Numerical Methods

Math 3338 – Spring 2022

Worksheet 28

Logistic Regression

1 Reading

CP	-
NMEP	-

Table 1: Sections Covered

Consider the graph in Figure 1. You want to create a condition to separate the red and blue dots[1]. In

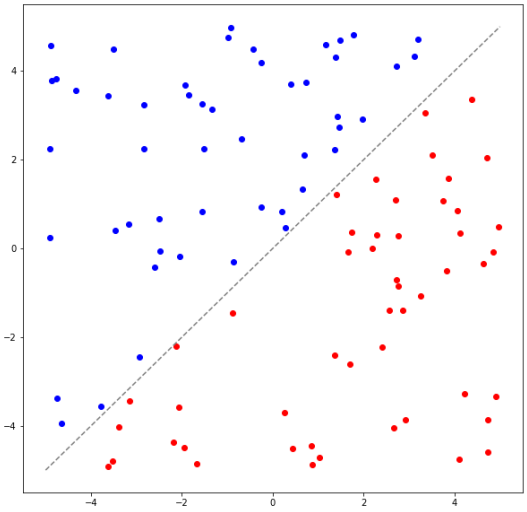

Figure 1: A basic classification problem.

this case the condition is the dots are red if $x > y$ and blue otherwise, it's a very simple condition. I put the line $y = x$ dashed to emphasize the boundary. However, if this is a much higher dimensional problem it becomes far more difficult. This is where *logistic regression* is useful.

Logistic regression is used to classify items that in two states, or binary problems. In this case we'll say the blue dots correspond it 1 and red is 0. So if we are given the point $(5, 2)$ that is 0 since $x > y$ and $(2, 8)$ is 1 since $x \leq y$. We are going to build a logistic regression model to separate these points.

Here is the big idea. We are going to create a function F that takes a point x (in our case $x = [x_1, x_2]$ since we have two coordinates) and outputs a number between 0 and 1. If $F(x) \geq .5$ then we'll say that x is a 1 and vice-versa. The easiest way to understand this is to build it in reverse. First we want a function that always returns a value between 0 and 1 and doesn't faff about. The answer is the sigmoid function as seen in Figure 2. The equation of this graph is

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

[1] Let's hope I didn't print this in black and white.

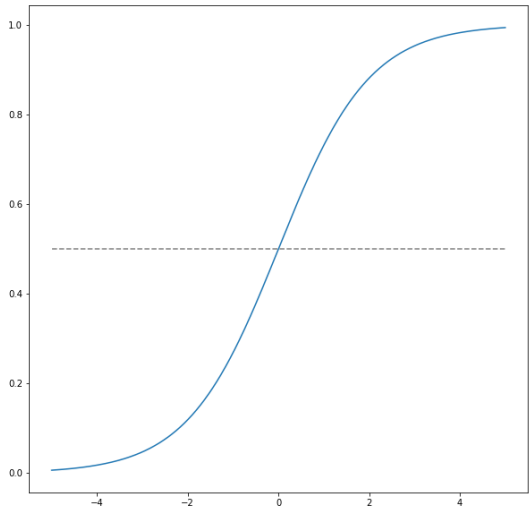

Figure 2: The sigmoid function.

Notice this function spends very little time between 0 and 1, that makes it ideal as a final step[2], it separates values effectively.

What do we put into the sigmoid? A weighted sum of the inputs. In our case that means something of the form $w_1x_1 + w_2x_2 + b$. What is the b? It's a bias term that helps correct the weighted sum, it tends to be useful. You probably recognized this as an offset plane, that's right it's essentially linear regression. How do we actually find the correct weights and bias? Optimization.

Let's start by saying $w_1 = w_2 = b = 1$ with an input of $x = [2, 1]$. We know $y = 0$ (this is the true output), let's make a prediction based on our current weights.

$$\begin{aligned}\hat{y} =& \sigma(1 \cdot 2 + 1 \cdot 1 + 1)\\ =& \sigma(4)\\ =& 0.9820137900379085\end{aligned}$$

This is wrong, we really wanted $\hat{y} \sim 0$. We need a function that will tell us how wrong we are. That turns out to be something called *binary crossentropy*.

$$\mathcal{L}(\hat{y}, y) = -y \ln(\hat{y}) - (1 - y) \ln(1 - \hat{y}) \tag{2}$$

Let's compute this for our previous prediction,

$$\begin{aligned}\mathcal{L}(0.9820137900379085, 0) =& -0 \ln(0.9820137900379085) - (1 - 0) \ln(1 - 0.9820137900379085)\\ =& 4.01814992791781\end{aligned}$$

This is fairly large. You should stop and think about why this is the "right" thing to use. Think about the possible values of y and what will happen when $y \sim \hat{y}$.

We want to minimize $\mathcal{L}(\hat{y}, y)$. We've done this in the past and we'll use gradient descent. Our variables are w and b, but it will be useful to have a couple of temporary variables as well $z = w \cdot x + b$ and $a = \sigma(z)$. The variable a isn't really useful right now, but it will be next time.

There is one final note to make before we start doing some real math. The standard notation. We will, by default, be caring about $\frac{d\mathcal{L}}{dv}$ where v is some variable. This tends to be quite a bit to type into computes

[2] tanh is another good option

and gets cumbersome. We will abbreviate $\frac{d\mathcal{L}}{dv}$ as just $\mathrm{d}v$. As a mathematician, this kinda hurts. But it does make some things nice.

We want to perform gradient descent on w and b. That means we will be updating $w = w - \alpha \mathrm{d}w$ and $b = b - \alpha \mathrm{d}b$, so we need to find $\mathrm{d}w$ and $\mathrm{d}b$. This just the chain rule,

$$\begin{aligned} \mathrm{d}w =& \frac{d\mathcal{L}}{dw} \\ =& \mathrm{d}z \frac{dz}{dw} \\ =& \mathrm{d}z * x \end{aligned} \qquad \begin{aligned} \mathrm{d}b =& \frac{d\mathcal{L}}{db} \\ =& \mathrm{d}z \frac{dz}{db} \\ =& \mathrm{d}z \end{aligned}$$

The $*$ is coordinate wise multiplication. Of course, now we need to find $\mathrm{d}z$.

$$\begin{aligned} \mathrm{d}z =& \frac{d\mathcal{L}}{dz} \\ =& \frac{d\mathcal{L}}{d\sigma}\frac{d\sigma}{dz} \\ =& \left(\frac{-y}{\sigma(z)} + \frac{(1-y)}{1-\sigma(z)}\right)\sigma(z)(1-\sigma(z)) \\ =& \sigma(z) - y \end{aligned}$$

Notice, $\mathrm{d}z$ is kinda $\hat{y} - y$, which it really should be.

We're almost done, but we've only done this for a single sample. In reality we'll have a lot[3] of samples. We need to generalize and make it vectorized. The main difference should be capital letters vs lower case.

$$X = \begin{bmatrix} x_1^1 & x_2^2 & \dots & x_1^m \\ x_2^1 & x_2^2 & \dots & x_2^m \end{bmatrix} \qquad W = [w_1, w_2] \qquad \vec{b} = [b]$$

The samples in X are arranged in columns, so there are m samples each with 2 observations. W is a row vector with 2 columns, which matches the number of observations. $\vec{b}$ is a vector with a single column, which matches the number of rows in W. Using these,

$$Z = W^T X + \vec{b}$$

that is a transpose on W. This is a vectorized operation, so W^T and $\vec{b}$ are being broadcast to the correct shapes. The loss function is just a summation,

$$\mathcal{L}(\hat{y}, y) = \sum_{i=1}^{m} \mathcal{L}(\hat{y}_i, y_i)$$

I abused notation here slightly by using $\mathcal{L}$ twice to mean different things. But oh well. Finally,

$$\mathrm{d}Z = \sigma(z) - y \qquad \mathrm{d}W = \frac{1}{m} dZ \cdot X^T \qquad \mathrm{d}b = \frac{1}{m} \sum dZ$$

The $\frac{1}{m}$ act as regularization to prevent the summations from blowing up.

[3] Potentially millions

Numerical Methods

Math 3338 – Spring 2022

Homework 28 (Due: Thursday, April 28)

Problem 1 (1 pt) Verify the derivatives in dz.

Problem 2 (1 pt) Make a function called `logistic_regression(X,y,alpha=.1,N=100)` that implements logistic regression and does `N` iterations. Initialize the weights randomly using `numpy.random.randn` and make them small by multiplying by .01. Return the weights and bias term.

Also make a function called `predict(X,W,b)` that makes predictions based on the weights and bias.

Problem 3 (1 pt) The file "hw_28-data_01.pickle" contains a pair $[X, y]$ and the file "hw_28-predict_01.pickle contains an $\hat{X}$.

1. Apply logistic regression.
2. Plot this data using two different colors for X based on their y value. Add the boundary line given by the weights and bias to you graph. Remember, we have $w_1x + w_2y + b = 0$. You can solve for y to find the boundary. You should use a subplot with 2 rows (you'll be adding another graph to it).
3. Use your model to make predictions based on $\hat{X}$.
4. Plot the $\hat{X}$ data using two different colors based on their $\hat{y}$ predictions. Does your data fit your predicted boundary?

Problem 4 (1 pt) The file "hw_28-data_02.pickle" contains a pair $[X, y]$ and the file "hw_28-predict_02.pickle contains an $\hat{X}$.

1. Apply logistic regression.
2. Plot this data using two different colors for X based on their y value. Add the boundary line given by the weights and bias to you graph. Remember, we have $w_1x + w_2y + b = 0$. You can solve for y to find the boundary. You should use a subplot with 2 rows (you'll be adding another graph to it).
3. Use your model to make predictions based on $\hat{X}$.
4. Plot the $\hat{X}$ data using two different colors based on their $\hat{y}$ predictions. Does your data fit your predicted boundary?

Problem 5 (1 pt) You should have noticed that the boundary for Problem 3 was pretty good, but Problem 4 was awful. Why did this happen? Further, why was the boundary for the predictions perfect?