# Machine Learning

## Lecture 4: Linear Regression

Prof. Dr. Stephan Günnemann

Data Analytics and Machine Learning
Technical University of Munich

Winter term 2023/2024

# 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 $\boldsymbol{X}$ |

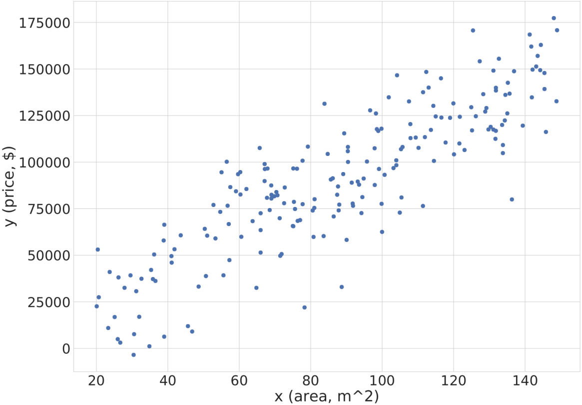
There is not a special symbol for vectors or matrices augmented by the bias term, $w_0$. Assume it is always included.

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 $y_i$.

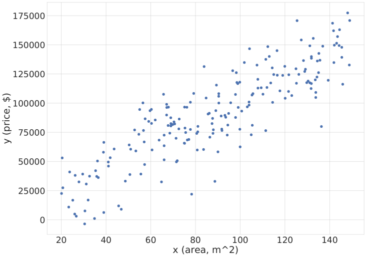



How do we estimate a price of a new house with area $x_{new}$?

# Regression problem

## Given

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




## Find

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

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

[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

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

Noise follows a noram distribution.

The data is not perfect, we need the noise

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

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

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

If we have d inputs, we set to all of them a weight. xi is the vector with all the weights.

Xd are the dimensions!!

wo is teh buas

# Absorbing the bias term

The linear function is given by

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_D x_D \quad (4)$$
$$= w_0 + \boldsymbol{w}^T \boldsymbol{x} \quad (5)$$

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

adding the vias term at beginnihn and wo, faster

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

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 $\tilde{\boldsymbol{x}}$.

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

# Loss function

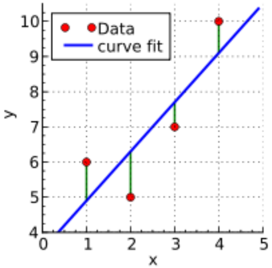
Now, how do we choose the "best" $\boldsymbol{w}$ that fits our data?

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)

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

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




Loss function, Least squares to reduce the result

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

# Objective

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

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

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

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

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

need to know

matrix is N and D.  N instances and D are the dimentsion

r

w is D an 1 dimebnsionality

y is like N*1

# Optimal solution

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

$$
\begin{aligned}
\nabla_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) &= \nabla_{\boldsymbol{w}} \frac{1}{2} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) && (11) \\
&= \nabla_{\boldsymbol{w}} \frac{1}{2} \left( \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - 2 \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y} \right) && (12) \\
&= \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y} && (13)
\end{aligned}
$$

Set the gradient to 0.

1

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

# Optimal solution

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

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

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

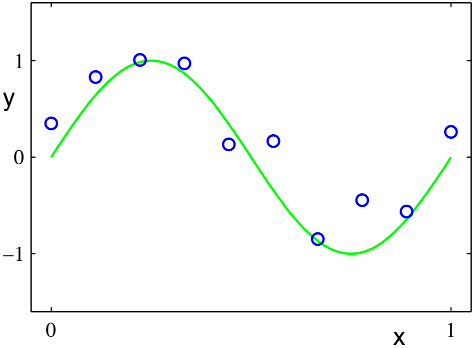
$$\boldsymbol{w}^* = \underbrace{(\boldsymbol{X}^\mathsf{T} \boldsymbol{X})^{-1} \boldsymbol{X}^\mathsf{T}}_{=\boldsymbol{X}^\dagger} \boldsymbol{y} \tag{15}$$

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

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

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

Setting polynomial, this example is mor 1 dimesnitons
M is the degree of the polynomial

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

Or more generally

Basis transformation

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

Sam trick, setting first as 1, and we can add the bias. and the other thing is a function of x

Define $\phi_0 = 1$

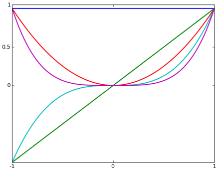
$$= \boldsymbol{w}^T \boldsymbol{\phi}(x) \tag{18}$$

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

# Typical basis functions

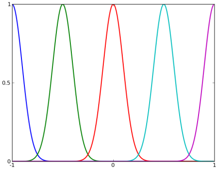
Polynomials

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




Gaussian

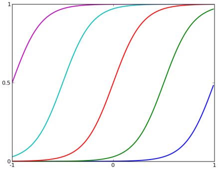
$$\phi_j(x) = \mathrm{e}^{\frac{-(x-\mu_j)^2}{2s^2}}$$




Logistic Sigmoid

$$\phi_j(x) = \sigma(\tfrac{x-\mu_j}{s}),$$

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



# Linear basis function model

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

For d-dimensional data. All teh dimanesion return a numner. So I wheight

Prediction for one sample

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

Using the same least squares error function as before

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

with

Each crows

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

Here, apply all trasnformation at the same time. Each column is a function. for instance x^1, x^2, x^3....

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 $\boldsymbol{X}$

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

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

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

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:

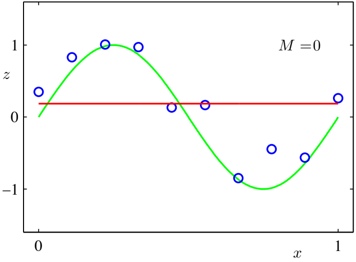
$$\boldsymbol{w}^* = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} = \boldsymbol{X}^\dagger\boldsymbol{y} \tag{23}$$

# Choosing degree of the polynomial

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

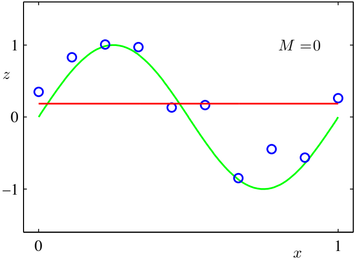
# Choosing degree of the polynomial

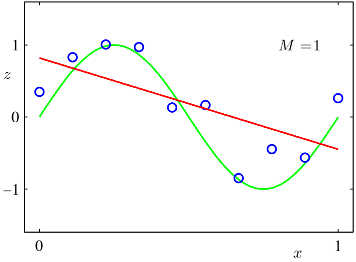
How do we choose the degree of the polynomial $M$?



# Choosing degree of the polynomial

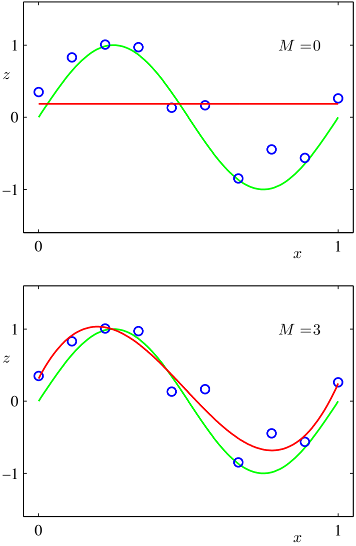
How do we choose the degree of the polynomial $M$?



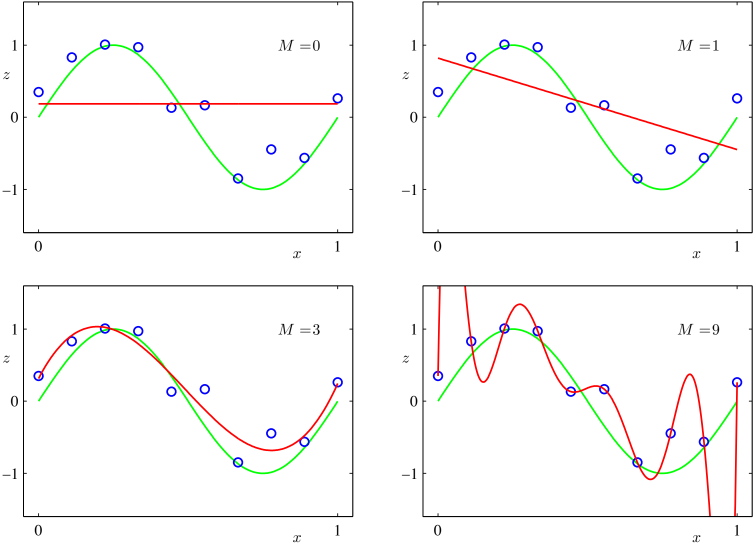
# Choosing degree of the polynomial

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

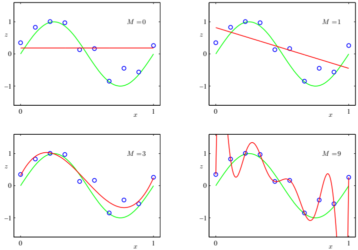


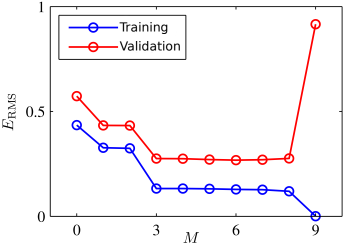
# Choosing degree of the polynomial

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



# Choosing degree of the polynomial

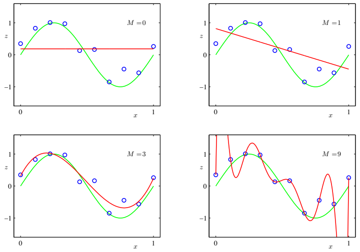



One valid solution is to choose $M$ using the standard train-validation split approach.

trainign gets to 0 because it mactehss. But with validation nope.

# Choosing degree of the polynomial




| | $M=0$ | $M=1$ | $M=3$ | $M=9$ |
|---|---|---|---|---|
| $w_0^\star$ | 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 |

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)

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

Penalization of large weights of w

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 $\boldsymbol{w}$
- $\lambda$ - regularization strength

Square sum of everityihn

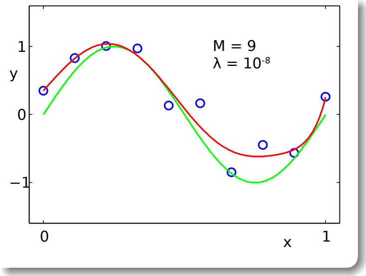
# Controlling overfitting with regularization

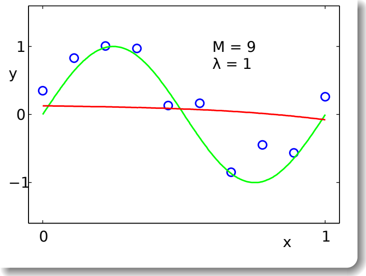
Least squares loss with L2 regularization (also called ridge regression)

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

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 $\boldsymbol{w}$
- $\lambda$ - regularization strength How strenght. Big strenmg s all weights



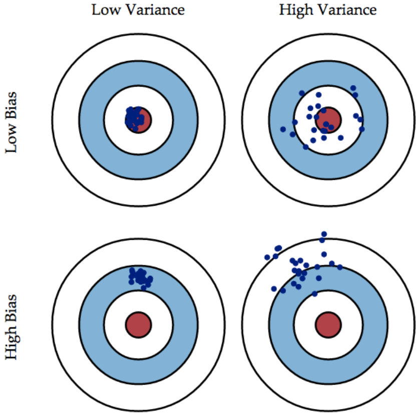




Larger regularization strength $\lambda$ leads to smaller weights $\boldsymbol{w}$

# 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.

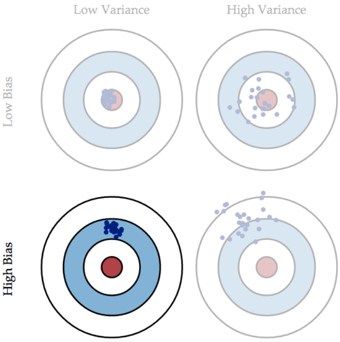
different hits (the blue dots):
different realizations of model
given different training data.

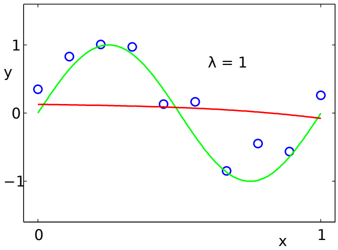
[3] See Bishop Section 3.2 for a more rigorous mathematical derivation

# Bias-variance tradeoff: high bias

- 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.

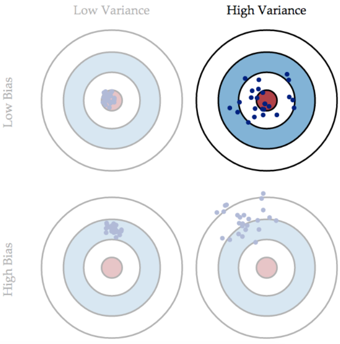
too high, no importance ot the data error. w super small



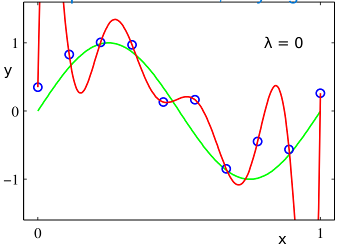



# Bias-variance tradeoff: high variance

- 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.

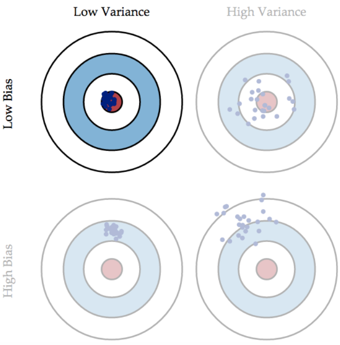


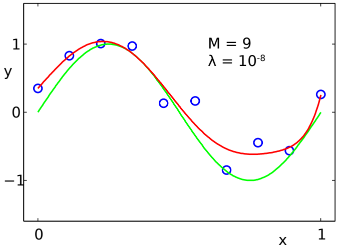

Too much importance to error, very high avraince



# 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 $\lambda$.



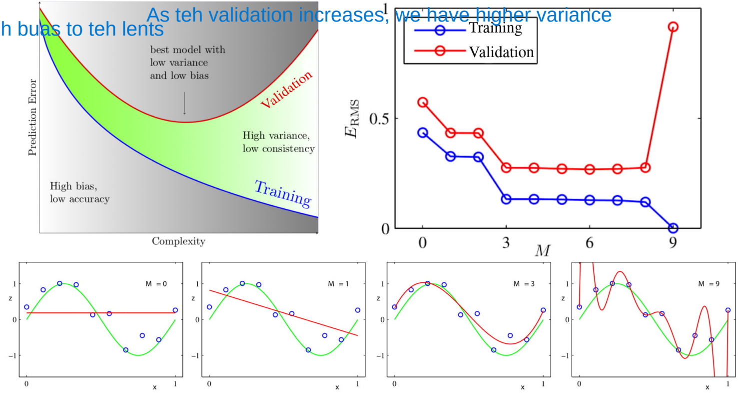



# Bias-variance tradeoff

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

High buas to teh lents

As teh validation increases, we have higher variance



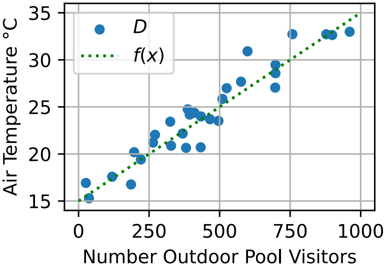

The upper-left figure from: https://eissanematollahi.com/wp-content/uploads/2018/09/Machine-Learning-Basics-1.pdf.

# Correlation

Least squares fit

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

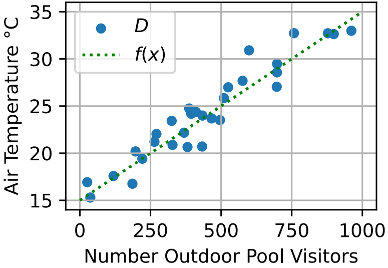
Correlation does not mean causality.
Hgth weight to a feature means taht
is more important

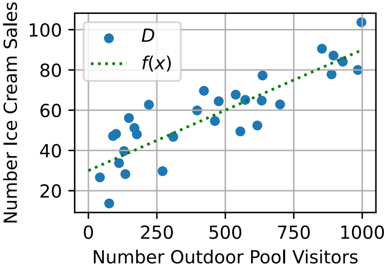



- The weights $\boldsymbol{w}_i$ can be interpreted as the strength of the (linear) relationship between feature $\boldsymbol{x}_i$ and $\boldsymbol{y}$
- A weight of $0.018$ shows a strong correlation (considering the different scales)
  - With actual data, you would normalize the data to handle the different scales of $\boldsymbol{X}$ and $\boldsymbol{y}$ and find a weight of about $1$

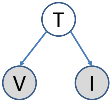
# Correlation vs. Causation

Correlation does not imply causation! Putting more people in the pool does not increase the air temperature.




Be aware of Confounding Variables!



# 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,

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

Here we had the error and added the noise distribution. NOw we make a model with teh noise. We just make teh sum.

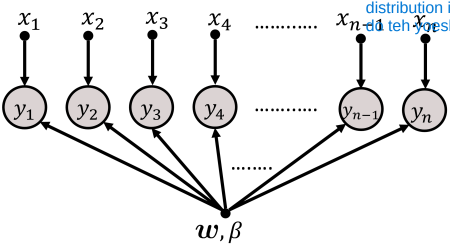
Noise has zero-mean Gaussian distribution with a fixed precision $\beta = \frac{1}{\sigma^2}$

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

This implies that the distribution of the targets is

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

Normal dsitribution with mean our function and a precision. Parameters of the distribution is a predicted value, and eth we do teh yoesl




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

# Maximum likelihood

Likelihood of a single sample

Probabilty of getting yi given our model, is teh likelihood that finding yi given the gaussian

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

Assume that the samples are drawn independently
$\implies$ likelihood of the entire dataset 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 $\beta$

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

We create f(x) with the data and teh weights

# Maximum likelihood

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

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

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

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

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

Simplify the error function

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

Mean

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

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

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

N is number of data points

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

$$
\begin{aligned}
\boldsymbol{w}_{\mathrm{ML}} &= \arg\min_{\boldsymbol{w}} E_{\mathrm{ML}}(\boldsymbol{w}, \beta) && (36)\\
&= \arg\min_{\boldsymbol{w}} \left[ \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] && (37)\\
&= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} && (38)\\
&= \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) && (39)
\end{aligned}
$$

Least squares fit

# 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!

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

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

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

Beta tells us some more poitns the variance

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

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

For any beta we get the same. Does not matter the beta

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

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

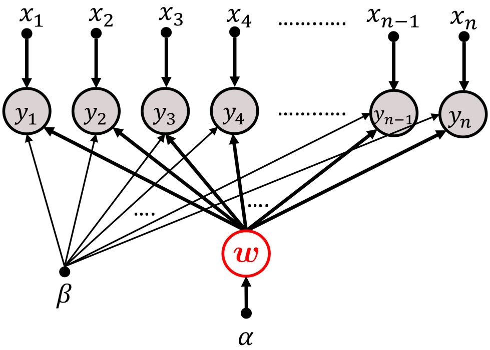
Solving for $\beta$

We get the same solution

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

# 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

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

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

Given our data, predictions and beta, waht is the porbability that w is that one. Ans we want teh max w.

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: | $\mathcal{D} = \boldsymbol{X}$ | $p(\boldsymbol{X} \mid \theta)$ | $p(\theta \mid a, b)$ | $p(\theta \mid \boldsymbol{X})$ |
| 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 $p(\boldsymbol{w} \mid \cdot)$?

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

# Prior for $\boldsymbol{w}$

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

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

where,

Mean is 0, beacuse is the hiogh withs are bad

$\alpha$ - precision of the distribution

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

Motivation:

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

# Maximum a posteriori (MAP)

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

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

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

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

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

CAll it error beacause we can, beacuse we fucking can

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

---

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

# MAP error function

Simplify the error function

$$
\begin{aligned}
E_{MAP}(\boldsymbol{w}) = & -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \\
&\quad - \frac{M}{2} \ln \left(\frac{\alpha}{2\pi}\right) + \frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w} \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2 + \text{ const} \\
&\propto \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2}_{\text{ridge regression error fn!}} + \text{const} \qquad \text{where } \lambda = \frac{\alpha}{\beta} \\
&\propto E_{\text{ridge}}(\boldsymbol{w}) + \text{const}
\end{aligned}
\tag{52}
$$

With map, we get tha MAP

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

$$\mathrm{p}(\boldsymbol{w} \mid \mathcal{D}) \propto \mathrm{p}(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)\ \mathrm{p}(\boldsymbol{w} \mid \alpha). \tag{53}$$

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

Full bayesian crazy

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

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

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

# 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 we assume $\beta$ 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 $\beta_{\mathrm{ML}}$

$$\mathrm{p}(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(\hat{y}_{new} \mid \boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta_{\mathrm{ML}}^{-1}\right) \quad (54)$$

- Maximum a posteriori: $\boldsymbol{w}_{\mathrm{MAP}}$ In Map, B-1 we do not get it

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

---

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

# Posterior predictive distribution

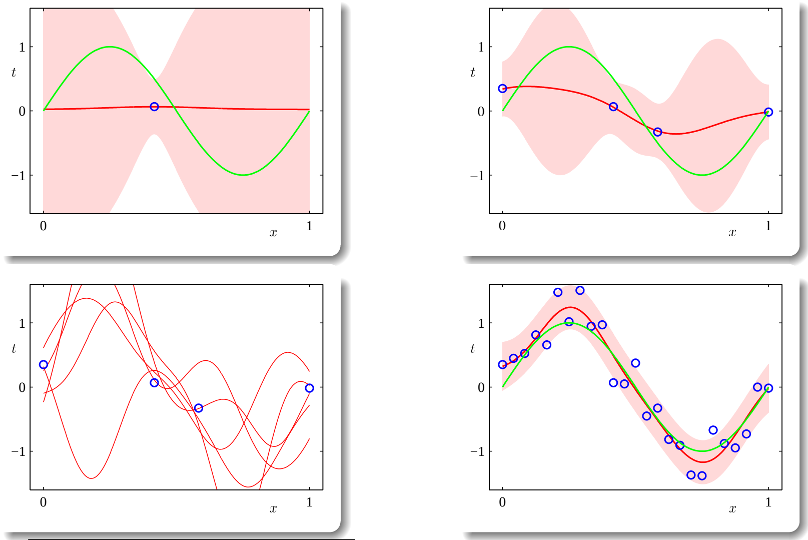
Alternatively, we can use the full posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$.

This allows us to compute the posterior predictive distribution

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

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

# 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 33)
  - Ridge regression $\iff$ Maximum a posteriori (Slide 39)
- Even nonlinear dependencies in the data can be captured by a model linear w.r.t. weights $\boldsymbol{w}$ (Slide 13)
- Penalizing large weights helps to reduce overfitting (Slide 20)
- Full Bayesian gives us data-dependent uncertainty estimates (Slide 42)

# 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".