SHUBH JIGNESH VANIA

MACHINE LEARNING
SPRING 2024 ASSNG-1

Q.1) How many parameters are there in Logistic regression?

→ (2-D Vectors) In logistic regression, there are following parameters:

- The intercept term (bias)
- Two coefficients corresponding to the features.

Q.2) Write down the loss as a function of parameters.

→ The logistic regression loss function (cross-~~entropy~~ entropy loss) for a dataset D with N samples is given by summing the loss over all samples:

$$L(x) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(\sigma(wx_i)) + (1-y_i)\cdot\log(1-\sigma(wx_i))\right]$$

Here, w is the parameter vector, x_i is the feature vector for the i-th sample, $\sigma(z)$ is the sigmoid function, and y_i is the target label for the i-th sample.

Q.3) Calculate the partial derivative of the loss function with respect to each parameter:

→ Derivation for loss function:-

$$L(x) = -[y \cdot \log(\sigma(wx)) + (1-y) \cdot \log(1-\sigma(wx))]$$

$\sigma(d)$ is the sigmoid function, $\sigma(d) = \frac{1}{1+e^{-d}}$

x = feature vetor w = parameter vector

($d = w.x$)

with respect to w_0:

$$\frac{\partial L}{\partial w_0} = -\left(y \cdot \frac{\sigma'(d)}{\sigma(d)} - (1-y) \cdot \frac{\sigma'(d)}{1-\sigma(d)}\right) \cdot \frac{\partial d}{\partial w_0}$$

as $\frac{\partial d}{\partial w_0} = 1$

$$\frac{\partial L}{\partial w_0} = -(y \cdot (1-\sigma(d)) - (1-y) \cdot \sigma(d))$$

with respect to w_1:

$$\frac{\partial L}{\partial w_1} = -\left(y \cdot \frac{\sigma'(d)}{\sigma(d)} - (1-y) \cdot \frac{\sigma'(d)}{1-\sigma(d)}\right) \cdot \frac{\partial d}{\partial w_1}$$

$$\frac{\partial d}{\partial w_1} = x_1$$

$$\frac{\partial L}{\partial w_1} = -(y \cdot (1 - \sigma(d)) - (1-y) \cdot \sigma(d)) \cdot x_1$$

with respect to w_2:

$$\frac{\partial L}{\partial w_2} = -\left(y \cdot \frac{\sigma'(d)}{\sigma(d)} - (1-y) \cdot \frac{\sigma'(d)}{1-\sigma(d)}\right) \cdot \frac{\partial z}{\partial w_2}$$

$$\frac{\partial d}{\partial w_2} = x_2$$

$$\frac{\partial L}{\partial w_2} = -(y \cdot (1 - \sigma(d)) - (1-y) \cdot \sigma(d)) \cdot x_2$$

Where $\sigma'(d)$ is derivative of sigmoid function,

and $\sigma'(d) = \sigma(d) \cdot (1 - \sigma(d))$.

Q.4) Write the iterative gradient descent update algorith, assuming a step size η.

→ Iterative gradient descent update Algo:-

Define sigmoid function:-

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

Cross-entropy loss function:

$$L(x) = -[y \cdot \log(\sigma(wx)) + (1-y) \cdot \log(1-\sigma(wx))]$$

parameter vector - w

learning rate - η

For each iteration we have to compute the gradient of the loss w.r.t to w_i

$$\frac{\partial L}{\partial w_i} = \frac{1}{N} \sum_{j=1}^{N} (\sigma(w x^{(j)}) - y^{(j)}) \cdot x^{(j)}$$

Update →

$$w_i^{(t+1)} = w_i^{(t)} - \eta \cdot \frac{\partial L}{\partial w_i}$$

The update equation for each parameter w_i states that the new value ($w_i^{(t+1)}$) is equal to its current value ($w_i^{(t)}$) minus a small step in direction that decreases the loss.

Q.5) Assuming that we calculate stochastic gradient, i.e, pick a random point and compute the gradient with respect to that point only. Write down the algorithm for the ADAM update rule.

→

```
import math
def initialize_adam_parameters(num_params):
    we will initialize parameters
        beta1, beta2, epsilon,
        m(first moment vector),
        v(second moment vector)
        t(time step.

def adam_update_rule

    update time step
        t.t+=1

-update moment estimates

- correct bias in moment estimates.

Upate parameters using the Adam formula:

parameters = [parameters[i] - learning_rate * m_corrected[i]
    /(math.sqrt(v_corrected[i]) + epsilon) for i in range
                                        (len(parameter)]

return parameters m, v, t.
```

- Stochastic gradient calculation - replace this with your actual gradient calculation

- Update parameters using the ADAM update rule..

Q.2) A short conclusion:-

I ~~observe~~ observed that step size plays an important role

In large step size - loss drops very quickly close to 0.
But as iterations progressed, there are small spikes/increaments in the loss ~~grad~~ graph. I think this is because large step size can give some oscillations around the local minima. However, I observed ADAM and the full gradient descent have the same performance overall. For small data, full gradient descent was convinient but in real-life ADM would be better.

Q.3) A short conclusion: For logistic regression as classifier in double descent takes time and crashes system so I chose SGD classifier. In my graph, train error increases after some features because we have lot of features to train.