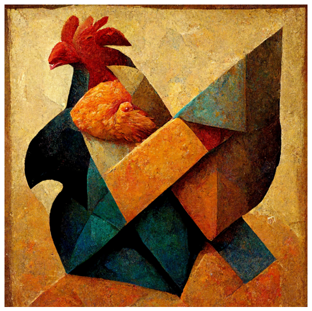
LTAT.02.004 Machine Learning II

Performance evaluation

Sven Laur
University of Tartu

Can we quantify model performance at all?



For some machine learning tasks there is no direct performance measure

- ▷ speech synthesis
- ▷ conversational chat bots
- ▷ image and music generation tasks

For all of these tasks we can define many surrogate measures

- ▷ understandability and expressiveness of speech
- ▷ grammatical correctness and adherence of other implicit patterns

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

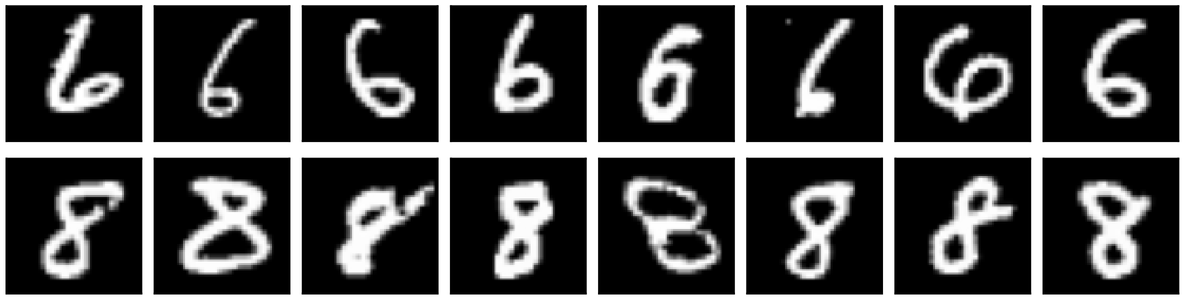
Some goodness measures for regression

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

Handwritten digit recognition task

MNIST database of handwritten digits

- 28×28 grayscale images of numbers
- Training set contains 60,000 images.
- Test set contains 10,000 images.
- We consider separation of two classes.



Scaling laws

The performance of a classification algorithm depends on three factors:

▷ complexity of a model,

▷ the amount of training data,

▷ the amount of computational resources.

There are theoretical scaling laws.

◇ Statistical Learning Theory

◇ Gives conservative relation between dataset size and performance.

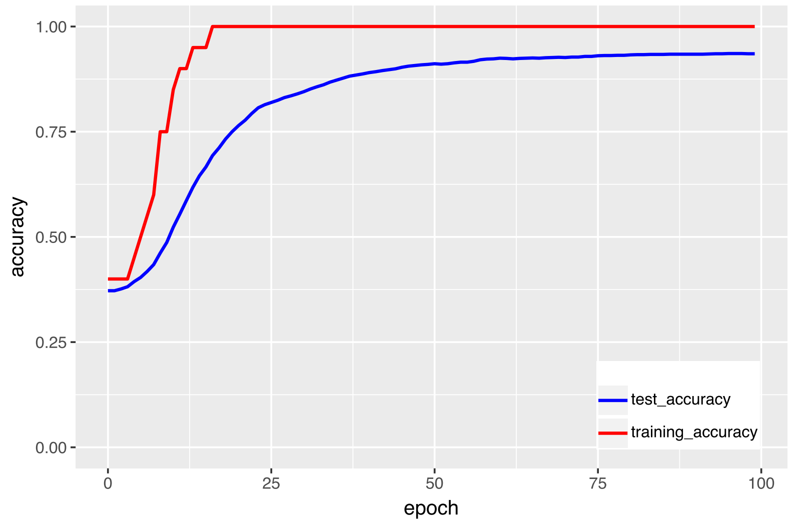
There are empirical scaling laws.

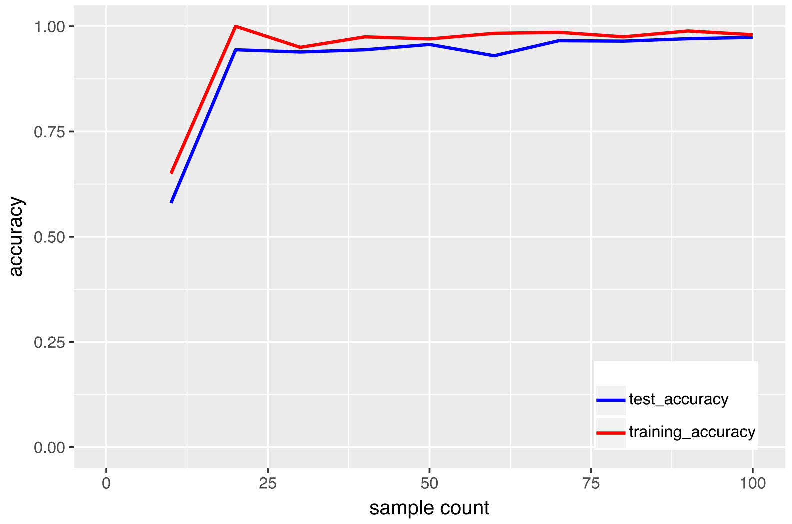
◇ Govern the architectural search of foundational models.

◇ Tells how much data and compute are needed to get target performance.

Examples of scaling laws

We can always fix the model and vary the compute or the dataset size.

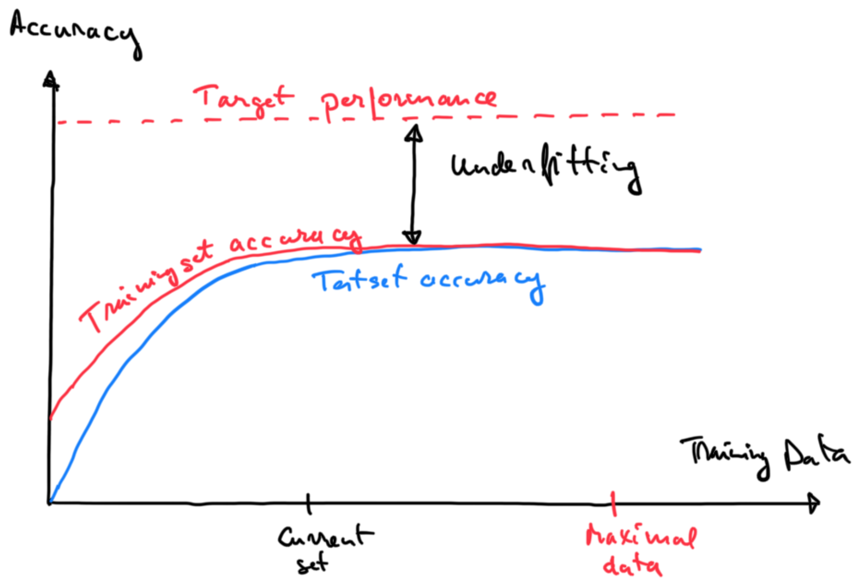




There is always a discrepancy between test and training performance.

- ▷ This leaves some ambiguity about the limiting performance.
- ▷ Random choices are another source of ambiguity that limits utility.

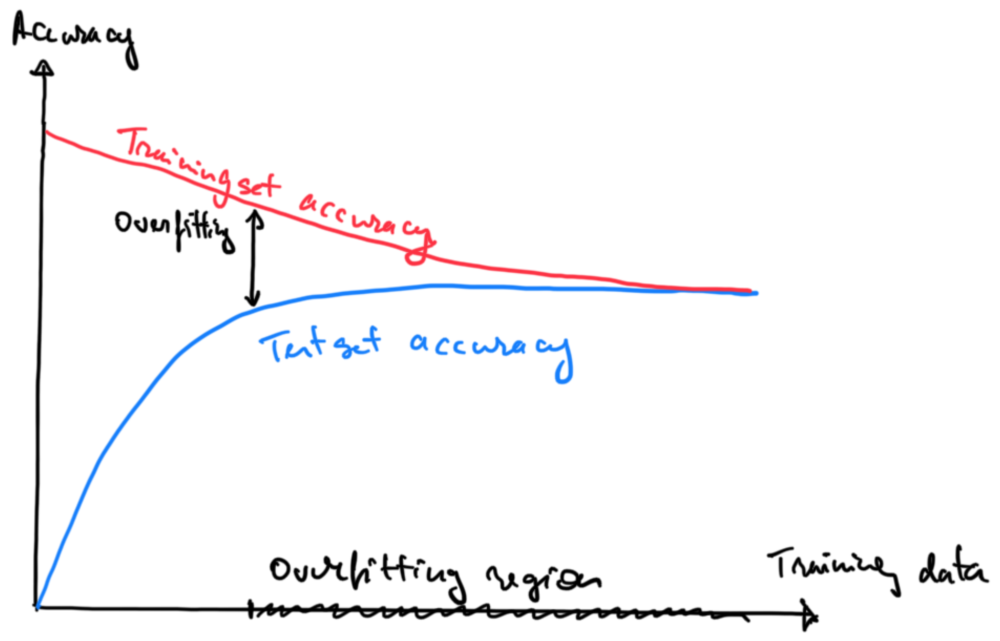
Data-bound learning tasks



A learning task can be hard for following reasons:

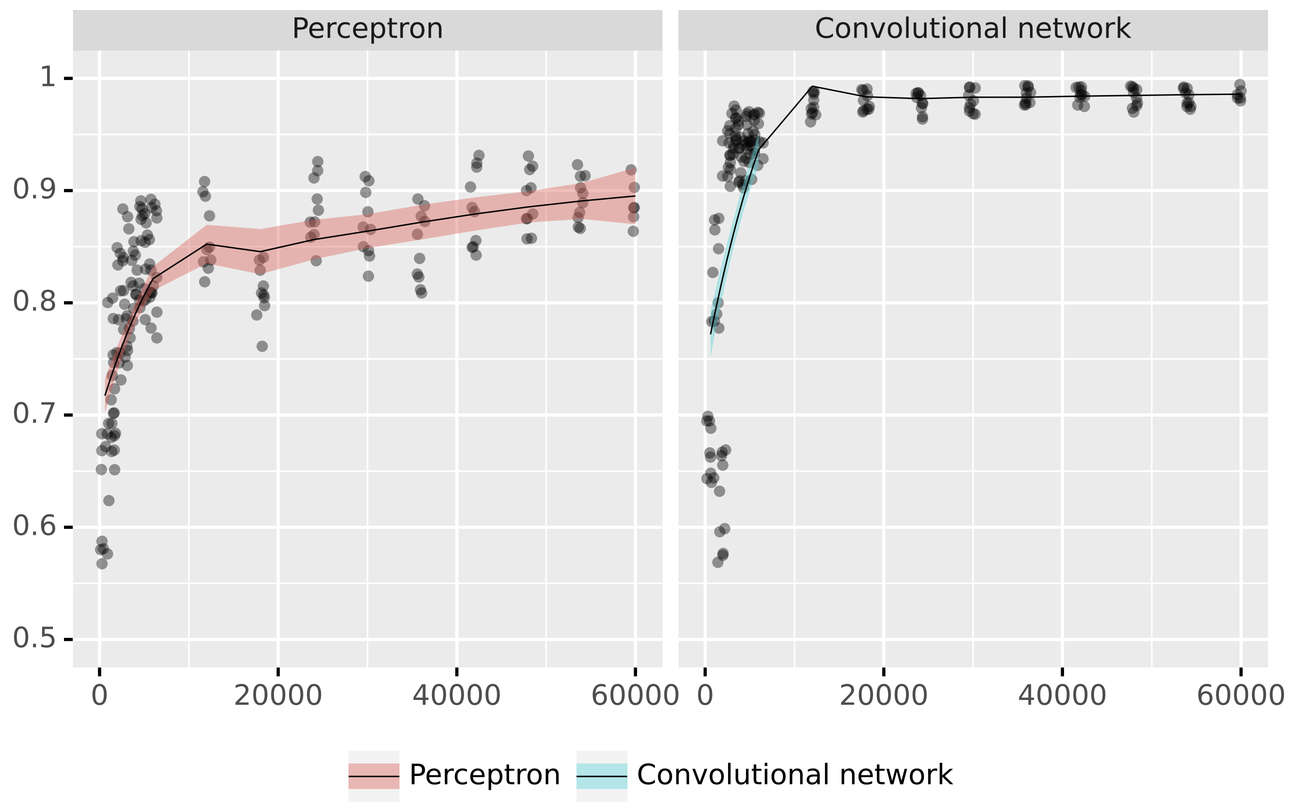
- data bound – there is not enough (labelled) data
- algorithm bound – we do not have the right algorithm
- compute bound – we cannot finalise optimisation

What is overfitting?



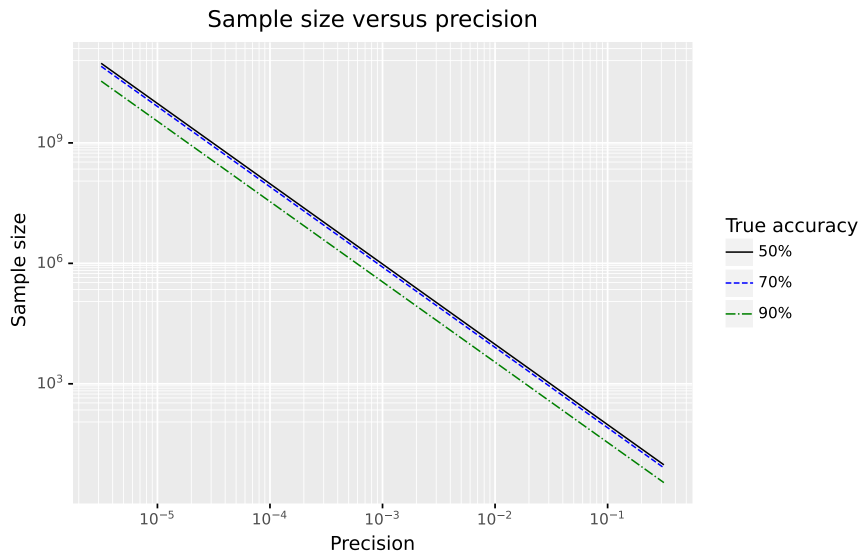
- ▷ Overfitting occurs when only the training performance decreases
- ▷ Overfitting = difference between training and limiting performance
- ▷ Overfitting is inevitable for stabilised models. The extent is important

Models with distinct performance



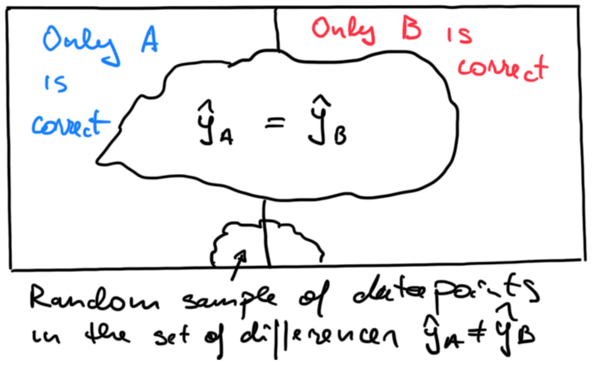
- ▷ The MNIST test set consits of 10000 samples
- ▷ Accuracy for reduced test sets consisting of 1000 samples

How much samples are needed?



- ▷ 9600 samples are needed to estimate accuracy with precision 1%
- ▷ If true accuracy $\geq$ 90% then the number of samples drops to 3500
- ▷ Performance increments of size $0.1 - 0.5\%$ are relevant in practice
- ▷ This means test sets with sizes around $35,000 - 960,000$

Absolute vs relative performance

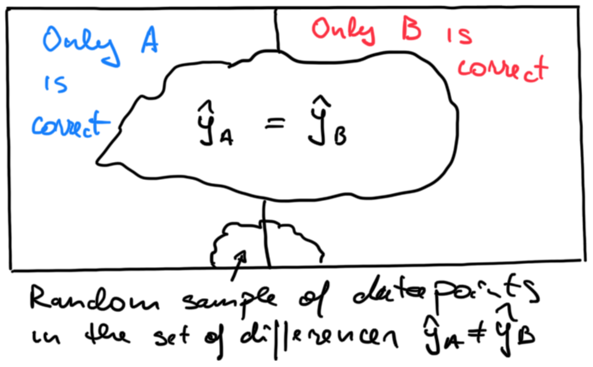

Only A is correct
Only B is correct
$\hat{y}_A = \hat{y}_B$
Random sample of data points in the set of differences $\hat{y}_A \neq \hat{y}_B$

Relative difference in accuracy can be measured more precisely.

- ▷ Make predictions $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$ on unlabelled data.
- ▷ Find data points on which predictions differ $\mathcal{D} = \{i : \hat{\boldsymbol{y}}_A[i] \neq \hat{\boldsymbol{y}}_B[i]\}$.
- ▷ Estimate relative difference in accuracies $\Delta_{\mathcal{D}}$ on the set of differences $\mathcal{D}$.
- ▷ Estimate the relative size $p_{\mathcal{D}}$ of $\mathcal{D}$ and rescale the difference:

$$\text{accuracy}_A - \text{accuracy}_B = p_{\mathcal{D}} \cdot \Delta_{\mathcal{D}} \quad .$$

Shortcuts for relative performance


Only A is correct
Only B is correct
$\hat{y}_A = \hat{y}_B$
Random sample of datapoints in the set of differences $\hat{y}_A \neq \hat{y}_B$

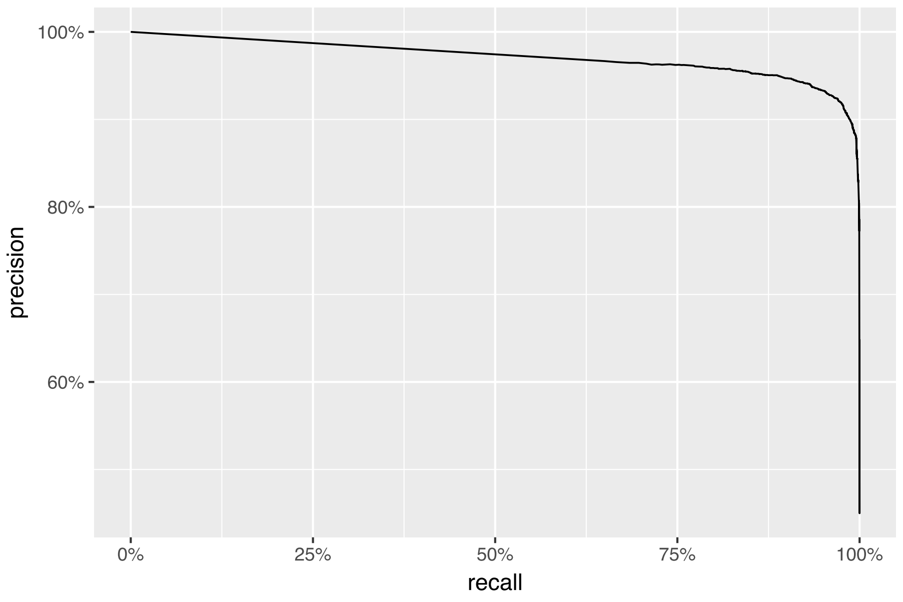
The alternative formula $p_{\mathcal{D}} \cdot \Delta_{\mathcal{D}}$ is more approachable

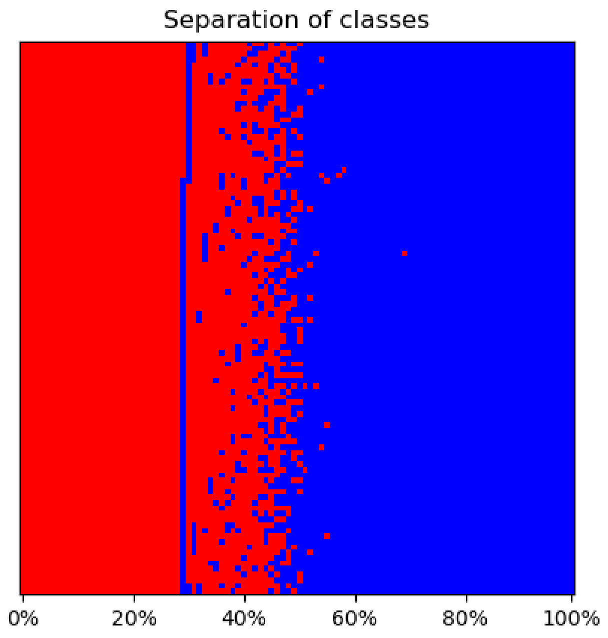
- ▷ The proportion $p_{\mathcal{D}}$ can be computed without knowing true labels
- ▷ $\Delta_{\mathcal{D}}$ can be estimated by labelling 1000 random samples form $\mathcal{D}$

We can bound the relative difference without knowing true labels:

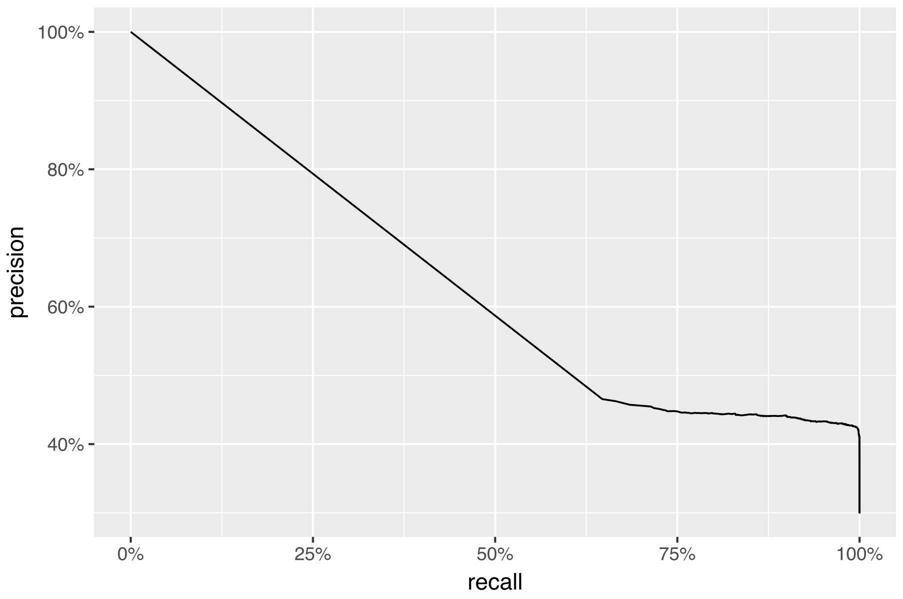
$$\left|\text{accuracy}_A - \text{accuracy}_B\right| \leq p$$

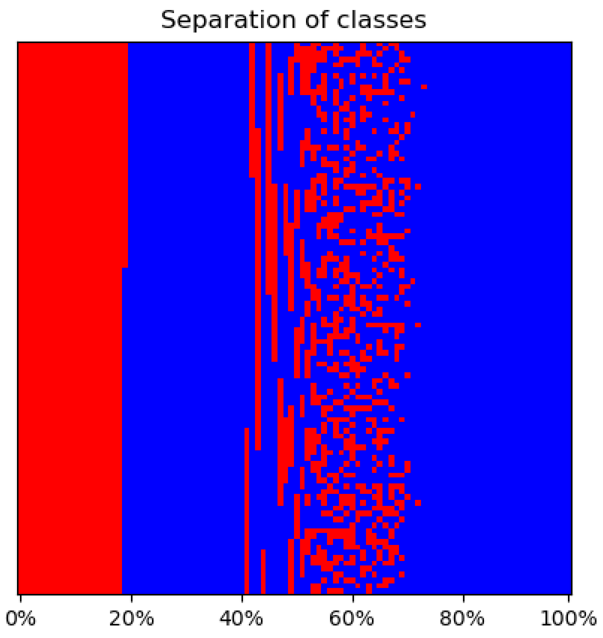
Near ideal precision-recall graph





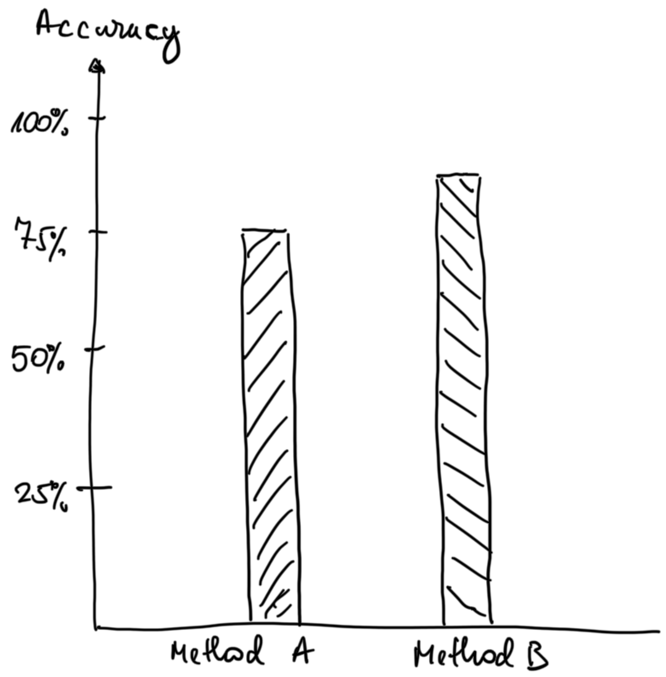
Quick dropp-offs in precision-recall graph





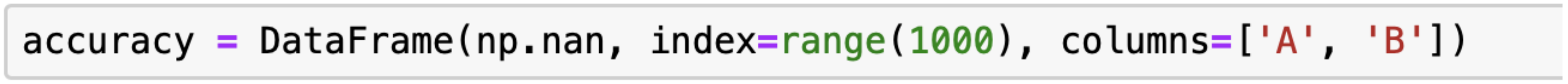
- ▷ There must be a batch of negative data points with high scores.
- ▷ The score stabilises when the ratio of positives and negatives stabilises.

Which model is better?

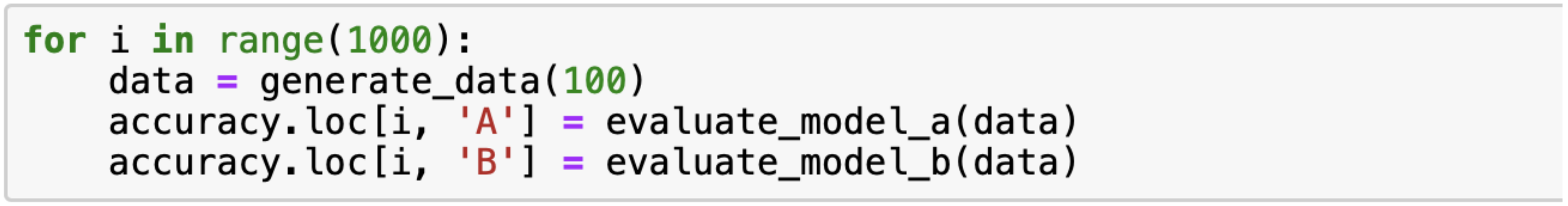


- ▷ Method A outputs the correct label in 75% cases on average
- ▷ Method B outputs the correct lable in 85% cases on average
- ▷ Which method works significantly better on future data?

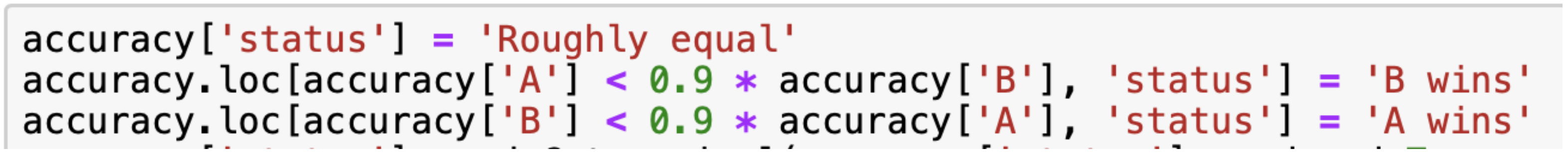
How to model future?



```python
accuracy = DataFrame(np.nan, index=range(1000), columns=['A', 'B'])
```

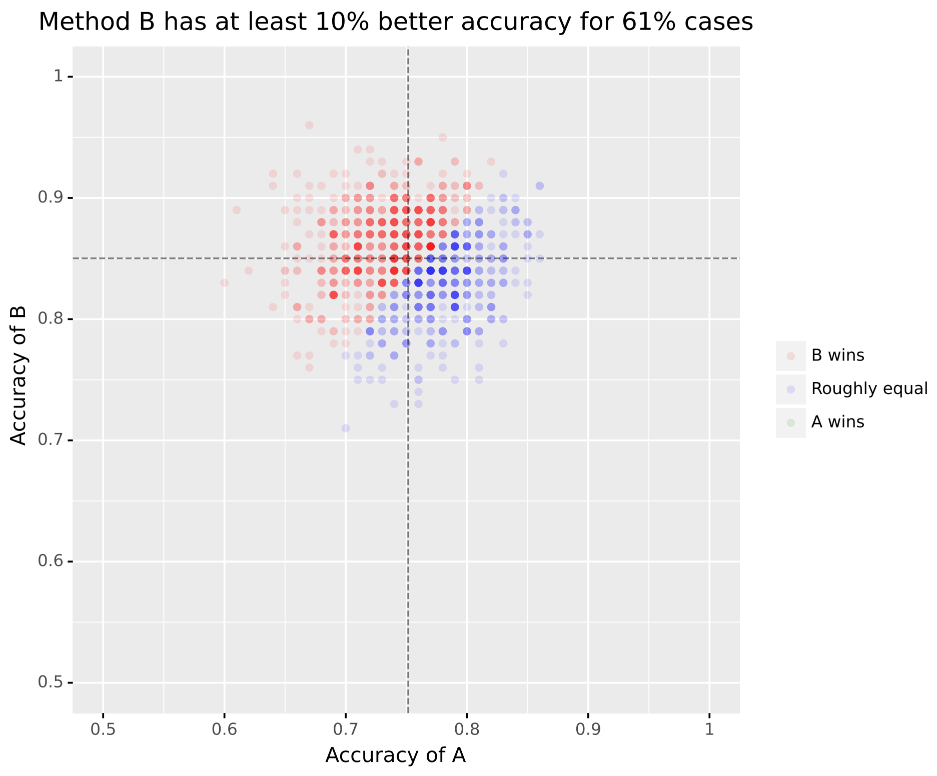


```python
for i in range(1000):
    data = generate_data(100)
    accuracy.loc[i, 'A'] = evaluate_model_a(data)
    accuracy.loc[i, 'B'] = evaluate_model_b(data)
```



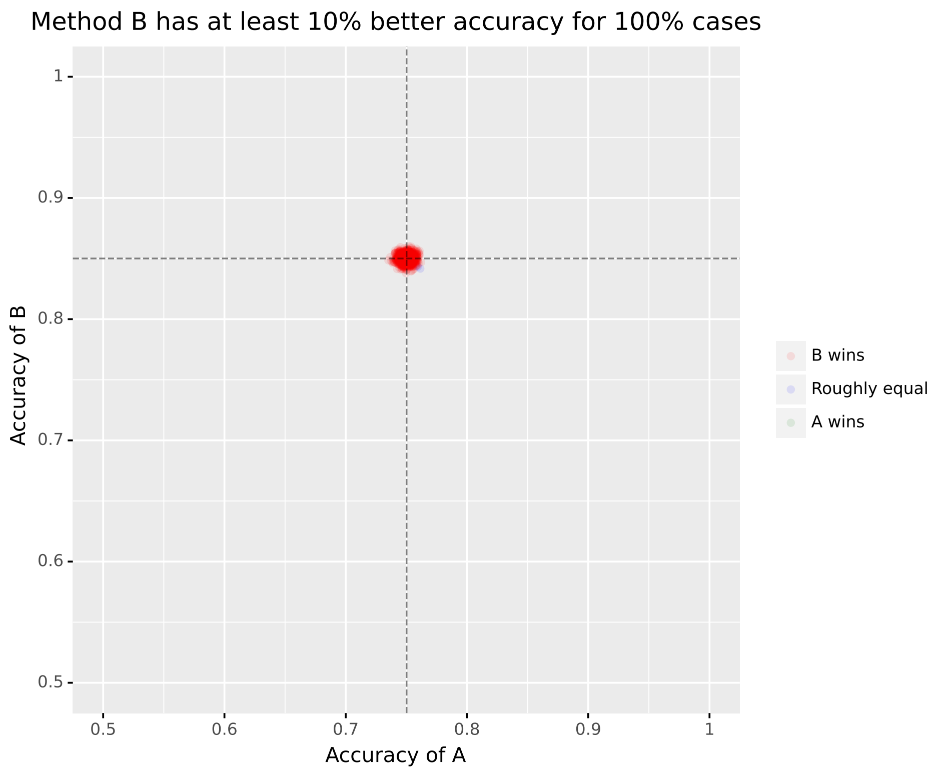
```python
accuracy['status'] = 'Roughly equal'
accuracy.loc[accuracy['A'] < 0.9 * accuracy['B'], 'status'] = 'B wins'
accuracy.loc[accuracy['B'] < 0.9 * accuracy['A'], 'status'] = 'A wins'
```

Future holds only 100 prediction tasks



Both methods are roughly equal although the accuracies are very different.

Future holds 10000 prediction tasks



The method B is clearly superior to the method A as expected.

Direct consequences

- ▷ The performance of a method fluctuates over all possible futures
- ▷ The magnitude of fluctuations decreases with the sample size
- ▷ True performance is defined as a limit over the infinite sample
- ▷ A finite estimate will always fluctuate around the true performance

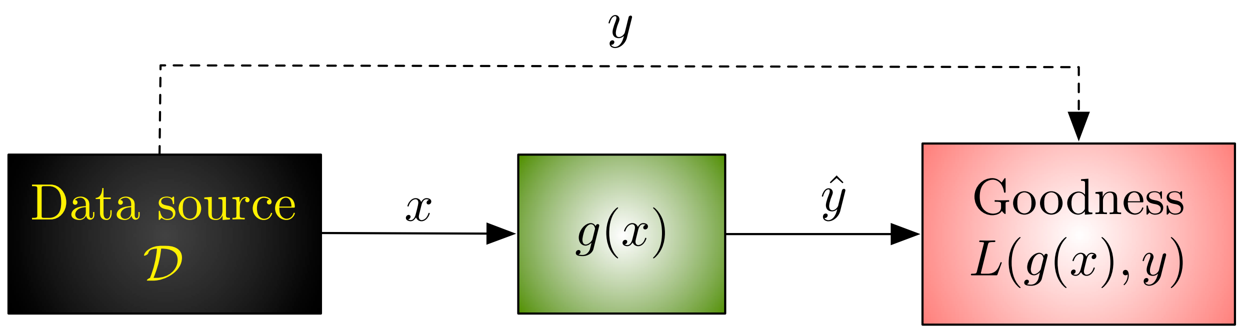
Finite number of new observations in the future means that

- ▷ certain performance differences are irrelevant
- ▷ the pursuit of optimal performance is pointless

Simplifying assumption

- ▷ We assume that the number of new observations is unbounded.

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*

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

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*:

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

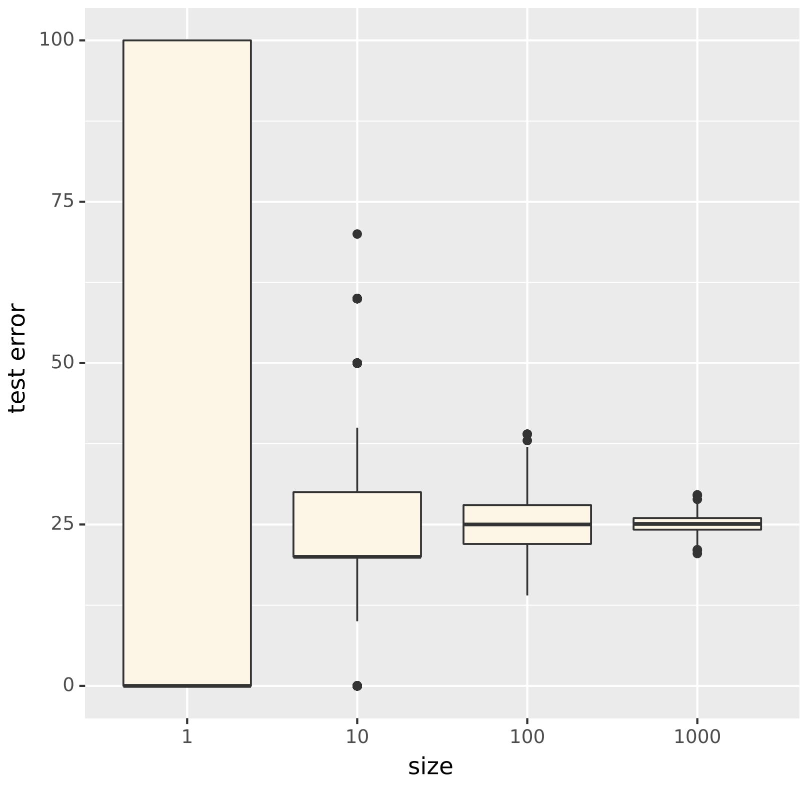
IID sampling assumption. The following conditions assure that the sample data D_N is representative (with high probability).

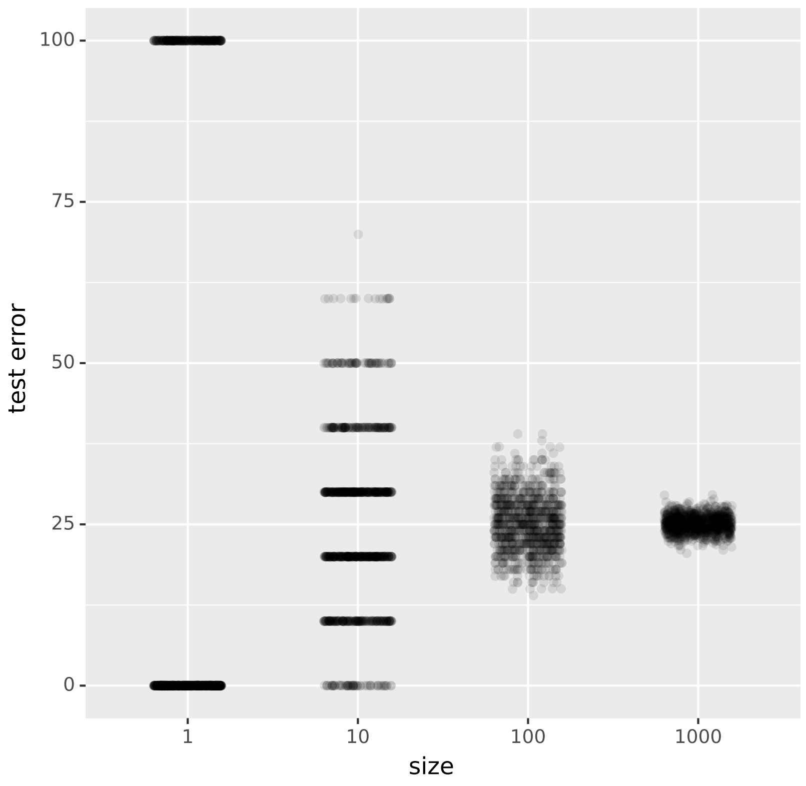
▷ All samples are independent from each other.

▷ 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 $\mathcal{N}(mean = 0, sd = \sigma)$.

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:

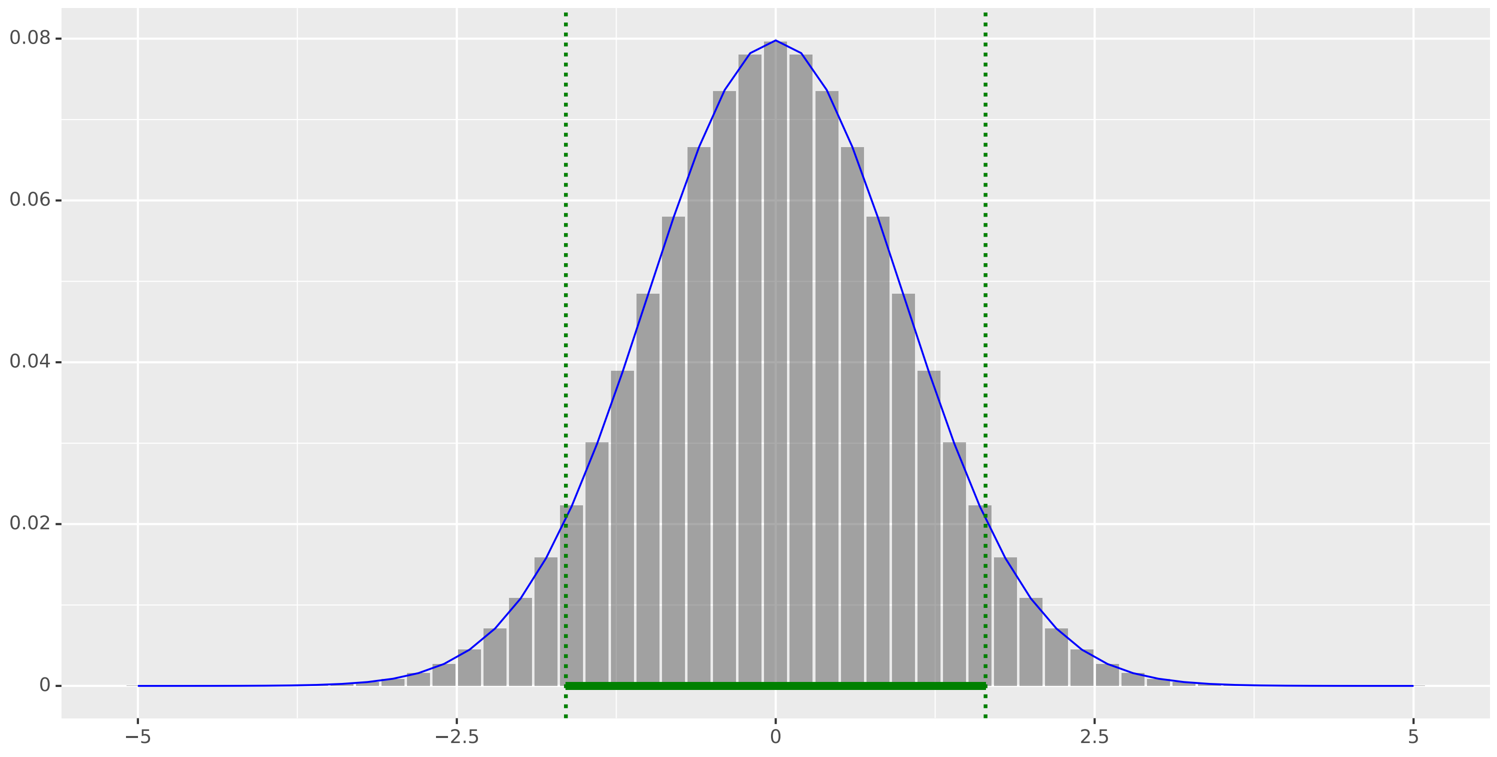
$$\Pr\left[|R_N(f) - R(f)| \geq \varepsilon\right] \lesssim 2 \cdot \int\limits_{-\infty}^{\varepsilon} \frac{\sqrt{N}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{Nt^2}{2\sigma^2}\right) dt$$

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.

Visual representation



Convergence implies that the centre area of is well approximated

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