Ahmed Ibrahim

ECE 9039/9309 MACHINE LEARNING

Outline

- Course Information
  - Instructor Information
  - Class Schedule
  - Evaluation
- Machine Learning Landscape
- Linear Regression

Class Schedule & More

Lectures

Monday: 3:30 pm – 6:30 pm – **P&AB 148** (Physics & Astronomy Building)

Consultation hours

Thursday: 6:30 pm – 7:30 pm @ **P&AB 148** Or by appointment

Email **–** aibrah64 AT UWO

Contact – Please contact me through email

Read the e-mail policy **FIRST**!

Required Text / References

- **Notes and other supplemental materials:**
  - http://owl.uwo.ca
  - Books (Course outline)
  - Book chapters, papers, etc…

Course Website

- https://owl.uwo.ca
    - All course information (e.g., Lecture notes, Assignments, coding examples, …) will be posted on the website. Check the website regularly for course announcements.
- Python programming language, PyTorch framework, and various Python-based machine learning packages will be used in this course.
- Programming background; worked-through examples.
- Tools: https://colab.google/ , https://jupyter.org/

Prerequisites

- ECE 9063: Data Analytics Foundations Or ECE 9013 (Programming for Engineers) and ECE 9014 (Data Management & Applications)
- Knowledge of **probability** and **statistics**, **calculus**, and **linear algebra** is required, as well as strong programming skills in Python.
- **Enrollment Restrictions –** Enrollment in this course is restricted to graduate students in the Electrical and Computer Engineering Program, as well as any student who has obtained special permission to enroll in this course from the course instructor as well as the Graduate Chair (or equivalent) from the student's home program.

Evaluation

- Project 40%
  - Proposal, Presentation, Code, and Final Report
  - Project information and deadlines will be released before the end of January 2023.
  - Presentations will be scheduled during the last 3 weeks of the semester
- Final Exam 30%
  - The exam will include practical (Python programming) & theory components
  - You need to bring a laptop/computer to complete the exam (You are responsible for ensuring that your laptop/computer meets the specified technical requirements needed and is in good condition to comply with the conditions and requirements of the exam.)
  - The final examination will take place during the regular examination period
- Assignments (including programming assignments) 20%
- Quizzes 8%
  - Multiple-choice, programming-based, ..
- Participation 2%
  - Multiple-choice, programming-based take-home exercises

Attendance

You can use the provided link if you don't have a mobile phone or if your phone lacks a QR-Code reader – https://rebrand.ly/ECEJan16

1.0: Machine Learning Landscape

Challenging Problems

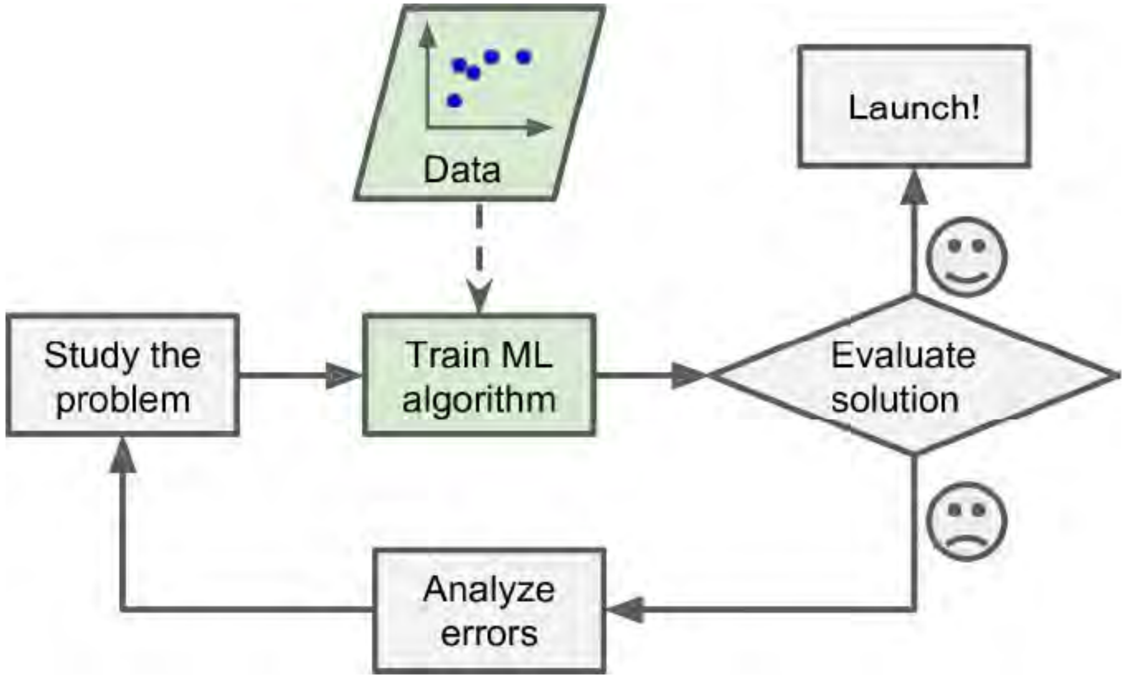
- For many problems, it's difficult to program the correct behavior by hand.
- **Problems with Inherent Complexity**
  - Ex. Natural Language Processing (NLP) – Understanding and generating human language, which is full of nuances, ambiguities, and contextual dependencies. Tasks like machine translation, sentiment analysis, and text summarization often require machine-learning approaches.
- **Problems with Uncertain or Incomplete Information**
  - Ex. Financial Forecasting – Predicting future market trends or stock prices, inherently uncertain due to various factors and interactions.
- **Problems with Continuously Changing Conditions**
  - Ex. Self-driving Cars: Navigating safely in real-world traffic, handling unpredictable events, and adjusting to changing road conditions.

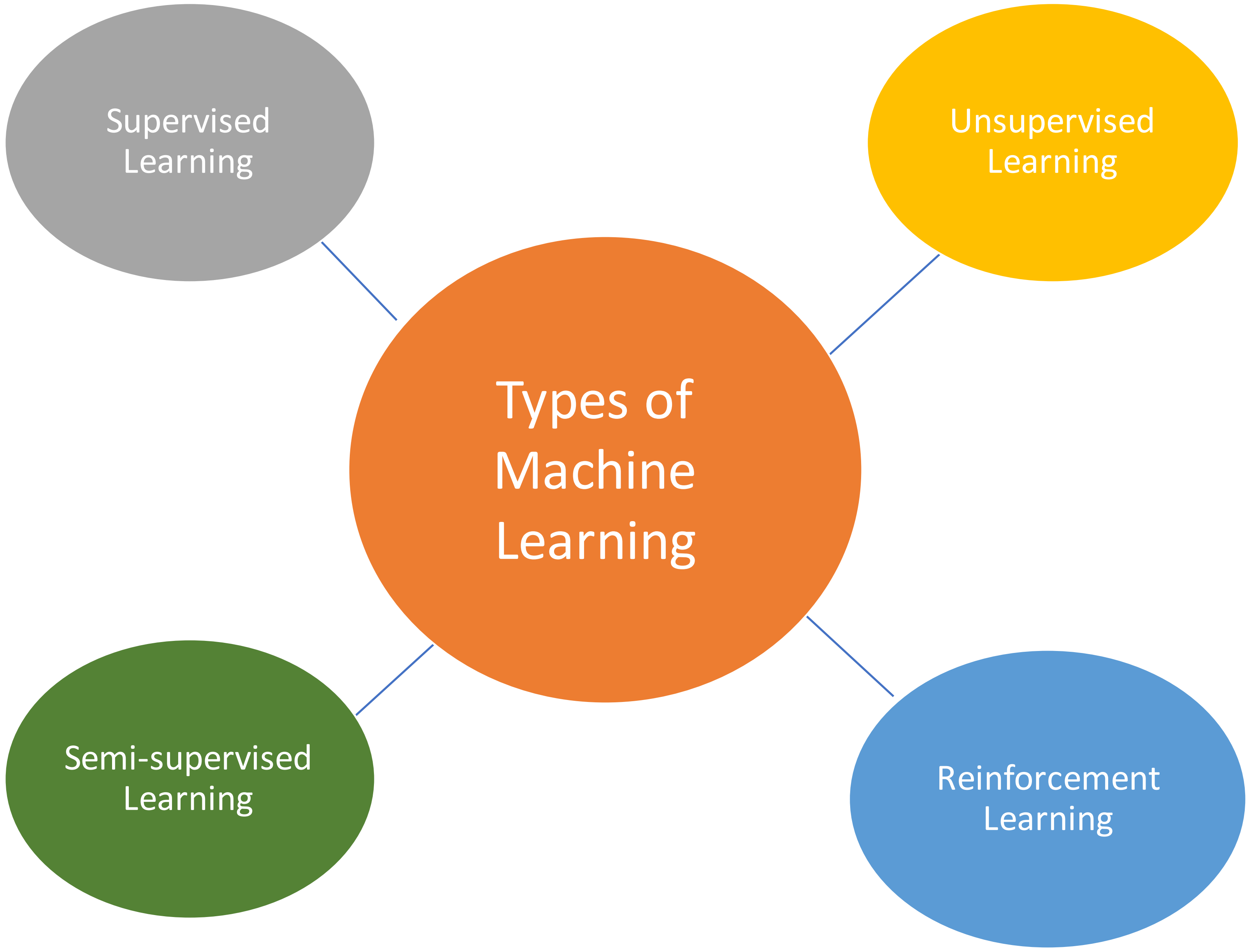
What is Machine Learning?

- Machine learning is a field of artificial intelligence that focuses on building systems that can learn from data.
- These systems rely on **learning algorithms** that are trained on large amounts of data, and they can then use this data to make predictions, classify information, or make decisions.

What is Machine Learning?

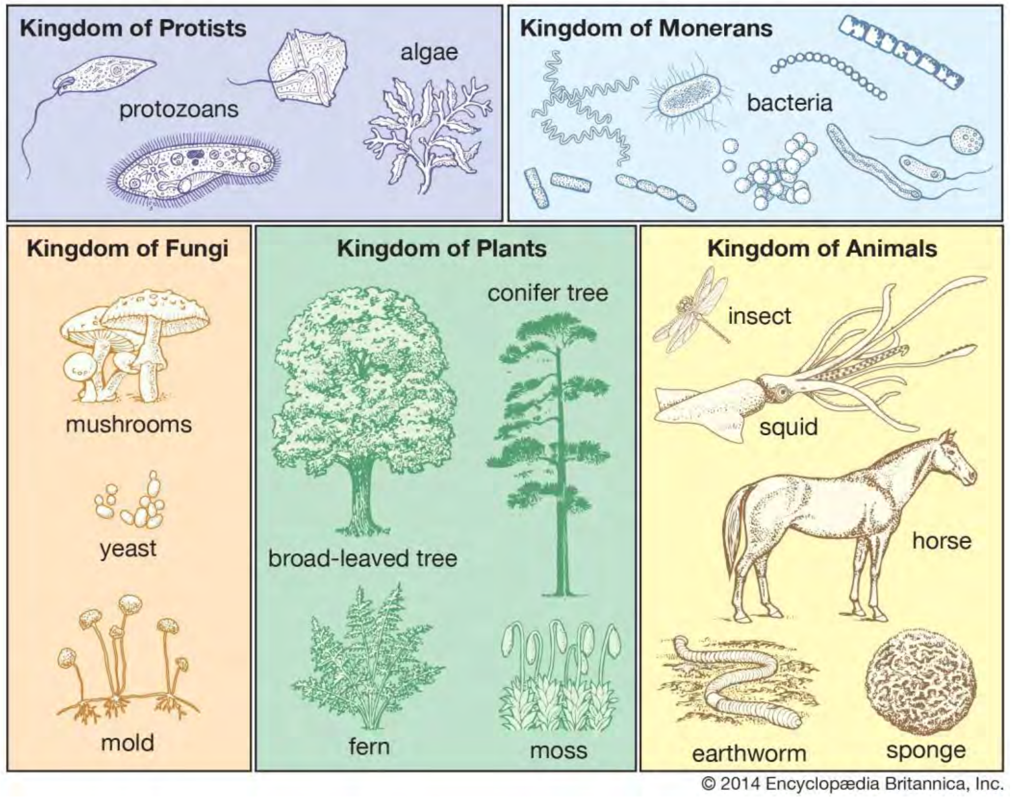
- **Input**: large amounts of data (*e.g., images, audio files, healthcare records, stock price*, …).
- **Output**: A prediction or decision (*e.g.* this image is a stop sign, this stock will go up 10% next quarter, turn the car right).



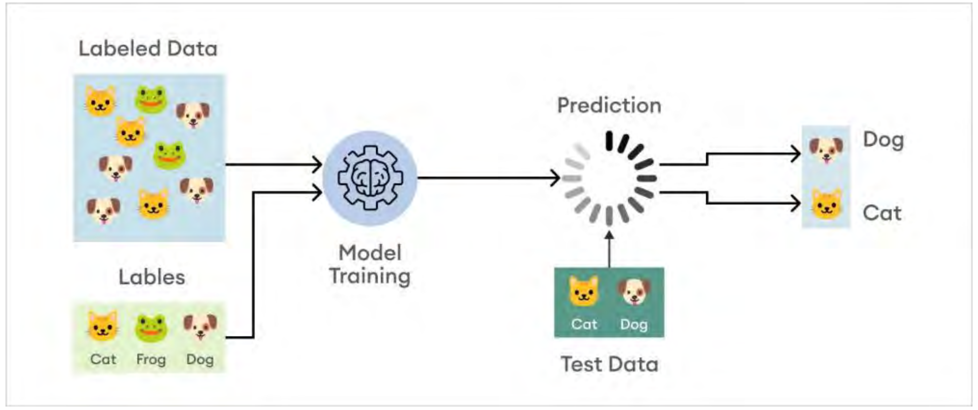

Types of Machine Learning
Supervised Learning
Unsupervised Learning
Semi-supervised Learning
Reinforcement Learning

Supervised Learning

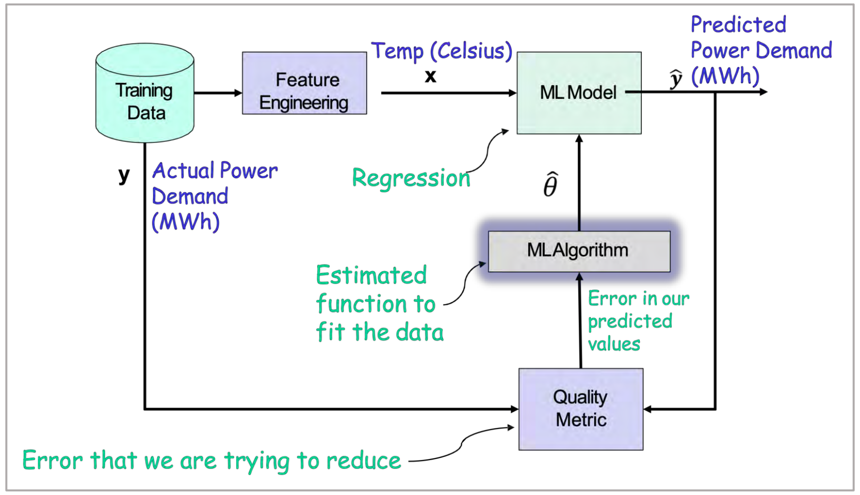
- Discover characteristics and make predictions about behaviors based on labeled examples of correct behavior
- Pre-existing knowledge
- Types of Supervised Learning
  - **Classification** – predict a discrete class label.
  - **Regression** – predict a continuous value (response variable).



Types of Supervised Learning



Source: https://www.superannotate.com/blog/image-classification-basics



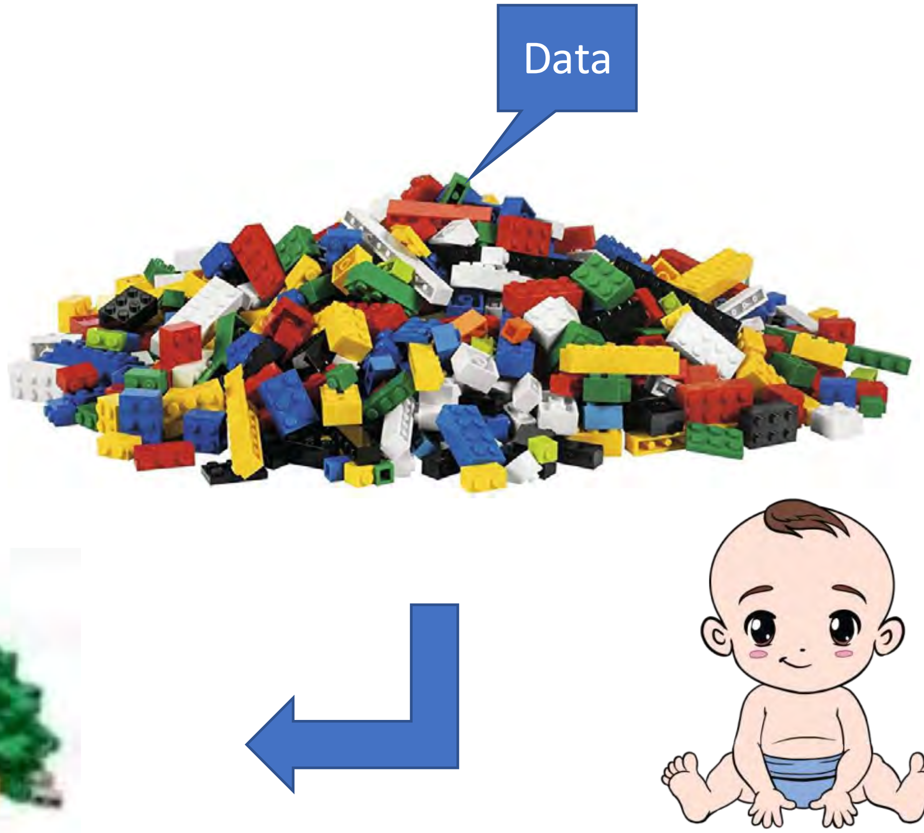
- **Classification** example: Image recognition
- **Task**: given an image, predict the class label
  - Training data: millions of labeled images
  - Target: the image class

- **Regression** example: Car price prediction
- Task: given a car, predict its price
  - Training data: many examples of cars, including their features (mileage, age, brand, etc.) and their labels (price)
  - Target: price of car

Unsupervised Learning

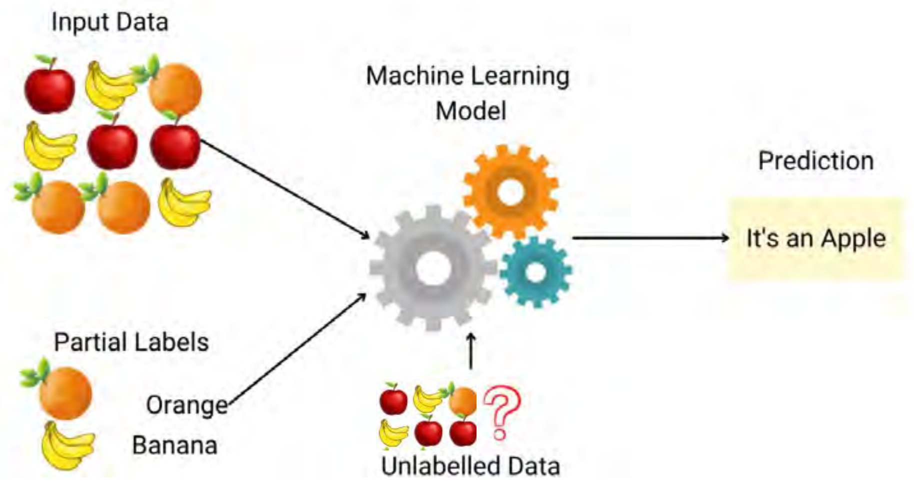
- Discover hidden groups (patterns) within the data
- NO pre-existing knowledge

❑ Example
  - Task: detect groups (patterns) of similar objects
  - Input: a lot of complex and unlabeled data



Semi-supervised Learning

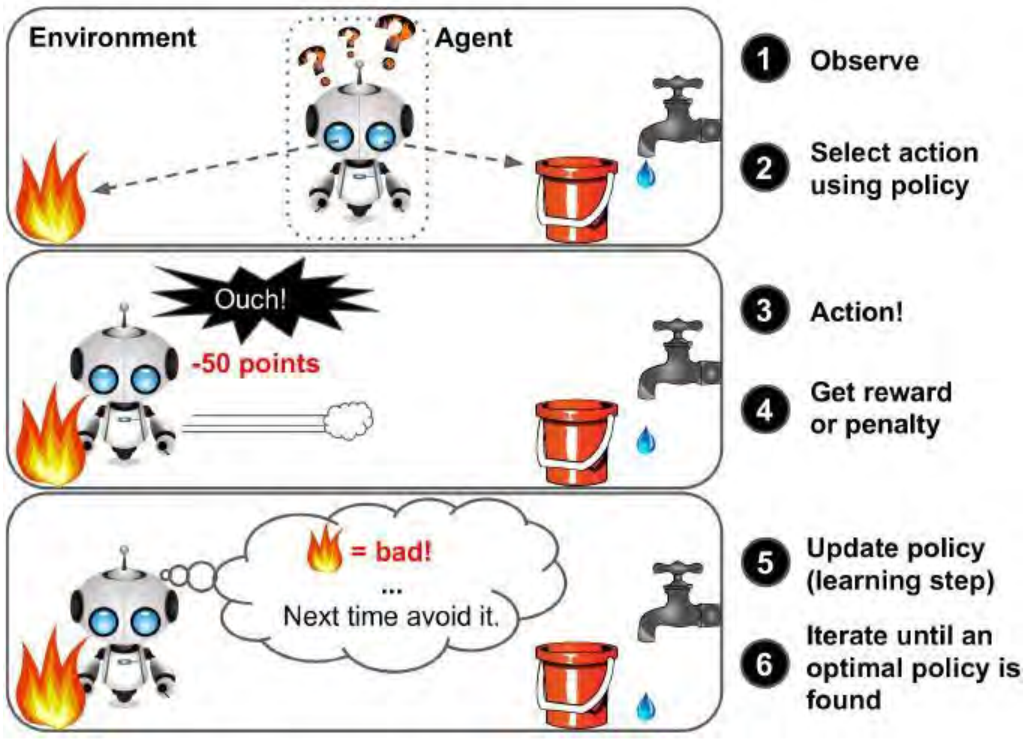
- Semi-supervised learning is a machine learning paradigm that combines labeled and unlabeled data to improve model performance.
- Key Challenge – Limited availability of labeled data in real-world applications.
- How Semi-Supervised Learning Works
  - Seed Labeled Data: A small subset of the data is labeled manually.
  - Propagation: The model leverages these labels to learn patterns in the unlabeled data.
  - Iterative Improvement: The process iterates, refining predictions and incorporating more unlabeled data.



Source: https://medium.com/@gayatri_sharma/a-gentle-introduction-to-semi-supervised-learning-7afa5539beea

Reinforcement Learning

- The learning system (agent) observes the environment, selects, performs actions and receives a reward signal, tries to learn a strategy (policy) to maximize the positive reward
- Example – Robotics
  - Task: detect groups of similar objects
  - Input: a lot of complex and unlabeled data


Environment
Agent
1 Observe
2 Select action using policy
Ouch!
-50 points
3 Action!
4 Get reward or penalty
= bad!
...
Next time avoid it.
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Source: https://spectrum.ieee.org/irobot-roomba-combo-j7-vacuum

Supervised Learning - **Regression**

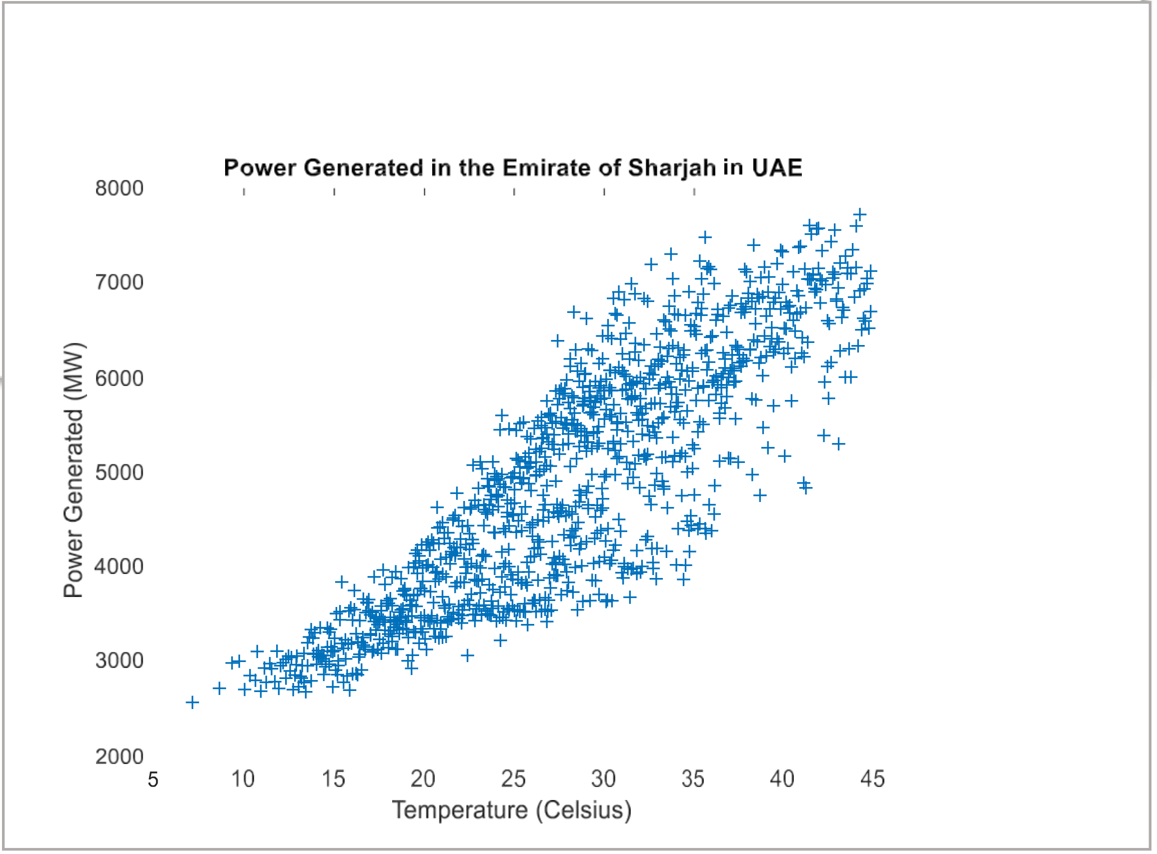
1.1: Linear Regression: Introduction

Regression

- A statistical **tool** for predicting **the continuous-valued** response variable
  - Features/Attributes
  - "correct/right" answers/observations are given
  - Model how our observations that are associated with the features change as we change the values of the features
- Predict more correct answers (the goal)
- Can be used to analyze the importance of the features themselves (feature engineering)
- **Feature engineering** is a critical step in the data preprocessing phase of machine learning and data science. It involves choosing the most **informative variables** for use in model construction.

Example

- Example:
  - Power Demand Prediction
  - One way of predicting/estimating the power demand is to fit a straight line to this data
  - Or we may find a better fit (quadratic, exponential, etc.)
- Regression Problem - Output of the algorithm continuous valued number (power demand)



Temperature (x)	Power Demand (y)
10	2300
20	3200
..	..

Terminology:
- x: feature or predictor
- y: response

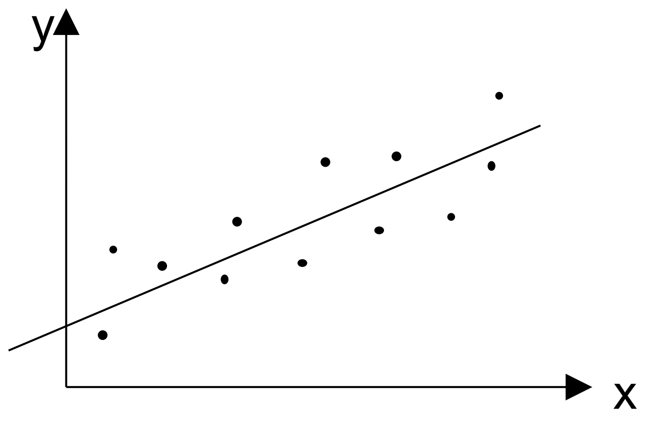
1.2: Simple Linear Regression

Regression Model

- We need to understand/examine the effects that some variables exert (or appear to exert) on other variables
- How can we find a "simple" <u>functional relationship between these variables</u>?
- Usually, we may wish to **approximate** this functional relationship by some simple mathematical function, such as a polynomial
    - ❍ By **examining** that function, we may be able to learn more about the underlying true relationship and separate/joint effects produced in certain important variables.
    - ❍ While this equation may be physically meaningless, it may be extremely valuable for predicting the values of some variables from knowledge of other variables.

Linear Regression Model

- Notations:
  - m: denotes the number of training examples/ number of observations
  - x: denotes “input” variable (feature) – Temperature
    - Predictor variable → A change made in this variable affects the values of the output variable(s).
  - y: denotes output/target variable/response variable that we are going to predict
  - One pair (x,y) → one training example / one observation
  - $(x^{(i)}, y^{(i)})$: denotes the i^{th} training example / i^{th} observation
  - n: denotes the number of features



$$\boxed{f(x) = \theta_1 x + \theta_0}$$

(or)

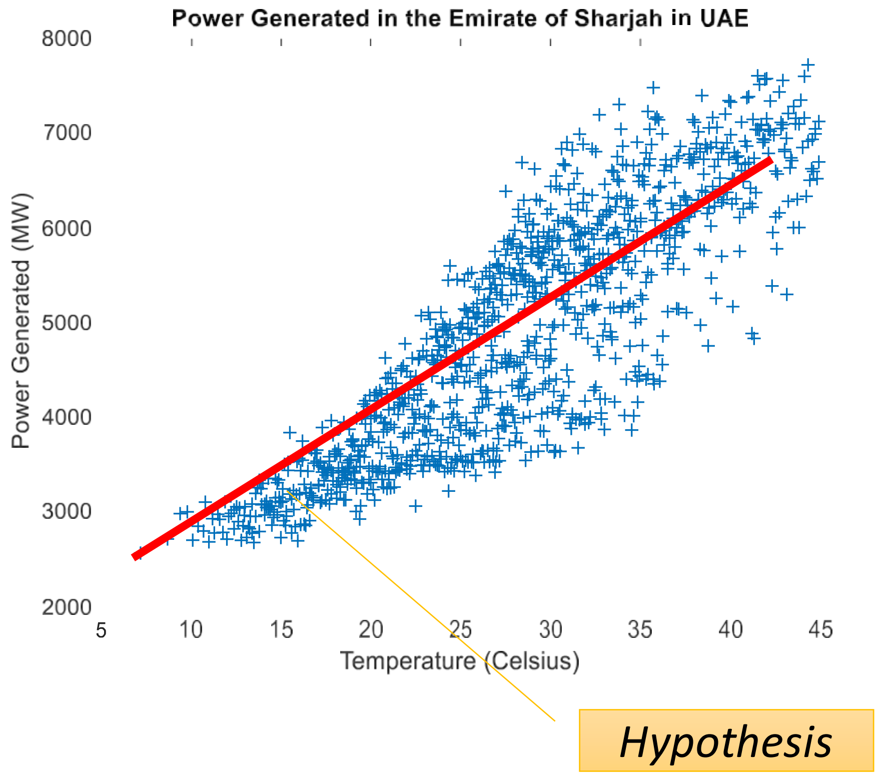
$$\boxed{\underbrace{y}_{\text{dependent variable}} = \underbrace{\theta_1}_{\text{slope}}\underbrace{x}_{\text{Independent variable}} + \underbrace{\theta_0}_{\text{y -intercept}}}$$

Linear Regression Model (cont.)

- Example – Predict Power Demand ("Short-Term forecasting of electricity Load")
- Data Set (Training Set): given labeled data
  - Temperature (Celsius) -> feature (x)
  - Power Demand (MWh) -> label (y)

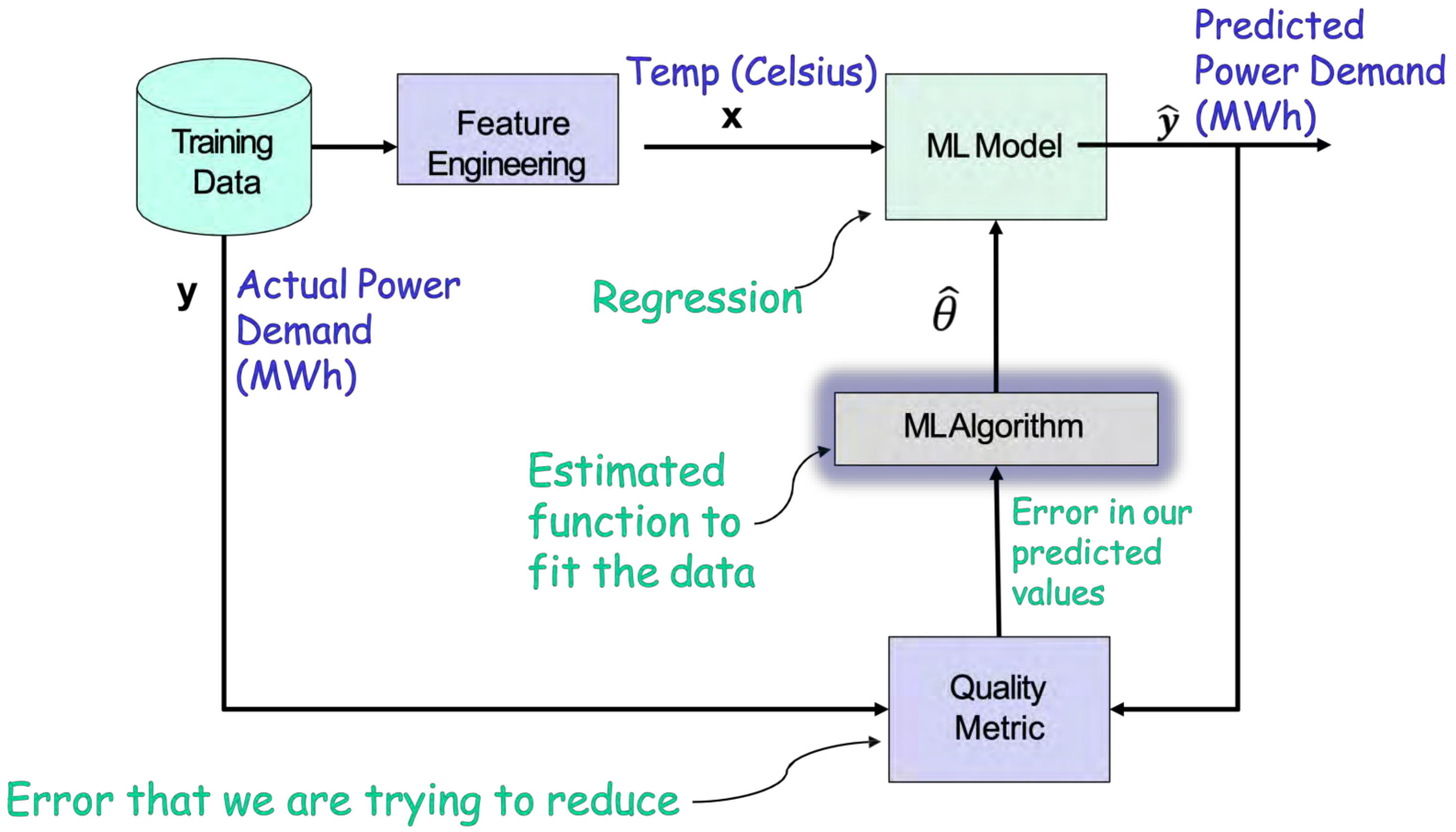
Temp(°C)	LOAD
-24.2	18128
-10.9	19437
-5	14070
0	13955
3	14974
..	..
..	..
..	..

Sample Data



Algorithms for fitting the model

- Using the prediction of power demand from temperature, we have the following system:



Linear Regression Model (cont.)

- **Training Data**: We are going to assume that we have enough **training data**
  - We are going to discuss the concepts of training data, test data, validation sets, and other ideas about fitting our models and assessing our fits.
- **Feature Extraction** (part of the feature engineering):
  - We need to figure out what input we need to use for our regression model. Is temperature enough? Are there other relevant attributes that we can consider? Dates? Humidity? What day of the week?
- **Machine Learning Model**: In this lecture, we are using regression.

  $\hat{y}$ represents our predicted power demand. We predict $\hat{y}$ based on some estimated function (line or curve; fit to our training data) which is represented by the equation $f(x)$
- **Quality Metric**: y is the actual/observed power demand. $\hat{y}$ is going to be compared with y to see how well the model is doing.

Simple Linear Regression

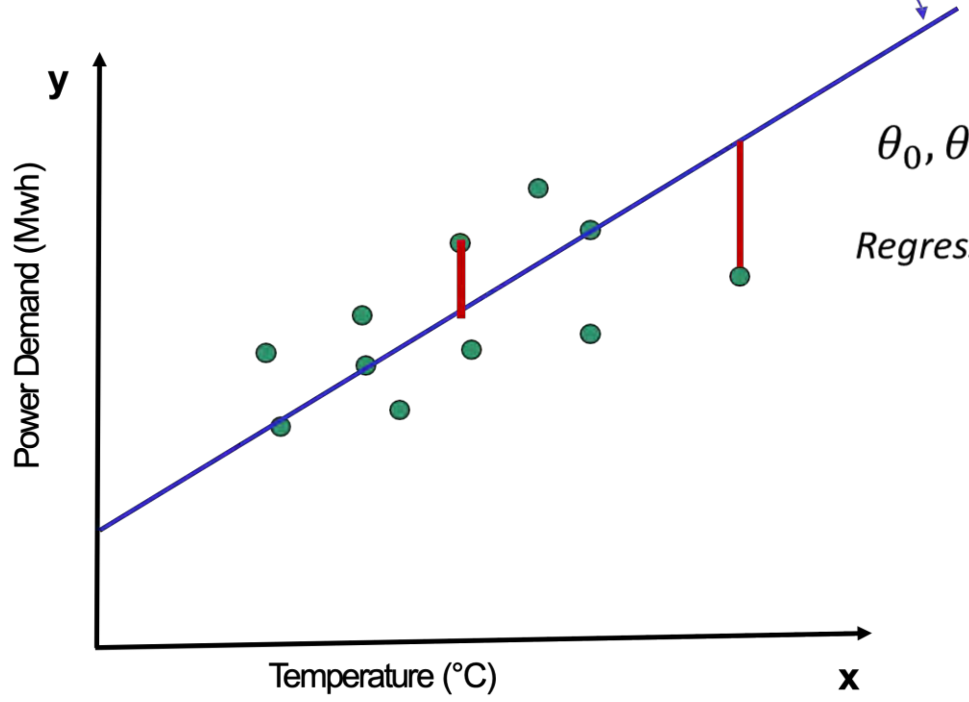
- Model: represents the expected relationship between x and y. Given the relationship/model, **we can predict y if x is given**
  - The model is not 100% accurate. There is an error in the model
  - I may have the same temperatures on different days but with different power demands

Expected relationship between *x* and *y*

$$f(x) = \theta_0 + \theta_1 x$$

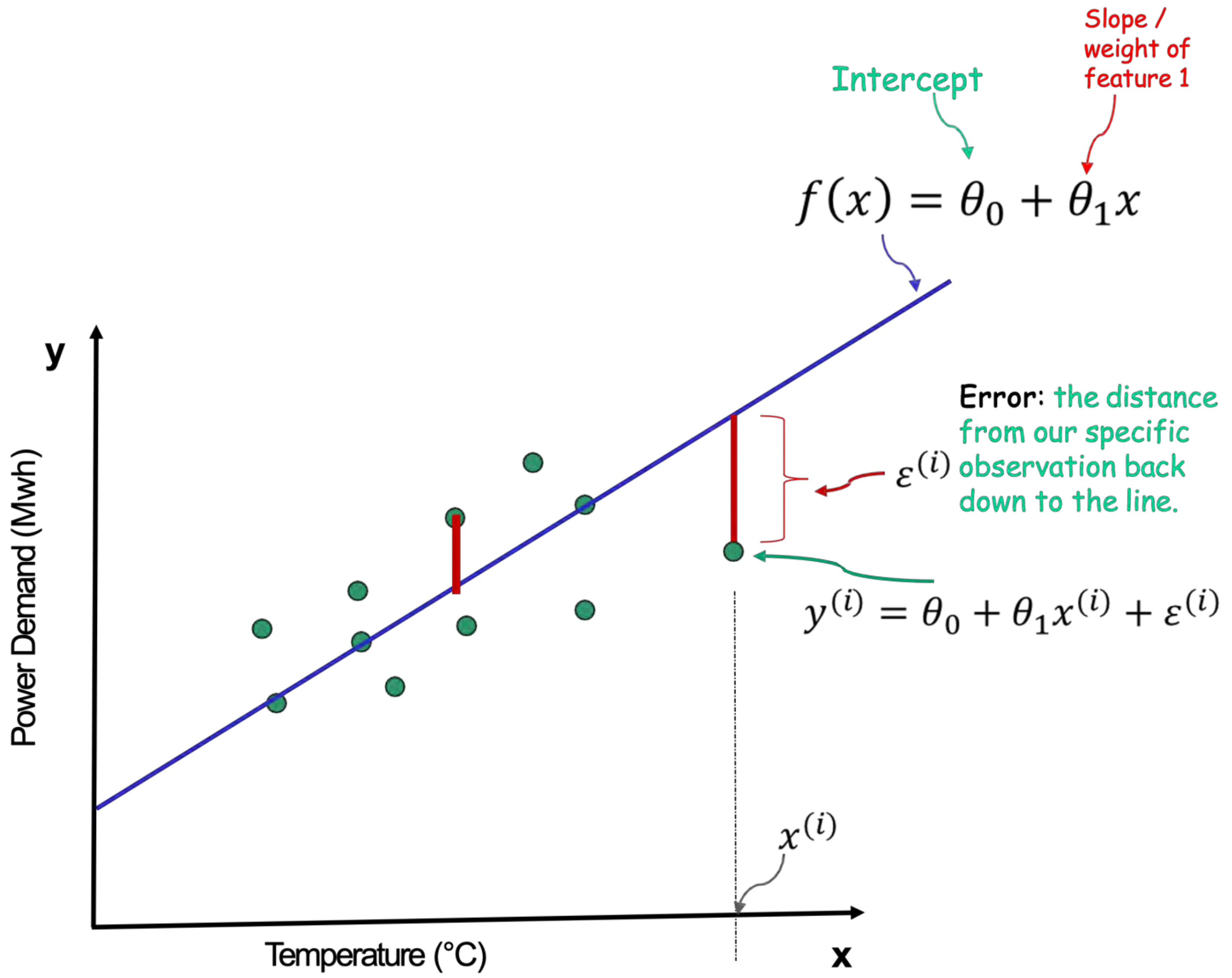
$\theta_0, \theta_1 : Parameters$

Regression coefficients



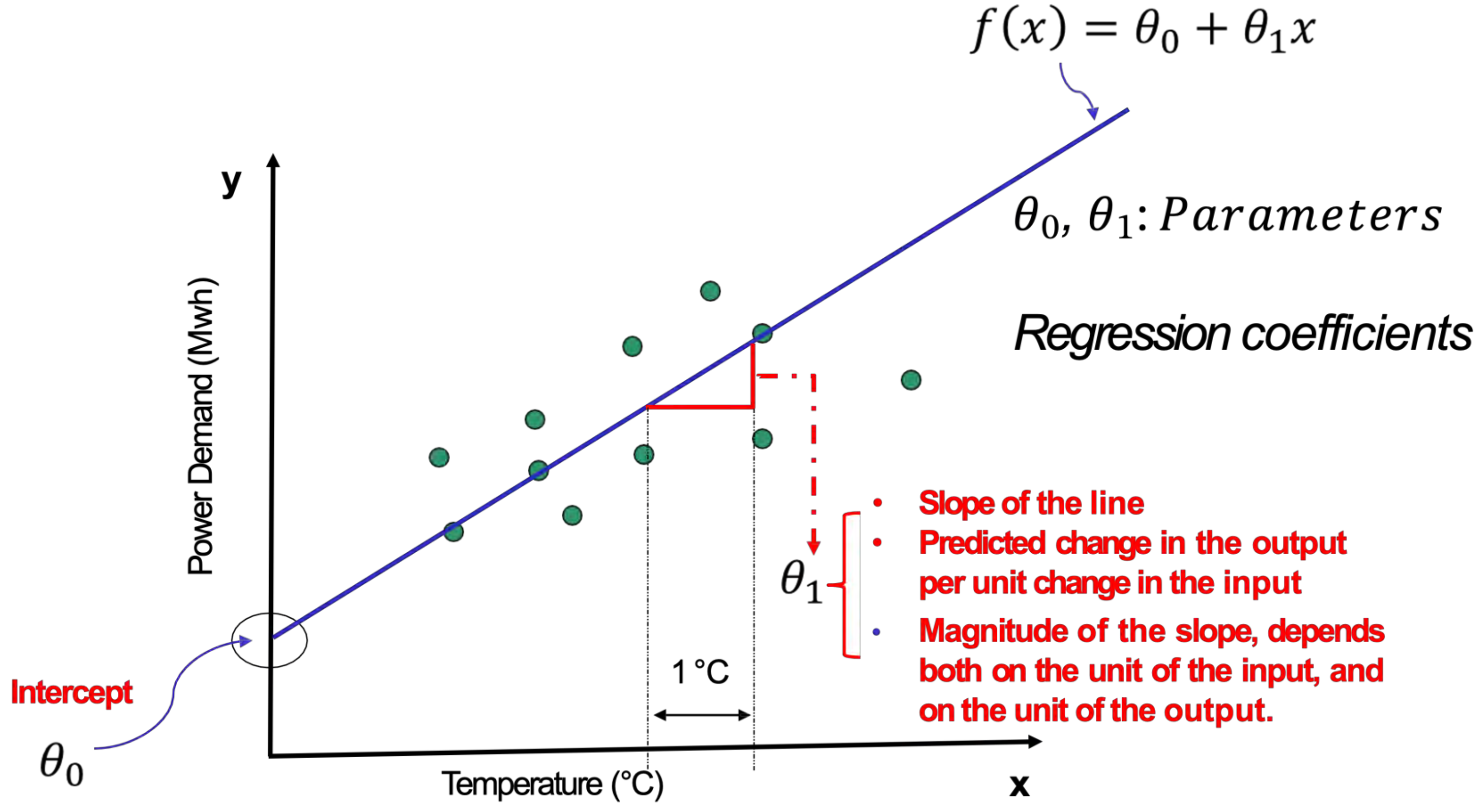
Simple Linear Regression (cont.)

We try to fit a line through the data points



Intercept
Slope / weight of feature 1
$f(x) = \theta_0 + \theta_1 x$
y
Power Demand (Mwh)
Error: the distance from our specific observation back down to the line.
$\varepsilon^{(i)}$
$y^{(i)} = \theta_0 + \theta_1 x^{(i)} + \varepsilon^{(i)}$
$x^{(i)}$
Temperature (°C)
x

Simple Linear Regression (cont.)

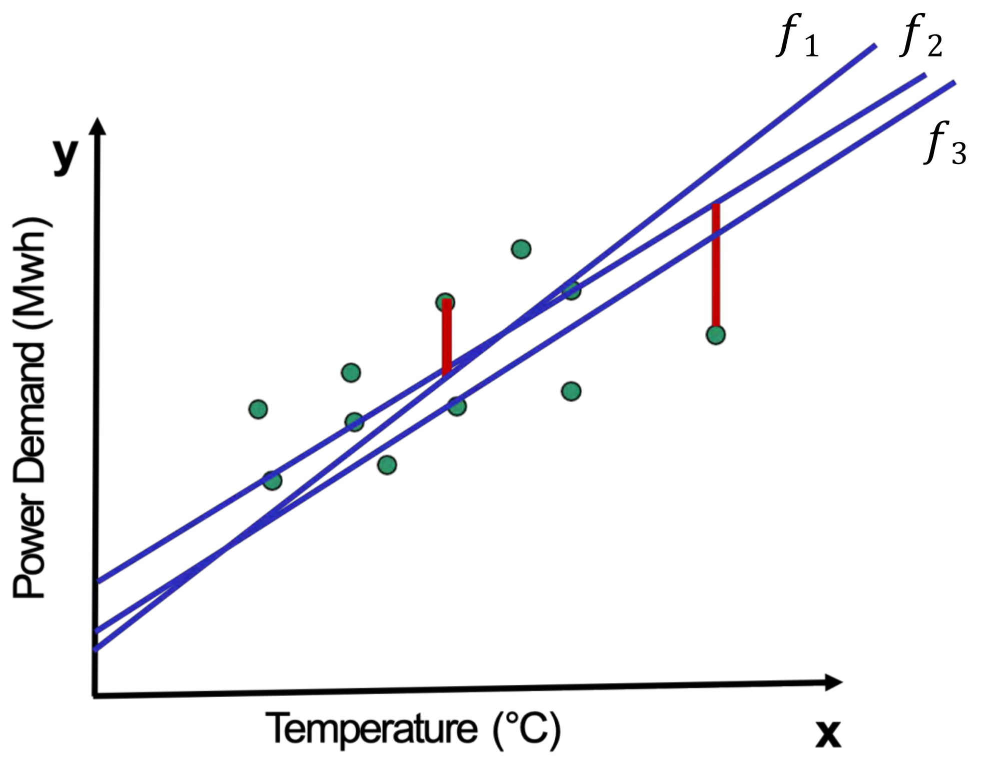


$f(x) = \theta_0 + \theta_1 x$
θ_0, θ_1: *Parameters*
Regression coefficients
y
Power Demand (Mwh)
Temperature (°C)
x
1 °C
θ_1
Intercept
θ_0

- Slope of the line
- Predicted change in the output per unit change in the input
- Magnitude of the slope, depends both on the unit of the input, and on the unit of the output.

Find the “best” line: How to measure the quality of the fit?

- Which line is the right line or the best fit for a given data set?
- Each one of these is given by a different set of parameters.
- We need to define a cost for every given line – Residual Sum of Squares (RSS)
- The cost of each line will be based on RSS
  - We take our fitted line and look at each of our observations (historical data/training data).
  - We look at how far is that observation from what the model would predict (the point on the line)
  - We look at all these distances
  - We are going to use the sum of the square of all these distances (residual)
  - The residual is the difference between your prediction and your actual observation.

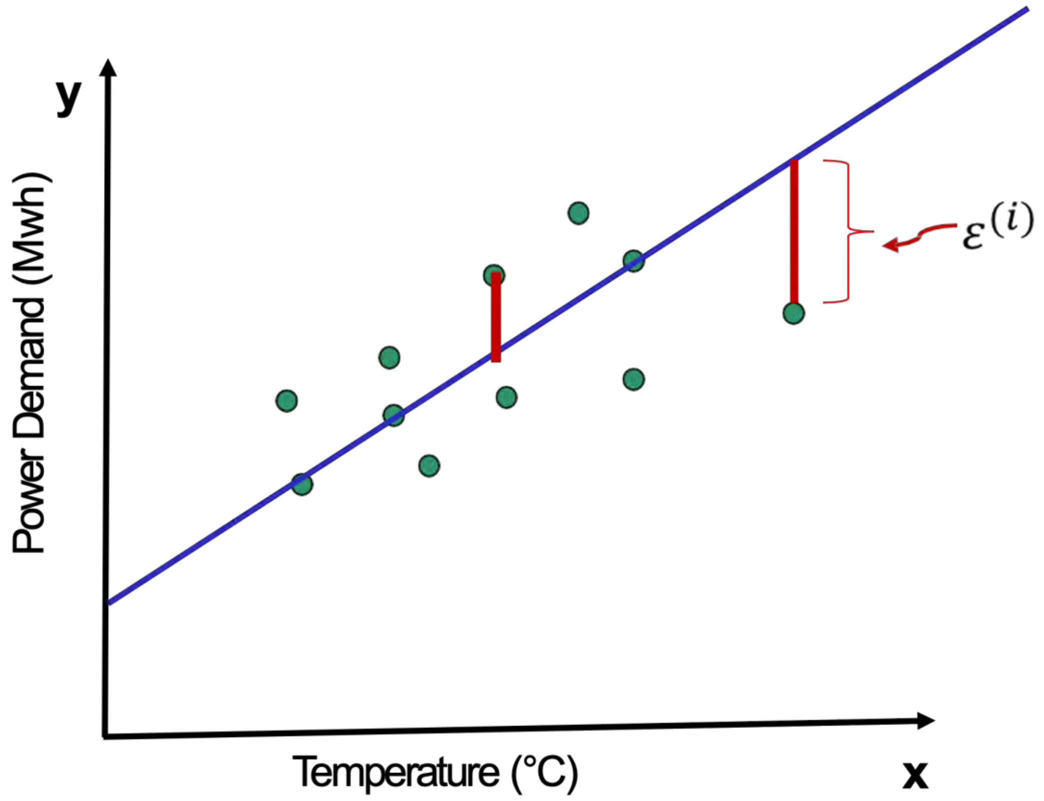


Residual Sum of Squares (RSS)

- We are going to calculate the average of the errors for all data points. Error is the difference between the estimated output and the observed output. RSS is going to be a function of θ_0 and θ_1

$$RSS(\theta_0, \theta_1) = \left[y^{(1)} - (\theta_0 + \theta_1 x^{(1)})\right]^2 + \left[y^{(2)} - (\theta_0 + \theta_1 x^{(2)})\right]^2 + \dots\dots\dots\dots \left[y^{(m)} - (\theta_0 + \theta_1 x^{(m)})\right]^2$$

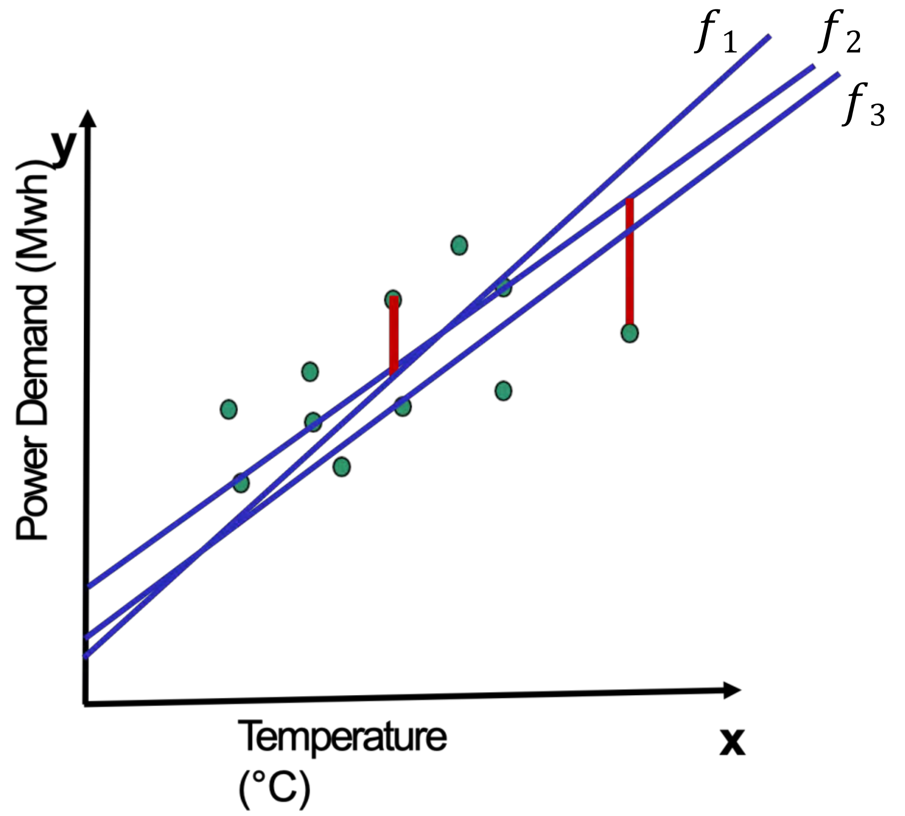
$$RSS(\theta_0, \theta_1) = \sum_{j=1}^{m} \left[y^{(j)} - (\theta_0 + \theta_1 x^{(j)})\right]^2$$



Find the "Best" Line

- We will try to find the best line according to the RSS.
- Ideally, we must look at all possible lines and choose the one that minimizes the residual sum of squares (RSS).

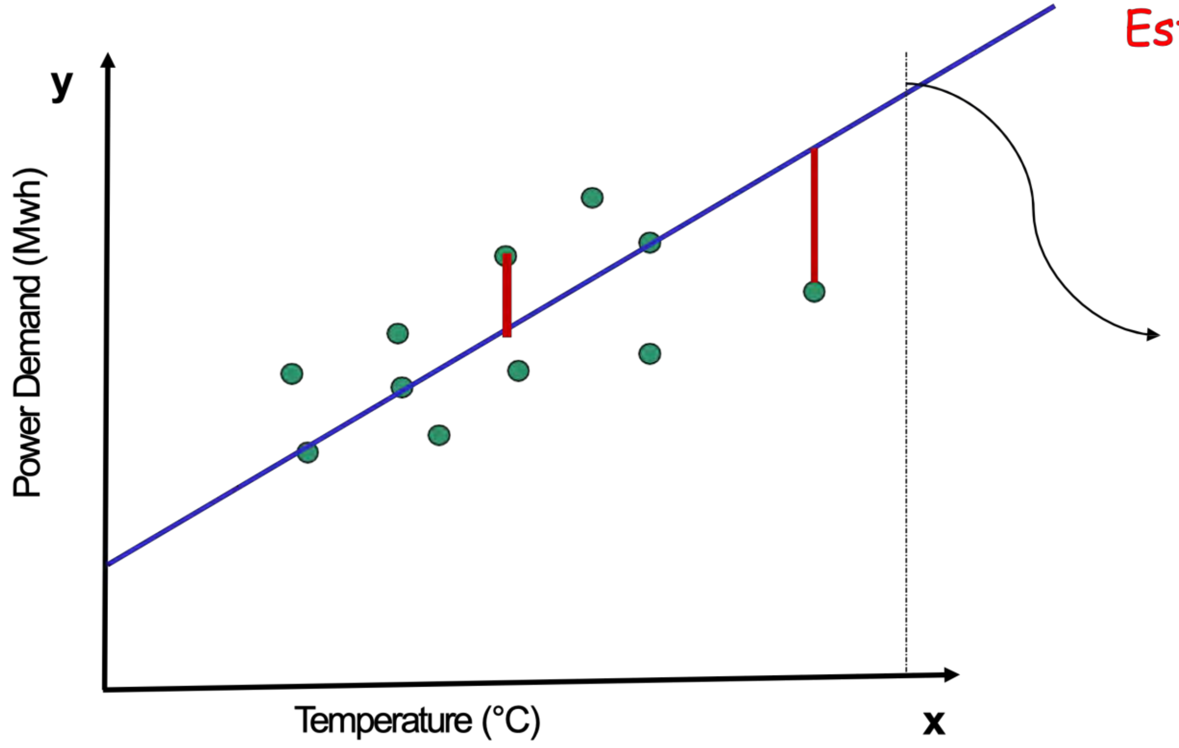
We are going denote the resulting θ_0 and θ_1 as $\hat{\theta}_0$ and $\hat{\theta}_1$
There exists nice and efficient algorithms for computing this as $\hat{\theta}_0$ and $\hat{\theta}_1$.
The algorithm searches over all possible θs.



Estimated Parameters

$$\hat{y} = \hat{f}(x) = \hat{\theta}_0 + \hat{\theta}_1 x$$

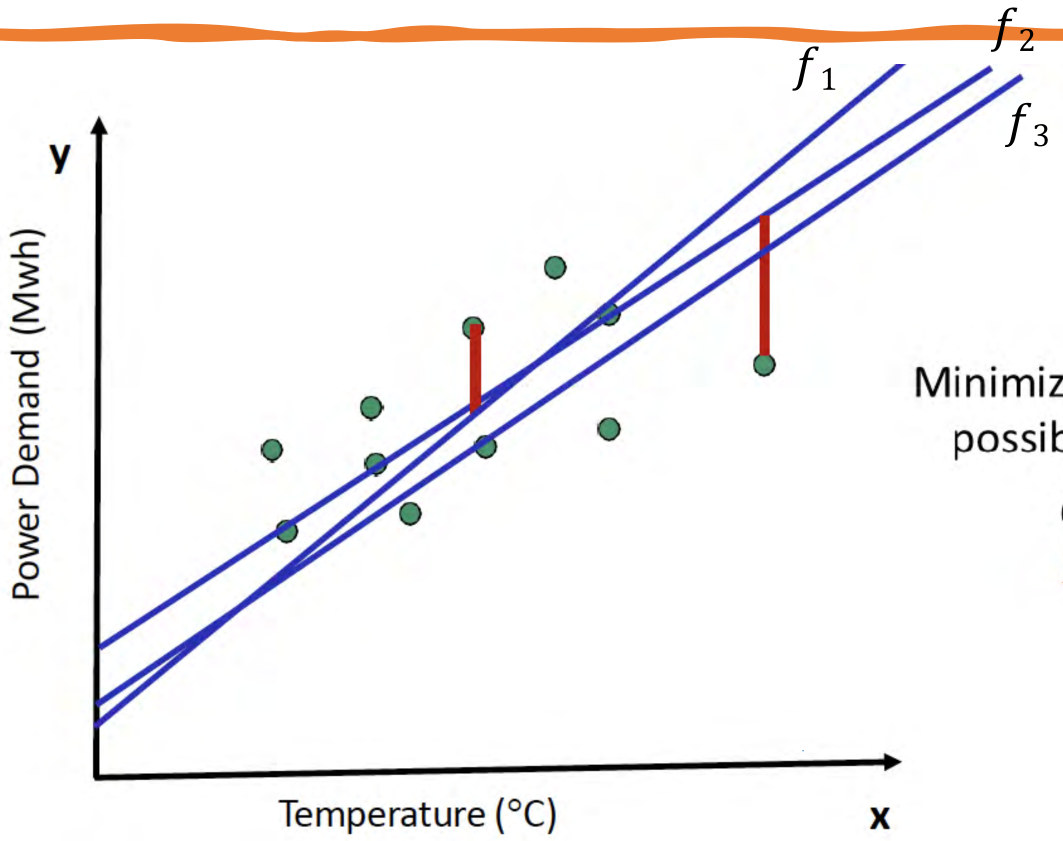
Estimated parameters



$$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 x$$

1.3: Find the Best Model: Optimization Techniques

Using Simple Linear Regression



Minimize cost over all possible values of

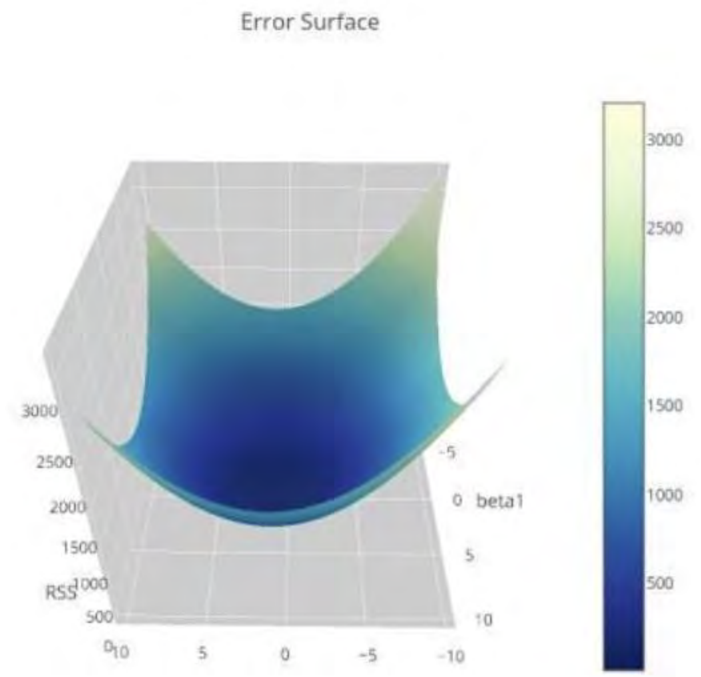
θ_0, θ_1

How?

Residual Sum of Squares (RSS)

$$RSS(\theta_0, \theta_1) = \sum_{j=1}^{m} \left[y^{(j)} - (\theta_0 + \theta_1 x^{(j)})\right]^2$$

Minimizing the Cost



Optimization Problem

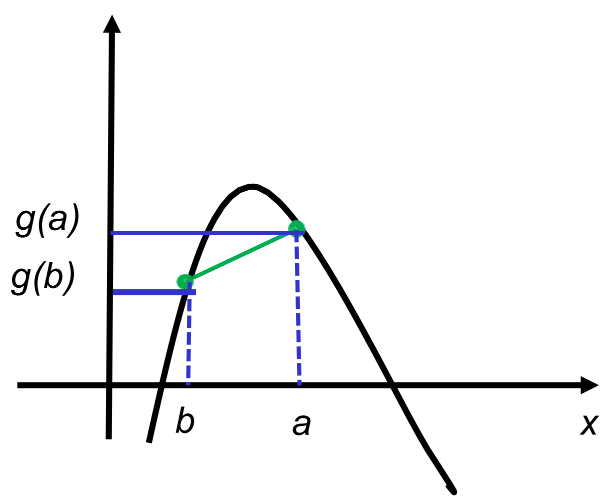
Minimize function over all possible θ_0 and θ_1

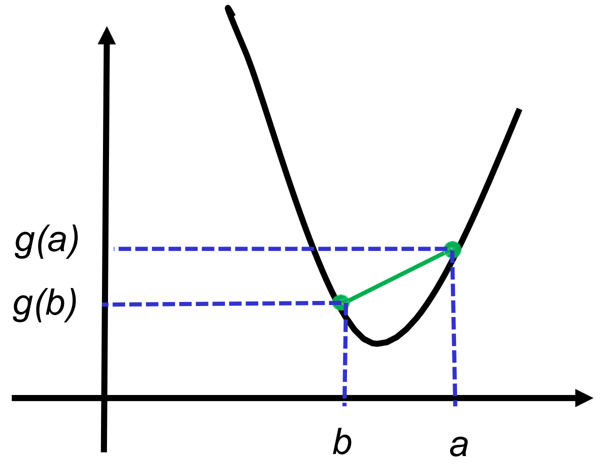
$$\min_{\theta_0,\theta_1} \sum_{j=1}^{m} \left[y^{(j)} - \left(\theta_0 + \theta_1 x^{(j)}\right)\right]^2$$

$RSS(\theta_0, \theta_1)$ **is a function of 2 variables** $\boldsymbol{H(\theta_0, \theta_1)}$

Optimization Problem – Quick Review

Optimization: Review



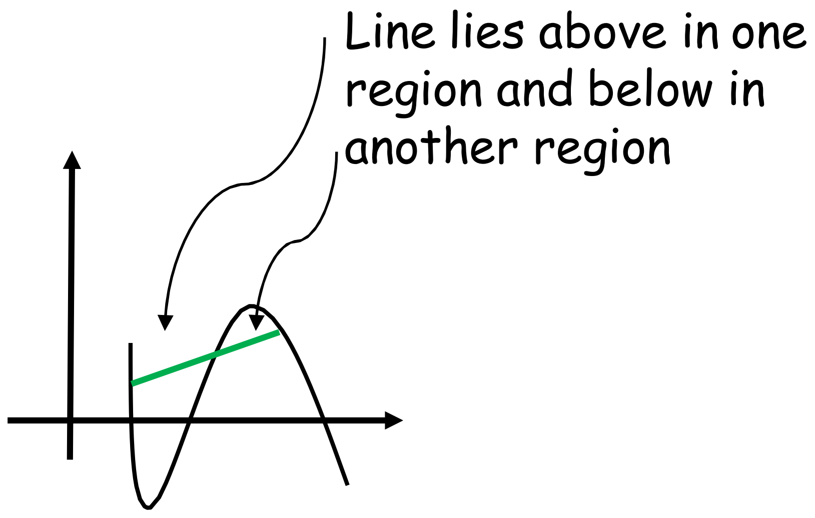


- Concave: For any two values of x. Let's call them ‘a’ and ‘b’. We look at the value of the function at a and b, g(a) and g(b). We draw a line between g(a) and g(b) (the green line above).
- This line lies below g(x) everywhere.
- It looks like you are in a cave (arch)

- Convex: has the opposite property. This line lies above g(x) everywhere.

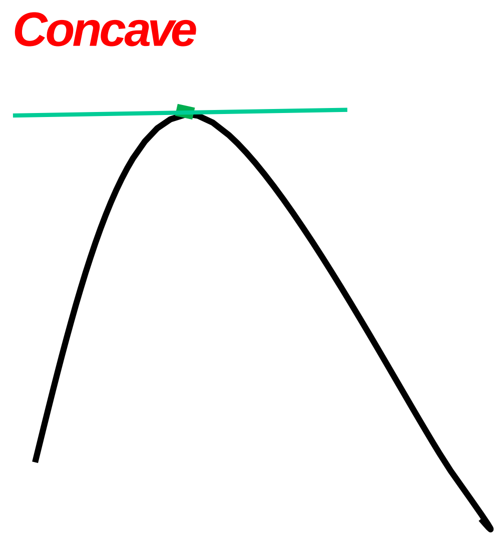
Optimization: Review (cont.)

- In the following case, the function does not satisfy either the concave or convex criteria.



Neither

Finding the max or min **Analytically**

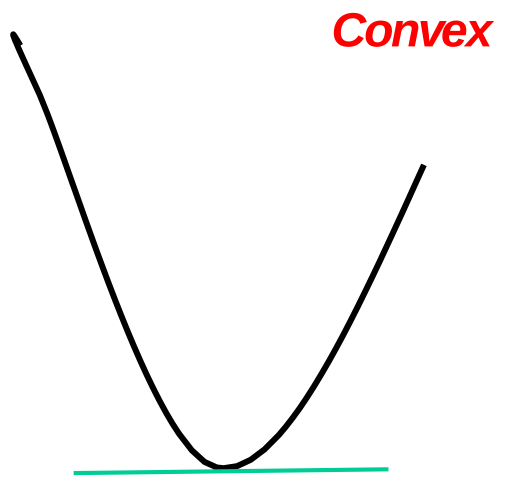


- The maximum of the concave function

$$\max_{x} g(x)$$

- The point where the derivative is 0.
- There is no rate of change of this function at this point

Finding the max or min Analytically (cont.)



- The maximum of the convex function

$$\min_{x} g(x)$$

- The point where the derivative is 0.
- There is no rate of change of this function at this point

Finding the max or min Analytically (cont.)

- For the **concave** and **convex** functions, there is only **one place** where the derivative is equal to zero.
- In contrast, for functions that are **neither**, we see that there are **multiple locations** where we have the derivative equaling 0.
- Thus, there are multiple solutions to the derivative, which is equal to 0.

Finding the max or min Analytically (cont.)

- Assume we have the following function:

$$g(\theta) = 1 - (\theta - 5)^2$$

- Analytically, to find the max or min we must first take the derivative:

$$\frac{dg(\theta)}{d\theta} = \frac{d(1)}{d\theta} - \frac{d(\theta - 5)^2}{d\theta}$$

$$\frac{dg(\theta)}{d\theta} = \overset{0}{\cancel{\frac{d(1)}{d\theta}}} - 2(\theta - 5) * 1$$

Finding the max or min Analytically (cont.)

- Next, to determine where the rate of change is zero, we must set the first derivative to 0 and solve for θ

$$0 = -2(\theta - 5) * 1$$

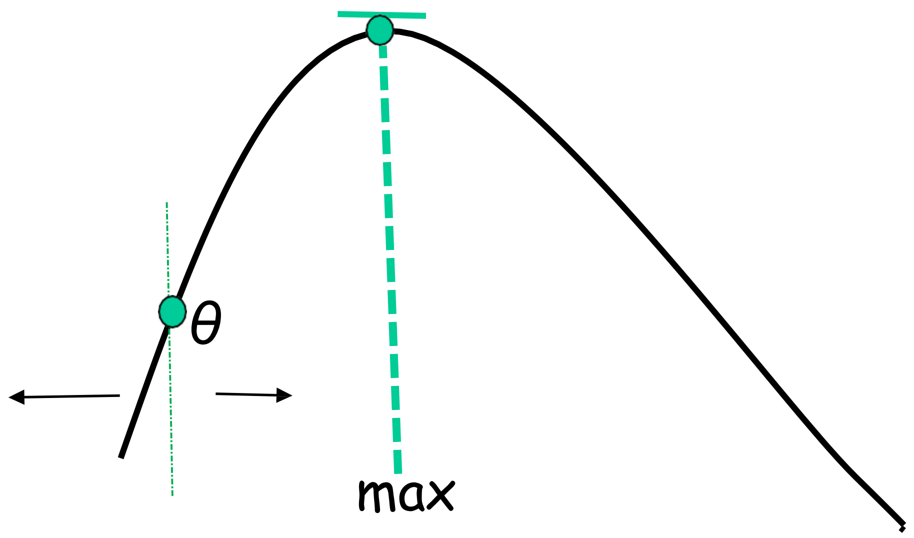
$$0 = \boxed{-2\theta} + 10$$

$$\frac{2\theta}{2} = \frac{10}{2}$$

$$\theta = 5$$

Hill-Climbing Algorithm

- The hill-climbing algorithm is an integrative algorithm where we start somewhere in this space of possible θ's, and then we keep changing θ, hoping to get closer and closer to the optimum.
- If we start at a certain value θ.
- The question is, should I increase θ, move θ to the right, or should I decrease θ and move θ to the left to get closer to the optimal?
- How do we know whether to move θ to the right or left and equivalently mean increase or decrease the value of θ?

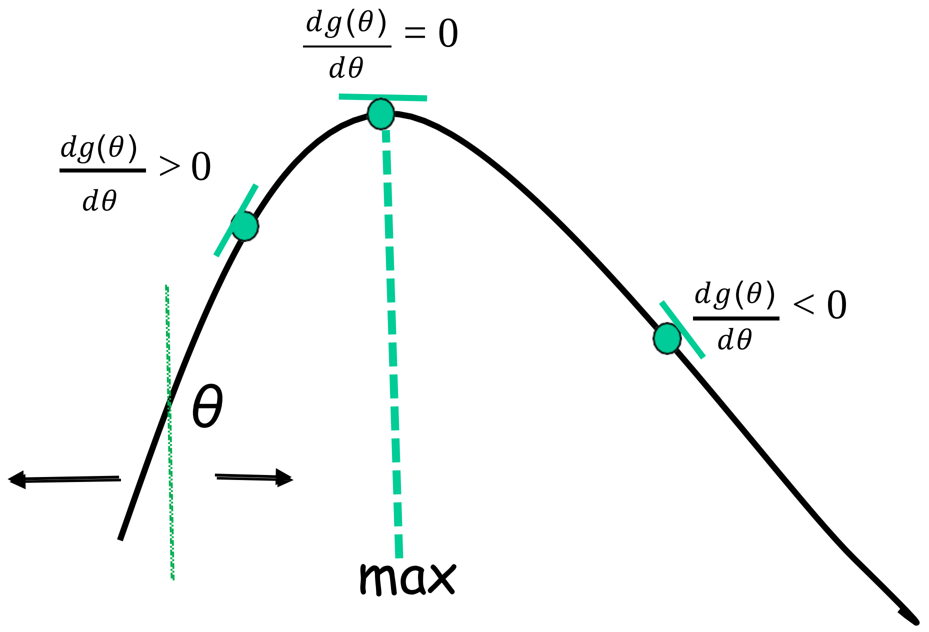


Finding the Max via Hill Climbing Algorithm

- I can look at the function at θ, and I can take the derivative, and if the derivative is positive (like it is in the example), ➔ I want to increase θ.
- But what if, on the other hand, I had started the algorithm over the other side of the optimum point? The derivative would be negative ➔ I would have to decrease θ
- Algorithm: **while** not converged

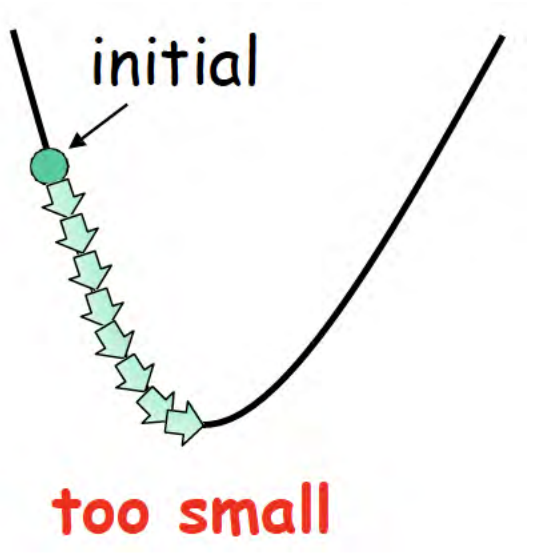
$$\theta_{iteration\ t+1} \leftarrow \theta_{iteration\ t} - \alpha \frac{dg(\theta)}{d\theta}\Big|_{\theta_{iteration\ t}}$$

- α is the step size (learning rate)

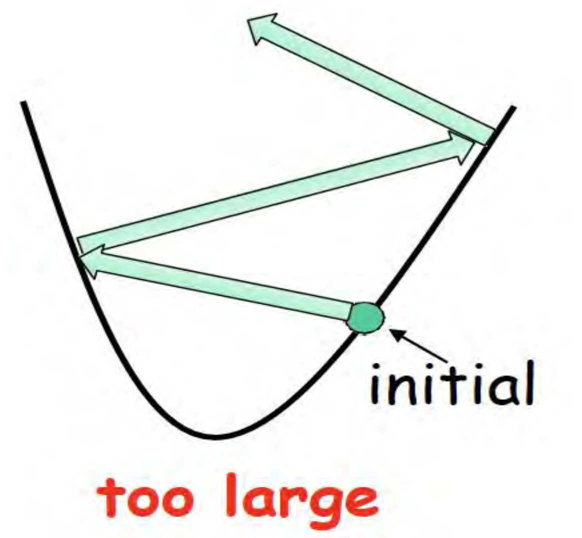

$\frac{dg(\theta)}{d\theta} = 0$
$\frac{dg(\theta)}{d\theta} > 0$
$\frac{dg(\theta)}{d\theta} < 0$
θ
max

Static Learning Rate

- A static learning rate is one that remains constant during all iterations.



When selecting a static learning rate, care must be taken to ensure that the learning rate is appropriate

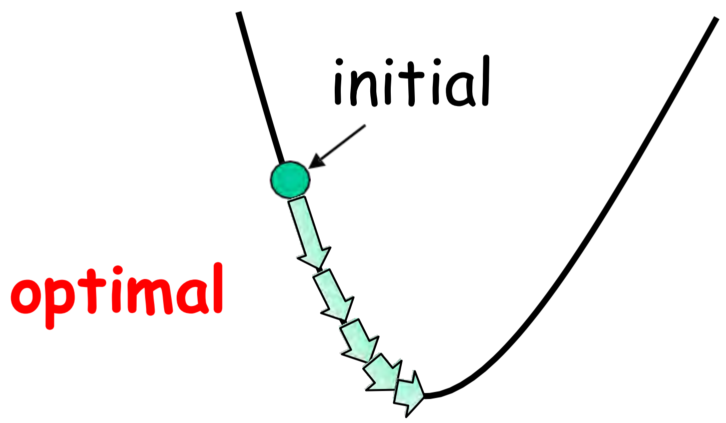


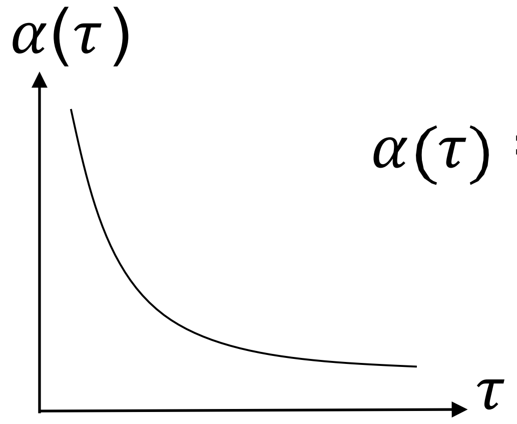
- A learning rate which is too small and requires many iterations before reaching the minimum

- A learning rate that is too large leads to severe updates and results in divergence

Choosing the Learning Rate (stepsize)

- **Decreasing learning rate** – A better practice when selecting a learning rate is the idea of a **dynamic** (decreasing) learning rate, which decreases over time. This is ideal because it allows the algorithm to swiftly identify the minimum point.
- The most basic decreasing learning rate schedule is:

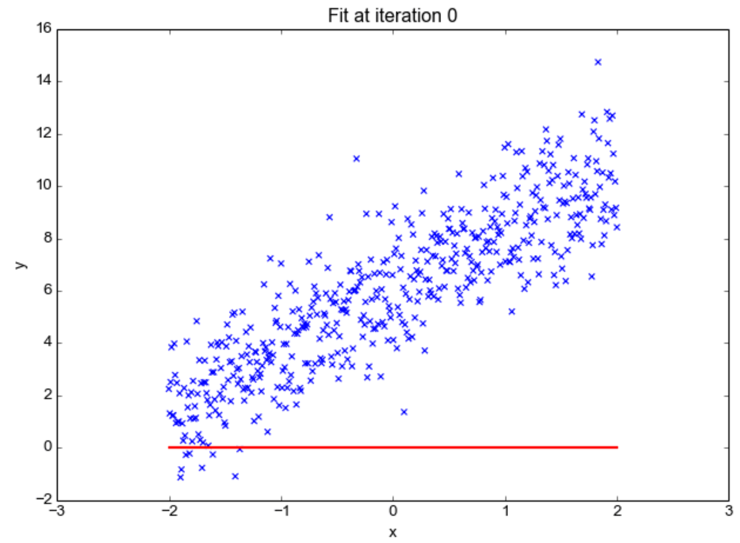




$$\alpha(\tau) = \frac{\alpha}{\tau}$$

initial learning rate

iteration number



Convergence Criteria

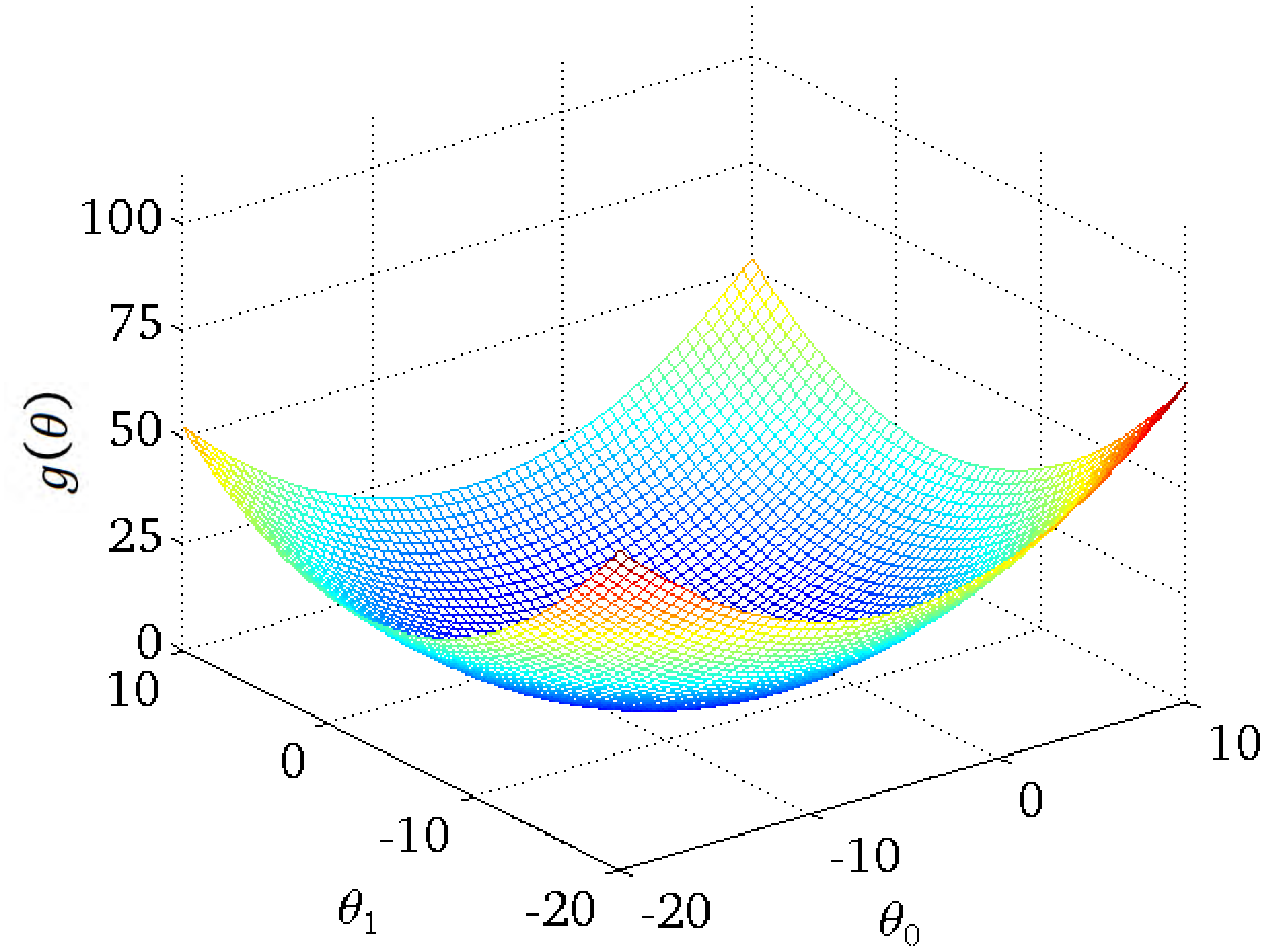
- For convex functions, the optimum occurs when

$$\frac{dg(\theta)}{d\theta} = 0$$

- However, in practice

$$\left|\frac{dg(\theta)}{d\theta}\right| < \varepsilon \xrightarrow{} \text{threshold}$$

- We set a threshold value that defines what is "close enough" to the optimal solution to be acceptable to our problem

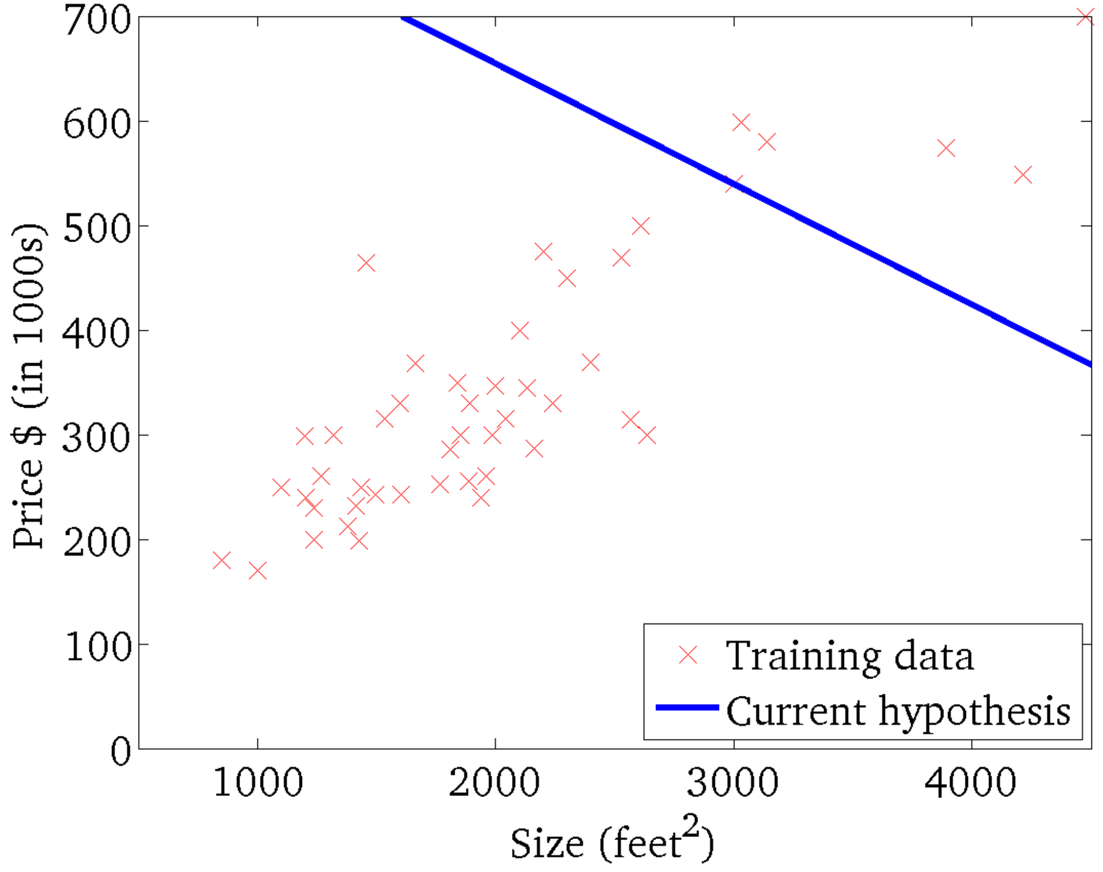

100
75
50
25
0
$g(\theta)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

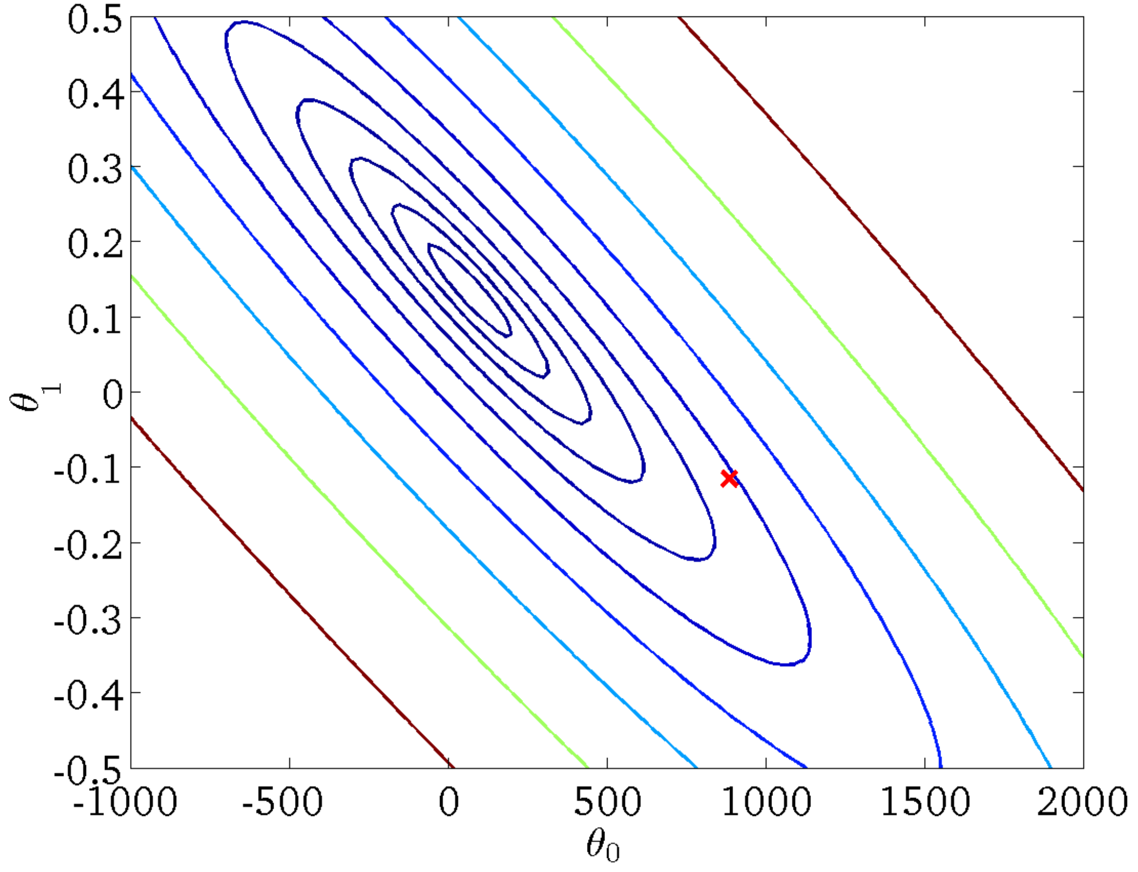
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

(function of the parameters θ_0, θ_1)



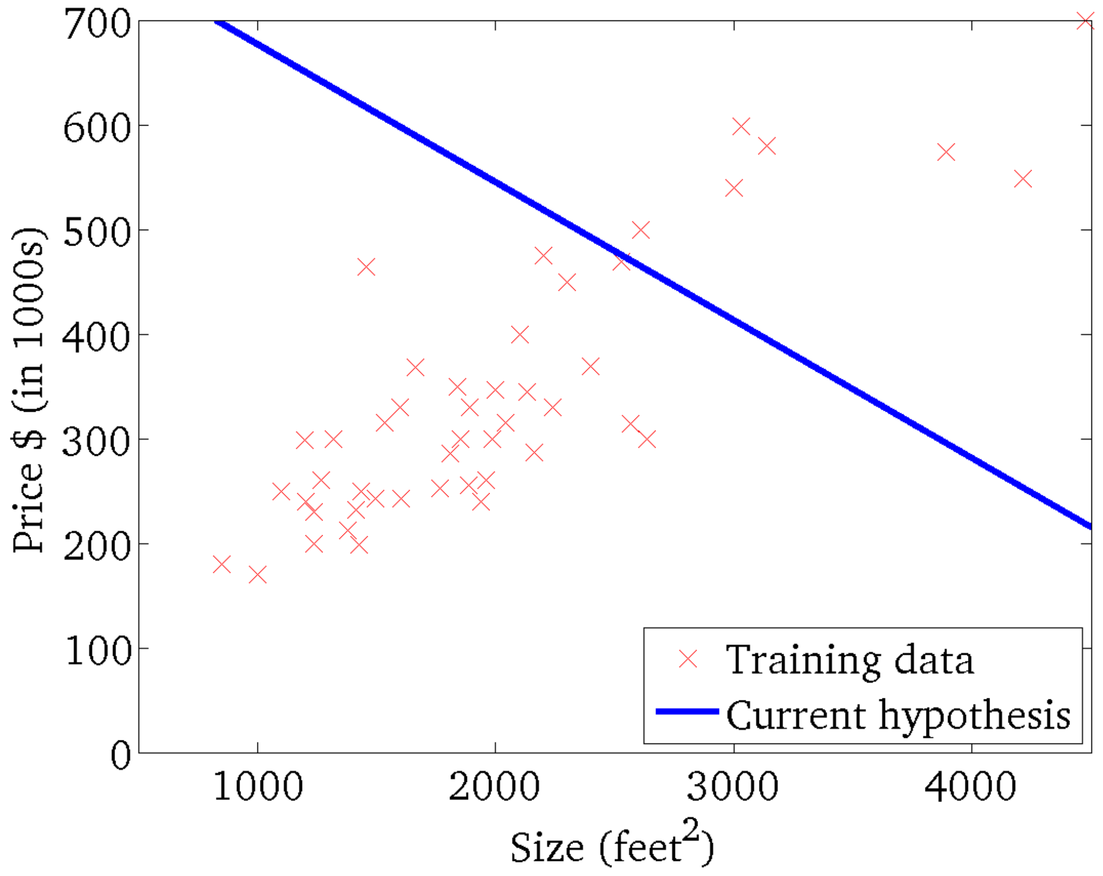


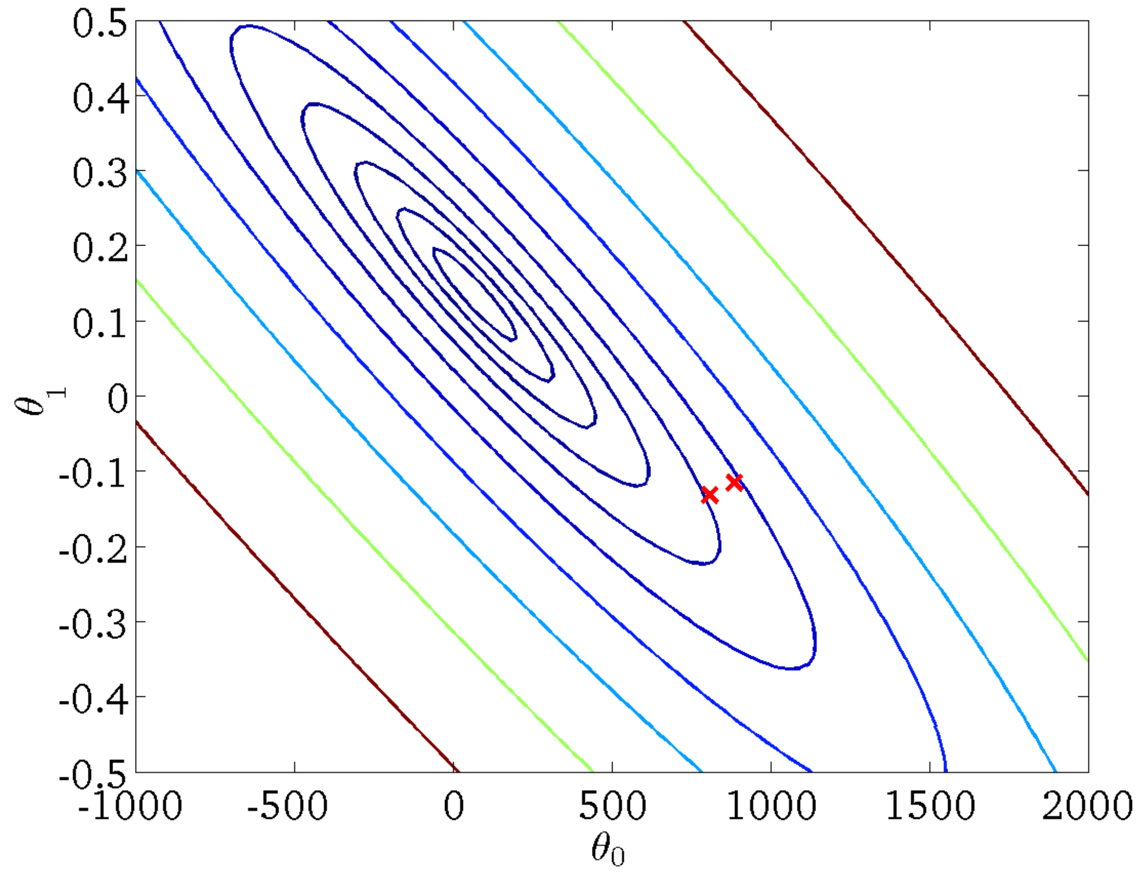
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

(function of the parameters θ_0, θ_1)



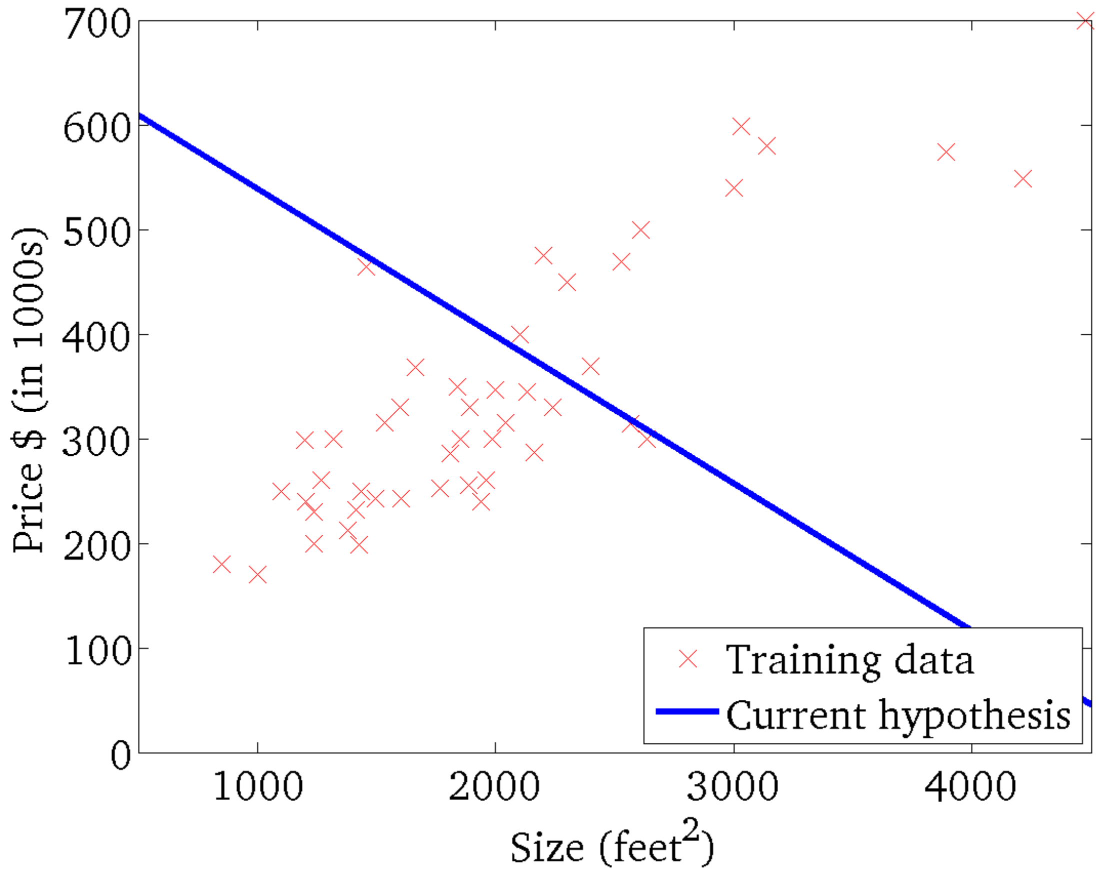


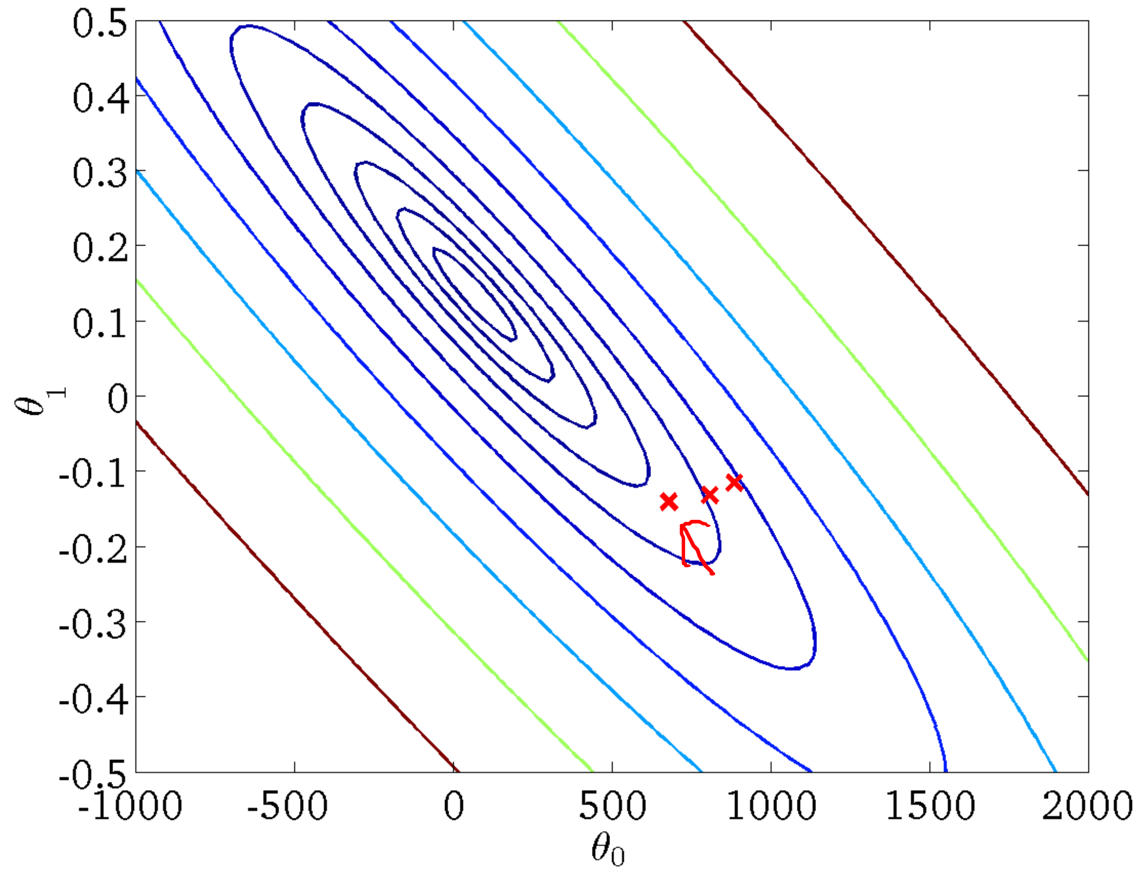
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

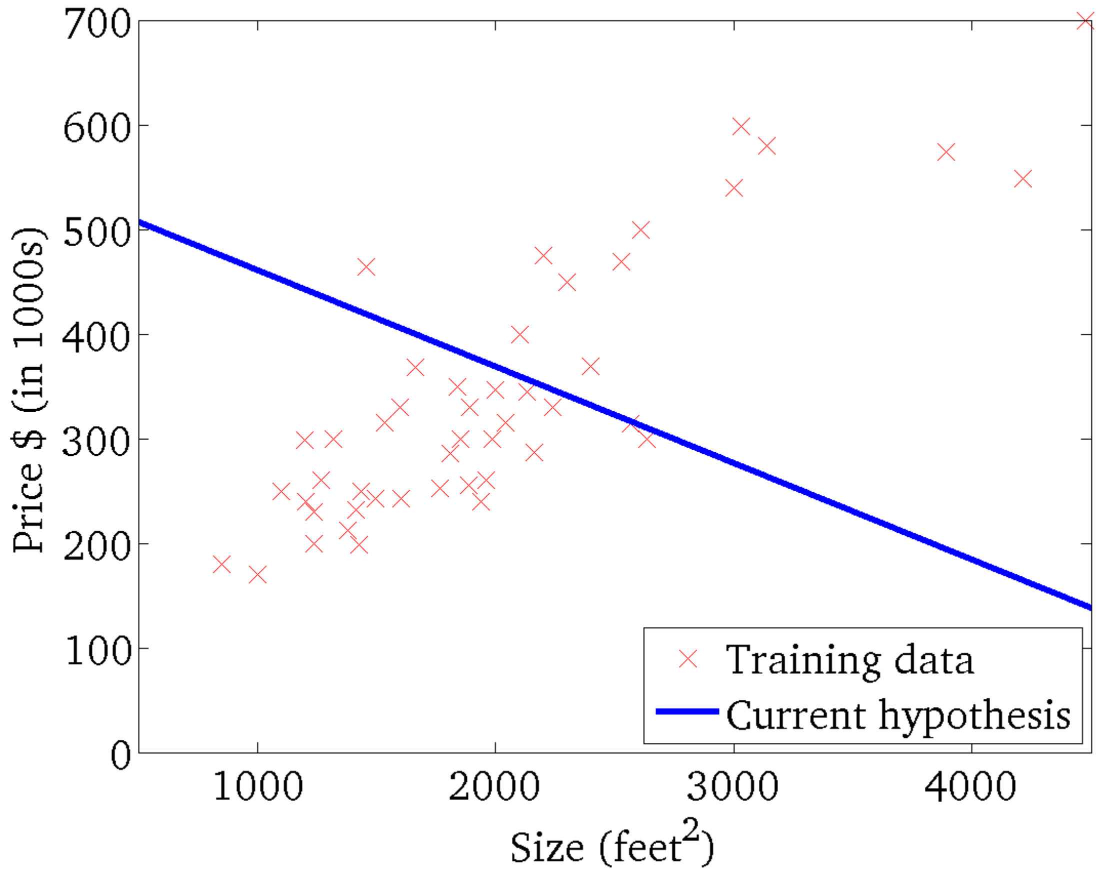
(function of the parameters θ_0, θ_1)





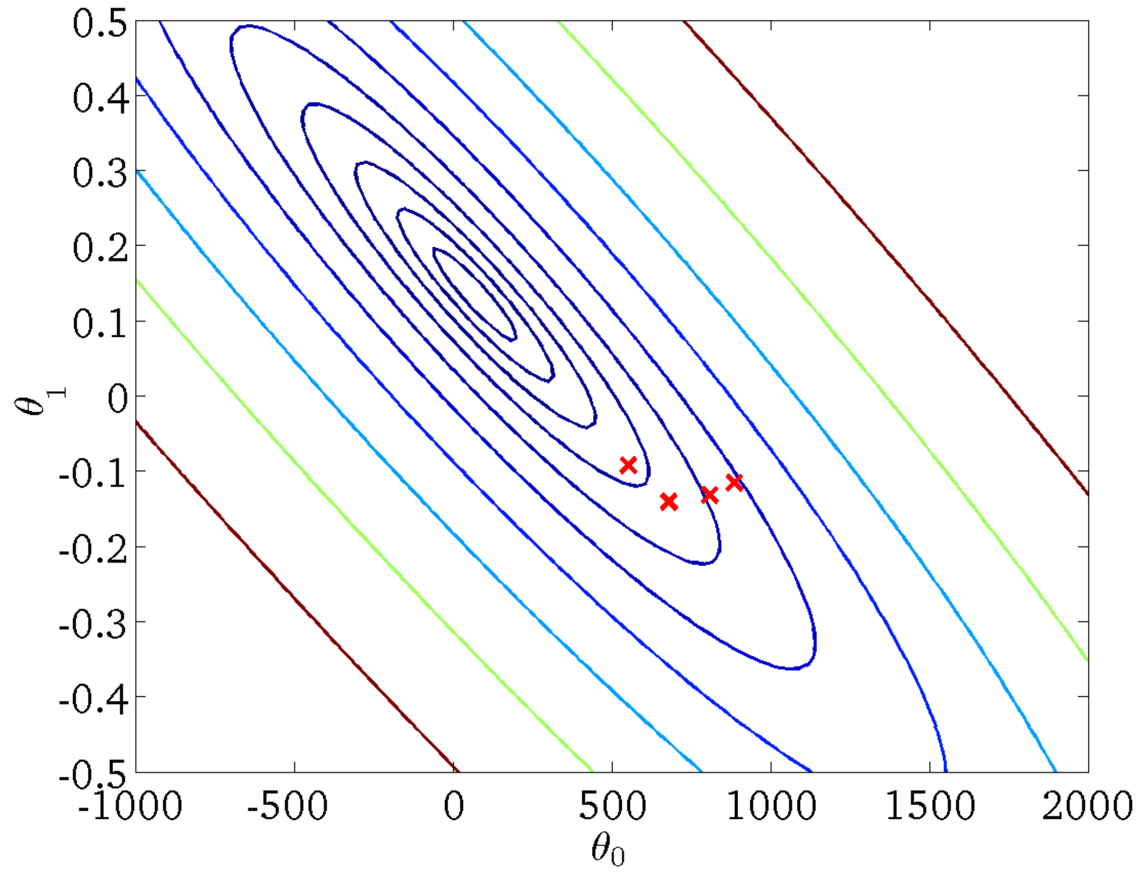
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)



$g(\theta)$

(function of the parameters θ_0, θ_1)

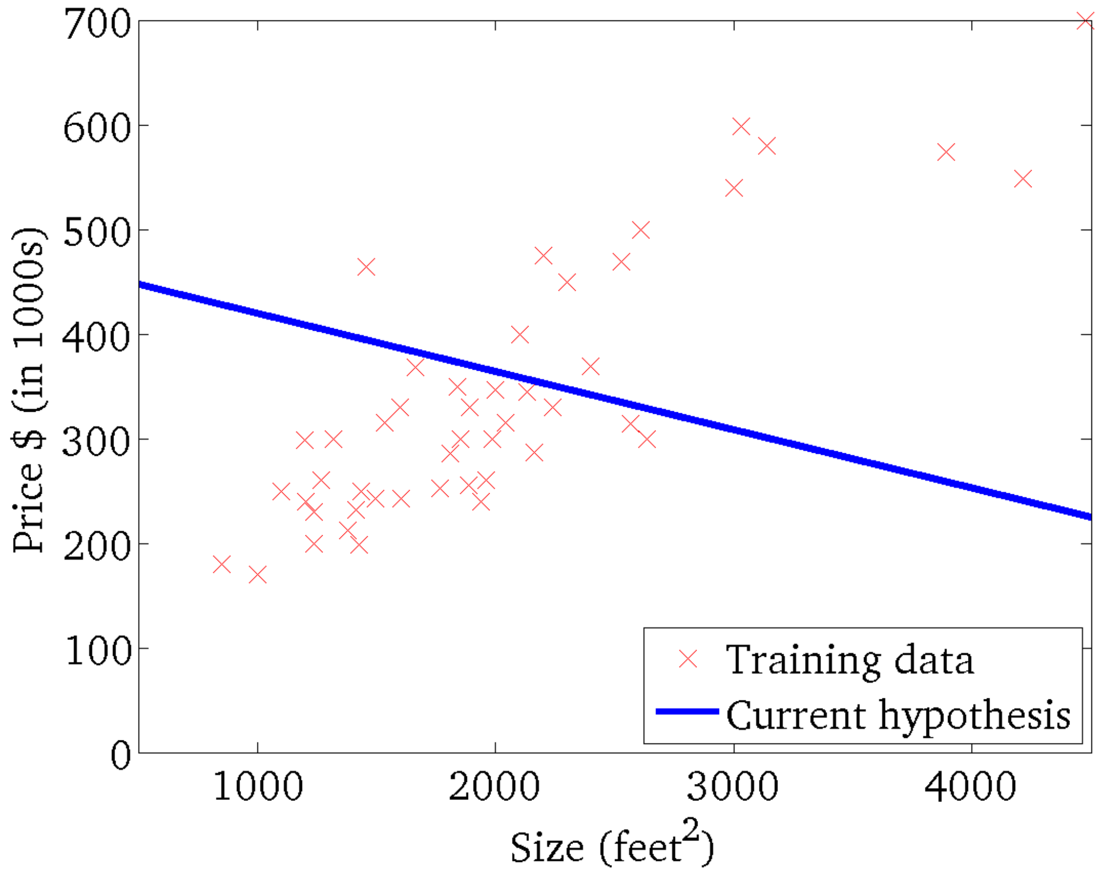


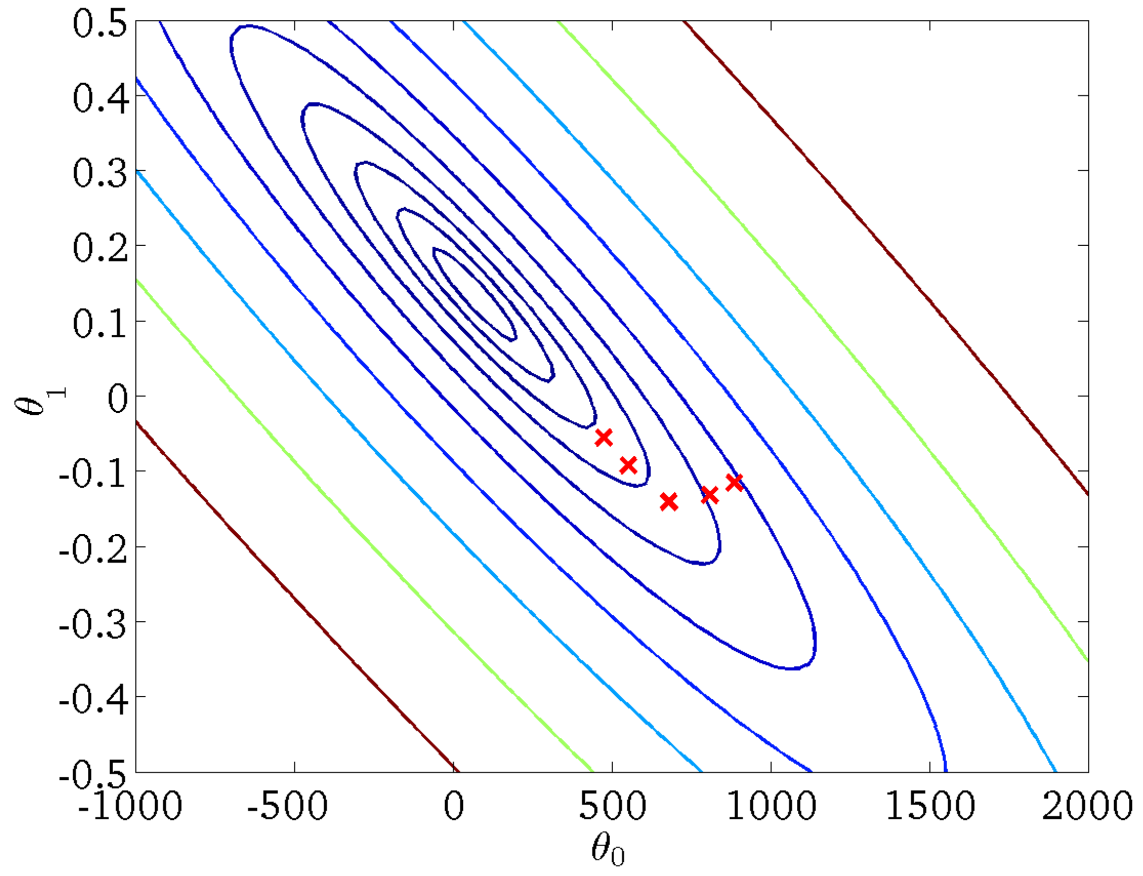
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

(function of the parameters θ_0, θ_1)



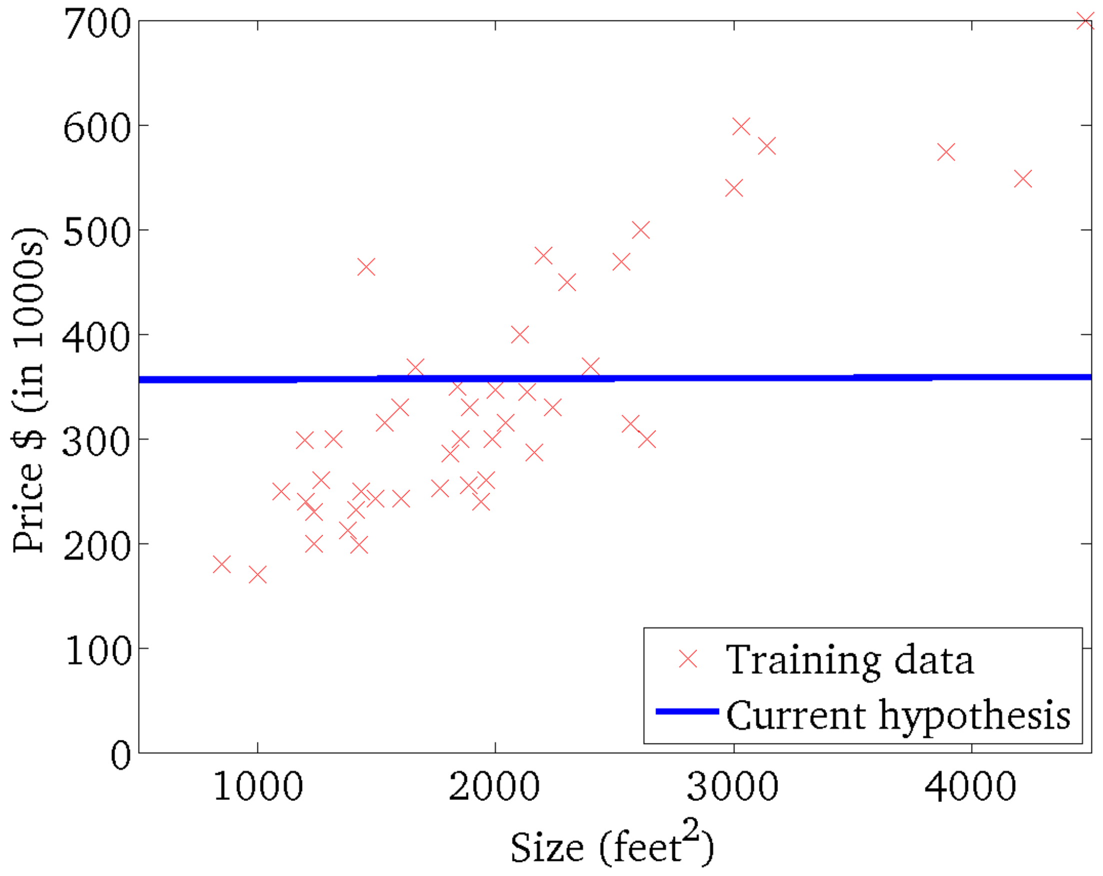


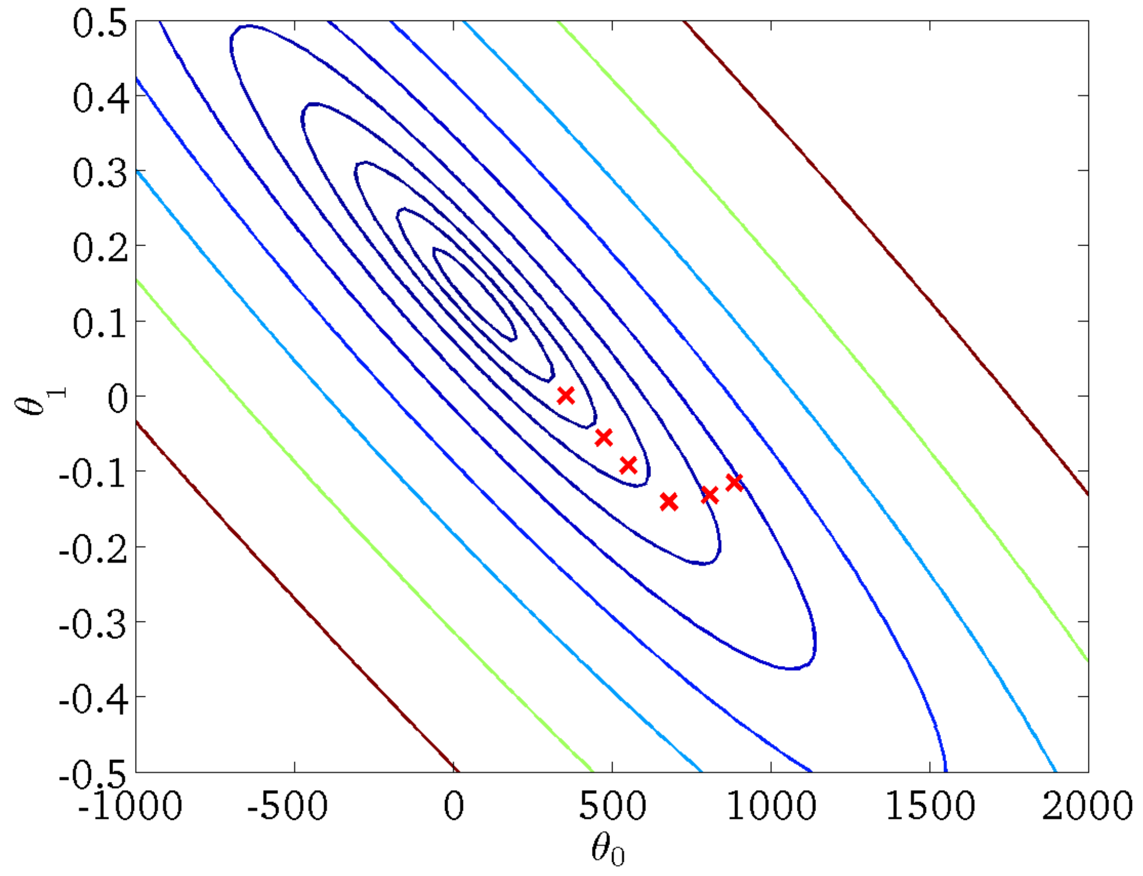
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

(function of the parameters θ_0, θ_1)



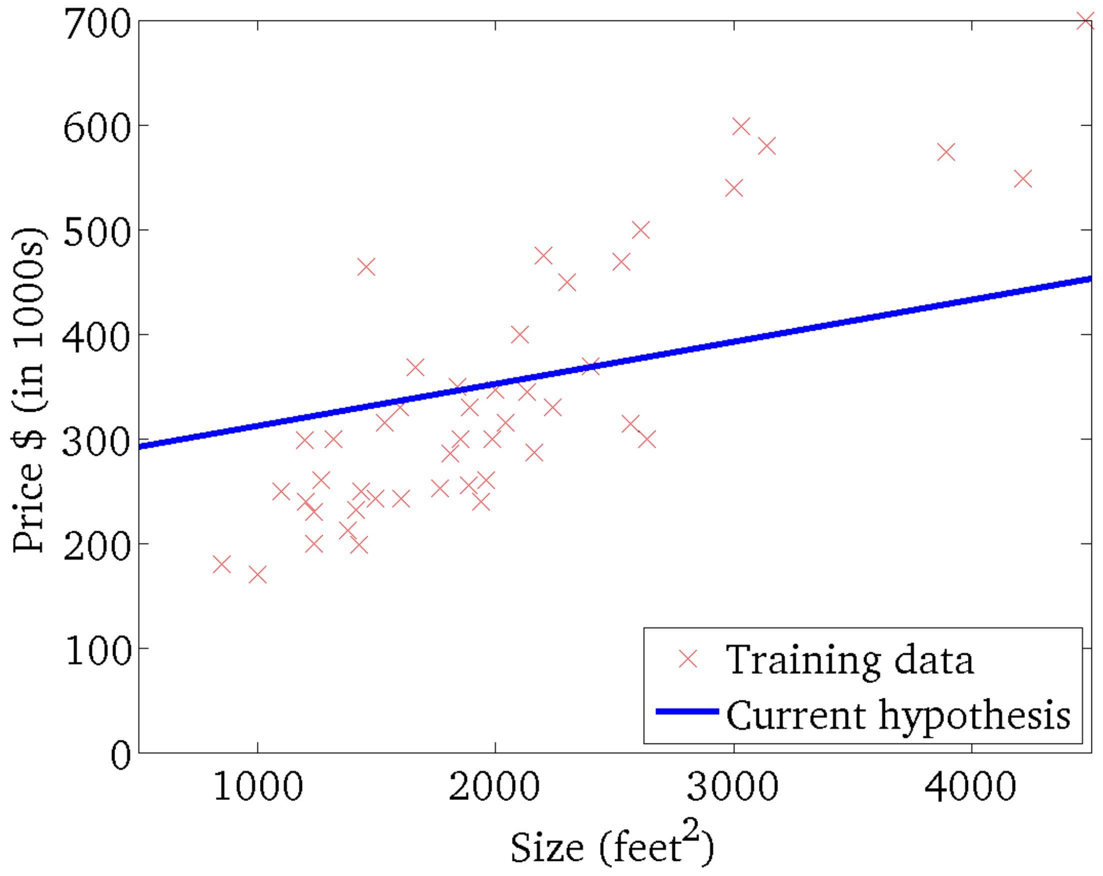


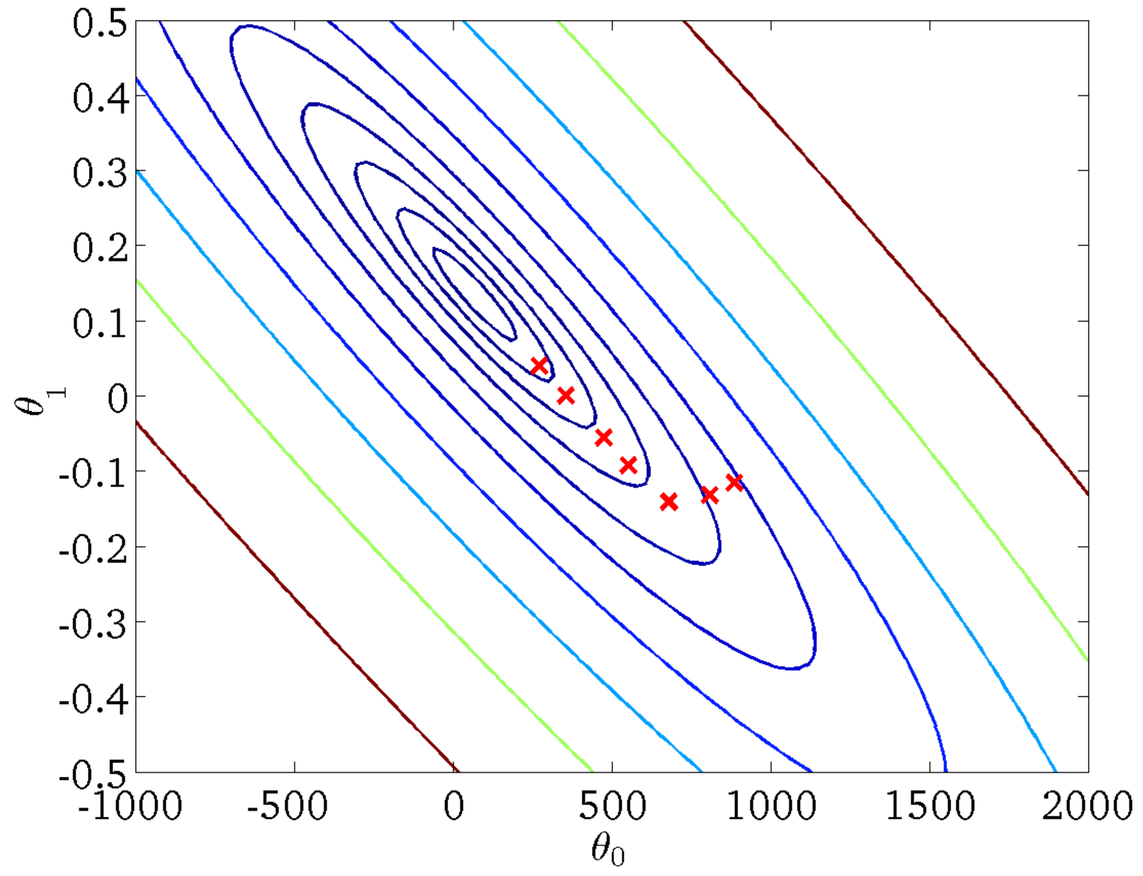
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

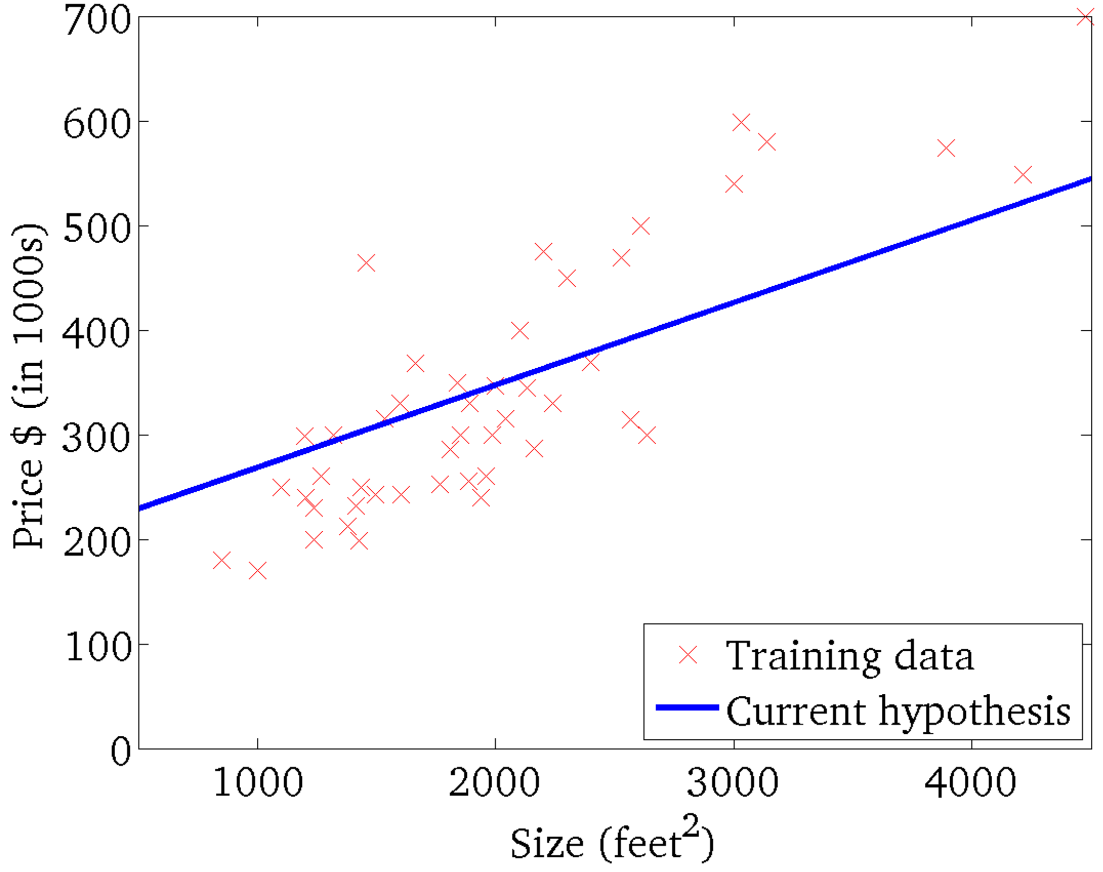
(function of the parameters θ_0, θ_1)





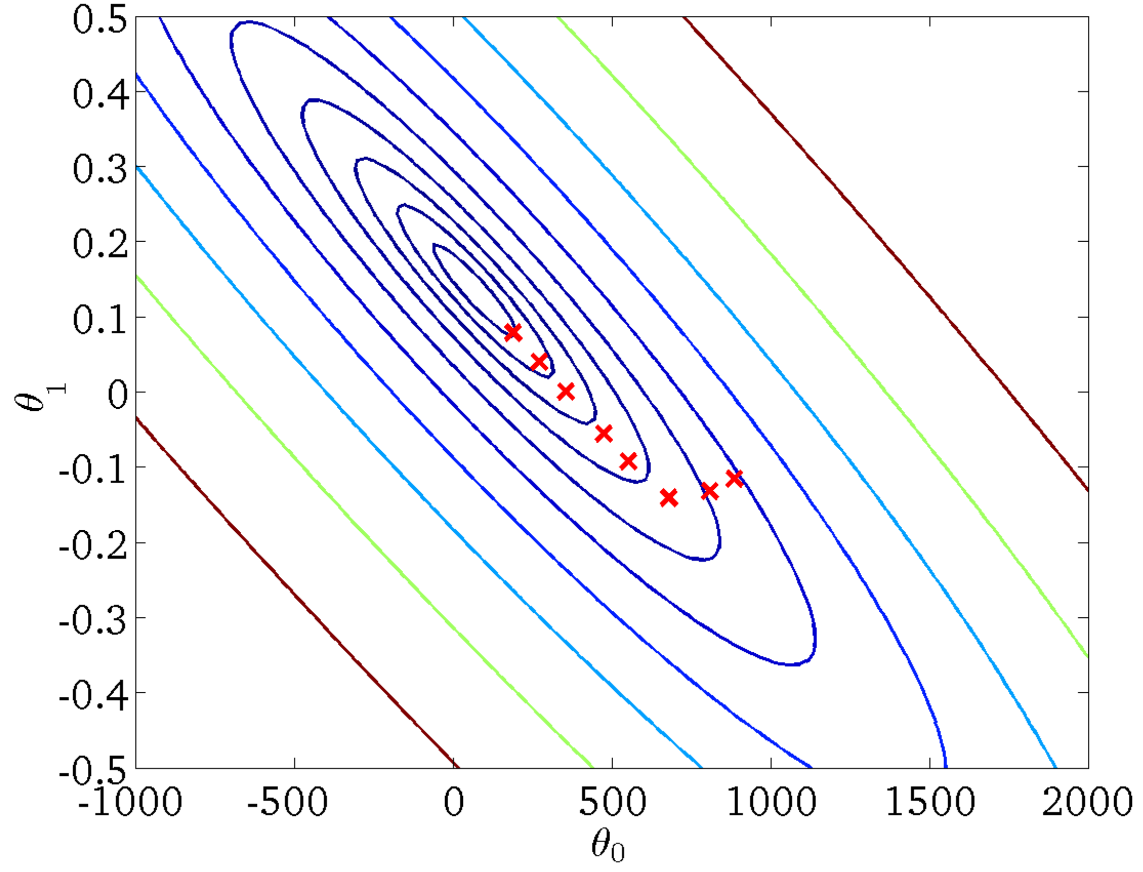
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)



$g(\theta)$

(function of the parameters θ_0, θ_1)

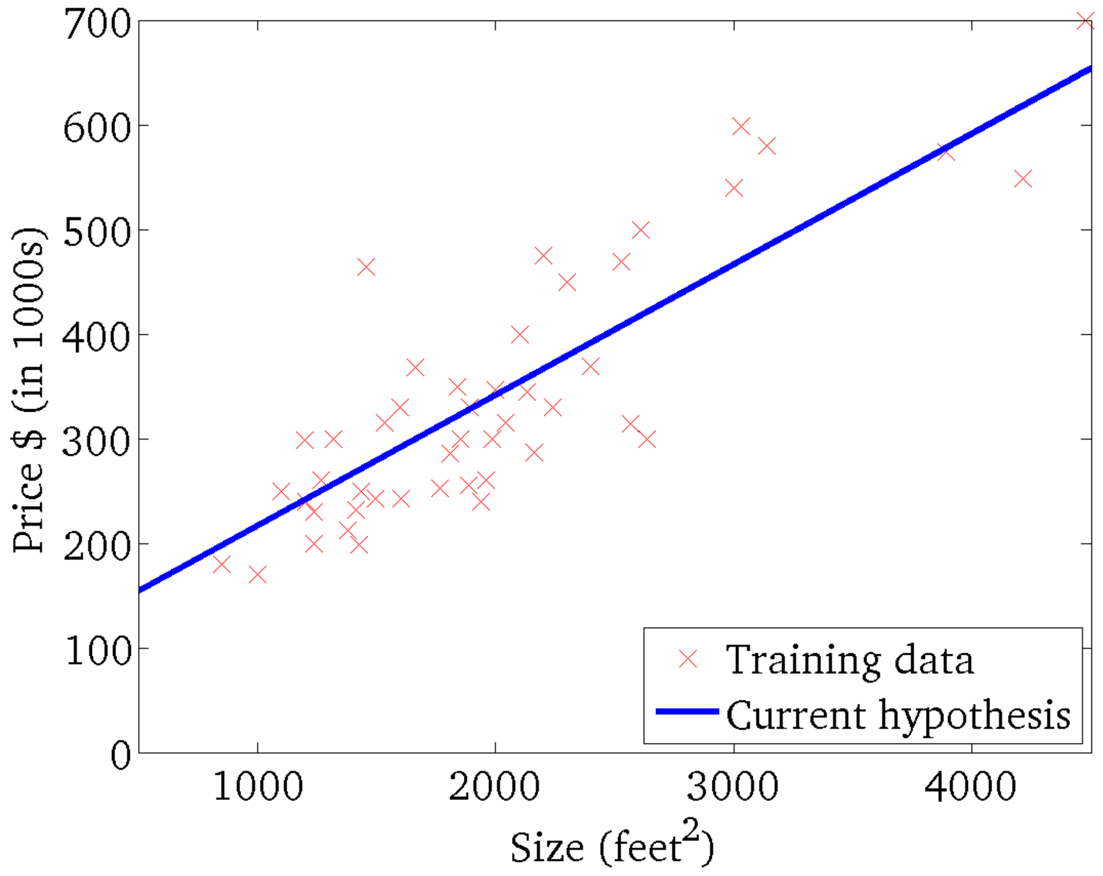


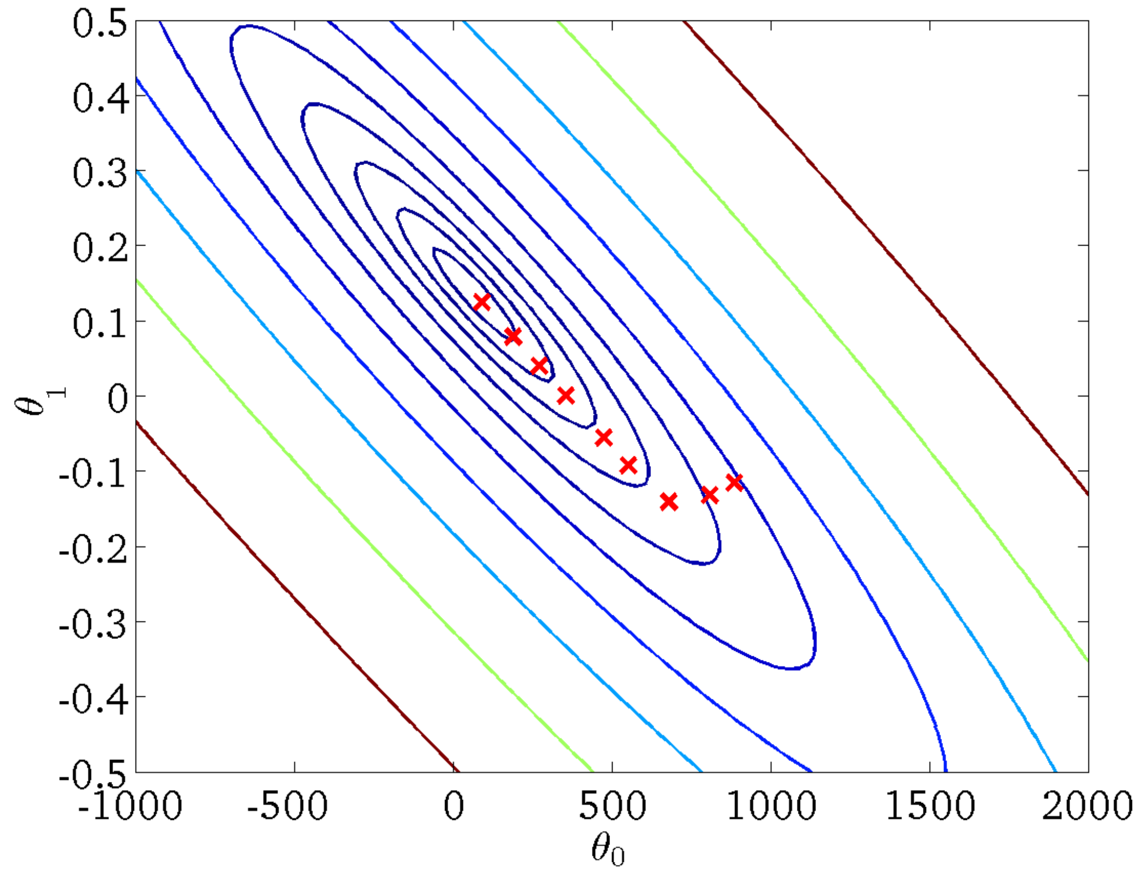
$f(x)$

(for fixed θ_0, θ_1, this is a function of x)

$g(\theta)$

(function of the parameters θ_0, θ_1)





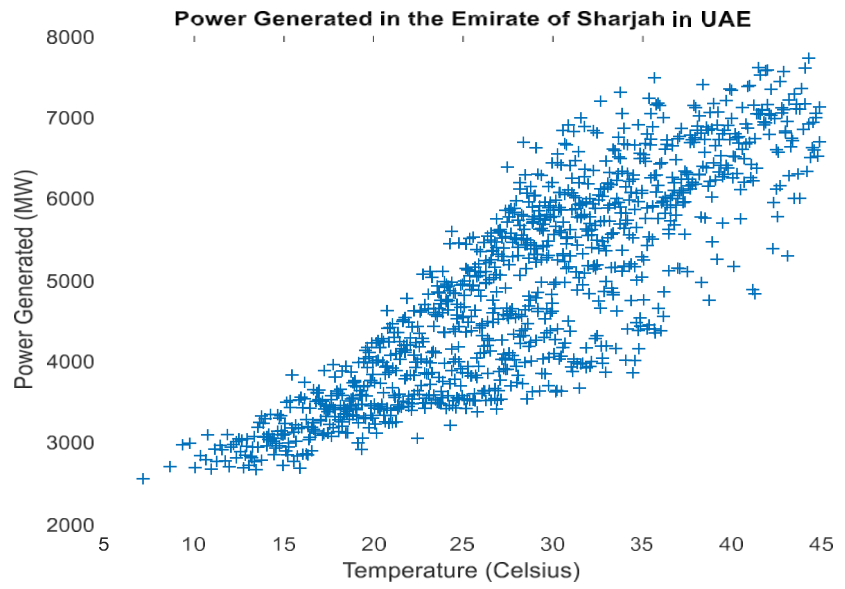
Using Simple Linear Regression

- Simple Linear Regression can be used in the following situations:
  - To see if IQ impacts income (IQ is the independent variable, and income is the dependent variable).
  - To examine if the number of working hours affects the number of hours slept (the dependent variable is hours of sleep, and the hours of work is the independent variable).
  - The number of cigarettes smoked is being tested to see if it affects blood pressure (the number of cigarettes smoked is the independent variable and blood pressure is the dependent variable).

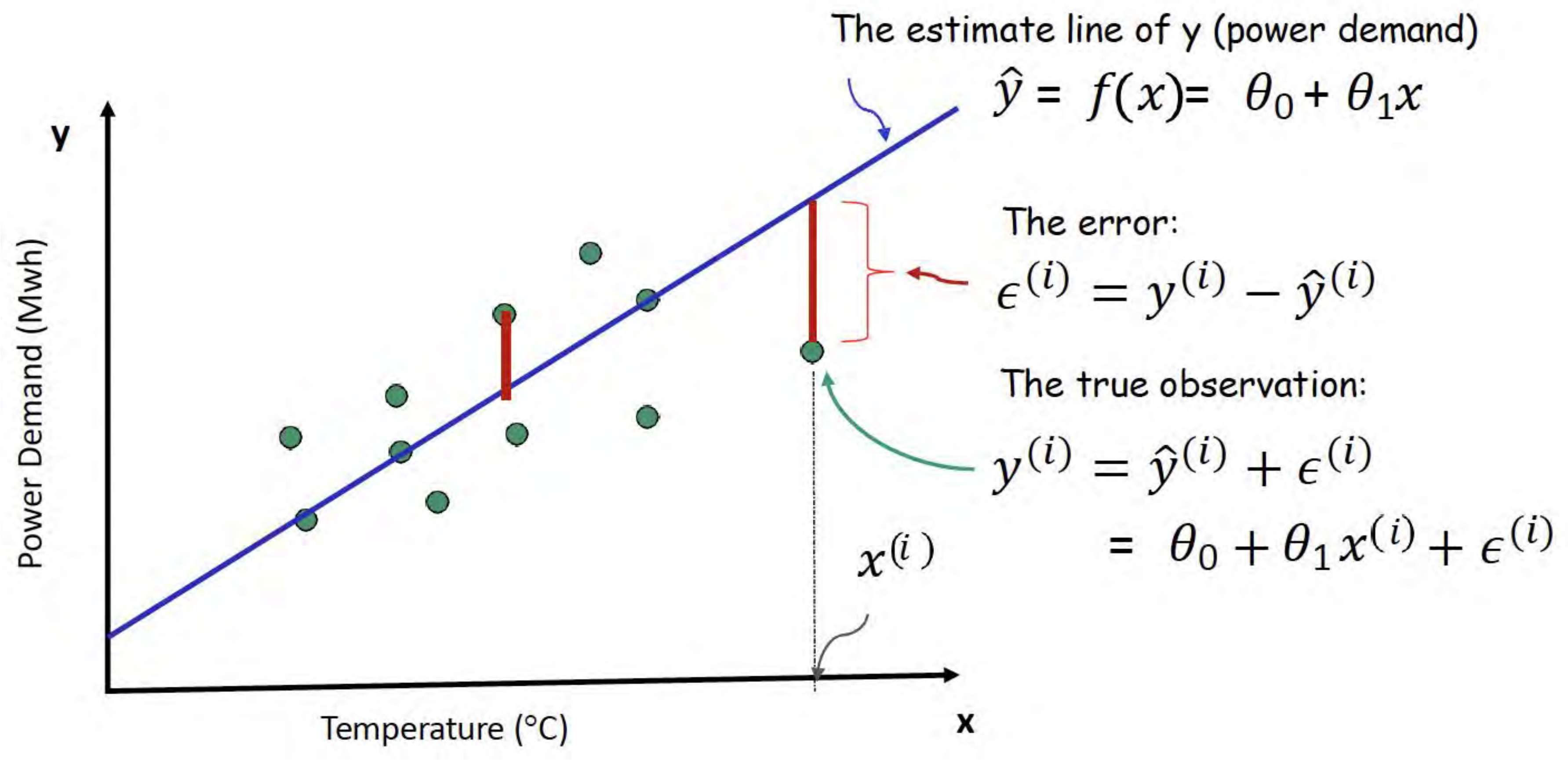
1.4: Linear Regression with Multiple Features

Attributes/Features

- In a previous example, we were looking at one “attribute” or “feature”:
  - Temperature
- Usually, we look at multiple features/input attributes. Example:
  - Temperature
  - Humidity
  - Date
  - Time
  - Holiday
  - House income
  - Type of customers: Residential / Commercial
  - ….
- Challenge: How to deal with a large number of features?



Simple Linear Regression (Power Generation Example)



The estimate line of y (power demand)
$\hat{y} = f(x) = \theta_0 + \theta_1 x$
The error:
$\epsilon^{(i)} = y^{(i)} - \hat{y}^{(i)}$
The true observation:
$y^{(i)} = \hat{y}^{(i)} + \epsilon^{(i)}$
$= \theta_0 + \theta_1 x^{(i)} + \epsilon^{(i)}$
$x^{(i)}$
y
Power Demand (Mwh)
Temperature (°C)
x

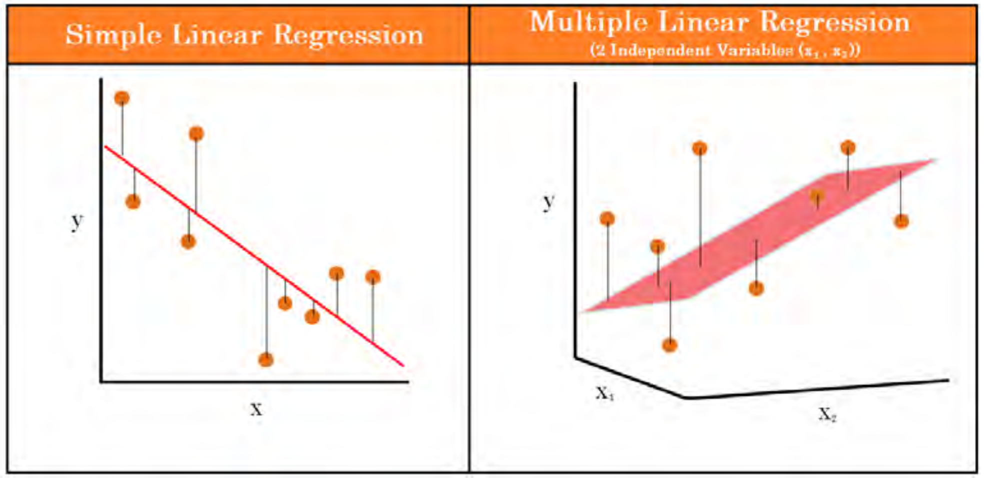
Attributes/Features

- The goal of linear regression with multiple features is to find the best line that fits the target variable y based on ***n*** input variables\features
- Mathematically: $f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- (x^i, y^i): i^{th} training example

  $y^{(i)}$: target variable (Scaler)

  $x^{(i)}$: input variables (Vector)

$$x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ \cdot \\ \cdot \\ \cdot \\ x_n^{(i)} \end{bmatrix}$$



$f(x) = \theta_0 + \theta_1 x_1$ $\qquad$ $f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

Source: https://www.analyticsvidhya.com/blog/2021/11/startups-profit-prediction-using-multiple-linear-regression/

Matrix Notation of the Regression Model

- In multiple linear regression, you can express the relationship between your dependent variable Y and independent variables X in matrix form as: $Y = X\beta$

Simplified form with assumption

Here:
  - *Y* is a vector of observed values for the dependent variable.
  - *X* is a matrix of the independent variables (predictors).
  - β is a vector of unknown coefficients.
- The goal of the least squares method is to find the values of the coefficients in β that minimize the residual sum of the squared. In other words, you want to minimize:

$$RSS\ (\theta) = \sum(\epsilon_i)^2 = \sum(Y_i - X_i\beta)^2$$

- Where, ϵ is a vector of residuals of errors.

1.5: Find the Best Model: Multiple Features

Multiple Dimensions: Gradients

- In the context of multiple features linear regression, "gradients" refer to the partial derivatives of the loss function (RSS) with respect to each feature's coefficient.
- To calculate each partial derivative, we use the chain rule and obtain:

$$\frac{\partial}{\partial \beta_0}\mathrm{RSS}(\beta) = -2\sum (Y_i - X_i\beta)$$
$$\frac{\partial}{\partial \beta_1}\mathrm{RSS}(\beta) = -2\sum (Y_i - X_i\beta)X_{i1}$$
$$\vdots$$
$$\frac{\partial}{\partial \beta_n}\mathrm{RSS}(\beta) = -2\sum (Y_i - X_i\beta)X_{in}$$

- Where X_{ij} is the value of the *j*-th feature for the *i*-th observation.

Multiple Dimensions: Gradients (cont.)

- In compact matrix form, the gradient of the RSS with respect to β and includes all the partial derivatives with respect to each coefficient is:

Called *nabla*, but often pronounced as *del* or *gradient*

$$\nabla \mathrm{RSS}(\beta) = -2X^T(Y - X\beta)$$

- For simplicity, we can say that gradient is a way of packing together all the partial derivative information of a function.

- Example: Consider $f(x, y) = x^2y + \sin(y)$

$$\frac{\partial f}{\partial x} = 2xy$$

$$\frac{\partial f}{\partial y} = x^2 + cos(y)$$

- Gradient puts these two partial derivatives together in a vector as follows:

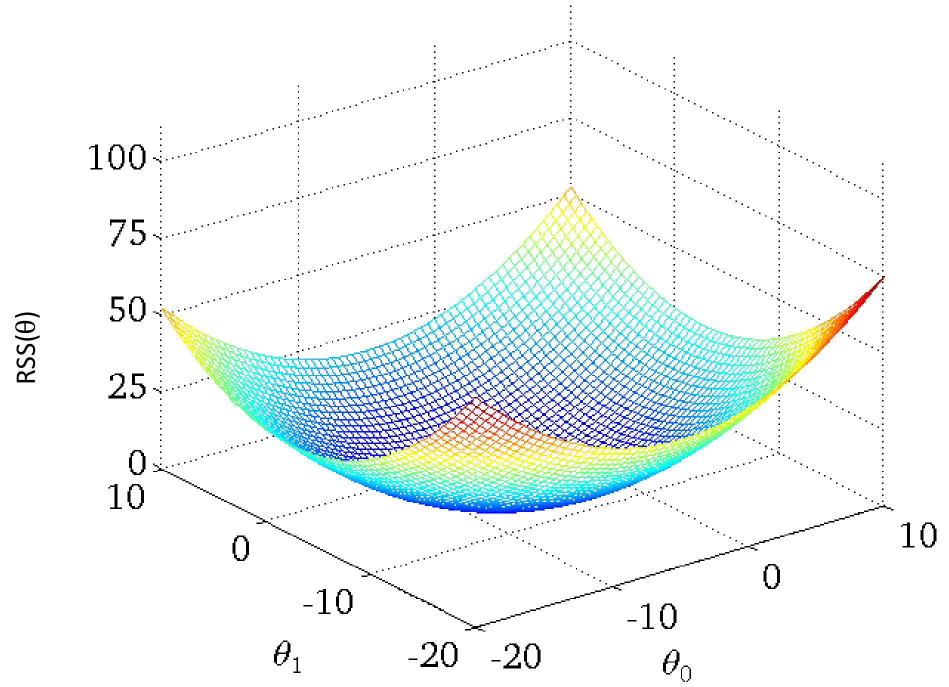
$$\nabla f(x, y) = \nabla\, x^2y + \sin(y) = \begin{bmatrix} 2xy \\ x^2 + \cos(y) \end{bmatrix}$$

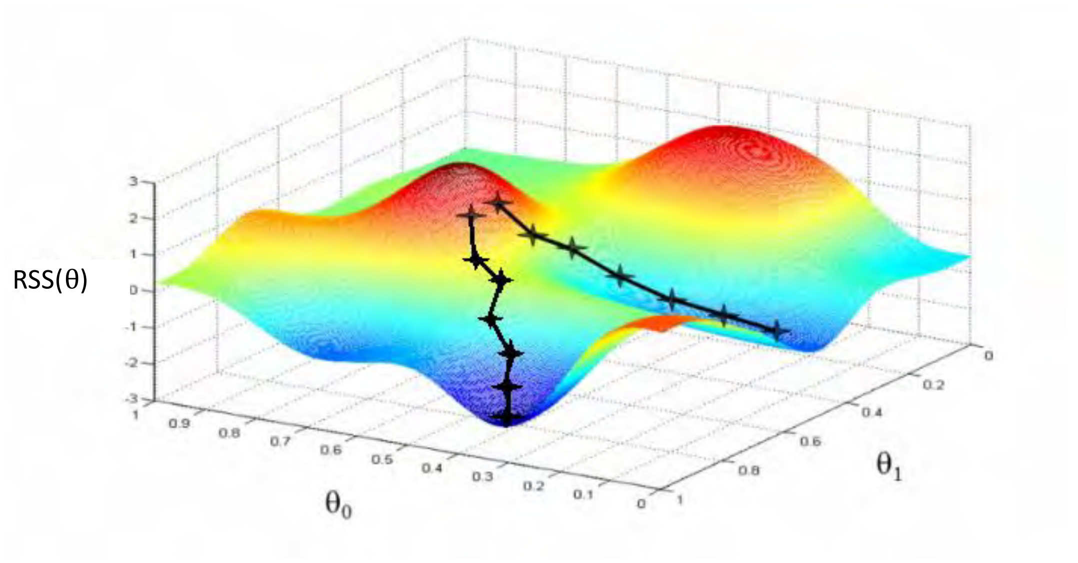
Gradient Descent Algorithm

- Gradient Descent is an algorithm that finds the best-fit line for a given training dataset in a smaller number of iterations.
- The gradient descent algorithm starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function. This iterative minimization is achieved by taking steps in the direction that minimizes the residuals.

1. Start with initial guesses
2. Start at 0,0 (or any other value)
3. Keep changing β a little bit to try and reduce RSS
4. Each time you change the parameters, you select the gradient that reduces RSS to the most possible
5. Repeat
6. Do so until you converge to a minimum (local or global)
7. Note: The gradient descent algorithm has an interesting property. Where you start can determine which minimum you end up

Gradient Descent Algorithm (cont.)





For Illustration

Conclusion

- Machine learning is categorized into several types, including supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.
- Linear regression is introduced as a statistical tool used for predicting **continuous-valued response** variables. It involves features, observations, and feature engineering.
- Simple linear regression is discussed, focusing on finding the best-fit line using techniques like the Residual Sum of Squares (RSS).
- Optimization techniques are explored, including the concept of concave and convex functions, finding maxima and minima analytically, and using algorithms.
- Simple linear regression is applied to real-world scenarios, including examining the impact of independent variables on dependent variables.

Next

1.6: Model Performance

Thank you