Stanford Machine Learning Notes

Xiaolong Zhang

2015-01-16 Fri

Info

URL is `https://class.coursera.org/ml-005/lecture`

Introduciton

What is machine learning

- Arthur Samuel (1959): **Field of study of that gives computers the ability to learn without being explicitly programmed.**
- Tom Mitchel (1998): A Well-posed Learning Problem is *A computer program is said to learn from experience E with repsect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine learning algorithms:

We mainly talks about **two types of algorithm**:

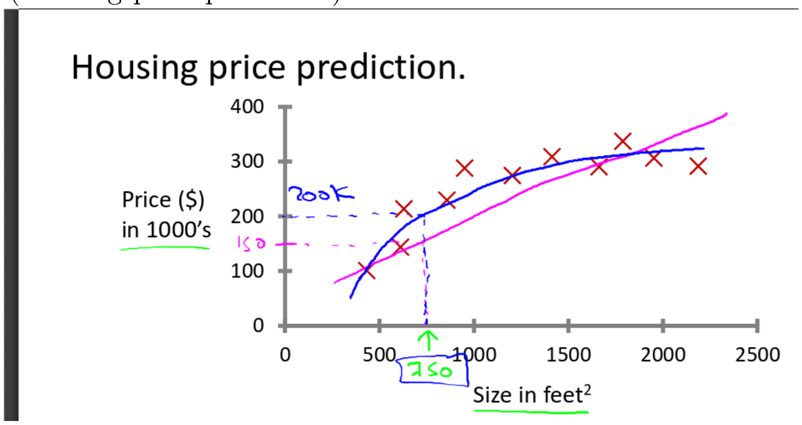
- Supervised Learning
- Unsupervised Learning
- Others: Reinforcement Learning, Recommender System.

And **Practical advice for applying learning algorithm**

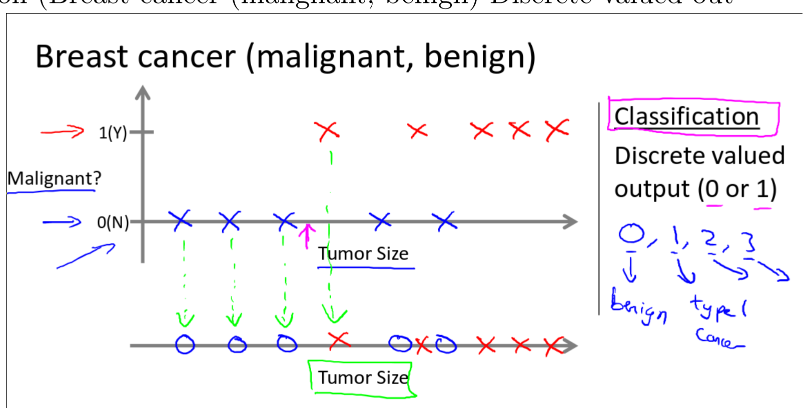
Supervised Learning

"Right Answers" are given.

- Regression (Housing price prediction) Predict continuous valued output (price)



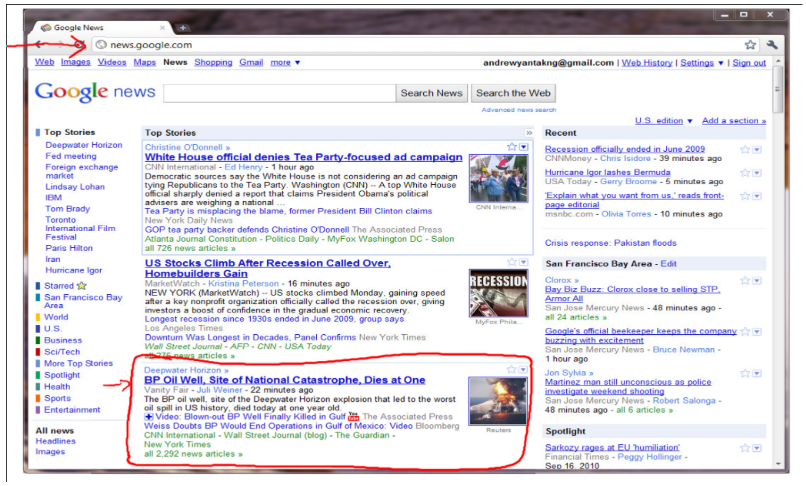
- Classification (Breast cancer (malignant, benign) Discrete valued output (0 or 1)



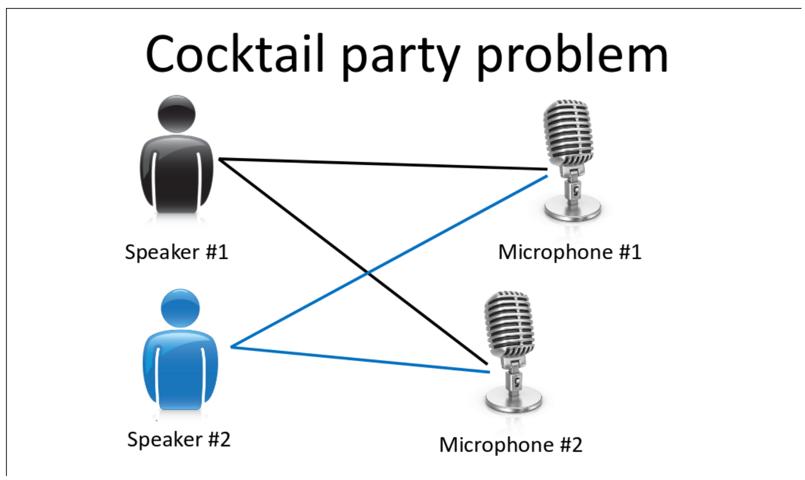
Unsupervised Learning

No knowledge is given.

- Google News Grouping



- Cocktail party problem



Linear Regression

Model Representation

Notations:

1. m is Number of training examples.
2. x's is "input" variable / feature.
3. y's is "output" variable / "target" variables.
4. (x, y) is one training example.

5. $(x^{(i)}, y^{(i)})$ is the ith example, i starts from 0.

6. h is called hypothesis, it maps the input to output. In this lecture, we represent h using linear function, thus it's called **linear regression**. For linear regression with one variable, it's called **univariate linear regression**. For example, the **univariate linear regression** is usually written as: $h_\theta(x) = \theta_0 + \theta_1 x$. The θ here represents the coefficient variables. Sometimes it's written as $h(x)$ as a shorthand.

Cost Function

Since the hypothesis is written as $h_\theta(x) = \theta_0 + \theta_1 x$, where θ_i's are the parameters, then how to choose θ_i's? The idea is to choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y). More formally, we want to

$$\underset{\theta_0,\theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

, where m is the number of trainnig examples. To recap, we're minimizing half of the averaging error. Note that the variable here are θs, and x and y's are constants.

By convention, we define the **cost function** $J(\theta_0, \theta_1)$ to represent the objective function. That is

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\underset{\theta_0,\theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

This cost function is also called **squared error function**. There are other cost functions, but it turns out that squared error function is a resonable choice and will work for most of regression problem.

Cost Function Intuition

Before getting a intuition about the cost function, let's have a recap, we now have

1. Hypothesis:
$$h_\theta(x) = \theta_0 + \theta_1 x$$

2. Parameters:
$$\theta_0, \theta_1$$

3. Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

4. Goal:

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

In order to visualize our cost function, we use a simplified hypothesis function: $h_\theta(x) = \theta_1 x$, which sets θ_0 to 0. So now we have

1. Hypothesis:

$$h_\theta(x) = \theta_1 x$$

2. Parameters:

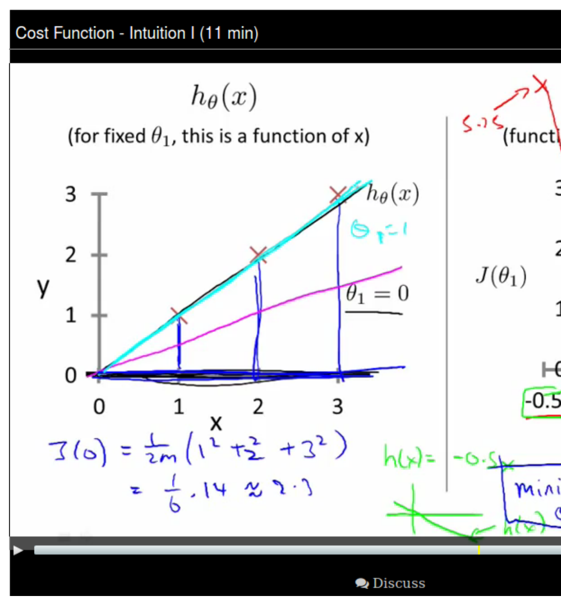
$$\theta_1$$

3. Cost Function:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

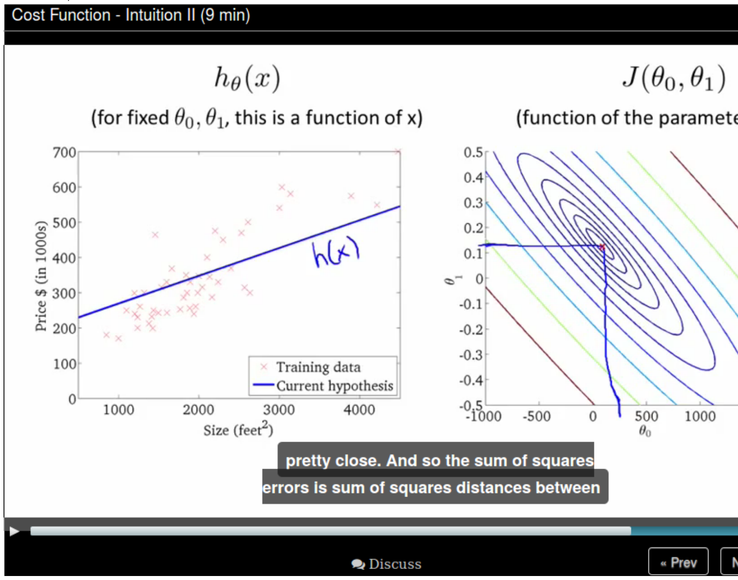
4. Goal:

$$\underset{\theta_1}{\text{minimize}} J(\theta_1)$$



So now let's compare function $h_\theta(x)$ and function $J(\theta_1)$:

Then let's come back to the original function, where we don't have the constrain that $\theta_0 = 0$. The comparison is like

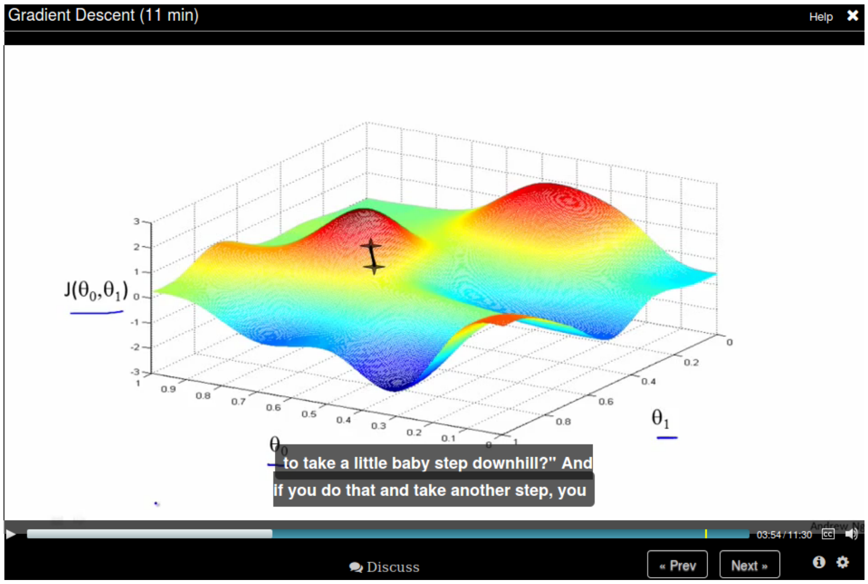


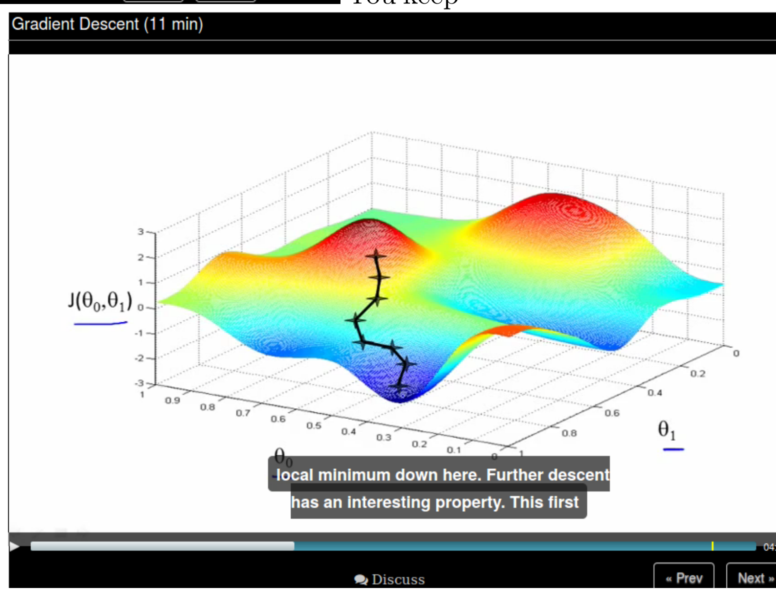
Gradient Descent

Now we have some function $J(\theta_0, \theta_1)$ and we want to $\underset{\theta_0,\theta_1}{\text{minimize}} J(\theta_0, \theta_1)$, we use **gradient descent** here, which

1. Start with some θ_0, θ_1,
2. Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$, until we hopefully end up at a minimum.

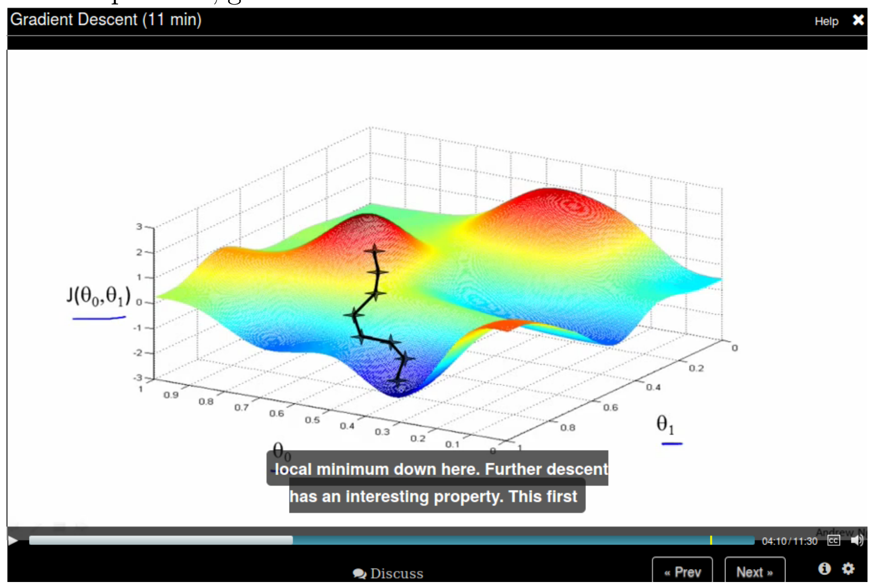
To help understand gradient descent, suppose you are standing at one point on the hill, and you want to take a small step to step downhill as quickly as possible, then you would choose the deepest direction to downhill.



You keep


doing this until to get to a local minimum.

But if you start with a different initial position, gradient descent will take



you to a (very) different position.

Gradient Descent algorithm

We use $a := b$ to represent **assignment** and $a = b$ to represent **truth assertion**.

repeat

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ {for $j = 0$ and $j = 1$}

until convergence

The α here is called learning rate.

Pay attention that when implementing gradient descent, we need to update all θs simultaneous.