Moduli Spaces and Pruning

Maria Walch

June 2022

1 Moduli Space of Quiver Representations

1.1 Representation of a Quiver

A quiver Q is a tuple (Q_0, Q_1, t, h) consisting of a finite set $Q_0 = \{1, ..., n\}$ of vertices, a finite set $Q_1 = \{\phi\}$ of arrows between these vertices and two maps $t, h : Q_1 \to Q_0$ assigning to an arrow ϕ its tail $t(\phi)$ and head $h(\phi)$ respectively. Let k be a field. A representation V of a quiver Q is a family $\{V(i) : i \in Q_0\}$ of finite dimensional vector spaces over k together with a family of linear maps $\{V(\phi) : V(t(\phi)) \to V(h(\phi));\ \phi \in Q_1\}$. In the following we consider $V(i)$ to be $\mathbb{C}$ vector spaces. The n-tuple of integers $\dim(V) = (\dim(V(i))_i \in \mathbb{N}^n$ is called the dimension vector of the representation V. A morphism between two representations $f : V \to W$ is a family of linear morphisms $\{f(i) : V(i) \to W(i); i \in Q_0\}$ such that for all arrows $\phi \in Q_1$ we have that $W(\phi) \circ f(t(\phi)) = f(h(\phi)) \circ V(\phi)$. For a fixed dimension vector $\alpha = (\alpha(1), ..., \alpha(n)) \in \mathbb{N}^n$ we define the representation space $R(Q, \alpha)$ of the quiver Q to be the set of all representations V of Q such that $V(i) \cong \mathbb{C}^{\alpha(i)}$ for all $i \in Q_0$. Because $V \in R(Q, \alpha)$ is completely determined by the linear morphisms $V(\phi)$, we have that

$$R(Q, \alpha) = \bigoplus_{\phi \in Q_1} \mathrm{Hom}_{\mathbb{C}}(\mathbb{C}^{\alpha(t(\phi)}, \mathbb{C}^{\alpha(h(\phi)}) = \bigoplus_{\phi \in Q_1} M_\phi(\mathbb{C}), \tag{1}$$

where for each $\phi \in Q_1$ we denote by $M_\phi(\mathbb{C})$ the vectorspace of all $\alpha(h(\phi))$ by $\alpha(t(\phi))$ matrices with entries in $\mathbb{C}$. The vectorspace $R(Q, \alpha)$ has dimension $\sum_\phi \alpha(h(\phi))\alpha(t(\phi))$. It is an affine space (and thus an affine variety) endowed with the regular action of the linear reductive group

$$\mathrm{GL}(\alpha) = \Pi_{i=1}^n \mathrm{GL}_{\alpha(i)}(\mathbb{C}) \tag{2}$$

determined for all representations $V \in R(Q, \alpha)$ and all group elements $g = (g(1), ..., g(n)) \in \mathrm{GL}(\alpha)$ by the rule

$$(g \cdot V)(\phi) = g(h(\phi))V(\phi)g(t(\phi))^{-1}. \tag{3}$$

Thus we have by $\mathrm{GL}(\alpha)$ a linear algebraic group acting on a finite-dimensional vector space. By the general theory of algebraic groups, we get the following properties of the respective G_α-orbits:

- Each orbit is a nonsingular variety ([OV90], Theorem 3.7);
- The stabilizer $\mathrm{GL}(\alpha)_V$ with respect to an element V in $R(Q,\alpha)$ is an algebraic subgroup of $\mathrm{GL}(\alpha)$ ([OV90], Theorem 3.7);
- For every $V \in R(Q,\alpha)$ the stabilizer $\mathrm{GL}(\alpha)_V$ is connected in Zariski topology; for $k = \mathbb{C}$, it is also connected in analytic topology ([Kir16]);
- $\mathrm{GL}(\alpha)$ possesses at least one closed orbit on $R(Q,\alpha)$ ([Kir16]);
- There exists at most one orbit of dimension equal to $\dim(R(Q,\alpha))$ and if it exists, it is open and dense in $R(Q,\alpha)$ (in Zariski topology and for $k = \mathbb{C}$ also in the analytic topology) [Kir16];

By definition, the G_α-orbits in $R(Q,\alpha)$ correspond bijectively to the isomorphism classes of k-representations of Q of dimension vector α. Closed orbits correspond to semisimple representations and thus the algebraic quotient of the representation space by the given group action is an affine variety parametrizing semisimple representations. By general arguments, one obtains a stratification of the quotient variety into finitely many irreducible locally closed subvarieties belonging to conjugacy classes of stabilizers of points with closed orbit.

1.2 The Jordan Decomposition of V

Every representation $V \in R(Q,\alpha)$ can be written (not necessarily uniquely) as $V = V_s + V_n$, where V_s is a semisimple representation and V_n is such that the zero representation lies in the closure of the orbit of V_n under the stabilizer subgroup of V_s. An element $v \in V$ is called semisimple (resp. nilpotent) with respect to G if an only if the orbit $G \cdot v$ is closed (resp. $0 \in \overline{G \cdot v}$). Herein, $\overline{G \cdot v}$ is the Zariski closure of the orbit in V.

We then call $V = V_s + V_n$ a Jordan decomposition of the representation V, if $v_s \in V_s$ is a semisimple element with respect to G and v_n is a nilpotent element with respect to the stabilizer subgroup of v_s, G_{v_s}, which is again a linear reductive group by a result of Matsushima. V. Kac has shown that every element $v \in V$ admits a Jordan decomposition. Therefore the classification of the orbit structure of $GL(\alpha)$ on $R(Q,\alpha)$ can be divided up in two subproblems:

1. the study of all semisimple representations of Q and
2. the study of nilpotent representations of Q with respect to certain linear reductive subgroups of $GL(\alpha)$.

2 Pruning and Moduli Spaces

2.1 Moduli Space and Training

In [Arm+21] the authors define a neural network (W, f) as follows:

Definition 2.1. Neural Network, ([Arm+21], Definition 7.1) Let Q be a connected finite acyclic quiver without multiple arrows. A neural network over Q is a pair (W, f) where W is a *thin representation* ([AJ20], Definition 3.6) of Q and $f = (f_q)_{Q \in \tilde{Q}}$ are activation functions, i.e. almost everywhere differentiable functions $f_q : \mathbb{C} \to \mathbb{C}$.

At a vertex v in the delooped quiver $\tilde{Q}_0$ ([AJ20], Definition 4.12) the activation output of (W, f) is defined with respect to the input x as ([Arm+21]):

$$\mathbf{a}(W,f)_v = \begin{cases} x_v & \text{if } v \text{ is an input vertex} \\ 1 & \text{if } v \text{ is a bias vertex} \\ f_v\left(\sum_{\alpha \in \chi_v} W_\alpha \mathbf{a}(W,f)_{s(\alpha)}(x)\right) & \text{in any other case} \end{cases} \tag{4}$$

where χ_v is the set of arrows of Q with target v. Similarly, a pre-activation at a vertex v is defined via ([Arm+21]):

$$\mathbf{pre}-\mathbf{a}(W,f)_v(x) = \begin{cases} 1 & \text{if } v \text{ is a source vertes} \\ \sum_{\alpha \in \chi_v} W_\alpha \mathbf{a}(W,f)_{s(\alpha)}(x) & \text{in any other case} \end{cases} \tag{5}$$

Using the definition of a neural network (W, f) and its activations, [Arm+21] define a network function $\Psi(W, f) : \mathbb{C}^d \to \mathbb{C}^q$ given by

$$\Psi(W,f)(x) = (\mathbf{a}(W,f)_{t1}(x), ..., \mathbf{a}(W,f)_{tk}(x)) \in \mathbb{C}^q. \tag{6}$$

They also define a knowledge map of the neural network (W, f) as

$$\begin{aligned} \phi(W,f) : \mathbb{C}^d &\to \text{thin}(Q) \\ x &\to W_x^f \end{aligned} \tag{7}$$

where

$$(W_x^f)_\alpha = \begin{cases} W_\alpha x_{s(\alpha)} & \text{if } s(\alpha) \text{ is an input vertex} \\ W_\alpha & \text{if } s(\alpha) \text{ is a bias vertex} \\ W_\alpha \frac{\mathbf{a}(W,f)_{s(\alpha)}(x)}{\mathbf{pre}-\mathbf{a}(W,f)_{s(\alpha)}(x)} & \text{if } s(\alpha) \text{ is a hidden vertex} \end{cases} \tag{8}$$

Now let (W, f) be a neural network and $x \in \mathbb{C}^d$ an input vector. Then the following diagram is commutative:

$$\begin{array}{ccc} \mathbb{C}^d & \xrightarrow{\Psi(W,f)} & \mathbb{C}^q \\ & {\scriptstyle \phi(W,f)} \searrow \quad \nearrow {\scriptstyle \hat{\Psi}} & \\ & \mathcal{M} & \end{array}$$

In the above commutative diagram, $\hat{\Psi}$ denotes the following map:

$$\begin{aligned} \hat{\Psi} : \mathcal{M}(Q) &\to \mathbb{C}^q \\ [V] &\mapsto \Psi(V,1)(1, ..., 1) \end{aligned} \tag{9}$$

where $(V, 1)$ denotes a neural network whose activation functions are all equal to the identity map. Thus the network function factors through the moduli space $\mathcal{M}$. On the other hand, [Arm+21] define the back-propagation algorithm as a map

$$\begin{aligned} b(W, f) : \mathbb{C}^p &\to R(Q, \alpha) \\ x &\mapsto b(W, f)(x) := dW \end{aligned} \tag{10}$$

where dW is the gradient and defines a map:

$$d(W, f) : R(Q, \alpha) \to R(Q, \alpha). \tag{11}$$

The authors of [Arm+21] prove that the back-propagation algorithm also factors through the moduli space $\mathcal{M}$ of a given representation $R(Q, \alpha)$ ([Arm+21], Theroem A.4):

$$\begin{array}{ccc} \mathbb{C}^d & \xrightarrow{b(W,f)} & R(Q,\alpha) \\ & {\scriptstyle \phi(W,f)} \searrow \quad \nearrow {\scriptstyle d(W,f)} & \\ & \mathcal{M} & \end{array}$$

Thus the maps of one forward as well as one backward pass factorize through the moduli space of a given representation $R(Q, \alpha)$.

Thus [AJ20; Arm+21] propose to investigate the geometry of the moduli space of a given network quiver in order to understand the dynamics of training this network ([AJ20], Consequence 3). In ([Arm+21], Theorem A.4) the authors prove that one back-propagation step factors through the same moduli space as the preceeding training step. By Inquiry 2 in [AJ20], moduli spaces can be used to formulate what training does to the data quiver representations. The authors of [AJ20] aim to "translate more deep learning objects into the quiver representation language". One example they mention is *dropout*. They conclude that "Dropout is a restriction of the training to several network subquivers. This translates into adjustments of the configurations of the data inside the moduli space via sub-spaces given by sub-quivers." In the following, dropout as well as pruning will be investigated by the quiver representation language of [AJ20].

2.2 Overparametrization, Dropout and Pruning

In practice, deep neural networks are initialized in an overparametrized fashion, meaning that they exhibit a large number of neurons and weights. The authors of [BS21] proved that this is a necessity in order to find a *smooth* function approximation for a broad number of data distributions and model classes. However, during iterative training, this overparametrization can cause numerical and computational problems. To the first class belongs the Exploding and Vanishing Gradient problem (EVGP), which very often occurs due to poor random

weight initialization. Let W denote the weights of a given network architecture, indexed by $W_{ij},\ i \in L, j \in N$, where $L \in \mathbb{N}$ denotes cardinality of layers and $N \in \mathbb{N}$ denotes the cardinality of nodes. We assume that $W_{ij} \in \mathbb{R}$.

2.2.1 Stochastic Gradient Descent

Let $\mathcal{L}$ be the loss function that defines the objective function for the neural network parameter optimization. It is calculated via a sum over all $n \in \mathbb{N}$ input training samples:

$$\mathcal{L}(W) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i(W), \tag{12}$$

where W are the weights of the neural network.. For stochastic gradient descent (SGD), the true gradient $\mathcal{L}(W)$ is approximated by a gradient at one single sample i:

$$W \mapsto W - \lambda \frac{\partial \mathcal{L}_i}{\partial W} \tag{13}$$

In (13), $\lambda \in \mathbb{R}$ is a parameter called the *learning rate* used within the optimization algorithm that describes how "fast" the algorithm converges towards a local minimum.

2.2.2 EVGP

Let $\epsilon \in \mathbb{R},\ \epsilon > 0$. Let $B \in \mathbb{R}$ be an upper bound determined by computational resources. Then the EVGP appears, when the absolute value of the gradient is very large for some trainable parameters W and very small for others:

$$\left| \frac{\partial \mathcal{L}}{\partial W} \right| < 0 + \epsilon \quad \text{or} \quad > B + \epsilon.$$

As a result of the EVGP, the incremental weight update in Eq. 13 is either too small to be meaningful or too large to be precise. The deeper and higher connected the network is, the more probable and faster the EVGP might arise. On the other hand, weights and also neuron activations might become very small after some training steps and contribute barely to the evaluated loss. Including them further in the training costs computational capacity.

2.2.3 Pruning

One way to get rid of weights and activated neurons causing the EVGP is *pruning*. Network pruning is the method of removing "unnecessary" neurons or weights. Herein, whether a neuron or weight is unnecessary is evaluated with respect to a predefined numerical value ϵ_{node} either of a neuron's activation or a weight magnitude, ϵ_{weight}. The latter is very simple and thus common. One removes weights that are already close to zero, i.e. $W_{ij} < \epsilon_{\text{weight}}$. On the other hand, one can observe during training that some neurons always output near-zero-values, i.e. $f(W_{ij} + b) < \epsilon_{\text{node}}$, for a nonlinear activation function f, a

bias term $b \in \mathbb{R}$ and the weight W_{ij} at one specific node. These neurons are removed by neuron pruning.

2.2.4 Dropout

Dropout is another regularization method to prevent overfitting. Let $\mathcal{N} \in [0, 100]$. This corresponds to a pre-fixed percentage of nodes per layer that are randomly deactivated ("dropped out") during one instance of training $I \in \mathcal{I}$, where $\mathcal{I}$ is the total number of iterations for which the optimization algorithm is applied.

2.2.5 Comparison of Pruning and Dropout

Pruning is an algorithmic way of removing weights and activated neurons that fall below a predefined criterion. Dropout, on the other hand, randomly removes a pre-fixed number of neurons per layer regardless of their output.

2.3 Pruning and Dropout with respect to Moduli Spaces

There are two interpretations regarding dropout and pruning in the context of moduli spaces. The first considers dropout and pruning as processes that alter the network quiver itself and thus its representation and moduli space. The second regards the network quiver as fixed and thus dropout and pruning only change its representation.

2.3.1 Interpretation 1: Change of the Network Quiver

The representation space of a network quiver is defined via the direct sum of the hom-spaces over an edge of the quiver, (1).

Neuron Activation Pruning and Dropout Removing a neuron from a deep neural network corresponds to removing a vertex in the respective network quiver. This vertex might be the source and target of some edges. As a consequence, these get also removed. This describes Neuron activation pruning and dropout within the perspective of interpretation 1.

Weight Pruning Removing a weight via pruning corresponds to removing an edge in the network quiver.

Reduced Network Quiver Hence, both weight and neuron pruning as well as dropout lead to the definition of a new, reduced network quiver Q_{red}.

2.3.2 Interpretation 2: Change of the Network Quiver Representation

Alternatively, one could say that pruning and dropout let the corresponding network quiver unchanged. Let $\mathcal{V}$ be a node deactivated via neuron activation pruning or dropout. Correspondingly, let $\mathcal{E}$ be an edge corresponding to a weight removed by weight pruning.

Neuron Pruning and Dropout When a node N gets deactivated, it produces no output any more. Thus, within the neural quiver representation, (1), all $\alpha(h(\mathcal{E}_N))$ by $\alpha(t(\mathcal{E}_N))$ matrices represent the zero map.

Weight Pruning For weight pruning, the edge $\mathcal{E}$ itself is considered to transport zero weights. Thus again $M_{\mathcal{E}}(\mathbb{C})$ in (1) has only zero entries.

Consequences for the Representation Space For neuron pruning and dropout as well as for weight pruning the corresponding Hom-spaces for $\phi = \mathcal{E}$ or $\phi = \mathcal{E}_N$ are represented by vector spaces of $\alpha(h(\mathcal{E}))$ by $\alpha(t(\mathcal{E}))$ or $\alpha(h(\mathcal{E}_N))$ by $\alpha(t(\mathcal{E}_N))$ zero matrices, $M_{\mathcal{E}_N}$ or M_N. Our direct sum representation of $R(Q, \alpha)$ thus splits into a pruned and a non-pruned part:

$$R(Q,\alpha) = \bigoplus_{\phi \in Q_{1,np}} M_\phi(\mathbb{C}) \oplus \bigoplus_{\phi \in Q_{1,p}} M_\phi(\mathbb{C}), \tag{14}$$

where $Q_{1,np}$ and $Q_{1,p}$ correspond to the non-pruned and pruned edges, respectively. Let M_{np} the non-pruned and M_p the pruned vector space for short. We thus get a split ([Ros02], Lemma 2.21) ses of vector spaces:

$$0 \longrightarrow M_p \longrightarrow R(Q,\alpha) \longrightarrow M_{np} \longrightarrow 0.$$

As the zero vector space corresponds to the neural element with respect to the direct sum, the representation space $R(Q, \alpha)$ factors through the non-pruned subspace M_{np}. By construction, the latter is isomorphic to the representation space of the reduced quiver $R(Q_{\text{red}}, \alpha_{\text{red}})$.

Theorem 2.1. The representation space of a neural network quiver Q at an instance of training $\mathcal{I}$ involving pruning or dropout is isomoprhic to the representation space of a reduced quiver Q_{red} constructed from Q in the way described in section 2.3.1.

References

[AJ20] Marco Antonio Armenta and Pierre-Marc Jodoin. *The Representation Theory of Neural Networks*. 2020. DOI: 10.48550/ARXIV.2007.12213. URL: https://arxiv.org/abs/2007.12213.

[Arm+21] Marco Armenta et al. *Double framed moduli spaces of quiver representations*. 2021. DOI: 10.48550/ARXIV.2109.14589. URL: https://arxiv.org/abs/2109.14589.

[BS21] Sébastien Bubeck and Mark Sellke. *A Universal Law of Robustness via Isoperimetry*. 2021. DOI: 10.48550/ARXIV.2105.12806. URL: https://arxiv.org/abs/2105.12806.

[Kir16] Alexander Kirillov. *Quiver Representations and Quiver Varieties*. Providence, Rhode Island: American Mathematical Society, 2016.

[OV90] Arkadij L. Onishchik and Ernest B. Vinberg. *Lie Groups and Algebraic Groups*. Berlin Heidelberg: Springer-Verlag, 1990.

[Ros02] Harvey E. Rose. *Linear Algebra. A Pure Mathematical Approach*. Basel: Birkhäuser Verlag, 2002.