UPPSALA
UNIVERSITET

Statistical Machine Learning

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

UPPSALA
UNIVERSITET

David Sumpter
Division of Systems and Control
Department of Information Technology
Uppsala University

Summary of Lecture 4 (I/III)

Linear Discriminant Analysis (LDA) models the conditional class probabilities as

$$p(y = m \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, y = m) p(y = m)}{\sum_{j=1}^{M} p(\mathbf{x} \,|\, y = j) p(y = j)}.$$

where

- $P(y = m)$ estimated with n_m/n, is the *prior* probability of class m.
- $p(\mathbf{x} \,|\, y = m) = \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_m, \boldsymbol{\Sigma})$ is the probability density of $\mathbf{x}$ for an observation that comes from the mth class.

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

UPPSALA UNIVERSITET

Summary of Lecture 4 (II/III)

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

$$\widehat{\boldsymbol{\mu}}_m = \frac{1}{n_m} \sum_{i:y_i=m} \mathbf{x}_i, \qquad m = 1, \dots, M,$$
$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-M} \sum_{m=1}^{M} \sum_{i:y_i=j} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_m)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_m)^{\mathsf{T}}.$$

The LDA classifier assigns a test input $\mathbf{x}_\star$ to class m with the maximum predicted probability $p(y = m \,|\, \mathbf{x}_\star)$. LDA is a **linear classifier**.

Summary of Lecture 4 (III/III)

Parametric models are specified using a fixed-dimensional vector of parameters.

Non-parametric models allow the flexibility of the model to grow with data.

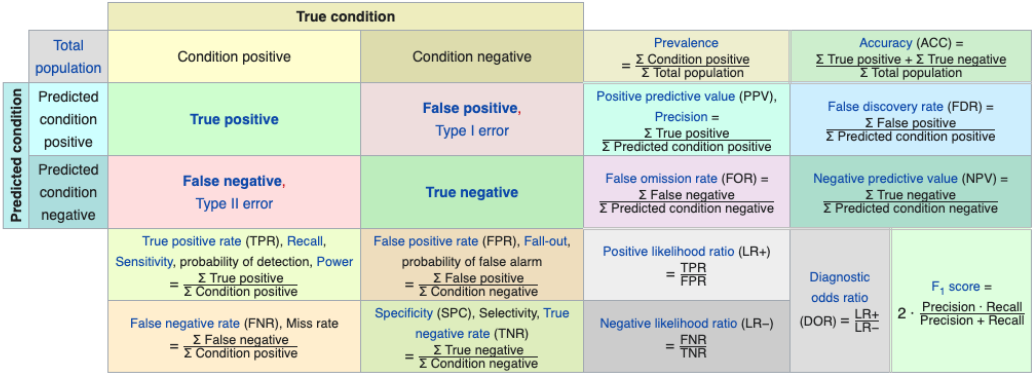
One non-parametric model is the **k-nearest neighbour classifier**.

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 $\mathbf{x}_i$ closest to $\mathbf{x}_\star$.
2. Classify $\mathbf{x}_\star$ according to a majority vote amongst these k training samples.

Evaluating a supervised machine learning method

The confusion matrix (again)



		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

Some commonly used error functions

- True positive rate (Recall): $\textbf{TPR} = \frac{\textbf{TP}}{\textbf{P}} = \frac{\textbf{TP}}{\textbf{FN} + \textbf{TP}} \in [0, 1]$

Some commonly used error functions

- True positive rate (Recall): $\textbf{TPR} = \frac{\textbf{TP}}{\textbf{P}} = \frac{\textbf{TP}}{\textbf{FN} + \textbf{TP}} \in [0, 1]$

- False positive rate: $\textbf{FPR} = \frac{\textbf{FP}}{\textbf{N}} = \frac{\textbf{FP}}{\textbf{FP} + \textbf{TN}} \in [0, 1]$

Some commonly used error functions

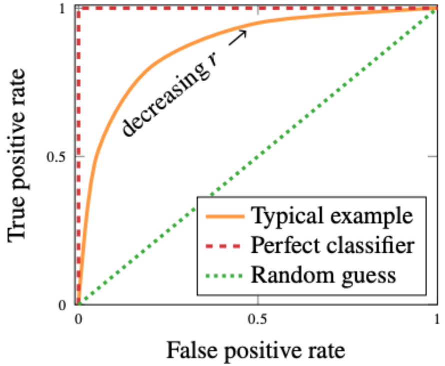
- True positive rate (Recall): $\textbf{TPR} = \frac{\textbf{TP}}{\textbf{P}} = \frac{\textbf{TP}}{\textbf{FN} + \textbf{TP}} \in [0, 1]$

- False positive rate: $\textbf{FPR} = \frac{\textbf{FP}}{\textbf{N}} = \frac{\textbf{FP}}{\textbf{FP} + \textbf{TN}} \in [0, 1]$

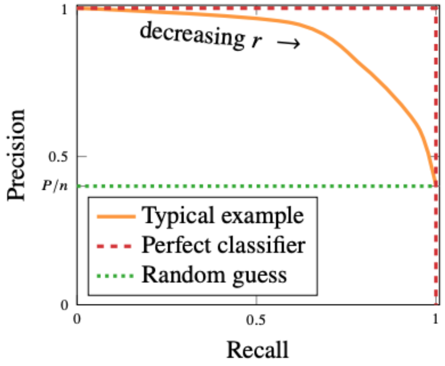
- Precision: $\textbf{Prec} = \frac{\textbf{TP}}{\textbf{P*}} = \frac{\textbf{TP}}{\textbf{FP} + \textbf{TP}} \in [0, 1]$

UPPSALA UNIVERSITET

AUC



(a) The ROC curve



(b) The precision-recall curve

Area Under Curve (AUC): condensed performance measure for the classifier, taking all possible thresholds into account (can apply to ROC and Precission-recall curve)

Loss functions are used in leaning

Training/learning: Minimizes the average loss on training data.

Examples (from maximising log- likelihood):

Linear regression: $L(\widehat{y}(\mathbf{x};\boldsymbol{\theta}), y) = (y_i - \widehat{y}(\mathbf{x};\boldsymbol{\theta})))^2\,.$

Logistic regression: $L(\widehat{y}(\mathbf{x};\boldsymbol{\theta}), y) = \log(1 + \exp(-y \cdot \widehat{y}(\mathbf{x};\boldsymbol{\theta}))).$

Some other examples:

Exponential loss: $L(y, \widehat{y}(\mathbf{x};\boldsymbol{\theta})) = \exp(-y \cdot \widehat{y}(\mathbf{x};\boldsymbol{\theta})).$

Hinge loss: $L(y, \widehat{y}(\mathbf{x};\boldsymbol{\theta})) = \begin{cases} 1 - y \cdot \widehat{y}(\mathbf{x};\boldsymbol{\theta}) & \text{for } y \cdot \widehat{y}(\mathbf{x};\boldsymbol{\theta}) < 1, \\ 0 & \text{otherwise.} \end{cases}$

Loss functions differ between methods and the values can not directly be compared.

UPPSALA UNIVERSITET

Error functions allow us to evaluate performance

We want to be able to compare different methods (LDA, logistic regression, neural net).

We would like to evaluate already trained models with respect to hyperparameters (eg. regularization parameter γ or k in k-NN)

Error function

$$E\big(\widehat{y}(\mathbf{x}), y\big) = \begin{cases} \mathbb{I}\{\widehat{y}(\mathbf{x}) = y\} & \text{(classification)} \\ \text{False Postive rate} & \text{(classification)} \\ \text{Area Under the Curve (AUC)} & \text{(classification)} \\ \big(\widehat{y}(\mathbf{x}) - y\big)^2 & \text{(regression)} \end{cases}$$

Error function not necessarily the same as loss function (but can be).

Evaluating performance

Let $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ be training data.

Training error

$$E_{\text{train}} = \frac{1}{n}\sum_{i=1}^{n} E\big(\widehat{y}(\mathbf{x}_i), y_i\big)$$

Measures how well a predictor $\widehat{y}$ performs on training data, but we are interested in new data. Let $p(\mathbf{x}, y)$ be the joint distribution over data.

Evaluating performance

Let $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ be training data.

Training error

$$E_{\text{train}} = \frac{1}{n}\sum_{i=1}^{n} E\big(\widehat{y}(\mathbf{x}_i), y_i\big)$$

Measures how well a predictor $\widehat{y}$ performs on training data, but we are interested in new data. Let $p(\mathbf{x}, y)$ be the joint distribution over data.

Expected new data error

$$E_{\text{new}} = \mathbb{E}_\star\left[E\big(\widehat{y}(x_\star), y_\star\big)\right] = \int E\big(\widehat{y}(x_\star), y_\star\big)p(x_\star, y_\star)dx_\star dy_\star$$

Impossible to compute since $p(\mathbf{x}, y)$ is unknown, but minimizing E_{new} is our ultimate goal.

UPPSALA UNIVERSITET

Evaluating performance

Let $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ be training data.

Training error

$$E_{\text{train}} = \frac{1}{n} \sum_{i=1}^{n} E\big(\widehat{y}(\mathbf{x}_i), y_i\big)$$

Measures how well a predictor $\widehat{y}$ performs on training data, but we are interested in new data. Let $p(\mathbf{x}, y)$ be the joint distribution over data.

Expected new data error

$$E_{\text{new}} = \mathbb{E}_\star \left[E\big(\widehat{y}(x_\star), y_\star\big)\right] = \int E\big(\widehat{y}(x_\star), y_\star\big) p(x_\star, y_\star) dx_\star dy_\star$$

Impossible to compute since $p(\mathbf{x}, y)$ is unknown, but minimizing E_{new} is our ultimate goal.

Maybe we can learn it?

UPPSALA UNIVERSITET

Approximating integrals

By ’ergodicity’, we can approximate integrals using samples:

$$\mathbb{E}\left[h(\mathbf{x})\right] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i), \quad \mathbf{x}_i \overset{\text{i.i.d.}}{\sim} p(\mathbf{x}_i),\ i = 1, \dots, n$$

With samples from $p(\mathbf{x}, y)$, we can estimate E_{new}!

Note: Important that samples come from real world, “in-production” distribution.

UPPSALA UNIVERSITET

Approximating integrals

By 'ergodicity', we can approximate integrals using samples:

$$\mathbb{E}\left[h(\mathbf{x})\right] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i), \quad \mathbf{x}_i \overset{\text{i.i.d.}}{\sim} p(\mathbf{x}_i),\ i = 1, \ldots, n$$

With samples from $p(\mathbf{x}, y)$, we can estimate E_{new}!

Note: Important that samples come from real world, "in-production" distribution.

Training data are (or should be) n samples from $p(\mathbf{x}, y)$.

Approximate $E_{\text{new}} \overset{?}{\approx} E_{\text{train}}$?

Approximating integrals

By 'ergodicity', we can approximate integrals using samples:

$$\mathbb{E}\left[h(\mathbf{x})\right] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i), \quad \mathbf{x}_i \overset{\text{i.i.d.}}{\sim} p(\mathbf{x}_i),\ i = 1, \dots, n$$

With samples from $p(\mathbf{x}, y)$, we can estimate E_{new}!

Note: Important that samples come from real world, "in-production" distribution.

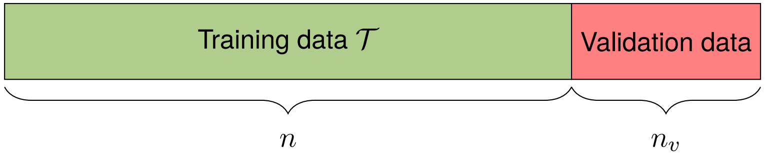
Training data are (or should be) n samples from $p(\mathbf{x}, y)$.

Approximate $E_{\text{new}} \overset{?}{\approx} E_{\text{train}}$? **NO!**

Training data is part of the predictor $\widehat{y}(\mathbf{x}; \mathcal{T})$!

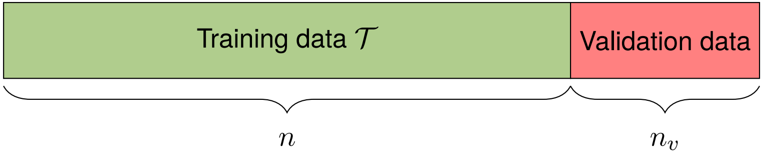
Estimating E_{new}: hold-out validation data

Split data in **training data** and **validation data**.



Estimating E_{new}: hold-out validation data

Split data in **training data** and **validation data**.

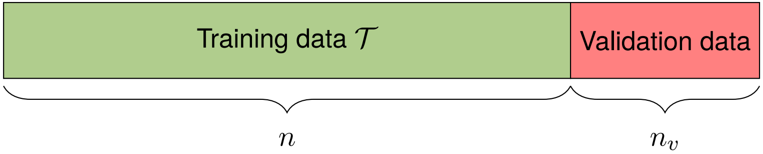


Hold-out validation error:

$$E_{\text{new}} \approx E_{\text{hold-out}} = \frac{1}{n_v} \sum_{i=1}^{n_v} E\big(\widehat{y}(\mathbf{x}_i; \mathcal{T}), y_i\big)$$

Estimating E_{new}: hold-out validation data

Split data in **training data** and **validation data**.



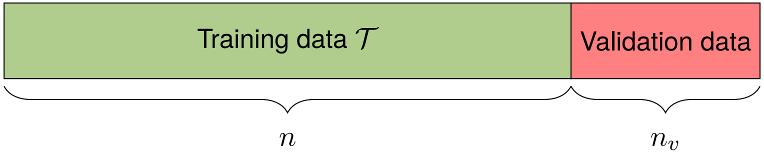
Hold-out validation error:

$$E_{\text{new}} \approx E_{\text{hold-out}} = \frac{1}{n_v} \sum_{i=1}^{n_v} E\big(\widehat{y}(\mathbf{x}_i; \mathcal{T}), y_i\big)$$

+ Simple
! Good estimate of E_{new} requires the validation set to be large
! Good predictor $\widehat{y}$ requires the training set to be large
– Not all data is used for learning

Estimating E_{new}: hold-out validation data

Split data in **training data** and **validation data**.



Hold-out validation error:

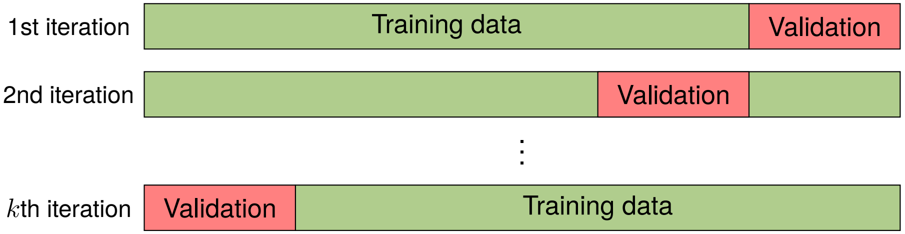
$$E_{\text{new}} \approx E_{\text{hold-out}} = \frac{1}{n_v} \sum_{i=1}^{n_v} E\big(\widehat{y}(\mathbf{x}_i; \mathcal{T}), y_i\big)$$

- + Simple
- ! Good estimate of E_{new} requires the validation set to be large
- ! Good predictor $\widehat{y}$ requires the training set to be large
- – Not all data is used for learning

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

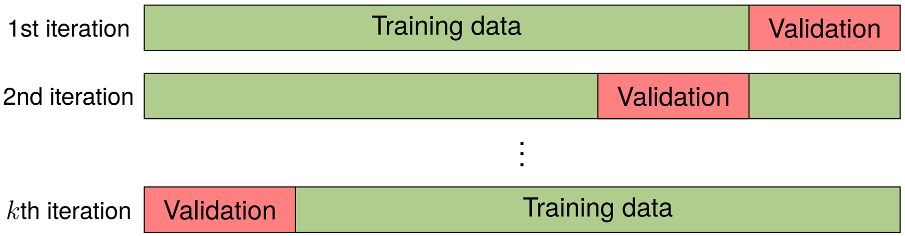
Estimating E_{new}: k-fold cross-validation

Split data in k batches and hold out batch ℓ when estimating model. Use batch ℓ to estimate E_{new} and average over all k estimates. Estimate final model using whole dataset.



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

Split data in k batches and hold out batch ℓ when estimating model. Use batch ℓ to estimate E_{new} and average over all k estimates. Estimate final model using whole dataset.



k-fold cross-validation error

$$E_{\text{new}} \approx E_{\text{k-fold}} = \frac{1}{k}\sum_{\ell=1}^{k} E_{\text{hold-out}}^{(\ell)}$$

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

Using a test set

An important use of $E_{\text{k-fold}}$ is to choose between models or select hyperparameters (k in k-NN or λ in regularization). If a good estimate of E_{new} is important:

We can no longer use $E_{\text{k-fold}}$ to estimate E_{new}!

Set aside a test set and use **only** to esimate E_{new}.

Flavors of cross-validation

- k-fold cross-validation: Typically $k \approx 10$
- Leave-one-out cross-validation: k-fold cross-validation with $k = n$
- Monte Carlo cross-validation: Random selection of validation set at each iteration

UPPSALA UNIVERSITET

Understanding E_{new}

Collecting data is a random process where $\mathcal{T}$ is sampled from $p(\mathbf{x}, y)$. Because $\mathcal{T}$ is random, so is our learned model $\widehat{y}(\mathbf{x}; \mathcal{T})$.

To better understand the behavior of E_{new}, we need to introduce

$$\begin{aligned}\bar{E}_{\text{train}} &= \mathbb{E}_{\mathcal{T}}\left[E_{\text{train}}\right], \\ \bar{E}_{\text{new}} &= \mathbb{E}_{\mathcal{T}}\left[E_{\text{new}}\right],\end{aligned}$$

where $\mathbb{E}_{\mathcal{T}}\left[\cdot\right]$ is the average over **training data** $\mathcal{T}$.

Note: k-fold cross-validation estimates $\bar{E}_{new}$ rather than E_{new}.

Understanding E_{new}

Collecting data is a random process where $\mathcal{T}$ is sampled from $p(\mathbf{x}, y)$. Because $\mathcal{T}$ is random, so is our learned model $\widehat{y}(\mathbf{x}; \mathcal{T})$.

To better understand the behavior of E_{new}, we need to introduce

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

where $\mathbb{E}_{\mathcal{T}}\left[\cdot\right]$ is the average over **training data** $\mathcal{T}$.

Note: k-fold cross-validation estimates $\bar{E}_{new}$ rather than E_{new}.

It usually holds that

$$\bar{E}_{\text{train}} < \bar{E}_{\text{new}}.$$

On average, a method usually performs better on training data than new data.

Model complexity

Since $\bar{E}_{\text{train}} < \bar{E}_{\text{new}}$, define generalization gap:

$$\bar{E}_{\text{new}} = \bar{E}_{\text{train}} + \textbf{generalization gap}$$

UPPSALA UNIVERSITET

Model complexity

Since $\bar{E}_{\text{train}} < \bar{E}_{\text{new}}$, define generalization gap:

$$\bar{E}_{\text{new}} = \bar{E}_{\text{train}} + \textbf{generalization gap}$$

A models ability to adapt to patterns in the data, we call the **model complexity**[1].

Model complexity ↗

- ↘ $\bar{E}_{\text{train}}$
- ↗ Generalization gap

Model complexity ↘

- ↗ $\bar{E}_{\text{train}}$
- ↘ Generalization gap

UPPSALA UNIVERSITET

Model complexity

Since $\bar{E}_{\text{train}} < \bar{E}_{\text{new}}$, define generalization gap:

$$\bar{E}_{\text{new}} = \bar{E}_{\text{train}} + \textbf{generalization gap}$$

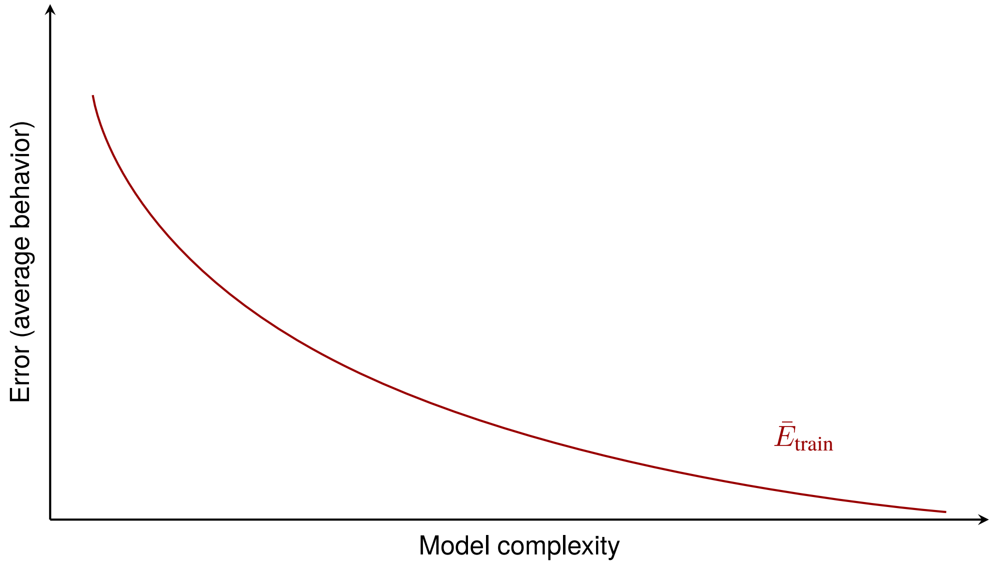
A models ability to adapt to patterns in the data, we call the **model complexity**[1].

Model complexity $\nearrow$
- $\searrow$ $\bar{E}_{\text{train}}$
- $\nearrow$ Generalization gap

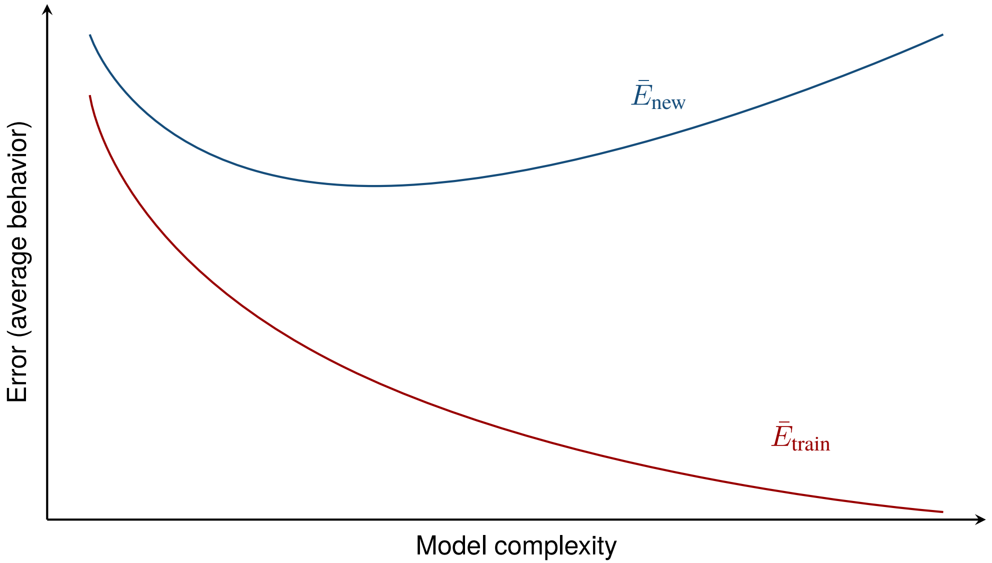
Model complexity $\searrow$
- $\nearrow$ $\bar{E}_{\text{train}}$
- $\searrow$ Generalization gap

$\bar{E}_{\text{new}}$ usually attains a minimum at some intermediate complexity.

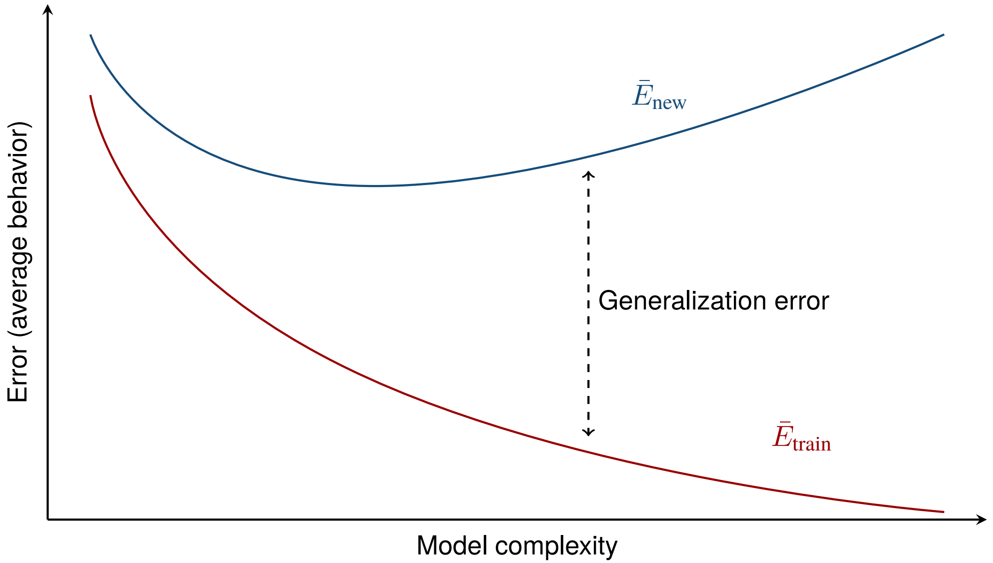
Model complexity, $\bar{E}_{\text{train}}$ and $\bar{E}_{\text{new}}$



Model complexity, $\bar{E}_{\text{train}}$ and $\bar{E}_{\text{new}}$

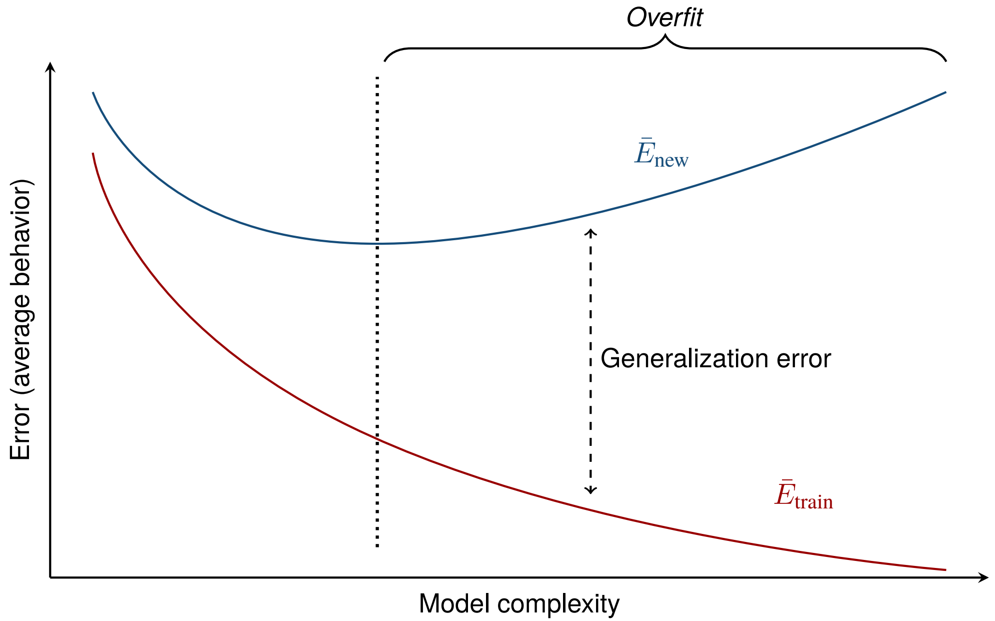


Model complexity, $\bar{E}_{\text{train}}$ and $\bar{E}_{\text{new}}$

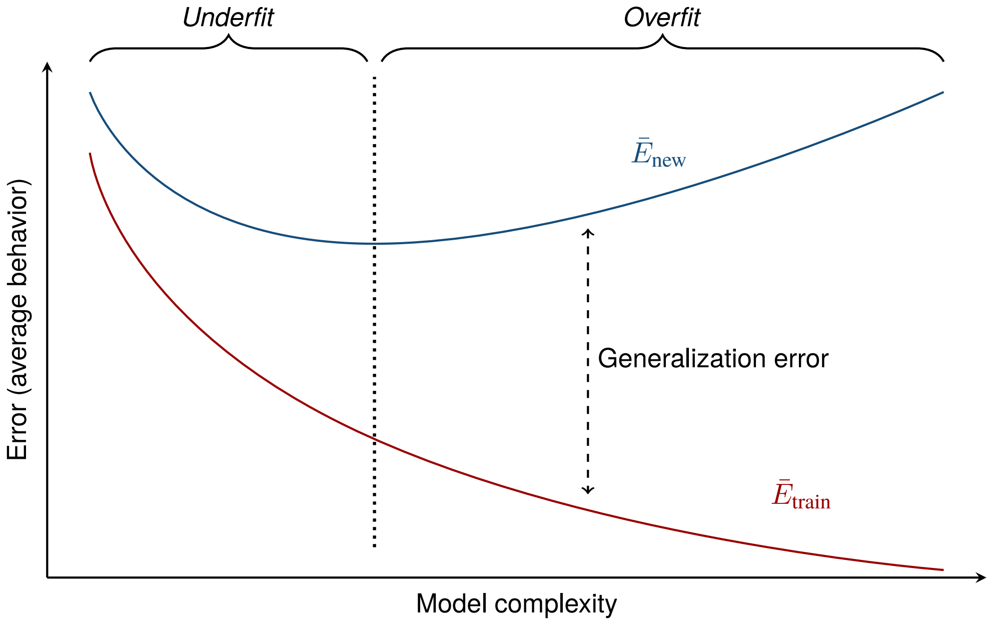


UPPSALA UNIVERSITET

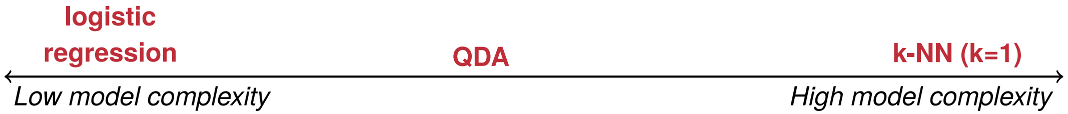
Model complexity, $\bar{E}_{\text{train}}$ and $\bar{E}_{\text{new}}$



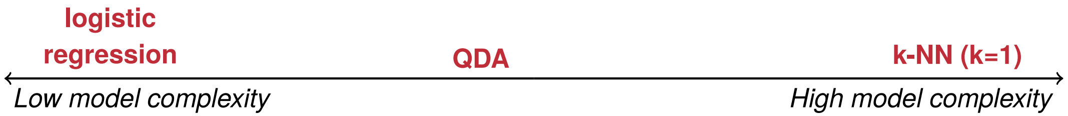
Model complexity, $\bar{E}_{\text{train}}$ and $\bar{E}_{\text{new}}$



Example: cross-validation for model selection



Example: cross-validation for model selection

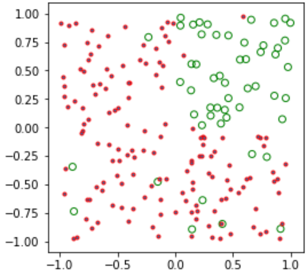


All models are wrong, but some are useful. — George Box

Example: cross-validation for model selection

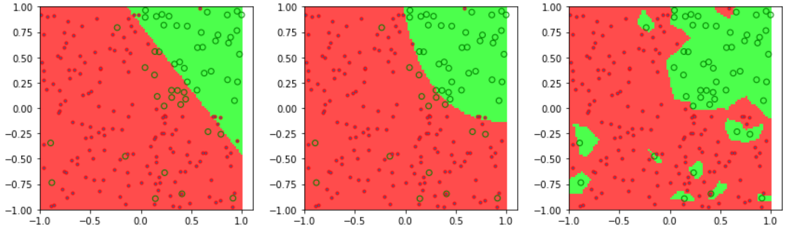
Binary classification, $p = 2$.
Evaluate the following methods

- logistic regression
- QDA
- k-NN



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

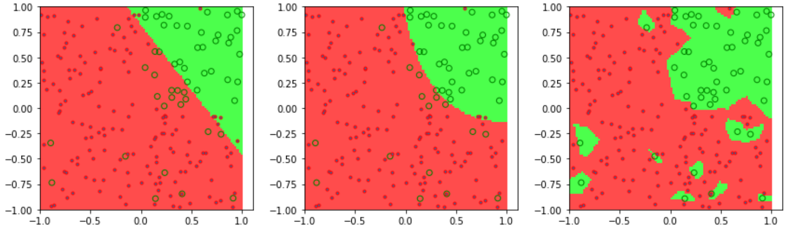
Example: cross-validation for model selection



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

UPPSALA UNIVERSITET

Example: cross-validation for model selection

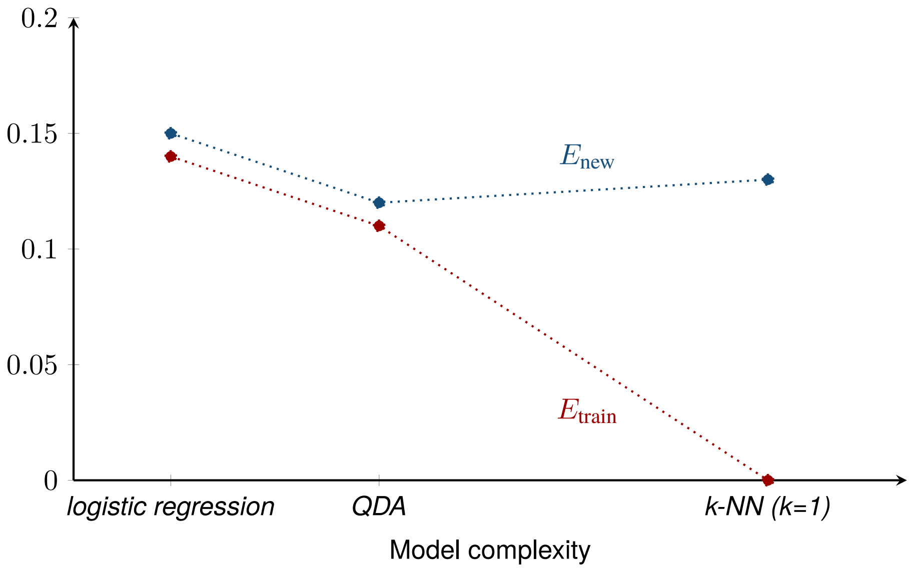


	logistic regression	QDA	k-NN ($k = 1$)
E_{train}	0.14	0.11	0.0
E_{new}	0.15	0.12	0.13

E_{new} is estimated using cross-validation.

UPPSALA UNIVERSITET

Example: cross-validation for model selection



Finding a target

Imagine z_0 as being the (true) position of an object, and z of being noisy GPS measurements of that position. The average of our measurements is $\mathbb{E}[z] = \bar{z}$. We can now define

$$\text{Bias: } \bar{z} - z_0 \tag{1a}$$

$$\text{Variance: } \mathbb{E}\left[(z-\bar{z})^2\right] = \mathbb{E}\left[z^2\right] - \bar{z}^2. \tag{1b}$$

For every measurement our error is can be measured by the expected squared error between z and z_0. That is

$$\begin{aligned}\mathbb{E}\left[(z-z_0)^2\right] &= \mathbb{E}\left[\left((z-\bar{z}) + (\bar{z}-z_0)\right)^2\right] = \\ &= \underbrace{\mathbb{E}\left[(z-\bar{z})^2\right]}_{\text{Variance}} + 2\underbrace{\left(\mathbb{E}[z]-\bar{z}\right)}_{0}(\bar{z}-z_0) + \underbrace{(\bar{z}-z_0)^2}_{\text{bias}^2}.\end{aligned} \tag{2}$$

In practice, we often don't know which source of error (the variance or the bias) is most important.

Applying ’finding a target’ to machine learning

Assume that ”the real world” works as

$$y = f_0(\mathbf{x}) + \epsilon, \text{ where } \begin{cases} \epsilon \text{ random, independent of } \mathbf{x} \\ \mathbb{E}[\epsilon] = 0, \\ \mathbb{E}[\epsilon^2] = \sigma^2. \end{cases}$$

Denote the **average trained model**

$$\bar{f}(\mathbf{x}) \triangleq \mathbb{E}_{\mathcal{T}}\left[\widehat{y}(\mathbf{x}; \mathcal{T})\right].$$

The average model if we could re-train the model on new data an infinite number of times.

Bias-variance decomposition

$$\begin{aligned}\bar{E}_{\text{new}} &= \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{\star}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - y_\star\right)^2\right]\right] = \mathbb{E}_{\star}\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - f_0(\mathbf{x}_\star) - \epsilon\right)^2\right]\right] \\ &= \mathbb{E}_{\star}\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - \bar{f}(\mathbf{x}_\star) + \bar{f}(\mathbf{x}_\star) - f_0(\mathbf{x}_\star) - \epsilon\right)^2\right]\right] \\ &= \underbrace{\mathbb{E}_{\star}\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - \bar{f}(\mathbf{x}_\star)\right)^2\right]\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{\star}\left[\left(\bar{f}(\mathbf{x}_\star) - f_0(\mathbf{x}_\star)\right)^2\right]}_{\text{Bias}^2} + \underbrace{\sigma^2}_{\text{Irreducible error}}\end{aligned}$$

Technical interpretation:

- **Bias^2** $\mathbb{E}_{\star}\left[\left(\bar{f}(\mathbf{x}_\star) - f_0(\mathbf{x}_\star)\right)^2\right]$: The part of $\bar{E}_{\text{new}}$ that is due to the fact that the model cannot represent the true f_0.
- **Variance** $\mathbb{E}_{\star}\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - \bar{f}(\mathbf{x}_\star)\right)^2\right]\right]$: The part of $\bar{E}_{\text{new}}$ that is due to the variability in the training dataset.

UPPSALA UNIVERSITET

Bias-variance decomposition

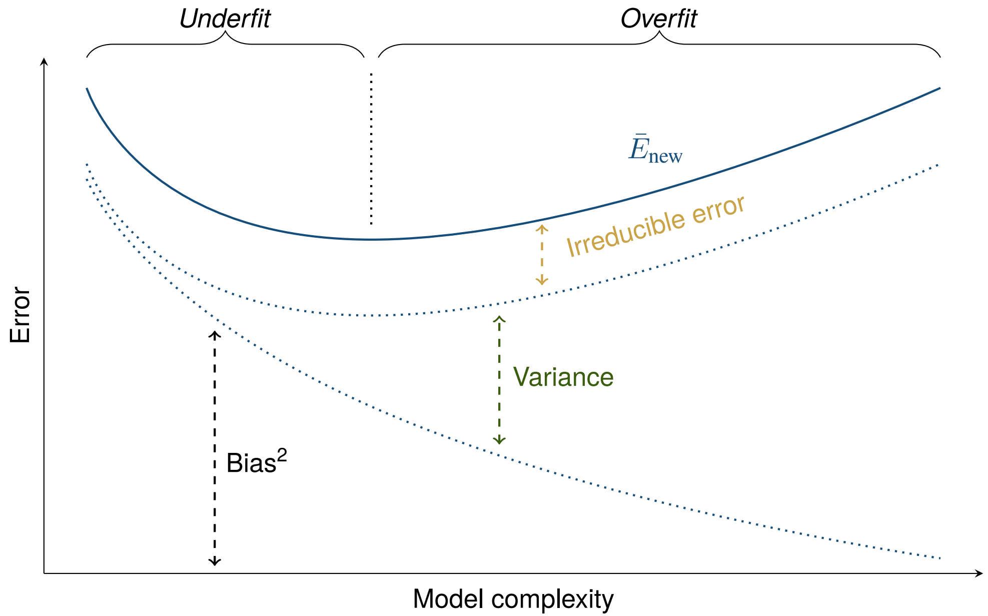
$$
\begin{aligned}
\bar{E}_{\text{new}} &= \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_\star\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - y_\star\right)^2\right]\right] = \mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - f_0(\mathbf{x}_\star) - \epsilon\right)^2\right]\right] \\
&= \mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - \bar{f}(\mathbf{x}_\star) + \bar{f}(\mathbf{x}_\star) - f_0(\mathbf{x}_\star) - \epsilon\right)^2\right]\right] \\
&= \underbrace{\mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - \bar{f}(\mathbf{x}_\star)\right)^2\right]\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_\star\left[\left(\bar{f}(\mathbf{x}_\star) - f_0(\mathbf{x}_\star)\right)^2\right]}_{\text{Bias}^2} + \underbrace{\sigma^2}_{\text{Irreducible error}}
\end{aligned}
$$

Technical interpretation:

- **Bias²** $\mathbb{E}_\star\left[\left(\bar{f}(\mathbf{x}_\star) - f_0(\mathbf{x}_\star)\right)^2\right]$: The part of $\bar{E}_{\text{new}}$ that is due to the fact that the model cannot represent the true f_0.
- **Variance** $\mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[\left(\widehat{y}(\mathbf{x}_\star;\mathcal{T}) - \bar{f}(\mathbf{x}_\star)\right)^2\right]\right]$: The part of $\bar{E}_{\text{new}}$ that is due to the variability 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

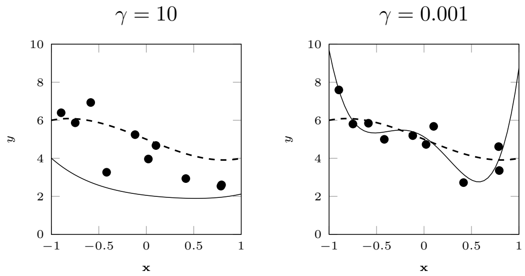
The data ($n = 10$) comes from

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

and our regression model is

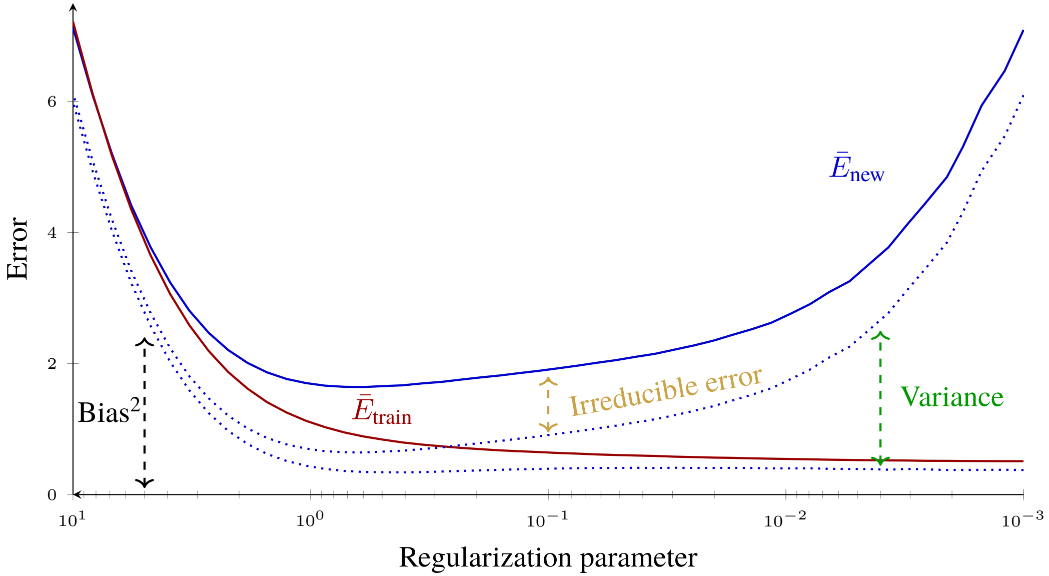
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \varepsilon.$$

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

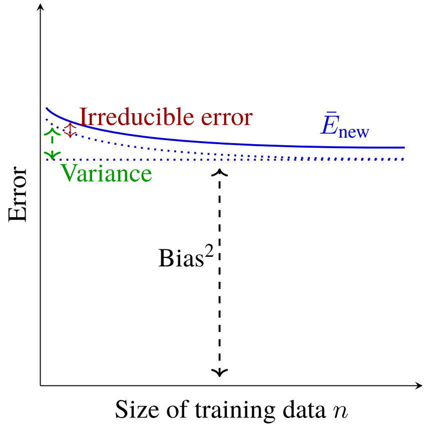


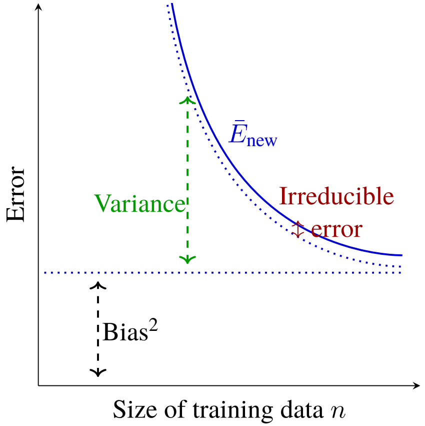
UPPSALA UNIVERSITET

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)

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! θ_0 in linear regression (and later deep learning) is sometimes called “bias term”. That is **completely unrelated** to bias in this context.

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.