Developing Microfeatures by Analogy

Eric R. Melz
Department of Psychology
University of California, Los Angeles, CA 90024
emelz@cognet.ucla.edu

Abstract[*]

A technique is described whereby the output of ACME, a localist constraint satisfaction model of analogical mapping (Holyoak & Thagard, 1989) is used to constrain the distributed representations of domain objects developed by Hinton's (1986) multilayer model of propositional learning. In a series of computational experiments, the ability of Hinton's network to transfer knowledge from a source domain to a target domain is systematically examined by first training the model on the full set of propositions representing a source domain together with a subset of propositions representing an isomorphic target domain, and then testing the network on the untrained target propositions. Without additional constraints, basic gradient descent can recover only a negligible proportion of the untrained propositions. Comparison of simulation results using various combinations of the distributed mapping technique and weight decay, indicate that general purpose network optimization techniques may go some ways towards improving the transfer performance of distributed network models. However, performance can be improved substantially more when optimization techniques are combined with the distributed representation mapping technique.

Introduction

Theoretical accounts of analogy posit at least two central stages in the process of analogizing: *mapping*, the establishment of systematic correspondences between objects and relations of a source domain and a target domain, and *transfer*, the importation of knowledge from the source domain into the target domain based on the correspondences established by the mapping phase. Numerous models of analogical transfer have been proposed, each of which explicitly implements the mapping and transfer phases, usually employing traditional techniques of symbolic processing (Hall, 1989; Falkenhainer, Forbus & Gentner, 1989; Holyoak, Novick & Melz, 1992). With the advent of distributed connectionist models, there appears to be some promise for eliminating the cumbersome symbolic machinery of traditional models of analogy. Using a single general purpose learning technique such as backpropagation (Rumelhart, Hinton & Williams, 1986), it may be possible to simply train a model on a set of propositions representing some domain of knowledge, and exploit the generalization capabilities of the network in order to generate useful inferences based on the knowledge the network has obtained.

This paper explores the ability of supervised gradient descent learning to (a) form representations of correspondences between two domains, and (b) use these representations to perform analogical transfer. I first demonstrate that without additional constraints on the learning procedure, gradient descent does not form representations that are optimal for the task of analogical transfer. I then examine two mechanisms which can induce the network to form optimal representations: (1) simple weight decay, and (2) "programming" the network with correspondenceds derived from a connectionist model of analogical mapping. In the next section, I describe the network and domain with which the rest of this paper is conerned.

Hinton's (1986) Family Tree Problem

Hinton (1986) describes a network that learns relationships between people in two isomorphic family trees. The family trees, shown in Fig. 1 can be represented by two sets of 52 propositions of the form *obj1 rel obj2*, where *obj1* and *obj2* are two relatives, and *rel* is the relationship between *obj1* and *obj2* (which may be one of the following: *has_husband*, *has_wife*, *has_son*, *has_daughter*, *has_mother*, *has_father*, *has_brother*, *has_sister*, *has_uncle*, *has_aunt*, *has_nephew*, and *has_niece*). For example, the fact that Christopher is married to Penelope is represented by the proposition *Christopher has_wife Penelope*. The family tree which contains the English people will henceforth be referred to as the source domain, and the Italian family tree will be referred to as the target domain.

The network which learns the relationships is shown in Fig. 2. The input layer is composed of localist representations of the 24 domain objects (12 English people, and 12 Italians) that can fill the *obj1* slot, and the localist representations of the 12 relationships listed above. The second layer converts the localist representations of the objects and the relations to distributed representations, and the third layer combines these two distributed representations into a distributed representation of the proposition. This layer is transformed into a distributed

[*] This research was supported by Contract MDA 903-89 K-0179 from the Army Research Institute, and NSF Grant DIR-9024251 to the UCLA Cognitive Science Program.

representation of *obj2* (the penultimate layer), which is finally converted into a localist representation of *obj2* (again consisting of 24 units representing the domain objects). The network is trained by the clamping the appropriate *obj1* and *rel* nodes on the input layer and *obj2* nodes on the output layer for each proposition, and performing backpropagation.

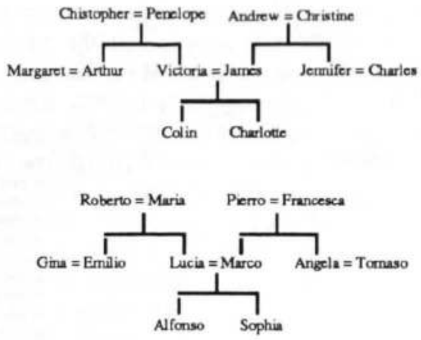


Figure 1: Two isomorphic family trees. The symbol "=" means "married to". (From Hinton, 1986)

To probe the analogical capacity of the network, we train the network on the full set of source propositions and a subset of target propositions. If the network has developed internal representations which allow it to utilize the similarity between the two domains of knowledge, it should be able to activate the correct *obj2* unit on the layer when the *obj1* and *rel* units of an untrained proposition are clamped on the input layer. Of course, success of the network in generating new target propositions does not licence the claim that the network is referring to specific propositions in the source domain. The network may, for example, be using mechanisms more closely related to rule-based inference. For example, if the network has learned that the sister of a person x is a female about the same age as x, it might be able to correctly infer that Sophia is the sister of Alfonso, even without reference to the fact that Charlotte is Colin's sister. However, we can reasonably claim that failure of the network in recreating untrained target propositions indicates that the network is *not* doing analogy. The point is that analogy is a sufficient but not necessary mechanism to make inferences within this problem domain.

The simulations reported throughout paper use Fahlman's (1988) quickprop algorithm, which is essentially a faster version of backpropagation. In standard backpropagation, each weight is adjusted in direct proportion to the magnitude of the partial derivative of the network's cost function with respect to the weight. In contrast, quickprop makes the simplifying assumption that the surface of the cost function is quadratic, and at each epoch, weights are set to the minimum value of the idealized surface by computing the curvature of the cost surface at the current point in weight space. In practical terms, this means that while backprop generally takes small steps in weight space, quickprop is capable of taking large leaps in weight space, which can greatly improve the overall learning times. For example, when the model was trained on all propositions in both domains, 4 runs of backprop using Hinton's (1986) parameters took an average of approximately 15,000 epochs to reduce the cost function to a value below 1.0[1], while 4 runs of quickprop took an average of approximately 500 epochs to meet the same criteria.

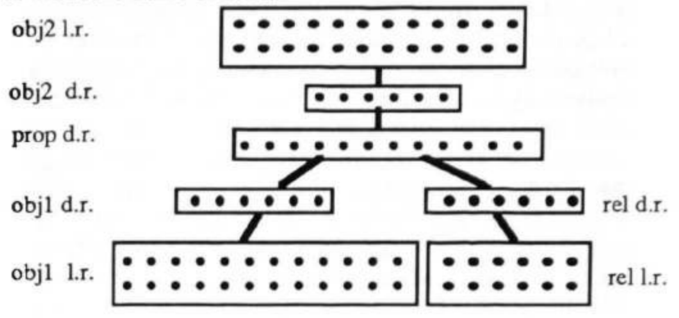


Figure 2: Hinton's (1986) model of propositional learning.

While the specific implementation details of the algorithms are not terribly important with respect to our present concerns, it is important to note that backprop and quickprop are fundamentally similar in that they both implement gradient descent on a cost function. Hence, we can reasonably expect that the internal representations that each algorithm develops will be qualitatively similar, although there may be slight differences as a side effect of the different implementations. The extent to which such differences exist remains an empirical issue, and is not explored further in this paper.

The parameters used for all simulations reported in this paper were $\mu = 1.75$, $\varepsilon = .01$ and *momentum* = 0.9 (see Fahlman 1988 for further explanation of the parameters and the algorithm). Weight decay was set either to 0.0 or 0.0005, depending on the set of simulations. Target values for the output were .8 if the output unit should be on, and .2 if the output should be off. The following cost function was used:

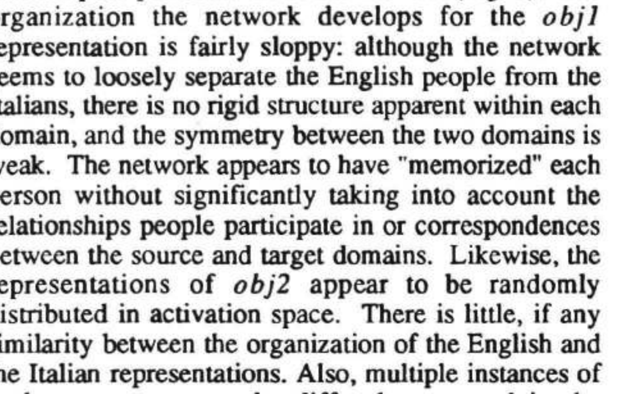
$$C = \Sigma_p \Sigma_o (t_{p,o} - o_{p,o})^2 + {}^1/_2 \lambda \Sigma_i w_i^2,$$

where p is an index over the training patterns, o is an index over the output units, i is an index over the weights, $t_{p,o}$ denotes the target value for unit o on pattern p, $o_{p,o}$ denotes the output value produced by unit o on pattern p, and λ is a weight decay parameter. In addition, one minor modification suggested by Hinton (1986) was implemented: if the network produces

[1] Note that the backprop training times significantly differ from those reported by Hinton. Hinton (personal communication, 1992) confirms that the training times reported in his paper are probably erroneous.

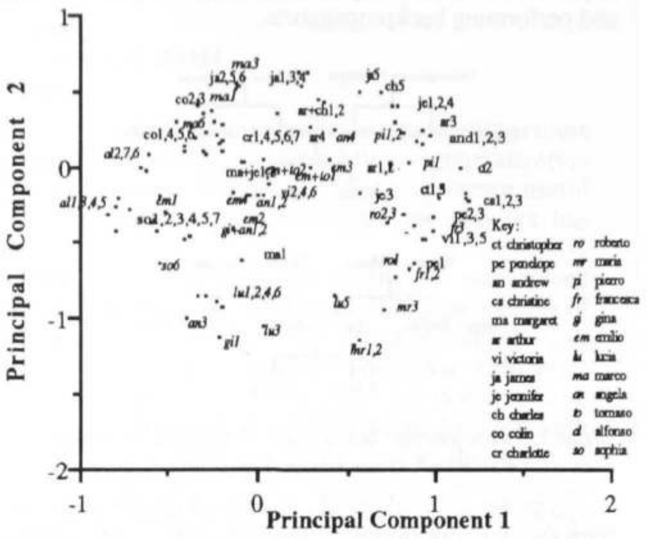
output values more extreme than the target values, the error for that unit is taken to be 0 rather than $(t_{p,o}-o_{p,o})^2$. When testing a training pattern, a liberal criterion is used: a case is considered to have "passed" if all units that should be off have activation values less than .4, and all units that should be on have activations values above .6.

For the first transfer test, the weight decay parameter was set to 0, and the network was trained on all source propositions and all but four randomly chosen target propositions. In the test phase, the network passed on all of the trained propositions, but failed on all of the untrained propositions. A principal components analysis of the distributed representations for *obj1* and *obj2* (i.e., the activations patterns on second and penultimate layers, respectively) yields some insight into the poor performance of the network (Fig. 3). The organization the network develops for the *obj1* representation is fairly sloppy: although the network seems to loosely separate the English people from the Italians, there is no rigid structure apparent within each domain, and the symmetry between the two domains is weak. The network appears to have "memorized" each person without significantly taking into account the relationships people participate in or correspondences between the source and target domains. Likewise, the representations of *obj2* appear to be randomly distributed in activation space. There is little, if any similarity between the organization of the English and the Italian representations. Also, multiple instances of each person seem to be diffusely scattered in the representation space. Ideally, we would like the network to develop a unitary representation for each person, so that the need for a many-to-one mapping between the distributed representation on the penultimate layer and the localist representation on the output layer is obviated. Such many-to-one mappings are bound to degrade the generalization of the network, as we shall see later.



(a)

Clearly, without additional constraints on the development of the network's internal representations, we can expect generalization to be poor. In the next two sections, I examine how the network can be induced to form sensible distributed representations, first by using knowledge of the domain derived from a connectionist model of analogical mapping, and secondly by employing domain- independent network optimization techniques.



(b)

Figure 3: Principal components analysis of (a) *obj1* and (b) *obj2* for basic gradient descent learning. Each instance of *obj2* is indexed by the order of occurence in the training set. English people are in plain text, and Italians are in italiacs.

Mapping Distributed Representations

Having entertained a couple of probable causes of the network's poor generalization performance, the path to improved generalization seems clear. We would like to be able to "program" the distributed representations to (1) reflect similarity between similar objects and (2) develop singular representations of multiple instances of each object. In this section, I describe how the first requirement can be met by making use of object correspondences produced by a connectionist model of analogical mapping.

In cases where regularities of the training environment are known a priori, hidden units can be "hardwired" to reflect the structure of the domain. For example, Rumelhart et al (1986) describe a network which develops translation-invariant "receptive fields" by constraining the weights of each hidden unit to take equal increments on each epoch. In general, however, it is desirable to do as little handcrafting of the network as is necessary; ideally we would like the network to be capable of flexibly extracting regularities from the environment using general purpose mechanisms. In the domain with which the present paper is concerned with (and in many other domains), analogy is one such mechanism which can assist in defining regularities.

By establishing correspondences between items in the source domain and target domain, we can pressure the network to form similar distributed representations for the mapped objects. However, we can't simply force representations of mapped objects to be identical; some representation of the domain must be present in the representation of each object in order for the network to be able to distinguish objects between domains. Hence, two components of each distributed representation are necessary: a *domain-invariant* component which represents the common aspects of objects between domains, and a *domain-specific* component which encodes the particular domain to which an object belongs. The domain-invariant component of the distributed representation will facilitate transfer between domains, while the domain-specific component will serve as a type of context indicator which will facilitiate specific modes of knowledge processing within each domain.

To create such representations, we first feed the propositional representations of the source and the (possibly incomplete) target domain into ACME (Holyoak & Thagard, 1989), a connectionist model of analogical mapping. ACME sets up a constraint-satisfaction network of mapping hypotheses based on structural, semantic, and pragmatic constraints, and produces a set of object and relation mappings by relaxing the network and returning the hypotheses with the highest activations (see Holyoak & Thagard, 1989, for further details). The object mappings produced by ACME are used to determine corresponding nodes on the input and output layers of Hinton's network (e.g., node 1, the localist representation of Christopher may correspond to node 13, the localist representation of Roberto). Before training the model, we randomize all weights in the network to values between -.3 and .3, with the following restrictions. Weights from corresponding nodes on the input layer projecting to all but one node in the *obj1* distributed representation layer are constrained to be equal. These nodes in the distributed representation layer will represent the domain-invariant component of *obj1*. The remaining second-layer node will encode the domain-specfic component: weights to this node from all source objects are set to +10.0, and weights to this node from all target objects are set to -10.0. Similarly, weights *to* all corresponding nodes on the output layer *from* all but one *obj2* distributed representation layer node are constrained to be equal, and the remaining node has weights of +10.0 to source objects, and weights of -10.0 to target objects. During training, weights that were set to be equal before training are constrained to stay equal by averaging the weight-error derivatives of the corresponding weights before each weight update. Weights that were set to ±10.0 are frozen at these values throughout training.

Note that while this scheme *forces* the network to form distributed representations of *obj1* that are "mapped", the distributed representations of *obj2* are only *pressured* to be mapped, since it is the weights feeding out of, rather than into, the penultimate layer that are constrained. Hence, it might be possible for the network to develop disparate representations of the source and target *obj2*s, although empirically this doesn't seem to occur. Figure 4 shows the distributed representations of *obj1* and *obj2* developed by training the network on the same training set as was used previously (i.e., the entire set of source propositions, and all but 4 target propositions) plotted against the two principal components derived from a principal components analysis. Unsurprisingly, the source and target representations developed for *obj1* are perfectly symmetric. More significantly, the *obj2* representations are also symmetric: the representation of each instance of each English person is adjacent to the corresponding Italian instance, although the instances are still scattered uniformly in representation space. The result of these constraints is a significant improvement in generalization: all four of the untrained target propositions were correctly constructed by the network.

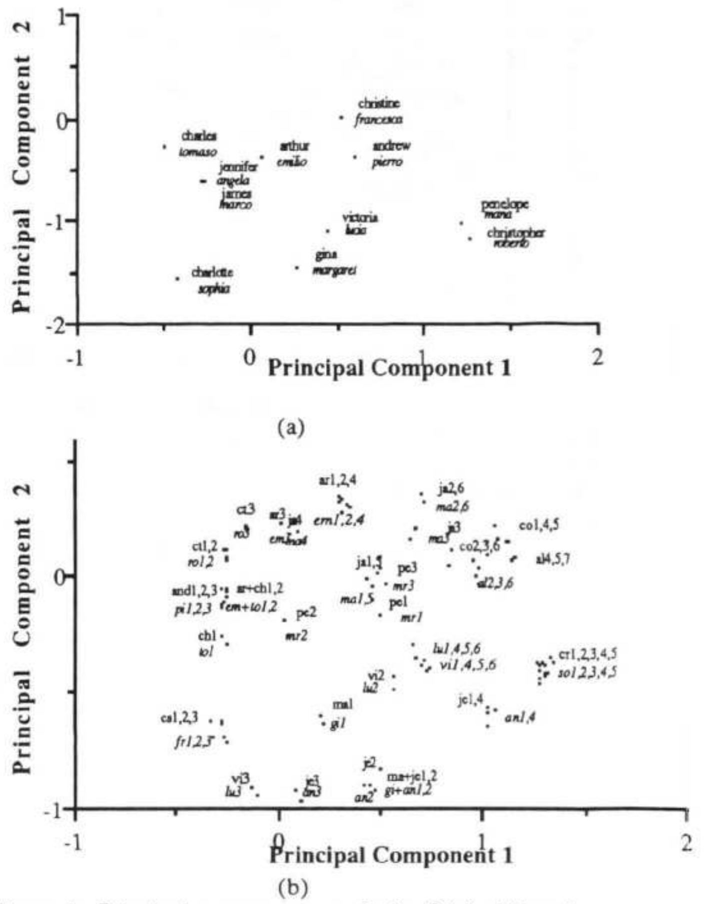


Figure 4: Principal components analysis of (a) *obj1* and (b) *obj2* using the distributed representation mapping technique.

The distributed-representation mapping technique successfuly fulfills our first condition for improved generalization: the network develops representations which reflect similarity between the source and target domains. However, our second condition remains unfulfilled: the network has failed to develop singular representations for multiple instances of objects. To induce the network to develop parsimonious representations, general-purpose network optimization techniques may be successfully employed.

Network Optimization Techniques

Numerous techniques for optimizing the performance of backpropagation have been proposed in the connectionist literature, including weight decay (e.g. Plaut, Nowlan & Hinton 1986; Weigend, Huberman & Rumelhart, 1990; Hanson & Pratt, 1989), eliminating weights (Chauvin, 1989), excising or attenuating hidden units (Kruschke, 1988, Mozer & Smolensky, 1989), limiting the total amount of hidden unit activation (LeCun, 1989), reducing the dimensionality spanned by the hidden weight vectors

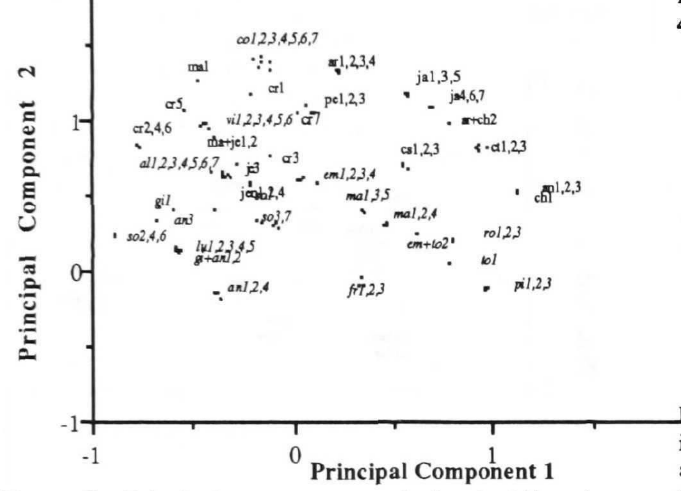


Figure 5: Principal components analysis of *obj2* using weight decay.

(Kruschke, 1988), and forcing hidden layer representations to have uncorrelated node activations (Kruschke & Movellan, 1990). Most of these techniques pressure backprop to make economical use of network resources such as weights, nodes or hidden-unit activation. For example, the second term in eq. 1 ($^1/_2\lambda\Sigma_i w_i^2$) implements weight decay by pressuring the network to minimize the sum of the squares of all the weights.

Figure 5 shows the results of a principal components analysis of the *obj2* representations when the network was trained on the same set of propositions as in the previous examples, and a weight-decay value of λ = .0005 was used. As expected, multiple instances of each *obj2* are tightly clustered together. The effect of this clustering is a substantial improvement in transfer performance over basic gradient descent: 3 out of the 4 deleted propositions were correctly recovered (compared with recovery levels of 0 for basic gradient descent and 4 for the distributed representation mapping technique). However, the symmetry between the structure of the source and target domains is still imperfect; hence we should not be surprised that knowledge transfer between domains is imperfect. In the next section, I describe a set of computational experiments which systematically test the efficacy of weight decay and the distributed representation mapping technique in improving transfer performance.

Transfer Tests

Four sets of simulations were run, in which the weight decay parameter was set to either 0 or .0005, and the distributed mapping technique was either used or not used. Within each set of simulations, three different amounts of target propositions were omitted from the training set: 4 (8% of the target propositions), 13 (25%), and 26 (50%). Learning was halted when the first term of the cost function dropped below 0.1, or 4000 epochs had elapsed.

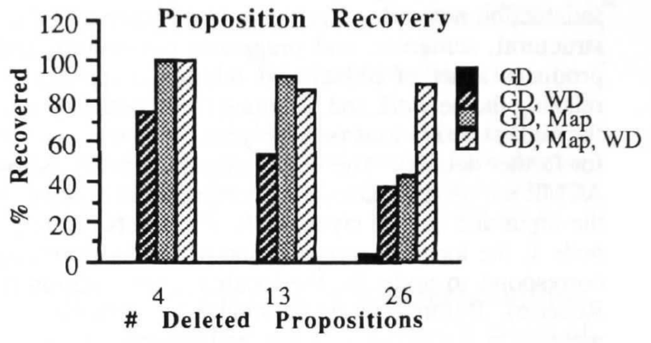


Figure 6: Recovery of Deleted Propositions. "GD" indicates gradient descent, "WD" indicates weight decay, and "Map" indicates the use of the distributed mapping technique.

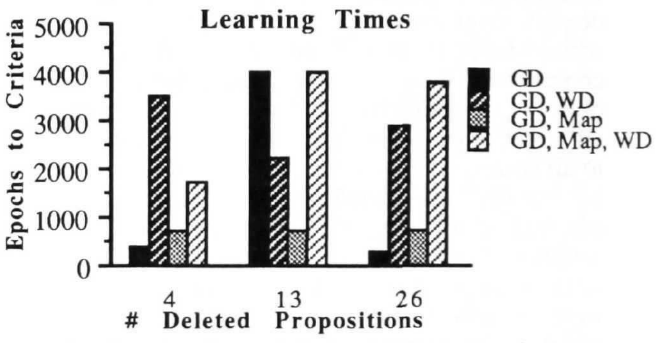


Figure 7: Learning times. Labels are the same as in the previous graph. The second GD run was stuck in a local minimum (2 of the training cases were unlearned at cycle 4000).

The transfer test results (Fig. 6) show that while basic gradient descent is incapable of recovering more than the tiniest fraction of deleted propositions, the distributed representation mapping technique and weight decay both substantially improve transfer, although performance is degraded at higher levels of deletion. The most significant result is that when weight decay is coupled with the distributed representation mapping technique, nearly all of the deleted propositions are recovered, even at the highest level of deletion. This result is obtained because the two conditions of good generalization performance earlier positied are satisified by the two respective techniques: weight decay forms unitary representations of the objects, while the distributed representation mapping technique ensures that similar objects have similar representations.

The order of learning times (Fig. 7) is roughly GD < GD+Map < GD+WD = GD+Map+WD Intuitively, these results are quite sensible: basic gradient descent is the quickest to reach the stopping criteria since it can arive at the closest possible solution without "worrying" about additional solution constraints. Gradient descent with weight decay takes longer to learn than gradient descent with the distributed representation mapping technique because in the case of weight decay, the network must satisfy the conflicting pressures of predicting the training set and driving all weights to 0. In contrast, the distributed representation mapping technique presents two harmonious pressures in the network: the two goals of training set prediction and object representation correspondences can be achieved without conflict.

Additional evidence supports the hypothesis that weight decay and the mapping technique perform complementary optimization functions. For example, computation of the average euclidian distance between each instance (e.g. *emilio1*) in the penultimate layer and the instance's class (e.g. *emilio*) reveals that instances are relatively diffuse when weight decay is not employed. Another important fact is that increasing the weight decay parameter does not substantially improve the generalization performance: as λ is increased, performance on the training set deterioriates as well as performance on the generalization set.

Conclusion

A technique has been described in which high-level domain knowledge produced by a connectionist model of analogical mapping guides the formation of distributed representations of domain objects formed by Hinton's (1986) multilayer model of propositional learning. Simulation results indicate that the use of this technique in isolation can produce substantial improvement in the generalization performance of Hinton's network, and use of this technique in conjunction with weight decay can produce nearly perfect transfer performance. More generally, this paper demonstrates the usefulness, and perhaps even the necessity, of the influence of high level knowledge processing mechanisms on low level subsymbolic learning mechanisms.

Acknowledgements

Keith Holyoak and John Hummel provided helpful comments on an earlier draft.

References

Chauvin, Y. (1989). A back-propagation algorithm with optimal use of hidden units. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems, I.* San Mateo, CA: Morgan Kaufman.

Fahlman, S. E. (1988). Faster learning variations on back-propagation: an empirical study. In D. Touretzky, G. Hinton, & T. Sejnowski (eds.), *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41,* 1-63.

Hall, R. P. (1989) Computational approaches to analogical reasoning: a comparative analysis. *Artificial Intelligence, 39,* 39-120.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

Holyoak, K. J., Novick, L. R., & Melz, E. R. (1992, in press). Component processes in analogical transfer: Mapping, pattern completion, and adaptation. To appear in K. J. Holyoak & J. A. Barnden (eds.), *Advances in Connectionist and Neural Computation Theory, Vol. 2: Analogical Connections.* Norwood, NJ: Ablex.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13,* 295-355.

Kruschke, J. K. (1989). Creating local and distributed bottlenecks in hidden layers of back propagation networks. In D. Touretzky, G. Hinton, & T. Sejnowski (eds.), *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

Kruschke, J. K., & Movellan, J. R. (1991). Benefits of gain: Speeded learning and minimal hidden layers in back-propagation networks. *IEEE Transactions on Systems, Man, and Cybernetics, 21 (1).*

LeCun, Y., Denker, J. S., & Solla, J. A. (1989). Optimal Brain Damage. In: D. Touretzky (ed.), *Advances in Neural Information Processing Systems, I.* San Mateo, CA: Morgan Kaufman.

Mozer, M. C., & Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems, I.* San Mateo, CA: Morgan Kaufman.

Plaut, D. C., Nowlan, S. J., & Hinton, G. E. (1986). Experiments on learning by back propagation. Carnegie Mellon University Department of Computer Science Technical Report CMU-CS-86-126.

Rumelhart, D. E., Hinton, G. E., & Williams, R. L. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1.* Cambridge, MA: MIT Press.

Weigend, A. S., Huberman, B. A., & Rumelhart, D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems 1 (3),* 193-209.