Business Analytics & Machine Learning

Model Selection and Learning Theory

Prof. Dr. Martin Bichler

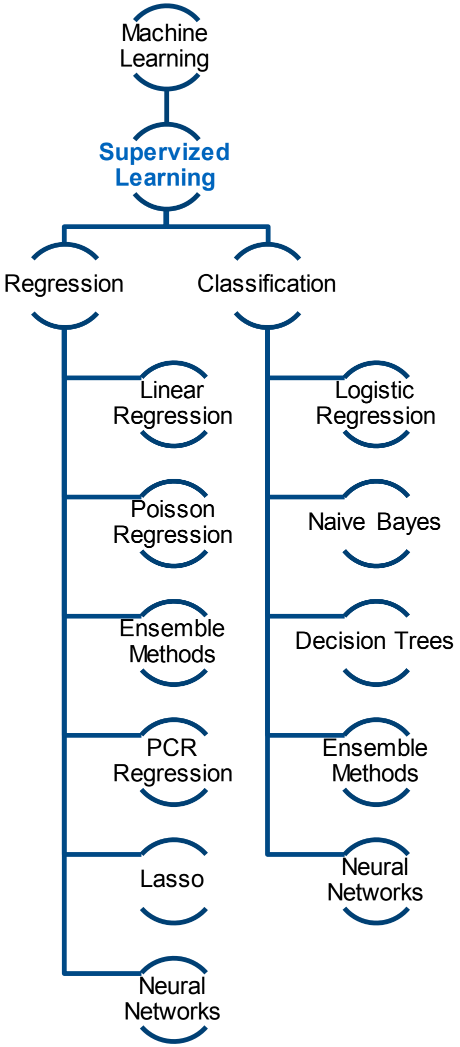
Department of Computer Science

School of Computation, Information, and Technology

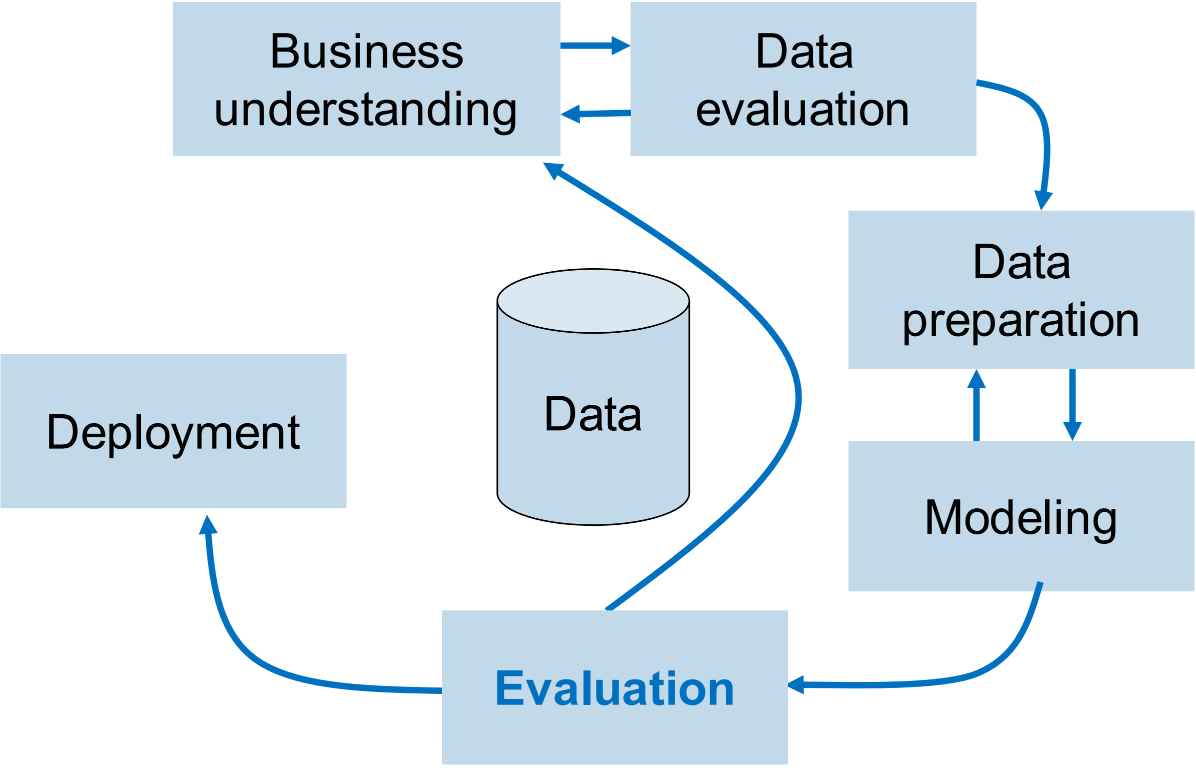
Technical University of Munich

Course Content

- Introduction
- Regression Analysis
- Regression Diagnostics
- Logistic and Poisson Regression
- Naive Bayes and Bayesian Networks
- Decision Tree Classifiers
- Data Preparation and Causal Inference
- **Model Selection and Learning Theory**
- Ensemble Methods and Clustering
- Dimensionality Reduction
- Association Rules and Recommenders
- Convex Optimization
- Neural Networks
- Reinforcement Learning



The CRISP Data Mining Process

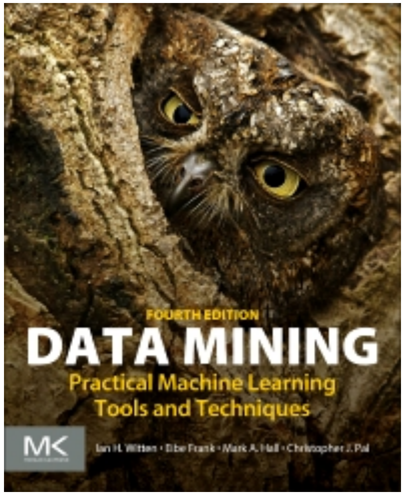


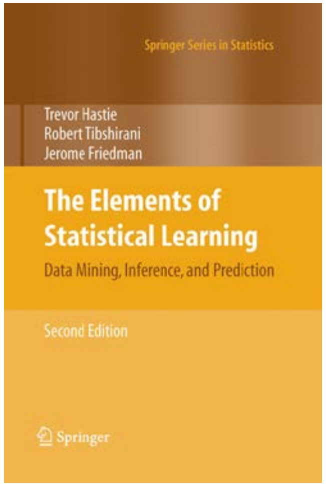
see also Witten&Frank: Chapter 5

Recommended Literature

- **Data Mining: Practical Machine Learning Tools and Techniques**
  - Ian H. Witten, Eibe Frank, Mark A. Hall
  - http://www.cs.waikato.ac.nz/ml/weka/book.html
  - Today's class: Chapter 5

- **The Elements of Statistical Learning**
  - Trevor Hastie, Robert Tibshirani, Jerome Friedman
  - https://web.stanford.edu/~hastie/ElemStatLearn/
  - Section 2.9, 7: Model Assessment and Selection


FOURTH EDITION
DATA MINING
Practical Machine Learning
Tools and Techniques
MK
Ian H. Witten · Eibe Frank · Mark A. Hall · Christopher J. Pal



Agenda for Today

- **Bias-Variance Tradeoff**
- Resampling Methods
- Gain and ROC Curves
- Comparison Studies
- Algorithmic Information Theory
- Computational Learning Theory

> *“All models are wrong; some are useful.”*
>
> \- George E. P. Box -

Supervised Learning

$\hat{y} = f(\boldsymbol{x})$

Supervised learning is inferring a function from labeled training data.

Training:

- given a *training set* of labeled examples $\{(x_1, y_1), \dots, (x_n, y_n)\}$
- estimate the prediction function f by minimizing the prediction error on the training set

Testing:

- apply f to a never-before-seen *test example* $\boldsymbol{x}$ and output the predicted value $\hat{y} = f(\boldsymbol{x})$

Statistical Inference vs. Prediction

Earlier, in the *statistics* classes, we also fit a model $\hat{y} = f(\boldsymbol{x})$

While this step is the same in supervised learning, the goals are usually different!

Inference ***(statistics):***

- Goal: Explicitly estimate the parameters β of the model $f(\boldsymbol{x})$ in order to
  - Reason about the entire population given data from a limited sample
  - Determine whether effects are significant
  - Directly interpret the meaning of parameters β

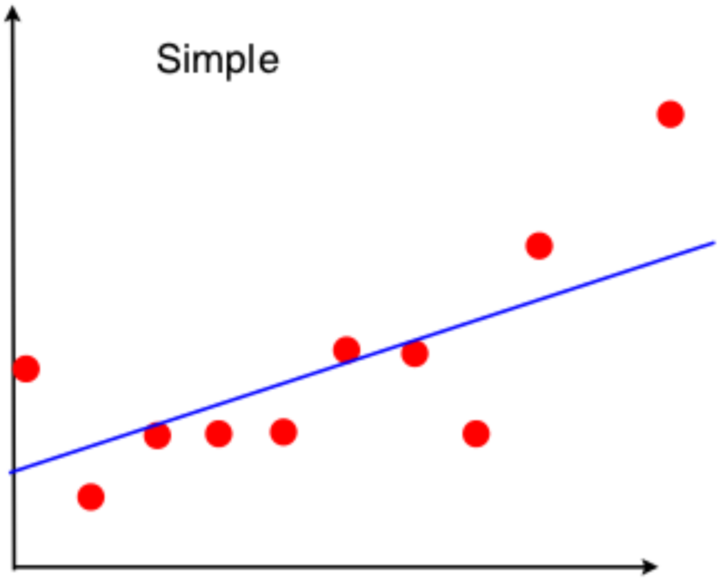
Prediction Task ***(machine learning)*****:**

- Goal: Make 'good' predictions $\hat{y}$ for unseen data points x
- Individual parameters/weights of the model are often not of interest unless interpretability/explainability is important.

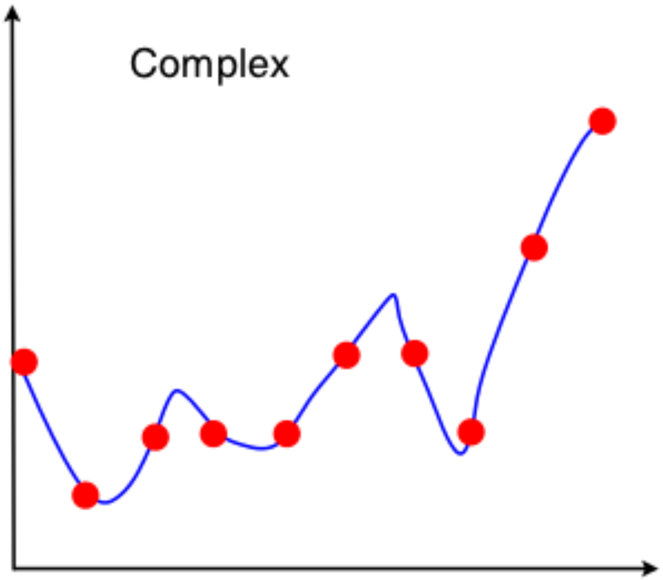
Note 1: Sometimes in the machine learning (especially deep learning) literature, the terms "*inference*" and "*prediction*" are confusingly used interchangeably referring to *prediction* above!

Note 2: Statistical inference methods like (Generalized) Linear Models can also be used for prediction tasks. When the Gauss-Markov assumptions are violated, inference becomes impossible, but prediction *may* still work well in practice.

Bias-Variance Tradeoff



Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Generalization Errors

Components of generalization error

- **Bias** is the error from erroneous assumptions in the learning algorithm. Error might be due to inaccurate assumptions/simplifications made by the model.
- **Variance** is the error from sensitivity to small fluctuations in the training set. High variance causes overfitting.

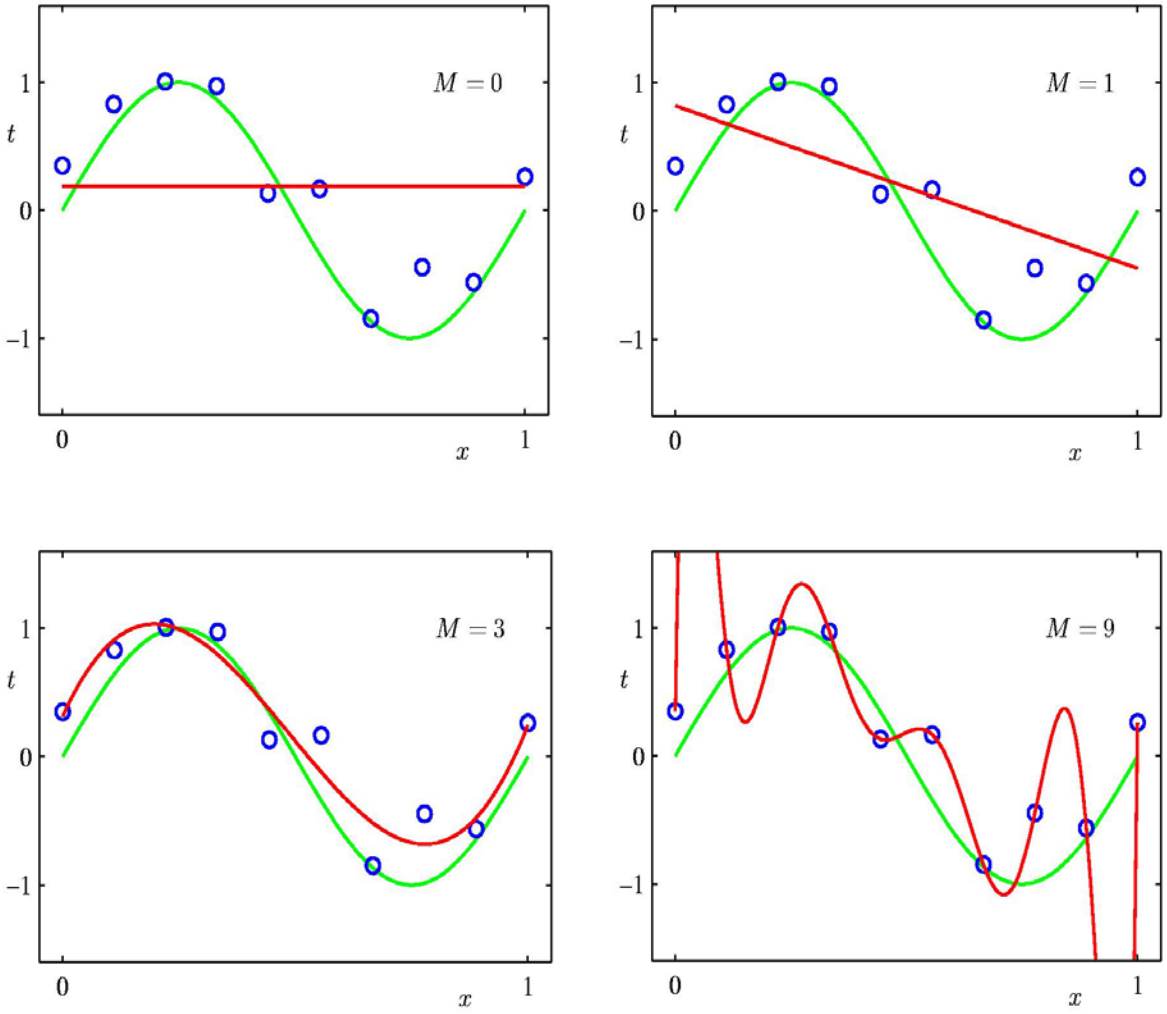
Underfitting: model is too "simple" to represent all relevant characteristics

- high bias and low variance
- high training error and high test error

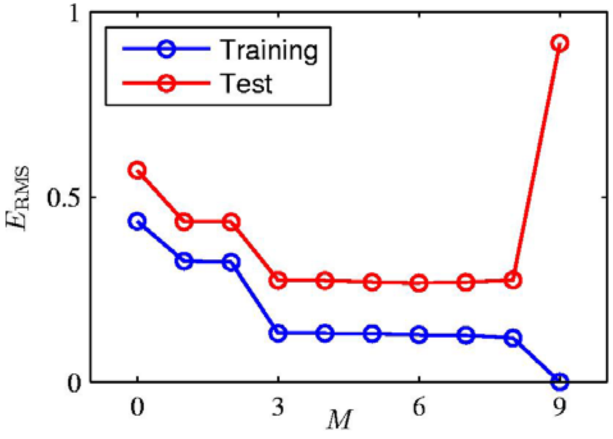
Overfitting: model is too "complex" and fits irrelevant characteristics/noise

- low bias and high variance
- low training error and high test error

More Complex Models, Less Bias, More Variance

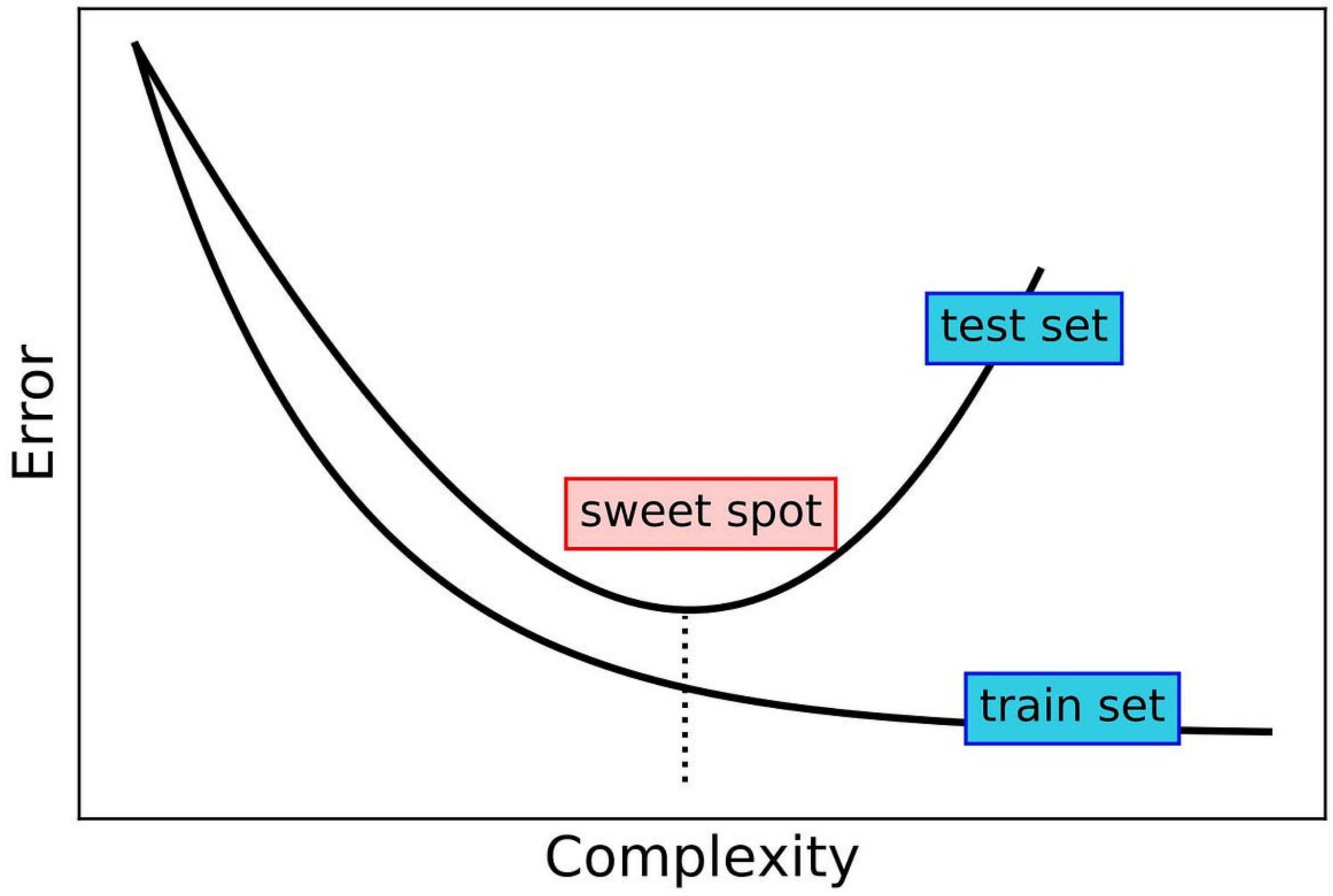


Fitting a polynomial

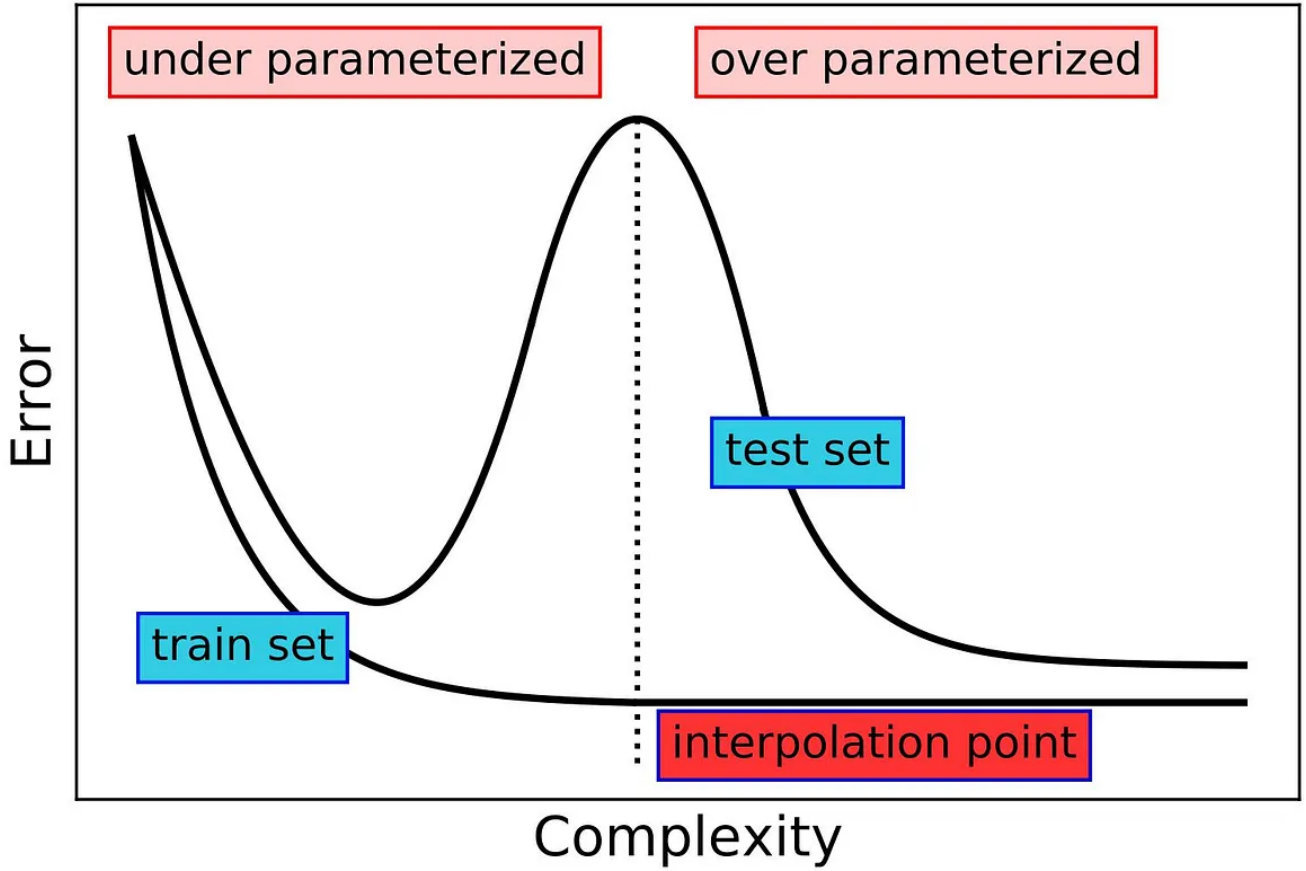


Root mean squared error

Bias-Variance Tradeoff



Double Descent



See https://arxiv.org/abs/2303.14151v1

Which Model Should be Selected?

The bias-variance tradeoff provides a conceptual framework for determining a good model (but it is not directly useful).

We need a *practical method* for optimizing bias-variance tradeoff.

Practical aim is to pick a model that minimizes a criterion:

f(fitting error from given data) + g(model complexity)

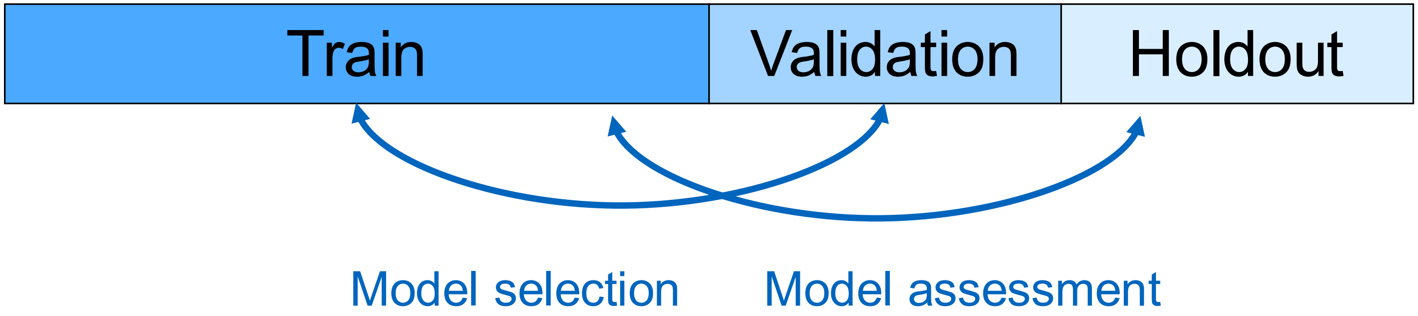
where f and g are increasing functions.

Most methods are based on a tradeoff between *fitting error* (high variance) and *model complexity* (low bias).

Model Selection and Model Assessment

Model selection:
Estimating performances of different models to choose the best one (produces the minimum of the test error).

Model assessment:
Having chosen a model, estimating the prediction error on new data.



Model Selection

Wide-spread methods for model selection are:

- Akaike Information Criterion (AIC)
    - $AIC = 2k - 2\ln(L)$, already discussed in the context of log. regression
    - k is the number of parameters, $\ln(L)$ the log likelihood
- Minimum description length (Risannen, 1978)
    - discussed later in this class
- Resampling methods
    - cross validation, jackknife, bootstrap, etc.
- etc.

How would you view decision tree pruning in the context of model selection?

Outline for today

- Bias-Variance Tradeoff
- **Resampling Methods**
- Gain and ROC Curves
- Comparison Studies
- Algorithmic Information Theory
- Computational Learning Theory

“All models are wrong; some are useful.”

- George E. P. Box -

Training and “Testing”

- Holdout procedure (i.e., training and “testing”)
  - reserve some data (the holdout set) for “testing” (usually ~1/3)
  - use remaining data for training
  - use holdout data set to estimate the error rate (assess the model)

- Stratified holdout
  - guarantee that classes/lables are (approximately) proportionally represented in the test and training set
  - important features may also be stratified, if enough data is available

- Repeated holdout (in addition)
  - randomly select holdout set several times and average the error rate estimates

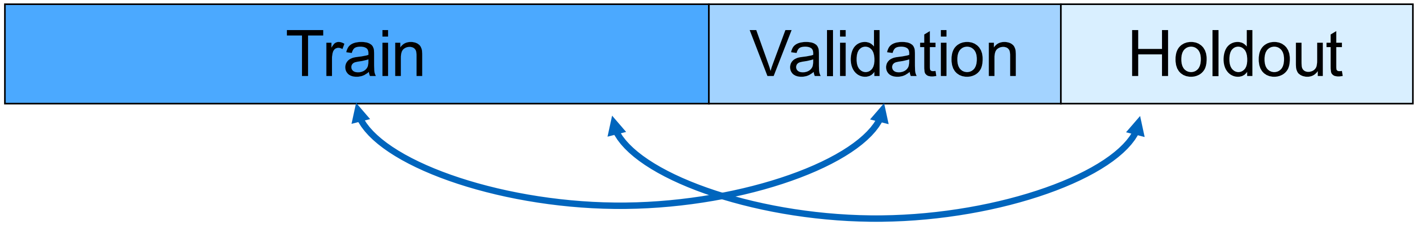
Train

Holdout

Train, Validate, “Test”

In addition, a part of the training set may be set apart for *validation*

1. The model is fit/trained on the **training set**
2. The performance is evaluated on the **validation set**
   - Validation performance inform design decisions about the model (different model? Different hyperparemeters?) → repeat 1 and 2
3. The selected final model is *retrained using* ***both training + validation*** *set.*
   The performance of the final model is assessed on the **holdout set** that has **not** been used for any design decisions.



Model selection

Model assessment

Detour: Terminology

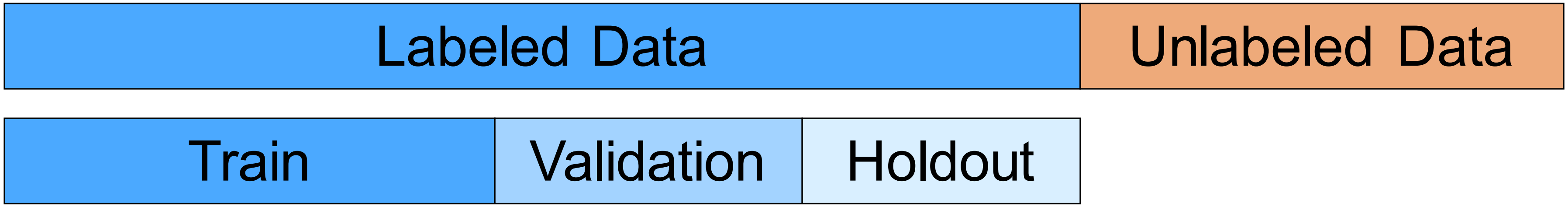
Labeled Data

Unlabeled Data

- **Labeled data** is the data for which you have access to pairs (x, y) of features and labels *at the time you design your machine learning model.*

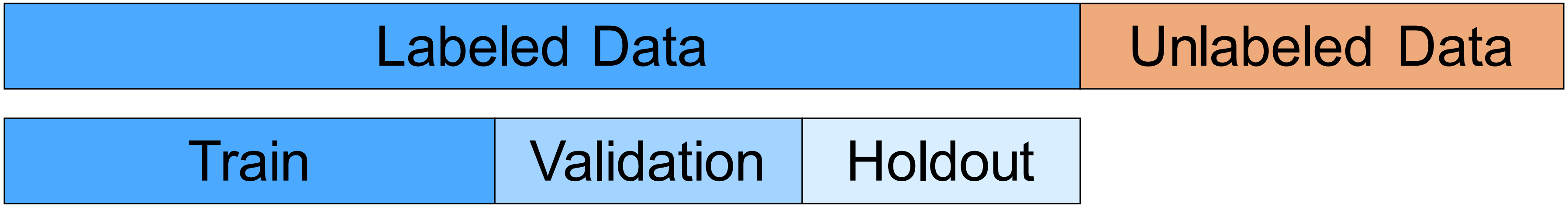
  Model training, model selection and model assessment need to be performed using the labeled data.
- **Unlabeled data** only consists of features, without access to labels y. Making good predictions $\hat{y}$ is the goal of the model. The unlabeled data can either
  - be a fixed data set X (e.g. offline prediction problems, machine learning competitions, the Analytics Cup in this class)
  - arrive over time as a stream of individual data points x (e.g., when a model is deployed in a business process. For example you may need to score the credit-worthiness of new customers as they arrive.)
  - be a combination of both.

Detour: Terminology

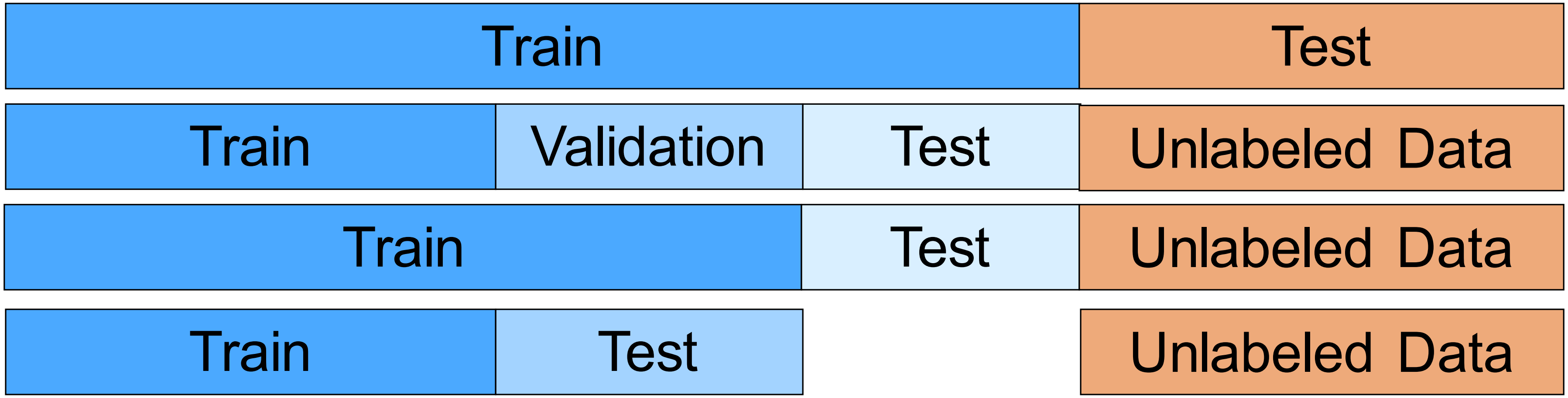


- **The labeled data** is commonly split into several sub-datasets in order to separate the data used for **model training, model selection,** and **model assessment**.
  - A **training set** is used to fit models, and thus find its *parameters.*
  - A **validation set** is a portion of the data used to *repeatedly* evaluate trained models. The information learned in validation can be used for model selection, e.g. choosing between different models or choosing the models‘ *hyperparameters.*
  - A **holdout set** is used for model assessment.
    Applying your model to the holdout set should give you a *final* estimation about how well your model can be expected to perform on unseen, unlabeled data.
    Using any information from holdout assessment in your model design will lead to **bias** this estimate and make it impossible to asess your model strength.

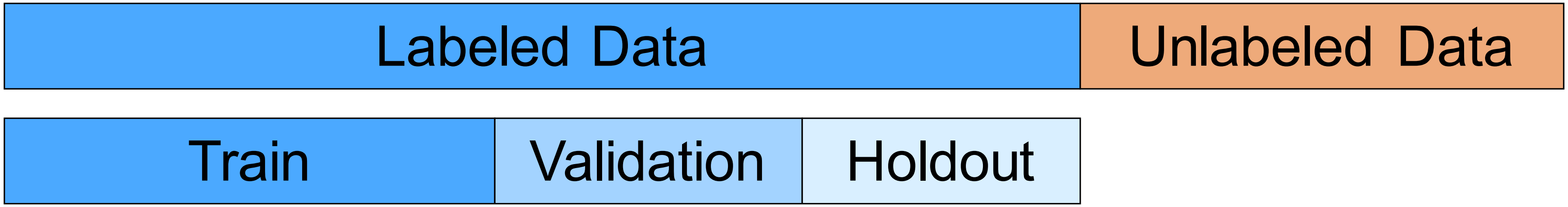
Detour: Terminology



- The term **test set** is used ambiguously based on the context and who is speaking. Online and in the literature, you may encounter the following combinations. Similarly, the term **training data** may sometimes have a wider definition than above. You may encounter all of the following combinations:



Detour: Terminology

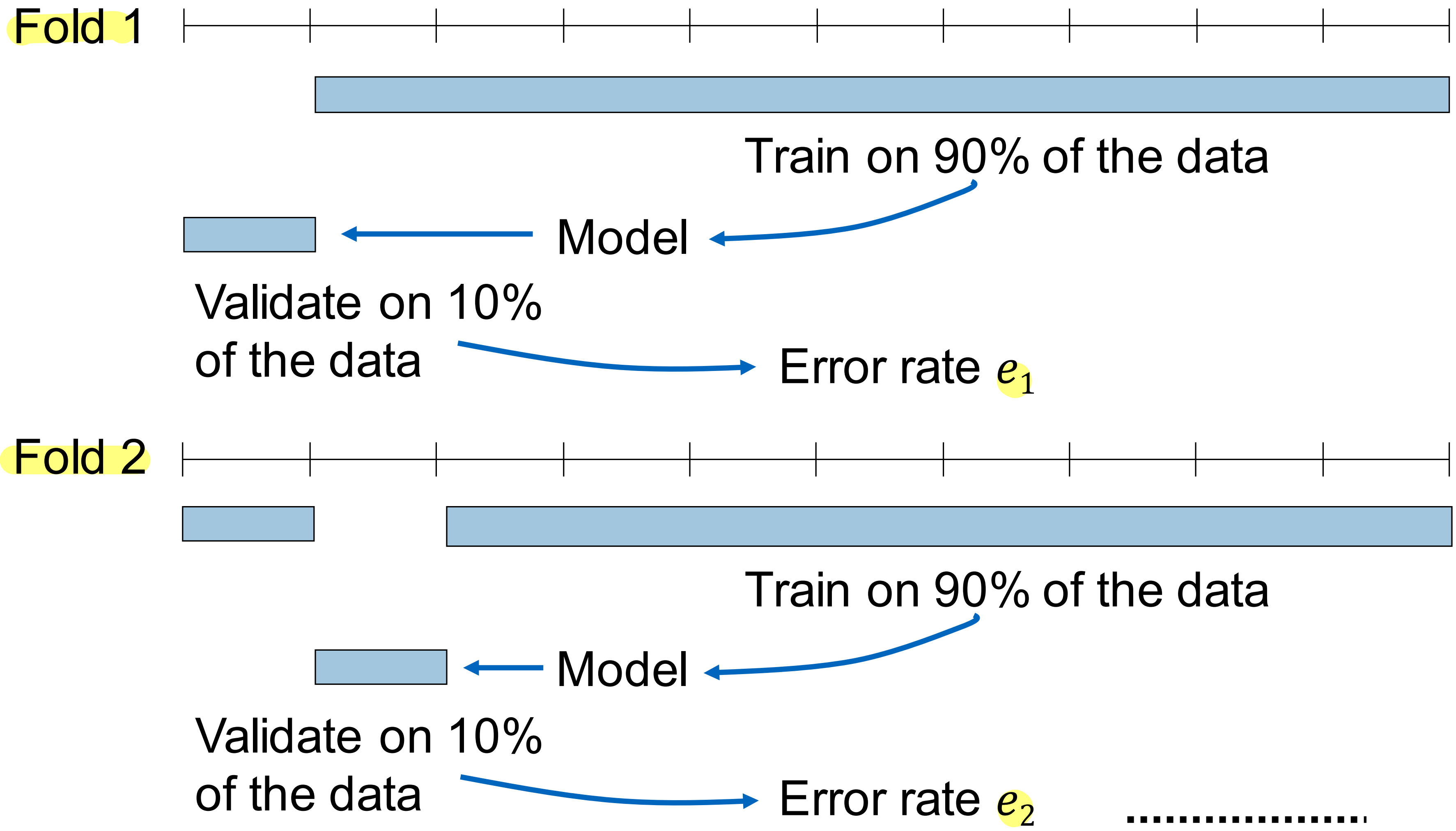


- The term **test set** is used ambiguously based on the context and who is speaking. Online and in the literature, you may encounter the following combinations. Similarly, the term **training data** may sometimes have a wider definition than above. You may encounter all of the following combinations.

- *Although ambiguous, avoiding the terms "testing" and "test set" altogether is hard, and often the context is clear.*

  If one of the instructors uses the term 'testing' or 'test set' and you are unsure what is meant in the current context, please stop us and ask!

Cross Validation



Fold 1

Train on 90% of the data

Model

Validate on 10% of the data

Error rate e_1

Fold 2

Train on 90% of the data

Model

Validate on 10% of the data

Error rate e_2

……………….

k-fold Cross-Validation

- fixed number of k partitions of the data (folds)
- in turn: each partition is used for validation and the remaining instances for training
  - – In total, each instance is used for validation exactly once
- may use stratification → preserve the classes distribution
- standard practice: stratified ten-fold cross-validation
- error rate is estimated by taking the average of error rates
- *select model* that performs best over all test subsets

$$\boxed{\hat{e} = \frac{1}{k}\sum_{i=1}^{k} e_i}$$

Comparing Error Rates

Suppose we have two algorithms

- obtain two different models
- estimate the error rates for the two models
- compare estimates

  $\hat{e}^{(1)} < \hat{e}^{(2)}$?
- select the better one

Problem?

- Are there “significant” differences in the error rates?

Comparing Error Rates

Estimated error rate is just an estimate (random).

Student's paired t-test tells us whether the means of two samples are significantly different.

Construct a t-test statistic:

- need variance as well as point estimates

$$t = \frac{\bar{d}}{s_d/\sqrt{k}}$$

Average of differences of error rates

Observed standard deviation of diff. in error rate

H_0: Difference = 0

Salzberg, Steven L. "On comparing classifiers: Pitfalls to avoid and a recommended approach." *Data mining and knowledge discovery* 1.3 (1997): 317-328.

Bootstrap as an Alternative to Cross-Validation

1. Number your observations $1,2,3,\ldots$
2. Draw a random sample of size n WITH REPLACEMENT.
3. Calculate your statistic (e.g. error rate, mean) with these data.
4. Repeat steps 1-3 many times (e.g., 500 times).
5. Calculate the variance of your statistic (e.g. error rate or the mean) directly from your sample of 500 statistics to learn about population statistics.
6. You can also calculate confidence intervals directly from your sample of 500 statistics. Where do 95% of statistics fall?

Example: Re-sample 500 samples of n=50 with replacement, run logistic regression and examine the distribution of error rates (or other metrics).

Measuring Errors

		Predicted class	
		Yes	No
Actual class	Yes	True positive (TP)	False negative (FN) (Type II error)
	No	False positive (FP) (Type I error)	True negative (TN)

Error rate = # of errors / # of instances = (FN+FP) / N

Recall (hit rate) = TP / (TP+FN) Actual Yes —

Precision = TP / (TP+FP) Predicted Yes |

Specificity = TN / (TN+FP) Actual No —

False alarm rate = FP / (FP+TN) = 1 - Specificity

↳ predicted as "Yes" but actual "No"

Counting the Costs

In practice, different types of classification errors often incur different costs.

Examples:

- Predicting when customers leave a company for competitors (churn prognosis)
  - Much more costly to lose a valuable customer
  - Much less costly to act on a false customer
- Loan decisions
- Oil-slick detection
- Fault diagnosis
- Promotional mailing

		Predicted class	
		Yes	No
Actual class	Yes	True positive	False negative
	No	False positive	True negative

Cost-Sensitive Learning

Most learning schemes minimize total error rate

- Costs were not considered at training time.
- They generate the same classifier no matter what costs are assigned to the different classes.
- Example: standard decision tree learner

Simple methods for cost-sensitive learning

- Weighting of instances according to costs
- Resampling of instances according to costs
  - E.g. increase the "*no*" instances in training, which yields a model that is biased towards avoiding errors on "*no*" instances. When testing on the original test data set, there will be fewer false positives.

Outline for today

- Bias-Variance Tradeoff
- Resampling Methods
- **Gain and ROC Curves**
- Comparison Studies
- Algorithmic Information Theory
- Computational Learning Theory

"All models are wrong; some are useful."

- George E. P. Box -

Direct Marketing

Find most likely prospects to contact.

Not everybody needs to be contacted.

Number of targets is usually much smaller than number of prospects.

Typical applications:

- retailers, catalogues, direct mail (and e-mail)
- customer acquisition, cross-sell, churn prediction
- ...

Direct Marketing Evaluation

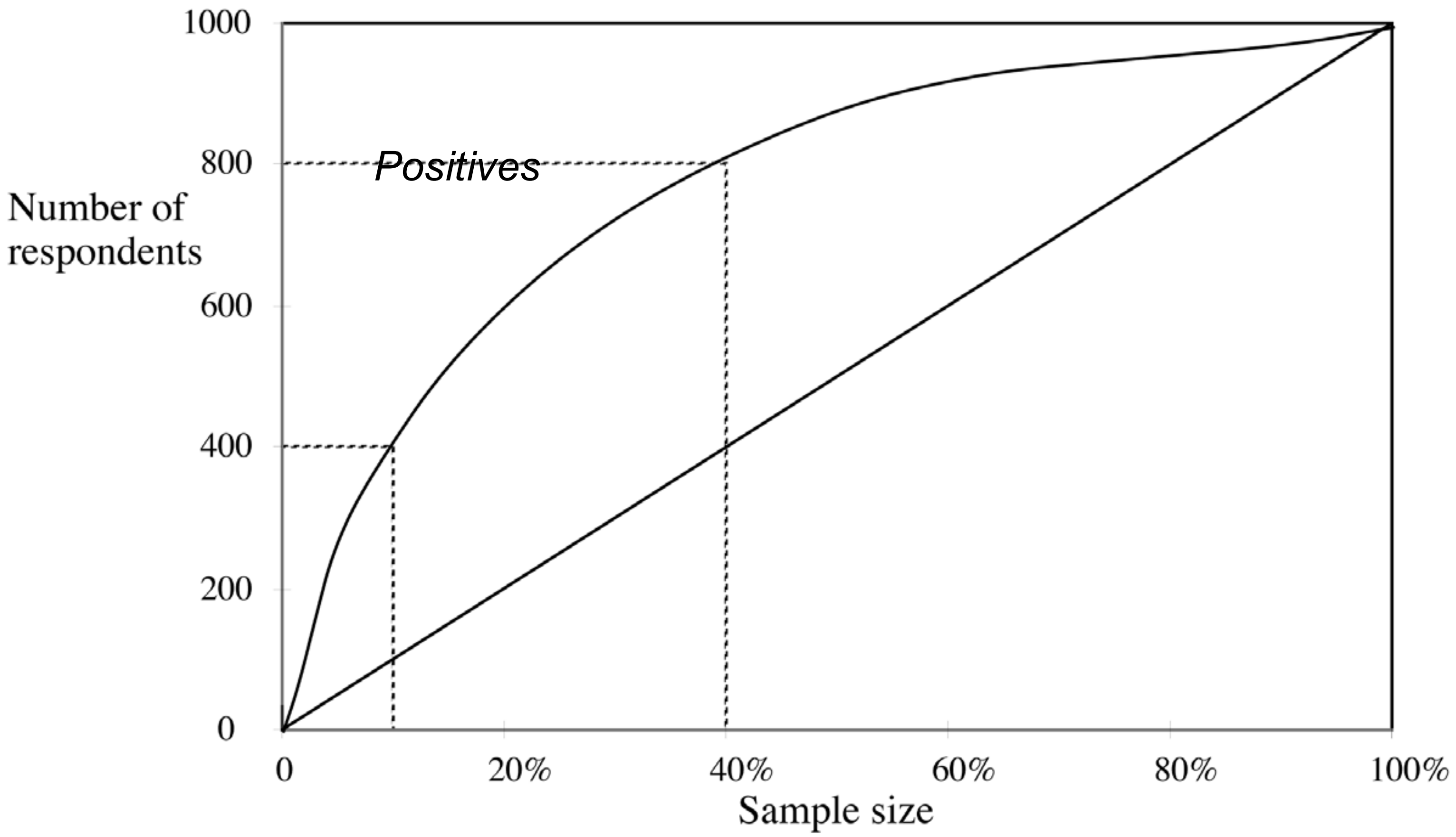
Accuracy on the entire dataset is not the right measure.

Approach:

- develop a target model
- score all prospects and rank them by decreasing score
- select top q % of prospects for action

How to decide what is the best selection?

A Hypothetical Gain Curve



Generating a Gain Curve

Instances are sorted according to their predicted probability:

Rank	Predicted probability	Actual class
1	0.95	Yes
2	0.93	Yes
3	0.93	No
4	0.88	Yes
…	…	…

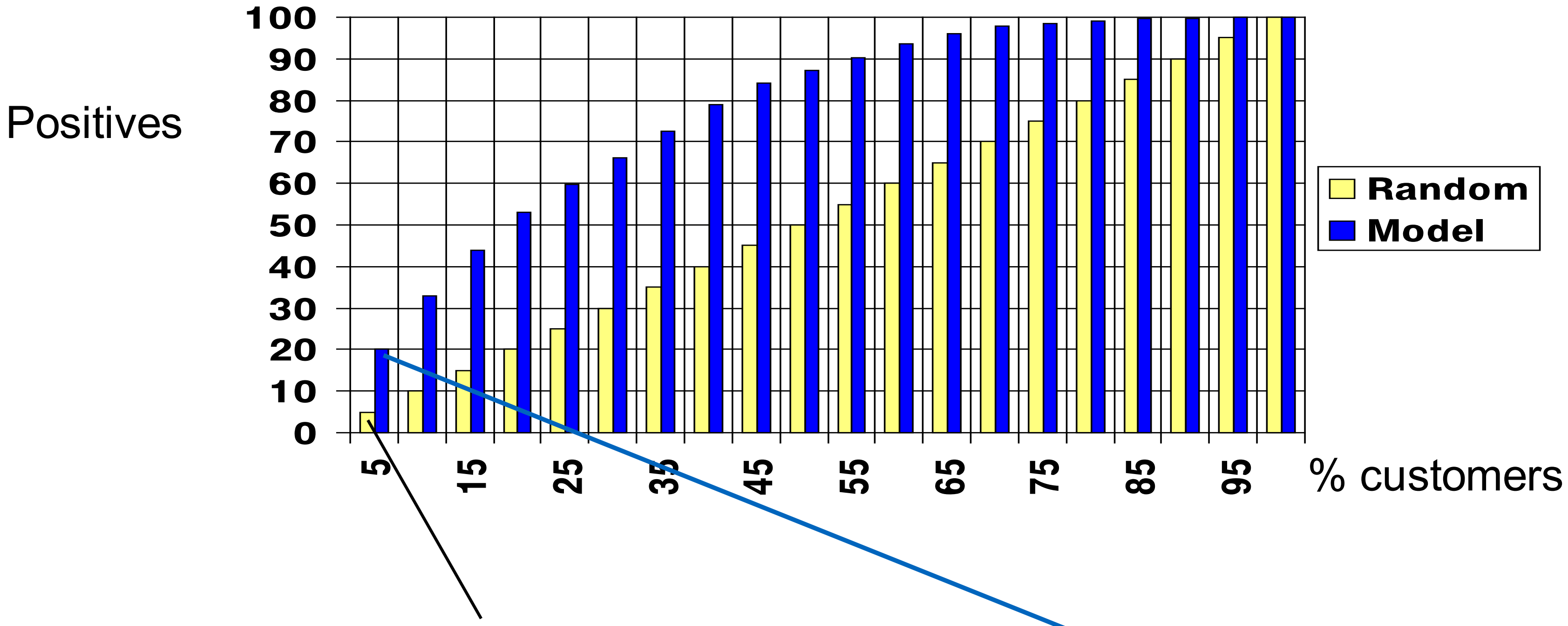
3 hits in top 5% of the list.

If there are 15 targets overall, then top 5 has 3/15=20% of targets.

In a gain curve

- x axis is sample size
- y axis is number of positives

Gain Curve: Random List vs. Model-Ranked List



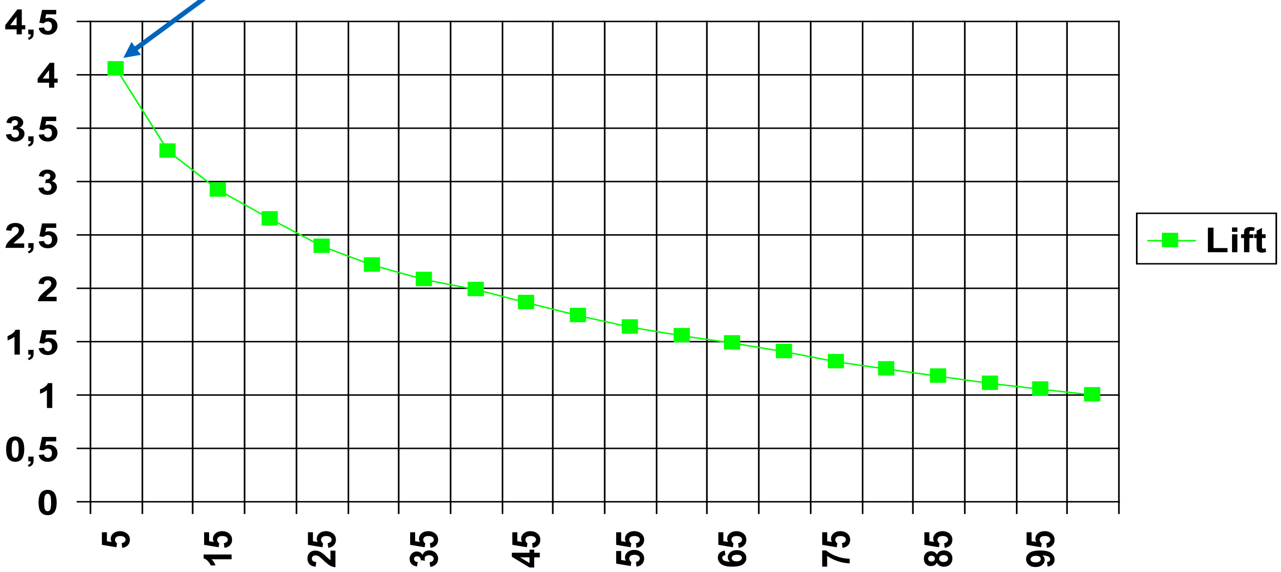
5% of random list have 5% of targets,

but 5% of model ranked list have 20% of targets Gain(5%)=20%.

Lift Curve

Lift(q)
= $\mathrm{Gain}(q)/q$

Lift (at 5%) = 20% / 5% = 4.0
times better than random



q (percent of the list)

Note: Some (including Witten & Frank) use "Lift" for what we call Gain.

ROC Curves

Differences to gain chart: recall vs. false alarm rate

- y axis shows percentage of true positives in sample (rather than absolute number):
  - $TP\,rate = tp = 100 * \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$
- x axis shows percentage of false positives in sample (rather than sample size):
  - $FP\,rate = fp = 100 * \mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$

usually, sort instances first by probability

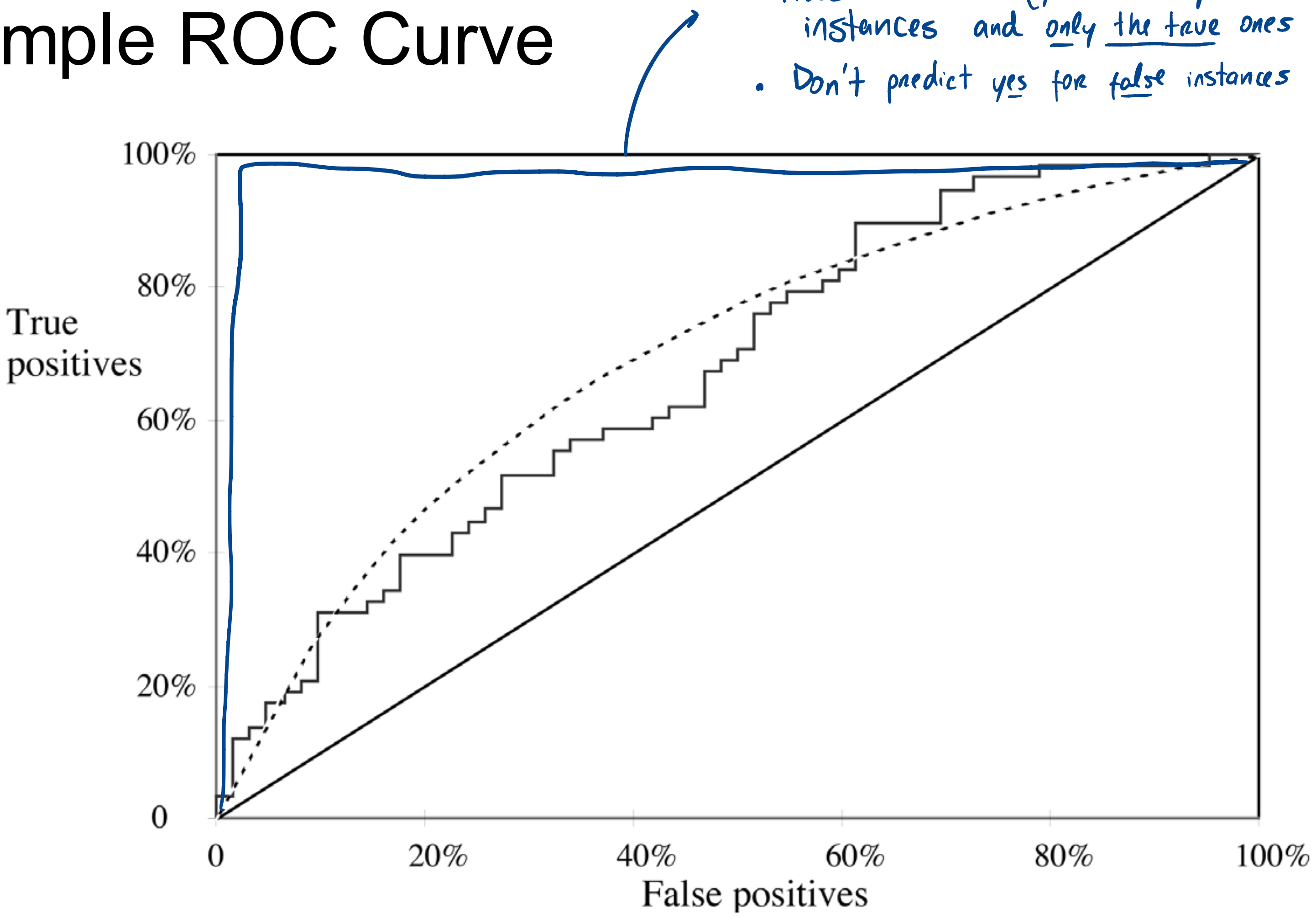
ROC curves are similar to gain curves

- "ROC" stands for "receiver operating characteristic"
- used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel
- plot tp vs. fp

A Sample ROC Curve

ideal: close to left corner

- Able to classify correctly all true instances and only the true ones
- Don't predict yes for false instances

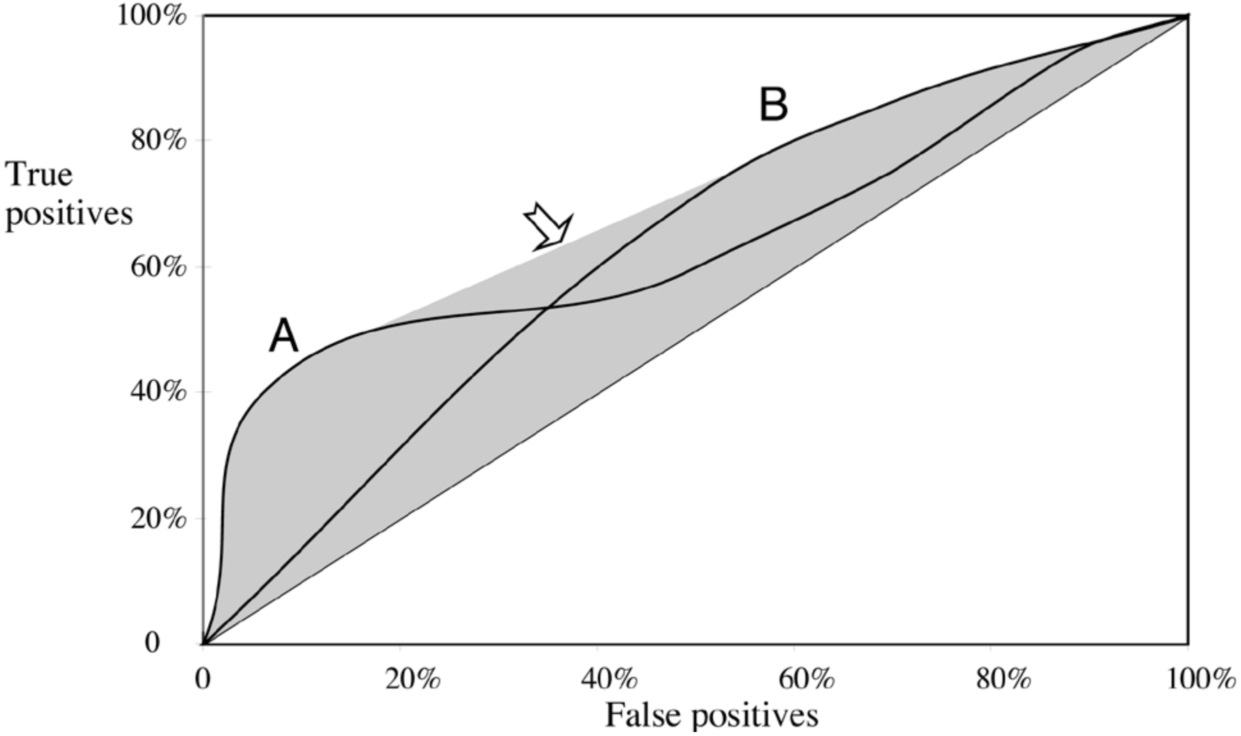


Jagged curve - one set of test data

Smooth curve - use cross-validation and average

ROC Curves for Two Classifiers

- For a small, focused sample, use method A (e.g., only interested in 40% of true positives).
- For a larger one, use method B.



Outline for today

- Bias-Variance Tradeoff
- Resampling Methods
- Gain and ROC Curves
- **Comparison Studies**
- Algorithmic Information Theory
- Computational Learning Theory

“All models are wrong; some are useful.”

- George E. P. Box -

Comparison Studies of Classification Methods

Large number of classification techniques from the field of ML, NN and Statistics

Many empirical comparisons in the 80‘s and 90‘s with contradictory results

StatLog Project (already in the mid 90‘s)

- more than 20 methods and 20 datasets
- performance of methods depends very much on the data set
- no general guidelines
- comparison study is advisable in special cases

State-of-the-practice

- comparisons of several methods on a particular data set
- sometimes even automated „mass modeling“

Recommendations from StatLog

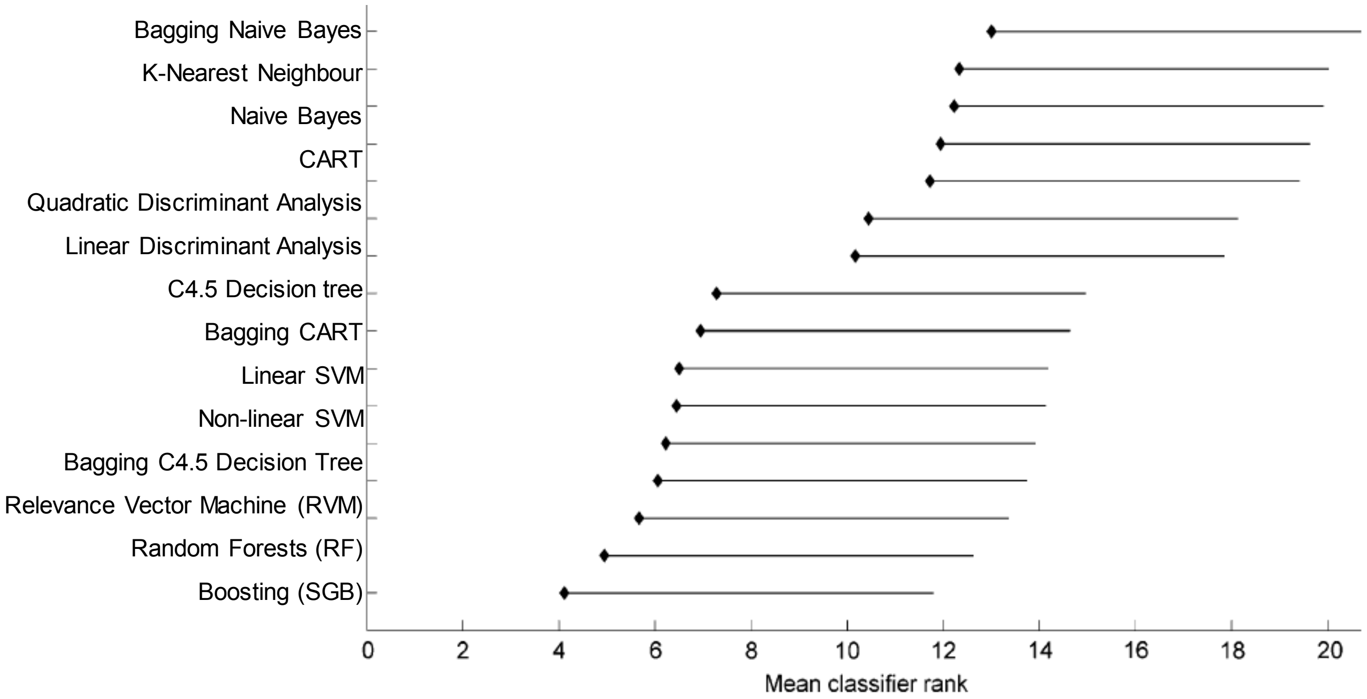
Following methods should be considered in real-world comparison studies

- Logistic Regression (Discriminant Analysis)
- Decision trees
- *K*-Nearest Neighbour
- Non-parametric statistical methods
- Neural Networks

Decision trees and logistic regression are widely used in practice

- High performance / low error rate
- Speed of model building and classification
- Easy to explain
  - as compared to NN or *k*NN

Another Study: Modern vs. Traditional Classifiers



Stefan Lessmann and Stefan Voß. "Customer-centric decision support." Business & Information Systems Engineering 2.2 (2010): 79-93.

When would you use error rates to select a model, when a gain curve?

Theoretical Foundations

Various theoretical streams aim for an understanding of machine learning:

Algorithmic information theory

- Kolmogorov complexity
- Minimum description length principle
- etc.

Computational learning theory

- Vapnik–Chervonenkis (VC) theory (led to SVMs)
- Probably approximately correct (PAC) learning (basis for boosting)
- Bayesian inference (basis for Bayesian networks)
- Algorithmic learning theory
- Online machine learning
- etc.

Outline for today

- Bias-Variance Tradeoff
- Resampling Methods
- Gain and ROC Curves
- Comparison Studies
- **Algorithmic Information Theory**
- Computational Learning Theory

“All models are wrong; some are useful.”

- George E. P. Box -

Traditional Model Selection Criteria in Science

Can we find a single 'best' model / classifier?

Model selection criteria attempt to find a good compromise between:

- the complexity of a model
- its prediction accuracy on the training data

Reasoning:
A good model is a simple model that achieves high accuracy on the given data.

Also known as Occam's Razor:
The best theory is the smallest one that describes all the facts.

William of Ockham, born in the village of Ockham in Surrey (England) about 1285, was the most influential philosopher of the 14th century and a controversial theologian.

Elegance vs. Errors

Theory 1: very simple, elegant theory that explains the data almost perfectly.

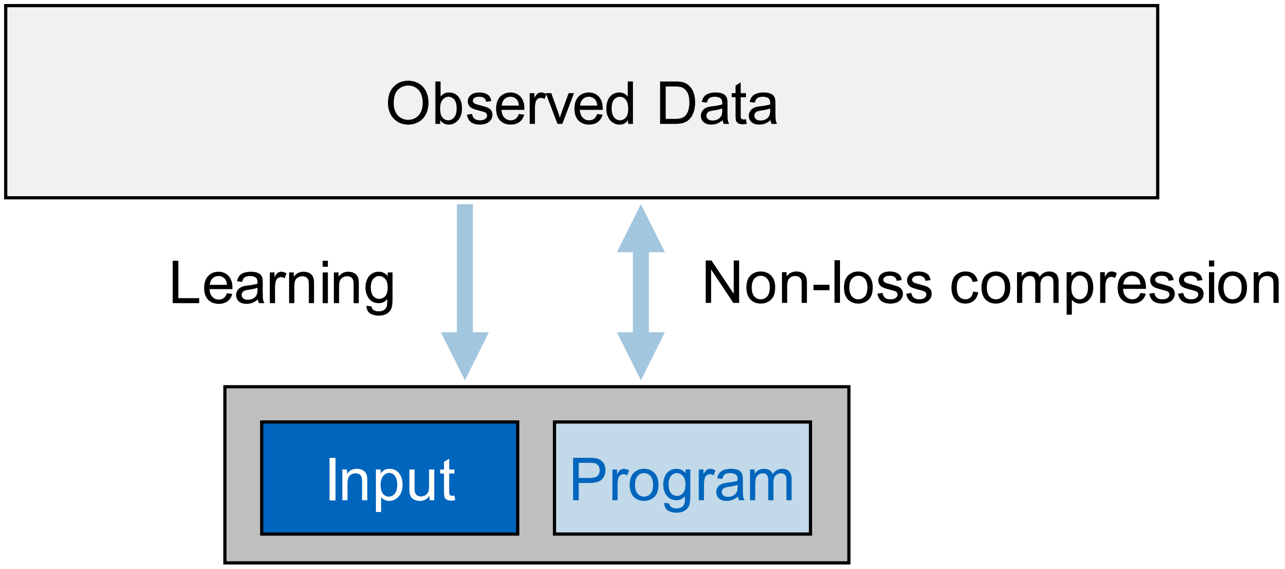
Theory 2: significantly more complex theory that reproduces the data without mistakes.

Theory 1 is probably preferable.

Classical example: Kepler’s three laws on planetary motion.

- Less accurate than Copernicus’s latest refinement of the Ptolemaic theory of epicycles.

Inductive Learning Systems



Observed Data

Learning

Non-loss compression

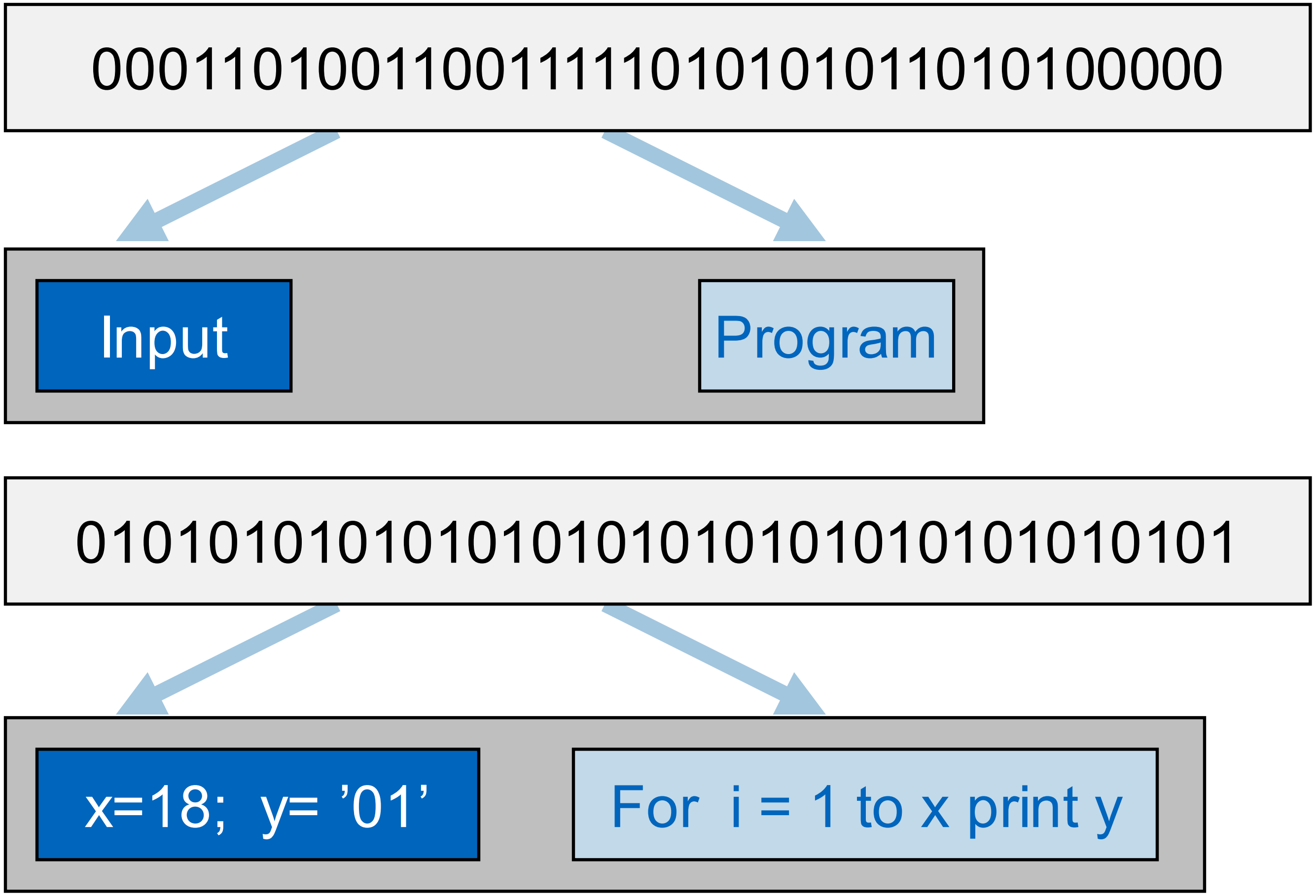
Input

Program

Learned Theory

|Theory| < |Data|

Example: Finite Binary String



Data:

000110100110011111010101011010100000

Theory:

Input

Program

Data:

010101010101010101010101010101010101

Theory:

x=18; y= ’01’

For i = 1 to x print y

|Theory| = |Program| + |Input| < |Data|

Randomness vs. Regularity

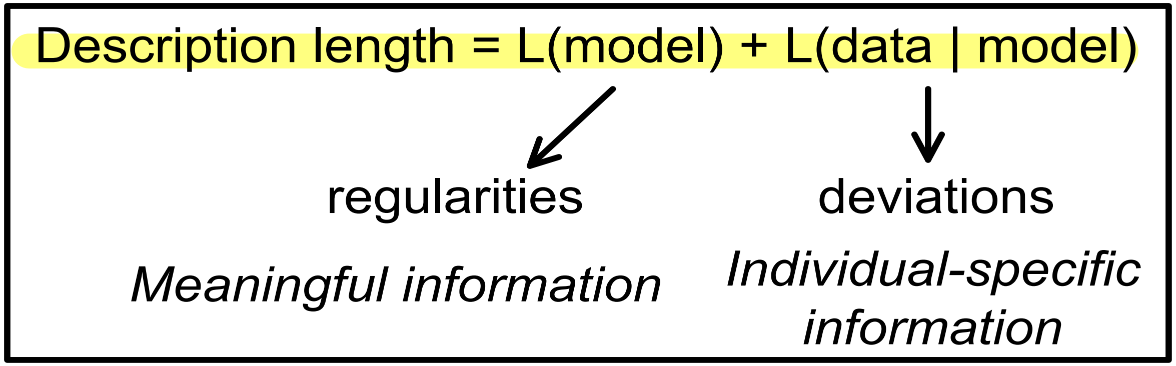
0110001101011010101:

- random string=incompressible=maximal information

010101010101010101:

- regularity of repetition allows compression
- if the training set is ‘representative’, then regularities in the training set will be representative for regularities in an unseen test set

A lot of forms of learning can be described as data compression in the following sense:



Description length = L(model) + L(data | model)

regularities

Meaningful information

deviations

Individual-specific information

Kolmogorov Complexity

The Kolmogorov complexity (K) of a binary object is the length of the shortest program that generates this object on a universal Turing machine.

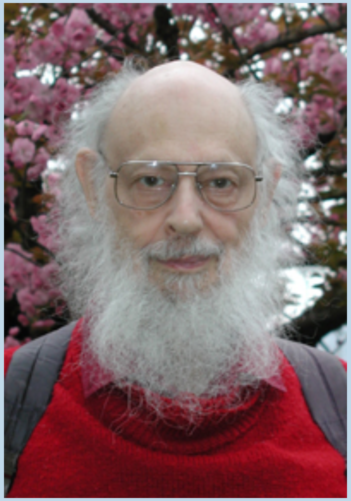
- Random strings are not compressible.
- A message with low Kolmogorov complexity is compressible.

K as complexity measure is incomputable.

- So in practical applications it always needs to be approximated, e.g. by Lempel-Ziv (used in zip and unzip) compression or others.

Ray Solomonoff founded the **theory of universal inductive inference**, which draws on Kolmogorov complexity. Kolmogorov and Solomonoff invented the concept in parallel.

Kolmogorov



Solomonoff

Kolmogorov Complexity

Let x be a finite length binary string and U a universal computer.

Let $U(p)$ be the output of U when presented with program p.

Let $l(p)$ be the length of program p.

The Kolmogorov complexity of string x is the minimal description length of x.

$$K_U(x) = \min_{p:U(p)=x} l(p)$$

· p that generates x with the minimum length

Kolmogorov Complexity is Not Computable

- Proof by contradiction (sketch):
  - suppose there is a **function** KolmogorovComplexity(**string** s)
  - then write a program that generates the shortest string w. Kolmogorov complexity $K(s) = n$

```
function GenerateComplexString(int n)
  for i = 1 to infinity:
      for each string s of length exactly i
        if KolmogorovComplexity(s) >= n return s
quit

function GenerateParadoxicalString()
      return GenerateComplexString($n_0$)
```

Length of GenerateComplexString: U
+ Number of bits to encode n_0: $\log_2(n_0)$
+ Overhead added by GenerateParadoxicalString
= $U + \log_2(n_0) + C$

$$U + \log_2(n_0) + C < n_0.$$

- GenerateComplexString generates the shortest string s with $K(s) = n_0$
- the length of this program is $< n_0$ if n_0 is large enough
- but s is not compressible, by definition of $K(s)$
- Contradiction!

The Minimum Description Length Principle

MDL principle is a model selection criterion (like AIC)

MDL restricts the set of allowed codes in such a way that it becomes possible (computable) to find the shortest code length of the data, relative to the allowed codes

The description length (DL) is defined as:

space required to describe a theory

+

space required to describe the theory’s mistakes

Learning = finding regularities = compression

Outline for today

- Bias-Variance Tradeoff
- Resampling Methods
- Gain and ROC Curves
- Comparison Studies
- Algorithmic Information Theory
- **Computational Learning Theory**

“All models are wrong; some are useful.”

- George E. P. Box -

Computational Learning Theory (Outlook)

Much of this theory is focused on supervised learning.

Sample complexity:
What is the number of training samples that we need to provide to an algorithm, so that a function returned by the algorithm is within an arbitrarily small error with high probability?

No free lunch theorem:
There is no algorithm that can learn the globally optimal target function using a finite number of training samples.

If we focus on specific target functions (e.g., linear functions), then the sample complexity is finite and it can be characterized. This can be used for **sample-complexity bounds**, i.e. bounds on the (training) sample size needed to achieve a particular test error (w/o seeing the test data).

$$\text{test error} \leq \text{training error} + \text{penalty}(d, n)$$

Testing different algorithms and target function is useful for a predictive modeling problem.