SYMMETRIES AND DISCRIMINABILITY IN FEEDFORWARD NETWORK ARCHITECTURES

John Shawe-Taylor
Department of Computer Science
Royal Holloway and Bedford New College
University of London

December 10, 1992

Abstract

The paper investigates the effects of introducing symmetries into feedforward neural networks in what are termed Symmetry Networks. This technique allows more efficient training for problems in which we require the output of a network to be invariant under a set of transformations of the input. The particular problem of graph recognition is considered. In this case the network is designed to deliver the same output for isomorphic graphs. This leads to the question of which inputs can be distinguished by such architectures. A theorem characterising when two inputs can be distinguished by a Symmetry Network is given. As a consequence a particular network design is shown to be able to distinguish non-isomorphic graphs if and only if the graph reconstruction conjecture holds.

Key words. feedforward neural network, Symmetry Network, graph reconstruction, invariance, discriminability, automorphism group.

AMS(MOS) subject classifications. 05C25, 05C60, 20B25, 68E10, 68G05, 68G10.

Abbreviated title. Symmetry Networks.

1 Introduction

In their book on perceptrons Minsky and Papert [10] prove a number of theorems about the limited computational power of various classes of perceptrons. One of their key tools is the so-called Group Invariance Theorem. This theorem shows that for a group whose action on the set of (input) predicates is closed, the output of the perceptron will be invariant under the action of the group if and only if the weights can be chosen to be preserved by the group. The 'only if' result is used by Minsky and Papert in their proofs that certain functions cannot be computed, since it simplifies the weight assignments that need to be considered. However, from a practical point of view the 'if' result could be viewed as suggesting a method of simplifying training in the case where the target function is known to be invariant under the action of a particular group. In this case we can restrict the weights to guarantee invariance. It is this aspect of the group invariance theorem which is generalised to multi-layer perceptrons in this paper. The idea of introducing weight equivalences or symmetries to multi-layer perceptrons is not new. Ad-hoc attempts to introduce invariance have not proved very successful, see for example [11, 3]. An example of a more structured and very successful approach is that of Time Delay Neural Networks or TDNNs [5]. In these networks a layer of neurons is constrained to be connected via a translated set of connections (with the same set of adaptable strengths). This can be viewed as introducing translation invariance into initial layers of a network. A similar technique has been used by Le Cun [6] with great success in recognising handwritten digit strings [8]. More explicit introduction of automorphisms has been reported by Simard *et al.* [16] in a technique termed Tangent Prop again with considerable success, though the introduction here is via adaptions to the learning rather than to the network itself. Our approach aims to simplify the training task as much as is possible by making the symmetries explicit in the network structure a priori.

As an example of the application of these ideas we consider networks designed to recognise isomorphism classes of graphs. This introduces the question of which inputs can be distinguished by a particular network architecture. This problem is studied both for particular networks and a general theorem about discriminability of Symmetry Networks is given. The theorem characterises the discriminability in terms of the symmetries of an associated 'dismembered' network. This gives a direct general tool for assessing the power of a Symmetry Network, and thus suggesting how to choose an appropriate network in a particular application. As an example we construct a particular network which can distinguish all pairs of non-isomorphic graphs if and only if the graph reconstruction conjecture holds. This is a conjecture which has held the attention of graph theorists for a number of years (see for example [2, 4]). It states that with the exception of two vertex graphs, we can identify the isomorphism class of a graph from its vertex deleted subgraphs.

2 Notation and Definitions

We begin with definitions relating to feedforward Neural Networks but will assume familiarity with the basic conecpts, see for example [9]. For simplicity we will restrict the activation function to be the same function for all nodes of the network and we denote the function by f. So we have

$$f : \mathcal{R} \to \mathcal{I},$$

which is traditionally a monotonic function where $\mathcal{R}$ denotes the set of real numbers and $\mathcal{I}$ an interval on the real line. This interval is usually taken as $[0,1]$ or $[-1,1]$. If we wish to limit consideration to networks with boolean values then we may take f to be a threshold function and the "interval" $\mathcal{I}$ to be the set $\{0,1\}$. A network $\mathcal{N}$ is specified by a set N of nodes some of which are distinguished as outputs and a set I of input nodes. The connectivity is given by a set E of directed connections between the nodes together with connections to all computational nodes from an additional 'threshold' input which always has activation 1 (see below).
With network $\mathcal{N}$ we associate a weight function on the set of connections,

$$w : E \to \mathcal{R}.$$

We say that the network $\mathcal{N}$ is in state w.
For notational convenience choose a numbering for all nodes and refer to each node by its number. For a connection $(i,j) \in E$ we denote $w(i,j)$ by w_{ji}. Note that the weight inherits its direction from the underlying connection.
For the purposes of this paper we will assume throughout that if node i is connected to node j then $j > i$. Such a numbering can always be found provided the network is cycle free. This is the feedforward condition on the connectivity.
For a feedforward network it makes sense to introduce the *level* of a node. This is defined as the number of connections in the longest (directed) path from an input node. Input nodes are at level 0 while nodes connected only to input nodes are at level 1, etc. Let ℓ_i denote the level of node i. The feedforward condition on the numbering will be satisfied by requiring

$$\ell_j > \ell_i \Rightarrow j > i,$$

since if node i is connected to node j then certainly $\ell_j > \ell_i$.
The input of a network is specified by a function $\mathbf{i}$ from the set of non threshold input nodes to $\mathcal{I}$. Each node j also has an activation value o_j. For the threshold node this is 1, for other input nodes it is the value of $\mathbf{i}$ at j, while for computational nodes it is the weighted sum of its inputs passed through the activation function. For a computational node the weighted sum of its inputs is also termed the *net input* of the node. The output of the whole network is the vector of activation values on the output nodes. We denote this vector function of the weights and inputs by $F_{\mathcal{N}}(w, \mathbf{i})$.

A *permutation* of a set X is a bijection of X onto itself. A permutation ϕ of X fixes $x \in X$ if $\phi(x) = x$. The permutation ϕ *stabilises* a set $Y \subseteq X$ if $\phi(Y) = Y$, that is elements of Y are mapped to elements of Y. A *permutation group* of a set X is a set Φ such that $\phi \in \Phi$ determines a permutation of X, and Φ is closed under composition and inverses of mappings. We also say that the group Φ *acts* on the set X. The group Φ is *faithful* if no two elements determine the same permutation of X. For a set X we denote by $\Sigma(X)$ the group of all permutations of X. The *orbits* of a set X under the action of a permutation group Φ are the minimal subsets of X that are stabilised by all elements of Φ. Note that the orbits partition X. The orbit containing a point $x \in X$ is given by

$$\{\phi(x) | \phi \in \Phi\}.$$

The group Φ is *transitive* on a set X if X is the single orbit. An *automorphism* or *symmetry* of a network $\mathcal{N}$ is a permutation of the set of nodes of $\mathcal{N}$ which fixes output and threshold nodes, stabilises the input nodes and whose action on the pairs of nodes stabilises connections.

We say that an automorphism γ of a network $\mathcal{N}$ *preserves* the weight assignment w if $w_{ji} = w_{(\gamma j)(\gamma i)}$ for all connections $(i, j) \in E$.

Figure 1 illustrates a network with a single weight preserving automorphism ϕ which (simultaneously) interchanges the two input and two hidden nodes. Note that the input nodes are depicted with squares, while computational nodes are depicted with circles. The network has a single output node at the top of the figure which is (by definition) fixed by ϕ. The connections to the threshold input are denoted by edges leaving the computational nodes horizontally. The connection strengths are given on the connections. If we give the weights the values, $w_1 = w_4 = +1$, $w_2 = -1$, $w_3 = w_5 = -0.5$, the network will compute the *xor* function.

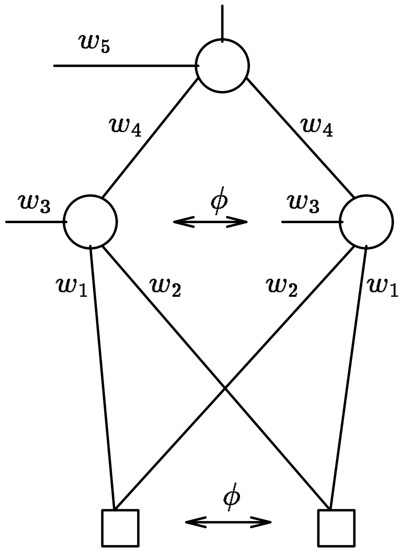
Let γ be an automorphism of a network $\mathcal{N}$ and $\mathbf{i}$ an input of $\mathcal{N}$. We denote by $\mathbf{i}^\gamma$ the input given by

$$\mathbf{i}^\gamma(k) = \mathbf{i}(\gamma^{-1}k).$$

A *graph* is a set of vertices together with a subset of the unordered pairs of vertices called the set of edges. We restrict graphs to have no loops, that is the edges must be pairs of distinct vertices. By $\mathcal{G}_n$ we denote the set of graphs on n vertices. A vertex u is *adjacent* to a vertex v in a graph G, denoted $u \sim v$, if (u, v) is an edge of G. The *neighbours* of a vertex v of a graph G are the vertices of G adjacent to v. The *degree* or *valency* of a vertex is the number of its neighbours. An *isomorphism* of a graph G_1 to a graph G_2 is a bijection ϕ of the vertices of G_1 to the vertices of G_2 such that for u, v vertices of G_1,

$$u \sim v \Leftrightarrow \phi(u) \sim \phi(v).$$

Figure 1: Example of a network automorphism



3 Multi-Layer Perceptrons

In their book on perceptrons Minsky and Papert [10] prove a number of theorems about the limited computational power of various classes of perceptrons. One of their key tools is the so-called Group Invariance Theorem. Stated in our terminology, this theorem shows that for a group whose action on the set of (input) predicates is closed, the output of the perceptron will be invariant under the action of the group if and only if the weights can be chosen to be preserved by the group. It is the 'if' part of the group invariance theorem which is generalised to multi-layer perceptrons in the following theorem. Our approach aims to simplify the training task as much as is possible by making the symmetries explicit in the network structure a priori.

Theorem 3.1 *Let γ be a weight preserving automorphism of the network $\mathcal{N}$ in state w. Then for every input vector* $\mathbf{i}$

$$F_{\mathcal{N}}(w, \mathbf{i}) = F_{\mathcal{N}}(w, \mathbf{i}^{\gamma}).$$

Proof: We prove by induction on the layers of the network that

$$o_t^{\gamma} = o_{\gamma^{-1}t},$$

where o (o^{γ}) denotes the vector of activations of nodes of the network when the input is $\mathbf{i}$ ($\mathbf{i}^{\gamma}$). Since for nodes on the input layer (level 0), this is true by the definition of $\mathbf{i}^{\gamma}$, we assume that it is true for all nodes on level $l-1$ and less. Let t

be a node on level l and $h = \gamma^{-1}$. We have

$$\begin{aligned} o_t^\gamma &= f\left(\sum_{\ell_j < l} w_{tj} o_j^\gamma\right) \\ &= f\left(\sum_{\ell_j < l} w_{(ht)(hj)} o_{hj}\right) \\ &= o_{ht}, \end{aligned}$$

as required. Hence by induction $o_t^\gamma = o_{\gamma^{-1}t}$ for all nodes t.
Let t be an output node. By the above $o_t^\gamma = o_{\gamma^{-1}t}$. But since t is an output node it is fixed by γ and so $\gamma^{-1}t = t$ giving $o_t^\gamma = o_t$. Hence

$$F_{\mathcal{N}}(w, \mathbf{i}) = F_{\mathcal{N}}(w, \mathbf{i}^\gamma),$$

as required. ∎

Following this theorem we introduce the concept of a *Symmetry Network.* This is a pair $(\mathcal{N}, \Gamma)$, where $\mathcal{N}$ is a network and Γ a group of automorphims of $\mathcal{N}$. An *allowable* weight assignment w for a Symmetry Network $(\mathcal{N}, \Gamma)$ is one which is preserved by all $\gamma \in \Gamma$. For a Symmetry Network $(\mathcal{N}, \Gamma)$, the orbits of the set E of connections of $\mathcal{N}$ under the action of Γ are called *weight classes.* An allowable weight assignment is constant on each weight class. A *weight class preserving* automorphism is an automorphism of $\mathcal{N}$ which stabilises weight classes. For a Symmetry Network $(\mathcal{N}, \Gamma)$ each $\gamma \in \Gamma$ is a weight class preserving automorphism, but there may be additional automorphisms which preserve weight classes (e.g. see Example 6.3).
A network $\mathcal{N}$ is invariant under an automorphism γ if $F_{\mathcal{N}}(w, \mathbf{i}) = F_{\mathcal{N}}(w, \mathbf{i}^\gamma)$, for all inputs $\mathbf{i}$. Hence Theorem 3.1 states that a network in state w is invariant under automorphisms preserving w. The theorem suggests a practical way of simplifying the task of training networks required to be invariant under certain automorphisms. The network is constructed so that the group of automorphisms is extended from the input layer to act as a group of automorphisms of the whole network. The weight assignments are restricted to those allowable under the automorphisms. In this way the network output will at all times be invariant under the automorphisms. Hence to train such a network we need train it with each input in only one 'position'. To do this learning algorithms must be adapted in order to ensure that weight assignments are at all times allowable. As an example the network $(\mathcal{N}, \{\phi\})$ of Figure 1 is a Symmetry Network whose output is invariant under interchanging the components of the input vector. The *xor* function is invariant under this automorphism and so we could train this network to compute the *xor* function with five parameters and three examples, instead of the more normal nine parameters and four examples.
If we view the network as a function whose input-output performance depends on parameters, the standard algorithms can be readily adapted to update the parameters of a Symmetry Network. In this case the parameters are the values of weights for each weight class.

For example, this can be done for the generalised delta rule [13] and the linear programming algorithm [15]. However, in the experiments reported below, it was found that the generalised delta rule exhibited inconsistent and unpredictable behaviour and we relied almost entirely on the linear programming algorithm. The linear programming algorithm is also a gradient descent approach, but the direction of descent is chosen so that none of the individual errors for the different inputs increases. This eliminates the problem commonly experienced with back-propagation where one error increases to 1 while the others decrease. The algorithm must therefore be run in batch mode and each iteration involves solving a linear programme to obtain the direction of descent. Though each iteration is significantly more expensive than for back-propagation, experience has shown that in many cases overall performance is superior both in reliability and speed.

Example 3.1 *A Graph Recognition Symmetry Network of order n is a Symmetry Network $(\mathcal{N}, S_n)$ with inputs indexed by unordered pairs of distinct graph vertices, where $S_n = \Sigma(\{1, \ldots, n\})$ is the set of all permutations of the n vertices of the graph. Since the action of the symmetric group corresponds to renumbering the vertices of the graph, the output of the network will be invariant under such renumberings, or in other words will be the same for isomorphic graphs.*

In the next section we describe experiments performed with one such network. The advantage obtained in this case is that only one graph in each isomorphism class needs to be presented during training. This considerably reduces the size of the training sample in the example given and guarantees correct output for all graphs isomorphic to a graph in the training sample. In later sections we will develop tools to investigate how such networks might be used to test for graph isomorphism.

4 Experiment

Our experiment was inspired by work reported by Dodd [3]. He succeeded in training a multi-layer perceptron to recognise three isomorphism classes of graphs on five vertices. The network contained two hidden layers of 11 and 7 nodes. Each of the three graphs considered had five edges and one automorphism of order 2. This meant that there were 60 distinct labellings of each graph. Hence the network was trained with 180 inputs each with 3 outputs, the one corresponding to the isomorphism class of the input graph being set to one.

We took a network as close as possible in layout to Dodd's. The input layer in Dodd's network consisted of 25 nodes one for each pair of vertices. Both pairs corresponding to an edge of the graph were set to 1 if the edge was present and 0 otherwise. In our network this was simplified to just 10 input nodes, one for each of the possible edges (that is unordered pairs of vertices).

The hidden layers matched Dodd's layout more closely. This was achieved by making the nodes of the first hidden layer correspond to triples of vertices (there are

10 of these) and the second hidden layer corresponds to 4-tuples (there are five of these). All possible connections consistent with the feedforward condition and non-interconnection within layer (including the output layer) were included.
It should be noted that if this network were generalised to n vertex graphs it would contain exponentially many nodes. We consider a network with similar layout but leaving out many of the connections later in this paper (see Example 7.1).
The action of the group S_5 on the pairs (edge inputs), triples (first layer), 4-tuples (second layer) and fixing the three output nodes dictated which connection weights should be constrained to be equal. For example between the input and triples there were three connection types, one corresponding to a pair being contained in a triple, one to a pair being disjoint from a triple and one to their having a one vertex intersection. In all there were 293 connections but only 21 different types.
The training set was composed of one example of each of the 34 isomorphism classes of graphs on five vertices. The desired outputs were all zero except for one output 1 for each of the three graphs used by Dodd.
The corresponding training set, were the symmetries not incorporated, would be all $2^{10} = 1024$ labelled graphs on five vertices. Hence by incorporating the symmetries we have reduced the training constraints from $3 \times 1024 = 3072$ to just $3 \times 34 = 102$, while the number of degrees of freedom has been reduced from 293 to 21. The training was successfully completed with just 350 iterations of the linear programming algorithm. This compares very favourably with the 40000 generalised delta rule iterations reported by Dodd [3]. The resulting network satisfied 3072 constraints with just 293 weights. Theory indicates that in training networks as many weights as constraints may be required [15], so it is clear that the network has encoded the problem in an efficient way. For information on further experiments with Symmetry Networks the reader is referred to [14].

5 Conclusions for Graph Isomorphism

Most computational problems that have been studied have either been classified as NP-complete or a polynomial time algorithm has been found for them. Note that NP-complete problems are the hardest modulo polynomial time reductions among those whose solutions can be checked in polynomial time. It is widely conjectured that such problems cannot be solved in polynomial time and certainly no polynomial algorithms are known for such problems. Graph Isomorphism is the problem of determining whether two graphs are isomorphic, in the sense that they are the same up to a relabelling of their vertices. This problem is unusual as it has not been shown to be NP-complete nor has a polynomial algorithm been found for it. A major inroad was made by Luks [7] who showed using group algorithms that the problem can be solved in polynomial time for bounded valency graphs. Stated more precisely he showed that given any number k, a polynomial time isomorphism algorithm existed for graphs whose vertices had valency at most k. For $k = 3$ this technique has been refined to give an algorithm which is also efficient in practice. Graph Recognition Symmetry Networks suggest a number of methods of tackling the graph isomorphism problem. Perhaps the most naive approach would involve building a network to recognise one of the graphs in such a way that it would give a different output for all non-isomorphic graphs. This network could then be observed when the two graphs are input. The graphs are isomorphic if and only if the outputs are identical. The problem with the approach is finding a design strategy calculable in polynomial time from the structure of the graph. This seems a daunting task and we would conjecture that in general an exponential number of nodes will be required. Indeed in Section 7 we will describe a network which can be made to distinguish any given pair of graphs and find we need exponentially many nodes even though we do not require that other graphs give outputs different from those of the two graphs.

There is, however, a more realistic prospect of a probabilistic algorithm for testing graph isomorphism. A *probabilistic algorithm* is a polynomial algorithm for a decision problem which is run with reference to a randomly generated number. If the algorithm returns 'no', this is the correct classification of the input. If the correct classification is 'no', there is, however, a certain probability that the algorithm returns 'yes', while if the correct classification is 'yes' the algorithm will always return 'yes'. By running the algorithm repeatedly with independently generated random numbers, we can ensure that the probability of our incorrectly classifying 'yes' as a result of repeated misclassifications is made arbitrarily small. Primality testing is an example of a problem for which there exist probabilistic algorithms [12] but no known polynomial deterministic algorithm. The conjectured probabilistic algorithm for graph isomorphism would involve the random generation of Graph Recognition Symmetry Networks which would be used to test the two input graphs. If the two graphs gave different outputs we can say with certainty that they are not isomorphic. If on the other hand they deliver the same result the probability of the graphs being not isomorphic should be reduced by a constant factor. Probabilistic algorithms should be distinguished from algorithms with polynomial expected complexity. Expected complexity results rely on specifying a probability distribution on the set of inputs and average the complexity over this distribution. They are therefore unable to give useful bounds for the time taken for a particular input. In the case of graph isomorphism Babai, Erdös and Selkow [1] show that for almost all graphs there is a linear time isomorphism test. It is the 'almost all' which indicates a probabilistic estimate under the uniform distribution. The proposed probabilistic algorithm for graph isomorphism would rely on generating random networks which distinguish the two graphs with positive probability. We therefore turn our attention to considering which graphs a given network architecture can distinguish. This question will motivate the development in the following sections, though our main result (Theorem 6.2) will characterise which inputs can be distinguished for all Symmetry Networks subject to reasonable restrictions on the activation function.

6 Discriminability of Symmetry Networks

For a Symmetry Network $(\mathcal{N}, \Gamma)$, we define an equivalence relation on its set of possible inputs by:

$$\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2 \iff F_{\mathcal{N}}(w, \mathbf{i}_1) = F_{\mathcal{N}}(w, \mathbf{i}_2) \quad \forall \text{ allowable } w.$$

If $\mathbf{i}_1 \not\cong_{\mathcal{N}} \mathbf{i}_2$, we say that $\mathcal{N}$ can distinguish $\mathbf{i}_1$ and $\mathbf{i}_2$. Theorem 3.1 states that $\mathbf{i} \cong_{(\mathcal{N},\Gamma)} \mathbf{i}^\gamma$, for all $\gamma \in \Gamma$. The question addressed in the remainder of this paper is how to determine which other pairs of inputs are equivalent and so cannot be distinguished by a particular network. In the case where the network $(\mathcal{N}, S_n)$ takes inputs which are n vertex graphs, we will define an equivalence relation on the set $\mathcal{G}_n$ of such graphs by:

$$G_1 \cong_{\mathcal{N}} G_2 \iff F_{\mathcal{N}}(w, \mathbf{i}_{G_1}) = F_{\mathcal{N}}(w, \mathbf{i}_{G_2}) \quad \forall \text{ allowable } w.$$

We will now give some examples of Symmetry Networks and discuss their discriminatory power. Our first example is due to an anonymous reviewer and demonstrates that perfect discriminability is always attainable if we introduce a hidden neuron for each element of the group of symmetries.

Example 6.1 *Consider an n input line network with a group Γ permuting the input lines. We will construct a Symmetry Network $(\mathcal{N}, \Gamma)$ with perfect discriminability. The input layer is given as n input nodes, while the hidden layer has $|\Gamma|$ nodes labelled by the elements of Γ. The network has a single output node connected to the hidden units. The permutation ϕ sends the hidden node indexed by permutation ψ to the node indexed by permutation $\psi\phi^{-1}$. All possible connections between the input and hidden nodes are included. There are n weight classes among these connections where the weight class of a connection from input i to hidden node ψ is characterised by the value $\psi(i)$. This follows since the connection (i, ψ) is mapped to $(\phi(i), \psi\phi^{-1})$ by a permutation ϕ, while if $\psi'(i') = \psi(i)$, then $\psi'^{-1}\psi$ maps the connection (ψ, i) to (ψ', i'), since*

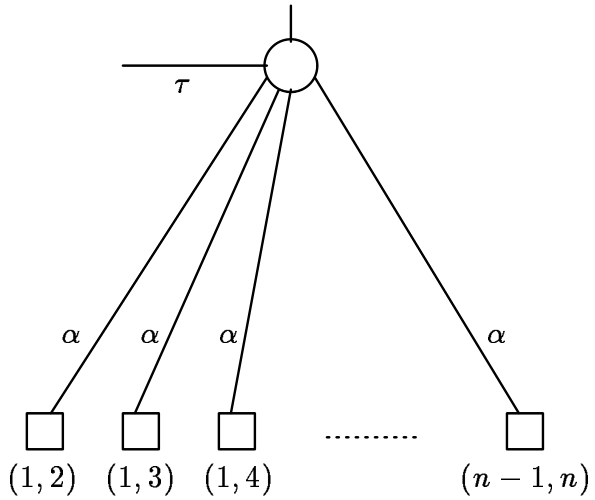
$$\psi'^{-1}\psi(i) = \psi'^{-1}\psi'(i') = i'.$$

We will use our main result to show later that with appropriate choice of activation function the Symmetry Network $(\mathcal{N}, \Gamma)$ of Example 6.1 has full discriminability on binary inputs, that is

$$\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2 \iff \mathbf{i}_1 = \mathbf{i}_2^\gamma \quad \text{for some } \gamma \in \Gamma.$$

Example 6.2 *The network illustrated in Figure 2 is a graph recognition network of order n having a single computational node with all $n(n-1)/2$ inputs connected to it. Since S_n is transitive on unordered pairs of distinct vertices, there are two weight classes, one for the threshold and the second for the connections. Hence the group of weight class preserving automorphisms is the group $S_{n(n-1)/2}$ of all permutations of the input nodes.*

Figure 2: Network of Example 6.1



The equivalence classes of this network are the graphs with equal numbers of edges, since the output is

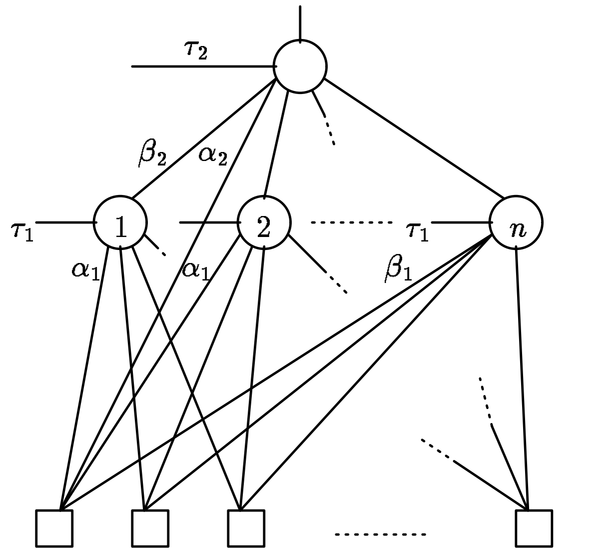
$$f(|E|\alpha + \tau),$$

where α is the weight on all connections except the threshold connection to the output node which has weight τ, and E is the set of edges of the input graph.
Note that there are also weight preserving automorphisms of this network mapping between two inputs if and only if the graphs have the same number of edges.
This observation leads us to ask whether this is true in general. In other words is $G \cong_{\mathcal{N}} G'$, if and only if there exists a weight class preserving automorphism of $\mathcal{N}$ which maps $\mathbf{i}_G$ to $\mathbf{i}_{G'}$? To investigate this question we introduce another Graph Recognition Symmetry Network.

Example 6.3 *The Graph Recognition Symmetry Network of order n illustrated in Figure 3 has a hidden layer of nodes indexed by the vertices* $\{1, \ldots, n\}$ *of the graph with* S_n *acting on the hidden layer in the natural way. The connections between input nodes indexed by an unordered pair and hidden nodes indexed by a member of the pair have weight* α_1*, while for hidden nodes indexed by a vertex not in the pair the weight is* β_1*. The output node has one weight class to input nodes with weight* α_2 *and one weight class to hidden nodes with weight* β_2*. The threshold connections are* τ_1 *for the hidden nodes and* τ_2 *for the output node. In Figure 3, the weights are only marked for connections from the input node* $(1, 2)$*, the hidden node* 1 *and the threshold node in order to avoid cluttering the diagram.*

The graphs which can be distinguished by this network are slightly harder to characterise. We will need the following definitions. The *degree sequence* of a graph G is the sequence of degrees $(d_1, d_2, \ldots, d_n)$ of the vertices of G placed in descending order. The degree sequence is clearly invariant under renaming of the vertices –

Figure 3: Network of Example 6.2



that is up to isomorphism. There are, however, graphs which are not isomorphic which have the same degree sequence. For example the cycle on six vertices has the same degree sequence as two cycles of three vertices. The following theorem characterises the distinguishing power of the network of Example 6.3 in terms of the degree sequence. This will also provide a counter example to the suggestion that $G \cong_{\mathcal{N}} G'$ implies the existence of a weight class preserving automorphism of $\mathcal{N}$ which maps $\mathbf{i}_G$ to $\mathbf{i}_{G'}$

Theorem 6.1 *For $(\mathcal{N}, S_n)$ the Symmetry Network of Example 6.3:*
(i) $G_1 \cong_{\mathcal{N}} G_2 \Leftrightarrow G_1$ and G_2 have the same degree sequence,
(ii) However $G_1 \cong_{\mathcal{N}} G_2$ does not imply the existence of a weight class preserving automorphism of $\mathcal{N}$ such that $\mathbf{i}^{\phi}_{G_1} = \mathbf{i}_{G_2}$.

Proof: (i) ($\Rightarrow$) Suppose that there is a degree d^* such that w.l.o.g. G_1 has more vertices than G_2 with degree greater than or equal to d^*. Assume further that d^* is the largest number with this property. By setting $\beta_1 = 0$ and α_1 large enough, with τ_1 sufficiently negative, we make the activation of all internal nodes except that corresponding to vertices with degree greater than or equal to d^* insignificant (If the activation function is a threshold function then we may arrange the weights so that for a hidden node to be switched on at least d^* of its inputs must be active). Hence since the effects of vertices with degrees greater than d^* are the same for both graphs (d^* is the largest number with the property), we may arrange for the network to give different outputs for the two graphs, a contradiction. We conclude that for all d, the two graphs have the same number of vertices with degree greater than or equal to d. This implies the two graphs have the same degree sequence.

($\Leftarrow$) If G_1 and G_2 have the same degree sequence the activations of hidden nodes corresponding to vertices of the same degree are the same. Hence the final node of the network has net input equal to β_2 times the sum of these hidden node activations plus α_2 times the number of edges in the graph. But graphs with the same degree sequence have the same number of edges and so the output of whole network is the same for both graphs.
(ii) To show this is not the case consider a weight class preserving automorphism ϕ of the network mapping $\mathbf{i}_{G_1}$ to $\mathbf{i}_{G_2}$. With an abuse of notation we label the nodes of the hidden layer by the vertices of G_1 (see Figure 3). Likewise the input layer is labelled by unordered pairs of G_1's vertices. The labelling is given by the input $\mathbf{i}_{G_1}$. This means that ϕ can also be viewed as acting on the vertices of G_1 through its action on the hidden layer. Consider network nodes u, v of the hidden layer and the input node of the pair (u, v) corresponding to two vertices u and v of G_1. Since the connections in $\mathcal{N}$ from (u, v) to the nodes u and v are the only two weights leaving (u, v) which are in the weight class denoted by α_1 in Example 6.3, the automorphism ϕ must map (u, v) to the input node of the pair $(\phi(u), \phi(v))$ by weight class preservation. Hence (u, v) is an edge of G_1 precisely when $(\phi(u), \phi(v))$ is an edge of G_2 under the labelling given by $\mathbf{i}_{G_2}$, thus making ϕ an isomorphism from G_1 to G_2. Since not all graphs with the same degree sequence are isomorphic, not all output invariance is a result of weight class preserving automorphisms of the network. ∎

Theorem 6.1 is disappointing in that it shows the automorphisms of a Symmetry Network do not in themselves characterise the equivalence classes of inputs to that network. In order to capture the discriminating power in a network structure it is necessary to transform the original Symmetry Network into a tree-like network. Intuitively we can think of cutting apart any connections which leave a node by duplicating the node once for each of the connections. The weight classes are inherited from the original network in the obvious way. We will show that for appropriate choice of activation functions this transformed network captures precisely the discriminating power of the original Symmetry Network.
We introduce for a network $\mathcal{N}$ with a single output node κ, its *dismembered network* $\mathcal{N}'$. Apart from a threshold node, the nodes of $\mathcal{N}'$ will correspond to paths up through the network $\mathcal{N}$ not including the threshold node and ending at the single output node:

$$(i_1, i_2, \ldots, i_\ell = \kappa)$$

where $(i_j, i_{j+1}) \in E$ for $j = 1, \ldots, \ell - 1$. The input nodes of $\mathcal{N}'$ will be those paths which start at an input node of $\mathcal{N}$, while the output node will be the simple path (κ) consisting of the output node of $\mathcal{N}$. The edges of $\mathcal{N}'$ are firstly one edge connecting the threshold node to each non-input node. The other edges connect each path to that obtained by deleting its first node. The weight class of this edge is that corresponding to the edge in $\mathcal{N}$ which connected the first two nodes of the longer path. This condition means that the action of a weight class preserving automorphism of

$\mathcal{N}$ on the paths of $\mathcal{N}$ induces a weight class preserving automorphism of $\mathcal{N}'$, which fixes the output node. Hence if $(\mathcal{N}, \Gamma)$ is a Symmetry Network, so is $(\mathcal{N}', \Gamma')$, where Γ' is the set of automorphisms of $\mathcal{N}'$ which stabilise the weight classes inherited from $\mathcal{N}$. The above observation means that Γ is a subgroup of Γ'. There is also a natural extension of an input $\mathbf{i}$ to $\mathcal{N}$ to the corresponding *dismembered* input $\mathbf{i}'$ of $\mathcal{N}'$, where $\mathbf{i}'$ at an input node $\mathbf{v}$ of $\mathcal{N}'$ is the value of $\mathbf{i}$ on the input node of $\mathcal{N}$ at the beginning of the path $\mathbf{v}$.

Note that for Example 6.2 the dismemberment of the network is just the network itself, while in Example 6.3 the dismemberment contains the same number of hidden nodes, but $n+1$ times as many input nodes. One copy of the inputs is connected directly to the ouput node and the other n copies are connected each to one of the hidden nodes.

The next proposition shows that when we present a dismembered input to a dismembered network, the activations correspond to those of the original network.

Proposition 6.1 *If the dismembered input* $\mathbf{i}'$ *of* $\mathbf{i}$ *is presented to the dismembered network* $\mathcal{N}'$ *of* $\mathcal{N}$*, then the activation at node* $\mathbf{v}$ *of* $\mathcal{N}'$ *is equal to that of the node of* $\mathcal{N}$ *at the beginning of the path* $\mathbf{v}$*.*

Proof: Let $\mathbf{v}$ be a longest path labelling a node of $\mathcal{N}'$ for which the activation of $\mathbf{v}$ in $\mathcal{N}'$ is not equal to that of the node of $\mathcal{N}$ at its beginning. If $\mathbf{v}$ is an input node, we have a trivial contradiction, since by the definition of a dismembered input the two activations must be equal. Assume therefore that $\mathbf{v}$ is a hidden node and let k be the node of $\mathcal{N}$ occurring at the beginning of the path $\mathbf{v}$. For each edge (j, k) of $\mathcal{N}$ there is an edge $(\mathbf{v}_j, \mathbf{v})$ of $\mathcal{N}'$ with the same weight, where $\mathbf{v}_j$ is the path obtained from $\mathbf{v}$ by adding j at the front. Since the paths are all longer than $\mathbf{v}$ and the assumption that $\mathbf{v}$ was the longest path to fail the proposition, the activation of the node j and $\mathbf{v}_j$ are equal, for each j. But this implies the net input to $\mathbf{v}$ in $\mathcal{N}'$ is equal to the net input to k in $\mathcal{N}$, implying that there activations are also equal. ∎

Before we make use of the idea of a dismembered network, we need to consider restrictions on the activation functions. To this end we introduce the following definitions.

A function f *differentiates polynomials* if for any polynomials $q_i(x)$, $i = 1, \ldots, n$, such that $q_i(x) - q_j(x)$ is not constant for $i \neq j$, then the functions $f(q_i(x))$, $i = 1, \ldots, n$, are linearly independent over the reals.

A function f *differentiates positive polynomials* if for any non constant polynomials $q_i(x)$, $i = 1, \ldots, n$, with linear and higher degree coefficients not negative, such that $q_i(x) - q_j(x)$ is not constant for $i \neq j$, the functions $f(q_i(x))$, $i = 1, \ldots, n$, are linearly independent over the reals.

Proposition 6.2 *An example of a function which differentiates polynomials is the exponential function*

$$f(x) = e^x.$$

Proof: We will prove the stronger result that for any polynomials $q_i(x)$ and rational functions $p_i(x)$, $i = 1, \ldots, n$, if

$$\sum_{i=1}^{n} p_i(x) f(q_i(x)) \equiv 0,$$

with the differences $q_i(x) - q_j(x)$ not constant for $i \neq j$, then $p_i(x) \equiv 0$, for all i. That is that the functions $f(q_i(x))$, $i = 1, \ldots, n$, are independent over rational functions. If $n = 1$ we must clearly have $p_1(x) \equiv 0$, so there is nothing to prove. Assume the result holds for smaller integers than n, and that all the $p_i(x) \not\equiv 0$. Dividing through by $p_1(x) f(q_1(x))$, gives the equation

$$1 + \sum_{i=2}^{n} p_i^\star(x) f(q_i(x) - q_1(x)) \equiv 0,$$

where $p_i^\star(x)$ is the rational function $p_i(x)/p_1(x)$. Since the above equation holds on a non-trivial neighbourhood of a point x_0, we may differentiate on this neighbourhood to obtain

$$\sum_{i=2}^{n} (p_i^{\star\prime}(x) + p_i^\star(x)(q_i'(x) - q_1'(x))) f(q_i(x) - q_1(x)) \equiv 0.$$

Since the difference between $q_i(x) - q_1(x)$ and $q_j(x) - q_1(x)$ is not constant, we can apply the induction hypothesis to conclude that

$$p_i^{\star\prime}(x) + p_i^\star(x)(q_i'(x) - q_1'(x)) \equiv 0,$$

for all i. Since $q_i(x) - q_1(x)$ is not constant, we obtain

$$p_i^\star(x) = A \exp(q_1(x) - q_i(x)),$$

for some constant A. The only value of A which leaves $p_i^\star(x)$ a rational function is $A = 0$, implying that $p_i(x) \equiv 0$ for all i, contradicting our assumption. ∎

Corollary 6.1 *The function*

$$f(x) = e^{-x^2},$$

differentiates positive polynomials.

Proof: Let $q_i(x), i = 1, \ldots, n$, with linear and higher coefficients positive and which satisfy $q_i(x) - q_j(x)$ is not constant for $i \neq j$, and suppose that

$$\sum_{i=1}^{n} p_i f(q_i(x)) \equiv 0,$$

for all x in some neighbourhood of a point x_0. Let $q_i'(x) = -q_i(x)^2$. Now we have

$$\begin{aligned}\sum_{i=1}^{n} p_i f(q_i(x)) &\equiv \sum_{i=1}^{n} p_i e^{-q_i(x)^2} \\ &\equiv \sum_{i=1}^{n} p_i e^{q_i'(x)} \\ &\equiv 0.\end{aligned}$$

For $i \neq j$, the difference

$$q_i'(x) - q_j'(x) = (q_j(x) - q_i(x))(q_j(x) + q_i(x)),$$

is not constant since $q_i(x) \neq -q_j(x)$ (non constant coefficients are not negative) and $q_i(x) - q_j(x)$ is not constant. We may therefore apply the Proposition 6.2 and conclude that $p_i = 0$ for all i. ∎

Corollary 6.2 *The sigmoid function*

$$f(x) = \frac{1}{1 + e^{-x}}$$

differentiates positive polynomials.

Proof: Let $q_i(x), i = 1, \ldots, n$, be non constant polynomials with linear and higher coefficients not negative and which satisfy $q_i(x) - q_j(x)$ is not constant for $i \neq j$, and suppose that

$$\sum_{i=1}^{n} p_i f(q_i(x)) \equiv 0,$$

for all x in some neighbourhood of a point x_0. Suppose further that the polynomials and numbers p_i represent a smallest contradiction to the assertion (i.e. a contradiction with smallest n). This implies that all $p_i \neq 0$. We have

$$\sum_{i=1}^{n} p_i \prod_{j \neq i} \left(1 + e^{-q_j(x)}\right) \equiv 0. \tag{1}$$

The product can be multiplied out to give

$$\prod_{j \neq i} \left(1 + e^{-q_j(x)}\right) \equiv 1 + \sum_{j \neq i} e^{-q_j(x)} + \sum_{j,j' \neq i} e^{-(q_j(x) + q_{j'}(x))} + \cdots + e^{-\sum_{j \neq i} q_j(x)}.$$

Let $S(q(x))$ denote the sum of the non constant coefficients in the polynomial $q(x)$. Order the polynomials $q_i(x)$ according to $S(q_i(x))$ and let I be the set of i such that $q_i(x)$ is minimal with respect to this ordering. Note that $S(p(x)+q(x)) = S(p(x)) + S(q(x))$ and that $S(p(x)) \neq S(q(x)) \Rightarrow p(x) \not\equiv q(x)$. For $i \in I$ the polynomial

$-\sum_{j\neq i} q_j(x)$ are minimal among the polynomials appearing in the exponents of (1). Hence they are distinct from all others. Note also that for $i \neq i'$ we have

$$-\sum_{j\neq i} q_j(x) - \left(-\sum_{j\neq i'} q_j(x)\right) = q_i(x) - q_{i'}(x),$$

which is not constant. Now regrouping (1) so that terms with the same exponents are grouped together and writing $1 = e^{q(x)}$, where $q(x) \equiv 0$, we have

$$\sum p'_j e^{q'_j(x)} \equiv 0.$$

Note that if two exponents differ by a constant they can be combined into a single term. Hence we ensure that $q'_j(x) - q'_{j'}(x)$ not constant and can therefore apply the proposition. Hence all the coefficients p'_j of the grouped terms are zero. For $i \in I$, the coefficient of the term

$$e^{-\sum_{j\neq i} q_j(x)}$$

is p_i and so $p_i = 0$ a contradiction. ∎

We are now in a position to present our main result characterising discriminability in Symmetry Networks. Its surprising conclusion is that the discriminability of a network with well-behaved real valued activation functions can be completely described by a combinatorial property of the network structure, namely is dismemberment. In other words it provides a finite characterisation of a network, which could be used in practice to determine whether it was adequate to a certain discriminatory task. We will show later that the result also provides a theoretical tool for the analysis of discriminatory power.

Theorem 6.2 *Let $(\mathcal{N}, \Gamma)$ be a Symmetry Network with a single output node and $(\mathcal{N}', \Gamma')$ its dismemberment. Assume further that the activation function is an analytic function which differentiates positive polynomials and that $\mathbf{i}_1$ and $\mathbf{i}_2$ are $0,1$ valued inputs. Then $\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2$ iff there exists $\phi \in \Gamma'$ such that $\mathbf{i}_1'^{\phi} = \mathbf{i}_2'$.*

Proof: ($\Leftarrow$) Let $\phi \in \Gamma'$ such that $\mathbf{i}_1'^{\phi} = \mathbf{i}_2'$. Since ϕ is a weight class preserving automorphism of $\mathcal{N}'$, by Theorem 3.1, the output of $\mathcal{N}'$ is the same under inputs $\mathbf{i}_1'$ and $\mathbf{i}_2'$. But by Proposition 6.1, the output of $\mathcal{N}$ under input $\mathbf{i}_1$ is the same as that of $\mathcal{N}'$ under input $\mathbf{i}_1'$ for any weight assignment, and likewise for $\mathbf{i}_2$. Hence the output of $\mathcal{N}$ is the same for inputs $\mathbf{i}_1$ and $\mathbf{i}_2$ whatever the (allowable) weight assignment. This implies $\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2$, as required.
($\Rightarrow$) In this part of the proof the following definitions will prove useful. We will denote by $W_1, \ldots, W_K$ the partition of the set of non-threshold connections of $\mathcal{N}$ into K weight classes. Let $\mathcal{O}$ be the collection of sets

$$J_{j,\ell} = \{i \in N \cup I | (i,j) \in W_\ell\} \text{ for } \ell = 1, \ldots, K, \text{ and } j \in N.$$

Informally these are the sets of nodes connected to a later node via connections in a single weight class.

The proof is by induction on the number of layers in the network. We will prove a slightly stronger result, namely that the implication holds for all real valued inputs satisfying the following condition. For sets $J_{i,\ell}, J_{i',\ell} \in \mathcal{O}$ (this implies i and i' in the same orbit) with both sets containing input nodes,

$$\sum_{j \in J_{i,\ell}} \mathbf{i}_1(j) = \sum_{j \in J_{i',\ell}} \mathbf{i}_2(j)$$

implies the existence of a bijection π from $J_{i',\ell}$ to $J_{i,\ell}$ such that

$$\mathbf{i}_1(\pi(j)) = \mathbf{i}_2(j) \text{ for } j \in J_{i',\ell}.$$

This condition certainly holds for 0, 1 inputs, since if two subsets of input nodes have the same sum for the two inputs they must both have the same number of 1's.

For the base case we have a single layer network with one output node. In this case $\mathcal{N} = \mathcal{N}'$ and Γ' is the set of all weight class preserving automorphisms. Let $\mathcal{O} = \{J_1, \ldots, J_K\}$ be the orbits of the action of Γ on the set I of non-threshold input nodes. We have that $\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2$, which means that

$$\sum_{\ell \in J_j} \mathbf{i}_1(\ell) = \sum_{\ell \in J_j} \mathbf{i}_2(\ell), \quad j = 1, \ldots, k,$$

since otherwise we could set all the weights connecting the nodes outside the set J_j to zero and ensure that $\mathbf{i}_1$ and $\mathbf{i}_2$ give different outputs. But by the induction hypothesis this means that

$$\exists \pi_j \in \Sigma(J_j) \text{ such that } \mathbf{i}_1(\pi_j(\ell)) = \mathbf{i}_2(\ell) \text{ for } \ell \in J_j, \quad \forall j,$$

and so by combining the π_j we can find a weight class preserving automorphism ϕ of $\mathcal{N}'$ such that $\mathbf{i}_1^\phi = \mathbf{i}_2$.

Now consider a network $\mathcal{N}$ with $l > 1$ layers. We will construct from $\mathcal{N}$ a network $\mathcal{N}_\perp$ with $l-1$ layers. This is done by simply deleting the edges of $\mathcal{N}$ which lie at the end of paths of length l from the output node. Any nodes that are left disconnected are also deleted. The input nodes of $\mathcal{N}_\perp$ are any input nodes of $\mathcal{N}$ which have not been deleted together with those nodes at the end of paths of length $l-1$ from the output node. The group Γ still acts on $\mathcal{N}_\perp$, though its action may no longer be faithful. We may however view $(\mathcal{N}_\perp, \Gamma)$ as a Symmetry Network, with $l-1$ layers. Note also that the set $\mathcal{O}_\perp$ for $\mathcal{N}_\perp$ can be obtained from $\mathcal{O}$ by deleting any sets of nodes that were deleted from $\mathcal{N}$. In order to apply the induction hypothesis we must specify inputs for the additional input nodes created by the deletion of edges of $\mathcal{N}$, to give complete inputs $\mathbf{i}_{\perp 1}$ and $\mathbf{i}_{\perp 2}$ to $\mathcal{N}_\perp$. This will be done by choosing particular weight values for the deleted connections and taking the resulting activations of the nodes which become input nodes in $\mathcal{N}_\perp$ as their input values in $\mathbf{i}_{\perp 1}$ and $\mathbf{i}_{\perp 2}$. The crux of the proof is to show that we can do this in such a way that the following two properties hold:

(i) For each new input nodes i and i' in the same orbit, if there exists a choice of weights for the deleted connections such that $\mathbf{i}_{\perp 1}(i) \neq \mathbf{i}_{\perp 2}(i')$, then the weights are chosen to ensure this inequality.

(ii) For each $J_{i,\ell}, J_{i',\ell} \in \mathcal{O}_\perp$ consisting of new input nodes,

$$\sum_{j \in J_{i,\ell}} \mathbf{i}_{\perp 1}(j) = \sum_{j \in J_{i',\ell}} \mathbf{i}_{\perp 2}(j)$$

implies the existence of a bijection π from $J_{i',\ell}$ to $J_{i,\ell}$ such that

$$\mathbf{i}_{\perp 1}(\pi(j)) = \mathbf{i}_{\perp 2}(j) \text{ for } j \in J_{i',\ell}.$$

Assuming for the moment that this can be done, we can now (thanks to property (ii)) apply the induction hypothesis to the network $\mathcal{N}_\perp$ and obtain a weight class preserving automorphism ϕ of $\mathcal{N}'_\perp$, such that $\mathbf{i}'^{\phi}_{\perp 1} = \mathbf{i}'_{\perp 2}$. The automorphism must stabilise the new input nodes. If node $\mathbf{v}_2 = \phi^{-1}(\mathbf{v}_1)$ then $\mathbf{i}'_{\perp 1}(\mathbf{v}_1) = \mathbf{i}'_{\perp 2}(\mathbf{v}_2)$. Let i_1 (i_2) be the node at the beginning of the path $\mathbf{v}_1$ ($\mathbf{v}_2$). But then by property (i), the activations they received in $\mathcal{N}$ must sum to the same value for each weight class:

$$\sum_{j \in J_{i_1,\ell}} \mathbf{i}_1(j) = \sum_{j \in J_{i_2,\ell}} \mathbf{i}_2(j) \text{ for each } \ell.$$

By property (ii) (assumed in the induction hypothesis for the inputs of $\mathcal{N}$) there exists a bijection π_{i_2} from $J_{i_2,\ell}$ to $J_{i_1,\ell}$ such that

$$\mathbf{i}_1(\pi_{i_2}(j)) = \mathbf{i}_2(j), \text{ for } j \in J_{i_2,\ell}.$$

This means that the automorphism ϕ can be extended to the network $\mathcal{N}'$ via the permutations π_{i_2}. This completes the proof.
It remains to show that in all cases we may choose the weights on the deleted edges in such a way that the new inputs satisfy properties (i) and (ii).
We introduce the following notation. Assume without loss of generality that the weight classes of the deleted edges are $W_1, \ldots, W_L$. For new input node i and weight class W_j, let n^1_{ij} (n^2_{ij}) denote the sum of the input values of $\mathbf{i}_1$ ($\mathbf{i}_2$) on the nodes $J_{i,j}$, for $j = 1, \ldots, L$. If the weight assigned to weight class W_j is w_j then the net input to node i on input $\mathbf{i}_\tau$ is

$$\sum_{j=1}^{L} n^\tau_{ij} w_j, \quad \tau = 1, 2.$$

Clearly if for some pair of nodes i, i' in the same orbit $n^1_{ij} = n^2_{i'j}$ for all j, then no choice of the w_j can make the two sums distinct. The requirement of property (i) is that if this is not the case then the two sums should be made distinct. To this end we define for each j in turn, the following two numbers:

$$\begin{aligned} K_j &= \min_{i,i'}\{|n^1_{ij} - n^2_{i'j}| \,\big|\, n^1_{ij} \neq n^2_{i'j}\} \\ K^j &= \max_{i,i'}\{|n^1_{ij} - n^2_{i'j}| \,\big|\, n^1_{ij} \neq n^2_{i'j}\} \end{aligned}$$

Choose w_j in descending order of their index so that

$$w_j K_j > \sum_{\ell=j+1}^{L} w_\ell K^\ell.$$

In this way if for i, i', any $n^1_{ij} \neq n^2_{i'j}$ then

$$\sum_{j=1}^{L} w_j n^1_{ij} \;\neq\; \sum_{j=1}^{L} w_j n^2_{i'j}. \tag{2}$$

This implies that property (i) holds. The choice of w_j also means that in sufficiently small neighbourhoods of the weight values these inequalities will still hold. Let $J_{k,\ell}, J_{k',\ell} \in \mathcal{O}$ with k and k' in the same orbit and consider the difference between the sums of the activations reaching the nodes k and k' via weight class W_ℓ:

$$\delta = \sum_{i \in J_{k,\ell}} \left[f\left(\sum_{j=1}^{L} n^1_{ij} w_j \right) \right] - \sum_{i \in J_{k',\ell}} \left[f\left(\sum_{j=1}^{L} n^2_{ij} w_j \right) \right].$$

We introduce an indeterminate x and the function:

$$\delta(x) = \sum_{i \in J_{k,\ell}} \left[f\left(\sum_{j=1}^{L} n^1_{ij} (w_j + x^j) \right) \right] - \sum_{i \in J_{k',\ell}} \left[f\left(\sum_{j=1}^{L} n^2_{ij} (w_j + x^j) \right) \right],$$

so that $\delta = \delta(0)$. We will choose weights of the form $\hat{w}_j = w_j + x^j$ for some value of x, sufficiently small to leave $(\hat{w}_1, \ldots, \hat{w}_L)$ in the neighbourhood of $(w_1, \ldots, w_L)$ where inequalities (2) hold for all $i, i' \in J_{k,\ell} \cup J_{k',\ell}$. Hence any such choice of weights would satisfy property (i). Assuming that $\delta(x) \not\equiv 0$ on this neighbourhood and since f is analytic, $\delta(x)$ is an analytic function of x and so has only a finite set of zeros $P_{k,k',\ell}$ in the (closed) neighbourhood. If on the other hand $\delta(x) \equiv 0$ then, since f differentiates positive polynomials (and $n^k_{ij} > 0$), we conclude that there is a bijection π from $J_{k',\ell}$ to $J_{k,\ell}$ such that $n^1_{\pi(i)j} = n^2_{ij}$ for all weight classes W_j and all $i \in J_{k',\ell}$ implying that no choice of weights w_j would make δ non zero. In this case the two inputs are both composed of the same set of values – the correspondence being given by π:

$$f\left(\sum_{j=1}^{L} n^1_{\pi(i)j} w_j \right) = f\left(\sum_{j=1}^{L} n^2_{ij} w_j \right).$$

Since for each pair $J_{k,\ell}, J_{k',\ell}$ for which $\delta(x) \not\equiv 0$, there are only finitely many choices of x which must be disallowed (the zeros $P_{k,k',\ell}$), and there are only finitely many such pairs of sets in $\mathcal{O}$, we may choose a value of x in the given interval which forces all sums distinct unless they are composed of the same set of values. This is precisely the requirement of condition (ii). ∎

It is natural to ask how difficult it might be to find a weight assignment for which the outputs are different in the case where $\mathbf{i}_1 \not\cong_{\mathcal{N}} \mathbf{i}_2$. The following proposition shows that in this case the weight vectors which fail to give an inequality occupy no volume in weight space. Hence a random weight assignment will deliver the required inequality with overwhelming probability.

Proposition 6.3 *Suppose $\mathbf{i}_1 \not\cong_{\mathcal{N}} \mathbf{i}_2$ for some Symmetry Network $(\mathcal{N}, \Gamma)$. There is no open neighbourhood in weight space where*

$$F_{\mathcal{N}}(w, \mathbf{i}_1) = F_{\mathcal{N}}(w, \mathbf{i}_2).$$

Proof: Suppose there is a weight assignment w_0 with an open neighbourhood where equality holds. By the fact that $\mathcal{N}$ distinguishes $\mathbf{i}_1$ and $\mathbf{i}_2$ there exists an assignment w_1 for which equality does not hold. Consider the function

$$g(t) = F_{\mathcal{N}}(w(t), \mathbf{i}_1) - F_{\mathcal{N}}(w(t), \mathbf{i}_2), \text{ where } w(t) = (1-t)w_0 + tw_1.$$

This function is zero for points in a neighbourhood of $t = 0$ but is non zero at $t = 1$. But $g(t)$ is an analytic function on a closed interval which is not everywhere zero. Hence it can have only finitely many zeros, a contradiction. ■

Corollary 6.3 *If an analytic activation function is chosen which differentiates positive polynomials, the Symmetry Networks $(\mathcal{N}, \Gamma)$ of Example 6.1 have full disciminability, that is for all binary input vectors $\mathbf{i}_1$, $\mathbf{i}_2$*

$$\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2 \iff \mathbf{i}_1 = \mathbf{i}_2^{\gamma} \quad \textit{for some } \gamma \in \Gamma.$$

Proof: By the observation at the beginning of Section 6 we need only show that $\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2$ implies the existence of $\gamma \in \Gamma$ satisfying $\mathbf{i}_1 = \mathbf{i}_2^{\gamma}$. Using Theorem 6.2 we have that $\mathbf{i}_1 \cong_{\mathcal{N}} \mathbf{i}_2$ implies the existence of $\gamma' \in \Gamma'$ such that $\mathbf{i}_1'^{\gamma'} = \mathbf{i}_2'$, where $\mathcal{N}'$ is the dismemberment of $\mathcal{N}$ and Γ' is the group of symmetries of $\mathcal{N}'$ which preserve the inherited weight classes (recall that the weight class of connection (i, ϕ) is indexed by $\phi(i)$). Consider a hidden node $(\psi, o) \in \mathcal{N}'$ whose image is (ψ', o) under the action of γ'. As γ' preserves the weight classes, for each i it maps node $(i, \psi, o) \in \mathcal{N}'$ to $(\psi'^{-1}\psi(i), \psi', o)$ (o is the output node of the network $\mathcal{N}$). Hence

$$\mathbf{i}_2'(\psi'^{-1}\psi(i), \psi', o) = \mathbf{i}_1'(i, \psi, o),$$

but this implies that

$$\mathbf{i}_1(i) = \mathbf{i}_2(\psi'^{-1}\psi(i)),$$

for all i. Now consider the permutation $\tau = \psi'^{-1}\psi \in \Gamma$. By the above observation we have that $\mathbf{i}_1^{\tau} = \mathbf{i}_2$. ■

7 Application

In this section we consider an application of the results of the previous section to the design of networks for discriminating non-isomorphic graphs. We first introduce a particular network. We will then show that the constructed network will distinguish non-isomorphic graphs if and only if the graph reconstruction conjecture holds.

Example 7.1 *Consider the Graph Recognition Symmetry Network $\mathcal{N}_n$ with nodes indexed by all the subsets of $U_n = \{1, \ldots, n\}$ with two or more elements, together with one threshold node. The inputs I are the two element sets and the threshold node, while the output is the set U_n itself. Two nodes (subsets) u and v are connected with an edge in the network if $u \subseteq v$ and v has one more element than u. The weight class depends only on the number of elements in u. There are also edges from the threshold node to all computational nodes. The symmetric group S_n acts in the natural way on the subsets of U_n.*

This network has exponentially many nodes since there is a node for each non-empty non-singleton subset of $\{1, \ldots, n\}$ together with a threshold node, that is a total of $2^n - n$ nodes. It should be noted however that there are only $2(n-2)$ parameters, two for each non-input layer. We will show that provided the graph reconstruction conjecture holds, then this network can distinguish non-isomorphic graphs. This result though a pleasing characterisation of the discriminability of the network, does not bring us any closer to a probabilistic polynomial algorithm for graph isomorphism (see Section 5) in view of its exponential size.
Before stating the proposition we give the definition of the graph reconstruction conjecture. For a graph $G \in \mathcal{G}_n$ we term the collection of vertex deleted subgraphs $G_i = G \setminus i$ of G, $i = 1, \ldots, n$, the *deck* of G, where $G \setminus i$ is the graph obtained from G by deleting vertex i and any edges containing i. Clearly the two graphs on two vertices both have the same deck. The *graph reconstruction conjecture* states that these two graphs are the only non-isomorphic graphs with the same deck. The conjecture has remained unsolved for a number of years though there is circumstantial evidence to suggest that it is valid [2, 4].

Proposition 7.1 *The network $\mathcal{N}_n$ can distinguish non-isomorphic graphs in $\mathcal{G}_n$ for all n if and only if the graph reconstruction conjecture holds.*

Proof: ($\Leftarrow$) Proof by contradiction. Let $\mathcal{N}_n$ be a minimal network such that there exists $G_1 \not\cong G_2$ with $G_1 \cong_{\mathcal{N}_n} G_2$. By Theorem 6.2, there exists $\phi \in \mathsf{Aut}(\mathcal{N}')$ with $\mathbf{i}'^{\phi}_{G_1} = \mathbf{i}'_{G_2}$, where $\mathcal{N}'$ is the dismemberment of $\mathcal{N}_n$. For $i = 1, \ldots, n$, there are n subnetworks $\mathcal{N}'^i$ of $\mathcal{N}'$ corresponding to including only nodes indexed by paths which contain the set $U_n \setminus \{i\}$ together with the output and threshold node. Note that the subnetworks form a partition of the non-output nodes of $\mathcal{N}'$, and because of the connectivity of $\mathcal{N}_n$, the element i does not occur in any of the sets of the node paths of $\mathcal{N}'^i$ except in the set at the end of each path. The automorphism ϕ acts as a permutation among these subnetworks. Let $\psi \in S_n$ such that $\phi(\mathcal{N}'^j) = \mathcal{N}'^{\psi(j)}$. Let $\mathcal{N}^i$, $i = 1, \ldots, n$, be the network obtained from $\mathcal{N}_n$ by deleting all nodes which contain the vertex i. Hence each $\mathcal{N}^i$ is a copy of $\mathcal{N}_{n-1}$. Clearly the dismemberment of $\mathcal{N}^i$ is the network $\mathcal{N}'^i$. Let $G_1 \setminus i$ be the vertex i deleted subgraph of G_1. This graph induces an input to $\mathcal{N}^i$ whose dismemberment is the same as the dismemberment of the input $\mathbf{i}_{G_1}$ restricted to the network $\mathcal{N}'^i$. Consider two inputs to the network $\mathcal{N}'^{\psi(i)}$. The first is $\mathbf{i}'_{\psi(G_1 \setminus i)}$, while the second is $\mathbf{i}'_{G_2 \setminus \psi(i)}$. By acting on U_n the automorphism ψ acts as an automorphism of $\mathcal{N}'$ which takes $\mathcal{N}'^i$ to $\mathcal{N}'^{\psi(i)}$. Hence the map $\phi\psi^{-1}$ is an automorphism of $\mathcal{N}'$, which maps $\mathcal{N}'^{\psi(i)}$ to itself and maps the first input given above to the second. We conclude that

$$\psi(G_1 \setminus i) \cong_{\mathcal{N}^{\psi(i)}} G_2 \setminus \psi(i).$$

But by the minimality of the network $\mathcal{N}_n$ we can conclude that

$$\begin{aligned} \psi(G_1 \setminus i) &\cong G_2 \setminus \psi(i), \\ \text{and therefore} \quad G_1 \setminus i &\cong G_2 \setminus \psi(i). \end{aligned}$$

This tells us that the vertex deleted subgraphs are pairwise isomorphic, while the graphs themselves are not, thus contradicting the graph reconstruction conjecture. ($\Rightarrow$) Suppose we have a pair of graphs $G \not\cong H$, which contradict the graph reconstruction conjecture. Let G_i be the vertex deleted subgraph $G \setminus i$. Similarly H_i, and assume without loss of generality that $G_i \cong H_i$. Consider the network $\mathcal{N}_n$, and the networks $\mathcal{N}^i$ and $\mathcal{N}'^i$ as above. Since $H_i \cong_{\mathcal{N}^i} G_i$, there exists an automorphism $\alpha_i \in \mathsf{Aut}(\mathcal{N}'^i)$ with ${}^{\alpha_i}\mathbf{i}_{H_i} = \mathbf{i}_{G_i}$. Since the networks $\mathcal{N}'^i$ partition $\mathcal{N}'$, we can combine the α_i to form an automorphism α of $\mathcal{N}'$ with $\mathbf{i}_H^\alpha = \mathbf{i}_G$. Hence we have $H \cong_{\mathcal{N}_n} G$ and so $\mathcal{N}_n$ cannot distinguish all non-isomorphic graphs in $\mathcal{G}_n$. ∎

8 Conclusions

We have described a general method of introducing symmetries into feedforward networks in such a way that the output is guaranteed to be invariant under corresponding transformations of the input. This has been shown to greatly reduce the complexity of the training of such networks and so may result in a widening of the scope of applicability of feedforward Neural Networks.

We have considered applying the techniques to the problem of graph recognition. This has led to the investigation of which inputs a given Symmetry Network can distinguish. We have been able to charaterise precisely when a network can distinguish two inputs provided reasonable restrictions are placed on the activation functions. This result has allowed us to say something about minimal networks for distinguishing pairs of inputs as well as to show that a particular network can distinguish non-isomorphic graphs if and only if the graph reconstruction conjecture holds.

There are many questions which remain unresolved. Perhaps the most immediate is which activation functions differentiate polynomials and whether our conjecture that the sigmoid function satisfies this condition is true. From the point of view of the graph isomorphism problem we might ask which networks can distinguish non-isomorphic graphs. Will such networks always require exponentially many nodes? While from the point of view of our original aim of finding a probabilitistic algorithm for graph isomorphism, the question we must ask is whether we can design polynomially sized networks which with non-zero probability distinguish any given pair of graphs.
More generally we might ask whether there are any inequalities which hold between the size of the automorphism group of a Symmetry Network and the size of the network required to distinguish all equivalence classes of inputs.

References

[1] Laszlo Babai, Paul Erdös and Stanley M. Selkow, Random Graph Isomorphism, SIAM J. Comput., 9 (3) (1980) 628–635.

[2] J.A. Bondy and R.L. Hemminger, Graph Reconstruction – A Survey, J. Graph Theory, 1 (1977) pp. 227–268.

[3] Nigel Dodd, Graph Recognition Strategies, RIPR RSRE, Malvern, UK, Report No. RIPREP/1000/28/88, 1988.

[4] W.L. Kocay, Hypomorphisms, Orbits and Reconstruction, J. Combin. Theory Ser. B, 44 (1988) pp. 187–200.

[5] K. Lang and G.E. Hinton, The development of TDNN architecture for speech recognition, Technical Report CMU-CS-88-152, Carnegie-Mellon University, 1988.

[6] Y. le Cun, A theoretical framework for back propagation, in D. Touretzsky, editor, *Connectionist Models: A Summer School*, Morgan-Kaufmann, 1988.

[7] Eugene M. Luks, Isomorphism of graphs of bounded Valence can be tested in polynomial time, J. Comp. Sys. Sci., 25 (1982) pp. 42–65.

[8] Ofer Matan, Christopher J.C. Burges, Yann Le Cun and John S. Denker, Multi-Digit Recognition Using A Space Displacement Neural Network, Proceedings of NIPS-4, 1991, Denver, USA, pp. 488–495.

[9] J.L. McClelland and D. Rumelhart, *Parallel Distributed Processing, Vol 1.* MIT Press, Cambridge, USA, 1986.

[10] M. Minsky and S. Papert, Perceptrons, expanded edition, MIT Press, Cambridge, USA, 1988.

[11] M. M. Moya, R. J. Fogler and L. D. Hostetler, Back-propagation for Perspective-invariant Pattern Recognition in SAR Imagery, Abstracts of 1st Meeting of INNS, 1988.

[12] Michael O. Rabin, Probabilistic Algorithm for Testing Primality, J. of Number Theory, 12 (1980) 128–138.

[13] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, Learning representations by back-propagating errors, Nature, 323 (1986) pp. 533–536.

[14] John Shawe-Taylor, Building Symmetries into Feedforward Networks, Proceedings of First IEE Conference on Artificial Neural Networks, London, (1989) pp. 158–162.

[15] John Shawe-Taylor and David Cohen, The Linear Programming Algorithm, Neural Networks (in press).

[16] Patrice Simard, Bernard Victorri, Yann Le Cun and John Denker, Tangent Prop – A formalism for specifying selected invariances in an adaptive network, Proceedings of NIPS-4, 1991, Denver, USA, pp. 895–903.