

# Machine Learning

## Lecture 4: Linear Regression

---

Prof. Dr. Stephan Günnemann

Data Analytics and Machine Learning  
Technical University of Munich

Winter term 2021/2022

# 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 $\boldsymbol{X}$

---

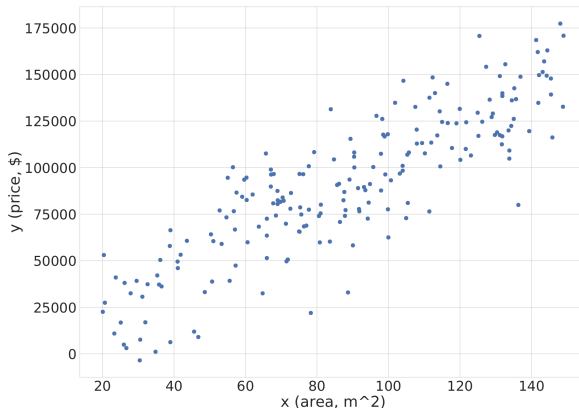
There is not a special symbol for vectors or matrices augmented by the bias term,  $w_0$ . Assume it is always included.

# 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

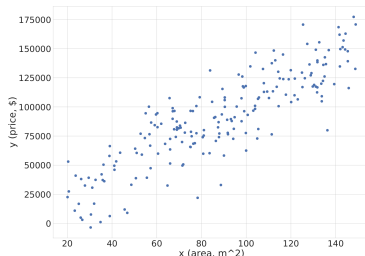
## Given

- observations <sup>1</sup>  
 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^D$
- targets  
 $\mathbf{y} = \{y_1, y_2, \dots, y_N\}, y_i \in \mathbb{R}$

## Find

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

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



---

<sup>1</sup>A common way to represent the samples is as a **data matrix**  $\mathbf{X} \in \mathbb{R}^{N \times D}$ , where each row represents one sample.

# Linear model

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

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

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

$$f_{\mathbf{w}}(\mathbf{x}_i) = w_0 + w_1 x_{i1} + w_2 x_{i2} + \dots + w_D x_{iD} \quad (2)$$

$$= w_0 + \mathbf{w}^T \mathbf{x}_i \quad (3)$$

---

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

# Absorbing the bias term

The linear function is given by

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_Dx_D \quad (4)$$

$$= w_0 + \mathbf{w}^T \mathbf{x} \quad (5)$$

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

$$\tilde{\mathbf{x}} = (1, x_1, \dots, x_D)^T \quad \tilde{\mathbf{w}} = (w_0, w_1, \dots, w_D)^T$$

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

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

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

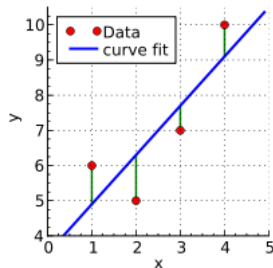
# Loss function

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

Standard choice - **least squares** (LS)

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

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



---

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



# Objective

Find the optimal weight vector  $\mathbf{w}^*$  that minimizes the error

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} E_{\text{LS}}(\mathbf{w}) \quad (8)$$

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

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

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

# Optimal solution

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

$$\nabla_{\mathbf{w}} E_{\text{LS}}(\mathbf{w}) = \nabla_{\mathbf{w}} \frac{1}{2} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) \quad (11)$$

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

$$= \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y} \quad (13)$$

---

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

# Optimal solution

Now set the gradient to zero and solve for  $w$  to obtain the minimizer <sup>2</sup>

$$X^T X w - X^T y \stackrel{!}{=} 0 \quad (14)$$

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

$$w^* = \underbrace{(X^T X)^{-1} X^T}_{=X^\dagger} y \quad (15)$$

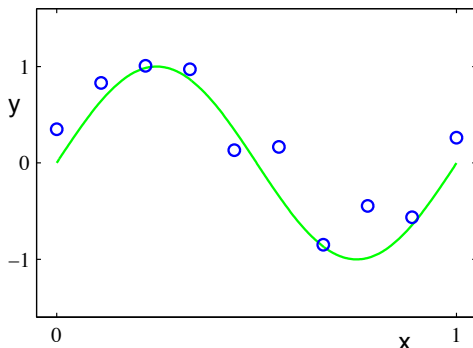
$X^\dagger$  is called **Moore-Penrose pseudo-inverse** of  $X$  (because for an invertible square matrix,  $X^\dagger = X^{-1}$ ).

---

<sup>2</sup>Because Hessian  $\nabla_w \nabla_w E(w)$  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:  $y_i = \sin(2\pi x_i) + \epsilon_i$ ,  $\epsilon_i \sim \mathcal{N}(0, \beta^{-1})$

---

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

# Polynomials

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

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

Or more generally

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

Define  $\phi_0 = 1$

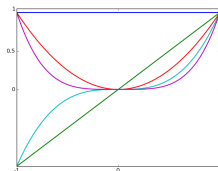
$$= \mathbf{w}^T \boldsymbol{\phi}(x) \quad (18)$$

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

# Typical basis functions

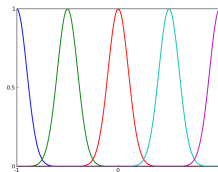
Polynomials

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



Gaussian

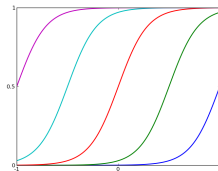
$$\phi_j(x) = e^{\frac{-(x-\mu_j)^2}{2s^2}}$$



Logistic Sigmoid

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right),$$

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



# Linear basis function model

For  $d$ -dimensional data  $\mathbf{x}$ :  $\phi_j : \mathbb{R}^d \rightarrow \mathbb{R}$

Prediction for one sample

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

Using the same least squares error function as before

$$E_{\text{LS}}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2} (\boldsymbol{\Phi} \mathbf{w} - \mathbf{y})^T (\boldsymbol{\Phi} \mathbf{w} - \mathbf{y}) \quad (20)$$

with

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

being the **design matrix** of  $\boldsymbol{\phi}$ .

# Optimal solution

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

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

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

$$E_{\text{LS}}(\mathbf{w}) = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y}). \quad (21)$$

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

$$\mathbf{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y} = \Phi^\dagger \mathbf{y} \quad (22)$$

Compare this to Equation 15:

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{X}^\dagger \mathbf{y} \quad (23)$$

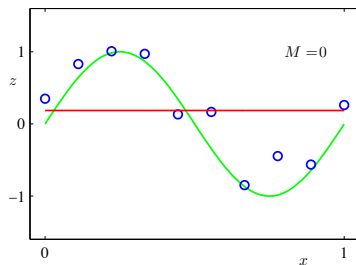


# Choosing degree of the polynomial

How do we choose the degree of the polynomial  $M$ ?

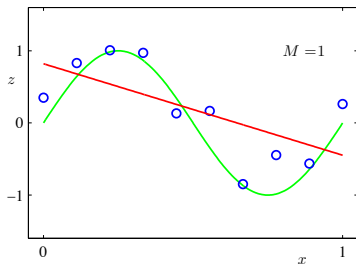
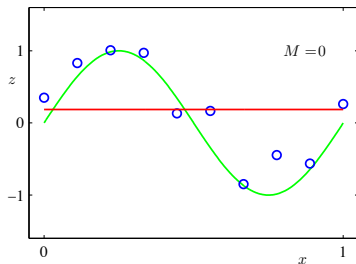
# Choosing degree of the polynomial $M$

How do we choose the degree of the polynomial  $M$ ?



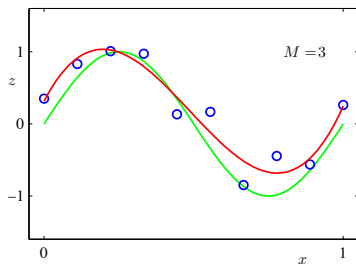
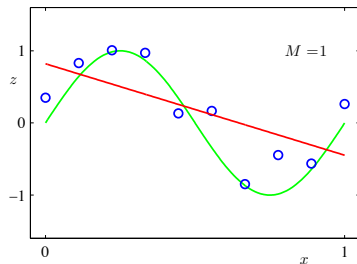
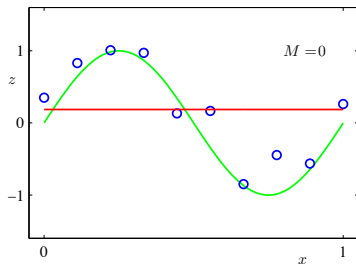
# Choosing degree of the polynomial

How do we choose the degree of the polynomial  $M$ ?



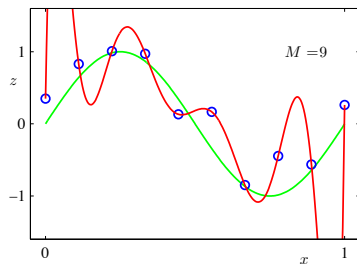
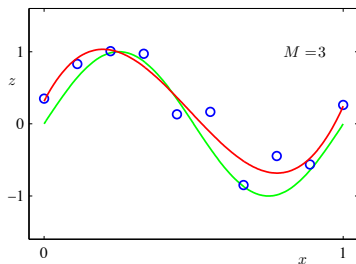
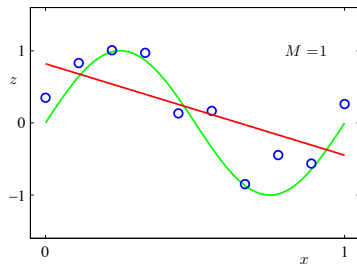
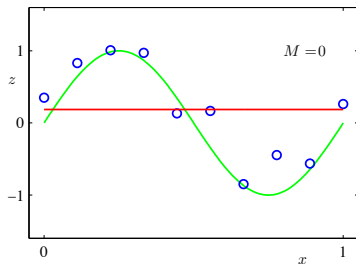
# Choosing degree of the polynomial

How do we choose the degree of the polynomial  $M$ ?

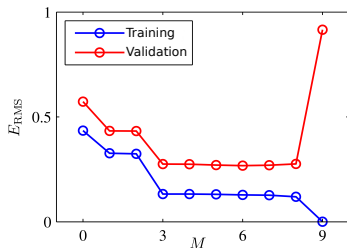
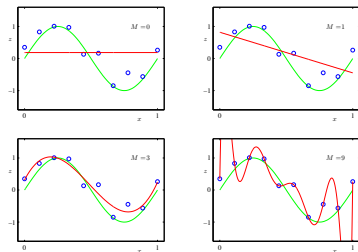


# Choosing degree of the polynomial

How do we choose the degree of the polynomial  $M$ ?

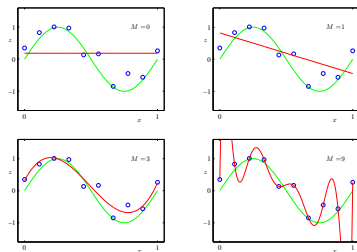


# Choosing degree of the polynomial



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

# Choosing degree of the polynomial



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

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

What if we penalize large weights?

# Controlling overfitting with regularization

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

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

where

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



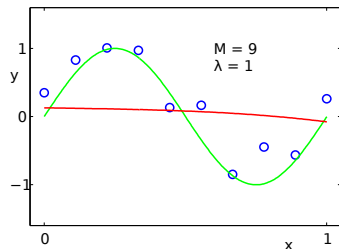
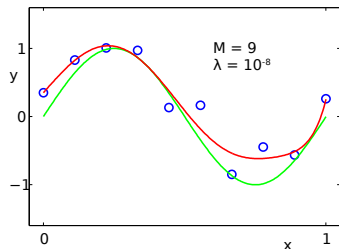
# Controlling overfitting with regularization

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

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

where

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



Larger regularization strength  $\lambda$  leads to smaller weights  $\mathbf{w}$

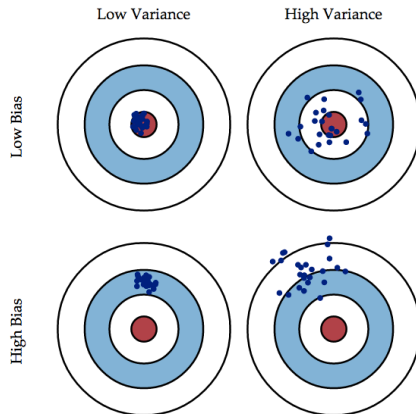
# Bias-variance tradeoff

The error of an estimator can be decomposed into two parts: <sup>3</sup>

- **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.

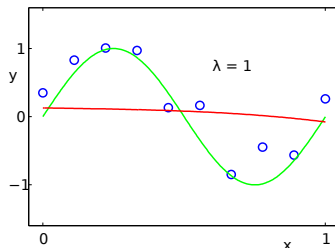
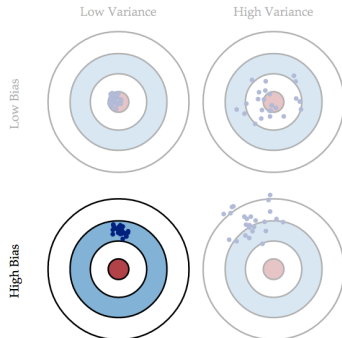
different hits (the blue dots):  
different realizations of model  
given different training data.



<sup>3</sup>See Bishop Section 3.2 for a more rigorous mathematical derivation

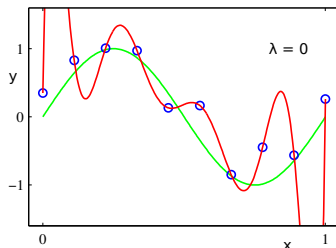
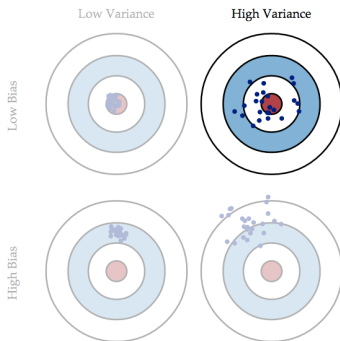
# Bias-variance tradeoff: high bias

- In case of **high bias**, the model is too rigid to fit the underlying data distribution.
- This typically happens if the model is misspecified and/or the regularization strength  $\lambda$  is too high.



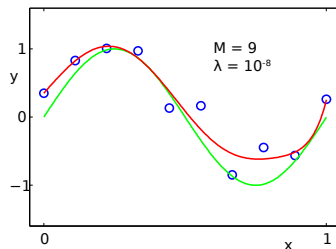
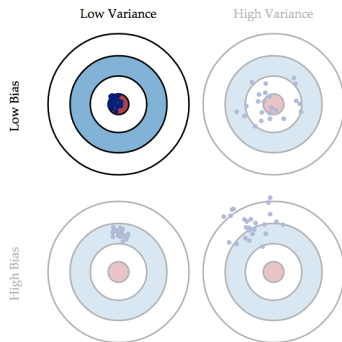
# Bias-variance tradeoff: high variance

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



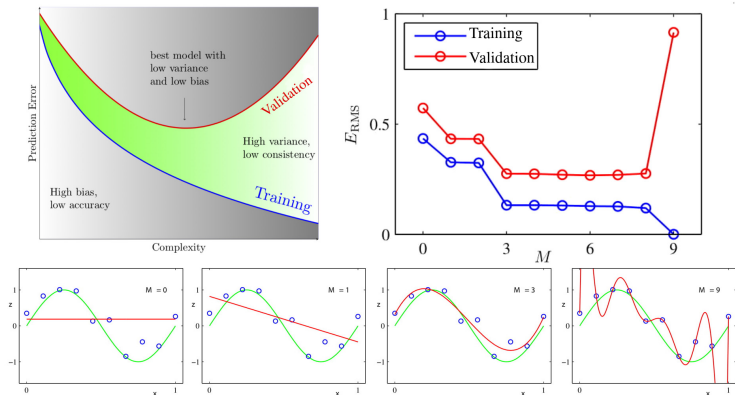
# Bias-variance tradeoff

- 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 (e.g. high degree polynomial), and keep the variance in check by choosing appropriate regularization strength  $\lambda$ .



# Bias-variance tradeoff

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

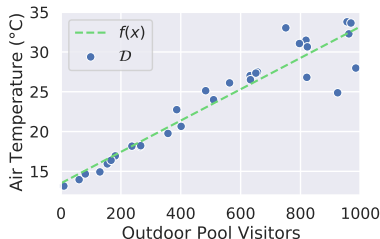


The upper-left figure from: <https://eissanematollahi.com/wp-content/uploads/2018/09/Machine-Learning-Basics-1.pdf>.

# Correlation vs. Causation

Least squares fit

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



- The weights  $w_i$  can be interpreted as the strength of the (linear) relationship between feature  $x_i$  and  $y$
- 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  $\mathbf{X}$  and  $\mathbf{y}$  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

---

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,

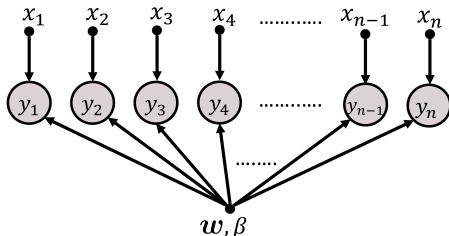
$$y_i = f_{\mathbf{w}}(\mathbf{x}_i) + \underbrace{\epsilon_i}_{\text{noise}}$$

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

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



---

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

# Maximum likelihood

Likelihood of a single sample

$$p(y_i | f_{\mathbf{w}}(\mathbf{x}_i), \beta) = \mathcal{N}(y_i | f_{\mathbf{w}}(\mathbf{x}_i), \beta^{-1}) \quad (25)$$

Assume that the samples are drawn independently

$\implies$  likelihood of the entire dataset  $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$  is

$$p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^N p(y_i | f_{\mathbf{w}}(\mathbf{x}_i), \beta) \quad (26)$$

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

$$\mathbf{w}_{\text{ML}}, \beta_{\text{ML}} = \arg \max_{\mathbf{w}, \beta} p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) \quad (27)$$

# Maximum likelihood

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

$$\mathbf{w}_{\text{ML}}, \beta_{\text{ML}} = \arg \max_{\mathbf{w}, \beta} p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \beta) \quad (28)$$

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

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

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

$$E_{\text{ML}}(\mathbf{w}, \beta) = -\ln p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \beta) \quad (31)$$

# Maximum likelihood

Simplify the error function

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

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

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

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

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

$$\mathbf{w}_{\text{ML}} = \arg \min_{\mathbf{w}} E_{\text{ML}}(\mathbf{w}, \beta) \quad (36)$$

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

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

$$= \arg \min_{\mathbf{w}} E_{\text{LS}}(\mathbf{w}) \quad (39)$$

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

$$\mathbf{w}_{\text{ML}} = \arg \min_{\mathbf{w}} E_{\text{ML}}(\mathbf{w}, \beta) \quad (36)$$

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

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

$$= \arg \min_{\mathbf{w}} E_{\text{LS}}(\mathbf{w}) \quad (39)$$

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

$$\mathbf{w}_{\text{ML}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y} = \Phi^\dagger \mathbf{y} \quad (40)$$

# Optimizing log-likelihood w.r.t. $\beta$

Plug in the estimate for  $\mathbf{w}$  and minimize w.r.t.  $\beta$

$$\beta_{\text{ML}} = \arg \min_{\beta} E_{\text{ML}}(\mathbf{w}_{\text{ML}}, \beta) \quad (41)$$

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

Take derivative w.r.t.  $\beta$  and set it to zero

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

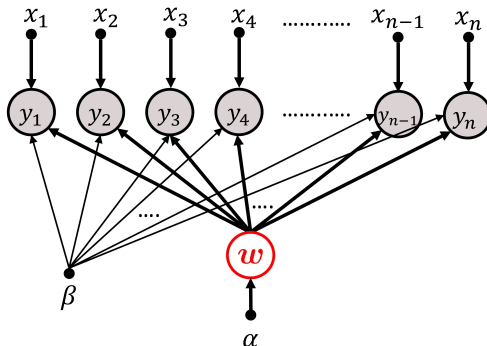
Solving for  $\beta$

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

# 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





# 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

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

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

---

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

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

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

Connection to the coin flip example

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

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

---

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

## Prior for $\mathbf{w}$

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

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

where,

$\alpha$  - precision of the distribution

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

Motivation:

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

# Maximum a posteriori (MAP)

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

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

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

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

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

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

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

---

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

# MAP error function

Simplify the error function

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

MAP estimation with Gaussian prior is equivalent to ridge regression!

# Full Bayesian approach

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

$$p(\mathbf{w} \mid \mathcal{D}) \propto p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \beta) p(\mathbf{w} \mid \alpha). \quad (54)$$

Since both likelihood and prior are Gaussian, the posterior is as well!<sup>4</sup>

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

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

## Observations

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

---

<sup>4</sup>The Gaussian distribution is a *conjugate prior* of itself

# Predicting for new data: MLE and MAP

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

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

---

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

# Predicting for new data: MLE and MAP

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

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

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

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

- Maximum likelihood:  $\mathbf{w}_{\text{ML}}$  and  $\beta_{\text{ML}}$

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

- Maximum a posteriori:  $\mathbf{w}_{\text{MAP}}$

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

---

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



# Posterior predictive distribution

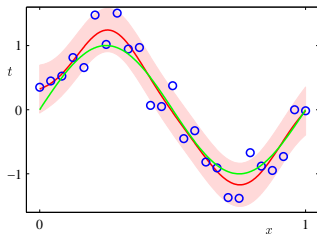
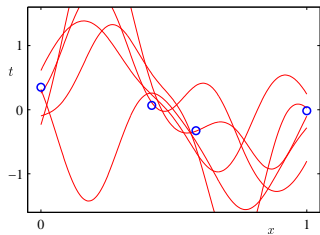
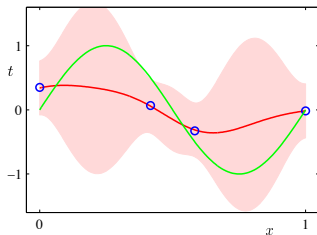
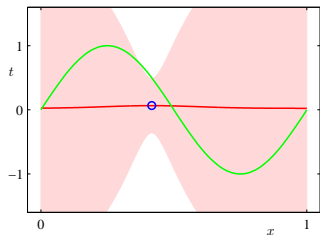
Alternatively, we can use the full posterior distribution  $p(\mathbf{w} \mid \mathcal{D})$ .

This allows us to compute the **posterior predictive distribution**

$$\begin{aligned} p(\hat{y}_{new} \mid \mathbf{x}_{new}, \mathcal{D}) &= \int p(\hat{y}_{new}, \mathbf{w} \mid \mathbf{x}_{new}, \mathcal{D}) d\mathbf{w} \\ &= \int p(\hat{y}_{new} \mid \mathbf{x}_{new}, \mathbf{w}) p(\mathbf{w} \mid \mathcal{D}) d\mathbf{w} \\ &= \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{\mu}^T \phi(\mathbf{x}_{new}), \beta^{-1} + \phi(\mathbf{x}_{new})^T \boldsymbol{\Sigma} \phi(\mathbf{x}_{new})) \end{aligned}$$

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  $\mathbf{x}_{new}$ )

# Example of posterior predictive distribution



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

# Summary

- Optimization-based approaches to regression have probabilistic interpretations
  - Least squares regression  $\iff$  Maximum likelihood (Slide 32)
  - Ridge regression  $\iff$  Maximum a posteriori (Slide 38)
- Even nonlinear dependencies in the data can be captured by a model linear w.r.t. weights  $w$  (Slide 13)
- 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”.