Solutions Exercise 4

Question 1

Let us first look at the gradient of the regularized problem at $\mathbf{w}_0$. The regularized loss is given by

$$L(\mathbf{X}, y) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T\mathbf{w},$$

and the gradient w.r.t to $\mathbf{x}$ is hence

$$\frac{\partial L(\mathbf{X}, y)}{\partial \mathbf{w}} = \frac{\partial (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})}{\partial \mathbf{w}} + \frac{\partial \lambda \mathbf{w}^T\mathbf{w}}{\partial \mathbf{w}}.$$

Using

$$\frac{\partial}{\partial \mathbf{w}}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

we get the final gradient as

$$\begin{aligned}\frac{\partial L(\mathbf{X}, y)}{\partial \mathbf{w}} &= 2\mathbf{w}\lambda - 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= 2\mathbf{w}\lambda - 2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w}\end{aligned}$$

Now, we evaluate the gradient at w_0, using $\mathbf{w}_0 = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ or $2(\mathbf{X}^T\mathbf{X})\mathbf{w}_0 = 2\mathbf{X}^T\mathbf{y}$, to get:

$$\begin{aligned}&= 2\mathbf{w}_0\lambda - 2\mathbf{X}^T\mathbf{X}\mathbf{w}_0 + 2\mathbf{X}^T\mathbf{X}\mathbf{w}_0 \\ &= 2\mathbf{w}_0\lambda\end{aligned}$$

We see that at the solution of the unregularized problem the gradient of the regularized problem aligns exactly with the solution, pointing away from the zero. This means that the solution is being pulled towards zero along the solution vector itself.

We can reach a similar observation by directly expressing $\mathbf{w}_\lambda$ in terms of $\mathbf{w}_0$, starting with the solution

$$\mathbf{w}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1})^{-1}\mathbf{X}^T\mathbf{y}.$$

Since the solution for the unregularized problem exists we know that $\mathbf{X}^T\mathbf{X}$ is invertible: $(\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1} = \mathbb{1}$. We can then write the solution as

$$\mathbf{w}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1})^{-1}(\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

where the last few terms are easily recognized as the solution for the unregularized problem, resulting in

$$\mathbf{w}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1})^{-1}(\mathbf{X}^T\mathbf{X})\mathbf{w}_0.$$

To further simplify this expression we can use $B^{-1}A^{-1} = (AB)^{-1}$ to reach

$$\begin{aligned}
\mathbf{w}_\lambda &= \left((\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1})\right)^{-1}\mathbf{w}_0 \\
\mathbf{w}_\lambda &= \left((\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X}) + \lambda(\mathbf{X}^T\mathbf{X})^{-1}\mathbb{1})\right)^{-1}\mathbf{w}_0 \\
\mathbf{w}_\lambda &= (\mathbb{1} + \lambda(\mathbf{X}^T\mathbf{X})^{-1})^{-1}\mathbf{w}_0.
\end{aligned}$$

We see that the regularized solution is obtained by pre-multiplying the unregularized solution with something that becomes the identity matrix if $\lambda = 0$. For $\lambda \to \infty$ the solution becomes a zero vector.

Question 2

The code for this question can be found in the Moodle page. Using the parameters given in the problem setting we ran the algorithm for 1000 iterations. The two parameters(σ_p and σ_q) determine the scale of neighborhood. Since the original input data has large values and hence large distances between the points, the neighborhoods would collapse to single points if σ_p was too small. With too large value all points are each others neighbors.

The parameter σ_q determines the range of the output values. Since we anyway optimize over the coordinates directly, we would get the same solution by scaling the coordinates and σ_q in inverse relation, so we can just as well fix σ_q to some value that is reasonably good for the initialization.

Both the PCA initialization and the resulting visualization are shown in Figure 1 for 1000 samples. The algorithm pushes the samples towards the outer circle, and the resulting visualization is not much better than PCA. It does, however, better utilize the visualization space. It is possible to get better visualizations by more careful choice of the parameters – the ones given were intentionally not perfect so that you could play around and find a better solution if interested. The optimization problem is definitely not easy, and even the original authors of the algorithm suggest some rather weird hacks like adding random noise after each gradient update.

Figure 1 also shows one example of a more interesting visualization, obtained with the step-size 0.001 Note that the illustration shown on the lectures was for t-SNE, which was designed to retain the local structure better, not for classical SNE algorithm. Hence we should not expect as nice visualizations here.

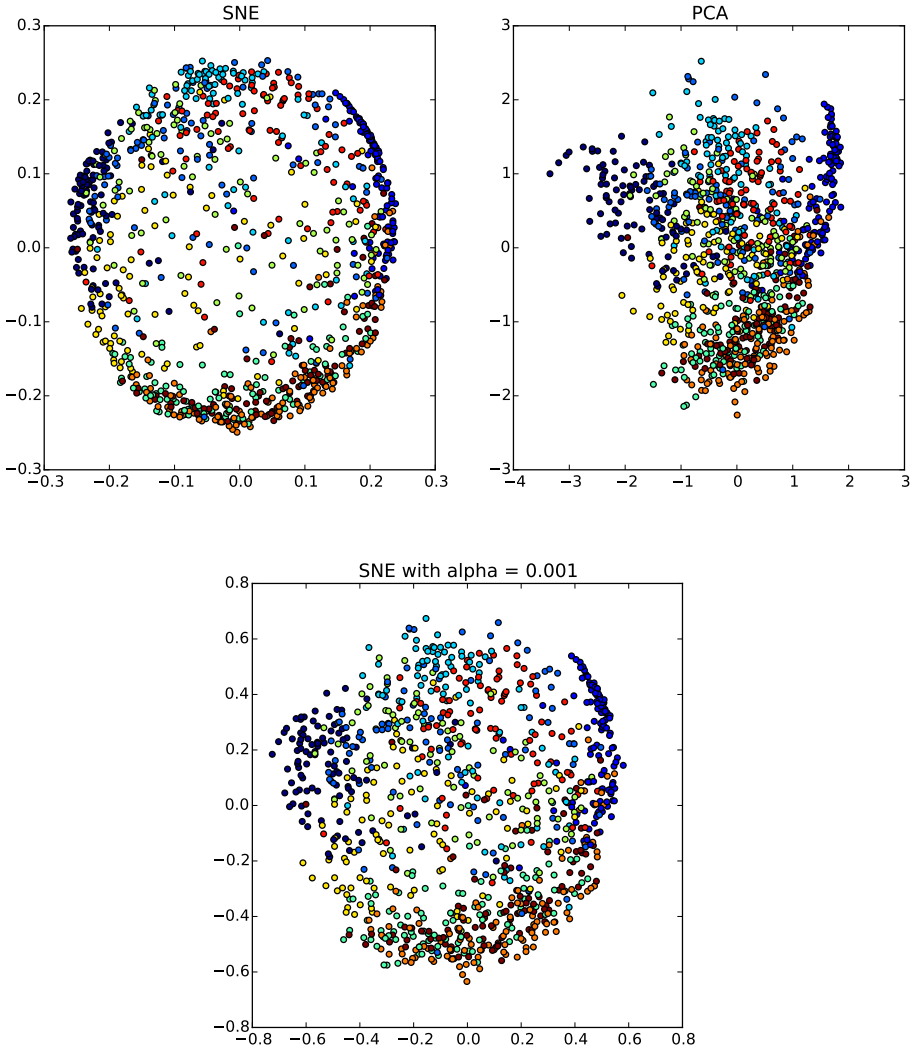

Figure 1: The top-right plot shows PCA representation with 2 dimensions, and the top-left plot is the SNE representation. Your solution should look pretty much like these since all of the optimization parameters were fixed in the exercise, but it is definitely possible to get prettier visualizations out of SNE. The bottom plot is SNE representation for smaller step size(0.001) for 1000 iterations.

Question 3

The code for this question can be found in the Moodle page. The solutions for three choices: $K = 8$, $K = 16$ and $K = 64$ are illustrated in Figure 3. The solutions with smaller K start to resemble the filters as shown in the lectures, but to learn proper filters we would need both more training data and better preprocessing for the data –

here the patches were simply forced to be non-negative. Figure 2, shows the training and validation losses as a function of iterations for all choices of K. It took around 503, 722, and 1132 iterations, for K = 8, 16, and 64 respectively, using $\mid \text{cost}_{train}^{t+1} - \text{cost}_{train}^{t} \mid < 10^{-3}$ as stopping criterion. We can see that the loss decreases with increase in K. Table 1 compares the corresponding training and validation errors against PCA. Note that PCA should by definition have smaller training loss – it is guaranteed to find a global optimum and the constraints in NMF can only make the training loss higher. For the validation error, NMF could in some cases be better, but this is not the case here.

	Training loss	Validation loss
NMF(K =64)	16	22
PCA(K=64)	13	17
NMF(K = 16)	43	55
PCA(K = 16)	39	52
NMF(K=8)	56	71
PCA(K=8)	53	68

Table 1: PCA implementation has lower loss for training and validation sets compared to NMF. The loss values are normalized with their corresponding sample size.

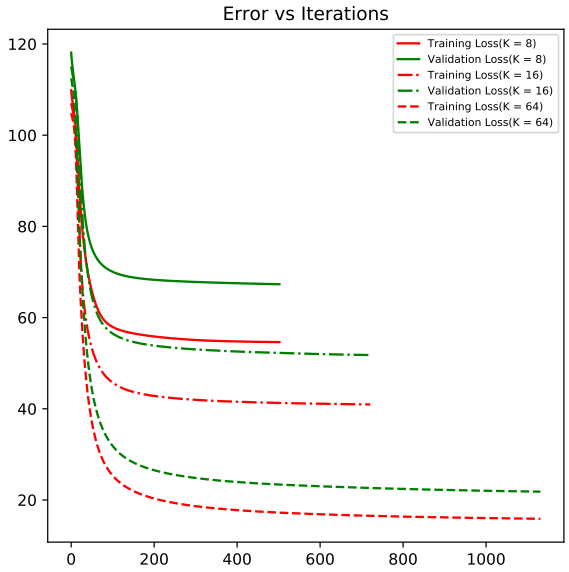

Figure 2: Both the training and validation errors have similar behaviour, with number of iterations. The plot is for NMF with ranks K = 8, 16, 64. The case with 64 basis vectors has the least error

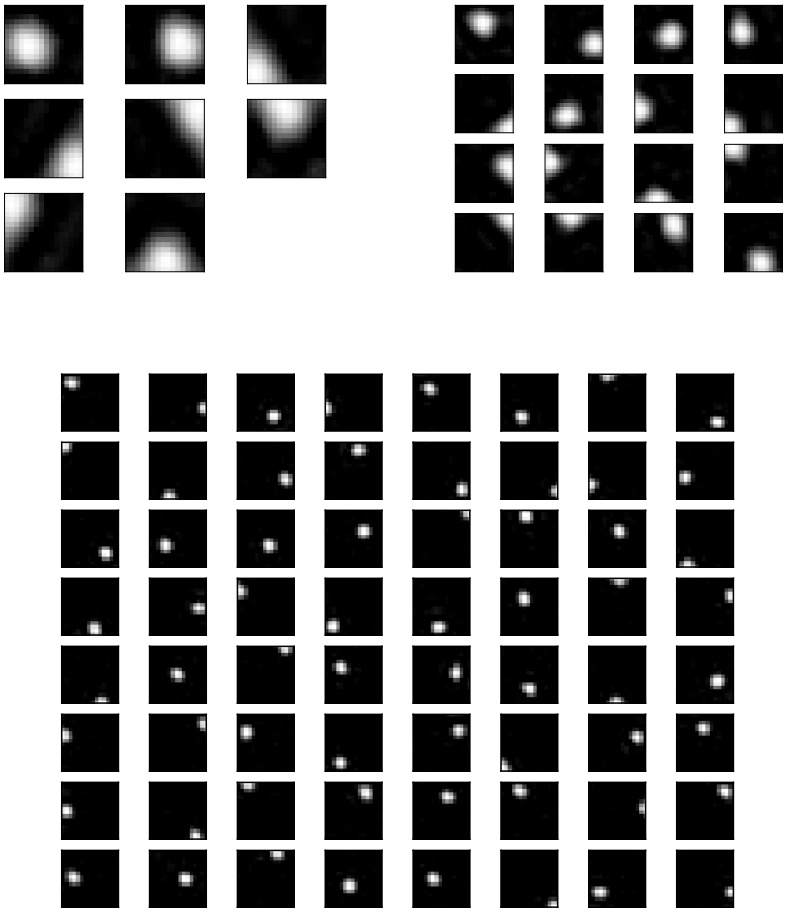

Figure 3: The figure illustrates the basis vectors from NMF using different ranks. The top-left image uses 8 basis vectors, top-right figure uses 16 basis vector followed by the bottom figure using 64 basis vector. Increasing the basis vectors gives very sparse vectors. The figure with 8 basis vector looks like a good choice for NMF.

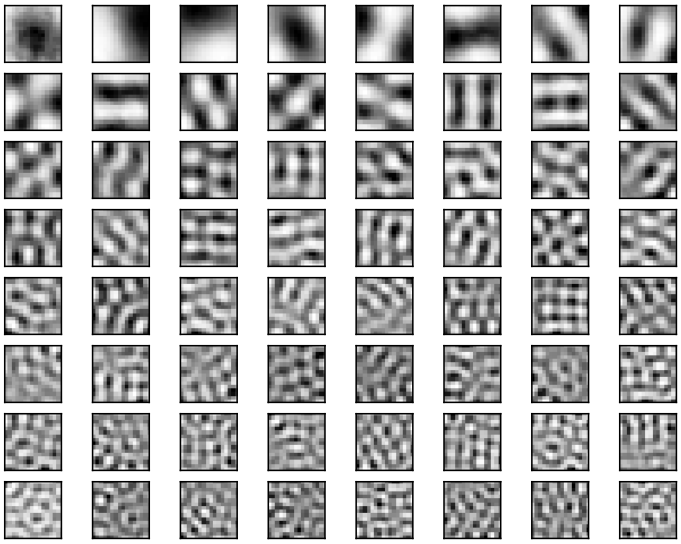

Figure 4: Image visualisation of 64 principal componenets arranged in the order of variance explained.