Coursera Machine Learning

Andrew Ng

January 14, 2017

Contents

Chapter 1

Introduction to Machine Learning

Machine learning is the science of getting computers to learn, without being explicitly programmed. It is used in

- database mining
- applications that can't be programmed by hand (e.g. autonomous piloting, NLP, computer vision)
- self-customizing programs
- understanding human learning

Supervised learning involves learning from a given data set where r̈ightänswers are given.

- regression - determining a continuous value of output
- classification - determine a discrete value of output

Unsupervised learning involves providing the machine a set of data points that the machine must group into cohesive, related sets or groups. The machine must find patterns and structure in the data.

- analyzing social networks to determine social groups
- dividing market into segments in market research
- organizing astronomical data

Note: Originally part of Lesson One

Chapter 2

Lesson One: Linear Regression with One Variable

2.1 Linear Regression Equation

Linear regression

- the data set used to determine the regression line (the test data) is called the 'training set'
- the hypothesis refers to the function that takes the given input and returns the estimated output (in this case the linear regression equation

The linear regression function takes the form: $h_\theta(x) = \theta_0 + \theta_1 * x$ where *theta* is the parameters of the function.

2.2 Cost Functions

A cost function evaluates the 'error' of a hypothesis, that is how far the hypothesis' expected output differs from the true answer. The goal is to minimize the cost in order to minimize the deviation of individual points from the expected output.

The square error cost function is one of the common kinds of cost functions, shown below:
$J_\theta(\theta_0, \theta_1) = \frac{1}{2m} \sum\limits_{i=1}^{m} (h_\theta(x^i) - y^i)^2, minimize J_\theta(\theta_0, \theta_1)$

2.3 Gradient Descent

Gradient descent is a general algorithm used to minimize any function, commonly used in machine learning. It iteratively tests parameter values and adjusts the parameters until the cost function is as minimized as possible. This will always return a local optimum, however there are cases where the process does not necessarily return the global optimum (smallest cost). Fortunately in the case of linear regression which have a convex cost function, gradient descent will always return the global optimum.

repeat until convergence: $\{\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} J_\theta(\theta_0, \theta_1) for j = 0, j = 1\}$

θ are the equation parameters and α is the learning rate (the amount by which the gradient descent progresses each iteration.If the α is too small, the algorithm takes longer to complete while if it is too large, there is a chance that the equation can miss the optimum and potentially diverge. As a minimum approaches, gradient descent takes smaller steps as the derivative term approaches 0. WHen it reaches 0, it is at an optimum.

There are batch and non-batch gradient descents algorithms. Batch algorithms use the entire training set each step while non-batch algorithms only use subsets of data at a time. There are also non iterative methods, such as the ïiormalëquation method but gradient descent works better with large sets of data.

Chapter 3

Lesson Two: Linear Regression with Multiple Variables

3.1 Multivariable Linear Regression Equation

Similar to multivariate linear regression, but with multiple dimensions and parameters. The hypothesis takes the for:

$h_\theta(x) = \theta_0 * x_0 + \theta_1 * x_1 + \theta_2 * x_2 + ... + \theta_n * x_n$ where $x_0 = 1$

If vectorized, the equation can be written:

$h_\theta(x) = \theta * X^T$ where theta is a matrix of coefficients and X^T is the transpose of the matrix of independent variables.

3.2 Gradient Descent Algorithm

repeat until convergence: $\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} J_\theta(\theta_0, \theta_1) for j = 0$

To make sure gradient descent is working properly, it is a good idea to plot the cost against the the number of iterations. The graph should always decrease if gradient descent works properly. When selecting an alpha, it is always best to test a low value for α (around 0.001) and increase it until a reasonably large learning rate is found.

Quadratic functions can also be applied by performing the appropriate operations on the data before being entered into the algorithm. In such cases, feature scaling can become increasingly important to improve calculation time.

3.2.1 Feature Scaling

When one has mulitple features, scaling the values to similar ranges will allow the algorithm to progress more quickly. A good rule of thumb would be to get every feature in the range of (-1,1), a smiliarly small value, or to divide by the standard deviation.

3.2.2 Mean Normalization

Subtracting the mean from the training set to center the distribution of the training set around 0. $x_i = x_i - \mu$

3.3 Normal Function

The normal equation is another way to solve for the minimum in linear regression problems. Gradient descent requires the programmer to select a learning rate and requires multiple iterations to work, however it works well with large sets of data. In comparison, the normal equation does not require a learning rate or any iteration, but the algorithm is $O(n^3)$, which is very inefficient.

Chapter 4

Logistic Regression

Classification problems involve putting data into groups or classes, whether binary or multiclass.

The hypothesis for a logistic regression problem takes to form:
$g_\theta(x) = g(\theta^T * X)$, where $g(z) = \frac{1}{1+e^{-z}}$

This function is called a sigmoid or logistic function. It has two asymptotes at g(z)=0 and g(z)=1. In a binary case, the value of $h_\theta(x)$ can be interpreted as the probability of x matching to 1. $h_\theta(x) = P(y = 1|x; 0)$, or the probability that y=1 given x parameterized by 0.

A **decision boundary** is created where if the sigmoid function returns a value above the threshold, the output will match to 1 while if the function returns a value less than the threshold, the output will match to 0.

4.1 Cost Function

The graph of the squared error cost function applied to the sigmoid function is non convex and possess many local optima, so a different cost function is needed. The logistic regression cost function is as follows:
$J(h_\theta(x), y) = -log(h_\theta(x)) ify = 1, -log(1 - h_\theta(x)) ify = 0$

which can be combined into:
$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} cost(h_\theta(x), y^i) = \frac{1}{m} \sum_{i=1}^{m} [-y^i log(h_\theta(x^i)) - (1 - y^i) log(1 - h_\theta(x))]$

In order to minimize the cost: repeat until convergence $\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} J(\theta) or \theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x) - y^i) x_j^i$

4.2 Ither Options

There are other methods besides gradient descent used to minimize the cost of a function. Given a θ, the algorithms would need to cimpute:

- $J(\theta)$
- $\frac{\delta J(\theta)}{\delta \theta_j} for(j = 0, 1, ..., n)$

Optimization Algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages of Other Algorithms:

- No need to manually pick alpha. A line search algorithm is used to select a learning rate.
- Tend to converge more quickly than gradient descent.

Disadvantages of Other Algorithms:

- Greater Complexity

4.3 Multiclass Classification

Multiclass classifiation, also known as one-versus-all classification or one-versus-rest classification, involves dividing data points into one of multiple classes. Similarly to how dummy variables are used in statistics for multiclass classification, a matrix is used where 1 represents the presence of the class and 0 represents the absence of a class.

$h_\theta^i(x) = P(y = 1|x = 0)(i = 1, 2, 3)$

One vs All

Train a logistic regression classifier $h_\theta^i(x)$ for each class i to predict the probability that $y = i$; On a new input x, to make a prediction, pick the class i that maximizes $max_i h_\theta^i(x)$

Chapter 5

Regularization

5.1 Regression Model Errors

When creating a model, there is a danger of underfitting or overfitting based on the sample data.

- underfitting - a result of the assumption of a high bias that the data varies linearly, as a result the model oversimplifies the model and does not fit the data well
- overfitting - assumes high variance in the data, an overfitted model has too many features and though may specifically match the training set ($J(\theta) \ = 0$), will not generalize well to new examples

To fix underfitting, new features should be added to the model. To fix overfitting, there are two options:

- Reduce the number of features - Manually choose some features to keep and throw away others (model selection algorithm can be used). This has the drawback of potentially throwing away a feature that is useful.
- Regularization - Keeps all features but reduces the magnitude and values of the parameters that are deemed less important. This works well if all parameters have some useful impact on model.

5.2 Regularization

A regularized cost formula follows the form:
$\frac{1}{2m}[\sum_{i=1} m(h_\theta(x^i) - y^i)^2 + \lambda \sum_{i=1}^{m} \theta_j^2]$

$\lambda \sum_{i=1}^{m} \theta_j^2$ is called the regularization term and should be added to all parameters. λ is the regularization parameter. It controls the tradeoff between fitting the model to a training set and reducing the parameters too much. If the regularization parameter is too large, the model will oversimplify and become linear, constant, and underfitted.

5.2.1 Linear Regression

$\frac{1}{2m}[\sum_{i=1} m(h_\theta(x^i) - y^i)^2 + \lambda \sum_{i=1}^{m} \theta_j^2]$
$(\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)x^i)$
$(\theta_j := \theta_j - \alpha \frac{1}{m}[\sum_{i=1}^{m} (h_\theta(x^i) - y^i)x^i + \lambda \frac{1}{m}\theta_j])$
θ_0 is not regularized because there is little to no effect from regularization. The equation for updating the parameters can also be rewritten as:
$\theta_j := \theta_j(1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)x^i$
The parameter is first multiplied by $(1 - \alpha \frac{\lambda}{m})$, which is usually a value slightly less than 1. The second term is the same as ausual. As a result, the parameter decreases a little quicker than usual.

5.2.2 Normal Equation

$x = [x^{1^T} ... x^{m^T}], y = [y^1 ... y^m]$
$\theta = (x^T x)^{-1} x^T y is the original normal equation.$
$\theta = (x^T x + \lambda[\begin{smallmatrix} 0 & 0 \\ \cdot 0 & \cdot 1 \end{smallmatrix}])^{-1} x^T y$ is the regularized form.
$(x^T x + \lambda[\begin{smallmatrix} 0 & 0 \\ \cdot 0 & \cdot 1 \end{smallmatrix}])^{-1}$ is invertible.

5.2.3 Regularized Logistic Regression

Combines gradient descent and advanced optimization methods for logistic regression.
Cost function: $J(\theta) = -[\frac{1}{m} \sum_{i=1}^{m} y^i log(h_\theta(x^i)) + (1 - y^i) log(1 - h_\theta(x^i)) + \lambda \frac{1}{2m} \sum_{j=1}^{n} \theta_j^2 | \theta_1, \theta_2, ..., \theta_n]$
Updates like linear regression:
repeat until convergence:$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{m} (h_\theta(x^i) - y^i)$,
$\theta_j := \theta_j - \alpha[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)x_j^i + \lambda \frac{1}{m} \theta_j]$
which is the partial derivative of the regularized logistic regression cost equation
Reminder: When working with other optimization methods, a cost function that takes the parameter matrix θ and optimizes it to minimize the cost function, use $j_v al$.