# Integrating Causal Learning Rules with Backpropagation in PDS Networks

**Ronald A. Sumida**
Artificial Intelligence Laboratory
Computer Science Department
University of California
Los Angeles, CA, 90024
sumida@cs.ucla.edu

## Abstract

This paper presents a method for training PDP networks that, unlike backpropagation, does not require excessive amounts of training data or massive amounts of training time to generate appropriate generalizations. The method that we present uses general conceptual knowledge about cause-and-effect relationships within a single training instance to constrain the number of possible generalizations. We describe how this approach has been previously implemented in rule-based systems and we present a method for implementing the rules within the framework of Parallel Distributed Semantic (PDS) Networks, which use multiple PDP networks structured in the form of a semantic network. Integrating rules about causality with backprop in PDS Networks retains the advantages of PDP, while avoiding the problems of enormous numbers of training instances and excessive amounts of training time.

## Introduction

Parallel Distributed Processing (PDP) models [Rumelhart and McClelland, 1986] have demonstrated many desirable characteristics for learning and generalization. Specifically, PDP models: (1) use a simple learning mechanism such as backpropagation that merely modifies link weight values for each training pattern, (2) automatically generalize as a result of the learning process by averaging the training patterns so that the network can correctly respond to new inputs, (3) use the generalizations that are created to perform pattern completion from partial or noisy inputs, (4) naturally account for interference effects since similar concepts share similar representations and (5) are robust against noise and damage to the network.

Backpropagation performs extremely well and provides the advantages listed above when the size of the training set and the number of input features is relatively small. However, there are serious problems with applying backprop to store the large amounts of knowledge needed for higher-level cognitive tasks, such as natural language. Since backprop learns by similarity-based generalization, it must be shown enough training instances so that all relevant features are correlated while all other features are not. As the number of input features increases with the complexity of the problem domain, the number of training examples needed becomes enormous. For example, suppose that we would like to teach the network about the concept of strength, given a number of instances of people successfully and unsuccessfully lifting a heavy object[1]. Two such instances are presented below:

> John, a tall man with short brown hair and a dark complexion successfully lifts a heavy object in the dining room.
>
> Mary, a small child with long blond hair and a light complexion is unsuccessful at lifting the same heavy object in the living room.

Every feature that is present when John lifts the object and every feature that is absent when Mary fails to lift it are possible causes for the different result. Thus, from the data given above, it is possible to conclude that age, height, hair length, hair color, complexion, location or even the person's name are responsible for John's success at lifting

[1] This example is based on one provided in [Pazzani, 1988].

the object and Mary's failure. In order for the network to correctly learn that strength is correlated with age and not with any of the other features, examples of people with all different types of heights, hair lengths, hair colors and complexions in all types of environments must be presented to the network. Assuming that there are 3 values for height (short, medium and tall), 2 for hair length (short and long), 4 for hair color (blond, brown, black, red), 2 for complexion (light and dark), 10 different names, and 10 different locations, 4800 (= 3x2x4x2x10x10) different training instances are needed for the network to only correlate age with strength. If we were to scale the system up further and add more features such as eye color or more values to the features listed above (e.g. adding grey to the hair color list), the number of necessary training instances could easily exceed 100,000. As the number of training instances grows to such unreasonable numbers, the training time becomes so enormous that it is practically impossible to train the network.

One possible solution is to train the network on only a subset of the patterns. The problem with this approach is that the network will find illusory correlations and form improper or very complicated generalizations. For example, if the network is not shown that a specific conjunction of features is not relevant, it may conclude that short brown hair and a dark complexion are correlated with strength.

This paper presents a method for training PDP networks that does not require excessive amounts of training data to generate appropriate generalizations. The method that we present uses information about cause-and-effect relationships within a single training instance to constrain the number of possible generalizations. The following section shows how this method has previously been implemented in rule-based systems. This is followed by a discussion that indicates how the rules are implemented within the PDS Network [Sumida and Dyer, 1989] framework.

## Symbolic Approaches to Learning and Generalization

Considerable work has been done on symbolic, rule-based approaches to learning and generalization, examples of which include [Mitchell *et al.*, 1988, Lebowitz, 1990, Pazzani, 1988]. Previous learning systems vary in the amount of prior knowledge that they apply during the learning process. They range from similarity-based learning (SBL) systems, which operate in much the same way (and share the same problems) as backprop and which assume no prior knowledge, to explanation-based generalization (EBL) systems which assume enough prior knowledge so that the system can construct an entire explanation for why an event occured. Our focus here is on the intermediate position between the two extremes, where the only prior knowledge that we assume are rules about causality. [Pazzani, 1988] refers to this form of generalization as theory-driven learning (TDL) and discusses TDL (along with SBL and EBL) in the program OCCAM. In OCCAM, three types of causal generalization rules are used: (1) *exceptionless*, which apply when similar actions yield the same result, (2) *dispositional*, which apply when similar actions yield differing results and focus on the differing features of the actor or object of the action to explain the differing result, and (3) *historical*, which (like the dispositional rules) apply when similar actions yield differing results, but assume that the events that precede the action explain the differing result. In this paper, our focus will be devoted to discussing how dispositional generalization rules are employed.

The first step in employing a dispositional generalization rule is to create a generalized event that represents all the shared features of the set of events with the same result. The generalized event is then matched with the dispositional generalization rule to determine what role of the cause is responsible for the differing effect. If the same feature of that role occurs in two events that have different results, then it could not be responsible for the differing result so it is no longer considered. Of the remaining features, the one that has been most successful at accounting for different results in previous similar situations (i.e., previous situations involving the same act and role) is selected. If there is no reason to prefer one feature over another, then one is selected randomly. The feature that is hypothesized to be responsible is then added to the generalized event. As new inputs are presented, the hypothesis will either be confirmed, in which case a confidence measure associated with the hypothesis will be increased, or it will be refuted, in which case a new feature will need to hypothesized as responsible for the different result.

As an example of the procedure described above, consider the example from the first section. When the events of John successfully lifting the heavy object and Mary failing to lift it are

presented, the system applies the following dispositional generalization rule: *If similar actions performed on an object have different results, and they are performed by different actors, the differing features of the actor are responsible for the different results.* The rule indicates that one of the features of the actor is responsible for the difference in their ability to lift the weight. At this point, there is no reason to prefer one attribute over another so a feature is selected at random, for example, hair color (in this case brown). An example of a blond-haired person lifting the weight is then presented, and the system notices that the prediction that brown hair is responsible has been contradicted. The confidence measure associated with the hair color hypothesis is very low, so the system rejects the hypothesis and selects another feature at random as being responsible for the difference. As other features are selected and refuted, the system will quickly select age as being responsible. As further events are encountered that substantiate this hypothesis, its confidence measure grows and the strength *disposition* is created. The disposition is associated with the action (PROPEL) and the role (ACTOR). Thus, when a similar set of events is encountered in the future, such John being able to remove a tightly attached lid from a jar and Mary not being able to, the system will prefer the strength dispositional attribute to features such as hair color and eye color.

## Integrating Backpropagation with Causal Learning Rules

In order to implement the above rules with PDP, we first need to represent the information provided in the training instances. In previous papers, [Sumida and Dyer, 1989, Sumida and Dyer, 1991, Sumida, 1991] we showed how Parallel Distibuted Semantic (PDS) Networks store high-level knowledge. The following section describes how PDS Networks store information using only backprop. This is followed by a description of how causal learning rules are integrated with backprop to avoid the problems with traditional PDP approaches.

### PDS Networks

The PDS Network approach is to store all knowledge over multiple PDP networks using backprop, with each network representing a class of concepts and with related networks connected in the general manner of a semantic net. For example, the network shown in Figure 1 encodes acts that involve an application of force (PROPEL) and has roles for the actor, object and result. The network is connected to other PDP networks, such as HUMAN, PHYS-OBJ and OUTCOME, that store information about humans, physical objects, and outcomes of events. Each network functions as a type of *encoder net*, where: (1) the input and output layers have the same number of units and are presented with exactly the same pattern, (2) the weights of the network are modified so that the input pattern will recreate itself as output, and (3) the resulting hidden unit pattern represents a *reduced description* of the input. In the networks that we use, a single set of units is used for both the input and output layers. The net can thus be viewed as an encoder with the output layer folded back onto the input layer and with two sets of connections: one from the single input/output layer to the hidden layer, and one from the hidden layer back to the i/o layer. In Figure 1 for example, the actor, object, and result role-groups collectively constitute the input/output layer, and the PROPEL ensemble constitutes the hidden layer.

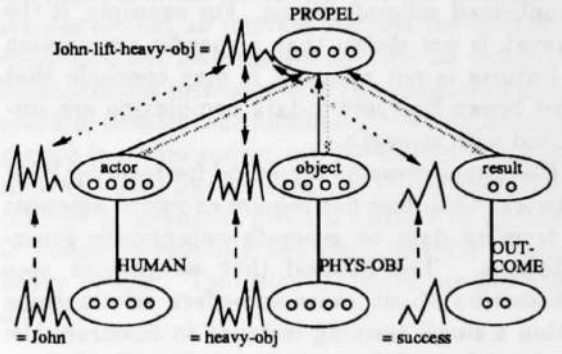


Figure 1: The network that stores information about acts involving an application of force, in this case John-lift-heavy-object. The black arrows represent links from the input layer to the hidden layer and the grey arrows indicate links from the hidden layer to the output layer. The thick lines represent links between networks that propagate a pattern without changing it.

Knowledge is stored in a network by teaching it to encode the items in its training set. For each item, the patterns that represent the features of the item are presented to the input role groups, and the weights are modified using backpropagation so that the patterns recreate themselves as

output. For example, in Figure 1, the pattern for John successfully lifting the heavy object is presented to the PROPEL network by propagating the John pattern[2] from the HUMAN network to the actor role, the pattern for the heavy object from the PHYS-OBJ network to the object role, and the success pattern from the OUTCOME network to the result role. The PROPEL network is then trained on this pattern by modifying the weights between the input/output role groups and the PROPEL hidden units so that the John-lift-heavy-object pattern recreates itself as output. The network automatically generalizes since the hidden units: (1) become sensitive to common features of the training patterns and (2) classify a new concept that was not seen during training based on its similarity to familiar concepts.

## Implementing Causal Learning Rules in PDS Networks

In order to implement causal learning rules within the PDS Network framework, we need to: (1) accomplish the same result as backprop, that is, modifying link weights so that a hidden unit becomes responsible for recognizing significant correlations in the input as in [Hinton, 1986], but (2) use a a theory-driven learning algorithm rather than one based upon SBL. We therefore need to implement the equivalent of the structure/hypothesis building and rule matching operations by using weight modifications within the network. The idea is to modify the weights so that a particular hidden unit represents the current hypothesis and correlates the pattern for the significant feature (i.e., the feature that is hypothesized to be responsible for the different result) with the pattern for the result. For example, the hypothesis that age is responsible for John's ability to lift the object and for Mary's inability to do so is represented by a hidden unit that is assigned to correlate the pattern for age with the pattern for success (Figure 2).

The following steps are used to apply a dispositional generalization rule and to generate an appropriate hypothesis in PDS Networks. Recall that the first step in symbolic systems is to create a generalized event that contains the shared features of those events with the same result. The

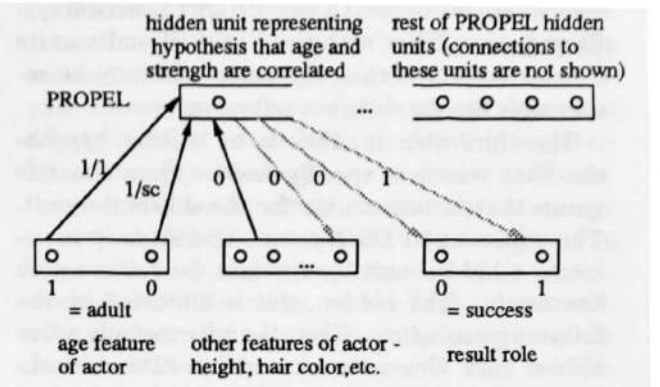


Figure 2: The connections between the hidden unit (that represents the hypothesis that the age feature is responsible for the difference in ability to lift a heavy object) and the actor and result role groups. The dark arrows indicate the connection *to* the hidden unit and the grey arrows indicate the connection *from* the hidden unit. The numbers to the left of each arrow indicate the weight (sc = small constant). All connections other than the ones to the age and result units have a weight value of 0.

equivalent step is accomplished in PDS Networks by clamping the pattern for success or failure (depending on the result of the new event) over the result units and letting the network settle into a stable configuration. Since the network's knowledge of previous events is stored using backprop, the resultant pattern represents the shared features of events with the same result. The next step in symbolic systems is to match the generalized event with the dispositional generalization rule to determine what role is responsible for the different result. This step is accomplished using a mechanism similar to a Propagation Filter [Sumida and Dyer, 1991, Sumida, 1991].

Propagation Filters use the pattern over a selector group of units to determine which of a number of filter groups to enable. Each filter group: (1) gates the connection from a group of source units to a group of destination units, (2) is sensitive to a particular pattern over the selector, and (3) allows the pattern over the source to be propagated to the destination when the particular pattern occurs over the selector. We apply a mechanism similar to Propagation Filters since the pattern for the generalized event acts as a selector. However, rather than have the selector open up a particular group of units, it merely indicates which role group is potentially responsible for the different

[2]The John pattern represents a reduced description of John's features, such as age, height, hair-color, etc. The procedure for obtaining the John pattern is the same as that described here for the John-lift-heavy-object pattern.

result. For example, in the PROPEL network, a Propagation Filter with the PROPEL units as its selector indicates that the actor role may be responsible for the different outcome.

The third step in TDL is to build a hypothesis that selects a specific feature from the role group that is responsible for the different result. The equivalent PDS Network operation is to allocate a hidden unit to correlate the feature with the result. The hidden unit is allocated by the following procedure: First, the system finds a free hidden unit. Concepts are stored in PDS networks by training an individual network so that patterns recreate themselves as output. The network only uses as many hidden units as is necessary for learning the training data (i.e., if there are additional hidden units, then backprop leaves them unused). A free hidden unit is chosen from among the unused units. Note the resemblance between the algorithm we are using and a destructive learning algorithm. In a destructive learning algorithm, the system starts with a large number of hidden units and progressively deletes the ones that aren't used. Instead of deleting the unused hidden units, we are using them to represent a hypothesis for which feature is responsible for the different result.

If the connection to the selected hidden unit is from a unit that represents an *irrelevant* feature, its weight is set to 0. Thus, the hidden unit will be unaffected by the values of the irrelevant features. If a unit represents a *relevant* feature, then we would like to have that unit send the hidden unit a value of 1[3]. Thus, we set the weight from the feature to the hidden unit to be 1/(component of the pattern that represents the feature). If the component of the pattern includes a 0, then the weight is set to 1/(a very small constant) since 1/0 is undefined. For example, if age is a relevant feature for success in lifting the heavy object and adult is represented by the pattern "1 0", then the weight from the first unit of age to the hidden unit is set to 1/1 or 1, and the weight from the second unit of age is set to 1/(a very small constant) (see the weights on the links in Figure 2 for an example). The thresold for the hidden unit is set equal to the number of units that represent relevant features. For example, since the pattern for adult is represented over 2 units, the threshold is set to 2. The hidden unit only responds when its activation value is near threshold, not when it is too far above or below. This assures that the unit will only be active when the proper pattern for the relevant feature occurs. Thus, when the "1 0" pattern is encountered over the age units, the hidden unit will be turned on. We now need to have the hidden unit correlate the relevant features with the result. Thus, when the hidden unit is active, we need it to cause the pattern for the correct result to occur. We set the weight from the hidden unit to each result unit to be the value that is expected for that unit. For example, if we expect the result to be success, and success is represented by the pattern "0 1", then the weight to the first success unit is 0 and the weight to the second result unit is 1 (again see Figure 2).

To illustrate the above procedure, consider again the example from the first section. When we show the system the patterns for John successfully lifting the object and Mary failing to lift it, the system notices that the dispositional generalization rule from the second section (in our discussion of symbolic approaches) is appropriate since the pattern for the object role is the same in both events, while the patterns for the actor and the result are different. The mechanism similar to a Propagation Filter suggests that the actor role is responsible for the different result. Since there is no reason to prefer one feature over another, hair color is selected at random. The system now selects one of the free hidden units to represent the hypothesis that hair color is responsible for the different result. Since all other features besides hair color are hypothesized to be irrelevant, the weights from the units representing all features besides hair color are set to 0 (Figure 3). The pattern for brown hair is "1 1 0", so the weight from the first hair color unit to the hidden unit is set to 1/1, the weight from the second hair color unit is set to 1/1, and the weight from the third unit is set to 1/(a very small constant). The threshold for the hidden unit is set to 3, since there are three units for the relevant feature of hair color. The pattern for success is "0 1", so to correlate brown hair with success at lifting the weight, the weight from the hidden unit to the first result unit is set to 0, and the weight to the second hidden unit is set to 1.

An example of a blond-haired person lifting the weight is then presented, which contradicts our hypothesis so another feature is hypothesized to be responsible for the different result and the weights to the hidden unit are changed so that the new feature is correlated with success. As with the symbolic approach, after a small number of examples, the system refutes the hypotheses involving

[3]For the sake of simplicity, we choose the value 1. In reality we can choose a different constant and merely adjust the threshold appropriately.

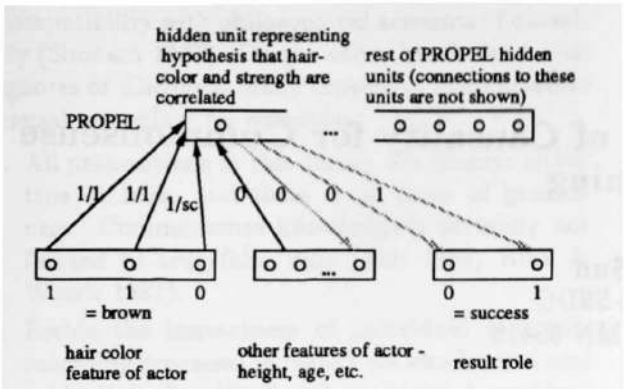


Figure 3: The connections between the hidden unit (that represents the hypothesis that hair color is responsible for the difference in the ability to lift the heavy object) and the age and result units.

irrelevant features and realizes that age is the important feature. The network configuration shown in Figure 2 is therefore the hypothesis that the system builds.

## Related Work

Recent work has been done in combining explanation-based learning with neural network approaches, for instance [Shavlik and Towell, 1989, Katz, 1989]. Some of this recent work is similar to the work presented in this paper. For example, some of the rules discussed in [Shavlik and Towell, 1989] bear a resemblance to the rules that we have presented here. However, these systems have not yet provided a neural network framework for representing the type of high-level knowledge that is needed for generating complex explanation chains. We believe that PDS Networks will provide the framework that is necessary for representing complicated knowledge structures. Thus, we have chosen to integrate rules about causality with backpropagation in the PDS Network framework.

## Conclusions

In this paper, we have presented a method for training PDP networks that integrates knowledge about cause-and-effect relationships with backpropagation. This approach has a number of important advantages over PDP systems that only use backpropagation: (1) It does not require enormous amounts of training data since rules about causality within a single training instance are used to constrain the number of possible generalizations. In contrast, similarity-based generalization methods such as backpropagation need to compare enormous numbers of instances to determine which features are relevant in forming a generalization. (2) Training time is dramatically decreased because far fewer training instances are examined. (3) The integration of causal learning rules with backprop is implemented in the framework of PDS Networks, so the high-level knowledge necessary for natural language processing can be represented, the advantages of PDP are retained, and the problems encountered in training PDP networks are avoided.

## References

G. E. Hinton. 1986. Learning Distributed Representations of Concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA.

B. F. Katz. 1989. Integrated Learning in a Neural Network. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY.

M. Lebowitz. 1990. Utility of Similarity-Based Learning in a World Needing Explanation. In *Machine Learning, Vol. 3*, Morgan Kaufmann, Los Altos, CA.

Mitchell, T., Kedar-Cabelli, S. and Keller, R. 1988. Explanation-based Learning: A Unifying View. *Machine Learning*, Vol. 1(1).

M. J. Pazzani. 1988. Learning Causal Relationships: An Integration of Empirical and Explanation-Based Learning Methods. UCLA Artificial Intelligence Laboratory Technical Report UCLA-AI-88-10 (PhD Dissertation).

D. E. Rumelhart and J. L. McClelland. 1986. *Parallel Distributed Processing*, Volume 1. MIT Press, Cambridge, Massachusetts.

J. W. Shavlik and G. G. Towell. 1989. Combining Explanation-Based Learning and Artificial Neural Networks. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY.

R. A. Sumida and M. G. Dyer. 1989. Storing and Generalizing Multiple Instances while Maintaining Knowledge-Level Parallelism. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI.

R. A. Sumida. 1991. Dynamic Inferencing in Parallel Distributed Semantic Networks. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, IL.

R. A. Sumida and M. G. Dyer. 1991. Propagation Filters in PDS Networks for Sequencing and Ambiguity Resolution. In *Advances in Neural Information Processing 4 (NIPS-91)*, Denver, CO.