Pattern Recognition - Project 02
Removing Clouds from Sattelite Pictures

Tutor: Sebastian Mueller
Handed out: **Jan 11**
Discussion: **Feb 08**
Submission deadline: **Feb 04, 1000h**

Dataset

You are given the Dortmund From Space 2018 / GER dataset from [1] that contains three-channel (RGB) satellite images taken over a region around 51°33'21.5"N 7°32'14.3"E. All images show the same region at different points in time. For each image there is a mask that indicates whether a pixel shows an unobstructed view on the ground (white, ground-pixel) or a cloud (black) (see Figure 1). The dataset provided here is not complete, images showing clouds only were removed.

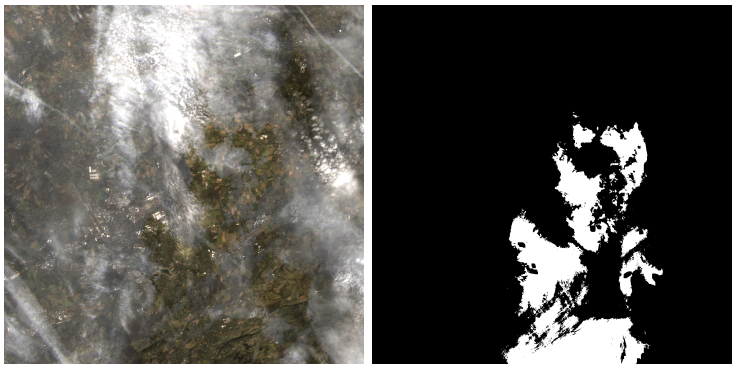

Figure 1: Sample image from the dataset (left) with its accompanying mask (right). Black pixels in the mask indicate a cloudy area in the image, white pixels indicate an unobstructed view.

Tasks

In this project you will train a classifier-based model that predicts the color of a ground-pixel given the colors of the adjacent (neighbor) pixels.

Colors are (more or less) continuous values, would a regression then not be a better fit for the problem? Think of the linear regression model from the last project. The input now are the color values from neighboring pixels. The linear regression would take those values and compute a weighted sum (or interpolation). This works great for homogeneous regions in the image, but at borders of eg. fields and roads, where the colors change abruptly, this approach breaks down. Using a classifier circumvents this problem because we are free to assign any

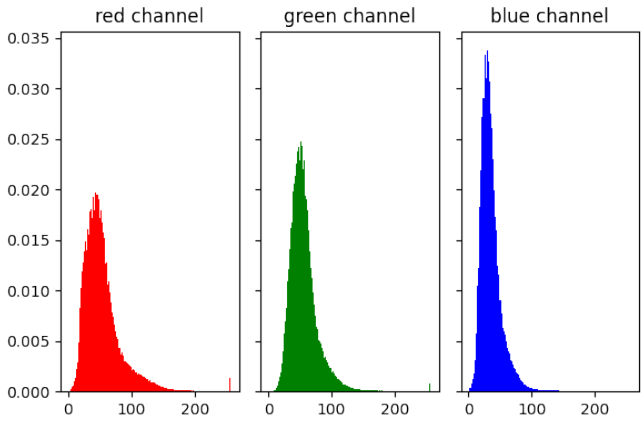

Figure 2: Histogram of the individual channels over all images from the dataset.

kind of label to a pattern. The drawback now is: We need to assign each pixel to a class to be able to train a classifier. So how do we define the classes?

1 - Color Discretization

Because in the end we want to predict a color, the color is what a class will have to represent. The images are given in RGB color space which encompasses 16777216 different colors. Analyzing the images we find that the images contain 294494 different color values for ground-pixels, so way less than the possible 16.7 million, but still too many to make each color its own class.

To reduce this number we need a way to map multiple colors to a single value. A naïve approach would be to just divide the observed color range into regular intervals and have every chunk be its own class. However, that would treat regions in the color space that appear often in the data exactly the same as regions that are less densely populated (see histogram over color channels in Figure 2).

A datadriven approach that takes the density into account and that you learned about in the lecture is k-means clustering. Using k-means also gives us direct control over the number of classes we divide our data into.

Task 1 Implement *both*, the **Lloyd** and **Hartigan** variants of k-means clustering. Compute clusterings for different k on the RGB data of ground-pixels. Compare the different clusterings

a) via the final quantization errors

b) visually, by replacing each pixel in an image with the coordinates of its associated prototype

Further, visualize how the prototypes are distributed in the data by reproducing Figure 2 (code provided) and marking the positions of the prototypes in the histograms.

Finally, to obtain the labels, assign a unique number to each prototype and for each pixel set the number of its associated prototype as its label.

Remark You may run into issues regarding memory and time complexity as computing k-means on a large dataset is quite expensive. How can you obtain a good clustering regardless?

2 - Classification

The input for the classifier is going to be the concatenated color values of the pixels in the neighborhood of the target pixel. The class label of the target pixel is the label obtained from the k-means clustering.

Task 2 Implement, train (remember cross-validation) and compare performance of multi-class variants of **LDA**, **primal SVM with L_1 loss** and **primal SVM with L_2 loss**. To avoid redundancy in your code, note that the training scheme for all models is the same. Compare the models in two ways:

a) classification performance

b) visually, by classifying all (available) pixels of an image and mapping them back to the colors associated with their class.

Remark These models have some pitfalls related to the dimensionality of the input:

- A base LDA/ linear SVM can only discriminate 2 classes. The multi-class extensions require the training of at least as many classifiers as there are classes.
- These classifiers require the dimensionality of the data to be higher than the number of classes. That means if you for example use a 4-neighborhood as an input, the input vector is only of dimensionality 12 and your model hence would be limited to only 11 classes. Increasing the neighborhood size is one way to increase the input dimensionality, but maybe you can find a more general approach.

Presenting you results

As with the last project you have the opportunity to present your work in the first tutorial after the deadline. Additionally, registered groups can again submit a written report in the form of a PDF or an executable Jupyter Notebook. The focus of the report should lie on written descriptions of what you did and what your results are, enriched with visualizations where beneficial. Code is only complementary material.

References

[1] R. Fischer et al. "No Cloud on the Horizon: Probabilistic Gap Filling in Satellite Image Series". In: *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. 2020, pp. 546–555. DOI: 10.1109/DSAA49011.2020.00069.