Advanced Topics in Deep Learning

Jesús Fernández-Villaverde[1] and Galo Nuño[2]

August 7, 2023

[1]University of Pennsylvania

[2]Banco de España

The double descent phenomenon

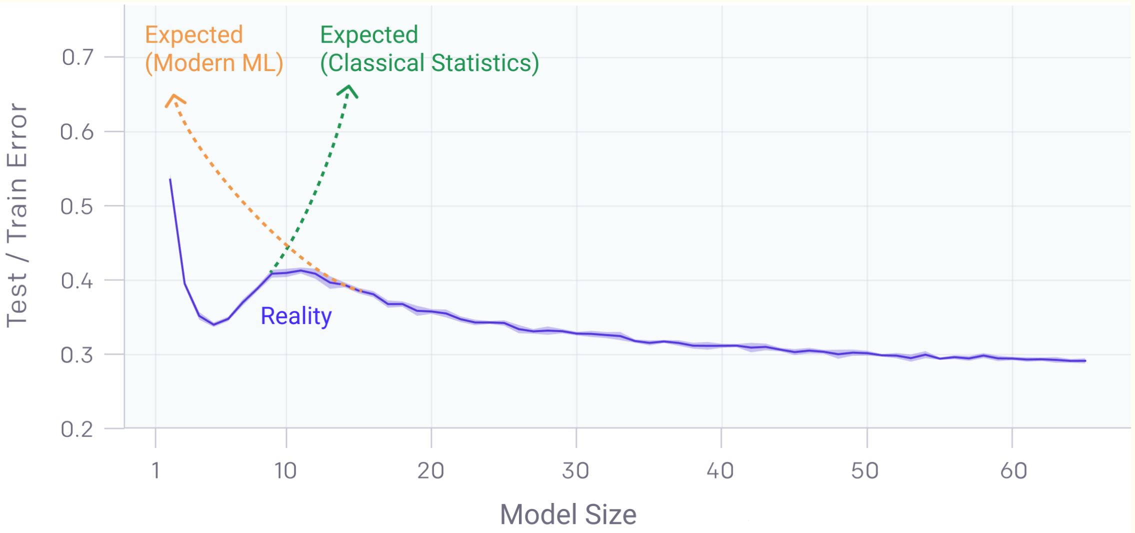
Generalization vs. overfitting

The classical view: Enrico Fermi, 1953

I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.

The modern view: Ruslan Salakhutdinov, 2017

The best way to solve the problem from practical standpoint is you build a very big system. If you remove any of these regularizations like dropout or L2, basically you want to make sure you hit the zero training error. Because if you don't, you somehow waste the capacity of the model.


Expected (Modern ML)
Expected (Classical Statistics)
Reality
0.7
0.6
0.5
0.4
0.3
0.2
Test / Train Error
1
10
20
30
40
50
60
Model Size

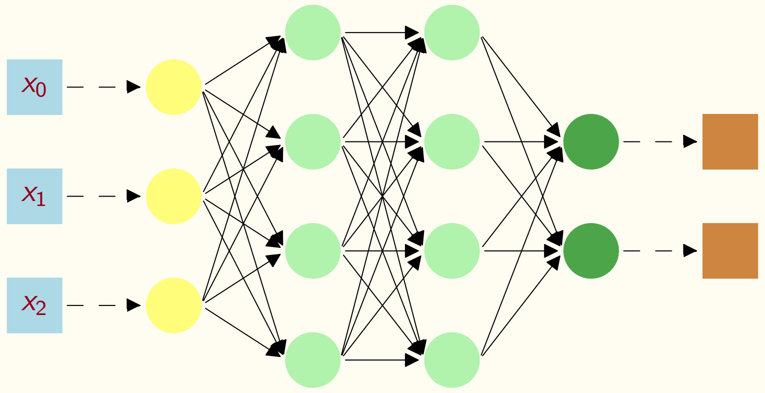
Some recent results

- Bubeck and Sellke (2021).
- They prove that for a broad class of data distributions and model classes, overparametrization is necessary if one wants to interpolate the data smoothly.
- Intuition: a tradeoff between the size of a model and its robustness.

Alternative architectures

Alternative architectures

- We have studied the basic, densely connected deep neural network.
- However, there is a rich set of alternative architectures.
- Particularly active area of research with important recent breakthroughs.

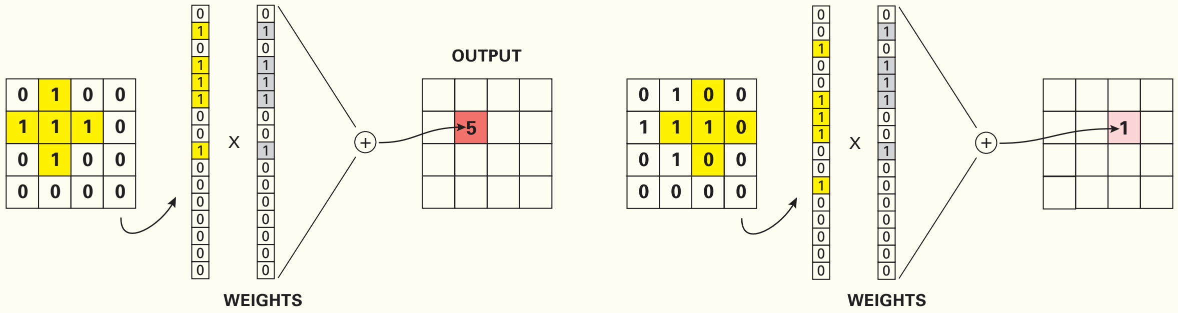


Input Values　　Hidden Layer 1　　Output Layer

Input Layer　　Hidden Layer 2

Convolutional neural networks, I

- Regular deep neural networks do not work well with a grid-like topology (e.g., 2-D images) because:
    1. Large number of weights to train.
    2. The method is not translation invariant because the weights are not shared across locations.
- Natural analog in econometrics: spatial correlations.
- Alternative: deal with specialized networks explicitly designed for these topologies.
- Most popular approach: convolutional neural networks.

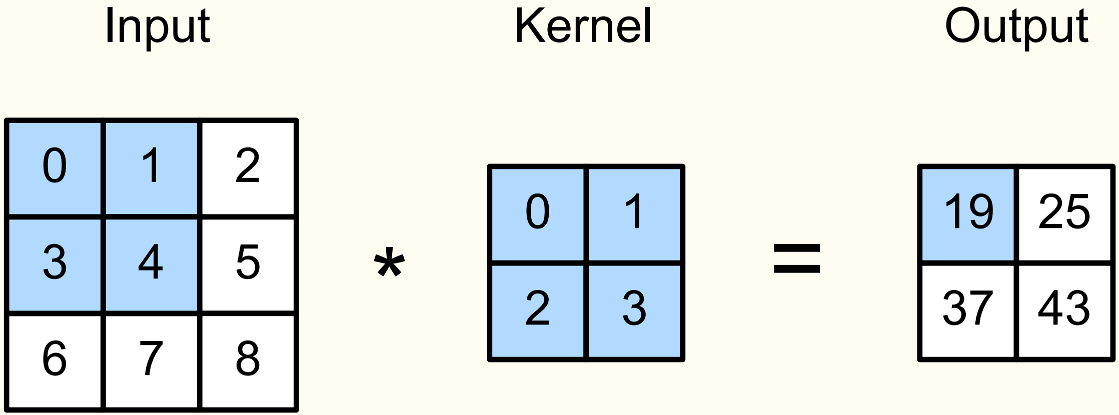

0 1 0 0
1 1 1 0
0 1 0 0
0 0 0 0
X
WEIGHTS
OUTPUT
5
0 1 0 0
1 1 1 0
0 1 0 0
0 0 0 0
X
WEIGHTS
1

Convolutional neural networks, II

- We reduce parameters by introducing a convolution ("filter"):

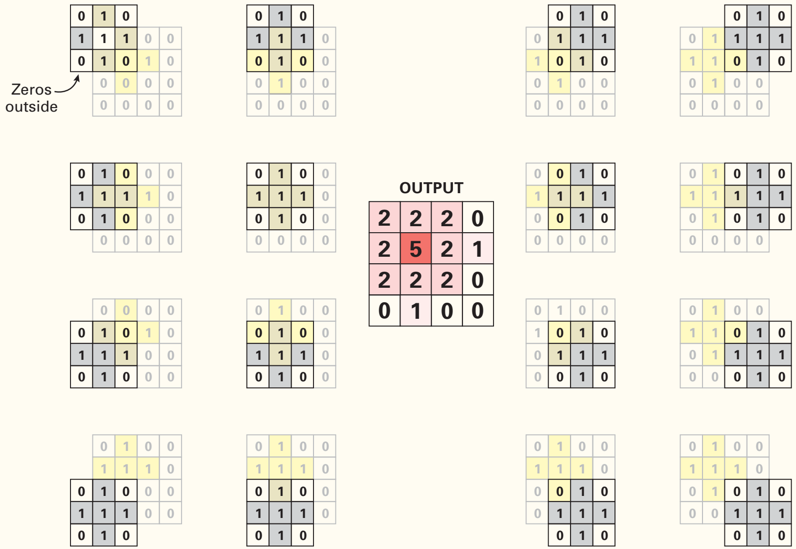
$$[f \circledast g](z) = \int f(u)g(z-u)du$$

- Very similar in spirit to a simple Kalman filter.
- Three components: Input, a kernel, and and output (or "feature map").
- We train the kernel instead of the whole set of connections.
- We can have different kernels and layer them.

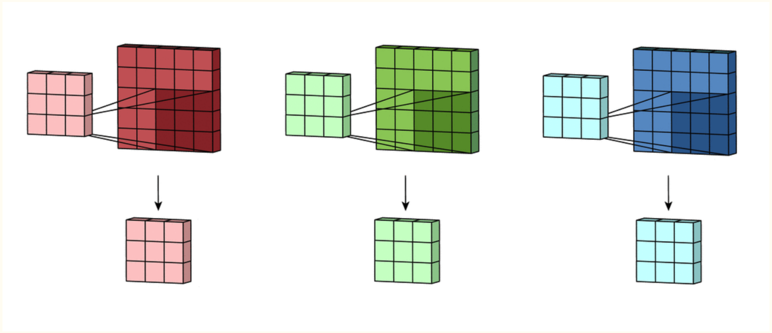

Input
0
1
2
3
4
5
6
7
8
*
Kernel
0
1
2
3
=
Output
19
25
37
43

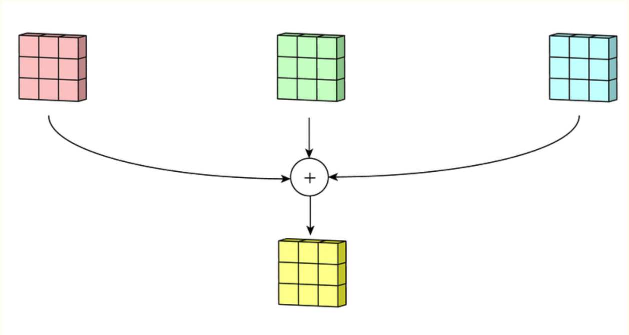
Convolutional neural networks, III

- Padding at boundaries to ensures the output is the same size as the input.
- Maxpooling as an extreme case of convolution.
- Notice how attractive convolutional neural networks are for computation with a GPU.



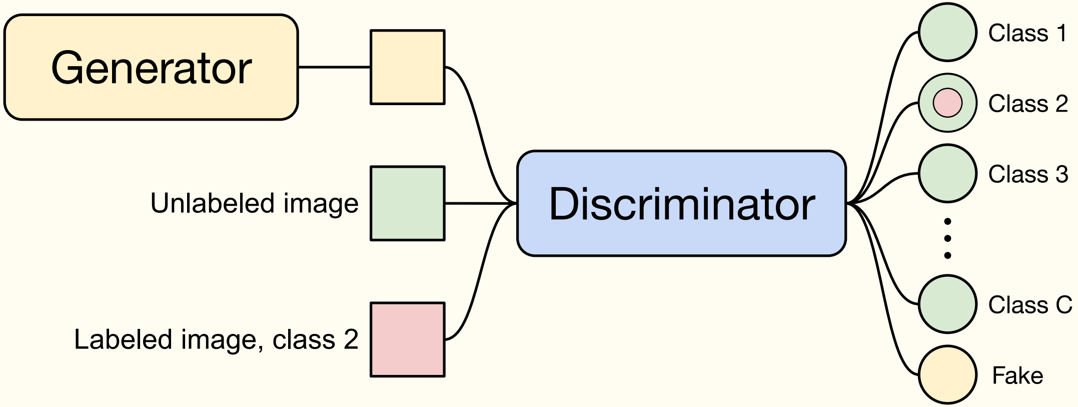
2	2	2	0
2	5	2	1
2	2	2	0
0	1	0	0





Generative adversarial networks, I

- Goodfellow et al. (2014).
- A generator network generates candidates (e.g., images).
- A discriminative network evaluates whether they are real or generated.
- Both networks play a minimax problem.
- Applications in economics: Athey et al. (2019) and Liang (2020).


Generator
Unlabeled image
Labeled image, class 2
Discriminator
Class 1
Class 2
Class 3
Class C
Fake

Generative adversarial networks, II

- Looking for G^* such that:

$$G^* = \arg\min_G \max_D V(D, G)$$

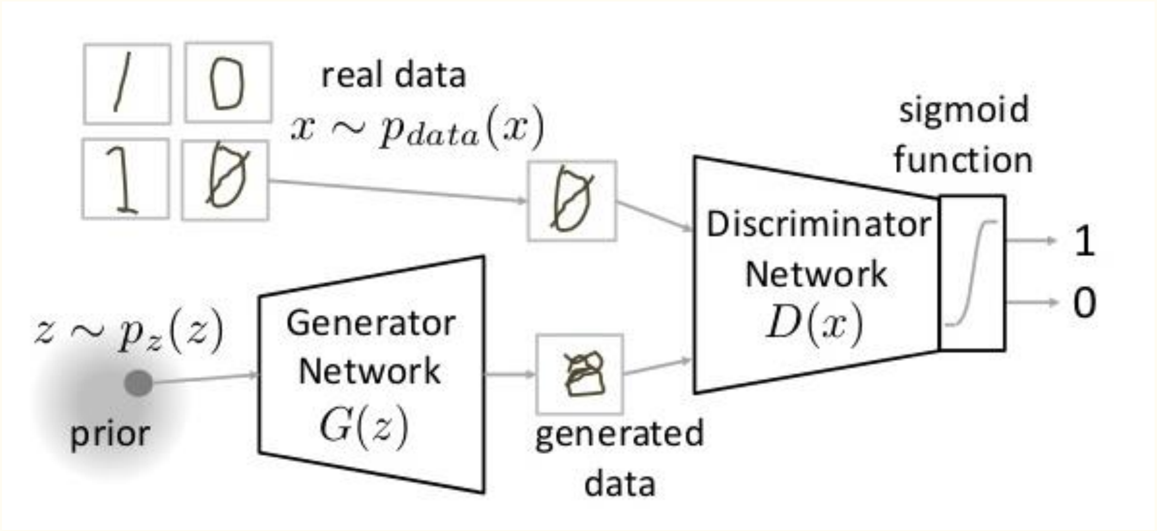
where

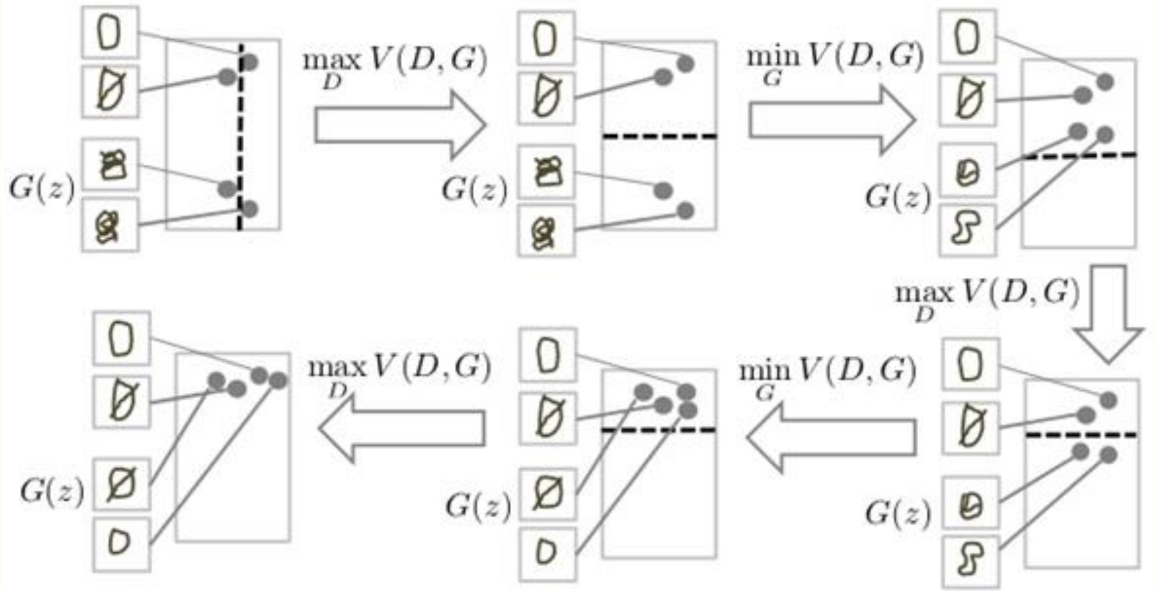
$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))]$$

Generative adversarial networks, III

- Then:

$$
\begin{aligned}
V(D,G) &= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1-\log D(G(z))] \\
&= \int_x p_{\text{data}}(x)\log(D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz \\
x = G(z) &\Rightarrow z = G^{-1}(x) \Rightarrow dz = \left(G^{-1}\right)'(x)dx \\
&\Rightarrow p_g(x) = p_z\left(G^{-1}(x)\right)\left(G^{-1}\right)'(x) \\
\int_x p_{\text{data}}(x)\log(D(x))dx &+ \int_x p_z\left(G^{-1}(x)\right)\log(1-D(x))\left(G^{-1}\right)'(x)dx \\
&= \int_x p_{\text{data}}(x)\log(D(x))dx + \int_x p_g(x)\log(1-D(x))dx \\
&= \int_x p_{\text{data}}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx
\end{aligned}
$$


real data
$x \sim p_{data}(x)$
sigmoid function
Discriminator Network
$D(x)$
1
0
$z \sim p_z(z)$
prior
Generator Network
$G(z)$
generated data


$\max_D V(D,G)$
$\min_G V(D,G)$
$\max_D V(D,G)$
$\min_G V(D,G)$
$\max_D V(D,G)$
$G(z)$

Understanding the objective function, I

- Notice:

$$\max_D V(D,G) = \max_D \int_x p_{\text{data}}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx$$

$$\frac{\partial}{\partial D(x)}\left(p_{\text{data}}(x)\log(D(x)) + p_g(x)\log(1-D(x))\right) = 0$$

$$\Rightarrow \frac{p_{\text{data}}(x)}{D(x)} - \frac{p_g(x)}{1-D(x)} = 0$$

$$\Rightarrow D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

- If the discriminator is optimal $D_G^*(x)$, the optimal generator makes:

$$p_{\text{data}}(x) = p_g(x) \Rightarrow D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

Understanding the objective function, II

- Now, let us work with $C(G)$:

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \max_D \int_x p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \\
&= \int_x p_{\text{data}}(x) \log\left(D_G^*(x)\right) + p_g(x) \log\left(1 - D_G^*(x)\right) dx \\
&= \int_x p_{\text{data}}(x) \log\left(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}\right) + p_g(x) \log\left(\frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}\right) dx \\
&= \int_x p_{\text{data}}(x) \log\left(\frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_g(x)}{2}}\right) + p_g(x) \log\left(\frac{p_g(x)}{\frac{p_{\text{data}}(x) + p_g(x)}{2}}\right) dx - \log(4) \\
&= KL\left[p_{\text{data}}(x) \| \frac{p_{\text{data}}(x) + p_g(x)}{2}\right] + KL\left[p_g(x) \| \frac{p_{\text{data}}(x) + p_g(x)}{2}\right] - \log(4)
\end{aligned}
$$

Understanding the objective function, III

- Working with this expression:

$$C(G) = KL\left[p_{\text{data}}(x) \| \frac{p_{\text{data}}(x) + p_g(x)}{2}\right] + KL\left[p_g(x) \| \frac{p_{\text{data}}(x) + p_g(x)}{2}\right] - \log(4) \geq 0$$

$$\min_G C(G) = 0 + 0 - \log(4) = -\log(4)$$

$$KL\left[p_{\text{data}}(x) \| \frac{p_{\text{data}}(x) + p_g(x)}{2}\right] = 0$$

$$\text{when } p_{\text{data}}(x) = \frac{p_{\text{data}}(x) + p_g(x)}{2}$$

$$\Rightarrow p_{\text{data}}(x) = p_g(x)$$

KL (Kullback-Leibler) divergence

- Jensen-Shannon divergency (symmetric KL):

$$JSD(P\|Q) = \frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M)$$
$$M = \frac{1}{2}(P + Q)$$

- Given $G, \max_D V(G, D)$:

$$= -2\log(2) + 2\,\mathrm{JSD}\left(P_{\mathrm{data}}\,(x)\|P_G(x)\right)$$

- The optimal G minimizes JSD at $= 0$:

$$P_G(x) = P_{data}(x)$$

Practical implementation

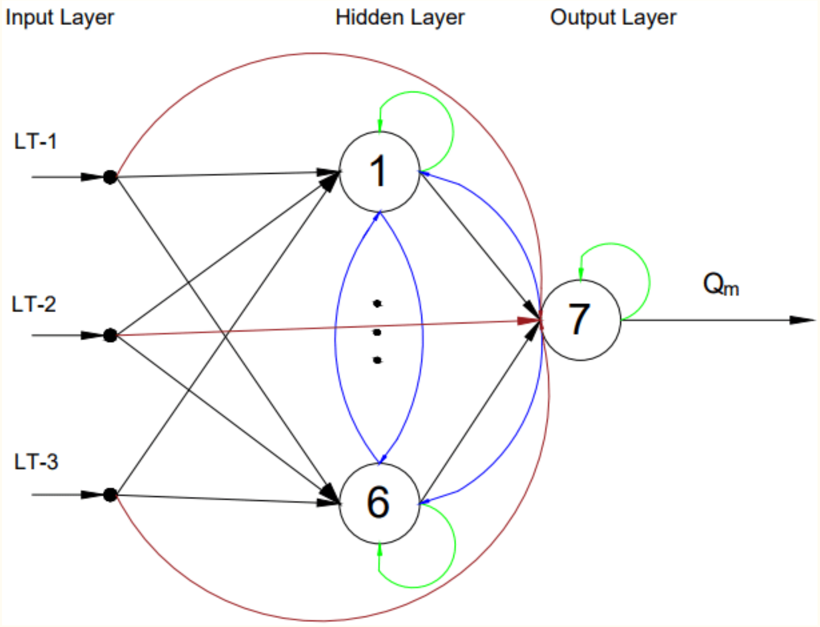
- Given G,we compute $\max_D V(G, D)$ by
  1. Sample $\left\{x^1, x^m\right\}$ from P_{data} .
  2. Sample $\left\{x^{*1}, , x^{*m}\right\}$ from generator P_G.
- And, then, we maximize:

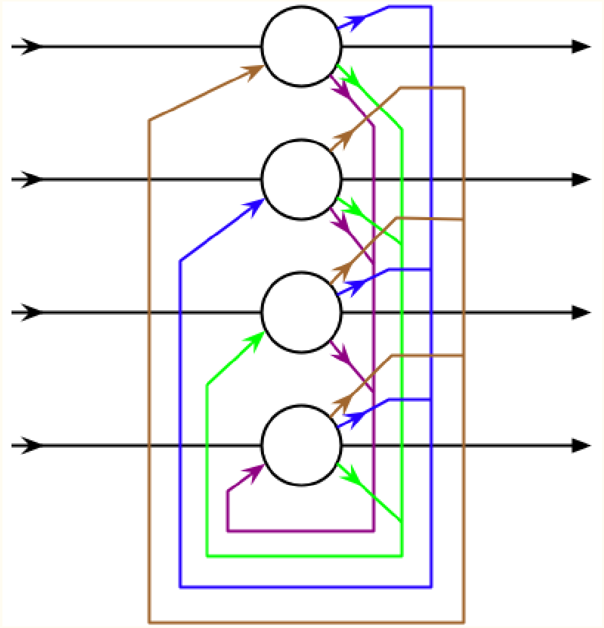
$$V' = \frac{1}{m \sum_{i=1}^{m} \log D\left(x^i\right)} + \frac{1}{m \sum_{i=1}^{m} \log\left(1 - D\left(x^{*i}\right)\right)}$$

- The binary classifier:
  - Output is $D(x)$, which minimizes cross-entropy.
  - If x is a positive example $\rightarrow$ Minimizes $-\log D(x)$.
  - If x is a negative example $\rightarrow$ Minimizes $-\log(1 - D(x))$.

Other architectures

- Bayesian neural networks.
- Recurrent neural networks: they built an internal state (memory) that can be used to process variable length sequences of inputs. Useful for time series
- Hopfield network (modern versions often known as dense associative memories).
- Boltzmann machines: stochastic version of a Hopfield network.
- Transformers: an attention-mechanism looks at an input sequence and picks, at each step, the parts of the sequence that are important for the task at hand.
- Self-organizing maps (SOM).


Input Layer
Hidden Layer
Output Layer
LT-1
LT-2
LT-3
1
6
7
Qm



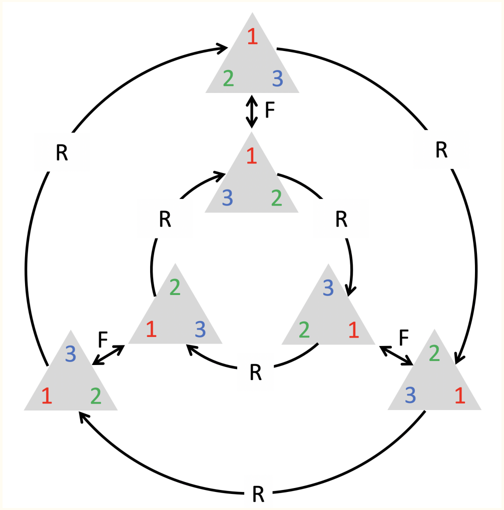
Data augmentation

Data augmentation

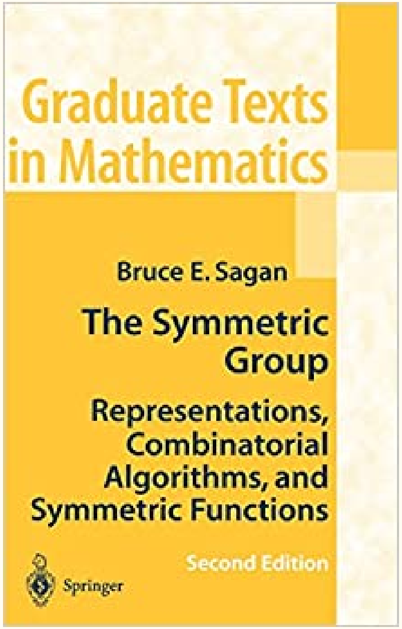
- How do we increase the observations to train a network efficiently?
  1. Synthetic Data: Rotations, translations, adding noise.
  2. Few-shot learning (Wang et al., 2020).
  3. Transfer learning.
  4. Janossy pooling: expressing a permutation-invariant function as the average of a permutation-sensitive function applied to all reorderings of the input sequence.
- How can this work?
- Remember that neural networks are, themselves, rotation-invariant.

Some algebra

- Group theory in math deals with how elements of a set compose with each other, not on what the elements are per se.
- Group: a set G together with a binary operation that satisfies associativity, identity element, and inverse element.
- Symmetric group $\mathcal{S}$ over a set G: group whose elements are all the bijections from G to itself, and whose group operation is the composition of functions.
- Symmetric group is really powerful!
- More general point: importance of algebra and geometry in deep learning.



	id	R	R^2	F	FR	FR^2
id	id	R	R^2	F	FR	FR^2
R	R	R^2	id	RF	RFR	RFR^2
R^2	R^2	id	R	R^2F	R^2FR	R^2FR^2
F	F	FR	FR^2	id	R	FR
FR	FR	FR^2	F	FRF	FRFR	$FRFR^2$
FR^2	FR^2	F	FR	FR^2F	FR^2FR	FR^2FR^2


Graduate Texts in Mathematics

Bruce E. Sagan

The Symmetric Group

Representations, Combinatorial Algorithms, and Symmetric Functions

Second Edition

Springer

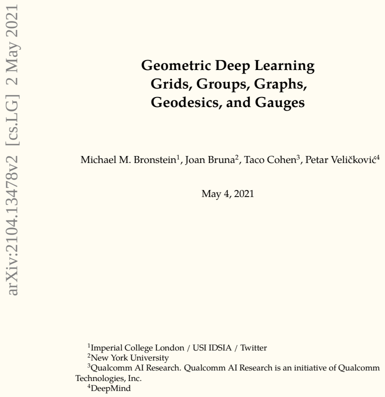
Deep learning and geometry

Deep learning and geometry, I

- Neural networks consist entirely of chains of tensor operations: we take $\mathbf{x}$, we perform affine transformations, and apply an activation function.
- Thus, these tensor operations are geometric transformations of $\mathbf{x}$.
- In other words: a neural network is a complex geometric transformation in a high-dimensional space.
- More in particular, deep neural networks look for convenient geometrical representations of high-dimensional manifolds.

Deep learning and geometry, II

- This suggests there are some deep theoretical links among different architectures.
- "Geometric unification" endeavor in the spirit of the *Erlangen Program* enunciated by Felix Klein: common framework to think about different architectures.
- See:
    1. `https://fabianfuchsml.github.io/learningonsets/`
    2. `https://www.youtube.com/watch?v=bIZB1hIJ4u8&t=368s&ab_channel=MachineLearningStreetTalk`
    3. `https://geometricdeeplearning.com/`


arXiv:2104.13478v2 [cs.LG] 2 May 2021

Geometric Deep Learning
Grids, Groups, Graphs,
Geodesics, and Gauges

Michael M. Bronstein[1], Joan Bruna[2], Taco Cohen[3], Petar Veličković[4]

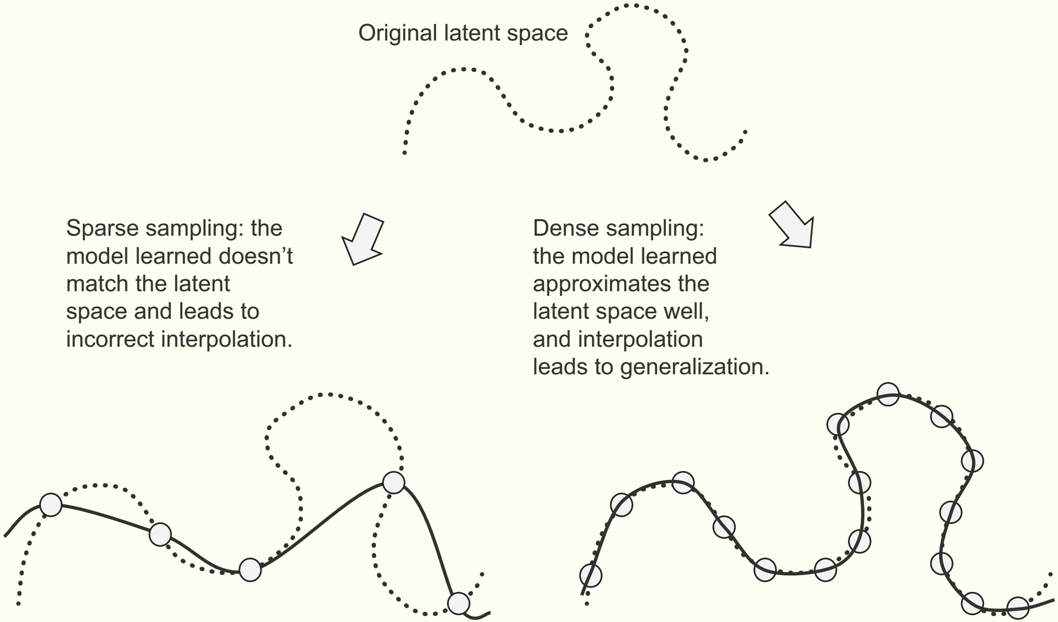
May 4, 2021

[1]Imperial College London / USI IDSIA / Twitter
[2]New York University
[3]Qualcomm AI Research. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.
[4]DeepMind

The manifold hypothesis

- The manifold hypothesis: all natural data lies on a low-dimensional manifold within the high-dimensional space where it is encoded (Cayton, 2005).
- Implications:

1. Deep learning only has to fit latent manifolds: simple, low-dimensional, highly structured subspaces within their potential input space.
2. Thus, we only need a local coordinate system of the underlying manifold.
3. Within one of these manifolds, one can interpolate between two inputs via a continuous path along which all points fall on the manifold.
4. Deep learning can account for complex data *and* generalize well if we sample from the right manifold.


Original latent space

Sparse sampling: the model learned doesn’t match the latent space and leads to incorrect interpolation.

Dense sampling: the model learned approximates the latent space well, and interpolation leads to generalization.