Machine Learning

Coursera

November 29, 2016

1 Introduction:

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

Machine Learning:
– Grew out of work in AI
– New capability for computers

Examples: – Database mining:
Large datasets from growth of automaton/web
e.g., Web click data, medical records, biology, engineering
– Applications can't program by hand.
e.g., Autonomous helicopeter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.
– Self-customizing programs
e.g., Amazon, Netflix product recommendations
– Understanding human learning (brain, real AI).

1.1 Supervised Learning

e.g. Housing price prediction

Definition: Supervised Learning: "right answers" were given. **e.g. house prices were *given* in the outsetthe** "That is, we gave it a data set of houses in which for every example in this data set, we told it what is the right price so what is the actual price that, that house sold for and the toss of the algorithm was to just produce more of these right answers such as for this new house, you know, that your friend may be trying to sell."

"To define with a bit more terminology this is also called a regression problem and by regression problem I mean we're trying to predict a continuous value output. Namely the price."

Regression: Prediction is trying to predict a CONTINUOUS VALUED OUTPUT e.g. price

e.g. Is breast cancer malignant or benign??
Tumor size vs. Malignant, and being binary.

This is a CLASSIFICATION problem. e.g. 0 or 1; malignant or benign; DISCRETE VALUE OUTPUT....

"Support Vector Machine", deals with an infinity long list of features (of data...).

e.g., Problem 1. "You have a large inventory of identical items. So imagine that you have thousands of copies of some identical items to sell and you want to predict how many of these items you sell within the next three months.(?)"

e.g., Problem 2. "You have lots of users and you want to write software to examine each individual of your customer's accounts, so each one of your customer's accounts; and for each account, decide whether or not the account has been hacked or compromised."

Answer: Treat Problem 1 as a Regression Problem; Problem 2 as a Classification Problem.

1.2 Unsupervised Learning

Supervised Learning, we are told explicitly what is the so-called "right" answer, ("Is it benign or malignant")

UNSupervised Learning, given data that doesn't have any labels or that all has the same label or really no labels. So we're given the data set and we're not told what to do with it and we're not told what each data point is. Instead we're just told, here is a data set. Can you find some structure in the data?

You don't know how the data should be "spilt up".

e.g., Google News (is Unsupervised Learning).

e.g., Understanding genomics
Unsupervised, here's a bunch of data; have "no idea what the answer is", but can you find structure in the data... :-)

e.g.'s Organize computer clusters; Social network Analysis; Market Segmentation; Astro image data (!!!!)

Using Octave.

Octave: single value decomposition; but that turns out to be a linear algebra routine, that is just built into Octave.

"What I've seen after having taught machine learning for almost a decade now, is that, you learn much faster if you use Octave as your programming environment, and if you use Octave as your learning tool and as your prototyping tool, it'll let you learn and prototype learning algorithms much more quickly. "

Review Question:
Of the following examples, which would you address using an UNsupervised Learning Algorithm?? NOT: Spam fliter, NOT diabetes diagnois. YES: News articles on the web; YES: customer data for market segments.

2 Linear Regression with One Variable

"Linear regression predicts a real-valued output based on an input value. We discuss the application of linear regression to housing price prediction, present the notion of a cost function, and introduce the gradient descent method for learning."

2.1 Model and Cost Function: Model Representation

e.g. Housing Pricing; Supervised Learning, Regression problem.

m = number of training examples
x = "input" variables/features
y = "output" variables/"target" variable
(x, y) – one training example
$(x^{(i)}, y^{(i)})$ – i^{th} training example

"So here's how this supervised learning algorithm works. We saw that with the training set like our training set of housing prices and we feed that to our learning algorithm. Is the job of a learning algorithm to then output a function which by convention is usually denoted lowercase h and h stands for hypothesis. And what the job of the hypothesis is, is, is a function that takes as input the size of a house like maybe the size of the new house your friend's trying to sell so it takes in the value of x and it tries to output the estimated value of y for the corresponding house."

So h is a function that maps from x's to y's.
(h is the "hypothesis", which isn't the best name!!)

How do we represent h??
$h_\theta(x) = \theta_0 + \theta_1 x$
just a simple linear function!!
Linear regression with one variable, x (aka "univariate linear regression" !!).

2.2 Model and Cost Function: Cost Function

θ_0 and θ_1 are the parameter of the model.
Linear regression. Have a training dataset.

Chose θ_0 and θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y).

So, an e.g., of a cost function:
$J(\theta_0, \theta_1)$
this Cost Function, is the Squared Error Function.

And we want to MINIMIZE IT.

2.3 Model and Cost Function: Cost Function – Intuition 1

Hypothesis:

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

Parameters:

θ_0 and θ_1

Cost Function:

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

Goal: Minimize $J(\theta_0, \theta_1)$

Now just fit the best straight line... ;-)

2.4 Model and Cost Function: Cost Function – Intuition 2

Same thing, now just with both parameters, θ_0 and θ_1.
Gives contour plots!! ;-)

2.5 Parameter Learning: Gradient Descent

Have some function $J(\theta_0, \theta_1)$, and what to minimize $J(\theta_0, \theta_1)$.

Outline: Start with some θ_0, θ_1 (say $\theta_0 = 0$ and $\theta_1 = 0$) and change until you (think you) find the minimum...
(The algorithm::) repeat until covergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \tag{1}$$

}.
for $j = 0$ and $j = 1$.

where := is CS lingo for “assignment”
$a := b$
(this means take the value in b and use it overwrite whatever value is a.)

α is the *learning rate.*
Now, need also to do *simultaneous update* ("this is a correct implementation of gradient descent").
Intution: *alpha* is the learning rate (how big the step is for the e.g. derivative...).
Obvious, but good to note anyway:
if α too small, gradient descent can be slow;
if α too big, gradient descent can overshoot the minimum. It may fail to converge, or even diverge(!!)

Note, if you initilize G.D. at the local minimum, the derivate term **is** zero, and hence leaves θ_1 unchanged (which is what you want ;-)

GD can converge to a local minimum, even with the learning rate α fixed. The derivate decreases as you approach the minimum. $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$.
Linear Regression:

$$\begin{aligned}
\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_1} \cdot \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)} - y^{(i)})^2 &\quad (2) \\
&= \frac{\partial}{\partial \theta_1} \cdot \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2 &\quad (3)
\end{aligned}$$

"But, it turns out that that the cost function for linear regression is always going to be a bow shaped function like this. The technical term for this is that this is called a convex function." Convex function has no local optima, other than the global optimum.

"Batch" Gradient Descent. "Batch": each step of GD uses all of the training examples (i.e. you are using the $\sum$).
G.D. is (obviosuly!!) an iterative method (cf. "normal equation" method...)
G.D. is a ML algorithm!!

3 Linear Algebra Review

3.1 Linear Algebra Review

3.1.1 Matrices and Vectors

The dimension of the matrix is going to be written as the number of row times the number of columns in the matrix: $R \times C$ $A_{i,j}$ is the i, j entry in the i^{th} row, j^{th} column.

So it turns out that in most of math, the one index version is more common For a lot of machine learning applications, zero index.

3.1.2 Addition and Scalar Multiplication

Python pages for Linear Algebra: http://docs.scipy.org/doc/numpy/reference/routines.linalg.html

3.1.3 Matrix Vector Multiplication

Recall a 3×4 times a 4×1 gives a 3×1 matrix.

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

Prediction = DataMatrix $\times$ ParameterVector.

3.1.4 Matrix Matrix Multiplication

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

m x n matrix $\times$ **n x o matrix gives you a m x o matrix**.

$$\begin{bmatrix}1 & 3\\2 & 5\end{bmatrix}\begin{bmatrix}0 & 1\\3 & 2\end{bmatrix} = \begin{bmatrix}9 & 4\\15 & 12\end{bmatrix}$$

3.1.5 Matrix Multiplication Properties

In general, $A \times B \neq B \times A$.
$A \times I = I \times A = A$.

3.1.6 Inverse and Transpose

Only Square Matrices have inverses. $A(A^{-1}) = A^{-1}A = I$.

$$A = \begin{bmatrix}1 & 2 & 0\\3 & 5 & 9\end{bmatrix} \Rightarrow A^T = A = \begin{bmatrix}1 & 3\\2 & 5\\0 & 9\end{bmatrix}$$

WEEK TWO

4 Linear Regression with Multiple Variables

4.1 Multivariate Linear Regression

4.1.1 Multiple Features

$x_j^{(i)}$ = value of feature j in $i^{th}za$ training example. e.g., $x_1^{(4)}$ is the first 'feature'/column for the fourth datapoint/row...

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

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_x \end{bmatrix}$$

$h_\theta(x) = \theta^T \mathbf{x}$

4.1.2 Gradient descent for Multiple Variables

$h_\theta(x) = \theta^T \mathbf{x} = \sum_{j=0}^{n} \theta_j x_j^{(i)}$.
$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)} - y^{(i)}) x_0^{(i)}$.

4.1.3 Gradient Descent in Practice I - Feature Scaling

Idea: make sure features are on a similar scale... x_1 =size (0-2000 feet^2) and x_2 =number of bedrooms (0-5)
Just scale by e.g. /2000 and /5, $\Rightarrow$ circular contours, and Gradient Desecent (GD) works a lot better...
Basically want to get every feature into approximately a $-1 \leq x \leq +1$ range.

Mean normalization: replace x_i with $x_i - \mu_i$.
Also can e.g. replace x_i with $(x_i - \mu_i)/s_i$, where s_i is the range of the feature and/or the e.g. standard deviation.

4.1.4 Gradient Descent in Practice II - Learning Rate

How to check GD is working correctly (!!)
If GD is working properly, $J(\theta)$ should decrease after every iteration.
(Recall $J(\theta)$ is the Cost Function.)

Convergence test: e.g. Declare convergence if $J(\theta)$ by less than some threshold ϵ, (but this can be tricky...) Look at e.g. plots of $J(\theta)$ vs. number of iterations....

4.1.5 Features and Polynomial Regression

Pretty self-explanatory... :-)

4.2 Computing Parameters Analytically

4.2.1 Normal Equation

$\theta \in \mathbb{R}^{n+1}$. Example of size(feet^2), no of bedrooms, no of floors, age of home (x_1, x_2, x_3, x_4) and $x_0 = 1$ for all e.g. four $m = 4$ examples. The cost of home is y.
Then

$$\theta = (X^T X)^{-1} X^T y \tag{4}$$

where X is matrix with all the m values of x_i.

So, what is $(X^T X)^{-1}$? It is the inverse of $X^T X$. Set $A = X^T X$, then $(X^T X)^{-1} = A$.

In Octave: `pinv(X' * X) * X' * y`.
m training examples; n features:
Gradient Descent: •Need to choose α; •needs many iterations.
Normal Equation: •No need to choose α; •Don't need to iterate.
vs.
Gradient Descent: •Works well even when n is large.
Normal Equation: •Need to compute $(X^T X)^{-1}$ (which is $n \times n$ and which scales as $\not{n}^3$. (Slow if n is very large (where $n \sim 10^5$ is the tipping point...)

4.2.2 Normal Equation Noninvertibilty

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

What if $X^T X$ is non-invertible?? Basically use `pinv` (pseudo-inv) (in e.g. Octave) to get round this.. Two causes for $X^T X$ being non-invertible are: (1) Redundant features (linearly dependent) e.g. x_1 is size in feet and x_2 is size in m^2 and (2) Too many feature (e.g. $m\ leqn$). If too many features, delete some features, or use regularization.

midterm exam	(midterm exam)^2	final exam
89	7921	96
72	5184	74
94	8836	87
69	4761	78

4.3 Submitting Programming Assignments

4.3.1 Programming tips from mentors

The lectures and exercise PDF files are based on Prof. Ng's feeling that novice programmers will adapt to for-loop techniques more readily than vectorized methods. So the videos (and PDF files) are organized toward processing one training example at a time. The course uses column vectors (in most cases), so h (a scalar for one training example) is theta' * x.

Lower-case x typically indicates a single training example.

The more efficient vectorized techniques always use X as a matrix of all training examples, with each example as a row, and the features as columns. That makes X have dimensions of (m x n). where m is the number of training examples. This leaves us with h (a vector of all the hypothesis values for the entire training set) as X * theta, with dimensions of (m x 1).

X (as a matrix of all training examples) is denoted as upper-case X.

Throughout this course, dimensional analysis is your friend.

Linear Regression with Multiple Variables QUIZ

1. Suppose $m = 4$ students have taken some class, and the class had a midterm exam and a final exam. You have collected a dataset of their scores on the two exams, which is as follows:

You'd like to use polynomial regression to predict a student's final exam score from their midterm exam score. Concretely, suppose you want to fit a model of the form $h_\theta(x) = \theta_0 + \theta_1 x1 + 2x2$, where x1 is the midterm score and x2 is (midterm score)2. Further, you plan to use both feature scaling (dividing by the "max-min", or range, of a feature) and mean normalization.

What is the normalized feature x(1)1? (Hint: midterm = 89, final = 96 is training example 1.) Please round off your answer to two decimal places and enter in the text box below. e.g. 1. $x_2^{(4)} \rightarrow 4761$.

2. Nomalized feature $\rightarrow \frac{x-u}{s}$ where u is average of X and $s = max - min$=8836-4761=4075.

3. Finally, 47616675.54075=0.47

Thus, (89 - ((89+72+94+69)/4)) / (94-69.) = 0.32. :-)

Normalised feature of $x_1^{(1)}$ is 0.32.

Q2. You run gradient descent for 15 iterations with α=0.3 and compute $J(\theta)$ after each iteration. You find that the value of $J(\theta)$ decreases quickly then levels off. Based on this, which of the following conclusions seems most plausible?

A2: $\times$Rather than use the current value of α, it'd be more promising to try a smaller value of α (say α=0.1).

$\surd$ α=0.3 is an effective choice of learning rate.

$\times$Rather than use the current value of α, it'd be more promising to try a larger value of α (say α=1.0).

Q3. Suppose you have $m = 23$ training examples with $n = 5$ features (excluding the additional all-ones feature for the intercept term, which you should add). The normal equation is $\theta = (X^T X)^{-1} X^T y$. For the given values of m and n, what are the dimensions of $, X, andy$ in this equation?

A3: $\surd$: (I think!) X is 23$\times$6, y is 23$\times$1, θ is 6 $\times$1

$\times$: X is 23$\times$5, y is 23$\times$1, θ is 5$\times$1

$\times$: X is 23$\times$5, y is 23$\times$1, θ is 5$\times$5

$\times$: X is 23$\times$6, y is 23$\times$6, θ is 6$\times$6

4. Suppose you have a dataset with m=50 examples and n=15 features for each example. You want to use multivariate linear regression to fit the parameters to our data. Should you prefer gradient descent or the normal equation?

$\times$: The normal equation, since gradient descent might be unable to find the optimal .

$\surd$: The normal equation, since it provides an efficient way to directly find the solution.

$\times$: Gradient descent, since it will always converge to the optimal .

$\times$: Gradient descent, since $(X^t X)^{-1}$ will be very slow to compute in the normal equation.

5. Which of the following are reasons for using feature scaling?

$\times$: It prevents the matrix XTX (used in the normal equation) from being non-invertable (singular/degenerate).

$\times$: It speeds up solving for using the normal equation.

$\surd$: It speeds up gradient descent by making it require fewer iterations to get to a good solution.

$\times$: It is necessary to prevent gradient descent from getting stuck in local optima.

5 Octave/Matlab Tutorial

5.1 Basic Operations

```
1 == 2 % false
ans = 0
1 ~= 2    % not equals, ~ is NOT in Octave
ans = 1
1 && 0 % AND
ans = 0
1 || 0 % OR
ans = 1
xor(1,0)
ans = 1
PS1('>> ');
>>
>> a = 3
a = 3
>> a = 3 ; % semicolon surpressing output...
>>
>> disp(sprintf('2 decimals: \%0.2f', a))
2 decimals: 3.14
>> disp(sprintf('6 decimals: \%0.6f', a))
6 decimals: 3.141593
>> A = [1 2; 3 4; 5 6]
A =

1   2
3   4
5   6

>> v = [1 2 3]
v =

1   2   3

>> v = [1; 2; 3;]
v =

1
2
3

>> v = 1:5
v =

1   2   3   4   5

>> ones(2,3)
```

```
ans =

1   1   1
1   1   1

>> twos(2,3)
error: 'twos' undefined near line 1 column 1
>> C = 2*ones(2,3)
C =

2   2   2
2   2   2

>>
>> w = zeros(1,3)
w =

0   0   0

>> w = rand(1,3)
w =

0.89676   0.65794   0.27405

>> w = rand(3,3)
w =

0.44070   0.29262   0.99302
0.75816   0.90571   0.96482
0.52197   0.85563   0.79157

>> w = randn(3,3) % Return a matrix with normally distributed
    random elements having zero mean and variance one.

w =

0.347484   0.957131 -1.135894
0.848703   0.014639   0.207640
1.569354 -1.489131   0.610893

>> setenv("GNUTERM","qt")

>> w = -6 + sqrt(10)*(randn(1,10000));
>> hist(w)
>> hist(w,50)
>> I = eye(4)
I =

Diagonal Matrix
```

```
1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1

>> help eye
```

http://stackoverflow.com/questions/22898609/octave-does-not-plot
http://stackoverflow.com/questions/13786754/octave-gnuplot-aquaterm-error-set-terminal-aqua-enhanced-title-figure-1-unk I had to add setenv("GNUTERM","X11") to OCTAVE_HOME/share/octave/site/m/startup/octaverc (OCTAVE_HOME usually is /usr/local) to make it work permanently.

5.2 Moving Data Arounds

```
>> load featuresX.dat
>> load ('featuresX.dat')
>> featuresX
>> who
>> whos
>> clear featuresX % gets ride of a variable
>> v = priceY(1:10) % v is now a 10x1 vector
>> save hello.mat v; %saves v to hello.mat
>> clear % clears all the variables
>> load hello.mat
>> save hello.txt v -ascii % save as text (ASCII)
>> A = [1 2; 3 4; 5 6;]
A =

1   2
3   4
5   6

>> A(3,2)
ans = 6
>> A(2,:)
ans =

3   4

>> A(2,:) % : means every element in that row/column
>> A([1 3], :)
ans =

1   2
5   6
```

```
>>
A(:, 2) = [10; 11; 12]
A =

1   10
3   11
5   12

>>  A = [A, [100; 101; 102]];
>> A
A =

1    10   100
3    11   101
5    12   102

>>  A(:) % put all elements of A into a single vector
ans =

1
3
5
10
11
12
100
101
102

>> A = [1 2; 3 4; 5 6;]
>> B = [11 12; 13 14; 15 16;]
>> C = [A B]
C =

1    2   11   12
3    4   13   14
5    6   15   16

>> C = [A; B]
C =

1    2
3    4
5    6
11   12
13   14
15   16
>> [A B] % is the same as
>> [A, B]
```

5.3 Computing on data

```
>> A = [1 2; 3 4; 5 6;];
>> B = [11 12; 13 14; 15 16;];
>> C = [1 1; 2 2];
>> A* C
ans =

5    5
11   11
17   17

>>  A .* B   % N.B. the dot . means element-wise multiplication
ans =

11   24
39   56
75   96

>>v = [1; 2; 3;]
v =

1
2
3

>> 1./v
ans =

1.00000
0.50000
0.33333

>>  1./A
ans =

1.00000   0.50000
0.33333   0.25000
0.20000   0.16667

>> v + ones(length(v),1)
ans =

2
3
4

>> v +1
ans =
2
3
4

>> A'   %  A transpose
ans =

1   3   5
2   4   6

>> val = max(a)
val = 15
>> [val, ind] = max(a)
val = 15
ind = 2
>> max(A)  % does **column wise** maximum
ans =

5   6

>> a <3  % elementwise operation...
ans =

1   0   1   1

>> find (a<3)
ans =

1   3   4

>>A = magic(3) %does a "magic" N-by-N matrix... :-)
A =

8   1   6
3   5   7
4   9   2

>> [r,c] = find(A>=7)
r =

1
3
2

c =

1
2
```

```
3

>> sum(A)
>> prod(A)
ans =

96   45   84

>> floor(A)
ans =

8   1   6
3   5   7
4   9   2

>> ceil(A)
ans =

8   1   6
3   5   7
4   9   2
>>max(A,[],1)

ans =

8   9   7
>> max(A,[],2)
ans =

8
7
9

>>max(max(A))
ans = 9
>> max(A(:))
>> A = magic(9)
A =

47   58   69   80    1   12   23   34   45
57   68   79    9   11   22   33   44   46
67   78    8   10   21   32   43   54   56
77    7   18   20   31   42   53   55   66
6   17   19   30   41   52   63   65   76
16   27   29   40   51   62   64   75    5
26   28   39   50   61   72   74    4   15
36   38   49   60   71   73    3   14   25
37   48   59   70   81    2   13   24   35
```

```
>> sum(A,1) % per-column sum
ans =

369   369   369   369   369   369   369   369   369

>> sum(A,2) % row-wise sum!!
>> A .* eye(9)
ans =

47    0    0    0    0    0    0    0    0
0   68    0    0    0    0    0    0    0
0    0    8    0    0    0    0    0    0
0    0    0   20    0    0    0    0    0
0    0    0    0   41    0    0    0    0
0    0    0    0    0   62    0    0    0
0    0    0    0    0    0   74    0    0
0    0    0    0    0    0    0   14    0
0    0    0    0    0    0    0    0   35

>> sum(sum(A.*eye(9)))
ans = 369
>> sum(sum(A.*flipud(eye(9))))
ans = 369
>> flipud(eye(9))

>> pinv(A) % Inverse of A
ans =

0.147222 -0.144444  0.063889
-0.061111  0.022222  0.105556
-0.019444  0.188889 -0.102778

>> temp = pinv(A)
>> temp * A
ans =

1.00000  0.00000 -0.00000
-0.00000  1.00000  0.00000
0.00000  0.00000  1.00000

>>
```

"Now, let's sum the diagonal elements of A and make sure that also sums up to the same thing. So what I'm gonna do is construct a nine by nine identity matrix, that's eye nine. And let me take A and construct, multiply A element wise, so here's my matrix A. I'm going to do A .êye(9). What this will do is take the element wise product of these two matrices, and so this should Wipe out everything in A, except for the diagonal entries. And now, I'm gonna do sum sum of A of that and this gives me the sum of these diagonal elements, and indeed that is 369."

5.4 Plotting Data

```
    octave:1> setenv("GNUTERM","qt")
    octave:2> PS1('>> ')
    >> t=[0:0.01:0.98];
    >> y1 = sin(2*pi*4*t);
    >> plot(t,y1);
    >> y2 = cos(2*pi*4*t);
    >> plot(t,y2);
    >> plot(t,y1);
    >> hold on;  % doesn't clear plot
    >> plot(t,y2,'r');
    >> xlabel('time')
    >> ylabel('value')
    >> legend('sin', 'cos')
    >> title('my plot')
    >> cd /Users/npr1/Desktop
    >> print -dpng 'myPlot.png'
       warning: print.m: fig2dev binary is not available.
       Some output formats are not available.
   >> graphics_toolkit gnuplot % just implemented after googling
       error...
   >> print -dpng 'myPlot.png'
   >> close % figure goes away
   >> figure(1); plot(t,y1);
   >> figure(2); plot(t,y2);
   >> subplot(1,2,1); % e.g. divides plot into a 1x2 grid...
   >> plot(t,y1)
   >> axis([0.5 1 -1 1]) % resets axis values...
   >> clf;                    % clears figure...
   >> A= magic(5);
   >> imagesc(A);
   >> imagesc(A), colorbar, colormap gray; % Cool; plots a grid
       of (gray) colors/shades, and a color bar
   >> A(1,2)  % row 1, 2nd column element of A
   ans = 24
   >> imagesc(magic(15)), colorbar, colormap gray;
   >> a=1, b=2, c=3 % Comma chaining of commands/function calls.
   a =  1
   b =  2
   c =  3
   >> a=1; b=2; c=3
   c =  3
   >> a=1; b=2; c=3;
```

5.5 Control Statements: for, while, if statement

```
   >> v = zeros(10,1)
   v =

   0
   0
   0
   0
   0
   0
   0
   0
   0
   0

   >> for i=1:10,
   >     v(i)=2^i;  % whitespace actually doesn't matter
   >   end;
   >> v
   v =

      2
      4
      8
     16
     32
     64
    128
    256
    512
   1024

>> indicies=1:10;
>> for i=indicies,
> disp(i)
> end;
1
2
3
4
5
6
7
8
9
10

>> i = 1;
>> while i <=5,
> v(i) = 100;
> i=i+1;
> end;
>> v
v =

100
100
100
100
100
64
128
256
512
1024

>> i=1;
>> while true,
>    v(i) = 999;
>    i= i+1;
>    if i==6;
>      break;
>    end; %end's if
>  end; %end's while...
>> v
v =

 999
 999
 999
 999
 999
  64
 128
 256
 512
1024

>> v(1) = 2;
>> if v(1) ==1,
>     disp('The value is one');
>   elseif v(1) ==2,
>     disp('The value is two');
>   else
>     disp('The value is not one or two');
>  end;
 The value is two
```

```
>> cd /cos_pc19a_npr/programs/Octave
>> pwd
ans = /cos_pc19a_npr/programs/Octave

>> squareThisNumber(5)   %  squareThisNumber is a function...
ans = 25
>> addpath('/cos_pc19a_npr/programs/Octave') % Octave search
    path (advanced/optical)
>> cd ~/Desktop
>> pwd
ans = /Users/npr1/Desktop
>> squareThisNumber(52) % still works fine...
ans = 2704
>>  [a,b] = squareAndCubeThisNumber(52)
a =  2704
b =  140608
>>

>> X = [1 1; 1 2; 1 3]
>> y  = [1; 2; 3]          % case-sensitive!!
>> theta = [0; 1]
>> j = costFunctionJ(X,y,theta) % costFunctionJ in the Octave
    directory...
j = 0
>>
```

Can also use `break` and `continue` in Octave.

5.6 Vectorization

Here's our usual hypothesis for linear regression,

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j \quad (6)$$

$$= \theta^T x \quad (7)$$

and if you want to compute h(x), notice that there's a sum on the right. And so one thing you could do is, compute the sum from $j = 0$ to $j = n$ yourself.

Unvectorized implementation:

```
    prediction = 0.0
    for j = 1:n+1
        prediction = prediction + theta(j) * x(j)
    end;
```

Vectorized implementation:
`prediction = theta' * x ;`
Just to remind you, here's our update rule for a gradient descent of a linear regression:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\ x_j^{(i)} \tag{8}$$

Vectorized implementation: $\theta := \theta - \alpha\delta$, where θ is vector, α is a Real number, δ is a vector and $\delta = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\ x^{(i)}$ and $x^{(i)}$ is a vector.

5.7 Octave/Matlab Quiz

1. Suppose I first execute the following in Octave/Matlab: A = [1 2; 3 4; 5 6];
B = [1 2 3; 4 5 6];
Which of the following are then valid commands? Check all that apply. (Hint: A' denotes the transpose of A.)
$\surd$ C = A' + B;
$\surd$ C = B * A;
$\times$ C = A + B;
$\times$ C = B' * A;

3. Let A be a 10x10 matrix and x be a 10-element vector. Your friend wants to compute the product Ax and writes the following code: v = zeros(10, 1);
for i = 1:10
for j = 1:10
v(i) = v(i) + A(i, j) * x(j);
end
end
How would you vectorize this code to run without any FOR loops? Check all that apply.
$\surd$ v = A * x;

$\times$ v = Ax;

$\times$ v = A .* x;

$\times$ v = sum (A * x);

4. Say you have two column vectors v and w, each with 7 elements (i.e., they have dimensions 7x1). Consider the following code:

z = 0;
for i = 1:7
z = z + v(i) * w(i)
end

Which of the following vectorizations correctly compute z? Check all that apply.
$\surd$ z = sum (v .* w);
$\surd$ z = w' * v;
$\times$ z = v * w';
$\times$ z = w * v';

5. In Octave/Matlab, many functions work on single numbers, vectors, and matrices. For example, the sin function when applied to a matrix will return a new matrix with the sin of each element. But you have to be careful, as certain functions have different behavior. Suppose you have an 7x7 matrix X. You want to compute the log of every element, the square of every element, add 1 to every element, and divide every element by 4. You will store the results in four matrices, A,B,C,D. One way to do so is the following code:
for i = 1:7 for j = 1:7 A(i, j) = log(X(i, j)); B(i, j) = X(i, j) $\hat{2}$; C(i, j) = X(i, j) + 1; D(i, j) = X(i, j) / 4; end end
Which of the following correctly compute A,B,C, or D? Check all that apply.
C = X + 1;
D = X / 4;
A = log (X);
B = X $\hat{2}$;
Still not sure on this last one ??!!!

6 Programming Assignment: Linear Regression

2.2 Gradient Descent

In this part, you will fit the linear regression parameters to our dataset using gradient descent.

2.2.1 Update Equations

The objective of linear regression is to minimize the cost function

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \qquad (9)$$

where the hypothesis $h_\theta(x)$ is given by the linear model

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 \tag{10}$$

Recall that the parameters of your model are the θ_j values. These are the values you will adjust to minimize cost $J(\theta)$. One way to do this is to use the batch gradient descent algorithm. In batch gradient descent, each iteration performs the update

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\ x_j^{(i)} \tag{11}$$

(simultaneously update θ_j for all j).

WEEK THREE

7 Logistic Regression

7.1 Classification and Representation

7.2 Logistic Regression Model

7.3 Multiclass Classification

8 Regularization

8.1 Solving the Problem of Overfitting

8.1.1 The Problem of Overfitting

WEEK FOUR

9 Neural Networks: Representation

9.1 Motivations

9.2 Neural Networks

9.3 Applications

10 Neural Networks: Learning

10.1 Cost Function and Backpropagation

10.2 Backpropagation in Practice

10.3 Application of Neural Networks

10.3.1 Autonomous Driving

11 Advice for Applying Machine Learning

11.1 Evaluating a Learning Algorithm

11.2 Bias vs. Variance

12 Machine Learning System Design

12.1 Building a Spam Classifier

12.2 Handling Skewed Data

12.3 Using Large Data Sets

13 Support Vector Machines

Ross et al. (2015)

References

Ross N. P., et al., 2015, MNRAS, 453, 3932