My Notes to Andrew Ng's Coursera "Machine Learning"

Eric Xia

Last Updated 5 September 2020

Contents

1 Introduction **2**

2 Univariate Linear Regression **3**

3 Linear Algebra Review **6**

4 Multivariate Linear Regression **8**

5 Octave / Matlab Tutorial **8**

6 Logistic Regression **8**

7 Regularization **8**

8 Neural Networks: Representation **8**

9 Neural Networks: Learning **8**

10 Advice for Applying Machine Learning **8**

11 Machine Learning System Design **8**

12 Support Vector Machines (SVMs) **8**

13 Unsupervised Learning **8**

14 Dimensionality Reduction **8**

15 Anomaly Detection **8**

16 Recommender Systems **8**

17 Large Scale Machine Learning **8**

18 Application Example: Photo OCR **8**

1 Introduction

Machine Learning (ML): the field of study that gives computers the ability to learn without being explicitly programmed
Learning Problems: a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

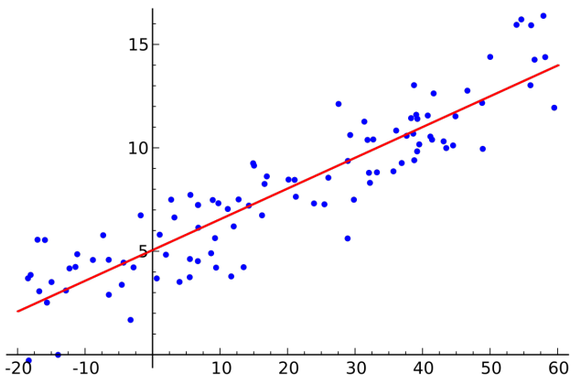
Example: Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. Identify the experience E, task T, and performance measure P in this scenario.
T: Classifying emails as spam or not spam.
E: Watching you label emails as spam or not spam.
P: The number (or fraction) of emails correctly classified as spam/not spam.

Supervised Learning: we give the computer a dataset and the right/wrong answers and then attempt to produce more correct answers.
Unsupervised Learning: we approach problems with little to no idea of what our results should look like and we derive structure from data where we don't necessarily know the effect of the variables; we let the computer learn itself, e.g. organizing computing clusters, social network analysis, market segmentation, astronomical data analysis.

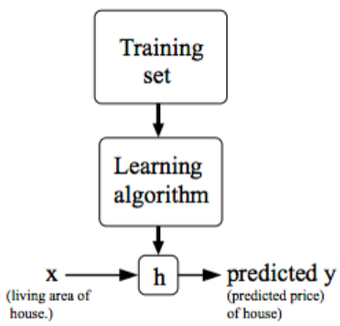
Clustering: finding natural groups in the feature space of input data
Regression: attempting to predict continuous values
Classification: attemping to predict discrete-valued output or map input variables into discrete categories, e.g. Benign vs. Malignant Tumor

2 Univariate Linear Regression

Training Set: the dataset used to train a model
$\boldsymbol{m}$: number of training examples
$\boldsymbol{x}$'s: input variables/features
$\boldsymbol{y}$'s: output variables/features
$(\boldsymbol{x}, \boldsymbol{y})$: one training example
$(\boldsymbol{x^{(i)}}, \boldsymbol{y^{(i)}})$: the ith training example
Hypothesis Function $(h : x \to y, h(x) = h_\theta(x) = \theta_0 + \theta_1 x)$: given a training set, we want to choose **parameters** (θ_0 **and** θ_1)such that the mean distance between $h(x)$ and the training examples are as close to the true mean distance:



Linear Regression



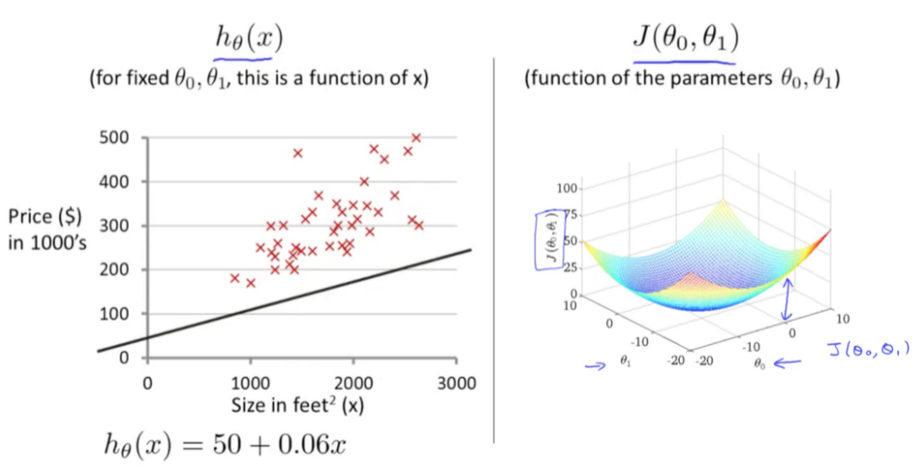
Hypothesis Function

We can measure the error of a hypothesis function by using a **cost function** J, given by

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

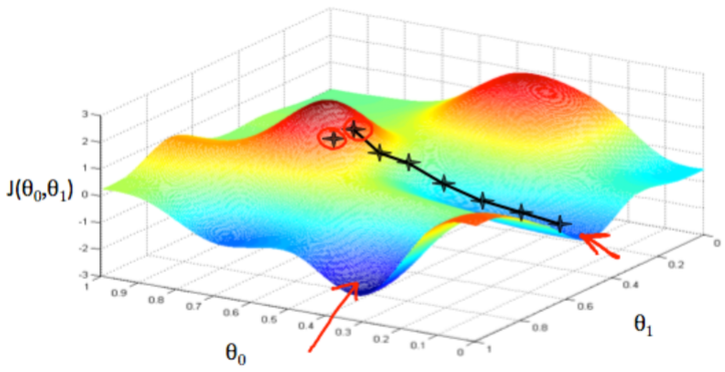
This function is really just the average of the differences between the produced outputs with the hypothesis function being tested, and the actual outputs. The cost function is also referred to as the squared error function. We half the mean in the cost function out of convenience for computing the gradient descent, since the derivative term of the square function will cancel out the half.

The contour plot display of the cost function for a particular hypothesis function:



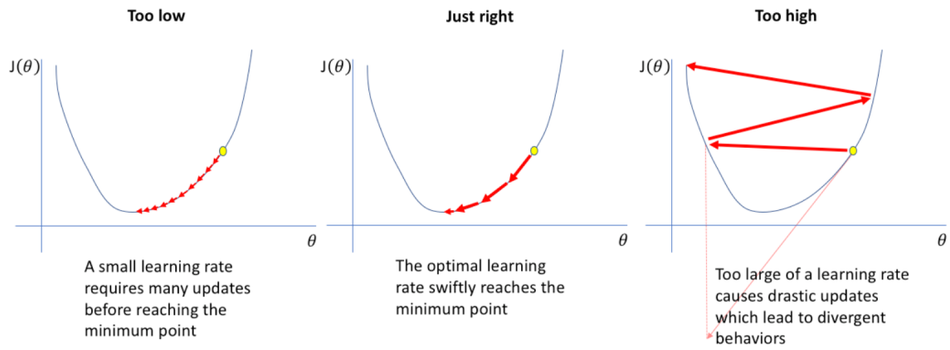
Gradient Descent: an optimization algorithm used to minimize some function, e.g. the cost function, by iteratively moving in the direction of the steepest descent as defined by the negative of the gradient.

Putting θ_0 on the x-axis and θ_1 on the y-axis, with the cost function on the z-axis, the points on the graph are the outputs of our cost function. We know we have succeeded in finding parameters for $J(\theta_0, \theta_1)$ that minimize the cost when our gradient descent leads us to the blue area of the graph (lowest z-value). The red arrows on the graph show the minimum points on the graph.



We implement this algorithm by taking the derivative of the cost function. This gives us the slope of the cost function at a point, and a direction to move towards. We then iteratively step down the cost function in the direction of the steepest descent. The size of each step is determined by the **learning rate**, the parameter α. Note that α is fixed and its value need not variate because the cost function is parabolic, hence as we approach the minimum, the derivative decreases in magnitude and flattens out.

We want to choose a value of α that is not too small or too large:

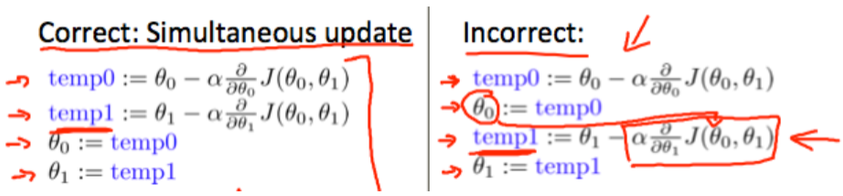


Then our implementation of gradient descent is given by repeating the following equation until convergence.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

where $j = 0, 1$ represents the index and ":=" is the assignment operator.

Note that the gradient descent equations for θ_0 and θ_1 should be updated simultaneously, such that updating of θ_0 does not interfere with the accurate updating of θ_1:



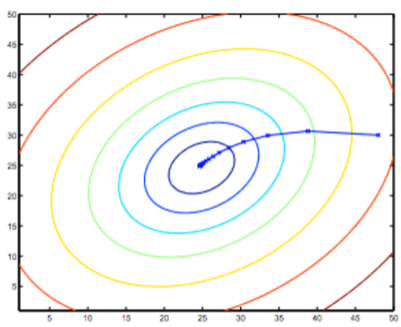
The current gradient descent algorithm we are using is called **batch gradient descent** because it accounts for every training example to determine the descent steps. For a single training example,

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y) x_j
\end{aligned}$$

Hence, we repeat the following equations until convergence.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y) x_i$$

While gradient descent can be susceptible to local minima, simple optimization problems for linear regression only have one global, and no other local or optima. Thus, gradient descent will always converge here to the global minimum. Assuming we have a well-chosen α, we see that J is indeed a convex quadratic function, and the below contour plot depicts the trajectory of an implementation of gradient descent to minimize a quadratic function. The points start from the warm outermost colors and move towards the cooler innermost colors, until they converge at the global minimum of the cost function.



3 Linear Algebra Review

Matrix: a 2D array
- The dimension of a matrix is given by "number of rows × number of columns"
- Dimension may be written $\mathbb{R}^{4\times 2}$, representing a 4×2 matrix of the real-valued set $\mathbb{R}$
- In the declaration below, A_{iy} = "i, j entry" in the ith row, jth column

$$A = \begin{bmatrix} 1402 & 191 \\ 1371 & 821 \\ 949 & 1437 \\ 147 & 1448 \end{bmatrix}$$

Vector: A $n \times 1$ matrix
- Can be 1-indexed or 0-indexed, where 1-indexed is more conventional
- It is conventional to name matrices with capital letters and vectors with lowercase letters
- In the declaration below, $y_i = i$th element of the given 4D Vector, $\mathbb{R}^4$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

Matrix Addition: a defined matrix sum must have two addends with the same dimensions.

$$\begin{bmatrix} a_1 & d_1 \\ b_1 & e_1 \\ c_1 & f_1 \end{bmatrix} + \begin{bmatrix} a_2 & d_2 \\ b_2 & e_2 \\ c_2 & f_2 \end{bmatrix} = \begin{bmatrix} a_1 + a_2 & = d_1 + d_2 \\ b_1 + b_2 & = e_1 + e_2 \\ c_1 + c_2 & = f_1 + f_2 \end{bmatrix}$$

Scalar Matrix Multiplication:

$$C \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix} = \begin{bmatrix} Ca & Cd \\ Cb & Ce \\ Cc & Cf \end{bmatrix}$$

Division example:

$$\begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} /4 = \frac{1}{4} \begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3/2 & 3/4 \end{bmatrix}$$

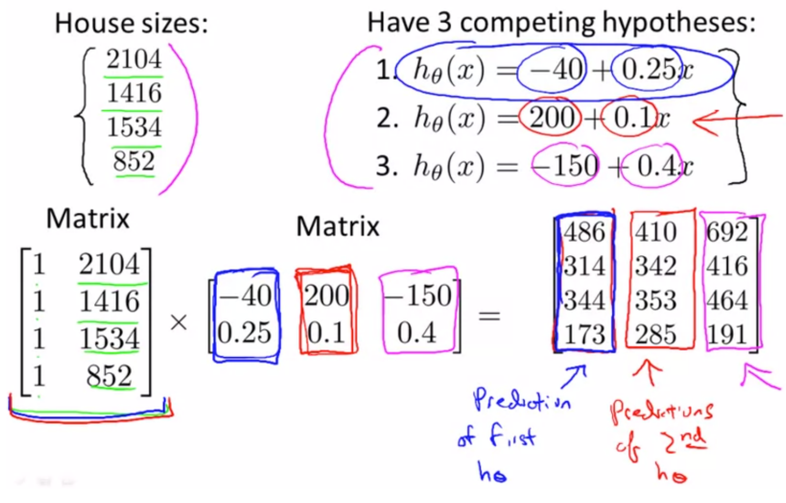
Matrix-Vector Multiplication: if A is a $m \times n$ matrix then the product Ax is defined by $n \times 1$ column vectors x. If $Ax = b$ then b is a $m \times 1$ column vector.

$$\text{Ax} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

Example:

$$\begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & -1 \cdot 1 & 0 \cdot 2 \\ 2 \cdot 0 & -1 \cdot 3 & 0 \cdot 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

Matrix-Matrix Multiplication to Combine Hypotheses: If we wanted to make a hypothesis from 3 hypotheses we could represent the hypotheses by matrices and the dataset by another matrix and multiply them to get the product matrix which would be the combined hypothesis:



Properties of Matrix-Matrix Multiplication:
• In general, $A \times B \neq B \times A$ (not commutative)
• $A \times (B \times C) = (A \times B) \times C$ (is associative)

Identity Matrix: a special square matrix that has 1's along the main diagonal (upper left to lower right) and 0's for all other elements
• Denoted I or I_n
• For any matrix $A, A_{m\times n} \cdot I_{n\times n} = I_{m\times m} \cdot A_{m\times n} = A_{m\times n}$ (the subscripts are usually omitted)
• $A = B$ when $B = I$ (matrices are commutative when at least one of them is the identity matrix)

Example of an identity matrix:

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Informally,

Matrix Inverse: If A is a $m \times m$ matrix, and if it has an inverse (only square matrices have inverses), then $A(A^{-1}) = A^{-1}A = I$

Matrix Transpose: If A is a $m \times n$ matrix, $B = A^T$ then B is a $n \times m$ matrix, and $B_{ij} = A_{ij}$
• Basically, the rows of A become the columns of B and the columns of A become the rows of B

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 5 & 6 \\ 7 & 0 & 9 \end{bmatrix}, A^T = \begin{bmatrix} 1 & 0 & 7 \\ 2 & 5 & 0 \\ 0 & 6 & 9 \end{bmatrix}$$

4 Multivariate Linear Regression

5 Octave / Matlab Tutorial

6 Logistic Regression

7 Regularization

8 Neural Networks: Representation

9 Neural Networks: Learning

10 Advice for Applying Machine Learning

11 Machine Learning System Design

12 Support Vector Machines (SVMs)

13 Unsupervised Learning

14 Dimensionality Reduction

15 Anomaly Detection

16 Recommender Systems

17 Large Scale Machine Learning

18 Application Example: Photo OCR