CIS 522: Lecture 2

Linear Deep Theory/ Intro to theory

Penn Engineering

Today

(1) Some admin

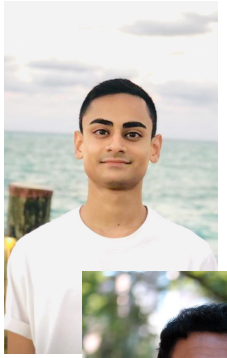
(2) Review first set of notebooks

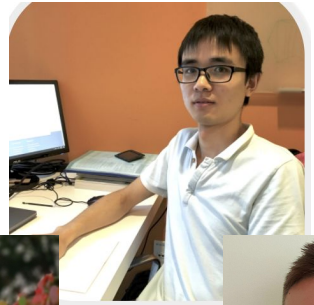
(3) Questions and Answers

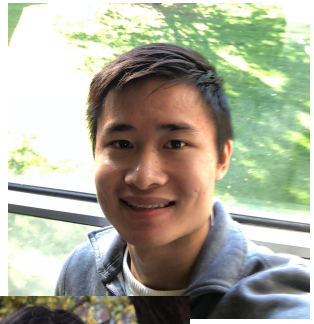
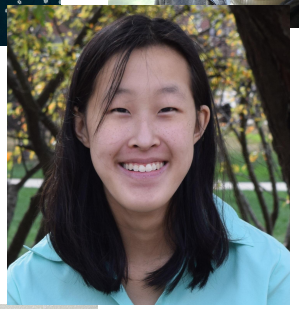
Please start asking on chat now so we can prioritize/ order

(4) DL vs other kinds of ML / theory/ why week 2 matters

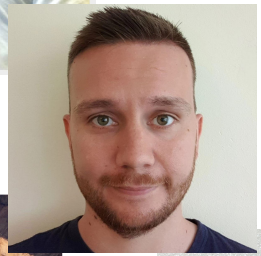
Who we are - (Lots of) TAs

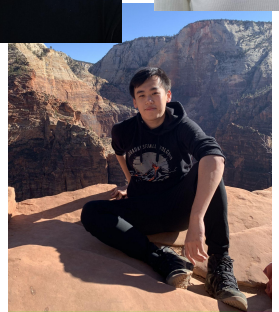
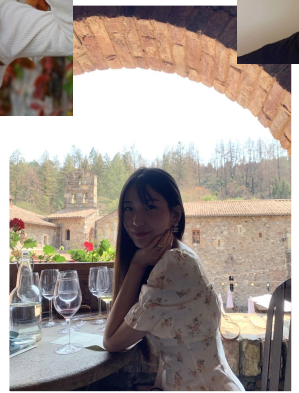
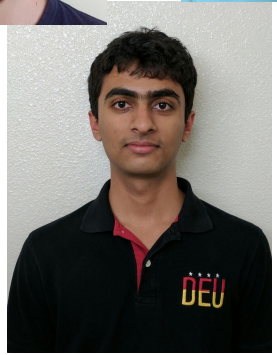









Permits

Should all be sent today. Contact Anka via email tomorrow (akreuel@seas.upenn.edu) if something is missing.

Slack vs Piazza

To dos this week

Pods start! Make sure you've joined your pod's Slack channel; reach out to Anka if you don't have a pod yet

Submit workbooks W2D1 and W2D2 by Monday @1pm ET

Submit first homework by Tuesday @ noon ET

(2) Let us talk about this week's notebooks

Life is so easy

Network construction

Initialization

Autograd

Data load/ process

Optimization

Getting to know pytorch

Making tensors

Tensor operations and methods

Reshaping/ Permuting/ Concatenating tensors

Interfacing with Python

Basic dataset loaders including transformations

Tools we will use

Using GPUs - memory and compute

Airtable

Colab

A first neural network

Making a neural network

Training loop

Questions and Answers

Let us spend some time talking about them

Looking towards to the week ahead

Learning in linear networks

We can understand quite a bit about them!

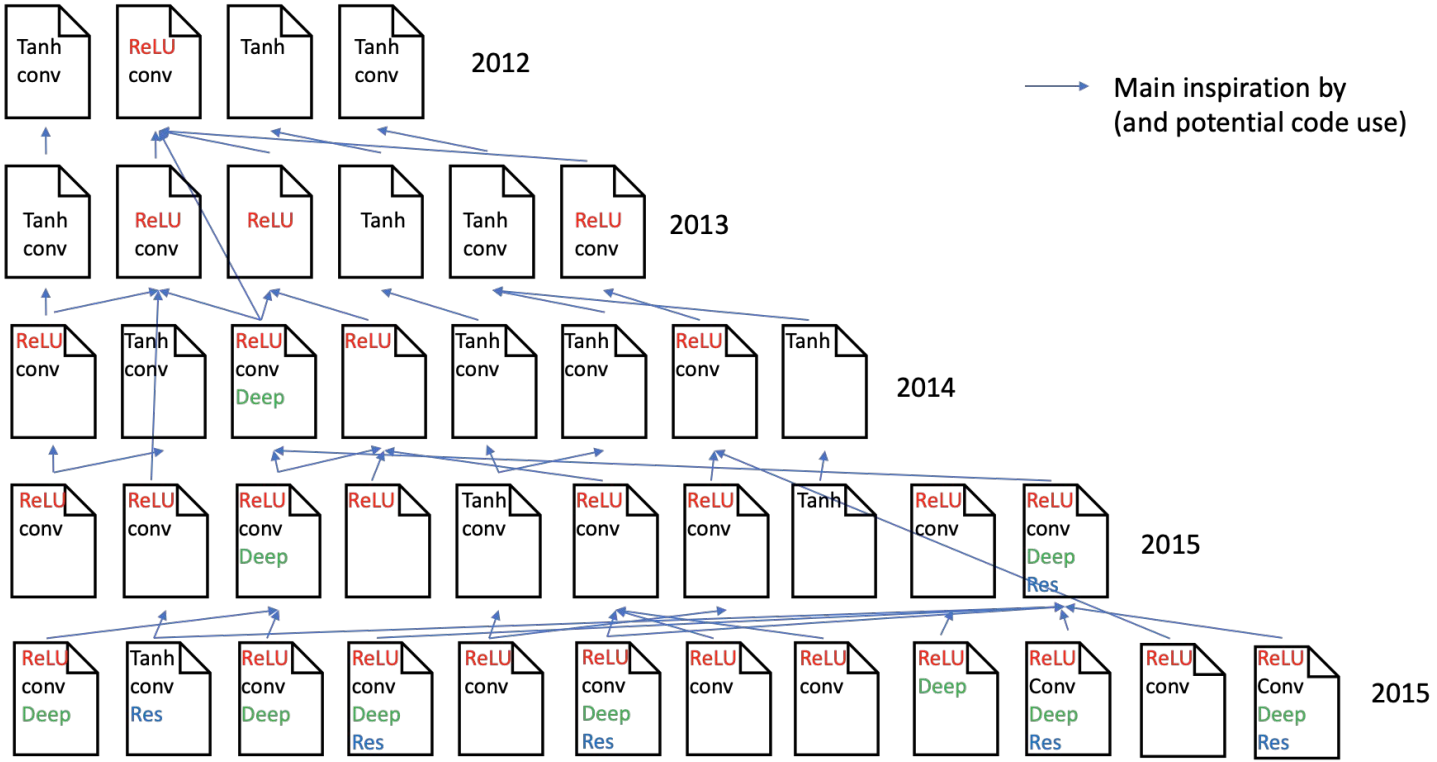
Learning in multilayer linear neural networks

We can understand a few aspects

Let us first think a bit about how the DL field works in practice

Insert picture of a crowd

One way of thinking about deep learning



DL = countless cool tricks

General

Attention

84 methods

13149 papers with code

Stochastic Optimization

56 methods

9142 papers with code

Regularization

53 methods

20401 papers with code

Attention Mechanisms

47 methods

6878 papers with code

Activation Functions

46 methods

20488 papers with code

See all 128 categories

Computer Vision

Convolutional Neural Networks

114 methods

4136 papers with code

Image Model Blocks

90 methods

4988 papers with code

Generative Models

87 methods

5542 papers with code

Object Detection Models

65 methods

1564 papers with code

Image Feature Extractors

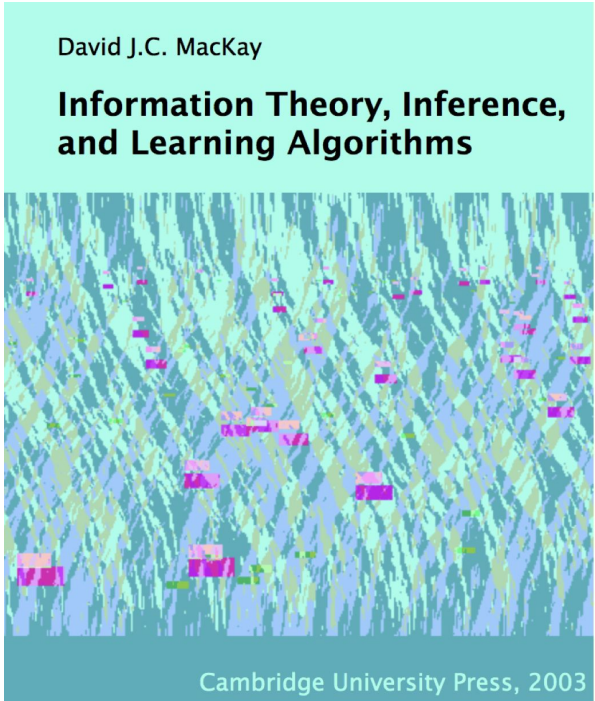
59 methods

99 papers with code

Complexity in Evolution

How many bits can we have acquired in the 4M years since we were apes?

Naïve answer? 400k, 1 bit/generation



From: David MacKay: Information Theory, Inference, and Learning algorithms.

Learning easy problems in big population = fast

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Mutations

Randomly flip bits

Better half survives

Reaches perfect f=1 in

$$\approx 2N$$

Generations

Crossovers

Randomly inherit bits

Better half survives

Reaches perfect f=1 in

$$\approx \pi/\sqrt{2/(\pi+2)}\sqrt{N}$$

Generations

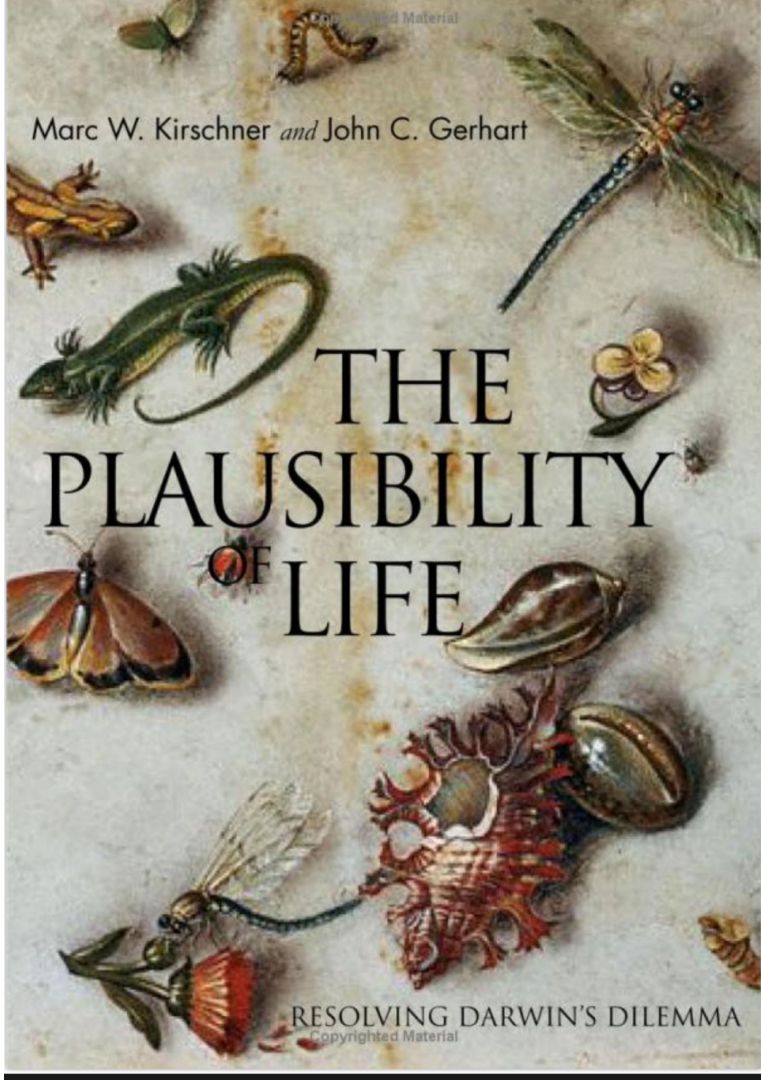
A great book on evolutionary biology

Constraints

Deconstraints

Development

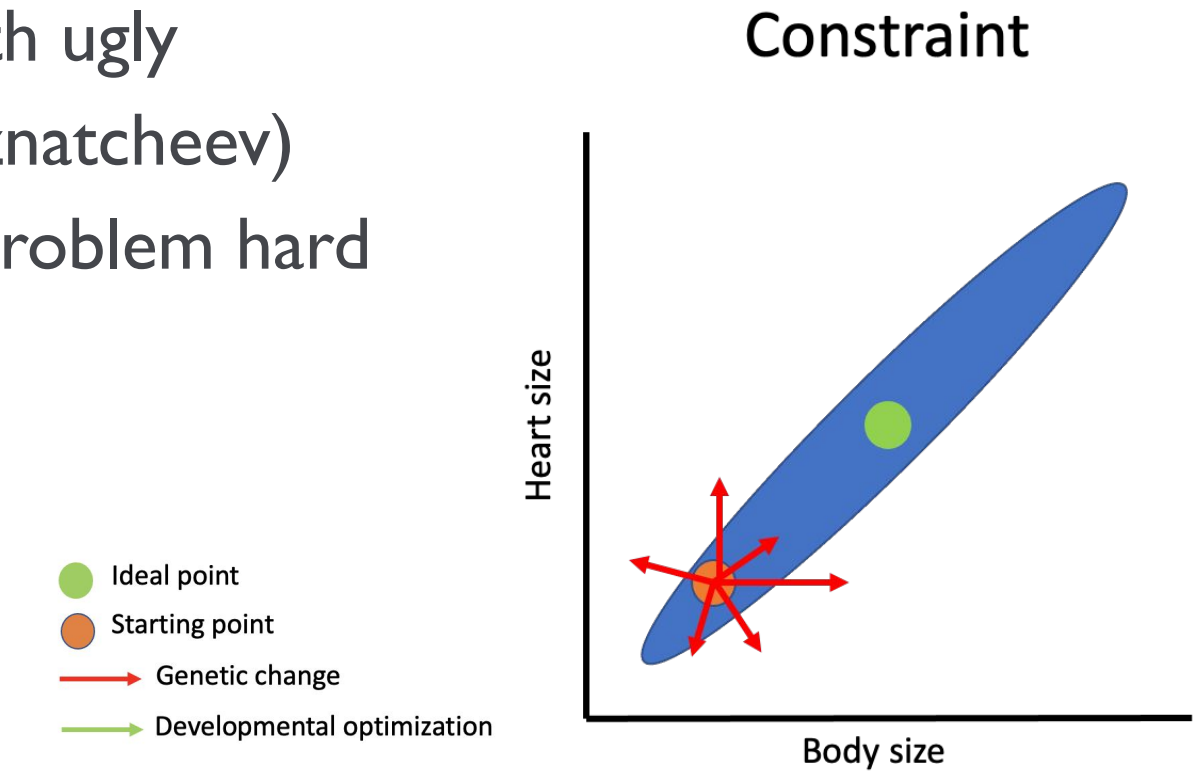
Weak regulation



What did I learn? Constraints

Learning is hard with ugly landscapes (see Kaznatcheev)

Constraints make problem hard



Deconstraints

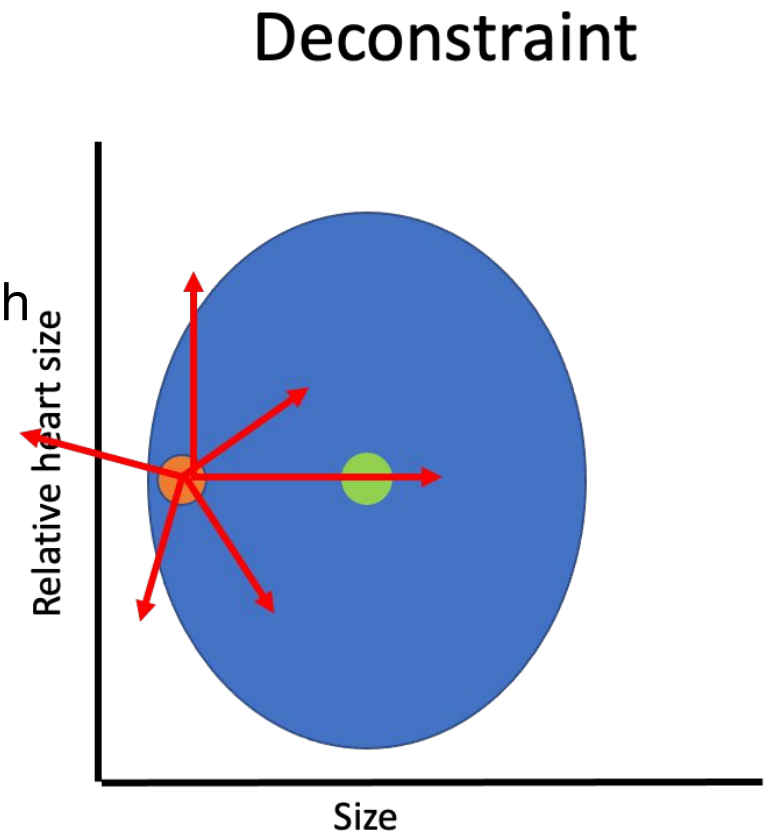
Examples in DL

Learning rate vs Batch size

Instead keep ratio constant

Choose network/ gotta make initialization match

Xavier/ He



Development

Examples in DL

Networks and learning rules

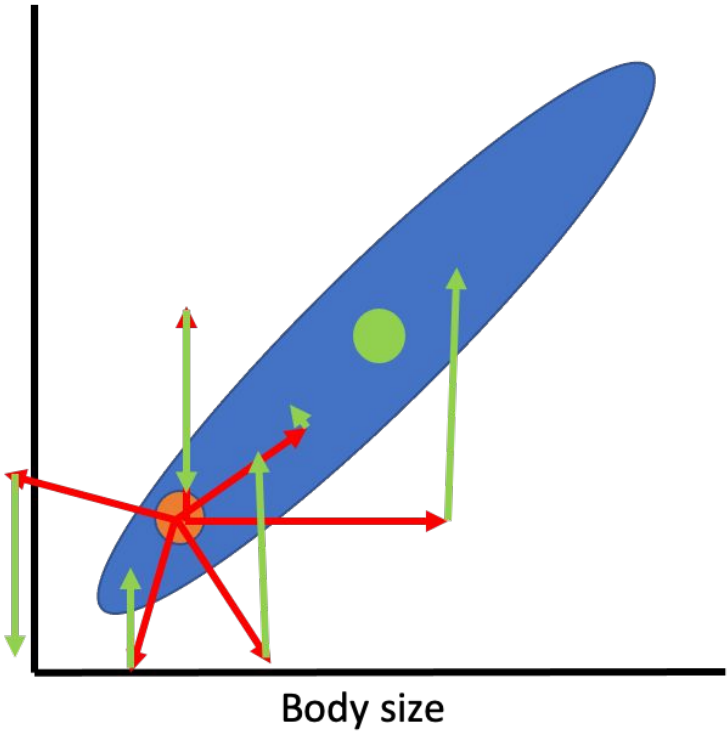
GD

A network and a gradient I calculate

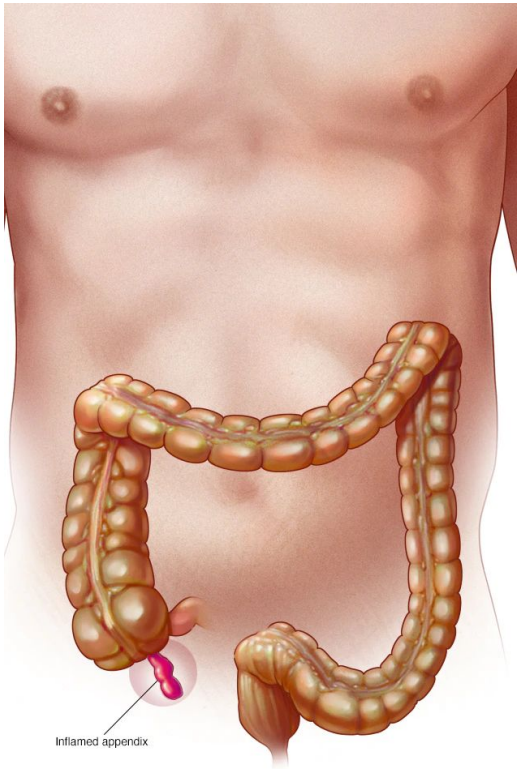
Autograd

Lots of dependent parameters

Hyperparameter optimizers



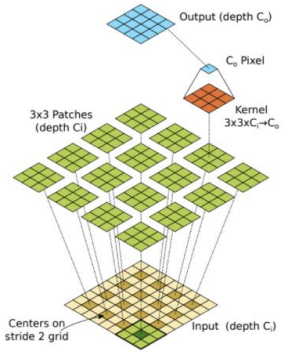
We already see co-evolution



Machine Learning Systems are Stuck in a Rut

Paul Barham
Google Brain

Michael Isard
Google Brain



Abstract

In this paper we argue that systems for numerical computing are stuck in a local basin of performance and programmability. Systems researchers are doing an excellent job improving the performance of 5-year-old benchmarks, but gradually making it harder to explore innovative machine learning research ideas.

We explain how the evolution of hardware accelerators favors compiler back ends that hyper-optimize large monolithic kernels, show how this reliance on high-performance but inflexible kernels reinforces the dominant style of programming model, and argue these programming abstractions lack expressiveness, maintainability, and modularity; all of which hinders research progress.

This view is correct

But sad!

But it is obviously not everything

I hope that you will all contribute to make DL applications be less random

What I love about week 2

Produces some deep understanding

Helps us escape the evolution trap

Let's make this be real engineering

A superquick review of a few high level theoretical concepts

The basic idea of most of ML

Similar data points have similar labels.

Kernel based ML

What is the time complexity of SVM?

Time complexity of SVM vs DL

O(N^3)

Neural networks?

O(NM) where M is the number of parameters

Is DL a Kernel system?

Deep learning has that property when initialized with large weights (Kernel regime)

$$K_{\mathbf{w}(t)}(x, x') = \langle \nabla_{\mathbf{w}} f(\mathbf{w}(t), \mathbf{x}), \nabla_{\mathbf{w}} f(\mathbf{w}(t), \mathbf{x}') \rangle$$

Feature learning

Intuition: early areas figure out which dimensions matter. Makes learning easier. Make Kernels better.

So-called Rich regime.

We see them in this week.

Lottery ticket hypothesis

Lucky initializations (lottery tickets)

Allow relatively small networks to learn well

(Frankle and Carbin, 2018)

No Free Lunch Theorem

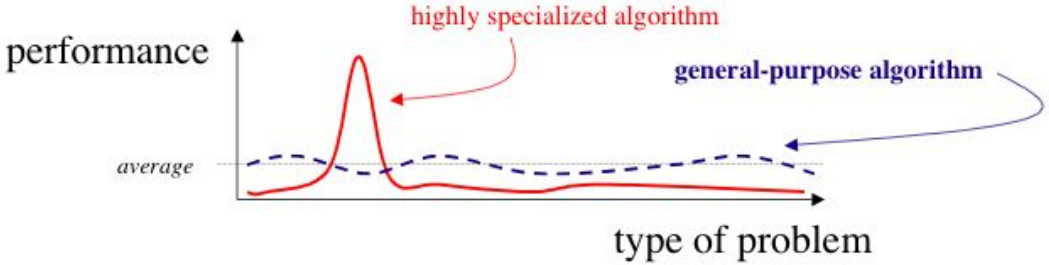
Theorem: If an algorithm performs better than random search on some class of problems then it **must** perform worse than random search on the remaining problems

In other words:

- A superior black-box optimization strategy is impossible
- If our feature space is unstructured, we can do no better than random
- "A jack of all trades is a master of none"

Wolpert and MacReady (1997)

No Free Lunch Theorem (example)



Published: December 1983

On arbitrarily slow rates of global convergence in density estimation

Luc Devroye

Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete **62**, 475–483 (1983) | Cite this article

Engineering Deep learning system

Which regularizers work and why?

Why do neural networks generalize?

How can we make them work fast?

How can we build in what we know about the world?

Can I tell you how the world works and you tell me which algorithm I should use?

New developments (2021)

DOI:10.1145/3446776

Understanding Deep Learning (Still) Requires Rethinking Generalization

By Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals

Nonvacuous bounds

PAC Bayes for DL

Reasons for implicit regularization

And another

On the role of data in PAC-Bayes bounds

Gintare Karolina Dziugaite[1,2] **Kyle Hsu**[3,4] **Waseem Gharbieh**[1] **Gabriel Arpino**[3,4] **Daniel M. Roy**[3,4]
[1]Element AI ServiceNow [2]Mila [3]U. of Toronto [4]Vector Institute

PAC Bayes uses priors

Make this be data dependent

Sometimes this makes a huge difference

To dos this week

Pods start! Make sure you've joined your pod's Slack channel; reach out to Anka if you don't have a pod yet

Submit workbooks W2D1 and W2D2 by Monday @1pm ET

Submit first homework by Tuesday @ noon ET

Individual questions?

Feel free to stay on and talk with me