1 Foreword

Author : Anderson Chau

Disclaimer : This notes is written only for my own memorization purpose after I have studied online lecture notes and blogs.

2 K-Means Clustering

Simple Description : Identify clusters by finding the centroid of data points

Algorithm :

1. Initialize μ_1,μ_2.... μ_k randomly (k is hyper-parameter)

2. Repeated until converge :

(i) $c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2, j \in [1 : k]$, (i.e. $c^{(i)}$ denote which μ the $x^{(i)}$ is linked to. Link each data point to nearest μ_j. If $x^{(i)}$ is nearest to μ_s, then $c^{(i)} = j$. Thus, k partitions are created.)

(ii) $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)}=j\}}$, (i.e. For each data point in each partition from (i) , find the new centroid and assign to μ_k

Proof of convergence of the algorithm : consider

$J(c, \mu) = \sum_{i=1}^m ||x^{(i)} - \mu_{c^{(i)}}||^2$
Observation : J must be monotonically decreasing. It is because for step (i) It is adjusting $c^(i)$ to reduce J, for step (ii) we are adjusting μ_j to reduce J
J is non-convext, it may get to local minimum. To try several random initial values, and choose the lowest J.

3 Linear Regression(MSE approach)

Hypothesis :

$$h_\theta(x) = \sum_j \theta_j x_j = \theta^\top x$$

We want to minimize MSE (Mean Square Error)

$$J(\theta) = \frac{1}{2}\sum_i \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2}\sum_i \left(\theta^\top x^{(i)} - y^{(i)}\right)^2$$

Gradient of J :

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_i x_j^{(i)} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

Each θ_j is updated for each step by gradient descent algorithm.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Practically :
(i) Learning rate α is hyperparameter and data dependent , larger, fewer steps to get to min. but may miss the minimum. (Monitor the loss curve, J value vs iteration).
(ii) Batch GD is slow, may be Mini-Batch GD or Stochastic GD.
(iii) If α is small but the loss oscillate , converged and stop learning.

4 Linear Regression(MLE approach)

5 Logistic Regression

$$P(y=1|x) = h_\theta(x) = \frac{1}{1+\exp(-\theta^\top x)} \equiv \sigma(\theta^\top x)$$
$$P(y=0|x) = 1 - P(y=1|x) = 1 - h_\theta(x)$$

Loss function is

$$J(\theta) = -\sum_i \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})) \right]$$

Again , BGD for following gradient of J :

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_i x^{(i)} j \left(h\theta(x^{(i)}) - y^{(i)} \right)$$

Interpretation : For a particular sample : if h return 1/0 and y return 1/0 , the term is 0. if h return 1/0 and y return 0/1 , the term is positive infinity.

6 Logistic Regression(MLE approach)

7 Softmax Regression(Multi-Class Logistic)

k classes, n x k parameters , and the hypothesis is :

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix} \quad (1)$$

Below Loss function is simple to understand : By referencing to previous hpothesis, we want to maximize the y=k associated probility.

$$J(\theta) = -\left[\sum_{i=1}^{m}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log\left(\frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K}\exp(\theta^{(j)\top} x^{(i)})}\right)\right]$$

Gradient of J is, we solve the problem by GD :

$$\nabla_{\theta^{(k)}} J(\theta) = -\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = k\} - P(y^{(i)} = k|x^{(i)};\theta)\right)\right]$$

Where :

$$P(y^{(i)} = k|x^{(i)};\theta) = \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K}\exp(\theta^{(j)\top} x^{(i)})}$$

8 BGD variation : Mini BGD/SGD

BGD use all training data in a single step, which is exteremely costly.

9 Loss function in Classification(Binary) Problem - General treatment

General Hypothesis : $h_\theta(x) = x^T\theta$

Adjustment for binary classification :

$$\text{sign}(h_\theta(x)) = \text{sign}(\theta^T x) = \text{sign}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \\ -1 & \text{if } t < 0 \end{cases}$$

Measure of confidence : $h_\theta(x) = x^T\theta$ gives larger value, more confident

Margin ($yx^T\theta$) : (i) if $h_\theta(x)$ classify correctly, margin is positive, otherwise negative.
(ii) Therefore our objective is to maximize the margin (we want both correct classification and be confident)

Consider the following loss function :

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\phi\left(y^{(i)}\theta^T x^{(i)}\right)$$

We want penalize wrong classfication and encourage correct one , we design ϕ as $\phi(z) \to 0$ as $z \to \infty$, while $\phi(z) \to \infty$ as $z \to -\infty$ where $z = yx^T\theta$, and examples are :

logistic loss : $\phi_{\text{logistic}}(z) = \log(1 + e^{-z})$,used in logistic regression

hinge loss : $\phi_{\text{hinge}}(z) = [1 - z]_+ = \max 1 - z, 0$, used in SVM

Exponential loss $\phi_{\exp}(z) = e^{-z}$, used in boosting

10 Kernel Mapping

(I) Purpose : To x map from lower higher dimension. Useful when data are non-linearly separable(Transform to a curve)
(II) Computation complexity does not necessarily increase proportionately.
(III) Example : a mapping function $\varphi : R \rightarrow R^4$, $x \rightarrow [1, x, x^2, x^3]$, and h is $\theta^T x$ having $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$
(IV) Terms : x is called attribute, $x \rightarrow [1, x, x^2, x^3]$ called feature, φ feature map, $\varphi : R^1 \rightarrow R^4$ in this case. d =1 p = 4
(IV) Another Example : a mapping function $\varphi : R^3 \rightarrow R^{1000}$, $x \rightarrow [1, x_1, x_1^2, x_1^3, x_1x_2, x_1x_2^2....]$ (*) ,let d = 3 , p =1000. If we exhaust all possibilities, then p = 1 + d + d^2 + d^3 (**)
Recall GD stepping :

$$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$$

$$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})x^{(i)}$$

Putting kernel mapping to the equation :

$$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)})$$

We pause here to evaluate the cost of computing each of update (Curse of Demensionality...), considering (**). If we just use the kernel direction, we suffer the curse of demensionality : Suppose d (data dimension) = 1000, then by using the mapping in (**) we have p = 10^9. $\theta^T \phi(x^{(i)})$ need O(p) (dot product) , and O(np) for summing up all data in each step.
Going back to BGD.

$$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)})$$

, assuming $\theta = \sum_{i=1}^{n} \beta_i \phi(x^{(i)})$ at some point, with initialization $\theta = 0 = \beta$
It becomes

$$\theta := \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T \phi(x^{(i)}))\phi(x^{(i)})$$

Rearranging :

$$\theta := \sum_{i=1}^{n} (\beta_i + \alpha(y^{(i)} - \theta^T \phi(x^{(i)})))\phi(x^{(i)})$$

Therefore it is equivalent to updating β_i instead of θ_i by

$$\beta_i := \beta_i + \alpha(y^{(i)} - \theta^T \phi(x^{(i)}))$$

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right)$$