Data 598 (Winter 2022): Homework 4

1 Denoising AutoEncoders and Step Decay Learning Rates

In this exercise, we will use autoencoders to denoise (= de-noise, or remove the noise from) an image. We will also implement a step decay learning rate, a commonly used trick (for all deep kinds of deep nets, not just autoencoders).

Suppose we have and image x and "corrupt" it by some means to get $x' = C(x)$. Example corruptions including adding Gaussian noise or deleting random patches in the image. A denoising autoencoder with encoder h_w and decoder g_v (with respective parameters w and v) takes in the corrupted input x' and returns $\hat{x} = g(h(x'))$ that approximates well the noise-less image x.

We will train a denoising autoencoder to reconstruct the noiseless images from the noisy ones, by minimizing the corresponding reconstruction error:

$$\min_{w,v} \mathbb{E}_x \left\| x - g_v \circ h_w\big(C(x)\big) \right\|^2 .$$

We will use a step decay learning rate schedule

$$\gamma_t = \frac{\gamma_0}{2^{\lfloor t/t_0 \rfloor}} ,$$

in epoch t, where γ_0 is a given initial learning rate, and t_0 threshold.[1] The learning is cut by a factor of 2 every t_0 epochs:

$$\underbrace{\gamma_0, \cdots, \gamma_0}_{t_0 \text{ epochs}}, \underbrace{\frac{\gamma_0}{2}, \cdots, \frac{\gamma_0}{4}}_{t_0 \text{ epochs}}, \underbrace{\frac{\gamma_0}{4}, \cdots, \frac{\gamma_0}{4}}_{t_0 \text{ epochs}}, \cdots$$

A larger learning rate makes faster progress initially whereas a smaller learning rate is more helpful closer to convergence. The step-decay schedule aims to get the best of both worlds.

Task Your task is as follows:

- Use the MNIST dataset. Perform the same preprocessing as in this week's lab.
- Use the same convolutional autoencoder as in this week's lab, with a lower latent dimension of 40.
- As the corruption function $C(\cdot)$, we zero out a randomly chosen 14×14 patch in the original image. The code for this is provided in the lab again.
- Train the model for 40 epochs starting with $\gamma_0 = 2.5 \times 10^{-4}$ and take $t_0 = 10$ (i.e., halve the learning rate every 10 epochs).

In your submission, show some examples of the denoising process from the test set.

[1] In practice, the base of the exponent can also be any other number, e.g., 4 or 10.

2 (Bonus) AutoEncoders as a non-linear PCA

In this exercise, we will compare autoencoders versus PCA for dimensionality reduction. We will note their usefulness on the end goal of training a linear model using the extracted low-dimensional features.

In the first few labs, we used images as 784 dimensional vectors. Here, you will use either autoencoders or PCA on the training dataset to project the data onto a lower dimension d. You will then train a multinomial logistic regression model with scikit-learn and keep track of the test accuracy.

Task Your tasks are as follows:

- We will use the MNSIT dataset – use the same data setup as the lab.
- Given a lower dimension d, use PCA to reduce the dimensionality of the training set to d dimensions. Transform the test set by projecting on to the same space. You may use scikit-learn's PCA implementation.
- Train an autoencoder using the same settings as the lab, but with a hidden dimension as d. Train it for 40 epochs. Use the encoder to obtain d-dimensional representations for all training and test images.
- Train a multinomial logistic regression model with scikit-learn using each of the representations you have obtained.
- Repeat this for lower dimension $d = 10, 25, 50, 100$.

Make a plot with d on the x-axis and the best test accuracy of the logistic regression model on the y-axis with the d-dimensional representations. The plot should have two lines, corresponding to PCA and autoencoders.

Based on these observations, could you speculate why one of the two might be better or worse than the other?