Notes for Neumann Networks for Linear Inverse Problems in Imaging

Yao Zhang https://zhims.github.io/

The general model for liner inverse problem is

$$y = X\beta^* + \varepsilon \tag{1.1}$$

where $y, \epsilon \in \mathbb{R}^m, \beta \in \mathbb{R}^p$ and$X \in \mathbb{R}^{m\times p}$.

To solve Eq 1.1,

$$\beta^* = \arg\min_{\beta} \ \frac{1}{2}\left\|y - X\beta\right\|_2^2 + \ r\left(\beta\right) \tag{1.2}$$

where $r\left(\beta\right)$ is a regular term from the data. (regular term is also called prior knowledge.)

In this paper [1], Eq 1.1 is equivalent to a quadratic form

$$X^T y = X^T X\beta + X^T\varepsilon \tag{1.3}$$

In [1], assumes that

$$r\left(\beta\right) = \frac{1}{2}\beta^T R\beta \tag{1.4}$$

where $R \in \mathbb{R}^{p\times p}$, then $\nabla r\left(\beta\right) = R\beta$.

Then by the Fermat lemma, the condition for the optional solution for 1.2 is

$$\left(X^T X + R\right)\beta^* = X^T y \tag{1.5}$$

then

$$\beta^* = \left(X^T X + R\right)^{-1} X^T y \tag{1.6}$$

where $\left(X^T X + R\right)$ is invertible.

To solve Eq 1.6, [1] considers a Neumann series expansion of linear operators as below

$$(I-A)^{-1}=\sum_{k=0}^{\infty} A^k = I + A + A^2 + \cdots A^k + \cdots \tag{1.7}$$

where $I, A \in \mathbb{R}^{p\times p}$, here we leave out the conditions for Neumann series expansion in details.

(Just recall that $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots \; where \; |x| < 1$, Eq 1.7 is well-defined if you want to check by using SVD of A.)

Now, let

$$I - A = \eta B \tag{1.8}$$

then $B = \frac{I-A}{\eta}$, so we can get that

$$B^{-1} = \eta \sum_{k=0}^{\infty} (I - \eta B)^k \tag{1.9}$$

Finally,

$$\beta^* = \sum_{j=0}^{\infty} \left(I - \eta X^T X - \eta R\right)^j \left(\eta X^T y\right) \tag{1.10}$$

Then, [1] considers designing the special type of neural networks which called Neumann Network ro solve Eq 1.10.

It is is just a technical report I mentioned above.

Week 48

Truncating the series in 1.10 to $B+1$, and replacing multiplication by matrix R with a general mapping R: $\mathbb{R}^p \to \mathbb{R}^p$, motivates an estimator $\widehat{\beta}$ of the form

$$\widehat{\beta}\,(y) \triangleq \sum_{j=0}^{B} \left[\left(I - \eta X^T X\right)(\,\cdot\,) - \eta R\,(\,\cdot\,)\right]^j \left(\eta X^T y\right) \tag{1.11}$$

Remark 1.1. *In Eq 1.11, $(\,\cdot\,)$ stands for parameters which will be learned via the neural networks.*

We turn 1.11 into a trainable estimator by $R = R_\theta, \eta = \eta_{\theta_1}$ be a trainable mapping depending on a vector of parameters $\theta \in \mathbb{R}^q, \theta_1 \in \mathbb{R}^{q_1}$ to be learned from the training data. In other words,

$$\widehat{\beta}\,(y) = \widehat{\beta}\,(y, \theta, \eta) \tag{1.12}$$

Remark 1.2. *The trainable network $R = R_\theta$ which we call Neumann Network.*

To see how 1.11 can be formulated as a network, observe that the terms in 1.11 have the following recursive form:

$$\begin{aligned}
&\widetilde{\beta}^0 = \eta X^T y \\
&\widetilde{\beta}^1 = \left(I - \eta X^T X\right) \widetilde{\beta}^0 - \eta R\left(\widetilde{\beta}^0\right) = \left(I - \eta X^T X\right) \eta X^T y - \eta R\left(\widetilde{\beta}^0\right) \\
&\widetilde{\beta}^2 = \left(I - \eta X^T X\right) \widetilde{\beta}^1 - \eta R\left(\widetilde{\beta}^1\right) = \left(I - \eta X^T X\right)^2 \eta X^T y - \left(I - \eta X^T X\right) \eta R\left(\widetilde{\beta}^0\right) - \eta R\left(\widetilde{\beta}^1\right) \\
&\cdots = \cdots
\end{aligned} \tag{1.13}$$

1.13 is equal to

$$\widetilde{\beta}^j = \left(I - \eta X^T X\right) \widetilde{\beta}^{j-1} - \eta R\left(\widetilde{\beta}^{(j-1)}\right) \tag{1.14}$$

where $\widetilde{\beta}^0 = \eta X^T y$.

Finally, we want to

$$\widehat{\beta}\left(y\right) = \sum_{j=0}^{B} \widetilde{\beta}^j \tag{1.15}$$

The Newmann network is as the following:

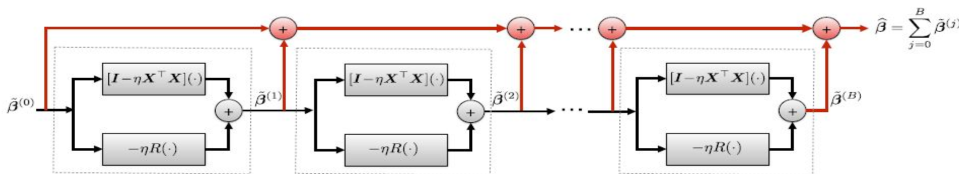


Figure 1: Neumann network. Here R is a trained neural network, and η is also trained.

It is a challenge task when $X^T X + R$ is ill-conditioned in 1.5. So, [1] derive a variant of Neumann networks inspired by a preconditioning of 1.5.

Starting from 1.5, for any $\lambda > 0$ we have

$$\left(X^T X + \lambda I\right) \beta^* + \left(R - \lambda I\right) \beta^* = X^T y \tag{1.16}$$

Denote that $T_\lambda = \left(X^T X + \lambda I\right)^{-1}$, then applying T_λ to both sides of 1.16 and rearranging terms gives

$$\left(I - \lambda T_\lambda + \widetilde{R}\right) \beta^* = T_\lambda X^T y \tag{1.17}$$

where $\widetilde{R} = T_\lambda R$.

Similar to 1.11, we can get that

$$\widetilde{\beta}_{pc}\,(y) = \sum_{j=0}^{B} \left(\lambda T_\lambda\,(\,\cdot\,) - \widetilde{R}\ \,(\,\cdot\,)\right)^j T_\lambda X^T y \tag{1.18}$$

The Peconditioned Newmann network is as the following:

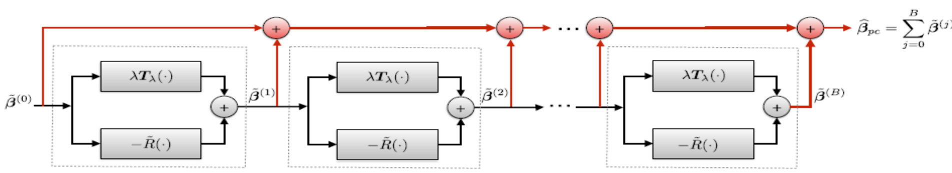
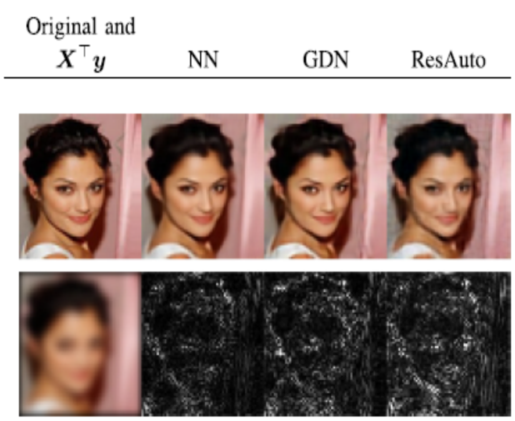


Figure 2: Peconditioned Neumann network. Here R is a trained neural network, η is trained, and λ is trained, respectively

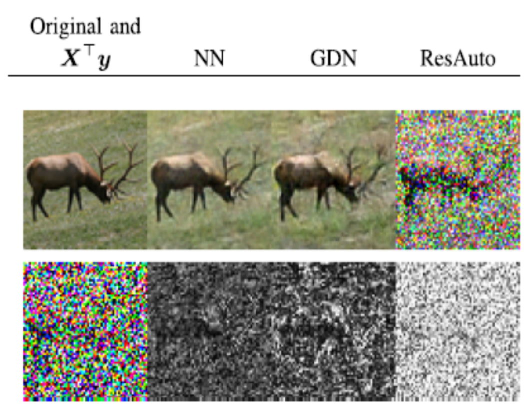
Applications

		Inpaint	Deblur	Deblur+ϵ	CS2	CS8	SR4	SR10
CIFAR10	NN	28.20	36.55	29.43	33.83	**25.15**	24.48	**23.09**
	PNN	28.40	**37.83**	**30.47**	33.75	23.43	**26.06**	21.79
	GDN	27.76	31.25	29.02	**34.99**	25.00	24.49	20.47
	MoDL	28.18	34.89	29.72	33.47	23.72	24.54	21.90
	TNRD	27.87	34.84	29.70	32.74	25.11	23.84	21.99
	ResAuto	**29.05**	31.04	25.24	18.51	9.29	24.84	21.92
	CSGM	17.88	15.20	14.61	17.99	19.33	16.87	16.66
	TV	25.90	27.57	26.64	25.41	20.68	24.71	20.68
CelebA	NN	**31.06**	31.01	30.43	**35.12**	**28.38**	27.31	23.57
	PNN	30.45	**33.79**	**30.89**	32.61	26.41	**28.70**	23.74
	GDN	30.99	30.19	29.27	34.93	28.33	27.14	23.46
	MoDL	30.75	30.80	29.59	30.22	25.84	26.42	24.12
	TNRD	30.21	29.92	29.79	33.89	28.19	25.75	22.73
	ResAuto	29.66	25.65	25.29	19.41	9.16	25.62	**24.92**
	CSGM	17.75	15.68	15.30	17.99	18.21	18.11	17.88
	TV	24.07	30.96	26.24	25.91	23.01	26.83	20.70
STL10	NN	27.47	29.43	26.12	**31.98**	**26.65**	24.88	21.80
	PNN	28.00	**30.66**	**27.21**	31.40	23.43	**25.95**	**22.19**
	GDN	**28.07**	30.19	25.61	31.11	26.19	24.88	21.46
	MoDL	28.03	29.42	26.06	27.29	23.16	24.67	16.88
	TNRD	27.88	29.33	26.32	31.05	25.38	24.55	21.21
	ResAuto	27.28	25.42	25.13	19.48	9.30	24.12	21.13
	CSGM	16.50	14.04	15.59	16.67	16.39	16.58	16.47
	TV	26.29	29.96	26.85	24.82	22.04	26.37	20.12

Figure 3: PSNR comparison for the CIFAR, CELEBA, and STL10 datasets with the tasks which including inpaint, deblur, deblurg & denoising, Compressed sensing Cs and super resolution.



(a) deblur + denoise



(b) 8 $\times$ CS

Figure 4: Examples.

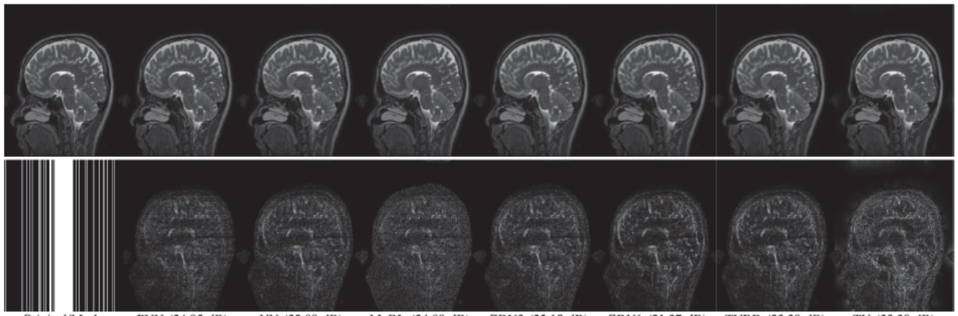


Figure 5: MRI reconstruction

References

[1] D. Gilton, G. Ongie and R. Willett, "Neumann Networks for Linear Inverse Problems in Imaging", *IEEE Transactions on Computational Imaging*, 2019, Vol. 6, pp. 328 –343.