Assignment 1
IFT 6085, Winter 2013

Date posted: February 5, 2013
Due: February 12, 2013, at the beginning of class

---

1. (3/20) In class we defined the discrete Fourier Transform (DFT) of a one-dimensional signal $s(t)$ as

$$S(k) = \sum_{t=0}^{T-1} s(t) e^{-i\frac{2\pi}{T}kt} \quad k = 0, \ldots, T-1$$

Show that for *real valued signals* $s(t)$ the following is true:

$$S(l) = \bar{S}(T-l)$$

where $\bar{c}$ is the complex conjugate of c.

Hint:

- You may use the following property of complex numbers c_t: $\sum_t \bar{c}_t = \overline{(\sum_t c_t)}$

2. (7/20) Download the images

`http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/nao_bw.jpg`

and

`http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/guido_bw.jpg`

and save them locally. Display the amplitude and phase spectrum of each image. Discuss briefly the differences in the spectra.

Then create two new images by combining the phase spectrum of one image with the amplitude spectrum of the other image (and vice versa), and transforming back using the inverse DFT. Discuss briefly what you see.

Hints:

- The amplitude spectrum should be clearly visible. Use a log transform if necessary.
- All spectra should be displayed with the **0**-frequency vector in the center.
- Display real-valued images only. If the inverse FFT returns complex numbers you will need to deal with this somehow.

- Use gray values not colors to display the spectra.
- For those using Python 2.x/pylab:
  - The complex number i is represented by the expression "1j" in python.
  - You can access the real and imaginary parts of a complex number c using "c.real" and "c.imag", respectively.
  - To display the spectra you can use the function "imshow".
  - You may want to make use the numpy functions "fft2", "ifft2", "fftshift", and "ifftshift".
  - To load an image from a file into a numpy array you can use "imread".
  - Be sure to represent your data as floats, otherwise strange things will happen.

What to hand in: (a) Images of the amplitude and phase spectra (4 images in total) and your short description. (b) The two combined images and your interpretation.

Do *not* hand in any program code.

3. (3/20) The inverse of the ZCA transform is also called the *square-root of the data covariance matrix*. Why?

4. (7/20) Download the image

   `http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/guido_bw.jpg`

   and save it locally (same image as in question 2).

   Cut out 1000 patches of size 8×8 pixels from random positions in the image. DC center and contrast normalize each patch. Display some of the resulting patches as gray value images.

   Compute the PCA whitening matrix and the ZCA whitening matrix of the (vectorized) patches.

   Display the 64 learned PCA und ZCA basis images as gray value images, sorted according to the corresponding eigenvalues.

   Hints:

   - Do not forget to mean center the images before computing the covariance matrix.
   - For those using Python 2.x/pylab:
     - You may make use of the function dispims at `http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/dispims.py` to display a set of filters.
     - Be sure to represent your data as floats, otherwise strange things will happen.

   What to hand in: Three images in total: One image showing example patches, one image showing the 64 PCA filters, one image showing the 64 ZCA filters.

   Do *not* hand in any program code.