GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Data Literacy

Tutor : Ghassan Al-Falouji
Supervisor : Prof. Dr. Roland Mandl

AUFSTIEG DURCH BILDUNG
OFFENE HOCHSCHULEN

In [1]:
```
## CSS coloring for the dataframe tables

from IPython.core.display import HTML
#
css = open('../style/style-table.css').read() + open('../style/style-notebook.css').r
ead()
HTML('<style>{}</style>'.format(css))
```

Out[1]:

Introduction

Linear regression is a linear approach that allows us to analyze the relationship between a dependent variable and one or more explanatory (independent) variables.

The simplest model for linear regression is a linear combination of the elements of $\mathbf{x}$, with $\phi_j(\mathbf{x}) = x_j$ for j>0 and $\phi_0(\mathbf{x}) = 1$,

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_D \cdot x_D$$

where $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$.

A model often presented in literature is the linear regression of one-dimensional observations using a regression line. Here, the dimensionality D equals one, and, therefore, $\mathbf{x} = (1, x_1)^{\mathrm{T}}$, resulting in the equation of a line:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 \cdot x_1$$

Simple Linear Regression

- Least Square Method as quality quality function

For fitting the parameters $\mathbf{w} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ of a function $y(\mathbf{x}, \mathbf{w})$ to a data set of N observations $\{\mathbf{x}_n\}$ and their corresponding target values $\{t_n\}$, one can apply the least squares method. It aims to find the values for $\mathbf{w}$ that minimize the sum of squared distances between the measured target values $\{t_n\}$ and the respective predicted target values $\{\hat{t}_n\} = \{y(\mathbf{x}_n, \mathbf{w})\}$.

$$S(\mathbf{w}) = \sum_{n=1}^{N} \left(\hat{t}_n - t_n \right)^2$$

Univariate Linear Regression with LSE optimization

With the regression line we have to determine the values for the two parameters w_0 and w_1 that minimize the sum of the squared distances. The input vector $\mathbf{x}$ has only one element, x_1. To avoid unnecessary clutter in the equations, we rename x_1 to x in this section.

$$\begin{aligned}\min\ S(\mathbf{w}) &= \min \sum_{n=1}^{N} (\hat{\mathbf{t}}_n - t_n)^2 \\ &= \min \sum_{n=1}^{N} (w_0 + w_1 \cdot x_n - t_n)^2\end{aligned}$$

For this we set the partial differential equations $\frac{\partial S}{\partial w_0}$ and $\frac{\partial S}{\partial w_1}$ equal to zero:

$$\begin{aligned}\frac{\partial S}{\partial w_0} &= -2 \cdot \sum_{n=1}^{N} (t_n - w_0 - w_1 \cdot x_n) &&= 0 \\ \frac{\partial S}{\partial w_1} &= -2 \cdot \sum_{n=1}^{N} x_n (t_n - w_0 - w_1 \cdot x_n) &&= 0\end{aligned}$$

We now find the values of w_0 and w_1 for wich the partial differential equations become zero, which we name w_0^{opt} and w_1^{opt}:

$$\begin{aligned}\sum_{n=1}^{N} (t_n - w_0^{opt} - w_1^{opt} x_n) &= 0 \\ \sum_{n=1}^{N} (t_n x_n - w_0^{opt} x_n - w_1^{opt} x_n^2) &= 0\end{aligned}$$

From (5) we get:

$$\sum_{n=1}^{N} t_n - N \cdot w_0^{opt} - w_1^{opt} \sum_{n=1}^{N} x_n = 0$$

With $\bar{t} = \frac{1}{N} \sum_{n=1}^{N} t_n,\ \bar{x}_1 = \frac{1}{N} \sum_{n=1}^{N} x_n$, and dividing (7) by N we get:

$$\bar{t} = w_0^{opt} + w_1^{opt} \bar{x}_1$$

It is an interesting intermediate result, that the center point of the data set, $(\bar{x}_1, \bar{t})$, lies on the regression line.

From (6) we get:

$$\begin{aligned}\sum_{n=1}^{N} t_n x_n - \sum_{n=1}^{N} w_0 x_n - \sum_{n=1}^{N} w_1^{opt} x_n^2 &= 0 \\ \sum_{n=1}^{N} t_n x_n - N w_0^{opt} \bar{x}_1 - w_1^{opt} \sum_{n=1}^{N} x_n^2 &= 0\end{aligned}$$

Together with (8), we get

$$\begin{aligned}\sum_{n=1}^{N} t_n x_n - N \bar{x}_1 (\bar{t} - w_1^{opt} \bar{x}_1) &= w_1^{opt} \sum_{n=1}^{N} x_n^2 \\ \sum_{n=1}^{N} t_n x_n - N \bar{x}_1 \bar{t} &= w_1^{opt} \left(\sum_{n=1}^{N} x_n^2 - N \bar{x}_1^2 \right)\end{aligned}$$

, which can be resolved to w_1^{opt}:

$$w_1^{opt} = \frac{\sum_{n=1}^{N} t_n x_n - N \bar{x}_1 \bar{t}}{\sum_{n=1}^{N} x_n^2 - N \bar{x}_1^2}$$

So, the solution for a line regression with the least squares method is:

$$w_0^{opt} = \bar{t} - w_1^{opt}\bar{x}_1$$
$$w_1^{opt} = \frac{\sum_{n=1}^{N} t_n x_n - N\bar{x}_1\bar{t}}{\sum_{n=1}^{N} x_n^2 - N\bar{x}_1^2}$$

For $N \geq 2$ we can write the *sample variance* of x_1 as

$$s_{x_1}^2 = \frac{1}{N-1}(\sum_{n=1}^{N} x_n^2 - N\bar{x}_1^2)$$

and the *sample covariance* of t and x_1 as

$$s_{x_1,t} = \mathrm{cov}(x_1, t) = \frac{1}{N-1}\left(\sum_{n=1}^{N} t_n x_n - N\bar{x}_1\bar{t}\right)$$

With (14) to (17) we can write the solution in a shorter manner:

$$w_0^{opt} = \bar{t} - \frac{s_{x_1,t}}{s_{x_1}^2} \cdot \bar{x}_1$$
$$w_1^{opt} = \frac{s_{x_1,t}}{s_{x_1}^2}$$

Up to this point we only know that the result (w_0^{opt}, w_1^{opt}) presents an optimum, not if it is a maximum, minimum, or saddle point. We have to check if it is actually a minimum by applying the second partial derivative test for functions of two variables. For this we need the second partial derivative $\frac{\partial^2 S}{\partial w_0^2}$ and the determinant of the Hessian matrix of S, $\det(H(w_0, w_1))$. If the following two conditions are met, (w_0^{opt}, w_1^{opt}) presents a minimum:

$$\frac{\partial^2 S}{\partial w_0^2} \text{ is positive for } (w_0^{opt}, w_1^{opt}).$$
$$D(w_0, w_1) = \det(H(w_0, w_1)) = \frac{\partial^2 S}{\partial w_0^2}\frac{\partial^2 S}{\partial w_1^2} - \left(\frac{\partial^2 S}{\partial w_0 \partial w_1}\right)^2 \text{ is positive for } (w_0^{opt}, w_1^{opt})$$

Exercise

After importing the dataset from `../assets/Uni_linear.txt` , brew your own Linear Regression model by following the blow steps:

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

%matplotlib inline
```

(A) Load the dataset `../assets/Uni_linear.txt` :

In [3]:
```
data_df = pd.read_csv("../assets/Uni_linear.txt", header=None)
data_df.columns=["X", "Y"]
data_df.head()
```

Out[3]:

	X	Y
0	6.1101	17.5920
1	5.5277	9.1302
2	8.5186	13.6620
3	7.0032	11.8540
4	5.8598	6.8233

(B) Explose the statistics of your dataset

In [4]:
```
#### YOUR CODE HERE
#
#
#
####
```

(C) Plot the dataset

In [6]:
```
#### YOUR CODE HERE
#
#
#
####
```

(D) Define MSE cost function ($J(\Theta)$)

$J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2$

In [8]:
```
#### YOUR CODE HERE
#
#
#
####
```

(F) Apply the cost function of your dataset with taking the initial model parameter values($\Theta = 0$) :

In [10]:
```
#### YOUR CODE HERE
#
#
#
####
```

(E) Visualize the cost function using 3D surface plot function (`plot_surface`).

`tip` : you have to import

```
from mpl_toolkits.mplot3d import Axes3D
```

In [12]:
```
from mpl_toolkits.mplot3d import Axes3D
```

In [13]:
```
#### YOUR CODE HERE
#
#
#
####
```

(G) Create a Gradient Descent function :

Minimize the cost function $J(\Theta)$

By updating Equation and repeat unitil convergence

$\Theta_j := \Theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ (simultaneously update Θ_j for all j)

In [15]:
```
#### YOUR CODE HERE
#
#
#
####
```

(H) Optimize taking $\alpha = 0.01$ and maximum number of iterations is 1500, then print the trained model:

In [17]:
```
#### YOUR CODE HERE
#
#
#
####
```

(I) Plot the *epoch* against $J(\Theta)$:

In [19]:
```
#### YOUR CODE HERE
#
#
#
####
```

(J) Make prediction function `predict` using your optimized theta :

In [21]:
```
#### YOUR CODE HERE
#
#
#
####
```

(K) Plot the line with best fit

In [23]:
```
#### YOUR CODE HERE
#
#
#
####
```

Exercise *[[4](#ref4)]*

Using `diabetes` dataset (https://bit.ly/2m3mip8) provided in `sklearn` , perform linear regression model using `sklearn.linear_model.LinearRegression` with xxx cost function.

In [25]:
```
from sklearn.datasets import load_diabetes
```

In [26]:
```
diabetes_rawset = load_diabetes()
diabetes_rawset
```

Out[26]:

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
          0.01990842, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06832974, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
          0.00286377, -0.02593034],
        ...,
        [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
         -0.04687948,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
          0.04452837, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00421986,  0.00306441]]),
 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen baseli
ne variables, age, sex, body mass index, average blood\npressure, and six blood seru
m measurements were obtained for each of n =\n442 diabetes patients, as well as the
response of interest, a\nquantitative measure of disease progression one year after
baseline.\n\n**Data Set Characteristics:**\n\n  :Number of Instances: 442\n\n  :Numb
er of Attributes: First 10 columns are numeric predictive values\n\n  :Target: Colum
n 11 is a quantitative measure of disease progression one year after baseline\n\n  :
Attribute Information:\n      - Age\n      - Sex\n      - Body mass index\n      - A
verage blood pressure\n      - S1\n      - S2\n      - S3\n      - S4\n      - S5\n
- S6\n\nNote: Each of these 10 feature variables have been mean centered and scaled
by the standard deviation times `n_samples` (i.e. the sum of squares of each column
```

```
totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html
\n\nFor more information see:\nBradley Efron, Trevor Hastie, Iain Johnstone and Robe
rt Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussio
n), 407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'c:\\users\\g\\appdata\\local\\programs\\python\\python36\\lib\\si
te-packages\\sklearn\\datasets\\data\\diabetes_data.csv.gz',
 'target_filename': 'c:\\users\\g\\appdata\\local\\programs\\python\\python36\\lib
\\site-packages\\sklearn\\datasets\\data\\diabetes_target.csv.gz'}
```

In [27]:
```
# Organize your dataset into feature, targets
features_rawset = diabetes_rawset.data
Y = diabetes_rawset.target
```

In [28]:
```
features_rawset.shape
```

Out[28]:
```
(442, 10)
```

In [29]:
```
# Organize your dataset into DataFrame for better visualization
diabetes_df = pd.DataFrame(features_rawset,
                           columns=diabetes_rawset.feature_names)
diabetes_df["targets"]=Y
diabetes_df.head()
```

Out[29]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	ta
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	

In [30]:
```
diabetes_df.shape
```

Out[30]:
```
(442, 11)
```

In [31]:
```
# Make sure the dimensionalies are correct
assert len(diabetes_df.columns)-1 == len(diabetes_rawset.feature_names), "ERROR"
```

In this Exercise, we will use only one independent variable for building our linear regression model, and this feature is `bmi` .

In [32]:
```
focused_df = diabetes_df[["bmi", "targets"]]
focused_df.head()
```

Out[32]:

	bmi	targets
0	0.061696	151.0
1	-0.051474	75.0
2	0.044451	141.0
3	-0.011595	206.0
4	-0.036385	135.0

(A) Split your dataset in `focused_df` into:

- 80% Training
- 20% Test

In [33]:
```
#### YOUR CODE HERE
#
#
####
```

(B) Train a linear a linear regression model using scikit-learn `LinearRegression` .

> `tip` : To Evaluate the performance of your model better use 10-fold cross validation

In [35]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import r2_score
```

In [36]:
```
#### YOUR CODE HERE
#
#
####
```

(C) Predict the `x_ts` , then use the `r2_score` (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score) to qualify the performance of your model on the test data

In [41]:
```
#### YOUR CODE HERE
#
#
####
```

(D) Plot your model with the train and test data with distinct colors

In [43]:
```
#### YOUR CODE HERE
#
#
####
```

Multivariate Regression

The `LinearRegression` estimator is much more capable than this, however—in addition to simple straight-line fits, it can also handle multidimensional linear models of the form

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots$$

where there are multiple x values. Geometrically, this is akin to fitting a plane to points in three dimensions, or fitting a hyper-plane to points in higher dimensions.

The multidimensional nature of such regressions makes them more difficult to visualize, but we can see one of these fits in action by building some example data, using NumPy's matrix multiplication operator:

Exercise *[[1](#ref1)]*

From the Diabetes dataset is the previous exercise, use the age as the second feature to:

(A) train a linear regression model with hypothesis function:

$$y = a_0 + a_1 \, \text{age} + a_2 \, \text{bmi}$$

(B) print the model score (r2 score) for:

- Training set
- 10-fold CV for the training set
- test set

(C) plot your data and model plane.

In [45]:
```
#### YOUR CODE HERE
#
#
####
```

Linear Basis Function Models

(see **Bishop**, "Pattern Recognition and Machine Learning", Springer 2006, Sec. 3.1)

Given a training data set of N observations $\{\mathbf{x}_n\}$ and their corresponding target values $\{t_n\}$, where n=1..N, directly constructing an appropriate function $y(\mathbf{x})$ to predict the corresponding values of t for observations $\mathbf{x}$ not part of the training data set can prove difficult in many cases.

One way of approaching this problem is modelling $y(\mathbf{x})$ as a **linear** combination of functions of $\mathbf{x}$:

$$\begin{aligned} y(\mathbf{x}, \mathbf{w}) &= w_0 + w_1 \cdot \phi_1(\mathbf{x}) + w_2 \cdot \phi_2(\mathbf{x}) + \ldots + w_{M-1} \cdot \phi_{M-1}(\mathbf{x}) \\ &= w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \end{aligned}$$

where $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$. D is the dimensionality of the observations. The fixed functions $\phi_j(\mathbf{x})$ are known as *basis* functions.
The parameter w_0 allows to model a fixed offset in the data. Including w_0, the model has a number of **M** parameters.

By defining a dummy basis function $\phi_0(\mathbf{x}) = 1$, we get

$$\begin{aligned} y(\mathbf{x}, \mathbf{w}) &= w_0 \cdot \phi_0(\mathbf{x}) + w_1 \cdot \phi_1(\mathbf{x}) + w_2 \cdot \phi_2(\mathbf{x}) + \ldots + w_{M-1} \cdot \phi_{M-1}(\mathbf{x}) \\ &= \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) \end{aligned}$$

where $\mathbf{w} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ and $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^{\mathrm{T}}$.

Basis function examples

Valid:
$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 sin(\mathbf{x}) + w_2 cos(\mathbf{x})$
The basis function $sin(x)$ and $cos(x)$ are fixed.

Not valid:
$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 sin(w_1\ \mathbf{x}) + w_2 cos(\mathbf{x})$
$sin(w_1\ \mathbf{x})$ the parameter w_1 doesn't go into $y(\mathbf{x}, \mathbf{w})$ in a linear fashion.

Alternative Notation

This section show an alternative matrix-based notation of a linear regression solved with the least-squares approach.

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(x_1) & \cdots & \phi_{M-1}(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix} \text{ with } \mathbf{\Phi} \in \mathbb{R}^{N \times M},$$

$$\mathbf{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_{M-1} \end{pmatrix} \text{ with } \mathbf{w} \in \mathbb{R}^{M}, \qquad \mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix} \text{ with } \mathbf{t} \in \mathbb{R}^{N}$$

$$\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{t} = \begin{pmatrix} w_0\phi_0(x_1) + & \cdots & +w_{M-1}\phi_{M-1}(x_1) - y_1 \\ & \vdots & \\ w_0\phi_0(x_N) + & \cdots & +w_{M-1}\phi_{M-1}(x_N) - y_N \end{pmatrix}$$

$$= \begin{pmatrix} \left(\sum_{j=0}^{M-1} w_j\phi_j(x_1)\right) - y_1 \\ \vdots \\ \left(\sum_{j=0}^{M-1} w_j\phi_j(x_N)\right) - y_N \end{pmatrix}$$

$$||\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{t}||^2 = \left(\left(\sum_{j=0}^{M-1} w_j\phi_j(x_1)\right) - y_1\right)^2 + \cdots + \left(\left(\sum_{j=0}^{M-1} w_j\phi_j(x_N)\right) - y_N\right)^2$$
$$= \sum_{n=1}^{N} \left(\left(\sum_{j=0}^{M-1} w_j\phi_j(x_n)\right) - y_n\right)^2$$

The alternative notation for the least squares minimization problem, $\min ||\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{t}||^2$, is as follows:

Derivation

$$\frac{\partial}{\partial \mathbf{w}} ||\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{t}||^2 = 2 \cdot \mathbf{\Phi}^{\mathrm{T}} \cdot (\mathbf{\Phi} \cdot \mathbf{w} - \mathbf{t}) \qquad (\text{chain rule})$$

Set the derivative equal to 0 to find the optimal solution $\mathbf{w}^{opt}$:

$$2\mathbf{\Phi}^{\mathrm{T}} \cdot (\mathbf{\Phi} \cdot \mathbf{w}^{opt} - \mathbf{t}) = 0$$
$$2\mathbf{\Phi}^{\mathrm{T}} \cdot \mathbf{\Phi} \cdot \mathbf{w}^{opt} - 2\mathbf{\Phi}^{\mathrm{T}} \cdot \mathbf{t} = 0$$
$$\mathbf{\Phi}^{\mathrm{T}} \cdot \mathbf{\Phi} \cdot \mathbf{w}^{opt} = \mathbf{\Phi}^{\mathrm{T}} \cdot \mathbf{t}$$

$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$ is quadratic and must be invertible (must be assured, e.g., through the choice of ϕ_j), then multiplication on both sides from the left with $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}$:

$$\mathbf{w^{opt}} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}} \cdot \mathbf{t} \qquad \text{with: } \mathbf{\Phi}^{\dagger} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}} \text{ is the pseudo-inverse of } \mathbf{\Phi}.$$

$\mathbf{w}^{opt} = \mathbf{\Phi}^{\dagger} \cdot \mathbf{t}$ is the optimal solution.

Exercise

learn the linear model with:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 sin(\mathbf{2}\pi\ \mathbf{x}) + w_2 cos(\mathbf{2}\pi\ \mathbf{x})$$

In [8]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

%matplotlib inline
```

In [30]:
```
### Generate a dataset
def data_generator(m=400,
                   noise_mu=0,
                   noise_sigma=0.2,
                   x_range=(0,2*np.pi)):
    """
    Generate m d 1D data set of target_func.
    Add normal dist. noise with mean=noise_mu and
    var=noise_sigma**2. The range of the generated features
    is x_range.

    return the features with its generated labels
    """

    # Target function (ground truth) generator
    target_func = lambda x : np.sin(0.5*np.pi*x) + 1.5

    # Noise generator
    noise = lambda x : 1-np.random.normal(noise_mu, noise_sigma, x.shape)

    # Generate data
    x = np.random.rand(m)*abs(x_range[1]-x_range[0])+x_range[0]
    y = target_func(x) + noise(x)

    # Reshape
    x, y = x.reshape(-1,1), y.reshape(-1,1)

    # Integrity check
    assert len(x)==len(y), "Error in data generator"
    return x,y
```

(A) Generate Train/Test dataset with 400 observations, then plot:

In [23]:
```
#### YOUR CODE HERE
#
#
###
```

(B) Split your Data into (Train,Test) $(80\%,20\%)$ randomly:

> `tip` : your seed 56

In [ ]:
```
#### YOUR CODE HERE
#
#
###
```

(C) Plot the train dataset

In [ ]:
```
#### YOUR CODE HERE
#
#
###
```

(D) learn the linear model with:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 sin(\mathbf{2}\pi\ \mathbf{x}) + w_2 cos(\mathbf{2}\pi\ \mathbf{x})$$

Then evaluate the performance of your model using 10-fold CV.

In [ ]:
```
#### YOUR CODE HERE
#
#
###
```

(E) Print your model equation

In [ ]:
```
#### YOUR CODE HERE
#
#
###
```

(F) Get the Generalization performance of your trained model

In [ ]:
```
#### YOUR CODE HERE
#
#
###
```

(F) Plot your model, with training- and test data in distinct colors

In [ ]:
```
#### YOUR CODE HERE
#
#
###
```


[01] BOOK Python Data Science Handbook, J.VanderPlas (https://amzn.to/2znPlYg)

[02] BOOK Mastering Machine Learning with Python in Six Steps, M.Swamynathan (https://amzn.to/2m1uXsa)

[03] BOOK Pattern Recognition and Machine Learning, C.Bishop (https://amzn.to/2m3iB2L)

[04] BLOG Scikit-learn Diabetes Regression (https://bit.ly/2TuZzyc)