

# Unit 3

# Multiple Linear Regression

---

EE-UY 4563/EL-GY 6143: INTRODUCTION TO MACHINE LEARNING

PROF. SUNDEEP RANGAN

(WITH MODIFICATION BY YAO WANG)

# Learning Objectives

---

- ❑ Formulate a machine learning model as a multiple linear regression model.
  - Identify prediction vector and target for the problem.
- ❑ Write the regression model in matrix form. Write the feature matrix
- ❑ Compute the least-squares solution for the regression coefficients on training data.
- ❑ Derive the least-squares formula from minimization of the RSS
  
- ❑ Manipulate 2D arrays in python (indexing, stacking, computing shapes, ...)
- ❑ Compute the LS solution using python linear algebra and machine learning packages

# Pre-Requisites for this Lecture

---

## ❑ Undergraduate students:

- We will cover Unit 2 (Simple Linear Regression) in class first
- Some of the material in this lecture is a duplicate of Lecture 2
- I will go through this lecture more slowly, esp. for the linear algebra

## ❑ Graduate students:


- We can skip Lecture 2 and start this lecture directly after Lecture 1.
- Do lecture 2 at home.
- Will not review basic linear algebra in class. You should review this on your own.

## ❑ Pre-requisites for this lecture (graduate or undergraduate): You should know how to:

- Install python and run python notebooks
- Describe simple linear models mathematically
- Derive equations for fitting simple linear models
- Perform basic manipulations and plotting of data in python

# Outline

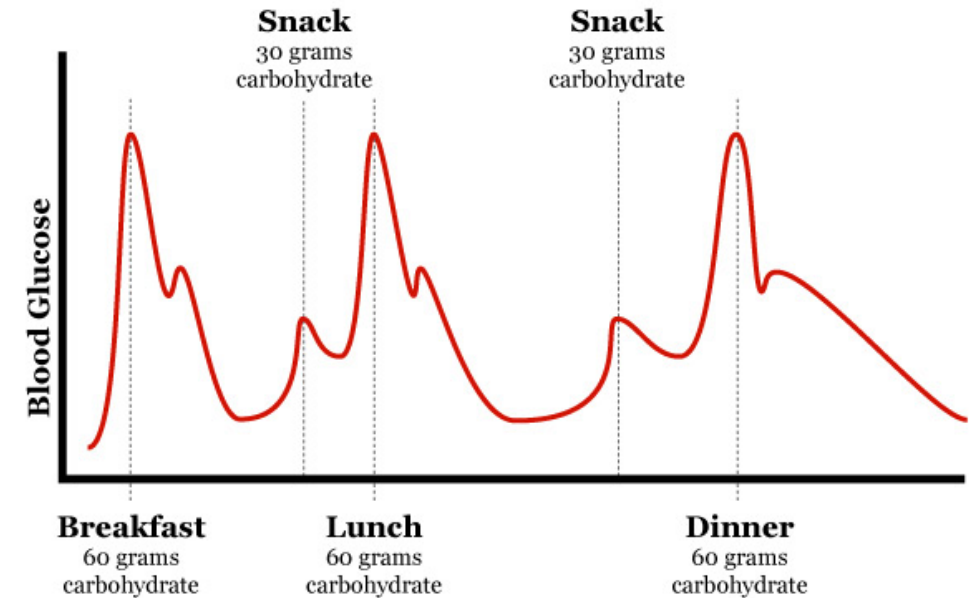
---

 Motivating Example: Understanding glucose levels in diabetes patients

- ☐ Multiple variable linear models
- ☐ Least squares solutions
- ☐ Computing the solutions in python
- ☐ Special case: Simple linear regression
- ☐ Extensions

# Example: Blood Glucose Level

- ❑ Diabetes patients must monitor glucose level
- ❑ What causes blood glucose levels to rise and fall?
- ❑ Many factors
- ❑ We know mechanisms **qualitatively**
- ❑ But, **quantitative** models are difficult to obtain
  - Hard to derive from first principles
  - Difficult to model physiological process precisely
- ❑ Can machine learning help?



# Data from AIM 94 Experiment



## Diabetes Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: This diabetes dataset is from AIM '94

Data Set Characteristics:	Multivariate, Time-Series	Number of Instances:	N/A	Area:	Life
Attribute Characteristics:	Categorical, Integer	Number of Attributes:	20	Date Donated	N/A
Associated Tasks:	N/A	Missing Values?	N/A	Number of Web Hits:	161379

### ☐ Data collected as series of events

- Eating
- Exercise
- Insulin dosage

### ☐ Target variable glucose level monitored

## Data Set Information:

Diabetes patient records were obtained from two sources: an electronic clock to timestamp events, whereas the paper records only provide times assigned to breakfast (08:00), lunch (12:00), dinner (18:00), and bedtime. The electronic records have more realistic time stamps.

Diabetes files consist of four fields per record. Each field is separated by a space.

File Names and format:

- (1) Date in MM-DD-YYYY format
- (2) Time in XX:YY format
- (3) Code
- (4) Value

The Code field is deciphered as follows:

- 33 = Regular insulin dose
- 34 = NPH insulin dose
- 35 = UltraLente insulin dose
- 48 = Unspecified blood glucose measurement
- 57 = Unspecified blood glucose measurement
- 58 = Pre-breakfast blood glucose measurement
- 59 = Post-breakfast blood glucose measurement
- 60 = Pre-lunch blood glucose measurement
- 61 = Post-lunch blood glucose measurement
- 62 = Pre-supper blood glucose measurement
- 63 = Post-supper blood glucose measurement

# Demo on GitHub

---

☐ All code is available in github:

[https://github.com/sdrangan/introml/blob/master/unit03\\_mult\\_lin\\_reg/demo2\\_glucose.ipynb](https://github.com/sdrangan/introml/blob/master/unit03_mult_lin_reg/demo2_glucose.ipynb)

## Demo: Predicting Glucose Levels using Multiple Linear Regression

In this demo, you will learn how to:

- Fit multiple linear regression models using python's sklearn package.
- Split data into training and test.
- Manipulating and visualizing multivariable arrays.

We first load the packages as usual.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

## Diabetes Data Example

To illustrate the concepts, we load the well-known diabetes data set. This dataset is included in the sklearn.datasets module and can be loaded as follows.

```
from sklearn import datasets, linear_model, preprocessing
```

# Using Google Colaboratory

❑ Two options for running the demos:

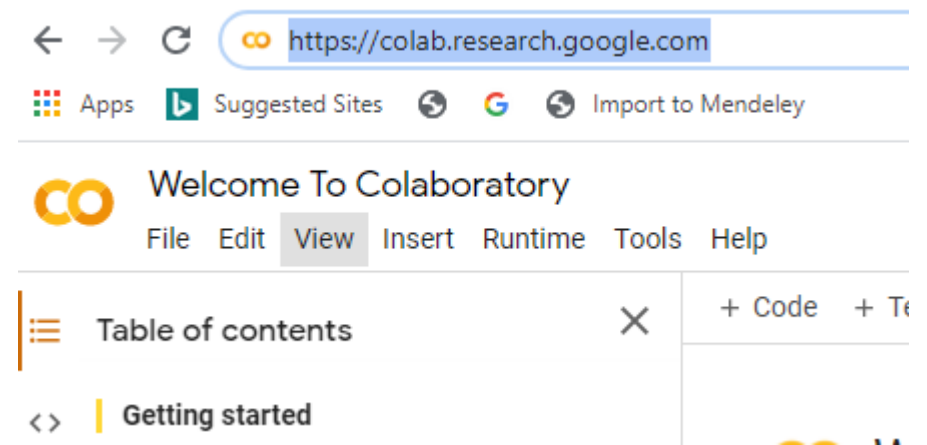
❑ Option 1:

- Clone the github repository to your local machine and run it using jupyter notebook
- Need to install all the software correctly

❑ Option 2: Run on the cloud in Google Colaboratory

❑ For Option 2:

- Go to <https://colab.research.google.com/>
- File->Open Notebook
- Select GitHub tab
- Enter github URL:  
<https://github.com/sdrangan/introml>
- Select the unit03\_mult\_lin\_reg/demo1\_glucose.ipynb





# Demo on Google Colab

The screenshot shows the Google Colab interface with the 'GitHub' tab selected. The search bar contains the URL 'https://github.com/sdrangan/introml'. Below the search bar, the 'Repository' is set to 'sdrangan/introml' and the 'Branch' is set to 'master'. The search results show two files: 'unit02\_simp\_lin\_reg/lab\_housing\_partial.ipynb' and 'unit03\_mult\_lin\_reg/demo1\_glucose.ipynb', with the latter being highlighted.

Examples	Recent	Google Drive	GitHub	Upload
Enter a GitHub URL or search by organization or user <input type="checkbox"/> Include private repos				
<input type="text" value="https://github.com/sdrangan/introml"/>				
Repository: <input type="text" value="sdrangan/introml"/> Branch: <input type="text" value="master"/>				
Path				
<input type="text" value="unit02_simp_lin_reg/lab_housing_partial.ipynb"/>				
<input type="text" value="unit03_mult_lin_reg/demo1_glucose.ipynb"/>				

# Loading the Data

```
from sklearn import datasets, linear_model, preprocessing

# Load the diabetes dataset
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target
```

## ❑ Sklearn package:

- Many methods for machine learning
- Datasets
- Will use throughout this class

## ❑ Diabetes dataset is one example

```
print(diabetes.DESCR)
```

```
.. _diabetes_dataset:
```

Diabetes dataset  
-----

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

**\*\*Data Set Characteristics:\*\***

:Number of Instances: 442

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression one year after baseline

:Attribute Information:

- Age
- Sex
- Body mass index
- Average blood pressure
- S1
- S2
- S3
- S4
- S5
- S6

# Finding a Mathematical Model

## Attributes

$x_1$ : Age  
 $x_2$ : Sex  
 $x_3$ : BMI  
 $x_4$ : BP  
 $x_5$ : S1  
 $\vdots$   
 $x_{10}$ : S6



## Target

$y$  = Glucose level

$$y \approx \hat{y} = f(x_1, \dots, x_{10})$$

❑ **Goal:** Find a function to predict glucose level from the 10 attributes

❑ **Problem:** Several attributes

- Need a **multi-variable function**

# Matrix Representation of Data

□ Data is a **matrix** and a target **vector**

□  $n$  samples:

- One sample per row

□  $k$  features / attributes / predictors:

- One feature per column

Attributes

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{bmatrix}$$

Target vector

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Samples

□ This example:

- $y_i$  = blood glucose measurement of  $i$ -th sample
- $x_{i,j}$ :  $j$ -th feature of  $i$ -th sample
- $\mathbf{x}_i^T = [x_{i,1}, x_{i,2}, \dots, x_{i,k}]$ : feature or predictor vector
- $i$ -th sample contains  $\mathbf{x}_i, y_i$

# In class exercise

---

In class exercise: What are the (normalized) ages of the first 5 subjects?

```
[18] # TODO
```

In-class exercise: Print the attributes S1-S3 for subjects 10-15

```
[ ] # TODO
```

Double-click (or enter) to edit

In class exercise: Create a scatter plot of the target variable,  $y$  vs. the BMI. Does there seem to be a relation? What about  $y$  vs. the age? Which is a better predictor?

```
[19] # TODO
```

# Outline

---

- ❑ Motivating Example: Understanding glucose levels in diabetes patients

-  Multiple variable linear models

- ❑ Least squares solutions

- ❑ Computing the solutions in python

- ❑ Special case: Simple linear regression

- ❑ Extensions

# Multivariable Linear Model for Glucose

## Attributes

Age, Sex, BMI, BP, S1, ..., S6  
 $\mathbf{x} = [x_1, \dots, x_{10}]$

$$y \approx \hat{y} = f(x_1, \dots, x_{10})$$



## Target

$y$  = Glucose level

□ **Goal:** Find a function to predict glucose level from the 10 attributes

□ **Linear Model:** Assume glucose is a **linear function** of the predictors:

$$[\text{glucose}] \approx [\text{prediction}] = \beta_0 + \beta_1[\text{Age}] + \dots + \beta_4[\text{BP}] + \beta_5[\text{S1}] + \dots + \beta_{10}[\text{S6}]$$

□ **General form:**

$$\begin{array}{ccccccc} y & \approx & \hat{y} & = & \beta_0 + & \underbrace{\beta_1 x_1 + \dots + \beta_4 x_4 + \beta_5 x_5 + \dots + \beta_{10} x_{10}}_{10 \text{ Features}} \\ \text{Target} & & & \uparrow & & & \\ & & & \text{Intercept} & & & \end{array}$$



# Multiple Variable Linear Model

- ❑ Vector of **features**:  $\mathbf{x} = [x_1, \dots, x_k]$ 
  - $k$  features (also known as predictors, independent variable, attributes, covariates, ...)
- ❑ Single **target variable**  $y$ 
  - What we want to predict
- ❑ Linear model: Make a **prediction**  $\hat{y}$

$$y \approx \hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$$

- ❑ **Data** for training
  - Samples are  $(\mathbf{x}_i, y_i)$ ,  $i=1,2,\dots,n$ .
  - Each sample has a vector of features:  $\mathbf{x}_i = [x_{i1}, \dots, x_{ik}]$  and scalar target  $y_i$
- ❑ **Problem**: Learn the best **coefficients**  $\boldsymbol{\beta} = [\beta_0, \beta_1, \dots, \beta_k]$  from the training data



# Example: Heart Rate Increase

□ **Linear Model:**  $[\text{HR increase}] \approx \beta_0 + \beta_1[\text{mins exercise}] + \beta_2[\text{exercise intensity}]$

□ **Data:**

Subject number	HR before	HR after	Mins on treadmill	Speed (min/km)	Days exercise / week
123	60	90	1	5.2	3
456	80	110	2	4.1	1
789	70	130	5	3.5	2
⋮	⋮	⋮	⋮	⋮	⋮



Measuring fitness of athletes

<https://www.mercurynews.com/2017/10/29/4851089/>

# Why Use a Linear Model?

---

❑ Many natural phenomena have linear relationship

❑ Predictor has small variation

- Suppose  $y = f(x)$
- If variation of  $x$  is small around some value  $x_0$ , then

$$y \approx f(x_0) + f'(x_0)(x - x_0) = \beta_0 + \beta_1 x,$$

$$\beta_0 = f(x_0) - f'(x_0)x_0, \quad \beta_1 = f'(x_0)$$

❑ Simple to compute

❑ Easy to interpret relation

- Coefficient  $\beta_j$  indicates the importance of feature  $j$  for the target.

❑ Advanced: Gaussian random variables:

- If two variables are jointly Gaussian, the optimal predictor of one from the other is linear predictor

# Matrix Review

---

□ Consider

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 0 \\ 3 & 2 \end{bmatrix}, \quad x = \begin{bmatrix} 2 \\ 3 \end{bmatrix},$$

□ Compute (computations on the board):

- Matrix vector multiply:  $Ax$
- Transpose:  $A^T$
- Matrix multiply:  $AB$
- Solution to linear equations: Solve for  $u$ :  $x = Bu$
- Matrix inverse:  $B^{-1}$



# Slopes, Intercept and Inner Products

❑ Model with **coefficients**  $\beta$ :  $\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$

❑ Sometimes use **weight bias version**:

$$\hat{y} = b + w_1 x_1 + \cdots + w_k x_k$$

- $b = \beta_0$  : **Bias** or **intercept**
- $\mathbf{w} = \boldsymbol{\beta}_{1:k} = [\beta_1, \dots, \beta_k]$ : **Weights** or **slope vector**

❑ Can write either with **inner product**:

$$\hat{y} = \beta_0 + \boldsymbol{\beta}_{1:k} \cdot \mathbf{x}$$

or

$$\hat{y} = b + \mathbf{w} \cdot \mathbf{x}$$

❑ Inner product:

- $\mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^k w_j x_j$
- Will use alternate notation:  $\mathbf{w}^T \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle$

# Matrix Form of Linear Regression

□ Data:  $(x_i, y_i), i = 1, \dots, n$

□ Predicted value for  $i$ -th sample:  $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik}$

□ Matrix form

$$\hat{\mathbf{y}} \text{ a } n \text{ predicted values } \left\{ \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} \right\} = \underbrace{\begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix}}_{\mathbf{A} \text{ a } n \times p \text{ feature matrix}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}}_{\boldsymbol{\beta} \text{ with } p = k + 1 \text{ coefficient vector}}$$

□ Matrix equation:

$$\hat{\mathbf{y}} = \mathbf{A} \boldsymbol{\beta}$$

# In-Class Exercise

Consider a linear model:

$$[\text{HR increase}] \approx \beta_0 + \beta_1[\text{mins exercise}] + \beta_2[\text{exercise intensity}].$$

We are given the following data: Only the first three rows and the final entry are shown.


<u>Subject number</u>	<u>HR before</u>	<u>HR after</u>	<u>Mins on treadmill</u>	<u>Speed (min/km)</u>	<u>Days exercise / week</u>
123	60	90	1	5.2	3
456	80	110	2	4.1	1
789	70	130	5	3.5	2
⋮	⋮	⋮	⋮	⋮	⋮
283	75	100	1	4.8	0

} 100 subjects

- Q1: What is the feature matrix  $A$  and target vector  $y$ . What are their dimensions?
  - Fill in only the values from the first three rows and the last row
- Q2. Suppose that after training, we find parameters  $\beta = [0, 15, 3]$ . If the initial HR is 70 bpm, what is the predicted HR after 2 minutes of exercise at 5 km/hr.

# Outline

---

- ❑ Motivating Example: Understanding glucose levels in diabetes patients
- ❑ Multiple variable linear models
- ❑ Least squares solutions
- ❑ Computing the solutions in python
- ❑ Special case: Simple linear regression
- ❑ Extensions

# Least Squares Model Fitting

---

□ How do we select parameters  $\boldsymbol{\beta} = (\beta_0, \dots, \beta_k)$ ?

□ Define  $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik}$

- Predicted value on sample  $i$  for parameters  $\boldsymbol{\beta} = (\beta_0, \dots, \beta_k)$

□ Define average residual sum of squares:

$$\text{RSS}(\boldsymbol{\beta}) := \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- Note that  $\hat{y}_i$  is implicitly a function of  $\boldsymbol{\beta} = (\beta_0, \dots, \beta_k)$
- Also called the sum of squared residuals (SSR) and sum of squared errors (SSE)

□ Least squares solution: Find  $\boldsymbol{\beta}$  to minimize RSS.



# Variants of RSS

---

❑ Often use some variant of RSS

- Note: these are not standard

❑ Residual sum of squares:  $RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$

❑ RSS per sample or Mean Squared Error:

$$MSE = \frac{RSS}{n} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

❑ Normalized RSS or Normalized MSE:

$$\frac{RSS/n}{s_y^2} = \frac{MSE}{s_y^2} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

# Finding Parameters via Optimization





## A general ML recipe

---

### General ML problem

- Pick a **model** with **parameters**
- Get **data**
- Pick a **loss function**
  - Measures goodness of fit model to data
  - Function of the parameters

### Multiple linear regression

- Pick a **model** with **parameters**  Linear model:  $\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$
- Get **data**  Data:  $(x_i, y_i), i = 1, 2, \dots, n$
- Pick a **loss function**  Loss function:  
$$RSS(\beta_0, \dots, \beta_k) := \sum (y_i - \hat{y}_i)^2$$
- Find parameters that **minimizes** loss  Select  $\boldsymbol{\beta} = (\beta_0, \dots, \beta_k)$  to minimize  $RSS(\boldsymbol{\beta})$

# RSS as a Vector Norm

□ RSS is given by sum:

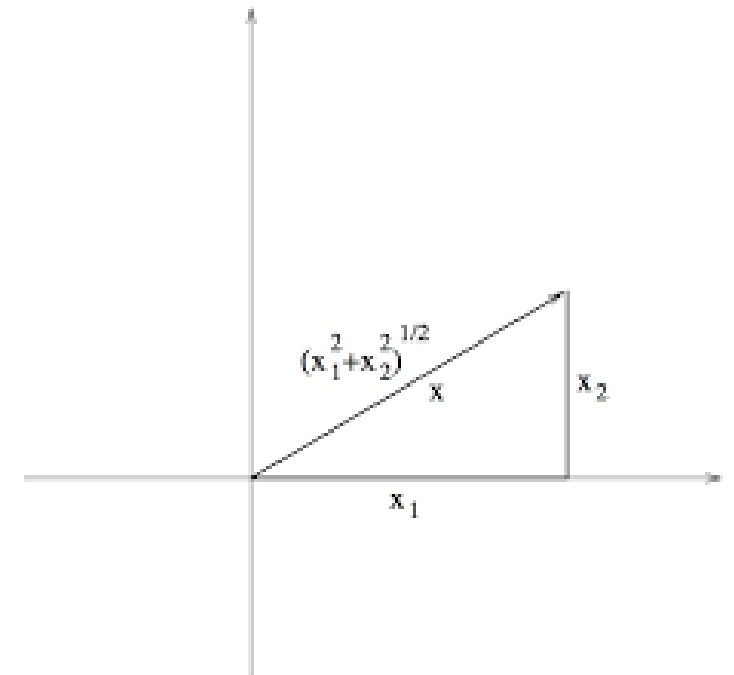
$$\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

□ Define **norm** of a vector:

- $\|\mathbf{x}\| = (x_1^2 + \dots + x_r^2)^{1/2}$
- Standard Euclidean norm.
- Sometimes called  $\ell$ -2 norm.  $\ell$  is for Lebesgue

□ Write RSS in vector form:

$$\text{RSS} = \|\mathbf{y} - \hat{\mathbf{y}}\|^2$$



# Least Squares Solution

□ Consider cost function of the RSS:

$$\text{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad \hat{y}_i = \sum_{j=0}^p A_{ij} \beta_j$$

- Vector  $\boldsymbol{\beta}$  that minimizes RSS called the **least-squares** solution

□ **Least squares solution**: The vector  $\boldsymbol{\beta}$  that minimizes the RSS is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$$

- Can compute the best coefficient vector analytically
- Just solve a linear set of equations
- Will show the proof below

# Proving the LS Formula

---

□ **Least squares formula:** The vector  $\boldsymbol{\beta}$  that minimizes the RSS is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$$

□ To prove this formula, we will:

- Review gradients of multi-variable functions
- Compute gradient  $\nabla \text{RSS}(\boldsymbol{\beta})$
- Solve  $\nabla \text{RSS}(\boldsymbol{\beta}) = 0$

# Gradients of Multi-Variable Functions

Consider scalar valued function of a vector:  $f(\boldsymbol{\beta}) = f(\beta_1, \dots, \beta_n)$

**Gradient** is the column vector:

$$\nabla f(\boldsymbol{\beta}) = \begin{bmatrix} \partial f(\boldsymbol{\beta}) / \partial \beta_1 \\ \vdots \\ \partial f(\boldsymbol{\beta}) / \partial \beta_n \end{bmatrix}$$

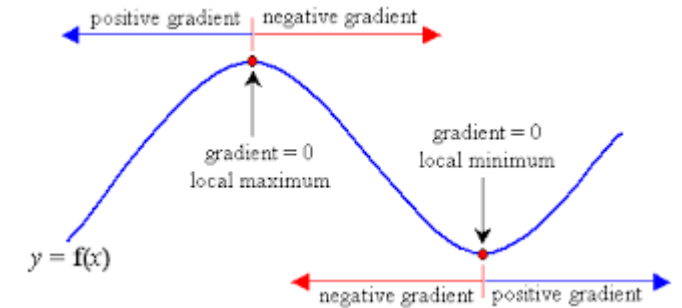
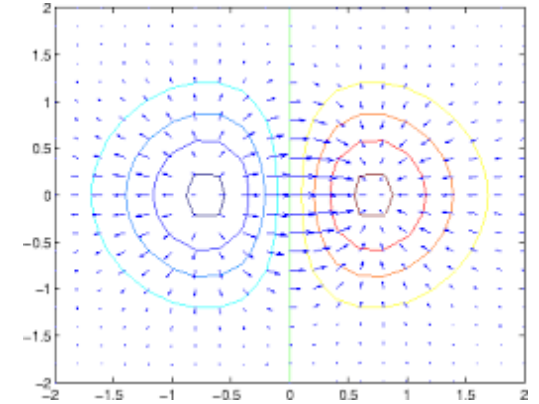
Represents direction of maximum increase

At a local minima or maxima:  $\nabla f(\boldsymbol{\beta}) = 0$

- Solve  $n$  equations and  $n$  unknowns

Ex:  $f(\beta_1, \beta_2) = \beta_1 \sin \beta_2 + \beta_1^2 \beta_2$ .

- Compute  $\nabla f(\boldsymbol{\beta})$ . Solution on board



# Proof of the LS Formula

---

- Consider cost function of the RSS:

$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad \hat{y}_i = \sum_{j=0}^p A_{ij}\beta_j$$

- Vector  $\boldsymbol{\beta}$  that minimizes RSS called the **least-squares** solution

- Compute partial derivatives via chain rule:  $\frac{\partial RSS}{\partial \beta_j} = -2 \sum_{i=1}^n (y_i - \hat{y}_i) A_{ij}, j = 1, 2, \dots, k$

- Matrix form:  $RSS = \|A\boldsymbol{\beta} - \mathbf{y}\|^2, \nabla RSS = -2A^T(\mathbf{y} - A\boldsymbol{\beta})$

- Solution:  $A^T(\mathbf{y} - A\boldsymbol{\beta}) = 0 \rightarrow \boldsymbol{\beta} = (A^T A)^{-1} A^T \mathbf{y}$  (least squares solution of equation  $A\boldsymbol{\beta} = \mathbf{y}$ )

- Minimum RSS:  $RSS = \mathbf{y}^T [I - A(A^T A)^{-1} A^T] \mathbf{y}$

- Proof on the board

# LS Solution via Auto-Correlation Functions

- Each data sample has a linear feature vector:

$$A_i = (A_{i0}, \dots, A_{ik}) = (1, x_{i1}, \dots, x_{ik})$$

- Define sample **auto-correlation** matrix and **cross-correlation** vector:

- $R_{AA} = \frac{1}{n} A^T A$ ,  $R_{AA}(\ell, m) = \frac{1}{n} \sum_{i=1}^n A_{i\ell} A_{im}$  (correlation of feature  $\ell$  and feature  $m$ )
- $R_{Ay} = \frac{1}{n} A^T y$ ,  $R_{yA}(\ell) = \frac{1}{n} \sum_{i=1}^n A_{i\ell} y_i$  (correlation of feature  $\ell$  and target)

- Least squares solution is:  $\beta = R_{AA}^{-1} R_{Ay}$



# Mean Removed Form of the LS Solution

- Often useful to remove mean from data before fitting
- Sample mean:  $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$ ,  $\bar{x}_j = \frac{1}{N} \sum_{i=1}^N x_{ij}$ ,  $\bar{\mathbf{x}} = [\bar{x}_1, \dots, \bar{x}_k]$
- Defined **mean removed data**:  $\tilde{X}_{ij} = x_{ij} - \bar{x}_j$ ,  $\tilde{y}_i = y_i - \bar{y}$
- Sample **covariance** matrix and **cross-covariance** vector:
  - $S_{xx}(\ell, m) = \frac{1}{N} \sum_{i=1}^N (x_{i\ell} - \bar{x}_\ell)(x_{im} - \bar{x}_m)$ ,  $S_{xx} = \frac{1}{N} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}}$
  - $S_{xy}(\ell) = \frac{1}{N} \sum_{i=1}^N (x_{i\ell} - \bar{x}_\ell)(y_i - \bar{y})$ ,  $S_{xy} = \frac{1}{N} \tilde{\mathbf{X}}^T \tilde{\mathbf{y}}$
- **Mean-Removed form** of the least squares solution:

$$\hat{y} = \boldsymbol{\beta}_{1:k} \cdot \mathbf{x} + \beta_0, \quad \boldsymbol{\beta}_{1:k} = S_{xx}^{-1} S_{xy}, \quad \beta_0 = \bar{y} - \boldsymbol{\beta}_{1:k} \cdot \bar{\mathbf{x}}$$

- Proof: On board

# $R^2$ : Goodness of Fit

❑ Multiple variable **coefficient of determination**:

$$R^2 = \frac{s_y^2 - MSE}{s_y^2} = 1 - \frac{MSE}{s_y^2}$$

- $MSE = \frac{RSS}{n} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
- Sample variance is:  $s_y^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$ ,  $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ ,

❑ **Interpretation**:

- $\frac{MSE}{s_y^2} = \frac{\text{Error with linear predictor}}{\text{Error predicting by mean}}$
- $R^2$  = fraction of variance reduced or “explained” by the model.

❑ On the training data (not necessarily on the test data):

- $R^2 \in [0,1]$  always
- $R^2 \approx 1 \Rightarrow$  linear model provides a good fit
- $R^2 \approx 0 \Rightarrow$  linear model provides a poor fit

# In-Class Exercise

---


We are given the following data

Sample number	Target $y_i$	Feature 1 $x_{i1}$	Feature 2 $x_{i2}$
1	3.0	0	1
2	5.0	2	3
3	9.0	4	8
4	10.0	6	10

- Q1. Write the equations to solve for the linear model using all four data points
  - Write the feature matrix and the equations for coefficients.
  - Do not solve them (you would need a computer)
- Q2. Can you find parameters that exactly fits the first three data points?
  - Just state if such parameters exist. You do not need to find them.

# Outline

---

- ❑ Motivating Example: Understanding glucose levels in diabetes patients
- ❑ Multiple variable linear models
- ❑ Least squares solutions
- ❑ Computing the solutions in python
- ❑ Special case: Simple linear regression
- ❑ Extensions

# Arrays and Vector in Python and MATLAB

❑ There are some key differences between MATLAB and Python that you need to get used to

## ❑ MATLAB

- All arrays are at least 2 dimensions
- Vectors are  $1 \times N$  (row vectors) or  $N \times 1$  (column) vectors
- Matrix vector multiplication syntax depends if vector is on left or right:  $x' * A$  or  $A * x$

## ❑ Python:

- Arrays can have 1, 2, 3, ... dimension
- Vectors can be 1D arrays; matrices are generally 2D arrays
- Vectors that are 1D arrays are neither row nor column vectors
- If  $x$  is 1D and  $A$  is 2D, then left and right multiplication are the same:  $x \cdot \text{dot}(A)$  and  $A \cdot \text{dot}(x)$

❑ **Lecture notes:** We will generally treat  $x$  and  $x^T$  the same.

- Can write  $x = (x_1, \dots, x_N)$  and still multiply by a matrix on left or right

# Fitting Using sklearn

```
: ns_train = 300
  ns_test = nsamp - ns_train
  X_tr = X[:ns_train,:]
  y_tr = y[:ns_train]
```

- ❑ Return to diabetes data example
- ❑ All code in demo
- ❑ Divide data into two portions:
  - Training data: First 300 samples
  - Test data: Remaining 142 samples
- ❑ Train model on training data.
- ❑ Test model (i.e. measure RSS) on test data
- ❑ Reason for splitting data discussed next lecture.

# Manually Computing the Solution

```
ones = np.ones((ns_train,1))  
A = np.hstack((ones,X_tr))
```

```
out = np.linalg.lstsq(A,y_tr)  
beta = out[0]
```

❑ Use numpy linear algebra routine to solve

$$\beta = (A^T A)^{-1} A^T y$$

❑ Common mistake:

- Compute matrix inverse  $P = (A^T A)^{-1}$ ,
- Then compute  $\beta = P A^T y$
- Full matrix inverse is VERY slow. Not needed.
- Can directly solve linear system:  $A \beta = y$
- Numpy has routines to solve this directly

# Calling the sklearn Linear Regression method

```
regr = linear_model.LinearRegression()
regr.fit(X_tr,y_tr)
```

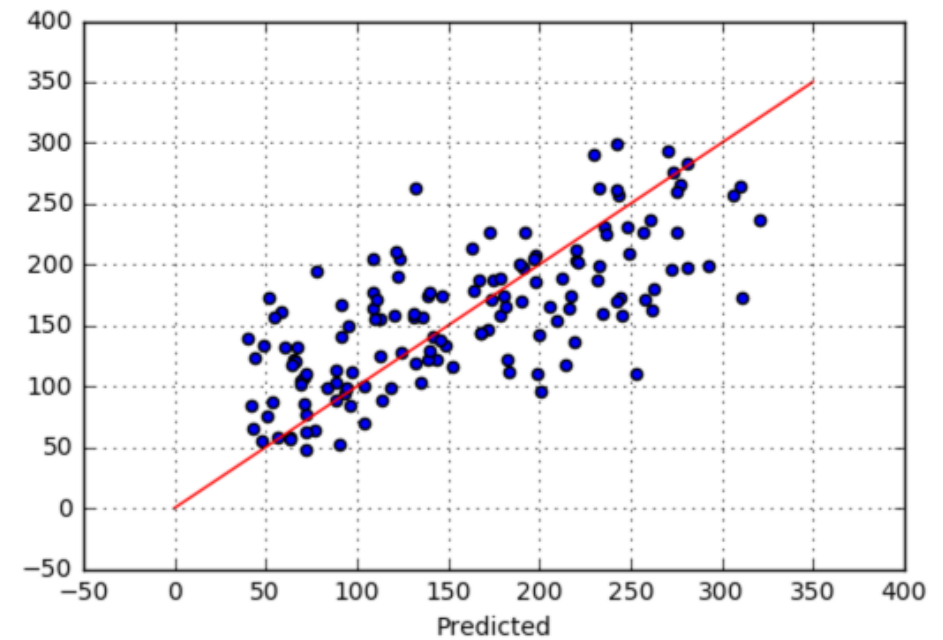
```
X_test = X[ns_train:,:]
y_test = y[ns_train:]
y_test_pred = regr.predict(X_test)
RSS_test = np.mean((y_test_pred-y_test)**2)/(np.std(y_test)**2)
Rsquared_test = 1-RSS_test
print("RSS per sample = {0:f}".format(RSS_test))
print("R^2 = {0:f}".format(Rsquared_test))
```

```
RSS per sample = 0.492801
R^2 = 0.507199
```

We see that the model predicts new samples almost as well as it did the training :

```
plt.scatter(y_test,y_test_pred)
plt.plot([0,350],[0,350], 'r')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.grid()
```

- ❑ Construct a linear regression object
- ❑ Run it on the training data
- ❑ Predict values on the test data





# In-Class Exercise

---

## In-Class Simple Exercise


You are given target values `y` and features `x1` and `x2` below. Fit the model on the first 4 data points and test the model on the fifth data point. You may want to use the following steps

- Construct the training data `X_tr, y_tr`
- Create a regression object `regr = linear_model.LinearRegression()`
- Fit the model with the `regr.fit()` method
- Predict the value on the test value with the `regr.predict()`

```
: ▶ 1 x1 = np.array([0,1,3,5,4])
    2 x2 = np.array([0,0.7, 4.3, 15.1, 13.2])
    3 y = np.array([-2, -0.9, 1.5, 18, 13])
    4
    5 # TODO
```

# Outline

---

- ❑ Motivating Example: Understanding glucose levels in diabetes patients
- ❑ Multiple variable linear models
- ❑ Least squares solutions
- ❑ Computing the solutions in python
-  ❑ Special case: Simple linear regression
- ❑ Extensions

# Simple vs. Multiple Regression

---

## □ Simple linear regression: One predictor (feature)

- Scalar predictor  $x$
- Linear model:  $\hat{y} = \beta_0 + \beta_1 x$
- Can only account for one variable

## □ Multiple linear regression: Multiple predictors (features)

- Vector predictor  $\mathbf{x} = (x_1, \dots, x_k)$
- Linear model:  $\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$
- Can account for multiple predictors
- Turns into simple linear regression when  $k = 1$

# Comparison to Single Variable Models

---

- ❑ We could compute models for each variable separately:

$$\begin{aligned}y &= a_1 + b_1x_1 \\y &= a_2 + b_2x_2 \\&\vdots\end{aligned}$$

- ❑ But, doesn't provide a way to account for joint effects
- ❑ Example: Consider three linear models to predicting longevity:
  - A: Longevity vs. some factor in diet (e.g. amount of fiber consumed)
  - B: Longevity vs. exercise
  - C: Longevity vs. diet AND exercise
  - What does C tell you that A and B do not?



# Special Case: Single Variable

□ Suppose  $k = 1$  predictor.

□ Feature matrix and coefficient vector:

$$A = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

□ LS soln:  $\beta = \left( \frac{1}{N} A^T A \right)^{-1} \left( \frac{1}{N} A^T y \right) = P^{-1} r$

$$P = \begin{bmatrix} 1 & \bar{x} \\ \bar{x} & \frac{\bar{x}^2}{x^2} \end{bmatrix}, \quad r = \begin{bmatrix} \bar{y} \\ \frac{\bar{y}}{xy} \end{bmatrix}$$

□ Obtain single variable solutions for coefficients (after some algebra):

$$\beta_1 = \frac{s_{xy}}{s_x^2}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}, \quad R^2 = \frac{s_{xy}^2}{s_x^2 s_y^2}$$

# Simple Linear Regression for Diabetes Data

```
ym = np.mean(y)
syy = np.mean((y-ym)**2)
Rsq = np.zeros(natt)
for k in range(natt):
    xm = np.mean(X[:,k])
    sxy = np.mean((X[:,k]-xm)*(y-ym))
    sxx = np.mean((X[:,k]-xm)**2)
    Rsq[k] = (sxy)**2/sxx/syy

print("{0:2d} Rsq={1:f}".format(k, Rsq[k]))
```

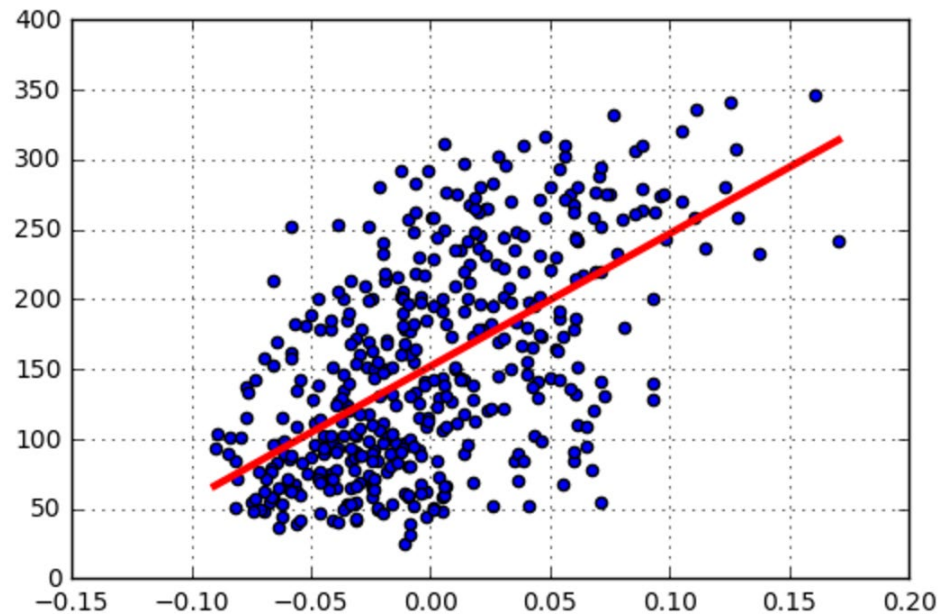
```
0 Rsq=0.035302
1 Rsq=0.001854
2 Rsq=0.343924
3 Rsq=0.194908
4 Rsq=0.044954
5 Rsq=0.030295
6 Rsq=0.155859
7 Rsq=0.185290
8 Rsq=0.320224
9 Rsq=0.146294
```

← Best individual variable

- ❑ Try a fit of each variable individually
- ❑ Compute  $R_k^2$  coefficient for each variable
- ❑ Use formula on previous slide
- ❑ “Best” individual variable is a poor fit
  - $R_k^2 \approx 0.34$

# Scatter Plot

- ❑ No one variable explains glucose well
- ❑ Multiple linear regression is much better




```
# Find the index of the single variable with the best R^2
imax = np.argmax(Rsq)

# Regression line over the range of x values
xmin = np.min(X[:,imax])
xmax = np.max(X[:,imax])
ymin = beta0[imax] + beta1[imax]*xmin
ymax = beta0[imax] + beta1[imax]*xmax
plt.plot([xmin,xmax], [ymin,ymax], 'r-', linewidth=3)

# Scatter plot of points
plt.scatter(X[:,imax],y)
plt.grid()
```

# Outline

---

- ❑ Motivating Example: Understanding glucose levels in diabetes patients
- ❑ Multiple variable linear models
- ❑ Least squares solutions
- ❑ Computing in python
- ❑ Extensions



# Transformed Linear Models

□ Standard linear model:  $\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d$

□ Linear model may be too restrictive

- Relation between  $x$  and  $y$  can be **nonlinear**

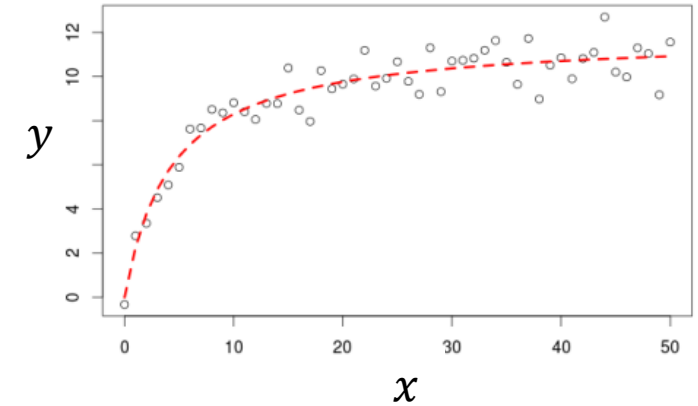
□ Useful to look at models in **transformed form**:

$$\hat{y} = \beta_1 \phi_1(\mathbf{x}) + \dots + \beta_p \phi_p(\mathbf{x})$$

- Each function  $\phi_j(\mathbf{x}) = \phi_j(x_1, \dots, x_d)$  is called a **basis function**
- Each basis function may be nonlinear and a function of multiple variables

□ Can write in vector form:  $\hat{y} = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\beta}$

- $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \dots, \phi_p(\mathbf{x})]$ ,  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_p]$



# Fitting Transformed Linear Models

- Consider transformed linear model

$$\hat{y} = \beta_1 \phi_1(\mathbf{x}) + \cdots + \beta_p \phi_p(\mathbf{x})$$

- We can fit this model exactly as before

- Given data  $(\mathbf{x}_i, y_i), i = 1, \dots, N$
- Want to fit the model from the transformed variables  $\phi_j(\mathbf{x})$  to target  $y$
- Define the transformed matrix:

$$A = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_p(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ \phi_1(\mathbf{x}_N) & \cdots & \phi_p(\mathbf{x}_N) \end{bmatrix}$$

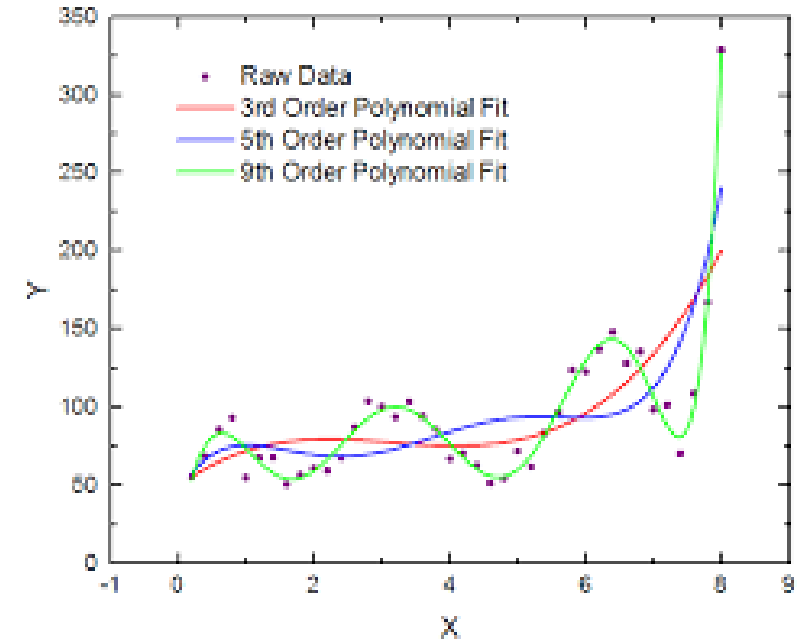
- Predictions:  $\hat{y} = A\beta$
- Least squares fit  $\hat{\beta} = (A^T A)^{-1} A^T y$

# Example: Polynomial Fitting

- ❑ Suppose  $y$  only depends on a single variable  $x$ ,
- ❑ Want to fit a polynomial model
  - $y \approx \beta_0 + \beta_1 x + \cdots \beta_d x^d$
- ❑ Given data  $(x_i, y_i), i = 1, \dots, n$
- ❑ Take basis functions  $\phi_j(x) = x^j, j = 0, \dots, d$
- ❑ Transformed model:  $\hat{y} = \beta_0 \phi_0(x) + \cdots + \beta_d \phi_d(x)$
- ❑ Transformed matrix is:

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^d \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & \cdots & x_n^d \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_d \end{bmatrix}$$

- $p = d + 1$  **transformed features** from 1 original feature
- ❑ Will discuss how to select  $d$  in the next lecture



# Other Nonlinear Examples

---

□ **Multinomial model:**  $\hat{y} = a + b_1x_1 + b_2x_2 + c_1x_1^2 + c_2x_1x_2 + c_3x_2^2$

- Contains all second order terms
- Define parameter vector  $\beta = [a, b_1, b_2, c_1, c_2, c_3]$
- Transformed vector  $\phi(x_1, x_2) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2]$
- Note that the features are nonlinear functions of  $\mathbf{x} = [x_1, x_2]$

□ **Exponential model:**  $\hat{y} = a_1e^{-b_1x} + \dots + a_de^{-b_dx}$

- If the parameters  $b_1, \dots, b_d$  are fixed, then the model is linear in the parameters  $a_1, \dots, a_d$
- Parameter vector  $\beta = [a_1, \dots, a_d]$
- Transformed vector  $\phi(x) = [e^{-b_1x}, \dots, e^{-b_dx}]$
- But, if the parameters  $b_1, \dots, b_d$  are not fixed, the model is nonlinear in  $b_1, \dots, b_d$

# Linear Models via Re-Parametrization

- Sometimes models can be made into a linear model via re-parametrization
- Example: Consider the model:  $\hat{y} = Ax_1(1 + Be^{-x_2})$ 
  - Parameters  $(A, B)$
- This is **nonlinear** in  $(A, B)$  due to the product  $AB$ :  $\hat{y} = Ax_1 + ABx_1e^{-x_2}$
- But, we can define a new set of parameters:
  - $\beta_1 = A$  and  $\beta_2 = AB$
- Then,  $\hat{y} = \beta_1x_1 + \beta_2x_1e^{-x_2}$
- Basis functions:  $\phi(x_1, x_2) = [x_1, x_1e^{-x_2}]$
- After we solve for  $\beta_1, \beta_2$  we can recover  $A, B$  via inverting the equations:

$$A = \beta_1, \quad B = \frac{\beta_2}{A}$$

# Example: Learning Linear Systems

□ Linear system:  $y_k = a_1 y_{k-1} + \dots + a_m y_{k-m} + b_0 x_k + \dots + b_n x_{k-n} + w_k$

□ Transfer function:  $H(z) = \frac{b_0 + \dots + b_n z^{-n}}{1 - a_1 z^{-1} - \dots - a_m z^{-m}}$

□ Given input sequence and output sequence for  $T$  samples,

How do we determine  $\beta = (a_1, \dots, a_m, b_0, \dots, b_n)^T$ .

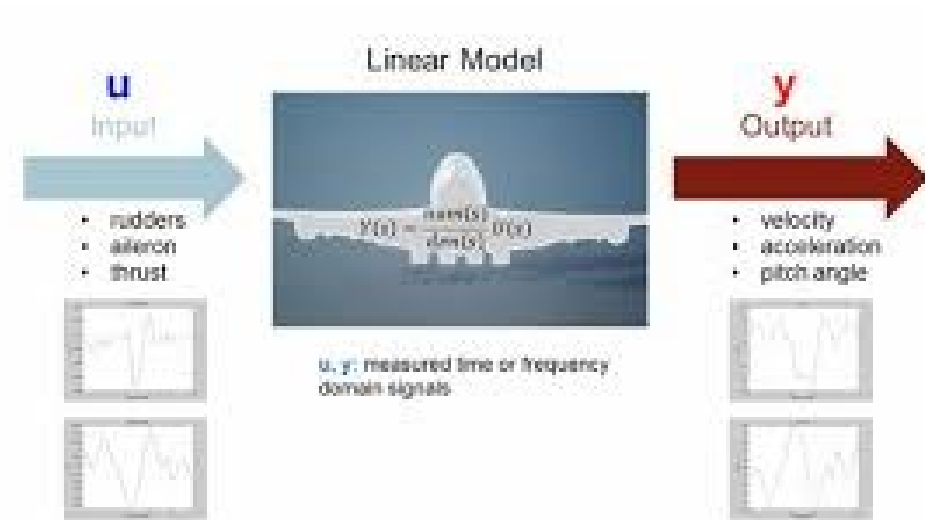
□ Can be solved using linear regression!

□ Write  $y = A\beta + w$  and define  $A, y$

- See homework problem

□ Many applications

- Learning dynamics in robots / mechanical systems
- Modeling responses in neural systems
- Stock market time series
- Speech modeling. Fit a model each 25 ms.



# One Hot Coding

- ❑ Suppose that one feature  $x_j$  is a **categorical** variable
- ❑ Ex: Predict the price of a car,  $y$ , given model  $x_1$  and interior space  $x_2$ 
  - Suppose there are 3 different models of a car (Ford, BMW, GM)
  - Bad idea: Arbitrarily assign an index to each possible car model
  - Can give unreasonable relations
- ❑ One-hot coding example:
  - With 3 possible categories, represent  $x_1$  using 3 binary features  $(\phi_1, \phi_2, \phi_3)$
  - Model:  $y = \beta_0 + \beta_1\phi_1 + \beta_2\phi_2 + \beta_3\phi_3 + \beta_4x_2$
  - Essentially obtain 3 different models:
    - Ford:  $y = \beta_0 + \beta_1 + \beta_4x_2$
    - BMW:  $y = \beta_0 + \beta_2 + \beta_4x_2$
    - GM:  $y = \beta_0 + \beta_3 + \beta_4x_2$
  - Allows different intercepts (or mean values) for different categories!

Model	$\phi_1$	$\phi_2$	$\phi_3$
Ford	1	0	0
BMW	0	1	0
GM	0	0	1

# In-Class Exercise

---

- ❑ Go to github: <https://github.com/sdrangan/introml>
- ❑ Notebook: [unit03\\_mult\\_lin\\_reg/linreg\\_inclass.ipynb](#)
- ❑ You can do this on Google colab or on your local machine

## Multiple Linear Regression In-class Exercise

Before starting to the lab, the following exercise, we will do a very simple example of fitting a model on synthetic data. In doing this exercise, you will learn to:

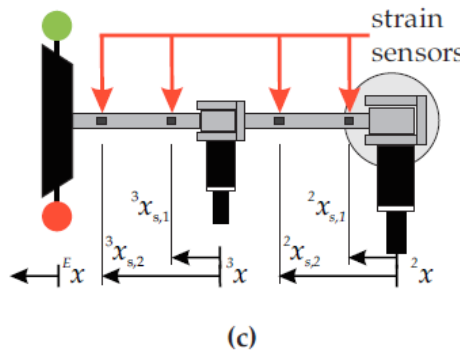
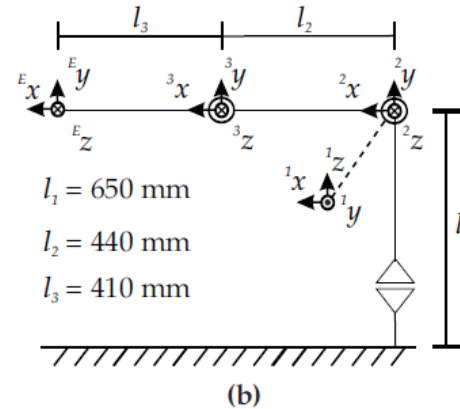
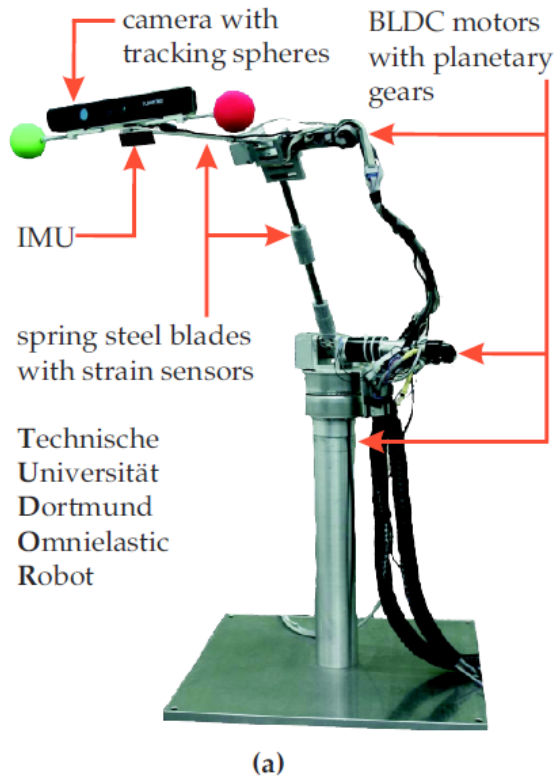
- Evaluate methods using synthetic data
- Describe nonlinear basis functions for a model
- Define the transformations to the basis functions with a `transform` method
- Fit the linear model using the transformed features

We begin by loading the packages we will need.

```
In [1]: 1 import numpy as np
        2 import matplotlib
        3 import matplotlib.pyplot as plt
        4 from sklearn import linear_model
```



# Lab: Robot Calibration



## □ Predict the current draw

- Needed to predict power consumption

## □ Predictors:

- Joint angles, velocity and acceleration
- Strain gauge readings (measure of load)

## □ Full website at TU Dortmund, Germany

- [http://www.rst.e-technik.tu-dortmund.de/cms/en/research/robotics/TUDOR\\_engl/index.html](http://www.rst.e-technik.tu-dortmund.de/cms/en/research/robotics/TUDOR_engl/index.html)