Machine Learning

Lecture 3: Linear Regression

Prof. Dr. Stephan Günnemann

Data Mining and Analytics
Technical University of Munich

06.11.2018

Reading material

Reading material

- Bishop [ch. 1.1, 3.1, 3.2, 3.3.1, 3.3.2, 3.6]
- Murphy [ch. 7.2 - 7.3, 7.5.1, 7.6.1, 7.6.2]

Acknowledgements

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

Notation

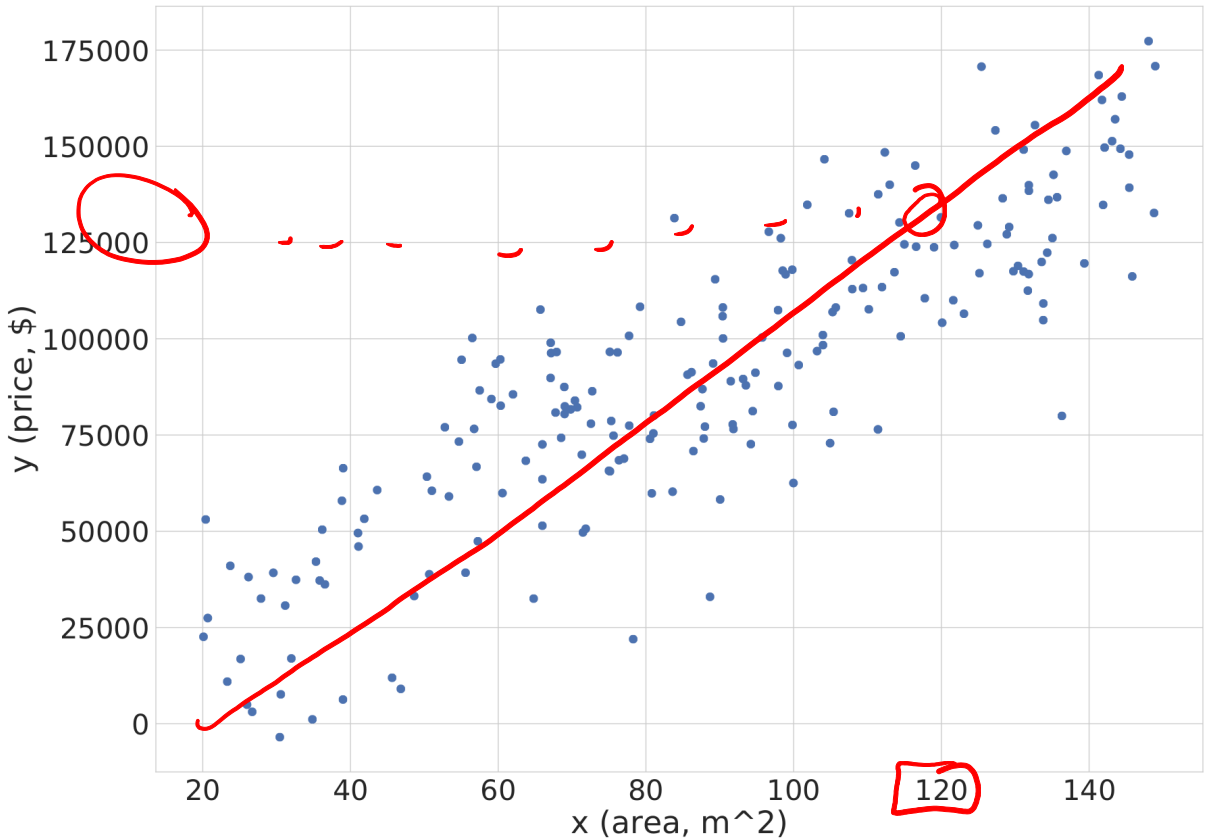
Symbol	Meaning
s	scalar is lowercase and not bold
$\boldsymbol{s}$	vector is lowercase and bold
$\boldsymbol{S}$	matrix is uppercase and bold
$f(\boldsymbol{x})$	predicted value for inputs $\boldsymbol{x}$
$\boldsymbol{y}$	vector of targets
y_i	target of the i'th example
w_0	bias term (not to be confused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

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.

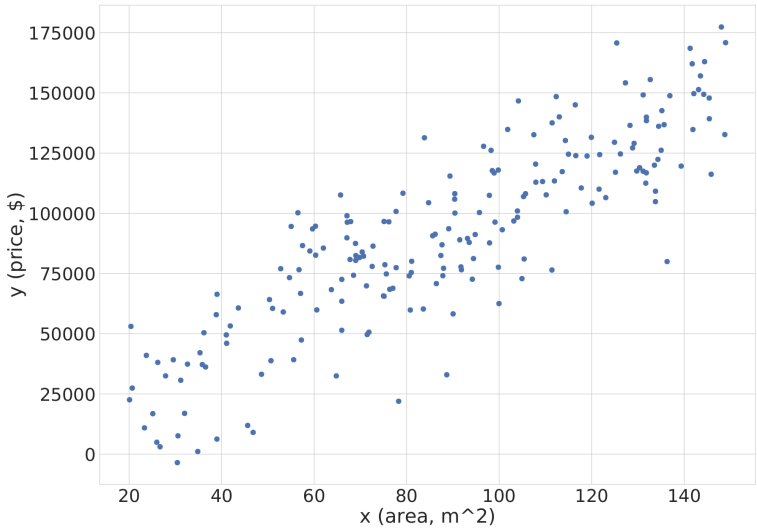


How do we estimate a price of a new house with area x_{new}?

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
  $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}, \quad y_i \in \mathbb{R}$



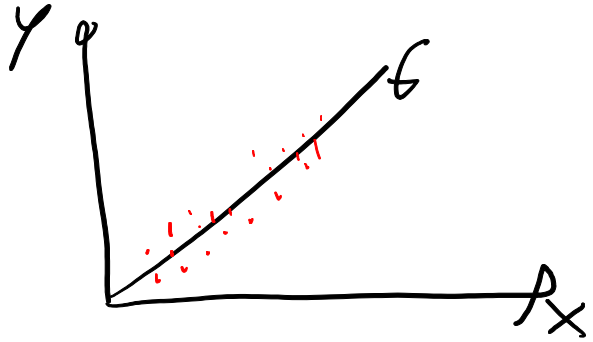
Find

- 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

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

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

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

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

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 $\boldsymbol{w}$:

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

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?

Error function

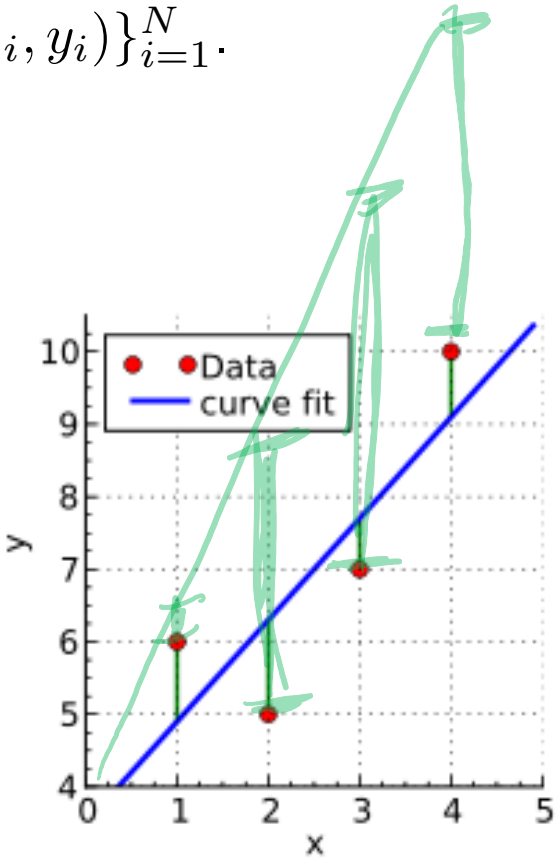
Loss function

Error function gives a measure of "misfit" 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) function

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

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



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 $\boldsymbol{X} \in \mathbb{R}^{N \times D}$

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

Optimal solution

Matrix cookbook

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

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

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

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

Optimal solution

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

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

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

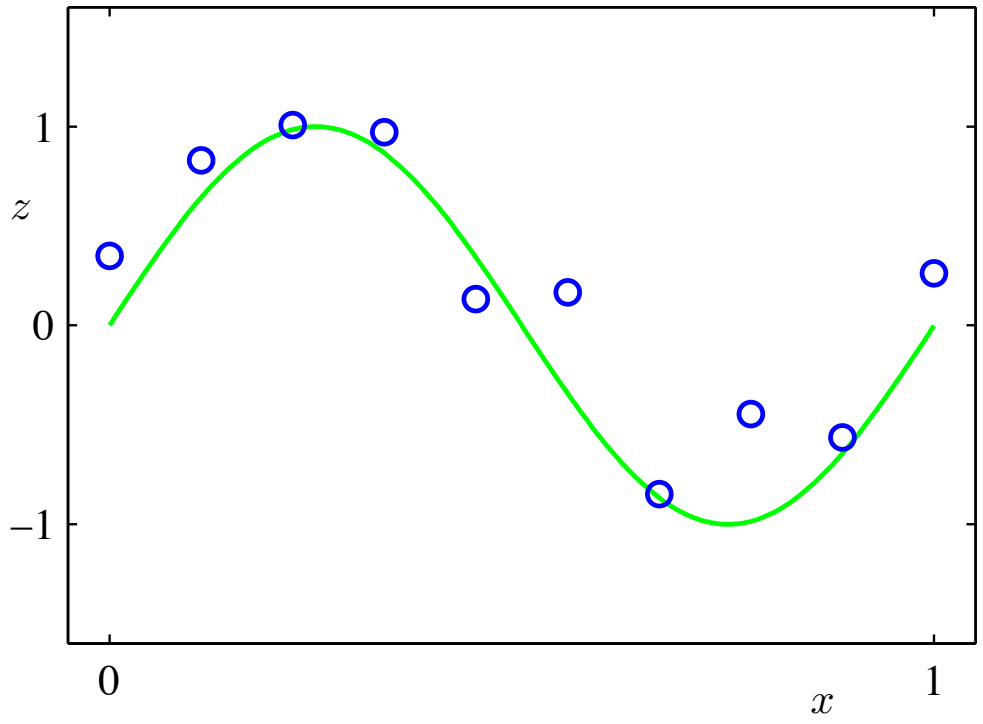
$$\boldsymbol{w}^* = \underbrace{(\boldsymbol{X}^\mathsf{T} \boldsymbol{X})^{-1} \boldsymbol{X}^\mathsf{T}}_{=\boldsymbol{X}^\dagger} \boldsymbol{y} \tag{15}$$

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

$w_1 = 1$

Polynomials

ATTENTION: HERE 1-D ON THIS SLIDE AND NEXT ONE

Solution: Polynomials are universal function approximators, so we can define f as

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

Or more generally

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

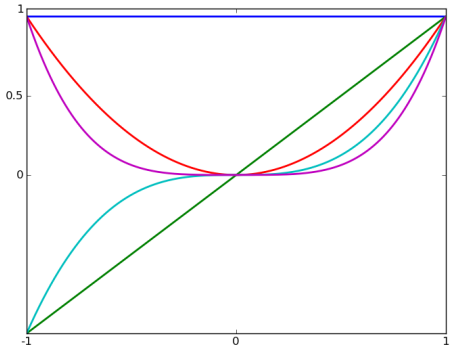
Define $\phi_0 = 1$

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

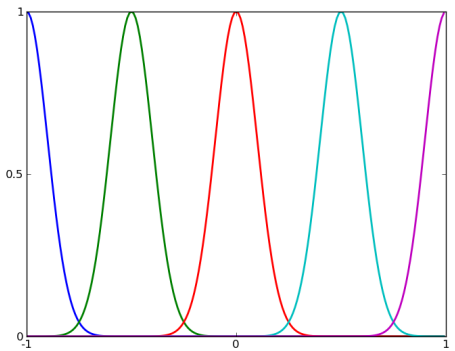
The function f is still linear in $\boldsymbol{w}$ (despite not being linear in $\boldsymbol{x}$)!

Typical basis functions

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

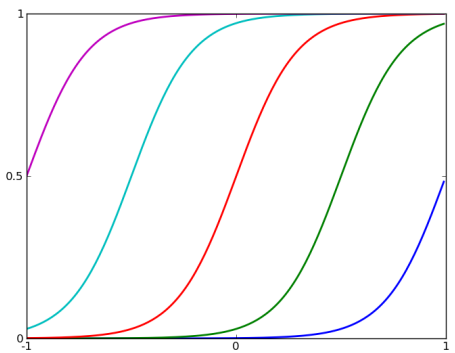


Gaussian $\phi_j(x) = \mathrm{e}^{\frac{-(x-\mu_j)^2}{2s^2}}$



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

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



Linear basis function model

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

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

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

with

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

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

Optimal solution

Recall Equation 10 - we have the same expression except that data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is replaced by design matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N \times (M+1)}$

$$E_{\mathrm{LS}}(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}) \tag{22}$$

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

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

$$= \boldsymbol{\Phi}^\dagger \boldsymbol{y} \tag{24}$$

Compare this to Equation 15:

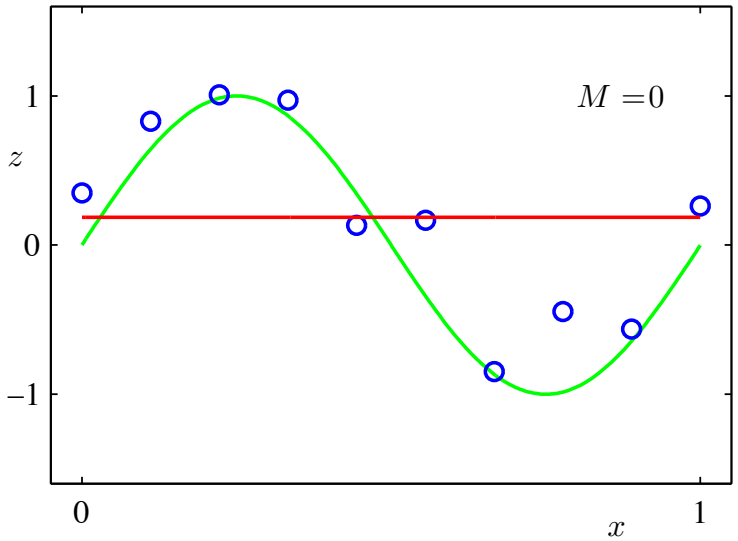
$$\boldsymbol{w}^* = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} \tag{25}$$

Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

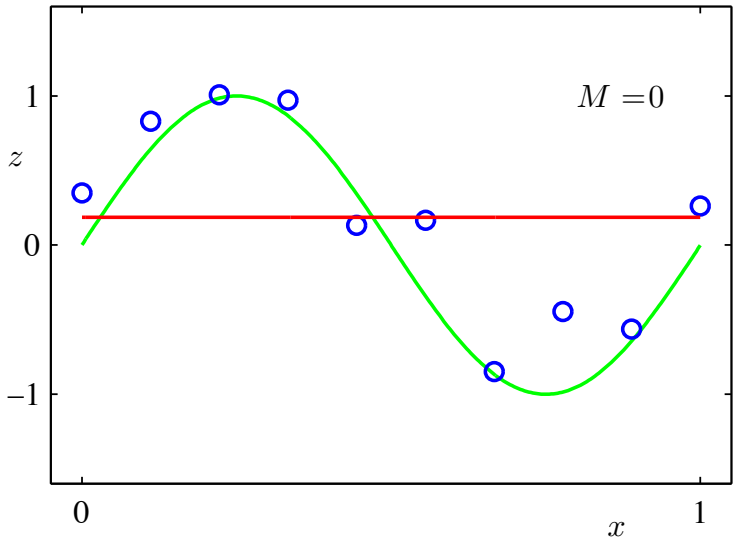
Choosing degree of the polynomial

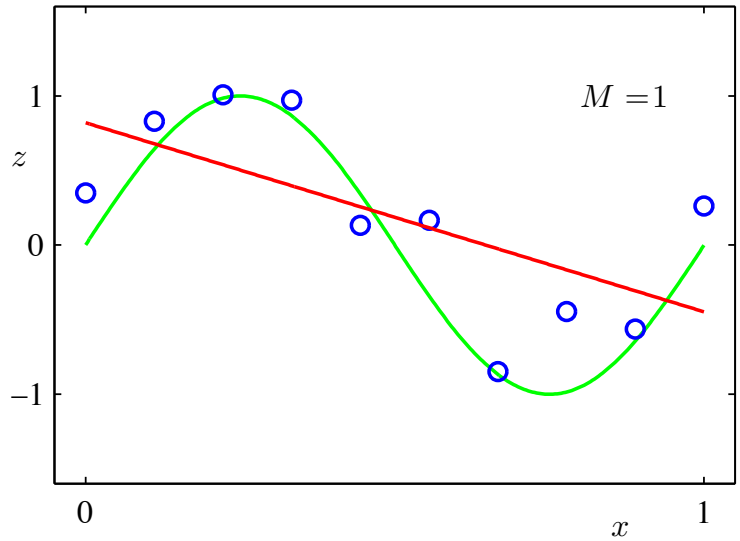
How do we choose the degree of the polynomial M?



Choosing degree of the polynomial

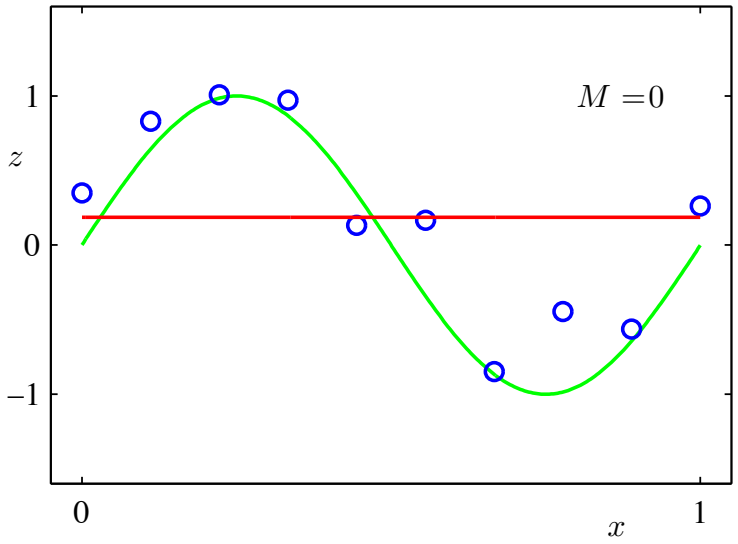
How do we choose the degree of the polynomial M?

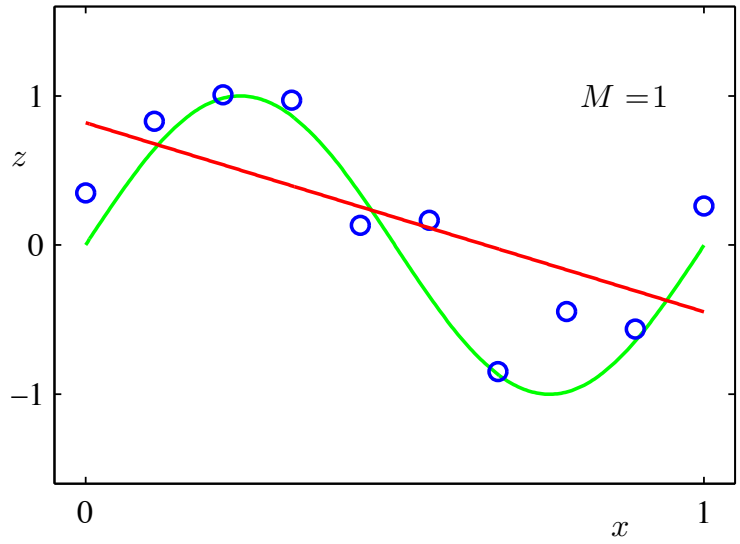


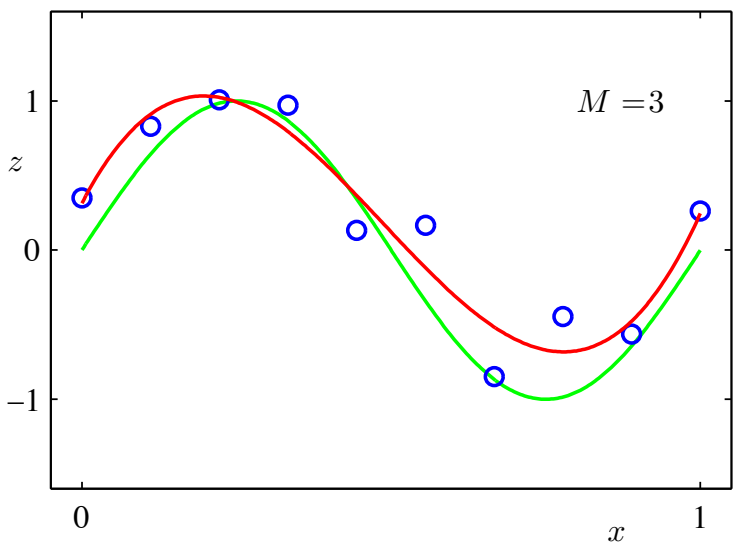


Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

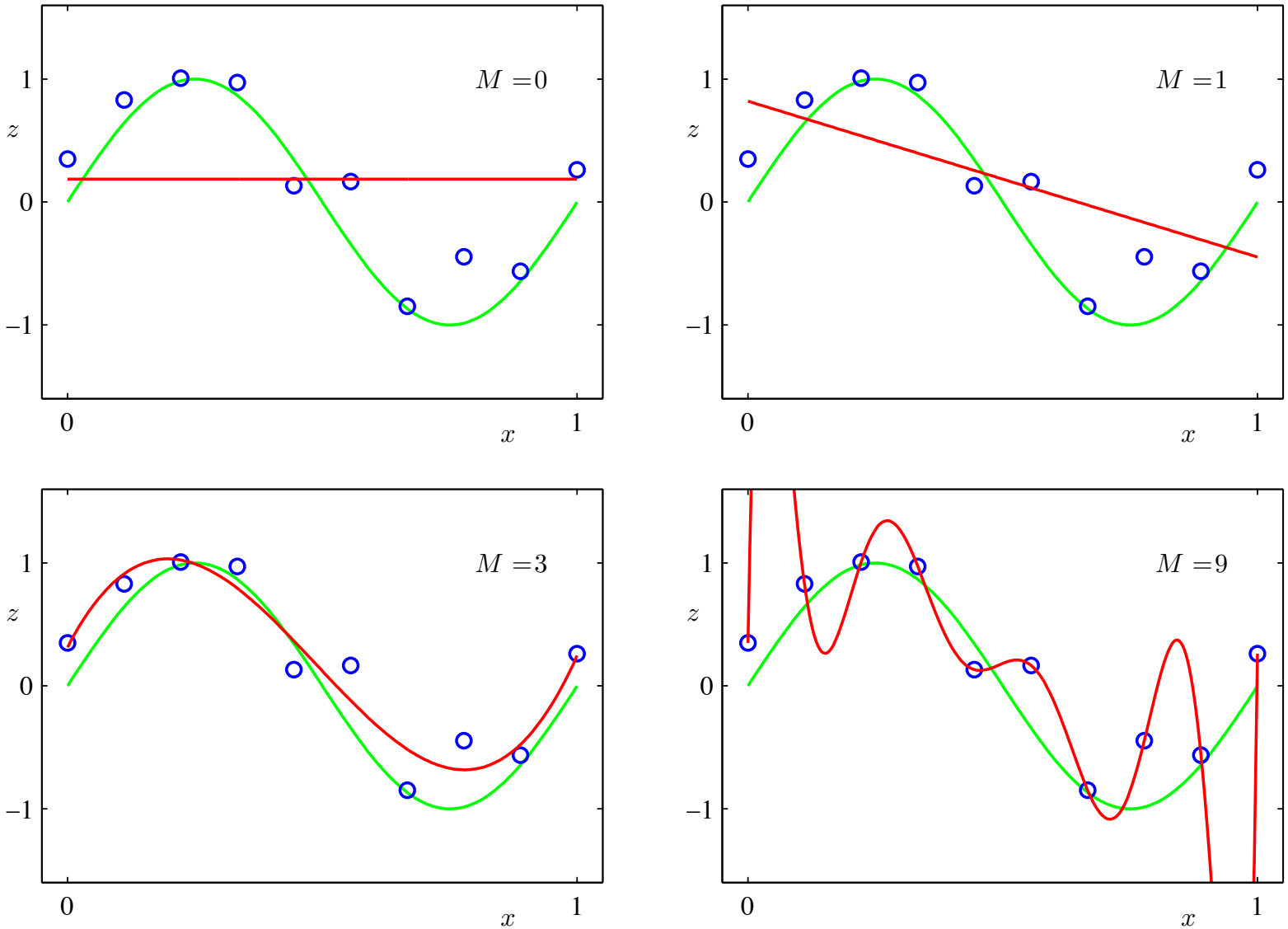




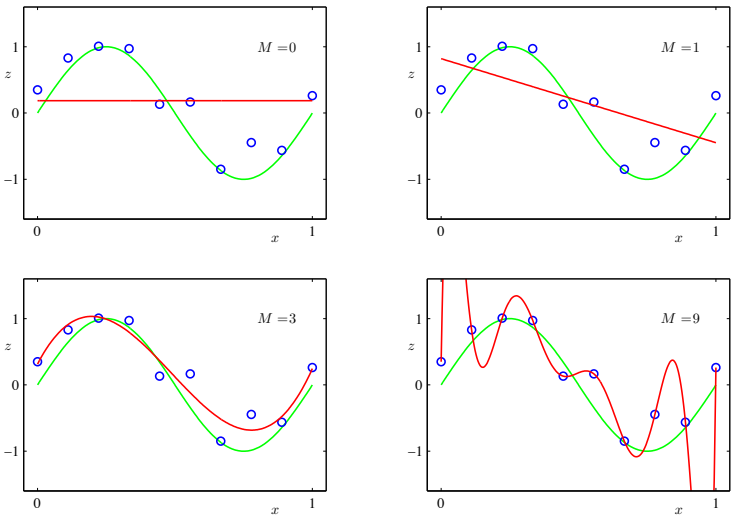


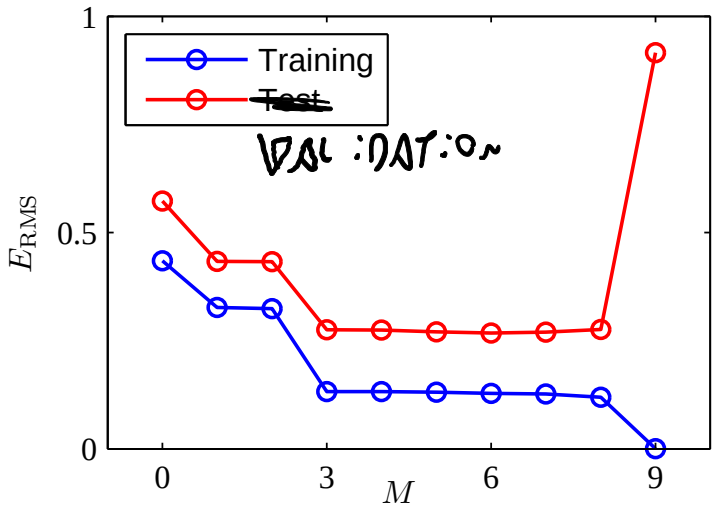
Choosing degree of the polynomial

How do we choose the degree of the polynomial M?



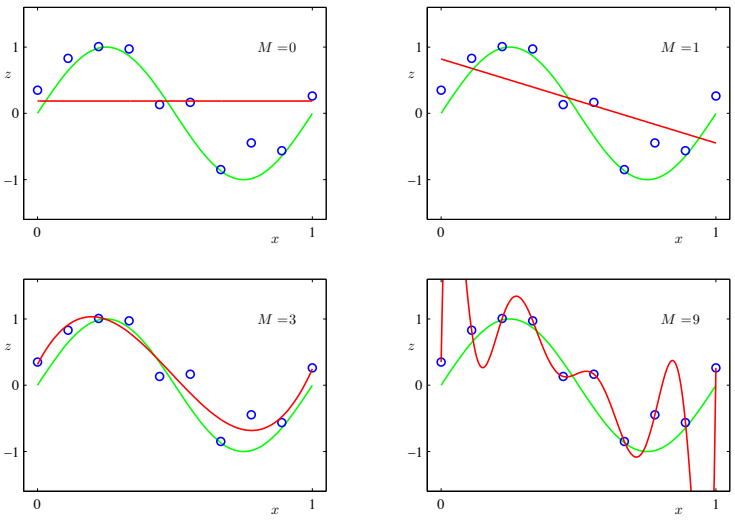
Choosing degree of the polynomial





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

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 function with L2 regularization
(also called ridge regression)

$$E_{\text{ridge}}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i\right]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{26}$$

Where,

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

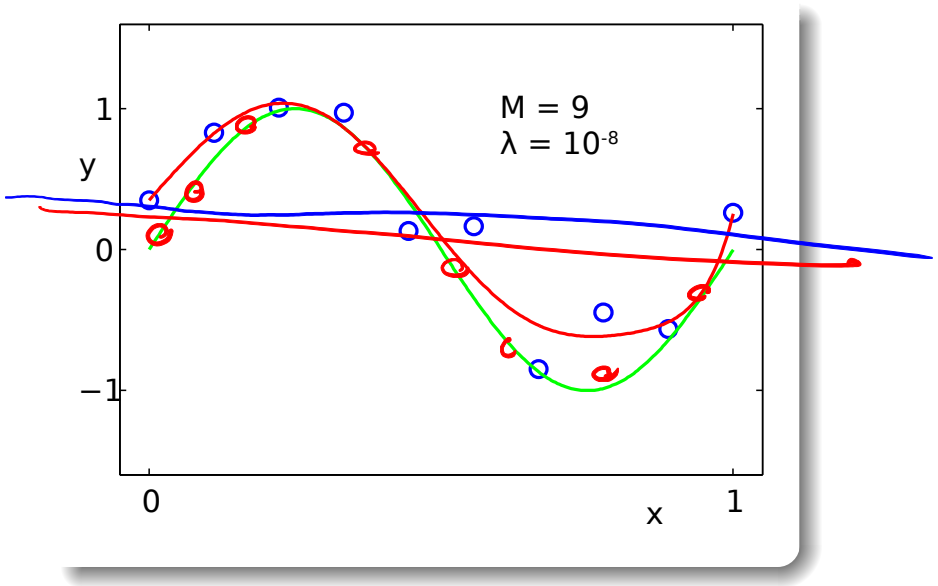
Controlling overfitting with regularization

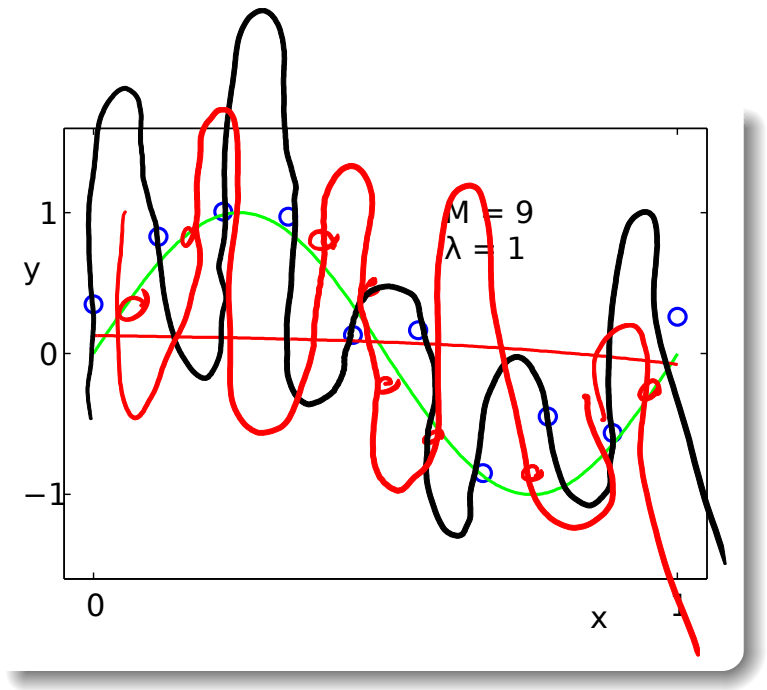
Least squares loss function with L2 regularization
(also called ridge regression)

$$E_{\text{ridge}}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i\right]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{26}$$

Where,

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



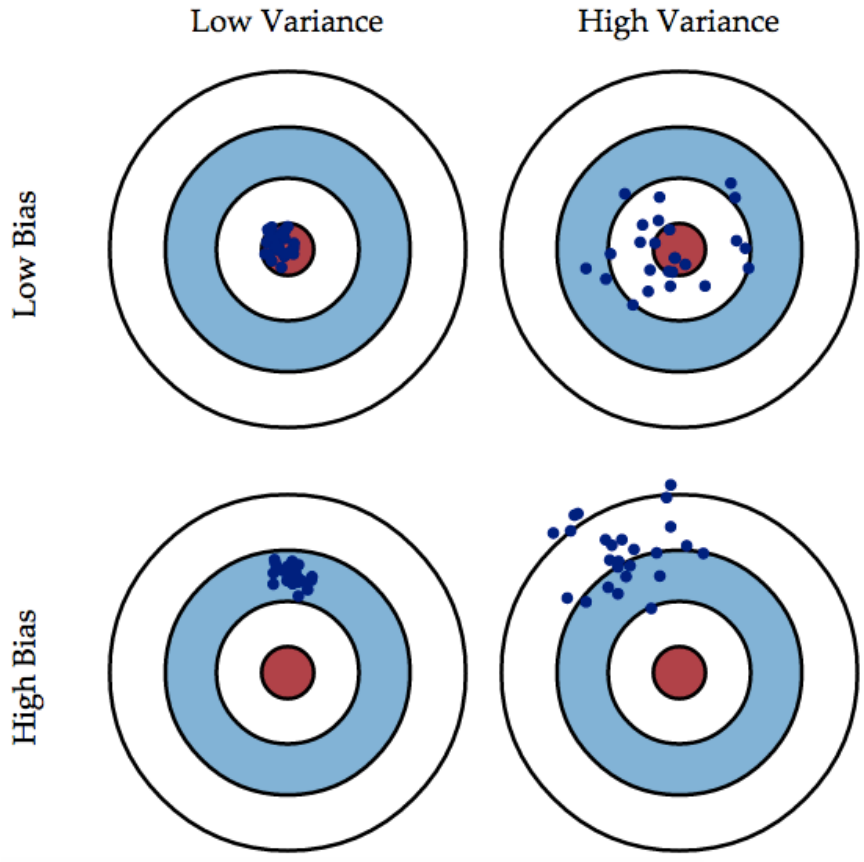


Larger regularization strength λ leads to smaller weights $\boldsymbol{w}$

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



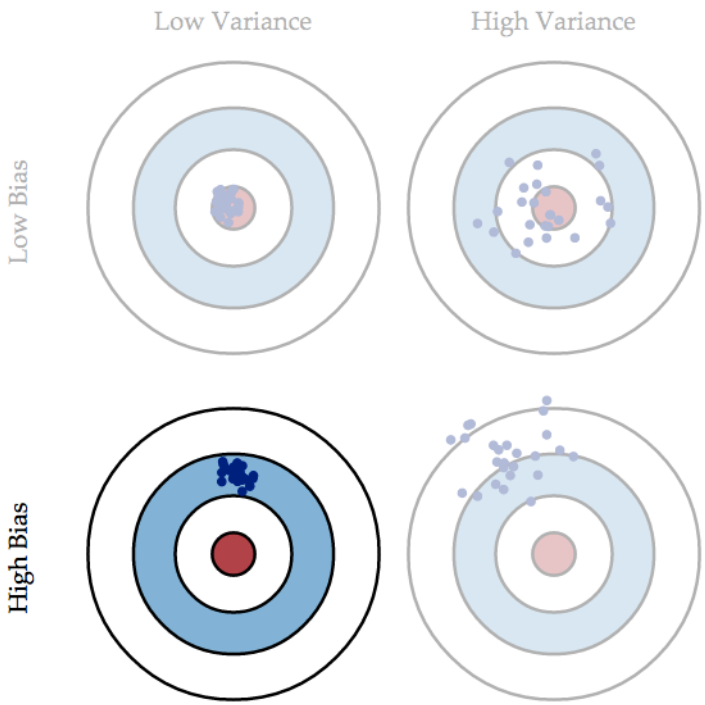
the center of the target:
the true model that predicts the correct values.

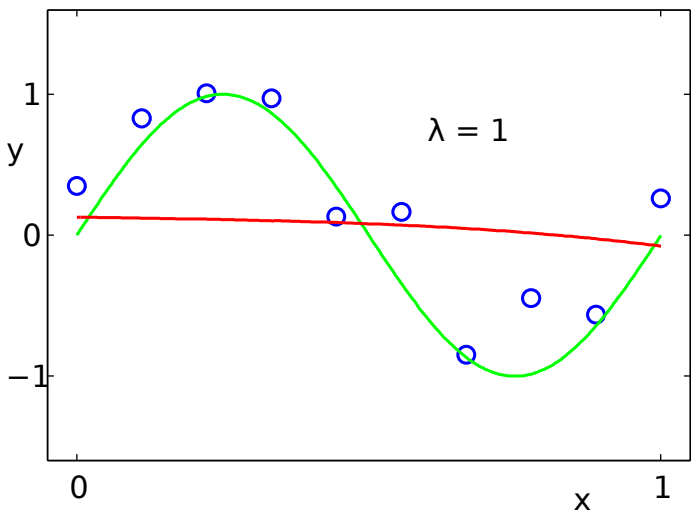
different hits (the blue dots):
different realizations of model given different training data.

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

Bias-variance tradeoff: high bias

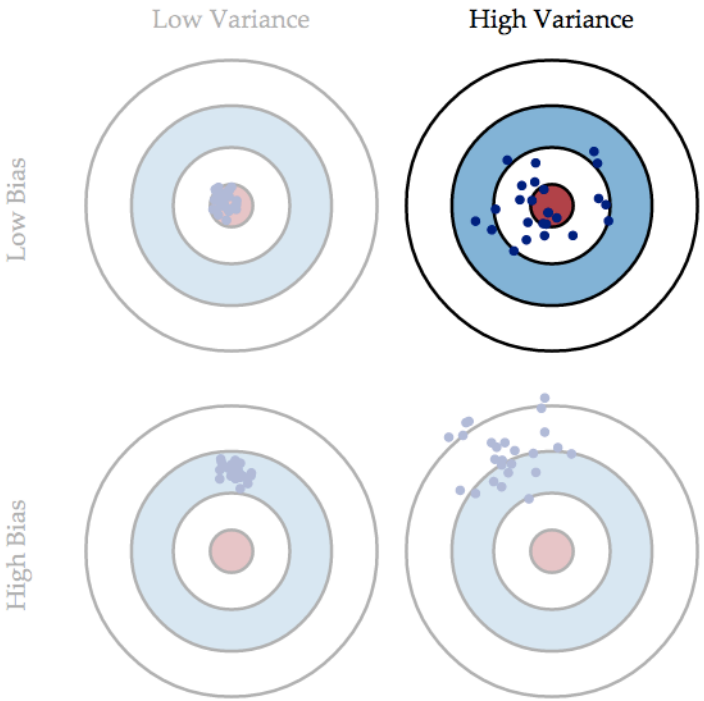
- In case of high bias, the model is too rigid to fit the underlying data distribution.
- This typically happens if the model is misspecified and/or the regularization strength λ is too high.

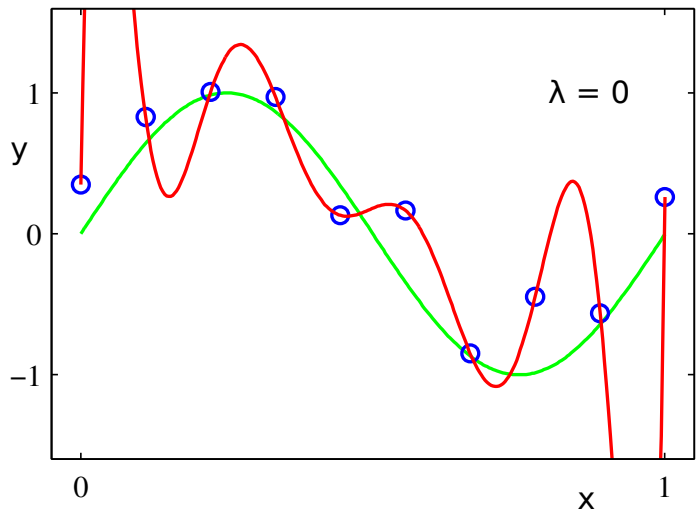




Bias-variance tradeoff: high variance

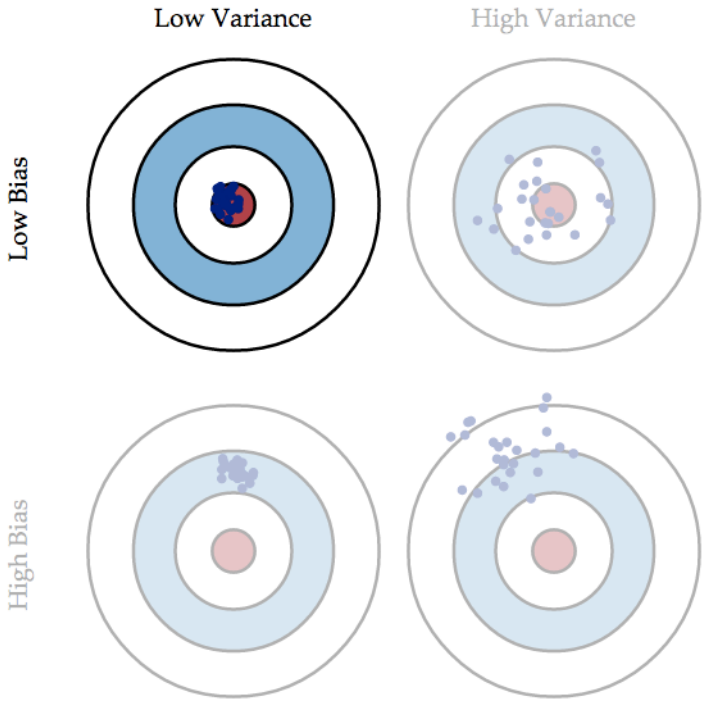
- In case of high variance, the model is too flexible, and therefore captures noise in the data.
- This is exactly what we call overfitting.
- This typically happens when the model has high capacity (= it "memorizes" the training data) and/or λ is too low.

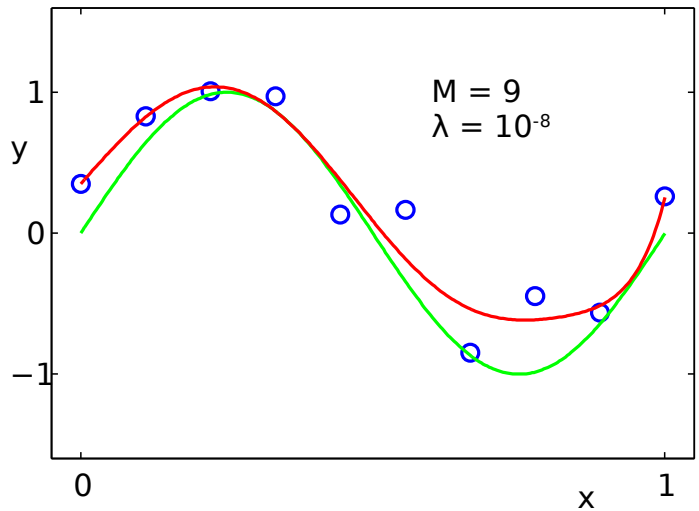




Bias-variance tradeoff

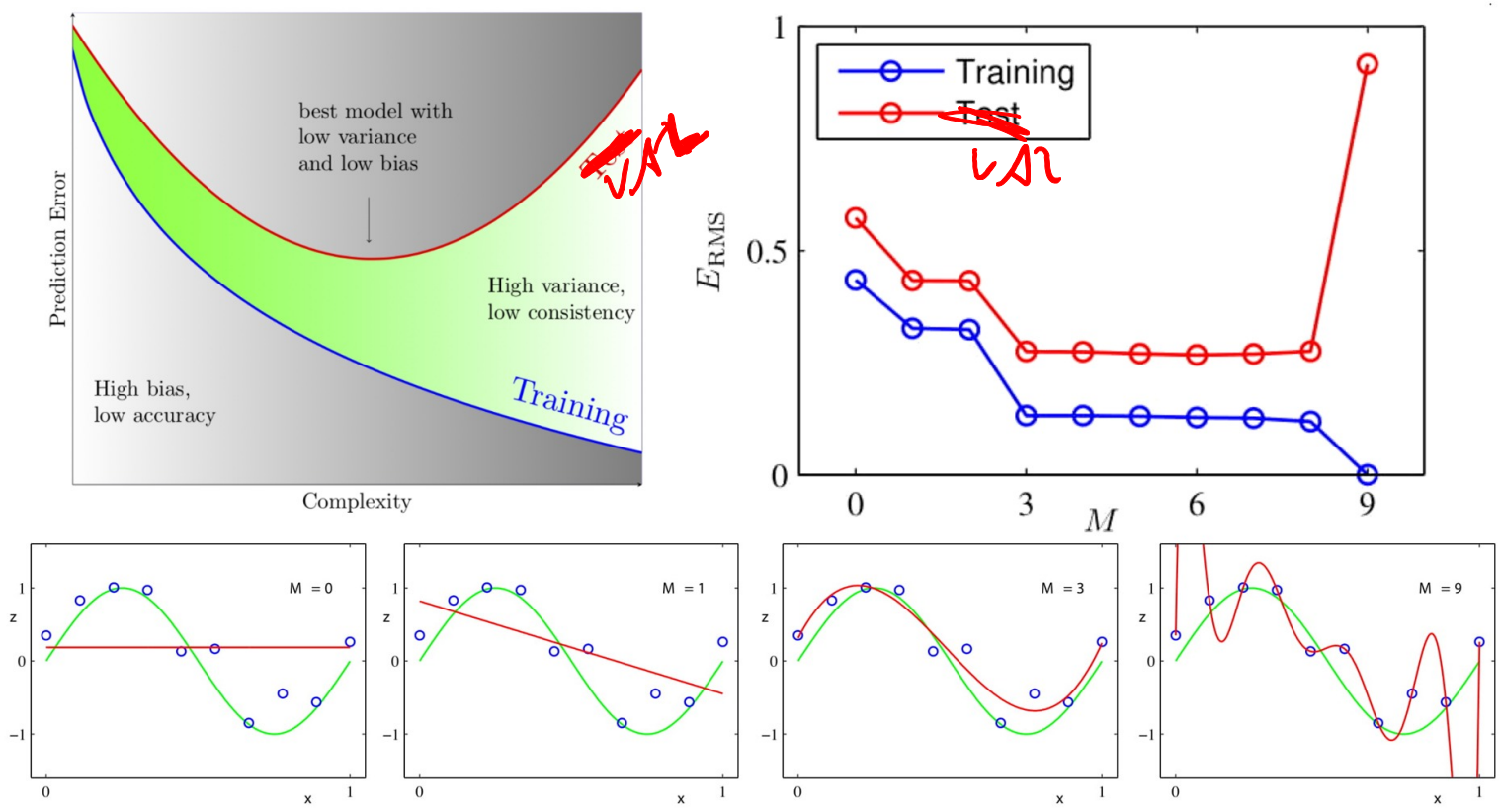
- Of course, we want models that have low bias and low variance, but often those are conflicting goals.
- A popular technique is to select a model with large capacity (e.g. high degree polynomial), and keep the variance in check by choosing appropriate regularization strength λ.





Bias-variance tradeoff

- Bias-variance tradeoff in the case of slide 22 ($\lambda = 0$).



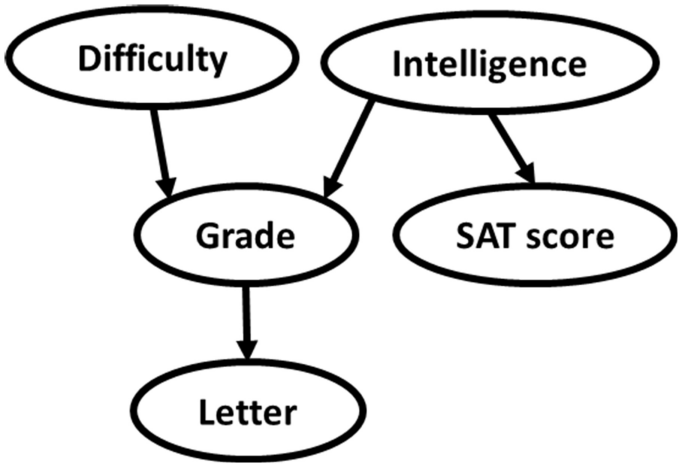
The upper-left figure from: https://eissanematollahi.com/wp-content/uploads/2018/09/Machine-Learning-Basics-1.pdf.

Section 2

Probabilistic Linear Regression

Introduction to probabilistic graphical models (PGMs) [4]

- Bayesian network: is a way to represnt a joint distribution via a directed acyclic graph $G = (V, E)$.
  - Each node represents a random variable.
  - Directed edges are often interpreted as causal relations.



random variables $\in \{0 : low,\ 1 : high\}$

- Graph G provides:
  1) a particular factorization for the joint distribution $p(x_1, ..., x_{|V|})$.
  2) a set of conditional independencies,
     inferred by a routine d_ separation(G).

Example from: [Daphne Koller and Nir Friedman. 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press].

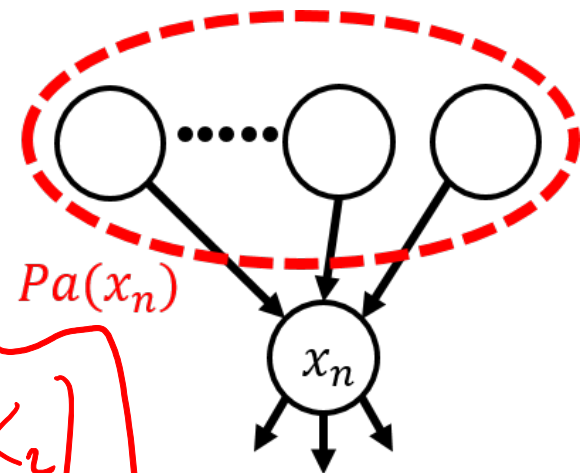
[4] Here we introduce only "undirected" graphical models, i.e. Bayesian networks.

Bayesian networks - factorization

- A Bayesian network specifies the following factorization for the joint distribution $p(x_1, ..., x_{|V|})$.

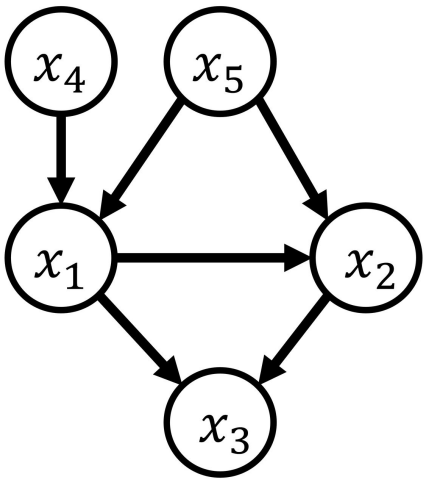
$$p(x_1, ..., x_{|V|}) = \prod_{n=1}^{|V|} p\big(x_n | Pa(x_n)\big) \tag{27}$$

, where $Pa(x_n)$ is the set containing the parents of node x_n in graph G.



Bayesian networks - factorization

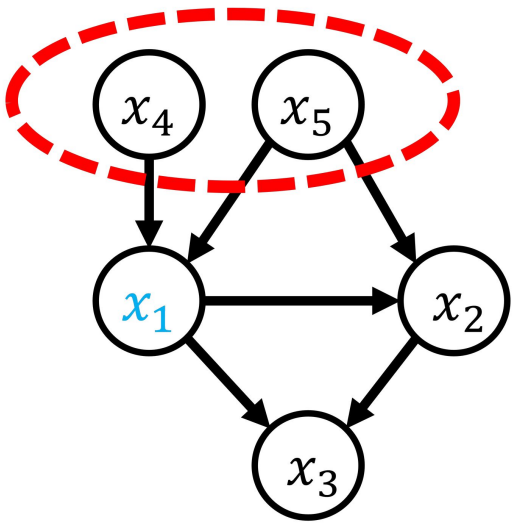
- Example:



$$p(x_1, x_2, x_3, x_4, x_5) = \times \times \times \times$$

Bayesian networks - factorization

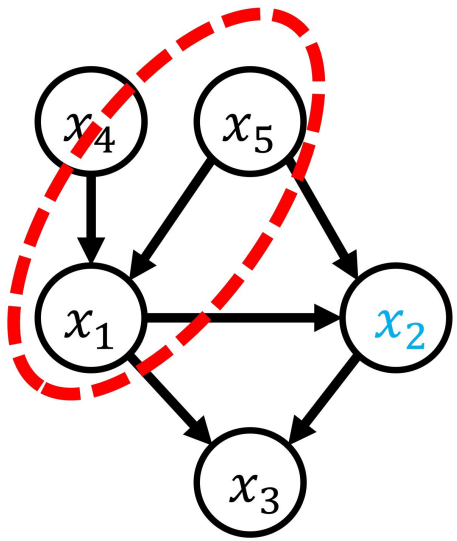
- Example:



$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_4, x_5) \times$$
$$\times$$
$$\times$$
$$\times$$

Bayesian networks - factorization

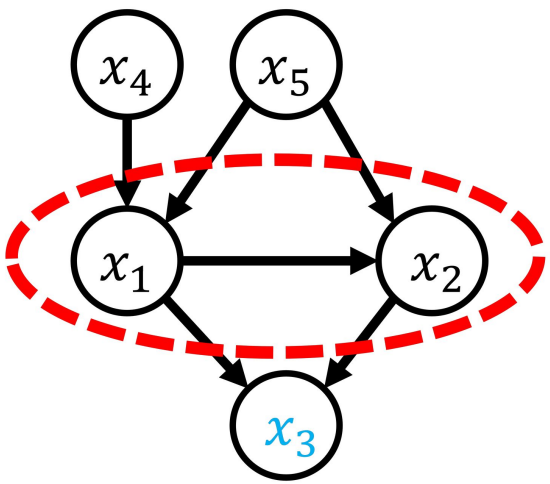
- Example:



$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_4, x_5) \times$$
$$p(x_2|x_1, x_5) \times$$
$$\times$$
$$\times$$

Bayesian networks - factorization

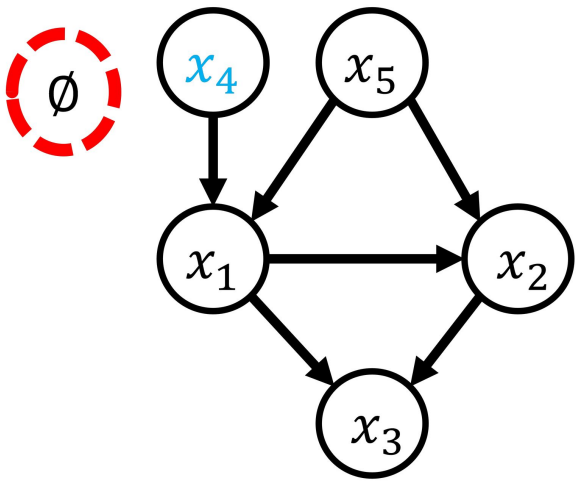
- Example:



$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_4, x_5) \times$$
$$p(x_2|x_1, x_5) \times$$
$$p(x_3|x_1, x_2) \times$$
$$\times$$

Bayesian networks - factorization

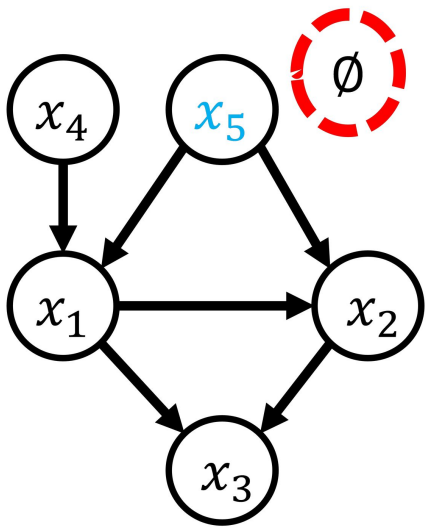
- Example:



$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_4, x_5) \times$$
$$p(x_2|x_1, x_5) \times$$
$$p(x_3|x_1, x_2) \times$$
$$p(x_4) \times$$

Bayesian networks - factorization

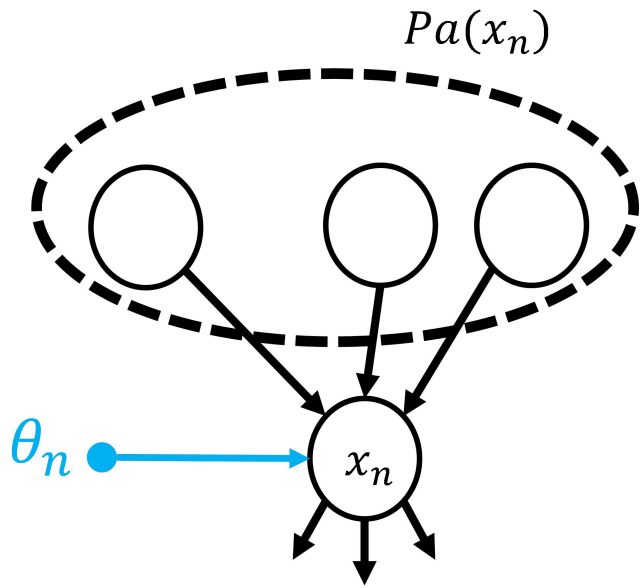
- Example:

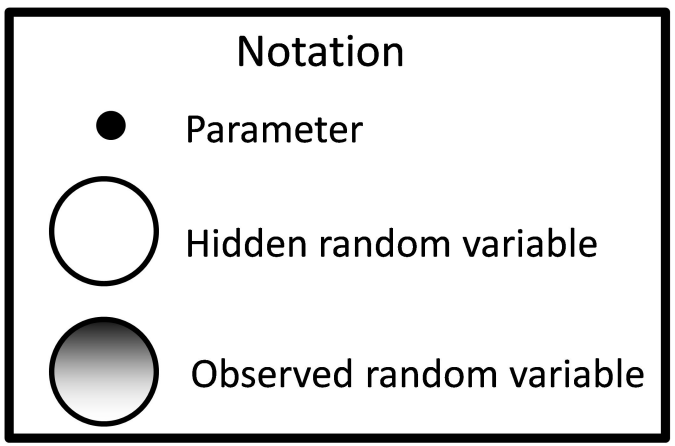


$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1|x_4, x_5) \times p(x_2|x_1, x_5) \times p(x_3|x_1, x_2) \times p(x_4) \times p(x_5)$$

Bayesian networks - model parameters

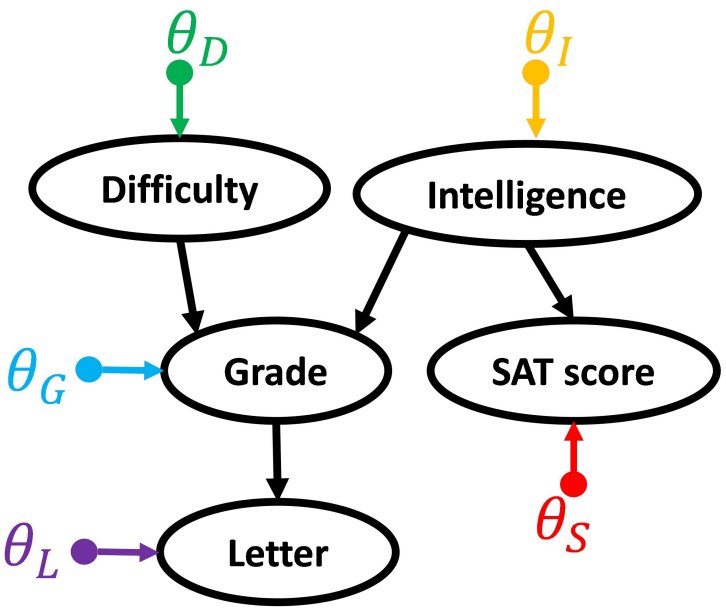
- A set of parameters $\{\theta_1, ..., \theta_{|V|}\}$ parameterize the joint distribution.
- θ_n parameterizes the factor $p\big(x_n|Pa(x_n)\big)$
  - Indeed, the factor is a function of θ_n as well.
    we can write $p\big(x_n|Pa(x_n), \theta_n\big)$
  - In a graphical model, we show θ_n using a filled circle connected to x_n.





Bayesian networks - model parameters

- Example:



$p(D=0)$	$p(D=1)$
0.5	0.5

$p(I=0)$	$p(I=1)$
0.5	0.5

D	I	$p(G=0\|D,I)$	$p(G=1\|D,I)$
0	0	0.5	0.5
0	1	0.1	0.9
1	0	0.9	0.1
1	1	0.5	0.5

I	$p(S=0\|I)$	$p(S=1\|I)$
0	0.9	0.1
1	0.1	0.9

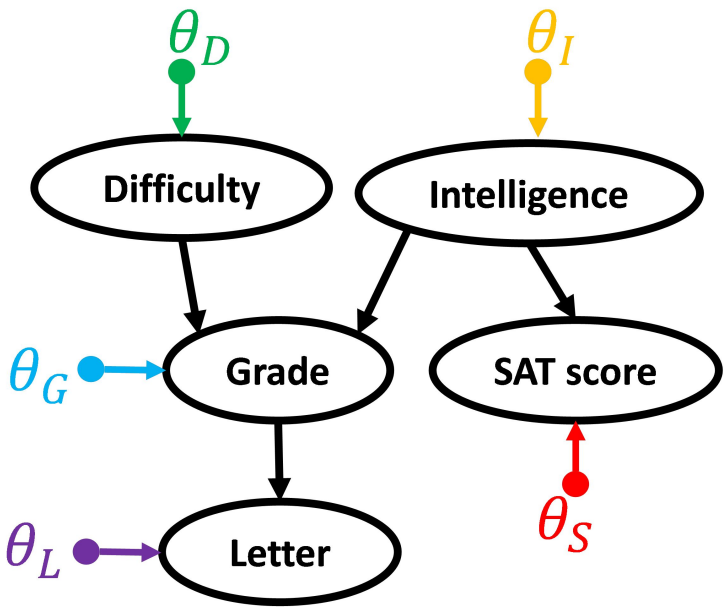
G	$p(L=0\|G)$	$p(L=1\|G)$
0	0.9	0.1
1	0.1	0.9

$p(x_0, x_1, x_2, x_3, x_4, x_5) =$

Example from: [Daphne Koller and Nir Friedman. 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press].

Bayesian networks - generative process

- We can think of the model as a generative process.



$$Difficulty \sim p(D|\theta_D)$$
$$Intelligence \sim p(I|\theta_I)$$
$$Grade \sim p(G|D, I, \theta_G)$$
$$SAT \sim p(S|I, \theta_S)$$
$$Letter \sim p(L|G, \theta_L)$$

Example from: [Daphne Koller and Nir Friedman. 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press].

Probabilistic formulation of linear regression

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

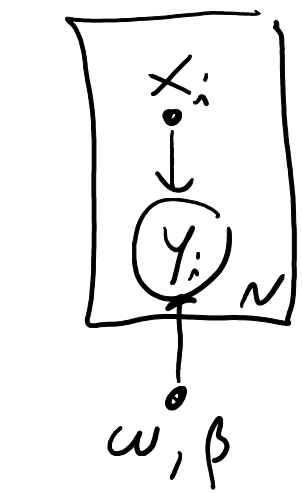
$$y_i = f_{\boldsymbol{w}}(\boldsymbol{x}_i) + \underbrace{\epsilon_i}_{\text{noise}}$$

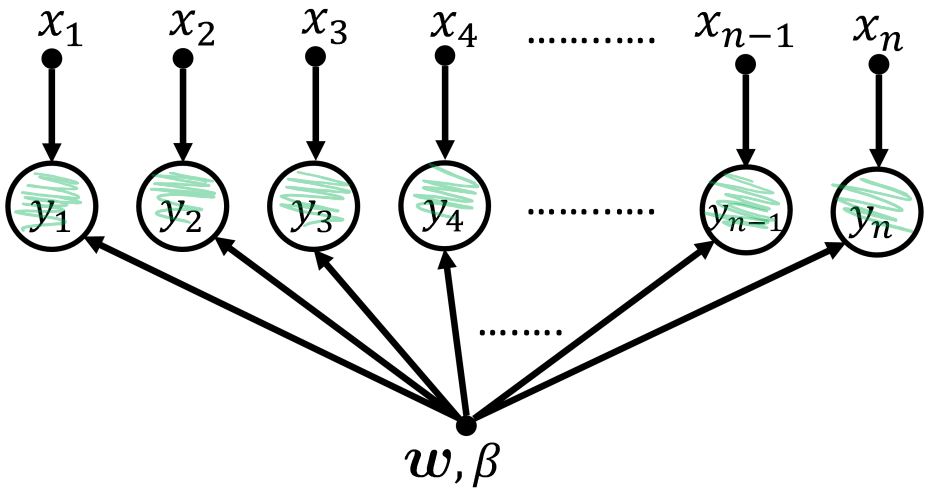
Noise has zero-mean Gaussian distribution with a fixed precision $\beta = \frac{1}{\sigma^2}$

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

This implies that the distribution of the targets is

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





Maximum likelihood

Likelihood of a single sample

$$p(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta) = \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \tag{28}$$

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

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{i=1}^{N} p(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta) \tag{29}$$

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

$$\boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}} = \arg\max_{\boldsymbol{w},\beta} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{30}$$

Maximum likelihood

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

$$\boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}} = \arg\max_{\boldsymbol{w},\beta} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{31}$$

$$= \arg\max_{\boldsymbol{w},\beta} \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{32}$$

$$= \arg\min_{\boldsymbol{w},\beta} -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{33}$$

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

$$E_{\mathrm{ML}}(\boldsymbol{w}, \beta) = -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{34}$$

Maximum likelihood

Simplify the error function

$$E_{\mathrm{ML}}(\boldsymbol{w}, \beta) = -\ln \left[\prod_{i=1}^{N} \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \right] \tag{35}$$

$$= -\ln \left[\prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \tag{36}$$

$$= -\sum_{i=1}^{N} \ln \left[\sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \tag{37}$$

$$= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \tag{38}$$

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

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E_{\mathrm{ML}}(\boldsymbol{w}, \beta) \tag{39}$$

$$= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] \tag{40}$$

$$= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} \tag{41}$$

$$= \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) \tag{42}$$

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

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E_{\mathrm{ML}}(\boldsymbol{w}, \beta) \tag{39}$$

$$= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] \tag{40}$$

$$= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} \tag{41}$$

$$= \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) \tag{42}$$

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

$$\boldsymbol{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{y} = \boldsymbol{\Phi}^{\dagger} \boldsymbol{y} \tag{43}$$

Optimizing log-likelihood w.r.t. β

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

$$\beta_{\mathrm{ML}} = \arg\min_{\beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) \tag{44}$$

$$= \arg\min_{\beta} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \right] \tag{45}$$

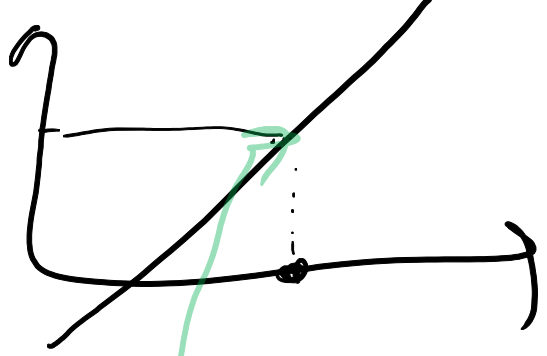
Take derivative w.r.t. β and set it to zero

$$\frac{\partial}{\partial \beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) = \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2\beta} \stackrel{!}{=} 0 \tag{46}$$

Solving for β

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \tag{47}$$

Predicting for new data

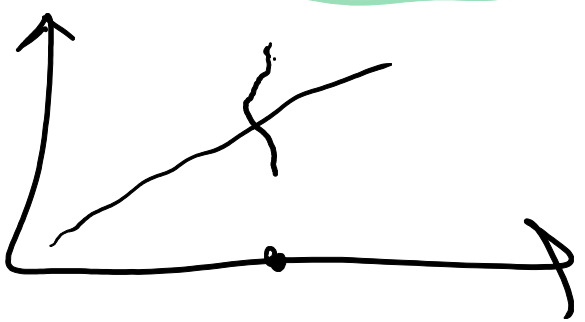


Recall, that

$$y \sim \mathcal{N}(f_{\boldsymbol{w}}(\boldsymbol{x}), \beta^{-1}) \tag{48}$$

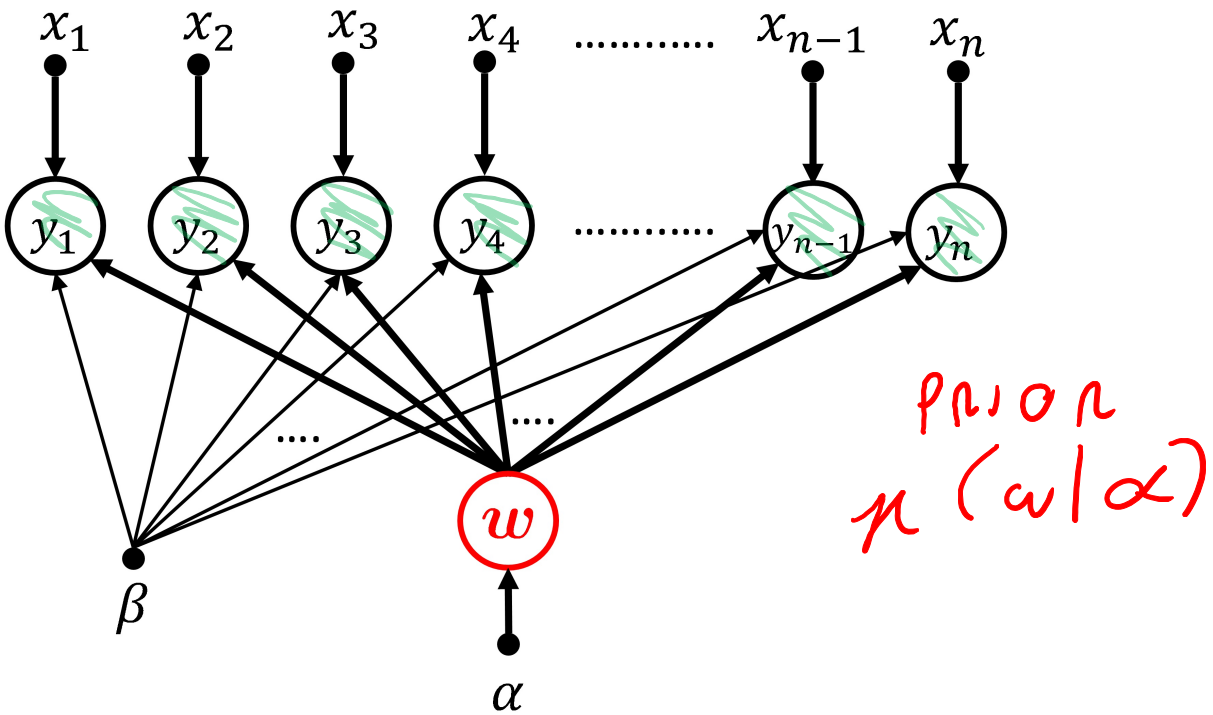
Plugging in the $\boldsymbol{w}_{\mathrm{ML}}$ and β_{ML} into our likelihood we get a predictive distribution that allows us to make prediction $\hat{y}_{new}$ for the new data $\boldsymbol{x}_{new}$.

$$p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta_{\mathrm{ML}}^{-1}) \tag{49}$$



Posterior distribution

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



Posterior distribution

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

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot) = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{w} \mid \cdot)}^{\text{prior}}}{\underbrace{p(\boldsymbol{X}, \boldsymbol{y})}_{\text{normalizing constant}}} \tag{50}$$

$$\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w} \mid \cdot) \tag{51}$$

Posterior distribution

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

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot) = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{w} \mid \cdot)}^{\text{prior}}}{\underbrace{p(\boldsymbol{X}, \boldsymbol{y})}_{\text{normalizing constant}}} \tag{50}$$

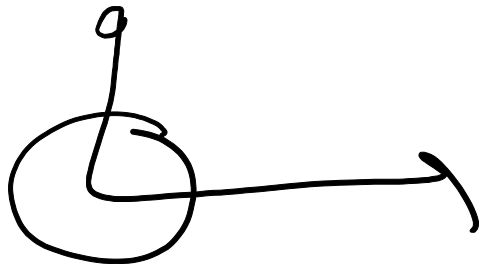
$$\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w} \mid \cdot) \tag{51}$$

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 $\boldsymbol{w}$



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

$$p(\boldsymbol{w} \mid \alpha) = \mathcal{N}(\boldsymbol{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}\right) \tag{52}$$

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 21)
- Likelihood is also Gaussian - simplified calculations

Maximum a posteriori (MAP)

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

$$\boldsymbol{w}_{\text{MAP}} = \arg\max_{\boldsymbol{w}} \; p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \tag{53}$$

$$= \arg\max_{\boldsymbol{w}} \; \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) + \ln p(\boldsymbol{w} \mid \alpha) - \underbrace{\ln p(\boldsymbol{X}, \boldsymbol{y})}_{=\text{const}} \tag{54}$$

$$= \arg\min_{\boldsymbol{w}} - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \tag{55}$$

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

$$E_{\text{MAP}}(\boldsymbol{w}) = - \ln p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \tag{56}$$

$$= - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) + \text{const} \tag{57}$$

MAP error function

Simplify the error function

$$
\begin{aligned}
E_{MAP} &= -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \\
&\quad - \ln \left(\frac{\alpha}{2\pi} \right)^{\frac{M}{2}} + \frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w} \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2 + \text{ const} \\
&\propto \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2}_{\text{ridge regression error fn!}} + \text{const} \qquad \text{where } \lambda = \frac{\alpha}{\beta} \\
&= E_{\text{ridge}}(\boldsymbol{w}) + \text{const}
\end{aligned}
\tag{58}
$$

$\propto$ $\alpha = \lambda \cdot \beta$

MAP estimation with Gaussian prior is equivalent to ridge regression!

Predicting for new data

Recall, that

$$y \sim \mathcal{N}(f_{\boldsymbol{w}}(\boldsymbol{x}), \beta^{-1}) \tag{59}$$

Plugging in the $\boldsymbol{w}_{\text{MAP}}$ into our likelihood we get a predictive distribution that lets us make prediction $\hat{y}_{new}$ for the new data $\boldsymbol{x}_{new}$.

$$p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\text{MAP}}, \beta) = \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{w}_{\text{MAP}}^T \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta^{-1}) \tag{60}$$

Full Bayesian approach

Why limit ourself to the mode $\boldsymbol{w}_{\text{MAP}}$ of the posterior?
Instead, we can try to estimate the full posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$ [5]

$$p(\boldsymbol{w} \mid \mathcal{D}) \propto \overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \overbrace{p(\boldsymbol{w} \mid \alpha)}^{\text{prior}} \tag{61}$$

Since both the likelihood and the prior are Gaussian, we have a closed form for the posterior! (Conjugate prior)

$$p(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_N, \boldsymbol{S}_N) \tag{62}$$

with

$$\text{Mean} = \boldsymbol{m}_N = \boldsymbol{S}_N \left(\boldsymbol{S}_0^{-1} \boldsymbol{m}_0 + \beta \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{y} \right) \tag{63}$$

$$\text{Covariance} = \boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi} \tag{64}$$

$\boldsymbol{m}_0 -$ prior mean, $\boldsymbol{S}_0 -$ prior covariance

[5] To avoid clutter, we use $p(\boldsymbol{w} \mid \mathcal{D})$ as a shorthand for $p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta)$

Posterior distribution

Posterior parameter distribution

$$p(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_N, \boldsymbol{S}_N)$$

with

$$\text{Mean} = \boldsymbol{m}_N = \boldsymbol{S}_N \left(\boldsymbol{S}_0^{-1} \boldsymbol{m}_0 + \beta \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{y} \right)$$

$$\text{Covariance} = \boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi}$$

$\boldsymbol{m}_0 -$ prior mean, $\boldsymbol{S}_0 -$ prior covariance

Observations

- Since we again have a Gaussian, the MAP solution (i.e the mode) equals the mean: $\boldsymbol{w}_{\text{MAP}} = \boldsymbol{m}_N$.
- In the limit of an infinitely broad prior, $\boldsymbol{S}_0^{-1} \to 0$, $\boldsymbol{w}_{\text{MAP}} \to \boldsymbol{w}_{\text{ML}}$
- For $N = 0$, i.e. no data points, we get the prior back.

Sequential Bayesian linear regression

Consider following scenarios

- What if the dataset is too large and can't fit into memory all at once?
- What if the data arrives sequentially (in a stream) and has to be processed in an online manner?

Bayesian framework provides a solution!

1. After processing batch of data $\mathcal{D}_1$ at the first time step $t = 1$, we obtain posterior

$$p(\boldsymbol{w} \mid \mathcal{D}_1) \propto p(\mathcal{D}_1 \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \alpha) \tag{65}$$

2. Use the posterior from step t as a prior for step $t + 1$!

$$\begin{aligned} p(\boldsymbol{w} \mid \mathcal{D}_2, \mathcal{D}_1) &\propto p(\mathcal{D}_2 \mid \boldsymbol{w}) p(\mathcal{D}_1 \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \alpha) \quad \text{(i.i.d.)} && (66) \\ &\propto p(\mathcal{D}_2 \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}_1) && (67) \end{aligned}$$

Sequential Bayesian linear regression: Example

Bayesian regression for the target values

$$y_i = -0.3 + 0.5x_i + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 0.2^2)$$

Sequential Bayesian linear regression: Example

Bayesian regression for the target values

$$y_i = -0.3 + 0.5x_i + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 0.2^2)$$

To model this, we set $\boldsymbol{\phi}(x) = \left[\begin{smallmatrix} 1 \\ x \end{smallmatrix}\right]$ and thus

$$f_{\boldsymbol{w}}(x) = w_0 + w_1 x$$

Sequential Estimation

The demo shows how the posterior's breadth gets smaller as more and more points are taken into account, and how its mode converges to the optimal (true) values of the weights (white cross).

Sequential Bayesian linear regression: Example

middle column:

$p\big(w_0, w_1 | \{(x_i, y_i)\}_{i=1}^{t}\big)$, for $t > 1$.

the prior $p\big(w_0, w_1\big)$ in the first row.

left-hand column:

value of $p\big(x_t, y_t | w_0, w_1\big)$

as a function of (w_0, w_1).

right-hand column:

six sample lines from the current posterior distribution.

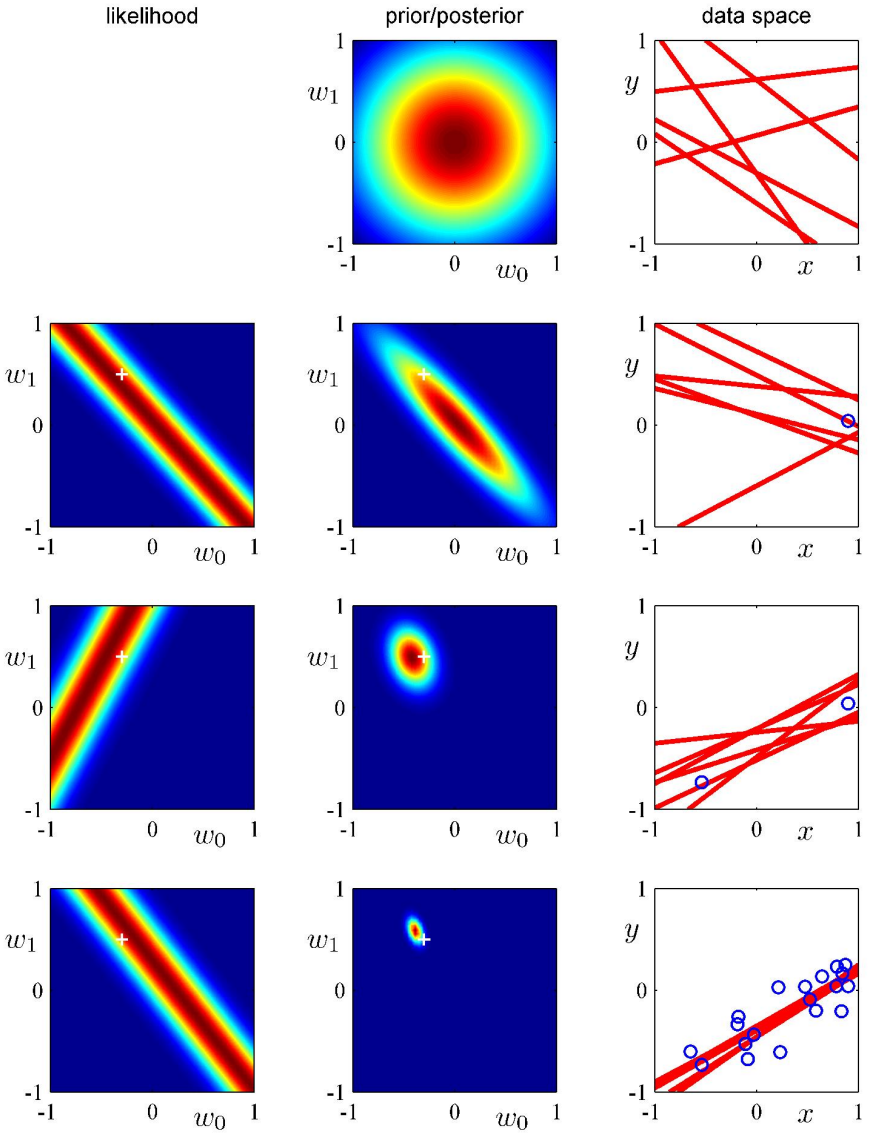
blue circles in the right column:

the observed data-points.

up to time t.

white cross:

the true value of (w_0, w_1).



Posterior predictive distribution

After observing the data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, we can compute the full posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$.

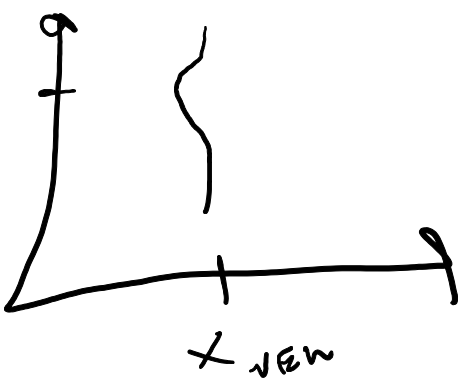
Usually, what we are actually interested in is the prediction $\hat{y}_{new}$ for a new data point $\boldsymbol{x}_{new}$ - the model parameters $\boldsymbol{w}$ are just a means to achieve this.

The posterior predictive distribution is computed as

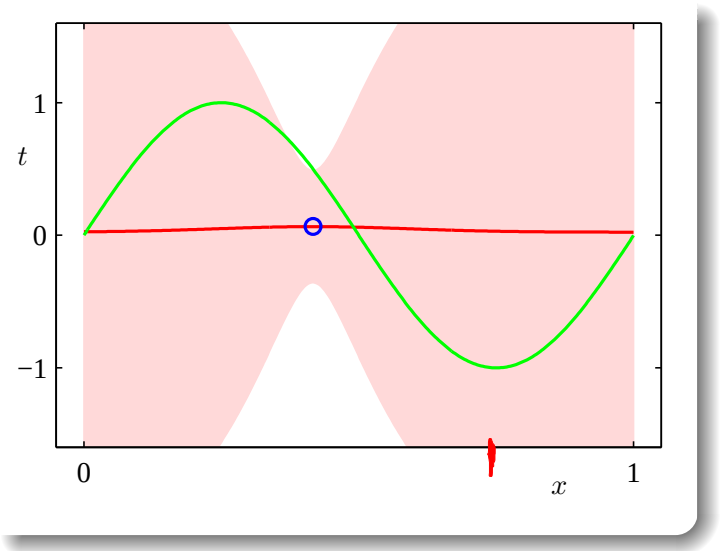
$$p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \mathcal{D}) = \int p(\hat{y}_{new}, \boldsymbol{w} \mid \boldsymbol{x}_{new}, \mathcal{D}) \mathrm{d}\boldsymbol{w} \tag{68}$$

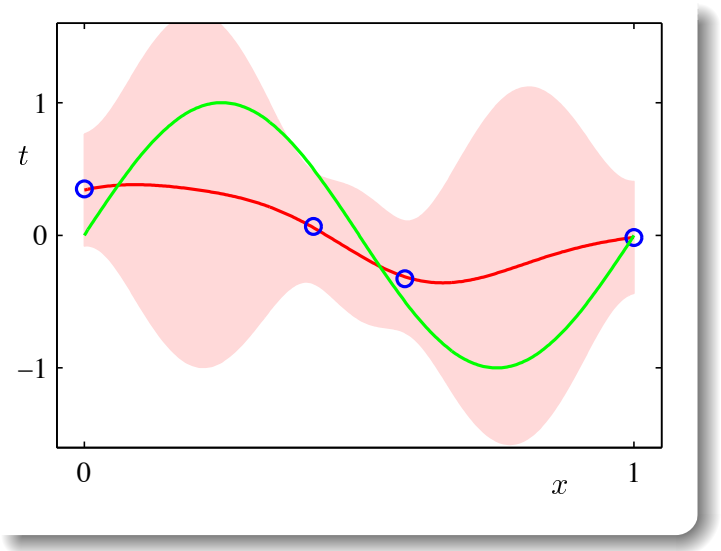
$$= \int p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}) \mathrm{d}\boldsymbol{w} \tag{69}$$

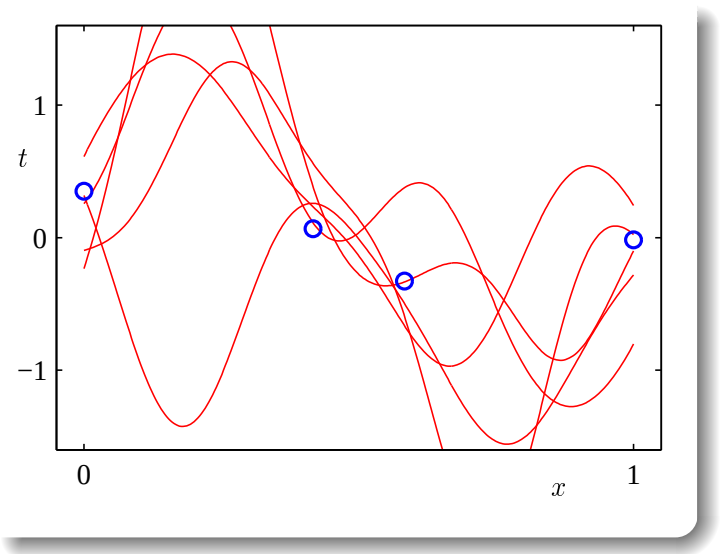
$$= \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{m}_N^T \phi(\boldsymbol{x}_{new}), \beta^{-1} + \phi(\boldsymbol{x}_{new})^T \boldsymbol{S}_N \phi(\boldsymbol{x}_{new})) \tag{70}$$

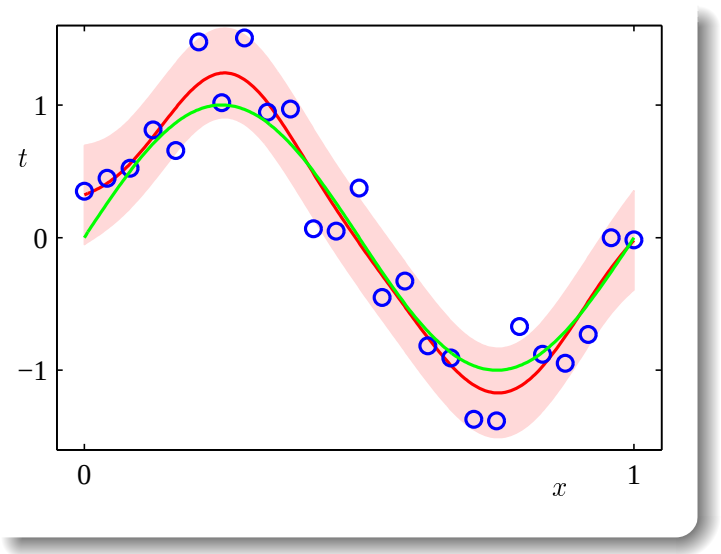


Example of posterior predictive distribution









Summary

- Optimization-based approaches to regression have probabilistic interpretations
  - Least squares regression $\iff$ Maximum likelihood (Slide 45)
  - Ridge regression $\iff$ Maximum a posteriori (Slide 50)
- Even nonlinear dependencies in the data can be captured by a model linear w.r.t. weights $\boldsymbol{w}$ (Slide 14)
- Penalizing large weights helps to reduce overfitting (Slide 21)
- Full Bayesian can be processed sequentially - same result (Slide 55)