Can neural networks always be trained?

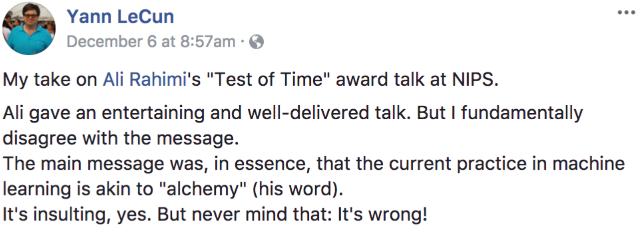
On the boundaries of deep learning

Matthew Colbrook
DAMTP, University of Cambridge

Collaborators: Anders Hansen, Vegard Antun, Kristian Haug

Is machine learning like alchemy?

Google's Ali Rahimi, winner of the Test-of-Time award 2017 (NIPS), "Machine learning has become alchemy. ... I would like to live in a society whose systems are built on top of verifiable, rigorous, thorough knowledge, and not on alchemy."



Yann LeCun
December 6 at 8:57am ·

My take on Ali Rahimi's "Test of Time" award talk at NIPS.

Ali gave an entertaining and well-delivered talk. But I fundamentally disagree with the message.
The main message was, in essence, that the current practice in machine learning is akin to "alchemy" (his word).
It's insulting, yes. But never mind that: It's wrong!

Is machine learning like alchemy?

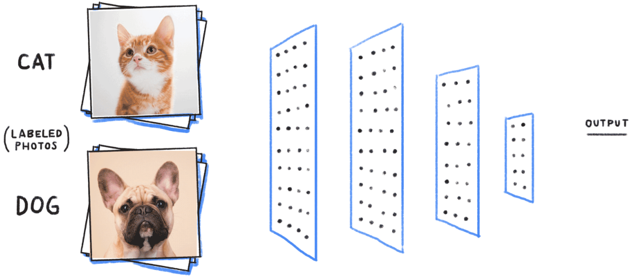
Outline of talk

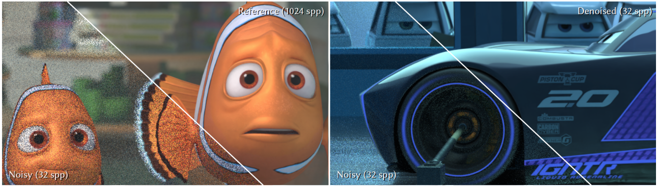
- Motivation I: Construction of neural networks.
- Motivation II: Stability of neural networks.
- Some precise notions.
- Theorem: Stable neural networks can (in some cases) be constructed.
- Numerical example.
- Conclusion.

Motivation I: Construction of neural networks.

What is the key problem in machine learning?

Learning a function.





TRAINING TEST

Setup

Image $x \in \mathbb{C}^N$, we are given access to measurements of the form

$$y = Ax + e,$$

where $A \in \mathbb{C}^{m \times N}$ represents sampling modality, $m \ll N$.

Setup

Image $x \in \mathbb{C}^N$, we are given access to measurements of the form

$$y = Ax + e,$$

where $A \in \mathbb{C}^{m \times N}$ represents sampling modality, $m \ll N$.

Task is to reconstruct x from the noisy measurements y.

Setup

Image $x \in \mathbb{C}^N$, we are given access to measurements of the form

$$y = Ax + e,$$

where $A \in \mathbb{C}^{m \times N}$ represents sampling modality, $m \ll N$.

Task is to reconstruct x from the noisy measurements y.

Without additional assumptions, such as sparsity of x, this problem is highly ill-posed.

Setup

Image $x \in \mathbb{C}^N$, we are given access to measurements of the form

$$y = Ax + e,$$

where $A \in \mathbb{C}^{m \times N}$ represents sampling modality, $m \ll N$.

Task is to reconstruct x from the noisy measurements y.

Without additional assumptions, such as sparsity of x, this problem is highly ill-posed.

Might try to solve via a solution of

$$\min_{z \in \mathbb{C}^N} \|z\|_1 \quad \text{s.t.} \quad \|Az - y\|_2 \leq \nu,$$

Setup

Image $x \in \mathbb{C}^N$, we are given access to measurements of the form

$$y = Ax + e,$$

where $A \in \mathbb{C}^{m \times N}$ represents sampling modality, $m \ll N$.

Task is to reconstruct x from the noisy measurements y.

Without additional assumptions, such as sparsity of x, this problem is highly ill-posed.

Might try to solve via a solution of

$$\min_{z \in \mathbb{C}^N} \|z\|_1 \quad \text{s.t.} \quad \|Az - y\|_2 \leq \nu,$$

or

$$\min_{z \in \mathbb{C}^N} \|z\|_1 + \|Az - y\|_2^2,$$

Setup

Image $x \in \mathbb{C}^N$, we are given access to measurements of the form

$$y = Ax + e,$$

where $A \in \mathbb{C}^{m \times N}$ represents sampling modality, $m \ll N$.

Task is to reconstruct x from the noisy measurements y.

Without additional assumptions, such as sparsity of x, this problem is highly ill-posed.

Might try to solve via a solution of

$$\min_{z \in \mathbb{C}^N} \|z\|_1 \quad \text{s.t.} \quad \|Az - y\|_2 \leq \nu,$$

or

$$\min_{z \in \mathbb{C}^N} \|z\|_1 + \|Az - y\|_2^2,$$

or etc.

Neural networks are FANTASTIC approximators!

Consider the following mapping $\varphi_{A,\nu} : \mathcal{M} \to \mathbb{R}^N$ where

$$\mathcal{M} = \{y_j\}_{j=1}^{r} \subset \mathbb{R}^m, \quad r < \infty, \, m < N$$

given by

$$\varphi_{A,\nu}(y) = w, \quad w \in \operatorname*{argmin}_z \|z\|_1 \text{ subject to } \|Az - y\|_2 \leq \nu.$$

Neural networks are FANTASTIC approximators!

Consider the following mapping $\varphi_{A,\nu} : \mathcal{M} \to \mathbb{R}^N$ where

$$\mathcal{M} = \{y_j\}_{j=1}^{r} \subset \mathbb{R}^m, \quad r < \infty, \, m < N$$

given by

$$\varphi_{A,\nu}(y) = w, \quad w \in \underset{z}{\operatorname{argmin}} \|z\|_1 \text{ subject to } \|Az - y\|_2 \leq \nu.$$

Theorem ([Pinkus, 1999])

Let $\nu, \delta \geq 0$. If the non-linear function ρ in each layer is not a polynomial, there exists a neural network Φ, depending on A and $\mathcal{M}$, such that

$$\|\Phi(y) - \varphi_{A,\nu}(y)\|_2 \leq \delta, \quad \forall y \in \mathcal{M}.$$

Neural networks are FANTASTIC approximators!

Consider the following mapping $\varphi_{A,\nu} : \mathcal{M} \to \mathbb{R}^N$ where

$$\mathcal{M} = \{y_j\}_{j=1}^r \subset \mathbb{R}^m, \quad r < \infty, \, m < N$$

given by

$$\varphi_{A,\nu}(y) = w, \quad w \in \operatorname*{argmin}_z \|z\|_1 \text{ subject to } \|Az - y\|_2 \leq \nu.$$

Theorem ([Pinkus, 1999])

Let $\nu, \delta \geq 0$. If the non-linear function ρ in each layer is not a polynomial, there exists a neural network Φ, depending on A and $\mathcal{M}$, such that

$$\|\Phi(y) - \varphi_{A,\nu}(y)\|_2 \leq \delta, \quad \forall y \in \mathcal{M}.$$

But: need a constructive training model.

Constructive?

In reality given approximations: $\{y_{j,n}\}_{j=1}^{r}$, $\{\phi_{j,n}\}_{j=1}^{r}$ and A_n such that:

$$\|y_{j,n} - y_j\|,\ \|\phi_{j,n} - \varphi_{A_n,\nu}(y_{j,n})\|,\ \|A_n - A\| \leq 2^{-n}.$$

Constructive?

In reality given approximations: $\{y_{j,n}\}_{j=1}^{r}$, $\{\phi_{j,n}\}_{j=1}^{r}$ and A_n such that:

$$\|y_{j,n} - y_j\|, \ \|\phi_{j,n} - \varphi_{A_n,\nu}(y_{j,n})\|, \ \|A_n - A\| \leq 2^{-n}.$$

This is what we can store on a computer in real life, models irrational A etc.

Constructive?

In reality given approximations: $\{y_{j,n}\}_{j=1}^r$, $\{\phi_{j,n}\}_{j=1}^r$ and A_n such that:

$$\|y_{j,n} - y_j\|,\ \|\phi_{j,n} - \varphi_{A_n,\nu}(y_{j,n})\|,\ \|A_n - A\| \leq 2^{-n}.$$

This is what we can store on a computer in real life, models irrational A etc.

Training set must be

$$\mathcal{T} := \{(y_{j,n}, \phi_{j,n}, A_n) \,|\, j = 1, \ldots, r, n \in \mathbb{N}\}.$$

Constructive?

In reality given approximations: $\{y_{j,n}\}_{j=1}^r$, $\{\phi_{j,n}\}_{j=1}^r$ and A_n such that:

$$\|y_{j,n} - y_j\|, \ \|\phi_{j,n} - \varphi_{A_n,\nu}(y_{j,n})\|, \ \|A_n - A\| \leq 2^{-n}.$$

This is what we can store on a computer in real life, models irrational A etc.

Training set must be

$$\mathcal{T} := \{(y_{j,n}, \phi_{j,n}, A_n) \,|\, j = 1, \ldots, r, n \in \mathbb{N}\}.$$

Can we train a neural network that can approximate Φ based on the training set $\mathcal{T}$?

Constructive?

In reality given approximations: $\{y_{j,n}\}_{j=1}^{r}$, $\{\phi_{j,n}\}_{j=1}^{r}$ and A_n such that:

$$\|y_{j,n} - y_j\|, \ \|\phi_{j,n} - \varphi_{A_n,\nu}(y_{j,n})\|, \ \|A_n - A\| \leq 2^{-n}.$$

This is what we can store on a computer in real life, models irrational A etc.

Training set must be

$$\mathcal{T} := \{(y_{j,n}, \phi_{j,n}, A_n) \,|\, j = 1, \ldots, r, n \in \mathbb{N}\}.$$

Can we train a neural network that can approximate Φ based on the training set $\mathcal{T}$?

Maybe we expect to be able to do this by unravelling standard (iterative) optimisation algorithms? Like ISTA, FISTA, NESTA,...

Constructive?

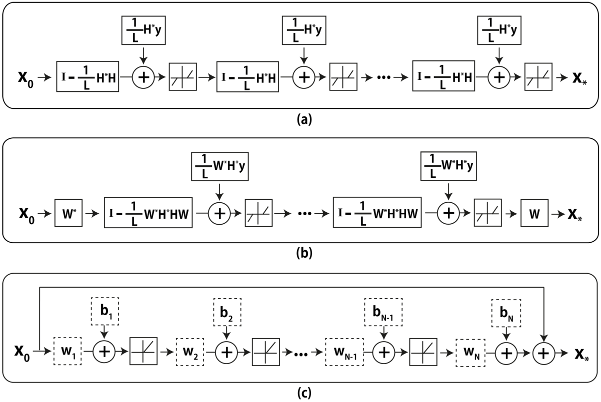


Fig. 1. Block diagrams for (a) unfolded version of iterative shrinkage method [31], (b) unfolded version of iterative shrinkage method with sparsifying transform (**W**) and (c) convolutional network with the residual framework. L is the Lipschitz constant, $\mathbf{x}_0$ is the initial estimates, b_i is the learned bias, w_i is the learned convolutional kernel. The broken line boxes in (c) indicate the variables to be learned.

Figure: Source: *Deep convolutional neural network for inverse problems in imaging* [Jin et al., 2017].

What could go wrong?

(i) There does not exist a neural network that approximates the function we are interested in.

What could go wrong?

(i) ~~There does not exist a neural network that approximates the function we are interested in.~~

What could go wrong?

(i) ~~There does not exist a neural network that approximates the function we are interested in.~~

(ii) There does exist a neural network that approximates the function, however, there does not exist an algorithm that can construct the neural network.

What could go wrong?

(i) ~~There does not exist a neural network that approximates the function we are interested in.~~

(ii) There does exist a neural network that approximates the function, however, there does not exist an algorithm that can construct the neural network.

(iii) There does exist a neural network that approximates the function, and an algorithm to construct it. However, the algorithm will need prohibitively many samples.

What could go wrong?

(i) ~~There does not exist a neural network that approximates the function we are interested in.~~

(ii) There does exist a neural network that approximates the function, however, there does not exist an algorithm that can construct the neural network.

(iii) There does exist a neural network that approximates the function, and an algorithm to construct it. However, the algorithm will need prohibitively many samples.

Both of these last two can happen!

Theorem (Impossible in general)

Let $K > 2, L \in \mathbb{N}$ and d be any metric on $\mathbb{R}^N$ where $N \geq 6$. Then there exists a well conditioned *class Ω of elements $(A, \mathcal{M})$, such that we have the following three conditions. Consider the neural network Φ from Theorem 1.*

(i) *There does not exist any algorithm taking elements from $\mathcal{T}$ as input and producing a neural network Ψ such that Ψ approximates Φ on $\mathcal{M}$ to K correct digits in the metric d for all $(A, \mathcal{M}) \in \Omega$.*

(ii) *There exists an algorithm taking elements from $\mathcal{T}$ as input that produces a neural network Ψ that approximates Φ on $\mathcal{M}$ to $K - 1$ correct digits in the metric d for all $(A, \mathcal{M}) \in \Omega$. However, any algorithm producing such a network will need arbitrary many samples of elements from $\mathcal{T}$, where accessing $(y_{j,n}, \phi_{j,n}, A_n)$ for one j and n counts as one sample.*

(iii) *There exists an algorithm using L samples from $\mathcal{T}$ as input that produces a neural network Ψ that approximates Φ on $\mathcal{M}$ to $K - 2$ correct digits in the metric d for all $(A, \mathcal{M}) \in \Omega$.*

Well conditioned

- Condition of a matrix $\mathrm{Cond}(A) = \|A\| \|A^{-1}\|$.

Well conditioned

- Condition of a matrix $\mathrm{Cond}(A) = \|A\|\|A^{-1}\|$.
- Condition of the mapping $\Psi : \Omega \subset \mathbb{C}^n \to \mathbb{C}^m$, linear or non-linear, is often given by

$$\mathrm{Cond}(\Psi) = \sup_{x\in\Omega} \lim_{\epsilon\to 0^+} \sup_{\substack{x+z\in\Omega \\ 0<\|z\|\leq\epsilon}} \frac{\mathrm{dist}(\Psi(x+z), \Psi(x))}{\|z\|},$$

where we allow for multivalued functions by defining $\mathrm{dist}(\Psi(x), \Psi(z)) = \min_{\tilde{x}\in\Psi(x), \tilde{z}\in\Psi(z)} \|\tilde{x} - \tilde{z}\|$.

Well conditioned

- If Ψ denotes the solution map to our problem (in this example basis pursuit) with domain Ω, we define

  $\rho(A, y) = \sup\{\delta \mid \|\tilde{A}\|, \|\tilde{y}\| \leq \delta \Rightarrow (A + \tilde{A}, y + \tilde{y}) \in \Omega \text{ are feasible}\},$

  and this yields the Feasibility Primal (FP) condition number

$$C_{\mathrm{FP}}(A, y) := \frac{\max(\|A\|, \|y\|)}{\rho(A, y)}.$$

Hence, it is not enough to use universal approximation. When we seek to construct neural networks via an algorithm, we are led to classification theory.

Hence, it is not enough to use universal approximation. When we seek to construct neural networks via an algorithm, we are led to classification theory.

It is NOT enough to just “unravel” your favourite algorithm.

Hence, it is not enough to use universal approximation. When we seek to construct neural networks via an algorithm, we are led to classification theory.

It is NOT enough to just “unravel” your favourite algorithm.

Question: Which functions can be approximated by a neural network that can be computed by an algorithm?

Motivation II: Stability of neural networks.

A growing problem

Most "state-of-the-art" neural networks are unstable. Now well-known for image classification:

A growing problem

Most “state-of-the-art” neural networks are unstable. Now well-known for image classification:

- Universal small perturbations
  [Moosavi-Dezfooli et al., 2017]

A growing problem

Most “state-of-the-art” neural networks are unstable. Now well-known for image classification:

- Universal small perturbations [Moosavi-Dezfooli et al., 2017]
- Across different networks [Szegedy et al., 2013]

A growing problem

Most "state-of-the-art" neural networks are unstable. Now well-known for image classification:

- Universal small perturbations [Moosavi-Dezfooli et al., 2017]
- Across different networks [Szegedy et al., 2013]
- Unrecognisable images confidently classified [Nguyen et al., 2015]

A growing problem

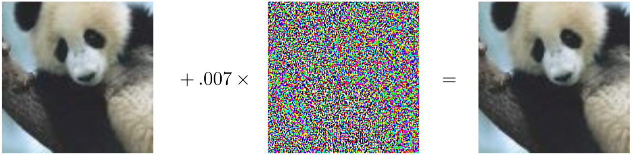


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Figure: Source: *Explaining and harnessing adversarial examples* [Goodfellow et al., 2014].

A growing problem

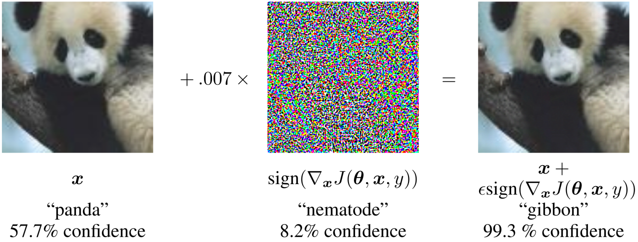


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Figure: Source: *Explaining and harnessing adversarial examples* [Goodfellow et al., 2014].

BUT can also happen with image denoising/reconstruction...

A stability test

Consider a neural network $\phi : \mathbb{C}^m \to \mathbb{C}^N$ which aims to reconstruct the image $\phi(y) \approx x$ from the (noisy) measurements $y = Ax + e$.

A stability test

Consider a neural network $\phi : \mathbb{C}^m \to \mathbb{C}^N$ which aims to reconstruct the image $\phi(y) \approx x$ from the (noisy) measurements $y = Ax + e$.

Algorithm seeks a vector $r \in \mathbb{R}^N$ such that

$$\|\phi(y + Ar) - \phi(y)\|_2 \text{ is large, while } \|r\|_2 \text{ is small.}$$

A stability test

Consider a neural network $\phi : \mathbb{C}^m \to \mathbb{C}^N$ which aims to reconstruct the image $\phi(y) \approx x$ from the (noisy) measurements $y = Ax + e$.

Algorithm seeks a vector $r \in \mathbb{R}^N$ such that

$$\|\phi(y + Ar) - \phi(y)\|_2 \text{ is large, while } \|r\|_2 \text{ is small.}$$

Consider the optimisation problem

$$r^*(y) \in \underset{r}{\operatorname{argmax}} \frac{1}{2}\|\phi(y + Ar) - x\|_2^2 - \frac{\lambda}{2}\|r\|_2^2.$$

A stability test

Consider a neural network $\phi : \mathbb{C}^m \to \mathbb{C}^N$ which aims to reconstruct the image $\phi(y) \approx x$ from the (noisy) measurements $y = Ax + e$.

Algorithm seeks a vector $r \in \mathbb{R}^N$ such that

$$\|\phi(y + Ar) - \phi(y)\|_2 \text{ is large, while } \|r\|_2 \text{ is small.}$$

Consider the optimisation problem

$$r^*(y) \in \operatorname*{argmax}_r \frac{1}{2}\|\phi(y + Ar) - x\|_2^2 - \frac{\lambda}{2}\|r\|_2^2.$$

Test aims to locate local maxima by using a gradient ascent with momentum on

$$Q_y^{\phi}(r) = \frac{1}{2}\|\phi(y + Ar) - x\|_2^2 - \frac{\lambda}{2}\|r\|_2^2$$

Example

Simple example for the AUTOMAP network, reported in *Nature* as a “state-of-the-art” network:

“Furthermore, AUTOMAP reconstructions exhibit superior noise immunity compared to those from conventional methods, as quantified by image signal-to-noise ratio and root-mean-squared error (RMSE) metrics.”

Example

Simple example for the AUTOMAP network, reported in *Nature* as a “state-of-the-art” network:

“Furthermore, AUTOMAP reconstructions exhibit superior noise immunity compared to those from conventional methods, as quantified by image signal-to-noise ratio and root-mean-squared error (RMSE) metrics.”

Do we believe this?

Example

Simple example for the AUTOMAP network, reported in *Nature* as a “state-of-the-art” network:

Not so state-of-the-art in terms of stability...

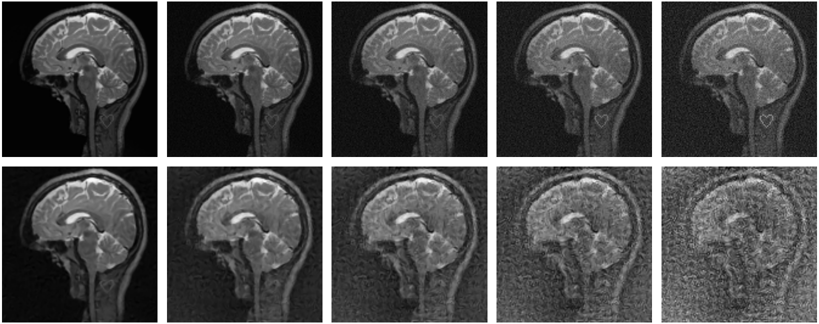


Figure: Stability test for AUTOMAP taken from [Antun et al., 2019], and where A is a subsampled Fourier transform. Top row: original image with perturbations. Bottom row: reconstructions using AUTOMAP.

Hence, I would not want my doctor to test for cancer using neural networks (at least not yet) - we need stability guarantees.

Hence, I would not want my doctor to test for cancer using neural networks (at least not yet) - we need stability guarantees.

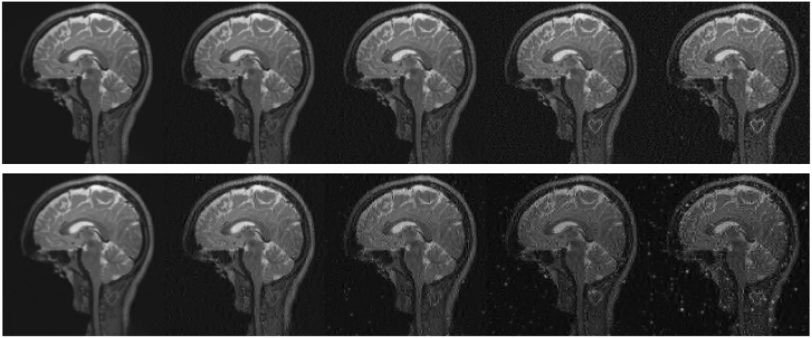
Question: Can stability be hard-wired into the networks?

Hence, I would not want my doctor to test for cancer using neural networks (at least not yet) - we need stability guarantees.

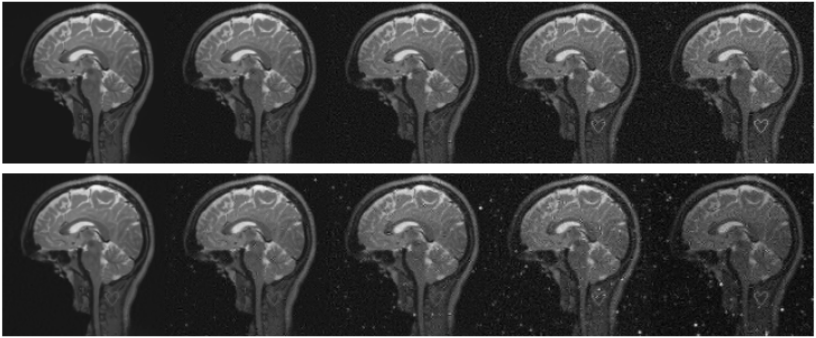
Question: Can stability be hard-wired into the networks?

This is subtle, preliminary results (received last night) suggest that this can't be done with FISTA on LASSO or Chambolle-Pock on basis pursuit...

Stability test on FISTA



Stability test on Chambolle-Pock



Some precise notions.

What is a neural network?

What is a neural network?

$\phi : \mathbb{C}^m \to \mathbb{C}^N$ such that

$$\phi(y) = W_L(\rho_{L-1}(...\rho_1(W_1(y)))).$$

What is a neural network?

$\phi : \mathbb{C}^m \to \mathbb{C}^N$ such that

$$\phi(y) = W_L(\rho_{L-1}(...\rho_1(W_1(y)))).$$

- Affine map $W_j(z) = A_j z + b_j(y)$ where $A_j \in \mathbb{C}^{N_j \times N_{j-1}}$ and $b_j(y) = B_j y + c_j \in \mathbb{C}^{N_j}$ (affine function of input y).

What is a neural network?

$\phi : \mathbb{C}^m \to \mathbb{C}^N$ such that

$$\phi(y) = W_L(\rho_{L-1}(...\rho_1(W_1(y)))).$$

- Affine map $W_j(z) = A_j z + b_j(y)$ where $A_j \in \mathbb{C}^{N_j \times N_{j-1}}$ and $b_j(y) = B_j y + c_j \in \mathbb{C}^{N_j}$ (affine function of input y).
- Each ρ_j is a non-linear function and is one of two forms:

What is a neural network?

$\phi : \mathbb{C}^m \to \mathbb{C}^N$ such that

$$\phi(y) = W_L(\rho_{L-1}(...\rho_1(W_1(y)))).$$

- Affine map $W_j(z) = A_j z + b_j(y)$ where $A_j \in \mathbb{C}^{N_j \times N_{j-1}}$ and $b_j(y) = B_j y + c_j \in \mathbb{C}^{N_j}$ (affine function of input y).
- Each ρ_j is a non-linear function and is one of two forms:
  1. Index set $I_j \subset \{1, ..., N_j\}$ such that ρ_j applies a non-linear function f_j element-wise on the input vector's components with indices in I_j.

What is a neural network?

$\phi : \mathbb{C}^m \to \mathbb{C}^N$ such that

$$\phi(y) = W_L(\rho_{L-1}(...\rho_1(W_1(y)))).$$

- Affine map $W_j(z) = A_j z + b_j(y)$ where $A_j \in \mathbb{C}^{N_j \times N_{j-1}}$ and $b_j(y) = B_j y + c_j \in \mathbb{C}^{N_j}$ (affine function of input y).
- Each ρ_j is a non-linear function and is one of two forms:
  1. Index set $I_j \subset \{1, ..., N_j\}$ such that ρ_j applies a non-linear function f_j element-wise on the input vector's components with indices in I_j.
  2. Non-linear function f_j such that, after decomposing the input vector x as $(x_0, X, Y)^T$ for scalar x_0 and $X \in \mathbb{C}^{m_j}$, we have
$$\rho_j : \begin{pmatrix} x_0 \\ X \\ Y \end{pmatrix} \to \begin{pmatrix} 0 \\ f_j(x_0)X \\ Y \end{pmatrix}.$$

What do we mean by computable?

What do we mean by computable?

- Affine maps (matrix and bias) have rational entries - we can store this on our computer (in practice use floats).

What do we mean by computable?

- Affine maps (matrix and bias) have rational entries - we can store this on our computer (in practice use floats).
- Non-linear map constructed using sqrt_θ with

$$|\mathrm{sqrt}_\theta(x) - \sqrt{x}| \leq \theta, \text{for all } x \in \mathbb{R}_{\geq 0}$$

We can access this for rational input. Just one simple example, other choices possible...

What do we mean by computable?

- Affine maps (matrix and bias) have rational entries - we can store this on our computer (in practice use floats).
- Non-linear map constructed using sqrt_θ with

$$|\mathrm{sqrt}_\theta(x) - \sqrt{x}| \leq \theta, \text{for all } x \in \mathbb{R}_{\geq 0}$$

  We can access this for rational input. Just one simple example, other choices possible...
- Architecture (including affine maps etc.) constructed explicitly from the A_n's.

What do we mean by computable?

- Affine maps (matrix and bias) have rational entries - we can store this on our computer (in practice use floats).
- Non-linear map constructed using sqrt_θ with

$$|\mathrm{sqrt}_\theta(x) - \sqrt{x}| \leq \theta, \text{for all } x \in \mathbb{R}_{\geq 0}$$

  We can access this for rational input. Just one simple example, other choices possible...
- Architecture (including affine maps etc.) constructed explicitly from the A_n's.

In other words, we can build these in real life.

What do we mean by computable?

- Affine maps (matrix and bias) have rational entries - we can store this on our computer (in practice use floats).
- Non-linear map constructed using sqrt_θ with

  $$|\mathrm{sqrt}_\theta(x) - \sqrt{x}| \leq \theta, \text{for all } x \in \mathbb{R}_{\geq 0}$$

  We can access this for rational input. Just one simple example, other choices possible...
- Architecture (including affine maps etc.) constructed explicitly from the A_n's.

In other words, we can build these in real life.

Note: choice of non-linear function above makes proof of following theorem much easier. Above impossibility result works for any choice.

What do we mean by stable?

What do we mean by stable?

Definition (Framework for stability)

Let $\Phi = \{\phi_n, \theta_n\}_{n\in\mathbb{N}}$ *be a computable sequence of neural networks such that each* ϕ_n *has* l_n *layers and* $l_n \to \infty$*. Given* $\epsilon \geq 0$*,* $\gamma > 0$*, and a subset* $S \subset \mathbb{C}^N$*, we say that* Φ *is* ***stably*** $(\epsilon, \gamma)-$***accurate*** *over* S *if the following holds:*

1. *(Linear growth in depth) There exists a constant* $C > 1$ *independent of* A *such that* $C^{-1}n \leq l_n \leq Cn$ *and* $N_{j,n} \leq CN$*.*
2. *(Algebraic rate of accuracy of execution) There exists a polynomial* P_1 *independent of* A *and a constant* C_1 *(possibly dependent on* A*) such that* $\theta_n^{-1} \leq C_1(A)P_1(n)$*.*
3. *(Stable recovery to error* ϵ*) There exists constants* C_2, C_3 *(possibly dependent on* A, x*) such that for any* $x \in S$

$$\boxed{\|\phi_n(y) - x\|_2 \leq \epsilon + \frac{C_2(A,x)}{n^\gamma} + C_3(A,x)\|Ax - y\|_2.}$$

Goal: Explicitly construct stably $(\epsilon, \gamma)-$accurate networks for typical problems with ϵ small and γ as large as possible.

Goal: Explicitly construct stably $(\epsilon, \gamma)-$accurate networks for typical problems with ϵ small and γ as large as possible.

$P_\Omega \colon \mathbb{C}^N \to \mathbb{C}^m$ projection onto canonical basis e_j indexed by Ω.

$$A = P_\Omega U W^{-1}$$

where W is sparsifying transform and U measurement matrix.

Goal: Explicitly construct stably $(\epsilon, \gamma)-$accurate networks for typical problems with ϵ small and γ as large as possible.

$P_\Omega \colon \mathbb{C}^N \to \mathbb{C}^m$ projection onto canonical basis e_j indexed by Ω.

$$A = P_\Omega U W^{-1}$$

where W is sparsifying transform and U measurement matrix.

Some ideas from compressed sensing

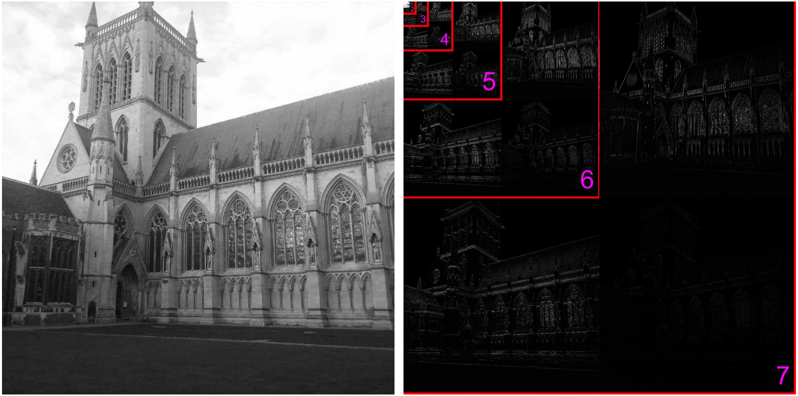


Figure: An image and its wavelet coefficients, where a brighter colour corresponds to a larger value.

Idea: Fully sample rows that correspond to the coarser wavelet levels and subsample the rows that correspond to the finer wavelet levels.

Definition (Sparsity in levels)

For $r \in \mathbb{N}$, *let* $\mathbf{M} = (M_1, ..., M_r)$, *where* $1 \leq M_1 < ... < M_r = N$, *and* $\mathbf{s} = (s_1, ..., s_r)$, *where* $s_k \leq M_k - M_{k-1}$ *for* $k = 1, ..., r$ *and* $M_0 = 0$. *A vector* $x \in \mathbb{C}^N$ *is* $(\mathbf{s}, \mathbf{M})$*-sparse in levels if*

$$|\text{supp}(x) \cap \{M_{k-1} + 1, ..., M_k\}| \leq s_k, \quad k = 1, ..., r.$$

We denote the set of $(\mathbf{s}, \mathbf{M})$*-sparse vectors by* $\Sigma_{\mathbf{s},\mathbf{M}}$.

$$\|x\|_{l^1_w} = \sum_{i=1}^{N} w_i |x_i|,$$

$$\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} = \inf\{\|x - z\|_{l^1_w} : z \in \Sigma_{\mathbf{s},\mathbf{M}}\}.$$

In practice, expect $\sigma_{\mathbf{s},\mathbf{M}}(Wx)_{l^1_w}$ to be small if we use wavelet levels.

Definition (Multilevel random sampling)

Let $l \in \mathbb{N}, \mathbf{N} = (N_1, \ldots, N_l) \in \mathbb{N}^l$ *with* $1 \leq N_1 < \ldots < N_l$, $\mathbf{m} = (m_1, \ldots, m_l) \in \mathbb{N}^l$, *with* $m_k \leq N_k - N_{k-1}$, $k = 1, \ldots, l$, *and suppose that*

$$\Omega_k \subset \{N_{k-1} + 1, \ldots, N_k\}, \ |\Omega_k| = m_k, \quad k = 1, \ldots, l,$$

are chosen uniformly at random, where $N_0 = 0$. *We refer to the set* $\Omega = \Omega_{\mathbf{N},\mathbf{m}} = \Omega_1 \cup \ldots \cup \Omega_l$ *as an* $(\mathbf{N}, \mathbf{m})$*- multilevel sampling scheme.*

Positive Results

(sparsifying transform: Haar wavelets with matrix W, others possible)

Case 1: Fourier measurements

U corresponds to the d-dimensional discrete Fourier transform.

Case 1: Fourier measurements

U corresponds to the d-dimensional discrete Fourier transform.

We divide the different frequencies into dyadic bands B_k, where $B_1 = \{0, 1\}$ and for $k = 2, ..., r$

$$B_k = \{-2^{k-1} + 1, ..., -2^{k-2}\} \cup \{2^{k-2} + 1, ..., 2^{k-1}\}.$$

Case 1: Fourier measurements

U corresponds to the d-dimensional discrete Fourier transform.

We divide the different frequencies into dyadic bands B_k, where $B_1 = \{0, 1\}$ and for $k = 2, ..., r$

$$B_k = \{-2^{k-1} + 1, ..., -2^{k-2}\} \cup \{2^{k-2} + 1, ..., 2^{k-1}\}.$$

In d dimensions set

$$B_{\mathbf{k}}^{(d)} = B_{k_1} \times ... \times B_{k_d}, \quad \mathbf{k} = (k_1, ..., k_d) \in \mathbb{N}^d.$$

Multilevel random sampling with $(m_{\mathbf{k}=(k_1,...,k_d)})_{k_1,...,k_d=1}^{r}$, $|m_{\mathbf{k}}| \le \left| B_{\mathbf{k}}^{(d)} \right|$.

Some final quantities...

Assume that if $M_{j-1} + 1 \leq i \leq M_j$ then $w_i = w_{(j)}$ (i.e. constant in each level).

Some final quantities...

Assume that if $M_{j-1} + 1 \leq i \leq M_j$ then $w_i = w_{(j)}$ (i.e. constant in each level).

To simplify formulae (simply for the purpose of presentation), will choose $w_{(j)} = \sqrt{s/s_j}$, but this can be avoided.

Some final quantities...

Assume that if $M_{j-1} + 1 \leq i \leq M_j$ then $w_i = w_{(j)}$ (i.e. constant in each level).

To simplify formulae (simply for the purpose of presentation), will choose $w_{(j)} = \sqrt{s/s_j}$, but this can be avoided.

One horrible looking formula...

$$\mathcal{M}_{\mathcal{F}}(\mathbf{s}, \mathbf{k}) = \sum_{l=1}^{\|\mathbf{k}\|_\infty} s_l \prod_{i=1}^{d} 2^{-|k_i - l|} + \sum_{l=\|\mathbf{k}\|_\infty+1}^{r} s_l 2^{-2(l-\|\mathbf{k}\|_\infty)} \prod_{i=1}^{d} 2^{-|k_i - l|}.$$

Theorem (Stable Neural Networks Exist)

Let $\epsilon_{\mathbb{P}} \in (0,1)$, $r, d \in \mathbb{N}$, $N = 2^{r \cdot d}$ and $\mathbf{M} = (M_1, ..., M_r)$, $\mathbf{s} = (s_1, ..., s_r)$ describe $(\mathbf{s}, \mathbf{M})$-sparse vectors corresponding to the scales in a d-dimensional wavelet basis. Suppose

$$m_{\mathbf{k}} \gtrsim \mathcal{M}_{\mathcal{F}}(\mathbf{s}, \mathbf{k}) \cdot L,$$
$$L = d \cdot r^3 \cdot \log(m) \cdot \log^2(rs) + \log(\epsilon_{\mathbb{P}}^{-1}).$$

Then, for each $n \in \mathbb{N}$, there exists a computable neural network ϕ_n^A with $3n$ layers such that with probability at least $1 - \epsilon_{\mathbb{P}}$, the following uniform recovery guarantee holds. For any $x \in \mathbb{C}^N$ with $\|x\|_{l^2} \lesssim 1$ and any $y \in \mathbb{C}^m$,

$$\|\phi_n^A(y) - x\|_{l^2} \lesssim \frac{\sigma_{\mathbf{s},\mathbf{M}}(Wx)_{l^1_w}}{\sqrt{s\sqrt{r}}} + \frac{r^{\frac{1}{4}}\|A\|}{n} + r^{\frac{1}{4}}\|Ax - y\|_{l^2}.$$

Theorem (Stable Neural Networks Exist)

Let $\epsilon_{\mathbb{P}} \in (0,1)$, $r, d \in \mathbb{N}$, $N = 2^{r \cdot d}$ *and* $\mathbf{M} = (M_1, ..., M_r)$, $\mathbf{s} = (s_1, ..., s_r)$ *describe* $(\mathbf{s}, \mathbf{M})$*-sparse vectors corresponding to the scales in a d-dimensional wavelet basis. Suppose*

$$m_{\mathbf{k}} \gtrsim \mathcal{M}_{\mathcal{F}}(\mathbf{s}, \mathbf{k}) \cdot L,$$
$$L = d \cdot r^3 \cdot \log(m) \cdot \log^2(rs) + \log(\epsilon_{\mathbb{P}}^{-1}).$$

Then, for each $n \in \mathbb{N}$, *there exists a computable neural network* ϕ_n^A *with* $3n$ *layers such that with probability at least* $1 - \epsilon_{\mathbb{P}}$, *the following uniform recovery guarantee holds. For any* $x \in \mathbb{C}^N$ *with* $\|x\|_{l^2} \lesssim 1$ *and any* $y \in \mathbb{C}^m$,

$$\|\phi_n^A(y) - x\|_{l^2} \lesssim \frac{\sigma_{\mathbf{s},\mathbf{M}}(Wx)_{l_w^1}}{\sqrt{s\sqrt{r}}} + \frac{r^{\frac{1}{4}}\|A\|}{n} + r^{\frac{1}{4}}\|Ax - y\|_{l^2}.$$

Stably $(\epsilon, 1)$−accurate, $\epsilon = \dfrac{\sigma_{\mathbf{s},\mathbf{M}}(Wx)_{l_w^1}}{\sqrt{s\sqrt{r}}}$.

How to interpret?

- Up to log-factors, equivalent to oracle estimator (as $n \to \infty$).
- For sparse vectors and large n, neural networks are locally Lipschitz so stable.
- Number of samples required in each annular region

$$\sum_{\|\mathbf{k}\|=k} m_{\mathbf{k}} \gtrsim \left(s_k + \sum_{l=1}^{k-1} s_l 2^{-(k-l)} + \sum_{l=k+1}^{r} s_l 2^{-3(l-k)} \right) \cdot L.$$

  is (up to logarithmic factors) proportional to s_k + exponentially decaying terms.

Remarks

- Proof uses some state-of-the-art compressed sensing techniques + a carefully constructed optimisation problem + a careful solver of this optimisation problem + careful track of approximations errors +... (paper out soon)
- Care must be taken (especially given previous negative result) - not just a question of picking your favourite optimisation problem/solver. E.g. won't work with Chambolle-Pock with basis pursuit or FISTA with LASSO.
- As of last night we have some numerical evidence of this also!
- Don't know at the moment whether γ can be made larger. Would expect this given universal approximation theorem, but then might become unstable. For instance, accelerated solvers look numerically unstable (e.g. tried NESTA).

Case 2: Binary measurements

U corresponds to Walsh-Hadamard transform with tensor product basis.

Case 2: Binary measurements

U corresponds to Walsh-Hadamard transform with tensor product basis.

$$\mathcal{M}_{\mathcal{B}}(\mathbf{s}, \mathbf{k}) = s_{\|\mathbf{k}\|_\infty} \prod_{i=1}^{d} 2^{-|k_i - \|\mathbf{k}\|_\infty|}$$

Case 2: Binary measurements

U corresponds to Walsh-Hadamard transform with tensor product basis.

$$\mathcal{M}_{\mathcal{B}}(\mathbf{s}, \mathbf{k}) = s_{\|\mathbf{k}\|_\infty} \prod_{i=1}^{d} 2^{-|k_i - \|\mathbf{k}\|_\infty|}$$

Theorem then the same but now

$$\sum_{\|\mathbf{k}\|=k} m_{\mathbf{k}} \gtrsim 2^d d s_k L,$$

and there are no terms from the sparsity levels $s_l, l \neq k$.

Numerical Example

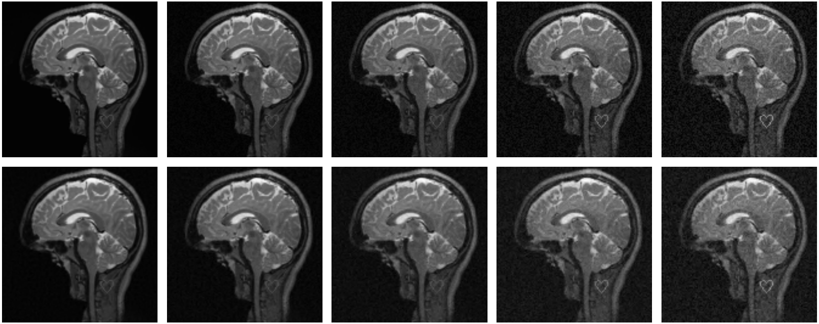


Figure: Stability test for new networks. Top row: original image with perturbations. Bottom row: reconstructions.

Numerical Example

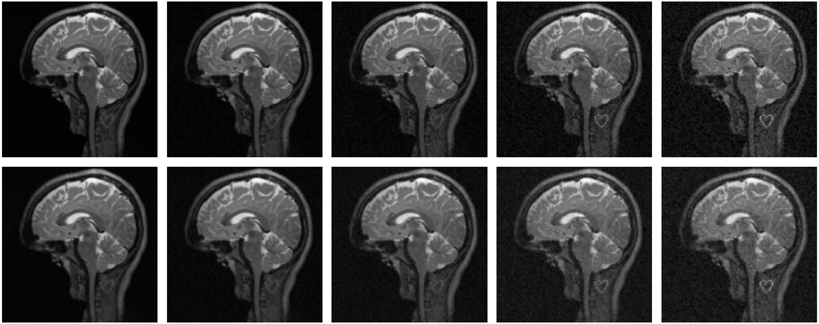


Figure: Stability test for new networks. Top row: original image with perturbations. Bottom row: reconstructions.

STABLE!

Conclusions

- The ridiculously impressive performance of neural networks may come at a high price in terms of stability. Given the last fifty years of the studying stability via inverse problems, this is an important issue that should **not be overlooked**.
- There is likely a rich classification theory, stating limits on the performance of stable methods - trade-off.
- One such example was presented with explicitly constructed stable neural networks.
- Next step: extensively assessing the performance of these new neural networks.
- Next step: applying these ideas to other compressed sensing type problems, e.g. continuous setting (BLASSO?)

Conclusions

- The <u>ridiculously</u> impressive performance of neural networks may come at a high price in terms of stability. Given the last fifty years of the studying stability via inverse problems, this is an important issue that should **not be overlooked**.
- There is likely a rich classification theory, stating limits on the performance of stable methods - trade-off.
- One such example was presented with explicitly constructed stable neural networks.
- Next step: extensively assessing the performance of these new neural networks.
- Next step: applying these ideas to other compressed sensing type problems, e.g. continuous setting (BLASSO?)

This talk was somewhat a test: please ask lots of questions (am very interested in feedback) and feel free to disagree - this issue is likely to be an ongoing debate in the community.

References

Antun, V., Renna, F., Poon, C., Adcock, B., and Hansen, A. C. (2019).
On instabilities of deep learning in image reconstruction - Does AI come at a cost?
Submitted.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014).
Explaining and harnessing adversarial examples.
arXiv preprint arXiv:1412.6572.

Jin, K. H., McCann, M. T., Froustey, E., and Unser, M. (2017).
Deep convolutional neural network for inverse problems in imaging.
IEEE Transactions on Image Processing, 26(9):4509–4522.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017).
Universal adversarial perturbations.
In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1765–1773.

Nguyen, A., Yosinski, J., and Clune, J. (2015).
Deep neural networks are easily fooled: High confidence predictions for unrecognizable images.
In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 427–436.

Pinkus, A. (1999).
Approximation theory of the mlp model in neural networks.
Acta numerica, 8:143–195.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013).
Intriguing properties of neural networks.
arXiv preprint arXiv:1312.6199.