Optimization Models

EECS 127 / EECS 227AT

Laurent El Ghaoui

EECS department
UC Berkeley

Spring 2020

LECTURE 26

Implicit Deep Learning

The Matrix is everywhere. It is all around us.

Morpheus

Outline

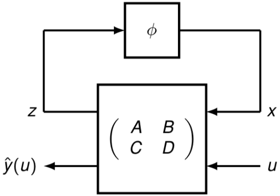
Collaborators

Joint work with:

- Armin Askari, Fangda Gu, Bert Travacca, Alicia Tsai (UC Berkeley);
- Mert Pilanci (Stanford);
- Emmanuel Vallod, Stefano Proto (`www.sumup.ai`).

Sponsors:

Implicit prediction rule



Equilibrium equation:

$$x = \phi(Ax + Bu)$$

Prediction:

$$\hat{y}(u) = Cx + Du$$

- Input $u \in \mathbb{R}^p$, predicted output $\hat{y}(u) \in \mathbb{R}^q$, hidden "state" vector $x \in \mathbb{R}^n$.
- Model parameter matrix:

$$M = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right).$$

- Activation: vector map $\phi : \mathbb{R}^n \to \mathbb{R}^n$, *e.g.* the ReLU: $\phi(\cdot) = \max(\cdot, 0)$ (acting componentwise on vectors).

Deep neural nets as implicit models

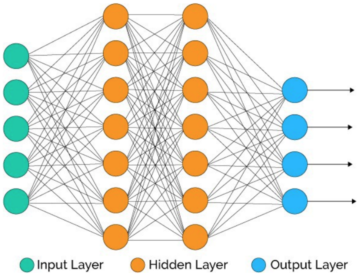


Figure: A neural network.

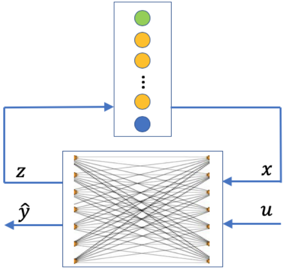


Figure: An implicit model.

Implicit models are more general: they allow loops in the network graph.

Example

Fully connected, feedforward neural network:

$$\hat{y}(u) = W_L x_L, \quad x_{l+1} = \phi_l(W_l x_l), \quad l = 1, \ldots, L-1, \quad x_0 = u.$$

Implicit model:

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cccc|c} 0 & W_{L-1} & \ldots & 0 & 0 \\ & 0 & \ddots & \vdots & \vdots \\ & & \ddots & W_1 & 0 \\ & & & 0 & W_0 \\ \hline W_L & 0 & \ldots & 0 & 0 \end{array}\right), \quad \begin{array}{c} x = \\ \begin{pmatrix} x_L \\ \vdots \\ x_1 \end{pmatrix} \end{array}, \quad \begin{array}{c} \phi(z) = \\ \begin{pmatrix} \phi_L(z_L) \\ \vdots \\ \phi_1(z_1) \end{pmatrix} \end{array}.$$

The equilibrium equation $x = \phi(Ax + Bu)$ is easily solved via backward substitution (forward pass).

Example: ResNet20

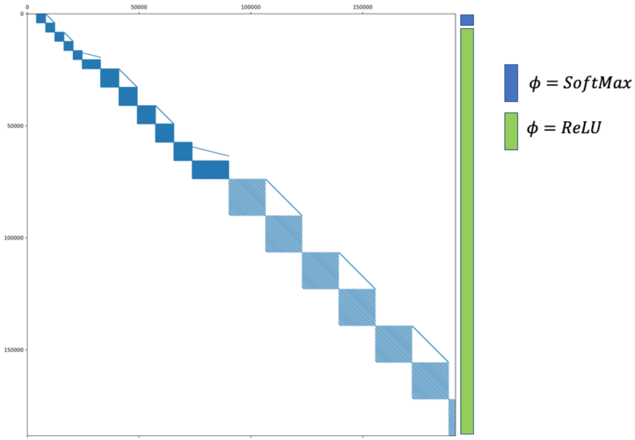


Figure: The A matrix for ResNet20.

- 20-layer network, implicit model of order $n \sim 180000$.
- Convolutional layers have blocks with Toeplitz structure.
- Residual connections appear as lines.

Neural networks as implicit models

Framework covers most neural network architectures:

- Neural nets have strictly upper triangular matrix A.
- Equilibrium equation solved by substitution, *i.e.* "forward pass".
- State vector x contains all the hidden features.
- Activation ϕ can be different for each component or blocks of x.
- Covers CNNs, RNNs, recurrent neural networks, (Bi-)LSTM, attention, transformers, etc.

Related concept: state-space models

The so-called "state-space" models for dynamical systems use the same idea to represent high-order differential equations . . .

Linear, time-invariant (LTI) dynamical system:

$$\dot{x} = Ax + Bu, \quad y = Cx + Du$$

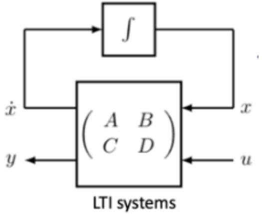


Figure: LTI system

Well-posedness

The matrix $A \in \mathbb{R}^{n \times n}$ is said to be well-posed for ϕ if, for every $b \in \mathbb{R}^n$, a solution $x \in \mathbb{R}^n$ to the equation

$$x = \phi(Ax + b),$$

exists, and it is unique.

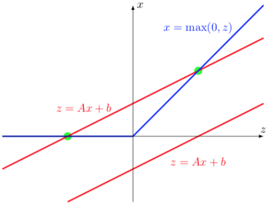


Figure: Equation has two or no solutions, depending on $\mathrm{sgn}(b)$.

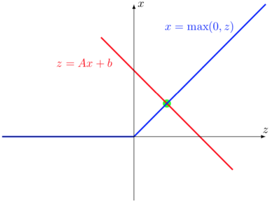


Figure: Solution is unique for every b.

Perron-Frobenius theory [1]

A square matrix P with non-negative entries admits a real eigenvalue λ with a non-negative eigenvector $v \neq 0$:

$$Pv = \lambda v.$$

The value λ dominates all the other eigenvalues: for any other (complex) eigenvalue $\mu \in \mathbf{C}$, we have $|\mu| \leq \lambda_{\mathrm{PF}}$.

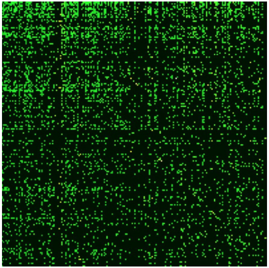


Figure: A web link matrix.

Google's Page rank search engine relies on computing the Perron-Frobenius eigenvector of the web link matrix.

PF Sufficient condition for well-posedness

Fact: Assume that ϕ is componentwise non-expansive (*e.g.*, $\phi =$ ReLU):

$$\forall\, u, v \in \mathbb{R}^n \;:\; |\phi(u) - \phi(v)| \leq |u - v|.$$

Then the matrix A is well-posed for ϕ if the non-negative matrix $|A|$ satisfies

$$\lambda_{pf}(|A|) < 1,$$

in which case the solution can be found via the fixed-point iterations:

$$x(t+1) = \phi(Ax(t) + b), \quad t = 0, 1, 2, \ldots$$

Covers neural networks: since then $|A|$ is strictly upper triangular, thus $\lambda_{pf}(|A|) = 0$.

Proof: existence

We have

$$|x(t+1) - x(t)| = |\phi(Ax(t)+b) - \phi(Ax(t-1)+b)| \leq |A||x(t) - x(t-1)|,$$

which implies that for every $t, h \geq 0$:

$$|x(t+\tau) - x(t)| \leq \sum_{k=t}^{t+\tau} |A|^k |x(1) - x(0)| \leq |A|^t \sum_{k=0}^{\tau} |A|^k |x(1) - x(0)| \leq |A|^t w,$$

where

$$w := \sum_{k=0}^{+\infty} |A|^k |x(1) - x(0)| = (I - |A|)^{-1} |x(1) - x(0)|,$$

since, due to $\lambda_{\mathrm{PF}}(|A|) < 1$, $I - |A|$ is invertible, and the series above converges.

Since $\lim_{t \to 0} |A|^t = 0$, we obtain that $x(t)$ is a Cauchy sequence, hence it has a limit point, x_∞. By continuity of ϕ we further obtain that $x_\infty = \phi(Ax_\infty + b)$, which establishes the existence of a solution.

Proof: unicity

To prove unicity, consider $x^1, x^2 \in \mathbb{R}^n_+$ two solutions to the equation. Using the hypotheses in the theorem, we have, for any $k \geq 1$:

$$|x^1 - x^2| \leq |A||x^1 - x^2| \leq |A|^k|x^1 - x^2|.$$

The fact that $|A|^k \to 0$ as $k \to +\infty$ then establishes unicity.

Norm condition

More conservative condition: $\|A\|_\infty < 1$, where

$$\lambda_{\mathrm{PF}}(|A|) \leq \|A\|_\infty := \max_i \sum_j |A_{ij}|.$$

Under previous PF conditions for well-posedness:

- we can always rescale the model so that $\|A\|_\infty < 1$, without altering the prediction rule;
- scaling related to PF eigenvector of $|A|$.

Hence during training we may simply use norm condition.

Composing implicit models

Cascade connection

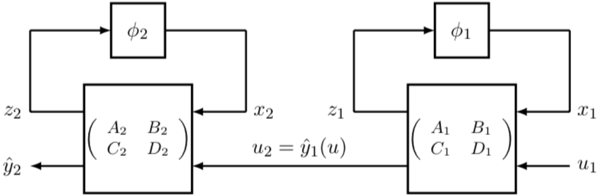


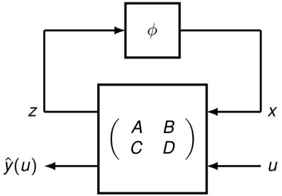
Figure: A cascade connection.

Class of implicit models closed under the following connections:

- Cascade
- Parallel and sum
- Multiplicative
- Feedback

Robustness analysis

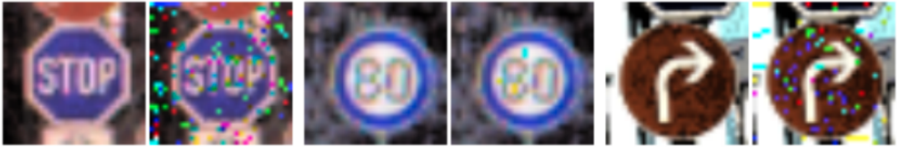
Goal: analyze the impact of input perturbations on the state and outputs.



Motivations:

- Diagnose a given (implicit) model.
- Generate adversarial attacks.
- Defense: modify the training problem so as to improve robustness properties.

Why does it matter?



stop	30m speed limit	80m speed limit	30m speed limit	go right	go straight

Changing a few carefully chosen pixels in a test image can cause a classifier to mis-categorize the image (Kwiatkowska *et al.*, 2019).

Robustness analysis

Input is unknown-but-bounded: $u \in \mathcal{U}$, with

$$\mathcal{U} := \left\{ u^0 + \delta \in \mathbb{R}^p \;:\; |\delta| \le \sigma_u \right\},$$

- $u^0 \in \mathbb{R}^n$ is a "nominal" input;
- $\sigma_u \in \mathbb{R}^n_+$ is a measure of componentwise uncertainty around it.

Assume (sufficient condition for) well-posedness:

- ϕ componentwise non-expansive;
- $\lambda_{\mathrm{PF}}(|A|) < 1$.

Nominal prediction:

$$x^0 = \phi(Ax^0 + Bu^0), \;\; \hat{y}(u^0) = Cx^0 + Du^0.$$

Component-wise bounds on the state and output

Fact: If $\lambda_{\mathrm{PF}}(|A|) < 1$, then $I - |A|$ is invertible, and

$$|\hat{y}(u) - \hat{y}(u^0)| \leq S|u - u^0|,$$

where

$$S := |C|(I - |A|)^{-1}|B| + |D|$$

is a "sensitivity matrix" of the implicit model.

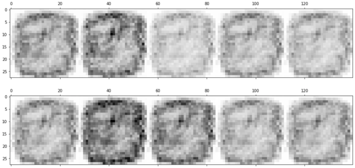
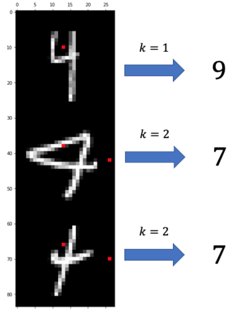


Figure: Sensitivity matrix of a classification network with 10 outputs (each image is a row).

Generate a sparse attack on a targeted output

Attack method:

- select the output to attack based on the rows (class) of sensitivity matrix;
- select top k entries in chosen row;
- randomly alter corresponding pixels.

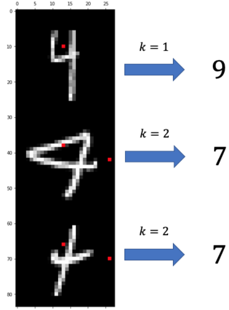


Changing $k = 1$ (top) $k = 2$ (mid, bot) pixels, images are wrongly classified, and accuracy decreases from 99% to 74%.

Generate a sparse attack on a targeted output

Attack method:

- select the output to attack based on the rows (class) of sensitivity matrix;
- select top k entries in chosen row;
- randomly alter corresponding pixels.



Changing $k = 1$ (top) $k = 2$ (mid, bot) pixels, images are wrongly classified, and accuracy decreases from 99% to 74%.

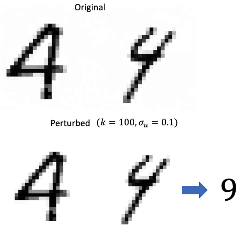
Generate a sparse bounded attack on a targeted output

Target a specific output with sparse attacks:

$$\mathcal{U} := \left\{ u^0 + \delta \in \mathbb{R}^p \ : \ |\delta| \leq \sigma_u, \ \ \textbf{Card}(\delta) \leq k \right\},$$

With $k \leq n$. Solve a linear program, with c related to chosen target:

$$\max_{x,\, u} \ c^\top x \ : \quad \begin{array}{l} x \geq Ax + Bu, \ \ x \geq 0, \ \ |x - x^0| \leq \sigma_x, \ \ |u - u^0| \leq \sigma_u \\ \|\mathrm{diag}\,(()\, \sigma_u)^{-1}(u - u^0)\|_1 \leq k. \end{array}$$



Changing $k = 100$ pixels by a tiny amount ($\sigma_u = 0.1$), targe images are wrongly classified a network with 99% nominal accuracy.

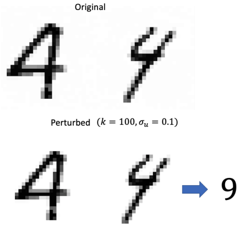
Generate a sparse bounded attack on a targeted output

Target a specific output with sparse attacks:

$$\mathcal{U} := \left\{ u^0 + \delta \in \mathbb{R}^p \ : \ |\delta| \le \sigma_u, \ \ \mathbf{Card}(\delta) \le k \right\},$$

With $k \le n$. Solve a linear program, with c related to chosen target:

$$\max_{x,\, u} \ c^\top x \ : \quad \begin{array}{l} x \ge Ax + Bu, \ \ x \ge 0, \ \ |x - x^0| \le \sigma_x, \ \ |u - u^0| \le \sigma_u \\ \|\mathrm{diag}\,(()\,\sigma_u)^{-1}(u - u^0)\|_1 \le k. \end{array}$$



Changing $k = 100$ pixels by a tiny amount ($\sigma_u = 0.1$), targe images are wrongly classified a network with 99% nominal accuracy.

Training problem

Setup

- Inputs: $U = [u_1, \ldots, u_m]$, with m data points $u_i \in \mathbb{R}^p$, $i \in [m]$.
- Outputs: $Y = [y_1, \ldots, y_m]$, with m responses $y_i \in \mathbb{R}^q$, $i \in [m]$.

Predictions: with $X = [x_1, \ldots, x_m] \in \mathbb{R}^{n \times m}$ the matrix of hidden feature vectors, and ϕ acting columnwise,

$$\hat{Y} = CX + DU, \quad X = \phi(AX + BU).$$

Training problem

Constrained problem

$$\begin{aligned}\min_{X,A,B,C,D} \quad & \mathcal{L}(Y,\hat{Y}) + \pi(A,B,C,D) \\ \text{s.t.} \quad & \hat{Y} = CX + DU, \;\; X = \phi(AX + BU), \;\; \|A\|_\infty \le \kappa.\end{aligned}$$

- Constraint on A with $\kappa < 1$ ensures well-posedness.
- $\pi(\cdot)$ is a (convex) penalty, *e.g.* one that encourages robustness:

$$\pi(A,B,C,D) \propto \frac{1}{2}\frac{\|B\|_\infty^2 + \|C\|_\infty^2}{1 - \|A\|_\infty} + \|D\|_\infty.$$

- May also incorporate penalties to encourage sparsity, low-rank, etc., *e.g.*:

$$\sum_{i\in[p]} \|Be_i\|_\infty$$

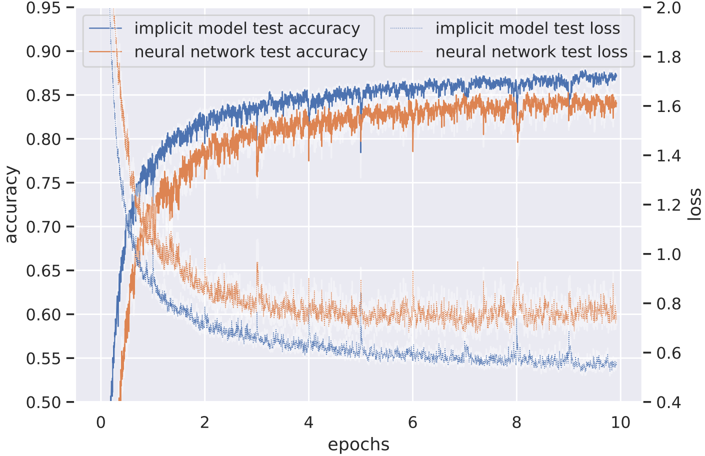
encourages entire columns of B to be zero, for feature selection.

Projected (sub) gradient

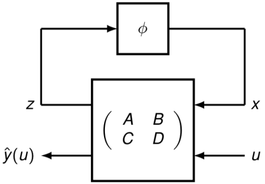
SGD can be adapted to the problem:

- Differentiating through the equilibrium equation is possible.
- Need to deal with the constraint of well-posedness via projection.
- Projection on constraint $\|A\|_\infty \leq \kappa$ can be done extremely fast using (vectorized) bisection, solving for each row of A in parallel.
- Can extend to Frank-Wolfe methods, which are suited to seeking sparse models.

Example: traffic sign data set

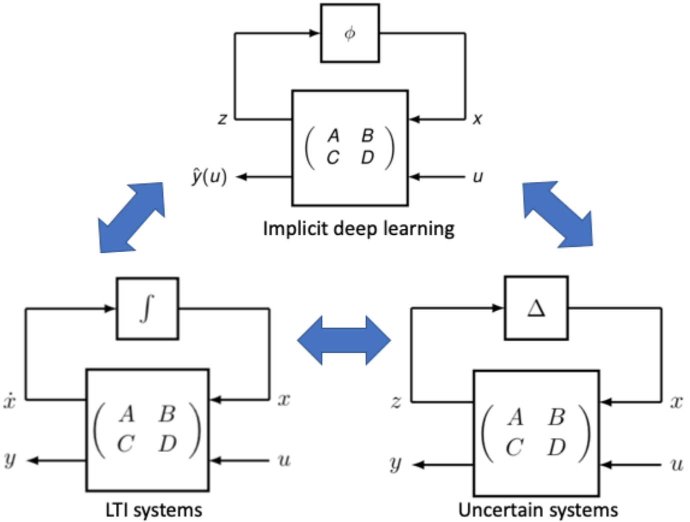


Take-aways



- Implicit models are more general than standard neural networks.
- Well-posedness is a key property that can be enforced via norm or eigenvalue conditions.
- Models can be composed together in modular fashion.
- The notationally very simple framework allows for rigorous analyses for robustness, model compression, architecture optimization, etc.
- The corresponding training problem is amenable to SGD methods.

Towards a general theory?



References

Stephen Boyd.
Perron-Frobenius theory, 2008.
Lecture slides for EE 363, Stanford University.

Geir E Dullerud and Fernando Paganini.
A course in robust control theory: a convex approach, volume 36.
Springer Science & Business Media, 2013.

L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai.
Implicit deep learning.
Submitted to ICML, preliminary version at https://arxiv.org/abs/1908.06315, February 2020.