4. Multinomial (Softmax) Regression and Gradient Descent

Project due Mar 15, 2023 08:59 -03 Completed

Daniel suggests that instead of building ten models, we can expand a single logistic reg
multinomial regression and solve it with similar gradient descent algorithm.

The main function which you will call to run the code you will implement in this section
`run_softmax_on_MNIST` in main.py (already implemented). In the appendix at the bott
describe a number of the methods that are already implemented for you in `softmax.p`

In order for the regression to work, you will need to implement three methods. Below w
functions should do. We have included some test cases in `test.py` to help you verify t
have implemented are behaving sensibly.

You will be working in the file `part1/softmax.py` in this problem

Computing Probabilities for Softmax

5.0/5.0 points (graded)

Write a function `compute_probabilities` that computes, for each data point $x^{(i)}$, the
labeled as j for $j = 0, 1, \ldots, k-1$.

The softmax function h for a particular vector x requires computing

$$h(x) = \frac{1}{\sum_{j=0}^{k-1} e^{\theta_j \cdot x/\tau}} \begin{bmatrix} e^{\theta_0 \cdot x/\tau} \\ e^{\theta_1 \cdot x/\tau} \\ \vdots \\ e^{\theta_{k-1} \cdot x/\tau} \end{bmatrix},$$

where $\tau > 0$ is the **temperature parameter** . When computing the output probabilities
in the range $[0, 1]$), the terms $e^{\theta_j \cdot x/\tau}$ may be very large or very small, due to the use of
function. This can cause numerical or overflow errors. To deal with this, we can simply
amount c from each exponent to keep the resulting number from getting too large. Sinc

$$h(x) = \frac{e^{-c}}{e^{-c} \sum_{j=0}^{k-1} e^{\theta_j \cdot x/\tau}} \begin{bmatrix} e^{\theta_0 \cdot x/\tau} \\ e^{\theta_1 \cdot x/\tau} \\ \vdots \\ e^{\theta_{k-1} \cdot x/\tau} \end{bmatrix}$$

$$= \begin{bmatrix} e^{\theta_0 \cdot x/\tau - c} \end{bmatrix}$$

```
    for j = 0, 1, ..., k-1
    Args:
        X - (n, d) NumPy array (n datapoints each with d features)
        theta - (k, d) NumPy array, where row j represents the parameters o
        temp_parameter - the temperature parameter of softmax function (sca
    Returns:
        H - (k, n) NumPy array, where each entry H[j][i] is the probability
    """
    H = np.empty((theta.shape[0], X.shape[0]))
    temp = np.empty((theta.shape[0],))
    for i in range(X.shape[0]):
        temp[:] = (theta @ X[i,:].reshape(X.shape[1],1))[:,0]
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

CORRECT

Submit You have used 4 of 25 attempts

Softmax: Overview and layman perspective

0:00 / 0:00 1.0x

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Cost Function

3.333333333333333/5.0 points (graded)
Write a function `compute_cost_function` that computes the total cost over every data

The cost function is given by: (Use natural log)

Available Functions: You have access to the NumPy python library as `np` and the pre
`compute_probabilities`

```
def compute_cost_function(X, Y, theta, lambda_factor, temp_parameter):
    """
    Computes the total cost over every datapoint.

    Args:
        X - (n, d) NumPy array (n datapoints each with d features)
        Y - (n, ) NumPy array containing the labels (a number from 0-9) fo
```

Submit You have used 1 of 25 attempts

Gradient Descent

5.0/5.0 points (graded)

Solution to this problem available before due date: The function `run_gradient_desce` necessary for the rest of the project. Hence, once you have either submitted the correc your attempts for this problem, the solution to this function will be available.

Now, in order to run the gradient descent algorithm to minimize the cost function, we n derivative of with respect to a particular . Notice that within , we have:

so we first compute: ,

when ,

when ,

Now we compute

you previously implemented and `scipy.sparse` as `sparse`.

You should use `sparse.coo_matrix` so that your function can handle larger matrices e
out for the online graders). The sparse matrix representation can handle sparse matrice

Hint

```
def run_gradient_descent_iteration(X, Y, theta, alpha, lambda_factor, temp
    """
    Runs one step of batch gradient descent

    Args:
        X - (n, d) NumPy array (n datapoints each with d features)
        Y - (n, ) NumPy array containing the labels (a number from 0-9) fo
            data point
        theta - (k, d) NumPy array, where row j represents the parameters
                model for label j
        alpha - the learning rate (scalar)
        lambda_factor - the regularization constant (scalar)
        temp_parameter - the temperature parameter of softmax function (sc

    Returns:
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

CORRECT

Submit You have used 1 of 20 attempts

Test Error on Softmax Regression

1.0/1.0 point (graded)

Finally, report the final test error by running the `main.py` file, using the temperature p
have implemented everything correctly, the error on the test set should be around 0.1, v
softmax regression model is able to recognize MNIST digits with around 90 percent acc

Note: For this project we will be looking at the error rate defined as the fraction of labe
target labels, also known as the "gold labels" or ground truth. (In other contexts, you m

Previous Next

Show all posts

- Using Numpy functions instead of sparse.coo
  It's mentioned that this can be done with numpy alone; Has anyone managed to do that?
- Derivation of Gradient of the Loss Function
- Ability to be unsure
- [STAFF]: Test error on Softmax regression = CORRECT, yet the "compute_cost_function()" is incorrect!?
  My code for "compute_probabilities()" and "run_gradient_descent_iteration()" functions passed both the Gra
- Second video : nice video, but bad way explaining chain rule
- Exact formula for Loss ?? -log(p) ??
  I can follow the chain rules but from where does it comes that he says : the loss function - -log(prob)? I found
- Help - stuck at 4 part 1 on partial error related to temp paramenter
  I get partial credit (3.9/5) for my solution (haven't cracked vectorise yet, too much on the past two weeks) w
- Hint for Q2
  1. If you do not match the second output for case2, it is very likely that your regularization is wrong. Norm in
- Why I get lots of "nan" in Q1?
  I got most of the cases right, except for some cases I got lots of nan in the results. I use matrix method to de
- Gradient descent function
  I am stuck while implementing the gradient descent function. Could any one help me in solving this??
- understanding Sam's code guidelines
  In his quick and dirty programming example, Sam finds the gradient for dl_dz. I believe we want dl_dtheta. A
- Recommendation for Q2) Cost Function
  For everyone struggling same as me, I would suggest you try to implement both terms (Loss and Regularizat
- Softmax: Overview and 5 Different Perspectives. Hinge Loss as probabilistic model
  Hi Sam, thank you so much for your help on the forum and with the videos. You said in the Softmax video to

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Do Not Sell My Personal Information

Connect

Blog
Contact Us
Help Center
Security
Media Kit

深圳市恒宇博科技有限公司 粤ICP备17044299号-2