Assignment 3: CS 335 & CS 337

Richeek Das : 190260036

17th October 2021

1.1 CS337: Logistic Regression

$$P(Y = k|\mathbf{w}_k, \phi(\mathbf{x})) = \frac{\exp(\mathbf{w}_k^T \phi(\mathbf{x}))}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^T \phi(\mathbf{x}))}$$

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

We will show that cross entropy used to train a binary logistic regression in Lecture 9 is a special case of this.

We can expand $\sum_{k=1}^{K} y_k^{(i)} \log(P(Y = k|\mathbf{w}_k, \phi(\mathbf{x}^{(i)})))$ as:

$$y_1^{(i)} \log(P(Y = 1|\mathbf{w}_1, \phi(\mathbf{x}^{(i)}))) + y_2^{(i)} \log(P(Y = 2|\mathbf{w}_2, \phi(\mathbf{x}^{(i)})))$$

In this binary case we can choose not to use a one-hot encoding. We know the class is y_2 when its not y_1. Then let $y = 1$ when class is y_1 and $y = 0$ when class is y_2. Also observe:

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

Choose $-\mathbf{w}$ as $\mathbf{w}_2 - \mathbf{w}_1$. Therefore:

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

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

Rewriting $E(\mathbf{W})$ we get:

$$E(\mathbf{W}) = -\frac{1}{N} \sum_{i=1}^{N} \Big(y \log \sigma_{\mathbf{w}}(\mathbf{x}^{(i)}) + (1 - y) \log(1 - \sigma_{\mathbf{w}}(\mathbf{x}^{(i)})) \Big) \quad \textbf{QED.}$$

1.2 CS337: Logistic Regression's Decision surface

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

We can rewrite this expression as:

$$\frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} \geq \theta$$

$$\implies 1 \geq \theta + \exp(-\mathbf{w}^T \mathbf{x})\theta$$

$$\implies \mathbf{w}^T \mathbf{x} \geq \log\left(\frac{\theta}{1 - \theta}\right)$$

Therefore we see the decision boundary is nonlinear only in terms of the decision **threshold** θ. Here we have a fixed threshold $\theta = 0.5$.

$$\mathbf{w}^T\mathbf{x} \geq 0$$

Therfore Logistic Regression depends only on the sum of its inputs and parameters, hence the decision boundary is **linear**. Therefore Logistic Regression is a linear model.

1.3 CS337: Multi Class Logistic Regression

- i to index training examples
- j to index features of a given example
- k to index classes
- Assumption: no bias parameter involved

Qn (a): $\phi(x)$ would be a **1000 dimensional binary vector with exactly 6 ones and 994 zeros.** 1s will be at the indices of { the, food, in, restaurant, tastes, good } in the vocabulary.

(Since we have the prior information: Each text x is a sequence of words sampled from a dictionary containing 1000 words. i.e., the vocabulary size is 1000. We have 6 distinct words in x).

Qn (b): Drawbacks of **BoW** model:

- The vector output size of the feature fucntions depends on the vocabulary size. If we increase our vocabulary size, the output vector size of ϕ will also go up.
- **BoW** model doesn't keep a track of the grammar or ordering of the words either.
- In **BoW** model the representation is not efficient. The vectors will most likely be very sparse and result in unnecessary memory consumption.

Qn (c): In standard Logistic Regression as defined in this problem statement we assume we are using Softmax and we have 3 weight vectors for 3 classes. Each data point has 1000 features. Therefore a standard MLE approach will need to learn **3000** weight parameters.

Qn (d): Sum normalization proposed:

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

It is quite possible that we might **negative** values for the corresponding probabilities in this case. For example: $w_0^{*T}x = -1$, $w_1^{*T}x = 1$, $w_2^{*T}x = 3$. Then $P(y = 0|x) = \frac{-1}{-1+1+3} = -0.33$. This is definitely a severe flaw.

Qn (e): $[w_0^{*T}x, w_1^{*T}x, w_2^{*T}x = [0.1, 0.5, 2]$.

(a) **Sum normalization:**

$$P(y = 0|x) = \frac{w_0^{*T}x}{w_0^{*T}x + w_1^{*T}x + w_2^{*T}x} = \frac{0.1}{0.1 + 0.5 + 2} = 0.03846$$

$$P(y = 1|x) = \frac{0.5}{2.6} = 0.1923$$

$$P(y = 2|x) = \frac{2}{2.6} = 0.7692$$

(b) **Softmax normalization:**

$$P(y=0|x) = \frac{\exp(w_0^{*T}x)}{\exp(w_0^{*T}x)+\exp(w_1^{*T}x)+\exp(w_2^{*T}x)} = \frac{e^{0.1}}{e^{0.1}+e^{0.5}+e^{2}} = \frac{1.1052}{10.1429} = 0.10896$$

$$P(y=1|x) = \frac{1.6487}{10.1429} = 0.16255$$

$$P(y=2|x) = \frac{7.3891}{10.1429} = 0.72850$$

Comments: For smaller weights softmax provides similar probabilities as compared to sum normalization. It can represent the change in 0.5 to 2.0 better than 0.1 to 0.5. So there's an element of smoothing in softmax which makes it a good differentiable surrogate for the max function

Qn (f): We know:

$$P(Y=k|\mathbf{w}_k,\phi(\mathbf{x})) = \frac{\exp(\mathbf{w}_k^T\phi(\mathbf{x}))}{\sum_{k=1}^{K}\exp(\mathbf{w}_k^T\phi(\mathbf{x}))}$$

Joint probability of data given W:

$$\begin{aligned} P(D|W) &= \prod_{i=1}^{n} P(Y=y_i|W,\phi(\mathbf{x}_i)) \\ &= \prod_{i=1}^{n} \frac{\exp(\mathbf{w}_{y_i+1}^T\phi(\mathbf{x}_i))}{\sum_{k=0}^{K-1}\exp(\mathbf{w}_k^T\phi(\mathbf{x}_i))} \\ &= \frac{\exp\left(\sum_{i=1}^{n}\mathbf{w}_{y_i+1}^T\phi(\mathbf{x}_i)\right)}{\prod_{i=1}^{n}\sum_{k=0}^{K-1}\exp(\mathbf{w}_k^T\phi(\mathbf{x}_i))} \end{aligned}$$

where $K=3$, $y_i \in \{-1,0,1\}$. The weights are w_0, w_1, w_2. So we index w as y_i+1.
$W = [[\mathbf{w}_0],[\mathbf{w}_1],[\mathbf{w}_2]]$

Qn (g): Log data likelihood $\longrightarrow$

$$\begin{aligned} &\log\left(P(D|W)\right) \\ &= \left[\sum_{i=1}^{n}\mathbf{w}_{y_i+1}^T\phi(\mathbf{x}_i)\right] - \sum_{i=1}^{n}\log\left(\sum_{k=0}^{2}\exp(\mathbf{w}_k^T\phi(\mathbf{x}_i))\right) \end{aligned}$$

Numerator $= \sum_{i=1}^{n}\mathbf{w}_{y_i+1}^T\phi(\mathbf{x}_i)$

Denominator $= \sum_{i=1}^{n}\log\left(\sum_{k=0}^{2}\exp(\mathbf{w}_k^T\phi(\mathbf{x}_i))\right)$

Qn (h): Next we compute the gradient $\longrightarrow$

We represent features using j and classes using k.

$$\begin{aligned} \frac{\partial \text{Numerator}}{\partial \mathbf{w}_{k,j}} &= \sum_{i=1}^{n}\frac{\partial \mathbf{w}_{y_i+1}^T\phi(\mathbf{x}_i)}{\partial \mathbf{w}_{k,j}} \\ &= \sum_{i=1}^{n}(y_i+1==k)\phi_j(\mathbf{x}_i) \\ &= \sum_{i:y_i+1==k}^{n}\phi_j(\mathbf{x}_i) \end{aligned}$$

Qn (i): Gradient of denominator $\longrightarrow$

$$\frac{\partial \text{Denominator}}{\partial \mathbf{w}_{k,j}} = \sum_{i=1}^{n} \frac{\partial \log\left(\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))\right)}{\partial \mathbf{w}_{k,j}}$$
$$= \sum_{i=1}^{n} \frac{1}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))} \cdot \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i)) \cdot \frac{\partial \mathbf{w}_k^T \phi(\mathbf{x}_i)}{\partial \mathbf{w}_{k,j}}$$
$$= \sum_{i=1}^{n} \frac{1}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))} \cdot \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i)) \cdot \phi_j(\mathbf{x}_i)$$
$$= \sum_{i=1}^{n} \frac{\phi_j(\mathbf{x}_i) \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}$$

Qn (j): Consolidate the two expressions $\longrightarrow$

$$\frac{\partial \log P(D|W)}{\partial \mathbf{w}_{k,j}} = \frac{\partial \text{Numerator}}{\partial \mathbf{w}_{k,j}} - \frac{\partial \text{Denominator}}{\partial \mathbf{w}_{k,j}}$$
$$= \sum_{i:y_i+1==k}^{n} \phi_j(\mathbf{x}_i) - \sum_{i=1}^{n} \frac{\phi_j(\mathbf{x}_i) \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}$$
$$= \sum_{i=1}^{n} \left(\delta(k, y_i + 1) - \frac{\exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))} \right) \phi_j(\mathbf{x}_i)$$

Qn (k): Balance Equations $\longrightarrow$

Using First Order Optimality conditions:

$$\sum_{i=1}^{n} \left(\delta(k, y_i + 1) - \frac{\exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))} \right) \phi_j(\mathbf{x}_i) = 0$$

We will have **n_classes** $\times$ **n_features** such equations. We will form a matrix interpretation for the same:

Define:

- $\mathcal{C}$ as a **n_samples** $\times$ **n_classes** matrix, where $\mathcal{C}[i][k] = \delta(k, y_i + 1)$.
- $\mathcal{P}$ as a **n_samples** $\times$ **n_classes** matrix, where $\mathcal{P}[i][k] = \frac{\exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}{\sum_{k=0}^{2} \exp(\mathbf{w}_k^T \phi(\mathbf{x}_i))}$
- Φ as a **n_samples** $\times$ **n_features** matrix, where $\Phi[i][j] = \phi_j(\mathbf{x}_i)$

We can rewrite the entire thing as:

$$(\mathcal{C} - \mathcal{P})^T \Phi = \mathbf{0}$$

This the compact form of the **Balance Equations**.

Qn (l): Interpretation of Balance Equations $\longrightarrow$

Columns of $\mathcal{C} - \mathcal{P}$ are orthogonal to columns of Φ. In an ideal case we would have wanted $\mathcal{C} = \mathcal{P}$. In this case we are solving for an approximate solution.

The balance equations state that the sum over the probabilities in each column in $\mathcal{P}$ equals the sum over the corresponding column in $\mathcal{C}$ when we consider only those sentences in which the word is present.

1.4 CS335: Lab Problems

We use: `epochs=1200, lr=0.01, sample_size=200`.

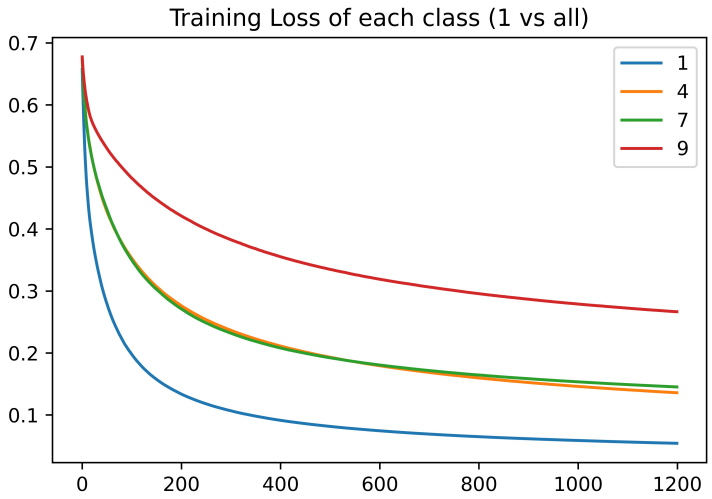

Figure 1: Training Loss

(b) Validation Accuracy Observed ⟶

```
Total Accuracy :  0.9225
Accuracy class 1 :  0.9875
Accuracy class 4 :  0.975
Accuracy class 7 :  0.945
Accuracy class 9 :  0.9375
```

Say we have a model M which predicts 1 for any image. Accuracy of this model on validation set:

```
Total Accuracy :  0.2525
Accuracy class 1 :  0.2525
Accuracy class 4 :  0.76
Accuracy class 7 :  0.7325
Accuracy class 9 :  0.76
```

No, Accuracy is not a good metric here. The accuracy for classes 4,7,9 are very high even though the model never predicts a single instance of positive 4,7,9 correctly. This is sort of survivorship bias, since the number of positive instances are much lesser than the number of negative instances for a particular class.

(c) Precision, Recall and F_1 score on the validation set ⟶

	Total	Class 1	Class 4	Class 7	Class 9
Precision	0.9222	0.9615	0.9479	0.8829	0.8989
Recall	0.9225	0.9901	0.9479	0.9159	0.8333
F_1 Score	0.9219	0.9756	0.9479	0.8991	0.8649

Table 1: Precision, Recall and F_1 Scores on the validation set.

We calculate the F_1 score for all the classes by taking a weighted average based on the support of each class.

Say we have a model M which predicts 1 for any image. Accuracy of this model on validation set:

	Class 1	Class 4	Class 7	Class 9
TP	106	0	0	0
FP	294	0	0	0
FN	0	96	108	90

	Total	Class 1	Class 4	Class 7	Class 9
Precision	0.0702	0.265	0	0	0
Recall	0.265	1	0	0	0
F_1 Score	0.1110	0.4190	0	0	0

Table 2: Precision, Recall and F_1 Scores on the validation set for the **custom model**. Assuming $0/0 = 0$

Yes, Precision, recall and F_1 are good evaluation metrics for this task. Due to high amount of FP, the precision of Class 1 goes down. Since TP=0 for Classes 4,7,9, we observe 0 recall and precision for each of them. Here we check the FP and FN too, unlike just TP and TN in plain accuracy metric. This solves the problem we were previously facing with plain validation accuracy.