Tensor programs

Eugene Golikov

May 17, 2021

École Polytechnique Fédérale de Lausanne, Switzerland

Tensor programs series:

1. Formalism:
   - Scaling Limits of Wide Neural Networks with Weight Sharing.
2. Formalism explained: $\leftarrow$ today's talk
   - TP1: Wide Networks of any Architecture are Gaussian Processes.
   - TP2: Neural Tangent Kernel of any Architecture.
   - TP3: Neural Matrix Laws.
3. Applications: $\leftarrow$ Greg's talk
   - Feature Learning in Infinite-Width Neural Networks.

Motivation: maximal-update parameterization

Consider an L-hidden-layer MLP:

$$\underbrace{f = W^{L+1}x^L}_{\text{output}}, \quad \underbrace{x^l = \phi(h^l)}_{\text{activations}}, \quad \underbrace{h^l = W^l x^{l-1}}_{\text{pre-activations}} \quad \forall l \in [L], \tag{1}$$

where $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$ and suppose $n_1 = \ldots = n_L = n$.

Define a **weight update:** $\Delta W_t^l = W_t^l - W_0^l$.

This gives a **hidden representation update:**

$$\Delta h_t^l = \Delta W_t^l x_t^{l-1} + W_0^l \Delta x_t^{l-1}, \tag{2}$$

and an **output update:**

$$\Delta f_t = \Delta W_t^{L+1} x_t^L + W_0^{L+1} \Delta x_t^L. \tag{3}$$

Definition
For $l \in [L+1]$, we say W^l is *updated maximally* if $\Delta W_t^l x_t^{l-1} = \Theta(1)$ $\forall t \geq 1$ and input ξ:

$$\Delta h_t^l = \underbrace{\Delta W_t^l x_t^{l-1}}_{\text{finite as } n \to \infty} + W_0^l \Delta x_t^{l-1}, \tag{4}$$

Definition
We say W^{L+1} is *initialized maximally* if $W_t^{L+1} \Delta x_t^L = \Theta(1)$ $\forall t \geq 1$ and input ξ.

$$\Delta f_t = \underbrace{\Delta W_t^{L+1} x_t^L}_{\substack{\text{finite when} \\ \text{updated maximally}}} + \underbrace{W_0^{L+1} \Delta x_t^L}_{\text{finite as } n \to \infty}. \tag{5}$$

- **Standard parameterization:** $\Delta f_t = \omega(1)$ — output blows up;
- **NTK parameterization:** $W_0^{L+1} \Delta x_t^L = o(1)$ — features are not learned;
- μ**P-paramaterization:** to be discussed below — OK.

Suppose $L = 2$:

$$\underbrace{f = V\bar{x}}_{\text{output}}, \qquad \underbrace{\bar{x} = \phi(\bar{h}),\ \bar{h} = Wx}_{\text{second hidden layer}}, \qquad \underbrace{x = \phi(h),\ h = U\xi}_{\text{first hidden layer}},$$

where $\xi \in \mathbb{R}$, $V \in \mathbb{R}^{1\times n}$, $W \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times 1}$.

Apply **maximal update parameterization (μP):**

$$V = v/\sqrt{n}, \quad W = w, \quad U = \sqrt{n}u,$$

where v, w, and u are the trainable parameters initialized as $v_{1\beta}, w_{\alpha\beta}, u_{\alpha 1} \sim \mathcal{N}(0, 1/n)$.

Compare with other parameterizations:

- **Standard:** $V = v$, $W = w$, $U = u$ initialized as $v_{1\beta}, w_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$, $u_{\alpha 1} \sim \mathcal{N}(0, 1)$.
- **NTK:** $V = v/\sqrt{n}$, $W = w/\sqrt{n}$, $U = u$ initialized as $v_{1\beta}, w_{\alpha\beta}, u_{\alpha 1} \sim \mathcal{N}(0, 1)$.

Substitute V, W, and U with $v/\sqrt{n}$, w, and $\sqrt{n}u$:

$$\underbrace{f = v\bar{x}/\sqrt{n},}_{\text{output}} \qquad \underbrace{\bar{x} = \phi(\bar{h}),\ \bar{h} = wx,}_{\text{second hidden layer}} \qquad \underbrace{x = \phi(h),\ h = \sqrt{n}u\xi,}_{\text{first hidden layer}} \tag{6}$$

Consider an SGD update with batch (x_t, y_t) and learning rate $= 1$:

$$v_{t+1} = v_t - \chi_t \bar{x}_t^\top/\sqrt{n}, \quad w_{t+1} = w_t - \chi_t d\bar{h}_t x_t^\top/n, \quad u_{t+1} = u_t - \chi_t dh_t \xi_t/\sqrt{n}, \tag{7}$$

where

- $\chi_t = \mathcal{L}'(f_t(\xi_t), y_t) \in \mathbb{R}$ — gradient wrt output;
- $dh_t = n\nabla_h f_t(\xi_t)$ — (scaled) gradient wrt pre-activations.

Recall $w = W$ and $W_{t+1} = W_0 + \Delta W_{t+1}$; we get then:

$$\Delta W_{t+1} = \Delta W_t - \chi_t \frac{d\bar{h}_t x_t^\top}{n} = \Delta W_t - \sum_{s=0}^{t} \chi_s \frac{d\bar{h}_s x_s^\top}{n}. \tag{8}$$

Do we update maximally?

$$\bar{h}_{t+1} = W_{t+1}x_{t+1} = W_0 x_{t+1} + \underbrace{\Delta W_{t+1} x_{t+1}}_{\text{needs to be }\Theta(1)} = W_0 x_{t+1} - \sum_{s=0}^{t} \chi_s \underbrace{d\bar{h}_s}_{\substack{\text{a vector with}\\ \Theta(1)\text{ components?}}} \underbrace{\frac{x_s^\top x_{t+1}}{n}}_{\text{a }\Theta(1)\text{ scalar?}} . \tag{9}$$

Do we update maximally?

$$d\bar{h}_s = \Theta_{n\to\infty}(1)? \qquad \frac{x_s^\top x_{t+1}}{n} = \frac{1}{n}\sum_{\alpha=1}^{n} x_{s,\alpha} x_{t+1,\alpha} = \Theta_{n\to\infty}(1)?$$

Yes! Moreover,

- $d\bar{h}_s \to$ a Gaussian with iid components with computable mean and var;
- $\frac{x_s^\top x_{t+1}}{n} \to$ a finite computable scalar.

Tensor programs is a general framework for expressing neural network computations (i.e. forward/backward pass).

The main theoretical result: for a set of "pre-activation-like" vectors $g^1, \ldots, g^M$ of a tensor program satisfying certain initialization assumptions,

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi(g_\alpha^1, \ldots, g_\alpha^M) \to \text{a finite computable scalar}$$

for any "well-behaved" ψ a.s. as $n \to \infty$.

Goals of the talk:

1. Introduce a machinery of tensor programs;
2. Apply it to prove several well-known properties of neural nets as $n \to \infty$.

Convergence to Gaussian processes

Consider a forward pass:

$$\underbrace{h^l = W^l x^{l-1}}_{\text{pre-activations, } \in \mathbb{R}^{n_l}}, \qquad \underbrace{x^{l-1} = \phi(h^{l-1});}_{\text{activations, } \in \mathbb{R}^{n_{l-1}}} \qquad W^l \sim \mathcal{N}\left(0, \frac{\sigma_W^2}{n_{l-1}}\right). \tag{10}$$

In a matrix form:

$$\begin{pmatrix} h_1^l \\ h_2^l \\ \cdots \\ h_{n_l}^l \end{pmatrix} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l 1}^l & W_{n_l 2}^l & \cdots & W_{n_l n_{l-1}}^l \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2 / n_{l-1})} \times \begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \cdots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}.$$

$$
\underbrace{\begin{pmatrix} h_1^l \\ h_2^l \\ \cdots \\ h_{n_l}^l \end{pmatrix}}_{\substack{\text{components tend to Gaussians} \\ \text{as } n_{l-1} \to \infty \text{ by CLT}}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l1}^l & W_{n_l2}^l & \cdots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n_{l-1})} \times \underbrace{\begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \cdots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}}_{\text{suppose all components are iid}} .
$$

$$\underbrace{\begin{pmatrix} h_1^l \\ h_2^l \\ \cdots \\ h_{n_l}^l \end{pmatrix}}_{\text{different components are independent}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l1}^l & W_{n_l2}^l & \cdots & W_{n_l n_{l-1}}^l \end{pmatrix}}_{\text{rows are iid}} \times \underbrace{\begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \cdots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}}_{\text{suppose all components are iid}} .$$

$$\underbrace{\begin{pmatrix} h_1^l \\ h_2^l \\ \dots \\ h_{n_l}^l \end{pmatrix}}_{\substack{\text{tends to a Gaussian vector} \\ \text{with iid components as } n_{l-1} \to \infty \text{ by CLT}}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \dots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \dots & W_{2n_{l-1}}^l \\ \dots & \dots & \dots & \dots \\ W_{n_l1}^l & W_{n_l2}^l & \dots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n_{l-1})} \times \underbrace{\begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \dots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}}_{\text{suppose all components are iid}} .$$

$$\underbrace{\begin{pmatrix} h_1^l & \bar{h}_1^l \\ h_2^l & \bar{h}_2^l \\ \cdots & \cdots \\ h_{n_l}^l & \bar{h}_{n_l}^l \end{pmatrix}}_{\substack{\text{each row converges to} \\ \text{a multivariate Gaussian vector} \\ \text{as } n_{l-1}\to\infty \text{ by the vector CLT}}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l1}^l & W_{n_l2}^l & \cdots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n_{l-1})} \times \underbrace{\begin{pmatrix} x_1^{l-1} & \bar{x}_1^{l-1} \\ x_2^{l-1} & \bar{x}_2^{l-1} \\ \cdots & \cdots \\ x_{n_{l-1}}^{l-1} & \bar{x}_{n_{l-1}}^{l-1} \end{pmatrix}}_{\substack{\text{suppose all components} \\ \text{of each column are iid}}}.$$

Let $\xi_{1:M}$ be a batch of M inputs.

$$\begin{pmatrix} h_1^l(\xi_1) & h_1^l(\xi_2) & \dots & h_1^l(\xi_M) \\ h_2^l(\xi_1) & h_2^l(\xi_2) & \dots & h_2^l(\xi_M) \\ \dots & \dots & \dots & \dots \\ h_{n_l}^l(\xi_1) & h_{n_l}^l(\xi_2) & \dots & h_{n_l}^l(\xi_M) \end{pmatrix}$$

- Batch dimension — tends to a multivariate zero-mean Gaussian as $n_{l-1} \to \infty$;
- Neuron dimension — components tend to iid Gaussians.

Theorem ([Matthews et al., 2018])
Given a model of the form

$$h^{l+1}(\xi) = W^{l+1}x^l(\xi), \quad x^l(\xi) = \phi(h^l(\xi)), \quad h^1(\xi) = W^1\xi, \tag{11}$$

where $W^{l+1} \in \mathbb{R}^{n_{l+1}\times n_l}$, *suppose* $W^{l+1}_{ij} \sim \mathcal{N}(0, \sigma_w^2/n_l)$ *iid.*

Then, $\forall l$ *as* $n_{1:l} \to \infty$ *sequentially,*

1. *all components of* $h^{l+1}(\xi)$ *become iid* $\forall \xi$*, and*
2. $\forall \alpha \in [n_{l+1}]\ \forall M \in \mathbb{N}\ \forall \xi_{1:M}\ \{h^{l+1}_\alpha(\xi_1), \ldots, h^{l+1}_\alpha(\xi_M)\}$ *converges weakly to* $\mathcal{N}(0, \Sigma^{l+1})$*, where*

$$\Sigma^{l+1}_{ij} = \sigma_W^2 \mathbb{E}_{z_{1:M}\sim\mathcal{N}(0,\Sigma^l)}\phi(z_i)\phi(z_j), \tag{12}$$

   and $\Sigma^1_{ij} = \sigma_W^2 \xi_i^T \xi_j$.

Does the previous result hold if some **weights are shared?**

Suppose we want to compute the limit distribution of WWx:

$$\underbrace{\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n2} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} \times \underbrace{\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n2} & \cdots & W_{nn} \end{pmatrix}}_{\text{the same matrix}} \times \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}}_{\text{vector with iid components}} .$$

We can directly apply CLT if WW **has iid components.**

Let us compute the limit distribution of $(WW)_\alpha$:

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{1\alpha} & \cdots & W_{1n} \\ W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n\alpha} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ W_{\alpha\alpha}^2 + \sum(\text{iid w. mean} = 0) \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}^\top}_{\text{all iid except for } \alpha\text{'s term}}$$

Fortunately, since $W_{\alpha\alpha} \sim \mathcal{N}(0, \sigma_W^2/n)$, $W_{\alpha\alpha}^2 x_\alpha \propto 1/n \to 0$.

Conclusion:

1. **CLT is still applicable** even if the same matrix W is used several times;
2. $(WWx)_\alpha$ and $(W\tilde{W}x)_\alpha$ with $\tilde{W} \stackrel{\mathrm{d}}{=} W$ have **the same distribution** in the limit.

However, we **cannot substitute** W with its iid copy $\tilde{W}$:

- WWx and Wx are correlated in the limit;
- $W\tilde{W}x$ and $\tilde{W}x$ are not.

Consider $h = Wx$ and $\bar{h} = W\bar{x}$ where x and $\bar{x}$ **can depend on W.**

$$\underbrace{\begin{pmatrix} h_1 & \bar{h}_1 \\ h_2 & \bar{h}_2 \\ \cdots & \cdots \\ h_n & \bar{h}_n \end{pmatrix}}_{\substack{\text{each row converges to a zero-mean Gaussian} \\ \text{with covariance } \sigma_W^2 \mathbb{E}(x^\top \bar{x})/n \\ \text{as } n \to \infty \text{ by the vector CLT}}} = \underbrace{\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n2} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n)} \times \underbrace{\begin{pmatrix} x_1 & \bar{x}_1 \\ x_2 & \bar{x}_2 \\ \cdots & \cdots \\ x_n & \bar{x}_n \end{pmatrix}}_{\substack{\text{suppose all components} \\ \text{of each column are iid}}}.$$

$$\text{A } \textsc{Netsor} \text{ program} = (\text{a set of input vars}, \text{ a sequence of commands}),$$

where variables are of three different **types:**

1. A-vars: matrices with iid Gaussian entries;
2. G-vars: vectors with *asymptotically* iid Gaussian entries;
3. H-vars: images of G-vars by coordinatewise nonlinearities.

Each command generates a new variable from the previous ones using one of the following **ops:**

1. MatMul: $(W : \mathsf{A},\ x : \mathsf{H}) \to Wx : \mathsf{G}$;
2. $\mathrm{LinComb}$: $(\{x_i : \mathsf{G},\ a_i \in \mathbb{R}\}_{i=1}^k) \to \sum_{i=1}^k a_i x_i : \mathsf{G}$;
3. Nonlin: $(\{x_i : \mathsf{G}\}_{i=1}^k,\ \phi : \mathbb{R}^k \to \mathbb{R}) \to \phi(x_{1:k}) : \mathsf{H}$.

Algorithm 1 Example: MLP with two hidden layers

Input: $W^1x : \mathsf{G}(n^1)$ {layer 1 embedding of input}
Input: $b^1 : \mathsf{G}(n^1)$ {layer 1 bias}
Input: $W^2 : \mathsf{A}(n^2, n^1)$ {layer 2 weights}
Input: $b^2 : \mathsf{G}(n^2)$ {layer 2 bias}
Input: $v : \mathsf{G}(n^2)$ {readout layer weights}
  $h^1 := W^1x + b^1 : \mathsf{G}(n^1)$ {$\mathrm{LinComb}$}
  $x^1 := \phi(h^1) : \mathsf{H}(n^1)$ {layer 1 activation; Nonlin}
  $\tilde{h}^2 := W^2x^1 : \mathsf{G}(n^2)$ {MatMul}
  $h^2 := \tilde{h}^2 + b^2 : \mathsf{G}(n^2)$ {layer 2 preactivation; $\mathrm{LinComb}$}
  $x^2 := \phi(h^2) : \mathsf{H}(n^2)$ {layer 2 activation; Nonlin}
Output: $v^\top x^2/\sqrt{n^2}$

We can absorb $\mathrm{LinComb} + \mathrm{Nonlin}$ into a single Nonlin: $x^2 = \phi(h^2) = \phi(\tilde{h}^2 + b^2) = \bar{\phi}(h^2, b^2)$.

Algorithm 2 Example: MLP with two hidden layers and a batch-norm

Input: $\{W^1 x_k : \mathsf{G}(n^1)\}_{k=1}^{B}$ {layer 1 embeddings of inputs in a batch}
Input: $b^1 : \mathsf{G}(n^1)$ {layer 1 bias}
Input: $W^2 : \mathsf{A}(n^2, n^1)$ {layer 2 weights}
Input: $b^2 : \mathsf{G}(n^2)$ {layer 2 bias}
Input: $v : \mathsf{G}(n^2)$ {readout layer weights}
$\{h_k^1 := W^1 x_k + b^1 : \mathsf{G}(n^1)\}_{k=1}^{B}$ {LinComb}
$\{x_k^1 := \tilde{\phi}_k(h_{1:B}^1) : \mathsf{H}(n^1)\}_{k=1}^{B}$ {BN + activation for layer 1 (see below); Nonlin}
$\{\tilde{h}_k^2 := W^2 x_k^1 : \mathsf{G}(n^2)\}_{k=1}^{B}$ {MatMul}
$\{h_k^2 := \tilde{h}_k^2 + b^2 : \mathsf{G}(n^2)\}_{k=1}^{B}$ {layer 2 preactivation; LinComb}
$\{x_k^2 := \tilde{\phi}_k(h_{1:B}^2) : \mathsf{H}(n^2)\}_{k=1}^{B}$ {BN + activation for layer 2; Nonlin}
Output: $\{v^\top x_k^2 / \sqrt{n^2}\}_{k=1}^{B}$

Here $\tilde{\phi} : \mathbb{R}^B \to \mathbb{R}^B$ is defined as

$$\tilde{\phi}(h^{1:B}) = \phi\left(\frac{h^{1:B} - \mu(h^{1:B})}{\sigma(h^{1:B})}\right), \quad \mu(h^{1:B}) = \frac{1}{B}\sum_{k=1}^{B} h^k, \quad \sigma(h^{1:B}) = \sqrt{\frac{1}{B}\sum_{k=1}^{B}(h^k - \mu(h^{1:B})_k)^2}.$$

Initialization assumption:

1. All hidden dimensions are equal to n;
2. $\forall W$: A we sample $W_{\alpha\beta} \sim \mathcal{N}(0, \sigma_W^2/n)$ iid;

Our goal: compute the distributions of all G-vars in the program in the limit of $n \to \infty$.

Claim:

Let $g^{1:M}$ be a set of all G-vars in the program.

$$\begin{pmatrix} g_1^1 & g_1^2 & \dots & g_1^M \\ g_2^1 & g_2^2 & \dots & g_2^M \\ \dots & \dots & \dots & \dots \\ g_n^1 & g_n^2 & \dots & g_n^M \end{pmatrix}$$

- "Batch" dimension — converges to $\mathcal{N}(\mu, \Sigma)$ with $\mu = \{\mu(g^i)\}_{i=1}^M$ and $\Sigma = \{\Sigma(g^i, g^j)\}_{i,j=1}^M$ defined below;
- Neuron dimension — components tend to iid Gaussians.

$(g_\alpha^1, \ldots, g_\alpha^M)$ becomes jointly Gaussian with mean and covariance **defined by CLT**[1]:

$$\mu(g) = \begin{cases} 0 & \text{if } g = Wy. \end{cases} \tag{13}$$

$$\Sigma(g, \bar{g}) = \begin{cases} \sigma_W^2 \mathbb{E}_Z \phi(Z)\bar{\phi}(Z) & \text{if } g = {\color{red}W}\phi(Z) \text{ and } \bar{g} = {\color{red}W}\bar{\phi}(Z); \\ 0 & \text{else.} \end{cases} \tag{14}$$

Here $Z \sim \mathcal{N}(\mu, \Sigma)$ is a set of all previous G-vars.

[1] we have suppressed the $\mathrm{LinComb}$ op for brevity.

Definition

We say $\phi : \mathbb{R}^k \to \mathbb{R}$ is controlled if $\exists C, c, \epsilon > 0 : \forall x \in \mathbb{R}^k \ |\phi(x)| \leq e^{C\|x\|_2^{2-\epsilon}+c}$.

Theorem (Netsor Master Theorem, [Yang, 2019])

Let the $\textsc{Netsor}$ *program satisfy the initialization assumption and let all nonlinearities be controlled. Let* $g^{1:M}$ *be a set of all G-vars in the program. Then, for any controlled* ψ*,*

$$\underbrace{\frac{1}{n}\sum_{\alpha=1}^{n}\psi(g_\alpha^1, \ldots, g_\alpha^M)}_{\textit{empirical mean over rows}} \to \underbrace{\mathbb{E}_{Z\sim\mathcal{N}(\mu,\Sigma)}\psi(Z)}_{\substack{\textit{expectation} \\ \textit{over the corresponding Gaussian}}} \tag{15}$$

a.s. as $n \to \infty$*, where* $\mu = \{\mu(g^i)\}_{i=1}^M$ *and* $\Sigma = \{\Sigma(g^i, g^j)\}_{i,j=1}^M$.

A NETSOR program

- is able to express the **first forward pass** of a wide class of neural nets (i.e. with shared/structured weights, with BNs etc.);
- reveals its limiting Gaussian process behavior.

Questions:

1. Can we express a **backward pass** as a NETSOR program?
2. What is its limiting behavior?

Intermedia: a Neural Tangent Kernel

Let $f(\cdot;\theta)$ be a parametric model.

Define a **neural tangent kernel** as

$$\Theta_t(\xi, \bar{\xi}) = \underbrace{\nabla_\theta^T f(\xi;\theta_t) \nabla_\theta f(\bar{\xi};\theta_t)}_{\text{"gradient similarity"}}. \tag{16}$$

It drives evolution of **model predictions;** e.g. for square loss:

$$\dot{f}(\bar{\xi};\theta_t) = (\vec{y} - f(\vec{\xi};\theta_t))^\top \Theta_t(\vec{\xi}, \bar{\xi}), \quad \text{where } (\vec{\xi}, \vec{y}) \text{ is a train dataset.}$$

Assuming $\Theta_t(\cdot,\cdot) \approx \Theta_0(\cdot,\cdot)$ makes the dynamics **analytically tractable.**

Theorem ([Jacot et al., 2018], informal)
Suppose we have a feedforward neural net of width n parameterized in a certain way. Then, as $n \to \infty$,

1. $\Theta_0(\xi, \bar{\xi})$ *converges to a deterministic* $\mathring{\Theta}(\xi, \bar{\xi})$*;*
2. *Moreover,* $\Theta_t(\xi, \bar{\xi})$ *converges to the same* $\mathring{\Theta}(\xi, \bar{\xi})$*.*

Parameterize W^l as $\omega^l/\sqrt{n_{l-1}}$:

$$\underbrace{f = \frac{1}{\sqrt{n_L}}\omega^{L+1}x^L}_{\text{the network output}}, \qquad \underbrace{x^l = \phi(h^l)}_{\text{activations}}, \quad \underbrace{h^l = \frac{1}{\sqrt{n_{l-1}}}\omega^l x^{l-1}}_{\text{preactivations}}, \quad l \leq L; \qquad \omega^l_{ij} \sim \mathcal{N}(0, \sigma^2_W) \text{ iid.}$$

NTK is defined as

$$\Theta(\xi, \bar{\xi}) = \nabla^T_\theta f(\xi;\theta)\nabla_\theta f(\bar{\xi};\theta) = \sum_{l=1}^{L+1} \underbrace{\operatorname{tr}(\nabla^T_{\omega^l} f(\xi)\nabla_{\omega^l} f(\bar{\xi}))}_{\text{"layer-wise gradient similarity"}} . \tag{17}$$

$$\Theta(\xi, \bar{\xi}) = \nabla_\theta^T f(\xi; \theta) \nabla_\theta f(\bar{\xi}; \theta) = \sum_{l=1}^{L+1} \underbrace{\mathrm{tr}(\nabla_{\omega^l}^T f(\xi) \nabla_{\omega^l} f(\bar{\xi}))}_{\text{"layer-wise gradient similarity"}} . \tag{18}$$

Weight gradient can be expressed as

$$\nabla_{\omega^l} f = \frac{1}{\sqrt{n_{l-1} n_l}} \underbrace{dh^l}_{\substack{\text{backward pass} \\ \text{up to the layer } l}} \times \underbrace{x^{l-1,\top}}_{\substack{\text{forward pass} \\ \text{up to the layer } l-1}} , \quad \text{where } dh^l \propto \nabla_{h_l} f. \tag{19}$$

Plug (19) into (18):

$$\Theta(\xi, \bar{\xi}) = \sum_{l=1}^{L+1} \underbrace{\left(\frac{dh^{l,\top} d\bar{h}^l}{n_l} \right)}_{\text{"backward pass similarity"}} \times \underbrace{\left(\frac{x^{l-1,\top} \bar{x}^{l-1}}{n_{l-1}} \right)}_{\text{"forward pass similarity"}} . \tag{20}$$

Consider the second multiplier:

$$\frac{x^{l-1,\top}\bar{x}^{l-1}}{n_{l-1}} = \frac{1}{n_{l-1}}\sum_{\alpha=1}^{n_{l-1}} \phi(h_\alpha^{l-1})\phi(\bar{h}_\alpha^{l-1}) = \underbrace{\frac{1}{n_{l-1}}\sum_{\alpha=1}^{n_{l-1}} \psi(h_\alpha^{l-1}, \bar{h}_\alpha^{l-1})}_{\text{the limit is given by the Master theorem!}} \quad \text{for } \psi(x,y) = \phi(x)\phi(y).$$

Can we compute the limit of the first multiplier in the same way?

For simplicity, assume $n_1 = \ldots = n_L = n$. Recall $W^l = \omega^l / \sqrt{n}$.

Relations between forward and backward passes:

Forward pass:	Backward pass:
$x^l = \phi(h^l) : \mathrm{Nonlin}$	$dh^l = dx^l \odot \phi'(h^l) : \mathrm{Nonlin}$
$h^l = W^l x^{l-1} : \mathrm{MatMul}$	$dx^{l-1} = W^{l,\top} dh^l : \mathrm{MatMul?}$

Problems:

1. W and $W^\top$ cannot be both input variables since they are dependent;
2. A Netsor program does not allow for multiplying by a transposed A-var.

A Netsor program cannot express the backward pass!

$$\text{A } \textsc{Netsor}\top \text{ program} = (\text{a set of input vars, a sequence of commands}),$$

where variables are of three different **types:**

1. A-vars: matrices with iid Gaussian entries;
2. G-vars: vectors with *asymptotically* iid Gaussian entries;
3. H-vars: images of G-vars by coordinatewise nonlinearities.

Each command generates a new variable from the previous ones using one of the following **ops:**

1. Trsp: $W : \mathsf{A} \to W^\top : \mathsf{A}$;
2. MatMul: $(W : \mathsf{A},\ x : \mathsf{H}) \to Wx : \mathsf{G}$;
3. LinComb: $(\{x_i : \mathsf{G},\ a_i \in \mathbb{R}\}_{i=1}^k) \to \sum_{i=1}^k a_i x_i : \mathsf{G}$;
4. Nonlin: $(\{x_i : \mathsf{G}\}_{i=1}^k,\ \phi : \mathbb{R}^k \to \mathbb{R}) \to \phi(x_{1:k}) : \mathsf{H}$.

Can we keep the same symbolic rules for mean and covariance of G-vars?

$$\mu(g) = \begin{cases} 0 & \text{if } g = Wy; \\ 0? & \text{if } g = W^\top y. \end{cases} \tag{21}$$

$$\Sigma(g, \bar{g}) = \begin{cases} \sigma_W^2 \mathbb{E}_Z \phi(Z)\bar{\phi}(Z) & \text{if } g = W\phi(Z) \text{ and } \bar{g} = W\bar{\phi}(Z); \\ \text{some other rule?} & \text{if } g = W\phi(Z) \text{ and } \bar{g} = W^\top\bar{\phi}(Z); \\ 0 & \text{else.} \end{cases} \tag{22}$$

Here $Z \sim \mathcal{N}(\mu, \Sigma)$ is a set of all previous G-vars.

Let us check that $\mu(WWx) = 0$:

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0,\sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{1\alpha} & \cdots & W_{1n} \\ W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n\alpha} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0,\sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ W_{\alpha\alpha}^2 + \sum(\text{iid w. mean} = 0) \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}^\top}_{\text{all iid except for } \alpha\text{'s term}}$$

$$\mu(WWx) = \mathbb{E}\left((WWx)_\alpha\right) = \underbrace{\mathbb{E}\left(\sum_{\beta\neq\alpha}\sum_\gamma W_{\alpha\beta}W_{\beta\gamma}x_\gamma\right)}_{\mathbb{E}\left(\sum \text{iid w. mean}=0\right)=0} + \underbrace{\mathbb{E}\left(\sum_\gamma W_{\alpha\alpha}W_{\alpha\gamma}x_\gamma\right)}_{=\mathbb{E}\,W_{\alpha\alpha}^2 x_\alpha \propto 1/n \to 0}. \tag{23}$$

Do we have $\mu(WW^Tx) = 0$?

Let us compute $\mu(WW^T x)$:

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0,\sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{\alpha 1} & \cdots & W_{n1} \\ W_{1\alpha} & W_{\alpha\alpha} & \cdots & W_{n\alpha} \\ \cdots & \cdots & \cdots & \cdots \\ W_{1n} & W_{\alpha n} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0,\sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ \sum_\beta W_{\alpha\beta}^2 \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}^\top}_{\text{all iid except for } \alpha\text{'s term}}$$

$$\mu(WW^\top x) = \mathbb{E}\left((WW^\top x)_\alpha\right) = \underbrace{\mathbb{E}\left(\sum_\beta \sum_{\gamma\neq\alpha} W_{\alpha\beta} W_{\gamma\beta} x_\gamma\right)}_{\mathbb{E}\,(\sum \text{iid w. mean}=0)=0} + \underbrace{\mathbb{E}\left(\sum_\beta W_{\alpha\beta}^2 x_\beta\right)}_{\text{converges to } \sigma_W^2 \mu(x) \neq 0} . \quad (24)$$

The previous symbolic rules are not applicable for **general** $\textsc{Netsor}\top$ programs,

but

they are applicable to $\textsc{Netsor}\top$ programs expressing backpropagation.

Claim: the rule

$$\mu(g) = 0 \quad \text{if } g = Wy. \tag{25}$$

works for NETSORT programs expressing backpropagation.

Evidence: consider $dx^{l-1} = W^{l,T}(dx^l \odot \phi'(h^l))$. Let $\phi(z) = z^2/2$:

$$\mu(dx^{l-1}) = \mathbb{E}\left(dx_\alpha^{l-1}\right) = \underbrace{\mathbb{E}\left(\sum_\beta W^l_{\beta\alpha} dx^l_\beta \sum_{\gamma\neq\beta} W^l_{\beta\gamma} x^{l-1}_\gamma\right)}_{\mathbb{E}(\sum \text{iid w. mean=0})=0} + \underbrace{\mathbb{E}\left(x^{l-1}_\alpha \sum_\beta (W^l_{\beta\alpha})^2 dx^l_\beta\right)}_{\text{converges to } \mu(x^{l-1})\sigma_W^2 \mu(dx^l)}. \tag{26}$$

Hence $\mu(dx^{l-1}) \propto \mu(dx^l)$ which by induction implies $\mu(dx^{l-1}) \propto \mu(dx^L)$.

But $dx^L = \omega^{L+1}$! Hence $\mu(dx^{l-1}) = \mu(\omega^{L+1}) = 0$.

Proposition ([2])
Consider a neural network and a NETSOR⊤ *program expressing its backward pass.*

The symbolic rules for μ *and* Σ *are valid, if*

1. *The output layer has zero mean;*
2. *It is sampled independently from other parameters;*
3. *It is not used anywhere else in the program.*

[2] There is a more general condition called "BP-likeness" which we do not show here.

Definition
We say $\phi : \mathbb{R}^k \to \mathbb{R}$ is polynomially bounded if $\exists C, c, p > 0 : |\phi(x)| \leq C\|x\|_2^p + c$.

Theorem (Netsor$\top$ Master Theorem, [Yang, 2020a])
Let a $\textsc{Netsor}\top$ *program be* **BP-like** *in a neural network, satisfy the initialization assumption, and let all nonlinearities be polynomially bounded. Let* $g^{1:M}$ *be a set of all* G-vars *in the program. Then, for any polynomially bounded* $\psi : \mathbb{R}^M \to \mathbb{R}$,

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi(g_\alpha^1, \ldots, g_\alpha^M) \to \mathbb{E}_{Z\sim\mathcal{N}(\mu,\Sigma)}\psi(Z)$$

a.s. as $n \to \infty$, *where* $\mu = \{\mu(g^i)\}_{i=1}^M$ *and* $\Sigma = \{\Sigma(g^i, g^j)\}_{i,j=1}^M$.

Back to NTK computation:

$$\Theta(\xi, \bar{\xi}) = \sum_{l=1}^{L+1} \nabla_{\omega^l}^T f(\xi) \nabla_{\omega^l} f(\bar{\xi}) = \sum_{l=1}^{L+1} \left(\frac{dh^{l,\top} d\bar{h}^l}{n_l} \right) \left(\frac{x^{l-1,\top} \bar{x}^{l-1}}{n_{l-1}} \right). \tag{27}$$

Consider the first multiplier:

$$\frac{dh^{l,\top} \bar{d}h^l}{n_l} = \frac{1}{n_l} \sum_{\alpha=1}^{n_l} dx_\alpha^l d\bar{x}_\alpha^l \phi'(h_\alpha^l) \phi'(\bar{h}_\alpha^l) = \underbrace{\frac{1}{n_l} \sum_{\alpha=1}^{n_l} \psi(dx_\alpha^l, d\bar{x}_\alpha^l, h_\alpha^l, \bar{h}_\alpha^l)}_{\text{the limit is given by the Master theorem!}}$$

$$\text{for } \psi(x, y, z, w) = xy\phi'(z)\phi'(w). \tag{28}$$

A BP-like NETSOR⊤ program

- is able to express the **first forward and backward passes** of a wide class of neural nets (i.e. with shared/structured weights, with BNs etc.);
- reveals their limiting Gaussian process behavior;
- can be applied to initial NTK computation.

Intermedia 2: the limiting jacobian spectrum

Consider a forward pass:

$$\underbrace{h^l = W^l x^{l-1}}_{\text{pre-activations, } \in \mathbb{R}^{n_l}}, \qquad \underbrace{x^{l-1} = \phi(h^{l-1})}_{\text{activations, } \in \mathbb{R}^{n_{l-1}}}. \tag{29}$$

Define an **input-output jacobian:**

$$J = \frac{\partial h^L}{\partial h^1} = W^L D^{L-1} \dots W^2 D^1 \in \mathbb{R}^{n_L \times n_1}, \quad D^l = \mathrm{diag}(\phi'(h^l)).$$

We are interested in singular values of J, or, equivalently, **eigenvalues of $J^\top J$.**

Assume $n_L = \ldots = n_1 = n$.

Define the **empirical spectral distribution:**

$$\hat{\rho}_{J^\top J} = \frac{1}{n} \sum_{k=1}^{n} \delta_{\lambda_k}, \quad \text{where } \lambda_{1:n} \text{ are eigenvalues of } J^\top J.$$

Claim: if $\mathbb{E}\, W^l = 0$ and $\mathbb{V}\mathrm{ar}\, W^l \propto n^{-1}$ then $\exists \lim_{n\to\infty} \hat{\rho}_{J^\top J} =: \rho_{J^\top J}$.

$$J = \frac{\partial h^L}{\partial h^1} = W^L D^{L-1} \dots W^2 D^1 \in \mathbb{R}^{n_L \times n_1}, \quad D^l = \mathrm{diag}(\phi'(h^l)).$$

For $L = 2$ and $\phi = \mathrm{id}$, $J = W$.

Given $W_{ij} \sim \mathcal{N}(0, n^{-1})$, $J^\top J = W^\top W$ is known as a **Wishart ensemble.**

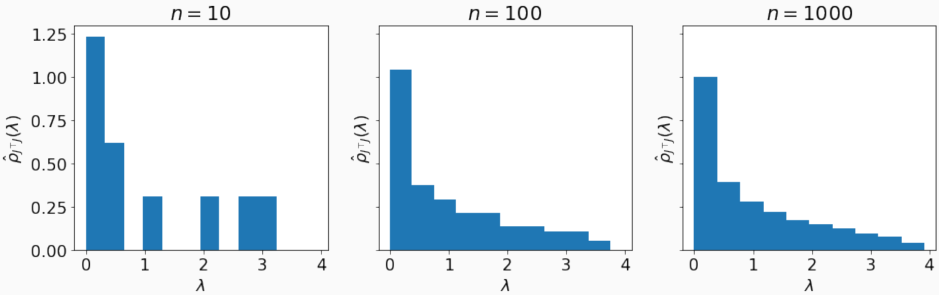

Figure 1: $\hat{\rho}_{J^\top J}(\lambda)$ for a Wishart ensemble: $J = W$, $W_{ij} \sim \mathcal{N}(0, n^{-1})$.

$$J = \frac{\partial h^L}{\partial h^1} = W^L D^{L-1} \ldots W^2 D^1 \in \mathbb{R}^{n_L \times n_1}, \quad D^l = \mathrm{diag}(\phi'(h^l)).$$

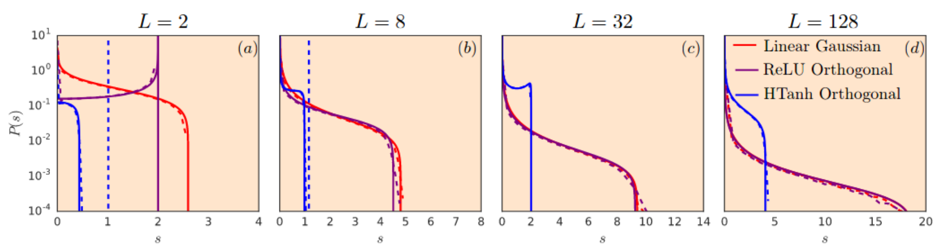

Figure 2: The limiting spectrum $\rho_{J^\top J}$ depends on L![3]

[3]The figure is borrowed from [Pennington et al., 2017].

Good situation: $\rho_{J^\top J}$ neither diffuses nor squeezes to zero as $L \to \infty$.

Why is it good?

- It allows for training very deep nets! [Pennington et al., 2017, Xiao et al., 2018]

For **linear nets,** this can be achieved with orthogonal weight initialization:

$$J^\top J = I \text{ when } J = W^L \dots W^2 \text{ and } W^l \in O_{n\times n}\, \forall l \qquad \Rightarrow \qquad \rho_{J^\top J} = \delta_1.$$

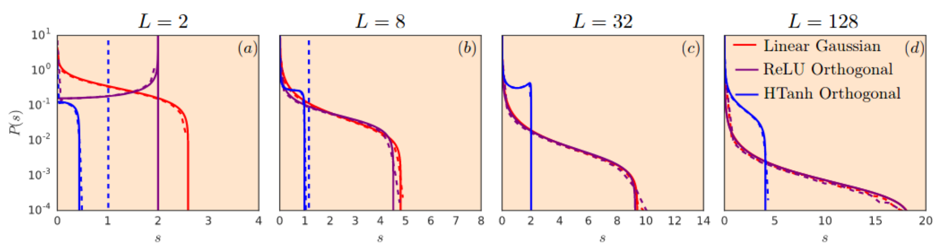

Figure 3: Linear nets with iid Gaussian initialization ($W^l_{ij} \sim \mathcal{N}(0, 1/n)$) give $\lambda_{\max} = (L+1)^{L+1}/L^L \sim eL$ — **diffuses with L!**

How to compute $\rho_{J^\top J}$?

- **Claim:** it suffices to compute the moments:

$$\tau((J^\top J)^k) = \frac{1}{n}\mathbb{E}\,\mathrm{tr}((J^\top J)^k). \tag{30}$$

- **Analogy:** probability density $p_X(x)$ of a scalar random variable X is expressible in terms of its moments $\mathbb{E}\,X^k$:

$$M_X(t) = \mathbb{E}_X e^{tX} = \sum_{k=0}^{\infty} \frac{t^k \mathbb{E}\,X^k}{k!}; \qquad p_X(x) = \int_{-\infty}^{\infty} M_X(it)e^{itx}\,dt. \tag{31}$$

How to compute $\mathrm{tr}((J^\top J)^k)/n$**?** Use the identity:

$$\mathrm{tr}(X) = \mathbb{E}_{v \sim \mathcal{N}(0,I)}(v^\top X v). \tag{32}$$

This gives:

$$\frac{1}{n}\mathrm{tr}((J^\top J)^k) = \mathbb{E}_{v \sim \mathcal{N}(0,I)} \underbrace{\left(\frac{1}{n} \sum_{\alpha=1}^{n} v_\alpha ((J^\top J)^k v)_\alpha \right)}_{\text{the limit is given by the Master theorem?}} . \tag{33}$$

Indeed, $(J^\top J)^k v$ is expressible as a G-var in some NETSOR⊤ program[4].

But does the Master theorem work?

[4] $D^1 v = \phi'(h^1) \odot v$ is given by Nonlin, $W^2 D^1 v$ is given by MatMul, and so on.

Consider a Wishart ensemble: $J = W$.

Let us compute $\mathbb{E}\, v_\alpha (WW^\top v)_\alpha$.

We have the following symbolic rule:

$$\Sigma(g, \bar{g}) = \begin{cases} \sigma_W^2 \mathbb{E}_Z \phi(Z) \bar{\phi}(Z) & \text{if } g = W\phi(Z) \text{ and } \bar{g} = W\bar{\phi}(Z); \\ 0 & \text{else.} \end{cases} \tag{34}$$

According to this rule,

$$\mathbb{E}\, v_\alpha (WW^\top v)_\alpha = \Sigma(v, WW^\top v) = 0.$$

We expect to have $\mathbb{E}\, v_\alpha (WW^\top v)_\alpha = 0$.

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{\alpha 1} & \cdots & W_{n1} \\ W_{1\alpha} & W_{\alpha\alpha} & \cdots & W_{n\alpha} \\ \cdots & \cdots & \cdots & \cdots \\ W_{1n} & W_{\alpha n} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ \sum_\beta W_{\alpha\beta}^2 \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}}_{\text{all iid except for } \alpha\text{'s term}}^\top$$

$$v_\alpha (WW^\top v)_\alpha = v_\alpha \sum_{\beta,\gamma} W_{\alpha\beta} W_{\gamma\beta} v_\gamma = v_\alpha \underbrace{\sum_\beta W_{\alpha\beta} \sum_{\gamma \neq \alpha} W_{\gamma\beta} v_\gamma}_{\text{converges to a Gaussian indep. of } v_\alpha} + \underbrace{\sum_\beta (W_{\alpha\beta})^2 v_\alpha^2}_{\text{converges to } \sigma_W^2 v_\alpha^2}.$$

Hence $\mathbb{E}\, v_\alpha (WW^\top v)_\alpha = \sigma_W^2$!

The previous rule:

$g = W\phi(Z)$ tends to a Gaussian with iid components that is correlated **only** with G-vars of the form $W\bar{\phi}(Z)$:

$$\Sigma(g, \bar{g}) = \begin{cases} \sigma_W^2 \mathbb{E}_Z \phi(Z)\bar{\phi}(Z) & \text{if } g = W\phi(Z) \text{ and } \bar{g} = W\bar{\phi}(Z); \\ 0 & \text{else.} \end{cases} \tag{35}$$

The corrected rule:

$g = W\phi(Z)$ tends to a sum of **two terms:**

1. A **Gaussian** with iid components that is correlated only with G-vars of the form $W\bar{\phi}(Z)$;
2. A **linear combination** of previously-generated vectors.

Example:

$$(WW^\top v)_\alpha = \sum_\beta W_{\alpha\beta} \sum_{\gamma\neq\alpha} W_{\gamma\beta} v_\gamma + \sum_\beta (W_{\alpha\beta})^2 {\color{red} v_\alpha} \to \underbrace{g_\alpha}_{\substack{\text{Gaussian;} \\ \text{does not correlate with } v_\alpha}} + \underbrace{\sigma_W^2 {\color{red} v_\alpha}}_{\text{correlates with } v_\alpha} .$$

Back to learning a 2-layer perceptron!

An illustration for W_0 — $W_0^\top$ **interaction:**

Suppose $\phi(z) = z$:

$$\bar{h}_1 = W_0 x_1 - \chi_0 d\bar{h}_0 \frac{x_0^\top x_1}{n},$$

$$x_1 = h_1 = \sqrt{n} u_1 \xi_1 = u_0 \xi_1 - \chi_0 dh_0 \xi_0 \xi_1 = u_0 \xi_1 - \chi_0 \xi_0 \xi_1 W_0^\top d\bar{h}_0.$$

$$W_0 x_1 = W_0 u_0 \xi_1 - \chi_0 \xi_0 \xi_1 W_0 W_0^\top d\bar{h}_0.$$

Hence **for** $t \geq 1$ **a Netsor$\top$ program is not BP-like!**

Theorem (Netsor$\top$ Master Theorem, [Yang, 2020b])
Let a **general** $\textsc{Netsor}\top$ *program satisfy the initialization assumption, and let all nonlinearities be polynomially bounded. Let $g^{1:M}$ be a set of all G-vars in the program. Then, for any polynomially bounded $\psi : \mathbb{R}^M \to \mathbb{R}$,*

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi(g_\alpha^1, \ldots, g_\alpha^M) \to \mathbb{E}_{Z\sim\mathcal{N}(\mu,\Sigma)}\psi(Z)$$

a.s. as $n \to \infty$, where $\mu = \{\mu(g^i)\}_{i=1}^M$ and $\Sigma = \{\Sigma(g^i, g^j)\}_{i,j=1}^M$, and $\mu(g)$ and $\Sigma(g, \bar{g})$ are computed according to the **corrected rules** *(which we do not write here).*

Jacot, A., Gabriel, F., and Hongler, C. (2018).
Neural tangent kernel: Convergence and generalization in neural networks.
In *Advances in neural information processing systems*, pages 8571–8580.

Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018).
Gaussian process behaviour in wide deep neural networks.
In *International Conference on Learning Representations*.

Pennington, J., Schoenholz, S., and Ganguli, S. (2017).
Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice.
In *Advances in neural information processing systems*, pages 4785–4795.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., and Pennington, J. (2018).
Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks.
In *International Conference on Machine Learning*, pages 5393–5402. PMLR.

Yang, G. (2019).

Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes.
arXiv preprint arXiv:1910.12478.

Yang, G. (2020a).
Tensor programs ii: Neural tangent kernel for any architecture.
arXiv preprint arXiv:2006.14548.

Yang, G. (2020b).
Tensor programs iii: Neural matrix laws.
arXiv preprint arXiv:2009.10685.