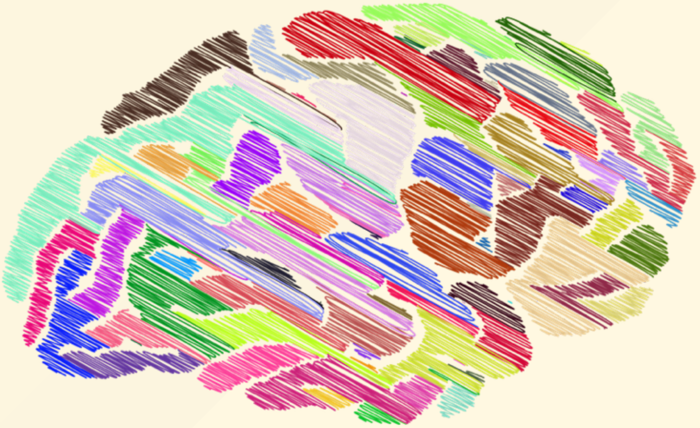
Network Function Spaces



Jeremy Bernstein
bernstein@caltech.edu

Agenda for today

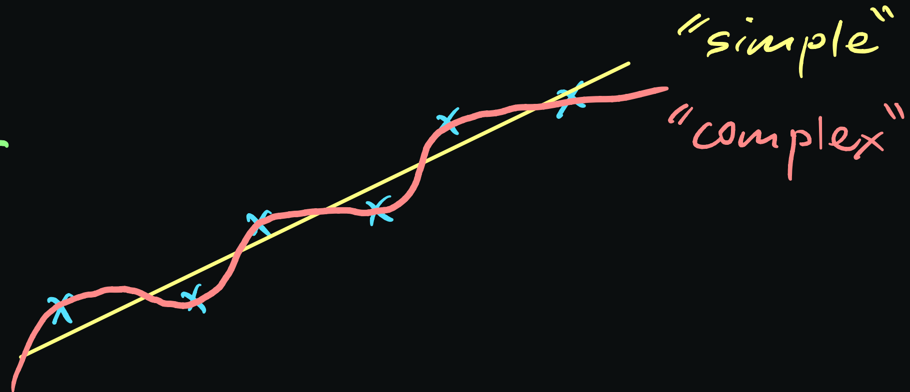
1. Complexity of a function space
2. Universal function approximators?
3. Studying random functions
4. Neural networks as Gaussian processes

A basic question

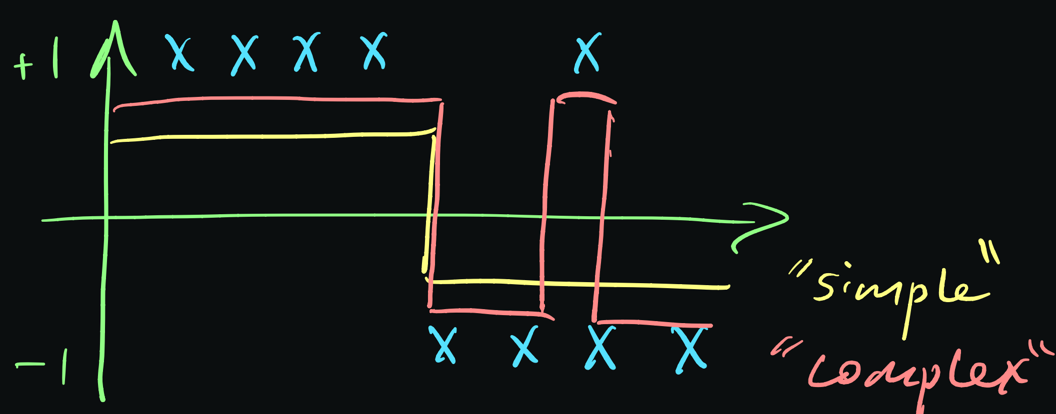
- What does an NN's function space look like?

How "complex" are the functions?

Regression



Classification



here we are being vague about what "complex" means

Universal function approximation

- Theorems like:

  "a wide enough NN can fit any function"

- Empirical findings like:

  "my NN can fit any labelling of the train set"

But this does not address what kinds of function are common — what kind of function is the architecture biased toward?

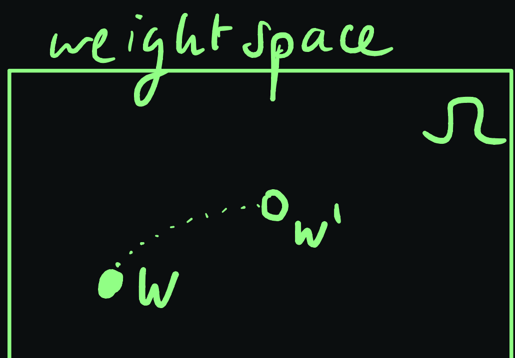
a.k.a. the NN's inductive bias — the focus of this lecture.

Weight space $\mapsto$ function space

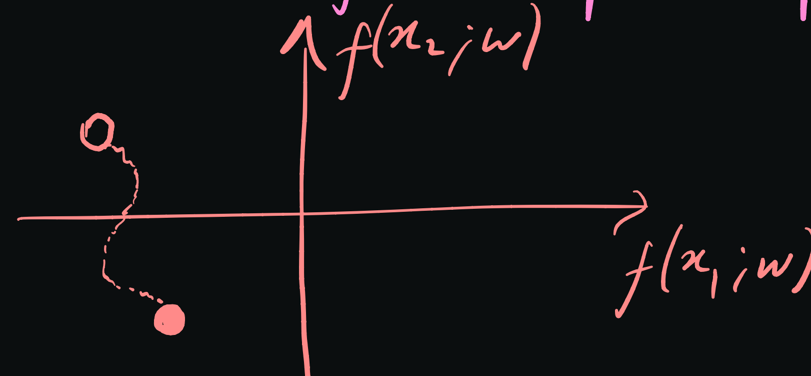
Assuming n inputs and an NN with a 1D output,

weight vector w $\mapsto$ function space representation $\begin{bmatrix} f(x_1; w) \\ f(x_2; w) \\ \vdots \\ f(x_n; w) \end{bmatrix} \in \mathbb{R}^n$

Varying w in weight space also moves the function space rep.

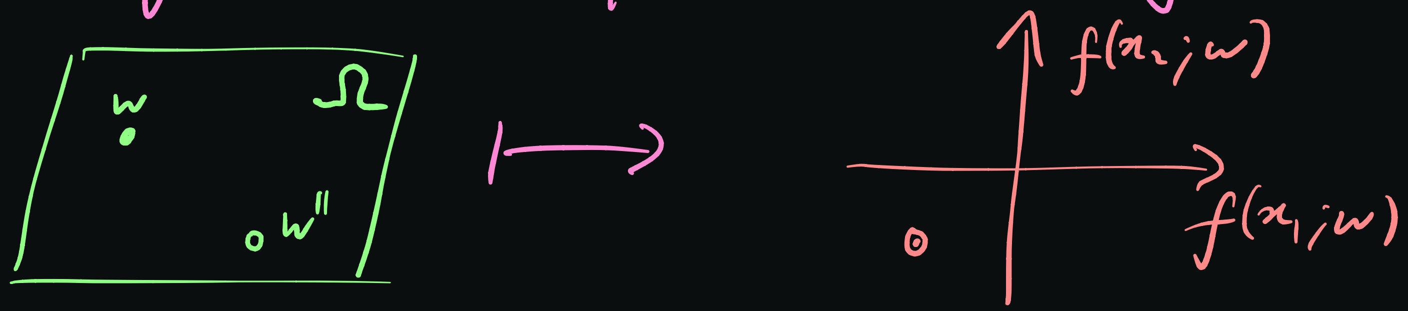


$\mapsto$

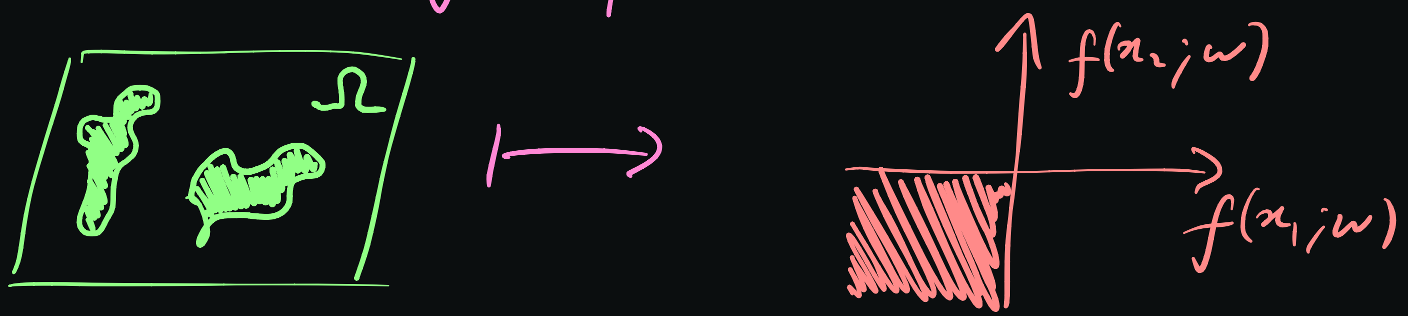


Life is simpler in function space

① Many weight vectors map to the same function:



② Simple geometries in function space may be intractably complicated in weight space:



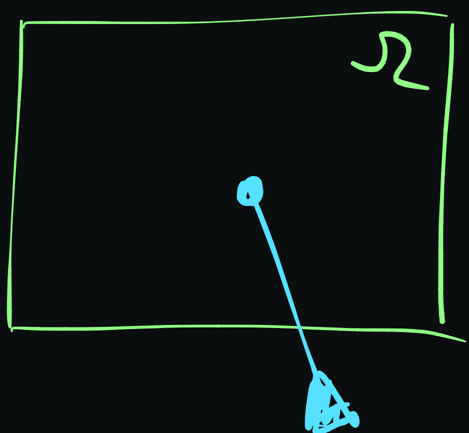
...so how do we move there?

Random functions

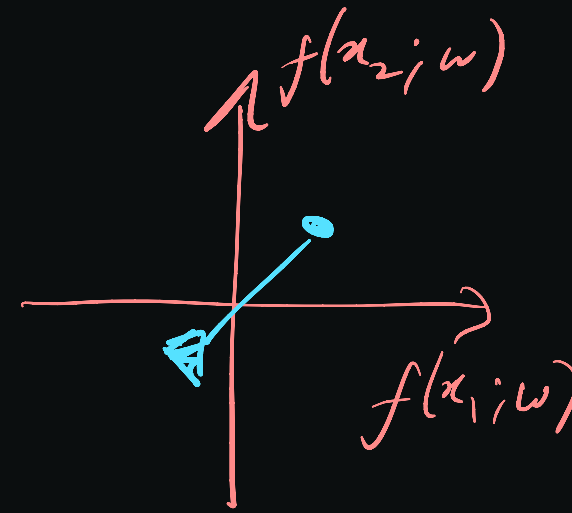
Key idea: to study what kinds of function an NN is biased towards, study the properties of random functions that the NN implements.

Sampling functions

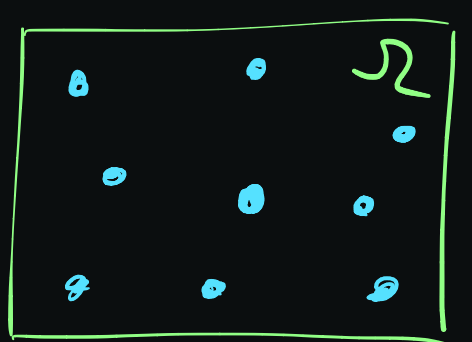
Imagine throwing a dart at weight space...

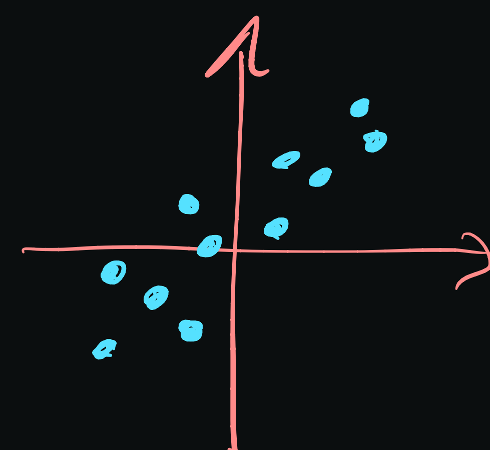


and inspecting the corresponding fn.



... then throw more darts

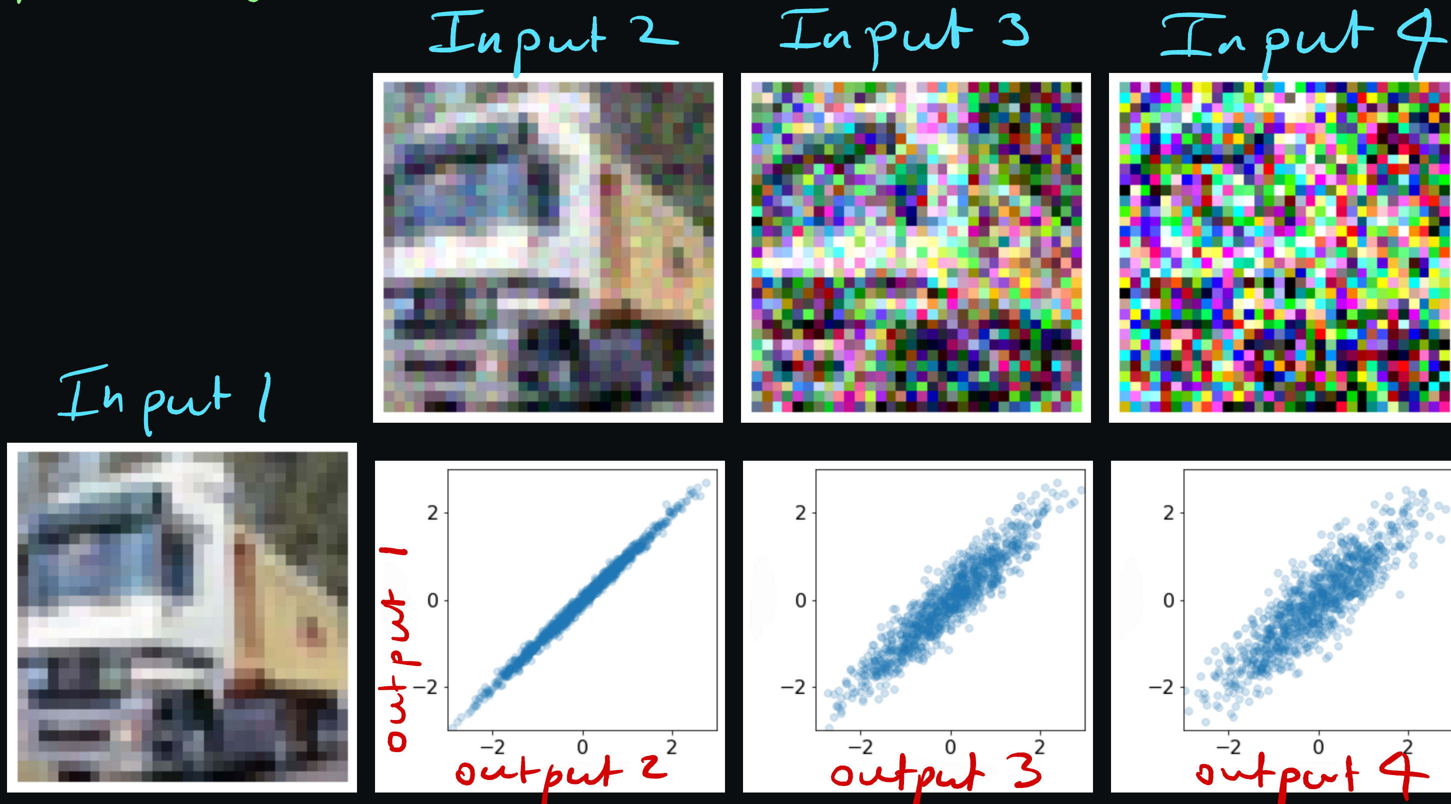




An example

network details { 3 layer MLP
width 1000
1 output

Repeatedly sample the weights randomly.



Outputs jointly Gaussian!

Jupyter notebook on the course website.

Output correlations

A Gaussian with mean zero is fully specified by its covariance matrix.

When sampling random networks, this is given by

$$\Sigma_{ij} := \mathbb{E}_{w \sim \mathbb{P}}\left[\underbrace{f(x_i; w)}_{\text{output on input } i}\ \underbrace{f(x_j; w)}_{\text{output on input } j}\right]$$

measure on weight space

enough

Wide NNs are Gaussian processes

- For any finite collection of inputs
  $x_1, x_2, \ldots, x_n$.
- For weights $w \overset{iid}{\sim}$ Gaussian.
- For a wide enough NN.

— the outputs are jointly Gaussian.

$$\begin{bmatrix} f(x_1; w) \\ \vdots \\ f(x_n; w) \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma).$$

Aside: Gaussian processes

If for all finite collections of inputs

$$x^{(1)}, \ldots, x^{(n)}$$

the following holds:

$$f(x^{(1)}), \ldots, f(x^{(n)}) \sim \mathcal{N}(\mu, \Sigma)$$

then we say that f is a Gaussian process.

To prove the claim that random, sufficiently wide NNs behave like GPs, we will need some...

Gaussian facts!

Classical CLT

central limit theorem

Let $X_1, \ldots, X_n$ be i.i.d. random variables each with mean 0 and finite variance σ^2.

Then as $n \to \infty$,

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \longrightarrow 0$$

with probability one.

$$\sqrt{n} \cdot \overline{X} = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_i \longrightarrow \mathcal{N}(0, \sigma^2).$$

Multivariate CLT

Let $Y_1, \ldots, Y_n$ be i.i.d. random vectors each with mean 0 and finite covariance Σ.

Then as $n \to \infty$,

$$\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \longrightarrow \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ with probability one.}$$

$$\sqrt{n}\cdot\overline{Y} = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} Y_i \longrightarrow \mathcal{N}(\underline{0}, \Sigma).$$

Linear transformation of Gaussian

Let Y be a random vector $Y \sim \mathcal{N}(\mu, \Sigma)$.

What is the distribution of AY?

fixed matrix.

AY is also Gaussian, by moment generating fns.

To see this,

- MGF of Y is: $\mathbb{E}e^{t^TY} = e^{t^T\mu + \frac{1}{2}t^T\Sigma t}$

$\Rightarrow$ MGF of AY is: $\mathbb{E}e^{t^TAY} = e^{t^T(A\mu) + \frac{1}{2}t^T(A\Sigma A^T)t}$

$\Rightarrow AY \sim \mathcal{N}(A\mu, A\Sigma A^T)$ (by uniqueness of MGFs.)

NNGP correspondence

- We now have everything we need to prove that NNs behave like GPs.
- we'll start simple and build up.

Linear neuron

Consider n inputs:

$$x^{(1)}, \dots, x^{(n)} \in \mathbb{R}^d$$

Stack the inputs into a "data matrix":

$$X = \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ & \vdots & \\ — & x^{(n)} & — \end{bmatrix}$$

Then the n outputs may be written as Xw

If the components of w are jointly Gaussian, then the outputs are given by a linear transformation of w $\Rightarrow$ the outputs are jointly Gaussian.

Linear neuron: covariance

What is the covariance of the linear neuron GP?

Consider two outputs

$$y^{(1)} = \sum_{i=1}^{d} w_i x_i^{(1)} \qquad y^{(2)} = \sum_{i=1}^{d} w_i x_i^{(2)}$$

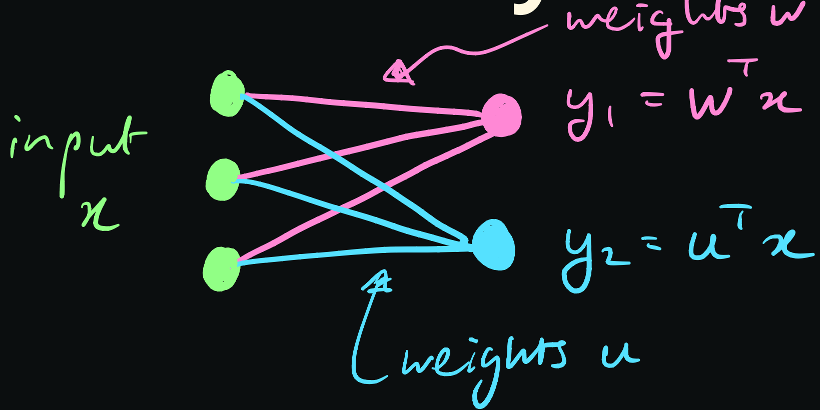
Suppose the weights $w_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$.

The means are $\mathbb{E}\, y^{(1)} = \mathbb{E}\, y^{(2)} = 0$.

Then the covariance is

$$\mathbb{E}\left[y^{(1)} y^{(2)}\right] = \mathbb{E}\left[\sum_{ij} w_i^{(1)} w_j^{(2)}\right] x_i^{(1)} x_j^{(2)} = \sigma^2 x^{(1)T} x^{(2)}$$

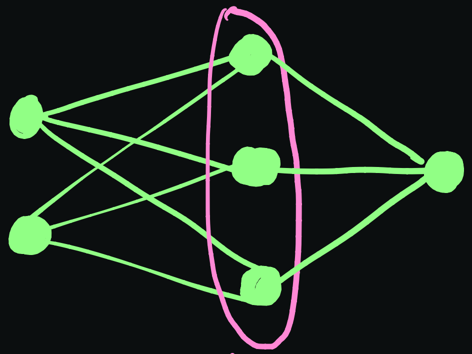
Linear layer



Provided the two weight vectors w and u are sampled independently of each other, then two neurons in a linear layer are just independent linear neurons.

The linear neuron yields a GP $\Rightarrow$ the linear layer does too.

One hidden layer



d_1 hidden units

nonlinearity

$$y(x) = \sum_{i=1}^{d_1} u_i \, \phi(w_i^T x)$$

second layer weights

weight vector of ith hidden neuron

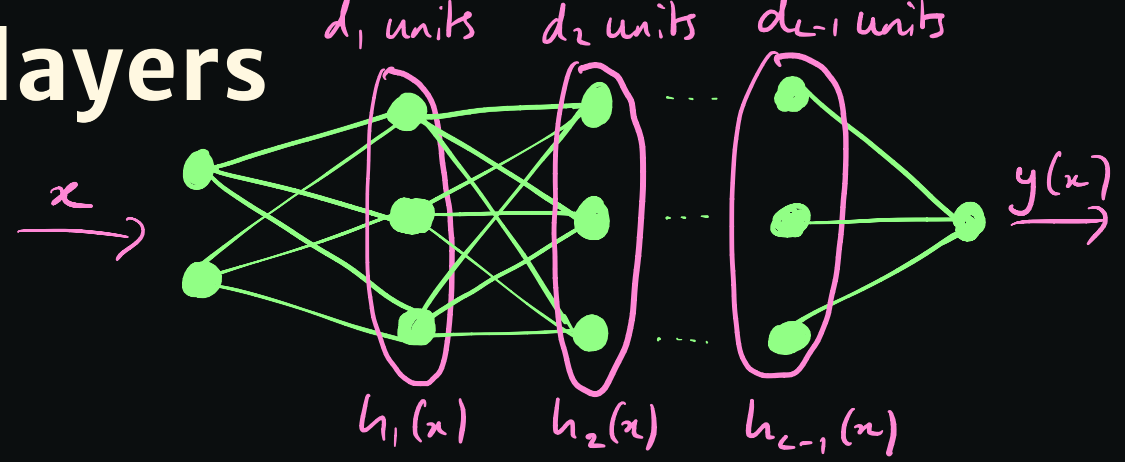
For inputs $x^{(1)}, \dots, x^{(n)}$, consider vector

$$\left[y(x^{(1)}), \dots, y(x^{(n)}) \right] = \sum_{i=1}^{d_1} \left[u_i \phi(w_i^T x^{(1)}), \dots, u_i \phi(w_i^T x^{(n)}) \right]$$

If all the weights are drawn iid with finite variance (and ϕ does not blow up variances) then each summand is an iid random vector with finite covariance.

So, as $d_1 \to \infty$, $[y(x^{(1)}), \dots, y(x^{(n)})]$ is Gaussian by the MV-CLT.

Many hidden layers



Again, for inputs $x^{(1)}, x^{(2)}, \dots, x^{(n)}$, consider vector

$$\left[y(x^{(1)}), \dots, y(x^{(n)})\right] = \sum_{i=1}^{d_{L-1}} \left[u_i \phi(h_i^{L-1}(x^{(1)})), \dots, u_i \phi(h_i^{L-1}(x^{(n)}))\right]$$

We'd like to take $d_{L-1} \to \infty$ and apply the MV-CLT as we did for one hidden layer, but first we need to check that for a fixed input x, the hidden units $h_1^{L-1}(x), h_2^{L-1}(x), \dots, h_{d_{L-1}}^{L-1}(x)$ are independent.

Surprisingly, this does hold in the limit that $d_1, d_2, \dots, d_{L-2} \to \infty$.

The proof is by induction, using the MV-CLT.

Relu networks

Relu MLP covariance

- L-layer MLP, hidden layer width $\to \infty$
- nonlinearity $\phi(z) := \sqrt{2} \cdot \max(0, z)$
- inputs $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ with $\|x_i\|_2 = \sqrt{d}$
- iid Gaussian weights $\mathcal{N}\left(0, \frac{1}{\text{fan in}}\right)$

Define $h(t) := \frac{1}{\pi}\left[\sqrt{1-t^2} + t(\pi - \arccos t)\right]$.

Then for two inputs $x, x' \in \mathbb{R}^d$,

$$\boxed{\mathbb{E}[f(x) f(x')] = \underbrace{h \circ \ldots \circ h}_{L-1 \text{ times}}\left(\frac{x^T x'}{d}\right)}$$

the compositional arccosine kernel.

Interpretation: $\frac{x^T x'}{d}$ is the covariance for a linear neuron

The additional layers modify this covariance via the fn. h.

Summary

We found that for a finite collection of inputs, the outputs of a sufficiently wide NN are jointly Gaussian w.r.t. random sampling of the weights.

The covariance matrix of this Gaussian depends on how the network architecture transforms the input correlations.

Next lecture

The first application of our tools:

— optimisation of neural networks.

🎉