Best Practice in Machine Learning for Computer Vision

Christoph H. Lampert
IST Austria (Institute of Science and Technology Austria), Vienna

Vision and Sports Summer School 2017
August 21-26, 2017 — Prague, Czech Republic

Ad: PhD/PostDoc Positions at IST Austria, Vienna

IST Austria Graduate School

- English language program
- 1 + 3 yr PhD program
- full salary

PostDoc positions in my group

- *machine learning*
  - structured output learning
  - transfer learning
- *computer vision*
  - visual scene understanding
- curiosity driven basic research
  - competitive salary,
  - no mandatory teaching, ...

Internships: ask me!

More information: www.ist.ac.at or ask me during a break

Schedule

Machine Learning for Computer Vision

The Basics

Best Practice and How to Avoid Pitfalls

Schedule

Computer Vision

Machine Learning for Computer Vision

The Basics

Best Practice and How to Avoid Pitfalls

Computer Vision

Computer Vision

Robotics (e.g. *autonomous cars*)

Healthcare (e.g. *visual aids*)

Consumer Electronics
(e.g. *human computer interaction*)

Augmented Reality
(e.g. *HoloLens, Pokemon Go*)

Computer vision systems should

- perform interesting/relevant tasks, e.g. drive a car

but mainly they should

- work in the real world,
- be usable by non-experts,
- do what they are supposed to do without failures.

Computer vision systems should

- perform interesting/relevant tasks, e.g. drive a car

but mainly they should

- work in the real world,
- be usable by non-experts,
- do what they are supposed to do without failures.

Machine learning is not particularly good at those...

- we have to be extra careful to know what we're going!

Machine Learning for Computer Vision

Example: Solving a Computer Vision Task using Machine Learning

Step 0) Sanity checks...

Step 1) Decide what exactly you want

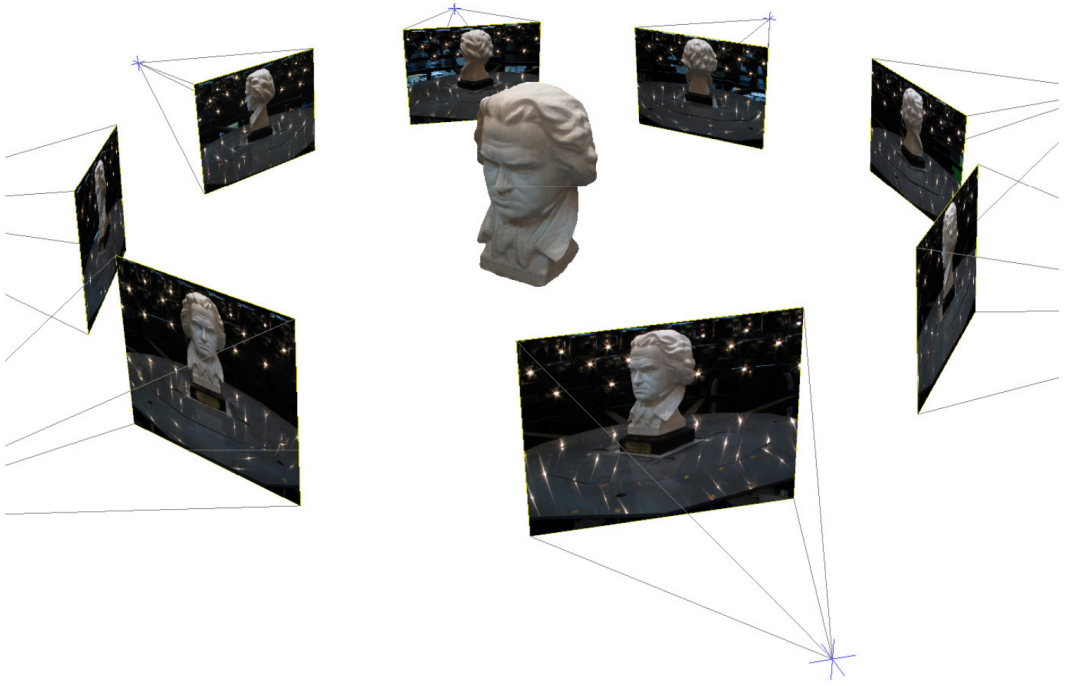
Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

Example Task: 3D Reconstruction

Create a 3D model from many 2D images



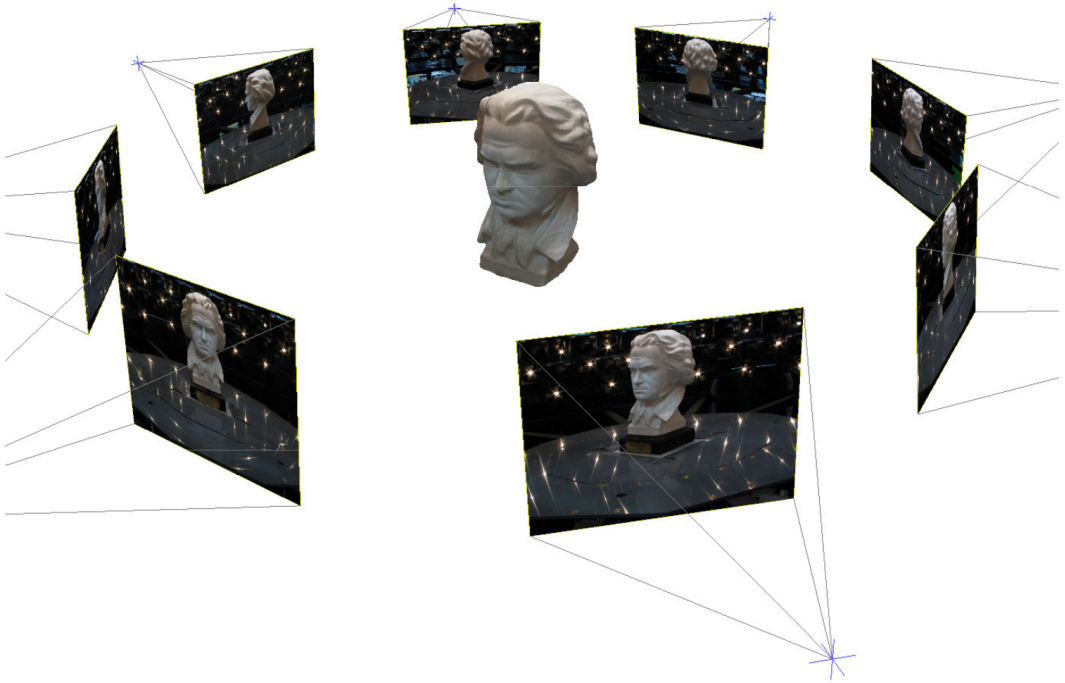
Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem?

  Yes: optimization with geometric constraints!

Example Task: 3D Reconstruction

Create a 3D model from many 2D images



Sanity Check: Is Machine Learning the right way to solve it?

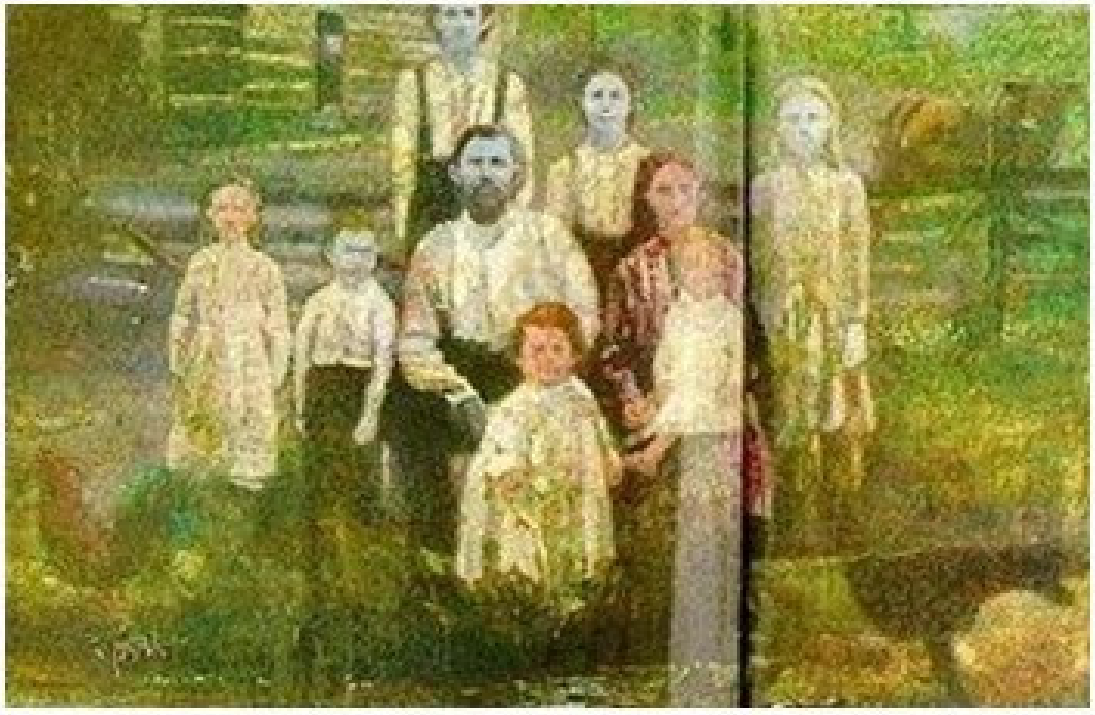
- Can you think of an algorithmic way to solve the problem?

  Yes: optimization with geometric constraints!

→ not a strong case for using machine learning. ✗

images: http://vision.in.tum.de/research/image-based_3d_reconstruction/multiviewreconstruction

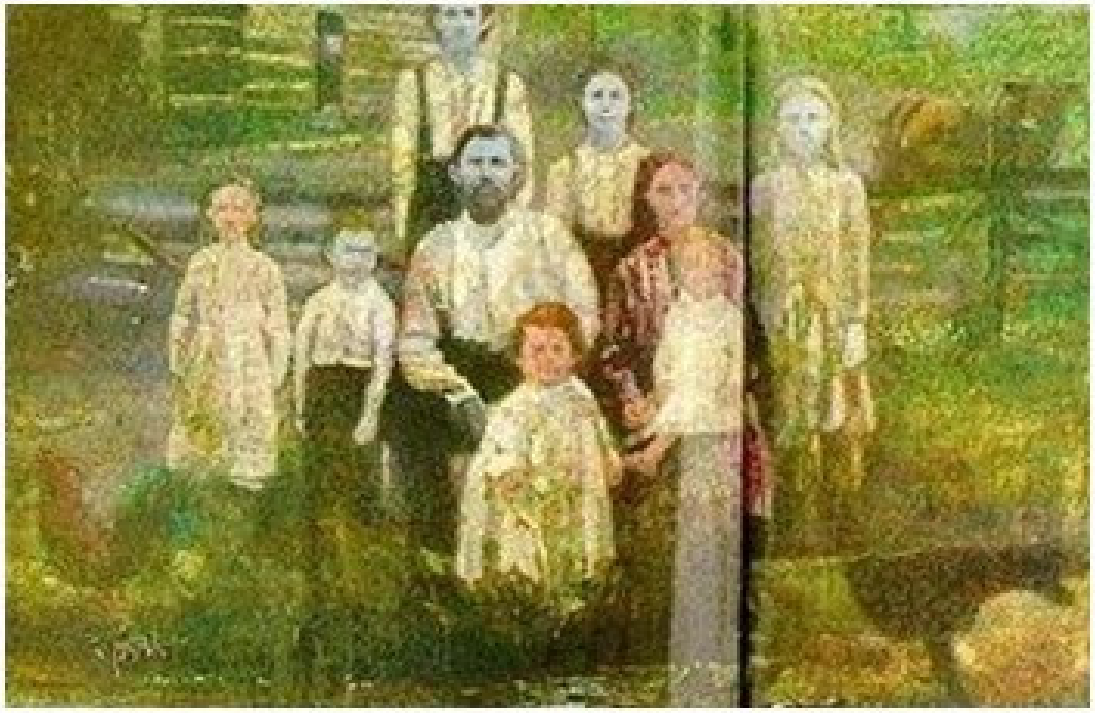
Example Task: Diagnose *methemoglobinemia* (aka *blue skin disorder*)



Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem? **No.**

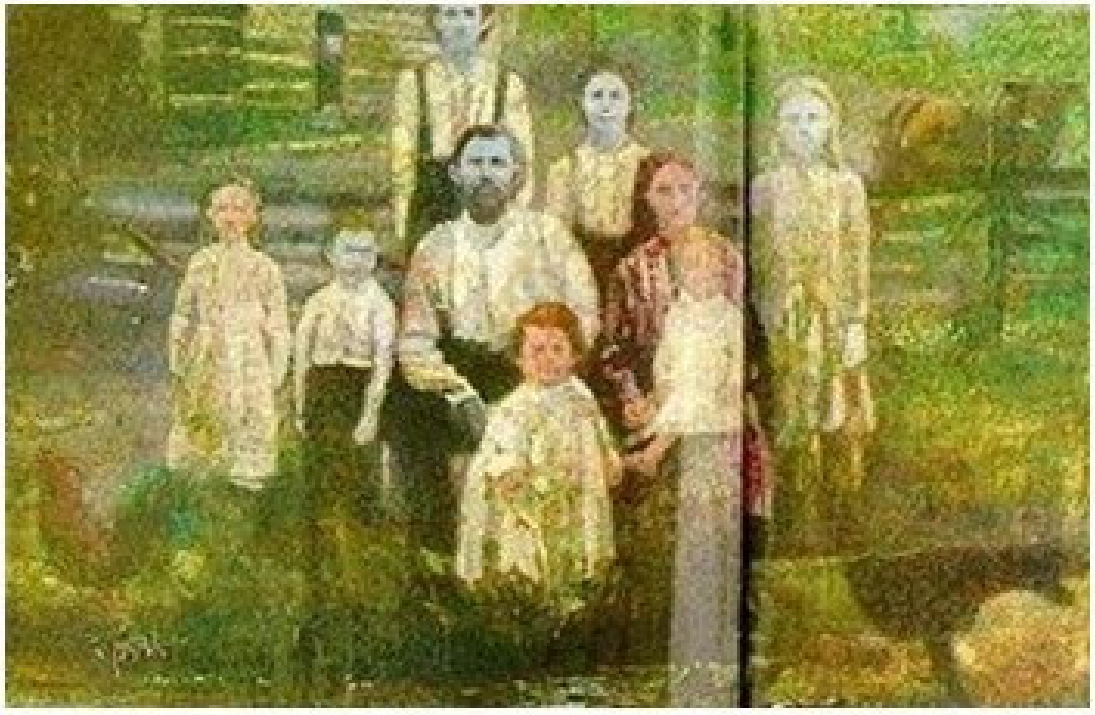
Example Task: Diagnose *methemoglobinemia* (aka *blue skin disorder*)



Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem? **No.**
- It is possible to get data for the problem? **No.**

Example Task: Diagnose *methemoglobinemia* (aka *blue skin disorder*)



Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem? **No.**
- It is possible to get data for the problem? **No.**

→ not a strong case for using machine learning. ✗

image: `http://www.geneticdisorders.info/`

Example Task: Detecting Driver Fatigue

Example Task: Detecting Driver Fatigue

Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem? **No.**

Example Task: Detecting Driver Fatigue

Example Task: Detecting Driver Fatigue

Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem? **No.**
- It is possible to get data for the problem? **Yes.**

Example Task: Detecting Driver Fatigue

Example Task: Detecting Driver Fatigue

Sanity Check: Is Machine Learning the right way to solve it?

- Can you think of an algorithmic way to solve the problem? **No.**
- It is possible to get data for the problem? **Yes.**

$\rightarrow$ machine learning sounds worth trying. ✓

Image: RobeSafe Driver Monitoring Video Dataset (RS-DMV), http://www.robesafe.com/personal/jnuevo/Datasets.html

Example Task: Detecting Driver Fatigue

Step 1) Decide what exactly you want

- **input** x: images (driver of a car)
- **output** $y \in [0, 1]$: e.g. *"how tired does the driver look?"*
  $y = 0$: totally awake, $y = 1$: sound asleep
- **quality measure**: e.g. $\ell(\, y, f(x)\,) = (y - f(x))^2$
- **model** f_θ: e.g. ConvNet with certain topology, e.g. *AlexNet*
  - input layer: fixed size image (scaled version of input x)
  - output layer: single output, value $f_\theta(x) \in [0, 1]$
  - parameters: θ (all weights of all layers)
- **goal**: find parameters θ^* such that model makes good predictions

Example Task: Detecting Driver Fatigue

Step 2) Collect and annotate data

- collect examples: $x_1, x_2, \ldots$
- have an expert annotate them with 'correct' outputs: $y_1, y_2, \ldots$

Step 3) Model training

Take a training set, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, and find θ^* by solving

$$\min_{\theta} J(\theta) \quad \text{with} \quad J(\theta) = \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

(that's the step where you call "SGD" or "Adam" or "RMSProp" etc.)

Step 4) Model evaluation

Take new data (not the training set), $S' = \{(x'_1, y'_1), \ldots, (x'_m, y'_m)\}$, and compute the model performance as

$$\hat{\mathcal{R}}_{\text{tst}} = \frac{1}{m} \sum_{j=1}^{m} \ell(y'_j, f^*(x'_j))$$

for $f^* = f_{\theta^*}$

Example: Solving a Computer Vision Task using Machine Learning

Step 0) Sanity checks...

Step 1) Decide what exactly you want

Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

- **Question 1: why do we do it like this?**
- **Question 2: what can go wrong, and how to avoid that?**

Step 1) Decide what exactly you want

- **input** x: E.g. images of the driver of a car
- **output** $y \in [0, 1]$: E.g. *"how tired does the driver look?"*
  $y = 0$: totally awake, $y = 1$: sound asleep
- **quality measure**: $\ell(\, y, f(x)\,) = (y - f(x))^2$
- **model** f_θ: convolution network with certain topology
- **goal**: find parameters θ^* such that f_{θ^*} makes good predictions

Take care that

- the outputs make sense for the inputs
  - e.g., what about images that don't show a person at all?
- the inputs are informative about the output you're after
  - e.g., frontal pictures? or profile? or back of the head?
- the quality measure makes sense for the task
  - 'fatigue' (real-valued): regression, e.g. $\ell(y, f(x)) = (y - f(x))^2$
  - 'driver identification': classification, e.g. $\ell(y, f(x)) = [\![y \neq f(x)]\!]$
  - 'failure probability', e.g. $\ell(y, f(x)) = y \log f(x) + (1 - y) \log(1 - f(x))$
- the model class makes sense (later...)

Step 2) Collect and annotate data

- **collect data**: images $x_1, x_2, \ldots$
- **annotate data**: have an expert assign 'correct' outputs: $y_1, y_2, \ldots$

Take care that

- the data reflects the situation of interest well
  - same conditions (resolution, perspective, lighting) as in actual car, ...
- you collect and annotate enough data
  - the more the better
- the annotation is of high quality (i.e. exactly what you want)
  - 'fatigue' might be subjective
  - how tired is '0.7' compared to '0.6' ?
  - define common standards if multiple annotators are involved

... will be made more precise later...

Step 3) Model training

- **take a training set**, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$,
- **solve the following optimization problem** to find θ^*

$$\min_{\theta} J(\theta) \quad \text{with} \quad J(\theta) = \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

Step 3) Model training

- **take a training set**, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$,
- **solve the following optimization problem** to find θ^*

$$\min_{\theta} J(\theta) \quad \text{with} \quad J(\theta) = \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

What to take care of? That's a bit of a longer story... we'll need:

- Refresher of probabilities
- Empirical risk minimization
- Overfitting / underfitting
- Regularization
- Model selection

Refresher of probabilities

Most quantities in computer vision are not fully deterministic.

- true randomness of events
  - a photon reaches a camera's CCD chip, is it detected or not?
    it depends on quantum effects, which -to our knowledge- are stochastic
- incomplete knowledge
  - what will be the next picture I take with my smartphone?
  - who will be in my floorball group this afternoon?
- insufficient representation
  - what material corresponds to that green pixel in the image?
    with RGB impossible to tell, with hyperspectral maybe possible

In practice, there is no difference between these!

Probability theory allows us to deal with this.

Random variables randomly take one of multiple possible values:

- number of photons reaching a CCD chip
- next picture I will take with my smartphone
- names of all people in my floorball group tomorrow

Notation:

- random variables: capital letters, e.g. X
- set of possible values: curly letters, e.g. $\mathcal{X}$ (for simplicity: discrete)
- individual values: lowercase letters, e.g. x

Likelihood of each value $x \in \mathcal{X}$ is specified by a **probability distribution**:

- $p(X = x)$ is the probability that X takes the value $x \in \mathcal{X}$.
  (or just $p(x)$ if the context is clear).
- for example, rolling a die, $p(X = 3) = p(3) = 1/6$
- we write $x \sim p(x)$ to indicate that the distribution of X is $p(x)$

Properties of probabilities

Elementary rules of probability distributions:

$$0 \leq p(x) \leq 1 \quad \text{for all } x \in \mathcal{X} \qquad \text{(positivity)}$$

$$\sum_{x \in \mathcal{X}} p(x) = 1 \qquad \text{(normalization)}$$

If X has only two possible values, e.g. $\mathcal{X} = \{\texttt{true}, \texttt{false}\}$,

$$p(X = \texttt{false}) = 1 - p(X = \texttt{true})$$

Example: *PASCAL VOC2006* dataset

Define random variables

- X_{obj}: does a randomly picked image contain an object *"obj"*?
- $\mathcal{X}_{obj} = \{\texttt{true}, \texttt{false}\}$

$$p(X_{person} = \texttt{true}) = 0.254 \qquad p(X_{person} = \texttt{false}) = 0.746$$

$$p(X_{horse} = \texttt{true}) = 0.094 \qquad p(X_{horse} = \texttt{false}) = 0.916$$

Joint probabilities

Probabilities can be assigned to more than one random variable at a time:

- $p(X = x, Y = y)$ is the probability that $X = x$ and $Y = y$ (at the same time)

joint probability

Example: *PASCAL VOC2006* dataset

- $p(X_{person} = \texttt{true}, X_{horse} = \texttt{true}) = 0.050$
- $p(X_{dog} = \texttt{true}, X_{person} = \texttt{true}, X_{cat} = \texttt{false}) = 0.014$
- $p(X_{aeroplane} = \texttt{true}, X_{aeroplane} = \texttt{false}) = 0$

Marginalization

We can recover the probabilities of individual variables from the joint probability by summing over all variables we are not interested in.

- $p(X = x) = \sum_{y \in \mathcal{Y}} p(X = x, Y = y)$
- $p(X_2 = z) = \sum_{x_1 \in \mathcal{X}_1} \sum_{x_3 \in \mathcal{X}_3} \sum_{x_4 \in \mathcal{X}_4} p(X_1 = x_1, X_2 = z, X_3 = x_3, X_4 = x_4)$

marginalization

Marginalization

We can recover the probabilities of individual variables from the joint probability by summing over all variables we are not interested in.

- $p(X = x) = \sum_{y \in \mathcal{Y}} p(X = x, Y = y)$
- $p(X_2 = z) = \sum_{x_1 \in \mathcal{X}_1} \sum_{x_3 \in \mathcal{X}_3} \sum_{x_4 \in \mathcal{X}_4} p(X_1 = x_1, X_2 = z, X_3 = x_3, X_4 = x_4)$

marginalization

Example: *PASCAL VOC2006* dataset

- $p(X_{person} = \texttt{true}, X_{horse} = \texttt{true}) = 0.050$
- $p(X_{person} = \texttt{true}, X_{horse} = \texttt{false}) = 0.204$
- $p(X_{person} = \texttt{false}, X_{horse} = \texttt{true}) = 0.044$
- $p(X_{person} = \texttt{false}, X_{horse} = \texttt{false}) = 0.702$

	horse	no horse	
person	0.050	0.204	
no person	0.044	0.702	

Marginalization

We can recover the probabilities of individual variables from the joint probability by summing over all variables we are not interested in.

- $p(X = x) = \sum_{y \in \mathcal{Y}} p(X = x, Y = y)$
- $p(X_2 = z) = \sum_{x_1 \in \mathcal{X}_1} \sum_{x_3 \in \mathcal{X}_3} \sum_{x_4 \in \mathcal{X}_4} p(X_1 = x_1, X_2 = z, X_3 = x_3, X_4 = x_4)$

marginalization

Example: *PASCAL VOC2006* dataset

- $p(X_{person} = \texttt{true}, X_{horse} = \texttt{true}) = 0.050$
- $p(X_{person} = \texttt{true}, X_{horse} = \texttt{false}) = 0.204$
- $p(X_{person} = \texttt{false}, X_{horse} = \texttt{true}) = 0.044$
- $p(X_{person} = \texttt{false}, X_{horse} = \texttt{false}) = 0.702$

	horse	no horse	$\sum$
person	0.050	0.204	0.254
no person	0.044	0.702	0.746
$\sum$	0.094	0.906	

- $p(X_{person} = \texttt{true}) = 0.050 + 0.204 = 0.254$
- $p(X_{horse} = \texttt{false}) = 0.204 + 0.702 = 0.906$

Conditional probabilities

One random variable can contain information about another one:

- $p(X = x \,|\, Y = y)$: **conditional probability**

  what is the probability of $X = x$, if we already know that $Y = y$?

- conditional probabilities can be computed from joint and marginal:

$$p(X = x|Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)} \quad \text{(don't do this if } p(Y = y) = 0\text{)}$$

  useful equivalence: $p(X = x, Y = y) = p(X = x|Y = y)p(Y = y)$

Conditional probabilities

One random variable can contain information about another one:

- $p(X = x \mid Y = y)$: **conditional probability**

  what is the probability of $X = x$, if we already know that $Y = y$?

- conditional probabilities can be computed from joint and marginal:

$$p(X = x|Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)} \quad \text{(don't do this if } p(Y = y) = 0\text{)}$$

  useful equivalence: $p(X = x, Y = y) = p(X = x|Y = y)p(Y = y)$

Example: *PASCAL VOC2006* dataset

- $p(X_{person} = \texttt{true}) = 0.254$
- $p(X_{person} = \texttt{true}|X_{horse} = \texttt{true}) = \frac{0.050}{0.094} = 0.534$
- $p(X_{dog} = \texttt{true}) = 0.139$
- $p(X_{dog} = \texttt{true}|X_{cat} = \texttt{true}) = \frac{0.002}{0.147} = 0.016$
- $p(X_{dog} = \texttt{true}|X_{cat} = \texttt{false}) = \frac{0.137}{0.853} = 0.161$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$
- $p(X_1 = 1, X_2 = 0) = 0.4 \quad p(X_1 = 1, X_2 = 1) = 0.6,$
  $p(X_1 = 2, X_2 = 0) = 0.2 \quad p(X_1 = 2, X_2 = 1) = 0.8,$
  $p(X_1 = 3, X_2 = 0) = 0.5 \quad p(X_1 = 3, X_2 = 1) = 0.5$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$
- $p(X_1 = 1, X_2 = 0) = 0.4 \quad p(X_1 = 1, X_2 = 1) = 0.6,$
  $p(X_1 = 2, X_2 = 0) = 0.2 \quad p(X_1 = 2, X_2 = 1) = 0.8,$
  $p(X_1 = 3, X_2 = 0) = 0.5 \quad p(X_1 = 3, X_2 = 1) = 0.5$

True or false?

- $p(X = x, Y = y) \leq p(X = x)$ and $p(X = x, Y = y) \leq p(Y = y)$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$
- $p(X_1 = 1, X_2 = 0) = 0.4 \quad p(X_1 = 1, X_2 = 1) = 0.6,$
  $p(X_1 = 2, X_2 = 0) = 0.2 \quad p(X_1 = 2, X_2 = 1) = 0.8,$
  $p(X_1 = 3, X_2 = 0) = 0.5 \quad p(X_1 = 3, X_2 = 1) = 0.5$

True or false?

- $p(X = x, Y = y) \leq p(X = x)$ and $p(X = x, Y = y) \leq p(Y = y)$
- $p(X = x|Y = y) \geq p(X = x)$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$
- $p(X_1 = 1, X_2 = 0) = 0.4 \quad p(X_1 = 1, X_2 = 1) = 0.6,$
  $p(X_1 = 2, X_2 = 0) = 0.2 \quad p(X_1 = 2, X_2 = 1) = 0.8,$
  $p(X_1 = 3, X_2 = 0) = 0.5 \quad p(X_1 = 3, X_2 = 1) = 0.5$

True or false?

- $p(X = x, Y = y) \leq p(X = x)$ and $p(X = x, Y = y) \leq p(Y = y)$
- $p(X = x | Y = y) \geq p(X = x)$
- $p(X = x, Y = y) \leq p(X = x | Y = y)$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$
- $p(X_1 = 1, X_2 = 0) = 0.4 \quad p(X_1 = 1, X_2 = 1) = 0.6,$
  $p(X_1 = 2, X_2 = 0) = 0.2 \quad p(X_1 = 2, X_2 = 1) = 0.8,$
  $p(X_1 = 3, X_2 = 0) = 0.5 \quad p(X_1 = 3, X_2 = 1) = 0.5$

True or false?

- $p(X = x, Y = y) \leq p(X = x)$ and $p(X = x, Y = y) \leq p(Y = y)$
- $p(X = x | Y = y) \geq p(X = x)$
- $p(X = x, Y = y) \leq p(X = x | Y = y)$

Quiz: Probabilities

X_1, X_2 random variables with $\mathcal{X}_1 = \{1, 2, 3\}$ and $\mathcal{X}_2 = \{0, 1\}$

What's wrong here?

- $p(X_1 = 1) = 1 \quad p(X_1 = 2) = 0 \quad p(X_1 = 3) = -1$
- $p(X_1 = 1) = 0.1 \quad p(X_1 = 2) = 0.2 \quad p(X_1 = 3) = 0.3$
- $p(X_1 = 1, X_2 = 0) = 0.4 \quad p(X_1 = 1, X_2 = 1) = 0.6,$
  $p(X_1 = 2, X_2 = 0) = 0.2 \quad p(X_1 = 2, X_2 = 1) = 0.8,$
  $p(X_1 = 3, X_2 = 0) = 0.5 \quad p(X_1 = 3, X_2 = 1) = 0.5$

True or false?

- $p(X = x, Y = y) \leq p(X = x)$ and $p(X = x, Y = y) \leq p(Y = y)$
- $p(X = x | Y = y) \geq p(X = x)$
- $\underbrace{p(X = x, Y = y)}_{= p(X=x|Y=y)p(Y=y)} \leq p(X = x | Y = y)$

Dependence/Independence

Not every random variable is informative about every other.

- We say X **is independent of** Y if

  $$P(X = x, Y = y) = P(X = x)P(Y = y) \qquad \text{for all } x \in \mathcal{X} \text{ and } y \in \mathcal{Y}$$

- equivalent (if defined):

  $$P(X = x|Y = y) = P(X = x), \qquad P(Y = y|X = x) = P(Y = y)$$

Dependence/Independence

Not every random variable is informative about every other.

- We say X **is independent of** Y if

  $P(X = x, Y = y) = P(X = x)P(Y = y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$

- equivalent (if defined):

  $P(X = x|Y = y) = P(X = x), \quad P(Y = y|X = x) = P(Y = y)$

Example: Image datasets

- X_1: pick a random image from VOC2006. Does it show a `cat`?
- X_2: again pick a random image from VOC2006. Does it show a `cat`?
- $p(X_1 = \texttt{true}, X_2 = \texttt{true}) = p(X_1 = \texttt{true})p(X_2 = \texttt{true})$

Example: Video

- Y_1: does the first frame of a video show a `cat`?
- Y_2: does the second image of video show a `cat`?
- $p(Y_1 = \texttt{true}, Y_2 = \texttt{true}) \gg p(Y_1 = \texttt{true})p(Y_2 = \texttt{true})$

Expected value

We apply a function to (the values of) one or more random variables:

- $f(x) = \sqrt{x}$ or $f(x_1, x_2, \ldots, x_k) = \dfrac{x_1 + x_2 + \cdots + x_k}{k}$

The **expected value** or **expectation** of a function f with respect to a probability distribution is the weighted average of the possible values:

$$\mathbb{E}_{x \sim p(x)}[f(x)] := \sum_{x \in \mathcal{X}} p(x) f(x)$$

In short, we just write $\mathbb{E}_x[f(x)]$ or $\mathbb{E}[f(x)]$ or $\mathbb{E}[f]$ or $\mathbb{E}f$.

Example: rolling dice

Let X be the outcome of rolling a die and let $f(x) = x$

$$\mathbb{E}_{x \sim p(x)}[f(x)] = \mathbb{E}_{x \sim p(x)}[x] = \frac{1}{6}1 + \frac{1}{6}2 + \frac{1}{6}3 + \frac{1}{6}4 + \frac{1}{6}5 + \frac{1}{6}6 = 3.5$$

Properties of expected values

Example: rolling dice

X_1, X_2: the outcome of rolling two dice independently, $f(x, y) = x + y$

$\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] =$

Properties of expected values

Example: rolling dice

X_1, X_2: the outcome of rolling two dice independently, $f(x, y) = x + y$

$\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] =$

The expected value has a useful property: 1) it is *linear* in its argument.

- $\mathbb{E}_{x\sim p(x)}[f(x) + g(x)] = \mathbb{E}_{x\sim p(x)}[f(x)] + \mathbb{E}_{x\sim p(x)}[g(x)]$
- $\mathbb{E}_{x\sim p(x)}[\lambda f(x)] = \lambda \mathbb{E}_{x\sim p(x)}[f(x)]$

2) If a random variables does not show up in a function, we can ignore the expectation operation with respect to it

- $\mathbb{E}_{(x,y)\sim p(x,y)}[f(x)] = \mathbb{E}_{x\sim p(x)}[f(x)]$

Properties of expected values

Example: rolling dice

X_1, X_2: the outcome of rolling two dice independently, $f(x, y) = x + y$

$$\begin{aligned}\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] &= \mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[x_1 + x_2] \\ &= \mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[x_1] + \mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[x_2] \\ &= \mathbb{E}_{x_1\sim p(x_1)}[x_1] + \mathbb{E}_{x_2\sim p(x_2)}[x_2] = 3.5+3.5 = \mathbf{7}\end{aligned}$$

The expected value has a useful property: 1) it is *linear* in its argument.

- $\mathbb{E}_{x\sim p(x)}[f(x) + g(x)] = \mathbb{E}_{x\sim p(x)}[f(x)] + \mathbb{E}_{x\sim p(x)}[g(x)]$
- $\mathbb{E}_{x\sim p(x)}[\lambda f(x)] = \lambda \mathbb{E}_{x\sim p(x)}[f(x)]$

2) If a random variables does not show up in a function, we can ignore the expectation operation with respect to it

- $\mathbb{E}_{(x,y)\sim p(x,y)}[f(x)] = \mathbb{E}_{x\sim p(x)}[f(x)]$

1) and 2) hold for any random variables, not just independent ones!

Example: rolling dice

- we roll one die
- X_1: number facing up, X_2: number facing down
- $f(x_1, x_2) = x_1 + x_2$

$$\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] =$$

Example: rolling dice

- we roll one die
- X_1: number facing up, X_2: number facing down
- $f(x_1, x_2) = x_1 + x_2$

$$\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] = \mathbf{7}$$

Answer 1: explicit calculation with dependent X_1 and X_2

$$p(x_1, x_2) = \begin{cases} \frac{1}{6} & \text{for combinations } (1,6),\ (2,5),\ (3,4),\ (4,3),\ (5,2),\ (6,1) \\ 0 & \text{for all other combinations.} \end{cases}$$

$$\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] = \sum_{(x_1,x_2)} p(x_1, x_2)(x_1 + x_2)$$

$$= 0(1+1) + 0(1+2) + \cdots + \frac{1}{6}(1+6) + 0(2+1) + \cdots = 6 \cdot \frac{7}{6} = 7$$

Example: rolling dice

- we roll one die
- X_1: number facing up, X_2: number facing down
- $f(x_1, x_2) = x_1 + x_2$

$$\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] = \mathbf{7}$$

Answer 2: use properties of expectation as earlier

$$\begin{aligned}\mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[f(x_1, x_2)] &= \mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[x_1 + x_2] \\ &= \mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[x_1] + \mathbb{E}_{(x_1,x_2)\sim p(x_1,x_2)}[x_2] \\ &= \mathbb{E}_{x_1\sim p(x_1)}[x_1] + \mathbb{E}_{x_2\sim p(x_2)}[x_2] = 3.5 + 3.5 = 7\end{aligned}$$

The rules of probability take care of dependence, etc.

back to learning...

Empirical risk minimization

What do we (really, really) want?

A model f_θ that works well (=has small loss) when we apply it to future data.

Problem: we don't know what the future will bring!

What do we (really, really) want?

A model f_θ that works well (=has small loss) when we apply it to future data.

Problem: we don't know what the future will bring!

Probabilities to the rescue:

- we are **uncertain** about future data $\rightarrow$ use a **random variable** X
- $\mathcal{X}$: all possible images, $p(x)$ probability to see any $x \in \mathcal{X}$
- assume: for every input $x \in \mathcal{X}$, there's a correct output $y_x^{\text{gt}} \in \mathcal{Y}$

  also possible: multiple correct y are possible with conditional probabilities $p(y|x)$

Note: we don't pretend that we know p or y^{gt}, we just assume they exist.

What do we (really, really) want?

A model f_θ that works well (=has small loss) when we apply it to future data.

Problem: we don't know what the future will bring!

Probabilities to the rescue:

- we are **uncertain** about future data $\rightarrow$ use a **random variable** X
- $\mathcal{X}$: all possible images, $p(x)$ probability to see any $x \in \mathcal{X}$
- assume: for every input $x \in \mathcal{X}$, there's a correct output $y_x^{\text{gt}} \in \mathcal{Y}$

  also possible: multiple correct y are possible with conditional probabilities $p(y|x)$

Note: we don't pretend that we know p or y^{gt}, we just assume they exist.

What do we want formally?

A model f_θ with small expected loss (aka *"risk"* or *"test error"*)

$$\mathcal{R} = \mathbb{E}_{x \sim p(x)}[\, \ell(\, y_x^{\text{gt}}, f_\theta(x))\,]$$

What do we want formally?

A model f_θ with small risk, $\mathcal{R} = \mathbb{E}_{x \sim p(x)}[\, \ell(\, y_x^{\mathsf{gt}}, f_\theta(x))\,]$.

New problem: we don't know p, so we can't compute $\mathcal{R}$

What do we want formally?

A model f_θ with small risk, $\mathcal{R} = \mathbb{E}_{x \sim p(x)}[\, \ell(\, y_x^{\text{gt}}, f_\theta(x))\,]$.

New problem: we don't know p, so we can't compute $\mathcal{R}$

But we can *estimate* it!

Empirical risk

Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of input-output pairs. Then

$$\hat{\mathcal{R}} = \frac{1}{n} \sum_{i=1}^{n} \ell(\, y_i, f_\theta(x_i))$$

is called the *empirical risk* of $\mathcal{R}$ w.r.t. S. (also called training error)

We would like to use $\hat{\mathcal{R}}$ as a drop-in replacement for $\mathcal{R}$. **Under which conditions is this a good idea?**

Estimating an unknown value from data

We look at $\hat{\mathcal{R}} = \frac{1}{n}\sum_i \ell(y_i, f(x_i))$ as an **estimator** of $\mathcal{R} = \mathbb{E}_x \ell(y_x^{gt}, f(x))$

Estimators

An estimator is a rule for calculating an estimate, $\hat{E}(S)$, of a quantity E based on observed data, S. If S is random, then $\hat{E}(S)$ is also random.

Properties of estimators: unbiasedness

We can compute the expected value of the estimate, $\mathbb{E}_S[\hat{E}(S)]$.

- if $\mathbb{E}_S[\hat{E}(S)] = E$, we call the estimator unbiased.
  We can think of $\hat{E}$ as a noisy version of E then.
- $\text{bias}(\hat{E}) = \mathbb{E}_S[\hat{E}(S)] - E$

Properties of estimators: variance

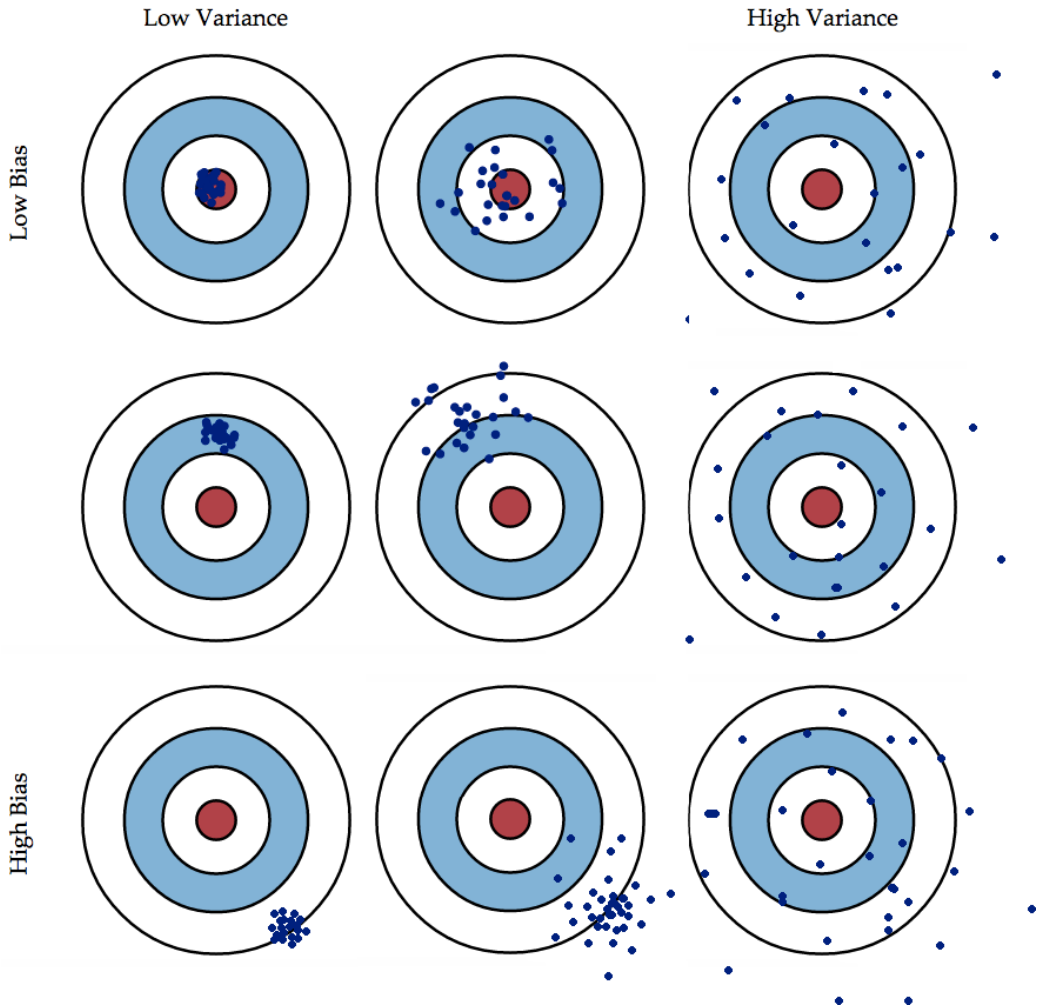
How far is one estimate from the expected value? $(\hat{E}(S) - \mathbb{E}_S[\hat{E}(S)])^2$

- $\text{Var}(\hat{E}) = \mathbb{E}_S[\big(\hat{E}(S) - \mathbb{E}_S[\hat{E}(S)]\big)^2]$

If $\text{Var}(\hat{E})$ is large, then the estimate fluctuates a lot for different S.

Bias-Variance Trade-Off

It's good to have small or no bias, and it's good to have small variance.



If you can't have both at the same time, look for a reasonable trade-off.

Image: adapted from http://scott.fortmann-roe.com/docs/BiasVariance.html

Is $\hat{\mathcal{R}} = \frac{1}{n} \sum_{i=1}^{n} \ell(\, y_i, f_\theta(x_i))$ a good estimator of $\mathcal{R}$?

Is $\hat{\mathcal{R}} = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$ a good estimator of $\mathcal{R}$?

That depends on data we use:

Independent and identically distributed (i.i.d.) training data

If the samples $x_1, \ldots, x_n$ are sampled independently from the distribution $p(x)$ and the outputs $y_1, \ldots, y_n$ are the ground truth ones ($y_i = y^{\text{gt}}_{x_i}$), then $\hat{\mathcal{R}}$ is an unbiased and consistent estimator of $\mathcal{R}$:

$$\mathbb{E}_{x_1,\ldots,x_n}[\hat{\mathcal{R}}] = \mathcal{R} \qquad \text{and} \qquad \mathrm{Var}(\hat{\mathcal{R}}) \to 0 \ \text{ with speed } O(\tfrac{1}{n})$$

(follows from the law of large numbers)

What if the samples $x_1, \ldots, x_n$ are not independent (e.g. video frames)?

- $\hat{\mathcal{R}}$ is unbiased, but $\mathrm{Var}[\hat{\mathcal{R}}]$ will converge slower to 0

What if the distribution of the samples $x_1, \ldots, x_n$ is different from p?
What if the outputs are not the ground truth ones?

- $\hat{\mathcal{R}}$ might be very different from $\mathcal{R}$ $\quad (\rightarrow$ "domain adaptation")

Step 2) Collect and annotate data

- collect examples: $x_1, x_2, \ldots$
- have an expert annotate them with 'correct' outputs: $y_1, y_2, \ldots$

Take care that:

- data comes from the real data distribution
- examples are independent
- output annotation is a good as possible

Step 2) Collect and annotate data

- collect examples: $x_1, x_2, \ldots$
- have an expert annotate them with 'correct' outputs: $y_1, y_2, \ldots$

Take care that:

- data comes from the real data distribution
- examples are independent
- output annotation is a good as possible

Step 3) Model training

Take a training set, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, and find θ^* by solving

$$\min_\theta J(\theta) \quad \text{with} \quad J(\theta) = \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

Observation: $J(\theta)$ is the *empirical risk* (up to an irrelevant factor $\frac{1}{n}$)

Learning by **Empirical Risk Minimization**

How to solve the optimization problem in practice?

- Best option: rely on existing packages
  - deep networks: `tensorflow`, `pytorch`, `caffe`, `theano`, `MatConvNet`, ...
  - support vector machines: `libSVM`, `SVMlight`, `liblinear`, ...
  - generic ML framework: `Weka`, `scikit-learn`, ...

Example: regression in `scikit-learn`

```
import sklearn                                          # load scikit-learn package

data_trn, labels_trn = ..., ...                         # obtain your data somehow

model = sklearn.linear_model.LinearRegression()         # create linear regression model

model.fit(data_trn, labels_trn)                         # train model on the training set

data_new = ...                                          # obtain new data

predictions = model.predict(data_new)                   # predict values for new data
```

(more in exercises and other lectures...)

- second best option: write your own optimizer, e.g.

Gradient descent optimization

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

- required gradients might be computable automatically
  - automatic differentiation (`autodiff`)
  - backpropagation
- if loss function is not convex/differentiable
  - use surrogate loss $\tilde{\ell}$ instead of real loss, e.g.

$$\tilde{\ell}(y, f(x)) = \max\{0, yf(x)\} \text{ instead of } \ell(y, f(x)) = [\![y \neq \operatorname{sign} f(x)]\!]$$

- why second best? easy to make mistakes or be inefficient...

Summary

Learning is about finding a model that works on future data.

The true quantity of interest is the expected error on future data:

$$\mathcal{R} = \mathbb{E}_{x \sim p(x)} \ell(\, y_x^{gt}, f(x)) \qquad \text{(test error)}$$

We cannot compute $\mathcal{R}$, but we can estimate it.

For a training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the training error is

$$\hat{\mathcal{R}} = \sum_{i=1}^{n} \ell(\, y_i, f(x_i))$$

Training data should be i.i.d.

If the training examples are sampled independently and all from the distribution $p(x)$ then $\hat{\mathcal{R}}$ is an unbiased estimate of $\mathcal{R}$.

Learning $\equiv$ empirical risk minimization

The core of most learning methods is to minimize the training error.

Overfitting / underfitting

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

Q: Will it work well on future data, i.e. have small generalization error, $\mathcal{R}$?

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

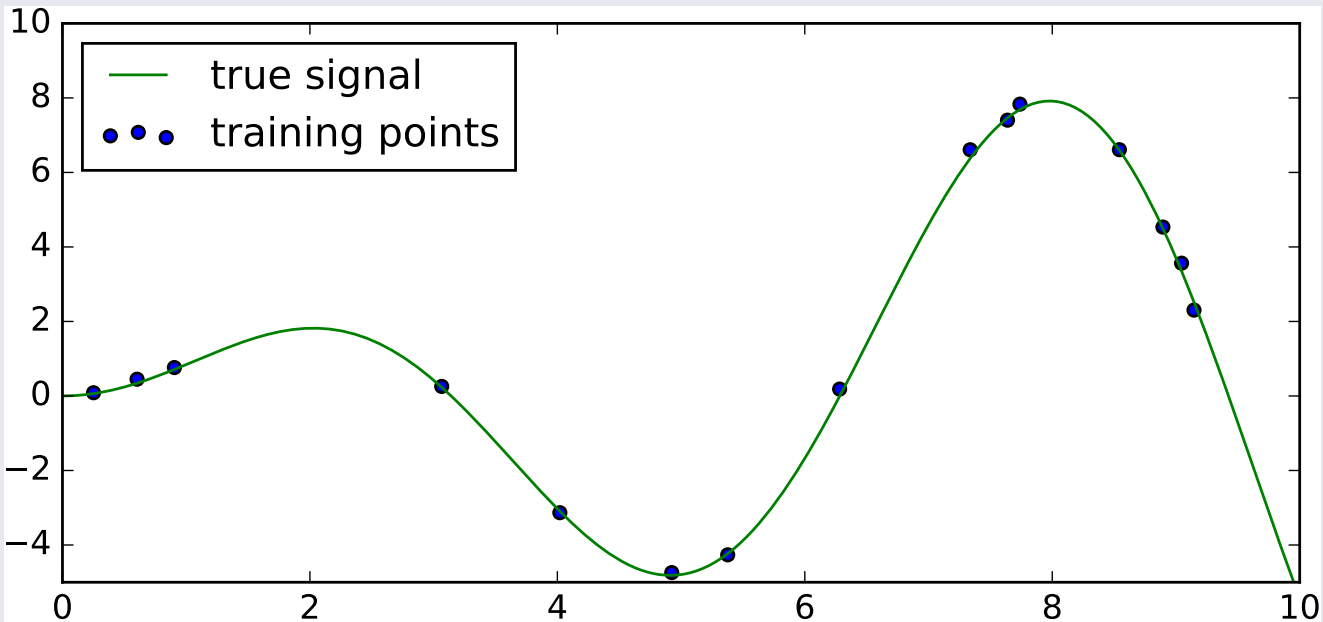
We found a model f_{θ^*} by minimizing the training error $\hat{\mathcal{R}}$.

Q: Will it work well on future data, i.e. have small generalization error, $\mathcal{R}$?

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

Relation between training error and generalization error

Example: 1D curve fitting



training points

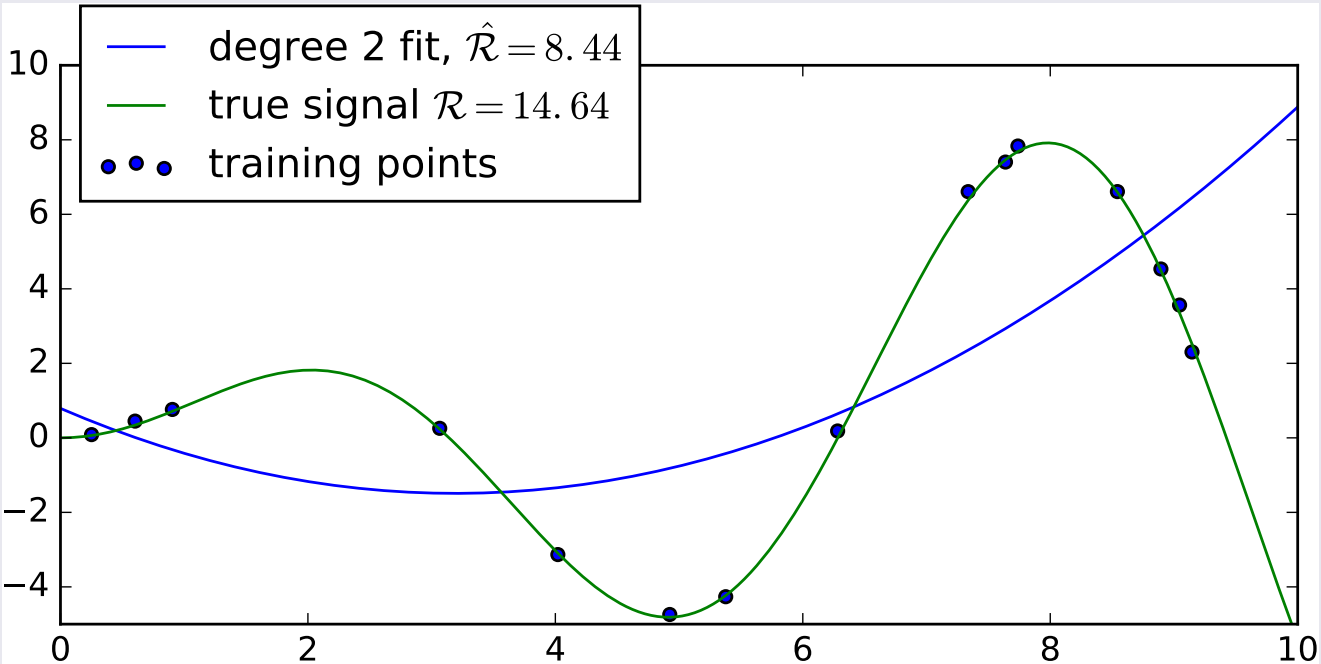
We found a model f_{θ^*} by minimizing the training error $\hat{\mathcal{R}}$.

Q: Will it work well on future data, i.e. have small generalization error, $\mathcal{R}$?

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

Relation between training error and generalization error

Example: 1D curve fitting



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

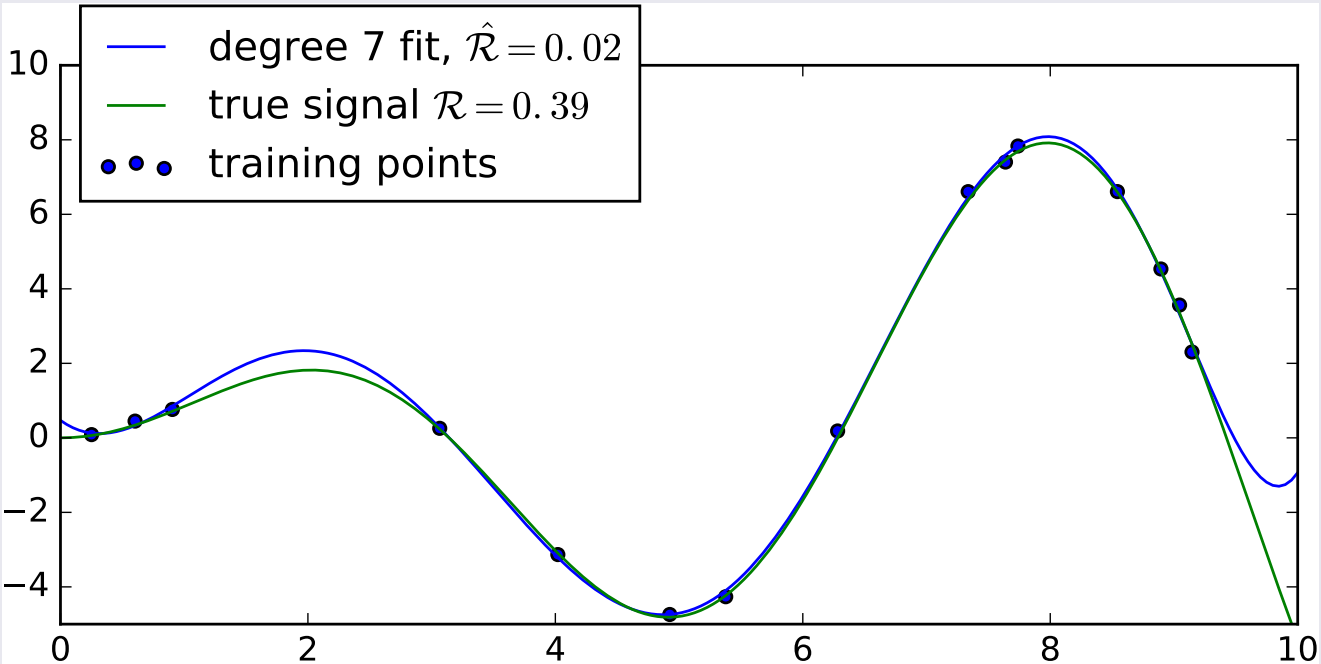
We found a model f_{θ^*} by minimizing the training error $\hat{\mathcal{R}}$.

Q: Will it work well on future data, i.e. have small generalization error, $\mathcal{R}$?

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

Relation between training error and generalization error

Example: 1D curve fitting



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

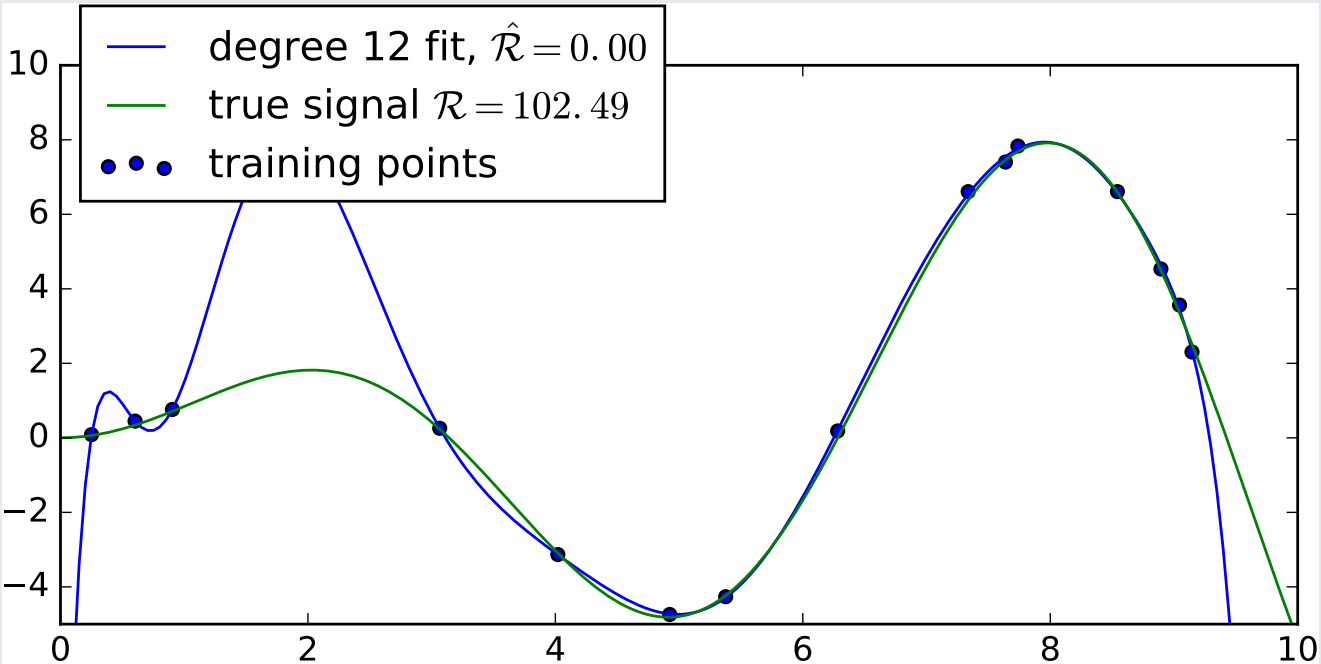
We found a model f_{θ^*} by minimizing the training error $\hat{\mathcal{R}}$.

Q: Will it work well on future data, i.e. have small generalization error, $\mathcal{R}$?

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

Relation between training error and generalization error

Example: 1D curve fitting



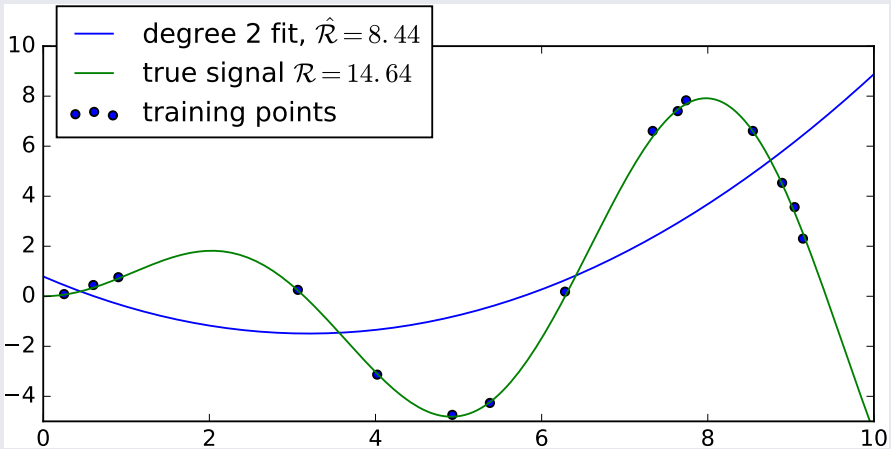
best learned polynomial of degree 12: small $\hat{\mathcal{R}}$, large $\mathcal{R}$

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

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

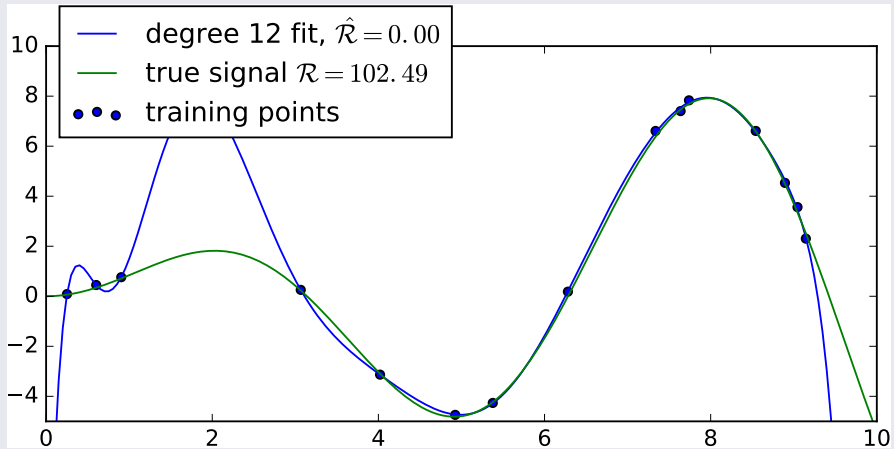
A: **Unfortunately, that is not guaranteed.**

Underfitting/Overfitting



Underfitting

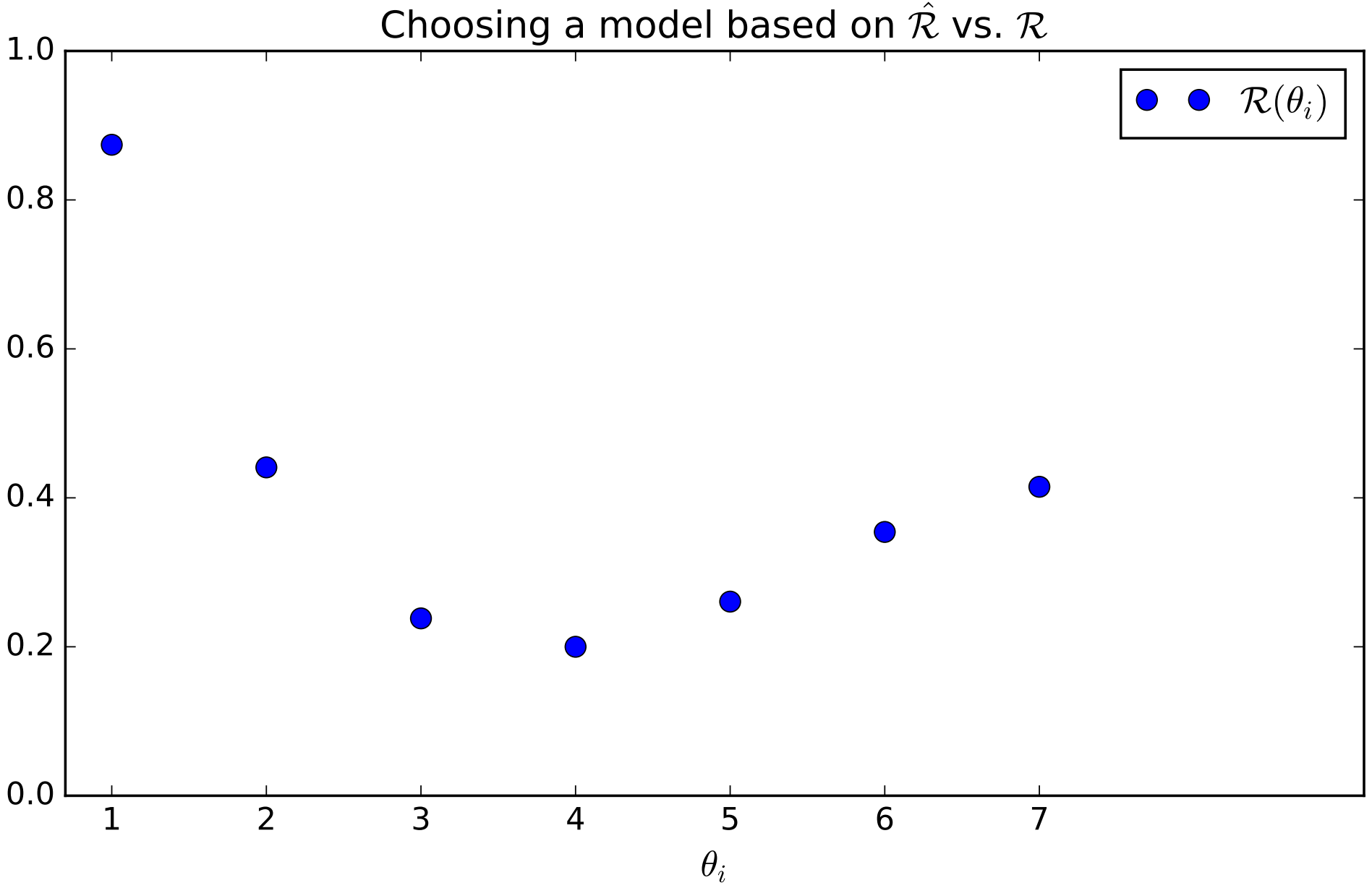
(to some extent) detectable from $\hat{\mathcal{R}}$



Overfitting

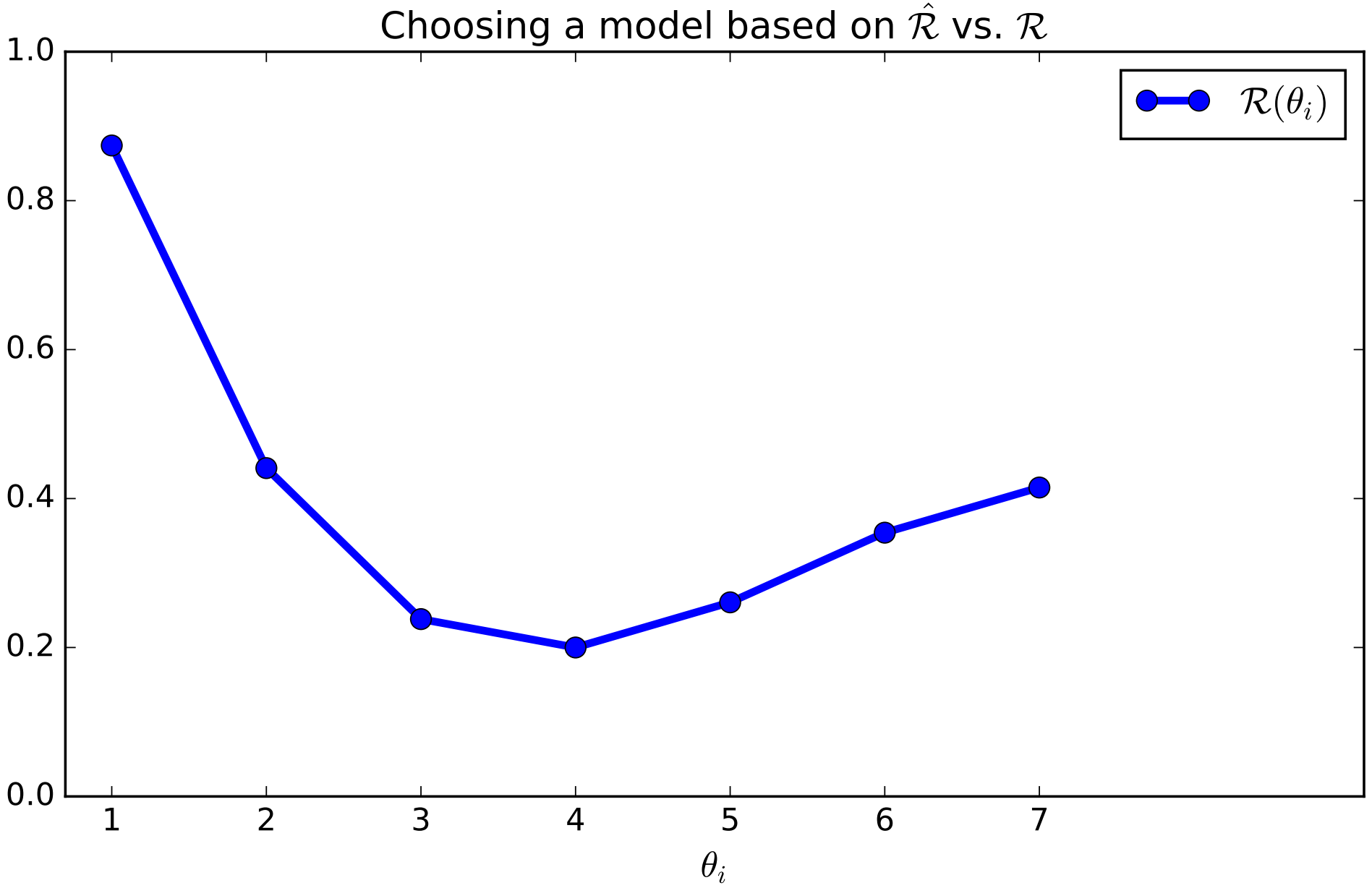
not detectable from $\hat{\mathcal{R}}$!

Where does overfitting come from?



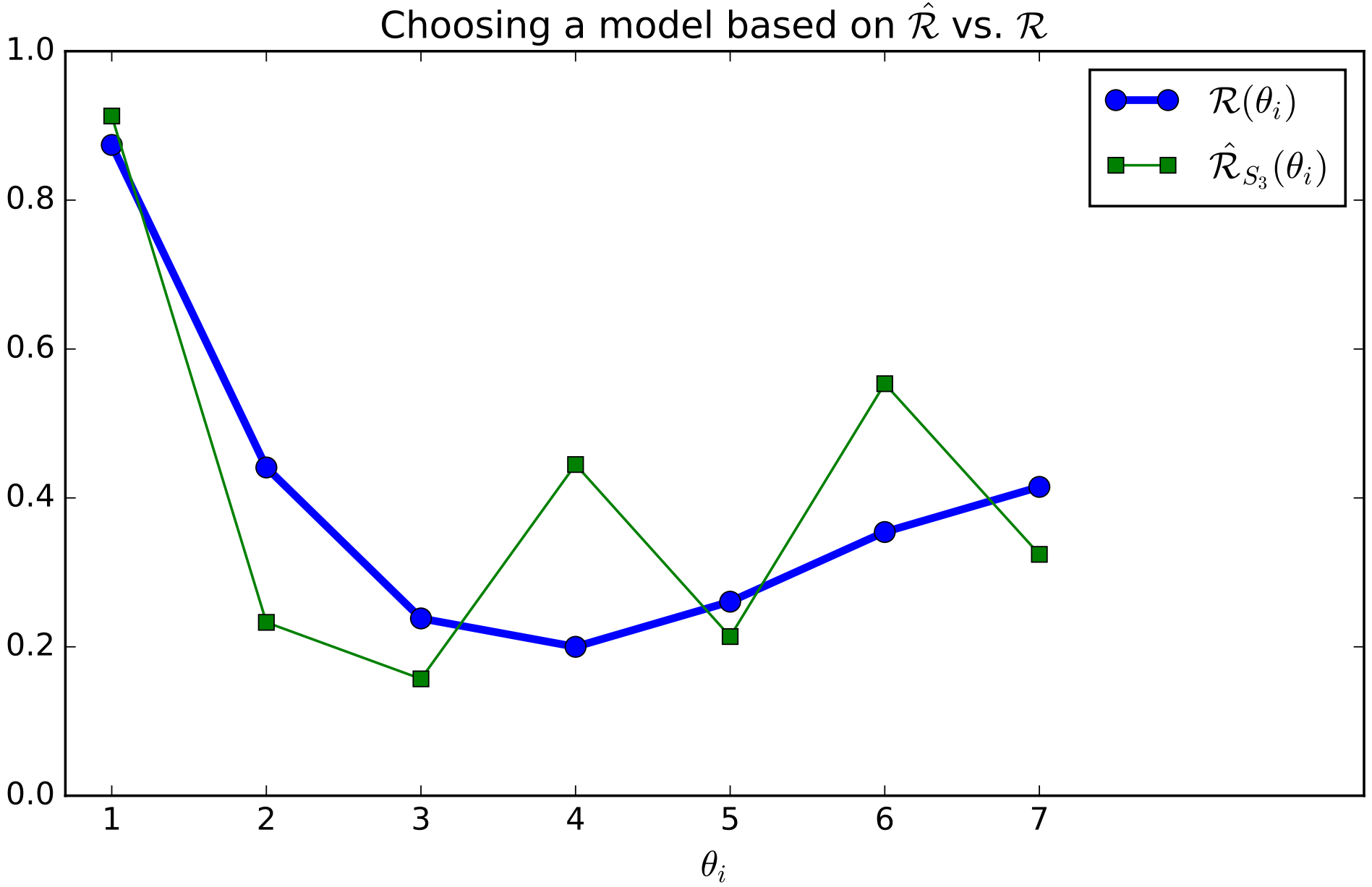
generalization error $\mathcal{R}$ for 7 different models/parameter choices

Where does overfitting come from?



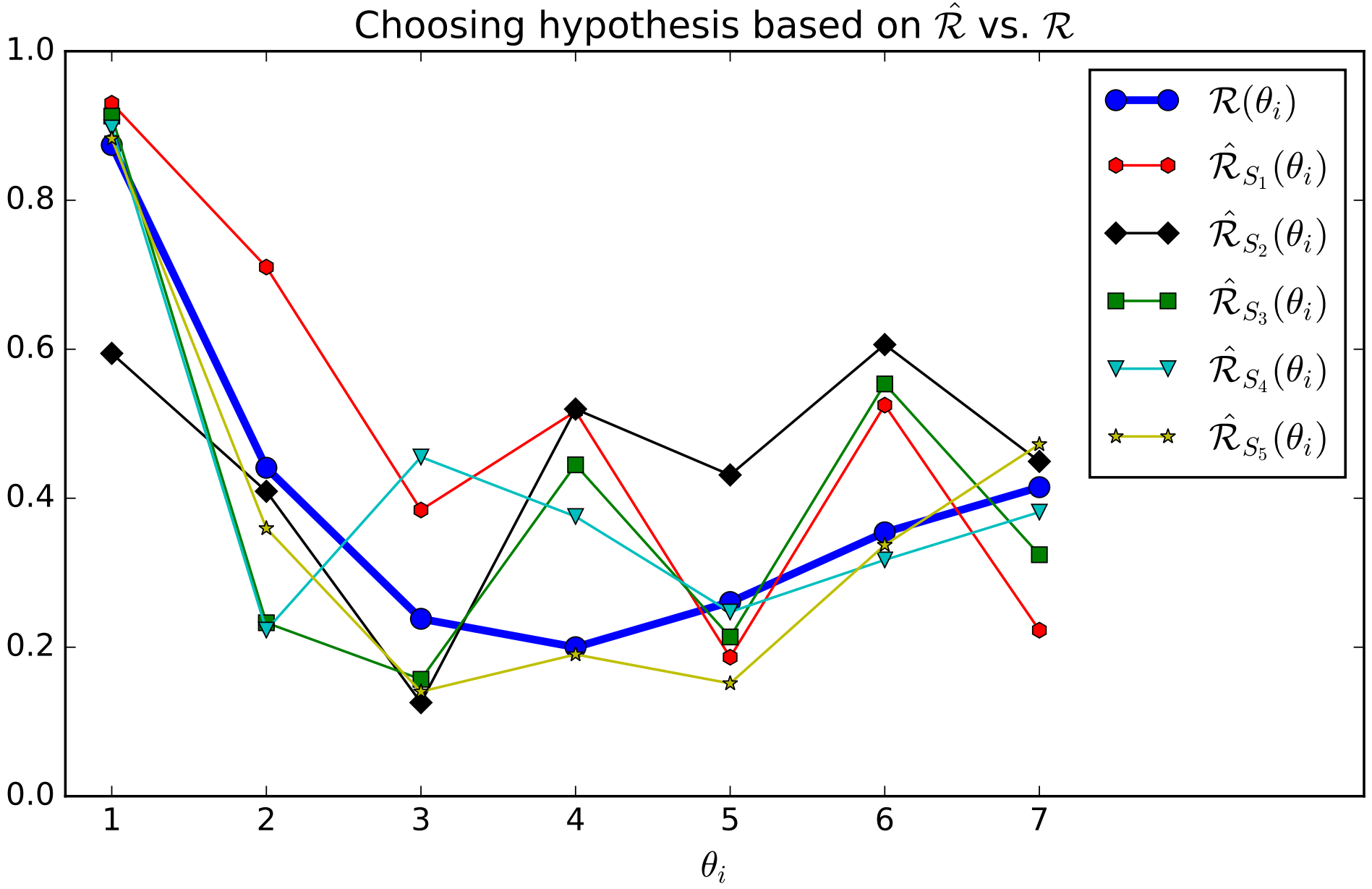
generalization error $\mathcal{R}$ for 7 different models/parameter choices

Where does overfitting come from?



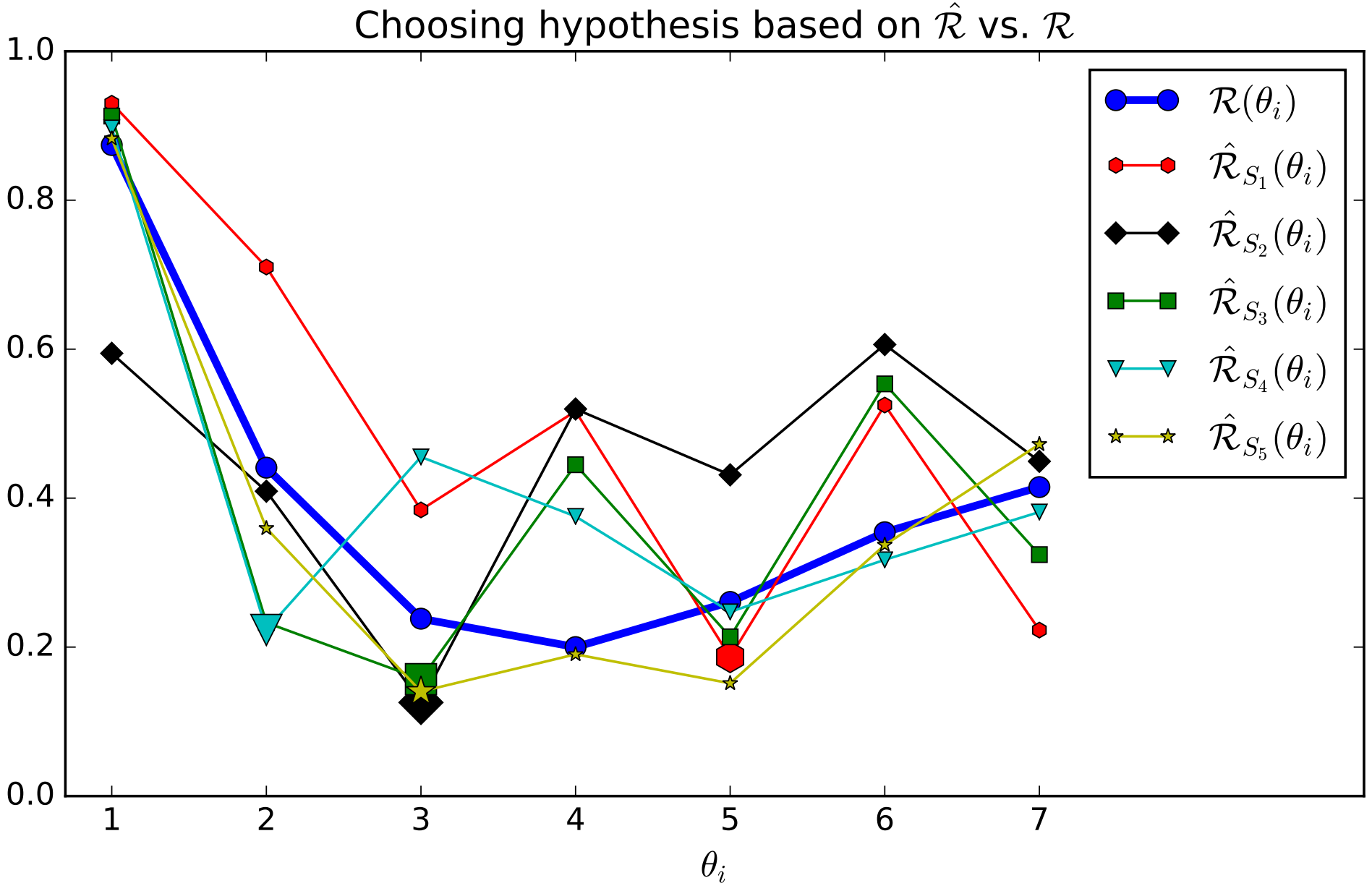
training error $\hat{\mathcal{R}}$ for a training set, S

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

Quiz: Training error

True or false?

Quiz: Training error

True or false?

- The training error is what we care about in learning.

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing.

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other.

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗
- Small training error implies small generalization error.

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗
- Small training error implies small generalization error. ✗

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗
- Small training error implies small generalization error. ✗
- Large training error implies large generalization error.

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗
- Small training error implies small generalization error. ✗
- Large training error implies large generalization error. ✗

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗
- Small training error implies small generalization error. ✗
- Large training error implies large generalization error. ✗
- In expectation across all possible training sets, small training error corresponds to small generalization error, but for individual training sets that does not have to be true.

Quiz: Training error

True or false?

- The training error is what we care about in learning. ✗
- Training error and generalization error are the same thing. ✗
- Training error and generalization error have nothing to do with each other. ✗
- Small training error implies small generalization error. ✗
- Large training error implies large generalization error. ✗
- In expectation across all possible training sets, small training error corresponds to small generalization error, but for individual training sets that does not have to be true. ✓

The training error does not tell us how good a model really is.
So, what does?

The training error does not tell us how good a model really is. So, what does?

Step 4) Model evaluation

Take new data, $S' = \{(x'_1, y'_1), \ldots, (x'_m, y'_m)\}$, and compute the model performance as

$$\hat{\mathcal{R}}_{\text{tst}} = \frac{1}{m} \sum_{j=1}^{m} \ell(y'_j, f_{\theta^*}(x'_j))$$

Take care that:

- data for evaluation comes from the real data distribution
- examples are independent from each other
- output annotation is a good as possible
- data is independent from the chosen model (in particular, we didn't train on it, we didn't use it decide to pick the model class, etc.)

If all of these are fulfilled:

- $\hat{\mathcal{R}}_{\text{tst}}$ is an unbiased estimate of $\mathcal{R}$
- if m is large enough, we can expect $\hat{\mathcal{R}}_{\text{tst}} \approx \mathcal{R}$

The training error does not tell us how good a model really is. So, what does?

Step 4) Model evaluation

Take new data, $S' = \{(x'_1, y'_1), \ldots, (x'_m, y'_m)\}$, and compute the model performance as

$$\hat{\mathcal{R}}_{\text{tst}} = \frac{1}{m} \sum_{j=1}^{m} \ell(y'_j, f_{\theta^*}(x'_j))$$

Take care that:

- data for evaluation comes from the real data distribution
- examples are independent from each other
- output annotation is a good as possible
- data is independent from the chosen model (in particular, we didn't train on it, we didn't use it decide to pick the model class, etc.)

If all of these are fulfilled:

- $\hat{\mathcal{R}}_{\text{tst}}$ is an unbiased estimate of $\mathcal{R}$
- if m is large enough, we can expect $\hat{\mathcal{R}}_{\text{tst}} \approx \mathcal{R}$

The training error does not tell us how good a model really is. So, what does?

Step 4) Model evaluation

Take new data, $S' = \{(x'_1, y'_1), \ldots, (x'_m, y'_m)\}$, and compute the model performance as

$$\hat{\mathcal{R}}_{\text{tst}} = \frac{1}{m} \sum_{j=1}^{m} \ell(y'_j, f_{\theta^*}(x'_j))$$

Take care that:

- data for evaluation comes from the real data distribution
- examples are independent from each other
- output annotation is a good as possible
- data is independent from the chosen model (in particular, we didn't train on it, we didn't use it decide to pick the model class, etc.)

If all of these are fulfilled:

- $\hat{\mathcal{R}}_{\text{tst}}$ is an unbiased estimate of $\mathcal{R}$
- if m is large enough, we can expect $\hat{\mathcal{R}}_{\text{tst}} \approx \mathcal{R}$

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L`

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L` (✓)

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L` (✓)
- all images of male drivers, if the training set is all female drivers

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L` (✓)
- all images of male drivers, if the training set is all female drivers ✗

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L` (✓)
- all images of male drivers, if the training set is all female drivers ✗
- split the drivers randomly into two sets. Use all images of one set for training and the others for testing

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L` (✓)
- all images of male drivers, if the training set is all female drivers ✗
- split the drivers randomly into two sets. Use all images of one set for training and the others for testing ✓

Quiz: test sets

Task: **Driver fatigue**. Which of these are proper **test sets**?

- images of a person that is also part of the training set ✗
- images of a person asleep if this person is awake in the training set ✗
- randomly chosen half of video frames, with the system trained on the other half ✗
- all images of drivers with names starting with `M-Z`, if the system was trained on all images of drivers with names starting with `A-L` (✓)
- all images of male drivers, if the training set is all female drivers ✗
- split the drivers randomly into two sets. Use all images of one set for training and the others for testing ✓

It's not always obvious how to form a test set (e.g. for time series/videos).

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore)

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗
- improve the model class (e.g. switch network architectures)

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗
- improve the model class (e.g. switch network architectures) ✗

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗
- improve the model class (e.g. switch network architectures) ✗
- compare your methods' accuracy to the literature

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗
- improve the model class (e.g. switch network architectures) ✗
- compare your methods' accuracy to the literature ✓

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗
- improve the model class (e.g. switch network architectures) ✗

} **"overfitting to the test set"**

- compare your methods' accuracy to the literature ✓

Quiz: overfitting to the test set

You are given a **test set**. What are you allowed to use it for?

- finetune model parameters ✗
- stop training early (e.g. because the test error does not change anymore) ✗
- improve the model class (e.g. switch network architectures) ✗

} **"overfitting to the test set"**

- compare your methods' accuracy to the literature ✓

Avoid overfitting to the test set! It

- invalidates your results (you'll think your model is better than it is),
- undermines the credibility of your experimental evaluation.

Test sets are precious! They can only be used once!

A good way to avoid overfitting the test set is not to give the test data to the person building the model (or at least withhold its annotation).

Example: ChaLearn Connectomics Challenge

- You are given labeled training data.
- You build/train a model.
- You upload the model in exectuable form to a server.
- The server applies the model to test data.
- The server evaluates the results.

Example: ImageNet ILSVR Challenge

- You are given labeled training data and unlabeled test data.
- You build/train a model.
- You apply the model to the test data.
- You upload the model predictions to a server.
- The server evaluates the results.

Other Forms of Evaluation: Information Retrieval Measures

Some computer vision tasks are really retrieval tasks, e.g.

- online image search: return K images that are relevant for a keyword
- face detection: return all regions in an image that contain a face

Typical properties:

- prediction is performed on a fixed (test) database
- we have access to all elements of the test set at the same time
- positives ($y = 1$) are important, negative ($y = -1$) are a nuisanse

For some scenarios, e.g. web search:

- we don't need all decisions, a few correct positives are enough

We can use a trained model $f : \mathcal{X} \to \mathbb{R}$:

- interpret model output $f(x)$ as *confidence* of relevance
- return objects in order of decreasing confidence
- if only K outputs are needed, return K most confident ones

Other Forms of Evaluation: Information Retrieval Measures

Retrieval quality is often measure in terms of precision and recall:

- $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$: *database*
  with $y_j = 1$ if x_j is relevant for the query and $y_j = -1$ otherwise
- $f : \mathcal{X} \to \mathbb{R}$: model that predicts confidences of being relevant

Precision, Recall, F-Score

For any threshold $\vartheta \in \mathbb{R}$:

$$\textit{precision} = \frac{\textit{number of test samples with } f(x_j) \geq \vartheta \textit{ and } y_j = 1}{\textit{number of test samples with } f(x_j) \geq \theta}$$

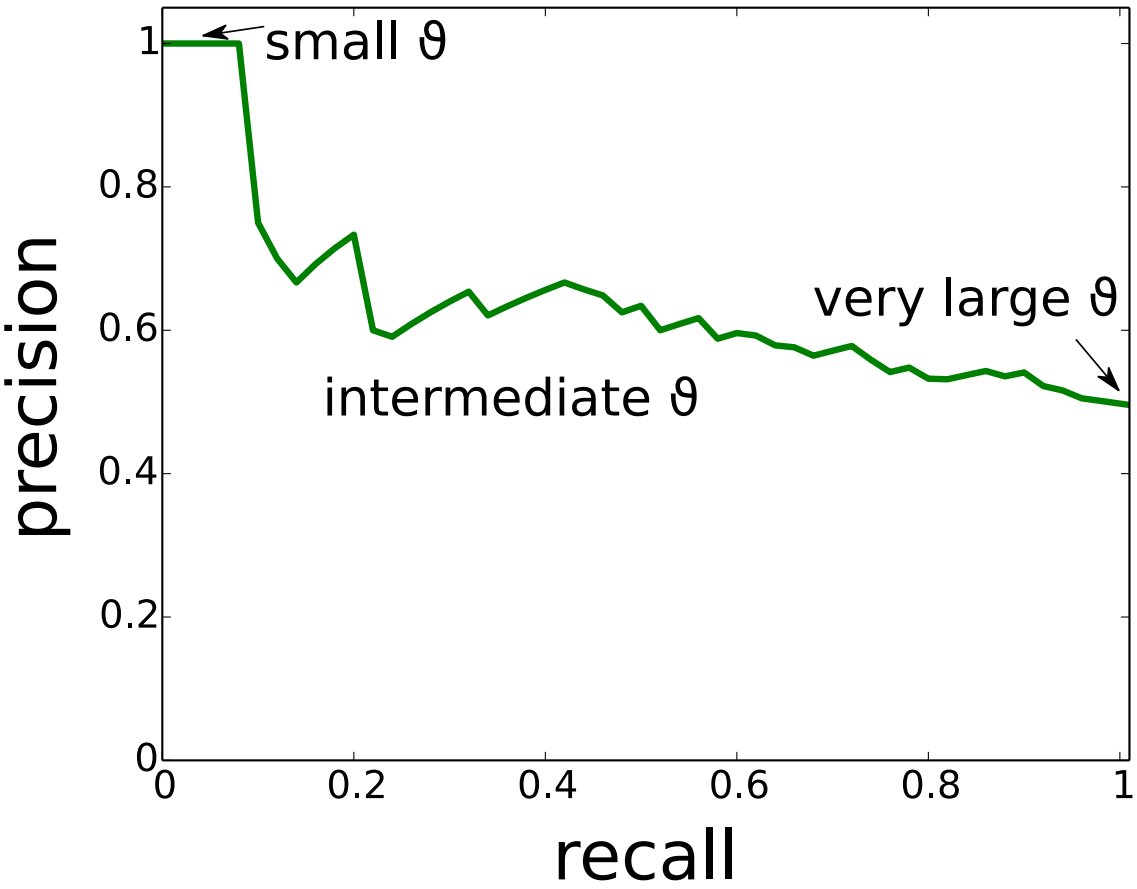
$$\textit{recall} = \frac{\textit{number of test samples with } f(x_j) \geq \vartheta \textit{ and } y_j = 1}{\textit{number of test samples with } y_j = 1}$$

$$F\textit{-score} = 2\frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}}$$

F-score is a tough measure: high values only if simultaneously high precision and high recall

For different thresholds, ϑ, we obtain different precision and recall values.

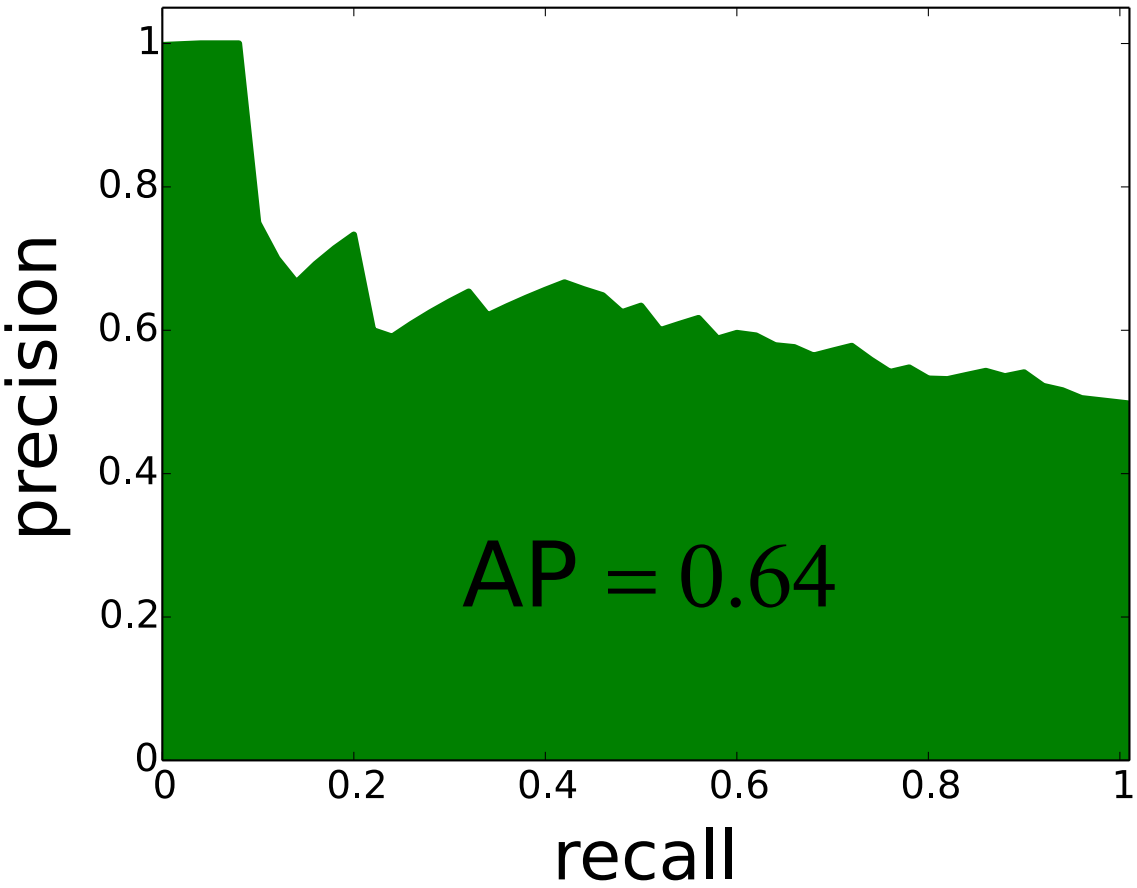
They are summarized by a **precision-recall curve**:



- typically decreasing, but not monotonic

For different thresholds, ϑ, we obtain different precision and recall values.

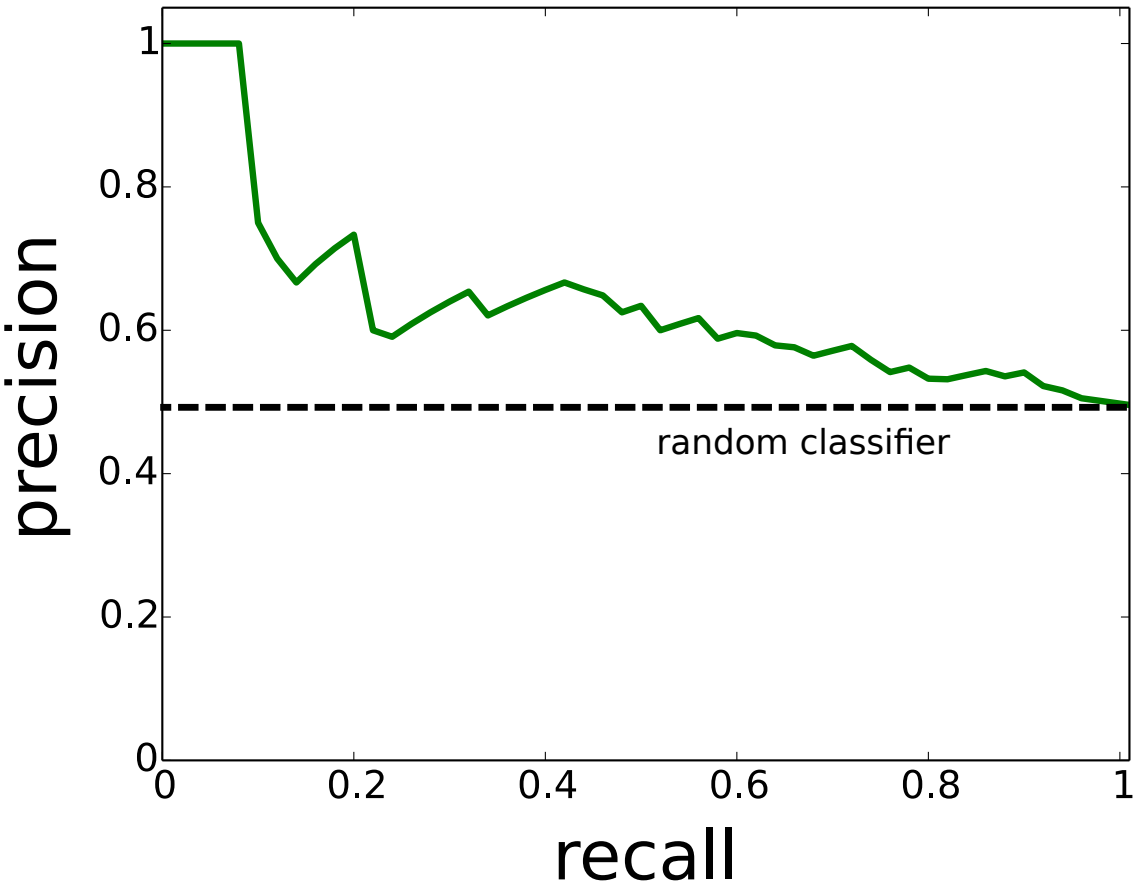
They are summarized by a **precision-recall curve**:



- typically decreasing, but not monotonic
- if forced, summarize into one number: **average precision**.

For different thresholds, ϑ, we obtain different precision and recall values.

They are summarized by a **precision-recall curve**:



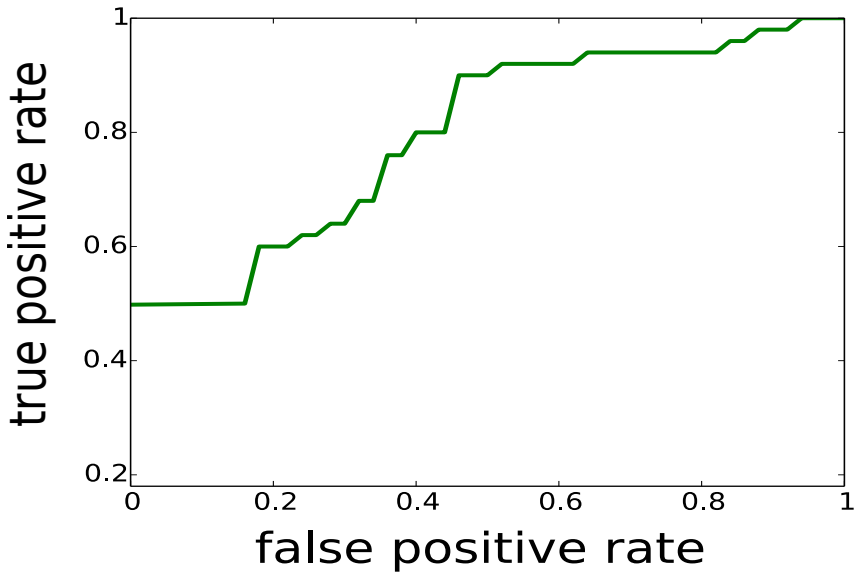
- typically decreasing, but not monotonic
- if forced, summarize into one number: **average precision**.
- values depends on positive/negative ratio: higher if more positives
- random classifier: flat line where precision equals pos/neg ratio

Receiver Operating Characteristic (ROC) Curve

For any threshhold $\vartheta \in \mathbb{R}$:

$$\textit{true-positive-rate} = \frac{\textit{number of samples with } f(x_j) \geq \vartheta \textit{ and } y_j = 1}{\textit{number of samples with } y_j = 1}$$

$$\textit{false-positive-rate} = \frac{\textit{number of samples with } f(x_j) \geq \vartheta \textit{ and } y_j = -1}{\textit{number of samples with } y_j = -1}$$

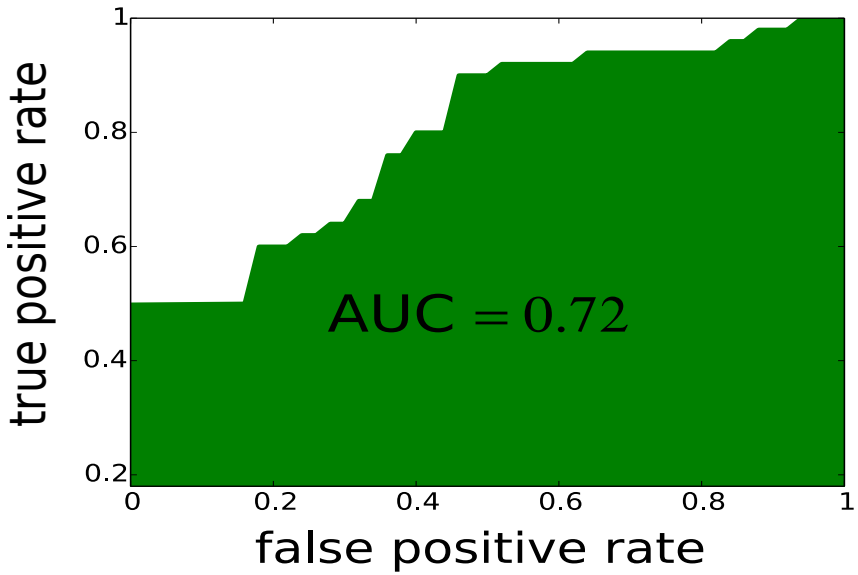


Receiver Operating Characteristic (ROC) Curve

For any threshhold $\vartheta \in \mathbb{R}$:

$$\textit{true-positive-rate} = \frac{\textit{number of samples with } f(x_j) \geq \vartheta \textit{ and } y_j = 1}{\textit{number of samples with } y_j = 1}$$

$$\textit{false-positive-rate} = \frac{\textit{number of samples with } f(x_j) \geq \vartheta \textit{ and } y_j = -1}{\textit{number of samples with } y_j = -1}$$



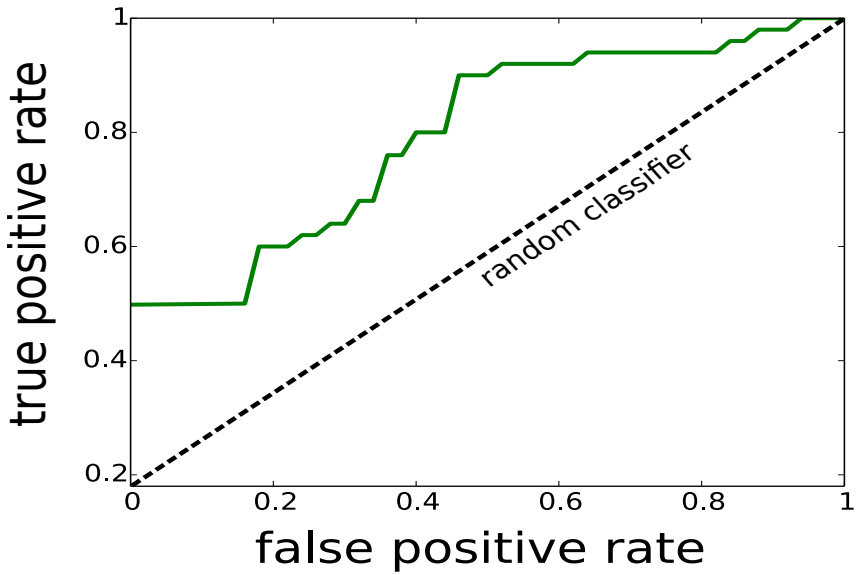
Summarize into: **area under ROC curve (AUC)**.

Receiver Operating Characteristic (ROC) Curve

For any threshhold $\vartheta \in \mathbb{R}$:

$$\textit{true-positive-rate} = \frac{\textit{number of samples with } f(x_j) \geq \vartheta \textit{ and } y_j = 1}{\textit{number of samples with } y_j = 1}$$

$$\textit{false-positive-rate} = \frac{\textit{number of samples with } f(x_j) \geq \vartheta \textit{ and } y_j = -1}{\textit{number of samples with } y_j = -1}$$

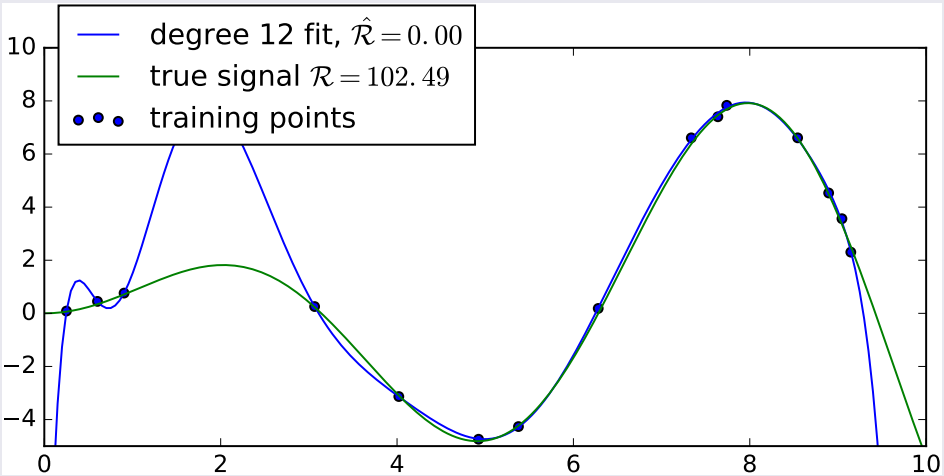


Summarize into: **area under ROC curve (AUC)**.

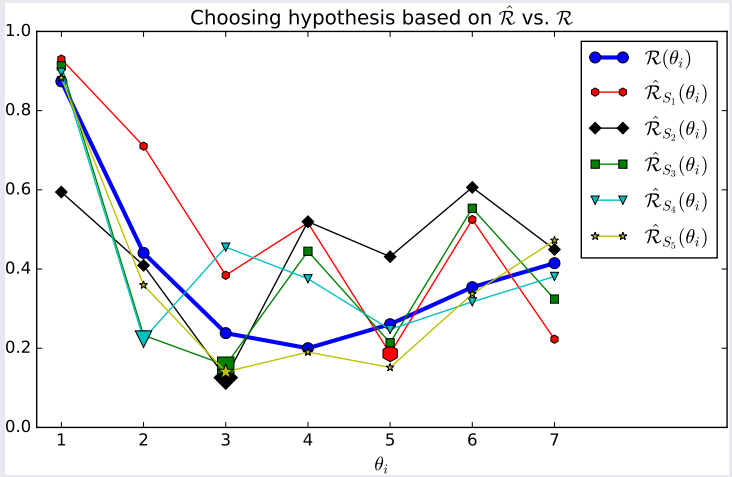
Random classifier: $AUC = 0.5$, regardless of positive/negative ratio.

Regularization

Reminder: Overfitting



overfitting

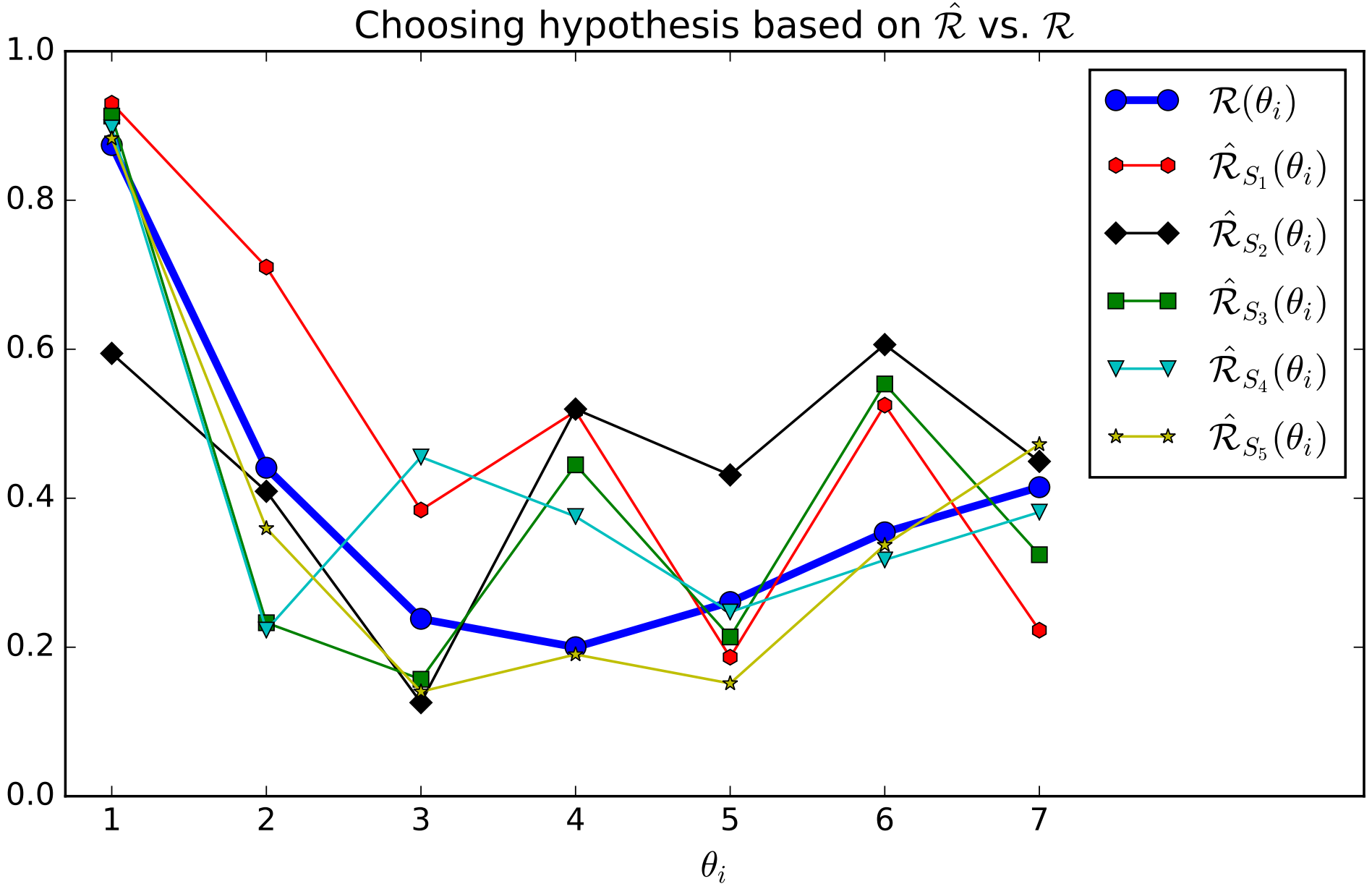


$\hat{\mathcal{R}}$ **vs.** $\mathcal{R}$

How can we prevent overfitting when learning a model?

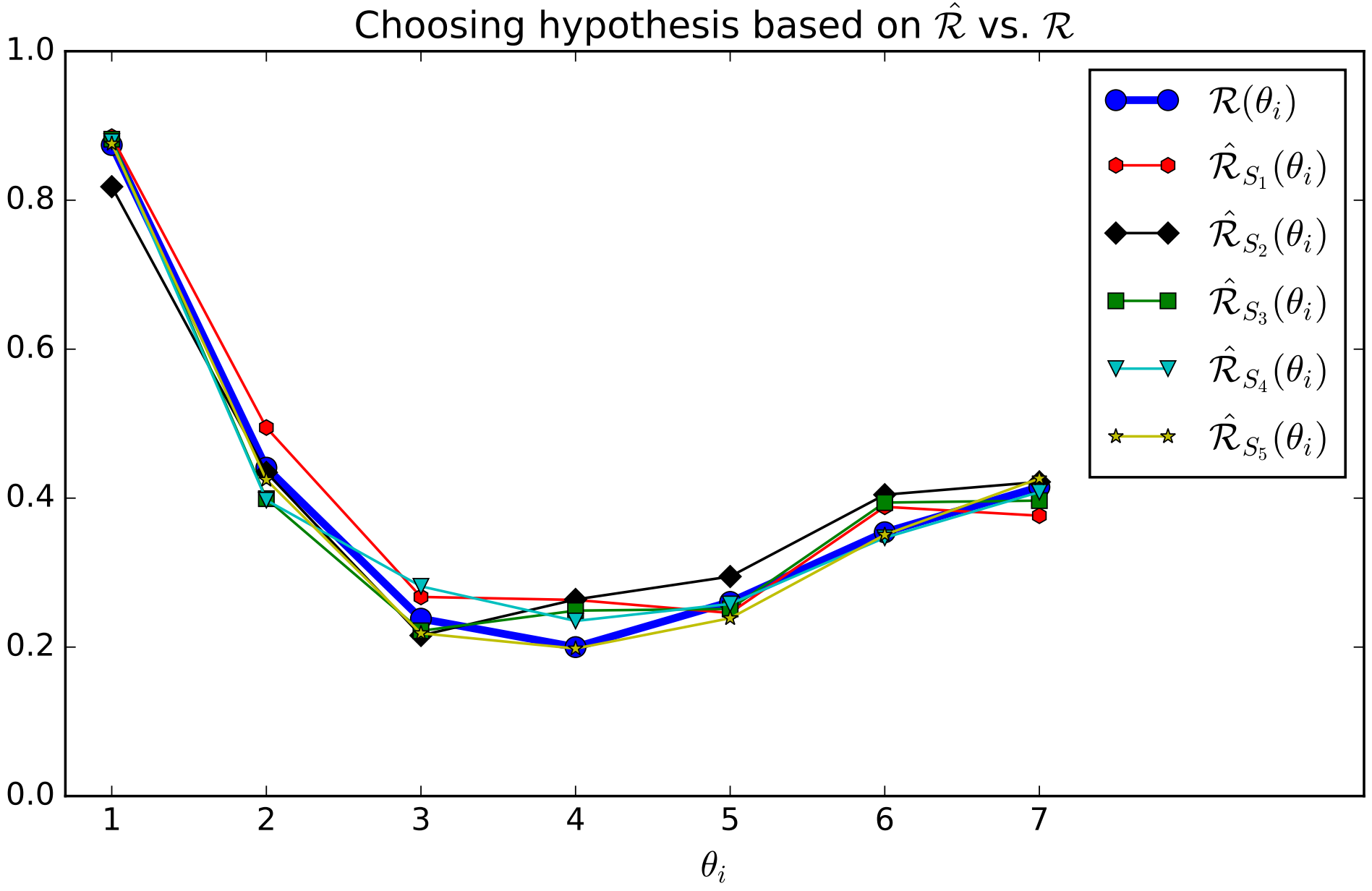
Preventing overfitting

Preventing overfitting 1) larger training set



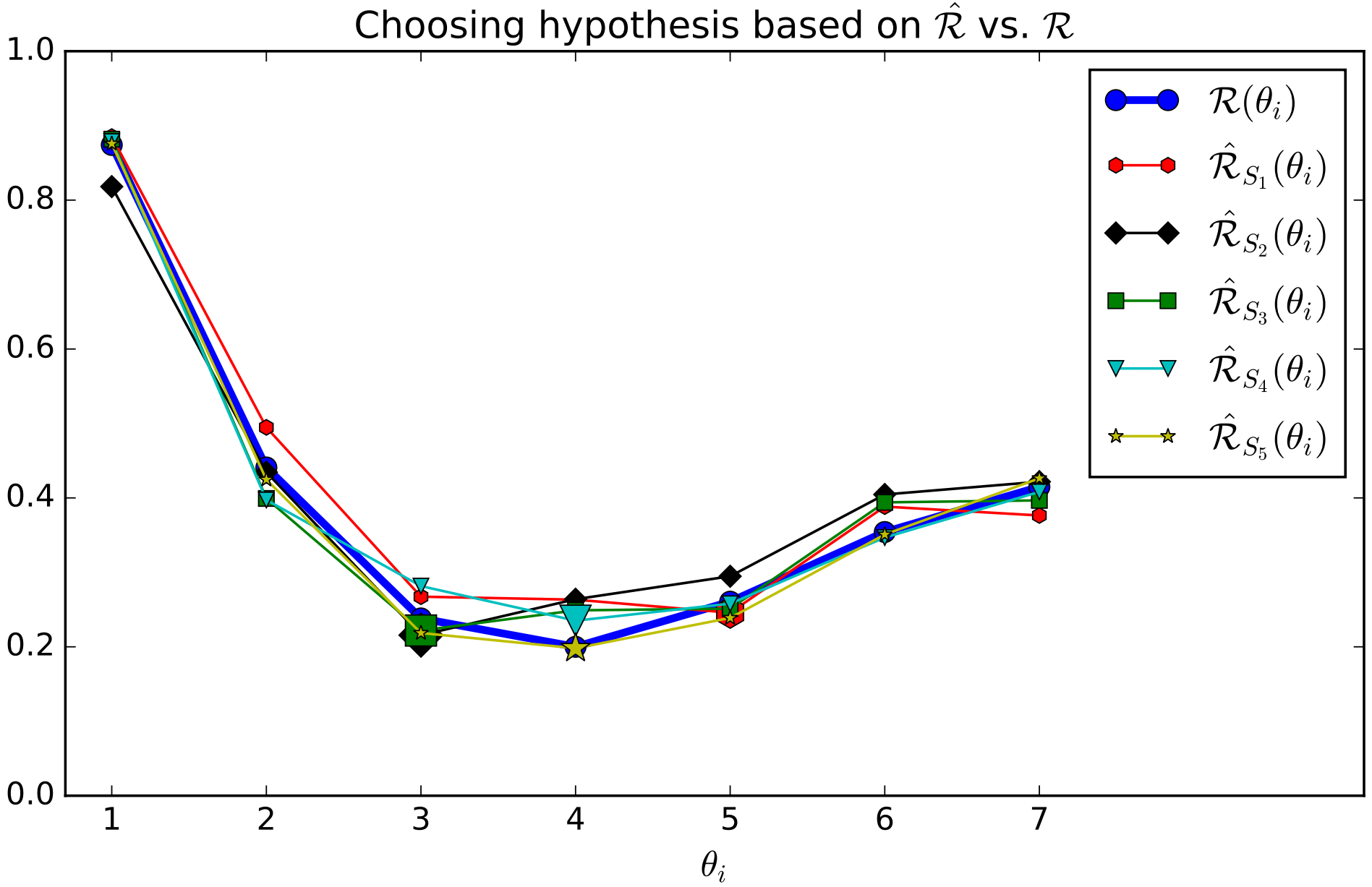
larger training set $\rightarrow$ smaller variance of $\hat{\mathcal{R}}$

Preventing overfitting 1) larger training set



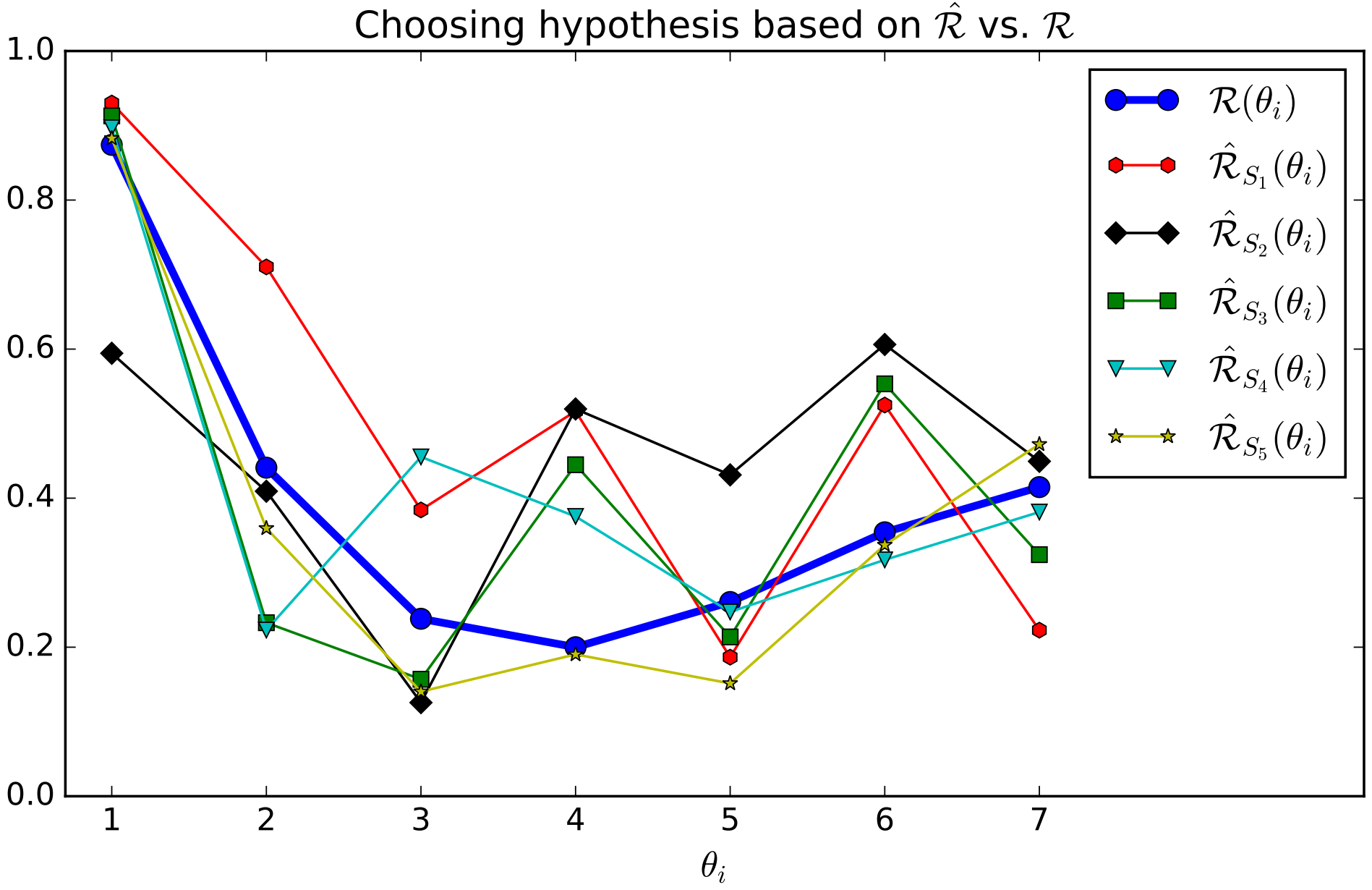
lower probability that $\hat{\mathcal{R}}$ differs strongly from $\mathcal{R}$

Preventing overfitting 1) larger training set

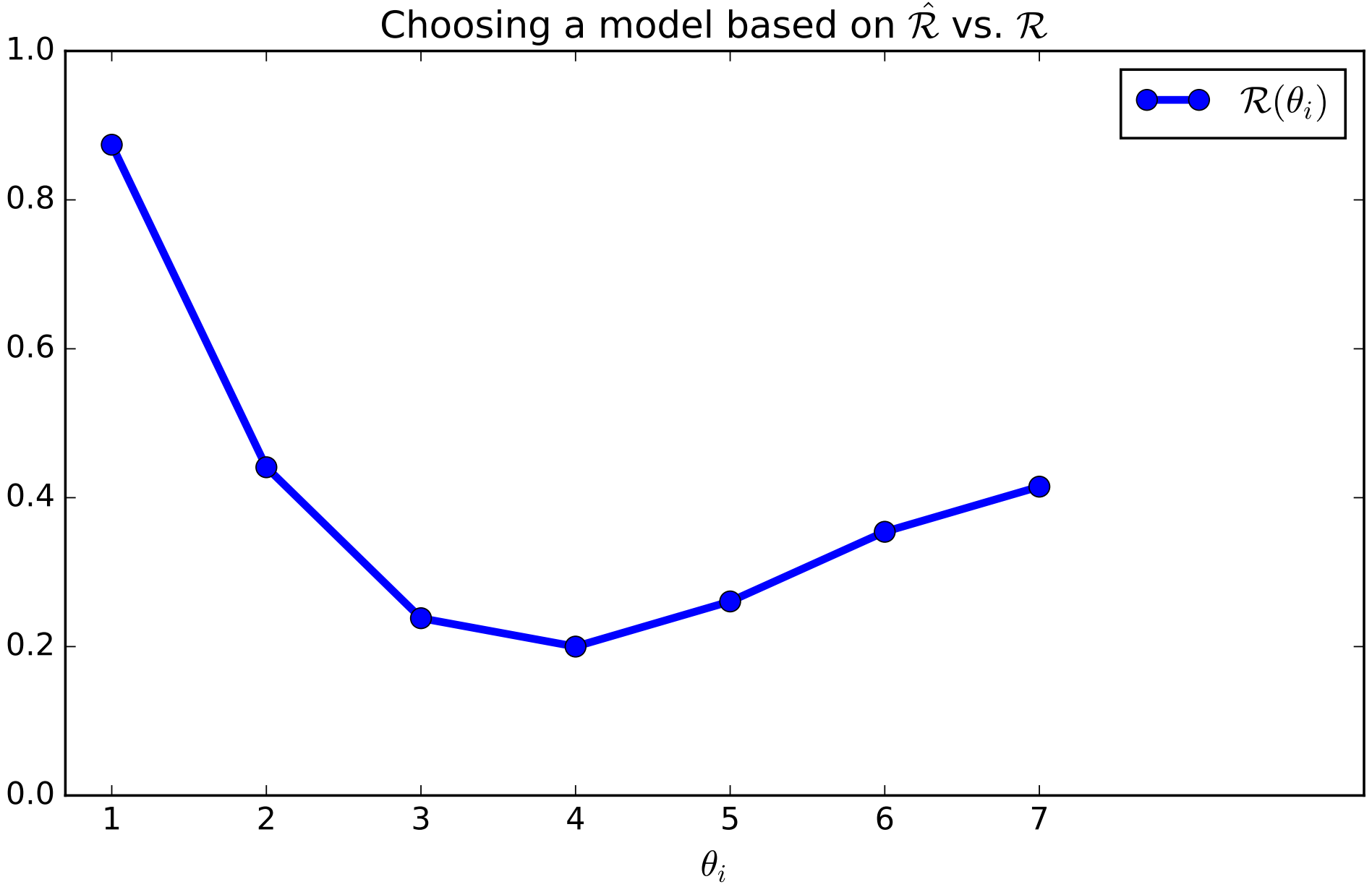


lower probability that $\hat{\mathcal{R}}$ differs strongly from $\mathcal{R} \rightarrow$ less overfitting

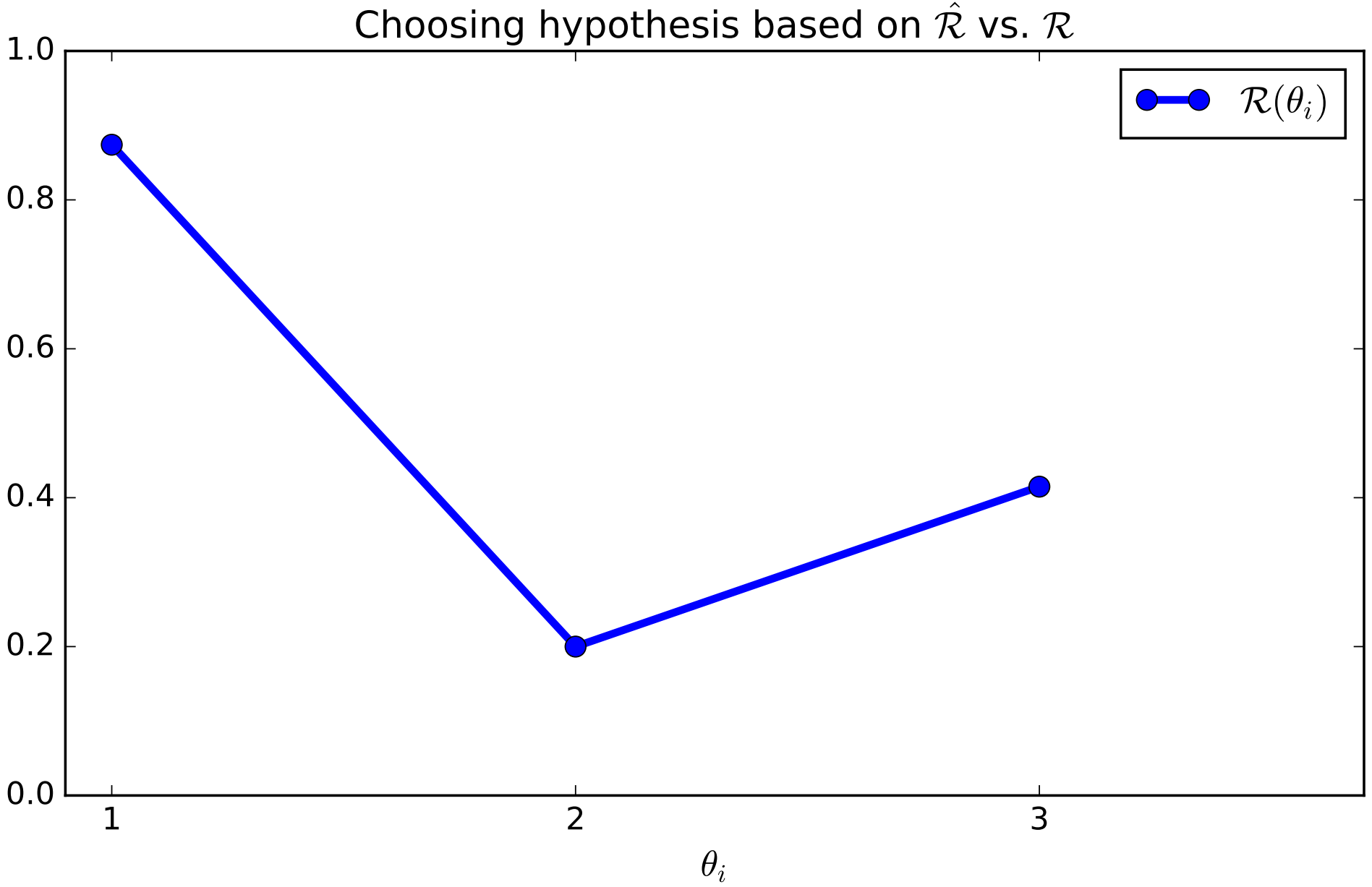
Preventing overfitting 2) reduce the number of models to choose from



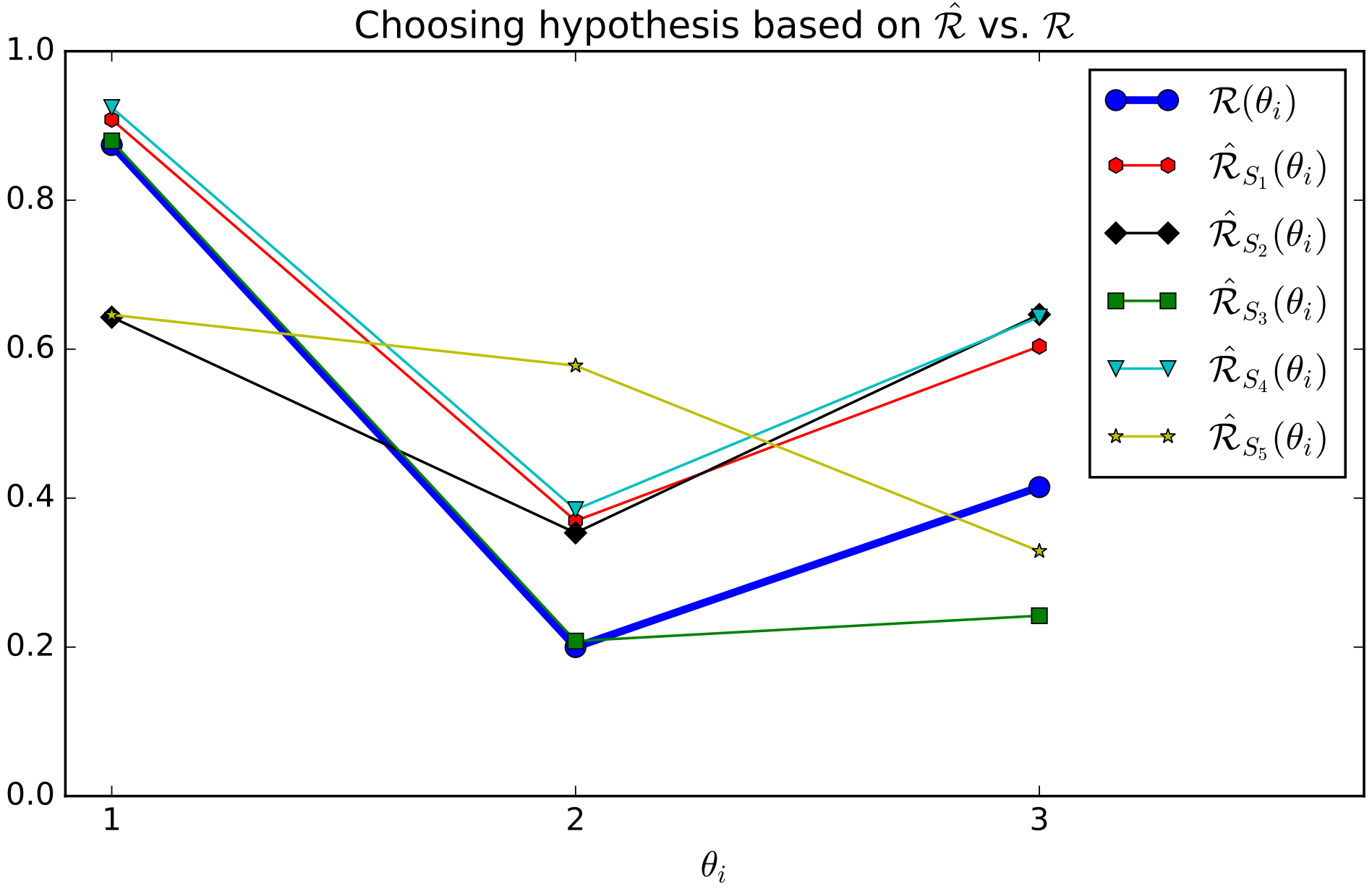
Preventing overfitting 2) reduce the number of models to choose from



Preventing overfitting 2) reduce the number of models to choose from

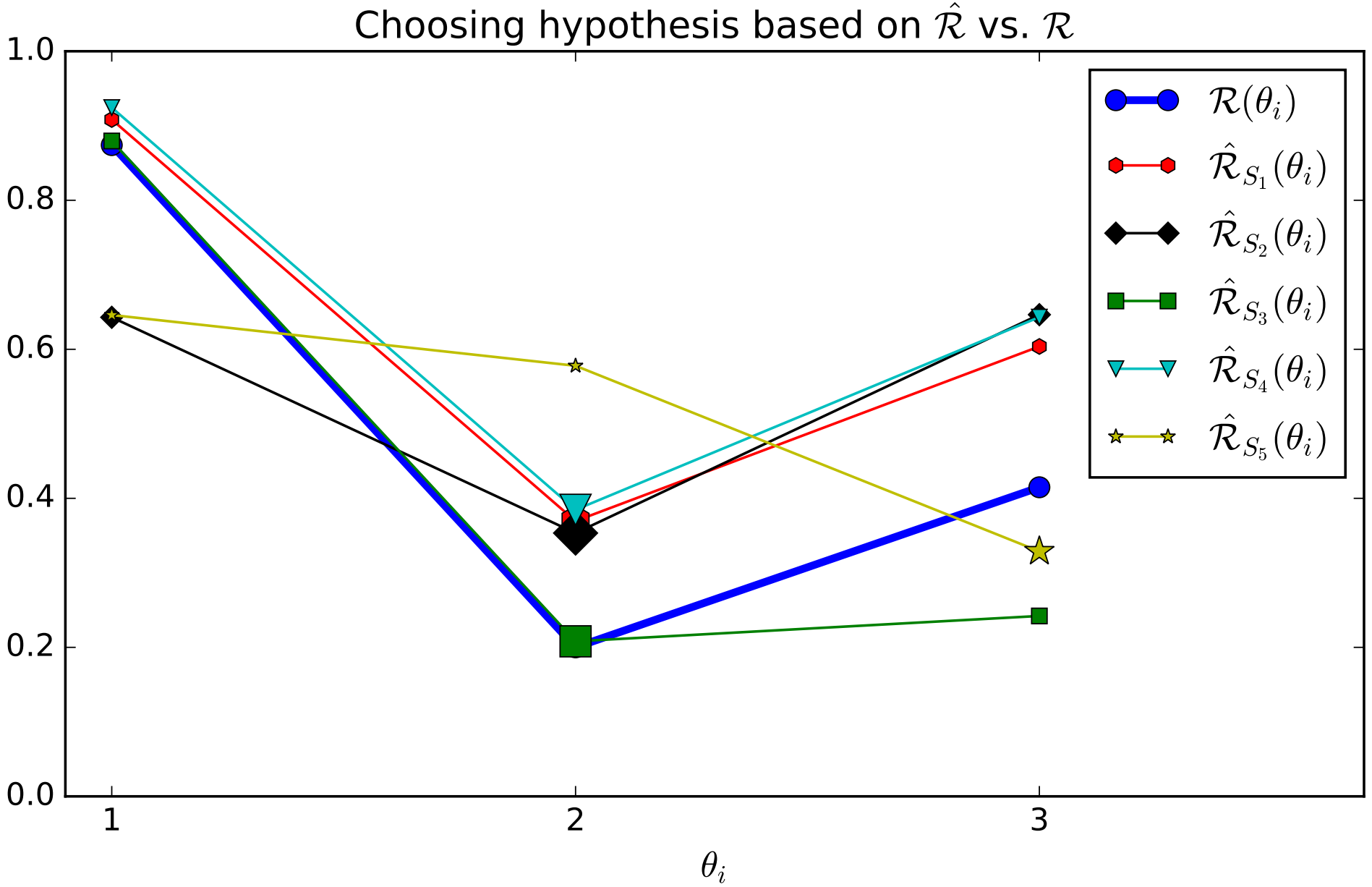


Preventing overfitting 2) reduce the number of models to choose from



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

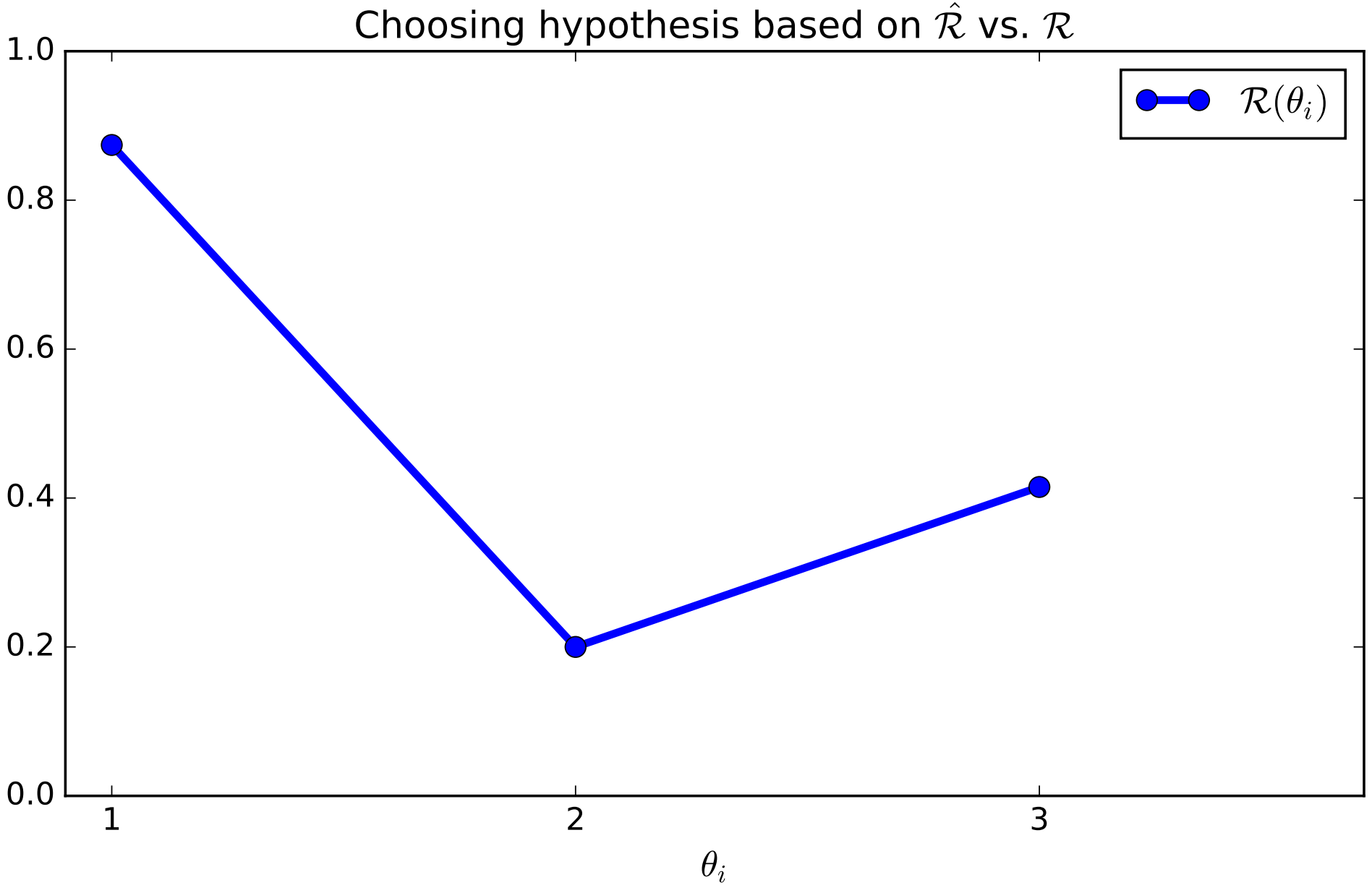
Preventing overfitting 2) reduce the number of models to choose from



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

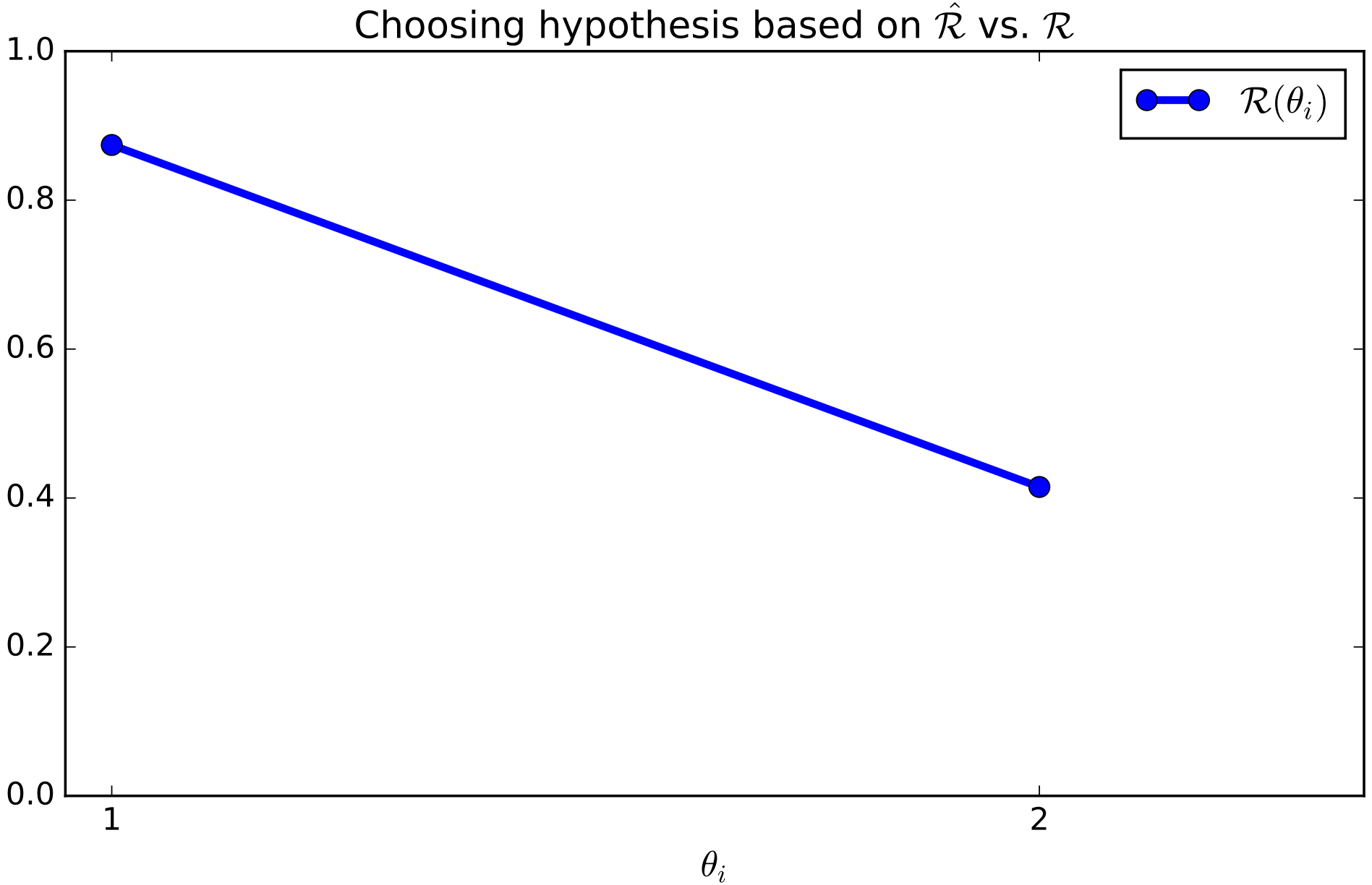
But: danger of underfitting

But: danger of underfitting



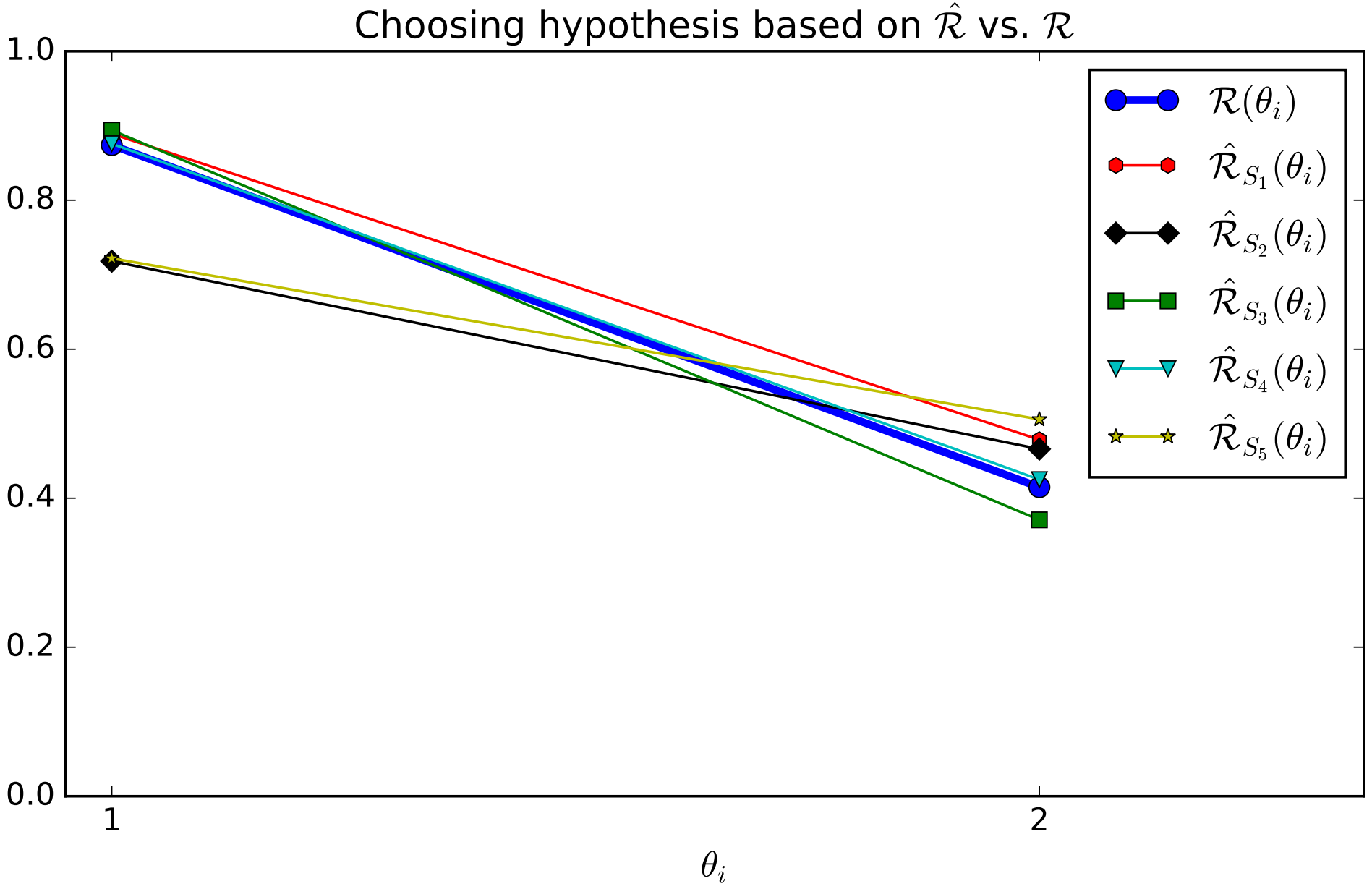
to few models select to from $\rightarrow$ danger that no model with low $\mathcal{R}$ is left!

But: danger of underfitting



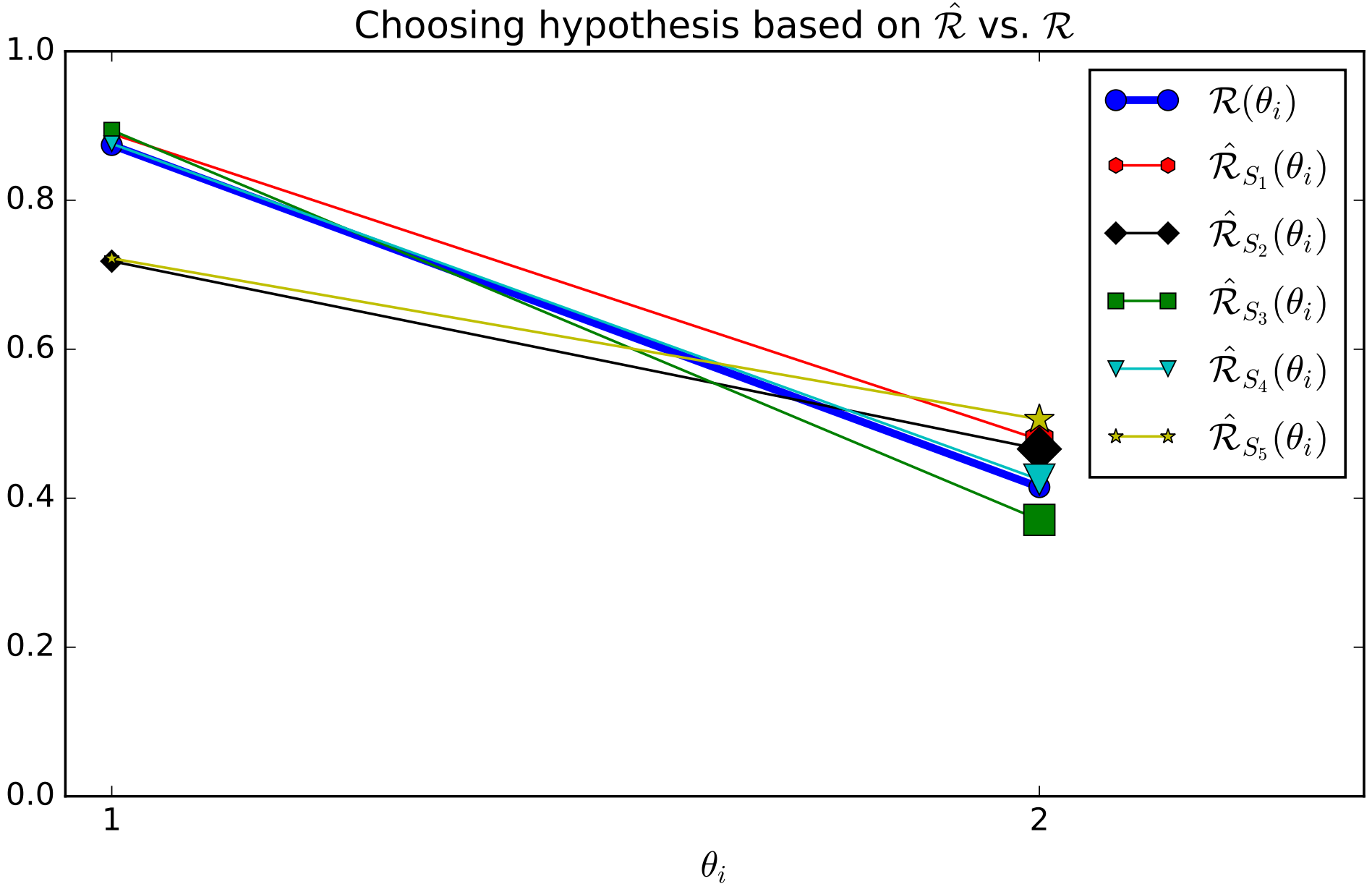
to few models select to from $\rightarrow$ danger that no model with low $\mathcal{R}$ is left!

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)

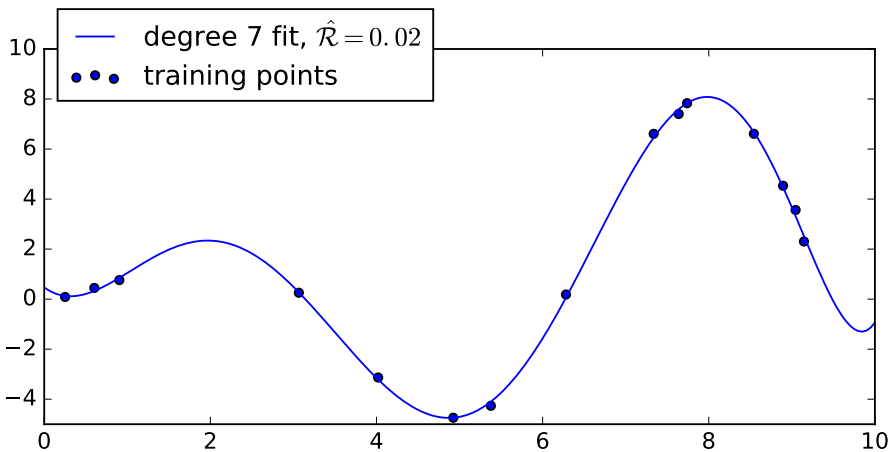
How to avoid overfitting? Use a model class that is

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

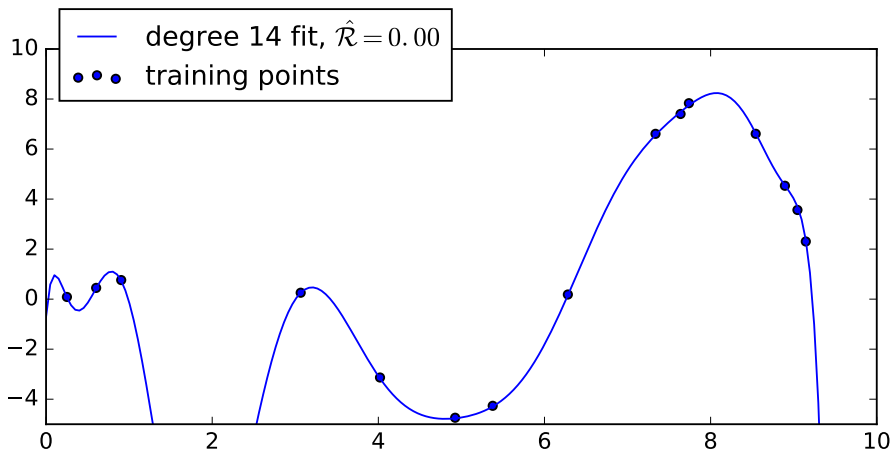
Regularization

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

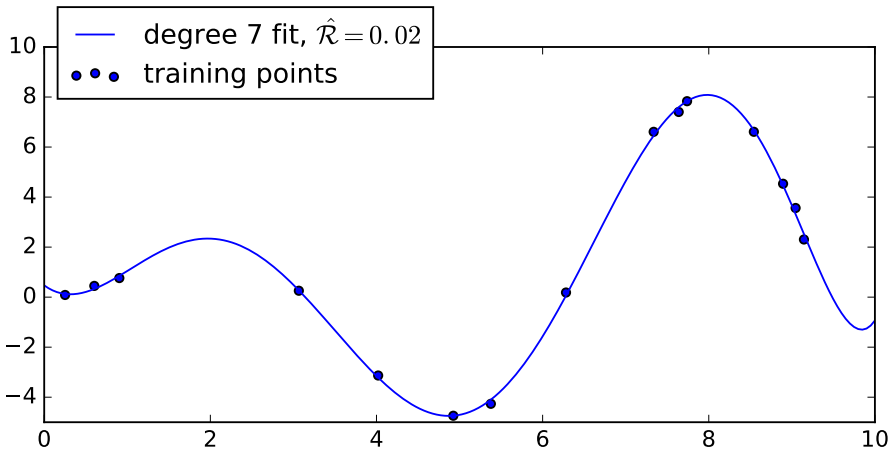


coeffs: $\theta_i \in [-1312.5, 1136.6]$

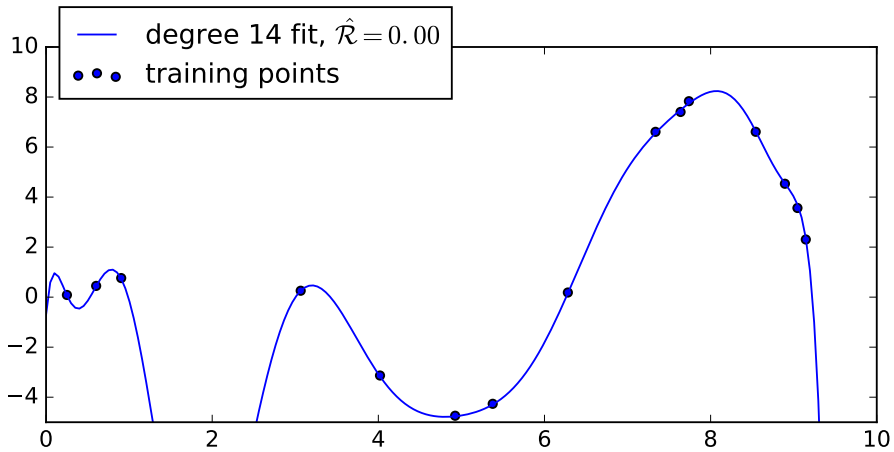
Regularization

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, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, find θ^* 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, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, find θ^* 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)

Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noisy 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

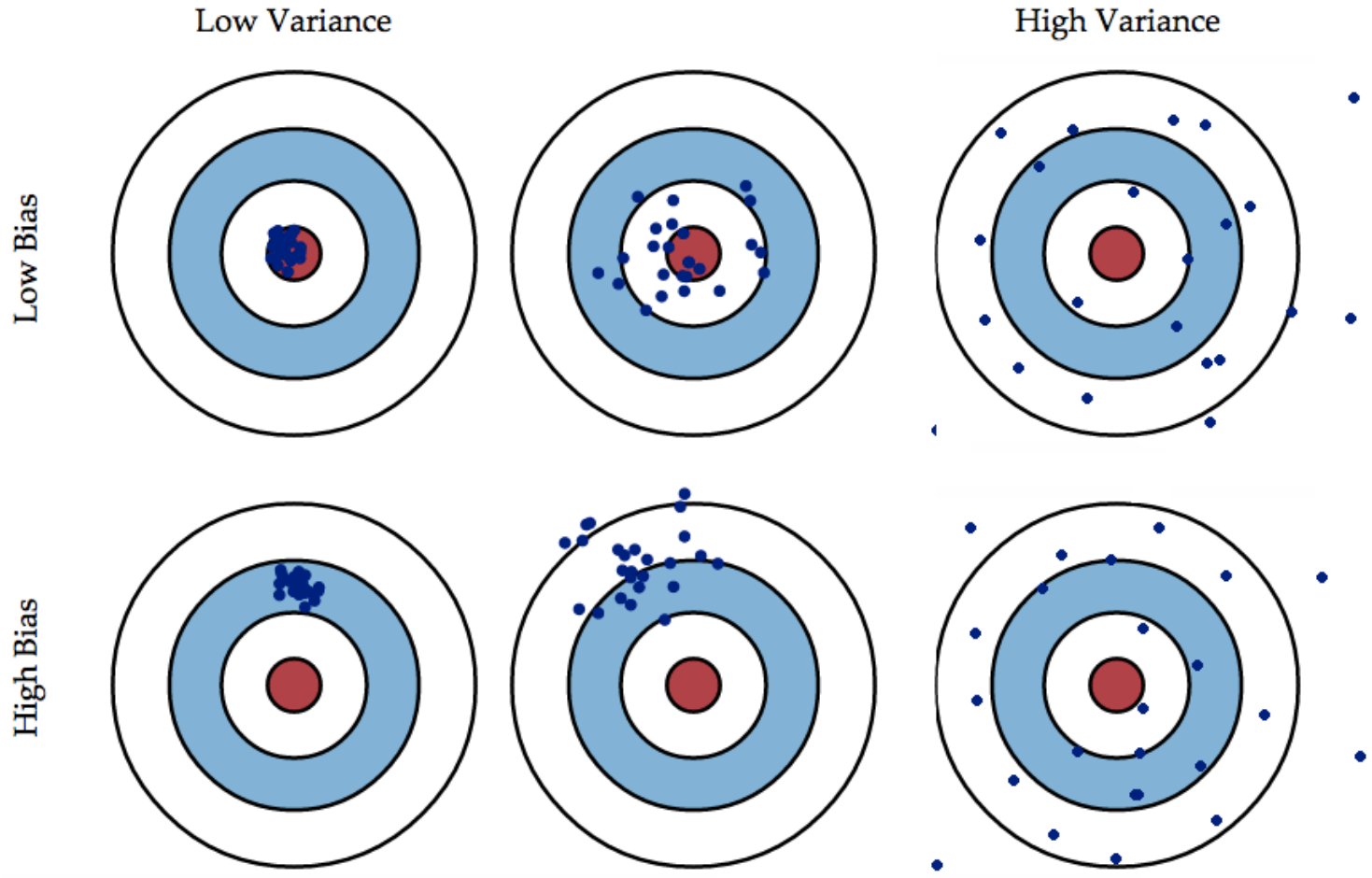


Image: adapted from http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noisy 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

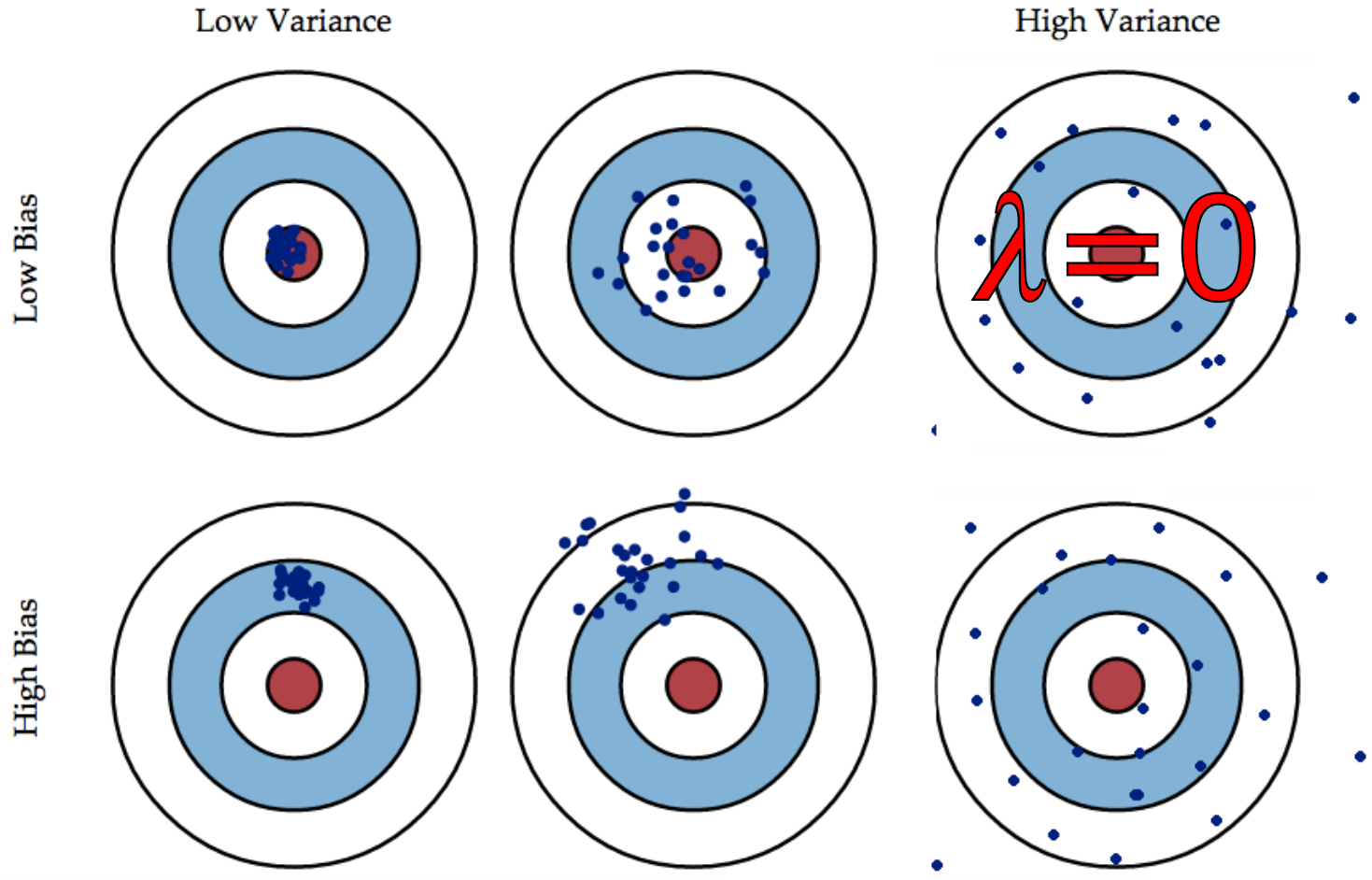


Image: adapted from http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noisy 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

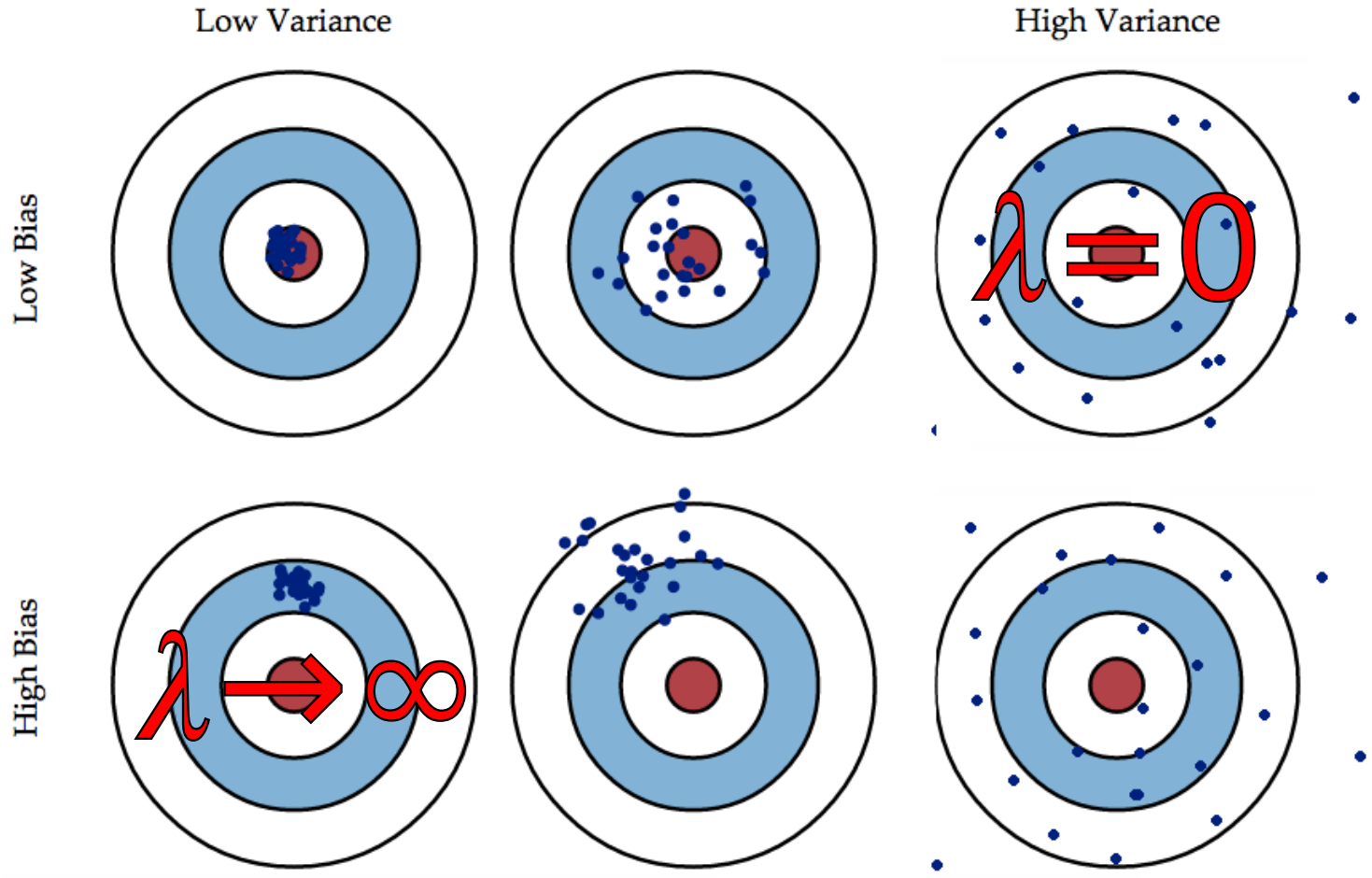


Image: adapted from http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularization as Trading Off Bias and Variance

Training error, $\hat{\mathcal{R}}$, is a noisy 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

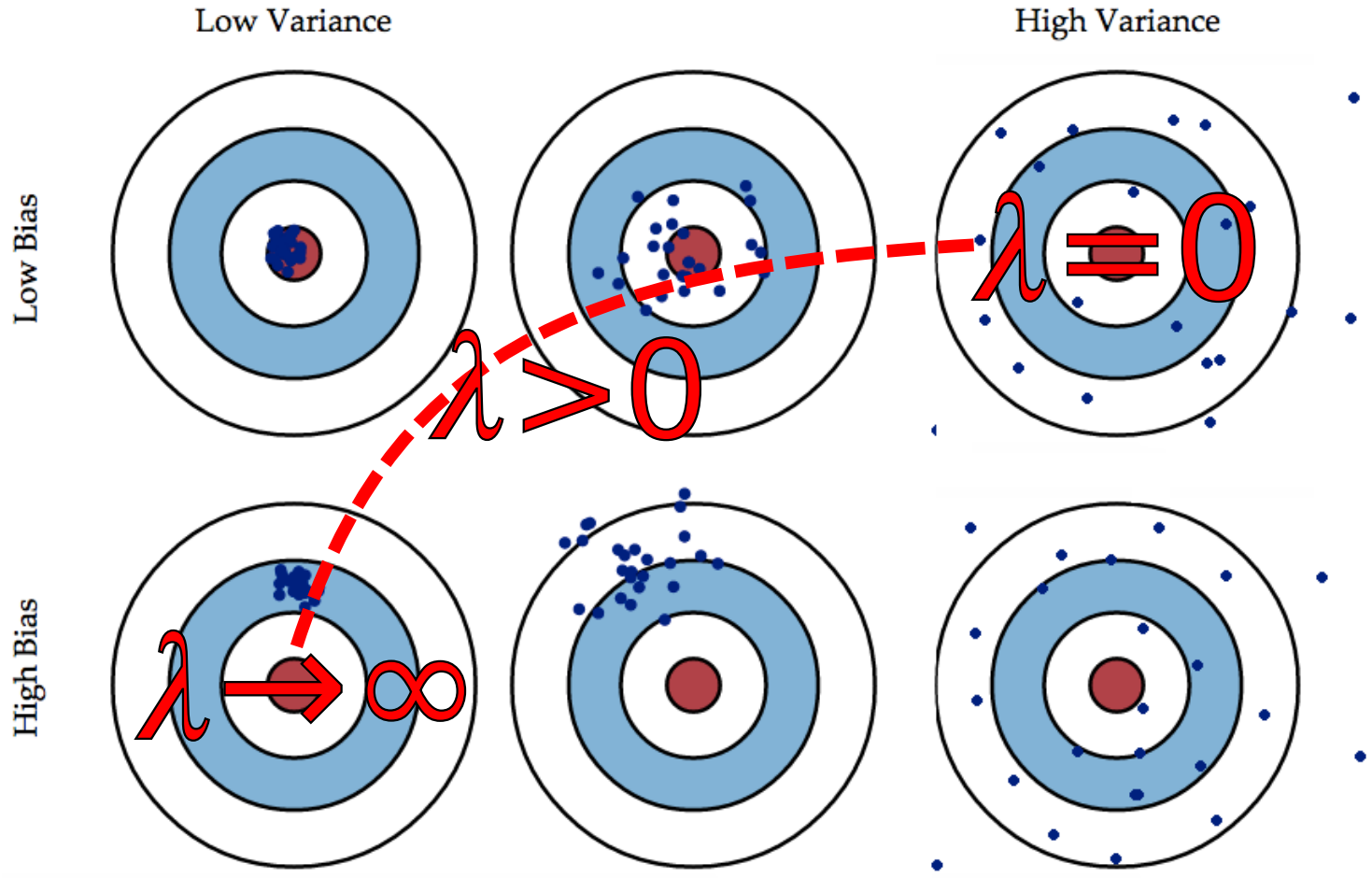


Image: adapted from http://scott.fortmann-roe.com/docs/BiasVariance.html

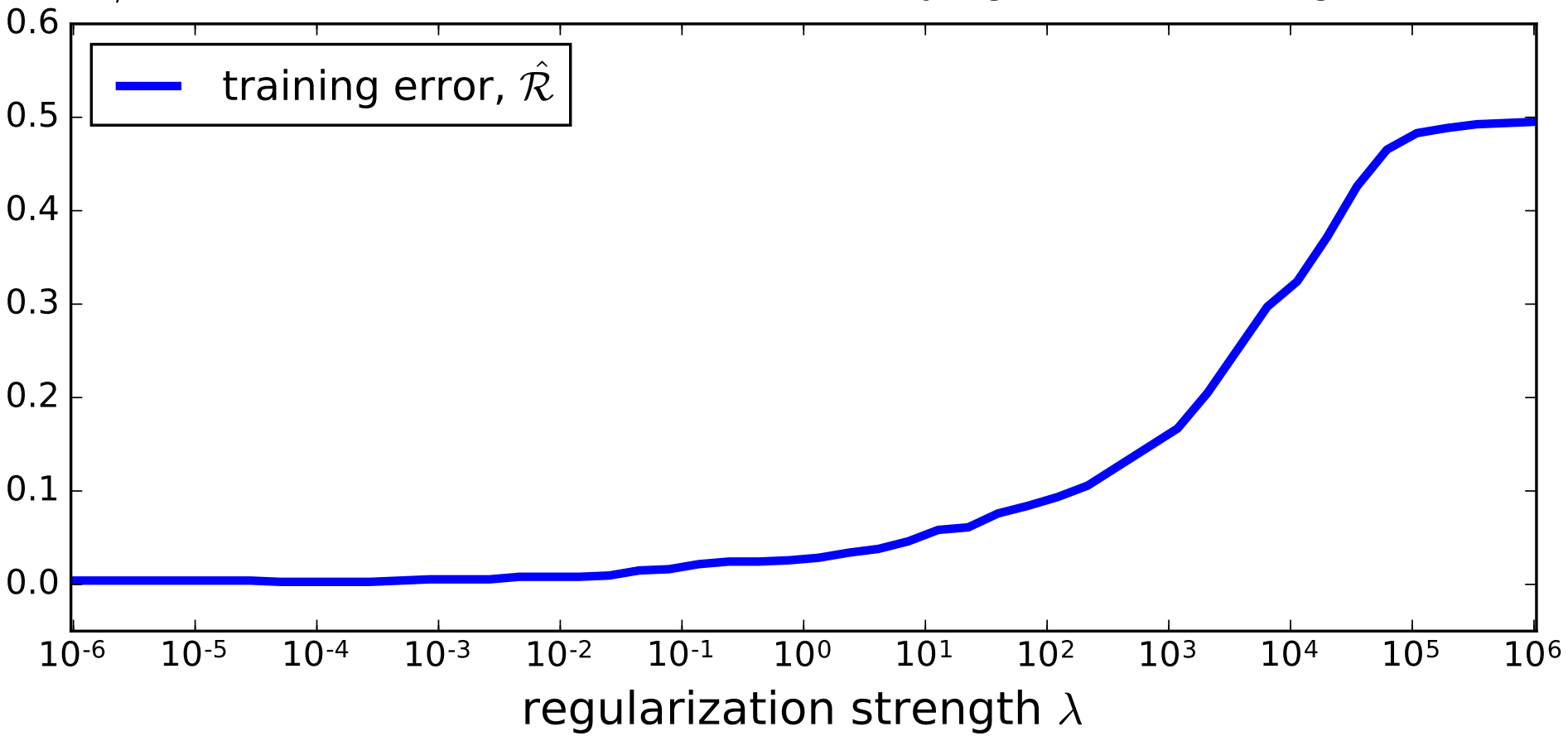
Example: regularized linear least-squared regression

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

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 minimizer of J_λ with varying amounts of regularization:

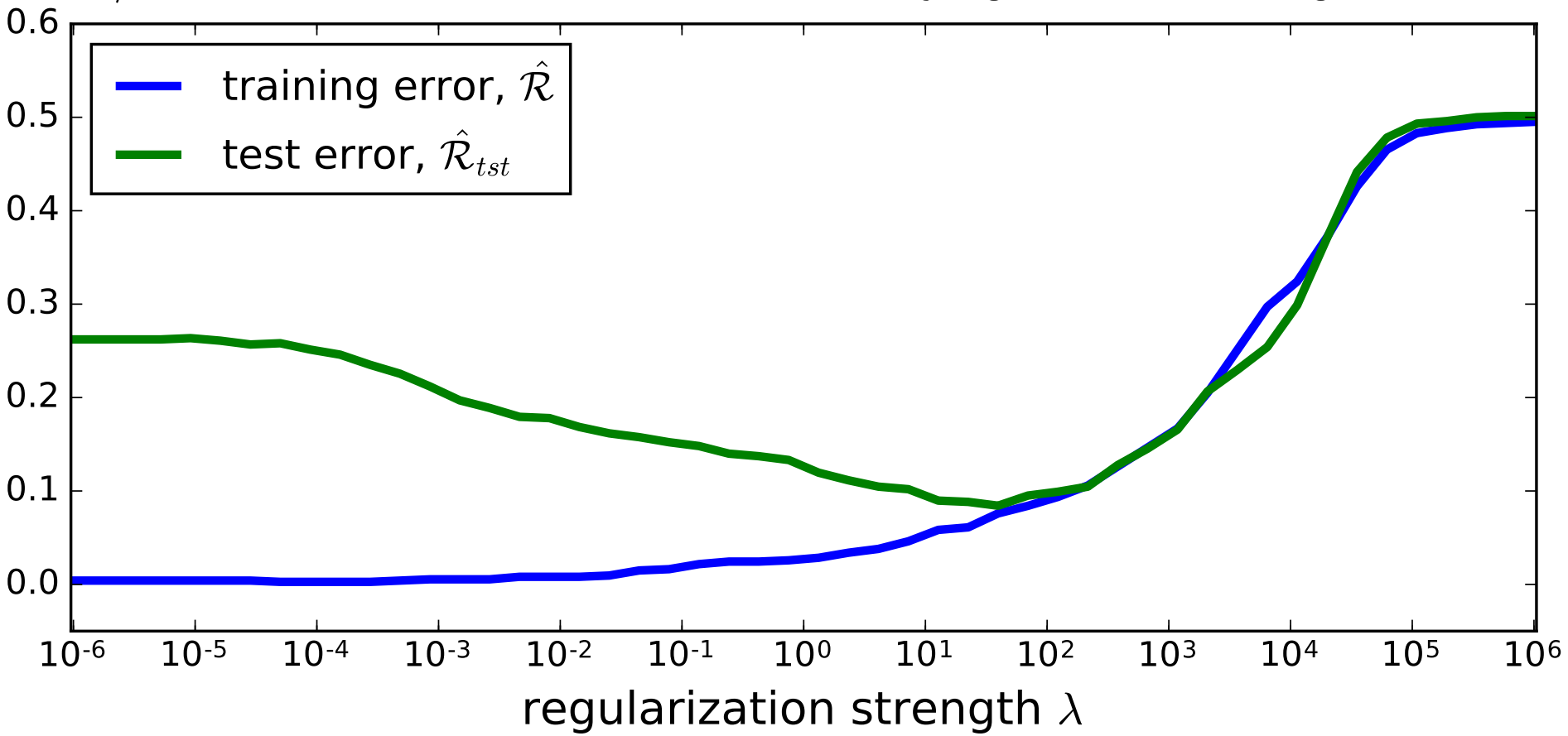


eye dataset: 737 examples for training, 736 examples for evaluation

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 minimizer of J_λ with varying amounts of regularization:

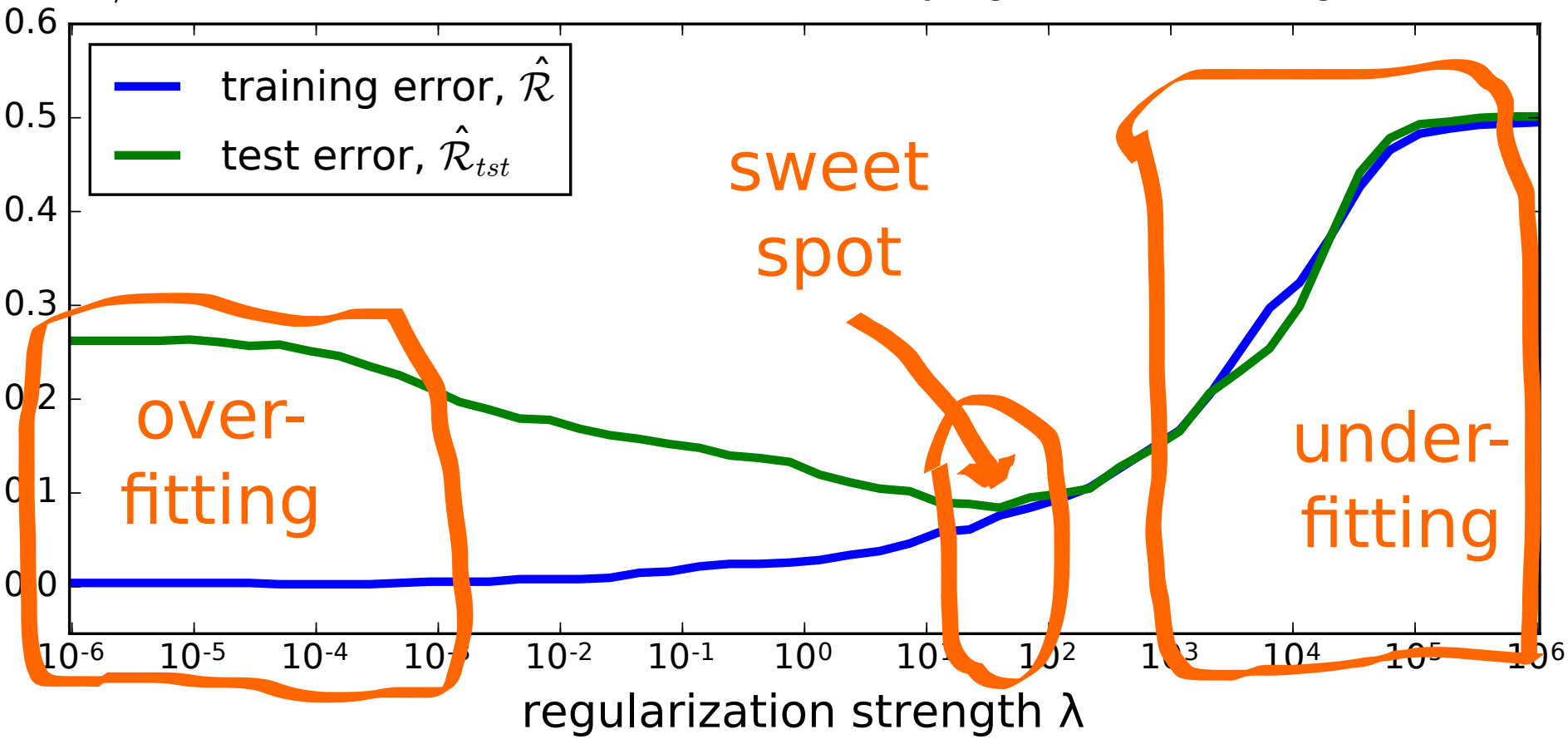


eye dataset: 737 examples for training, 736 examples for evaluation

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 minimizer of J_λ with varying amounts of regularization:



eye dataset: 737 examples for training, 736 examples for evaluation

Implicit regularization

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

Gradient descent optimization

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

Implicit regularization methods modify these steps, e.g.

- early stopping
- weight decay
- data jittering
- dropout

Gradient descent optimization with early stopping

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

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 should not reach extreme values
- T hyperparameter controls trade-off:
  - large T: parameters approach risk minimizer $\rightarrow$ risk of overfitting
  - small T: parameters stay close to initialization $\rightarrow$ risk of underfitting

Implicit regularization: weight decay

Gradient descent optimization with weight decay

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

Implicit regularization: weight decay

Gradient descent optimization with weight decay

- initialize $\theta^{(0)}$
- **for** $t = 1, 2, \ldots$
- $\quad \theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
- $\quad \theta^{(t)} \leftarrow \gamma \theta^{(t)}$ 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 $\rightarrow$ risk of overfitting
  - $\theta^{(t)} \leftarrow \gamma \theta^{(t)}$ pulls towards 0 $\rightarrow$ risk of underfitting
- convergence: both effects balance out $\rightarrow$ trade-off controlled by η_t, γ

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 $\rightarrow$ hopefully less overfitting
  (also possible: just create large training set of jittered examples in the beginning)
- problem: coming up with perturbations needs *domain knowledge*

Gradient descent optimization with dropout

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

Gradient descent optimization with dropout

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

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.

Model Selection

Which of model class?

- support vector machine, boosting?
- convolutional neural network? how many and which layers?

Which of data representation?

- pixel intensities, SIFT features, pretrained CNN features

What regularizers to apply? And how much?

Which of model class?

- support vector machine, boosting?
- convolutional neural network? how many and which layers?

Which of data representation?

- pixel intensities, SIFT features, pretrained CNN features

What regularizers to apply? And how much?

Model Selection

Which of model class?

- support vector machine, boosting?
- convolutional neural network? how many and which layers?

Which of data representation?

- pixel intensities, SIFT features, pretrained CNN features

What regularizers to apply? And how much?

Model Selection

Model selection is a difficult (and often underestimated) problem:

- we can't decide based on training error: we won't catch overfitting
- we can't decide based on test error: if we use test data to select (hyper)parameters, we're overfitting to the test set and the result is not a good estimate of true risk anymore

We want to evaluate a model on different data than the data it was trained on, but not the test set.

$\rightarrow$ emulate the disjoint train/test split using only the training data

Model selection with a validation set

Given: training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, possible hyperparameters $A = \{\eta_1, \ldots, \eta_m\}$ (including, e.g., which model class to use).

- Split available training data in to disjoint *real train* and *validation* set

$$S = S_{\text{trn}} \dot{\cup} S_{\text{val}}$$

- for all $\eta \in A$:
- $\quad f^\eta \leftarrow$ train a model with hyperparameters η using data S_{trn}
- $\quad E^\eta_{\text{val}} \leftarrow$ evaluate model f^η on set S_{val}
- $\eta^* = \operatorname{argmin}_{\eta \in A} E^\eta_{\text{val}}$ (select most promising hyperparameters)
- optionally: retrain model with hyperparameters η^* on complete S

Illustration: learning polynomials (of different degrees)

Illustration: learning polynomials (of different degrees)

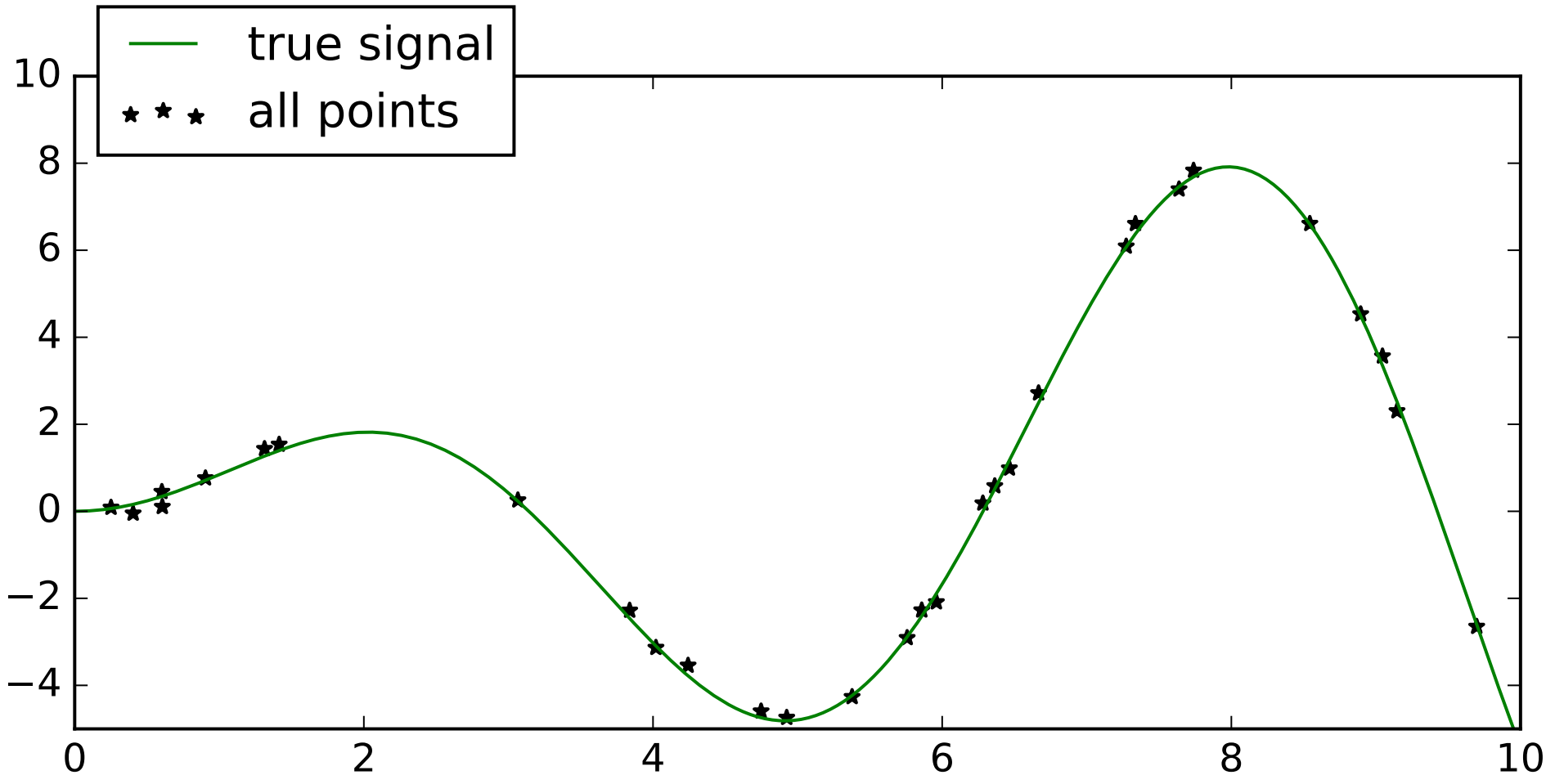


Illustration: learning polynomials (of different degrees)

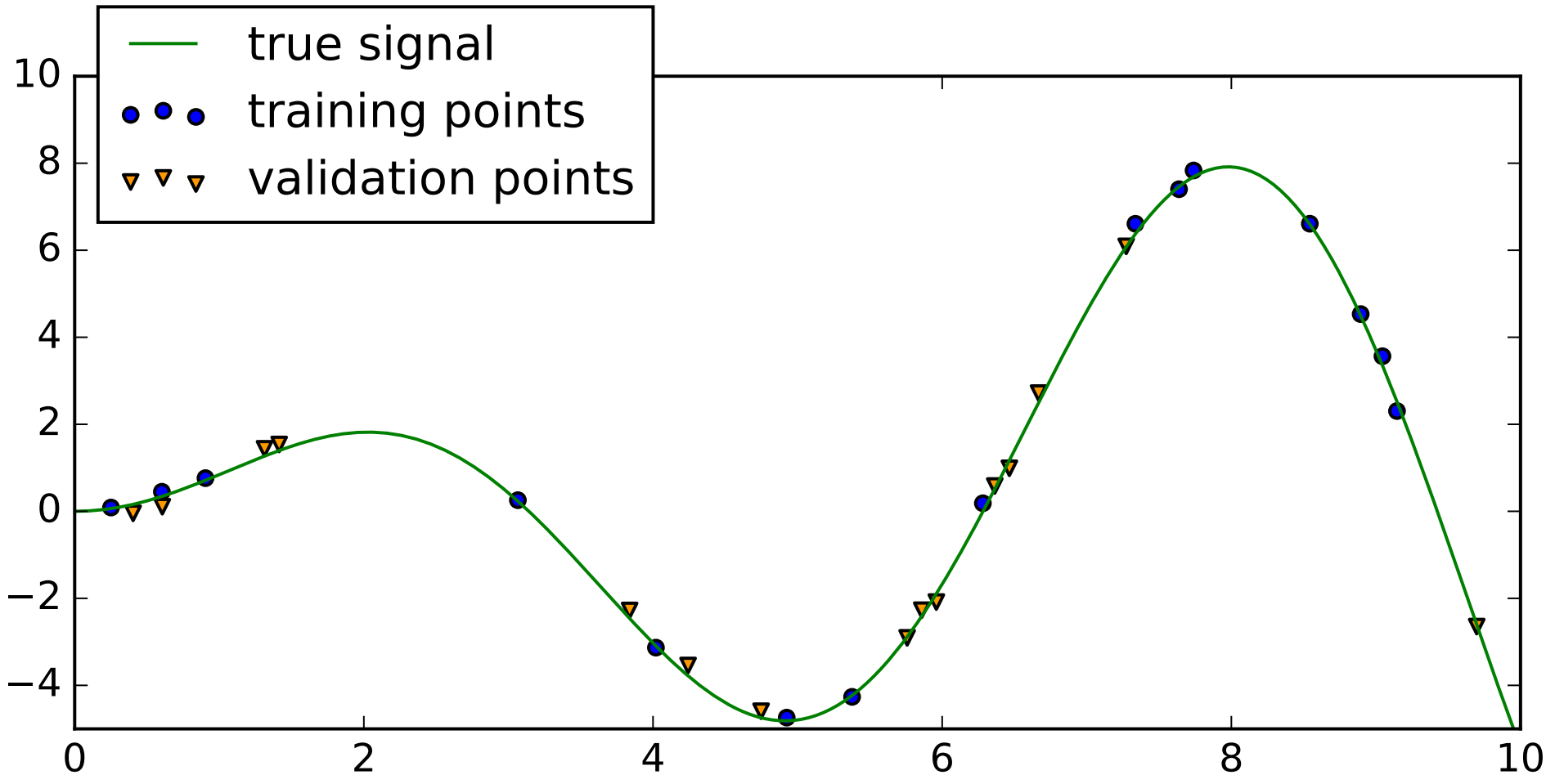


Illustration: learning polynomials (of different degrees)

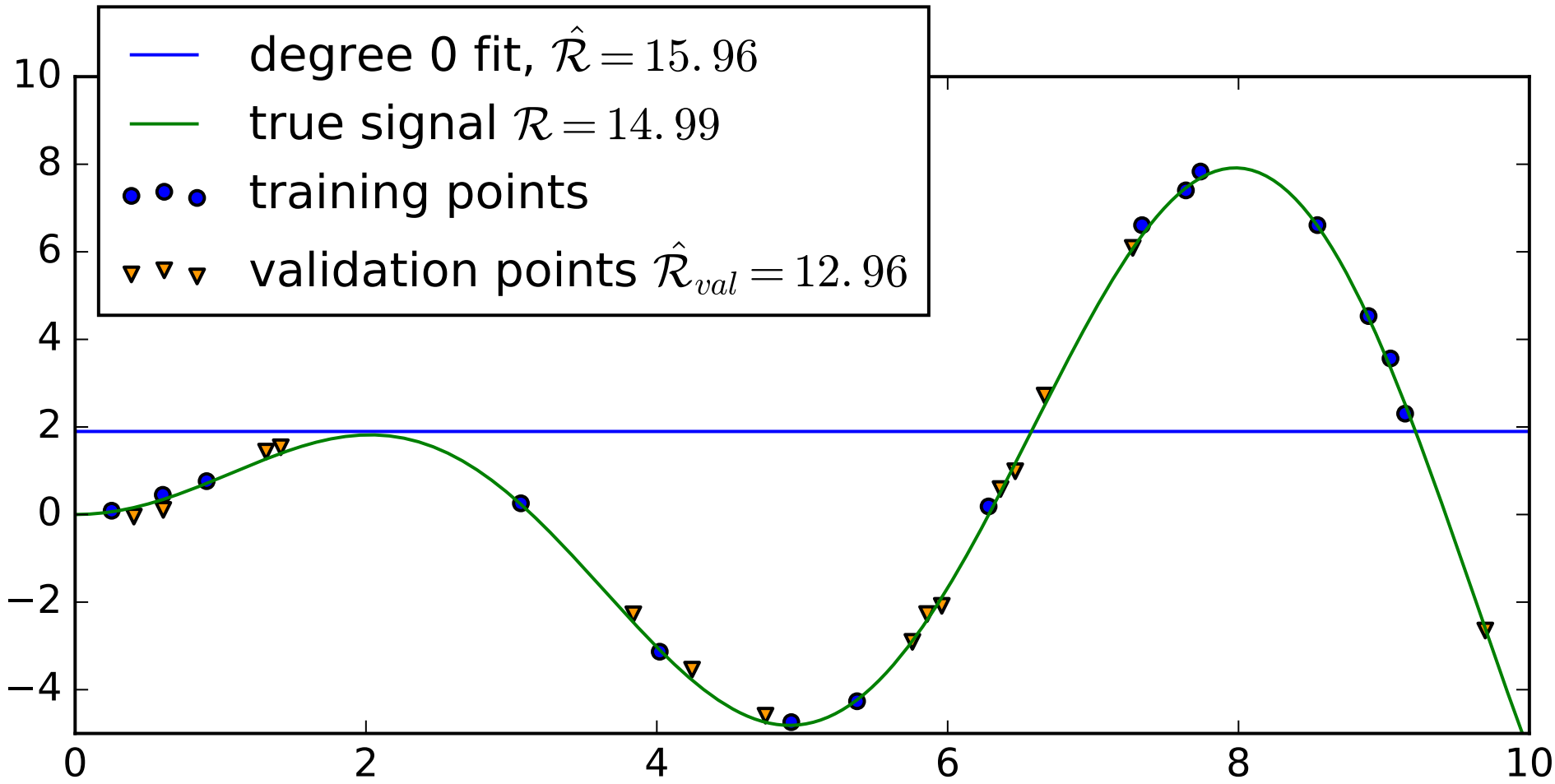


Illustration: learning polynomials (of different degrees)

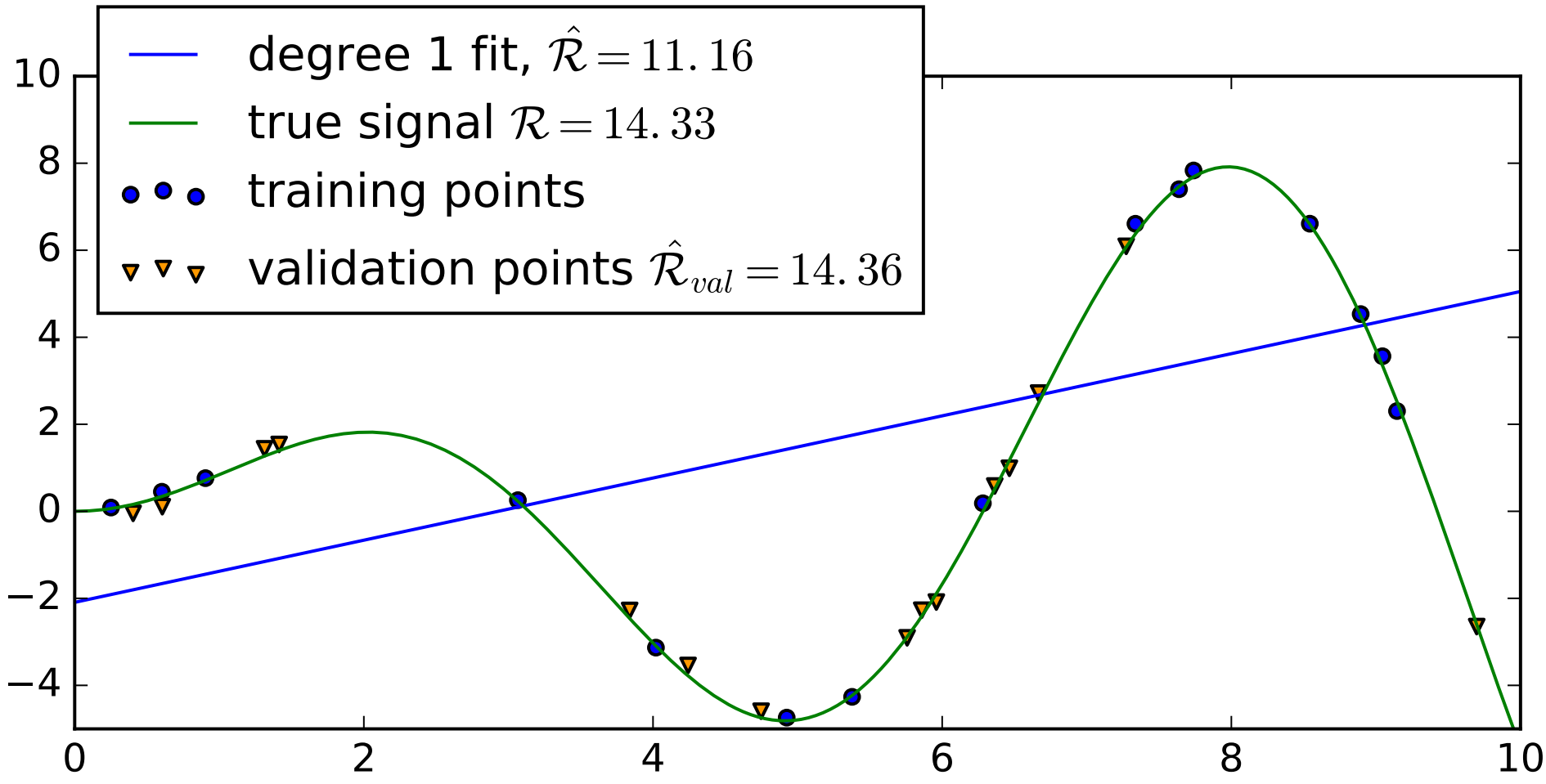


Illustration: learning polynomials (of different degrees)

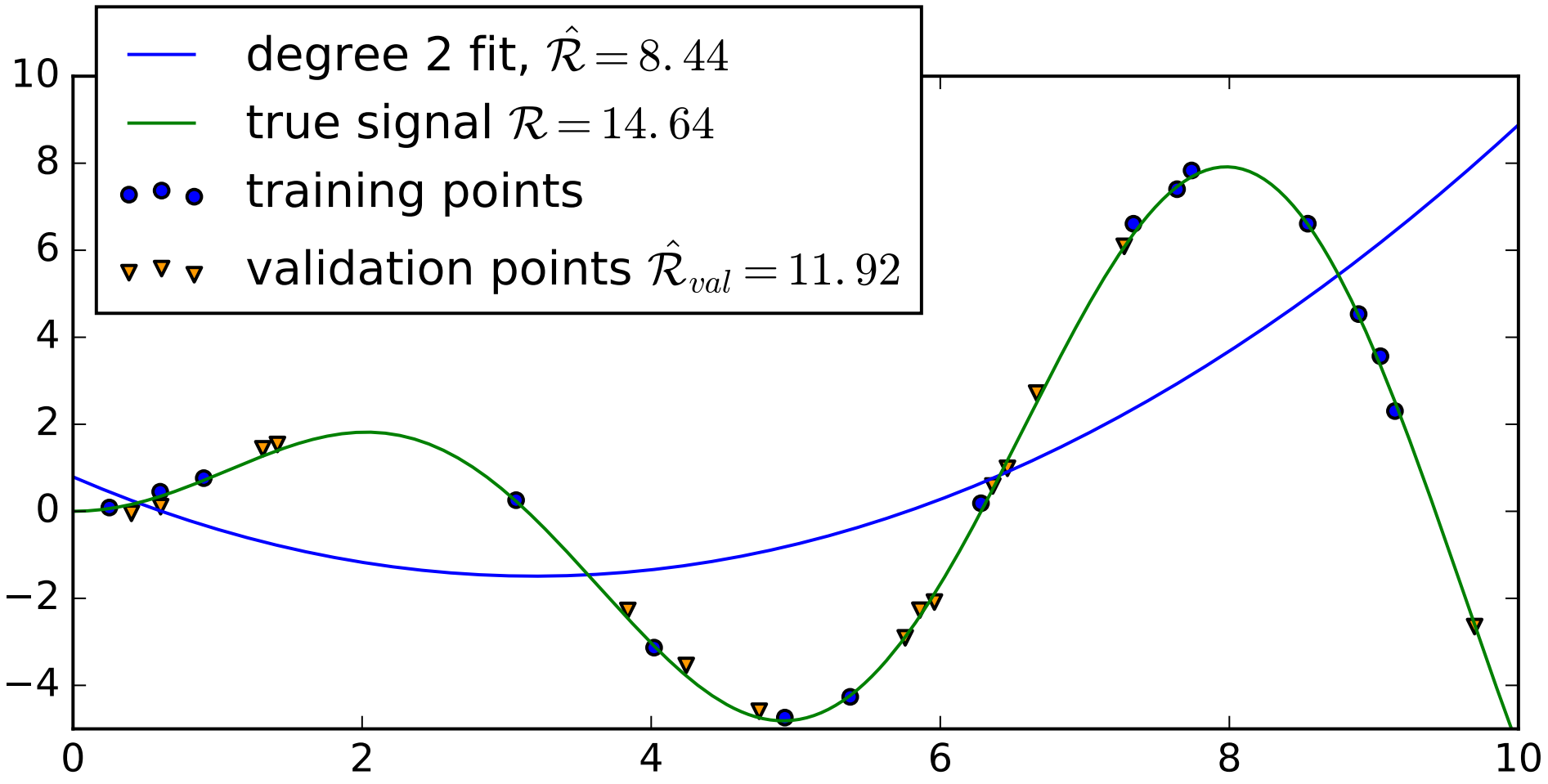


Illustration: learning polynomials (of different degrees)

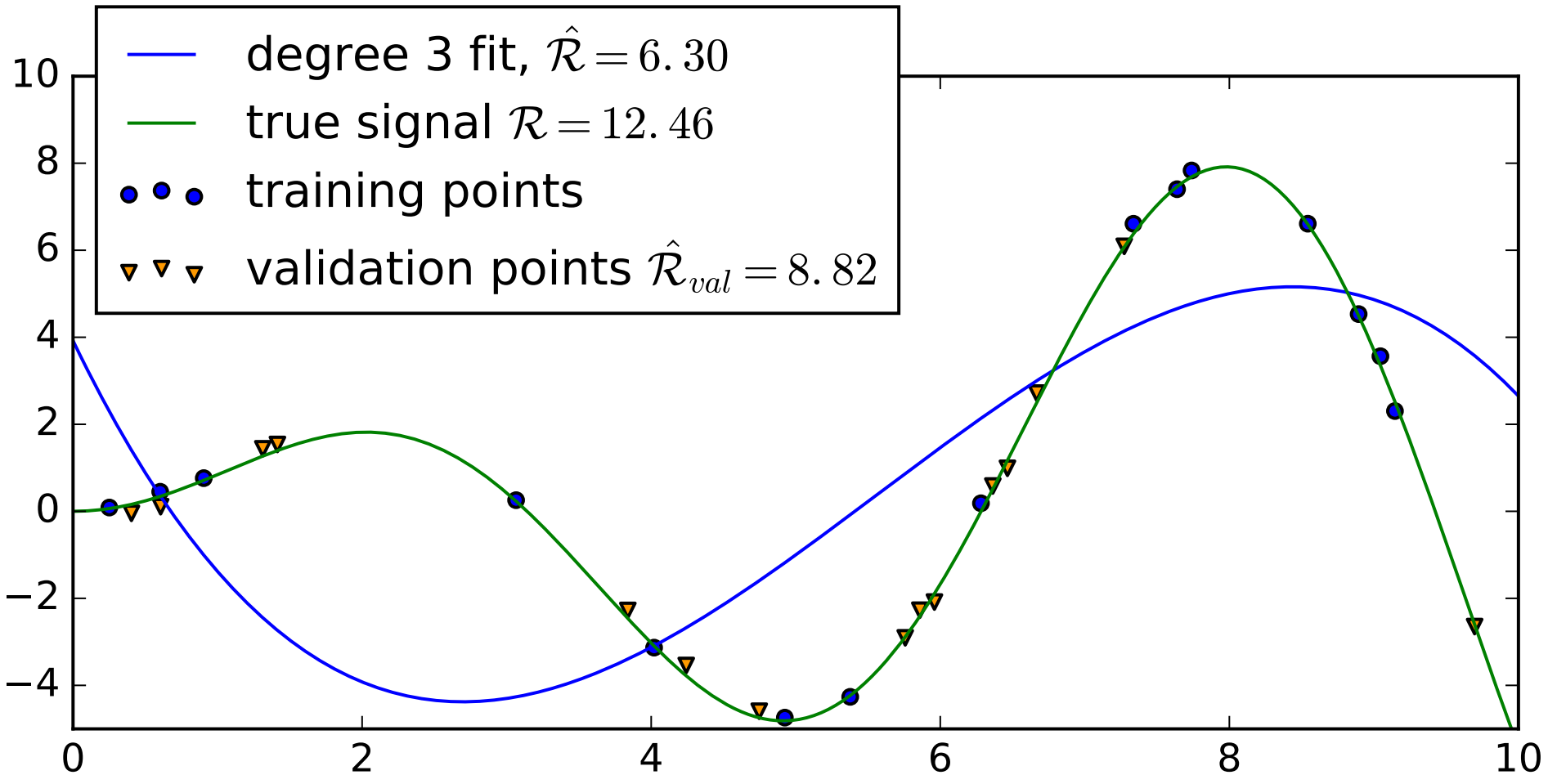


Illustration: learning polynomials (of different degrees)

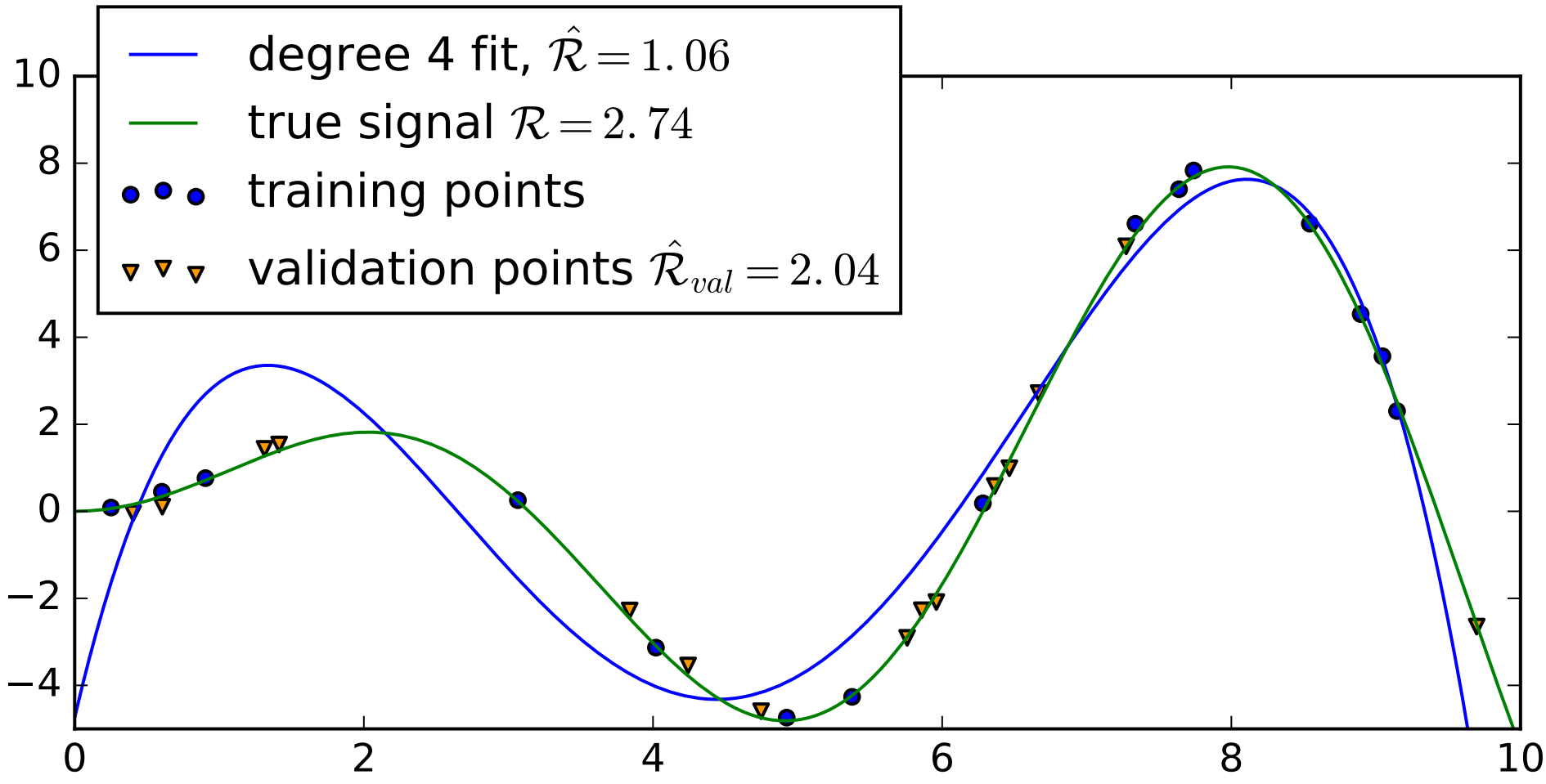


Illustration: learning polynomials (of different degrees)

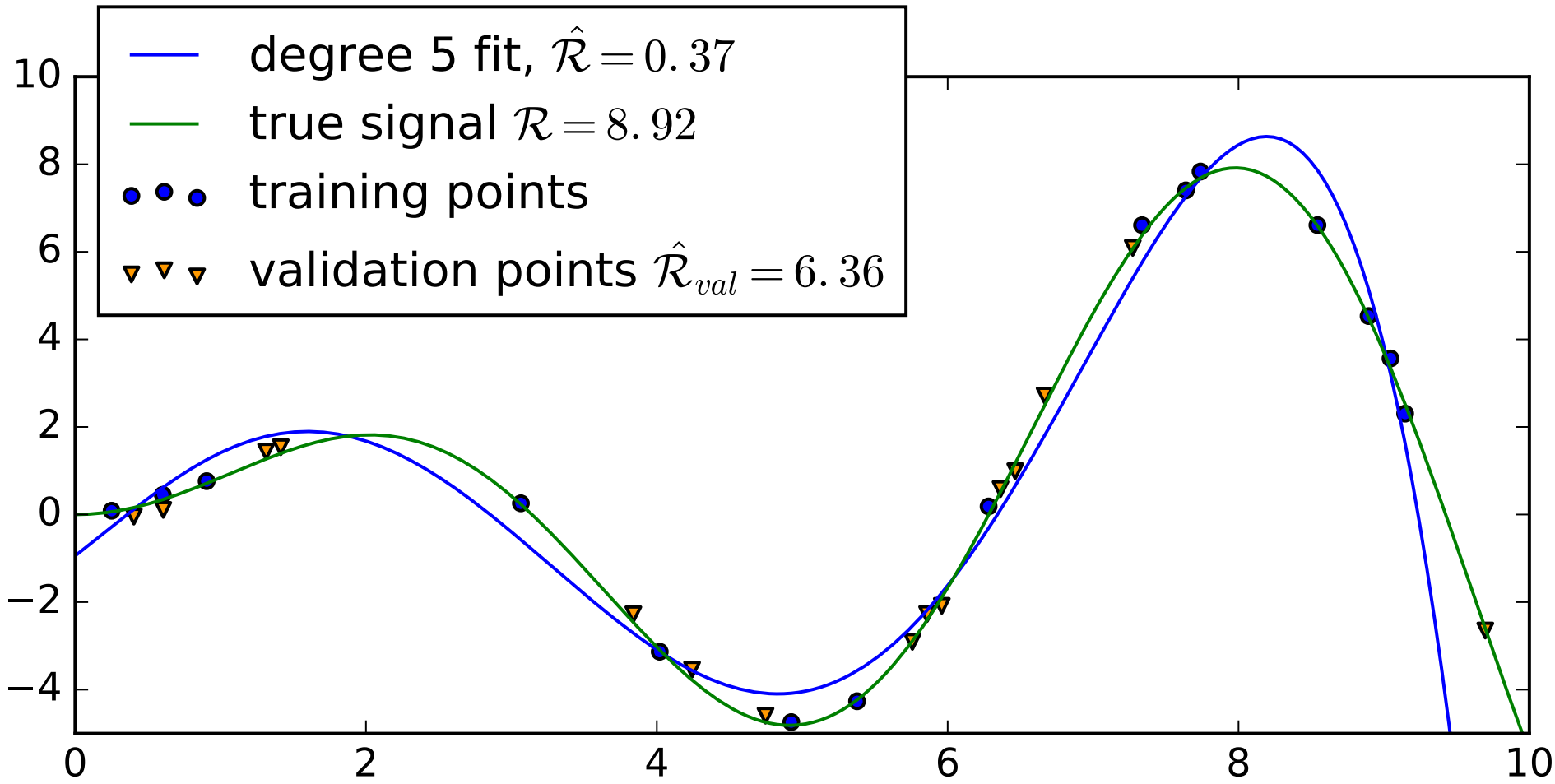


Illustration: learning polynomials (of different degrees)

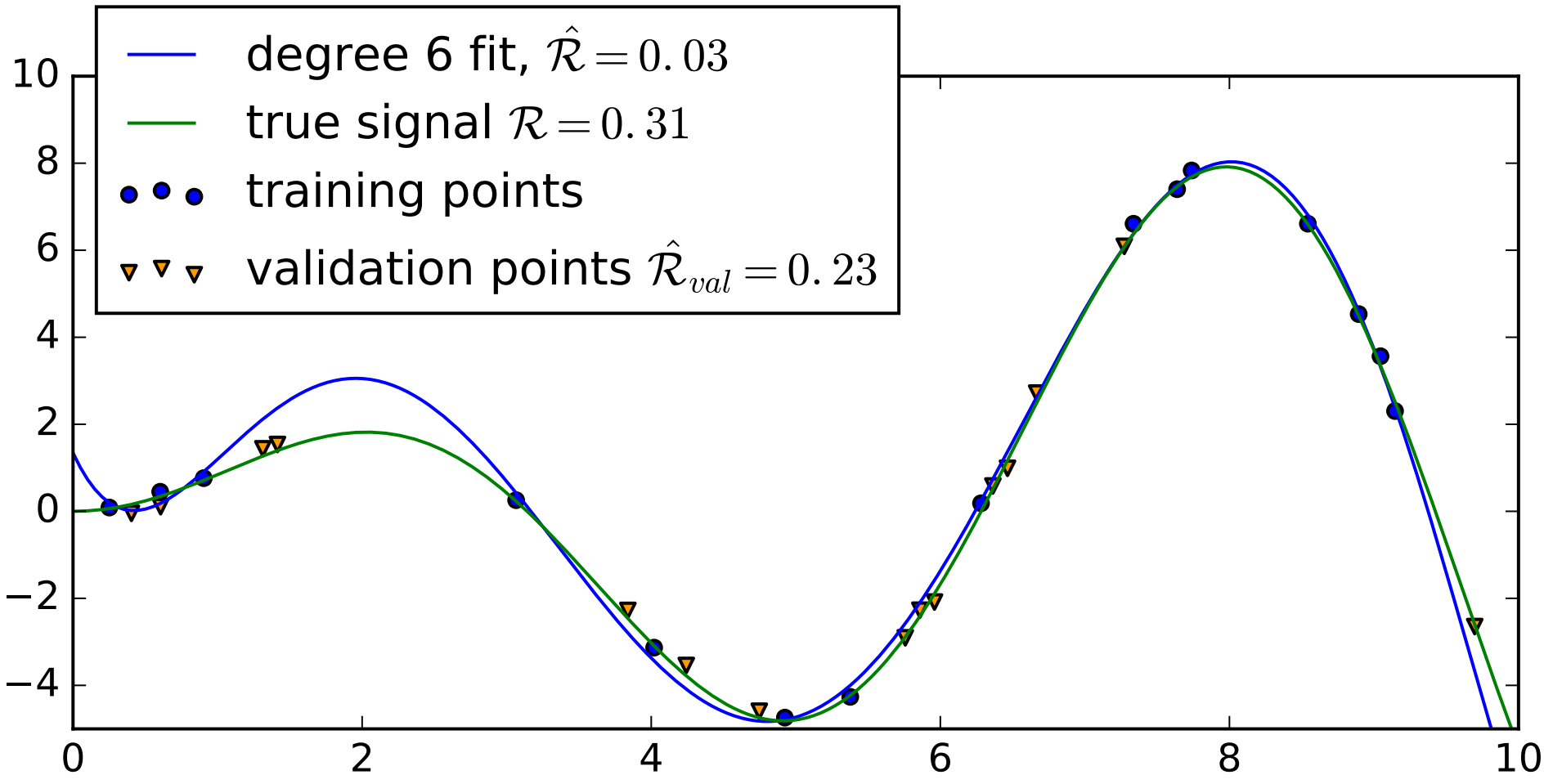


Illustration: learning polynomials (of different degrees)

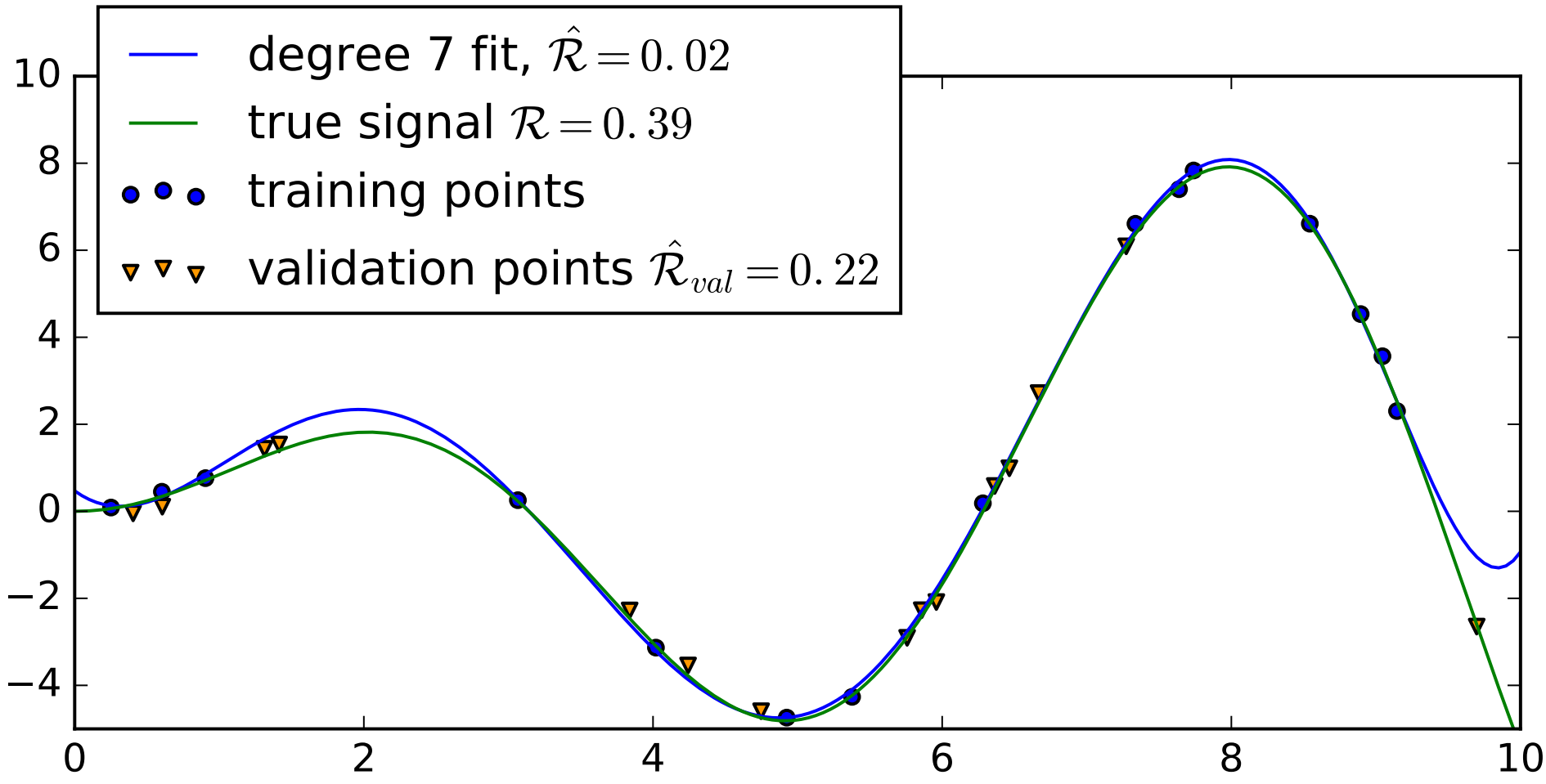


Illustration: learning polynomials (of different degrees)

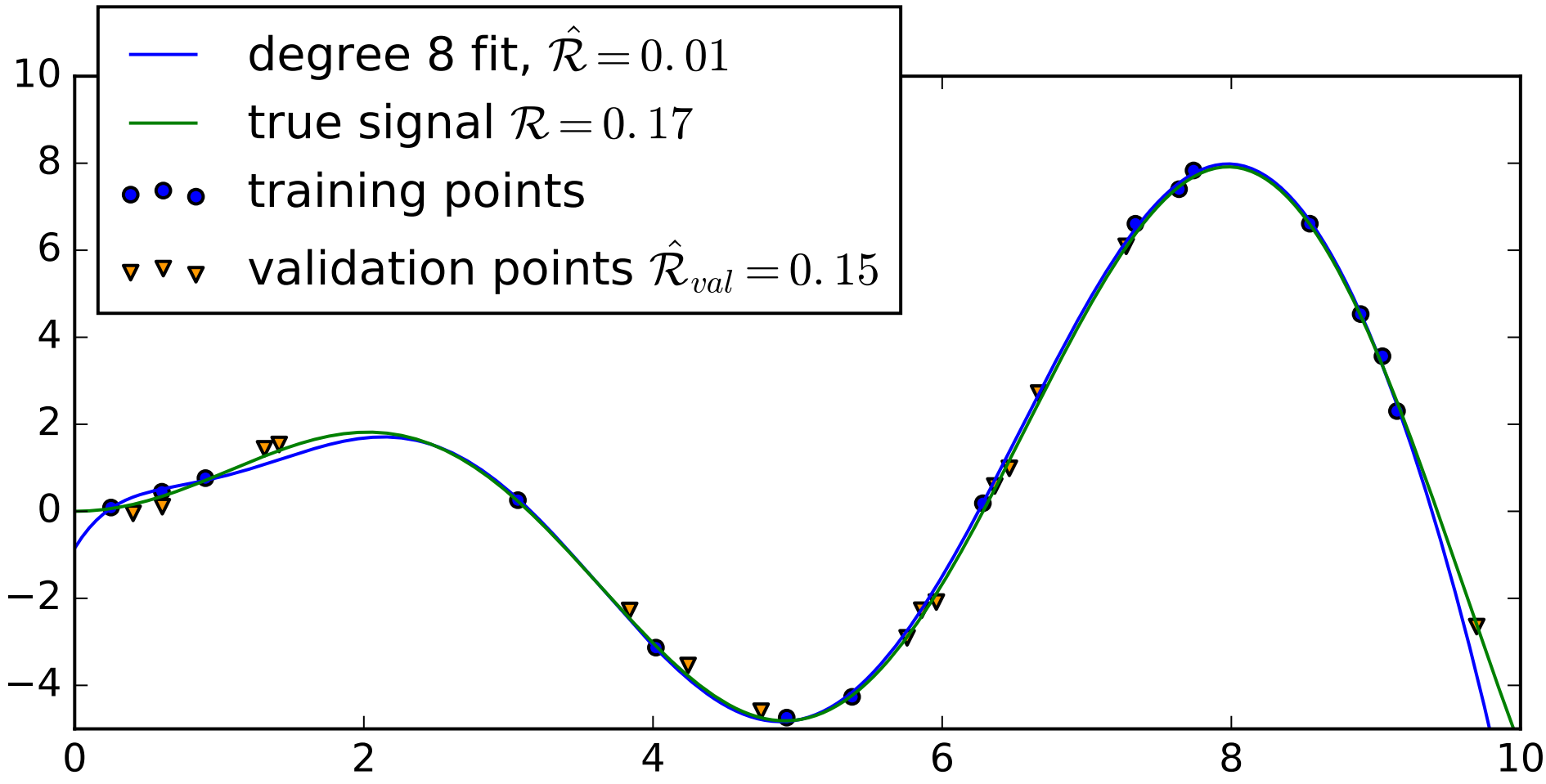


Illustration: learning polynomials (of different degrees)

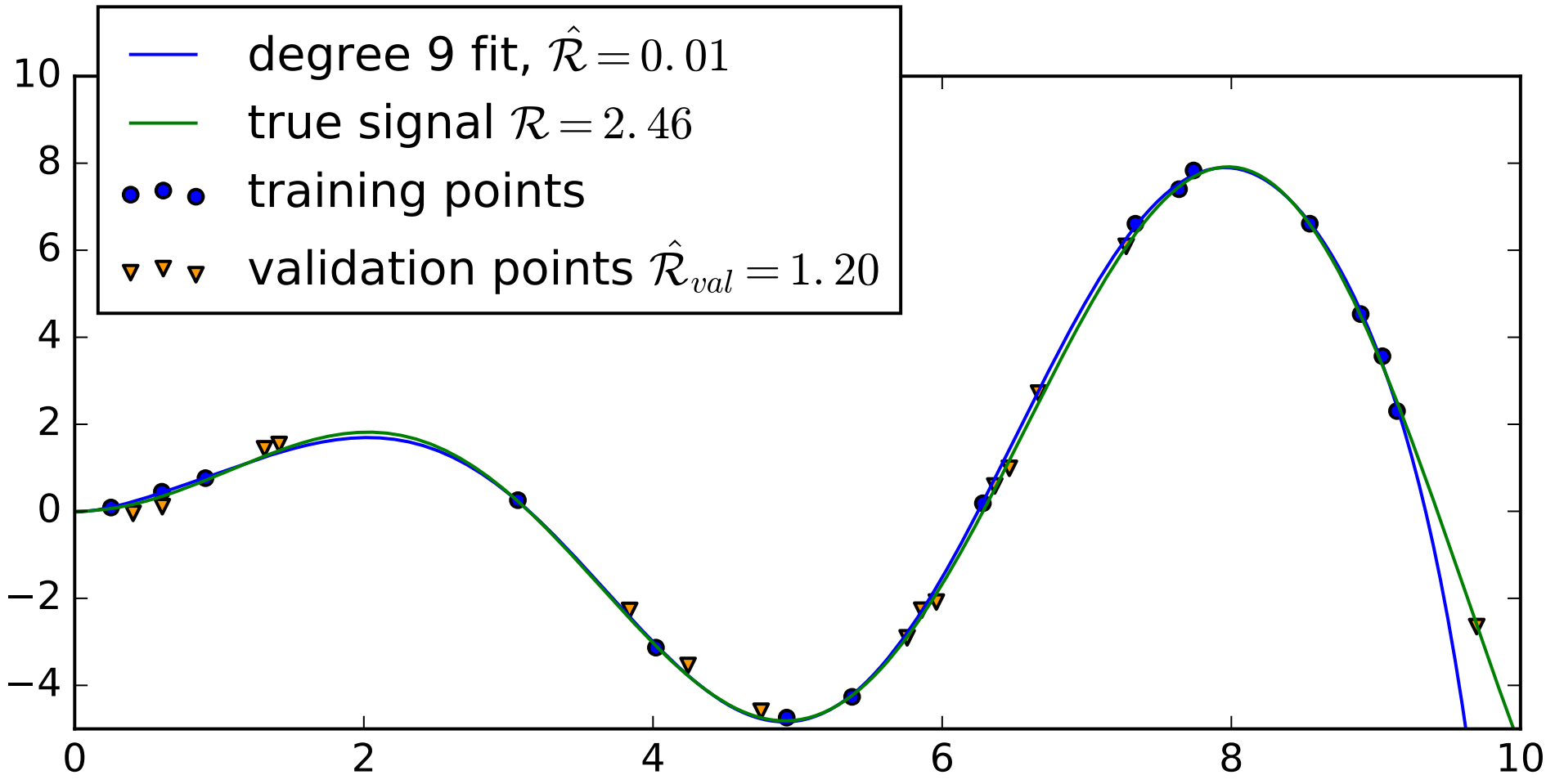


Illustration: learning polynomials (of different degrees)

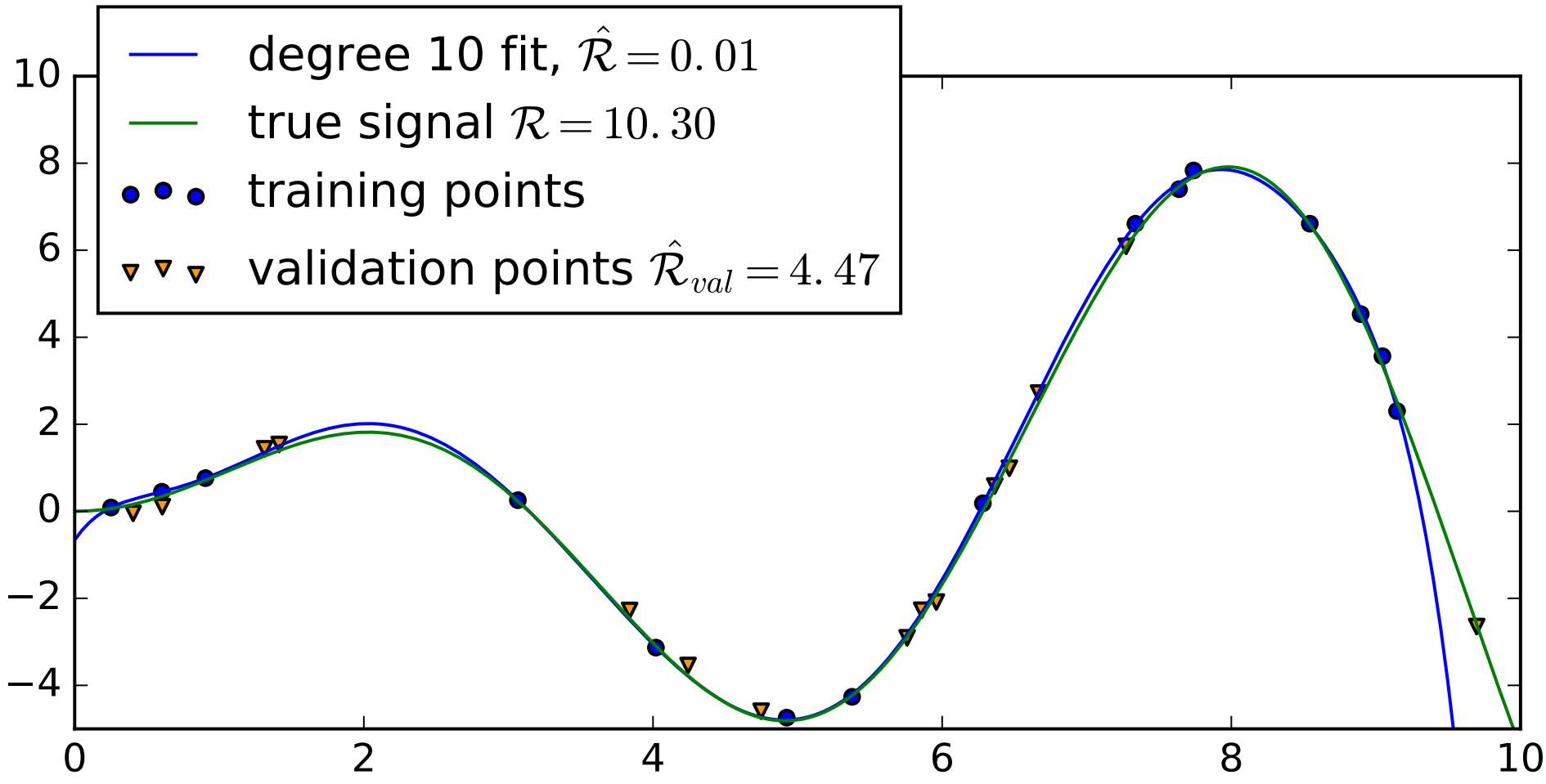


Illustration: learning polynomials (of different degrees)

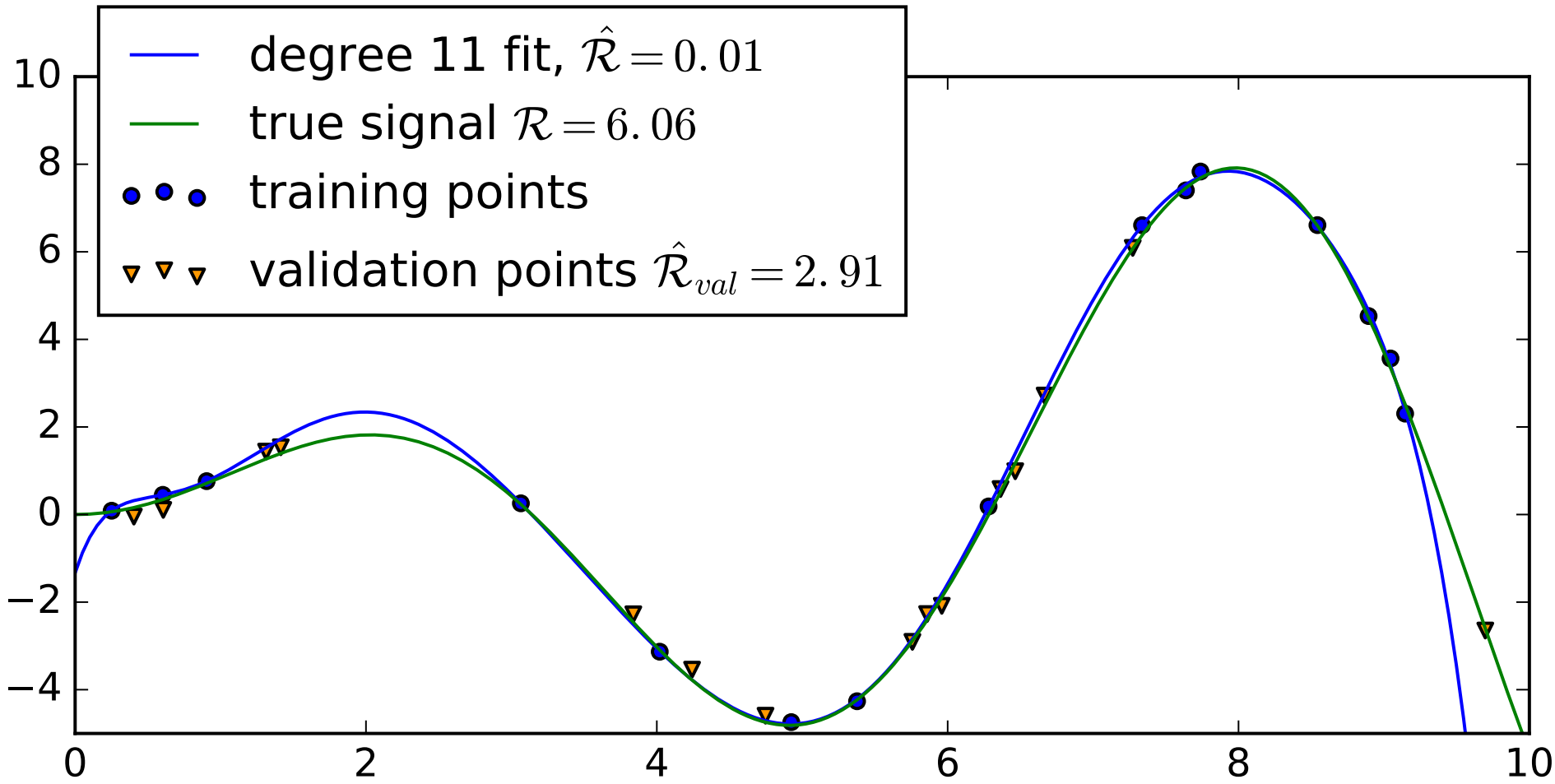


Illustration: learning polynomials (of different degrees)

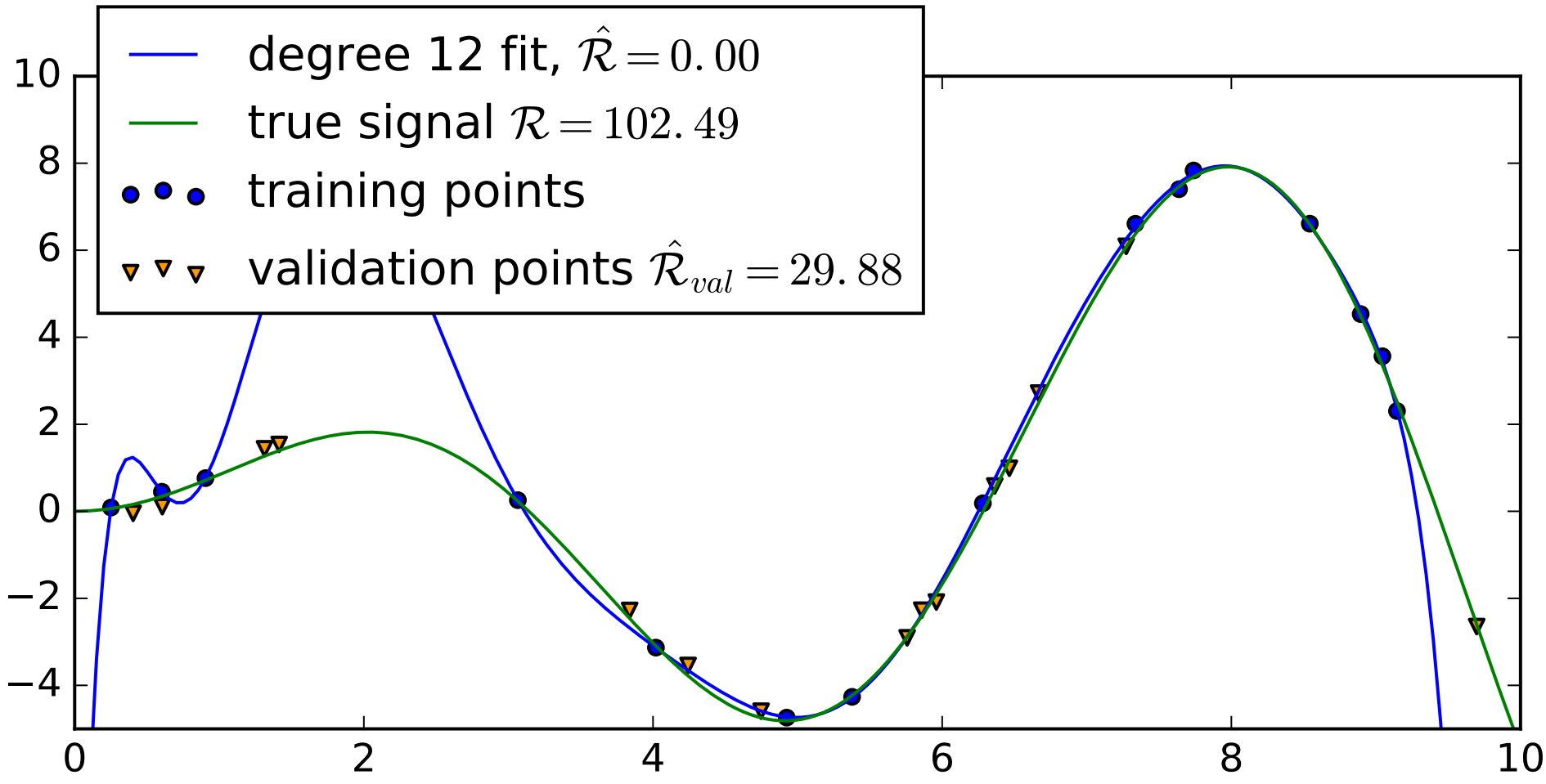


Illustration: learning polynomials (of different degrees)

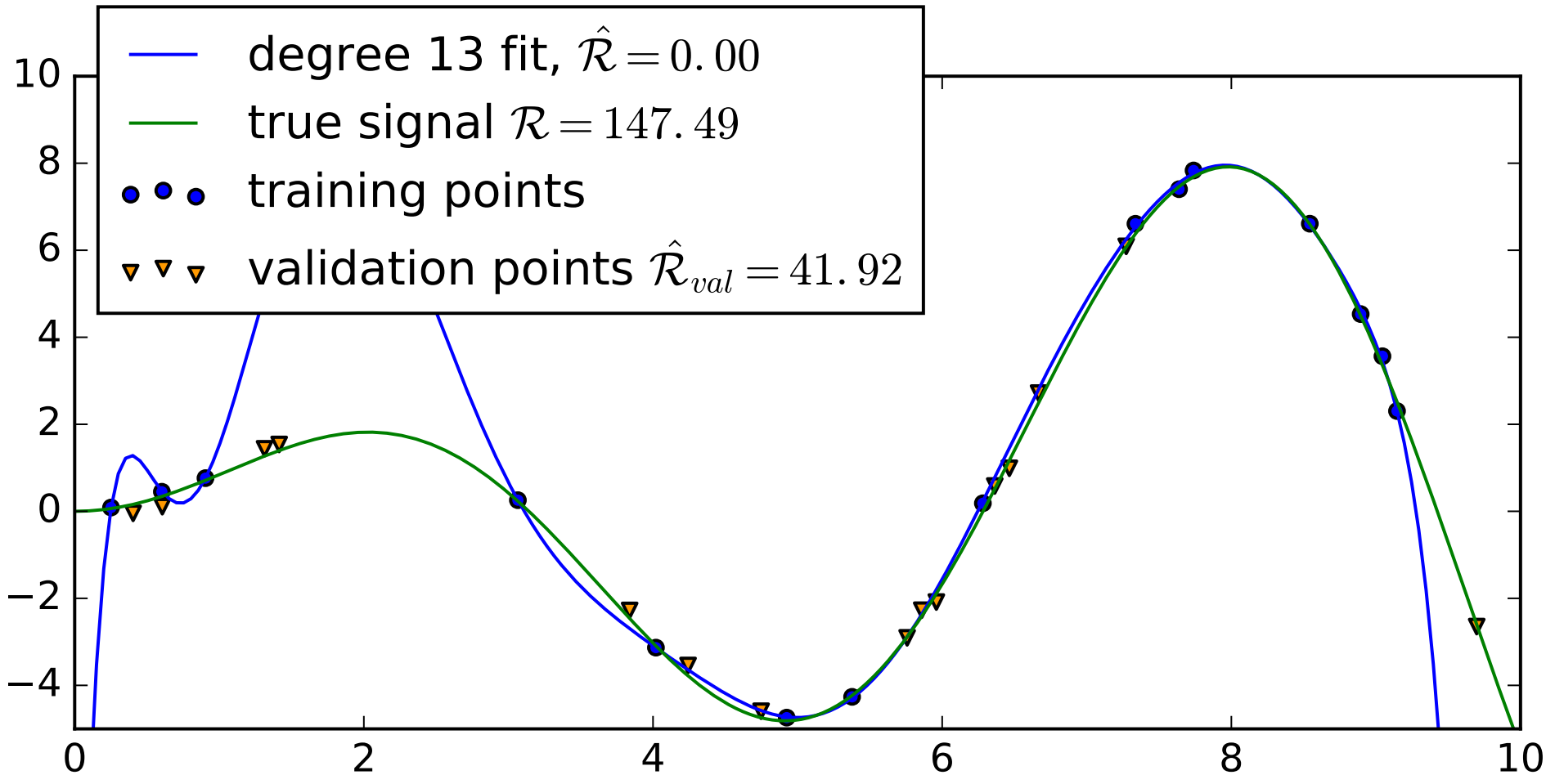
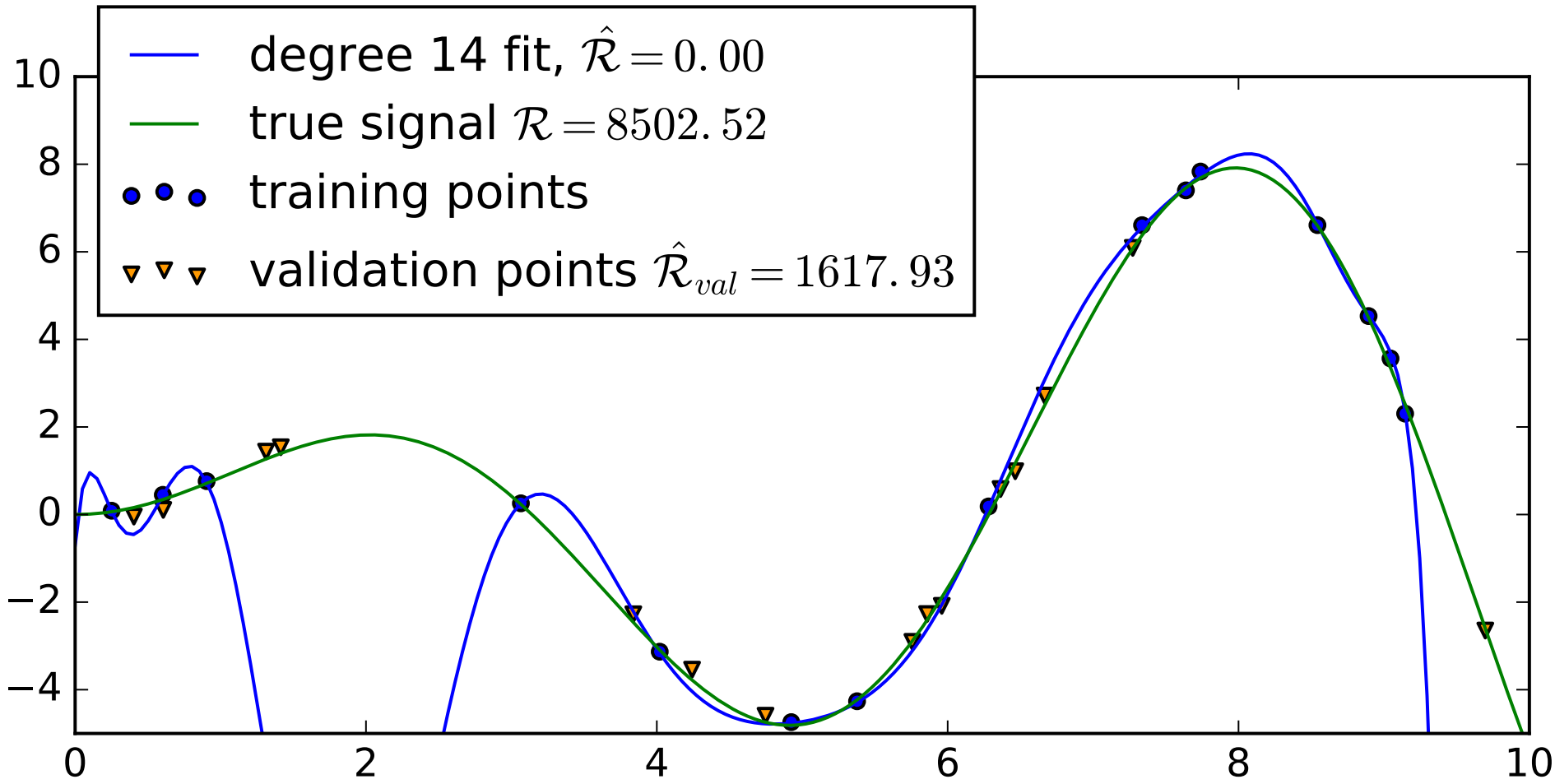
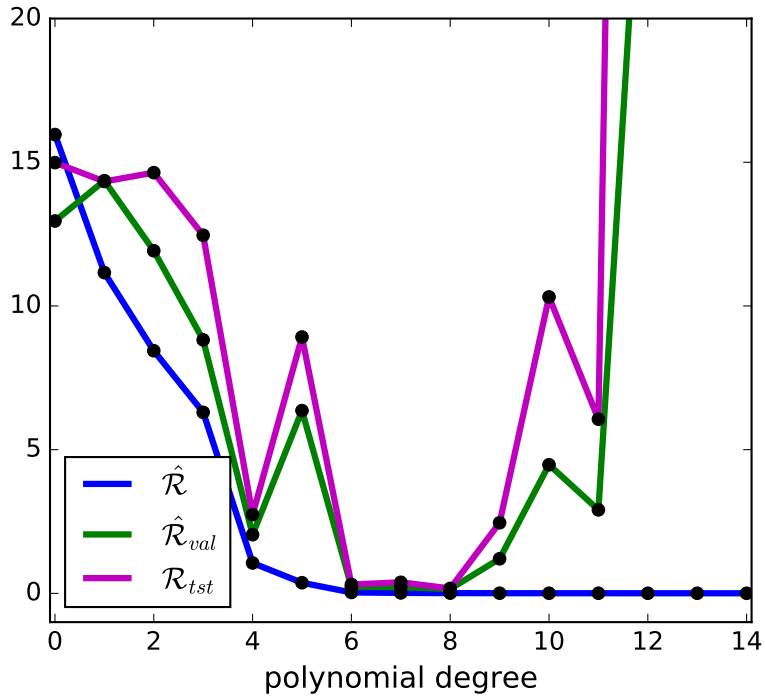


Illustration: learning polynomials (of different degrees)



degree	$\hat{\mathcal{R}}$	$\hat{\mathcal{R}}_{val}$	$\mathcal{R}$
0	15.96	12.96	14.99
1	11.16	14.36	14.33
2	8.44	11.92	14.64
3	6.30	8.82	12.46
4	1.06	2.04	2.74
5	0.37	6.36	8.92
6	0.03	0.23	0.31
7	0.02	0.22	0.39
8	0.01	0.15	0.17
9	0.01	1.20	2.46
10	0.01	4.48	10.31
11	0.01	2.91	6.06
12	0.00	30.34	104.11
13	0.00	40.87	142.73
14	0.00	1622.37	8494.42



We want to evaluate the model on different data than the data it was trained on, but not the test set.

Model selection by K-fold cross-validation (typically: $K = 5$ or $K = 10$)

Given: training set S, possible hyperparameters A

- split disjointly $S = S_1 \dot\cup S_2 \dot\cup \ldots \dot\cup S_K$
- **for all** $\eta \in A$
- **for** $k = 1, \ldots, K$
- $f_k^\eta \leftarrow$ train model on $S \setminus S_k$ using hyperparameters η
- $E_k^\eta \leftarrow$ evaluate model f_k^η on set S_k
- $E_{\text{CV}}^\eta \leftarrow \frac{1}{K} \sum_{k=1}^K E_k^\eta$
- $\eta^* = \operatorname{argmin}_{\eta \in A} E_{\text{CV}}^\eta$ (select most promising hyperparameters)
- retrain model with hyperparameters η^* on complete S

More robust than just a validation set, computationally more expensive.

From infinite to finite hyperparameter set

Typically, finitely many choices for model classes and feature:

- we can try them all

For hyperparameters typically infinitely many choices:

- stepsizes $\eta_t \in \mathbb{R}$ for $t = 1, 2, \ldots,$ even single fixed stepsize $\eta \in \mathbb{R}$
- regularization constants $\lambda \in \mathbb{R}$
- number of iterations $T \in \mathbb{N}$
- dropout ratio $p \in (0, 1)$

Discretize in a reasonable way, e.g.

- stepsizes: exponentially scale, e.g. $\eta \in \{10^{-8}, 10^{-7}, \ldots, 1\}$
- regularization: exponentially, e.g. $\lambda \in \{2^{-20}, 2^{-19}, \ldots, 2^{20}\}$
- iterations: linearly , e.g. $T \in \{1, 5, 10, 15, 20, \ldots\}$ (or use E_{val})
  (depends on size/speed of each iteration, could also be $T \in \{1000, 2000, 3000, \ldots\}$)
- dropout: linearly, e.g. $p \in \{0.1, 0.2, \ldots, 0.9\}$

If model selection picks value at boundary of range, choose a larger range.

Selecting multiple hyperparameters: $\eta_1 \in A_1, \eta_2 \in A_2, \ldots, \eta_J \in A_J$

Selecting multiple hyperparameters: $\eta_1 \in A_1, \eta_2 \in A_2, \ldots, \eta_J \in A_J$

Grid search (good, but rarely practical for more than $J > 2$)

- $\eta_{\text{total}} = (\eta_1, \ldots, \eta_K)$, $A_{\text{total}} = A_1 \times A_2 \times \cdots \times A_J$
- try all $|A_1| \times |A_2| \times \cdots \times |A_J|$ combinations

Selecting multiple hyperparameters: $\eta_1 \in A_1, \eta_2 \in A_2, \ldots, \eta_J \in A_J$

Grid search (good, but rarely practical for more than $J > 2$)

- $\eta_{\text{total}} = (\eta_1, \ldots, \eta_K)$, $A_{\text{total}} = A_1 \times A_2 \times \cdots \times A_J$
- try all $|A_1| \times |A_2| \times \cdots \times |A_J|$ combinations

Greedy search (sometimes practical, but suboptimal)

- for $j = 1, \ldots, J$:
- $\eta_j^* \leftarrow$ pick $\eta_j \in A_j$, with η_k for $k > j$ fixed at a reasonable default

Problem: what is a 'reasonable default'?

Selecting multiple hyperparameters: $\eta_1 \in A_1, \eta_2 \in A_2, \ldots, \eta_J \in A_J$

Grid search (good, but rarely practical for more than $J > 2$)

- $\eta_{\text{total}} = (\eta_1, \ldots, \eta_K)$, $A_{\text{total}} = A_1 \times A_2 \times \cdots \times A_J$
- try all $|A_1| \times |A_2| \times \cdots \times |A_J|$ combinations

Greedy search (sometimes practical, but suboptimal)

- for $j = 1, \ldots, J$:
- $\eta_j^* \leftarrow$ pick $\eta_j \in A_j$, with η_k for $k > j$ fixed at a reasonable default

Problem: what is a 'reasonable default'?

Trust in a higher authority (avoid this!)

- set hyperparameters to values from the literature, e.g. $\lambda = 1$

Problem: this can fail horribly if the situation isn't completely identical.

Avoid models with many hyperparameters!

Summary

Step 1) Decide what exactly you want

- **inputs** x, **outputs** y, **loss** ℓ, **model class** f_θ (with hyperparameters)

Step 2) Collect and annotate data

- **collect and annotate data**, ideally i.i.d. from true distribution

Step 3) Model training

- **perform model selection** and **model training** on a training set

Step 4) Model evaluation

- **evaluate model** on test set (that is disjoint from training set)

Comparing yourself to others

You develop a new solution to an existing problem.

How to know if its good enough?

How do you know it is better than what was there before?

How do you convince **others** that it's good?

One of the cornerstones of science is that results must be reproducible:

Any person knowledgeable in the field should be able to repeat your experiments and get the same results.

Reproducibility

- reproducing the experiments requires access to the data
  - use public data and provide information where you got it from
  - if you have to use your own data, make it publicly available
- reproducing the experiments requires knowledge of all components
  - clearly describe all components used, not just the new ones
  - list all implementation details (ideally also release the source code)
  - list all (hyper)parameter values and how they were obtained

The goal of research is not getting papers accepted, but creating new knowledge.

Be your own method's harshest critic: you know best what's going on under the hood and what could have gone wrong in the process.

Scientific scrutiny

- is the solved problem really relevant?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, is the comparison *fair*?

Just that you invested a lot of work into a project doesn't mean it deserves to be published. A project must be able to *fail*, otherwise it's not research.

The goal of research is not getting papers accepted, but creating new knowledge.

Be your own method's harshest critic: you know best what's going on under the hood and what could have gone wrong in the process.

Scientific scrutiny

- is the solved problem really relevant?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, is the comparison *fair*?

Just that you invested a lot of work into a project doesn't mean it deserves to be published. A project must be able to *fail*, otherwise it's not research.

Repeated Experiments / Randomization

In the natural sciences, to show that an effect was not just random chance one uses repeated experiments.

In computer science, running the same code multiple times should give identical results, so on top we must use a form of randomization.

- different random splits of the data, e.g. for training / evaluation
- different random initialization, e.g. for neural network training

Which of these make sense depends on the situation

- for benchmark datasets, test set is usually fixed
- for convex optimization methods, initialization plays no role, etc.

We can think of accuracy on a validation/test set as repeated experiments:

- we apply a fixed model many times, each time to a single example
- does not work for other quality measures, e.g. *average precision*

Repeats

Having done repeated experiments, one reports either

- all outcomes, $r_1, r_2, \ldots, r_m$

or

- the mean of outcomes and standard deviations

$$\bar{r} \pm \sigma \qquad \text{for} \qquad \bar{r} = \frac{1}{m}\sum_j r_t, \qquad \sigma = \sqrt{\frac{1}{m}\sum_j (r_j - \bar{r})^2}$$

"If you try this once, where can you expect the result to lie?"

or

- the mean and standard error of the mean

$$\bar{r} \pm \mathrm{SE} \qquad \text{for} \qquad \bar{r} = \frac{1}{m}\sum_j r_t, \quad \mathrm{SE} = \frac{\sigma}{\sqrt{m}}$$

"If you try this many times, where can you expect the mean to lie?"

Illustration as figure with error bars:

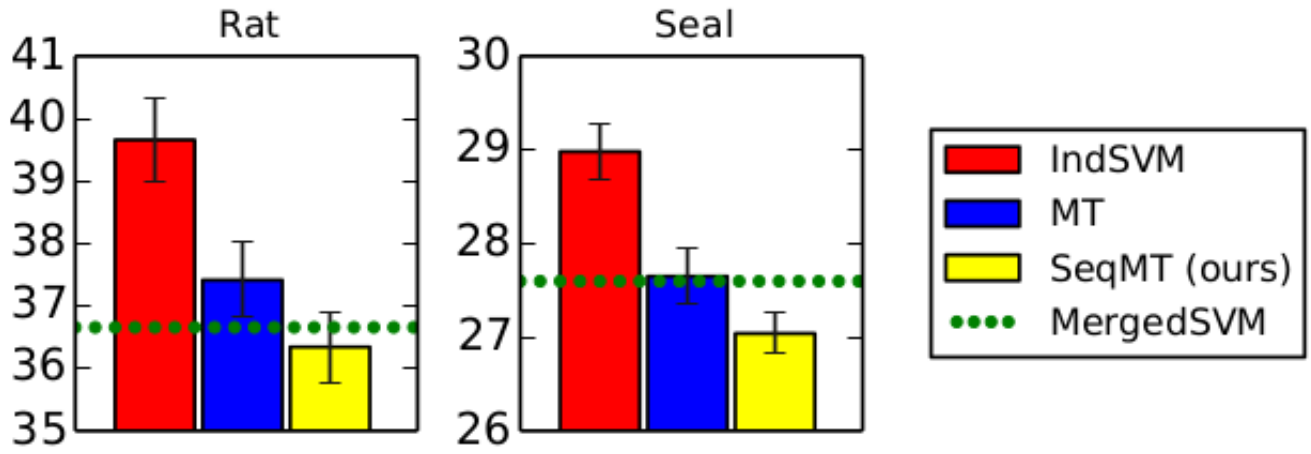


Illustration as table with error intervals:

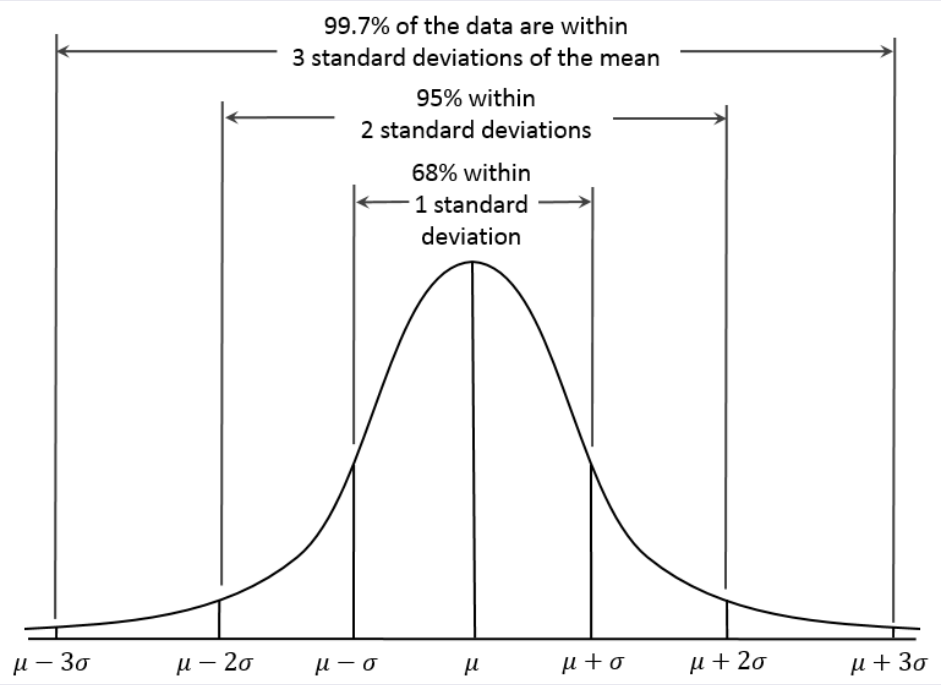
error rate	Chimpanzee	Giant panda
component 1	24.47 ± 0.42	20.02 ± 0.58
component 2	23.94 ± 0.32	19.44 ± 0.50
combination	23.86 ± 0.33	19.33 ± 0.52

Quick (and dirty) impression if results could be due to random chance:
do mean $\pm$ error bars overlap?

Normally distributed values

Usually, mean/std.dev. trigger association with Gaussian distribution:

- mode (most likely value) = median (middle value) = mean
- $\approx 68\%$ of observations lie within ± 1 std.dev. around the mean



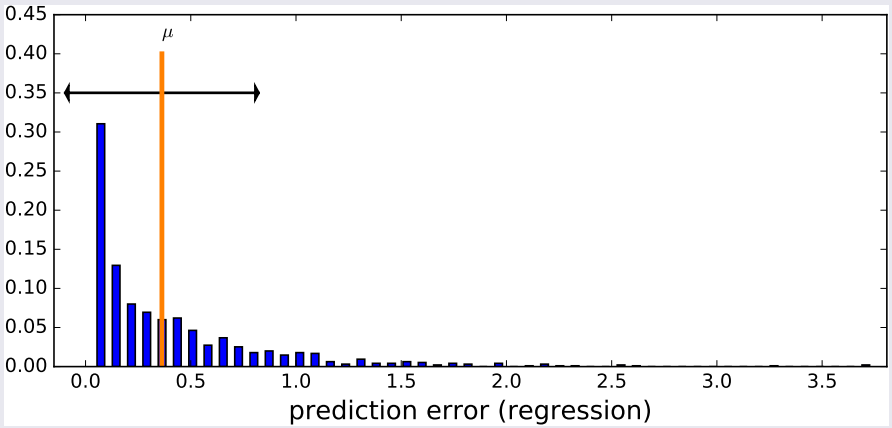
Overlapping error bar: high chance that order between methods flips between experiments

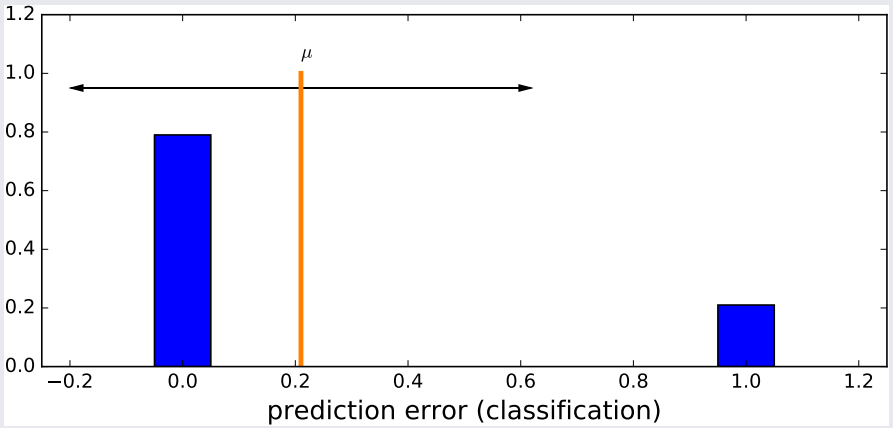
image: Dan Kernler - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=36506025

Other data

Distribution of errors can be very different from Gaussian

- Little or no data near mean value
- Mode, median and mean can be very different





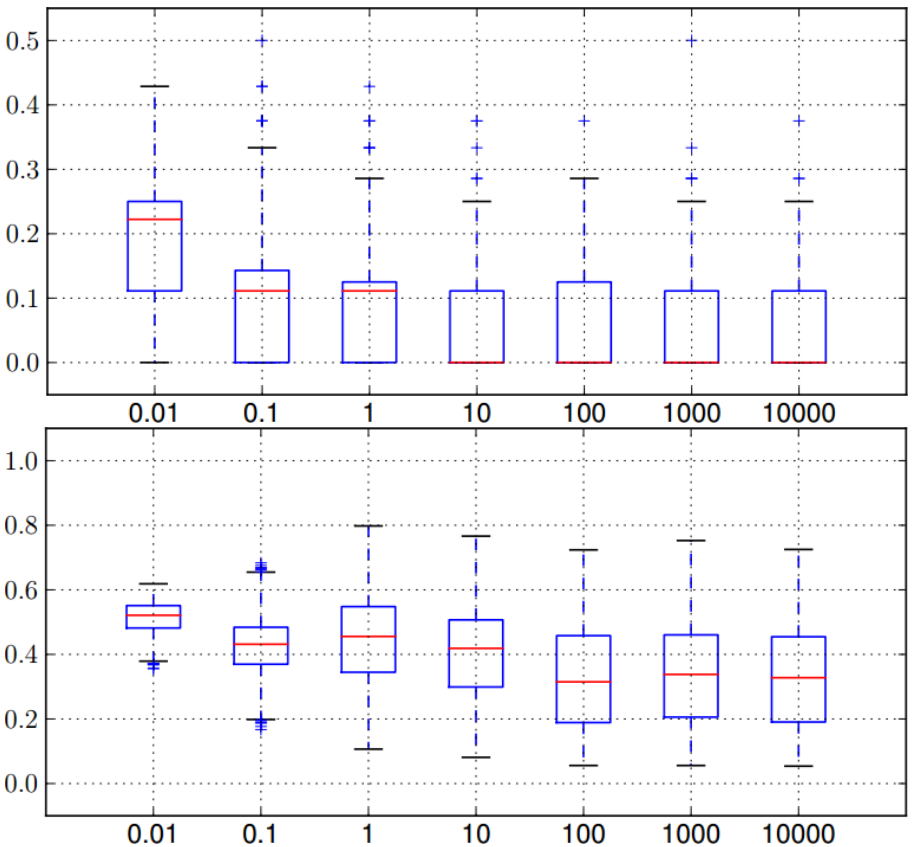
- less obvious how to interpret, e.g., what overlapping error bars mean
- ... but still not a good sign if they do

Note: comparing error bars correspond to an unpaired test

- methods could be tested on different data
- if all methods are tested on the same data, we can use a paired test and get stronger statements (see later...)

Box and Whisker Plot

Even more informative: **box and whisker plots** (short: box plot)



- red line: data median
- blue body: 25%–75% quantile
- whiskers: value range
- blue markers: outliers

sometimes additional symbols,
e.g. for the mean

Quiz: Randomness in Repeated Experiments

We can think of accuracy on a test set as repeated experiments: apply a fixed model many times, each time to a single example.

Which of these differences are real and which are random chance?

Quiz: Randomness in Repeated Experiments

We can think of accuracy on a test set as repeated experiments: apply a fixed model many times, each time to a single example.

Which of these differences are real and which are random chance?

- method 1: 71% accuracy
- method 2: 85% accuracy

Quiz: Randomness in Repeated Experiments

We can think of accuracy on a test set as repeated experiments: apply a fixed model many times, each time to a single example.

Which of these differences are real and which are random chance?

- method 1: 71% accuracy
- method 2: 85% accuracy

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

Quiz: Randomness in Repeated Experiments

We can think of accuracy on a test set as repeated experiments: apply a fixed model many times, each time to a single example.

Which of these differences are real and which are random chance?

- method 1: 71% accuracy
- method 2: 85% accuracy

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

- method 1: 40% accuracy on 1000 examples
- method 2: 60% accuracy on 1000 examples

Quiz: Randomness in Repeated Experiments

We can think of accuracy on a test set as repeated experiments: apply a fixed model many times, each time to a single example.

Which of these differences are real and which are random chance?

- method 1: 71% accuracy
- method 2: 85% accuracy

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

- method 1: 40% accuracy on 1000 examples
- method 2: 60% accuracy on 1000 examples

- method 1: 99.77% accuracy on 10000 examples
- method 2: 99.79% accuracy on 10000 examples

Quiz: Randomness in Repeated Experiments

We can think of accuracy on a test set as repeated experiments: apply a fixed model many times, each time to a single example.

Which of these differences are real and which are random chance?

- method 1: 71% accuracy
- method 2: 85% accuracy

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

- method 1: 40% accuracy on 1000 examples
- method 2: 60% accuracy on 1000 examples

- method 1: 99.77% accuracy on 10000 examples
- method 2: 99.79% accuracy on 10000 examples

- method 1: 0.737 average precision on 7518 examples
- method 2: 0.713 average precision on 7518 examples

Let's analyze...

- method 1: 71% accuracy
- method 2: 85% accuracy

Averages alone do not contain enough information...

- could be 5/7 correct versus 6/7 correct
- could also be 7100/10000 correct versus 8500/10000 correct

If you report averages, always also report
the number of repeats / size of the test set!

Let's analyze...

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

Accuracy is average outcome across 10 repeated experiments.

Let's analyze...

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

Accuracy is average outcome across 10 repeated experiments.

- can we explain the difference by random fluctuations?
- for example. what if both methods were equally good with 50% correct answers?
- under this assumption, how likely is the observed outcome, or a more extreme one?

- X_1/X_2 number of correct answers for method 1/method 2.
- $\Pr(\, X_1 \leq 4 \wedge X_2 \geq 6\,) \overset{\text{indep.}}{=} \Pr(\, X_1 \leq 4\,)\Pr(\, X_2 \geq 6\,)$
  - $\Pr(\, X_1 \leq 4\,) = \frac{\binom{10}{0}+\binom{10}{1}+\binom{10}{2}+\binom{10}{3}+\binom{10}{4}}{2^{10}} = 0.377$
  - $\Pr(\, X_2 \geq 6\,) = \frac{\binom{10}{6}+\binom{10}{7}+\binom{10}{8}+\binom{10}{9}+\binom{10}{10}}{2^{10}} = 0.377$

Let's analyze...

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

Accuracy is average outcome across 10 repeated experiments.

- can we explain the difference by random fluctuations?
- for example. what if both methods were equally good with 50% correct answers?
- under this assumption, how likely is the observed outcome, or a more extreme one?

- X_1/X_2 number of correct answers for method 1/method 2.
- $\Pr(\, X_1 \leq 4 \wedge X_2 \geq 6\,) \overset{\text{indep.}}{=} \Pr(\, X_1 \leq 4\,)\Pr(\, X_2 \geq 6\,) = \mathbf{0.142}$
  - $\Pr(\, X_1 \leq 4\,) = \frac{\binom{10}{0}+\binom{10}{1}+\binom{10}{2}+\binom{10}{3}+\binom{10}{4}}{2^{10}} = 0.377$
  - $\Pr(\, X_2 \geq 6\,) = \frac{\binom{10}{6}+\binom{10}{7}+\binom{10}{8}+\binom{10}{9}+\binom{10}{10}}{2^{10}} = 0.377$

High probability $\rightarrow$ result could easily be due to chance

General framework: significance testing

Observation:

- method 1: 40% accuracy on 10 examples
- method 2: 60% accuracy on 10 examples

Hypothesis we want to test:

"Method 2 is better than method 1."

Step 1) formulate alternative (null hypothesis):

$\mathbf{H_0}$: **"Both methods are equally good with 50% correct answers."**

Step 2) compute probability of the observed or a more extreme outcome under the condition that the null hypothesis is true: p-value

$$p = \text{Pr}(\, X_1 \leq 4 \wedge X_2 \geq 6 \,) = 0.142$$

Step 3) interpret p-value

p large (e.g. ≥ 0.05) $\rightarrow$ observations can be explained by null hypothesis

p small (e.g. $\ll 0.05$) $\rightarrow$ null hypothesis unlikely, evidence of hypothesis

Observations:

- method 1: 40% accuracy on 1000 examples
- method 2: 60% accuracy on 1000 examples

Null hypothesis:

$\mathbf{H_0}$: "Both methods are equally good with 50% correct answers."

- X_1/X_2 number of correct answers for method 1/method 2.

$$p = \mathsf{Pr}(\, X_1 \leq 400 \wedge X_2 \geq 600 \,) \leq 5 \cdot 10^{-11}$$

Advantage of method 1 over method 2 is probably real.

Observations:

- method 1: 40% accuracy on 1000 examples
- method 2: 60% accuracy on 1000 examples

Null hypothesis:

$\mathbf{H_0}$: "Both methods are equally good with 50% correct answers."

- X_1/X_2 number of correct answers for method 1/method 2.

$$p = \mathsf{Pr}(\, X_1 \leq 400 \wedge X_2 \geq 600\,) \leq 5 \cdot 10^{-11}$$

Advantage of method 1 over method 2 is probably real.

How to compute?

Let q_i be the probability of method i being correct and $n = 1000$.

- $\mathsf{Pr}(\, X_i = k) = \mathsf{Pr}(\, k \text{ out of } n \text{ correct}\,) = \binom{n}{k} q_i^k (1-q_i)^{n-k}$
- $\mathsf{Pr}(\, X_1 \leq 400) = \sum_{k=0}^{400} \mathsf{Pr}(X_i = k)$
- $\mathsf{Pr}(\, X_2 \geq 600) = 1 - \mathsf{Pr}(\, X_2 \leq 599) = \sum_{k=601}^{1000} \mathsf{Pr}(X_2 = k)$

Implemented as `binocdf` in Matlab or `binom.cdf` in Python.

Observations:

- method 1: 99.77% accuracy on 10000 examples
- method 2: 99.79% accuracy on 10000 examples

Null hypothesis:

$\mathbf{H_0}$: "Both methods are equally good with 99.78% correct answers."

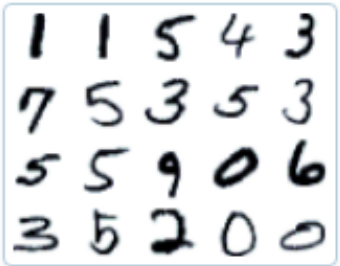
- X_1/X_2 number of correct answers for method 1/method 2.

$$p = \mathsf{Pr}(\, X_1 \leq 9977 \wedge X_2 \geq 9979 \,) \approx 20.9\%$$

Results could have easily happened by chance without method 2 actually being better than method 1.

MNIST

who is the best in MNIST ?



MNIST 50 results collected

Units: error %

> Classify handwriten digits. Some additional results are available on the original dataset page.

Result	Method	Venue	Details
0.21%	Regularization of Neural Networks using DropConnect	ICML 2013	
0.23%	Multi-column Deep Neural Networks for Image Classification	CVPR 2012	
0.23%	APAC: Augmented PAttern Classification with Neural Networks	arXiv 2015	
0.24%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
0.29%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
0.31%	Recurrent Convolutional Neural Network for Object Recognition	CVPR 2015	

Source: http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html

Observations:

- method 1: 0.740 average precision on 3412 examples
- method 2: 0.715 average precision on 3412 examples

Observations:

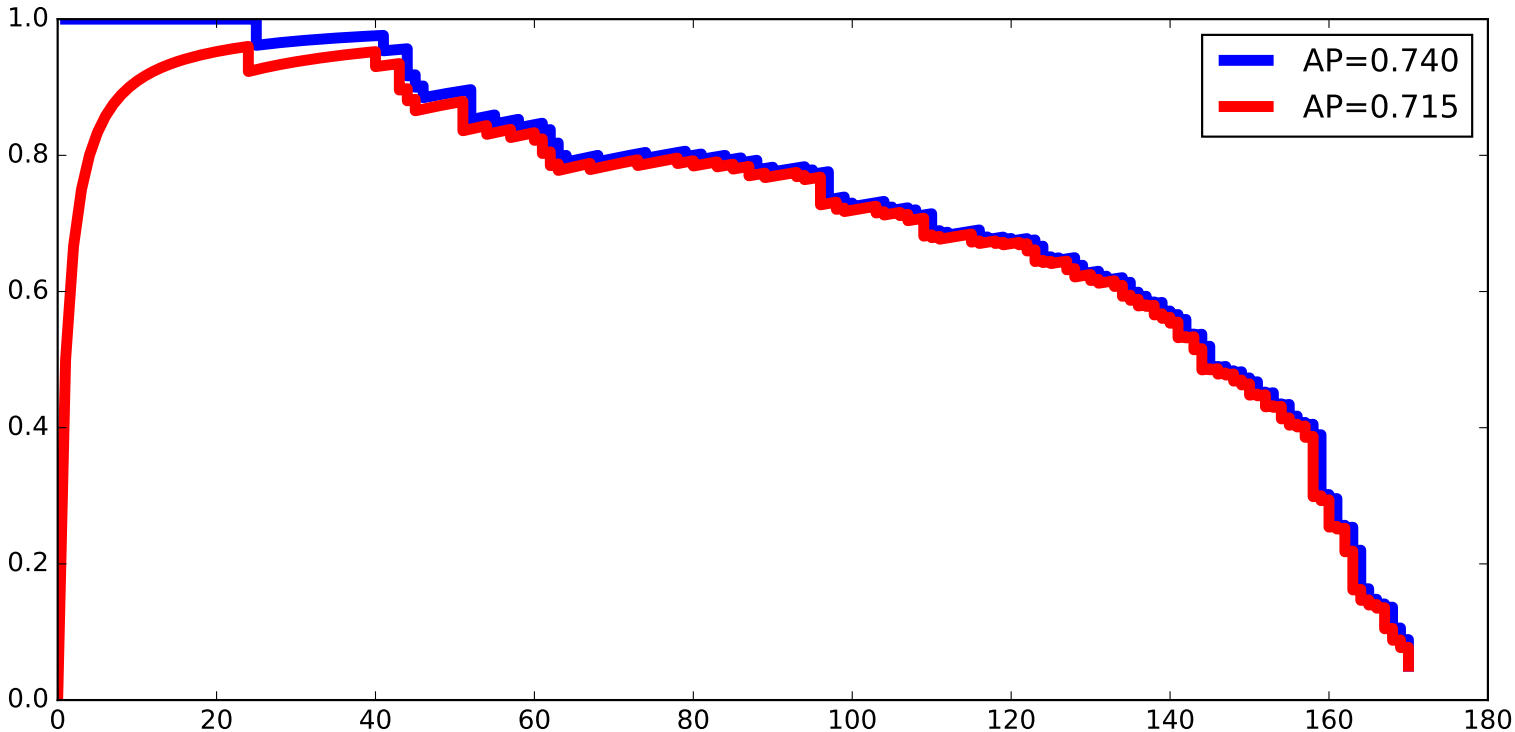
- method 1: 0.740 average precision on 3412 examples
- method 2: 0.715 average precision on 3412 examples

Average precision is not an average over repeated experiments.
$\rightarrow$ we can't judge significance
$\rightarrow$ to be convincing, do experiments on more than one dataset

Observations:

- method 1: 0.740 average precision on 3412 examples
- method 2: 0.715 average precision on 3412 examples

Average precision is not an average over repeated experiments.
$\rightarrow$ we can't judge significance
$\rightarrow$ to be convincing, do experiments on more than one dataset



Difference: highest scored example correct (blue) or misclassified (red)

Scientific scrutiny

- is the solved problem really important?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, are they truly comparable?

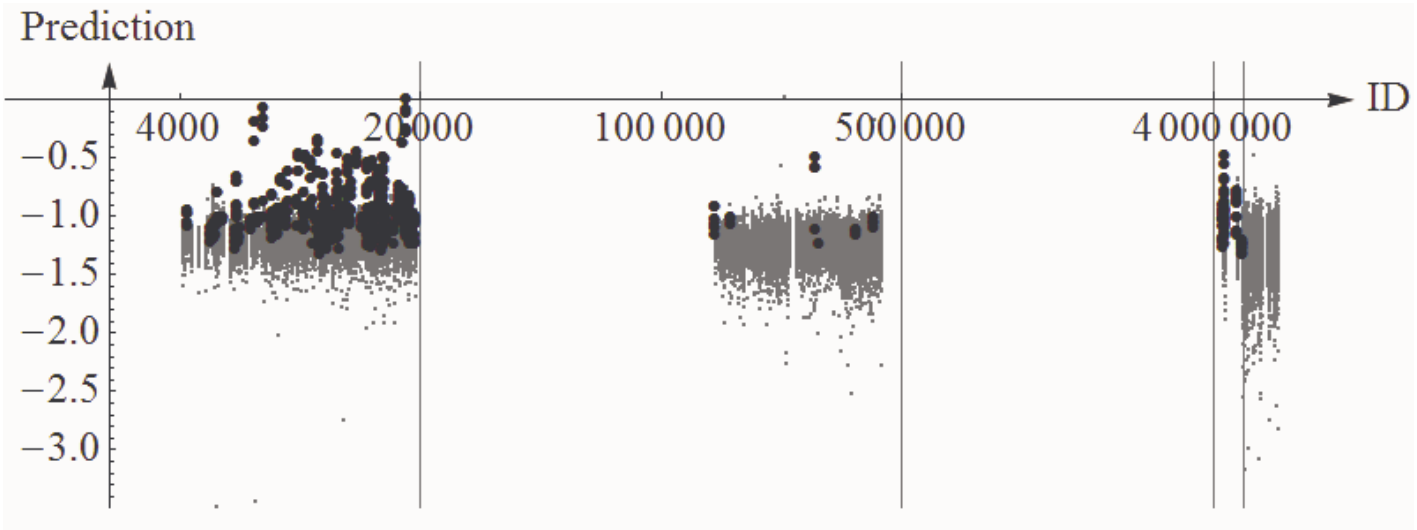
Scientific scrutiny

- is the solved problem really important?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, are they truly comparable?

Did we (accidentially) rely on artifacts in the data?

KDD Cup 2008: Breast cancer detection from images

- patient ID carried significant information about label
  - data was anonymized, but not properly randomized
  - patients from the first and third group had higher chance of cancer
- classifier bases decision on ID $\rightarrow$ not going to work for future patients



Scientific scrutiny

- is the solved problem really important?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, are they truly comparable?

Scientific scrutiny

- is the solved problem really important?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, are they truly comparable?

Is the data representative for the actual problem?

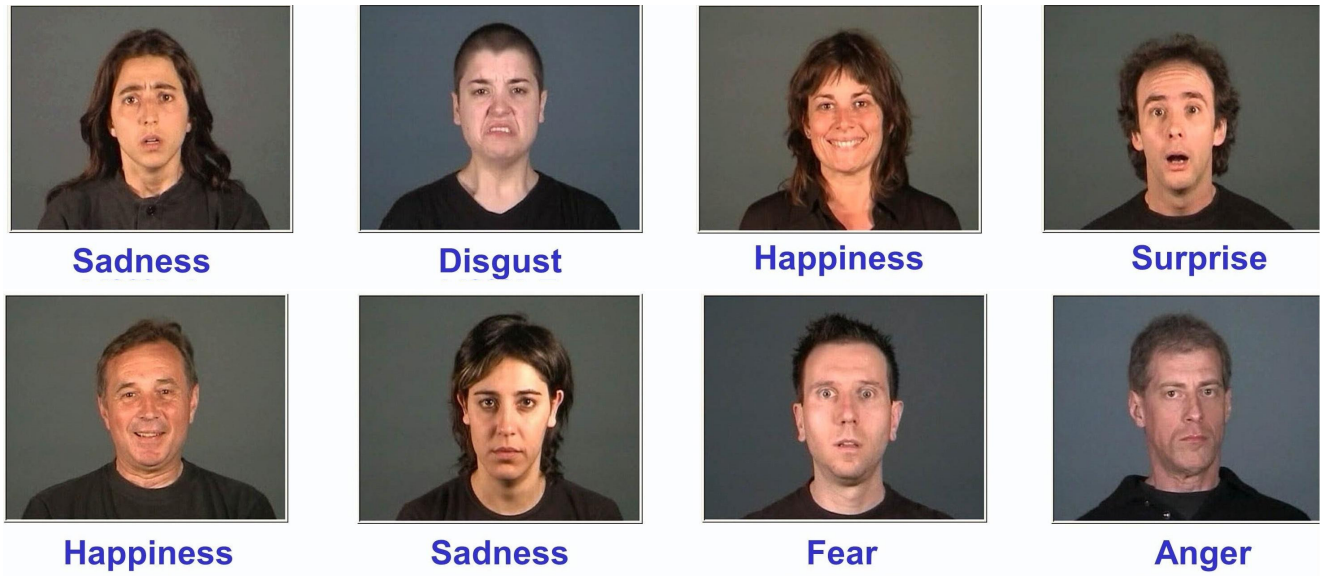
Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions

Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

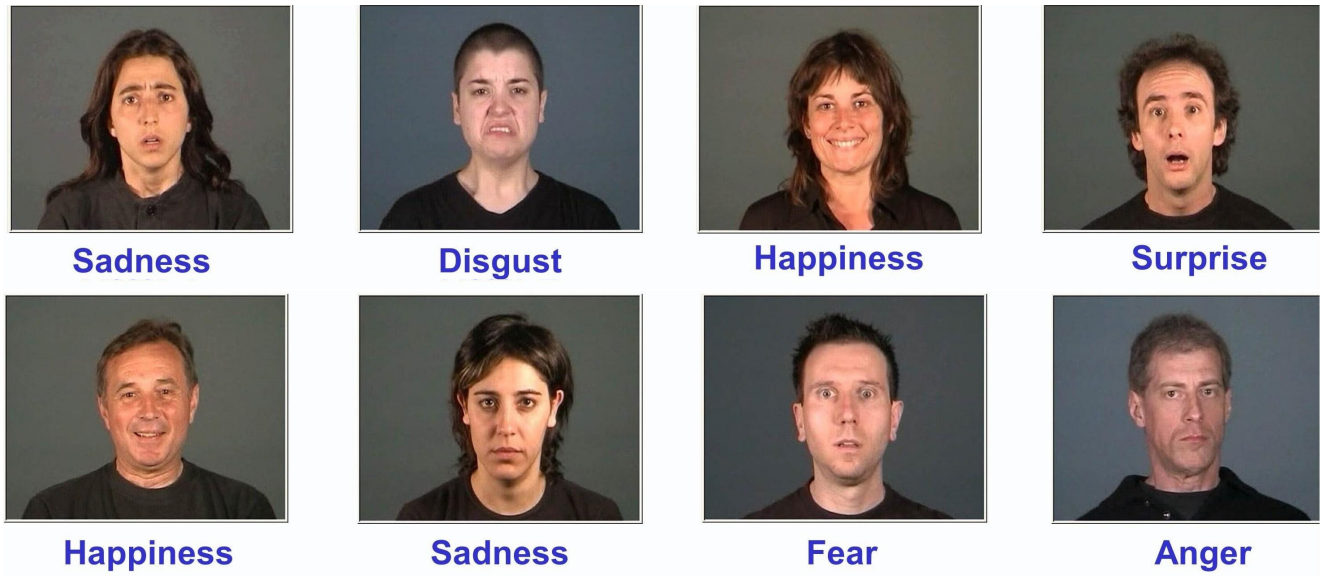
- e.g. 99% accuracy on DAFEX database for facial expressions



Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions

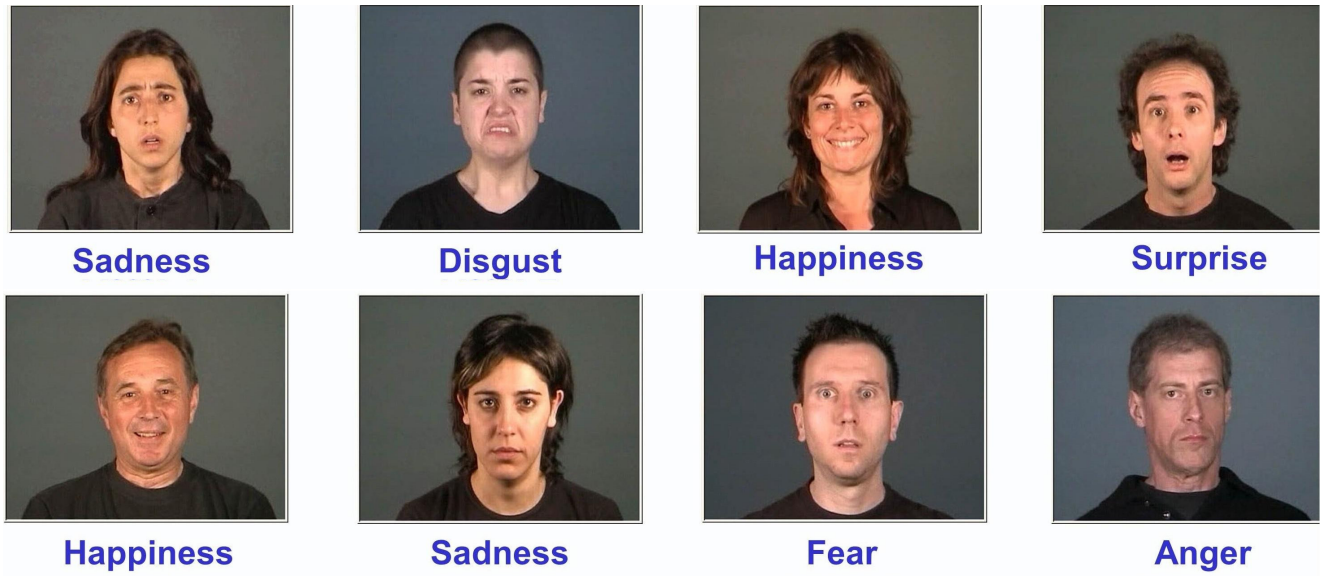


- frontal faces at fixed distance. Will that be the case in the car?

Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions

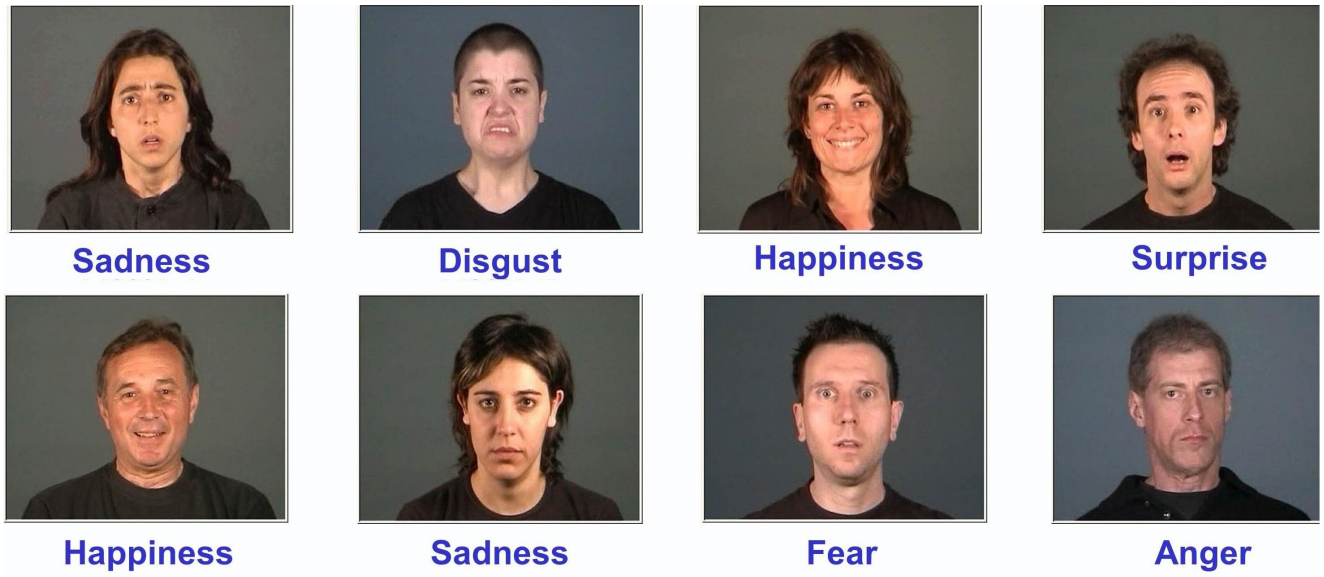


- frontal faces at fixed distance. Will that be the case in the car?
- perfect lighting conditions. Will that be the case in the car?

Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions

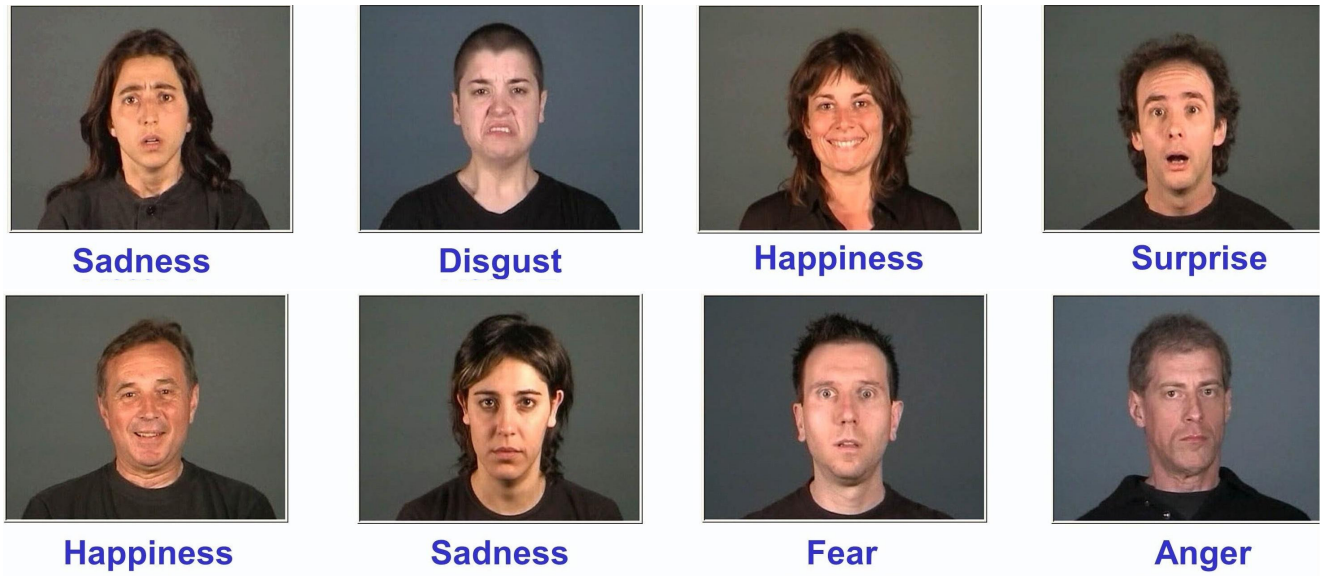


- frontal faces at fixed distance. Will that be the case in the car?
- perfect lighting conditions. Will that be the case in the car?
- homogeneous gray background. Will that be the case in the car?

Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions

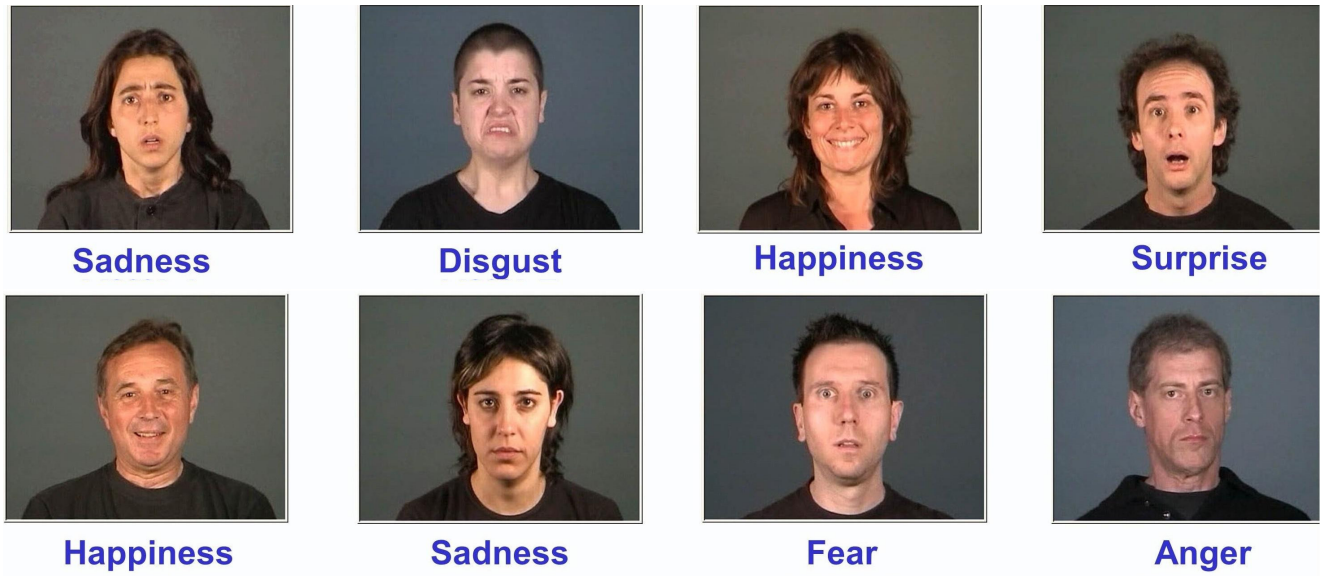


- frontal faces at fixed distance. Will that be the case in the car?
- perfect lighting conditions. Will that be the case in the car?
- homogeneous gray background. Will that be the case in the car?
- all faces are Italian actors. Will all drivers be Italian actors?

Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions

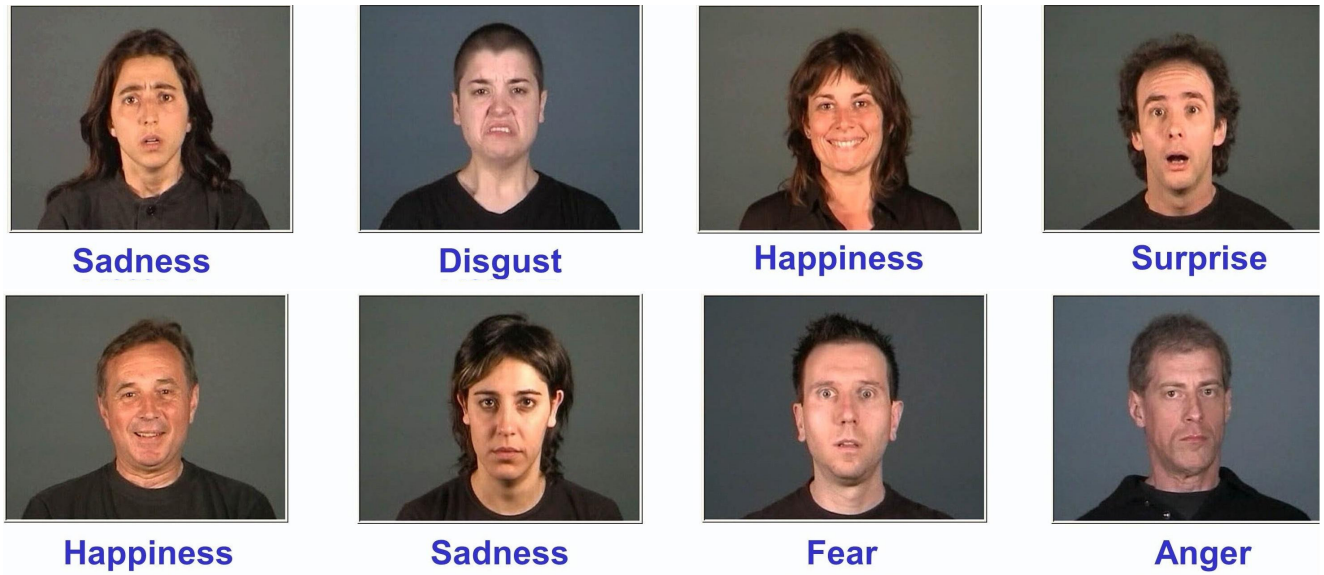


- frontal faces at fixed distance. Will that be the case in the car?
- perfect lighting conditions. Will that be the case in the car?
- homogeneous gray background. Will that be the case in the car?
- all faces are Italian actors. Will all drivers be Italian actors?
- the emotions are not real, but (over)acted!

Is the data representative for the actual problem?

Example task: emotion recognition for autonomous driving

- e.g. 99% accuracy on DAFEX database for facial expressions



- frontal faces at fixed distance. Will that be the case in the car?
- perfect lighting conditions. Will that be the case in the car?
- homogeneous gray background. Will that be the case in the car?
- all faces are Italian actors. Will all drivers be Italian actors?
- the emotions are not real, but (over)acted!

$\rightarrow$ **little reason to believe that the system will work in practice**

Images: https://i3.fbk.eu/resources/dafex-database-kinetic-facial-expressions

Scientific scrutiny

- is the solved problem really important?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, are they truly comparable?

Scientific scrutiny

- is the solved problem really important?
- are the results explainable just by random chance/noise?
- does the system make use of artifacts in the data?
- is the data representative for the actual problem?
- if you compare to baselines, are they truly comparable?

If you compare to other work, is the comparison fair?

If you compare to other work, is the comparison fair?

Not always easy to answer:

Do the methods have solve the same problem?

- previous work: object classification
- proposed work: object detection

If you compare to other work, is the comparison fair?

Not always easy to answer:

Do the methods have solve the same problem?

- previous work: object classification
- proposed work: object detection

Do they evaluate their performance on the same data?

- previous work: PASCAL VOC 2010
- proposed work: PASCAL VOC 2012

If you compare to other work, is the comparison fair?

Not always easy to answer:

Do the methods have solve the same problem?

- previous work: object classification
- proposed work: object detection

Do they evaluate their performance on the same data?

- previous work: PASCAL VOC 2010
- proposed work: PASCAL VOC 2012

Do they have access to the same data for training?

- previous work: PASCAL VOC 2012
- proposed work: PASCAL VOC 2012, network pretrained on ImageNet

Do they have access to the same pre-processing tools?

- previous work: *fc7* features extracted from AlexNet
- proposed work: *fc7* features extracted from ResNet-151

Do they have access to the same pre-processing tools?

- previous work: *fc7* features extracted from AlexNet
- proposed work: *fc7* features extracted from ResNet-151

Do they have access to the same annotation?

- previous work: PASCAL VOC 2012 with per-image annotation
- proposed work: PASCAL VOC 2012 with bounding-box annotation

Do they have access to the same pre-processing tools?

- previous work: *fc7* features extracted from AlexNet
- proposed work: *fc7* features extracted from ResNet-151

Do they have access to the same annotation?

- previous work: PASCAL VOC 2012 with per-image annotation
- proposed work: PASCAL VOC 2012 with bounding-box annotation

for runtime comparisons: Do you use comparable hardware?

- previous work: laptop with 1.4 GHz single-core CPU
- proposed work: workstation with 4 GHz quad-core CPU

for runtime comparisons: Do you use comparable languages?

- previous work: method implemented in Matlab
- proposed work: method implemented in C++

Is model selection for them done thoroughly?

- baseline: hyperparameter set ad-hoc or to "default value", e.g. $\lambda = 1$
- proposed method: model selection done properly

Now, is the new method better, or just the baseline worse than it could be?

Is model selection for them done thoroughly?

- baseline: hyperparameter set ad-hoc or to "default value", e.g. $\lambda = 1$
- proposed method: model selection done properly

Now, is the new method better, or just the baseline worse than it could be?

Positive exception:

Nowozin, Bakır. "Discriminative Subsequence Mining for Action Classification", ICCV 2007

Method	**KTH accuracy**
Niebles et al. [11], LOO, pLSA	81.50
Dollár et al. [4], LOO, SVM RBF	80.66
Schuldt et al. [17], splits, SVM match	71.71
Ke et al. [7], splits, forward feat.-sel.	62.94
Subsequence Boosting, $B = 12$, splits	84.72

Is model selection for them done thoroughly?

- baseline: hyperparameter set ad-hoc or to "default value", e.g. $\lambda = 1$
- proposed method: model selection done properly

Now, is the new method better, or just the baseline worse than it could be?

Positive exception:

Nowozin, Bakır. "Discriminative Subsequence Mining for Action Classification", ICCV 2007

Method	**KTH accuracy**
Niebles et al. [11], LOO, pLSA	81.50
Dollár et al. [4], LOO, SVM RBF	80.66
Schuldt et al. [17], splits, SVM match	71.71
Ke et al. [7], splits, forward feat.-sel.	62.94
baseline SVM linear bin, 1-vs-1	83.33
baseline SVM RBF bin, 1-vs-1	85.19
baseline SVM χ^2 bin, 1-vs-1	87.04
Subsequence Boosting, $B = 12$, splits	84.72

Take home messages

Don't use machine learning unless you have good reasons.

Keep in mind: all data is random and all numbers are uncertain.

Learning is about finding a model that works on future data.

Proper model-selection is hard. Avoid free (hyper)parameters!

A test set is for evaluating a single final ultimate model, nothing else.

You don't do research for yourself, but for others (users, readers, ...).

Results must be convincing: honest, reproducible, not overclaimed.

Fast-forward... to 2017

The year is 2017 AD. Computer Vision Research is entirely dominated by Machine Learning approaches. Well, not entirely...

Statistics

- 783 papers at CVPR conference in 2017
- 721 (92%) contain the expression "[Ll]earn", 62 don't
- mean: 26.1 times
- median: 20 times
- maximum: 127 times

...but almost.

Examples

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

Examples

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

CVPR 2017

JOIN US IN Honolulu, Hawaii

Main Conference July 22 to July 25

Workshops July 21 and July 26


intel
AMAZING
POWERED BY INTEL
FEATURE X
We use computer vision and machine learning to discover meaningful
AMAZING
POWERED BY INTEL

Applications with industrial interest:

- Self-driving cars
- Augmented reality
- Visual effects (e.g. style transfer)
- Vision for shopping/products
- Vision for sport analysis

Examples

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

New model classes, e.g. ResNet, DenseNet

Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections—one between each layer and its subsequent layer—our network has $\frac{L(L+1)}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage fea-

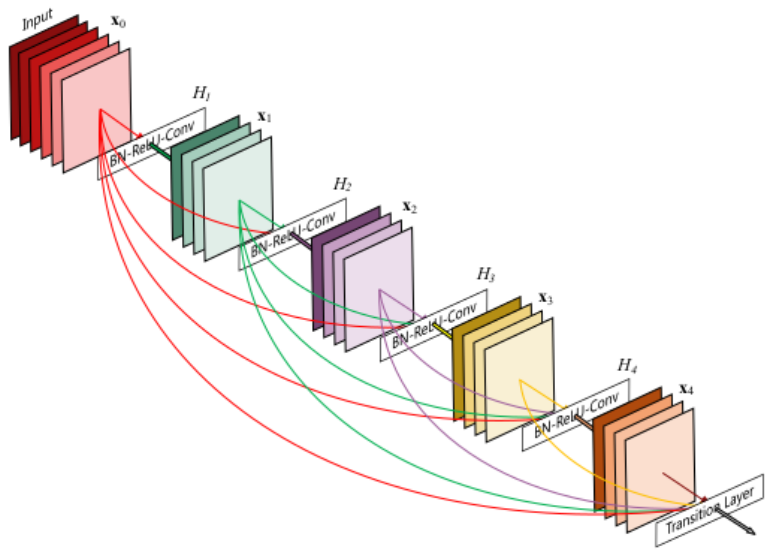


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

New loss functions: e.g. adversarial networks

Task: create naturally looking images

How to quantify success?

New loss functions: e.g. adversarial networks

Task: create naturally looking images

How to quantify success?

- Learn a loss function:
  - training set of "true" images
  - learn a 2nd model to distinguish between true and generated images
  - loss function for image generation $\equiv$ accuracy of the 2nd model

Generative Adversarial Networks (GANs)

$\rightarrow$ Viktor Lempitsky's lecture on Tuesday

Examples

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

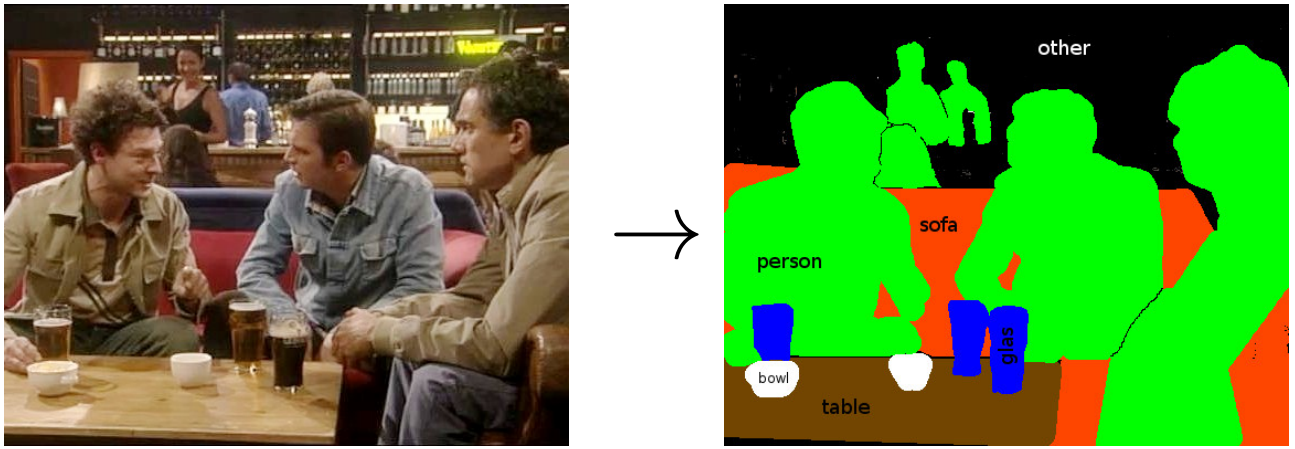
Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

New form of training data: weakly supervised

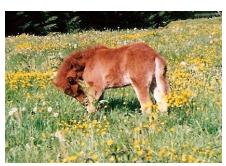
Task: semantic image segmentation (predict a label for each pixel)



Training data: images, annotated with per-image class labels


cat
sofa

,


horse

,

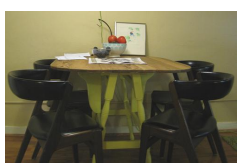


table
chair

, ...

- annotation contains less information, but is much easier to generate
  → Vittorio Ferrari's lecture on Friday

Examples

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

2017 IEEE Conference on Computer Vision and Pattern Recognition

Improving training of deep neural networks via Singular Value Bounding

Kui Jia[1], Dacheng Tao[2], Shenghua Gao[3], and Xiangmin Xu[1]

[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China
[2]UBTech Sydney AI Institute, SIT, FEIT, The University of Sydney, Australia
[3]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
{kuijia,xmxu}@scut.edu.cn, dacheng.tao@sydney.edu.au, gaoshh@shanghaitech.edu.cn

Abstract

Deep learning methods achieve great success recently on many computer vision problems. In spite of these practical successes, optimization of deep networks remains an active topic in deep learning research. In this work, we focus on investigation of the network solution properties that can potentially lead to good performance. Our research is inspired

newly proposed deep architectures that have huge model capacities, such as Inception [25] and ResNet [7]. Training of these ultra-deep/ultra-wide networks are enabled by modern techniques such as Batch Normalization (BN) [11] and residual learning [7].

In spite of these practical successes, however, optimization of deep networks remains an active topic in deep learning research. Until recently, deep networks are considered

[Jia, Tao, Gao, Xu. "Improving training of deep neural networks via Singular Value Bounding", CVPR 2017]

Examples

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

Step 2) Collect and annotate data

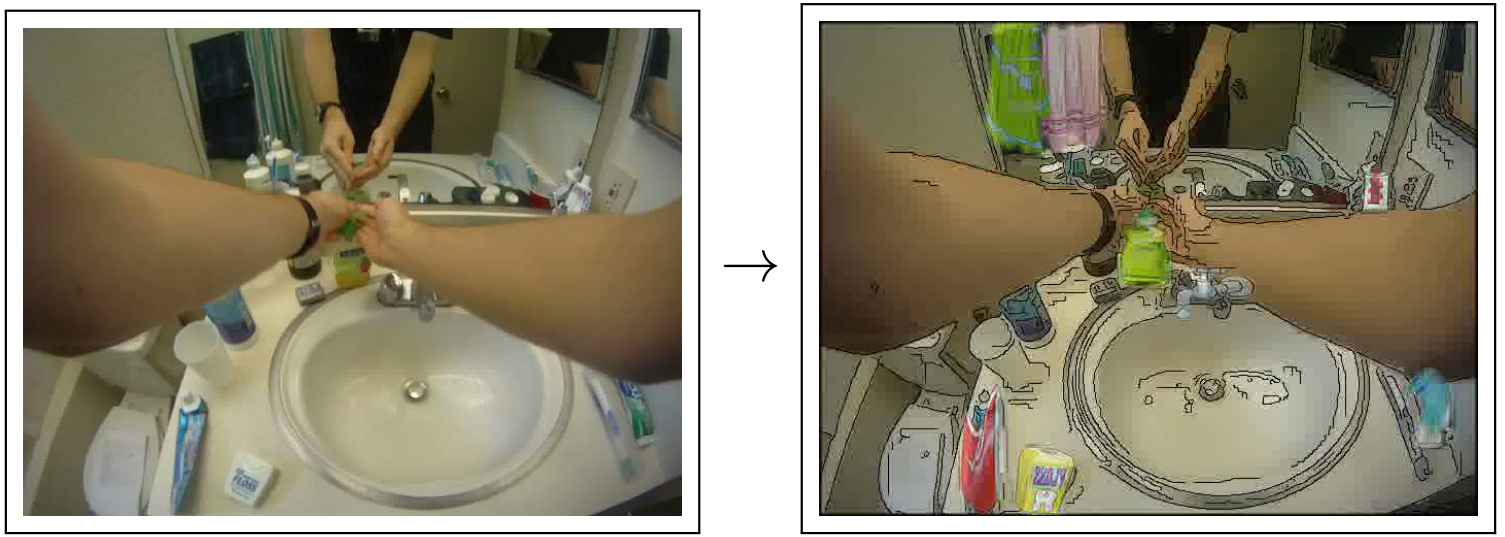
Step 3) Model training

Step 4) Model evaluation

New developments: hard to evaluate tasks

Some models are hard to evaluate, e.g. because answers are subjective:

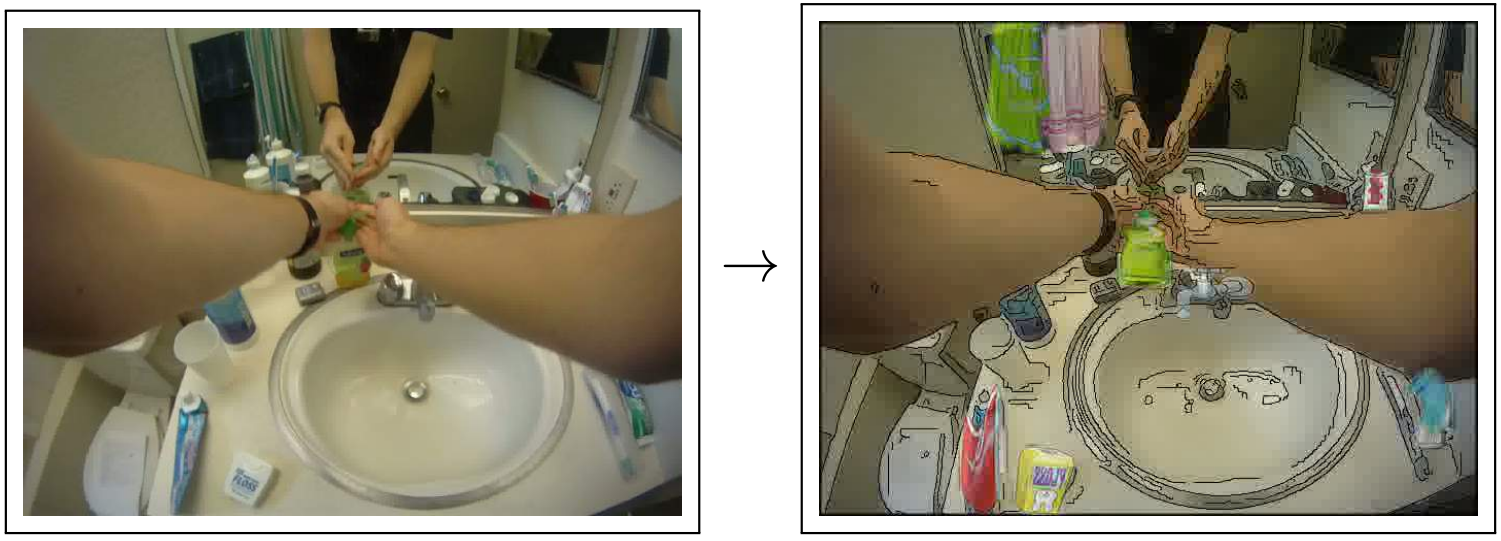
Example: video cartooning



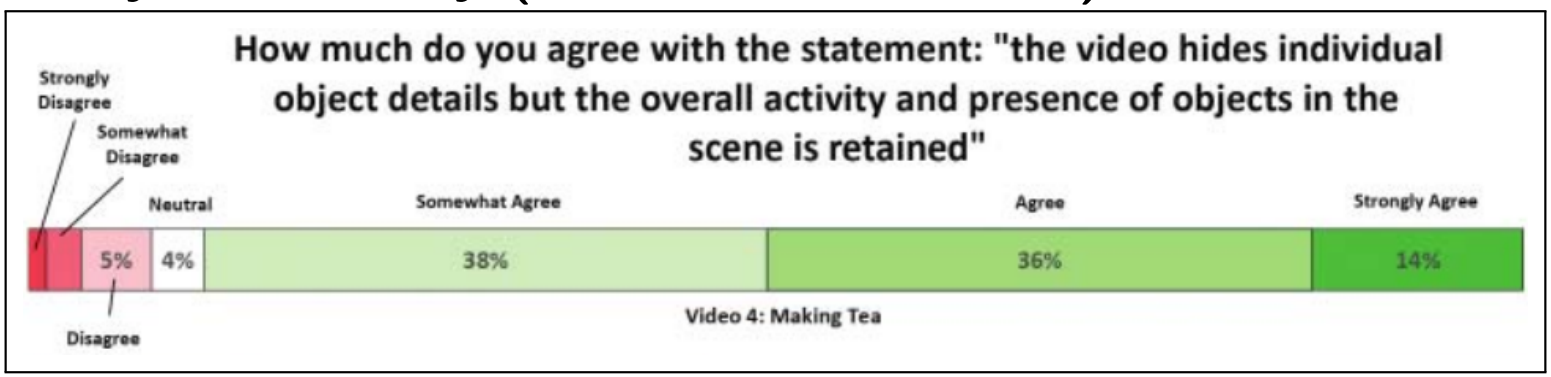
New developments: hard to evaluate tasks

Some models are hard to evaluate, e.g. because answers are subjective:

Example: video cartooning



Evaluate by user survey (on Mechanical Turk):



[Hassan et al. "Cartooning for Enhanced Privacy in Lifelogging and Streaming Videos", Workshop@CVPR 2017]

Summary

Step 0) Choosing an application...

Step 1) Decide on model class and loss function

Step 2) Collect and annotate data

Step 3) Model training

Step 4) Model evaluation

Summary: active research in all aspects