Session: Linear Regression

One Variable

Gradient Descent

Many Variables

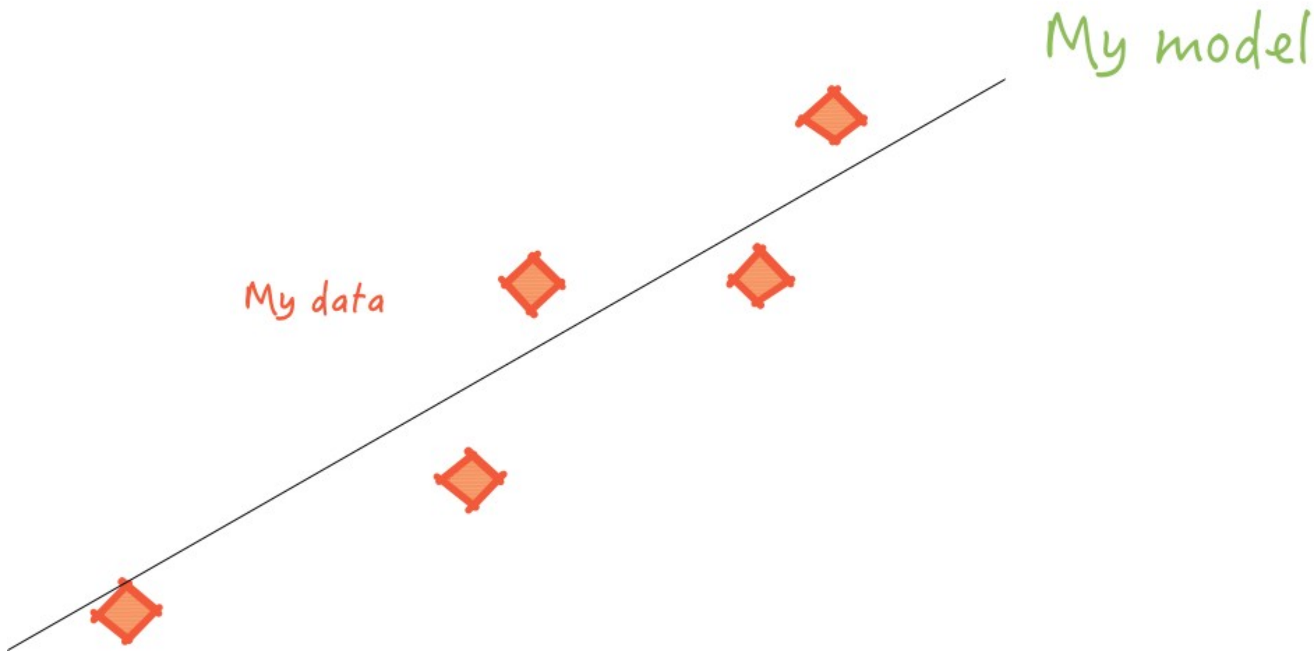
Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Lesson Objectives

- Understand linear regression with one variable
- Introduce the idea of the model
- Explain the gradient descent solution
- Understand linear regression with many variables

What Is Linear Regression?

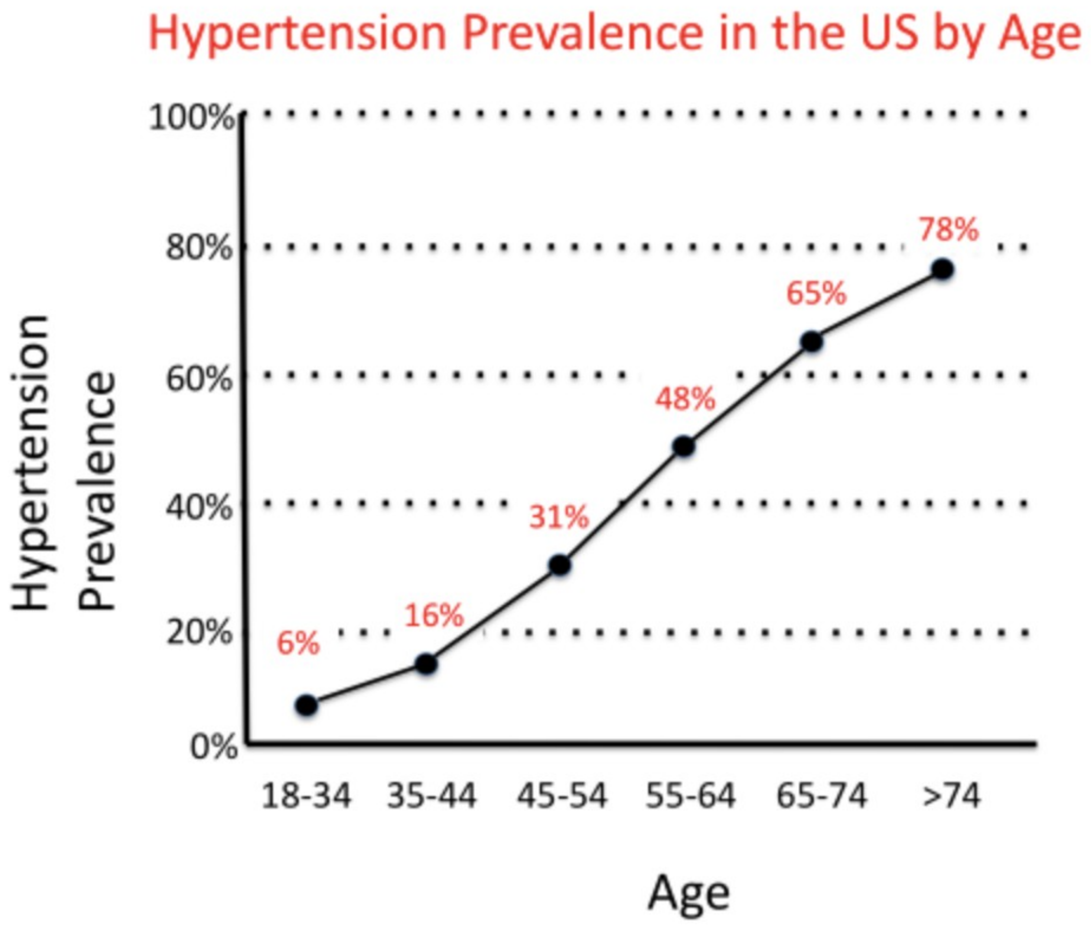
- Data



- Input variables ➔ continuous output

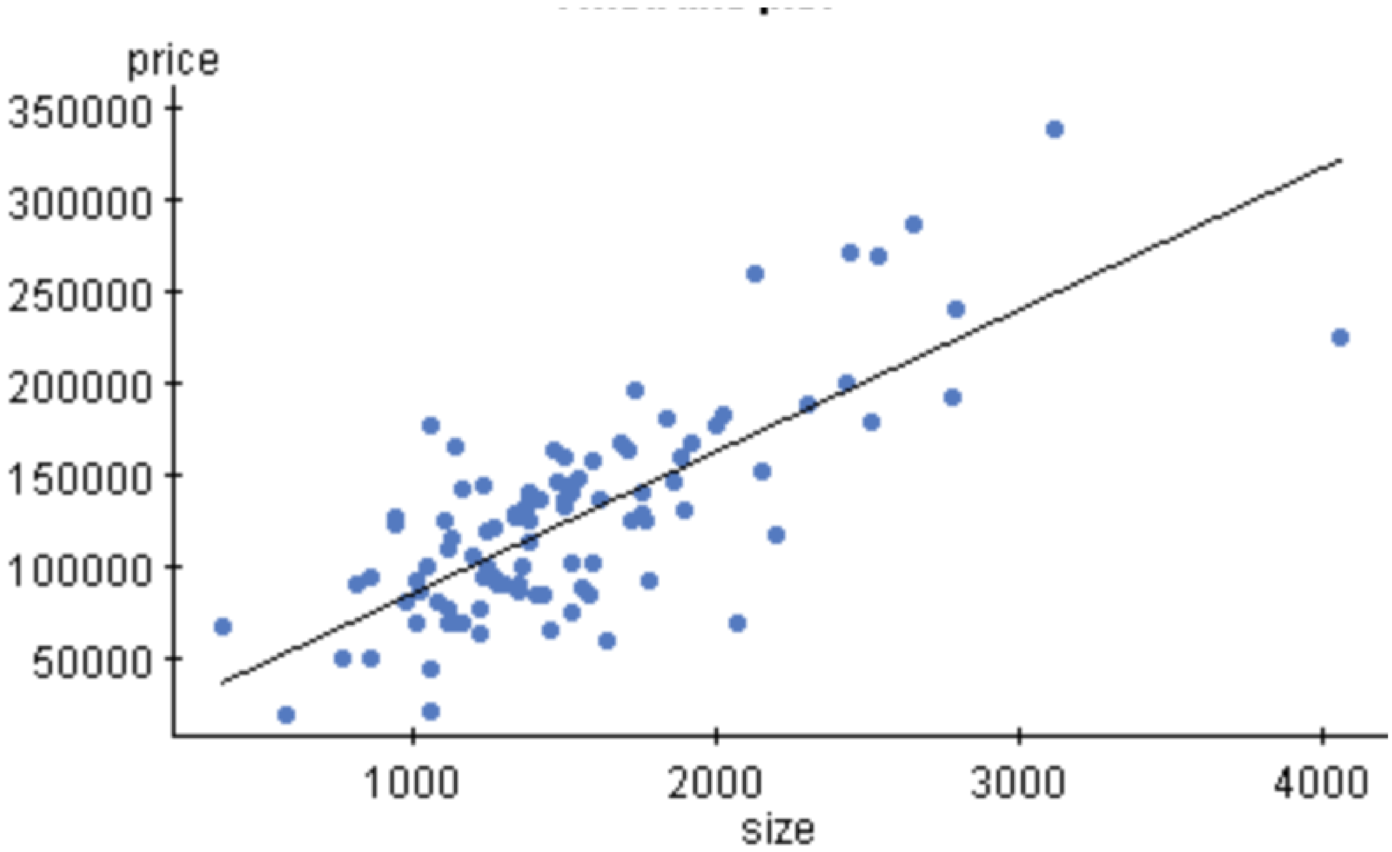
Linear Regression Example

- Blood pressure as function of age



Fields LE et al (2004) Hypertension 44:398-404

House Price vs Size



- Not a perfect fit
- More predictors may be needed

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

One Variable

➔ **One Variable**

Gradient Descent

Many Variables

SageMaker Linear Learner

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Linear Regression With One Variable

- **Univariate (simple) linear regression**

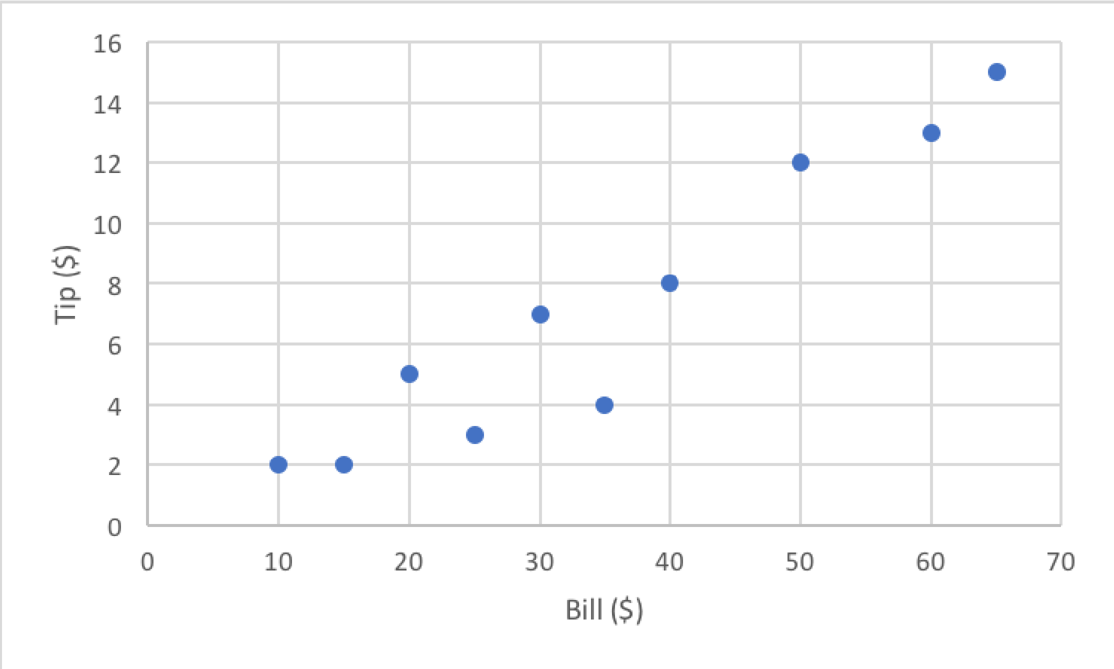
- One input ➔ One output
- x ➔ y
- Our hypothesis (or model)

$$y = T_0 + T_1 x$$

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Problem: Tip Calculation

- Bill ➔ Tip
- Imagine we do not know 15% rule
- Looks like a linear dependency



Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Solution Attempt

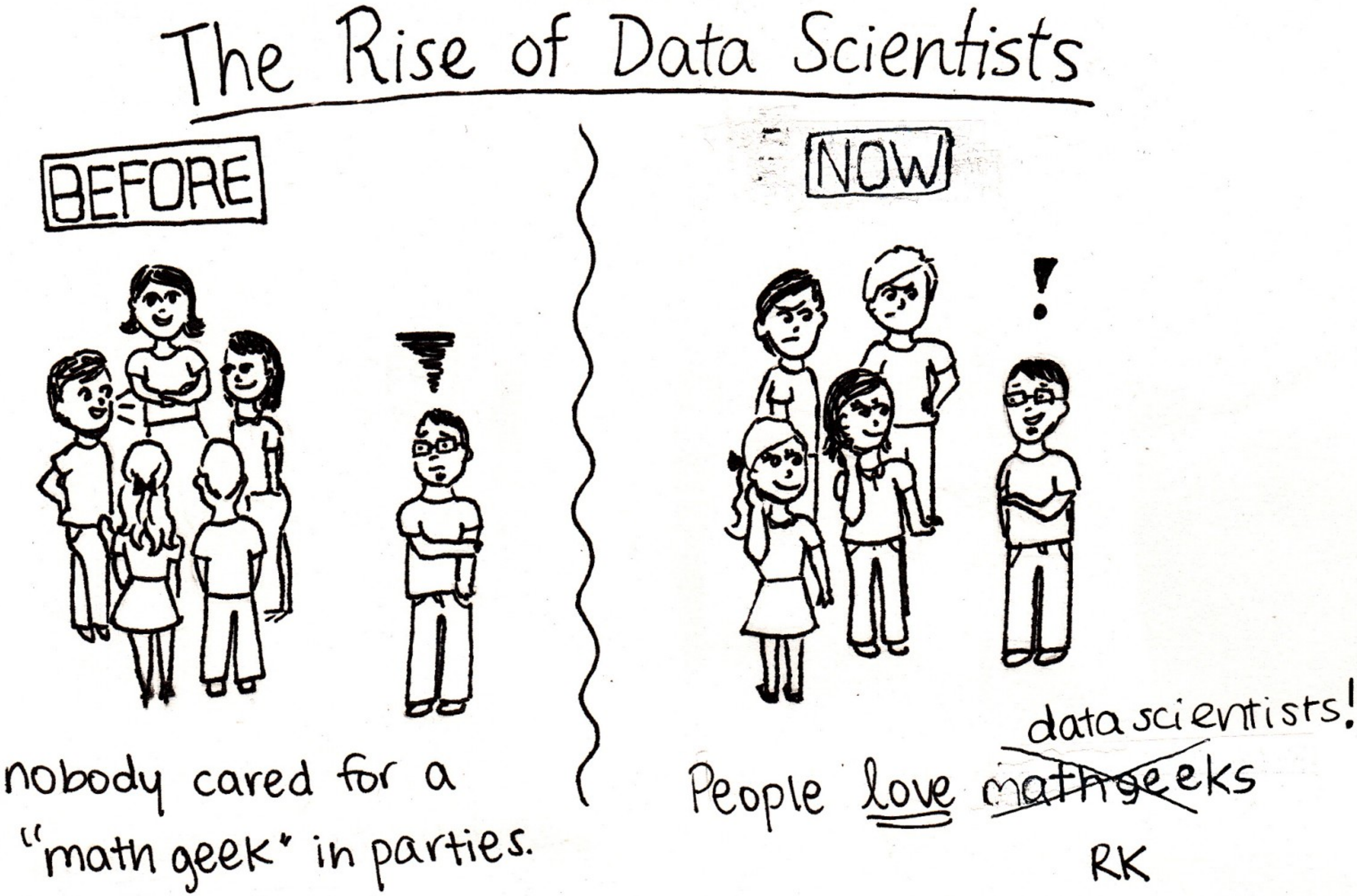
- Hypothesis

  tip = T_0 + T_1 bill

- But what are T_0 and T_1?
- Say T_0=0 and T_1=0.15
- Not a fantastic success
- ☹

Prediction
7.5
4.5
9
6
9.75
3
1.5
2.25
3.75
5.25

We Need a Data Scientist!



Cost Function

- Cost of replacing data with our model
- Measures the accuracy of our hypothesis (model)

$$C(T_0, T_1) = \frac{1}{2m} \sum_{1}^{m} (y'_i - y_i)^2$$

- Where
  - y' is estimated
  - y is real value

- Where
  - y' is estimated
  - y is real value

Cost Function Explained

- Error

- Error

$$h_T(x_i) - y_i$$

- Square error

- Square error

$$\sum_{1}^{m} (h_T(x_i) - y_i)^2$$

- Special multiplier

- Special multiplier

$$\frac{1}{2m}$$

- To do **mean** and to cancel out in subsequent gradient descent

- To do **mean** and to cancel out in subsequent gradient descent

Cost Function Breakdown

-

- Better, use function h_T instead of

$$C(T_0, T_1) = \frac{1}{2m}\sum_{1}^{m}(y'_i - y_i)^2$$

- Better, use function h_T instead of y'_i

$$C(T_0, T_1) = \frac{1}{2m}\sum_{1}^{m}(h_T(xi) - y_i)^2$$

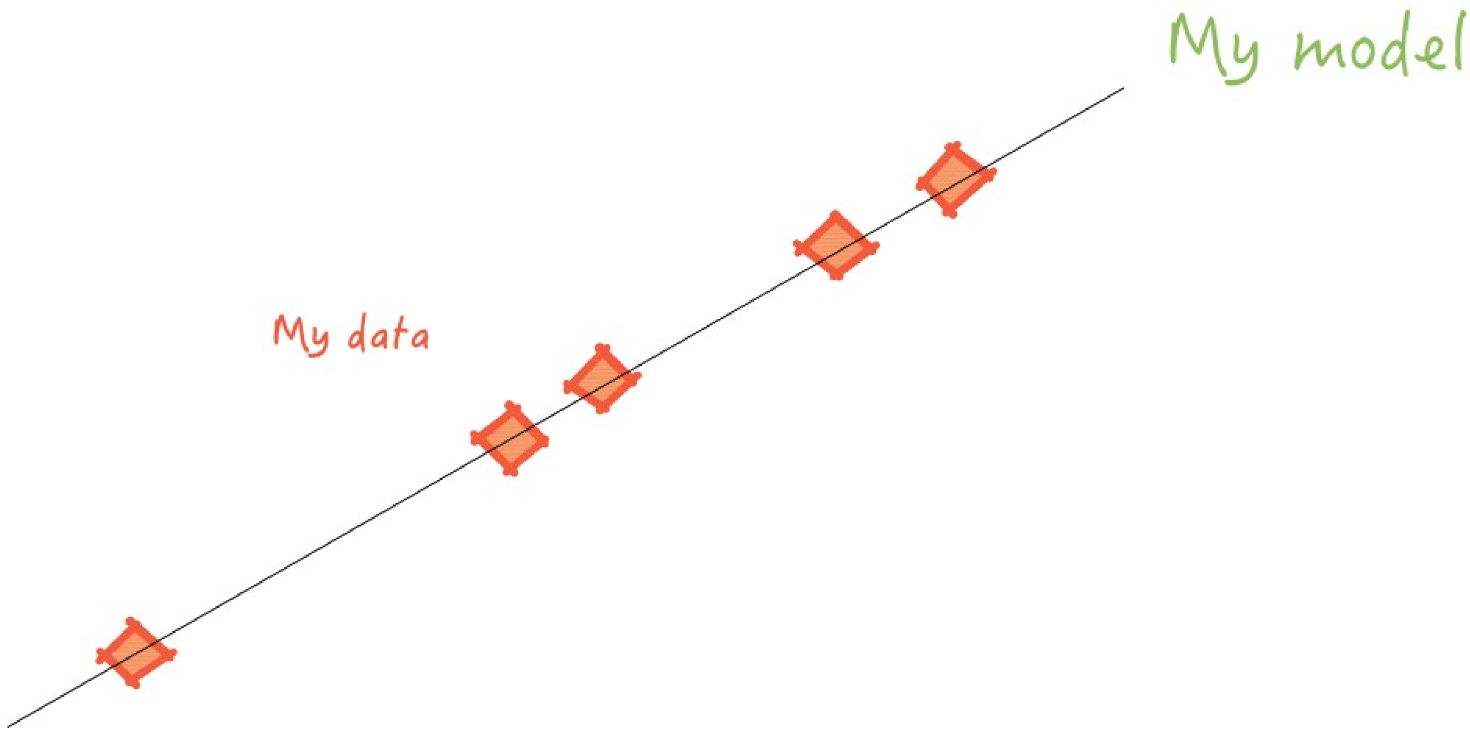
Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Here the Cost Would Be Zero

- If the line passes through all points
- The the cost would be zero

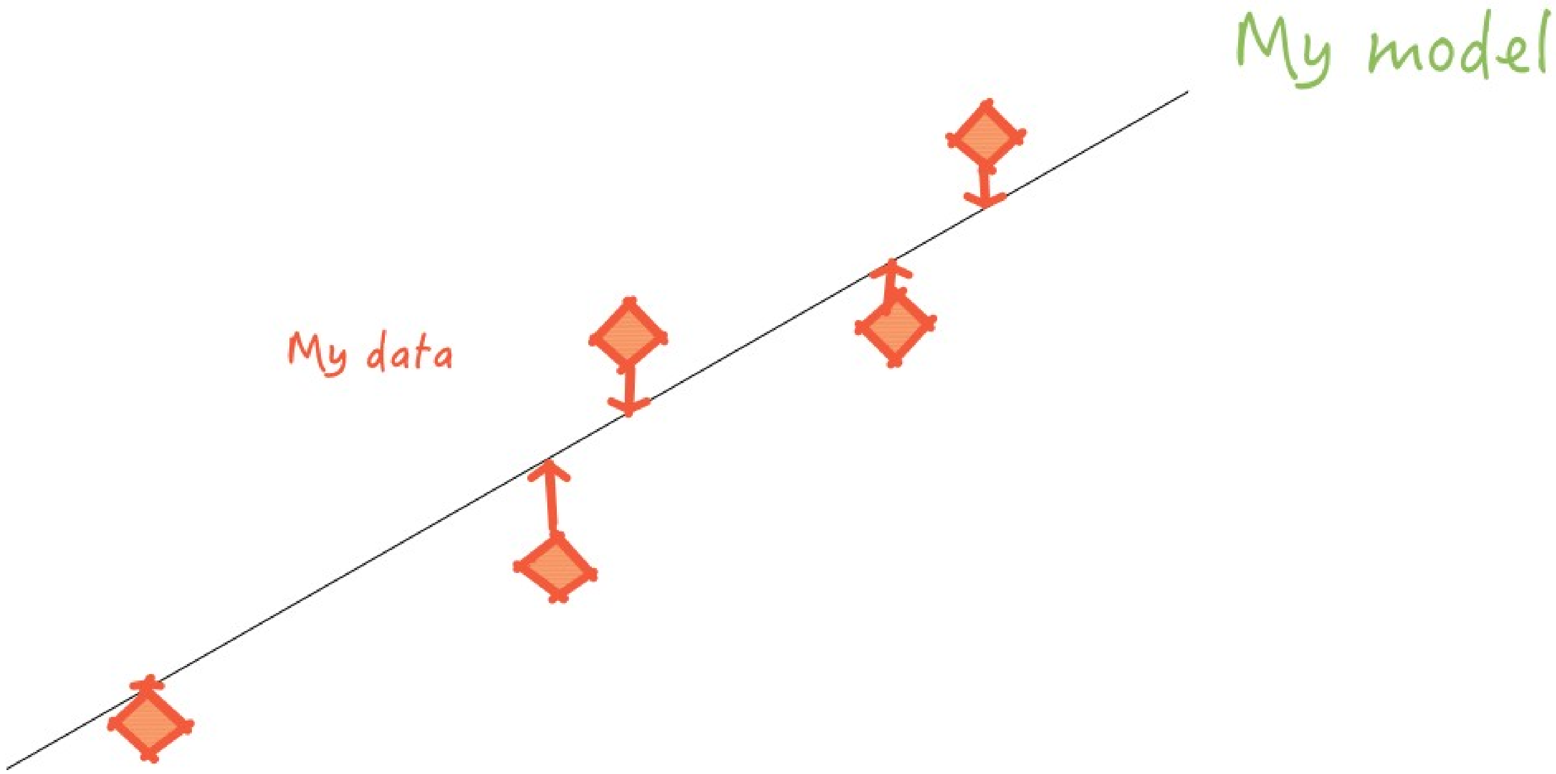
0

$$C(T_0, T_1) = 0$$



Cost Function Illuminated

- We want to find the best possible line to approximate the data
- Each line will have a cost
- We want to find a line that will minimize the cost



Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Gradient Descent

One Variable

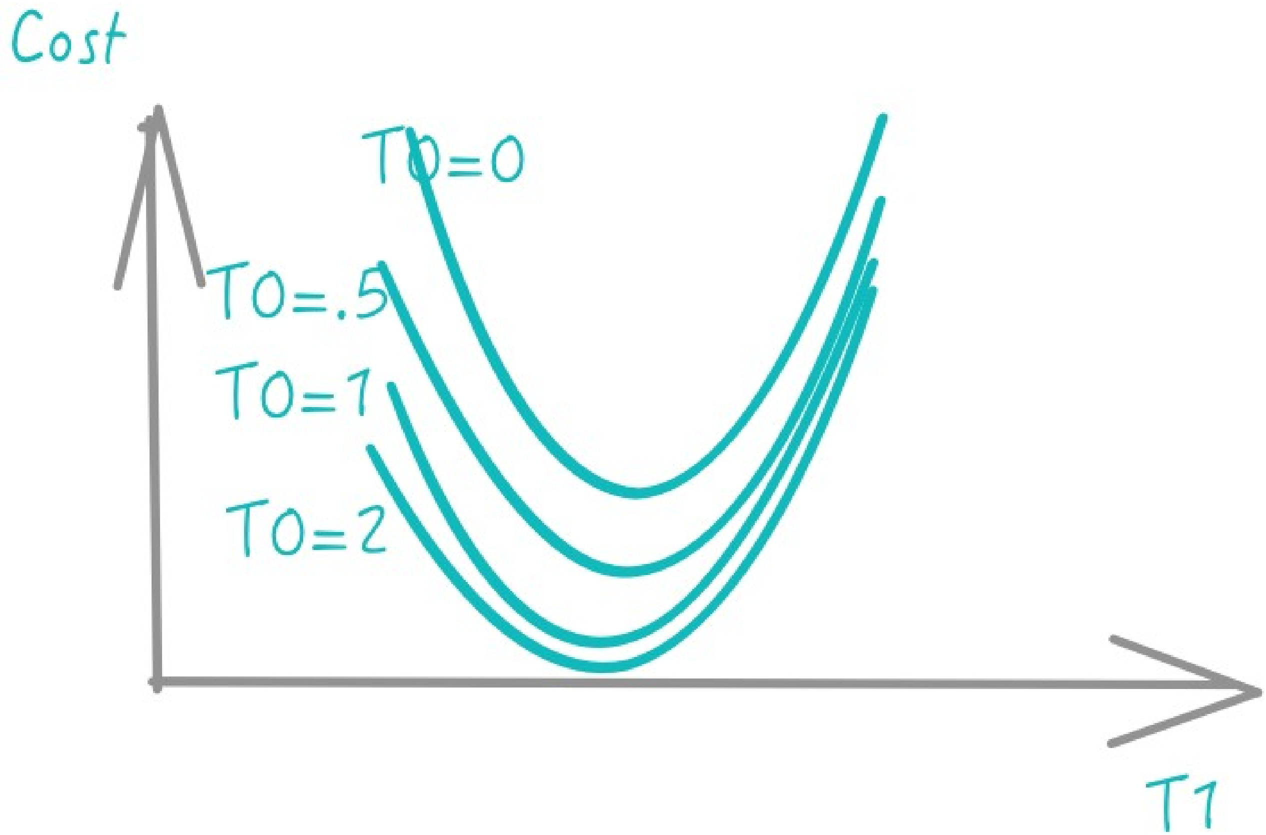
➔ **Gradient Descent**

Many Variables

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Optimization Problem

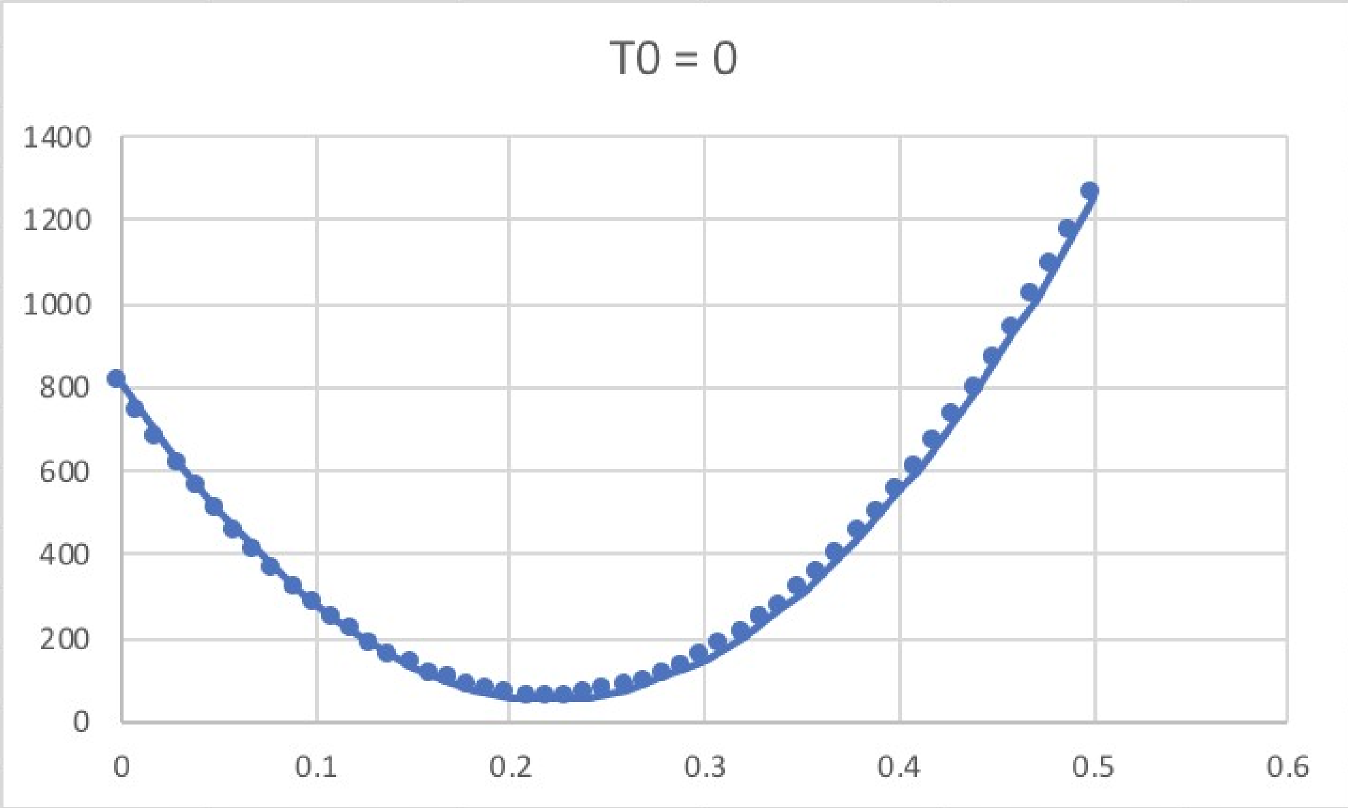
- We have the cost formula
- We can calculate the cost for every T_0 and T_1
- We want to find the best values of T_0 and T_1



Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

MSE for T_0=0.0

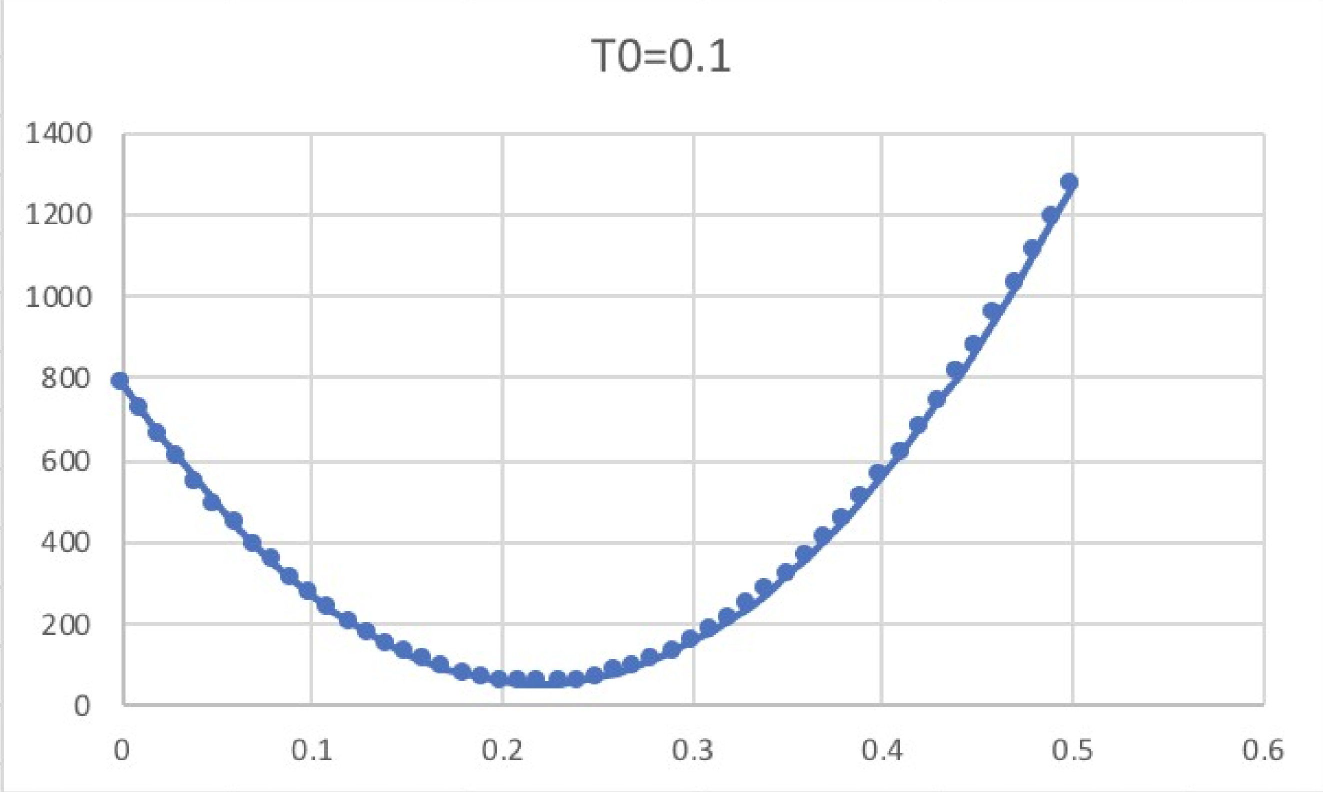
1	0	804
2	0.01	737.54
3	0.02	674.16
4	0.03	613.86
5	0.04	556.64
6	0.05	502.5
7	0.06	451.44
8	0.07	403.46
9	0.08	358.56
10	0.09	316.74
11	0.1	278
12	0.11	242.34
13	0.12	209.76
14	0.13	180.26
15	0.14	153.84
16	0.15	130.5
17	0.16	110.24
18	0.17	93.06
19	0.18	78.96



Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

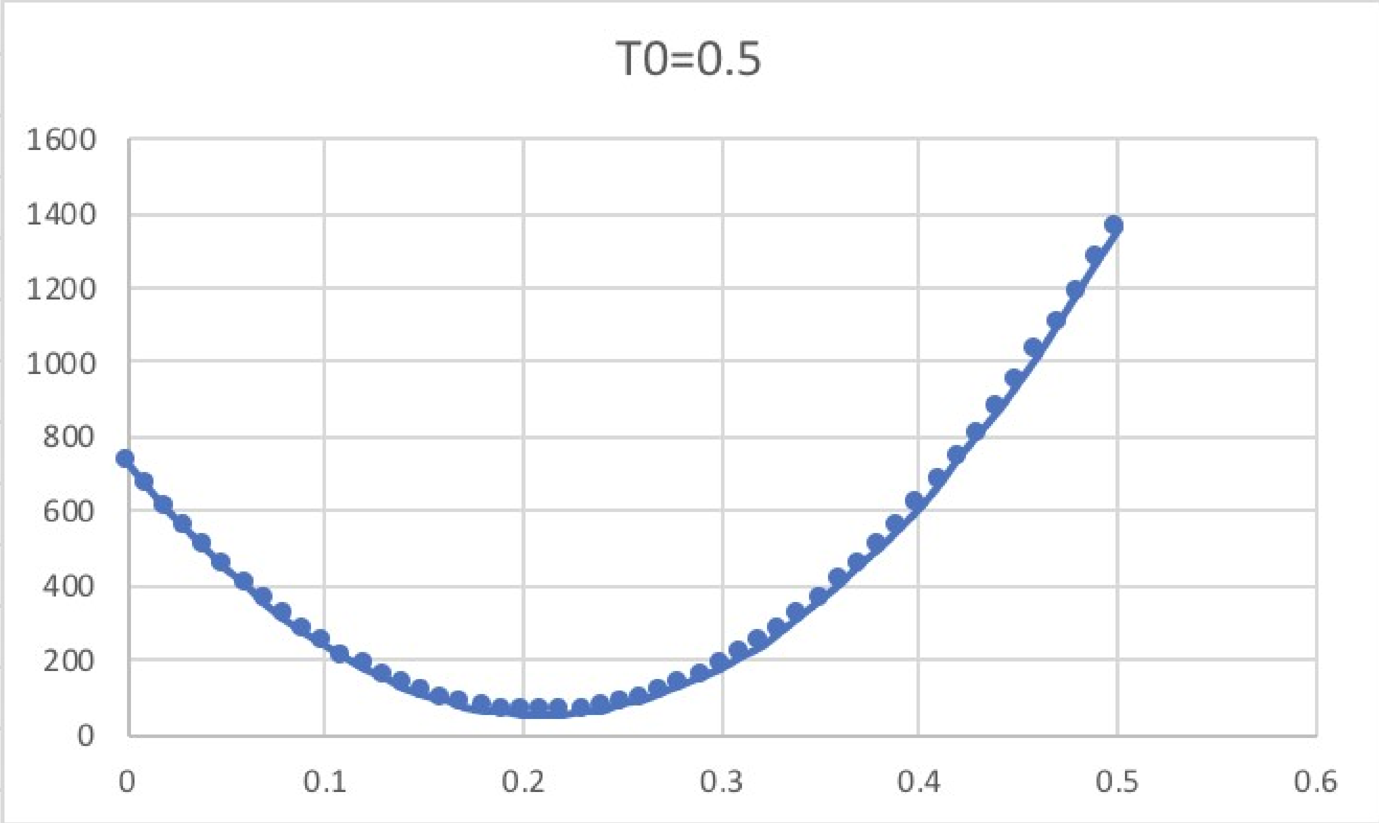
MSE for T_0=0.1

	A	B
1	0	788.9
2	0.01	723.14
3	0.02	660.46
4	0.03	600.86
5	0.04	544.34
6	0.05	490.9
7	0.06	440.54
8	0.07	393.26
9	0.08	349.06
10	0.09	307.94
11	0.1	269.9
12	0.11	234.94
13	0.12	203.06
14	0.13	174.26
15	0.14	148.54
16	0.15	125.9
17	0.16	106.34
18	0.17	89.86
19	0.18	76.46



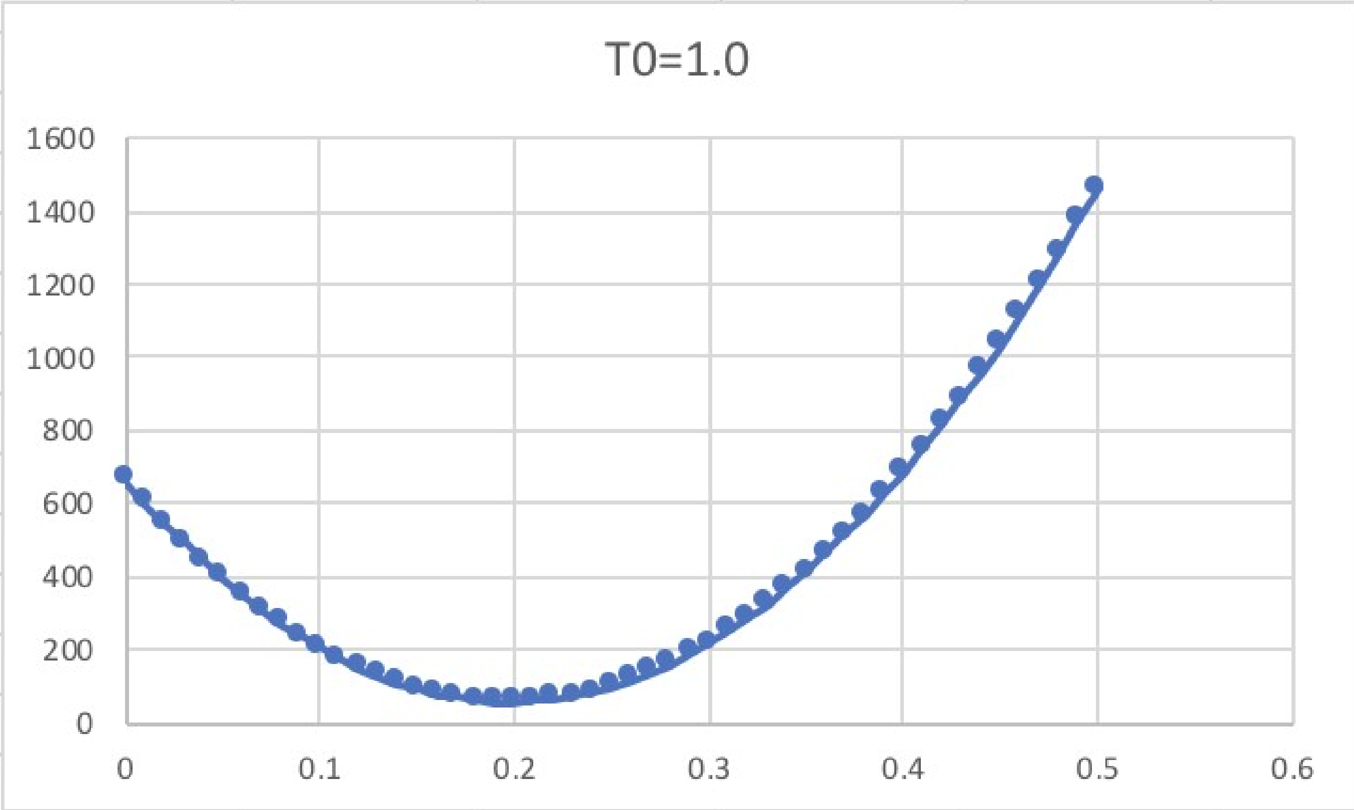
MSE for T_0=0.5

1	0	730.5
2	0.01	667.54
3	0.02	607.66
4	0.03	550.86
5	0.04	497.14
6	0.05	446.5
7	0.06	398.94
8	0.07	354.46
9	0.08	313.06
10	0.09	274.74
11	0.1	239.5
12	0.11	207.34
13	0.12	178.26
14	0.13	152.26
15	0.14	129.34
16	0.15	109.5
17	0.16	92.74
18	0.17	79.06
19	0.18	68.46



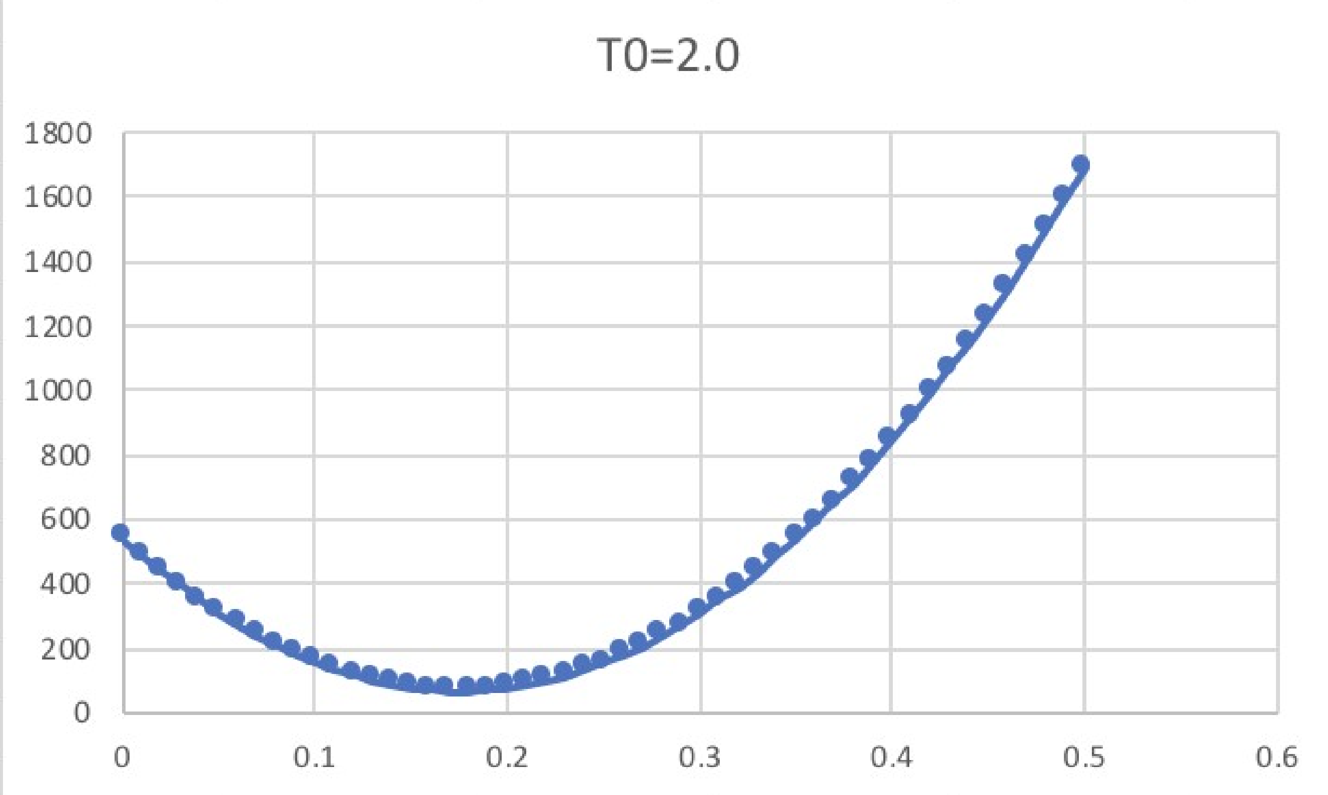
MSE for T_0=1.0

1	0	662
2	0.01	602.54
3	0.02	546.16
4	0.03	492.86
5	0.04	442.64
6	0.05	395.5
7	0.06	351.44
8	0.07	310.46
9	0.08	272.56
10	0.09	237.74
11	0.1	206
12	0.11	177.34
13	0.12	151.76
14	0.13	129.26
15	0.14	109.84
16	0.15	93.5
17	0.16	80.24
18	0.17	70.06
19	0.18	62.96

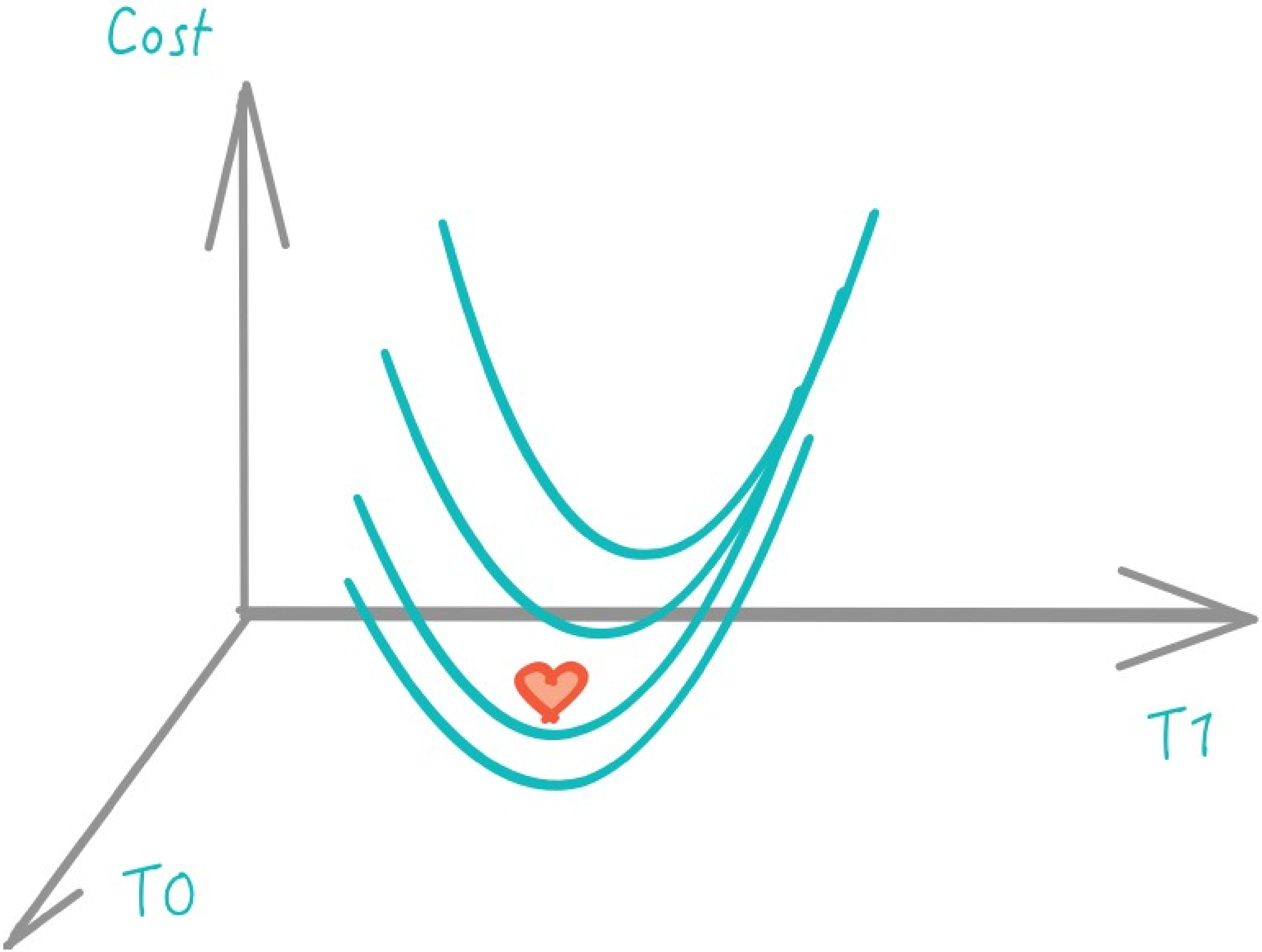


MSE for T_0=2.0

1	0	540
2	0.01	487.54
3	0.02	438.16
4	0.03	391.86
5	0.04	348.64
6	0.05	308.5
7	0.06	271.44
8	0.07	237.46
9	0.08	206.56
10	0.09	178.74
11	0.1	154
12	0.11	132.34
13	0.12	113.76
14	0.13	98.26
15	0.14	85.84
16	0.15	76.5
17	0.16	70.24
18	0.17	67.06
19	0.18	66.96

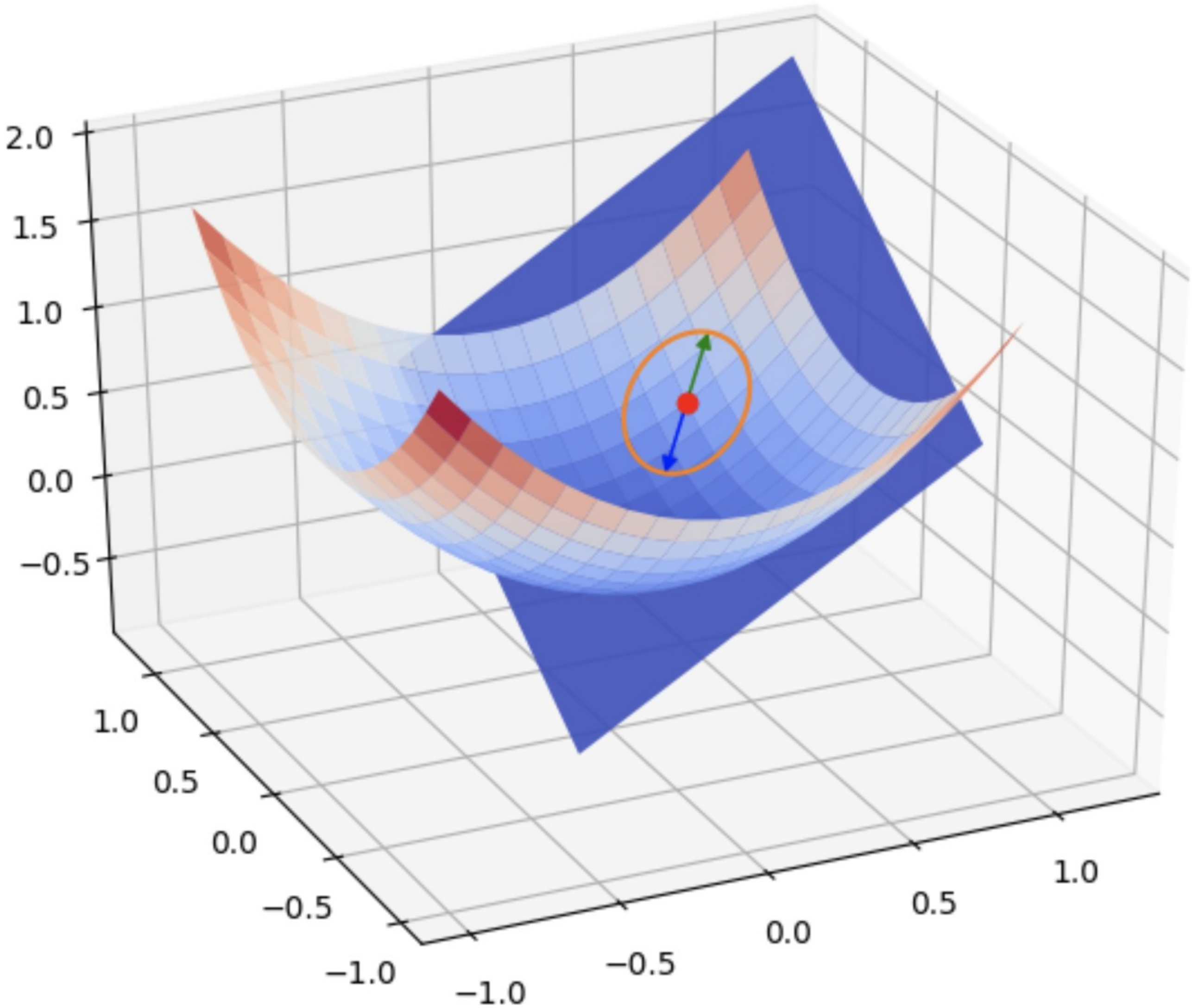


Cost vs T_0 and T_1



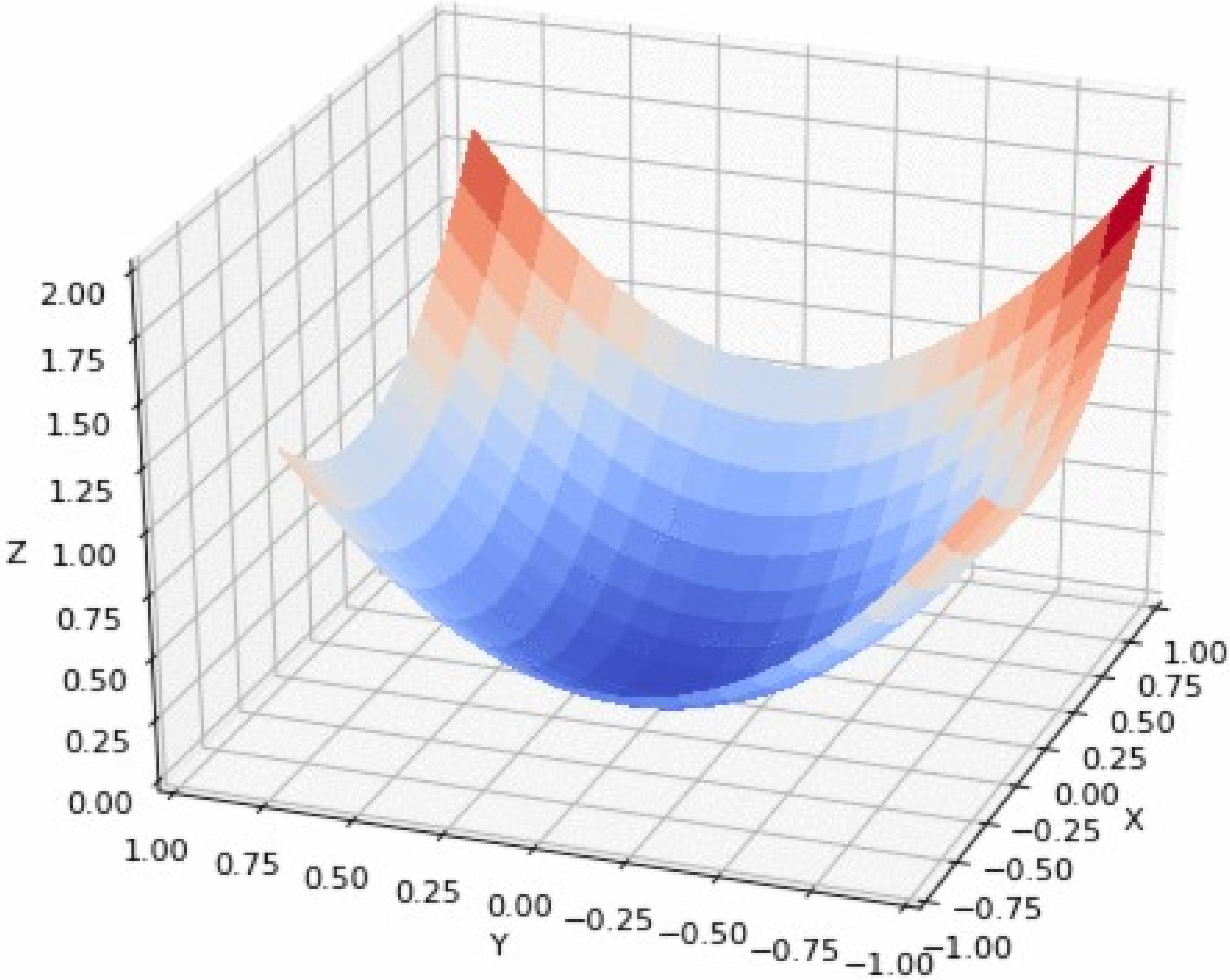
Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

Gradient Descent Idea

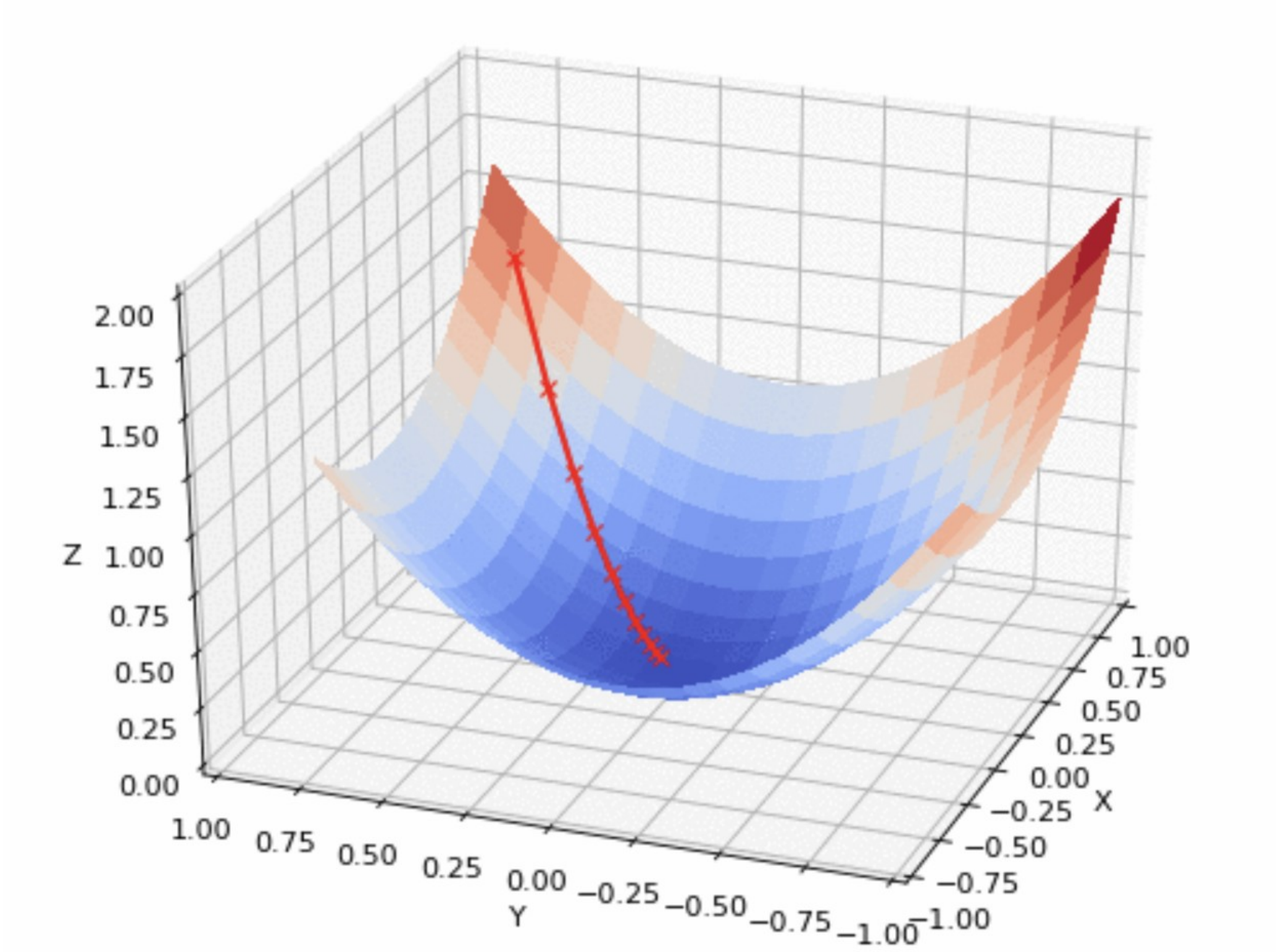


Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

Gradient Descent Demo



Gradient Descent Result



Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

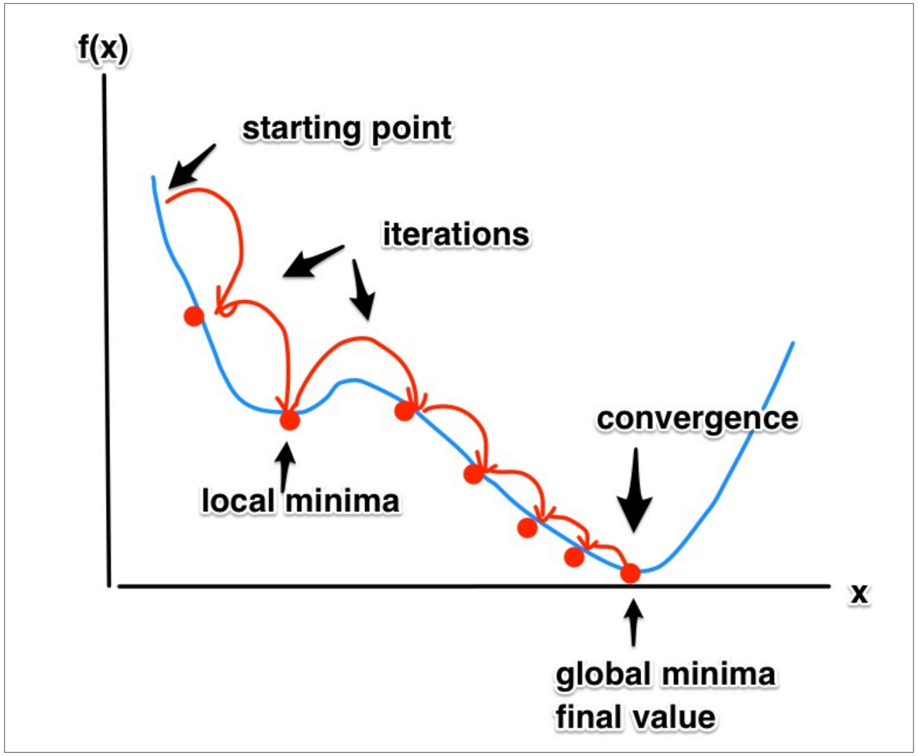
Algorithm for Gradient Descent

- $T = (T_0, T_1)$
  - Generally, $T = (T_0, T_1, T_2, \ldots, T_n)$

1. $T_{next} = Tcurrent + Direction * Step$
2. $Direction = \frac{dCost}{dT}$
3. $Step = Learning\ Step$

- Repeat until convergence
  - Next values are worse, or
  - The improvement is too small

- Repeat until convergence
  - Next values are worse, or
  - The improvement is too small



Batch Gradient Descent

- We did "**batch**" gradient descent
  - We used all training samples
  - There are versions that will use some groups of samples
- There exists a precise solution in linear algebra
  - But gradient descent scales better
- Gradient descent will be used everywhere

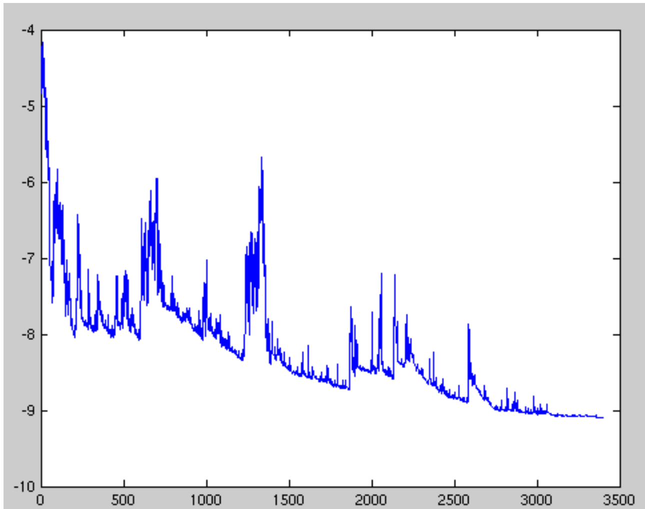
Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Stochastic Gradient Descent

- For each step of the Gradient Descent, we need a derivative

$$Direction = \frac{dCost}{dT}$$

- How many calculations is that?
- How many calculations is that?
  - Proportionate to m (cost)
  - Proportionate to m (cost)
  - Proportionate to n (predictors)
  - Proportionate to n (predictors)
- Stochastic Gradient Descent
- Stochastic Gradient Descent
  - samples data at each step
  - samples data at each step



Lab: Linear Regressions

Lab

- **Overview**:
  Practice Linear Regression

- **Approximate Time**:
  30 mins

- **Instructions**:
  - **Linear-regression / 1-lr**
  - Follow appropriate Python / R / Spark instructions
  - (1T-lr-tips.ipynb, 1K-lr-tips.ipynb)

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Many Variables

One Variable

Gradient Descent

➔**Many Variables**

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Problem: House Prices

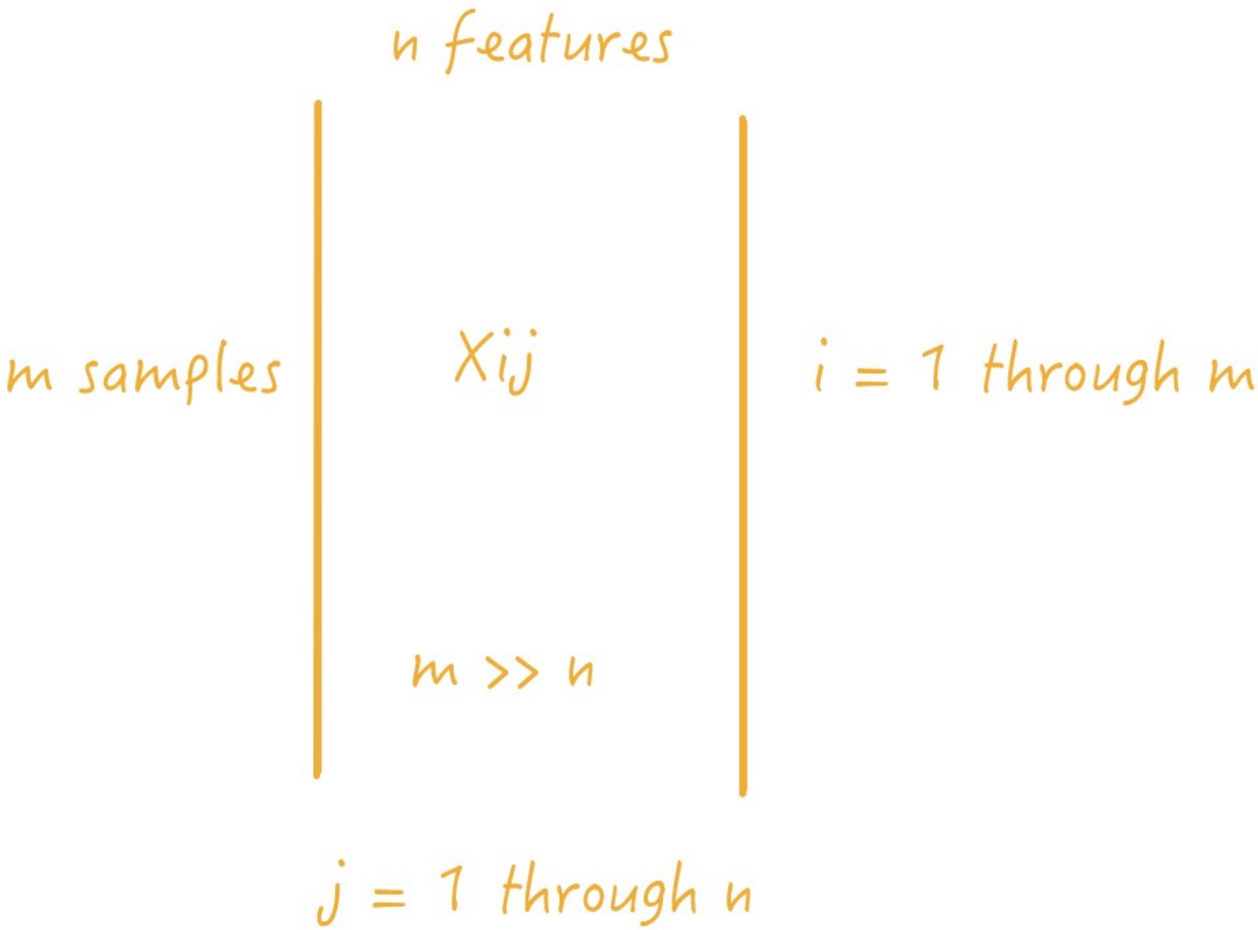
Sale Price $	Bedrooms	Bathrooms	Sq.ft Living	Sq.ft Lot
280,000	6	3	2,400	9,373
1,000,000	4	3.75	3,764	20,156
745,000	4	1.75	2.060	26,036
425,000	5	3.75	3,200	8,618
240,000	4	1.75	1,720	8,620
327,000	3	1.5	1,750	34,465
347,000	4	1.75	1,860	14,650

- Multiple factors are needed to predict house prices
- This is called **multiple linear regression**
- Terminology note
  - "Multivariable" would mean a vector output, not scalar

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

Our Data is a Matrix

- What are the m and n for the previous slides?



Linear Regression With Many Variables

- **Multiple linear regression**

- Many inputs ➔ One output
- $(x_1, x_2, x_3, \ldots, x_n)$ ➔ y
- Our hypothesis (or model)

$$y = T_0 + T_1 x_1 + T_2 x_2 + T_3 x_3 + \ldots + T_n x_n$$

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Cost Function

- Cost of replacing data with our model
- Measures the accuracy of our hypothesis (model)

$$C(T_0, T_1, , , Tn) = \frac{1}{2m} \sum_{1}^{m} (y'_i - y_i)^2$$

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

Cost Function Breakdown

- Better, use function h_T instead of

$$C(T_0, T_1, , , Tn) = \frac{1}{2m}\sum_{1}^{m}(y'_i - y_i)^2$$

- Where
- Better, use function h_T instead of y'_i

$$C(T_0, T_1, , , Tn) = \frac{1}{2m}\sum_{1}^{m}(h_T(x_i) - y_i)^2$$

- Where

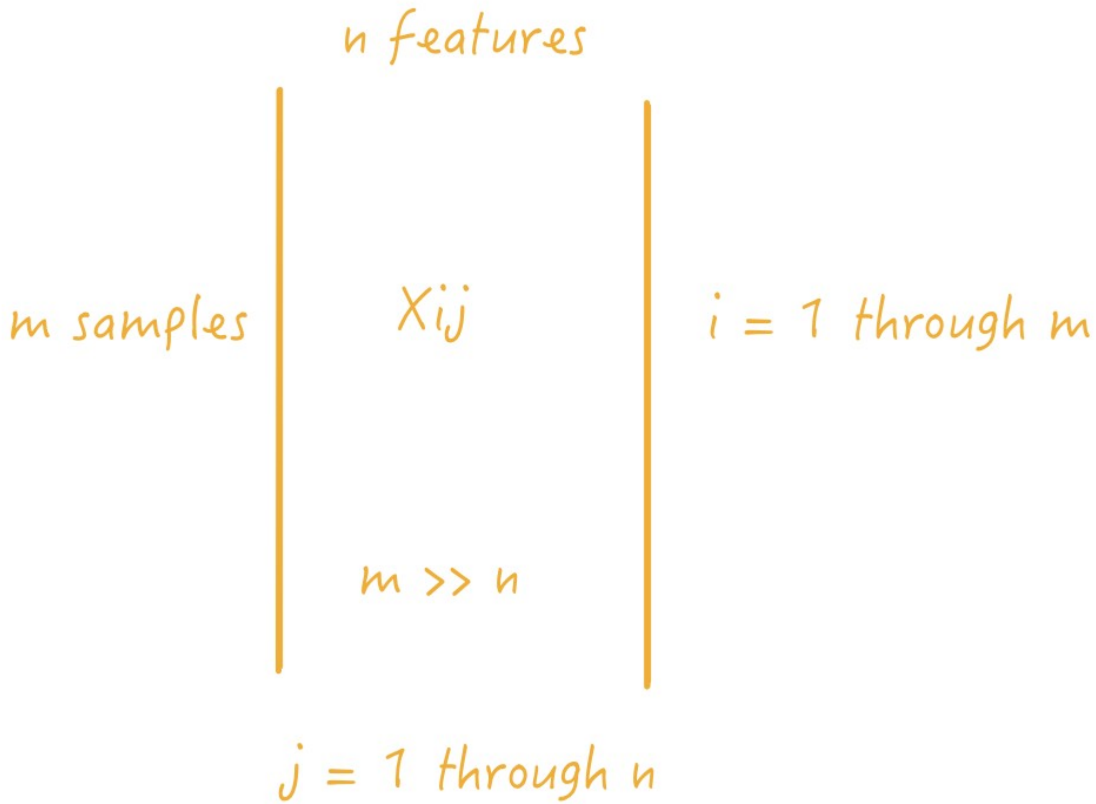
$$h_{T(x_i)} = T_0 + T_1x_1 + T_2x_2 + \ \dots + T_nx_n$$

Solution Advice

- Verify the dimensions
- Feature scaling
- Choose learning rate

Verify the Dimensions

- Number of features: n
- Feature index j = 1 to n
- Number of data points: m
- Data index i = 1 to m



Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Feature Scaling

- Approximately

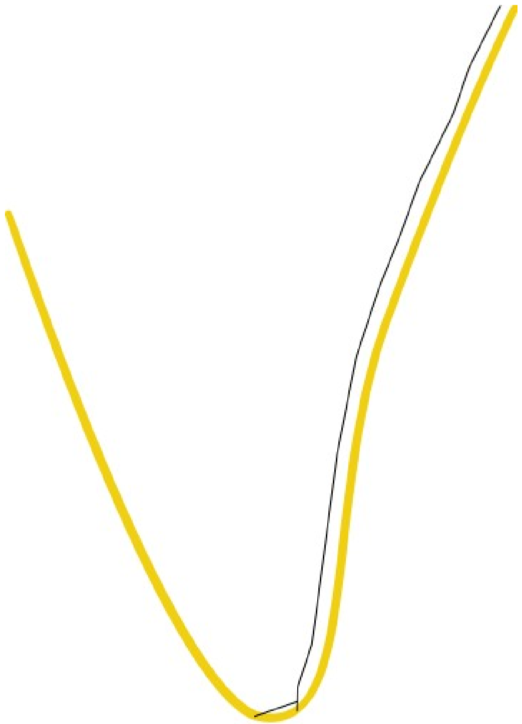
$$-0.5 \leq x_j \leq 0.5$$

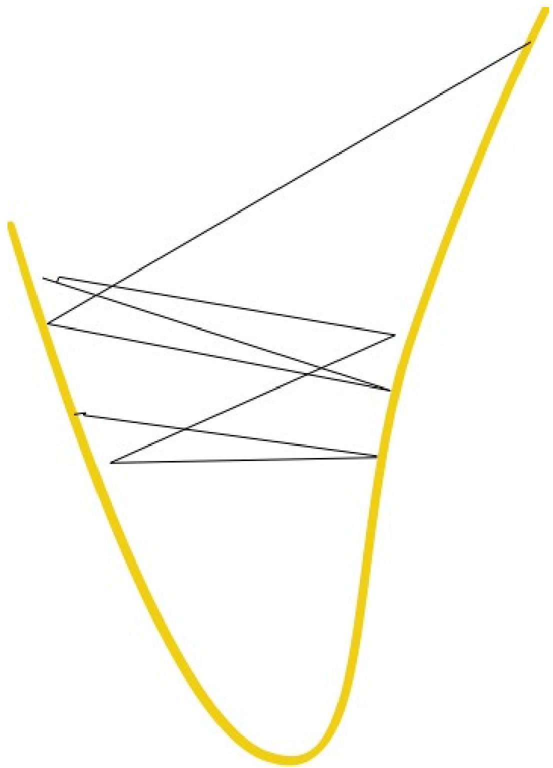
- For example, floor area

$$x_3 = \frac{sq.ft - 2000}{4000}$$

Chose Learning Rate

- Sufficiently small learning rate gives always improving cost
- Avoid jumps up and down
- Good Bad





Lab: Multiple Linear Regression

Lab

- **Overview**:
  Practice Multiple Linear Regressions
- **Approximate Time**:
  30 mins
- **Instructions**:
  Follow appropriate Python / R / Spark instructions
  - LIR-2: House prices
  - (2T-mlr-house-prices.ipynb, 2K-mlr-house-prices.ipynb)

Regularization

- Regularization seeks to
  - Minimize RSS of model
  - And to reduce the complexity of the model
- How to reduce the complexity of model
  - Removing unnecessary coefficients (b_1, b_2 ..etc)
  - Keeping coefficient values getting too large (**parameter shrinkage**)
    - Large coefficients amplify certain parameters
- **Minimize = RSS + ƛ * penalty on the parameters**
- Two types of regularizations
  - **Lasso Regression** (**L1 regularization**): Minimizes the absolute sum of the coefficients
  - **Ridge Regression (L2 regularization)** : Minimizes the squared absolute sum of the coefficients

$$Y = b_0 + b_1X_1 + b_2X_2 + \ \dots + b_pX_p + e$$

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Ridge vs. Lasso

- Ridge regression will **'minimize' coefficients** but not to zero
  - Called **parameter shrinkage**
- Lasso regression **can shrink parameters can also set them zero**!
  - By setting some coefficients to zero, it eliminates certain features
  - Called **variable/feature selection**
- Lambda (ƛ) can be calculated using cross validation

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_p X_p + e$$

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12

Lab: Multiple Linear Regression

Lab

- **Overview**:
  Practice Multiple Linear Regressions

- **Approximate Time**:
  30 mins

- **Instructions**:
  Follow appropriate Python / R / Spark instructions
  - LIR-2: House prices
  - LIR-3: AIC

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

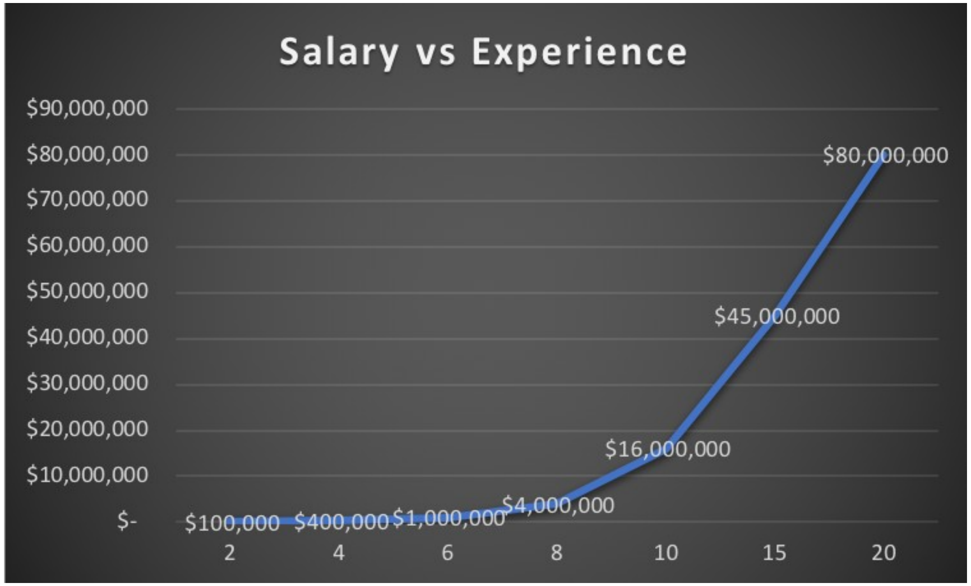
Preparing Data for Linear Regressions

- **Linear Assumption:**
  Linear Regression assumes linear relationship between input and output.
  May be need to transform data (e.g. log transform) to make the relationship linear
- **Remove Noise:**
  Remove outlier data
- **Remove Collinearity**:
  Linear regression will over-fit your data when you have highly correlated input variables
- **Gaussian Distributions**
  Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution.
  Transform data (e.g. logarithmic) make their distribution more Gaussian looking for better results

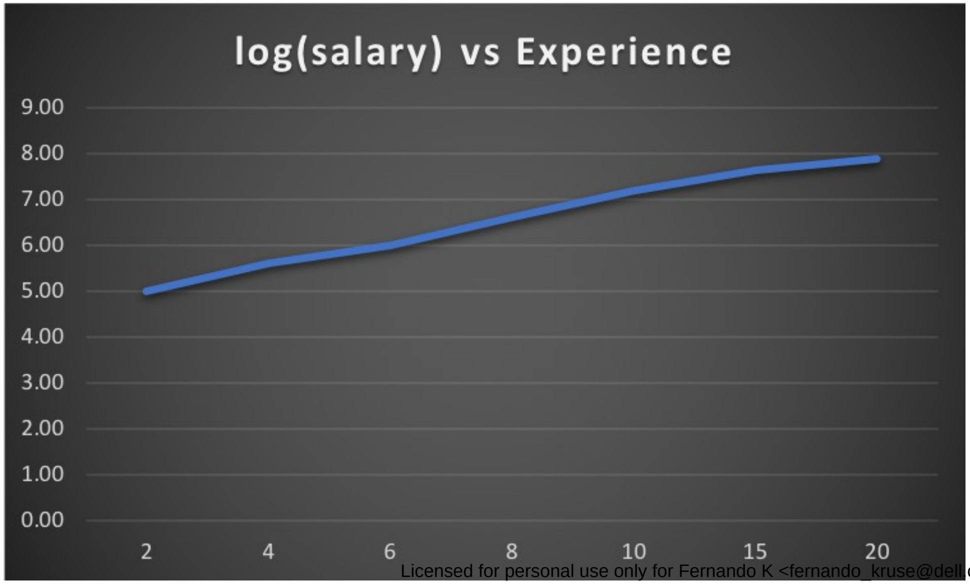
Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @ 2019-03-12

Preparing Data for Linear Regression

experience (years)	salary	log(salary)
2	$ 100,000	5.00
4	$ 400,000	5.60
6	$ 1,000,000	6.00
8	$ 4,000,000	6.60
10	$ 16,000,000	7.20
15	$ 45,000,000	7.65
20	$ 80,000,000	7.90



Actual data.
Not Linear



Data scaled at logarithmic scale.
Linear!

Review Questions

- What is Linear Regression?

Licensed for personal use only for Fernando K <fernando_kruse@dell.com> from Machine Learning at Dell Brazil (QE) @
2019-03-12