Subject: Regression Notes
Date: Friday, March 16, 2018 at 4:08:56 PM Pacific Daylight Time
From: Amar Pendala
To: Amar Pendala
Attachments: image001.png, image002.png, image003.png, image004.png, image005.png, image006.png, image007.png, image008.png, image009.png, image010.png, image011.png, image012.png, image013.png, image014.png, image015.png, image016.png, image017.png, image018.png, image019.png, image020.png, image021.png, image022.png, image023.png, image024.png, image025.png, image026.png, image027.png, image028.png, image029.png, image030.png, image031.png, image032.png, image033.png, image034.png, image035.png, image036.png, image037.png, image038.png

Week 1: Simple Regression

- Linear regression with a single feature
- Find a simple line that best fits the data

Yi = W0 + W1*XJ + Ei
F(x) = W0 + W1*x

Week 2: Multiple Regression

- Linear regression w/ Multiple Features (ie. sq.ft, # of bathrooms, garage == house price)
- Most widely use tool out there in ML
- Polynomial Regression
    - Quadratic function: F(x) = W0 + W1*X + W2*X2^2
    - Higher Order function: F(x) = W0 + W1*X + W2*X2^2 + …. +WN*XN^N
    - Yi = W0 + W1Xi + W2Xi^2 + … + WNXi^N + Ei
        - Treat each power of X as a different **feature**
            - Feature 1 = 1 (constant) , parameter 1 = W0
            - Feature 2 = X , parameter 2 = W1
            - Feature 3 = X^2 , parameter 3 = W2
            - Feature N+1 = X^N , parameter N+1 = WN
    - Very useful in "detrending time series" (ie. values over time – house prices on average over time)
    - Also good to model seasonality (using sin/cosine features)
        - Housing sells best in summer
        - Weather modeling – varies annually &daily
        - Flu monitoring
        - Demand forcasting (jacket purchases)
- Notation (x1 = sq ft, x2 = # of bathrooms)
        - Additional notes:
            - **x** = a vector that contains a list of features (ie. sq ft, bathrooms, etc…)
            - **x**[j] = get the jth item in the vector (ie. sq ft)
            - **x**[i] = vector for ith data point (ie. house i) in the data set
            - **x**i[j] = get the jth item (ie. sq ft) in the vector for the ith data point (ie. house i)

General notation

Output: y ← scalar
Inputs: $\mathbf{x} = (\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[d])$
↖ d-dim vector

Notational conventions:
- $\mathbf{x}[j]$ = j^{th} input (*scalar*)
- $h_j(\mathbf{x})$ = j^{th} feature (*scalar*)
- $\mathbf{x}_i$ = input of i^{th} data point (*vector*)
- $\mathbf{x}_i[j]$ = j^{th} input of i^{th} data point (*scalar*)

So, simple hyperplane (ie. a curve without variations – ie weather model where it varies time of year, but also time of day)

$$y_i = w_0 + w_1\,\mathbf{x}_i[1] + \ldots + w_d\,\mathbf{x}_i[d] + \varepsilon_i$$

feature 1 = 1
feature 2 = $\mathbf{x}[1]$ ... e.g., sq. ft.
feature 3 = $\mathbf{x}[2]$ ... e.g., #bath
...
feature d+1 = $\mathbf{x}[d]$ ... e.g., lot size

For a d-dimensional curve (ie. a curve with variations – ie. weather model where it varies time of year, but also time of day)

- rather than a scalar to define the feature, you'd define a function (ie. sin/cosine) of h1(**x**1)

Model:

$$y_i = w_0\,h_0(\mathbf{x}_i) + w_1\,h_1(\mathbf{x}_i) + \ldots + w_D\,h_D(\mathbf{x}_i) + \varepsilon_i$$
$$= \sum_{j=0}^{D} w_j\,h_j(\mathbf{x}_i) + \varepsilon_i$$

feature 1 = $h_0(\mathbf{x})$ ... e.g., 1
feature 2 = $h_1(\mathbf{x})$ ... e.g., $\mathbf{x}[1]$ = sq. ft.
feature 3 = $h_2(\mathbf{x})$ ... e.g., $\mathbf{x}[2]$ = #bath
or, $\log(\mathbf{x}[7])\,\mathbf{x}[2] = \log(\text{\#bed}) \times \text{\#bath}$
...

- *feature D+1* = $h_D(\mathbf{x})$ ... some other function of $\mathbf{x}[1], \ldots, \mathbf{x}[d]$
- # observations $(\mathbf{x}_i, y_i)$: N
  # inputs $\mathbf{x}[j]$: d
  # features $h_j(\mathbf{x})$: D

- Matrices & Vectors - addition/subtraction & multiplication:
  - http://tutorial.math.lamar.edu/Classes/DE/LA_Matrix.aspx
  - Special Matrices:
    - Zero Matrix:
      - $$0_{n\times m} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}_{n\times m}$$
    - identity Matrix:
      - $$I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}_{n\times n}$$
  - Adding matrices: (matrices must be same dimensions and add)

Example 1 Given the following two matrices,

$$A=\begin{pmatrix}3 & -2\\ -9 & 1\end{pmatrix} \qquad B=\begin{pmatrix}-4 & 1\\ 0 & -5\end{pmatrix}$$

compute A-5B.

Solution
There isn't much to do here other than the work.

$$\begin{aligned}A-5B &= \begin{pmatrix}3 & -2\\ -9 & 1\end{pmatrix}-5\begin{pmatrix}-4 & 1\\ 0 & -5\end{pmatrix}\\ &= \begin{pmatrix}3 & -2\\ -9 & 1\end{pmatrix}-\begin{pmatrix}-20 & 5\\ 0 & -25\end{pmatrix}\\ &= \begin{pmatrix}23 & -7\\ -9 & 26\end{pmatrix}\end{aligned}$$

- Multiplying matrices (multiple row of Matrix A by column of Matrix B and add together)

Example 2 Given

$$A=\begin{pmatrix}2 & -1 & 0\\ -3 & 6 & 1\end{pmatrix}_{2\times 3} \qquad B=\begin{pmatrix}1 & 0 & -1 & 2\\ -4 & 3 & 1 & 0\\ 0 & 3 & 0 & -2\end{pmatrix}_{3\times 4}$$

compute *AB*.

Solution
The new matrix will have size 2 x 4. The entry in row 1 and column 1 of th
corresponding entries from the row of *A* and the column of *B* and then add tl

$$\begin{aligned}c_{11} &= (2)(1)+(-1)(-4)+(0)(0)=6\\ c_{13} &= (2)(-1)+(-1)(1)+(0)(0)=-3\\ c_{24} &= (-3)(2)+(6)(0)+(1)(-2)=-8\end{aligned}$$

Here's the complete solution.

$$C=\begin{pmatrix}6 & -3 & -3 & 4\\ -27 & 21 & 9 & -8\end{pmatrix}$$

- Determinant of matrices

$$\det(A)=|A|$$

re the formulas for the determinant of 2 x 2 and 3 x 3 matric

$$\begin{vmatrix}a & c\\ b & d\end{vmatrix}=ad-cb$$

$$\begin{vmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{vmatrix}=a_{11}\begin{vmatrix}a_{22} & a_{23}\\ a_{32} & a_{33}\end{vmatrix}-a_{12}\begin{vmatrix}a_{21} & a_{23}\\ a_{31} & a_{33}\end{vmatrix}+a_{13}\begin{vmatrix}a_{21} & a_{22}\\ a_{31} & a_{32}\end{vmatrix}$$

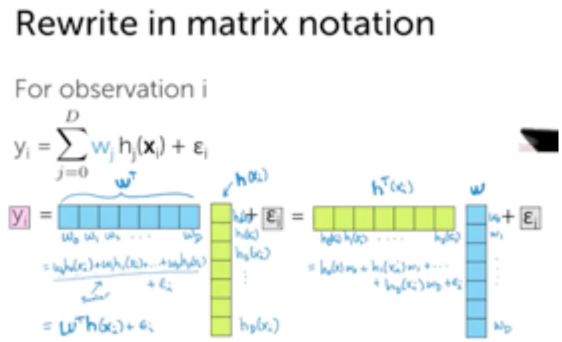
- OR

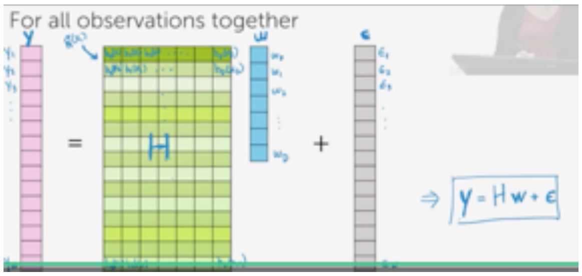
Now, notice that there are three diagonals that run from left to right and three diagonals that run from right to left. What we do is multiply the entries on each di
up and the if the diagonal runs from left to right we add them up and if the diagonal runs from right to left we subtract them.

Here is the work for this matrix.

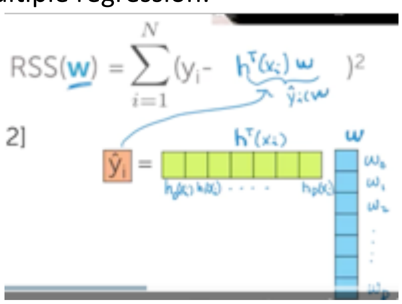
$$\begin{aligned}\det(B) &= \begin{vmatrix}2 & 3 & 1\\ -1 & -6 & 7\\ 4 & 5 & -1\end{vmatrix}\begin{matrix}2 & 3\\ -1 & -6\\ 4 & 5\end{matrix}\\ &= (2)(-6)(-1)+(3)(7)(4)+(1)(-1)(5)-\\ &\qquad (3)(-1)(-1)-(2)(7)(5)-(1)(-6)(4)\\ &= 42\end{aligned}$$

- Inverse Matrix
  - If Matrix AB = BA = In (Identity Matrix), then B is inverse of A ($B=A^{-1}$)
  - To determine inverse of A, do this – create a new matrix of
- Multiple Regression in Matrix format





where **H** is a matrix of transposed **h**(x) vectors

- Residual sum of squares (RSS)
  - For simple linear regression:
    - $$RSS(w_0,w_1) = \sum_{i=1}^{N}(y_i-\underbrace{[w_0+w_1x_i]}_{\hat{y}_i(w_0,w_1)})^2$$
  - For multiple regression:
    - 

  - For multiple regression (in matrix notation):
    - $$RSS(\mathbf{w}) = \sum_{i=1}^{N}(y_i - h(\mathbf{x}_i)^T\mathbf{w})^2 = (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$
- Gradient of RSS:
  - $$\nabla RSS(\mathbf{w}) = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})] = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$
- Least squares D-Dimensional Curve
  - Closed Form Approach – set gradient to 0 & solve for w

- $$\nabla RSS(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) = 0$$

  Solve for **w**:

  $$-\cancel{2}H^Ty + \cancel{2}H^TH\hat{w} = 0$$

  $$H^TH\hat{w} = H^Ty$$

  $$\underbrace{(H^TH)^{-1}H^TH}_{I}\hat{w} = (H^TH)^{-1}H^Ty$$

  $$\hat{w} = (H^TH)^{-1}H^Ty$$

  - $A^{-1}A = I$
  - $Iv = v$

    $IV = V$

- Complexity is O(features^3) – very computationally expensive

○ **Gradient Descent approach (simpler) –** **most widely used algorithm in ML**

- ```
  init w(1)=0 (or randomly, or smartly), t=1
  while ||∇RSS(w(t))|| > ε   ← tolerance
        = sqrt(partial[0]^2 + ... + partial[D]^2)
     for j=0,...,D
        partial[j] = -2 Σ_{i=1}^{N} h_j(x_i)(y_i - ŷ_i(w(t)))
        w_j(t+1) ← w_j(t) – η partial[j]
     t ← t + 1
  ```
  ```