

修士論文

ニューラルネットワークを用いたリングダウン
重力波のデータ解析

東京工業大学理学院物理学系物理学コース

宗宮研究室

学籍番号 17M00682 久富 正博

平成31年 2月7日 提出

概要

時空の歪みが光速で伝播する現象を重力波と呼ぶ。Einsteinの一般相対論で予言された重力波だが、2015年9月にアメリカのレーザー干渉計型重力波検出器LIGOが初の直接検出を達成した。その際に観測されたのは、太陽質量の36倍のブラックホールと29倍のブラックホールの連星合体によるものだった。最終的に太陽質量の62倍のブラックホールになったため、残りの太陽質量の3倍分のエネルギーが重力波のエネルギーとなって検出された。このLIGOの直接検出によって重力波の存在は直接的に確かめられたわけだが、今後の重力波検出の意義がなくなったわけではない。今後の重力波検出はより様々な周波数帯における重力波検出を目標としている。より高周波数帯の重力波のソースとしては中性子星連星合体などがあり、低周波数帯の重力波のソースとしてはさらに重いブラックホールの連星合体や初期宇宙のインフレーション理論の裏付けにもなる背景重力放射などがある。それらの直接検出に成功すれば、今後の天文学の発展を大いにもたらす。これら重力波を検出するために、世界中ではレーザー干渉計を用いた重力波検出器が建設中および稼働中である。

現在稼働している重力波検出器では、主に中性子星練成合体、ブラックホール連星合体や中性子星とブラックホールからなる連星の合体をターゲットにしており、これら連星合体の最後には新たなブラックホールが生成されると考えられている。このような連星合体後やブラックホールに物体が落下した際に、ブラックホールに振動が励起され、重力波を放出しながら減衰していく準固有振動という現象が発生する。この準固有振動から発生する重力波はリングダウンと呼ばれ、減衰正弦波の形をしている。一方、重力波検出器で確認されている雑音の中には、リングダウンのような減衰正弦波の波形を持ったものが多く存在している。そのため、従来の解析手法である、リングダウンの基本モードのみを用いたマッチドフィルタリングでは、リングダウン重力波を解析することは困難である。しかし、基本モードに加えて高次モードも加えてマッチドフィルタリングを行うことにより、一定の条件下で従来の解析手法よりも優れた解析を行うことができると参考論文[1]で実証された。一方、マッチドフィルタリングは事前に予測される理論波形群（テンプレート群）と検出器から得られたデータとの畳み込みを計算する解析手法であるため、そのテンプレート空間が増大するに伴い、計算コストも急激に増大してしまうという問題がある。

本研究では、より精度良くかつより低い計算コストでの、リングダウン重力波の解析を行うために、畳み込みニューラルネットワークを適用した解析を行った。畳み込みニューラルネットワークはニューラルネットワークの一種であり、画像解析などの分野において非常に優れた結果を残している。畳み込みニュー

ラルネットワークは事前にデータセットを用いて学習を行うことができ、解析では単純な四則演算によって表現されるため計算コストを低減することができる。ここでは、複数の条件下におけるリングダウン重力波の解析を、マッチドフィルタリング及び畳み込みニューラルネットワークを用いて行い、その比較を考察した。

Abstract

Gravitational waves(GW) are disturbances in the curvature of spacetime and propagate as waves at the speed of light. They were predicted by Albert Einstein on the basis of his general theory of relativity, and the first direct detection of the GW was achieved in 2015 by LIGO, the laser-interferometer GW detector in the U.S.. This event is called GW150914, and this GW was generated by the coalescence of binary black hole(BH). Each mass of BHs was 29 and 36 solar mass and the mass of BH generated after the coalescence was 62 solar mass, so the energy equivalent to 3 solar mass was radiated as GW. This event ensures the existence of GW, but this fact doesn't mean that detection of GW in future is meaningless. Now, their goal is widening the frequency band of GW detection. Expected source of higher frequency GW is the coalescence of binary neutron star(NS), and the first detection of binary NS was achieved in 2017. On the other hand, expected source of lower frequency GW is the coalescence of binary BH that has heavier mass and inflation of the early universe. So, detection of GW generated by their sources can impact future development of astronomy greatly. In order to detect their GWs, various laser interferometer GW detector are under construction or operation.

Main target of GW detectors that are under construction or operation is the coalescence of BH or NS, and new BH is generated after the coalescence. After the coalescence or when test mass is fallen down to BH, vibration is excited to BH and quasi-normal mode vibration that is attenuated with radiating GW is generated. GW generated by this quasi-normal mode vibration is called ringdown, and the shape of damped sine wave. Ringdown signal is described as the sum of fundamental mode and higher mode. On the other hand, there are some noises from detectors which shape is like ringdown signal(damped sine wave), so it's difficult for conventional method 'Matched Filtering' to analyze ringdown signal using only fundamental mode. But, it was demonstrated that we can analyze ringdown signal with better accuracy using higher mode under some restriction according to reference[1]. By the way, matched filtering is the method that calculate the convolution between data from detectors and template calculated on the basis of numerical relativity. So, the larger template is, the much higher the computational cost will be. This is the problem for low-latency analysis.

In this thesis, I adopt 'convolutional neural network(CNN)' to ringdown analysis in order to execute the analysis with higher accuracy and lower computational cost. CNN is a type of feed-forward neural network, and has been most effective for image and video processing. CNN learns optimal model from datasets in advance, so it can lower the computational cost. In this thesis, I analyzed ringdown signal using both matched filtering and CNN, and compared them.

目次

概要	i
Abstract	iii
第1章 重力波	1
1.1 一般相対論の表記	1
1.2 Einstein方程式	2
1.3 線形化されたEinstein方程式	2
1.4 重力波の解	4
1.5 レーザー干渉計型重力波検出器の重力波応答	6
1.6 重力波源	8
第2章 カー・ブラックホールからの重力波	11
2.1 曲率を持った時空における摂動	11
2.2 カー・ブラックホールからの重力波	12
第3章 ベイズ統計学とマッチドフィルタリング	15
3.1 ベイズ統計学	15
3.2 マッチドフィルタリング	18
第4章 畳み込みニューラルネットワーク	25
4.1 パーセプトロン	25
4.2 ニューラルネットワークとその学習	29
4.3 畳み込みニューラルネットワーク	44
第5章 リングダウン重力波の解析	52
5.1 リングダウン重力波の信号生成	52
5.2 干渉計のノイズスペクトル	57

目次	vi
5.3 マッチドフィルタリング	60
5.4 トレインデータ・テストデータの作成	65
5.5 畳み込みニューラルネットワークの構造とその実装	66
5.6 ネットワークの学習	68
第6章 解析結果と考察	73
6.1 $29M_{\text{sun}}$ と $36M_{\text{sun}}$ のブラックホール連星からのリングダウン重力波における解析	73
6.2 $1.4M_{\text{sun}}$ と $5M_{\text{sun}}$ の中性子星・ブラックホール連星からのリングダウン重力波における解析	76
6.3 結果のまとめと考察	77
付録A	80
参考文献	82
謝辞	83

目次

1.1	紙面に垂直な方向から重力波が入射した時の自由質点間距離の変化	6
1.2	Michelson干渉計	7
4.1	単純パーセプトロン（2入力）	25
4.2	単純パーセプトロンによるAND回路を可視化したもの	27
4.3	XOR回路を可視化したもの	27
4.4	XOR回路の実現	28
4.5	XOR回路のパーセプトロンによる表記	28
4.6	パーセプトロンによるXOR回路の実現を可視化したもの	29
4.7	ニューラルネットワークの学習プロセス	30
4.8	ニューラルネットワークの例	31
4.9	各ニューロンにおける処理を明示化したもの	32
4.10	様々な活性化関数	34
4.11	恒等関数	34
4.12	ソフトマックス関数	35
4.13	出力層	36
4.14	シグモイドレイヤの計算グラフ（順伝播）	38
4.15	シグモイドレイヤの計算グラフ（順伝播・逆伝播）	40
4.16	SGDのイメージ図	41
4.17	畳み込みニューラルネットワークの構造例	45
4.18	畳み込み演算の例	46
4.19	畳み込み演算のプロセス	47
4.20	畳み込み層における処理（バイアス付加）	47
4.21	パディング処理の例（パディング幅1）	48
4.22	ストライドの例（ストライド2）	49
4.23	ミニバッチ対応の畳み込み演算処理	50

4.24	Max Poolingによる処理例	51
5.1	質量比と各モードの振幅比の関係。S. Gossan[6]より引用。	54
5.2	疑似的干渉計ノイズスペクトル：青実線が元のデータ。橙点線がパーミュテーションテスト後の疑似的干渉計データ	59
5.3	疑似的干渉計ノイズデータ（タイムドメイン）：青実線が元のデータ。橙実線がパーミュテーションテスト後の疑似的干渉計データ	59
5.4	トレインデータ・テストデータ用の連星質量	65
5.5	本研究におけるネットワーク構造	66
5.6	本研究におけるネットワーク構造（詳細）	67
5.7	学習経過の様子（損失関数）	72
5.8	学習経過の様子（精度）	72
6.1	重力波検出の二値分類	73
6.2	$29M_{\text{sun}}$ と $36M_{\text{sun}}$ のブラックホール連星合体によるリングダウン重力波における適合率 . .	74
6.3	$29M_{\text{sun}}$ と $36M_{\text{sun}}$ のブラックホール連星合体によるリングダウン重力波における再現率 . .	75
6.4	$1.4M_{\text{sun}}$ と $5M_{\text{sun}}$ の中性子星・ブラックホール連星合体によるリングダウン重力波における適合率	76
6.5	$1.4M_{\text{sun}}$ と $5M_{\text{sun}}$ の中性子星・ブラックホール連星合体によるリングダウン重力波における再現率	77
6.6	論点整理	78
A.0	各モードのInclinationによる寄与	81

ソースコード目次

5.1	Ringdown_signal.py	55
5.2	noise.py	57
5.3	template.py	60
5.4	matchedfilter.py	61
5.5	network.py	66
5.6	train.py	69

第1章

重力波

重力波は重力場を記述するEinstein方程式を線形近似することにより導かれる。本章では、Einstein方程式から、真空中における重力波の満たす解を求める。

1.1 一般相対論の表記

一般相対論に登場する記法をここにまとめる。

添え字の表記

添え字の表記を以下のように定める：

$$\text{添え字がアルファベットの場合} \quad \cdots \quad 1, 2, 3 \quad (1.1.1)$$

$$\text{添え字がギリシャ文字の場合} \quad \cdots \quad 0, 1, 2, 3 \quad (1.1.2)$$

なお、0が時間、1, 2, 3は空間に関する添え字である。

Einsteinの縮約記法

ある一つの添え字が上付きでそれと同じ添え字が下付きで出てくる表式においては、その添え字の取りうる値すべてについての総和をとることにする。この記法をEinsteinの縮約記法という。たとえば

$$A_a B^a \quad (1.1.3)$$

と書かれていたとき

$$\sum_{a=1}^3 A_a B^a \quad (1.1.4)$$

を意味する。同様に

$$C^\mu D_{\mu\nu} \quad (1.1.5)$$

と書かれていたとき

$$\sum_{\mu=0}^3 C^\mu D_{\mu\nu} \quad (1.1.6)$$

を意味する。

1.2 Einstein方程式

一般相対論において、4次元時空内の異なる2点 x^μ 、 $x^\mu + dx^\mu$ の間の線素 ds は

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu \quad (1.2.1)$$

で表される。ここで $g_{\mu\nu}$ はメトリックテンソルと呼ばれ、重力場が存在する曲がった時空を記述する計量テンソルである。一方、重力場がない平坦な時空はMinkowski時空と呼ばれ、そのメトリックテンソル $\eta_{\mu\nu}$ は

$$\eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.2.2)$$

と書かれる。そして、重力場が存在する時空でメトリックテンソル $g_{\mu\nu}$ が満たす方程式がEinstein方程式(1.2.3)である：

$$G_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu} \quad (1.2.3)$$

$$G_{\mu\nu} \equiv R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R \quad (1.2.4)$$

$$R_{\mu\nu} \equiv R^\alpha_{\mu\alpha\nu} \quad (1.2.5)$$

$$R \equiv R^\alpha_\alpha \quad (1.2.6)$$

ただし、 $G_{\mu\nu}$ はEinsteinテンソル、 $T_{\mu\nu}$ はエネルギー・運動量テンソル、 $R_{\mu\nu}$ はRicciテンソル、 R はRicciスカラー、 G は万有引力定数、 c は光速である。ここで、Ricciテンソル $R_{\mu\nu}$ 、Ricciスカラー R はRiemannテンソル $R^\mu_{\nu\alpha\beta}$ 、Christoffel記号 $\Gamma^\mu_{\nu\lambda}$ を用いて

$$\Gamma^\mu_{\nu\lambda} = \frac{1}{2} g^{\mu\alpha} (g_{\alpha\nu,\lambda} + g_{\alpha\lambda,\nu} - g_{\nu\lambda,\alpha}) \quad (1.2.7)$$

$$R^\mu_{\nu\alpha\beta} = \Gamma^\mu_{\nu\beta,\alpha} - \Gamma^\mu_{\nu\alpha,\beta} + \Gamma^\mu_{\gamma\alpha} \Gamma^\gamma_{\nu\beta} - \Gamma^\mu_{\gamma\beta} \Gamma^\gamma_{\nu\alpha} \quad (1.2.8)$$

である。なお、添え字中のカンマは偏微分を表す：

$$\frac{\partial A^\mu}{\partial x^\nu} = \partial_\nu A^\mu = A^\mu_{,\nu} \quad (1.2.9)$$

1.3 線形化されたEinstein方程式

重力場が弱い場合を考える。このとき、重力場をMinkowski時空からの摂動として考えることができる。そのMinkowski時空からの摂動を $h_{\mu\nu}$ とするとメトリックテンソルは

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \quad (1.3.1)$$

と書ける。この線形近似において、添え字の上げ下げは η を用いる：

$$h^\mu{}_\nu = \eta^{\mu\sigma} h_{\sigma\nu} \quad (1.3.2)$$

また、

$$g^{\mu\nu} = \eta^{\mu\nu} - h^{\mu\nu} \quad (1.3.3)$$

となる。これより、Christoffel記号、Riemannテンソル、Ricciテンソル、Ricciスカラーを順々に求めていくと

$$\Gamma^\mu{}_{\nu\lambda} = \frac{\eta^{\mu\sigma}}{2} (h_{\sigma\lambda,\nu} + h_{\sigma\nu,\lambda} - h_{\nu\lambda,\sigma}) \quad (1.3.4)$$

$$R^\mu{}_{\nu\delta\lambda} = \frac{\eta^{\mu\sigma}}{2} (h_{\sigma\lambda,\nu\delta} + h_{\nu\delta,\sigma\lambda} - h_{\nu\lambda,\sigma\delta} - h_{\sigma\delta,\nu\lambda}) \quad (1.3.5)$$

$$R_{\nu\lambda} = \frac{\eta^{\delta\sigma}}{2} (h_{\sigma\lambda,\nu\delta} + h_{\nu\delta,\sigma\lambda} - h_{\nu\lambda,\sigma\delta} - h_{\sigma\delta,\nu\lambda}) \quad (1.3.6)$$

$$R = \frac{\eta^{\nu\lambda}\eta^{\delta\sigma}}{2} (h_{\sigma\lambda,\nu\delta} + h_{\nu\delta,\sigma\lambda} - h_{\nu\lambda,\sigma\delta} - h_{\sigma\delta,\nu\lambda}) \quad (1.3.7)$$

となる。これらより、Einstein方程式(1.2.3)の左辺は

$$G_{\nu\lambda} = \frac{1}{2} [h^\delta{}_{\lambda,\nu\delta} + h^\delta{}_{\nu,\lambda\delta} - \square h_{\nu\lambda} - h_{,\nu\lambda} - \eta_{\nu\lambda} (h^{\delta\sigma}{}_{,\delta\sigma} - \square h)] \quad (1.3.8)$$

と書ける。ここで

$$h = \eta^{\mu\nu} h_{\mu\nu} \quad (1.3.9)$$

$$\square = \eta^{\mu\nu} \partial_\mu \partial_\nu \quad (1.3.10)$$

とした。さらに $h_{\mu\nu}$ の逆トレーステンソル $\tilde{h}_{\mu\nu}$ を

$$\tilde{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2} \eta_{\mu\nu} h \quad (1.3.11)$$

と定義すれば、

$$\tilde{h}_{\mu\nu} = \eta^{\mu\nu} \tilde{h}_{\mu\nu} = -h \quad (1.3.12)$$

であり、

$$G_{\nu\lambda} = \frac{1}{2} (\tilde{h}^\delta{}_{\lambda,\nu\delta} + \tilde{h}^\delta{}_{\nu,\lambda\delta} - \square \tilde{h}_{\nu\lambda} - \eta_{\nu\lambda} \tilde{h}^{\delta\sigma}{}_{,\delta\sigma}) \quad (1.3.13)$$

と表される。ここで、式(1.3.13)を簡略化するために以下のようなゲージ変換を考える：

$$x'^\mu = x^\mu + \xi^\mu(x) \quad (1.3.14)$$

このとき

$$\begin{aligned} g'_{\mu\nu} &= \frac{\partial x^\alpha}{\partial x'^\mu} \frac{\partial x^\beta}{\partial x'^\nu} g_{\alpha\beta} \\ &= (\delta^\alpha{}_\mu - \xi^\alpha{}_{,\mu}) (\delta^\beta{}_\nu - \xi^\beta{}_{,\nu}) (\eta_{\alpha\beta} + h_{\alpha\beta}) \\ &= \eta_{\mu\nu} + h_{\mu\nu} - \xi_{\mu,\nu} - \xi_{\nu,\mu} \end{aligned} \quad (1.3.15)$$

なので

$$h'_{\mu\nu} = h_{\mu\nu} - \xi_{\mu,\nu} - h_{\nu,\mu} \quad (1.3.16)$$

$$h' = h - 2\xi^\sigma{}_{,\sigma} \quad (1.3.17)$$

で表される。これを用いると

$$\tilde{h}'_{\mu\nu} = h'_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h = \tilde{h}_{\mu\nu} - \xi_{\mu,\nu} - \xi_{\nu,\mu} + \eta_{\mu\nu}\xi^\sigma{}_{,\sigma} \quad (1.3.18)$$

となる。ここで

$$\tilde{h}'^\mu{}_{\nu,\mu} = \tilde{h}^\mu{}_{\nu,\mu} - \square\xi_\nu \quad (1.3.19)$$

となるから、適当な ξ^μ を選ぶことによって常に

$$\tilde{h}'^\mu{}_{\nu,\mu} = 0 \quad (1.3.20)$$

とできる。この条件(調和条件)を課すと、Einsteinテンソルの成分は以下のように書き表される：

$$G_{\mu\nu} = -\frac{1}{2}\square\tilde{h}_{\mu\nu} \quad (1.3.21)$$

したがって、これを式(1.2.3)に代入することで、線形化されたEinstein方程式

$$\square\tilde{h}_{\mu\nu} = -\frac{16\pi G}{c^4}T_{\mu\nu} \quad (1.3.22)$$

を得る。

1.4 重力波の解

線形化されたEinstein方程式は式(1.3.22)で与えられた。真空中においては $T_{\mu\nu} = 0$ であるから、式(1.3.22)は次のように書き換わる：

$$\square\tilde{h}_{\mu\nu} = 0 \quad (1.4.1)$$

式(1.4.1)の解として、平面波

$$\tilde{h}_{\mu\nu} = A_{\mu\nu} \exp(ik_\lambda x^\lambda) \quad (1.4.2)$$

を考える。式(1.4.2)を式(1.4.1)に代入すると

$$\eta^{\lambda\sigma}k_\lambda k_\sigma A_{\mu\nu} = 0 \quad (1.4.3)$$

を得る。同様に式(1.3.20)に代入すると

$$\eta^{\lambda\nu}k_\nu A_{\mu\lambda} = 0 \quad (1.4.4)$$

を得る。式(1.4.3)、式(1.4.4)より、式(1.4.2)で表される平面波の解は以下の条件を満たす：

$$A_{\mu\nu}k^\nu = 0 \quad (1.4.5)$$

$$k_\mu k^\mu = 0 \quad (1.4.6)$$

式(1.4.5)は、平面波の振幅と波の進行方向が直交していることから、この平面波が横波であることを表している。一方、式(1.4.6)は、この平面波が光速で伝播していくことを表している。

さらに、式(1.3.14)のゲージ変換を行う。ここで、式(1.3.19)から調和条件を満たすためには、 $\square \xi_\mu = 0$ を満たす関数でなければならないことが分かる。そこで、定数 B_μ を用いて

$$\xi_\mu = B_\mu \exp(ik_\lambda x^\lambda) \quad (1.4.7)$$

とする。式(1.3.16)、式(1.3.17)に従って計算すれば

$$h'_{\mu\nu} = h_{\mu\nu} - i(B_\mu k_\nu + B_\nu k_\mu) \exp(ik_\lambda x^\lambda) \quad (1.4.8)$$

$$\begin{aligned} h' &= h - i\eta^{\mu\nu}(B_\mu k_\nu + B_\nu k_\mu) \exp(ik_\lambda x^\lambda) \\ &= -[\eta^{\mu\nu} A_{\mu\nu} + i\eta^{\mu\nu}(B_\mu k_\nu + B_\nu k_\mu)] \exp(ik_\lambda x^\lambda) \end{aligned} \quad (1.4.9)$$

となる。上式を見ると、適当な B_μ を選べば常に $h' = 0$ とできる。一般に、 $h = 0$ となるとき、トレースレス条件を満たすという。この場合においては $h_{\mu\nu} = \tilde{h}_{\mu\nu}$ である。

ここで、簡単な場合、 z 方向に重力波が伝播する場合を考える。このとき、 $-k_0 = k_3 = k, k_1 = k_2 = 0$ であるから、調和条件、トレースレス条件はそれぞれ

$$k(A_{\mu 0} + A_{\mu 3}) = 0 \quad (1.4.10)$$

$$-A_{00} + A_{11} + A_{22} + A_{33} = 0 \quad (1.4.11)$$

である。 $h'_{\mu\nu} = A'_{\mu\nu} \exp(ik_\lambda x^\lambda)$ として、式(1.4.8)に代入すれば

$$A'_{00} = A_{00} + 2iB_0 k \quad (1.4.12)$$

$$A'_{01} = A_{01} + iB_1 k \quad (1.4.13)$$

$$A'_{02} = A_{02} + iB_2 k \quad (1.4.14)$$

$$A'_{03} = A_{03} - i(B_0 k - B_3 k) = A_{03} - 2iB_0 k \quad (1.4.15)$$

となるから、 $A'_{0\mu} = 0$ となるように B_μ を決める。以上と調和条件、トレースレス条件から $A_{\mu\nu}$ の成分のうち0でない成分は $A_{11} = -A_{22}$ と $A_{12} = A_{21}$ の2つの量のみである。 $A_{\mu\nu}$ を行列で表すと

$$A_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (1.4.16)$$

となる。ここで、 h_+ と $h_\times$ は重力波の持つ2つの自由度を表しており、前者はプラスモード、後者はクロスモードと呼ばれる。

このように適当なゲージを選ぶことで、横波でトレースレス条件を満たすようにできる。このような

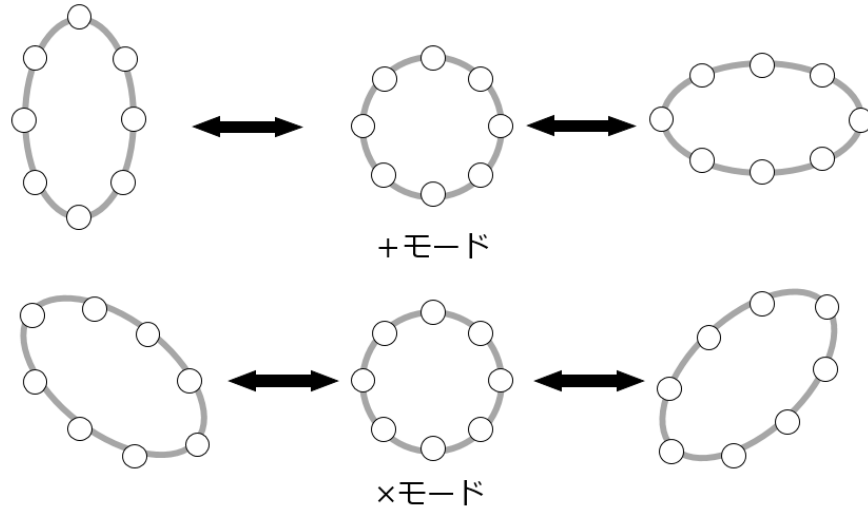


図1.1 紙面に垂直な方向から重力波が入射した時の自由質点間距離の変化

ゲージのことをトランスバース・トレースレスゲージという。

以上で、真空中における重力波の満たす方程式の解が求まった：

$$\tilde{h}_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \exp[i(\omega t - kz)] \quad (1.4.17)$$

このような重力波が到来すると、自由質点間の距離をそれぞれのモードが四重局に変化させる。その様相を可視化したものが図1.1である。

1.5 レーザー干渉計型重力波検出器の重力波応答

前節までで、一般相対論における重力波の記述について説明した。

現在までに、重力波を直接検出するために大きく2種類の重力波検出器が建設されてきた。1つは共振型重力波検出器であり、測定器の共振を用いて重力波を検出するものである。もう一方は、Michelson干渉計を基にしたレーザー干渉計型重力波検出器である。現在世界中で主流となっている重力波検出器は後者であり、本節では後者における重力波検出の原理について説明する。

Michelson干渉計は図1.2の通りである。レーザー光源から出たレーザー光は、ビームスプリッタによって二方向に分けられる。その後、それぞれのエンドミラーによって反射され、再びビームスプリッタに入射し、干渉する。そして、その干渉光をフォトディテクタによって検出する。

重力波の検出においては、あらかじめ二方向に分けられた光の位相差を制御することによって、干渉光が最も弱まるような状態に維持する。この状態で重力波が到来すると、自由質点間の距離を四重極に変化させる、つまりそれぞれのエンドミラーとビームスプリッタ間（以下、“腕”と記述）の距離差が変化する。その距離さの変化により、干渉光の干渉度合いが変化し、フォトディテクタ側に干渉光が漏れ出す。この

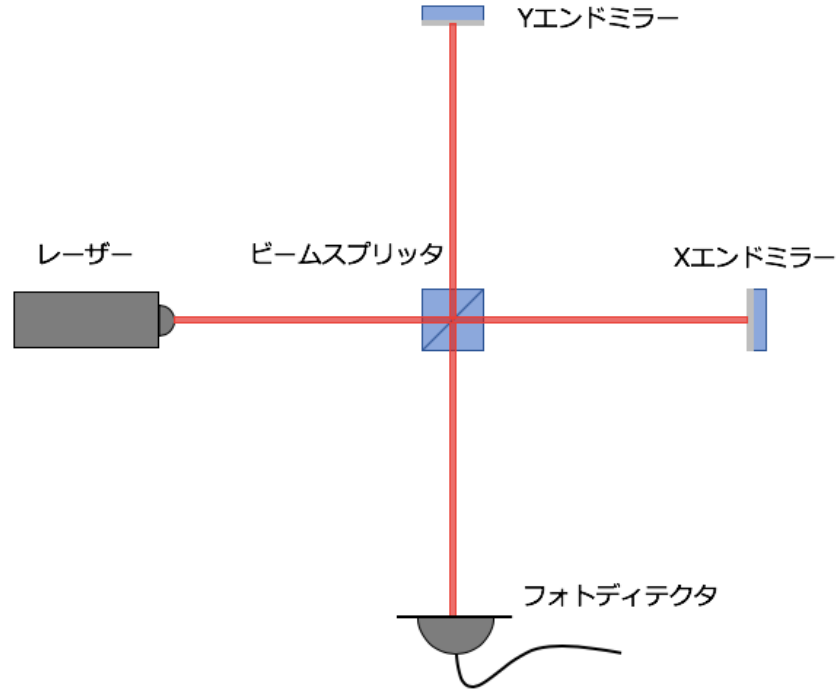


図1.2 Michelson干渉計

漏れ出した干渉光をフォトディテクタによって検出することにより、重力波を検出することが可能になる。以上がレーザー干渉計型重力波検出器の検出原理であり、以下でこの検出原理を式によって表現する。

今、ビームスプリッタの中心を座標原点、そしてそこから x 軸正方向と y 軸正方向に伸びる腕からなるMichelson干渉計に対して、 z 軸正方向に重力波が入射した場合を考える。ここでは簡略化のためプラスモードのみの寄与を考えることにする。つまり、 $h_+ = \tilde{h}(t)$, $h_\times = 0$ である。すると、 x 軸方向を運動する光子の世界線に沿った線素は、光の伝播はヌルベクトルであることから

$$ds^2 = -c^2 dt^2 + \{1 + \tilde{h}(t)\} dx^2 = 0 \quad (1.5.1)$$

である。さらに、 $\tilde{h}(t) \ll 1$ であるから、 $\frac{dx}{dt} > 0$ のとき、

$$\begin{aligned} (1.5.1) \Leftrightarrow dx &= \frac{c}{\sqrt{1 + \tilde{h}(t)}} \\ &\simeq \left\{1 - \frac{1}{2}\tilde{h}(t)\right\} c dt \end{aligned} \quad (1.5.2)$$

となる。ここで、 x 軸、 y 軸方向の腕の長さをそれぞれ l_x, l_y とする。さらに、光子が x 軸の腕を往復するのに要する時間を Δt_x とし、式(1.5.2)を両辺積分すると

$$\frac{2l_x}{c} = \int_{t-\Delta t_x}^t \left\{1 - \frac{1}{2}\tilde{h}(t')\right\} dt' \quad (1.5.3)$$

となるから、式変形して

$$\Delta t_x = \frac{2l_x}{c} + \frac{1}{2} \int_{t-\Delta t_x}^t \tilde{h}(t') dt' \quad (1.5.4)$$

となる。ここで $\tilde{h}(t) \ll 1$ であるから、積分の下限を $t - \frac{2l_x}{c}$ として近似すると

$$\Delta t_x \simeq \frac{2l_x}{c} + \frac{1}{2} \int_{t - \frac{2l_x}{c}}^t \frac{1}{2} \tilde{h}(t') dt' \quad (1.5.5)$$

となる。よって、光が x 軸上を往復するときの位相変化は

$$\begin{aligned} \phi_x &= \Omega \Delta t_x \\ &= \frac{2l_x \Omega}{c} + \frac{\Omega}{2} \int_{t - \frac{2l_x}{c}}^t \tilde{h}(t') dt' \end{aligned} \quad (1.5.6)$$

と求められる。同様にして、光が y 軸上を往復するときの位相変化は

$$\begin{aligned} \phi_y &= \Omega \Delta t_y \\ &= \frac{2l_y \Omega}{c} - \frac{\Omega}{2} \int_{t - \frac{2l_y}{c}}^t \tilde{h}(t') dt' \end{aligned} \quad (1.5.7)$$

と求められる。よって、基線長 $l \simeq l_x \simeq l_y$ とし、 $l_- = l_x - l_y$ とすると

$$\phi_- \equiv \phi_x - \phi_y = \frac{2l_- \Omega}{c} + \delta\phi_{GW} \quad (1.5.8)$$

$$\delta\phi_{GW} = \Omega \int_{t - \frac{2l_y}{c}}^t \frac{1}{2} \tilde{h}(t') dt' \quad (1.5.9)$$

と計算できる。式(1.5.8)において、第1項はビームスプリッタから2つの鏡までの距離差で生じる位相変化、第2項は重力波の影響による位相変化を表している。

レーザー干渉計型重力波検出器は、このような位相差を検出することにより重力波を検出している。

1.6 重力波源

前節で、重力波に対する重力波検出器の応答を見てきたが、現実では重力波によって引き起こされる位相差は非常に小さい。そのため、重力波を検出するためには、そのソースが非常に大きい天体現象に限られる。以下では、検出可能な重力波のソースとなる具体的な天体現象について紹介する。

コンパクト連星合体

コンパクト星とは、中性子星、ブラックホール、白色矮星などの総称であり、通常の恒星や惑星と比較して非常に高密度な天体である。これらの星が連星を成し、互いに回転しながら軌道運動をし、最終的に衝突してブラックホール等の天体になる過程をコンパクト連星合体と呼ぶ。実際に、現在建設・稼働している重力波検出器の検出ターゲットは主にコンパクト連星合体であり、特に中性子星やブラックホールからなるものである。

このコンパクト連星合体においては、大きく分けて3つのフェイズが存在し、それぞれのフェイズに対して重力波の波形や解析について研究が進められている。以下で、それぞれについて説明する。

Inspiral Phase

Inspiral Phaseは、連星が互いに連星重心の周りを公転しながら近づいていくフェイズである。このフェイズは現在最も有力な重力波源であり、GW150914、GW151226、GW170817等の実際に検出された重力波イベントはどれもInspiral Phaseからの重力波によって検出されている。

このフェイズが最も有力な重力波源となった所以は、放出される重力波の波形がポスト・ニュートニアン展開によって予測されていることである。波形が予測されていることによって、後述するマッチドフィルタリングという解析手法を用いることが可能になり、より効率のいい解析が可能になる。また、このフェイズの波形は、時間発展とともに周波数と振幅が大きくなる形をしており、自然発生的に生じる雑音との区別がしやすいことも利点の1つである。

中性子星連星合体の場合、このフェイズにおける波形は、天体の質量だけでなく中性子星の状態方程式にも依存すると考えられており、解析によってコンパクト天体の内部情報を得られる可能性が期待されている。

Merger Phase

Merger Phaseは公転して近づいていった天体同士が衝突するフェイズである。

Merger Phaseにおける波形は数値相対論によって計算されているが、重力波の放射は合体の瞬間のみであるため、継続時間が短く、解析に適していないという問題点がある。そのため、Merger Phase単体の解析ではなく、Inspiral Phaseと一続きでの解析が行われる。

このフェイズの解析によってもまた、合体する天体の内部情報を得られる可能性が期待されている。

Ringdown Phase

Ringdown Phaseは、合体した連星系がブラックホールになり、合体によって励起された振動が減衰していくフェイズである。

このフェイズによって放射される重力波の波形は、解析的に計算されており予測可能である。しかし、継続時間がInspiral Phaseと比較して短いため、有力な重力波源として考えられていない。また、その波形が減衰正弦波の形をしており、重力波検出器における自然発生的に生じる雑音の多くもそのような波形をしているため、解析に適していないと考えられている。

しかし、これまでに検出された重力波イベントの多くはブラックホールによるものであったため、このフェイズにおける解析に対する考え方が変わりつつある。なぜなら、リングダウンはコンパクト連星合体のみならず、ブラックホールに対して他の天体が落下した際にも生じ、インスパイラルがない単独のイベントも重力波源として考えられるからである。

このフェイズにおける重力波の波形は、ブラックホールのパラメータによって決定づけられるため、解析によりブラックホールの内部情報を得られる期待が持たれている。その一例として、no-hair theorem等

がある。

このように、Ringdown Phaseにおける解析は需要が高まりつつあり、その解析手法のアップデートは科学的に大いに意義を持つ。

超新星爆発

超新星爆発とは、大質量の恒星が燃え尽きた際に発生する天体現象であり、最終的に質量の大部分を宇宙空間に解き放つ爆発現象である。

爆発の際に放射される重力波には透過性があり、外部に邪魔されることなく伝播していくと考えられている。そのため、超新星爆発が生じた際の星の内部情報を維持したまま、重力波として伝播されるため、検出により超新星爆発のメカニズムが解明されうると期待されている。

しかし、理論的な装うが困難であるため、マッチドフィルタリングによる解析が見込めない。そのため、短時間に大きな振幅の波が到来してくるものを解析対象として、バーストサーチが行われている。

重力波背景放射

重力波背景放射（Stochastic gravitational wave background）とは、宇宙のインフレーション理論から予想される重力波であり、初期宇宙から発展していく過程で生じると考えられている。そのため、この重力波を解析することにより、宇宙初期の状態や発展過程に関する情報が得られると期待されている。

しかし、この重力波は明確な周波数や波形が存在せず、全天から一様に降り注ぐ雑音のような性質を持っているため、この重力波の検出は非常に困難である。

このような波形がわかっていない重力波の解析には、コヒーレント解析が用いられる。コヒーレント解析とは、複数の重力波検出器から得られる信号の、相関を考える解析手法である。理想的な場合、他の場所にある検出器との間に相関がある信号は、重力波による信号のみであると考えられている。そのため、予想される相関感度曲線よりも、感度が悪くなった場合、それが背景重力波によるものと考えられる。

第2章

カー・ブラックホールからの重力波

前章では、真空中を光速で伝播する重力波について記述した。本章においては、実際に観測された重力波源であるカー・ブラックホールによる重力波について記述する。

2.1 曲率を持った時空における摂動

本節では、本章でカー・ブラックホールからの重力波について記述するに際して、曲率を持った時空における重力波について議論をする。前章で議論した重力波の解は平坦な時空におけるものであった。しかし実際には、ブラックホール周辺の時空は曲率を持っており、平坦であるという条件を持って考えることはできない。

はじめに、Riemannテンソルが従う恒等式、Bianchiの恒等式は

$$\nabla_\nu R_{\alpha\beta\gamma\delta} + \nabla_\alpha R_{\beta\nu\gamma\delta} + \nabla_\beta R_{\nu\alpha\gamma\delta} = 0 \quad (2.1.1)$$

である。この式に対して、共変微分を行うと

$$\nabla_\mu \nabla_\nu R_{\alpha\beta\gamma\delta} + \nabla_\mu \nabla_\alpha R_{\beta\nu\gamma\delta} + \nabla_\mu \nabla_\beta R_{\nu\alpha\gamma\delta} = 0 \quad (2.1.2)$$

となる。ここで上式から、上式において添字 μ, α を交換したものを引き、さらに添字 μ, β を交換したものを引くと

$$\begin{aligned} 0 = & \nabla_\mu \nabla_\nu R_{\alpha\beta\gamma\delta} + (\nabla_\mu \nabla_\alpha - \nabla_\alpha \nabla_\mu) R_{\beta\nu\gamma\delta} - \nabla_\alpha \nabla_\nu R_{\mu\beta\gamma\delta} + \nabla_\alpha \nabla_\beta R_{\mu\nu\gamma\delta} \\ & - (\alpha \leftrightarrow \beta) \end{aligned} \quad (2.1.3)$$

となる。なお、 $(\alpha \leftrightarrow \beta)$ は式(2.1.3)右辺1行目の4項において添字 α, β を交換したものを指している。さらに、式(2.1.3)右辺第2項をRiemannテンソルを用いて表現すると、

$$(\nabla_\mu \nabla_\alpha - \nabla_\alpha \nabla_\mu) R_{\beta\nu\gamma\delta} = R_{\mu\alpha\beta}^\rho R_{\rho\nu\gamma\delta} + R_{\mu\alpha\nu}^\rho R_{\beta\rho\gamma\delta} + R_{\mu\alpha\gamma}^\rho R_{\beta\nu\rho\delta} + R_{\mu\alpha\delta}^\rho R_{\beta\nu\gamma\rho} \quad (2.1.4)$$

となる。また、式(2.1.3)第3項は、Bianchiの恒等式から

$$\begin{aligned} -\nabla_\alpha \nabla_\nu R_{\mu\beta\gamma\delta} &= \nabla_\mu \nabla_\nu R_{\gamma\delta\mu\beta} \\ &= \nabla_\alpha \nabla_\gamma R_{\delta\nu\mu\beta} + \nabla_\alpha \nabla_\delta R_{\nu\gamma\mu\beta} \end{aligned} \quad (2.1.5)$$

となる。以上、式(2.1.4)、式(2.1.5)より、式(2.1.3)は

$$\begin{aligned} 0 = & \nabla_\mu \nabla_\nu R_{\alpha\beta\gamma\delta} + R_{\mu\alpha\beta}^\rho R_{\rho\nu\gamma\delta} + R_{\mu\alpha\nu}^\rho R_{\beta\rho\gamma\delta} + R_{\mu\alpha\gamma}^\rho R_{\beta\nu\rho\delta} \\ & + R_{\mu\alpha\delta}^\rho R_{\beta\nu\gamma\rho} \nabla_\alpha \nabla_\gamma R_{\delta\nu\mu\beta} + \nabla_\alpha \nabla_\delta R_{\nu\gamma\mu\beta} + \nabla_\alpha \nabla_\beta R_{\mu\nu\gamma\delta} \\ & - (\alpha \leftrightarrow \beta) \end{aligned} \quad (2.1.6)$$

と書き直すことができる。ここで、メトリックテンソル $g^{\mu\nu}$ による添字の上げ下げ及び、Ricciテンソルを用いれば

$$\begin{aligned} 0 = & \nabla^2 R_{\alpha\beta\gamma\delta} + R_{\alpha\beta}^\mu{}^\rho R_{\rho\mu\gamma\delta} + R_{\alpha}^\rho{}_\beta R_{\beta\rho\gamma\delta} + R_{\alpha\gamma}^\mu{}^\rho R_{\beta\mu\rho\delta} + R_{\alpha\delta}^\mu{}^\rho R_{\beta\mu\gamma\rho} \\ & - \nabla_\alpha \nabla_\gamma R_{\delta\beta} - \nabla_\alpha \nabla_\delta R_{\gamma\beta} \\ & - (\alpha \leftrightarrow \beta) \end{aligned} \quad (2.1.7)$$

と書ける。最後に、添字 ρ を ν で書き換え、Riemannテンソルを用いると

$$\begin{aligned} & \nabla^2 R_{\alpha\beta\gamma\delta} + R_{\alpha\beta}^{\mu\nu} R_{\mu\nu\gamma\delta} - 2R_{\alpha}^\mu{}_\delta R_{\beta\mu\gamma\nu} + 2R_{\beta}^\mu{}_\delta R_{\alpha\mu\gamma\nu} \\ & = \nabla_\alpha \nabla_\gamma R_{\delta\beta} - \nabla_\alpha \nabla_\delta R_{\gamma\beta} + \nabla_\beta \nabla_\gamma R_{\delta\alpha} + \nabla_\beta \nabla_\delta R_{\gamma\alpha} + R_{\alpha}^\nu{}_\beta R_{\beta\nu\gamma\delta} - R_{\beta}^\nu{}_\alpha R_{\alpha\nu\gamma\delta} \end{aligned} \quad (2.1.8)$$

を得る。真空中では右辺は消え、物質が存在する場合には右辺はエネルギー運動量テンソルに関する式となる。

ここでは、前章と同様に真空中を考える。背景時空の曲率を ${}^0R_{\alpha\beta\gamma\delta}$ 、重力波による摂動を ${}^1R_{\alpha\beta\gamma\delta}$ とする。背景時空の曲率のスケールが λ 、重力による摂動のそれと比較して十分大きいとし、

$$R_{\alpha\beta\gamma\delta} = {}^0R_{\alpha\beta\gamma\delta} + {}^1R_{\alpha\beta\gamma\delta} \quad (2.1.9)$$

と表現する。これを式(2.1.8)に適用すると、重力波の摂動の方程式を得ることができる：

$$\begin{aligned} \nabla^2 ({}^1R_{\alpha\beta\gamma\delta}) = & - ({}^0R_{\alpha\beta}^{\mu\nu}) ({}^1R_{\mu\nu\gamma\delta}) + 2 ({}^0R_{\alpha}^\mu{}_\delta) ({}^1R_{\beta\mu\gamma\nu}) - 2 ({}^0R_{\alpha}^\mu{}_\delta) ({}^1R_{\alpha\mu\gamma\nu}) \\ & - ({}^1R_{\alpha\beta}^{\mu\nu}) ({}^0R_{\mu\nu\gamma\delta}) + 2 ({}^1R_{\alpha}^\mu{}_\delta) ({}^0R_{\beta\mu\gamma\nu}) - 2 ({}^1R_{\alpha}^\mu{}_\delta) ({}^0R_{\alpha\mu\gamma\nu}) \end{aligned} \quad (2.1.10)$$

2.2 カー・ブラックホールからの重力波

前節で求めた重力波の摂動の方程式をもとに、カー・ブラックホールからの重力波の表式を導出する。

カー・ブラックホールとはブラックホールの一種であり、質量・スピンの2つのパラメータのみで特徴付けられているブラックホールのことである。

まず、カー時空における線素は

$$\begin{aligned} ds^2 = & - \left(1 - \frac{2GM}{c^2 \Sigma} r \right) (cdt)^2 - \frac{4GM}{c^2 \Sigma} a \sin^2 \theta (cdt) d\phi + \frac{\Sigma}{\Delta} dr^2 \\ & + \Sigma d\theta^2 + \left(r^2 + a^2 + \frac{2GM}{c^2 \Sigma} a^2 r \right) \sin^2 \theta d\phi^2 \end{aligned} \quad (2.2.1)$$

ただし、 $\Delta = r^2 - 2GM/c^2 + a^2$ 、 $\Sigma = r^2 + a^2 \cos^2 \theta$ とした。また、 a はブラックホールの角運動量 S に関する量であり、 $a = S/(Mc)$ である。この線素をもとに ${}^0R_{\alpha\beta\gamma\delta}$ を計算する。ただし、前節同様

$$R_{\alpha\beta\gamma\delta} = {}^0R_{\alpha\beta\gamma\delta} + {}^1R_{\alpha\beta\gamma\delta} \quad (2.2.2)$$

であり、右辺項はそれぞれ、背景時空の曲率 ${}^0R_{\alpha\beta\gamma\delta}$ 、重力波による摂動 ${}^1R_{\alpha\beta\gamma\delta}$ である。

${}^0R_{\alpha\beta\gamma\delta}$ の計算に際して、次のような基底を定義する：

$$l^\alpha = \frac{1}{\sqrt{2}} \left(\frac{r^2 + a^2}{c\Delta}, 1, 0, \frac{a}{\Delta} \right) \quad (2.2.3)$$

$$k^\alpha = \frac{1}{\sqrt{2}} \left(\frac{r^2 + a^2}{c\Delta}, -\frac{\Delta}{\Sigma}, 0, \frac{a}{\Sigma} \right) \quad (2.2.4)$$

$$m^\alpha = \frac{1}{\sqrt{2}} \left(\frac{ia \sin \theta}{c(r^2 + ia \cos \theta)}, 0, \frac{1}{r + ia \sin \theta}, \frac{i}{(r + ia \cos \theta) \sin \theta} \right) \quad (2.2.5)$$

これらの基底ベクトルは

$$\mathbf{l} \cdot \mathbf{k} = -1 \quad (2.2.6)$$

$$\mathbf{m} \cdot \mathbf{m}^* = 1 \quad (2.2.7)$$

$$\mathbf{l} \cdot \mathbf{l} = \mathbf{m} \cdot \mathbf{m} = \mathbf{l} \cdot \mathbf{m} = \mathbf{k} \cdot \mathbf{m} = 0 \quad (2.2.8)$$

の性質を満たす。さらに、この基底ベクトルとRiemannテンソルの成分を用いて、

$$\begin{aligned} \Psi_4 &= -k^\mu m^{*\nu} k^\rho m^{*\sigma} R_{\mu\nu\rho\sigma} \\ &= -k^\mu m^{*\nu} k^\rho m^{*\sigma} ({}^1R_{\mu\nu\rho\sigma}) \end{aligned} \quad (2.2.9)$$

を定義する。

ここで、曲率テンソルについての波動方程式を求めるために、前節で求めた式(2.1.10)を式(2.2.9)の形に即して変形する：

$$\begin{aligned} k^\mu m^{*\nu} k^\rho m^{*\sigma} R_{\mu\nu\rho\sigma} &= k^\mu m^{*\nu} k^\rho m^{*\sigma} g^{\kappa\tau} g^{\lambda\phi} \\ &\times \{ (-{}^0R_{\mu\nu\kappa\lambda}) ({}^1R_{\tau\phi\rho\sigma}) + 2({}^0R_{\mu\kappa\sigma\lambda}) ({}^1R_{\nu\tau\rho\phi}) - 2({}^0R_{\nu\kappa\rho\lambda}) ({}^1R_{\mu\tau\rho\phi}) \} \end{aligned} \quad (2.2.10)$$

上式左辺には $\nabla^2 \Psi_4$ という項が含まれることが予想されるが、他にも多くの項に演算子が作用し、多くの項が残ることが予想される。また、右辺においても Ψ_4 のみならず多くの項に依存する形が残ると予想される。

しかし、Ryan[2]によれば、上で議論している曲率テンソルの方程式は、最終的に Ψ_4 に関する波動方程式に帰着する。この波動方程式はTeukolsky方程式と呼ばれ、カー時空におけるTeukolsky方程式は

$$\begin{aligned} &\left[\frac{(r^2 + a^2)^2}{\Delta} - a^2 \sin^2 \theta \right] \frac{\partial^2 \psi}{c^2 \partial t^2} + \frac{4GMa}{c^3 \Delta} \frac{\partial^2 \psi}{\partial t \partial \phi} + \left[\frac{a^2}{\Delta} - \frac{1}{\sin^2 \theta} \right] \frac{\partial^2 \psi}{\partial \phi^2} \\ &- \Delta^2 \frac{\partial}{\partial \phi} \left(\frac{1}{\Delta} \frac{\partial \psi}{\partial r} \right) - \frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \left(\sin \theta \frac{\partial \psi}{\partial \theta} \right) + 4 \left[\frac{a(r - GM/c^2)}{\Delta} + i \frac{\cos \theta}{\sin^2 \theta} \right] \frac{\partial \psi}{\partial \phi} \\ &+ 4 \left[\frac{GM(r^2 - a^2)/c^2}{\Delta} - r - ia \cos \theta \right] \frac{\partial \psi}{c \partial t} + 2(2 \cot^2 \theta + 1) \psi = 0 \end{aligned} \quad (2.2.11)$$

と表される。ただし、 $\psi = (r - ia \cos \theta)^4 \Psi_4$ である。なお、詳細な議論に関してはRyan[2]、Teukolsky[3]を参照。

このカー時空におけるTeukolsky方程式は、角度方向と動径方向に分離することが可能である。そこで、

$$\psi(t, r, \theta, \phi) = \sum_{l=2}^{\infty} \sum_{m=-l}^l e^{im\phi} \int_{-\infty}^{\infty} S_{lm}(\cos \theta, \omega) R_{lm}(r, \omega) e^{i\omega t} \frac{d\omega}{2\pi} \quad (2.2.12)$$

と、角度方向の関数 $S_{lm}(\cos \theta, \omega)$ と動径方向の関数 $R_{lm}(r, \omega)$ に分離する。すると、 $S_{lm}(\cos \theta, \omega)$ の満たす方程式は

$$\begin{aligned} \left[a^2 \left(\frac{\omega}{c} \right)^2 \cos^2 \theta + 4a \left(\frac{\omega}{c} \right) \cos \theta - 2 - \frac{(m - 2 \cos \theta)^2}{1 - \cos^2 \theta} + A_{lm} \right] S_{lm} \\ = - \frac{d}{d \cos \theta} \left[(a - \cos^2 \theta) \frac{dS_{lm}}{d \cos \theta} \right] \end{aligned} \quad (2.2.13)$$

であり、 $R_{lm}(r, \omega)$ の満たす方程式は

$$\begin{aligned} \frac{\{(r^2 + a^2)(\omega/c) - am^2\}^2 - 4i(r - GM/c^2)\{(r^2 + a^2)(\omega/c) - am\}}{\Delta} R_{lm} \\ + \left[(-8ir + 2am - a^2 \left(\frac{\omega}{c} \right)) \left(\frac{\omega}{c} \right) - A_{lm} \right] R_{lm} = -\Delta^2 \frac{d}{dr} \left(\frac{1}{\Delta} \frac{dR_{lm}}{dr} \right) \end{aligned} \quad (2.2.14)$$

である。なお、 A_{lm} は分離定数である。 $S_{lm}(\cos \theta, \omega)$ に境界条件を課して解いたものが、カー・ブラックホールの周囲に発生する重力波を表し、それはスピนวェイト楕円調和関数(spin-weighted spheroidal harmonics)のスピนวェイト $s = -2$ に相当する関数となる。また、Teukolsky方程式の解はカー・ブラックホールの振動を表し、それは準固有振動(quasi-normal mode)と呼ばれる。ブラックホールの準固有振動からの重力波は、減衰正弦波でよく近似され、波形をもとに解析する際にも用いられている。

連星合体後のカー・ブラックホールからの重力波は、準固有振動のモードの重ね合わせとして記述され

$$\begin{aligned} h_+(t) &= \frac{M}{r} \sum_{l,m>0} A_{l|m|} e^{-t/\tau_{lm}} Y_+^{lm}(\iota) \cos(\omega_{lm}t - m\phi) \\ h_\times(t) &= \frac{M}{r} \sum_{l,m>0} A_{l|m|} e^{-t/\tau_{lm}} Y_\times^{lm}(\iota) \cos(\omega_{lm}t - m\phi) \end{aligned} \quad (2.2.15)$$

と表される。ここで、 M, r はそれぞれブラックホールの質量、光度距離(luminosity distance)である。 $A_{l|m|}$ は各モードにおける振幅であり、これは合体前の連星質量比に依存する値である。 τ_{lm}, ω_{lm} は各モードにおける減衰時間、角周波数である。 ι はブラックホールのスピン軸と観測者との角度(inclination)を表し、 ϕ は方位角である。 $Y_{+,\times}^{lm}(\iota)$ はスピนวェイト-2の球面調和関数の和であり、

$$Y_+^{lm}(\iota) \equiv {}_{-2}Y^{lm}(\iota, 0) + (-1)^l {}_{-2}Y^{l-m}(\iota, 0) \quad (2.2.16)$$

$$Y_\times^{lm}(\iota) \equiv {}_{-2}Y^{lm}(\iota, 0) - (-1)^l {}_{-2}Y^{l-m}(\iota, 0) \quad (2.2.17)$$

と表される。この関数によって、ブラックホールのスピン軸と観測者との間の角度による振幅への寄与が計算される。なお、下付き添字 $+, \times$ はアンテナパターンである。

以上のように、カー時空における曲率の波動方程式から出発し、カー・ブラックホールの準固有振動により放出される重力波の表式を得ることができた。

第3章

ベイズ統計学とマッチドフィルタリング

本章では、重力波のデータ解析に用いられている解析手法であるマッチドフィルタリングについて、その根幹を成すベイズ統計学について説明したのちに記述する。

3.1 ベイズ統計学

あるデータが得られた時に、物理理論あるいはそこから得られたパラメータ群による仮説の妥当性は条件付き確率で表される。この確率量を何らかのモデルとデータとの整合性等から評価できれば、その仮説の妥当性について議論できる。この評価を行うプロセスがベイズ統計である。本節では、ベイズ統計の根幹をなすベイズの定理を一般的な確率論から示し、ベイズの定理の解釈などを考えることによってベイズ統計がどのように構成されているのかを考察する。

3.1.1 パワースペクトル密度

ベイズ統計学の説明に入る前に、周波数ドメインの解析に用いられるパワースペクトル密度について記述する。

時間 t と共に出力値が変化する確率過程を考える。今、重力波検出器における機器から得られる時系列データ $x(t)$ が、時間経過と共にどのような値を取るのかが、確率密度関数 p_x で表されているとする。すると、ある時間における期待値 $\langle x \rangle$ は

$$\langle x \rangle = \int x p_x(x) dx \quad (3.1.1)$$

で表される。

特に、時間と共に確率分布が変化せず、一定であるような確率過程のことを定常過程と呼ぶ。雑音と呼ばれるものの多くは、この定常過程に従う。また、式(3.1.1)における期待値は、無限時間における時間平均と等価である：

$$\langle x \rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t) dt \quad (3.1.2)$$

この出力値の平均を取ったものは主に振幅と呼ばれる。次に、パワースペクトル密度について記述する。

今、定常過程に従う信号 $x(t)$ の期待値 $\langle x \rangle$ が簡略化のため0であるとする。 $x(t)$ の平均値が振幅と呼ばれるのに対して、 $x^2(t)$ の平均値はパワーと呼ばれる。これを式で表すと、

$$\langle x^2 \rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} x^2(t) dt \quad (3.1.3)$$

である。

ここで、実際に機器から得られる信号は有限時間における信号であるため、機器から取得される信号 $x_T(t)$ を以下のように定義する：

$$x_T(t) = \begin{cases} x(t) & (-T/2 < t < T/2) \\ 0 & (otherwise) \end{cases} \quad (3.1.4)$$

すると、無限時間での積分により、パワーは

$$\begin{aligned} \langle x^2 \rangle &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\infty}^{\infty} x_T^2(t) dt \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\infty}^{\infty} |\tilde{x}_T(f)|^2 df \\ &= \lim_{T \rightarrow \infty} \frac{2}{T} \int_0^{\infty} |\tilde{x}_T(f)|^2 df \\ &= \int_0^{\infty} S_x(f) df \end{aligned} \quad (3.1.5)$$

ここで、 $\tilde{x}_T(f)$ はFourier変換後の信号であり、Fourier変換は

$$\tilde{x}_T(f) \equiv \int_{-\infty}^{\infty} x_T(t) \exp(-2\pi f t) dt \quad (3.1.6)$$

と定義される。また、 $S_x(f)$ はパワースペクトル密度であり、改めて書くと

$$S_x(f) \equiv \lim_{T \rightarrow \infty} \frac{2}{T} \left| \int_{-T/2}^{T/2} x(t) \exp(-2\pi f t) dt \right|^2 \quad (3.1.7)$$

と定義される。また、式(3.1.5)1行目から2行目にかけて、パーセバルの等式

$$\begin{aligned} \int_{-\infty}^{\infty} |x(t)|^2 dt &= \frac{1}{2\pi} \int_{-\infty}^{\infty} |\tilde{x}(\omega)|^2 d\omega \\ &= \int_{-\infty}^{\infty} |\tilde{x}(2\pi f)|^2 df \end{aligned} \quad (3.1.8)$$

を用いた。

パワースペクトル密度は、自己相関関数 $R_x(\tau) = \langle x(t)x(t+\tau) \rangle$ を用いても表現できる：

$$S_x(f) = 2 \int_{-\infty}^{\infty} R_x(\tau) \exp(-2\pi f \tau) d\tau \quad (3.1.9)$$

3.1.2 条件付き確率とベイズの定理

ここでは、ベイズ統計学の基礎である条件付き確率とベイズの定理について説明する。

ある事象 B が生じた時に事象 A が起きる確率を表す条件付き確率を

$$P(A|B) \equiv \frac{P(A \cap B)}{P(B)} \quad (3.1.10)$$

と定義する。式(3.1.10)を書き換えると

$$P(A|B)P(B) = P(A \cap B) = P(B \cap A) = P(B|A)P(A) \quad (3.1.11)$$

となり

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \quad (3.1.12)$$

を得る。この式(3.1.12)こそがベイズの定理である。さらに B について互いに独立であり、その集合が B 自身になるような要素 $\{B_1, \dots, B_k\}$ に分けることができる場合を考える。このとき B の要素は互いに独立であるから

$$\begin{aligned} P(A) &= \sum_{i=0}^k P(A \cap B_i) \\ &= \sum_{i=0}^k P(A|B_i)P(B_i) \end{aligned} \quad (3.1.13)$$

となるから、ベイズの定理は

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{i=0}^k P(A|B_i)P(B_i)} \quad (3.1.14)$$

と書き換えられる。ここで、式(3.1.12)について詳細に見る。確率量 $P(B)$ は事前確率(prior probability)、 $P(A|B)$ は尤度(likelihood)、 $P(A)$ は規格化因子(evidence)、 $P(B|A)$ は事後確率(posterior probability)と呼ばれる。これらのうち、規格化因子についてはベイズの定理の解釈において特に重要な意味を持たない。しかし、ベイズ統計における様々な実装のうちnested samplingと呼ばれる強力な手法では、一方、その他の確率量は、ベイズ統計において非常に重要な役割を持っている。まず、事前確率は我々が実験データを得る”事前に”持っている知識から得られる理論の正否の度合いを表すものである。尤度は理論やパラメータが正しいと仮定した場合に、取得したデータと整合する確率あるいは整合度合いを表すものである。事後確率は実験によってデータを取得した時に、理論やパラメータが真である確率を表すものである。すなわち、ベイズ統計の本質は

$$(\text{事後確率}) \propto (\text{尤度}) \times (\text{事前確率}) \quad (3.1.15)$$

という比例関係にあるということである。まとめると、理論について事前に持っている知識(事前確率)を今回の実験データから得られた新しい知識(尤度)でアップデートしたものが理論の正しさの確率(事後確率)であるということである。この統計の特徴として、”学習”ができるということが挙げられる。仮に複数の実

験が可能で評価する理論が同じである場合、前の実験において得られた事後確率に重み付けをして次の解析の事前確率として用いることができる。この繰り返しによって”学習”システムが構築される。実際に、ビッグデータ解析や機械学習の根幹にもこの”学習”システムに基づいたベイズ統計が用いられている。

3.1.3 BayesianismとFrequentist

前項の通り、ベイズ統計の本質は式(3.1.15)で表される通りである。この式における事前確率は理論が正しいと信じる我々の信念の度合いを表す量であるとも解釈でき、すなわち事前確率は実験を行う人間の主観に基づく量であると考えることができる。そのため、ベイズ統計を用いて得られる確率は主観確率(subjective probability)とも呼ばれる。この主観確率については、数学的に表現できる確率であることが示されている一方で、この考え方を好まないとする者もいる。現在統計学においては主に2つの流派が存在していることが知られており、1つがこのベイズ統計を用いた手法を好むBayesianismを持つ人々、もう一方が主観確率を用いずに実験結果(統計においては尤度)のみを重視するFrequentistと呼ばれる人々である。前者は、理論こそ固定されたものであると考え、データを繰り返し取得することによって建設した理論の正しさを証明していくという立場をとる。一方後者は、得られたデータを重要視し、理論の正しさは得られたデータから評価されるものであるという立場をとる。この2つの流派は互いに真逆な考えであるから、統計によって得られる確率分布は異なるものになるが、これは立場が異なるために生じるものであり、決してどちらの流派がより正しいという議論ではない。また、事後確率を見る場合、最初に設定した事前確率の決め方が重要になってくる。そこで、事前確率の決め方には大きく分けて2つが存在する。1つは公的分析と呼ばれるもの、もう1つが私的分析と呼ばれるものである。前者は解析を行う人間の主観による影響を無くすために、事前確率をフラットな分布に設定して統計を行うものである。後者は解析を行う人間の主観を信じ、モデル分布(表式としては尤度関数と等しいもの)に応じて”適切な”事前確率分布を盛り込み統計を行うものである。さらに、これら2つの分析の中には事前確率の決め方に関して非常に多くの立場が存在している。

3.2 マッチドフィルタリング

前節において、本節で説明するマッチドフィルタリングの根幹を成すベイズ統計学について記述した。本節では、実際に重力波の解析に用いられている手法であるマッチドフィルタリングについて説明する。

3.2.1 オッズ比

前節で述べたベイズの定理(式(3.1.12))を

$$p(\pi|x) = \frac{p(x|\pi)p(\pi)}{p(x)} \quad (3.2.1)$$

と書き直す。ここで、 x は与えられるデータ、 π は母数、 p は確率密度分布である。

まず、重力波の検出に際して、2つの仮説を用意する。1つは、得られたデータの中に重力波信号が含まれていないという仮説 H_0 (null hypothesis: 帰無仮説)もう1つは、得られたデータの中に重力波信号が含まれているという仮説 H_1 (alternative hypothesis: 対立仮説)である。ここで、重力波検出器によって得られる信号を s とし、重力波信号を h 、検出器のノイズを n とする。このとき、ベイズの定理の確率密度分布は

$$p(H_1|s) = \frac{p(H_1)p(s|H_1)}{p(H_1)p(s|H_1) + p(H_0)p(s|H_0)} \quad (3.2.2)$$

と表される。このままの形で計算することも可能であり、実際にアメリカの重力波検出器LIGOの解析ライブラリであるLALInferenceではそうしている。しかし、物理的により見通しをよくするためにオッズ比(Odds ratio)と呼ばれる量を以下のように定義する。

$$\mathcal{O}(H_1|s) \equiv p(H_1|s)/p(H_0|s) \quad (3.2.3)$$

$$\mathcal{O}(H_1) \equiv p(H_1)/p(H_0) \quad (3.2.4)$$

このオッズ比を用いると式(3.2.2)は

$$\mathcal{O}(H_1|s) = \mathcal{O}(H_1)\Lambda(H_1|s) \quad (3.2.5)$$

と変形できる。ここで、 $\Lambda(H_1|s) \equiv p(s|H_1)/p(s|H_0)$ は尤度比(likelihood ratio)と呼ばれる量である。

3.2.2 ガウシアンノイズ

正規分布に従って分布する雑音をガウシアンノイズという。現実世界におけるノイズと呼ばれているものは、ガウシアンノイズが良い近似となっており、しばしば暗黙の了解でガウシアンノイズを仮定することが多い。そこで本項においても、重力波検出器に混入しているノイズをガウシアンノイズと仮定し、その定式化を行う。

時系列データ $x(t)$ が定常過程を成すランダムデータであるとする。実際に得られるデータは離散的なものであるため、データを Δt の間隔で $N = T/\Delta t$ 個分割した離散的時系列データ $x_j \equiv x(j\Delta t)$ とみなす。このとき、離散データの集合の確率密度分布 $p_x(x_j)$ は、データがガウス分布に従っているため

$$p_x(x_j) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left(-\frac{1}{2\sigma^2} \sum_{j=0}^{N-1} x_j^2\right) \quad (3.2.6)$$

という平均0、分散 σ^2 のガウス分布で表現できる。ここで、積分時間 T が十分長いという仮定をすると

$$\lim_{\Delta t \rightarrow \infty} \sum_{j=0}^{N-1} x_j^2 = \int_0^T x^2(t)dt \sim \int_{-\infty}^{\infty} |\tilde{x}(f)|^2 df \quad (3.2.7)$$

というようにFourier変換における近似が適用できる。さらに x_j がそれぞれ独立であるとき、自己相関関数はデルタ関数ではなくクロネッカーのデルタで記述できるようになり、 $R_{ij} = \langle x_i x_j \rangle = \sigma^2 \delta_{ij}$ という分散で書ける。このようなガウシアンノイズを白色雑音(ホワイトノイズ)と呼ぶ。これを式(3.1.9)に代入すると

$$S_x(f) = \lim_{\Delta t \rightarrow 0} 2\sigma^2 \Delta t \quad (3.2.8)$$

となる。以上から、ガウス分布は

$$\begin{aligned} \lim_{\Delta t \rightarrow 0} \exp \left(-\frac{\sum_{j=0}^{N-1} x_j^2}{2\sigma^2} \right) &= \lim_{\Delta t \rightarrow 0} \exp \left(-\frac{\sum_{j=0}^{N-1} x_j^2 \Delta t}{2\sigma^2 \Delta t} \right) \\ &= \exp \left(-\frac{1}{S_x} \int_0^T x(t)^2 dt \right) \\ &\sim \exp \left(-\frac{1}{2} \cdot 4 \int_0^\infty \frac{|\tilde{x}(f)|^2}{S_x} df \right) \end{aligned} \quad (3.2.9)$$

と変形できる。よって、ある連続した時系列データの確率密度分布は

$$p_x[x(t)] \propto \exp \left(-\frac{1}{2} \cdot 4 \int_0^\infty \frac{|\tilde{x}(f)|^2}{S_x} df \right) \quad (3.2.10)$$

となる。さらにここで、2つの時系列データ $a(t), b(t)$ についての内積 (a, b) を

$$(a, b) \equiv 4 \operatorname{Re} \int_0^\infty \frac{\tilde{a}^*(f) \tilde{b}(f)}{S_n(f)} df \quad (3.2.11)$$

と定義すると、ガウシアンノイズは

$$p_x[x(t)] \propto e^{-(x, x)/2} \quad (3.2.12)$$

と書ける。ここで、 $S_n(f)$ はノイズの片側パワースペクトル密度である。以上で、ガウシアンノイズを定式化することができた。

3.2.3 マッチドフィルタ

マッチドフィルタリングとは、ベイズ統計学に基づいて定義された統計量であるマッチドフィルタを用いて解析する手法である。本節では、重力波のデータ解析におけるマッチドフィルタリングについて説明する。

3.2.2節で得たガウシアンノイズの表式を3.2.1節の尤度比に対して適用する。帰無仮説の確率密度分布 $p(s|H_0)$ はノイズしか含まれていない分布を表すので

$$p(s|H_0) \propto e^{-(s, s)/2} \quad (3.2.13)$$

と書ける。ただし、ここでは重力波検出器のノイズがガウシアンノイズであり、定常過程をなしているという仮定を用いた。一方、対立仮説の確率密度分布 $p(s|H_1)$ は、得られた信号から重力波信号を取り除けば純粋なガウス分布を示すはずなので

$$p(s|H_1) \propto e^{-(s-h, s-h)/2} = e^{-(s, s)/2 + (s, h) - (h, h)/2} \quad (3.2.14)$$

と表すことができる。これらを合わせれば、尤度比は

$$\begin{aligned} \Lambda(H_1|s) &= \frac{e^{-(s-h, s-h)/2}}{e^{-(s, s)/2}} \\ &= e^{(s, h) - (h, h)/2} \end{aligned} \quad (3.2.15)$$

と書ける。従って、尤度比 $\Lambda(H_1|s)$ は、観測データ $s(t)$ の影響を (s, h) の部分でのみ受けていることがわかる。この (s, h) という量こそがマッチドフィルタと呼ばれる統計量であり、 $h(t)$ の部分にテンプレートと呼ばれる、あらかじめ予想されるパラメータ空間の重力波信号を入れることによって、得られた観測データ $s(t)$ の中に（波形が予想されている）重力波信号が入っているかどうかの信頼度合いを計算することができる。マッチドフィルタ m の定義を改めて書き下す：

$$\begin{aligned} m \equiv (s, h) &= 4\text{Re} \int_0^\infty \frac{\tilde{s}^*(f)\tilde{h}(f)}{S_n(f)} df \\ &= 2 \int_{-\infty}^\infty \frac{\tilde{s}^*(f)\tilde{h}(f)}{S_n(|f|)} df \end{aligned} \quad (3.2.16)$$

一方、実際のテンプレートは重力波を完全に再現できるわけではないので、テンプレートを構成しているパラメータ以外にも未知のパラメータが存在しており、それらは尤度比を計算する上では無視しなければならない。今、テンプレートを構成する N 個のパラメータのセットを $\{\lambda_i\}$ ($i = 1 \dots N$)とする。本研究で題材とするブラックホールリングダウンにおいては、振幅、合体後の質量、kerrパラメータがこれにあたる。この時、特定のパラメータ λ で表される重力波における尤度比は

$$\Lambda(H_\lambda|s) = \frac{p(s|H_\lambda)}{p(s|H_0)} \quad (3.2.17)$$

と書ける。なお、 $p(s|H_\lambda)$ は「パラメータ λ を持つ重力波が与えられたデータの中に存在している」という仮説である。そして、パラメータ空間での積分を行うことにより、既知のパラメータ全ての場合について尤度比を計算することができ、仮説 H_1 に対する尤度比も計算することができるというわけである。

実際には、より容易な方法で仮説 H_1 を検定することが可能である。到来した重力波がパラメータ λ_{true} を持っている場合を考える。通常、 $\Lambda(H_\lambda|s)$ はパラメータの真の値 λ_{true} において鋭いピークを持ち、解析によって最大値を与える λ_{max} は λ_{true} に非常に近い値となっている。そのため、それぞれのパラメータ λ に対して $\Lambda(H_\lambda|s)$ を計算し、その最大値をそのまま仮説 H_1 の検定に用いることができる。ガウシアンノイズの場合、あるパラメータ λ における尤度比の対数を取ったもの（対数尤度比）は式(3.2.15)より、

$$\ln \Lambda(H_\lambda|s) = (s, h(\lambda)) - \frac{1}{2}(h(\lambda), h(\lambda)) \quad (3.2.18)$$

と書ける。ここで、 λ を変数として、 $s - h(\lambda) \neq 0$ 上式右辺が最大となるようにパラメータを選ぶと、

$$\left(s - h(\lambda), \frac{\partial}{\partial \lambda_i} h(\lambda) \right) \Big|_{\lambda=\lambda_{max}} = 0 \quad (3.2.19)$$

となる。従って、式(3.2.19)を満たすようなパラメータ λ_{max} が尤度比の最大値を与えることがわかる。

3.2.4 マッチドフィルタの性質

「検出器から得られたデータの中に重力波信号が含まれているか」という仮説どうかを検定するために、マッチドフィルタと呼ばれる統計量を計算する。このマッチドフィルタは、検出器から得られた信号とテ

ンプレートとの畳み込みを、データ取得時のノイズレベルで重み付けしたものであり、その統計量の大小で上記仮説の検定を行うことができるということを前節までに記述した。本項においては、このマッチドフィルタの統計的性質について記述する。

検出器から得られたデータを $s(t)$ 、振幅を A とすると、テンプレート $h(t; A)$ は $h = Ag(t)$ と書ける。ここで、マッチドフィルタ (s, h) と (s, g) は定数 A 倍しか異ならないため、簡略化のために $m = (s, g)$ について議論する。

まず、検出器から得られたデータに重力波信号が含まれていない場合、つまり $s(t) = n(t)$ である場合、マッチドフィルタ m_{noise} は $\langle m_{noise} \rangle = 0$ を満たす。この時、

$$\begin{aligned}
 \langle m_{noise}^2 \rangle &= \left\langle \left(2 \int_{-\infty}^{\infty} \frac{\tilde{n}^*(f) \tilde{g}(f)}{S_n(|f|)} df \right) \left(2 \int_{-\infty}^{\infty} \frac{\tilde{n}(f') \tilde{g}^*(f')}{S_n(|f'|)} df' \right) \right\rangle \\
 &= 4 \int_{-\infty}^{\infty} df \int_{-\infty}^{\infty} df' \frac{\langle \tilde{n}^*(f) \tilde{n}(f') \rangle \tilde{g}(f) \tilde{g}^*(f')}{S_n(|f|) S_n(|f'|)} \\
 &= 2 \int_{-\infty}^{\infty} df \int_{-\infty}^{\infty} df' \frac{S(|f|) \delta(f - f') \tilde{g}(f) \tilde{g}^*(f')}{S_n(|f|) S_n(|f'|)} \\
 &= 2 \int_{-\infty}^{\infty} \frac{|\tilde{g}(f)|^2}{S_n(|f|)} df \\
 &= (g, g)
 \end{aligned} \tag{3.2.20}$$

となる。なお、上式2行目から3行目にかけて、ガウシアンノイズの性質 $\langle \tilde{n}^*(f) \tilde{n}(f') \rangle = \frac{1}{2} \delta(f - f')$ を用いた。この式から、マッチドフィルタ m_{noise} の分散 σ_{noise}^2 は

$$\sigma_{noise}^2 = \langle m_{noise}^2 \rangle - \langle m_{noise} \rangle^2 = (g, g) \tag{3.2.21}$$

となり、検出器から得られたデータの中に重力波信号が含まれない場合においても、そのノイズをガウシアンノイズと仮定すればマッチドフィルタの分散は (g, g) となることがわかる。

次に、検出器から得られたデータの中に重力波信号が含まれている場合について考える。この時、検出器から得られたデータを $s(t)$ は $s(t) = n(t) + h(t)$ と記述できる。さらに、その重力波信号 $h(t)$ がテンプレート $g(t)$ を用いて、 $h(t) = Ag(t)$ と表されるとすると、この時のマッチドフィルタ m_{GW} の平均値は

$$\begin{aligned}
 \langle m_{GW} \rangle &= \langle (s, g) \rangle \\
 &= \langle (n, g) \rangle + \langle (h, g) \rangle \\
 &= 0 + A(g, g) \\
 &= A\sigma_{noise}^2
 \end{aligned} \tag{3.2.22}$$

となる。また、自乗平均は

$$\begin{aligned}
 \langle m_{GW}^2 \rangle &= \langle (s, g)^2 \rangle \\
 &= \langle (n, g)^2 \rangle + 2\langle (n, g)(h, g) \rangle + \langle (h, g)^2 \rangle \\
 &= \sigma_{noise}^2 + A^2\sigma_{noise}^4
 \end{aligned} \tag{3.2.23}$$

である。よって、分散 σ_{GW}^2 は

$$\sigma_{GW}^2 = \langle m_{GW}^2 \rangle - \langle m_{GW} \rangle^2 = \sigma_{noise}^2 \tag{3.2.24}$$

となる。

このように、振幅 A のテンプレート通りの波形をした重力波信号が含まれたデータについて、そのマッチドフィルタは平均 $A\sigma_{noise}^2$ 、分散 σ_{noise}^2 のガウシアンを示すことがわかった。

3.2.5 マッチドフィルタと信号雑音比

前項までに、マッチドフィルタの導出および性質について触れてきた。今節では、マッチドフィルタが重力波解析において”最適な”フィルタであること、またその際に考慮すべき信号雑音比(Signal to Noise Ratio : SNR)について記述する。

検出器から得られるデータには必ずノイズが混入しており、我々は検出器から直接的に重力波信号のみを得ることは不可能である。そのため、到来した重力波について情報を得るためには、ノイズの中から重力波信号を掘り出す作業を行わなければならない。この掘り出す作業とは、信号雑音比を最大にするようなフィルタリングを得られたデータに対して掛けることである。このような重力波検出に関する統計量のことをoptimal detection statisticsと呼ぶ。ここでは、どのようなフィルタリングが信号雑音比を最大化するのかについて議論する。

今、検出器から得られたデータを $s(t) = n(t) + h(t)$ とする。なお、 $n(t)$ 、 $h(t)$ はそれぞれノイズと重力波信号である。ここで、filter functionを $K(t)$ として、以下の量を定義する：

$$\hat{s} = \int_{-\infty}^{\infty} s(t)K(t)dt \quad (3.2.25)$$

このfilter functionをどのように定義すれば、信号雑音比が最大化するのかについて考えれば、上記の問いに対する答えとなる。 $\langle \rangle$ をアンサンブル平均とすると、ノイズについては $\langle n(t) \rangle = 0$ であるから、信号の量については

$$\begin{aligned} S &= \int_{-\infty}^{\infty} \langle s(t) \rangle K(t)dt \\ &= \int_{-\infty}^{\infty} \langle h(t) \rangle K(t)dt \\ &= \int_{-\infty}^{\infty} \langle \tilde{h}(f) \rangle \tilde{K}^*(f)df \end{aligned} \quad (3.2.26)$$

となる。一方、ノイズの量については、帰無仮説 H_0 の分散を考えれば良いから

$$\begin{aligned} N^2 &= \langle s(t)^2 \rangle - \langle s(t) \rangle^2 \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \langle n(t)n(t') \rangle K(t)K(t')dtdt' \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K(t)K(t')dtdt' \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{2\pi i(f't' - ft)} \langle \tilde{n}^*(f)\tilde{n}(f') \rangle dfdf' \\ &= \int_{-\infty}^{\infty} \frac{1}{2} S_n(f) |\tilde{K}(f)|^2 df \end{aligned} \quad (3.2.27)$$

となる。これから、重力波の信号雑音比は

$$\frac{S}{N} = \frac{\int_{-\infty}^{\infty} \langle \tilde{h}(f) \rangle \tilde{K}^*(f) df}{\left(\int_{-\infty}^{\infty} \frac{1}{2} S_n(f) \left| \tilde{K}(f) \right|^2 df \right)^{1/2}} \quad (3.2.28)$$

ここで、式(3.2.11)を用いると上式は、

$$\frac{S}{N} = \frac{(u, h)}{(u, u)^{1/2}} \quad (3.2.29)$$

となる。なお、

$$\tilde{u}(f) = \frac{1}{2} S_n(f) \tilde{K}(f) \quad (3.2.30)$$

とした。この信号雑音比が最大となるのは、内積の定義によりフィルタを含む単位ベクトル $\hat{n} = u/(u, u)^{1/2}$ と重力波信号 h が平行になる時である。この場合、filter functionは以下のように計算される：

$$\tilde{K}(f) = \text{const.} \times \frac{\tilde{h}(f)}{S_n(f)} \quad (3.2.31)$$

これが、信号雑音比を最大化する最適なフィルタである。この式を式(3.2.25)に代入するとマッチドフィルタ (s, h) の表式を表す。つまり、マッチドフィルタが信号雑音比を最大化する最適なフィルタであるという結論を得たことになる。なお、信号雑音比は式(3.2.29)、式(3.2.30)、式(3.2.31)とから

$$\frac{S}{N} = (h, h)^{1/2} \quad (3.2.32)$$

となり、簡単な式で表現することができる。

このように、マッチドフィルタと呼ばれるフィルタを検出器から得られたデータに対して掛け、埋もれている重力波信号を掘り出していくことが重力波のデータ解析である。

第4章

畳み込みニューラルネットワーク

前章では、重力波のデータ解析に現在使用されている手法マッチドフィルタリングについて記述した。本章では、マッチドフィルタリングに代わりうる解析手法である畳み込みニューラルネットワークについて記述する。

畳み込みニューラルネットワークについて記述するにあたって、ニューラルネットワークの原理、及びその根幹のアルゴリズムであるパーセプトロンの概念について説明する。

4.1 パーセプトロン

パーセプトロン（Perceptron）とは、1957年にRosenblattによって考案されたアルゴリズムであり、ニューラルネットワークをはじめとするディープラーニングの起源となるものである。そのため、パーセプトロンの仕組みを学ぶことは、後ほど説明するニューラルネットワークを理解する上で必要不可欠である。本節では、パーセプトロンの概念について記述する。

4.1.1 単純パーセプトロン

単純パーセプトロンとは、複数の信号を入力として受け取り、一つの信号（信号を流す/流さないの二値）を出力するアルゴリズムである。その一例（2入力の単純パーセプトロン）を図4.1に示す。図4.1におけ

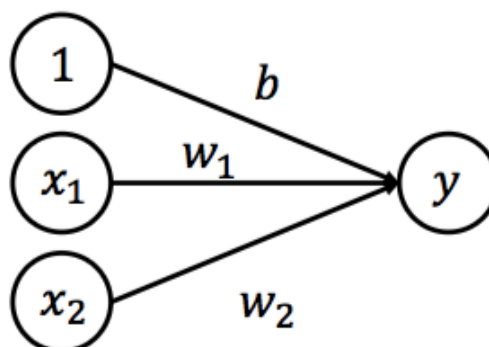


図4.1 単純パーセプトロン（2入力）

表4.1 AND回路の真理値表

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

る x_1, x_2 は入力信号、 w_1, w_2 はウェイト、 b はバイアスである。また、○はニューロンやノードと呼ばれる。この単純パーセプトロンから出力される信号 y は

$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases} \quad (4.1.1)$$

である。つまり、各々の入力信号に対してウェイトをかけ、それらとバイアスを足し合わせたものが0を超えていれば信号を流し、そうでなければ流さないという仕組みである。これを n 個の入力を持つ単純パーセプトロンの場合に拡張すると、

$$y = \begin{cases} 0 & (b + \sum_k^n w_k x_k \leq 0) \\ 1 & (b + \sum_k^n w_k x_k > 0) \end{cases} \quad (4.1.2)$$

となる。なお、パーセプトロンにおける「信号を流す」ことを「発火する」と表現することが多い。

単純パーセプトロンを用いて、簡単な分類を行うことができる。その一例として、AND回路を考える。

AND回路の真理値表は表4.1の通りである。この表4.1における x_1, x_2, y をそれぞれ図4.1におけるそれと対応させる。ここで、AND回路を実現するために、ウェイト w_1, w_2 及びバイアス b を手動で設定する。AND回路を実現するためのパラメータ群（ウェイトとバイアス）には無限個の解があり、その一つは $(w_1, w_2, b) = (0.5, 0.5, -0.7)$ である。このように単純パーセプトロンのパラメータを設定しておくことで、表4.1の真理値表に従うAND回路を実現することができる。

ここで、上述の単純パーセプトロンによるAND回路の実現を可視化すると、図4.2のようになる。図4.2において、赤い点が0として出力すべき入力群（ x_1 と x_2 ）であり、青い点が1として出力すべき入力群を表している。単純パーセプトロンによってなされた処理は、「線形分離によって入力を二値（赤い点か青い点か）分類する」ことである。具体的には、ウェイトとバイアスの値によって決定された傾きと切片を持つ線形な境界線を判断材料に、0或いは1を出力をするということである。

このように、単純パーセプトロンは線形分離による二値分類を行うことができる。そのため、OR回路やNAND回路も同様に、パラメータ群を設定することで実現することができる。

一方、XOR回路は単純パーセプトロンでは実現不可能である。XOR回路の真理値表は表4.2の通りである。これをAND回路の時と同様に可視化すると図4.3の通りである。図4.3を見るとわかる通り、赤い点と青い点を線形な境界線によって分離することができない。つまり、XOR回路を実現するような単純パーセプトロンにおけるパラメータ群が存在しないということである。

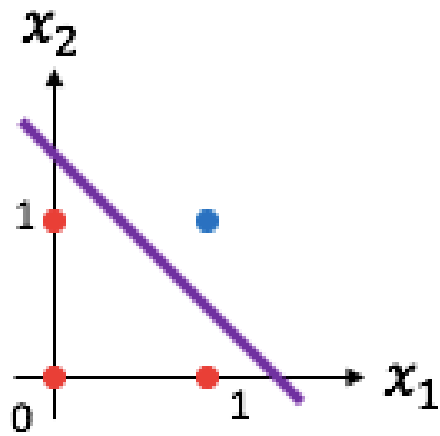


図4.2 単純パーセプトロンによるAND回路を可視化したもの

表4.2 XOR回路の真理値表

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

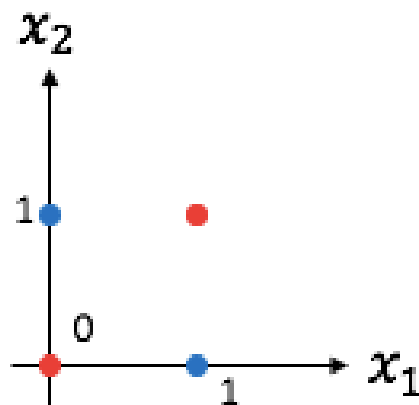


図4.3 XOR回路を可視化したもの

このように単純パーセプトロンでは線形分離によって分類可能な二値分類問題しか解くことができない。

4.1.2 多層パーセプトロン

多層パーセプトロンとは、単純パーセプトロンを複数個組み合わせたパーセプトロンである。多層パーセプトロンは非線形分離が可能である。以下本項では、多層パーセプトロンについて説明する。

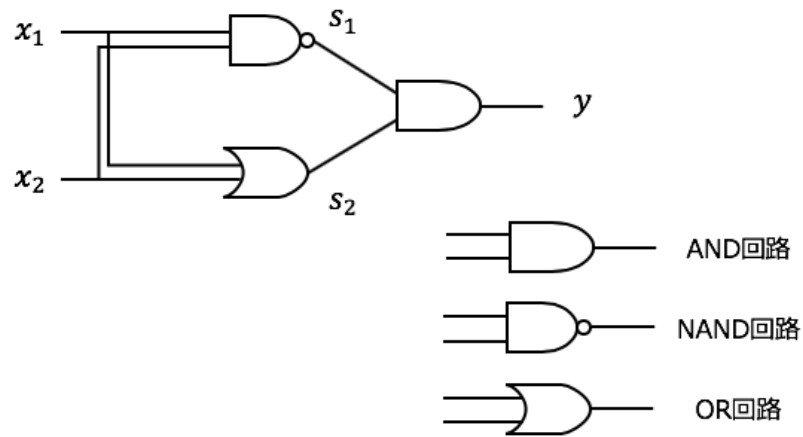


図4.4 XOR回路の実現

表4.3 XOR回路の真理値表

x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

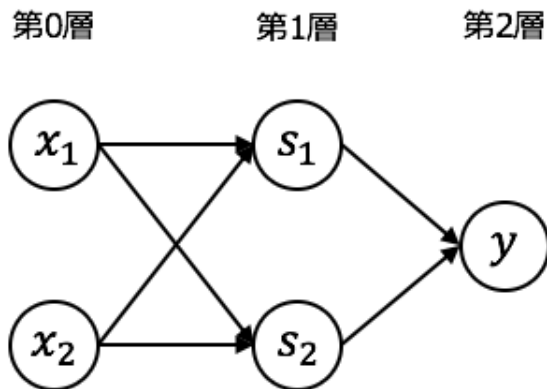


図4.5 XOR回路のパーセプトロンによる表記

前項と同様に、XOR回路を題材に考える。ところで、XOR回路はAND回路、OR回路、NAND回路を組み合わせることで実現が可能である（図4.4）。真理値表は表4.3のようになる。ここで、AND回路、OR回路、NAND回路はそれぞれ単純パーセプトロンで実現することができる。つまり、単純パーセプトロンを組み合わせることで、XOR回路をパーセプトロンで実現できるということである。XOR回路のパーセプトロンによる表記は図4.5のようになる。

パーセプトロンの表記に基づくと、第0層（入力層）に入った入力値は、NAND回路とOR回路を実現す

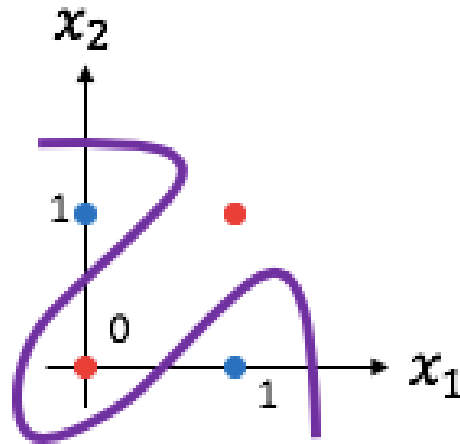


図4.6 パーセプトロンによるXOR回路の実現を可視化したもの

るウェイトがかけられバイアスが付加される。そして第1層へと伝播する。さらに第1層から第2層にかけてAND回路を実現するウェイトがかけられバイアスが付加される。その値が出力値として第2層から出力される。パーセプトロンによるXOR回路ではこのような処理がなされる。

前項におけるAND回路の時と同様に、パーセプトロンによるXOR回路の実現を可視化すると図4.6のようになる。つまり、パーセプトロンによるXOR回路の実現は、非線形分離によって2値分類問題を解くことができたということである。

上記の例のように、入力層以外に $n(\geq 2)$ 層のニューロンを持つパーセプトロンのことを n 層の多層パーセプトロンと呼ぶ。

このように、単純パーセプトロンをつなげて多層パーセプトロンを実現することで、非線形分離による分類問題を解くことができるようになることがわかった。言い換えると、層を深くすることでパーセプトロンはより柔軟な表現をすることが可能になったということである。

4.2 ニューラルネットワークとその学習

前節において、パーセプトロンについて記述した。パーセプトロンは層を深くすることで、複雑な関数を表現できるだけの可能性を秘めている。一方で、ウェイトやバイアスを人工的に設定しなければならない、つまり期待する入力と出力を満たすようにパラメータを人工的に設定しなければならない。この作業は、前節で触れた例のような簡単な問題の場合には可能だが、実際に我々が解きたいような複雑な問題においては不可能である。

本節で記述するニューラルネットワークはこの問題を解決することができる。具体的には、ニューラルネットワークの性質の一つとして、適切なウェイトやバイアスの値をデータから自動で学習することが挙げられる。本節では、ニューラルネットワークの構造、学習方法について記述する。

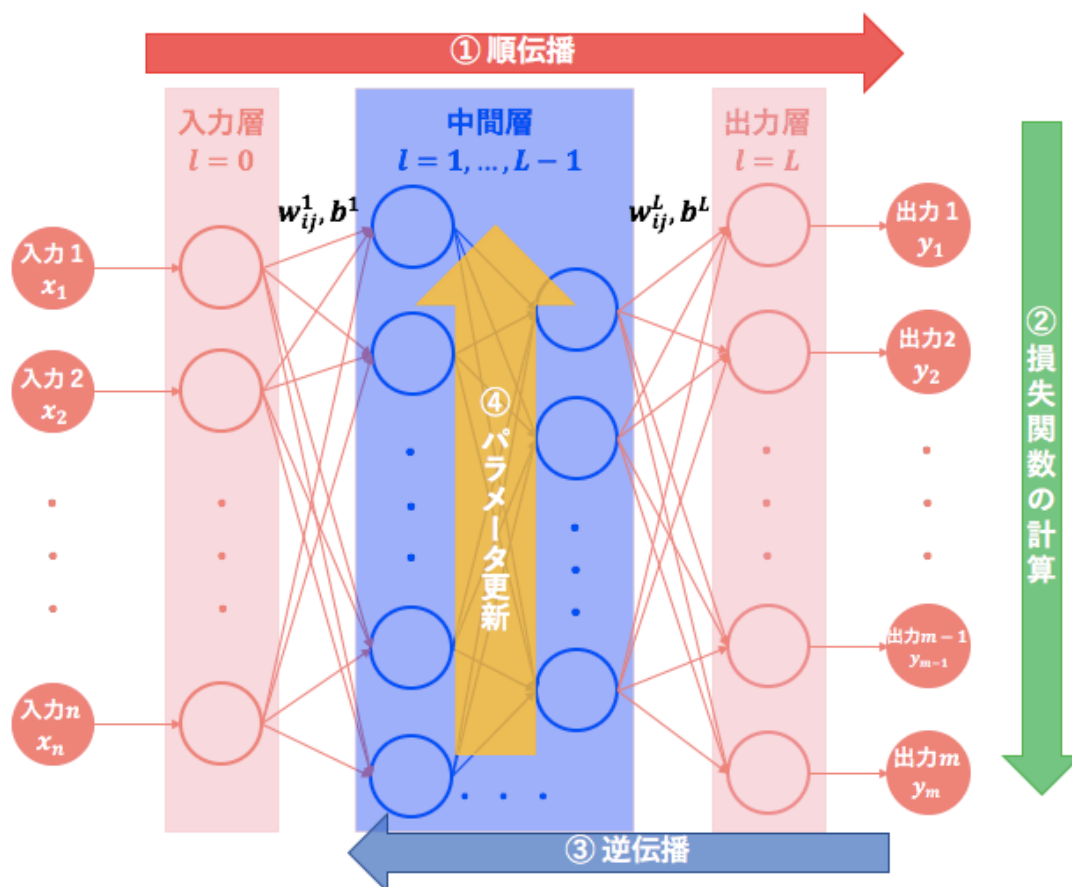


図4.7 ニューラルネットワークの学習プロセス

図4.7は、ニューラルネットワークの学習プロセスを可視化したものであり、この図に則って順に説明していく。

ニューラルネットワークに入力された入力信号はまず①順伝播によって出力値を算出する。次に、その出力値と正解データ（ラベル）とから②損失関数（エラー）を計算する。次に、損失関数の各パラメータ（ウェイト、バイアス）に対する勾配（偏微分値）を③逆伝播によって伝播させていく、最後に、伝播された勾配をもとに各パラメータの更新を行う（④）。このループを複数回繰り返すことで、各パラメータが最適化され、ネットワークの学習が完了する。

4.2.1 トレインデータとテストデータ

上記各プロセスについて記述する前に、機械学習において重要なトレインデータとテストデータについて記述する。

ニューラルネットワークなどの機械学習では、トレインデータとテストデータの2つのデータを用いて、モデルの学習・評価、実験を行うのが一般的である。

トレインデータとは、モデルを学習する際に用いられるデータセットである。一方、テストデータとは、

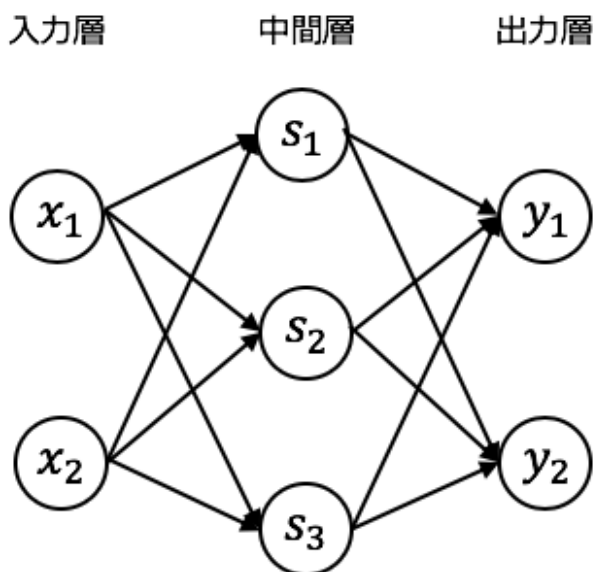


図4.8 ニューラルネットワークの例

モデルの汎化能力を検証する際に用いられるデータセットである。汎化能力とは、トレインデータに含まれないデータに対するモデルの精度（推論能力）のことであり、この汎化能力を獲得することが機械学習の目標である。

一方、特定のデータセットのみに精度が良い状態を過学習という。過学習を避けるために様々な手法が開発されてきた。詳細は後述する。

以下、トレインデータによる学習について記述する。

4.2.2 ニューロンの構造

まず、図4.7における①順伝播について説明する。順伝播は前節で説明したパーセプトロンと同様、各ニューロンにおいて処理を行い、その出力結果を次の層へと伝播させていき、最終的な出力値を算出することを意味する。ニューラルネットワークの例を図4.8に示す。これは2層のニューラルネットワークの例である。はじめの層から順に入力層、中間層（隠れ層）、出力層と呼ぶ。 n 層のニューラルネットワークの場合は、中間層を $n - 1$ 層持つ。ここで、各ニューロンにおける処理を具体的にみる。

各ニューロンでなされる処理はパーセプトロンのそれとほとんど同じである。ここで、式(4.1.2)を次のように書き換える：

$$y = h \left(b + \sum_k^n w_k x_k \right) \quad (4.2.1)$$

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases} \quad (4.2.2)$$

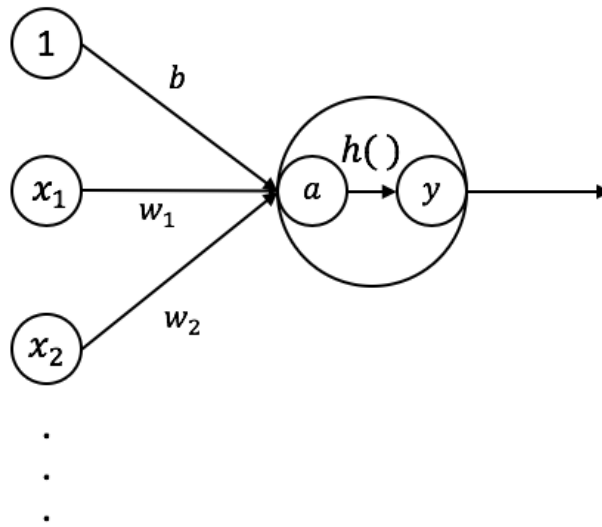


図4.9 各ニューロンにおける処理を明示化したもの

上式が意味することは、各ニューロンに入力された入力値は式(4.2.1)で表されるように総和にバイアスが付加され、その後式(4.2.2)で表されるように $h(x)$ という関数によって変換された値を出力するということである。この式(4.2.1)における処理を新たな層と見なし、Affine層と呼ぶことが多い。また、 $h(x)$ は活性化関数（Activation Function）と呼ばれ、パーセプトロンの場合はステップ関数が活性化関数として用いられている。

上記を踏まえ、各ニューロンにおける処理を明示的に表すと下式、図に表すと図4.9のようになる。

$$a = b + \sum_k^n w_k x_k \quad (4.2.3)$$

$$y = h(a) \quad (4.2.4)$$

上述の通り、パーセプトロンでは活性化関数としてステップ関数が用いられている。しかし、ニューラルネットワークにおいては、より柔軟な表現をするために、活性化関数として連続関数を用いている。次項では、ニューラルネットワークにおいて用いられている活性化関数について記述する。

4.2.3 活性化関数

活性化関数は入力信号の総和がどのように活性化（発火）ということを決める役割を担っている。活性化関数として用いられる関数は全て非線形関数である。なぜなら、線形関数を活性化関数として用いると、層を深くする意義がなくなってしまうからである。本項では、実際にニューラルネットワークに用いられている活性化関数について記述する。

シグモイド関数

シグモイド関数は以下の式で表される関数である：

$$h(x) = \frac{1}{1 + \exp(-x)} \quad (4.2.5)$$

ステップ関数と比較すると、連続関数になっており、より滑らかな出力をすることができる。ニューラルネットワークが誕生してから長い間、シグモイド関数はネットワークを数学的に扱うことが容易にするため、活性化関数として主に用いられてきた。

また、後述する出力層での関数として、2値分類問題においてこのシグモイド関数が使用されることが多い。

ReLU関数

ReLU(Rectified Linear Unit)関数は以下の式で表される関数である：

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases} \quad (4.2.6)$$

ReLU関数は上式で表されるように、入力が0を超えていればそのまま出力し、0以下なら0を出力する関数である。最近のニューラルネットワークでは、経験則的にReLUの方が他の活性化関数よりも精度の良いネットワークを構成できるということがわかっているため、主にReLU関数が用いられている。

本研究においても、このReLU関数を活性化関数として用いた。

前項および本項の通り、各ニューロンにおける入力値はAffine層を通ったのち活性化関数にかけられ、次の層へと伝播される。この処理を繰り返し、最終的に出力層へと伝播されていく。図4.10はここで紹介した活性化関数のグラフである。

4.2.4 出力層

入力信号は順伝播によって値を変えながら、最終的に出力層へとたどり着く。出力層における各ニューロンの構造は中間層と同様だが、用いる活性化関数が異なる。本項では、出力層で用いられる活性化関数について記述する。

ニューラルネットワークが解く問題には主に2つあり、1つが回帰問題であり、他方が分類問題である。ニューラルネットワークにおいては、解きたい問題の種類によって出力層の活性化関数が異なる。一般に、回帰問題においては活性化関数として恒等関数が、分類問題においてはソフトマックス関数が用いられる。以下で、それぞれについて記述する。

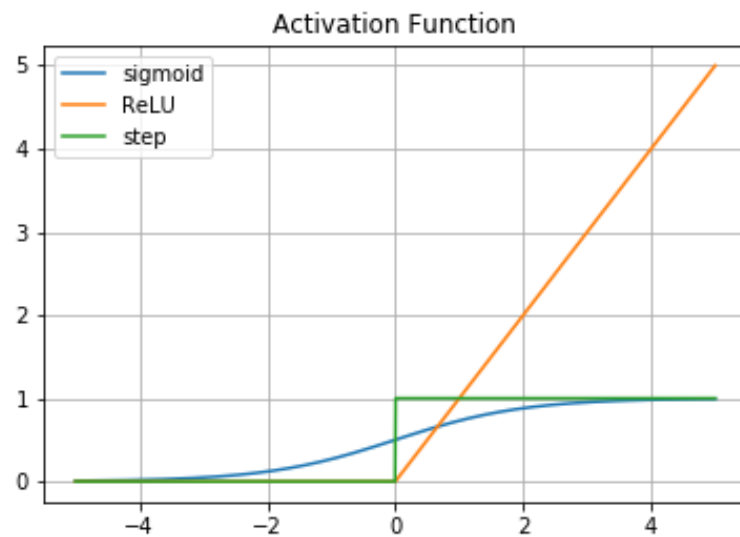


図4.10 様々な活性化関数

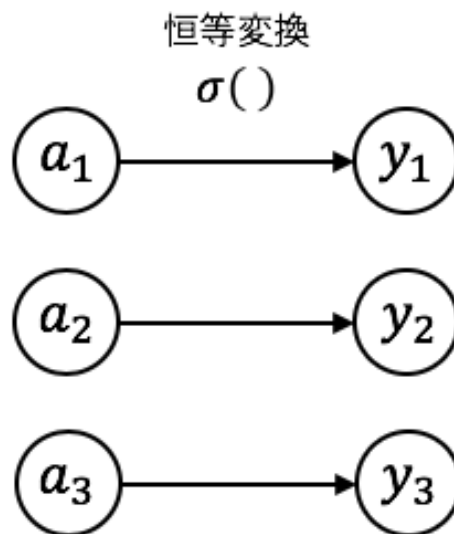


図4.11 恒等関数

恒等関数

回帰問題に用いられる恒等関数は、出力層における入力をそのまま出力する関数である。図4.11は恒等関数による処理を可視化したものである。構造としては中間層で用いた活性化関数と同様で、入力1つに対して変換を行う。

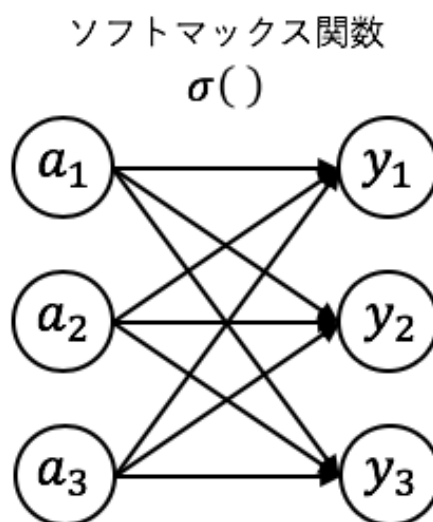


図4.12 ソフトマックス関数

ソフトマックス関数

分類問題に用いられるソフトマックス関数は、入力 $a_i (i = 1, \dots, n)$ に対して、以下のような出力値 $y_i (i = 1, \dots, n)$ を出力する：

$$y_i = \frac{\exp(a_i)}{\sum_k^n \exp(a_k)} \quad (4.2.7)$$

しかし、ソフトマックス関数を上式のまま実装すると、指数関数の計算があるためオーバーフローを起こしてしまう可能性がある。そのため、実装をするにあたって上式を以下のように変形する：

$$\begin{aligned} y_i &= \frac{\exp(a_i)}{\sum_k^n \exp(a_k)} = \frac{C \exp(a_i)}{C \sum_k^n \exp(a_k)} \\ &= \frac{\exp(a_i + \log C)}{\sum_k^n \exp(a_k + \log C)} \\ &= \frac{\exp(a_i + C')}{\sum_k^n \exp(a_k + C')} \end{aligned} \quad (4.2.8)$$

ここで、 C または C' はオーバーフロー対策のための定数である。主にこの定数として、入力信号の最大値 $\text{maximum}(a_i)$ が用いられる。

なお、ソフトマックス関数による処理を可視化すると、図4.12のようになる。図のように、ソフトマックス関数においては、すべての入力信号から矢印による結びつきがあることが特徴の1つである。

上式を見ると、ソフトマックス関数の出力は0から1.0の間の実数となり、また出力の総和は1になる。これはソフトマックス関数の出力が確率として解釈できることを意味する。

このように、出力層における入力信号はソフトマックス関数により、それぞれ対応ラベルの確率に変換されることがわかった。この確率をもとに、分類問題の推論を行うというわけである。

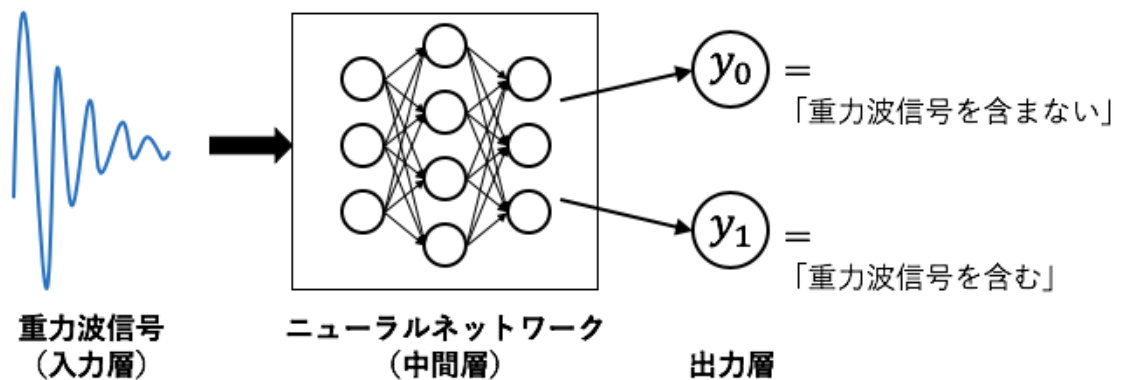


図4.13 出力層

出力層のニューロン数

最後に、出力層のニューロン数の設定に関して説明する。

出力層のニューロン数は、分類したいクラスの数に設定する。つまり、 n クラス分類問題の場合は、出力層のニューロン数は n と設定する。本研究では、重力波検出器から得られたデータに重力波信号が「含まれていないか」、「含まれているか」の2クラス分類問題を解くため、出力層のニューロン数は2と設定している。つまり、本研究におけるニューラルネットワークの概観は図4.13のようになっている。

以上本項まで、①順伝播が完了する。

4.2.5 ミニバッチ

②損失関数の計算について記述する前に、ミニバッチについて記述する。

ニューラルネットワークを学習させるにあたって、データを1つ1つ学習させるわけではない。実際にはミニバッチと呼ばれる、トレインデータをいくつかまとめて束として学習させる。この束をミニバッチという。また、このミニバッチのサイズ、つまりいくつかのデータをまとめて束にするかという値のことをバッチサイズという。

数値計算を扱うライブラリの多くは、大きな配列の計算を効率よく処理できるよう最適化がなされている。そのため、本研究のようにデータ転送がボトルネックとなる場合には、ミニバッチによる学習を行うことでバス帯域の負荷を軽減させることができ、処理時間を短縮することができる。

4.2.6 損失関数の計算

本項では、②損失関数の計算について記述する。

前述の通り、順伝播によって得られた出力値とトレインデータの正解ラベルとのエラーを元に、各パラメータの更新を行う。このエラーを計算する関数のことを損失関数(Loss Function)という。この損失関数として、任意の関数を用いることができるが、一般には下記2つの関数が用いられる。

二乗和誤差

二乗和誤差(Mean Squared Error)は以下の式で定義される関数である：

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2 \quad (4.2.9)$$

ここで、 y_k はニューラルネットワークの出力、 t_k はトレインデータの正解ラベルであり、正解のとき1、そうでないとき0である。である。また、 k はニューラルネットワークの出力層のニューロン数、つまりデータの次元である。

クロスエントロピー誤差

近年主流となっている損失関数がクロスエントロピー誤差(Cross Entropy Error)である。クロスエントロピー誤差は以下の式で定義される：

$$E = - \sum_k t_k \log y_k \quad (4.2.10)$$

ここで、二乗和誤差のときと同様、 y_k はニューラルネットワークの出力、 t_k はトレインデータの正解ラベルである。

本研究において、損失関数としてこのクロスエントロピー誤差を用いた。

ミニバッチ版クロスエントロピー誤差

実際の学習では、上記のようにミニバッチを用いる。そのため、損失関数の形をミニバッチに対応した形に変形する必要がある。

ミニバッチ学習におけるクロスエントロピー誤差の形は

$$E_N = -\frac{1}{N} \sum_n \sum_k t_{nk} \log y_{nk} \quad (4.2.11)$$

である。なお、添字 n, k はそれぞれミニバッチにおける n 個目のデータ、出力値における k 番目の値である。また、 N はバッチサイズであり、 n ごとに計算されたクロスエントロピー誤差の総和を、バッチサイズで割り正規化を行っている。これにより、バッチサイズの大きさにかかわらず1個のデータあたりの損失関数を計算することができる。

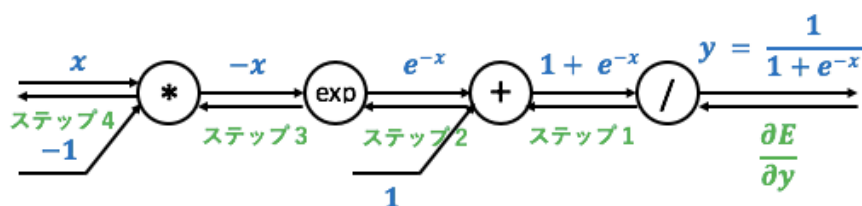


図4.14 シグモイドレイヤの計算グラフ（順伝播）

以上で、②損失関数の計算が完了する。

4.2.7 誤差逆伝播法

前項までに、②損失関数の計算が完了した。本項では、③逆伝播を実現する手法、誤差逆伝播法（Back Propagation）について記述する。

上述の通り、パラメータを更新するためには、各パラメータに対する損失関数の勾配を計算する必要がある。勾配計算の実装には数値微分が用いられることがあるが、実装が容易な一方で計算コストが非常に高い。そのため、パラメータ数が膨大なニューラルネットワークにおける勾配計算では、数値微分に代わって誤差逆伝播法が用いられる。

計算グラフ

誤差逆伝播法を直感的に理解しやすくするために、計算グラフを用いた記述を行う。

計算グラフとは、計算の過程をグラフによって表したものであり、複数のノードのエッジによって表現される。その一例として、入力値をシグモイド関数に通すシグモイドレイヤにおける計算グラフを考える。活性化関数の一種であるシグモイド関数は、式(4.2.5)の通りであり、ここに再掲する：

$$h(x) = \frac{1}{1 + \exp(-x)} \quad (4.2.12)$$

このシグモイドレイヤにおける順伝播の計算グラフは図4.14のようになる。このように、各演算プロセスを明示化することで、次に行う逆伝播を直感的に理解することができる。

誤差逆伝播法

誤差逆伝播法とは、連鎖律（Chain Rule）を用いて、出力層から入力層に向かって勾配を伝播させていく手法である。

連鎖律は、関数が $f(y_1, y_2, \dots, y_K), y_k(w)$ ($k = 1, 2, \dots, K$) で表される多変数関数の場合、

$$\frac{\partial f}{\partial w} = \sum_{k=1}^K \frac{\partial f}{\partial y_k} \frac{\partial y_k}{\partial w} \quad (4.2.13)$$

で表される。

この連鎖律に則り、上記シグモイドレイヤにおける逆伝播を計算する。

まずはじめに、損失関数を E 、シグモイドレイヤの出力を y とすると、シグモイドレイヤには $\frac{\partial E}{\partial y}$ という値が逆伝播で流れ込む。「/」ノードは入力 x に対して出力 $y = \frac{1}{x}$ を計算するノードである。この微分は

$$\frac{\partial y}{\partial x} = -\frac{1}{x^2} = -y^2 \quad (4.2.14)$$

であるから、連鎖律より「/」ノードから「+」ノードへ $\frac{\partial E}{\partial y}(-y^2)$ が逆伝播する。(ステップ1)

「+」ノードから「exp」ノードへは、単なる可算であるからそのまま $\frac{\partial E}{\partial y}(-y^2)$ が逆伝播する。(ステップ2)

「exp」ノードは、入力 x に対して出力 $y = \exp(x)$ を計算するノードであるから、その微分は

$$\frac{\partial y}{\partial x} = \exp(x) \quad (4.2.15)$$

である。今、入力は $-x$ であるから、「exp」ノードから「*」ノードへ $\frac{\partial E}{\partial y}(-y^2) \exp(-x)$ が逆伝播する。(ステップ3)

最後に、「*」ノードでは、入力 x に対して出力 $y = -x$ を計算するノードであるから、その微分は-1であり、「*」ノードから入力層方向へ $\frac{\partial E}{\partial y}(-y^2) \exp(-x) * (-1)$ が逆伝播する。(ステップ4)

ここで、

$$\begin{aligned} \frac{\partial E}{\partial y}(-y^2) \exp(-x) * (-1) &= \frac{\partial E}{\partial y} y^2 \exp(-x) \\ &= \frac{\partial E}{\partial y} \frac{1}{(1 + \exp(-x))^2} \exp(-x) \\ &= \frac{\partial E}{\partial y} \frac{1}{1 + \exp(-x)} \frac{\exp(-x)}{1 + \exp(-x)} \\ &= \frac{\partial E}{\partial y} y(1 - y) \end{aligned} \quad (4.2.16)$$

と変形できる。シグモイドレイヤにおける逆伝播では、シグモイドレイヤにおける出力のみを用いて計算することができる。つまり、順伝播によって計算されたシグモイドレイヤの出力値を保持しておく必要がある。

このように、連鎖律を用いた誤差逆伝播法では、単なる乗算のみにより勾配を計算することができる。そのため、数値微分と比較すると計算コストを非常に下げることが可能になる。以上をまとめると、シグモイドレイヤにおける逆伝播の様子は図4.15のようになる。

上記はシグモイドレイヤにおける逆伝播であるが、ニューラルネットワークにおける他の全種類の層に関して同様の計算により逆伝播を行う。この逆伝播を出力層から入力層にかけて順々に行い、全てのパラメータに対する損失関数の勾配を計算する。こうして、誤差逆伝播法による③逆伝播が完了する。

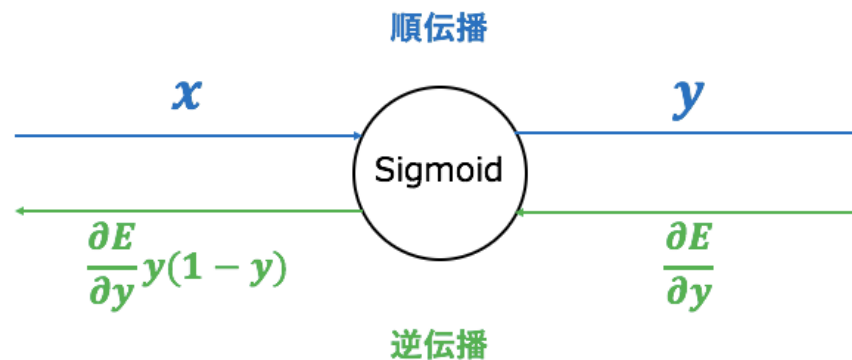


図4.15 シグモイドレイヤの計算グラフ（順伝播・逆伝播）

4.2.8 パラメータ更新

前項までに、③逆伝播による、全パラメータに対する損失関数の勾配を計算することができた。本項では、損失関数の勾配を用いた④パラメータ更新について記述する。

前述の通り、ニューラルネットワークにおける学習とは「損失関数の値を最小化するパラメータの最適化問題」である。最適化に際して、損失関数の勾配をもとにパラメータ更新を行う手法の代表的なものが確率的勾配降下法（Stochastic Gradient Descent : SGD）である。はじめに、SGDのアルゴリズムについて記述する。

確率的勾配降下法（SGD）

SGDはニューラルネットワークの学習によく用いられる最適化アルゴリズムである。SGDにおけるパラメータ更新は以下の式で表される：

$$\vec{W} \leftarrow \vec{W} - \eta \frac{\partial E}{\partial \vec{W}} \quad (4.2.17)$$

ここで、 $\vec{W}$ はウェイトやバイアスなどのパラメータであり、パラメータ空間を成すベクトルである。また、 η は学習率と呼ばれる値であり、パラメータ更新の度合いを表すハイパーパラメータである。一般的な機械学習フレームワークではデフォルトとして0.01が設定されている。式中の $\leftarrow$ は、右辺の値で左辺の値を更新するということを表している。

図4.16はSGDにおけるパラメータ更新のイメージである。逆伝播によって得られた損失関数の勾配を元に、損失関数の値が減少する方向にパラメータ更新を行う。この処理を1つのミニバッチごとに行い、パラメータを最適化している。

このようにして、損失関数の勾配を元にしたSGDにより、パラメータは最適化され、ネットワークは学習していく。

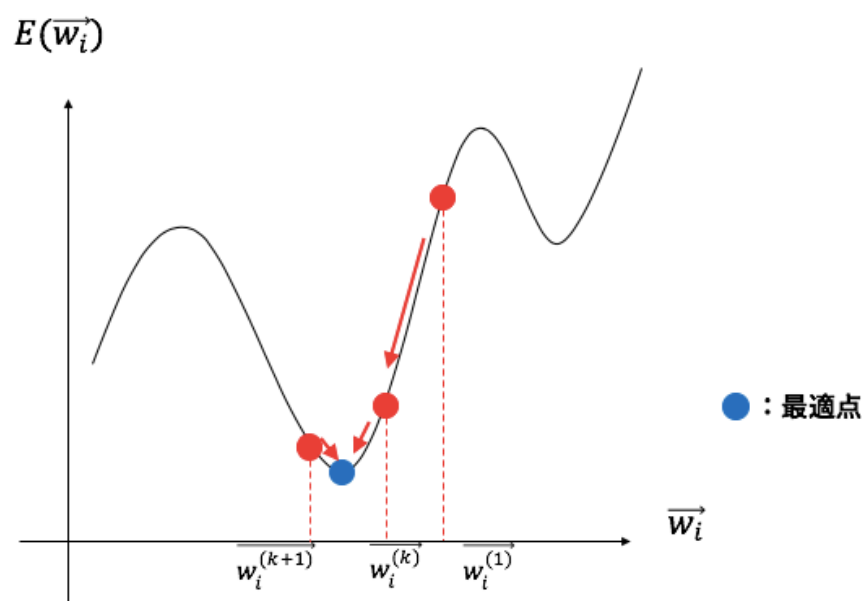


図4.16 SGDのイメージ図

しかし、SGDには欠点がある。具体的には、SGDは損失関数の形状が等方的でないとパラメータの探索が非効率になってしまうという点である。この欠点を解決するために、現在ではSGDに代わる様々なパラメータ更新手法が開発されている。代表的なものに関して、以下で記述する。

Momentum SGD

Momentum SGDは以下の式で表されるパラメータ更新手法である：

$$\vec{v} \leftarrow \alpha \vec{v} - \eta \frac{\partial E}{\partial \vec{W}} \quad (4.2.18)$$

$$\vec{W} \leftarrow \vec{W} + \vec{v} \quad (4.2.19)$$

SGDの時と同様、 $\vec{W}$ はウェイトやバイアスなどのパラメータ、 η は学習率である。また、式中の $\vec{v}$ は物理の「速度」に相当する量である。この手法は、物理現象における摩擦減衰などの外力を模したパラメータ更新手法である。減衰の度合いを決定するパラメータが α であり、ハイパーパラメータである。一般的な機械学習フレームワークではデフォルトとして0.9が設定されている。

SGDと比較すると、勾配の大きい方向では摩擦のような外力が加わり、勾配の小さい方向では加速的に外力が加わるため、効率的なパラメータ探索が行えるようになる。

AdaGrad

ニューラルネットワークの学習では、学習率の値が重要である。というのも、学習率が小さすぎると学習に時間がかかりすぎてしまい、逆に大きすぎると勾配が発散して正しい学習がなされない。

この学習率に関する有効なテクニックとして学習率の減衰という方法がある。具体的には、学習が進む

につれて学習率を小さくする、つまり学習率の自動更新である。

AdaGradでは、パラメータの要素ごとに適応的に学習率を調整しながら学習を行うことができる手法である。AdaGradにおけるパラメータ更新は以下の式で表される：

$$\vec{h} \leftarrow \vec{h} + \frac{\partial E}{\partial \vec{W}} \odot \frac{\partial E}{\partial \vec{W}} \quad (4.2.20)$$

$$\vec{W} \leftarrow \vec{W} - \eta \frac{1}{\sqrt{\vec{h}}} \frac{\partial E}{\partial \vec{W}} \quad (4.2.21)$$

SGDの時と同様、 $\vec{W}$ はウェイトやバイアスなどのパラメータ、 η は学習率である。また、 $\odot$ は行列の要素ごとの乗算を意味する。 $\vec{h}$ はこれまでに経験した勾配の値を二乗和として保持し、パラメータ更新の際に学習スケールの調整として用いられる。つまり、更新度合いの大きいパラメータにおいては、学習率が小さくなるように調整される。このようにAdaGradでは、更新度合いの大きいパラメータの学習率の減衰を、パラメータの要素ごとに行うことができる。

Adam

AdamはMomentum SGDとAdaGradを融合したようなパラメータ更新手法である。つまり、Momentum SGDにおける「外力の作用」と、AdaGradにおける「パラメータの要素ごとに適応的に更新ステップを調整すること」の双方を搭載した手法である。式で表すと、

$$\vec{m} \leftarrow \beta_1 \vec{m} + (1 - \beta_1) \frac{\partial E}{\partial \vec{W}} \quad (4.2.22)$$

$$\vec{v} \leftarrow \beta_2 \vec{v} + (1 - \beta_2) \left(\frac{\partial E}{\partial \vec{W}} \right)^2 \quad (4.2.23)$$

$$\hat{m} = \frac{\vec{m}}{1 - \beta_1} \quad (4.2.24)$$

$$\hat{v} = \frac{\vec{v}}{1 - \beta_2} \quad (4.2.25)$$

$$\vec{W} \leftarrow \vec{W} - \alpha \frac{\hat{m}}{\sqrt{\hat{m}} + \epsilon} \quad (4.2.26)$$

である。ここで、 α, β_1, β_2 はハイパーパラメータであり、一般的な機械学習フレームワークではデフォルトとしてそれぞれ0.001, 0.9, 0.999が設定されている。また、 ϵ は0で除算されてしまうことを防ぐための微小定数であり、デフォルトとして1e-08が設定されている。

このように、様々なパラメータ更新手法が存在する。一見、Adamが最も先進的な手法でより良い精度を出すことができるように思えるが、実際はそうではない。その時々ネットワークやトレインデータなどによって、最も精度の良いパラメータ更新手法は変わってくる。そのため、様々なパラメータ更新手法による学習を試した中で、最も精度が良い手法を選択するのがベターである。

以上のパラメータ更新により、「1 ミニバッチにおける1 ループ分」の学習が完了し、次のミニバッチ

における1ループ（つまり次の順伝播）へと移る。この「1ミニバッチにおける1ループ」のことをイテレーション（Iteration）という。このイテレーションを全てのミニバッチ分行うことで、全トレインデータ分の学習を完了する。この「全トレインデータ分の学習に関する1ループ」をエポック（Epoch）といい、ニューラルネットワークの学習においてはこのエポックを複数回行う。また、ネットワークの評価においては、イテレーションではなくエポック単位での評価を行う。なぜなら、イテレーションはバッチサイズに依存しているため、イテレーションでの評価では事前に準備したデータの量・質を元に正確に評価することはできないためである。

以上までのプロセスで、ネットワークの学習プロセスが終了する。このように、「入力信号が①順伝播され、正解データとの比較により②損失関数が計算される。さらに、全パラメータに関する損失関数の勾配が③逆伝播され、その勾配に基づき④パラメータ更新がなされる。」というプロセスによってニューラルネットワークの学習が行われる。

4.2.9 過学習とBatch Normalization

前項までで、ニューラルネットワークの学習は完了する。しかし、ニューラルネットワークの学習において、過学習（Overfitting）という問題が起きる可能性がある。過学習とは、トレインデータにのみ適応し過ぎてしまい、トレインデータに含まれない他のデータに対して精度が良くない状態をいう。つまり、汎化能力のないネットワークができてしまうということである。過学習が起きてしまう原因として、主に2つのことが考えられる。1つは、「パラメータを大量に持ち、表現力の高いネットワークモデルであること」である。他方は、「トレインデータが少ないこと」である。後者に関しては、本研究ではトレインデータとしてシミュレーションによる擬似重力波信号を用いるため、トレインデータの数を増やすことは容易であるため問題にならない。しかし、前者に関しては、本研究では後述する畳み込みニューラルネットワークを何層にも重ねたモデル（ディープラーニング）を用いるために、問題となる。そのため、過学習を抑制する対策をしなければならない。

機械学習においては汎化能力を高めることが主目的であるため、この過学習を抑制するテクニックが開発されてきた。その手法こそがBatch Normalizationであり、本項ではその記述を行う。

Batch Normalizationは、中間層におけるアクティベーション分布（活性化関数の後の出力データ分布）を調整し、適度な広がりを持たせる手法である。つまり、ミニバッチに分けたことで生じてしまう特徴量（全トレインデータ・テストデータで見たときに特徴量とはならないもの）による過学習を抑制するために、

ミニバッチの正規化を行うというものである。数式で表現すると、次のようになる：

$$\mu_B = \frac{1}{n} \sum_{i=1}^n x_i \quad (4.2.27)$$

$$\sigma_B^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_B)^2 \quad (4.2.28)$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (4.2.29)$$

ここでは1行目から2行目にかけて、ミニバッチとして $B = \{x_1, x_2, \dots, x_n\}$ という n 個の入力データの集合に対して、平均 μ_B 、分散 σ_B^2 を求めている。その後3行目において、入力データの平均が0、分散が1になるように正規化している。また、 ϵ は0で除算されてしまうことを防ぐための微小係数である。

このように、トレインデータをミニバッチに分割することによる「偏り」をBatch Normalizationによって無くしている。

通常、Batch Normalizationは活性化関数の前に行うことが多い。しかし、Batch Normalizationを活性化関数の後に行うこともあり、どちらが良いかの議論がなされている。

一方、Batch Normalizationは過学習を抑制するだけでなく、次の利点を持っている。1つ目は、「学習をより速く進行させることができる」ことであり、2つ目は、「初期値に対する依存性が下がる」ことであり。後者に関して、ネットワークを定義する際に全パラメータの初期値が設定されるが、この初期値によってアクティベーション分布の偏りが生じてしまうことがある。しかし、この問題もBatch Normalizationによって解決されるため、初期値の設定に関して神経質になる必要がなくなる。

以上が、Batch Normalizationに関する記述であるが、本研究ではBatch Normalizationを活性化関数の前に行っている。

4.3 畳み込みニューラルネットワーク

前節までに、ニューラルネットワークの構造、およびその学習について記述した。本研究では、ニューラルネットワークの派生版である畳み込みニューラルネットワーク (Convolutional Neural Network : CNN) を用いる。本節では、畳み込みニューラルネットワークの構造について記述する。

畳み込みニューラルネットワークは、画像認識や音声認識などの広い分野で用いられているニューラルネットワークの一種である。実際に、Facebookではタグ付けのための顔認証として、Googleでは写真検索や音声認証として実用化もなされている。また、重力波解析の分野においても、参考論文[1]において畳み込みニューラルネットワークの有用性について議論されている。

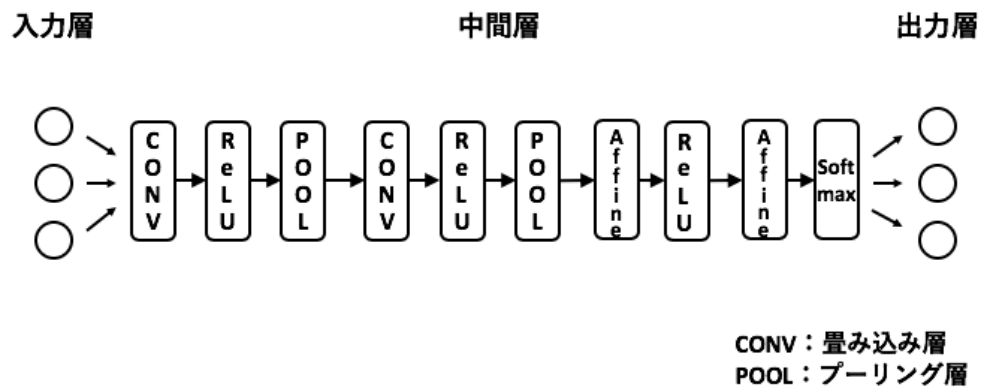


図4.17 畳み込みニューラルネットワークの構造例

4.3.1 畳み込みニューラルネットワークの構造

本項では、畳み込みニューラルネットワークの構造について、前述のニューラルネットワークとの相違点を交えながら記述する。

図4.17は畳み込みニューラルネットワークの構造例である。畳み込みニューラルネットワークにおいては、前節で記述した「Affine層」、「活性化関数層」、「出力層」に加えて、「畳み込み層」と「プーリング層」が登場する。これらの層については、次項にて記述する。

前節までに取り扱った層では、隣接する層のすべてのニューロン間で結合があった。これを全結合 (fully-connected) という。全結合のニューラルネットワークでは、「Affine層」+「活性化関数層」を基本単位とした全結合構造をしていたのに対し、畳み込みニューラルネットワークでは「畳み込み層」+「活性化関数層」+「プーリング層」を基本単位とした構造をしている（プーリング層は省略可能）。

全結合のニューラルネットワークには、入力データの形状が無視されてしまうという問題点がある。例えば画像認識の場合、入力データとして用いられる画像データは縦・横・チャンネル方向の3次元データである。しかし、全結合のニューラルネットワークにおいては、この3次元データを1次元にする必要があるため、画像が持つ空間的な情報を失ってしまうことになる。例えば、近接するピクセルは似たような値であったり、逆に距離の離れたピクセル同士は相関が少ないという「本質的なパターン」に関する情報を失ってしまう。重力波信号の場合にも、重力波源のパラメータに依存した関数の形で記述されるため、近接する入力信号同士は相関を持つはずである。そのため本研究においても、全結合のニューラルネットワークよりも畳み込みニューラルネットワークの方が有用であると考えられる。

以上のように、畳み込みニューラルネットワークでは全結合のニューラルネットワークに加えて、「畳み込み層」と「プーリング層」という新たな2つの層を擁する。次項にて、それぞれの層で行われる処理につ

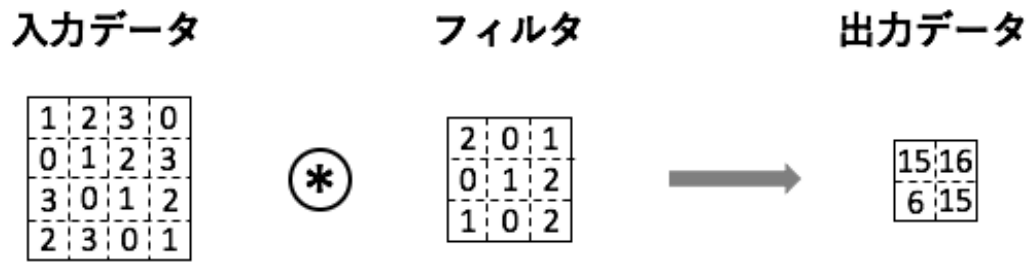


図4.18 畳み込み演算の例

いて記述する。

4.3.2 畳み込み層

畳み込み層で行われる処理は畳み込み演算である。畳み込み演算は、入力データに対してフィルタ（カーネルとも呼ぶ）を適用し、入力データとフィルタとの畳み込みを計算することを指す。

畳み込み演算

図4.18は畳み込み演算の例である。データとフィルタの形状を(縦, 横)と表記すると、この場合入力サイズとフィルタサイズはそれぞれ(4, 4)、(3, 3)である。畳み込み演算は、入力データに対して、フィルタのウィンドウを一定間隔でスライドしながら適用させる。ウィンドウとは図4.19における灰色で示された3 × 3の部分である。図4.19で示すように、それぞれのウィンドウ領域において、入力データの要素と対応するフィルタの要素を乗算し、その和を求める。そして、その結果を対応する出力の場所に格納する。このプロセスを各ウィンドウに対して行うことで、畳み込み演算の出力を得ることができる。

ここで、畳み込みニューラルネットワークにおけるフィルタの要素は、全結合のニューラルネットワークにおけるウェイトに相当する。また、全結合のニューラルネットワークの時と同様に、畳み込み層においてもバイアスが存在する。畳み込み層におけるバイアス付加はフィルタによる畳み込み演算ののちに行われ、図4.20のようになる。図4.20で示すように、畳み込み層におけるバイアスは1つ（1 × 1）のみの値を持つ。

以上が、畳み込み層における処理である。

この例は、シンプルな畳み込み演算の例であるが、実際には畳み込み層において主なハイパーパラメータが4つ存在する。以下で、それぞれについて記述する。

フィルタサイズ

1つ目がフィルタサイズである。図4.18における例ではフィルタサイズは3であったが、実際にはこの数を変えることができる。フィルタサイズが n のとき、フィルタの形状は (n, n) である。

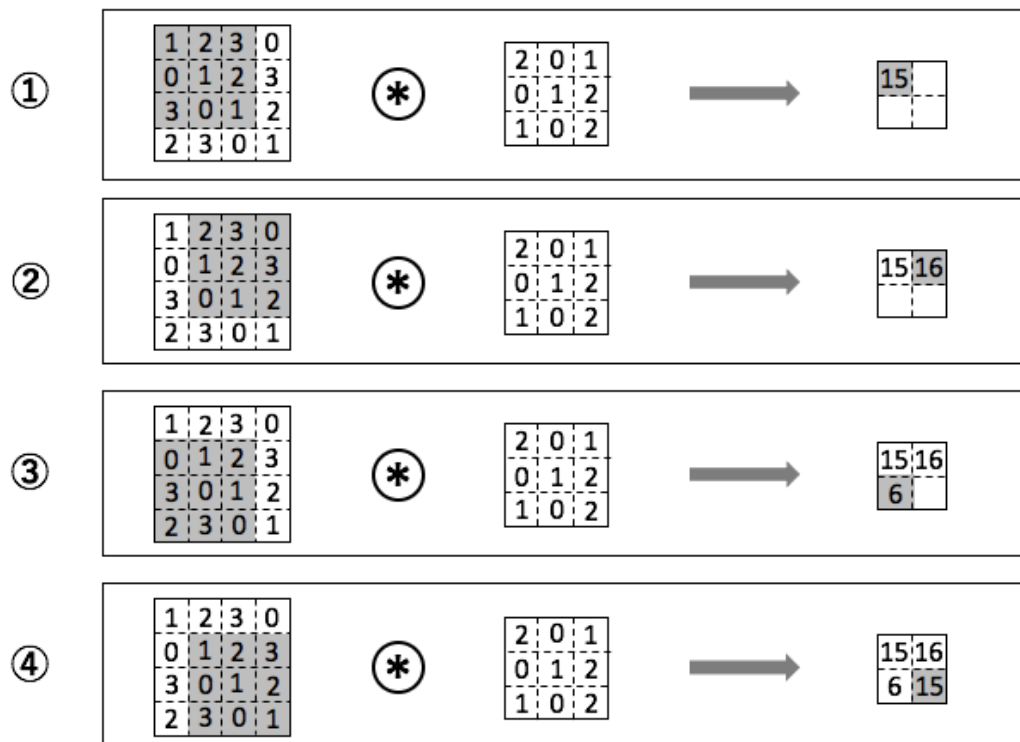


図4.19 畳み込み演算のプロセス

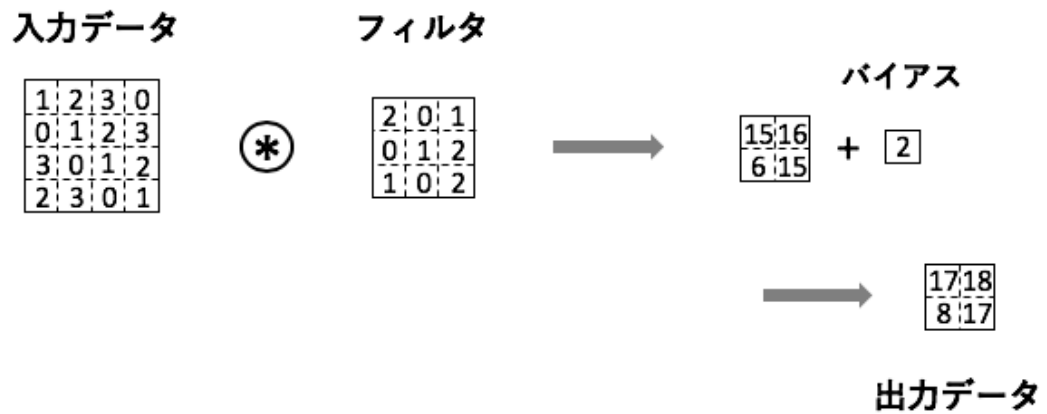


図4.20 畳み込み層における処理（バイアス付加）

フィルタ数

2つ目がフィルタ数である。図4.18における例ではフィルタ数は1であった。実際には、事前に用意しておくフィルタの数を設定することができ、一般に複数個のフィルタを用意する。

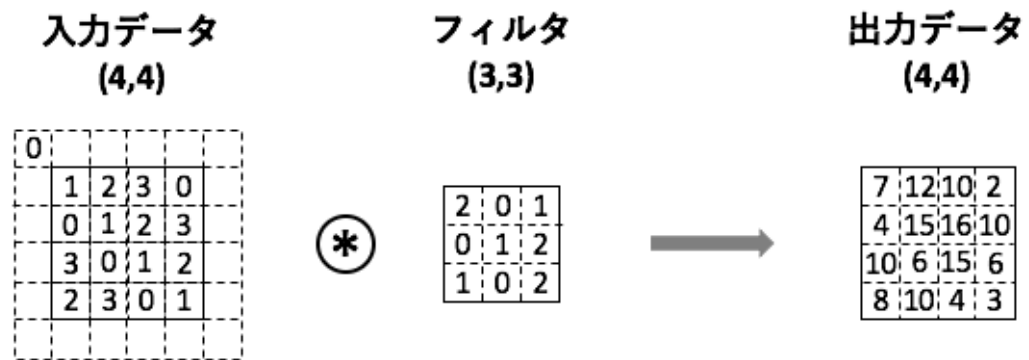


図4.21 パディング処理の例（パディング幅1）

パディング

3つ目はがパディング（padding）である。パディングは、畳み込み層の処理を行う前に、入力データの周囲に固定のデータ（一般には0）を埋め込むことである。図4.18における例ではパディングは設定していない、つまりパディング幅は0である。

パディング処理の例を図4.21に示す。この例では、パディング幅を1と設定しており、サイズ(4, 4)の入力データの周囲に固定データ（0）を埋め込んでいる。なお、図4.21において空となっている要素は全て0であることを示している。こうしてパディング処理をなされた入力データは(4, 4)のサイズから(6, 6)のサイズに変更され、畳み込み演算によって出力データ（サイズ(4, 4)）を得ている。

パディングを用いる主な理由は、出力サイズを調整するためである。例えば、図4.18の例の場合、出力データのサイズは(2, 2)となっており、入力データのサイズから2要素分縮小されていることがわかる。ディープラーニングのように畳み込み層を複数個繋げる構造をしている場合、畳み込み演算のたびにデータのサイズが縮小されてしまい、繰り返し畳み込み演算を行うことができなくなってしまう。このような事態を避けるために、パディングは用いられている。

ストライド

4つ目がストライド（stride）である。ストライドは、フィルタを適用する位置の間隔のことである。図4.18に例の場合、ストライドは1であったが、これを2に変更するとフィルタを適用するウィンドウの間隔が2要素ごとになる。図4.22はストライドを2と設定した時の例である。

以上までの例を見ると、パディングやストライドの設定値によって出力データのサイズが変化する。入力データのサイズを (H, W) 、フィルタサイズを (FH, FW) 、出力データのサイズを (OH, OW) 、パディン

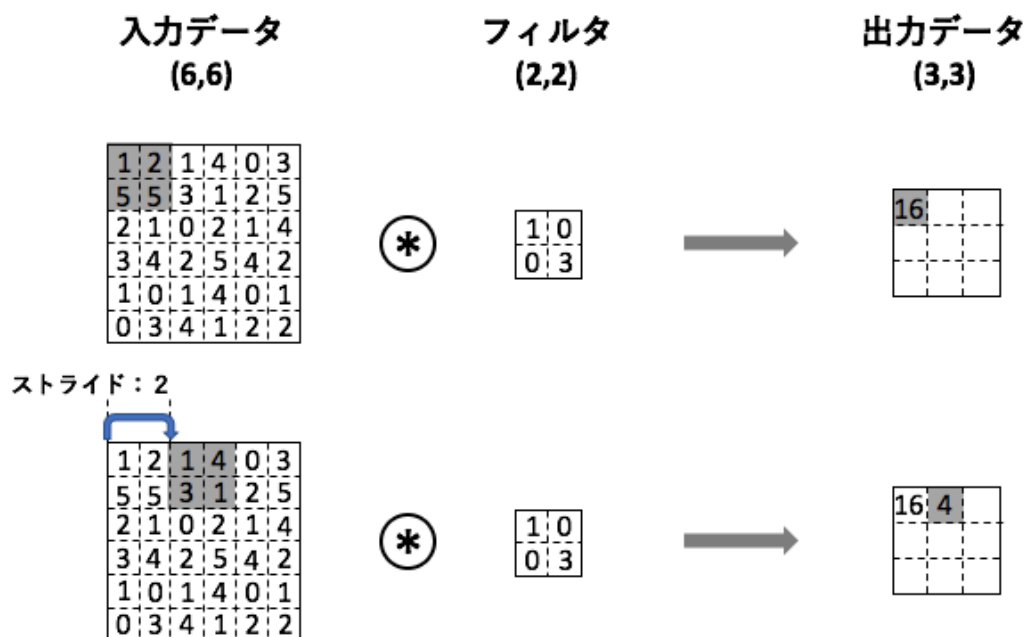


図4.22 ストライドの例 (ストライド2)

グを P 、ストライドを S としたとき、出力データのサイズは以下のように定式化される：

$$OH = \frac{H + 2P - FH}{S} + 1 \quad (4.3.1)$$

$$OW = \frac{W + 2P - FW}{S} + 1 \quad (4.3.2)$$

出力データのサイズは整数でなければならないため、上式を考慮してフィルタサイズ、パディング、ストライドの値を設定しなければならない。ただし、用いるディープラーニングのフレームワークによっては、出力データのサイズを最も近い整数に丸め込むように実装されている。

以上が、畳み込み層におけるハイパーパラメータである。

ミニバッチ

全結合のニューラルネットワークにおいて、学習時にミニバッチを用いた。畳み込みニューラルネットワークにおいても同様に、ミニバッチを用いる。

畳み込みニューラルネットワークにおいてミニバッチに対応するために、各層を流れるデータを4次元データとして格納する。具体的には、(バッチサイズ, チャンネル, 縦, 横)という順にデータを格納する。このミニバッチを用いた畳み込み演算の処理フローを図4.23に示す。図4.23の通り、ミニバッチに対応した畳み込み演算処理では、バッチサイズ N 個のデータに対して一回の畳み込み演算によって処理されている。

以上が、畳み込み層における処理である。この畳み込み層によって処理されたデータはそのまま活性化

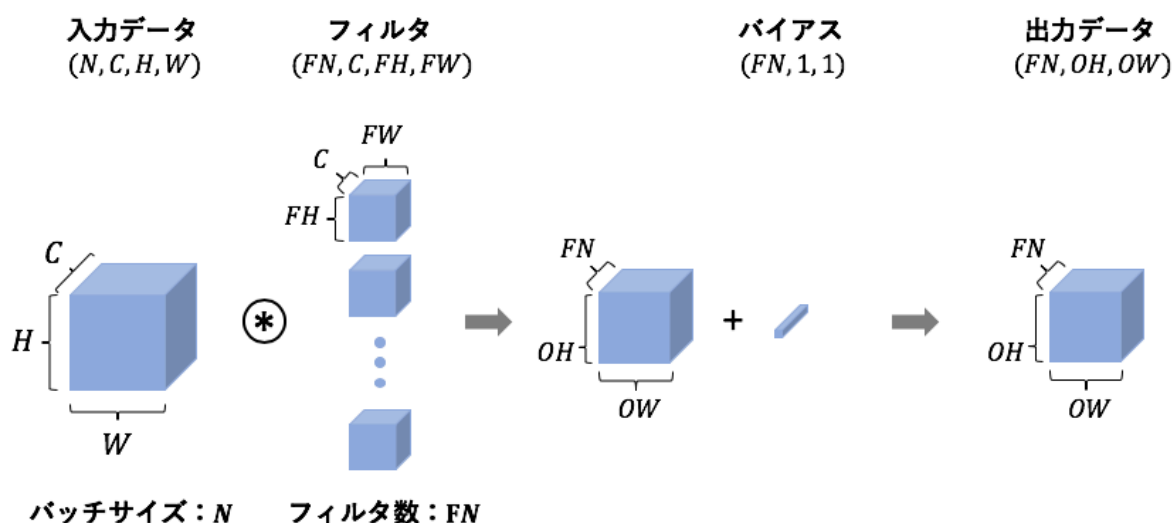


図4.23 ミニバッチ対応の畳み込み演算処理

関数層に流れたのち、次項で記述するプーリング層へと流れ込む。

4.3.3 プーリング層

畳み込み層によって処理されたデータは、活性化関数にかけられたのちに、プーリング層へと流れ込む。

プーリング層ではプーリングと呼ばれる、縦・横方向の空間を小さくする演算を行う層である。具体的には、プーリングフィルタを設定し、そのフィルタがかけられたウィンドウに対してプーリングを行う。プーリングにはいくつか種類があるが、一般にMax Poolingが用いられる。Max Poolingとは、ウィンドウ領域内の要素の中で最大の要素を抽出するというプーリング手法である。Max Poolingの他に、ウィンドウ領域内の要素平均を抽出するAverage Poolingなどが存在する。

プーリング層においても、フィルタサイズやストライドといったハイパーパラメータが存在する。Max Poolingを用いたプーリング層における処理例を図4.24に示す。図4.24ではサイズ(4, 4)の入力データに対して、フィルタサイズ2、ストライド2のプーリングフィルタをかけ、サイズ(2, 2)の出力データを得ている。なお、一般的にはプーリングのフィルタサイズとストライドは同じ値を設定する。

プーリング層における処理は以上だが、プーリング層にはいくつか特徴がある。

1つ目は、学習するパラメータがないということだ。Max Poolingには学習するパラメータを持たないため、プーリング層をネットワークに組み込んでも計算コストは対して増加しない。2つ目は、チャンネル数が増えないことである。Max Poolingはチャンネルごとに独立して行われるためである。3つ目は、微小な位置変化に対してロバストであることだ。入力データの小さな位置ズレに対して、プーリングはほぼ同じような結果を出力する。そのため、入力データの微小なズレに対してロバストであり、畳み込みニュー

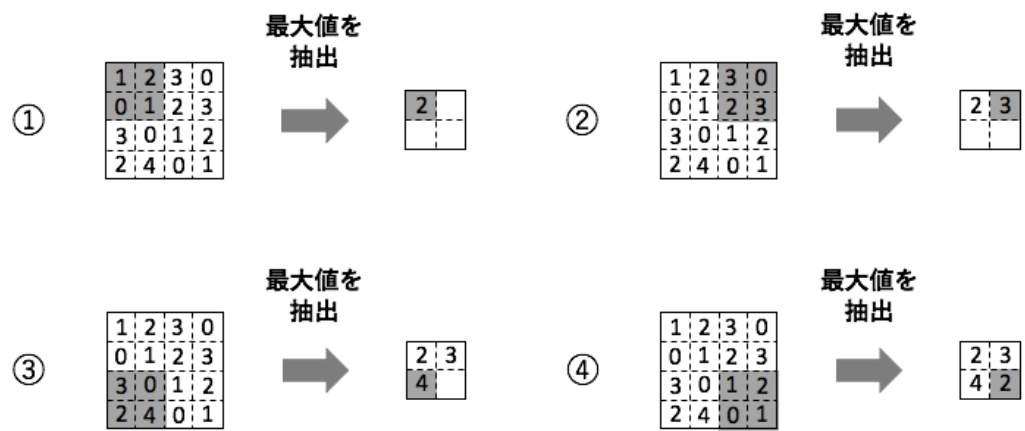


図4.24 Max Poolingによる処理例

ラルネットワークに対して相性がいい。

以上が畳み込みニューラルネットワークにおける層であり、これらを図4.17のように組み合わせることでディープラーニングを行うことができる。

第5章

リングダウン重力波の解析

本章では、リングダウン重力波の解析において取り扱うデータ、マッチドフィルタリングの実装、及び畳み込みニューラルネットワークの実装について記述する。以下で、解析対象とするブラックホールリングダウンは全て、連星合体によって生じたブラックホールからのリングダウンであるとした。本解析の目標は、マッチドフィルタリングによる解析と畳み込みニューラルネットワークによる解析の比較である。

なお、本研究における解析は全て、Google社が無償提供しているGoogle ColaboratoryのGPUを使用している。Google Colaboratoryとは、教育・研究機関へ機械学習の普及を目的としたGoogle社の研究プロジェクトであり、ブラウザ上でJupyter Notebookライクに取り扱うことができる仕様になっている。また、Google Colaboratory上における使用言語はPython3系である。

5.1 リングダウン重力波の信号生成

2.2節では連星合体後のカー・ブラックホールからの重力波についての式を導出した。本研究では簡略化のため、よく近似された式を用いる。

ブラックホールリングダウンによって放出される重力波は、球面調和関数のインデックス (l, m) 及び、モードを表すインデックス n をパラメータとして持つ減衰正弦波の形をしている。これは、Echenverria[4]によれば、

$$h(t) = Ae^{-t/\tau_{lmn}} \sin(\omega_{lmn}t + \phi_{lmn}) \quad (5.1.1)$$

と表すことができる。ここで、 A は振幅、 $\tau_{lmn}, \omega_{lmn}, \phi_{lmn}$ はそれぞれ各モードの減衰時間、角周波数、位相差を表している。なお、この式において、基本モード、つまり22モードと高次モードにおけるInclinationの寄与は無視されている。Inclinationによる寄与に関する詳細な議論は付録にて行う。

表5.1 各モードにおけるフィッティング係数 ($n = 0$)、Emanuele[5]より

(l, m)	f_1	f_2	f_3	q_1	q_2	q_3
(2, 1)	0.6000	-0.2339	0.4175	-0.3000	2.3561	-0.2277
(2, 2)	1.5251	-1.1568	0.1292	0.7000	1.4187	-0.4990
(3, 3)	1.8956	-1.3043	0.1818	0.9000	2.3430	-0.4810
(4, 4)	2.3000	-1.5056	0.2244	1.1929	3.1191	-0.4825

5.1.1 周波数とQ値

各モードにおける周波数及び減衰時間はEmanuele[5]らによって近似式を与えられており、以下のように表される：

$$F_{lmn} = M\omega_{lmn} = f_1 + f_2(1-j)^{f_3} \quad (5.1.2)$$

$$Q_{lmn} = \pi f_{lmn} \tau_{lmn} = q_1 + (1-j)^{q_3} \quad (5.1.3)$$

なお、 $f_i, q_i (i = 1, 2, 3)$ はフィッティングによって求められた係数であり、各値は表5.1の通りである。また、 j はブラックホールの角運動量を無次元化したもの、 F_{lmn}, Q_{lmn} はそれぞれ無次元化された周波数とQ値（振動の安定度合いを表す量）、 M は万有引力定数 G 及び光速 c によって規格化されたブラックホールの質量である。具体的には、ブラックホールの質量を $M_{BH}[\text{kg}]$ 、角運動量を $J[\text{m}^2\text{kg/s}]$ 、万有引力定数を $G[\text{m}^3\text{kg}^{-1}\text{s}^{-2}]$ 、光速を $c[\text{m/s}]$ としたとき、

$$j = \frac{cJ}{GM_{BH}^2} \quad (5.1.4)$$

$$M = \frac{M_{BH}G}{c^3} \quad (5.1.5)$$

であり、 j の取りうる値は $0 \leq j \leq 1$ である。

以上の量を用いると、求めるべき波形の式は

$$h(t) = A \exp\left(-\frac{\pi f_{lmn}}{Q_{lmn}}t\right) \sin(2\pi f_{lmn}t + \phi_{lmn}) \quad (5.1.6)$$

$$f_{lmn} = \frac{F_{lmn}}{2\pi M} \quad (5.1.7)$$

とかける。ここで、計算のために重力波の周波数 f_{lmn} を無次元化された周波数 F_{lmn} を用いて記述する：

$$\begin{aligned}
 f_{lmn} &= \frac{F_{lmn}}{2\pi M} \\
 &= \frac{c^3}{2\pi G} \frac{1}{M_{BH}} F_{lmn} \\
 &= \frac{c^3}{2\pi G M_{sun}} \frac{M_{sun}}{M_{BH}} F_{lmn} \\
 &\sim 32 \times \left(\frac{M_{BH}}{M_{sun}}\right)^{-1} F_{lmn} \times 10^3 \quad [\text{Hz}]
 \end{aligned} \quad (5.1.8)$$

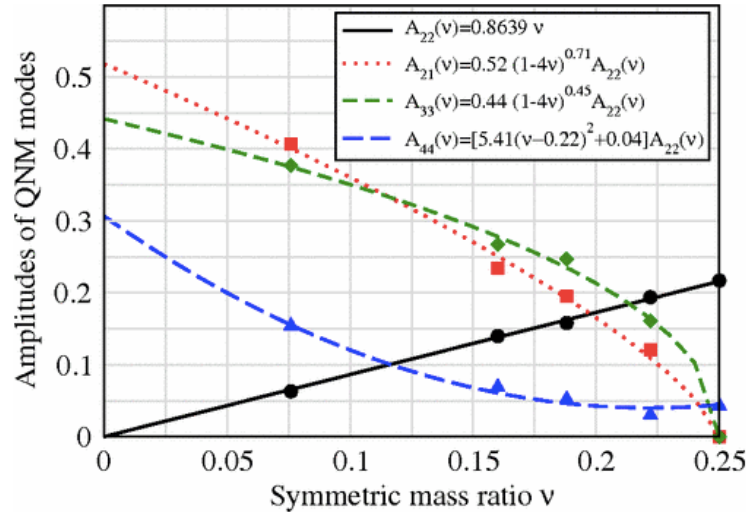


図5.1 質量比と各モードの振幅比の関係。S. Gossan[6]より引用。

ただし、 M_{sun} は太陽質量である。本研究では、上式で求めた周波数及び式(5.1.3)で求めたQ値を用いて、式(5.1.6)に従って計算されたリングダウン重力波信号を用いる。

このように、リングダウン重力波の各モードにおける周波数とQ値は、ブラックホールの質量及びスピンによって決定される。

5.1.2 振幅

前項に引き続き、リングダウン重力波における振幅についての議論を行う。

各モードにおける振幅は、連星をなす2天体の質量比によって決定される。具体的にはS. Gossan[6]らによって計算されており、以下のように表される：

$$A_{22} = 0.864\nu \quad (5.1.9)$$

$$A_{21} = 0.52(1 - 4\nu)^{0.71} A_{22} \quad (5.1.10)$$

$$A_{33} = 0.44(1 - 4\nu)^{0.45} A_{22} \quad (5.1.11)$$

$$A_{44} = [5.4(\nu - 0.22)^2 + 0.04] A_{22} \quad (5.1.12)$$

ここで、各振幅における下付き添字は球面調和関数のインデックス(l, m)に対応している。また、 ν は、連星における2天体の質量比を表しており、それぞれの質量を m_1, m_2 とすると、

$$\nu = \frac{m_1 m_2}{(m_1 + m_2)^2} \quad (5.1.13)$$

である。この質量比 ν を変数として、各モードの振幅比をプロットしたものが図5.1である。なお、 ν が最大となるのは $m_1 = m_2$ の時であり、その値は0.25である。

以上のように、連星をなす2天体の質量比によって、各モードの振幅が変化する。まとめると、リングダウン重力波（モード $n = 0$ のみ）の波形は、球面調和関数のインデックス (l, m) を用いて、

$$h(t) = \sum_{l,m} A_{lm} \exp\left(-\frac{\pi f_{lm} t}{Q_{lm}}\right) \sin(2\pi f_{lm} t + \phi_{lm}) \quad (5.1.14)$$

で書かれる。

5.1.3 リングダウン重力波信号の生成コード

前項までに導出した、近似されたリングダウン重力波信号の生成コードをソースコード5.1に示す。

ソースコード 5.1 Ringdown_signal.py

```

1  import math
2  import numpy as np
3
4
5  def Ringdown_signal_generator(m1, m2, a, t, A22, theta, tau_index):  # リングダウン重力波信号生成
    関数
6      q = m2 / m1
7      if q < 1:
8          q = 1 / q
9
10     nu = q / ((1 + q)**2)          # 質量比
11
12     m = round((m1 + m2) * 0.96, 1)  # 連星合体後の質量(4%は重力波のエネルギーに変換さ
    れる)
13     a = round(a, 3)
14
15     t[:tau_index] = 0
16     t[tau_index:] = t[tau_index:] - t[tau_index]
17
18     # QNMパラメータ
19     f, q = QNM_parameter(a, m)
20
21     f_22 = f[0]
22     f_33 = f[1]
23     f_44 = f[2]
24     q_22 = q[0]
25     q_33 = q[1]
26     q_44 = q[2]
27
28     # 球面調和関数(Inclinationによる寄与)
29     Y22 = plus_GW_spherical_harmonics(theta, 2, 2)
30     Y33 = plus_GW_spherical_harmonics(theta, 3, 3) / Y22
31     Y44 = plus_GW_spherical_harmonics(theta, 4, 4) / Y22
32
33     # 信号振幅
34     A33 = Y33 * A22 * 0.44 * (1 - 4*nu)**0.45
35     A44 = Y44 * A22 * (5.4 * (nu - 0.22)**2 + 0.04)
36
37     # 信号生成
38     signal_22 = A22 * np.exp(-math.pi * f_22 / q_22 * t) * np.sin(2 * math.pi * f_22 * t)
39     signal_33 = A33 * np.exp(-math.pi * f_33 / q_33 * t) * np.sin(2 * math.pi * f_33 * t)
40     signal_44 = A44 * np.exp(-math.pi * f_44 / q_44 * t) * np.sin(2 * math.pi * f_44 * t)
41     signal = signal_22 + signal_33 + signal_44
42
43     return np.real(signal)

```

```

44
45 def QNM_parameter(a,m):    # QNMパラメータ計算関数
46     f_sig = np.array([[1.5251, -1.1568, 0.1292], [1.8956, -1.3043, 0.1818], [2.3000, -1.5056, 0.2244]])
47     # f_i
48     q_sig = np.array([[0.7000, 1.4187, -0.4990], [0.9000, 2.3430, -0.4810], [1.1929, 3.1191, -0.4825]])
49     # q_i
50
51     f = np.zeros((3, a.shape[0], m.shape[0]))
52     q = np.zeros((3, a.shape[0], m.shape[0]))
53
54     for i in range(len(f_sig)):
55         for j in range(a.shape[0]):
56             for k in range(m.shape[0]):
57                 f[i,j,k] = 32 * (f_sig[i, 0] + f_sig[i, 1] * (1 - a[j])**f_sig[i, 2]) / m[k] * 10**3
58                 q[i,j,k] = q_sig[i, 0] + q_sig[i, 1] * (1 - a[j])**q_sig[i, 2]
59
60     return f, q
61
62 def plus_GW_spherical_harmonics(theta, l, m):    # 球面調和関数の計算
63
64     s = -2
65     phi = 0
66
67     y1 = spinweighted_spherical_harmonics(theta, phi, l, m, s)
68     y2 = (-1)**(-1 * s + m) * np.conj(spinweighted_spherical_harmonics(theta, phi, l, m, s))
69
70     Ylm = y1 + (-1)**l * y2
71
72     return Ylm
73
74 def spinweighted_spherical_harmonics(theta, phi, l, m, s):    # スピンウェイト球面調和関数の計算
75
76     n = -1 * s
77     d = small_dmatrix(theta, l, m, n)
78
79     sYlm = (-1)**s * np.sqrt((2*l + 1) / 4 * math.pi) * d * np.exp(1j * m * phi)
80
81     return sYlm
82
83 def small_dmatrix(theta, l, m, n):    # Wigner small D-matrixの計算関数
84
85     maxk = min(l - m, l - n) + 1
86     SUM = np.zeros(theta.shape[0],1)
87
88     for k in range(maxk):
89         SUM[:,] += (-1)**k * (np.sin(theta / 2))**(2*l - m - n - 2*k) * (np.cos(theta / 2))**(m +
90             n + 2*k) / (math.factorial(k) * math.factorial(l - m - k) * math.factorial(l - n - k) * math.
91             factorial(m + n + k))
92
93     d = (-1)**(l - n) * np.sqrt(math.factorial(l + m) * math.factorial(l - m) * math.factorial(l + n)
94         * math.factorial(l - n)) * SUM[:,]
95
96     return d

```

4行目から42行目にかけて定義されている関数が、リングダウン重力波信号を生成するメイン関数である。この関数は連星をなす2天体の質量、スピン、時間軸、基本モード(22モード)の振幅、Inclination、重力波信号到達時間を引数とし、時間ドメインでのリングダウン重力波信号を返す。17行目から25行目にかけて、5.1.1項で記述した各モードに対する周波数及びQ値を計算しており、その際44行目から55行目にかけて定義されている関数を用いている。27行目から30行目にかけて、Inclinationによる各

モードの振幅に対する寄与（付録にて後述）を計算しており、その際57行目以降にかけて定義されている関数を用いている。32行目から34行目にかけて、5.1.2項で記述した各モードにおける振幅を計算している。36行目から40行目にかけて、以上で求めた周波数、Q値、振幅をもとに各モードの信号を計算し、最後にそれらを足し合わせることによってリングダウン重力波の信号を計算している。

5.2 干渉計のノイズスペクトル

本研究では、干渉計のノイズスペクトルとして、アメリカの重力波検出器であるLIGOがオープンソース[7]として公開しているストレインデータ（O1時のもの）を用いて実データ解析を行った。実際には、リングダウン部分のみでの検出は不可能なノイズレベルであったため、重力波信号を含んでいない部分を用いて擬似的な干渉計データを作成し、実データ解析に限りなく近い解析を行うことにした。

LIGOがオープンソースとして提供しているストレインデータには4096Hzサンプリングのものと16384Hzサンプリングのものがあるが、本研究において解析対象とする重力波の周波数帯域を考慮すると、4096Hzサンプリングのもので十分であったためそちらを用いた。

干渉計からのデータは理想的にはガウシアンノイズであるが、実際の干渉計からのデータにはそうでないものが含まれている可能性がある。そこで、パーミュテーションテストという手法を用い、ノイズスペクトルをそのままにデータを並び替えることでノイズのガウス性を担保したノイズデータを作成する。

通常、干渉計からのデータはホワイトノイズではなく、周波数に依存したノイズを多く含んでいる。そのため、LIGOが提供しているデータをそのまま並び替えると、低周波数帯や高周波数帯の感度が悪い部分によって、全体的に感度も悪く、ホワイトノイズに近いノイズスペクトルになってしまう。そこで、干渉計からのデータに対して一度ホワイトニングを行い、周波数依存性のないデータに変える。その後、データをランダムに並べ替え、再度元のノイズスペクトルによって色付けすることで擬似的なノイズスペクトルを生成する。この一連の作業をパーミュテーションテストという。

以下ソースコード5.2が、擬似的干渉計ノイズスペクトルを生成するソースコードである。

ソースコード 5.2 noise.py

```
1 import numpy as np
2 import matplotlib.mlab as mlab
3
4
5
6 # データの読み込み
7 with open("GW150914_4096Hz_1s.txt", "r") as f:
8     strain = f.read()
9
10 strain = np.array(strain) # numpy配列への変換
11
12
13 # 準備
14 fs = 4096 # サンプリング周波数
15 L = 4096 # データ長
```

```
16
17
18 def rand_sample(data, fs, L):
19     # PSD推定
20     dt = 1.0 / fs
21     NFFT = fs      # Fourier変換長
22
23     psd, freqs = mlab.psd(data, Fs=fs, NFFT=NFFT)
24
25     # ホワイトニング
26     Nt = L
27     data2 = data
28
29     hf = np.fft.rfft(data2)
30     white_hf = hf / np.sqrt(psd / dt / 2)
31     white_hf = np.fft.irfft(white_hf, Nt)
32
33     # ランダム化
34     rand_hf = np.random.permutation(white_hf)
35     rand_hf = np.fft.rfft(rand_hf)
36     rand_hf = rand_hf * np.sqrt(psd / dt / 2)
37     rand_hf = np.fft.irfft(rand_hf)
38
39
40     # 規格化
41     norm_factor = np.median(np.abs(rand_hf)) / np.median(np.abs(data))
42     rand_hf = rand_hf / norm_factor
43
44     return rand_hf
45
46
47 # パーミュテーションテスト
48 strain = rand_sample(strain, fs, L)
```

6行目から10行目にかけて、LIGOの干渉計データの読み込みとNumpy配列への変換を行っている。LIGOがオープンソースとして提供しているデータは4096Hzサンプリングの4096秒分のデータであるが、本研究ではこのうち1秒分のデータを切り取って使用した。13行目から15行目ではその情報をセットアップとして記述している。18行目から44行目にかけて定義されている関数は、上述のパーミュテーションテストを実装したものである。この関数は元のデータ（タイムドメイン）、サンプリング周波数、データ長を引数とし、パーミュテーション後のストレインデータを返す。この関数では、23行目においてパワースペクトル密度（Power Spectrum Density : PSD）の推定を行い、25行目から31行目にかけてホワイトニングを行っている。その後33行目から37行目にかけてデータの入れ替えと再度色付けを行い、40行目から42行目にかけて規格化を行っている。最後に48行目において、元のデータを上述のように定義された関数にかけ、パーミュテーションテスト後の疑似的干渉計ノイズデータを得ている。

このソースコードを元に生成した疑似的干渉計ノイズスペクトルを図5.2に示す。また、これをタイムドメインに変換した時の信号の様子を図5.3に示す。図5.2を見てわかる通り、元の干渉計のノイズスペクトルとほぼ同等のノイズスペクトルを実現している。

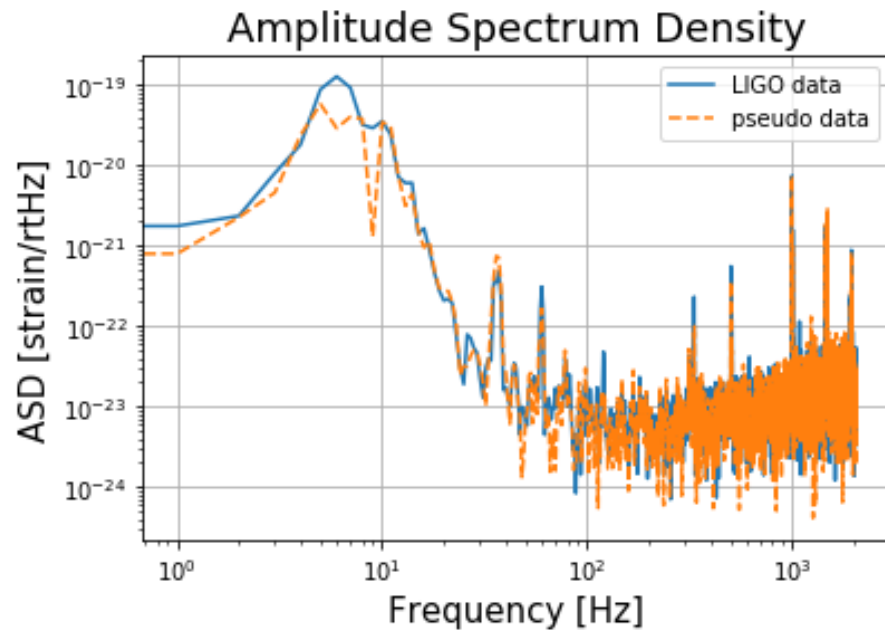


図5.2 疑似的干渉計ノイズスペクトル：青実線が元のデータ。橙点線がパーミュテーションテスト後の疑似的干渉計データ

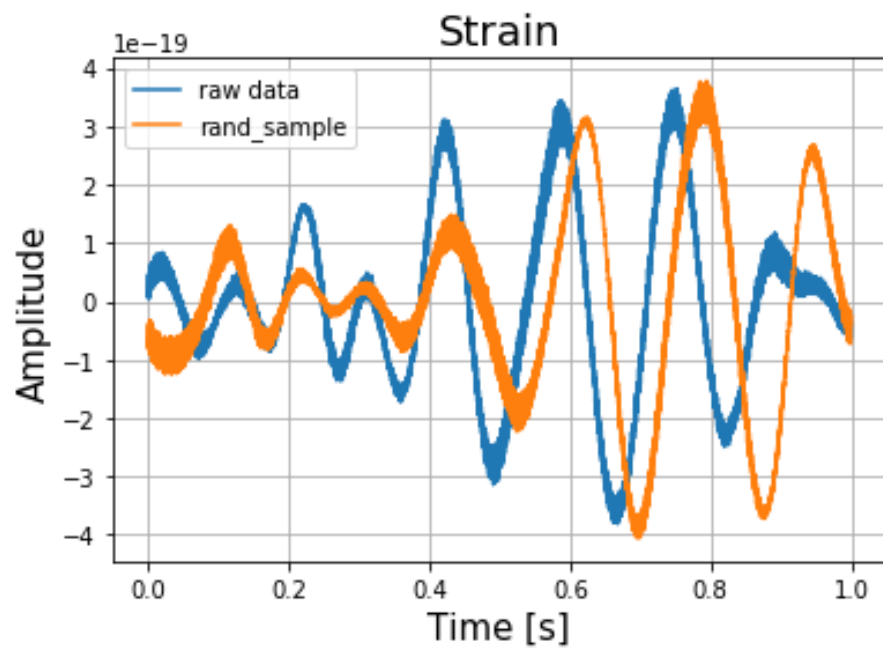


図5.3 疑似的干渉計ノイズデータ（タイムドメイン）：青実線が元のデータ。橙実線がパーミュテーションテスト後の疑似的干渉計データ

5.3 マッチドフィルタリング

前々節においてリングダウン重力波信号を生成することができたので、マッチドフィルタリングに用いるテンプレートを作成することができる。また、前節において干渉計のノイズスペクトルを実現することができたので、テンプレートとからマッチドフィルタを計算することができる。本節ではマッチドフィルタリングの実装について記述する。マッチドフィルタリングの実装について記述するにあたって、はじめにテンプレートの作成について記述する。

5.3.1 テンプレートの作成

前章において、マッチドフィルタリングは検出器から得られるデータと、理論波形のテンプレート群との畳み込みを計算することで、検出・非検出の検定を行う手法であると記述した。そのため、テンプレートの作成では予めサーチに用いるサーチパラメータを設定し、そのパラメータに基づいた理論波形群を作成する。今回サーチに用いるサーチパラメータは、連星合体後のブラックホールの質量、スピン、リングダウン重力波信号の到達時刻、連星をなす2天体の質量比（高調波を含めたサーチのため）、Inclinationの5つである。

テンプレートのソースコードを以下のソースコード5.3に示す。

ソースコード 5.3 template.py

```

1  import math
2  import numpy as np
3  from Ringdown_signal import *
4
5  # データの設定
6  L = 4096      # データ長
7  t = np.linspace(0, 1, L)
8
9  # サーチパラメータの設定
10 massvector = np.linspace(mass_lower, mass_upper, mass_num)    # 連星合体後のブラックホールの質量
11 massvector = np.round(massvector, 3)
12
13 kerrvector = np.linspace(kerr_lower, kerr_upper, kerr_num)    # ブラックホールのスピン
14 kerrvector = np.round(kerrvector, 3)
15
16 timevector = np.linspace(time_lower, time_upper, time_num)    # 重力波の到達時刻
17 timevector = np.round(timevector, 3)
18
19 nuvector = np.linspace(nu_lower, nu_upper, nu_num)    # 連星をなす2天体の質量比
20 nuvector = np.round(nuvector, 3)
21
22 thetavector = np.linspace(theta_lower, theta_upper, theta_num)    # Inclination
23 thetavector = np.round(thetavector, 3)
24
25 # QNMパラメータの計算
26 f, q = QNM_parameter(kerrvector, massvector)
27
28 f_22 = f[0, :, :]
29 f_33 = f[1, :, :]

```

```

30 f_44 = f[2, :, :]
31
32 q_22 = q[0, :, :]
33 q_33 = q[1, :, :]
34 q_44 = q[2, :, :]
35
36 # 球面調和関数(Inclinationによる寄与)
37 Y22 = plus_GW_spherical_harmonics(thetavector, 2, 2)
38 Y33 = plus_GW_spherical_harmonics(thetavector, 3, 3) / Y22
39 Y44 = plus_GW_spherical_harmonics(thetavector, 4, 4) / Y22
40
41 # 信号振幅
42 A22 = 1.0 # テンプレートのため
43 A33 = np.outer(Y33, A22 * 0.44 * (1 - 4 * nuvector) ** 0.45)
44 A44 = np.outer(Y44, A22 * (5.4 * (nuvector - 0.22) ** 2 + 0.04))
45
46
47 # テンプレートの作成
48 templates = np.zeros((L, len(massvector), len(kerrvector), len(nuvector), len(thetavector)))
49
50 for i in range(len(massvector)):
51     for j in range(len(kerrvector)):
52         for k in range(len(nuvector)):
53             for l in range(len(thetavector)):
54                 temp22 = np.exp(-math.pi * f_22[j, i] / q_22[j, i] * t) * np.sin(2 * math.pi * f_22[j, i] * t)
55                 temp33 = A33[l, k] * np.exp(-math.pi * f_33[j, i] / q_33[j, i] * t) * np.sin(2 * math.pi * f_33
[j, i] * t)
56                 temp44 = A44[l, k] * np.exp(-math.pi * f_44[j, i] / q_44[j, i] * t) * np.sin(2 * math.pi * f_44
[j, i] * t)
57
58                 templates[:, i, j, k, l] = temp22 + temp33 + temp44

```

9 行目から 23 行目にかけて、テンプレートに用いるサーチパラメータの設定を行っている。変数の添え字である `_lower`、`_upper`、`_num` はそれぞれ各サーチパラメータにおける下限値、上限値、パラメータ数である。25 行目から 34 行目にかけて、設定したサーチパラメータに基づいた QNM パラメータ（周波数と Q 値）を計算している。また、36 行目から 42 行目にかけて、設定したサーチパラメータに基づいた各モードの振幅を計算している。最後に 47 行目以降において、これまでに計算した QNM パラメータと信号振幅を用いたテンプレート群を作成している。なお、重力波の信号到達時刻は、後のマッチドフィルタ計算において使用される。

5.3.2 マッチドフィルタリング

前項において作成したテンプレート群を用いて、マッチドフィルタを計算する。本項では、マッチドフィルタリングの実装について記述する。マッチドフィルタリングの実装ソースコードを、ソースコード 5.4 に示す。（`matchedfilter` 以降、要改良）

ソースコード 5.4 `matchedfilter.py`

```

1 import math
2 import numpy as np
3 import matplotlib.mlab as mlab
4 from scipy import signal

```

```

5
6
7 ## Matched Filtering計算
8
9 def matched_filter(noise, signal, fs, massvector, kerrvector, timevector, nuvector, thetavector):
10
11     # 準備
12     dt = 1.0 / fs
13     L = len(noise)      # データ長
14     t = np.linspace(0, 1, L)
15     NFFT = fs          # Fourier変換長
16
17     # PSD推定
18     psd, freqs = mlab.psd(noise, Fs=fs, NFFT=NFFT)
19     df = 1.0           # 積分におけるbin幅
20     Lf = len(freqs)
21
22     # 擬似信号作成(Fourier変換まで)
23     s = noise + signal
24     w = signal.hann(L)  # 窓関数(ハン関数)
25     w = w / np.mean(w)
26     s_fft = np.fft.rfft(s * w) / L
27
28     # テンプレート作成
29     #-----QNMパラメータの計算-----#
30     f, q = QNM_parameter(kerrvector, massvector)
31
32     f_22 = f[0, :, :]
33     f_33 = f[1, :, :]
34     f_44 = f[2, :, :]
35
36     q_22 = q[0, :, :]
37     q_33 = q[1, :, :]
38     q_44 = q[2, :, :]
39
40     #-----球面調和関数(Inclinationによる寄与-----#
41     Y22 = plus_GW_spherical_harmonics(thetavector, 2, 2)
42     Y33 = plus_GW_spherical_harmonics(thetavector, 3, 3) / Y22
43     Y44 = plus_GW_spherical_harmonics(thetavector, 4, 4) / Y22
44
45     #-----信号振幅-----#
46     A22 = 1.0      # テンプレートのため
47     A33 = np.outer(Y33, A22 * 0.44 * (1 - 4*nuvector)**0.45)
48     A44 = np.outer(Y44, A22 * (5.4 * (nuvector - 0.22)**2 + 0.04))
49
50
51     #-----テンプレートの作成-----#
52     templates = np.zeros((L, len(massvector), len(kerrvector), len(nuvector), len(thetavector)))
53
54     for i in range(len(massvector)):
55         for j in range(len(kerrvector)):
56             for k in range(len(nuvector)):
57                 for l in range(len(thetavector)):
58                     temp22 = np.exp(-math.pi * f_22[j,i] / q_22[j,i] * t) * np.sin(2 * math.pi * f_22[j,i] * t)
59                     temp33 = A33[l,k] * np.exp(-math.pi * f_33[j,i] / q_33[j,i] * t) * np.sin(2 * math.pi *
60                     f_33[j,i] * t)
61                     temp44 = A44[l,k] * np.exp(-math.pi * f_44[j,i] / q_44[j,i] * t) * np.sin(2 * math.pi *
62                     f_44[j,i] * t)
63
64                     templates[:,i,j,k,l] = temp22 + temp33 + temp44

```

```

64  # Matched Filter
65  #-----規格化-----#
66  temp_fft = np.zeros((NFFT/2.0, len(massvector), len(kerrvector), len(nuvector), len(thetavector)))
67  sigma_sqr = np.zeros((len(massvector), len(kerrvector), len(nuvector), len(thetavector)))
68  templates_cal = np.zeros((L, len(massvector), len(kerrvector), len(nuvector), len(thetavector)))
69  temp_fft_cal = np.zeros((NFFT/2.0, len(massvector), len(kerrvector), len(nuvector), len(thetavector)
70  )))
71  sigma_sqr_cal = np.zeros((len(massvector), len(kerrvector), len(nuvector), len(thetavector)))
72
73  w = signal.hann(NFFT)
74  w = w / np.mean(w)
75
76  for i in range(len(massvector)):
77      for j in range(len(kerrvector)):
78          for k in range(len(nuvector)):
79              for l in range(len(thetavector)):
80                  temp_fft[:,i,j,k,l] = np.fft.rfft(templates[:,i,j,k,l] * w) / NFFT
81                  sigma_sqr[i,j,k,l] = 4 * np.real(np.sum((temp_fft[1:,i,j,k,l] * np.conj(temp_fft[1:,i,j,k,l]) /
82                  psd[1:] * df))
83                  templates_cal[:,i,j,k,l] = templates[:,i,j,k,l] / np.sqrt(sigma_sqr[i,j,k,l])
84                  temp_fft_cal[:,i,j,k,l] = np.fft.rfft(templates_cal[:,i,j,k,l] * w) / NFFT
85                  sigma_sqr_cal[i,j,k,l] = 4 * np.real(np.sum((temp_fft_cal[1:,i,j,k,l] * np.conj(temp_fft_cal
86                  [1:,i,j,k,l]) / psd[1:] * df))
87
88  matchedfilter_cal = np.zeros((len(massvector), len(kerrvector), len(nuvector), len(timevector), len(
89  thetavector)))
90  matchedfilter = np.zeros((len(massvector), len(kerrvector), len(nuvector), len(timevector), len(
91  thetavector)))
92  Amplitude = np.zeros((len(massvector), len(kerrvector), len(nuvector), len(timevector), len(
93  thetavector)))
94
95  for i in range(len(massvector)):
96      for j in range(len(kerrvector)):
97          for k in range(len(nuvector)):
98              for l in range(len(timevector)):
99                  for m in range(len(thetavector)):
100                      matchedfilter_cal[i,j,k,l,m] = 4 * np.real(np.sum((s_fft[1:] * np.conj(temp_fft_cal[:,i,j,
101                      k,m]) * np.exp(2 * math.pi * 1j * freqs[1:] * timevector[l]) / psd[1:] * df))
102                      matchedfilter[i,j,k,l,m] = 4 * np.real(np.sum((s_fft[1:] * np.conj(temp_fft[:,i,j,k,m]) *
103                      np.exp(2 * math.pi * 1j * freqs[1:] * timevector[l]) / psd[1:] * df))
104                      Amplitude[i,j,k,l,m] = matchedfilter[i,j,k,l,m] / sigma_sqr_cal[i,j,k,m]
105
106  SNR = np.sqrt(np.abs(matchedfilter_cal)**2)
107
108  return SNR

```

ここでは、9行目以降においてマッチドフィルタを計算する関数の定義を行っている。この関数は、パースミュレーションテスト後のノイズデータ、疑似的リングダウン重力波信号、サンプリング周波数、各種サーチパラメータを引数とし、マッチドフィルタリングによって計算されたSNRを返す関数である。

11行目から15行目にかけて、マッチドフィルタを計算するにあって必要な準備を行っている。17行目から20行目にかけて、式(3.2.16)右辺の分母で用いる、ノイズのパワースペクトル密度を計算している。22行目から26行目にかけて、引数のノイズデータと疑似的リングダウン重力波信号から、疑似的信号を作成している。その際、信号解析を行うためデータに対して窓関数をかけており、本研究では窓関数としてHann窓を用いている。28行目から62行目にかけて、前項で記述したテンプレート群の作成を

行っている。65行目から83行目にかけて、上で作成されたテンプレート群の規格化を行っている。ここでの規格化とは、各テンプレートに対して同じテンプレート同士の内積を取った時に、その値が1になるように調整することである。つまり、

$$1 = 2 \int_{-\infty}^{\infty} \frac{\hat{h}_{\lambda}(f) \hat{h}_{\lambda}^*(f)}{S_n(|f|)} df \quad (5.3.1)$$

が成り立つような規格化である。ただし、 λ はテンプレート番号を指す。最後に、89行目以降において、テンプレート群と疑似的信号の内積、つまりマッチドフィルタを計算している。

このようにして計算された各テンプレートに対するSNRの中から最大のものを取り出し、その値が予め設定された閾値に対して大きいか否かで、リングダウン重力波の検定を行う。以上が、マッチドフィルタリングの実装である。

5.3.3 サーチパラメータ

本項では、本研究のマッチドフィルタリングにおけるテンプレート群のサーチパラメータについて記述する。

本研究において用いるサーチパラメータは先述の通り、連星合体後のブラックホールの質量、スピン、リングダウン重力波信号の到達時刻、連星をなす2天体の質量比（高調波を含めたサーチのため）、Inclinationの5つである。

本研究では、計算コストを下げるため、比較的小さいテンプレート空間を設定した。具体的には以下の通りである：

- massvector（合体後ブラックホールの質量）： $10[M_{\text{sun}}] \sim 100[M_{\text{sun}}]$ の範囲で $10[M_{\text{sun}}]$ 刻みの10パラメータ。
- kerrvector（合体後ブラックホールのスピン）：0.48 ～ 0.88の範囲で0.1刻みの5パラメータ。
- timevector（リングダウン重力波信号の到達時刻）：0.3[s] ～ 0.7[s]の範囲で0.1[s]刻みの5パラメータ。
- nuvector（連星をなす2天体の質量比）：0 ～ 0.25の範囲で0.05刻みの6パラメータ。
- thetavecvector（Inclination）：0, $\pi/4$, $\pi/2$, $5\pi/4$, $3\pi/2$ の5パラメータ。

合体後ブラックホールの質量に関しては、現在観測された重力波イベントにおける実際の質量を考慮し、大まかに設定した。また、スピンは重力波イベントGW150914における実際の値とその周辺値で設定した。Inclinationに関しては後の付録にて後述するが、包括的な設定値を用いた。

以上が、本研究のマッチドフィルタリングで用いるサーチパラメータである。

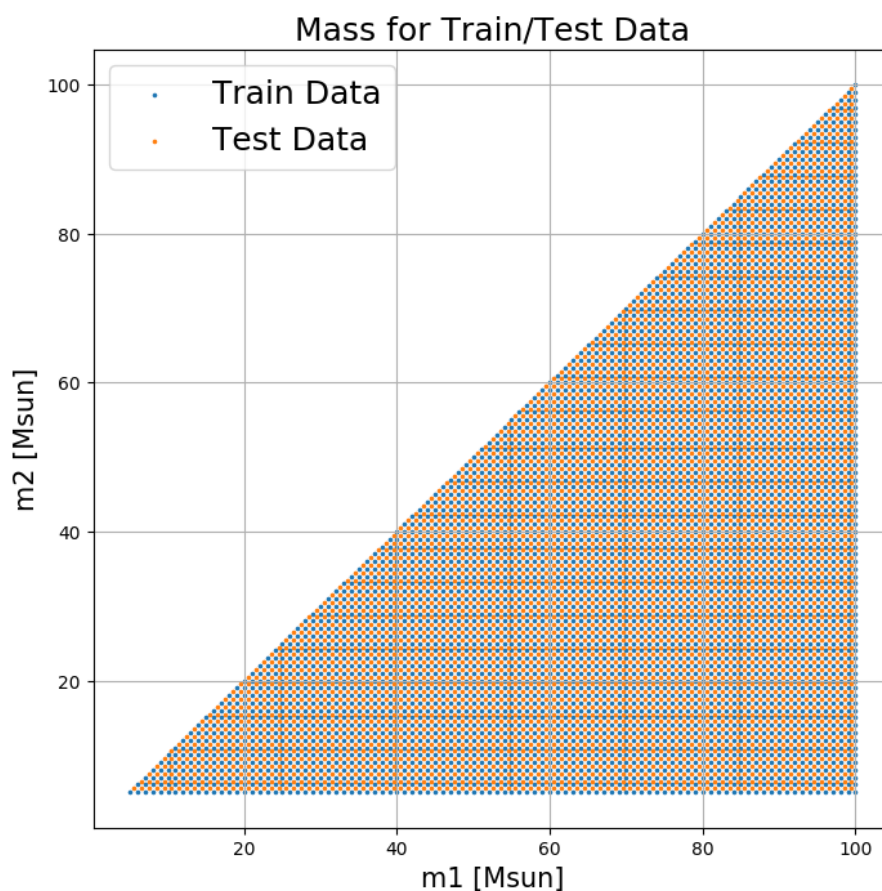


図5.4 トレインデータ・テストデータ用の連星質量

5.4 トレインデータ・テストデータの作成

本節では、畳み込みニューラルネットワークを学習させるためのトレインデータと、汎化能力の評価を行うためのテストデータの作成について記述する。

本研究では、トレインデータ・テストデータとして、連星をなす2天体の質量及びリングダウン重力波信号到達時間を様々な値に設定したものをを用いる。具体的には以下の通りである。

- 2天体の質量は、 $5[M_{\text{sun}}] \sim 100[M_{\text{sun}}]$ の範囲で $1.0[M_{\text{sun}}]$ 刻みをトレインデータ用に、 $5.5[M_{\text{sun}}] \sim 99.5[M_{\text{sun}}]$ の範囲で $1.0[M_{\text{sun}}]$ 刻みをテストデータ用に設定した。
- リングダウン重力波信号到達時間は、1秒分のデータにおいて $0.3[\text{s}] \sim 0.7[\text{s}]$ の範囲で $0.1[\text{s}]$ 刻みの5種類で設定した。

連星をなす2天体の質量の設定値を散布図としてプロットしたものが図5.4である。図を見るとわかる通り、テストデータはトレインデータに含まれないように、質量パラメータを設定している。これにより質

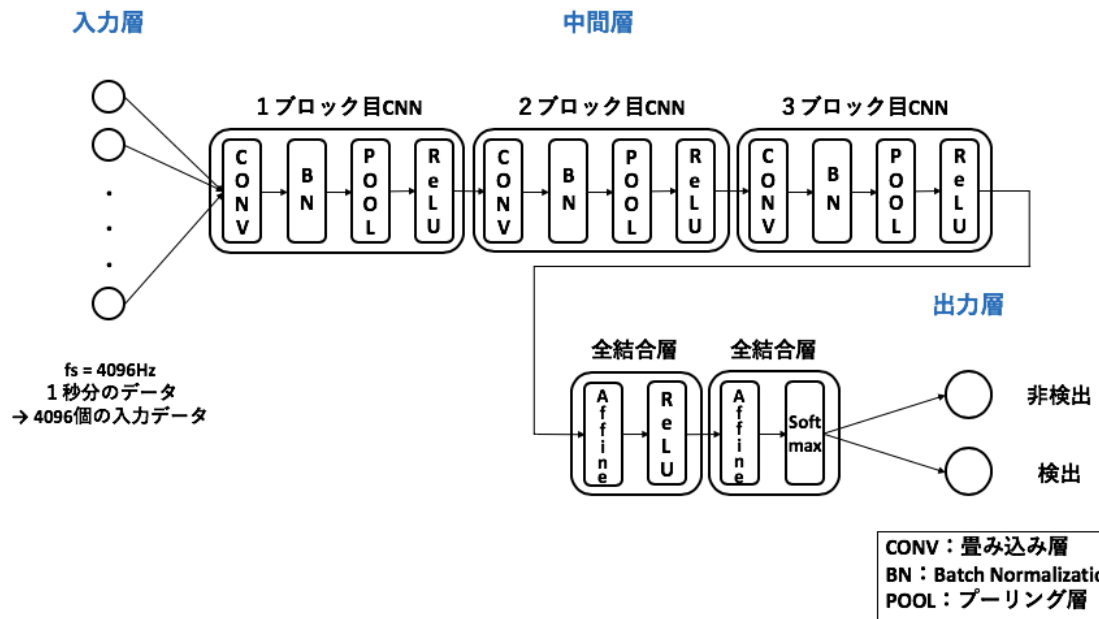


図5.5 本研究におけるネットワーク構造

量に関する汎化能力を検証することができる。

5.5 畳み込みニューラルネットワークの構造とその実装

本節では、本研究において用いた畳み込みニューラルネットワークの構造とその実装について記述する。

本研究で用いたネットワークの構造は図5.5の通りである。この構造は参考文献[8]をヒントに設計した。また、各畳み込み層及びプーリング層のハイパーパラメータを明示したものを図5.6に示す。

図5.5及び図5.6で示されるネットワークの実装ソースコードをソースコード5.5に示す。

ソースコード 5.5 network.py

```

1 import chainer.links as L
2 import chainer.functions as F
3
4
5 # CNNの1ブロック形成
6
7 class CNNBlock(chainer.Chain):
8     def __init__(self, in_channels, out_channels, ksize, stride):
9         super(CNNBlock, self).__init__()
10        with self.init_scope():
11            self.conv = L.ConvolutionND(ndim=1, in_channels=in_channels, out_channels=out_channels,
12                                       ksize=ksize, stride=stride)
13            self.bn = L.BatchNormalization(out_channels)
14
15    def __call__(self, x):
16        h = self.conv(x)
17        h = self.bn(h)
18
19    return h

```

Input	(1, 4096)	
Convolution① (f_num=16, k_size=9, stride=1)	(16, 4088)	
Batch Normalization	(16, 4088)	
Pooling① (k_size=4, stride=4)	(16, 1022)	
ReLU	(16, 1022)	
Convolution② (f_num=32, k_size=7, stride=1)	(32, 1016)	
Batch Normalization	(32, 1016)	
Pooling② (k_size=4, stride=4)	(32, 254)	
ReLU	(32, 254)	
Convolution③ (f_num=64, k_size=7, stride=1)	(64, 248)	
Batch Normalization	(64, 248)	
Pooling③ (k_size=4, stride=4)	(64, 62)	
ReLU	(64, 62)	
Flatten	(1, 3968)	f_num : フィルタ数
Affine	(1, 32)	k_size : フィルタサイズ
ReLU	(1, 32)	stride : ストライド
Affine(Output)	(1, 2)	(チャンネル数, 1次元データサイズ)

図5.6 本研究におけるネットワーク構造（詳細）

```

19
20 # CNN形成
21
22 class RingdownNet(chainer.Chain):
23     def __init__(self, class_labels=2):
24         super(RingdownNet, self).__init__()
25         with self.init_scope():
26             self.conv1 = CNNBlock(1, 16, 9, 1)
27             self.conv2 = CNNBlock(16, 32, 7, 1)
28             self.conv3 = CNNBlock(32, 64, 7, 1)
29             self.affine1 = L.Linear(None, 32)
30             self.affine2 = L.Linear(None, class_labels)
31
32
33     def __call__(self, x):
34         # 1 ブロック目CNN(channelが16に)
35         h = self.conv1(x)
36         h = F.max_pooling_nd(h, ksize=4, stride=4)
37         h = F.relu(h)
38
39         # 2 ブロック目CNN(channelが32に)
40         h = self.conv2(h)
41         h = F.max_pooling_nd(h, ksize=4, stride=4)
42         h = F.relu(h)
43
44         # 3 ブロック目CNN(channelが64に)
45         h = self.conv3(h)
46         h = F.max_pooling_nd(h, ksize=4, stride=4)
47         h = F.relu(h)
48
49         # 4 ブロック目全結合層

```

```
50     h = self.affine1(h)
51     h = F.relu(h)
52
53     # 出力層
54     h = self.affine2(h)
55
56     return h
```

本研究では、ネットワークの定義に際して、機械学習フレームワークであるChainerを使用した。

5行目から18行目にかけて、畳み込み層及びBatch Normalization層のかたまり（1ブロック）の定義を行っている。8行目から12行目にかけて、機械学習フレームワークであるChainerのChainを継承し、ブロックの初期化を行っている。このブロックは、入力データのチャンネル数、出力データのチャンネル数（フィルタ数）、畳み込み層のカーネルサイズ（フィルタサイズ）、畳み込み層のストライドを引数にもつ。なお、本研究で用いたネットワークのパディングは全て0である。14行目から16行目にかけて、このブロックが呼び出された（インスタンス化された）際のメインの記述となる。

20行目以降において、ネットワークの定義を行っている。22行目から30行目にかけて、Chainと上述のブロックを継承し、ネットワークの初期化を行っている。このネットワークは出力層のノード数（class.labels）を引数にもち、2が初期設定されている（「非検出」、「検出」の二値分類問題であるため）。この初期化メソッドにおいては、畳み込み層と全結合層（Affine層）の定義を行う必要がある。33行目以降において、このネットワークがインスタンス化された際のメインの記述となっている。上述のブロックに加え、プーリング層やReLU関数等の関数群に関する記述をする必要がある。なお、ここではソフトマックス関数の記述は省略している。

このように、ネットワークの定義の部分でハイパーパラメータの設定や構造の定義を行い、後にインスタンス化を行うことでネットワークの学習や推論を行うことができる。次節において、本節で定義したネットワークの学習について記述する。

5.6 ネットワークの学習

本節では、前々節で作成したトレインデータを用いて、前節で定義したネットワークの学習を行う。

5.6.1 学習戦略

本研究では、以下のような学習戦略をとった。

- トレインデータとして、リングダウン重力波信号を含むものと含まないものを1：1の割合で混ぜた。

- パラメータ更新手法として、Adamを用いた。
- 学習の初めにSNRが非常に高いデータを用いて学習を行い、次第に学習させる信号のSNRを下げていった。
- SNRの高いデータによる学習では、パラメータ更新の学習率を高めめに設定し、SNRの降下に従い学習率を下げていった。

2つ目のパラメータ更新手法の選択に関しては、Adamがより最短経路でパラメータ更新することができたからである。3、4つ目に関しては、SNRの低いデータを初めに学習させてしまうと、初期値のチューニングが困難であり、しばしばパラメータが発散してしまうためである。最終的におよそ20万個ほどのトレーニングデータを用いて学習を行った。

5.6.2 ネットワークの学習

ここで、ネットワークの学習に関するソースコードをソースコード5.6に示す。

ソースコード 5.6 train.py

```

1 import numpy as np
2 import cupy as cp
3 import chainer, random
4 import chainer.links as L
5 import chainer.functions as F
6 from chainer.dataset import convert
7 from chainer.datasets import TupleDataset
8 from chainer import serializers
9 from chainer.cuda import to_cpu
10
11
12 # GPUの設定
13
14 GPU = True
15 GPU_ID = 0
16
17 ##-----データのロードと加工-----##
18 # trainデータ、testデータのロード
19
20 PATH_TRAIN = 'PATH'
21 PATH_TEST = 'PATH'
22
23 x_train, _, t_train, _ = np.split(np.loadtxt(PATH_TRAIN, delimiter=','), [4096, 4099, 4100], axis=1)
24 x_test, _, t_test, _ = np.split(np.loadtxt(PATH_TEST, delimiter=','), [4096, 4099, 4100], axis=1)
25
26 # numpy -> cupy への変換(GPU対応)
27
28 x_train, t_train, x_test, t_test = cp.asarray(x_train), cp.asarray(t_train), cp.asarray(x_test), cp.asarray(
    t_test)
29
30
31 # ミニバッチの作成
32
33 train_data = TupleDataset(x_train, t_train)
34 test_data = TupleDataset(x_test, t_test)
35

```

```

36 MINIBATCH_SIZE = 128      # ミニバッチサイズの設定
37 train_iter = chainer.iterators.SerialIterator(train_data, MINIBATCH_SIZE)
38 test_iter = chainer.iterators.SerialIterator(test_data, MINIBATCH_SIZE)
39
40
41 ##-----ネットワークの学習-----##
42 # 学習モデル生成
43
44 model = RingdownNet()      # ネットワークのインスタンス化
45
46 if GPU:
47     chainer.cuda.get_device(GPU_ID).use()
48     model.to_gpu(GPU_ID)
49
50 # Optimizer生成
51
52 optimizer = chainer.optimizers.Adam(alpha=0.001)      #
53     OptimizerとしてAdamを設定 (alphaは学習率)
54 optimizer.setup(model)
55
56 # 学習 + 精度の確認
57
58 EPOCH = 100      # エポックの設定
59
60 train_loss_list = cp.zeros(EPOCH)
61
62 while train_iter.epoch < EPOCH:
63     batch = train_iter.next()
64     x_array, t_array = convert.concat_examples(batch, GPU_ID)
65     x = chainer.Variable(x_array.astype(cp.float32))      # Chainer変数への変換
66     t = chainer.Variable(t_array.astype(cp.int32))      # Chainer変数への変換
67
68     y = model(x)      # トレーンデータをネットワークに通す(順伝播)
69     loss_train = F.softmax_cross_entropy(y, t)      # 損失関数の計算
70     model.cleargrads()      # 勾配の初期化
71     loss_train.backward()      # 勾配をネットワークに通す(逆伝播)
72     optimizer.update()      # パラメータ更新
73
74 # テストデータによるモデルの汎化性能検証
75 if train_iter.is_new_epoch:
76
77     # ロスの表示
78     print('epoch:{:02d} train_loss:{:.06f}'.format(
79         train_iter.epoch, float(to_cpu(loss_train.data))), end='')
80     train_loss_list[train_iter.epoch-1] = loss_train.data
81
82     test_losses = []
83     test accuracies = []
84     while True:
85         test_batch = test_iter.next()
86         x_test_array, t_test_array = convert.concat_examples(test_batch, GPU_ID)
87         x_test = chainer.Variable(x_test_array.astype(cp.float32))      # Chainer変数への変換
88         t_test = chainer.Variable(t_test_array.astype(cp.int32))      # Chainer変数への変換
89
90         # testデータを順伝播
91         y_test = model(x_test)
92
93         # 損失関数の計算
94         loss_test = F.softmax_cross_entropy(y_test, t_test)
95         test_losses.append(to_cpu(loss_test.array))

```

```
96
97     # 精度の計算
98     accuracy = F.accuracy(y_test, t_test)
99     accuracy.to_cpu()
100     test_accuracies.append(accuracy.array)
101
102     if test_iter.is_new_epoch:
103         test_iter.reset()
104         break
105
106     print('test_loss:{:.04f} test_accuracy:{:.06f}'.format(
107         np.mean(test_losses), np.mean(test_accuracies)))
108
109 print('Training is finished')
110
111
112 ##-----学習済みパラメータの保存-----##
113
114 model.to_cpu()
115 SAVE_MODEL = 'PATH'
116 serializers.save_npz(SAVE_MODEL, model)
117 print('Saved')
```

1 2 行目から 1 5 行目にかけて、Google ColaboratoryにおけるGPUの設定を行っている。

1 7 行目から 3 8 行目にかけて、トレインデータ及びテストデータのロードと加工を行っている。具体的には 2 8 行目において、ロードしたデータセットのデータ型をNumpy配列からCupy配列に変換している。ネットワークの学習はGPU上で行われるため、Numpy配列をGPU対応のCupy配列に変換する必要があるからである。また、3 1 行目から 3 8 行目において、トレインデータ及びテストデータに対してミニバッチを作成している。

4 1 行目以降において、ネットワークの学習及びその評価に関する記述がなされている。4 4 行目において、ソースコード5.5で定義されたネットワーククラスをインスタンス化している。5 2 行目から 5 3 行目にかけて、パラメータ更新手法のセットアップを行っている。ここでは更新手法としてAdamを用いており、引数で学習率をはじめとするパラメータの設定を行うことができる。6 1 行目から 7 1 行目にかけて、トレインデータによる学習の 1 エポック分の記述がなされている。具体的には以下のような記述がなされている：

- ミニバッチを受け取り、それを学習データと正解データに分ける。(6 2、6 3 行目)
- データセットをChainer変数（Chainerで定義されたネットワークに入力するデータ）に変換し、トレインデータをネットワークに順伝播させる。(6 4 行目から 6 7 行目にかけて)
- ネットワークの出力値と正解データを用いて、損失関数を計算する。(6 8 行目)
- ネットワークの勾配をリセットする。(6 9 行目)
- 勾配をネットワークに対して逆伝播させ、パラメータを更新する。(7 0、7 1 行目)



図5.7 学習経過の様子（損失関数）



図5.8 学習経過の様子（精度）

74行目から107行目にかけて、学習が1エポック分終了するたびに行われる、テストデータを用いた評価に関する記述がなされている。ここでは、損失関数の値（ロス）及び学習精度（二値分類の正誤）を記録してある。

上記ステップを設定されたエポック分行うことで、ネットワークの学習が完了する。最後に、112行目以降において、学習済みネットワークのパラメータを保存している。

以上ソースコードによって、ネットワークの学習を行う。実際に行った学習とその汎化能力検証の様子を図5.7及び図5.8に示す。

両図における赤線はトレインデータのデータセットによるものであり、緑線はテストデータのデータセットによるものである。また横軸はエポック数であり、ここでは30エポック分の学習を行っている。図5.7を見ると、エポックの経過に伴って、トレインデータ及びテストデータの損失関数は減少していることがわかる。また、図5.8における精度とは、ネットワークによる二値分類の正誤をパーセント表示したものである。エポックの経過に伴って、トレインデータ及びテストデータによる推論の精度が上昇していることがわかる。このトレインデータに対する精度は100%に到達しており、一方このテストデータに対する精度は最終的に99%に到達している。

先述の通り、このような学習をSNRの大きいデータセットから始めて、徐々にSNRの低いデータセットによる学習を行っていった。学習されたネットワークを用いて、次章ではリングダウン重力波のデータ解析を行う。

第6章

解析結果と考察

前章では、リングダウン重力波のデータ解析におけるマッチドフィルタリングの実装と、畳み込みニューラルネットワークの構造・実装・学習について記述した。

本章では、マッチドフィルタリング及び学習済みの畳み込みニューラルネットワークを用いた、リングダウン重力波の解析結果について記述したのち、その結果に対する考察を記述する。

6.1 $29M_{\text{sun}}$ と $36M_{\text{sun}}$ のブラックホール連星からのリングダウン重力波における解析

ここでは、初の重力波直接観測となったイベント、GW150914のパラメータ（天体の質量、スピン）を用いたリングダウン重力波の解析結果を記述する。

はじめに、重力波検出の二値分類における評価指標（精度の観点において）について記述する。評価指標

重力波検出の二値分類		正解 (データのラベル)	
		重力波あり	重力波なし
予想 (推論による結果のラベル)	重力波あり	真陽性 (TP)	偽陽性 (FP)
	重力波なし	偽陰性 (FN)	真陰性 (TN)

図6.1 重力波検出の二値分類

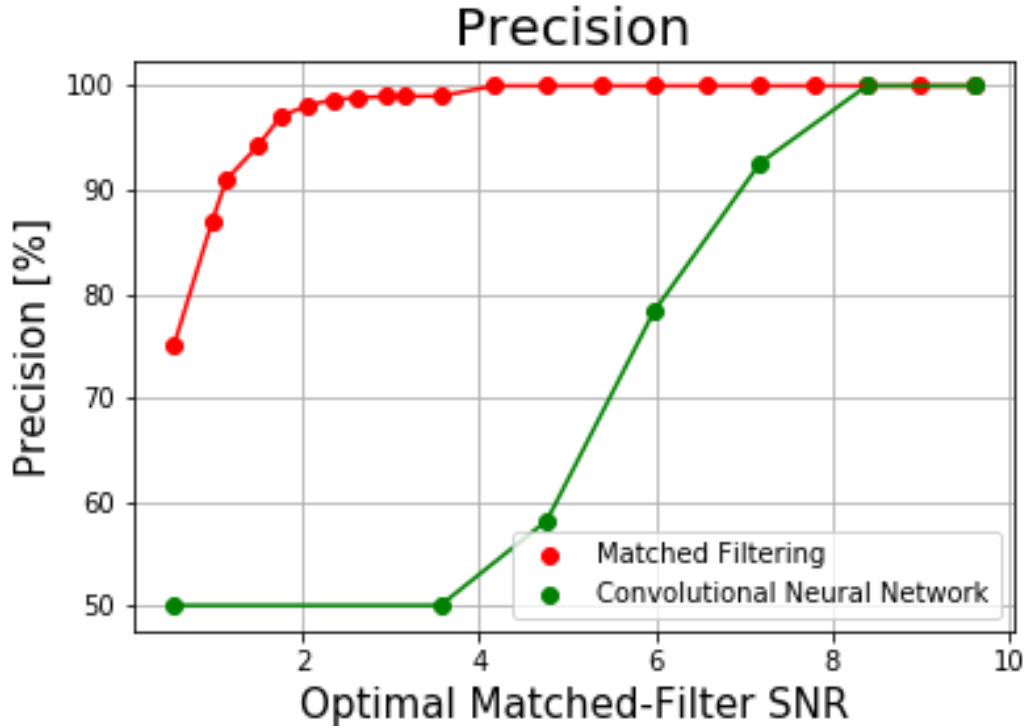


図6.2 29M_{sun}と36M_{sun}のブラックホール連星合体によるリングダウン重力波における適合率

を記述するにあたって、二値分類における統計用語を図6.1に示す。本研究において用いる評価指標は、適合率（Precision）及び敏感度（Sensitivity）である。それぞれの指標は以下の式で表される：

$$Precision = \frac{TP}{TP + FP} \quad (6.1.1)$$

$$Sensitivity = \frac{TP}{TP + FN} \quad (6.1.2)$$

ここで、 TP, FP, TN はそれぞれ図6.1に示されているものに対応している。適合率は「重力波信号を含むと推論したものの中の正解率」を表しており、敏感度は「重力波信号を含むと推論しなければならないデータの中で、どのくらい検出として推論できているかを示す指標」である。

ここで上記の設定における、マッチドフィルタリング及び畳み込みニューラルネットワークによる解析結果を図6.2、図6.3に示す。

横軸はOptimal Matched-Filter SNR（以下、SNR）であり、テストしたリングダウン重力波の振幅の大きさに相当する量である。縦軸はそれぞれ適合率と再現率である。

両結果を通して言えることは、SNRが8.0付近以上では両手法ともに適合率・再現率が100%であるが、それ以下の帯域ではマッチドフィルタリングが完全に優勢である。特に、適合率においては、両手法で圧倒的な差が見受けられる。これは、式(6.1.1)右辺の分母を見るとわかる通り、偽陽性が高いことが原因であると考えられる。SNRの低い帯域において適合率と再現率の比較から、本研究におけるネットワークは、「重

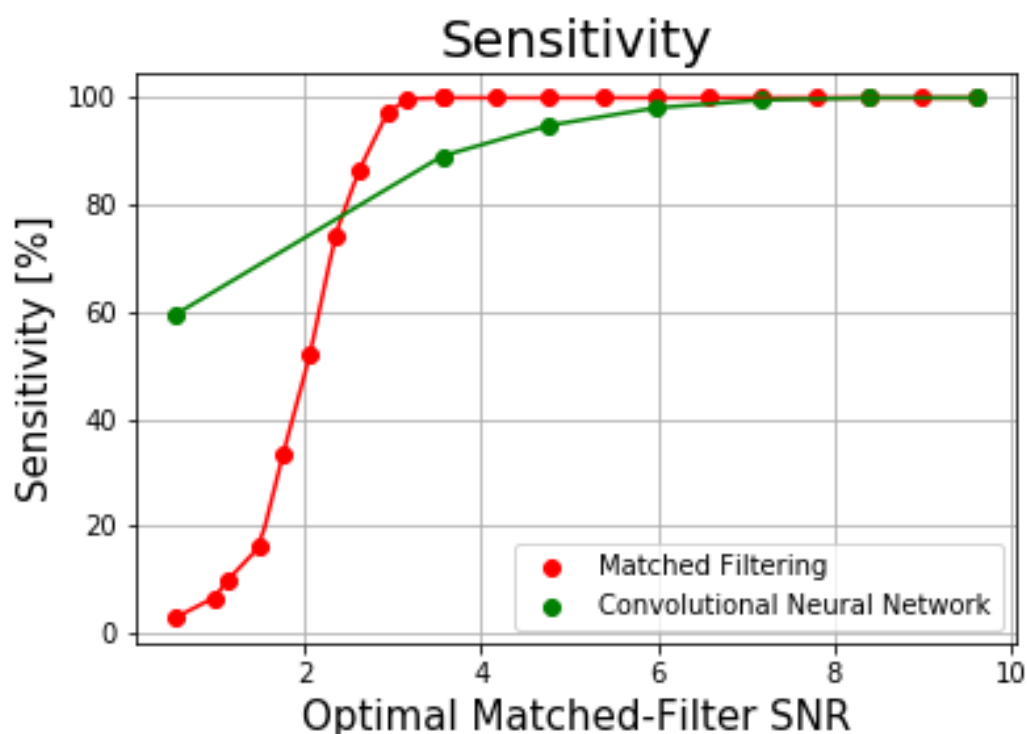


図6.3 $29M_{\text{sun}}$ と $36M_{\text{sun}}$ のブラックホール連星合体によるリングダウン重力波における再現率

力波信号を含んでいない」データに対する認識精度が低いと考えられる。このように、本研究の2つの論点のうちの1つである「精度」に関してはいい結果を得ることができなかった。

一方、もう1つの論点である計算コストに関して、単純に解析にかかる時間を尺度として考えた時、マッチドフィルタリングの解析には1データあたり約1.5[s]かかり、畳み込みニューラルネットワークの解析には1データあたり約12.5[ms]かかった。つまり、畳み込みニューラルネットワークでは $1.5[\text{s}]/12.5[\text{ms}] = 120$ 倍ほどの速さで解析を行うことができる。つまり、重力波イベントに対する光学的/近赤外線フォローアップ観測において重要視されるローレイテンシー（Low Latency）解析では有用な解析手法であると考えられる。

このように、 $29M_{\text{sun}}$ と $36M_{\text{sun}}$ のブラックホール連星からのリングダウン重力波における解析では、高いSNRの重力波のみに対してはマッチドフィルタリングに取って代わることができるが、それ以外では精度の観点で厳しい結果が得られた。

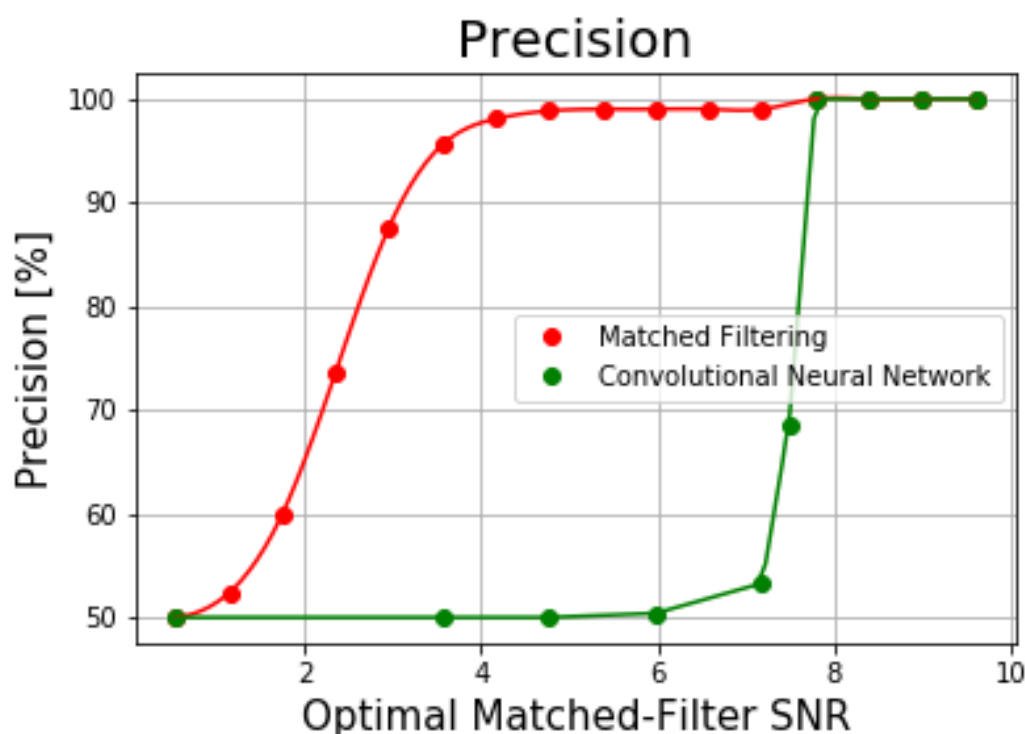


図6.4 1.4 M_{sun} と5 M_{sun} の中性子星・ブラックホール連星合体によるリングダウン重力波における適合率

6.2 1.4 M_{sun} と5 M_{sun} の中性子星・ブラックホール連星からのリングダウン重力波における解析

ここでは、中性子星とブラックホールからなる連星の合体において放出される、リングダウン重力波の解析結果を記述する。中性子星の質量の代表的な値である1.4 M_{sun} は、マッチドフィルタリングのサーチパラメータ外であり、また、畳み込みニューラルネットワークのトレインデータの範囲外である。このような場合においてどのような結果が得られるのか検証した。

ここで上記の設定における、マッチドフィルタリング及び畳み込みニューラルネットワークによる解析結果を図6.4、図6.5に示す。

まずマッチドフィルタリングにおける結果を考察すると、前節の設定と比較して適合率と再現率ともに低くなっていることがわかる。これは、テンプレート群から外れた範囲のデータであるため、偽陰性と偽陽性が高くなってしまったことが原因である。

一方、SNRが4～7程度の帯域で、適合率と再現率がともに低くなっていることがわかる。これは、本設定がトレインデータの範囲外であることから、この帯域で過学習が起きているのではないかと考える。

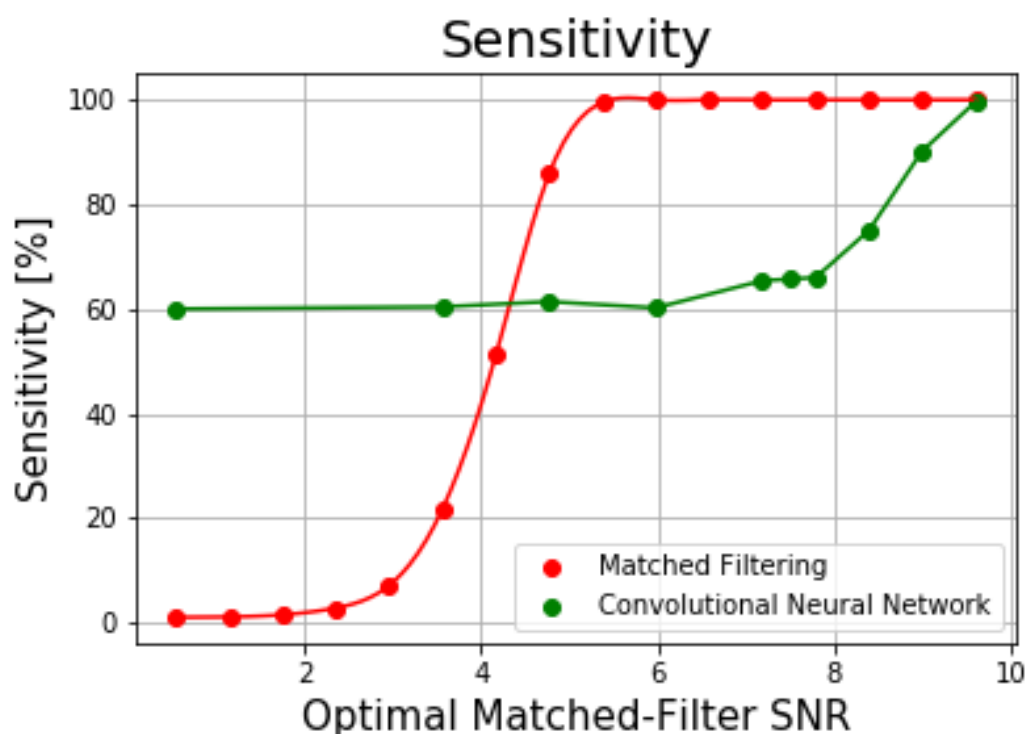


図6.5 1.4 M_{sun} と5 M_{sun} の中性子星・ブラックホール連星合体によるリングダウン重力波における再現率

このように、本設定においても、本研究におけるネットワークがマッチドフィルタリングに取って代わることは、精度の観点で厳しいという結論が得られた。

6.3 結果のまとめと考察

本研究における結果を再度まとめると以下の通りである：

- 29 M_{sun} と36 M_{sun} のブラックホール連星合体によるリングダウン重力波の解析では、低いSNRに対する精度が低く、高いSNRに対してはマッチドフィルタリングに取って代わる精度が得られた。
- 1.4 M_{sun} と10 M_{sun} の中性子星・ブラックホール連星からのリングダウン重力波における解析では、SNRが4～7程度の帯域で適合率と再現率がともに低くなっており、過学習が見られた。
- 計算コストの観点では、畳み込みニューラルネットワークを用いると、マッチドフィルタリングと比較してより少ない計算時間での推論を行うことができた。

以上の結果より、今後畳み込みニューラルネットワークによる解析がマッチドフィルタリングによる解析に取って代わるためには、精度の向上がマストである。そこで、精度を向上させるための解決策を考えるにあたって、論点を整理する。図6.6に論点を整理したものを示す。

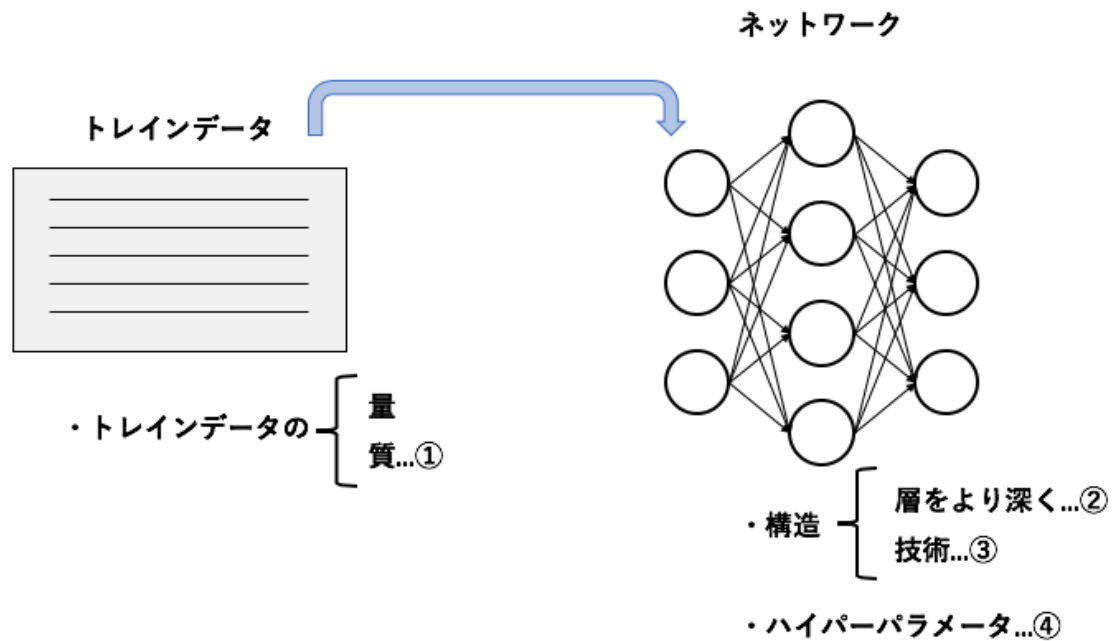


図6.6 論点整理

ナンバリングされた各論点に対して以下で言及する。

1つ目は①トレーニングデータの質である。本研究ではネットワークの学習において、高いSNRのトレーニングデータから低いSNRのトレーニングデータへとスライド学習させたが、SNRの刻み幅を大きく設定していた。仮に、ネットワークが学習した特徴量の1つにSNRが含まれるとすれば、学習させるSNRの刻み幅が大きいことは問題である。そのため、将来的にはさらに細かいSNRの刻み幅でスライド学習させる必要がある。

2つ目は②層構造である。仮に、トレーニングデータの量に対して、特徴量を表現するためのパラメータ数が十分でなかった場合、層を深くすることによって精度は向上すると考えられる。実際に、参考論文[8]では畳み込み層とプーリング層をそれぞれ3層から5層に増やしたことで、認識精度は向上している。

一方、層構造に関して、プーリング層を過度に組み込ませたことによって、特徴量が減少している可能性が考えられる。参考論文[8]ではプーリング層を数多く組み込ませているが、最近の畳み込みニューラルネットワークを用いた画像認識等の分野では、プーリング層を全く組み込まない構造が流行している。そのため、プーリング層を組み込まない構造を試す価値はある。まとめると、ネットワークの構造の見直しは検討する必要がある。

3つ目は③新たなテクニックである。精度が向上しなかった原因の1つとして、パラメータ空間における損失関数の（悪い）極小点に最適化された可能性が考えられる。通常、精度が良ければ極小点に最適化

されることは問題とならないが、本研究の場合は問題として考えられる。その解決策として、参考論文[9]に示された手法が有効であるかもしれない。この手法は、「二値分類問題であること」、「特殊な損失関数を用いること（クロスエントロピーは不可）」の2つを前提条件に、Exponential neuronと呼ばれる特別なニューロンを1つ加えるという手法である。この手法によって、そのニューラルネットワークの局所解は大域最適な解となることが主張されている。しかし、この手法は数学的にのみ証明されており、実験的には証明されていない。そのため、実装さえ可能であれば試す価値はある。

4つ目は④ハイパーパラメータである。より精度の出るハイパーパラメータにチューニングすることで、精度の向上が見込める。そのため、本研究ではマニュアルにハイパーパラメータのチューニングを行ってきたが、時間がかかるがグリッドサーチやランダムサーチなどで機械的にハイパーパラメータのチューニングを行うことが1つの解決策であると考えられる。

以上が本研究の結果に対する考察であり、今後の課題である。

付録A：Inclinationによる寄与

ここでは、Inclinationによる、QNM波の各モードに対する寄与について記述する。

スピンウェイト球面調和関数 ${}_sY_{lm}(\iota, \phi)$ は、球面調和関数 $Y_{lm}(\iota, \phi)$ を用いて定義されるが、定義に用いる演算子を先に定義しておく。スピンウェイト s の関数 $\eta(\iota, \phi)$ に対する、演算子 $\hat{\alpha}$ と $\hat{\beta}$ の応答を以下で定義する：

$$\hat{\alpha}\eta = -(\sin\theta)^s \left[\frac{\partial}{\partial\theta} + i \csc\theta \frac{\partial}{\partial\phi} \right] (\sin\theta)^{-s} \eta \quad (\text{A.1})$$

$$\hat{\beta}\eta = -(\sin\theta)^{-s} \left[\frac{\partial}{\partial\theta} - i \csc\theta \frac{\partial}{\partial\phi} \right] (\sin\theta)^s \eta \quad (\text{A.2})$$

実はこの二つの演算子は、スピンに対する昇降演算子のような役目を果たしている。演算子 $\hat{\alpha}$ は、作用させた関数のスピンウェイトを1上昇させ、一方演算子 $\hat{\beta}$ はスピンウェイトを1下降させる働きがある。

また、スピンウェイト s の球面調和関数は以下のように定義される：

$${}_sY_{lm}(\iota, \phi) = \begin{cases} \sqrt{\frac{(l-s)!}{(l+s)!}} \hat{\alpha}^s Y_{lm}(\iota, \phi) & (0 \leq s \leq l) \\ \sqrt{\frac{(l+s)!}{(l-s)!}} (-1)^s \hat{\beta}^s Y_{lm}(\iota, \phi) & (-l \leq s \leq 0) \end{cases} \quad (\text{A.3})$$

なお、 $|s| > l$ の物には定義されない。

計算の見通しをよくするため、以下の式で表現されるものを用いる：

$${}_sY_{lm}(\iota, \phi) = (-1)^s \sqrt{\frac{2l+1}{4\pi}} d_{m(-s)}^l(\iota) \exp im\phi \quad (\text{A.4})$$

ここで、 $d_{mn}^l(\iota)$ はWigner' s small d-matrixであり、次のように定義される：

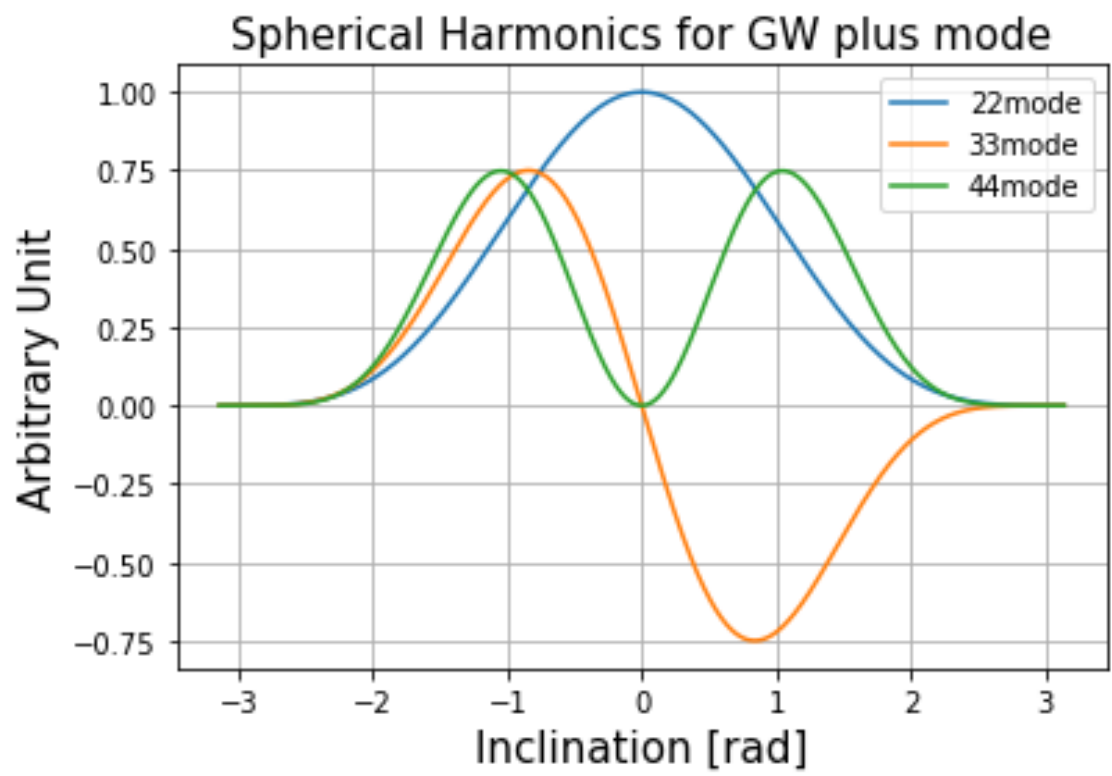
$$d_{mn}^l(\iota) = (-1)^{(l-m)} \sqrt{(l+m)!(l-m)!(l+n)!(l-n)!} \\ \times \sum_k \frac{(-1)^k (\sin(\iota/2))^{2l-m-n-2k} (\cos(\iota/2))^{m+n+2k}}{k!(l-m-k)!(l-n-k)!(m+n+k)!} \quad (\text{A.5})$$

また、次の性質が成り立つ：

$$(-1)_n^{n+m} Y_{lm}^*(\iota, \phi) = {}_{-n} Y_{l-m}(\iota, \phi) \quad (\text{A.6})$$

これらを用いて、式(2.2.16)の計算を行う。横軸をInclinationとした、各モードの寄与は図A.0のようになる。

このように、Inclinationの値によって、各モードに対する寄与の大きさが異なることがわかった。



図A.0 各モードのInclinationによる寄与

参考文献

- [1] 熱田将 「ブラックホールリングダウンからの重力波における高調波を含めた解析」 東京工業大学大学院修士論文 (2017)
- [2] Micheal P. Ryan, Jr 「Teukolsky equation and Penrose wave equation」 Phys. Rev. D 10, 1736 (1974)
- [3] Saul A. Teukolsky 「Rotating Black Holes: Separable Wave Equations for Gravitational and Electromagnetic Perturbations」 Phys. Rev. Lett. 29, 1114 (1972)
- [4] Fernando Echeverria 「Gravitational-wave measurements of the mass and angular momentum of a black hole」 Phys.Rev.D 40 10 (1989)
- [5] Emanuele Berti,Vitor Cardoso,and Clifford M.Will 「On gravitational wave spectroscopy of massive black holes with the space interferometer LISA」 Phys.Rev.D73 064030 (2006)
- [6] S. Gossan, J. Veitch, and B. S. Sathyaprakash 「Bayesian model selection for testing the no-hair theorem with black hole ringdowns」 Phys. Rev. D85, 124056 (2012)
- [7] LIGO Scientific Collaboration, " LIGO Open Science Center release of GW150914" , 2016, DOI 10.7935/K5MW2F23
- [8] Daniel George and E. A. Huerta 「Deep Neural Networks to Enable Real-time Multimessenger Astrophysics」 Phys.Rev.D97, 044039 (2018)
- [9] Shiyu Liang, Ruoyu Sun, Jason D. Lee, and R. Srikant 「Adding One Neuron Can Eliminate All Bad Local Minima」 arXiv:1805.08671 (2018)

謝辞

様々な方のご協力のもとに、本論文を書くことができたと心の底から感じています。この場を借りて感謝の気持ちを書き連ねたいと思います。

指導教員である宗宮先生には、私が学部時代に宗宮研究室に所属してから3年間お世話になりました。重力波検出器における物理の醍醐味や検証により得られた結果に対する考察の仕方などを教えていただいたり、物理の枠に留まらない柔軟な発想を目の当たりにして、様々な知見を得ることができました。また、ゼミなどの活動を通して、資料作成力やプレゼンテーション能力など、ジェネラルなスキルも得ることができました。何より、光学実験がメインである本研究室において、データ解析を研究テーマとして認めていただきありがとうございました。宗宮研究室で作られた楽しい思い出は宗宮先生のおかげです。

福岡大学の端山先生には、データ解析の初歩から、重力波におけるデータ解析についてご教授いただきました。端山先生と行った数々のディスカッションは、現在の自身のスキルを形作る良い経験となりました。それだけでなく、釣りに連れて行ってくださったり、楽しい思い出をありがとうございました。

また、研究室の方々への感謝も欠かすことはできません。藤本先生にはゼミの際に、専門的かつ高度な質問を投げかけてくださり、勉強になることが非常に多かったです。

秘書の前川さんには、基礎の事務員のときから事務関係でお世話になり、感謝しております。

同期の草柳くんには、研究でお互い切磋琢磨しあった他に、普段の生活の中で私のくだらない話にも耳を傾けてくださり、非常に感謝しています。

修士1年の小田部くん、佐々木くん、中島くん、学部4年の井上くんはゼミの際に、多角的な視点から質問を投げかけてくださり、いい意味で私に緊張感を与えてくれました。

元宗宮研究室所属の留学生であるメロディーさんには、英会話の練習やPythonの基礎を教えていただいたり、感謝しています。メロディーさんが様々なイベントを企画してくれたおかげで、より研究室に楽しい空気が流れ込んだと思います。

現在宗宮研究室所属の留学生であるリュウくんには、ゼミでの質問や、カードゲーム遊びなどでお世話に

なりました。

他にも宗宮研究室OBOGの方々や、KAGRAに関わっていた先生や学生の方々など、感謝の気持ちを述べたい人を挙げればキリがありません。そういった様々な方々の協力を経て、現在の自身があると再確認いたしました。

そして何よりも感謝しているのが家族です。大学院進学という選択にも理解を示していただき、生まれてから現在に至るまで支えていただいたことに心より感謝しています。