Lecture 3: Under-fitting, Over-fitting, Regularization, and Model Selection

Tao LIN

March 2, 2023

WESTLAKE UNIVERSITY | SCHOOL OF ENGINEERING

Reading materials

- Chapter 3.1 & 3.2, Bishop, Pattern Recognition and Machine Learning
- Read about over-fitting in the paper by Pedro Domingos (Sections 3 and 5 of “A few useful things to know about machine learning”)

Reference

- EPFL, CS-433 Machine Learning, `https://github.com/epfml/ML_course`

Table of Contents

Table of Contents

Definition 1 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

$$\min_{\mathbf{w}} \; \mathcal{L}(\mathbf{w}) \quad \text{subject to } \mathbf{w} \in \mathbb{R}^D \tag{1}$$

We will use an optimization algorithm (e.g., Gradient Descent) to find a good $\mathbf{w}$.

Considering a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and learnable weights $\mathbf{w} \in \mathbb{R}^D$ for $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N\times D} \tag{2}$$

Using Gradient Descent for Linear Regression with MSE

The MSE is defined as:

$$\mathcal{L}(\mathbf{w}) := \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \tfrac{1}{2N} \mathbf{e}^\top \mathbf{e} \,. \tag{3}$$

The error vector $\mathbf{e}$ is defined as:

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} = \begin{pmatrix} e_1 \\ \vdots \\ e_N \end{pmatrix} \in \mathbb{R}^N \,, \tag{4}$$

where $e_i := y_n - \mathbf{x}_n^\top \mathbf{w}$.

The gradient is given by

$$\nabla \mathcal{L}(\mathbf{w}) = -\tfrac{1}{N} \mathbf{X}^\top \mathbf{e} \tag{5}$$

A probabilistic model for linear regression

Definition 2 (Data generation process)

We assume that the data is generated by the model,

$$y_n = \mathbf{x}_n^\top \mathbf{w} + \epsilon_n, \tag{6}$$

where

- the ϵ_n (the noise) is a zero-mean Gaussian random variable with variance σ^2
- the noise is independent of each other and independent of the input.
- the model $\mathbf{w}$ is unknown.

The likelihood of the data vector $\mathbf{y} = (y_1, \cdots, y_N)$ given the input $\mathbf{X}$ and the model $\mathbf{w}$ is

$$p(\mathbf{y} \,|\, \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n \,|\, \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(y_n \,|\, \mathbf{x}_n^\top \mathbf{w}, \sigma^2). \tag{7}$$

The probabilistic view point: maximize this likelihood over the choice of model $\mathbf{w}$.

Maximum-Likelihood Estimator (MLE)

Instead of maximizing the likelihood, we can maximize the logarithm of the likelihood, i.e., log-likelihood (LL):

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) := \log\, p(\mathbf{y} \,|\, \mathbf{X}, \mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst.} \tag{8}$$

Compare the LL to the MSE (Mean Squared Error)

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst} \tag{9}$$

$$\mathcal{L}_{\text{MSE}}(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 \tag{10}$$

Maximizing the LL is equivalent to minimizing the MSE:

$$\arg\min_{\mathbf{w}}\, \mathcal{L}_{\text{MSE}}(\mathbf{w}) = \arg\max_{\mathbf{w}}\, \mathcal{L}_{\text{LL}}(\mathbf{w}). \tag{11}$$

Table of Contents

Motivation

- In rare cases, one can compute the optimum of the cost function analytically.
- Linear regression using an MSE cost function is one such case.
- Here its solution can be obtained explicitly, by solving a linear system of equations.

$\Rightarrow$ These equations are sometimes called the normal equations.

$\Rightarrow$ Solving the normal equations is called the least squares.

Normal Equations

To derive the normal equations,

1. we first show that the problem is convex.

2. we then use the optimality conditions for convex functions, i.e.,

$$\nabla \mathcal{L}(\mathbf{w}^\star) = \mathbf{0}\,, \tag{12}$$

   where $\mathbf{w}^\star$ corresponds to the parameter at the optimum point.

   Given the definition $\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$, we have

$$\nabla \mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0\,, \tag{13}$$

   where we can get the normal equations for linear regression.

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{14}$$

If the Gram matrix is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,, \tag{15}$$

where we can get a closed-form expression for the minimum.

Last lecture:

- Basic concept of regression and classification
- Linear regression
  - Definition
  - Gradient Descent (GD) optimization
  - Least Square
  - The probabilistic interpretation of linear regression

Last lecture:

- Basic concept of regression and classification
- Linear regression
  - Definition
  - Gradient Descent (GD) optimization
  - Least Square
  - The probabilistic interpretation of linear regression

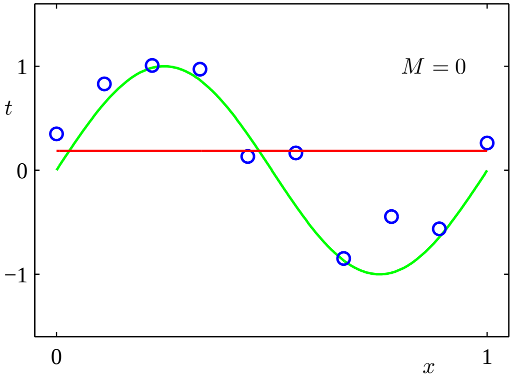
This lecture:

- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso Regression
- Generalization, and Model selection

Table of Contents

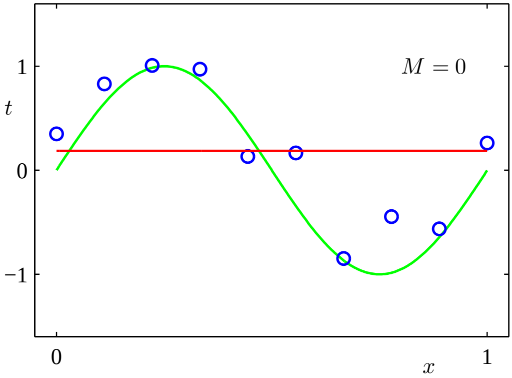
Table of Contents

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

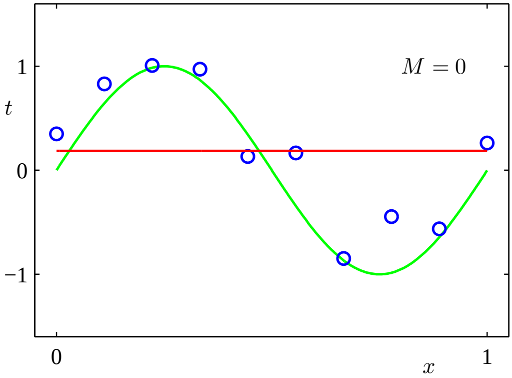
Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

- A scalar function $g(x)$

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$

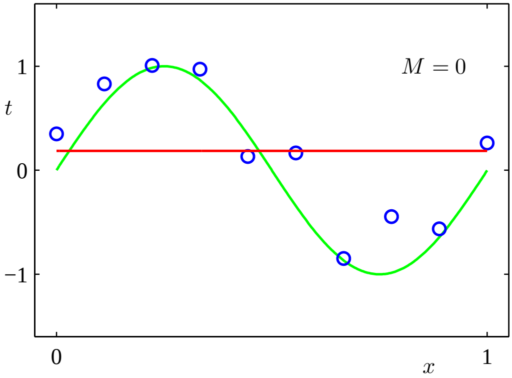


Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \,. \tag{16}$$

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



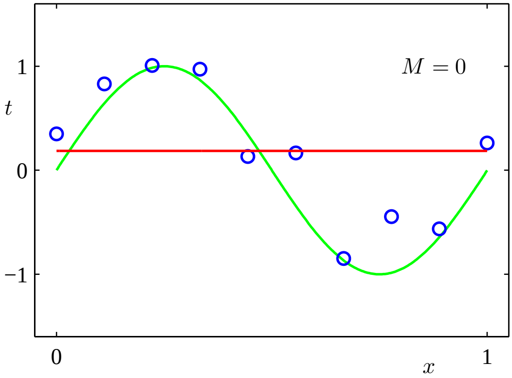
Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \,. \tag{16}$$

- The noise Z_n might be due to some measurement inaccuracies.

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



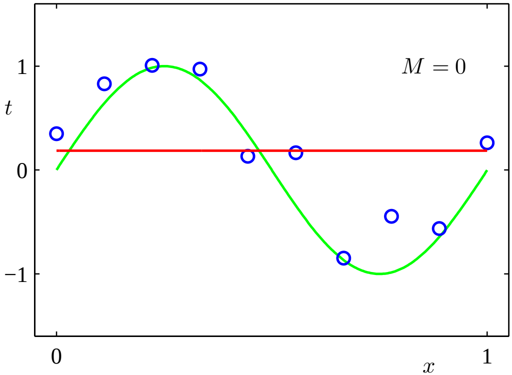
Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \,. \tag{16}$$

- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

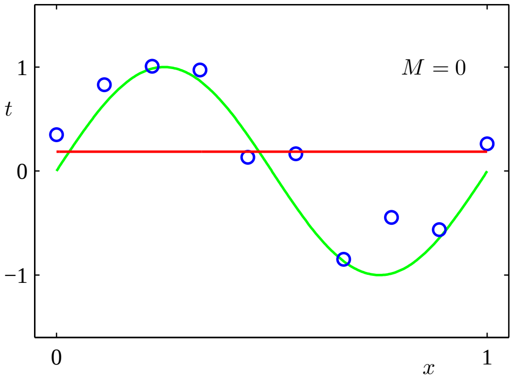
- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \, . \tag{16}$$

- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.
- Our model family consists of only linear functions of the scalar input x:

$$\mathcal{F} = \{f_w(x) = xw\} \, . \tag{17}$$

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

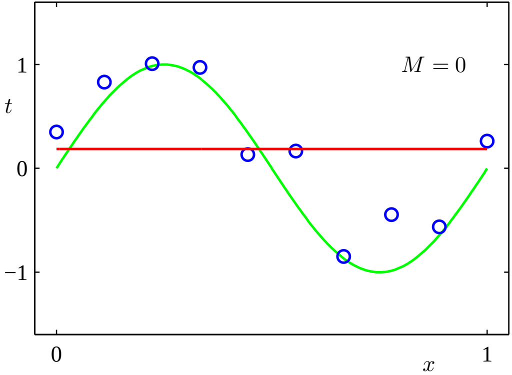
$$y_n = g(x_n) + Z_n \,. \tag{16}$$

- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.
- Our model family consists of only linear functions of the scalar input x:

$$\mathcal{F} = \{f_w(x) = xw\} \,. \tag{17}$$

Observations:

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \,. \tag{16}$$

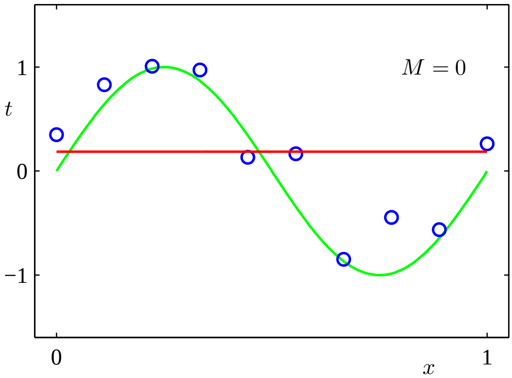
- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.
- Our model family consists of only linear functions of the scalar input x:

$$\mathcal{F} = \{f_w(x) = xw\} \,. \tag{17}$$

Observations:

- The solid curve is the underlying function (the slope of the function).

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \,. \tag{16}$$

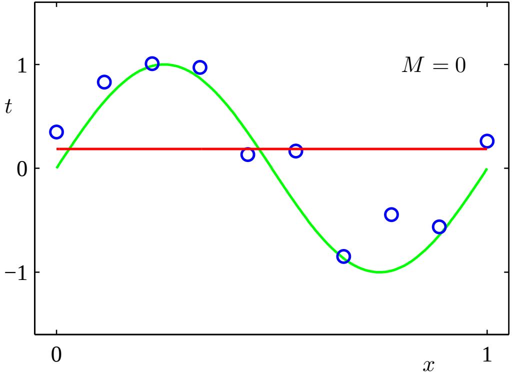
- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.
- Our model family consists of only linear functions of the scalar input x:

$$\mathcal{F} = \{f_w(x) = xw\} \,. \tag{17}$$

Observations:

- The solid curve is the underlying function (the slope of the function).
- We cannot match the given function accurately,

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

$$y_n = g(x_n) + Z_n \, . \tag{16}$$

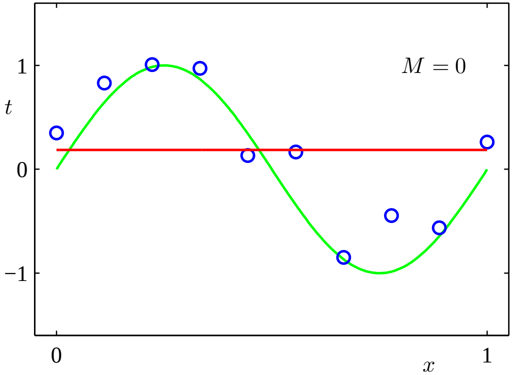
- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.
- Our model family consists of only linear functions of the scalar input x:

$$\mathcal{F} = \{f_w(x) = xw\} \, . \tag{17}$$

Observations:

- The solid curve is the underlying function (the slope of the function).
- We cannot match the given function accurately,
  regardless of how many samples we get and how small the noise is.

Under-fitting with Linear Models $f_{\mathbf{w}}(\mathbf{X}) := \mathbf{X}\mathbf{w}$



Settings:

- A scalar function $g(x)$
- We do not observe $g(x_n)$ directly:

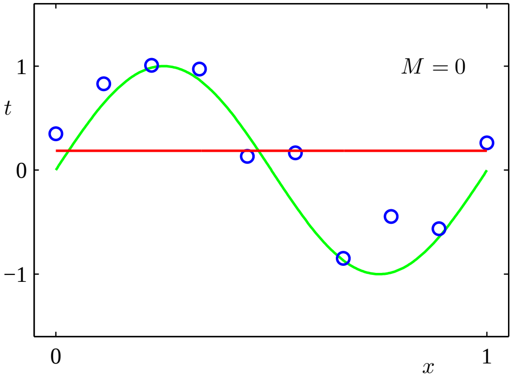
$$y_n = g(x_n) + Z_n \,. \tag{16}$$

- The noise Z_n might be due to some measurement inaccuracies.
- The y_n are shown as blue circles.
- Our model family consists of only linear functions of the scalar input x:

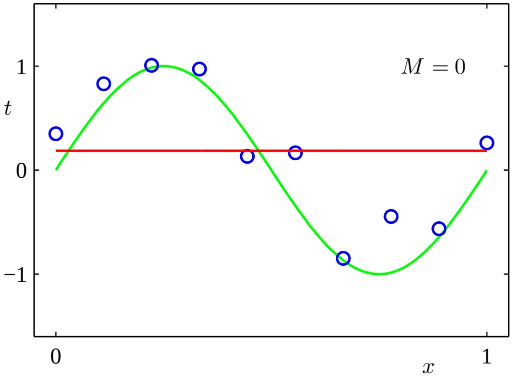
$$\mathcal{F} = \{f_w(x) = xw\} \,. \tag{17}$$

Linear Model might under-fit!

Table of Contents



Linear Model might under-fit!



~~Linear Model might under-fit!~~

Extended/Augmented feature vectors

We can increase the representational power of linear models by “augmenting” the input!

Extended/Augmented feature vectors

We can increase the representational power of linear models by “augmenting” the input!

Using a one-dimensional input (feature) x_n as an example:

Extended/Augmented feature vectors

We can increase the representational power of linear models by "augmenting" the input!

Using a one-dimensional input (feature) x_n as an example:

- Instead of only using the input feature x_n,

Extended/Augmented feature vectors

We can increase the representational power of linear models by "augmenting" the input!

Using a one-dimensional input (feature) x_n as an example:

- Instead of only using the input feature x_n,
- we might add a polynomial basis to get an extended feature vector $\phi(x_n)$, i.e.,

$$\phi(x_n) := \left[1,\, x_n,\, x_n^2,\, x_n^3,\, \ldots,\, x_n^M\right] \;, \tag{18}$$

Extended/Augmented feature vectors

We can increase the representational power of linear models by "augmenting" the input!
Using a one-dimensional input (feature) x_n as an example:

- Instead of only using the input feature x_n,
- we might add a polynomial basis to get an extended feature vector $\phi(x_n)$, i.e.,

$$\phi(x_n) := \left[1,\, x_n,\, x_n^2,\, x_n^3,\, \ldots,\, x_n^M\right]\ , \tag{18}$$

  where M is of arbitrary degree.

Extended/Augmented feature vectors

We can increase the representational power of linear models by "augmenting" the input!
Using a one-dimensional input (feature) x_n as an example:

- Instead of only using the input feature x_n,
- we might add a polynomial basis to get an extended feature vector $\phi(x_n)$, i.e.,

$$\phi(x_n) := \left[1,\, x_n,\, x_n^2,\, x_n^3,\, \ldots,\, x_n^M\right] \;, \tag{18}$$

  where M is of arbitrary degree.
- We then fit a linear model to this extended feature vector $\phi(x_n)$:

Extended/Augmented feature vectors

We can increase the representational power of linear models by "augmenting" the input!
Using a one-dimensional input (feature) x_n as an example:

- Instead of only using the input feature x_n,
- we might add a polynomial basis to get an extended feature vector $\phi(x_n)$, i.e.,

$$\phi(x_n) := \left[1,\, x_n,\, x_n^2,\, x_n^3,\, \ldots,\, x_n^M\right] \,, \tag{18}$$

  where M is of arbitrary degree.
- We then fit a linear model to this extended feature vector $\phi(x_n)$:

$$\begin{aligned} y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M && (19) \\ &=: \phi(x_n)^\top \mathbf{w}\,. && (20) \end{aligned}$$

Extended/Augmented feature vectors

We can increase the representational power of linear models by “augmenting” the input!
Using a one-dimensional input (feature) x_n as an example:

- Instead of only using the input feature x_n,
- we might add a polynomial basis to get an extended feature vector $\phi(x_n)$, i.e.,

$$\phi(x_n) := \left[1,\, x_n,\, x_n^2,\, x_n^3,\, \ldots,\, x_n^M\right]\ , \tag{18}$$

  where M is of arbitrary degree.
- We then fit a linear model to this extended feature vector $\phi(x_n)$:

$$\begin{aligned} y_n &\approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M && (19)\\ &=: \phi(x_n)^\top \mathbf{w}\,. && (20) \end{aligned}$$

Is it all good?

Table of Contents

Interpolation between under-fitting and over-fitting

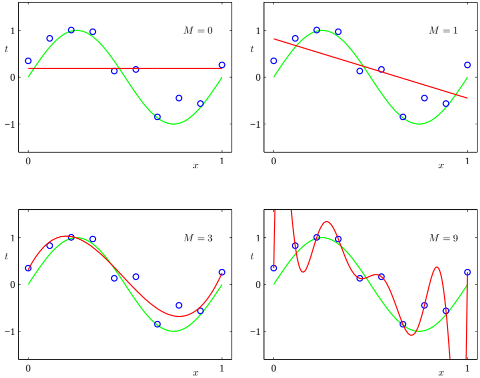
Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \boldsymbol{\phi}(x_n)^\top \mathbf{w}\,. \tag{21}$$

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,. \tag{21}$$

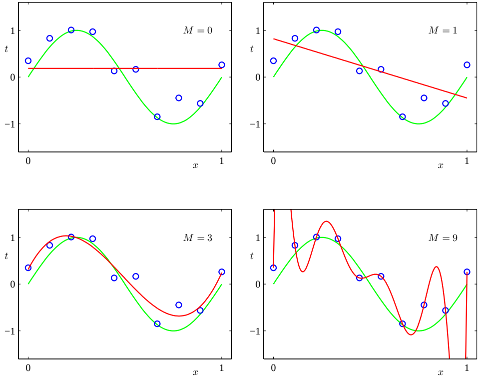


Settings:

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{21}$$



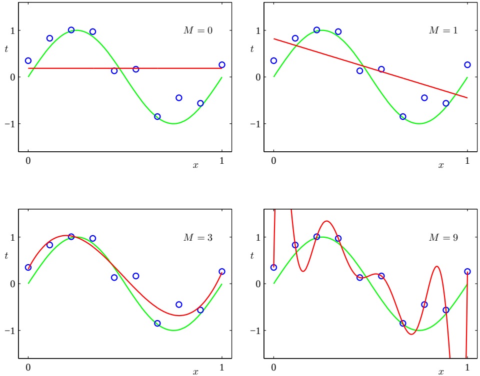
Settings:

- The circles are data points

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{21}$$



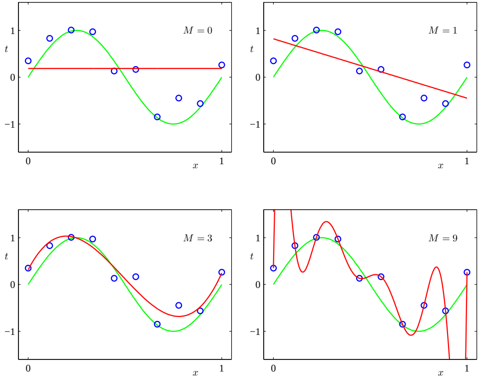
Settings:

- The circles are data points
- The green line represents the "true function"

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,. \tag{21}$$



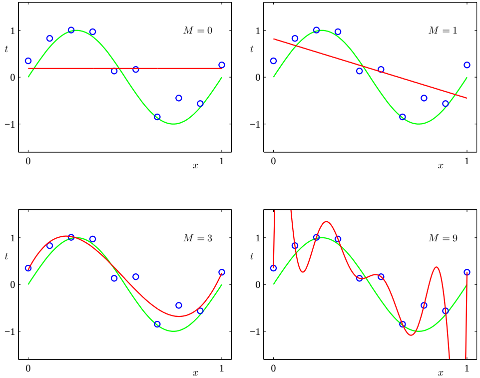
Settings:

- The circles are data points
- The green line represents the "true function"
- The red line is the learned model.

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,. \tag{21}$$



Settings:

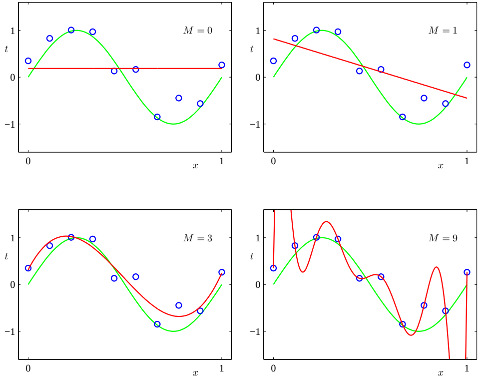
- The circles are data points
- The green line represents the "true function"
- The red line is the learned model.

Observations:

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{21}$$



Settings:

- The circles are data points
- The green line represents the "true function"
- The red line is the learned model.

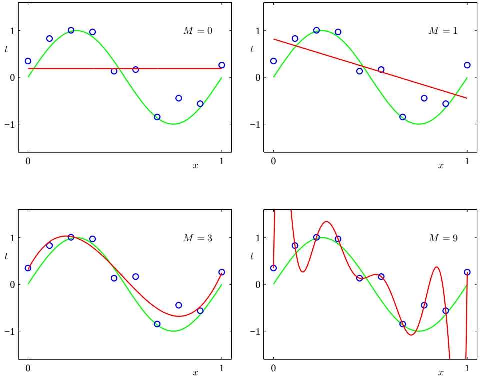
Observations:

- The model is under-fitting for $M = 0$ & $M = 1$

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{21}$$



Settings:

- The circles are data points
- The green line represents the "true function"
- The red line is the learned model.

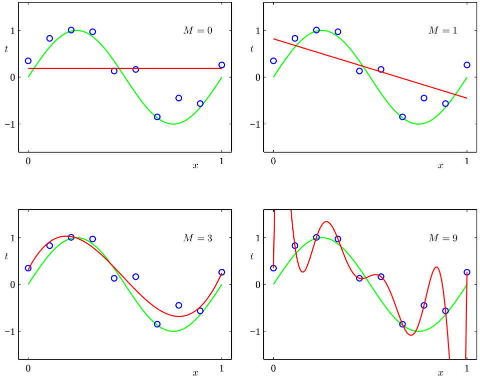
Observations:

- The model is under-fitting for $M = 0$ & $M = 1$
- For $M = 3$, the model fits the data

Interpolation between under-fitting and over-fitting

Let’s consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,. \tag{21}$$



Settings:

- The circles are data points
- The green line represents the “true function”
- The red line is the learned model.

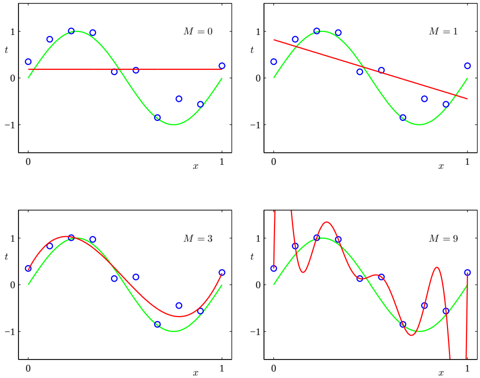
Observations:

- The model is under-fitting for $M = 0$ & $M = 1$
- For $M = 3$, the model fits the data
  $\rightarrow$ but can be improved

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{21}$$



Settings:

- The circles are data points
- The green line represents the "true function"
- The red line is the learned model.

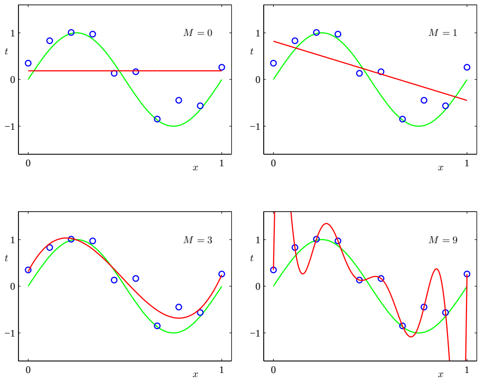
Observations:

- The model is under-fitting for $M = 0$ & $M = 1$
- For $M = 3$, the model fits the data
  $\rightarrow$ but can be improved
- For $M = 9$, the model fits every single points

Interpolation between under-fitting and over-fitting

Let's consider the polynomial regression problem (for a one-dimensional input feature x_n):

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w} \,. \tag{21}$$



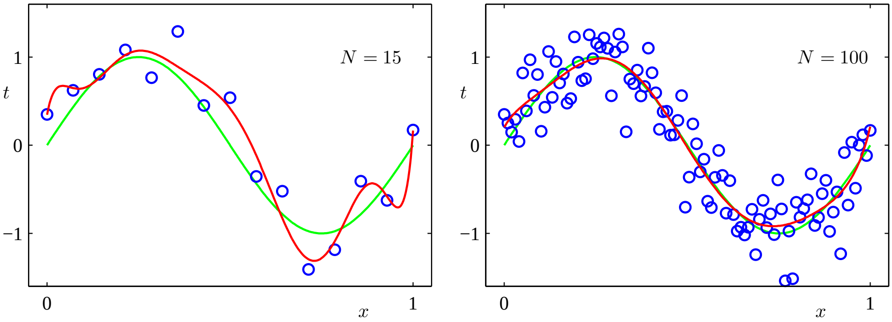
Settings:

- The circles are data points
- The green line represents the "true function"
- The red line is the learned model.

Observations:

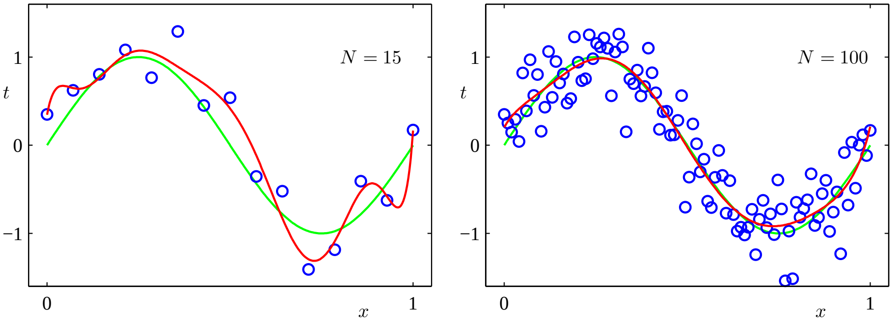
- The model is under-fitting for $M = 0$ & $M = 1$
- For $M = 3$, the model fits the data
  $\rightarrow$ but can be improved
- For $M = 9$, the model fits every single points
  $\rightarrow$ severe over-fitting

- **Question:** How to avoid over-fitting?



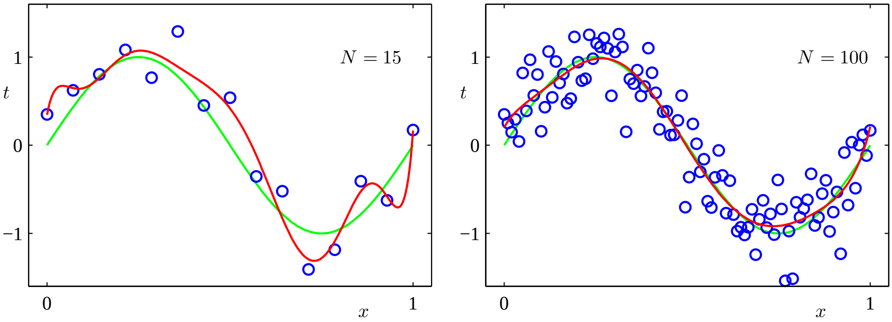
Increasing N but keeping M fixed.

- **Question:** How to avoid over-fitting?
- **Potential solution:** Increasing the amount of data might reduce over-fitting.



Increasing N but keeping M fixed.

- **Question:** How to avoid over-fitting?
- **Potential solution:** Increasing the amount of data might reduce over-fitting.



Increasing N but keeping M fixed.

We will elaborate on this question in the next section!

Definitions of under-fit and over-fit

Definitions of under-fit and over-fit

- under-fit: cannot find the *the underlying function* of the data.

Definitions of under-fit and over-fit

- under-fit: cannot find the the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Definitions of under-fit and over-fit

- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.

Definitions of under-fit and over-fit

- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:

Definitions of under-fit and over-fit

- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:
  - since all we have is data

Definitions of under-fit and over-fit

- under-fit: cannot find the *the underlying function* of the data.
- over-fit: can fit both *the underlying function* and *the noise in the data*.

Discussion:

- Both of these phenomena (under-fit and over-fit) are undesirable.
- This discussion is made more difficult:
  - since all we have is data
  - we do not know a priori what part is the underlying signal and what part is noise.

Table of Contents

We have seen that by augmenting the feature vector:

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,, \tag{22}$$

we can increase the representation power (or complexity) of the linear model.

We have seen that by augmenting the feature vector:

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \boldsymbol{\phi}(x_n)^\top \mathbf{w}\,, \tag{22}$$

we can increase the representation power (or complexity) of the linear model.

The issue of over-fitting!

We have seen that by augmenting the feature vector:

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,, \tag{22}$$

we can increase the representation power (or complexity) of the linear model.

The issue of over-fitting!

Regularization is a way to mitigate this undesirable behavior.

We have seen that by augmenting the feature vector:

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,, \tag{22}$$

we can increase the representation power (or complexity) of the linear model.

The issue of over-fitting!

Regularization is a way to mitigate this undesirable behavior.

- We will discuss regularization in the context of linear models

We have seen that by augmenting the feature vector:

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \phi(x_n)^\top \mathbf{w}\,, \tag{22}$$

we can increase the representation power (or complexity) of the linear model.

The issue of over-fitting!

Regularization is a way to mitigate this undesirable behavior.

- We will discuss regularization in the context of linear models
- The same principle applies also to more complex models such as neural nets.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \boxed{\Omega(\mathbf{w})} \tag{23}$$

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \boxed{\Omega(\mathbf{w})} \tag{23}$$

- Ω is a regularizer.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \boxed{\Omega(\mathbf{w})} \tag{23}$$

- Ω is a regularizer.
- Ω measures the complexity of the model given by $\mathbf{w}$.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \boxed{\Omega(\mathbf{w})} \tag{23}$$

- Ω is a regularizer.
- Ω measures the complexity of the model given by $\mathbf{w}$.
- Model Complexity $\iff$ The richness of the model space.

Table of Contents

L_2-regularization: Ridge Regression

The most frequently used regularizer is the standard Euclidean norm (L_2-norm), which is

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 \qquad \text{where } \|\mathbf{w}\|_2^2 = \sum_i w_i^2 \ . \tag{24}$$

L_2-regularization: Ridge Regression

The most frequently used regularizer is the standard Euclidean norm (L_2-norm), which is

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 \qquad \text{where } \|\mathbf{w}\|_2^2 = \sum_i w_i^2 \ . \tag{24}$$

- The main effect: large model weights w_i will be penalized (avoided).

L_2-regularization: Ridge Regression

The most frequently used regularizer is the standard Euclidean norm (L_2-norm), which is

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 \qquad \text{where } \|\mathbf{w}\|_2^2 = \sum_i w_i^2 \ . \tag{24}$$

- The main effect: large model weights w_i will be penalized (avoided).
- It encourages weight values to decay toward zero unless supported by the data.

L_2-regularization: Ridge Regression

The most frequently used regularizer is the standard Euclidean norm (L_2-norm), which is

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 \qquad \text{where } \|\mathbf{w}\|_2^2 = \sum_i w_i^2 \ . \tag{24}$$

- The main effect: large model weights w_i will be penalized (avoided).
- It encourages weight values to decay toward zero unless supported by the data.
- When $\mathcal{L}$ is MSE, we can define the ridge regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \left[y_n - \mathbf{x}_n^\top \mathbf{w}\right]^2 + \lambda \|\mathbf{w}\|_2^2 \tag{25}$$

L_2-regularization: Ridge Regression

The most frequently used regularizer is the standard Euclidean norm (L_2-norm), which is

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 \qquad \text{where } \|\mathbf{w}\|_2^2 = \sum_i w_i^2 \ . \tag{24}$$

- The main effect: large model weights w_i will be penalized (avoided).
- It encourages weight values to decay toward zero unless supported by the data.
- When $\mathcal{L}$ is MSE, we can define the ridge regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \left[y_n - \mathbf{x}_n^\top \mathbf{w}\right]^2 \ + \ \lambda \|\mathbf{w}\|_2^2 \tag{25}$$

- *Linear Regression* is a special case of this: by setting $\lambda := 0$.

Explicit solution of Ridge Regression for $\mathbf{w}$

Ridge Regression can be defined as:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \ + \ \lambda \|\mathbf{w}\|_2^2 \ . \tag{26}$$

Explicit solution of Ridge Regression for $\mathbf{w}$

Ridge Regression can be defined as:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; . \tag{26}$$

Differentiating and setting to zero (following a similar procedure as least squares):

$$\mathbf{w}^{\star}_{\text{ridge}} = (\mathbf{X}^{\top}\mathbf{X} + \lambda'\mathbf{I})^{-1}\mathbf{X}^{\top}\mathbf{y} \tag{27}$$

(here for simpler notation $\lambda'/2N = \lambda$)

Ridge Regression to fight ill-conditioning

Ridge Regression to fight ill-conditioning

Lemma 3 (Lifting the igenvalues)

The eigenvalues of $(\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I})$ *are all at least* λ' *and so the inverse always exists.*

Ridge Regression to fight ill-conditioning

Lemma 3 (Lifting the igenvalues)

The eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})$ *are all at least* λ' *and so the inverse always exists.*

Proof.

Write the Eigenvalue decomposition of $\mathbf{X}^\top\mathbf{X}$ as $\mathbf{USU}^\top$.

$\square$

Ridge Regression to fight ill-conditioning

Lemma 3 (Lifting the igenvalues)

The eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})$ *are all at least* λ' *and so the inverse always exists.*

Proof.

Write the Eigenvalue decomposition of $\mathbf{X}^\top\mathbf{X}$ as $\mathbf{USU}^\top$. We then have

$$\begin{aligned}\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I} &= \mathbf{USU}^\top + \lambda'\mathbf{UIU}^\top && (28)\\ &= \mathbf{U}[\mathbf{S} + \lambda'\mathbf{I}]\mathbf{U}^\top. && (29)\end{aligned}$$

□

Ridge Regression to fight ill-conditioning

Lemma 3 (Lifting the igenvalues)

The eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})$ *are all at least* λ' *and so the inverse always exists.*

Proof.

Write the Eigenvalue decomposition of $\mathbf{X}^\top\mathbf{X}$ as $\mathbf{USU}^\top$. We then have

$$\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I} = \mathbf{USU}^\top + \lambda'\mathbf{UIU}^\top \tag{28}$$

$$= \mathbf{U}[\mathbf{S} + \lambda'\mathbf{I}]\mathbf{U}^\top. \tag{29}$$

We see now that every Eigenvalue is “lifted” by an amount λ'. $\square$

An alternative proof (optional reading).

Recall that for a symmetric matrix $\mathbf{A}$ we can also compute eigenvalues by looking at the so-called Rayleigh ratio,

$$R(\mathbf{A}, \mathbf{v}) = \frac{\mathbf{v}^\top \mathbf{A}\mathbf{v}}{\mathbf{v}^\top \mathbf{v}}. \tag{30}$$

Note that if $\mathbf{v}$ is an eigenvector with eigenvalue λ, then the Rayleigh coefficient indeed gives us λ.

We can find the smallest and largest eigenvalue by minimizing and maximizing this coefficient.

But note that if we apply this to the symmetric matrix $\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I}$, then for any vector $\mathbf{v}$ we have

$$\frac{\mathbf{v}^\top (\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I})\mathbf{v}}{\mathbf{v}^\top \mathbf{v}} \geq \frac{\lambda' \mathbf{v}^\top \mathbf{v}}{\mathbf{v}^\top \mathbf{v}} = \lambda' . \tag{31}$$

$\square$

Table of Contents

L_1-regularization: the Lasso

Previously, we regularize the complexity of the model:

- e.g., L_2-regularization: Ridge Regression

L_1-regularization: the Lasso

Previously, we regularize the complexity of the model:

- e.g., L_2-regularization: Ridge Regression
- What if we consider an alternative complexity measure?

L_1-regularization: the Lasso

Previously, we regularize the complexity of the model:

- e.g., L_2-regularization: Ridge Regression
- What if we consider an alternative complexity measure?
- Just by considering a different norm!

L_1-regularization: the Lasso

Previously, we regularize the complexity of the model:

- e.g., L_2-regularization: Ridge Regression
- What if we consider an alternative complexity measure?
- Just by considering a different norm!
- A very important case is the L_1-norm, leading to L_1-regularization.

L_1-regularization: the Lasso

Previously, we regularize the complexity of the model:

- e.g., L_2-regularization: Ridge Regression
- What if we consider an alternative complexity measure?
- Just by considering a different norm!
- A very important case is the L_1-norm, leading to L_1-regularization.

In combination with the MSE cost function (i.e., L_2-norm), this is known as the Lasso:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \left[y_n - \mathbf{x}_n^\top \mathbf{w} \right]^2 + \lambda \left\| \mathbf{w} \right\|_1 \qquad \text{where } \left\| \mathbf{w} \right\|_1 := \sum_i |w_i| \,. \tag{32}$$

Table of Contents

Geometric interpretation for the Ridge Regression

Recall the definition of the Ridge Regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; . \tag{33}$$

Geometric interpretation for the Ridge Regression

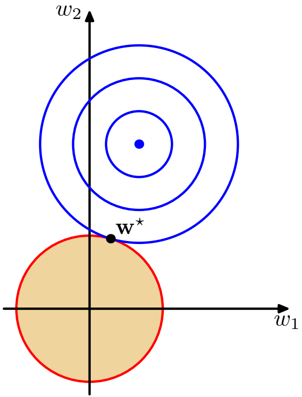
Recall the definition of the Ridge Regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; . \tag{33}$$

The Ridge Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_2^2 \leq \lambda \tag{34}$$

Geometric interpretation for the Ridge Regression



• Blue lines indicate the level sets of the un-regularized MSE.

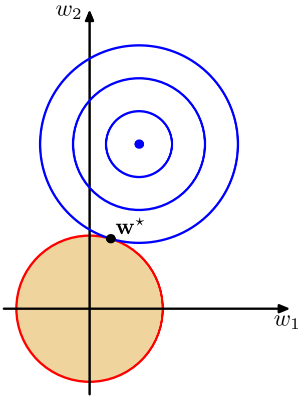
Recall the definition of the Ridge Regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \; . \tag{33}$$

The Ridge Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_2^2 \leq \lambda \tag{34}$$

Geometric interpretation for the Ridge Regression



- Blue lines indicate the level sets of the un-regularized MSE.
- The optimum value for $\mathbf{w}$ is $\mathbf{w}^\star$.

Recall the definition of the Ridge Regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \; + \; \lambda \|\mathbf{w}\|_2^2 \, . \tag{33}$$

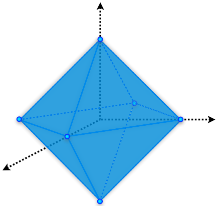
The Ridge Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_2^2 \leq \lambda \tag{34}$$

Geometric interpretation for the Lasso Regression

Similarly, the Lasso Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_1 \leq \lambda \tag{35}$$

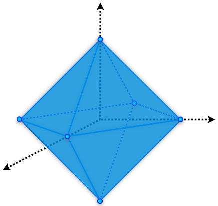


A “ball” of constant L_1 norm

Geometric interpretation for the Lasso Regression

Similarly, the Lasso Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N} \|\mathbf{y}-\mathbf{Xw}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_1 \leq \lambda \tag{35}$$



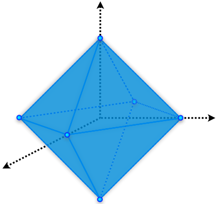
A “ball” of constant L_1 norm

- The optimal point is somewhere on the surface of this “ball”.

Geometric interpretation for the Lasso Regression

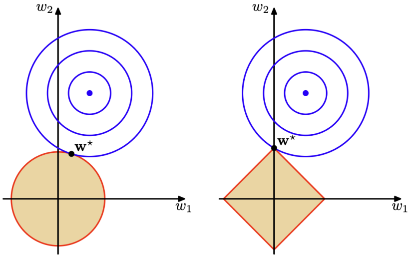
Similarly, the Lasso Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_1 \leq \lambda \tag{35}$$



A "ball" of constant L_1 norm

- The optimal point is somewhere on the surface of this "ball".

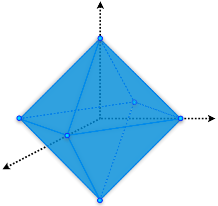


L_2-norm. L_1-norm.

Geometric interpretation for the Lasso Regression

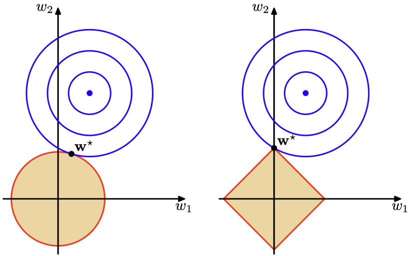
Similarly, the Lasso Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_1 \leq \lambda \tag{35}$$



A "ball" of constant L_1 norm

- The optimal point is somewhere on the surface of this "ball".



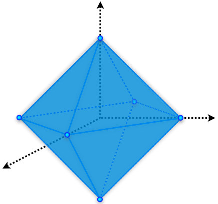
L_2-norm. L_1-norm.

- This forces some of the elements of $\mathbf{w}$ to be strictly 0.

Geometric interpretation for the Lasso Regression

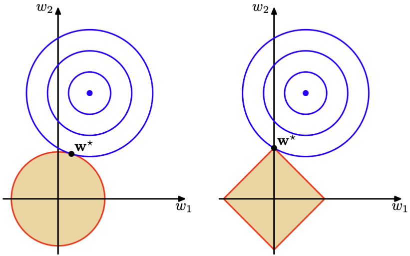
Similarly, the Lasso Regression can be reformulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \quad \text{such that } \|\mathbf{w}\|_1 \leq \lambda \tag{35}$$



A "ball" of constant L_1 norm

- The optimal point is somewhere on the surface of this "ball".



L_2-norm. L_1-norm.

- This forces some of the elements of $\mathbf{w}$ to be strictly 0.
- As λ is increased, an increasing number of parameters are driven to 0.

Table of Contents

The probabilistic interpretation of least-squares linear regression

Least-Squares linear regression can be interpreted as the Maximum Likelihood Estimator:

$$
\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w}) && \text{(by the definition of log-likelihood)}\\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}|\mathbf{w}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(by factoring the likelihood)}\\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{X}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && \text{(the choice of the input } \mathbf{x}_n \text{ is independent of } \mathbf{w})\\
&\overset{(d)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) && (p(\mathbf{X}) \text{ is independent of } \mathbf{w})\\
&\overset{(e)}{=} \arg\min_{\mathbf{w}} \quad -\log \left[\prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w})\right] && \text{(we assume samples are iid.)}\\
&= \arg\min_{\mathbf{w}} \quad \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 && \text{(by definition and calculus)}
\end{aligned}
$$

The probabilistic interpretation of Ridge Regression

We start with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ and chose $\mathbf{w}$ to maximize this posterior.

The Maximum-A-Posteriori (MAP) estimate:

$$
\begin{aligned}
\mathbf{w}_{\text{ridge}} &= \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) && \text{(by the definition of posterior)} \\
&\overset{(a)}{=} \arg\min_{\mathbf{w}} \quad -\log \frac{p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}, \mathbf{X})} && \text{(by the Bayes' law)} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)} \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} \quad -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) && \text{(eliminate quantities that do not depend on } \mathbf{w}\text{)} \\
&= \arg\min_{\mathbf{w}} \quad \sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 . && (36)
\end{aligned}
$$

Table of Contents

Table of Contents

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^{\top}\mathbf{w})^2 \quad + \quad \boxed{\lambda} \, \|\mathbf{w}\|^2 \tag{37}$$

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \lambda \left\|\mathbf{w}\right\|^2 \tag{37}$$

- λ can be tuned to control the model complexity

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda} \, \|\mathbf{w}\|^2 \tag{37}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda} \; \|\mathbf{w}\|^2 \tag{37}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda} \, \|\mathbf{w}\|^2 \tag{37}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

Similarly, in Polynomial Regression: $(x_1, x_2) \xrightarrow{\phi} \begin{matrix} (x_1, x_2, x_1^2 + x_2^2) \\ (x_1, x_2, 5x_1^2 + 2x_2^2, x_2^3 + 2x_1) \end{matrix}$

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \boxed{\lambda} \, \|\mathbf{w}\|^2 \tag{37}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

Similarly, in Polynomial Regression: $(x_1, x_2) \xrightarrow{\phi} \begin{matrix} (x_1, x_2, x_1^2 + x_2^2) \\ (x_1, x_2, 5x_1^2 + 2x_2^2, x_2^3 + 2x_1) \end{matrix}$

- we can enrich the model complexity, by augmenting the feature vector $\mathbf{x}$

What is the model selection problem?

Recall the Ridge Regression problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \quad + \quad \lambda \; \|\mathbf{w}\|^2 \tag{37}$$

- λ can be tuned to control the model complexity
- In practice: $(\lambda_1, \ldots, \lambda_k) \longrightarrow$ Algorithm $\longrightarrow (\mathbf{w}_1, \ldots, \mathbf{w}_k)$.
- Which λ should we use?

Similarly, in Polynomial Regression: $(x_1, x_2) \xrightarrow{\phi} \begin{array}{c} (x_1, x_2, x_1^2 + x_2^2) \\ (x_1, x_2, 5x_1^2 + 2x_2^2, x_2^3 + 2x_1) \end{array}$

- we can enrich the model complexity, by augmenting the feature vector $\mathbf{x}$

How do we choose these hyper-parameters?

Model selection for neural networks

Optimization Algorithms?

SGD
Adam
Which step-size?
Which batch-size?
Which momentum?

Neural Architectures?

FullyConnected
ConvNet
ResNet
Transformer
Which width?
Which depth?
Batch normalization?

Regularizations?

Weight decay?
Dropout?
Early stopping?
Data augmentation?

To give a meaningful answer to the above questions,
we first need to specify our data model!

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{38}$$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{38}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{39}$$

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} \,. \tag{38}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{39}$$

- f_S denotes the output.

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{38}$$

Learning algorithm:

$$\boxed{f_S} = \boxed{\mathcal{A}}(\boxed{S}) \tag{39}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{38}$$

Learning algorithm:

$$f_S = \mathcal{A}(S) \tag{39}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.
- S denotes the input (dataset).

Probabilistic setup

Data model:

- We assume an (unknown) underlying distribution $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$
- We see a dataset S of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \quad \text{i.i.d.} \sim \mathcal{D}\}_{n=1}^{N} . \tag{38}$$

Learning algorithm:

$$\boxed{f_S} = \boxed{\mathcal{A}}(\boxed{S}) \tag{39}$$

- f_S denotes the output.
- $\mathcal{A}$ denotes the learning algorithm.
- S denotes the input (dataset).
- Can add a subscript $f_{S,\lambda}$ to indicate the model dependency (for Ridge Regression).

Given a model f, how can we assess if f is any good?

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{40}$$

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{40}$$

- The quantity $\mathcal{L}_{\mathcal{D}}(f)$ has many names $\left\{ \begin{array}{l} \textit{True} \\ \textit{Expected} \end{array} \right. \left\{ \begin{array}{l} \textit{Risk} \\ \textit{Error} \\ \textit{Loss} \end{array} \right.$

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{40}$$

- The quantity $\mathcal{L}_{\mathcal{D}}(f)$ has many names $\left\{ \begin{array}{l} \textit{True} \\ \textit{Expected} \end{array} \right. \left\{ \begin{array}{l} \textit{Risk} \\ \textit{Error} \\ \textit{Loss} \end{array} \right.$

This is the quantity we are fundamentally interested in

Given a model f, how can we assess if f is any good?

- We should compute the **expected error** over all samples chosen according to $\mathcal{D}$:

$$\mathcal{L}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}\left[\ell(y, f(\mathbf{x}))\right] \qquad \text{where } \ell(\cdot,\cdot) \text{ is our loss function} \tag{40}$$

- The quantity $\mathcal{L}_{\mathcal{D}}(f)$ has many names $\left\{\begin{array}{l} \textit{True} \\ \textit{Expected} \end{array}\right. \left\{\begin{array}{l} \textit{Risk} \\ \textit{Error} \\ \textit{Loss} \end{array}\right.$

This is the quantity we are fundamentally interested in
but we cannot estimate it since $\mathcal{D}$ is not known.

Empirical Error: what we can compute

- We are given some data S

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{41}$$

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{41}$$

- This is called *empirical risk/error/loss*.

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{41}$$

- This is called *empirical risk/error/loss*.
- The samples are random thus $L_S(f)$ is a random variable.

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{41}$$

- This is called *empirical risk/error/loss*.
- The samples are random thus $L_S(f)$ is a random variable.
- It is an unbiased estimator of the true error.

Empirical Error: what we can compute

- We are given some data S
- We can approximate the true error by **averaging the loss function on the dataset** S

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)) \,. \tag{41}$$

- This is called *empirical risk/error/loss*.
- The samples are random thus $L_S(f)$ is a random variable.
- It is an unbiased estimator of the true error.
- **Generalization gap:** $|L_{\mathcal{D}}(f) - L_S(f)|$.

Training Error: what we are minimizing

$$\boxed{\text{The prediction function } f_S \text{ is itself a function of the data } S.}$$

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.
  - Then in fact we are computing

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) \, . \tag{42}$$

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.
  - Then in fact we are computing

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) \,. \tag{42}$$

  - This is called *the training (risk/loss/error)*.

Training Error: what we are minimizing

The prediction function f_S is itself a function of the data S.

Assume that we are given the data S.

- The empirical error is called the **training error**: when the model has been trained on the same data it is applied to, namely,
  - If we first learn the model from S, i.e., we compute $f_S = \mathcal{A}(S)$,
  - and then we compute the empirical risk of f_S using the same data S.
  - Then in fact we are computing

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)) \,. \tag{42}$$

  - This is called *the training (risk/loss/error)*.

- The reason that $L_S(f_S)$ might not be close to $L_{\mathcal{D}}(f_S)$ is of course over-fitting.

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{43}$$

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{43}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{43}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}
2. We **validate** it computing the error on the **test set** S_{test}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{44}$$

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{43}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}
2. We **validate** it computing the error on the **test set** S_{test}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n, \mathbf{x}_n) \in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{44}$$

- Since S_{train} and S_{test} are independent, we hope that $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$

Splitting the data and Test Error

So how we can address the potential over-fitting problem that occurs
if we train and test the model on the same data.

- **Problem:** Validating model on the same data subset we trained it on!
- **Fix:** Split the data into a *training* set S_{train} and a *test* set S_{test} (a.k.a. *validation* set):

$$S = S_{\text{train}} \bigcup S_{\text{test}} \tag{43}$$

1. We **learn** the function $f_{S_{\text{train}}}$ using the train set S_{train}
2. We **validate** it computing the error on the **test set** S_{test}:

$$L_{S_{\text{test}}}(f_{S_{\text{train}}}) = \frac{1}{|S_{\text{test}}|} \sum_{(y_n,\mathbf{x}_n)\in S_{\text{test}}} \ell\left(y_n, f_{S_{\text{train}}}(\mathbf{x}_n)\right). \tag{44}$$

- Since S_{train} and S_{test} are independent, we hope that $L_{S_{\text{test}}}(f_{S_{\text{train}}}) \approx L_{\mathcal{D}}(f_{S_{\text{train}}})$
- Issues: we have fewer data both for the learning and validation tasks (trade-off)

Generalization gap: How far is the test from the true error?

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

Generalization gap: How far is the test from the true error?

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).

Generalization gap: How far is the test from the true error?

Assume that we have a model f, and that our loss function $\ell(\cdot, \cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{45}$$

Generalization gap: How far is the test from the true error?

Assume that we have a model f, and that our loss function $\ell(\cdot,\cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{45}$$

- The true error is

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x}) \sim \mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right].$$

Generalization gap: How far is the test from the true error?

Assume that we have a model f, and that our loss function $\ell(\cdot,\cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{45}$$

- The true error is

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x}) \sim \mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right].$$

- How far are these apart?

Generalization gap: How far is the test from the true error?

Assume that we have a model f, and that our loss function $\ell(\cdot,\cdot)$ is bounded.

- We are given a test set S_{test} chosen i.i.d. from the underlying distribution $\mathcal{D}$ (and this test set was *not* used to train the model).
- The test/empirical error is

$$L_{S_{\text{test}}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(\mathbf{x}_n, y_n) \in S_{\text{test}}} \ell\left(y_n, f(\mathbf{x}_n)\right). \tag{45}$$

- The true error is

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(y,\mathbf{x})\sim\mathcal{D}}\left[\ell\left(y, f(\mathbf{x})\right)\right].$$

- How far are these apart?

Definition 4 (Generalization error)

The *generalization error* is given by

$$\left|L_{\mathcal{D}}(f) - L_{S_{\text{test}}}(f)\right|.$$

This lecture:

- Over-fitting and under-fitting
- Polynomial Regression, Ridge Regression, and Lasso Regression
- Generalization, and Model Selection

This lecture:

- Over-fitting and under-fitting
- Polynomial Regression, Ridge Regression, and Lasso Regression
- Generalization, and Model Selection

Next lecture:

- Model Selection (contd)
- Bias-Variance Decomposition
- Multi-Layer Perceptron (MLP)
- Back-Propagation (BP)