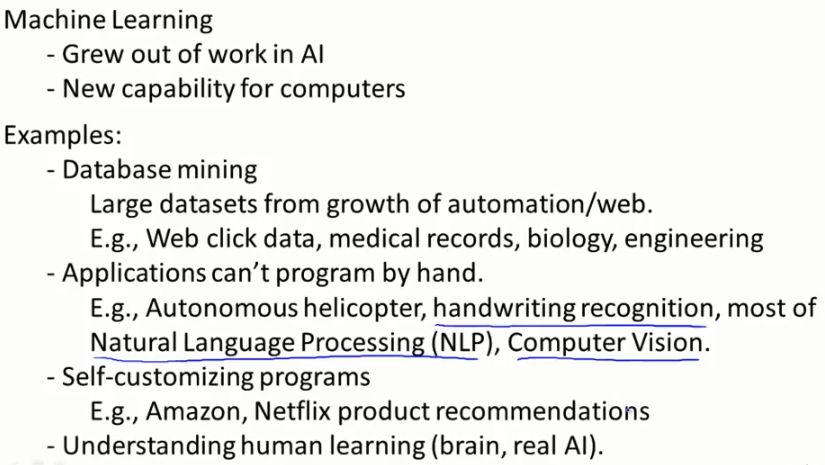
1. **SEMANA 1**
   1. **INTRODUCTION**

Welcome to Machine Learning! This week, we introduce the core idea of teaching a computer to learn concepts using data—without being explicitly programmed.

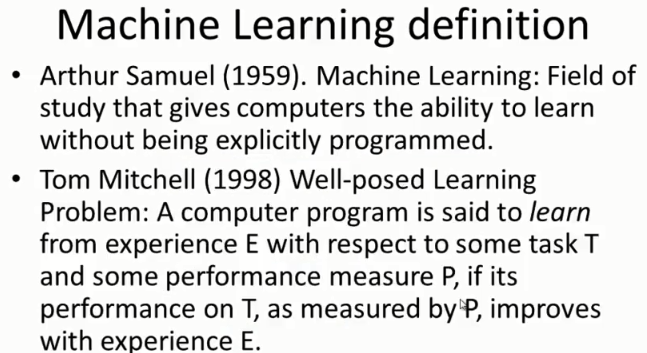
We are going to start by covering linear regression with one variable. Linear regression predicts a real-valued output based on an input value. We discuss the application of linear regression to housing price prediction, present the notion of a cost function, and introduce the gradient descent method for learning.

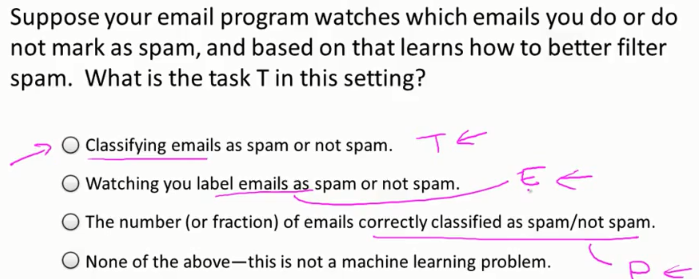
We’ll also have optional lessons that provide a refresher on linear algebra concepts. Basic understanding of linear algebra is necessary for the rest of the course, especially as we begin to cover models with multiple variables. If you feel confident in your understanding of linear algebra, feel free to take a break or help other students out in the forums.

* + 1. **Introduction**
       1. **Video: Welcome**

****

* + - 1. **Video: What is Machine Learning?**

****

****

**Machine Learning Algorithms**: Supervised Learning, Unsupervised Learning, Reinforcement Learning, Recommender Systems.

* + - 1. **Leitura: What is Machine Learning?**

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

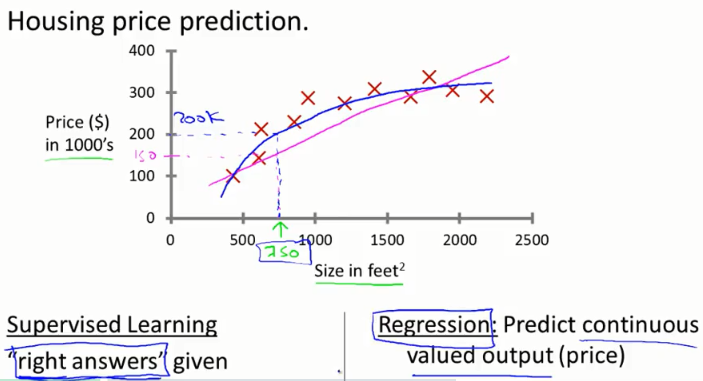
E = the experience of playing many games of checkers

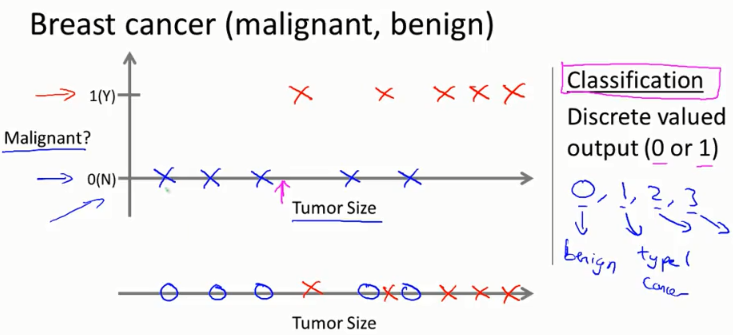
T = the task of playing checkers.

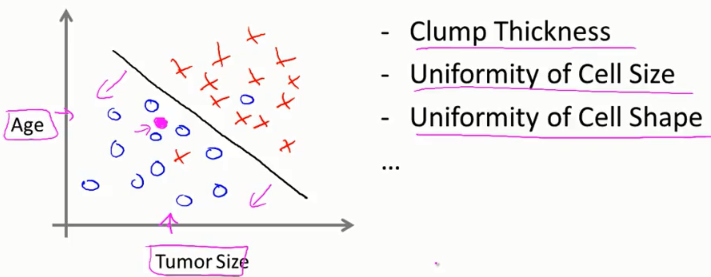
P = the probability that the program will win the next game.

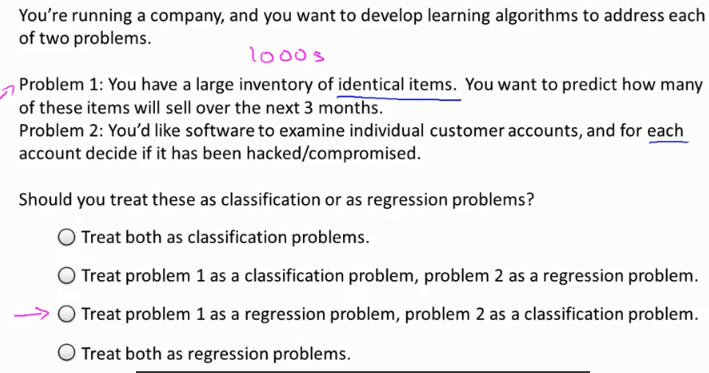
In general, any machine learning problem can be assigned to one of two broad classifications: Supervised learning and Unsupervised learning.

* + - 1. **Video: Supervised Learning**

****

****

****

****

* + - 1. **Leitura: Supervised Learning**

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

**Example 1:** Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

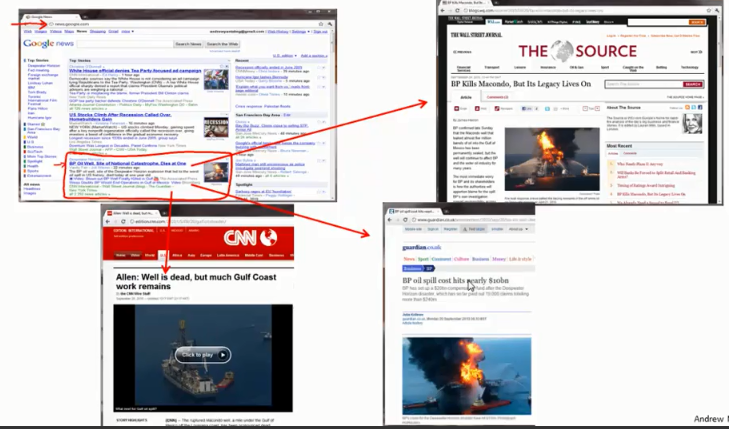
**Example 2:**

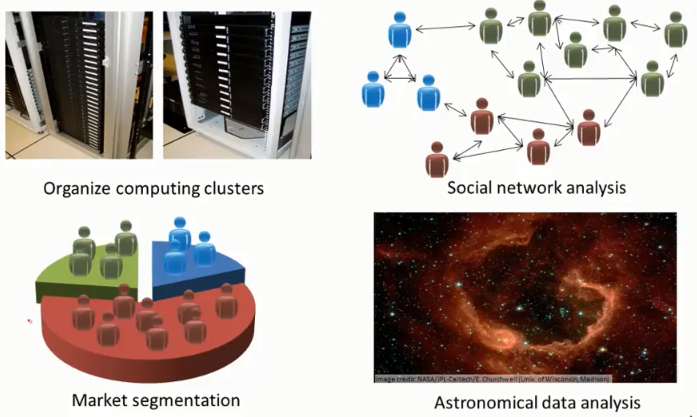
(a) Regression - Given a picture of a person, we have to predict their age on the basis of the given picture

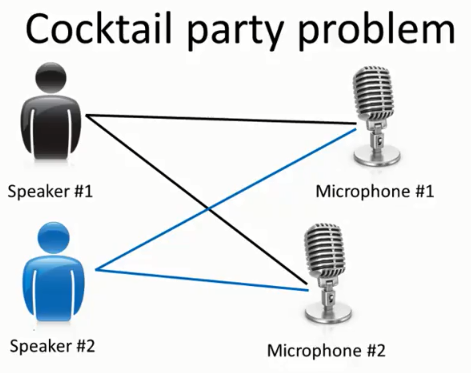
(b) Classification - Given a patient with a tumor, we have to predict whether the tumor is malignant or benign.

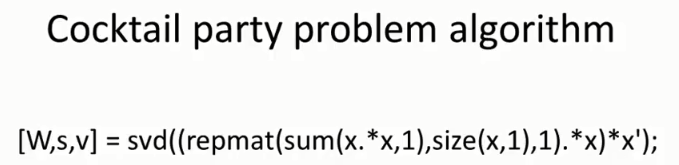
* + - 1. **Video: Unsupervised Learning**

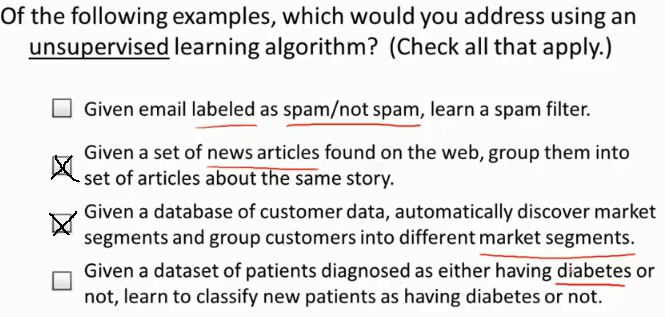
Google News uses clustering to group related News











* + - 1. **Leitura: Unsupervised Learning**

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

**Example:**

Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a cocktail party (<https://en.wikipedia.org/wiki/Cocktail_party_effect>).

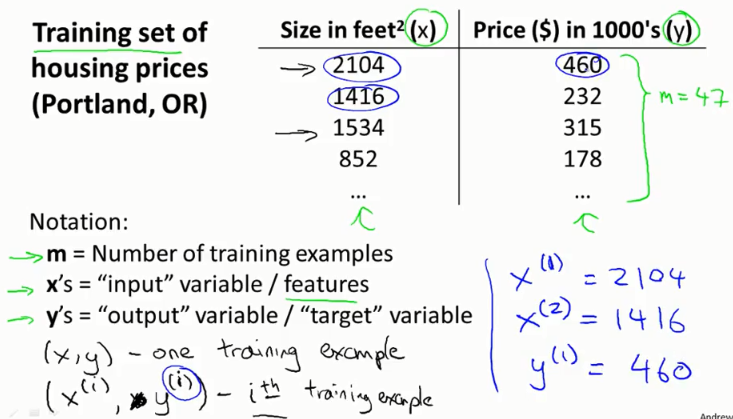
* + 1. **Review**
       1. **Leitura: Lecture Slides**

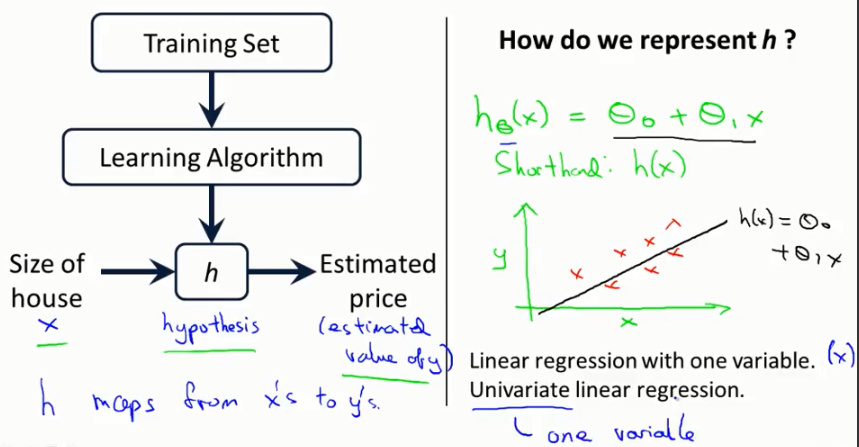
Lecture1.pdf

* + - 1. **Teste: Introduction**
  1. **LINEAR REGRESSION WITH ONE VARIABLE**

Linear regression predicts a real-valued output based on an input value. We discuss the application of linear regression to housing price prediction, present the notion of a cost function, and introduce the gradient descent method for learning.

* + 1. **Model and Cost Function**
       1. **Video: Model Representation**

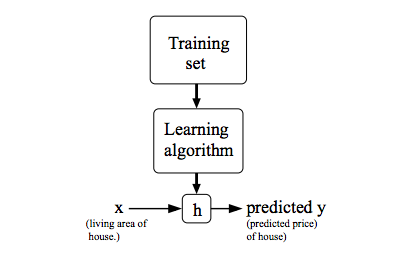
****

****

* + - 1. **Leitura: Model Representation**

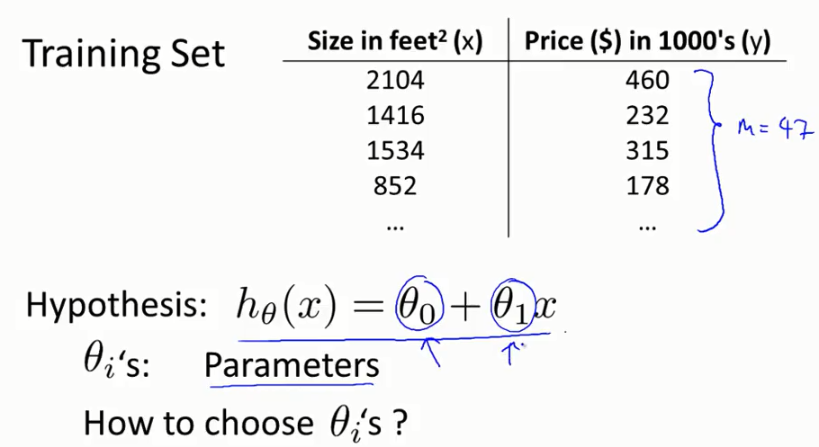
To establish notation for future use, we’ll use x(i) to denote the “input” variables (living area in this example), also called input features, and y(i) to denote the “output” or target variable that we are trying to predict (price). A pair (x(i), y(i)) is called a training example, and the dataset that we’ll be using to learn—a list of m training examples (x(i),y(i));i=1,...,m—is called a training set. Note that the superscript “(i)” in the notation is simply an index into the training set, and has nothing to do with exponentiation. We will also use X to denote the space of input values, and Y to denote the space of output values. In this example, X = Y = ℝ.

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like this:

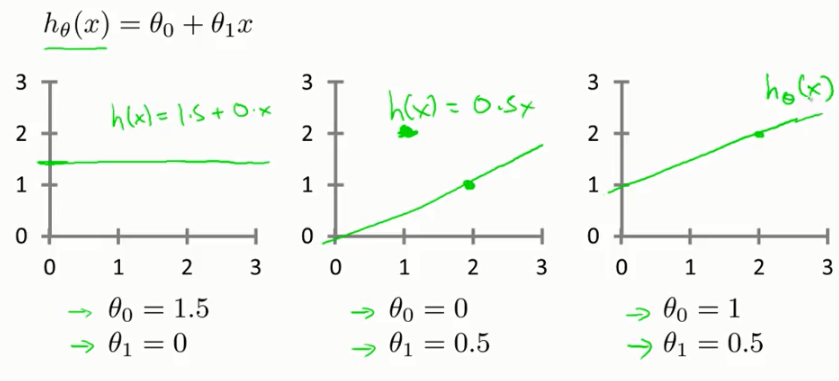


When the target variable that we’re trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

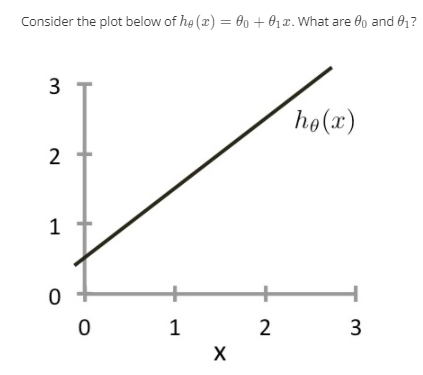
* + - 1. **Video: Cost Function**

****

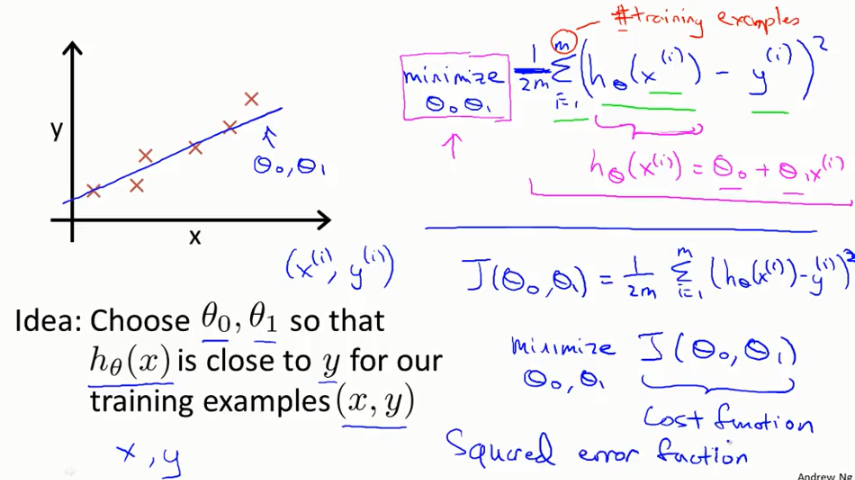
Theta = the parameters of the model



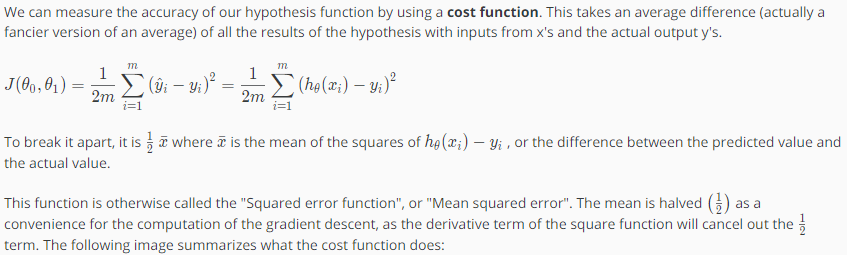
Examples of functions

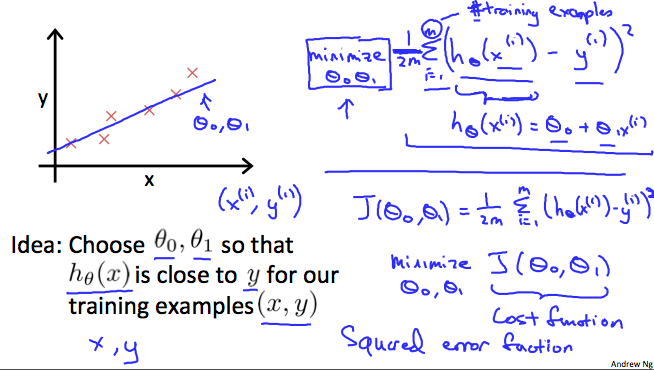


*θ*0​=0.5, *θ*1​=1

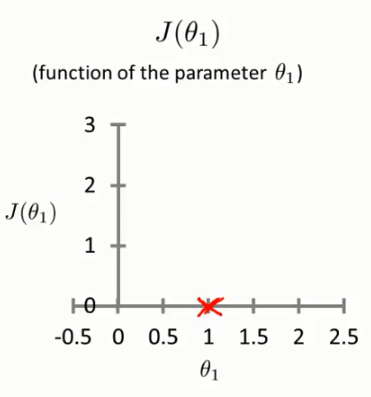
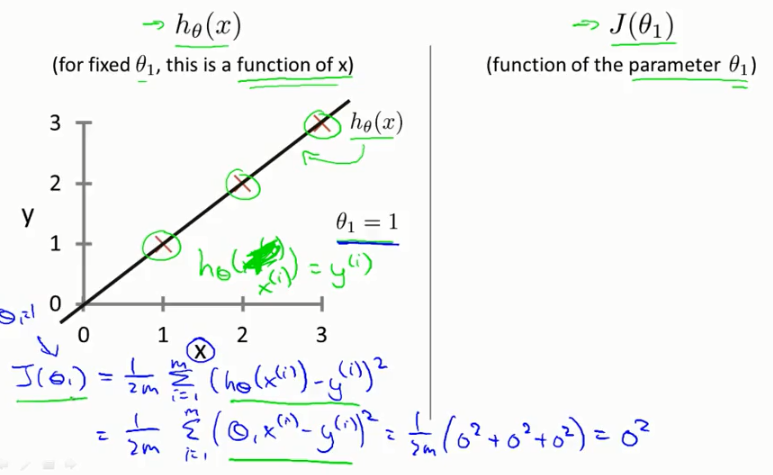


* + - 1. **Leitura: Cost Function**

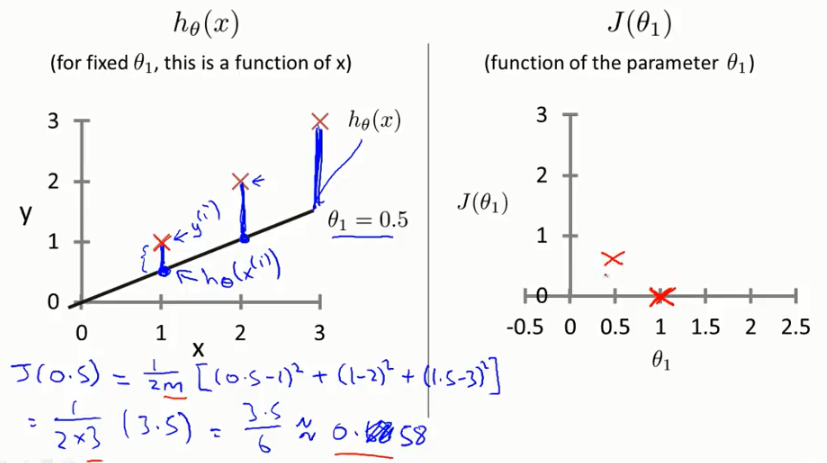
****



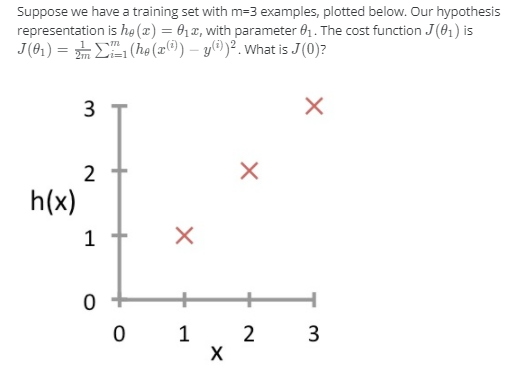
* + - 1. **Video: Cost Function Intuition I**

****

When *θ*1​=1, then J(*θ*1​) = 0. (Perfect fit)

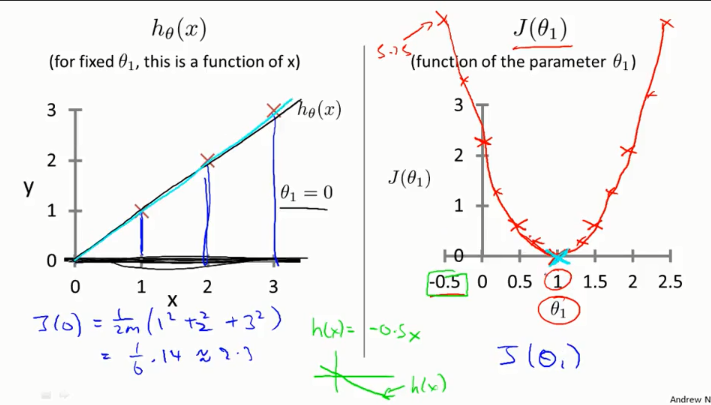


When *θ*1​=0.5, then J(*θ*1​) ~= 0.58





1/6 \* (1 + 4 + 9) = 1/6 \* 14 = 14/6

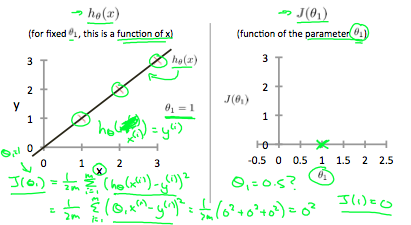


Different Theta values (x) and Cost Function (y)

* + - 1. **Leitura: Cost Function Intuition I**

If we try to think of it in visual terms, our training data set is scattered on the x-y plane. We are trying to make a straight line (defined by *hθ*​(*x*)) which passes through these scattered data points.

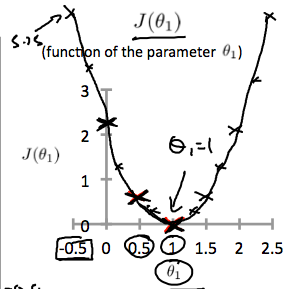
Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of *J*(*θ*0​, *θ*1) will be 0. The following example shows the ideal situation where we have a cost function of 0.



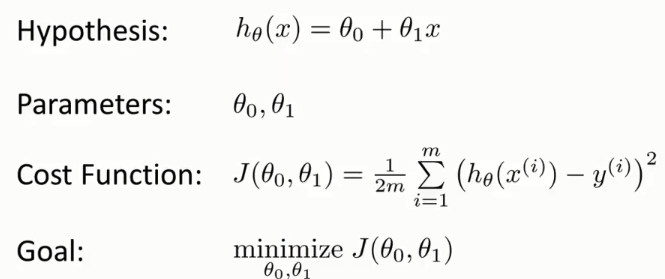
When θ1​=1, we get a slope of 1 which goes through every single data point in our model. Conversely, when θ1​=0.5, we see the vertical distance from our fit to the data points increase.

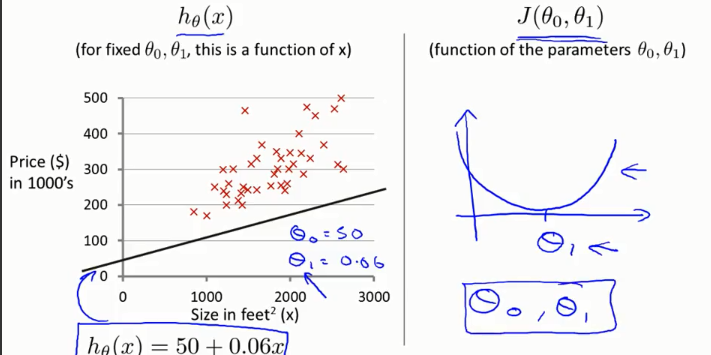


This increases our cost function to 0.58. Plotting several other points yields to the following graph:

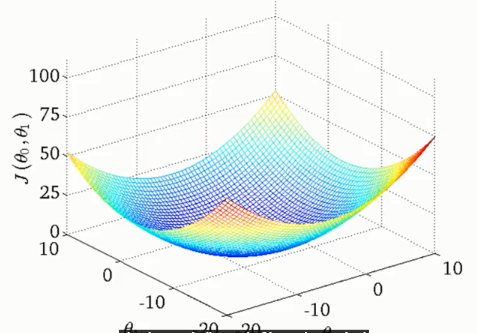


* + - 1. **Video: Cost Function Intuition II**



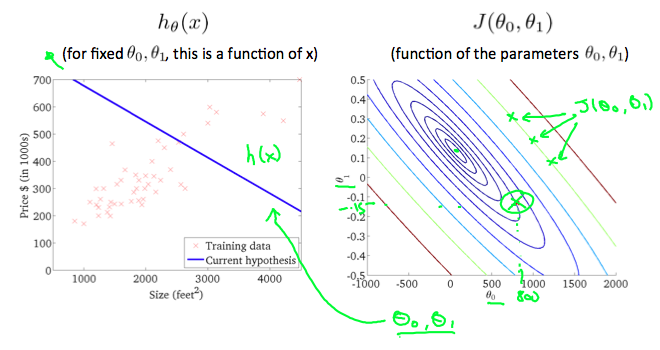


When we had only one parameter, the cost function looked like a bowl. Now, when we have2 parameters, it turns out the cost function also has a similar sort of bowl shape:

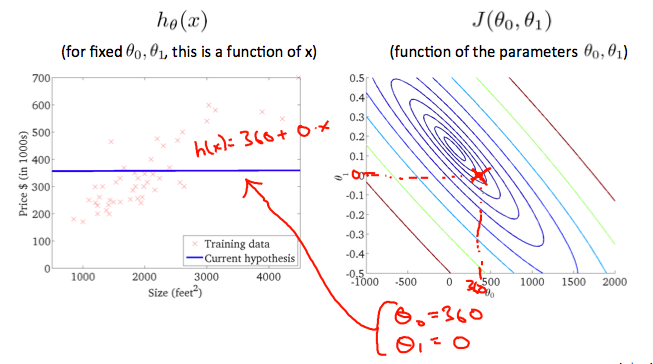


* + - 1. **Leitura: Cost Function Intuition II**

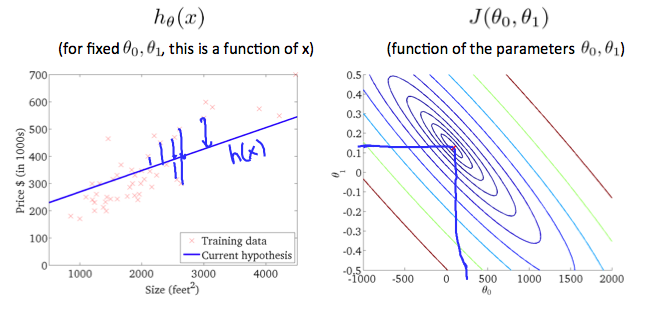
A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line. An example of such a graph is the one to the right below.



Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for *J*(*θ*0​, *θ*1​) and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when *θ*0​ = 800 and *θ*1​= -0.15. Taking another h(x) and plotting its contour plot, one gets the following graphs:



When *θ*0​ = 360 and *θ*1​ = 0, the value of  *J*(*θ*0​, *θ*1​) in the contour plot gets closer to the center thus reducing the cost function error. Now giving our hypothesis function a slightly positive slope results in a better fit of the data.

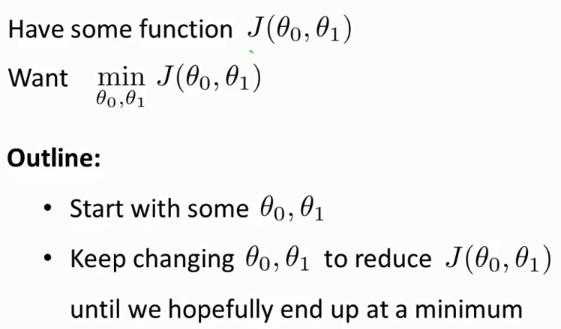


The graph above minimizes the cost function as much as possible and consequently, the result of *θ*1​ and *θ*0​ tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the center of the inner most 'circle'.

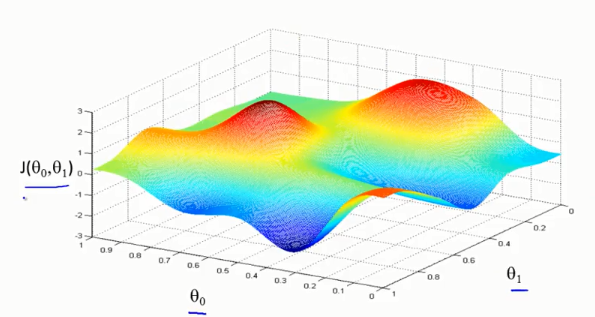
What we want is to have software to find the value of *θ*0, *θ*1 that minimizes this function and in the next video we start to talk about an algorithm for automatically finding that value of theta zero and theta one that minimizes the cost function J.

* + 1. **Parameter Learning**
       1. **Video: Gradient Descent**

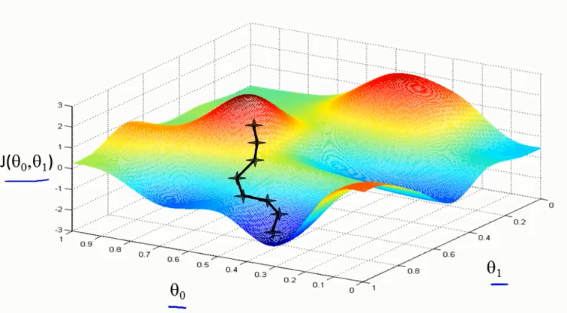
Linear regression with one variable: Gradient Descent, na algorithm to minimize the cost function J. GD is not only used in linear regression, it’s used all over the place in machine learning to minimize other functions as well.



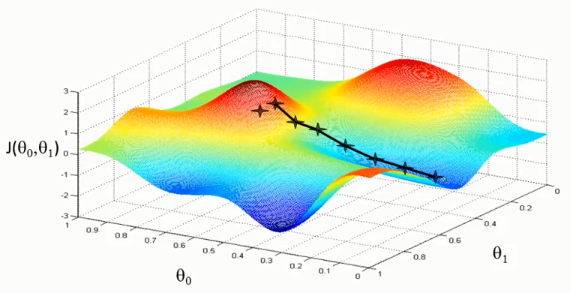
Say we’re trying to minimize the function below:



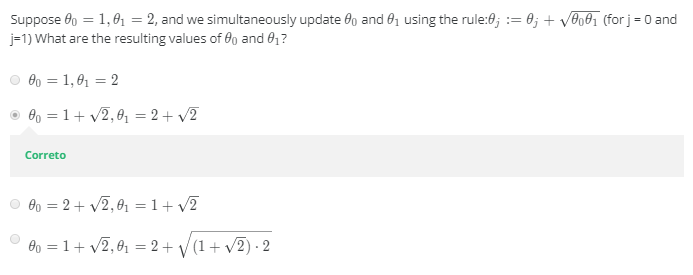
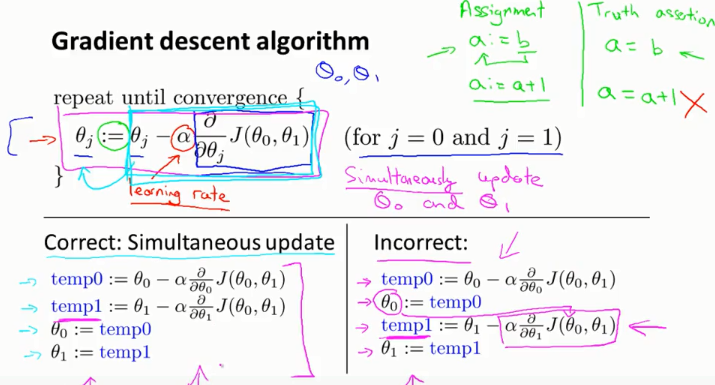
Then you initialize theta 0 and 1, take steps in direction to the minimum:



If you started in a point just a little right, the result would be this (another local optimum):



This is the intuition in pictures. Let’s take a look at the math:

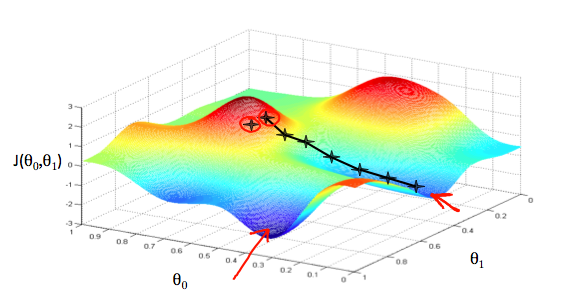


* + - 1. **Leitura: Gradient Descent**

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields *θ*0​ and *θ*1​ (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put *θ*0​ on the x axis and *θ*1​ on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction in which the step is taken is determined by the partial derivative of *J*(*θ*0​ ,*θ*1​). Depending on where one starts on the graph, one could end up at different points. The image above shows us two different starting points that end up in two different places.

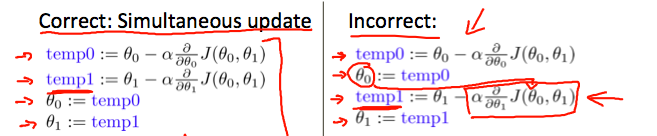
The gradient descent algorithm is:

repeat until convergence:

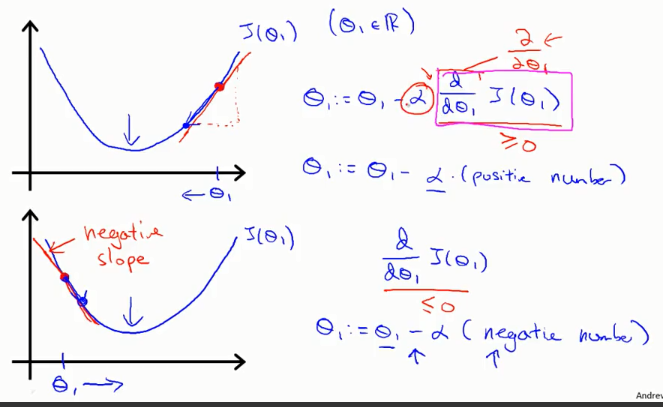


Where j=0,1 represents the feature index number.

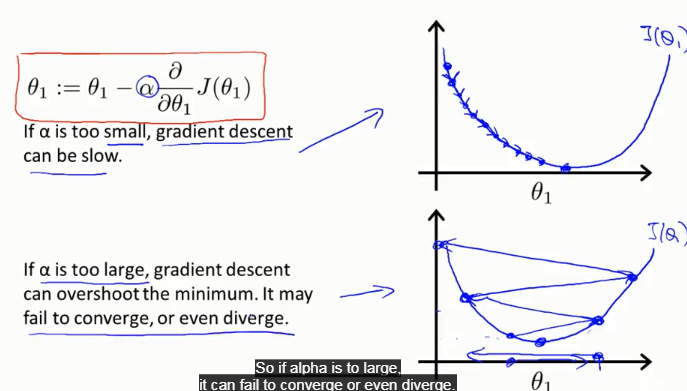
At each iteration j, one should simultaneously update the parameters *θ*1​,*θ*2​,...,*θn*​. Updating a specific parameter prior to calculating another one on the *j*(*th*) iteration would yield to a wrong implementation.

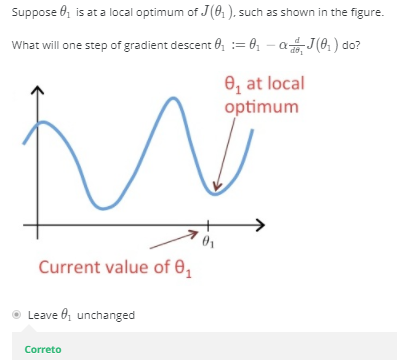


* + - 1. **Video: Gradient Descent Intuition**

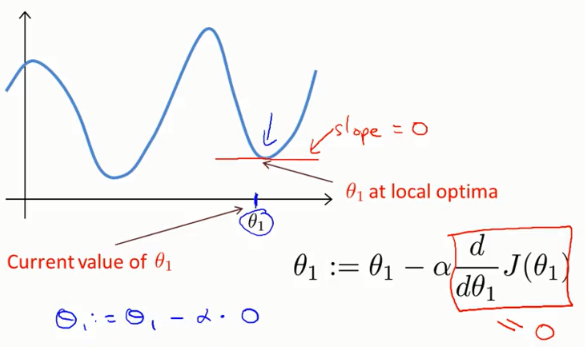
****

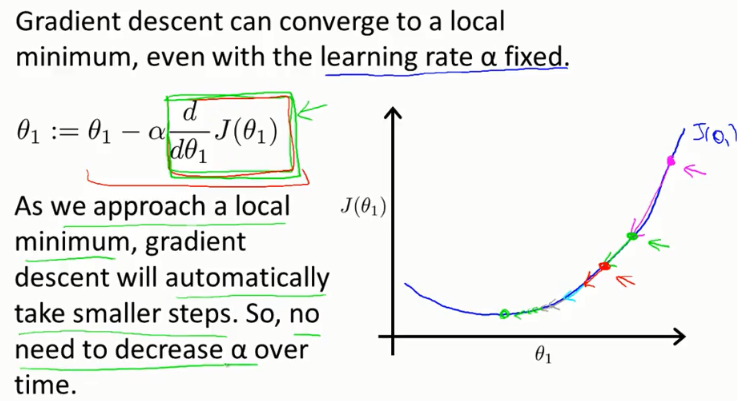
Trying to find the minimum with derivative \* alpha





Because the slope will be 0:





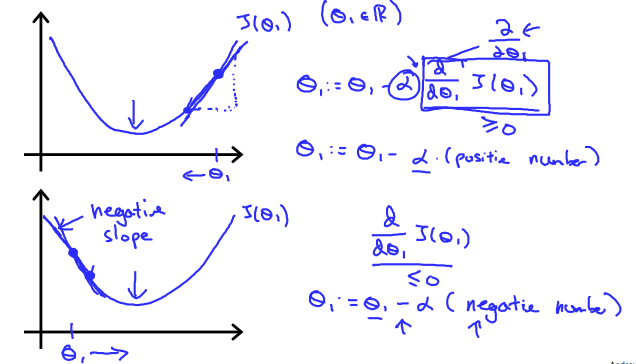
* + - 1. **Leitura: Gradient Descent Intuition**

In this video we explored the scenario where we used one parameter *θ*1​ and plotted its cost function to implement a gradient descent. Our formula for a single parameter was :

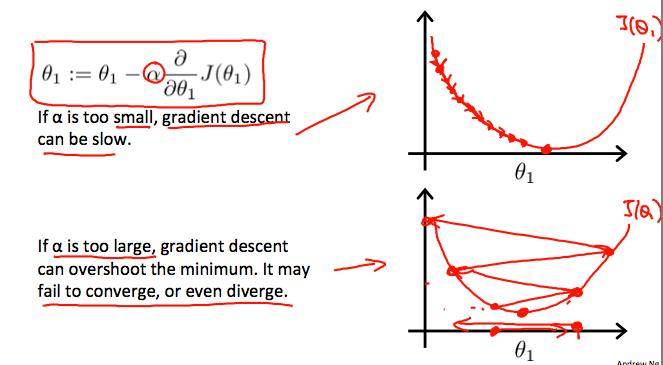
Repeat until convergence:



Regardless of the slope's sign for ​ eventually converges to its minimum value. The following graph shows that when the slope is negative, the value of *θ*1​ increases and when it is positive, the value of *θ*1​ decreases.



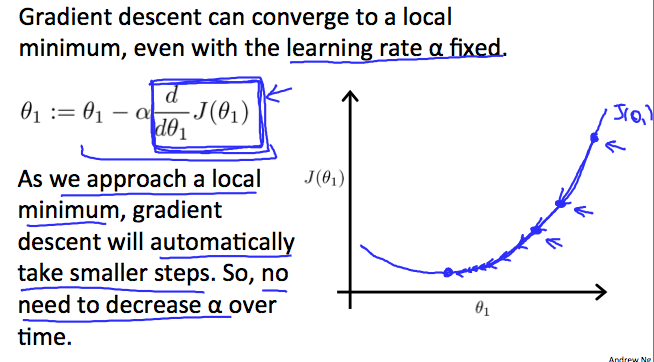
On a side note, we should adjust our parameter *α* to ensure that the gradient descent algorithm converges in a reasonable time. Failure to converge or too much time to obtain the minimum value imply that our step size is wrong.



### **How does gradient descent converge with a fixed step size *α*?**

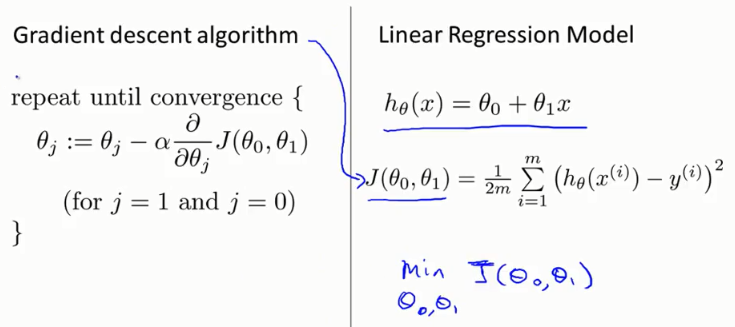
The intuition behind the convergence is that  approaches 0 as we approach the bottom of our convex function. At the minimum, the derivative will always be 0 and thus we get:

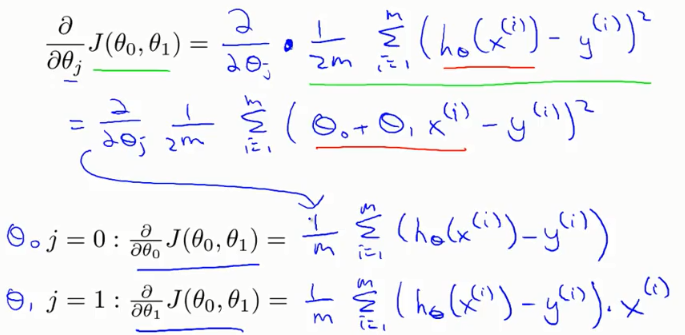
*θ*1​:=*θ*1​−*α* ∗ 0



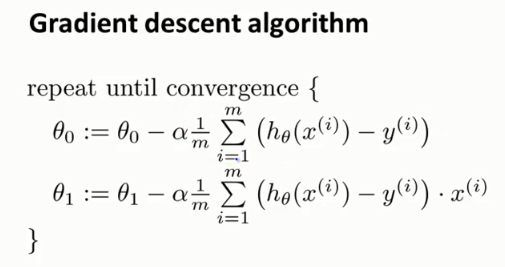
* + - 1. **Video: Gradient Descent For Linear Regression**

We will apply Gradient Descent to minimize the Squared Error Function of Linear Regression:

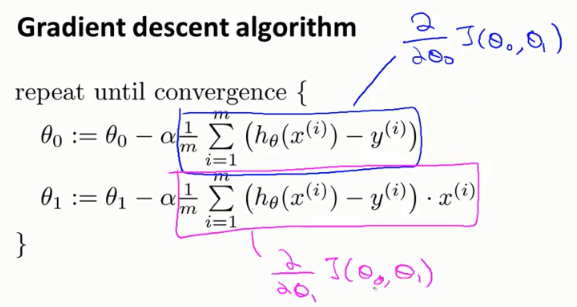


****

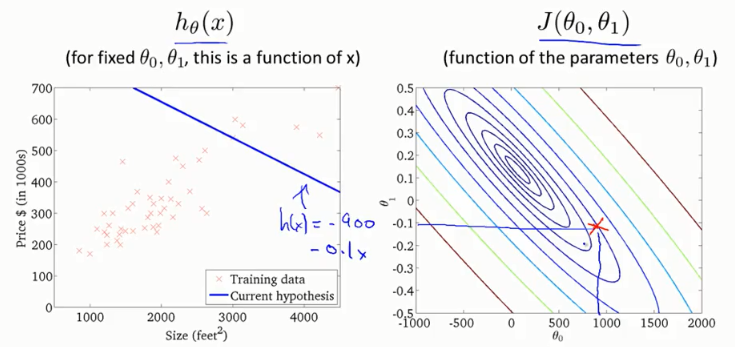
What becomes:

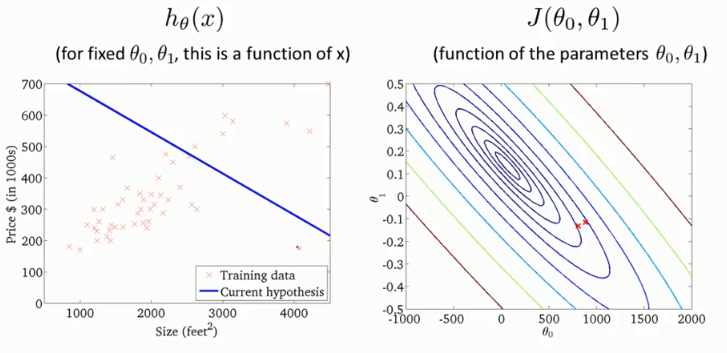


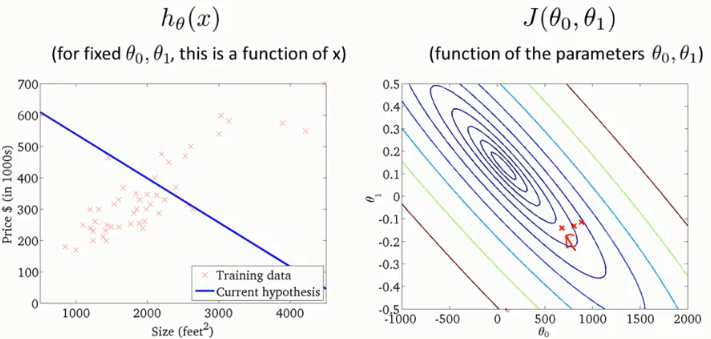
The first term is the partial derivative with respect to Theta 0 and the second term (in pink) is the partial derivative with respect to Theta 1.



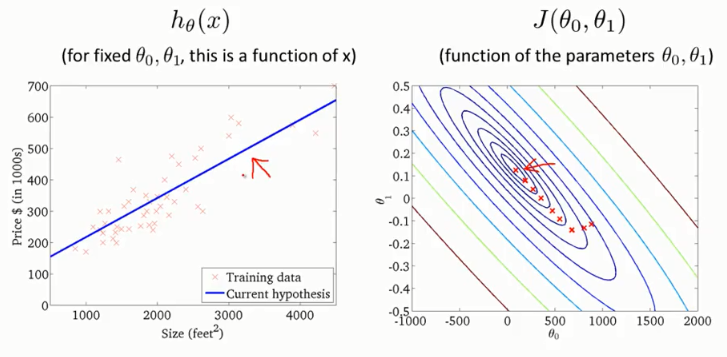
Example of Gradient Descent in action:

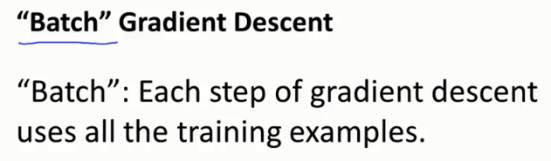


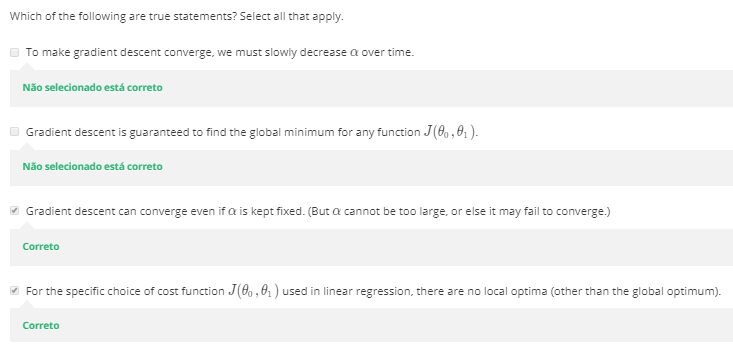




Until...







With advanced Linear Algebra, there exists a solution for numerically solving for the minimum of the cost function j without needing to use an iterative algorithm like gradiente descent. It’s called the **normal equations method.** Gradient Descent scales better on large datasets than Normal Equations Method.

* + - 1. **Leitura: Gradient Descent For Linear Regression**
    1. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Linear Regression with One Variable**
  1. **LINEAR ALGEBRA REVIEW**

This optional module provides a refresher on linear algebra concepts. Basic understanding of linear algebra is necessary for the rest of the course, especially as we begin to cover models with multiple variables.

* + 1. **Linear Algebra Review**
       1. **Video: Matrices and Vectors**
       2. **Leitura: Matrices and Vectors**
       3. **Video: Addition and Scalar Multiplication**
       4. **Leitura: Addition and Scalar Multiplication**
       5. **Video: Matrix Vector Multiplication**
       6. **Leitura: Matrix Vector Multiplication**
       7. **Video: Matrix Matrix Multiplication**
       8. **Leitura: Matrix Matrix Multiplication**
       9. **Video: Matrix Multiplication Properties**
       10. **Leitura: Matrix Multiplication Properties**
       11. **Video: Inverse and Transpose**
       12. **Leitura: Inverse and Transpose**
    2. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste para praticar: Linear Algebra**

1. **SEMANA 2**
   1. **LINEAR REGRESSION WITH MULTIPLE VARIABLES**

Welcome to week 2! I hope everyone has been enjoying the course and learning a lot! This week we’re covering linear regression with multiple variables. we’ll show how linear regression can be extended to accommodate multiple input features. We also discuss best practices for implementing linear regression.

We’re also going to go over how to use Octave. You’ll work on programming assignments designed to help you understand how to implement the learning algorithms in practice. To complete the programming assignments, you will need to use Octave or MATLAB.

As always, if you get stuck on the quiz and programming assignment, you should post on the Discussions to ask for help. (And if you finish early, I hope you'll go there to help your fellow classmates as well.)

* + 1. **Environment Setup Instructions**
       1. **Leitura: Setting Up Your Programming Assignment Environment**
       2. **Leitura: Acessing MATLAB Online and Uploading the Exercise**
       3. **Leitura: Installing Octave on Windows**
       4. **Installing Octave on GNU/Linux**
       5. **More Octave/MATLAB Resources**
    2. **Multivariate Linear Regression**
       1. **Video: Multiple Features**
       2. **Leitura: Multiple Features**
       3. **Video: Gradient Descent for Multiple Features**
       4. **Leitura: Gradient Descent for Multiple Features**
       5. **Video: Gradient Descent in Practice I – Feature Scaling**
       6. **Lecture: Gradient Descent in Practice I – Feature Scaling**
       7. **Video: Gradient Descent in Practice II – Learning Rate**
       8. **Lecture: Gradient Descent in Practice II – Learning Rate**
       9. **Video: Features and Polynomial Regression**
       10. **Lecture: Features and Polynomial Regression**
    3. **Computing Parameters Analytically**
       1. **Video: Normal Equation**
       2. **Lecture: Normal Equation**
       3. **Video: Normal Equation Noninvertibility**
       4. **Lecture: Normal Equation Noninvertibility**
    4. **Submitting Programming Assignments**
       1. **Video: Working on and Submitting Programming Assignments**
       2. **Lecture: Programming Tips from Mentors**
    5. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Linear Regression with Multiple Variables**
  1. **OCTAVE/MATLAB TUTORIAL**

This course includes programming assignments designed to help you understand how to implement the learning algorithms in practice. To complete the programming assignments, you will need to use Octave or MATLAB. This module introduces Octave/Matlab and shows you how to submit an assignment.

* + 1. **Video: Basic Operations**
    2. **Video: Moving Data Around**
    3. **Video: Computing on Data**
    4. **Video: Plotting Data**
    5. **Video: Control Statements: for, while, if statement**
    6. **Video: Vectorization**
  1. **Review**
     1. **Leitura: Lecture Slides**
     2. **Teste: Octave/Matlab Tutorial**
     3. **Tarefa de Programação: Linear Regression**

1. **SEMANA 3**
   1. **LOGISTIC REGRESSION**

Welcome to week 3! This week, we’ll be covering logistic regression. Logistic regression is a method for classifying data into discrete outcomes. For example, we might use logistic regression to classify an email as spam or not spam. In this module, we introduce the notion of classification, the cost function for logistic regression, and the application of logistic regression to multi-class classification.

We are also covering regularization. Machine learning models need to generalize well to new examples that the model has not seen in practice. We’ll introduce regularization, which helps prevent models from overfitting the training data.

As always, if you get stuck on the quiz and programming assignment, you should post on the Discussions to ask for help. (And if you finish early, I hope you'll go there to help your fellow classmates as well.)

* + 1. **Classification and Representation**
       1. **Video: Classification**
       2. **Leitura: Classification**
       3. **Video: Hypothesis Representation**
       4. **Leitura: Hypothesis Representation**
       5. **Video: Decision Boundary**
       6. **Leitura: Decision Boundary**
    2. **Logistic Regression Model**
       1. **Video: Cost Function**
       2. **Leitura: Cost Function**
       3. **Vídeo: Simplified Cost Function and Gradient Descent**
       4. **Leitura: Simplified Cost Function and Gradient Descent**
       5. **Video: Advanced Optimization**
       6. **Leitura: Advanced Optimization**
    3. **Multiclass Classification**
       1. **Video: Multiclass Classification: One-vs-all**
       2. **Leitura: Multiclass Classification: One-vs-all**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Logistic Regression**
  1. **REGULARIZATION**

Machine learning models need to generalize well to new examples that the model has not seen in practice. In this module, we introduce regularization, which helps prevent models from overfitting the training data.

* + 1. **Solving the Problem of Overfitting**
       1. **Video: The Problem of Overfitting**
       2. **Leitura: The Problem of Overfitting**
       3. **Video: Cost Function**
       4. **Leitura: Cost Function**
       5. **Video: Regularized Linear Regression**
       6. **Leitura: Regularized Linear Regression**
       7. **Video: Regularized Logistic Regression**
       8. **Leitura: Regularized Logistic Regression**
    2. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Regularization**
       3. **Tarefa de Programação: Logistic Regression**

1. **SEMANA 4**
   1. **NEURAL NETWORKS: REPRESENTATION**

Welcome to week 4! This week, we are covering neural networks. Neural networks is a model inspired by how the brain works. It is widely used today in many applications: when your phone interprets and understand your voice commands, it is likely that a neural network is helping to understand your speech; when you cash a check, the machines that automatically read the digits also use neural networks.

* + 1. **Motivations**
       1. **Video: Non-linear Hypothesis**
       2. **Video: Neurons and the Brain**
    2. **Neural Networks**
       1. **Video: Model Representation I**
       2. **Leitura: Model Representation I**
       3. **Video: Model Representation II**
       4. **Leitura: Model Representation II**
    3. **Applications**
       1. **Video: Examples and Intuitions I**
       2. **Leitura: Examples and Intuitions I**
       3. **Video: Examples and Intuitions II**
       4. **Leitura: Examples and Intuitions II**
       5. **Video: Multiclass Classification**
       6. **Leitura: Multiclass Classification**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Neural Networks: Representation**
       3. **Tarefa de Programação: Multi-class Classification and Neural Networks**

1. **SEMANA 5**
   1. **NEURAL NETWORKS: LEARNING**

In Week 5, you will be learning how to train Neural Networks. The Neural Network is one of the most powerful learning algorithms (when a linear classifier doesn't work, this is what I usually turn to), and this week's videos explain the 'backpropagation' algorithm for training these models. In this week's programming assignment, you'll also get to implement this algorithm and see it work for yourself.

The Neural Network programming exercise will be one of the more challenging ones of this class. So please start early and do leave extra time to get it done, and I hope you'll stick with it until you get it to work! As always, if you get stuck on the quiz and programming assignment, you should post on the Discussions to ask for help. (And if you finish early, I hope you'll go there to help your fellow classmates as well.)

* + 1. **Cost Function and Backpropagation**
       1. **Video: Cost Function**
       2. **Leitura: Cost Function**
       3. **Video: Backpropagation Algorithm**
       4. **Leitura: Backpropagation Algorithm**
       5. **Video: Backpropagation Intuition**
       6. **Leitura: Backpropagation Intuition**
    2. **Backpropagation in Practice**
       1. **Video: Implementation Note: Unrolling Parameters**
       2. **Leitura: Implementation Note: Unrolling Parameters**
       3. **Video: Gradient Checking**
       4. **Leitura: Gradient Checking**
       5. **Video: Random Initialization**
       6. **Leitura: Random Initialization**
       7. **Video: Putting it Together**
       8. **Leitura: Putting it Together**
    3. **Application of Neural Networks**
       1. **Video: Autonomous Driving**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Neural Networks: Learning**
       3. **Tarefa de Programação: Neural Networks Learning**

1. **SEMANA 6**
   1. **ADVICE FOR APPLYING MACHINE LEARNING**

In Week 6, you will be learning about systematically improving your learning algorithm. The videos for this week will teach you how to tell when a learning algorithm is doing poorly, and describe the 'best practices' for how to 'debug' your learning algorithm and go about improving its performance.

We will also be covering machine learning system design. To optimize a machine learning algorithm, you’ll need to first understand where the biggest improvements can be made. In these lessons, we discuss how to understand the performance of a machine learning system with multiple parts, and also how to deal with skewed data.

When you're applying machine learning to real problems, a solid grasp of this week's content will easily save you a large amount of work.

* + 1. **Evaluating a Learning Algorithm**
       1. **Video: Deciding What to Try Next**
       2. **Video: Evaluating a Hypothesis**
       3. **Leitura: Evaluating a Hypothesis**
       4. **Video: Model Selection and Train/Validation/Test Sets**
       5. **Leitura: Model Selection and Train/Validation/Test Sets**
    2. **Bias vs. Variance**
       1. **Video: Diagnosing Bias vs. Variance**
       2. **Leitura: Diagnosing Bias vs. Variance**
       3. **Video: Regularization and Bias/Variance**
       4. **Leitura: Regularization and Bias/Variance**
       5. **Video: Learning Curves**
       6. **Leitura: Learning Curves**
       7. **Video: Deciding What to Do Next Revisited**
       8. **Leitura: Deciding What to Do Next Revisited**
    3. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Advice for Applying Machine Learning**
       3. **Tarefa de Programação: Regularized Linear Regression and Bias/Variance**

1. **SEMANA 7**
   1. **SUPPORT VECTOR MACHINES**

Welcome to week 7! This week, you will be learning about the support vector machine (SVM) algorithm. SVMs are considered by many to be the most powerful 'black box' learning algorithm, and by posing a cleverly-chosen optimization objective, one of the most widely used learning algorithms today.

As always, if you get stuck on the quiz and programming assignment, you should post on the Discussions to ask for help. (And if you finish early, I hope you'll go there to help your fellow classmates as well.)

* + 1. **Large Margin Classification**
       1. **Video: Optimization Objective**
       2. **Video: Large Margin Intuition**
       3. **Video: Mathematics Behind Large Margin Classification**
    2. **Kernels**
       1. **Video: Kernels I**
       2. **Video: Kernels II**
    3. **SVMs in Practice**
       1. **Video: Using na SVM**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Support Vector Machines**
       3. **Tarefa de Programação: Support Vector Machines**

1. **SEMANA 8**
   1. **UNSUPERVISED LEARNING**

Hello all! I hope everyone has been enjoying the course and learning a lot! This week, you will be learning about unsupervised learning. While supervised learning algorithms need labeled examples (x,y), unsupervised learning algorithms need only the input (x). You will learn about clustering—which is used for market segmentation, text summarization, among many other applications.

We will also be introducing Principal Components Analysis, which is used to speed up learning algorithms, and is sometimes incredibly useful for visualizing and helping you to understand your data.

As always, if you get stuck on the quiz and programming assignment, you should post on the Discussions to ask for help. (And if you finish early, I hope you'll go there to help your fellow classmates as well.)

* + 1. **Clustering**
       1. **Video: Unsupervised Learning: Introduction**
       2. **Video: K-Means Algorithm**
       3. **Video: Optimization Objective**
       4. **Video: Random Initialization**
       5. **Video: Choosing the Number of Clusters**
    2. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Unsupervised Learning**
  1. **DIMENSIONALITY REDUCTION**

In this module, we introduce Principal Components Analysis, and show how it can be used for data compression to speed up learning algorithms as well as for visualizations of complex datasets.

* + 1. **Motivation**
       1. **Motivation I: Data Compression**
       2. **Motivation II: Visualization**
    2. **Principal Component Analysis**
       1. **Video: Principal Component Analysis Problem Formulation**
       2. **Video: Principal Component Analysis Algorithm**
    3. **Applying PCA**
       1. **Video: Reconstruction from Compressed Representation**
       2. **Video: Choosing the Number of Principal Components**
       3. **Video: Advice for Applying PCA**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Principal Component Analysis**
       3. **Tarefa de Programação: K-Means Clustering and PCA**

1. **SEMANA 9**
   1. **ANOMALY DETECTION**

Hello all! I hope everyone has been enjoying the course and learning a lot! This week, we will be covering anomaly detection which is widely used in fraud detection (e.g. ‘has this credit card been stolen?’). Given a large number of data points, we may sometimes want to figure out which ones vary significantly from the average. For example, in manufacturing, we may want to detect defects or anomalies. We show how a dataset can be modeled using a Gaussian distribution, and how the model can be used for anomaly detection.

We will also be covering recommender systems, which are used by companies like Amazon, Netflix and Apple to recommend products to their users. Recommender systems look at patterns of activities between different users and different products to produce these recommendations. In these lessons, we introduce recommender algorithms such as the collaborative filtering algorithm and low-rank matrix factorization.

As always, if you get stuck on the quiz and programming assignment, you should post on the Discussions to ask for help. (And if you finish early, I hope you'll go there to help your fellow classmates as well.)

* + 1. **Density Estimation**
       1. **Video: Problem Motivation**
       2. **Video: Gaussian Distribution**
       3. **Video: Algorithm**
    2. **Building an Anomaly Detection System**
       1. **Video: Developing and Evaluating na Anomaly Detection System**
       2. **Video: Anomaly Detection vs. Supervised Learning**
       3. **Choosing What Features to Use**
    3. **Multivariate Gaussian Distribution (Optional)**
       1. **Video: Multivariate Gaussian Distribution**
       2. **Video: Anomaly Detection using the Multivariate Gaussian Distribution**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Anomaly Detection**
  1. **RECOMMENDER SYSTEMS**

When you buy a product online, most websites automatically recommend other products that you may like. Recommender systems look at patterns of activities between different users and different products to produce these recommendations. In this module, we introduce recommender algorithms such as the collaborative filtering algorithm and low-rank matrix factorization.

* + 1. **Predicting Movies Ratings**
       1. **Video: Problem Formulation**
       2. **Video: Content Based Recommendations**
    2. **Collaborative Filtering**
       1. **Video: Collaborative Filtering**
       2. **Video: Collaborative Filtering Algorithm**
    3. **Low Rank Matrix Factorization**
       1. **Video: Vectorization: Low Rank Matrix Factorization**
       2. **Video: Implementational Detail: Mean Normalization**
    4. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Recommender Systems**
       3. **Tarefa de Programação: Anomaly Detection and Recommender Systems**

1. **SEMANA 10**
   1. **LARGE SCALE MACHINE LEARNING**

Welcome to week 10! This week, we will be covering large scale machine learning. Machine learning works best when there is an abundance of data to leverage for training. With the amount data that many websites/companies are gathering today, knowing how to handle ‘big data’ is one of the most sought after skills in Silicon Valley.

* + 1. **Gradient Descent with Large Datasets**
       1. **Video: Learning with Large Datasets**
       2. **Video: Stochastic Gradient Descent**
       3. **Video: Mini-Batch Gradient Descent**
       4. **Video: Stochastic Gradient Descent Convergence**
    2. **Advanced Topics**
       1. **Video: Online Learning**
       2. **Video: Map Reduce and Data Parallelism**
    3. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Large Scale Machine Learning**

1. **SEMANA 11**
   1. **APPLICATION EXAMPLE: PHOTO OCR**

Congratulations on making it to the eleventh and final week! This week, we will walk you through a complex, end-to-end application of machine learning, to the application of Photo OCR. Identifying and recognizing objects, words, and digits in an image is a challenging task. We discuss how a pipeline can be built to tackle this problem and how to analyze and improve the performance of such a system.

* + 1. **Photo OCR**
       1. **Video: Problem Description and Pipeline**
       2. **Video: Sliding Windows**
       3. **Video: Getting Lots of Data and Artificial Data**
       4. **Video: Ceiling Analysis: What Part of the Pipeline to Work on Next**
    2. **Review**
       1. **Leitura: Lecture Slides**
       2. **Teste: Application: Photo OCR**
    3. **Conclusion**
       1. **Video: Summary and Thank You**