SGN-41006 Signal Interpretation

Exercise Set 3: January 27 - 28, 2016

Exercises consist of both pen&paper and computer assignments. Pen&paper questions are solved at home before exercises, while computer assignments are solved during exercise hours. The computer assignments are marked by text `python` and Pen&paper questions by text `pen&paper`

1. `pen&paper` *Design an optimal detector for step signal.*

   The lecture slides describe an optimal detector for a known waveform $s[n]$. Apply it to design the optimal detector for a step edge:

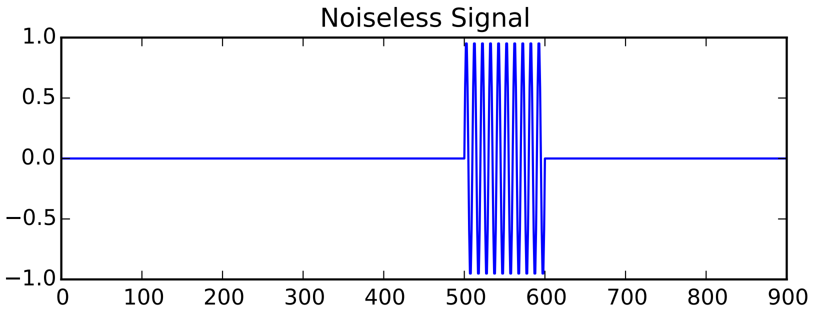
   $$s[n] = \begin{cases} -1, & \text{for } 0 \leq n < 10 \\ 1, & \text{for } 10 \leq n < 20 \end{cases}$$

   Simplify the expression as far as you can.

2. `python` *Implement a sinusoid detector.*

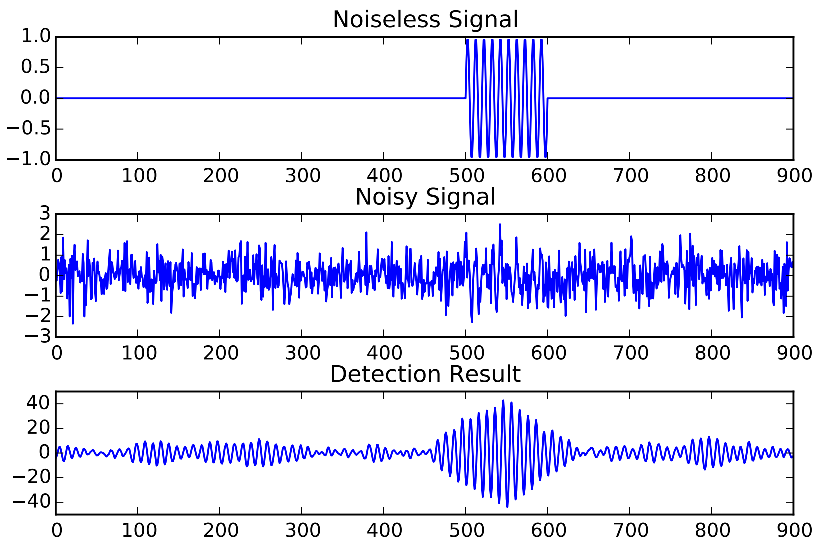
   In this exercise we generate a noisy sinusoid with known frequency and see how the sinusoid detector of the lecture slides performs.

   a) Create a vector of zero and sinusoidal components that looks like the plot below. Commands: `np.zeros`, `np.concatenate`. Sinusoid is generated by `np.cos(2 * np.pi * 0.1 * n)`.

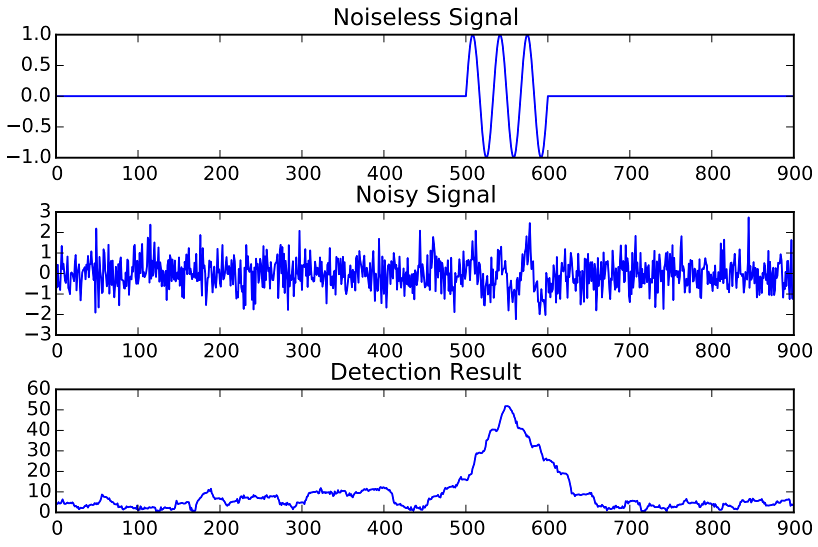
   b) Create a noisy version of the signal by adding Gaussian noise with variance 0.5: `y_n = y` np.sqrt(0.5) * np.random.randn(y.size)+.

   c) Implement the two detectors and reproduce the below plot.



3. `python` *Same as previous but different frequency and detector.*

   Change the code of the previous exercise such that the frequency is 0.03 and the detector is the random signal version.



4. `python` *Load a set of images and extract histogram features*

   Download the following file and extract the contents:

   ```
   http://www.cs.tut.fi/courses/SGN-41006/exercises/copper_images.zip
   ```

   The folder consists of five microscope images of copper (extracted from the course competition dataset). The task is to load them all into memory and collect the histogram from each. The histogram might be one feature for classifying the images (we will study more advanced features later).

a) Find a list of all images in the folder using `glob.glob` or `os.listdir`.

b) Initialize an empty feature matrix `X = []` and the class label vector `y = []`.

c) For each filename in the list:

- Load the image into numpy array:

```
>>> from matplotlib import pyplot as plt
>>> im = plt.imread(filename)
>>> im.shape
(512L, 512L, 3L)
```

- Compute the histogram using `numpy.histogram`.[1]
- Append the histogram to a feature matrix using `X.append()`.
- Class label is stored in the filename, *e.g.,* `0.jpg` belongs to class 0. Parse this from the name as `c = int(os.path.basename(filename).split(".")[0])` and append to `y`. Try to explain what the parsing line does.

d) Finally convert the lists `X` and `y` into a numpy arrays: `X = numpy.array(X)`. Check the array shapes.

5. **python** *Train a classifier using the 5 images.*

In this exercise we will train a Logistic Regression classifier with the array of histograms of exercise 4.

- Train a logistic regression classifier with `sklearn.linear_model.LogisticRegression()`.
- See what are the predicted class labels for the training data. Note, this is not evidence that the classifier really works, just a sanity check that the training was done properly.
- Use the `LogisticRegression().predict_proba()` method to predict class probabilities, as well.

[1]In fact, `numpy.histogram` computes the distribution from all RGB channels jointly. You would probably get a better separation by computing the histograms for each channel separately and concatenating into a 768-long vector:

```
histograms = [numpy.histogram(im[..., channel]) for channel in [0,1,2]]
h = np.concatenate(histograms)
```