**Mathematical methods for machine learning and signal processing** SS 19

# Problem set 5

*V. Morgenshtern*

*Due date: 23. July 2019*

## Problem 1: Nuclear norm ball in matrix completion

Let

$$\mathbf{A} = \begin{bmatrix} x & y \\ y & z \end{bmatrix}$$

be a real symmetric matrix.

Recall that the nuclear norm of $\mathbf{A}$ is defined as

$$\|\mathbf{A}\|_* = \operatorname{tr} \sqrt{\mathbf{A}^\mathsf{T}\mathbf{A}} = \sigma_1(\mathbf{A}) + \sigma_2(\mathbf{A})$$

where $\sigma_{1,2}(\mathbf{A})$ are the singular values of $\mathbf{A}$.

Plot the unit nuclear ball of real symmetric $2 \times 2$ matrices in 3D as a function of $x, y, z$.

*Hint:* Express the singular values of $\mathbf{A}$ through $x, y, z$. Make the necessary derivations and simplifications.

## Problem 2: Basic matrix completion experiment

This is a computer exercise.

Use `cvxpy` to perform a matrix completion experiment:

1. Generate a low rank matrix.
2. Erase some of its entries.
3. Solve the appropriate nuclear norm minimization problem to restore the missing entries.
4. Compare the reconstructed matrix with the original. Report the reconstruction error in a meaningful way.

## Problem 3: Approximate invariance of scattering transform to translations

The goal of this exercise is to obtain basic practical experience in working with Scattering Transform.

A modern software package for computing the Scattering Transform is `kymat.io`, it allows to compute scattering transforms of order 1 and 2 for 1D, 2D, and 3D signals.

You will work with MNIST dataset. An easy way to interact with the dataset and to work with tensors is to use `pytorch.org`

1. Install *kymatio*. Understand the basics of the package from the online documentation

2. Install *PyTorch*. Understand the basics of the package from the "Deep Learning with PyTorch: A 60 minute blitz.". *Hint: You don't need to go through the whole tutorial. You need to understand the basics on how to manipulate tensors and how to access standard databases such as MNIST.*

3. Take a $28 * 28$ image from MNIST dataset using `torchvision.datasets.MNIST`. Pad the PyTorch tensor to obtain an image of size $32 * 32$. Display the image.

4. Use `Scattering2D` function of kymatio to obtain the scattering transform of the image. Select $L = 6$ angle bins for the directional wavelets; take $J = 3$; and `max_order` equal to 1.

   - What is the shape of the output tensor? Indicate, what are the coordinate dimensions, what are the filter dimension.
   - Visualize the scattering transform by displaying 2D slices of the 3D tensor.

5. Compute the scattering transform again, this time with `max_order` equal to 2. Visualize the output tensor. What differences do you observe. Explain why.

6. Display second order scattering coefficients separately from the first order scattering coefficients. How the sizes of the first order coefficients compare to the second order coefficients?

7. Implement a function that can shift an image along an $x$ axis and along a $y$ axis, analogous to `np.roll` but operating on a PyTorch tensor.

8. Explore how much the scattering transform changes when you shift the image. Report the change rates for different values of $J = 2, 3, 4, 5$ as a function of the size of the shift in pixels.

**Problem 4: Learning is scattering domain via PCA**

The goal of this exercise is to reproduce the near-state-of-the-art results for MNIST problem reported in [1]. Your goal is to reproduce the 0.7 percent accuracy result reported in Table 4 in the paper by using PCA algorithm applied to scattering transform coefficients.

1. Follow the instructions in the previous problem to access the MNIST dataset.

2. Use scattering transform with $J = 3$ and order 2.

3. For each class, 0 to 9, fit the PCA model to the training data of that class. You will have 10 pca models at this point.

   *Hint:* You can implement PCA yourself via SVD or use a function from `sklearn`; if you implement PCA yourself, don't forget to standardize the data and the record the mean values for the data.

   *Hint:* Using 140 eigenvectors in PCA is reported as an optimal choice in [1].

4. Let $\mathbf{x}$ denote an image from the training set, $S(\mathbf{x})$ denote its scattering transform, $P_i(S(\mathbf{x}))$ denote the projection onto the PCA eigenspace of the $i$th class.

*Hint:* If you implement PCA yourself don't forget to subtract the mean of the data before projecting onto the PCA eigenspace. If you use `sklearn`, the projection is implemented via `pca.transform()` followed by `pca.inverse_transform()`.

5. The prediction is
$$\hat{j} = \arg\min_j \|S(\mathbf{x}) - P_i(S(\mathbf{x}))\|.$$

6. Compute the error rate of your predictions on the test set. If you implement everything correctly, you should see an error of around 0.7 percent, which is an exceptionally good result.

## References

[1] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1885, 2013.