Assignment 3

Devansh Jain, 190100044

17th Oct 2021

Contents

1 Logistic Regression **1**
1.1 CS 337: Logistic Regression . . . 1
1.2 CS 337: Logistic Regressions Decision surface . . . 1
1.3 CS 337: Multi Class Logistic Regression . . . 2
(a) . . . 2
(b) . . . 2
(c) . . . 2
(d) . . . 2
(e) . . . 3
(f) . . . 3
(g) . . . 3
(h) . . . 4
(i) . . . 4
(j) . . . 5
(k) . . . 5
(l) . . . 5
1.4 CS 335: Lab Problems . . . 6
(a) . . . 6
(b) . . . 6
(c) . . . 6

1 Logistic Regression

1.1 CS 337: Logistic Regression

Given Soft-max expression and Categorical cross-entropy function of Multi-class Logistic Regression, we substitute $K = 2$.

$$P(Y = 1|\mathbf{w}_1, \phi(\mathbf{x})) = \frac{e^{\mathbf{w}_1^T \phi(\mathbf{x})}}{e^{\mathbf{w}_1^T \phi(\mathbf{x})} + e^{\mathbf{w}_2^T \phi(\mathbf{x})}} = \frac{1}{1 + e^{(\mathbf{w}_2^T - \mathbf{w}_1^T)\phi(\mathbf{x})}}$$

$$P(Y = 2|\mathbf{w}_2, \phi(\mathbf{x})) = \frac{e^{\mathbf{w}_2^T \phi(\mathbf{x})}}{e^{\mathbf{w}_1^T \phi(\mathbf{x})} + e^{\mathbf{w}_2^T \phi(\mathbf{x})}} = \frac{e^{(\mathbf{w}_2^T - \mathbf{w}_1^T)\phi(\mathbf{x})}}{1 + e^{(\mathbf{w}_2^T - \mathbf{w}_1^T)\phi(\mathbf{x})}}$$

$$\begin{aligned} E(\mathbf{w}_1, \mathbf{w}_2) &= -\frac{1}{N} \sum_{i=1}^{N} y_1^{(i)} \log(P(Y = 1|\mathbf{w}_1, \phi(\mathbf{x}^{(i)}))) + y_2^{(i)} \log(P(Y = 2|\mathbf{w}_2, \phi(\mathbf{x}^{(i)}))) \\ &= -\frac{1}{N} \sum_{i=1}^{N} \frac{y_1^{(i)} + y_2^{(i)} e^{(\mathbf{w}_2^T - \mathbf{w}_1^T)\phi(\mathbf{x})}}{1 + e^{(\mathbf{w}_2^T - \mathbf{w}_1^T)\phi(\mathbf{x})}} \end{aligned}$$

Now, we substitute the one-hot vector $[y_1^{(i)}, y_2^{(i)}]$ with $[y^{(i)}, 1 - y^{(i)}]$ and $\mathbf{w}_1 - \mathbf{w}_2$ with $\mathbf{w}$.

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

$$P(Y = 2|\mathbf{w}, \phi(\mathbf{x})) = \frac{e^{-\mathbf{w}^T \phi(\mathbf{x})}}{1 + e^{-\mathbf{w}^T \phi(\mathbf{x})}} = 1 - \sigma_{\mathbf{w}}(\mathbf{x})$$

$$\begin{aligned} E(\mathbf{w}) &= -\frac{1}{N} \sum_{i=1}^{N} \frac{y^{(i)} + (1 - y^{(i)}) e^{-\mathbf{w}^T \phi(\mathbf{x})}}{1 + e^{-\mathbf{w}^T \phi(\mathbf{x})}} \\ &= -\frac{1}{N} \sum_{i=1}^{N} \frac{y^{(i)}}{1 + e^{-\mathbf{w}^T \phi(\mathbf{x})}} + \frac{(1 - y^{(i)}) e^{-\mathbf{w}^T \phi(\mathbf{x})}}{1 + e^{-\mathbf{w}^T \phi(\mathbf{x})}} \\ &= -\frac{1}{N} \sum_{i=1}^{N} y^{(i)} \sigma_{\mathbf{w}}(\mathbf{x}) + (1 - y^{(i)})(1 - \sigma_{\mathbf{w}}(\mathbf{x})) \end{aligned}$$

Hence, cross entropy function used to train a binary logistic regression is a special case of the categorical cross entropy function given above.

1.2 CS 337: Logistic Regressions Decision surface

The model assigns a value of $y = +1$ to a point $\mathbf{x}$ if $P(y = +1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} \geq \theta$ which is equivalent to $\mathbf{w}^T \mathbf{x} \geq -\log\left(\frac{1 - \theta}{\theta}\right)$ and this forms the decision boundary.

When we say that a model is linear, we mean that its predictions are a linear function of its parameters.

Logistic regression is considered as a linear model because the **decision boundary used for classification purpose is linear** (i.e. $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$).

1.3 CS 337: Multi Class Logistic Regression

(a)

Assuming that we don't lower case all letters during featurization:
Let 'The' be the j_1th word, 'food' be the j_2th word, 'in' be the j_3th word, 'the' be the j_4th word, 'restaurant' be the j_5th word, 'tastes' be the j_6th word and, 'good' be the j_7th word in the vocabulary.
ϕ_j("The food in the restaurant tastes good") = 1 if $j \in \{j_1, j_2, j_3, j_4, j_5, j_6, j_7\}$ else 0.

Assuming that we lower case all letters during featurization:
Let 'the' be the j_1th word, 'food' be the j_2th word, 'in' be the j_3th word, 'restaurant' be the j_4th word, 'tastes' be the j_5th word and, 'good' be the j_6th word in the vocabulary.
ϕ_j("The food in the restaurant tastes good") = 1 if $j \in \{j_1, j_2, j_3, j_4, j_5, j_6\}$ else 0.

(b)

Few notable limitations or drawbacks of this featurization technique:

- Lose of order: We lose the order of words which might be important in determining the meaning and sentiment of the text.
  Example: "It tastes bad, not good" and "It tastes good, not bad"
- Size of dictionary: Dictionary has a lot of words when compared to a text sample, thus the feature matrix which we obtain is sparse.
  It leads to inefficiency in computation.
- In the specified technique, we aren't applying stemming or lower casing of words or removing stop words. All these help in decreasing the dictionary size a lot.

(c)

We would have to learn 3 vectors w_-, w_0, w_+, each of length 1000 (feature vector length = vocabulary size). Thus, we require to learn 3000 parameters.

(d)

If we use sum normalization on [1, 2, 3] and [100, 200, 300], we get the same probabilities.
However, as we have more resolution in the latter case, we should judge that the last class is the more probable than it was in the former case.
This is taken care by the soft-max function.
If we try to normalize following unnormalized values [1, 2, 3] and [101, 102, 103], we should get same probabilities as it is just shifted by constant.
However, we get probability of last class as 0.5 in former case and 0.34 in latter case.
This is taken care by the soft-max function.

(e)

(i) Posterior values are sum normalization are 0.04, 0.19, 0.77.
(ii) Posterior values for soft-max normalization are 0.11, 0.16, 0.73.
Values for $y = 0$ increases in soft-max while the values of $y = 1$ and $y = 2$ decreases.
Usually soft-max is a better approximate of hard-max but here the values are close to 1.

*Note: For **(f)** to **(l)**, assuming that $y_i = k$ means that the point x_i is assigned class k, where $k \in \{1, \ldots, K\}$. $W = \{w_k : k \in \{1, \ldots, K\}\}$ where w_k is the weight vector for class k.*

(f)

$$\begin{aligned}
\mathbb{P}(D|W) &= \mathbb{P}(\{(x_i, y_i)\}_{i=1}^n | \{w_k\}_{k=1}^K) \\
&= \prod_{i=1}^n \mathbb{P}((x_i, y_i)|\{w_k\}_{k=1}^K) \\
&= \prod_{i=1}^n \prod_{k=1}^K \mathbb{P}(Y = k|x_i)^{\mathcal{I}(y_i=k)} \\
&= \prod_{i=1}^n \prod_{k=1}^K \left(\frac{\exp(w_k^T \phi(x_i))}{\sum_{k'=1}^K \exp(w_{k'}^T \phi(x_i))} \right)^{\mathcal{I}(y_i=k)} \\
&= \prod_{i=1}^n \frac{\prod_{k=1}^K (\exp(w_k^T \phi(x_i)))^{\mathcal{I}(y_i=k)}}{\sum_{k'=1}^K \exp(w_{k'}^T \phi(x_i))}
\end{aligned}$$

(g)

$$\begin{aligned}
\log(\mathbb{P}(D|W)) &= \sum_{i=1}^n \left(\sum_{k=1}^K \mathcal{I}(y_i = k)(w_k^T \phi(x_i)) - \log \left(\sum_{k'=1}^K \exp(w_{k'}^T \phi(x_i)) \right) \right) \\
&= \left[\sum_{i=1}^n \sum_{k=1}^K \mathcal{I}(y_i = k)(w_k^T \phi(x_i)) \right] - \left[\sum_{i=1}^n \log \left(\sum_{k'=1}^K \exp(w_{k'}^T \phi(x_i)) \right) \right] \\
&= Numer - Denom
\end{aligned}$$

$$\begin{aligned}
Numer &= \sum_{i=1}^n \sum_{k=1}^K \mathcal{I}(y_i = k)(w_k^T \phi(x_i)) \\
Denom &= \sum_{i=1}^n \log \left(\sum_{k'=1}^K \exp(w_{k'}^T \phi(x_i)) \right)
\end{aligned}$$

(h)

$w_{j,k}$ is the weight of jth feature for class k.

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

(i)

$w_{j,k}$ is the weight of jth feature for class k.

$$\begin{aligned}
\frac{\partial Denom}{\partial w_{j,k}} &= \frac{\partial}{\partial w_{j,k}}\left(\sum_{i=1}^{n}\log\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)\right) \\
&= \sum_{i=1}^{n}\frac{\partial}{\partial w_{j,k}}\left(\log\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)\right) \\
&= \sum_{i=1}^{n}\frac{1}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\frac{\partial}{\partial w_{j,k}}\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right) \\
&= \sum_{i=1}^{n}\frac{1}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\frac{\partial}{\partial w_{j,k}}\left(\exp(w_k^T\phi(x_i))\right) \\
&= \sum_{i=1}^{n}\frac{\exp(w_k^T\phi(x_i))}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\frac{\partial}{\partial w_{j,k}}\left(w_{j,k}\phi_j(x_i)\right) \\
&= \sum_{i=1}^{n}\frac{\exp(w_k^T\phi(x_i))}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\phi_j(x_i)
\end{aligned}$$

(j)

$$\begin{aligned}\frac{\partial \log(\mathbb{P}(D|W)}{\partial w_{j,k}} &= \frac{\partial(Numer - Denom)}{\partial w_{j,k}} \\ &= \left[\sum_{i=1}^{n} \mathcal{I}(y_i = k)\phi_j(x_i)\right] - \left[\sum_{i=1}^{n} \frac{\exp(w_k^T\phi(x_i))}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\phi_j(x_i)\right] \\ &= \sum_{i=1}^{n}\left[\mathcal{I}(y_i = k)\phi_j(x_i) - \frac{\exp(w_k^T\phi(x_i))}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\right]\phi_j(x_i)\end{aligned}$$

(k)

$$\forall\ j,k; \frac{\partial \log(\mathbb{P}(D|W)}{\partial w_{j,k}} = 0 \Bigg|_{W=W^*}$$

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

$$\forall\ k \in \{1,\ldots,K\}; \sum_{i=1}^{n}\mathcal{I}(y_i = k)\phi(x_i) = \sum_{i=1}^{n} \frac{\exp(w_k^T\phi(x_i))}{\left(\sum_{k'=1}^{K}\exp(w_{k'}^T\phi(x_i))\right)}\phi(x_i)$$

$$\forall\ k \in \{1,\ldots,K\}; \sum_{i=1}^{n}\mathcal{I}(y_i = k)\phi(x_i) = \sum_{i=1}^{n}\mathbb{P}(Y = k|x_i)\phi(x_i)$$

These are the Balance Equations.

(l)

The equations mean "For optimal W^*, the sum of feature vectors belonging to a given class k is equal to the weighted sum of all feature vectors where weights are the probability of the point belonging to class k as predicted by the model".

1.4 CS 335: Lab Problems

(a)

(i) Code for the functions `normalize()`, `scaling()` and `split_data` updated in notebook. `split_data` works better with scaling than normalize.

(ii) Code for the function `get_data_for_class()` updated in notebook.

(iii) Code for the function `sample_training_points()` updated in notebook.

(iv) Code for the functions `sigmoid()`, `cross_entropy_loss()` and `grad()` updated in notebook.

(v) Code for the function `logistic_regression()` updated in notebook.

(vi) Code for the function `prediction()` updated in notebook.

(vii) Code for the function `accuracy()` updated in notebook.

(b)

Trained classifier:
Hyper-parameters used - `epochs=500, lr=0.1, sample_size=600`

	Accuracy
Total	0.9425
Class 1	1.0
Class 4	0.9063
Class 7	0.8958
Class 9	0.9612

Model M:

	Accuracy
Total	0.2625
Class 1	1.0
Class 4	0.0
Class 7	0.0
Class 9	0.0

Class based accuracy is not a good metrics as even though there are a lot of false positives for class 1 in model M, it gives accuracy of 1.
And total accuracy of Model M is fraction of images with true value 1. If this number was high then we would get a higher total accuracy without changing the model (which obviously is a really bad model)
However, here as the classes have similar number of images, total accuracy is a good metrics.

(c)

Code for the function `calculate_metrics()` updated in notebook.

Trained classifier:
Hyper-parameters used - `epochs=500, lr=0.1, sample_size=600`

	Precision	Recall	F1-score
Total	0.9451	0.9425	0.9425
Class 1	0.9633	1.0000	0.9813
Class 4	0.9886	0.9063	0.9457
Class 7	0.9556	0.8958	0.9247
Class 9	0.8761	0.9612	0.9167

Model M:

	Precision	Recall	F1-score
Total	0.0689	0.2625	0.1092
Class 1	0.2625	1.0000	0.4158
Class 4	0.0000	0.0000	0.0000
Class 7	0.0000	0.0000	0.0000
Class 9	0.0000	0.0000	0.0000

Recall is same as accuracy. Precision takes into account false positives. This makes F1-score a much better than just accuracy.

Note: To avoid division by zero, I have assumed that the precision, recall and F1-score is 0 if count of true positives is 0.