Statistical Machine Learning

Lecture 5 – Cross-validation and the bias-variance trade-off

UPPSALA
UNIVERSITET

Andreas Lindholm
Division of Systems and Control
Department of Information Technology
Uppsala University

andreas.lindholm@it.uu.se

Summary of Lecture 4 (I/III)

Linear Discriminant Analysis (LDA) uses Bayes' theorem for classification, by writing the conditional class probabilities as,

$$\Pr(y = k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y = k)\Pr(y = k)}{\sum_{j=1}^{K} p(\mathbf{x} \mid y = j)\Pr(y = j)}.$$

where

- $\pi_k = \Pr(y = k)$ is the *prior* probability of class k.
- $p(\mathbf{x} \mid y = k) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ is the (assumed) probability density of $\mathbf{x}$ for an observation that comes from the kth class.

The **parameters** are: $\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}$

UPPSALA UNIVERSITET

Summary of Lecture 4 (II/III)

The parameters are estimated as the class frequencies and (within class) sample means and covariances, respectively,

$$\begin{aligned} \widehat{\pi}_k &= n_k/n & k &= 1, \dots, K, \\ \widehat{\boldsymbol{\mu}}_k &= \frac{1}{n_k} \sum_{i:y_i=k} \mathbf{x}_i, & k &= 1, \dots, K, \\ \widehat{\boldsymbol{\Sigma}} &= \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^{\mathsf{T}}. \end{aligned}$$

The LDA classifier assigns a test input $\mathbf{x}_\star$ to class k for which the predicted probability $\Pr(y = k \,|\, \mathbf{x}_\star)$, or equivalently,

$$\widehat{\delta}_k(\mathbf{x}_\star) = \mathbf{x}_\star^{\mathsf{T}} \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k - \frac{1}{2} \widehat{\boldsymbol{\mu}}_k^{\mathsf{T}} \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k + \log \widehat{\pi}_k$$

is largest. Thus, LDA is a **linear classifier**.

Summary of Lecture 4 (III/III)

Non-parametric models are not specified using a fixed-dimensional vector of parameters. Instead, they allow the flexibility of the model to grow with the amount of available data.

One example is the **k-nearest neighbour classifier**.

Given training data $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, for a test input $\mathbf{x}_\star$,

1. Identify the k training inputs $\mathbf{x}_i$ nearest to $\mathbf{x}_\star$
2. Classify $\mathbf{x}_\star$ according to a majority vote amongst these k training samples.

UPPSALA
UNIVERSITET

Evaluating a supervised machine learning method

Example

Let’s say we have to design a subsystem for a self-driving car, which distinguishes between cyclists and pedestrians in camera images

- Input $\mathbf{x}$: The color (RGB value) of each pixel
- Output $y \in \{\texttt{cyclist},\texttt{pedestrian}\}$

$\rightarrow$ a classification problem

UPPSALA UNIVERSITET

Example

1. We spend a lot of resources to collect training data:
   $10\,000$ images of cyclists and pedestrians (each with a class label).
2. We train a classifier[1] using all training data. It classifies 99.7% of the training data correctly.
3. The system is integrated into the self-driving car, and eventually sold to a customer.

How many cyclists and pedestrians will be classified correctly when the car drives around in the real world?

[1] Perhaps a convolutional neural network, Lecture 9

Let's generalize using math

- Training data $\mathcal{T} \triangleq \{\mathbf{x}_i, y_i\}_{i=1}^n$
- Prediction $\widehat{y}(\mathbf{x}; \mathcal{T})$, output from a method learned using $\mathcal{T}$ *($\hat{g}(\mathbf{x})$ in previous lecture)*
- Error function $E(\widehat{y}, y) = \begin{cases} \mathbb{I}(y \neq \widehat{y}) \text{ misclassification (classification problems)} \\ (y - \widehat{y})^2 \text{mean square error (regression problems)} \end{cases}$
- Training data error $E_{\text{train}} \triangleq \frac{1}{n}\sum_{i=1}^n E(\widehat{y}(\mathbf{x}_i; \mathcal{T}), y_i)$
- The distribution of inputs and outputs in the real world $p(\mathbf{x}_\star, y_\star)$
- New data error $E_{\text{new}} \triangleq \int E(\widehat{y}(\mathbf{x}_\star; \mathcal{T}), y_\star) p(\mathbf{x}_\star, y_\star) \, d\mathbf{x}_\star dy_\star \triangleq \mathbb{E}_\star \left[E(\widehat{y}(\mathbf{x}_\star; \mathcal{T}), y_\star)\right]$.

($\mathbb{E}_\star$ = expected value over all future test inputs $\mathbf{x}_\star, y_\star$)

Connecting to the example

- $\mathcal{T}$: The 10 000 images
- $\widehat{y}(\mathbf{x}; \mathcal{T})$: the prediction (cyclist/pedestrian) from the trained classifier
- $E(\widehat{y}, y)$: whether the prediction for an image is correct or not
- $E_{\text{train}} = 0.003$ (how many cyclists and pedestrians it predicted wrong in the training data)
- $p(\mathbf{x}_\star, y_\star)$: The distribution of cyclists and pedestrians in the real world (appearance on camera and how often they occur)
- $E_{\text{new}} =$ **???** (how many cyclists and pedestrians it will predict wrong in the real world)

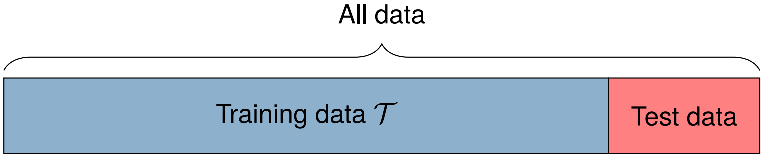
What to be said about E_{new}?

- A useful machine learning method gives a small E_{new} for our problem
- It would be useful to know E_{new}, so that we can tell if the method is good for our problem
- E_{new} depends on $p(\mathbf{x}, y)$, which we do not know

Let's try to *estimate* E_{new}!

UPPSALA UNIVERSITET

Estimating E_{new}: test data approach



Set aside a part of the data and use only as **test data** for estimating E_{new}:

1. Use the training data $\mathcal{T}$ for learning the model
2. Compute the predictions $\widehat{y}(\mathbf{x}_\star; \mathcal{T})$ for all data points $\mathbf{x}_\star$ in the test data set
3. Take the average of $E(\widehat{y}(\mathbf{x}_\star; \mathcal{T}), y_\star)$ as an estimate of E_{new}

+ Simple

! To have a good estimate of E_{new}, the test dataset has to be large

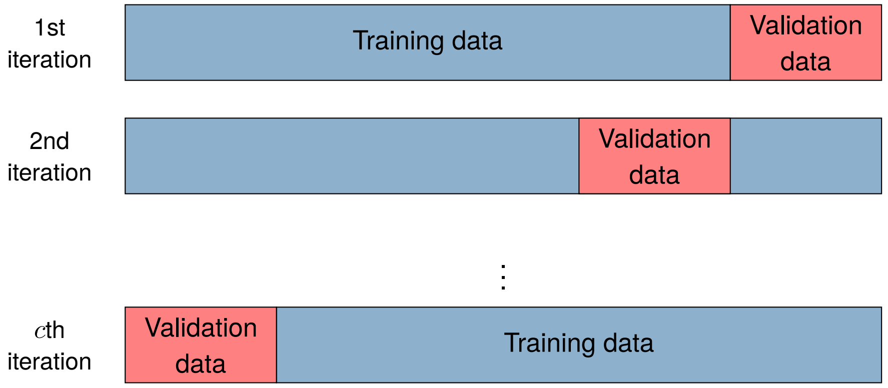
! To get a good training of the model, the training dataset has to be large

– Not all data is used for learning

Always split **randomly** between training and test data!

UPPSALA UNIVERSITET

Estimating E_{new}: c-fold cross-validation



Iterate the test data approach c times, and take the average as an estimate of E_{new}.

+ Gives a better estimate of E_{new}
– Computationally more demanding than the test data approach

UPPSALA UNIVERSITET

Other flavors of cross-validation

- Leave-one-out cross-validation: c-fold cross-validation with $c = n$
- Monte Carlo cross-validation: random selection of validation set at each iteration

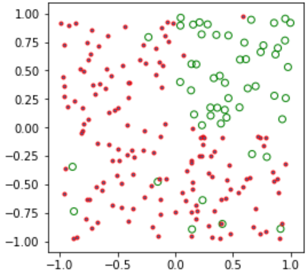
UPPSALA UNIVERSITET

Example: cross-validation for model selection

All models are wrong, but some are useful. — George Box

Binary classification, $p = 2$. Should we use logistic regression, QDA or k-NN for this problem?

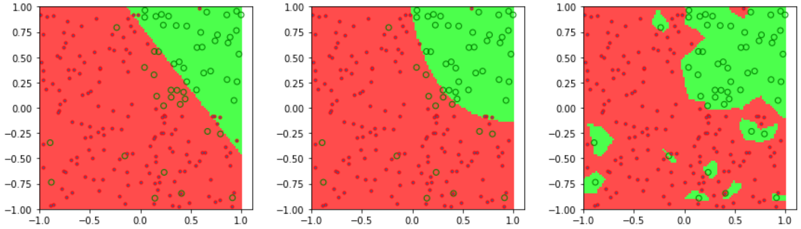
We would like to pick the method which has lowest E_{new}. Since we only have the data, we can only *estimate* E_{new}.



	logistic regression	QDA	k-NN ($k = 1$)
Estimated* E_{new}	0.145	0.115	0.13

*using cross-validation

Example: cross-validation for model selection



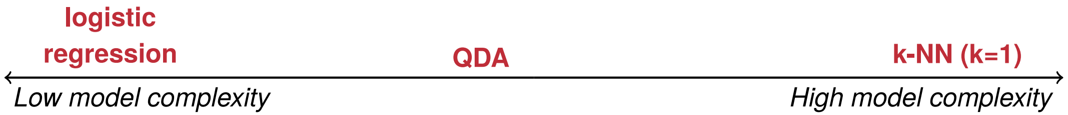
	logistic regression	QDA	k-NN ($k = 1$)
Estimated E_{new}	0.15	0.12	0.13
E_{train}	0.14	0.11	0.0

Different methods have different $E_{\textbf{new}}$ for different problems.

Why?

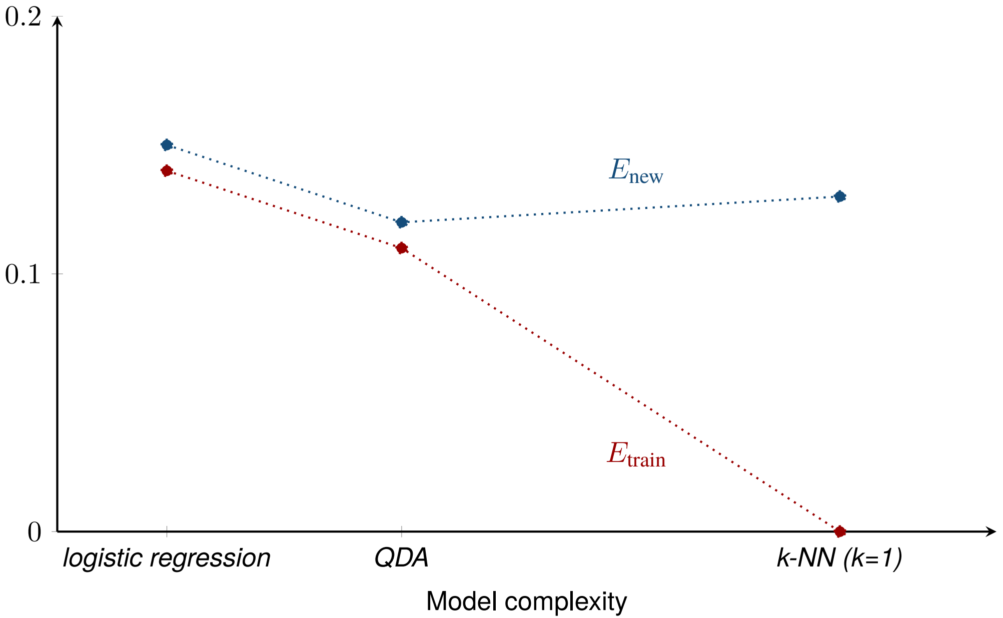
Model complexity

The more prone a model is to adapt to complicated pattern in the data, the higher the model complexity (or model flexibility):



It is possible to make a formal definition of model complexity. We will not do that in this course.

The example again



Generalization error

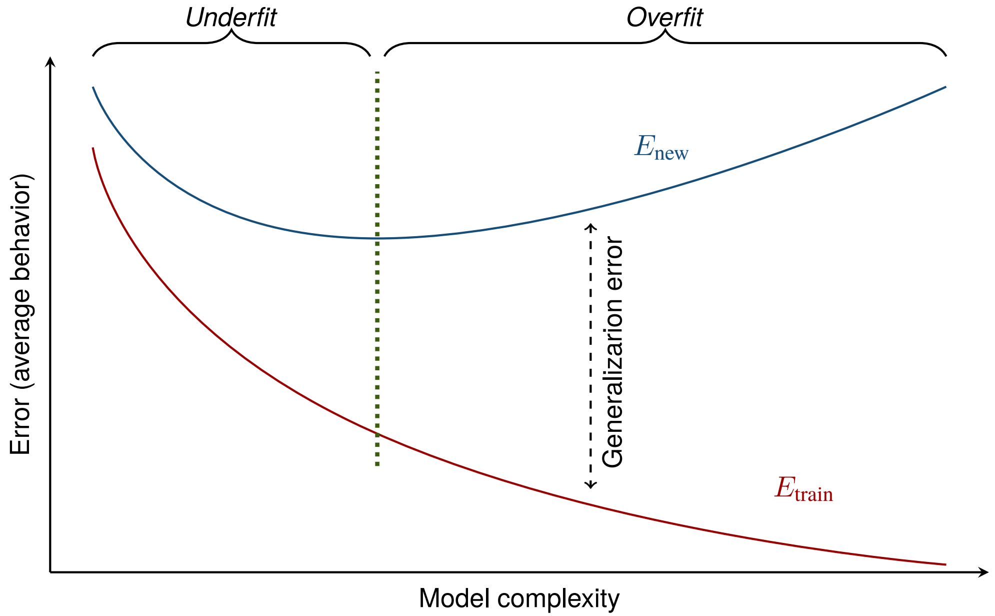
In general, we can say

$$E_{\text{new}} = E_{\text{train}} + \text{generalization error}$$

Typically,

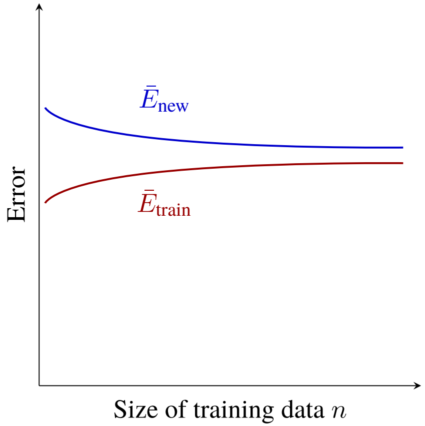
- Higher model complexity $\Longrightarrow$ larger generalization error
- Higher model complexity $\Longrightarrow$ smaller E_{train}

Model complexity, E_{train} and E_{new}

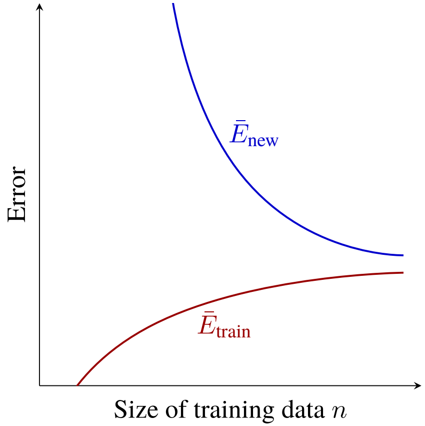


UPPSALA UNIVERSITET

Generalization error and training data size



Low model complexity



High model complexity

UPPSALA UNIVERSITET

Digging deeper to understand E_{new}

Collecting training data is a random process (sampling $\mathcal{T}$ from $p(\mathbf{x}, y)$) meaning there is randomness in the learned model.

Idea: Use probability theory to describe E_{new} in terms of **random effects in the model** *(variance)* and **systematic effects in the model** *(bias)*.

Let's focus on the regression setting.

Bias-variance decomposition I/III

First, introduce $\bar{E}_{\text{new}}$ where we average E_{new} over different training datasets,

$$\bar{E}_{\text{new}} \triangleq \mathbb{E}_{\mathcal{T}}\left[E_{\text{new}}\right]$$

where $\mathbb{E}_{\mathcal{T}} =$ expected value over all possible training data sets $\mathcal{T}$ drawn from $p(\mathbf{x}, y)$).
Technically, it is $\bar{E}_{\text{new}}$ (and not E_{new}) which we are going to decompose into bias and variance.

Now assume that ”the real world” works as $y = f(\mathbf{x}) + \epsilon$, where ϵ is uncorrelated with $\mathbf{x}$ and has mean 0

$$\begin{aligned} E_{\text{new}} =& \mathbb{E}_\star\left[E(\widehat{y}(\mathbf{x}_\star;\mathcal{T}), y_\star)\right] = \mathbb{E}_\star\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - f(\mathbf{x}_\star) - \epsilon\right)^2\right] \\ =& \mathbb{E}_\star\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T})\right)^2\right] - 2\mathbb{E}_\star\left[\widehat{y}(\mathbf{x}_\star;\mathcal{T})f(\mathbf{x}_\star)\right] + 2\mathbb{E}_\star\left[f(\mathbf{x}_\star) - \widehat{y}(\mathbf{x}_\star;\mathcal{T})\right]\underbrace{\mathbb{E}_\star\left[\epsilon\right]}_{0} + \mathbb{E}_\star\left[\left(f(\mathbf{x}_\star)\right)^2\right] + \underbrace{\mathbb{E}_\star\left[\epsilon^2\right]}_{\sigma^2} \\ =& \mathbb{E}_\star\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T})\right)^2 - 2\widehat{y}(\mathbf{x}_\star;\mathcal{T})f(\mathbf{x}_\star) + \left(f(\mathbf{x}_\star)\right)^2\right] + \sigma^2 \end{aligned}$$

UPPSALA UNIVERSITET

Bias-variance decomposition II/III

Introduce

$$g(\mathbf{x}_\star) = \mathbb{E}_{\mathcal{T}}\left[\widehat{y}(\mathbf{x}_\star;\mathcal{T})\right],$$

the predicted output from the “average model”.

$$\begin{aligned}
\bar{E}_{\text{new}} =& \mathbb{E}_{\mathcal{T}}\left[E_{\text{new}}\right] = \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_\star\left[(\widehat{y}(\mathbf{x}_\star;\mathcal{T}))^2 - 2\widehat{y}(\mathbf{x}_\star;\mathcal{T})f(\mathbf{x}_\star) + (f(\mathbf{x}_\star))^2\right] + \sigma^2\right] \\
=& \mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[(\widehat{y}(\mathbf{x}_\star;\mathcal{T}))^2\right] - 2\,\underbrace{\mathbb{E}_{\mathcal{T}}\left[\widehat{y}(\mathbf{x}_\star;\mathcal{T})\right]}_{g(\mathbf{x}_\star)} f(\mathbf{x}_\star) + (f(\mathbf{x}_\star))^2\right] + \sigma^2 \\
=& \mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[(\widehat{y}(\mathbf{x}_\star;\mathcal{T}))^2\right] - (g(\mathbf{x}_\star))^2 + (g(\mathbf{x}_\star))^2 - 2g(\mathbf{x}_\star)f(\mathbf{x}_\star) + (f(\mathbf{x}_\star))^2\right] + \sigma^2 \\
=& \underbrace{\mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[(\widehat{y}(\mathbf{x}_\star;\mathcal{T}))^2\right] - (g(\mathbf{x}_\star))^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_\star\left[(g(\mathbf{x}_\star) - f(\mathbf{x}_\star))^2\right]}_{\text{Bias}^2} + \underbrace{\sigma^2}_{\text{Irreducible error}}
\end{aligned}$$

UPPSALA UNIVERSITET

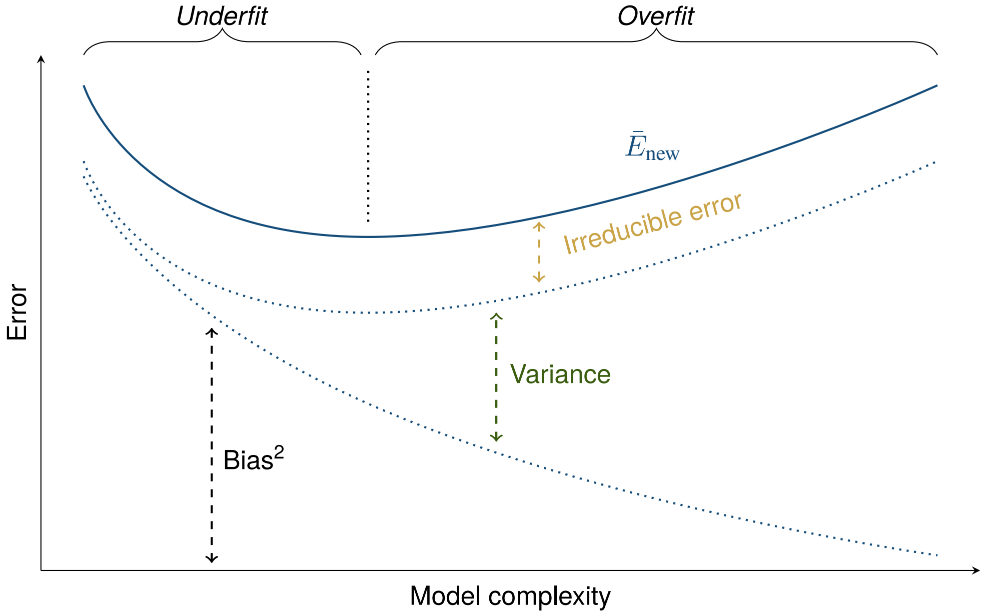
Bias-variance decomposition III/III

Technical interpretation:

- **Bias²** $\mathbb{E}_\star\left[(g(\mathbf{x}_\star) - f(\mathbf{x}_\star))^2\right]$: The part of $\bar{E}_{\text{new}}$ that is due to the fact that, no matter how much training data is used, the model cannot represent the true f
- **Variance** $\mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[(\widehat{y}(\mathbf{x}_\star;\mathcal{T}))^2\right] - (g(\mathbf{x}_\star))^2\right]$: The part of $\bar{E}_{\text{new}}$ that is due to the variance in the training dataset

Intuitive interpretation:

- **Bias**: The inability of a method to describe the complicated patterns we would like it to describe. Low model complexity.
- **Variance**: How sensitive a method is to the training data. High model complexity.



Finding a balanced fit (neither over- nor underfit) is called the **the bias-variance tradeoff**.

UPPSALA UNIVERSITET

Regression example

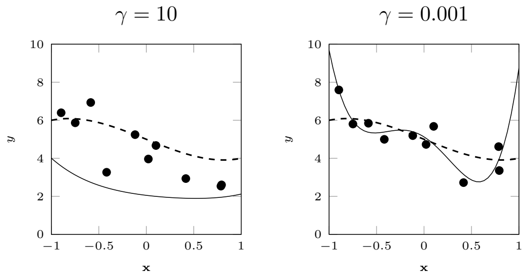
The data ($n = 10$) comes from

$$y = 5 - 2x + x^3 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1),$$

and our regression model is

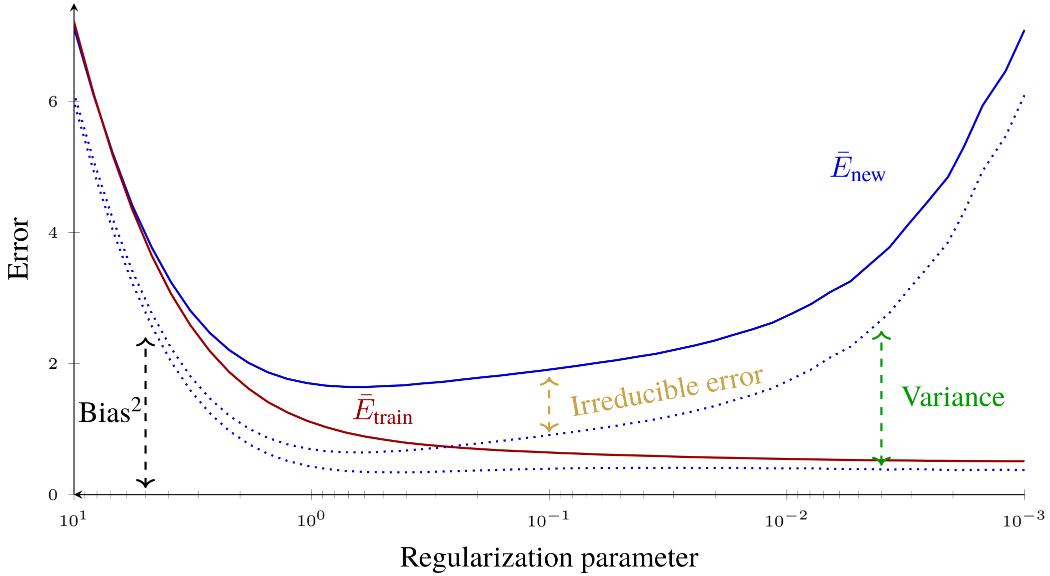
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \varepsilon.$$

We use ridge regression to tune model complexity/bias-variance.

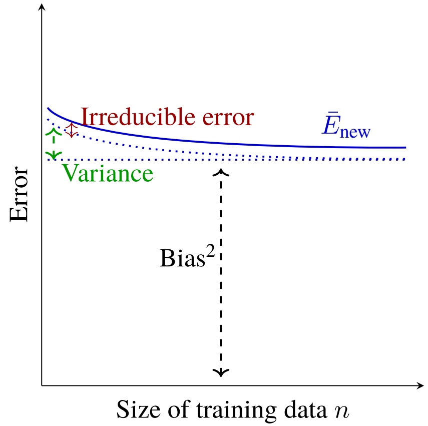


UPPSALA UNIVERSITET

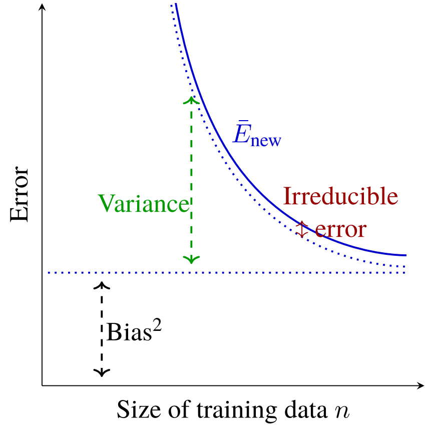
Regression example



Bias, variance and training data size



Low model complexity



High model complexity

Make the tradeoff

Some methods for **decreasing the model complexity/increasing the bias/ decreasing the variance**:

- Increase k in k-NN
- Regularization
- Bagging
- Early stopping (for methods trained using optimization, notably deep learning)
- Dropout (deep learning)

Warning! β_0 in linear regression (and later deep learning) is sometimes called “bias term”.
That is **completely unrelated** to bias in this context.

UPPSALA UNIVERSITET

Summary

- E_{new} = how good a method will perform "in production"
- We can *estimate* E_{new} well using cross-validation $\rightarrow$ practical tool for model selection (choosing between models, k in k-NN, regularization parameter, ...)
- We have made two decompositions of E_{new} (or technically its average $\bar{E}_{\text{new}}$):
  - $\bar{E}_{\text{new}} = \bar{E}_{\text{train}} +$ generalization error
  - $\bar{E}_{\text{new}} =$ Bias^2 + Variance + Irreducible error

UPPSALA UNIVERSITET

A few concepts to summarize lecture 5

$E(y, \widehat{y})$: Error function which compares predictions $\widehat{y}$ to true output y: MSE for regression, misclassification for classification.

E_{train}: The training data error (E_{train} small = the method fits the training data well).

E_{new}: The expected new data error; how well a method will perform when faced with an endless stream of new data.

Cross-validation: A method for estimating E_{new} using the training data.

Model complexity: How prone a method is to adapt to complicated patterns in the training data.

Overfitting: When a given method yields a smaller E_{train} and larger E_{new} than a model with lower model complexity would have done. That happens because the method/model is capturing patterns in the training data caused by random chance rather than true properties of the underlying function.

Bias: The inability of a method to describe the true patterns in the classification or regression problem. Low model complexity.

Variance: Sensitivity to random effects (noise) in the training data. High model complexity.