Lab 07 – Analytical and Numerical Differentiation

CS3400 Machine Learning

Learning Outcomes

- Familiarize yourself with the difference between analytical and numerical methods for finding a function's first derivative
- Develop the skill of numeric differentiation in discrete space
- Develop intuition for the first derivative's shape and features
- Use the first derivative of a function to solve for the function's optima

Overview

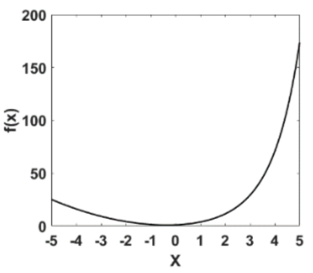
The previous lab focused on exploring cost functions and plotting the *error* over a given parameter space. We decided that the "solution" for our parameters is a set of parameters that produce the lowest error between our model and the data that we are given. While we used grid search in the previous lab to find these solutions, it should be obvious that grid search itself is a very costly approach. Because of this, we need to find other approaches that make more efficient use of our resources – enter the derivative.

The derivative is a tool that shows how a function is changing. Instead of searching all points over a grid, it would be much more helpful if we had a guide. A guide could tell us "Hey listen! I think the value (error) of our function is decreasing if we head over that way!" The derivative is our guide.

After completing the required calculus classes, you hopefully have a general knowledge of the methods and formula for integrating functions and solving for their derivatives. This course (and specifically the next few labs) will have you applying this knowledge to relevant problems and have you using those tools to do some meaningful work. While your classes probably covered *analytical* solutions or finding a solution for the derivative that takes an exact form, we can't always rely on having an analytical solution to the functions that we are investigating. This is one reason that we might want to rely on *numerical* methods for solving the derivatives of a function.

Let's use a trivial function as an example. If we have the following function,

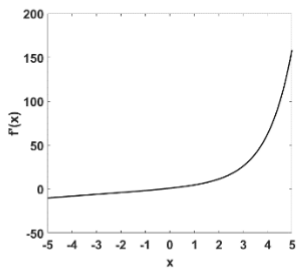
$$f(x) = x^2 + e^x$$



we can appreciate that this function is *dependent* on x. To explore this function, we can control x and see how the function responds. In this case we can sweep x over the range 5 ≥ x ≥ -5.

We can then use our calculus toolbox to solve for exact form of the first derivative and subsequently plot the derivative using the same range of x values while solving for $f'(x)$.

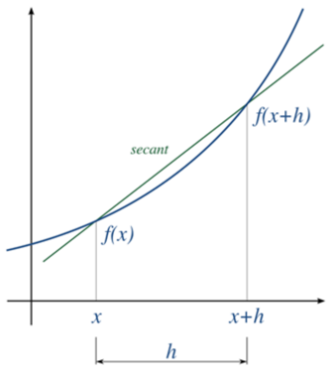
$$f'(x) = 2x + e^x$$



This lab will explore some interesting properties of the first derivative that we can leverage for our own goals.

While these are analytical solutions, we want to build out methods for solving these problems *numerically*. You can use the following formula to find an approximation of the first derivative without having to find an analytical solution (like we did above). All you need to this are values of the function (f(x) and f(x+h)) at two points (x and x+h). We can approximate the first derivative with a numerical solution using the following formula:

$$f'(x) \approx \frac{f(x+\mathrm{h}) - f(x)}{\mathrm{h}}$$



Here you should appreciate that f(x) refers to a function that is *dependent* on the x variable. The h (sometimes also represented as Δ) is a constant that we can specify that relates to our resolution, or *step size*. Take a moment to build a little intuition about what this approximation is doing. Specifically consider what the numerator and denominator each represent. Hint: What does a derivative evaluated at a single point tell you about the function?

Instructions

This lab begins with two experiments in your jupyter notebook to help build some intuition and experience with derivatives. In experiment 3, you will be packaging what you learned into a new python class and then revisiting the gaussian distribution. This means you will want your cost function and gaussdist.csv dataset from the previous lab.

Jupyter Notebook – Running experiments

Create a Jupyter notebook named <lastname>_lab07. The notebook should have a title, your name, and an introduction. For you introduction, make sure to include information about analytical and numerical derivatives. You will be running the following experiments in the notebook.

Each of the following steps should be performed in individual cells in your jupyter notebook.

Experiment 1: Analytical vs. Numerical Differentiation

In experiment 1 you will be comparing analytical solutions to a numerical approximation. The introduction gave you a numerical method for approximating the derivative. You will implement this method on the given function and investigate the role that constant *h* plays in the approximation.

Cubic Model:

$$f(x) = x^3 - 3x^2 - 144x + 432$$

1. Use a markdown cell in your notebook and solve (analytically) for the exact form of the derivative - $f'(x)$.
2. In the notebook, plot the function, $f(x)$, within the bounds of $-12 \le x \le 15$ with a step size of 0.01. In the same plot, graph $f'(x)$ over the same range using the analytical solution that you found. Observe the shape and values of each function.
3. Using the numerical method given above you are now going to estimate the derivative of this function. Make a plot for each of the *h* values listed below. For each plot, graph a line plot of analytical derivative solved in 2. Then in each figure, overlay the numerical approximation of the derivative using a scatter plot.

   - *h* = 0.01
   - *h* = 0.1
   - *h* = 1
   - *h* = 2

   To best visualize this, slice the approximated derivative vectors when making the scatter plot.

Experiment 2: 1-Dimensional Grid Search – Derivative Space

In experiment 1 you found two different ways to solve for the derivative of the given function. In this experiment you will put these solved derivatives to use. Just like in Lab 6, we can perform a grid search to find extrema of the function. However, in this lab we will perform the grid search on the *derivative* **NOT** the *cost function*. Because of this, we aren't looking for an extrema in the derivative space – we are looking for zeros. Take a moment to consider why we are searching for zeros in the derivative space.

1. Using your analytical solution for the derivative, solve by hand for values of x where the derivative is equal to zero. Show your work in the markdown cell.
2. Using the x-values that you found, create a new figure that has a line plot of the original function and overlay the individual points of (found_x , f(found_x)).
3. Copy the vectors that you used to store the derivative's numerical approximations from step 3 of experiment 1.
4. By using slicing techniques, search through each of numerically approximated derivative vectors to find the 2 locations (x values) that are closest to f'(x) = 0 (derivative equal to zero).
5. With these found x values, plot the original function using a line plot and place a dot over each of the found f(x) values.

Experiment 3: Gaussian Model Grid Search – Derivative Space

You have now seen how to estimate the numerical derivative of a 1-dimensional function – f(x). You may at this point be asking, "Calculating the numerical approximation is more steps. Why wouldn't I just use the analytical solution for every function that I come across!?". I would counter with "That's a great question! Now find the analytical solution for the first derivative of the gaussian cost function that you created in lab last week!"

While some of you may be so inclined to do that, you now have the tools to avoid that arduous process. It is a much simpler task to approximate this derivative using numerical techniques, and in fact this is now you task. Enter gradients and partial derivatives.

Gradients describe the change of a function (the derivative) in multiple dimensions. They are commonly represented using the nabla symbol and require the use of partial derivatives:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix}$$

While that may look like a mess, there is some good news! We have all of the tools to approximate partial derivatives with our numerical methods. Following the above 2-dimensional gradient example, our numerical techniques expand to:

$$\frac{\delta f}{\delta x} \approx \frac{f(x+\mathrm{h},\mathrm{y}) - f(x,y)}{\mathrm{h}} \qquad \frac{\delta f}{\delta y} \approx \frac{f(x,y+\mathrm{h}) - f(x,y)}{\mathrm{h}}$$

If you then calculate this gradient over the two dimensions ($\frac{\delta f}{\delta x}$ and $\frac{\delta f}{\delta y}$) you can perform a grid search just like you did in experiment 2. If you locate the zero crossings in derivative space, you have found the extrema in the function space.

1. Import the cost function class that you wrote last week and load in the data from the gaussdist.csv file.
2. Write a gradient class that has a single method named gradient. You will implement a gradient solver with this method. You have been provided with a numerical_differentiation_stub.py file and a corresponding unit test file. The following are a few points to consider:
    - This class is just consolidating (and expanding to multiple dimensions) what you have done in experiments 1 and 2.
    - Think carefully about what f(x,y) is. This is the evaluation of your function, and for the case of our gaussian model – the *cost* function.
    - This method should return a vector. Give consideration for what each of the elements in the vector correspond to.
3. Perform a grid search on the model parameters like you did in lab06 experiment 2 ($6 \geq \mu \geq 5$, $1.75 \geq \sigma \geq 1$), but this time you are searching over the *derivatives* (gradient) not the *error*. At the end of this step you should have 2 2-dimensional arrays, one for each of the model parameters. You should use a *h*/delta value of 1e-5.
4. Generate heatmaps for each of the two derivative arrays you calculated.
5. Find the locations in the derivative arrays that equal zero. Hint: You may have to search for the values that are closest to zero. Double Hint: If you use argsort on a 2-dimensional array, you can use array.flat() with the resulting index.
6. Finally, use the cost_function.predict() method with the found parameter values and plot the dataset and model output.

Questions:

After you run all the experiments create a markdown cell to answer questions in. Copy and paste each question into the cell prior to answering it. The following questions will be graded:

1. For experiment 1, how did your approximated derivatives (numerical solutions) compare to the analytical solution. Describe what effect the *h* term had on the accuracy of the approximation. Discuss considerations/tradeoffs when picking an optimal *h* value.
2. In experiment 2 you should have found both local extrema of the function. Describe what information the derivative provides and how you found these extrema. If each numerical approximation of the derivative didn't always lead to both extrema of the function, explain why.
3. Be creative - brainstorm a way that you could leverage the information of the derivative to only find the minima, not all extrema.
4. For experiment 3, you performed another grid search. Compare and contrast what features you were looking for in the error-space grid search (lab 06) and the derivative-space grid search. Make sure to compare the computational complexity between these spaces. Depending on your answer do you think it makes more sense to perform a grid search in error space or derivative space?
5. Be even more creative – brainstorm a way (or algorithm) that you could use to leverage the gradient, without having to use a grid search method.

I will be will looking for the following:

- That all of the tests pass
- That you used Numpy functions as much as possible and avoided writing loops where possible
- You avoided unnecessary calculations and memory copies where reasonable
- An introduction (including your own statement of the problem/lab purpose) and a written summary of your results at the top of the notebook in Markdown. Make sure to put your name at the top of the notebook.
- Plots of your data look reasonable. Plots have labels for each axis.
- Reasonable answers to the questions.

Submission Instructions

Save the notebook as a pdf file named <lastname>_lab07.pdf. Put your pdf file and cost_functions.py file into a zip file. Upload the zip file to Canvas.

Rubric

Presentation	**10%**
Followed submission and formatting instructions	5%
Plots: axes are properly labeled, used correct axes for variables, points were colored as required, lines were coloring as required, used a legend, chose appropriate axes limits to make plot readable and do not cause misleading interpretations, etc.	5%
Experiment #1	**15%**
Analytical Solution Numerical approximations of the derivative (4 plots)	5% 10%
Experiment #2	**15%**
Evaluating the original function with the found extrema Plot of original function and found points	10% 5%
Experiment #3	**30%**
Gradient Class implementation Solving for the derivatives of the gaussian cost function Plotting the heatmaps and gaussian model solution	15% 10% 5%
Reflection Questions	**25%**
Problem 1 Problem 2 Problem 3 Problem 4 Problem 5	5% 5% 5% 5% 5%
Exceeds Expectations	**5%**