知識工学と人工知能 64 － 6
（１９８９．５．１８）

コネクショニストモデルを用いた知識ベースシステム

萩原 撃[1]
ソニー（株）総合研究所

概念の検索要求と、演えき型知識ベース内のファクトの集合から、動的にコネクショニストネットワークを生成する手法を提案する。この手法を用いることにより、従来のコネクショニスト型知識ベースの特徴である、頑丈性、文脈依存性に加えて、複数概念の並列検索と、ネットワークの大きさの削減が可能になる。また、ネットワークに変換する前の知識の状態が記号表現で得られるため、知識ベースの可読性も向上する。

KNOWLEDGE BASE SYSTEM BASED ON CONNECTIONIST MODEL

Kaoru Hagiwara
Sony Corporate Research Laboratories

We propose a method which dynamically creates a connectionist network from deductive knowledge base and the query given by a user. By creating the network dynamically, we can retrieve multiple conept in parallel from the network, and we can reduce the network size drastically. Also, we can read and maintain the knowledge base in pretrasnformed symbolic representation.

[1]現在 (財) 新世代コンピュータ技術開発機構に出向中。メイルアドレス、 hagiwara@icot.junet

1 はじめに

知識ベースシステムの概念検索にコネクショニストモデルを用いる手法と、その実験結果について報告する。

概念の検索とは、与えられた一つ又は複数の制約を満たす概念を知識ベース内の事実の集合の中から検索することである。従来の演えき型知識ベースは、システム内のファクトの集合と、推論規則から完全に推論されるもののみを検索結果としていた。その結果、知識ベースの完全性、健全性は保証されるが、ユーザが与える概念の制約の中や、システム自身の知識に誤りがあると、概念の検索に失敗してしまうことがある。

知識ベースの管理を十分行うことにより、システム自身の知識の完全性、健全性の維持は可能となる。しかし、ユーザの質問中の誤りのチェックは困難である。特に、一般のエンドユーザを対象とした知識ベースシステムは、質問文中に多少の誤りがあっても、もっともらしい答を返してくれる頑丈性が必要となる。

このような知識ベースの頑丈性を実現する手法として、コネクショニストモデル[4]を用いた知識ベースシステムが提案されている [1, 4, 5]。コネクショニストモデルを用いることにより、ユーザが与えた制約の中に知識ベースとマッチングしないものが存在しても、最も多くの制約を満たす概念が選ばれるようになり、知識ベースの頑丈性が自然に実現できる。さらに、検索結果にあらかじめ予測を与えることにより、文脈に依存した検索も可能になる。

本報告では、概念検索のためのコネクショニストネットワークを、ユーザの概念の検索要求と、知識ベース内の事実の集合からダイナミックに作り出す手法を提案する。従来のコネクショニスト型知識ベース[1, 4, 5] と比較して以下のような特徴が得られる。

ユーザの質問に応じてコネクショニストネットワークを生成するため、複数の概念の並列検索が可能になり、しかも検索に関係のないユニットが生成されないので、ネットワークの大きさを著しく小さくすることができる。また、知識ベース内の事実と、ユーザが与える概念の検索要求が共に演えき型知識ベースと同じ記号表現で与えられるため、知識ベース内部の知識の可読性が従来のコネクショニスト型知識ベースと比較して向上する。

2 コネクショニストモデル

コネクショニスト知識ベース（以下われわれのシステムをこう呼ぶことにする）の説明をする前に、簡単に我々のシステムで用いるコネクショニストモデルについて述べる。

2.1 コネクショニストモデルの特徴

パターン認識や音声認識の分野で盛んに研究されているニューラルネットワークを人工知能の分野に応用しようという研究が最近盛んに行われるようになってきた。知識ベースシステム[4, 5, 1]、自然言語の理解のモデル [4, 7]、エキスパートシステム[2, 6] などへの応用が研究されている。

人工知能の観点から見ると、コネクショニストモデルは以下のような特徴を持っている。

- 多数のユニットの活性値の更新を動作アルゴリズムとする、並列分散処理
- ユニット間のリンクの重みの更新による、学習
- 多数のユニットの活性化状態のパターンにより意味を表現する、分散表現

これらの特徴のうち我々のシステムは並列分散処理の特徴のみを用いることにする。他の特徴を用いていないという点では、コネクショニストモデルの機能を十分生かしていないのだが、コネクショニス型の並列分散処理を導入するだけでも、知識ベースの頑丈さは実現できる。学習機能と、分散表現に関しては、最後の考察で触れることにする。

2.2 活性値の拡散

コネクショニストモデルの動作原理を簡単に説明する。コネクショニストネットワークは、多数のユニットとそれらの間を結ぶリンクからなっている。各ユニットは活性値と呼ばれる内部状態を持っており、リンクもそれぞれの重みを持っている。

全てのユニットの活性値を更新するサイクルを活性化拡散サイクルと呼ぶ。我々のモデルの活性値拡散サイクルは、以下の活性値の更新手順を、全てのユニットで同期を取って実行する。

if($net_i > 0$)
$\Delta_i = (\max - a_i) net_i - decay(a_i - rest)$
else
$\Delta_i = (a_i - \min) net_i - decay(a_i - rest)$
ただし、
$net_i = \sum_j w_{ij} output_j + extinput_i$

a_i : ユニットu_i の活性値
$extinput_i$: ユニットu_i への外部からの入力 (0)

$output_i$: ユニットu_i の外部への出力

$$output_i = \begin{cases} a_i & \text{if } a_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

max : 活性値の最大値 (1.0)
min : 活性値の最小値 (-0.2)
$rest$: ユニットに入力がない時に落ち着く活性値 (-0.1)
$decay$: 活性値を $rest$ に落ち着かせる割合 (0.1)

括弧の中の値は、我々のシステムで実際に用いている値である。この活性化拡散サイクルを与えられたネットワークと外部入力に対して繰り返すことがコネクショニストモデルの計算モデルである。コネクショニストモデルにおける学習はリンクの重みを修正することにより行われるが[4]、我々のモデルでは、リンクの重みは与えられた初期に固定しており、学習は行わない。

上記の式から、ネットワーク中の全てのユニットの活性値は以下のような変化をすることが分かる。

- 活性値は予め定められた最小値 min 以上、最大値 max 以下の値のみをとる。
  $\min \leq a_i \leq \max$
- ネット入力が正の場合は、そのユニットの活性値は増加する。負の場合はその逆。
- 他のユニット及び、外部からの入力の総和 (net_i) がゼロの状態が続くと、そのユニットの活性値は次第に$rest$ に落ち着いて行く。つまり、外部から刺激のないユニットは非活性化して行く。

2.3 interactive activation and competition

我々のコネクショニスト知識ベースでは、interactive activation and competition モデル（以下iacモデルと呼ぶことにする）と言うコネクショニストネットワークを用いている。

iac モデルのネットワークは、いくつかのクラスタ（ユニットの集まり）に分割されている。つまり、全てのユニットは何れか一つのクラスタに属している。同一クラスタ内の全てのユニットの間には相方向の抑制性のリンクが張られ、異なるクラスタ内のユニット間には興奮性のリンクが張られているか、あるいはリンクは張られていない。抑制性のリンクとは負の重みを持つリンクで、興奮性のリンクとは正の重みを持つリンクである。興奮性のリンクによる結合を 興奮性結合、抑制性のリンクによる結合を抑制性結合と言う。 iac モデルでは、すべてのリンクは相方向である。

iac モデルの特徴は、各クラスタ内で一つのユニットのみを活性化（活性値を大きくする）し、他のユニットは非活性化（活性値を小さくする）するように動作する点である。これは、同一クラスタ内のすべてのユニットが抑制性結合されていることによる。iac モデル上での活性値拡散サイクルは、一般的には、停止する（全てのユニットの活性値が変動しなくなる）保証はない、しかし経験的には殆どの場合停止することが確認されている[4]。

3 コネクショニスト知識ベース

このセクションでは、まず Rumelhart らのコネクショニストモデルを用いた知識ベースシステムを紹介した後に、我々のコネクショニスト知識ベースを説明する。

3.1 コネクショニストモデルを用いた知識ベース

コネクショニストモデルを用いた知識ベースが提案されている [4, 5]。これらのモデルは、iac モデルのユニットの選択機能を用いている。この知識ベースは、検索対照となる概念を表すユニットの集合からなるクラスタと、それらの概念が取り得る属性値を表すユニットを属性ごとに分けるクラスタ（属性クラスタ）からできている。iac モデルであるから、同一クラスタ内の全てのユニットは抑制性結合されている。このことにより、複数の概念の中から一つの概念のみを選択する機能と、概念の属性値としては一つのみが選択されるという機能が得られる。また、各概念ユニットと、それが持つ属性値ユニットの間は興奮性結合されている。

概念の検索をする際には、各属性クラスタ内の、検索すべき概念が持つ属性のユニットの活性値は高くし、持たない属性のユニットは低く設定する。このような活性値の初期設定から活性値の拡散を繰り返すことにより、検索すべき概念のユニットのみが活性化する。これが iac モデルを用いた概念の検索である。

iac モデルを用いることにより、以下のような演えき型知識ベースにない特徴が得られる。

- 頑丈性：システム内の知識や、ユーザの質問の中に多少の誤りが含まれていも、ほぼ正しい答を返す。
- 文脈依存性：検索に先だって、あらかじめ検索結果に予測を与えられる。

これらの特徴については、後にまとめて述べる。

3.2 コネクショニスト知識ベースの概要

われわれのコネクショニスト知識ベースも iac モデルを用いている。Rumelhart らのモデルと異なっているのは、知識ベース内の知識（ファクトの集合）とユーザからの概念検索要求に応じて、必要なネットワークを動的に生成する点である。動的にネットワークを生成することにより、先に上げた Rumelhart らのモデルの特徴に加えて、以下のような特徴が新たに得られる。

- 複数の概念を並列に検索できる。
- ユーザの質問に関係のあるユニットのみを生成するので、ネットワークの大きさが小さくなり、計算時間も速くなる。
- 知識ベース内の知識（ファクトの集合）が記号で表現されているので、プログラムの可読性が向上し、知識ベースの保守が容易になる。

3.3 知識ベース内の知識

知識ベース内の知識はファクトの集合からなる。各ファクトは、kb(概念、属性名、属性値) の三つ組で表す。例えば、

kb(cat, hates, dog)

は、「猫は犬を嫌う」という事実を表している。それぞれの引数はすべて定数で、変数は含まないことにする。また、推論規則は知識ベース内には含まないことにする。

3.4 概念の検索の入力形式

ユーザが知識ベースに与える概念の検索は、三つ組の中に変数を含めることにより表現する。例えば、

kb(X, hates, dog)

の場合、X は変数で、「犬を嫌っている物 (X) はなんだ？」という概念の検索を表している。ここでは、概念検索のための変数を含んだファクトを変数の制約と呼ぶことにする。上の例の場合、変数 X に対する制約を表していると考える。

概念の検索の際に、変数の制約を複数与えることができる。この場合、それら全ての制約を最も良く満足する概念を検索する。論理積とは異なり、必ずしも全ての制約が満たされる必要はない。例えば、

[kb(X, hates, dog), kb(X,eats, mouse)]

は、「犬が嫌いで、鼠を食べるもの (X) は何だ？」という概念の検索を表しており、二つの変数の制約を満たす概念がない場合は、どちらか一方を満たす概念を返す。

概念の検索要求の中に複数の変数を含んでも良い。その場合、変数の数だけの概念を並列に検索することになる。例えば、

[kb(X, hates, Y),kb(Y, eats, mouse)]

は、「鼠を食べるある何 (Y) か、を嫌っているもの (X) は何だ？」という概念の検索を表しており、これらの制約を最もよく充足する二つの概念が同時に検索される。

3.5 ネットワークの構造

上で説明した、知識ベース内のファクトの集合と、ユーザからの概念の検索要求から、システムはコネクショニストネットワークを生成する。このネットワークは、概念の検索要求として与えられた変数の制約を表しているので、**変数制約ネットワーク** と呼ぶことにする。変数制約ネットワークの生成アルゴリズムを述べる前に、生成すべきネットワークの構造を説明する。

いま、変数の制約の集合が与えられ、その中に n 個の変数が含まれているとする。変数制約ネットワークには、検索すべき概念の候補のユニットを集めたクラスタがある。このクラスタを、**変数クラスタ** と呼び、その中のユニットを **概念ユニット** と呼ぶ。n 個の変数があるの場合、n 個の変数クラスタが存在する。もちろんそのクラスタ内の全てのユニットは抑制性のリンクで結合されており、検索すべき概念を一つに選択する性質を持っている。

概念のユニットの他に、変数の制約式に対応するユニット（**制約ユニット**）が存在する。一変数の制約式の場合、対応する制約ユニットが一つ存在する。そして、その制約式を満足する概念に対応する概念ユニットとの間に相方向の興奮性結合が張られている。二つの変数を含む制約式の場合、その制約式を充足する知識ベース内のファクトの組み合わせの数だけ制約ユニットが存在する。それらのユニットは同じクラスタ（**制約クラスタ**）内にあり、相方向の抑制性結合で結ばれている。また、それぞれの制約ユニットとそれが満足する概念ユニットとの間には相方向の興奮性結合が張られている。

例えば、以下のような知識ベースのファクトの集合が与えれた時、

kb(cat, hates, dog).
kb(monkey, hates, dog).

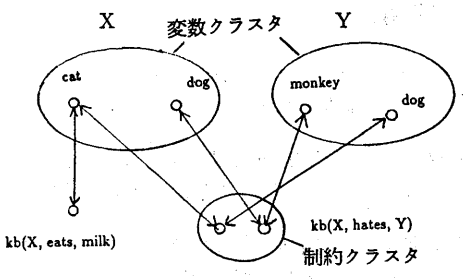

図 1: 2 変数の制約ネットワーク

kb(cat, eats, milk).
kb(dog, eats, milk).

次のような概念の検索を要求したとする。

[kb(X, eats, milk), kb(X, hates, Y)]

すると、図1 のような変数制約ネットワークが生成される。

3.6 ネットワーク生成アルゴリズム

知識ベース内のファクトの集合と、ユーザが与える変数の制約式の集合から、変数の制約ネットワークを生成するアルゴリズムは以下のようになる。

1. 制約式の中に含まれている全ての変数に対応する変数クラスタを作る
2. 全ての制約式が処理済みならば、ステップ 4 へ。そうでなければ、制約式の集合の中から未処理の制約式 e を一つ選ぶ

3.1 制約式 e がただ一つの変数を含む場合、
- 3.1.1 その制約式を満たす概念を、知識ベース内のファクトの集合の中から全て選びだし、対応する概念ユニットを対応する変数クラスタ内に作成する。
- 3.1.2 制約式 e に対応する制約ユニットを作り、上のステップで作った全ての概念ユニットとの間に興奮性のリンクを張り、ステップ 2 に戻る。

3.2 制約式 e が二つの変数を含む場合、
- 3.2.1 制約クラスタを一つ作る
- 3.2.2 制約式 e を満たす概念の組み合わせを、ファクト集合の中から全て選び出す。
- 3.2.3 上記の全ての概念の組み合わせについて、
  - a.変数クラスタの中に対応する概念ユニットが存在していないならば、新しい概念ユニット$u_{1,2}$を変数クラスタ内に作る
  - b. 新しい制約ユニットを制約クラスタ内に作り、概念ユニット$u_{1,2}$ との間に興奮性のリンクを張る。
- 3.2.4 制約クラスタ内の全てのノードを互いに抑制性のリンクで結び、 2 に戻る。

4. 各変数クラスタ内の全ての概念ユニット間に抑制性のリンクを張って、終了。

このネットワーク生成アルゴリズムの計算時間は、知識ベース内のファクトの検索手法によって異なるが、ハッシングをおこなうことにより、ファクトの数にほぼ無関係になる（少なくとも比例はしない）。Rumelhart らの知識ベースでは、概念の検索時間はユニット数の二乗に比例するので、我々のアルゴリズムの計算時間は、知識ベースのファクトの数が大きい場合には、無視できる。

3.7 概念の検索手順

概念の検索は、上記のアルゴリズムにより生成された変数制約ネットワークに活性化拡散サイクルを適用することにより、行われる。

各ユニットの活性値の初期値は以下のように決めている。

- 概念ユニットならば活性値は負 (-0.1)
- 制約ユニットならば活性値は正 (0.3)

制約ユニット活性化しておくのは、その制約が成立していることを表している。逆に概念ユニットを非活性化しておくのは、まだその概念が検索すべき概念の一つの候補に過ぎずないためである。これらの値は、経験的に定めたものであり、数学的な根拠はない。例えば、制約ユニットの値を全て 0.2 にすると複数の概念が選ばれてしまうこともある。活性値の初期値の設定の仕方が検索の結果に微妙に影響してくる。

逆にこの性質を用いると、概念ユニットの活性値の設定の仕方によって、検索結果に予測を与えることができる。この点については後に説明する。また、制約ユニットの活性値の初期の与え方により、制約に重み付けをすることも可能である。

活性化拡散サイクルを繰り返すと、ほとんどの場合、次第にネットワーク全体の活性値の変動が落ち着いてくる。活性化拡散サイクルには自分を停止する機構はないので、一定の条件でこのサイクルを停止させる必要がある。全てのユニットの活性値が変化しなくなることを停止条件とすることも可能であるが、これには以下のような問題点がある。

- 活性値の変動が永遠に止まらないことが稀にある。

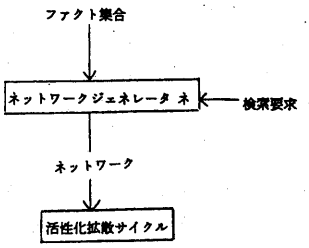

図 2: システムの構成

- 活性値の変動が完全に止まるの以前に、検索対象の概念は特定できてしまうので、完全に止まるまでサイクルを繰り返すのは効率が悪い。

現在のところ、サイクルの停止判断は人間がネットワークの状態を見ながら行っている。効率の良い停止条件を定式化することは我々のシステムの課題の一つである。

3.8 システムの構成

以上述べてきたコネクショニスト知識ベースの実験システムを作成した。構成は図2 のようになっている。知識ベースのファクト及び、変数制約ネットワークのジェネレータは Prolog を用いて作成した。ネットワークジェネレータはファクトの集合及び、ユーザからの概念検索要求を入力として受け取り、変数制約ネットワークを活性化拡散サイクルに受け渡す。

活性化拡散サイクルは、与えれた変数制約ネットワークに活性値拡散を繰り返すことにより、概念を検索する。現在のところ、このサイクルの停止は人間が行う。活性化拡散サイクルのプログラムは Rumelhart らの実験システム [5] を一部改変して用いている。

Rumelhart らのシステムの活性化拡散サイクルは、ネットワークの活性値の状態変化を画面上にグラフィカルに表示できるので、検索の経過を見ながら、活性値を更新したり、ネットワークのトポロージーを変更できる。このように、ユーザとシステムがある種の対話を行いながら概念の検索をすることが可能である。

4 コネクショニスト知識ベースの特徴

我々のコネクショニスト知識ベースの特徴をまとめると以下のようになる。

頑丈性： システム内の知識やユーザの検索要求中の誤りに対してロバストな概念の検索が可能。

文脈依存性： 検索結果に予測を与えることにより、文脈に依存した検索が可能になる。

以上の特徴は Rumelhart らの知識ベースでも実現可能。以下の特徴は、我々のコネクショニスト知識ベース特有のものである。

複数概念の並列検索 検索要求の中に複数の変数がある場合、それらを並列に検索できる。

ネットワークの大きさの削減： 検索要求に応じて必要最小限のネットワークを生成するので、ネットワークの大きさを削減できる。その結果、概念検索の計算時間も短縮される。

知識ベースの可読性の向上： ネットワークに変換される前の知識ベースの状態が記号表現で得られるので、従来のコネクショニスト型知識ベースと比較し、知識の可読性が著しく向上する。

これらの特徴のうち、頑丈性、文脈依存性についてそれぞれ例を用いて説明する。複数概念の並列検索、ネットワークの大きさの削減、及び可読性の向上についてはそれぞれの例の中で触れることにする。

以降の例では、図3 に与えられたファクトの集合を用いている。

4.1 頑丈性

知識ベース内のファクトの誤りやユーザからの検索要求の中に多少の誤りが含まれていても、我々のシステムはもっともらしい検索結果を返す。これは、コネクショニストモデルが、連続した活性値を使っていることから得られる性質である。

しかし、ノイズに強いということは、知識ベースないのファクトからは本来演えきできない事実を答えていることになり、知識ベースの健全性は失われている。つまり、コネクショニスト知識ベースの検索した概念は常に完全に正しい分けではない。しかし、得られた答えは、少なくとも知識ベース内のファクトの中ではもっともらしいものだと思われる。

図3 のファクト集合に対して、以下の検索を要求したとする。

kb(dog, hates, monkey).
kb(dog, eates, meat).

kb(monkey, hates, dog).
kb(monkey, eats, banana).

kb(cat, hates, dog).
kb(cat, eats, milk).
kb(cat, eats, mouse).

kb(mouse, hates, cat).
kb(mouse, eats, cheese).
kb(mouse, eats, milk).

kb(banana, isa, food).
kb(banana, color, yellow).

kb(milk, isa, food).
kb(milk, color, white).

kb(cheese, isa, food).
kb(cheese, color, yellow).

図 3: ファクト集合の例

[kb(X, hates, dog), kb(X, eats, milk),
kb(X, eats, cheese)]

これらの制約式をすべて充足する概念は、図3 のファクト集合の中には存在しない。kb(X, eats, cheese) を取り除いた場合は、X = cat という結果が得られる。我々のシステムの場合、生成されるネットワークは図4 のようになる。一つの概念の検索なので、一つの変数クラスタが作られ、その中に候補の概念ユニットがある。

この変数制約ネットワークに対して、40 回の活性化拡散サイクルを繰り返すと、以下のような結果が得られる。ただし、概念ユニットのラベルの先頭には、変数クラスタ名を付けてある。例えば、x_cat は、変数 X の変数クラスタ内の cat という概念ユニットを表すラベルである。また、各制約ユニットの末尾には、その制約が何番目の制約式に対応しているかをを示す番号が付けてある。例えば、hate1 は、第一式に対応する制約ユニットである。

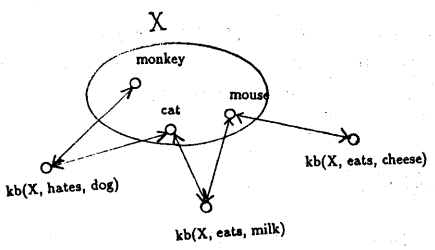

図 4: 頑丈性の例

サイクル数 0
x_cat -0.1 x_dog -0.1 x_monkey -0.1
x_mouse -0.1
hate1 0.3 eat2 0.3 eat3 0.3

サイクル数 40
x_cat 0.63 x_dog -0.15 x_monkey -0.11
x_mouse -0.11
hate1 0.51 eat2 0.51 eat3 -0.9
X = cat

40 サイクル後の結果を見ると、検索すべき概念が「猫」であることが分かる。制約ユニットを見ると、 eat3 ユニットのみが非活性化されており、このことから制約式 kb(X, eats, cheese) が充足されていないことが推測できる。

この例の場合、ノイズである kb(X, eats, cheese) の活性値の初期値を大きくすることにより、全ての概念の検索に失敗させることも可能である。

4.2 文脈依存性

コネクショニスト知識ベースでは、概念を検索する際に、結果に対する予測（偏見）を与えることができる。これは、複数の検索結果がある場合に、文脈に依存した検索ができることになり、自然言語の意味処理などに応用が考えられる。

先のファクト集合に対して、以下の検索を要求したとする。

[kb(X, hates, Y), kb(X, eats, Z)]

これは、三つの変数を含んでいるので、三つの概念が並列に検索される。図5 のように三つの変数クラスタを持つ変数制約ネットワークが生成される。

この変数制約ネットワークに対して予測を与えない場合と、与えた場合の検索結果は以下のようになる。

予測を与えない場合
サイクル数 0

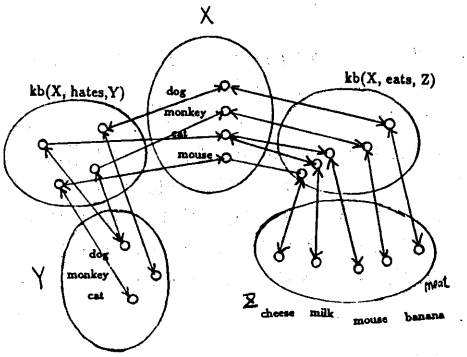

図 5: 文脈依存性の例

x_cat -0.1 x_dog -0.1 x_monkey -0.1
x_mouse -0.1
y_cat -0.1 y_dog -0.1 y_monkey -0.1
z_banana -0.1 z_cheese -0.1 z_meat -0.1
z_milk -0.1 z_mouse -0.1

サイクル数 40
x_cat 0.74 x_dog -0.14 x_monkey -0.13
x_mouse -0.13
y_cat -0.13 y_dog 0.54 y_monkey -0.13
z_banana -0.13 z_cheese -0.13
z_meat -0.13 z_milk 5.3 z_mouse -0.13

X = cat, Y = dog, Z = milk

Z = bananaと予測する場合
サイクル数 0
x_cat -0.1 x_dog -0.1 x_monkey -0.1
x_mouse -0.1
y_cat -0.1 y_dog -0.1 y_monkey -0.1
z_banana 0.4 z_cheese -0.1 z_meat -0.1
z_milk -0.1 z_mouse -0.1

サイクル数 30
x_cat -0.15 x_dog -0.15 x_monkey 0.79
x_mouse -0.15
y_cat -0.15 y_dog 0.63 y_monkey -0.15
z_banana 0.65 z_cheese -0.15 z_meat -0.15
z_milk -0.15 z_mouse -0.15

X = monkey, Y = dog, Z = banana

このように、ユーザが概念ユニットの初期値を操作することにより、検索結果を変えることができる。

5 考察

記号表現で与えられるユーザの検索要求と、知識ベース内のファクト集合から、ダイナミックにコネクショニストネットワークを生成する手法を提案した。この手法を用いることにより、従来の演えき型知識ベースあるいは、コネクショニスト型知識ベースでは実現が困難であった、複数概念の並列検索、ネットワークの大きさの削減、ファクト集合の保守性の向上、などの新たな特徴が得られることが確認された。

制約プログラミング[3]では、制約をプログラムとして捉え、制約の変換を計算と考える。コネクショニストモデルは、ネットワークによって与えられた制約（の集合）を、活性化拡散サイクルにより変換しているので、一種の制約プログラミングである。

6 課題

我々のコネクショニスト知識ベースには、以下のような課題がある。

推論規則の埋め込み： 知識ベース内に推論規則を埋む必要がある。

階層構造： 概念の間にisa階層を持たすことにより、知識の記述の量が減少し、知識ベースの保守性も向上する。

検索終了の判定： 活性化拡散サイクルを終了する条件の設定する。現在のシステムは、人間がその判断をしている。

初期値の設定： 活性値の初期値の設定の仕方に何らかの法則性を見出す必要がある。

分散表現： コネクショニストモデルの特徴のひとつである概念の分散表現を知識ベースシステムに導入することにより、より柔軟な概念の検索が可能になる。その場合、各概念を表現する活性パターンを学習させる必要がある。

実用的なシステムを構築するためには、ネットワーク状況の表示、修正をグラフィカルに行うユーザインタフェイスを整備する必要がある。

参考文献

[1] Diederich, J., (1988) *Knowledge-Intensive Recruitment Learning*, TR-88-010, International Computer Science Institute.

[2] Gallant,S.I., (1988) *Connectionist Expert Systems*, Communications of the ACM, Volume 31, Number 2, pp. 152-169.

[3] 橋田 浩一 (1989) 制約と言語 「ディスコースと形式意味論ワークショップ」論文集、ソフトウェア科学会「論理と自然言語研究会」

[4] Rumelhart,D.E., and McClelland,J.E.,Eds.(1986) *Parallel Distributed Processing* Vol.1, MIT Press/Bradford Books, Cambridge, Mass.

[5] Rumelhart,D.E., and McClelland,J.E.,Eds. (1988) *Parallel Distributed Processing* Vol.3, MIT Press/Bradford Books, Cambridge, Mass.

[6] Touretzky,D.S., and Hinton,G.E., (1985) *Symbols Among the Neuron: Details of a Connectionist Inference Architecture*, Proceedings of International Joint Conference of Artificial Intelligence 1985.

[7] Waltz,D.L., and Pollack,J.B., (1985) *Massively Parallel Parsing: A Strong Interacive Model of Natural Language Interpretation*, *Cognitive Science* 9, pp. 51-74.