Deep learning for trivial inverse problems

Peter Maass

November 15, 2018

Abstract Deep learning is producing most remarkable results when applied to some of the toughest large scale non-linear problems such as classification tasks in computer vision or speech recognition. Recently, deep learning has also been applied to inverse problems, in particular in medical imaging. Some of these applications are motivated by mathematical reasoning, but a solid and at least partially complete mathematical theory for understanding neural networks and deep learning is missing.
In this paper we do not address large scale problems but aim at understanding neural networks for solving some small and rather naive inverse problems. Nevertheless, the results of this paper highlight the particular complications of inverse problems, e.g. we show that applying a natural network design for mimicking Tikhonov regularization fails when applied to even the most trivial inverse problems.
The proofs of this paper utilize basic and well known results from the theory of statistical inverse problems. We include the proofs in order to provide some material ready to be used in student projects or general mathematical courses on data analysis. We only assume, that the reader is familiar with the standard definitions of feed forward networks, e.g. the backpropagation algorithm for training such networks. We also include - without proof - numerical experiments for analyzing the influence of the network design, which include comparisons with LISTA (learned iterative soft thresholding algorithm).

1 Motivation and outline

No matter in which field of science we are working and no matter which type of conferences or meetings we are attending one of the hottest topics being discussed over the last few years are neural networks for large data applications. The success of such deep learning (DL) applications are stunning indeed in terms of their apparent success for large scale real life applications, see e.g. [12, 18], but also with respect to the almost complete lack of theoretical justification.
While arguing and having experimental results is a sufficient foundation in some fields of sciences, it is not satisfactory in mathematics, where the concept of having a strict proof is essential. Some mathematical concepts for analyzing deep learning approaches are slowly emerging [19, 23, 3, 7, 1] and we want to add some basic results for the particular case of DL for inverse problems on the low and almost trivial side of complexity.
Our starting point are numerical experiments for linear systems $Ax = y$ with given noisy data y^δ. Surprisingly, even small two by two examples cannot be solved reliably by straight forward neural networks. Here we employ minimal networks which are capable of learning a matrix-vector multiplication, i.e. such networks should be able to learn classical Tikhonov regularizers $(A^*A + \alpha I)^{-1}A^*$ or even better approximations. This small scale setting allows a somewhat complete analysis of the neural network, in particular we can prove the shortcomings of such neural networks if the condition number of the matrix and the noise level in the data are in a critical relation. The extension to linear systems of arbitrary dimension is straight forward.
On a conceptual level, when comparing inverse problems defined by analytical models $A : Z \to V$ with data driven approaches such as deep learning one would expect the data driven approaches to have certain advantages if e.g. A is an incomplete model of the underlying physical or engineering system or if the searched for parameter x is actually restricted to a characteristic subset $Z_d \subset Z$, which however escapes precise mathematical modeling. In both situations the missing information is implicitly contained in

sufficiently large sets of experimentally measured pairs of data $(x^i, y^i)_{i=1,..,n}$. The assumption, that A is only an incomplete model is not relevant for our small examples. However, we can test the potential of neural networks for learning the underlying structure or prior distribution of restricted parameter sets. Hence, we extend our experiments to the non-linear problem of solving linear inverse problems with sparsity constraints [9, 4, 15, 2]. Here we compare the results obtained by classical ISTA (iterated soft thresholding) with its learned counterpart LISTA, [13]. The experimental results demonstrate that LISTA is not better than ISTA if one takes any set of sparse vectors as inputs. However, it performs significantly better if we assume structured sparsity of the inputs, i.e. if we restrict them to a low dimensional subspace. In this case the performance of ISTA does not change, but LISTA seems to unveil the underlying subspace and produces significantly improved results.

2 Basic example

We consider the basic inverse problem given by an operator $A : Z \to V$ and some noisy data $y^\delta = Ax + \eta$. We further assume that a set of training data $(x^{(i)}, y^{(i)})_{i=1,..,n}$ is available for training a neural network. We will use this for training the forward operator, i.e. the set of $x^{(i)}$'s is input and $y^{(i)}$'s are the output, as well as for training the inverse problem, i.e. the set of $y^{(i)}$'s is input and $x^{(i)}$'s are the output.
To be precise, in our most basic example we set $Z = V = I\!R^2$ and

$$A_\varepsilon = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{pmatrix}.$$

This matrix has an orthogonal basis of eigenvectors $u_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \mathcal{O}(\epsilon^2)$, $u_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} + \mathcal{O}(\epsilon^2)$ and eigenvalues $\lambda_1 = 2 + \frac{\epsilon}{2} + \mathcal{O}(\epsilon^2)$, $\lambda_2 = \frac{\epsilon}{2} + \mathcal{O}(\epsilon^2)$. The ill-posedness of the problem - or rather the condition number of A - is controlled by $1/\varepsilon$, typical values are $\varepsilon = 10^{-k}$, $k = 0, ..., 10$.
As training data we draw n vectors $x^{(i)}, i = 1, .., n$, where each coefficient is i.i.d. $N(0, 1)$ normally distributed. The corresponding data vectors $Ax^{(i)}$ are corrupted with noise vectors $\eta^{(i)}$ where each coefficient of $\eta^{(i)}$ is drawn independently from a $N(0, \sigma^2)$ normal distribution, i.e.

$$y^{(i)} = Ax^{(i)} + \eta^{(i)} \ . \tag{1}$$

For later use we define $2 \times n$ matrices X (parameter matrix), Y (data matrix) and Θ (noise matrix) by storing columnwise the vectors $x^{(i)}, y^{(i)}, \eta^{(i)}$, i.e.

$$Y = AX + \Theta \ . \tag{2}$$

We now compare two methods for solving the inverse problem. The first one is the classical Tikhonov regularization, which only uses information about the operator A. I.e. for given data y we estimate the parameter x by $\hat{x}_{Tik} = (A^*A + \sigma^2 I)^{-1} A^* y$. The second inversion is based on a neural network Φ_W, which depends on a weight matrix W. W is obtained by training the neural network with respect to a so-called loss function. We use the standard least squares loss function $L_1(W) = \frac{1}{n} \sum_{i=1}^n \|\Phi_W(x^{(i)}) - y^{(i)}\|^2$ for training a net with parameters $W_{FP} = argmin\ L_1(W)$ for the forward problem and

$$L_2(W) = \frac{1}{n} \sum_{i=1}^n \|\Phi_W(y^{(i)}) - x^{(i)}\|^2 \tag{3}$$

for training $W_{IP} = argmin\ L_2(W)$ for solving the inverse problem. I.e. this approach does not use any knowledge about the operator A. After training the inverse problem is solved by simply applying the inverse net $\hat{x}_\Phi = \Phi_{W_{IP}}(y)$.
The training is followed by an evaluation using a different set of test data. We compare these methods by computing the mean error for the test data

$$E_{Tik} := \frac{1}{n} \sum_{i=1}^n \|\hat{x}_{Tik}^{(i)} - x^{(i)}\|^2 \text{ resp. } E_\Phi := \frac{1}{n} \sum_{i=1}^n \|\hat{x}_\Phi^{(i)} - x^{(i)}\|^2$$

The design of the network is crucial. For our first tests we use a minimal network which allows to reproduce a matrix vector multiplication. Hence, the network is capable - in principle - to recover the Tikhonov regularization operator or even an improvement of it. Here, we use a network with a single hidden layer with 4 nodes and the standard $ReLU$-activation function, i.e. $ReLU(z) = \max\{z, 0\}$ (Rectified linear units, $z \in I\!R$). For the motivation of our network design we observe $ReLU(z) - ReLU(-z) = z$. We restrict the eights weights connecting the input variables with the first layer by setting $w_1 = -w_3 = w_{11}, w_2 = -w_4 = w_{12}, w_5 = -w_7 = w_{21}, w_6 = -w_8 = w_{22}$ as depicted in Figure 1. We obtain a neural network depending on four variables $w_{11}, w_{12}, w_{21}, w_{22}$ and the networks acts as multiplication with matrix $W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ on the input vector $z = (z_1, z_2)$. We denote the output of such a neural network by $\phi_W(z) = Wz$.

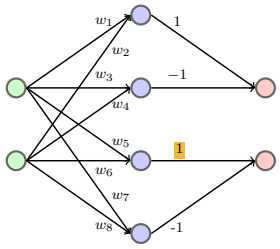

Figure 1: The network design with 8 parameters, setting $w_1 = -w_3 = w_{11}, w_2 = -w_4 = w_{12}, w_5 = -w_7 = w_{21}, w_6 = -w_8 = w_{22}$ yields a matrix-vector multiplication of the input.

The training of such a network for modeling the forward problem is equivalent (using the Frobenius norm for matrices) to minimizing the expected mean square error

$$min_{W \in I\!R^{2\times 2}} \frac{1}{n} \sum_{i=1}^{n} \|Wx^{(i)} - y^{(i)}\|^2 = min_W \frac{1}{n} \|WX - Y\|^2 \tag{4}$$

and training a model for the inverse problem is done by

$$min_W \frac{1}{n} \sum_{i=1}^{n} \|Wy^{(i)} - x^{(i)}\|^2 = min_W \frac{1}{n} \|WY - X\|^2 \quad . \tag{5}$$

In the next subsection we report some numerical examples before we analyze these networks.

2.1 Testing error convergence for various values of ε

We train these networks using[1] a set of training data $(x^{(i)}, y^{(i)})_{i=1,..n}$ with $n = 10.000$, i.e. $y^{(i)} = A_\epsilon x^{(i)} + \eta^{(i)}$. The network design with restricted coefficients as described above has 4 degrees of freedom $w = (w_{11}, w_{12}, w_{21}, w_{22})$. The corresponding loss function is minimized by a gradient descent algorithm, i.e. the gradient of the loss function with respect to w is computed by backpropagation [22, 20, 5]. We used 3.000 iterations (epochs) of this gradient descent for minimizing the loss function of a network for the forward operator using (4), resp. for training a network for solving the inverse problem using (5). The MSE errors on the training data were close to zero in both cases.
After training we tested the resulting networks by drawing $n = 10.000$ new data vectors $x^{(i)}$ as well as errors vectors $\eta^{(i)}$. The $y^{(i)}$ where computed as above.
In the following table we show the resulting values using this set of test data $NMSE_{forward} = \frac{1}{n} \sum_{i=1}^{n} \|Wx^{(i)} - y^{(i)}\|^2$ for the network trained for the forward problem, resp. $NMSE_{inverse} \frac{1}{n} \sum_{i=1}^{n} \|Wy^{(i)} - x^{(i)}\|^2$ for the network trained for the inverse problem.

Error/choice of ε	1	0.1	0.01	0.0001
NMSE (direct problem)	0.002	0.013	0.003	0.003
NMSE (inverse problem)	0.012	0.8	10	10

The errors of the inverse net are large and the computed reconstructions with the test data are meaningless. We have also evaluated the mean squared errors after each iteration of the training as depicted in Figures 2a and 2b. Here the values of the errors are shown for the first 3.000 iterates and for data produced with different values of ϵ.
We observe, that the training of the forward operator produces reliable results as well does the network for the inverse problem with $\varepsilon \geq 0.1$. However, training a network for the inverse problem with an ill-conditioned matrix A_ϵ with $\epsilon \leq 0.01$ fails.

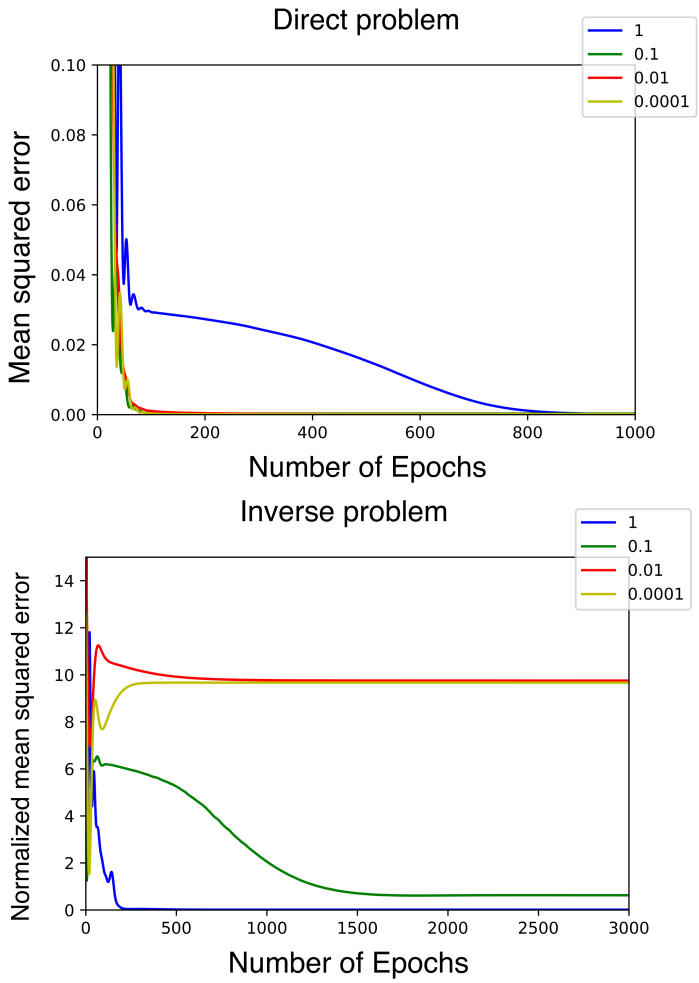

Figure 2: History of the values of the loss function during training of the forward map (left, Figure 2a) and the inverse problem (right, Figure 2b). ϵ is set to $1, 0.1, 0.01, 0.0001$.

This is confirmed by analyzing the values of w and of the resulting matrix W after training: We would assume, that in training the forward problem we produce values for w such that $W \sim A_\epsilon$ and that training the inverse problems leads to $W \sim (A^*A + \sigma^2)^{-1}A^*$. For the forward problem, the difference between W and A is of the order 10^{-3} or below, but for $\epsilon \leq 0.01$ the training of the inverse problem leads to a matrix, which has no similarity with the Tikhonov regularized inverse. Using a network with a single internal layer, but with more nodes and no restriction on the structure of the weights did not yield any

significant improvements.

3 Analysis of trivial neural networks for inverse problems

The numerical examples indicate, that training even a most simple neural network for a well posed matrix-vector multiplication (forward operator) yields good results. However, the natural approach for training a network for an inverse problem by reversing inputs and outputs fails in certain cases. In this section we aim to analyze, why this approach has its limitations for inverse problems.

3.1 Matrix case

We consider the training of the trivial network, see Figure 1, for solving an inverse problem with matrix $A = A_\epsilon$. I.e. $y^{(i)} = Ax^{(i)} + \eta^{(i)}$ are inputs to the network and the loss function is the mean squared error between the outputs of the network and the exact solutions $x^{(i)}$.
Hence, in this section we analyze the special case, were the application of the neural network is strictly equivalent to a matrix-vector multiplication. In our test example this refers to restricting the coefficients of the neural network to four coefficient $\tilde{w}_{11}, \tilde{w}_{12}, \tilde{w}_{21}, \tilde{w}_{22}$ and setting $w_1 = -w_3 = \tilde{w}_{11}, w_2 = -w_4 = \tilde{w}_{12}, w_5 = -w_7 = \tilde{w}_{21}, w_6 = -w_8 = \tilde{w}_{22}$. The output of the network yields

$$\phi(W, y) = Wy \quad \text{where} \quad W = \begin{pmatrix} \tilde{w}_{11} & \tilde{w}_{12} \\ \tilde{w}_{21} & \tilde{w}_{22} \end{pmatrix} .$$

Training of the network is equivalent to determining a matrix W which minimizes

$$min_W \ \frac{1}{n}\|WY - X\|^2 .$$

Analyzing this discrepancy functional is the classical situation in statistical inverse problems theory [8, 21, 16], where training W is equivalent to determining the MAP (maximum a posteriori) estimator. The optimal W is obtained as

$$W^T = (YY^T)^{-1}YX^T . \tag{6}$$

We analyze W by using (1) and obtain the following expression for YY^T:

$$YY^T = (AX + \Theta)(AX + \Theta)^T = AXX^TA^T + AX\Theta^T + \Theta X^TA^T + \Theta\Theta^T . \tag{7}$$

Lemma 3.1. *For a given A, we denote by Y the matrix of training data, by X the matrix containing the parameters in the training set and by Θ a noise matrix as defined in Section 2. Then*

$$\frac{1}{n}YY^T = (AA^T + \sigma^2 I) + R \quad \textit{with} \quad R = AB_1A^T + B_2 + AB_3 + B_4A^T \quad ,$$

where $B_1 = \frac{1}{n}XX^T - I$, $B_2 = \frac{1}{n}\Theta\Theta^T - \sigma^2 I$, $B_3 = \frac{1}{n}X\Theta^T$, $B_4 = \frac{1}{n}\Theta X^T$.
Then

$$0 = I\!E(B_1) = I\!E(B_2) = I\!E(B_3) = I\!E(B_4)$$

$$var(B_1) = \frac{1}{n}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \ , \ var(B_2) = \frac{\sigma^4}{n}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \ , \ var(B_3) = var(B_4) = \frac{\sigma^2}{n}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad .$$

Remark 3.1. *We will use some basic results for normally distributed random variables $Z \sim N(0, \sigma^2)$, see e.g. [11, 10]:*

$$I\!E(Z) = 0 \ , \ var(Z) = I\!E(Z^2) = \sigma^2 \ , \ var(Z^2) = I\!E(Z^4) - I\!E(Z^2)^2 = 2\sigma^4 .$$

If $Z^{(i)}$ are i.i.d. (not necessarily normally distributed), then

$$I\!E(\sum_{i=1}^{n} Z^{(i)}) = \sum_{i=1}^{n} I\!E(Z^{(i)}) \ , \ var(\sum_{i=1}^{n} Z^{(i)}) = \sum_{i=1}^{n} var(Z^{(i)}) \ .$$

If $Z, \tilde{Z}$ are i.i.d. $N(0,\sigma^2)$ random variables, then

$$I\!E(Z\tilde{Z}) = 0 \ , \ var(Z\tilde{Z}) = I\!E(Z^2\tilde{Z}^2) = I\!E(Z^2)I\!E(\tilde{Z}^2) = \sigma^4 \ .$$

Proof. By definition we have $\eta_1^{(i)}, \eta_2^{(i)}, i = 1, .., n$ are i.i.d. $N(0,\sigma^2)$ random variables and Θ is the corresponding $2 \times n$ matrix. Then

$$\Theta\Theta^T = \begin{pmatrix} \sum_{i=1}^n (\eta_1^{(i)})^2 & \sum_{i=1}^n \eta_1^{(i)}\eta_2^{(i)} \\ \sum_{i=1}^n \eta_1^{(i)}\eta_2^{(i)} & \sum_{i=1}^n (\eta_2^{(i)})^2 \end{pmatrix} = n(\sigma^2 I + B_2)$$

with $B_2 = \frac{1}{n}\begin{pmatrix} -n\sigma^2 + \sum_{i=1}^n (\eta_1^{(i)})^2 & \sum_{i=1}^n \eta_1^{(i)}\eta_2^{(i)} \\ \sum_{i=1}^n \eta_1^{(i)}\eta_2^{(i)} & -n\sigma^2 + \sum_{i=1}^n (\eta_2^{(i)})^2 \end{pmatrix}$.
For fixed n we obtain

$$I\!E(-n\sigma^2 + \sum_{i=1}^{n} (\eta_1^{(i)})^2) = -n\sigma^2 + \sum_{i=1}^{n} I\!E\left((\eta_1^{(i)})^2\right) = 0 \quad \text{and}$$

$$var\left(-\sigma^2 + \frac{1}{n}\sum_{i=1}^{n} (\eta_1^{(i)})^2\right) = \frac{1}{n^2}\sum_{i=1}^{n} var\left((\eta_1^{(i)})^2\right) = \frac{2}{n}\sigma^4 \ .$$

Similarly, by using the rules above and by defining the variance of a matrix componentwise we obtain

$$I\!E(\frac{1}{n}\Theta\Theta^T) = \sigma^2 I \ , I\!E(B_2) = 0 \text{ and } \ var(B_2) = var\left(\frac{1}{n}\Theta\Theta^T\right) = \frac{\sigma^4}{n}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \ .$$

Similarly, with $x_1^{(i)} \sim N(0,1)$ we obtain $XX^T = n(I + B_1)$ and

$$I\!E(\frac{1}{n}XX^T) = I \ \text{ and } \ var(\frac{1}{n}XX^T) = \frac{1}{n}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \ .$$

Setting $B_3 := \frac{1}{n}X\Theta^T$ and $B_4 := B_3^T$ yields

$$I\!E(X\Theta^T) = I\!E(\Theta X^T) = 0 \ \text{ and } \ var(B_3) = var(B_4) = \frac{\sigma^2}{n}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \ .$$

□

We now analyze $W^T = (\frac{1}{n}YY^T)^{-1}\frac{1}{n}YX^T$. With the notation for $B_1, .., B_4$ as in the lemma above and assuming that the reminder term R is small enough we obtain by using Neumann series

$$(\frac{1}{n}YY^T)^{-1} = (AA^T + \sigma^2 I)^{-1} + (AA^T + \sigma^2 I)^{-1}\sum_{k\geq 1} Q^k \quad . \tag{8}$$

with

$$Q = (AA^T + \sigma^2 I)^{-1}R = (AA^T + \sigma^2)^{-1}(AB_1A^T + B_2 + AB_3 + B_4A^T) \ . \tag{9}$$

This gives the expected result, namely, that training our specific network for the inverse problems tries to mimick Tikhonov regularization. However, this is only valid if Q is indeed small. We analyse the norms of Q and R in a series of lemmata. First we analyze the deterministic part of Q and R.

Lemma 3.2. *Let $A = A_\epsilon$ be defined as above. Then*

$$\|(AA^T + \sigma^2 I)^{-1}\| = \mathcal{O}\left(\frac{1}{\epsilon^2 + \sigma^2}\right) = \mathcal{O}\left(min(1/\epsilon^2, 1/\sigma^2)\right) \ , \quad \|(AA^T + \sigma^2 I)^{-1} A\| = \mathcal{O}\left(\frac{\epsilon}{\epsilon^2 + \sigma^2}\right) \ .$$

Proof. We use the specific form of our matrix A and the values of its eigenvalues as computed above. Classical arguments using the singular value decomposition of A show, that the eigenvalues ν_1, ν_2 of $(AA^T + \sigma^2 I)^{-1}$, resp. of $(AA^T + \sigma^2 I)^{-1} A$, are given by

$$\nu_1 = \frac{1}{\lambda_1^2 + \sigma^2} = \frac{1}{4} + \mathcal{O}(\epsilon + \sigma^2) \ \text{ and } \nu_2 = \frac{1}{\lambda_2^2 + \sigma^2} = \frac{1}{\epsilon^2/4 + \sigma^2}\left(1 + \mathcal{O}(\epsilon)\right),$$

$$\text{resp. } \ \nu_1 = \frac{\lambda_1}{\lambda_1^2 + \sigma^2} = \frac{1}{2} + \mathcal{O}(\epsilon + \sigma^2) \text{ and } \nu_2 = \frac{\lambda_2}{\lambda_2^2 + \sigma^2} = \frac{\epsilon/2}{\epsilon^2/4 + \sigma^2}\left(1 + \mathcal{O}(\epsilon)\right).$$

For symmetric matrices the spectral radius is equivalent to matrix norms. Hence, for small values of ϵ and σ we obtain the asymptotic estimate of the norm of these matrices, which is determined by the value of the second eigenvalue λ_2 of A, i.e.

$$\|(AA^T + \sigma^2 I)^{-1}\| = \mathcal{O}\left(\frac{1}{\epsilon^2 + \sigma^2}\right) = \mathcal{O}\left(min(1/\epsilon^2, 1/\sigma^2)\right) \ , \quad \|(AA^T + \sigma^2 I)^{-1} A\| = \mathcal{O}\left(\frac{\epsilon}{\epsilon^2 + \sigma^2}\right) \ .$$

□

Estimating the spectral radius of products of normally distributed random matrices with variable variances is the topic on ongoing research, see e.g. [24, 17]. Motivated by the results and conjectures stated in these papers, we take the expectation values of the individual entries of our random matrices as representative values for their respective norms. I.e. we take the componentwise estimates as an estimate for the spectral radius of B. These values are the square roots of the variances computed in Lemma 3.1. and we obtain the following corollary.

Corollary 3.1. *We define the 'pseudo spectral' radius $\tilde{\rho}(B)$ for $B \in \{B_i, i = 1, .., 4\}$, where B_i is defined as above by*

$$\tilde{\rho}(B) = max_{i,j}\left(\sqrt{I\!E(b_{ij}^2)}\right) \ ,$$

where $b_{i,j}$ are the entries of the matrix B. Then Lemma 3.1 implies

$$\tilde{\rho}(B_1) = \sqrt{\frac{2}{n}} \ , \quad \tilde{\rho}(B_2) = \sqrt{\frac{2\sigma^4}{n}} \ , \quad \tilde{\rho}(B_3) = \sqrt{\frac{2\sigma^2}{n}} \ , \quad \tilde{\rho}(B_4) = \sqrt{\frac{2\sigma^2}{n}} \ .$$

Combining the last two statements allows us to obtain an estimate of the norm of the four terms of Q in (9), e.g. we obtain for two of these expressions

$$\|(AA^T + \sigma^2 I)^{-1} A B_1 A^T\| \leq \|(AA^T + \sigma^2 I)^{-1} A\| \|B_1\| \|A^T\| = \mathcal{O}\left(\frac{\epsilon}{\sqrt{n}(\epsilon^2 + \sigma^2)}\right)$$

$$\|(AA^T + \sigma^2 I)^{-1} B_4 A^T\| \leq \|(AA^T + \sigma^2 I)^{-1}\| \|B_4\| \|A^T\| = \mathcal{O}\left(\frac{\sigma}{\sqrt{n}(\epsilon^2 + \sigma^2)}\right) \ .$$

Similarly we obtain estimates for the other terms, hence, componentwise

$$I\!E(\|Q\|) = \mathcal{O}\left(\frac{\epsilon + \sigma}{\sqrt{n}(\epsilon^2 + \sigma^2)}\right) = \mathcal{O}\left(\frac{1}{\sqrt{n}(\epsilon + \sigma)}\right) \ . \tag{10}$$

As we will see in the next lemma, this is actually a sharp estimate under rather weak assumptions. We only need to ensure, that the eigenvectors of the random matrices $B_1, .., B_4$ are in random position and do not align with the eigenvectors of A. We state the result only for the first term in the expression for Q the estimates for the other terms follows equivalently.

Lemma 3.3. *Let $\lambda_1, \lambda_2, u_1, u_2$, resp. ν_1, ν_2, v_1, v_2, denote eigenvalues and normalized eigenvectors of A, resp. B_1, such that the spectral radius is given by $\rho(A) = \lambda_1$, resp. $\rho(B_1) = \nu_1$. Assume, that there exist constants $c, \tilde{c} > 0$ such that*

$$| < u_1, v_1 > | \geq c \ , \ | < u_1, v_2 > | \geq c \ \text{ and } \ |1 - \nu_2/\nu_1| \geq \tilde{c} \ .$$

Then

$$\|(AA^T + \sigma^2 I)^{-1} A B_1 A^T\| \geq \tilde{c} c^2 \rho(AA^T + \sigma^2 I)^{-1} A) \rho(B_1) \rho(A) = \mathcal{O}\left(\frac{\epsilon}{\sqrt{n}(\epsilon^2 + \sigma^2)}\right) \ .$$

Proof. A and B_1 are symmetric matrices, hence the respective eigenvalues are real, the eigenvectors form an orthonormal basis and u_1, u_2 are also eigenvectors of $(AA^T + \sigma^2 I)^{-1} A$. However, the spectral radius of this matrix is given by the eigenvalue for u_2.
We consider $C = (AA^T + \sigma^2 I)^{-1} A B_1 A^T$ and obtain a lower estimate by computing Cu_1, where u_1 is an eigenvector corresponding to the largest eigenvalue of A, i.e. $Au_1 = \lambda_1 u_1 = \rho(A) u_1$. We use the expansion $u_1 =< u_1, v_1 > v_1 + < u_1, v_2 > v_2$ and obtain

$$B_1 u_1 = \nu_1 < u_1, v_1 > v_1 + \nu_2 < u_1, v_2 > v_2 \ .$$

We now expand $v_1, \ v_2$ in $\{u_1, \ u_2\}$ and observe the orthogonality of the eigenfunctions implies $< u_1, v_2 >< v_2, u_2 >= - < u_1, v_1 >< v_1, u_2 >$. Rearranging some terms we finally obtain

$$\|Cu_1\| \geq | < Cu_1, u_2 > | \geq \tilde{c} c^2 \rho(AA^T + \sigma^2 I)^{-1} A) \rho(B_1) \rho(A) \ .$$

□

This leads to the final result on the structure of W, which just summarizes the previous statements.

Theorem 3.1. *Let W, Q be defined as in Lemma 3.1 and (9). If $\|Q\| < 1$ then*

$$W^T = (\frac{1}{n} Y Y^T)^{-1} \frac{1}{n} Y X^T = (AA^T + \sigma^2 I)^{-1} A + (AA^T + \sigma^2 I)^{-1} \sum_{k \geq 1} Q^k A (I + B_1) + (AA^T + \sigma^2 I)^{-1} A$$

and $\mathbb{E}(W) = (A^T A + \sigma^2 I)^{-1} A^T$.
The coefficients of the matrix Q satisfy componentwise

$$\mathbb{E}(\|Q\|) = \mathcal{O}\left(\frac{1}{\sqrt{n}(\epsilon + \sigma)}\right) \ .$$

Proof. By definition $YX^T = A + AB_1$, hence, the claim for W^T follows directly from (9). The second part of the theorem follows from (10). □

Remark 3.2. *The neural network will train a matrix $W = (YY^T)^{-1} YX^T$, whose expectation values coincide with the Tikhonov regularizers $= (AA^T + \sigma^2)^{-1} A$. This is not a surprising result since T coincides with the classical MAP estimator of statistical inverse problems, see [16]. However, analysing its variance $\mathbb{E}(\|W - T\|^2)$ we are lead to analyze the spectral radius or norm of Q, which determines the convergence of the Neumann series in (8). Its behaviour is characterized in (10), which reflects the ill-posedness of the problem. No matter how many data points we have (fixed n), the deviation of W from T will be arbitrarily large if ϵ and σ are both small. Of course, we can also give this a positive meaning, e.g. the noise level acts as a regularizer, large σ yields more stable matrices W.*
This 2×2 problem is only a toy problem. If dealing with approximations to infinite dimensional inverse problems, then ϵ will tend to zero with increasing accuracy of any numerical approximation. In this case we cannot expect that simple neural networks as described above will yield meaning full results.

The above described derivations are validated by numerical experiments with variable ϵ and σ.

Error for $\sigma\backslash\varepsilon$	1	$1e-1$	$1e-2$	$1e-3$	$1e-4$
1	0.00889	0.01063	0.01082	0.01090	0.01076
1e-1	0.00548	0.06605	0.08061	0.08008	0.07962
1e-2	0.00058	0.03269	0.64470	0.79562	0.80357
1e-3	0.00006	0.00334	0.30427	6.36951	7.96213
1e-4	0.00001	0.00033	0.03193	3.03402	64.84077
1e-5	0.00000	0.00003	0.00320	0.31916	30.81500

Table 3. Analysis of absolute error rates. Errors are computed as $err = 1/n \sum_{i=1}^{n} \|\Phi_W(y_i) - x_i\|^2$

Error for $\sigma\backslash\varepsilon$	1	$1e-1$	$1e-2$	$1e-3$	$1e-4$
1	0.01881	0.02704	0.02682	0.02708	0.02682
1e-1	0.00220	0.01625	0.11276	0.15959	0.16173
1e-2	0.00022	0.00166	0.01590	0.15882	1.12600
1e-3	0.00002	0.00016	0.00159	0.01595	0.16058
1e-4	0.00000	0.00002	0.00016	0.00159	0.01601
1e-5	0.00000	0.00000	0.00002	0.00016	0.00158

Table 4: Analysis of relative error rates. Errors are computed as $err = 1/n \sum_{i=1}^{n} \|\Phi_W(y_i) - x_i\|^2 / \|x_i\|^2$

These tests demonstrate that there exists a critical linear relation between ϵ and σ, which leads to large error rates. For fixed ϵ the error rates can get smaller if the noise level σ is increased. This might be counter intuitive, since large noise levels should lead to less quality in the reconstructions. This is actually also the case here, but the table states the deviation from the Tikhonov matrix T, which depends itself on σ.

4 Further numerical tests

The findings of the previous section should be understood as a warning, that inverse problems do have their rather specific complications and just applying seemingly suitable networks for solving inverse problems can fail miserably. Hence, investigating neural networks specifically for inverse problems makes sense and, of course, this has been done for all kinds of inverse problems already. Some most remarkable papers address unrolling of iteration schemes for solving inverse problems [13, 1], optimizing proximal mappings [14] or on constructing suitable penalty terms by neural networks [6, 7]. However, they lack a thorough convergence analysis.
Nevertheless, one has to admit that these more advanced schemes perform well for large scale applications and also for our small toy problem. We just report on some of our numerical experiments for sparsity constrained matrix equations using ISTA (iterated soft thresholding) and LISTA (learned ISTA). We start with defining the algorithms.

ISTA: Let $y \in \mathbb{R}^{d_2}$, $A \in \mathbb{R}^{d_2 \times d_1}$ be given, choose λ, α and set $x^0 = 0$.
For k=1,... do

$$x^k = \mathcal{S}_\alpha \left((I - \lambda A^T A) x^{k-1} \right)$$

until *stopping criterion*.

Here, $\mathcal{S}_\alpha(x)$ is defined componentwise by

$$\mathcal{S}_\alpha(x)_j = sign(x_j) \max\{|x_j| - \alpha, 0\}.$$

One can reformulate the iteration step as

$$x^k = \mathcal{S}_\alpha \left((I - \lambda A^T A) x^{k-1} + \lambda A^T A y \right) = \mathcal{S}_\alpha \left(W x^{k-1} + B y \right), \tag{11}$$

where $B = \lambda A^T \in \mathbb{R}^{d_1 \times d_2}$ and $W = I - BA \in \mathbb{R}^{d_1 \times d_1}$.

This is exactly the structure of the computations performed by a neural network $\Phi_{W,B}$ with activation function $\mathcal{S}_\alpha$. Hence, one can determine two matrices W, B by training a fully connected feed forward neural network with K internal layers by using the loss function L_2 for the inverse problem as above. Applying the trained network on an input y^δ one expects $\Phi_{W,B}(y^\delta) \sim x$.

LISTA: Let $\Phi_{W,B}$ denote a fully connected feed forward network with K internal layers with d_1 nodes in each layer. The linear maps as in (11) are optimized during training , $\mathcal{S}_\alpha$ is kept fixed as as activation function and the identity is used as output layer. Let $(y^{(i)} \in I\!R^{d_2}, x^{(i)})_{i=1,..,n}, y^{(i)}, x^{(i)} \in I\!R^d_1$ denote a set of training data and determine

$$(W^*, B^*) = argmin_{W,B} \sum_{i=1,n} \|\Phi_{W,B}(y^{(i)} - x^{(i)}\|^2 \ .$$

After training we use a separate set of test data for evaluation by the same procedure as above. ISTA is initialized with $\alpha = \sigma$ and $\lambda = 0.01$, the stopping criterion is $x^k - x^{k-1} \leq 10^{-6}$. Even for the small 2×2 example, this leads to 100 iterations or more before convergence. For LISTA we set $K = 10$, i.e. it optimizes 10 iterations. Using more internal layers does not improve performance. Using uniformly sampled input data, i.e. the data are drawn uniformly form a ball around 0, the performance of LISTA is stable but does not improve, when compared with ISTA.

	ISTA	LISTA
MSE	0.6672	1.6274

Table 5: Comparison of error rates for ISTA and LISTA

This comparison is somewhat unfair, as it compares a fully converged ISTA with many iterations with a partially converged LISTA mimicking K iterations. Other papers, see [13], compare LISTA with K internal layers with ISTA with K steps, which - at least for K not too large - shows an advantage for LISTA.
In order to exploit the potential of neural networks for discovering prior distributions of the training data, we have changed the setup of the experiment. We did choose 10-dimensional input and output vectors and a singular matrix A. Moreover, the test data were drawn from a 2- or 3− dimensional linear subspace. In this case, ISTA performs as before, which is not surprising. ISTA is build just by using A and does not incorporate any knowledge about the input data. LISTA however performs much better, as it seems to discover the underlying low dimensional structure.
Let $A = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1+\epsilon \end{pmatrix}$ and $\epsilon = 1$. Let $x \in [0,1]^{10}$ be sparse. The feedforward neural network is trained for 50 epochs and has $K = 15$ layers. We observe the following error rates.

Sparsity /MSE	ISTA	LISTA
70% (unstructured)	0.6578	0.6642
70% (structured)	0.5439	0.0786
80% (unstructured)	0.4471	0.4778
80% (structured)	0.3083	0.0196

Table 6: Comparison of error rates for ISTA and LISTA with low dimensional input data

Acknowledgement

The author acknowledges the final support provided by the Deutsche Forschungsgemeinschaft (DFG) under grant GRK 2224/1 "Pi3: Parameter Identification - Analysis, Algorithms, Applications". The Numerical examples were done by Hannes Albers and Alexander Denker, who in particular designed the experiment with sparse input data. The statistical analysis was supported by Max Westphal. Furthermore, the author wants to thank Carola Schönlieb for her hospitality, part of the paper was written during the authors sabbatical in Cambridge. Finally, the author wants to thank a reviewer for careful reading and suggesting several improvements.

References

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[2] Jose Bioucas-Dias and Mário Figueiredo. A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. 16:2992–3004, 01 2008.

[3] Helmut Bölcskei, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. Optimal approximation with sparsely connected deep neural networks. *CoRR*, abs/1705.01714, 2017.

[4] Thomas Bonesky, Kristian Bredies, Dirk A Lorenz, and Peter Maass. A generalized conditional gradient method for nonlinear operator equations with sparsity constraints. *Inverse Problems*, 23(5):2041, 2007.

[5] Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, Aug 2012.

[6] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017.

[7] Cao Van Chung, J C De los Reyes, and C B Schoenlieb. Learning optimal spatially-dependent regularization parameters in total variation image denoising. *Inverse Problems*, 33(7):074005, 2017.

[8] David Colton, Heinz Engl, Alfred K. Louis, Joyce McLaughlin, and William Rundell. *Surveys on solution methods for inverse problems*. Springer, 2000. http://www.deeplearningbook.org.

[9] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[10] A. Edelman, B. D. Sutton, and Y. Wang. *Random matrix theory, numerical computation and applications*.

[11] Alan Edelman and N. Raj Rao. Random matrix theory. *Acta Numerica*, 14:233?297, 2005.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 399–406, USA, 2010. Omnipress.

[14] A. Hauptmann, F. Lucka, M. Betcke, N. Huynh, J. Adler, B. Cox, P. Beard, S. Ourselin, and S. Arridge. Model based learning for accelerated, limited-view 3D photoacoustic tomography. *IEEE Transactions on Medical Imaging*, 2018. In Press.

[15] B. Jin and P. Maass. Sparsity regularization for parameter identification problems. *Inverse Problems*, 28(12):123001, December 2012.

[16] Jaari Kaipio and Eerki Somersalo. *Statistical and computational inverse problems*. Springer, 2005.

[17] Rafal Latala. Some estimates of norms of random matrices. *Proceedings of the American Mathematical Society*, 133(5):1273–1282, 2005.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] Stéphane Mallat. Understanding deep convolutional networks. *CoRR*, abs/1601.04920, 2016.

[20] James Martens and Ilya Sutskever. *Training Deep and Recurrent Networks with Hessian-Free Optimization*, pages 479–535. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[21] Jennifer L. Mueller and Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. SIAM, 2012.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[23] M. Unser. A representer theorem for deep neural networks. *ArXiv e-prints*, February 2018.

[24] R. van Handel. On the spectral norm of Gaussian random matrices. *ArXiv e-prints*, February 2015.