Problem 1: 1D logistic regression, cross entropy, and gradient descent

Pertinent readings for Problem #1:

Geron: Hands-on machine learning with Scikit Learn, Keras, and TensorFlow (2nd ed)

Blackboard location: /Resources / Our main textbooks for this class

Ch. 2: Pages 206 - 210 (logistic regression + the cross-entropy cost)

Your homework tasks: (turn in the parts highlighted in yellow

Simple 1D logistic data:

Suppose 20 students spent between 0 and 6 hours studying for an exam, and you wanted to know whether they had passed the exam or not as a function of study hours. This table comes straight out of the wiki page example !

https://en.wikipedia.org/wiki/Logistic_regression

Student ID	Hours	Passing the exam		Student ID	Hours	Passing the exam
1	0.50	0		11	2.75	1
2	0.75	0		12	3.00	0
3	1.00	0		13	3.25	1
4	1.25	0		14	3.50	0
5	1.50	0		15	4.00	1
6	1.75	0		16	4.25	1
7	1.75	1		17	4.50	1
8	2.00	0		18	4.75	1
9	2.25	1		19	5.00	1
10	2.50	0		20	5.50	1

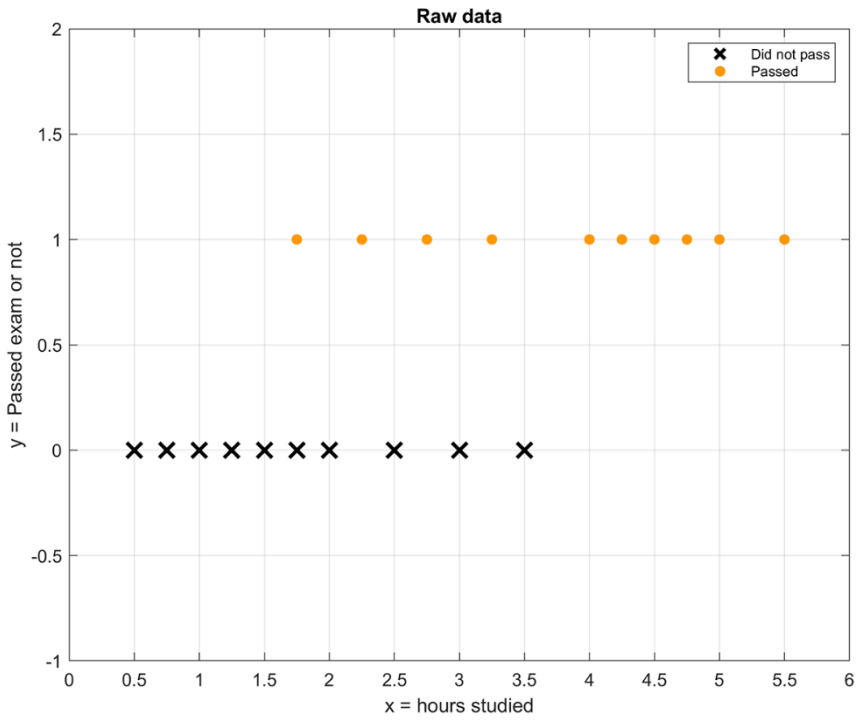


Figure 1: The raw data plot of the table above

Your main job for this problem is to verify the logistic fit from that wiki page using gradient descent ! =)
Recall the equation for a simple, "linear" logistic model, where:

$$f(x) = \frac{1}{1 + e^{-(w_0 + w_1 x)}} \quad , \quad where \begin{cases} w_0 = the\ intercept\ term \\ w_1 = the\ slope\ term \end{cases}$$

To find the best logistic fit, you have to find the optimum ${w_0}^*$ and ${w_1}^*$ that minimizes the ***cross-entropy cost function*** J for $N = 20$ student:

$$J = -\frac{1}{N} \sum_{i=1}^{N} \left[\; y_i \; ln(\, f(x_i) \,) + (1 - y_i) \; ln(\, 1 - f(x_i) \,) \; \right]$$

Where:

$$y_i = \begin{matrix} The\ class\ label\ of \\ each\ data\ point \end{matrix} = \begin{cases} 0 \quad (did\ not\ pass\ the\ exam) \\ 1 \quad (passed\ the\ exam\ !) \end{cases}$$

Now, from the wiki page, they claim the optimal fit that minimizes the is:

$$Optimum\ fit: \quad w^* = \begin{bmatrix} {w_0}^* \\ {w_1}^* \end{bmatrix} = \begin{bmatrix} -4.077 \\ 1.5046 \end{bmatrix} \rightarrow \begin{matrix} You\ will\ prove\ this \\ using\ gradient\ descent\ ! \end{matrix}$$

Probability of passing an exam versus hours of study [edit]

To answer the following question:

> A group of 20 students spends between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability of the student passing the exam?

The reason for using logistic regression for this problem is that the values of the dependent variable, pass and fail, while represented by "1" and "0", are not cardinal numbers. If the problem was changed so that pass/fail was replaced with the grade 0–100 (cardinal numbers), then simple regression analysis could be used.

The table shows the number of hours each student spent studying, and whether they passed (1) or failed (0).

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

The graph shows the probability of passing the exam versus the number of hours studying, with the logistic regression curve fitted to the data.

The logistic regression analysis gives the following output.

	Coefficient	Std.Error	z-value	P-value (Wald)
Intercept	−4.0777	1.7610	−2.316	0.0206
Hours	1.5046	0.6287	2.393	0.0167

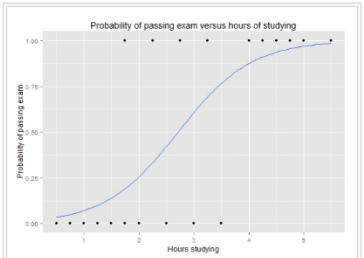
The output indicates that hours studying is significantly associated with the probability of passing the exam ($p = 0.0167$, Wald test). The output also provides the coefficients for $\text{Intercept} = -4.0777$ and $\text{Hours} = 1.5046$. These coefficients are entered in the logistic regression equation to estimate the odds (probability) of passing the exam:

$$\text{Log-odds of passing exam} = 1.5046 \cdot \text{Hours} - 4.0777 = 1.5046 \cdot (\text{Hours} - 2.71)$$
$$\text{Odds of passing exam} = \exp(1.5046 \cdot \text{Hours} - 4.0777) = \exp(1.5046 \cdot (\text{Hours} - 2.71))$$
$$\text{Probability of passing exam} = \frac{1}{1 + \exp(-(1.5046 \cdot \text{Hours} - 4.0777))}$$

One additional hour of study is estimated to increase log-odds of passing by 1.5046, so multiplying odds of passing by $\exp(1.5046) \approx 4.5$. The form with the *x*-intercept (2.71) shows that this estimates even odds (log-odds 0, odds 1, probability 1/2) for a student who studies 2.71 hours.

For example, for a student who studies 2 hours, entering the value $\text{Hours} = 2$ in the equation gives the estimated probability of passing the exam of 0.26:

$$\text{Probability of passing exam} = \frac{1}{1 + \exp(-(1.5046 \cdot 2 - 4.0777))} = 0.26$$



Graph of a logistic regression curve showing probability of passing an exam versus hours studying

Part 1: Plotting the contour map of the cross-entropy cost function J

You know your job here is to find the optimum parameters $w_0{}^*$ and $w_1{}^*$ that minimizes the cost function J. This automatically means that J is a function of w_0 and w_1 :

$$J(w_0,\ w_1) = -\frac{1}{N}\sum_{i=1}^{N}\ [\ y_i\ ln(\ f(x_i)\ )\ +\ (1-y_i)\ ln(\ 1-f(x_i)\ )\ ]$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\ \left[\ y_i\ ln\left(\ \frac{1}{1+\ e^{-(w_0\ +\ w_1 x_i)}}\right) +\ (1-y_i)\ ln\left(1-\frac{1}{1+\ e^{-(w_0\ +\ w_1 x_i)}}\right)\ \right]$$

And if you expanded this out for all 20 data points, you would have:

$$J(w_0,\ w_1) = -\frac{1}{20}\ \left[\ 0\cdot\ ln\left(\ \frac{1}{1+\ e^{-(w_0\ +\ w_1\ (0.5))}}\right) +\ (1-0)\ ln\left(1-\frac{1}{1+\ e^{-(w_0\ +\ w_1(0.5))}}\right)\ \right]\quad Data\ point\ \#1$$

$$-\frac{1}{20}\ \left[\ 0\cdot\ ln\left(\ \frac{1}{1+\ e^{-(w_0\ +\ w_1\ (0.75))}}\right) +\ (1-0)\ ln\left(1-\frac{1}{1+\ e^{-(w_0\ +\ w_1(0.75))}}\right)\ \right]\quad Data\ point\ \#2$$

$$= \ \ldots$$

$$-\frac{1}{20}\ \left[\ 1\cdot\ ln\left(\ \frac{1}{1+\ e^{-(w_0\ +\ w_1\ (5.5))}}\right) +\ (1-1)\ ln\left(1-\frac{1}{1+\ e^{-(w_0\ +\ w_1(5.5))}}\right)\ \right]\quad Data\ point\ \#20$$

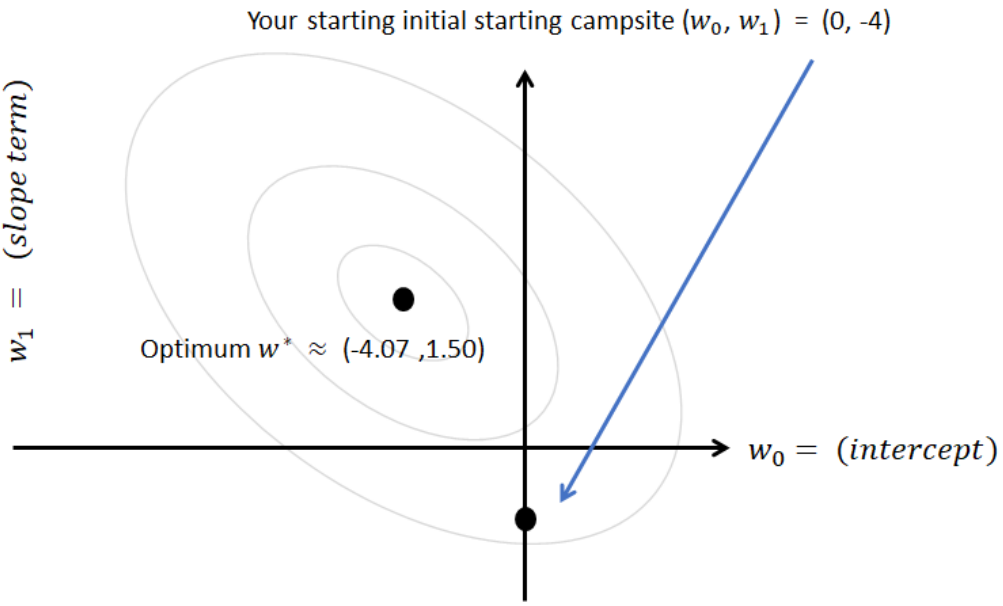
Your tasks for Part 1:

1) Using matlab's *surf* command, plot the 3D surface of J

2) Uisng matlab's *meshgrid* and *contour* commands, plot a contour map of J as a function of w_0 and w_1 on a separate figure. Also:

a) Set your axis limits as: axis([xmin xmax ymin ymax]) = axis([-10 10 -10 10])
b) Set the *axis square* command to set your grid scaling as squares.

Note: You will add more stuff on your contour plot !!! Your initial contour plot may look something like Figure 2:



Your plot should contain 8 – 10 contours that emphasize the size, tilt, and the minimum location of the 3D surface

Figure 2: The contour plot setup for your gradient descent run !! =)

Part 2: Perform gradient descent on the cross-entropy cost function J

Given that your cross-entropy cost function is:

$$J(w_0,\ w_1) = -\frac{1}{N}\sum_{i=1}^{N}\ [\ y_i\ ln(\ f(x_i)\ )\ +\ (1-y_i\ )\ ln(\ 1-f(x_i)\ )\ ]$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\ \left[\ y_i\ ln\left(\ \frac{1}{1+\ e^{-(w_0\ +\ w_1 x_i)}}\right) +\ (1-y_i\ )\ ln\left(1-\frac{1}{1+\ e^{-(w_0\ +\ w_1 x_i)}}\right)\ \right]$$

1) On a piece of paper, write down the expression for the gradient of J , where:

$$\nabla J\ =\ \begin{bmatrix}\frac{\partial J}{\partial w_0}\\ \\ \frac{\partial J}{\partial w_1}\end{bmatrix} = \begin{bmatrix}Some\ calculus\ expression\ with\ summation\ signs\ !\\ \\ Some\ calculus\ expression\ with\ summation\ signs\ !\end{bmatrix}$$

Hints:

1) You can consult Geron, Ch. 2 for the gradient of the cross-entropy function (read the pages at the 1st page of this PDF !!)

*2) *** IMORTANT ***: Regarding the expression for ∇J in Geron's book:*

$\frac{\partial J}{\partial w_0}$ = *You CANNOT trust his formula in Ch. 2 for this term … you have to derive the correct one yourself … = (*

$\frac{\partial J}{\partial w_1}$ = *You can trust his formula in Ch. 2 for this term*

Now, recall the general formula for gradient descent:

$$w^{k+1} = w^{k} - \alpha \left. \nabla J \right|_{at\ campsite\ w^k}$$

$$\left(\begin{matrix} new \\ campsite \end{matrix}\right) \quad \left(\begin{matrix} old \\ campsite \end{matrix}\right) \quad \left(\begin{matrix} learning \\ rate \end{matrix}\right) \quad \left(\begin{matrix} downhill \\ gradient \end{matrix}\right)$$

2) Using the following parameters:

Initial shitty guess	Constant learning rate	# of downhill hikes
$w^0 = \begin{bmatrix} {w_0}^0 \\ {w_1}^0 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$	$\alpha = 2$	20 iterations

Code your gradient descent run and overlay your results on your existing contour plot !! As you do this, please mark the following items on your plot:

a) Every iterated solution w^k (All of your "campsites" as you hike downhill; black dots in Fig 3)
b) The downhill gradient path that connects each campsite (the gray lines in Figure 3).

Hint: Remember: Since you forced the plot grid lines to be "square," each of your gray downhill gradient paths should be orthogonal to your contour lines !! If they are not orthogonal, there is either something wrong with your calculus answer for ∇J, or you have some typos in your code =(

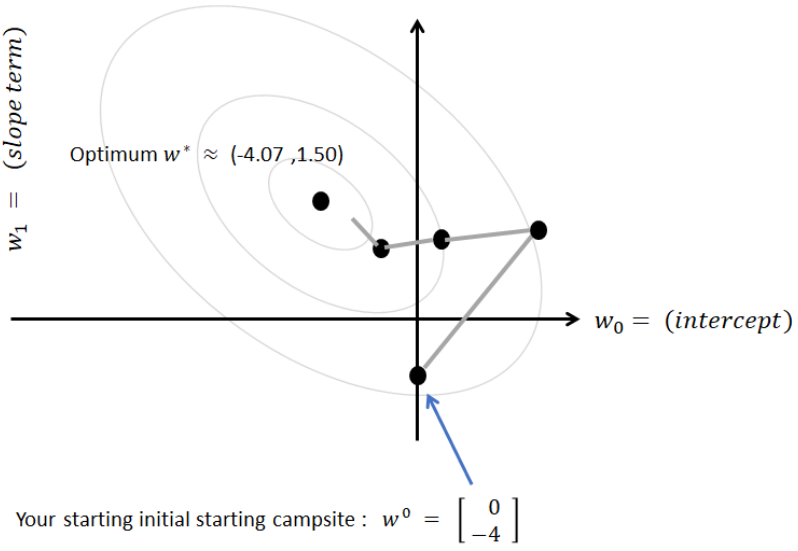


Figure 3: Your gradient descent contour plot may look something like this !

3) Now…. your downhill hikes may look really crappy !! =(This is because:

a) Your learning rate α is too large, and moreover

b) Your learning rate is a constant value =(Theoretically, it would be nice if we can dynamically change the learning rate as you hike downhill, right ?

** By the way: If you took BE 604 with me last semester, this is what Levenberg-Marquardt buys you: As you hike downhill, you can dynamically change your learning rate by making it smaller (more "conservative" hikes) or larger (more "adventurous" hikes) !

Ok ! now, on a separate plot: Change the learning rate such that you'll reach the optimum w^* value in about 100 iterations !

Initial shitty guess	Constant learning rate	# of downhill hikes
$w^0 = \begin{bmatrix} {w_0}^0 \\ {w_1}^0 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$	$\alpha = You\ decide$!	100 iterations

Code your gradient descent run and overlay your results on your new contour plot. As usual, please mark:

a) Every iterated solution w^k (All of your "campsites" as you hike downhill; black dots in Fig 3)
b) The downhill gradient path that connects each campsite (the gray lines in Figure 3).

Epilogue:

This exercise is intended to give you more practice on thinking about the numerical computation issues involving gradient descent problems. This will become important when we start our neural networks lectures in a week or so ! =)