CS2810 Mathematics of Data Models	Gold
Spring 2019	March 11, 2019

Homework 06

Due: March 20, 9PM
Point total: 60

Instructions:

- Submit your PDF and/or .py file to Blackboard by the due date and time. Please do not zip your files together, as this interferes with Blackboard's preview functionality. Always show all your work, and for full credit, you must use the method that the problem instructs you to use (unless none is mentioned). Handwritten or typeset solutions are both acceptable, but unreadable submissions will be penalized. You may discuss problems with other students, but you may not write up solutions together, copy solutions from a common whiteboard, or otherwise share your written work or code. Do not use code or language that is copied from the Internet or other students; attribute the ideas *and* rephrase in your own words.

Problem 1 (9 points)

Find the gradient of each function f at the specified point P, then indicate a vector of movement we should follow to maximally *decrease* f (for gradient descent, for example).

i. $f(x, y) = x^2 + 2xy + y^3$; $P = (1, 1)$

ii. $f(x, y) = 0$ if $x \leq 0$ or $y \leq 0$, else $f(x, y) = xy$; $P = (2, 3)$

iii. $f(w, x, y, z) = 2w + 3x + 4y + 5z$, $P = (6, 7, 8, 9)$ (The gradient works analogously for more dimensions than two.)

Problem 2 (15 points, 5 each)

For each function, find the partial derivatives according to x and y, and then find the location(s) where its gradient is zero, and thus, we potentially have a local minimum or maximum. Then find f_{xx} and f_{yy} and determine whether the curvature in each direction is positive (curving up), negative (curving down), or neither.

i. $f(x, y) = 3x^2 + 10y^2 + 3$

ii. $f(x, y) = -3x^2y - 3x + 27y + 4$

iii. $f(x, y) = -y^2/(x + 1)^4$ (Recall that this could be interpreted as $-y^2(x + 1)^{-4}$.)

Problem 3 (15 points [4, 4, 3, 4])

In each case, find the tangent plane and a normal vector of $z = f(x, y)$ at the point $P = (x_0, y_0, z_0)$. Give your tangent plane in the form $c_1x+c_2y+c_3z = c_4$; for example, $-x+2y+3z = 10$. (You don't need to normalize the normal vector.)

i. $z = x^2 + y^2$, $P = (3, 4, 5)$

ii. $z = \sin x + 2\sin y$; $P = (\pi, \pi, 0)$ (Recall if $f(x) = \sin x$, $f'(x) = \cos x$.)

iii. $z = 0, P = (1, 2, 0)$

iv. Normal vectors are often used in graphics to determine how bright a surface should be as a result of illumination. Lambert's Cosine Law says that the light reflected from a light-scattering surface has a brightness equal to $I\cos\theta$, where I is the original intensity of the light and θ is the angle between the normal vector at the point and a vector from the point to the light source. Find the brightness of illumination at the point $(3, 4, 5)$ (the same as the first part of this problem) if a light source is shining from (1,1,100) with intensity 100. (Flip the direction of the normal so that it's pointing toward the light source.)

Problem 4 (9 points, 3 each)

i. What is the directional derivative of a function in a direction orthogonal to the gradient? Explain why your claim must be true. (You can limit your proof to the case of the two-dimensional gradient.)

ii. Find a function where a finite global minimum exists, but gradient descent may not find it. Give both the function equation and a plot of the function. Your function need not use more than one variable.

iii. If we don't know a formula for a function that we want to perform gradient descent on, we could simply sample the formula in different directions, and estimate the gradient from that. Explain how we could estimate the gradient from the values $f(x, y), f(x + \Delta x)$, and $f(x, y + \Delta y)$. Is it worse to have a Δx that is too big, or one that is too small?

Problem 5 (12 points [4, 2, 4, 2])

For this problem, you're going to gain some experience using the machine learning tools in `scikit-learn`, a popular Python package for ML. You can find starter code in `sgd.py`.

Install the `scikit-learn` module before performing these exercises.

i. Refer to the documentation for `sklearn.linear_model.SGDRegressor` to train a stochastic gradient descent model on 100 noisy datapoints created with `create_data()`. The parameters of the model should be: use squared error as loss; iterate for 10^6 iterations (set tol to "None" and ignore the warning), use L1 regularization, and set the regularization parameter alpha to 0.001. Use `plot_fits()` and `solve_least_squares()` to compare the fit at $y = 0, z = 0$ and at $y = 3, z = 3$ to a least squares fit, and include the latter plot in your PDF submission.

ii. Why do you think SGD captures the true function better than least squares, which we thought was optimal? (Hint: Notice that small nonzero values for y and z coefficients will have a big effect at $y = 3, z = 3$, which are larger than any actually observed inputs. How did SGD avoid this effect?)

iii. Now train a neural network on the same data using `sklearn.neural_network.MLPRegressor` using the following parameters: 50 hidden units, L2 regularization with $\alpha = 0.001$, a million iterations, stochastic gradient descent, learning rate 0.01, tolerance 0.001, and allow 1000 iterations without improving more than the tolerance. Plot the performance versus your other two models for $y = 0, z = 0$, and include this plot in your PDF submission.

iv. Suggest one change to the neural network that might help it match the underlying function better. (You can suggest a direction and a parameter, without needing an exact value to set it to; for example, "increase the number of hidden units." There is more than one possible answer.)