Contents

- Least squares optimization
  - Linear regression
  - Coefficient of determination(R^2): goodness of fit
- Nonlinear regression
  - Piecewise linear regression
  - Moving average
  - Moving least squares
- Regularization and cross-validation
  - Lp-norm
  - K-fold

Least Squares Optimization: Linear Regression

- Regression Analysis
  - Set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome' or 'response' or ' target ') and one or more independent variables (often called 'predictors', 'covariates', 'explanatory variables' or 'features')
- Linear regression
  - Simplest method to build a relationship between input and output while many relationships are nonlinear in science and engineering
  - Fundamental to understanding more advanced regression methods
  - Adrien-Marie Legendre(1805), Johann Carl Friedrich Gauss(1809)
    - Orbits of celestial bodies
  - Term "regression": Francis Galton (1800's)
    - Genetics of sweet peas: weights of planted and harvested peas

https://en.wikipedia.org/wiki/Regression_analysis

Linear Regression Model

consider a set of N data points $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

a generic equation through these points : regression model made of two weights

$$y_n^* = w_0 + w_1 x_n \approx y_n \quad \text{for} \quad n = 1, \ldots, N$$

$$\begin{cases} w_0 : \text{ bias } (\text{constant weight}), w_1 : \text{ weight} \\ y_n^* : \text{ computed approximate value, } y_n : \text{ "}true\text{" value} \end{cases}$$

assess how accurate the model is: total error $= \sum_{n=1}^{N} \left(y_n^* - y_n \right)^2 = \sum_{n=1}^{N} \left(w_0 + w_1 x_n - y_n \right)^2$

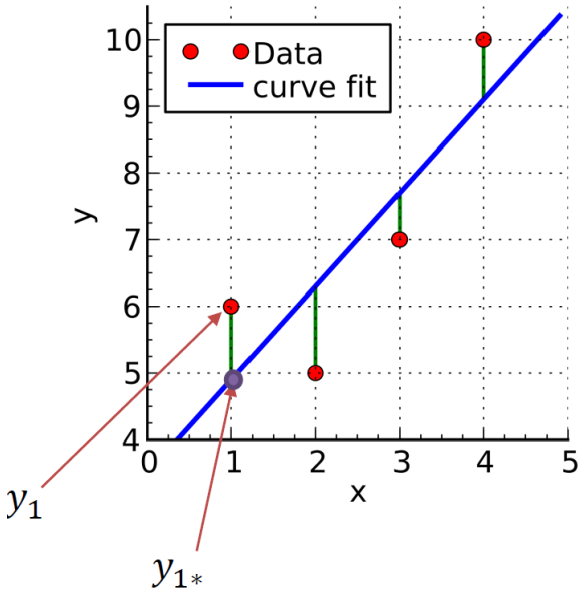
cost function $= c(w_0, w_1) = \frac{1}{N} \sum_{n=1}^{N} \left(w_0 + w_1 x_n - y_n \right)^2 = \text{MSE}(\text{Mean Squared Error})$

$\rightarrow$ minimize the cost function to find optimal weights: $\arg\min_{\mathbf{w}} \text{MSE}$

quantify how well the regression fit is: goodness of fit

$$\text{coefficient of determination} = R^2 = \frac{\text{regression sum of squares}}{\text{total sum of squares}} = \frac{\sum_n \left(y_n^* - \bar{y} \right)^2}{\sum_n \left(y_n - \bar{y} \right)^2}$$

$\bar{y}$: average of the data points y_n

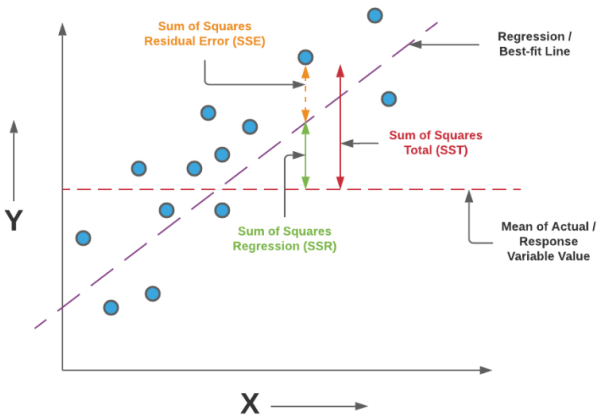


Sum of Squares (SS)

수치	R	E
설명 가능	Regression	Explained
설명 불가능	Residual	Error

- sum of squares total (SST) or the total sum of squares (TSS)
  - differences between the observed dependent variables and the overall mean
  - similar to the variance in descriptive statistics
  - total variability of a dataset, commonly used in regression analysis and ANOVA
- sum of squares due to regression (SSR) or explained sum of squares (ESS)
  - differences between the predicted value and the mean of the dependent variable
  - how well our line fits the data
- sum of squares error (SSE) or residual sum of squares (RSS, where residual means remaining or unexplained)
  - difference between the observed and predicted values

$$SS_{tot} = \sum \underbrace{\left(y - \bar{y} \right)}_{\text{total variability 편차}}^2, \quad SS_{res} = \sum \underbrace{\left(y - \hat{y} \right)}_{\text{residual 잔차}}^2, \quad SS_{reg} = \sum \left(\left(y - \bar{y} \right) - \left(y - \hat{y} \right) \right)^2 = \sum \underbrace{\left(\hat{y} - \bar{y} \right)}_{\text{regression}}^2$$



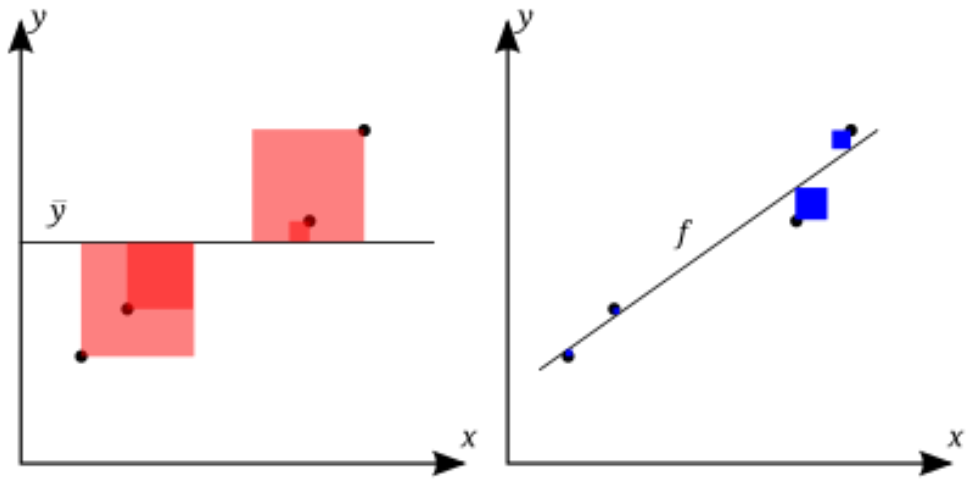
Coefficient of Determination (R-squared)

- Proportion of the variance in the dependent variable that is explained by the independent variable(s): (explained variation)/(total variation)
- How well a model explains and predicts future outcomes: goodness of fit
- How much smaller SSE is than SST

$$R^2 = \frac{SS_{tot} - SS_{res}}{SS_{tot}} = 1 - \frac{SS_{res}}{SS_{tot}} \rightarrow \begin{cases} R^2 = 1 \\ R^2 = -100 \end{cases}$$

$$SS_{tot} = \sum \left(y - \bar{y}\right)^2 = \sum \left(y - y^* + y^* - \bar{y}\right)^2$$

$$= \underbrace{\sum \left(y - y^*\right)^2}_{SS_{res}} + \underbrace{\sum \left(y^* - \bar{y}\right)^2}_{SS_{reg}} + 2\sum \left(y - y^*\right)\left(y^* - \bar{y}\right)$$

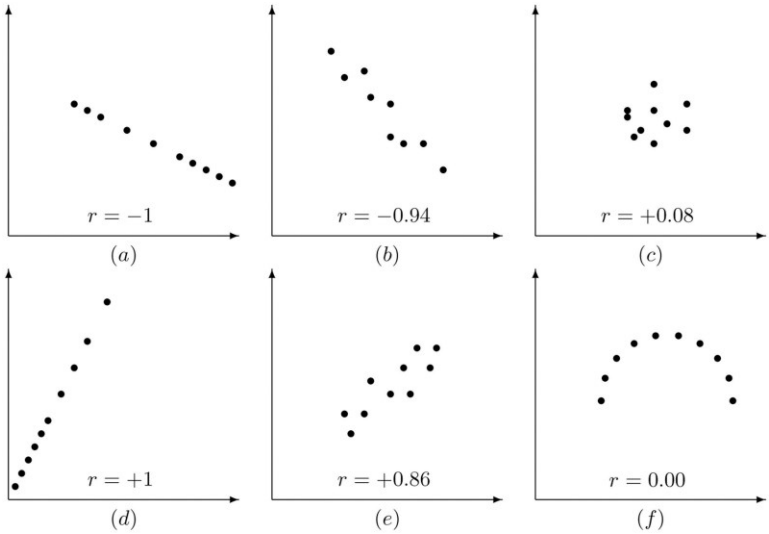


linear regression using least squares

$$\sum \left(y - y^*\right)\left(y^* - \bar{y}\right) = 0 \rightarrow SS_{tot} = SS_{res} + SS_{reg} \rightarrow R^2 = \frac{SS_{reg}}{SS_{tot}} = r^2 \rightarrow \text{multiple regression?}$$

Correlation Coefficient (r)

- Covariance
  - whether both variables vary in same direction (+) or opposite direction (-)
  - no significance of covariance numerical value only sign is useful
- (Pearson) Correlation Coefficient
  - 공분산을 각각의 변수의 표준 편차로 나누어 준 값
  - how closely the points in a scatter plot follow a straight line



$$\left.\begin{array}{l} E(X)=\mu \\ E(Y)=\nu \end{array}\right\} \rightarrow Cov(X,Y)=E\left((X-\mu)(Y-\nu)\right)=\frac{\sum_{i=1}^{N}(X_i-\mu)(Y_i-\nu)}{N-1}$$

$$r_{XY}=\frac{\sum_{i=1}^{N}(X_i-\mu)(Y_i-\nu)}{\sqrt{\sum_{i=1}^{N}(X_i-\mu)^2}\sqrt{\sum_{i=1}^{N}(Y_i-\nu)^2}} \rightarrow (-1\le r\le 1) \rightarrow \begin{cases} sign(r): \text{ direction (increase/decrease)} \\ |r|: \text{ strength (strong/weak)} \end{cases}$$

Relationship Analysis: R vs. r

- R: how well a model explains the variability in the dependent variable
- r: strength and direction of a linear relationship between two variables

$$r\left(y,y^*\right)^2=\frac{\left[\sum_{n=1}^{N}\left(y_n-\bar{y}\right)\left(y_n^*-\bar{y}\right)\right]^2}{\sum_{n=1}^{N}\left(y_n-\bar{y}\right)^2\sum_{n=1}^{N}\left(y_n^*-\bar{y}\right)^2}=\frac{\left[\sum_{n=1}^{N}\left(y_n-y_n^*+y_n^*-\bar{y}\right)\left(y_n^*-\bar{y}\right)\right]^2}{\sum_{n=1}^{N}\left(y_n-\bar{y}\right)^2\sum_{n=1}^{N}\left(y_n^*-\bar{y}\right)^2}=\frac{\left[\underbrace{\sum_{n=1}^{N}\left(y_n-y_n^*\right)\left(y_n^*-\bar{y}\right)}_{=0}+\sum_{n=1}^{N}\left(y_n^*-\bar{y}\right)^2\right]^2}{\sum_{n=1}^{N}\left(y_n-\bar{y}\right)^2\sum_{n=1}^{N}\left(y_n^*-\bar{y}\right)^2}=\frac{\sum_{n=1}^{N}\left(y_n^*-\bar{y}\right)^2}{\sum_{n=1}^{N}\left(y_n-\bar{y}\right)^2}=\frac{SS_{reg}}{SS_{tot}}$$

$$\sum_{n=1}^{N}\left(y_n-y_n^*\right)\left(y_n^*-\bar{y}\right)=-\sum_{n=1}^{N}e_n\left(y_n^*-\bar{y}\right)=-\left[\sum_{n=1}^{N}y_n^*e_n-\bar{y}\sum_{n=1}^{N}e_n\right]=0$$

linear regression using least squares

$$c=\sum_{n=1}^{N}e_n{}^2=\sum_{n=1}^{N}\left(y_n^*-y_n\right)^2=\sum_{n=1}^{N}\left(w_0+w_1x_n-y_n\right)^2\rightarrow\begin{cases}\dfrac{\partial c}{\partial w_0}=2\sum_{n=1}^{N}\left(w_0+w_1x_n-y_n\right)=0\rightarrow\sum_{n=1}^{N}e_n=0\rightarrow\sum_{n=1}^{N}\left(y_n^*-y_n\right)=0\\ \dfrac{\partial c}{\partial w_1}=2\sum_{n=1}^{N}\left(w_0+w_1x_n-y_n\right)x_n=0\rightarrow\sum_{n=1}^{N}x_ne_n=0\end{cases}$$

$$y_n^*=w_0+w_1x_n\rightarrow\sum_{n=1}^{N}y_n^*e_n=\sum_{n=1}^{N}\left(w_0+w_1x_n\right)e_n=w_0\sum_{n=1}^{N}e_n+w_1\sum_{n=1}^{N}x_ne_n=0$$

Adjusted R-squared (Coefficient of Determination)

  - R-squared value always increases or remains the same when more predictors are added to the model, even if those predictors do not significantly improve the model's explanatory power: misleading impression of the model's effectiveness

- Interpreting Adjusted R-Squared in Practice
  - Model 1 has an r-squared of 0.9 and an adjusted r-squared of 0.75
  - Model 2 has an r-squared of 0.85 and an adjusted R-squared of 0.8

$$\bar{R}^2 = R^2_{adj} = 1 - \frac{SS_{res}/dof_{res}}{SS_{tot}/dof_{tot}} = 1 - \frac{SS_{res}}{SS_{tot}} \frac{n-1}{n-p-1} = 1 - \left(1 - R^2\right) \frac{n-1}{n-p-1}$$

n : sample size

p : number of variables

Evaluation of Regression Analysis (1)

Metric	Full Name	Interpretation and Use Cases	Robust to outlier?
MAE	Mean Absolute Error	Simple, interpretable measure of average error magnitude. Less sensitive to outliers. Useful for typical error assessment.	Yes
MAPE	Mean Absolute Percentage Error	Intuitive for relative errors. Ideal for varying scales or when relative errors are more relevant.	Yes
MSE	Mean Squared Error	Penalizes larger errors heavily. Useful for optimizing models by minimizing large errors.	No
RMSE	Root Mean Squared Error	Interpretable measure of average error distance in original units. Retains sensitivity to larger errors like MSE.	No

Evaluation of Regression Analysis (2)

$$MAE = \frac{1}{n}\sum\left|y - \hat{y}\right|$$

Divide by the total number of data points

Predicted output value

Actual output value

Sum of

The absolute value of the residual

$$MSE = \frac{1}{n}\sum\left(y - \hat{y}\right)^2$$

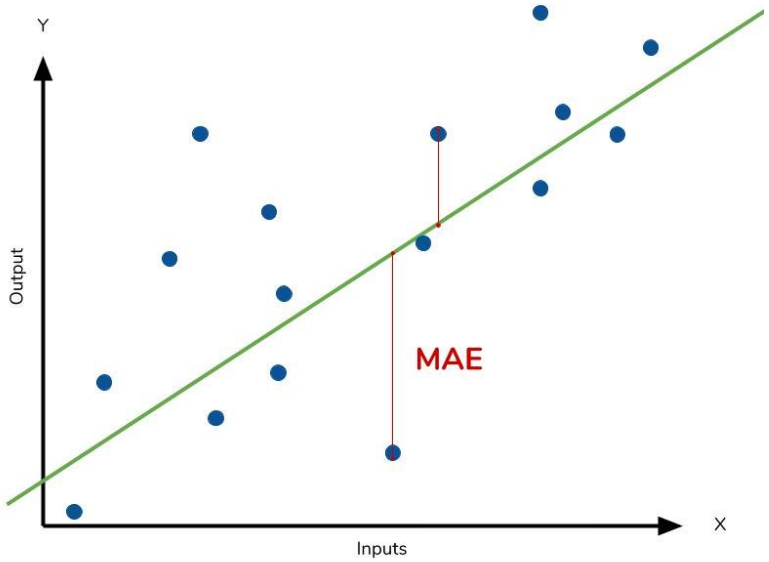
The square of the difference

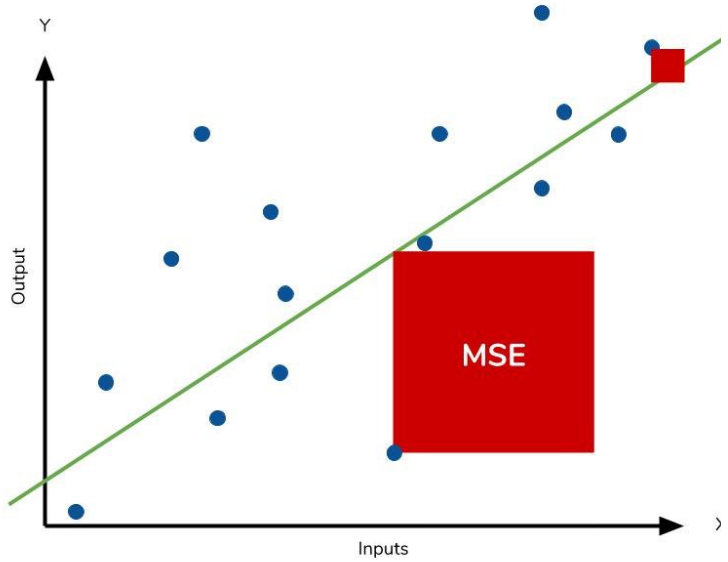
$$MAPE = \frac{100\%}{n}\sum\left|\frac{y - \hat{y}}{y}\right|$$

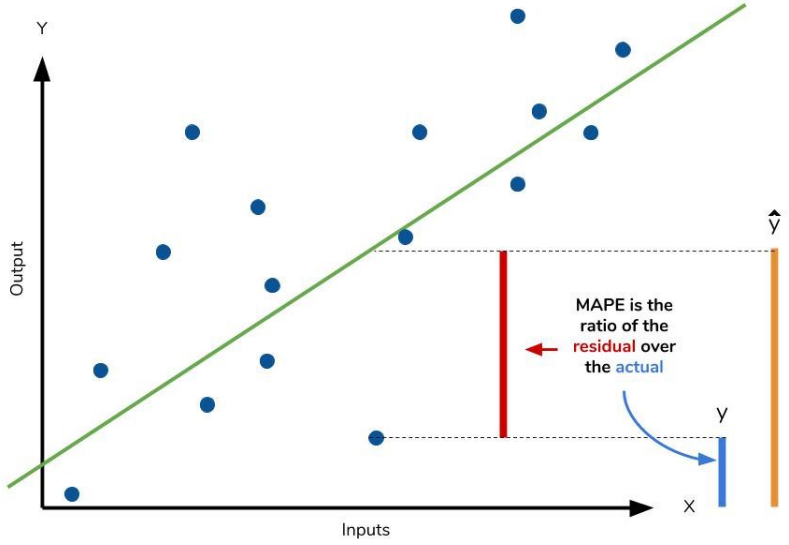
Multiplying by 100% converts to percentage

The residual

Each residual is scaled against the actual value



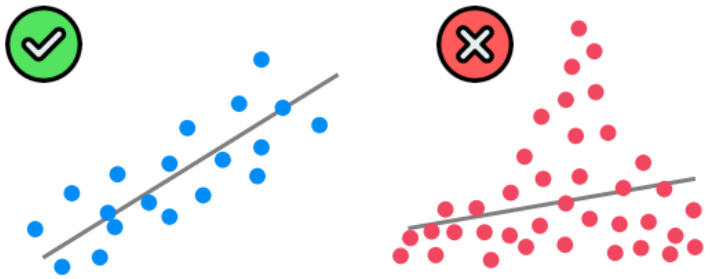




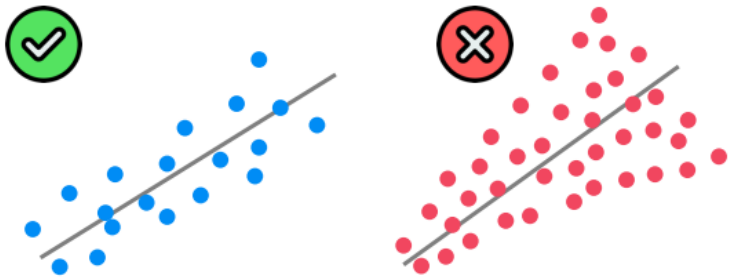
https://www.dataquest.io/blog/understanding-regression-error-metrics/

Assumptions of Linear Regression

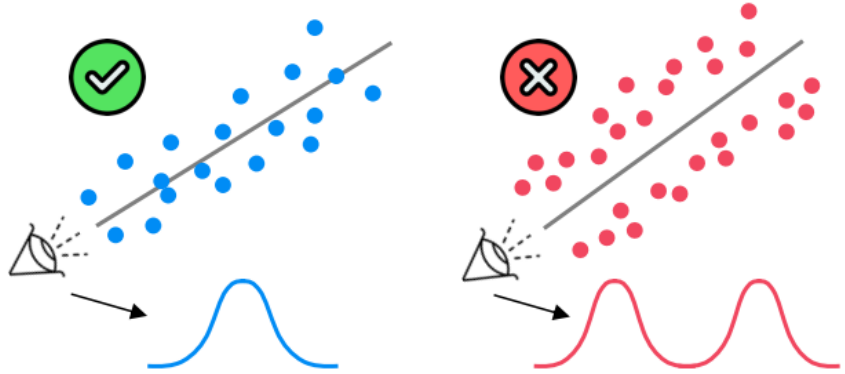
1. Linearity
(Linear relationship between Y and each X)



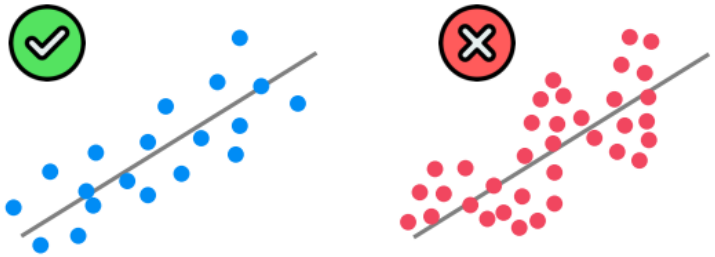
2. Homoscedasticity
(Equal variance)



3. Multivariate Normality
(Normality of error distribution)



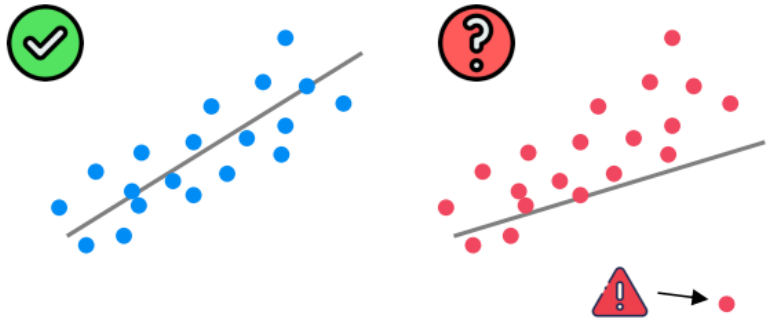
4. Independence
(of observations. Includes “no autocorrelation”)



5. Lack of Multicollinearity
(Predictors are not correlated with each other)

$X_1 \not\sim X_2$ $\quad X_1 \sim X_2$

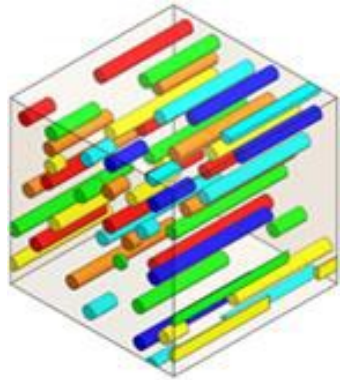
6. The Outlier Check
(This is not an assumption, but an “extra”)



https://www.superdatascience.com/blogs/assumptions-of-linear-regression

Background: Optimization (1)

- Process of finding the minimum (or maximum) value of a set of data or a function, generally performed mathematically
- Example
  - Let's say I'm an engineer at an automobile company tasked to design a new part. The part must satisfy a pre-determined list of requirements: strength, fatigue life weight manufacturability, etc. The part must also be economical to produce to keep profit margins up. As such we want minimize the cost.
  - We've decided to go with Carbon Fiber Reinforced Polymer (CFRP) as our material. Yet, we need to determine several factors: volume fraction, fiber orientation, fiber radius, etc. while satisfying the pre-determined list of requirements. As such our cost function will be of high-dimensionality.



Background: Optimization (2)

- What is a cost function?
- Maximum and minimum of a cost function: local vs. global
- First, second, and higher order derivatives in multiple dimensions: gradients
- Gradient descents: a key concept behind optimization

Optimization: c vs. w

Optimization involves finding the maximum or minimum of a function.

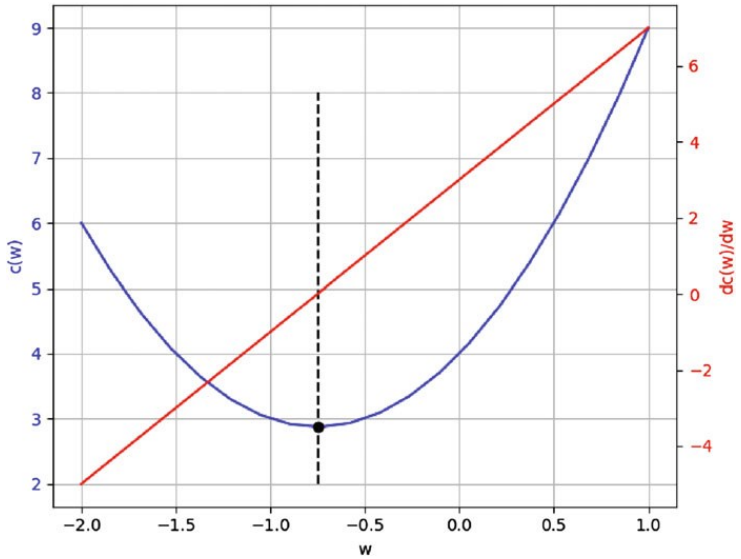
$$c(w) = 2w^2 + 3w + 4$$

First derivate test : $\frac{dc}{dw} = 0$ at maximum or minimum, find w^*

$$\frac{dc}{dw} = 4w + 3 = 0 \rightarrow w^* = -\frac{3}{4}$$

Second derivate test : $\frac{d^2c(w^*)}{dw^2}\begin{cases} > 0 \rightarrow \text{convex}(\text{minimum}) \\ < 0 \rightarrow \text{concave}(\text{maximum}) \\ = 0 \rightarrow (\text{inflection}) \end{cases}$

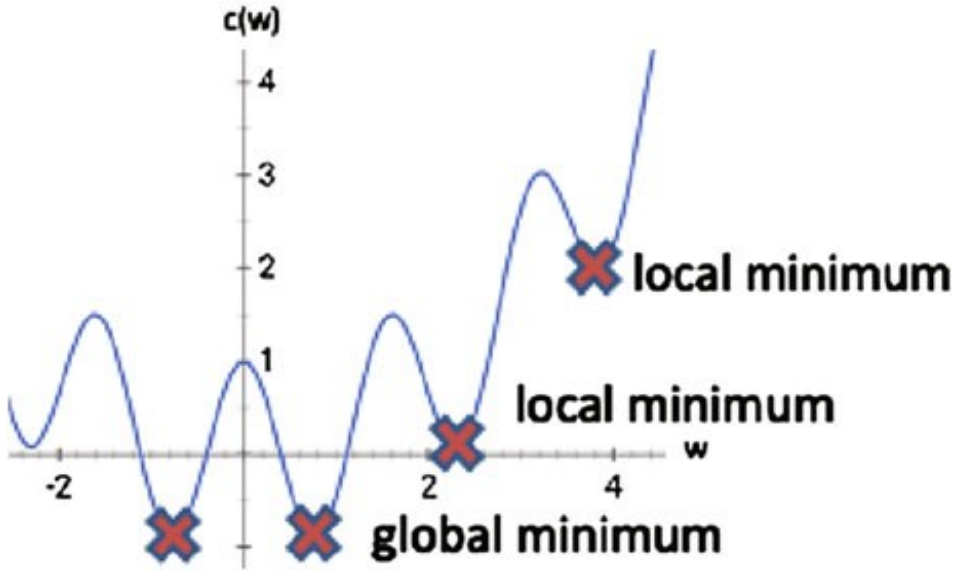
$$\frac{d^2c(w^*)}{dw^2} = 4 > 0 \rightarrow \text{minimum}$$



non-convex function with multiple local minimum

$$c(w) = \cos(4w) + 0.2w^2$$

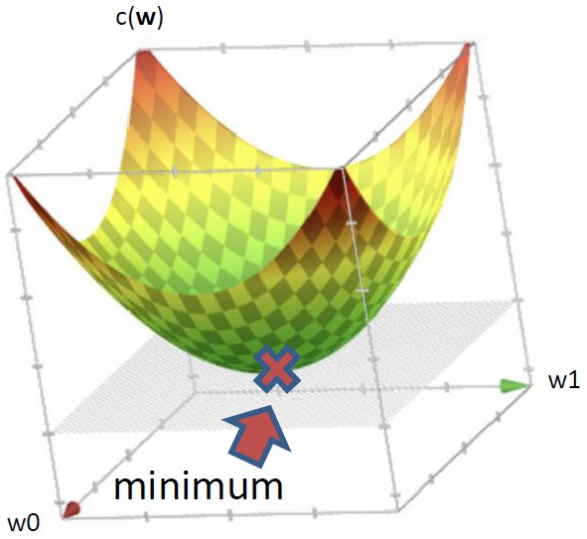
$$\frac{dc}{dw} = -4\sin(4w) + 0.4w = 0 \rightarrow w^* = ?$$

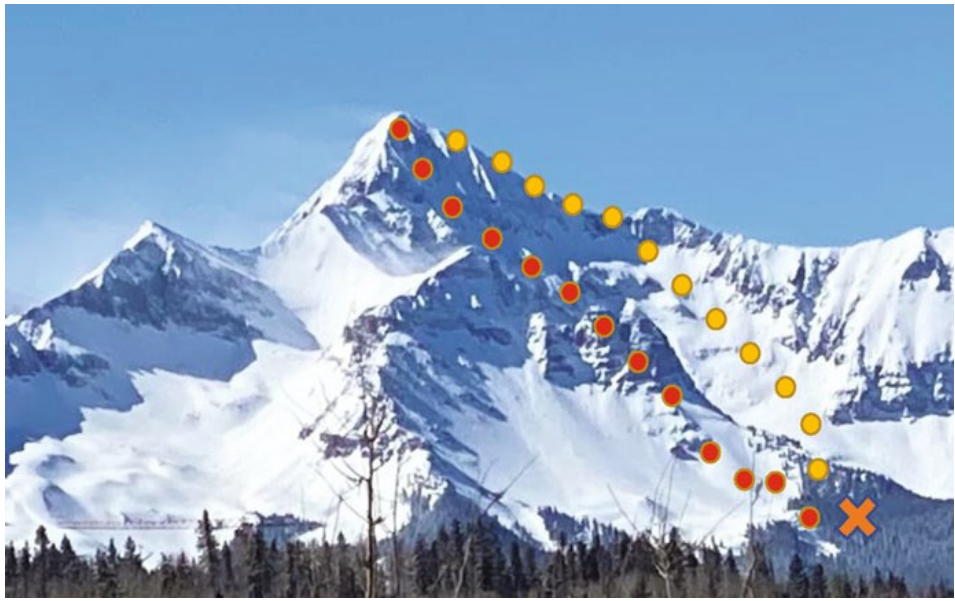


Optimization: c vs. (w_0, w_1,…, w_S)

- Multidimensional Derivatives
- Gradient: slope or rate of change in a particular direction
  - find the minimum of high dimensionality functions

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_S \end{bmatrix} \rightarrow c(\mathbf{w}) \rightarrow \nabla c(\mathbf{w}) = \begin{bmatrix} \frac{\partial c(\mathbf{w})}{\partial w_0} \\ \frac{\partial c(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial c(\mathbf{w})}{\partial w_S} \end{bmatrix}$$





$$c(\mathbf{w}) = (w_0)^2 + 2(w_1)^2 + 1$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \rightarrow \nabla c(\mathbf{w}) = \begin{bmatrix} \frac{\partial c(\mathbf{w})}{\partial w_0} \\ \frac{\partial c(\mathbf{w})}{\partial w_1} \end{bmatrix} = \begin{bmatrix} 2w_0 \\ 4w_1 \end{bmatrix} = 0 \rightarrow \mathbf{w}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

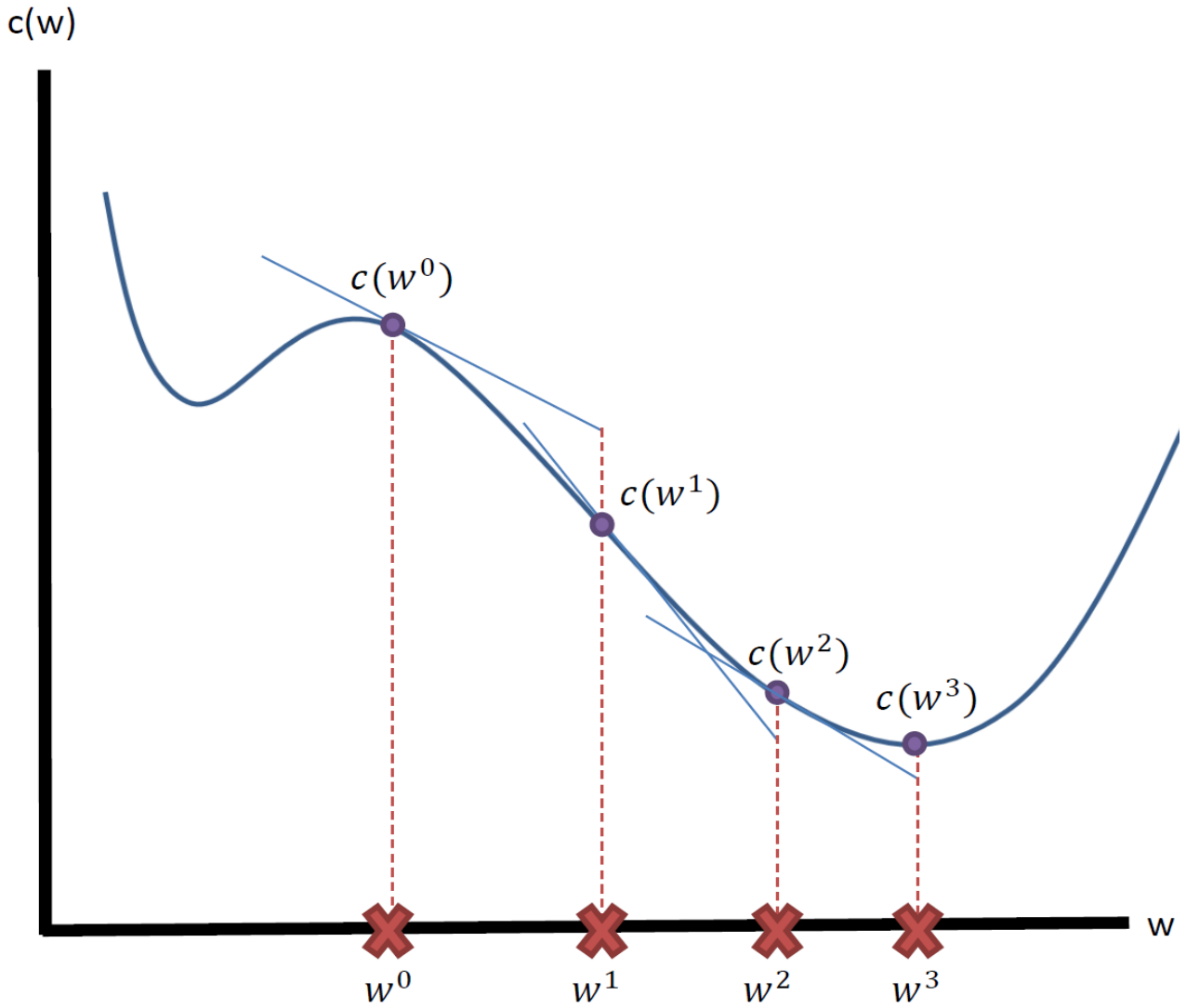
Gradient Descent

$$\begin{cases} w^{k+1} = w^k - \underbrace{\alpha}_{\substack{\text{step}\\ \text{size}}} \underbrace{\frac{dc\left(w^k\right)}{dw}}_{\substack{\text{search}\\ \text{direction}}} \\ \mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \nabla c\left(\mathbf{w}^k\right) \end{cases}$$

1. Start at an arbitrary point w^0
2. Find the derivative of c at w^0
3. Descend to the next point through

the gradient descent equation: $w^1 = w^0 - \alpha \frac{dc\left(w^0\right)}{dw}$

4. Repeat the process: $w^2 = w^1 - \alpha \frac{dc\left(w^1\right)}{dw}$



Gradient Descent: Example

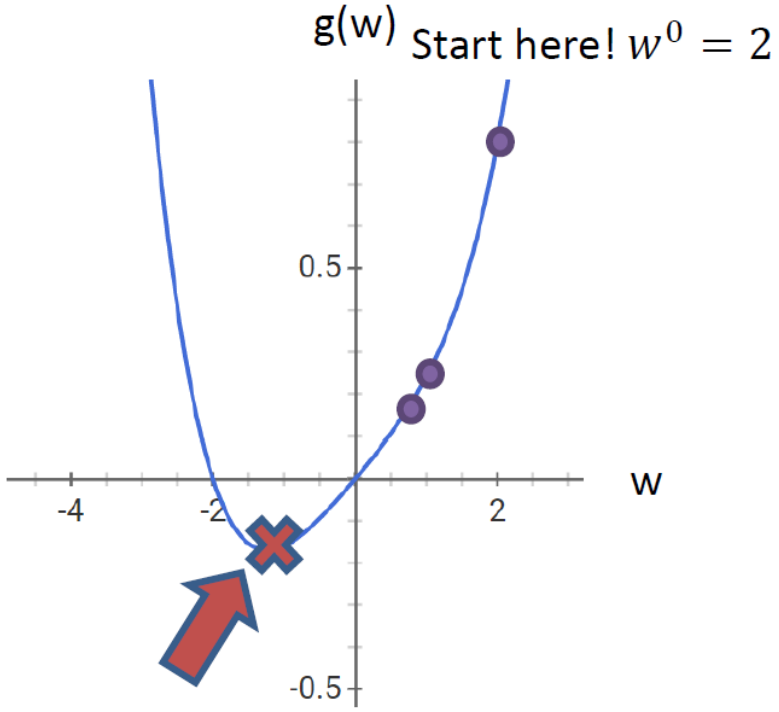
$$g(w) = \frac{1}{50}\left(w^4 + w^2 + 10w\right)$$

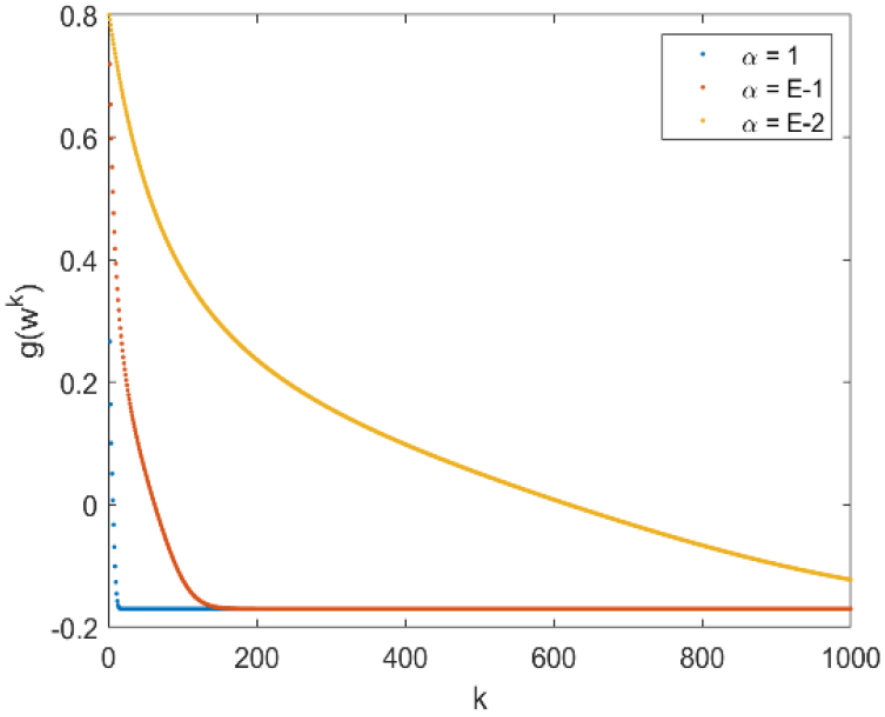
$$\frac{dg}{dw} = \frac{1}{50}\left(4w^3 + 2w + 10\right)$$

$$w^0 = 2,\ \alpha = 1$$

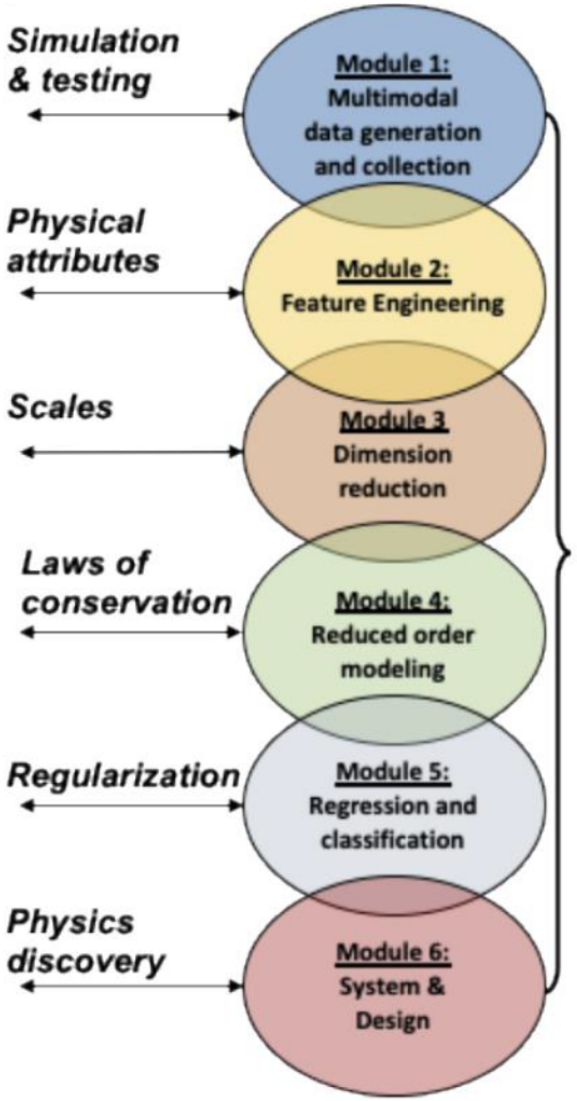
$$w^1 = w^0 - \alpha \frac{dg\left(w^0\right)}{dw} = 1.08$$

$$w^2 = w^1 - \alpha \frac{dg\left(w^1\right)}{dw} = 0.736$$





Example: Moneyball (1)



Baseball statistics are readily available. One such database is in Kaggle.

Various features are present in baseball sports analytics (OBP, SLG, RS, etc.).

Reduce dimension by only considering certain features (OBP, SLG, RS).

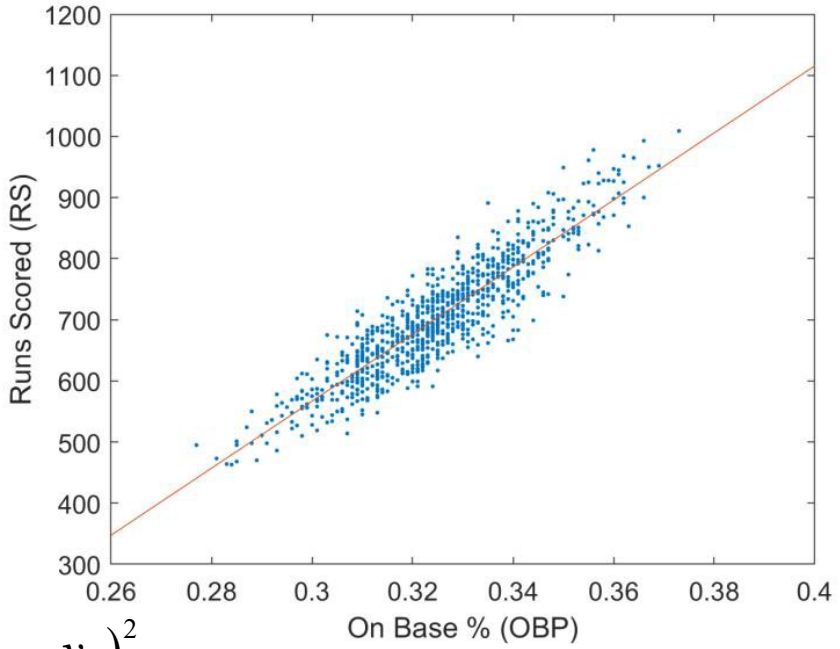
We reduce the order of the model by assuming it is linear.

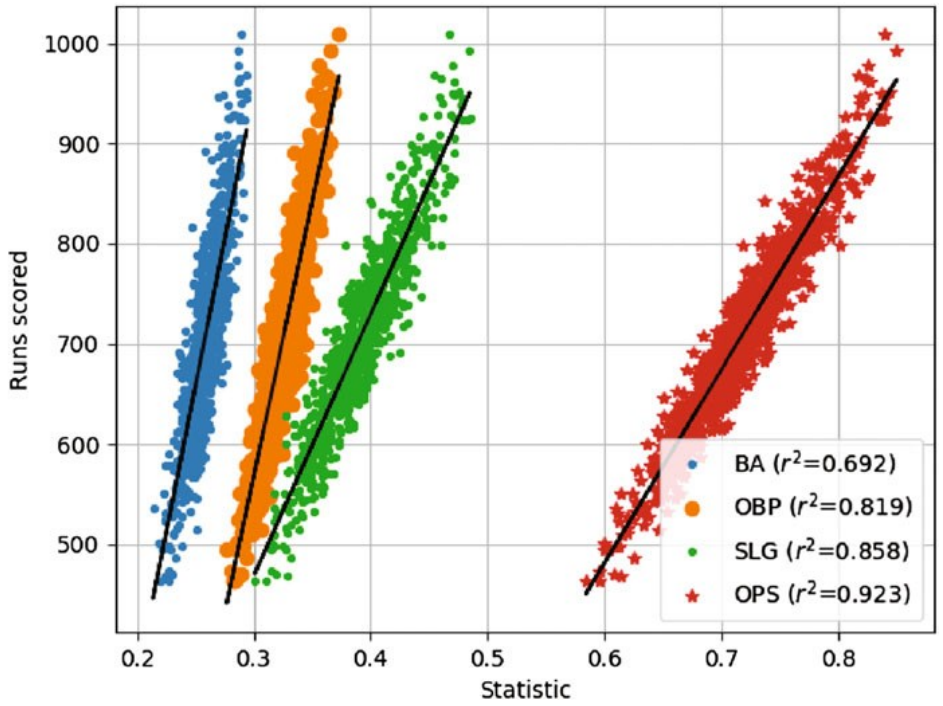
Use regression to determine model parameters

Use OBP and SLG to predict performance and therefore potential recruitment.

Example: Moneyball (2)

n	y_n RS	x_n OBP
	691	0.327
	818	0.341
	729	0.324
	687	0.319
	772	0.334
	777	0.336
	798	0.334
	735	0.324
	897	0.35





$$c(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(w_0 + w_1 x_n - y_n\right)^2$$

$x_n = \text{OBP} = \text{On Base Percentage}$

$y_n = \text{RS} = \text{Run Scored}$

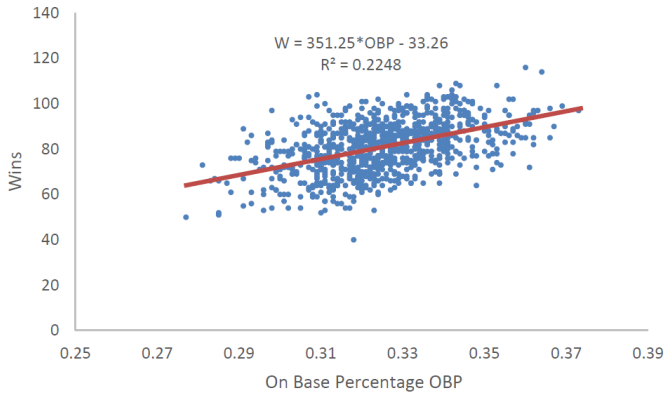
$$c(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left[\left(w_0 + w_1(0.327) - 691\right)^2 + \left(w_0 + w_1(0.341) - 818\right)^2 + \cdots\right]$$

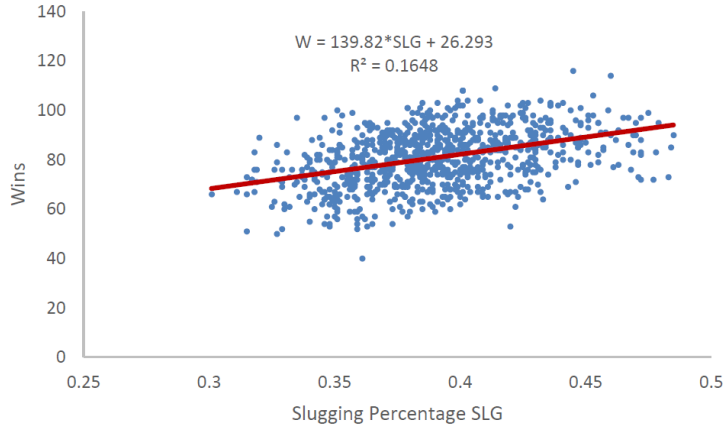
$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha\nabla c\left(\mathbf{w}^k\right)$$

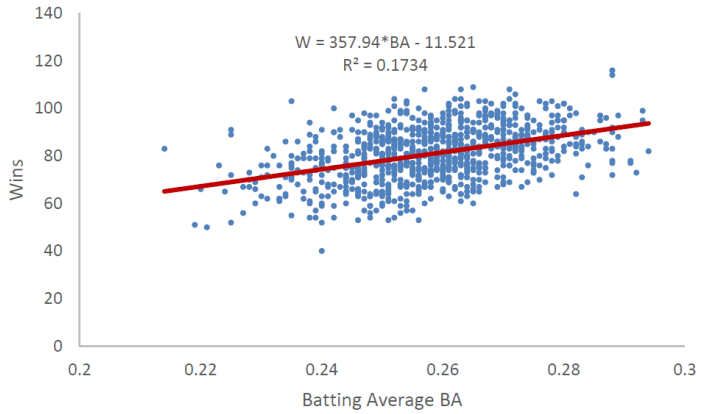
Can we get a better R-squared value if we include both OBP and SLG in a single regression?

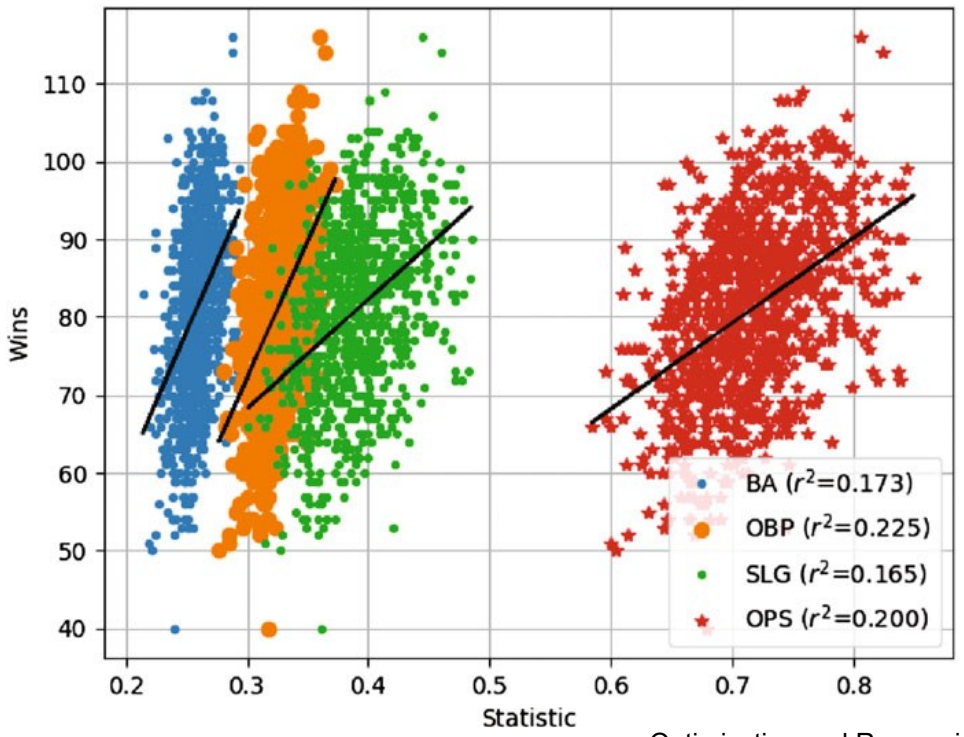
Example: Moneyball (3)

- correlation between W and any of these statistics? not good
  - do not account for pitching and defense, which are other important parts of winning baseball games









Multivariate Linear Regression Model

For high dimensions, consider a set of N data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

a generic equation through these points : regression model made of two weights

$$y_n^* = w_0 + w_1 x_{1,n} + w_2 x_{2,n} + \cdots + w_S x_{S,n} \approx y_n \quad \text{for} \quad n = 1, \ldots, N$$

$$\hat{\mathbf{x}}_n = \begin{bmatrix} 1 \\ x_{1,n} \\ x_{2,n} \\ \vdots \\ x_{S,n} \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_S \end{bmatrix} \rightarrow y_n^* = \hat{\mathbf{x}}_n{}^T \mathbf{w} \rightarrow c_n = \left(\hat{\mathbf{x}}_n{}^T \mathbf{w} - y_n \right)^2$$

$$c = \frac{1}{N} \sum_{n=1}^{N} \left(\hat{\mathbf{x}}_n{}^T \mathbf{w} - y_n \right)^2 \rightarrow \nabla c = \frac{2}{N} \sum_{n=1}^{N} \left(\hat{\mathbf{x}}_n{}^T \mathbf{w} - y_n \right) \hat{\mathbf{x}}_n$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \nabla c \left(\mathbf{w}^k \right) = \mathbf{w}^k - \alpha \frac{2}{N} \sum_{n=1}^{N} \left(\hat{\mathbf{x}}_n{}^T \mathbf{w} - y_n \right) \hat{\mathbf{x}}_n$$

Pitfall: For various features of dissimilar magnitudes, you might face convergence issues

$\rightarrow$ normalize your features: $z = \frac{x - \bar{x}}{\sigma}$ (standard normalization)

Example: Moneyball

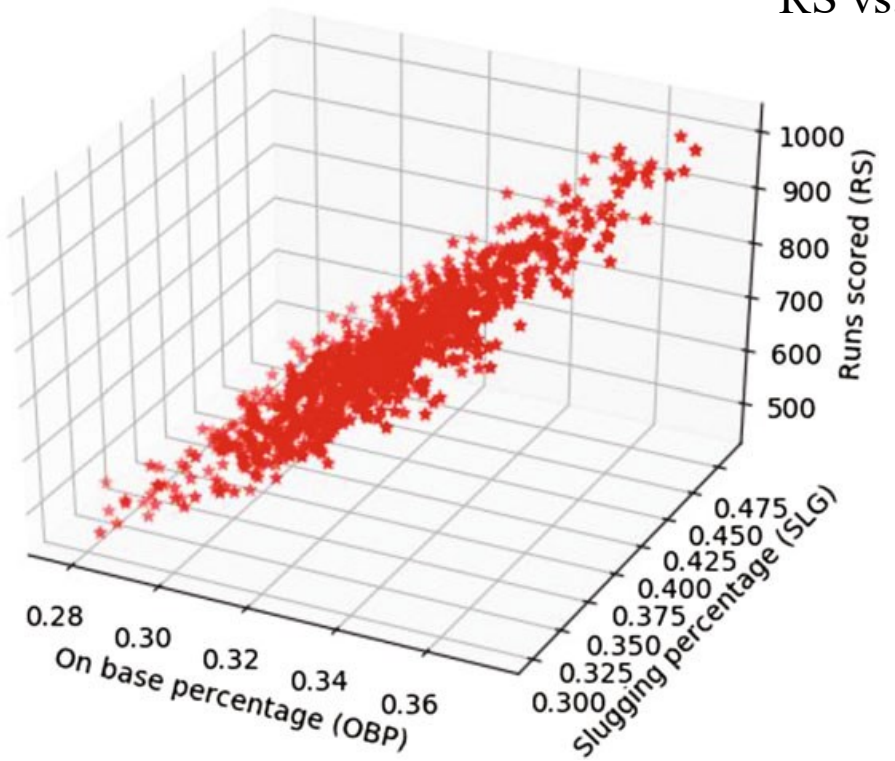
y_n	$x_{1,n}$	$x_{2,n}$
RS (y_1)	OBP	SLG
691	0.327	0.405
818	0.341	0.442
729	0.324	0.412
687	0.319	0.38
772	0.334	0.439
777	0.336	0.43
798	0.334	0.451
735	0.324	0.419
897	0.35	0.458
923	0.354	0.483

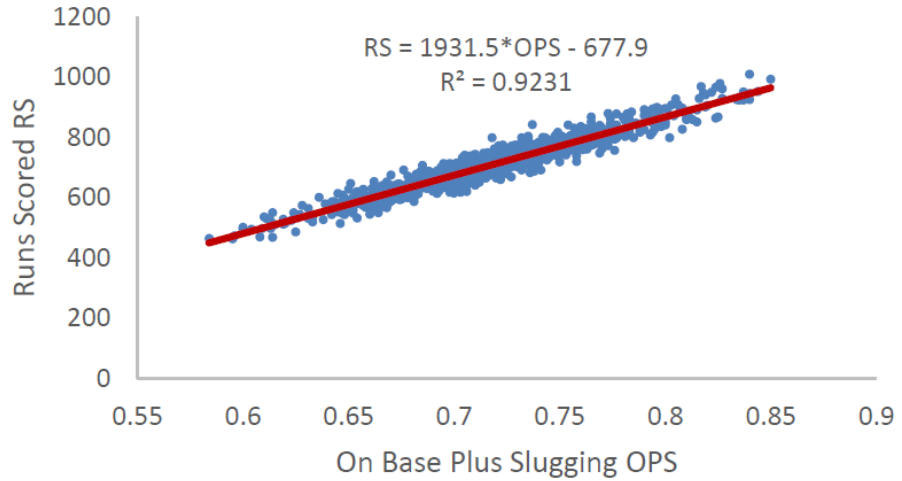
$\mathbf{x}_1$ (OBP column), n (rows)

$$w_0 + w_1 (\text{OBP})_n + w_2 (\text{SLG})_n \approx (\text{RS})_n \quad \text{for } n = 1, \ldots, N$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -803 \\ 2729 \\ 1587 \end{bmatrix} \rightarrow R^2 = 0.93$$

$$\text{RS vs. OPS}(= \text{OBP} + \text{SLG}) \rightarrow R^2 = 0.92$$





Python Code: Fig.3.24 (1)

```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Load team data
df = pd.read_csv('baseball.csv',sep=',').fillna(0)
df['OPS'] = df.OBP + df.SLG
df2002 = df.loc[df.Year < 2002]

# Linear regression for Runs scored

slBA, intBA, r_valBA, p_valBA, ste_errBA = stats.linregress(df2002.BA,df2002.RS)
rsqBA = r_valBA**2
slOBP, intOBP, r_valOBP, p_valOBP, ste_errOBP = stats.linregress(df2002.OBP,df2002.RS)
rsqOBP = r_valOBP**2
slSLG, intSLG, r_valSLG, p_valSLG, ste_errSLG = stats.linregress(df2002.SLG,df2002.RS)
rsqSLG = r_valSLG**2
slOPS, intOPS, r_valOPS, p_valOPS, ste_errOPS = stats.linregress(df2002.OPS,df2002.RS)
rsqOPS = r_valOPS**2

plt.plot(df2002.BA,df2002.RS,'.',label='BA ($r^2$=%.3f)' %rsqBA)
plt.plot(df2002.OBP,df2002.RS,'o',label='OBP ($r^2$=%.3f)' %rsqOBP)
plt.plot(df2002.SLG,df2002.RS,'.',label='SLG ($r^2$=%.3f)' %rsqSLG)
plt.plot(df2002.OPS,df2002.RS,'*',label='OPS ($r^2$=%.3f)' %rsqOPS)
plt.xlabel('Statistic')
plt.ylabel('Runs scored')
plt.legend(loc='lower right')
plt.grid()
```

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html

Python Code: Fig.3.24 (2)

```
yBA = slBA*df2002.BA + intBA
plt.plot(df2002.BA,yBA,'k-')
yOBP = slOBP*df2002.OBP + intOBP
plt.plot(df2002.OBP,yOBP, 'k-')
ySLG = slSLG*df2002.SLG + intSLG
plt.plot(df2002.SLG,ySLG, 'k-')
yOPS = slOPS*df2002.OPS + intOPS
plt.plot(df2002.OPS,yOPS, 'k-')
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df2002.OBP,df2002.SLG,df2002.RS,marker='*',color='r')
ax.set_xlabel('On base percentage (OBP)')
ax.set_ylabel('Slugging percentage (SLG)')
ax.set_zlabel('Runs scored (RS)’)

x = df2002.OBP
y = df2002.SLG
x,y = np.meshgrid(x,y)
z = -803 + 2729*x + 1587*y
# Linear regression for Wins
slWBA, intWBA, r_valWBA, p_valWBA, ste_errWBA = stats.linregress(df2002.BA,df2002.W)
```

Matlab Code: regression_multivariate.m

```
%Gradient Descent
clear all
close all
%num of iterations, learning rate, initial guess
ite = 50000;
alpha = .0005;
wa = [-1000; 1000; 1000];
%imported data, 3 columns: RS, OBP, SLG
A = readtable ('baseball.csv');
%size(A)
%a1 is OBP, a2 is SLG, a3 is RS
a1 = A.OBP;
a2 = A.SLG;
a3 = A.RS;
%length of data
spac = length(a1);
%group OBP and SLG in a matrix
Af = [a1, a2];
%define symbolic variables
syms w0 w1 w2
whold = [w1; w2];
%cost function for first point
g(w0,w1,w2) = (w0+Af(1,:)*whold - a3(1))^2;
%cost function for the rest of points (wo+x.T *w y)^2
for i=2:spac
g = g + (w0+Af(1,:)*whold - a3(i))^2;
end
%take the grad of g
gradd(w0,w1,w2) = gradient(g,[w0,w1,w2]);
%loop through iterations
for i=1:ite
%value of gradient at wa
bloop1 = double(gradd(wa(1),wa(2),wa(3)));
%gradient descent, new value of w
wnew = wa - alpha*bloop1;
%update wa
wa = wnew;
end
wnew
```

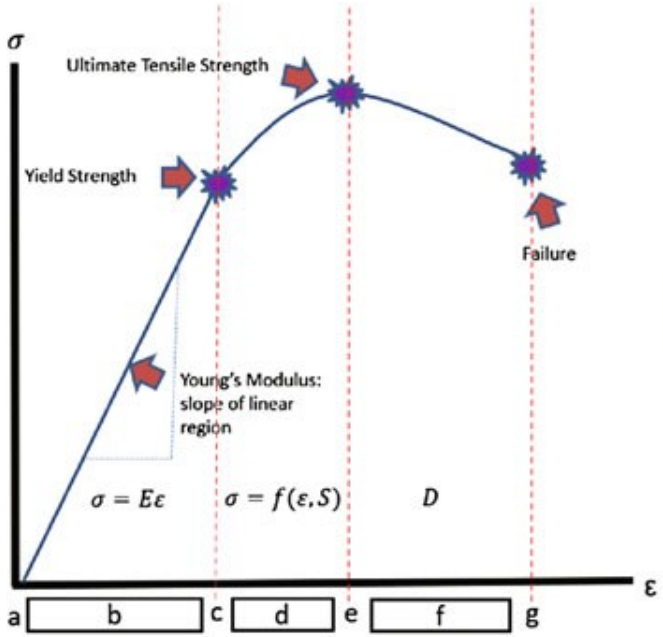
Matlab: Statistics and Machine Learning Toolbox

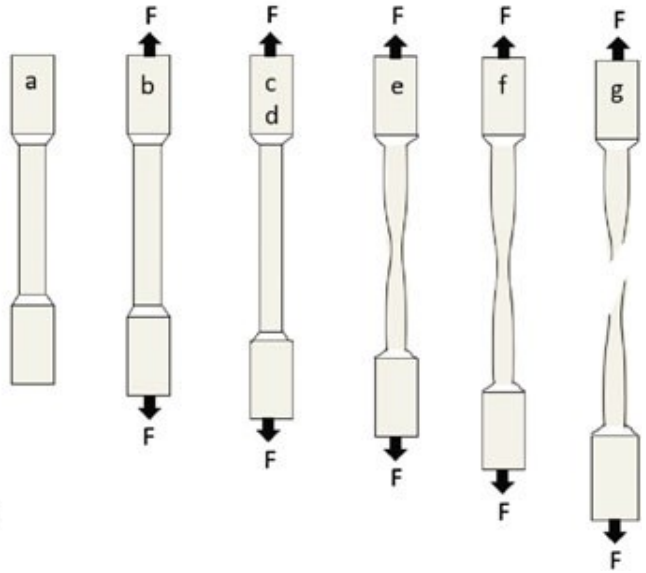
- regress
- lasso

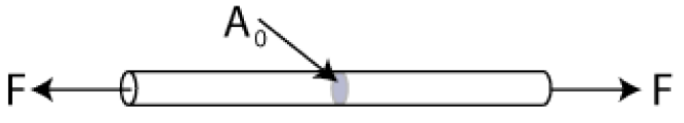
Example: Indentation for Material Hardness and Strength

- Stress-strain curve

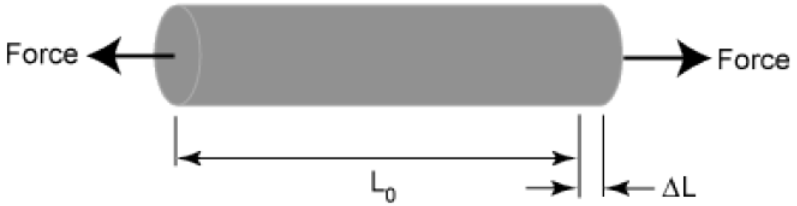
Why stress instead of force?







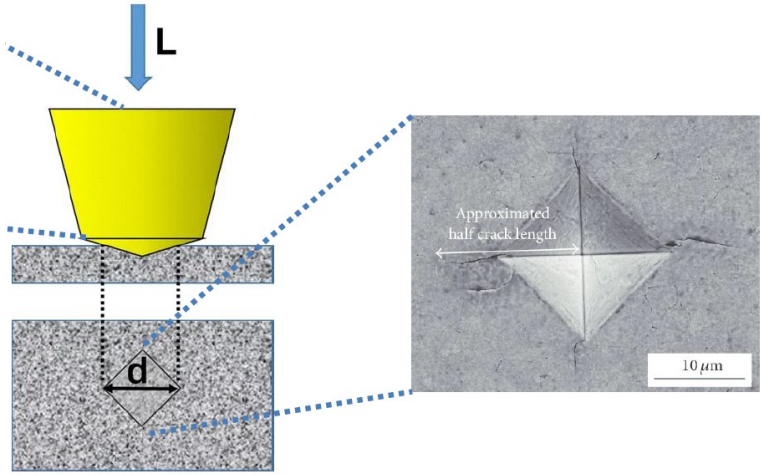
$$\text{Stress, } \sigma = \frac{\text{Force}}{\text{Cross-Sectional Area}} = \frac{F}{A_0}$$



$$\text{Strain} = \frac{\text{Elongation}}{\text{Orininal Length}} = \frac{\Delta L}{L_0}$$

- Vickers Hardness (HV)

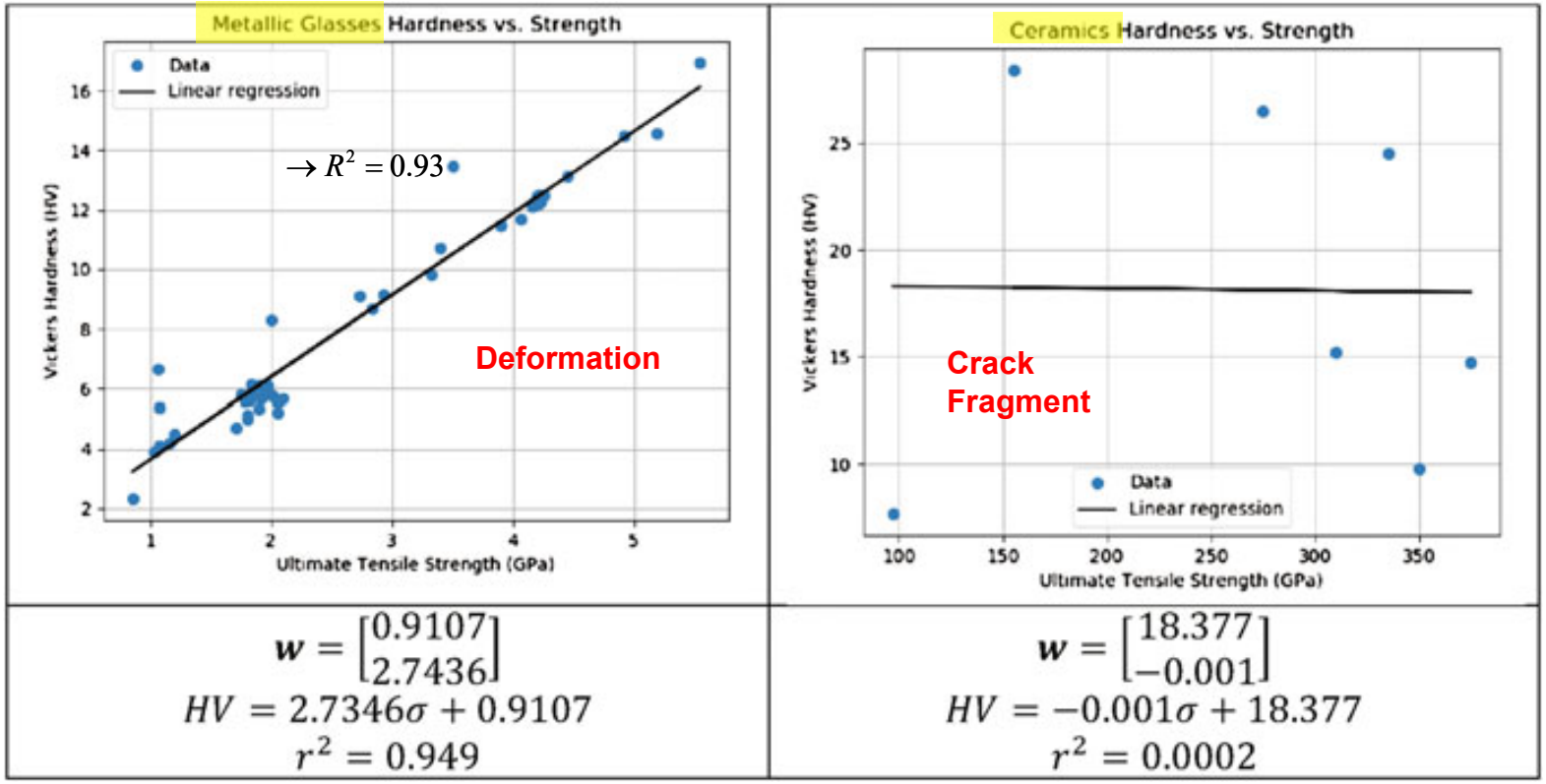
$$HV = \frac{F}{A} = \frac{F}{\frac{d^2}{2\sin(68°)}} = 1.8544\frac{F}{d^2}$$



Example: Indentation for Material Hardness and Strength

- Previous linear empirical equations in literature have been found to accurately relate Vickers Hardness (HV) and yield strength (σ) for some materials
- Why there should be a relationship between HV and σ?

Metallic glass Material	Ultimate Tensile Strength (Gpa)	Vickers Hardness (HV)
Zr52.5Ni14.6Al10Cu17.9Ti5	1.8	5.15
(Co0.942Fe0.058)69Nb3B22.4Si5.6	3.32	9.82
Zr55Cu30Al10Ni5	1.8	5
Pd40Ni40P20	1.78	5.6
Fe41Co7Cr15Mo14C15B6Y2	3.5	13.45
Fe74Ni9Cr4Si3B10	2.93	9.16
Fe66Ni7Zr6Cr8Si3B10	2	8.31
Fe63Ni7Zr6Cr8W3Si3B10	2.73	9.1
Zr53Cu30Ni9Al8	2.05	5.22
(Zr53Cu30Ni9Al8)99.75Si0.25	2.05	5.54
(Zr53Cu30Ni9Al8)99.5Si0.5	1.82	5.64
(Zr53Cu30Ni9Al8)99.25Si0.75	2.1	5.67
(Zr53Cu30Ni9Al8)99Si	1.75	5.82
Zr41.2Ti13.8Cu12.5Ni10Be22.5	1.95	5.95
Zr-400°C×5min	1.97	6.1



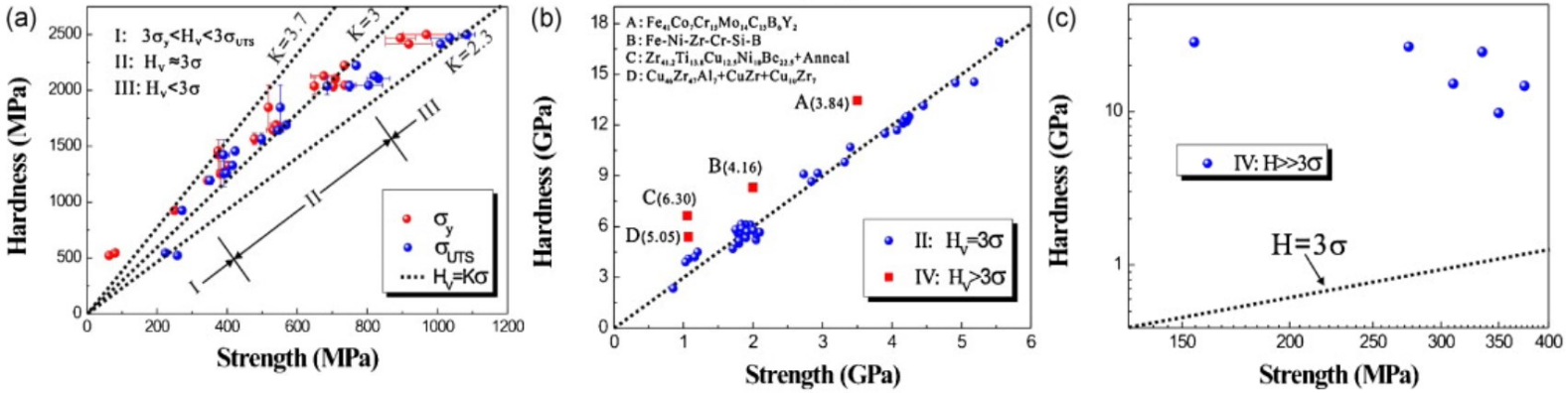
$$\boldsymbol{w} = \begin{bmatrix} 0.9107 \\ 2.7436 \end{bmatrix}$$
$$HV = 2.7346\sigma + 0.9107$$
$$r^2 = 0.949$$

$$\boldsymbol{w} = \begin{bmatrix} 18.377 \\ -0.001 \end{bmatrix}$$
$$HV = -0.001\sigma + 18.377$$
$$r^2 = 0.0002$$

Zhang, P. et al. (2011). General relationship between strength and hardness. Materials Science and Engineering: A, 529, 62-73.

Empirical relationship between hardness and strength

- Hv~3σ_y ⇔ Hardness~3x(Yield Strength)
- Is this relationship supported by all materials?



The relationship between strength and hardness in: (a) Cu and Cu–Zn alloys with different pretreatment; (b) metallic glasses; (c) ceramics.

Tensile strength	Vickers hardness	Brinell hardness	Rockwell hardness							
MPa	HV10	HBa	HRB	HRF	HRC	HRA	HRD	HR15N	HR30N	HR45N
1030	320	304	–	–	32,2	66,4	49,4	76,2	52,3	33,9
1060	330	314	–	–	33,3	67	50,2	76,8	53,6	35,2
1095	340	323	–	–	34,4	67,6	51,1	77,4	54,4	36,5
1125	350	333	–	–	35,5	68,1	51,9	78	55,4	37,8
1155	360	342	–	–	36,6	68,7	52,8	78,6	56,4	39,1
1190	370	352	–	–	37,7	69,2	53,6	79,2	57,4	40,4
1220	380	361	–	–	38,8	69,8	54,4	79,8	58,4	41,7
1255	390	371	–	–	39,8	70,3	55,3	80,3	59,3	42,9
1290	400	380	–	–	40,8	70,8	56	80,8	60,2	44,1
1320	410	390	–	–	41,8	71,4	56,8	81,4	61,1	45,3
1350	420	399	–	–	42,7	71,8	57,5	81,8	61,9	46,4
1385	430	409	–	–	43,6	72,3	58,2	82,3	62,7	47,4
1420	440	418	–	–	44,5	72,8	58,8	82,8	63,5	48,4
1455	450	428	–	–	45,3	73,3	59,4	83,2	64,3	49,4
1485	460	437	–	–	46,1	73,6	60,1	83,6	64,9	50,4
1520	470	447	–	–	46,9	74,1	60,7	83,9	65,7	51,3
1555	480	456	–	–	47,7	74,5	61,3	84,3	66,4	52,2
1595	490	466	–	–	48,4	74,9	61,6	84,7	67,1	53,1
1630	500	475	–	–	49,1	75,3	62,2	85	67,7	53,9
1665	510	485	–	–	49,8	75,7	62,9	85,4	68,3	54,7
1700	520	494	–	–	50,5	76,1	63,5	85,7	69	55,6
1740	530	504	–	–	51,1	76,4	63,9	86	69,5	56,2
1775	540	513	–	–	51,7	76,7	64,4	86,3	70	57
1810	550	523	–	–	52,3	77	64,8	86,6	70,5	57,8
1845	560	532	–	–	53	77,4	64,4	86,9	71,2	58,6
1880	570	542	–	–	53,6	77,8	65,8	87,2	71,7	59,3
1920	580	551	–	–	54,1	78	66,2	87,5	72,1	59,9
1955	590	561	–	–	54,7	78,4	66,7	87,8	73,7	60,5
1995	600	570	–	–	55,2	78,6	67	88	73,2	61,2
2030	610	580	–	–	55,7	78,9	67,5	88,2	73,7	61,7
2070	620	589	–	–	56,3	79,2	67,9	88,5	74,2	62,4
2105	630	599	–	–	56,8	79,5	68,3	88,8	74,6	63
2145	640	608	–	–	57,3	79,8	68,7	89	75,1	63,5
2180	650	618	–	–	57,8	80	69	89,2	75,5	64,1
–	660	–	–	–	58,3	80,3	69,4	89,5	75,9	64,7
–	670	–	–	–	58,8	80,6	69,8	89,7	76,4	65,3
–	680	–	–	–	59,2	80,8	70,1	89,8	76,8	65,7
–	690	–	–	–	59,7	81,1	70,5	90,1	77,2	66,2
–	700	–	–	–	60,1	81,3	70,8	90,3	77,6	66,7
–	720	–	–	–	61,5	81,8	71,5	90,7	78,4	67,7
–	740	–	–	–	61,8	82,2	72,1	91	79,1	68,6
–	760	–	–	–	62,5	82,6	72,6	91,2	79,7	69,4
–	780	–	–	–	63,3	83	73,3	91,5	80,4	70,2
–	800	–	–	–	64	83,4	73,8	91,8	81,1	71
–	820	–	–	–	64,7	83,8	74,3	92,1	81,7	71,8
–	840	–	–	–	65,3	84,1	74,8	92,3	82,2	72,2
–	860	–	–	–	65,9	84,4	75,3	92,5	82,7	73,1
–	880	–	–	–	66,4	84,7	75,7	92,7	83,1	73,6
–	900	–	–	–	67	85	76,1	92,9	83,6	74,2
–	920	–	–	–	67,5	85,3	76,5	93	84	74,8
–	940	–	–	–	68	85,6	75,9	93,2	84,4	75,4

Nonlinear Regression: Piecewise Linear Regression

- Subdividing a set of nonlinear data into a series of segments that are approximately linear

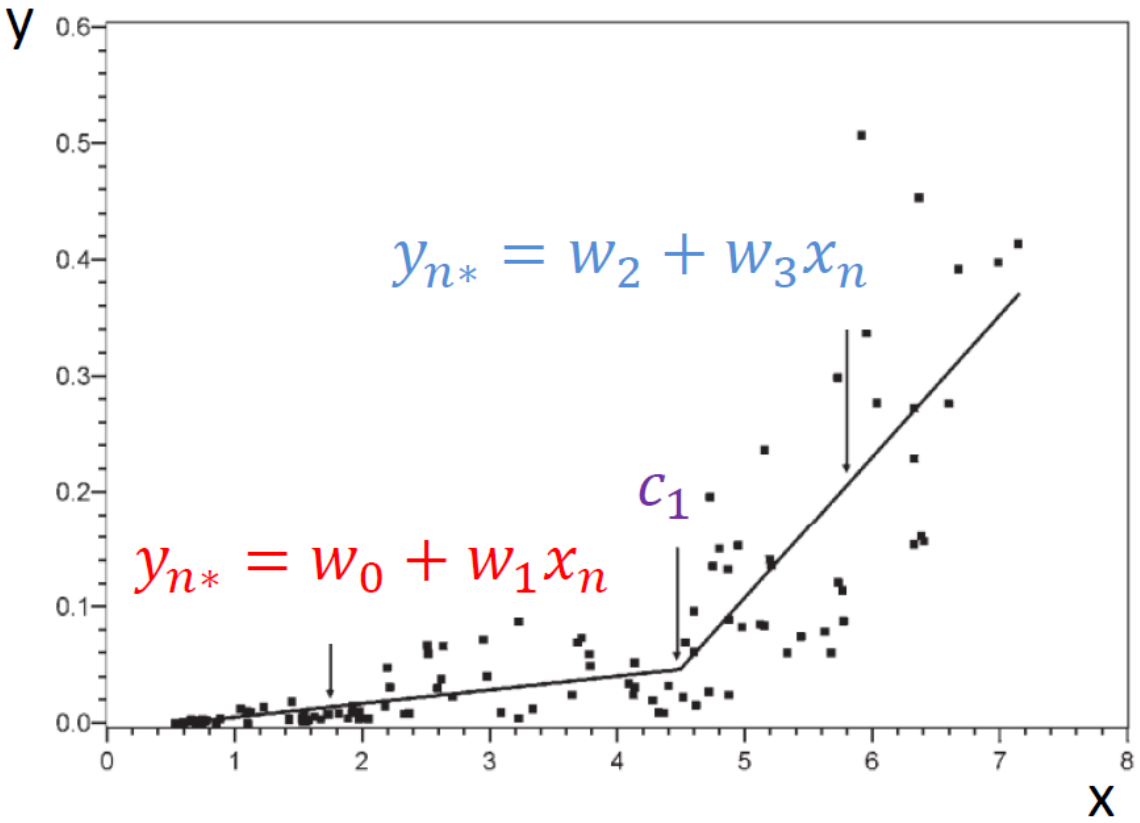
Split the data into two sets: N_1 and N_2

$$\left.\begin{aligned} y_n^* &= w_0 + w_1 x_n \quad (x < c_1) \\ y_n^* &= w_2 + w_3 x_n \quad (x > c_1) \end{aligned}\right\}$$

$$\xrightarrow[c_1 = -\frac{w_0 - w_2}{w_1 - w_3}]{w_0 + w_1 c_1 = w_2 + w_3 c_1} \begin{cases} y_n^* = w_0 + w_1 x_n & (x < c_1) \\ y_n^* = w_0 + (w_1 - w_3) c_1 + w_3 x_n & (x > c_1) \end{cases}$$

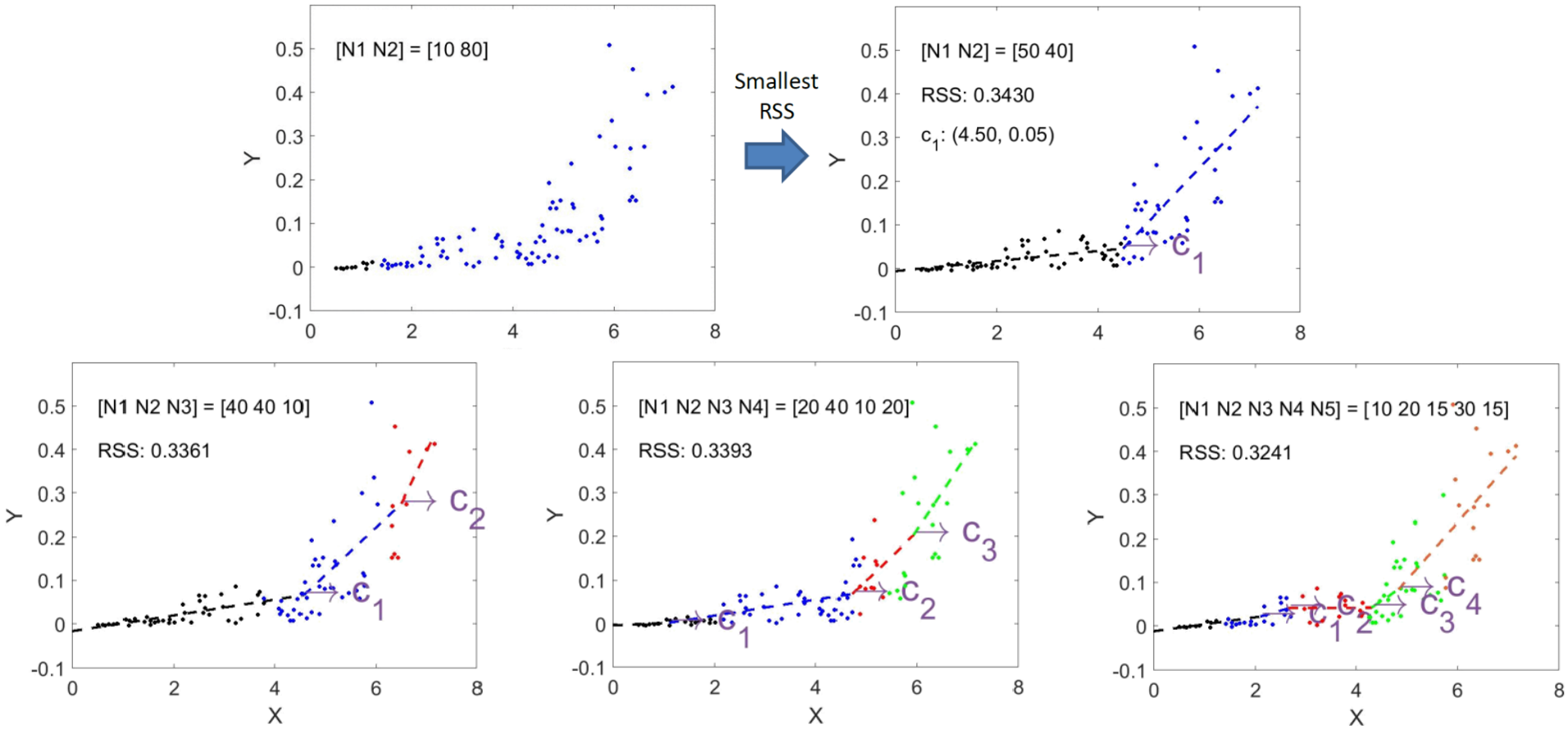
residual sum of squares (RSS)

$$c(\mathbf{w}) = \frac{1}{N_1} \sum_{n=1}^{N_1} \left(y_n^* - y_n\right)^2 + \frac{1}{N_2 - N_1} \sum_{n=N_1+1}^{N_2} \left(y_n^* - y_n\right)^2$$



Nonlinear Regression: Piecewise Linear Regression

- How can we choose the split point?
  - Use residual sum of squares (RSS) as a cost function

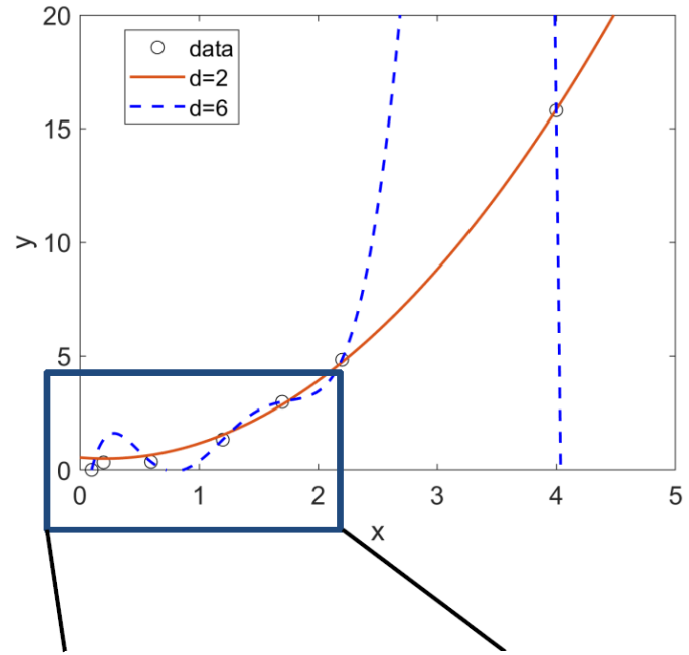


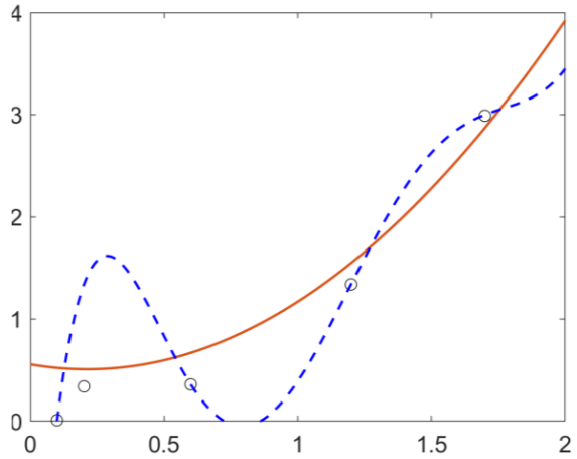
Polynomial Nonlinear Features

- We can increase our feature space by taking increasingly higher degrees of polynomials
- Feature space: The n dimensions where the variables exist

$$\begin{cases} \text{1-feature } x_n\text{, } d\text{-degree polynomial features: } 1, x_n, {x_n}^2, \ldots, {x_n}^{d-1}, {x_n}^d \\ \text{polynomial basis: } \mathbf{p}\left(x_n\right)^T = \left[1, x_n, {x_n}^2, \ldots, {x_n}^{d-1}, {x_n}^d\right] \\ \text{polynomial model: } y_n^* = \mathbf{p}\left(x_n\right)^T \mathbf{w} = \left[1, x_n, {x_n}^2, \ldots, {x_n}^{d-1}, {x_n}^d\right] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \\ d = 1 \rightarrow \text{linear} \end{cases}$$

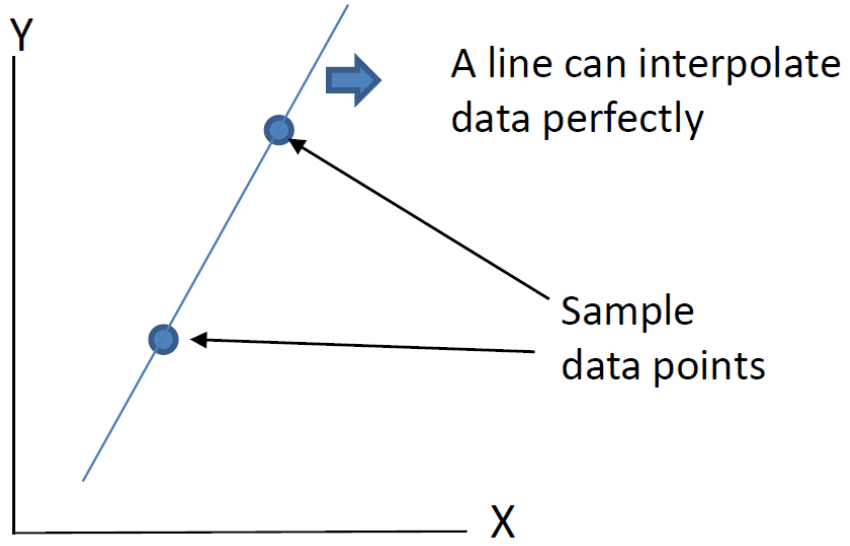
- Pitfall: For an increasingly higher d degree, you can start overfitting
- Overfitting: A model that fits the training data too well and therefore lacks generality.

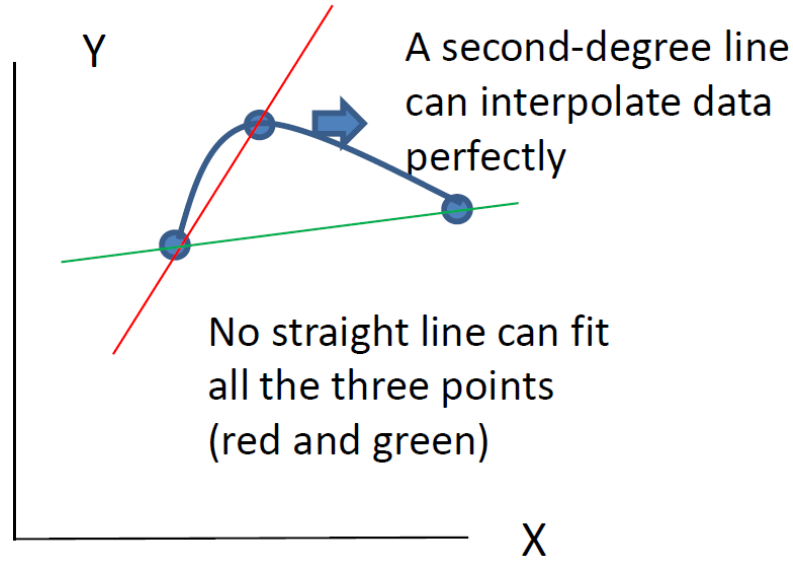




Higher Order Interpolation May Lead to Overfitting

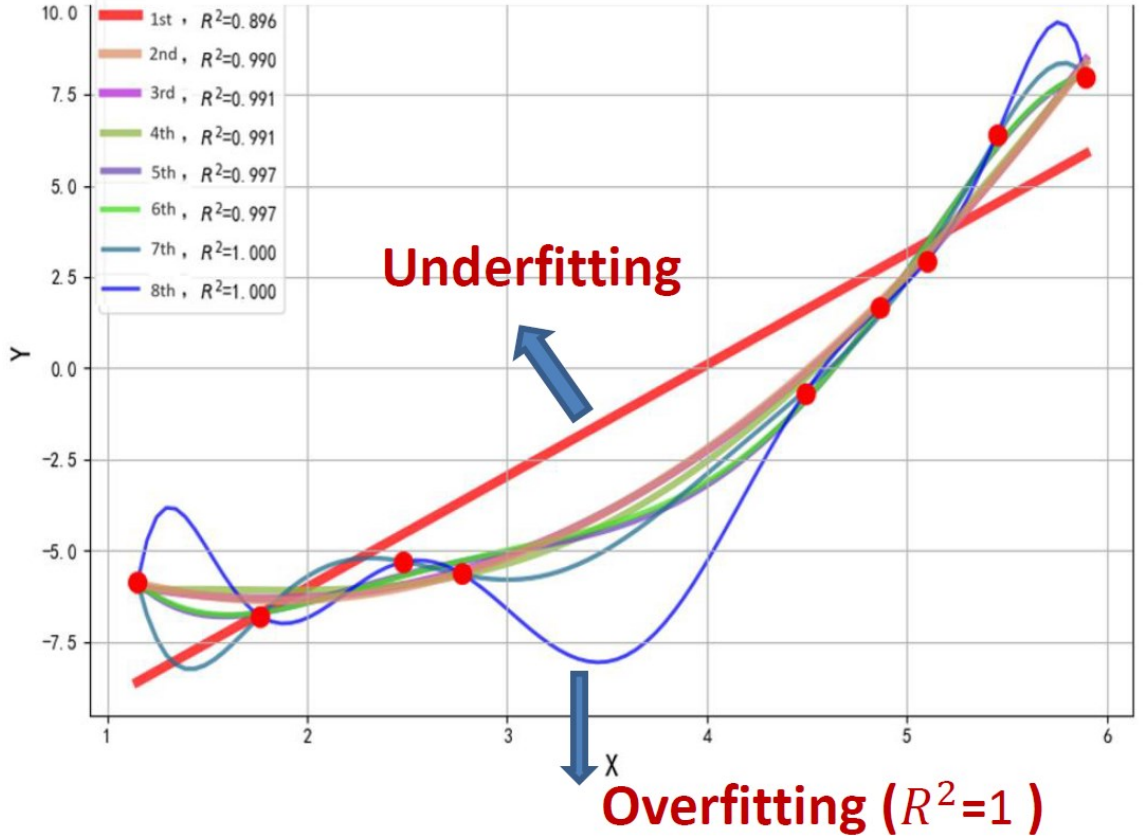
- Suppose you have a set N data points (x(i),y(i)) in the plane where no two x(i) are the same. Then there exists a polynomial P of degree N−1 or less which perfectly interpolates the data points. That is, P(x(i))=y(i) for all I
- According to the theorem, for two points there exists a line and for three points there exists a quadratic polynomial that perfectly fits the data points.
- But if we have a large number of data say 30,000, should we use a very high degree polynomial to fit it? The answer is NO.





Avoid High Order Polynomials

- By increasing the order of regression model, we can have more accurate regression result (regression curve corresponds too closely to data).
- This causes “Overfitting” and "Runge's Phenomenon oscillation".

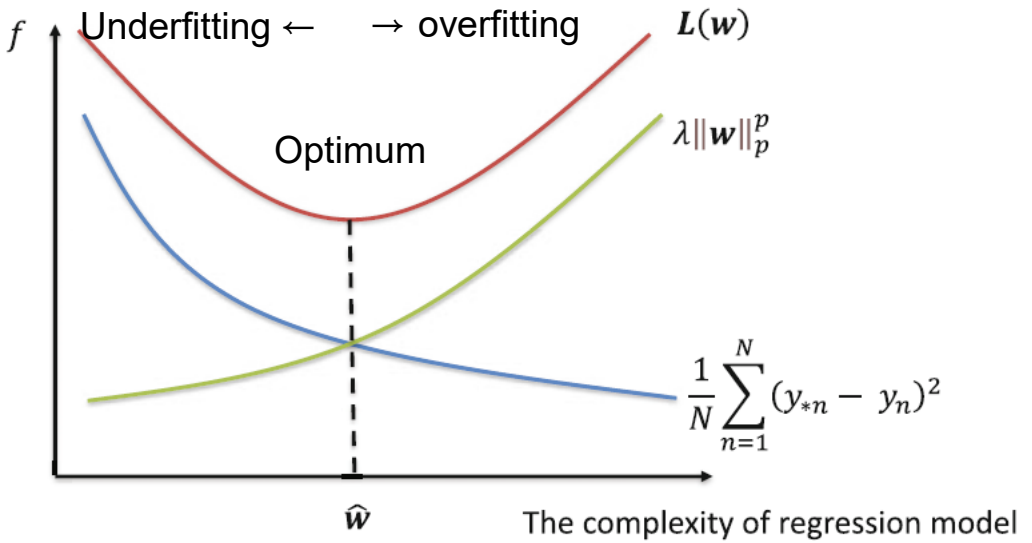


Use Regularization To Avoid Overfitting

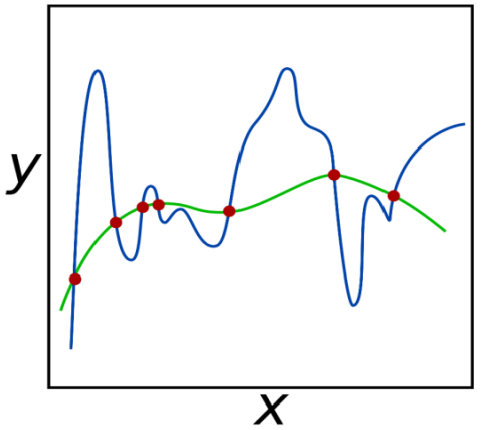
- Find a good balance between model complexity and accuracy
  - complicated, higher order regression models to achieve accuracy → may lose generality for the regression models
- Regularized loss function
  - By tuning λ a model can be pushed to converge to the actual function
  - Larger λ: more simplicity, smaller λ: more accuracy
  - How to choose λ?

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\|\mathbf{w}\|_p \quad \text{where} \quad \|\mathbf{w}\|_p = \left(\sum_{j=1}^{N}|w_j|^p\right)^{1/p}$$

$$\begin{cases} \lambda: \text{ predefined regularization parameter with nonnegative value} \\ y_n: \text{ original data} \\ y_n^*: \text{ regression model} \\ N: \text{ number of data points} \end{cases}$$



Regularization Can Avoid Overfitting



- We consider the green curve to be smoother an “easier" path to traverse in comparison to the blue curve. Smaller weights will get us better or smoother results. By tuning λ a model can be pushed to converge at the green curve.
- Think of regularization in two ways
  - You are penalizing high weights by adding a positive term to the cost function. The higher the magnitude of w the higher the cost will be.
  - By adjusting λ, you can modify the cost function to achieve convergence to a minimum.

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\left\|\mathbf{w}\right\|_{p=2}^2 \xrightarrow{\lambda\to\infty} L(\mathbf{w}) \approx \lambda\left\|\mathbf{w}\right\|_{p=2}^2 \ (\text{convex})$$

$$\begin{cases} L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\left\|\mathbf{w}\right\|_1 = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\sum_{n=1}^{S}\left|w_n\right| : \text{LASSO}(\text{least absolute shrinkage and selection operator}) \\ L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\left\|\mathbf{w}\right\|_2 = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\sum_{n=1}^{S}\left|w_n\right|^2 : \text{Ridge} \\ L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda\left(\alpha\left\|\mathbf{w}\right\|_1 + (1-\alpha)\left\|\mathbf{w}\right\|_2\right) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \beta_1\left\|\mathbf{w}\right\|_1 + \beta_2\left\|\mathbf{w}\right\|_2 : \text{Elastic Net} \end{cases}$$

L_p-Norm

- L1-norm can be used to relieve overfitting: eliminate some high order terms in the regression model (may omit the intricate details)
- L2-norm uses the concept of "sum of squares", and thus has useful properties such as convexity, smoothness and differentiability (capture those details)

$$\|\mathbf{w}\|_p = \left(\sum_{j=1}^{N} |w_j|^p \right)^{1/p}$$

$$p = 1: \quad \|\mathbf{w}\|_1 = \sum_{j=1}^{N} |w_j| = |w_1| + |w_2| + \cdots + |w_N|$$

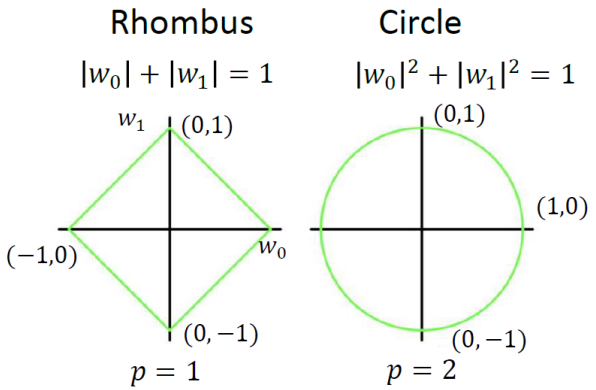
$$p = 2: \quad \|\mathbf{w}\|_2 = \left(\sum_{j=1}^{N} |w_j|^2 \right)^{1/2} = \sqrt{|w_1|^2 + |w_2|^2 + \cdots + |w_N|^2}$$

$$p = \infty: \quad \|\mathbf{w}\|_\infty = \max\left(|w_j| \right) = \max\left(|w_1|, |w_2|, \ldots, |w_N| \right)$$

$$N = 2, \ \|\mathbf{w}\|_p = 1$$

$\|\boldsymbol{w}\|_{p=1}$ Rhombus $|w_0| + |w_1| = 1$

$\|\boldsymbol{w}\|_{p=2}$ Circle $|w_0|^2 + |w_1|^2 = 1$

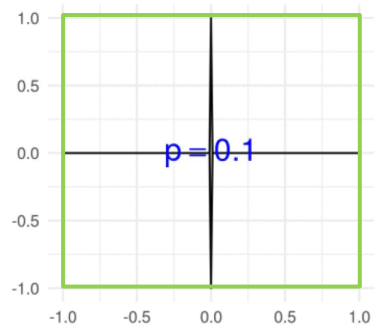


L1 norm L2 norm

Increase *p*

$\|\omega\|_\infty = \max(|\omega_j|)$

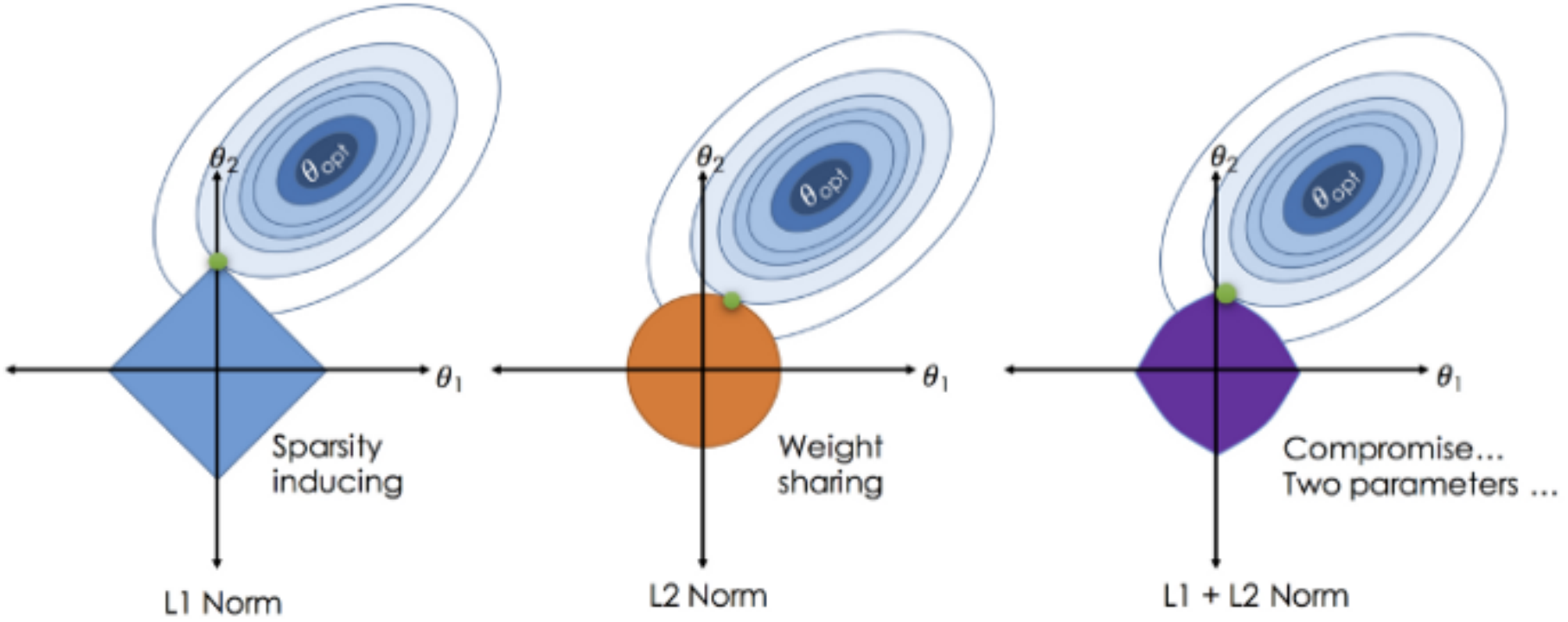
Square



Convergence to L∞ norm

Regularization: Geometric Interpretation

Regularization Term	Method
L1 norm	LASSO
L2 norm	Ridge regression
L1 + L2 norm	Elastic net



Example: Regularized Regression

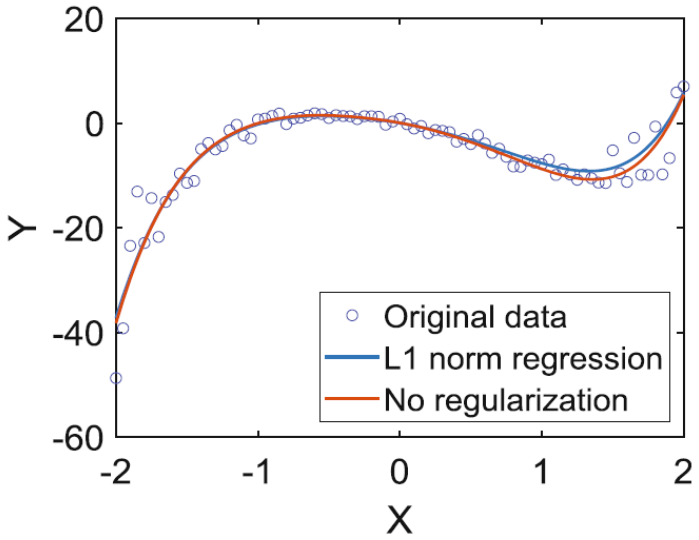
$$y = 1(x+\varepsilon_1)^5 - 4(x+\varepsilon_2)^2 - 5(x+\varepsilon_3) \quad \text{where} \quad \varepsilon_i \sim Normal(0, 0.05), \text{ Gaussian noise}$$

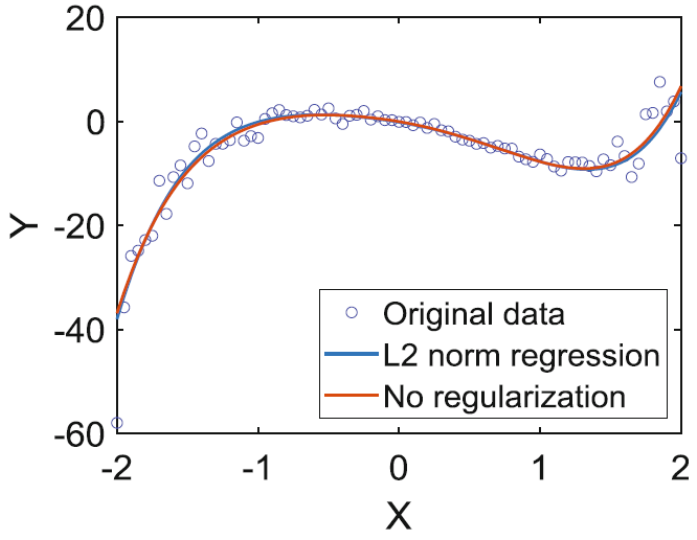
$$y_n^* = w_0 + w_1 x_n + w_2 {x_n}^2 + \cdots + w_5 {x_n}^5$$

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 \rightarrow \mathbf{w} = \begin{bmatrix} -0.0018 & -4.8456 & -4.5166 & -0.1003 & 0.2190 & 1.0498 \end{bmatrix}^T \text{ (MATLAB: polyfit)}$$

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda \|\mathbf{w}\|_{p=1} \xrightarrow{\lambda=0.074} \mathbf{w} = \begin{bmatrix} 0 & -4.6723 & -3.9175 & 0 & 0 & 0.9671 \end{bmatrix}^T$$

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2 + \lambda \|\mathbf{w}\|_{p=2} \xrightarrow{\lambda=1} \mathbf{w} = \begin{bmatrix} -0.0707 & -5.3544 & -3.7368 & 0.0495 & -0.0800 & 1.0119 \end{bmatrix}^T$$

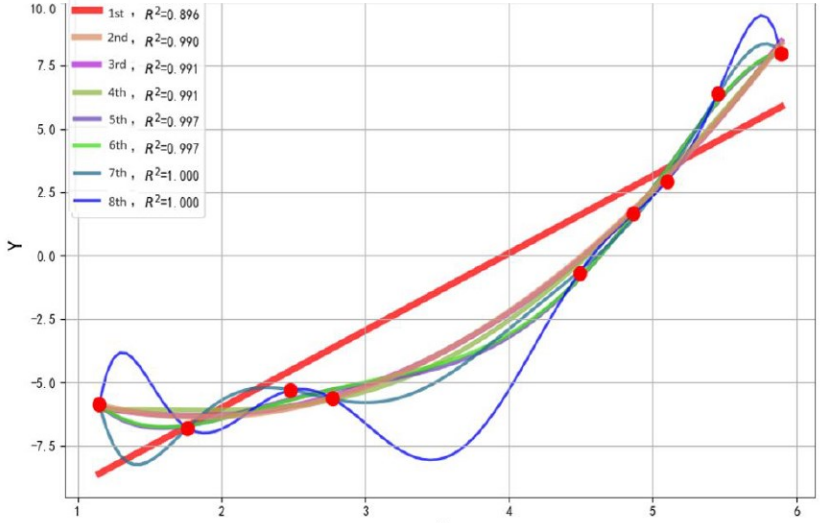




Comparison

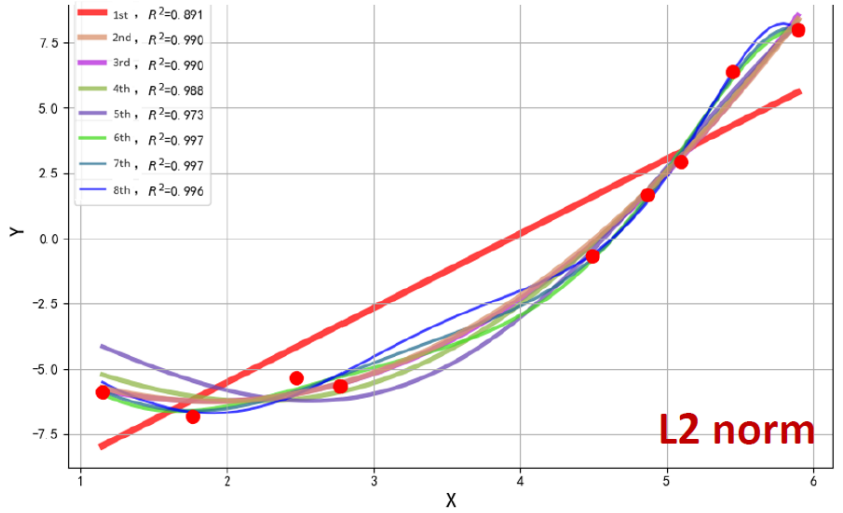
- Lasso (L1 norm) regression
  - Better performance in feature selection
- Ridge (L2 norm) regression
  - Can preserve details and detect sophisticated patterns in data

Example: Comparison of Regularization

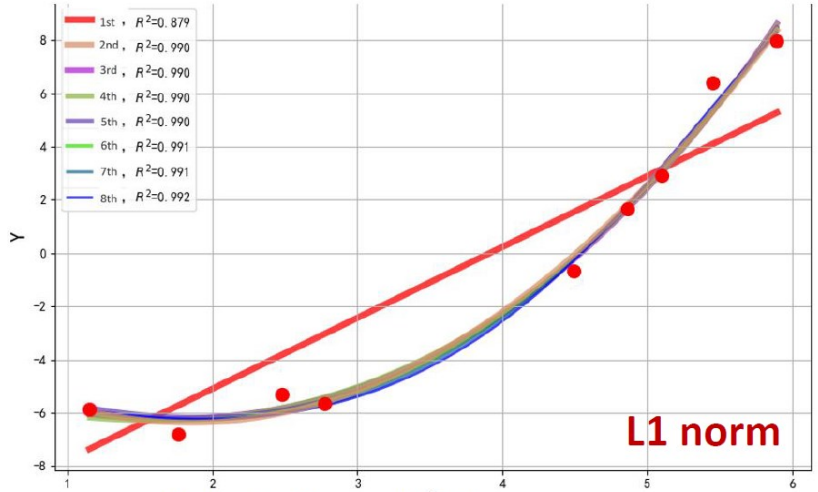


Overfitting.

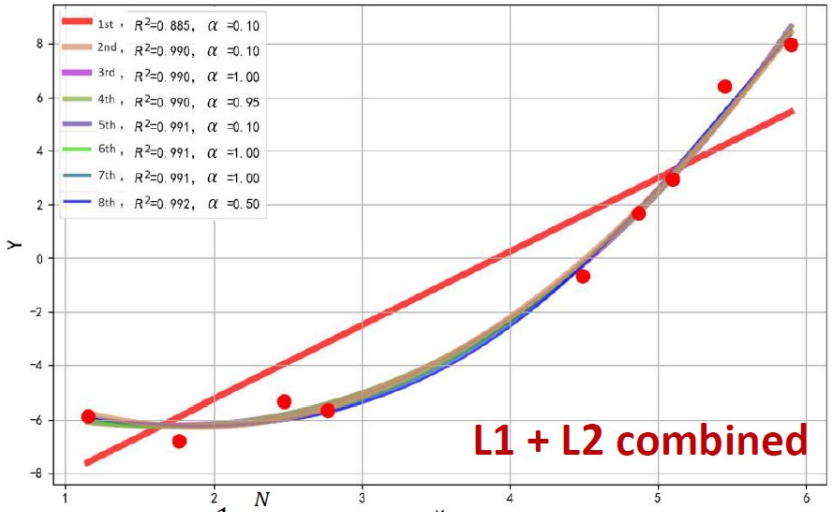
Without regularization



Oscillation is alleviated.



Fitting is clean with L1 norm.



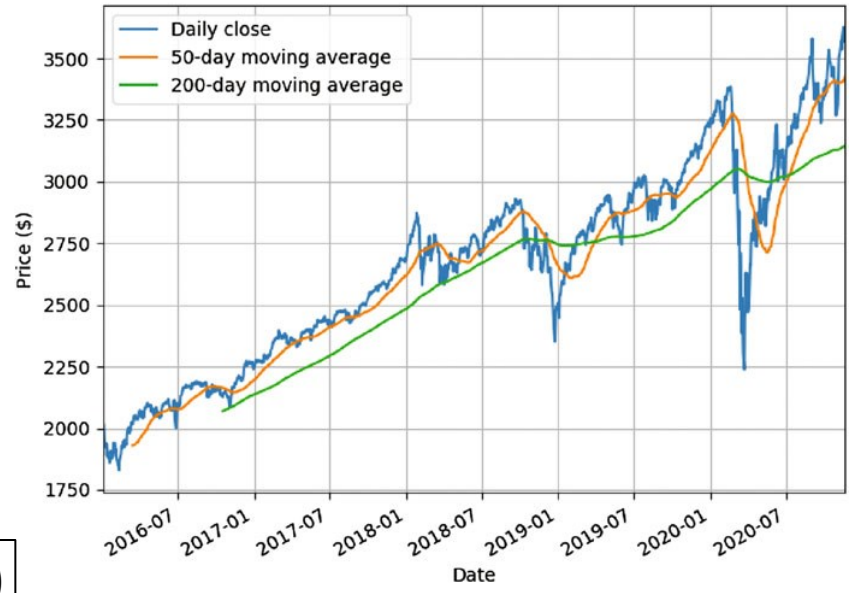
$$c(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_{n*} - y_n)^2 + \lambda(\alpha||\boldsymbol{\omega}||_1 + (1-\alpha)||\boldsymbol{\omega}||_2^2)$$

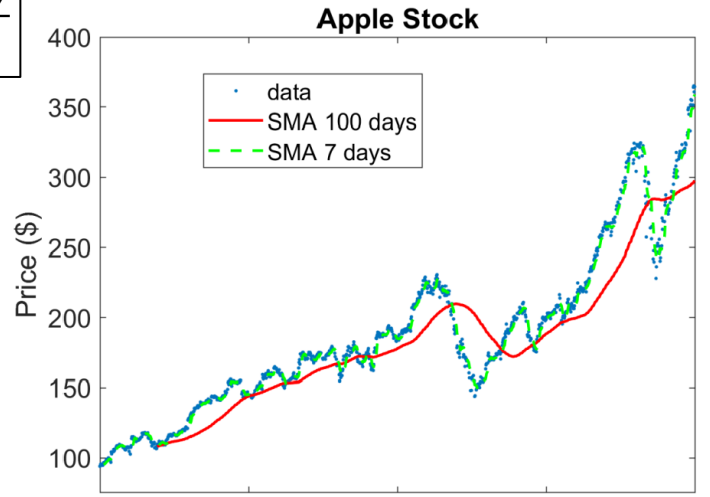
Nonlinear Regression: Moving Average

- Good method to smooth out data and mute the effects of spikes in the data
  - analyzing trends with stock prices in order to smooth out the effects of day to day movement of the stock price (volatile or downturn)
  - 200-day can also act as a “floor” or lower limit—buying opportunities exist when the price drops down to that level or below
- Simple Moving Average (SMA)

$$\text{SMA} = \frac{\sum \left(P_c + P_{c-1} + \cdots + P_{c-k} \right)}{k}$$

  - sum of the stock price for the previous k amount of days divided by k
- Appearance of being off of the original data
  - at the average of the independent variable instead of at the extent





Python code: Fig.3.20

```
#!pip install yfinance
import yfinance as yf
#import pandas_datareader.data as web

start = '2016-01-01'
#df2 = web.DataReader('^GSPC', 'yahoo',start)
df2 = yf.download('^GSPC', start=start)
df2.to_csv('gspc.csv')
df2['Close'].plot()
df2['Close'].rolling(50).mean().plot()
df2['Close'].rolling(200).mean().plot()
plt.legend(['Daily close','50-day moving average','200-
day moving average'])
plt.ylabel('Price ($)')
plt.grid()
plt.show()
```

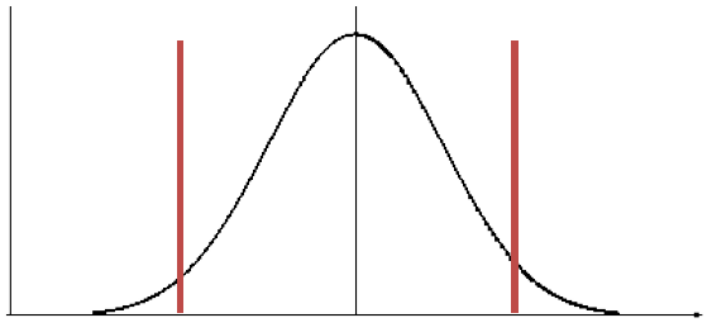
Nonlinear Regression: Moving Least Squares

- So far, we have considered all our data of equal importance. However, we might want to place more weight on certain data points for a variety of reasons including:
  - Emphasize data points closer to our point of interest
  - Minimize fitting towards outliers
- Weight function results in a localized point-by-point least square fit instead of a global least squares fit
  - Weighting functions move so that the regression is always being done with a few of the data points

$$c(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta(x_n)\left(y_n^*(\mathbf{w}) - y_n\right)^2$$

$\theta = 1$: regular least squares

Typical weighting function



Example: Apple Stock

- Original 252 data points
- MLS: only 45 evenly spaced points
- Cubic spline with a coverage radius of 3

Linear regression (one line through all data)

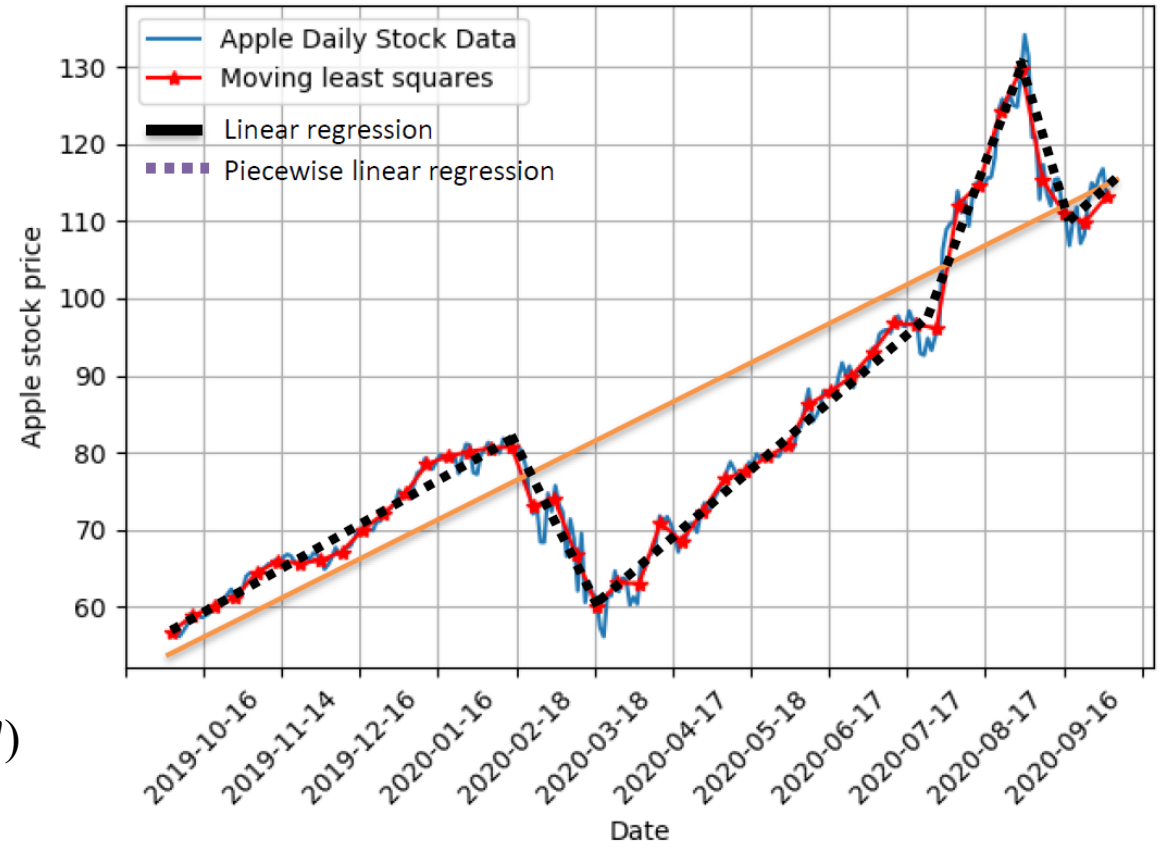
$$c(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n^* - y_n\right)^2$$

Piecewise Linear regression (multiple lines through data)

$$c(\mathbf{w}) = \frac{1}{N_1}\sum_{n=1}^{N_1}\left(y_n^* - y_n\right)^2 + \frac{1}{N_2}\sum_{m=1}^{N_2}\left(y_m^* - y_m\right)^2 + \cdots$$

Weighted regression ("bend" line through data using "weight")

$$c(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\theta(x_n)\left(y_n^* - y_n\right)^2$$



1) Local influence only choose a few points at a time
2) Often use a bell-shaped curve: focus on center point, taper down at edges

Moving Least Squares (MLS) Approximation

$$y_n^* = \mathbf{p}(x_n)^T \mathbf{w}(x) = \left[1, x_n, x_n^2, \ldots, x_n^{d-1}, x_n^d\right] \begin{bmatrix} w_0(x) \\ w_1(x) \\ \vdots \\ w_d(x) \end{bmatrix} = \mathbf{p}(\mathbf{x})^T \mathbf{w}(\mathbf{x})$$

$$c(\mathbf{w}(\mathbf{x})) = \sum_{n=1}^{N} \theta(\mathbf{x} - \mathbf{x}_n)\left(\mathbf{p}(\mathbf{x}_n)^T \mathbf{w}(\mathbf{x}) - y_n\right)^2$$

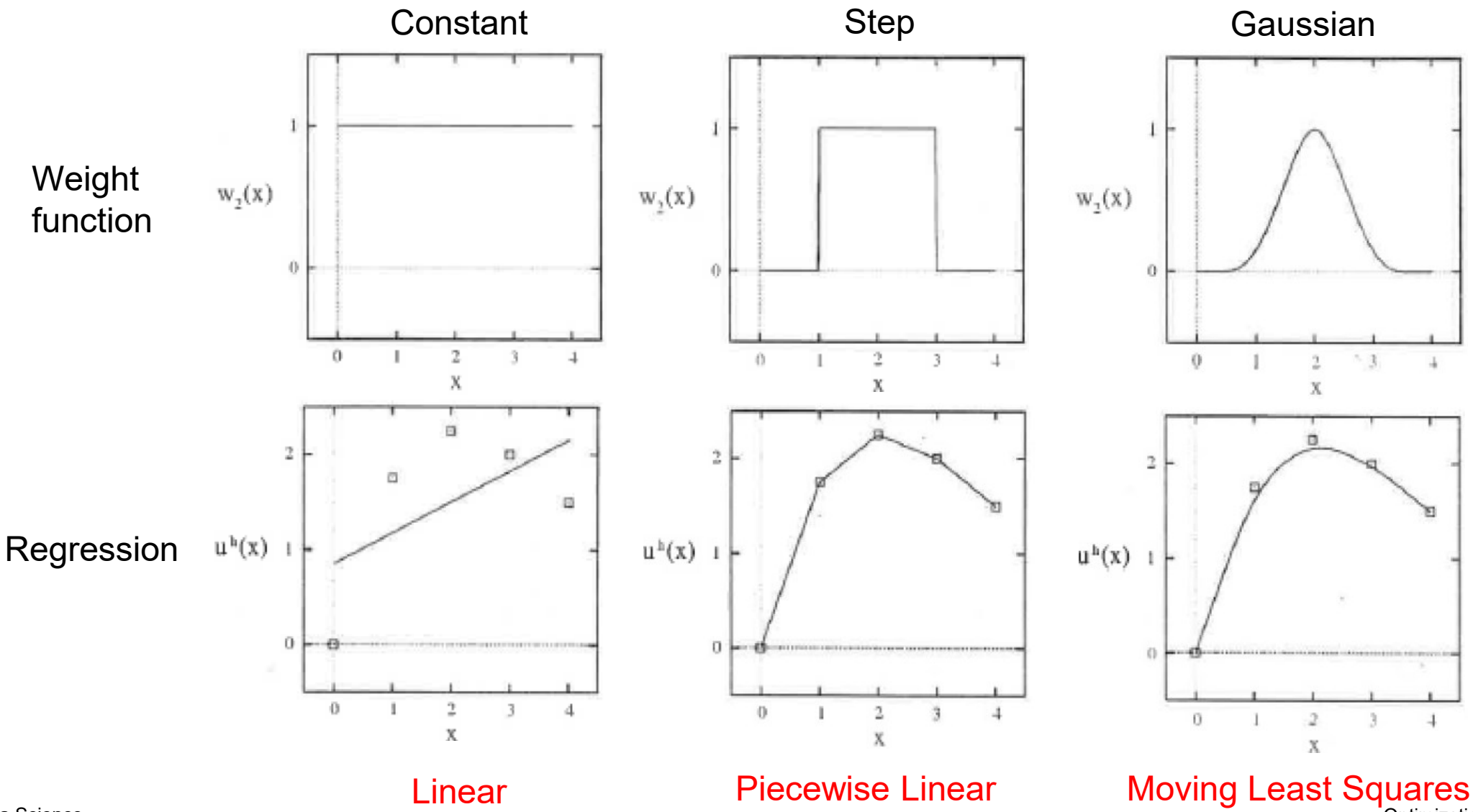
$$\frac{\partial c}{\partial \mathbf{w}(\mathbf{x})} = \sum_{n=1}^{N} \theta(\mathbf{x} - \mathbf{x}_n)\left(\mathbf{p}(\mathbf{x}_n)^T \mathbf{w}(\mathbf{x}) - y_n\right)\mathbf{p}(\mathbf{x}_n) = 0$$

$$\underbrace{\sum_{n=1}^{N} \theta(\mathbf{x} - \mathbf{x}_n)\mathbf{p}(\mathbf{x}_n)\mathbf{p}(\mathbf{x}_n)^T}_{\mathbf{A}(\mathbf{x})} \mathbf{w}(\mathbf{x}) = \underbrace{\sum_{n=1}^{N} \theta(\mathbf{x} - \mathbf{x}_n)\mathbf{p}(\mathbf{x}_n)}_{\mathbf{B}_n(\mathbf{x})} \mathbf{y} \rightarrow \mathbf{w}(\mathbf{x}) = \mathbf{A}(\mathbf{x})^{-1} \mathbf{B}_n(\mathbf{x}) \mathbf{y}$$

$$y_n^* = \mathbf{p}(\mathbf{x})^T \mathbf{w}(\mathbf{x}) = \underbrace{\mathbf{p}(\mathbf{x})^T \mathbf{A}(\mathbf{x})^{-1} \mathbf{B}_n(\mathbf{x})}_{\phi_n(\mathbf{x})} \mathbf{y} = \sum_{n=1}^{N} \phi_n(\mathbf{x}) y_n$$

y_n^* : reduced order MLS approximation

Moving Least Squares: Weighting Function Effect



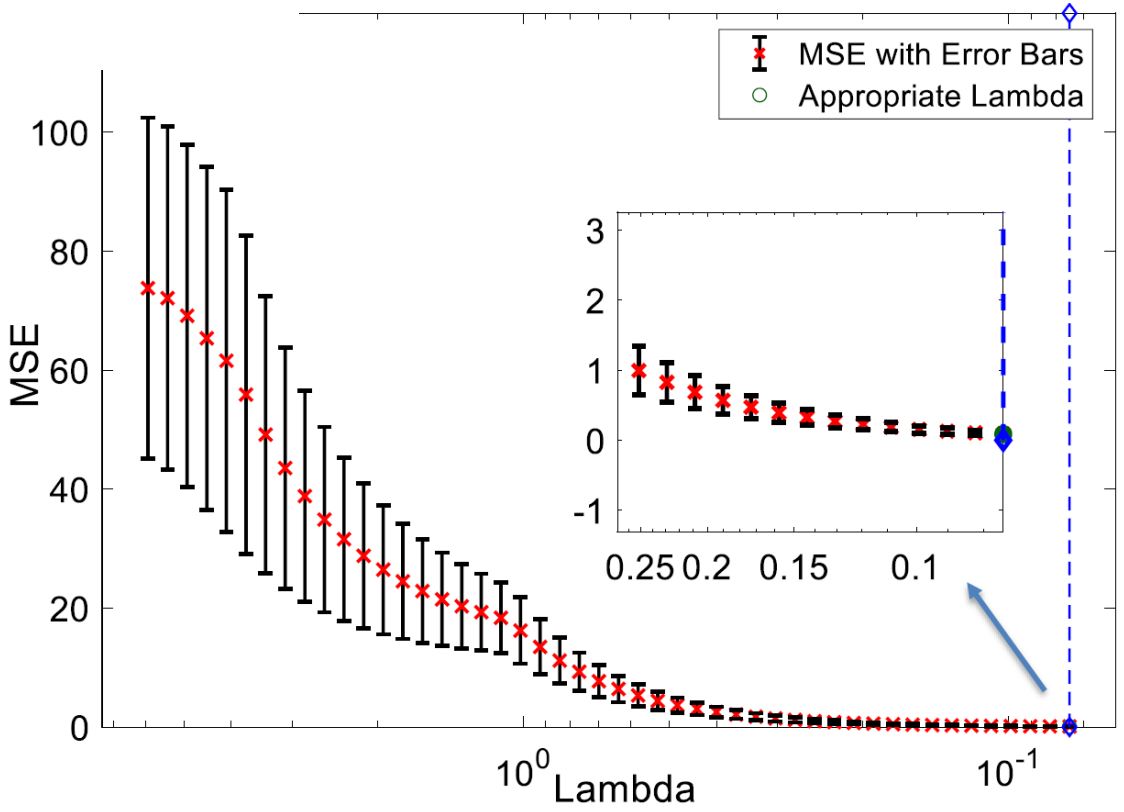
K-fold Cross Validation (1)

- Rotation estimation or out-of-sample testing
  - Assess how the results of a statistical analysis will generalize to an independent data set
- Can remove bias in choosing training and test set and improve model confidence
  - Step 1: Divide the original data set into equal K folds (parts)
  - Step 2: Use one part as the test set and the rest as the training sets
  - Step 3: Train model and calculate mean square error (MSE) on test set
  - Step 4: Repeat steps 2 and 3 K times each time using a different section as the test set
  - Step 5: The average accuracy is taken as the final model accuracy

	Data				
	Fold 1	Fold 2	Fold 3	...	Fold 10
Set 1	**Test**	Train	Train	Train	Train
Set 2	Train	**Test**	Train	Train	Train
Set 3	Train	Train	**Test**	Train	Train
...	Train	Train	Train	**Test**	Train
Set 10	Train	Train	Train	Train	**Test**

K-fold Cross Validation (2)

- For each regularization parameter λ, K-fold cross-validation can be used to find MSE of data.
- By comparing result of K-fold cross-validation, appropriate λ can be found.



Matlab Code: Fig.3.30 (1)

```
%% L1 and L2 norm regression example
%% Generation of data
clc
clear
x0 = -2:0.05:2; % 81 linearly spaced x coordinates are spaced between interval [-2,2]
n = length(x0); % The total number of data points (81)
x1 = x0+randn(1,n)*0.05; % x+epsilon1
x2 = x0+randn(1,n)*0.05; % x+epsilon2
x3 = x0+randn(1,n)*0.05; % x+epsilon3
x = [x1.^5;x0.^4;x0.^3;x2.^2;x3;ones(1,n)]';
weights = [1;0;0;-4;-5;0]; % Weights
y = x*weights; % Simulated data y = x^5 - 4*x^2 - 5*x


%% Regressions
[b_lasso,fitinfo] = lasso(x, y,'CV',10); % L1 norm regularized regression
lam = fitinfo.Index1SE; % Index of appropriate Lambda
b_lasso_opt = b_lasso(:,lam) % Weights for L1 norm regularized regression
lambda = 1; % Set lambda equals to 1 for L2 norm regularized regression (You can also find an appropriate Lambda yourself)
b_ridge = (x'*x+lambda*eye(size(x,2)))^-1*x'*y % Weights for L2 norm regularized regression (Has analytical solution)


b_ols = polyfit (x1',y,5) % Weights for non regularized regression (Ordinary Least Squares)
```

Matlab Code: Fig.3.30 (2)

```
xplot = [x0.^5;x0.^4;x0.^3;x0.^2;x0;ones(1,n)]';
y_lasso = xplot*b_lasso_opt ; % L1 norm regression result
y_ols = xplot*b_ols'; % Non regularized regression result
y_ridge = xplot*b_ridge ; % L2 norm regression result


%% Plots
plot(x0,y,'bo') % Plot of origin data
hold on
plot(x0,y_lasso,'LineWidth',1) % Plot of L1 norm regression
hold on
plot(x0,y_ridge,'LineWidth',1) % Plot of L2 norm regression
hold on
plot(x0,y_ols,'LineWidth',1) % Plot of non regularized regression
ylabel('Y','fontsize',20)
xlabel('X','fontsize',20)
legendset = legend('Original data','L1 norm regression','L2 norm regression','No regularization','location','southeast')
set(gca,'FontSize',20);
lassoPlot(b_lasso,fitinfo,'PlotType','CV'); % Cross validated MSE
legend('show') % Show legend
```

Homework #2: Regression

- (1) Perform Multivariate Linear Regression on baseball data
  - (OBP, SLG) vs. RS
  - Compare the results from Python and Matlab

- (2) Perform non-linear regression without any regularization, and with L1 and L2 regularization.

$$y = \frac{1}{4}e^{x} - \frac{1}{8}\sin x - \cos x + 0.1x^{3}$$

  - Generate your training data with Gaussian noise and compare the performance with true data.
  - Show how changing the penalty parameter can affect your prediction.