Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Evaluation of Models

Niels Landwehr

Learning and Prediction

- Classification, Regression: Learning problem
  - Input: training data $L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
  - Output: model $f : X \rightarrow Y$

- Model will be used to obtain predictions for novel test instances

$$f(\mathbf{x}) = ? \in \mathcal{Y} \qquad \mathbf{x} \in \mathcal{X} \quad \text{test instance}$$

- Have seen several different types of models
  - Linear models, decision trees…

Evaluation of Models

- Question: After having implemented learning algorithm, having trained model etc.: how accurate are our predictions?
  - What exactly do we mean by accurate?
  - How do we calculate / measure / estimate accuracy?

- We care about accuracy of predictions when applying the model to unseen, novel test data (not about accuracy on observable training data).

- Evaluation of models: Estimate accuracy of predictions of learned models.

Evaluation of models: Assumptions

- In order to study the evaluation of models formally, we have to make assumptions about the properties of training and test data.

- Central assumption: all data are drawn from fixed (unknown) distribution $p(\mathbf{x},y)$

Distribution over instances

Distribution over labels given instance

$$p(\mathbf{x}, y) = p(\mathbf{x})\, p(y \mid \mathbf{x})$$

- Example spam-filtering
  - $p(\mathbf{x})$ Probability to see email $\mathbf{x}$
  - $p(y|\mathbf{x})$ Probability to see label $y \in \{\text{Spam/Ok}\}$ for email $\mathbf{x}$.

Evaluation of models: Assumptions

- „i.i.d.“-assumption: Examples are „independent and identically distributed“.
  - Training instances are drawn independently from distribution $p(\mathbf{x},y)$:

    $$(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$$ training examples

  - Test instances are also drawn independently from this distribution

    $$(\mathbf{x}, y) \sim p(\mathbf{x}, y)$$ test instance (seen when applying the model)

  - Is this always realistic?

- In the following, we will always assume i.i.d. data.

Loss functions

- We have made assumption about the instances $(\mathbf{x},y)$ that we will see
- New test instance $(\mathbf{x},y)$ arrives, model predicts $f(\mathbf{x})$.
- Loss function defines how good/bad the prediction is:

$$\ell(y, f(\mathbf{x}))$$ Loss of prediction $f(\mathbf{x})$ for instance $(\mathbf{x},y)$

  - Non-negative: $\forall y, y' : \ell(y, y') \geq 0$
  - Problem specific, given a priori.
- Loss functions for classification
  - Zero-one loss: $\ell(y, y') = 0, \text{ if } \; y = y'; \; 1, \text{ otherwise}$
  - Class-dependent cost matrix
- Loss functions for regression
  - Squared loss: $\ell(y, y') = (y - y')^2$

Evaluating models: Risk of model

- Central definition when evaluating models: risk.
- Risk of a model: expected loss for novel test instance

$(\mathbf{x}, y) \sim p(\mathbf{x}, y)$ test instance $\mathbf{x}$ with label y (random variable)

$\ell(y, f(\mathbf{x}))$ loss for test instance (random variable)

$$R(f) = E[\ell(y, f(\mathbf{x}))] = \int \ell(y, f(\mathbf{x}))\, p(\mathbf{x}, y)\, d\mathbf{x}\, dy$$

  - For zero-one-loss risk is also called error rate.
  - For squared loss risk is also called mean squared error.

- Main goal when evaluating models: determine risk of model.
- Risk cannot be determined exactly, because $p(\mathbf{x},y)$ is unknown → estimation problem.

Evaluation of models: risk estimate

- Estimating risk from data.
- If data sampled from $p(\mathbf{x},y)$ is available,

$$T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \qquad (\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$$

we can estimate the risk:

$$\hat{R}(f) = \frac{1}{m} \sum_{j=1}^{m} \ell(y_j, f(\mathbf{x}_j)) \qquad \text{"empirical risk"}$$

- **Important**: Which data T to use?
  - Training data (T=L)?
  - Split available data into L und T.
  - Cross-validation.

Estimator as a random variable

- Estimator

$$\hat{R}(f) = \frac{1}{m}\sum_{j=1}^{m} \ell(y_j, f(\mathbf{x}_j))$$

- Estimator is random variable:
  - Instances in *T* have been drawn randomly

$$(\mathbf{x}_j, y_j) \sim p(\mathbf{x}, y) \qquad \text{Which } (\mathbf{x}_j, y_j) \text{ where drawn?}$$

  - Value of estimator depends on randomly sampled instances, thus it is the result of a random process.

- Estimator has an expected value $E[\hat{R}(f)]$.
- Estimator is unbiased if and only if:
  - Expectation of empirical risk = true risk.

Bias of estimator

- Estimator $\hat{R}(f)$ is unbiased if and only if:

  $E[\hat{R}(f)] = R(f)$

- Otherwise, $\hat{R}(f)$ has a bias:

  $Bias = E[\hat{R}(f)] - R(f).$

- Estimator is optimistic, if $E[\hat{R}(f)] < R(f)$.

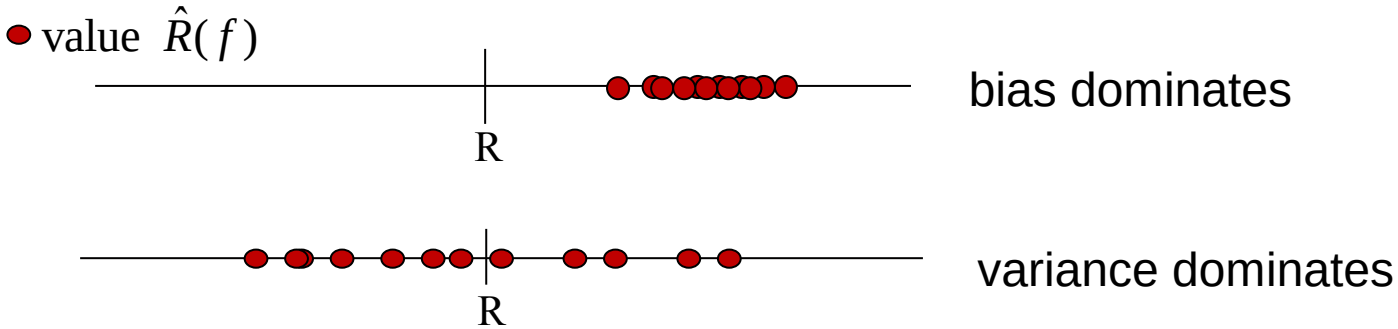
- Estimator is pessimistic, if $E[\hat{R}(f)] > R(f)$.

Variance of estimator

- Estimator $\hat{R}(f)$ has a variance

$$Var[\hat{R}(f)] = E[\hat{R}(f)^2] - E[\hat{R}(f)]^2$$

- The larger the sample T used for computing the estimate is, the lower the resulting variance.
- Variance vs. bias:
  - High variance: large random component in empirical risk estimate.
  - Large bias: systematic error in empirical risk estimate.



Risk estimate on training data

- Which set T should we use?
- 1. Try: training data L

- Model f, trained on $L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
- Empirical risk measured on training data

$$\hat{R}_L(f) = \frac{1}{m} \sum_{j=1}^{m} \ell(y_j, f(\mathbf{x}_j))$$

Risk estimated on *L*

- Is this risk estimate an
  - unbiased
  - optimistic
  - pessimistic

  estimator of the true risk $R(f)$?

Risk estimate on training data

- Empirical risk on training data is an optimistic estimator of the true risk.

- Empirical risk of *all possible* models for a fixed L?
  - Due to random effects it holds for some models f, that $\hat{R}_L(f) < R(f)$ and for other models f, that $\hat{R}_L(f) > R(f)$.

  - Learning algorithm chooses a model f_L with small empirical risk $\hat{R}_L(f_L)$.

  - Likely that $\hat{R}_L(f_L) < R(f_L)$ (optimistic risk estimate).

Risk estimate on training data

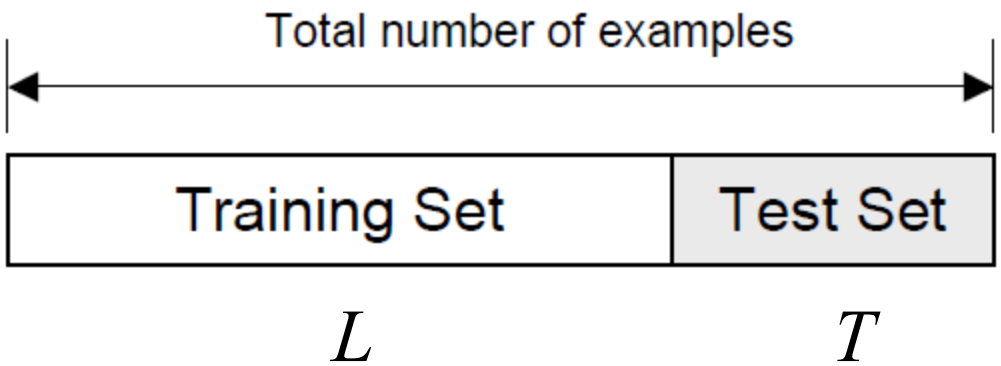
- Empirical risk of the model chosen by the learning algorithm on the training data („training error“) is optimistic estimator of true risk:

$$E[\hat{R}_L(f_L)] < R(f_L).$$

- The problem is caused by the dependency of the chosen model on the data used for evaluation.
- Approch to fix the problem: use test data that are independent of the training data.

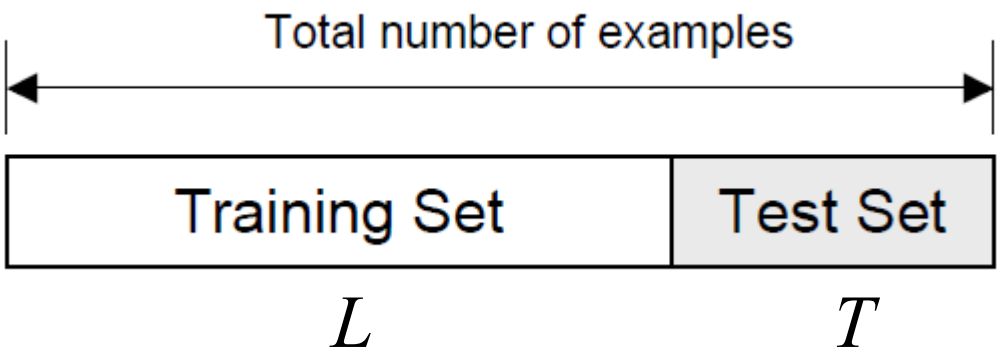
Holdout-Testing

- Idea: estimate risk on independent test data
- Given: data $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_d, y_d)\}$
- Split data into
  - Training data $L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ and
  - Test data $T = \{(\mathbf{x}_{m+1}, y_{m+1}), \ldots, (\mathbf{x}_d, y_d)\}$



Holdout-Testing

- Run learning algorithm on data L, this yields model f_L .
- Compute empirical risk $\hat{R}_T(f_L)$ on test data T.
- Run learning algorithm on data D, this yields model f_D .
- Output: Model f_D , use $\hat{R}_T(f_L)$ as estimator for true risk of the model f_D.



Holdout-Testing: Analysis

- Is the estimator $\hat{R}_T(f_L)$ for the risk of the model f_D
  - unbiased,
  - optimistic,
  - pessimistic?

Holdout-Testing: Analysis

- Estimator $\hat{R}_T(f_L)$ is pessimistic for $R(f_D)$:
  - $\hat{R}_T(f_L)$ is unbiased for f_L
  - f_L was learned on fewer training examples than f_D, and therefore has a higher risk (in expectation).
- But the estimator $\hat{R}_T(f_L)$ is useful in practice, while the estimator $\hat{R}_L(f_L)$ is usually wildly optimistic (often close to 0).
- Why do we train and return model f_D ?
- Final model f_D rather than f_L , because f_D has a lower risk, and is therefore better.

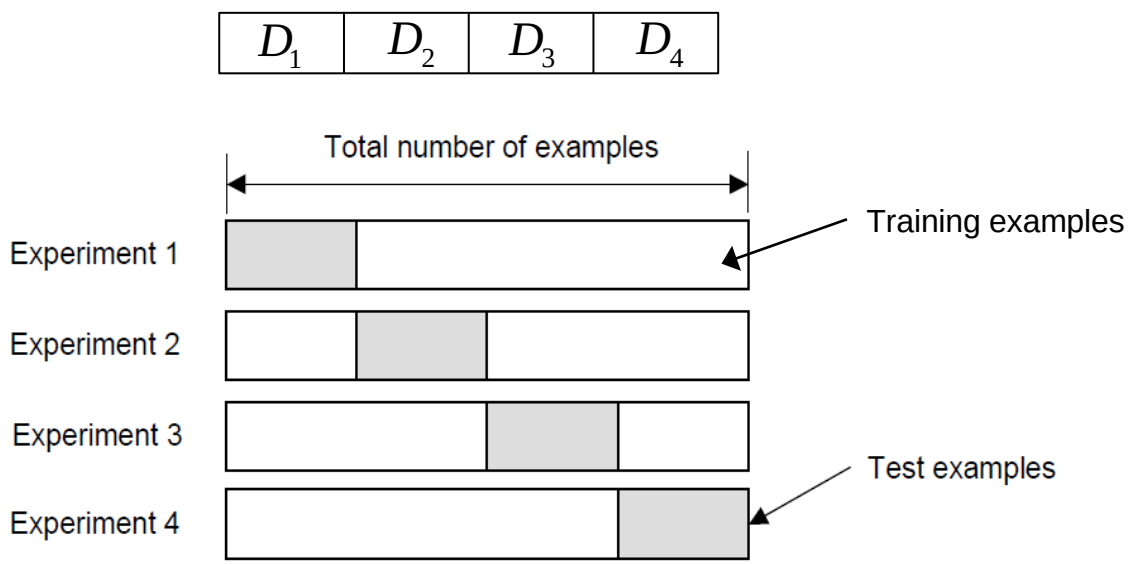
Holdout-Testing: Analysis

- What are the advantages/disadvantages when choosing the test set T
  - large
  - small

  ?
- T should be large to ensure that risk estimate $\hat{R}_T(f_L)$ has low variance.
- T should be small to ensure that $\hat{R}_T(f_L)$ has low bias, that is, is not too pessimistic.
- We need a lot of data in order to obtain good estimates
  - In practice, holdout-testing is only used when data is plentyful.
  - Cross-validation (next slide) usually gives better results.

Cross-Validation

- Given: data $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_d, y_d)\}$
- Split D into n equally sized blocks $D_1, \ldots, D_n$ with $D = \bigcup_{i=1}^{n} D_i$ and $D_i \cap D_j = 0$
- Repeat for $i = 1 \ldots n$
  - Learn model f_i with $L_i = D \setminus D_i$.
  - Compute empirical risk $\hat{R}_{D_i}(f_i)$ on D_i.

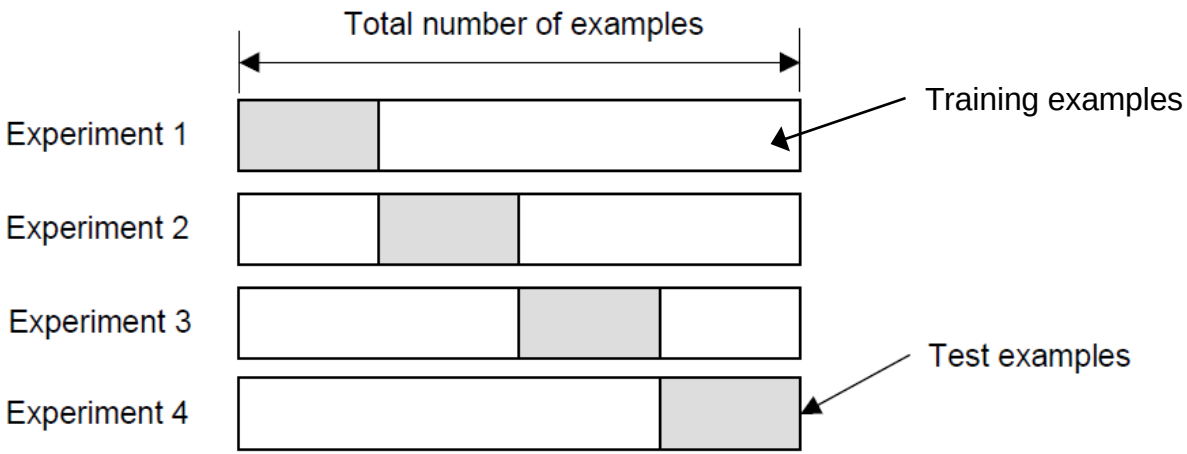


Cross-Validation

- Average empirical risk estimates from the different test sets D_i:

$$\overline{R} = \frac{1}{n}\sum_{i=1}^{n} \hat{R}_{D_i}(f_i)$$

- Learn model f_D on complete data set D.
- Return model f_D and estimator $\overline{R}$.



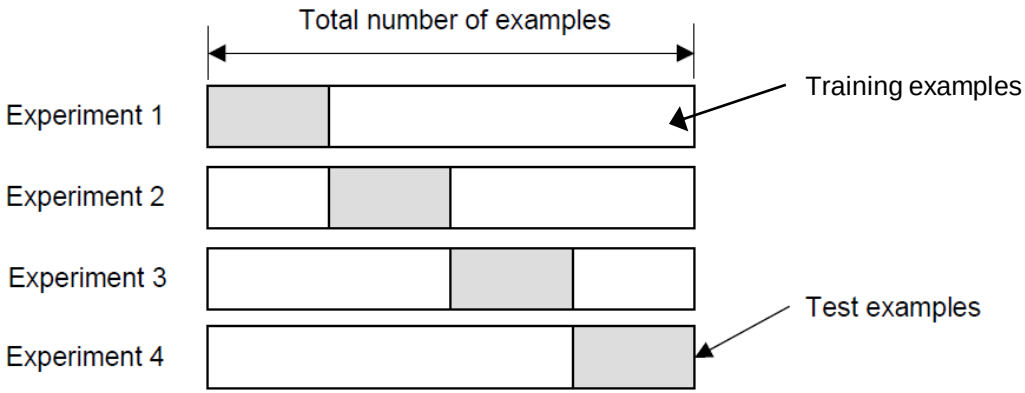
Cross-Validation: Analysis

- Is the estimator
  - optimistic / pessimistic / unbiased ?

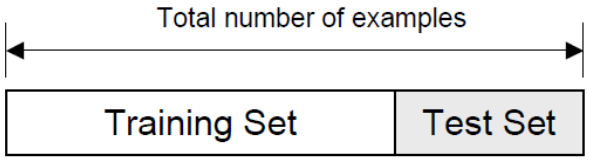
Cross-Validation: Analysis

- Is the estimator
  - optimistic / pessimistic / unbiased ?
- Estimator is pessimistic:
  - Models f_i are trained on fraction $(n-1)/n$ of overall data.
  - Model f_D will be trained on all data.

Cross-Validation



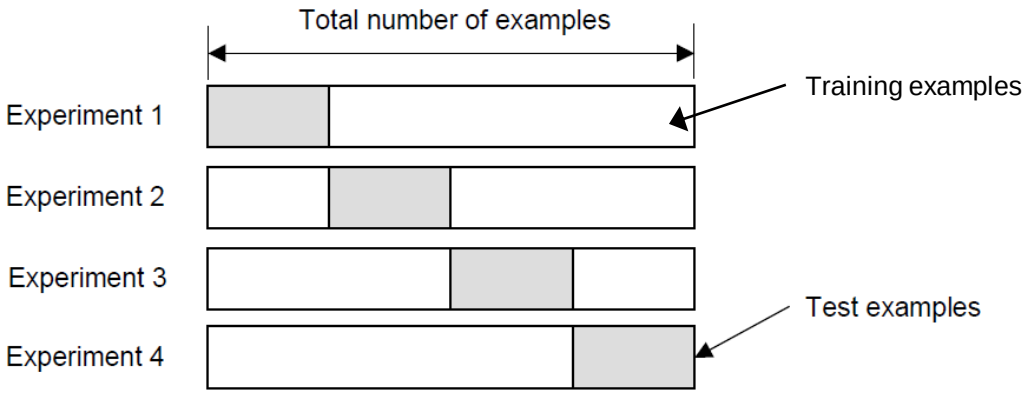
Holdout



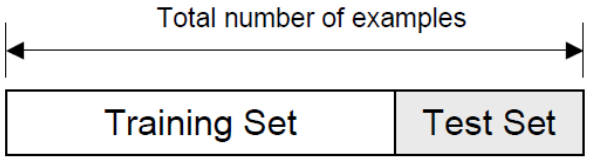
Cross-Validation: Analysis

- Bias/Variance compared to holdout-testing?
- Variance is lower than for holdout-testing
  - Averaging over several holdout experiments, this reduces variance
  - Estimator is based on all data, because all instances appear as test instances in some block.
- Bias similar as for holdout-testing, depends on number of blocks.

Cross-Validation



Holdout



Example: regularized polynomial regression

- Polynomial model $f_{\mathbf{w}}(x) = \sum_{i=0}^{M} w_i x^i$

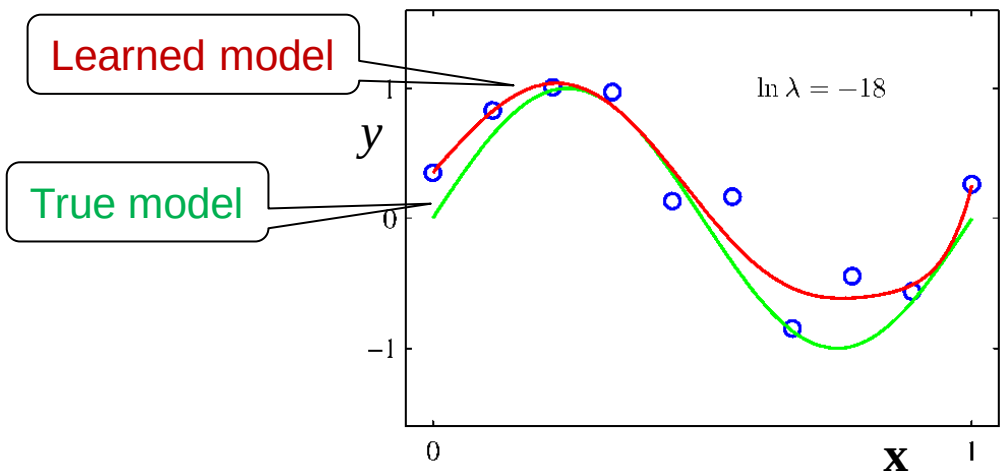
- Learn model by minimizing regularized loss

$$\mathbf{w}_* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (f_{\mathbf{w}}(x_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

Training data

$$L = \{(x_1, y_1), \ldots, (x_m, y_m)\}$$

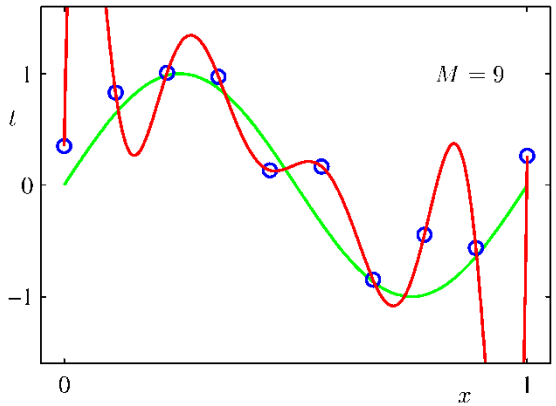
$$\ln \lambda = -18$$



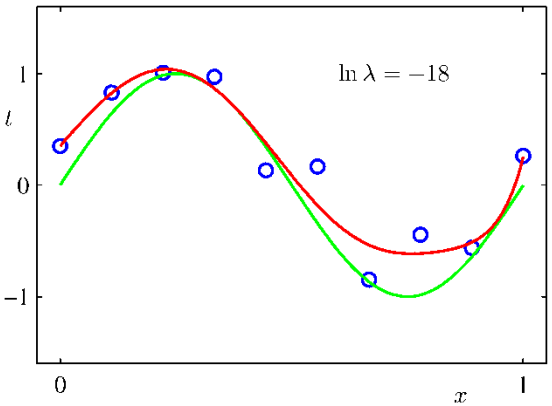
Tune regularization parameter

- We have to determine a good regularization parameter λ.
- Regularization parameter controls complexity of model.

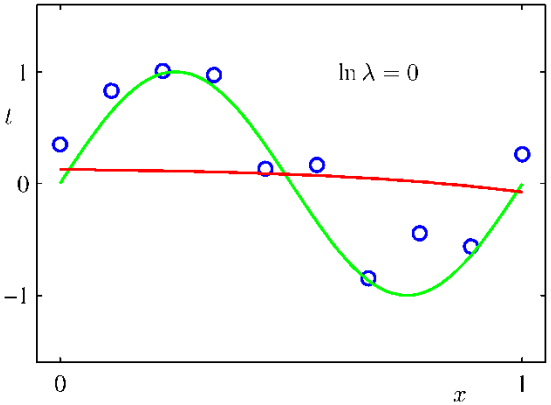
$\ln \lambda = -\infty$



$\ln \lambda = -18$

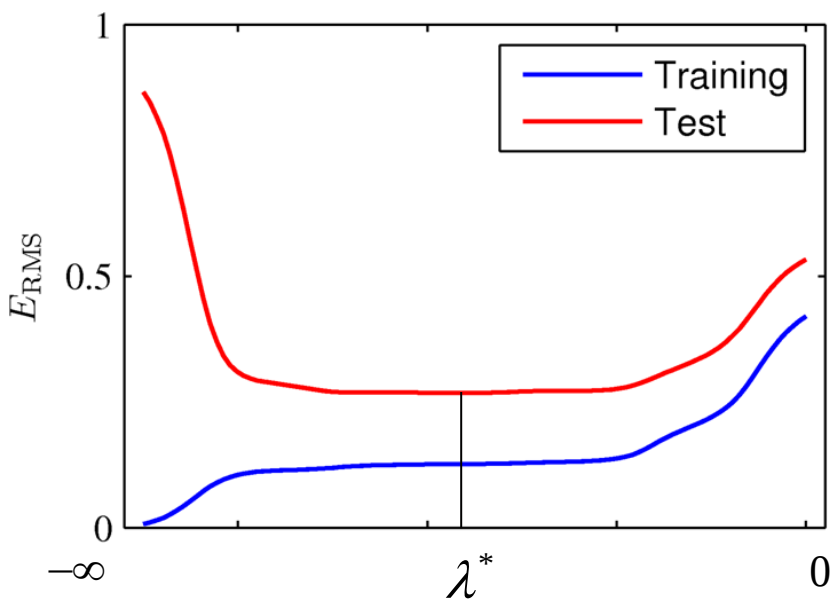


$\ln \lambda = 0$



Tune regularization parameter

- Perform cross-validation for different parameters λ, save the corresponding risk estimates.
- When training the final model on all of the data, use the parameter λ^* that resulted in smallest risk estimate.



Training error minimal for unregularized model, but test error better for moderate regularization.

Tune regularization parameter

- Algorithm: Learn model with optimal regularization parameter λ.

- Function $\text{trainModelOptimalLambda}(D)$
  - For $\lambda \in \{2^{-k}, 2^{-k+1} \ldots, 2^{k-1}, 2^{k}\}$:
    - Determine $error(\lambda) = \text{crossValidation}(\lambda, D)$

cross-validation risk estimate for model with parameter λ on data *D*.

  - Set $\lambda^* = \arg\min_{\lambda} error(\lambda)$
  - Learn $f^* = \text{trainModel}(\lambda^*, D)$

Learning model with parameter λ^* on data *D*.

- Output: model f^*.

Estimating error of model with tuned regularization parameter

- How do we estimate the error of the model with tuned regularization parameter λ^* ?

- Warning: we can not simply use the error estimate $error(\lambda^*)$!
  - The parameter λ^* was chosen such that $error(\lambda^*)$ is as small as possible.
  - The error estimate $error(\lambda^*)$ is therefore optimistic.
  - Compare with earlier argument: training error is optimistic, because model parameters have been chosen based on training data.

- Instead, what we need is a „nested“ cross-validation (see next slide).

Nested Cross-Validation

- Algorithm: risk estimate with tuned regularization parameter λ

- Function $\text{trainAndEvaluateOptimalLambda}(D)$
  - Split D into n equally sized blocks $D_1, \ldots, D_n$ with $D = \bigcup_{i=1}^{n} D_i$ and $D_i \cap D_j = 0.$

  - For $i \in \{1, \ldots, n\}$:
    - Learn $f_i^* = \text{trainModelOptimalLambda}(D \setminus D_i)$

    - Determine empirical risk $\hat{R}_{D_i}(f_i^*)$ on data D_i

  - Average the different empirical risk estimates: $\bar{R} = \frac{1}{n} \sum_{i=1}^{n} \hat{R}_{D_i}(f_i).$
  - Learn $f^* = \text{trainModelOptimalLambda}(D)$

- Output: model f^* and risk estimate $\bar{R}.$

Evaluation: Summary

- Studied the problem of risk estimation: expected loss on novel test data.

- Training error optimistic, cannot be used as risk estimate.
- Appropriate approaches are holdout-testing and cross-validation.

- Cross-validation is also used to tune hyperparameters such as regularization parameter λ.
- Error estimate for model with tuned hyperparameters requires „nested“ cross-validation.