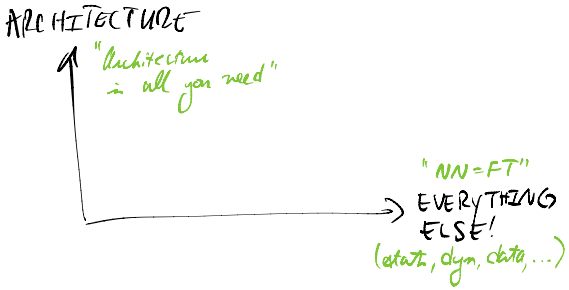
Heidelberg School

NN = FT & ARCH IS ALL YOU NEED

Heidelberg Lectures



NN=FT OUTLINE

1. FT → NN (invertable!)
2. NN → FT — Q?
   - i) NN-FT DEFINITION
   - ii) FREE THEORIES (NNGP)
   - iii) INTERACTIONS
   - iv) SYMMETRIES
   - v) YOUR FAVS ϕ^4, CFT, QM

1 FT → NN

The Setup.

Understanding ML at the very least means understanding neural networks. A neural network is a function

$$\phi_\theta : \mathbb{R}^d \to \mathbb{R} \tag{1}$$

with parameters θ. We've chosen outputs in $\mathbb{R}$ because, channeling Coleman, scalars already exhibit the essentials. We'll use the lingo

$$\begin{aligned} \text{Input:} \quad & x \in \mathbb{R}^d && (2)\\ \text{Output:} \quad & \phi_\theta(x) \in \mathbb{R} && (3)\\ \text{Network:} \quad & \phi_\theta \in \text{Maps}(\mathbb{R}^d, \mathbb{R}) && (4)\\ \text{Data:} \quad & \mathcal{D}, && (5) \end{aligned}$$

where the data $\mathcal{D}$ depends on the problem, but involves at least a subset of $\mathbb{R}^d$, potentially paired with labels $y \in \mathbb{R}$.

With this minimal background, let's ask our central question:

Question: What does a NN predict?

For any fixed value of θ, the answer is clear: $\phi_\theta(x)$. However, the answer is complicated by issues of both dynamics and statistics.

First, dynamics. In ML, parameters are updated to solve problems and we really have **trajectories** in

$$\begin{aligned} \text{Parameter Space:} \quad & \theta(t) \in \mathbb{R}^{|\theta|} && (6)\\ \text{Output Space:} \quad & \phi_{\theta(t)}(x) \in \mathbb{R} && (7)\\ \text{Function Space:} \quad & \phi_{\theta(t)} \in \text{Maps}(\mathbb{R}^d, \mathbb{R}). && (8) \end{aligned}$$

governed by some learning dynamics determined by the optimization algorithm and the nature of the learning problem. For instance, in supervised learning we have data

$$\mathcal{D} = \{(x_\alpha, y_\alpha) \in \mathbb{R}^d \times \mathbb{R}\}_{\alpha=1}^{|\mathcal{D}|}, \tag{9}$$

and a loss function

$$\mathcal{L}[\phi_\theta] = \sum_{\alpha=1}^{|\mathcal{D}|} \ell(\phi_\theta(x_\alpha), y_\alpha), \tag{10}$$

where ℓ is a loss function such as $\ell_{\text{MSE}} = (\phi_\theta(x_\alpha) - y_\alpha)^2$. One may optimize θ by gradient descent

$$\frac{d\theta_i}{dt} = -\nabla_{\theta_i} \mathcal{L}[\phi_\theta], \tag{11}$$

or other algorithms, e.g., classics like stochastic gradient descent (SGD) [23, 24] or Adam [25], or a more recent technique such as Energy Conserving Descent [26, 27]. Throughout, t is training time of the learning algorithm unless otherwise noted.

Second, statistics. When a NN is initialized on your computer, the parameters θ are initialized as draws

$$\theta \sim P(\theta) \tag{12}$$

from a distribution $P(\theta)$, where $\sim$ means "drawn from" in this context. Different draws of θ will give different functions ϕ_θ, and a priori we have no reason to prefer one over another. The prediction $\phi_\theta(x)$ therefore can't be fundamental! Instead, what is fundamental is the average prediction and second moment or variance:

$$\mathbb{E}[\phi_\theta(x)] = \int d\theta P(\theta)\, \phi_\theta(x) \tag{13}$$

$$\mathbb{E}[\phi_\theta(x)\phi_\theta(y)] = \int d\theta P(\theta)\, \phi_\theta(x)\phi_\theta(y), \tag{14}$$

as well as the higher moments. Expectations are across different initializations. Since we're physicists, we henceforth replace $\mathbb{E}[\cdot] = \langle\cdot\rangle$ and we remember this is a statistical expectation value. It's useful to put this in our language:

$$G^{(1)}(x) = \langle \phi_\theta(x) \rangle \tag{15}$$

$$G^{(2)}(x,y) = \langle \phi_\theta(x)\phi_\theta(y) \rangle, \tag{16}$$

the mean prediction and second moment are just the one-point and two-point correlation functions of the statistical ensemble of neural networks. Apparently ML has something to do with field theory.

Putting the dynamics and statistics together, we have an ensemble of initial θ-values, each of which is the starting point of a trajectory

$\theta(t)$, and therefore we have an ensemble of trajectories. We choose to think of $\theta(t)$ drawn as

$$\theta(t) \sim P(\theta(t)), \tag{17}$$

a density on parameters that depends on the training time and yields time-dependent correlators

$$G_t^{(1)}(x) = \langle \phi_\theta(x) \rangle_t \tag{18}$$

$$G_t^{(2)}(x,y) = \langle \phi_\theta(x)\phi_\theta(y) \rangle_t, \tag{19}$$

where the subscript t indicates time-dependence and the expectation is with respect to $P(\theta(t))$. Of course, assuming that learning is helping, we wish to take $t \to \infty$ and are interested in

$G_\infty^{(1)}(x)$ = mean prediction of ∞-number of NNs as $t \to \infty$.

Remarkably, we will see that in a certain supervised setting there is an exact analytic solution for this quantity.

WHY YOU CARE

PREDICTION → $G^{(1)}$

UNCERTAINTY QUANT → $G^{(2)}$

2 NN → FT

i) DEFN

Question: What is a field theory?

At the very least, a field theory needs

- **Fields**, functions from an appropriate function space, or sections of an appropriate bundle, more generally.
- **Correlation Functions** of fields, here expressed as scalars

$$G^{(n)}(x_1, \ldots, x_n) = \langle \phi(x_1) \ldots \phi(x_n) \rangle. \tag{128}$$

You might already be wanting to add more beyond these minimal requirements – we'll discuss that in a second. For now, we have

Answer: a FT is an ensemble of functions with a way to compute their correlators.

DEFINING DATA OF NN-FT

$(\phi_\theta, P(\theta))$

In the Euclidean case, when the expectation is a statistical expectation, one my say

Euclidean Answer: a FT is a statistical ensemble of functions.

Our minimal requirements get us a partition function

$$Z[J] = \langle e^{\int d^d x J(x)\phi(x)} \rangle \tag{129}$$

that we can use to compute correlators, where at this stage we are agnostic about the definition of $\langle\cdot\rangle$. In normal field theory, the $\langle\cdot\rangle$ is defined by the Feynman path integral

$$Z[J] = \int \mathcal{D}\phi\, e^{-S[\phi] + \int d^d x J(x)\phi(x)}, \tag{130}$$

which requires specifying an action $S[\phi]$ that determines a density on functions $\exp(-S[\phi])$. But that's not the data we specify when we specify a NN. The NN data $(\phi_\theta, P(\theta))$ instead defines

$$Z[J] = \int d\theta P(\theta) e^{\int d^d x J(x)\phi_\theta(x)}. \tag{131}$$

↝ for quantum & for gaussian!

Q: WHEN NN-QFT?

ONE A: $G^{(n)}$ SATISFY OS-AXIOMS (see 2021 paper & paper w/ Fabio)

ii) FREE THEORIES (NNGP correspondence)

For simplicity, we again consider a single-layer fully connected network of width N, with the so-called biases turned off for simplicity:

$$\phi(x) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \sum_{j=1}^{d} w_i^{(1)} \sigma(w_{ij}^{(0)} x_j), \tag{32}$$

where the set of network parameters is $\theta = \{w_{ij}^{(0)}, w_i^{(1)}\}$ independently and identically distributed (i.i.d.).

$$w_{ij}^{(0)} \sim P(w^{(0)}) \qquad w_i^{(1)} \sim P(w^{(1)}). \tag{33}$$

Under this assumption, we see

Observation: The network is a sum of N i.i.d. functions.

This is a function version of the Central Limit Theorem, generalizing the review in Appendix A, and gives us the Neural Network / Gaussian Process (NNGP) correspondence,

NNGP Correspondence: in the $N \to \infty$ limit, ϕ is drawn from a Gaussian Process (GP),

$$\lim_{N\to\infty} \phi(x) \sim \mathcal{N}(\mu(x), K(x,y)), \tag{34}$$

with mean and covariance (or kernel) $\mu(x)$ and $K(x,y)$.

By the CLT, $\exp(-S[\phi])$ is Gaussian and therefore $S[\phi]$ is quadratic in networks. Now this really feels like physics, since the infinite neural network is drawn from a Gaussian density on functions, which defines a generalized free field theory.

We will address generality of the NNGP correspondence momentarily, but let's first get a feel for how to do computations. To facilitate then, we take $P(w^{(1)})$ to have zero mean and finite variance,

$$\langle w^{(1)} \rangle = 0 \qquad \langle w^{(1)} w^{(1)} \rangle = \mu_2, \tag{35}$$

which causes the one-point function to vanish $G^{(1)}(x) = 0$. Following Williams [34], we compute the two-point function in parameter space (with Einstein summation)

$$G^{(2)}(x,y) = \frac{1}{N} \langle w_i^{(1)} \sigma(w_{ij}^{(0)} x_j)\; w_k^{(1)} \sigma(w_{kl}^{(0)} y_l) \rangle \tag{36}$$

$$= \frac{1}{N} \langle w_i^{(1)} w_k^{(1)} \rangle \langle \sigma(w_{ij}^{(0)} x_j) \sigma(w_{kl}^{(0)} y_l) \rangle \tag{37}$$

$$= \frac{\mu_2}{N} \langle \sigma(w_{ij}^{(0)} x_j)\, \sigma(w_{il}^{(0)} y_l) \rangle, \tag{38}$$

where the last equality follows from the ones being i.i.d., $\langle w_i^{(1)} w_k^{(1)} \rangle = \mu_2 \delta_{ik}$. The sum over i gives us N copies of the same function, leaving us with

$$G^{(2)}(x,y) = \mu_2\, \langle \sigma(w_{ij}^{(0)} x_j)\, \sigma(w_{il}^{(0)} y_l) \rangle, \tag{39}$$

where we emphasize there is now *no summation on* i. This is an exact-in-N two-point function that now requires only on the computation of the quantity in bra-kets. One may try to evaluate it exactly by doing the integral over $w^{(0)}$. If it can't be done, Monte Carlo estimates may be obtained from M samples of $w^{(0)} \sim P(w^{(0)})$ as

$$G^{(2)}(x,y) \simeq \frac{\mu_2}{M} \sum_{\text{samples}}^{M} \sigma(w_{ij}^{(0)} x_j)\, \sigma(w_{il}^{(0)} y_l). \tag{40}$$

In typical NN settings, parameter densities are easy to sample for convenience, allowing for easy computation of the estimate. If the density is more complicated, one may always resort to Markov chains, e.g. as in lattice field theory.

With this computation in hand, we have the defining data of this NNGP,

$$\lim_{N\to\infty} \phi(x) \sim \mathcal{N}\left(0, G^{(2)}(x,y)\right). \tag{41}$$

The associated action is

$$S[\phi] = \int d^d x d^d y\, \phi(x)\, G^{(2)}(x,y)^{-1}\, \phi(y), \tag{42}$$

where

$$\int d^d y\, G^{(2)}(x,y)^{-1} G^{(2)}(y,z) = \delta^{(d)}(x-z). \tag{43}$$

defines the inverse two-point function. In fact, this allows us to determine the action of any NNGP with $\mu(x) = G^{(1)}(x) = 0$, by computing the $G^{(2)}$ in parameter space and inverting it.

So certain large neural networks are function draws from generalized free field theories. But at this point you might be asking yourself

Question: How general is the NNGP correspondence?

VERY! MLP N = WIDTH

CNN N = CHANNELS

ATTN N = HEADS ... (see Yang)

iii) INTERACTIONS

CLT → FREE

INTERACTIONS = NOT FREE → NOT CLT

→ parametric control from "breaking of CLT assump."

CONTROLLED INTERACTIONS

1) N FINITE $1/N$ CORRECTIONS

$$G_c^{(2r)} \propto \frac{1}{N^{r-1}}$$

($\to 0 \ \forall\ r > 2$, ⇒ Gaussian thy)

2) BREAK INDEPENDENCE

eg $e^{-\frac{w_i w_i}{\sigma^2} - \lambda (w_i w_i)(w_j w_j)}$

mixing terms forbid factorization

LEADING INTERACTIONS (in $1/N$)

$$\phi(x) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} w_i \varphi_i(x)$$

$$G^{(4)} = \langle \phi(x)\phi(y)\phi(z)\phi(w) \rangle \tag{45}$$
$$= \sum_{i,j,k,l} \langle w_i w_j w_k w_l \rangle \langle \varphi_i(x)\varphi_j(y)\varphi_k(z)\varphi_l(w) \rangle \tag{46}$$
$$= \sum_i \langle w_i^4 \rangle \langle \varphi_i(x)\varphi_i(y)\varphi_i(z)\varphi_i(w) \rangle \tag{47}$$
$$+ \sum_{i \neq j} \langle w_i^2 \rangle \langle w_j^2 \rangle \langle \varphi_i(x)\varphi_i(y)\varphi_j(z)\varphi_j(w) + \text{perms} \rangle. \tag{48}$$

One can see that you have to be careful with indices. The connected 4-pt function is [38]

$$G_c^{(4)}(x,y,z,w) = G^{(4)}(x,y,z,w) - \left(G^{(2)}(x,y) G^{(2)}(z,w) + \text{perms} \right), \tag{49}$$

and watching indices carefully we obtain

$$G_c^{(4)}(x,y,z,w) = \frac{1}{N} \Big(\mu_4 \langle \varphi_i(x)\varphi_i(y)\varphi_i(z)\varphi_i(w) \rangle \tag{50}$$
$$- \mu_2^2 \left(\langle \varphi_i(x)\varphi_i(y) \rangle \langle \varphi_i(z)\varphi_i(w) \rangle + \text{perms} \right) \Big), \tag{51}$$

iv) SYMMETRIES

MECHANISM ACT, REDEFINE WEIGHTS, SHOW INVCE

EG

$\phi(x) = a\,\sigma(bx)$, $x \to -x$, $P(b) = P(-b)$

TRAFO'D CORR → $\mathbb{E}_{P(a)P(b)}[\phi(-x_1)\dots\phi(-x_n)]$

ABSORB AS $b \to -b$ $= \mathbb{E}_{P(a)P(-b)}[\phi(x_1)\dots\phi(-x_n)]$

$P(b) = P(-b)$ $= \mathbb{E}_{P(a)P(b)}[\phi(x_1)\dots\phi(x_n)]$

EG FREE SCALAR AS $N \to \infty$

$$\phi(x) = \sqrt{\frac{2\text{vol}(B_\Lambda^d)}{(2\pi)^d \sigma_{w_1}^2}} \frac{1}{\sqrt{\mathbf{w}^{(0)2}_i + m^2}} w_i^{(1)} \cos\left(w_{ij}^{(0)} x_j + b_i^{(0)} \right), \tag{72}$$

and specific parameter densities

$$w^{(1)} \sim \mathcal{N}\left(0, \frac{\sigma_{w_1}^2}{N}\right) \qquad w^{(0)} \sim \text{Unif}(B_\Lambda^d) \qquad b^{(0)} \sim \text{Unif}[-\pi, \pi]. \tag{73}$$

where B_Λ^d is a d-ball of radius Λ. The theory is translation invariant by construction, and so we compute the power spectrum of the two-point function $G^{(2)}(x-y)$ to be

$$G^{(2)}(p) = \frac{1}{p^2 + m^2}. \tag{74}$$

note ϕ^4 by interp breaking of symmetries

v) YOUR FAVS

ϕ^4 ← op in def'n action

$$I_{\phi^4}[J] = \left\langle e^{\int J\phi}\, e^{-\lambda \int \phi^4(x)} \right\rangle_{FS}$$

NN-INTERP: $\phi = \phi_{w^{(1)}, w^{(0)}, b^{(0)}}$

$\Rightarrow e^{-\int \phi^4(x)}$ DEFORMS $P(w^1 w^0 b^0)$

BREAKS INDEP!

CFT: EMBEDDING FORMALISM (due to Dirac)

$SO(D+1,1) = \text{CFT}_D = \text{LOR}_{D+1,1}$

NN-CFT HOMOG LI. NN ON $\mathbb{R}^{D+1,1}$

↓ P.S. OF P.N.C.

CFT ON $\mathbb{R}^D$

QM: i) ANY QM THY HAS NN REP

ii) NN (MARKOV PROCESS)

→ REF. POSITIVE → UNITARY QM

iii) CLASSIC RESULTS (COMMUTATORS, HUP, ETC)

FERMS & SUSY IN PROGRESS

13

Architecture is All You Need Outline

1. UAT & FFNN
2. Kolmogorov-Arnold Nets (KANs)
3. Aparsity: MLP → KAN
   ↳ plots, if time
4. if necessary, KANs & symb reg

14

1 UAT & FFNN

The Universal Approximation Theorem (UAT) is the first result in this direction. It states that a neural network with a single hidden layer can approximate any continuous function on a compact domain to arbitrary accuracy. More precisely, the origin version of the theorem due to Cybenko is

Theorem 2.1 (Cybenko). *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous function on a compact set $K \subset \mathbb{R}^d$. Then for any $\epsilon > 0$ there exists a neural network with a single hidden layer of the form*

$$\phi(x) = \sum_{i=1}^{N} \sum_{j=1}^{d} w_i^{(1)} \sigma(w_{ij}^{(0)} x_j + b_i^{(0)}) + b^{(1)}, \tag{21}$$

$\theta = \{w_{ij}^{(0)}, w_i^{(1)}, b_i^{(0)}, b^{(1)}\}$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-polynomial non-linear activation function, such that

$$\sup_{x \in K} |f(x) - \phi(x)| < \epsilon. \tag{22}$$

The architecture in Eq. (21) is known by many names, including the perceptron, a fully-connected network, or a **feedforward network**. The parameter N is known as the **width**. It may be generalized to include a **depth** dimension L encoding the number of compositions of affine transformations and non-linearities. Such an architecture is known as a *multi-layer* perceptron (MLP) or a deep feedforward network.

The UAT is a powerful result, but it has some limitations. First, though the error ϵ does get better with N, it doesn't say how many neurons are needed. Second, it doesn't say how to train the network: though there is a point θ^* in parameter space that is a good approximation to any f, existence doesn't imply that we can find it, which is a question of learning dynamics.

To ask the obvious,

Question: Why does this UAT work?

15

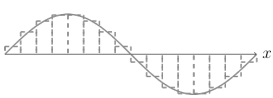


Figure 1: The Universal Approximation Theorem can be understood by approximating a function like $\sin(x)$ with a series of bumps.

In Cybenko's original work, he focused on the case that σ was the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{23}$$

which allows us to get a picture of what's happening. The sigmoids appear in the network function of the form

$$\sigma\left(w_i^{(0)} \cdot x + b_i^{(0)}\right) \tag{24}$$

which approximates a shifted step function as $w^{(0)} \to \infty$. A linear combination can turn it into a bump with approximately compact support that gets scaled by $w^{(1)}$. These bumps can then be put together to approximate any function; see, e.g., Fig. 2.1.

Cybenko's version of the UAT was just the first, and there are many generalizations, including to deeper networks, to other activation functions, and to other domains. See [28] for the original paper and [29] for a generalization to the case of multiple hidden layers.

(Q: thm in 1980's why so long for deep learning to take off?)

16

② KAN

THM (Kolmogorov-Arnold)

GIVEN MULTIVARIATE CONTINUOUS

$$f: [0,1]^n \to \mathbb{R}$$

$\exists$ A REPRESENTATION

$$f(x_1,\dots,x_n) = \sum_{q=0}^{2n} \Phi_q\left(\sum_{p=1}^{n} \phi_{p,q}(x_p)\right)$$

w/ $\phi_{qp}: [0,1] \to \mathbb{R}$ $\quad \Phi_q: \mathbb{R} \to \mathbb{R}$

"Univariate and addition are all you need"

CAVEAT: INNER FNS ϕ_{pq} MAY NOT BE SMTH.

EX 1: $xy = e^{\log x + \log y}$

EX 2 $e^{\sin(x_1^2+x_2^2) + \sin(x_3^2+x_4^2)}$ → technically not right form!

NOTE:

① $\exists$ MANY VERSIONS! see eg 2412.16738

② $\exists$ SMOOTH APPROX's (w/ assumptions) eg f_{target} α-Hölder $\frac{|f(x)-f(y)|}{|x-y|^\alpha} < \infty$

③ LOOKS LIKE NN!

ACTIVATIONS ON EDGES

DEFN: (KAN) A DEPTH L $f: \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ KOLMOGOROV-ARNOLD NETWORK IS

$$f_{i_L}(x) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi^{(L-1)}_{i_L i_{L-1}}\left(\sum_{i_{L-2}} \phi^{(L-2)}_{i_{L-1} i_{L-2}}\left(\dots\left(\sum_{i_0=1}^{n_0} \phi^{(0)}_{i_1 i_0}(x_{i_0})\right)\right)\right)$$

WHERE $\phi^{(k)}$ ACTIVS ON EDGES.

NB ϕ HAS PARAMS, LEARNED

"don't worry, take inspiration, generalize nets & train!"

2404.19756 COMMENTS

① MANY APPLICATIONS

② $\exists$ IMPLEMENTATION (see slides)

③ B-SPLINES (truncable)

$$\phi = w_b\, b(x) + w_s \sum_i c_i B_i(x)$$

- LOCALITY HELPS CAT. FORGET.
- SCALING THEOREM (2.1)

N = # PARAMS, ℓ = TEST RMSE, d = DATA DIM $\Big\}$ $\ell \sim N^{-\alpha(d)}$

THEN $\alpha_{\text{B-SPLINE-KAN}} = k+1$ (poly deg in spline) (DIM INDE)

COMPARE: $\alpha_{\text{ReLU-NN}} = \frac{k+1}{d}$

② SPARSITY

KAN = SPARSE MLP

SPARSITY MAP: ① HAS GUARANTEE ② MAPS ACTIVS FROM NODES → EDGES

"MLP BIG" WIDTH BN^2 $\sigma_1,\dots,\sigma_B$

$$z^{(\ell+2)}_{\alpha;\alpha} = \sum_b w^{(\ell+1)}_{\alpha b}\, \sigma_{b \bmod B}\left(\sum_c w^{(\ell)}_{bc} h^{(\ell)}_{c;\alpha}\right)$$

we Mat(BN^2, BN^2)

$W = AB \in \det_N(\text{Mat}(\cdot))$ ($BN^2 \times N$)

$$\widehat{z}^{(\ell+1)}_{i;\alpha} = \sum_{b=1}^{BN^2} B^{(\ell+1)}_{ib}\, \sigma_{b \bmod B}\left(\sum_{j=1}^{N} A^{(\ell)}_{bj} z^{(\ell)}_{j;\alpha}\right)$$

"MLP SMALL"

(middle datum = KAN indices, early datum = MLP indices)

KEY $b = 1,\dots,BN^2$ ↓ factor $k,\ell = 1,\dots,N$ $\quad g = 1,\dots,B$

"MLP SMALL"

$w = AB \in \det_N(\mathrm{Mat}(\cdot))$

$$z_{i;\alpha}^{(\ell+1)} = \sum_{b=1}^{BN^2} B_{ib}^{(\ell+1)} \sigma_{b\%B}\left(\sum_{j=1}^{N} A_{bj}^{(\ell)} z_{j;\alpha}^{(\ell)}\right)$$

(middle station = KAN indices
early station = MLP indices)

RESHAPE $b \longrightarrow \ell g k$

BN^2 $\quad$ N B N

$$= \sum_{\ell g k} B_{i;\ell g k}^{(\ell+1)} \sigma_q\left(\sum_{j=1}^{N} A_{\ell g k j} z_{j;\alpha}^{(\ell)}\right)$$

learned act between i,k

SPARSIFY w $\delta_{i\ell}$ δ_{kj}

$$= \sum_{gk} B_{i;gk}^{(\ell+1)} \sigma_g\left(A_{jgkh} z_{k;\alpha}^{(\ell)}\right) =: \sum_{k} \Phi_{ij}^{(\ell+1)}(z_{kj;\alpha}^{(\ell)})$$

(varying σ for MLP become "basis" for learned activation

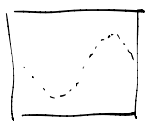
Q: MLP SMALL $\longrightarrow$ KAN SPARSITY VS RANDOM UNIFORM SPARSITY?

③ SYMBOLS

METHOD GIVEN DATA SYMBOLIC REGRESSION TO FIND INTERPRETABLE EQNS FOR IT.

Qs: WHAT FORM? 1D? HIGHER D?

LLM-LEX VISUAL SYMB REG



CAN LM GUESS FUNCTIONAL FORM?

YES, BUT ITER W/ GA

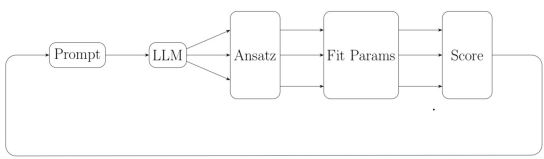


Figure 2: The structure of LLM-LEx.

(result: competitive w/ Mathematica, even w/ open source LMs

LIMITATION: 1) FUNCTIONS (b/c trained on 2D images)

KAN-LEx 1D $\longrightarrow$ HIGHER D?

NOTE: ORIG KAN PACKAGE HAS IMAGES!

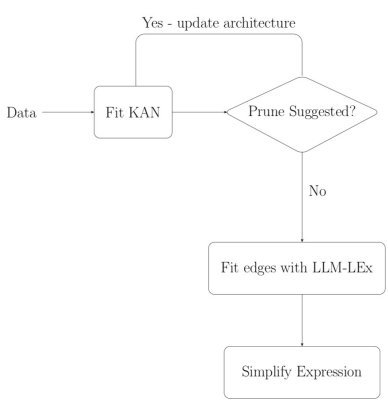
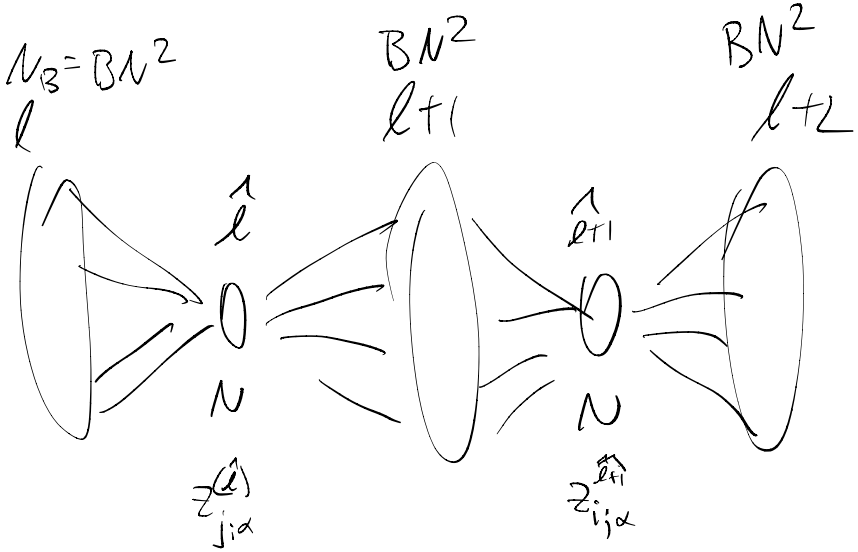


Figure 6: The general structure of KAN-LEx.

MLP SMALL



$$z^{(\ell+2)}_{a;\alpha} = \sum_b W^{(\ell+1)}_{ab}\, \sigma_{b \bmod B}\left(\sum_c W^{(\ell)}_{bc}\, h^{(\ell)}_{c;\alpha} \right)$$

$$\downarrow$$

$$z^{(\ell+2)}_{a;\alpha} = \sum_{i=1}^{N} A^{(\ell+1)}_{ai} \sum_{b=1}^{BN^2} B^{(\ell+1)}_{ib}\, \sigma_{b \bmod B}\left(\sum_{j=1}^{N} A^{(\ell)}_{bj} \sum_{c=1}^{BN^2} B^{(\ell)}_{jc}\, h^{(\ell)}_{c;\alpha} \right)$$

$$\sum_{c=1}^{BN^2} B^{(\ell)}_{jc}\, h^{(\ell)}_{c;\alpha} =: z^{\hat{\ell}}_{j;\alpha}$$

$$\sum_{j=1}^{N} A^{(\ell)}_{bj}\, z^{\hat{\ell}}_{j;\alpha} = z^{(\ell+1)}_{b;\alpha}$$

$$\equiv \sum_{i=1}^{N} A^{(\ell+1)}_{ai}\, z^{\hat{\ell+1}}_{i;\alpha}$$

MLP SMALL $\qquad z^{\hat{\ell+1}}_{i;\alpha} = \sum_{b=1}^{BN^2} B^{(\ell+1)}_{ib}\, \sigma_{b \bmod B}\left(\sum_{j=1}^{N} A^{(\ell)}_{bj}\, z^{\hat{\ell}}_{j;\alpha} \right)$