connections. In a nutshell, models with small capacity struggle to fit the training data and to capture its distribution; models with high capacity tend to overfit the training data and poorly generalize to new data sets. The usage of validation sets (again, distinct from the training sets) allows algorithms to optimize their hyper-parameters.

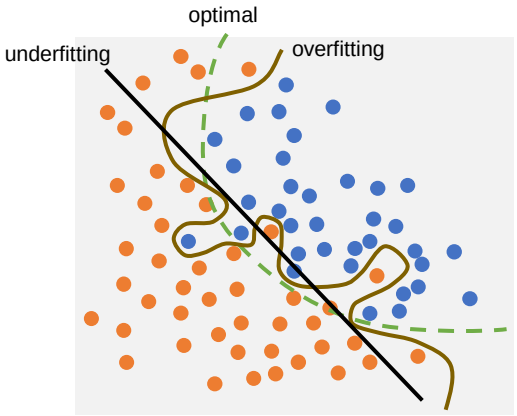
- To drive the learning process, we partition the original data set (and its targets) into a training set (70-80% of data) and test set (20-30% of data). If the model has need to optimize some hyper-parameters, we further partition the data to obtain the validation set (20-30% of data):
  - The **training set** is used for learning, i.e., to fit the parameters/weights minimizing training error
  - The **validation set** is used to tune hyperparameters (models, capacity) to prevent underfitting and overfitting issues. Validation data is not used for training and also not used for final testing
  - The **test set** is used to assess the performance, i.e., the ability of the model to generalize
- Ideally, the three data sets are large enough to represent the true distribution equally well. If the data set is too small, however, validation and testing lack statistical certainty on average errors making it difficult to assess and compare performance. **Cross-validation** uses rotation schemes an multiple iterations to improve the accuracy of validation and testing.
  - **k-fold cross validation** partitions the original data set into k equal sized subsamples. In each iteration, one subsample denotes the test set, and the remaining k-1 subsample form the training set. The k results are averaged to produce a single value. $k=10$ is a typical value. The same approach can be used for the validation set.

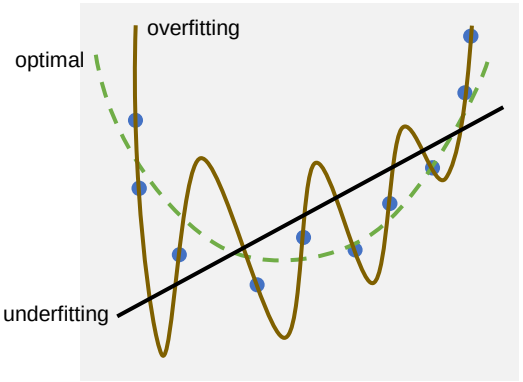


- What is the right complexity and the right capacity of a model to explain observed facts sufficiently?
  - Too simple: “if the sun is out, it is warm”
  - Too complex: “if the sun is out and it is a summer month and you are on the north side or it is a winter month and you are on the south side or you are equatorial or you are in a dessert and it is not an ice dessert and it is not cloudy or raining or snowing and there is not a strong wind and there is not a sun eclipse and there is not a volcano eruption and you are not in the water or in a cave or in the shadows or in a house with air conditioning or in a car with air conditioning or in a freezer … then it is warm”

- Our brain is excellent in finding the right level of abstraction despite the limitations of a simple model
  - Example: “birds can fly” (but wait, not all birds can fly)
    - we use a simple model to generalize (80-95%) and then deal with exceptions separately
  - Example: “describe what makes a chair a chair”
    - write down 3-5 attributes that describe how a chair must look like
    - compare with some examples from the web → no definition is good enough
    - look at counter examples: sofa, bank, anything else you can sit on
  - Example: “horse” → much narrower in terms of what is accepted to match the concept of a horse (e.g., donkey, zebra, giraffe and other hoofed animals do not count as horses)
  - Example: “dog” → wide variety of forms that count as dogs yet we recognize them immediately (different ratios of body lengths, colors, face features, tail, hair)
  - Example: “sketches of people”, comics and face features that make a person recognizable (cartoon, caricature of real people)

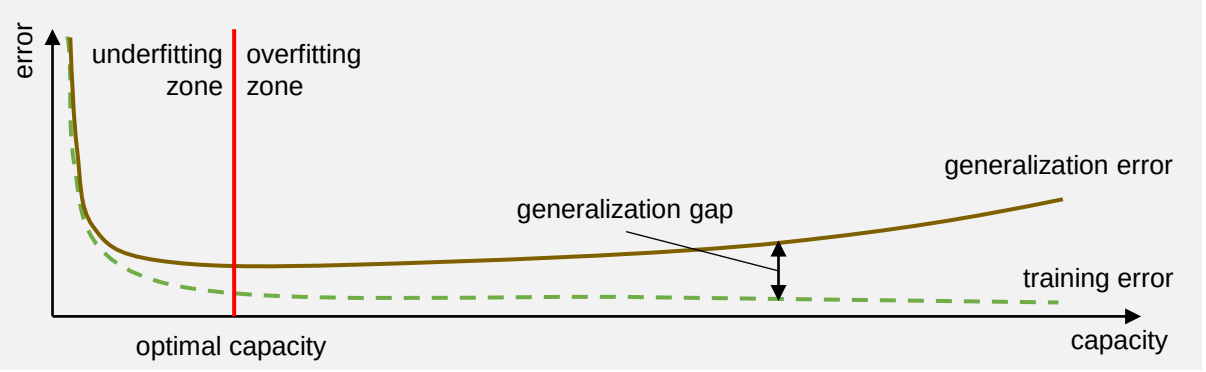
- **Overfitting** and **underfitting** are common problems in machine learning. Overfitting occurs when the model is excessively complex to match the training data as accurately as possible. Often, such a model has too many parameters relative to the number of training items. But even worse, the model is likely to overreact to minor changes leading to poor predictive performance (see figure on the right hand side as an example). Underfitting, on the other side, occurs when the model cannot capture the underlying trend of data and over-simplifies the distribution. For instance, fitting a linear model to a non-linear data distribution will result in a high training error and poor predictive performance.





  - As illustrated above, we can observe that overfitting is the result of optimizing for the training data with too many parameters. Typically, an overfitting model shows small errors indicating its ability to adapt nicely to the training data, but it can not predict new data points well enough.
  - Underfitting, on the other side, shows both large errors on the training data and poor prediction performance for new data points; it obviously cannot capture the true essence of the distribution.
  - We can control overfitting and underfitting by altering the **capacity of the model**. Optimal capacity is reached if the model exhibits small errors on both the training set and the validation set. To work best, training set and validation set must be distinct; but we can run several iterations to adjust the capacity with different partitioning of training and validation set.

- When altering the capacity of the model, **Occam's razor** provides an intuitive heuristic. The principle was first stated by William of Ockham (c. 1287-1347) and has been made more precise over time, most notably in the 20th century for statistical learning. The principle states:
  - Numquam ponenda est pluralitas sine necessitate [Plurality must never be posited without necessity]
  - In a more modern language, the principle states that among competing hypothesis that explain observations equally well, one should choose the "simplest" one
  - Indeed, simpler models are better able to generalize but we must choose a sufficiently complex model to achieve low training error. Typically, training error decreases gradually as capacity increases. The generalization error, however, has a U-shaped curve as a function of capacity:



- The **bias-variance tradeoff** (or **dilemma**) is the problem of simultaneously minimizing two sources of errors that prevent models to generalize well beyond their training data
  - The bias is the test error of a model causing it to miss relevant relations in the data (underfitting)
  - The variance is the error from sensitivity to small changes in the input. High variance can cause the model to adopt to noise in the training data rather than to the data (overfitting)

  The **bias-variance decomposition** is a way to analyze the expected generalization error. It uses the sum of the bias, variance, and irreducible error (noise) in the problem.

1.7 Perfomance of Machine Learning

- **Binary classification** (0-1 decisions) uses a **confusion matrix** to assess the performance, and provides numeric summary values to optimize for a desired optimum for the task

		Actual Condition (as observed)			
	Population	Positive (P)	Negative (N)		
Predicted Condition (as computed)	"Yes"	**True Positive** (TP)	**False Positive** (FP)	Positive Predictive Value (PPV), Precision	False Discovery Rate (FDR)
	"No"	**False Negative** (FN)	**True Negative** (TN)	False Omission Rate (FOR)	Negative Predictive Value (NPV)
		True Positive Rate (TPR), Sensitivity, Recall, Hit Rate	False Positive Rate (FPR), Fall-Out	Accuracy (ACC)	
		False Negative Rate (FNR), Miss Rate	True Negative Rate (TNR), Specificity		Error Rate (ERR), Misclassification Rate

$$TPR = \frac{TP}{P} \qquad FNR = \frac{FN}{P} = 1 - TPR \qquad ACC = \frac{TP + TN}{P + N}$$

$$TNR = \frac{TN}{N} \qquad FPR = \frac{FP}{N} = 1 - TNR \qquad ERR = \frac{FP + FN}{P + N} = 1 - ACC$$

$$PPV = \frac{TP}{TP + FP} \qquad FDR = \frac{FP}{FP + TP} = 1 - PPV$$

$$NPV = \frac{TN}{TN + FN} \qquad FOR = \frac{FN}{FN + TN} = 1 - NPV$$

– Example: Cancer test

	Population (*2030*)	Actual Condition (as observed): Positive (*P=30*)	Actual Condition (as observed): Negative (*N=2000*)	
Predicted Condition (as computed)	"Yes" (*200*)	**True Positive** (*TP=20*)	**False Positive** (*FP=180*)	$PPV = \frac{20}{200} = 10\%$ (precision)
	"No" (*1830*)	**False Negative** (*FN=10*)	**True Negative** (*TN=1820*)	$NPV = \frac{1820}{1830} = 99.5\%$
		$TPR = \frac{20}{30} = 67\%$ (recall)	$TNR = \frac{1820}{2000} = 91\%$	$ACC = \frac{1840}{2030} = 90.6\%$

– Is this a good test for cancer?
  - We note that the false discovery rate ($1 - PPV = 90\%$) is very high, i.e., a lot of tests are positive but the patient does not have cancer. Hence, there is little confidence in positive outcomes and further tests are required.
  - We further note that the false omission rate ($1 - NPV = 0.5\%$) is very low, i.e., a negative test result is almost always a true negative case. This is an important element of the diagnosis of exclusion, especially if the above test is very cheap to conduct. The high true negative rate ($TNR = 91\%$) indicates that the elimination is in 91% successful.

– Using NPV as a driving performance metric is very common in cases where most of the population is considered negative.

– Accuracy (ACC) is not a reliable metric: assume an "oracle" that always predicts "No". This oracle yields an accuracy of $\frac{0+2000}{2030} = 98.5\%$ and, hence, beats the predictions in the above example. On the other side, $PPV = 0\%$, $NPV = 98.5\%$, $TPR = 0\%$ and $TNR = 100\%$ clearly indicate the limitations of this oracle.

- **Multi-class classification** (one out of a set of classes) requires a generalized **confusion matrix** resulting in a table such as the example below with people recognition in images:

		Actual Class		
	Population (*100*)	Woman (*20*)	Man (*20*)	*Child (60)*
Recognized Class	Woman (*19*)	*13*	*4*	*2*
	Man (*18*)	*2*	*15*	*1*
	Child (*63*)	*5*	*1*	*57*

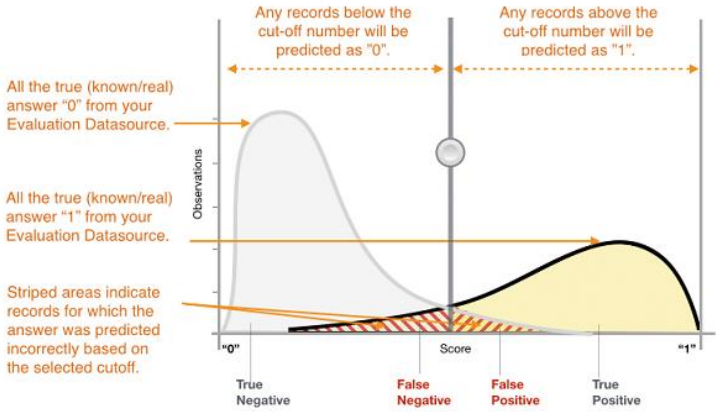
  - The confusion matrix allows to easily spot correct classifications (on the diagonal) and prediction errors (outside the diagonal). The table also depicts the cases for which the algorithm struggles to distinguish classes. In the example above, the algorithm recognized
    - 13 out of 20 women correctly, but 2 were wrongly classified as man and 5 as children
    - 19 women in total but only 68% (13) were actually women
    - 57 out of 60 children correctly, and children were more often confused with women than men
  - Accuracy is given by the sum of the diagonal over all examples, i.e., $ACC = \frac{13+15+57}{100} = 85\%$, and the error rate is $ERR = 1 - ACC = 15\%$. Again, accuracy alone is not capable to tell us the entire story; in the running example, the algorithm struggles with recognizing women. To better analyze the situation, we can create additional confusion matrices focusing on the correct classification of one class only. See next page for an example for the class “Woman” and “Child”

		Actual Class		
	Total Population	Woman ($P=20$)	Not a Woman ($N=80$)	
Recognized Class	Woman (19)	**True Positive** ($TP=13$)	**False Positive** ($FP=6$)	precision $PPV = \frac{13}{19} = 68\%$
	Not a Woman (81)	**False Negative** ($FN=7$)	**True Negative** ($TN=74$)	$NPV = \frac{74}{81} = 91\%$
		$TPR = \frac{13}{20} = 65\%$ recall	$TNR = \frac{74}{80} = 93\%$	$ACC = \frac{87}{100} = 87\%$

		Actual Class		
	Total Population	Child ($P=60$)	Not a Child ($N=40$)	
Recognized Class	Child (63)	**True Positive** ($TP=57$)	**False Positive** ($FP=6$)	precision $PPV = \frac{57}{63} = 90\%$
	Not a Child (37)	**False Negative** ($FN=3$)	**True Negative** ($TN=34$)	$NPV = \frac{34}{37} = 92\%$
		$TPR = \frac{57}{60} = 95\%$ recall	$TNR = \frac{34}{40} = 85\%$	$ACC = \frac{91}{100} = 91\%$

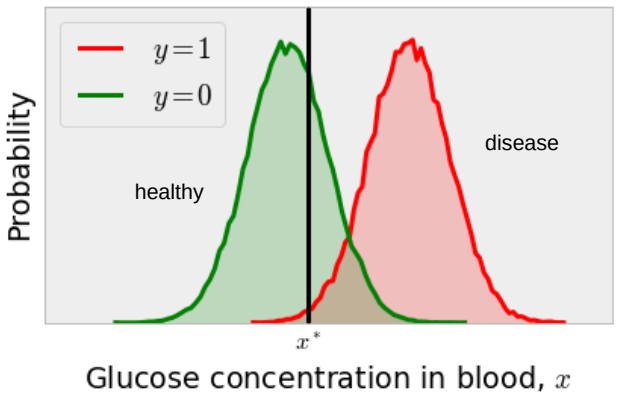
– Note that the accuracy for both classes “Woman” and “Child” are high and almost the same. However, it is wrong to conclude that the recognition of both classes works equally good. The reason for the good accuracy of class “Woman” is due to the large number of negative examples that are correctly dismissed. But precision (68%) and recall (65%) are much lower than for class “Child” documenting only mediocre capabilities to recognize women correctly.

- **Binary classification with scores and thresholds:** assume we have an algorithm that decides, based on a metric, whether an object belongs to a class or not. A good example is video shot detection: if the ‘distance’ between subsequent frames is large enough, we assume that a new shot has started (see application in later chapters of this course). The challenge is to set a threshold value for the distance in such a way, that the smallest number of errors occur (false positives, false negatives). In this scenario, we need:
  - a way to train ‘good’ thresholds as the overall performance of the method depends on it
  - a way to compare methods regardless of the chosen threshold to assess how well they can separate the positive from the negative cases



source: https://docs.aws.amazon.com/machine-learning/latest/dg/binary-classification.html

Medical Example: a new test shall distinguish between ‘healthy’ and ‘disease’ based on glucose concentration in the blood. The values of known populations are depicted on the right (green for healthy population on the left, and red for ‘disease’ population on the right). Given the test, we want to asses how well the test works and what thresholds we employ during medical examinations.
Is this a good medical test?
What threshold would you choose?



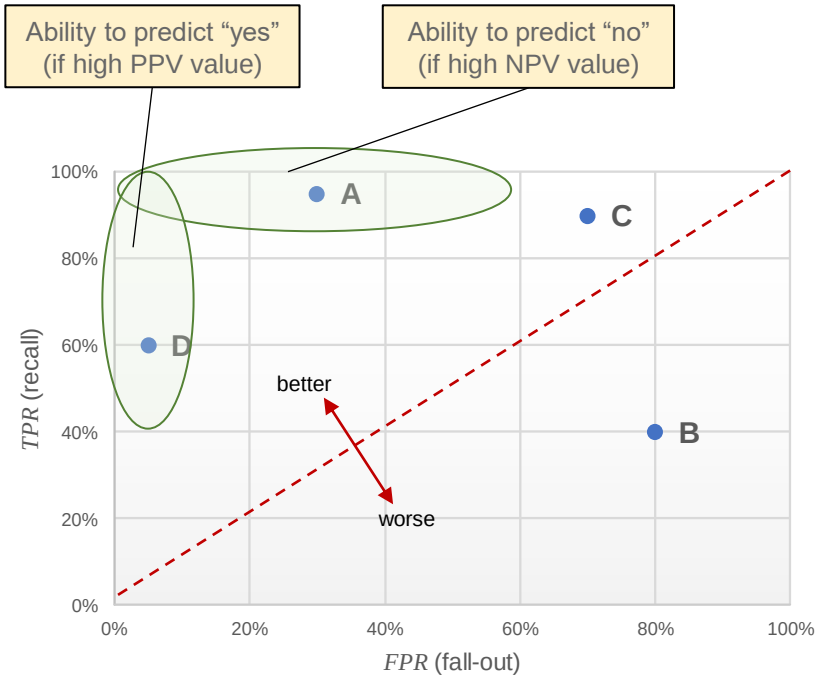
- **Binary classification with scores and thresholds** require an extension to the simple confusion matrix. Firstly, how do we calculate true/false positives/negatives if the algorithm says “Yes” only if the score exceeds a given threshold? Secondly, how do we favor algorithms that assign higher scores for positives (and lower for negatives)? The Receiver Operating Characteristic Curve (ROC Curve) is a simple tool to answer these questions.
  - The ROC curve is a 2-dimensional plot with the x-axis denoting the false positive rate (FPR) and the y-axis denoting the true positive rate (TPR). The ideal point is $(0,1)$, i.e., the upper-left corner with accuracy (ACC), precision (PPV), and recall (TPR) at 100% and fall-out (FPR) and miss rate (FNR) at 0%. In general, the more north-west the better, the more south-east the worse the performance is.
  - Example without scores and thresholds:

A	
$TP = 95$	$FP = 30$
$FN = 5$	$TN = 70$
$TPR = 95\%$	$FPR = 30\%$
$PPV = 76\%$	$NPV = 93\%$
$ACC = 83\%$	

B	
$TP = 40$	$FP = 80$
$FN = 60$	$TN = 20$
$TPR = 40\%$	$FPR = 80\%$
$PPV = 33\%$	$NPV = 25\%$
$ACC = 30\%$	

C	
$TP = 90$	$FP = 70$
$FN = 10$	$TN = 30$
$TPR = 90\%$	$FPR = 70\%$
$PPV = 56\%$	$NPV = 75\%$
$ACC = 60\%$	

D	
$TP = 60$	$FP = 5$
$FN = 40$	$TN = 95$
$TPR = 60\%$	$FPR = 5\%$
$PPV = 92\%$	$NPV = 70\%$
$ACC = 78\%$	

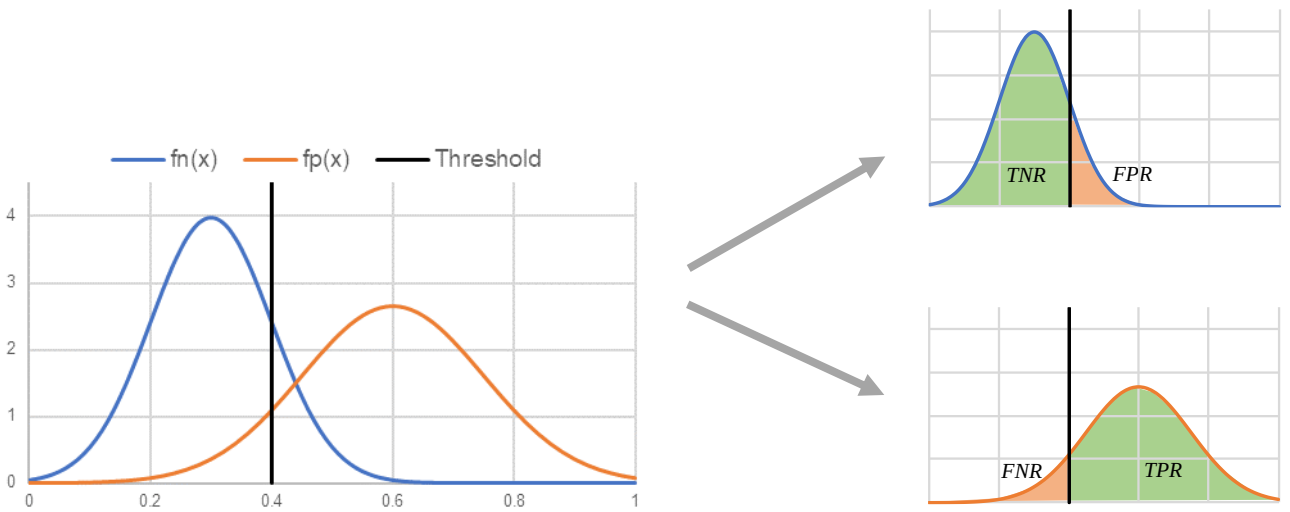


– Adding scores and threshold changes the way the algorithm decides. With binary classification, assume that the prediction is based on a random variable X which is a score for the current instance. The higher the score, the more likely it is a positive case, the lower the score the more likely it is a negative case. A threshold T is required such that the algorithms yields "Yes" if $X > T$ and "No" otherwise.
  - Let $f_p(x)$ denote the probability density of X if the instance belongs to class "positive"
  - Let $f_n(x)$ denote the probability density of X if the instance belongs to class "negative"
– We can calculate the various rates as a function of the threshold T as follows

$$TPR(T) = \int_T^{\infty} f_p(x)\,dx \qquad FNR(T) = \int_{-\infty}^{T} f_p(x)\,dx$$

$$TNR(T) = \int_{-\infty}^{T} f_n(x)\,dx \qquad FPR(T) = \int_T^{\infty} f_n(x)\,dx$$

or visually

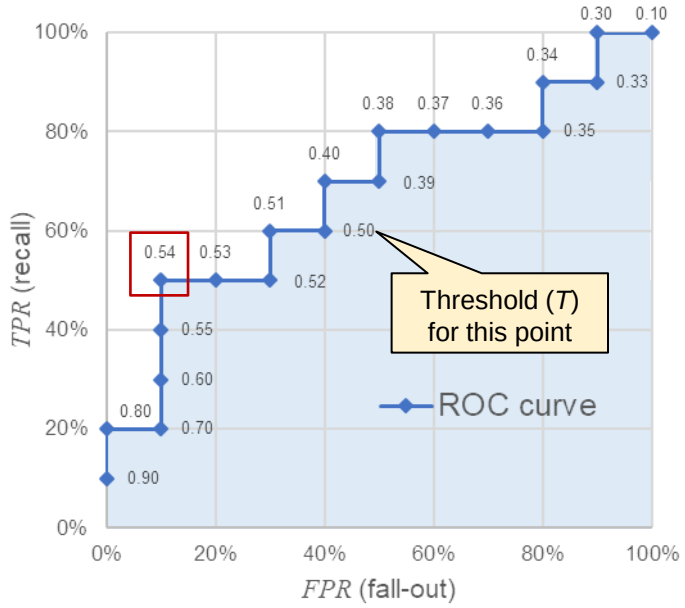


– The ROC curve serves two purposes: 1) optimize the threshold T, and 2) assess the performance of the algorithm. Let us consider the following simple example with 20 instances with labels P("positive") and N ("negative"). The medical test (in this example) returns a score between 0 and 1, the higher the score the more likely it is positive (i.e., yields a "yes").

Use score of current row as threshold

Highest accuracy with T=0.54

Class	Score	TP	FP	FN	TN	TPR	FPR	ACC
P	0.90	1	0	9	10	10%	0%	55%
P	0.80	2	0	8	10	20%	0%	60%
N	0.70	2	1	8	9	20%	10%	55%
P	0.60	3	1	7	9	30%	10%	60%
P	0.55	4	1	6	9	40%	10%	65%
P	0.54	5	1	5	9	50%	10%	70%
N	0.53	5	2	5	8	50%	20%	65%
N	0.52	5	3	5	7	50%	30%	60%
P	0.51	6	3	4	7	60%	30%	65%
N	0.50	6	4	4	6	60%	40%	60%
P	0.40	7	4	3	6	70%	40%	65%
N	0.39	7	5	3	5	70%	50%	60%
P	0.38	8	5	2	5	80%	50%	65%
N	0.37	8	6	2	4	80%	60%	60%
N	0.36	8	7	2	3	80%	70%	55%
N	0.35	8	8	2	2	80%	80%	50%
P	0.34	9	8	1	2	90%	80%	55%
N	0.33	9	9	1	1	90%	90%	50%
P	0.30	10	9	0	1	100%	90%	55%
N	0.10	10	10	0	0	100%	100%	50%



– The table is ordered by the scores of the 20 instances. In each row, we consider the score as the threshold and compute TP, FP, FN, and TN with that threshold across all 20 instances. The resulting TPR and FPR values are then depicted in the ROC cure on the right. In this example, we selected the optimal threshold (0.54) based on the row with highest accuracy (70%)
– In general, higher thresholds tend to be more "conservative" (less false positive) while lower thresholds are more "liberal" (more true positives). Accuracy is only one way to select a threshold. Other values like precision, recall or fall-out can be used as well.
– Performance of an algorithm can be measured regardless of the selected threshold with the **area under the ROC curve** (blue area, right figure); the bigger the area, the better the algorithm.

1.8 Literature and Links

- [MS MARCO] – MS MARCO (Microsoft Machine Reading Comprehension Dataset), https://microsoft.github.io/msmarco/
- [TREC] – Text REtrieval Conference, http://trec.nist.gov/
- Kent, Allen; Berry, Madeline M.; Luehrs, Jr., Fred U.; Perry, J.W. (1955). "Machine literature searching VIII. Operational criteria for designing information retrieval systems". American Documentation. **6** (2): 93. doi:10.1002/asi.5090060209.
- Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier (1999). *Modern Information Retrieval*. New York, NY: ACM Press, Addison-Wesley. ISBN 0-201-39829-X
- David A. Grossman, Ophir Frieder, “Information Retrieval Algorithms and Heuristics.“, Kluwer Academic Publishers, 1998
- Zou, Kelly H.; O'Malley, A. James; Mauri, Laura (2007); *Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models*, Circulation, 115(5):654–7
- Hanley, James A.; McNeil, Barbara J. (1982). "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve". *Radiology*. **143** (1): 29-36. PMID 7063747. doi:10.1148/radiology.143.1.7063747.
- Fawcett, Tom (2006). "An Introduction to ROC Analysis" (PDF). *Pattern Recognition Letters*. **27** (8): 861–874. doi:10.1016/j.patrec.2005.10.010.
- Amazon AWS: https://docs.aws.amazon.com/machine-learning/latest/dg/evaluating-model-accuracy.html