Homework 0 | Datasets

Corce: CS7616
Due: Wed, Jan 13, 11:55pm

Problem

---

1. Bid for the papers you'd like to present

- Follow this google forum to rate your selection of papers, further instuctions can be found there. If you'd like to know how it's done, you can checkout the code here yourself.
- We'll post the assignments after the due date. If your outcome was not locally optimal for you, you can ask someone to trade and have the two you both notify the instuctors on Piazza.

2. Find an available data set that corresponds to "modest" number of features and "small" number of classes

- **Modest** - plausible to try all or many possible subsets of features, 5 and 30 would be fine. It would be great if the dimension were low enough that you could think about learning a joint density from the training data.
- **Small** - maybe less than 5. 2 is ideal. 30 would be too many.

3. Label your assigned video segment.

- We've recorded, divided, uploaded, and segmented the new dataset for you. Now we need your help in labeling the videos using the web based video segmentation tool using the classes below. The TAs have sent each of you a link by email to one minute long clip that should take you about an hour to label in one sitting. If you did not receive a link from the TA, or have any questions in labeling, please contact us on Piazza.
- Remember to always click **save** before you finish and exit the session. Even if you get tiered, you can save your progress whenever and resume after a well deserved snack.

4. Setup your system.

- Whether you end up using Python or MATLAB, you should make sure your system is configured and is familiar to you so that any challenges you may encounter during your course work will at least not be technical or last minute.
- On the couse github repo, we've added created a small wiki that includes instalation help and helpful resources; Like how to edit the notebook you are currently reading to submit your assignments with inline figures, and documention and save to PDF.

Datasets

You can create your own that fits the criteria above or search online for publicly available datasets. You'll be working with this dataset frequently in your assignments, so make sure it's something that interests you and will prove to be of some sport or challenge to show off your learned pattern recognition skills.

Resources

Here are some additonal resources you can use to find an available dataset that interests you

- UCI Machine Learning Repository
- mldata.org

- WEKA Datasets
- DMOZ category entry
- Kaggle
- Quora Wiki

Submit

A brief half page description of the dataset you choose.

- Where it comes from and why did you choose it?
- What do the features and classes define?
- How could we would access it?
- What domain expertise would be relevant to know for the dataset?

Include a few simple plots of your dataset demonstrating you have a working setup and know how to interpret the dataset. This could be graphing a distribution of some features over classes, or plotting labels over some 2D space. A few examples: Iris, Digit, Scatterplot.

Subbit your PDF report, code, and test results in a zip file. You can use any programming language, but we will prefer `Python` or `Matlab`. The code should be easily runnable. Please include a README.md file that describes how the TAs are to run your code. The test results should be in a CSV file format that we will send an example of.

Suggestions

- To delete segments, like a wall segment, click the wall label, then shift and click to delete the segment. Make sure the correct label is selected otherwise the label won't delete.
- You can drag and paint to select multiple superpixels. Superpixels that are already selected can't be replaced. You have to deselect them and repaint them to change the label.
- Make sure you select different levels of the hierarchy to paint smaller and larger segments. The smaller the level of hierarchy, the smaller the superpixel and less it will track generally.
- If a segment overlaps a little it's fine. However if it overlaps a lot and it's the smallest hierarchy (ie the floor also selects a whole wall), then just leave it blank.
- You can hold shift and hover over a segment to see where it's superpixel extends to.
- **MAKE SURE YOU SAVE WHEN FINISHED!!!!!**

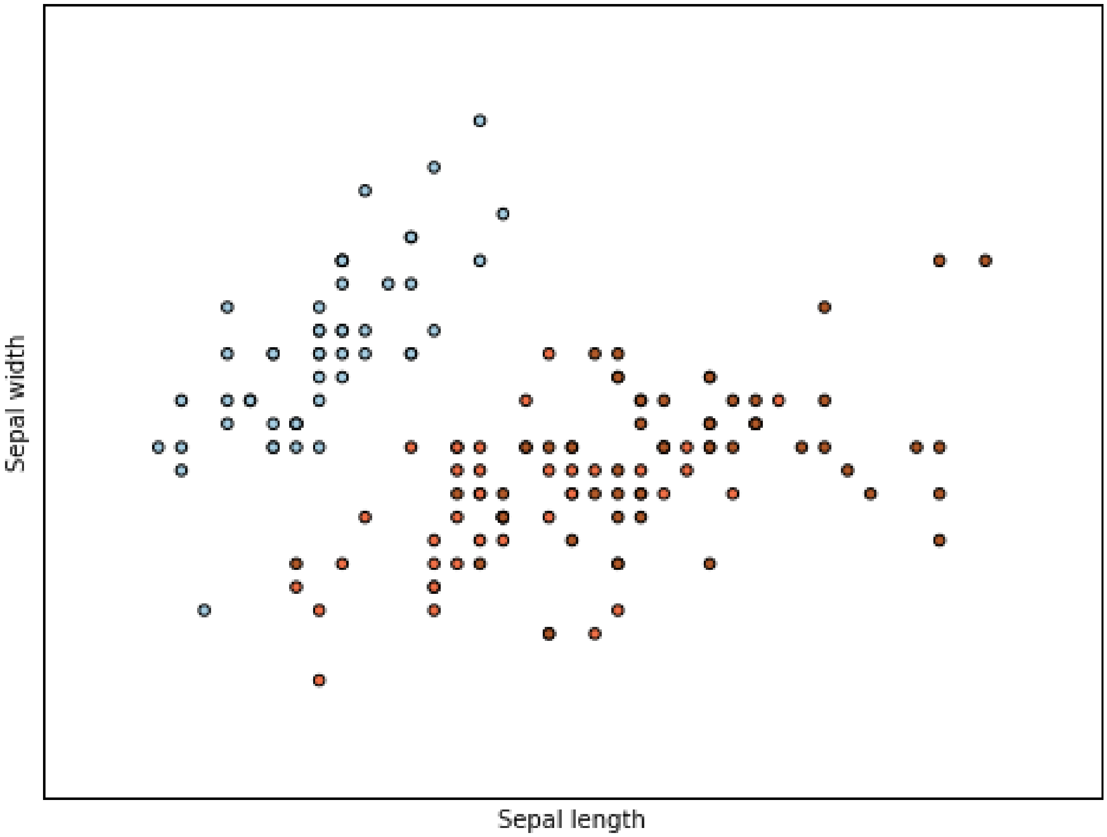
Labels

Leave out #comments in the label string

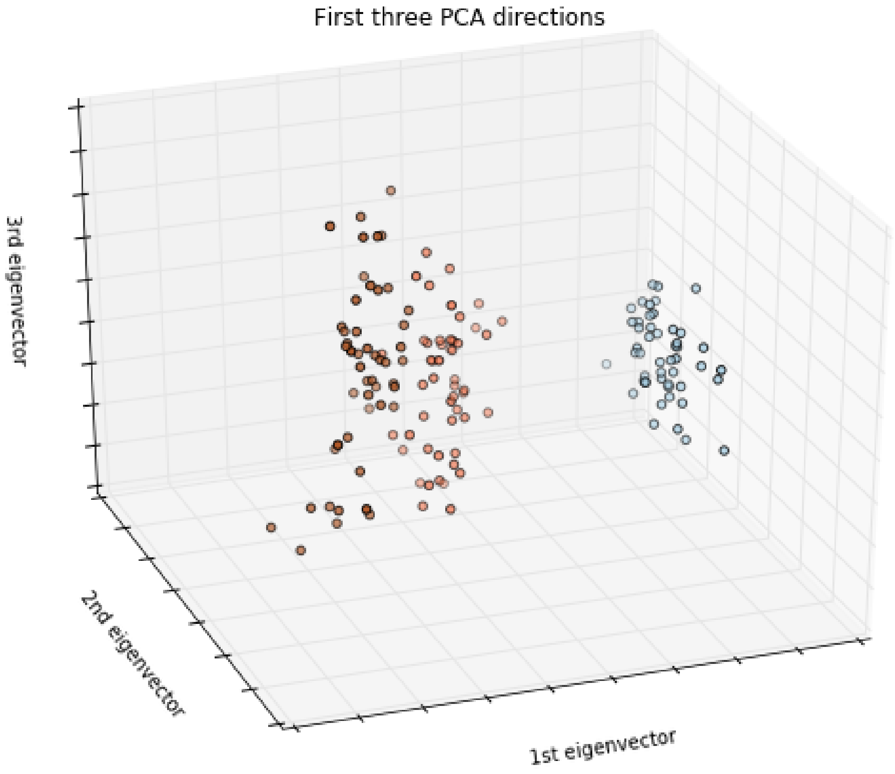
```
books
cabinets
ceiling
chair
computer
cup #(including bottles)
door
fire_extinguisher
floor
fridge
```

```
keyboard
monitor
person
poster
signs #(including exit/door signs)
table
trashcan #(including recycling)
walls
whiteboard
```

```
In [1]: # Code source: Gaël Varoquaux
        # Modified for documentation by Jaques Grobler
        # License: BSD 3 clause

        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        from sklearn import datasets
        from sklearn.decomposition import PCA
        %matplotlib inline

        # import some data to play with
        iris = datasets.load_iris()
        X = iris.data[:, :2]  # we only take the first two features.
        Y = iris.target

        x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
        y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5

        plt.figure(2, figsize=(8, 6))
        plt.clf()

        # Plot the training points
        plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)
        plt.xlabel('Sepal length')
        plt.ylabel('Sepal width')

        plt.xlim(x_min, x_max)
        plt.ylim(y_min, y_max)
        plt.xticks(())
        plt.yticks(())
        plt.show()
```



```
In [2]: # To getter a better understanding of interaction of the dimensions
        # plot the first three PCA dimensions
        fig = plt.figure(1, figsize=(8, 6))
        ax = Axes3D(fig, elev=-150, azim=110)
        X_reduced = PCA(n_components=3).fit_transform(iris.data)
        ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2], c=Y,
                   cmap=plt.cm.Paired)
        ax.set_title("First three PCA directions")
        ax.set_xlabel("1st eigenvector")
        ax.w_xaxis.set_ticklabels([])
        ax.set_ylabel("2nd eigenvector")
        ax.w_yaxis.set_ticklabels([])
        ax.set_zlabel("3rd eigenvector")
        ax.w_zaxis.set_ticklabels([])
        plt.show()
```


First three PCA directions
3rd eigenvector
2nd eigenvector
1st eigenvector