Deep Learning and Inverse Problems
April 2025
Reinhard Heckel

Problem Set 1

Problem 1 (Denoising)**.** In the first chapter, we saw that signal models are central for solving inverse problems. Here, we consider a denoising problem and show that for a n-dimensional signal that lies in a k-dimensional subspace, we can remove a fraction of $\frac{n-k}{n}$ of additive Gaussian noise.

Consider a signal $\mathbf{x}^* \in \mathbb{R}^n$ that lies in a k-dimensional subspace, and suppose we are given a noisy measurement

$$\mathbf{y} = \mathbf{x}^* + \mathbf{z},$$

where $\mathbf{z}$ is zero-mean Gaussian noise with co-variance matrix $(\sigma^2/n)\mathbf{I}$. Let $\mathbf{U} \in \mathbb{R}^{n\times k}$ be an orthonormal basis of the signal subspace. We denoise the signal by projecting the measurement onto the subspace, i.e., we consider the estimate $\hat{\mathbf{x}} = \mathbf{U}\mathbf{U}^T\mathbf{y}$.

1. Show that

   $$\mathbb{E}\left[\|\hat{\mathbf{x}} - \mathbf{x}^*\|_2^2\right] = \sigma^2 \frac{k}{n},$$

   where expectation is over the random noise $\mathbf{z}$.

   Hint: Recall that if $\mathbf{V} \in \mathbb{R}^{n\times n}$ is a unitary matrix (i.e., a matrix with orthonormal columns) and $\mathbf{z}$ has iid, zero-mean Gaussian entries, then $\mathbf{V}\mathbf{z}$ has the same distribution as $\mathbf{z}$.

2. Does the estimator $\hat{\mathbf{x}} = \mathbf{U}\mathbf{U}^T\mathbf{y}$ remove more noise as the subspace dimension shrinks? And intuitively, do you think a better denoising algorithm exists?

3. Next, we study this denoising algorithm numerically (ideally with python in a jupyter notebook using the libary numpy; if you are not familiar with those, this exercise is a good exercise to familiarize yourself).

   Generate a random k-dimensional subspace in $\mathbb{R}^{1000}$, and generate 500 random points in that subspace. Next, denoise each of those data points with the estimator above, and plot the average of the mean-squared error $\|\hat{\mathbf{x}} - \mathbf{x}^*\|_2^2/\|\mathbf{x}^*\|_2^2$ along with corresponding standard deviations as error bar for different values of $k = 1, 100, 200, \ldots, 1000$.

   We intentionally did not specify the method for generating a random subspace or for sampling points within it; you are encouraged to make a reasonable choice.