CS 335
Assignment 3

Aakriti

Due 17th October 2021

Contents

1 Logistic Regression

1.1 CS 337: Logistic Regression

We consider the following formulation for extending the logistic regression to a multi-class classification setup:

$$P(Y = k|w_k, \phi(x)) = \frac{e^{w_k^T \phi(x)}}{\sum_{k=1}^{K} e^{w_k^T \phi(x)}}$$

Substituting $K = 2$,

$$\begin{aligned} P(Y = 1|w_1, \phi(x)) &= \frac{e^{w_1^T \phi(x)}}{e^{w_1^T \phi(x)} + e^{w_0^T \phi(x)}} \\ &= \frac{1}{1 + e^{-(w_1 - w_0)^T \phi(x)}} = \sigma_{w_1 - w_0}(x) \qquad (1) \\ P(Y = 0|w_0, \phi(x)) &= \frac{e^{w_0^T \phi(x)}}{e^{w_1^T \phi(x)} + e^{w_0^T \phi(x)}} \\ &= 1 - \sigma_{w_1 - w_0}(x) \qquad (2) \end{aligned}$$

Note that, $y_0^{(i)} + y_1^{(i)} = 1$ for any general i^{th} training example. (3)

In multi-class logistic method, we optimize the weight vectors w_k by optimizing the cross-entropy loss function:

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

Substituting $K = 2$,

$$= -\frac{1}{N} \sum_{i=1}^{N} \left(y_1^{(i)} \log\left(P(Y = 1|w_1, \phi(x^{(i)}))\right) + y_0^{(i)} \log\left(P(Y = 0|w_0, \phi(x^{(i)}))\right) \right)$$

Using eq.(1), (2) :

$$= -\frac{1}{N} \sum_{i=1}^{N} \left(y_1^{(i)} \log\left(\sigma_{w_1 - w_0}(x^{(i)})\right) + y_0^{(i)} \log\left(1 - \sigma_{w_1 - w_0}(x^{(i)})\right) \right)$$

Using eq.(3) :

$$= -\frac{1}{N} \sum_{i=1}^{N} \left(y_1^{(i)} \log\left(\sigma_{w_1 - w_0}(x^{(i)})\right) + (1 - y_1^{(i)}) \log\left(1 - \sigma_{w_1 - w_0}(x^{(i)})\right) \right)$$

Therefore, cross entropy used to train a binary logistic regression is a special case of multi-class regression.

1.2 Decision surface of Logistic Regression

$$\begin{aligned}
P(y=+1\,|\,\mathbf{x}) &\geq \theta \\
\frac{1}{1+e^{-\mathbf{w^T x}}} &\geq \theta \\
\frac{1}{\theta} &\geq 1+e^{-\mathbf{w^T x}} \\
e^{-\mathbf{w^T x}} &\leq \frac{1}{\theta}-1 \\
\mathbf{w^T x} &\geq -\log(\frac{1}{\theta}-1) \\
\mathbf{w^T x} &\geq 0 \text{ Substituting } \theta = 0.5
\end{aligned}$$

This is a linear decision boundary. The separating boundary is a hyperplane in $\mathcal{R}^d$ with the plane parameters = $\mathbf{w}$.

Therefore, Logistic Regression is a linear model.

1.3 CS 337: Multi Class Logistic Regression

(a)

Let $\mathcal{D}$ be the dictionary containing the vocabulary of 1000 words in lexicographic order. Let $\mathcal{I}(.)$ be the index function that inputs a word and outputs the index of that word in the dictionary $\mathcal{D}$.
Further, define a set $S = \{\mathcal{I}(\text{food}), \mathcal{I}(\text{good}), \mathcal{I}(\text{in}), \mathcal{I}(\text{restaurant}), \mathcal{I}(\text{tastes}), \mathcal{I}(\text{the})\}$.

So, $\phi_j(x) = 1\ \forall j \in S$, otherwise $\phi_j(x) = 0\ \forall j \notin S$.

(b)

1. We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text. This is quite useful in many natural language processing tasks, and in storing sentences this way, we lose that information
2. The dictionary's size might be quite large, resulting in a high-dimensional but sparse Y. This increases the cost of storage and calculation (unless one makes use of data structures for storing sparse objects)
3. A phrase with a double negation will be marked as negative, even when it is actually positive
4. If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too. If we chose not to increase them, we will be compelled to dismiss either the term or the phrase.
5. Equivalent words are counted twice and are differentiated. For example, "apple" and "apples" will both have $\phi(x)_{index} = 1$

(c)

We have to learn a weight vector for each class : $\mathbf{w_k}$
For each class, weight vector $\mathbf{w}_k$ will have dimension = $|dictionary|$.

The number of classes = K
Let the size of dictionary $|dictionary| = D$, We are given that K = 3, D = 1000
$\therefore$ The number of features/parameters to be learnt are $K \times \dim(\mathbf{w_k}) = 3 \times 1000 = \boxed{3000}$

(d)

Sum normalization:

$$P(y=k|x) = \frac{{w_k^*}^T x}{\sum_{k=0}^{2} {w_k^*}^T x}$$

Since the weight vectors w_k^* can be arbitrary, the quantity ${w_k^*}^T x$ can be negative. It may happen that ${w_k^*}^T x > \sum_{k=0}^{2} {w_k^*}^T x$, leading to a probability greater than one, or even a negative probability.

(e)

The posterior values are given as:

`Normalisation approach`	$P(y=0\|x)$	$P(y=1\|x)$	$P(y=2\|x)$
`Sum Normalisation`	0.03846	0.19230	0.76923
`Softmax Normalisation`	0.10895	0.16254	0.72849

The posterior values are more distributed in case of softmax, whereas in sum normalisation, the probability $P(y=k|x)$ is proportional to ${w_k^*}^T x$.

(f)

Dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$
Let d_i denote the i^{th} training example (x_i, y_i)
Let $\mathcal{I}(y, c) = ((y+1) == c)$, indicate whether y belongs to class c.
Therefore:

$$\begin{aligned} P(D|W) &= \prod_{i=1}^{n} P(d_i|W) \\ &= \prod_{i=1}^{n} \frac{exp\left(w_{y_i+1}^T \phi(x_i)\right)}{\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)} \\ &= \prod_{i=1}^{n} \frac{exp\left(\sum_{k'=0}^{2} \mathcal{I}(y_i, k') w_{k'}^T \phi(x_i)\right)}{\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)} \end{aligned}$$

(g)

$$\begin{aligned}
P(D|W) &= \prod_{i=1}^{n} \frac{exp\left(\sum_{k'=0}^{2} \mathcal{I}(y_i, k') w_{k'}^T \phi(x_i)\right)}{\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)} \\
\log(P(D|W)) &= \log\left[\prod_{i=1}^{n}\left(\sum_{k'=0}^{2} \mathcal{I}(y_i, k') w_{k'}^T \phi(x_i)\right)\right] - \log\left[\prod_{i=1}^{n}\left(\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)\right)\right] \\
\log(P(D|W)) &= \left(\sum_{i=1}^{n}\sum_{k'=0}^{2} \mathcal{I}(y_i, k') w_{k'}^T \phi(x_i)\right) - \left(\sum_{i=1}^{n} \log\left(\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)\right)\right) \\
\text{Numerator: } & \sum_{i=1}^{n}\sum_{k'=0}^{2} \mathcal{I}(y_i, k') w_{k'}^T \phi(x_i) \\
\text{Denominator: } & \sum_{i=1}^{n} \log\left(\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)\right)
\end{aligned}$$

(h)

$$\begin{aligned}
\frac{\partial(Numer)}{\partial w_k^j} &= \frac{\partial(\sum_{i=1}^{n}\sum_{k'=0}^{2} \mathcal{I}(y_i, k') \cdot w_{k'}^T \phi(x_i))}{\partial w_k^j} \\
\frac{\partial(Numer)}{\partial w_k^j} &= \frac{\partial(\sum_{i=1}^{i=n}\sum_{k'=0}^{2} \mathcal{I}(y_i, k') \cdot w_{k'}^T \phi(x_i))}{\partial w_k^j} \\
\frac{\partial(Numer)}{\partial w_k^j} &= \sum_{i=1}^{i=n} \frac{\partial(\sum_{k'=0}^{2} \mathcal{I}(y_i, k') \cdot w_{k'}^T \phi(x_i))}{\partial w_k^j} \\
\frac{\partial(Numer)}{\partial w_k^j} &= \sum_{i=1}^{i=n} \frac{\partial(\mathcal{I}(y_i, k) \cdot w_k^T \phi(x_i))}{\partial w_k^j} \\
\frac{\partial(Numer)}{\partial w_k^j} &= \sum_{i=1}^{i=n} \mathcal{I}(y_i, k) \cdot \phi(x_i)^j
\end{aligned}$$

(i)

$$\frac{\partial(Denom)}{\partial w_k^j} = \frac{\partial\left(\sum_{i=1}^{i=n} \log \left(\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i)) \right)\right)}{\partial w_k^j}$$

$$\frac{\partial(Denom)}{\partial w_k^j} = \sum_{i=1}^{i=n} \frac{\partial\left(\log \left(\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i)) \right)\right)}{\partial w_k^j}$$

$$\frac{\partial(Denom)}{\partial w_k^j} = \sum_{i=1}^{i=n} \frac{1}{\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i))} \cdot \frac{\partial(\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i)))}{\partial w_k^j}$$

$$\frac{\partial(Denom)}{\partial w_k^j} = \sum_{i=1}^{i=n} \frac{1}{\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i))} \cdot \exp(w_k^T \phi(x_i)) \cdot \frac{\partial(w_k^T \phi(x_i))}{\partial w_k^j}$$

$$\frac{\partial(Denom)}{\partial w_k^j} = \sum_{i=1}^{i=n} \frac{1}{\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i))} \cdot \exp(w_k^T \phi(x_i)) \cdot (\phi(x_i)^j)$$

$$\frac{\partial(Denom)}{\partial w_k^j} = \sum_{i=1}^{i=n} \frac{\phi(x_i)^j \exp(w_k^T \phi(x_i))}{\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i))}$$

(j)

$$\frac{\partial \log P(D|W)}{\partial w_k^j} = \frac{\partial(Numer)}{\partial w_k^j} - \frac{\partial(Denom)}{\partial w_k^j}$$

$$\frac{\partial \log P(D|W)}{\partial w_k^j} = \left(\sum_{i=1}^{i=n} \mathcal{I}(y_i, k) \cdot \phi(x_i)^j \right) - \left(\sum_{i=1}^{i=n} \frac{\phi(x_i)^j \exp(w_k^T \phi(x_i))}{\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i))} \right)$$

$$= \phi(x_i)^j \sum_{i=1}^{n} \left(\mathcal{I}(y_i, k) - \frac{exp\left(w_k^T \phi(x_i)\right)}{\sum_{k'=0}^{2} exp\left(w_{k'}^T \phi(x_i)\right)} \right)$$

(k)

Equating the gradient to zero:

$$\frac{\partial \ \log(P(D|W))}{\partial w_k^j} = 0$$

$$\left(\sum_{i=1}^{i=n} \mathcal{I}(y_i, k) \cdot \phi(x_i)^j \right) - \left(\sum_{i=1}^{i=n} \frac{\exp(w_k^T \phi(x_i))\phi(x_i)^j}{\sum_{k'=0}^{2} \exp(w_{k'}^T \phi(x_i))} \right) = 0$$

Let I be a matrix defined as $I_{i,k} = \mathcal{I}(y_i, k)$. Further, define a matrix S such that

$$S_{i,k} = \frac{\exp\left(w_k^T \phi(x_i)\right)}{\sum_{k'=0}^{2} \exp\left(w_{k'}^T \phi(x_i)\right)}$$

Also, $\Phi_{i,j} = \phi(x_i)^j$

$\therefore$ The **balanced equation** is given as:

$$(I - S)^T \Phi = 0$$

(l)

Balanced Equation obtained:

$$(I - S)^T \Phi = 0$$

The optimal W^* should be such that the softmax posterior matrix S should be nearest to the indicator matrix I weighted by the values of $\phi(x_i)$s.

Suppose we train a logistic regression model on our dataset and have found the optimal weights. We will have $\hat{Y}$ as the model distribution, i.e. the distribution yielded by evaluating the model on the training set. For each data point x_i, the model outputs a probability distribution $\hat{Y}$ over $Y \in 1, 2, 3$. Here, Y are the true class labels.

The balance equations state that for each class, the sum over the probabilities over all data points in the model output $\hat{Y}$ equals the number of true labels belonging to thatt class. Seen in this light, the balance equations seem like very reasonable conditions for a classifier—we would like to allocate the same amount of probability mass to each class in the model distribution as in training distribution. In some sense, the balance equations ensure that the training distribution's probability mass is 'preserved' for each class in our model.

1.4 CS 335: Lab Problems

(a)

Refer to `assignment_3.pynb` for the same.

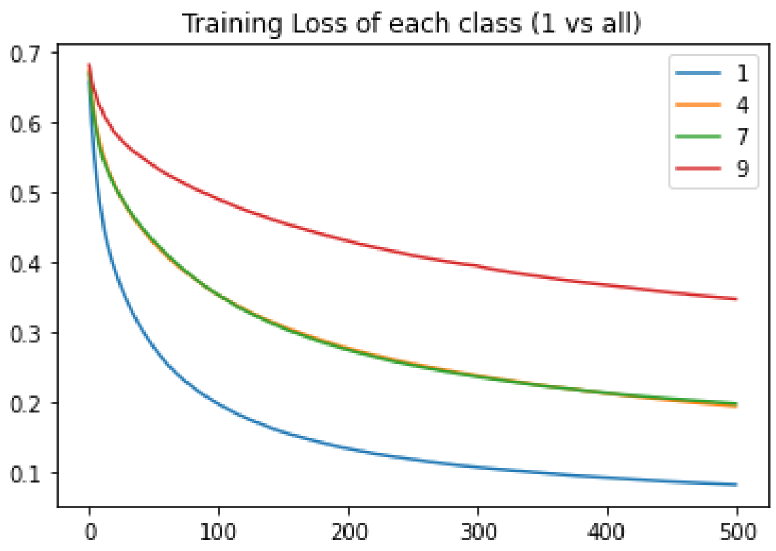
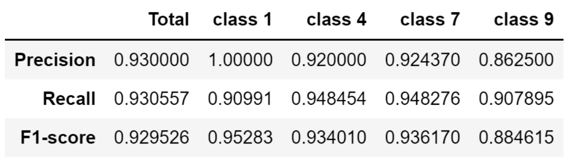


Figure 1: Loss with iterations

Here are the precision, recall and F1-score values:

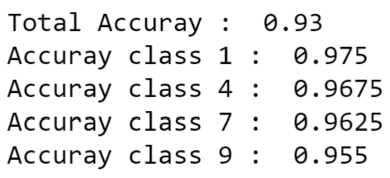


	Total	class 1	class 4	class 7	class 9
Precision	0.930000	1.00000	0.920000	0.924370	0.862500
Recall	0.930557	0.90991	0.948454	0.948276	0.907895
F1-score	0.929526	0.95283	0.934010	0.936170	0.884615

Figure 2: Logistic Regression

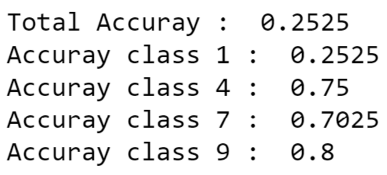
Here are the accuracy scores:

(b)



```
Total Accuray :  0.93
Accuray class 1 :  0.975
Accuray class 4 :  0.9675
Accuray class 7 :  0.9625
Accuray class 9 :  0.955
```

(a) Logistic Regression



```
Total Accuray :  0.2525
Accuray class 1 :  0.2525
Accuray class 4 :  0.75
Accuray class 7 :  0.7025
Accuray class 9 :  0.8
```

(b) Digit-1 Model

Comparing different models using various metrics

Model	**Accuracy**
Logistic Regression	0.93
Digit-1 model	0.2525

Accuracy is a good metric here since the number of test examples for each class are quite evenly distributed. Here is a table that shows distribution of digits in the validation set:

Digit	**Number of Samples**
1	101
4	100
7	119
9	80
`total`	400

Data points of all classes - 1, 4, 7, 9 are distributed almost uniformly as 108, 97, 94, 101. The accuracy for model M should be close to 25% which is precisely the case ($\sim$ 25.25%). So, accuracy is a decent metric in this case if all classes are equally important.

However, we must always look the confusion matrix too to get better idea of out model performance. While, accuracy does tell us about overall performance, we cannot predict the model's behaviour on particular class. Confusion matrix always gives better overview of model performance in any case.

(c)

Here are the precision, recall and F_1 scores:

Model	**precision**	**recall**	F_1 **score**
Logistic Regression	0.9300	0.9305	0.9295
Digit-1 model	0.2525	0.8112	0.1018

F_1 score is a not good metric because it is a harmonic mean between recall and precision. It evaluates how well does the model do on each class, by taking into account the false positives and false negatives. In this case, it gives a very low F_1 score for model M which should not be the case. This is happening because, it does not take TN into account.

When True Positives and True Negatives are more essential, Accuracy is utilised, whereas F1-score is used when False Negatives and False Positives are critical. When the class distribution is comparable, accuracy can be employed, but F1-score is a superior statistic when there are unbalanced classes.