Machine Learning

Lecture 4: Linear Regression

Prof. Dr. Stephan Günnemann

Data Analytics and Machine Learning
Technical University of Munich

Winter term 2020/2021

Notation

Symbol	Meaning
x	scalar is lowercase and not bold
$\boldsymbol{x}$	vector is lowercase and bold
$\boldsymbol{\Sigma}$	matrix is uppercase and bold
$f(\boldsymbol{x})$	predicted value for inputs $\boldsymbol{x}$
$\boldsymbol{y}$	vector of targets
y_i	target of the i'th example
w_0	bias term (not to be confused with bias in general)
$\phi(\cdot)$	basis function
$E(\cdot)$	error function
$\mathcal{D}$	training data
$\boldsymbol{X}^\dagger$	Moore-Penrose pseudoinverse of $\boldsymbol{X}$

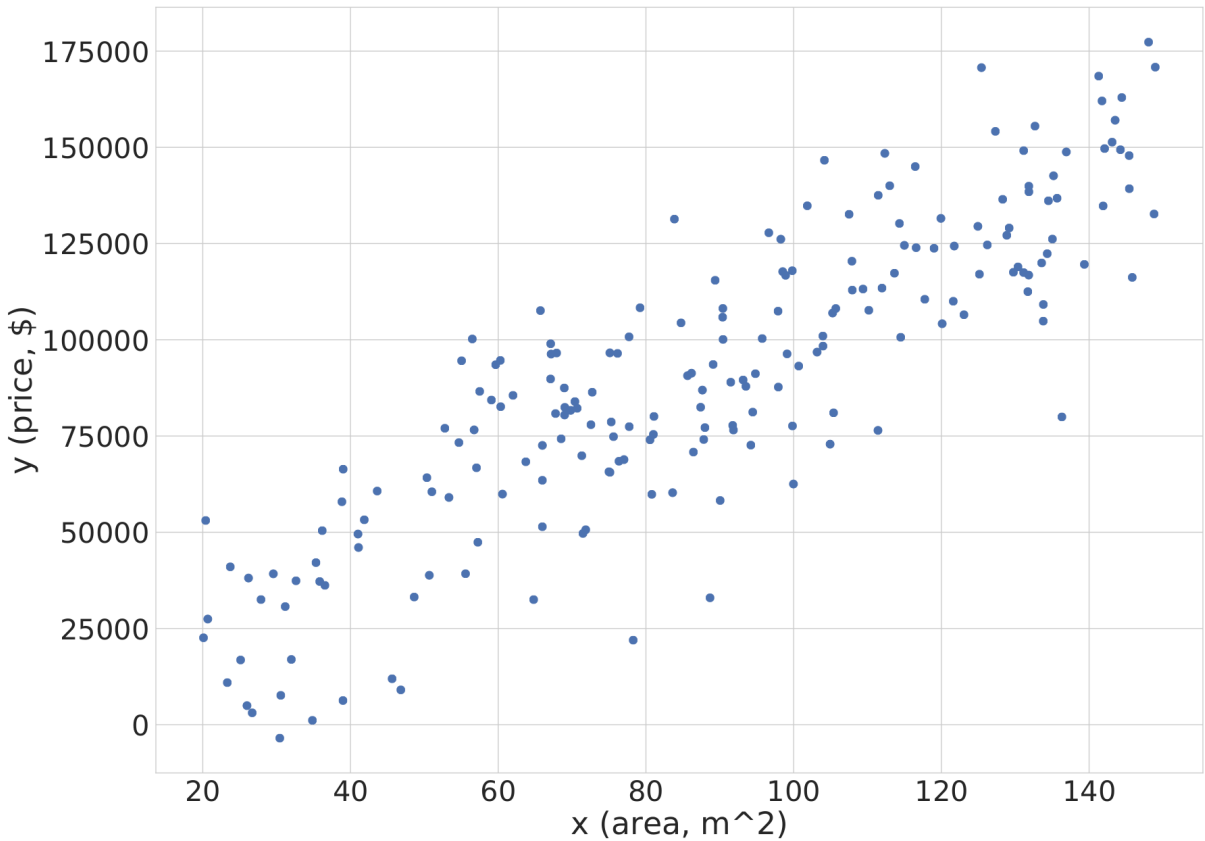
There is not a special symbol for vectors or matrices augmented by the bias term, w_0. Assume it is always included.

Section 1

Basic Linear Regression

Example: Housing price prediction

Given is a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, of house areas x_i and corresponding prices y_i.



How do we estimate a price of a new house with area x_{new}?

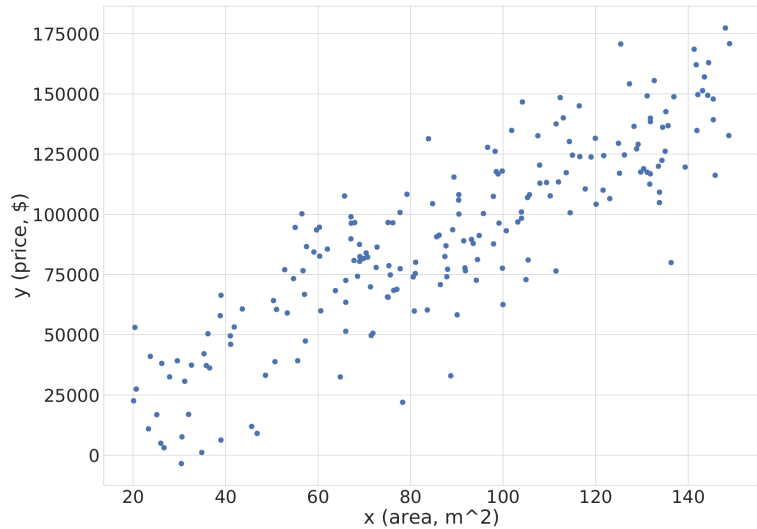
Regression problem

Line-Fitting Problem

Given

- observations [1]
  $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_i \in \mathbb{R}^D$
- targets
  $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$, $\quad y_i \in \mathbb{R}$

1-d



Find

- Mapping $f(\cdot)$ from inputs to targets

$$y_i \approx f(\boldsymbol{x}_i)$$

D: # features
N: # samples

[1] A common way to represent the samples is as a data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, where each row represents one sample.

Linear model

Target y is generated by a deterministic function f of $\boldsymbol{x}$ plus noise

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \beta^{-1}) \tag{1}$$

some noise

Let's choose $f(\boldsymbol{x})$ to be a linear function

$$f_{\boldsymbol{w}}(\boldsymbol{x}_i) = w_0 + w_1 x_{i1} + w_2 x_{i2} + ... + w_D x_{iD} \tag{2}$$

$$= w_0 + \boldsymbol{w}^T \boldsymbol{x}_i \tag{3}$$

From now we will always assume that the bias term is absorbed into the $\boldsymbol{x}$ vector

Absorbing the bias term

The linear function is given by

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_D x_D \tag{4}$$

$$= w_0 + \boldsymbol{w}^T \boldsymbol{x} \tag{5}$$

Here w_0 is called bias or offset term. For simplicity, we can "absorb" it by prepending a 1 to the feature vector $\boldsymbol{x}$ and respectively adding w_0 to the weight vector $\boldsymbol{w}$:

Each has dim of D+1

$$\tilde{\boldsymbol{x}} = (1, x_1, ..., x_D)^T \qquad \tilde{\boldsymbol{w}} = (w_0, w_1, ..., w_D)^T$$

The function $f_{\boldsymbol{w}}$ can compactly be written as $f_{\boldsymbol{w}}(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}$.

To unclutter the notation, we will assume the bias term is always absorbed and write $\boldsymbol{w}$ and $\boldsymbol{x}$ instead of $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{x}}$.

Now, how do we choose the "best" $\boldsymbol{w}$ that fits our data?

Learning Phase

Loss function

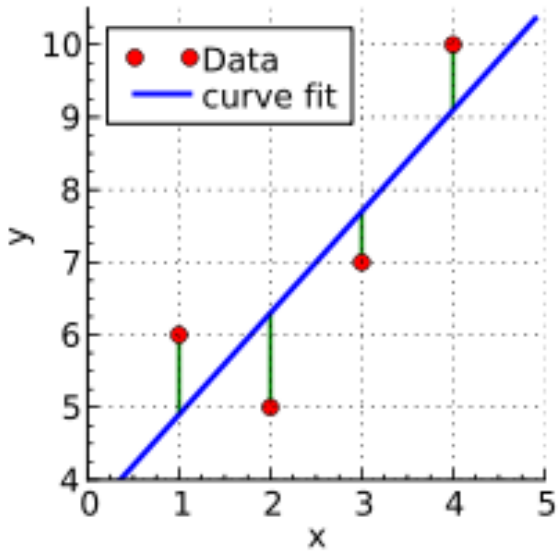
Different loss functions lead to different interpretations.

A loss function measures the "misfit" or error between our model (parametrized by $\boldsymbol{w}$) and observed data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$.

Standard choice - least squares (LS)

$$E_{\mathrm{LS}}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 \quad (6)$$

$$= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 \quad (7)$$



Notice: we are measuring the error bet model prediction (x) & target (y); which is the green line in the graph. We are measuring wrt the y-axis.

We are NOT measuring the perpendicular distance from point to the line!

Factor $\frac{1}{2}$ is for later convenience

Objective

Find the optimal weight vector $\boldsymbol{w}^\star$ that minimizes the error

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) \tag{8}$$

$$= \arg\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{x}_i^T \boldsymbol{w} - y_i)^2 \tag{9}$$

By stacking the observations $\boldsymbol{x}_i$ as rows of the matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$

Matrix Notation

$$= \arg\min_{\boldsymbol{w}} \frac{1}{2} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) \tag{10}$$

Dimensionality; X: N*D
w: D*1
y: N*1

Optimal solution

To find the minimum of the loss $E(\boldsymbol{w})$, compute the gradient $\nabla_{\boldsymbol{w}} E(\boldsymbol{w})$:

$$\nabla_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \frac{1}{2} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) \tag{11}$$

$$= \nabla_{\boldsymbol{w}} \frac{1}{2} \left(\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y} \right) \tag{12}$$

$$= \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y} \tag{13}$$

(69) $\frac{\partial \boldsymbol{x}^T \boldsymbol{a}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{a}^T \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}$

(81) $\frac{\partial \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x}}{\partial \boldsymbol{x}} = (\boldsymbol{B} + \boldsymbol{B}^T)\boldsymbol{x}$

See Equations (69), (81) from Matrix cookbook for details

Now set the gradient to zero and solve for $\boldsymbol{w}$ to obtain the minimizer [2]

$$\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y} \stackrel{!}{=} 0 \tag{14}$$

This leads to the so-called normal equation of the least squares problem

$$\boldsymbol{w}^* = \underbrace{(\boldsymbol{X}^\mathsf{T} \boldsymbol{X})^{-1} \boldsymbol{X}^\mathsf{T}}_{=\boldsymbol{X}^\dagger \text{ D*N dim}} \boldsymbol{y} \quad \text{N*1 dim} \tag{15}$$

$\boldsymbol{X}^\dagger$ is called Moore-Penrose pseudo-inverse of $\boldsymbol{X}$ (because for an invertible square matrix, $\boldsymbol{X}^\dagger = \boldsymbol{X}^{-1}$).

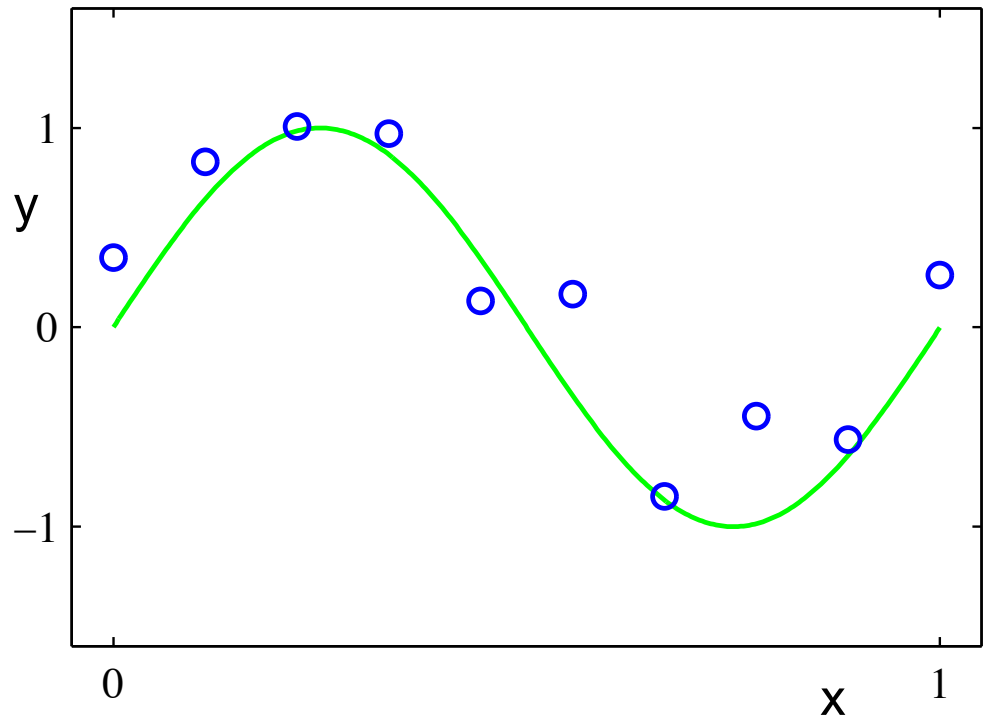
It's generalization of inverse matrix.

One common use is computing best fit "least squares" soln to a sys of linear eq that lacks a soln.
Another use is finding the minimum (Euclidean) norm soln to a sys of linear eq that has multiple soln.

[2] Because Hessian $\nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} E(\boldsymbol{w})$ is positive (semi)definite $\rightarrow$ see *Optimization*

Nonlinear dependency in data

What if the dependency between y and x is not linear?



Data generating process: $y_i = \sin(2\pi x_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \beta^{-1})$

Non Linear Fn

For this example assume that the data dimensionality is $D = 1$ for simplicity

Polynomials

think about this as preprocessing the data. i.e. feature extraction

Solution: Polynomials are universal function approximators, so for 1-dimensional x we can define f as

M: degree of polynomial

$$f_{\boldsymbol{w}}(x) = w_0 + \sum_{j=1}^{M} w_j x^j \tag{16}$$

Or more generally

Instead of polynomial, we can use whole transformation to input data

$$= w_0 + \sum_{j=1}^{M} w_j \phi_j(x) \tag{17}$$

Basis Transformation

Define $\phi_0 = 1$

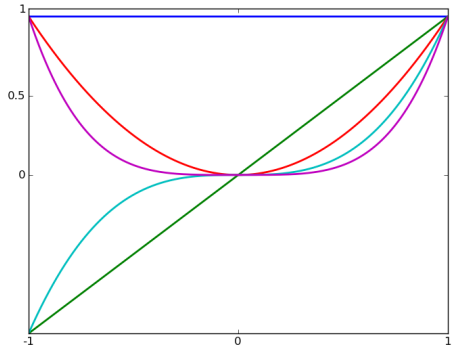
corresponds to absorbing bias term

$$= \boldsymbol{w}^T \boldsymbol{\phi}(x) \tag{18}$$

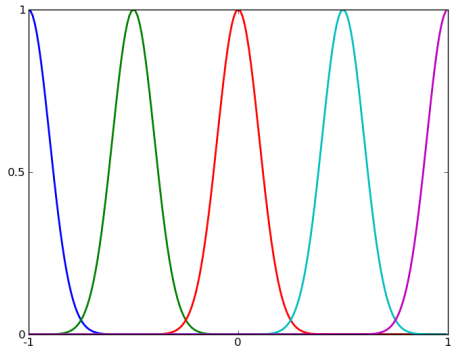
The function f is still linear in $\boldsymbol{w}$ (despite not being linear in x)!

Typical basis functions

Polynomials $\phi_j(x) = x^j$

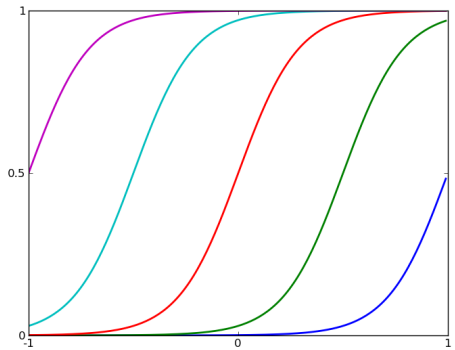


Gaussian $\phi_j(x) = \mathrm{e}^{\frac{-(x-\mu_j)^2}{2s^2}}$



Logistic Sigmoid $\phi_j(x) = \sigma(\frac{x-\mu_j}{s})$,

where $\sigma(a) = \frac{1}{1+\mathrm{e}^{-a}}$



Linear basis function model

For d-dimensional data $\boldsymbol{x}$: $\phi_j : \mathbb{R}^d \to \mathbb{R}$

Prediction for one sample

φ is mapping input x from d to M+1

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + \sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{19}$$

Using the same least squares error function as before

such arbitrary mapping could be done dimension-wise.
i.e. different operation at each dim

$$E_{\mathrm{LS}}(\boldsymbol{w}) = \frac{1}{2} \sum^{N} \left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i \right)^2 = \frac{1}{2} (\boldsymbol{\Phi w} - \boldsymbol{y})^T (\boldsymbol{\Phi w} - \boldsymbol{y}) \tag{20}$$

with

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \dots & \phi_M(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & & \vdots \\ \vdots & \vdots & \ddots & \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \dots & \phi_M(\boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N \times (M+1)}$$

N: row for each sample
M+1: col for operation

being the design matrix of $\boldsymbol{\phi}$. is a matrix of basis fn, applied on each input data.

Optimal solution

handling non-linear dependency of input data for linear regression; simplying by transforming input data so to say.

Recall the final form of the least squares loss that we arrived at for the original feature matrix $\boldsymbol{X}$

Linearity

$$E_{\mathrm{LS}}(w) = \frac{1}{2}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

and compare it to the expression we found with the design matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N\times(M+1)}$

Non Linearity

$$E_{\mathrm{LS}}(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}). \tag{21}$$

This means that the optimal weights $\boldsymbol{w}^*$ can be obtained in the same way

$$\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{y} = \boldsymbol{\Phi}^{\dagger}\boldsymbol{y} \tag{22}$$

Compare this to Equation 15:

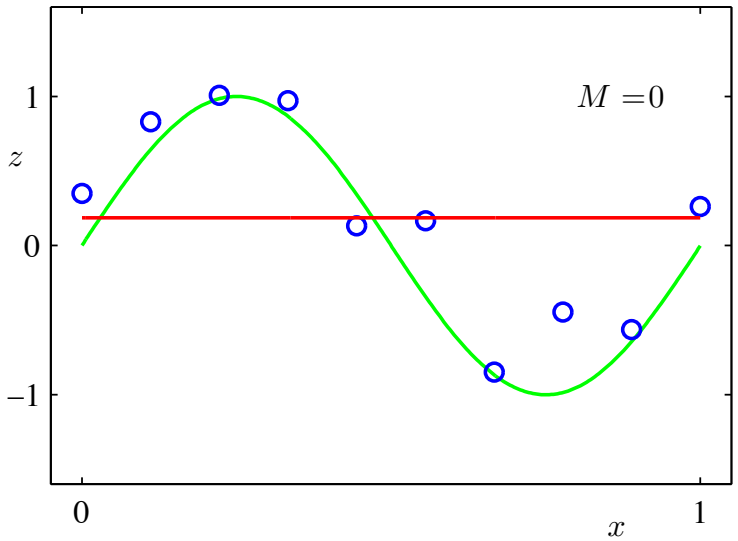
$$\boldsymbol{w}^* = (\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathsf{T}}\boldsymbol{y} = \boldsymbol{X}^{\dagger}\boldsymbol{y} \tag{23}$$

Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

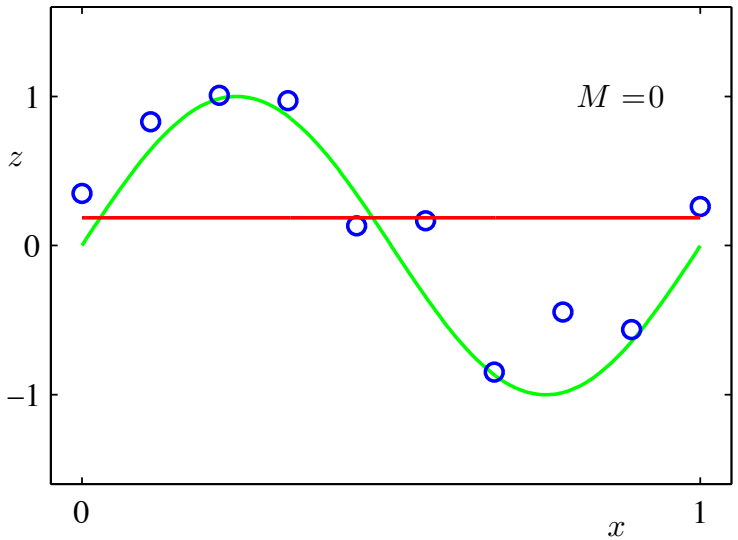
Choosing degree of the polynomial

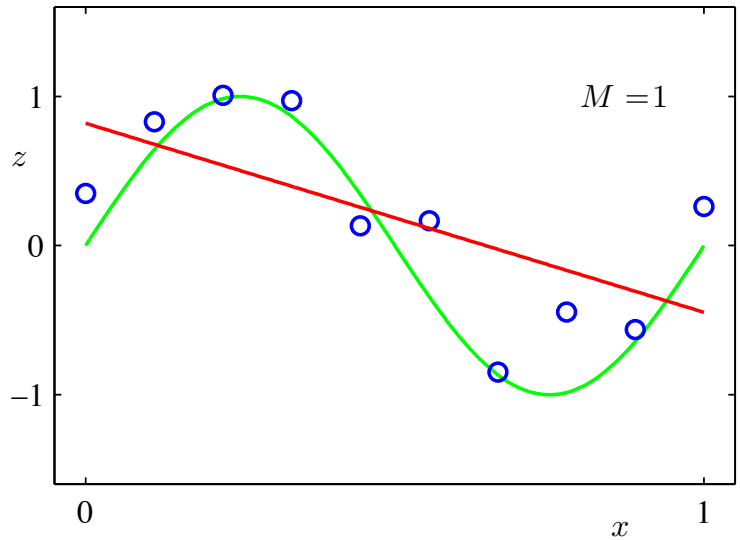
How do we choose the degree of the polynomial M?



Choosing degree of the polynomial

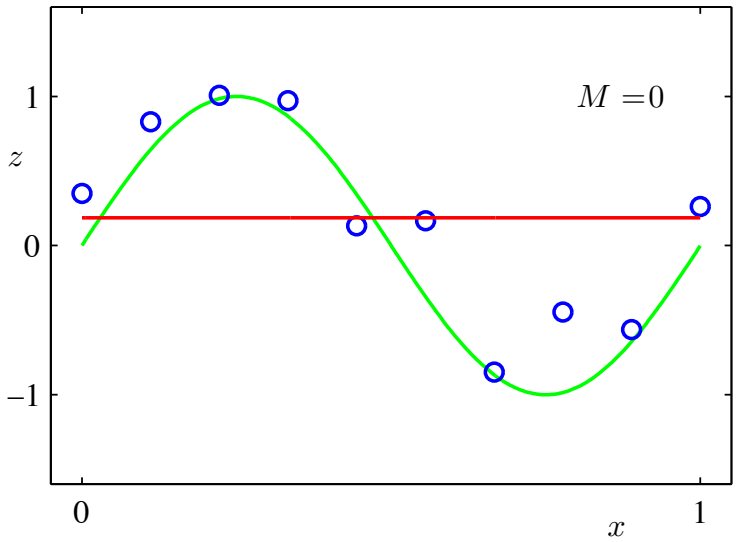
How do we choose the degree of the polynomial M?

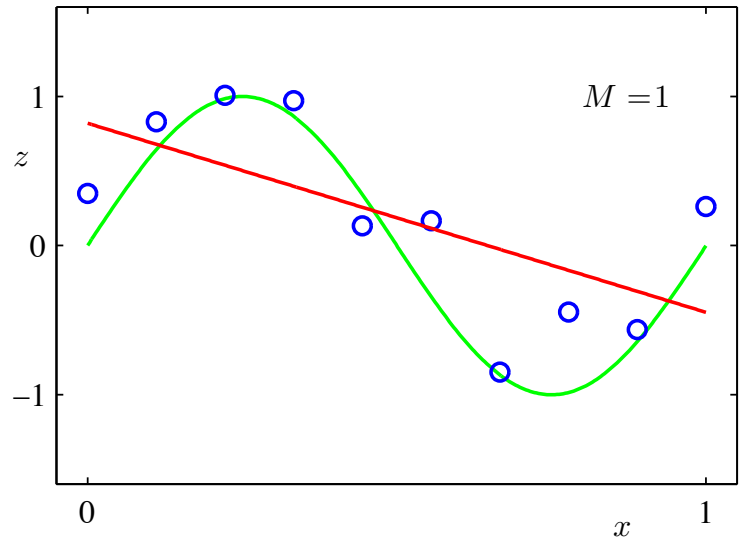


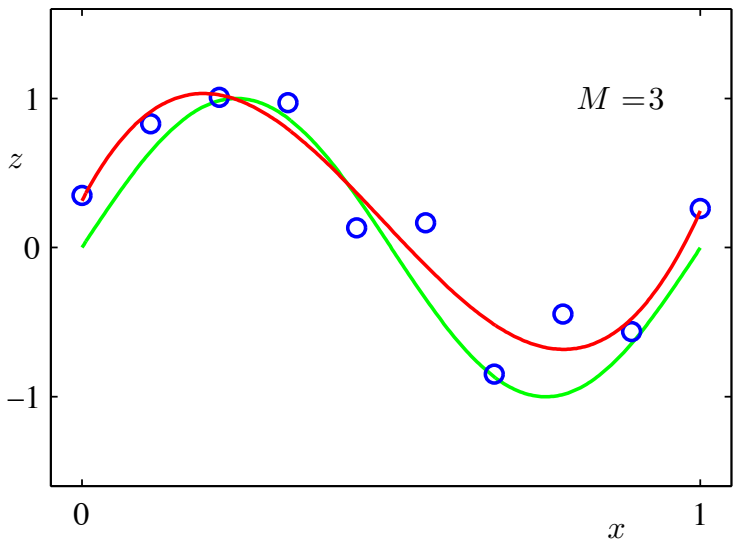


Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

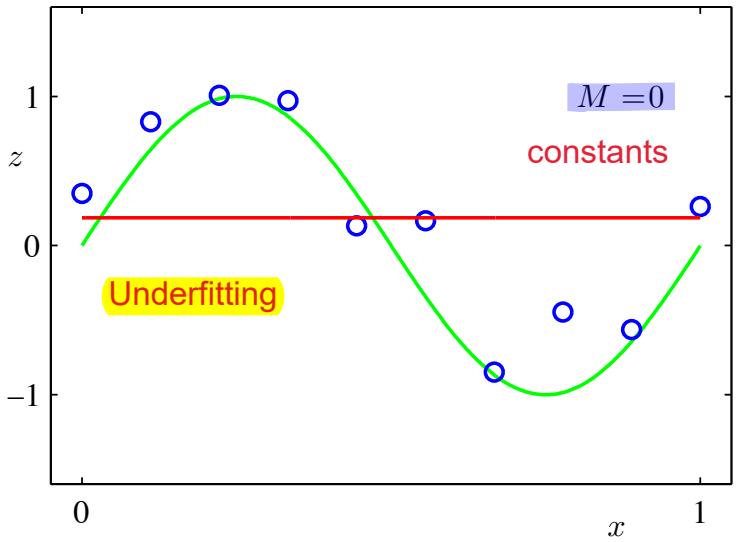


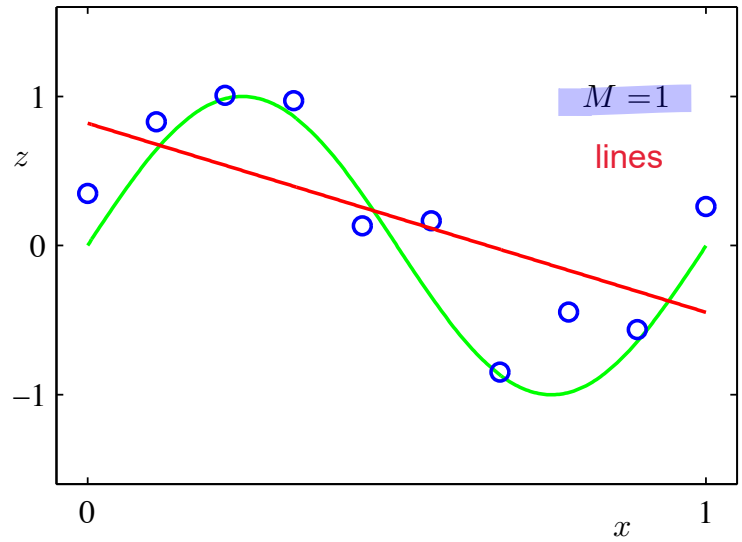


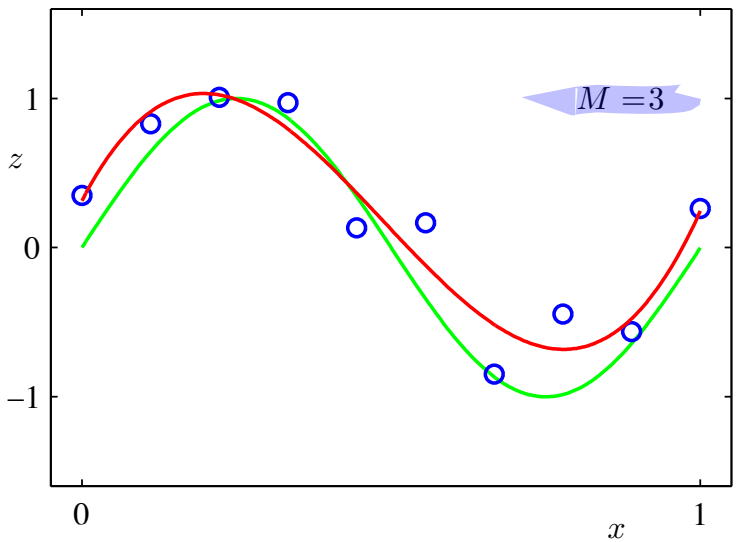


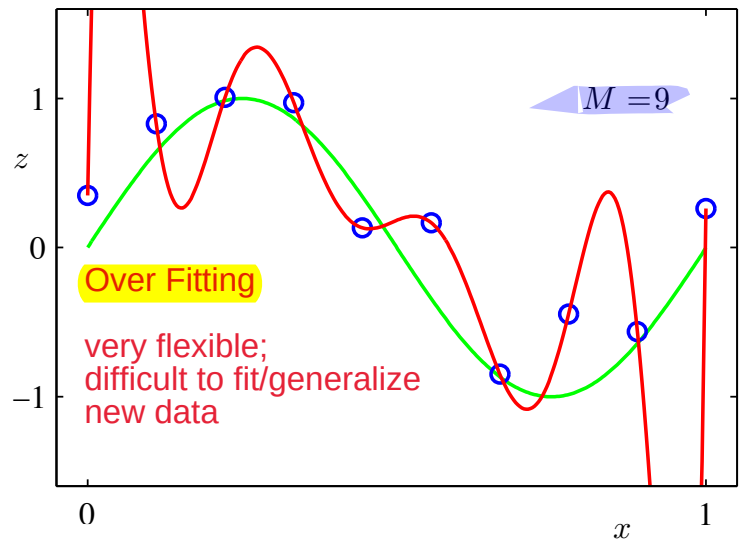
Choosing degree of the polynomial

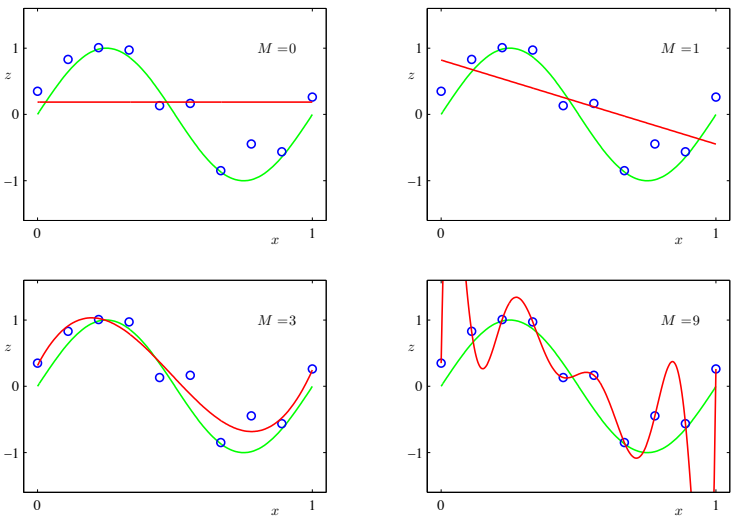
How do we choose the degree of the polynomial M?

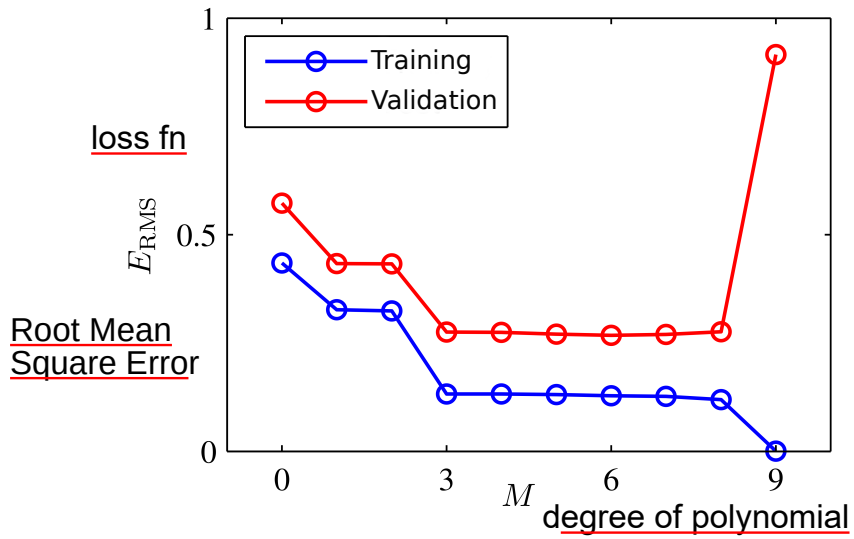










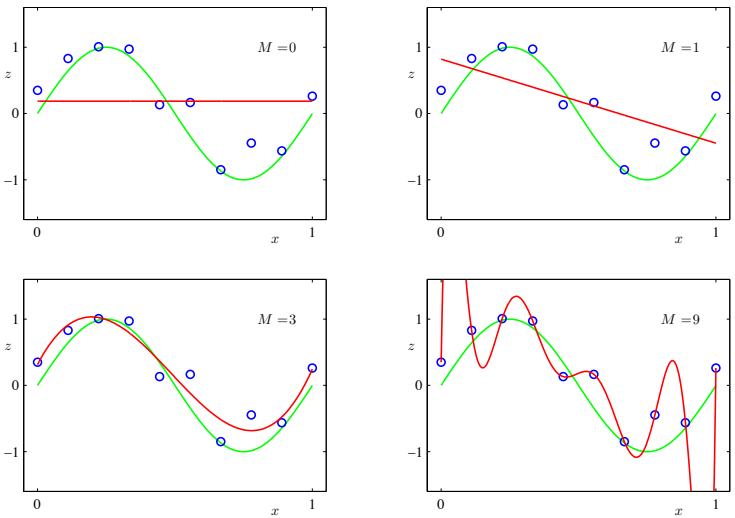


One valid solution is to choose M using the standard train-validation split approach.

train our polynomial on training data and check for the loss

Notice, this approach can be generalized not only to polynomial, but basis transforms as well.

Observing weights



	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

exploding weights
i.e. overfitting

We also make another observation: overfitting occurs when the coefficients $\boldsymbol{w}$ become large.

What if we penalize large weights?

Controlling overfitting with regularization

Least squares loss with L2 regularization (also called ridge regression)

$$E_{\text{ridge}}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i\right]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{24}$$

where

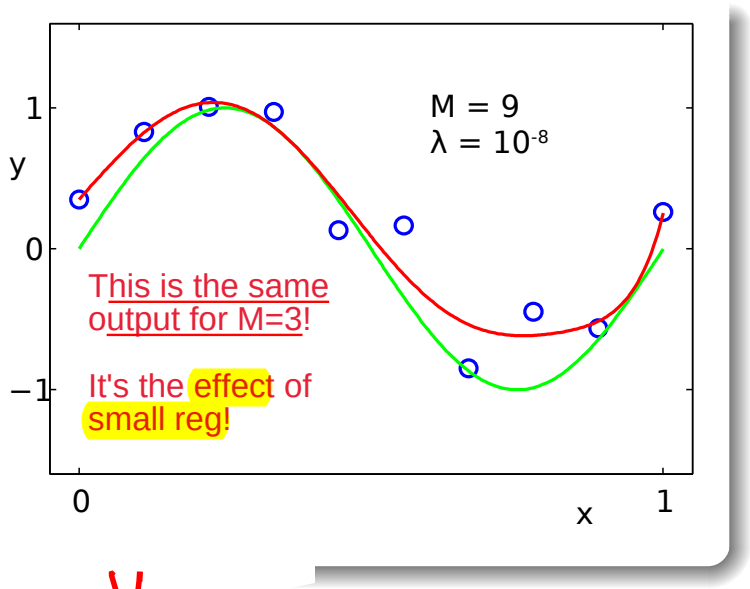
- $\|\boldsymbol{w}\|_2^2 \equiv \boldsymbol{w}^T\boldsymbol{w} = w_0^2 + w_1^2 + w_2^2 + \cdots + w_M^2$ - squared L2 norm of $\boldsymbol{w}$
- λ - regularization strength

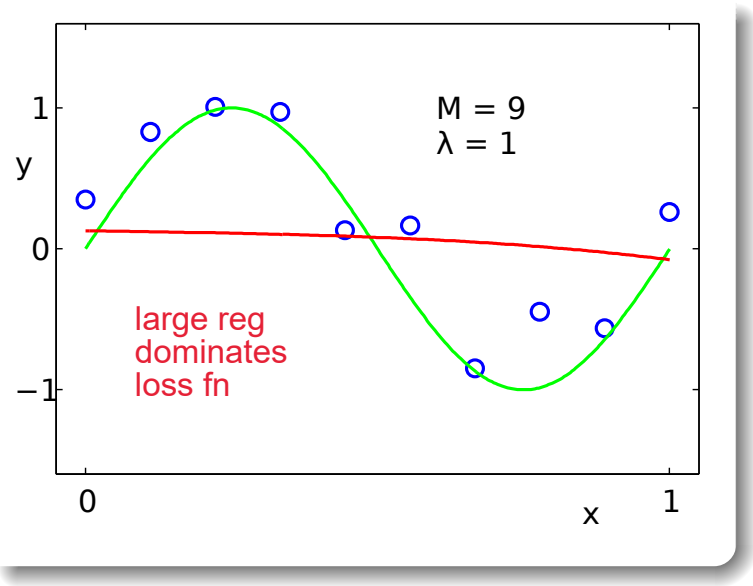
Least squares loss with L2 regularization (also called ridge regression)

$$E_{\text{ridge}}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i\right]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{24}$$

where

- $\|\boldsymbol{w}\|_2^2 \equiv \boldsymbol{w}^T\boldsymbol{w} = w_0^2 + w_1^2 + w_2^2 + \cdots + w_M^2$ - squared L2 norm of $\boldsymbol{w}$
- λ - regularization strength



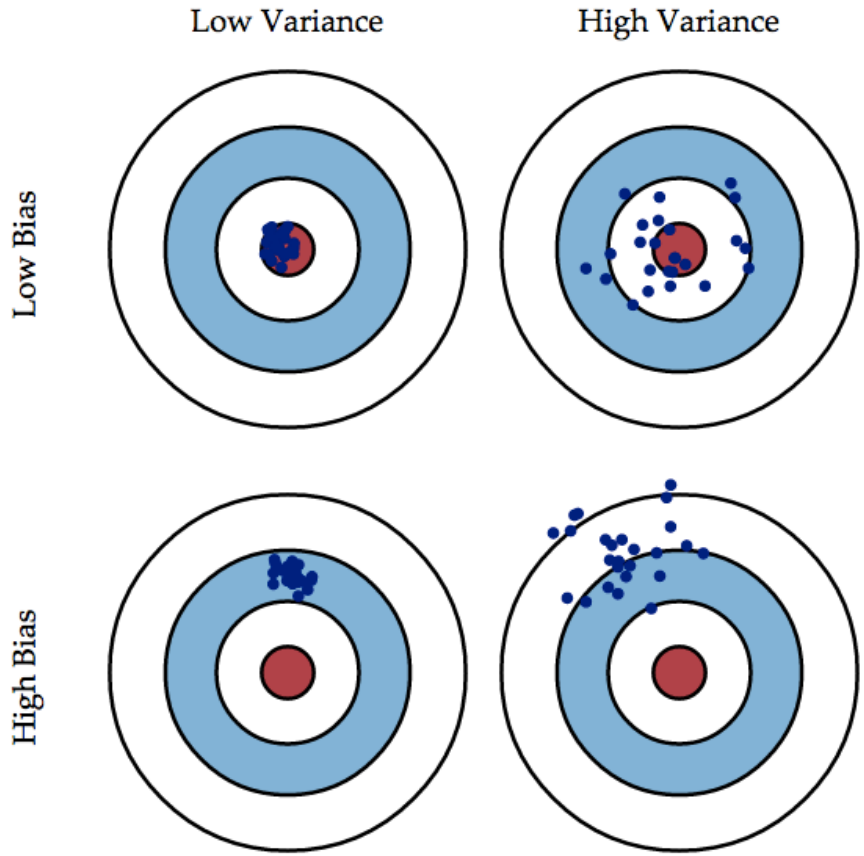


Larger regularization strength λ leads to smaller weights $\boldsymbol{w}$

Bias-variance tradeoff

The error of an estimator can be decomposed into two parts: [3]

- **Bias** - expected error due to model mismatch
- **Variance** - variation due to randomness in training data



the center of the target:
the true model that predicts the correct values.

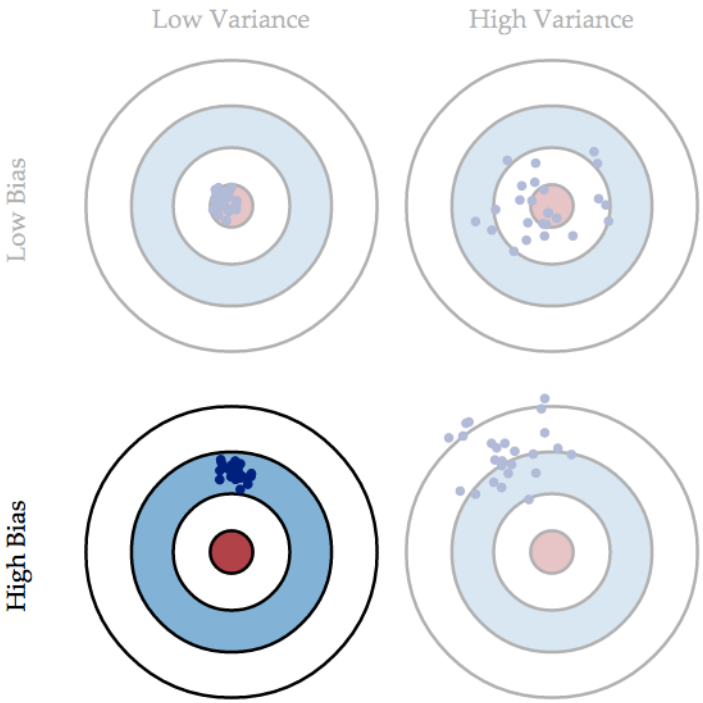
different hits (the blue dots):
different realizations of model given different training data.

[3] See Bishop Section 3.2 for a more rigorous mathematical derivation

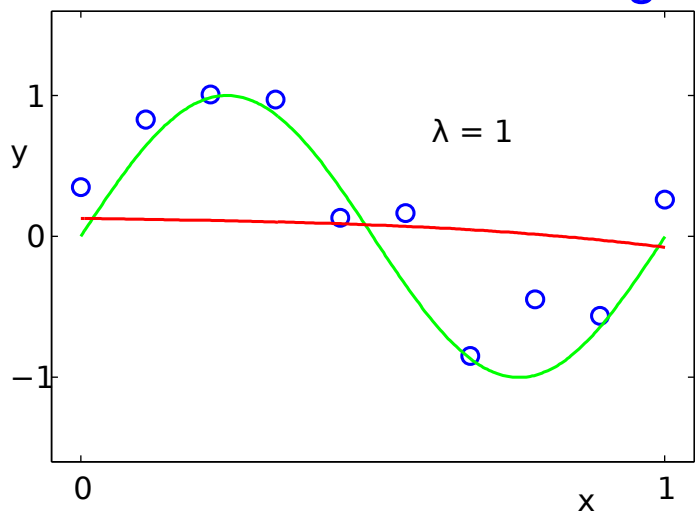
Bias-variance tradeoff: high bias

i.e. model is not even predicting correct results
„you already missed your shot“

- In case of high bias, the model is too rigid to fit the underlying data distribution. Model has low capacity i.e dof
- This typically happens if the model is misspecified and/or the regularization strength λ is too high.



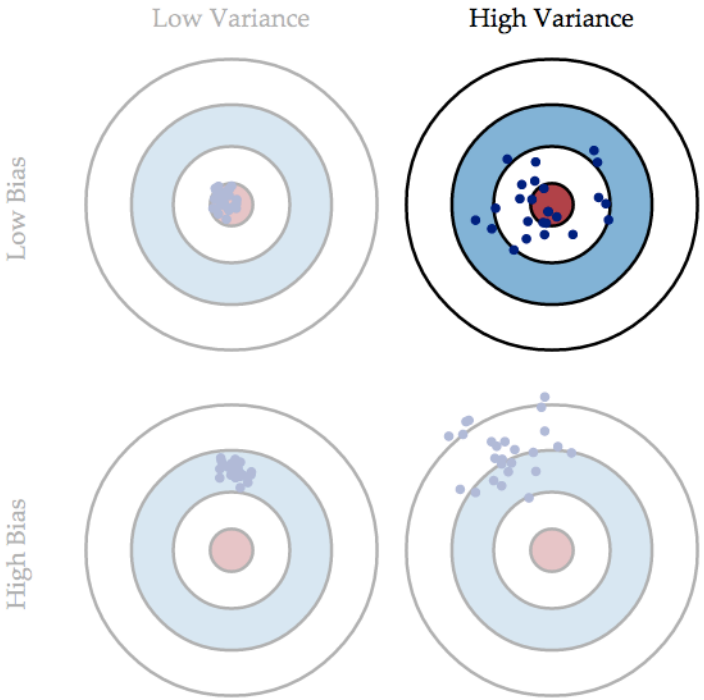
Underfitting

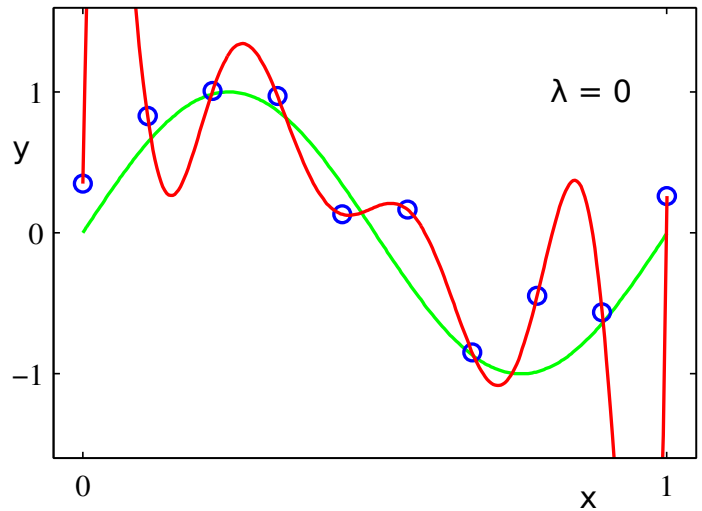


low variance means models being trained on different subsets of data.

Bias-variance tradeoff: high variance

- In case of high variance, the model is too flexible, and therefore captures noise in the data.
- This is exactly what we call overfitting. i.e low inconsistency
- This typically happens when the model has high capacity (= it "memorizes" the training data) and/or λ is too low.



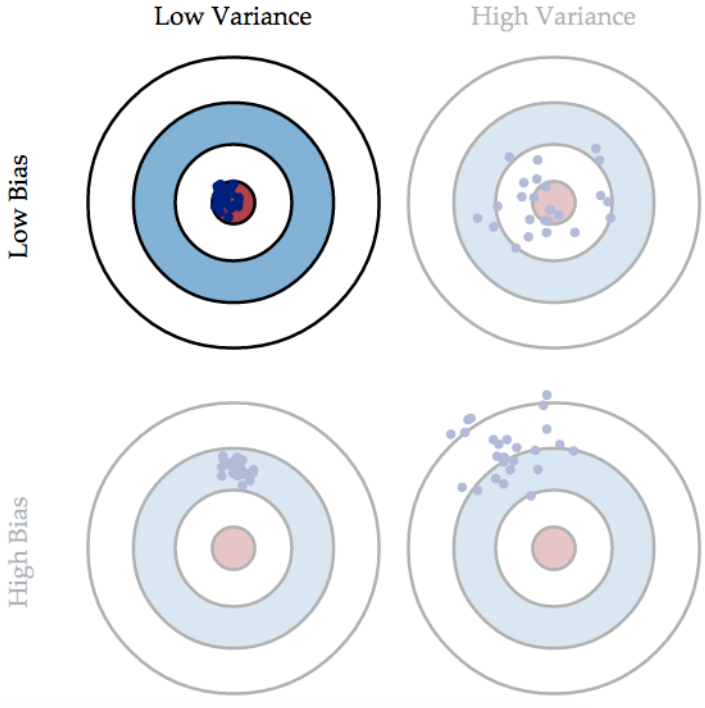


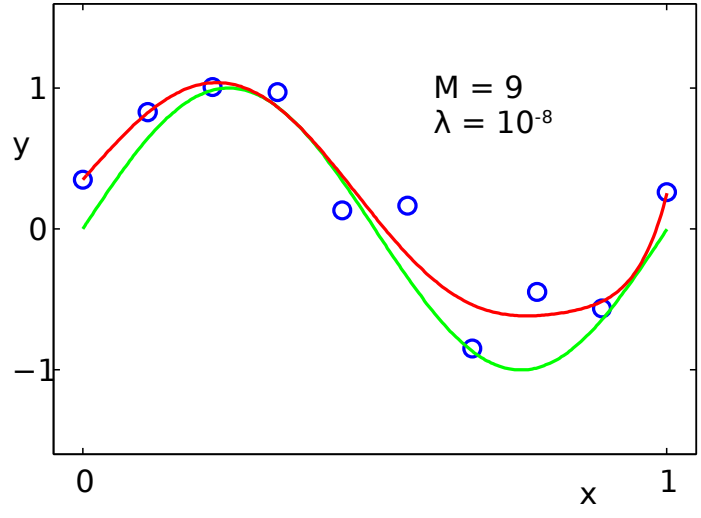
Doing Ensemble here is one good way to reduce such high variance and still yield a good model fit; by averaging over all trained models.

No regularization would be needed in such case.

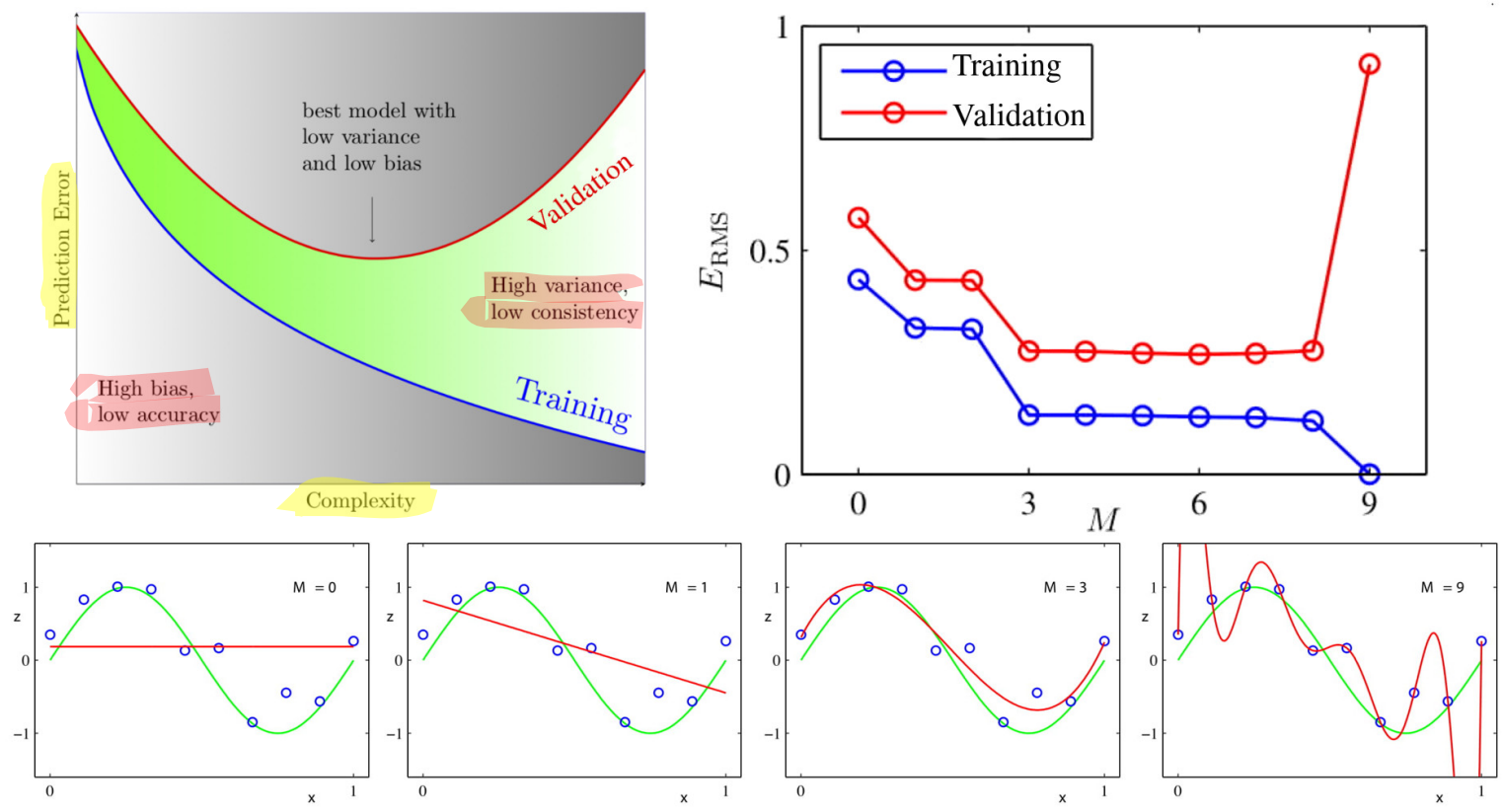
Goal

- Of course, we want models that have low bias and low variance, but often those are conflicting goals.
- A popular technique is to select a model with large capacity i.e high var (high degree polynomial), and keep the variance in check by choosing appropriate regularization strength λ.





- Bias-variance tradeoff in the case of unregularized least squares regression ($\lambda = 0$)

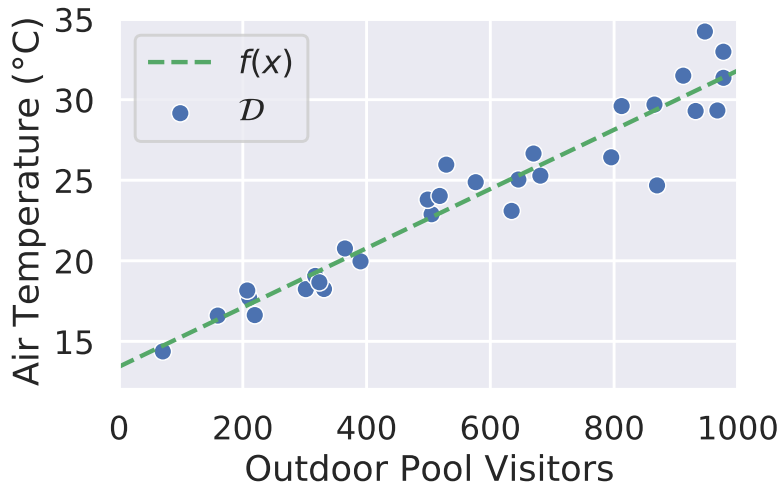


The upper-left figure from: https://eissanematollahi.com/wp-content/uploads/2018/09/Machine-Learning-Basics-1.pdf.

Correlation vs. Causation

Least squares fit

$$f(x) = 0.018x + 13.43$$



- The weights $\boldsymbol{w}_i$ can be interpreted as the strength of the (linear) relationship between feature $\boldsymbol{x}_i$ and $\boldsymbol{y}$ „slope“
- A weight of 0.018 shows a strong correlation (considering the different scales)
  - With actual data, you would normalize the data to handle the different scales of $\boldsymbol{X}$ and $\boldsymbol{y}$ and find a weight of about 1
- But correlation does not imply causation! Putting more people in the pool does not increase the air temperature.

Section 2

Probabilistic Linear Regression

Let graphs show you the path

In the following section, we will use probabilistic graphical models. If you do not know them yet, watch our separate Introduction to PGMs video.

Probabilistic formulation of linear regression

Remember from our problem definition at the start of the lecture,

$$y_i = f_{\boldsymbol{w}}(\boldsymbol{x}_i) + \underbrace{\epsilon_i}_{\text{noise}}$$

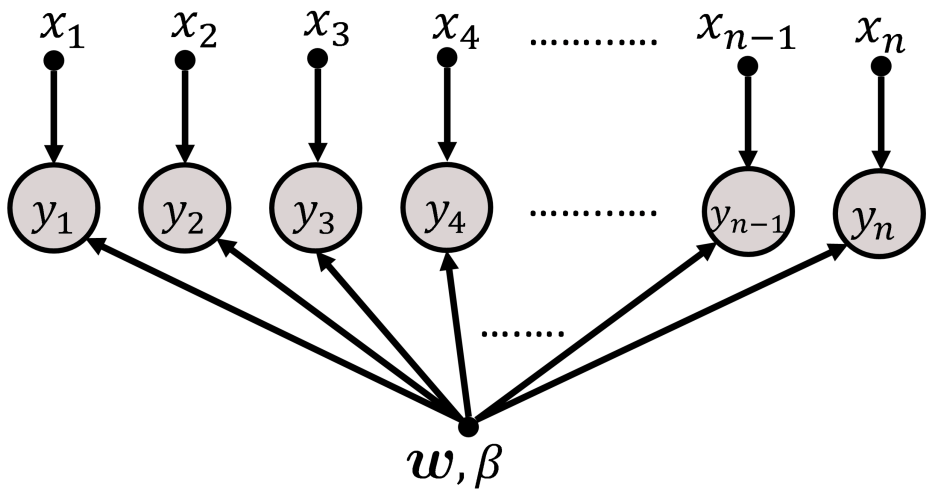
Var: how scattered

Noise has zero-mean Gaussian distribution with a fixed precision $\beta = \frac{1}{\sigma^2}$

$$\epsilon_i \sim \mathcal{N}(0, \beta^{-1})$$

This implies that the distribution of the targets is

$$y_i \sim \mathcal{N}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1})$$



* We try to do the same as with the coin; predicting the output by learning some parameters

---

Remember: any function can be represented as $f_{\boldsymbol{w}}(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i)$

Maximum likelihood

Similar to the coin flip example

Likelihood of a single sample

$$p(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta) = \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \tag{25}$$

Assume that the samples are drawn independently
$\implies$ likelihood of the entire dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ is

$$\mathrm{p}(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{i=1}^{N} p(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta) \tag{26}$$

We can now use the same approach we used in previous lecture - maximize the likelihood w.r.t. $\boldsymbol{w}$ and β

given

$$\boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}} = \arg\max_{\boldsymbol{w}, \beta} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{27}$$

Maximum likelihood

Like in the coin flip example, we can make a few simplifications

$$\boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}} = \arg\max_{\boldsymbol{w},\beta} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{28}$$

applying ln „monotonic fn“ does not change optimal soln

$$= \arg\max_{\boldsymbol{w},\beta} \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{29}$$

$$= \arg\min_{\boldsymbol{w},\beta} -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{30}$$

Let's denote this quantity as maximum likelihood error function that we need to minimize

$$E_{\mathrm{ML}}(\boldsymbol{w}, \beta) = -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{31}$$

Maximum likelihood

Assuming that the targets follow Gaussian dist

Simplify the error function

$$E_{\mathrm{ML}}(\boldsymbol{w}, \beta) = -\ln \left[\prod_{i=1}^{N} \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \right] \tag{32}$$

$$= -\ln \left[\prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \tag{33}$$

$$= -\sum_{i=1}^{N} \ln \left[\sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \tag{34}$$

$$= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \tag{35}$$

Optimizing log-likelihood w.r.t. $\boldsymbol{w}$

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E_{\mathrm{ML}}(\boldsymbol{w}, \beta) \tag{36}$$

$$= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] \tag{37}$$

$$= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} \tag{38}$$

$$= \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) \tag{39}$$

* Whichever value for β chosen, $\boldsymbol{w}$ will always be found

Optimizing log-likelihood w.r.t. $\boldsymbol{w}$

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E_{\mathrm{ML}}(\boldsymbol{w}, \beta) \tag{36}$$

$$= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] \tag{37}$$

$$= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} \tag{38}$$

$$= \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) \tag{39}$$

Maximizing the likelihood is equivalent to minimizing the least squares error function!

assuming targets follow Gaussian dist

$$\boldsymbol{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{y} = \boldsymbol{\Phi}^{\dagger} \boldsymbol{y} \tag{40}$$

Optimizing log-likelihood w.r.t. β

$\frac{\partial}{\partial x} \ln(x) = 1/x$

$\frac{\partial}{\partial x} \log_a(x) = 1/x \ln a$

Plug in the estimate for $\boldsymbol{w}$ and minimize w.r.t. β

$$\beta_{\mathrm{ML}} = \arg\min_{\beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) \tag{41}$$

$$= \arg\min_{\beta} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \right] \tag{42}$$

Take derivative w.r.t. β and set it to zero

$$\frac{\partial}{\partial \beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) = \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2\beta} \stackrel{!}{=} 0 \tag{43}$$

Solving for β

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \tag{44}$$

* Actually, β does not only provide the mean, but also some variation around it i.e. it adds more info to the prediction

β corresponds to the mean squared residual

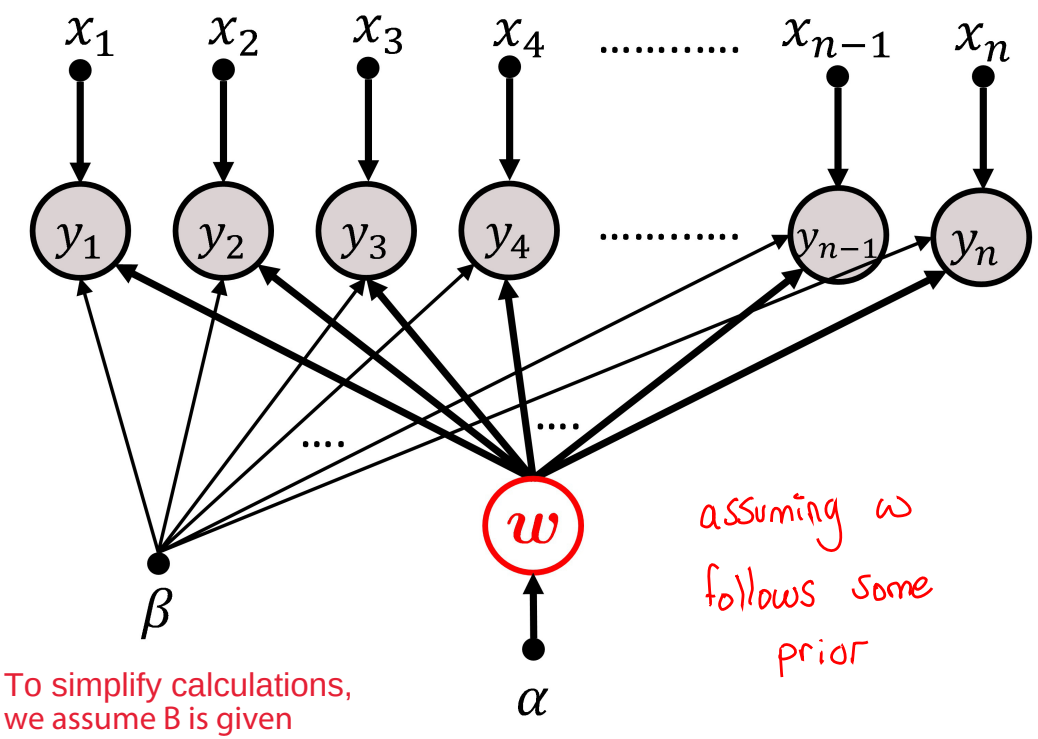
Posterior distribution

aiming for better soln

Recall from the Lecture 3, that the MLE leads to overfitting (especially, when little training data is available).
Solution - consider the posterior distribution instead

Deterministic input

target, following Gaussian



assuming w follows some prior

To simplify calculations, we assume B is given

Posterior distribution

Recall from the Lecture 3, that the MLE leads to overfitting (especially, when little training data is available).
Solution - consider the posterior distribution instead

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot) = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{w} \mid \cdot)}^{\text{prior}}}{\underbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \beta, \cdot)}_{\text{normalizing constant}}} \tag{45}$$

$$\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w} \mid \cdot) \tag{46}$$

Precision $\beta = 1/\sigma^2$ is treated as a known parameter to simplify the calculations.

Posterior distribution

Recall from the Lecture 3, that the MLE leads to overfitting (especially, when little training data is available).
Solution - consider the posterior distribution instead

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot) = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{w} \mid \cdot)}^{\text{prior}}}{\underbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \beta, \cdot)}_{\text{normalizing constant}}} \tag{45}$$

$$\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w} \mid \cdot) \tag{46}$$

Connection to the coin flip example

	train data	likelihood	prior	posterior
coin:	$\mathcal{D} = \boldsymbol{X}$	$p(\mathcal{D} \mid \theta)$	$p(\theta \mid a, b)$	$p(\theta \mid \mathcal{D})$
regr.:	$\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$	$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)$	$p(\boldsymbol{w} \mid \cdot)$	$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot)$

How do we choose the prior $p(\boldsymbol{w} \mid \cdot)$?

Prior for $\boldsymbol{w}$

* all dims are independent
* same variance along each dim
* Gaussian is spherical / circular

i.e. weights are uncorrelated

Choosing it Gaussian also

We set the prior over $\boldsymbol{w}$ to an isotropic multivariate normal distribution with zero mean

$$\mathrm{p}(\boldsymbol{w} \mid \alpha) = \mathcal{N}(\boldsymbol{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}\right) \qquad (47)$$

where,

α - precision of the distribution

M - number of elements in the vector $\boldsymbol{w}$

Motivation:

- Higher probability is assigned to small values of $\boldsymbol{w}$ $\Longrightarrow$ prevents overfitting (recall slide 20)
- Likelihood is also Gaussian - simplified calculations

✗ Our prior is that large weights cause overfitting models,
We choose zero-centered weights instead.

Maximum a posteriori (MAP)

No interest in Likelihood anymore!

We are looking for $\boldsymbol{w}$ that corresponds to the mode of the posterior

$$\boldsymbol{w}_{\text{MAP}} = \arg\max_{\boldsymbol{w}} \; p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \tag{48}$$

$$= \arg\max_{\boldsymbol{w}} \; \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) + \ln p(\boldsymbol{w} \mid \alpha) - \underbrace{\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \beta, \alpha)}_{=\text{const}} \tag{49}$$

Likelihood

Prior

Evidence

$$= \arg\min_{\boldsymbol{w}} - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \tag{50}$$

Similar to ML, define the MAP error function as negative log-posterior

$$E_{\text{MAP}}(\boldsymbol{w}) = - \ln p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \tag{51}$$

$$= - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) + \text{const} \tag{52}$$

We ignore the constant terms in the error function, as they are independent of $\boldsymbol{w}$

MAP error function

Simplify the error function

By choosing different prior, regularization shall be changed; to better match it

$$
\begin{aligned}
E_{MAP} &= -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \\
&\quad - \ln \left(\frac{\alpha}{2\pi} \right)^{\frac{M}{2}} + \frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w} \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2 + \text{ const} \\
&\propto \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2}_{\text{ridge regression error fn!}} + \text{const} \qquad \text{where } \lambda = \frac{\alpha}{\beta} \\
&= E_{\text{ridge}}(\boldsymbol{w}) + \text{const}
\end{aligned}
\tag{53}
$$

MAP estimation with Gaussian prior is equivalent to ridge regression!

Full Bayesian approach

Instead of representing $\mathrm{p}(\boldsymbol{w} \mid \mathcal{D})$ with the point estimate $\boldsymbol{w}_{\mathsf{MAP}}$, we can compute the full posterior distribution

argmax

$$\mathrm{p}(\boldsymbol{w} \mid \mathcal{D}) \propto \mathrm{p}(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)\, \mathrm{p}(\boldsymbol{w} \mid \alpha). \tag{54}$$

Since both likelihood and prior are Gaussian, the posterior is as well![4]

$$\mathrm{p}(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\mu} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{y}$ and $\boldsymbol{\Sigma}^{-1} = \alpha \boldsymbol{I} + \beta \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi}$.

Observations

- The posterior is Gaussian, so its mode is the mean and $\boldsymbol{w}_{\mathsf{MAP}} = \boldsymbol{\mu}$
- In the limit of an infinitely broad prior $\alpha \to 0$, $\boldsymbol{w}_{\mathsf{MAP}} \to \boldsymbol{w}_{\mathsf{ML}}$
- For $N = 0$, i.e. no data points, the posterior equals the prior
- Even though we assume an isotropic prior $\mathrm{p}(\boldsymbol{w})$, the posterior covariance is in general not diagonal

variance is so large, flat prior

uncorrelated weights

w-space

this is NOT the case with posterior

[4] The Gaussian distribution is a *conjugate prior* of itself

Predicting for new data: MLE and MAP

Learning phase

After observing data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, we can compute the MLE/MAP.

Usually, what we are actually interested in is the prediction $\hat{y}_{new}$ for a new data point $\boldsymbol{x}_{new}$ - the model parameters $\boldsymbol{w}$ are just a means to achieve this.

Inference phase

Recall, that we assume β to be known a priori (for simplified calculations).

Predicting for new data: MLE and MAP

After observing data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, we can compute the MLE/MAP.

Usually, what we are actually interested in is the prediction $\hat{y}_{new}$ for a new data point $\boldsymbol{x}_{new}$ - the model parameters $\boldsymbol{w}$ are just a means to achieve this.

Recall, that $y \sim \mathcal{N}(f_{\boldsymbol{w}}(\boldsymbol{x}), \beta^{-1})$

Plugging in the estimated parameters we get a predictive distribution that lets us make prediction $\hat{y}_{new}$ for new data $\boldsymbol{x}_{new}$.

- Maximum likelihood: $\boldsymbol{w}_{\mathrm{ML}}$ and β_{ML}

$$\mathrm{p}(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(\hat{y}_{new} \mid \boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta_{\mathrm{ML}}^{-1}\right) \quad (55)$$

predicted mean; fixed variance

- Maximum a posteriori: $\boldsymbol{w}_{\mathrm{MAP}}$

$$\mathrm{p}(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\mathrm{MAP}}, \beta) = \mathcal{N}\left(\hat{y}_{new} \mid \boldsymbol{w}_{\mathrm{MAP}}^T \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta^{-1}\right) \quad (56)$$

already known

Recall, that we assume β to be known a priori (for simplified calculations).

Posterior predictive distribution

Alternatively, we can use the full posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$.

This allows us to compute the posterior predictive distribution

Very common trick

$$\mathrm{p}(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \mathcal{D}) = \int \mathrm{p}(\hat{y}_{new}, \boldsymbol{w} \mid \boldsymbol{x}_{new}, \mathcal{D})\, \mathrm{d}\boldsymbol{w}$$

adding w, then marginalizing it out

$$= \int \mathrm{p}(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w})\ \mathrm{p}(\boldsymbol{w} \mid \mathcal{D})\, \mathrm{d}\boldsymbol{w}$$

no more w!

once w are known, y_new is independent on data

$$= \mathcal{N}\left(\hat{y}_{new} \mid \boldsymbol{\mu}^T \phi(\boldsymbol{x}_{new}), \beta^{-1} + \phi(\boldsymbol{x}_{new})^T \boldsymbol{\Sigma} \phi(\boldsymbol{x}_{new})\right)$$

w_MAP

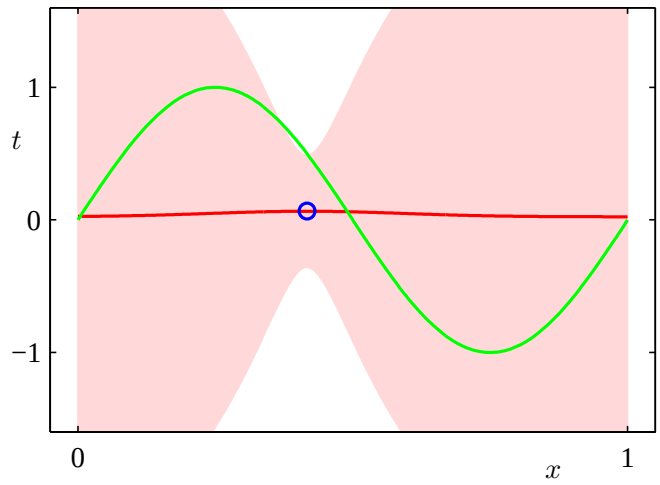
Advantage: We get a more accurate estimate about the uncertainty in the prediction (i.e. the variance of the Gaussian, which now also depends on the input $\boldsymbol{x}_{new}$)

variance here depends on new data

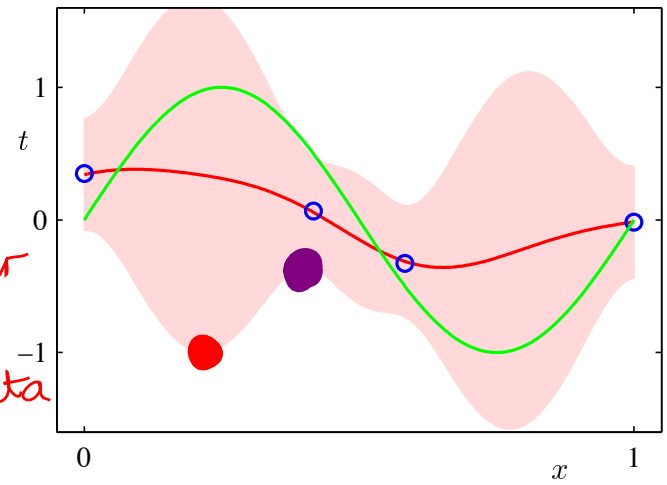
Full Bayesian gives us data-dependent uncertainty estimates.

Example of posterior predictive distribution

Uncertainity starts very wide

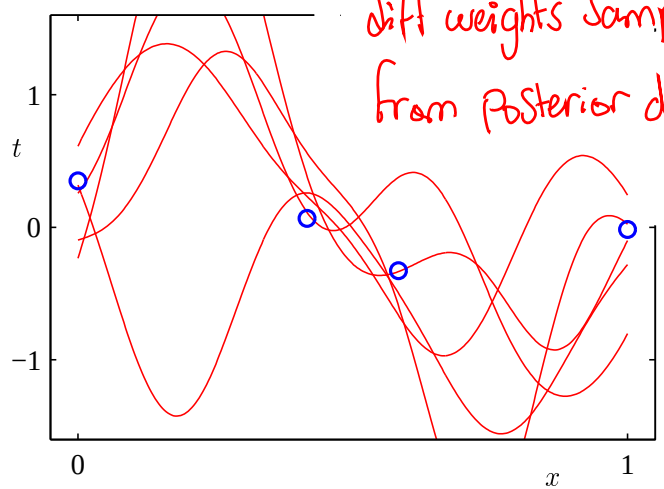


Uncertainty starts to adapt
to which data you're looking at.

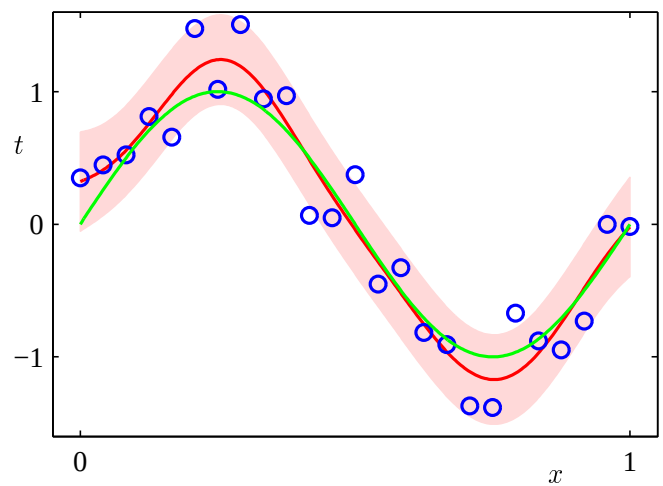


high var where few data

less var where more data



diff weights sampled from posterior dist



Green: Underlying function, Blue: Observations, Dark-Red: Mode, Light-Red: Variance

i.e mean prediction

i.e uncertainty

Summary

- Optimization-based approaches to regression have probabilistic interpretations
  - Least squares regression $\Longleftrightarrow$ Maximum likelihood (Slide 32)
  - Ridge regression $\Longleftrightarrow$ Maximum a posteriori (Slide 38)
- Even nonlinear dependencies in the data can be captured by a model linear w.r.t. weights $\boldsymbol{w}$ (Slide 13) by applying some basis transformation
- Penalizing large weights helps to reduce overfitting (Slide 20)
- Full Bayesian gives us data-dependent uncertainty estimates (Slide 41)

Reading material

Main reading

- "Pattern Recognition and Machine Learning" by Bishop
  [ch. 1.1, 3.1, 3.2, 3.3.1, 3.3.2, 3.6]

Extra reading

- "Machine Learning: A Probabilistic Perspective" by Murphy
  [ch. 7.2–7.3, 7.5.1, 7.6.1, 7.6.2]

Slides are based on an older version by G. Jensen and C. Osendorfer. Some figures are from Bishop's "Pattern Recognition and Machine Learning".