高階微分を学習するニューラル・ネットワーク

益岡 竜介 †‡
masuoka@flab.fujitsu.co.jp

山田 道夫†
yamadam@tansei.cc.u-tokyo.ac.jp

†東京大学大学院数理科学研究科
〒 153 東京都 目黒区 駒場 3-8-1

‡(株) 富士通研究所
〒 211 神奈川県 川崎市 中原区 上小田中 1015

ニューラルネットワークによるモデルの学習に関してそのデータ自身を入手する際に, 観測点を増やすことは難しいが, 一方で微分情報が値の情報とは独立に得られる場合がある. そのような場合には多少計算コストがかかっても得られた微分情報を使って学習することができれば, より精密なモデルを学習することができる.
本報告書では通常の入出力の値の関係に加えて, 入力に関する出力の一般の微分を学習することができるニューラルネットワークについて, その構造やアルゴリズム, その応用の可能性などを報告する.

和文キーワード: tangent prop, 微分情報, 多層パーセプトロン, MASCON

Neural Networks Which Learn From Differential Data

Ryusuke Masuoka †‡
masuoka@flab.fujitsu.co.jp

Michio Yamada †
yamadam@tansei.cc.u-tokyo.ac.jp

†Department of Mathematical Sciences
The University of Tokyo
3-8-1 Komaba, Meguro-KU
Tokyo, 153, Japan

‡FUJITSU LABORATORIES LTD.
1015 Kamikodanaka, Nakahara-KU
Kawasaki 211, Japan

There are some cases for modeling by neural networks, in which it is costly to get more data points, but in which differential data is available at less or no cost. In such cases it is reasonable to make neural networks learn from differential data along with usual value data at the sacrifice of high computational cost to obtain a more precise model.
We report, in this paper, the structure of and the learning algorithm for such neural networks of multilayer perceptron type which learn from differential data. We also present possibilities of their applications .

English Key Words: tangent prop, differential data, multilayer perceptron, MASCON

1 はじめに

ニューラルネットワークによるモデルの学習に関してそのデータ自身を入手する際に, 観測点を増やすことは難しいが, 一方で微分情報が値の情報とは独立に得られる場合がある. そのような場合には多少計算コストがかかっても得られた微分情報を使って学習することができれば, より精密なモデルを学習することができる.

上のような場合には, 例えば [3] において Mitchell and Thrun が導入した Explanation-Based Neural Network (EBNN) がある. その中では explanaion というものを一階の微分の情報としてとらえ, ニューラルネットワークでその微分の情報を学習させるということによりよりよいモデルを獲得することを行なった. この EBNN では適用対象の一つをロボット制御問題においており, そのような問題においてはやはり新しい観測点を得ることのコストが高くなっている.

あるいはパターン認識の問題で, 回転や平行移動による認識の不変性は, 出力の入力空間の特定の方向に関する微分がゼロであることによって表される [8].

また以下のような場合も考えられる. ある (線型) 偏微分方程式を満たす物理量に関して空間のいくつかの点に関する値が与えられる場合である. 物理量が直接与えられたそれらの点に関しては通常の multilayer perceptron の back propagation におけるエラー関数の最小化の枠組を用いる. 偏微分方程式 ("... = 0" の形をしているとする) で与えられる制約条件は, 例えばランダムに領域内の点を取り, それらの点での偏微分方程式の左辺の 2 乗の和を最小化するという条件に変えることにより, やはり通常の multilayer perceptron で使われる back propagation などの方法と同じような問題の枠組に持っていくことができる. 前者の通常のエラー関数とともに, 後者の 2 乗の和の微分情報に関しても back propagation することができれば, 両方の制約条件を (近似的に) 満たす解を見つけることができる.

本報告書では上記のような場合に通常の入出力の値の関係に加えて, 入力に関する出力の一般の階数の微分を学習することができる multilayer perceptron 型のニューラルネットワークについて, その構造やアルゴリズム, その応用の方法などを報告する.

出力の入力に関する一階の微分の multilayer perceptron 型のネットワークによる学習に関しては, すでに Simard らの tangent prop の研究がある [8]. 本報告書のネットワークは tangent prop を一般の階数の微分に拡張した形になっている. また一階の微分の学習に関しての noise robustness に関して調べた結果が [2] にある.

2 一般階数の微分情報の学習

以下に multilayer perceptron を拡張した微分情報を学習するニューラルネットワークの構造や, そのネットワークの forward propagation, back propagation の方法を報告する. 式の細かい導出などは省略する.

2.1 定義

以下の説明で主に用いる記号を説明する.

n: ネットワークへの入力の次元

$\delta = (a_1, ..., a_n) \in \{N \cup \{0\}\}^n$:

特別に $0 = (0, ..., 0)$ と定義する. また半順序 $>$ を以下で定義する.

$$\delta_1 = (a_1, ..., a_n) > \delta_2 = (b_1, ..., b_n)$$
$$\leftrightarrow$$
$$\exists i \quad a_i > b_i \ \wedge \ \forall i \quad a_i \geq b_i$$

また $\delta_1 + \delta_2$ や $c\delta$ などは通常の線型空間の定義とする. また $N(\delta)$ を $N(\delta) = \sum_{i=1}^{n} a_i$ として定義する. この δ は微分オペレータを表すものとして用いる. すなわち $\delta = (a_1, ..., a_n)$ に対して

$$\frac{\partial^{N(\delta)}}{\partial^{\delta} x} = \frac{\partial^{N(\delta)}}{\partial^{a_1} x_1 \ ... \ \partial^{a_n} x_n} \tag{1}$$

と定義し, δ に対して $\frac{\partial^{N(\delta)}}{\partial^\delta x}$ が対応するとして扱う.

$\Delta = \{(c_i, \delta_i) \mid c_i \in N, \delta_i > 0, i = 1, ..., m\}$:

この Δ に対して $N(\Delta) = m$ と定義する. さらに以下のように定義する. 後者の $\Delta(f)$ は, Δ を関数 f に関する汎関数として定義している.

$$\begin{aligned} Sum(\Delta) &= \sum_{i=1}^{m} c_i \times \delta_i \\ \Delta(f) &= \prod_{(c,\delta)\in\Delta} \left(\frac{\partial^{N(\delta)}}{\partial^\delta x} \right)^c \end{aligned}$$

$V(\delta) = \{(d_{\delta,\Delta}, \Delta) \mid \delta = Sum(\Delta)\}$:

ただし $d_{\delta,\Delta}$ は以下の式で決まる自然数である. f は入力空間から実数への任意の C^∞ 関数であるとする.

$$\frac{\partial^{N(\delta)}}{\partial^\delta x}\, e^f = \sum d_{\delta,\Delta}\ \Delta(f)\ e^f \tag{2}$$

2.2 問題の枠組

以下に問題の枠組を述べる. まず n 次元の入力空間と何次元か[1]の出力空間と, その入力空間から出力空間への何階か微分可能な写像が与えられている. さらに入力空間のいくつかの点と, それらの点における上の写像の出力空間における値, あるいはその写像の何階かの微分[2]の値, あるいはそれらの両方が与えられているとする[3].

それらの条件のもと, 与えられた値から元の写像をできるだけよく近似するネットワークを見つけようというものである. 通常ネットワークの構造は適当に固定して, 与えられた値からネットワーク内の内部状態 (結合の重みなど) を変更して, よりよく近似することが問題となる.

2.3 ネットワークの構造

図 2.3 は高階の微分を学習するためのネットワークの構造である. このネットワークは, 通常の multilayer perceptron のネットワークに対して, 微分情報を伝える部分を付加した形になっており, [8] の tangent prop に用いられた Jacobian net を拡張した形になっている. 図 2.3 は通常の multilayer perceptron のネットワーク中の一つのユニットとそのユニットに入力を持つユニットの部分を取りだし, その部分に対応する付加部分を示している. 図中左側に通常の multilayer perceptron の部分があり, それを value net と呼ぶ. value net に対して, それぞれの学習したい微分の微分オペレータ δ に対応したネットワーク δ net の部分が図中右側にある. δ net は基本的に value net の各ユニット $x^0 = x$ に対応した x^δ ユニットおよび y^δ ユニットからなる. また他に value net の各ユニットに対応した $\sigma^{(m)}$ ユニットがある. $\sigma^{(m)}$ ユニットは図中に示されていないが, value net からの入力を持っている. 結合の重み $w_{i,j}$ などは value net とその対応する δ net たちの間で同じものが用いられる. (図中および以下では, 間違いの可能性がない場合に簡単のため $x^0_{l,i}$ を $x_{l,i}$, $net \triangleright x^0_{l,i}$ を $net_{l,i}$ などとして表す.)

各ユニットの出力は, ユニットの名前と同じ記号で表される. また各ユニットへの入力をユニットの記号の前に "$net \triangleright$" をつけて表す. 例えば δ-net の第 l 層の第 i 番目の $x^\delta_{l,i}$ ユニットへの入力は $net \triangleright x^\delta_{l,i}$ であり, 出力は $x^\delta_{l,i}$ である.

また back propagation されるものは [6] では δ の記号が使われたが, ここでは区別のため, ϵ を用いる. 各ユニットへ back propagation されるものはユニットの記号の前に "$\epsilon \triangleright$" をつけて表す. 例えば δ-net の第 l 層の第 i 番目の $x^\delta_{l,i}$ ユニットへ back propagation されるものは $\epsilon \triangleright x^\delta_{l,i}$ と表される.

---

[1] 以下の取り扱いにおいて, 一般性を失うことなく出力空間は一次元とする.

[2] 必ずしもその入力空間の軸に沿った微分でなくてもよい.

[3] 各点ごとに与えられる値の種類や数は異なってもよい. またそれらの値にノイズが含まれてもよいとする.

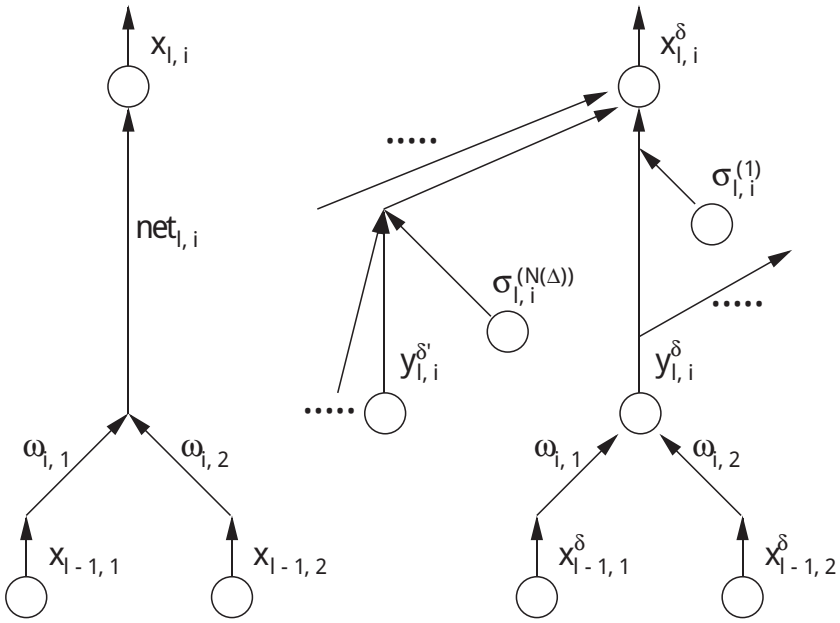

図 1: 微分を学習するネットワークの構造: 図中左側に通常の multilayer perceptron の部分があり, それを value net と呼ぶ. value net に対して, それぞれの学習したい微分の微分オペレータ δ に対応したネットワーク δ net の部分が図中右側にある. δ net は基本的に value net の各ユニット $x^0 = x$ に対応した x^δ ユニットおよび y^δ ユニットからなる. また他に value net の各ユニットに対応した $\sigma^{(m)}$ ユニットがある. $\sigma^{(m)}$ ユニットは図中に示されていないが, value net からの入力を持っている. 結合の重み $w_{i,j}$ などは value net とその対応する δ net たちの間で同じものが用いられる. (図中および以下では, 間違いの可能性がない場合に簡単のため $x^0_{l,i}$ を $x_{l,i}$, $net \triangleright x^0_{l,i}$ を $net_{l,i}$ などとして表す.)

2.4 ネットワークに与えるデータ

以下にネットワークに与えるデータの形を示す. ネットワークに与えるデータの微分の方向は必ずしも軸に沿っている必要はないが, そういったデータを扱うためには本質的ではないがさらに構造を必要とし, 説明が複雑になるので省略する. ネットワークに与えるデータはいくつかの入力と出力が組になったもの (それをパターンと呼ぶ) からなる.

ネットワークに与える入力のデータは $I = (I^0, ..., I^\delta, ...)$ の形をしている. ただし I^0 はそのデータがとられた入力空間における座標であり, 入力空間が 2 次元以上なら複数の実数の組からなる. その他の I^δ は単なる実数であり, 特に $\{0.0, 1.0\}$ の元である. もし $I^\delta = 1.0$ であれば, $0 < \delta' < \delta$ なる δ' に対して, $I^{\delta'} = 1.0$ である[4].

出力の学習データは $O = (O^0, ..., O^\delta, ...)$ の形をしている. これらはすべて出力空間の元, あるいは "$*$" である. $O^\delta = *$ であるということは, その δ に対応する微分の値が与えられていないということを表し, その際にはその δ に関しては back propagation をしない. 出力の学習データの方では $O^\delta \neq *$ なる δ に関して $0 < \delta' < \delta$ なる δ' に対して, $O^{\delta'} = *$ であっても構わない.

2.5 Forward Propagation

まず forward propagation のアルゴリズムを与える. ネットワークに入力 $I = (I^0, ..., I^\delta, ...)$ が与えられた時に, ネットワーク中をどのように forward propagation されるかを以下に説明する.

以下において l は層の番号である. しかしネットワークに層構造があることは本質的ではない. わかりやすいように層構造を使って説明しているだけである. 層構造がなくてもアルゴリズムは基本的に同じであ

[4] これは単に表現上の問題で, そうしておかないと以下の forward propagation や back propagation の説明が複雑になるためにそうしている.

る.

2.5.1 Value Net のユニット

value net には特に y^0 ユニットは存在しないが, $y^0_{l,i} = net \triangleright y^0_{l,i} = net \triangleright x^0_{l,i}$ としておく. これはそうしておくことにより, value net と δ net の記述の整合性がとれるからである.

value net のユニットに関する forward propagation は以下のようになる. (以下で σ は value net のユニットの smashing 関数である. 本報告書ではすべてのユニットに対して同じ σ を用いているが, 各ユニットごとに違っていても, 全く同様にアルゴリズムは記述される.)

$$net \triangleright x^0_{l,i} = net_{l,i} = y^0_{l,i} = \sum_{j \in P(i)} w_{ij} x_{l-1,j} \tag{3}$$

$$x^0_{l,i} = x_{l,i} = \sigma(net_{l,i}) \tag{4}$$

2.5.2 δ Net のユニット

$\sigma^{(m)}_{l,i}$ ユニットの入出力は次で与えられる. (以下で $\sigma^{(m)}$ は関数 σ の m 階の微分である.)

$$net \triangleright \sigma^{(m)}_{l,i} = net_{l,i} \tag{5}$$

$$\sigma^{(m)}_{l,i} = \sigma^{(m)}(net_{l,i}) \tag{6}$$

$y^\delta_{l,i}$ ユニットの入出力は次で与えられる. (このユニットに関し入力と出力は同じである.)

$$y^\delta_{l,i} = net \triangleright y^\delta_{l,i} = \sum_j w_{i,j} x^\delta_{l-1,j} \tag{7}$$

その時 $x^\delta_{l,i}$ ユニットの入出力は次で与えられる. この x^δ ユニットたちは [6] でいう Sigma-Pi ユニットになる. (このユニットに関し入力と出力は同じである.)

$$x^\delta_{l,i} = net \triangleright x^\delta_{l,i} = net^\delta_{l,i} = \frac{\partial^{N(\delta)}}{\partial x_\delta}(x_{l,i}) \tag{8}$$

$$= \sum_{(d_{\delta,\Delta},\Delta) \in V(\delta)} d_{\delta,\Delta} \left\{ \sigma^{(N(\Delta))}_{l,i} \prod_{(c,\delta') \in \Delta} \left(y^{\delta'}_{l,i} \right)^c \right\} \tag{9}$$

上の式で現れた Δ に関する積は, back propagation の時にも用いられるので, それを以下のように置き, とっておくようにする.

$$p^\Delta_{l,i} = \prod_{(c,\delta') \in \Delta} \left(y^{\delta'}_{l,i} \right)^c \tag{10}$$

2.6 エラー関数

エラー関数は以下のような和とする.

$$E = \sum_{\delta \geq 0} \alpha^\delta E^\delta \tag{11}$$

上の式で α^δ は δ に対応した学習定数である.

各 E^δ は, 与えられたパターン p, (I_p, O_p) に対して以下のように与えられる. まず入力 I_p をネットワークに与え, ネットワークを forward propagation する. そしてネットワークの出力と与えられた出力 (後者を教師信号と呼ぶことがある) を使って以下のように定義される. (以下で繁雑になるので x にはパターンの添字しかつけない.)

$$E^\delta = \frac{1}{2} \sum_p \left(O^\delta_p - x^\delta_p \right)^2 \tag{12}$$

この E を使って, 各 $w_{i,j}$ は以下のように更新される.

$$\Delta w_{i,j} = -\frac{\partial E}{\partial w_{i,j}} \tag{13}$$

通常 back propagation では学習定数をエネルギーの中に含まず, 結合の重みを更新する際にその更新量にかけるようにしている [6] . このアルゴリズムでは, エネルギーの中にすでに学習定数が組み込まれている. これは微分 (δ) ごとに学習定数があるので, 更新量が決まってから学習定数をかけるようにするには, 微分 (δ) ごとに更新量を管理しなくてはいけなくなり, 記憶容量などが余分に必要になるからである. 1 階の微分程度なら更新量などを分けて管理することもできるが, それより高階になると難しいし, また分けることにそれほど意味がない. したがって以下のアルゴリズムでは, 出力層のユニットに ϵ を与える段階で学習定数をかけておく形になっている.

まず以下でこの更新量を計算するために, 各ユニットごとに back propagation される ϵ の back propagation のルールを示した後に, 結合の重みの更新ルールを示す. なお以下では簡単のためにパターンは一つだけとし, パターンの添字は基本的に省略する.

2.7 Back Propagation

back propagation の対象とするものは, [6] で用いられたものに対応させる. すなわち, そのユニットへの入力によるエラー関数の微分とする. エラー関数を E として, ユニット u とすれば, そのユニットに関する back propagation の対象は $\epsilon \triangleright u$ と表され, その定義は以下のようになる.

$$\epsilon \triangleright u = -\frac{\partial E}{\partial net \triangleright u} \tag{14}$$

以下に与えられたパターン (I_p, O_p) に対する back propagation のアルゴリズムを記述する. すでに入力 I_p をネットワークに与え, ネットワークを forward propagation しているとする.

2.7.1 出力層のユニット

従って, 出力層の x^δ units には, 以下のように ϵ を設定する. (以下で層の番号とユニットの番号は繁雑になるので省略されている.)

$\delta = 0$ (すなわち value net) の場合には以下のようになる.

$$\epsilon \triangleright x^0 = -\frac{\partial E}{\partial net \triangleright x^0} = \alpha^0 \left(O^0 - x^0\right) \sigma^{(1)}(net) \tag{15}$$

$\delta > 0$ の場合には以下のようになる.

$$\epsilon \triangleright x^\delta = -\frac{\partial E}{\partial net \triangleright x^\delta} = \alpha^\delta \left(O^\delta - x^\delta\right) \tag{16}$$

2.7.2 Value Net のユニット

value net に関しての back propagation は, 通常の multilayer perceptron の場合に, $\sigma^{(m)}$ ユニットからの ϵ を加えたものとなる.

$$\epsilon \triangleright x^0_{l-1,j} = \epsilon^0_{l-1,j} = -\frac{\partial E}{\partial net_{l-1,j}} \tag{17}$$

$$= \sum_i \left(\epsilon_{l,i} \sigma^{(1)}(net_{l-1,j}) + \sum_{m \geq 1} \epsilon \triangleright \sigma^{(m)}_{l,i} \right) \omega_{i,j} \tag{18}$$

ちなみに $net \triangleright x^0_{l,i} = net \triangleright y^0_{l,i} = net_{l,i}$ であるから以下のようになる.

$$\epsilon \triangleright x^0_{l,i} = \epsilon \triangleright y^0_{l,i} = -\frac{\partial E}{\partial net_{l,i}} \tag{19}$$

2.7.3 δ Net のユニット

$x^{\delta}_{l-1,j}$ ユニット, $y^{\delta}_{l,i}$ ユニット, $\sigma^{(m)}_{l,i}$ ユニットたちに関する back propagation は以下のようになる.

$$\epsilon^{\delta}_{l-1,j} = \epsilon \triangleright x^{\delta}_{l-1,j} = \sum_{i} \epsilon \triangleright y^{\delta}_{l,i}\ \omega_{i,j} \tag{20}$$

$$\epsilon \triangleright \sigma^{(m)}_{l,i} = \sum_{\delta:N(\delta)\geq m} \epsilon^{\delta}_{l,i} \left(\sum_{\Delta:N(\Delta)=m\ \wedge\ Sum(\Delta)=\delta} p^{\Delta} \right) \sigma^{(m+1)}_{l,i} \tag{21}$$

$$\epsilon \triangleright y^{\delta}_{l,i} = \sum_{\delta':\delta'\geq\delta} \epsilon^{\delta'}_{l,i} \sum_{\Delta:Sum(\Delta)=\delta'\wedge\exists c\,(c,\delta)\in\Delta} \sigma^{(N(\Delta))}_{l,i} \prod_{\delta''\in(\Delta-\{\delta\})} y^{\delta''}_{l,i} \tag{22}$$

2.8 重みの更新ルール

以下にネットワーク中の結合の重みの更新ルールを示す. 図 2.3 にあるように同じ結合の重み $w_{i,j}$ が value net と δ net の対応したところに現れているので, 更新量はそれらに対する更新量を加え合わせたものになる. 章 2.7 で得られた各ユニットの ϵ を用いると各結合の重み $w_{i,j}$ の update rules は以下のようになる.

$$\Delta w_{i,j} = -\frac{\partial E}{\partial w_{i,j}} \tag{23}$$

$$= -\frac{\partial E}{\partial net_i}\frac{net_i}{\partial w_{i,j}} - \sum_{\delta>0}\frac{\partial E}{\partial net \triangleright y^{\delta}_{l,i}}\frac{\partial net \triangleright y^{\delta}_{l,i}}{\partial w_{i,j}} - \sum_{m\geq 1}\frac{\partial E}{\partial net \triangleright \sigma^{(m)}_{l,i}}\frac{\partial net \triangleright \sigma^{(m)}_{l,i}}{\partial w_{i,j}} \tag{24}$$

$$= \epsilon \triangleright x_{l,i}\ x_{l-1,j} + \sum_{\delta>0} \epsilon \triangleright y^{\delta}_{l,i}\ x^{\delta}_{l-1,j} + \sum_{m\geq 1} \epsilon \triangleright \sigma^{(m)}_{l,i}\ x^{0}_{l-1,j} \tag{25}$$

あるいは value net も含めて, $\sum_{\delta}$ に $\delta = 0$ の場合も含めると, $(\epsilon \triangleright x^{0}_{l,i} = \epsilon \triangleright y^{0}_{l,i}$ であるから) 以下のように簡単に表される.

$$\Delta w_{i,j} = \sum_{\delta\geq 0} \epsilon \triangleright y^{\delta}_{l,i}\ x^{\delta}_{l-1,j} + \sum_{m\geq 1} \epsilon \triangleright \sigma^{(m)}_{l,i}\ x^{0}_{l-1,j} \tag{26}$$

3 応用の可能性

導入部で微分情報を学習できるネットワークの応用例として, Mitchell and Thrun が導入した EBNN [3] やパターン認識の問題 [8] に簡単に触れたが, ここでは偏微分方程式を満たす物理量に応用する場合を気象学からの例をもとに考察してみる.

気象学の方では, 風速場の (比較的) 少ない観測点 (数個から数百程度) の情報からそのグローバルな情報を求める MASCON (Mass-Consistent) という方法が使われている [7], [1]. 本当は風速場に関する完全な流体の方程式を解く必要があるのだが, この方法では観測点における観測データによる拘束条件とその他の点で以下のような質量保存の法則を満たすことだけを条件に全体の状態を求める.

$$\nabla f(x) = 0 \tag{27}$$

観測点 $\{x_i\}$ における値を $\{y_i\}$ (値はベクトル) とする. $f_{ob}(x_i) = y_i$ が成り立つように適当に関数 f_{ob} を定めておいて, 解となるべき f を次の値を最小化するものとして求める.

$$\int (f(x) - f_{ob}(x))^2 dx + \int \lambda(x)\nabla f(x) dx \tag{28}$$

この問題に我々のネットワークを応用するには,

$$\sum (f(x_i) - y_i)^2 + \int (\nabla f)^2 dx \tag{29}$$

を最小化するような f を multilayer perceptron 型のネットワークで表されるような範囲の関数から求めるという問題に直す. 前者の項に関しては通常の multilayer perceptron における back propagation と同等となるので本報告書のネットワークの value net の部分を用いて, 前者の項を E^0 とおく. 後者の項に関しては対象領域中の点を (例えば) ランダムにとってきて近似するような方法が考えられる. 後者の項を近似したそれらの点に関する和を E^δ として本報告書のネットワークの δ net の部分を用いる.

4 まとめ

本報告書では, データを入手する際に観測点を増やすことは難しいが, 一方で微分情報が値の情報とは独立に得られる場合などに計算コストはかかるが, より正確な学習を可能とする, 微分情報を使って学習することができる multilayer perceptron 型のネットワークについて報告した. そのようなネットワークを使ってもらえるようにネットワークの構造やアルゴリズムを特に詳しく報告した. また応用の可能性として, 気象学の MASCON という手法に応用する際の方法を考察した.

これからの課題としては, このネットワークとアルゴリズムを実際に計算機にプログラムとしてインプリメントし, 具体的な問題に適用して, その効果などを調べることがある. 現在プログラムを C++ で書きつつあり, 3,000 行程度になり核の部分はほとんど完成している. 次回には実験の結果を報告できると考えている.

また他の課題としては, radial basis function (RBF) に関する Poggio and Girosi ([5], [4]) などの取り扱いを, 本報告書の multilayer perceptron の場合と同様の問題の枠組で, 一般の微分の情報を含む場合に拡張することがある.

参考文献

[1] M. H. Dickerson. MASCON-mass consistent atmospheric flux model for regions with complex terrain. *J. Appl. Meteor.*, Vol. 17, pp. 241–253, 1978.

[2] Ryusuke Masuoka. Noise robustness of ebnn learning. In *Proceedings of 1993 International Joint Conference on Neural Networks*, volume 2, pp. 1665–1668. IEEE, 1993.

[3] Thomas M. Mitchell and Sebastian Thrun. Explanation-based neural network learning for robot control. In Stephen J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5*, pp. 287–294. Morgan Kaufmann, San Mateo, CA, 1993.

[4] Tomaso Poggio and Federico Girosi. Networks and the best approximation property. *Bilogical Cybernetics*, Vol. 63, pp. 169–176, 1990.

[5] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481–1497, september 1990.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II*, pp. 318–362. MIT Press, 1986.

[7] Y. K. Sasaki. *Lecture notes on variational methods for environmental analysis and prediction problems, Severe Storm Research Notes.1.* Disaster Prevention Research Institute, Kyoto University, 1979.

[8] Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pp. 895–903. Morgan Kaufmann, San Mateo, CA, 1992.