15/11/2021 ML- HW5 Sai Satya Charan. Malladi

Q1) (a)

$$\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \phi(y_i(w^T x_i + b)) + \lambda\|w\|^2$$

$$\phi(t) = \log(1+e^{-t})$$

$$J = \min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \log\left(1+e^{-y_i(w^T x_i+b)}\right) + \lambda\|w\|^2$$

$$\frac{\partial J}{\partial w} \Rightarrow \underbrace{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{1+e^{-(y_i(w^T x_i+b))}}\left[-y_i x_i e^{-y_i(w^T x_i+b)}\right]}_{\text{dividing the numerator and denominator by } e^{-y_i(w^T x_i+b)}} + 2\lambda w$$

We get,

$$\boxed{\frac{\partial J}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i(w^T x_i+b)}} + 2\lambda w}$$

Similarly,

$$\frac{\partial J}{\partial b} = \underbrace{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{1+e^{-y_i(w^T x_i+b)}}\left[-y_i e^{-y_i(w^T x_i+b)}\right]}_{\text{dividing numerator and denominator by } e^{-y_i(w^T x_i+b)}} + 0$$

We get,

$$\boxed{\frac{\partial J}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i}{1+e^{y_i(w^T x_i+b)}}}$$

Gradient descent update rule

$$w^{k+1} = w^k - \eta \frac{\partial J}{\partial w^k}$$
$$b^{k+1} = b^k - \eta \frac{\partial J}{\partial b^k}$$

where w^k, b^k indicate hyperparameters at kth iteration.

$$\boxed{\begin{aligned} w^{k+1} &= w^k - \eta\left[\frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i(w^{kT} x_i + b^k)}} + 2\lambda w^k\right] \\ b^{k+1} &= b^k - \eta\left[\frac{1}{n}\sum_{i=1}^{n} \frac{-y_i}{1+e^{y_i(w^{kT} x_i + b^k)}}\right] \end{aligned}}$$

→ This is $(w^k)^T$

→ Update rules

Q1 (b) In Q1 (a) we have got

$$\frac{\partial J}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i(w^T x_i + b)}} + 2\lambda w$$

As the objective J is given to be convex, a unique global minimizer exists., (w^*, b^*). And @ global minimizer

$$\underbrace{\frac{\partial J}{\partial w^*} = 0}_{\downarrow} \qquad \frac{\partial J}{\partial b^*} = 0$$

We will just take $\frac{\partial J}{\partial w^*} = 0$ and examine it.

$$\frac{\partial J}{\partial w^*} = 0 \Rightarrow \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i((w^*)^T x_i + b^*)}} + 2\lambda w^* = 0$$

$$\Rightarrow \quad 2\lambda w^* = +\frac{1}{n}\sum_{i=1}^{n} \frac{y_i x_i}{1+e^{y_i(w^{*T}x_i+b^*)}}$$

$$\Rightarrow \quad 2\lambda w^* = \frac{1}{n} \; X^T \begin{bmatrix} \dfrac{y_1}{1+e^{y_1(w^{*T}x_1+b^*)}} \\ \vdots \\ \dfrac{y_n}{1+e^{y_n(w^{*T}x_n+b)}} \end{bmatrix}$$

where $X^T = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}$

$$\Rightarrow \quad w^* = X^T \underbrace{\left[\frac{1}{2\lambda n} \begin{bmatrix} \dfrac{y_1}{1+e^{y_1(x_1 w^{*T}x_1+b^*)}} \\ \vdots \\ \dfrac{y_n}{1+e^{y_n(w^{*T}x_n+b^*)}} \end{bmatrix} \right]}_{\alpha^*}$$

$$\therefore \quad \boxed{w^* = X^T \alpha^*} \quad \text{for some } \alpha^*$$

Q1

(c) (a) $w^{old} = X^T \alpha^{old}$

We know from Q1 (a) update formula

$$w^{new} = w^{old} - \eta \left[\frac{1}{n} \sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i(\underbrace{w^{oldT}}_{\text{This is } (w^{old})^T} x_i+b^{old})}} + 2\lambda w^{old} \right] \quad - (1)$$

Let us substitute $w^{old} = X^T \alpha^{old}$ in eqn (1)

$$w^{new} = X^T\alpha^{old} - \eta\left[\underbrace{\frac{1}{n}\sum \frac{-y_i x_i}{1+e^{y_i((X^T\alpha^{old})^T x_i + b^{old})}}} + 2\lambda X^T\alpha^{old}\right] \quad -(2)$$

Let us try to analyze this term

$$\Rightarrow \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i(\alpha^{old\,T} X x_i + b^{old})}} \Rightarrow \frac{1}{n} X^T \begin{bmatrix} \frac{-y_1}{1+e^{y_1(\alpha^{old\,T} X x_1 + b^{old})}} \\ \vdots \\ \frac{-y_n}{1+e^{y_n(\alpha^{old\,T} X x_n + b^{old})}} \end{bmatrix} \quad -(3)$$

Substituting eqn (3) back in (2)

$$w^{new} = X^T\alpha^{old} - \eta\left[\frac{1}{n} X^T \begin{bmatrix} \frac{-y_1}{1+e^{y_1(\alpha^{old\,T} X x_1 + b^{old})}} \\ \vdots \\ \frac{-y_n}{1+e^{y_n(\alpha^{old} X x_n + b^{old})}} \end{bmatrix} + 2\lambda X^T \alpha^{old}\right]$$

$$\Rightarrow w^{new} = X^T \underbrace{\left[\alpha^{old} - \eta\left[\frac{1}{n}\begin{bmatrix} \frac{-y_1}{1+e^{y_1(\alpha^{old\,T} X x_1 + b^{old})}} \\ \vdots \\ \frac{-y_n}{1+e^{y_n(\alpha^{old\,T} X x_n + b^{old})}} \end{bmatrix} + 2\lambda\alpha^{old}\right]\right]}_{\alpha^{new}}$$

$$\boxed{\begin{aligned} &\Rightarrow w^{new} = X^T \alpha^{new} \\ &\text{where} \quad \alpha^{new} = \left[\alpha^{old} - \eta\left[\frac{1}{n}\begin{bmatrix} \frac{-y_1}{1+e^{y_1(\alpha^{old\,T} X x_1 + b^{old})}} \\ \vdots \\ \frac{-y_n}{1+e^{y_n(\alpha^{old\,T} X x_n + b^{old})}} \end{bmatrix} + 2\lambda\alpha^{old}\right]\right] \end{aligned}} \rightarrow \text{update}$$

Q1 (d) From Q1(c) we know that

$$\alpha^{new} = \alpha^{old} - \eta \left[\frac{1}{n} \begin{bmatrix} \dfrac{-y_1}{1+e^{y_1(\alpha^{old\,T} X x_1 + b^{old})}} \\ \vdots \\ \dfrac{-y_n}{1+e^{y_n(\alpha^{old\,T} X x_n + b^{old})}} \end{bmatrix} + 2\lambda\alpha^{old} \right] \quad -(1)$$

Let us just examine a general term in this vector

$$\Rightarrow \frac{-y_i}{1+e^{y_i(\alpha^{old\,T} X x_i + b^{old})}}$$

we see that Xx_i is an inner product

(ie) $$X x_i = \begin{bmatrix} x_1^T x_i \\ x_2^T x_i \\ \vdots \\ x_n^T x_i \end{bmatrix} \quad -(2)$$

This inner product can be replaced by using a kernel.

$$X x_i = \begin{bmatrix} k(x_1, x_i) \\ k(x_2, x_i) \\ \vdots \\ k(x_n, x_i) \end{bmatrix} = K(X, x_i) \quad -(3)$$

I am denoting this column vector as $K(X, x_i)$

eq ③

Now substituting the same, in back in eqn ①

we get

$$\alpha^{new} = \alpha^{old} - \eta \left[\frac{1}{n} \left[\frac{y_i}{1+e^{y_i(\alpha^{oldT}K(X,x_i)+b^{old})}} \right]_{i=1\ldots n} + 2\lambda\alpha^{old} \right]$$

also

This is b^{old}

$$b^{new} = b^{old} - \eta \sum_{i=1}^{n} \frac{-y_i}{1+e^{y_i((\alpha^{old})^T K(X,x_i)+b^{old})}}$$

Thus the update rule is kernelized.

We can set α^{old} to zero vector to start with and keep updating a^{new}, b^{new} till we end up with (α^*, b^*), the we know $w^* = X^T\alpha^*$

Final classifier,

$$\eta(y=1|x) = \frac{1}{1+e^{-(\alpha^{*T}K(X,x_i)+b^*)}}$$

This is the probability that class $=1$,

and

$$\begin{cases} \eta(y=1|x) \geq \frac{1}{2} & \text{predict class} = 1 \\ \eta(y=1|x) < \frac{1}{2} & \text{predict class} = -1 \end{cases}$$

This is -1

Kernel Logistic Regression

Q 1(e) Accuracy = 0.796

Error => 1 - 0.796 => 0.204

Q 2(a) Statistics X_mean = [0.50161345 , 0.45612671, 0.38244407]

X std = [0.24617303 , 0.23615181 , 0.23905821]

(1)
• Resizing:

✓ Benefit – resizing is useful to have images with the same pixels in the data set
The raw dataset might contain different images with different dimensions
for example 1000 × 800 , 200 × 400 , etc , but resizing helps us
to work with uniform dimensions through out data set.

✓ Draw back: By doing resizing, some important features are interpolated,
and also, when ever we resize a smaller image, we would create a
padding around, it which migh mis lead our classifier, if there are
several images with padding.

• Normalizing:

✓ Benifit: Normalizing the pixel intensities, will help the convergence of the
optimizer, there wont be much of ~~zig~~ zig-zagging during optimization. The
~~Drawback~~ centered data will keep the gradients in control when we use back propagation

Normalizing data

Drawback - Normalizing data can sometimes lead to loss of information.

For example, Normalizing just preserves relative variations in data.

If we want to classify day and night images; normalizing our data will affect the classification performance as it only captures relative information

Q2(a)

(2) per-channel mean and standard deviation are use to scale our training data. Validation set is not a part of our training data, when we scale our training data using training per channel mean and per-channel standard deviation, it helps us center the data for training, but during validation and testing phase we should use this same scaling as the neural network that is trained is learnt for the information scaled this particular way. Hence, it doesnot make sense to use a different scaling metric for validation images.

Q 2 (b)

(1) Layer - 1

Filter = 5x5

Input channel = 3

Output channel = 16

No. of parameters $\Rightarrow$ $(5\times5\times3 + \underbrace{1}_{\text{This is for bias}})\times 16$

$\boxed{= 1216}$ Convolution layer-1

Layer-2

Filter = 5x5

Input channel = 16

Output channels = 32

No. of parameters $\Rightarrow$ $(5\times5\times16 + \underbrace{1}_{\text{Bias}})\times 32$

$\boxed{\Rightarrow 12832}$ Convolution layer-2

Layer-3

Filter = 5×5

Input channel = 32

Output channel = 64

No. of parameters = $(5 \times 5 \times 32 + \overset{\text{Bias}}{1}) \times 64$

$= \boxed{51264}$ - convolution layer-3

Layer-4

Filter = 5×5

Input channel = 64

Output channels = 128

No. of parameters = $(5 \times 5 \times 64 + \overset{\text{Bias}}{1}) \times 128$

$= \boxed{204,928}$ - convolution layer-4

Layer-5 (Fully connected

Inputs = 128×2×2

Output = 64

No. of parameters = $(128 \times 2 \times 2 + \overset{\text{Bias}}{1}) \times 64$

$= \boxed{32,832}$ - fully connected layer-1

Layer-6 (Fully connected)

Inputs = 64

Outputs = 5

No. of parameters = $(64 + \overset{\text{Bias}}{1}) \times 5$

$= \boxed{325}$ - fully connected layer-2

Total parameters =

1216 +
12832 +
51264 +
204928 +
32832 +
325

303397

Q2(b) Q. One reason as to why we don't want to initialize our neural network to zero is because, when intialized with 0's then all the layers perform the same calculation, there wont be any symmetry breaking, the gradients computed take the learning no where, This is the reason why we should initialize the network weights randomly.

Q2 (d) Final accuracies and Losses

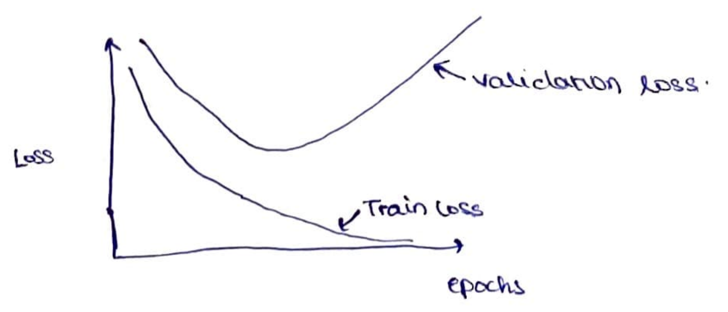
Validation Loss :- 2.3131

Validation accuracy :- 0.4872

Train Loss :- 0.007707

Train accuracy :- 1

(e) 1) The training Loss keeps decreasing, and the validation Loss drops at first and then starts increasing after a point.



As we keep on training the model our training loss would be going to zero, while our validation loss will shoot up to higher values.

This means we have done an overfit to our data.

Q2 (e) a. We should always try to minimize Validation loss so the training loss.

Based on the plot, we should stop training our model after 2 epochs as after 2 epochs the validation loss starts increasing.

We should not try to maximize training accuracy because, it leads to overfitting for the data, and the model learnt, won't be a good generalization of the true nature of data.

~~Q2 (g)~~ Q2(f) Final accuracies and Losses:

Validation Loss :- 0.18

Validation accuracy = 0.935897

train Loss : 0.11598

train accuracy: 0.956

Q2 (g) We can't use accuracy during an im balance dataset case because, it is not fully reflective of the correct classification for each class. Per-class accuracy metric makes more sense in this scenario to asses how good our model is.

Q 2 (h)

(Non-Weighted) Loss Imbalance dataset (5 epochs)	Weighted Cross entropy Loss Imbalance dataset (5 epochs)
Validation loss :- 0.1246	Validation Loss :- 0.12327
Validation accuracy :- 0.9545	Validation accuracy :- 0.963636
train Loss : 0.03430	train loss :- 0.047591
train accuracy : 0.99046	train accuracy :- 0.9857142
Per class accuracy :- [1 0.5] (Cat, dog (+ve class))	Per class accuracy :- [0.98 0.8] (Cat, dog)
(dog +ve class) precision = 1	Precision = 0.8
recall = 0.5	Recall = 0.8
F1-Score = 0.66666	F1 Score = 0.8 (dog is +ve class)

The un-weighted model has more training accuracy and train loss, but as the data set is imbalanced we can see it performs poorly on other metrics like F-1 score, recall, per-class accuracy. What was happening here is that the model was being overfit for the features of Cat as the examples are more in number.

In the weighted case, it can be observed that weighting dog class by a factor improved the performance of the model, this can be seen from metrics like F1 score, precision, recall. Here due to weight, the ~~lean~~ model was able to distinguish features of dog from that of cat, though the examples are pretty less in number.