Multinomial Logistic Example

Jan. 31, 2020

Example

Suppose we're fitting a model to classify an animal as either dog, cat, or bird based on its weight:

$$y^{(i)} = \begin{cases} 1 \text{ if animial } i \text{ is a dog} \\ 2 \text{ if animial } i \text{ is a cat} \\ 3 \text{ if animial } i \text{ is a bird} \end{cases}$$

$x^{(i)}$ = weight of animal i in pounds

Suppose we have fit a model and obtained the following estimates:

- $b_1 = -4$, $w_{11} = 2.3$
- $b_2 = 0$, $w_{21} = 2$
- $b_3 = 10$, $w_{31} = -5$

Let's make a plot of the estimated probability of each class, as a function of the animal's weight x.

I have secretly defined (because I'm going to ask you to implement this function in your next homework) a function called `softmax` with the following docstring:

```
'''
Calculate softmax(z) where z is a K by M matrix

Arguments:
 - z, a K by M matrix: row j and column m of z contains b_j + w_j^T x^(i),
   the linear input to softmax for class j and observation number m

Return:
 - a K by M matrix where column m is calculated as softmax of column m of z
'''
```

Import libraries:

```
import numpy as np
import matplotlib.pyplot as plt
```

Define a grid of 101 values of x between 0 and 20 at which to compute class probabilities

```
x = np.linspace(0, 20, 101).reshape((1, 101))
print("x: " + str(x))
```

```
## x: [[ 0.   0.2  0.4  0.6  0.8  1.   1.2  1.4  1.6  1.8  2.   2.2  2.4  2.6
##    2.8  3.   3.2  3.4  3.6  3.8  4.   4.2  4.4  4.6  4.8  5.   5.2  5.4
##    5.6  5.8  6.   6.2  6.4  6.6  6.8  7.   7.2  7.4  7.6  7.8  8.   8.2
##    8.4  8.6  8.8  9.   9.2  9.4  9.6  9.8 10.  10.2 10.4 10.6 10.8 11.
##   11.2 11.4 11.6 11.8 12.  12.2 12.4 12.6 12.8 13.  13.2 13.4 13.6 13.8
##   14.  14.2 14.4 14.6 14.8 15.  15.2 15.4 15.6 15.8 16.  16.2 16.4 16.6
##   16.8 17.  17.2 17.4 17.6 17.8 18.  18.2 18.4 18.6 18.8 19.  19.2 19.4
##   19.6 19.8 20. ]]
```

Parameter values

```
b = np.array([[-4], [0], [10]])
print("b: " + str(b))
```

```
## b: [[-4]
##  [ 0]
##  [10]]
```

```
w_T = np.array([[2.3], [2], [-5]])
print("w_T: " + str(w_T))
```

```
## w_T: [[ 2.3]
##  [ 2. ]
##  [-5. ]]
```

Compute z and a (both are 3 by 101)

```
z = b + np.dot(w_T, x)
print("z shape: " + str(z.shape))
```

```
## z shape: (3, 101)
```

```
a = softmax(z)
print("a shape: " + str(a.shape))
```

```
## a shape: (3, 101)
```

Make a plot

```
plt.plot(x[0, :], a[0, :], c = "blue", label = "Probability of Dog")
plt.plot(x[0, :], a[1, :], c = "orange", label = "Probability of Cat")
plt.plot(x[0, :], a[2, :], c = "purple", label = "Probability of Bird")
plt.legend(loc = "upper right")
plt.show()
```