CMSC 678/Ferraro Exam 1 March 30, 2023–April 3, 2023

Name: Faisal Rasheed Khan

ID: VB02734

Question 1----------

1.

a.

X	output class
(0,0)	1
(1,1)	1
(0,1)	2
(2,1)	2
(1,0)	3
(1,2)	3

b. initialize all the weights to 0, we need a 2x3 weight matrix for the input size 2x1.

$w^T x = 3x1$

w = [[0,0], [0,0], [0,0]]

argmax(w * [0,0]) = argmax([0, 0, 0]) = 1 , no update

argmax(w * [1,1]) = argmax([0, 0, 0]) = 1 , no update

argmax(w * [0,1]) = argmax([0, 0, 0]) = 1 , update

w = [[0,0] – [0,1], [0,0] + [0,1], [0,0] – [0,1]]

w = [[0,-1], [0,1], [0,-1]]

argmax(w * [2,1]) = argmax([-1, 1, -1]) = 2 , no update

argmax(w * [1,0]) = argmax([0, 0, 0]) = 1 , update

w = [[0,-1] – [1,0], [0,1] - [1,0], [0,-1] + [1,0]]

w = [[-1,-1], [-1,1], [1,-1]]

argmax(w * [1,2]) = argmax([-3, 1, -1]) = 2 , update

w = [[-1,-1] – [1,2], [-1,1] - [1,2], [1,-1] + [1,2]]

w = [[-2,-3], [-2,-1], [2,1]]

c. We arrive at the classic perceptron algorithm using ERM by using the loss function and make it differentiable over the loss function.

The ERM is defined as average over the sum of the loss over input.

ERM = $\frac{1}{N}\sum L(y, \hat{y})$; y – actual label, $\hat{y}$ – predicted label

The classic perceptron uses surrogate loss function, hinge loss.

The Hinge loss penalizes the function when it makes misclassifications. The Hinge loss updates the weights when the model makes incorrect predictions.

L(w) = $\sum_n max\ (0,1-y_n w^T x_n)$

Its not differentiable, so we define subgradients to make it differentiable.

$\frac{dl^{hinge}}{dw} = 0$ if $y\ w^T x > 1$

-yx otherwise

$\frac{dL}{dw} = \sum_n -1\ [y_n w^T x_n]\ y_n x_n$

w <- w – n ($\sum_n -1\ [y_n w^T x_n]\ y_n x_n$)

w <- w + n $y_n x_n$ only for points $y_n w^T x_n \leq 1$

perceptron update $y_n w^T x_n \leq 0$, n=1

w <- w + $y_n x_n$

Question 2----------

2.

a. Both softmax and sigmoid has different decoding rules.

Softmax takes a vector input of size L, and returns L dimension vector consisting of scores of different classes.

softmax(u), $u \in R^L$,

The output class score which has maximum probability is predicted

$P(y \mid x) \alpha \exp(\theta f(x))$

y=argmax(softmax(u))

The softmax function sums all the class probabilities to 1

The sigmoid function considers individual class score and sends the dot product of weights and input through the sigmoid activation function.

If the score of the sigmoid is >=0.5 that class is predicted

$\sigma(x) = \frac{1}{1+e^{-x}}$

$max_x \sum_i p(\hat{y}i = yi \mid xi)$

b. Softmax develops a probabilistic classifier over L weights such that the data is nicely distributed and all the probabilities sums to 1.

Softmax normalizes input across all the different labels that computes each class score which sums to 1.

Softmax gives a more mathematical correct representation and balanced b/w [0,1]

Softmax is the generalization of sigmoid applies to the output score rather than applying sigmoid to every score we are computing.

Sigmoid doesn't develop classifier across all the labels, it just computes individual label scores across the input.

$\sigma(x) \geq 0.5$ then $x \geq 0$

softmax([x,0])=$[\frac{\exp(x)}{\exp(x)+\exp(0)}, \frac{\exp(0)}{\exp(x)+\exp(0)}]$

if x>0 then 1st coordinate and if x<0 then 2nd coordinate

Sigmoid is the simplified version of softmax based on the 2nd coordinate of value 0

c. Option 2 and option 3 are same if we implement one vs all classifier using cross entropy loss, it uses the decision function as same as of the sigmoid.

Multiple binary classifiers are trained and which ever score is maximum, its label id predicted.

Train c different classifiers $Y_c(x)$

$Y_c(x)$ predicts 1 if x is likely of class c, 0 otherwise

Predicting new instance z,

Scores $s^c = Y_c(z)$

$\hat{y}$=argmax s^c

Option 2 and option 3 are different if we implement one vs all classifier using loss function. Hinge loss is used for the perceptron update.

L(w) = $\sum_n max\,(0,1-y_n w^T x_n)$

Hinge loss is not differentiable, but we make it differentiable using subgradients.

d. consider 3 class labels l_1 ,l_2 ,l_3 and the data which is balanced.

l_1 , l_1 , l_1 , l_1 , l_1

l_2 , l_2 , l_2 , l_2 , l_2

l_3 , l_3 , l_3 , l_3 , l_3

l_1 has 5 correct label instances and 10 incorrect label instances, similarlily its same for l_2 and l_3 .

Training classifiers provides heavily imbalanced class views.

The classifiers doesn't get learned properly

Dataset is balanced but the classifier is not balanced.

Data redistribution issues.

We can overcome this by down sampling the instances.

Question 3----------

3.

a. The connection between minimizing cross entropy and optimizing expected posterior 0-1 loss is that both are the probability classifiers for categorical labels. They both works as discrete classifiers over the input. Cross entropy is the lower bound of posterior 0-1.

The posterior 0-1 loss considers the prior distribution while predicting the new input. So the prior distribution it considers is of Bayesian decision theory. The Optimizing expected posterior 0-1 loss minimizes the misclassified data in order to improve its decision.

$min_w \sum_i E_{\hat{y}_i \sim p(. \mid x_i)}[l(y_i, \hat{y}_i)] = min_w \sum_i \sum_j (1 - \delta_{y_i j}) p(\hat{y}_i = j \mid x_i)$

$kronecker\ delta$: 1 if y_i = j, 0 otherwise

$min_w \sum_i [\, 1 - \sum_j (\delta_{y_i j}) p(\hat{y}_i = j \mid x_i) \,]$

$$max_w \sum_i p(\hat{y}_i = j \mid x_i)$$

On the other hand, cross entropy measures the uncertainty over the input. Minimizing cross entropy loss ensures the removal of uncertainty to the input and considers maximizing the probability for the true label. minimize cross entropy loss measures the expected probability distribution to actual probability distribution. By minimizing the cross entropy we see how likely is the label associated to it. Minimizing the cross entropy loss considers maximizing the log likelihood. By this we get high probability score for the correct label i.e maximizing the likelihood.

$l^{xent} = -\sum_{j \in y} \vec{y}^*[j] \log p(y = j|xi)$.

b. Given the probabilistic classifier model p(y|x) = softmax(θf(x)) trained by maximizing the posterior label likelihood i.e it considers the prior probability p(y) and checks the likelihood p(x | y) which means how well does input x represent label y

These likelihood is given by Bayes classification p(y|x) = $\frac{p(x \mid y) * p(y)}{p(x)}$

So, the learned classifier is softmax, it sums all the probabilities to 1.

Θ – weights to be learned

f(x) – features of the input

softmax normalizes the input and it contains scores in the range [0,1], and its sum of all scores = 1

To respect the constraints, the model should remove uncertainty from the input and maximize the probability of the true label.

Maximizing the probability of true labels, minimizes the other probabilities, as softmax sums to 1.

We consider cross entropy here. Cross entropy measures uncertainty over the input. Minimizing cross entropy loss helps in removing uncertainty over the input and maximizing the log likelihood of the true label.

$l^{xent} = -\sum_{j \in y} \vec{y}^*[j] \log p(y = j|xi)$.

The gradients of the cross entropy loss helps in adjusting the weights θ.

Question 4----------

4.

a. Y takes the values of 0,1,2,3,……..

The range consists of non-negative integers.

b. this discrete classifier approach is not a good option because the value of Y is not discrete value, it is continuous value.

The conditional classifier may not be able to predict the values accurately as it is not categorical and also the range of the discrete value is finite but we have continuous value feeding the classifier.

We get errors in predictions if we use discrete classifiers.

c. ϕ(X) is a k-dimensional, so the weight vector also takes k-dimensional in order to be learned for this model

d. $l(y^*, \hat{y}) = \hat{y} - y^* \log \hat{y} - \log((y^*)!)$

y^* – actual value

$\hat{y}$ – predicted value

The learning weights depend on the predicted value which is $\hat{y}$

$\hat{y}$ and $y^* \log \hat{y}$ are the terms that depends on the weights to be learned.

e. $l = \exp(w^T \phi(X)) - y^* \log(\exp(w^T \phi(X))) - \log((y^*)!)$

$l = \exp(w^T \phi(X)) - y^* w^T \phi(X) - \log((y^*)!)$

f. $L(w) = \frac{1}{N}\sum_{i=1}^{N} l(y_i^*, \hat{y}_i)$

$L(w) = \frac{1}{N}\sum_{i=1}^{N} \exp(w^T \varphi(x_i)) - y_i^* w^T \varphi(x_i) - \log((y_i^*)!)$

g. So, we have N=250000 labelled tweets.

We perform data cleaning on this N labelled tweets, if it has inconsistent data or duplicate data.

Next, we extract the useful features of our data. What features to include and what to not include.

Next, we use poisson regression model to train our data. Split your data into train, dev, test.

Adjust the weights while training the model, such that the weights are optimized.

The weights are optimized by using gradient of the given loss function.

For minimizing difference between actual and predicted value use loss function, here we use loss function given to us.

$$l(y^*, \hat{y}) = \hat{y} - y^* \log \hat{y} - \log((y^*)!)$$

With the help of the Evaluation metrics evaluate the performance on the model created. We can check the performance using accuracy, precision, recall.

References:

04-linear-perceptron-grad.pdf (umbc.edu)

07-maxent.pdf (umbc.edu)

05-beyond-binary-classification.pdf (umbc.edu)

https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf

http://www.inference.phy.cam.ac.uk/itprnn/book.pdf