Machine Learning

Lecture 4: Linear Regression

Prof. Dr. Stephan Günnemann

Data Mining and Analytics
Technical University of Munich

13.11.2017

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 $\boldsymbol{X}$

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

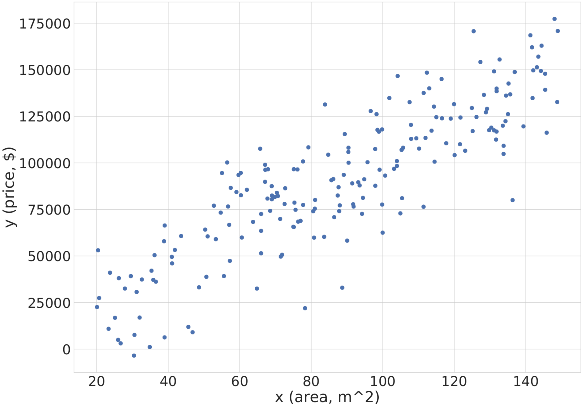
Section 1

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.

Goal of regression is to find f(x)

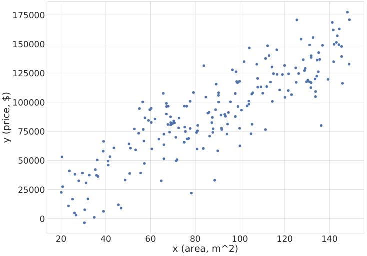


How do we estimate a price of a new house with area x_{new}?

Regression problem

Given

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



Find

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

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

x_i = D dimensional vector

D features for each x, N data poins

[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

Normal Dist(Mean,Variance)

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

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

Normal Distribution for noise = epsilon

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

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

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

w0 = offset

w = weight vector

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

Absorbing the bias term

The linear function is given by

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

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

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 $\boldsymbol{w}$:

D + 1 dimension vector

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

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?

best line that fits the data

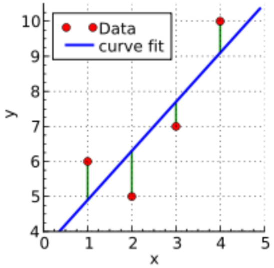
Error function

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

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

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



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

Objective

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

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

Best line or w is the one that minimizes the error

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

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

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

X = N*D dimesion
w = D dimension vector
y = N dimensional vector

Optimal solution

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

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

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

$$= \underbrace{\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w}}_{\boxed{\text{Eq 81}}} - \underbrace{\boldsymbol{X}^T \boldsymbol{y}}_{\boxed{\text{Eq 69}}} \tag{13}$$

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

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

X^T X = D*D
dimensional

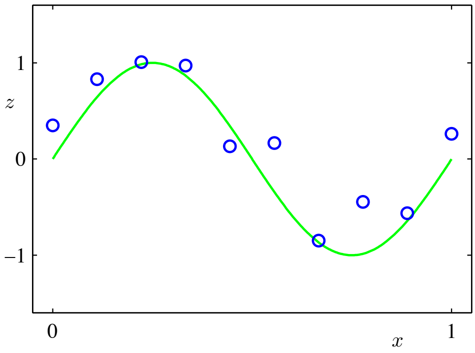
$\boldsymbol{X}^\dagger$ is called Moore-Penrose pseudo-inverse of $\boldsymbol{X}$ (because for an invertible square matrix, $\boldsymbol{X}^\dagger = \boldsymbol{X}^{-1}$).

Second derivative = Hessian matrix and
should be positive for minima

[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 $\boldsymbol{x}$ is not linear?



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

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

Polynomials

Here basis function are fixed, but they vary in neural networks and deep learning

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

Instead of having a linear function of x , we use a polynomial function of x , i.e x^j

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

Or more generally

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

$\phi_0(x) = x^0 = 1$

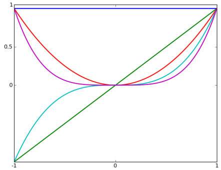
Define $\phi_0 = 1$

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

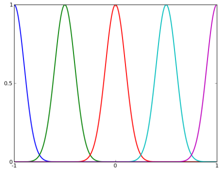
The function f is still linear in $\boldsymbol{w}$ (despite not being linear in $\boldsymbol{x}$)!

Typical basis functions

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

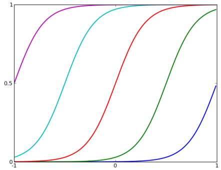


Gaussian $\phi_j(x) = \mathrm{e}^{\frac{-(x-\mu_j)^2}{2s^2}}$



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

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

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

with

Basis Function = Fi = applies to all the dimesion of vector x1

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

N data points and M dimensional basis function = Design Matrix

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

$$\begin{align} \boldsymbol{w}^* &= (\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{y} \tag{23} \\ &= \boldsymbol{\Phi}^\dagger \boldsymbol{y} \tag{24} \end{align}$$

Compare this to Equation 15:

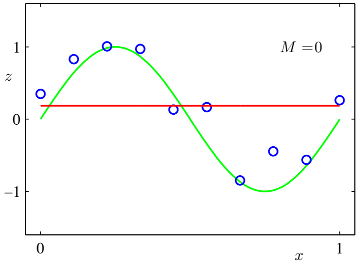
$$\boldsymbol{w}^* = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} \tag{25}$$

Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

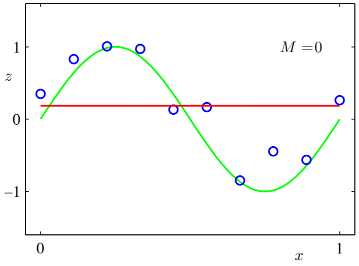
Choosing degree of the polynomial

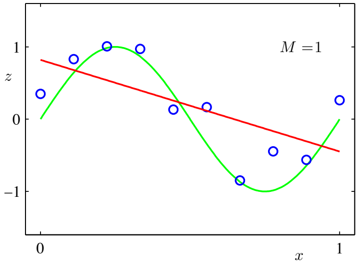
How do we choose the degree of the polynomial M?



Choosing degree of the polynomial

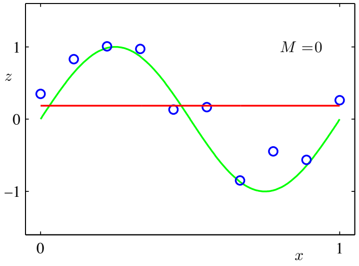
How do we choose the degree of the polynomial M?

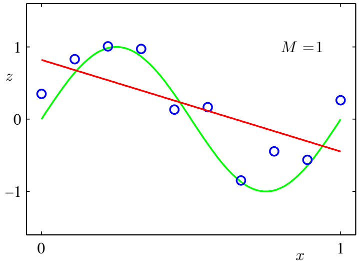


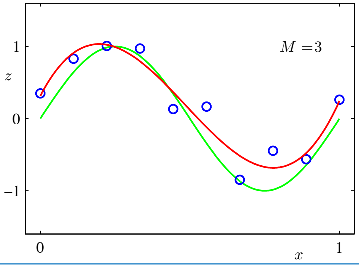


Choosing degree of the polynomial

How do we choose the degree of the polynomial M?



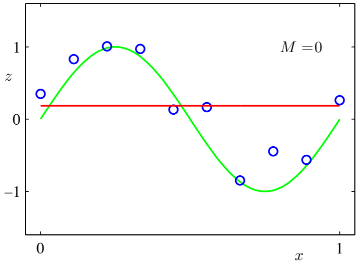


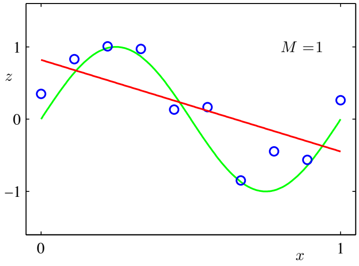


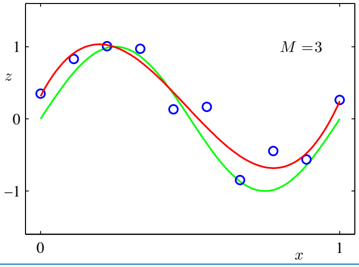
Choosing degree of the polynomial

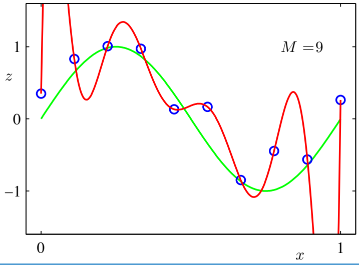
Exactly opposite to what happened in KNN classification.

How do we choose the degree of the polynomial M?

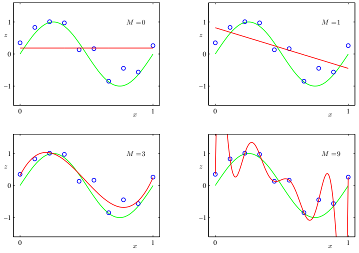


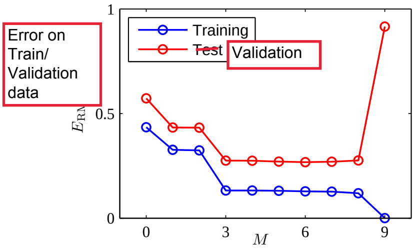






Choosing degree of the polynomial



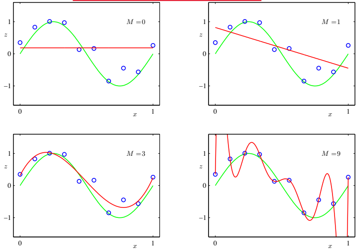


One valid solution is to choose M using the standard train-validation split approach.

Choosing degree of the polynomial

As degree increases, weight w values increases.
Its because of high fluctuations in high degree
polynomial function

1,2 = too simple model
so underfitting



too
flexible -
so
overfittin
g

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	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)

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

L2 Norm

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

At lambda value gets large, we penalize a the weight , ie. w gets smooth or st. line

Controlling overfitting with regularization

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

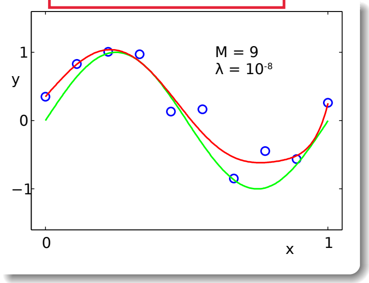
Instead of finding polynomial, we find the value of lambda

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

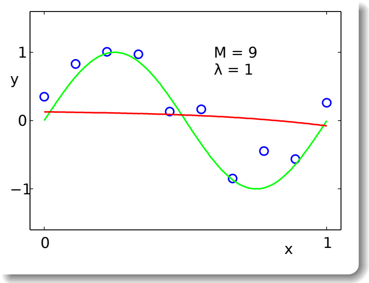
Where,

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

Small lambda



High lambda

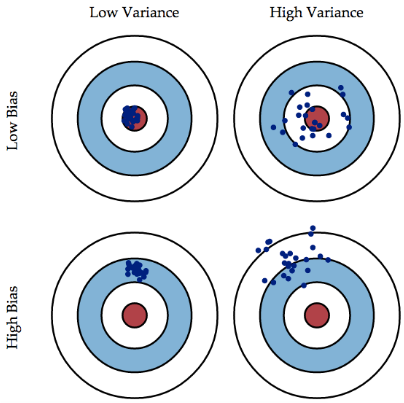


Larger regularization strength λ leads to smaller weights $\boldsymbol{w}$

Bias-variance tradeoff

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

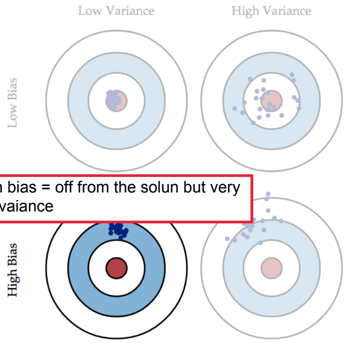
- **Bias** - expected error due to model mismatch
- **Variance** - variation due to randomness in training data



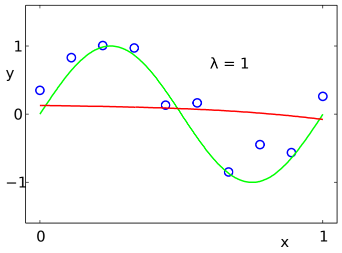
[3]See Bishop Section 3.2 for a more rigorous mathematical derivation

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.



High bias = off from the solun but very low vaiance

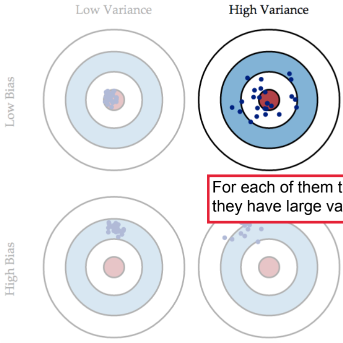


High Bias, Low Variance = Underfitting
Low Bias, High Variance = Overfitting

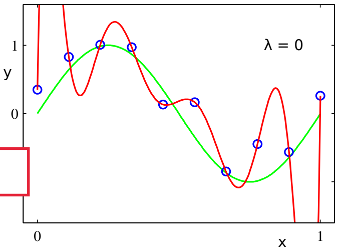
Bias-variance tradeoff: high variance

- 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.

capacity means degree of freedom

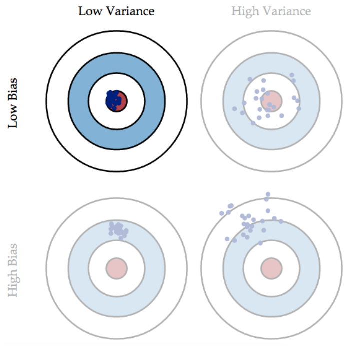


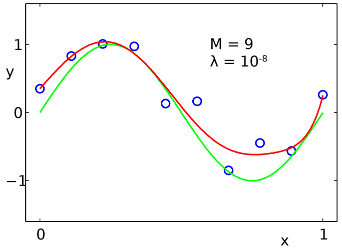
For each of them the fit is good, but they have large variance/spread



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.





We cant minimize variance and bias at the same time.
Select the model with high variance and low bias and regularize to reduce the overfitting

Section 2

Probabilistic Linear Regression

Probabilistic formulation

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

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

Property of Normal / Gaussian distribution

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

$$p(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta) = \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \tag{27}$$

independent, not identical distributed

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

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{i=1}^{N} p(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta) \tag{28}$$

p = density , not probability

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

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

Maximum likelihood

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

$$\begin{aligned} \boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}} &= \underset{\boldsymbol{w},\beta}{\arg\max}\, p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) && (30) \\ &= \underset{\boldsymbol{w},\beta}{\arg\max}\, \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) && (31) \\ &= \underset{\boldsymbol{w},\beta}{\arg\min}\, -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) && (32) \end{aligned}$$

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

$$E_{\mathrm{ML}}(\boldsymbol{w}, \beta) = -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{33}$$

Maximum likelihood

Simplify the error function

$$
\begin{aligned}
E_{\mathrm{ML}}(\boldsymbol{w}, \beta) &= -\ln \left[\prod_{i=1}^{N} \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \right] && (34) \\
&= -\ln \left[\prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] && (35) \\
&= -\sum_{i=1}^{N} \ln \left[\sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] && (36) \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi && (37)
\end{aligned}
$$

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

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E_{\mathrm{ML}}(\boldsymbol{w}, \beta) \tag{38}$$

$$= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] \tag{39}$$

$$= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} \tag{40}$$

$$= \arg\min_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) \tag{41}$$

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

$$\boldsymbol{w}_{\text{ML}} = \arg\min_{\boldsymbol{w}} E_{\text{ML}}(\boldsymbol{w}, \beta) \tag{38}$$

$$= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{= \text{const}} \right] \tag{39}$$

$$= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares error fn!}} \tag{40}$$

$$= \arg\min_{\boldsymbol{w}} E_{\text{LS}}(\boldsymbol{w}) \tag{41}$$

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

$$\boldsymbol{w}_{\text{ML}} = (\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{y} = \boldsymbol{\Phi}^{\dagger} \boldsymbol{y} \tag{42}$$

Optimizing log-likelihood w.r.t. β

1/beta = Variance
beta = Precision

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

$$\beta_{\mathrm{ML}} = \arg\min_{\beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) \tag{43}$$

$$= \arg\min_{\beta} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \right] \tag{44}$$

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

$$\frac{\partial}{\partial \beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) = \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2\beta} \overset{!}{=} 0 \tag{45}$$

Solving for β

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \tag{46}$$

Predicting for new data

Recall, that

$$y \sim \mathcal{N}(f_{\boldsymbol{w}}(\boldsymbol{x}), \beta^{-1}) \tag{47}$$

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

$$p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{w}_{\mathrm{ML}}^{T} \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta_{\mathrm{ML}}^{-1}) \tag{48}$$

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

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot) = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{w} \mid \cdot)}^{\text{prior}}}{\underbrace{p(\boldsymbol{X}, \boldsymbol{y})}_{\text{normalizing constant}}} \tag{49}$$

$$\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w} \mid \cdot) \tag{50}$$

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

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

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot) = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{w} \mid \cdot)}^{\text{prior}}}{\underbrace{p(\boldsymbol{X}, \boldsymbol{y})}_{\text{normalizing constant}}} \tag{49}$$

$$\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \cdot p(\boldsymbol{w} \mid \cdot) \tag{50}$$

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 $p(\boldsymbol{w} \mid \cdot)$?

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

Multiple univariate Gaussian

$$p(\boldsymbol{w} \mid \alpha) = \mathcal{N}(\boldsymbol{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}\right) \tag{51}$$

where,

α - precision of the distribution

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

Motivation:

- Higher probability is assigned to small values of $\boldsymbol{w}$ $\implies$ prevents overfitting (recall slide 21)
- Likelihood is also Gaussian - simplified calculations

Large weights wi = overfitting

Gaussian = normalizes things

Maximum a posteriori (MAP)

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

Evidence

$$\begin{aligned}
\boldsymbol{w}_{\mathrm{MAP}} &= \arg\max_{\boldsymbol{w}} \ p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \qquad (52)\\
&= \arg\max_{\boldsymbol{w}} \ \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) + \ln p(\boldsymbol{w} \mid \alpha) - \underbrace{\ln p(\boldsymbol{X}, \boldsymbol{y})}_{=\text{const}} \qquad (53)\\
&= \arg\min_{\boldsymbol{w}} - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \qquad (54)
\end{aligned}$$

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

$$\begin{aligned}
E_{\mathrm{MAP}}(\boldsymbol{w}) &= - \ln p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \qquad (55)\\
&= - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) + \text{const} \qquad (56)
\end{aligned}$$

minimize

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

MAP error function

Simplify the error function

$$
\begin{aligned}
E_{MAP} &= -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \\
&= \frac{\beta}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2}\ln\beta + \frac{N}{2}\ln 2\pi \\
&\quad - \ln\left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} + \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} \\
&= \frac{\beta}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2 + \text{const} \\
&= \underbrace{\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2}_{\text{ridge regression error fn!}} + \text{const} \qquad \text{where } \lambda = \frac{\alpha}{\beta} \\
&= E_{\text{ridge}}(\boldsymbol{w}) + \text{const}
\end{aligned}
\tag{57}
$$

likelihood

Prior

Divide this term by beta

MAP estimation with Gaussian prior is equivalent to ridge regression!

Predicting for new data

Recall, that

$$y \sim \mathcal{N}(f_{\boldsymbol{w}}(\boldsymbol{x}), \beta^{-1}) \tag{58}$$

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

$$p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \boldsymbol{w}_{\mathrm{MAP}}, \beta_{\mathrm{MAP}}) = \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{w}_{\mathrm{MAP}}^T \boldsymbol{\phi}(\boldsymbol{x}_{new}), \beta^{-1}) \tag{59}$$

Full Bayesian approach

S0 -1 = Covariance matrix , the alpha matrix

Why limit ourself to the mode $\boldsymbol{w}_{\text{MAP}}$ of the posterior?
Instead, we can try to estimate the full posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$ [4]

$$p(\boldsymbol{w} \mid \mathcal{D}) \propto \overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)}^{\text{likelihood}} \overbrace{p(\boldsymbol{w} \mid \alpha)}^{\text{prior}} \tag{60}$$

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

$$p(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_N, \boldsymbol{S}_N) \tag{61}$$

with

$$\text{Mean} = \boldsymbol{m}_N = \boldsymbol{S}_N \left(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \beta\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{y}\right) \tag{62}$$

$$\text{Covariance} = \boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} \tag{63}$$

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

[4] 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

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

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

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.

fi =0 if no data

No data means no experiment , so likelihood

Sequential Bayesian linear regression

Data D is very large. So we load sub parts.
D = D1 + D2 + ...
We use probabilistic model so that we can compute in parts Di and then multiply.
Assumption : All Di are IID

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

$$p(\boldsymbol{w} \mid \mathcal{D}_1) \propto p(\mathcal{D}_1 \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \alpha) \tag{64}$$

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

$$p(\boldsymbol{w} \mid \mathcal{D}_2, \mathcal{D}_1) \propto p(\mathcal{D}_2 \mid \boldsymbol{w}) p(\mathcal{D}_1 \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \alpha) \qquad \text{(i.i.d.)} \tag{65}$$

$$\propto p(\mathcal{D}_2 \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}_1) \tag{66}$$

Posterior of D1 becomes the prior of data D2

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

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

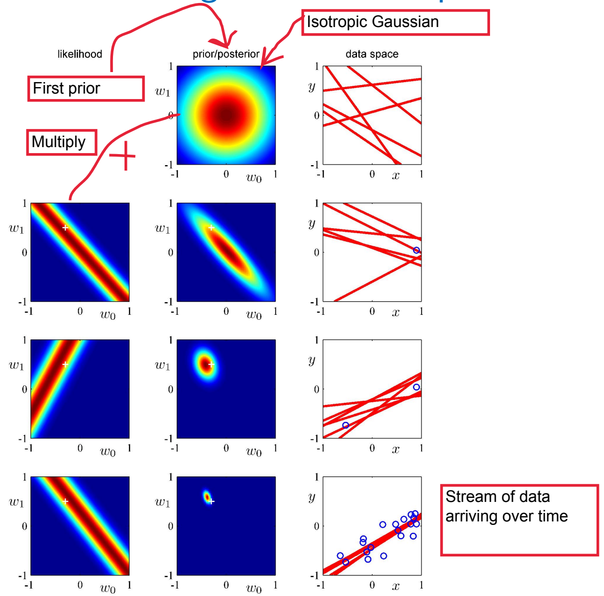
To model this, we set $\boldsymbol{\phi}(x) = \left[\begin{smallmatrix} 1 \\ x \end{smallmatrix}\right]$ and thus

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

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



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

$$p(\hat{y}_{new} \mid \boldsymbol{x}_{new}, \mathcal{D}) = \int p(\hat{y}_{new}, \boldsymbol{w} \mid \boldsymbol{x}_{new}, \mathcal{D}) \mathrm{d}\boldsymbol{w} \tag{67}$$

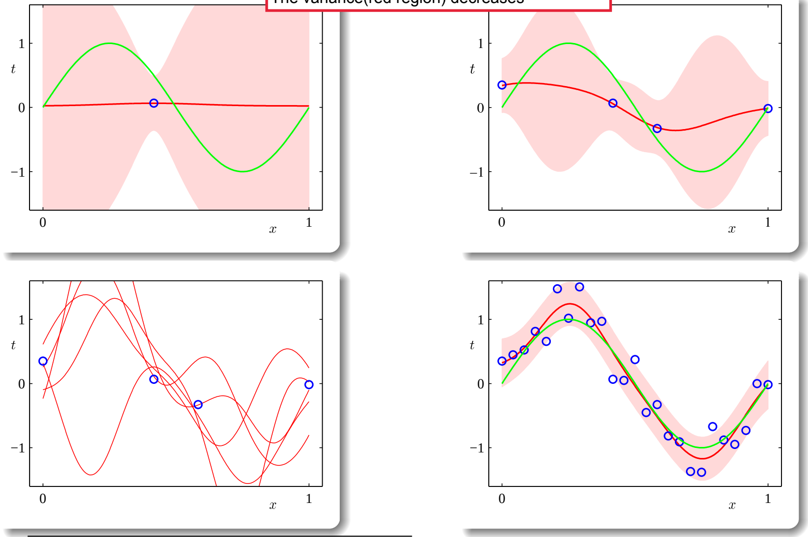
p(w|D) = the more higher the weight the more probabilistic it becomes

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

$$= \mathcal{N}(\hat{y}_{new} \mid \boldsymbol{m}_N^T \phi(\boldsymbol{x}_{new}), \beta^{-1} + \phi(\boldsymbol{x}_{new})^T \boldsymbol{S}_N \phi(\boldsymbol{x}_{new})) \tag{69}$$

Example of posterior predictive distribution

More points you get , more sure you get about distribution.
The variance(red region) decreases



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

Summary

- Optimization-based approaches to regression have probabilistic interpretations
  - Least squares regression $\iff$ Maximum likelihood (Slide 33)
  - Ridge regression $\iff$ Maximum a posteriori (Slide 37)
- 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) like Putting prior on the distribution
- Full Bayesian can be processed sequentially - same result (Slide 42)