1 Statement

The goal of this analysis is to understand the shared representations model.

We have words in a dictionary:

$$\mathcal{D} \triangleq \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_V\} \subseteq \mathbb{R}^D,$$

and a set of L languages, indexed by $[L] = \{1, \dots, L\}$. For simplicity, we make the following assumption:

Assumption 1. *The dictionary $\mathcal{D}$ is a set of one-hot vectors that span $\mathbb{R}^D$. So, $V = D$.*

We train a gated deep linear network (GDLN) to "translate" words from one language into another. The gated deep linear network is represented by a collection of L input and output weight matrices of dimensions $D \times H$ and $H \times D$, respectively, and a hidden layer of size $H \times H$. The forward pass of the GDLN is given by:

$$\hat{y}(\boldsymbol{x}) = \sum_{l' \in [L]} g_{l'}(\boldsymbol{x}) \boldsymbol{W}_o^{l'} \boldsymbol{W}_h \left(\sum_{l \in [L]} g_l(\boldsymbol{x}) \boldsymbol{W}_i^l \boldsymbol{x} \right) = \sum_{l, l' \in [L]} g_l(\boldsymbol{x}) g_{l'}(\boldsymbol{x}) \boldsymbol{W}_o^{l'} \boldsymbol{W}_h \boldsymbol{W}_i^l \boldsymbol{x}.$$

Here, the input $\boldsymbol{x}$ contains a one-hot vector for the word we are translating, and an input-output language pair. To make this more explicit, we will now write inputs as $(\boldsymbol{x}, p)$, where $\boldsymbol{x}$ is the one-hot word and $p = (k, k')$ is the language pair, where $k, k' \in [L]$ are the languages of the input and output, respectively. Thus, we write,

$$\hat{y}(\boldsymbol{x}, p) = \underbrace{\left[\sum_{l, l' \in [L]} g_l(p) g_{l'}(p) \boldsymbol{W}_o^{l'} \boldsymbol{W}_h \boldsymbol{W}_i^l \right]}_{\triangleq \boldsymbol{A}_p} \boldsymbol{x},$$

where $\boldsymbol{A}_p$ is a matrix of size $V \times V$ that depends only on the input via the language pair p.

During training, we will sample input-output pairs p uniformly from a subset of the language pairs, which we denote $T \subseteq [L]^2$. We will equivalently use T to denote the cardinality of the set T, and $\boldsymbol{T}$ to denote a binary matrix, where $\boldsymbol{T}_{k,k'} = 1$ if $(k, k') \in T$ and 0 otherwise. The gates operate on p as follows: $g_l(p) = \mathbb{1}(l = k)$ and $g_{l'}(p) = \mathbb{1}(l' = k')$.

We study *full-batch gradient flow*. For simplicity, we will assume that $y(\boldsymbol{x}_v, p) \equiv \boldsymbol{x}_{\pi_p(v)}$ where π_p is a permutation on $[L]$ corresponding to language pair p, i.e., the target is another one-hot vector in the dictionary $\mathcal{D}$[1]. In particular, we assume $\pi_p = \pi_{k'} \circ \pi_k^{-1}$, where π_k is a permutation on the input language k and $\pi_{k'}$ is a permutation on the output language k'. Intuitively, this maps each word in the input language to a word to an abstract representation, and then maps that abstract representation to a word in the output language. Since these are one-hot vectors (by assumption 1), we can also write $\boldsymbol{x}_{\pi_p(v)} = \boldsymbol{\Pi}_p \boldsymbol{x}_v = \boldsymbol{\Pi}_{k'} \boldsymbol{\Pi}_k^\top \boldsymbol{x}_v$, where $\boldsymbol{\Pi}_k$ is the permutation matrix corresponding to π_k.

[1] I believe this is how Andrew set it up.

With this structure, we can write the loss function as:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},p\sim T}\left[\frac{1}{2}\|\hat{y}(\boldsymbol{x},p)-y(\boldsymbol{x},p)\|^2\right]\\ &= \frac{1}{2}\,\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},p\sim T}\left[\|\boldsymbol{A}_p\boldsymbol{x}-\boldsymbol{\Pi}_p\boldsymbol{x}\|^2\right]\\ &= \frac{1}{2}\,\mathbb{E}_{p\sim T}\left[\frac{1}{D}\sum_{i\in[D]}\left[\|(\boldsymbol{A}_p-\boldsymbol{\Pi}_p)\boldsymbol{e}_i\|^2\right]\right]\\ &= \frac{1}{2D}\,\mathbb{E}_{p\sim T}\left[\|\boldsymbol{A}_p-\boldsymbol{\Pi}_p\|_F^2\right].\end{aligned}$$

Expanding $\boldsymbol{A}_p$ and summing over p gives:

$$\mathcal{L} = \frac{1}{2DT}\sum_{(k,k')\in T}\left\|\sum_{l,l'\in[L]} g_l(p)g_{l'}(p)\boldsymbol{W}_o^{l'}\boldsymbol{W}_h\boldsymbol{W}_i^{l}-\boldsymbol{\Pi}_{k'}\boldsymbol{\Pi}_k^\top\right\|_F^2.$$

We use block matrix notation to clean this up; define:

$$\begin{aligned}\boldsymbol{W}_o &= \begin{pmatrix}\boldsymbol{W}_o^1\\ \vdots\\ \boldsymbol{W}_o^L\end{pmatrix}\in\mathbb{R}^{LH\times D},\\ \boldsymbol{W}_i &= \begin{pmatrix}\boldsymbol{W}_i^1 & \dots & \boldsymbol{W}_i^L\end{pmatrix}\in\mathbb{R}^{D\times LH},\\ \boldsymbol{\Pi} &= \begin{pmatrix}\boldsymbol{\Pi}_1\\ \vdots\\ \boldsymbol{\Pi}_L\end{pmatrix}\in\mathbb{R}^{LD\times D},\end{aligned}$$

Then, using $\boldsymbol{J}_L$ to denote the $L\times L$ matrix of all ones, we can write:

$$\mathcal{L} = \frac{1}{2DT}\left\|(\boldsymbol{W}_o\boldsymbol{W}_h\boldsymbol{W}_i-\boldsymbol{\Pi}\boldsymbol{\Pi}^\top)\odot(\boldsymbol{T}\otimes\boldsymbol{J}_L)\right\|_F^2,$$

where $\otimes$ is the Kronecker product and $\odot$ is the Hadamard product.

I believe Andrew said we could make the following, additional, assumption:

Assumption 2. *The permutation matrices $\boldsymbol{\Pi}_k$ can all be taken to be the identity matrix, i.e. $\boldsymbol{\Pi}_k = \boldsymbol{I}_D$ for all $k\in[L]$.*

Under assumption 2, the loss function simplifies to:

$$\mathcal{L} = \frac{1}{2DT}\left\|(\boldsymbol{W}_o\boldsymbol{W}_h\boldsymbol{W}_i-\boldsymbol{J}_L\otimes\boldsymbol{I}_L)\odot(\boldsymbol{T}\otimes\boldsymbol{J}_L)\right\|_F^2. \tag{1}$$

This is the loss function we simulate below. It is worth noting that we would have obtained the same loss function if we had used $\boldsymbol{x}\sim\mathcal{N}(0,\frac{1}{D}I_D)$ instead of one-hot vectors.

2 Simulations

To validate and gain insight our model, we simulate the loss function in equation (1). We sample $\boldsymbol{T}$ by fixing the diagonal to be 1 and randomly making off-diagonal entries 1 until we achieve a desired fraction of 1s;

all other entries are set to 0[2]. We use JAX to run these experiments, giving us much greater control over randomness and intialization. In particular, for each seed, when increasing the number of 1s by one over a previous experiment, we do not resample the entire matrix $\boldsymbol{T}$, but rather we only sample one new entry. Additionally, when change the initialization scale σ, we do not resample the entire matrix $\boldsymbol{W}_o\boldsymbol{W}_h\boldsymbol{W}_i$, but rather we only rescale the entries of those matrices.

Figure 1 shows the results of our simulations. We see that generalization typically emerges after roughly 30% of the pairs have been trained on. However, this is not always the case, and we do see spikes in test loss for some experiments after this point (my initial guess is that this is numerical, but I could also believe that these simulations are highly sensitive to the structure of $\boldsymbol{T}$, though I believe Andrew said he did not see this). Additionally, the generalization threshold appears to *decrease* as the initialization scale σ decreases, which suggests that, as $\sigma \to 0$, the generalization threshold may go to zero as well, a counterintuitive result. Morever, we see considerable variation across random seeds.

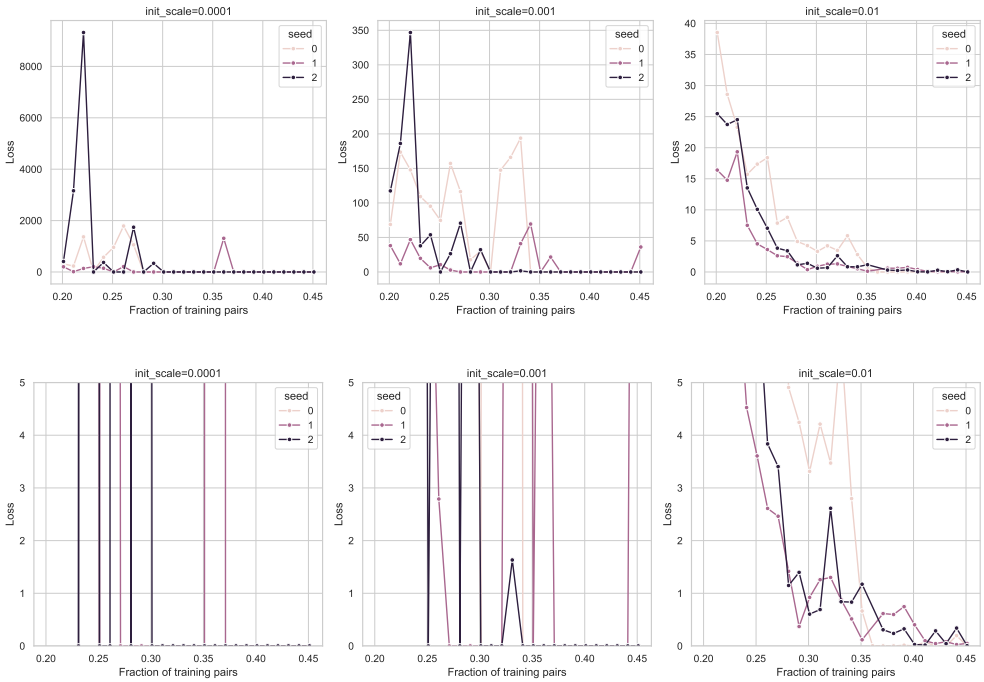

Figure 1: (a) MSE loss on *untrained* language pairs after gradient flow achieving a value of 10^{-6} on the loss $\mathcal{L}$ from equation (1) for various σ and three random seeds. (b) Same setup as in (a), but the loss is truncated at $\mathcal{L} = 5$ to show spikes later in training. The model is a GDLN with $L = 10$, $H = 32$, and $D = 32$. Each point represents a single training run.

[2]I don't see any inherent reason to make the diagonal all 1s; this is something I would change, but I had sunk too much compute into these sims by the time I realized this.

3 A Simple Case

The block structure of the loss suggests we may be able to gain insight from studying a simpler, non-block, analogue:

$$\mathcal{L} = \frac{1}{2} \left\| (\boldsymbol{u}s\boldsymbol{v}^\top - \boldsymbol{J}_n) \odot \boldsymbol{T} \right\|_F^2. \tag{2}$$

Here, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ and $s \in \mathbb{R}$ are learnable parameters and $\boldsymbol{T} \in \{0,1\}^{n \times n}$ is a binary matrix. We would obtain a similar expression if we diagonalized the matrices in the loss function in equation (1).

The gradients of the loss function are given by:

$$\nabla_{\boldsymbol{u}} \mathcal{L} = s\boldsymbol{E}\boldsymbol{v}, \quad \nabla_s \mathcal{L} = \langle \boldsymbol{E}, \boldsymbol{u}\boldsymbol{v}^\top \rangle_F, \quad \nabla_{\boldsymbol{v}} \mathcal{L} = s\boldsymbol{E}^\top \boldsymbol{u},$$

where $\boldsymbol{E} \triangleq (\boldsymbol{u}s\boldsymbol{v}^\top - \boldsymbol{J}_L) \odot \boldsymbol{T}$ is the "error" matrix. This implies that gradient flow (under unit learning rate) is:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = -s\boldsymbol{E}\boldsymbol{v}, \quad \frac{\mathrm{d}s}{\mathrm{d}t} = -\langle \boldsymbol{E}, \boldsymbol{u}\boldsymbol{v}^\top \rangle_F, \quad \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = -s\boldsymbol{E}^\top \boldsymbol{u}.$$

Consider the behavior of the model when σ, the initialization scale, is small. Then, we have $\boldsymbol{E} \approx -\boldsymbol{J}_L \odot \boldsymbol{T} = \boldsymbol{T}$. So,

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} \approx s\boldsymbol{T}\boldsymbol{v}, \quad \frac{\mathrm{d}s}{\mathrm{d}t} \approx \langle \boldsymbol{T}, \boldsymbol{u}\boldsymbol{v}^\top \rangle_F, \quad \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} \approx s\boldsymbol{T}^\top \boldsymbol{u}.$$

Consider the SVD of $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. We write

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} \approx s\boldsymbol{T}\boldsymbol{v} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \boldsymbol{v} \implies \frac{\mathrm{d}\boldsymbol{U}^\top \boldsymbol{u}}{\mathrm{d}t} \approx \boldsymbol{\Sigma}\boldsymbol{V}^\top \boldsymbol{v}.$$

Similarly,

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} \approx s\boldsymbol{T}^\top \boldsymbol{v} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u} \implies \frac{\mathrm{d}\boldsymbol{V}^\top \boldsymbol{v}}{\mathrm{d}t} \approx \boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u}.$$

Defining $\tilde{\boldsymbol{u}} \triangleq \boldsymbol{U}^\top \boldsymbol{u}$ and $\tilde{\boldsymbol{v}} \triangleq \boldsymbol{V}^\top \boldsymbol{v}$, we have:

$$\begin{aligned}
\frac{\mathrm{d}\tilde{\boldsymbol{u}}}{\mathrm{d}t} &\approx s\boldsymbol{\Sigma}\tilde{\boldsymbol{v}}, \\
\frac{\mathrm{d}s}{\mathrm{d}t} &\approx \langle \boldsymbol{T}, \boldsymbol{u}\boldsymbol{v}^\top \rangle_F = \langle \boldsymbol{\Sigma}, \tilde{\boldsymbol{u}}\tilde{\boldsymbol{v}}^\top \rangle_F, \\
\frac{\mathrm{d}\tilde{\boldsymbol{v}}}{\mathrm{d}t} &\approx s\boldsymbol{\Sigma}\tilde{\boldsymbol{u}}.
\end{aligned}$$

Focusing just on $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$, we have:

$$\begin{aligned}
\tilde{\boldsymbol{u}}(t) &= \cosh(\boldsymbol{\Sigma}S(t))\tilde{\boldsymbol{u}}(0) + \sinh(\boldsymbol{\Sigma}S(t))\tilde{\boldsymbol{v}}(0), \\
\tilde{\boldsymbol{v}}(t) &= \sinh(\boldsymbol{\Sigma}S(t))\tilde{\boldsymbol{u}}(0) + \cosh(\boldsymbol{\Sigma}S(t))\tilde{\boldsymbol{v}}(0),
\end{aligned}$$

where $S(t) \triangleq \int_0^t s(t')dt'$ and functions f are applied to matrices $\boldsymbol{X}$ via their singular values, i.e. $f(\boldsymbol{X}) = \boldsymbol{U}f(\boldsymbol{\Sigma})\boldsymbol{V}^\top$; so, in this case, we only apply it along the diagonal of $\boldsymbol{\Sigma}$. Thus, assuming $S(t) \to \infty$ as $t \to \infty$ (which clearly must be the case), we see that $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ will align, since $\cosh(x) \sim \sinh(x) \sim e^x$ for large x. [3]

[3] I expect this happens around time $t \sim -\log(\sigma)$ as $\sigma \to 0$.

Moreover, it is known that for $\boldsymbol{T}$ with entries i.i.d. Bern(p), there is an outlier singular value on the order of pn, while the rest follow the Marchenko-Pastur law, which concentrate in the interval $[0, 2\sqrt{p(1-p)}]$. Assuming we have a $\boldsymbol{T}$ of this form, the largest singular value will dominate the dynamics of the system *by a lot*. In fact, as σ is sufficiently small, we will always see that the largest singular value dominates because the hyperbolic functions grow exponentially. Therefore, $\tilde{\boldsymbol{u}} \approx c_u \boldsymbol{e}_1, \tilde{\boldsymbol{v}} \approx c_v \boldsymbol{e}_1$, which implies $\boldsymbol{u} \approx c_u \boldsymbol{u}_*$ and $\boldsymbol{v} \approx c_v \boldsymbol{v}_*$, where $\boldsymbol{u}_*, \boldsymbol{v}_*$ are the first columns of $\boldsymbol{U}, \boldsymbol{V}$, respectively, i.e. the first left and right singular vectors of $\boldsymbol{T}$.

This means that $\boldsymbol{u}s\boldsymbol{v}^\top$ emerges to be the best rank-1 approximation of $\boldsymbol{T}$, in Frobenius norm, up to a scaling factor. Note that if $\boldsymbol{u} = \boldsymbol{v} = \boldsymbol{1}$, the vector of all 1s, then we immediately obtain perfect generalization. This would happen, for instance, if $\boldsymbol{T}$ is a non-identity circulant matrix.

(So, does this mean any such circulant $\boldsymbol{T}$ *should* generalize perfectly, asymptotically as $\sigma \to 0$? This seems surprising ...)

I tried to test this observation, and it generally seems to be true, but I can't make the initialization scale small enough to get it to always work. It seems like the gap between the largest singular value and the rest needs to be sufficiently large, but mathematically, I don't see why this should be the case. As the circulant width increases, this gap appears to grow, which supports why we only see this for wider circulant matrices.