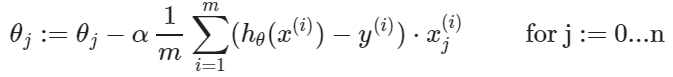
**Week 2**

* **Multivariate Linear Regression**
  + **Text

    Description automatically generated**The multivariate form of the hypothesis function:
  + **Gradient descent for multiple variables:**
    - **Cost function**:

Usually, .

* + - **Techniques for speed up gradient descent**.

1. **Feature scaling**: divides the input values by the range.
2. **A picture containing table

   Description automatically generatedMean normalization**: subtract the average value for an input.
   * + **Learning rate:**

If α is too small, the convergence will be too slow.

If α is too large, the cost function may not decrease on every iteration; may not converge.

\*Debugging gradient descent: make a plot with number of iterations on the x-axis and the cost function J(θ) on the y-axis.

* + Polynomial regression:

The hypothesis function need not to be linear. We can change the behavior or curve of our hypothesis function by making it quadratic, cubic, or square root function.

We can also combine multiple features into one.

In summary, we can choose what features we want to include in the hypothesis function based on our insights.

* **Normal Equation:**

Method to solve for θ analytically.

* + **Text

    Description automatically generated with low confidenceNormal equation formula**:

\*There is no need to do feature scaling with normal equation.

* + **Table

    Description automatically generated with low confidenceCompare normal equation and gradient descent**:
  + Normal equation Noninvertibility:

What if is noninvertible?

Two common causes:

1. Redundant features (two features are closely related, or they are linearly dependent).
2. Too many features (m ≤ n).

Solutions: deleting a feature that is linearly dependent with another / deleting unnecessary features.

* Usage in Octave:
  1. pinv: pseudo-inverse. (always use this!)
  2. inv: inverse.
* Octave/Matlab Tutorial:
  + Basic Operations:
    - 加减乘除 平方 开方：+ - \* / ^ sqrt()
    - 相等判定： ==

True = 1, False = 0

* + - 不相等判定：~=
    - And: &&
    - Or: ||
    - Only one of them is not equal to zero: xor()
    - Assignment (=)

a = 3

* + - Matrix definition:

A = [1 2; 3 4; 5 6]

* + - m x n Matrix of random numbers uniform distribution from 0 to 1:

rand(m, n)

* + - m x n Matrix of random numbers of normal distribution (Gaussian distribution):

randn(m, n)

* + Moving data around:
    - Dimension of a matrix: size()
    - Length of vector or longest dimension of a matrix: length()
    - Get the working directory: pwd
    - Change directory: cd
    - List the current working directory: ls
    - Load the data:

Load xxx

Load(‘xxx’)

* + - Show the current variables in memory:

Who

Whos (detailed view)

* + - Remove a variable: clear xxx
    - Save file: save file.name(xxx.mat xxx.txt) variable.name
    - Call a matrix:

Row m, column n: A(m, n)

Row m: A(m,:)

Column n: A(:,n)

Row a and b: A([a b], :)

* + - Append a column: A = [A, [a;b;c]]
  + Computing on data:
    - Matrix multiplication

Normal (num of first column = num of second row): A\*B

Element-wise (same dimension): A .\*(^) B

* + - Log(), exp(), abs()
    - Matrix of ones: zeros(), ones()
    - Max(matrix,[], dimension(1 or 2))
    - Max(matrix(:))
    - flipud()
    - pinv()
  + Plotting data:
    - Plot(x,y)
    - Plot together: hold on
    - Color: plot(x,y,’r’)
    - Xlabel(‘’)
    - Ylabel(‘’)
    - Lagend(‘’,’’)
    - Title(‘’)
    - Cd ‘saving address’; print -dpng ‘file name(xxx.png)’
    - Figure(1); plot(x,y);
    - Subplot(1,2,1); subplot(1,2,2)
    - Set the x and y ranges: Axis([x1,x2,y1,y2])
    - Clf
    - Imagesc(A)
  + Control statements:
    - For:
      * For i = indices,

Disp(i);

End;

* While:
  + While i <= 5;

v(i) = 100;

i = i+1;

end;

* If, elseif, else, break, continue…
  + Vectorization:

Instead of using for loop to process, using vectors to calculate saves time and uses less code.