Machine Learning

Lecture 6: Linear Regression

Prof. Dr. Aleksandar Bojchevski

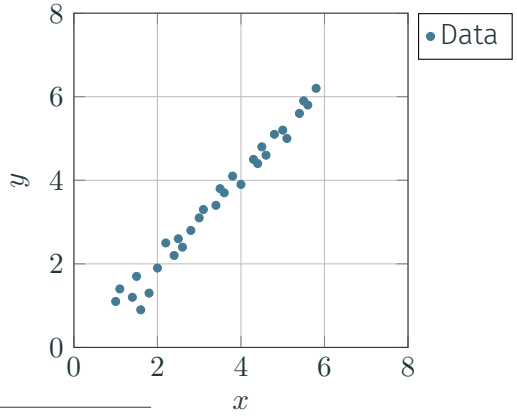
02.05.23

Outline

Regression problem

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ of observations, predict y_{new} for a x_{new}?

Given features $\boldsymbol{x}_i \in \mathbb{R}^D$ and labels $y_i \in \mathbb{R}$, find a mapping f such that $y_i \approx f(\boldsymbol{x}_i)$

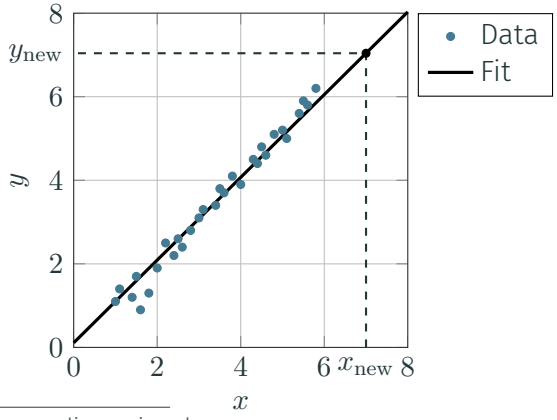


On the figure $x_i \in \mathbb{R}$ is one dimensional.

Regression problem

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ of observations, predict y_{new} for a x_{new}?

Given features $\boldsymbol{x}_i \in \mathbb{R}^D$ and labels $y_i \in \mathbb{R}$, find a mapping f such that $y_i \approx f(\boldsymbol{x}_i)$



On the figure $x_i \in \mathbb{R}$ is one dimensional.

Linear model

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

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

We choose $f(\boldsymbol{x})$ to be a linear function

$$\begin{aligned} f_{\boldsymbol{w}}(\boldsymbol{x}_i) &= w_0 + w_1 x_{i1} + w_2 x_{i2} + ... + w_D x_{iD} \\ &= w_0 + \boldsymbol{w}^T \boldsymbol{x}_i \end{aligned}$$

Here w_0 is called the bias or offset term.

Absorbing the bias term

The linear function is given by

$$\begin{aligned} f_{\boldsymbol{w}}(\boldsymbol{x}) &= w_0 + w_1 x_1 + w_2 x_2 + ... + w_D x_D \\ &= w_0 + \boldsymbol{w}^T \boldsymbol{x} \end{aligned}$$

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 $f_{\boldsymbol{w}}(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}$.

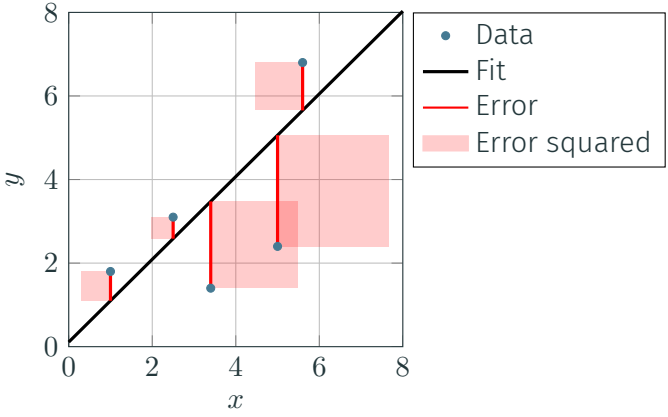
To unclutter the notation, we will assume the bias term is always absorbed and write $\boldsymbol{w}$ and $\boldsymbol{x}$ instead of $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{x}}$.

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

The least squares loss

Standard choice to measure the error between our model (parametrized by $\boldsymbol{w}$) and the observed data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ is the **least squares** loss.

$$\begin{aligned}\mathcal{L}_{\text{LS}}(\boldsymbol{w}) &= \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2\end{aligned}$$



The factor $\frac{1}{2}$ is for later convenience.

Minimizing the loss

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

$$\begin{aligned}\boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \\ &= \arg\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{x}_i^T \boldsymbol{w} - y_i)^2\end{aligned}$$

By stacking the observations $\boldsymbol{x}_i$ as rows of a matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T \in \mathbb{R}^{N \times D}$ and the targets y_i into a vector $\boldsymbol{y} = (y_1, \ldots, y_N) \in \mathbb{R}^N$.

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

We omit the LS subscript in $\mathcal{L}_{\mathrm{LS}}(\boldsymbol{w})$ when it is clear from the context.

Optimal solution

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

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}) &= \nabla_{\boldsymbol{w}}\frac{1}{2}(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}) \\ &= \nabla_{\boldsymbol{w}}\frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}-2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y}+\boldsymbol{y}^T\boldsymbol{y}\right) \\ &= \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}-\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

See Equations (69), (81) from The Matrix Cookbook for details.

Optimal solution

Now set the gradient to zero and solve for $\boldsymbol{w}$ to obtain the minimizer[1]

$$\begin{aligned} \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} &\stackrel{!}{=} 0 \\ \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} &= \boldsymbol{X}^T\boldsymbol{y} \end{aligned}$$

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

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

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

[1] Because the Hessian $\nabla_{\boldsymbol{w}}\nabla_{\boldsymbol{w}}E(\boldsymbol{w}) = \boldsymbol{X}^T\boldsymbol{X}$ is positive (semi)definite.

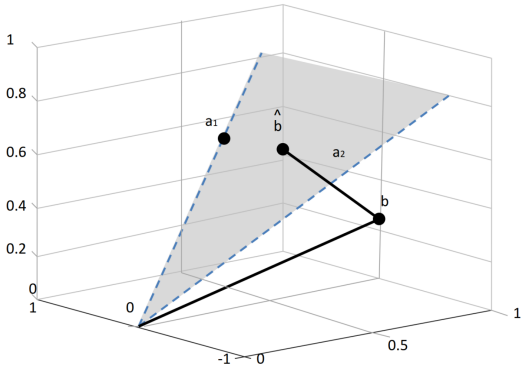
Geometric interpretation

Assume more observations than unknowns ($N > D$) – an overdetermined system.

Example $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{A} \in \mathbb{R}^{3\times 2}$.
3 equations and 2 unknowns.

The columns of $\boldsymbol{A}$, $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, define a 2D linear subspace embedded in $\mathbb{R}^3$.

$\hat{\boldsymbol{b}}$ is the orthogonal projection of the target $\boldsymbol{b}$ onto the linear subspace.



In the special case that $\boldsymbol{X} = \boldsymbol{x}$ is a column vector, the orthogonal projection of y onto the line x becomes $\boldsymbol{x}\frac{\boldsymbol{x}^T\boldsymbol{y}}{\boldsymbol{x}^T\boldsymbol{x}}$.

Algorithmic issues

The closed from ordinary least squares (OLS) solution is $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.

Possible numerical issues if $\boldsymbol{X}^T\boldsymbol{X}$ is ill-conditioned or singular leading.

Alternatives:

- Compute the pseudo-inverse using the SVD (default in sklearn)
- If $\boldsymbol{X}$ is tall and skinny ($N \gg D$) we can use QR decomposition[2]
- Iterative solvers such as conjugate gradient methods and GMRES[3]

[2] Here $\boldsymbol{w}^* = \boldsymbol{R}^{-1}\boldsymbol{Q}^T\boldsymbol{y}$, where $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$ and $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R}$ is upper triangular.

[3] Suitable when $\boldsymbol{X}$ is sparse or structured since they only need the ability to perform matrix-vector multiplications.

Outline

Simple linear regression

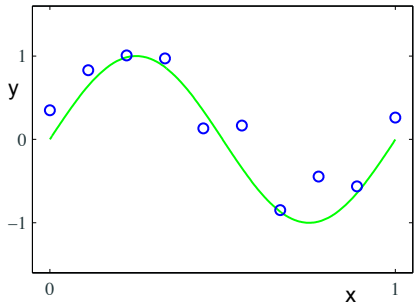
Non-linear relationships

Probabilistic Interpretation

Nonlinear dependency in data

What if the dependency between y and $\boldsymbol{x}$ is not linear?

Example 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 a 1-dimensional x we can define f as

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

or more generally for any basis function $\phi_j(x)$

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

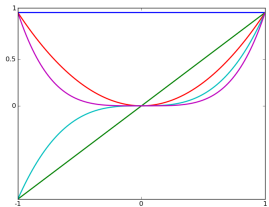
Absoribing the bias term by defining $\phi_0 = 1$

$$f_{\boldsymbol{w}}(x) = \boldsymbol{w}^T \boldsymbol{\phi}(x) = \boldsymbol{w}^T \left[\phi_0(x), \phi_1(x), \ldots, \phi_M(x)\right]$$

The function f is still linear in $\boldsymbol{w}$ despite being potentially non-linear in x!

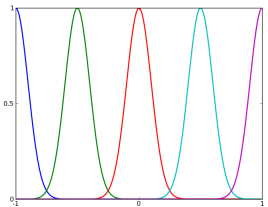
Typical basis functions

Polynomials



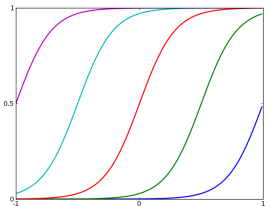
$$\phi_j(x) = x^j$$

Gaussian



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

Logistic Sigmoid



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

where $\sigma(a) = \frac{1}{1+\exp^{-a}}$ is the sigmoid, and μ_j and s are hyperparameters.

Linear basis function model

For D-dimensional data $\boldsymbol{x}$, $\phi_j : \mathbb{R}^D \to \mathbb{R}$ and we have

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

Using the same least squares error function as before

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2} \sum^{N} \left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i \right)^2 = \frac{1}{2} (\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})$$

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

$\boldsymbol{\Phi}$ is called the **design matrix** of $\boldsymbol{\phi}$.

Optimal solution

Recall least squares loss for the original feature matrix $\boldsymbol{X}$

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

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

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}).$$

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

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

Compare this to Equation 1: $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^\dagger\boldsymbol{y}$

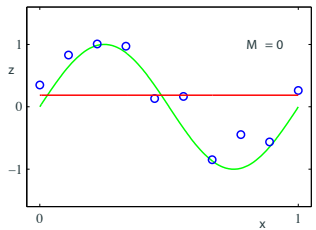
Only thing we've done is transformed our original features $\boldsymbol{X}$ into new features $\boldsymbol{\Phi}$.

Choosing the degree of the polynomial

How do we choose the degree of the polynomial M?

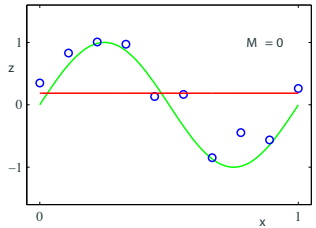
Choosing the degree of the polynomial

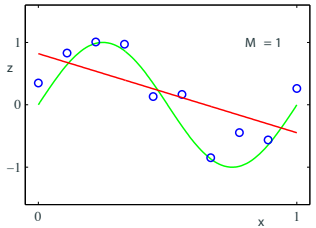
How do we choose the degree of the polynomial M?



Choosing the degree of the polynomial

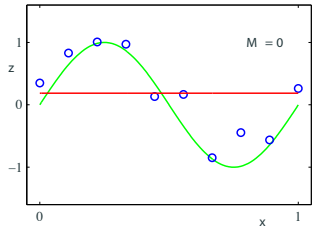
How do we choose the degree of the polynomial M?

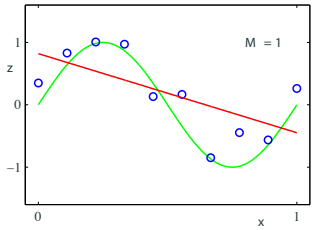


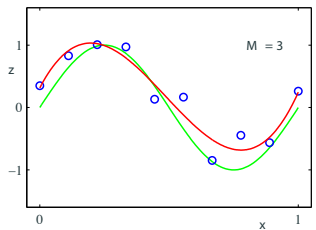


Choosing the degree of the polynomial

How do we choose the degree of the polynomial M?

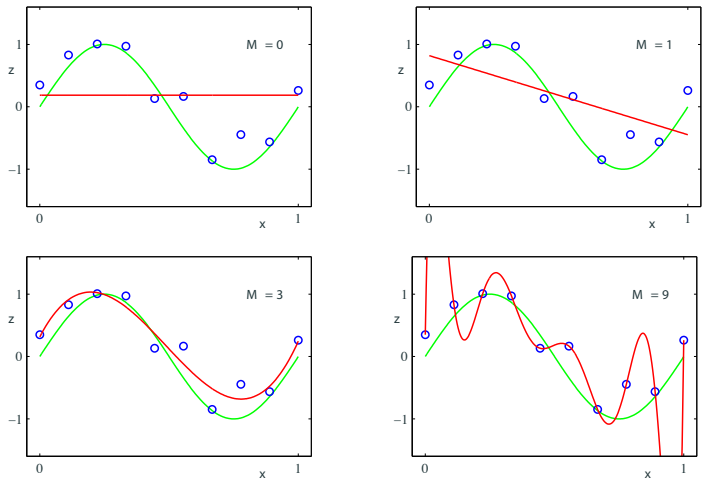






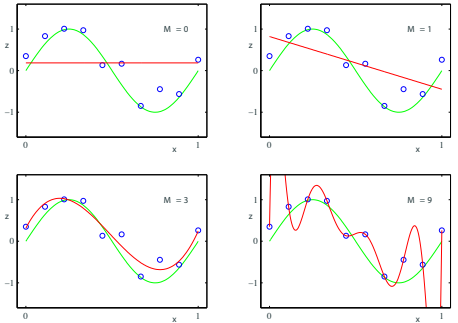
Choosing the degree of the polynomial

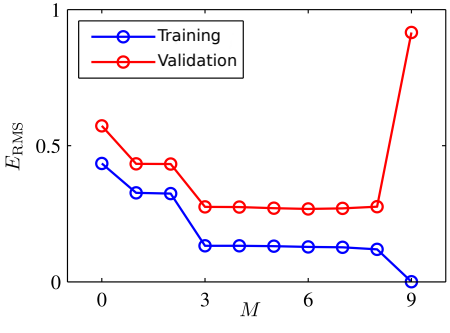
How do we choose the degree of the polynomial M?



Choosing the degree of the polynomial

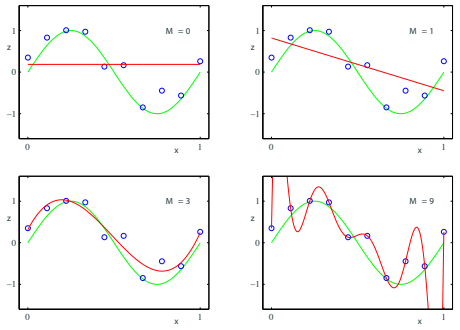
One valid solution is to choose M using the train-validation split.





Manifestation of overfitting

Observation: overfitting occurs when the coefficients $\boldsymbol{w}$ become large.



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

What if we penalize large weights?

Controlling overfitting with regularization

Least squares loss with **L2 regularization**, also called **ridge regression**

$$\mathcal{L}_{\text{ridge}}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i\right]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{3}$$

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

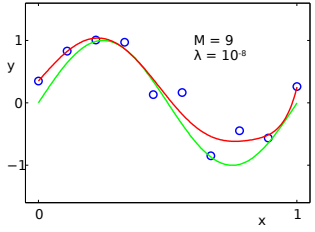
Controlling overfitting with regularization

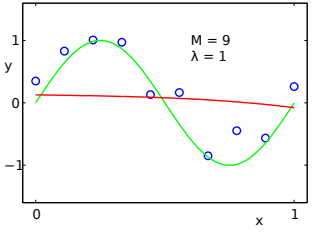
Least squares loss with **L2 regularization**, also called **ridge regression**

$$\mathcal{L}_{\text{ridge}}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) - y_i\right]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{3}$$

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

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





Bias-variance trade-off

Increasing the degree of the polynomial M

Increasing the regularization strength λ

Bias-variance trade-off

Increasing the degree of the polynomial M

decreases bias – the function is more flexible

Increasing the regularization strength λ

Bias-variance trade-off

Increasing the degree of the polynomial M

- decreases bias – the function is more flexible
- increases variance – we can more easily fit the noise

Increasing the regularization strength λ

Bias-variance trade-off

Increasing the degree of the polynomial M

- decreases bias – the function is more flexible
- increases variance – we can more easily fit the noise

Increasing the regularization strength λ

- increases bias

Bias-variance trade-off

Increasing the degree of the polynomial M

- decreases bias – the function is more flexible
- increases variance – we can more easily fit the noise

Increasing the regularization strength λ

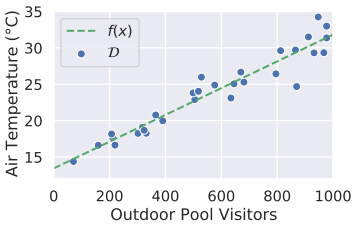
- increases bias
- decreases variance

Intepretation and correlation vs. causation

The weight w_d can be interpreted as the strength of the (linear) relationship between feature x_d and y.

Least squares fit $f(x) = 0.018x + 13.43$.

A weight of 0.018 shows a strong correlation (considering the different scales).



Correlation $\not\Rightarrow$ causation: Increasing pool visitors doesn't increase temperature.

If you normalize the data to handle the different scales of $\boldsymbol{x}$ and $\boldsymbol{y}$ (as we often do in practice) we find a weight of about 1.

Outline

Simple linear regression

Non-linear relationships

Probabilistic Interpretation

Maximum likelihood

Assuming Gaussian likelihood (with a fixed precision β) for a single sample

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

and assuming that the samples are drawn independently, the likelihood of the entire dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N = \{\boldsymbol{X}, \boldsymbol{y}\}$ is

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

We use the same approach as previously – maximize the likelihood w.r.t. $\boldsymbol{w}$ and β

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

Maximum likelihood

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

$$\begin{aligned}\boldsymbol{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}} &= \arg\max_{\boldsymbol{w},\beta} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \\ &= \arg\max_{\boldsymbol{w},\beta} \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \\ &= \arg\min_{\boldsymbol{w},\beta} -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)\end{aligned}$$

We need to minimize the negative log-likelihood which is our loss

$$\mathcal{L}_{\mathrm{NLL}}(\boldsymbol{w}, \beta) = -\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)$$

Simplifing the loss

$$
\begin{aligned}
\mathcal{L}_{\text{NLL}}(\boldsymbol{w}, \beta) &= -\ln \left[\prod_{i=1}^{N} \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \right] \\
&= -\ln \left[\prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \\
&= -\sum_{i=1}^{N} \ln \left[\sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \\
&= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi
\end{aligned}
$$

Minimizing the negative log-likelihood w.r.t. $\boldsymbol{w}$

$$
\begin{aligned}
\boldsymbol{w}_{\mathrm{ML}} &= \arg\min_{\boldsymbol{w}} \mathcal{L}_{\mathrm{NLL}}(\boldsymbol{w}, \beta) \\
&= \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \underbrace{- \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi}_{=\text{const}} \right] \\
&= \arg\min_{\boldsymbol{w}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2}_{\text{least squares loss}} \\
&= \arg\min_{\boldsymbol{w}} \mathcal{L}_{\mathrm{LS}}(\boldsymbol{w})
\end{aligned}
$$

Maximizing the likelihood is equivalent to minimizing the least squares loss!

Thus, we have $\boldsymbol{w}^* = \boldsymbol{w}_{\mathrm{LS}} = \boldsymbol{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y} = \boldsymbol{\Phi}^\dagger \boldsymbol{y}$

Minimizing the negative log-likelihood w.r.t. β

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

$$\begin{aligned}\beta_{\mathrm{ML}} &= \arg\min_{\beta} E_{\mathrm{ML}}(\boldsymbol{w}_{\mathrm{ML}}, \beta) \\ &= \arg\min_{\beta} \left[\frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}_{\mathrm{ML}}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \right]\end{aligned}$$

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

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

Solving for β

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

Posterior distribution

Since MLE can overfit with little data consider the posterior distribution instead.

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

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

Posterior distribution

Since MLE can overfit with little data consider the posterior distribution instead.

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

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.:	$\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$	$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)$	$p(\boldsymbol{w} \mid \cdot)$	$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \beta, \cdot)$

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

Choosing the prior for $\boldsymbol{w}$

Isotropic multivariate normal distribution with zero mean as prior for $\boldsymbol{w}$

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

where α is the precision, and M is the dimension.

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

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

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

$$\begin{aligned}\mathcal{L}_{\text{MAP}}(\boldsymbol{w}) &= - \ln p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \\ &= - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) + \text{const}\end{aligned}$$

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

Simplify the MAP loss function

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

MAP estimation with Gaussian prior is equivalent to ridge regression!

Full Bayesian approach

Full posterior $p(\boldsymbol{w} \mid \mathcal{D}) \propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)\, p(\boldsymbol{w} \mid \alpha)$ instead of a point estimate $\boldsymbol{w}_{\text{MAP}}$. Since both likelihood and prior are Gaussian, the posterior is Gaussian as well![4]

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

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

Observations

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

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

Predictions 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 $\hat{y}_{\text{new}}$ for a new data point $\boldsymbol{x}_{\text{new}}$ - 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 $\hat{y}_{\text{new}}$ for new data $\boldsymbol{x}_{\text{new}}$.

Maximum likelihood given $\boldsymbol{w}_{\text{ML}}$ and β_{ML}:

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

Maximum a posteriori given $\boldsymbol{w}_{\text{MAP}}$:

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

For MAP we assume β to be known a priori to simplify the calculations.

Predictions 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 $\hat{y}_{\text{new}}$ for a new data point $\boldsymbol{x}_{\text{new}}$ - 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 $\hat{y}_{\text{new}}$ for new data $\boldsymbol{x}_{\text{new}}$.

Maximum likelihood given $\boldsymbol{w}_{\text{ML}}$ and β_{ML}:

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

Maximum a posteriori given $\boldsymbol{w}_{\text{MAP}}$:

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

For MAP we assume β to be known a priori to simplify the calculations.

Posterior predictive distribution

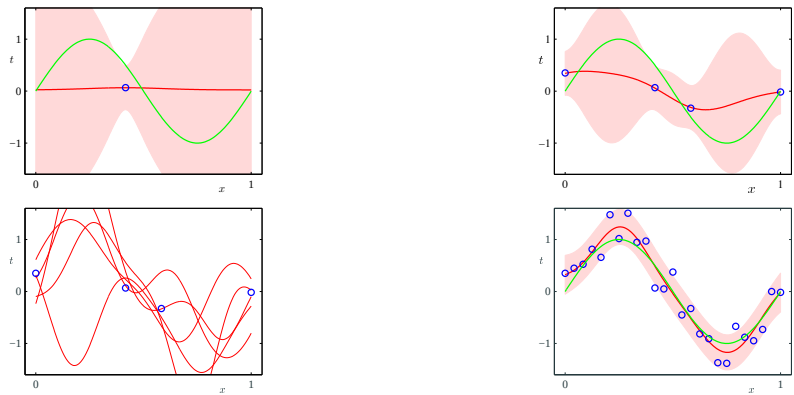
Alternatively, we can use the full posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$.

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

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

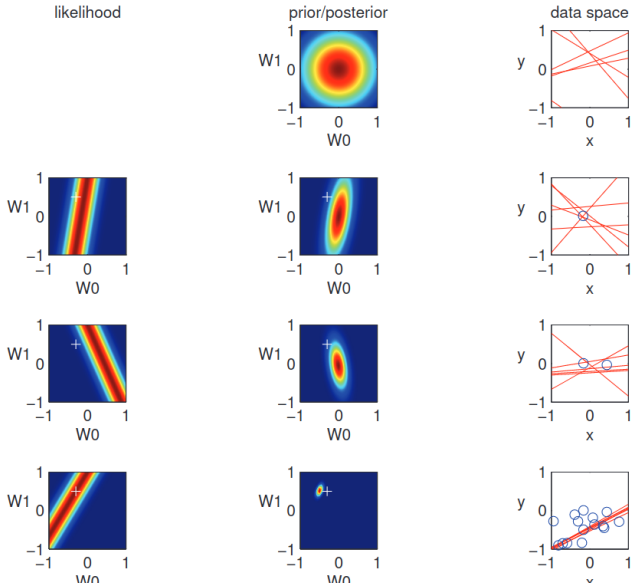
Advantage: We get a data-dependent uncertainty estimate – the variance of the Gaussian now also depends on the input $\boldsymbol{x}_{\text{new}}$.

Example of posterior predictive distribution



True function (green), observations (blue), mode (dark red), variance (light red).

Sequential Bayesian updating



Summary

Optimization-based approaches to regression have probabilistic interpretations:

- Least squares regression $\iff$ Maximum likelihood (Slide 27)
- Ridge regression $\iff$ Maximum a posteriori (Slide 32)

Even non-linear dependencies can be captured by a model linear w.r.t. weights $\boldsymbol{w}$ by applying a non-linear transformation to the input features (Slide 13).

Penalizing large weights helps to reduce overfitting (Slide 20).

Full Bayesian gives us data-dependent uncertainty estimates (Slide 35).

Reading material

Main reading

”Pattern Recognition and Machine Learning” by Bishop
[ch. 1.1, 3.1 – 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, C. Osendorfer, and S. Günnemann. Some figures are from Bishop’s “Pattern Recognition and Machine Learning”.