[DLSS’18; Toronto] http://unsupervised.cs.princeton.edu/deeplearningtutorial.html

Toward Theoretical Understanding of Deep Learning

Sanjeev Arora
Princeton University
Institute for Advanced Study

http://www.cs.princeton.edu/~arora/ @prfsanjeevarora
Group website: unsupervised.princeton.edu

Support: NSF, ONR, Simons Foundation, Schmidt Foundation, Amazon Resarch, Mozilla Research. DARPA/SRC

Some History

1950s and 1960s: "Perceptron" (single-layer, 1 output node)

1969: Perceptrons are very limited [Minsky,Papert]

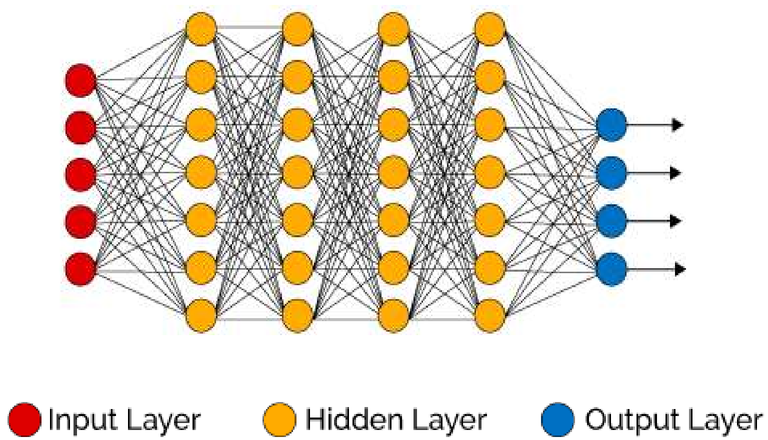
1990s-2010: Perceptron+ (namely, SVMs) strikes back.
(Riding on convex optimization theory, generalization theory, expressivity theory of kernels, etc.).

Last 10 years: Age of deep learning (multilayer perceptron).

Theory??

?

Deep Learning: Terminology



θ: Parameters of deep net

$(x_1,y_1), (x_2,y_2),\ldots$ iid (point, label) from distribution *D*
(training data)

$$\ell(\theta, x, y)$$

Loss function (how well net output matched true label y on point x) Can be l_2, cross-entropy….

Objective $\quad \text{argmin}_\theta E_i[\ell(\theta, x_i, y_i)]$

Gradient Descent $\quad \theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta (E_i[\ell(\theta^{(t)}, x_i, y_i)])$

Stochastic GD: Estimate ∇ via small sample of training data.

Optimization concepts already shape deep learning

(together with GPUs, large datasets)

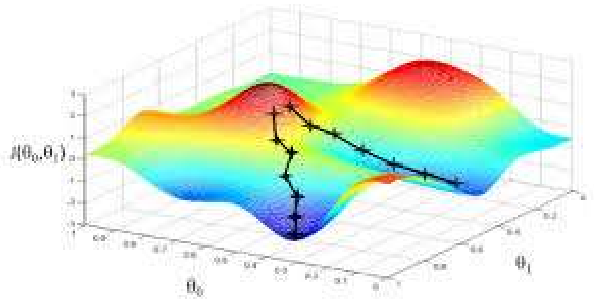
- Backpropagation: Linear time algorithm to compute gradient.

- "Gradient/landscape shaping" drives innovations such as resnets, wavenets, batch-normalization, …

- Gradient Descent++ (Momentum, Regularization, AdaGrad,..)
  Came from convex world.

Goal of theory: Theorems that sort through/support competing intuitions, leading to new insights and concepts.

Talk overview



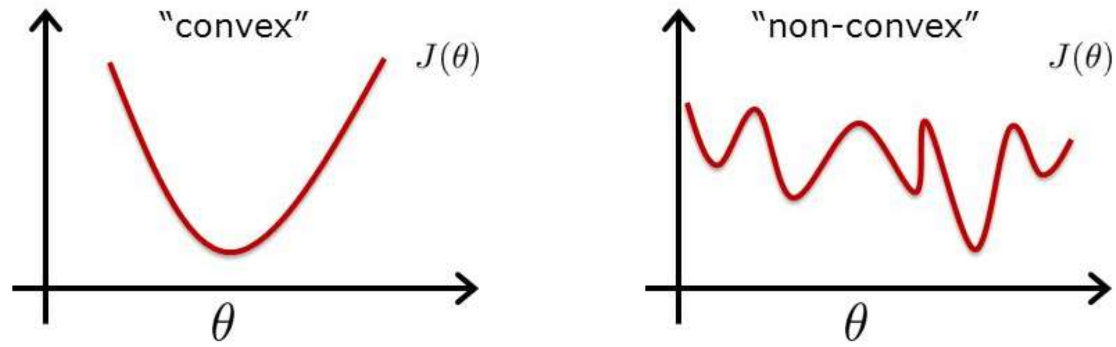
Training error

$$E_i[\ell(\theta, x_i, y_i)]$$

Test error

$$E_{(x,y)\in\mathcal{D}}[\ell(\theta, x, y)]$$

- Optimization: When/how can it find decent solutions? Highly nonconvex.
- Overparametrization/Generalization: # parameters $\gg$ training samples. Does it help? Why do nets generalize (**predict** well on **unseen** data)?
- Role of **depth**?
- Unsupervised learning/GANs
- Simpler methods to replace deep learning? (Examples of Linearization from NLP, RL…)

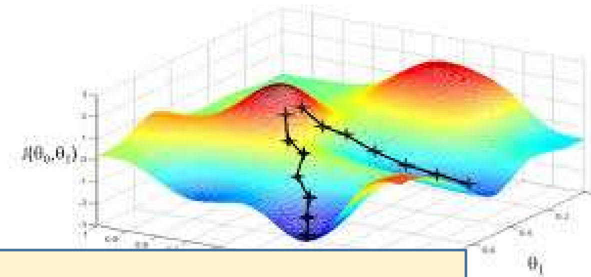


Part 1: Optimization in deep learning

Hurdle: Most optimization problems in deep learning are nonconvex, and on worst-case instances are NP-hard.

Basic concepts

Note: $\nabla \neq 0$ ➔ $\exists$ descent direction.



Possible goals: Find critical point ($\nabla = 0$).

Find "local optimum", ie bottom of some valley (∇^2 is psd/ $\nabla^2 \succcurlyeq 0$)

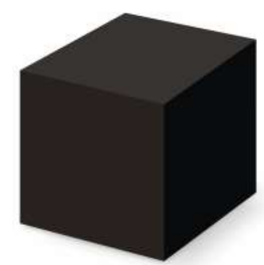
Find global optimum $\boldsymbol{\theta}^*$.

Assumption about initialization:

Convergence from all starting points $\boldsymbol{\theta}$?

Random initial point? Special initial points?

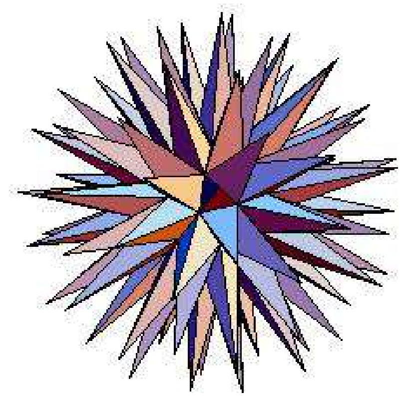
Black box vs nonblack box



In R^d, want run times poly(d, 1/ε) where ε= accuracy.

(naive: exp(d/ε)). Recall: d > 10^6

Curse of dimensionality

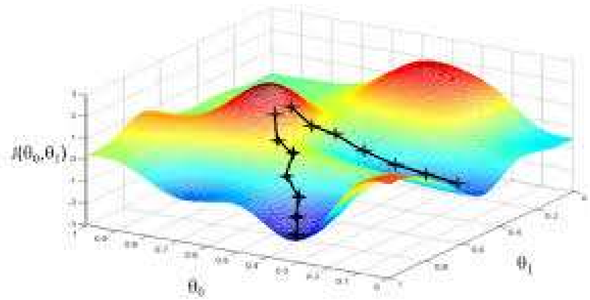


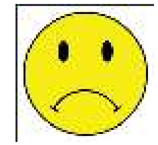
In R^d, $\exists$ exp(d) directions
whose pairwise angle is > 60 degrees

$\exists$ exp(d/ε) special directions s.t. all directions have angle at most ε with one of these ("ε-net", "ε–cover")

"Time to explore d-dimensional parameter space."
(INFEASIBLE)

Black box analysis for deep learning



Why: Don't know the landscape, really 

$$\text{argmin}_\theta E_i[\ell(\theta, x_i, y_i)]$$

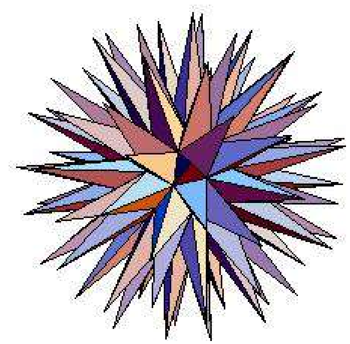
No clean math. characterization of (x_i, y_i) !

θ →  → $f(\theta)$

$\nabla f|_\theta$

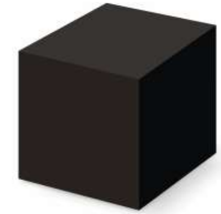
f = training error

INFEASIBLE to find global optimum; must settle for weaker solutions

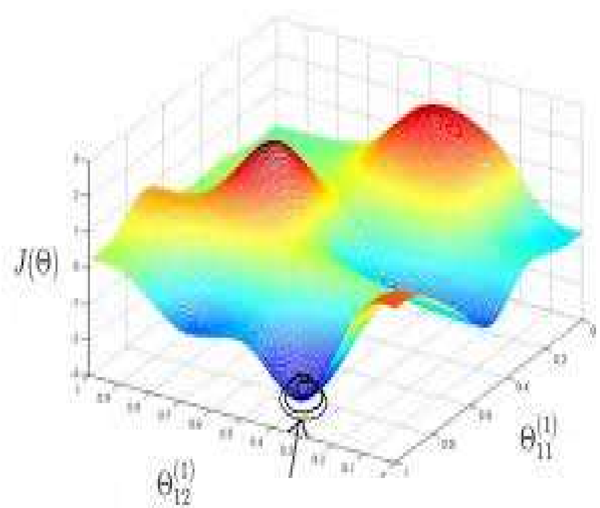


[NB: Some attempts to understand landscape via statistical physics; not much success so far..]

Gradient descent in unknown landscape.



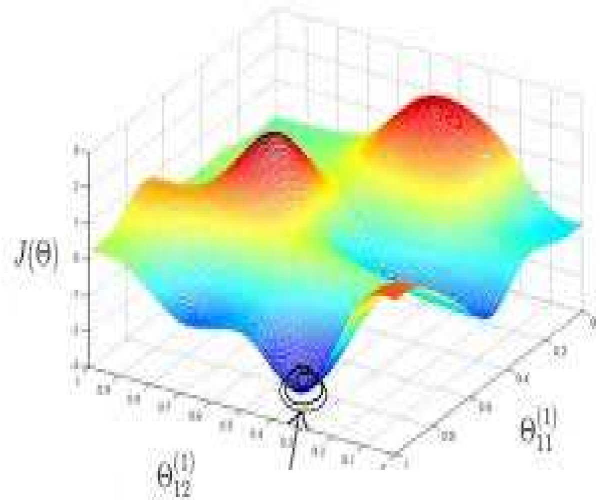
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \, \nabla f(\boldsymbol{\theta}_t)$$

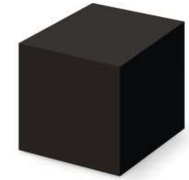


- □ $\nabla \neq 0$ ➔ $\exists$ descent direction
- □ But if 2^{nd} derivative (∇^2) high, allows ∇ to fluctuate a lot!
- □ ➔ To ensure descent, take small steps determined by smoothness $\nabla^2 f(\boldsymbol{\theta}) \preccurlyeq \boldsymbol{\beta} I$

Can be assumed via gaussian smoothening of f.

Gradient descent in unknown landscape (contd.)





$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \, \nabla f(\boldsymbol{\theta}_t)$$

- Smoothness $-\boldsymbol{\beta} I \preccurlyeq \nabla^2 f(\boldsymbol{\theta}) \preccurlyeq \boldsymbol{\beta} I$

CLAIM: $\eta = 1/2\beta \Rightarrow$ achieve $|\nabla f| < \boldsymbol{\varepsilon}$ in #steps proportional to $\beta/\boldsymbol{\varepsilon}^2$.

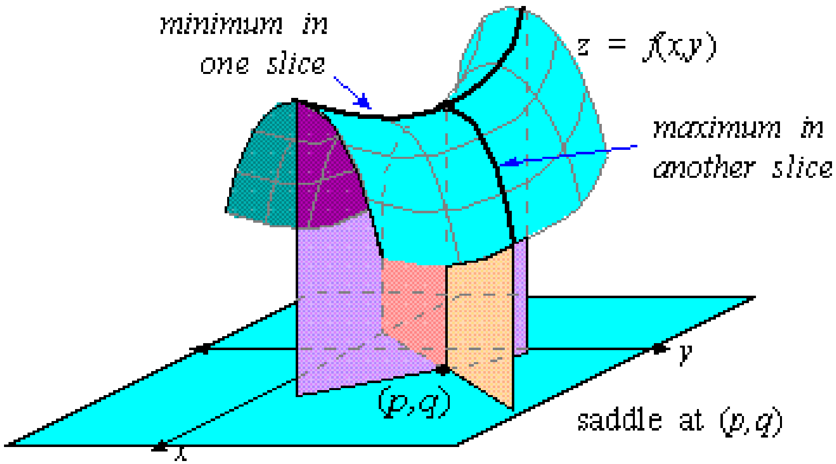
Pf: $f(\theta_t) - f(\theta_{t+1}) \geq -\nabla f(\theta_t)(\theta_{t+1} - \theta_t) - \frac{1}{2}\beta|\theta_t - \theta_{t+1}|^2$

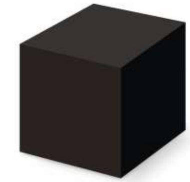
$$= \eta|\nabla_t|^2 - \frac{1}{2}\beta\eta^2|\nabla_t|^2 = \frac{1}{2\beta}|\nabla_t|^2$$

➜ update reduces function value by $\boldsymbol{\varepsilon}^2/2\beta$

But critical point ($\nabla f = 0$) is a weak solution concept.

Evading saddle points..





[Ge, Huang, Jin, Yuan'15] Add noise to gradient ("perturbed GD")

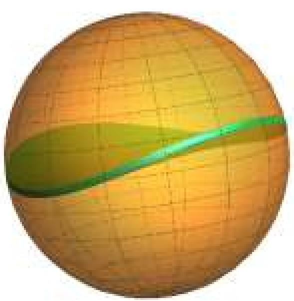
(Runtime improvements: Jin et al'17]

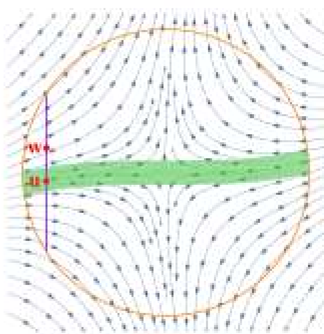
Min in n-1 dimensions, max in one

Analysis of random walk

➔ Within poly(d/ε) time "escape" all saddle points and achieve "Approx 2nd order minimum"

$$||\nabla f|| \leq \epsilon \quad \nabla^2 f \succcurlyeq -\sqrt{\epsilon}\, I$$





(Analyses of probability flow out of "stuck region")

[NB: perturbed GD with large noise = "Langevin dynamics" in stat. physics)

2^{nd} order optimization for deep learning? (Newton method?)

$$x_{t+1} = x_t - \eta\,[\nabla^2 f(x)]^{-1}\,\nabla f(x)$$

θ

$f(\theta)$

$\nabla f|_\theta$

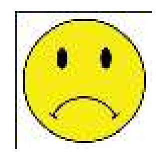
$\nabla^2 f|_\theta$ [Werbos'74]

More relevant: For any vector v, Backprop can compute $(\nabla^2 f)v$ in linear time. [Pearlmutter'94]

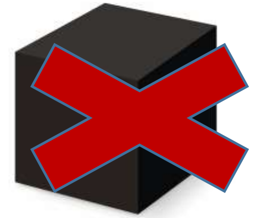
Can do approximate 2^{nd} order optimization asymptotically faster than 1^{st} order! (empirically, slightly slower) [Agarwal et al'17, Carmon et al'17]

Idea: $$(\nabla^2)^{-1} = \sum_{i=0 \text{ to } \infty} (I - \nabla^2)^i$$ (but use finite truncation)

But 2^{nd} order doesn't seem to find better quality nets so far.



Non-black box analyses



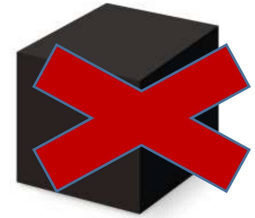
Various ML problems that're subcases of depth 2 nets
(i.e., one hidden layer between input and output).

- Often make assumptions about the net's structure, data distribution, etc. (landscape is mathematically known) s

- May use different algorithm (eg, tensor decomposition, alternating minimization, convex optimization …) than GD/SGD.

Topic modeling [A., Ge, Moitra'12] [A. et al'13]
Sparse coding [A., Ge, Ma, Moitra'14, '15] [Anandkumar et al'14]
Phase retrieval [Candes et al'15]
Matrix completion [Many papers, eg Jain et al.'13]
Matrix sensing
Learning Noisy Or nets [A., Ge, Ma'17]

Convergence to global optimum from arbitrary initial point (via understanding the landscape)

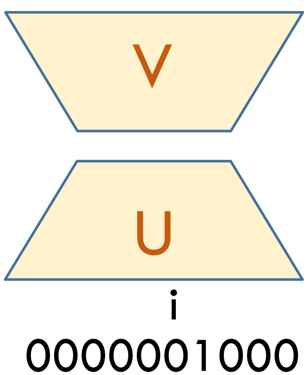


Matrix completion

$$\begin{bmatrix} & & \\ & M & \\ & & \end{bmatrix} = \begin{bmatrix} \\ U \\ \\ \end{bmatrix} \cdot \begin{bmatrix} & V^{\top} & \end{bmatrix}$$

Given O(nr) random entries of an nxn matrix M of rank r, predict missing entries.

Subcase of learning depth 2 nets



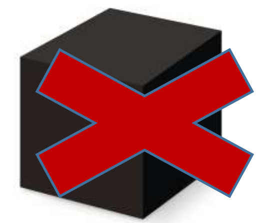
Feeding1-hot inputs into unknown net; seeting output at one random output node. Learn the net!

[Ge, Lee, Ma'17] All local minima are global minima. So perturbed GD finds global minimum from arbitrary initial point. (proof is fairly nontrivial)

Any theorems about learning multilayer nets?

Yes, but usually only for linear nets (i.e., hidden nodes compute f(x) = x).

Overall net = product of matrix transformation
= itself a linear transformation



But optimization landscape still holds surprises…

Some papers: [Baldi, Hornik'88], [Saxe et al '13] (dynamics of training) [Kawaguchi'16] [Hardt and Ma'16] (landscape of linear resnets); [A., Cohen, Hazan'18]

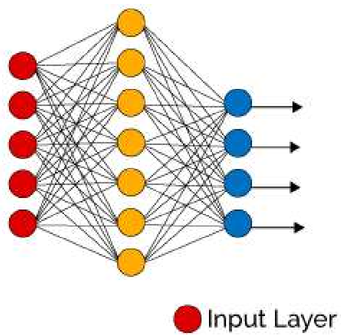
Some other optimization ideas I did not cover (some are only semi-theorems)

- Budding connections with "physics" ideas: natural gradient, Lagrangian method, ..
- Adversarial examples and efforts to combat them
- Optimization for unsupervised learning (esp. probabilistic models), reinforcement learning.
- Information-theoretic interpretation of training algorithms, eg "information bottleneck"

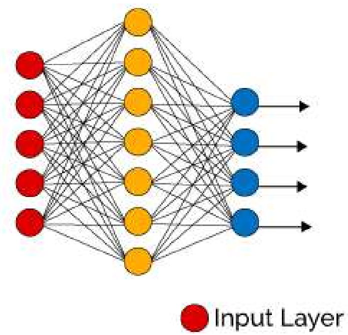
Part 3: Overparametrization and Generalization theory

e.g., Why is it a good idea to train VGG19 (20M parameters) on CIFAR10 (50K samples)? No overfitting?

Overparametrization may help optimization : folklore experiment e.g [Livni et al'14]



Generate labeled data by feeding random input vectors Into depth 2 net with hidden layer of size n

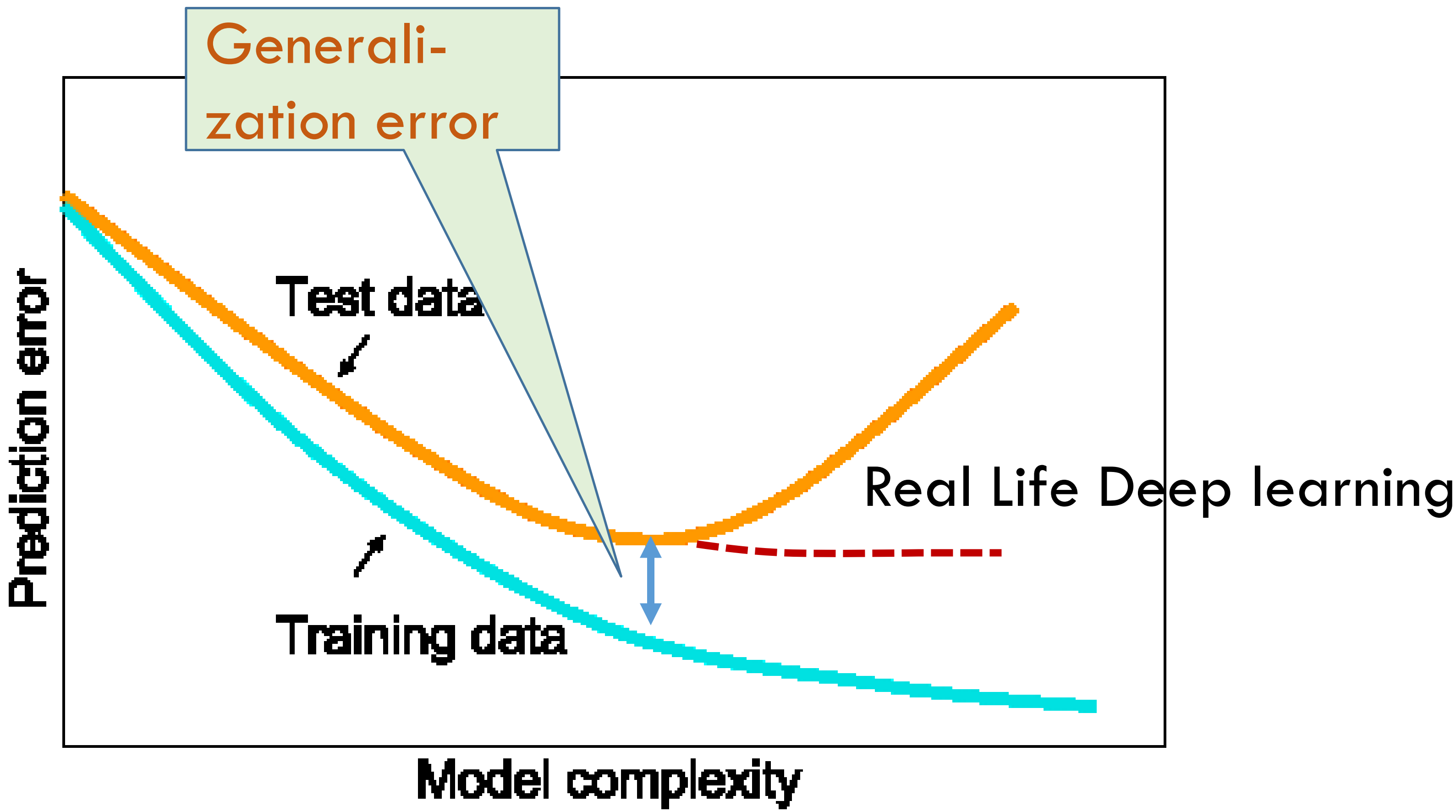


Difficult to train a new net using this labeled data with same # of hidden nodes

Still no theorem explaining this…

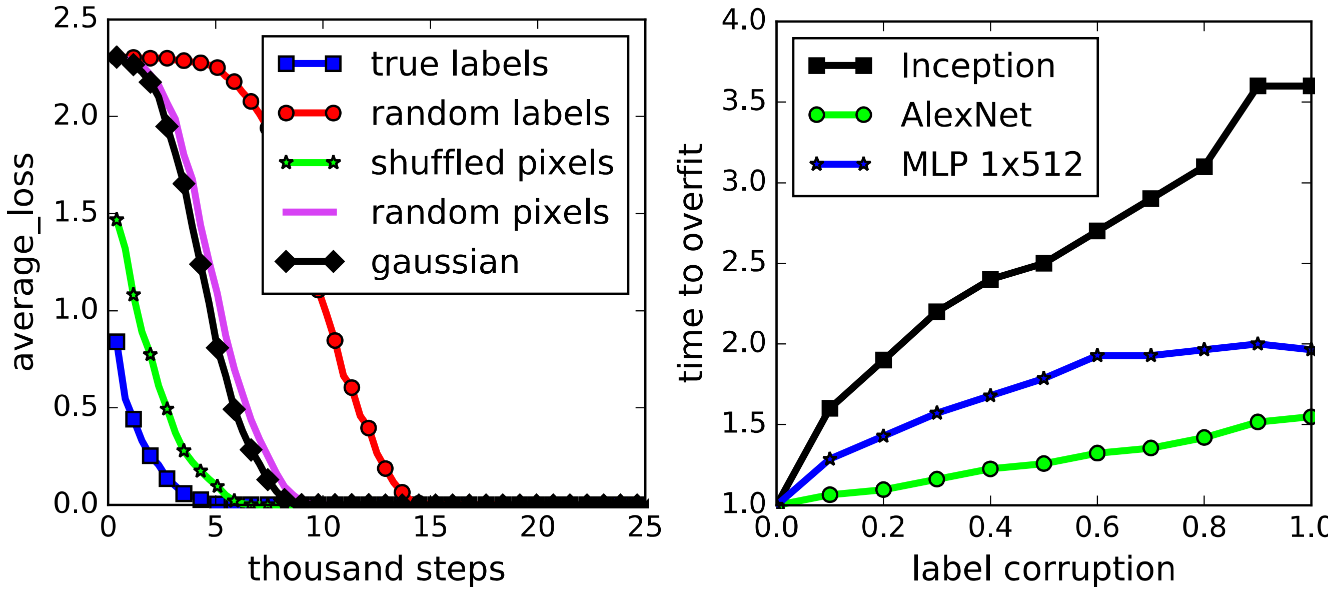
Much easier to train a new net with bigger hidden layer!

But textbooks warn us: Large models can "Overfit"



Longtime belief: SGD + regularization eliminates "excess capacity" of the net

But, excess capacity is still there!

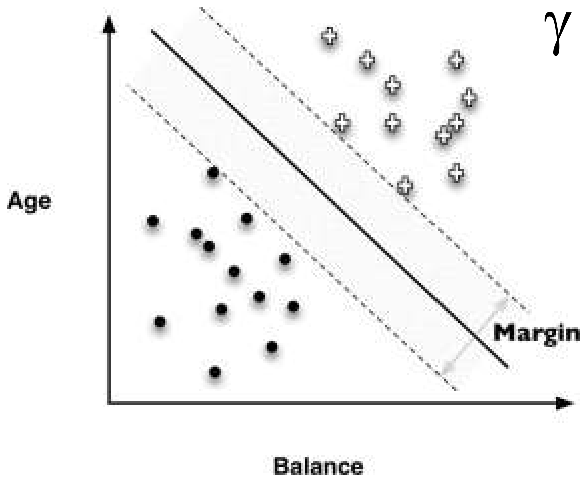


Inception v3 net on CIFAR10

Hope: An explanation will also identify intrinsic structure of a “well-trained” net.

[Understanding deep learning requires rethinking general]

BTW: Excess capacity phenomenon exists in linear models!



γ = margin

Classifier has d parameters.

Quantify effective capacity of deep nets??

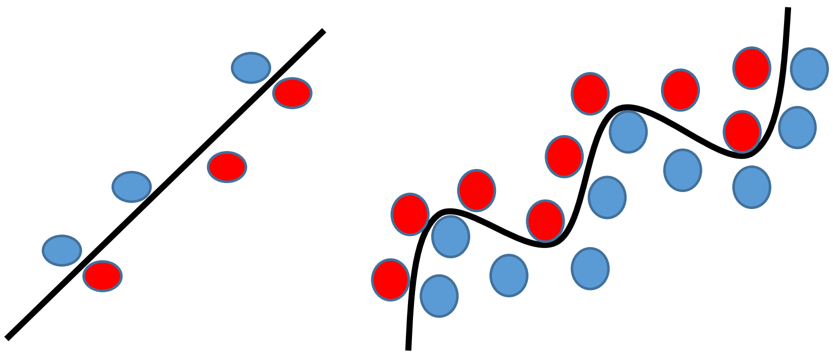
If has margin γ , then possible to learn it with
$< O(\log d)/\gamma^2$ datapoints! (low effective capacity!)

But always possible to fit linear classifier to d-1
randomly labeled datapoints as well.

See also [Understanding deep learning requires understanding kernel learning. Belkin et al'18]

Effective Capacity

Roughly, log (# distinct a priori models)



(Rough Analogy: If a system exhibits 2^k distinct states we say it has k bits of memory.)

Generalization Theory (Main Thm)

$$\text{Test loss} - \text{training loss} \leq \sqrt{\frac{N}{m}}$$

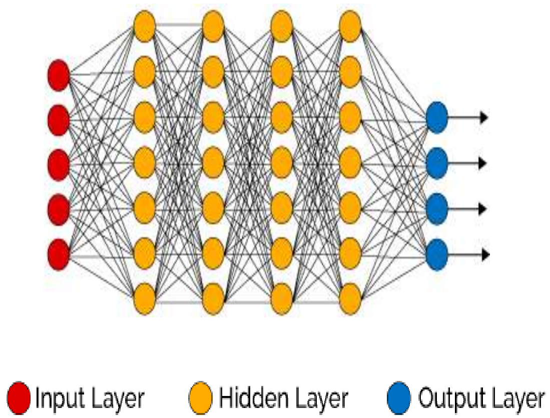
m = # training samples. N = effective capacity

Usual upperbounds on N: # of parameters, VC dimension, Rademacher

For deep nets, all of these are about the same (usually vacuous)

$$\text{Test loss} - \text{training loss} \leq \sqrt{\frac{N}{m}}$$

Proof Sketch



- Fix deep net $\boldsymbol{\theta}$ and its parameters
  $Err_{\boldsymbol{\theta}}$ = test error
- Take iid sample S of m datapoints,
  $Err_{\boldsymbol{\theta},S}$ = avg error on S = training error
- By concentration bounds, for fixed net $\boldsymbol{\theta}$
  Pr_S[diff. between them $\leq \boldsymbol{\varepsilon}$] > 1 - exp(- $\boldsymbol{\varepsilon}^2$m)

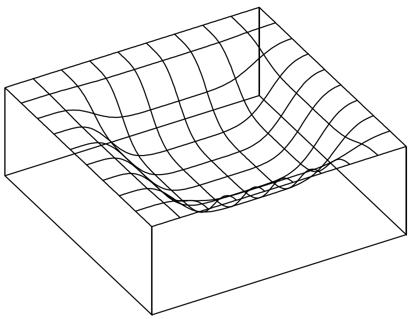
"Effective capacity"
= log $\mathcal{W}$, = N

Complication: depends upon training sample S

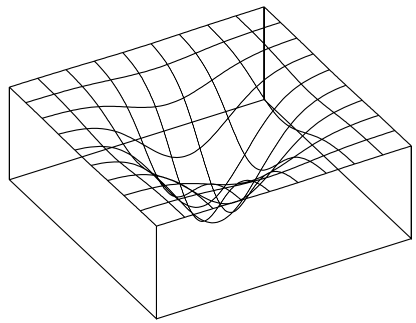
Solution: Union bound over "all possible" $\boldsymbol{\theta}$.
If # (possible $\boldsymbol{\theta}$) = $\mathcal{W}$, suffices to let m > (log $\mathcal{W}$)/$\boldsymbol{\varepsilon}^2$

Old notion: Flat Minima

[Hinton-Camp'93][Hochreiter, Schmidhuber'95]



Flat



Sharp

Multiple arguments ➔ "Noise in SGD favors flat minima"

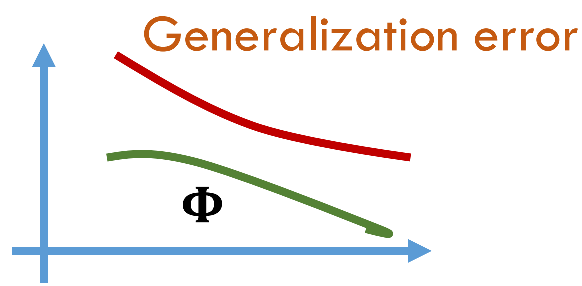
"Flat minima" generalize better empirically [Keskar et all'16]

Flat minimum has lower description length
➔ Fewer # of "possible models" with such small descriptions

Makes intuitive sense but hard to make quantitative…
(and false for some defns of "sharp" [Dinh et al'17])

Current status of generalization theory: postmortem analysis…

It has Property Φ shared by very few nets…

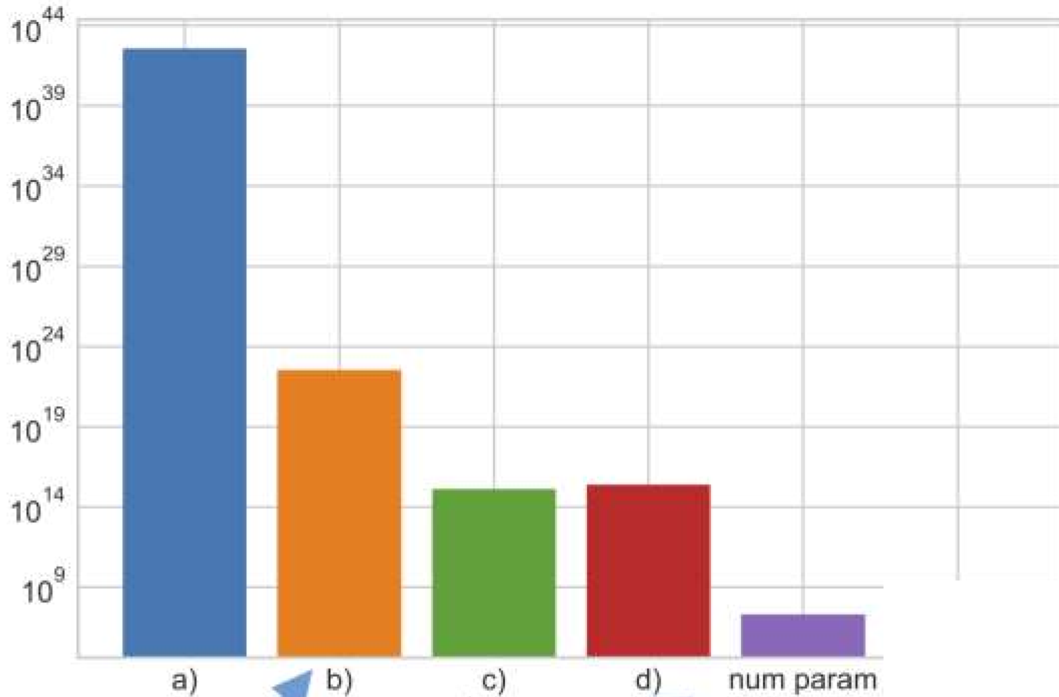


Qualitative check: Correlates with generalization?

Quantitative: Use property Φ to compute upper bound on "very few" and hence effective capacity (much harder!)

Nonvacuous estimate of "true capacity" has proved elusive (starting point [Langfod-Caruana02]

VGG19
(19 layers)



[Bartlett-Mendelson'02]

[Neyshabur et al '17]

[Bartlett et al NIPS17],

[Neyshabur et al ICLR18]

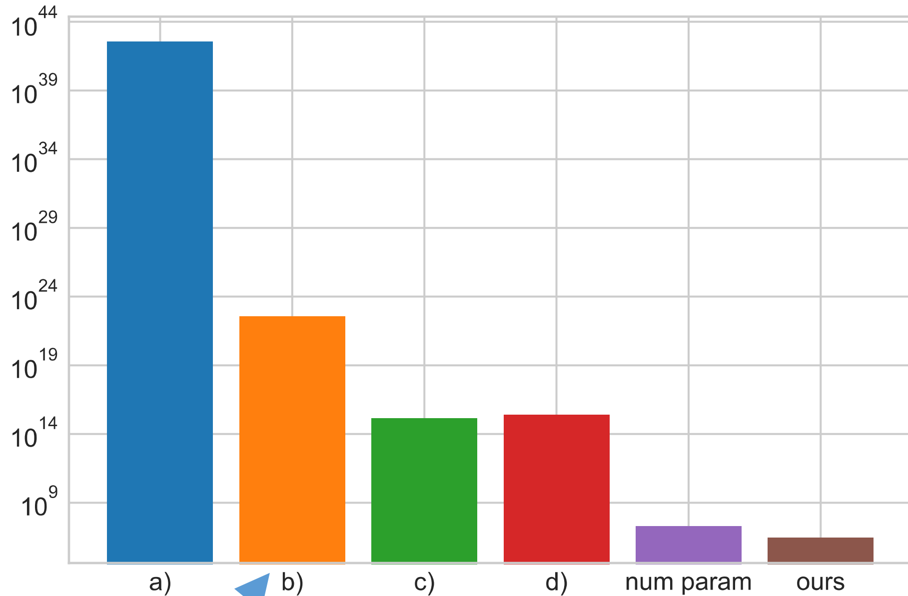
Zhang'18]

Ideas include: PAC-Bayes, Margin, Rademacher, ..

[Dziugaite-Roy'17] have more nonvacuous bounds for MNIST** but not an asymptotic "complexity measure"

Nonvacuous bound on "true parameters" has proved elusive..

VGG19
(19 layers)



[Bartlett-Mendelson'02]

[Neyshabur et al'17]

[Bartlett et al'17],

[Neyshabur et al'17]

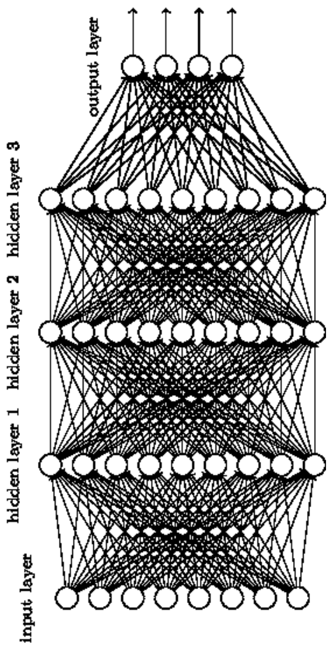
[A., Ge, Neyshabur, Zhang'18]

Main idea: Using noise stability to bound effective Capacity.

Noise stability for deep nets

(can be seen as a "margin" notion for deep nets)

[A., Ge, Neyshabur, Zhang ICML'18]

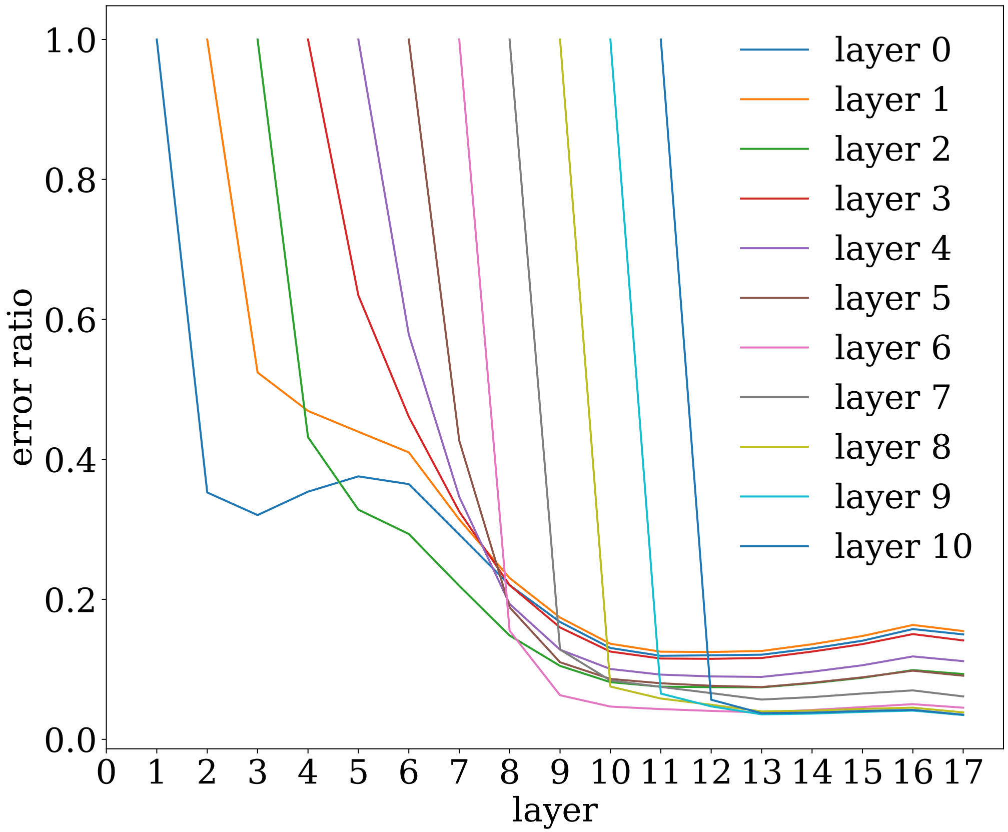


Noise injection: Add gaussian η to output x of a layer ($|\eta| = |x|$)

Measure change in higher layers. (If small, then net is noise stable.)

Noise stability of VGG19

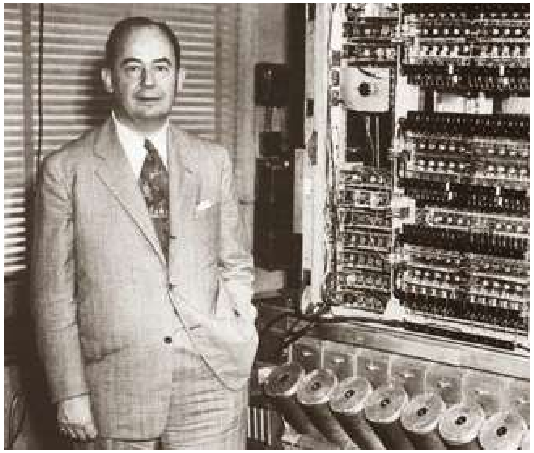
[A., Ge, Neyshabur, Zhang ICML'18]



How injected gaussian noise gets attenuated as it passes through to higher layers.

(Each layer fairly stable to noise introduced at lower layers!)

Related to other noise stability phenomena independently discovered in experiments of [Morcos et al ICLR'18]



Von Neumann, J. (1956).
Probabilistic logics and the synthesis of reliable organisms from unreliable components.

Shannon, C. E. (1958).
Von Neumann's contributions to automata theory.

"Reliable machines and unreliable components…

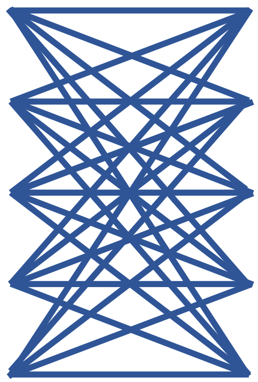
We have, in human and animal brains, examples of very large and relatively reliable systems constructed from individual components, the neurons, which would appear to be anything but reliable.

…

In communication theory this can be done by properly introduced redundancy."

Understanding noise stability for one layer

(no nonlinearity)



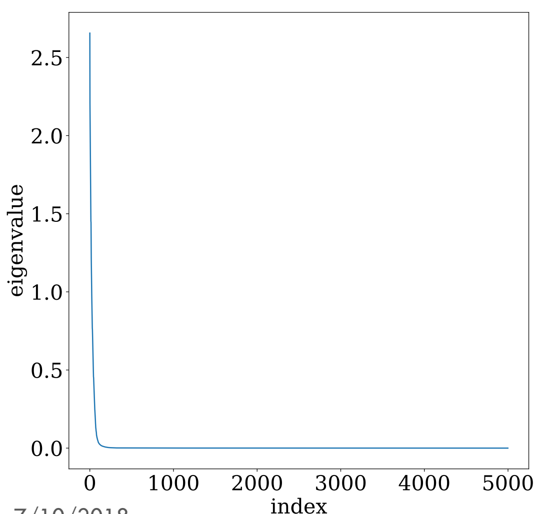
η : Gaussian noise

$$|Mx|/|x| \gg |M\eta|/|\eta|$$

$$\wedge\!| \qquad\qquad\qquad\qquad ||$$

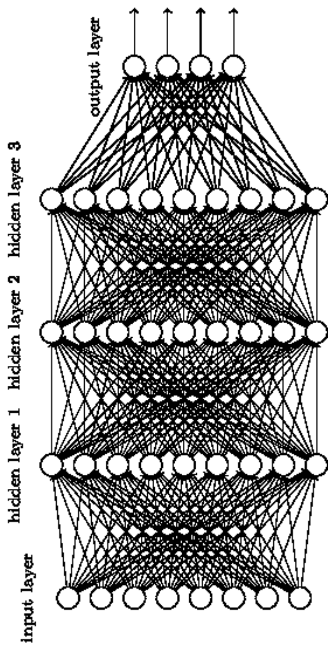
$$\sigma_{max}(M) \qquad \sum_i \sigma_i(M)^2)^{1/2}/\sqrt{n}$$

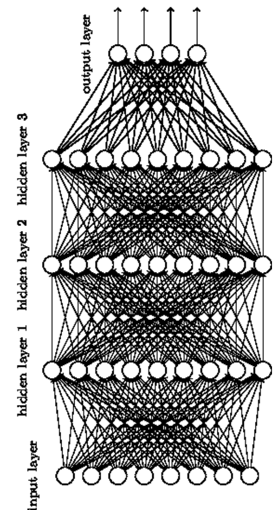
Layer Cushion = ratio
(roughly speaking..)



Distribution of singular values in a filter of layer 10 of VGG19. Such matrices are compressible…

Overview of compression-based method for generalization bounds [A.,Ge, Neyshabur, Zhang'18]; user-friendly version of PAC-Bayes





parameters $\ll$ # datapoints

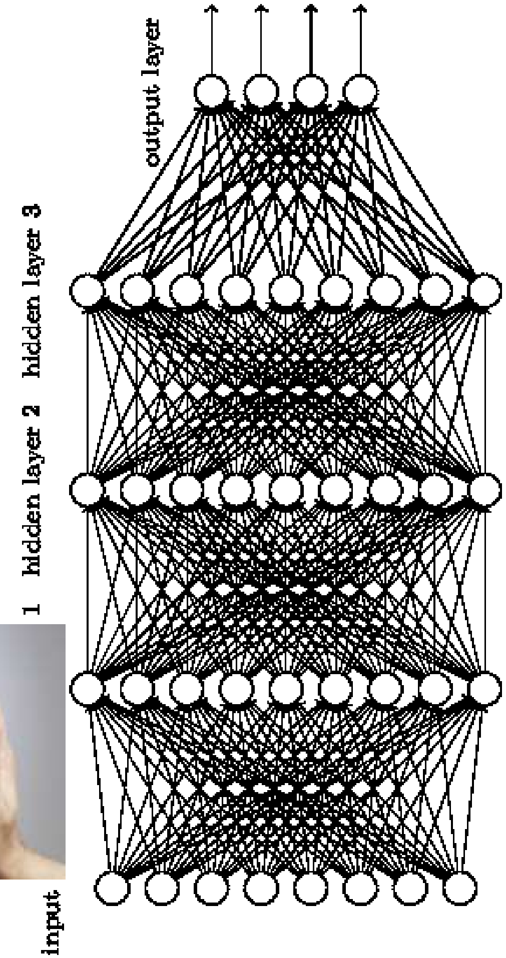
parameters $\gg$ # datapoints

Important: compression method allowed to use any number of new random bits, provided they don't depend on data.

Proof sketch : Noise stability → deep net compressible with minimal change to training error

Idea 1: Compress a layer (randomized; errors introduced are "Gaussian like")

Idea 2: Errors attenuate as they go through network, as noted earlier.



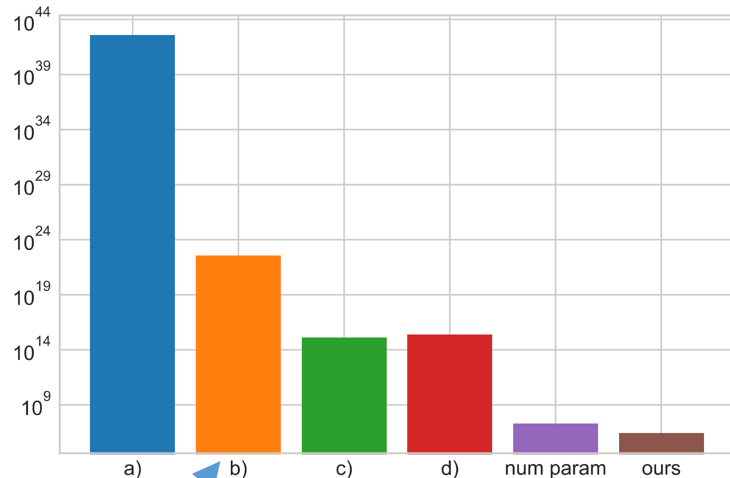
Compression:
(1) *Generate k random sign matrices $M_1, \ldots, M_k$ (impt: picked before seeing data)*
(2) $\hat{A} = \frac{1}{k}\sum_{t=1}^{k}\langle A, M_t\rangle M_t$

(Nontrivial extension to convolutional nets)

The Quantitative Bound

$$\text{capacity} \approx \left(\frac{\text{depth} \times \textit{activation contraction}}{\text{layer cushion} \times \text{interlayer cushion}}\right)^2$$

VGG19
(19 layers)



[Bartlett-Mendelson'02]

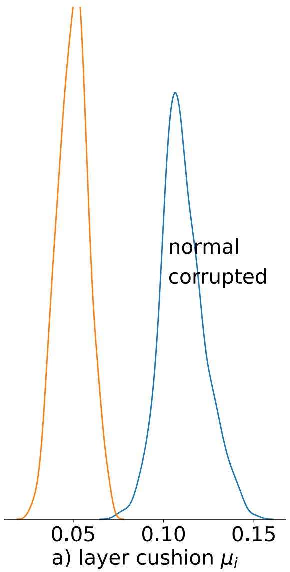
[Neyshabur et al'17]

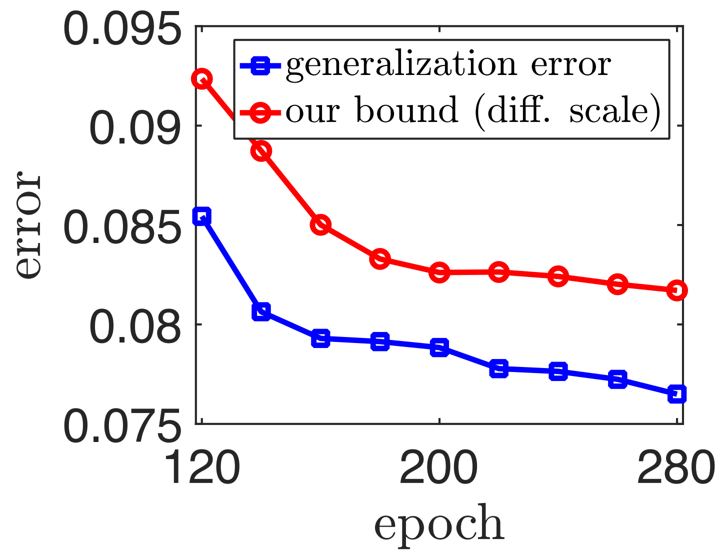
[Bartlett et al'17],

[Neyshabur et al'17]

[A., Ge, Neyshabur, Zhang'18]

Correlation with Generalization (qualitative check)





Layer cushion much higher when trained on normal data than on corrupted data

Evolution during training on normal data..

Concluding thoughts on generalization

Some progress, but final story still to be written.

I don't ultimately know why trained nets are noise stable.

Quantitative bounds too weak to explain why net with
20M parameters generalizes with 50k training datapoints.

NB: Argument needs to involve more properties of training algorithm and/or data distribution.

([Gunasekar et al'18] Quantify "Implicit bias" of gradient descent towards "low capacity" models in simple settings.)

Part 3: Role of depth

Ultimate hope: theory informs what architectures are “best” for a given task.

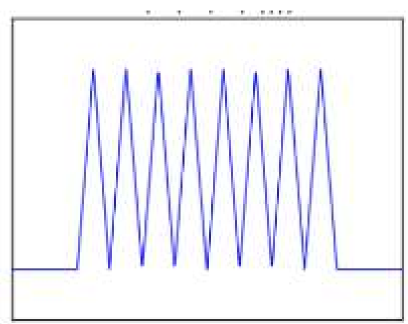
An old question: Role of depth?

Ideal result: exhibit natural learning problem which cannot be done using depth d but can be done with depth d+1

Currently, not within reach of theory, lacking mathematical formalization of "natural" learning problem…

[Eldan-Shamir'16], [Telgarsky'17]: Such results for less natural problems

Sketch: (i) Characterize max. # of "oscillations" in function computed by depth d net of some size
(ii) Show depth d+1 can compute function with more oscillations .



So does more depth help or hurt in deep learning?

Pros: Better expressiveness (as we just saw)

Cons: More difficult optimization ("vanishing/exploding gradients" unless use special architectures like resnets..)

[A., Cohen, Hazan ICML'18] Increasing depth can sometimes "accelerate" the optimization (including for classic convex problems…)

Acceleration by increasing depth: Simple example

[A., Cohen, Hazan ICML'18]

l_p regression. $$L(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\tfrac{1}{p}(\mathbf{x}^\top \mathbf{w} - y)^p\right]$$

$\mathbf{x} \in \mathbb{R}^d$ here are instances, $y \in \mathbb{R}$ are continuous labels,

Replace with depth-2 linear circuit

Replace vector w by vector w_1 multiplied by scalar w_2 (overparametrize!)

$$L(\mathbf{w}_1, w_2) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\tfrac{1}{p}(\mathbf{x}^\top \mathbf{w}_1 w_2 - y)^p\right]$$

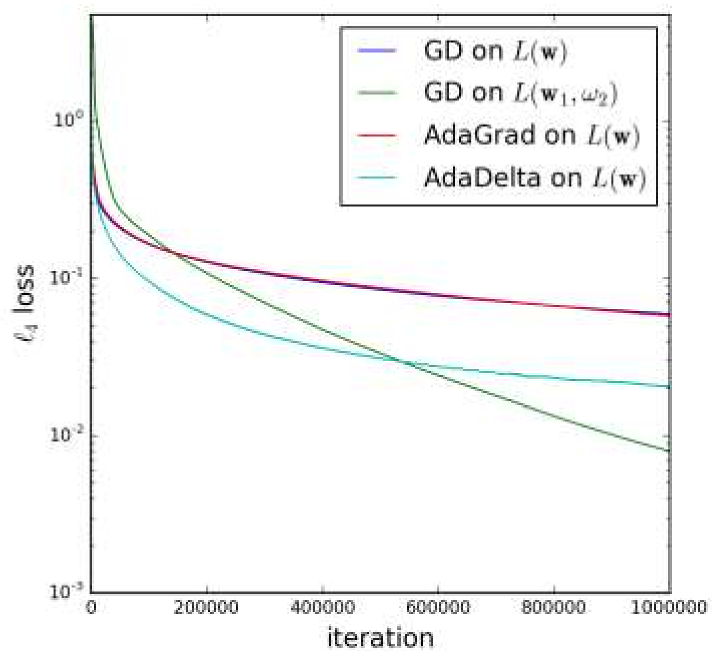
GD now amounts to $$\mathbf{w}^{(t+1)} \approx \mathbf{w}^{(t)} - \rho^{(t)}\nabla_{\mathbf{w}^{(t)}} - \sum_{\tau=1}^{t-1} \mu^{(t,\tau)}\nabla_{\mathbf{w}^{(\tau)}}$$

Adaptive learning rate + "memory" of past gradients!

Overparametrization acceleration effect

(UCI regression task…)

l_p regression, p=4

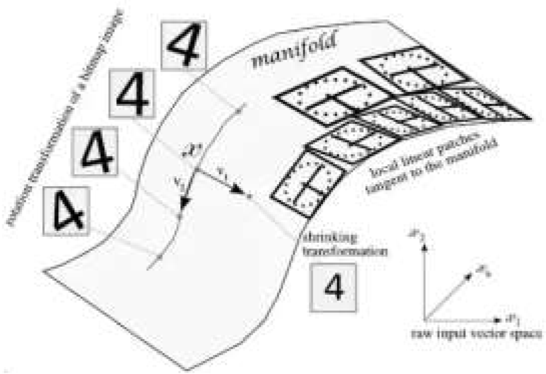


- Similar effects observed in nonlinear deep net; eg replace fully connected layer by two layers.
- Some theoretical analysis for multilayer linear nets.
- Proof that acceleration effect due to increase of depth not obtainable via any regularizer on original architecture.

Part 4: Theory for Generative Models and Generative Adversarial Nets (GANs)

Unsupervised learning motivation: “Manifold assumption”

X : Image



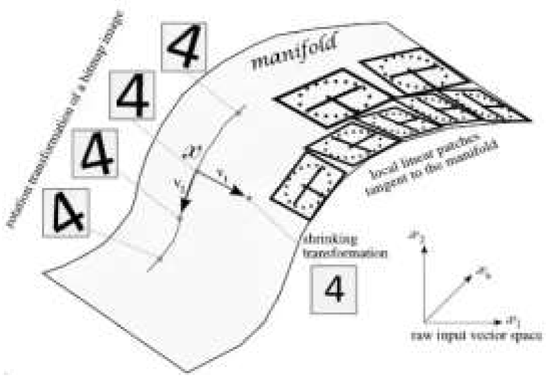
Z: Its ”code” on manifold

Goal: Using large unlabeled dataset learn the Image ➔ Code mapping

Typically modeled as learning the joint prob. density p(X, Z)
(Code of X = sample from Z|X)

Unsupervised learning Motivation: “Manifold assumption” (contd)

X : Image



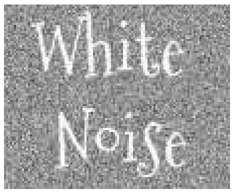
Z: Its ”code” on manifold

Hope: Code Z (= “High level Representation”) is good substitute for Image X in downstream classification tasks.

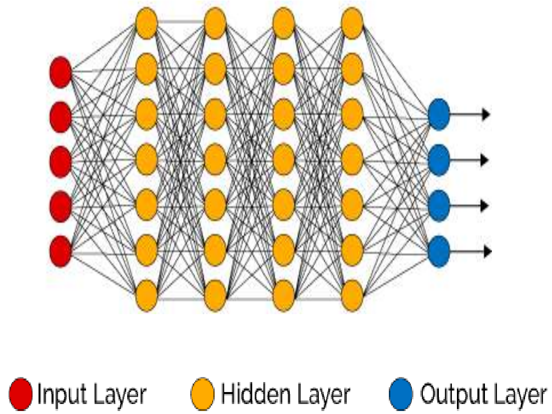
(ie solving those tasks requires fewer labeled samples if use Z instead of X)

Typically modeled as learning the joint prob. density p(X, Z)
(Code of X = sample from Z|X)

Deep generative models
(e.g., Variational AutoEncoders)



Code Z



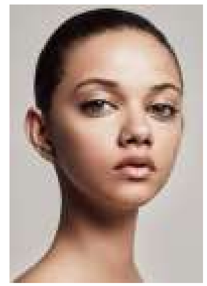
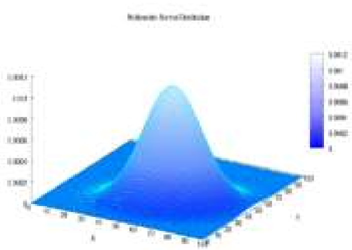


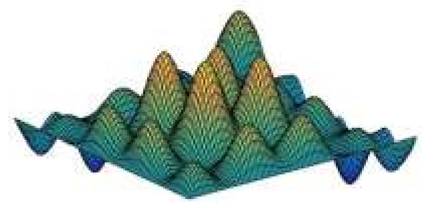
Image X

Usual training:
Max E_x[log p(x)]

("log likelihood")



N(O, I)



D_{real}

Implicit assumption: D_{real} *generatable* by deep net of *reasonable* size.

Generative Adversarial Nets (GANs) [Goodfellow et al. 2014]

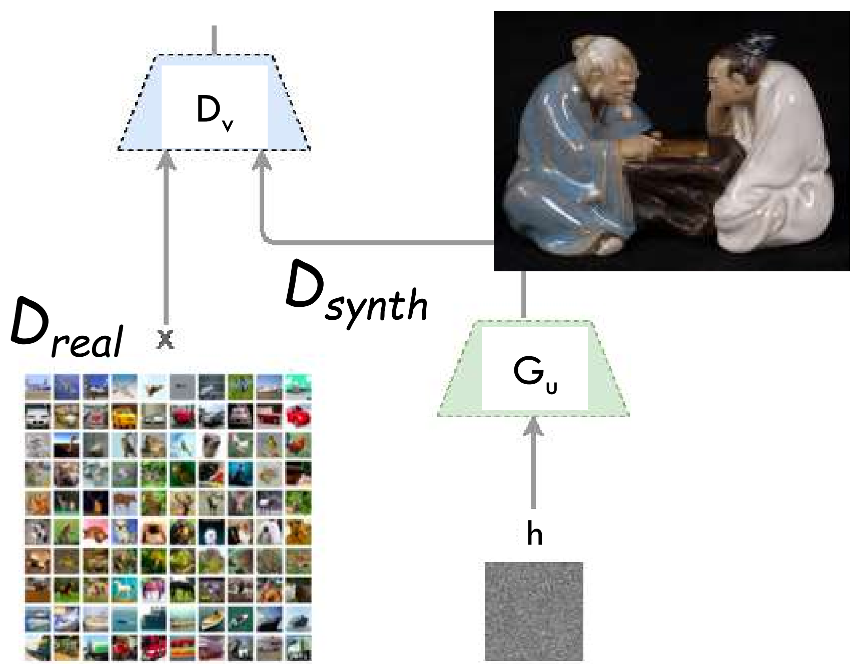
Motivations : (1) Avoid loglikelihood objective; it favors outputting fuzzy images.

(2) Instead of loglikelihood, use power of discriminative deep learning to improve the generative model.

Generative Adversarial Nets

“Difference in expected output on real vs synthetic images” Wasserstein GAN [Arjovsky et al’17] **

014]



$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \quad \mathbf{E}_{x \sim \mathcal{D}_{real}}[D_v(x)] - \mathbf{E}_h[D_v(G_u(h))].$$

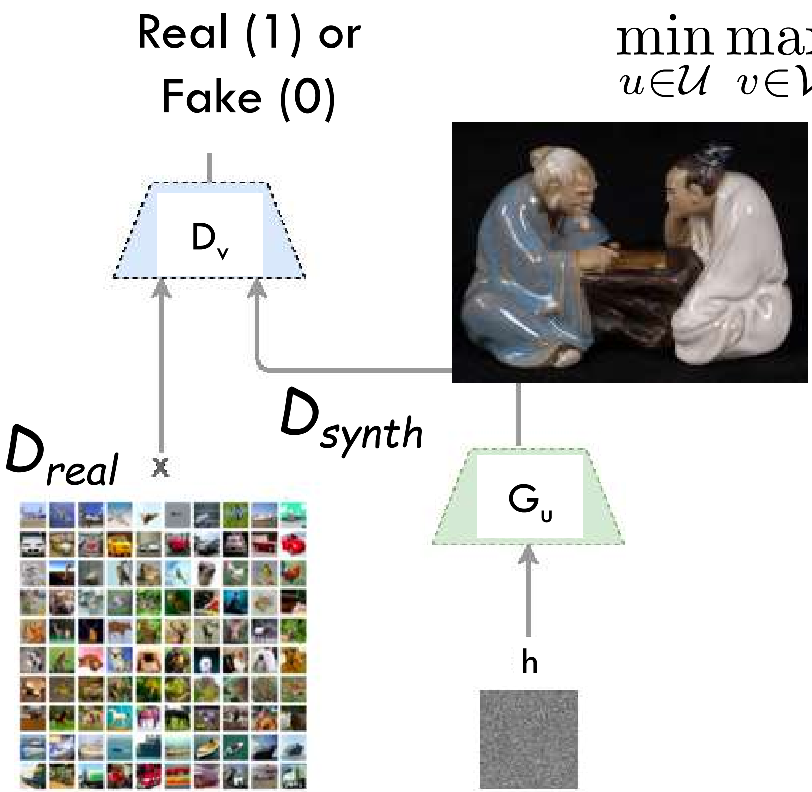
- Discriminator trained to output 1 on real inputs, and 0 on synthetic inputs.
- Generator trained to produce synthetic outputs that make discriminator output high values.

[Excellent resource: [Goodfellow’s survey]

u= trainable parameters of Generator net

v = trainable parameters of Discriminator net

Generative Adversarial Nets (GANs) [Goodfellow et al. 2014]



$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \; \mathbf{E}_{x \sim \mathcal{D}_{real}}[D_v(x)] - \mathbf{E}_h[D_v(G_u(h))].$$

- Discriminator trained to output 1 on real inputs, and 0 on synthetic inputs.
- Generator trained to produce synthetic outputs that make discriminator output high values.

Generator "wins" if objective ≈ 0 and further training of discriminator doesn't help. ("Equilibrium.")

u= trainable parameters of Generator

v = trainable parameters of Discriminator

What spoils a GANs trainer's day: Mode Collapse

- Since discriminator only learns from a few samples, it may be unable to teach generator to produce distribution D_{synth} with sufficiently large diversity

- (many ad hoc qualitative checks for mode collapse..)

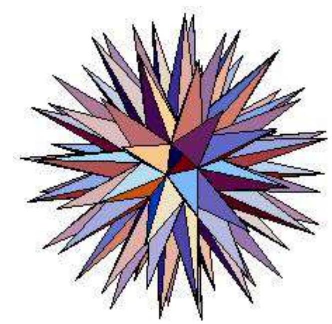
New Insight from theory: problem not with # of training samples, but size (capacity) of the discriminator!

Thm [A., Ge, Liang, Ma, Zhang ICML'17] : If discriminator size = N, then ∃ generator that generates a distribution supported on O(Nlog N) images, and still wins against all possible discriminators.
(tweaking objectives or increasing training set doesn't help..)

(NB: D_{real} presumably has infinite support..)

→ Small discriminators inherently incapable of detecting "mode collapse."

Pf sketch: Consider generator that learns to produce O(N logN) random real images. Consider "all possible discriminators of size N" (suffices to consider "ε-net"). Use concentration bounds to argue that none of them can distinguish D_{real} from this low-support distribution.



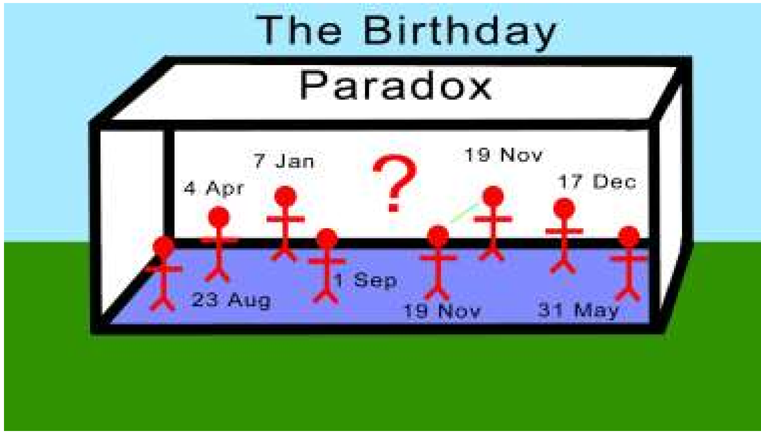
How to check support size of generator's distribution??

Theory suggests GANs training objective not guaranteed to avoid mode-collapse.

Does this happen during real life training???

Empirically detecting mode collapse (Birthday Paradox Test)

(A, Risteski, Zhang ICLR'18)



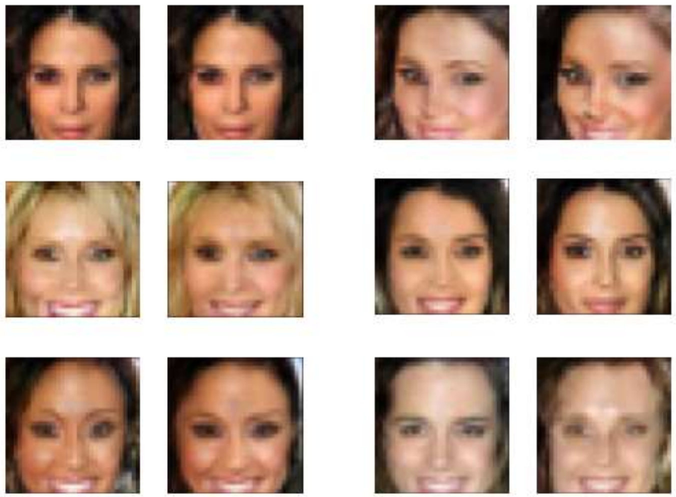
If you put 23 random people in a room, chance is > 1/2 that two of them share a birthday.

Suppose a distribution is supported on N images.
Then Pr[sample of size $\sqrt{N}$ has a duplicate image] > ½.

Birthday paradox test* [A, Risteski, Zhang] : If a sample of size s has near-duplicate images with prob. > 1/2, then distribution has only s^2 distinct images.

Implementation: Draw sample of size s; use heuristic method to flag possible near-duplicates. Rely on human in the loop verify duplicates.

Estimated support size from well-known GANs



DC-GAN [Radford et al'15]: Duplicates in 500 samples. Support size $(500)^2 = 250K$

BiGAN [Donohue et al'17]
and ALI (Dumoulin et al'17]:
Support size = $(1000)^2 = 1M$

CelebA (faces):
200k training images

(Similar results on CIFAR10)

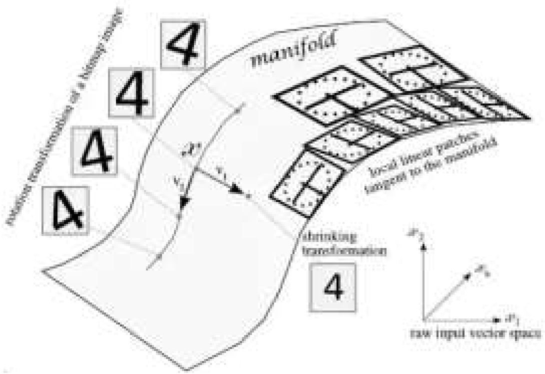
Followup: [Santurkar et al'17] Different test of diversity; confirms lack of diversity.

Part 4.1 (brief): Need to rethink unsupervised learning.

AKA “Representation Learning.”

Unsupervised learning Motivation: “Manifold assumption” (contd)

X : Image



Z: Its ”code” on manifold

Hope: Code Z is good substitute for Image X in downstream classification tasks.

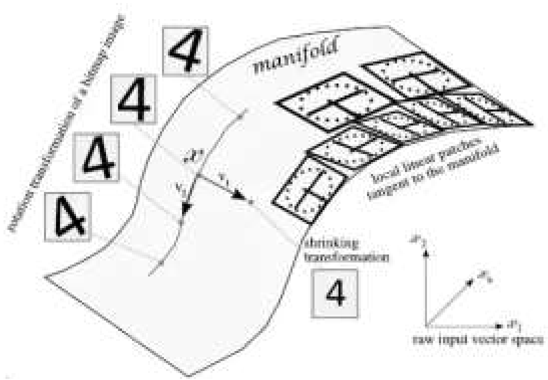
(ie solving those tasks requires fewer labeled samples if have the code)

“Semisupervised learning??”

Typically modeled as learning the joint prob. density p(X, Z)
(Code of X = sample from Z|X)

Caveat : Posssible hole in this whole story that I’m unable to resolve

X : Image



Z: Its ”code”
on manifold

Joint Density p(X,Z)

For Z to be good substitute for image X in downstream classification, density p(X,Z) needs to be learnt to very high numerical accuracy.

[See calculation in blog post
by A. + Risteski’17, www.offconvex.org]

Doesn’t mean unsupervised learning can’t happen, just that usual story doesn’t justify it.

Food for thought…

Maximizing log likelihood (presumably approximately) may lead to little usable insight into the data.

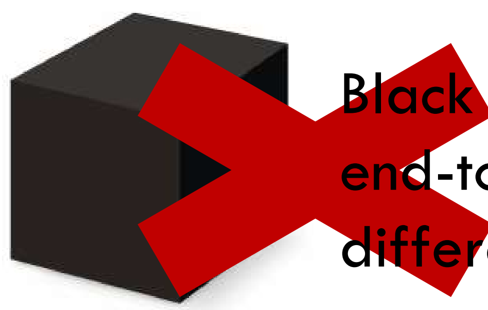
How to define utility of GANs (if not as distribution learners)?

Need to define unsupervised learning using a "utility" approach (What downstream tasks are we interested in and what info do they need about X?)

(Similar musings on INFERENCE blog, April'18.
e.g., What would a "representation learning competition" look like?)

Part 5: Deep Learning-free text embeddings (illustration of above principles)

"Hand-crafted with love and care…."



Black box that's end-to-end differentiable…

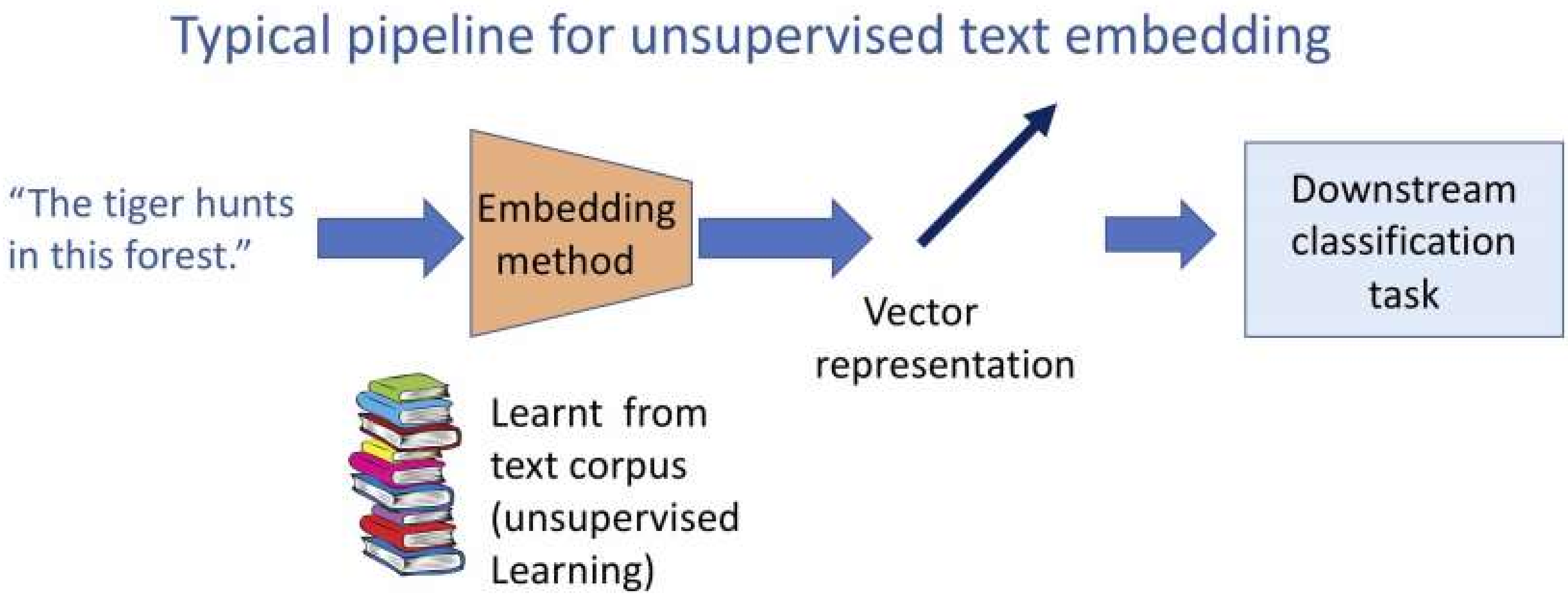
Two sentences that humans find quite similar

A lion rules the jungle.

The tiger hunts in this forest.

NB: No words in common!

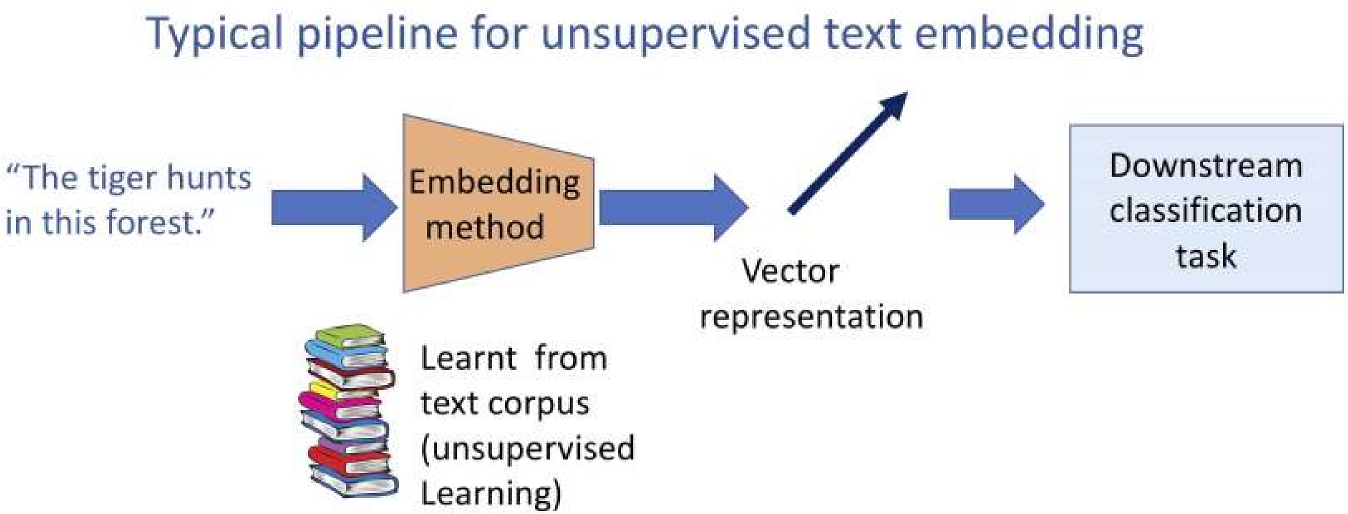
How to capture similarity and other properties of pieces of text?



Obvious inspiration: Word embeddings (word2vec, GloVe etc.)
Usual method: Recurrent neural net, LSTM (Long Short Term Memory), etc.

[Le,Mikolov'14] Paragraph vectors [Kiros et al'15]: SkipThought.

Much followup work including fastrer training/inference, or incorporating supervision (eg InferSent [Conneau et al'17]).



Learnt using LSTMs…

~~Ben Recht~~ Linearization Principle: *"Before committing to deep model figure out what the linear methods can do…."*

Cottage industry of text embeddings that're linear

"Word embeddings + linear algebra"

Simplest: Sum of word embeddings of constituent words
(Inspired by $\text{word2vec}_{\text{CBOW}}$)

(Wieting et al '16) Weighted sum; weights learnt via fit to paraphrase dataset. ("semi-supervised")

[A,Liang,Ma '17] "SIF" Smooth inverse weighted sum, followed by denoising using top singular vector….

Performance (similarity/entailment tasks)

Skip-Thought; BiLSTM based (Kiros et al'15)

Semi-supervised (Wieting et al'15)

Our best unsupervised

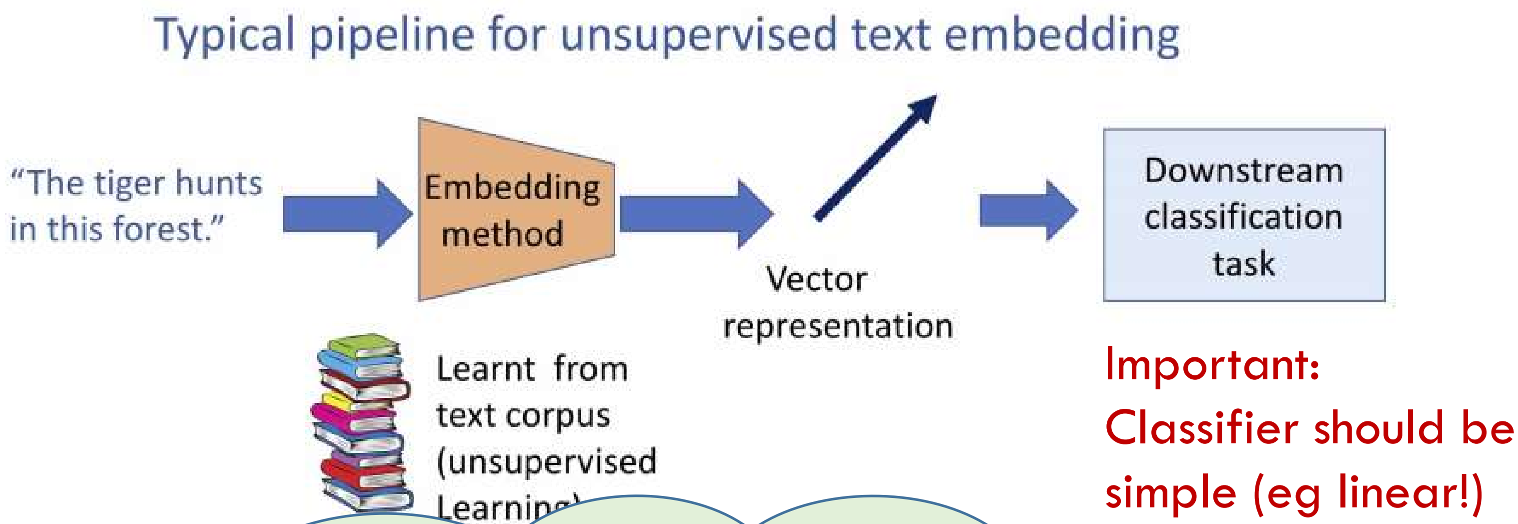
Our best semisupervised (combine w/ Wieting et al)

Tasks	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
STS'12	30.8	52.5	58.7	52.8	56.2	59.5
STS'13	24.8	42.3	52.1	46.4	56.6	**61.8**
STS'14	31.4	54.2	63.8	59.5	68.5	**73.5**
STS'15	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	24.7	30.3	33.8	36.3	48.0	49.0

SIF

Performance on standard SemEval tasks [Agirre et al.] and Twitter task[Xu et al'15]

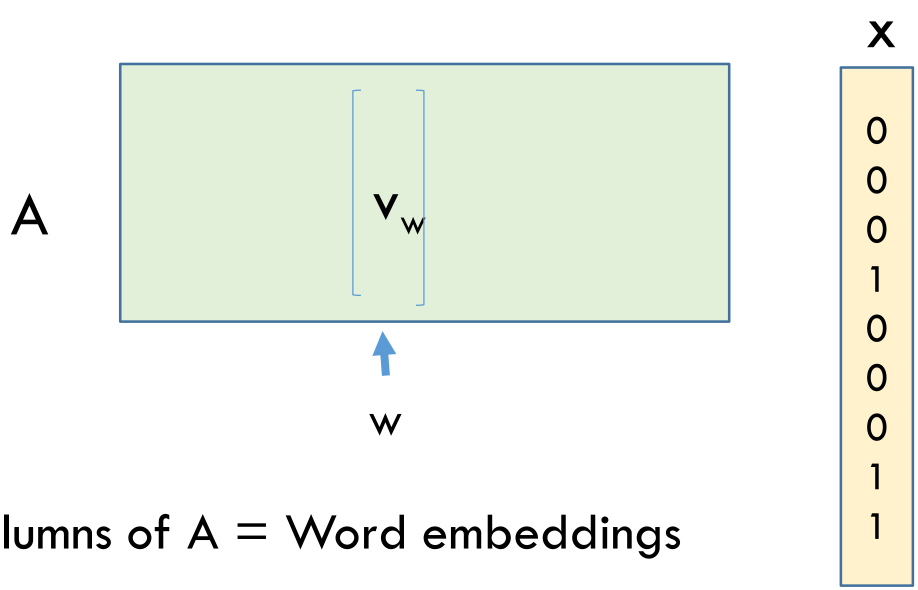
(For theory behind SIF embedding see original paper…)



Downstream task not known ahead of time! Perhaps embedding must capture all/most of the information in text? (e.g., Bag-of-Words info)

Word extraction out of linear embeddings ⇔ sparse recovery

- Recall sum-of-words embedding:

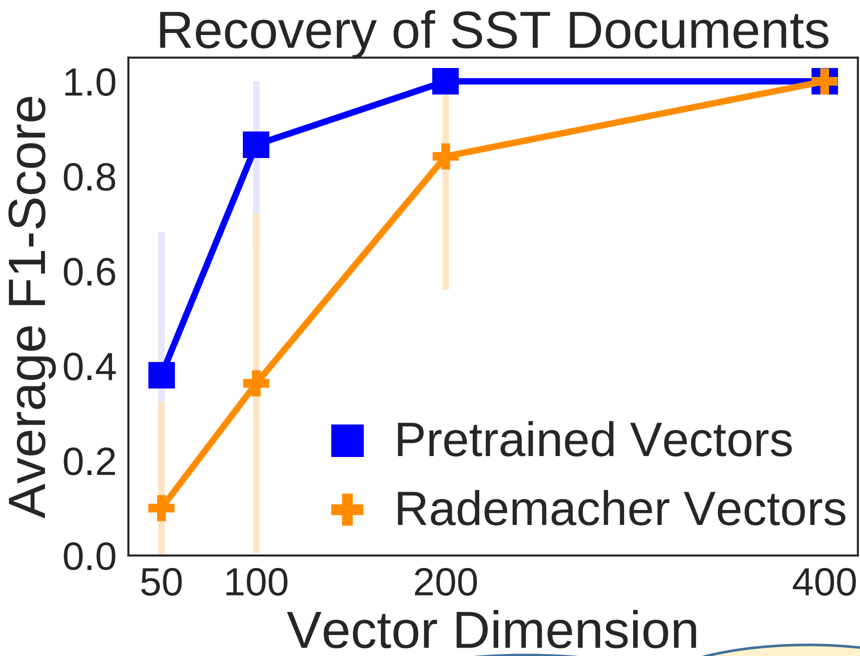


Columns of A = Word embeddings

Bag-of-word vector

Recovering Sparse x given Ax
≍ "Compressed Sensing"
(aka "Sparse recovery")
[Donoho06, Candes-Romberg-Tao06]

Do-able if A satisfies "RIP"/"Random"/"Incoherence".
(unfortunately none are satisfied by matrix of GloVe embeddings..)



(Bag-of-Words only)

Recovery algorithm: Basis Pursuit

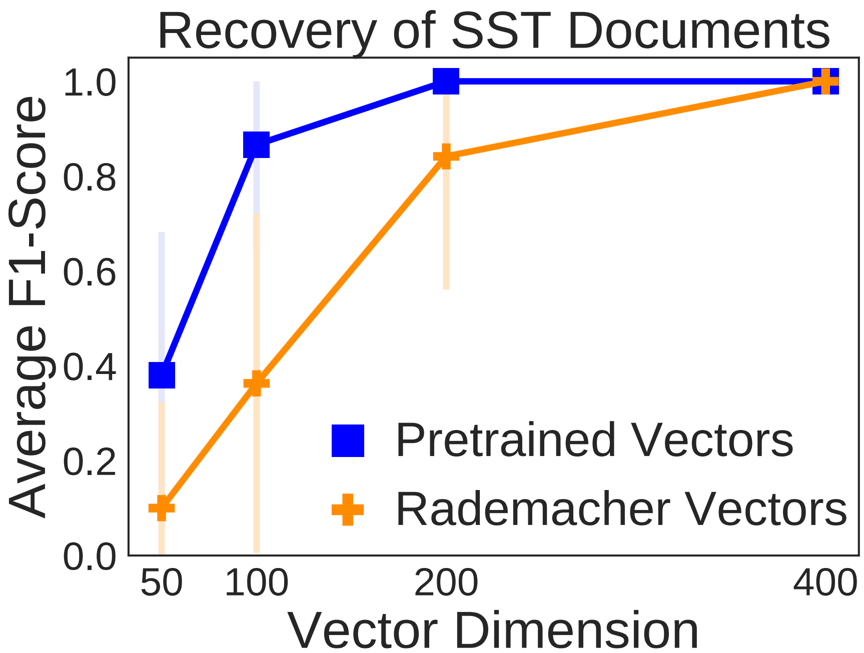
$$\min \quad |x|_1 \quad \text{s.t.} \quad Ax = b$$

Recovered BoW

Embedding

Ability to recover original words
doesn't imply the embedding is good
for classification via linear classifier....

[Rademacher = random +1/-1



Recovery algorithm: Basis Pursuit

$$\min \quad |x|_1 \quad \text{s.t.} \quad Ax = b$$

But Calderbank et al '09 showed linear classification on compressed vector Ax is essentially as good as x.

➔ Not surprising linear embeddings work well in downstream tasks! (Even provably so, under some compressed sensing type conditions.)

More powerful linear embeddings

[Some inspiration from [Plate 95] [Kanerva'09]]

Distributed Co-occurrence (DisC) Embeddings
[A, Khodak, Saunshi, Vodrahalli ICLR'18]

Use "Compressed sensing" of n-gram information
(bi-gram = word pairs, tri-gram = word triples..)
Vector for bigram (w,w') = $v_w \odot v_{w'}$ (entrywise product)

A La Carte Sentence Embeddings
[Khodak, Saunshi, Liang, Ma, Stewart, A. ACL'18]

Modify DisC by using "induced n-gram embeddings", created using linear regression. (Induced embeddings useful in other NLP tasks too!)

Comparison with state-of-the art text embeddings on suite of downstream classification tasks

Representation	n	d^*	MR	CR	SUBJ	MPQA	TREC	SST (± 1)	SST	IMDB
Bag of n-gram	1	V_1	77.1	77.0	91.0	85.1	86.8	80.7	36.8	88.3
	2	$V_1 + V_2$	77.8	78.1	91.8	85.8	90.0	80.9	39.0	90.0
	3	$V_1 + V_2 + V_3$	77.8	78.3	91.4	85.6	89.8	80.1	42.3	89.8
à la carte	1	1600	79.8	81.3	92.6	87.4	85.6	84.1	46.7	89.0
	2	3200	81.3	83.7	93.5	87.6	89.0	85.8	**47.8**	**90.3**
	3	4800	**81.8**	**84.3**	93.8	87.6	89.0	**86.7**	<u>**48.1**</u>	<u>**90.9**</u>
Sent2Vec[1]	1-2	700	76.3	79.1	91.2	87.2	85.8	80.2	31.0	85.5
DisC[2]	2-3	3200-4800	80.1	81.5	92.6	87.9	90.0	85.5	46.7	89.6
skip-thoughts[3]		4800	80.3	83.8	**94.2**	**88.9**	<u>**93.0**</u>	85.1	45.8	
SDAE[4]		2400	74.6	78.0	90.8	86.9	78.4			
CNN-LSTM[5]		4800	77.8	82.0	93.6	**89.4**	**92.6**			
MC-QT[6]		4800	<u>**82.4**</u>	<u>**86.0**</u>	<u>**94.8**</u>	<u>**90.2**</u>	**92.4**	<u>**87.6**</u>		

Logeswaran and Lee 2018

Open: Hand-crafted analogs of “Attention” and “Character-level LSTMs”?

Aside: Another illustration of Linearization principle in RL [Mania,Guy,Recht'18]

- Linear Quadratic Regulator (old model from control theory) plus simple Random Search beats state of the art deepRL on some standard RL tasks

		Maximum average reward		
Task	**RS**	**NG-lin**	**NG-rbf**	**TRPO-nn**
Swimmer-v1	365	366	365	131
Hopper-v1	3909	3651	3810	3668
HalfCheetah-v1	6722	4149	6620	4800
Walker	11389	5234	5867	5594
Ant	5146	4607	4816	5007
Humanoid	11600	6440	6849	6482

See Recht's talk

Wrapping up…

What to work on (suggestions for theorists)

1. Use Physics/PDE insights, such as calculus of variations (Lagrangians, Hamiltonians, etc.)

2. Look at unsupervised learning (Yes, everything is NP-hard and new but that's how theory grows.)

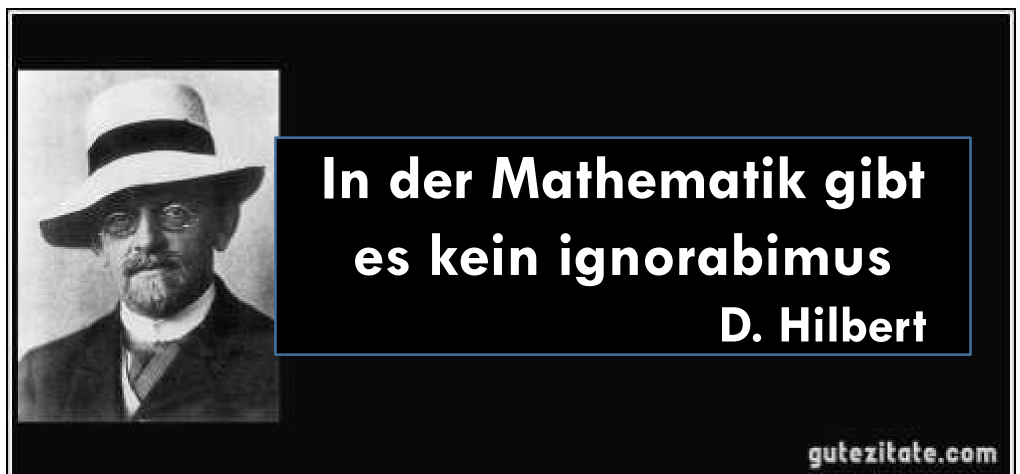
3. Theory for Deep Reinforcement learning. (Currently very little.)

4. Going beyond 3), design interesting models for interactive learning (of language, skills, etc.). Both theory and applied work here seems to be missing some basic idea. (Theory focuses on simple settings like linear classifiers/clustering.)

Concluding thoughts

- Deep learning is a new frontier for theory; many new avenues.
- Best theory will emerge from engaging with real data and real deep net training. (Nonconvexity and attendant complexity seems to make armchair theory less fruitful.)
- I am optimistic that deep learning methods can be understood and simplified.

THANK YOU!!



Advertisements

http://unsupervised.cs.princeton.edu/deeplearningtutorial.html

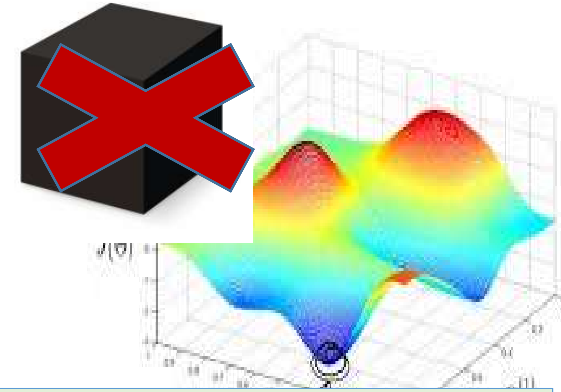
Come join special year at Institute for Advanced Study 2019-20 http://www.math.ias.edu/sp/

Resources: www.offconvex.org

Grad seminar (hope to put all notes online soon) http://www.cs.princeton.edu/courses/archive/fall17/cos597A/

Extra slides..

Convergence to global optimum: Measure of progress



Trying to reach global optimum z^*

update: $z^{s+1} = z^s - \eta\, g^s$

Need NOT be gradient!
Also, starting point special

Show: direction $-g^s$ is substantially correlated with best direction one could take, namely, $z^s - z^*$

Definition: g^s is $(\alpha, \beta, \varepsilon_s)$-correlated with z^* if for all s:

$$< g^s, z^s - z^* > \quad \geq \quad \alpha|z^s - z^*|^2 + \beta|g_s|^2 - \epsilon_s$$

("almost-convex"; generalizes several previous progress measures. Simple proof shows it implies quick convergence.)