CS 335 & CS 337

Adarsh Raj - 190050004

Assignment-3

October 2021

Question 1 - Logistic Regression

Theory - 1.1

Show that: The cross entropy used to train a binary logistic regression (Eqn (3) of Lecture 9) is a special case of the categorical cross entropy function:

$$E(\mathbf{W}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)} \log\left(P(Y=k|\mathbf{w}_k, \phi(x^{(i)})\right)$$

For binary classification, we have classes 1 and 0 (following the class convention in slides), now from the given equation of multiclass classification, we have:

$$P(Y=1|w_k, \phi(x)) = \frac{e^{w_1^T\phi(x)}}{e^{w_0^T\phi(x)} + e^{w_1^T\phi(x)}}$$
$$P(Y=0|w_k, \phi(x)) = \frac{e^{w_0^T\phi(x)}}{e^{w_0^T\phi(x)} + e^{w_1^T\phi(x)}}$$

The above function for $(Y=1)$ can also be represented as:

$$P(Y=1|w_k, \phi(x)) = \frac{1}{1 + e^{(w_0^T - w_1^T)\phi(x)}}$$

This is similar to **sigmoid function** as given in slides, with $w = w_1 - w_0$. Hence, the classification probability function can be represented in the form of sigmoid function (that was given in slides), from the given multi class classification function. Now, similarly binary cross entropy can be derived from the categorical cross entropy loss function as follows:

$$E(\mathbf{W}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_1^{(i)} \log\left(\frac{1}{1+e^{-(w_1^T - w_0^T)\phi(x)}}\right) + y_0^{(i)} \log\left(\frac{e^{-(w_1^T - w_0^T)\phi(x)}}{1+e^{-(w_1^T - w_0^T)\phi(x)}}\right)\right]$$

For the sake of simplicity, replacing $(w_1 - w_0) = w$ in the above function and rearranging logarithmic terms, we have:

$$E(\mathbf{W}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_1^{(i)} \log\left(\frac{1}{1+e^{-w^T\phi(x)}}\right) + y_0^{(i)} \log\left(1 - \frac{1}{1+e^{-w^T\phi(x)}}\right)\right]$$

Note that, for σ (sigmoid function):

$$P(Y = 1|w_k, \phi(x)) = \frac{1}{1 + e^{-w^T \phi(x)}} = \sigma_w(\phi(x))$$

Also, for binary classification with classes 1 and 0, $y_1 = 1 - y_0$ and $y_0 = 1 - y_1$, hence replacing y_1 by y and y_0 by $(1 - y)$. Hence the cross entropy function becomes:

$$E(\mathbf{W}) = -\frac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log \left(\sigma_w(x^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - \sigma_w(x^{(i)}) \right) \right]$$

The above derived expression is same as binary cross entropy function mentioned in the slides. Hence, cross entropy used to train a binary logistic regression is a special case of the given categorical cross entropy.

Theory - 1.2

Logistic Regression's Decision Surface:

Given, a two class dataset with d features, and labels $y \in -1, +1$. The decision rule with decision threshold $\theta \in (0, 1)$ is given as follows:

$$P(y = +1|x) = \frac{1}{1 + \exp(-w^T x)} \geq \theta = (\sigma_w(x) \geq \theta)$$

Also, given $\theta = 0.5$, this is important as the for sigmoid function is greater than 0.5 and less than 0.5, we can calculate the corresponding range of $w^T x$ as follows:

$$P(y = +1|x) \geq 0.5$$
$$1 + \exp(-w^T x) \leq 2$$
$$w^T x \geq 0$$

Similarly, $P(y = +1|x) < 0.5$, if and only if $w^T x < 0$. Hence we have the following representation for prediction y:

$$y = sign(w^T x)$$

Now, note that even if sigmoid function is non linear, sign function is linear. Hence, logistic regression with sigmoid function is considered a linear model.

Theory - 1.3

Multi Class Logistic Regression

(a) Given, x ="The food in the restaurant tastes good." For this data point, the feature vector will contain **994** zeros corresponding to the words not present in the sentence and are in dictionary, and **6** ones corresponding to the words (the, food, in, restaurant, tastes, good).

Suppose "the" is jth word in dictionary, "food" is kth, "in" is lth, and so on upto "good" being the oth word, and since there are 1000 words in the dictionary, the feature vector $\phi(x)$ (length 1000) will be:

- $\phi_i(x) = 1$, if $i \in \{j, k, l, m, n, o\}$
- $\phi_i(x) = 0$, if $i \in \{1, 2, 3....1000\} - \{j, k, l, m, n, o\}$

(b) Any two limitations of this featurization technique are as follows:

- For the given bag of words model, the semantics and the ordering of the words with respect to each other are lost, which are important for deciding the positiveness of a sentence. Similar sentences with same words can mean different when the words are rearranged. For example:

  - "I like eating noodles while on aeroplane"
  - "I like eating aeroplane while on noodles"

  Even though second sentence does not make sense, the model interprets it same as the first sentence.

- For some given data sentence or paragraphs, since the feature vector can be very sparse, computational power and data can be heavily utilized for them which leads to inefficiency while building and training the model.

(c) For the softmax function, we have weight vector corresponding to each class, now, since there are three classes, we have three weight vectors w_{+1}, w_0, w_{-1} for $(+1, 0, -1)$ classes.

Also, each weight vector is a 1 dimensional vector of size (number of features, 1). Hence, there are 1000 parameters for one vector. This makes total parameters to learn 3000.

(d) One of the serious flaws of sum normalization approach is that for any class k, there can exists a datapoint x, such that $w_k^{*T} x < 0$ which results in $P(y = k|x)$ being negative. Now, since probabilities can not be negative, therefore the above case does not make sense. Also, the denominator term can sum out to 0 for some specific datapoints, which is also not desirable.

Given,

$$w_0^{*T} x = 0.1$$
$$w_1^{*T} x = 0.5$$
$$w_2^{*T} x = 2$$

Using sum normalisation, we have:

$$P(y=0|x) = \frac{0.1}{0.1+0.5+2} = 0.038$$
$$P(y=1|x) = \frac{0.5}{0.1+0.5+2} = 0.192$$
$$P(y=2|x) = \frac{2}{0.1+0.5+2} = 0.769$$

Using softmax normalisation, we have:

$$P(y=0|x) = \frac{e^{0.1}}{e^{0.1}+e^{0.5}+e^{2}} = 0.11$$
$$P(y=1|x) = \frac{e^{0.5}}{e^{0.1}+e^{0.5}+e^{2}} = 0.16$$
$$P(y=2|x) = \frac{e^{2}}{e^{0.1}+e^{0.5}+e^{2}} = 0.73$$

It can be observed that maximum probability values corresponding to softmax normalisation is less than the maximum value of sum normalization. Also, the minimum value of softmax normalisation is greater than the minim of sum normalization. From this it can be inferred that softmax normalisation gives a smooth way from smaller values to greater values as compared to sum normalization. The values are more distributed towards the middle in softmax compared to sum normalization.

(e) Given, n training data examples $D = \{(x_i, y_i)\}_{i=1}^{n}$, we have to find the expression for data likelihood $P(D|W)$.

Noe, since the problem is a three class classification problem with classes $= \{-1, 0+1\}$, we can consider the corresponding weight vectors w_{-1}, w_0, w_{+1}. Now for a given data point with index i, we have:

$$P((x_i, y_i)|W) = \frac{e^{-w_{y_i}^T \phi(x_i)}}{\sum_k^K e^{-w_k^T \phi(x_i)}}$$

Since, all the data points can be assumed as independent and identically distributed, for the dataset, we can product the individual probabilities to get the final expression:

$$P(D|W) = \prod_{i=1}^{n} \frac{e^{-w_{y_i}^T \phi(x_i)}}{\sum_k^K e^{-w_k^T \phi(x_i)}}$$

Hence, the required expression for data likelihood is given by the above expression.

(f) For maximum likelihood estimation, the log likelihood for the data likelihood can be calculated as follows:

$$\log(P(D|W)) = Numer - Denom$$

From the above expression derived of $P(D|W)$, the $Numer$ and $Denom$ respectively are :

$$Numer = log(\prod_{i=1}^{n}(e^{-w_{y_i}^T \phi(x_i)})) = -\sum_{i=1}^{n} w_{y_i}^T \phi(x_i)$$

$$Denom = log(\prod_{i=1}^{n}(\sum_{k}^{K} e^{-w_k^T \phi(x_i)})) = \sum_{i=1}^{n} log(\sum_{k}^{K} e^{-w_k^T \phi(x_i)})$$

$$log(P(D|W) = Numer - Denom = -\sum_{i=1}^{n} w_{y_i}^T \phi(x_i) - \sum_{i=1}^{n} log(\sum_{k}^{K} e^{-w_k^T \phi(x_i)})$$

(g) For calculating the gradient for $Numer$, with repect to w_{jk} we have to consider only those points, which belong to the class k, corresponding to the weight class vector w_k. To do this, let use define an identity function as follows:

- $I(y_i == k) = 1$, if $y_i = k$
- $I(y_i == k) = 0$, if $y_i \neq k$

Now, the gradient can be calculated as follows, for data points (x_i, y_i) such that $y_i = k$:

$$\frac{\partial Numer}{\partial w_{jk}} = -\sum_{i=1}^{n}[1 * \phi_j(x_i)]$$

$$\frac{\partial Numer}{\partial w_{jk}} = -\sum_{i=1}^{n}[I(y_i == k)\phi_j(x_i)]$$

The above expression represents that the contribution in gradient comes from only the datapoints belonging to the class k in the given multiclass classification.

(h) For calclulating the gradient for *Denom*, we can directly use the chain rule which is as follows:

$$\frac{\partial Denom}{\partial w_{jk}} = \sum_{i=1}^{n} \frac{1}{\sum_{k'}^{K} e^{-w_{k'}^T \phi(x_i)}} . \exp(-w_k^T \phi(x_i)).(-\phi_j(x_i))$$

$$\frac{\partial Denom}{\partial w_{jk}} = \sum_{i=1}^{n} \frac{\exp(-w_k^T \phi(x_i)).(-\phi_j(x_i))}{\sum_{k'}^{K} e^{-w_{k'}^T \phi(x_i)}}$$

$$\frac{\partial Denom}{\partial w_{jk}} = \sum_{i=1}^{n} \phi_j(x_i) P(y_i = k | x_i, \mathbf{W})$$

(i) From the numerator and denominator gradient above derived, we have the derivative of log likelihood as follows:

$$\frac{\partial P(D|W)}{\partial w_{jk}} = \frac{\partial Numer}{\partial w_{jk}} - \frac{\partial Denom}{\partial w_{jk}}$$

$$\frac{\partial P(D|W)}{\partial w_{jk}} = -\sum_{i=1}^{n} [I(y_i == k)\phi_j(x_i)] - \sum_{i=1}^{n} \phi_j(x_i) P(y_i = k | x_i, \mathbf{W})$$

$$\frac{\partial P(D|W)}{\partial w_{jk}} = -\sum_{i=1}^{n} \phi_j(x_i) \left[I(y_i == k) - P(y_i = k | x_i, \mathbf{W}) \right]$$

(j) Now, using the first order optimality condition and equating the gradient obtained earlier to 0, to get optimal $\mathbf{W}^*$:

$$\left[\frac{\partial P(D|W)}{\partial w^{jk}} \right]_{W=W^*} = 0$$

$$\frac{\partial P(D|W^*)}{\partial w_{jk}^*} = -\sum_{i=1}^{n} \phi_j(x_i) \left[I(y_i == k) - P(y_i = k | x_i, \mathbf{W}^*) \right] = 0$$

$$\sum_{i=1}^{n} \phi_j(x_i) \left[I(y_i == k) - P(y_i = k | x_i, \mathbf{W}^*) \right] = 0, \forall k, j$$

Note that, the above equation with the definition of I (identity function), are the balance equations for different values of j, k (i.e features and classes).

(k) The above expression ensures that the real chance for the event $(y_i = k)$, ensured by the function $I(y_i == k)$, is as close as possible with the predicted probability $P(y_i = k|W)$ because of the sum of differences summing up to zero. Hence for every datapoint the above holds.

For any given class, if the predicted probability does not correspond to the correct chance, then for every data point we have the difference between identity and predicted probability which in laymen terms add weights accordingly to the feature value, like for true predictions, the difference will be positive and negative with small values, and for false predictions the weights will be more penalising the feature term more. Overall, the weighted sum will come out to be zero, hence we are we are discouraging the bias for classification for a given k.

Lab - 1.4

Multi Class Logistic Regression

(a) Functions completed in notebook `assignment_3.ipynb`.

(b) For, The model M which predicts the digit 1 for any image, accuracies were calculated using code, and for class 1, found out to be **0.2425**.

```
accuracy(Y_pred, Y_val)

Total Accuray :  0.2425
Accuray class 1 :  0.2425
Accuray class 4 :  0.74
Accuray class 7 :  0.745
Accuray class 9 :  0.7575

[0.2425, 0.2425, 0.74, 0.745, 0.7575]
```

Figure 1: Accuracy for Model M

For a certain seed, for the multiclass classification given in first part of lab, the accuracy for class 1 was found out to be equal to **0.99**.

```
In [29]: accuracy(Y_pred, Y_val)

         Total Accuray :  0.915
         Accuray class 1 :  0.99
         Accuray class 4 :  0.945
         Accuray class 7 :  0.9575
         Accuray class 9 :  0.9375

Out[29]: [0.915, 0.99, 0.945, 0.9575, 0.9375]
```

Figure 2: Accuracy for a the multi class classification model

Now,

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

In a problem where there is a large class imbalance, a model can predict the value of the majority class for all predictions and achieve a high classification accuracy, the problem is that that model may not be useful in the problem domain. Hence, accuracy is not a good metric for multi class classification. The Model M is an example of this were accuracies are about 75 % for classes other than 1, which clearly is a very bad metric for M.

(c) Completed function `caclulate_metrics()` in `assignment_3.ipynb`. The precision, recall and f1 score observed are as follows:

```
precision, recall, f1_score = calculate_metrics(Y_pred, Y_val)
```

```
print(precision)
print(recall)
print(f1_score)
```

```
[0.916102876993678, 0.9607843137254902, 0.85, 0.925531914893617, 0.923076923076923]
[0.915, 1.0, 0.9239130434782609, 0.8969072164948454, 0.8495575221238938]
[0.9146159807505166, 0.98, 0.8854166666666666, 0.9109947643979057, 0.8847926267281105]
```

Figure 3: Precision, recall and f1 score for a the multi class classification model

For the model M in the above subpart, precison, recall and f1 score are as follows:

```
print(precision)
print(recall)
print(f1_score)
```

```
[nan, 0.2425, nan, nan, nan]
[0.2425, 1.0, 0.0, 0.0, 0.0]
[nan, 0.3903420523138833, nan, nan, nan]
```

Figure 4: Precision, recall and f1 score for the model M

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively, it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if we have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.