Suggestions for final project ICCS – WiSe 2024

- Spend some time analyzing your models and make sure you understand all parameters well. While searching through the literature for the models pay close attention to what different parameters represent. This is very important in inferring results later. Some of you did correct computation but did not infer results properly therefore lost points. For example, you are asked to compare all three of your models and pick the best one, which parameters will you pick for this comparison and why?

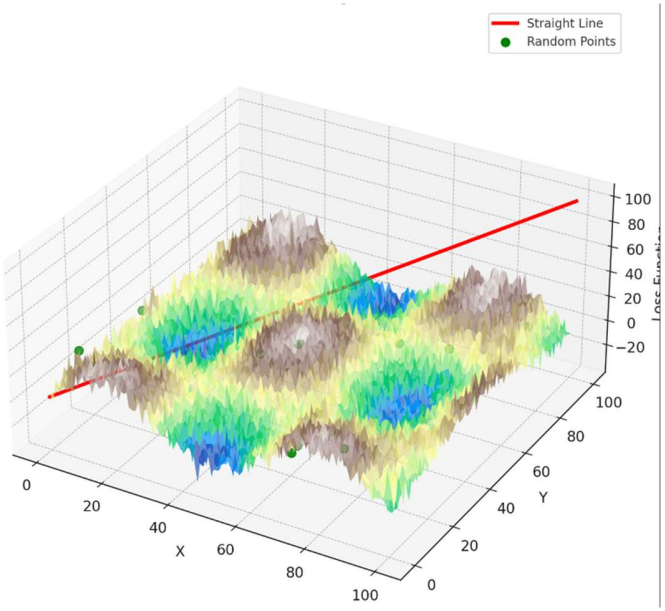
- Double check your log-likelihoods or other loss functions you use. If you calculate the loss function incorrectly your entire project will have dis-satisfactory or vague results.

- Go through the documentations of the inbuild functions you are using. For example if you are using '**scipy.optimize.minimize**' or '**scipy.stats.pearsonr**' make sure you understand the different arguments and what the functions is actually doing. You can also use LLM's for this since they are very quick at sorting through documentation and giving usage examples.

- Please write precise and legible comments in your code! The more comments you write the better. Reading someone's else's code is hard and grading it is time consuming; make sure you explain your programming correctly. Some of you do this well and some of you do not. If your code is perfect but illegible to understand because of lack of comments you might lose some points. Some of you have written comments in Chinese or German, unfortunately my knowledge of those languages is rather limited. The official language of this course is English please use that.

- Please follow the format given in the project description and make sure everything is visually in proportion. Sometimes I have to zoom in or zoom out 5-10x in order to fit your pdfs on my screen. Please be professional about this.

- When setting initial conditions for optimization over different points in the parameter space I suggest you use a random number generator. Some of you have tried to set initial conditions yourself but they end up being incorrect. For e.g.,
  initial = [(0,0),(5,5),(10,10),(50,50),(100,100)]
  are all points along the line x=y in the parameter space, these are not random points and you *will* miss some deeper minima of your loss function. This will lead to lower accuracy of your model.



- If your compute power is limited or you are running out of time you can reduce the initial number of points or use computationally inexpensive optimization methods. In this case, please mention specifically

in your submission that you lack compute power and print your CPU or GPU models within the code itself (must be executed on your machine)

[7]:
```
import platform
import psutil

def get_cpu_info():
    """Prints basic CPU information."""
    print("CPU:")
    print(platform.processor())  # Provides a generally informative string

if __name__ == "__main__":
    get_cpu_info()
```

```
CPU:
Intel64 Family 6 Model 183 Stepping 1, GenuineIntel
```

[8]:
```
import torch

if torch.cuda.is_available():
    print("CUDA is available!")
    print(f"Device name: {torch.cuda.get_device_name(0)}") #Prints name of the first CUDA device
else:
    print("CUDA is not available.")
```

```
CUDA is available!
Device name: NVIDIA GeForce RTX 4060 Laptop GPU
```

In this case make sure your model accuracy is at least 50% (the higher the better of course). Overall, don't worry about your compute power, you will be graded based on your effort and results not the power of your computer.

- For best results
  - Pick your parameter bounds very carefully
  - Pick multiple random initial conditions
  - Based on your model pick the appropriate loss function
  - Based on your model and available hardware choose the best optimization technique
  - More iteration/longer you can let your code run the better but don't overprioritize running time

- For doing test statistics for example p values you can use code- traditionally there are tables for this, they are well stored in most inbuilt libraries you need not calculate everything yourself.

- Take care of overflow and underflow errors. You can use logarithms and exponentials for this. DO NOT ignore overflow errors they will mess with your results and the overall accuracy of your model.

- Feel free to play around with different data and stats to make different types of Graphs/visualizations of your results. You have learned quite a bit of statistics in this course now is the time to apply them. The more graphs you make the easier it is for me to 'see' your results. I highly recommend box plots, heatmaps, histograms, contour plots or even basic graphs- you can use LLM's for this. In my experience they are good at plotting already systematic data. Always make co-relation scatter plots to visualize co-relation data this is a universal practice done professionally as well as in research settings.

Best of luck for your final project 👍

Valay Kumar Jain