Non-linear models. Basis expansion. Overfitting. Regularization.

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

When a linear model is not enough…

Basis expansion

a.k.a. **feature space straightening**.

Basis expansion

a.k.a. **feature space straightening**.

Why?

- Linear decision boundary (or linear regression model) may not be flexible enough to perform accurate classification (regression).
- The algorithms for fitting linear models can be used to fit (certain type of) *non-linear models*!

Basis expansion

a.k.a. **feature space straightening**.

Why?

- Linear decision boundary (or linear regression model) may not be flexible enough to perform accurate classification (regression).
- The algorithms for fitting linear models can be used to fit (certain type of) *non-linear models*!

How?

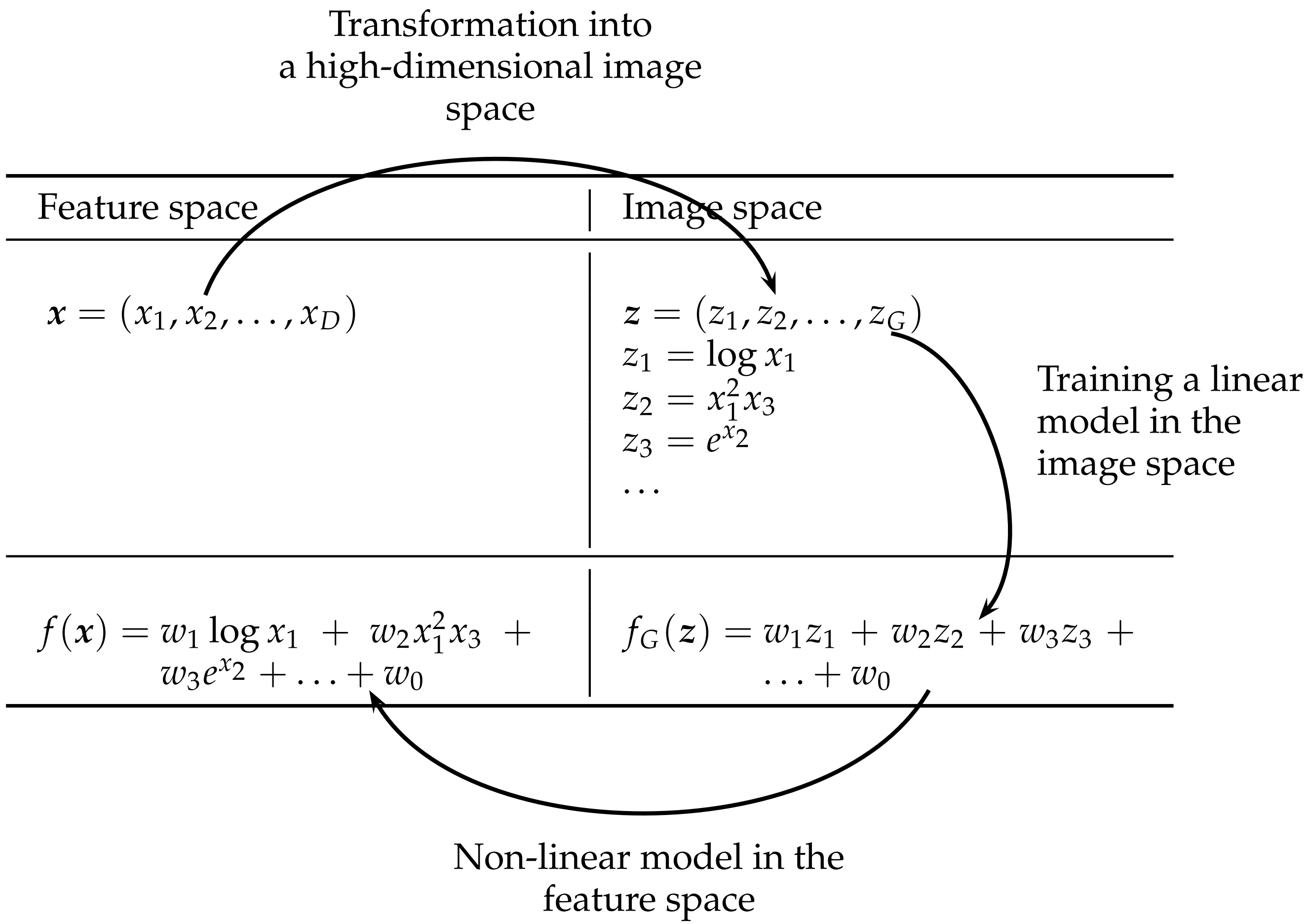
- Let's define a new multidimensional image space F.
- Feature vectors $\boldsymbol{x}$ are transformed into this image space F (new features are derived) using mapping Φ:

$$\begin{aligned} \boldsymbol{x} &\rightarrow \boldsymbol{z} = \Phi(\boldsymbol{x}), \\ \boldsymbol{x} = (x_1, x_2, \ldots, x_D) &\rightarrow \boldsymbol{z} = (\Phi_1(\boldsymbol{x}), \Phi_2(\boldsymbol{x}), \ldots, \Phi_G(\boldsymbol{x})), \end{aligned}$$

  while usually $D \ll G$.
- In the image space, a linear model is trained. However, this is equivalent to training a non-linear model in the original space.

$$\begin{aligned} f_G(\boldsymbol{z}) &= w_1 z_1 + w_2 z_2 + \ldots + w_G z_G + w_0 \\ f(\boldsymbol{x}) = f_G(\Phi(\boldsymbol{x})) &= w_1 \Phi_1(\boldsymbol{x}) + w_2 \Phi_2(\boldsymbol{x}) + \ldots + w_G \Phi_G(\boldsymbol{x}) + w_0 \end{aligned}$$

Two coordinate systems



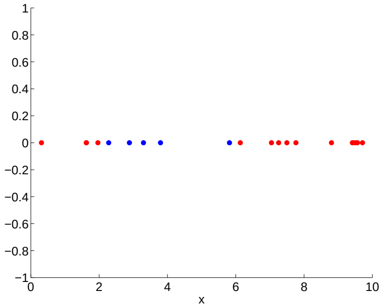
Transformation into a high-dimensional image space

Feature space	Image space
$\boldsymbol{x} = (x_1, x_2, \dots, x_D)$	$\boldsymbol{z} = (z_1, z_2, \dots, z_G)$ $z_1 = \log x_1$ $z_2 = x_1^2 x_3$ $z_3 = e^{x_2}$ $\dots$
$f(\boldsymbol{x}) = w_1 \log x_1 + w_2 x_1^2 x_3 + w_3 e^{x_2} + \dots + w_0$	$f_G(\boldsymbol{z}) = w_1 z_1 + w_2 z_2 + w_3 z_3 + \dots + w_0$

Training a linear model in the image space

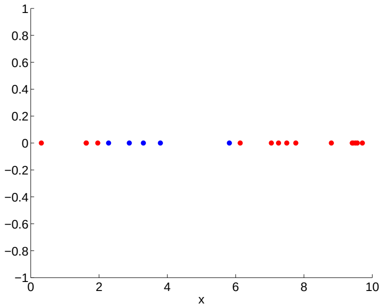
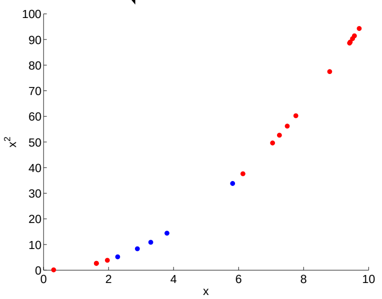
Non-linear model in the feature space

Two coordinate systems: simple graphical example

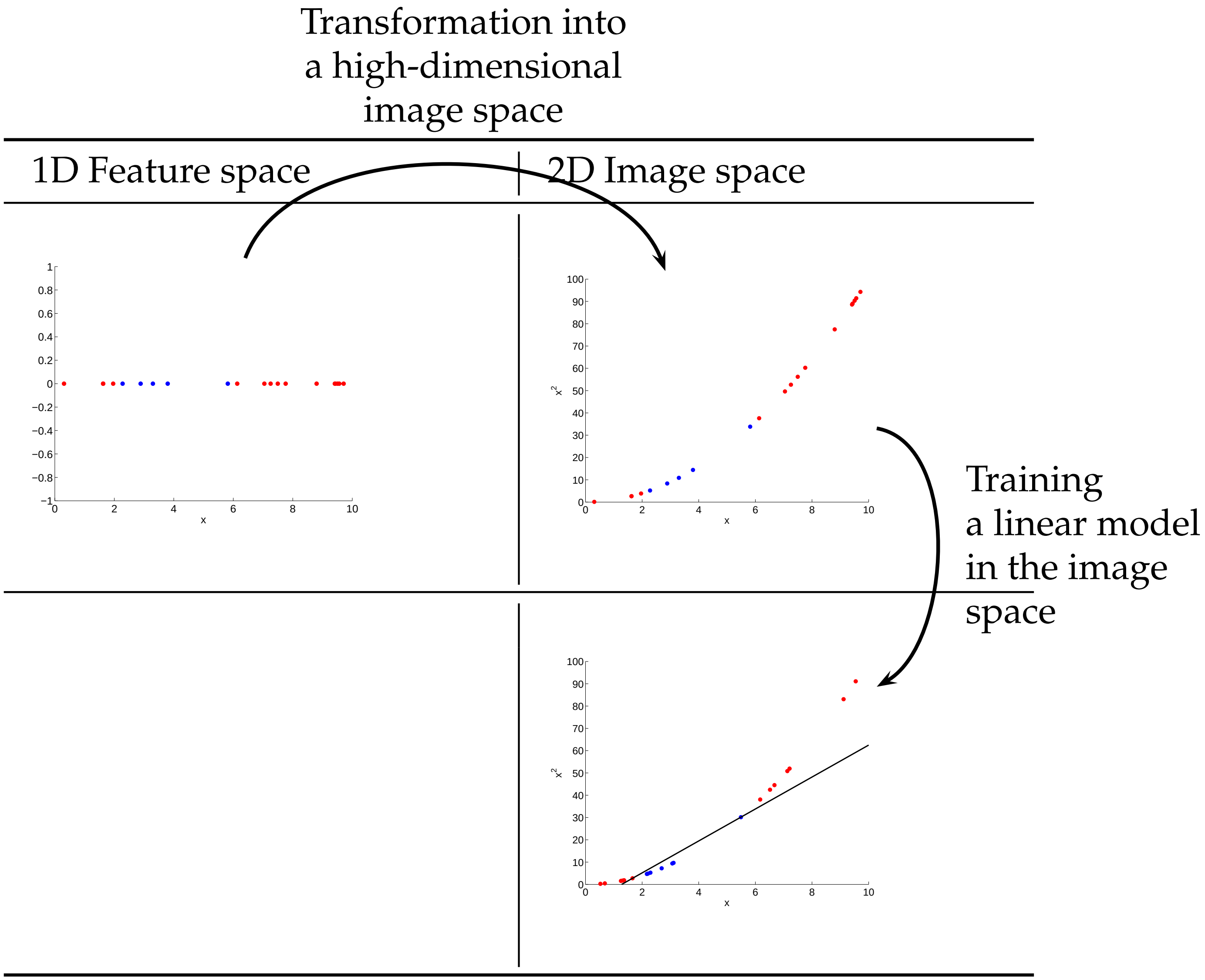
1D Feature space	2D Image space
	

Two coordinate systems: simple graphical example

Transformation into
a high-dimensional
image space

1D Feature space	2D Image space
	

Two coordinate systems: simple graphical example



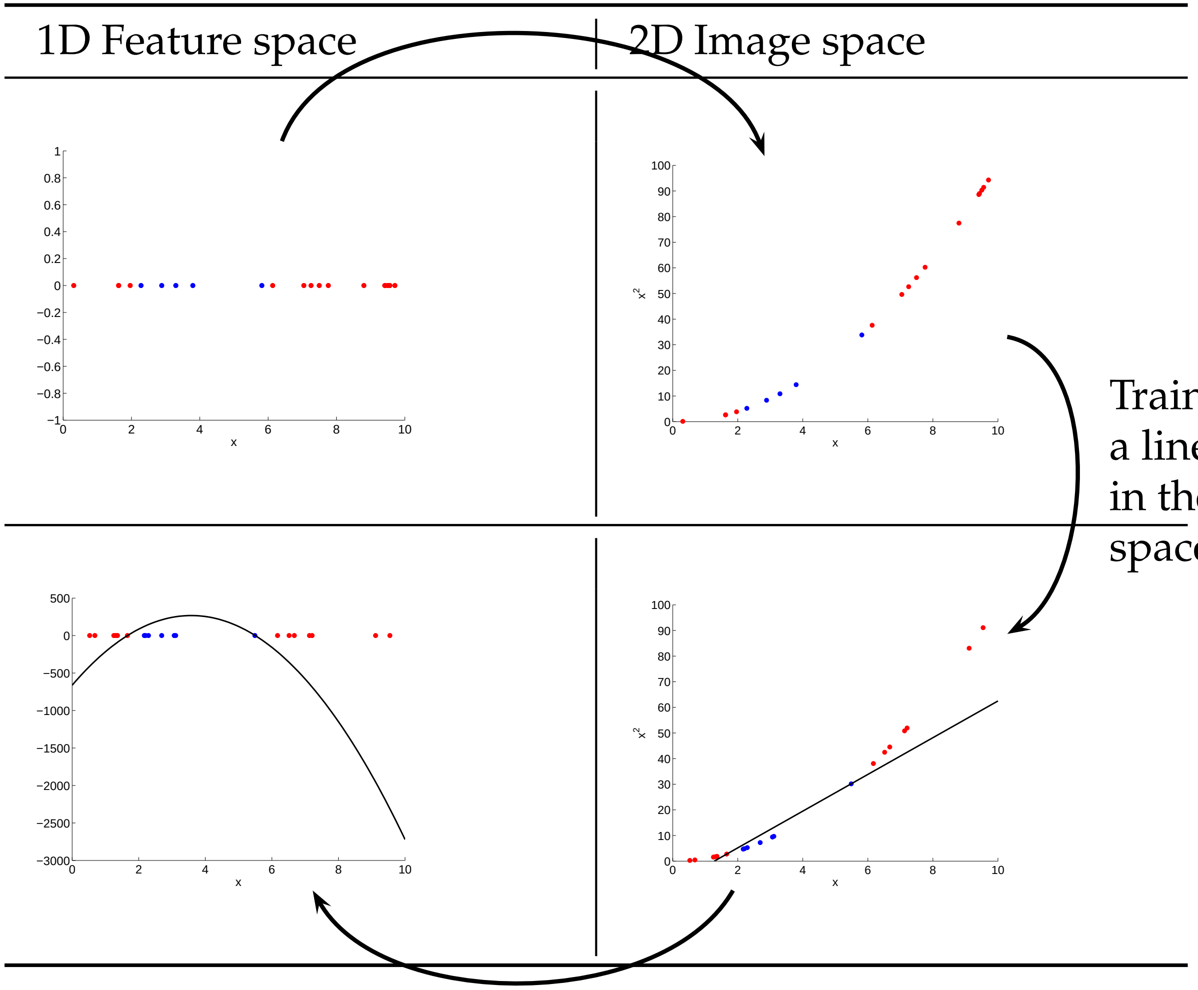
Transformation into a high-dimensional image space

1D Feature space | 2D Image space

Training a linear model in the image space

Two coordinate systems: simple graphical example

Transformation into
a high-dimensional
image space



1D Feature space

2D Image space

Training
a linear model
in the image
space

Non-linear model in the
feature space

Basis expansion: remarks

Advantages:

- Universal, generally usable method.

Disadvantages:

- We must define what new features shall form the high-dimensional space F.
- The examples must be really transformed into the high-dimensional space F.
- When too much derived features is used, the resulting models are prone to overfitting (see next slides).

For certain type of algorithms, there is a method how to perform the basis expansion without actually carrying out the mapping! (See the next lecture.)

How to evaluate a predictive model?

Model evaluation

Fundamental question: What is a good measure of "model quality" from the machine-learning standpoint?

Model evaluation

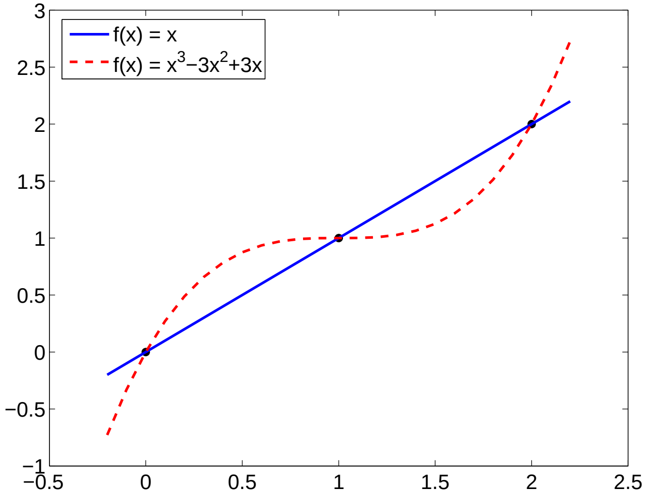
Fundamental question: What is a good measure of "model quality" from the machine-learning standpoint?

- We have various measures of model error:
  - For regression tasks: MSE, MAE, . . .
  - For classification tasks: misclassification rate, measures based on confusion matrix, . . .
- Some of them can be regarded as finite approximations of the *Bayes risk*.
- Are these functions *good approximations* when measured on the data the models were trained on?

Model evaluation

Fundamental question: What is a good measure of "model quality" from the machine-learning standpoint?

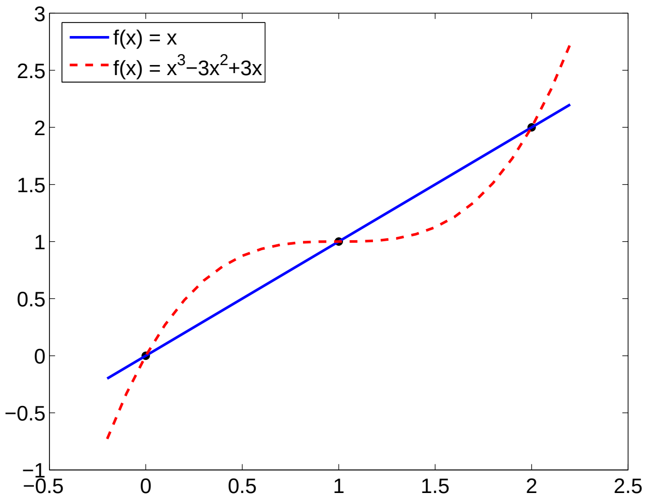
- We have various measures of model error:
  - For regression tasks: MSE, MAE, ...
  - For classification tasks: misclassification rate, measures based on confusion matrix, ...
- Some of them can be regarded as finite approximations of the *Bayes risk*.
- Are these functions *good approximations* when measured on the data the models were trained on?



Model evaluation

Fundamental question: What is a good measure of "model quality" from the machine-learning standpoint?

- We have various measures of model error:
  - For regression tasks: MSE, MAE, ...
  - For classification tasks: misclassification rate, measures based on confusion matrix, ...
- Some of them can be regarded as finite approximations of the *Bayes risk*.
- Are these functions *good approximations* when measured on the data the models were trained on?

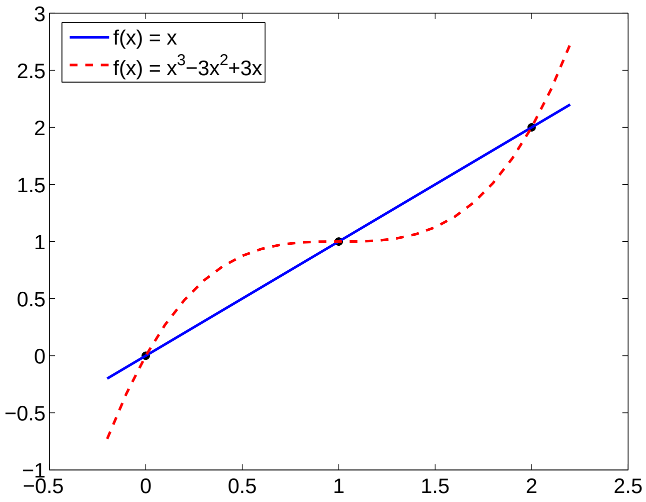


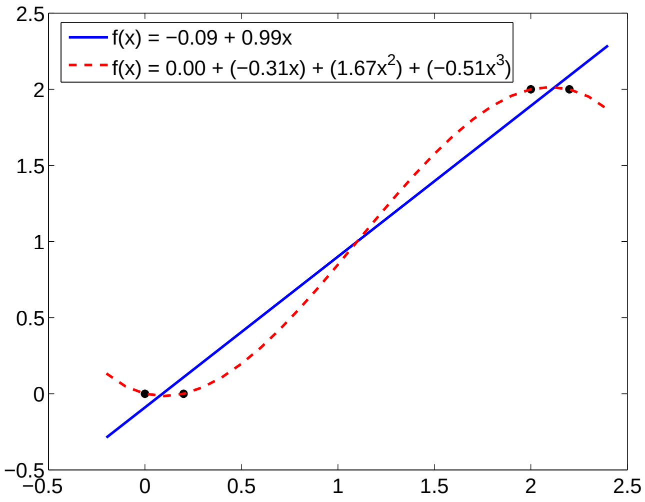
Using MSE only, both models are equivalent!!!

Model evaluation

Fundamental question: What is a good measure of “model quality” from the machine-learning standpoint?

- We have various measures of model error:
  - For regression tasks: MSE, MAE, . . .
  - For classification tasks: misclassification rate, measures based on confusion matrix, . . .
- Some of them can be regarded as finite approximations of the *Bayes risk*.
- Are these functions *good approximations* when measured on the data the models were trained on?



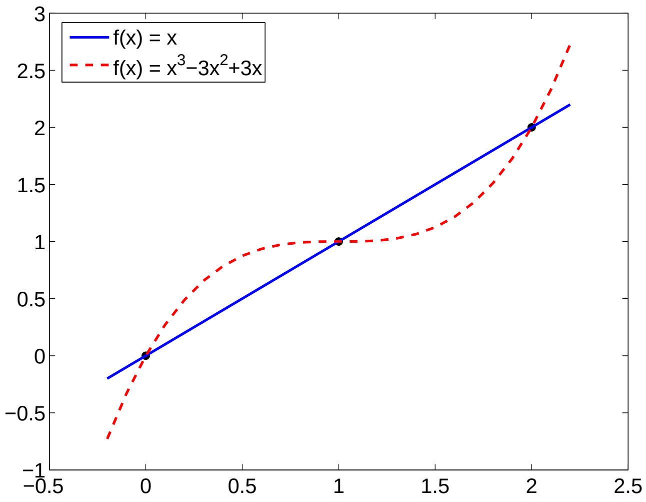


Using MSE only, both models are equivalent!!!

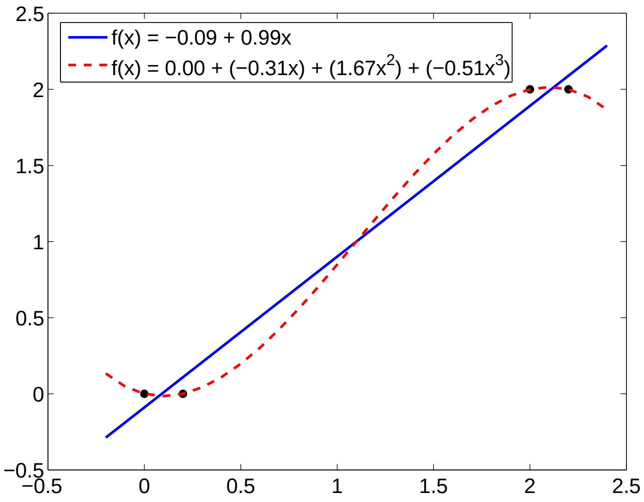
Model evaluation

Fundamental question: What is a good measure of “model quality” from the machine-learning standpoint?

- We have various measures of model error:
  - For regression tasks: MSE, MAE, . . .
  - For classification tasks: misclassification rate, measures based on confusion matrix, . . .
- Some of them can be regarded as finite approximations of the *Bayes risk*.
- Are these functions *good approximations* when measured on the data the models were trained on?



Using MSE only, both models are equivalent!!!

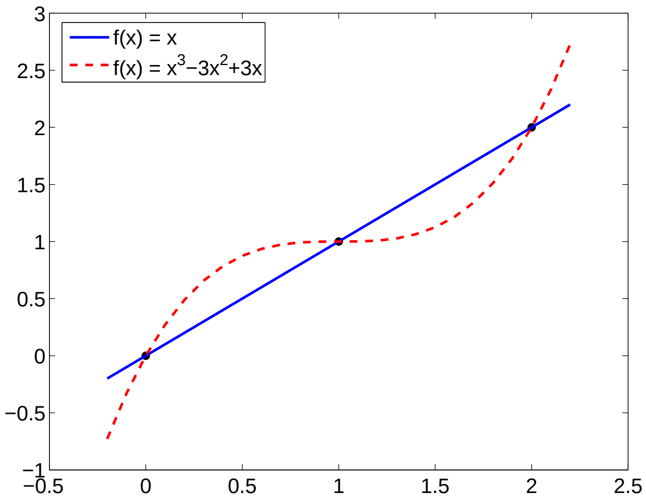


Using MSE only, the cubic model is better than linear!!!

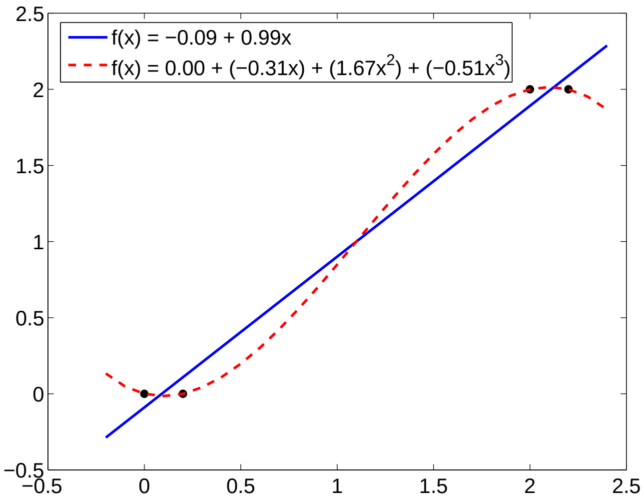
Model evaluation

Fundamental question: What is a good measure of "model quality" from the machine-learning standpoint?

- We have various measures of model error:
  - For regression tasks: MSE, MAE, . . .
  - For classification tasks: misclassification rate, measures based on confusion matrix, . . .
- Some of them can be regarded as finite approximations of the *Bayes risk*.
- Are these functions *good approximations* when measured on the data the models were trained on?



Using MSE only, both models are equivalent!!!



Using MSE only, the cubic model is better than linear!!!

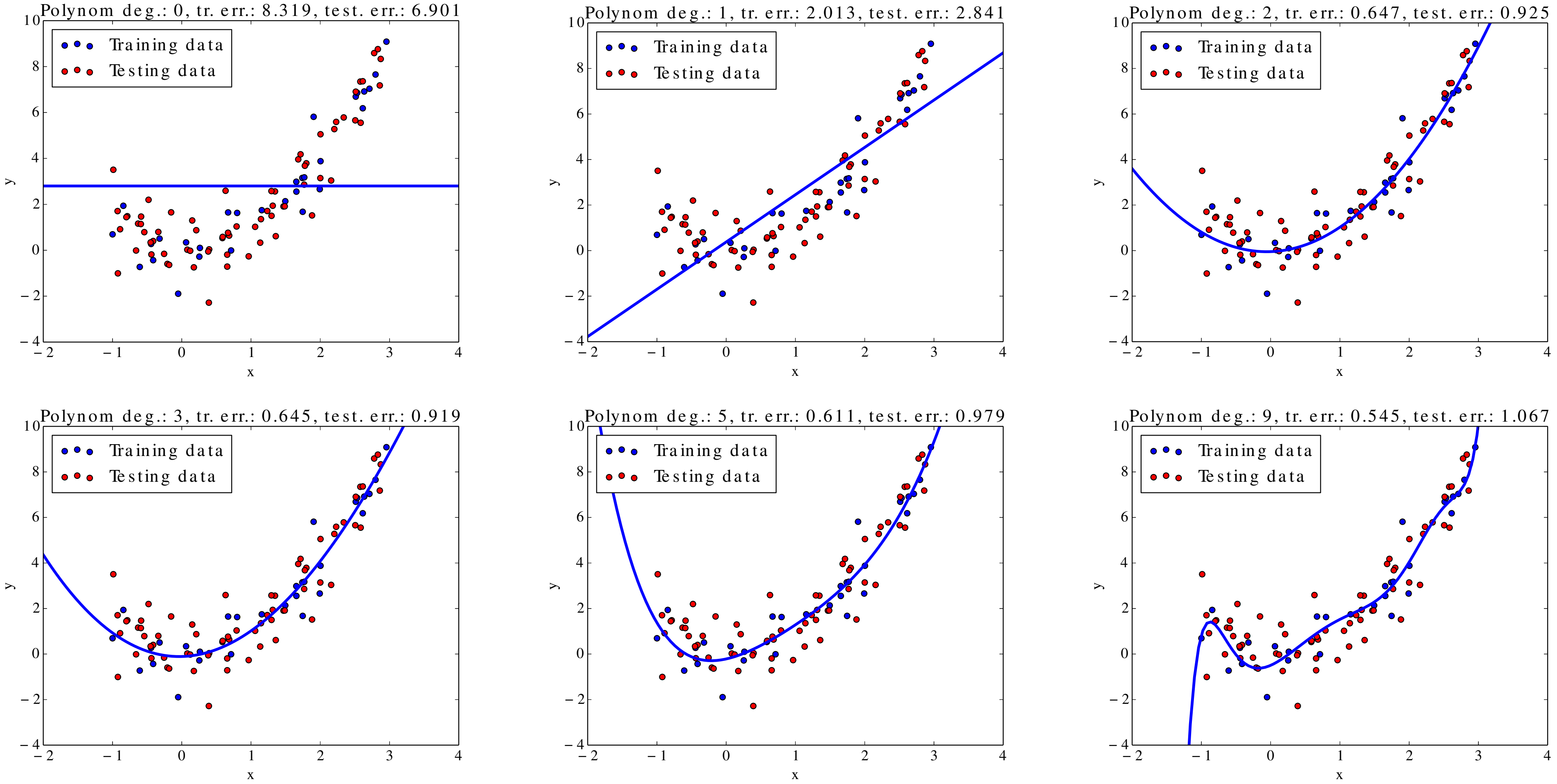
A basic method of evaluation is *model validation on a different, independent data set* from the same source, i.e. on **testing data**.

Validation on testing data

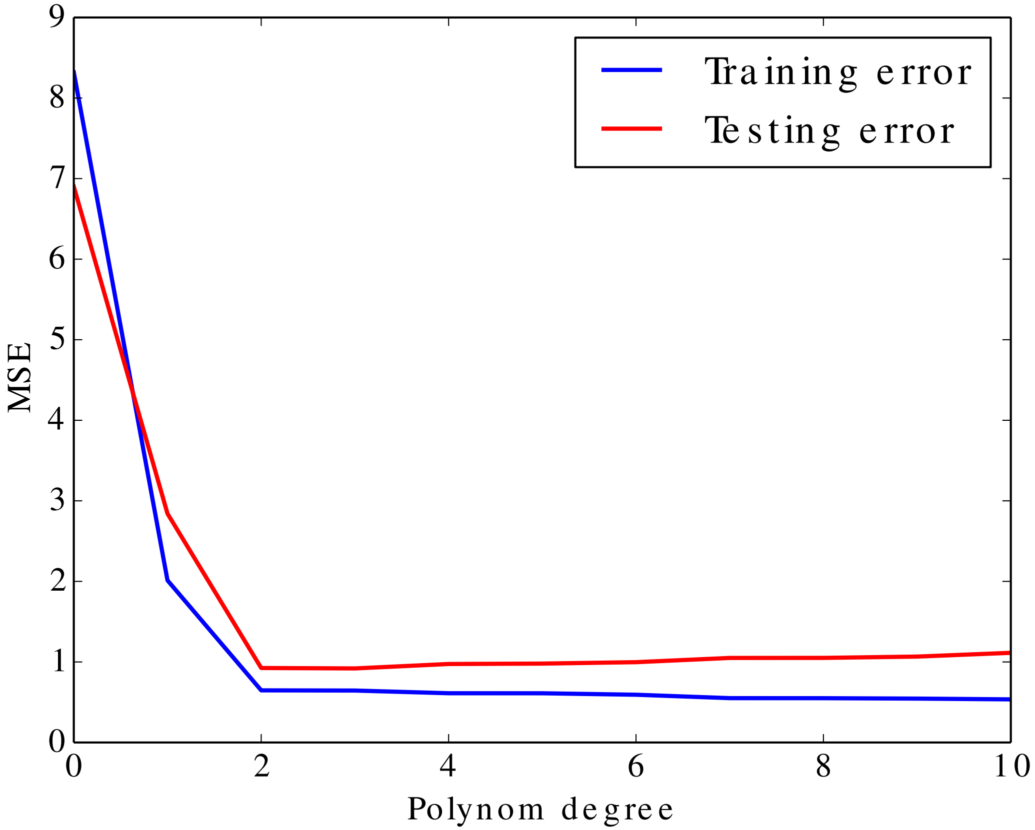
Example: Polynomial regression with varrying degree:

$$X \sim U(-1, 3)$$

$$Y \sim X^2 + N(0, 1)$$



Training and testing error



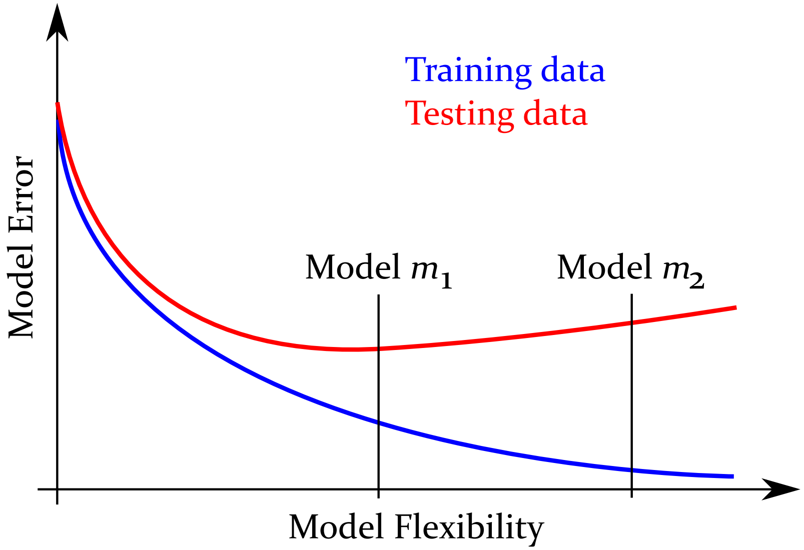
- The *training error* decreases with increasing model flexibility.
- The *testing error* is minimal for certain degree of model flexibility.

Overfitting

Definition of overfitting:

- Let M be the space of candidate models.
- Let $m_1 \in M$ and $m_2 \in M$ be 2 different models from this space.
- Let $\text{Err}_{\text{Tr}}(m)$ be an error of the model m measured on the training dataset (training error).
- Let $\text{Err}_{\text{Tst}}(m)$ be an error of the model m measured on the testing dataset (testing error).
- We say that m_2 is overfitted if there is another m_1 for which

$$\text{Err}_{\text{Tr}}(m_2) < \text{Err}_{\text{Tr}}(m_1) \wedge \text{Err}_{\text{Tst}}(m_2) > \text{Err}_{\text{Tst}}(m_1)$$

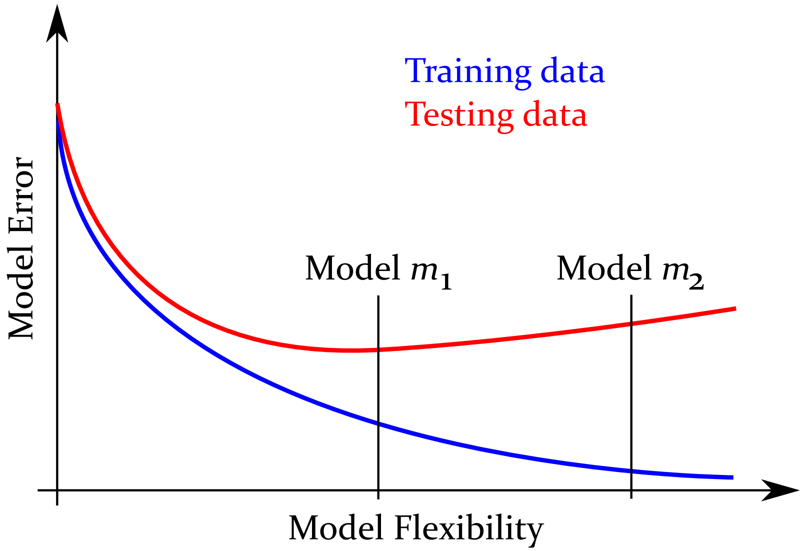


- "When overfitted, the model works well for the training data, but fails for new (testing) data."
- Overfitting is a general phenomenon *affecting all kinds of inductive learning* of models with tunable flexibility.

Overfitting

Definition of overfitting:

- Let M be the space of candidate models.
- Let $m_1 \in M$ and $m_2 \in M$ be 2 different models from this space.
- Let $\mathrm{Err}_{\mathrm{Tr}}(m)$ be an error of the model m measured on the training dataset (training error).
- Let $\mathrm{Err}_{\mathrm{Tst}}(m)$ be an error of the model m measured on the testing dataset (testing error).
- We say that m_2 is overfitted if there is another m_1 for which



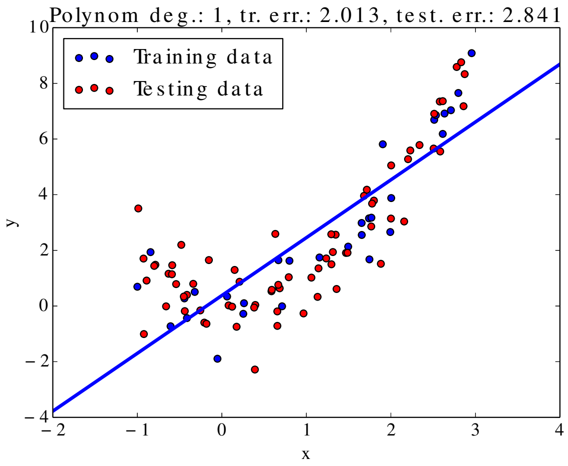
$$\mathrm{Err}_{\mathrm{Tr}}(m_2) < \mathrm{Err}_{\mathrm{Tr}}(m_1) \wedge \mathrm{Err}_{\mathrm{Tst}}(m_2) > \mathrm{Err}_{\mathrm{Tst}}(m_1)$$

- "When overfitted, the model works well for the training data, but fails for new (testing) data."
- Overfitting is a general phenomenon *affecting all kinds of inductive learning* of models with tunable flexibility.

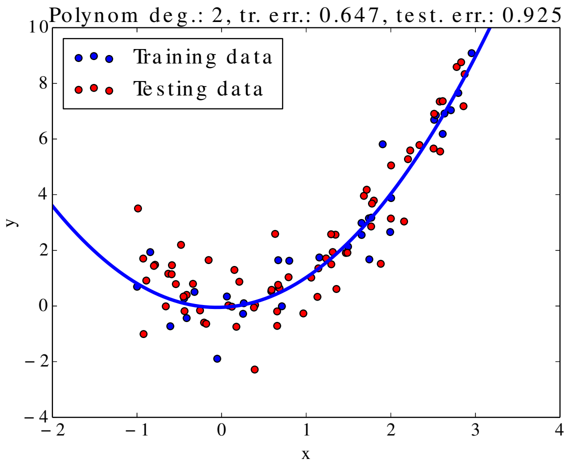
We want models and learning algorithms with a good **generalization ability**, i.e.

- we want models that encode *only the relationships valid in the whole domain*, not those that learned the specifics of the training data, i.e.
- we want algorithms able to find *only the relationships valid in the whole domain* and ignore specifics of the training data.

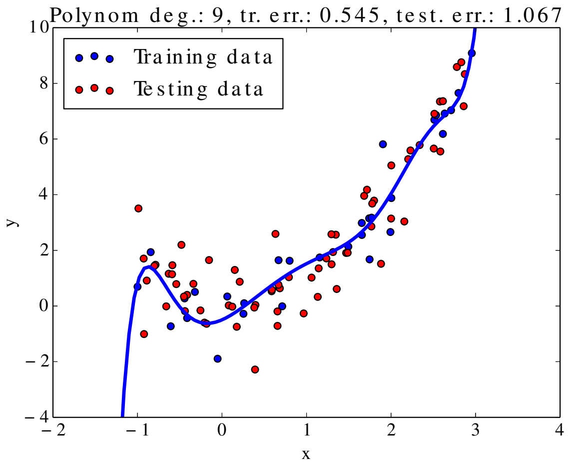
Bias vs Variance


Polynom deg.: 1, tr. err.: 2.013, test. err.: 2.841

High bias:
model not flexible enough
(Underfit)

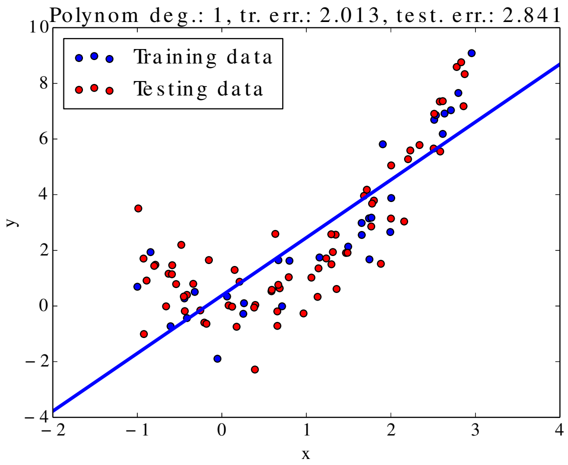

Polynom deg.: 2, tr. err.: 0.647, test. err.: 0.925

"Just right"
(Good fit)

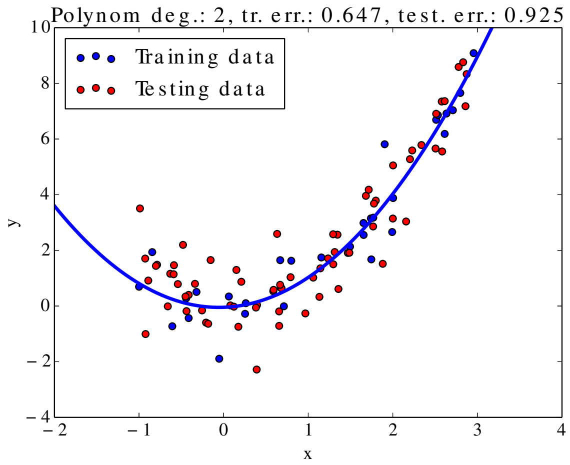

Polynom deg.: 9, tr. err.: 0.545, test. err.: 1.067

High variance:
model flexibility too high
(Overfit)

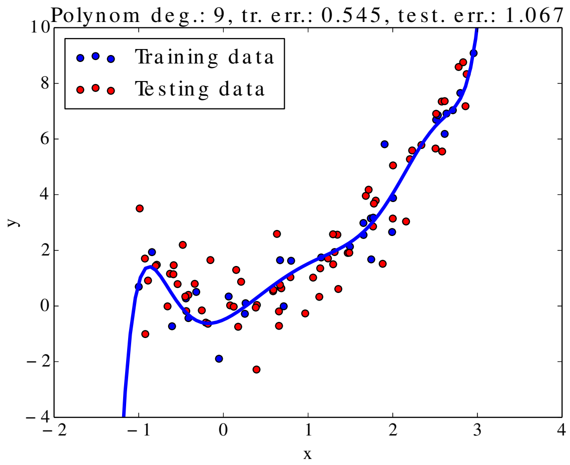
Bias vs Variance



High bias:
model not flexible enough
(Underfit)



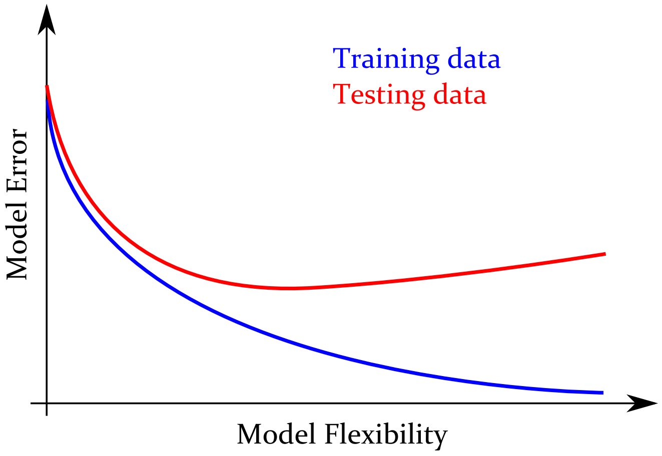
"Just right"
(Good fit)



High variance:
model flexibility too high
(Overfit)

High bias problem:

- $\mathrm{Err}_{\mathrm{Tr}}(h)$ is high
- $\mathrm{Err}_{\mathrm{Tst}}(h) \approx \mathrm{Err}_{\mathrm{Tr}}(h)$



High variance problem:

- $\mathrm{Err}_{\mathrm{Tr}}(h)$ is low
- $\mathrm{Err}_{\mathrm{Tst}}(h) >> \mathrm{Err}_{\mathrm{Tr}}(h)$

Crossvalidation

How to estimate the true error of a model on new, unseen data?

Crossvalidation

How to estimate the true error of a model on new, unseen data?

Simple crossvalidation:

- Split the data into training and testing subsets.
- Train the model on training data.
- Evaluate the model error on testing data.

Crossvalidation

How to estimate the true error of a model on new, unseen data?

Simple crossvalidation:

- Split the data into training and testing subsets.
- Train the model on training data.
- Evaluate the model error on testing data.

K-fold crossvalidation:

- Split the data into k folds (k is usually 5 or 10).
- In each iteration:
  - Use $k - 1$ folds to train the model.
  - Use 1 fold to test the model, i.e. measure error.

Iter. 1	Training	Training	Testing
Iter. 2	Training	Testing	Training
Iter. k	Testing	Training	Training

- Aggregate (average) the k error measurements to get the final error estimate.
- Train the model on the whole data set.

Crossvalidation

How to estimate the true error of a model on new, unseen data?

Simple crossvalidation:

- Split the data into training and testing subsets.
- Train the model on training data.
- Evaluate the model error on testing data.

K-fold crossvalidation:

- Split the data into k folds (k is usually 5 or 10).
- In each iteration:
  - Use $k - 1$ folds to train the model.
  - Use 1 fold to test the model, i.e. measure error.

Iter. 1	Training	Training	Testing
Iter. 2	Training	Testing	Training
Iter. k	Testing	Training	Training

- Aggregate (average) the k error measurements to get the final error estimate.
- Train the model on the whole data set.

Leave-one-out (LOO) crossvalidation:

- $k = |T|$, i.e. the number of folds is equal to the training set size.
- Time consuming for large $|T|$.

How to determine a suitable model flexibility

Simply test models of varying complexities and choose the one with the best testing error, right?

- The testing data are used here to *tune a meta-parameter* of the model.
- *The testing data* are used to train (a part of) the model, thus essentially *become part of training data*.
- The error on testing data is *no longer an unbiased estimate* of model error; it underestimates it.
- A new, separate data set is needed to estimate the model error.

How to determine a suitable model flexibility

Simply test models of varying complexities and choose the one with the best testing error, right?

- The testing data are used here to *tune a meta-parameter* of the model.
- *The testing data* are used to train (a part of) the model, thus essentially *become part of training data*.
- The error on testing data is *no longer an unbiased estimate* of model error; it underestimates it.
- A new, separate data set is needed to estimate the model error.

Using simple crossvalidation:

1. *Training data:* use cca 50 % of data for model building.
2. *Validation data:* use cca 25 % of data to search for the suitable model flexibility.
3. Train the suitable model on training + validation data.
4. *Testing data:* use cca 25 % of data for the final estimate of the model error.

How to determine a suitable model flexibility

Simply test models of varying complexities and choose the one with the best testing error, right?

- The testing data are used here to *tune a meta-parameter* of the model.
- *The testing data* are used to train (a part of) the model, thus essentially *become part of training data*.
- The error on testing data is *no longer an unbiased estimate* of model error; it underestimates it.
- A new, separate data set is needed to estimate the model error.

Using simple crossvalidation:

1. *Training data:* use cca 50 % of data for model building.
2. *Validation data:* use cca 25 % of data to search for the suitable model flexibility.
3. Train the suitable model on training + validation data.
4. *Testing data:* use cca 25 % of data for the final estimate of the model error.

Using k-fold crossvalidation

1. *Training data:* use cca 75 % of data to find and train a suitable model using crossvalidation.
2. *Testing data:* use cca 25 % of data for the final estimate of the model error.

How to determine a suitable model flexibility

Simply test models of varying complexities and choose the one with the best testing error, right?

- The testing data are used here to *tune a meta-parameter* of the model.
- *The testing data* are used to train (a part of) the model, thus essentially *become part of training data*.
- The error on testing data is *no longer an unbiased estimate* of model error; it underestimates it.
- A new, separate data set is needed to estimate the model error.

Using simple crossvalidation:

1. *Training data:* use cca 50 % of data for model building.
2. *Validation data:* use cca 25 % of data to search for the suitable model flexibility.
3. Train the suitable model on training + validation data.
4. *Testing data:* use cca 25 % of data for the final estimate of the model error.

Using k-fold crossvalidation

1. *Training data:* use cca 75 % of data to find and train a suitable model using crossvalidation.
2. *Testing data:* use cca 25 % of data for the final estimate of the model error.

The ratios are not set in stone, there are other possibilities, e.g. 60:20:20, etc.

How to prevent overfitting?

1. **Feature selection:** Reduce the number of features.
   - Select manually, which features to keep.
   - Try to identify a suitable subset of features during learning phase (many feature selection methods exist; none is perfect).
2. **Regularization:**
   - Keep all the features, but reduce the magnitude of their weights w.
   - Works well, if we have a lot of features each of which contributes a bit to predicting y.

Regularization

Ridge regularization (a.k.a. Tikhonov regularization)

Ridge regularization penalizes the size of the model coefficients:

- Modification of the optimization criterion:

$$J(\boldsymbol{w}) = \frac{1}{|T|}\sum_{i=1}^{|T|}\left(y^{(i)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})\right)^2 + \alpha\sum_{d=1}^{D} w_d^2.$$

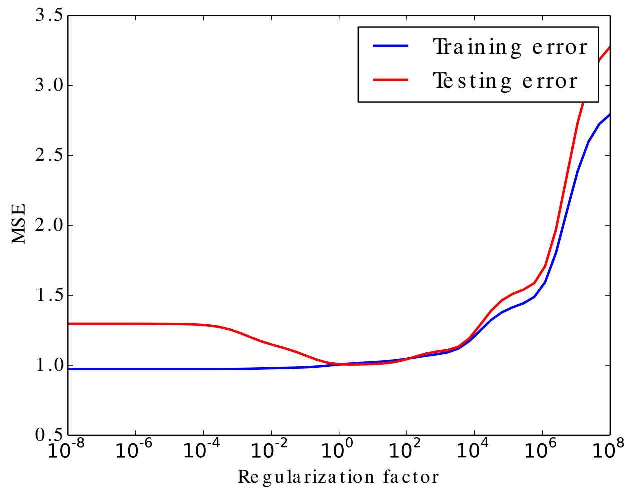
- The solution is given by a modified **Normal equation**

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\mathbf{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

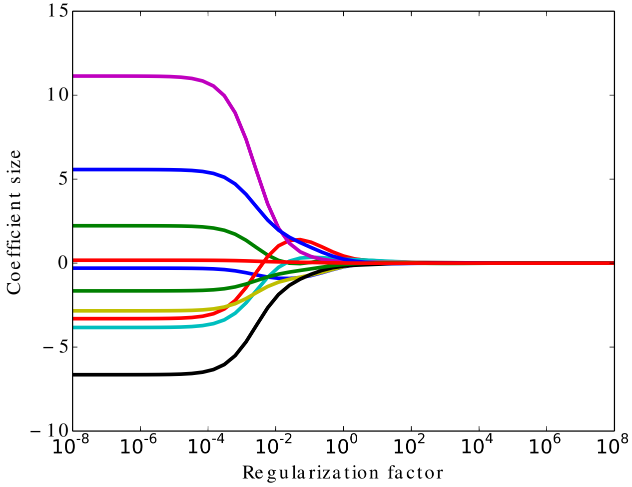
- As $\alpha \to 0$, $\boldsymbol{w}^{\text{ridge}} \to \boldsymbol{w}^{\text{OLS}}$.
- As $\alpha \to \infty$, $\boldsymbol{w}^{\text{ridge}} \to 0$.

OLS - ordinary least squares. Just a simple multiple linear regression.

Training and testing errors as functions of regularization parameter:



The values of coefficients (weights $\boldsymbol{w}$) as functions of regularization parameter:



Lasso regularization

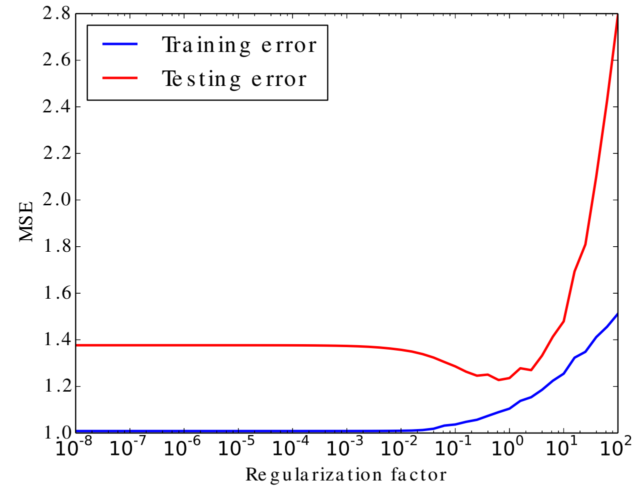
Lasso regularization penalizes the size of the model coefficients:

- Modification of the optimization criterion:

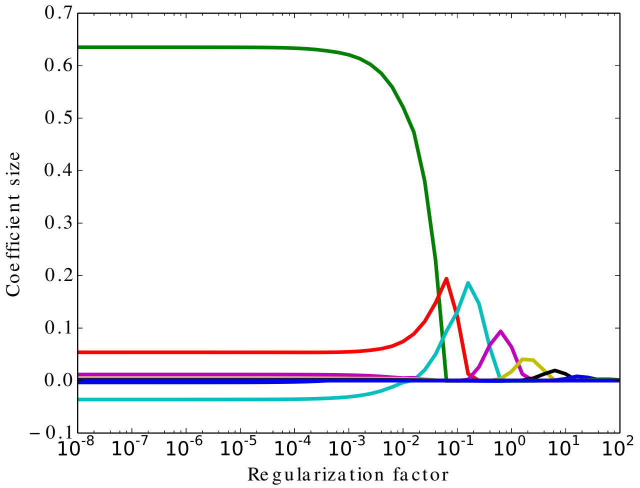
$$J(\boldsymbol{w}) = \frac{1}{|T|}\sum_{i=1}^{|T|}\left(y^{(i)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})\right)^2 + \alpha\sum_{d=1}^{D}|w_d|.$$

- As $\alpha \to \infty$, Lasso regularization *decreases the number of non-zero coefficients*, effectively also performing *feature selection* and creating *sparse models*.

Training and testing errors as functions of regularization parameter:



The values of coefficients as functions of regularization parameter:



Summary

Competencies

After this lecture, a student shall be able to . . .

- explain the reason for doing basis expansion (feature space straightening), and describe its principle;
- show the effect of basis expansion with a linear model on a simple example for both classification and regression settings;
- implement user-defined basis expansions in certain programming language;
- list advantages and disadvantages of basis expansion;
- explain why the error measured on the training data is not a good estimate of the expected error of the model for new data, and whether it under- or overestimates the true error;
- explain basic methods to get unbiased estimate of the true model error (testing data, k-fold crossvalidation, LOO crossvalidation);
- describe the general form of the dependency of training and testing errors on the model complexity/flexibility/capacity;
- define overfitting;
- discuss high bias and high variance problems of models;
- explain how to proceed if a suitable model complexity must be chosen as part of the training process;
- list 2 basic methods for overfitting prevention;
- describe the principles of ridge (Tikhonov) and lasso regularizations and their effects on the model parameters.