Machine Learning: Generalization 4

Hao Tang

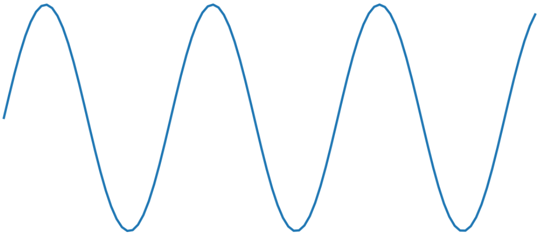
March 25, 2026

Computational and statistical thinking

Computational	Statistical
Runtime	Samples
How many steps do we need?	How many samples do we need?
Polynomial number of steps	Polynomial number of samples

Statistical properties of neural networks

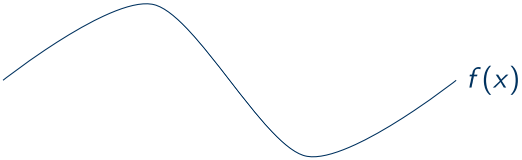
- What is the VC dimension of a sine function?

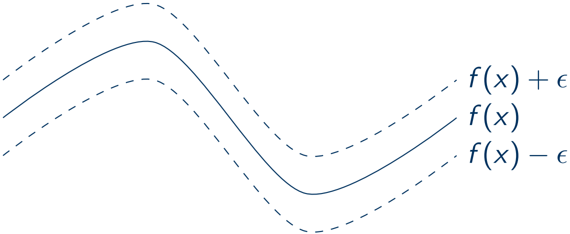


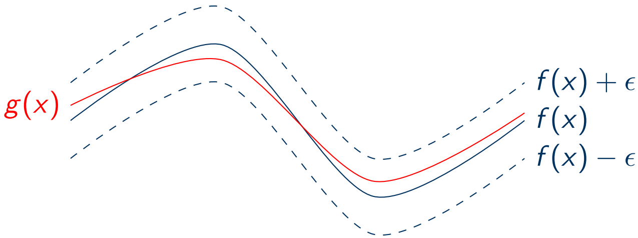
- What happens if we train a neural network to approximate a sine function?

- **Theorem (Universal approximation).** For every $\epsilon > 0$, given any Lipschitz function $f : [-1, 1]^d \to [-1, 1]$, there is a network g such that $|g(x) - f(x)| \leq \epsilon$ for any x.

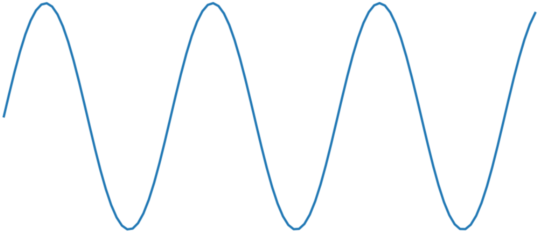
- The number of nodes needed to achieve this is $O(2^d)$.


$f(x)$


$f(x) + \epsilon$
$f(x)$
$f(x) - \epsilon$


$g(x)$
$f(x) + \epsilon$
$f(x)$
$f(x) - \epsilon$

- Can we approximate a sine function?



- The set of sine functions has an infinite VC dimension, but the set of neural networks has a VC dimension of $O(E \log E)$, where E is the number of edges in the network.
- Is there a contradiction?

- Polynomials are universal approximators.
- Decision trees are universal approximators.
- Gaussian mixture models are universal approximators.
- Universal approximation does not explain why neural networks are so “special.”

- There exists functions which can be approximated with small depth 3 networks, but cannot be approximated with depth 2 networks without using $O(2^d)$ nodes.
- This is known as depth separation of 2-layer and 3-layer neural networks.
- However, functions to show these results tend to oscillate a lot.
- Some believe the results are pathological and do not happen in practice.

Computational properties of neural networks

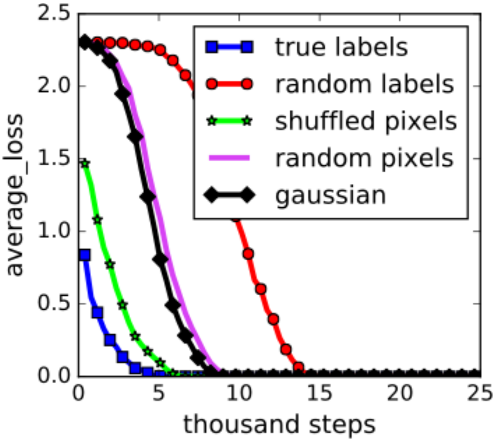
- What can be implemented with polynomial number of of nodes?
- Any Turing machine that runs in T operations can be implemented with a neural network of depth $O(T)$ with a total $O(T^2)$ nodes.
- The VC dimension of neural networks is $O(E \log E)$, where E is the number of edges in the network.
- Why do we need anything other than neural networks?

- Training a 2-layer 3-node neural network, i.e., doing ERM, is NP-complete.

- Training a 2-layer 3-node neural network, i.e., doing ERM, is NP-complete.
- The proof converts instances of an NP-complete problem into data points.
- If we can minimize the loss of the training set, we solve the NP-complete problem.

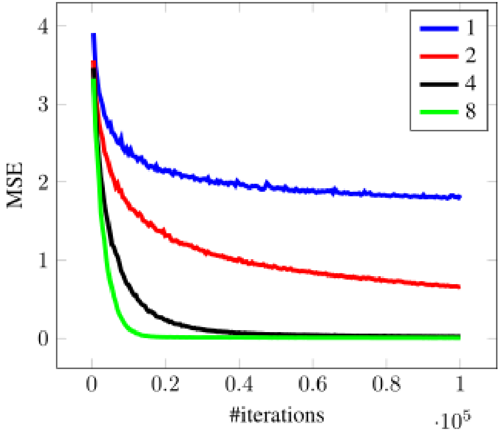
- Training a 2-layer 3-node neural network, i.e., doing ERM, is NP-complete.
- The proof converts instances of an NP-complete problem into data points.
- If we can minimize the loss of the training set, we solve the NP-complete problem.
- Maybe we don't need to solve this exactly?

- Approximating ERM is NP hard.
- The loss is not necessarily convex.
- ERM is hard for neural networks.



(Zhang *et al.*, 2017)

Overparametrization

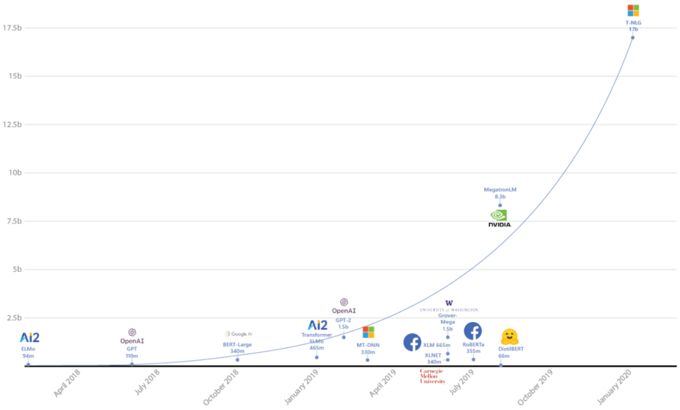


(Livni *et al.*, 2014)

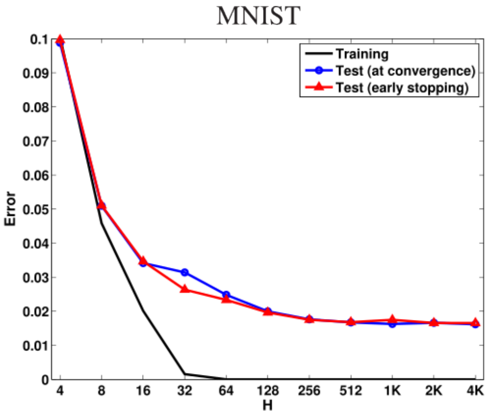
- Overparameterization means using a lot more nodes than the number of points.

- Overparameterization helps optimization.

- Overparameterization means using a lot more nodes than the number of points.
- Overparameterization helps optimization.
- Wouldn't the model just memorize the training set?
- Wouldn't the hypothesis class be too large to have good generalization error?


17.5b
15b
12.5b
10b
7.5b
5b
2.5b
ELMo 94m
GPT 110m
BERT-Large 340m
Transformer ELMo 465m
GPT-2 1.5b
MT-DNN 330m
XLM 665m
XLNET 340m
Grover-Mega 1.5b
RoBERTa 355m
DistilBERT 66m
MegatronLM 8.3b
T-NLG 17b
April 2018
July 2018
October 2018
January 2019
April 2019
July 2019
October 2019
January 2020

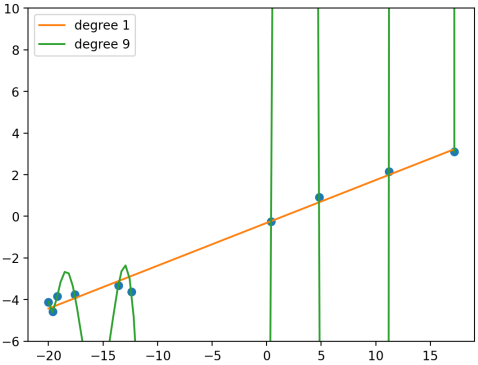
(Rosset, 2020)



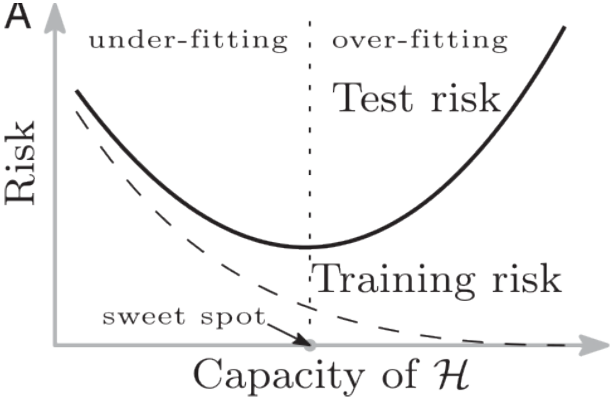
(Neyshabur *et al.*, 2014)

Interpolation and double descent

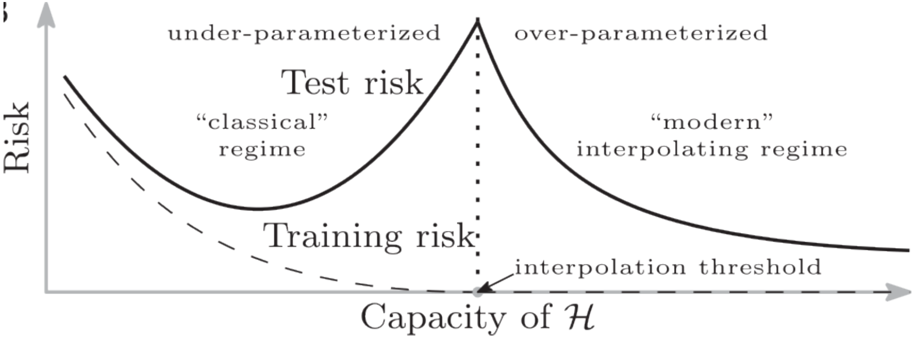
- Fitting a data set to training error zero is called interpolation.
- Why doesn't interpolation overfit?



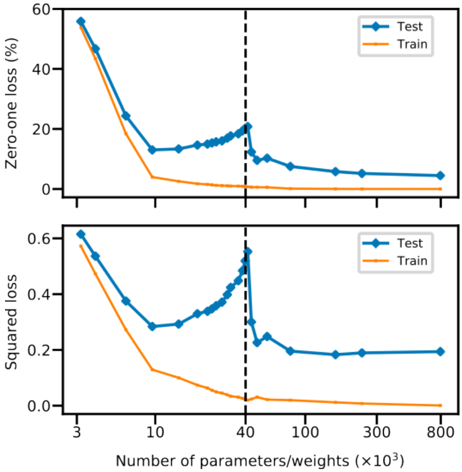
degree 1
degree 9
10
8
6
4
2
0
−2
−4
−6
−20
−15
−10
−5
0
5
10
15



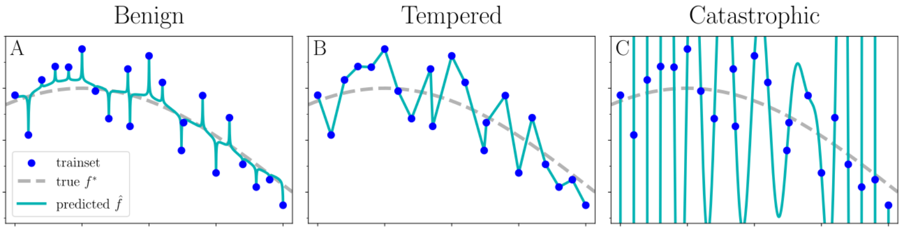
(Belkin *et al.*, 2019)



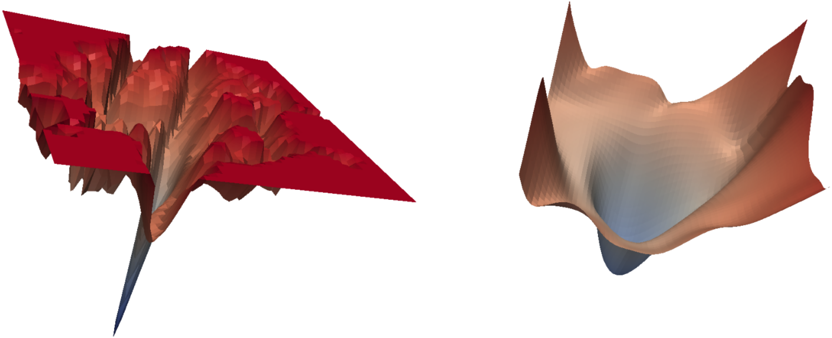
(Belkin *et al.*, 2019)



(Belkin *et al.*, 2019)



(Mallinar *et al.*, 2022)



(Foret *et al.*, 2021)

Back to basics

- Statistical learning theory tells us to minimize the training error while balancing the model capacity.
- What are the capacities we can control?

- Explicitly choosing the size of the hypothesis class

- Explicitly choosing the size of the hypothesis class
- Minimizing some measure of capacity, such as the norm

- Explicitly choosing the size of the hypothesis class
- Minimizing some measure of capacity, such as the norm
- Limiting the number of gradient updates

- Explicitly choosing the size of the hypothesis class
- Minimizing some measure of capacity, such as the norm
- Limiting the number of gradient updates
- Using a small step size (i.e., learning rate)

- Explicitly choosing the size of the hypothesis class
- Minimizing some measure of capacity, such as the norm
- Limiting the number of gradient updates
- Using a small step size (i.e., learning rate)
- Achieving stability, large margin, flat minima

In practice

- Always start with the training error.
- Always start with ERM.
- Why is the training error not close to zero?
- Regularize

Reference

- Zhang et al., Understanding deep learning requires rethinking generalization, 2017
- Livni et al., On the computational efficiency of training neural networks, 2014
- Rosset, Turing-NLG: A 17-billion-parameter language model by Microsoft, 2020
- Neyshabur et al., In search of the real inductive bias: On the role of implicit regularization in deep learning, 2014
- Belkin et al., Reconciling modern machine-learning practice and the classical bias–variance trade-off, 2019
- Mallinar et al., Benign, tempered, or catastrophic: Toward a refined taxonomy of overfitting, 2022
- Foret et al., Sharpness-aware minimization for efficiently improving generalization, 2021