Autoencoders

Eyal Gur

January 18, 2024

Let $\mathcal{X}$ be a dataset of N unlabeled points $\mathbf{x}^i \in \mathbb{R}^n$. Let $\mathcal{E}_\theta \colon \mathbb{R}^n \to \mathbb{R}^m$ and $\mathcal{D}_\phi \colon \mathbb{R}^m \to \mathbb{R}^n$ (usually $m < n$) be a decoder/encoder function, respectively, and consider the ℓ_1-regularized sparse autoencoder minimization problem

$$\min_{\phi,\theta} \ \mathcal{A}(\phi,\theta) \equiv \sum_{\mathbf{x}\in\mathcal{X}} \left(\left\| \mathbf{x} - \mathcal{D}_\phi\left(\mathcal{E}_\theta(\mathbf{x})\right) \right\|^2 + \lambda \left\| \mathcal{E}_\theta(\mathbf{x}) \right\|_1 \right).$$

For convenience, we define the two functions $f(\phi,\theta) \equiv \sum_{\mathbf{x}\in\mathcal{X}} \left\| \mathbf{x} - \mathcal{D}_\phi\left(\mathcal{E}_\theta(\mathbf{x})\right) \right\|^2$ and $g(\theta) \equiv \sum_{\mathbf{x}\in\mathcal{X}} \left\| \mathcal{E}_\theta(\mathbf{x}) \right\|_1$, thus the autoencoder can be written as

$$\min_{\phi,\theta} \ f(\phi,\theta) + \lambda g(\theta).$$

In this document, we assume that both decoder and encoder are parametarized with differentiable activation functions (for instance, sigmoid). A somewhat weaker suitable assumption is that the function f is differentiable at any data point $\mathbf{x} \in \mathcal{X}$.

Developing the ANAMO Algorithm

The structure of the autoencoder as a minimization problem that possesses two blocks of variables ϕ and θ, naturally calls for my *Nested Alternating Minimization* (NAM) scheme, which is an optimization scheme developed and analyzed in my PhD thesis. By applying this scheme, we solve the minimization problem iteratively in the following alternating fashion: first, we fix ϕ and minimize the partial function $\theta \mapsto \mathcal{A}(\phi,\theta)$ using some nested algorithm. Then, we fix θ with its updated value and minimize the partial function $\phi \mapsto \mathcal{A}(\phi,\theta)$ using some nested algorithm. The process is repeated until a stopping criterion is satisfied.

Focusing on the partial function with respect to the encoder, that is $\theta \mapsto \mathcal{A}(\phi,\theta)$, this function is the sum of the differentiable partial function $\theta \mapsto f(\phi,\theta)$, and the non-differentiable regularization function $g(\theta)$. Hence, the minimization problem with respect to the encoder can be minimized, for example, using the *FISTA* algorithm (a variant of Accelerated Gradient for the non-differentiable composite setting).

As for the decoder, we notice that the partial function $\phi \mapsto \mathcal{A}(\phi, \theta)$, is differentiable as it coincides with $\phi \mapsto f(\phi, \theta)$. Therefore, the minimization problem with respect to the decoder can be minimized using inner iterations of the Accelerated Gradient method (or its stochastic variants).

My overall suggested algorithm, called *Autoencoder NAM Optimizer* (ANAMO), is formulated now in Algorithm 1. The gradients in this algorithm can be calculated, for example, with backpropagation.

Algorithm 1 Autoencoder NAM Optimizer (ANAMO)

Initialization: Random (ϕ^0, θ^0), learning rates $\{e_j\}_{j\geq 0}$ and $\{d_j\}_{j\geq 0}$, a sequence $\{s_j\}_{j\geq 0}$ such that $s_{j+1} = \left(1+\sqrt{1+4s_j^2}\right)/2$ for $s_0 = 1$, and mini-batches $\{\mathcal{X}_k\}_{k\geq 0} \subseteq \mathcal{X}$.
for $k \geq 0$ **do**
  Update the encoder for mini-batch k:
  Set $\tilde{\theta}^0 = \mathbf{y}^0 = \theta^k$.
  for $j = 0, 1, \ldots, J-1$ **do**
    Update $\tilde{\theta}^{j+1} = \text{argmin}_{\mathbf{y}}\left\{\lambda g(\mathbf{y}) + \frac{1}{2}\left\|\mathbf{y} - \mathbf{y}^j + e_j \nabla_\theta f\left(\phi^k, \mathbf{y}^j\right)\right\|^2\right\}$.
    Update $\mathbf{y}^{j+1} = \tilde{\theta}^{j+1} + \frac{s_j - 1}{s_{j+1}}\left(\tilde{\theta}^{j+1} - \tilde{\theta}^j\right)$.
  end for
  Set $\theta^{k+1} = \tilde{\theta}^J$.
  Update the decoder for mini-batch k:
  Set $\tilde{\phi}^0 = \mathbf{y}^0 = \phi^k$.
  for $j = 0, 1, \ldots, J-1$ **do**
    Update $\tilde{\phi}^{j+1} = \mathbf{y}^j - d_j \nabla_\phi f\left(\mathbf{y}, \theta^{k+1}\right)$.
    Update $\mathbf{y}^{j+1} = \tilde{\phi}^{j+1} + \frac{s_j - 1}{s_{j+1}}\left(\tilde{\phi}^{j+1} - \tilde{\phi}^j\right)$.
  end for
  Set $\phi^{k+1} = \tilde{\phi}^J$.
end for

Theoretical Justifications for ANAMO

There are several theoretical justifications for ANAMO (that is, applying NAM with FISTA for autoencoders):

1. Convergence rate: the FISTA method is known to enjoy a fast linear convergence rate (that is, $\mathcal{O}(1/n^2)$ where n is the number of iterations). This is compared to other gradient-based methods, such as Gradient Descent and some of its stochastic variants, that only have sub-linear convergence rate (that is, $\mathcal{O}(1/n)$).

2. Global convergence: in my thesis, I formulated and proved convergence guarantees of the NAM scheme to a first-order optimal solution in some non-convex and non-differentiable cases. These results hold even though that FISTA, as a stand-alone algorithm, do not possesses such theoretical convergence guarantees. This implies that under certain settings, ANAMO may converge to first-order optimal solutions of the minimization problem.

3. Adjustable optimization process for encoder and decoder: the optimization landscape need not be similar for the encoder and decoder, and in fact it can be quite different as the encoder sets the input for the decoder. Therefore, it is favorable to allow flexibility in the optimization process of the decoder and encoder, which can be obtained by using ANAMO and *not* by other gradient-based methods:

   (a) The step-sizes (which can also be adaptable) can be set *differently* for the updates of the encoder and decoder.

   (b) One can apply a *different* number of iterations to update the weights of the encoder and decoder.

   (c) Even a different type of a nested optimization algorithm can be set for the encoder and the decoder.

4. Regularization: a regularization term can be easily incorporated to the autoencoder (though the optimization process need *not* be simple), as FISTA can handle such terms.

Comparison of ANAMO and ADAM

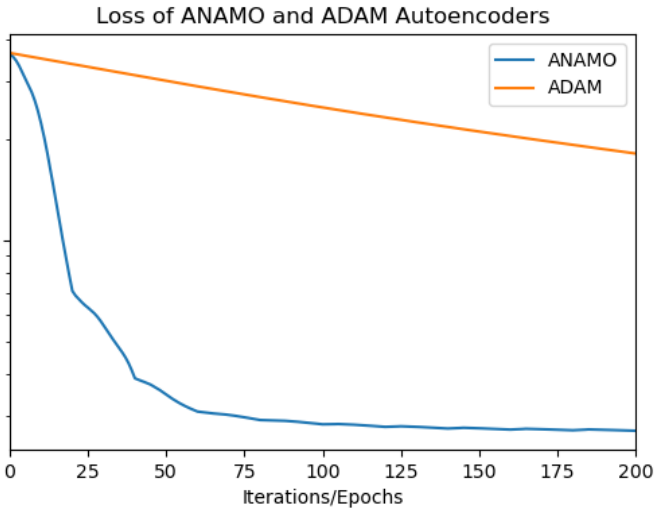
For the sake of a simple comparison, we compare the two methods for a single-layered encoder and decoder, both with a sigmoid activation function. The NumPy implementation of ANAMO for this specific instance (including backpropagation) is attached to this document.

We synthesized 100 random gray-scale training images of size 10×10 pixels, each with a whiter left side compared to its blacker right side. The size of the hidden layer was set to 10. For both algorithms, we set the same random initial weights for the encoder and decoder.

For ANAMO, we set 5 (inner) update iterations for the encoder and 15 for the decoder. In addition, we set 10 outer iterations. The learning rates for both were set to n, and the regularization constant was set to $\lambda = 0.1$.

For ADAM, we used its default hyper-parameters. The number of epochs was set to $10 \times (5 + 15) = 200$ (just for the sake of a simpler comparison, as this number includes more total iterations compared to ANAMO).

1. In the results below, we compared the value of the loss (as formulated at the beginning of this document) along the iterations/epochs.



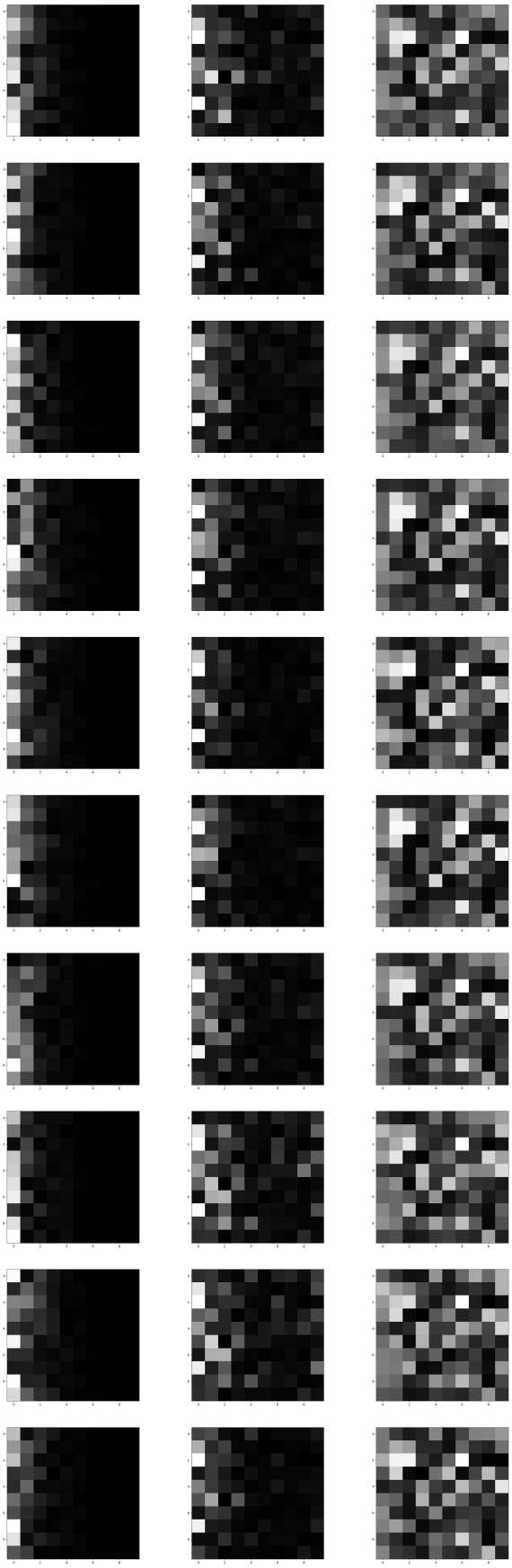
2. In addition, we synthesized 10 random test images and compared the average reconstruction bias for both algorithms. The formula for the average bias is

$$\frac{1}{M}\sum_{i=1}^{M}\left\|\mathbf{x}^i_{\text{real}}-\mathbf{x}^i_{\text{pred}}\right\|,$$

where M is the size of the test-set, $\mathbf{x}^i_{\text{real}}$ is a test image, and $\mathbf{x}^i_{\text{pred}}$ is its prediction by one of the algorithms.

For the test set, ANAMO obtained a value of 1.85 and ADAM of 3.70.

3. Last, we plot the test images (left column), along with the predictions made by ANAMO (middle column), and by ADAM (right column).



With the results above, there are still some questions that need to be addressed in the future regarding ANAMO. For example: how the implementation of ANAMO can be extended, theoretically and numerically, to more complex activation functions? How can we calculate or approximate the prox term (see step 6 in ANAMO) in more complex settings? How can we estimate suitable step-sizes for the encoder and decoder? What is a suitable number of inner iterations that will guarantee good results but will not increase the overall running time?

Binary Encoder

In this case, since the decoder is left unchanged, we can still update the decoder in the same way as previously discussed (see steps 10–16 in ANAMO). However, since $\mathcal{E}_\theta(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}$, is now a binary vector, there are no simple means to calculate the involved prox term for the encoder with its regularization term (see step 6 in ANAMO). We now propose three ways to try and tackle this issue, and to approximate a suitable encoder for the regularized problem.

One way, is to apply inner iterations of some nested algorithm (for example, sub-gradient descent) to approximate the prox term. This is formulated in Algorithm 2. However, a convenient way for finding sub-gradients for the partial function with respect to the encoder must be obtained.

Algorithm 2 ANAMO with Binary Encoder – V1

1: **Initialization:** Random (ϕ^0, θ^0), learning rates $\{e_j\}_{j\geq 0}$ and $\{d_j\}_{j\geq 0}$, a sequence $\{s_j\}_{j\geq 0}$ such that $s_{j+1} = \left(1+\sqrt{1+4s_j^2}\right)/2$ for $s_0 = 1$, and mini-batches $\{\mathcal{X}_k\}_{k\geq 0} \subseteq \mathcal{X}$.
2: **for** $k \geq 0$ **do**
3: *Update the encoder for mini-batch k:*
4: Set $\tilde{\theta}^0 = \mathbf{y}^0 = \theta^k$.
5: **for** $j = 0, 1, \ldots, J-1$ **do**
6: Apply iterations of sub-gradient descent to approximate a solution $\tilde{\theta}^{j+1}$ to step 6 of ANAMO.
7: Update $\mathbf{y}^{j+1} = \tilde{\theta}^{j+1} + \frac{s_j - 1}{s_{j+1}}\left(\tilde{\theta}^{j+1} - \tilde{\theta}^j\right)$.
8: **end for**
9: Set $\theta^{k+1} = \tilde{\theta}^J$.
10: *Update the decoder for mini-batch k:* repeat steps 11–16 of ANAMO
11: **end for**

Another way to tackle this problem is to apply some smooth approximation function of the $\{0,1\}$-step activation function (for example, the function $3\left(\frac{x}{\epsilon}\right)^2 - 2\left(\frac{x}{\epsilon}\right)^3$ is a suitable smooth choice, though it introduces $\epsilon > 0$ as an additional hyper-parameter). With this

smooth approximating activation function, we can use ANAMO under the setting discussed in Algorithm 1.

An additional way, is to notice that we are trying to find a binary representation $\mathbf{h}^i$ for each data point $\mathbf{x}^i \in \mathcal{X}$ in the dataset, $i = 1, 2, \ldots, N$. A possible way to find suitable binary representations is to solve the simplified regularized problem

$$\min_{\phi, \tilde{\mathbf{h}}} \mathcal{A}\left(\phi, \tilde{\mathbf{h}}\right) \equiv \sum_{\mathbf{x} \in \mathcal{X}} \left(\left\| \mathbf{x} - \mathcal{D}_\phi\left(\tilde{\mathbf{h}}^i\right) \right\|^2 + \lambda \left\| \tilde{\mathbf{h}}^i \right\|_1 \right),$$

which can be solved, for instance, using ANAMO (or any other suitable method, such as sub-gradient descent). Once the optimization process is completed, and in order to obtain the the final binary encodings, we update $\mathbf{h}^i \equiv \text{step}\left(\tilde{\mathbf{h}}^i\right)$ for all $i = 1, 2, \ldots, N$, where step is the $\{0, 1\}$-step function. Finally, in order to estimate the encoder $\mathcal{E}_\theta$, we solve the set of equations $\mathcal{E}_\theta\left(\mathbf{x}^i\right) = \mathbf{h}^i$ for $i = 1, 2, \ldots, N$, which can be performed, for example, by means of regression. This algorithm is given in Algorithm 3.

Algorithm 3 ANAMO with Binary Encoder - V3

1: **Initialization:** Random $\left(\phi^0, \tilde{\mathbf{h}}^0\right)$, learning rates $\{e_j\}_{j \geq 0}$ and $\{d_j\}_{j \geq 0}$, a sequence $\{s_j\}_{j \geq 0}$ such that $s_{j+1} = \left(1 + \sqrt{1 + 4s_j^2}\right)/2$ for $s_0 = 1$, and mini-batches $\{\mathcal{X}_k\}_{k \geq 0} \subseteq \mathcal{X}$.
2: **for** $k \geq 0$ **do**
3: *Update the representations* $\tilde{\mathbf{h}}$ *for mini-batch* k*:* repeat steps 4–9 of ANAMO, where $\tilde{\mathbf{h}}$ is replacing θ.
4: *Update the decoder for mini-batch* k*:* repeat steps 11–16 of ANAMO.
5: **end for**
6: Update $\mathbf{h} \equiv \text{step}\left(\tilde{\mathbf{h}}\right)$.
7: Approximate the encoder $\mathcal{E}_\theta$ by solving the system of equations $\mathcal{E}_\theta\left(\mathbf{X}\right) \approx \mathbf{h}$.