AutoML: Evaluation

Evaluation of ML Models (Review)

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Introduction

Performance estimation of a model Estimate generalization error of a model on new (unseen) data, drawn from the same data generating process.

Performance estimation of an algorithm Estimate generalization error of a learning algorithm, trained on a data set of a certain size, on new (unseen) data, all drawn from the same data generating process.

Model selection Select the best model from a set of potential candidate models (e.g., different model classes, different hyperparameter settings, different feature sets).

Performance Evaluation

ML performance evaluation provides clear and simple protocols for reliable model validation.

- often simpler than classical statistical model diagnosis
- relies only on few assumptions
- still hard enough and offers **lots** of options to cheat and make mistakes

Performance Measures

We measure performance using a statistical estimator for the **generalization error** (GE).
GE = expected loss of a fixed model
$\hat{\text{GE}}$ = estimated loss averaged across finite sample
Example: Mean squared error (L2 loss)

$$\hat{\text{GE}} = MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

Measures: Inner vs. Outer Loss

Inner loss = loss used in learning (training)
Outer loss = loss used in evaluation (testing)
= evaluation measure

Optimally: inner loss = outer loss

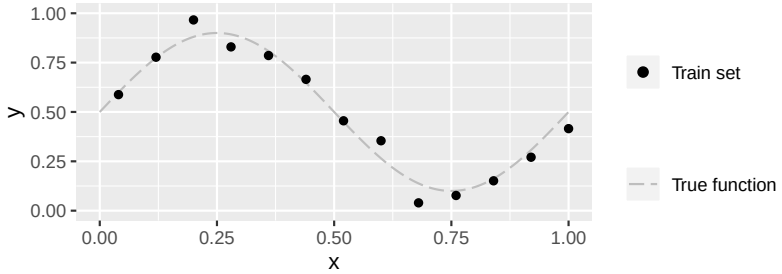
Not always possible:
some losses are hard to optimize or no loss is specified directly

Example:

Logistic Regression	$\rightarrow$ minimize binomial loss
kNN	$\rightarrow$ no explicit loss minimization

Inner Loss Example: Polynomial Regression I

Sample data from sinusoidal function $0.5 + 0.4 \cdot \sin(2\pi x) + \epsilon$ with measurement error ϵ.

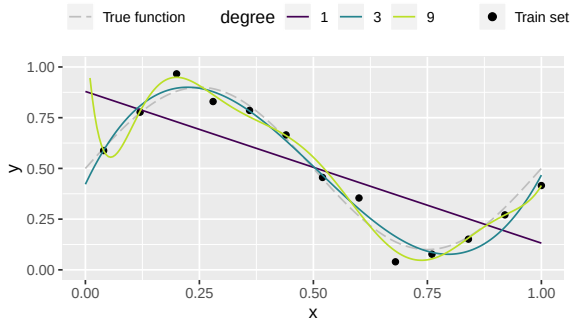


Assume data generating process unknown. Approximate with dth-degree polynomial:

$$f(\mathbf{x}|\theta) = \theta_0 + \theta_1 x + \cdots + \theta_d x^d = \sum_{j=0}^{d} \theta_j x^j$$

Inner Loss Example: Polynomial Regression II

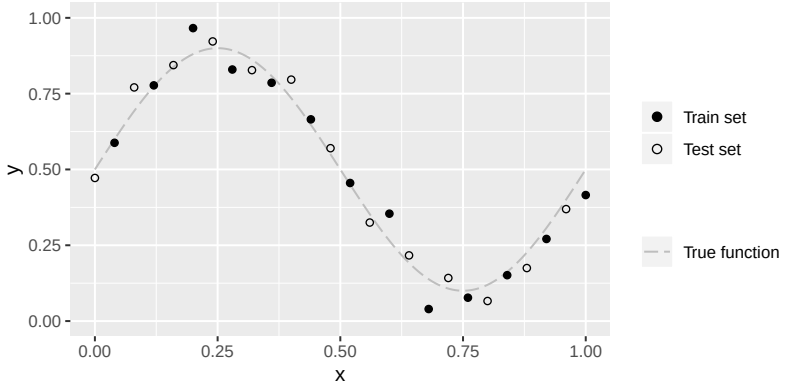
How should we choose d?



d=1: MSE $= 0.036$ – clear underfitting, d=3: MSE $= 0.003$ – ok?, d=9: MSE $= 0.001$ – clear overfitting

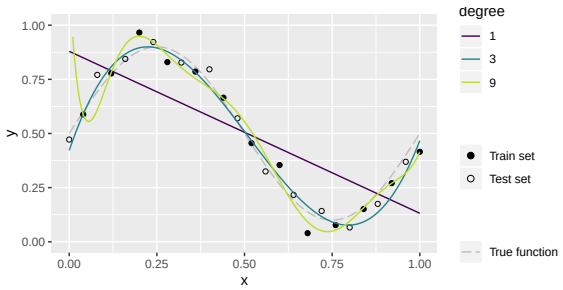
Simply using the training error seems to be a bad idea.

Outer Loss Example: Polynomial Regression I



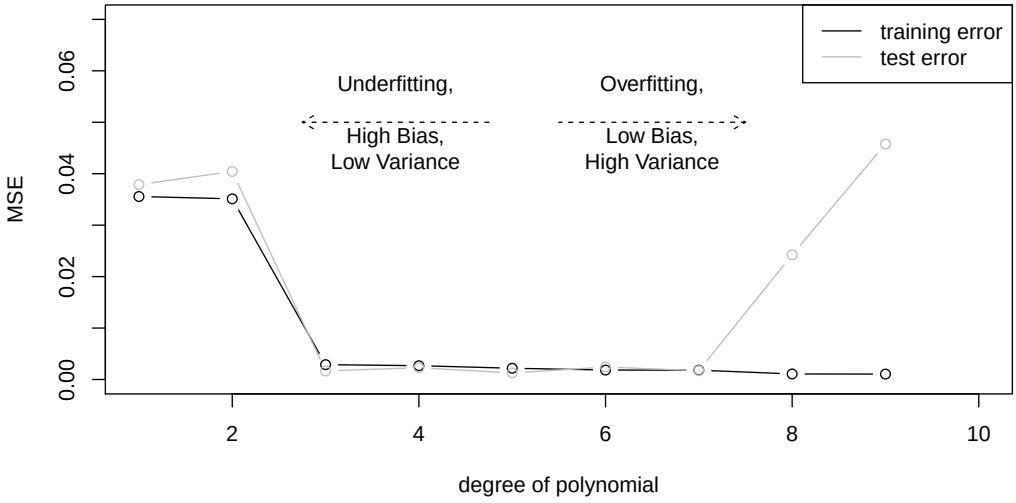
Outer Loss Example: Polynomial Regression II

How should we choose d?

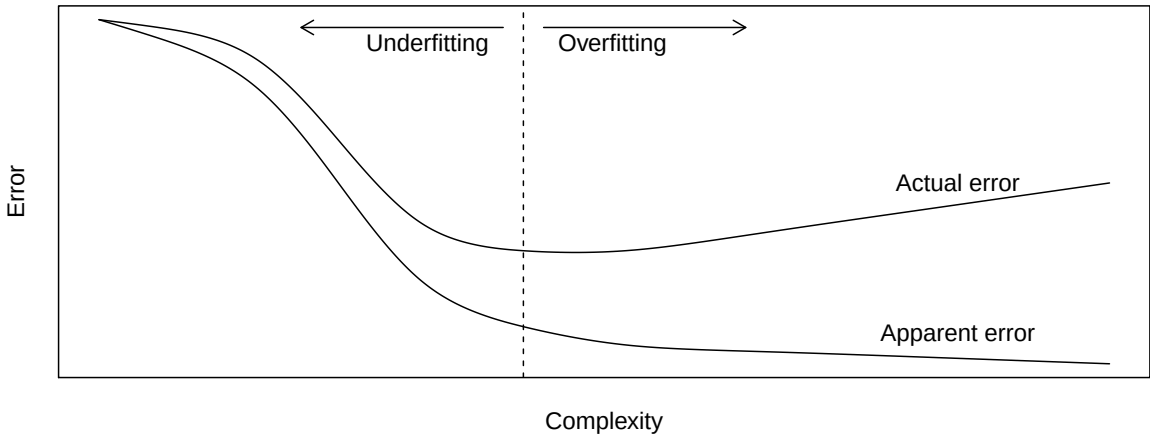


d=1: MSE = 0.038 – clear underfitting, d=3: MSE = 0.002 – ok?, d=9: MSE = 0.046 – clear overfitting

Outer Loss Example: Polynomial Regression III



General Trade-Off Between Error and Complexity



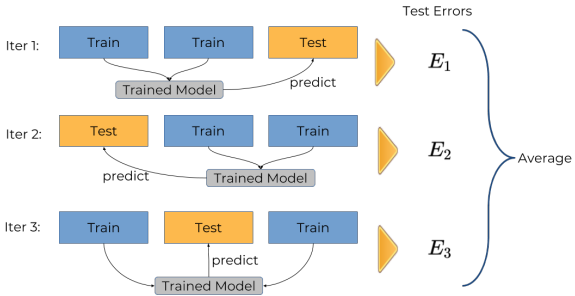
Resampling

- uses data efficiently
- repeatedly split in train and test, average results
- make training sets large (to keep the pessimistic bias small), reduce variance introduced by smaller test sets through many repetitions and averaging of results

Cross-Validation

- split data into k roughly equally-sized partitions
- use each part as test set and join the $k-1$ others for training, repeat for all k combinations
- obtain k test errors and average

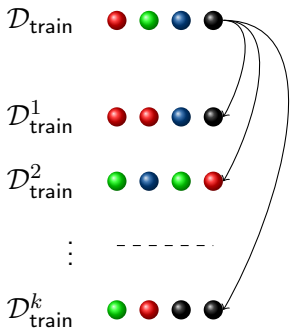
Example 3-fold cross-validation:



10-fold cross-validation is common.

Bootstrap

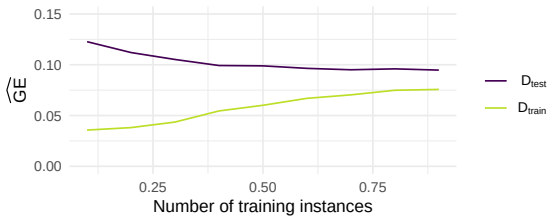
- randomly draw k training sets of size n with replacement from the data
- evaluate on observations from the original data that are not in the training set
- obtain k test errors and average



Training sets will contain about 63.2% of observations on average.

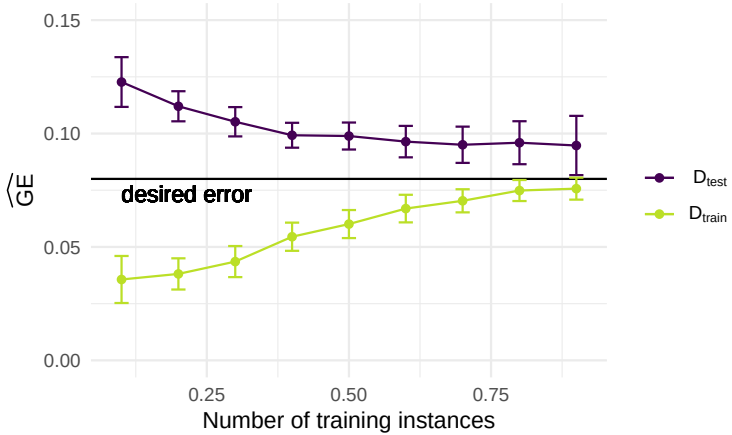
Learning Curves I

- compares performance of a model on training and test data over a varying number of training instances $\rightarrow$ how fast can a learner learn the given relationship in the data?
- can also be over number of iterations of a learner (e.g. epochs in deep learning), or AutoML system over time
- learning usually fast in the beginning
- visualizes when a learner has learned as much as it can:
  - when performance on training and test set reach a plateau
  - when gap between training and test error remains the same



Learning Curves II

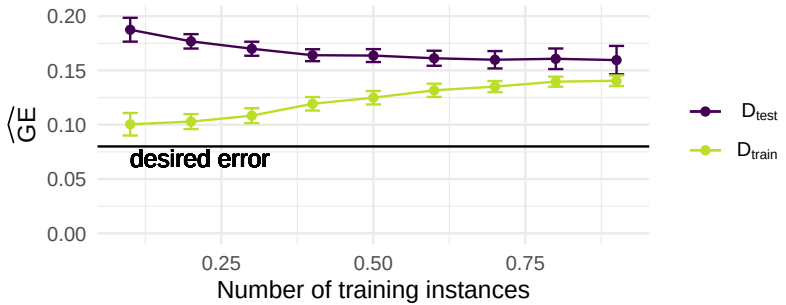
Ideal learning curve:



Learning Curves III

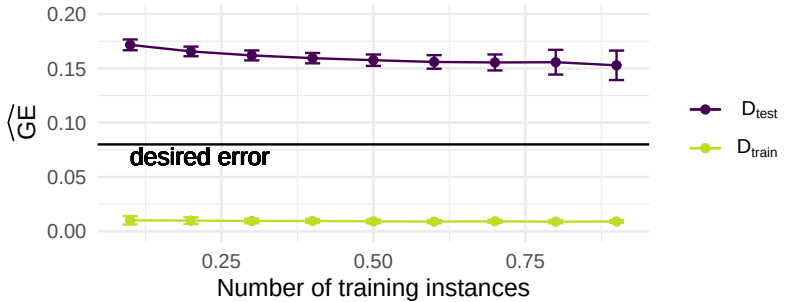
In general, there are two reasons for a bad learning curve:

1. high bias in model/underfitting
   - training and test errors converge at a high value
   - model can't learn underlying relationship and has high systematic errors, no matter how big the training set
   - poor fit, which also translates to high test error



Learning Curves IV

2. high variance in model/overfitting
   - large gap between training and test errors
   - model requires more training data to improve
   - model has a poor fit and does not generalize well



AutoML: Evaluation

Nested Resampling

Bernd Bischl[1] Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

[1] Some slides taken from Bernd Bischl's "Introduction to Machine Learning" lecture at LMU. [Bischl]

Motivation

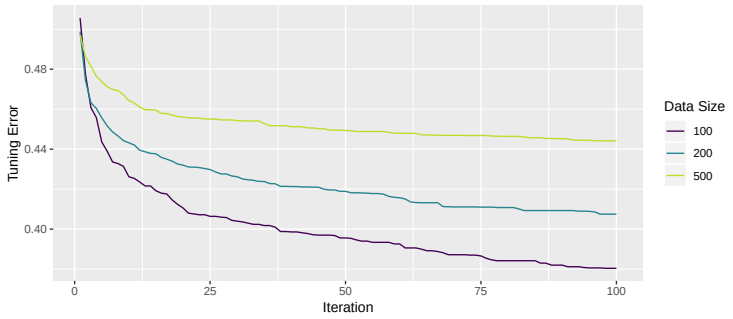
Selecting the best model from a set of potential candidates (e.g. different classes of learners, different hyperparameter settings, different feature sets, different preprocessing...) is an important part of most machine learning problems. However,

- cannot evaluate selected learner on the same resampling splits used to select it
- repeatedly evaluating learner on same test set or same CV splits "leaks" information about test set into evaluation
- danger of overfitting to the resampling splits or overtuning
- final performance estimate would be optimistically biased
- similar to multiple hypothesis testing

Motivating Example I

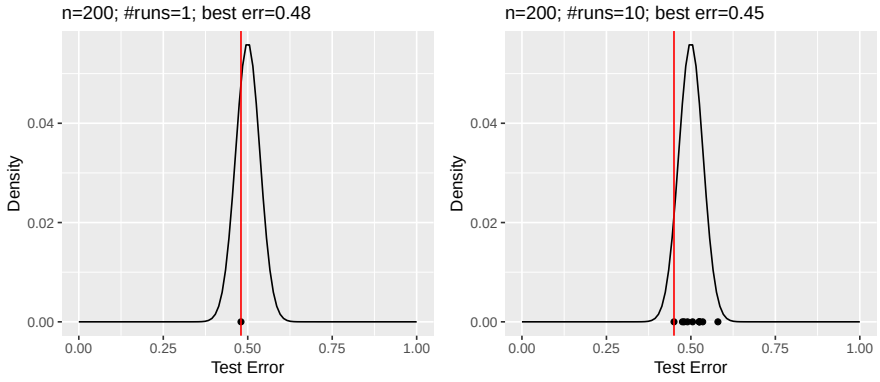
- binary classification problem with equal class sizes
- learner with hyperparameter $\boldsymbol{\lambda}$
- learner is (nonsense) feature-independent classifier that picks random labels with equal probability, $\boldsymbol{\lambda}$ has no effect
- true generalization error is 50%
- cross-validation of learner (with any fixed $\boldsymbol{\lambda}$) will easily show this (if the partitioned data set for CV is not too small)
- let's "tune" it by trying out 100 different $\boldsymbol{\lambda}$ values
- repeat this experiment 50 times and average results

Motivating Example II



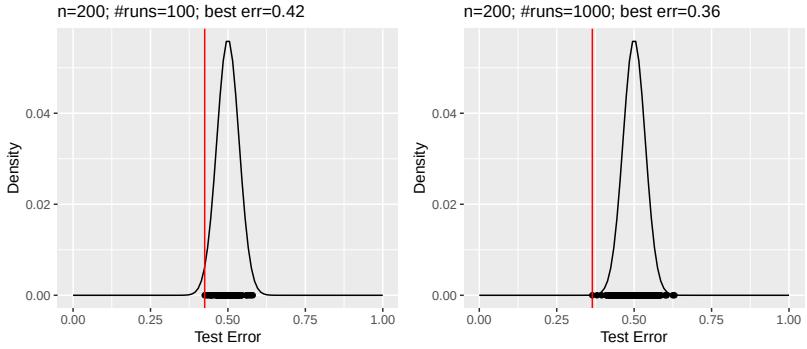
- shown is best "tuning error" (i.e. performance of model with fixed $\boldsymbol{\lambda}$ in cross-validation) after k tuning iterations
- evaluated for different data set sizes

Motivating Example III



- for one experiment, CV score is close to 0.5, as expected
- errors essentially sampled from (rescaled) binomial distribution
- scores from multiple experiments also arranged around expected mean of 0.5

Motivating Example IV



- tuning means we take the minimum of the scores
- not estimate of average performance, but best-case performance
- the more we sample, the more "biased" this value becomes $\rightarrow$ unrealistic generalization performance estimate

Untouched Test Set Principle I

Instead: simulate what actually happens when applying machine learning models

- all parts of model construction (including model selection, preprocessing) evaluated **on training data**
- test set only touched once, so no way of "cheating"
- test dataset is only used once *after* model is completely trained (including e.g. deciding hyper-parameter values)
- performance estimates from test set now **unbiased estimates** of the true performance

Untouched Test Set Principle II

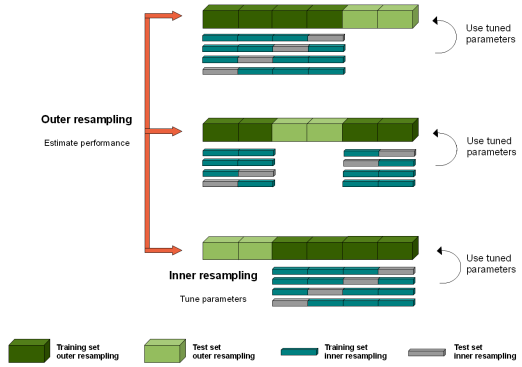
- for steps that themselves require resampling (e.g. hyperparameter tuning) this results in **nested resampling**, i.e. resampling strategies for both
  - inner evaluation to find what works best based on training data
  - outer evaluation on data not used in inner evaluation to get unbiased estimates of expected performance on new data

Nested Resampling I

- holdout can be generalized to resampling for more reliable generalization performance estimates
- resampling can be generalized to nested resampling
- nested resampling loops for inner and outer evaluation for hyperparameter tuning

Nested Resampling II

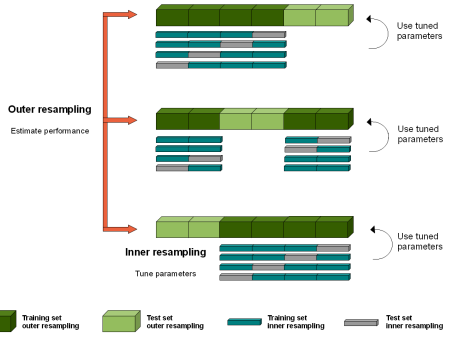
Example: four-fold CV for inner resampling, three-fold CV in outer resampling



Nested Resampling III

In each iteration of the outer loop:

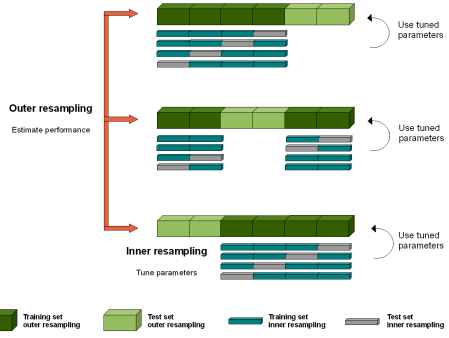
- split light green testing data
- run hyperparameter tuner on dark green part of data, i.e. evaluate each $\boldsymbol{\lambda}_i$ through four-fold CV on dark green part



Nested Resampling IV

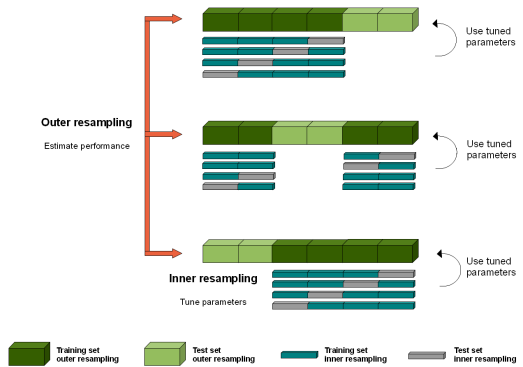
In each iteration of the outer loop:

- return winning $\hat{\boldsymbol{\lambda}}$ that performed best on the grey inner test sets
- re-train model on full outer dark green training set
- evaluate model on outer light green test set



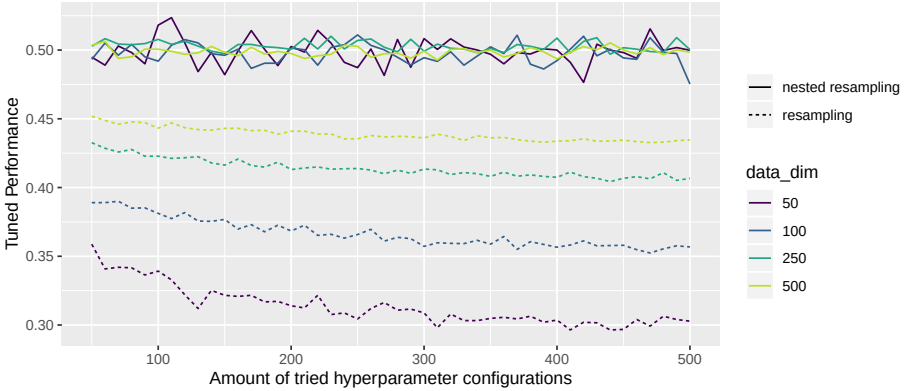
Nested Resampling V

$\rightarrow$ error estimates on outer samples (light green) are unbiased because this data was not used process constructing the tested model



Nested Resampling Example

Revisited motivating example: expected performance estimate with nested resampling:



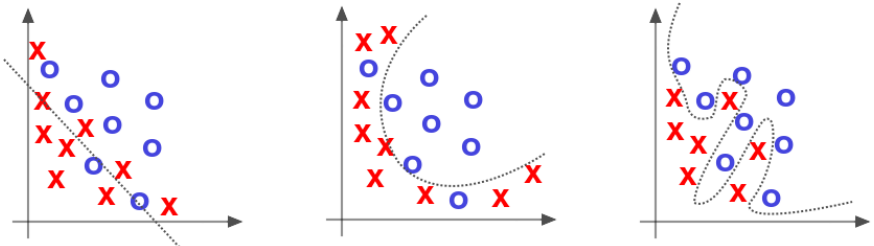
AutoML: Evaluation

Overview and Motivation

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Training Machine Learning Models

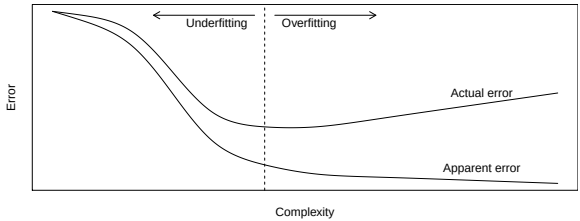
- fundamentally an optimization problem
- determine model parameters such that loss on data is minimized
- quality of fit depends on model class (i.e. degrees of freedom)



Which model is best?

Generalization

- we want models that *generalize* – make "reasonable" predictions on new data
  - ignore outliers
  - smooth
  - captures general trend

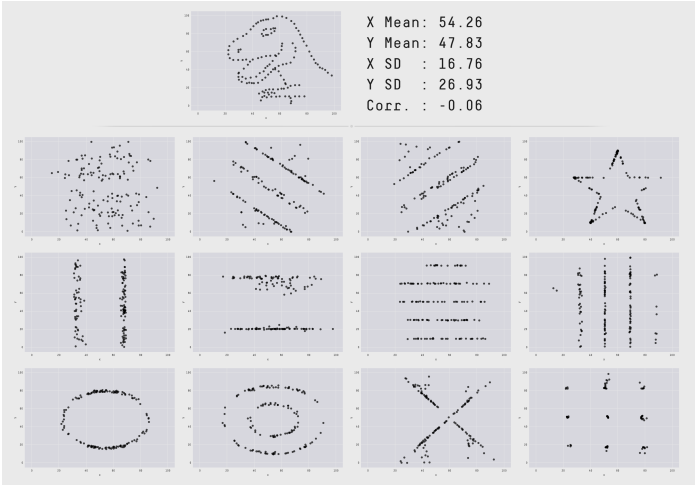


Usually model performance gets better with more data/higher model complexity and then worse, but see [Nakkiran et al. 2019]

Overview

- evaluating machine learning models and quantifying generalization performance
- learning curves
- comparing multiple models/learners on multiple data sets
- statistical tests
- higher levels of optimization, higher levels of evaluation
  - automated machine learning (meta-optimization) can lead to meta-overfitting
  - simple training/testing split(s) no longer sufficient $\rightarrow$ nested evaluation

Evaluate Early, Evaluate Often



https://www.autodeskresearch.com/publications/samestats