Notes on the Symmetries of 2-Layer ReLU-Networks

Henning Petzka*[1], Martin Trimmel[1], and Cristian Sminchisescu[1,2]

[1]Lund University
[2]Google Research

Abstract

Symmetries in neural networks allow different weight configurations that lead to the same network function. For odd activation functions, the set of transformations mapping between such configurations has been studied extensively, but less is known for neural networks with ReLU activation functions. We give a complete characterization for fully-connected networks with two layers. Apart from two well-known transformations, only degenerated situations allow additional transformations that leave the network function unchanged. Finally, we present a non-degenerate situation for deep neural networks in which transformations exist that leave the network function intact.

1 Introduction

Let $f_{\mathbf{W}}(x) = \mathbf{w}^L((\dots(\mathbf{w}^1 x + b_1)_+ + b_2)_+ + \dots)_+ + b_L$ denote a neural network with ReLU activation function $(x)_+ = \max\{x, 0\}$. We consider the map $\phi : \mathbf{W} \to \mathcal{F}$ from the set of weights to a set of realizable network functions. In other words, for a given collection of weights $\mathbf{W}$, we denote by $\phi(\mathbf{W}) = f_{\mathbf{W}}$ the neural network function defined by weights $\mathbf{W}$. Due to symmetries in the network, this function is not injective. Following terminology of [3], we call transformations on the weight space that leave the network function identical **equioutput transformations.**[1]

Definition 1. *Let $\mathbf{W}$ denote a weight selection of a neural network function f. A transformation α on the weight space is called an equioutput transformation, if $f_{\mathbf{W}}(\mathbf{x}) = f_{\alpha(\mathbf{W})}(\mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{W}$.*

For ReLU networks, there are well-known equioutput transformations as discussed, for example, in [4] and [6].
(1) π: A permutation π of neurons (together with all its in- and outgoing weights) within a layer.
(2) ρ_λ: A transformation ρ_λ given by multiplying all ingoing weights of a neuron by $\lambda > 0$ and multiplying all outgoing weights by $\frac{1}{\lambda}$. This leads to the same network function because of the positive homogeneity of the ReLU function: If $\lambda > 0$, then $\lambda(x)_+ = (\lambda x)_+$, hence for any scalars a, b we have that $a(bx)_+ = (a\lambda)(\frac{b}{\lambda}x)_+$.

Knowledge of these transformations is important for the understanding of the loss function. For example, it follows that a global minimum never comes alone: For a given global minimum $\mathbf{W}$, any equioutput transformation of $\mathbf{W}$ leads to a different global minimum. Further, knowledge of these equioutput transformations can be useful for the study of properties of neural networks. Assume a certain property should depend only on the function, but not on its specific representation. If we aim to describe this property in terms of its representation in form of weights, then this property must be independent under all equioutput transformations, otherwise the property is somewhat ill-defined. Such a dependence was observed by Dinh et al. [4] for the property of generalization and Hessian-based measures of flatness. While generalization performance only depends on the network function, the common measures depend on the representation given by specific weight values. Similarly, Neyshabur et al. [6] argue that for ReLU networks it is not sensible to use the usual gradient descent, because the steepest direction is then defined by a maximal reduction in loss for equal (infinites-

*Corresponding Author: henning.petzka@math.lth.se
This work was supported by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI (PN-III-P4-ID-PCE- 2016-0535, PN-III-P4-ID-PCCF-2016-0180)

[1]In contrast to [3], our transformations are not necessarily analytic.

https://doi.org/10.7557/18.5150

imal) step length in the l2-norm. But this measure of steepness depends on the specific parameterization. Hence, in this example, the property of interest is the update rule during optimization (which should be independent of the representation) but the direction of gradient descent is dependent on neuron-wise reparameterizations ρ_λ.

A natural question that arises is the following: Are there any other equioutput transformations than the ones described above? This has been studied quite thoroughly for odd activation functions, which we recap below. In short, there are many situation-dependent equioutput transformations that exist only for specific parameter values, but there are only the two types of equioutput transformations that are analytical and equally work no matter the given parameter values. To the knowledge of the authors, there has been little work on the case of ReLU activation functions. Only recently, by considering hyperplane configurations, [8] shows that under certain conditions it is often possible to compute the network parameters from the network function up to a composition of π and ρ_λ. [2] study whether the proximity of network functions implies proximity of their parameterizations. Finally, [7] consider networks of decreasing width.

Our work gives a complete characterization for fully-connected 2-layer regression networks with ReLU activation functions. For these networks, we outline all situation-dependent equioutput transformations, and we show how networks can be reduced to an irreducible form. Up to degenerate cases, irreducible networks with identical network function have weights that differ only by a composition of transformations π and ρ_λ. For deep neural networks, we describe additional situation-dependent equioutput transformations that are not degenerate, though unlikely to appear with proper initialization.

2 Odd activation functions

For neural networks with all activation functions given by the tangens hyperbolicus $tanh()$, equioutput transformations have been studied in several works. In this case, the well-known equioutput transformations are given by permutations π as above and by sign-flips. The latter is the negative-scalar variant ρ_{-1} and uses $tanh(-x) = -tanh(x)$. Sussmann [9] considers 2-layer networks. For certain weight-configurations, he identifies natural reduction steps that allow removing nodes while keeping the network function constant. Two irreducible networks then determine the same network function if and only if one weight configuration can be transformed to the other one by a composition of permutations π or sign-flips ρ_{-1}. Chen et al. [3] extended the result of Sussman to deeper networks to show the following theorem.

Theorem [3] For parameter values $\mathbf{W}$ and input $\mathbf{x}$, let $f_{\mathbf{W}}(\mathbf{x})$ denote a neural network function with all activation functions $tanh()$. All analytical (i.e. expandable in a power series around points) equioutput transformations on the weight space $\mathbf{W}$ are compositions of interchanges of neurons π within a layer and sign flip transformations ρ_{-1}.

The restriction to analytical transformations aims at the exclusion of equioutput transformations that only exist for specific parameters $\mathbf{W}$. We will call transformations that rely on certain weight values **situation-dependent**. In Sussmann [9], these situation-dependent transformations were excluded by the reduction steps to irreducible form. Since the transformations π and ρ_{-1} always exist, they have relevant consequences in practical applications. Situation-dependent transformations that exist only in degenerate cases are less likely to play a significant role in practice. However, effects are still possible when the degenerate situations only hold approximately.

Considering activation functions other than $tanh()$, Albertini et al. [1] extended the analysis to infinitely differentiable functions satisfying $\sigma(0) = 0, \sigma'(0) = 0, \sigma''(0) = 0$. Kurková and Kainen [5] generalize the result for two layer regression networks where the activation functions need not to be continuous, but bounded and asymptotically constant (this includes sigmoids, for example).

3 Two-layer ReLU regression networks

This section investigates fully connected ReLU neural networks for regression with one linear output neuron and one hidden layer,

$$f(\mathbf{x}) = \mathbf{v} \cdot (\mathbf{w} \cdot \mathbf{x} + \mathbf{b})_+ + c = \sum_j v_j (\mathsf{n}_j(\mathbf{x}))_+ + c$$

with $\mathbf{w} \in \mathbb{R}^{m\times d}$, $\mathbf{v} \in \mathbb{R}^{1\times m}, \mathbf{b} \in \mathbb{R}^m, c \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^d$ and neuron pre-activation functions $\mathsf{n}_j(\mathbf{x}) = \mathbf{w}_j \cdot \mathbf{x} + b_j$. We call a neural network **reducible**, if nodes can be removed without changing the network function, possibly by a redefinition of some weights. Two types of reducibility are as follows:

(R1) If all ingoing or outgoing weights of a neuron are zero, then this neuron is never active or never processed and can be removed without changing the network function. If all ingoing weights except for the bias are zero, then the neuron can be removed and its contribution to the next layer can be added to the biases of the following layer.

(R2) If $\mathsf{n}_1(\mathbf{x}) = \lambda \mathsf{n}_2(\mathbf{x})$ for some $\lambda > 0$ (two neurons in the hidden layer are equal up to a positive multiplicative factor), then one of the nodes can be removed using a linear combination of the corresponding outgoing weights, e.g. $v_{1,new} = v_1 + \lambda v_2$ for $\mathsf{n}_1(\mathbf{x})$ when removing the second one.

Many equioutput transformations exist for non-reduced neural networks. If all weights into a neuron are zero, then the outgoing weights can be changed arbitrarily. If $\lambda \cdot \mathsf{n}_1(\mathbf{x}) = \mathsf{n}_2(\mathbf{x})$ for some $\lambda > 0$, then weight be can be shifted from the outgoing weights of one neuron to the other neuron (i.e. $v_1(\mathsf{n}_1(\mathbf{x}))_+ + v_2(\mathsf{n}_2(\mathbf{x}))_+ = (v_1 - z)(\mathsf{n}_1(\mathbf{x}))_+ + (v_2 + \frac{z}{\lambda})(\mathsf{n}_2(\mathbf{x}))_+$ for all z). We investigate in this section whether there exist other symmetries than the one just named after reducing two-layer fully connected ReLU regression networks to an irreducible form. Our working hypothesis, which later needs to be adjusted slightly, is that ReLU networks show the same behavior as neural networks with odd activation functions.

Working hypothesis: Let $f_{\mathbf{W}_1}$ and $f_{\mathbf{W}_2}$ denote a pair of (R1,R2)-irreducible two-layer fully connected ReLU regression networks. If $f_{\mathbf{W}_1}(\mathbf{x}) = f_{\mathbf{W}_2}(\mathbf{x})$ for all $\mathbf{x}$, then $\mathbf{W}_1$ can be attained from $\mathbf{W}_2$ by a composition of an arbitrary number of symmetries π and ρ_λ as above.

We will take advantage of two short lemmas and the definition an of activation pattern.

Definition 2. *Let g denote a non-zero linear function $\mathbb{R}^d \to \mathbb{R}$. (i) The activation pattern of g is defined by the set of $\mathbf{x} \in \mathbb{R}^d$ such that $g(\mathbf{x}) > 0$, which we denote by $\mathbf{R}^+$, i.e. $\mathbf{R}^+(g) = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) > 0\}$. (ii) The zero value-hyperplane of g is denoted by $\mathbf{H}(g) = \{\mathbf{x} \mid g(\mathbf{x}) = 0\}$. (iii) The hyperplane $\mathbf{H}(g)$ divides $\mathbb{R}^d$ into the two regions $\mathbf{R}^+(g)$ and $\mathbf{R}^-(g) := \mathbb{R}^d \setminus (\mathbf{R}^+(g) \cup \mathbf{H}(g))$.*

Lemma 1. *Let $n_1(\mathbf{x}), n_2(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ be two linear functions with $n_1(\mathbf{x}) = \mathbf{w}_1 \cdot \mathbf{x} + b_1$, $n_2(\mathbf{x}) = \mathbf{w}_2 \cdot \mathbf{x} + b_2$. Then the following properties are equivalent.*

(1) $\mathbf{w}_1 = \lambda \mathbf{w}_2$ and $b_1 = \lambda b_2$ for some $\lambda > 0$.

(2) $n_1(\mathbf{x}) = \lambda n_2(\mathbf{x})$ for some $\lambda > 0$.

(3) $(n_1(\mathbf{x}))_+ = \lambda (n_2(\mathbf{x}))_+$ for some $\lambda > 0$.

(4) $\mathbf{R}^+(n_1) = \mathbf{R}^+(n_2)$.

We show a strong form of linear independence for neuron functions with activation $(\mathsf{n}_i(\mathbf{x}))_+$ assuming a little more than pairwise different activation patterns. This strong form of linear independence was named in Albertini et al. [1] as the independence property (IP): The set of functions $\{(\mathsf{n}_i(\mathbf{x}))_+ \mid i = 1, \dots, m\}$ has (IP) if pairwise inequality of the functions $(\mathsf{n}_i(\mathbf{x}))_+ \neq (\mathsf{n}_j(\mathbf{x}))_+$ implies linear independence of $\{1, (\mathsf{n}_i(\mathbf{x}))_+ \mid i = 1, \dots, m\}$. This property is required to find a pair of neurons with the same zero hyperplane whenever neurons are linearly dependent after their activation. The proof of Theorem 1 will take advantage of this property by matching neurons in different representations of the same network function.

After exclusion of a degenerate case, we obtain property (IP) for $\{(\mathsf{n}_i(\mathbf{x}))_+\}$ even when the constant function 1 is replaced by an arbitrary non-zero linear function. The degenerate case we exclude consists of two neurons $\mathsf{n}_k, \mathsf{n}_l$ with identical zero hyperplanes, i.e., $\mathbf{H}_k = \mathbf{H}_l$ with $\mathbf{H}_k = \mathbf{H}(n_k)$.

Lemma 2. *Let $\ell(\mathbf{x})$ be a non-zero linear function. Suppose that $n_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x} + b_i$, $1 \le i \le m$ are non-constant linear functions with pairwise different hyperplanes $\mathbf{H}_i$. Then the set of functions $\{\ell(\mathbf{x}),\ (n_i(\mathbf{x}))_+ \mid 1 \le i \le m\}$ is linearly independent.*

We now give an example that the assumption of pairwise different hyperplanes $\mathbf{H}_j$ was necessary and it would not have been sufficient to assume pairwise different activation patterns. If $\mathbf{H}_k = \mathbf{H}_l$ for some $k \neq l$, then we still have a pairwise different activation pattern when $\mathbf{R}_\mathbf{k}^+ = \mathbf{R}_\mathbf{l}^-$. In words, neuron k is active exactly when neuron l is inactive and the other way around (except for the points on the hyperplane $\mathbf{H}_k = \mathbf{H}_l$ where none of the neurons is active). We say that $\mathsf{n}_k, \mathsf{n}_l$ are a **pair of neurons with opposite activation pattern**.

Example 1. *In the network in Figure 1 all neurons have nontrivial and pairwise different activation patterns, yet the network implements the constant zero function.*

This example shows two things: (i) In such a situation where $\mathbf{H}_k = \mathbf{H}_l$ for some $k \neq l$, we see that new situation-dependent equioutput transformations can appear. (ii) Single layers of neural networks with ReLU activation can implement linear functions.

If $\mathbf{H}_k = \mathbf{H}_l$ for two neurons k, l with opposite activation pattern, then we can transform them into a linear neuron (no activation function) and a neuron with ReLU activation function as follows: With outgoing weights v_1 and v_2, we have $v_1(\mathsf{n}_k(\mathbf{x}))_+ + v_2(-\lambda \mathsf{n}_k(\mathbf{x}))_+ = v_1 \mathsf{n}_k(\mathbf{x}) + (\lambda v_2 + v_1)(-\mathsf{n}_k(\mathbf{x}))_+$. But we could have equally well decomposed the sum into a linear function plus rest to get $\lambda v_2 \mathsf{n}_k(\mathbf{x}) + (v_1 + \lambda v_2)(\mathsf{n}_k(\mathbf{x}))_+$. More generally, we can flip the activation region of a neuron $\mathsf{n}_i(\mathbf{x})$ by introducing a linear neuron: $(\mathsf{n}_i(\mathbf{x}))_+ = \mathsf{n}_i(\mathbf{x}) + (-\mathsf{n}_i(\mathbf{x}))_+$. Accordingly, irreducible ReLU networks will only be defined up to the (optional) existence of a single non-constant linear neuron $\mathsf{N}(\mathbf{x})$ (i.e., the linear function may only consist of the bias of the output layer).

(R3) Write each pair of neurons with opposite activation pattern as above as sum of a linear neuron $\{\mathsf{N}_i\}$ and a neuron with ReLU activation function. Add all linear neurons plus output bias to a single linear neuron $\mathsf{N}(\mathbf{x}) = \alpha\mathbf{x} + c$, which is subsequently split into two ReLU neurons with opposite activation pattern $(\alpha\mathbf{x})_+ - (-\alpha\mathbf{x})_+$ and bias c.

In the following, we represent an (R1,R2,R3)-irreducible network as $f_\mathbf{W}(\mathbf{x}) = \sum_j v_j(\mathsf{n}_j(\mathbf{x}))_+ + \mathsf{N}(\mathbf{x})$ with $\mathsf{N}(\mathbf{x})$ a linear function and all $\mathsf{n}_j(\mathbf{x})$ having pairwise different zero hyperplanes $\mathbf{H}_j$. We also find an additional equioutput transformation ψ_- which operates on a single neuron and the linear function: ψ_- multiplies all incoming weights of a neuron $\mathsf{n}_i(\mathbf{x})$ by -1, leaves the outgoing weights intact, and adds $v_i \mathsf{n}_i(\mathbf{x})$ to $\mathsf{N}(\mathbf{x})$. More generally, for each subset of indices $J \subseteq \{1, \ldots, m\}$, we define ψ_-^J as the application of ψ_- to all neurons $\mathsf{n}_j(\mathbf{x})$ with $j \in J$. Note that ψ_-^J may add two neurons with opposite activation pattern (forming a linear neuron) to the hidden layer when it is applied to a network representation without opposite activation patterns ($\mathsf{N}(\mathbf{x}) = b_L$ a constant) and the resulting larger network can still be (R1,R2,R3)-irreducible. We discuss later how we can recover a representation with constant $\mathsf{N}(\mathbf{x})$ as bias whenever it exists.

The following theorem states that irreducible two-layer ReLU-networks are unique up to equioutput transformations π, ρ_λ and ψ_-^J. The proof can be found in the appendix.

Theorem 1. *Let $f_{\mathbf{W}_1}$ and $f_{\mathbf{W}_2}$ denote a pair of $(R1, R2, R3)$-irreducible two-layer fully connected ReLU regression networks. If $f_{\mathbf{W}_1}(\mathbf{x}) = f_{\mathbf{W}_2}(\mathbf{x})$ for all $\mathbf{x}$, then $\mathbf{W}_2$ can be obtained from $\mathbf{W}_1$ by a composition of an arbitrary number of equioutput transformations π, ρ_λ and ψ_-^J as above.*

For any subset J of indices, applying ψ_-^J twice yields the identity function. Further, ψ_-^J is the only equioutput transformation that can change the linear function $\mathsf{N}_1(\mathbf{x})$ in $f_{\mathbf{W}_1}(\mathbf{x}) = \sum_j v_1^j(\mathsf{n}_1^j(\mathbf{x}))_+ + \mathsf{N}_1(\mathbf{x})$. This implies that a (R1,R2,R3)-irreducible ReLU network function $f_{\mathbf{W}_1}(\mathbf{x})$ has a representation $f_{\mathbf{W}_2}(\mathbf{x}) = \sum_j v_2^j(\mathsf{n}_2^j(\mathbf{x}))_+ + \mathsf{N}_2(\mathbf{x})$ with a constant $\mathsf{N}_2(\mathbf{x}) = b_L$, if and only if there is a transformation ψ_-^J that cancels $\mathsf{N}_1(\mathbf{x})$ up to a constant. In other words, we can find a representation without opposite activation patterns if and only if there exists a subset J of indices such that $\mathsf{N}_1(\mathbf{x}) + \sum_{j \in J} v_1^j \mathsf{n}_1^j(\mathbf{x}) = c$ for a constant c. In this case, we apply ψ_-^J and change the bias at the output layer to c. (R1 subsequently removes the ReLU neurons that were used for the linear function $\mathsf{N}_1(\mathbf{x})$.) These observations prove the following result that, apart from degenerate situations, π and ρ_λ are the only equioutput transformations.[2]

[2]A degenerate pair of ψ_-^J-equivalent networks violating (ii) is given in the appendix.

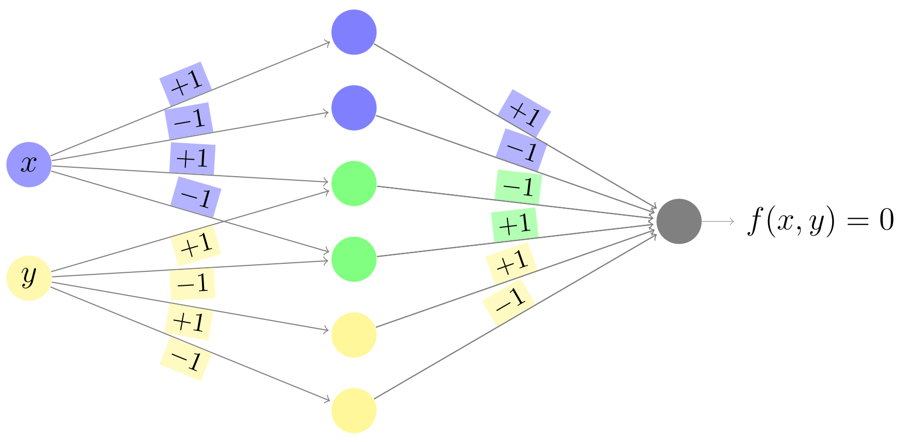


Figure 1: ReLU network with non-trivial activation pattern implementing network function $x_+ - (-x)_+ + y_+ - (-y)_+ - (x+y)_+ + (-x-y)_+ = x + y - (x+y) = 0$.

Theorem 2. *Let $f_{\mathbf{W}_1} = \sum_{j=1}^{m_1} v_1^j (\eta_1^j(\mathbf{x}))_+ + b_1$ and $f_{\mathbf{W}_2} = \sum_{j=1}^{m_2} v_2^j (\eta_2^j(\mathbf{x}))_+ + b_2$ denote a pair of $(R1, R2)$-irreducible two-layer fully connected ReLU regression networks such that for $i = 1, 2$*

(i) all $\eta_i^j(\mathbf{x})$ have pairwise different zero hyperplanes; and (ii) there exists no non-empty subset of indices $J_i \subseteq \{1, \ldots, m_i\}$ such that $\sum_{j \in J_i} v_i^j \eta_i^j(\mathbf{x}) = c$ is constant for all $\mathbf{x}$.

If $f_{\mathbf{W}_1}(\mathbf{x}) = f_{\mathbf{W}_2}(\mathbf{x})$ for all $\mathbf{x}$, then $\mathbf{W}_2$ can be obtained from $\mathbf{W}_1$ by a composition of equioutput transformations π and ρ_λ as above.

LeakyReLUs The same results analogously hold more generally for networks with LeakyReLU activation defined by $\sigma_\mu(x) = x$ for $x > 0$ and $\sigma_\mu(x) = \mu x$ for $x < 0$ ($\mu \neq 1$) by use of the identity $\sigma_\mu(\mathbf{x}) = (1+\mu)\mathbf{x} + \sigma_\mu(-\mathbf{x})$.

4 Deep neural networks

In the proof for the 2-layer case, we take advantage of the assumption that the input is all of $\mathbb{R}^d$. In deeper layers, we lose this property, because ReLU activations reduce the domain for future layers to the first quadrant where all coordinate values are positive. Neurons for which all ingoing weights are positive behave like linear neurons in deeper layers. This considerably complicates the treatment of reducibility and symmetries in the case of deep neural networks. There are now more than merely degenerate situations in which different parameter choices induce the same network function.

Example 2. *Suppose two neurons in a layer with index $l > 1$ (i.e., not the first hidden layer) have only positive weights. Since the output of the previous ReLU layer with index $l-1$ is always positive, this implies that these two neurons are always active and hence behave like linear neurons. Let $A \in \mathbb{R}^{n_{l-1} \times 2}$ denote the incoming weights into these two neurons and $C \in \mathbb{R}^{2 \times n_{l+1}}$ denote the outgoing weights. Then for any positive invertible matrix $B \in \mathbb{R}_+^{2\times 2}$, we can replace A by BA and C by CB^{-1}. Since B is positive, the neurons still have only positive incoming weights BA and still behave like linear neurons. Together, they implement the same function $CA\mathbf{x} = (CB^{-1})(BA)\mathbf{x}$.*

The question arises whether two neurons that are always active can be reduced to a single neuron that is always active, just as we reduced degenerate cases in the 2-layer case to obtain a minimal network. An example shows that this is not always possible.

Example 3. *We present an example, where two neurons that are always active cannot be combined to a single ReLU neuron: As before, let $A \in \mathbb{R}^{n_{l-1} \times 2}$ denote the incoming weights into two always-active neurons and $B \in \mathbb{R}^{2 \times n_{l+1}}$ denote their outgoing weights. For our example, it suffices to consider $n_{l-1} = 2$ and $n_{l+1} = 1$:*

$$A_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } B_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

To combine the two hidden neurons into one single neuron, we are looking for $D \in \mathbb{R}^{2\times 1}$ and $d \in \mathbb{R}$ such that $B(A\mathbf{x})_+ = BA\mathbf{x} = d(D\mathbf{x})_+$. The sign of d defines the sign of the contribution of $d(D\mathbf{x})_+$ to the output neuron independently of $\mathbf{x} \in \mathbb{R}^2$. But for $\mathbf{x} = (x_1, x_2)$ we have that the contribution of $BA\mathbf{x}$ is positive when $x_1 > x_2$ and negative when $x_1 < x_2$. Hence, no such D and d can exist.

In the 2-layer case, all situation-dependent equioutput transformations appeared at degenerate weight constellations. Hence, up to degenerate cases, the situation is quite simple. For deep neural networks, this situation changes drastically. Example 2 does not describe a degenerate condition. Within a neighborhood of such a weight configuration, weights can be changed arbitrarily and the resulting network will still possess two neurons that are always active and hence allow the described equioutput transformations. Empirically, however, the existence of neurons with only positive incoming weights is very unlikely to appear with proper initialization. We experimented with a simple MNIST network with He initialization. After training, we consistently observed for each neuron of the network that approximately half of the incoming weights are negative. This seems to empirically exclude the existence of neurons with only positive incoming weights. On the other hand, we note that there are other (more complex) possibilities to obtain neurons that are always active, in which case equioutput transformations as in Example 2 different to π, ρ_λ and ψ_-^J exist.

5 Conclusion

Different to the well-studied case with odd activation functions, the appearance of equioutput transformations of ReLU networks is more complicated. After removing neurons that are never active and merging of neurons with identical activation pattern, we have to additionally consider neurons with opposite activation patterns. These pairs of neurons with opposite activation pattern can implement linear functions and cannot be completely removed. This allows equioutput transformations different to the well-known ones. Constructing linear neurons in deep networks by using positive weights shows that equioutput transformations do not only exist in degenerate cases.

References

[1] Francesca Albertini, Eduardo D. Sontag, and Vincent Maillot. Uniqueness of weights for neural networks. In *in Artificial Neural Networks with Applications in Speech and Vision*, pages 115–125. Chapman and Hall, 1993.

[2] Julius Berner, Dennis Elbrächter, and Philipp Grohs. How degenerate is the parametrization of neural networks with the relu activation function? *Proceedings of the Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[3] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

[4] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

[5] Vera Kurková and Paul C Kainen. Functionally equivalent feedforward neural networks. *Neural Computation*, 6(3):543–558, 1994.

[6] Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2015.

[7] Mary Phuong and Christoph H. Lampert. Functional vs. parametric equivalence of relu networks. *International Conference on Learning Representations 2020*, 2020.

[8] David Rolnick and Konrad P. Kording. Identifying weights and architectures of unknown relu networks. *arxiv preprint, arXiv:1910.00744v1*, 2019.

[9] Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.

A Proof of Lemma 1

Proof. The implications (1) ⇒ (2) ⇒ (3) ⇒ (4) are trivial, so we only need to show that (4) implies (1).

If $\mathbf{R}^+(\mathsf{n}_1) = \mathbf{R}^+(\mathsf{n}_2)$, then in particular $\mathbf{H}(\mathsf{n}_1) = \mathbf{H}(\mathsf{n}_2)$, so there is $\mathbf{x}_0$ such that $\mathbf{w}_1\mathbf{x}_0 + b_1 = 0 = \mathbf{w}_2\mathbf{x}_0 + b_2$. Suppose $\mathbf{w}_1 \neq \lambda\mathbf{w}_2$ (the weight vectors are not parallel), then there exists some $\mathbf{z} \in \mathbb{R}^d$ such that $\mathbf{z}$ is orthogonal on $\mathbf{w}_1$ but not on $\mathbf{w}_2$, i.e., there is $\mathbf{z}$ such that $\mathbf{w}_1 \cdot \mathbf{z} = 0$ and $\mathbf{w}_2 \cdot \mathbf{z} \neq 0$. By choosing the sign of $\mathbf{z}$ appropriately, we get $\mathbf{w}_2 \cdot \mathbf{z} > 0$. Then

$$\mathsf{n}_1(\mathbf{x}_0 + \mathbf{z}) = \mathbf{w}_1 \cdot (\mathbf{x}_0 + \mathbf{z}) + b_0 = \mathbf{w}_1 \cdot \mathbf{x}_0 + b_0 + \mathbf{w}_1 \cdot \mathbf{z} = 0, \text{ and}$$

$$\mathsf{n}_2(\mathbf{x}_0 + \mathbf{z}) = \mathbf{w}_2 \cdot (\mathbf{x}_0 + \mathbf{z}) + b_2 = \mathbf{w}_2 \cdot \mathbf{x}_0 + b_2 + \mathbf{w}_2 \cdot \mathbf{z} = 0 + \mathbf{w}_2 \cdot \mathbf{z} > 0,$$

which stands in contradiction to $\mathbf{R}^+(\mathsf{n}_1) = \mathbf{R}^+(\mathsf{n}_2)$.

Therefore, we must have $\mathbf{w}_1 = \lambda\mathbf{w}_2$ for some λ. Assume $\lambda < 0$. We introduce $\mu \in \mathbb{R}$ such that $b_1 = \mu\lambda b_2$. (This exists without loss of generality, if necessary we can interchange the roles of b_1 and b_2.) If $\lambda < 0$, then

$$\mathbf{w}_1 \cdot \mathbf{x} + b_1 > 0 \Leftrightarrow \frac{1}{\lambda}\mathbf{w}_1 \cdot \mathbf{x} + \frac{1}{\lambda}b_1 < 0 \Leftrightarrow \mathbf{w}_2 \cdot \mathbf{x} + \mu b_2 < 0. \tag{1}$$

By assumption that $\mathbf{R}^+(\mathsf{n}_1) = \mathbf{R}^+(\mathsf{n}_2)$, we get that

$$\{\mathbf{x} \mid \mathbf{w}_2 \cdot \mathbf{x} < -\mu b_2\} \overset{(1)}{=} \mathbf{R}^+(\mathsf{n}_1) = \mathbf{R}^+(\mathsf{n}_2) \overset{\text{Def}}{=} \{\mathbf{x} \mid \mathbf{w}_2 \cdot \mathbf{x} > -b_2\}.$$

Since the first set contains $\gamma\mathbf{w}_2$ for all sufficiently small γ, but the the set of all γ such that $\gamma\mathbf{w}_2$ is contained in the last set is bounded below, this leads to a contradiction for any μ. Hence λ cannot be negative.

So we must have $\mathbf{w}_1 = \lambda\mathbf{w}_2$ for some $\lambda > 0$, and we again write $b_1 = \mu\lambda b_2$. Then

$$\mathbf{w}_1 \cdot \mathbf{x} + b_1 > 0 \Leftrightarrow \frac{1}{\lambda}\mathbf{w}_1 \cdot \mathbf{x} + \frac{1}{\lambda}b_1 > 0 \Leftrightarrow \mathbf{w}_2 \cdot \mathbf{x} + \mu b_2 > 0$$

Since $\mathbf{R}^+(\mathsf{n}_1) = \mathbf{R}^+(\mathsf{n}_2)$, $\{\mathbf{x} \mid \mathbf{w}_1 \cdot \mathbf{x} + b_1 > 0\} = \{\mathbf{x} \mid \mathbf{w}_2 \cdot \mathbf{x} + b_2 > 0\}$, we must have $\mu = 1$, which completes the proof. □

B Proof of Lemma 2

Proof. We consider the hyperplanes

$$\mathbf{H}_j = \{\mathbf{x} \mid \mathbf{w}_j \cdot \mathbf{x} + b_j = 0\},$$

with each dividing the space $\mathbb{R}^d$ into the two regions $\mathbf{R}_j^+ = \mathbf{R}^+(\mathsf{n}_j)$ and $\mathbf{R}_j^-$ where $\mathsf{n}_j(\mathbf{x}) > 0$ and $\mathsf{n}_j(\mathbf{x}) < 0$ respectively. By assumption, the regions $\mathbf{R}_j^+$ are pairwise different. Note that the union $\bigcup_j \mathbf{H}_j$ is a closed set of measure zero.

Suppose

$$a_0\ell(\mathbf{x}) + \sum_{j=1}^{m} a_j(\mathsf{n}_j(\mathbf{x}))_+ = 0 \tag{2}$$

for some scalar coefficients a_j and all $\mathbf{x}$. We need to show that $a_j = 0$ for all $0 \le j \le m$.

For any $\mathbf{z}$ we let $I_\mathbf{z} = \{j \in \{1, 2, \ldots, m\} \mid \mathbf{z} \in \mathbf{R}_j^+\}$ denote the set of indices of those $\mathbf{R}_j^+$ that contain $\mathbf{z}$. Then (2) reduces to

$$a_0\ell(\mathbf{z}) + \sum_{j \in I_\mathbf{z}} a_j(\mathbf{w}_j \cdot \mathbf{z} + b_j) = 0.$$

If $\mathbf{z}_0 \notin \bigcup_j \mathbf{H}_j$, then by standard continuity arguments there is $\epsilon > 0$ such that for any $||\mathbf{z}_1 - \mathbf{z}_0|| \le \epsilon$ we have $I_{\mathbf{z}_0} = I_{\mathbf{z}_1}$. Therefore,

$$a_0 \ell(\mathbf{z}_1) + \sum_{j \in I_{\mathbf{z}_0}} a_j(\mathbf{w}_j \cdot \mathbf{z}_1 + b_j) = 0 \text{ and hence } a_0(\ell(\mathbf{z}_0) - \ell(\mathbf{z}_1)) + \sum_{j \in I_{\mathbf{z}_0}} a_j \mathbf{w}_j \cdot (\mathbf{z}_0 - \mathbf{z}_1) = 0.$$

With $\ell(\mathbf{x}) = c + \mathbf{w}_0 \cdot \mathbf{x}$, this gives

$$\left(a_0 \mathbf{w}_0 + \sum_{j \in I_{\mathbf{z}_0}} a_j \mathbf{w}_j \right) \cdot (\mathbf{z}_0 - \mathbf{z}_1) = 0$$

Since $\mathbf{z}_1$ is arbitrary in the neighborhood of $\mathbf{z}_0$, this shows that

$$a_0 \mathbf{w}_0 + \sum_{j \in I_{\mathbf{z}_0}} a_j \mathbf{w}_j = 0 \;\text{ for all } \mathbf{z}_0 \notin \bigcup_j \mathbf{H}_j. \tag{3}$$

Consider a hyperplane $\mathbf{H}_k$ and choose a point $\mathbf{x}_k \in \mathbf{H}_k \setminus \bigcup_{i \neq k} \mathbf{H}_i$. The latter set is non-empty in the case of our assumption of pairwise distinct hyperplanes.

The $\mathbf{R}_i^+$ are also pairwise different by assumption, so there is a sufficiently small ball $B_\epsilon(\mathbf{x}_k)$ of radius $\epsilon > 0$ around $\mathbf{x}_k$ with $B_\epsilon(\mathbf{x}_k) \cap \mathbf{H}_l = \emptyset$ for all $l \neq k$. For any $\mathbf{y}_k \in B_\epsilon(\mathbf{x}_k) \cap \mathbf{R}_k^+$ and $\mathbf{z}_k \in B_\epsilon(\mathbf{x}_k) \cap \mathbf{R}_k^-$ we have $\mathbf{z}_k \notin \bigcup_j \mathbf{H}_j$ and

$$\begin{aligned} 0 = 0 - 0 &\overset{(2)}{=} \left(a_0 \ell(\mathbf{y}_k) + \sum_j a_j (n_j(\mathbf{y}_k))_+ \right) - \left(a_0 \ell(\mathbf{z}_k) + \sum_j a_j (n_j(\mathbf{z}_k))_+ \right) \\ &= \left(a_0 \mathbf{w}_0 + \sum_{j \in I_{\mathbf{z}_k}} a_j \mathbf{w}_j \right) \cdot (\mathbf{y}_k - \mathbf{z}_k) + a_k(\mathbf{w}_k \cdot \mathbf{y}_k + b_k) \\ &\overset{(3)}{=} 0 + a_k(\mathbf{w}_k \cdot \mathbf{y}_k + b_k). \end{aligned}$$

But $\mathbf{y}_k \in \mathbf{R}_k^+$, so $(\mathbf{w}_k \cdot \mathbf{y}_k + b_k) > 0$ and we must have $a_k = 0$. As k was arbitrary, this shows that $a_k = 0$ for all $k \ge 1$. Then (3) immediately shows that also a_0 unless $\mathbf{w}_0 = 0$.

If $\mathbf{w}_0 = 0$, then $c \neq 0$ as we assumed $\ell(x)$ to be non-zero. Choose an arbitrary $\mathbf{x}_0$ with $\mathsf{n}_j(\mathbf{x}_0) \neq 0$ for some j. Then

$$0 = a_0 c + \left(a_0 \mathbf{w}_0 + \sum_{j \in I_{\mathbf{x}_0}} a_j \mathbf{w}_j \cdot \right) \mathbf{x}_0 \overset{(3)}{=} a_0 c = 0$$

Since $c \neq 0$ we must have $a_0 = 0$.

Concluding the proof, we showed that $a_j = 0$ for all $j \ge 0$, hence $\{\ell(x),\ (\mathsf{n}_i(\mathbf{x}))_+ | 1 \le i \le m\}$ is linearly independent.

□

C Proof of Theorem 1

Proof. The functions we consider are (using R2, R3) of the form $f_{\mathbf{W}_i}(\mathbf{x}) = \sum_j v_i^j (\mathsf{n}_i^j(\mathbf{x}))_+ + \mathsf{N}_i(\mathbf{x})$ with neuron functions $\mathsf{n}_i^j(\mathbf{x}) = \mathbf{w}_i^j \cdot \mathbf{x} + b_i^j$ with pairwise different hyperplanes $\mathbf{H}_i^j$ for fixed i. Further $\mathsf{N}_i(\mathbf{x}) = \alpha_i \mathbf{x} + c_i$ is the linear function given by combining the bias of the output layer with a possible linear function implemented by neurons with opposite activation pattern.

By assumption, we have $f_{\mathbf{W}_1}(\mathbf{x}) = f_{\mathbf{W}_2}(\mathbf{x})$ for all $\mathbf{x}$ and therefore $\sum_j v_1^j(\mathsf{n}_1^j(\mathbf{x}))_+ + \alpha_1\mathbf{x} + c_1 = \sum_j v_2^j(\mathsf{n}_2^j(\mathbf{x}))_+ + \alpha_2\mathbf{x} + c_2$, giving

$$\sum_j v_1^j(\mathsf{n}_1^j(\mathbf{x}))_+ - \sum_j v_2^j(\mathsf{n}_2^j(\mathbf{x}))_+ + \ell(\mathbf{x}) = 0$$

where $\ell(\mathbf{x}) = \mathsf{N}_1(\mathbf{x}) - \mathsf{N}_2(\mathbf{x})$ is a linear function.

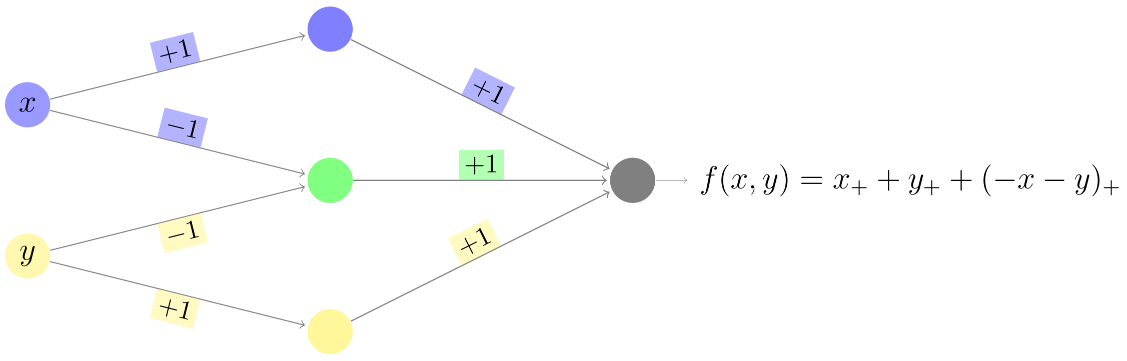
Because of (R1)-irreducibility, the weights v_i^j are nonzero and $\{\ell(\mathbf{x}), (\mathsf{n}_i^j(\mathbf{x}))_+ | \, i, j\}$ is a set of nonzero linearly dependent functions. (Remove ℓ from the set when $\ell(\mathbf{x}) = 0$.) Applying Lemma 2 (property (IP), there are two neurons with identical hyperplanes $\mathbf{H}(\mathsf{n}_{i_1}^{j_1}(\mathbf{x})) = \mathbf{H}(\mathsf{n}_{i_2}^{j_2}(\mathbf{x}))$. Using a suitable transformation ψ_-^J we can even ensure that $\mathbf{R}^+(\mathsf{n}_{i_1}^{j_1}(\mathbf{x})) = \mathbf{R}^+(\mathsf{n}_{i_2}^{j_2}(\mathbf{x}))$, which is equivalent by Lemma 1 to $\mathbf{w}_{i_1}^{j_1} = \lambda\mathbf{w}_{j_2}^{i_2}$ and $b_{i_1}^{j_1} = \lambda b_{i_2}^{j_2}$ for some $\lambda > 0$. By assumption, there are no $k_1 \neq k_2$ such that $(\mathsf{n}_1^{k_1}(\mathbf{x}))_+, (\mathsf{n}_1^{k_2}(\mathbf{x}))_+$ or $(\mathsf{n}_2^{k_1}(\mathbf{x}))_+, (\mathsf{n}_2^{k_2}(\mathbf{x}))_+$ are linearly dependent. Hence, without loss of generality we must have $i_1 = 1$ and $i_2 = 2$ and the linearly dependent pair $\left((\mathsf{n}_1^{j_1}(\mathbf{x}))_+, (\mathsf{n}_2^{j_2}(\mathbf{x}))_+\right)$ is linearly independent of all other $(\mathsf{n}_1^{k_1}(\mathbf{x}))_+$ and $(\mathsf{n}_2^{k_2}(\mathbf{x}))_+$. Therefore, with $\mathbf{w}_1^{j_1} = \lambda\mathbf{w}_2^{j_2}$ and $b_1^{j_1} = \lambda b_2^{j_2}$ we must have $v_1^{j_1} = \frac{1}{\lambda}v_2^{j_2}$ and

$$\sum_{j \neq j_1} v_1^j(\mathsf{n}_1^j(\mathbf{x}))_+ - \sum_{j \neq j_2} v_2^j(\mathsf{n}_2^j(\mathbf{x}))_+ + \ell(\mathbf{x}) = 0,$$

where $\ell(x)$ might have changed due to an application of some transformation ψ_-^J. We have reduced the expression by one summand on each side. Inductively, we can match each $(\mathsf{n}_1^j(\mathbf{x}))_+$ with some $(\mathsf{n}_2^j(\mathbf{x}))_+$ such that each pair differs only by a positive multiplicative number λ. We consecutively match pairs until we have reduced the equation to $\ell(\mathbf{x}) = 0$ since the number of neurons in both representations is necessarily equal. From $\ell(\mathbf{x}) = 0$, we obtain $(c_1 - c_2) = 0 \Leftrightarrow c_1 = c_2$, and $\alpha_1 - \alpha_2 = 0 \Leftrightarrow \alpha_1 = \alpha_2$. This shows that one network can be transformed into the other one by a permutation of nodes π in the hidden layer, transformations ρ_λ multiplying neuron values by a positive scalar which is corrected by the outgoing weights λ, and transformations ψ_-^J flipping activation regions of neurons and changing the linear neuron. $\square$

D Example of a pair of degenerate (R1,R2,R3)-irreducible networks that implement the same network function

The following (R1,R2,R3)-irreducible networks $f_{\mathbf{W}}(\mathbf{x}) = \sum_{j=1}^{3} v_j(\mathsf{n}_j(\mathbf{x}))_+$ have each pairwise different hyperplanes and implement the same network function. They are degenerate in the sense that the sum $\sum_{j=1}^{3} v_j \mathsf{n}_j(\mathbf{x}) = 0$ adds up to the constant zero.



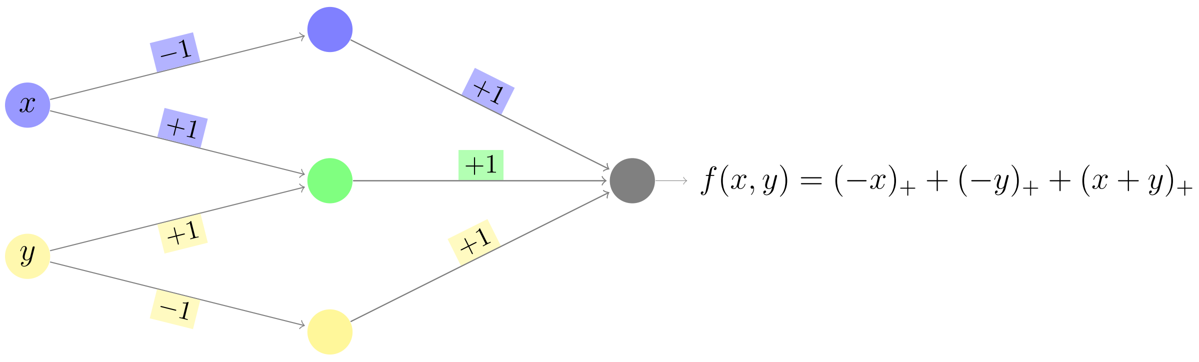
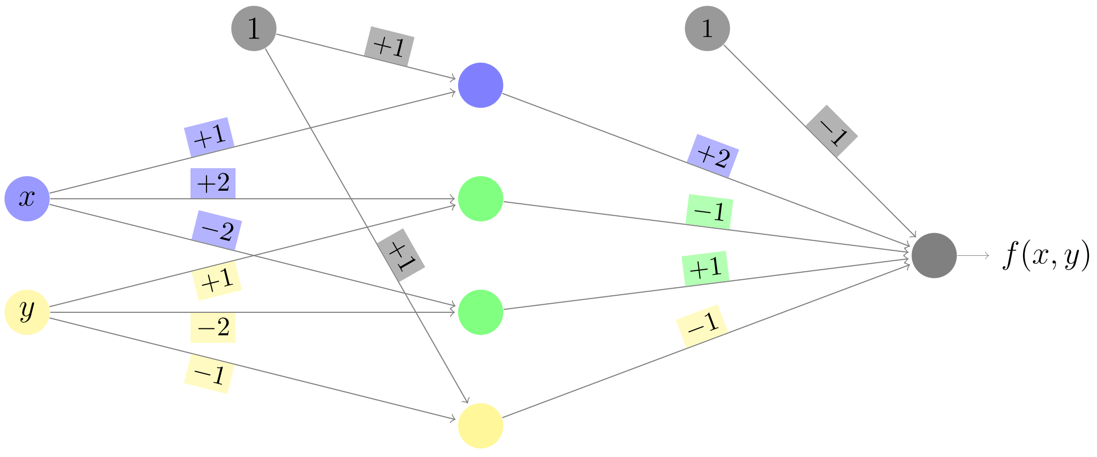


Figure 2: Two (R1,R2,R3)-irreducible ReLU networks implementing the same network function $x_+ + y_+ + (-x-y)_+ = (-x)_+ + (-y)_+ + (x+y)_+$. The two representations differ by $\psi_-^{\{1,2,3\}}$, which flips the sign of all weights of the first layer and adds a linear neuron with function $x + y + (-x-y) = 0$, which can be removed by R1.

A more intricate example of two (R1,R2,R3)-irreducible networks $f_{\mathbf{W}}(\mathbf{x}) = \sum_{j=1}^{4} v_j(\mathsf{n}_j(\mathbf{x}))_+ + b$ with pairwise different hyperplanes and implementing the same network function is the following. They are degenerate in the sense that the sum $\sum_{j\in\{1,2,4\}} v_j \mathsf{n}_j(\mathbf{x})$ adds up to a constant.



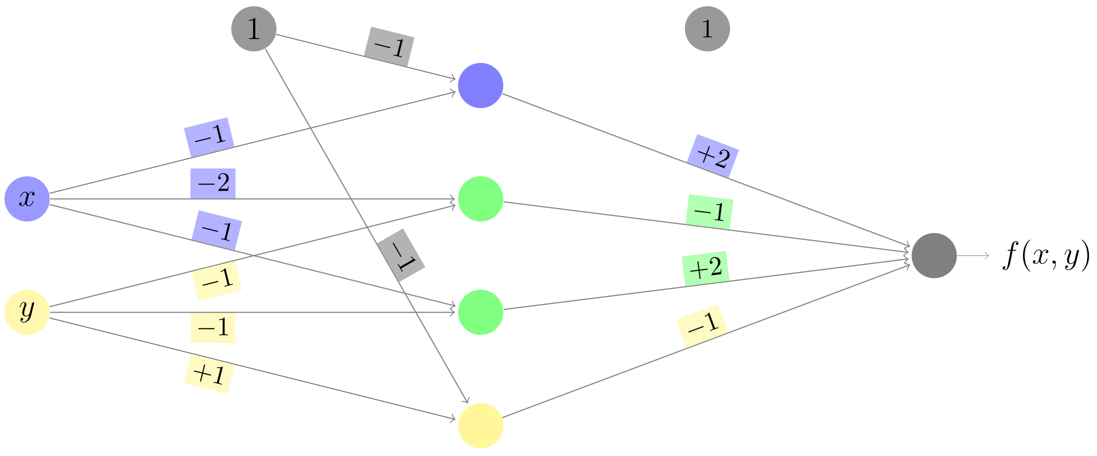


Figure 3: Two (R1,R2,R3)-irreducible ReLU networks implementing the same network function. The second representation can be obtained from the first one by ρ_2 applied to the 3rd neuron and $\psi_{-}^{\{1,2,4\}}$, which flips the sign of all ingoing weights of neurons 1,2 and 4 and adds a linear neuron with function $2(1+x)-(2x+y)-(1-y)=2+2x-2x-y-1+y=1$, which cancels the bias at the output layer.