Introduction to Machine Learning

Generalization and Model Validation

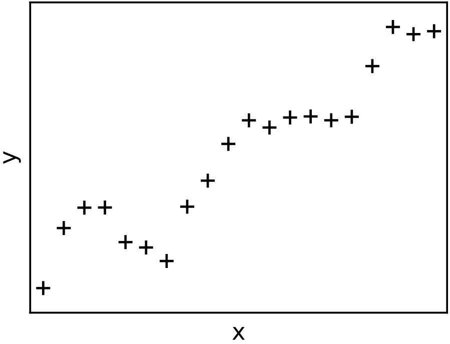
Nils M. Kriege

WS 2023

Data Mining and Machine Learning
Faculty of Computer Science
University of Vienna

Credit: Slides based on the IML Lectures by Sebastian Tschiatschek and Andreas Krause

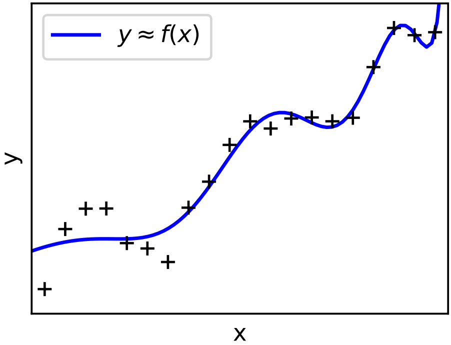
Regression



- *Goal*: learn real valued mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}$

[regression/regression-goal.py]

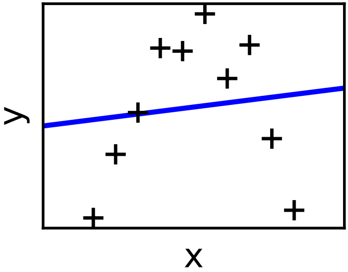
Regression



- *Goal*: learn real valued mapping $f : \mathbb{R}^d \to \mathbb{R}$

[regression/regression-goal.py]

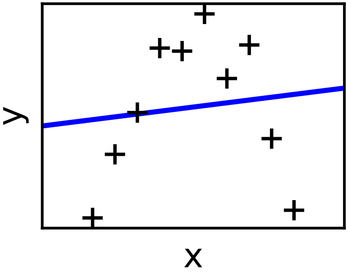
Fitting nonlinear functions



Goal: $y \approx f(x)$

Observation: Linear function not suitable

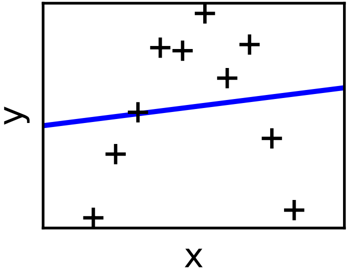
Fitting nonlinear functions



Goal: $y \approx f(x)$
Observation: Linear function not suitable

💡 Use quadratic function $f(x) = ax^2 + bx + c$

Fitting nonlinear functions



Goal: $y \approx f(x)$
Observation: Linear function not suitable

- Use quadratic function $f(x) = ax^2 + bx + c$
- Transform input data and use linear regression:

$$f(x) = ax^2 + bx + c = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} \quad \text{with} \quad \tilde{\mathbf{w}} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \quad \tilde{\mathbf{x}} = \begin{pmatrix} x^2 \\ x \\ 1 \end{pmatrix}$$

Linear regression for polynomials

We can fit non-linear functions via linear regression, using non-linear features of our data (basis functions):

$$\phi \colon \mathbb{R}^d \to \mathbb{R}^D \qquad f(\mathbf{x}) = \sum_{i=1}^{D} w_i \phi_i(\mathbf{x})$$

Example: Polynomials of degree k.

1-dim: $\phi(x) = (1, x, x^2, \ldots, x^k)^T$

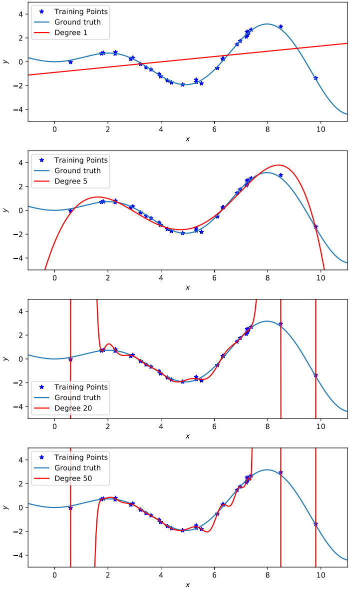
2-dim: $\phi(\mathbf{x}) = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^2x_2, \ldots)^T$

d-dim: $\phi(\mathbf{x}) =$ vector of all monomials $x_1, x_2, \ldots, x_d$ of degree k

Demo: Linear regression on polynomials

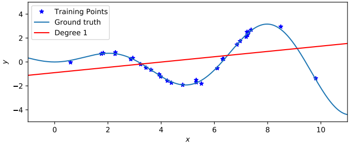
[regression/nonlinear-regression.ipynb]

Observation: Underfitting and overfitting

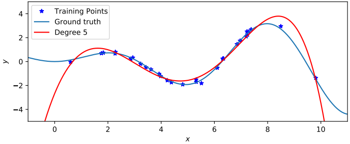


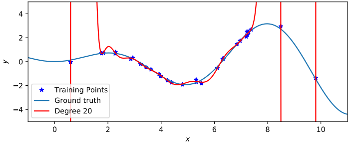
[regression/overfitting.py]

Observation: Underfitting and overfitting

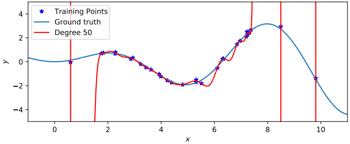


Underfitting





Overfitting



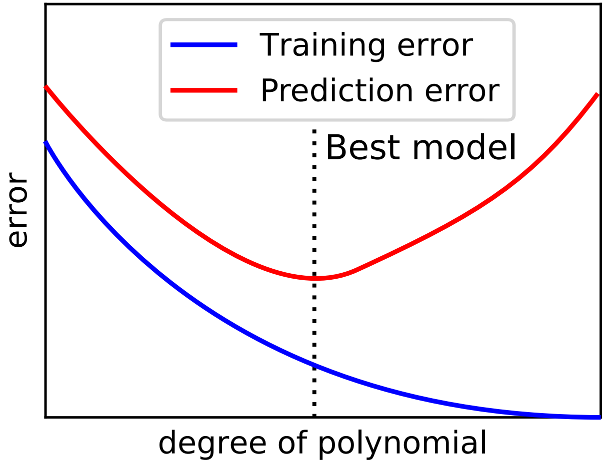
Overfitting

[regression/overfitting.py]

Supervised learning summary so far

Representation/ features	Linear hypotheses, nonlinear hypotheses through feature transformations
Model/ objective	Loss-function (squared loss, ℓ_p loss)
Method	Exact solution, Gradient Descent
Evaluation metric	Empirical risk = (mean) squared error

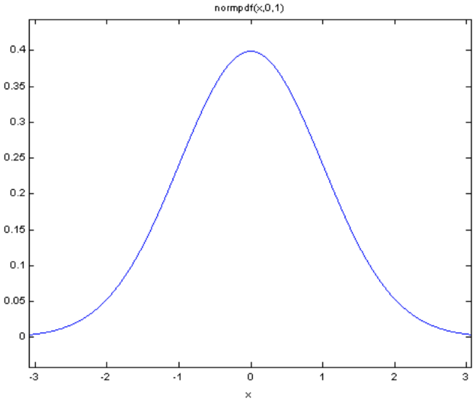
Model selection for linear regression with polynomials



Interlude: A note on probability

- You'll need to know about basic concepts in probability:
  - Random variables
  - Expectations (Mean, variance etc.)
  - Independence (i.i.d. samples from a distribution, ...)
  - ...

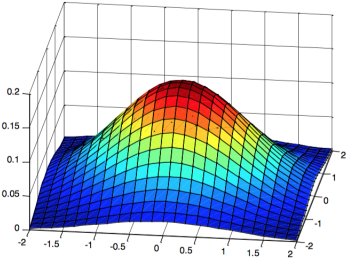
Example: Gaussian distribution



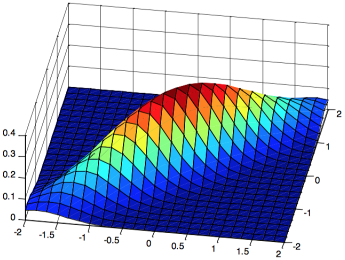
- Standard deviation σ
- Mean μ
- Probability density function $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

Example: Multivariate Gaussian

$$\frac{1}{2\pi\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \qquad \Sigma = \begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{21} & \sigma_2^2\end{pmatrix} \qquad \mu = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}$$



$$\Sigma = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$



$$\Sigma = \begin{pmatrix}1 & 0.9\\ 0.9 & 1\end{pmatrix}$$

Interlude: Expectations

- Expected value of random variable X with probability density function p:

$$\mathbb{E}[X] = \int_X xp(x)\,dx$$

- Expected value of some function of X:

$$\mathbb{E}[f(x)] = \int_X f(x)p(x)\,dx$$

- Linearity of expectation:

$$\mathbb{E}[X+Y] = \mathbb{E}(X) + \mathbb{E}(Y)$$
$$\mathbb{E}[aX] = a\mathbb{E}[X]$$

Achieving generalization

- **Fundamental assumption**: Our data set is generated independently and identically distributed (iid) from some unknown distribution P:

$$(\mathbf{x}_i, y_i) \sim P(\mathbf{X}, Y)$$

- Our goal is to minimize the expected error (true risk) under P:

$$\begin{aligned} R(\mathbf{w}) &= \int P(\mathbf{x}, y)(y - \mathbf{w}^T\mathbf{x})^2 d\mathbf{x}d\mathbf{y} \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}}[(y - \mathbf{w}^T\mathbf{x})^2] \end{aligned}$$

Side note on iid assumption

- When is iid assumption invalid?
  - Time series data
  - Spatially correlated data
  - Correlated noise
  - ...
- Often, can still use machine learning, but one has to be careful in interpreting results.
- Most important: Choose train/test to assess the desired generalization

Estimating the generalization error

- Estimate the true risk by the empirical risk on a sample data set $\mathcal{D}$:

$$\hat{R}_{\mathcal{D}}(\mathbf{w}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} (y - \mathbf{w}^T\mathbf{x})^2$$

- Why might this work?

Estimating the generalization error

- Estimate the true risk by the empirical risk on a sample data set $\mathcal{D}$:

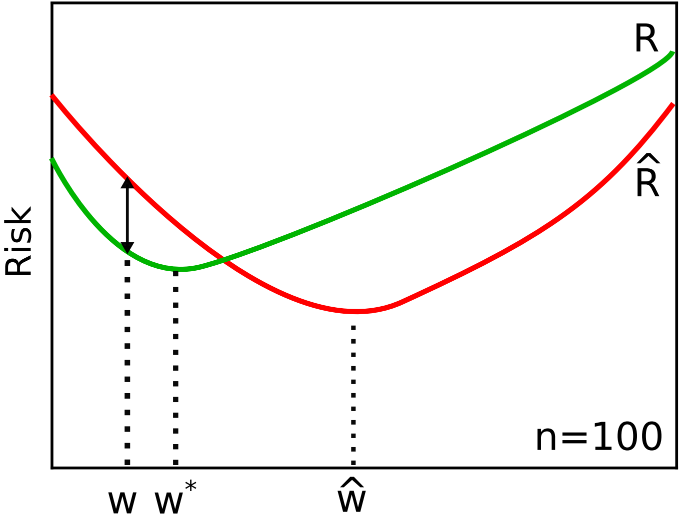
$$\hat{R}_{\mathcal{D}}(\mathbf{w}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} (y - \mathbf{w}^T\mathbf{x})^2$$

- Why might this work?
  Law of large numbers: $\hat{R}_{\mathcal{D}}(\mathbf{w}) \to R(\mathbf{w})$ for any fixed $\mathbf{w}$ almost surely as $|\mathcal{D}| \to \infty$

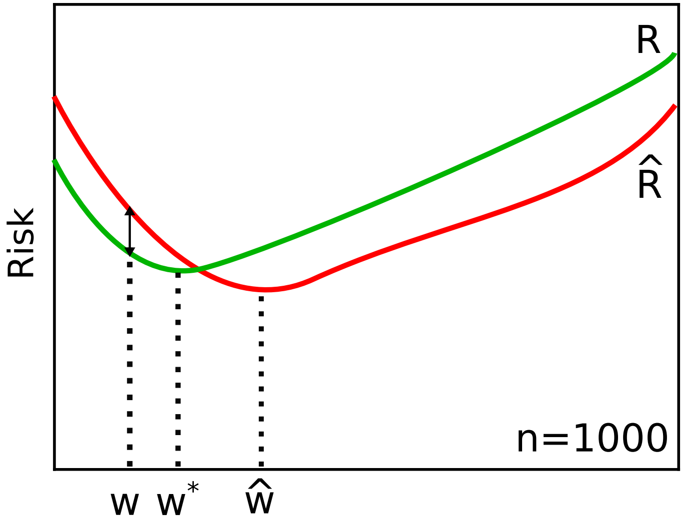
What happens if we optimize on training data?

- Suppose we are given training data $\mathcal{D}$
- Empirical Risk Minimization: $\hat{\mathbf{w}}_{\mathcal{D}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}}(\mathbf{w})$
- Ideally we wish to solve: $\mathbf{w}^* = \arg\min_{\mathbf{w}} R(\mathbf{w})$

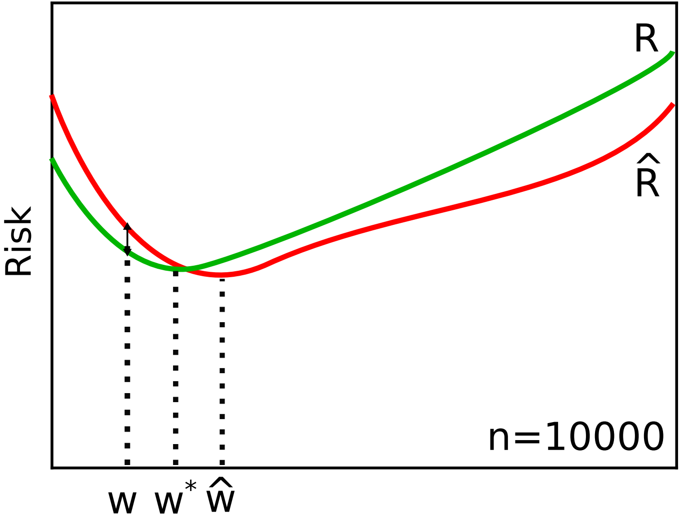
Empirical vs true risk



Empirical vs true risk



Empirical vs true risk



Outlook: Requirements for learning

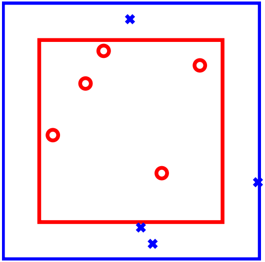
- For learning via empirical risk minimization to be successful, need uniform convergence:

$$\sup_{\mathbf{w}} |R(\mathbf{w}) - \hat{R}_{\mathcal{D}}(\mathbf{w})| \to 0 \quad \text{as} \quad |\mathcal{D}| \to \infty$$

- This is not implied by law of large numbers alone, but depends on model class (holds, e.g., for squared loss on data distributions with bounded support)
  $\to$ Statistical learning theory

What can go wrong? An example

Consider the following classification problem:

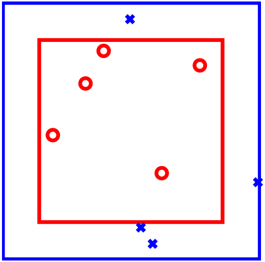


- $x_i \in \mathbb{R}^2$, $y_i \in \{o, x\}$
- Points in $\square$ are o and x otherwise.
- $\text{area}(\square) = 2 \cdot \text{area}(\square)$
- A point drawn uniformly at random is o or x with the same probability.

- Training data: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$
- Classifier: $h'_{\mathcal{D}}(\mathbf{x}) = \begin{cases} y & \text{if } (\mathbf{x}, y) \in \mathcal{D}, \\ o & \text{otherwise.} \end{cases}$
- Empirical risk of $h'_{\mathcal{D}}$: True risk of $h'_{\mathcal{D}}$:

What can go wrong? An example

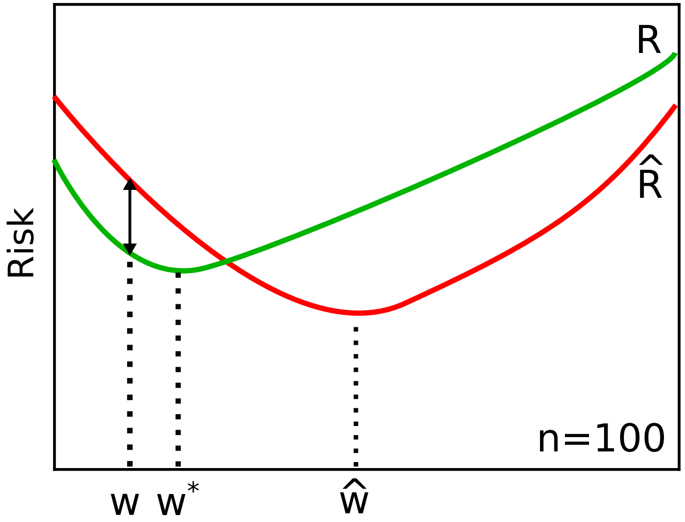
Consider the following classification problem:



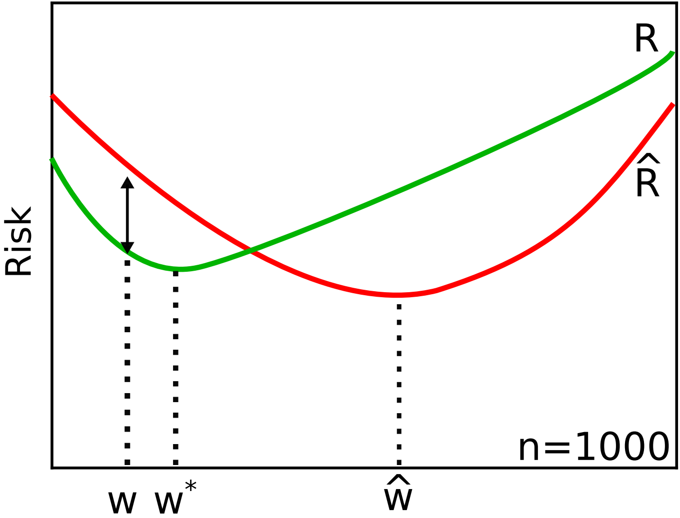
- $x_i \in \mathbb{R}^2$, $y_i \in \{o, x\}$
- Points in $\square$ are o and x otherwise.
- $\text{area}(\square) = 2 \cdot \text{area}(\square)$
- A point drawn uniformly at random is o or x with the same probability.

- Training data: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$
- Classifier: $h'_{\mathcal{D}}(\mathbf{x}) = \begin{cases} y & \text{if } (\mathbf{x}, y) \in \mathcal{D}, \\ o & \text{otherwise.} \end{cases}$
- Empirical risk of $h'_{\mathcal{D}}$: 0 True risk of $h'_{\mathcal{D}}$: $\frac{1}{2}$

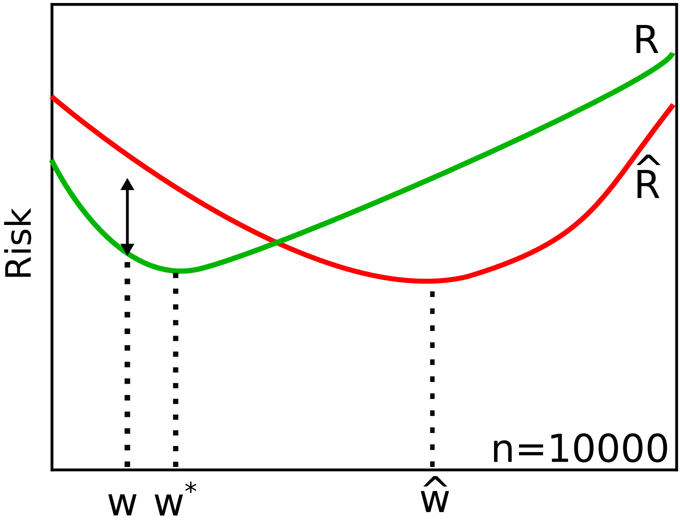
What can go wrong in ERM?



What can go wrong in ERM?



What can go wrong in ERM?



Learning from finite data

- Law of large numbers / uniform convergence are asymptotic statements (with $n \to \infty$)
- In practice one has finite amount of data.
- What can go wrong?

Simple example

$$\hat{\mathbf{w}}_{\mathcal{D}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}}(\mathbf{w}) \qquad \mathbf{w}^* = \arg\min_{\mathbf{w}} R(\mathbf{w})$$

What if we evaluate performance on training data?

$$\hat{\mathbf{w}}_{\mathcal{D}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}}(\mathbf{w}) \qquad\qquad \mathbf{w}^* = \arg\min_{\mathbf{w}} R(\mathbf{w})$$

- In general it holds that $\mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\hat{\mathbf{w}}_{\mathcal{D}})] \leq \mathbb{E}_{\mathcal{D}}[R(\hat{\mathbf{w}}_{\mathcal{D}})]$

What if we evaluate performance on training data?

$$\hat{\mathbf{w}}_{\mathcal{D}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}}(\mathbf{w}) \qquad \mathbf{w}^* = \arg\min_{\mathbf{w}} R(\mathbf{w})$$

- In general it holds that $\mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\hat{\mathbf{w}}_{\mathcal{D}})] \leq \mathbb{E}_{\mathcal{D}}[R(\hat{\mathbf{w}}_{\mathcal{D}})]$
- Thus we obtain an overly optimistic estimate!

Proof

$$\hat{\mathbf{w}}_{\mathcal{D}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}}(\mathbf{w}) \qquad \mathbf{w}^* = \arg\min_{\mathbf{w}} R(\mathbf{w})$$

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\hat{\mathbf{w}}_{\mathcal{D}})] &= \mathbb{E}_{\mathcal{D}}[\min_{\mathbf{w}} \hat{R}_{\mathcal{D}}(\mathbf{w})] && \text{(ERM)} \\
&\leq \min_{\mathbf{w}} \mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\mathbf{w})] && \text{(Jensen's inequality)} \\
&= \min_{\mathbf{w}} \mathbb{E}_{\mathcal{D}}\left[\frac{1}{|\mathcal{D}|}\sum_{i=1}^{\mathcal{D}}(y_i - \mathbf{w}\mathbf{x}_i)^2\right] && \text{(Def. } \hat{R}_{\mathcal{D}}(\mathbf{w})\text{)} \\
&= \min_{\mathbf{w}} \frac{1}{|\mathcal{D}|}\sum_{i=1}^{\mathcal{D}} \underbrace{\mathbb{E}_{(\mathbf{x}_i, y_i)\sim P}\left[(y_i - \mathbf{w}\mathbf{x}_i)^2\right]}_{R(\mathbf{w})} && \text{(lin. exp.)} \\
&= \min_{\mathbf{w}} R(\mathbf{w}) \leq \mathbb{E}_{\mathcal{D}}[R(\hat{\mathbf{w}}_{\mathcal{D}})]
\end{aligned}$$

More realistic evaluation?

- Want to avoid underestimating the prediction error
- Idea: Use separate test set from the same distribution P
- Obtain indep. training and test data $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$
- Optimize $\mathbf{w}$ on training set:

$$\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}_{\text{train}}}(\mathbf{w})$$

- Evaluate on test set:

$$\hat{R}_{\mathcal{D}_{\text{test}}}(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{test}}} (y - \hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}}^T \mathbf{x})^2$$

- Then:

$$\mathbb{E}_{\mathcal{D}_{\text{train}},\mathcal{D}_{\text{test}}}[\hat{R}_{\mathcal{D}_{\text{test}}}(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})] = \mathbb{E}_{\mathcal{D}_{\text{train}}}[R(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})]$$

Why?

Recap: Evaluating predictive performance

- Training error (empirical risk) systematically underestimates true risk:

$$\mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\hat{\mathbf{w}}_{\mathcal{D}})] \leq \mathbb{E}_{\mathcal{D}}[R(\hat{\mathbf{w}}_{\mathcal{D}})]$$

- Using an independent test set avoids this bias:

$$\mathbb{E}_{\mathcal{D}_{\text{train}},\mathcal{D}_{\text{test}}}[\hat{R}_{\mathcal{D}_{\text{test}}}(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})] = \mathbb{E}_{\mathcal{D}_{\text{train}}}[R(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})]$$

First Attempt: Evaluation for Model Selection

- Obtain training and test data $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$
- Fit each candidate model (e.g., degree m of polynomial):

$$\hat{\mathbf{w}}_m = \operatorname*{argmin}_{\mathbf{w}:\ \text{degree}(\mathbf{w}) \leq m} \hat{R}_{\text{train}}(\mathbf{w})$$

- Pick one which does best on test set:

$$\hat{m} = \operatorname*{argmin}_{m} \hat{R}_{\text{test}}(\hat{\mathbf{w}}_m)$$

First Attempt: Evaluation for Model Selection

- Obtain training and test data $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$
- Fit each candidate model (e.g., degree m of polynomial):

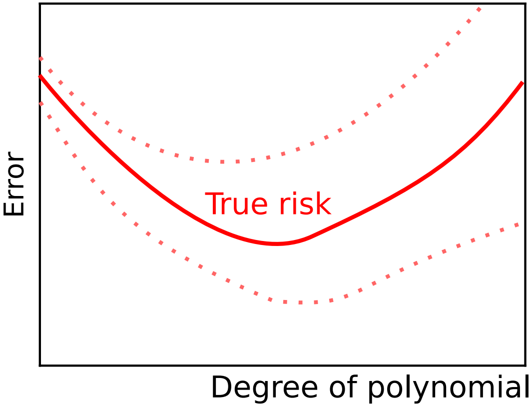
$$\hat{\mathbf{w}}_m = \operatorname*{argmin}_{\mathbf{w}:\ \text{degree}(\mathbf{w}) \leq m} \hat{R}_{\text{train}}(\mathbf{w})$$

- Pick one which does best on test set:

$$\hat{m} = \operatorname*{argmin}_{m} \hat{R}_{\text{test}}(\hat{\mathbf{w}}_m)$$

- Do you see a problem?

Overfitting to test set



- Test error is itself random! Variance usually increases for more complex models
- Optimizing for single test set creates bias

Solution: Pick Multiple Test Sets!

- Key idea: Instead of using a single test set, use multiple test sets and average to decrease variance
- Dilemma: Any data I use for testing I can't use for training

Solution: Pick Multiple Test Sets!

- Key idea: Instead of using a single test set, use multiple test sets and average to decrease variance
- Dilemma: Any data I use for testing I can't use for training
- $\Rightarrow$ Using multiple independent test sets is expensive and wasteful