# Statistical Machine Learning

`https://cvml.ist.ac.at/courses/SML_W18`

**Christoph Lampert**

IST AUSTRIA

*Institute of Science and Technology*

Spring Semester 2018/2019

Lecture 6

# Overview (tentative)

| Date | | no. | Topic |
|---|---|---|---|
| Oct 08 | Mon | 1 | A Hands-On Introduction |
| Oct 10 | Wed | – | self-study (Christoph traveling) |
| Oct 15 | Mon | 2 | Bayesian Decision Theory |
| | | | Generative Probabilistic Models |
| Oct 17 | Wed | 3 | Discriminative Probabilistic Models |
| | | | Maximum Margin Classifiers |
| Oct 22 | Mon | 4 | Generalized Linear Classifiers, Optimization |
| Oct 24 | Wed | 5 | Evaluating Predictors; Model Selection |
| Oct 29 | Mon | – | self-study (Christoph traveling) |
| Oct 31 | Wed | 6 | Overfitting/Underfitting, Regularization |
| Nov 05 | Mon | 7 | Learning Theory I: classical/Rademacher bounds |
| Nov 07 | Wed | 8 | Learning Theory II: miscellaneous |
| Nov 12 | Mon | 9 | Probabilistic Graphical Models I |
| Nov 14 | Wed | 10 | Probabilistic Graphical Models II |
| Nov 19 | Mon | 11 | Probabilistic Graphical Models III |
| Nov 21 | Wed | 12 | Probabilistic Graphical Models IV |
| until Nov 25 | | | final project |

# Learning from Data

In the real world, $p(x, y)$ is unknown, but we have a training set $\mathcal{D}$.

## Definition

Given a training set $\mathcal{D}$, we call it

- a **generative probabilistic approach**:
  if we use $\mathcal{D}$ to build a model $\hat{p}(x, y)$ of $p(x, y)$, and then define

$$f(x) := \mathop{\mathbf{argmin}}_{y \in \mathcal{Y}} \mathop{\mathbb{E}}_{\bar{y} \sim \hat{p}(x, \bar{y})} \ell(\bar{y}, y).$$

- a **discriminative probabilistic approach**:
  if we use $\mathcal{D}$ to build a model $\hat{p}(y|x)$ of $p(y|x)$ and define

$$f(x) := \mathop{\mathbf{argmin}}_{y \in \mathcal{Y}} \mathop{\mathbb{E}}_{\bar{y} \sim \hat{p}(\bar{y}|x)} \ell(\bar{y}, y).$$

- a **decision theoretic approach**: if we use $\mathcal{D}$ to directly seach for a classifier $f$ in a hypothesis class $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$.

# Empirical Risk Minimization

## Definition

Given a training set $\mathcal{D} = \{\,(x^1, y^1), \ldots, (x^n, y^n)\,\}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \operatorname*{\mathbf{argmin}}_{h\in\mathcal{H}} \hat{\mathcal{R}}(h) \qquad \text{for} \quad \hat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(\,y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

# Empirical Risk Minimization

## Definition

Given a training set $\mathcal{D} = \{ (x^1, y^1), \ldots, (x^n, y^n) \}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \operatorname*{\mathbf{argmin}}_{h \in \mathcal{H}} \hat{\mathcal{R}}(h) \qquad \text{for} \quad \hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell( y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

Examples:

- Least-Squared Regression: $\quad \mathbf{min}_w \quad \sum_i (\langle w, x^i \rangle - y^i)^2$

# Empirical Risk Minimization

## Definition

Given a training set $\mathcal{D} = \{\,(x^1, y^1), \dots, (x^n, y^n)\,\}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \operatorname*{\mathbf{argmin}}_{h \in \mathcal{H}} \hat{\mathcal{R}}(h) \qquad \text{for} \quad \hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(\,y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

Examples:

- Least-Squared Regression: $\mathbf{min}_w \quad \sum_i (\langle w, x^i \rangle - y^i)^2$
- Logistic Regression: $\mathbf{min}_w \quad \sum_i \log(1 + e^{-y^i \langle w, x^i \rangle})$

# Empirical Risk Minimization

## Definition

Given a training set $\mathcal{D} = \{ (x^1, y^1), \dots, (x^n, y^n) \}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \operatorname*{\mathbf{argmin}}_{h \in \mathcal{H}} \hat{\mathcal{R}}(h) \qquad \text{for} \quad \hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(\, y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

Examples:

- Least-Squared Regression: $\quad \mathbf{min}_w \quad \sum_i (\langle w, x^i \rangle - y^i)^2$
- Logistic Regression: $\quad \mathbf{min}_w \quad \sum_i \log(1 + e^{-y^i \langle w, x^i \rangle})$
- SVM: $\quad \mathbf{min}_w \quad C \sum_i \mathbf{max}\{0, 1 - y^i \langle w, x^i \rangle\} \quad + \|w\|^2$

# Empirical Risk Minimization

### Definition

Given a training set $\mathcal{D} = \{\,(x^1, y^1), \dots, (x^n, y^n)\,\}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \operatorname*{\mathbf{argmin}}_{h \in \mathcal{H}} \hat{\mathcal{R}}(h) \qquad \text{for} \quad \hat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(\,y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

Examples:

- Least-Squared Regression: $\quad \mathbf{min}_w \quad \sum_i (\langle w, x^i\rangle - y^i)^2$
- Logistic Regression: $\quad \mathbf{min}_w \quad \sum_i \log(1 + e^{-y^i\langle w, x^i\rangle})$
- SVM: $\quad \mathbf{min}_w \quad C\sum_i \mathbf{max}\{0, 1 - y^i\langle w, x^i\rangle\} \quad + \|w\|^2$

We know that for any fixed $h$, $\hat{\mathcal{R}}(h)$ is an unbiased estimate of $\mathcal{R}(h)$.

Does that mean that $\hat{\mathcal{R}}(f)$ is an unbiased estimate of $\mathcal{R}(f)$?

**No, unfortunately not!**

# Empirical Risk Minimization

1) first choose $f : \mathcal{X} \to \mathcal{Y}$, then observe $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\}$:

$$\hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(\, y^i, f(x^i)) \quad \text{unbiased, consistent estimator of } \mathcal{R}(f)$$

- $Z^i := \ell(\, y^i, f(x^i))$ are independent random variables

2) first observe $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\}$, then choose $f$ based on $\mathcal{D}$:

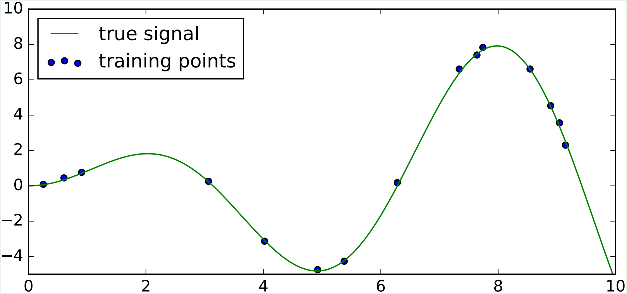
$$\hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(\, y^i, f(x^i)) \qquad\qquad \mathop{\mathbb{E}}_{\mathcal{D}}[\hat{\mathcal{R}}(f)] = ???$$

- $Z^i := \ell(\, y^i, f(x^i))$ are **not** independent, no law of large numbers.

So why would minimizing one be useful for the other?

# Relation between training loss and generalization loss

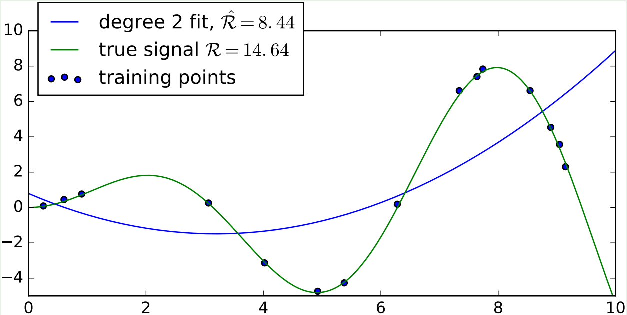
Example: 1D curve fitting



training points

## Relation between training loss and generalization loss

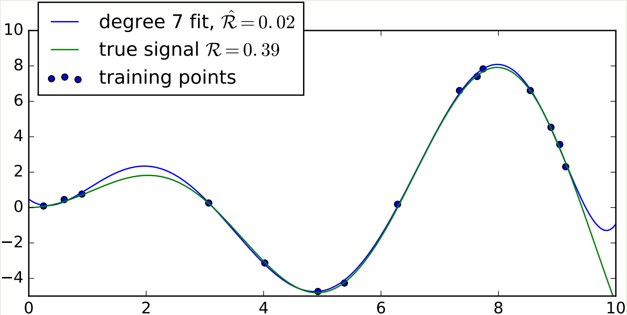
Example: 1D curve fitting



best learned polynomial of degree $2$: large $\hat{\mathcal{R}}$, large $\mathcal{R}$

## Relation between training loss and generalization loss

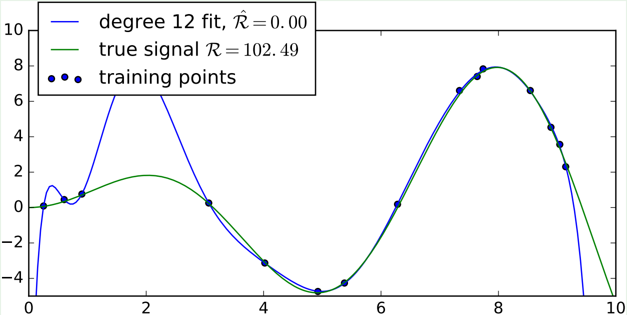
Example: 1D curve fitting



best learned polynomial of degree $7$: small $\hat{\mathcal{R}}$, small $\mathcal{R}$

## Relation between training loss and generalization loss

Example: 1D curve fitting



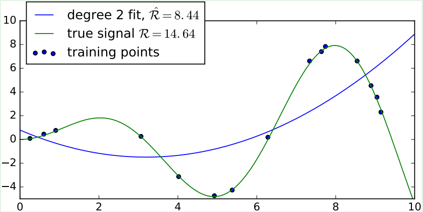
best learned polynomial of degree $12$: small $\hat{\mathcal{R}}$, large $\mathcal{R}$

We found a model $f_{\theta^*}$ by minimizing the training error $\hat{\mathcal{R}}$.

Q: Will its generalization error, $\mathcal{R}$, be small?

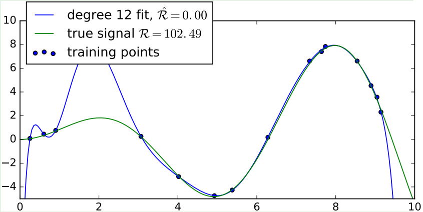
A: **Unfortunately, that is not guaranteed.**

## Underfitting/Overfitting



**Underfitting**

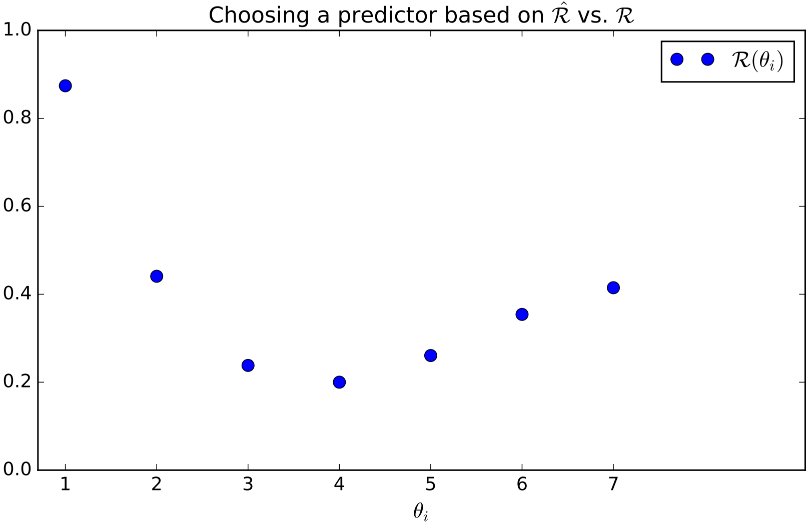
(to some extend) detectable from $\hat{\mathcal{R}}$



**Overfitting**

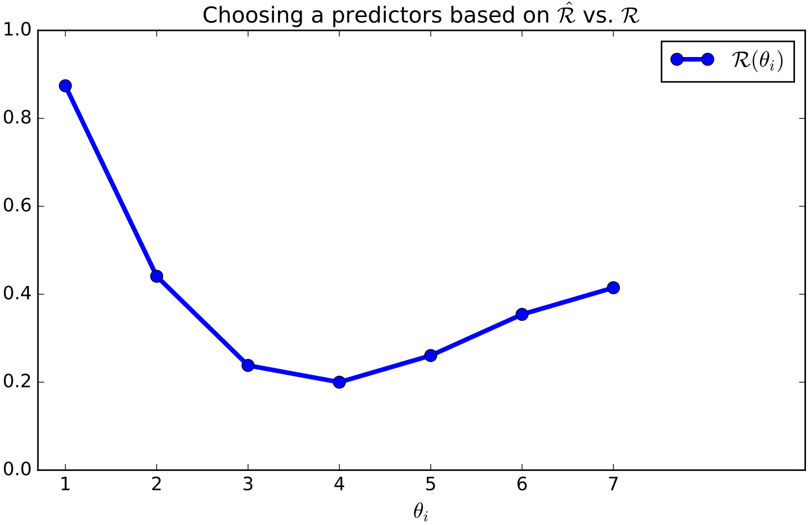
not detectable from $\hat{\mathcal{R}}$ !

# Where does overfitting come from?



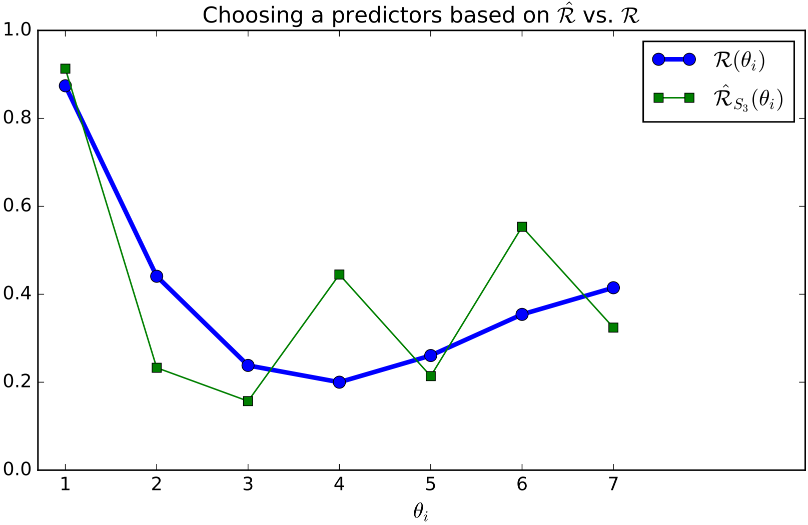
generalization error $\mathcal{R}$ for $7$ different predictors

# Where does overfitting come from?



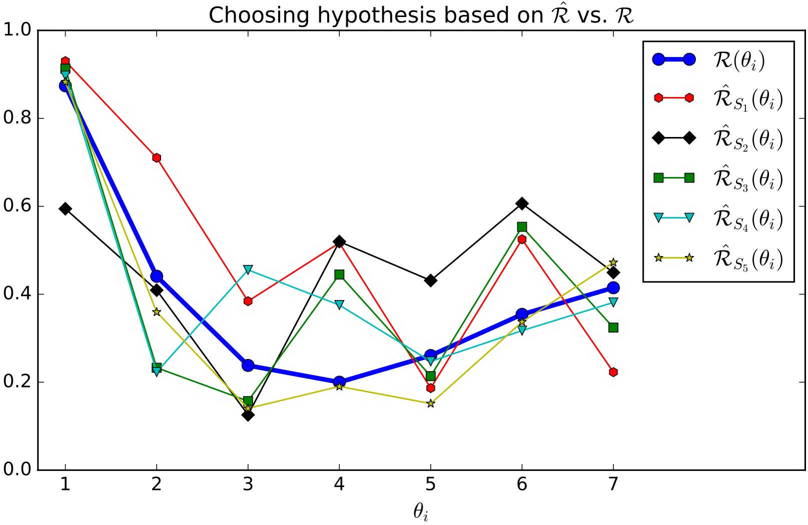
generalization error $\mathcal{R}$ for $7$ different predictors

# Where does overfitting come from?



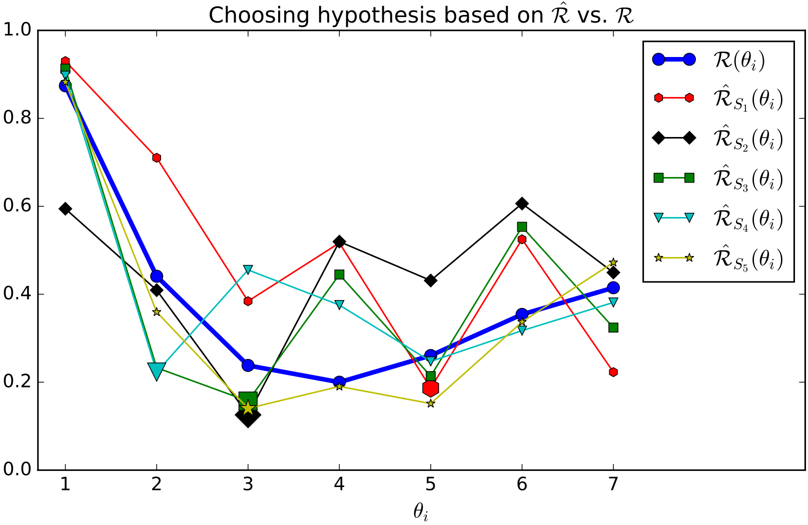
training error $\hat{\mathcal{R}}$ for a training set, $S$

# Where does overfitting come from?



training errors $\hat{\mathcal{R}}$ for $5$ possible training sets

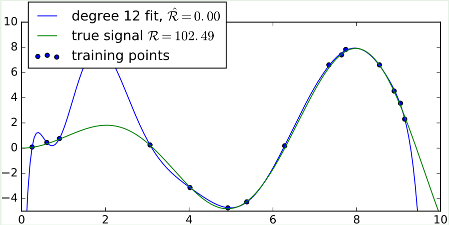
# Where does overfitting come from?



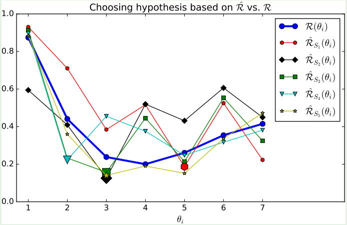
model with smallest training error can have high generalization error

# Preventing Overfitting

## Reminder: Overfitting



**overfitting**

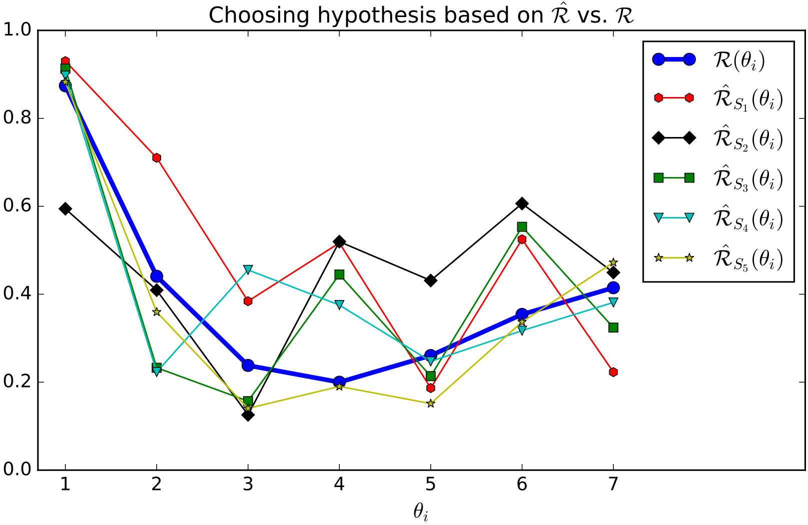


$\hat{\mathcal{R}}$ **vs.** $\mathcal{R}$

**How can we prevent overfitting when learning a model?**

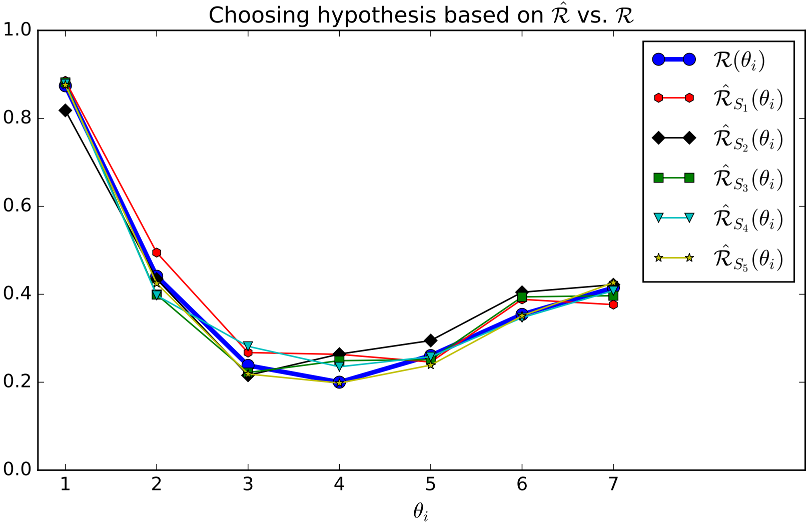
# Preventing overfitting

# Preventing overfitting 1) larger training set



larger training set → smaller variance of $\hat{\mathcal{R}}$

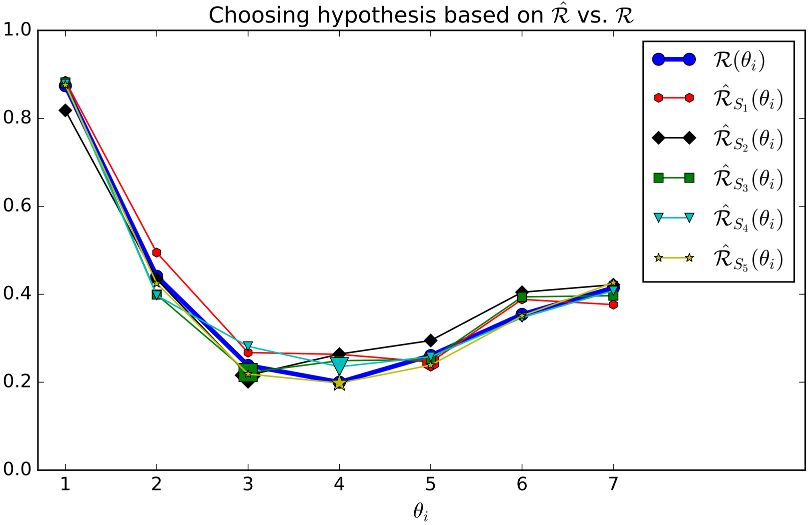
# Preventing overfitting 1) larger training set



Choosing hypothesis based on $\hat{\mathcal{R}}$ vs. $\mathcal{R}$

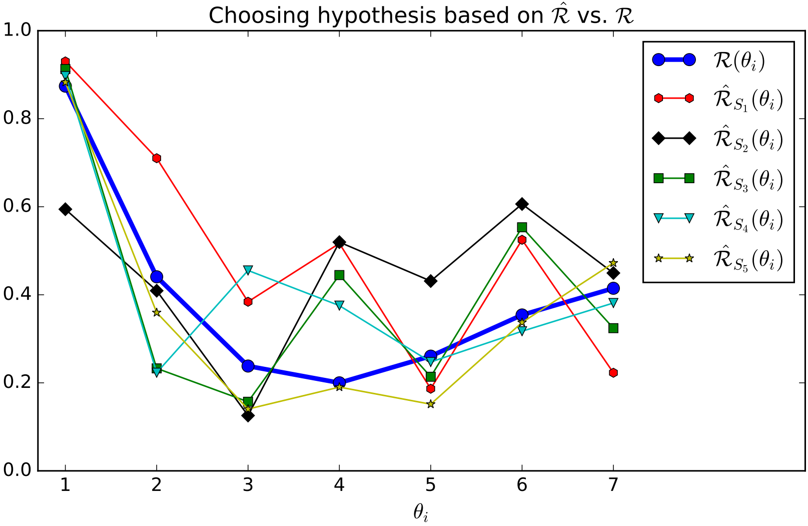
lower probability that $\hat{\mathcal{R}}$ differs strongly from $\mathcal{R}$

# Preventing overfitting 1) larger training set

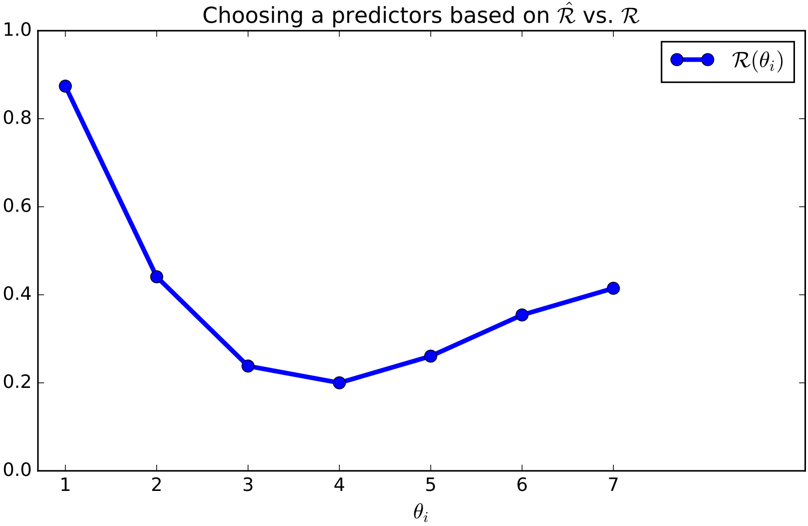


lower probability that $\hat{\mathcal{R}}$ differs strongly from $\mathcal{R} \rightarrow$ overfitting less likely

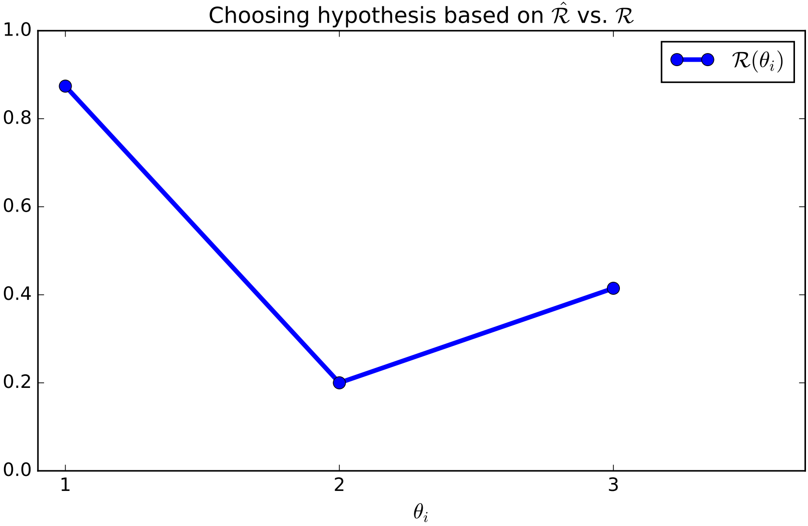
# Preventing overfitting 2) reduce the number of hypotheses



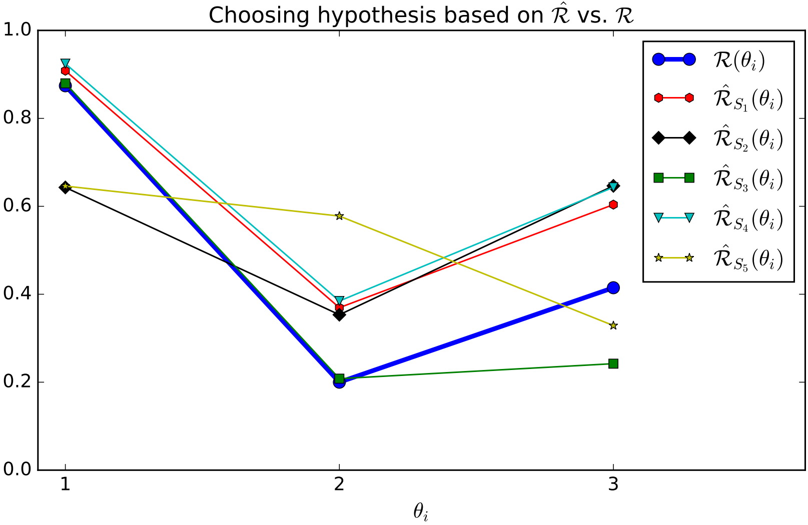
# Preventing overfitting 2) reduce the number of hypotheses



# Preventing overfitting 2) reduce the number of hypotheses

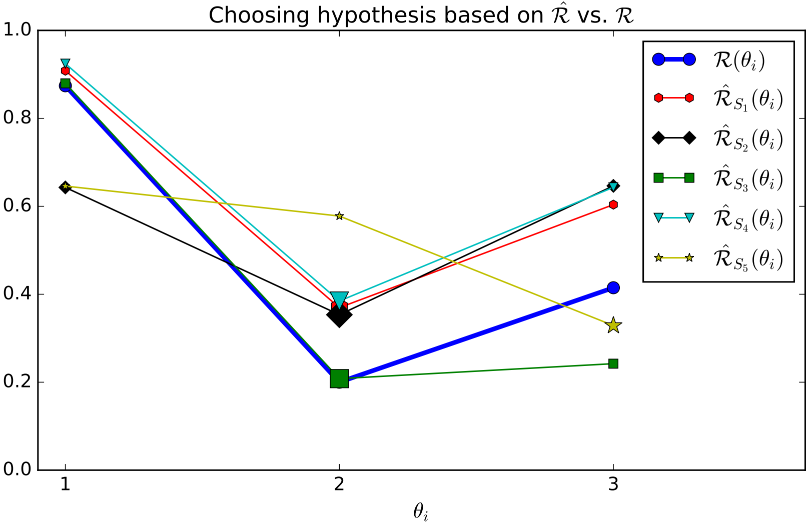


# Preventing overfitting 2) reduce the number of hypotheses



fewer models → lower probability of a model with small $\hat{\mathcal{R}}$ but high $\mathcal{R}$

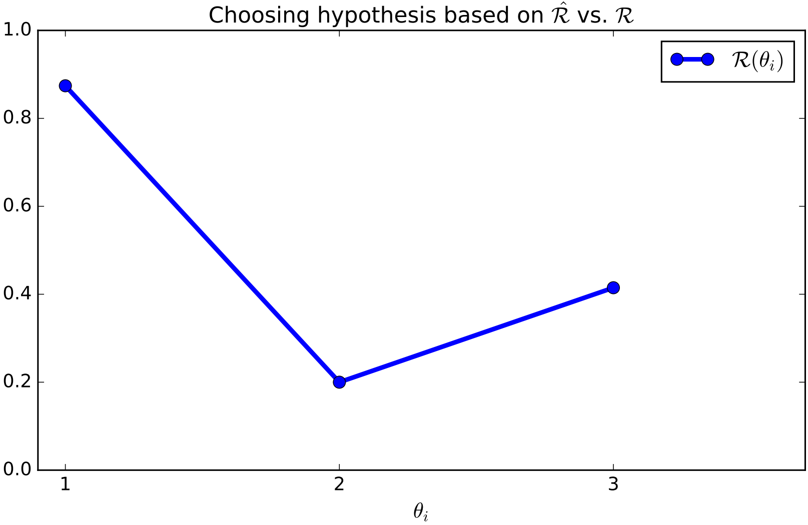
# Preventing overfitting 2) reduce the number of hypotheses



fewer models → lower probability of a model with small $\hat{\mathcal{R}}$ but high $\mathcal{R}$

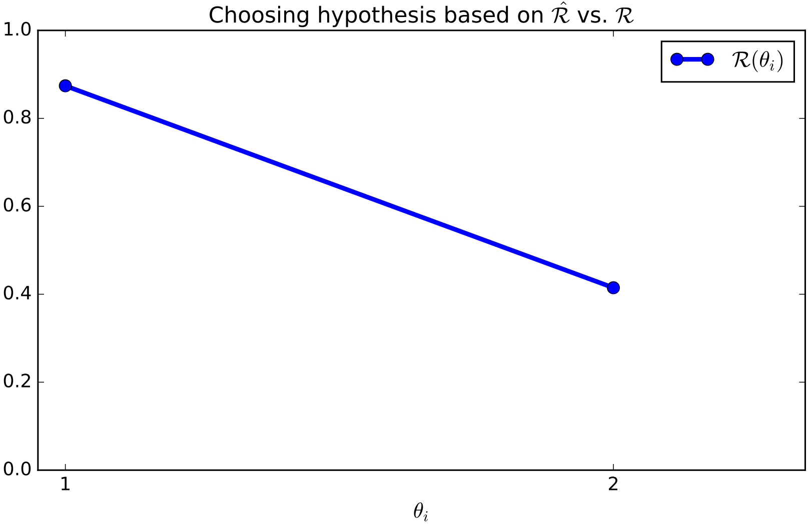
# But: danger of underfitting

# But: danger of underfitting



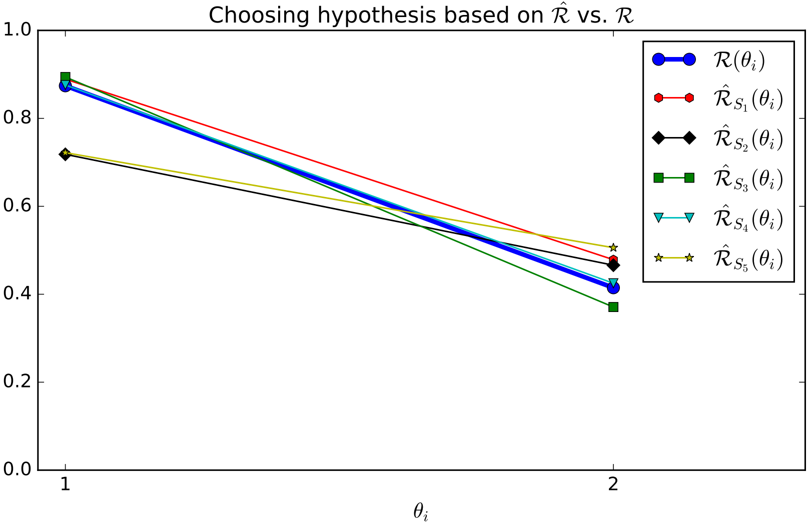
to few models select to from → danger that no model with low $\mathcal{R}$ is left!

# But: danger of underfitting



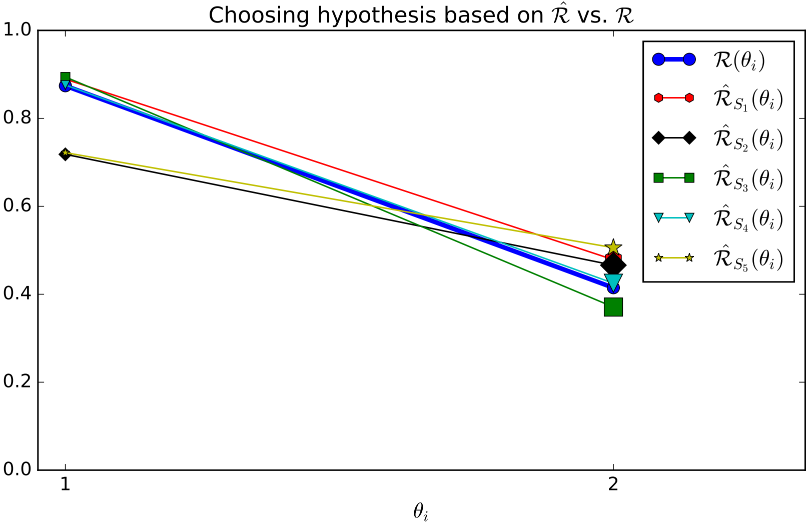
to few models select to from → danger that no model with low $\mathcal{R}$ is left!

# But: danger of underfitting



**Underfitting!**

# But: danger of underfitting



**Underfitting!**

**Overfitting happens when . . .**

- there are too many models to choose from
  (not strictly true: there's usually infinitely many models anyway)
- the models we search over are too "flexible", so they fit not only the signal but also the noise
  (not strictly true: the models themselves are not "flexible" at all)
- the models have too many free parameters
  (not strictly true: even models with very few parameters can overfit)

**How to avoid overfitting?** Use a model class that is

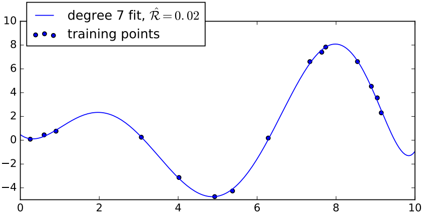
- "as simple as possible", but
- still contains a model with low $\hat{\mathcal{R}}$

# Regularization

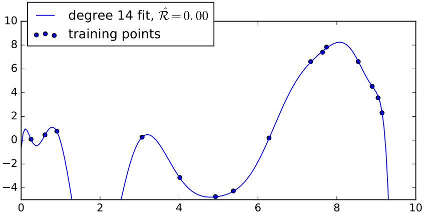
# Regularization

Models with big difference between training error and generalization error are typically **extreme cases**:

- a large number of model parameters
- large values of the model parameters
- for polynomials: high degree , etc.



coeffs: $\theta_i \in [-2.4, 4.6]$

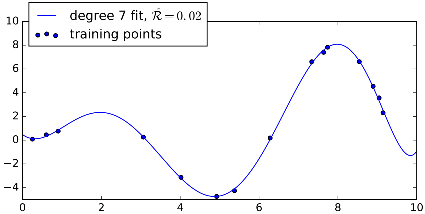


coeffs: $\theta_i \in [-1312.5, 1136.6]$

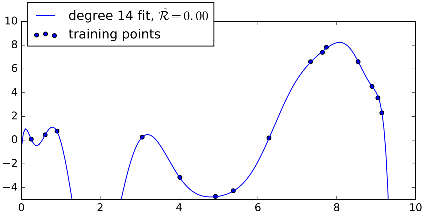
# Regularization

Models with big difference between training error and generalization error are typically **extreme cases**:

- a large number of model parameters
- large values of the model parameters
- for polynomials: high degree ,   etc.



coeffs: $\theta_i \in [-2.4, 4.6]$



coeffs: $\theta_i \in [-1312.5, 1136.6]$

**Regularization:** avoid overfitting by preventing extremes to occur

- explicit regularization (changing the objective function)
- implicit regularization (modifying the optimization procedure)

# Explicit regularization

Add a regularization term (=regularizer) to the empirical risk that gives large values to extreme parameter choices.

## Regularized risk minimization

Take a training set, $S = \{(x^1, y^1), \dots, (x^n, y^n)\}$, find $\theta^*$ by solving,

$$\min_{\theta} J_\lambda(\theta) \quad \text{with} \quad J_\lambda(\theta) = \underbrace{\sum_{i=1}^{n} \ell(y^i, f_\theta(x^i))}_{\text{empirical risk}} + \underbrace{\lambda\Omega(\theta)}_{\text{regularizer}}$$

e.g. with $\Omega(\theta) = \|\theta\|_{L^2}^2 = \sum_j \theta_j^2$ or $\Omega(\theta) = \|\theta\|_{L^1} = \sum_j |\theta_j|$

# Explicit regularization

Add a regularization term (=regularizer) to the empirical risk that gives large values to extreme parameter choices.

## Regularized risk minimization

Take a training set, $S = \{(x^1, y^1), \dots, (x^n, y^n)\}$, find $\theta^*$ by solving,

$$\min_\theta J_\lambda(\theta) \quad \text{with} \quad J_\lambda(\theta) = \underbrace{\sum_{i=1}^{n} \ell(y^i, f_\theta(x^i))}_{\text{empirical risk}} + \underbrace{\lambda\Omega(\theta)}_{\text{regularizer}}$$

e.g. with $\Omega(\theta) = \|\theta\|_{L^2}^2 = \sum_j \theta_j^2$ or $\Omega(\theta) = \|\theta\|_{L^1} = \sum_j |\theta_j|$

Optimization looks for model with small empirical risk, but also small absolute values of the model parameters.

**Regularization (hyper)parameter** $\lambda \geq 0$: trade-off between both.

- $\lambda = 0$: empirical risk minimization (risk of overfitting)
- $\lambda \to \infty$: all parameters $0$ (risk of underfitting)

## Explicit regularization

Add a regularization term (=regularizer) to the empirical risk that gives large values to extreme parameter choices.

### Regularized risk minimization

Take a training set, $S = \{(x^1, y^1), \dots, (x^n, y^n)\}$, find $\theta^*$ by solving,

$$\min_\theta J_\lambda(\theta) \quad \text{with} \quad J_\lambda(\theta) = \underbrace{\sum_{i=1}^{n} \ell(y^i, f_\theta(x^i))}_{\text{empirical risk}} + \underbrace{\lambda\Omega(\theta)}_{\text{regularizer}}$$

e.g. with $\Omega(\theta) = \|\theta\|^2_{L^2} = \sum_j \theta_j^2$ or $\Omega(\theta) = \|\theta\|_{L^1} = \sum_j |\theta_j|$

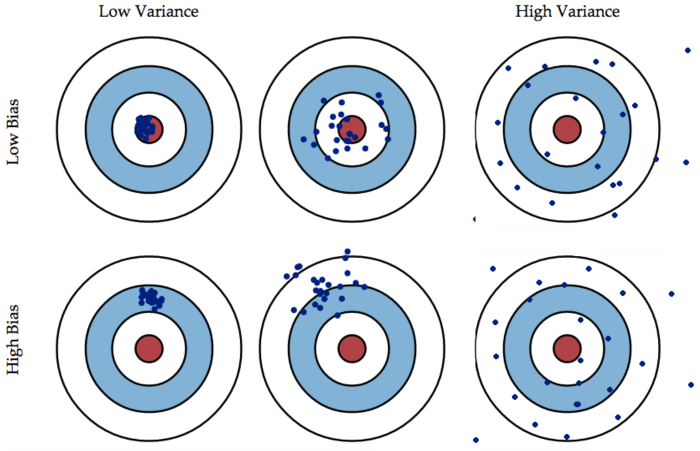
**Examples:**

- Ridge Regression: $\mathbf{min}_w \quad \lambda\|w\|^2 + \sum_i (\langle w, x^i \rangle - y^i)^2$
- Logistic Regression: $\mathbf{min}_w \quad \lambda\|w\|^2 + \sum_i \log(1 + e^{-y^i\langle w, x^i\rangle})$
- SVM: $\mathbf{min}_w \quad \|w\|^2 + C\sum_i \mathbf{max}\{0, 1 - y^i\langle w, x^i\rangle\}$

# Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noise estimate of the generalization error, $\mathcal{R}$

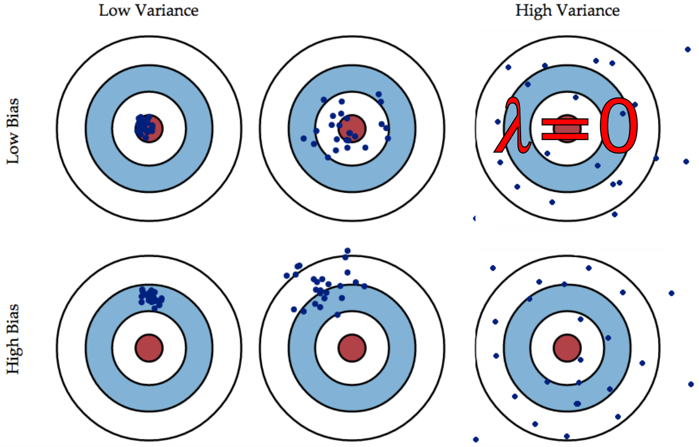
- original risk $\hat{\mathcal{R}}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to $0$, but the bias gets very big



# Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noise estimate of the generalization error, $\mathcal{R}$

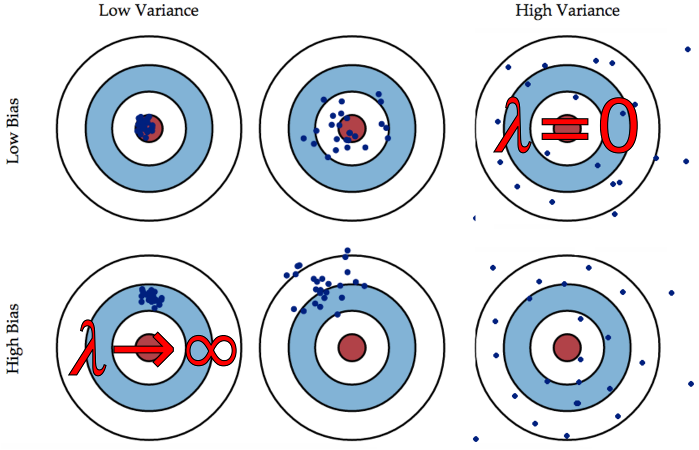
- original risk $\hat{\mathcal{R}}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to $0$, but the bias gets very big



# Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noise estimate of the generalization error, $\mathcal{R}$

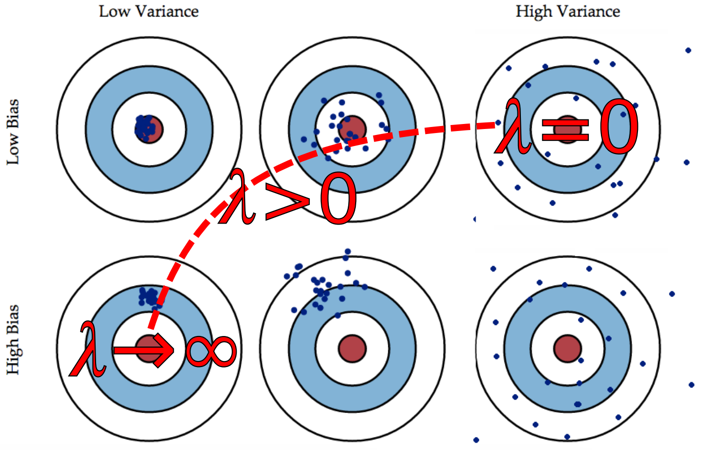
- original risk $\hat{\mathcal{R}}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to $0$, but the bias gets very big



# Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noise estimate of the generalization error, $\mathcal{R}$

- original risk $\hat{\mathcal{R}}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to $0$, but the bias gets very big



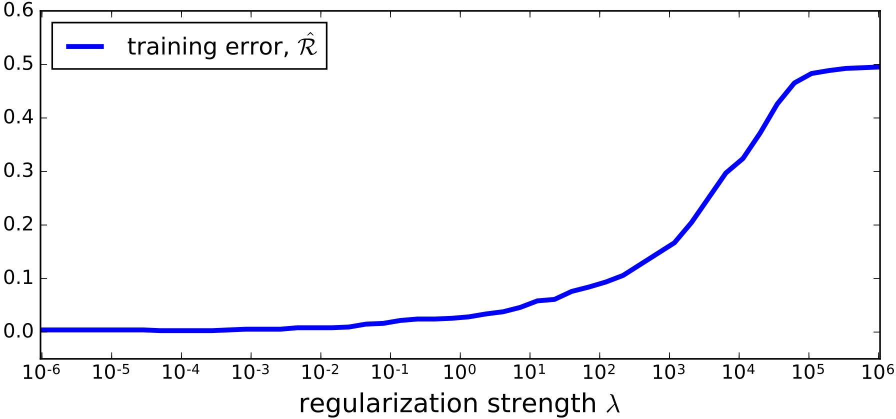
# Example: regularized linear least-squared regression

$$\min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x^i - y^i)^2 + \lambda \|w\|^2$$

# Example: regularized linear least-squared regression

$$\min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x^i - y^i)^2 + \lambda \|w\|^2$$

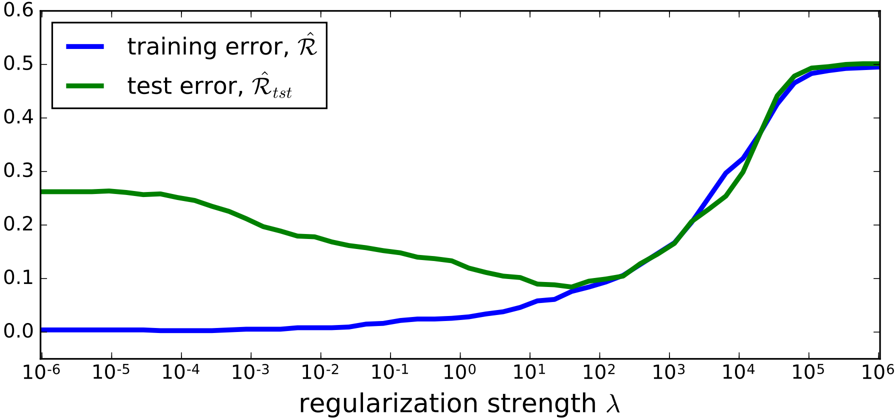
Train/test error for classifier $c(x) = \operatorname{sign}\langle w, x\rangle$ from minimizing $J_\lambda$ with varying amounts of regularization:



# Example: regularized linear least-squared regression

$$\min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x^i - y^i)^2 + \lambda \|w\|^2$$

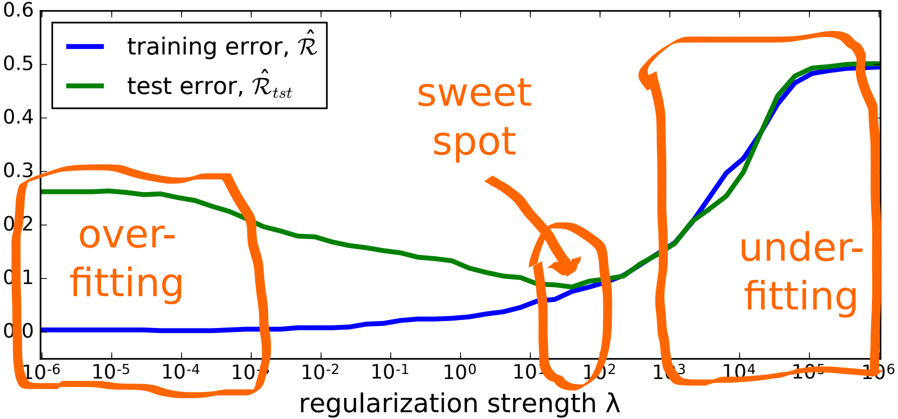
Train/test error for classifier $c(x) = \operatorname{sign}\langle w, x\rangle$ from minimizing $J_\lambda$ with varying amounts of regularization:



# Example: regularized linear least-squared regression

$$\min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x^i - y^i)^2 + \lambda \|w\|^2$$

Train/test error for classifier $c(x) = \operatorname{sign}\langle w, x\rangle$ from minimizing $J_\lambda$ with varying amounts of regularization:



# Implicit regularization

Numerical optimization is performed iteratively, e.g. gradient descent

### Gradient descent optimization

```
initialize $\theta^{(0)}$
for $t = 1, 2, \dots$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$        ($\eta_t \in \mathbb{R}$ is some stepsize rule)
until convergence
```

**Implicit regularization** methods modify these steps, e.g.

- early stopping
- weight decay
- data jittering
- dropout

# Implicit regularization: early stopping

## Gradient descent optimization with early stopping

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots, T$        ($T \in \mathbb{N}$ is number of steps)
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
```

# Implicit regularization: early stopping

## Gradient descent optimization with early stopping

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots, T$        ($T \in \mathbb{N}$ is number of steps)
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
```

**Early stopping:** stop optimization before convergence

- idea: if parameters are update only a small number of time, they might not reach extreme values
- $T$ hyperparameter controls trade-off:
  - large $T$: parameters approach risk minimizer → risk of overfitting
  - small $T$: parameters stay close to initialization → risk of underfitting

# Implicit regularization: weight decay

## Gradient descent optimization with weight decay

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
    $\theta^{(t)} \leftarrow \gamma \theta^{(t)}$        for, e.g., $\gamma = 0.99$
until convergence
```

# Implicit regularization: weight decay

## Gradient descent optimization with weight decay

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
    $\theta^{(t)} \leftarrow \gamma\theta^{(t)}$        for, e.g., $\gamma = 0.99$
until convergence
```

**Weight decay:**

Multiply parameters with a constant smaller than $1$ in each iteration

- two 'forces' in parameter update:
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
    pull towards empirical risk minimizer → risk of overfitting
  - $\theta^{(t)} \leftarrow \gamma\theta^{(t)}$ pulls towards $0$ → risk of underfitting
- convergence: both effects cancel out → trade-off controlled by $\eta_t, \gamma$

Note: essentially same effect as explicit regularization with $\Omega = \frac{\gamma}{2}\|\theta\|^2$

# Implicit regularization: data jittering (="virtual samples")

## Gradient descent optimization with data jittering

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots$
    for $i = 1, \ldots, n$:
        $\tilde{x}^i \leftarrow$ randomly perturbed version of $x^i$
    set $\tilde{J}(\theta) = \sum_{i=1}^{n} \ell( y^i, f_\theta(\tilde{x}^i))$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta \tilde{J}(\theta^{(t-1)})$
until convergence
```

# Implicit regularization: data jittering (="virtual samples")

## Gradient descent optimization with data jittering

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots$
    for $i = 1, \ldots, n$:
        $\tilde{x}^i \leftarrow$ randomly perturbed version of $x^i$
    set $\tilde{J}(\theta) = \sum_{i=1}^{n} \ell(\, y^i, f_\theta(\tilde{x}^i))$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta \tilde{J}(\theta^{(t-1)})$
until convergence
```

**Jittering:** use randomly perturbed examples in each iteration

- idea: a good model should be robust to small changes of the data
- simulate (infinitely-)large training set → hopefully less overfitting
  (also possible: just create large training set of jittered examples in the beginning)
- problem: coming up with perturbations needs *domain knowledge*

# Implicit regularization: dropout

## Gradient descent optimization with dropout

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots$
    $\tilde{\theta} \leftarrow \theta^{(t-1)}$ with a random fraction $p$ of values set to $0$, e.g. $p = \frac{1}{2}$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\tilde{\theta})$
until convergence
```

# Implicit regularization: dropout

## Gradient descent optimization with dropout

```
initialize $\theta^{(0)}$
for $t = 1, 2, \ldots$
    $\tilde{\theta} \leftarrow \theta^{(t-1)}$ with a random fraction $p$ of values set to $0$, e.g. $p = \frac{1}{2}$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\tilde{\theta})$
until convergence
```

**Dropout:** every time we evaluate the model, a random subset of its parameters are set to zero.

- aims for model with low empirical risk even if parameters are missing
- idea: no single parameter entry can become 'too important'
- similar to jittering, but without need for domain knowledge about $x$'s
- overfitting vs. underfitting tradeoff controlled by $p$

# Regularization

**Often, more than one regularization techniques are combined, e.g.**

Explicit regularization: e.g. *"elastic net"*

- $\Omega(\theta) = \alpha \|\theta\|_{L^2}^2 + (1-\alpha)\|\theta\|_{L^1}$

Explicit/implicit regularization: e.g. large-scale support vector machines

- $\Omega(\theta) = \|\theta\|_{L^2}^2$, early stopping, potentially jittering

Implicit regularization: e.g. deep networks

- early stopping, weight decay, dropout, potentially jittering

# Summary

## Regularization can prevent overfitting

Intuition: avoid "extreme" models, e.g. very large parameter values

## Explicit Regularization: modify object function

## Implicit Regularization: change optimization procedure

## Regularization introduces additional (hyper)parameters

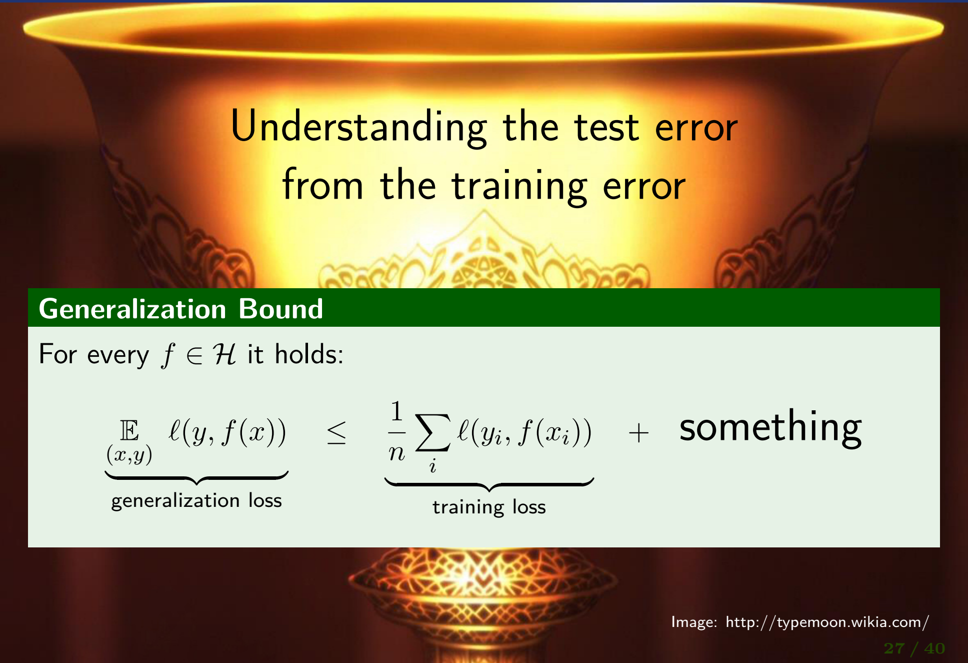
How much of a regularization method to apply is a free parameter, often called *regularization constant*. The optimal values are problem specific.

## The Holy Grail of Statistical Machine Learning

Understanding the test error from the training error

Image: http://typemoon.wikia.com/

# The Holy Grail of Statistical Machine Learning



Understanding the test error from the training error

## Generalization Bound

For every $f \in \mathcal{H}$ it holds:

$$\underbrace{\mathop{\mathbb{E}}_{(x,y)} \ell(y, f(x))}_{\text{generalization loss}} \leq \underbrace{\frac{1}{n} \sum_{i} \ell(y_i, f(x_i))}_{\text{training loss}} + \text{something}$$

Image: http://typemoon.wikia.com/

## Typical structure of a generalization bound

Fixed learning setting:

- input data $\mathcal{X}$, output space $\mathcal{Y}$,
- data distribution $p$ over $\mathcal{X} \times \mathcal{Y}$ (with some properties),
- hypothesis set $\mathcal{H} \subset \{f : \mathcal{X} \to \mathcal{Y}\}$,
- loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ (with some properties),

For any $\delta > 0$, the following statement holds with probablity at least $1 - \delta$ over the (random) training set $\mathcal{D}_n = \{(x^1, y^1), \dots, (x^n, y^n)\} \overset{i.i.d.}{\sim} p$.

For all $f \in \mathcal{H}$:

$$\mathop{\mathbb{E}}_{(x,y)} \ell(y, f(x)) \leq \frac{1}{n} \sum_{i=1}^{n} \ell(y, f(x)) \quad + \quad \text{something}$$

"something" typically increases for $\delta \to 0$ and decreases for $n \to \infty$.

Observation: if inequality holds, it holds uniformly for all $f$.
→ by minimizing the right hand side, we can find the "most promising" $f$

Reminder: (soft-margin) support vector machine (SVM):

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i} \mathbf{max}\{0, 1 - y_i\langle w, x_i\rangle\}$$

Reminder: (soft-margin) support vector machine (SVM):

$$\min_{w} \ \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i} \max\{0, 1 - y_i\langle w, x_i\rangle\}$$

### Example: SVM radius/margin bound

Let $\ell(x, y; w) := \max\{0, 1 - y\langle w, x\rangle\}$ be the *hinge loss*. Let $p$ be a distribution on $\mathbb{R}^d \times \mathcal{Y}$ such that $\Pr\{\,\|x\| \leq R\,\} = 1$ and let $\mathcal{H} = \{w : \|w\| \leq B\}$.

Then, with prob. at least $1 - \delta$ over $\mathcal{D}_m \overset{i.i.d.}{\sim} p$ the following inequality holds for all $w \in \mathcal{H}$:

$$\mathop{\mathbb{E}}_{(x,y)\sim p}[\![\langle w, x\rangle \neq y]\!] \leq \frac{1}{m}\sum_{i=1}^{m}\ell(x_i, y_i, w) + \frac{2BR}{\sqrt{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}.$$

Properties:

- uniform in $w$, i.e. holds even for minimizer of r.h.s. → almost SVM
- $B$ is a upper bound on $\|w\|$ → small $\|w\|$ are most promising
- dimensionality of $x$ does not show up, no curse of dimensionality!

# Excurse: Concentration of Measure II

# Hoeffding's Lemma and Inequality

**Lemma (Hoeffding's Lemma)**

*Let $Z$ be a random variable that takes values in $[a, b]$ and $\mathbb{E}[Z] = 0$. Then, for every $\lambda > 0$,*

$$\mathbb{E}[e^{\lambda X}] \leq e^{\frac{\lambda^2 (b-a)^2}{8}}.$$

Proof: Exercise...

## Lemma (Hoeffding's Inequality)

*Let $Z_1, \ldots, Z_m$ be i.i.d. random variables that take values in the interval $[a, b]$. Let $\bar{Z} = \frac{1}{m}\sum_{i=1}^{m} Z_i$ and denote $\mathbb{E}[\bar{Z}] = \mu$. Then, for any $\epsilon > 0$,*

$$\mathbb{P}\left[\left(\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right) > \epsilon\right] \leq e^{-m\frac{\epsilon^2}{(b-a)^2}}.$$

*and*

$$\mathbb{P}\left[\left(\mu - \frac{1}{m}\sum_{i=1}^{m} Z_i\right) > \epsilon\right] \leq e^{-m\frac{\epsilon^2}{(b-a)^2}}.$$

*and*

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right| > \epsilon\right] \leq 2e^{-m\frac{\epsilon^2}{(b-a)^2}}.$$

# Hoeffding's Inequality – Proof

Define new RVs: $X_i = Z_i - \mathbb{E}[Z_i]$, $\bar{X} = \frac{1}{m}\sum_i X_i$

- $\mathbb{E}[X_i] = 0$; $\mathbb{E}[\bar{X}] = 0$; each $X_i$ takes values in $[a - \mathbb{E}[Z_i], b - \mathbb{E}[Z_i]]$

Use 1) monotonicity of $\exp$ and 2) Markov's inequality to check

$$\mathbb{P}[\bar{X} \geq \epsilon] \;\overset{1)}{=}\; \mathbb{P}[e^{\lambda\bar{X}} \geq e^{\lambda\epsilon}] \;\overset{2)}{\leq}\; e^{-\lambda\epsilon}\,\mathbb{E}[e^{\lambda\bar{X}}]$$

From 3) the independence of the $X_i$ we have

$$\mathbb{E}[e^{\lambda\bar{X}}] \;=\; \mathbb{E}[\prod_{i=1}^{n} e^{\lambda X_i/m}] \;\overset{3)}{=} \prod_{i=1}^{n} \mathbb{E}[e^{\lambda X_i/m}]$$

Use 4) Hoeffding's Lemma for every $i$:

$$\mathbb{E}[e^{\lambda X_i/m}] \;\overset{4)}{\leq} e^{\frac{\lambda^2(b-a)^2}{8m^2}}.$$

In combination:

$$\mathbb{P}[\bar{X} \geq \epsilon] \;\leq\; e^{-\lambda\epsilon}e^{\frac{\lambda^2(b-a)^2}{8m}}$$

# Hoeffding's Inequality – Proof cont.

Previous step:

$$\mathbb{P}[\bar{X} \geq \epsilon] \;\leq\; e^{-\lambda\epsilon} e^{\frac{\lambda^2 (b-a)^2}{8m}}$$

So far, $\lambda$ was arbitrary. Now we set $\lambda = \frac{4m\epsilon}{(b-a)^2}$

$$\mathbb{P}[\bar{X} \geq \epsilon] \;\leq\; e^{-\frac{4m\epsilon}{(b-a)^2}\epsilon + \left(\frac{4m\epsilon}{(b-a)^2}\right)^2 \frac{(b-a)^2}{8m}} = e^{-\frac{2m\epsilon^2}{(b-a)^2}}$$

This proves the first statement.

If we repeat the same steps again for $-\bar{X}$ instead of $X$, we get

$$\mathbb{P}[\bar{X} \leq -\epsilon] \;\leq\; e^{-\frac{2m\epsilon^2}{(b-a)^2}}$$

This proves the second statement.

Use the *union bound*: $\mathbb{P}[A \vee B] \leq \mathbb{P}[A] + \mathbb{P}[B]$, to combine both directions:

$$\mathbb{P}[|\bar{X}| \geq \epsilon] \;=\; \mathbb{P}[\,(\bar{X} \geq \epsilon) \vee (\bar{X} \leq -\epsilon)\,] \;\leq\; 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.$$

□

# How large should my test set be?

$$\boxed{\mathbb{P}\Big[\,\Big|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\Big| > \epsilon\,\Big] \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.}$$

Setup: fixed classifier $g : \mathcal{X} \to \mathcal{Y}$

- test set $\mathcal{D} = \{(x^1, y^1) \ldots, (x^m, y^m)\} \overset{i.i.d.}{\sim} p(x, y)$,
- random variables $Z_i = [\![g(x^i) \neq y^i]\!] \in \{0, 1\}$, $\rightarrow \quad b - a = 1$
- $\mathbb{E}[Z_i] = \mathbb{E}\{[\![g(x^i) \neq y^i]\!]\} = \mu$ (test error of $g$)

Setup: $m = \frac{1}{2}\log(\frac{2}{\delta})/\epsilon^2$.

For fixed confidence $\delta = 0.1 \Rightarrow \epsilon = \sqrt{\log(20)/(2m)} \approx 1.22\sqrt{\frac{1}{m}}$

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right| \leq 1.22\sqrt{\frac{1}{m}}\right] \geq 0.9$$

To be 90%-certain that the error is within $0.05$, use $m \geq 600$.

# How large should my test set be?

$$\boxed{\mathbb{P}[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right| > \epsilon\,] \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.}$$

Setup: fixed classifier $g : \mathcal{X} \to \mathcal{Y}$

- test set $\mathcal{D} = \{(x^1, y^1)\ldots,(x^m, y^m)\} \overset{i.i.d.}{\sim} p(x,y)$,
- random variables $Z_i = [\![g(x^i) \neq y^i]\!] \in \{0,1\}$, $\to \quad b - a = 1$
- $\mathbb{E}[Z_i] = \mathbb{E}\{[\![g(x^i) \neq y^i]\!]\} = \mu \qquad$ (test error of $g$)

Setup: $m = \frac{1}{2}\log(\frac{2}{\delta})/\epsilon^2$.

For fixed confidence $\delta = 0.1 \Rightarrow \epsilon = \sqrt{\log(20)/(2m)} \approx 1.22\sqrt{\frac{1}{m}}$

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right| \leq 1.22\sqrt{\frac{1}{m}}\right] \geq 0.9$$

To be 90%-certain that the error is within $0.05$, use $m \geq 600$.

To be 99%-certain that the error is within $0.05$, use $m \geq 1060$.

To be 90%-certain that the error is within $0.005$, use $m \geq 59914$.

# Difference: Chebyshev's vs. Hoeffding's Inequality

With $\hat{\mathcal{R}} = \frac{1}{m}\sum_{i=1}^{m} Z_i$ and $\mathcal{R} = \mathbb{E}[\frac{1}{m}\sum_{i=1}^{m} Z_i]$:

- Chebyshev's: $\mathrm{Var}[Z_i] \leq C$

$$\mathbb{P}\left[|\hat{\mathcal{R}} - \mathcal{R}| > \sqrt{\frac{C}{\delta m}}\right] \leq \delta, \qquad \mathbb{P}\left[\,|\hat{\mathcal{R}} - \mathcal{R}| > \epsilon\,\right] \leq \frac{C}{\epsilon^2 m}$$

- interval decreases like $\frac{1}{\sqrt{m}}$, confidence grows like $1 - \frac{1}{m}$
- Hoeffding's: $Z_i$ takes values in $[a, b]$:

$$\mathbb{P}\left[|\hat{\mathcal{R}} - \mathcal{R}| > \sqrt{\frac{(b-a)^2 \log \frac{2}{\delta}}{m}}\right] \leq \delta, \quad \mathbb{P}\left[\,|\hat{\mathcal{R}} - \mathcal{R}| > \epsilon\,\right] \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.$$

- interval decreases like $\frac{1}{\sqrt{m}}$, confidence grows like $1 - e^{-m}$

Both are typical **PAC (probably approximately correct)** statements:
"With **prob.** $1 - \delta$, the estimated $\hat{\mathcal{R}}$ is an $\epsilon$-**close approximation** of $\mathcal{R}$."

# Back to Machine Learning

# Classical Generalization Bounds

## Finite Hypothesis Set

Setup:

- $\ell(y, \bar{y}) = [\![ y \neq \bar{y} ]\!]$ (0-1 loss)
- finite number of possible classifiers $\mathcal{H} = \{f_1, \dots, f_T\} \subset \mathcal{Y}^{\mathcal{X}}$

For any $\delta > 0$, the following statement holds with probability at least $1 - \delta$ over the training set $\mathcal{D} = \{(x^1, y^1) \dots, (x^n, y^n)\} \overset{i.i.d.}{\sim} p(x, y)$:

For all $f \in \mathcal{H}$:

$$\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \sqrt{\frac{\log |\mathcal{H}| + \log 1/\delta}{2n}}$$

Proof: blackboard...

# Classical Generalization Bounds

**Proof.**

1) For any fixed $f \in \mathcal{H}$, we get from Hoeffding's inequality:

$$\mathbb{P}[\underbrace{\mathcal{R}(f) - \hat{\mathcal{R}}(f) > \epsilon}_{=:C_f}] \leq e^{-2n\epsilon^2}.$$

2) By a union bound, $\mathbb{P}[\bigvee_{f \in \mathcal{H}} C_f] \leq \sum_{f \in \mathcal{H}} \mathbb{P}[C_f]$, we obtain

$$\mathbb{P}[\exists f \in \mathcal{H} : \mathcal{R}(f) > \hat{\mathcal{R}}(f) + \epsilon] \leq |\mathcal{H}| e^{-2n\epsilon^2}.$$

3) Right hand side should be $\delta$, solve for $\epsilon$:

$$\epsilon = \sqrt{\frac{\log(\frac{\mathcal{H}}{\delta})}{2n}}$$

4) Put together, using that

$$\mathbb{P}[\forall f \in \mathcal{H} : \mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \epsilon] = 1 - \mathbb{P}[\exists f \in \mathcal{H} : \mathcal{R}(f) > \hat{\mathcal{R}}(f) + \epsilon]$$

# Examples: Finite hypothesis classes

Model selection:

- Clients offer me trained classifiers: 1) *decision tree*, 2) *LogReg* or an 3) *SVM*? Which of the three should I buy?

Finite precision:

- For $\mathcal{X} \subset \mathbb{R}^d$, the hypothesis set $\mathcal{H} = \{f(x) = \operatorname{sign}\langle w, x\rangle\}$ is infinite.
- But: on a computer with $w$ restricted to 32-bit `floats`: $|\mathcal{H}| = 2^{32d}$. $\log|\mathcal{H}| \approx 22d$

Implementation:

- $\mathcal{H} = \{$ all algorithms implementable in 1 MB C-code $\}$ is finite.

Logarithmic dependence on $|\mathcal{H}|$ makes even large (finite) hypothesis sets (kind of) practical.