Multi Class Logistic Regression

Dr. Kalidas Y., IIT Tirupati

By the end of this lecture, you will learn about multi class logistic regression

Binary Logistic Regression

- $y_i \in \{+1, -1\}$
- $L(w) = \frac{1}{1+e^{-y_i\,(w \cdot x_i)}}$

- In general if we can define ***Loss Function Per Point***, that would do!

79) key phrase… "Multi Class Logistic Regression"

- y_i is one-hot-encoding of k-classes
- It is a k-dimensional point
- Having all 0's but 1 bit being 1!

- *How to define a loss function in this case???*

Prediction – Vector of Probabilities...

- Now, $y_i \in \{[0,1],[1,0]\}$ //one hot encoding format
- yi is two dimensional point
- Let us say, we have *two w vectors, w_1 and w_0*

- Probability of Class 1,
  - $w_1 \cdot x_i = \log\left(\frac{P(y=1|w_1,x_i)}{P(y=0|w_0,x_i)}\right)$
- Probability of Class 0,
  - $w_0 \cdot x_i = \log\left(\frac{P(y=0|w_0,x_i)}{P(y=1|w_1,x_i)}\right)$
- Combining these, Class yi..

- We need to ensure, that sum of probabilities, $P(y=0)+P(y=1)=1$
- ***Prediction*** is now, a vector of probabilities.. $\widehat{y_i} = [P(y=0|w_0,x_i), P(y=1|w_1,x_i)]$

80) key phrase… "Multi-class loss function"

- For example, given, $y_i = [0, 0, 1, 0, 0, 0]$
- Its prediction, $\widehat{y_i} = [0.01, 0.16, 0.79, 0.01, 0.012, 0.018]$
- Note that $\widehat{y_i}[0] + \widehat{y_i}[1] + \widehat{y_i}[2] + \widehat{y_i}[3] + \widehat{y_i}[4] + \widehat{y_i}[5] = 1$
- The red coloured components above have to match!
- The others should not match i.e. the probabilities *should ideally be zero* for others

- *What is the error function?*

… "Multi-class loss function"

- For example, given, $y_i = [0, 0, 1, 0, 0, 0]$
- Its prediction, $\widehat{y_i} = [0.01, 0.16, 0.79, 0.01, 0.012, 0.018]$
- Note that $\widehat{y_i}[0] + \widehat{y_i}[1] + \widehat{y_i}[2] + \widehat{y_i}[3] + \widehat{y_i}[4] + \widehat{y_i}[5] = 1$
- The red coloured components above have to match!
- The others should not match i.e. the probabilities *should ideally be zero* for others

- *What is the error function?* ***You can have many!***

… "Multi class loss functions"

- For example, given, $y_i = [0, 0, 1, 0, 0, 0]$
- Its prediction, $\widehat{y_i} = [0.01, 0.16, 0.79, 0.01, 0.012, 0.018]$
- Note that $\widehat{y_i}[0] + \widehat{y_i}[1] + \widehat{y_i}[2] + \widehat{y_i}[3] + \widehat{y_i}[4] + \widehat{y_i}[5] = 1$
- The red coloured components above have to match!
- The others should not match i.e. the probabilities *should ideally be zero* for others
- What is the error function? You can have many!
- Simple one.. Mean Squared Error, $\frac{1}{6} \times \sum_{j=1}^{j=6} (y_i[j] - \widehat{y_i}[j])^2$
- Another Simple one.. Mean Absolute Error, $\frac{1}{6} \times \sum_{j=1}^{j=6} |y_i[j] - \widehat{y_i}[j]|$

81) key phrase… “Cross Entropy Loss”

- For example, given, $y_i = [0, 0, 1, 0, 0, 0]$
- Its prediction, $\widehat{y_i} = [0.01, 0.16, 0.79, 0.01, 0.012, 0.018]$
- Note that $\widehat{y_i}[0] + \widehat{y_i}[1] + \widehat{y_i}[2] + \widehat{y_i}[3] + \widehat{y_i}[4] + \widehat{y_i}[5] = 1$
- The red coloured components above have to match!
- The others should not match i.e. the probabilities *should ideally be zero* for others
- What is the error function? You can have many!
- Simple one.. Mean Squared Error, $\frac{1}{6} \times \sum_{j=1}^{j=6} (y_i[j] - \widehat{y_i}[j])^2$
- Another Simple one.. Mean Absolute Error, $\frac{1}{6} \times \sum_{j=1}^{j=6} |y_i[j] - \widehat{y_i}[j]|$
- Cross Entropy Loss, $-\sum_{j=1}^{j=6} y_i[j] * \log(\widehat{y_i}[j])$

Derivation of Probability Vector for Multi-class logistic regression…

Extending Binary Logistic Regression…

- Given k-class problem
- i.e. yi is one-hot-encoding of k classes
- yi is k-dimensional point

- Binary logistic regression has, $w \cdot x_i = \log\left(\frac{P(y=+1|w,x_i)}{P(y=-1|w,x_i)}\right)$

- *But what is +1 and -1 here??*

Define the -1 class (aka Pivot class)… note this is only for derivation purposes

- Let us say, we select "class k" as the pivot class..
- Log odds for c-th class, $\log\left(\frac{P(y=c|w_c,x)}{P(y=k|w_k,x)}\right) = w_c \cdot x$
- $P(y=c|w_c,x) = e^{w_c \cdot x} P(y=k|w_k,x)$
- *But what is the expression for Probability of k-th class???*

Sum of probabilities of all classes has to be 1…
get an expression for the k-th pivot class

- We know,

$\forall c \in [1 \ldots k]$ for each class, $P(y = c|w_c, x) = e^{w_c \cdot x} P(y = k|w_k, x)$

- Sum of probabilities has to be 1
  - $1 = P(y = 1|w_1, x) + \cdots + P(y = k|w_k, x)$
- Arithmetic manipulation
  - $= e^{w_1 \cdot x} P(y = k|w_k, x) + e^{w_2 \cdot x} P(y = k|w_k, x) \ldots + P(y = k|w_k, x)$
  - $= P(y = k|w_k, x)\left(1 + \sum_{c=1}^{k-1} e^{w_c \cdot x}\right) = 1$
- We get an expression for Probability of k-th class,
  - $P(y = k|w_k, x) = \frac{1}{\left(1+\sum_{c=1}^{k-1} e^{w_c \cdot x}\right)}$

Simplify the multi class log odds formulation

- $P(y = k|w_k, x) = \frac{1}{(1+\sum_{c=1}^{k-1} e^{w_c \cdot x})}$
- $P(y = c|w_c, x) = e^{w_c \cdot x} P(y = k|w_k, x) = e^{w_c \cdot x} \times \frac{1}{(1+\sum_{h=1}^{h=k-1} e^{w_h \cdot x})}$
- Multiply and divide by $e^{w_k \cdot x}$
- $\frac{e^{w_k \cdot x}}{e^{w_k \cdot x}} P(y = c|w_c, x) = \frac{e^{w_k \cdot x}}{e^{w_k \cdot x}} \times e^{w_c \cdot x} \times \frac{1}{(1+\sum_{h=1}^{h=k-1} e^{w_h \cdot x})}$
- $= \frac{e^{(w_k + w_c) \cdot x}}{e^{w_k \cdot x} + \sum_{h=1}^{h=k-1} e^{(w_k \cdot x + w_h \cdot x)}} = \frac{e^{w'_c \cdot x}}{\sum_{h=1}^{k} e^{w'_h \cdot x}}$
- By re-defining, for the ith class, $w'_i = w_i + w_k$

- $P(y = y_i|[w'_1, \dots, w'_k], x_i) = \frac{e^{w'_{y_i} \cdot x_i}}{\sum_{c=1}^{k} e^{w'_c \cdot x_i}}$

(Now instead of w’ symbol, we can just use w symbol as well… right? There is nothing special about w’ symbol or w symbol, its our convenience now!)

82) key phrase… “Softmax Operation”

- Let $x = [x_1, \ldots x_m]$ be m-dimensional vector
- $smt([x_1, \ldots, x_m]) = [\frac{e^{x_1}}{\sum_{i=1}^{i=m} e^{x_i}}, \ldots, \frac{e^{x_m}}{\sum_{i=1}^{i=m} e^{x_i}}]$

Code: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.softmax.html

83) key phrase… “Softmax Loss Function”

- Performing two steps…
  - STEP 1: Compute class specific w-vector dot products
  - STEP 2: “softmax operation” followed by
  - STEP 3: application of cross-entropy loss function
- Compute, for k-class problem,
  - STEP 1: Compute this vector, $[w_1 \cdot x_i, \dots, w_k \cdot x_i]$ //for i-th point
  - STEP 2: Do, “softmax operation, $\left[\frac{e^{w_1 \cdot x_i}}{\sum_{j=1}^{j=k} e^{w_j \cdot x_i}}, \dots, \frac{e^{w_k \cdot x_i}}{\sum_{j=1}^{j=k} e^{w_j \cdot x_i}}\right] = (p_1, \dots, p_k)$ (say)
  - STEP 3: Apply “cross entropy loss function”,

$$\mathrm{L}\left(\left(y = (p_1, \dots, p_k)\right) \middle| \, [w_1, \dots, w_k], x_i\right) = -\sum_{j=1}^{j=k} y_i[j] * \log(p_j)$$

Intuition for Cross Entropy Loss Function

- Consider a k-class classification problem
- Let Output vector be, $y_i = [y_i[0], \dots, y_i[k-1]]$
- Let Predicted vector be, $\hat{y}_i = [\hat{y}_i[0], \dots, \hat{y}_i[k-1]]$
- Cross Entropy Loss, $-\sum_{j=1}^{j=k-1} y_i[j] * \log(\hat{y}_i[j])$
  - Each and every, $0 \le \hat{y}_i[j] \le 1$
  - If $\hat{y}_i[j]$ is 0.000001, then $\log(\hat{y}_i[j]) = -10000000$
  - Then the –ve of it, $-y_i[j] log(\hat{y}_i[j]) = y_i[j] * 10000000$
- If $y_i[j] == 1$ and $\hat{y}_i[j]$ is low, then the penalty is very high, if predicted probability is low
- If $y_i[j] == 1$ and $\hat{y}_i[j]$ is 1, then the penalty is 0

Summarizing Multi Class Logistic Regression…

1. Input xi is m-dimensional data point
2. Output yi is k-dimensional data point
   1. k-class classification problem
   2. One hot encoded representation
3. Model, $f(x) = softmax(W \times x)$
   1. $W_{k \times m}$ is a kxm matrix (that needs to be learnt)
4. Data set, $D = \{(x_1, y_1), \ldots (x_N, y_N)\}$
5. Loss function, $L(W) = \sum_{i=1}^{i=N} l_i$
   - l_i is the choice of sub-loss function between two arrays of numbers
   - Squared Error, $l_i = \sum_{j=1}^{j=k} (y_i[j] - \hat{y}_i[j])^2$
   - Absolute Error, $l_i = \sum_{j=1}^{j=k} |y_i[j] - \hat{y}_i[j]|$
   - Cross Entropy Loss, $l_i = -\sum_{j=1}^{j=k} y_i[j] * \log(\hat{y}_i[j])$ (Popular choice!)