Week 1 Notes

Introduction

These notes are for the Coursera Machine Learning course from Stanford University (taught by Andrew Ng).

Math and equations are written using KaTeX notation.

What is Machine Learning?

> "Field of study that gives computers the ability to learn without being explicity programmed" - Arthur Samuel (1959)

> "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**." - Tom Mitchel (1998)

Supervised Learning

"Right answers" are given for a dataset. Example: predict price of a house based on square footage. (regression problem)

Another example: whether or not a tumor is malignant *vs.* tumor size. (classification problem)

Can also use an infinite number of criteria to perform these tasks.

Unsupervised Learning

Answers are not provided to the algorithm.

examples of clustering:

- organize computer clusters
- social network analysis
- market segmentation
- astronomical data analysis

Cocktail party problem

Many people talking at the same time; how to separate the voices of individual speakers? Algorithm takes input of different microphones and outputs single speaker (or separates speaker from ambient noise).

```
[W,s,v] = svd((repmat(sum(x.*x,1),size(x,1),1).*x)\*x);
```

Model and Cost Function

Using a training set of housing prices *vs.* square footage in Portland, OR, develop a linear regression model to predict house price.

```
Training Set --> Learning Algorithm --> h(x)
```

`h` , called the 'hypothesis', maps `x` 's to `y` 's (`area` to `price`)

Linear regression with one variable:

$h_\theta(x) = \theta_0 + \theta_1 x$

Cost Function

Choose parameters so that `h(x)` is close to `y` for training values `(x,y)` . Minimize $J(\theta_0, \theta_1)$:

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

The function J is called the "cost function", the "squared error function", or the "mean squared error".

Cost Function - Intuition I

hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

parameters: θ_0, θ_1

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

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

Start by choosing $\theta_0 = 0$. It then follows that $J(1) = 0$ for a training set where $y = x$.

By plotting $J(\theta_1)$ *vs.* θ_1, you obtain a parabola with a minimum at $\theta_1 = 0$.

Cost Function - Intuition II

Plotting the minimization function with both variables θ_0 and θ_1 produces a paraboloid.

Parameter Learning

Gradient Descent

Gradient descent is an algorithm to achieve the goal of minimizing the parameters of the cost function.

Goal - $\min_{\theta_0 ... \theta_n} J(\theta_0, ..., \theta_n)$

Approach:

1. Start with some set of θ's
2. Keep changing the θ's until we hopefully end up at a minimum for J

Algorithm:

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

where α is the learning rate.

We need to simultaneously update both parameters. This means computing the algorithm for $j = 0$ and $j = 1$ and then updating both θ_0 and θ_1.

Correct	Incorrect
$\texttt{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ $\texttt{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ $\theta_0 := \texttt{temp0}$ $\theta_1 := \texttt{temp1}$	$\texttt{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ $\theta_0 := \texttt{temp0}$ $\texttt{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ $\theta_1 := \texttt{temp1}$

Gradient Descent - Intuition

Updating the parameters by using the gradient descent algorithm will always change the value to be closer to the minimized value because of the derivative term. If the gradient is positive, then the updated parameter will be less than the starting value, and *vice versa* for a negative gradient.

If the learning rate, α, is too small, the optimization will be slow. Conversely, if it is too large, you may overshoot the minimum or even diverge.

If the parameter is initialized at a *local* minumum, then the gradient descent algorithm will leave its value unchanged.

Gradient descent can converge to local minima even with a fixed α. This is due to the gradient decreasing as the parameter value approaches a minimum.

Note: gradient descent can be used to optimize any cost function, not just the linear regression one.

Gradient Descent for Linear Regression

Cost Function:

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

where $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$.

Derivative term from gradient descent:

$\text{for } j = 0 : \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)} \right)$

$\text{for } j = 1 : \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)} \right) \cdot x^{(i)}$

So the problem becomes:

$$\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)} \right) \\
&\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)} \right) \cdot x^{(i)} \\
&\}
\end{aligned}$$

Linear Algebra Review

Matrices & Vectors

The dimension of a matrix is (# $rows \times$ # $columns$). Matrix elements are referred to by their index, as in A_{ij} is the i, j entry in the i^{th} row and j^{th} column of matrix A.

A vector is an $n \times 1$ matrix. Indexing can be started from either 0 or 1. For this course, assume that indexing starts at 1 unless otherwise specified. Also by convention, uppercase letters are used for matrices and lowercase letters refer to vectors.

Addition & Scalar Multiplication

Matrices can only be added (or subtracted) when they have the same dimensions. Elements are added according to their corresponding indeces.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 1 & -2 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 1+1=2 & 2+(-2)=0 \\ 3+2=5 & 4+3=7 \end{bmatrix}$$

A matrix can also be multiplied by a scalar quantity.

These operations can be combined, and they follow the standard order of operations.

Aside: Octave Programming Language

Octave is similar to Matlab. Below is some example code for working with matrices in Octave:

```
% Initialize matrix A and B
A = [1, 2, 4; 5, 3, 2]
B = [1, 3, 4; 1, 1, 1]

% Initialize constant s
s = 2

% See how element-wise addition works
add_AB = A + B

% See how element-wise subtraction works
sub_AB = A - B

% See how scalar multiplication works
mult_As = A * s

% Divide A by s
div_As = A / s

% What happens if we have a Matrix + scalar?
add_As = A + s
```

Matrix-Vector Multiplication

In order to multiply two matrices together, the first matrix must have the same number of columns as the second matrix has rows. In general, multiplying a $m \times n$ matrix by a $n \times 1$ vector will result in a vector with dimensions $m \times 1$.

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} (1 \times 1) + (3 \times 5) = 16 \\ (4 \times 1) + (0 \times 5) = 4 \\ (2 \times 1) + (1 \times 5) = 7 \end{bmatrix}$$

Applied problem: for house sizes $2104, 1416, 1534, 852$ and $h_\theta(x) = -40 + 0.25x$, we can simultaneously solve for all values of h_θ using matrices.

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 486 \\ 314 \\ 343.5 \\ 173 \end{bmatrix}$$

Matrix-Matrix Multiplication

In order to multiply two matrices together, the first matrix must have the same number of columns as the second matrix has rows. In general, multiplying a $m \times n$ matrix by a $n \times p$ matrix will result in a matrix with dimensions $m \times p$.

To obtain the product of multiplying two matrices, treat each column of the second matrix as a vector and find the vector product. Then combine each product vector into a single matrix, where each column is one of the vectors.

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}$$

We can compare competing hypotheses quickly using these methods. For the same house size data set from the previous section and the following three hypotheses:

1. $h_\theta(x) = -40 + 0.25x$
2. $h_\theta(x) = 200 + 0.1x$
3. $h_\theta(x) = -150 + 0.4x$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Matrix Multiplication Properties

Let A, B, and C be matrices. Then in general,

$$A \times B \neq B \times A \quad \text{not commutative}$$
$$A \times B \times C = A \times (B \times C) = (A \times B) \times C \quad \text{associative}$$

Multiplying a matrix by the identity matrix, I, which has 1's along the diagonal and 0's everywhere else, returns the matrix. $A \cdot I = I \cdot A = A$

To create an identity matrix in Octave:

```
% the following are equivalent
I = [1,0;0,1]
I = eye(2)
```

Inverse and Transpose

The inverse, A^{-1}, of a matrix A, is the matrix that when multiplied by A produces the identity matrix. $AA^{-1} = A^{-1}A = I$.

Only square matrices have an inverse (except for matrices that are all 0's).

To find the inverse of a matrix in Octave:

```
>> A = [3, 4; 2 16]
A =
    3   4
    2  16

>> pinv(A)
ans =
    0.400   -0.100
   -0.050    0.075
```

To obtain the transpose of a matrix, take each row of the matrix and make it the column of another matrix.