LTAT.02.004 Machine Learning II

Performance evaluation

Sven Laur
University of Tartu

Why do we estimate performance?

- ▷ To estimate how does the algorithm perform in the future
  - ⋄ This is the most important question in the practice
  - ⋄ We are interested on performance of a particular predictor
- ▷ To find the best hyperparameter instance for our dataset
  - ⋄ It is quite tricky task if we consider all subtleties
  - ⋄ We are comparing different algorithm instances on our data
- ▷ To compare different algorithms and choose the best
  - ⋄ This is needed to justify the development of a new algorithm
  - ⋄ We are comparing average behaviour of algorithms
- ▷ To see if there is a dependence between input and the output
  - ⋄ Studies in biology or sociology are all about causal dependencies
  - ⋄ We are interested in statistically significant performance levels

Short list of goodness measures

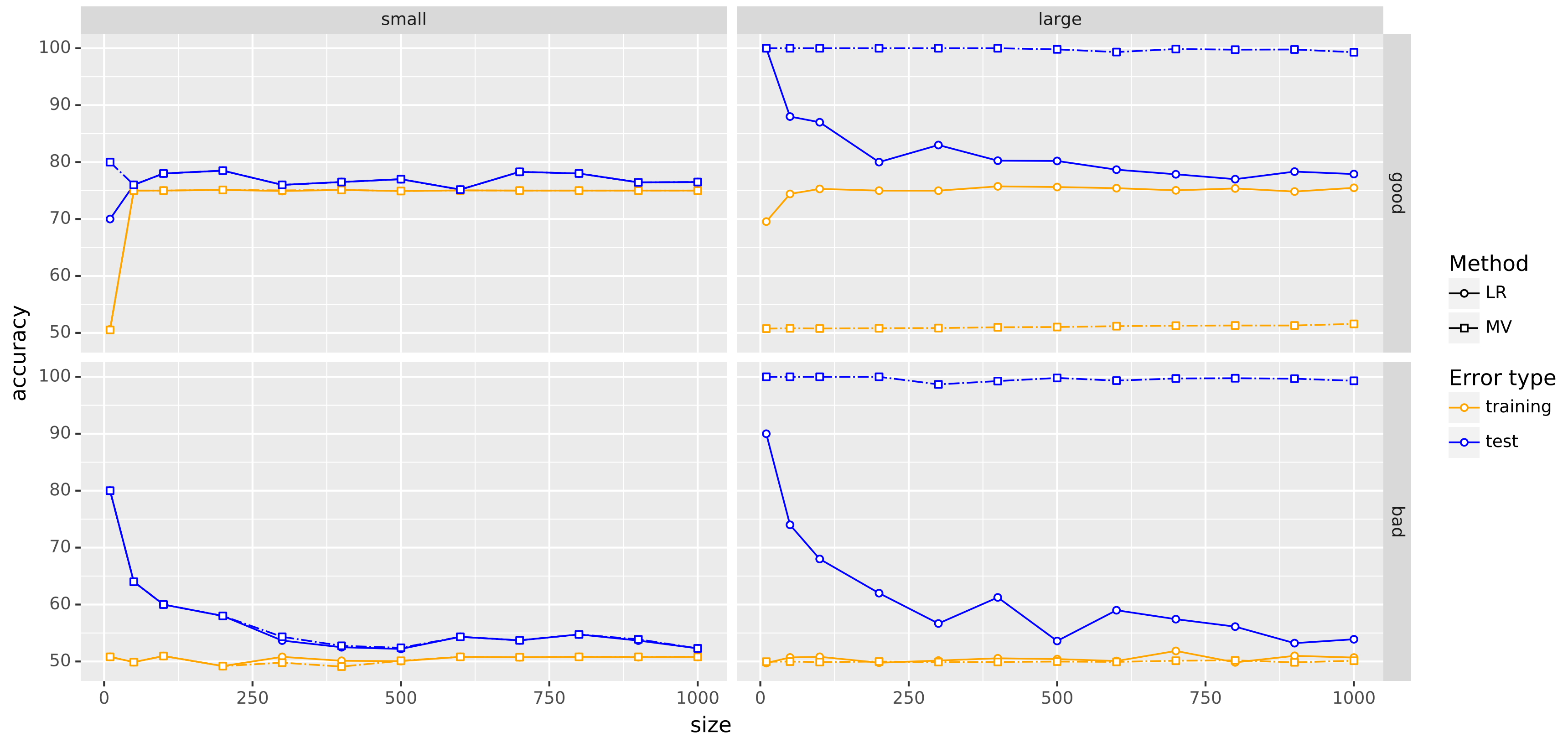
Some goodness measures for classification

- ▷ Accuracy – the percentage of correctly classified observations
- ▷ Precision – the percentage of correct labels among positive guesses
- ▷ Recall – the percentage of positive cases that are detected

Some goodness measures for regression

- ▷ Normalised mean square error
- ▷ Normalised mean absolute error
- ▷ Trimmed mean square and absolute error estimates

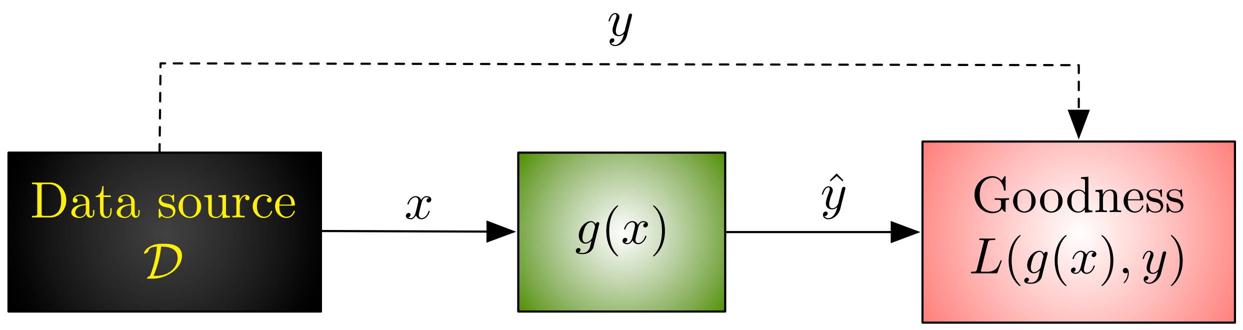
Performance



▷ depends on data and a target function

▷ depends on the size of training data and method itself

How to estimate performance in the future



For any prediction algorithm we can find its expected goodness in the future.

Practice. Average goodness over a long enough series of future samples.

▷ Sampling should not change the data source in the future.

▷ All future samples should be independent from each other.

Theory. We should find expected goodness over the data distribution.

▷ The distribution always exists although we might not know it.

▷ Expected value exists even if the number of future samples is limited.

Are these assumptions satisfied in practice?

Assumption I. Data distribution does not change

▷ Some changes in data can be modelled

▷ If radical changes occur the model must be retrained

▷ Sometimes predictions must be valid regardless of inputs

Assumption II. Future samples are independent from each other

▷ This assumption is always violated in text analysis

▷ This assumption is always violated in time-series analysis

▷ Correlation between future samples creates overconfidence

▷ This effect can be corrected with more careful sampling of a test set

Notation and terminology

Spaces

- ▷ $\mathcal{D}$ – data distribution
- ▷ $\mathcal{X}$ – input space, feature space
- ▷ $\mathcal{Y}$ – output space, target space
- ▷ $\mathcal{F} \subseteq \{f : \mathcal{X} \times \Omega \to \mathcal{Y}\}$ – model class

Instances

- ▷ $\boldsymbol{x} \in \mathcal{X}$ – instance
- ▷ y – true value of an instance, target value
- ▷ $\hat{y} = f(\boldsymbol{x})$ – predicted target value

Loss:

- ▷ $L : \mathcal{Y} \times \mathcal{Y} \to \mathcal{R}$ – the cost of using prediction $\hat{y}$ instead of y

Theoretical formulation

Let $\mathcal{D}$ be the distribution of (x, y) pairs where x is the input and y is the target of a prediction algorithm f.

Let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be the *loss function* which takes in the predicted value $\hat{y}$ and the actual value y and outputs resulting loss.

Then the corresponding *risk* $R(f)$ is computed as *mathematical expectation*

$$R(f) := \mathop{\mathbf{E}}_{\mathcal{D}}(L(f(x), y)) = \int\limits_{(x,y)\in\mathcal{D}} L(f(x), y) dF(x, y)$$

where F is the corresponding probability measure.

Practical example

▷ Let $f(x_1, x_2) \equiv 0$ and let $L(\hat{y}, y) = (y - \hat{y})^2$. What is the risk $R(f)$ if the next data sample is chosen uniformly from the following table.

x_1	x_2	y
0	0	0
0	0	1
0	1	1
1	0	0
1	1	1
0	0	0

▷ Propose a new prediction rule f_* that minimises the risk.

▷ Is there always a prediction rule that minimises the risk?

Empirical risk estimation

When the sample $D_N = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$ is *representative* then we can approximate risk $R(f)$ with *empirical risk*:

$$R_N(f) = \frac{1}{N} \cdot \sum_{i=1}^{N} L(f(\boldsymbol{x}_i), y_i) \quad .$$

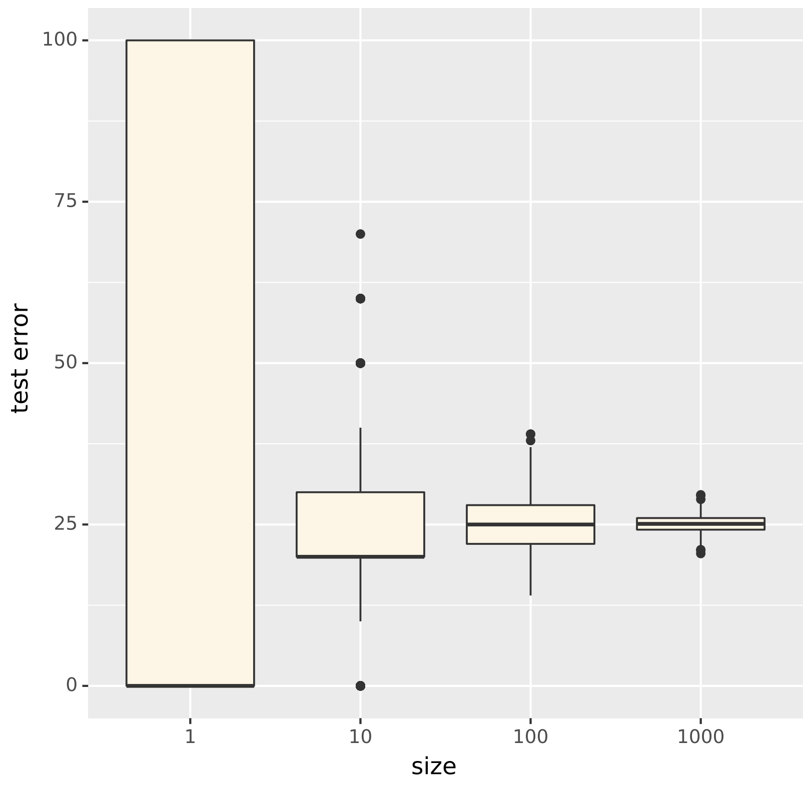
IID sampling assumption. The following conditions assure that the sample data D_N is representative (with high probability).

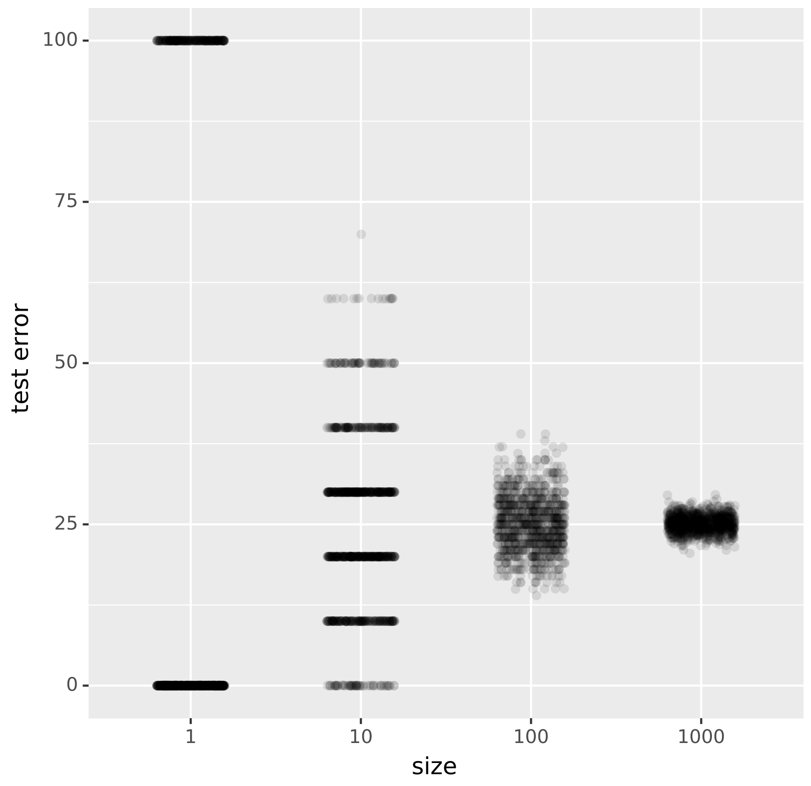
▷ All samples are independent from each other.

▷ All samples are drawn from the same distribution.

▷ Future samples come from the same distribution as the data D_N.

Empirical risk





▷ depends on the dataset

▷ statistical fluctuations decrease with size

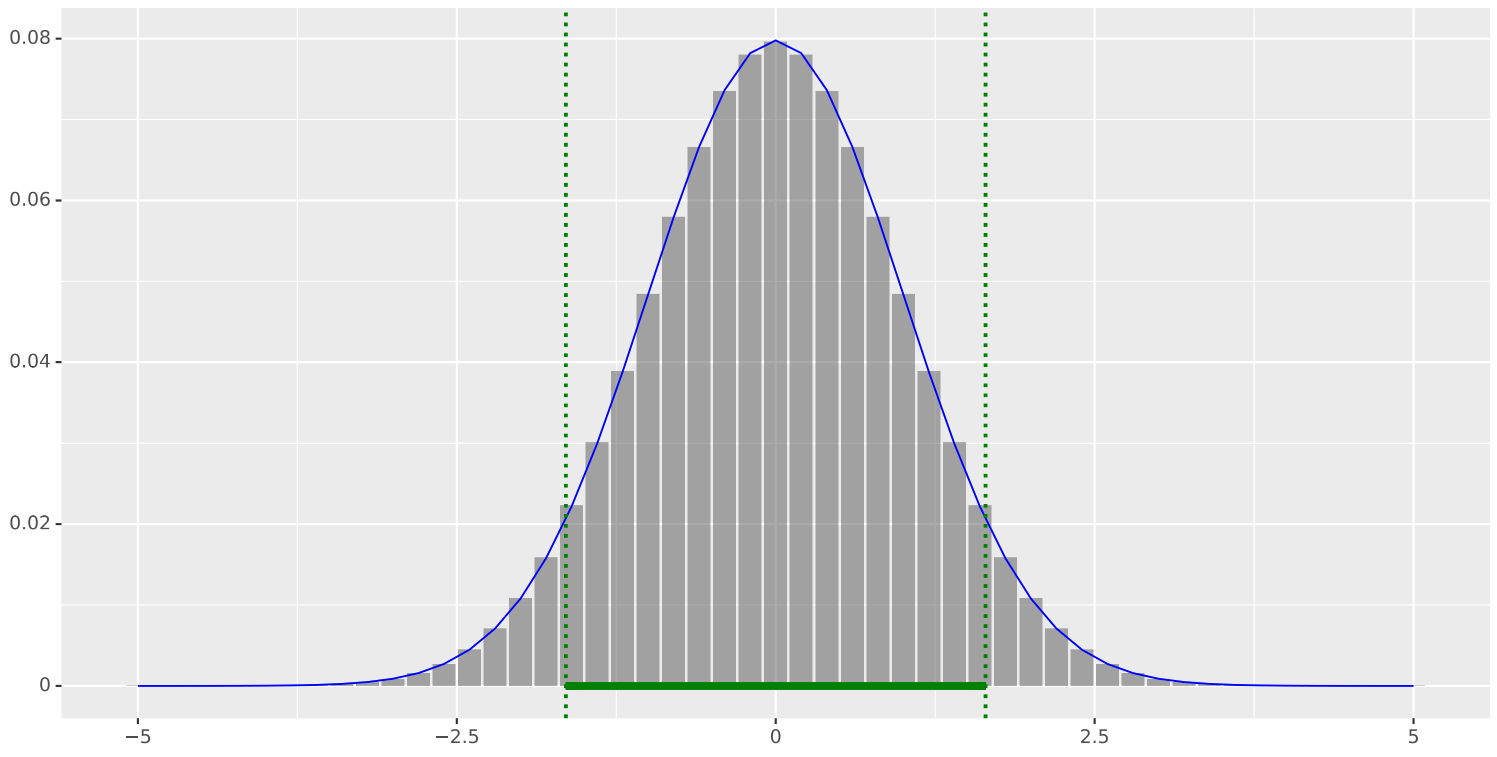
Law of large numbers

Central limit theorem. Let $z_1, \ldots, z_N$ be independent and identically distributed samples form a *real-valued distribution* with a *finite standard deviation* σ and *mean* μ. Then the random variable

$$S = \sqrt{N}\left(\frac{1}{N} \cdot \sum_{i=1}^{N} z_i - \mu\right)$$

converges *in distribution* to normal distribution $\mathcal{N}(mean = 0, sd = \sigma)$.

Visual representation



Convergence implies that the centre area of is well approximated

▷ 90% confidence intervals are roughly the same for both distributions

Translation

Under mild assumptions the empirical risk $R_N(f)$ converges to risk $R(f)$ and we can actually use normal distribution to estimate probabilities:

$$\Pr\left[|R_N(f) - R(f)| \geq \varepsilon\right] \lesssim 2 \cdot \int\limits_{-\infty}^{\varepsilon} \frac{\sqrt{N}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{Nt^2}{2\sigma^2}\right) dt$$

for a finite value σ where σ^2 is the variance of loss $\mathbf{D}(R(f))$.

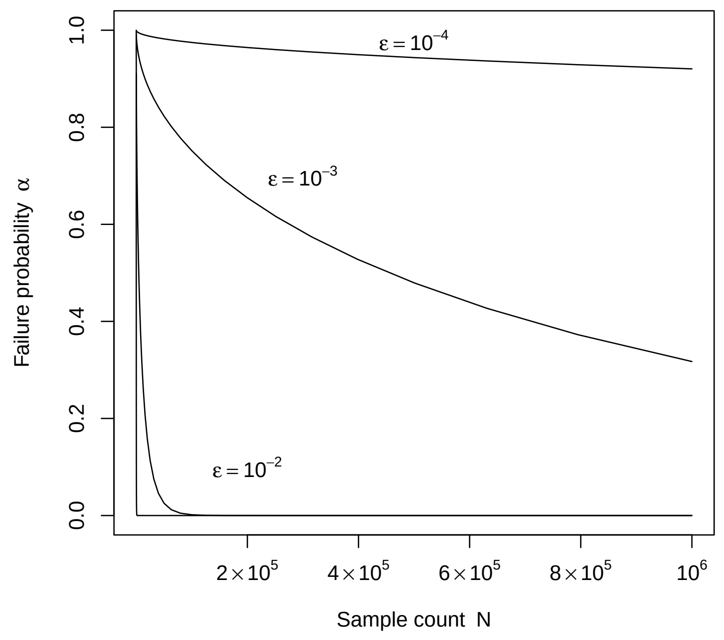
Reasoning

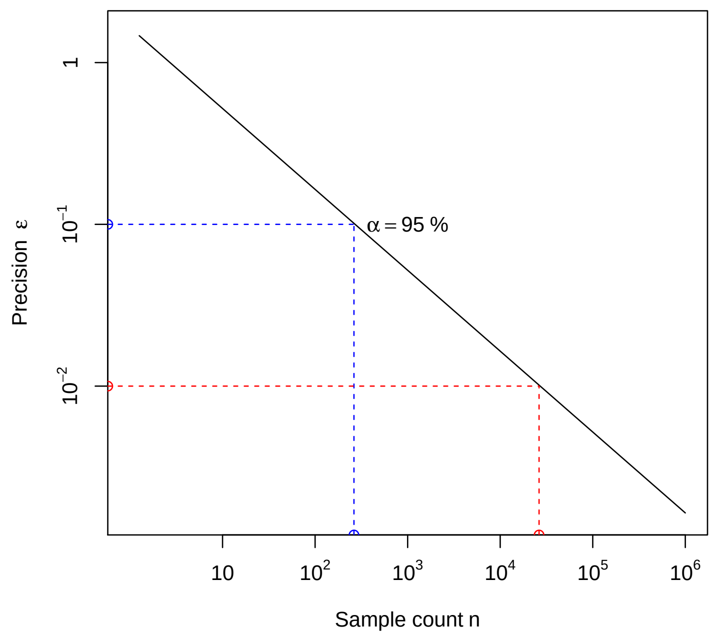
▷ If $(\boldsymbol{x}_i, y_i)$ are IID samples then $z_i = L(f(\boldsymbol{x}_i), y_i)$ are also IID samples.

▷ By definition $\mu = \mathbf{E}(z) = \mathbf{E}(L(f(\boldsymbol{x}), y)) = R(f)$.

▷ CLT assumes that risk μ is finite and standard deviation σ is finite.

What does the convergence speed mean





The number of samples needed to get a precision ε is $\mathrm{O}(1/\varepsilon^2)$.

▷ To increase precision 10 times you need 100 times more samples!

Why do we need a test set at all

Machine learning algorithm

- ▷ Count number of zeroes n_0 and number of ones n_1 in training sample.
- ▷ If $n_0 > n_1$ output $f_0(x) \equiv 0$, otherwise output $f_1(x) \equiv 1$.

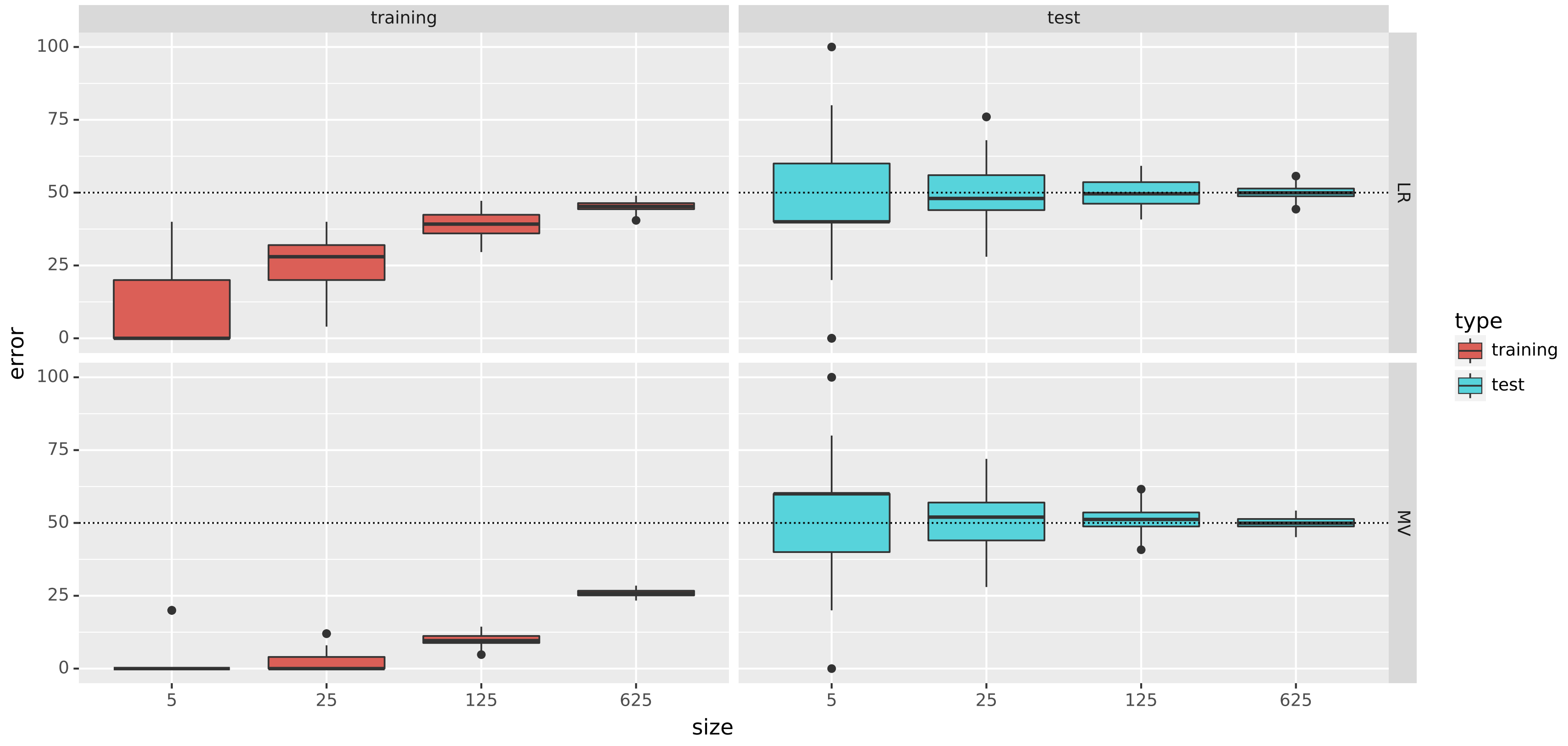
Data source

- ▷ Choose the input x randomly form the range $[0, 1]$
- ▷ Choose the label y randomly from the set $\{0, 1\}$.

True risk value

- ▷ Clearly the risk of both rules $R(f_0) = R(f_1) = 0.5$.
- ▷ The risk of of our learning algorithm $R(f)$ is also 0.5.

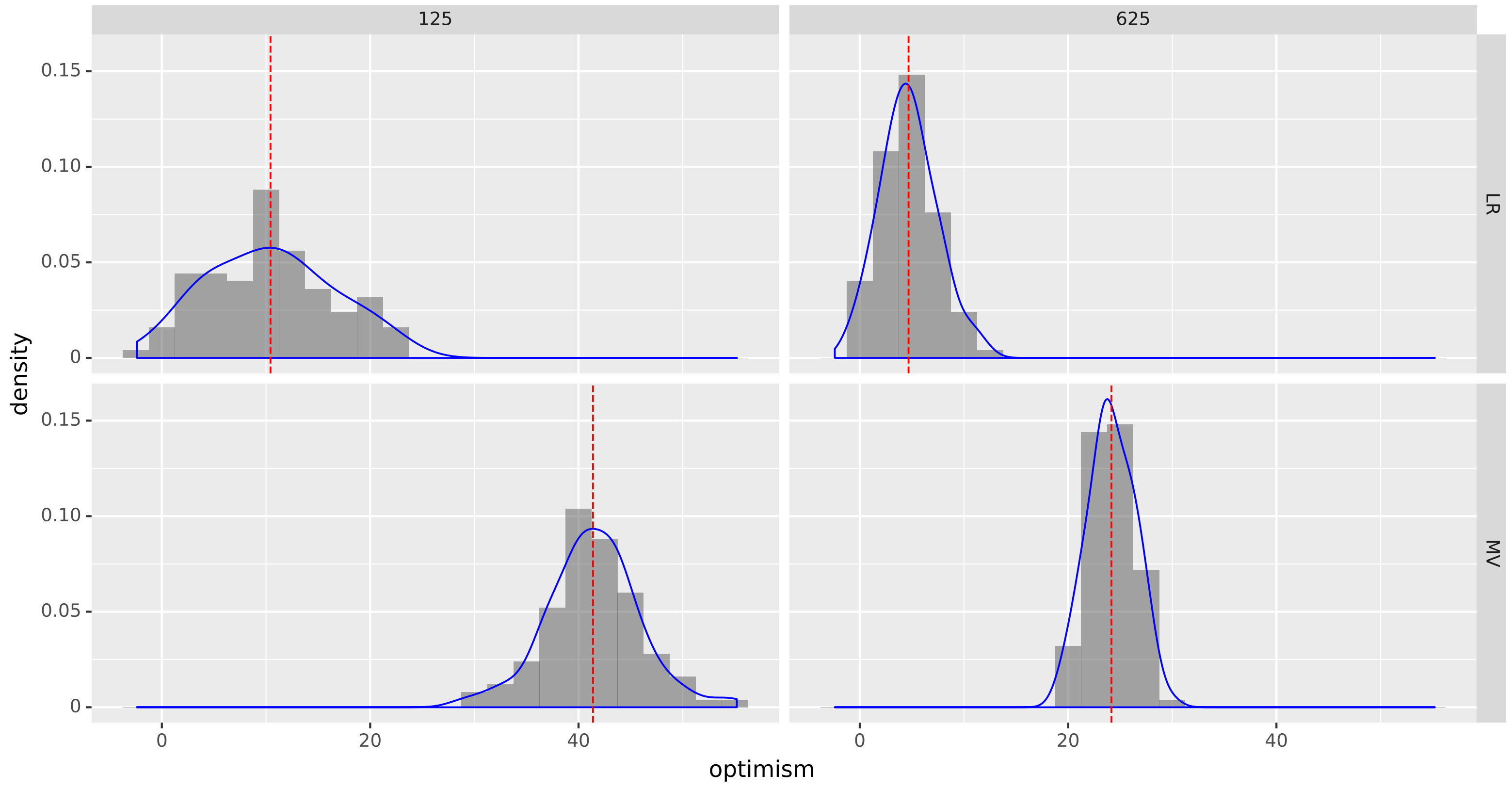
Simulation outcomes for other methods



Training error of the rule f is significantly smaller than 0.5.

▷ We bias the estimate by choosing the rule g_i for which $R_N(f_i) < R(f)$.

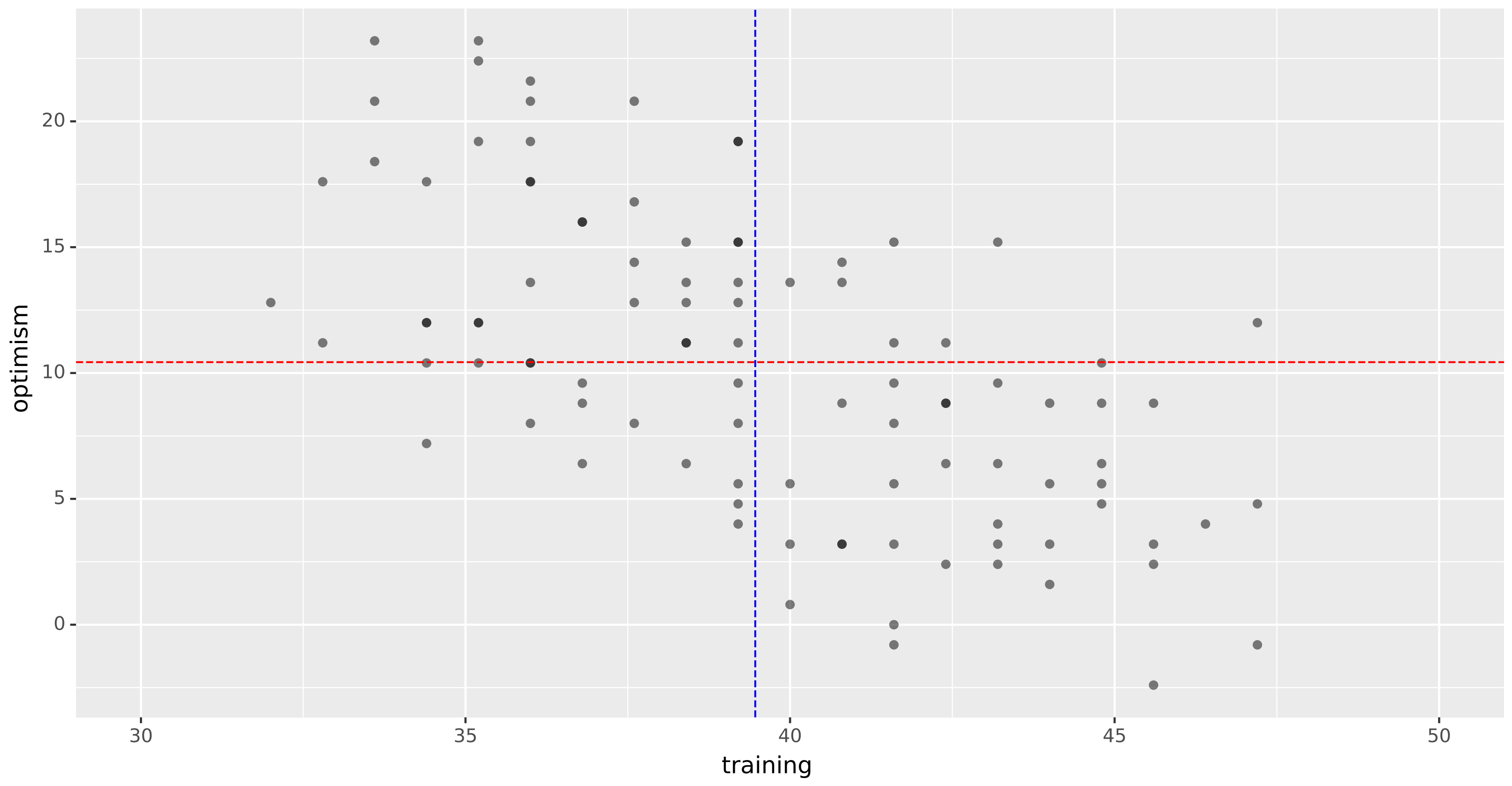
Optimism



By knowing the *optimism* $\Delta = R(f) - R_N(f_i)$ we can correct $R_N(f)$.

▷ Commonly mean value of Δ is used for the correction

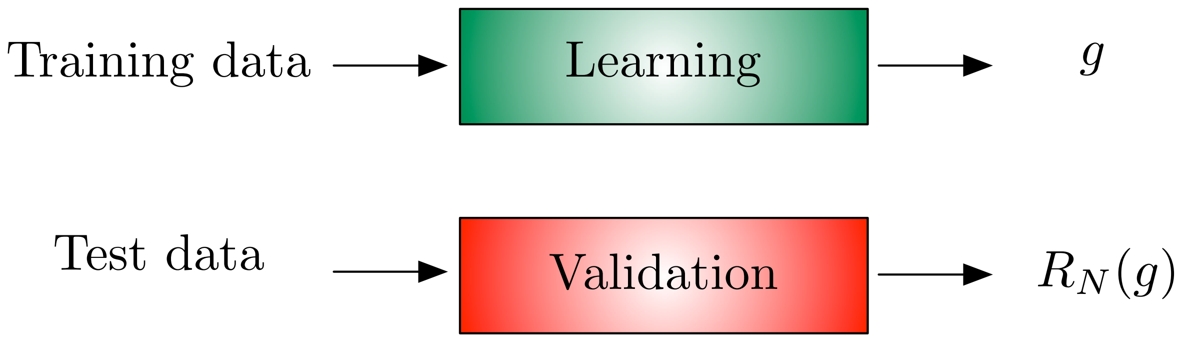
Optimism is only approximation



Optimism is usually anti-correlated with empirical risk $R_N(f)$

$\triangleright$ Simple shifting does not resolve the systematical bias

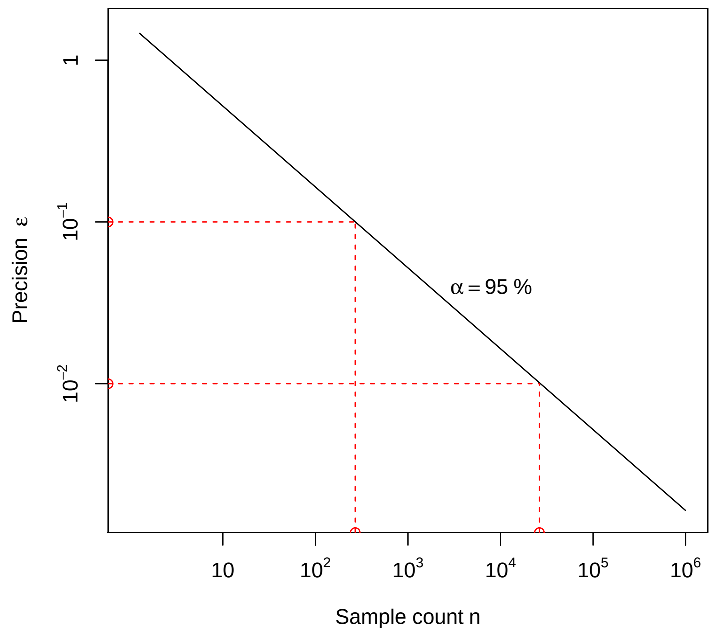
Why does the holdout testing work



By randomly splitting the data into training and test data we assure

- ▷ The training and test sets are independent under IID assumption.
- ▷ On a training set we compare many models and choose few winners.
- ▷ These functions are independent from the test set data.
- ▷ As there number of functions is small the law of large numbers holds.

What is the right size of the holdout sample



The holdout sample must be quite large or otherwise the precision is low

▷ Roughly 400 data points to get precision 0.1 in classification accuracy.

Moment matching

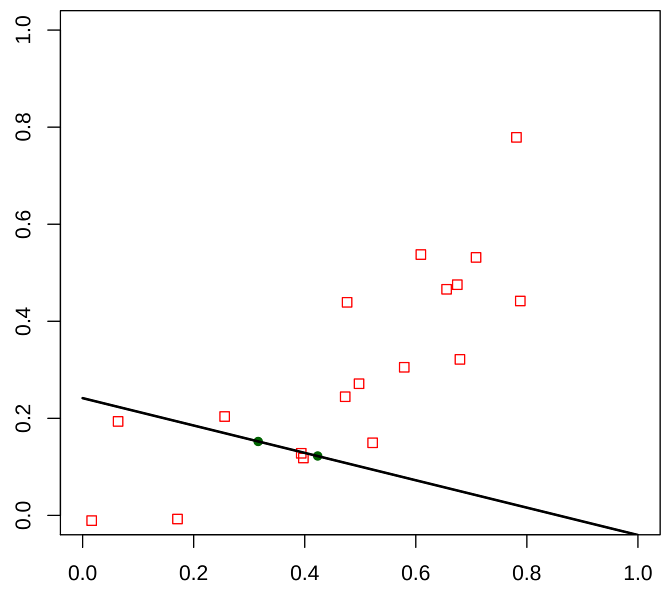
We know that the empirical risk $R_N(f)$ converges to normal distribution

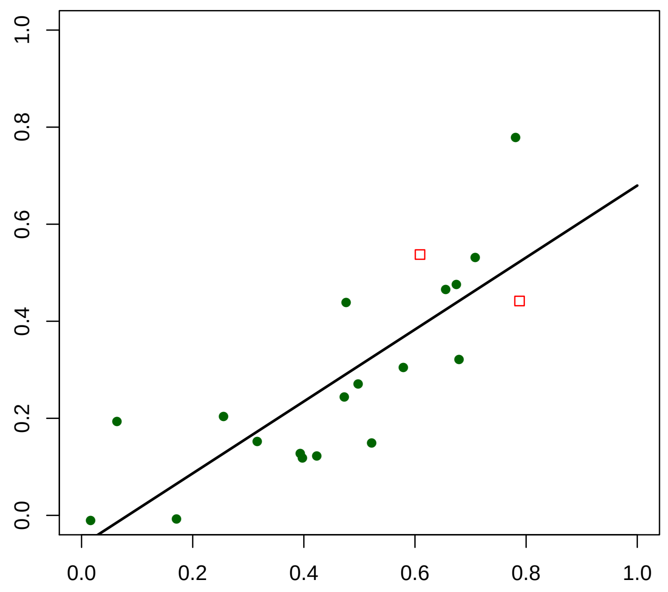
- ▷ Normal distribution is fixed by a mean μ and variance σ^2
- ▷ We can estimate mean $\hat{\mu}$ and variance $\hat{\sigma}^2$ of a loss term $L(f(\boldsymbol{x}), y)$
- ▷ Then the estimates of mean and variance of the empirical risk are

$$\mathbf{E}(R_N(f)) \approx \hat{\mu}$$
$$\mathbf{D}(R_N(f)) \approx \frac{\hat{\sigma}^2}{N}$$

- ▷ This allows us to approximate $R_N(f)$ with normal distribution

Why is holdout testing problematic



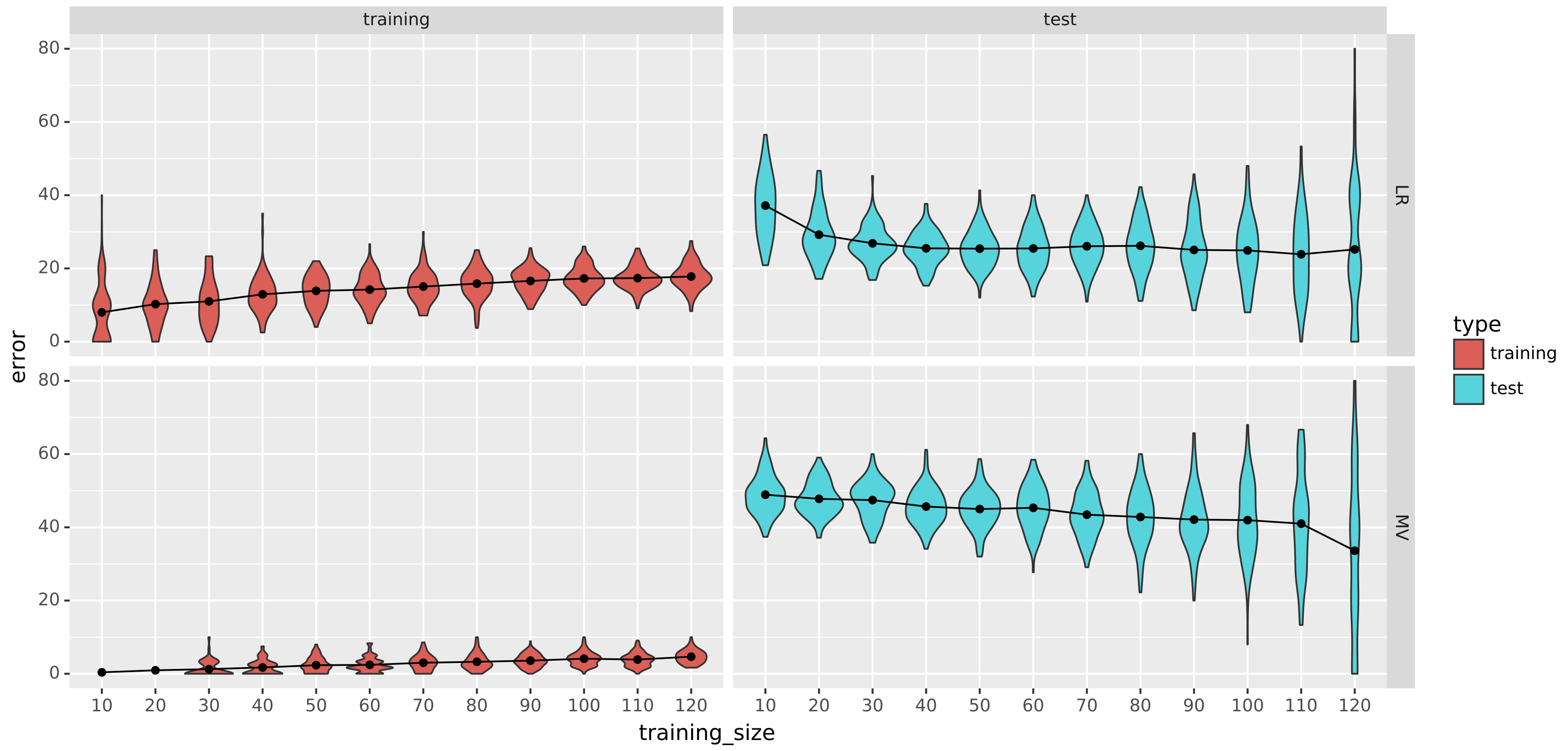


If the number of available data points is small we have to choose:

▷ a small training set and bad model but good estimate on risk

▷ a big training set and good model but bad estimate on risk

Why is holdout testing problematic



Typical tradeoffs between learning-bias and variance of the validation error.

Crossvalidation as an engineering trick

To reduce holdout error, we can do several holdout experiments. Since we have not enough data, we redo splitting and training on the same data.

This idea yields a generic crossvalidation scheme

1. Generate several splits of test and training data
2. For each split train the model and compute holdout error
3. Tabulate results

	Split 1	Split 2	$\ldots$	Split k
Training error	S_1	S_2	$\ldots$	S_k
Test error	E_1	E_2	$\ldots$	E_k
Optimism Δ	$E_1 - S_1$	$E_2 - S_2$	$\ldots$	$E_k - S_k$

4. Compute averages $E = \frac{1}{k}(E_1 + \cdots + E_k)$ and $\Delta = \frac{1}{k}(\Delta_1 + \cdots + \Delta_k)$
5. Visualise results and compute confidence intervals for estimates if needed.

What does crossvalidation measure?

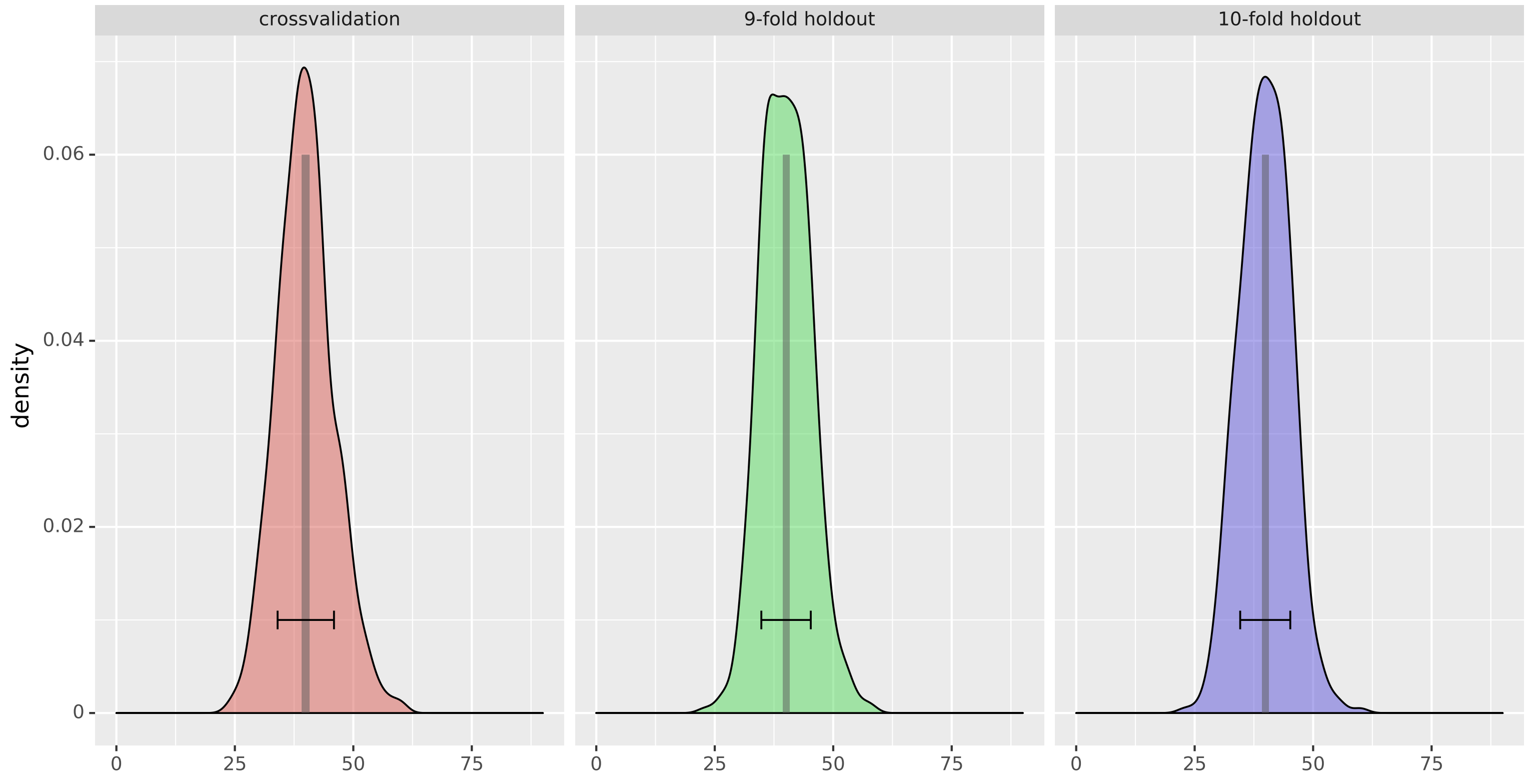
For each fold we have a separate predictor f_i and test error E_i:

- ▷ Average E characterises average behaviour of $f_1, \ldots, f_k$.
- ▷ Algorithm can use only (1-1/k) fraction of the available data.
- ▷ If there is not enough data for training E overestimates the error.

To estimate the performance of a classifier f trained on the entire data:

- ▷ We must estimate the difference between test and training error $\Delta(f)$.
- ▷ For normal ML algorithm optimism decreases by increasing the size n.
- ▷ Crossvalidation estimates Δ at the point $(1 - 1/k) \cdot n \lesssim n$.
- ▷ Hence we can go from training error to test error estimate.
- ▷ Training and test set fluctuations influence the outcome.

Crossvalidation vs holdout estimates



- ▷ Crossvalidation error is slightly larger as the training set is smaller.
- ▷ Crossvalidation error is slightly more fluctuating due to correlations.
- ▷ Quite often these effects are quite small in practice.

Theoretical explanation

Theorem. Crossvalidation error $E = \frac{1}{k}(E_1 + \cdots + E_k)$ is an unbiased estimate for the average test error that is taken over all models that are trained on $(1 - 1/k) \cdot n$ samples.

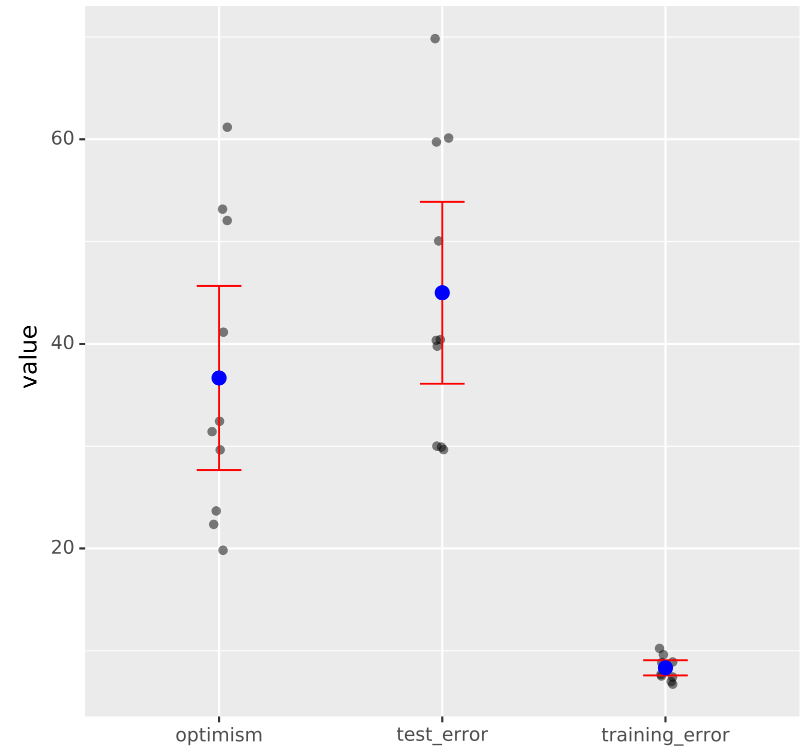
Proof

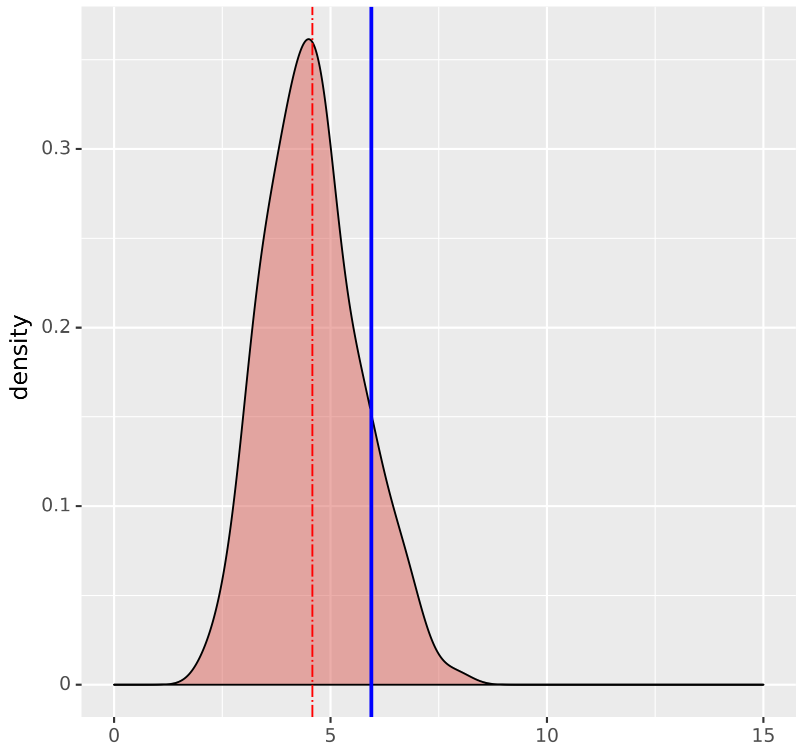
$$\mathbf{E}[E] = \frac{\mathbf{E}[E_1] + \cdots + \mathbf{E}[E_k]}{k} = \mathop{\mathbf{E}}_{tr}\left[\mathop{\mathbf{E}}_{\boldsymbol{x},y}\big[L(f(\boldsymbol{x}), y) | f = \mathcal{A}(train)\big]\right]$$

where

- ▷ the outer expectation is taken over all possible training sets
- ▷ the inner expectation measures the risk of the fitted model

Crossvalidation variance estimate





- ▷ The naive variance estimate for the crossvalidation error is biased.
- ▷ The estimate usually gives smaller confidence intervals as they are.
- ▷ This must be accounted in the estimates of optimism and test error.

Theoretical explanation

Theorem. The variance of crossvalidation error $E = \frac{1}{k}(E_1 + \cdots + E_k)$ is a weighted average consisting of three components

$$\theta = \frac{1}{n} \cdot \sigma^2 + \frac{m-1}{n} \cdot \omega + \frac{n-m}{n} \cdot \gamma$$

where

- ▷ σ^2 is the average variance of true test examples all possible training sets.
- ▷ ω^2 is the within-block covariance of test errors sharing the same test set.
- ▷ γ^2 is the between-block covariance of test errors cause by the fact that
  - ◇ training set have large intersection
  - ◇ test fold is inside the training set of another split.

What else can we do with crossvalidation?

Comparing different algorithms

- ▷ We can tune hyperparameters of the algorithm
- ▷ We can estimate which algorithm on average behaves better
- ▷ We can quantify the stability of the performance ranking

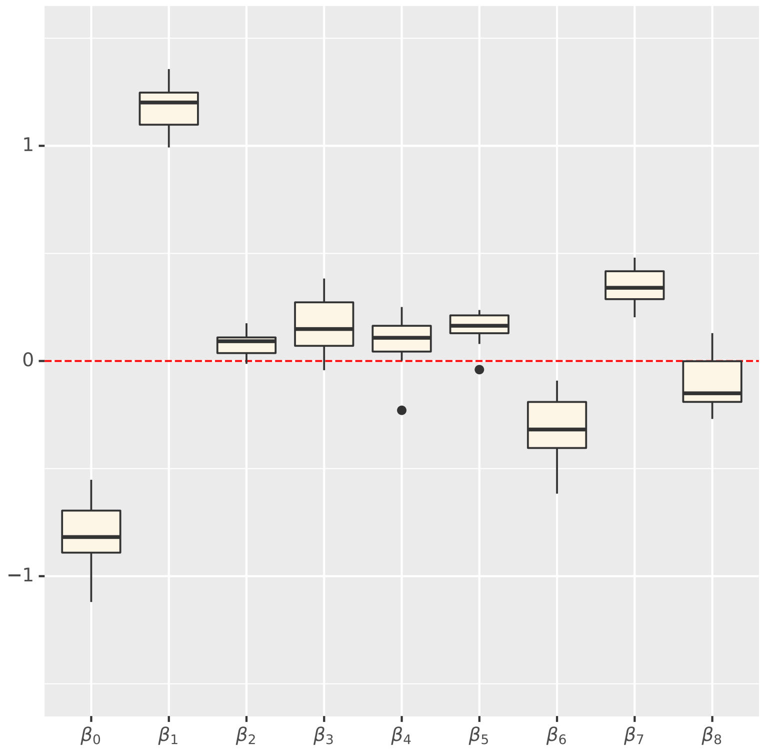
Estimating variance of model parameters

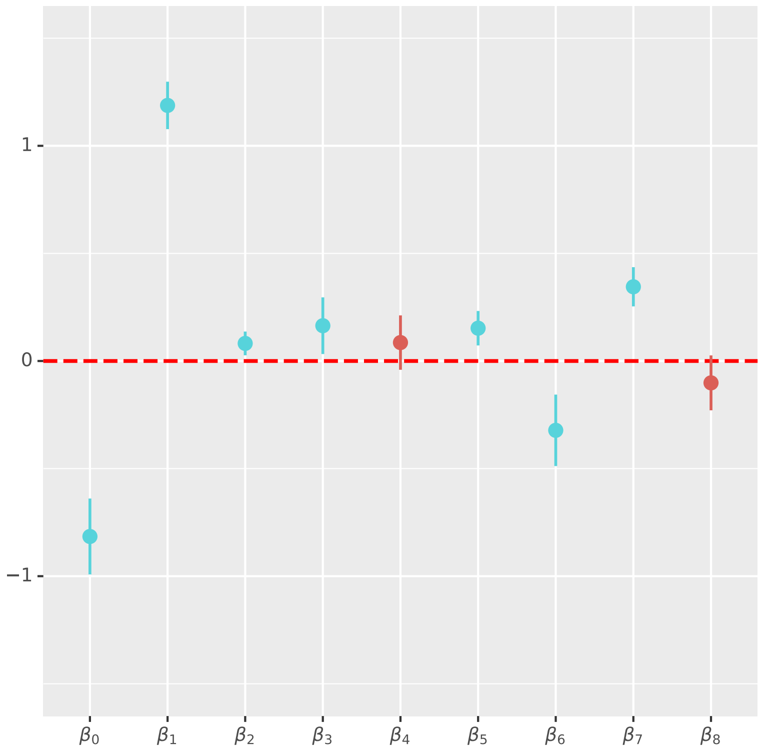
- ▷ Different folds give different parameter instances
- ▷ Parameter confidence intervals can be used for diagnostics
- ▷ Confidence intervals can be used for pruning spurious coefficients

Finding hard instances

- ▷ Different folds give different mismatches $\hat{y}_i \neq y_i$
- ▷ Corresponding problem instances $(\boldsymbol{x}_i, y_i)$ can be studied further

Estimating variance of model parameters





- ▷ The method is applicable for models with compact parametrisation.
- ▷ Each split defines a new model f_i with coefficient $\boldsymbol{\beta}$.
- ▷ The variability of a coefficient β_i shows its certainty and relevance.

Other flavours of cross validation

Exhaustive data splitting

▷ Leave-one-out method, leave-p-out method

Partial splitting

▷ K-fold cross validation for $K = 5, 10$

▷ Monte-Carlo crossvalidation with a fixed split ratio, e.g $1 : 9$.
Same split can occur more than once

▷ Repeated learning testing with a fixed split ratio, e.g $1 : 9$.
Same split can occur only once.

Bootstrapping as an alternative

We could use the entire date set for validation if we could get another dataset for training the model. Bootstrapping is an engineering trick to create a new dataset out of a thin air.

1. Draw N samples from the original dataset with replacement to get a *bootstrap sample* D_B, e.g. the same element can occur more than once.
2. Train the model on the bootstrap sample D_B.
3. Estimate the test error on the original dataset D.
4. Repeate the procedure 20-200 times.
5. Compute necessary statistics and visualise the results if needed.

Standard way how to use bootstrapping

Bootstrapping is mostly used to estimate optimism

- ▷ The model is trained and the training error S_i is computed.
- ▷ The test error E_i is usually computed on the entire dataset.
- ▷ Optimism is computed as $E_i - S_i$.

Note that it does not make sense to compute test error on the entire dataset as we have used some of the data to build a model. Advanced bootstrap methods like **.632 bootstrap** and **.632 bootstrap+** use only the out of training set error and later find a tradeoff between training an test error.

$$E_{\text{boot}} = 0.368 \cdot S_{\text{Train}} + 0.632 \cdot E_{\text{Out-of-training-set}}$$

Other uses of bootstrapping

Estimate the noise-tolerance of the machine learning method

- ▷ Generate a bootstrap sample.
- ▷ Corrupt with an appropriate noise.
- ▷ Train the model and estimate the performance.

Estimate the variance of model coefficients

- ▷ Generate a bootstrap sample.
- ▷ Estimate model parameters.
- ▷ Visualise parameters and compute empirical quantiles.
- ▷ Drop parameter which fluctuate around zero.