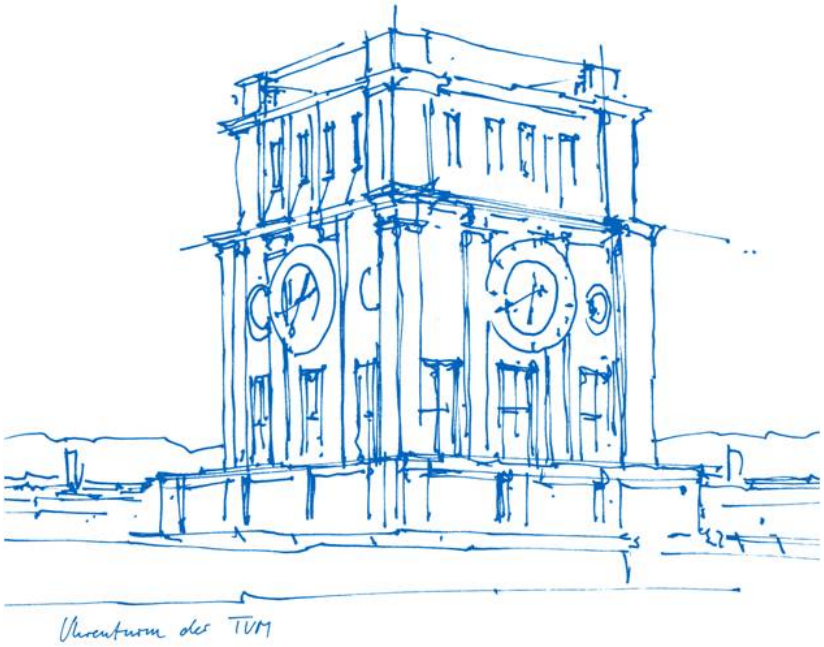
Exercises for Social Gaming and Social Computing (IN2241 + IN0040) – Introduction to Exercise 4



Exercise Content

Sheet Number	Exercise	Working Time
1	• Introduction to Python: basic Python programming language exercises • Graph Drawing using igraph	Monday, May 27 - Monday, June 3, 24:00
2	• Centrality measures	Monday, June 3 - Monday, June 17, 24:00
3	• Recommender Systems as an example for systems using simple forms of social context: Collaborative Filtering	Monday, June 17 - Monday, June 24, 24:00
4	• Inferring social tie strength from activity data in social networking platforms with linear regression	Monday, June 24- Monday, July 01, 24:00
5	• finding social groups with clustering methods on profile date (K-Means) and graph clustering (Girvan Newman method)	Monday, July 01 - Monday, July 08, 24:00
6	• analyzing short-term social context using mobile interaction data (Reality Mining)	Monday, July 08 - Monday, July 15, 24:00

Background on Linear Regression

- The slides are based on [3]
- Background reading: [2]

< open the pdf with the background slides >

Predicting Tie Strength

- input („predictive“) variables x:
  - Number of friends (I)
  - Friends' number of friends (I)
  - Days since last communication (I)
  - Appearance together in photos (I)
  - Wall intimacy words (I)
  - Inbox intimacy words (I)
  - Days since first communication (D)
  - Number of mutual friends (S)
  - Age distance (SD)
  - Educational distance (SD)

  (I) being **intimacy** variables, (D) **duration** variables, (S) **structural** variables and (SD) **social distance** variables.

- output („dependent“) variable y:

  tie strength

Predicting Tie Strength

- model: (linear) linear regression:

$$y_i = \alpha + \beta \boldsymbol{X}_i + \epsilon_i$$

- dataset: SocialGraph.gml with
  - input data X (incomplete → please complete ☺) for each edge and
  - ground truth tie strength y (for 0.7 of the edges → predict rest 0.3 of edges)

Problem 4.1: Completing the X for all Edges of the Social Graph

Problem 4.1: Completing The Social Graph

If you inspected the *SocialGraph.gml* file, you may have observed that some of the ten predictive variables are already present as node or edge attributes while others are not. In order to later compute the linear model with these ten predictive variables, the missing values are needed. Luckliy, they can be calculated from the already existing ones. **Therefore, your first task is to complete the social graph by calculating all the missing nodes/edges attributes.** Save the vectors with known tie strength as training data in a seperate dataframe as the unknown ones (since those are used for prediction later). Output the first ten entries of the training table.

Notes:

- It is recommended to first work with lists when computing the missing variables and converting it into a pandas dataframe afterwards. Look up the function `Dataframe()` in the manual for creating a one.
- Use `head(n)` on a dataframe to output the first n entries.
- Pay attention to the data types of the features.

Problem 4.2: Predicting Tie Strengths

Task 1: Log-Transformation

In data analysis, transformation is the replacement of a variable by a function of that variable. For example, replacing a variable x by its square root $\sqrt{x}$ or its logarithm $\log x$. In a stronger sense, it means a replacement that changes the shape of a distribution or relationship. There are many reasons for transformations. The following list are a few of them but it is not comprehensive:

1. Convenience
2. Reducing skewness
3. Equal spreads
4. Linear relationships
5. Additive relationships

If you are looking at just one explanatory variable, points 1, 2 and 3 are relevant. But concerning two or more variables, points 4 and 5 are more important. However, transformations that achieve 4 and 5 very often fulfill 2 and 3 as well. In our case, a logarithm transformation on the predictibe variables will be used.

With the help of numpy's `log` function, **apply the log-transformation on each feature vector for the training table (but not the tie strength)**. Pay attention to the fact that $\log 0$ is not defined, a small number (e.g. 0.000001) should be added before the transformation! Again, output the first ten entries of your dataframe.

Problem 4.2: Predicting Tie Strengths

Task 2: Building The Regression Model

Finally, the regression can be applied on the dataframe. For this purpose, split the dataframe into the target and predictive variables into variables y and X. A constant term, corresponding to the bias α in the mathematical definition above, needs to be added to the regression before its computation. It represents the target value when all explanatory variables are zero - think of it as a baseline. If there was no intercept, the model would be less meaningful. You can add the intercept with `add_constant(X)`.

Now apply a multiple linear regression on the training table, the statsmodels functions `OLS()` and `fit()` will help you with that. Output the summary with the help of `summary()`. **State what the coefficients tell you about the influence of the respective features for predicting the tie strength. Which kind of variables (I, D, S, SD) have the most influence? Does it match the findings of the paper referenced in [1]?** Note that the coefficients do not need to be high or within the [-1,1] range, but can be. Take into account other statistics for your evaluations. **For example, what does the (adjusted) R-squared value say about the prediction?** Do not write more than 7 sentences overall.

R-squared:

- model makes predictions $\{f_i\}_{\text{i}\in[1:N]}$ while the ground thruths are $\{y_i\}_{\text{i}\in[1:N]}$
- empirical variance of the ground truths: $SS_{tot} = \frac{1}{N}\sum_i (y_i - \bar{y})^2$
- squared error or predictions (sum of residuals): $SS_{res} = \frac{1}{N}\sum_i (y_i - f_i)^2$
- definition: $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$
- interpretation:
  R^2 equals the fraction of the variance of the y_i that is explained by the model,
  1-R^2 equals the fraction of the variance of the y_i that is not explained by the model,

Problem 4.2: Predicting Tie Strengths

Task 3: Prediction of Tie Strengths

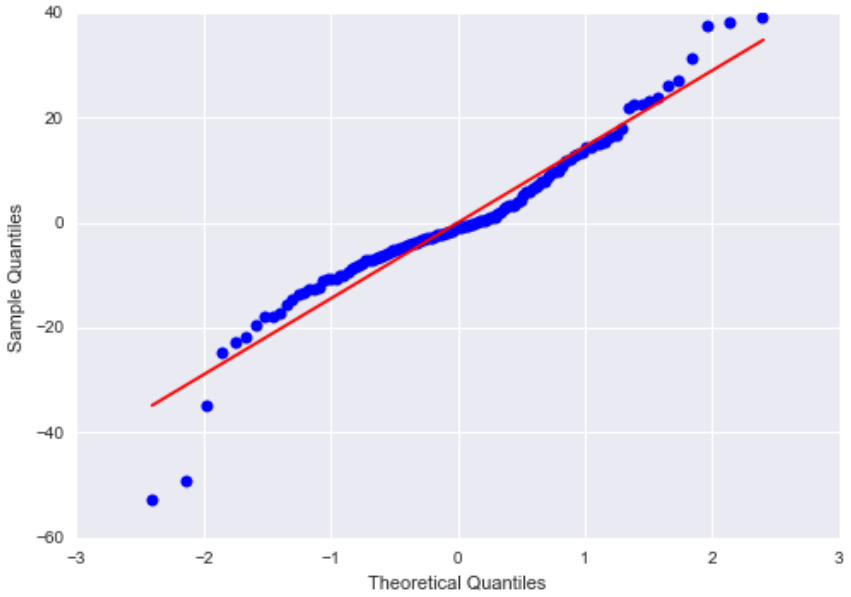
As a last step, the missing values (marked with -1) for some tie strength entries should be predicted using the before computed regression model. **Use the regression model to predict the missing tie strength values.** Statsmodels will be of help with that. Remember that we transformed the training data with a log-transform and added an intercept, so this needs to be done here as well. Output the first ten entries.

Are the predictions in line with the observations above? Pick a few entries to back up your observations. If you would like to talk about other than the first ten entries, you can query a pandas dataframe similar to SQL. More information on how to do this is available in the pandas documenation. Do not write more than 3 sentences.

Note: Values outside the realistic range in the interval of (0,1) for the tie strength might exist because of very low/high values for some predictive variables in the dataset. You can consider them as almost 0/1.

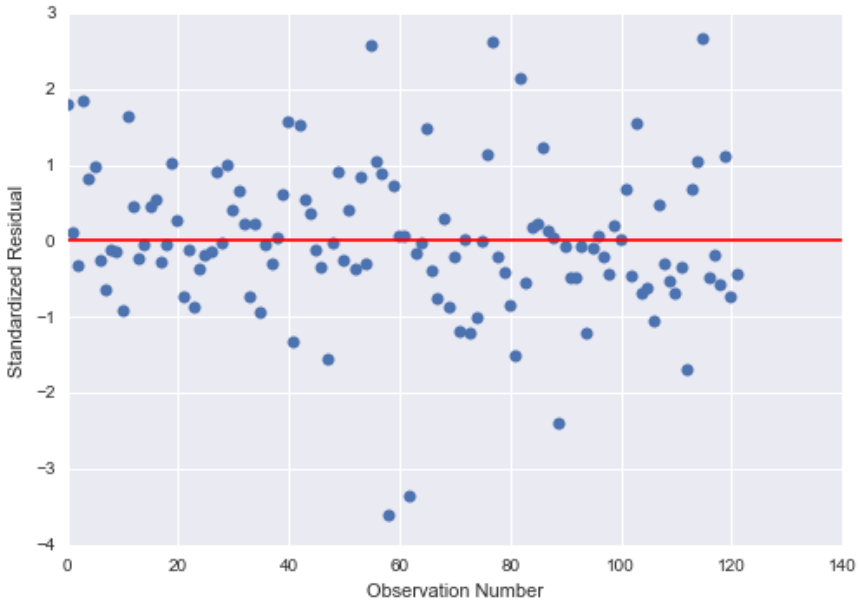
(Optional) Problem 4.3: Goodness of Fit

- **QQ Plot**:
  compare two distributions
  (here: $\{f_i\}_{i\in[1:N]}$ and $\{y_i\}_{i\in[1:N]}$)
  by plotting quantiles against each other



[4]

- **Residuals Plot**



[4]

Submitting Your Solution

- work by expanding the .ipynb iPython notebook for the exercise that you downloaded from Moodle.
- save your expanded .ipynb iPython notebook in your working directory.
  Submit your .ipynb iPython notebook via Moodle (nothing else)
- remember: working in groups is not permitted.
  Each student must submit her own ipynb notebook!
- we check for plagiarism. Each detected case will have the consequence of 5.0 for the whole exercise grade.
- deadline: please check slide nr 2 (this slide-set)

Citations

[1] E. Gilbert and K. Karahalios: *Predicting Tie Strength With Social Media*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009.

[2] C. Bishop: *Pattern Recognition and Machine Learning*. 2006, chapter 3

[3] Lecture Machine Learning I, TUM Informatics (Winter 2017)

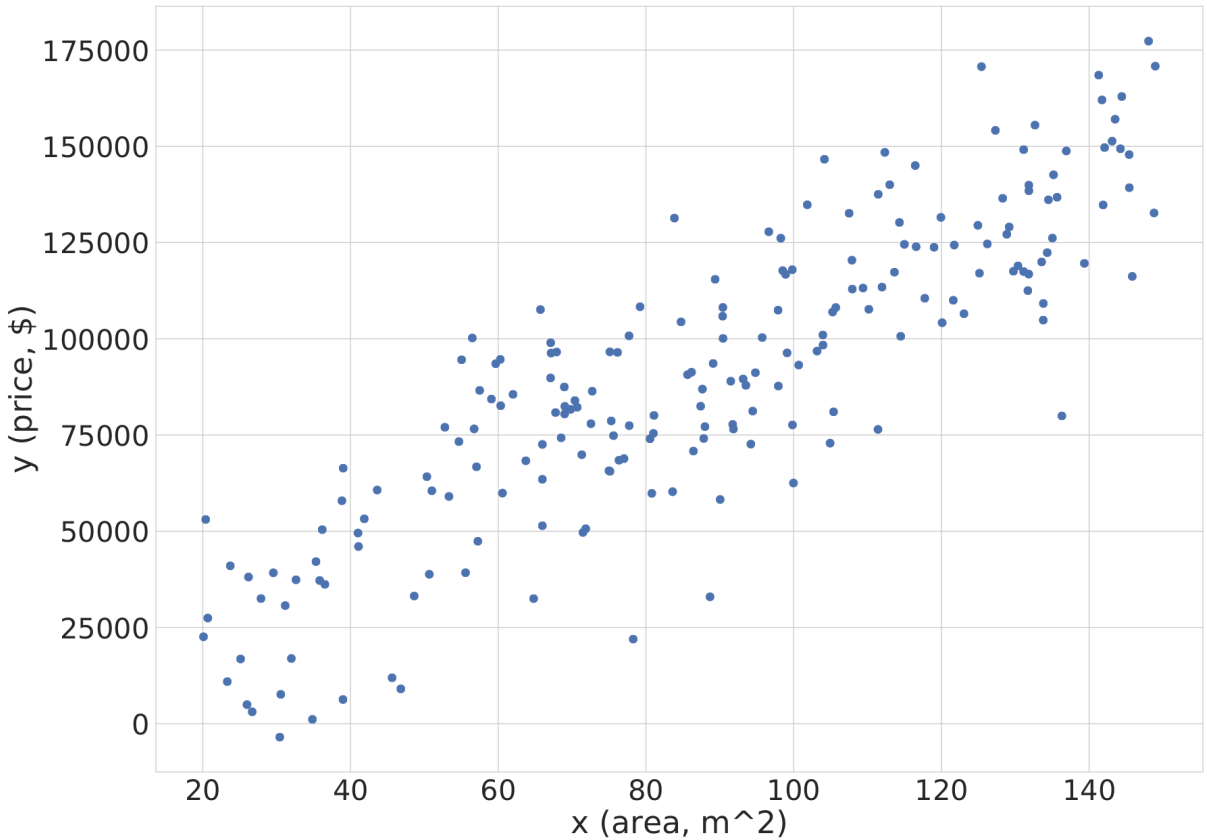
[4] https://degninou.net/2016/02/04/multiple-regression-and-diagnostics-with-python/

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.

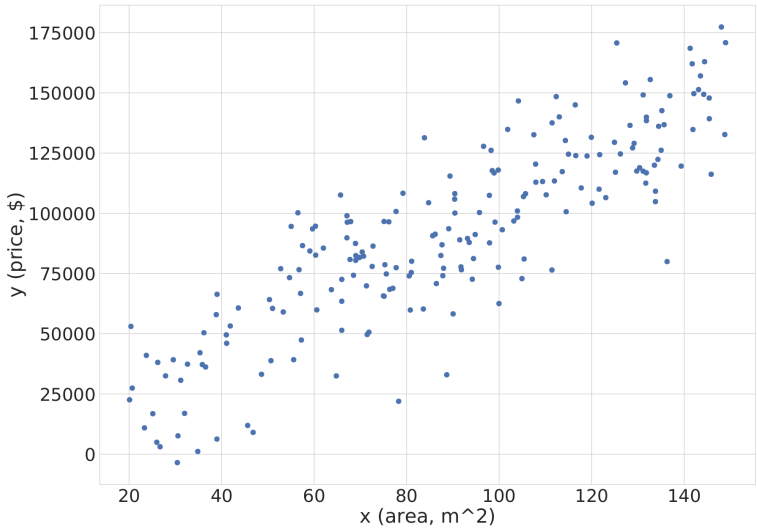


How do we estimate a price of a new house with area x_{new}?

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

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

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

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

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

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

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?

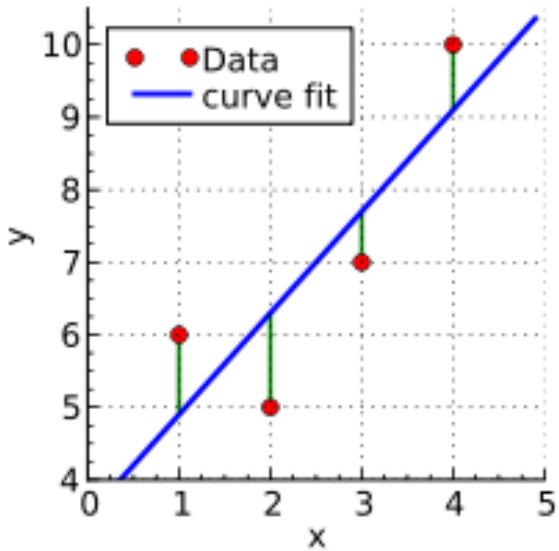
Error function

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

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

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



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

Objective

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

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

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

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

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

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

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

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

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]

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

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

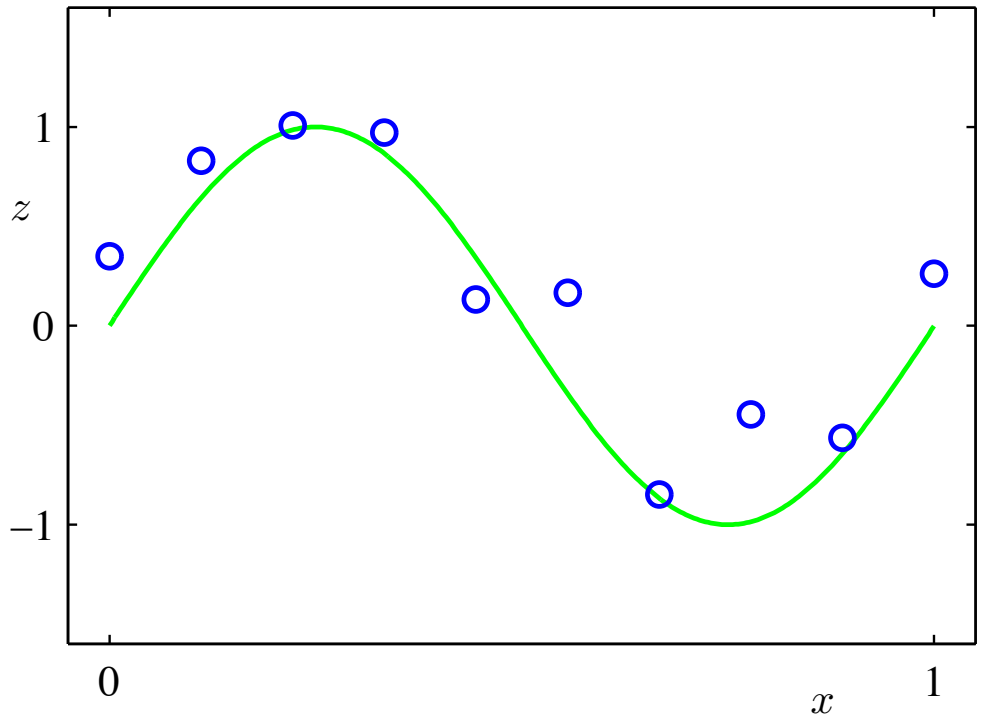
$$\boldsymbol{w}^* = \underbrace{(\boldsymbol{X}^\mathsf{T} \boldsymbol{X})^{-1} \boldsymbol{X}^\mathsf{T}}_{=\boldsymbol{X}^\dagger} \boldsymbol{y} \tag{15}$$

$\boldsymbol{X}^\dagger$ 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 $\boldsymbol{x}$ 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

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

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

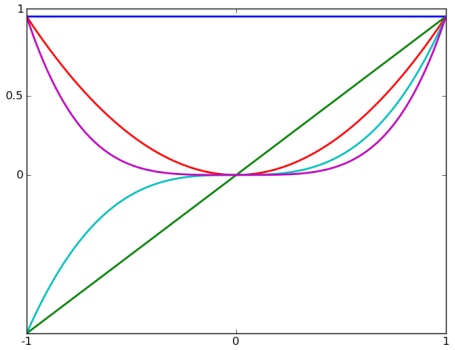
Define $\phi_0 = 1$

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

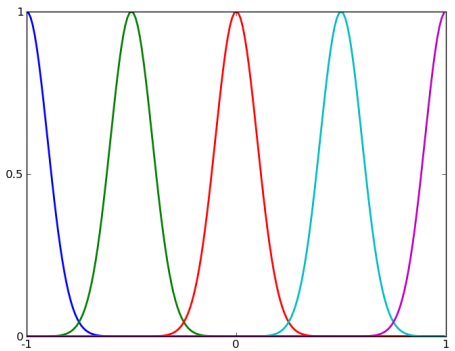
The function f is still linear in $\boldsymbol{w}$ (despite not being linear in $\boldsymbol{x}$)!

Typical basis functions

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

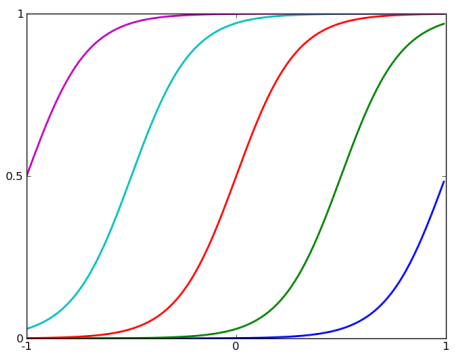


Gaussian $\phi_j(x) = \mathrm{e}^{\frac{-(x-\mu_j)^2}{2s^2}}$



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:

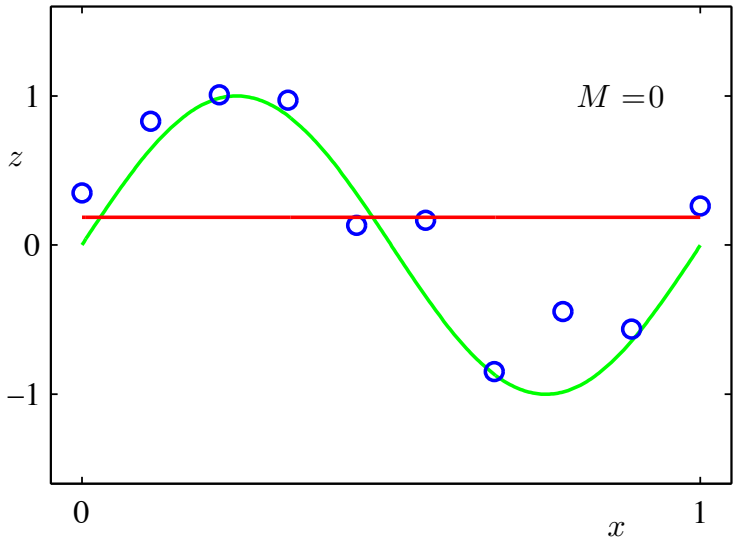
$$\boldsymbol{w}^* = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} \tag{25}$$

Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

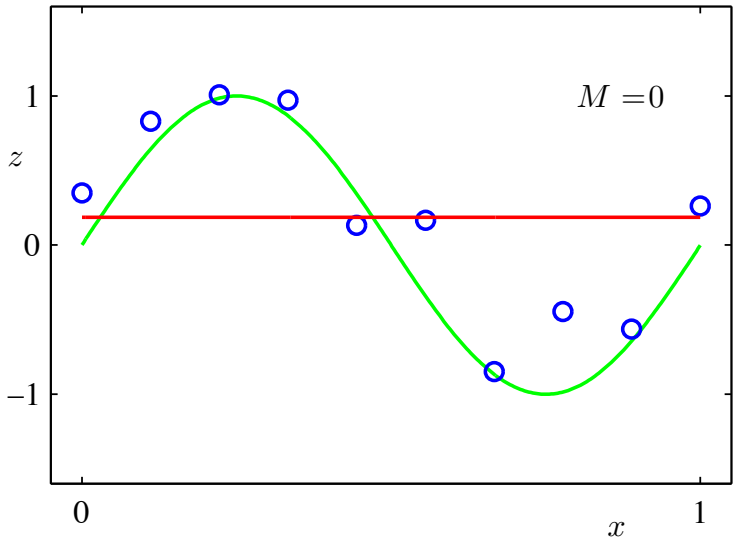
Choosing degree of the polynomial

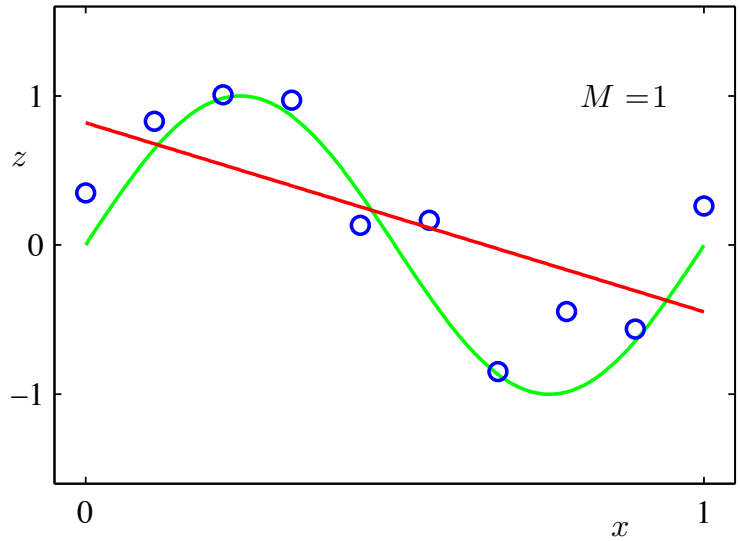
How do we choose the degree of the polynomial M?



Choosing degree of the polynomial

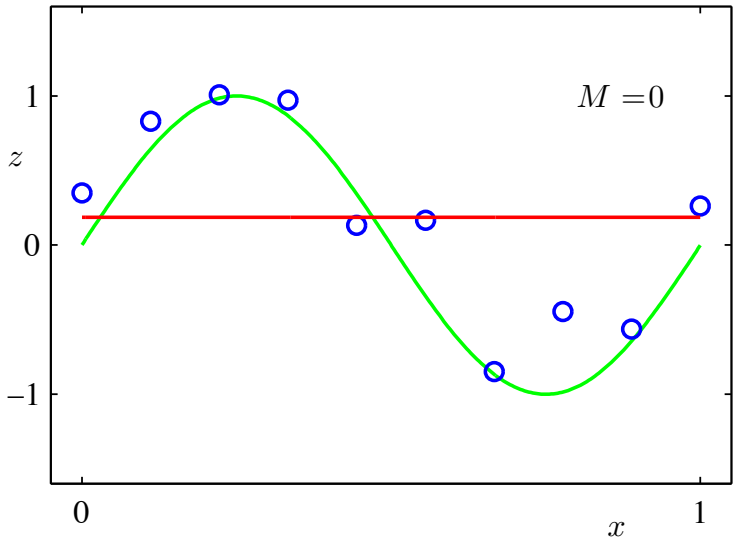
How do we choose the degree of the polynomial M?

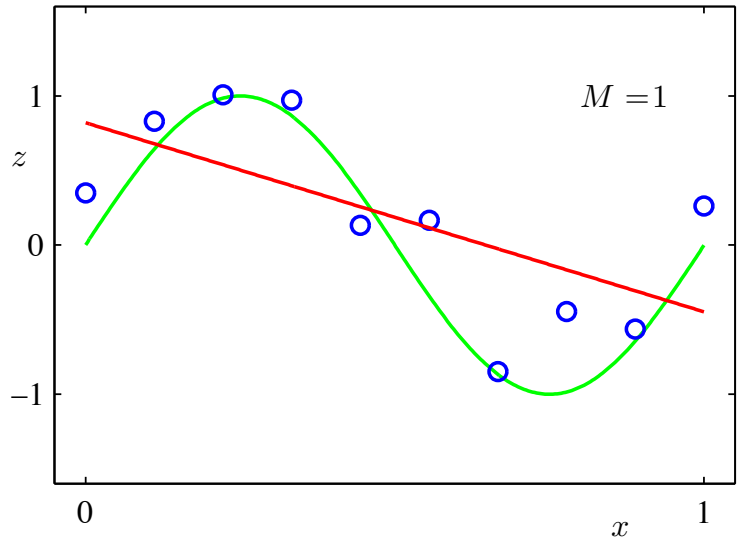


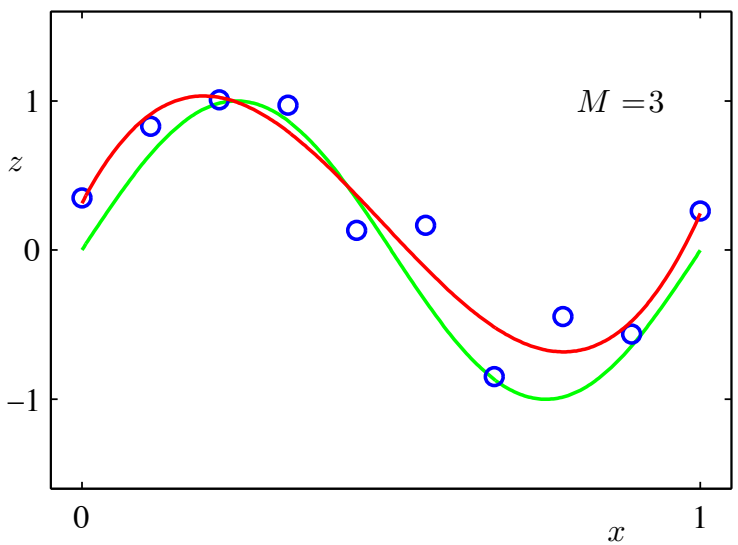


Choosing degree of the polynomial

How do we choose the degree of the polynomial M?

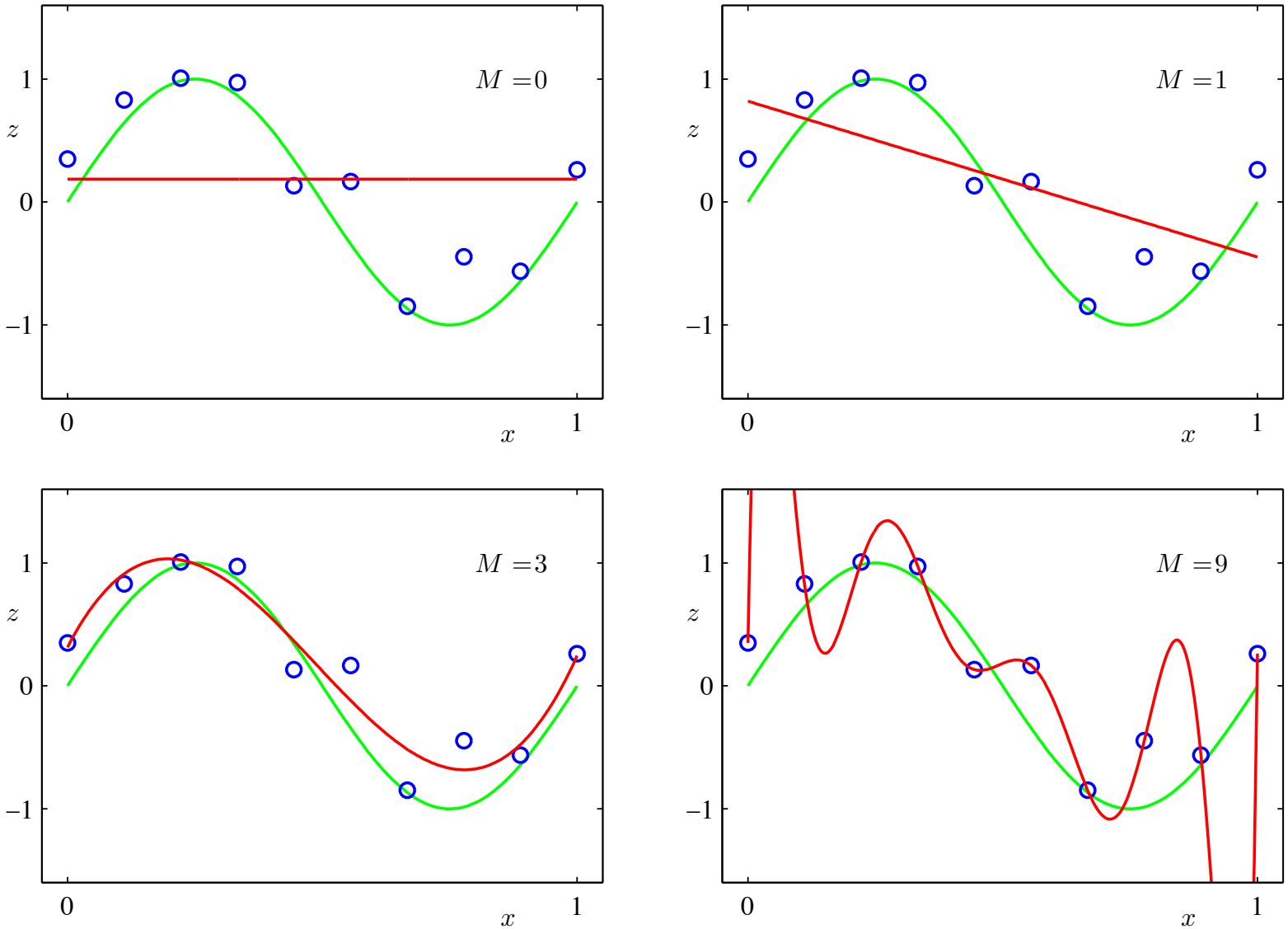




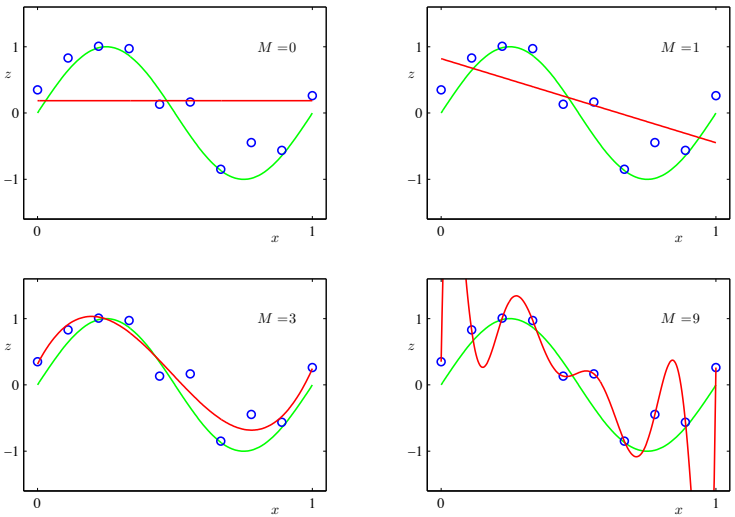


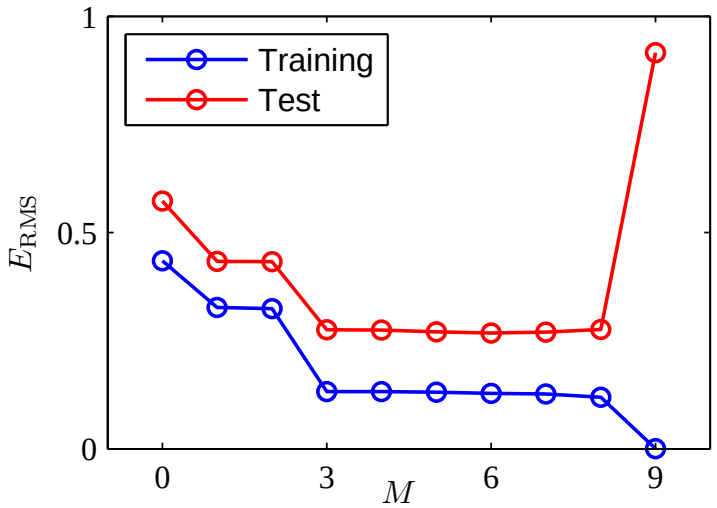
Choosing degree of the polynomial

How do we choose the degree of the polynomial M?



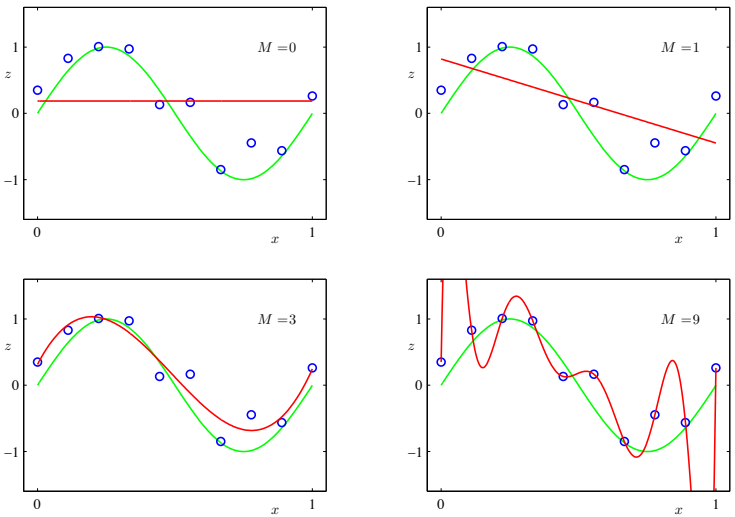
Choosing degree of the polynomial





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

Choosing degree of the polynomial



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

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

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

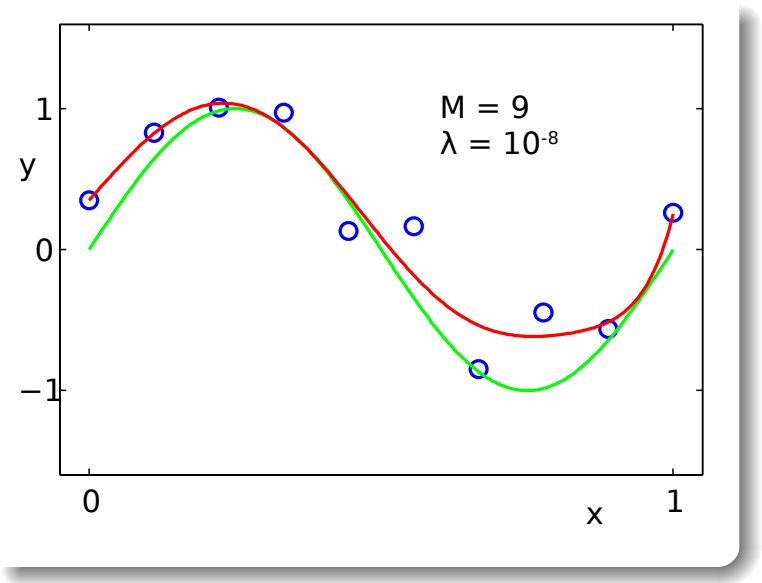
Controlling overfitting with regularization

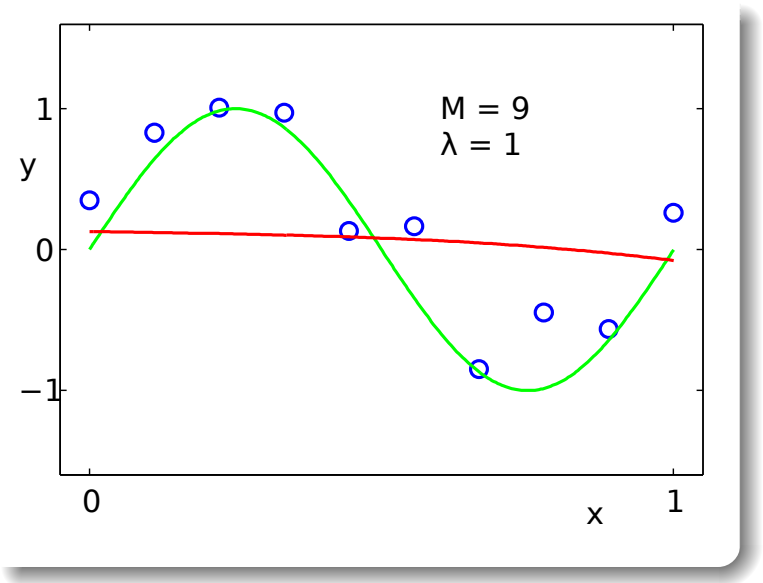
Least squares loss function with L2 regularization
(also called ridge regression)

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

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 $\boldsymbol{w}$

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

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

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

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

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

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

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

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

Maximum likelihood

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

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

$$= \arg\max_{\boldsymbol{w},\beta} \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) \tag{31}$$

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

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

$$E_{\mathrm{ML}}(\boldsymbol{w}, \beta) = -\ln \left[\prod_{i=1}^{N} \mathcal{N}(y_i \mid f_{\boldsymbol{w}}(\boldsymbol{x}_i), \beta^{-1}) \right] \tag{34}$$

$$= -\ln \left[\prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 \right) \right] \tag{35}$$

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

$$= \frac{\beta}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \tag{37}$$

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. $\boldsymbol{w}$

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

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

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

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

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

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

Optimizing log-likelihood w.r.t. β

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

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

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

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

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

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

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

where,

α - precision of the distribution

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

Motivation:

- Higher probability is assigned to small values of $\boldsymbol{w}$
  $\Longrightarrow$ 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

$$\boldsymbol{w}_{\text{MAP}} = \arg\max_{\boldsymbol{w}} \; p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) \tag{52}$$

$$= \arg\max_{\boldsymbol{w}} \; \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) + \ln p(\boldsymbol{w} \mid \alpha) - \underbrace{\ln p(\boldsymbol{X}, \boldsymbol{y})}_{=\text{const}} \tag{53}$$

$$= \arg\min_{\boldsymbol{w}} - \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w} \mid \alpha) \tag{54}$$

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

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

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

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

MAP error function

Simplify the error function

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

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

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

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