すざく衛星搭載宇宙硬X線検出器の機械学習によるバックグラウンド強度推定法の開発

B144190
森 和希

広島大学理学部物理科学科
高エネルギー宇宙・可視赤外天文研究室

指導教員：　深澤 泰司　教授
主査：　深澤 泰司　教授
副査：　山本 一博　准教授

2018年2月9日

概 要

X 線観測衛星「すざく」に搭載されている観測装置の１つに硬 X 線検出器 (HXD) がある。HXD は徹底されたバックグラウンドの削減により優れた感度を持つ検出器なのだが、天体とバックグラウンドを完全には分離することができないためバックグラウンドモデルを別途作成する必要がある。バックグラウンドは様々な要因により時間変化するために任意の時間、任意のエネルギーでのバックグラウンドをモデル化しなければならない。

従来のバックグラウンドモデルは経験的関数を用いて求められている。しかし、パラメータが最適か否かわからず、他の衛星に転用する際に再び半経験的手法を作成する必要がある。そこで、本研究では機械学習の手法の１つであるニューラルネットワークを用いてパラメータを定め、バックグラウンドモデルを自動生成し、汎化能力によって他の衛星へ転用可能なネットワークの開発を目的としている。

目 次

図 目 次

表 目 次

第1章　序論

「すざく」衛星は宇宙 X 線を観測するための衛星である。「すざく」衛星には硬 X 線検出器 (Hard X-ray Detector： HXD) が搭載されており、活動銀河核や超新星残骸などの高エネルギー天体の観測に貢献している。しかし、硬 X 線領域ではバックグラウンドが多いため天体からの光が埋もれてしまい、正確な観測は難しい。バックグラウンドは時間変化するため、任意の時間、任意のエネルギーにおけるバックグラウンドをモデル化する必要がある。現在バックグラウンドは半経験的モデルを用いて求められている。そこで本研究では、バックグラウンドのモデル化をニューラルネットワークによって行うことで適切なバックグラウンドモデルを自動生成することと、汎化能力によって他の衛星へ転用可能なネットワークを構築することを目的としている。

第2章　「すざく」HXD-PIN バックグラウンドモデル

2.1　X 線天文衛星「すざく」

「すざく」は日本の 5 機目の宇宙 X 線観測衛星であり、2005 年に打ち上げられ、2015 年に運用を終了した。「すざく」には 4 つの軟 X 線 CCD カメラと 1 つの硬 X 線検出器が搭載された。「すざく」の特徴として、両検出器ともバックグラウンド信号を非常に小さく抑えていることが挙げられる。これは軌道高度を約 550km と低くすることで地磁気によって宇宙からの荷電粒子を遮蔽していることに加えて、バックグラウンドを選別できる工夫を検出器に凝らしているからである。

2.2　硬 X 線検出器:HXD

硬 X 線検出器 (Hard X-ray Detector:HXD) は「すざく」に搭載されている検出器の 1 つである。HXD はイメージング機能を持っていない X 線分光器で、図 2.1 に示したように、PIN 型 Si フォトダイオードと GSO シンチレータで硬 X 線を検出し、それらを図 2.2 に示すように BGO シンチレータでアクティブにシールドしている。図 2.3 に示すように硬 X 線のうち、低いエネルギー帯 (約 10-60 keV) は PIN、PIN を透過する高いエネルギー帯 (約 50-500 kev) は GSO で検出する仕組みとなっている。

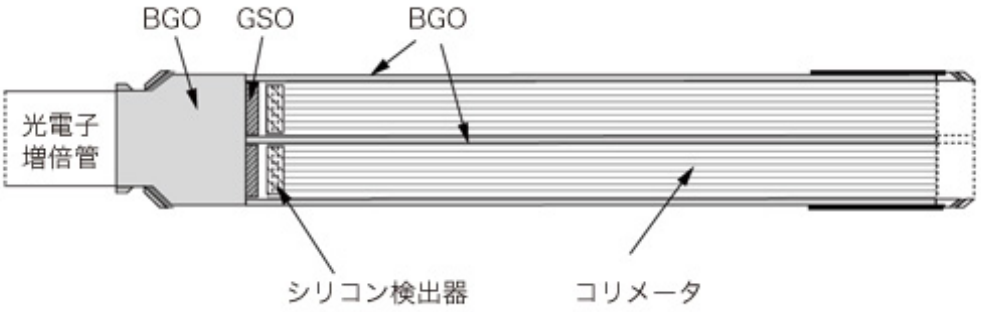


図 2.1: HXD のセンサユニット 1 本の断面図 [8]。X 線はコリメータを通して右から左へ入る。

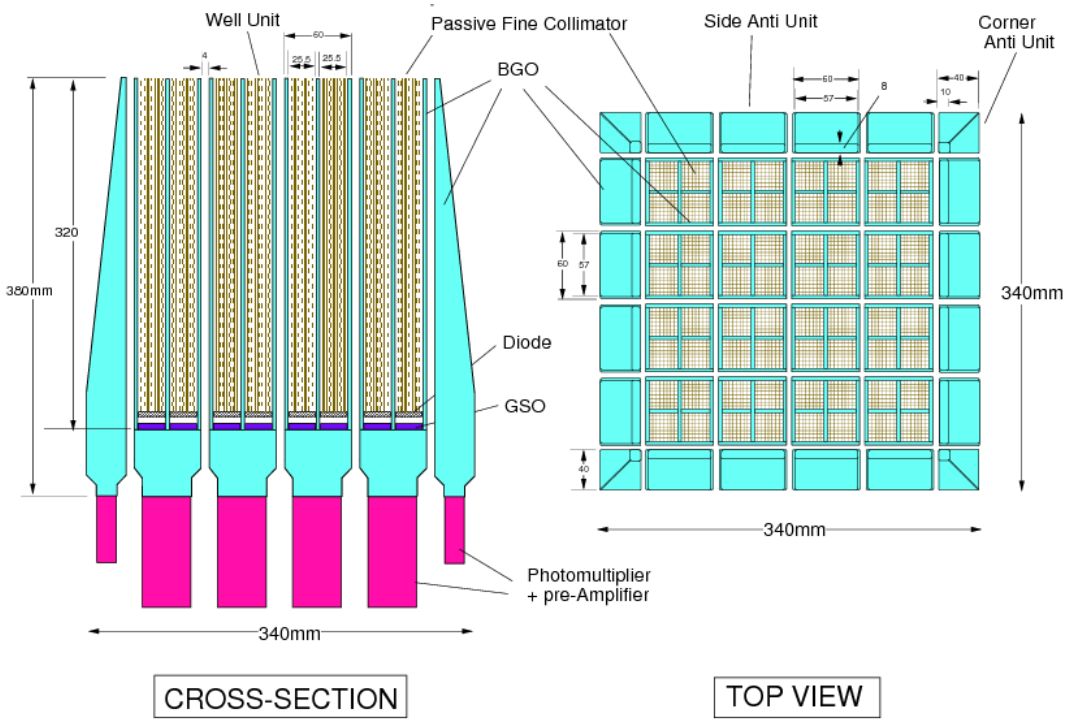


図 2.2: HXD の構造 [9]。センサユニットは 4 × 4 に並べており、周囲を BGO シールドで囲っている。

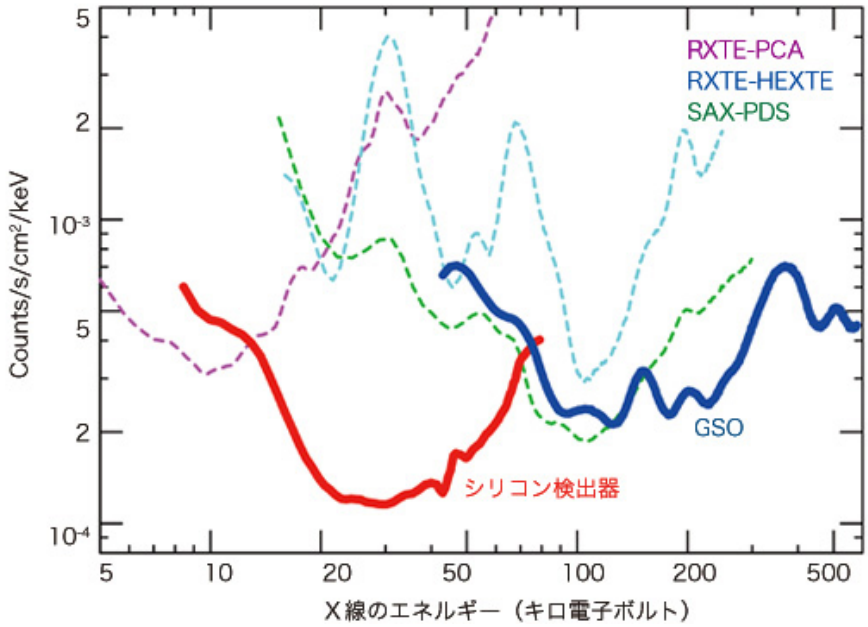


図 2.3: PIN 検出器、GSO 検出器の感度 [8]

2.3 PIN 検出器のバックグラウンドの成分

PIN 検出器のバックグラウンドになりうる要因は複数あるため、このセクションで説明する。

図 2.4 は SAA 通過後の PIN 検出器のバックグラウンドのライトカーブと COR の値であり、SAA による放射化や COR の大小などによって大きく時間変動していることがわかる。SAA と COR については 2.3.2 節で説明する。

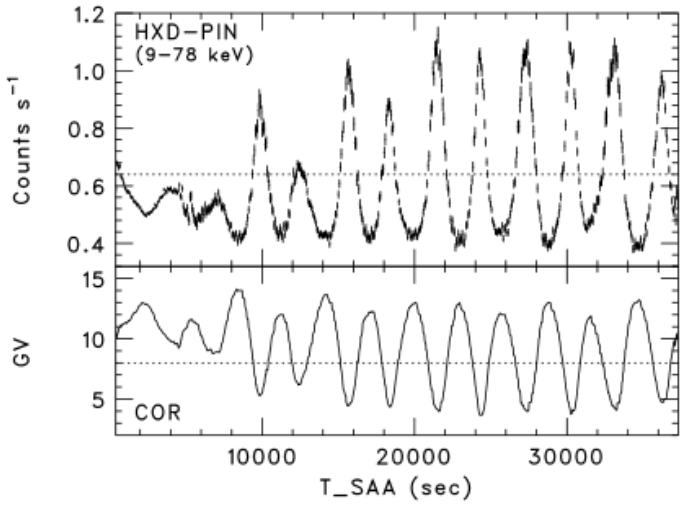


図 2.4: SAA 通過後の PIN バックグラウンドのライトカーブ (上)、COR(下)[3]

2.3.1 宇宙 X 線背景放射

宇宙 X 線背景放射 (cosmic X-ray background:CXB) は、全天からほぼ一様にやってくる X 線であり、遠方に存在する無数の活動銀河核からの放射の重ねあわせであると考えられている [2]。図 2.5 は、「あすか」衛星で撮られたかみのけ座方向の X 線画像である。これにより、ほぼ一様に X 線がやってきていることがわかる。

図 2.1 に示したように、HXD ではコリメータで視野を $34' \times 34'$ にまで絞り、CXB の漏れこみを抑えているが、それでも CXB はバックグラウンドのうちの約 10%を占める。

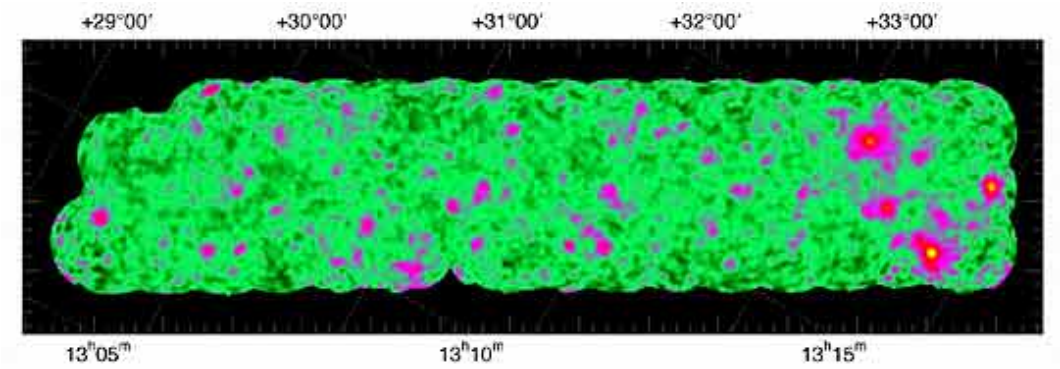


図 2.5: 「あすか」で撮ったかみのけ座方向の空域の画像 [7]。画像の色は X 線の強度を表している。

2.3.2 Non X-ray Background

天体由来のバックグラウンドである CXB の他には、検出器由来の Non X-ray Background(NXB) と呼ばれるバックグラウンドが存在する。NXB の原因は 3 種類存在する。

１つ目は、検出器の材料に混入している放射性同位体である。「すざく」HXD は放射性同位体の少ない材料を選んで作成したが、HXD の感度ではこのような微量な放射性同位体でも BGD となる。

２つ目は、検出器そのものが放射化することである。放射化とは、元々安定な同位体が、高エネルギー荷電粒子と衝突し核反応が起きて、不安定同位体に変化することで、不安定状態から安定状態へ遷移する際に放射線を放出し、バックグラウンドとなる。宇宙線（荷電粒子）は地磁気により、高度約 1000km 以上に捕らえられヴァン・アレン帯と呼ばれる放射線帯として集まっている。「すざく」は高度 500-600 km を周回してヴァン・アレン帯を避けているのだが、地磁軸と自転軸の中心が一致していないためにブラジル上空では放射線帯が高度約 300km まで下がっている。このヴァン・アレン帯の異常構造は南大西洋地磁気異常帯 (South Atlantic Anomaly:SAA) と呼ばれ、「すざく」は SAA を通過する度に放射化してしまう。図 2.6 は SAA の場所を示している。

３つ目は高エネルギー粒子が直接検出器に混入することである。２つ目のときに説明したように、「すざく」は地磁気によって宇宙線から守られているが、エネルギーの高いものは「すざく」まで到達することがある。地磁気によるシールド能力の指標を Cut Off Rigidity(COR) という。荷電粒子は BGO アクティブシールドと反同時計数法により除去できるが、宇宙線と大気などで作られた中性子は除去できない。

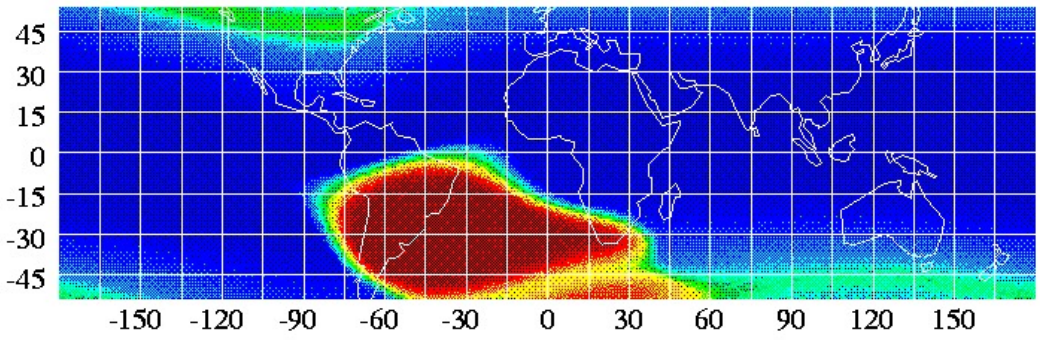


図 2.6: 南大西洋地磁気異常 [6]

2.4 PIN 検出器のバックグラウンドモデル

現在 PIN 検出器で用いられているバックグラウンドモデル作成法である bgd-a と bgd-d という 2 つの手法をここでは説明する。バックグラウンドモデルを作成する際は衛星が地球を向き、HXD の視野が完全に地球に覆われているため天体や CXB の信号を受けず、バックグラウンド成分は NXB のみとなっている地没データを用いている。

2.4.1 bgd-a

図 2.4 で示したように PIN 検出器のバックグラウンドは、COR に依存する部分と、SAA 通過による放射化成分に大別できる。そこで、bgd-a モデルでは、$PINUD$ と $PINUD_{buildup}$ という 2 つのパラメータを用いて、COR と SAA により変動するバックグラウンドをモデル化する手法である。$PINUD$ とは、2.3.2 節の 3 つ目に挙げた高エネルギー粒子を、PIN 検出器で数でえ上げたレートである。高エネルギー粒子が PIN 検出器に入ると通常の X 線信号よりも大きな波高値を生じるので、上限閾値 (Upper Discriminator: UD) を設けてそれを数えている。$PINUD_{buildup}$ とは、$PINUD$ をある時定数で畳み込み積分したもので、

$$PINUD_{buildup}(t) = \int_{-\infty}^{t} PINUD(t_0) exp(\frac{t_0 - t}{\tau}) dt_0$$

で定義されるパラメータである。$PINUD_{buildup}$ は SAA 中での被曝量を、放射化した原子核の寿命に応じて減衰したものであり、2.3.2 節で 2 つ目に挙げた成分を表現する。図 2.7 にそれぞれの例を示す。この 2 つのパラメータを用いて地没スペクトルを蓄積しデータベースを作る。そして天体観測時も、同パラメータでデータベースを参照して、バックグラウンドを得る。

bgd-a の欠点は、バックグラウンドはそれ以外のパラメータにも依存しているので、精度があまり良くない。

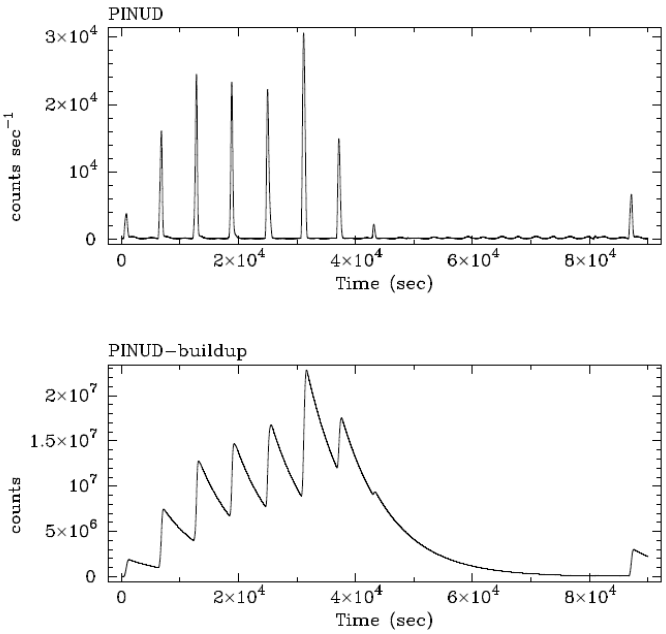


図 2.7: $PINUD$(上) と $PINUD_{buildup}$(下) の例 [1]。下図の $PINUD_{buildup}$ は時定数 $\tau = 8000$s で上図を畳み込んで作成したもの。

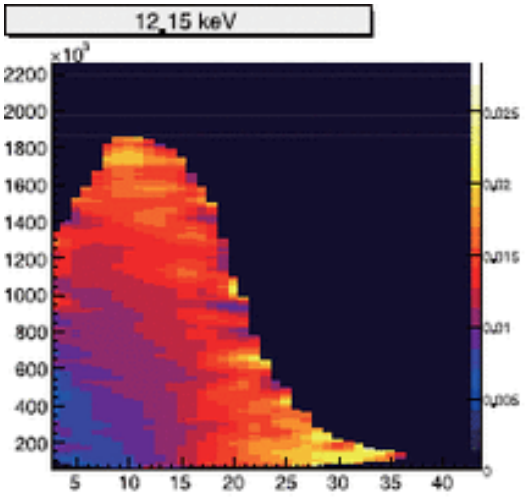


図 2.8: 12-15keV における NXB データベース [1]
横軸:$PINUD$ 縦軸:$PINUD_{buildup}$

2.4.2 bgd-d

bgd-d は bgd-a よりもパラメータ数を増やして精度の向上を図ったもので、bgd-a と違い経験的な関数によってバックグラウンドを見積もる手法である。もともとは GSO 検出器で用いられていたが、PIN 検出器

でも用いられており、PIN 検出器では 10-70 keV の範囲を一括で処理する。

パラメータは bgd-a で用いた $PINUD$ と $PINUD_{buildup}$ の他に、地磁気と HXD の成す角 θ_B と GSO 検出器の 450-700 keV のカウント数 $GSOHCNT(t)$ を導入し、$PINUD_{buildup}$ はいくつかの時定数で畳み込み積分する。経験的な関数は

$$\begin{aligned} BGD(t) =& a + \sum_{k=1}^{3} b_k \int (1 + c_k \frac{90^\circ - \theta_B(t')}{90^\circ}) \cdot PINUD(t') \cdot exp(-\frac{t-t'}{\tau_k}) \\ &+ d \cdot PINUD(t) + e \cdot PINUD^2(t) \\ &+ f \cdot GSOHCNT(t) \cdot [1 + g \cdot exp(-\frac{t - t_{SAA}}{\tau_g})] + h(t) \end{aligned} \tag{2.1}$$

で与えられる [1]。係数 a,b_k,c_k,d,e,f,g はモデルを表現するパラメータであり、t_{SAA} は SAA を抜けてからの時間であり $\tau_g = 10000$s である。$h(t)$ は導入できていない未知のパラメータのための補正項である。$PINUD(t)$,θ_B,t を入力した後、各月と前後 10 日の地没データでフィッティングすることでパラメータを決定する。そして任意の天体観測時間 t を代入して、バックグラウンドを得る。さらに上記のようにフィッティングしたバックグラウンドを、「すざく」衛星の高度、COR、θ_B、地球の縁からの仰角という 4 つのパラメータで補正する。

2.5 バックグラウンドモデルの不定性

バックグラウンドモデルの不定性を定量評価するために、天体からの信号が無い地没データを使って、実データとモデルのサーとの残差を計算し、その分布を取ったものが、図 2.9 である。この分布の標準偏差は、バックグラウンドモデルの不定性に加えて、統計誤差も含まれている。これらの値が独立であると仮定して、誤差伝播の法則から統計誤差を差し引いた値を、表 2.5 に示す。bgd-a も bgd-d も不定性は 4%未満であり、どの条件でも bgd-d の方が不定性は小さい。

手法	bgd-a			bgd-d		
積分時間	10 ks	20 ks	40 ks	10 ks	20 ks	40 ks
15-40 keV						
標準偏差	3.75%	3.23%	2.96%	2.31%	1.72%	0.99%
統計誤差	1.83%	1.36%	0.93%	1.83%	1.36%	0.93%
不定性	3.27%	2.93%	2.81%	1.40%	1.05%	0.34%
40-70 keV						
標準偏差	5.53%	4.34%	3.39%	4.92%	3.51%	2.87%
統計誤差	4.03%	3.01%	2.03%	4.03%	3.01%	2.03%
不定性	3.78%	3.12%	2.71%	2.82%	1.81%	2.02%

表 2.1: 各時間毎の誤差の値 [1]

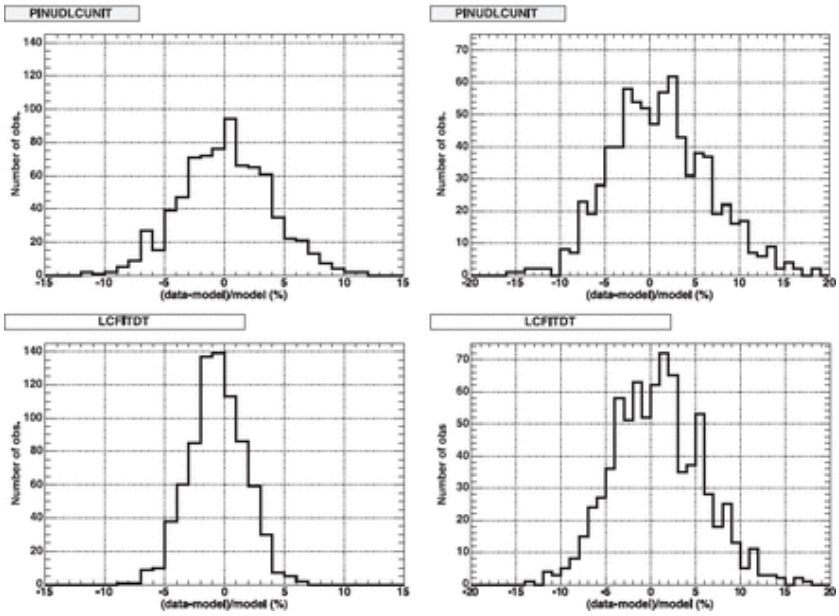


図 2.9: bgd-a(上) と bgd-d(下) の、モデルと実データとの残差ヒストグラム [1]
15-40keV(左)40-70keV(右) 積分時間 10 ks

2.6 研究の目的

現状のすざく HXD-PIN 検出器のバックグラウンドモデルは経験的かつ複雑な関数を用いて作成されている。従来の手法では、いくつかある時定数が最適か否か不明であり、モデルに組み込んでいない入力パラメータの寄与も考慮できていない。また、今後同様の衛星搭載検出器を運用する際に人間がモデルを作成する必要がある。そこで本研究では、機械学習を用いて、様々な入力パラメータの寄与を考慮したバックグラウンドモデルを自動的に生成する手法を開発することを目指す。

第3章　ニューラルネットワーク

ニューラルネットワークとは、日本語では神経回路網と訳され、脳機能に見られるいくつかの特性を表現する数学モデルであり、機械学習の手法の１つである。機械学習はパラメータを自動的にチューニングする手法であるため、複雑な問題を解く際に非常に有効である。昨今ディープラーニング (深層学習) という言葉を耳にすることが多いが、ディープラーニングとは、多層になったニューラルネットワークのことである。ここでは文献 [11] と [12] を参考にして説明する。

そもそも機械学習とは、データから結果を予測する精度を改善していくアルゴリズムすべてを指す言葉である。(弱い)AI と呼ばれることがあるのは、改善する (学習する)、ということを知能と見なしているからである。この章ではニューラルネットワークがどのように学習するのか、その際にどのような問題があるのか、どの程度学習が進んでいるのかを説明する。尚、学習の際はデータを学習に用いるためのトレーニング (学習) データと評価用のテストデータに分ける必要がある。

図 3.1 はニューラルネットワークの立ち位置を表したベン図である。図のようにニューラルネットワークは教師あり機械学習、教師なし機械学習両方にまたがる手法である。本研究では回帰問題を扱うため、教師あり機械学習という予想結果を実データ (正解データ) と比較して学習する手法を用いる。

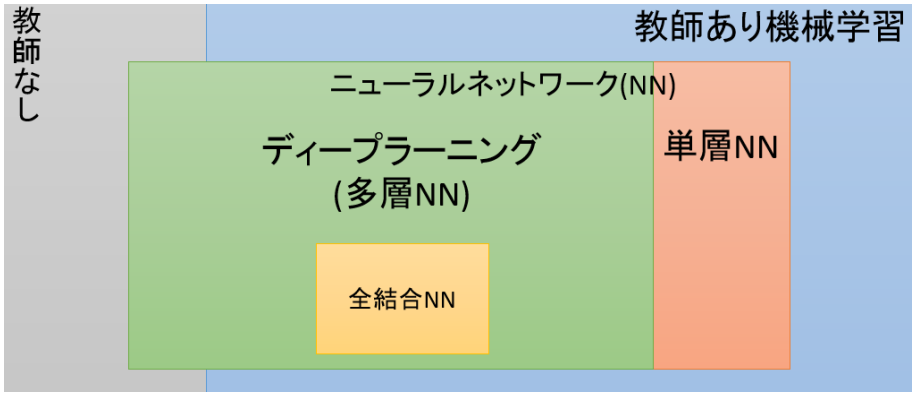


図 3.1: ニューラルネットワークの立ち位置

3.1　全結合ニューラルネットワーク

ニューラルネットワークはいくつかの層から成っており、１つの層には１つ以上のユニット (ニューロン) で構成されている。ニューラルネットワークにはいくつかの種類が存在するが、その中で最も基本的なものが全結合ニューラルネットワークと呼ばれるものである。k 層目の i 番目のユニットが k-1,k+1 層目のすべてのユニットと結合しているニューラルネットワークが全結合ニューラルネットワークである。

3.1.1 単層ニューラルネットワーク

図 3.2 は単層ニューラルネットワークの概念図である。単層ニューラルネットワークは入力層と出力層で構成されているが、入力層を層数に数えないため単層ニューラルネットワークと呼ばれている。

I 個の入力データは入力層の各ユニット $(x_1 \sim x_I)$ に格納される。入力データとは別に、常に 1 を格納しているバイアス用のユニット x_0 が存在する。入力層の各ユニットと出力層の各ユニットは結合しており、その結合ごとに重み (w_{ij}) が存在する。

出力層のユニット $(y_1 \sim y_J)$ に格納されるのは、

$$y_j = f\left(\sum_{i=0}^{I} x_i w_{ij}\right) \tag{3.1}$$

で定義される値である。f は活性化関数とよばれ、非線形もしくは恒等関数が使われる。

単層ニューラルネットワークは、線形分離可能な問題しか解けない。そこで単層ニューラルネットワークを拡張して、多層にして深く (ディープに) することで、非線形分離問題を解くことができる。

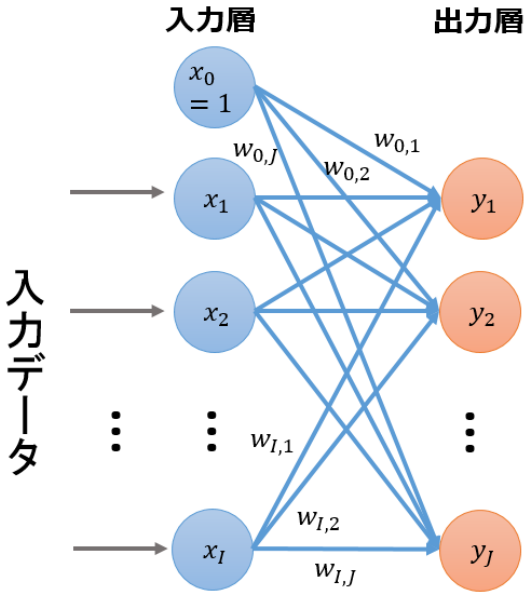


図 3.2: 単層ニューラルネットワーク

3.1.2 勾配法

ニューラルネットワークを用いた機械学習は、出力層 y_j の予想とトレーニングデータを比べ、それを評価する損失 (コスト) 関数の値を最小もしくは最大にするような重み w_{ij} の組み合わせを探索することであり、最適化問題の 1 つに帰着できる。最適化問題を解決するために、勾配法と呼ばれる、関数の勾配を元に解を探索するアルゴリズムが考案されている。代表的な 3 種類の勾配法を以下に紹介する。

○バッチ勾配降下法

バッチとは 1 つのまとまりのことであり、すべてのトレーニングデータを同時に用いて重みを更新する方法である。トレーニングデータすべてを何度も用いる必要があるため、大規模なデータの場合時間がかかるが、行列計算のため高度に並列化しやすく、計算効率が良い。

○確率的勾配降下法

トレーニングデータ 1 つごとに重みの更新をする方法で、トレーニングデータは無作為に抽出する。バッチ勾配降下法よりも一般的に収束が早く、浅い極小値を抜け出しやすく、パラメータの更新が小刻みであるため途中経過をより詳細に監視できる。また、途中で新しくサンプルが増えるオンライン学習にも利用可能である。ただし、1 つごとに更新をするためベクトル計算を用いることができず、計算効率が悪い。

○ミニバッチ勾配降下法

バッチ勾配降下法と確率的勾配降下法の折衷であり、ミニバッチとよばれる少数のデータ (例えば 50 個) で重みを更新する。両者の長所を生かした最も一般的な手法である。この方法を確率的勾配降下法と呼ぶこともある。

3.1.3 誤差逆伝播法

3.1.1 節で紹介した、入力層から出力層への計算を順伝播という。順伝播は入力データを元に予測結果を出力するものだが、学習のためには予測結果を元に重みを更新する必要がある。重みの更新する最も単純な手法は数値微分により重みの勾配を得る方法であるが、計算に時間がかかるという難点がある。誤差逆伝播法は重みを効率良く更新する手法であり、その名前は、出力層から入力層へ順伝播とは逆向きに計算していくことに由来する。

重みの更新は

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial}{\partial w_{ij}} E(W) \tag{3.2}$$

$$W \equiv \begin{pmatrix} w_{01} & w_{02} & \cdots & w_{0J} \\ w_{11} & w_{12} & \cdots & w_{1J} \\ \vdots & \vdots & \ddots & \vdots \\ w_{I1} & w_{I2} & \cdots & w_{IJ} \end{pmatrix}$$

と表すことができる。W は入力層の i 番目のユニットと出力層の j 番目のユニット間の重みを表す $(I+1)\times J$ の行列である。η は学習率とよばれるもので学習の速度に関するパラメータであり、小さすぎると学習がなかなか進まないが、大きすぎると発散したり最小値に到達できなくなったりする。$E(W)$ は損失関数である。要素すべてをまとめて表記すると

$$W \leftarrow W - \eta \frac{\partial}{\partial W} E(W) \tag{3.3}$$

とかける。ここで、単層ニューラルネットワークの順伝播は式 (3.1) に示したものをベクトルで表現すると

$$\vec{y} = f(\vec{p}) \tag{3.4}$$

$$\vec{p} \equiv \vec{x} W$$

となる。勾配は

$$\frac{\partial E(W)}{\partial w_{ij}} = \frac{\partial E(W)}{\partial p_j} \frac{\partial p_j}{\partial w_{ij}} = \frac{\partial E(W)}{\partial p_j} x_i \tag{3.5}$$

と計算できる。成分に分解せずに書くと

$$\frac{\partial E(W)}{\partial W} = \frac{\partial E(W)}{\partial \vec{p}} \frac{\partial \vec{p}}{\partial W} = \frac{\partial E(W)}{\partial \vec{p}} \vec{x} \tag{3.6}$$

となるので $\frac{\partial E(W)}{\partial \vec{p}}$ を計算すればよいことになる。

例えば損失関数が二乗誤差 $E(W) \equiv \sum \frac{1}{2}(\vec{t} - \vec{y})^2$ とする ($\vec{t}$はトレーニングデータの正解値) と、$\frac{\partial E(W)}{\partial \vec{p}} = -(\vec{t} - \vec{y})\vec{y'}$ となる。活性化関数が恒等写像のときは $\vec{y'}$ の成分はすべて 1 なので、$\frac{\partial E(W)}{\partial W} = -(\vec{t} - \vec{y})\vec{x}$ が求める勾配となる。

3.1.4 ディープラーニング

本章冒頭で述べた通り、ディープラーニングとは多層になったニューラルネットワークのことである。ニューラルネットワークは基本的に層の数、ユニットの数を増やすことで重みパラメータが増えるので自由度が増え、複雑な問題を解くことができるようになる。

図 3.3 はディープラーニングの概念図を示している。ディープラーニングでは入力層と出力層の間にいくつかの隠れ層 (中間層) が存在する以外は単層ニューラルネットワークと同様である。中間層には入力層と同様にバイアス用のユニットが存在し、それ以外のユニットには単層のときの出力層同様に活性化関数で写像した値が格納される。

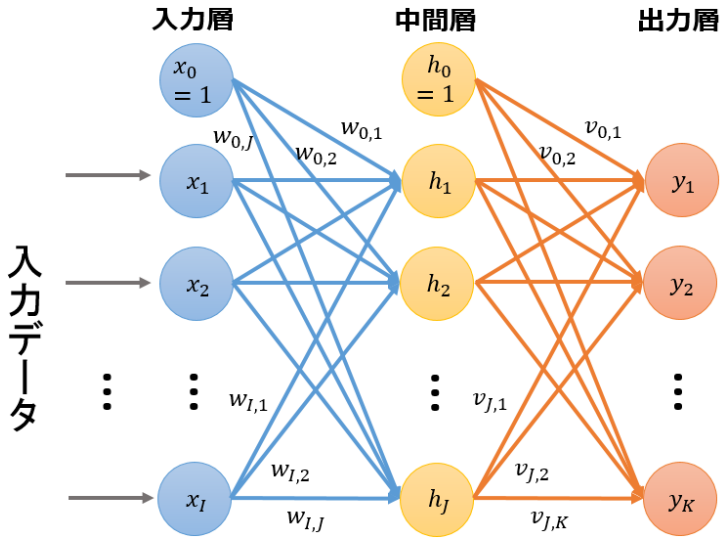


図 3.3: ディープラーニング (多層ニューラルネットワーク)

層が深くなるにつれて計算量が増えて学習に時間がかかるようになるが、誤差逆伝播法により効率的に計算することができる。3.1.3 節と同様に順伝播を表記すると、入力層-隠れ層は

$$\vec{h} = f(\vec{p}) \tag{3.7}$$
$$\vec{p} \equiv \vec{x}W$$

隠れ層-出力層は

$$\begin{aligned}\vec{y} &= g(\vec{q}) \\ &= g(f(\vec{p})V) \\ \vec{q} &\equiv \vec{h}V\end{aligned} \tag{3.8}$$

と表記できる。重みの勾配は

$$\frac{\partial E(W,V)}{\partial W} = \frac{\partial E(W,V)}{\partial \vec{p}}\frac{\partial \vec{p}}{\partial W} = \frac{\partial E(W,V)}{\partial \vec{p}}\vec{x} \tag{3.9}$$

$$\frac{\partial E(W,V)}{\partial V} = \frac{\partial E(W,V)}{\partial \vec{q}}\frac{\partial \vec{q}}{\partial V} = \frac{\partial E(W,V)}{\partial \vec{q}}\vec{h} \tag{3.10}$$

となる。ここで、

$$\begin{aligned}\frac{\partial E(W,V)}{\partial p_j} &= \sum_{k=1}^{K}\frac{\partial E(W,V)}{\partial q_k}\frac{\partial q_k}{\partial p_j} \\ &= \sum_{k=1}^{K}\frac{\partial E(W,V)}{\partial q_k}(f'(p_j)v_{jk})\end{aligned} \tag{3.11}$$

であるので、成分に分解せず表記すると

$$\frac{\partial E(W,V)}{\partial \vec{p}} = f'(\vec{p}) \bigcirc V\frac{\partial E(W,V)}{\partial \vec{q}} \tag{3.12}$$

となる。$\bigcirc$ はアダマール積 (要素積) とよばれる演算を表しており、例えば

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \bigcirc \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}a_{11} & a_{12}a_{12} \\ a_{21}a_{21} & a_{22}a_{22} \end{pmatrix}$$

となる。

式 (3.11) を中間層が N 個であるニューラルネットワークの任意の中間層 n での式に拡張すると、

$$\begin{aligned}\frac{\partial E(W^{(1)},\cdots,W^{(N)},V)}{\partial p^{(n)}_{j^{(n)}}} &= \sum_{j^{(n+1)}=1}^{J^{(n+1)}}\frac{\partial E(W^{(1)},\cdots,W^{(N)},V)}{\partial p^{(n+1)}_{j^{(n+1)}}}({f^{(n)}}'(p^{(n)}_{j^{(n)}})w^{(n+1)}_{j^{(n)}j^{(n+1)}}) \\ &= \sum_{j^{(n+1)}=1}^{J^{(n+1)}}\sum_{j^{(n+2)}=1}^{J^{(n+2)}}\frac{\partial E(W^{(1)},\cdots,W^{(N)},V)}{\partial \vec{p_{j^{(n+2)}}}^{(n+2)}}({f^{(n+1)}}'(p^{(n+1)}_{j^{(n+1)}})w^{(n+2)}_{j^{(n+1)}j^{(n+2)}})({f^{(n)}}'(p^{(n)}_{j^{(n)}})w^{(n+1)}_{j^{(n)}j^{(n+1)}}) \\ &= \cdots \\ &= \sum_{j^{(n+1)}=1}^{J^{(n+1)}}\cdots\sum_{j^{(N)}=1}^{J^{(N)}}\sum_{k=1}^{K}\frac{\partial E(W^{(1)},\cdots,W^{(N)},V)}{\partial q_k} \\ &\quad ({f^{(N)}}'(p^{(N)}_{j^{(N)}})v_{j^{(N)}k})({f^{(N-1)}}'(p^{(N-1)}_{j^{(N-1)}})w^{(N)}_{j^{(N-1)}j^{(N)}})\cdots({f^{(n)}}'(p^{(n)}_{j^{(n)}})w^{(n+1)}_{j^{(n)}j^{(n+1)}})\end{aligned} \tag{3.13}$$

ベクトル形式で書くと

$$\begin{aligned}\frac{\partial E(W^{(1)},\cdots,W^{(N)},V)}{\partial \vec{p}^{(n)}} =& ({f^{(n)}}'(\vec{p}^{(n)}) \bigcirc W^{(n+1)}) \bigcirc ({f^{(n+1)}}'(\vec{p}^{(n+1)}) \bigcirc W^{(n+2)}) \bigcirc \\ & \cdots \bigcirc ({f^{(N)}}'(\vec{p}^{(N)}) \bigcirc V)\frac{\partial E(W^{(1)},\cdots,W^{(N)},V)}{\partial \vec{q}}\end{aligned} \tag{3.14}$$

ここで、$W^{(n)}$ は中間層第 n-1 層と第 n 層間の重みであり、その他同様に上付き文字がついているものは第 n 層であるという表記である。

3.2 学習における問題

機械学習において、学習が進まなくなる、予測結果が悪化してしまうといった問題がしばしば発生する。ここではその諸問題について述べる。

3.2.1 勾配消失・爆発問題

ニューラルネットワークでは重みの勾配を計算することで学習をしているが勾配が 0 または非常に大きくなってしまい重みを更新できなくなることがある。これをそれぞれ勾配消失問題・勾配爆発問題という。式 (3.14) に示した重みの更新式は活性化関数の導関数と重みとのいくつもの積になっている。$(f^{(n)\prime}(\vec{p}^{(n)}) \bigcirc W^{(n+1)}) \ll 1$ の場合、積が多く連なると勾配が指数関数的に 0 に近づき、逆に $(f^{(n)\prime}(\vec{p}^{(n)}) \bigcirc W^{(n+1)}) \gg 1$ だと指数関数的に増加することがわかる。

勾配消失問題は主に活性化関数が原因になることが多い。例えば活性化関数として図 3.4 に示すシグモイド関数 $\sigma(x) = \frac{1}{1+e^{-x}}$ を採用すると、導関数は $\sigma'(x) = \sigma(x)(1-\sigma(x))$ で表され、$0 \leq \sigma'(x) \leq 0.25$ である。$\prod^n \sigma'(x) \leq 0.25^n$ であるので活性化関数にシグモイド関数を多く用いると深い層では勾配が消える。そのため、積が連なると非常に小さな値になってしまう関数を避ける必要がある。

勾配爆発問題は重みが大きな値になりすぎることが主な要因となる。例えば重みをすべて 100 に設定すると、活性化関数がシグモイド関数だとして $0 < \sigma(x)w \leq 25$ となる。$\prod^n \sigma'(x)w \leq 25^n$ であるので、勾配が発散することがある。勾配の爆発を防ぐためには重みの初期値を大きくし過ぎないようにしたり、重みの上限値を定めておいたりする方法がある。

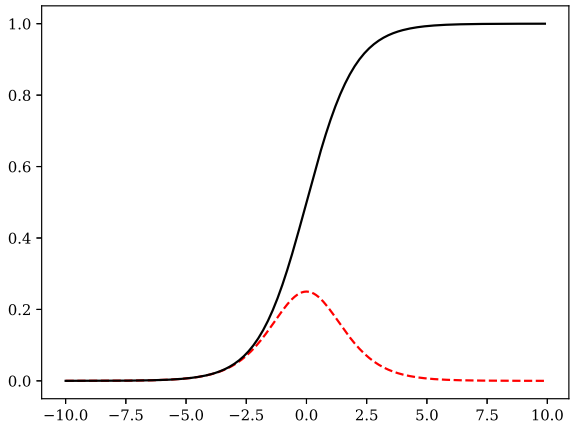
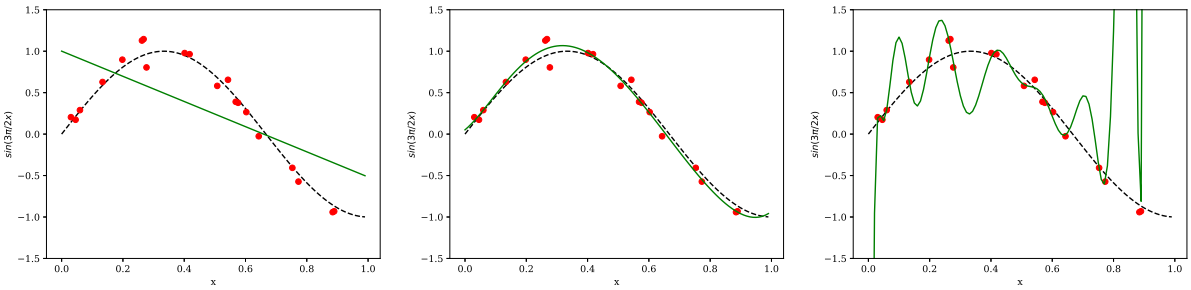


図 3.4: シグモイド関数 $\sigma(x) = \frac{1}{1+e^{-x}}$ (黒実線) とその導関数 (赤破線)

3.2.2　過学習

学習したニューラルネットワークを用いて未知のデータからも正しい結果を予測できること（この能力を汎化能力という）が期待されるが、学習データのみに適合しすぎて正しく予測できないことがある。この状態のことを過学習といい、過剰適合やオーバーフィッティング (overfitting) ともよばれている。図 3.5 に示した過学習の例では、6 次関数の場合は真の分布 $\sin(\frac{3\pi}{2}x)$ を表すことに成功しており、未知のデータを入力しても良い予測が得られる。しかし、15 次関数では学習データ点付近しか真の分布に近づいておらず、未知のデータを良く予測できていない。反対に 1 次関数ではパラメータが少なすぎてデータ点付近ですら真の分布とは乖離しており、このことをアンダーフィッテイングという。

以上は多項式による近似を例に挙げたが、ニューラルネットワークも同様で、その層やユニット数を増やすと重みの個数が増える (パラメータが増える) ため過学習が起きやすい。



1 次関数　　6 次関数　　15 次関数

図 3.5: $sin(\frac{3\pi}{2}x)$ 分布 (破線) からランダムに抽出し、誤差を加えて作成した学習データ点 (赤丸) に対し、多項式の曲線 (緑色) をフィットしたもの。

3.3　学習に関する手法の紹介

ニューラルネットワークには前述の問題を解決するための手法や予測精度を上げるための手法が多く存在する。ここではそれらについて紹介する。

3.3.1　勾配消失問題に関する手法

○活性化関数の選択

勾配損失問題に対処するために、活性化関数は注意深く選択し、状況に応じて使い分ける必要がある。出力層は出力したいデータの値域を満たした関数を用いる必要があるが、隠れ層では引数の値が小さければ写像の値も小さく引数の値が大きければ写像の値も大きい関数であれば良い。ただし、線形関数を用いた多層ニューラルネットワークは等価な単層ニューラルネットワークが存在するため、非線型関数を用いたほうがより良く学習を進めることができる。ここでは代表的な関数を紹介する。各活性化関数の導関数を図 3.6 に示す。

・双曲線正接関数

双曲線正接関数 (hyperbolic tangent function) は

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.15}$$

$$\tanh'(x) = \frac{4}{(e^x + e^{-x})^2} \tag{3.16}$$

で表され、シグモイド関数と形が似ているが値域が異なる。最大の利点は導関数の値域が $0 < \tanh'(x) < 1$ であるので勾配が消えにくいことである。

・ReLU(rectified linear unit)

日本語ではランプ関数や正規化線形関数とよばれ、

$$f(x) = \max(0, x) \tag{3.17}$$

$$f'(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases} \tag{3.18}$$

と表される。$x > 0$ のとき勾配が必ず 1 となるので勾配が消えることが無く、式も単純なため計算が高速であるというメリットがある。しかし、$x \leq 0$ になるとそのユニットはずっと 0 のままになってしまい学習に寄与しなくなってしまうこと、出力層には用いることができないことが知られている。

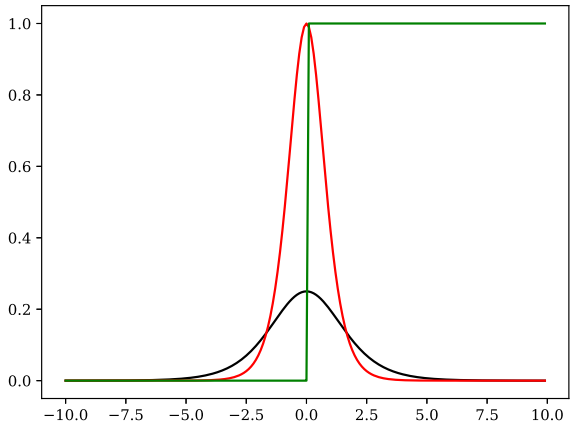


図 3.6: 活性化関数として使われるシグモイド関数 (黒)、tanh(赤)、ReLU(緑) の導関数

3.3.2 過学習に関する手法

○ Early Stopping

Early Stopping は過学習が起きる前に学習を打ち切る手法である。学習中にテストデータの予測精度が悪化し続けていたら学習をやめるといったもので、単純だが強力である。

○アンサンブル学習

学習に用いるデータは基本的に母集団では無く標本であるため、フィッテイングのパラメータが多いとどうしても過学習が起こってしまう。そのため、同じ構造（類似の構造を用いることもある）の複数のニューラルネットワークを独立に学習させて、予測結果の平均を取ると精度が良くなることが知られている。つまり集合知を得ているようなものである。

○ドロップアウト

ドロップアウトはアンサンブル学習を１つのニューラルネットワークで擬似的に行う手法である。図 3.7 に概念図を示す。名前の通り、重みの更新ごとにランダムでユニットを選出し、一時的に対応する重みをゼロにすることで除外 (ドロップアウト）して学習を行う。除外するユニットはランダムなので、毎回少しずつ異なるニューラルネットワークで学習を行うことと同義になるため擬似的なアンサンブル学習になる。

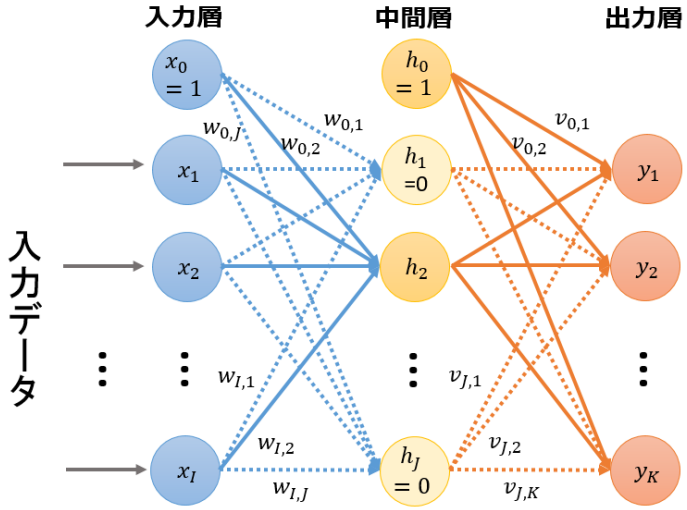


図 3.7: ドロップアウトの概念図。色の薄いユニットはランダムに選んで除外したもので、点線上の重みはゼロにする。

順伝播は式 (3.7) で示したが、ドロップアウトを適用する場合は

$$\vec{h} = f(\vec{p}) \bigcirc \vec{m} \tag{3.19}$$

で表され、$\vec{m}$ は各要素が確率 p で 0、$(1-p)$ で 1 を取るベクトルとアダマール積を取ることでマスクして除外する。確率 p は 0.5 がよく用いられている。逆伝播も式 (3.12) で示したが、同様に

$$\frac{\partial E(W,V)}{\partial \vec{p}} = f'(\vec{p}) \bigcirc \vec{m} \bigcirc V \frac{\partial E(W,V)}{\partial \vec{q}} \tag{3.20}$$

と表される。ドロップアウトにより学習時の重みの数が $(1-p)$ 倍に減っていたため重みの値が $\frac{1}{1-p}$ 倍になっており、テスト時には $(1-p)W$ とする必要がある。

3.3.3　学習の収束に関する手法

○重みの初期化

重みの初期化により学習の成否が左右されるので、その設定は非常に重要である。例えば重みを大きな値にしすぎると、3.2.1 節で説明したように勾配が爆発する。逆に重みを小さくしすぎると勾配も小さくなるため学習スピードが遅くなる。また、初期値をすべて同じにすると順伝播 (式 3.7) と逆伝播 (式 3.12) の式を見るとわかるように、すべての重みが同じように更新されるので、問題が発生する。これではたくさんのユニットを持つ意味が無くなってしまうので、重みの対照的な構造を崩すために重みの初期化はランダムな値で行う必要がある。

以上を踏まえた初期化の方法がいくつか提案されているので、ここで紹介する。

・正規分布による初期化

平均 0 の正規分布により初期化を行う手法で、標準偏差は 0.01 から 0.1 の間にされることが多い。

・Xavier の初期値 [13]

Xavier Glorot 氏らに提案された手法で、正規分布による初期化の発展型。前の層のユニット数を n とすると標準偏差を $\sqrt{1/n}$ にする。

・He の初期値 [14]

Kaiming He 氏らに提案された手法で、活性化関数が ReLU のときに用いると良いとされている正規分布による初期化の発展型。標準偏差を $\sqrt{2/n}$ にする。厳密には異なるが、直感的には、ReLU の範囲は $[0, \infty)$ であるため、より広がりをもたせるためには Xavier の初期値より 2 倍広くすると解釈できる。

○重みの最適化

重みを最適化するアルゴリズムをオプティマイザという。3.1.2 節で手法の 1 つである確率的勾配降下法（Stochastic Gradient Descent;SGD）を紹介したが、勾配の最小付近で振動してしまう欠点がある。そのため、発展させて欠点を補った手法をいくつか紹介する。各手法の収束経路を図 3.8 に示す。ただし、それぞれ得意な問題が異なるため後発の手法を用いれば学習効率は必ず改善するわけではない [12]。

・モメンタム法 [15]

SGD では式 (3.3) で重みを更新したが、モメンタム法では

$$V \leftarrow \alpha V - \eta \frac{\partial E(W)}{\partial W} \tag{3.21}$$

$$W \leftarrow W + V \tag{3.22}$$

という式で重みを更新する。この式は「空気抵抗の式」と類似しており、V は速度に対応する。αV はモメンタム項と呼ばれ、勾配が 1 つ前の更新と同じ方向の場合は加速、逆向きの場合は減速の役割を果たし、収束を早める効果がある。

・AdaGrad 法 [16]

モメンタム法は補正項を加えることで重みの更新スピードを調節したのに対し、AdaGrad は学習率に補正を加えることで更新スピードを調節する手法である。Ada は「適応的」を意味する Adaptive に由来している。重みの更新式は

$$H \leftarrow H + \frac{\partial E(W)}{\partial W} \bigcirc \frac{\partial E(W)}{\partial W} \tag{3.23}$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{H + \epsilon}} \frac{\partial E(W)}{\partial W} \tag{3.24}$$

で表される。H は今までの勾配の二乗和であるので、初期値から大きく変わった重みほど更新スピードが小さくなる。ϵ は分母が 0 にならないための微小量である。H が単調増加するために学習を進めていると更

新スピードが際限なく小さくなってしまう欠点がある。

・Adam(Adaptive Moment Estimation) 法 [17]

モメンタム法と AdaGrad 法を組み合わせた手法であり、その式は

$$\eta \leftarrow \eta \frac{\sqrt{1-\beta_2^n}}{1-\beta_1^n} \tag{3.25}$$

$$M \leftarrow \beta_1 M + (1-\beta_1)\frac{\partial E(W)}{\partial W} \tag{3.26}$$

$$V \leftarrow \beta_2 V + (1-\beta_2)(\frac{\partial E(W)}{\partial W})^2 \tag{3.27}$$

$$\hat{M} \leftarrow \frac{M}{1-\beta_1^n} \tag{3.28}$$

$$\hat{V} \leftarrow \frac{V}{1-\beta_2^n} \tag{3.29}$$

$$W \leftarrow W - \frac{\eta}{\sqrt{\hat{V}}+\epsilon}\hat{M} \tag{3.30}$$

である。β_1, β_2 は調整用のパラメータであり $0 < \beta_1, \beta_2 < 1$ の範囲で選ぶ。n 更新回数である。M がモメンタムのように働き、V が AdaGrad のように働く。ただし AdaGrad と異なり β_1, β_2 を用いて減衰させているので更新スピードが際限なく小さくならない。一般的に $\beta_1 = 0.9, \beta_2 = 0.999$ に設定される。

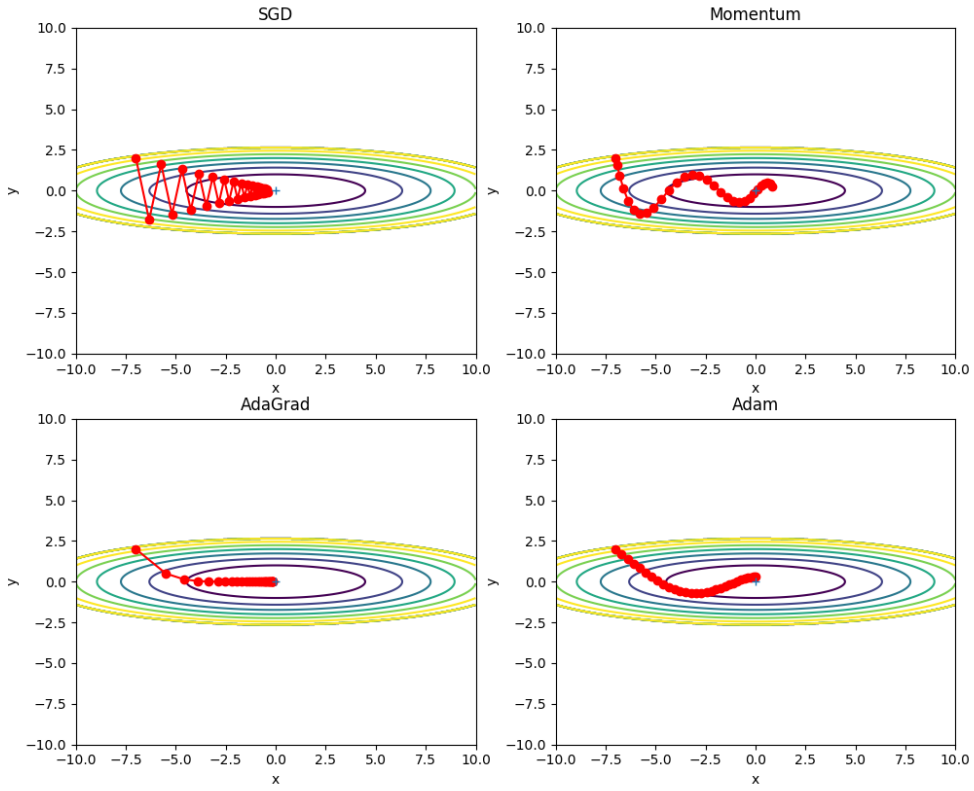


図 3.8: オプティマイザの収束経路の比較 [12]

3.4 学習の可視化

このセクションでは学習に関する可視化について述べる。

3.4.1 損失関数の値の推移

重みの更新は損失関数を最小化するように行なっているため、損失関数がどれだけ小さくなったかを見ることは学習の進み具合の把握や学習の成否、予測精度の改善のための指標となる。図 3.9 は、横軸が重みの更新回数、縦軸が損失関数の値をプロットしたグラフである。この例では、重みの更新を繰り返すと、損失関数値は学習データでもテストデータでも同様に下がり、学習がうまく進んでいることがわかる。しかし、学習データの損失関数値が下がる一方、テストデータでは上がる場合があり、過学習が起きている可能性が高い。

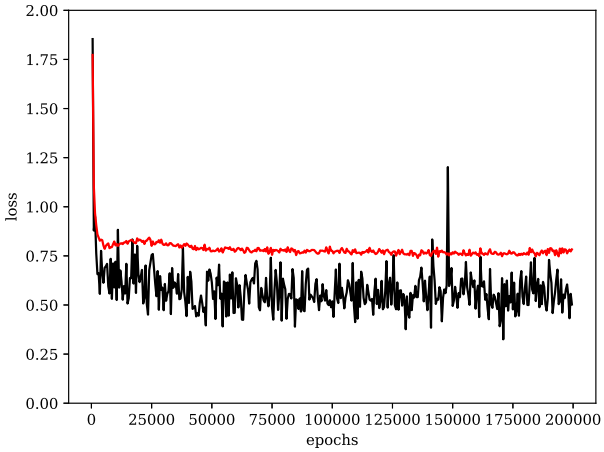


図 3.9: 損失関数の値の推移 (学習データ:黒、テストデータ:赤) の例

3.4.2 隠れ層のアクティベーション

アクティベーションとは、活性化関数での写像した値 $f(\vec{x}\cdot\vec{w})$ のことを指す。逆伝播 (式 3.12) を考えると、アクティベーションが 0 のユニットは勾配が消えていることがわかる。また、アクティベーションが同じユニットが複数あっても、$x_1w_1 + x_2w_2 = x_1(w_1 + w_2) = x_1w$ となり 1 つのユニットで表現できるため無意味である。そのため各層のアクティベーションの分布はある程度の広がりを持つと良い。ただし、アクティベーションが大きすぎると勾配が爆発してしまう。

本研究ではユニット毎に 26 回分を平均したものを 1 つのアクティベーションとしており、

$$(\text{アクティベーション}) = \frac{1}{26}\sum_{i=1}^{26} f(\vec{x^{(i)}}\cdot\vec{w}) \tag{3.31}$$

としている。x の上付き文字 i は、いくつ目のデータを入力しているのかを表している。(データ数)/26 個の度数がある。図 3.10 にアクティベーションの分布の例を示す。値域が 0 以上となっているのは、本研究で用いた活性化関数の値域が $[0, \infty)$ であるからである。学習がうまく進んでいる場合は、アクティベーションが大きすぎると勾配が爆発してしまうため、活性化関数の値域が無限まであっても、おおよそ図の範囲に収まる。

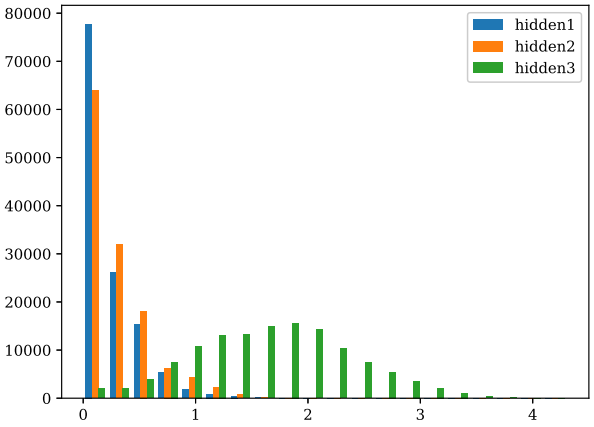


図 3.10: 隠れ層のユニットのアクティベーションの分布の例

3.4.3 学習結果のばらつき

ミニバッチの抽出およびドロップアウトするユニットの選択にランダム性があるために学習結果にはばらつきが存在する。図 3.11 は、図 2.9 と同様に作った残差ヒストグラムの標準偏差を、独立に学習させた値を並べたものである。誤差は不遍標準偏差としている。

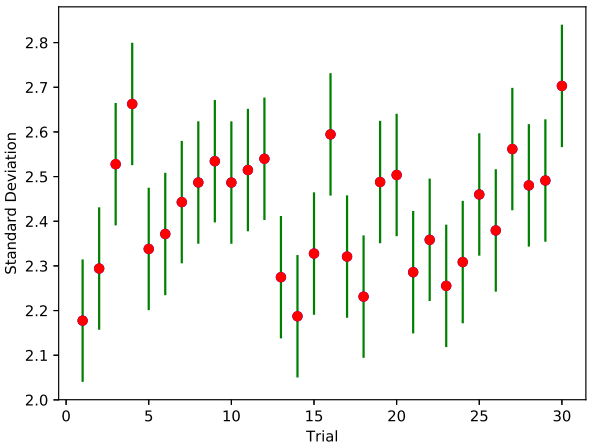


図 3.11: 学習結果のばらつきの例

第4章　機械学習によるHXD-PINバックグラウンドのモデル化

2.4.2 節で述べたように、天体観測中の HXD-PIN のバックグラウンドは、経験的かつ複雑なモデルで地没データをフィットして、時間方向に内挿して予測している。本研究では、経験的なモデルを機械学習によるモデルに置き換えることを目指す。本研究では、Google 社が開発しているオープンソースソフトウェアである TensorFlow ver 1.3.0 を用いて、以下に示すニューラルネットワークを構築した。

4.1　構築したニューラルネットワークの概要

本研究では、全結合ニューラルネットワークを採用した。入力層には 25 ユニット用意し、式 (2.1) と同様に「時刻 t」、「SAA ID」、「SAA からの経過時間」2 種類、「$PINUD$」、「COR」、「時間幅」、「GSO シンチレータの計数率」、「地球からの仰角」、「衛星の経度」、「衛星の緯度」、「HXD 光軸からの地磁気の天頂角 θ」「地磁気の方位角 ϕ」「$PINUD_{buildup}$」12 種類で、合計 25 パラメータを入力する。出力層は 1 ユニットで、時刻 t におけるバックグラウンドカウントである。最適な重みを持つネットワークを構築できれば、任意の時間 t において、その他 24 パラメータを実データから入力することで、バックグラウンドカウントを出力することができる。

入力層において、SAA からの経過時間が 2 種類あるのは、すざく衛星全体での定義と HXD での定義を両方使用しているからで、$PINUD_{buildup}$ が複数あるのは $PINUD$ を 12 種類の異異なる時定数で畳み込んでいるからである。時定数は参考文献 [1] の 3.2.1 節の方法で決定した。また事前処理として、すべての入力パラメータを [0,1] に規格化している。

同様に、出力パラメータであるバックグラウンドカウントも [0,1] に収まるように、256 で割り規格化している。

出力を比較し学習を評価する関数として、まずは χ^2 検定を採用した。ただし出力であるバックグラウンドカウントを 256 で割って規格化しているため、カウントに戻すように $256 \times \frac{1}{N} \sum (t-y)^2$ で、コストを計算した。ここで y は予測結果で、t は教師データである。

学習を効率良く進めるために、除外率 50%のドロップアウト (3.3.2 節) を適用し、さらにミニバッチ学習 (3.1.2 節) も使用して、50 個のデータをミニバッチとして入力した。実際の計算では、ミニバッチ学習を 200000 回繰り返し、その中でテストデータの損失関数が最も小さくなった重みを保存し、Early Stopping(3.3.2 節) を行った。

4.2　2007 年の観測地没データへの適用

まず手始めとして、2007 年の地没データに、構築した全結合ニューラルネットワークを適用する。地没データには、時間幅 256s でまとめたバックグラウンドカウントのライトカーブが含まれており、時間ビンの数 (全データ数) は 26467 個である。それぞれの時間ビンに対して、「時刻 t」や「SSA ID」などの 25 入力パラメータがある。

全 26467 個の時間ビンの中から、ランダムに 2000 個のビンを選び出しテストデータとした。残り 24467 個のデータを、教師データとして使用し、機械学習を行った。

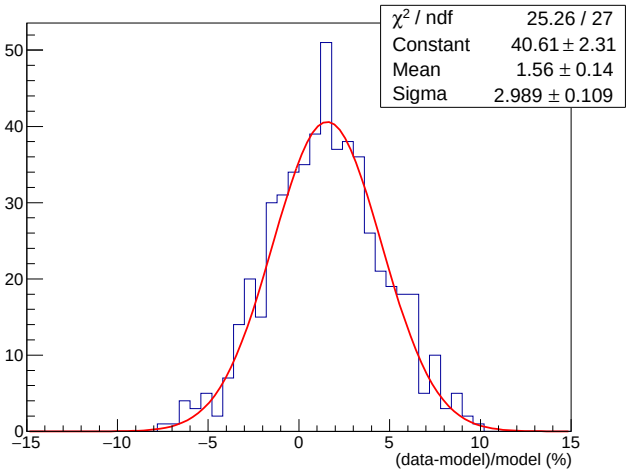
4.2.1　ひな形ネットワーク

隠れ層の層数や各層のユニット数、損失関数や活性化関数の選択、重みの初期化方法など、これらの組み合わせでネットワークは無限に構築できるが、まずは表 4.1 に示したように 3 層の隠れ層を用いた。最初にこのネットワークを評価し、ネットワークパラメータを一つずつ変えて比較することで、最良のネットワークを探索する。

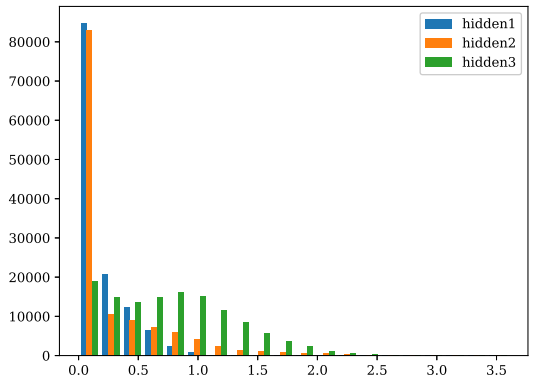
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	128	ReLU	標準偏差 0.01
隠れ層 2	128	ReLU	標準偏差 0.01
隠れ層 3	128	ReLU	標準偏差 0.01
出力層	1	ReLU	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (t-y)^2$
最適化法	Adam
学習率	0.0001

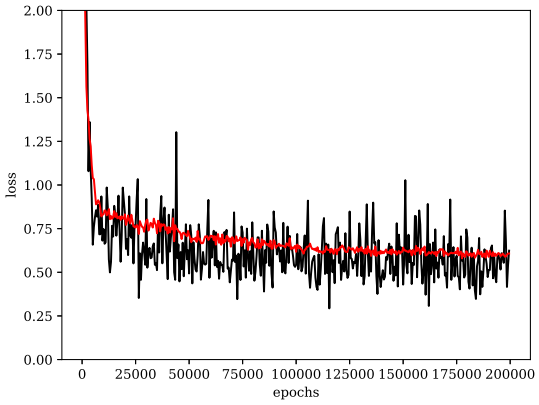
表 4.1: 初期のニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布
隠れ層 1(青)、隠れ層 2(橙)、隠れ層 3(緑)



損失関数の推移
学習データ (黒)、テストデータ (赤)

図 4.1: 初期のニューラルネットワークでの学習結果の例

ひな形ネットワークでの学習結果を、図 4.1 に示す。テストデータを用いて、ネットワークの出力と正解のバックグラウンドカウントの残差を見ると、中心が正にずれている。このネットワークの予測値では、バックグラウンドを過小評価していることを意味する。

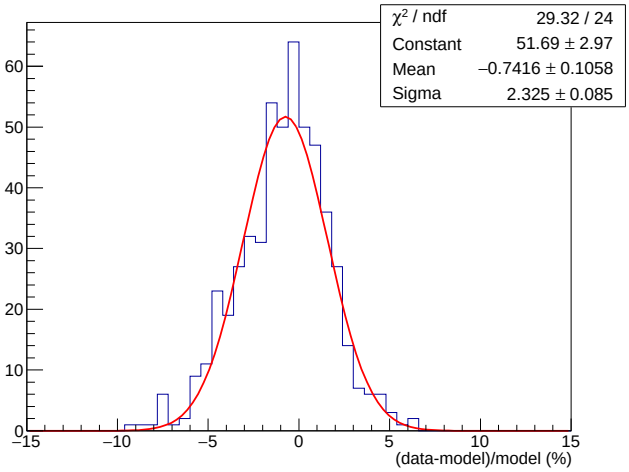
4.2.2 活性化関数を変えたニューラルネットワーク

ひな形のニューラルネットワークでは、出力の平均が性にずれていたので、それを修正するために活性化関数を変更した。変更した活性化関数は出力層のみで、ReLU からソフトプラス関数 $f(x) = \log(1+\exp(x))$ に変えた。ネットワークの構造を表 4.2 に示す。

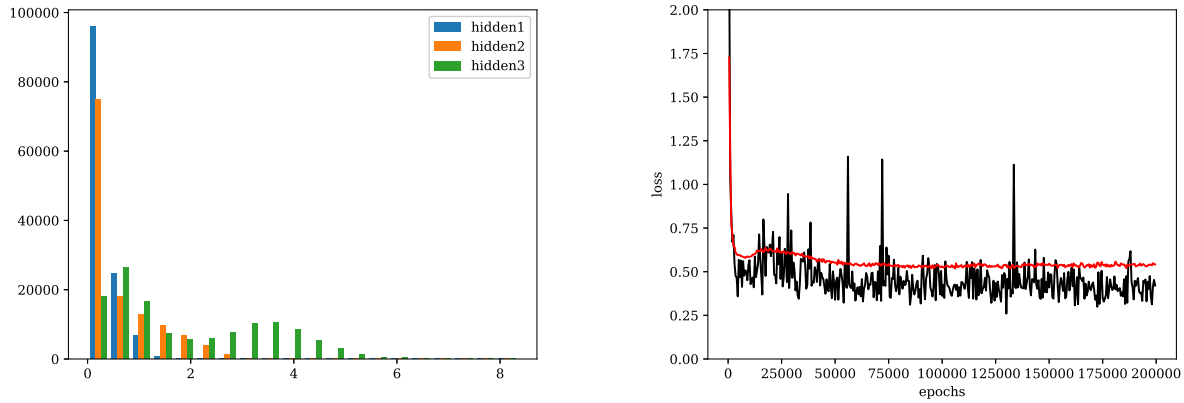
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	128	ReLU	標準偏差 0.01
隠れ層 2	128	ReLU	標準偏差 0.01
隠れ層 3	128	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (t-y)^2$
最適化法	Adam
学習率	0.0001

表 4.2: 活性化関数を変えたニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布　　損失関数の推移

図 4.2: 活性化関数を変えたニューラルネットワークでの学習結果の例

修正したネットワークにおける学習結果を、図 4.2 に示す。期待通りに出力の平均値は、ゼロ付近になっている。さらに早期に損失関数が減少していること、アクティベーションの分布が広がっていることがわかる。

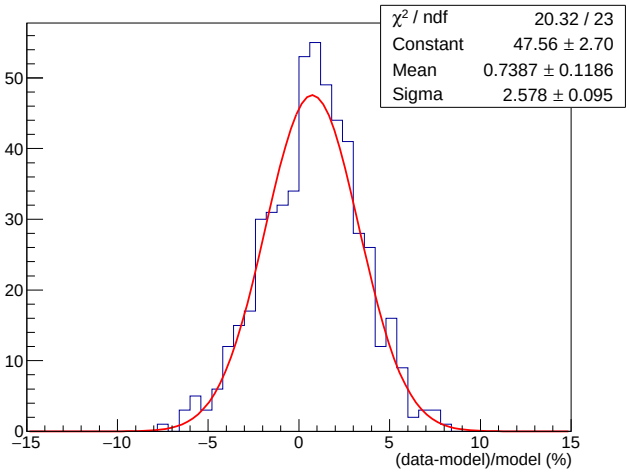
4.2.3 損失関数を変更したニューラルネットワーク

前節のニューラルネットワークでは、損失関数として χ^2 検定を採用したが、$256 \times \frac{1}{N}\sum(y - t + y \times (\log t - \log y))$ に変更した。この関数は cstat[10] とよばれており、ポアソン分布による統計誤差を考慮している。今回のようにバックグラウンドカウントがが小さい時は、統計誤差は正規分布ではなくポアソン分布に従うため、に有効である。ネットワークの構造を表 4.3 に示す。

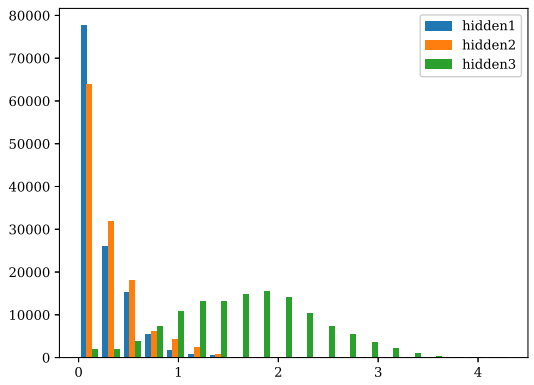
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	128	ReLU	標準偏差 0.01
隠れ層 2	128	ReLU	標準偏差 0.01
隠れ層 3	128	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N}\sum(y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

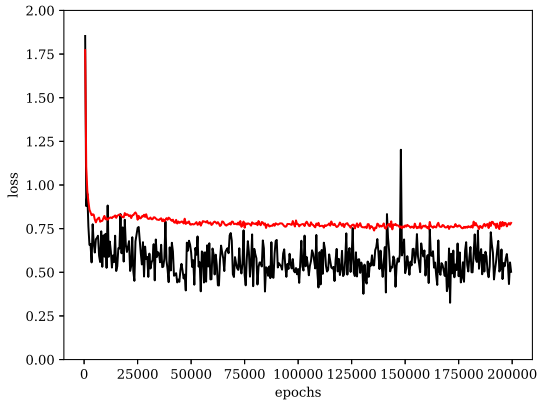
表 4.3: 損失関数を変えたニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布



損失関数の推移

図 4.3: 損失関数を変えたニューラルネットワークでの学習結果の例

図 4.3 を見ると、残差ヒストグラムのずれがやや改善したことがわかる。

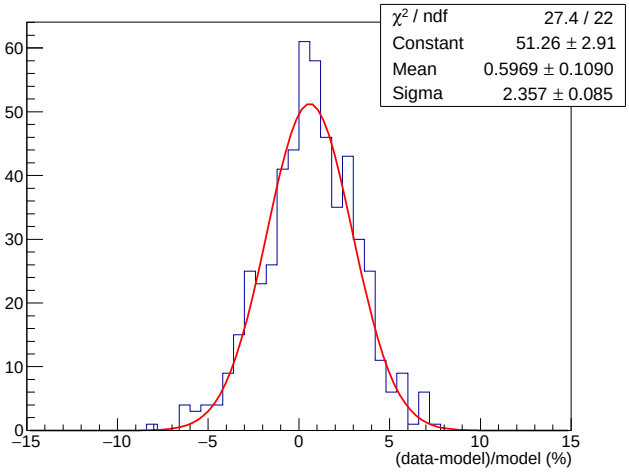
4.2.4 層の数を変更したニューラルネットワーク

前節までのアクティベーションの分布を見ると隠れ層 1 層目のアクティベーションがほとんど 0 になっていることがわかる。そのため、隠れ層は 3 つも必要ないと判断して層の数を減らして学習させた。隠れ層を 2 層にしたネットワークの構造を表 4.4 に、その学習結果を図 4.4 に示す。残差ヒストグラムの標準偏差は、隠れ層 3 つのニューラルネットワークと比べて、同等の値を得た。

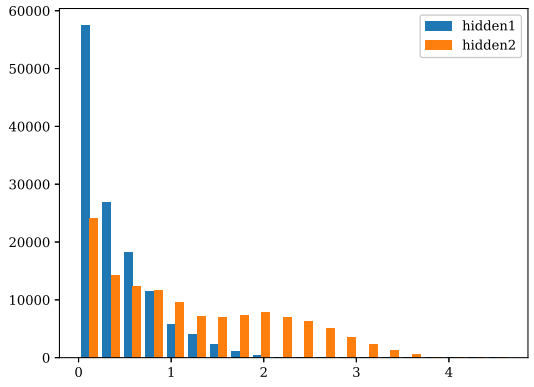
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	128	ReLU	標準偏差 0.01
隠れ層 2	128	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

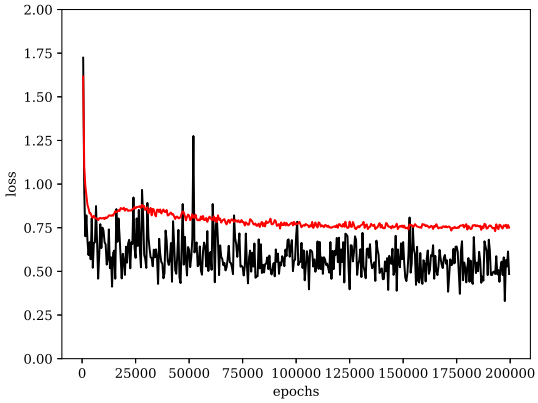
表 4.4: 隠れ層 2 つのニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布



損失関数の推移

図 4.4: 隠れ層 2 つのニューラルネットワークでの学習結果の例

さらに隠れ層を減らして、隠れ層 1 つのネットワークの構造を表 4.5 に、学習結果を図 4.5 にまとめた。アクティベーションの分布が狭くなっているものの、残差ヒストグラムの標準偏差は隠れ層が 2 層のときと大差ない。

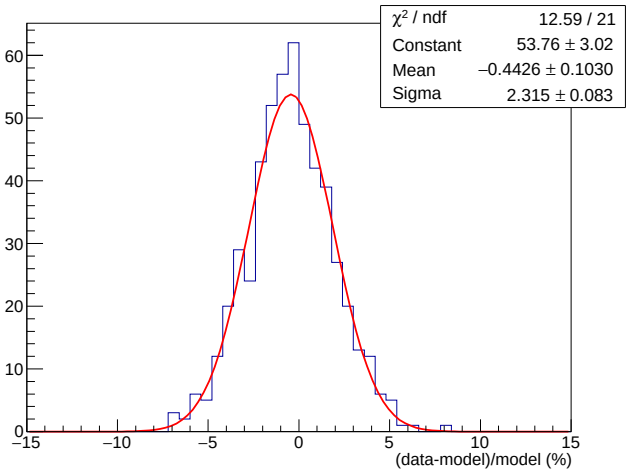
学習結果は 3.4.3 で述べたように、重みの初期値やドロップアウトの乱数でばらつく。そこで、乱数の種

以外は同一のネットワークで学習を独立に 30 回行った。各学習結果に対する残差ヒストグラムの標準偏差をプロットしたものが図 4.6 であり、隠れ層 2 つが最も低い値で安定していることがわかる。そこで、以後のニューラルネットワークでは隠れ層を 2 つとした。

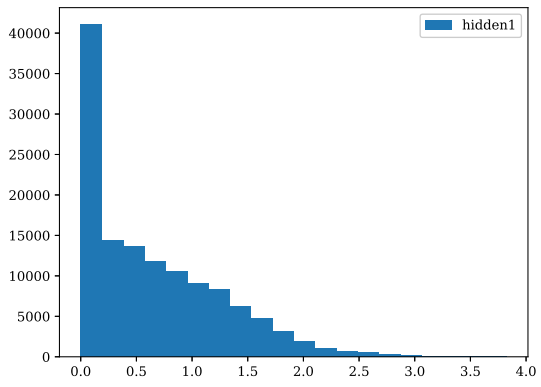
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	128	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

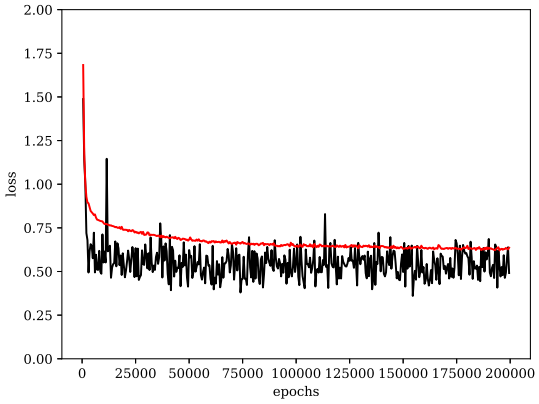
表 4.5: 隠れ層 1 つのニューラルネットワーク



部分残差ヒストグラム (10ks 積分)

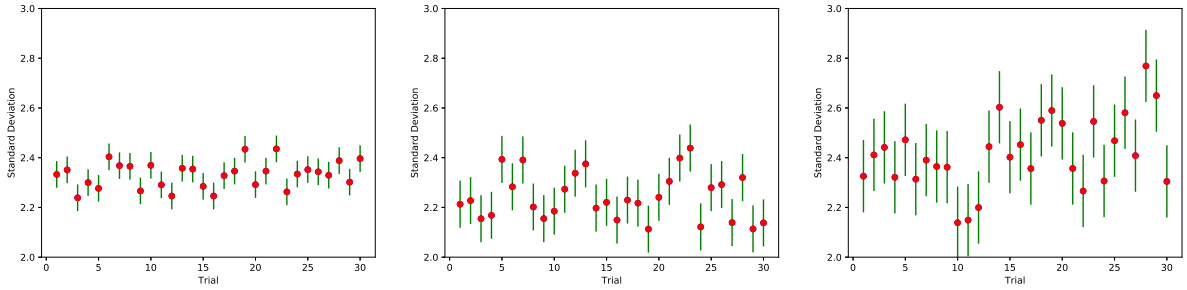


アクティベーションの分布



損失関数の推移

図 4.5: 隠れ層 1 つのニューラルネットワークでの学習結果の例



隠れ層 1 つ　　隠れ層 2 つ　　隠れ層 3 つ

図 4.6: 学習のばらつきの比較

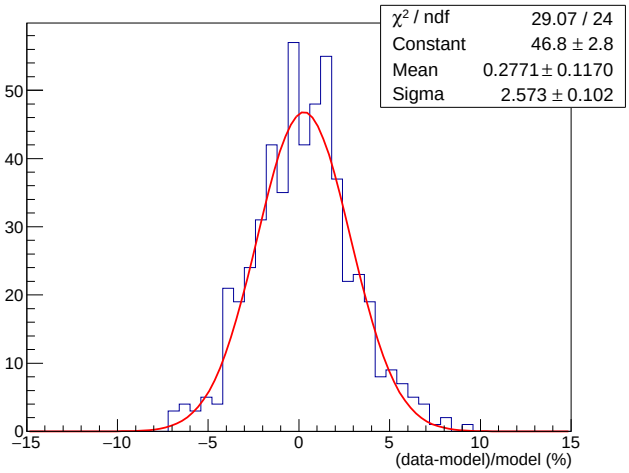
4.2.5　隠れ層のユニットの数を変更したニューラルネットワーク

隠れ層の数は 2 つが最適であることがわかったので、次にユニット数を変化させて学習させた。ユニット数を 128 から 64 に減らしたネットワークの構造を表 4.6 に示す。学習結果をまとめた図 4.7 を見ると、残差ヒストグラムの標準偏差は大きくなり、モデルの不定性が大きくなったことがわかる。これはユニット数を減らしたことで活性化しているユニット数の総数が減ってしまったことが原因だと考えられる。

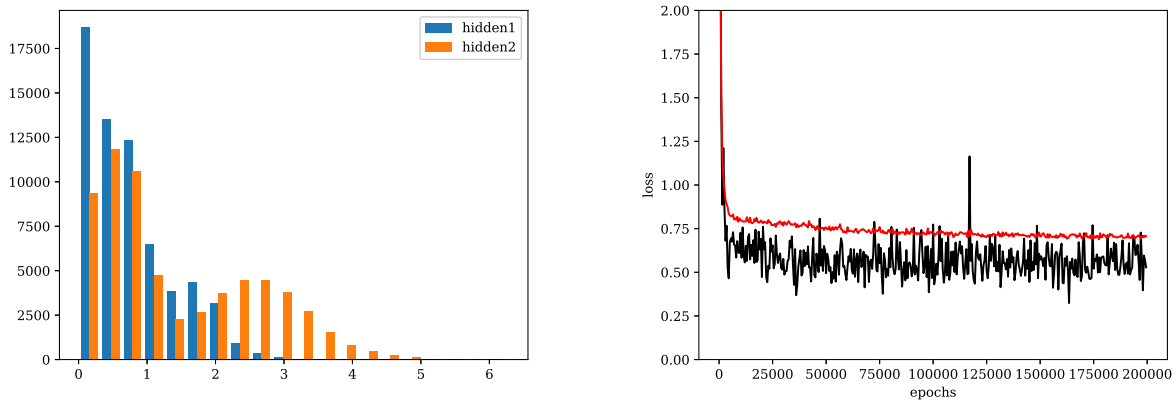
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	64	ReLU	標準偏差 0.01
隠れ層 2	64	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum(y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

表 4.6: ユニット数 64 のニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布　　損失関数の推移

図 4.7: ユニット数 64 のニューラルネットワークでの学習結果の例

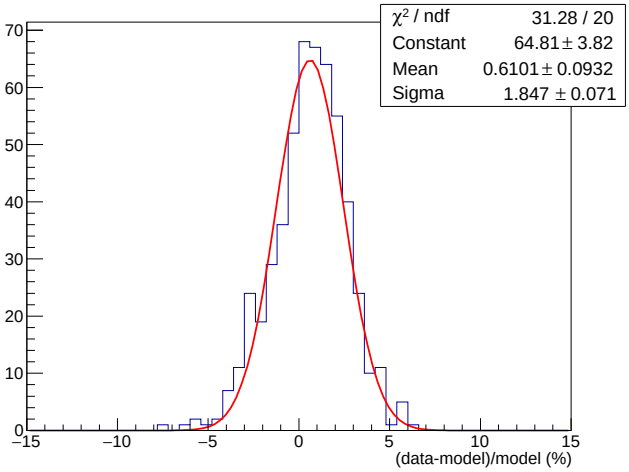
今度はユニット数を 200 に増やしたネットワークを試した。そのネットワーク構造を表 4.7 に、学習結果を図 4.8 に示す。残差ヒストグラムの標準偏差が小さくなっていることがわかる。今までのニューラルネットワークでは最後の隠れ層のアクティベーションは全体的に 10000 ほどだったが、今回の学習では約 20000 となっておりそれが改善の要因だと考えられる。

さらにユニット数を増やしたネットワークの構造を表 4.8 に、学習結果を図 4.9 に示す。損失関数の推移を見ると、学習データでは損失関数は下がり続けているが、テストデータではやや右肩上がりになっていて、過学習が起きていることがわかる。ユニット数 300 でも過学習は見えたので、以後のニューラルネットワークではユニット数を 200 とした。

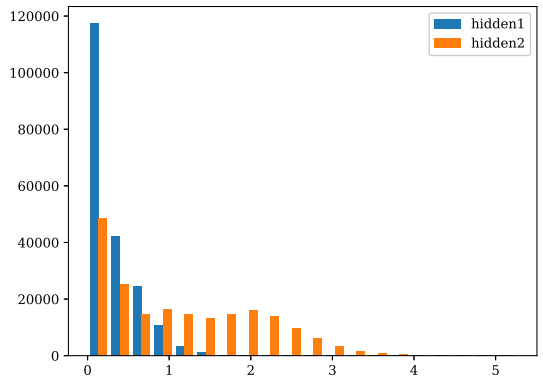
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	200	ReLU	標準偏差 0.01
隠れ層 2	200	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

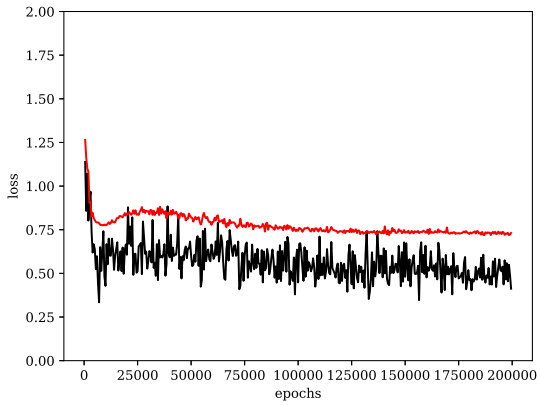
表 4.7: ユニット数 200 のニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布



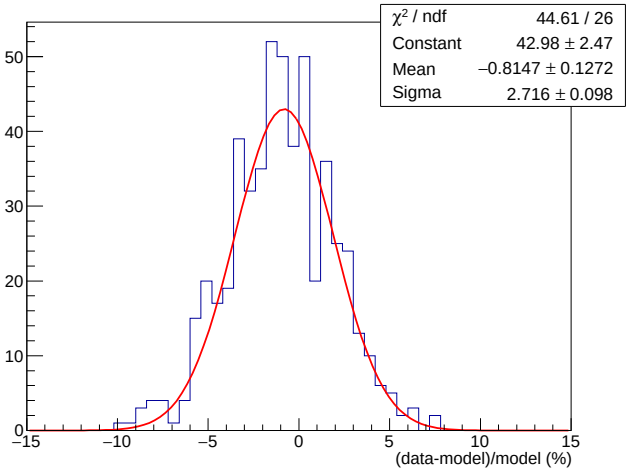
損失関数の推移

図 4.8: ユニット数 200 のニューラルネットワークでの学習結果の例

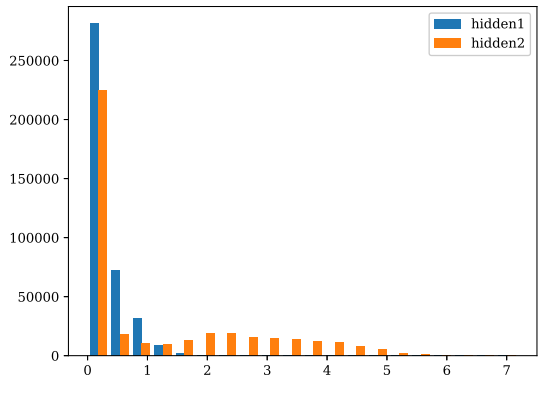
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	400	ReLU	標準偏差 0.01
隠れ層 2	400	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

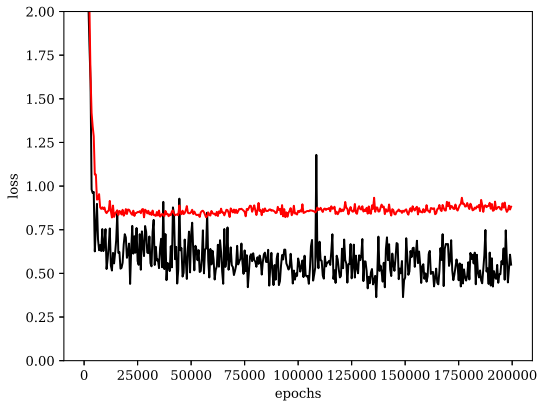
表 4.8: ユニット数 400 のニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布



損失関数の推移

図 4.9: ユニット数 400 のニューラルネットワークでの学習結果の例

4.2.6 重みの初期値を変更したニューラルネットワーク

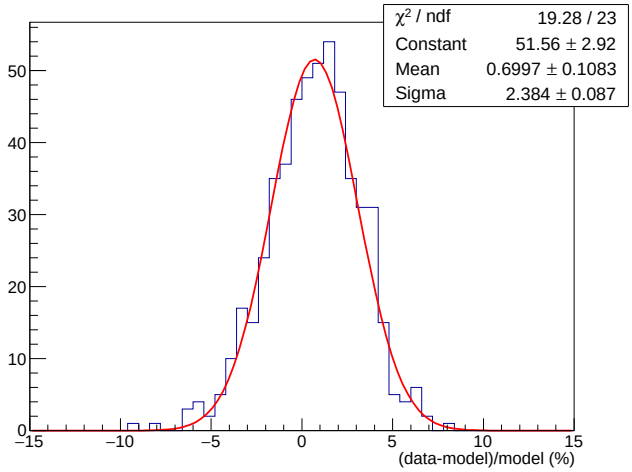
最良なネットワークの構成が見つかったので、次に重みの初期値による学習結果への影響を調べる。

隠れ層では活性化関数に ReLU を用いているため、重みの初期値を He の初期値に変更して学習させた。ソフトプラスの値域も $(0, \infty)$ であるため He の初期値を適用してある。ネットワークの構造を表 4.9 に示す。

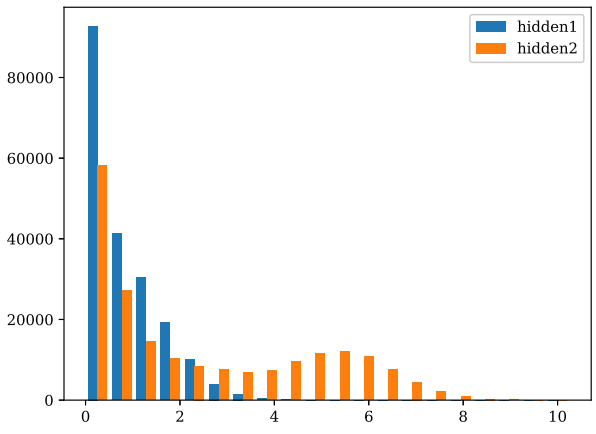
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	200	ReLU	He の初期値
隠れ層 2	200	ReLU	He の初期値
出力層	1	ソフトプラス	He の初期値

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

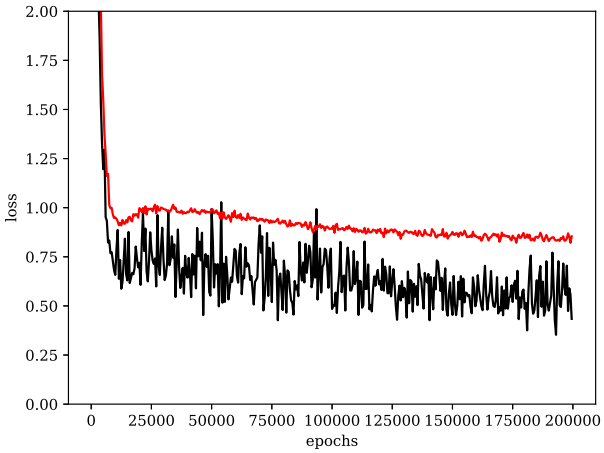
表 4.9: He の初期値を用いたニューラルネットワーク



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布



損失関数の推移

図 4.10: He の初期値を用いたニューラルネットワークでの学習結果の例

図 4.10 は He の初期値による学習結果であり、アクティベーションの分布がすべての隠れ層において広がっていることがわかる。ユニットの数を前節と同様に変化させてみたが、その結果も前節と同様であったため、He の初期値を利用しても結果は改善しなかった。

4.2.7 オプティマイザを変更したニューラルネットワーク

これまでオプティマイザは Adam を使用していたが、最適か確かめるために確率的勾配下降法、モメンタム法、AdaGrad 法でも学習を行った。それぞれのネットワークの構造を表 4.10、表 4.11、表 4.12 に示す。そして対応する学習結果を図 4.11 に示す。Adam 以外のオプティマイザでは、残差ヒストグラムの標準偏差、つまりモデルの不定性が大きくなった。これは、極小解に囚われたり AdaGrad のパラメータ H が学習中に大きくなりすぎて、重みの更新ができなくなるからだと考えられる。

	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	200	ReLU	標準偏差 0.01
隠れ層 2	200	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	確率的勾配降下法
学習率	0.0001

表 4.10: 確率的勾配降下法を用いたニューラルネットワーク

	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	200	ReLU	標準偏差 0.01
隠れ層 2	200	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

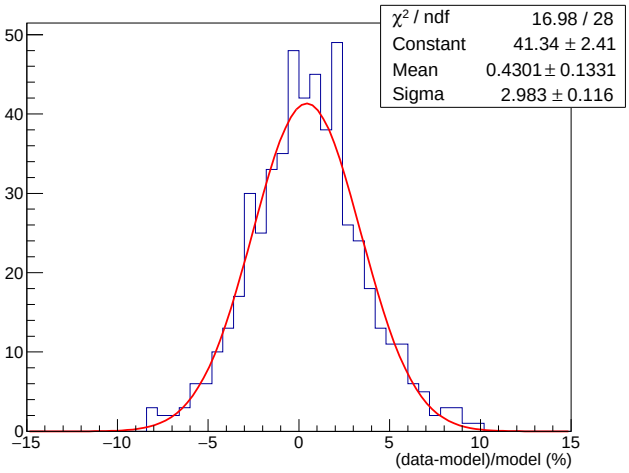
損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	モメンタム
学習率	0.0001
係数 α	0.5

表 4.11: モメンタムを用いたニューラルネットワーク

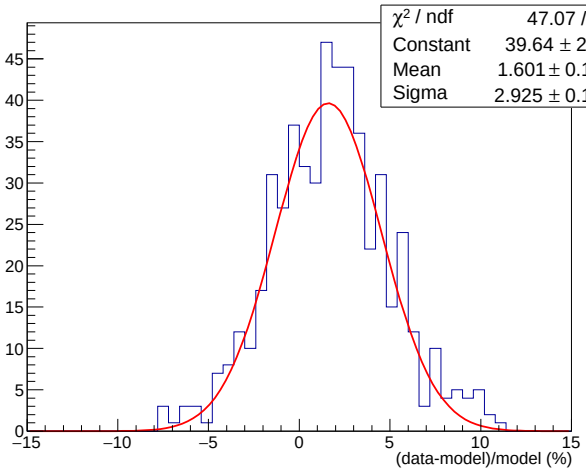
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	200	ReLU	標準偏差 0.01
隠れ層 2	200	ReLU	標準偏差 0.01
出力層	1	ソフトプラス	標準偏差 0.01

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	AdaGrad
学習率	0.0001

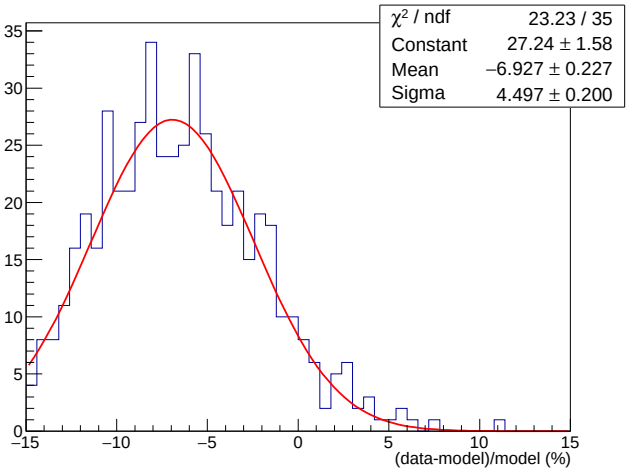
表 4.12: AdaGrad を用いたニューラルネットワーク



確率的勾配降下法



モメンタム



AdaGrad

図 4.11: 他のオプティマイザを用いた結果の例

4.2.8 最良のニューラルネットワーク

以上の中で最も良いニューラルネットワークは表 4.7 に示したものとなった。このニューラルネットワークの学習のばらつきを図 4.12 に示す。また、地没データの積分時間を変化させた時のモデルの不定性を、表 4.13 に示す。図 4.12 を見ると、標準偏差は 2.0 付近に集中していることがわかる。表 2.5 で示した通り、今までの経験的にモデル化したバックグラウンドモデルの不定性は、積分時間 10ks において bgd-a で 3.75%、bgd-d で 2.31%である。これらと比較すると、機械学習を用いたモデルの精度は bgd-a より優れていて、bgd-d と同等であることがわかる。

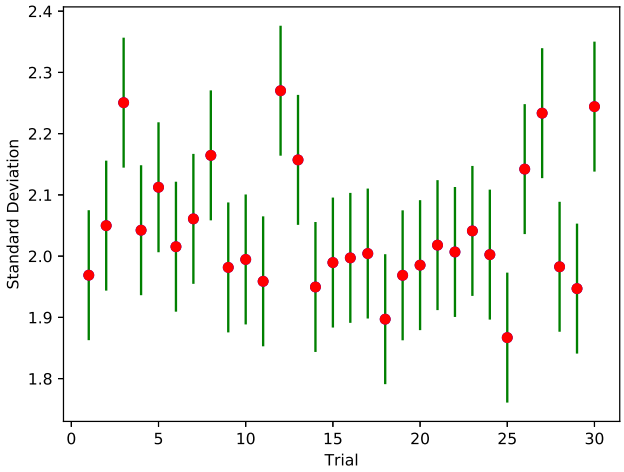


図 4.12: 学習結果のばらつき (10ks 積分)

12-70keV			
積分時間	10ks	20ks	40ks
標準誤差	2.06%	1.67%	1.46%

表 4.13: 各時間毎の誤差の値の例

4.3 各入力パラメータのバックグラウンドモデルへの寄与

ニューラルネットワークが自動的に最適な重みを決定したが、入力データがどのように寄与しているのかこのままでは不明である。そのため、バックグラウンドモデルを出力する際に入力データを 1 種類だけ 0 (つまり入力層のユニットを 1 つだけ除外) にしてバックグラウンドモデルを作成した。表 4.14 に示すのは、各入力データを除外したときの部分残差ヒストグラムのガウスフィットパラメータである。

パラメータ	平均	(平均) 除外なしとの差	標準誤差	(標準誤差) 除外なしとの差
除外なし	0.26		2.08	
時刻	-0.72	-0.98	3.20	1.12
SAA ID	-0.64	-0.90	2.70	0.62
SAA からの時間 (すざく)	0.42	0.16	2.19	0.11
PINUD	23.06	22.80	6.65	4.57
COR	-12.71	-12.97	4.33	2.25
GSO シンチレータの計数率	3.77	3.51	3.00	0.92
時間幅	14.65	14.40	0.57	-1.51
SAA からの時間 (HXD)	-0.13	-0.39	2.65	0.58
地球からの仰角	0.34	0.08	2.40	0.33
経度	0.15	-0.10	2.49	0.41
緯度	-0.67	-0.93	3.00	0.93
地磁気の θ 角	0.23	-0.02	2.61	0.54
地磁気の ϕ 角	-0.08	-0.33	2.44	0.36
$PINUD_{buildup}1$	1.23	0.97	2.33	0.25
$PINUD_{buildup}2$	5.06	4.80	3.58	1.51
$PINUD_{buildup}3$	0.50	0.24	2.07	-0.01
$PINUD_{buildup}4$	1.60	1.35	2.24	0.17
$PINUD_{buildup}5$	-5.62	-5.87	3.10	1.02
$PINUD_{buildup}6$	-1.20	-1.46	3.07	1.00
$PINUD_{buildup}7$	3.16	2.90	3.54	1.46
$PINUD_{buildup}8$	-2.81	-3.06	3.35	1.27
$PINUD_{buildup}9$	4.82	4.57	3.18	1.10
$PINUD_{buildup}10$	-4.43	-4.68	4.51	2.43
$PINUD_{buildup}11$	-3.89	-4.15	3.35	1.27
$PINUD_{buildup}12$	0.56	0.30	2.68	0.60

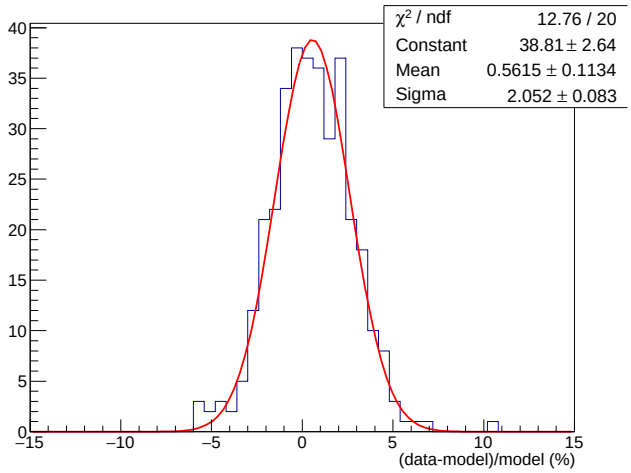
表 4.14: パラメータごとの依存性の例 (10ks 積分)

部分残差ヒストグラムは $\frac{(\text{実データ})-(\text{バックグラウンドモデル})}{(\text{バックグラウンドモデル})}$ となっているため、平均が 0 から大きくずれている (1 以上、$PINUD$、COR、GSO シンチレータの計数率、時間幅、$PINUD_{buildup}1$、$PINUD_{buildup}2$、$PINUD_{buildup}4$、$PINUD_{buildup}5$、$PINUD_{buildup}6$、$PINUD_{buildup}7$、$PINUD_{buildup}8$、$PINUD_{buildup}9$、$PINUD_{buildup}10$、$PINUD_{buildup}11$ はバックグラウンドを良く表しているパラメータであると考えられる。

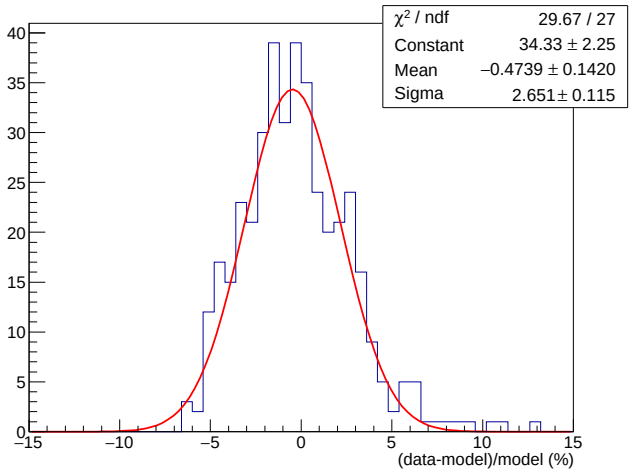
4.4　他の年度への適用

4.4.1　他の年度に適用した結果

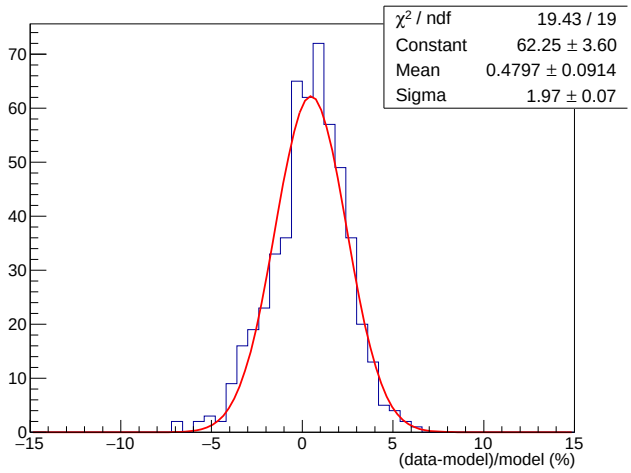
前述した通り、今まで学習に用いていたのは 2007 年のデータであったので、2005〜2014 年のデータもそれぞれ表 4.7 のニューラルネットワークで学習させた。各年における学習結果を図 4.13 に示す。



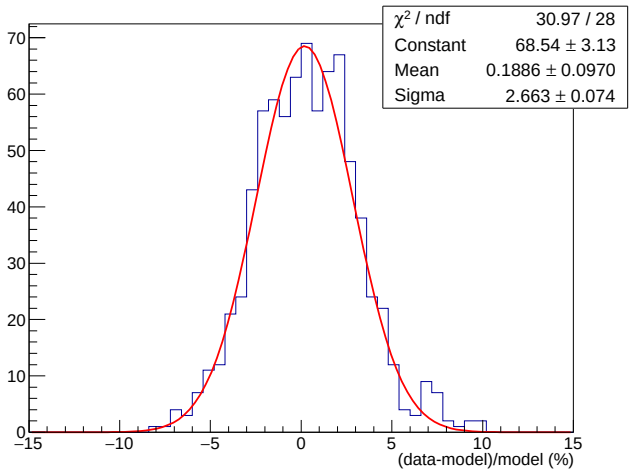
2005 年



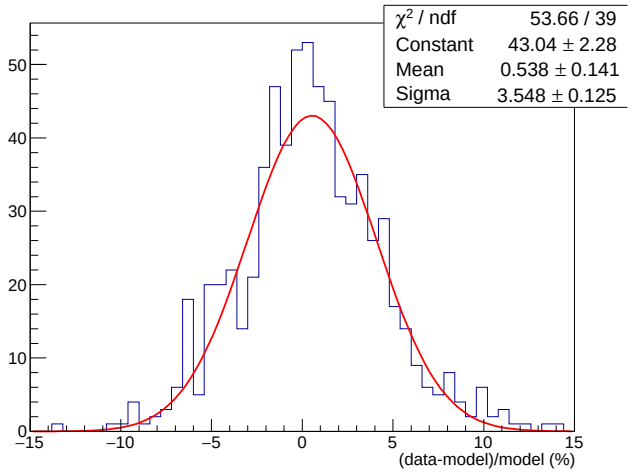
2006 年



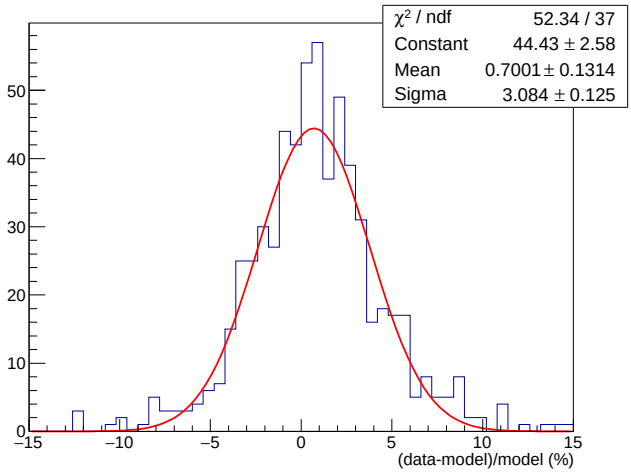
2007 年



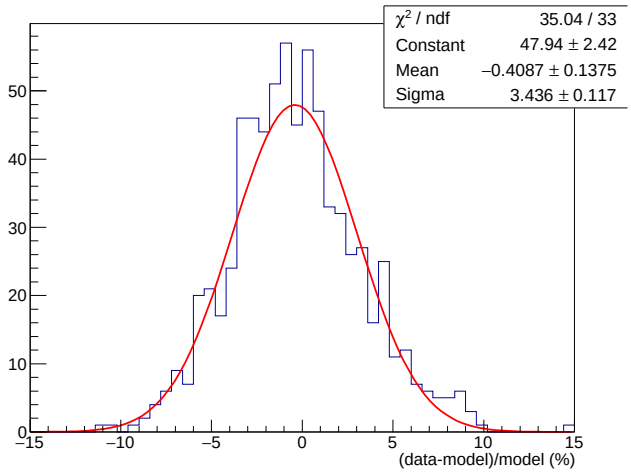
2008 年



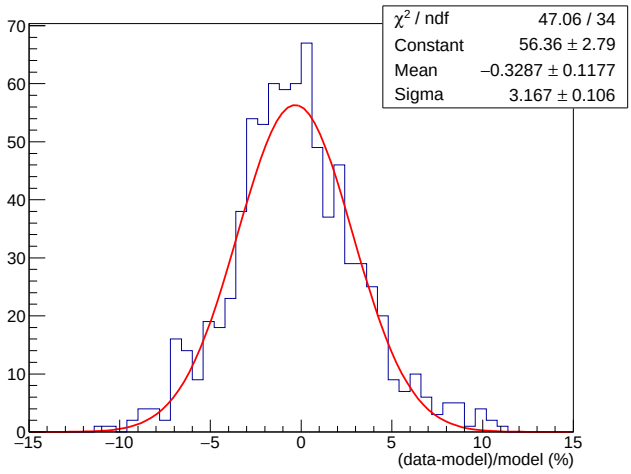
2009 年



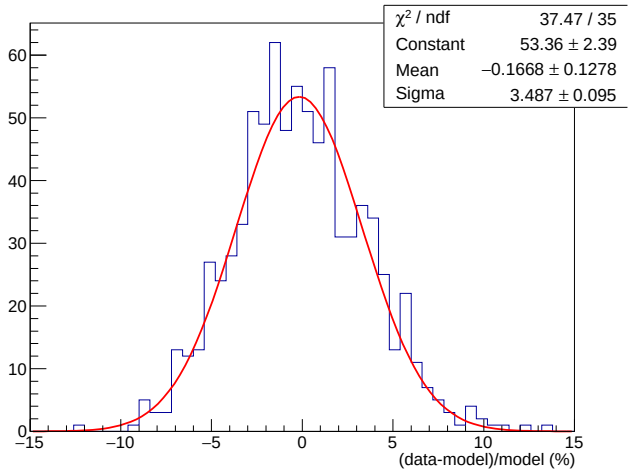
2010 年



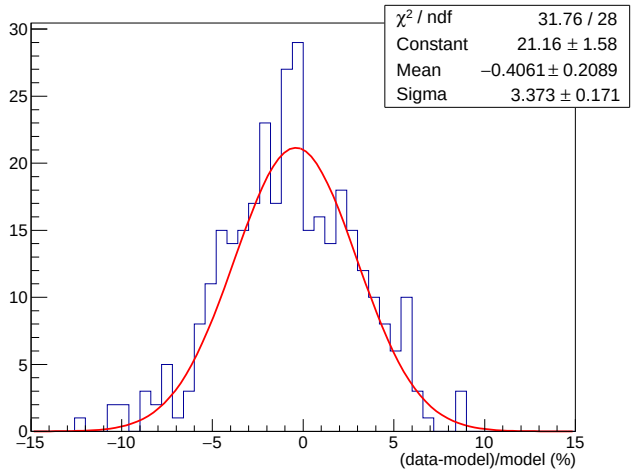
2011 年



2012 年



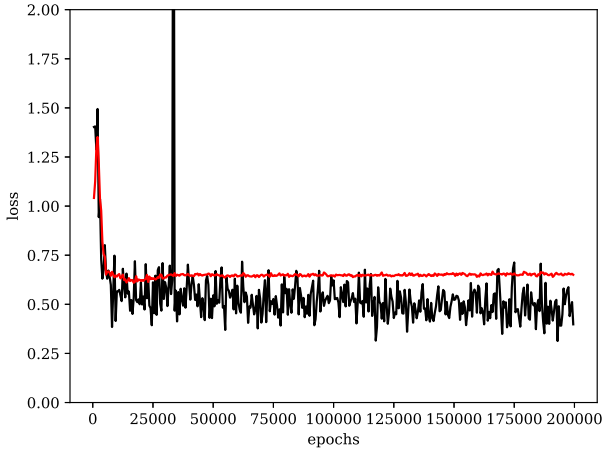
2013 年



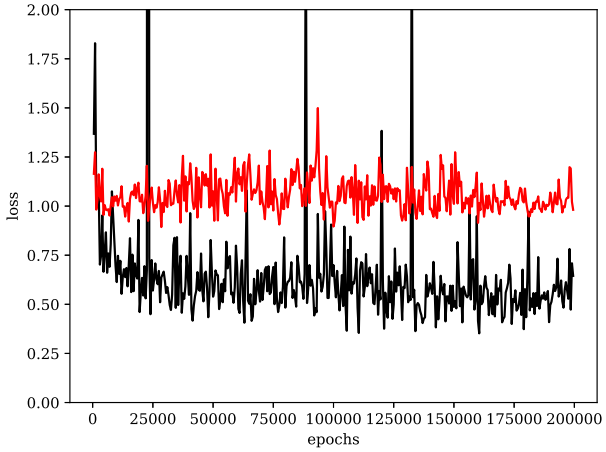
2014 年

図 4.13: 各年ごとの学習結果の例 (10ks 積分)

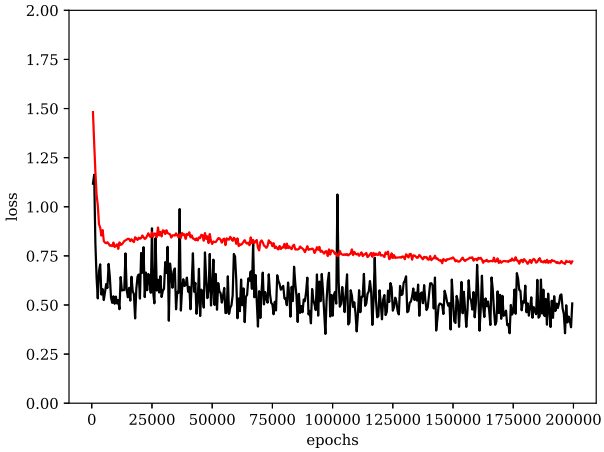
2010 年、2012 年、2014 年は平均が 0 から大きくずれてしまっている。その他の年も、2005 年はデータ数が少ないため標準偏差が小さくなっているが、他の年では 2007 年ほどの精度が出ていないことがわかる。図 4.14 に各年の損失関数の推移を示す。



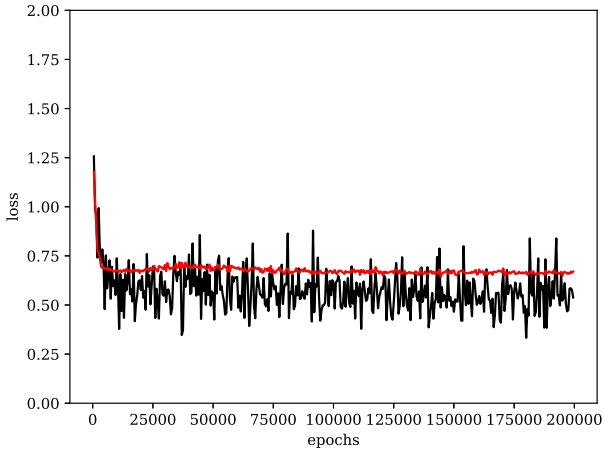
2005 年



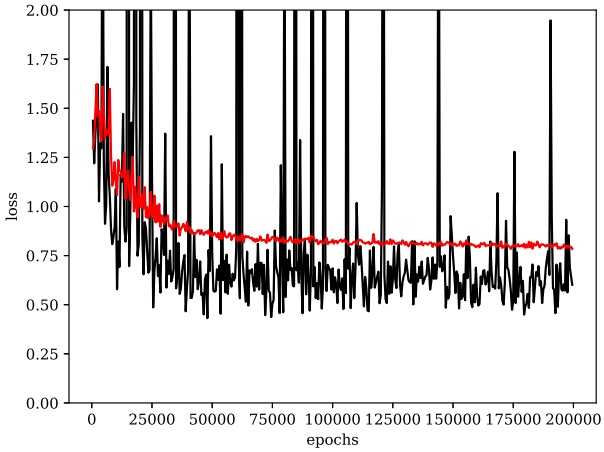
2006 年



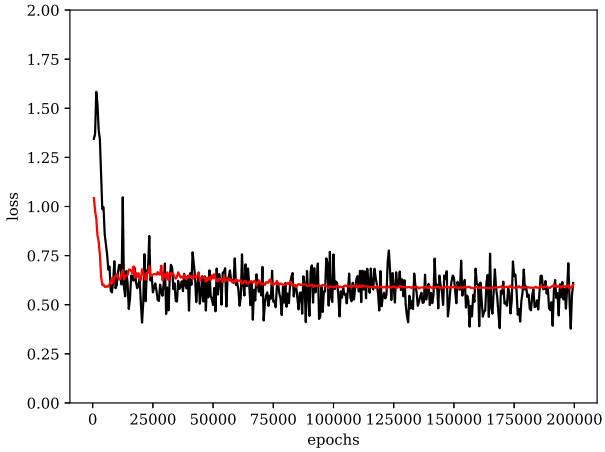
2007 年



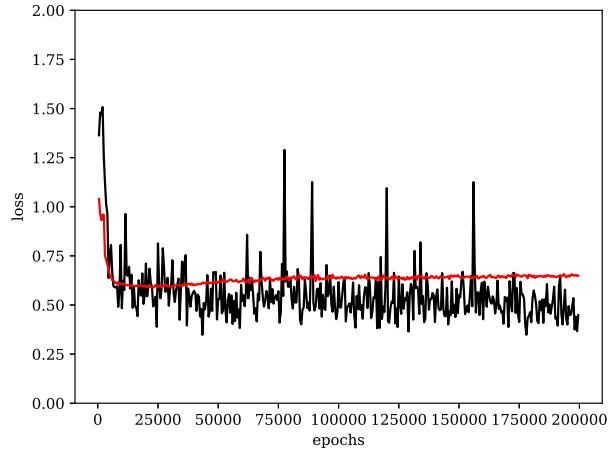
2008 年



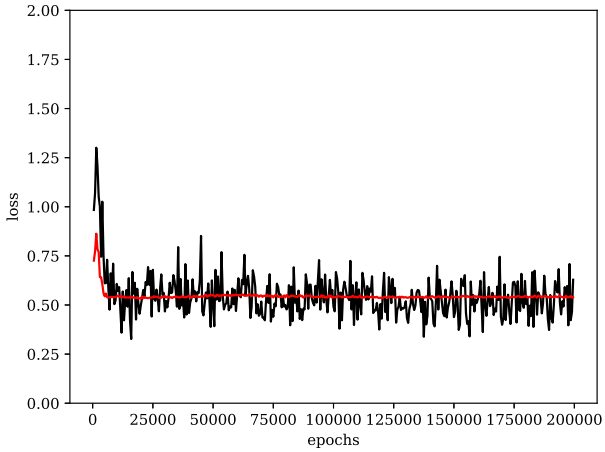
2009 年



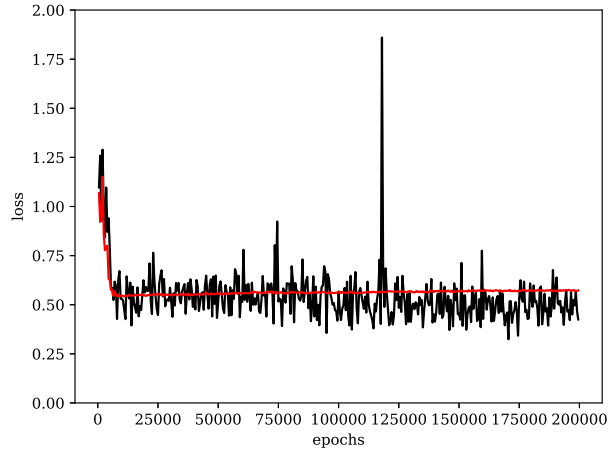
2010 年



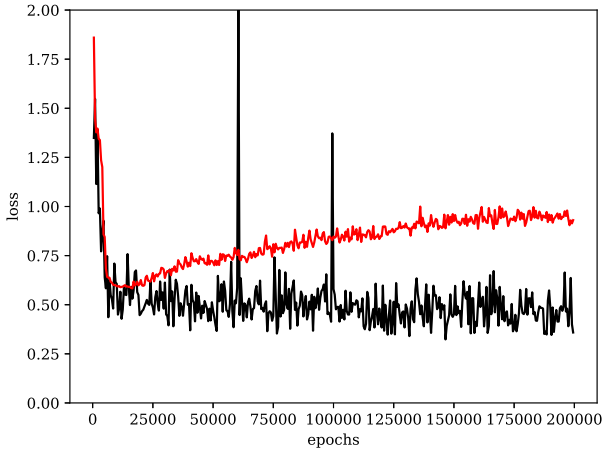
2011 年



2012 年



2013 年



2014 年

図 4.14: 各年ごとの損失関数の推移の比較

結果の悪かった年の 1 つである 2010 年の、入力データのバックグラウンドモデルへの寄与を表 4.15 に示す。影響の強いパラメータは 2007 年と同様だった。

パラメータ	平均	(平均) 除外なしとの差	標準誤差	(標準誤差) 除外なしとの差
除外なし	0.81		3.35	
時刻	-3.41	-4.21	6.66	3.31
SAA ID	3.12	2.31	3.93	0.58
SAA からの時間 (すざく)	0.74	-0.07	3.79	0.44
PINUD	185.62	184.81	50.61	47.25
COR	-10.40	-11.21	7.63	4.28
GSO シンチレータの計数率	14.30	13.50	5.46	2.11
時間幅	76.08	75.28	25.82	22.47
SAA からの時間 (HXD)	-0.36	-1.17	3.56	0.20
地球からの仰角	1.09	0.29	4.59	1.24
経度	0.31	-0.50	4.24	0.89
緯度	-1.85	-2.66	4.31	0.96
地磁気の θ 角	0.27	-0.54	4.10	0.74
地磁気の ϕ 角	0.23	-0.58	3.99	0.64
$PINUD_{buildup}1$	1.04	0.23	3.44	0.09
$PINUD_{buildup}2$	8.24	7.43	5.44	2.09
$PINUD_{buildup}3$	0.87	0.06	3.47	0.11
$PINUD_{buildup}4$	0.59	-0.22	3.49	0.14
$PINUD_{buildup}5$	-3.65	-4.46	4.87	1.52
$PINUD_{buildup}6$	0.23	-0.58	5.21	1.86
$PINUD_{buildup}7$	1.48	0.67	3.58	0.23
$PINUD_{buildup}8$	3.63	2.83	5.05	1.70
$PINUD_{buildup}9$	1.09	0.29	4.80	1.45
$PINUD_{buildup}10$	-0.82	-1.63	6.82	3.46
$PINUD_{buildup}11$	2.90	2.09	4.17	0.82
$PINUD_{buildup}12$	7.65	6.85	5.87	2.52

表 4.15: 2010 年のパラメータごとの依存性の例 (10ks 積分)

また、2010 年の学習のばらつきを図 4.15 と図 4.16 に示す。平均が約 1 で標準偏差が約 3 のグループと平均が約-4 で標準偏差が約 5 のグループに分けられる。これは、２つの極小値が存在し、学習毎にどちらかに囚われていることを示している。

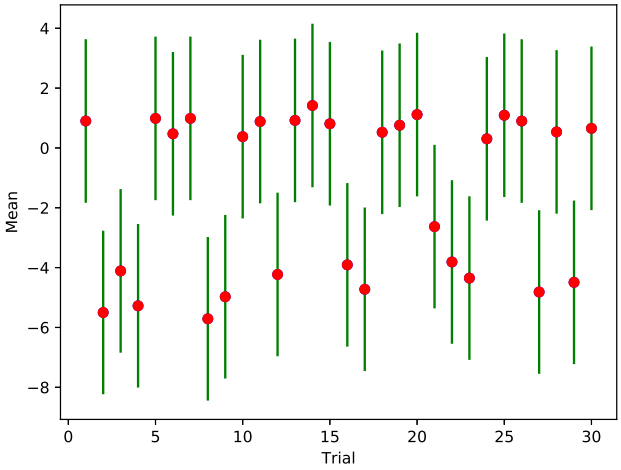


図 4.15: 2010 年の平均のばらつき (10ks 積分)

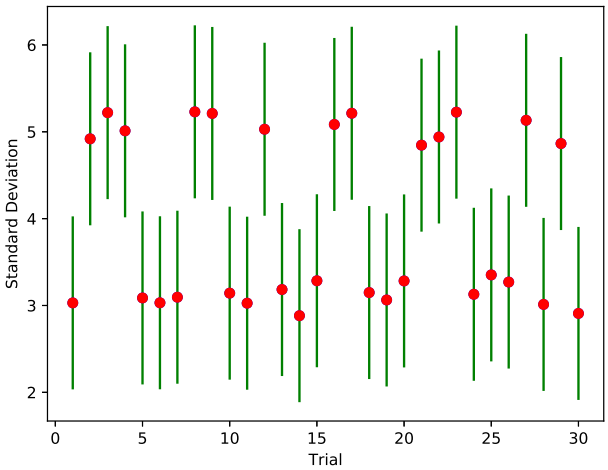
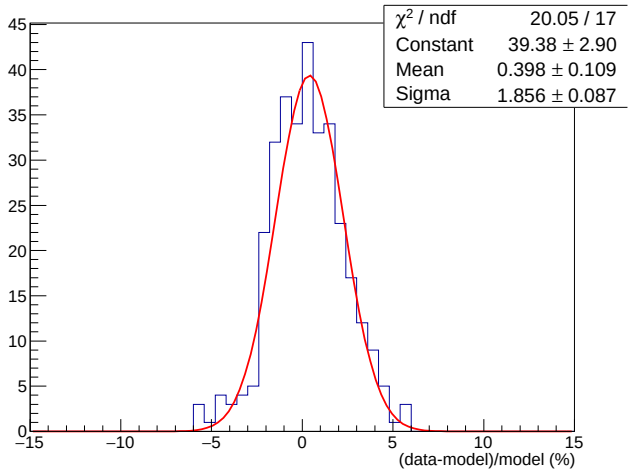


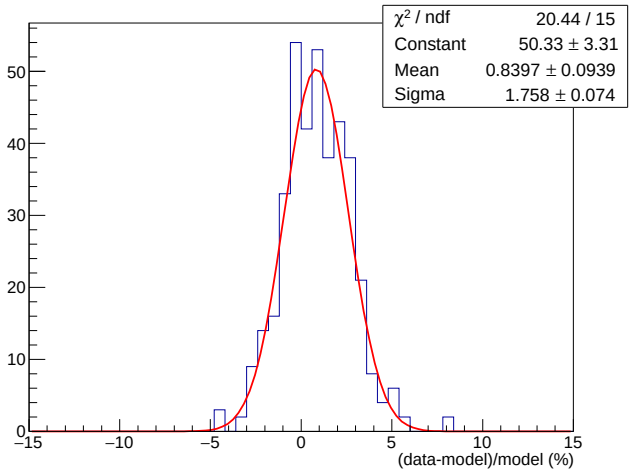
図 4.16: 2010 年の標準偏差のばらつき (10ks 積分)

4.4.2 外れ値を除外して適用

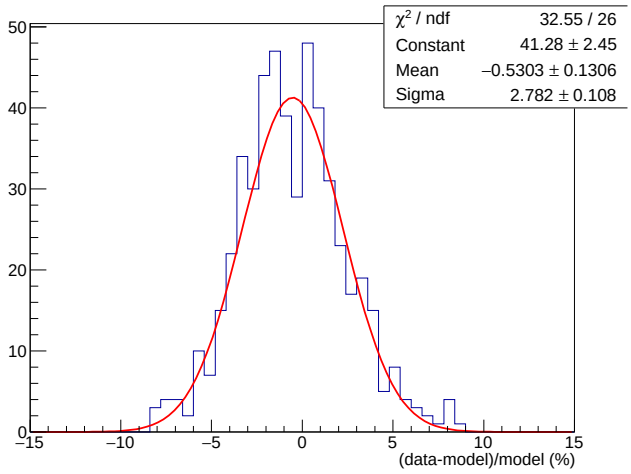
外れ値が学習に悪影響を与えている可能性があるので、一度学習させたあと 50%以上もモデルとデータが合わなかったデータを取り除いて、再び学習させた。結果を図 4.17 と表 4.16 に示す。



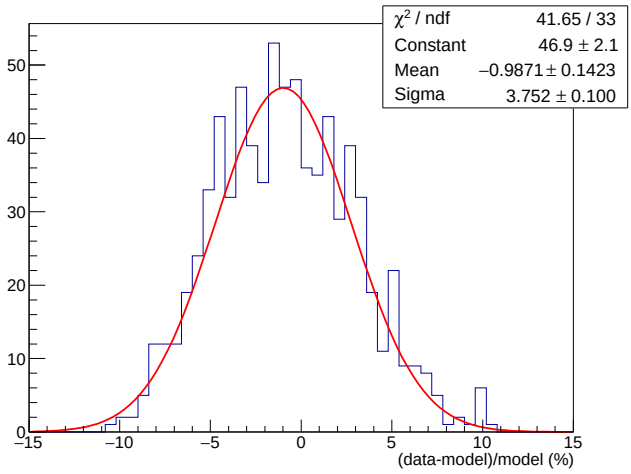
2005 年



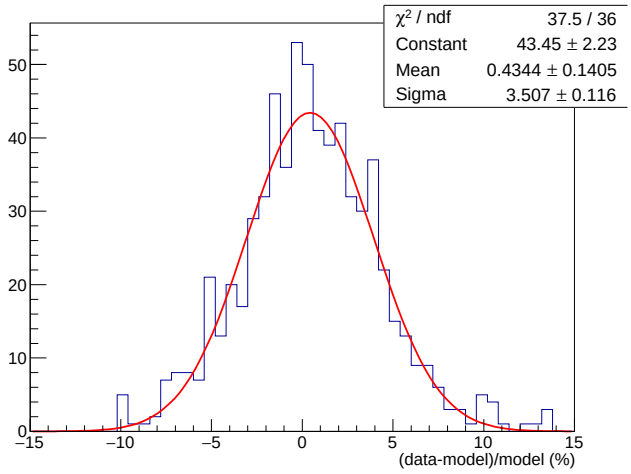
2006 年



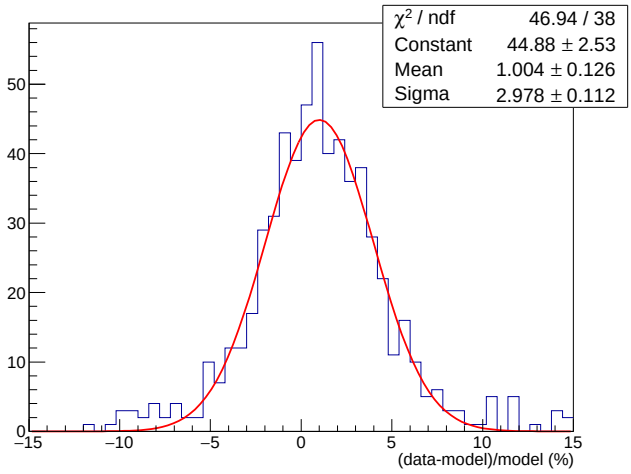
2007 年



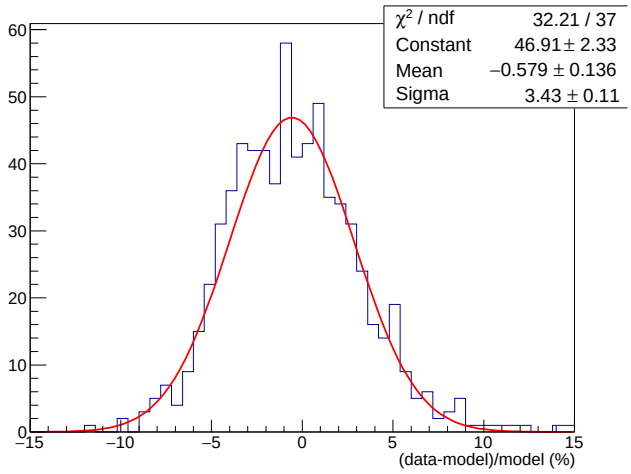
2008 年



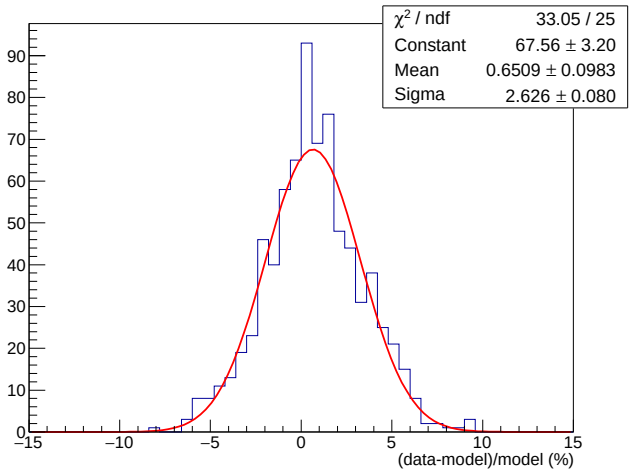
2009 年



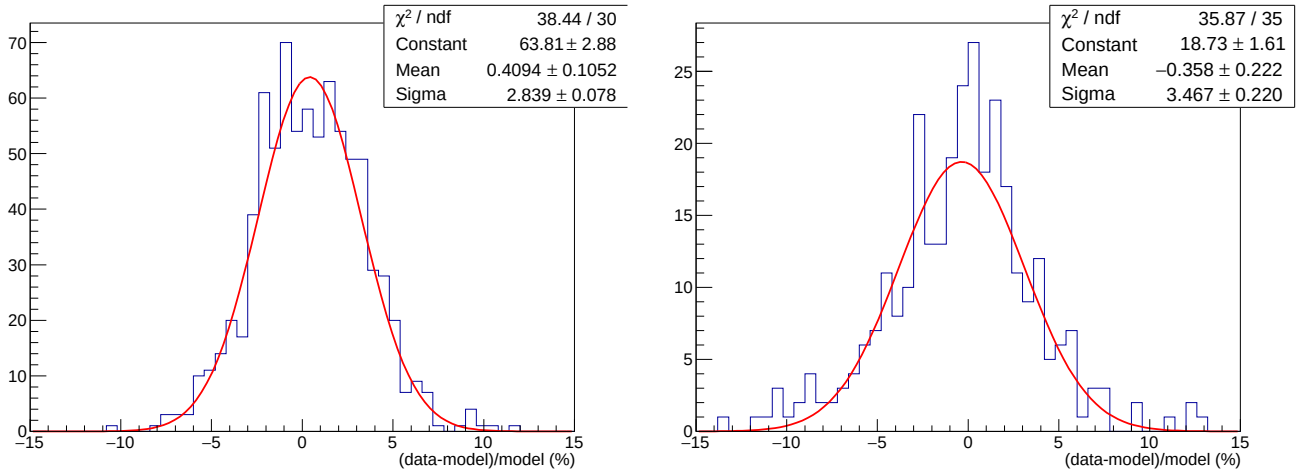
2010 年



2011 年



2012 年



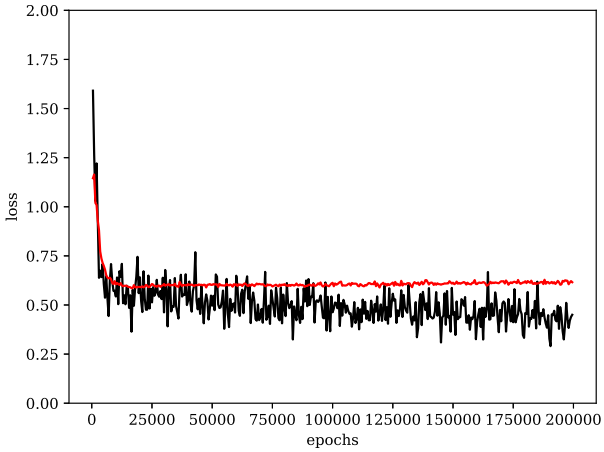
2013 年　　2014 年

図 4.17: 外れ値を除いた学習結果の例 (10ks 積分)

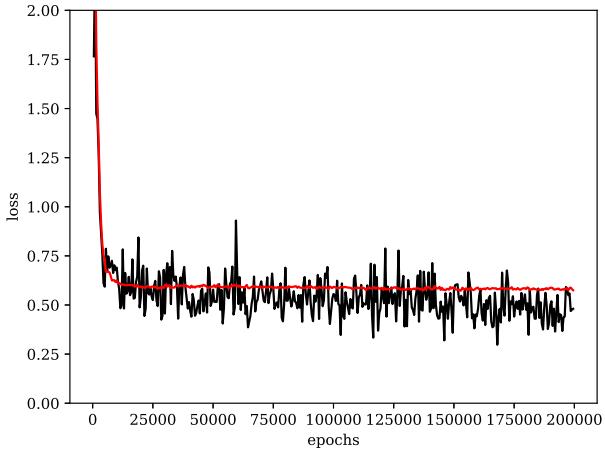
	平均	(平均) 外れ値ありとの差	標準偏差	(標準偏差) 外れ値ありとの差
2005 年	0.40	-0.16	1.86	-0.20
2006 年	0.84	1.31	1.76	-0.89
2007 年	-0.53	-1.01	2.78	0.81
2008 年	-0.99	-1.18	3.75	1.09
2009 年	0.43	-0.10	3.51	-0.04
2010 年	1.00	0.30	2.98	-0.11
2011 年	-0.58	-0.17	3.43	-0.01
2012 年	0.65	0.98	2.63	-0.54
2013 年	0.41	0.58	2.84	-0.65
2014 年	-0.36	0.05	3.47	0.09

表 4.16: 外れ値を除いた学習結果の例 (10ks 積分)

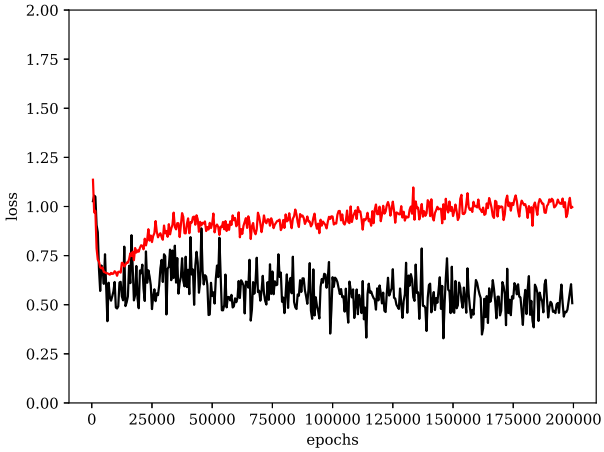
2007 年、2008 年、2014 年は悪化したことがわかる。図 4.18 に外れ値を除外したときの損失関数の推移を示す。



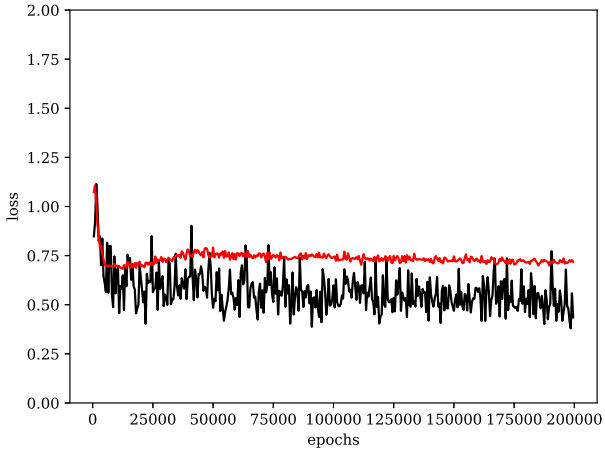
2005 年



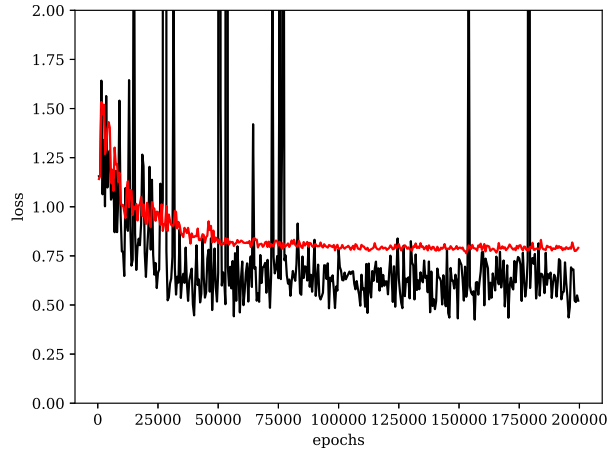
2006 年



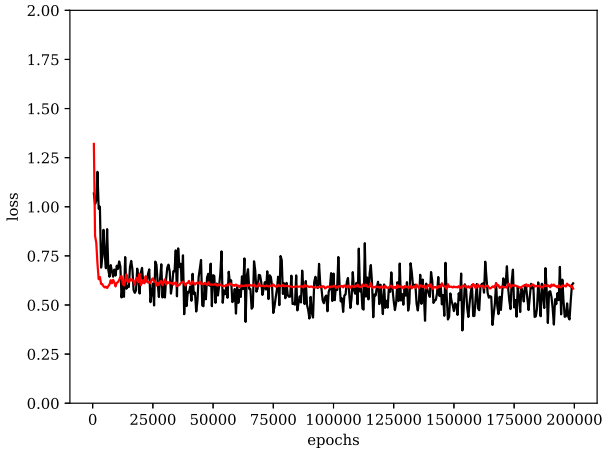
2007 年



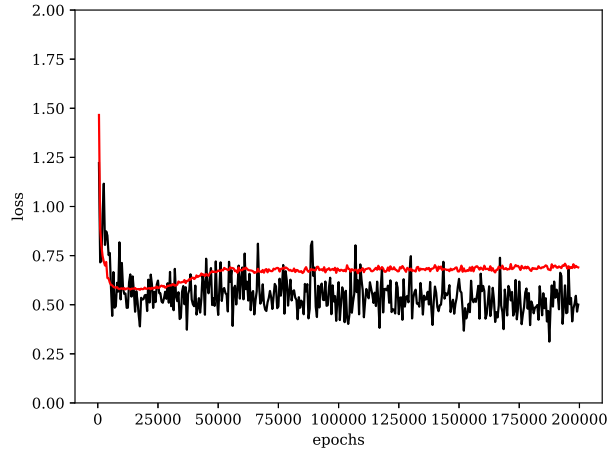
2008 年



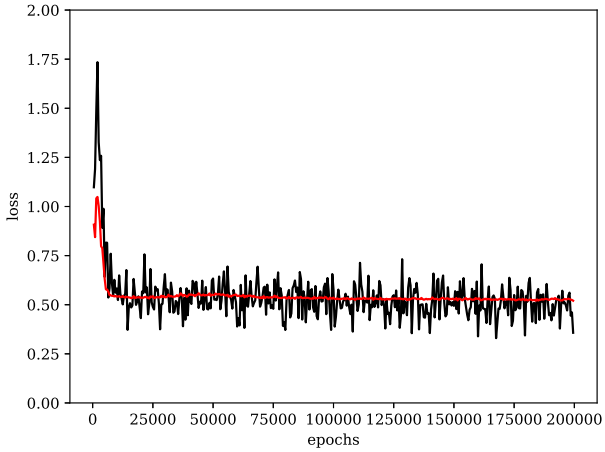
2009 年



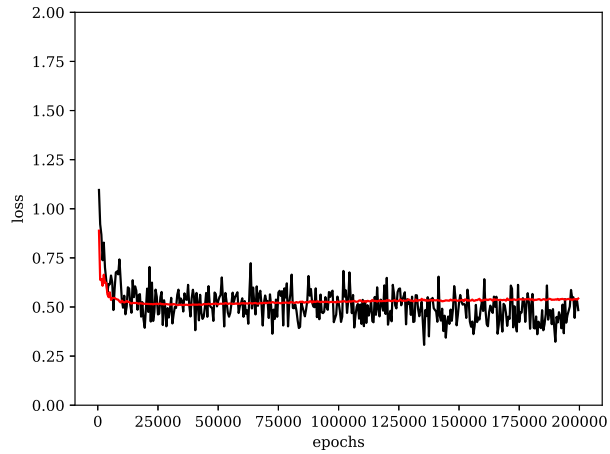
2010 年



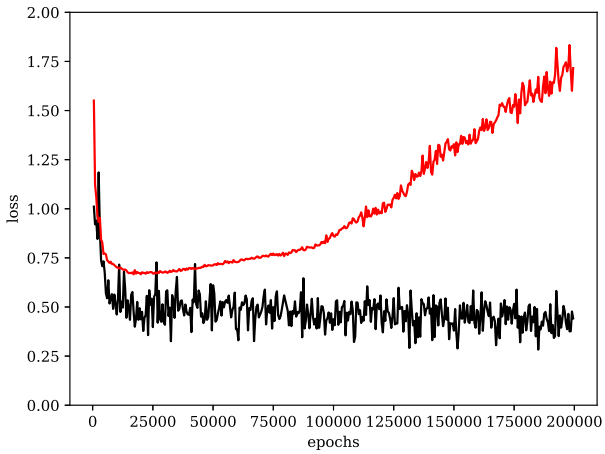
2011 年



2012 年



2013 年

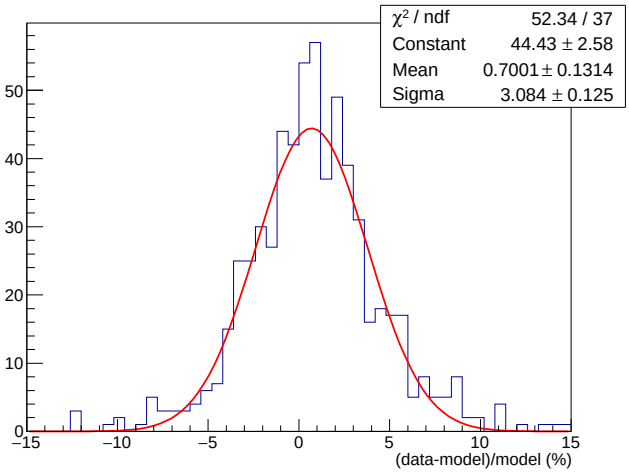


2014 年

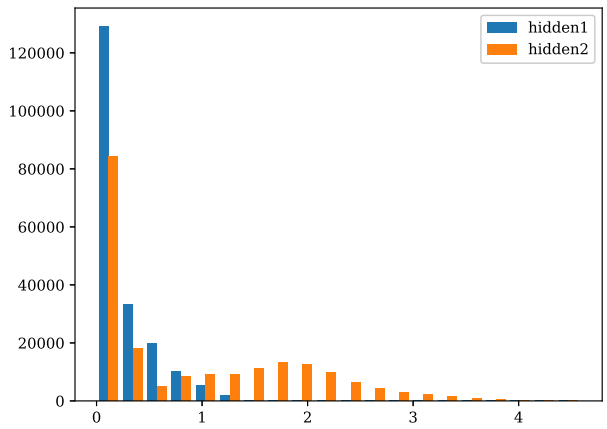
図 4.18: 外れ値を除外した各年ごとの損失関数の推移の比較

4.4.3 悪かった年の改善

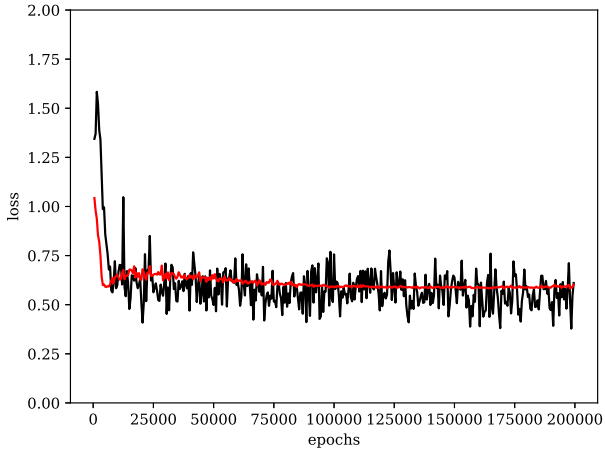
結果の悪かった年の 1 つである 2010 年の改善を図った。図 4.19 に示すのは、図 4.15 と図 4.16 においてより良かった、平均が約 1 で標準偏差が約 3 のグループに属する学習結果である。



部分残差ヒストグラム (10ks 積分)



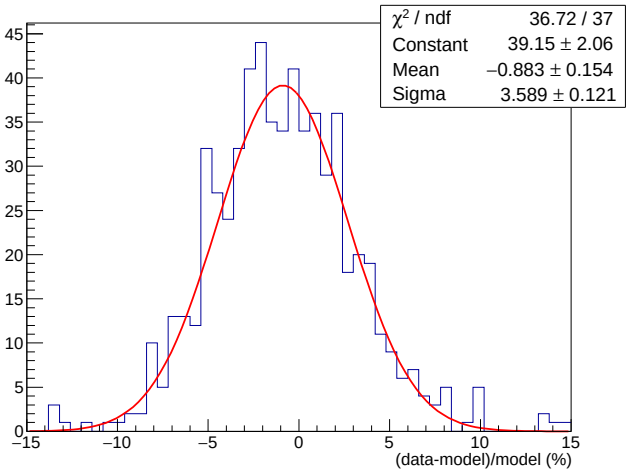
アクティベーションの分布



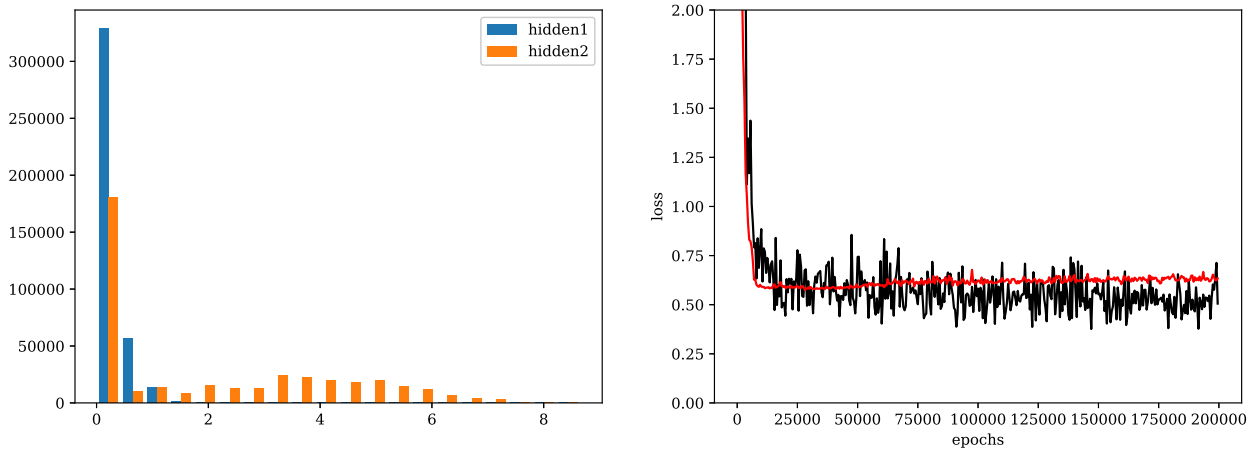
損失関数の推移

図 4.19: 2010 での学習結果の例

図 4.19 を見ると 2007 年と比べてアクティベーションが 0 のユニットが増えていることがわかる。表 4.8 の構造で学習した結果を図 4.20 に示す。



部分残差ヒストグラム (10ks 積分)

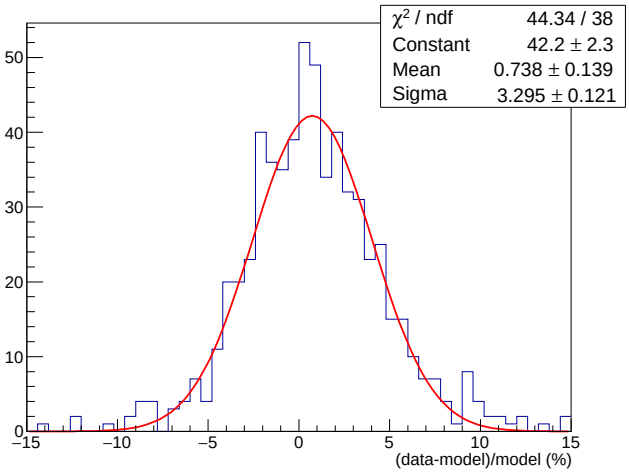


アクティベーションの分布　　損失関数の推移

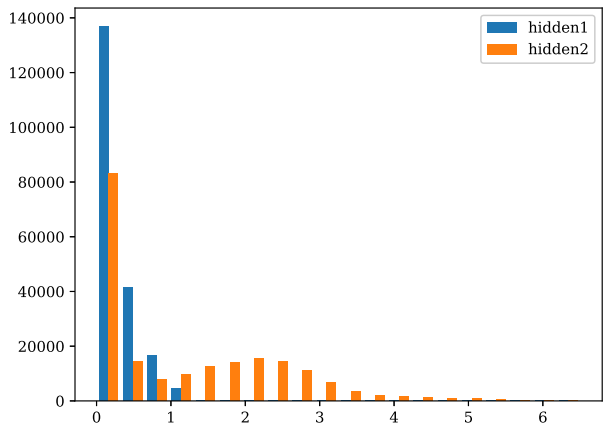
図 4.20: 隠れ層のユニット数を 400 にした、2010 での学習結果の例

図 4.20 を見ると図 4.19 と比べてアクティベーションの分布は広がったが 0 のユニットが増え、ヒストグラムの標準偏差は悪化した。

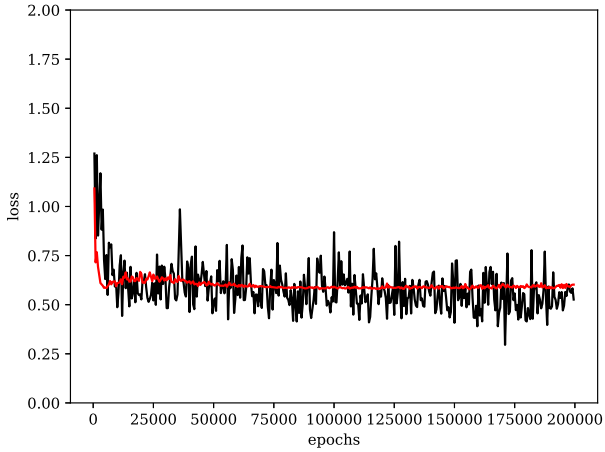
表 4.9 の構造で学習した結果を図 4.21 に示す。



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布



損失関数の推移

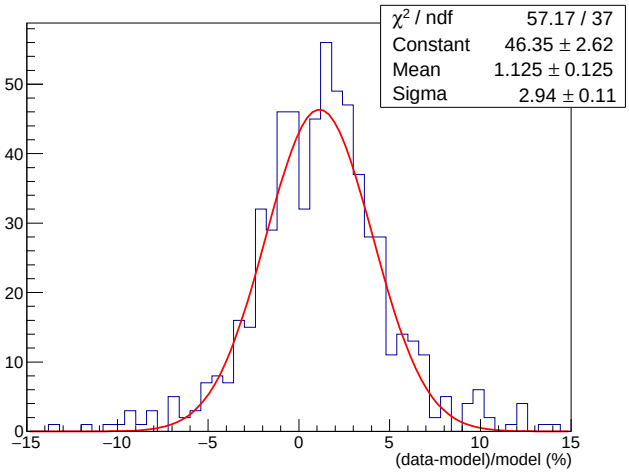
図 4.21: 重みの初期値を He にした、2010 での学習結果の例

図 4.21 を見ると図 4.19 と比べてアクティベーション分布の幅は少し広がったが、改善しなかった。
表 4.17 に隠れ層を 400 にして重みの初期値を He の初期値にしたニューラルネットワークを示す。

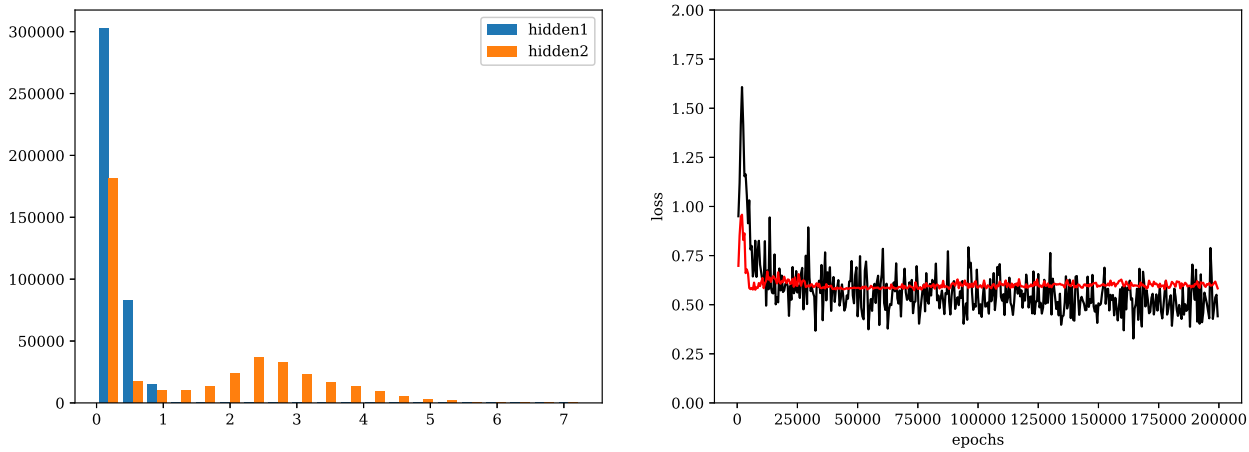
	ユニット数	活性化関数	重み初期値
入力層	25		
隠れ層 1	400	ReLU	He の初期値
隠れ層 2	400	ReLU	He の初期値
出力層	1	ソフトプラス	He の初期値

損失関数	$256 \times \frac{1}{N} \sum (y - t + y \times (\log t - \log y))$
最適化法	Adam
学習率	0.0001

表 4.17: 重みの初期値を He にして、隠れ層のユニット数を 400 にした、2010 での学習結果の例



部分残差ヒストグラム (10ks 積分)



アクティベーションの分布　　損失関数の推移

図 4.22: 重みの初期値を He にして、隠れ層のユニット数を 400 にした、2010 での学習結果の例

図 4.22 を見ると図 4.19 と同様で、アクティベーション分布の幅は少し広がったが、改善しなかった。

第5章　まとめと今後

全結合ニューラルネットワークを用いて「すざく」HXD-PIN のバックグラウンドモデルを作成し、その結果次のようなことがわかった。最適なネットワークの構成は、隠れ層は 2 層で各隠れ層のユニット数は 200 であり、活性化関数は隠れ層では ReLU で出力層ではソフトプラスを用いるのが良い。隠れ層を３つ以上にすると勾配消失が起きてしまい、また隠れ層が 300 を超えると過学習が起きることがわかった。そして、He の初期値を用いても結果は改善せず、Adam 以外のオプティマイザでは損失関数の最小化が十分に成されないことがわかった。

最適なネットワークから予測したバックグラウンドモデルの不定性は、2007 年では約 2.0%であり、今までの経験的なモデル bgd-a の 3.75%よりも優れており、bgd-d の 2.31%と同等である。これにより、経験的で複雑な式 (2.1) を、機械学習により置き換えられることを示した。

今後の課題についていくつか述べる。まず１つ目は、$PINUD_{buildup}$ を機械学習に置き換えることである。$PINUD_{buildup}$ の時定数は人間が設定したものであるため、機械学習により決定することで適切なパラメータにできる可能性がある。具体的には畳み込みニューラルネットワークで $PINUD_{buildup}$ に相当するものを学習させる。

２つ目は、他の衛星に転用可能なニューラルネットワークを構築することである。4.4 節を見るとバックグラウンドをあまり再現できていない年があることがわかる。まずは他の年でも良い再現ができるようにニューラルネットワークを構築することが必要であると思われる。

謝辞

本研究を行うに当たり指導していただいた深沢先生、北口先生に深謝致します。研究の方針、考察など様々なご指導ありがとうございました。研究室、事務の方々にも厚くお礼申し上げます。

参考文献

[1] Fukazawa et al.(2009) :Modeling and Reproducibility of Suzaku HXD PIN/GSO Background

[2] Hickox et al.(2006) : Absolute measurement of the unresolved cosmic X-ray background in the 0.5-8 keV band with Chandra

[3] Kokubun et al.(2007) : In-Orbit Performance of the Hard X-Ray Detector on Board Suzaku

[4] 笹田真人 2006 年度 卒業論文 (広島大学)

[5] 中本創 2005 年度 卒業論文 (広島大学)

[6] http://heasarc.gsfc.nasa.gov/docs/rosat/
ROSAT Web ページ

[7] http://www.astro.isas.ac.jp/xjapan/asca/
宇宙研 X 線天文グループ あすかの成果

[8] http://www.isas.jaxa.jp/j/special/2008/suzaku/19.shtml
JAXA X 線天文衛星「すざく」

[9] The Suzaku Technical Description (January, 2015)

[10] XspecManual version 12.9.1

[11] 巣籠悠輔 (2017)『詳解ディープラーニング　TensorFlow・Keras による時系列データ処理』

[12] 斎藤康毅 (2016)『ゼロから作る Deep Learning -Python で学ぶディープラーニングの理論と実践』

[13] Xavier Glorot and Yoshua Bengio (2010) : Understanding the difficulty of traing deep feedforward neural networks

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015): Delving Deep into Rectifiers: Surpassing human-level performance on imagenet classification

[15] N.Qian. (1999) : On the momentum term in gradient descent learning algorithms

[16] Duchi, John, Elad Hazan, and Yoram Singer. (2011) : Adaptive subgradient methods for online learning and stochastic optimization

[17] Diederik Kingma and Jimmy Ba.(2014) : Adam: A method for stochastic optimization