バックプロパゲーションアルゴリズムのニューラルネットワークモデル

Rudolf Jakša
Department of Cybernetics and Artificial Intelligence
Technical University of Košice
Letná 9, 041 20 Košice
Slovakia
jaksa@neuron.tuke.sk

Miroslav Katrák
Department of Cybernetics and Artificial Intelligence
Technical University of Košice
Letná 9, 041 20 Košice
Slovakia
bracek@mizu.sk

Abstract

ニューラルネットワークの学習アルゴリズムそのものをモデル化するために、ニューラルネットワークを適用する。ニューラルネットワークにおける重みの更新過程を観察し、ファイルに保存する。その後、このデータを使って別のネットワークを学習させ、学習させたアルゴリズムを真似てニューラルネットワークを学習させる。バックプロパゲーション・アルゴリズムは、学習と学習過程のサンプリングの両方に使用される。ネットワーク全体の訓練を模倣する。重みの相互依存性をモデル化するために、多層ニューラルネットワークのすべての重みと重みの変化を並列に処理する。実験結果を示す。

キーワード：メタラーニング、学習するための学習、誤差バックプロパゲーション。

1 Introduction

適応学習や最適化学習のアルゴリズムは、ニューラルネットワーク学習や機械学習の領域で使われるかもしれない。固定的な学習アルゴリズムの代わりに、これらのアルゴリズムは時間と共に自身の学習性能を向上させるか、あるいは特定の学習方法をゼロから開発する。このタイプの学習アルゴリズムは、「メタラーニング」または「学習する学習」アプローチとして知られている。Jürgen SchmidhuberとSepp Hochreiterによる研究[2] [3]は、この分野における最近の研究の代表的なものであり、Sebastian Thrunによる[4]では、より包括的な概要が述べられている。Thrunは、学習することを、アルゴリズムが前のタスクの経験を活かして次のタスクのパフォーマンスを向上させる能力と定義している[4]。Schmidhuberは、学習者が学習方法や学習過程を評価・比較し、この評価を用いて適切な学習戦略を選択する能力を重視している[2]。

メタラーニングのアプローチには、以下のようなパラダイムがある：

- 類似性の利用、
- 学習パラメータの適応、
- 学習アルゴリズムの発見。

特定の手法は、これらのパラダイムのいずれかに焦点を当てるかもしれないし、これらすべてに焦点を当てるかもしれない。

類似性の利用とは、タスクのグループには何らかの類似性があり、それを学習することで、別のタスクの学習を加速させることができるという考え方である。単純に類似性を利用するためにタスクの順序を学習することもできるが、タスクに特化した知識とタスク間に共通する知識を区別するメカニズムがあれば、パフォーマンスが向上するはずである。

学習パラメータの適応は、学習アルゴリズムの上位にあるメタ学習アルゴリズムによって行われるかもしれない。このパラダイムは、「学習するための学習 」ではなく、「学習についての学習 」に基づいている。しかし、学習に関する知識は、「学習するための学習」へのステップである。

学習アルゴリズムの発見とは、学習アルゴリズムをゼロから設計することである。これは「学習するための学習」ではなく、「学習を学習するための学習」である。ここでは、適応から学習へとフォーカスを移す。

メタラーニング・アルゴリズムは、強化学習または教師あり 学習に基づいている。強化学習の場合、学習者は試行錯誤の経験を通じて、特定のタスクに対す るパフォーマンスだけでなく、学習能力も向上させる。これは、学習アルゴリズムを解決されたタスクの一部として扱うことで達成できる。つまり、学習は学習者の行動の1つである。教師あり学習のシナリオでは、通常、学習とメタ学習は独立したプロセスとして扱われる。

2 バックプロパゲーションの模倣

このセクションでは、誤差バックプロパゲーションアルゴリズムのモデル化について述べる。我々はニューラルネットワークを訓練して、別のニューラルネットワークを訓練したい。誤差バックプロパゲーショ ンアルゴリズムのニューラルネットワークモデルは、オリジナルのバックプロパゲーションアルゴリズムと同じようにニューラルネットワークを学習できる必要がある。そのようなモデルを得るために、バックプロパゲーション学習の学習プロセスをサンプリングし、それを模倣してみる。これは単純でありながら、金属学習に対する一般的なアプローチでもある。これは次のような順序で行われる：

1.誤差バックプロパゲーションアルゴリズムで任意のニューラルネットワークを学習させ、学習過程をサンプリングする、

元の学習アルゴリズムを模倣するために、学習ネットワークを訓練する、

3.学習ネットワークを用いて任意のニューラルネットワークを学習する。

ニューロン活性化 x_i 、リンク重み w_{ij} 、バイアス θ_i 、ニューロン活性化関数 $f_i(in_i)$ を持つ多層ニューラルネットワークを考える：

$$x_i = f_i(in_i) \qquad in_i = \sum_{j=1}^{M} w_{ij}x_j + \theta_i \qquad (1)$$

in_iはi番目のニューロンへの入力であり、Mはi番目のニューロンへ接続するリンクの数である。教師あり学習モードにおける誤差Jは以下のように定義される：

$$J^p = \frac{1}{2}\sum_{i=1}^{N_0}(ev_i^p - x_i^p)^2 \qquad (2)$$

pはデータパターンのインデックス、N_0はニューラルネットワークの出力ニューロン数、ev_i^pはp番目のパターンに対するi番目のニューロンの期待出力である。簡単のため、パターンインデックスpは省略する。勾配に基づく誤差最小化適応は以下のようになる：

$$\Delta w_{ij} = -\gamma\frac{\partial J}{\partial w_{ij}} = -\gamma\frac{\partial J}{\partial in_i}\frac{\partial in_i}{\partial w_{ij}} = \gamma\delta_i x_j \qquad (3)$$

重みw_{ij}はj番目のニューロンからi番目のニューロンへのリンク、γは学習率定数、δ_iは以下のように定義される：

$$\delta_i = -\frac{\partial J}{\partial in_i} = -\frac{\partial J}{\partial x_i}\frac{\partial x_i}{\partial in_i} = -\frac{\partial J}{\partial x_i}f'(in_i) \qquad (4)$$

$f'(in_i)$ は活性化関数 $f(in_i)$ の微分である。出力ニューロンでは次のようになる：

$$\delta_i = -\frac{\partial J}{\partial x_i}f'(in_i) = (ev_i - x_i)f'(in_i) \qquad (5)$$

隠れ層のニューロンでは次が得られる：

$$\delta_i = -f'(in_i)\sum_{h=1}^{N_h}\frac{\partial J}{\partial in_h}\frac{\partial in_h}{\partial x_i} =$$

$$= -f'(in_i)\sum_{h=1}^{N_h}\frac{\partial J}{\partial in_h}\frac{\partial}{\partial x_i}\sum_{l=1}^{N_l}w_{hl}x_l =$$

$$= -f'(in_i)\sum_{h=1}^{N_h}\frac{\partial J}{\partial in_h}w_{hi} = f'(in_i)\sum_{h=1}^{N_h}\delta_h w_{hi} \qquad (6)$$

N_h はi番目のニューロンから来るリンクの数であり、hはこれらのリンクと対応するニューロンのインデックスである。N_lはh個のニューロンに接続するニューロンの数である（図1参照）。ルール(6)は誤差逆伝播ルールであり、ネットワークを介した誤差の逆伝播を定義する。ルール(3)はこの誤差を最小化する重みの変化を定義し、ルール(5)は誤差最小化のベースを設定する。

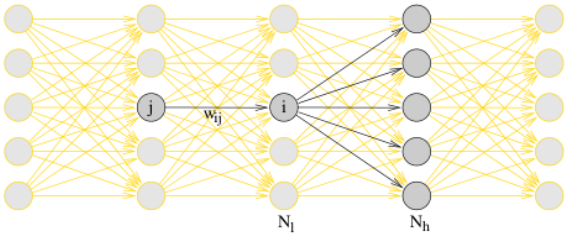

図1：規則(6)のニューロン指標。

誤差逆伝播アルゴリズムは、ルール(1)、(2)、(3)、(5)、(6)によって定義される。このアルゴリズムをモデル化するために、変数¤w, w, θ, δ, x, ev, J, in, f′(in)をサンプリングすることができる。これらの変数のいくつかは他の変数から導出できるので、完全なセットは必要ない。我々は、ルール(6)、またはルールのフルセットのいずれかをモデル化できる。ルール一式をモデル化する場合、ネットワーク全体の相互作用をモデルで処理することができる。ルール(6)のみをモデル化する場合、特定のリンクの近傍のみが考慮される。

学習ネットワークの入力と出力の数は、サンプリングされた変数の数に等しい。出力は¤wが変化し、¤θが変化する可能性がある。学習ネットワークを使用するには、元のバックプロパゲーション・アルゴリズムのルール(6)、(5)、(3)をこの学習ネットワークの出力に置き換える必要がある。

3 Experiments

2つの入力、1つの隠れニューロン、1つの出力を持つ図2のニューラルネットワークを考えよう。これには3つの重みと2つのバイアスがあり、これらの変化を学習ネットワークで近似しようとする。そこで、バックプロパゲーション・アルゴリズムで学習しながら、これらの変化をサンプリングする。そして、学習ネットワークを訓練して、これらの変化を近似する。これらの変化以外に、3つの重みと2つのバイアス、2つの入力、1つの出力、そして出力に対する期待値をすべてサンプルする。これは 学習ネットワークの入力は9個、出力は5個である。このような2つの隠れニューロンを持つ学習ネットワークを図3に示す。隠れニューロンの数は任意であり、学習するタスクや元の学習アルゴリズム（我々の場合は誤差バックプロパゲーション）の複雑さに依存する。

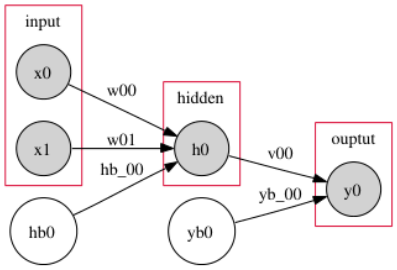

図2：学習する単純なネットワーク。x_0 ， x_1 は入力、 y_0 は出力、 w_{00} ， w_{01} ， v_{00} は重み、 hb_{00} ， yb_{00} はバイアス、 h_0 は隠れニューロンの活性化、 y_0 は出力である。

h_0は隠れニューロンの活性化、y_0は出力。 # 1つ目の実験では、図2のネットワークを訓練して、ブーリアン関数AND（表1）を近似する。これは単純なタスクであり、ネットワークはすぐに学習する。基本ネットワークの学習パラメータは以下の通りである：

γ=0.23、学習サイクル数は5000。学習ネットワークの学習パラメータは ネットワークのトポロジーは図3と同じである。

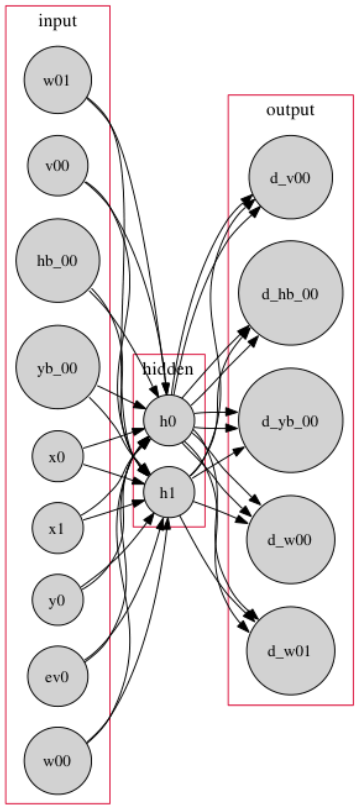

図3: 図2のネットワークを学習するための、2つの隠れニューロンを持つ学習ネットワーク。入力は図2のニューラルネットワークの状態を記述する変数であり、出力は学習アルゴリズムによって提供されるそれらの変化である。

AND

x_0	x_1	y_0
0	0	0
0	1	0
1	0	0
1	1	1

OR

x_0	x_1	y_0
0	0	0
0	1	1
1	0	1
1	1	1

表1：図2のネットワークのブールAND関数とOR関数の学習データ。

学習ネットワークの学習履歴はFig.4. バックプロパゲーション・アルゴリズムを用いた基本ネットワークの学習と、学習ネットワークを用いた学習の比較を図5に示す。学習ネットワークはオリジナルのバックプロパゲーション・アルゴリズムよりも収束性が高い。しかし、学習ネットワークの性能は学習にも依存し、オーバーフィッティングを起こしやすい。また、学習ネットワークと誤差バックプロパゲーション・アルゴリズムを用いた学習の実装は、速度が異なる可能性があることに注意されたい。我々の実験の場合、バックプロパゲーションの学習には0.408秒かかり、学習ネットワークの学習には3.781秒かかった。

第2実験では、隠れ層のない基本的なネットワークを使用する。これはAND関数の近似には十分である。基本ネットワークでは入力2、出力1、隠れニューロン0、学習ネットワークでは入力7、出力3、隠れニューロン0とする。この学習ネットワークの学習履歴を図6に示す。図7は、バックプロパゲーション・アルゴリズムを用いた隠れニューロンのない基本ネットワークと、学習ネットワークを用いた学習の比較である。このセットアップにおける学習ネットワークの性能は、バックプロパゲーション・アルゴリズムの性能と同等であるが、隠れニューロンを用いた1回目の実験よりも若干悪い。

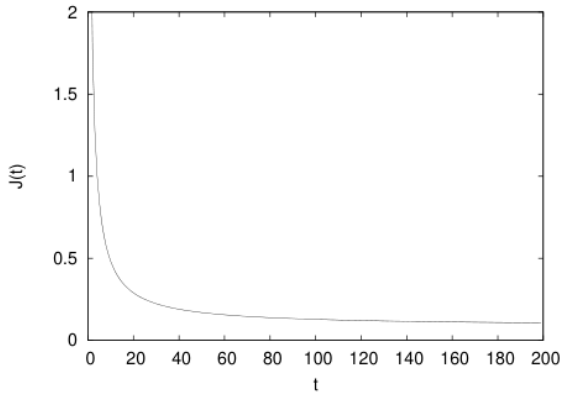

Figure 4: Fig.2の基本ネットワークに対する学習ネットワークの学習誤差。

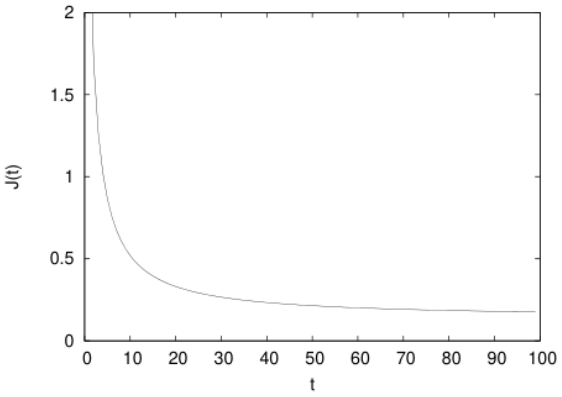

図6: 隠れニューロンを含まない基本ネットワークに対する学習ネットワークの学習誤差。

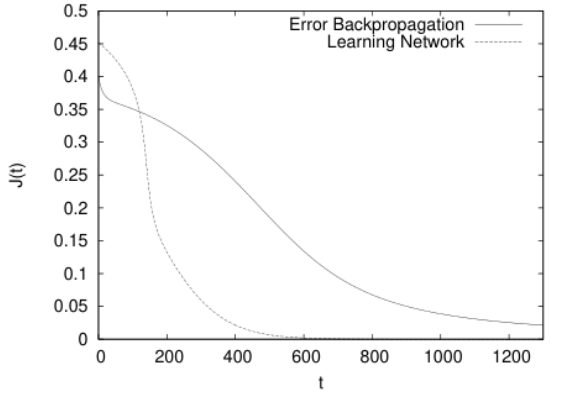

図5: 誤差バックプロパゲーションアルゴリズムを用いた図2の基本ネットワークの学習と、図3の学習ネットワークを用いた場合の誤差。

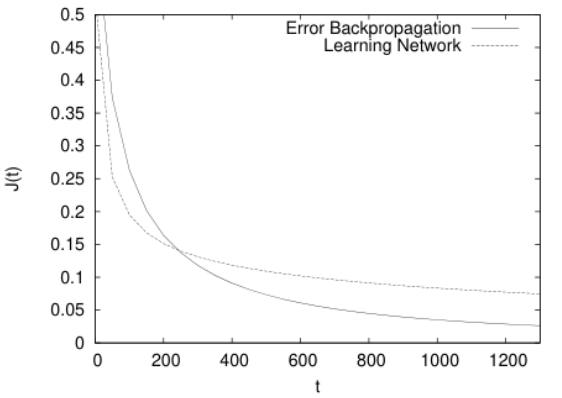

図7：バックプロパゲーション・アルゴリズムと学習ネットワークを用いた、隠れニューロンを含まない基本ネットワークの学習誤差。

第3の実験では、隠れニューロンの数を増やす。基本ネットワークでは入力2、出力1、隠れニューロン4、学習ネットワークでは入力21、出力17、隠れニューロン3とする。タスクはOR関数の近似である。この学習ネットワークの学習履歴は図8の通りである。バックプロパゲーション・アルゴリズムを用いた隠れニューロンのない基本ネットワークの学習と、学習ネットワークを用いた学習の比較は図9の通りである。このセットアップにおける学習ネットワークの性能は、元の誤差バックプロパゲーション・アルゴリズムよりも再び良くなっている。

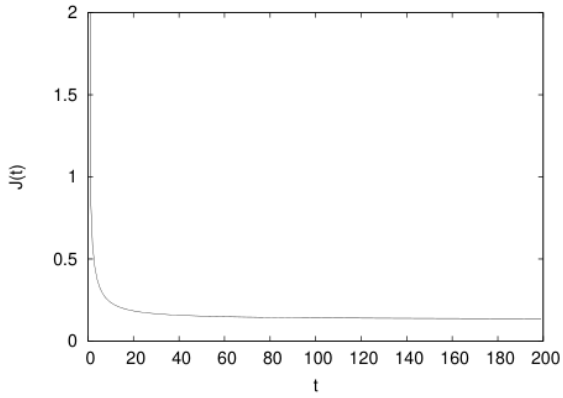

図8: 隠れニューロンを4つ持つ基本ネットワークに対する学習ネットワークの学習誤差。

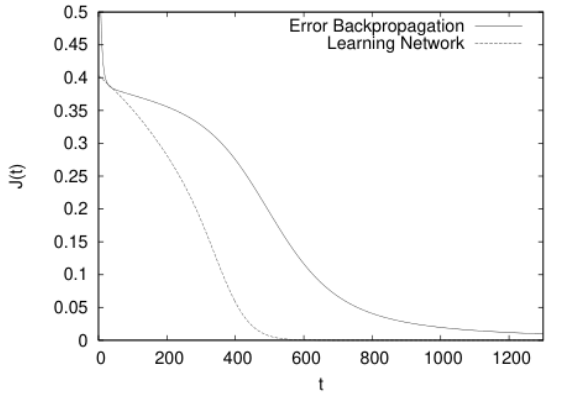

図9: 4つの隠れニューロンを持つ基本ネットワークの、誤差バックプロパゲーションアルゴリズムを用いた学習と、学習ネットワークを用いた学習の誤差。

第4の実験では、より難しい分類課題に挑戦する。図12は訓練とテストのデータセットである。ネットワークのトポロジーは以下の通りである：

基本ネットワークは2入力：xとy、2出力：innersquareとoutersquare、5隠れニューロン、学習ネットワークは38入力、27出力、3隠れニューロンである。この学習ネットワークの学習履歴を図10に示す。 バックプロパゲーション・アルゴリズムを用いた基本ネットワークの学習と、学習ネットワークを用いた学習の比較を図11に示す。 この課題では、学習ネットワークを用いた学習は発散しているが、バックプロパゲーション・アルゴリズムを用いた学習は、あるレベルで停止しているが、発散していない。

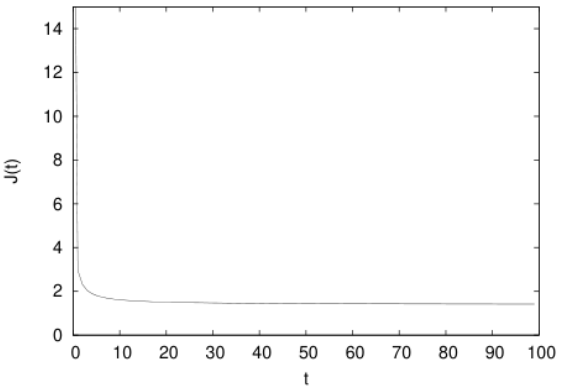

図10: 正方形分類タスクに対する基本ネットワークの学習誤差。

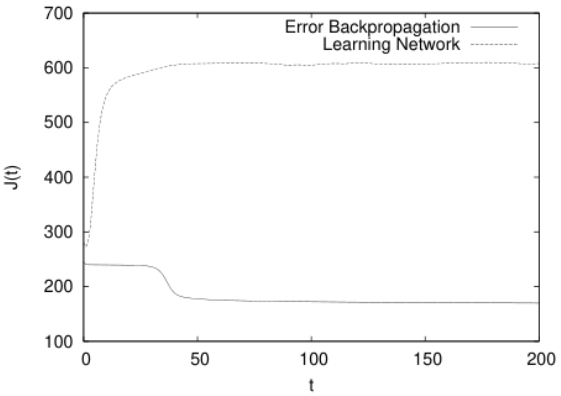

図11: エラー・バックプロパゲーション・アルゴリズムと学習ネットワークを使用した、正方形分類タスクに対する基本ネットワークの学習の誤差。

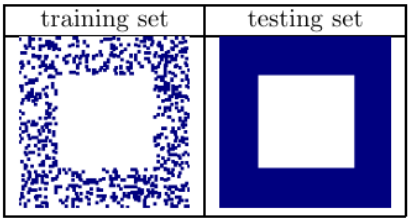

図12：分類タスクのトレーニングセットとテストセット。このタスクは、空間上の点をx座標とy座標によって分類し、それらが内側の正方形に収まるかどうかを判定するものである。

4 Analysis

隠れユニットを用いたAND／OR関数近似タスクは、学習ネットワークで学習させると良い結果を示す。学習ネットワークがバックプロパゲーションアルゴリズムを凌駕する能力は有望であると思われる。将来的には、複数の学習アルゴリズムの知識を学習ネットワークの学習に利用し、これら全てのアルゴリズムの長所を利用することで、より優れた性能を得ることができるかもしれない。試行錯誤モードは、学習ネットワークを訓練する際にさらに使用されるかもしれない。

隠れユニットを用いない実験では、バックプロパゲーション・アルゴリズムの学習ルール、もしくは学習させたニューラルネットワークの非線形性が若干悪化している。

より複雑な分類タスクに我々のアプローチを適用する場合の問題は、過剰訓練された学習ネットワークの不安定性と同様の性格を持つかもしれない。学習ネットワークによる学習が不安定になる可能性は、このアプローチに固有の性質である。より単純なネットワークでより良いパフォーマンスを得るには、バックプロパゲーションアルゴリズムのルール(6)のみをモデル化することが有利である。バックプロパゲーショ ンアルゴリズムのニューラルネットワークモデリング、ルール(6)のモ デリング、アルゴリズムサンプリングのための変数セットのより深い 分析をさらに調査することが役立つかもしれない。

5 Conclusion

バックプロパゲーションアルゴリズムのニューラルネットワークモデルを使用したニューラルネットワークのトレーニングは、有効なアプローチである。

このモデルを使用することで、学習アルゴリズムのパフォーマンスチューニングの新しい方法が可能になる。しかし、このアプローチには学習が不安定になるリスクがあり、バックプロパゲーションの実際のモデリングは、いくつかの異なるモードで行うことができる。

References

[1] M.Katrák, Metalearning methods for neural networks, (in Slovak), MS Thesis, Technical University Košice, (2005).
neuron.tuke.sk/~jaksa/theses

[2] J.Schmidhuber, J.Zhao, and M.Wiering, Simple principles of metalearning, Technical Report IDSIA-69-96, IDSIA, (1996).
citeseer.ist.psu.edu/schmidhuber96simple.html

[3] S.Hochreiter, A.S.Younger, and P.R.Conwell, Learning to Learn Using Gradient Descent, Lecture Notes in Computer Science, vol.2130, (2001).
citeseer.ist.psu.edu/hochreiter01learning.html

[4] S.Thrun, Learning To Learn: Introduction.
citeseer.ist.psu.edu/article/thrun96learning.html

[5] J.Schmidhuber, Evolutionary Principles in Self-Referential Learning, Diploma Thesis, Technische Universität München, (1987).
www.idsia.ch/~juergen/diploma.html

[6] J.Schmidhuber, On Learning How to Learn Learning Strategies, Technical Report FKI-198-94, Fakultt für Informatik, Technische Universitt München, (1994).
citeseer.ist.psu.edu/schmidhuber95learning.html

[7] J.Schmidhuber, A General Method for Incremental Self-Improvement and Multi-agent Learning in Unrestricted Environments, In X.Yao (Ed.), Evolutionary Computation: Theory and Applications, Scientific Publ. Co., Singapore, (1996).
citeseer.ist.psu.edu/article/schmidhuber96general.html

[8] J.Schmidhuber, A Neural Network That Embeds Its Own Meta-Levels, In Proc. of the International Conference on Neural Networks '93, San Francisco. IEEE, (1993).
citeseer.ist.psu.edu/schmidhuber93neural.html