CS 4758/6758: Robot Learning

Ashutosh Saxena
Cornell University

1 Supervised Learning

Sometimes we have a number of sensors (e.g., with output x_1 and x_2) and we want to combine them to get a better estimate (say y). One method is to combine them linearly. I.e.,

$$y = \theta_1 x_1 + \theta_2 x_2 + \theta_0 \tag{1}$$

However, we do not know what weights θ_i we should use. In this setting we can use linear regression, where we are given a training set to learn the weights from.

1.1 Notation

To establish notation for future use, well 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. A pair $(x^{(i)}, y^{(i)})$ is called a training example, and the dataset that well 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.

2 Linear Regression

To perform supervised learning, we must decide how were going to rep- resent functions/hypotheses h in a computer. As an initial choice, lets say we decide to approximate y as a linear function of x:

$h_\theta(x) = \theta_0 + \theta_1 x1 + \theta_2 x2$

Here, the θ_is are the parameters (also called weights) parameterizing the space of linear functions mapping from X to Y .

To simplify our notation, we also introduce the convention of letting $x_0 = 1$ (this is the intercept term), so that

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x, \tag{2}$$

where on the right-hand side above we are viewing θ and x both as vectors, and here n is the number of input variables (not counting x_0). Now, given a training set, how do we pick, or learn, the parameters θ? One reasonable method seems to be to make h(x) close to y , at least for the training examples we have. To formalize this, we will dene a function that measures, for each value of the θs, how close the $h(x^{(i)})$s are to the corresponding $y^{(i)}$'s. We dene the cost function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{3}$$

If youve seen linear regression before, you may recognize this as the familiar least-squares cost function that gives rise to the ordinary least squares regression model. Whether or not you have seen it previously, lets keep going, and well eventually show this to be a special case of a much broader family of algorithms.

2.1 Gradient Descent

One way to obtain the optimal value of θ is to use gradient descent. It starts with some initial θ, and repeatedly performs the update:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{4}$$

This update is simultaneously performed for all values of $j = 0, ..., n$.

2.2 Normal equations

Another method to obtain the parameters is by using normal equations. The closed form solution is given by:

$$\theta = (X^T X)^{-1} X^T y \tag{5}$$

where X is a matrix containing all the features, with each row a datapoint in the tranining set, and y is the vectorized form of the training set labels.

3 Acknowledgements

Parts of this text were taken from CS229 lecture notes by Andrew Ng. For more in-depth coverage of this material, please visit: http://www.cs.cornell.edu/Courses/CS6780/2009fa/materials/lecture2.pdf