Bivariate Colormaps

Abstract (Proposal On Github)

Matplotlib is a powerful visualization library widely used in the scientific community. I propose to develop a new feature for Matplotlib that allows users to create and use bivariate colormaps. Currently, Matplotlib provides a wide range of univariate colormaps to represent different types of data. However, there is no built-in support for bivariate colormaps, which are crucial for displaying complex data sets that require the representation of two variables simultaneously. To address this limitation, I propose to combine two univariate colormaps in various ways, such as averaging, blending, or creating a custom mapping function. This feature will enable Matplotlib users to create effective and visually appealing plots for complex data sets. I believe that the addition of bivariate colormaps to Matplotlib will be a significant step forward for data visualization and will benefit a broad range of scientific disciplines.

Technical Details

Project Description:

The objective of this project is to develop a new feature for Matplotlib that allows the creation of bivariate colormaps. Currently, Matplotlib only supports univariate colormaps, where a single value is used to determine the color of each pixel in an image. However, there are cases where it is useful to represent two variables in a single image, such as when visualizing complex data with both magnitude and phase information. The proposed feature would enable users to specify two input variables and generate a colormap that maps both variables to a color value.

The proposed approach for generating bivariate colormaps involves combining two univariate colormaps. One approach is to simply average the RGBA values from each univariate colormap to generate a new color for each pixel. Another approach is to use weighted average or more complex algorithms like using a radial basis function to map pairs of values to a single color. Furthermore, allowing the user to write their own custom function should also be possible to allow customization for a specific use-case. The resulting bivariate colormap can then be used to visualize data with two variables.

Note: The project approach could be altered based on mentor feedback if required.

Steps:

1. Normalization

Normalize input variables: Normalize each input variable in the range [0, 1]. This can be done using the normalization which works best with the dataset and the specific use-case. For example: min-max normalization, Z-score normalization. So, if we have 2 variables (N - variables can be chosen as this approach generalizes) v1, v2. We can obtain 2 new variables namely, norm_v1 and norm_v2.

The reason the variables are both individually normalized and then furthermore passed into a composite norm function is so that one variable doesn't dominate the colormap. for instance, if one variable has a range from 0 to 1 and another has a range from 0 to 100 the latter would dominate the color map if their norms are multiplied together. Therefore we first normalize the variables individually and then combine the normalized values using a composite norm function that takes into account the relative importance of each variable.

```
norm_v1 = normfunction1(v1)
norm_v2 = normfunction2(v2)
```

2. Apply individual color maps

Apply a separate color map to each normalized input variable to obtain two separate color maps (Using separate color maps helps reducing collisions in the map and has been discussed further later in the proposal). This can be achieved using functions like cm.get_cmap from matplotlibs code base.

```
    cmap1 = matplotlib.cm.get_cmap(map_name_1)
    cmap2 = matplotlib.cm.get_cmap(map_name_2)
    color1 = cmap1(nosrm_v1)
    color2 = cmap2(norm_v2)
```

3. Combine color maps using composite norm

This step involves creating a composite norm that can handle the normalization of all N variables, which can then be used to map the data onto a color space. To create the composite norm, we need to take into account the different normalization schemes that may be applied to each variable. We can achieve this by defining a normalization function for each variable, e.g., normfunction1(), normfunction2(), ..., normfunctionN(), and then combining them into a single composite norm.

One approach to combining the normalization functions is to use the product of the individual norms. For example, if we have two variables v1 and v2 with normalization functions normfunction1() and normfunction2(), respectively, the composite norm would be defined as:

```
composite_norm(v1, v2) = normfunction1(v1) * normfunction2(v2)
```

This means that the composite norm takes into account both variables and applies the appropriate normalization for each.

For N variables, we would extend this idea to:

```
composite_norm(v1, v2, ..., vn) = normfunction1(v1) * normfunction2(v2) * ... * normfunctionN(vn)
```

4. Apply alpha blending

Alpha blending is a technique used to combine colors with transparency. It involves specifying an alpha value (between 0 and 1) for each color, which represents its opacity. Colors with a higher alpha value are more opaque and less transparent, while colors with a lower alpha value are more transparent.

To blend two colors together using alpha blending, the colors are multiplied by their respective alpha values, added together, and then divided by the sum of the alpha values. This produces a new color that represents the combination of the two original colors with their respective opacities taken into account.

For n variables, we would first calculate the blended color for the first two variables using alpha blending, as described above. Then, we would calculate the blended color for the next variable and blend it with the previously calculated color using alpha blending. We would continue this process until we have blended all n colors together. The resulting color would represent the combination of all n colors with their respective opacities taken into account.

Collisions

Collisions occurr when 2 or more variable pairs are mapped to the same color. Collisions are minimized in this approach because of the following factors:

Normalization ensures that each variable is scaled to the same range, allowing for a more even distribution of data across the available color space. This can help reduce the likelihood of collisions occurring.

In addition, the use of alpha blending can help differentiate between data points that map to the same color value. By blending the colors of overlapping points, the resulting color will be a combination of the original colors, making it easier to distinguish between them.

However, if the number of overlapping points is too high, it may still be difficult to differentiate between them. In such cases, additional techniques such as interactive brushing and linking can be used to highlight individual data points or groups of data points.

Testing:

Unit tests will be written to test the algorithm for generating bivariate colormaps. These tests will ensure that the output of the algorithm is accurate and meets the expected results. Additional tests will also be written to ensure the new feature integrates properly with the rest of the Matplotlib library.

Schedule of Deliverables

List of Deliverables

1. Design and implement a set bivariate colormaps using the proposed methodology.
2. Write comprehensive documentation for the Python functionality and the methodology used to design the colormaps.
3. Write unit tests and integration tests to ensure the correctness and robustness of the package.
4. Integrate the functionality with matplotlibs code base while ensuring that good coding practices are followed.
5. Create a blog post for weekly updates on tasks performed during that specific week.

Community Bonding Period May 4th -> May 28th

During this period, I will engage with the community to get familiar with the project's codebase, discuss project requirements, and identify potential issues. I will ensure that my build environment is setup and that I am ready for development. Furthermore, I will create an RMD file seperately maintained as a blog. Regular posts would be made to this blog explaining the work that has been done.

Phase 1 May 29th -> July 10th

1. Research on different algorithms and techniques for generating bivariate colormaps.
2. Implementing a basic version of the bivariate colormap function in Matplotlib that combines two univariate colormaps.
3. Writing unit tests to ensure that the function works correctly and integrates with the rest of Matplotlib's codebase.
4. Submitting a progress report at the end of this phase.

Phase 2 July 14th -> August 21st

1. Improving and expanding the bivariate colormap function to include different algorithms for combining the two univariate colormaps.
2. Implementing support for N-Variate colormaps.
3. Adding support for different color spaces and data types.
4. Improving the performance and efficiency of the function.
5. Writing more unit tests and documentation.
6. Participating in code reviews and making necessary changes.
7. Submitting a progress report at the end of this phase.

Final Week August 21st -> August 28th

1. Finalizing the implementation of the bivariate/N-variate colormap function in Matplotlib.
2. Ensuring that the function works with different types of plots and use cases.
3. Updating the documentation and providing examples of usage.
4. Conducting further testing and fixing any bugs or issues that arise.

After GSoC

1. Continue Participating in code reviews and making changes as needed.
2. Fixing issues that I could feasibly tackle.
3. Creating new PRs for issues that I work on.

Development Experience

I am Arshdeep Singh, and I can be reached at +919815138659 / arsh.official2002@gmail.com / Linkedin / Github. I am currently a Third Year student pursuing a BE in Computer Science.

I have achieved several notable accomplishments in my field, which illustrate my development experience including winning Best Domain Project in MLH Hackhound Hackathon held at SRM University and being a finalist in Techgig Code Gladiator Codeathon 2022 among over 400,000 total participants. I have also acquired some certifications such as Data Analytics Specialization offered by Google and Neural Networks and Deep Learning offered by DeepLearning.ai.

In terms of experience, I was selected for Amazon Machine Learning Summer School, where I gained a deep understanding of techniques such as Supervised, Unsupervised, Sequential and Reinforcement Learning. I also worked with popular Python libraries such as scikit, pytorch, numpy, tensorflow, cv2, etc.

Currently, I am working as a Prism Research Intern @Samsung, where I am collaborating with a team of 4 members to successfully implement a Machine Learning Model for upscaling images by a factor of 4. I have also contributed to the development of the backend for a first-place winning project out of over 250 participating teams in the Hackhound MLH hackathon, where we utilized Node.js, ReactJS, CockroachDB, Twilio, and Cohere to develop a seamless web application for doctors, reservations, and AI-based food recommendations Project Link. One of my most successful projects is the Road Detection from Satellite Images, where I successfully developed a Generative Adversarial Network (GAN) model using U-Net architecture for generating road maps from satellite images using Pytorch. Project Link

Furthermore, most of my projects are published on github at https://github.com/arsh73552. I've also had some experience contributing to open-source in the past. I worked on an issue in scikit-image where incorrect output was generated for certain methods. I also helped in writing the unit tests to ensure that there wasn't a problem with the new functionality while fixing the bug.

Motivation Letter

Why this project?

The bivariate colormaps project is exciting to me because it has the potential to improve the visualization of complex data sets. As someone who has some experience in data science and visualizations, I understand the importance of effective and efficient data representation. Bivariate colormaps have the ability to represent two dimensions of data in a single color map, which can simplify the visualization process and allow for more accurate and intuitive interpretation of the data. Additionally, the project involves working with the matplotlib library, which is a widely used and respected tool for data visualization. Improving the functionality of such a widely used library would have a significant impact on the field of data science and the larger scientific community.

Furthermore, I've gone through the related issues (#4369 #8738 #14168 #18809), thoroughly understand the problem statement and feel like this is a problem statement that i could feasibly tackle.

Other commitments

If selected, I would be treating GSoC with the highest priority. I don't have any commitments for summer 2023 and would be able to devote a significant amount of time to completion of the project Bivariate Colormaps. I would basically be treating this as full-time job and ensuring that the project is integrated with matplotlibs codebase before final evaluation.