Completed • $20,000

Predict Closed Questions on Stack Overflow

Tue 21 Aug 2012 – Sat 3 Nov 2012 (2 years ago)

« Prev Topic

Multi Class Log Loss Function

Next Topic »

Start Watching

0

Is the following Python code an accurate representation of how submissions are evaluated? I've played with this to help me evaluate my modelling, but wanted to make sure I understood how the evaluator worked. I believe I'll need to add a PostId to the prediction data when I submit, but have not included that for simplicity's sake in this example code.

```
from __future__ import division

import csv
import os
import scipy as sp

def llfun(act, pred):
    epsilon = 1e-15
    pred = sp.maximum(epsilon, pred)
    pred = sp.minimum(1-epsilon, pred)
    ll = sum(act*sp.log(pred) + sp.subtract(1,act)*sp.log(sp.subtract(1,pred)))
    ll = ll * -1.0/len(act)
    return ll

def main():
    pred = [
        [0.05,0.05,0.05,0.8,0.05],
        [0.73,0.05,0.01,0.20,0.02],
        [0.02,0.03,0.01,0.75,0.19],
        [0.01,0.02,0.83,0.12,0.02]
        ]
    act = [
           [0,0,0,1,0],
           [1,0,0,0,0],
           [0,0,0,1,0],
           [0,0,1,0,0]
           ]
    scores = []
    for index in range(0, len(pred)):
        result = llfun(act[index], pred[index])
        scores.append(result)

    print(sum(scores) / len(scores)) # 0.0985725708595

if __name__ == '__main__':
    main()
```

#1 | Posted 3 years ago

Matthew Lesko

1

```
ll = sum(act*sp.log(pred) + sp.subtract(1,act)*sp.log(sp.subtract(1,pred)))
```

That's not right. Since the prediction is a normalized multinomial distribution, you just take log(pred[label]), and ignore the other predictions not covered by the label (their impact on the score is via the normalization). If your prediction is not actually normalized, you need to normalize it (after clamping to 1e-15).

#2 | Posted 3 years ago

Andy Sloane

5

Here's the function I use:

```
import numpy as np

def multiclass_log_loss(y_true, y_pred, eps=1e-15):
    """Multi class version of Logarithmic Loss metric.
    https://www.kaggle.com/wiki/MultiClassLogLoss

    idea from this post:
    http://www.kaggle.com/c/emc-data-science/forums/t/2149/is-anyone-noticing-difference-
betwen-validation-and-leaderboard-error/12209#post12209

    Parameters
    ----------
    y_true : array, shape = [n_samples]
    y_pred : array, shape = [n_samples, n_classes]

    Returns
    -------
    loss : float
    """
    predictions = np.clip(y_pred, eps, 1 - eps)

    # normalize row sums to 1
    predictions /= predictions.sum(axis=1)[:, np.newaxis]

    actual = np.zeros(y_pred.shape)
    rows = actual.shape[0]
    actual[np.arange(rows), y_true.astype(int)] = 1
    vsota = np.sum(actual * np.log(predictions))
    return -1.0 / rows * vsota
```

#3 | Posted 3 years ago

ephes

0

ephes wrote:

> Here's the function I use:
>
> ```
> import numpy as np
>
> def multiclass_log_loss(y_true, y_pred, eps=1e-15):
>     """Multi class version of Logarithmic Loss metric.
>     https://www.kaggle.com/wiki/MultiClassLogLoss
>
>     idea from this post:
>     http://www.kaggle.com/c/emc-data-science/forums/t/2149/is-anyone-noticing-difference-
> betwen-validation-and-leaderboard-error/12209#post12209
>
>     Parameters
>     ----------
>     y_true : array, shape = [n_samples]
>     y_pred : array, shape = [n_samples, n_classes]
>
>     Returns
>     -------
>     loss : float
>     """
>     predictions = np.clip(y_pred, eps, 1 - eps)
>
>     # normalize row sums to 1
>     predictions /= predictions.sum(axis=1)[:, np.newaxis]
>
>     actual = np.zeros(y_pred.shape)
>     rows = actual.shape[0]
>     actual[np.arange(rows), y_true.astype(int)] = 1
>     vsota = np.sum(actual * np.log(predictions))
>     return -1.0 / rows * vsota
> ```

What type of objects are the inputs?

> y_true : array, shape = [n_samples]
>
> y_pred : array, shape = [n_samples, n_classes]

I'm using a simple list of list and isn't working properly =(

thanks for any help

#4 | Posted 3 years ago

Alessandro Sena

3

The function assumes that two numpy ndarrays are supplied.

The first is a 1-d array, where each element is the goldstandard class ID of the instance.

The second is a 2-d array, where each element is the predicted distribution over the classes.

Here are some example uses:

```
>>> import numpy as np
>>> multiclass_log_loss(np.array([0,1,2]),np.array([[1,0,0],[0,1,0],[0,0,1]]))
2.1094237467877998e-15
>>> multiclass_log_loss(np.array([0,1,2]),np.array([[1,1,1],[0,1,0],[0,0,1]]))
0.36620409622270467
```

#5 | Posted 3 years ago

Marco Lui

Reply

+ Add attachment(s) ...

Post Reply ☑ Email me when someone replies

Start Watching « Back to forum