

卒業論文

ニューラルネットワークを用いた 重力波の到来方向推定

東京工業大学理学院物理学系 宗宮研究室

笹岡聖也

2022年2月14日

概要

マルチメッセンジャー天文学において重力波の検出とその到来方向推定を低遅延で行うことが重要であるが、コンパクト連星合体からの重力波のデータ解析に用いられているマッチドフィルタは計算コストが大きい。機械学習は推論の速さと精度の高さから、その問題を解決する新たなデータ解析手法としての可能性を秘めている。

本研究では、ニューラルネットワークを用いて重力波の到来方向推定を行った。1次元の畳み込みに基づくモデルを Chatterjee らが 2019 年に発表した先行研究の手法と組み合わせることにより、先行研究より推定精度が向上した。次に、この新たな手法を用いて LIGO, Virgo の 3 台の検出器による推定と KAGRA を加えた 4 台の検出器による推定を行い、KAGRA を加えることにより精度が向上することを確認した。最後に、実際に観測された重力波を模倣したデータを用いて検証を行い、新たに提案した手法が既存の手法より遥かに短い時間で到来方向推定ができることを確認した。

目次

第 1 章	はじめに	1
第 2 章	重力波	3
2.1	重力波の方程式	3
2.2	重力波の発生	7
2.3	連星系からの重力波	13
第 3 章	重力波の検出	18
3.1	Michelson 干渉計	18
3.2	アンテナパターン	20
3.3	重力波検出器の雑音	25
第 4 章	重力波のデータ解析	27
4.1	パワースペクトル密度	27
4.2	ガウシアンノイズ	28
4.3	マッチドフィルタ	29
4.4	パラメータ推定	31
4.5	到来方向推定	31
4.6	信号処理	32
第 5 章	ニューラルネットワーク	35
5.1	教師あり学習	35
5.2	最適化手法	36
5.3	多層パーセプトロン (MLP)	39
5.4	畳み込みニューラルネットワーク (CNN)	39
5.5	Temporal Convolutional Network (TCN)	44
第 6 章	重力波の到来方向推定	46
6.1	データ生成	47
6.2	分割方法	49
6.3	推定手法	51
第 7 章	結果	56

目次	iii
7.1 3 台の検出器による推定結果	56
7.2 4 台の検出器による推定結果	59
7.3 GW170814 データでの検証	61
第 8 章 結論	64
謝辞	65
参考文献	65
付録 A 開発環境	72
付録 B ソースコード	73

第 1 章

はじめに

重力波は時空の歪みが光速で伝播する現象である。それは一般相対性理論における Einstein 方程式の波動解であり、1916 年に A. Einstein によってその存在が予言された [1, 2]。重力波の特徴的な性質は透過性が高いことである。その性質から、重力波の観測によって従来の電磁波観測で得ることのできない高密度の天体内部の情報や電磁波を発しない天体の情報を得ることができる。

重力波の振幅は非常に小さいため直接的な検出を行うことは難しく、約 100 年間それは達成されなかったが、2015 年 9 月 14 日にアメリカの重力波検出器 LIGO [3] によって初めて重力波が直接検出された [4]。GW150914 と命名されたそのイベントは連星ブラックホール合体から発生した重力波であった。2017 年には LIGO の 2 つの検出器 H1, L1 とイタリアの Virgo [5] の計 3 台の検出器による同時観測が始まり、同年に初めて連星中性子星合体からの重力波 GW170817 が検出された [6]。この重力波は、検出後の追観測により可視光や X 線、電波などの各波長で重力波対応天体の残光が観測され、マルチメッセンジャー天文学の幕開けとなるイベントとなった [7]。重力波の初検出から 6 年の間に 3 回の長期観測 (O1, O2, O3) が行われ、LIGO と Virgo によって 90 個のコンパクト連星合体からの重力波が検出された [8]。岐阜県神岡にある日本の重力波検出器 KAGRA [9] は 2020 年 4 月にドイツの GEO600 [10] と 2 週間の共同観測を行った。次の長期観測 (O4) から LIGO, Virgo, KAGRA の計 4 台の検出器による同時観測が行われる。

現在、コンパクト連星合体からの重力波の検出にはマッチドフィルタが用いられている。それは観測データとテンプレートと呼ばれる理論的に計算された波形の相関をとるという方法であり、高精度で重力波を検出することができるが、欠点が 2 つある。1 つ目はマッチドフィルタは定常なガウシアンノイズに対してのみ最適であるということである。一般に検出器のノイズには非定常または非ガウシアンなノイズも含まれるため、マッチドフィルタは最適な手法とは言えない。2 つ目は計算コストが大きいことである。マッチドフィルタを用いると重力波が到来してアラートを出すまでに数十秒の時間がかかる [11] ため、電磁波による追観測を迅速に行うためには、より効率的な手法が必要になる。

機械学習は一度モデルを学習すれば新たなデータに対しての推論速度が速いことや、未知のデータに対する汎化性能が高いことが特徴であり、画像認識や音声認識などの分野で近年顕著な成果を上げている。2018 年の D. George と E. A. Huerta による論文 [12] を皮切りに、機

機械学習は重力波のデータ解析にも応用されている。例えば、重力波探索ではコンパクト連星合体からの重力波だけでなく、超新星爆発からの重力波 [13–16] や連続重力波 [17–21], 確率的背景重力波 [22] にも応用されている。また、パラメータ推定 [23, 24], ノイズ除去 [25, 26], 突発性雑音の分類 [27–31] などにも機械学習を応用する研究が行われている。

機械学習を用いた重力波の到来方向推定は C. Chatterjee ら [32] によって初めて行われた。彼らは天空をいくつかのセクタに分割し、連星ブラックホール合体からの重力波がどのセクタから到来してきたかという分類問題として多層パーセプトロン (MLP) を学習し、到来方向の推定を行った。MLP の入力には LIGO H1, LIGO L1, Virgo のデータから抽出した 7 種類の特徴量を用いた。さらに、彼らは実際に観測された重力波のパラメータを用いてシミュレーションした波形での検証を行い、BAYESTAR [33] で用いられているアルゴリズムを使用して 90% 信頼区間を構成した。

宗宮研究室 OB の Y. Liu [34] は畳み込みニューラルネットワーク (CNN) を用いて重力波の検出と到来方向推定を行った。到来方向推定においては LIGO と Virgo に加えて KAGRA のデータも用いた点が C. Chatterjee らの研究と異なる。KAGRA のデータを加えることで推定精度は向上すると期待されたが、LIGO と Virgo のみによる推定よりも精度が悪くなるという結果が得られた。

本研究では、C. Chatterjee ら [32] の方法、Temporal Convolutional Network (TCN) [35], この 2 つを組み合わせた手法の計 3 通りを考え、精度の向上を試みた。また、複数の分割方法や分割数を試し、精度の比較を行った。次に、3 通りの手法のうち最も精度の高い手法を用いて LIGO と Virgo の 3 台の検出器のデータを用いた場合と、KAGRA を含む 4 台の検出器のデータを用いた場合の精度の比較を行い、KAGRA のデータを用いることで精度が向上することを確かめた。最後に、初めて 3 台の検出器で同時観測された重力波である GW170814 [36] のパラメータを用いて波形をシミュレーションし、学習したモデルの性能を検証した。

本論文の構成は以下の通りである。第 2 章では Einstein 方程式の線形化から始め、重力波の理論について概説する。続く第 3 章では重力波の検出方法や重力波検出器の雑音について述べ、第 4 章ではマッチドフィルタを中心に重力波のデータ解析の方法について概説する。第 5 章では本研究で用いたニューラルネットワークの理論について述べる。第 6 章で本研究で用いた到来方向推定の手法について述べ、第 7 章でその結果を紹介する。最後に第 8 章で本研究をまとめる。

第 2 章

重力波

本章では、まず Einstein 方程式を線形近似することによって重力波が満たす方程式を導出する。次に重力波解の性質や重力波の発生および重力波が運ぶエネルギーについて述べ、観測が期待されている重力波源を紹介する。最後に、ニュートン近似により円軌道を描く連星系からの重力波の振幅を導出する。本章は Maggiore [37] の第 1 章から第 4 章を参考にした。

2.1 重力波の方程式

2.1.1 Einstein 方程式

4 次元時空間での微小距離 ds^2 は計量テンソル $g_{\mu\nu}$ を用いて

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu \quad (2.1)$$

と表せる。平坦な時空での計量テンソルは $\eta_{\mu\nu} = \text{diag}[-1, 1, 1, 1]$ である。計量テンソル $g_{\mu\nu}$ は Einstein 方程式

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \frac{8\pi G}{c^4}T_{\mu\nu} \quad (2.2)$$

に従う。 $T_{\mu\nu}$ はエネルギー運動量テンソルであり、Christoffel 記号 $\Gamma_{\mu\nu}^\rho$ 、Riemann テンソル $R_{\nu\rho\sigma}^\mu$ 、Ricci テンソル $R_{\mu\nu}$ 、Ricci スカラー R はそれぞれ次のように定義される。

$$\Gamma_{\mu\nu}^\rho := \frac{1}{2}g^{\rho\sigma}(\partial_\mu g_{\sigma\nu} + \partial_\nu g_{\sigma\mu} - \partial_\sigma g_{\mu\nu}) \quad (2.3)$$

$$R_{\nu\rho\sigma}^\mu := \partial_\rho \Gamma_{\nu\sigma}^\mu - \partial_\sigma \Gamma_{\nu\rho}^\mu + \Gamma_{\alpha\rho}^\mu \Gamma_{\nu\sigma}^\alpha - \Gamma_{\alpha\sigma}^\mu \Gamma_{\nu\rho}^\alpha \quad (2.4)$$

$$R_{\mu\nu} := R_{\mu\rho\nu}^\rho \quad (2.5)$$

$$R := g^{\mu\nu} R_{\mu\nu} \quad (2.6)$$

2.1.2 Einstein 方程式の線形化

弱い重力場を考え、計量テンソル $g_{\mu\nu}$ を平坦な時空での計量テンソル $\eta_{\mu\nu}$ と摂動項 $h_{\mu\nu}$ の和で表す：

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \quad (2.7)$$

$|h_{\mu\nu}| \ll 1$ とし, $h_{\mu\nu}$ の2次以上の項を無視すると Christoffel 記号, Ricci テンソル, Ricci スカラーはそれぞれ次のように表せる.

$$\Gamma_{\mu\nu}^{\rho} = \frac{1}{2}\eta^{\rho\sigma}(\partial_{\mu}h_{\sigma\nu} + \partial_{\nu}h_{\sigma\mu} - \partial_{\sigma}h_{\mu\nu}) \quad (2.8)$$

$$R_{\mu\nu} = \frac{1}{2}\eta^{\rho\sigma}(\partial_{\mu}\partial_{\rho}h_{\sigma\nu} + \partial_{\nu}\partial_{\sigma}h_{\mu\rho} - \partial_{\rho}\partial_{\sigma}h_{\mu\nu} - \partial_{\mu}\partial_{\nu}h_{\sigma\rho}) \quad (2.9)$$

$$R = \partial_{\mu}\partial_{\nu}h^{\mu\nu} - \square h \quad (2.10)$$

ここで

$$\square := \eta^{\mu\nu}\partial_{\mu}\partial_{\nu} \quad (2.11)$$

$$h := \eta^{\mu\nu}h_{\mu\nu} \quad (2.12)$$

と定義した. これらを Einstein 方程式 (2.2) に代入する. 新たに

$$\tilde{h}_{\mu\nu} := h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h \quad (2.13)$$

$$\tilde{h} := \eta^{\mu\nu}\tilde{h}_{\mu\nu} \quad (2.14)$$

を定義すると式 (2.2) は

$$\square\tilde{h}_{\mu\nu} + \eta_{\mu\nu}\partial^{\rho}\partial^{\sigma}\tilde{h}_{\rho\sigma} - \partial^{\rho}\partial_{\nu}\tilde{h}_{\mu\rho} - \partial^{\rho}\partial_{\mu}\tilde{h}_{\nu\rho} = -\frac{16\pi G}{c^4}T_{\mu\nu} \quad (2.15)$$

となる. この式を簡単化していく. ゲージ変換 $x^{\mu} \rightarrow x'^{\mu} = x^{\mu} + \xi^{\mu}(x)$ に対し, $h_{\mu\nu}, \tilde{h}_{\mu\nu}$ はそれぞれ

$$h_{\mu\nu}(x) \rightarrow h'_{\mu\nu}(x') = h_{\mu\nu}(x) - (\partial_{\mu}\xi_{\nu} + \partial_{\nu}\xi_{\mu}) \quad (2.16)$$

$$\tilde{h}_{\mu\nu}(x) \rightarrow \tilde{h}'_{\mu\nu}(x') = \tilde{h}_{\mu\nu}(x) - (\partial_{\mu}\xi_{\nu} + \partial_{\nu}\xi_{\mu} - \eta_{\mu\nu}\partial_{\rho}\xi^{\rho}) \quad (2.17)$$

と変換される. したがって

$$\partial^{\nu}\tilde{h}_{\mu\nu} \rightarrow (\partial^{\nu}\tilde{h}_{\mu\nu})' = \partial^{\nu}\tilde{h}_{\mu\nu} - \square\xi_{\mu} \quad (2.18)$$

となるから $\tilde{h}_{\mu\nu}$ が与えられたとき

$$\square\xi_{\mu} = \partial^{\nu}\tilde{h}_{\mu\nu} \quad (2.19)$$

を満たす ξ_{μ} を用いてゲージ変換すれば, 常に

$$(\partial^{\nu}\tilde{h}_{\mu\nu})' = 0 \quad (2.20)$$

とできる. この条件は Lorenz 条件と呼ばれる. また, 変換後の $\tilde{h}_{\mu\nu}$ は (2.15) の左辺2項目以降が0になるから

$$\square\tilde{h}_{\mu\nu} = -\frac{16\pi G}{c^4}T_{\mu\nu} \quad (2.21)$$

を満たす. これが線形化された Einstein 方程式である.

2.1.3 平面波解

真空中ではエネルギー運動量テンソル $T_{\mu\nu}$ は 0 であるから式 (2.21) は

$$\square \tilde{h}_{\mu\nu} = 0 \quad (2.22)$$

となる。この解として単色平面波解

$$\tilde{h}_{\mu\nu} = \text{Re}[A_{\mu\nu} \exp(ik_\alpha x^\alpha)] \quad (2.23)$$

を考える。 $A_{\mu\nu}$ は対称テンソルである。これを式 (2.22) に代入すると

$$k_\alpha k^\alpha = 0 \quad (2.24)$$

となる。また、Lorenz 条件 $\partial^\nu \tilde{h}_{\mu\nu} = 0$ から

$$k^\mu A_{\mu\nu} = 0 \quad (2.25)$$

が成り立つ。これは重力波が横波であることを表している。この条件によって、対称テンソル $A_{\mu\nu}$ の自由度は 10 個から 6 個に減る。さらに、Lorenz 条件 (2.19) は ξ_μ に $\square \chi_\mu = 0$ を満たす χ_μ を加えても成り立つため、 ξ_μ に 4 つの自由度がある。結果、重力波の自由度は $10 - 4 - 4 = 2$ となる。これを具体的に確認する。 χ_μ を

$$\chi_\mu = -\text{Re}[C_\mu \exp(ik_\alpha x^\alpha)] \quad (2.26)$$

とおくと、式 (2.24) から $\square \chi_\mu = 0$ を満たす。このゲージ変換に対し、 $A_{\mu\nu}$ は式 (2.17) から

$$\tilde{h}'_{\mu\nu} = \tilde{h}_{\mu\nu} - (\partial_\mu \chi_\nu + \partial_\nu \chi_\mu - \eta_{\mu\nu} \partial_\rho \chi^\rho) \quad (2.27)$$

$$\Rightarrow A'_{\mu\nu} = A_{\mu\nu} + C_\mu k_\nu + C_\nu k_\mu - \eta_{\mu\nu} C_\rho k^\rho \quad (2.28)$$

と変換されることが分かる。ここで、まず $A'_{\mu\nu}$ が空間成分のみであるとする。つまり $\nu = 0, 1, 2, 3$ に対し

$$A'_{0\nu} = 0 \quad (2.29)$$

という条件を課す。ここで 4 つの自由度を使っているように見えるが実際は、この 4 条件のうちの 3 つを仮定すれば横波の条件 (2.25) から残りの 1 つも成り立つため、ここでは 3 つの自由度を用いていることになる。 χ_μ の残る 1 つの自由度は $\text{Tr } A'_{\mu\nu} = 0$ 、つまり $\text{Tr } h'_{\mu\nu} = 0$ となるように用いる：

$$\eta^{\mu\nu} A'_{\mu\nu} = \eta^{\mu\nu} A_{\mu\nu} - 2C_\rho k^\rho = 0 \quad (2.30)$$

$$\Rightarrow C_\rho k^\rho = \frac{1}{2} \eta^{\mu\nu} A_{\mu\nu} \quad (2.31)$$

以上をまとめると、 $h_{\mu\nu}$ に次の 8 条件を課すことによって $h_{\mu\nu}$ の 10 個の自由度を 2 つに減らすことができることがわかった。

$$h_{\mu 0} = 0, \quad \partial^\nu h_{\mu\nu} = 0, \quad h^\mu{}_\mu = 0 \quad (2.32)$$

これはトランスバース・トレースレスゲージ (TT ゲージ) と呼ばれる。

TT ゲージの下で z 方向に伝播する重力波を考える．このとき式 (2.24) を満たす k^μ は

$$k^\mu = (k, 0, 0, k) \quad (2.33)$$

と表せる．また，横波の条件 (2.25) から

$$kA_{0\nu} + kA_{3\nu} = 0 \quad (2.34)$$

が成り立つ．この式と $A_{\mu\nu}$ は空間成分のみであるという仮定から

$$A_{0\nu} = 0, \quad A_{3\nu} = 0 \quad (2.35)$$

となる． $A_{11} = h_+$, $A_{12} = h_\times$ とすると, $\text{Tr } A_{\mu\nu} = 0$ から $A_{22} = -h_+$ となる．残りの成分は $A_{\mu\nu}$ が対称テンソルであるという仮定から決まる．以上より TT 条件の下での $h_{\mu\nu}$ は

$$h_{\mu\nu}^{\text{TT}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cos(kz - \omega t) \quad (2.36)$$

と2つの独立な成分 h_+ , h_\times を使って表せることが分かった． h_+ , h_\times はそれぞれ重力波のプラスモード，クロスモードと呼ばれる．

TT ゲージでは線素 $ds^2 = g_{\mu\nu}dx^\mu dx^\nu$ は

$$ds^2 = -c^2 dt^2 + (1 + h_+ \cos(kz - \omega t))dx^2 + (1 - h_+ \cos(kz - \omega t))dy^2 + 2h_\times \cos(kz - \omega t)dx dy + dz^2 \quad (2.37)$$

となるから， $h_\times = 0$ のとき

$$ds^2 = -c^2 dt^2 + (1 + h_+ \cos(kz - \omega t))dx^2 + (1 - h_+ \cos(kz - \omega t))dy^2 + dz^2 \quad (2.38)$$

となる．これは x 方向が伸びたとき y 方向が縮み， x 方向が縮んだとき y 方向が伸びることを表している．逆に $h_+ = 0$ のときは

$$ds^2 = -c^2 dt^2 + dx^2 + dy^2 + 2h_\times \cos(kz - \omega t)dx dy + dz^2 \quad (2.39)$$

$$\begin{aligned} &= -c^2 dt^2 + (1 - h_+ \cos(kz - \omega t)) \left(\frac{dx - dy}{\sqrt{2}} \right)^2 \\ &\quad + (1 + h_+ \cos(kz - \omega t)) \left(\frac{dx + dy}{\sqrt{2}} \right)^2 + dz^2 \end{aligned} \quad (2.40)$$

となる．これは x, y 軸をそれぞれ $\pi/4$ 回転させた軸の向きに伸び縮みすることを表している．したがってプラスモード，クロスモードそれぞれのモードの重力波が来たときの自由質点間の距離は図 2.1 のようになる．

ここで，TT ゲージでの h_{ij} ($h_{\mu\nu}$ の空間成分) を与える射影演算子を導入する． \hat{n} 方向に伝播する重力波が Lorenz 条件を満たすが，TT 条件を満たしていないとする．このとき演算子

$$P_{ij}(\hat{n}) := \delta_{ij} - n_i n_j \quad (2.41)$$

を用いてラムダ演算子

$$\Lambda_{ij,kl}(\hat{n}) := P_{ik}P_{jl} - \frac{1}{2}P_{ij}P_{kl} \quad (2.42)$$

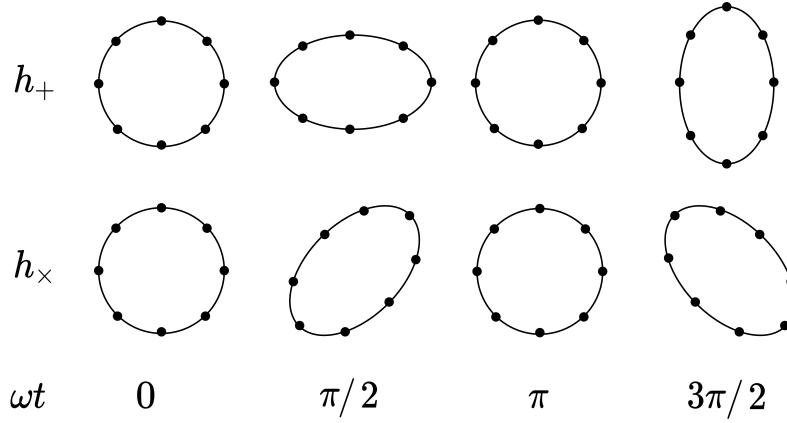


図 2.1: 紙面に垂直な方向に重力波が伝播するときの自由質点の変位

を定義すると、これは

$$\Lambda_{ij,kl}\Lambda_{kl,mn} = \Lambda_{ij,mn} \quad (2.43)$$

$$n^i\Lambda_{ij,kl} = n^j\Lambda_{ij,kl} = 0 \quad (2.44)$$

$$\Lambda_{ii,kl} = \Lambda_{ij,kk} = 0 \quad (2.45)$$

を満たし、TT ゲージでの重力波の空間成分は

$$h_{ij}^{\text{TT}} = \Lambda_{ij,kl}\tilde{h}_{kl} = \Lambda_{ij,kl}h_{kl} \quad (2.46)$$

で与えられる（2つ目の等号は式 (2.45) を用いた）。

2.2 重力波の発生

本節は重力波の振幅が四重極モーメントの時間2階微分で表されることを確認する。また、重力波が放射するエネルギーについて述べる。

2.2.1 四重極放射

弱い重力場に対しては、2.1.2 項で導出したように、線形化された Einstein 方程式

$$\square\tilde{h}_{\mu\nu} = -\frac{16\pi G}{c^4}T_{\mu\nu} \quad (2.47)$$

が成り立つ。この波動方程式は Green 関数を用いることで

$$\tilde{h}_{\mu\nu}(t, \mathbf{x}) = \frac{4G}{c^4} \int d^3x' \frac{1}{|\mathbf{x} - \mathbf{x}'|} T_{\mu\nu} \left(t - \frac{|\mathbf{x} - \mathbf{x}'|}{c}, \mathbf{x}' \right) \quad (2.48)$$

と解くことができる。TT ゲージでの重力波は 2.1.3 項で導入したラムダ演算子を用いて

$$h_{ij}^{\text{TT}}(t, \mathbf{x}) = \frac{4G}{c^4} \Lambda_{ij,kl} \int d^3x' \frac{1}{|\mathbf{x} - \mathbf{x}'|} T_{kl} \left(t - \frac{|\mathbf{x} - \mathbf{x}'|}{c}, \mathbf{x}' \right) \quad (2.49)$$

と表せる。

式 (2.49) 右辺の多重極展開を考える．重力波源の典型的な角周波数を ω_s ，大きさを d とすると速さは $v \sim \omega_s d$ と表せる．放射する重力波の周波数 ω は ω_s のオーダーであり

$$\omega \sim \omega_s \sim \frac{v}{d} \quad (2.50)$$

であるから

$$\lambda = \frac{c}{\omega} \sim \frac{c}{v} d \quad (2.51)$$

となる．よって非相対論的な系では $v \ll c$ だから重力波の波長は波源より十分大きいことが分かる：

$$\lambda \gg d \quad (2.52)$$

このような場合，波源内部の運動の詳細について知る必要はないため，多重極モーメントの最低次の項だけを考える．重力波源の大きさ d より $r = |\mathbf{x}|$ が十分大きいとすると

$$|\mathbf{x} - \mathbf{x}'| \simeq r - \mathbf{x}' \cdot \hat{\mathbf{x}} \quad (2.53)$$

と近似できるから式 (2.49) は

$$h_{ij}^{\text{TT}}(t, \mathbf{x}) = \frac{1}{r} \frac{4G}{c^4} \Lambda_{ij,kl} \int d^3x' T_{kl} \left(t - \frac{r}{c} + \frac{\mathbf{x}' \cdot \hat{\mathbf{x}}}{c}, \mathbf{x}' \right) \quad (2.54)$$

となる．ここで

$$\begin{aligned} T_{kl} \left(t - \frac{r}{c} + \frac{\mathbf{x}' \cdot \hat{\mathbf{x}}}{c}, \mathbf{x}' \right) &= T_{kl} \left(t - \frac{r}{c}, \mathbf{x}' \right) + \mathbf{x}' \cdot \hat{\mathbf{x}} \partial_0 T_{kl} \left(t - \frac{r}{c}, \mathbf{x}' \right) \\ &\quad + \frac{1}{2} (\mathbf{x}' \cdot \hat{\mathbf{x}})^2 \partial_0^2 T_{kl} \left(t - \frac{r}{c}, \mathbf{x}' \right) + \cdots \end{aligned} \quad (2.55)$$

と Taylor 展開する． T^{ij} に対応する運動量を

$$S^{ij}(t) := \int d^3x T^{ij}(t, \mathbf{x}) \quad (2.56)$$

$$S^{ij,k}(t) := \int d^3x T^{ij}(t, \mathbf{x}) x^k \quad (2.57)$$

$$S^{ij,kl}(t) := \int d^3x T^{ij}(t, \mathbf{x}) x^k x^l \quad (2.58)$$

と定義する．式 (2.55) を式 (2.54) に代入すると

$$h_{ij}^{\text{TT}}(t, \mathbf{x}) = \frac{1}{r} \frac{4G}{c^4} \Lambda_{ij,kl} \left[S^{kl} + \hat{x}_m \partial_0 S^{kl,m} + \frac{1}{2} \hat{x}_m \hat{x}_n \partial_0^2 S^{kl,mn} \right]_{\text{ret}} \quad (2.59)$$

となる．ここで ret は各運動量やその微分に $(t - r/c, \mathbf{x}')$ を代入することを意味する．さらにエネルギー密度 T^{00}/c^2 と運動量密度 T^{0i}/c に対する運動量を定義しておく：

$$M = \frac{1}{c^2} \int d^3x T^{00}(t, \mathbf{x}) \quad (2.60)$$

$$M^i = \frac{1}{c^2} \int d^3x T^{00}(t, \mathbf{x}) x^i \quad (2.61)$$

$$M^{ij} = \frac{1}{c^2} \int d^3x T^{00}(t, \mathbf{x}) x^i x^j \quad (2.62)$$

$$M^{ijk} = \frac{1}{c^2} \int d^3x T^{00}(t, \mathbf{x}) x^i x^j x^k \quad (2.63)$$

$$P^i = \frac{1}{c} \int d^3x T^{0i}(t, \mathbf{x}) \quad (2.64)$$

$$P^{i,j} = \frac{1}{c} \int d^3x T^{0i}(t, \mathbf{x}) x^j \quad (2.65)$$

$$P^{i,jk} = \frac{1}{c} \int d^3x T^{0i}(t, \mathbf{x}) x^j x^k \quad (2.66)$$

今、重力場の線形近似を考えているからエネルギー運動量テンソルは保存則

$$\partial_\nu T^{\mu\nu} = 0 \quad (2.67)$$

を満たす。したがって波源より十分大きな積分領域 V をとれば

$$c\dot{M} = \int d^3x \partial_0 T^{00} = - \int d^3x \partial_i T^{0i} = - \int_{\partial V} dS_i T^{0i} = 0 \quad (2.68)$$

が成り立つ（3つ目の等号は Gauss の定理を用いた）。同様に計算すると次の関係式が得られる：

$$\dot{M} = 0 \quad (2.69)$$

$$\dot{M}^i = P^i \quad (2.70)$$

$$\dot{M}^{ij} = P^{i,j} + P^{j,i} \quad (2.71)$$

$$\dot{M}^{ijk} = P^{i,jk} + P^{j,ki} + P^{k,ij} \quad (2.72)$$

$$\dot{P}^i = 0 \quad (2.73)$$

$$\dot{P}^{i,j} = S^{ij} \quad (2.74)$$

$$\dot{P}^{i,jk} = S^{ij,k} + S^{ik,j} \quad (2.75)$$

式 (2.71) を時間微分し、右辺に式 (2.74) と S^{ij} の対称性を用いることにより

$$\ddot{M}^{ij} = 2S^{ij} \quad (2.76)$$

を得る。これを用いると式 (2.59) の主要項は

$$h_{ij}^{\text{TT}}(t, \mathbf{x}) = \frac{1}{r} \frac{2G}{c^4} \Lambda_{ij,kl} \ddot{M}^{kl}(t - r/c) \quad (2.77)$$

と表せる。トレースレスな四重極モーメント

$$Q^{kl} := M^{kl} - \frac{1}{3} \delta^{kl} M_{ii} \quad (2.78)$$

を用いると、ラムダ演算子の性質 (2.45) から $\Lambda_{ij,kl} \delta^{kl} = 0$ となり

$$\Lambda_{ij,kl} Q^{kl} = \Lambda_{ij,kl} M^{kl} \quad (2.79)$$

が成り立つ。したがって式 (2.77) は

$$h_{ij}^{\text{TT}}(t, \mathbf{x}) = \frac{1}{r} \frac{2G}{c^4} \Lambda_{ij,kl} \ddot{Q}^{kl}(t - r/c) \quad (2.80)$$

$$= \frac{1}{r} \frac{2G}{c^4} \ddot{Q}_{ij}^{\text{TT}}(t - r/c) \quad (2.81)$$

と書き直すことができる。これは重力波の四重極公式と呼ばれる。

導出した式を用いて \hat{z} 方向に伝播する重力波の振幅を求める。式 (2.41) で定義した P_{ij} は \hat{z} 方向に伝播するため

$$P_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (2.82)$$

であるから

$$\begin{aligned} \Lambda_{ij,kl} \ddot{M}_{kl} &= \left(P_{ik} P_{jl} - \frac{1}{2} P_{ij} P_{kl} \right) \ddot{M}_{kl} \\ &= (P \ddot{M} P)_{ij} - \frac{1}{2} P_{ij} (\ddot{M}_{11} + \ddot{M}_{22}) \\ &= \begin{pmatrix} (\ddot{M}_{11} - \ddot{M}_{22})/2 & \ddot{M}_{12} & 0 \\ \ddot{M}_{21} & -(\ddot{M}_{11} - \ddot{M}_{22})/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}_{ij} \end{aligned} \quad (2.83)$$

となる。これを式 (2.77) に代入すると

$$h_+ = h_{11}^{\text{TT}} = \frac{1}{r} \frac{G}{c^4} (\ddot{M}_{11} - \ddot{M}_{22})_{\text{ret}} \quad (2.84)$$

$$h_\times = h_{12}^{\text{TT}} = \frac{2}{r} \frac{G}{c^4} (\ddot{M}_{12})_{\text{ret}} \quad (2.85)$$

を得る。

次にこれを用いて任意の方向 \hat{n} に伝播する重力波を求める。 \hat{z}' が \hat{n} に一致するような座標系 (x', y', z') をとり、図 2.2 のように角度 θ, ϕ を定義する。 (x', y', z') 系での M' と (x, y, z) 系での M は行列

$$\mathcal{R} = \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{pmatrix} \quad (2.86)$$

を用いて

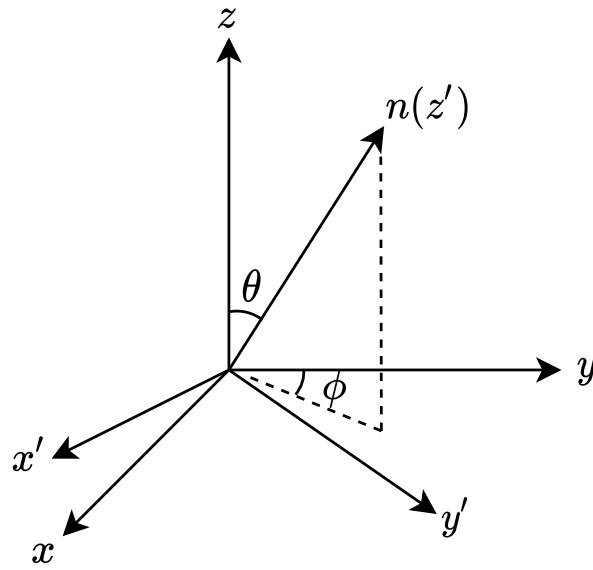
$$M'_{ij} = (\mathcal{R}^\top M \mathcal{R})_{ij} \quad (2.87)$$

と変換される。これを式 (2.84) と式 (2.85) に代入することで、 $\hat{n} = (\sin \theta \sin \phi, \sin \theta \cos \phi, \cos \theta)$ 方向に伝播する重力波の振幅が

$$\begin{aligned} h_+(t; \theta, \phi) &= \frac{1}{r} \frac{G}{c^4} [\ddot{M}_{11}(\cos^2 \phi - \sin^2 \phi \cos^2 \theta) + \ddot{M}_{22}(\sin^2 \phi - \cos^2 \phi \cos^2 \theta) \\ &\quad - \ddot{M}_{33} \sin^2 \theta - \ddot{M}_{12} \sin 2\phi (1 + \cos^2 \theta) \\ &\quad + \ddot{M}_{13} \sin \phi \sin 2\theta + \ddot{M}_{23} \cos \phi \sin 2\theta]_{\text{ret}} \end{aligned} \quad (2.88)$$

$$\begin{aligned} h_\times(t; \theta, \phi) &= \frac{1}{r} \frac{G}{c^4} [(\ddot{M}_{11} - \ddot{M}_{22}) \sin 2\phi \cos \theta + 2\ddot{M}_{12} \cos 2\phi \cos \theta \\ &\quad - 2\ddot{M}_{13} \cos \phi \sin \theta - \ddot{M}_{23} \sin \phi \sin \theta]_{\text{ret}} \end{aligned} \quad (2.89)$$

と表されることが分かる。

図 2.2: (x, y, z) 系と重力波の伝播方向の関係.

2.2.2 重力波のエネルギー

重力波のエネルギー運動量テンソルは R. A. Isaacson の方法 [38, 39] を用いることによって次のように表すことができる:

$$t_{\mu\nu} = \frac{c^4}{32\pi G} \langle \partial_\mu h_{\alpha\beta} \partial_\nu h^{\alpha\beta} \rangle \quad (2.90)$$

$\langle \cdot \rangle$ は空間的な平均をとることを意味し、右辺は TT ゲージをとっている. 重力波の持つエネルギーの放射率 (luminosity) は

$$\begin{aligned} \frac{dE}{dt} &= c \int dA t_{00} \\ &= \frac{c^3 r^2}{32\pi G} \int d\Omega \langle \dot{h}_{ij}^{\text{TT}} \dot{h}_{ij}^{\text{TT}} \rangle \\ &= \frac{c^3 r^2}{16\pi G} \int d\Omega \langle \dot{h}_+^2 + \dot{h}_\times^2 \rangle \end{aligned} \quad (2.91)$$

となる. 式 (2.80) を式 (2.91) に代入すると

$$\frac{dE}{dt} = \frac{G}{8\pi c^5} \int d\Omega \Lambda_{ij,kl} \langle \ddot{Q}_{ij} \ddot{Q}_{kl} \rangle \quad (2.92)$$

となる. Q は遅延時間 $t - r/c$ のみの関数であるから、右辺の積分は $\Lambda_{ij,kl}$ のみについて行う. ラムダ演算子を

$$\begin{aligned} \Lambda_{ij,kl} &= P_{ik} P_{jl} - \frac{1}{2} P_{ij} P_{kl} \\ &= \delta_{ik} \delta_{jl} - \delta_{ik} n_j n_l - \delta_{jl} n_i n_k - \frac{1}{2} \delta_{ij} \delta_{kl} + \frac{1}{2} \delta_{ij} n_k n_l + \frac{1}{2} \delta_{kl} n_i n_j + \frac{1}{2} n_i n_j n_k n_l \end{aligned} \quad (2.93)$$

と表し,

$$\int d\Omega n_i n_j = \frac{4\pi}{3} \delta_{ij} \quad (2.94)$$

$$\int d\Omega n_i n_j n_k n_l = \frac{4\pi}{15} (\delta_{ij} \delta_{kl} + \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \quad (2.95)$$

を用いるとラムダ演算子の積分は

$$\int d\Omega \Lambda_{ij,kl} = \frac{2\pi}{15} (11\delta_{ik} \delta_{jl} - 4\delta_{ij} \delta_{kl} + \delta_{il} \delta_{jk}) \quad (2.96)$$

となり, これを式 (2.92) に代入すると

$$\frac{dE}{dt} = \frac{G}{5c^5} \langle \ddot{Q}_{ij} \ddot{Q}_{ij} \rangle \quad (2.97)$$

を得る.

2.2.3 重力波源

前節で導出したように, 放射される重力波の振幅は四重極モーメントの時間2階微分で表されるため, 実際に観測できる重力波源は大きな天体現象に限られる. 本節では観測された, または観測が期待されている重力波源について述べる.

コンパクト連星合体

ブラックホール, 中性子星などの質量と密度の大きいコンパクト星の連星は互いの重力によって公転運動する. このとき重力波を放射しながら距離を縮め, 合体する. O1 から O3 の観測期間でブラックホール連星合体, 中性子星連星合体に加え, 中性子星ブラックホール連星合体からの重力波も観測されている [40]. コンパクト連星合体からの重力波には, 公転運動期 (inspiral), 衝突合体期 (merger), 減衰振動期 (ringdown) の3つのフェイズがあり, それぞれ図 2.3 のような波形をしている. 波形はポスト・ニュートン近似や数値相対論を用いて計算することができ, 4章で詳しく述べるマッチドフィルタによって解析することができる.

バースト

短時間で大きな振幅を示し, 波形の予測が困難な重力波または未知の波源からの重力波はバーストと呼ばれ, 超新星爆発からの重力波などがこれにあたる. 超新星爆発とは太陽の8倍以上の質量をもつ恒星が進化の最終段階で起こす爆発のことであり, 重力波の観測によってそのメカニズムを解明することが期待されている. バースト重力波は波形を正確に予測することが困難なため, マッチドフィルタを用いることはできない. そのため1台の検出器に対しては, 時間周波数領域でのパワーを用いる Excess power 法 [41] という解析手法が用いられる. しかし, この方法は信号と突発性雑音を区別できないため, 複数台の検出器を用いたコヒーレントネットワーク解析 [42] という手法が用いられる.

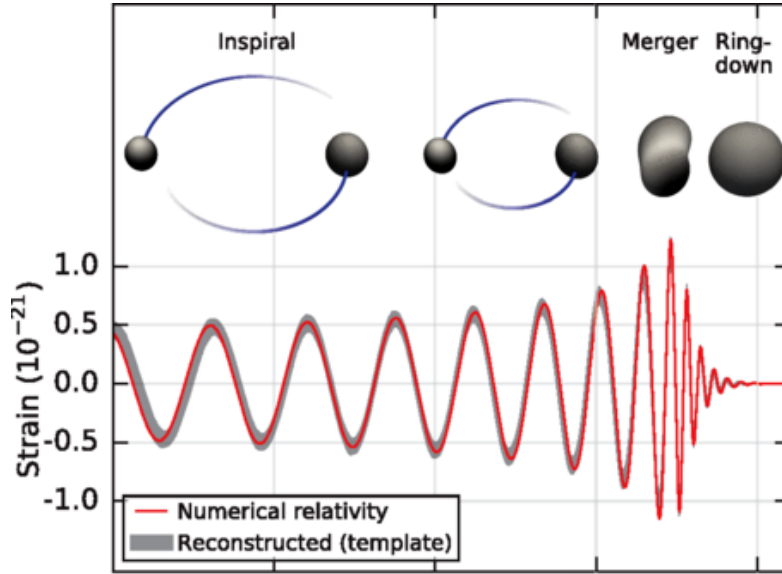


図 2.3: コンパクト連星合体からの重力波の波形 ([4] より引用).

連続重力波

連続重力波とは、長時間ほぼ一定の振幅と周波数で放射が続く重力波のことである。非軸対称な質量分布の中性子星が高速回転すると、その自転周波数の2倍の周波数の連続重力波が放射される。短時間で見ると振幅と周波数は一定だが、長期間観測をすると中性子星の角運動量の減少や地球の自転、公転の影響により振幅や振動数は変化する。波形は正弦波のため、マッチドフィルタによる解析が可能である。

確率的重力波

宇宙のあらゆるところから小さな重力波が定常的に発生しており、それらはランダムに混ざり合っている。このような小さな重力波に対しては統計的な分析はできるが、波形の予測はできないため確率的重力波と呼ばれる。波形のランダム性により、検出器のノイズと区別がつかないため、複数台の検出器の相関をとるという解析手法が使われる [22]。確率的重力波の中にはインフレーションの時期に生成された原始重力波が含まれていると考えられている。原始重力波は振幅が小さいため観測は困難であるが、その観測によって初期宇宙の情報が得られると期待されている。

2.3 連星系からの重力波

本節では前節で紹介した重力波源のひとつであるコンパクト連星合体の簡単な模型を考える。具体的には2つの連星を質点近似し、それらが円軌道を描くときの重力波の振幅をニュートン近似によって計算する。

xy 平面にある半径 R の円上を角周波数 ω_s で回転する質量 m_1, m_2 の2つの質点の位置は

$$(x_1, y_1, z_1) = \left(-\frac{m_2}{m_1 + m_2} R \sin(\omega_s t), \frac{m_2}{m_1 + m_2} R \cos(\omega_s t), 0 \right) \quad (2.98)$$

$$(x_2, y_2, z_2) = \left(\frac{m_1}{m_1 + m_2} R \sin(\omega_s t), -\frac{m_1}{m_1 + m_2} R \cos(\omega_s t), 0 \right) \quad (2.99)$$

と表せる。このとき四重極モーメントは

$$M_{11} = m_1 x_1^2 + m_2 x_2^2 = \mu R^2 \frac{1 - \cos(2\omega_s t)}{2} \quad (2.100)$$

$$M_{22} = m_1 y_1^2 + m_2 y_2^2 = \mu R^2 \frac{1 + \cos(2\omega_s t)}{2} \quad (2.101)$$

$$M_{12} = m_1 x_1 y_1 + m_2 x_2 y_2 = -\mu R^2 \frac{\sin(2\omega_s t)}{2} \quad (2.102)$$

$$M_{13} = M_{23} = M_{33} = 0 \quad (2.103)$$

である。ここで μ は換算質量

$$\mu := \frac{m_1 m_2}{m_1 + m_2} \quad (2.104)$$

を表す。まず、 ω_s や R は一定として考えると四重極モーメントの2階時間微分は

$$\ddot{M}_{11} = 2\mu R^2 \omega_s^2 \cos(2\omega_s t) \quad (2.105)$$

$$\ddot{M}_{22} = -2\mu R^2 \omega_s^2 \cos(2\omega_s t) = -\ddot{M}_{11} \quad (2.106)$$

$$\ddot{M}_{12} = 2\mu R^2 \omega_s^2 \sin(2\omega_s t) \quad (2.107)$$

となり、式 (2.88) と式 (2.89) に代入すると

$$h_+(t) = \frac{1}{r} \frac{4G\mu\omega_s^2 R^2}{c^4} \frac{1 + \cos^2 \theta}{2} \cos(2\omega_s(t - r/c) + 2\phi) \quad (2.108)$$

$$h_\times(t) = \frac{1}{r} \frac{4G\mu\omega_s^2 R^2}{c^4} \cos \theta \sin(2\omega_s(t - r/c) + 2\phi) \quad (2.109)$$

となる。角度 θ は軌道の法線方向と視線の間の角度 ι に一致する。また、時間の原点をずらせば $\cos(2\omega_s(t - r/c) + 2\phi)$ や $\sin(2\omega_s(t - r/c) + 2\phi)$ はそれぞれ $\cos(2\omega_s t)$ や $\sin(2\omega_s t)$ にできるから

$$h_+(t) = \frac{1}{r} \frac{4G\mu\omega_s^2 R^2}{c^4} \frac{1 + \cos^2 \iota}{2} \cos(2\omega_s t) \quad (2.110)$$

$$h_\times(t) = \frac{1}{r} \frac{4G\mu\omega_s^2 R^2}{c^4} \cos \iota \sin(2\omega_s t) \quad (2.111)$$

と表すことができる。ここから、重力波の角周波数は重力波源の角周波数 ω_s の2倍であることがわかる。ここで Kepler の第3法則

$$\omega_s^2 = \frac{G(m_1 + m_2)}{R^3} \quad (2.112)$$

を用いて R を消去し、チャープ質量

$$M_c := \mu^{3/5} (m_1 + m_2)^{2/5} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} \quad (2.113)$$

と重力波の周波数 $f_{\text{gw}} := 2\omega_s/(2\pi)$ を用いると

$$h_+(t) = \frac{4}{r} \left(\frac{GM_c}{c^2} \right)^{5/3} \left(\frac{\pi f_{\text{gw}}}{c} \right)^{2/3} \frac{1 + \cos^2 \iota}{2} \cos(2\pi f_{\text{gw}} t_{\text{ret}}) \quad (2.114)$$

$$h_\times(t) = \frac{4}{r} \left(\frac{GM_c}{c^2} \right)^{5/3} \left(\frac{\pi f_{\text{gw}}}{c} \right)^{2/3} \cos \iota \sin(2\pi f_{\text{gw}} t_{\text{ret}}) \quad (2.115)$$

と表せ、振幅はチャープ質量という組み合わせによってのみ構成天体の質量に依存することがわかる。エネルギー放射率は式 (2.91) に式 (2.110) と式 (2.111) を代入すると

$$\begin{aligned} \frac{dE_{\text{gw}}}{dt} &= \frac{4G\mu^2\omega_s^6 R^4}{\pi c^5} \int_0^\pi d\theta \sin \theta \left(\left(\frac{1 + \cos^2 \theta}{2} \right)^2 + \cos^2 \theta \right) \\ &= \frac{32}{5} \frac{c^5}{G} \left(\frac{GM_c \omega_{\text{gw}}}{2c^3} \right)^{10/3} \end{aligned} \quad (2.116)$$

となる。

ここまでは ω_s や R を一定としていたが、次にエネルギーの放射による ω_s や R の変化の影響を考える。Kepler の第3法則 (2.112) を両辺時間微分すると

$$\dot{R} = -\frac{2}{3} \omega_s R \frac{\dot{\omega}_s}{\omega_s^2} \quad (2.117)$$

という関係式が得られるため、 $\dot{\omega}_s \ll \omega_s^2$ という条件下では $|\dot{R}|$ が回転速度 $\omega_s R$ より十分小さく、半径がゆっくり変化する円軌道の近似を適用することができる。放射されるエネルギーの源は質点の運動エネルギーとポテンシャルエネルギーの和であり、式 (2.112) から R を消去すると

$$E_{\text{orbit}} = -\frac{Gm_1 m_2}{2R} \quad (2.118)$$

$$= -\left(\frac{G^2 M_c^5 \omega_{\text{gw}}^2}{32} \right)^{1/3} \quad (2.119)$$

と表せる。エネルギー放射率は

$$\frac{dE_{\text{gw}}}{dt} = -\frac{dE_{\text{orbit}}}{dt} = \frac{2}{3} \left(\frac{G^2 M_c^5}{32} \right)^{1/3} \dot{\omega}_{\text{gw}} \omega_{\text{gw}}^{-1/3} \quad (2.120)$$

と計算でき、式 (2.116) と等式で結ぶと微分方程式

$$\dot{\omega}_{\text{gw}} = \frac{12}{5} 2^{1/3} \left(\frac{GM_c}{c^3} \right)^{5/3} \omega_{\text{gw}}^{11/3} \quad (2.121)$$

が得られる。この微分方程式を初期条件 $\lim_{t \rightarrow t_{\text{coal}}} \omega_{\text{gw}}(t) = \infty$ のもとで解くと

$$\omega_{\text{gw}}(\tau) = \frac{1}{4} \left(\frac{5}{\tau} \right)^{3/8} \left(\frac{GM_c}{c^3} \right)^{-5/8} \quad (2.122)$$

$$f_{\text{gw}}(\tau) = \frac{\omega_{\text{gw}}}{2\pi} = \frac{1}{8\pi} \left(\frac{5}{\tau} \right)^{3/8} \left(\frac{GM_c}{c^3} \right)^{-5/8} \quad (2.123)$$

となる。 t_{coal} は合体時刻であり、 $\tau := t_{\text{coal}} - t$ とした。

エネルギーの放射を考慮した振幅を求めるために、まず質点の位置 (2.98), (2.99) において $\omega_s t$ を

$$\begin{aligned}\Phi(t) &:= \int_{t_0}^t dt' 2\omega_s(t') \\ &= \int_{t_0}^t dt' \omega_{\text{gw}}(t')\end{aligned}\tag{2.124}$$

に書き換える必要がある。求めた $\omega_{\text{gw}}(\tau)$ を代入して積分すると、 $\tau = 0$ で $\Phi = \Phi_0$ として

$$\Phi(\tau) = -2 \left(\frac{5GM_c}{c^3} \right)^{-5/8} \tau^{5/8} + \Phi_0\tag{2.125}$$

となる。また、 M_{ij} の時間微分において、条件 $\dot{\omega}_s \ll \omega_s^2$ のもとで考えているから R や ω_s の時間微分は無視してよい。したがって式 (2.114) と式 (2.115) の f_{gw} を $f_{\text{gw}}(t_{\text{ret}})$ に、位相 $2\pi f_{\text{gw}} t_{\text{ret}}$ を $\Phi(t_{\text{ret}})$ にそれぞれ変えればよい。計算すると

$$h_+(\tau) = \frac{1}{r} \left(\frac{GM_c}{c^2} \right)^{5/4} \left(\frac{5}{c\tau} \right)^{2/3} \left(\frac{1 + \cos^2 \iota}{2} \right) \cos(\Phi(t_{\text{ret}}))\tag{2.126}$$

$$h_\times(\tau) = \frac{1}{r} \left(\frac{GM_c}{c^2} \right)^{5/4} \left(\frac{5}{c\tau} \right)^{2/3} \cos \iota \sin(\Phi(t_{\text{ret}}))\tag{2.127}$$

となる。

GW150914 [4] の各パラメータの値を用いて式 (2.123) で表される周波数をプロットすると図 2.4 のようになる。また、 $h_+(\tau)$ と $h_\times(\tau)$ をプロットすると図 2.5 のようになる。合体時刻に近づくにつれて振幅と周波数がともに大きくなる様子が読み取れる。このような波形は鳥のさえずりになぞらえてチャープ波形と呼ばれる。

ニュートン近似による図 2.5 の波形は良い近似であるが、スピンを考慮していない。また、2つの天体が近づくにつれて時空が平坦であるという仮定が成り立たなくなり、近似は正確でなくなる。そこでポスト・ニュートン近似や数値相対論による計算が必要になる。また、図 2.5 は実際の検出器で得られる波形ではないことに注意する必要がある。検出器の応答は検出器の種類や幾何学的性質で決まる検出器テンソルと重力波の振幅によって表される。Michelson 干渉計の場合の検出器の応答を第 3 章で述べる。

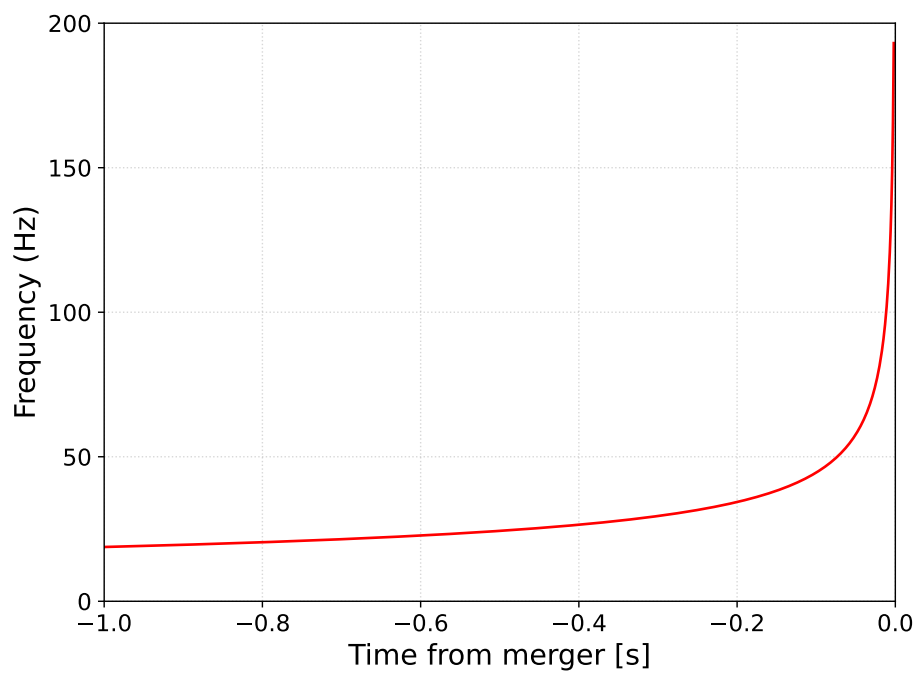


図 2.4: ニュートン近似による連星系からの重力波の周波数変化.

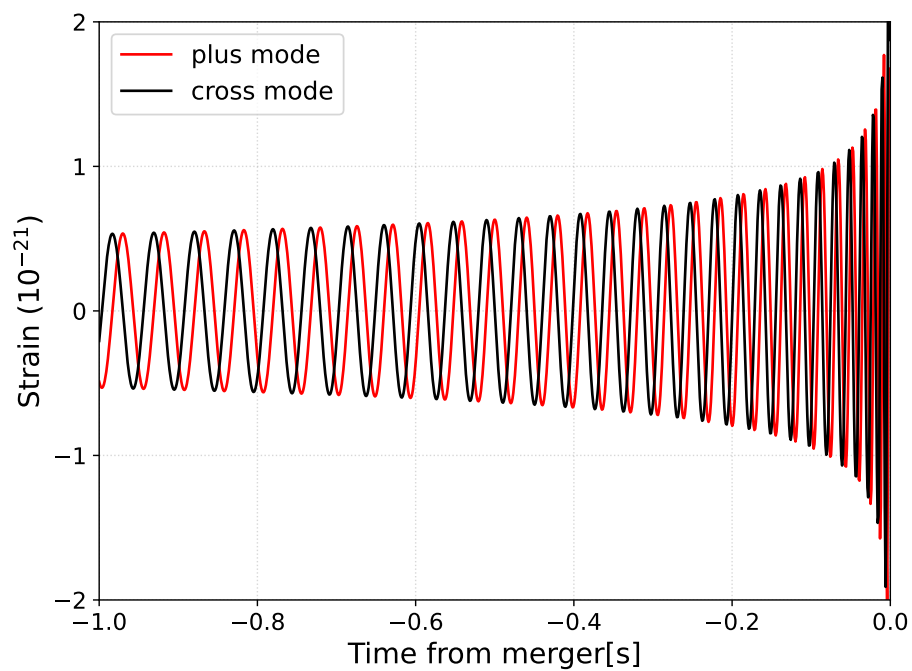


図 2.5: ニュートン近似による連星系からの重力波の振幅.

第 3 章

重力波の検出

現在重力波検出器として用いられているのは Michelson 干渉計を原理とするレーザー干渉計型重力波検出器である．本章ではまず Michelson 干渉計の原理を述べ，重力波に対する応答や各方向に対する感度を表すアンテナパターンを導出する．次に重力波検出器の主な雑音について述べる．

3.1 Michelson 干渉計

Michelson 干渉計の模式図を図 3.1 に示す．Michelson 干渉計では入射したレーザー光がビームスプリッターで 2 方向に分かれ，それぞれ x 軸方向と y 軸方向を進み，鏡で反射してビームスプリッターに戻り，干渉する．重力波は互いが 90 度をなす空間の長さを差動に変化させるため，Michelson 干渉計の干渉縞を変化させる．したがって Michelson 干渉計の干渉光強度の変化から重力波を検出することができる．以下で具体的にそれを計算してみる．

Michelson 干渉計の x 方向の腕の長さを L_x ， y 方向の腕の長さを L_y とし，TT ゲージの重力波が入射したときの Michelson 干渉計の応答を考える．プラスモードの重力波 $h_+(t)$ が入射するとすると 2.1.2 項で導出したように線素は

$$ds^2 = -c^2 dt^2 + (1 + h_+(t))dx^2 + (1 - h_+(t))dy^2 + dz^2 \quad (3.1)$$

と表せる．光は $ds^2 = 0$ に沿って進むから， x 方向を進む光は h_+ の 1 次近似で

$$dx = \pm c dt \left(1 - \frac{1}{2} h_+(t) \right) \quad (3.2)$$

に従う．プラス方向はビームスプリッターから鏡に進む光，マイナス方向はその逆を表す．この両辺を積分する．ビームスプリッターを時刻 t_0 で通過し，時刻 t_1 で鏡に到達したとすると

$$L_x = c(t_1 - t_0) - \frac{c}{2} \int_{t_0}^{t_1} dt' h_+(t') \quad (3.3)$$

となる．再び時刻 t でビームスプリッターに戻ったとすると，式 (3.2) のマイナスの方を積分して

$$-L_x = -c(t - t_1) + \frac{c}{2} \int_{t_1}^t dt' h_+(t') \quad (3.4)$$

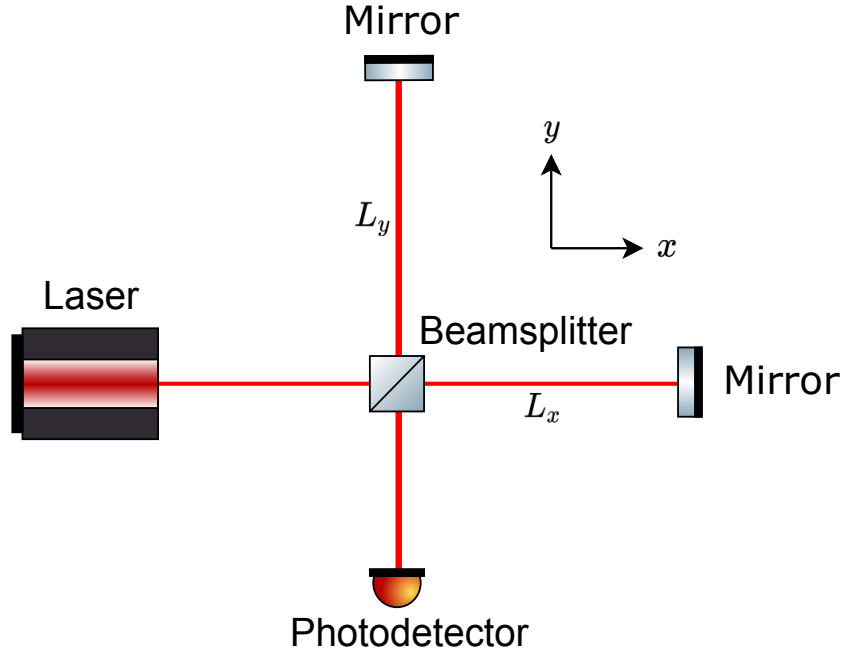


図 3.1: Michelson 干渉計

を得る．式 (3.3) と式 (3.4) を足す． h_+ の 1 次の近似では積分区間の t_0 を $t - 2L_x/c$ と近似できるから， x 方向の腕を往復する時間 Δt_x は

$$\begin{aligned}\Delta t_x &:= t - t_0 \\ &= \frac{2L_x}{c} + \frac{1}{2} \int_{t_0}^t dt' h_+(t') \\ &= \frac{2L_x}{c} + \frac{1}{2} \int_{t-2L_x/c}^t dt' h_+(t')\end{aligned}\quad (3.5)$$

と表せる． y 方向も同様に計算できるが，式 (3.1) から分かるように h_+ の符号だけが変わる：

$$\Delta t_y = \frac{2L_y}{c} - \frac{1}{2} \int_{t-2L_y/c}^t dt' h_+(t') \quad (3.6)$$

腕の往復によって生じる位相変化 $\Delta\phi_x, \Delta\phi_y$ は往復時間にレーザーの角周波数 Ω を掛けた値である：

$$\Delta\phi_x = \Omega \left(\frac{2L_x}{c} + \frac{1}{2} \int_{t-2L_x/c}^t dt' h_+(t') \right) \quad (3.7)$$

$$\Delta\phi_y = \Omega \left(\frac{2L_y}{c} - \frac{1}{2} \int_{t-2L_y/c}^t dt' h_+(t') \right) \quad (3.8)$$

通常 L_x と L_y はできるだけ等しくなるように作られる．今， h_+ の 1 次の近似を考えているから積分区間に含まれる L_x と L_y は $L = (L_x + L_y)/2$ に書き換えられる．したがって位相の差

動変化は

$$\begin{aligned}\Delta\phi &:= \Delta\phi_x - \Delta\phi_y \\ &= \frac{2(L_x - L_y)\Omega}{c} + \Omega \int_{t-2L/c}^t dt' h_+(t')\end{aligned}\quad (3.9)$$

と表せる．この第2項が重力波による位相変化を表している：

$$\Delta\phi_{\text{gw}} := \Omega \int_{t-2L/c}^t dt' h_+(t') \quad (3.10)$$

次に周波数応答を考える．重力波の振幅 $h_+(t)$ の Fourier 変換を $\tilde{h}(\omega)$ とする：

$$h_+(t) = \int_{-\infty}^{\infty} d\omega \tilde{h}(\omega) e^{i\omega t} \quad (3.11)$$

これを式 (3.10) に代入して t' 積分を実行すると

$$\begin{aligned}\Delta\phi_{\text{gw}} &= \Omega \int_{t-2L/c}^t dt' \int_{-\infty}^{\infty} d\omega \tilde{h}(\omega) e^{i\omega t'} \\ &= \int_{-\infty}^{\infty} d\omega \frac{2\Omega}{\omega} \sin\left(\frac{\omega L}{c}\right) e^{-i\omega L/c} \tilde{h}(\omega) e^{i\omega t}\end{aligned}\quad (3.12)$$

となる．したがって

$$H_M := \frac{2\Omega}{\omega} \sin\left(\frac{\omega L}{c}\right) e^{-i\omega L/c} \quad (3.13)$$

は重力波に対する干渉計の周波数応答関数と考えることができる．重力波を検出するためにはこの H_M をできるだけ大きくしたい． H_M の絶対値は $\omega L/c = \pi/2$ のとき最大になり，これを満たす L が最適な基線長である．最適な基線長を重力波の周波数 $f = \omega/(2\pi)$ で表すと

$$L \simeq 750\text{km} \left(\frac{100\text{Hz}}{f} \right) \quad (3.14)$$

となる．つまり 100Hz の重力波の検出に最適な基線長は 750km であるが，地上にこのような干渉計を設置するのは現実的に不可能である．そこで，Michelson 干渉計の腕に共振器を置いて実効的な基線長を伸ばすという手法が用いられている．

3.2 アンテナパターン

前節では x 方向に伝播するプラスモードの重力波を考えたが，ここでは任意の方向に伝播する任意の偏光を持つ重力波を考える．重力波 h^{ij} に対する検出器の応答は一般に検出器テンソル D_{ij} を用いて

$$h = D_{ij} h^{ij} \quad (3.15)$$

と表せる．検出器テンソルは検出器の幾何的な性質で決まり， x 軸と y 軸に腕を持つ干渉計では

$$D_{ij} = \frac{1}{2}(\hat{x}_i \hat{x}_j - \hat{y}_i \hat{y}_j) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}_{ij} \quad (3.16)$$

と表せる．つまり，この場合検出器の応答は $h = (h_{11} - h_{22})/2$ である．重力波による位相変化は式 (3.10) において h_+ をこの h に書き換えれば良いが， h_{11} や h_{22} は検出器の座標系で表したものであることに注意する必要がある．そこで，重力波の伝播方向に沿った座標系 (x', y', z') での重力波

$$h'_{ij} = \begin{pmatrix} h_+ & h_\times & 0 \\ h_\times & -h_+ & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (3.17)$$

を検出器の座標系 (x, y, z) に変換する．図 3.2 の θ, ϕ で指定される方向から重力波が入射するとし，偏極角は ψ とする．このとき (x', y', z') 系から (x, y, z) 系への回転行列

$$\begin{aligned} \mathcal{R} &= R_z^{-1}(\phi) R_y^{-1}(\theta) R_z^{-1}(\psi) \\ &= \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos \theta \cos \phi \cos \psi - \sin \phi \sin \psi & \cos \theta \cos \phi \sin \psi + \sin \phi \cos \psi & \sin \theta \cos \phi \\ -\cos \theta \sin \phi \cos \psi - \cos \phi \sin \psi & -\cos \theta \sin \phi \sin \psi + \cos \phi \cos \psi & -\sin \theta \sin \phi \\ -\sin \theta \cos \psi & -\sin \theta \sin \psi & \cos \theta \end{pmatrix} \end{aligned} \quad (3.18)$$

を用いて (x, y, z) 系での重力波の振幅は

$$h_{ij} = \mathcal{R}_{ik} \mathcal{R}_{jl} h'_{kl} \quad (3.19)$$

と変換される．実際に式 (3.18) を用いて h_{11} と h_{22} を計算すると

$$\begin{aligned} h_{11} &= h_+ (\cos^2 \theta \cos^2 \phi \cos 2\psi - \sin^2 \phi \cos 2\phi - \cos \theta \sin 2\phi \sin 2\psi) \\ &\quad + h_\times (\cos^2 \theta \cos^2 \phi \sin 2\psi + \cos \theta \sin 2\phi \cos 2\psi - \sin^2 \phi \sin 2\psi) \end{aligned} \quad (3.20)$$

$$\begin{aligned} h_{22} &= h_+ (\cos^2 \theta \cos 2\phi \sin^2 \psi - \cos 2\phi \cos^2 \psi + \cos \theta \sin 2\phi \sin 2\psi) \\ &\quad + h_\times (\cos^2 \theta \sin 2\phi \sin^2 \psi - \cos \theta \cos^2 \phi \sin 2\psi - \sin 2\phi \cos^2 \psi) \end{aligned} \quad (3.21)$$

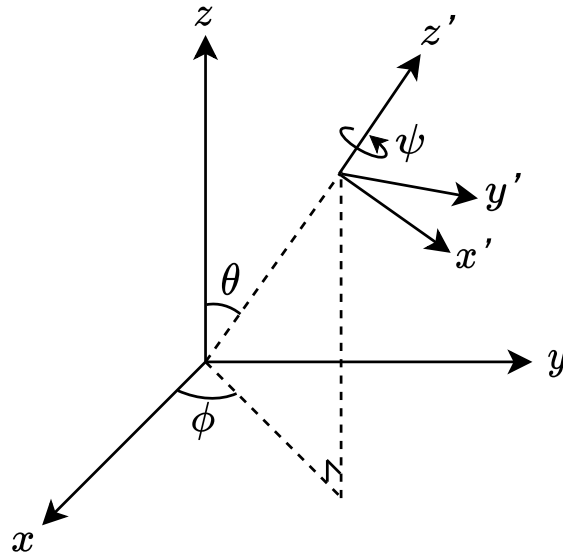


図 3.2: 検出器の座標系と重力波の座標系の関係

となるから $h = (h_{11} - h_{22})/2$ に代入して

$$h = h_+ \left(\frac{1 + \cos^2 \theta}{2} \cos 2\phi \cos 2\psi - \cos \theta \sin 2\phi \sin 2\psi \right) + h_\times \left(\frac{1 + \cos^2 \theta}{2} \cos 2\phi \sin 2\psi + \cos \theta \sin 2\phi \cos 2\psi \right) \quad (3.22)$$

を得る。したがって重力波の位相変化は

$$\Delta\phi_{\text{gw}} = \Omega \int_{t-2L/c}^t dt' (h_+(t')F_+(\theta, \phi, \psi) + h_\times(t')F_\times(\theta, \phi, \psi)) \quad (3.23)$$

$$F_+(\theta, \phi, \psi) = \frac{1 + \cos^2 \theta}{2} \cos 2\phi \cos 2\psi - \cos \theta \sin 2\phi \sin 2\psi \quad (3.24)$$

$$F_\times(\theta, \phi, \psi) = \frac{1 + \cos^2 \theta}{2} \cos 2\phi \sin 2\psi + \cos \theta \sin 2\phi \cos 2\psi \quad (3.25)$$

となる。 F_+, F_\times は各方向から到来する重力波に対する検出器の感度（アンテナパターン）を表す。 $\psi = 0$ に対して $|F_+|, |F_\times|$ 及びその平均 $\sqrt{F_+^2 + F_\times^2}$ をプロットすると図 3.3 のようになる。プラスモードでは、真上から到来すると最も感度が良く、干渉計と同じ平面上から到来すると感度が悪い。特に干渉計と同じ平面上で、腕と 45 度方向は感度が 0 である。これは x 方向と y 方向の位相差が打ち消しあってしまうからである。一方クロスモードでは $\phi = 0, \pi/2, \pi, 3\pi/2$ や $\theta = 0$ で感度が 0 になる。プラスモード、クロスモードの感度がともに 0 になるのは干渉計と同じ平面上で腕と 45 度方向から来る場合であり、 $\sqrt{F_+^2 + F_\times^2}$ の図を見てもそれが読み取れる。

LIGO H1, LIGO L1, Virgo, KAGRA の $\psi = 0$ に対するアンテナパターン関数 $\sqrt{F_+^2 + F_\times^2}$ を図 3.4 に示した。KAGRA は特に他の 3 台の検出器の感度の悪い方向に良い感度を持っていることが分かる。

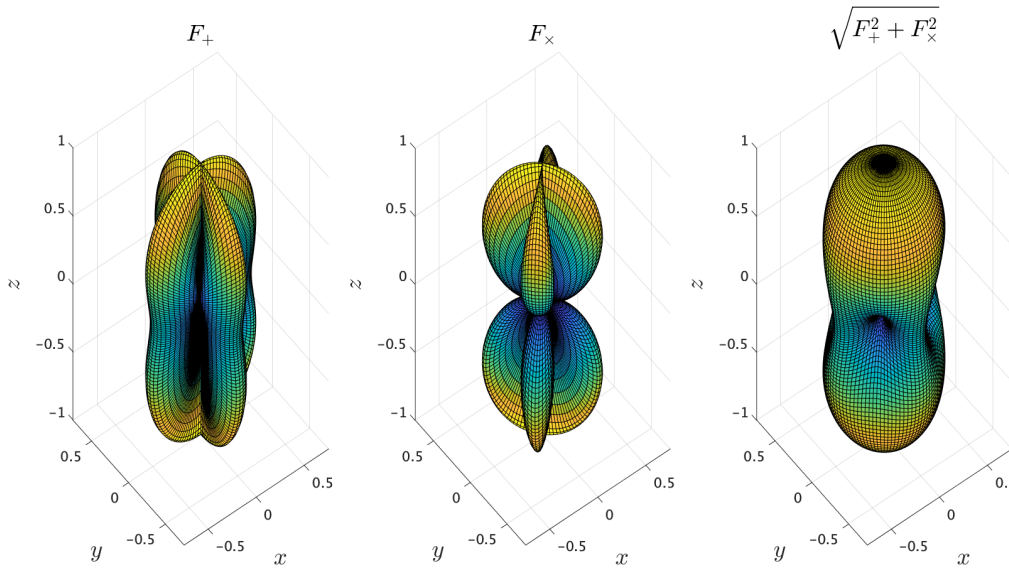
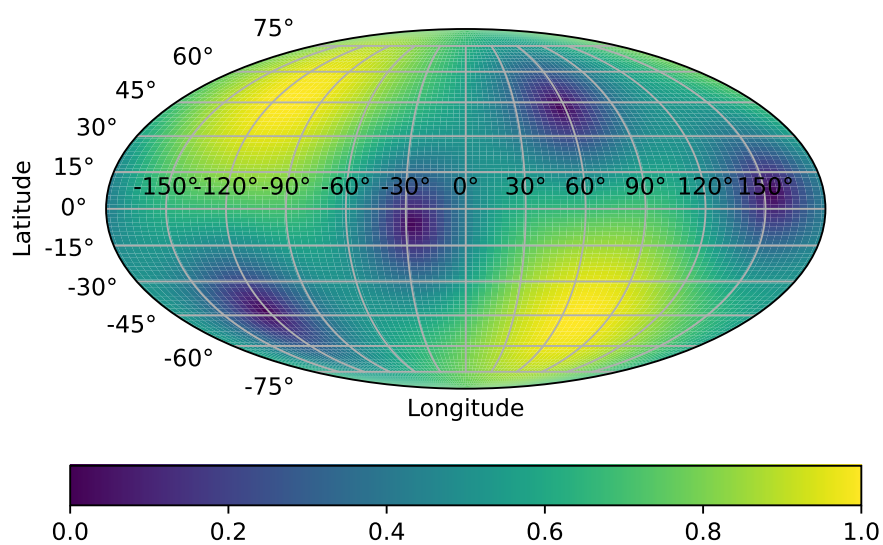
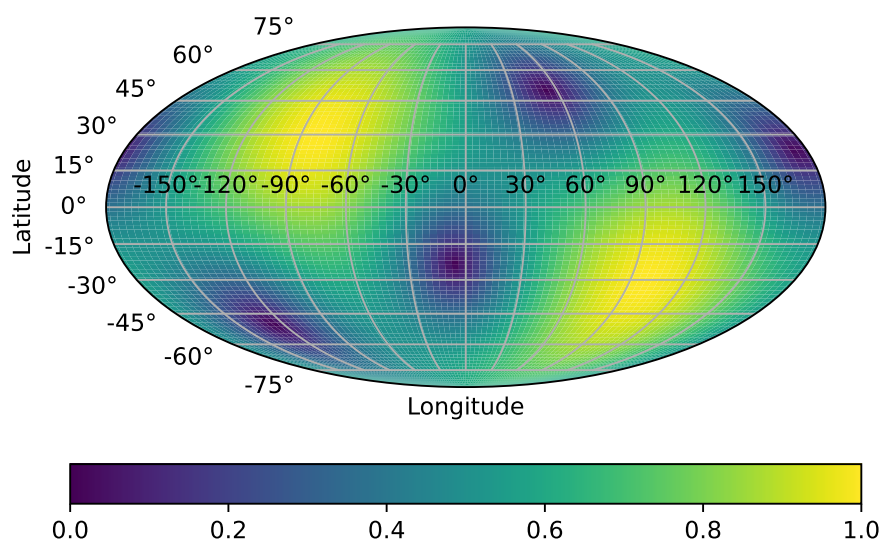


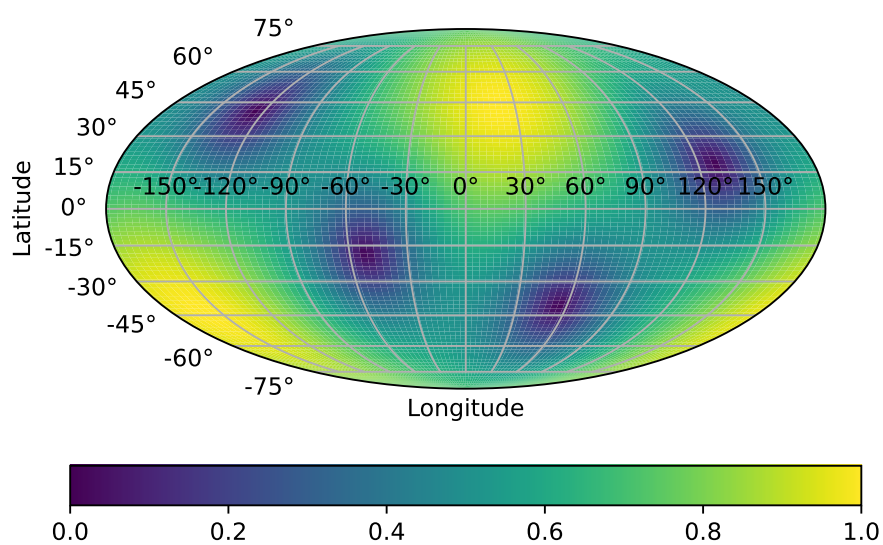
図 3.3: アンテナパターン。干渉計は $z = 0$ にあり、腕は x 方向と y 方向に伸びている。



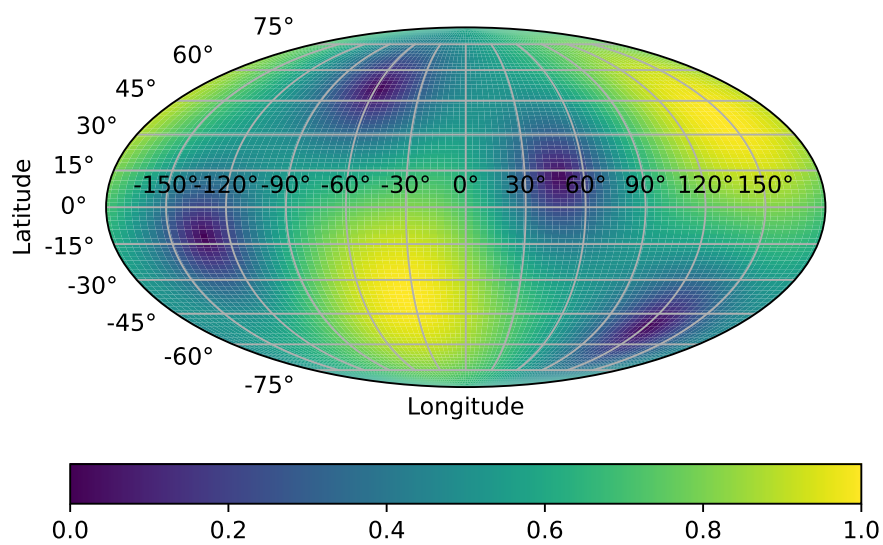
(a) LIGO H1 のアンテナパターン



(b) LIGO L1 のアンテナパターン



(c) Virgo のアンテナパターン



(d) KAGRA のアンテナパターン

図 3.4: 各検出器のアンテナパターン関数 $\sqrt{F_+^2(\theta, \phi, 0) + F_\times^2(\theta, \phi, 0)}$

3.3 重力波検出器の雑音

重力波検出器には、電気回路の雑音などの除去することのできる技術的な雑音の他に原理的な雑音が存在し、それによって感度が制限される。KAGRA の設計感度 [43] と代表的な雑音のスペクトルを図 3.5 に示した。図から周波数によって支配的な雑音は異なり、低周波数帯域から順に地面振動雑音、熱雑音、量子雑音が支配的であることが分かる。本節ではこれらの雑音について簡単に紹介する。

地面振動雑音

地上または地中にある重力波検出器は、地震や人間活動による地面振動の影響を受ける。地面振動雑音のスペクトルは周波数の二乗に反比例することが知られており、主に 10Hz 以下の低周波数帯域で感度を制限する雑音になる [44]。地面振動雑音は干渉計の鏡を振り子で吊るすことによって低減させることができる。実際、KAGRA では多段振り子を用いて防振を行っている。

熱雑音

熱雑音は鏡やそのコーティング、懸架系のブラウン運動によって生じ、10Hz から 100Hz 程度の帯域で感度を制限する。熱雑音を低減するために機械損失の小さい材料で鏡や懸架系を作ることが重要である。KAGRA では鏡を極低温に冷やし、極低温で機械損失の小さいサファイ

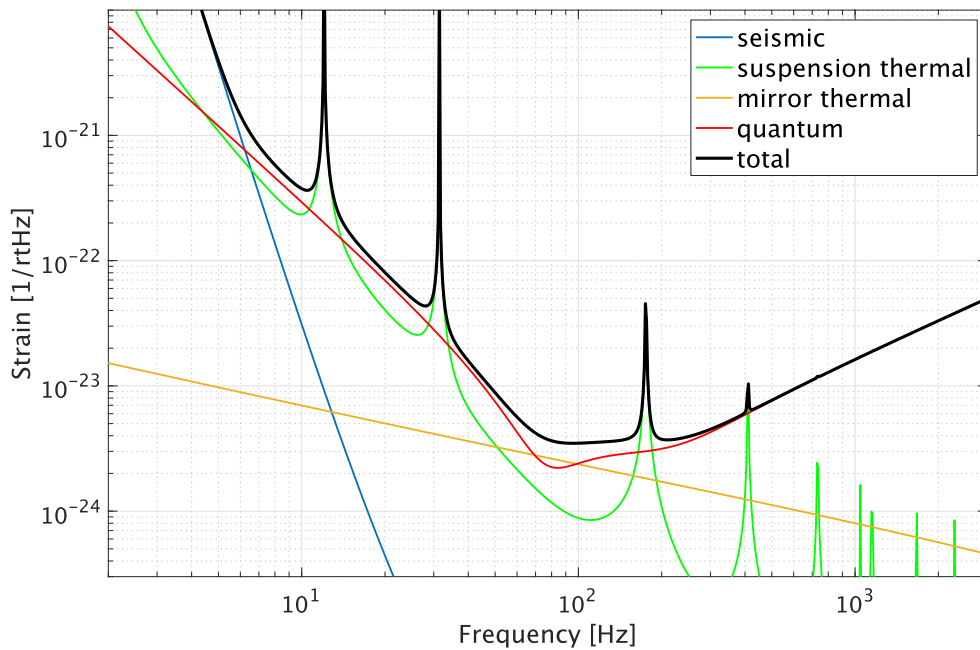


図 3.5: KAGRA の設計感度

アを鏡や懸架ワイヤーに用いている。

量子雑音

量子雑音はレーザー光の量子的な揺らぎによって生じる雑音であり、100Hz 以上の高周波数帯域で感度を制限する。輻射圧雑音と散射雑音の2つがあり、これらは光強度に対してトレードオフの関係にある。

輻射圧雑音は光子数の揺らぎによって光が鏡で反射されるときに鏡が光から受ける力 (輻射圧) が揺らぎ、鏡が動くというものである。輻射圧雑音はレーザー光強度の $1/2$ 乗に比例する。

散射雑音はフォトディテクタに入る光子数の揺らぎによって生じ、周波数依存性のない白色雑音である。スペクトルはレーザー光強度の $-1/2$ 乗に比例し、散射雑音を低減するために高出力のレーザー光源が使われる。

これらの量子雑音によって定められる干渉計の感度の限界を標準量子限界と呼ぶ。

第 4 章

重力波のデータ解析

重力波の検出において、ノイズの中から振幅の小さな信号を見つけ出すためのデータ解析手法が重要である。本章ではまずパワースペクトル密度やガウシアンノイズについて定義し、その性質を確認する。次に重力波の検出手法として用いられているマッチドフィルタやそれに基づくパラメータ推定と到来方向推定の原理について述べる。最後に、検出器で得られるデータの処理方法について簡単に述べる。本章は J. D. Creighton と W. G. Anderson の文献 [45] を参考にした。

4.1 パワースペクトル密度

検出器のノイズ $x(t)$ が確率密度関数 $p_X(x)$ で表されるとする。定常過程を考えるとアンサンブル平均 $\langle x \rangle$ は長時間平均と一致する (エルゴード性) :

$$\langle x \rangle := \int dx x p_X(x) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} dt x(t) \quad (4.1)$$

ここで

$$x_T(t) = \begin{cases} x(t) & (-T/2 \leq t \leq T/2) \\ 0 & (\text{otherwise}) \end{cases} \quad (4.2)$$

を定義すると式 (4.1) の積分区間は $(-\infty, \infty)$ にでき、 $x(t)^2$ の期待値は

$$\langle x^2 \rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\infty}^{\infty} dt x_T^2(t) \quad (4.3)$$

$$= \lim_{T \rightarrow \infty} \frac{2}{T} \int_0^{\infty} df |\tilde{x}_T(f)|^2 \quad (4.4)$$

$$= \int_0^{\infty} df S_x(f) \quad (4.5)$$

と表せる。 $x_T(t)$ の Fourier 変換を $\tilde{x}_T(f)$ とし、Parseval の等式を用いた。 $S_x(f)$ は (片側) パワースペクトル密度 (PSD) と呼ばれる量である。 PSD の定義から

$$\begin{aligned} S_x(f) &:= \lim_{T \rightarrow \infty} \frac{2}{T} |\tilde{x}_T(f)|^2 \\ &= \lim_{T \rightarrow \infty} \frac{2}{T} \int_{-T/2}^{T/2} dt x(t) e^{2\pi i f t} \int_{-T/2}^{T/2} dt' x(t') e^{-2\pi i f t'} \end{aligned} \quad (4.6)$$

と表せるが、ここで $\tau = t' - t$ として変形していくと

$$\begin{aligned} S_x(f) &= \lim_{T \rightarrow \infty} \frac{2}{T} \int_{-\infty}^{\infty} d\tau e^{-2\pi i f \tau} \int_{-T/2}^{T/2} dt x(t) x(t + \tau) \\ &= 2 \int_{-\infty}^{\infty} d\tau R_x(\tau) e^{-2\pi i f \tau} \\ &= 2\tilde{R}_x(f) \end{aligned} \quad (4.7)$$

となる． $R_x(\tau)$ は自己相関関数であり，次の式で定義される：

$$R_x(\tau) := \int_{-\infty}^{\infty} dt x(t) x(t + \tau) \quad (4.8)$$

したがって，片側 PSD は自己相関関数の Fourier 変換の 2 倍である．これは Wiener-Khinchin の定理と呼ばれている．PSD のもう 1 つの便利な表現を導出する．

$$\langle \tilde{x}^*(f') \tilde{x}(f) \rangle = \left\langle \int_{-\infty}^{\infty} dt x(t) e^{2\pi i f' t} \int_{-\infty}^{\infty} dt' x(t') e^{-2\pi i f t'} \right\rangle \quad (4.9)$$

において上と同様に $\tau = t' - t$ として式 (4.7) を用いると

$$\begin{aligned} \langle \tilde{x}^*(f') \tilde{x}(f) \rangle &= \int_{-\infty}^{\infty} dt e^{-2\pi i (f - f') t} \int_{-\infty}^{\infty} d\tau \langle x(t) x(t + \tau) \rangle e^{-2\pi i f \tau} \\ &= \frac{1}{2} \delta(f - f') S_x(f) \end{aligned} \quad (4.10)$$

と表せる．

4.2 ガウシアンノイズ

時間 T のノイズ $x(t)$ を Δt の間隔でサンプリングして $N = T/\Delta t$ 個の離散データ $\{x_j\}$ ($j = 0, \dots, N-1$) を得るとする． x_0, \dots, x_{N-1} が独立に正規分布に従うとき， $x(t)$ をガウシアンノイズと呼ぶ．以下では x_0, \dots, x_{N-1} が独立に平均 0，分散 σ^2 の正規分布に従うとする：

$$x_0, \dots, x_{N-1} \text{ i.i.d. } \sim \mathcal{N}(0, \sigma^2) \quad (4.11)$$

このとき $\{x_j\}$ の同時確率密度関数は

$$p(\{x_j\}) = \left(\frac{1}{2\pi\sigma^2} \right)^{N/2} \exp \left[-\frac{1}{2\sigma^2} \sum_{j=0}^{N-1} x_j^2 \right] \quad (4.12)$$

である．自己相関関数は独立性を用いると

$$R_x(k\Delta t) = \langle x_j x_{j+k} \rangle = \sigma^2 \delta_{k,0} \quad (4.13)$$

と表せるから，Wiener-Khinchin の定理 (4.7) より PSD は

$$\begin{aligned} S_x(f) &= 2 \int_{-\infty}^{\infty} d\tau R_x(\tau) e^{-2\pi i f \tau} \\ &= \lim_{\Delta t \rightarrow 0} 2\sigma^2 \Delta t \end{aligned} \quad (4.14)$$

となり、 f に依らない．このようなノイズを白色雑音 (white noise) と呼び、PSD が周波数に依るノイズを有色雑音 (colored noise) と呼ぶ．

PSD を用いると、同時確率密度関数で $\Delta t \rightarrow 0$ とした連続極限は

$$\begin{aligned} \lim_{\Delta t \rightarrow 0} \exp \left[-\frac{1}{2\sigma^2} \sum_{j=0}^{N-1} x_j^2 \right] &= \lim_{\Delta t \rightarrow 0} \exp \left[-\frac{1}{2\sigma^2 \Delta t} \sum_{j=0}^{N-1} x_j^2 \Delta t \right] \\ &= \exp \left[-\frac{1}{S_x} \int_{-T/2}^{T/2} dt x(t)^2 \right] \\ &\sim \exp \left[-\int_{-\infty}^{\infty} df \frac{|\tilde{x}(f)|^2}{S_x(f)} \right] \end{aligned} \quad (4.15)$$

と表せる．2行目から3行目では T が十分大きいとして Parseval の等式を用いた．この式は有色のガウシアンノイズに対しても成り立つ．有色ガウシアンノイズは周波数領域で一般に

$$\tilde{\gamma}(f) = \tilde{K}(f) \tilde{x}(f) \quad (4.16)$$

と表せる．式 (4.10) から $\gamma(t)$ の PSD は

$$S_\gamma(f) = |\tilde{K}(f)|^2 S_x(f) \quad (4.17)$$

となり、式 (4.15) と同様に

$$p_\gamma \propto \exp \left[-\int_{-\infty}^{\infty} df \frac{|\tilde{\gamma}(f)|^2}{S_\gamma(f)} \right] \quad (4.18)$$

となる．

ここで時系列データ $a(t), b(t)$ に対して内積 (a, b) を

$$(a, b) := 4 \operatorname{Re} \int_0^\infty df \frac{\tilde{a}(f) \tilde{b}^*(f)}{S_x(f)} \quad (4.19)$$

$$= 2 \int_{-\infty}^{\infty} df \frac{\tilde{a}(f) \tilde{b}^*(f)}{S_x(|f|)} \quad (4.20)$$

と定義するとガウシアンノイズの確率密度関数は

$$p_x \propto e^{-(x, x)/2} \quad (4.21)$$

であることが分かる．

4.3 マッチドフィルタ

本節ではノイズの中から波形を知っている重力波信号を見つけるための手法として用いられるマッチドフィルタについて述べる．

検出器の出力 $s(t)$ は重力波信号 $h(t)$ とノイズ $n(t)$ の和で表すことができる：

$$s(t) = h(t) + n(t) \quad (4.22)$$

ここではこれらの量を連続量とみなして計算する．波形を知っている重力波 $h(t)$ (テンプレート) の検出は次の仮説検定と考えることができる．

$$H_0 : s(t) = n(t) \quad \text{vs.} \quad H_1 : s(t) = h(t) + n(t) \quad (4.23)$$

H_0 は帰無仮説, H_1 は対立仮説を表す．ここでは尤度比検定を用いる．尤度比検定とは, 帰無仮説の下での尤度の最大値と対立仮説の下での尤度の最大値の比 Λ を統計検定量とする仮説検定であり, 単純二仮説 (帰無仮説と対立仮説のパラメータ空間がともに 1 点集合) の場合は一様最強力検定になる (Neyman–Pearson の補題 [46]).

実際に検定 (4.23) の尤度比を計算する．ノイズは定常で平均 0 のガウシアンノイズであると仮定する． H_0 のもとでは, $s(t) = n(t)$ がガウシアンに従うから前節の計算から尤度は $e^{-(s,s)/2}$ に比例する． H_1 のもとでは, $n(t) = s(t) - h(t)$ がガウシアンに従うから尤度は $e^{-(s-h,s-h)/2}$ に比例する．よって尤度比 Λ は

$$\Lambda = \frac{e^{-(s,s)/2}}{e^{-(s-h,s-h)/2}} = e^{(s,h)} e^{-(h,h)/2} \quad (4.24)$$

となるから検定 (4.23) の帰無仮説の棄却域はある定数 k, k' を用いて

$$\{s \mid \Lambda \geq k\} \iff \{s \mid (s, h) \geq k'\} \quad (4.25)$$

と構成される．この (s, h) をマッチドフィルタと呼び, 以下では m で表す:

$$m := (s, h) = 4 \operatorname{Re} \int_0^\infty df \frac{\tilde{s}(f) \tilde{h}^*(f)}{S_n(f)} \quad (4.26)$$

マッチドフィルタ m の統計的な性質を調べる．まず重力波信号 h がないとき, つまり $s(t) = n(t)$ のとき $m = (s, h) = (n, h)$ の期待値 $\langle m \rangle = \langle (n, h) \rangle$ は $\langle n \rangle = 0$ の仮定から 0 となる．分散 $\operatorname{Var}(m) = \langle m^2 \rangle$ は

$$\begin{aligned} \langle m^2 \rangle &= 4 \int_{-\infty}^\infty df \int_{-\infty}^\infty df' \frac{\langle \tilde{n}(f) \tilde{n}^*(f') \rangle \tilde{h}(f') \tilde{h}^*(f)}{S_n(|f|) S_n(|f'|)} \\ &= 4 \int_{-\infty}^\infty df \int_{-\infty}^\infty df' \frac{\frac{1}{2} S_n(|f|) \delta(f - f') \tilde{h}(f') \tilde{h}^*(f)}{S_n(|f|) S_n(|f'|)} \quad (\because (4.7)) \\ &= 2 \int_{-\infty}^\infty df \frac{\tilde{h}(f) \tilde{h}^*(f)}{S_n(|f|)} \\ &= (h, h) \end{aligned} \quad (4.27)$$

となる．したがって信号 h がないときのマッチドフィルタ m は平均 0, 分散 (h, h) のガウシアンに従う．

次に信号が存在するときを考える．振幅はテンプレート $h(t)$ の A 倍とする．つまり $s(t) = n(t) + Ah(t)$ である．このとき m, m^2 の期待値はそれぞれ

$$\langle m \rangle = \langle (n, h) \rangle + (Ah, h) = A(h, h) \quad (4.28)$$

$$\langle m^2 \rangle = \langle (n, h)^2 \rangle + 2(Ah, h) \langle (n, h) \rangle + (Ah, h)^2 = (h, h) + A^2(h, h)^2 \quad (4.29)$$

となるから分散は

$$\operatorname{Var}(m) = \langle m^2 \rangle - \langle m \rangle^2 = (h, h) \quad (4.30)$$

である。したがって信号 h があるときのマッチドフィルタ m は平均 $A(h, h)$, 分散 (h, h) のガウシアンに従う。

正規化されたマッチドフィルタ $\rho := m/\sqrt{(h, h)}$ は信号雑音比 (SNR) と呼ばれ, ガウシアンノイズのみのときの ρ は標準正規分布 $\mathcal{N}(0, 1)$ に従う。テンプレート h の A 倍の信号が存在するときは ρ の平均は $A\sqrt{(h, h)}$ になる。正規化されたマッチドフィルタである $\text{SNR}(\rho)$ がある閾値を超えたとき, 検定 (4.23) の帰無仮説は棄却され, 重力波信号が存在すると判断される。

4.4 パラメータ推定

重力波の波形はいくつかのパラメータによって特徴付けられる。例えば, コンパクト連星合体ではチャープ質量やスピンなどがパラメータである。パラメータが N 個あるとし, その集合を $\boldsymbol{\theta} = (\theta_1, \dots, \theta_N)$ と表す。パラメータ推定の目標は, ある仮説 H と確率密度 $p(\boldsymbol{\theta}|H)$ の下でのパラメータの集合 $\boldsymbol{\theta}$ の状態を記述することである。データ s が得られた後の $\boldsymbol{\theta}$ の事後確率は, Bayes の定理から

$$p(\boldsymbol{\theta}|s, H) = \frac{p(\boldsymbol{\theta}|H)p(s|\boldsymbol{\theta}, H)}{p(s|H)} \quad (4.31)$$

となる。あるパラメータ θ_i の事後確率はこれを周辺化することによって得られる：

$$p(\theta_i|s, H) = \int d\theta_1 \cdots d\theta_{i-1} d\theta_{i+1} \cdots d\theta_N p(\boldsymbol{\theta}|s, H) \quad (4.32)$$

この事後確率を用いて, あるパラメータ θ_i の期待値を

$$\langle \theta_i \rangle = \int d\theta_i \theta_i p(\theta_i|s, H) \quad (4.33)$$

と求めることができる。

このようなベイズ推定によるパラメータ推定は概念的には単純だが, パラメータ空間の次元や分析するデータ量に大きく依存する。重力波のパラメータ空間の次元は大きくデータ量も多いため, パラメータ空間のある固定されたサンプルによって推定することはできない。そこでマルコフ連鎖モンテカルロ法 [47, 48] やネストサンプリング [49] に基づいて, パラメータ空間から確率的にサンプリングをしてベイズ推定が行われる [50]。

4.5 到来方向推定

重力波の到来方向の推定は, 各検出器の到来時刻の差を用いて推定される。まず2つの検出器での推定を考える。単位球面上の位置 \mathbf{R} から到来するとし, 2つの検出器の相対位置を \mathbf{D} とすると, 2つの検出器間の信号の到来時刻の差は $T_1 - T_2 = \mathbf{D} \cdot \mathbf{R}$ と表せる。2つの検出器の到来時刻の精度を σ_1, σ_2 , 観測した到来時刻を t_1, t_2 と仮定すると推定方向 \mathbf{r} の事後分布は次のように表せる [51]。

$$p(\mathbf{r}|\mathbf{R}) \propto p(\mathbf{r}) \exp \left[-\frac{(\mathbf{D} \cdot (\mathbf{r} - \mathbf{R}))^2}{2(\sigma_1^2 + \sigma_2^2)} \right] \quad (4.34)$$

2つの検出器による推定では、検出器間の相対位置ベクトル \mathbf{D} に平行な方向にしか到来方向を限定することができない。

3つの検出器では、 $p(\mathbf{r}|\mathbf{R})$ は次のように表せる：

$$p(\mathbf{r}|\mathbf{R}) \propto p(\mathbf{r}) \exp \left[-\frac{1}{2} (\mathbf{r} - \mathbf{R})^\top \mathbf{M} (\mathbf{r} - \mathbf{R}) \right] \quad (4.35)$$

$$\mathbf{M} := \frac{\mathbf{D}_{12} \mathbf{D}_{12}^\top}{\sigma_{12}^2} + \frac{\mathbf{D}_{23} \mathbf{D}_{23}^\top}{\sigma_{23}^2} + \frac{\mathbf{D}_{31} \mathbf{D}_{31}^\top}{\sigma_{31}^2} \quad (4.36)$$

この方法を用いた3台の検出器による推定では検出器の平面に垂直な方向に1つ縮退がある。到来時刻だけでなく振幅を利用することによって縮退を解くことができるが、3台の干渉計全てが感度を持つ方向から到来するとは限らないため、到来方向を推定するためには少なくとも4台の検出器が必要である。

実際に用いられている到来方向推定の手法は低遅延の方法と高遅延の方法がある。低遅延の方法では BAYESTAR [33] と呼ばれるアルゴリズムが用いられる。BAYESTAR は各検出器のマッチドフィルタの結果を用いてベイズ推定を行い、数十秒から1分以内でパラメータ推定とほぼ同じ精度で推定ができる。高遅延の方法では、LALInference [50] というアルゴリズムが用いられる。この方法では、全ての未知パラメータをベイズ推定するため、計算コストが高い。例えばコンパクト連星合体の場合パラメータは15個あり、数時間から数日の時間がかかる。

4.6 信号処理

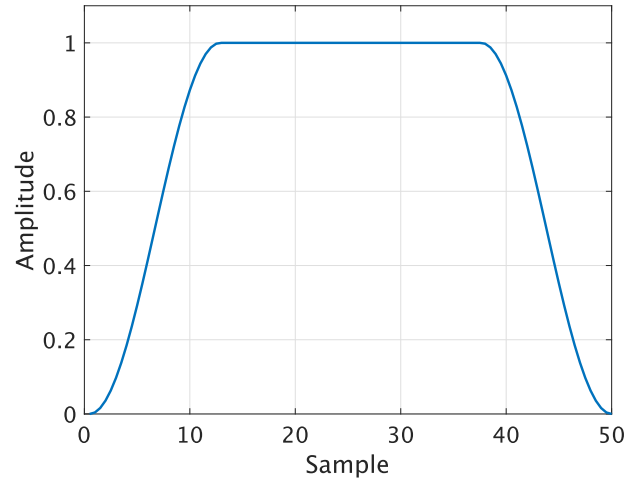
本節では、検出器で得られる生の時系列データを解析するための信号処理の方法について述べる。

4.6.1 PSD の推定

まずデータから PSD の推定を行う。PSD は Welch の方法 [52] を用いて推定することができる。Welch の方法は高速 Fourier 変換を用いるが、Fourier 変換を行うとき、時系列データの一部を切り取るにより不連続性が生じるため、データをそのまま Fourier 変換するとスペクトルに余分な寄与が生まれる。切り出した時系列データに窓関数を掛けることによって、その余分な寄与を減少させることができる。窓関数は矩形窓、Hann 窓、Hamming 窓など様々なものがあるが、重力波データに対しては Tukey 窓が有効であることが知られている [53]。Tukey 窓はパラメータ $\alpha \in [0, 1]$ を用いて次のように定義される。図 4.1 は $\alpha = 0.5$ の Tukey 窓をプロットしたものである。

$$w[n] = \begin{cases} 1 & (\alpha N/2 \leq n \leq (1 - \alpha/2)N) \\ \frac{1}{2} \left(1 - \cos \left(\frac{2\pi n}{\alpha N} \right) \right) & (0 \leq n \leq \alpha N/2, (1 - \alpha/2)N \leq n \leq N) \end{cases} \quad (4.37)$$

Tukey 窓を掛けられたデータから、Welch の方法を用いて PSD を推定する。その PSD の平方根である ASD が次項で述べる白色化に用いられる。

図 4.1: Tukey 窓 ($\alpha = 0.5$)

4.6.2 白色化

次に、時系列データのノイズを減らすために白色化 (whitening) という手法が使われる。白色化は時系列データ $s(t)$ の Fourier 変換 $\tilde{s}(f)$ を、ノイズの ASD の推定値で割り、それを逆 Fourier 変換して時間領域に戻す操作である：

$$s(t) \xrightarrow{\text{FFT}} \tilde{s}(f) \longrightarrow \tilde{s}_{\text{whitened}}(f) = \frac{\tilde{s}(f)}{\sqrt{S_n(f)}} \xrightarrow{\text{iFFT}} s_{\text{whitened}}(t) \quad (4.38)$$

白色化によって PSD がフラットに近くなる。

白色化された後のデータはバンドパスフィルタを用いて周波数の小さなノイズや周波数の大きなノイズが除去される。

図 4.2 に GW150914 の合体時刻前後 2 秒間の H1 でのデータを用いて以上の処理を行なった結果を示した。ここではバンドパスフィルタの通過帯域は 30Hz から 350Hz までとした。バンドパスフィルタを通した後のデータには重力波信号が現れている。

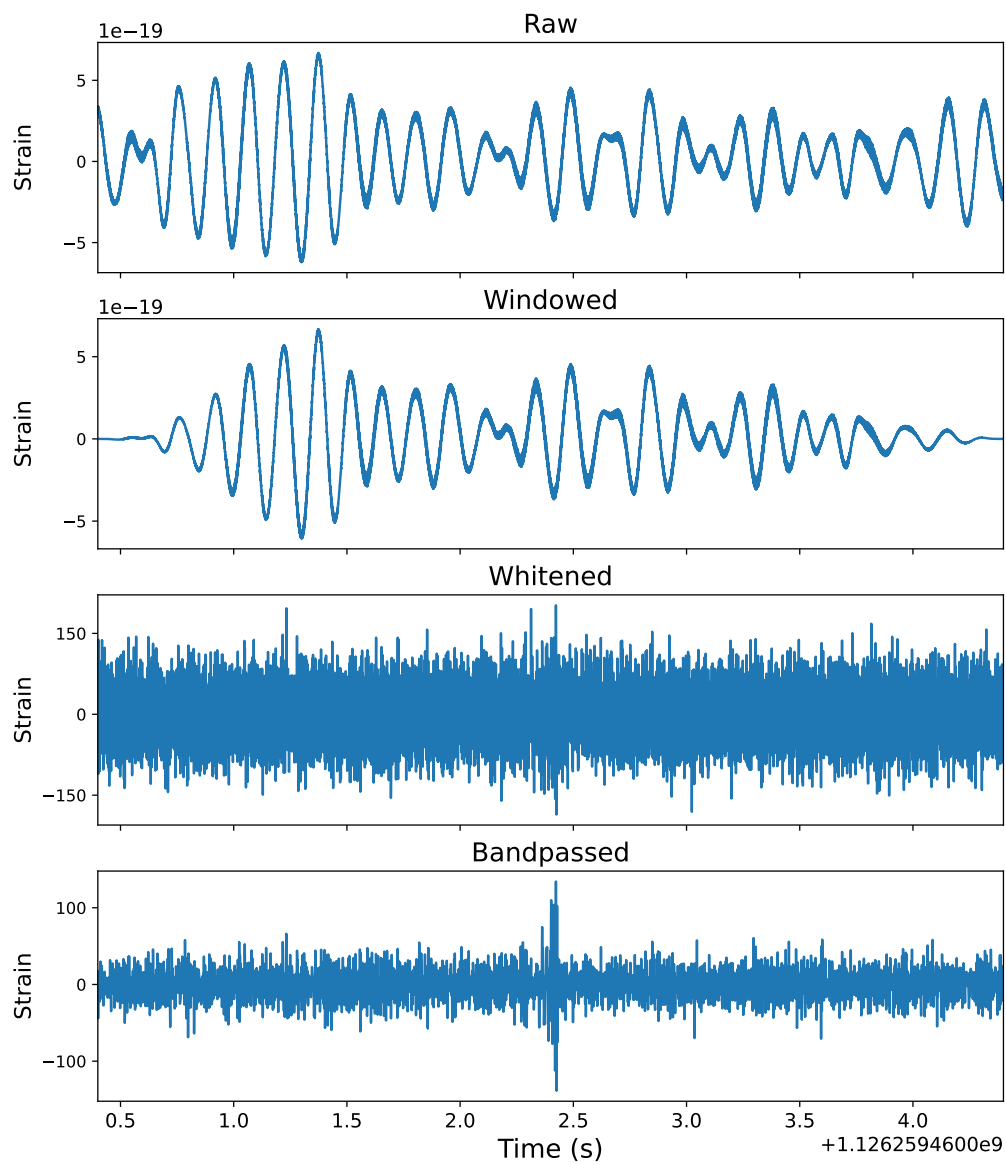


図 4.2: GW150914 のデータ処理結果

第5章

ニューラルネットワーク

本章では、本研究で用いたニューラルネットワークの理論について述べる．まず教師あり学習の定義や用語を確認し、ニューラルネットワークの学習に用いる最適化手法を概説する．次に、最も基本的なニューラルネットワークである多層パーセプトロン (MLP) 及び畳み込みニューラルネットワーク (CNN) について述べ、それを利用した Temporal Convolutional Network (TCN) を紹介する．本章は J. Berner らの論文 [54] と I. J. Goodfellow らの文献 [55] を参考にした．

5.1 教師あり学習

入力空間 \mathcal{X} と出力空間 \mathcal{Y} があるとし、 $(x, y) \in \mathcal{X} \times \mathcal{Y}$ は未知の確率分布 $P(x, y)$ に従うランダム変数とする．さらに x と y は未知の関数 g によって $g(x) = y$ という関係があるとする．パラメータ $\theta \in \Theta$ を用いて、ある x に対する y の予測値 $\hat{y} = f_{\theta}(x)$ を返す関数 $f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y}$ を定義する．この f_{θ} によって仮説空間 $\mathcal{F} = \{f_{\theta}\}$ が構成される．たとえば、 x と y が線形であると仮定すると仮説空間は

$$\mathcal{F} = \{f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y} \mid f_{\theta}(x) = \theta_1 x + \theta_2\} \quad (5.1)$$

である．学習とは、仮説空間 \mathcal{F} の中から g を最も良く推定するひとつの関数 f_{θ} を選ぶことである．この選択は $P(x, y)$ に独立に従う m 組のランダム変数の集合である訓練データ (training data) $\mathcal{S} = \{(x_1, y_1), \dots, (x_m, y_m)\}$ に基づいて行われる．

ある x に対する正解値 y_i と予測値 \hat{y}_i の差は損失関数 $\mathcal{L}(y_i, \hat{y}_i)$ によって測られる．各訓練データに対して損失関数を求め、その平均をとったものは経験損失あるいは訓練誤差 (training loss) と呼ばれる：

$$\mathcal{R}_s(f) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(y_i, \hat{y}_i) \quad (5.2)$$

経験損失最小化アルゴリズムでは \mathcal{F} の中から \mathcal{R}_s を最も小さくする \hat{f}_{θ} が f_{θ} の予測関数として選ばれる：

$$\hat{f}_{\theta} = \arg \min_{f_{\theta} \in \mathcal{F}} \mathcal{R}_s(f_{\theta}) \quad (5.3)$$

この \hat{f}_{θ} を用いて、新しいデータ $x \in \mathcal{X}$ に対する $y \in \mathcal{Y}$ を $\hat{y} = \hat{f}_{\theta}(x)$ と予測することがで

きる。

教師あり学習は大きく回帰問題と分類問題と2種類に分けられる。

回帰問題では、出力 y は連続値である： $\mathcal{Y} \subset \mathbb{R}$ 。損失関数は2乗誤差が用いられることが多い：

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2 \quad (5.4)$$

分類問題では出力 y は離散値をとる： $\mathcal{Y} \subset \mathbb{N}$ 。例えば分類したいクラスが k クラスあるとき、 $\mathcal{Y} = \{0, 1, \dots, k-1\}$ である。実際のモデルでは、まず各クラスに属する確率を表すベクトル $z \in [0, 1]^k$ を出力し、確率が最大となるクラスを選ぶ： $\hat{y} = \arg \max z$ 。損失関数は交差エントロピー (cross entropy) が用いられる：

$$\mathcal{L}(z, \hat{z}) = - \sum_{c=0}^{k-1} z_c \log \hat{z}_c \quad (5.5)$$

モデルの学習するために、まずパラメータの初期値を選び、訓練誤差を計算する。それを最小化するためのパラメータを求めたいが、一般に解析的にそれを求めることは不可能であるため、最適化アルゴリズムを用いてパラメータを何度も更新し、訓練誤差を小さくしていくという手法が取られる。そのパラメータの更新過程では同じ訓練データが何度も繰り返し用いられる。その回数はエポック (epoch) と呼ばれる。

ニューラルネットワークの学習では、過学習 (overfitting) と呼ばれる現象がよく起きる。過学習とはモデルが訓練データに過剰に適合してしまい、未知のデータに対する予測精度が低くなってしまいうことである。モデルが過学習をしているかを判定するために、訓練データの他にバリデーションデータ (validation data) を用いる。訓練誤差と同様にバリデーションデータ誤差も計算し、それを見てハイパーパラメータ (モデルが学習するパラメータではなく、学習率など人が設定するパラメータ) を調整してモデルを学習し、最後にテストデータを用いてモデルの性能を評価する。

5.2 最適化手法

本節では代表的な最適化アルゴリズムについて述べる。

パラメータの更新にはいくつか方法があり、それらは勾配法に基づいている。勾配法では、最小化したい関数 \mathcal{L} の微分係数を用いて以下の式に従って繰り返しパラメータを更新する。

$$\theta^{(i)} = \theta^{(i-1)} - \eta \nabla \mathcal{L}(\theta^{(i-1)}) \quad (5.6)$$

$\eta > 0$ は学習率と呼ばれる。

本節では勾配法に基づいたアルゴリズムである確率的勾配降下法 (SGD) [56], Momentum [57], Adagrad [58], Adam [59] について述べる。

5.2.1 確率的勾配降下法 (SGD)

訓練データ全てを使って訓練誤差を計算して勾配を計算し、パラメータを更新する方法 (最急降下法) は計算コストが高い。訓練データからランダムにいくつかをサンプリングして訓練

誤差を計算し、勾配法を用いてパラメータを更新することで1回の更新の計算コストを抑えることができる。この最適化アルゴリズムはSGDと呼ばれる。SGDのアルゴリズムをアルゴリズム1に示した。ここではサイズ m' のミニバッチ (訓練データの部分集合) をランダムに選んでパラメータを更新している。

過学習を抑えるために、 L_2 正則化と呼ばれる手法が使われる。これは最小化したい関数 \mathcal{L} に $(\gamma/2)\theta^2$ を加えたものを新たに \mathcal{L} として扱うものである。これは (5.6) において $\nabla\mathcal{L}(\theta^{(i)})$ の代わりに

$$\hat{\nabla}\mathcal{L}(\theta^{(i)}) = \nabla\mathcal{L}(\theta^{(i)}) + \gamma\theta^{(i)} \quad (5.7)$$

を用いて

$$\theta^{(i)} = \theta^{(i-1)} - \eta\hat{\nabla}\mathcal{L}(\theta^{(i-1)}) \quad (5.8)$$

と更新する方法とみなせる。 $\gamma > 0$ は weight decay と呼ばれる。

アルゴリズム 1 SGD

Input: Learning rate η

Input: Initial parameter $\Theta^{(0)}$

for $i = 1, \dots, k$ **do**

 Draw a random subset $S' = \{(x_1, y_1), \dots, (x_{m'}, y_{m'})\} \subset S$

 Update $\Theta^{(i)} \leftarrow \Theta^{(i-1)} - \eta \frac{1}{m'} \sum_{j=1}^{m'} \nabla\mathcal{L}(f(x_j; \Theta^{(i-1)}), y_j)$

end for

5.2.2 Momentum

SGD は極小値付近で振動しやすく、収束が遅いという欠点がある [60]。Momentum は SGD を極小値方向に加速させ、振動を減衰させる手法である。Momentum を導入した SGD では次の式に従って更新を行う。

$$\mathbf{v}^{(i)} = \mu\mathbf{v}^{(i-2)} + \hat{\nabla}\mathcal{L}(\theta^{(i-1)}) \quad (5.9)$$

$$\theta^{(i)} = \theta^{(i-1)} - \eta\mathbf{v}^{(i)} \quad (5.10)$$

この方法では、 $\mu \in [0, 1), \eta > 0, \gamma > 0$ がハイパーパラメータである。 \mathbf{v} の初期値は 0 とする。

5.2.3 Adagrad

SGD では全てのパラメータを同じ学習率を用いて更新していたが、学習率はパラメータごとに違う値を用いた方が収束は速い。そこで Adagrad では各パラメータに対する勾配の大きさに応じて学習率に重みをかける。更新は次の式に従って行われる。

$$\mathbf{g}^{(i)} = \nabla\mathcal{L}(\theta^{(i-1)}) \quad (5.11)$$

$$\theta^{(i)} = \theta^{(i-1)} - \frac{\eta}{\sqrt{\mathbf{G}^{(i)} + \epsilon}} \odot \mathbf{g}^{(i)} \quad (5.12)$$

$\varepsilon > 0$ はゼロ除算を避けるための小さな定数であり, 通常 10^{-8} が使われる [60]. $\mathbf{G}^{(i)}$ の j 成分は

$$G_j^{(i)} := \sum_{k \leq i} (g_j^{(k)})^2 \quad (5.13)$$

で定義され, 式 (5.12) の右辺第 2 項は要素ごとの積を表す.

5.2.4 Adam

Adam は Momentum のアイデアと Adagrad のアイデアを組み合わせたものであり, 次の式に従って更新する.

$$\mathbf{g}^{(i)} = \nabla \mathcal{L}(\boldsymbol{\theta}^{(i)}) \quad (5.14)$$

$$\mathbf{m}^{(i)} = \beta_1 \mathbf{m}^{(i-1)} + (1 - \beta_1) \mathbf{g}^{(i)} \quad (5.15)$$

$$\mathbf{v}^{(i)} = \beta_2 \mathbf{v}^{(i-1)} + (1 - \beta_2) \mathbf{g}^{(i)} \odot \mathbf{g}^{(i)} \quad (5.16)$$

$$\hat{\mathbf{m}}^{(i)} = \frac{\mathbf{m}^{(i)}}{1 - \beta_1^i} \quad (5.17)$$

$$\hat{\mathbf{v}}^{(i)} = \frac{\mathbf{v}^{(i)}}{1 - \beta_2^i} \quad (5.18)$$

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(i)} + \epsilon}} \odot \hat{\mathbf{m}}^{(i)} \quad (5.19)$$

ハイパーパラメータは $\beta_1, \beta_2 \in [0, 1)$ と $\eta > 0$ である. Adam のアルゴリズムをアルゴリズム 2 に示した.

アルゴリズム 2 Adam

Input: Learning rate η

Input: Exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0, 1)$

Input: Small constant ϵ

Input: Initial parameter $\boldsymbol{\Theta}^{(0)}$

Initialize $\mathbf{m}_0 \leftarrow 0, \mathbf{v}_0 \leftarrow 0$

for $i = 1, \dots, k$ **do**

 Draw a random subset $S' = \{(x_1, y_1), \dots, (x_{m'}, y_{m'})\} \subset S$

$\mathbf{g}^{(i)} \leftarrow \frac{1}{m'} \sum_{j=1}^{m'} \nabla \mathcal{L}(f(x_j; \boldsymbol{\Theta}^{(i-1)}), y_j)$

$\mathbf{m}^{(i)} \leftarrow \beta_1 \mathbf{m}^{(i-1)} + (1 - \beta_1) \mathbf{g}^{(i)}$

$\mathbf{v}^{(i)} \leftarrow \beta_2 \mathbf{v}^{(i-1)} + (1 - \beta_2) \mathbf{g}^{(i)} \odot \mathbf{g}^{(i)}$

$\hat{\mathbf{m}}^{(i)} \leftarrow \mathbf{m}^{(i)} / (1 - \beta_1^i)$

$\hat{\mathbf{v}}^{(i)} \leftarrow \mathbf{v}^{(i)} / (1 - \beta_2^i)$

 Update $\boldsymbol{\Theta}^{(i)} \leftarrow \boldsymbol{\Theta}^{(i-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(i)} + \epsilon}} \odot \hat{\mathbf{m}}^{(i)}$

end for

5.3 多層パーセプトロン (MLP)

ニューラルネットワークは一般にノードとエッジから成る。ノードはニューロンを表し、エッジはニューロン間の結合を表す。ノードは層として積み重なり、ある層のノードの値は前の層のノード、層間の結合を表す重み、及び各ノードに作用する活性化関数によって計算される。始めの層は入力層、最後の層は出力層であり、その間は中間層あるいは隠れ層と呼ばれ、隠れ層のあるニューラルネットワークは深層ニューラルネットワークと呼ばれる。最も基本的なニューラルネットワークは、ある層のすべてのノードが次の層のノードに結合する全結合型のニューラルネットワークである。これを多層パーセプトロン (MLP) と呼び、その各層を全結合層と呼ぶ。

多層パーセプトロンの構造は層の数、各層のノード数と活性化関数によって特徴付けられる。層の数を $L+1 \in \mathbb{N}$ とし、各層のノード数を並べたベクトルを $N = (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$ とする。つまり入力 N_0 次元、出力 N_L 次元である。第 ℓ 層の各ノードに対して要素ごとに作用する活性化関数 $\sigma_\ell : \mathbb{R} \rightarrow \mathbb{R}$ を定義する。学習するパラメータは重み行列 $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ ($1 \leq \ell \leq L$) とバイアス $b^{(\ell)} \in \mathbb{R}^{N_\ell}$ ($1 \leq \ell \leq L$) である。従って学習するパラメータの総数は

$$P(N) = \sum_{\ell=1}^L (N_\ell N_{\ell-1} + N_\ell) \quad (5.20)$$

で与えられる。ニューラルネットワークは N_0 次元の入力と $P(N)$ 次元のパラメータから N_L 次元のベクトルを出力する関数 Φ :

$$\Phi : \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \rightarrow \mathbb{R}^{N_L} \quad (5.21)$$

である。学習する全パラメータをまとめて $\theta = (\theta^{(\ell)})_{\ell=1}^L = (W^{(\ell)}, b^{(\ell)})_{\ell=1}^L$ と表すと、入力 x が与えられたときの出力は次のように計算される :

$$\bar{\Phi}^{(1)}(x, \theta) = W^{(1)}x + b^{(1)} \quad (5.22)$$

$$\bar{\Phi}^{(\ell+1)}(x, \theta) = W^{(\ell+1)}\bar{\Phi}^{(\ell)}(x, \theta) + b^{(\ell+1)} \quad (5.23)$$

$$\Phi^{(\ell)}(x, \theta) = \sigma_\ell(\bar{\Phi}^{(\ell)}(x, \theta)) \quad (5.24)$$

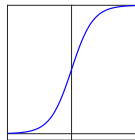
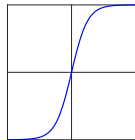
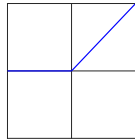
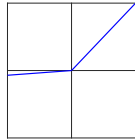
出力は $\Phi(x, \theta) = \Phi^{(L)}(x, \theta)$ である。活性化関数 σ_ℓ は表 5.1 の関数が用いられることが多い。

MLP の構造の例を図 5.1 に示した。このモデルは層の数は $L+1=3$ であり、ノード数は $N = (3, 5, 2)$ である。したがってパラメータの個数は $P(N) = 5 \cdot 3 + 5 + 2 \cdot 5 + 2 = 32$ 個である。

5.4 畳み込みニューラルネットワーク (CNN)

畳み込みニューラルネットワーク (CNN) は畳み込み層を用いたニューラルネットワークであり、特に画像認識や音声認識で優れた精度を出している。本節では CNN で用いられる畳み込み層、プーリング層、バッチ正規化について述べる。また、ニューラルネットワークの学習に用いられる Dropout について述べる。

表 5.1: 主な活性化関数

名前	定義	プロット
sigmoid	$\frac{1}{1 + e^{-x}}$	
tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	
ReLU	$\max(0, x)$	
parametric ReLU (PReLU)	$\max(ax, x)$ for $a \geq 0$	

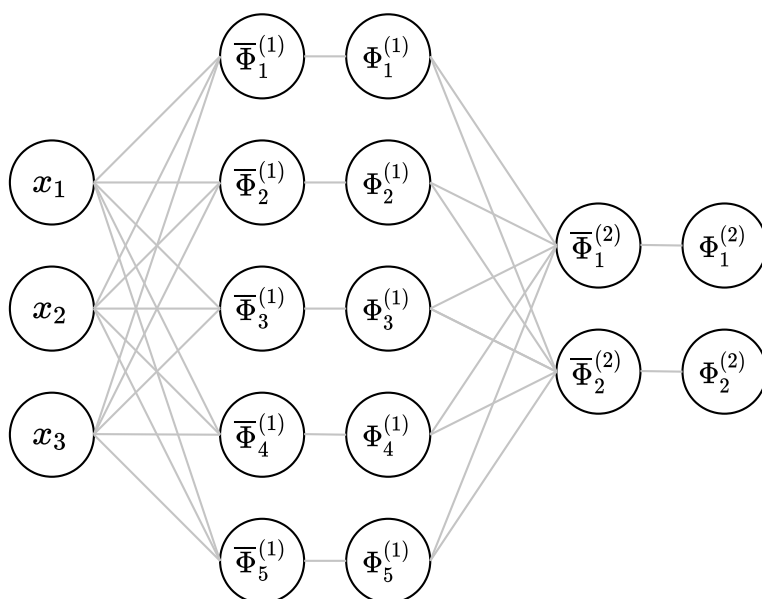


図 5.1: MLP の構造の例

5.4.1 畳み込み層

前節で導入した全結合層はある層の1つのノードが前の層のノード全てを用いて計算されるため、層の数が増えるとパラメータの数が膨大になる。また、画像を扱う際は2次元データを1次元に並べて入力するため、画像の空間的な相関関係などが失われる。これらの弱点を克服

するのが畳み込み層である。畳み込み層は、一般に全結合層よりパラメータの数が少なく、また、入力データの空間的な特徴を保つこともできる。

畳み込み層で行われる具体的な演算を説明する。ここでは1次元の畳み込み層について述べるが、2次元以上の場合も同様である。全結合層と異なり、畳み込み層はある層の1つのノードが前の層のいくつかのノードとフィルタを用いて計算される。入力データ x と重み行列 W 、バイアスベクトル b を用いて畳み込み層の出力は

$$\Phi_{\text{Conv}}(x, W, b) = Wx + b \quad (5.25)$$

と表せる。入力層の次元を N_0 、フィルタのサイズ (カーネルサイズとも呼ばれる) を k ($< N_0$) とすると、 $(N_0 - k + 1, N_0)$ 行列 W とサイズ $N_0 - k + 1$ のバイアスベクトル b は

$$W = \begin{pmatrix} w_1 & \cdots & w_k & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & w_1 & \cdots & w_k \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_{N_0-k+1} \end{pmatrix} \quad (5.26)$$

と表せる。つまり、出力層の1つ1つのノードの計算に用いられるフィルタの重みは共通である。 $N_0 = 5, k = 3$ の場合を図 5.2 に示した。図の右側の赤色のノードは前の層の赤線で繋がっている3つのノードから計算される。青色、緑色も同様である。この例ではフィルタを1つと仮定しているが、実際は1つの畳み込み層で複数のフィルタが用いられる。例えば図 5.2 でフィルタを100個用いた場合、入力層はサイズ5の1次元データだが、畳み込み層の出力はサイズ3の1次元データが100個できることになる。これを出力が100チャンネルであると言う。複数のフィルタを用いることで、複数の特徴量を学習することができる。

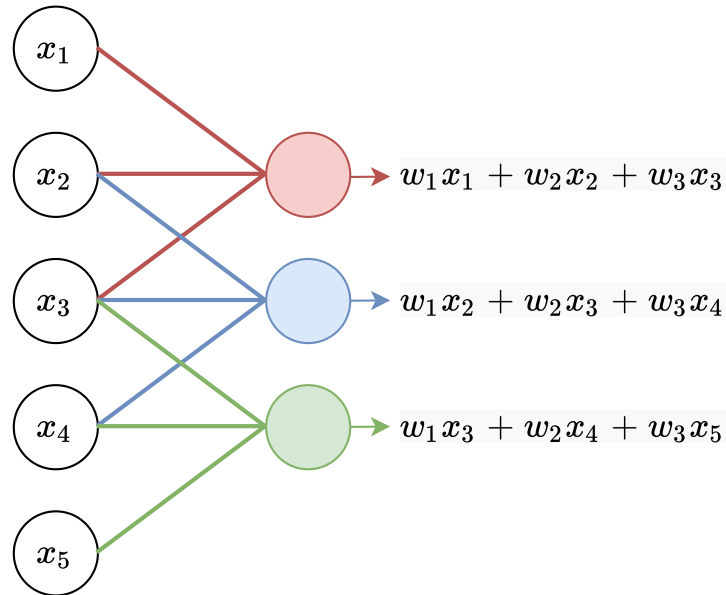


図 5.2: 畳み込み層の例

5.4.2 プーリング層

プーリング層は CNN の精度を上げるために、通常畳み込み層の後に置かれる層である。プーリング層の 1 種である最大値プーリングは、前の層のデータの連続したいくつかのノードの最大値を取るものである。図 5.3 はサイズ 2 の最大値プーリング層の例である。2 つの要素ごとに最大値をとるため入力サイズ 4 に対し出力サイズはその半分の 2 になっている。他には最大値の代わりに平均値をとる平均値プーリングもある。いずれの場合も、プーリング層は対象となるノードの最大値や平均値をとる演算であるから学習するパラメータを持たない。またチャンネルごとに最大値や平均値を計算するため、チャンネル数も変化しない。

プーリング層を用いることによって、より抽象的な特徴を抽出することができたり、少し位置のズレがあるデータに対してロバストなモデルにしたりすることができる。また、データのサイズを小さくすることができるため、モデル全体のパラメータ数を減らし、計算コストを抑えることができる。

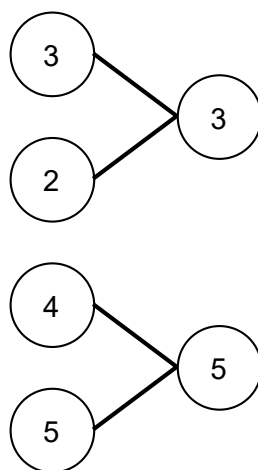


図 5.3: 最大値プーリングの例

5.4.3 バッチ正規化

ニューラルネットワークの学習では、通常入力データを平均が 0、標準偏差が 1 になるように正規化するが、それを層ごとに行うことで学習を安定させたり速めたりするテクニックがバッチ正規化 (Batch Normalization) [61] である。バッチ正規化は、訓練データのミニバッチの同じチャンネル毎に、平均が 0、標準偏差が 1 になるよう正規化を行う。ミニバッチのサイ

ズを m とすると次のように表される：

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad (5.27)$$

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (5.28)$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (5.29)$$

ϵ は 0 除算を避けるための小さな値である．バッチ正規化では，このように正規化された \hat{x}_i に対し，さらに次のような変換を行う：

$$y_i \leftarrow \gamma \hat{x}_i + \beta \quad (5.30)$$

これは正規化されたデータを β シフトさせ，スケールを γ 変化させる変換であり，もともと持っていた非線形性が正規化によって失われないようにするための変換である． γ と β はバッチ正規化層で学習されるパラメータである．

5.4.4 Dropout

Dropout [62] は CNN 以外のニューラルネットワークでも用いられるが，本項でまとめるおく．

Dropout は過学習を抑制するために用いられる手法であり，学習時にいくつかのノードの出力をランダムに 0 にする（ノードを不活性化する）というものである．Dropout 層のハイパーパラメータは 1 つあり，それを p とすると，その層の各ノードの出力にパラメータ $1 - p$ の Bernoulli 分布に従う数が掛けられることを意味する（図 5.4）．つまり確率 $1 - p$ でノードの出力が元の値を保ち，確率 p で不活性化される．

Dropout を用いることは入力にノイズを加えることと同じような意味を持つ．Dropout を用いることによってニューラルネットワークが特定のノードやエッジに依存せず，入力の特徴を表現するための様々な方法を学習することになり，汎化性能の高いモデルになる．

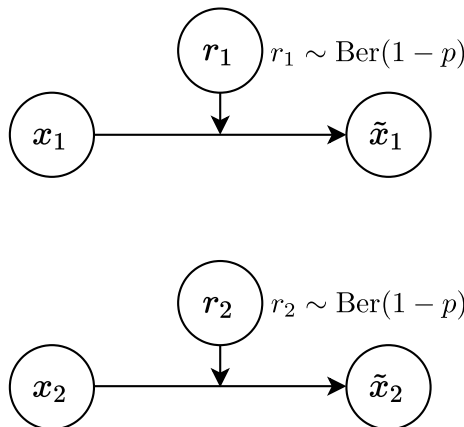


図 5.4: Dropout の模式図

5.5 Temporal Convolutional Network (TCN)

Temporal convolutional network (TCN) [35] は畳み込みニューラルネットワークに基づいたモデルであり、時系列データや自然言語などの系列データのモデリングに有効なニューラルネットワークモデルである。系列データに対しては、従来は LSTM [63] や GRU [64] などの RNN に基づくモデルが用いられていたが、S. Bai ら [35] は 1 次元畳み込みを用いた TCN というモデルが特定の系列データのタスクにおいて RNN ベースのモデルを上回ることを示した。本節では TCN の構造について述べる。

5.5.1 Dilated Causal Convolution

TCN は dilated causal convolution という畳み込みを用いている。dilated convolution とは dilation が 2 以外の畳み込み層のことを表す。dilation は畳み込み層のハイパーパラメータの 1 つであり、畳み込みフィルタとの積をとる入力データの間隔を表す。図 5.5 の 2 つのカーネルサイズ 3 の畳み込みフィルタのうち左は dilation が 1、つまり連続する 3 つの要素を畳み込むフィルタである。一方右側は dilation が 2、つまり 1 つ間を開けた要素 3 つを畳み込むフィルタである。TCN では dilation を $2^0, 2^1, 2^2, \dots$ と 2 のべき乗で増やしていく。そうすることで出力層の 1 つのノードに影響を与える入力層のノードの数 (受容野) を増やすことができる。

causal な畳み込みとは、出力がそれ以前の時刻のデータのみで計算される畳み込みのことである。通常の CNN では左右から畳み込んでいくが、時系列予測の場合は今の時刻より先のデータは使えないため、左右非対称な畳み込みになる。このような畳み込みにするのは時系列予測の場合のみで、分類問題などでは non-causal な畳み込みにすれば良い。以上のような特徴を持った TCN の畳み込みの模式図を図 5.6 に示した。この図は畳み込み層は 3 層用いており、dilation はそれぞれ $2^0, 2^1, 2^2$ となっている。また、causal な畳み込みになっている。

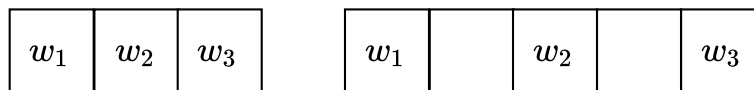


図 5.5: dilation の異なる畳み込みフィルタ。ともにカーネルサイズは 3 だが、左は dilation が 1、右は dilation が 2 である。

5.5.2 残差接続

TCN のもう 1 つの特徴は残差接続 (residual connection) を用いていることである。残差接続は ResNet [65] で開発された手法であり、図 5.7 のような構造をしている。畳み込み層に x が入力され、 $F(x)$ が出力されるとすると、その $F(x)$ に元の入力 x を足すという操作が残差接続である。ニューラルネットワークの層を増やしていくと、層の最後から逆伝播で計算される勾配が入力層に近い層で小さくなりすぎることにより、学習が進まなくなる勾配消失問題が

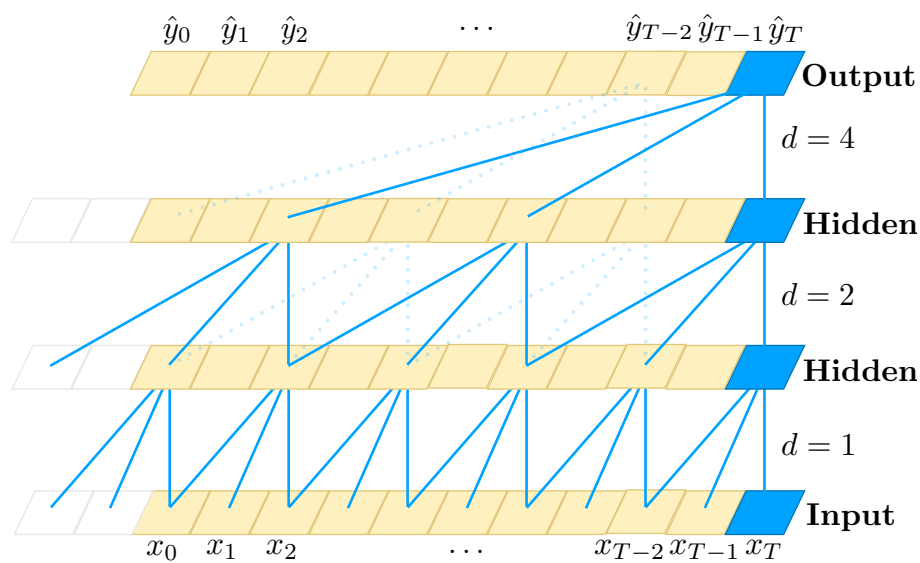


図 5.6: Dilated causal convolution ([35] より引用). d は dilation を表す. \hat{y}_T が時刻 T 以下の x から畳み込まれている.

生じるが、この残差接続を用いることによってこの問題を解決することができる。

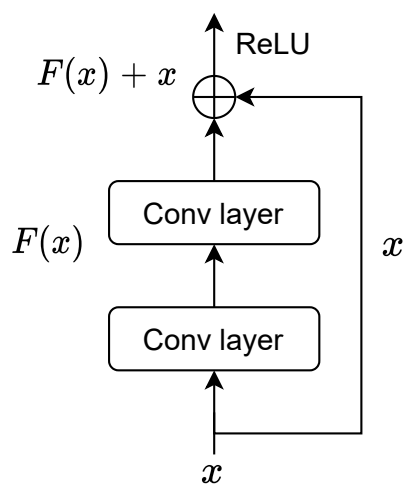


図 5.7: 残差接続

第 6 章

重力波の到来方向推定

本研究では、ブラックホール連星合体からの重力波の LIGO H1, LIGO L1, Virgo の計 3 台の検出器のデータ及び KAGRA を含む 4 台の検出器のデータを用いて到来方向の推定を行った。本研究で考えたのは図 6.1 の橙色の四角で囲まれた部分である。低遅延で到来方向推定まで行うために、検出はマッチドフィルタではなく機械学習で行うことを想定している。

本章では、まずデータ生成方法と天空の分割方法について述べ、次に 3 つの推定手法とその訓練方法について述べる。

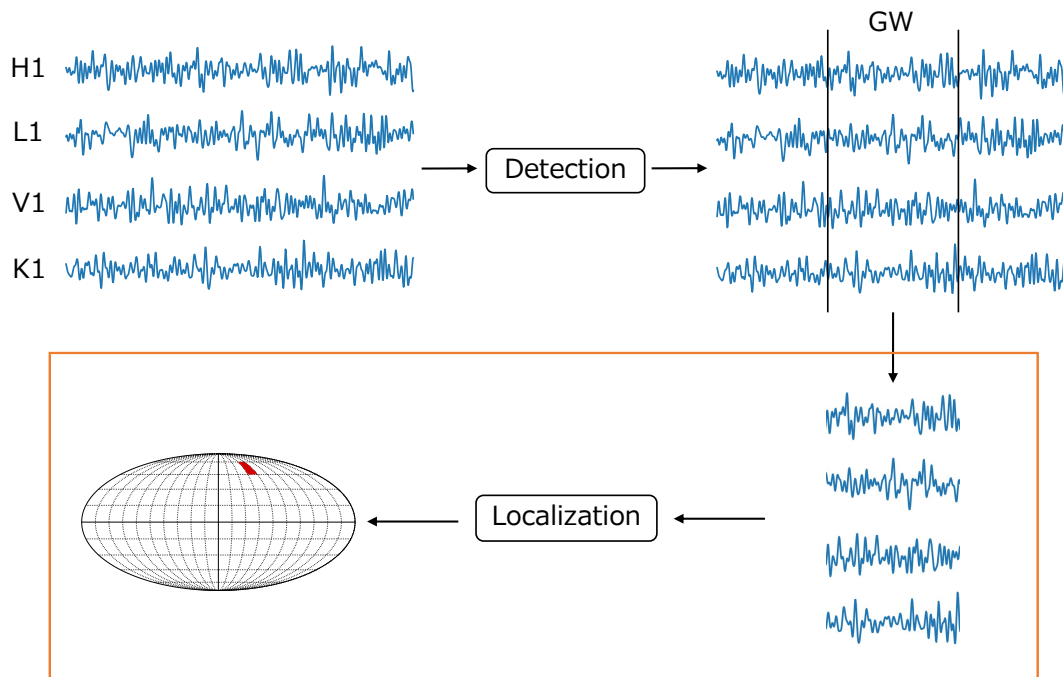


図 6.1: 本研究で考えた問題。橙色の四角で囲まれた部分を行った。

6.1 データ生成

6.1.1 波形のシミュレーション

コンパクト連星合体からの重力波は PyCBC [66] というソフトウェアを用いてシミュレーションすることができる。特に機械学習用のデータセットの生成には T. Gebhard ら [67] が開発した ggwd [68] という PyCBC のラッパーが便利である。本研究では ggwd を用いてデータを生成したが、公開されている ggwd のコードでは LIGO H1 と LIGO L1 からのデータのみしか生成できないため、Virgo と KAGRA のデータも生成できるようにコードを書き換えた。

波形は effective one-body 法 [69] によるモデル SEOBNRv4 [70] を用いて時間領域で生成した。用いたパラメータは表 6.1 に示した。サンプリングレート 2048Hz で生成し、生成した波形はイベント時刻の 0.2 秒前から 0.05 秒後までの計 0.25 秒に切り取った。つまり各検出器での重力波データは $2048 \text{ Hz} \times 0.25 \text{ s} = 512$ 個の値から成る配列である。この 0.25 秒という長さは、用いたパラメータの範囲での重力波のインスパイラル、合体、リングダウンを含む時間である [32]。

表 6.1: 重力波信号のシミュレーションに用いたパラメータ

Mass1	$[30M_{\odot}, 80M_{\odot}]$
Mass2	$[30M_{\odot}, 80M_{\odot}]$
Spin1z	$[0, 0.998]$
Spin2z	$[0, 0.998]$
Right ascension	$[0, 2\pi]$
Declination	$[-\pi/2, \pi/2]$
Coalescence phase	$[0, 2\pi]$
Inclination	$[0, \pi]$
Polarization	$[0, 2\pi]$
Network SNR	$[10, 50]$

6.1.2 ノイズ

ノイズは各検出器の感度を用いてシミュレーションしたガウシアンノイズを用いた。LIGO, Virgo, KAGRA の PSD はそれぞれ LIGO Document Control Center Portal [71], F. Acernese らの論文 [72], JGW Document Server [43] に公開されている設計感度を用いた。PyCBC にも `pycbc.psd.analytical.KAGRA` などの感度が収録されているが、これは A. Manzotti と A. Dietz の論文 [73] の式を用いてフィッティングしたものであり、フィッティングが正確でないと判断したため、本研究ではこれらは用いなかった。各検出器の感度は図 6.2 に示した。

シミュレーションした波形は、ネットワーク SNR が表 6.1 の範囲からランダムにサンプリ

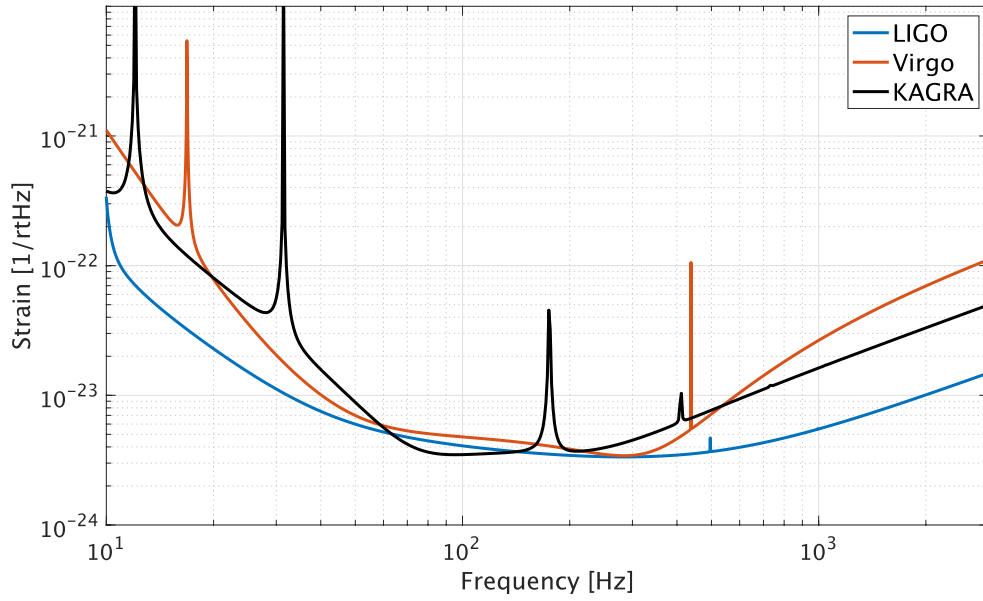


図 6.2: 用いた各検出器の設計感度

ングされた値になるように定数倍される．ネットワーク SNR は各検出器での SNR の 2 乗の和の平方根で定義される．スケーリングされた信号がノイズに挿入された後，ホワイトニングを行い，バンドパスフィルタを用いて 18 Hz 以下と 500 Hz 以上の周波数を除去した．以上のようにして訓練データを 20 万個，バリデーションデータを 4 万個，テストデータを 4 万個生成した．生成したデータのうちの 1 つを図示したものが図 6.3 である．

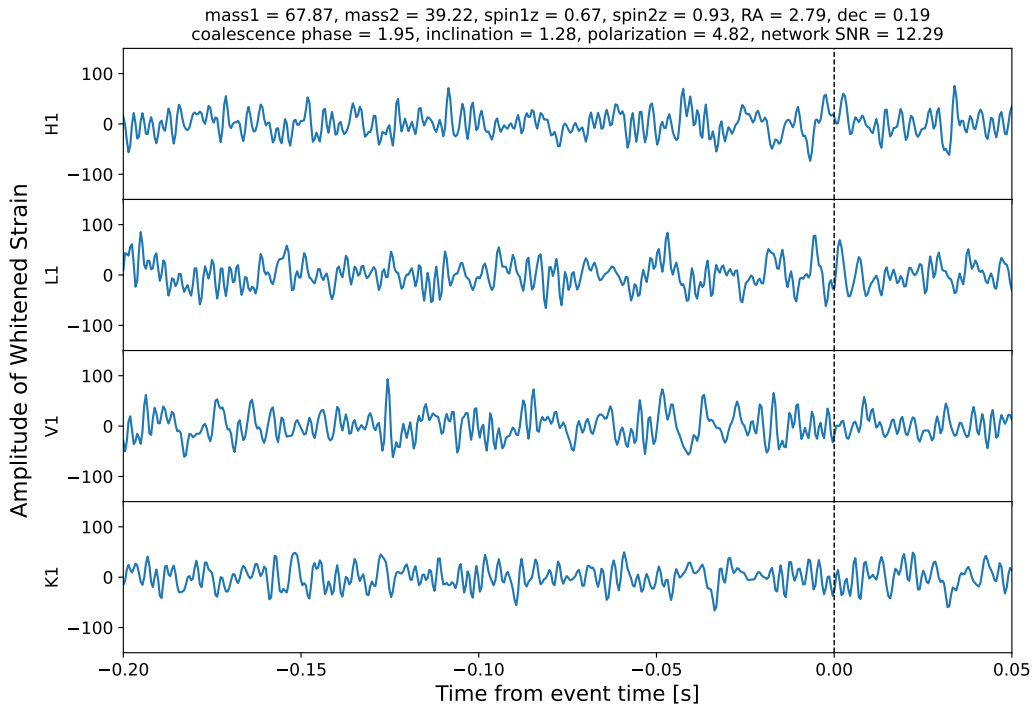


図 6.3: 訓練データのサンプル．

6.2 分割方法

到来方向は Declination (Dec) と Right Ascension (RA) の 2 つの角度で特徴付けられる．到来方向推定を分類問題として扱うために，天空をいくつかのセクタに分割する必要がある．本研究では 2 つの分割方法を試し，推定精度を比較した．方法 A は Dec 方向と RA 方向をそれぞれ同じ角度で分割する方法であり，方法 B は HEALPix (Hierarchical Equal Area isoLatitude Pixelation) [74] というアルゴリズムによる分割である．それぞれの分割方法を以下で説明する．

6.2.1 分割方法 A: 角度一定

分割方法 A は Dec 方向と RA 方向をそれぞれ同じ角度で分割し，Dec の小さいセクタから順にラベルを付けるというものである．Dec は $-\pi/2$ から $\pi/2$ の値をとり，RA は 0 から 2π までの値をとるから，Dec 方向を k 分割した場合，RA 方向は $2k$ 分割され，合計 $2k^2$ セクタできることになる．本研究では $k = 3$ から $k = 10$ まで，つまり 18 セクタから 200 セクタの分割を考えた．図 6.4 の左は 60 度ずつの分割，つまり 18 セクタへの分割であり，右は 18 度ずつの分解，つまり 200 セクタへの分割を表す．

角度一定で分割した場合，各セクタの立体角は一定ではない．表 6.2 に 18 分割のときと 200 分割の場合のそれぞれのセクタの最小立体角と最大立体角などを示した．200 分割の場合，立体角はセクタによって最大約 13 倍異なることが分かる．訓練データやテストデータの重力波は全ての方向から一様に到来してくると仮定しているため，分割方法 A では各セクタに割り振られた重力波サンプルの個数は異なる．テストデータ 4 万個を分割方法 A で 200 分割した場合の各ラベルのサンプル数の分布を図 6.5 に示した．Dec の絶対値が大きいセクタは立体角が小さく，到来する重力波の個数も少ないことが読み取れる．

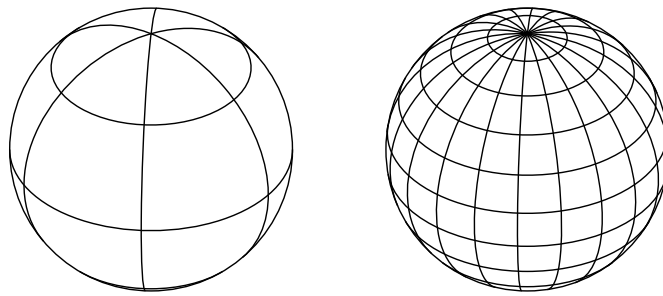


図 6.4: 分割方法 A による分割の例．左は 18 セクタ，右は 200 セクタへの分割を表す．

表 6.2: 分割方法 A のセクタ数と立体角

k	セクタ数 ($2k^2$)	最小セクタの立体角	最大セクタの立体角	平均立体角
3	18	1719 deg^2	3438 deg^2	2292 deg^2
10	200	50.48 deg^2	637.4 deg^2	206.3 deg^2

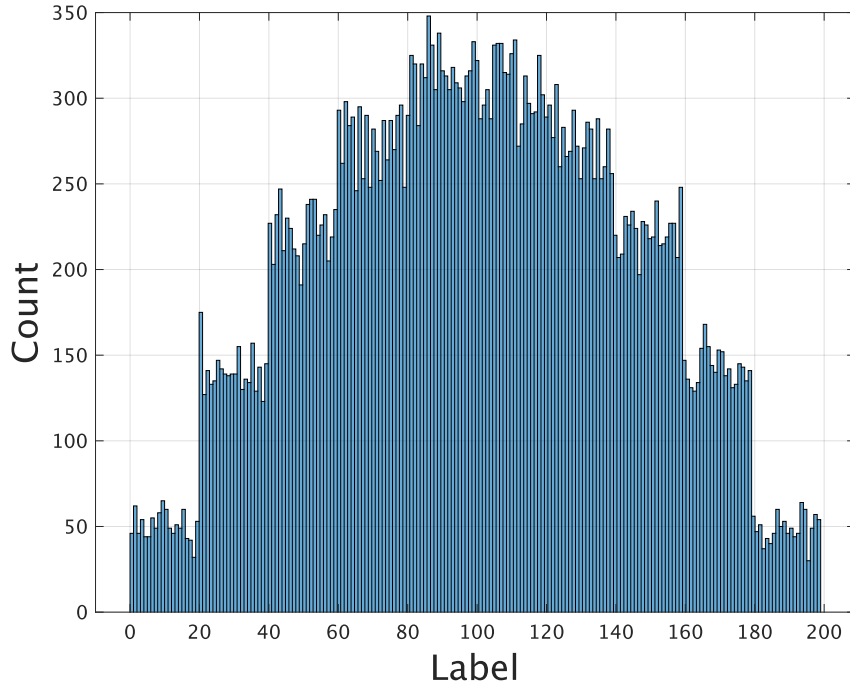


図 6.5: 分割方法 A で 200 分割したときの各ラベルに分類されるテストデータのサンプルの数.

6.2.2 分割方法 B: HEALPix

方法 B は HEALPix による分割である．HEALPix は主に天文データに用いられるアルゴリズムであり，`healpy` [75] という Python パッケージで用いることができる．HEALPix の特徴は各セクタの面積が一定なことと，階層的な分割になっていることである．最も粗い分割は 12 分割であり，次に細かい分割は 12 セクタをさらに 2^2 個に分けたものである．これを繰り返すため，セクタ数は 12×2^{2k} と表せる．表 6.3 に分割数と立体角を示した．また HEALPix の分割の様子を図 6.6 に示した．HEALPix は全てのセクタの面積が一定になるように分割するため，各ラベルに分類される重力波の個数はほぼ一定である．テストデータ 4 万個を分割方法 B で 192 分割した場合の各ラベルのサンプル数の分布を図 6.7 に示したが，実際に各ラベルに属する重力波の個数はほぼ一定である．

表 6.3: HEALPix の分割数と立体角

k	$N_{\text{side}} = 2^k$	$N_{\text{pix}} = 12N_{\text{side}}^2$	Ω_{pix}
0	1	12	3438 deg^2
1	2	48	859.4 deg^2
2	4	192	214.9 deg^2
3	8	768	53.71 deg^2

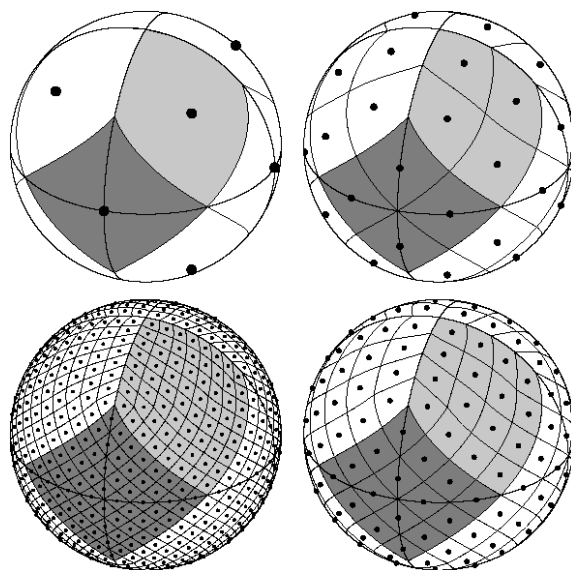


図 6.6: HEALPix による分割の例 ([74] より引用). 左上から時計回りに 12 セクタ, 48 セクタ, 192 セクタ, 768 セクタへの分割を表す.

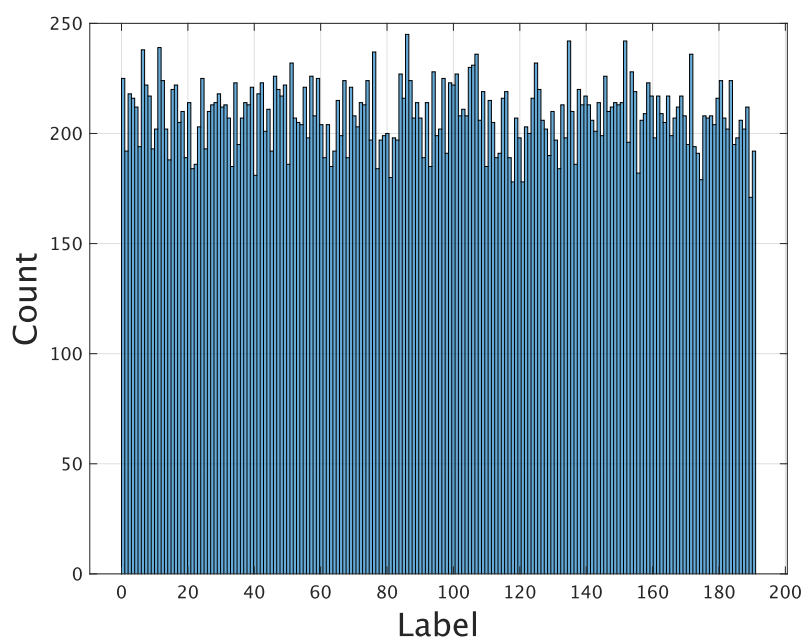


図 6.7: 分割方法 B で 200 分割したときの各ラベルに分類されるテストデータのサンプルの数.

6.3 推定手法

前節で述べた分割方法によって分けられた領域に各重力波を分類するためのニューラルネットワーク手法として次の 3 つを考えた. 1 つ目は先行研究 [32] で提案された MLP を用いる方法, 2 つ目は TCN を用いる方法, 3 つ目は 1 つ目と 2 つ目の方法を組み合わせた方法である.

本節でこの 3 つの手法を詳しく説明する。

6.3.1 手法 1：MLP

手法 1 は先行研究 [32] による方法である。まず，MLP の入力に用いる特徴量を計算する。

特徴量

特徴量は以下の 7 種類である。尚，特徴量の計算の前に各信号を中心化し，平均を 0 にした。

- 特徴量 1: 遅延時間。2 つの信号の遅延時間は相互相関関数によって求めることができる。信号 $x(t), y(t)$ の相互相関関数は次で定義される：

$$C_{xy}(t) := \int_{-\infty}^{\infty} d\tau x^*(\tau)y(\tau+t) \quad (6.1)$$

相互相関関数は 2 つの信号の時間をずらして内積をとったものであるから，相互相関関数が最大となる時間 t は 2 つの信号の遅延時間と考えることができる。今回は相互相関関数の絶対値が最大値をとる t を特徴量の 1 つ目とした。

- 特徴量 2: 相互相関関数の最大値。特徴量 1 で求めた時間での相互相関関数の値は，2 つの信号を遅延時間の分シフトしたときの類似度を表すため，重要な特徴である。そこで 2 つ目の特徴量は特徴量 1 の時間での $C_{xy}(t)$ の値とした。
- 特徴量 3: 解析信号の遅延時間。ノイズの大きいデータでの推定精度を上げるため，Hilbert 変換によって得られる解析信号を用いた。 $x(t)$ の Hilbert 変換は次のように定義される：

$$\mathcal{H}[x(t)] := \frac{1}{\pi} \int_{-\infty}^{\infty} d\tau \frac{x(\tau)}{t-\tau} \quad (6.2)$$

$x(t)$ の解析信号は実部に $x(t)$ ，虚部に $\mathcal{H}[x(t)]$ を持つ複素信号である：

$$x_a(t) := x(t) + i\mathcal{H}[x(t)] \quad (6.3)$$

Hilbert 変換は Scipy [76] の `signal.hilbert` を用いて計算することができる。特徴量 3 は特徴量 1 において 2 つの信号を解析信号に変えたものである。

- 特徴量 4: 解析信号の相互相関関数の最大値。特徴量 4 は特徴量 2 において 2 つの信号を解析信号に変えたものである。
- 特徴量 5: 合体時刻の振幅比。これを求めるためには各検出器のデータから合体時刻を求める必要がある。本研究では合体時刻は振幅の絶対値が最大になる時刻とした。
- 特徴量 6: 合体時刻の位相差。ある時刻での位相は解析信号の虚部，すなわち信号の Hilbert 変換で表される。特徴量 5 と同じ合体時刻を用いて，その時間での Hilbert 変

換の値の差を求めた．

- 特徴量 7: コサイン類似度．コサイン類似度は 2 つの信号の類似度を表す．信号 $x[n], y[n]$ のコサイン類似度は次のように定義される：

$$\cos \theta(x, y) = \frac{\sum_n x[n]y[n]}{\sqrt{\sum_n x[n]^2} \sqrt{\sum_n y[n]^2}} \quad (6.4)$$

各信号は中心化されているため，コサイン類似度は Pearson の相関係数に一致する．

これらの 7 種類の特徴量は全て 2 つのデータの組み合わせによって計算されるため，3 台の検出器で推定する場合は (H1, L1), (H1, V1), (L1, V1) の組み合わせによって計 ${}_3C_2 \times 7 = 21$ 個の特徴量ができる．4 台では計 ${}_4C_2 \times 7 = 42$ 個の特徴量ができる．MLP に入力する前に，これらの特徴量はそれぞれ平均が 0，標準偏差が 1 になるように標準化した．

モデル

用いた MLP のモデルを図 6.8 に示した．この図は 200 分割の場合を表しており，最後の出力が 200 次元になっている．先行研究 [32] と層の数やノード数が異なるが，この構造でも先行研究のモデルと同じまたは少し良い精度が得られることを確認した．図の Linear は全結合層を表す．3 台の場合入力次元は 21 次元であるからパラメータの総数は

$$(21 + 1) \times 512 + (512 + 1) \times 256 + (256 + 1) \times 256 + (256 + 1) \times 200 + 3 = 259787 \quad (6.5)$$

個である．最後の +3 は 3 つの PReLU 層によるものである．また Dropout のパラメータは全て $p = 0.2$ とした．

モデルは PyTorch [77] を用いて実装した．損失関数はクロスエントロピー誤差を用いた．また，最適化手法に Adam を用いて重みを更新した．学習率の初期値は 10^{-3} としたが，PyTorch の `ReduceLROnPlateau` を用いて学習の途中で減少させた．バッチサイズは 4000 とし，200 エポック学習した．また，最もバリデーションデータの正解率が高かったエポックでのモデルの重みを用いてテストを行った．

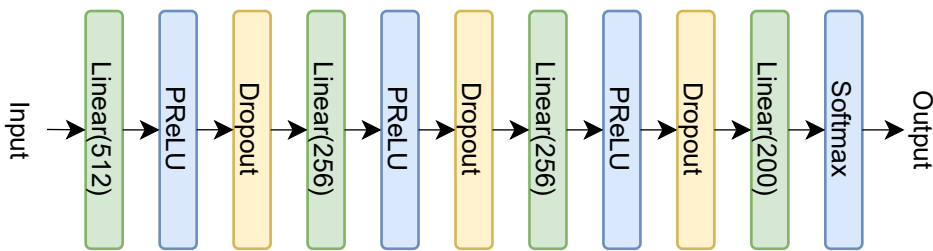


図 6.8: 手法 1 で用いた MLP モデル

6.3.2 手法 2：TCN

2 つ目の手法では生の時系列データを用いて TCN を学習した．用いた TCN のモデルは図 6.9 に示した．7 個の temporal block を積み重ね， i 番目の temporal block の dilation を 2^i

とし、カーネルサイズは全て 3 とした．ここでは TCN を時系列予測ではなく分類のために使うため、non-causal な畳み込みを用いた．入力データは -1 から 1 の間の値になるように規格化した．入力は 3 台の検出器のデータを用いるときは 3 チャンネル、4 台の検出器のデータを用いるときは 4 チャンネルあり、それぞれの temporal block の出力は 64 チャンネルとした．最後の線形層は分割数のサイズのテンソルを出力する．検出器 3 台、200 分割のときのパラメータ数は 176285 個となった．

TCN は S. Bai ら [35] による Pytorch の実装を利用した．損失関数、最適化手法、学習率は手法 1 の MLP と同様である．重みの初期化には He の初期値 [78] を用いた．バッチサイズ 512 のミニバッチを用いて 30 エポック学習し、バリデーションデータの正解率が最も高かったエポックでのモデルの重みでテストした．

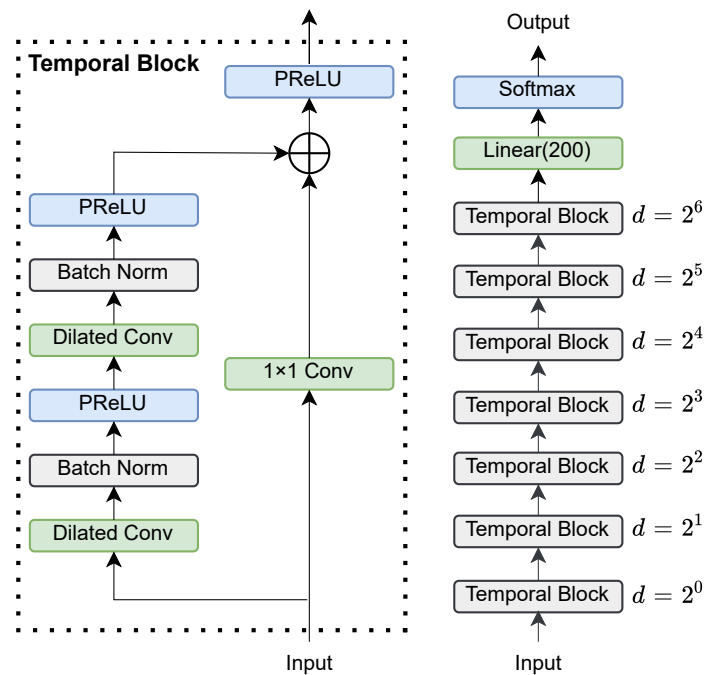


図 6.9: 手法 2 で用いた TCN モデル

6.3.3 手法 3：組み合わせ

手法 3 は手法 1 と手法 2 の組み合わせである．ニューラルネットワークのモデルを組み合わせる方法はバギング [79] やスタッキング [80] などいくつかの方法があるが、ここでは 2 つのモデルの出力の最大値や平均値を用いるシンプルな方法を用いた．具体的には以下の 2 通りの方法を試した．

1 つ目の方法は 2 つのモデルの出力確率のうち最も大きい確率のクラスを選ぶというものである．例えば 3 クラス分類においてモデル A が出力する確率が $[0.6, 0.3, 0.1]$ 、モデル B の出力が $[0.3, 0.2, 0.3]$ であれば、この中で最も大きい確率はモデル A の 0 番目のクラスであるため、0 と予測する．

2 つ目は 2 つのモデルの出力確率を重みをつけて平均し、その値が最も大きいクラスを選ぶ

というものである．重みはバリデーションデータで最も正解率が高くなるものを選び，テストデータでその重みを使って検証した．

バリデーションデータでの正解率は 2 つ目のモデルの方が高かったため，次章で紹介する結果は 2 つ目の方法を用いている．

第 7 章

結果

本章では、第 6 章で述べた手法を用いて行った到来方向推定の結果を述べる。まず LIGO H1, LIGO L1, Virgo の 3 台の検出器による推定結果をまとめ、推定手法 1 から 3 の正解率を比較する。次に、最も正解率の良い手法を用いて 3 台の検出器による推定精度と KAGRA を含む 4 台の推定精度を比較した結果を紹介する。また、Dec と正解率の関係や SNR と正解率の関係を分析した結果を述べる。最後に GW170814 を模倣したデータでの推定結果を述べる。

7.1 3 台の検出器による推定結果

本節では 2 つの分割方法のそれぞれに対して、LIGO H1, LIGO L1, Virgo の 3 台の検出器による推定結果を紹介する。

7.1.1 分割方法 A: 角度一定

角度一定で 200 分割したときの手法 1 の MLP 及び手法 2 の TCN の損失関数と正解率の推移はそれぞれ図 7.1, 図 7.2 のようになった。MLP では最初の 50 エポックほどまで損失関数が減少し、そこからはほぼ一定の値になっている。TCN のバリデーションデータの損失関数は 10 エポックほどまで減少して、そこからほぼ一定の値が続いているが、23 エポックを超えたところで訓練データの損失関数が減少している一方でバリデーションデータの損失関数は上昇傾向になっている。ここでは過学習が始まっていると考えられる。バリデーションデータの正解率が最も高いエポックでの重みを保存しているため、この過学習している部分はテストには影響しない。

各手法での分割数に対する正解率を図 7.3 に示した。手法 2 (TCN) の正解率は先行研究 [32] の手法である手法 1 を 1 ~ 3 % 上回っている。手法 3 は手法 1 と手法 2 の重み付き平均である。この重みはバリデーションデータから手法 1 の出力に対しては 0.6, 手法 2 の出力に対しては 0.4 と決定した。手法 3 の正解率は手法 1 から 5 ~ 8 % 改善している。また、分割数が大きくなるほど手法 1 と手法 3 の差も開いている。

推定時間は手法 1 は 0.0053 秒, 手法 2 は 0.2652 秒, 手法 3 は 0.2771 秒だった。

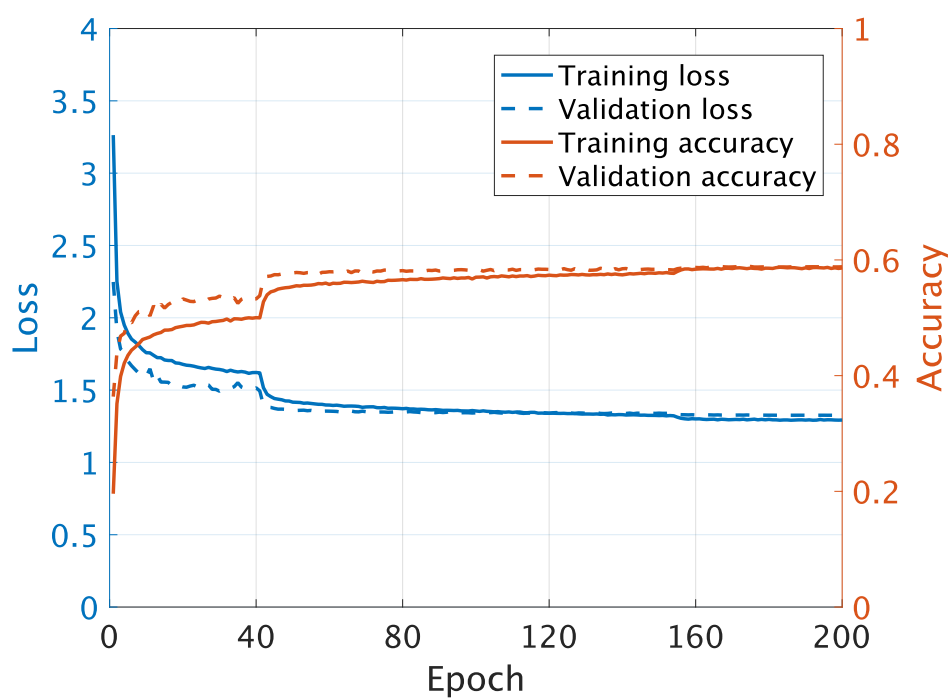


図 7.1: MLP の学習曲線. 200 分割で検出器は HLV .

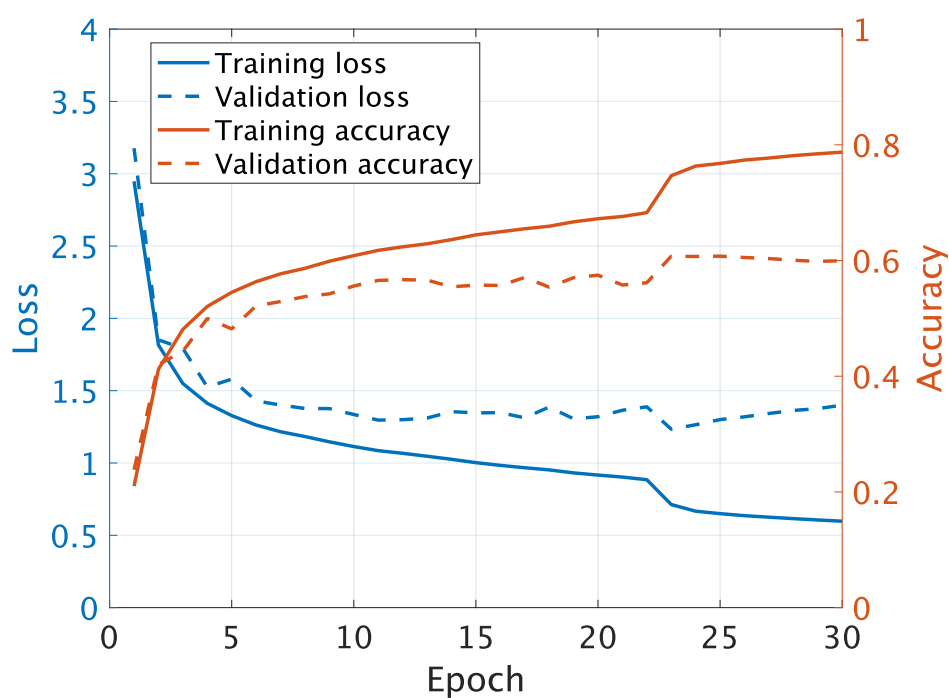


図 7.2: TCN の学習曲線. 200 分割で検出器は HLV .

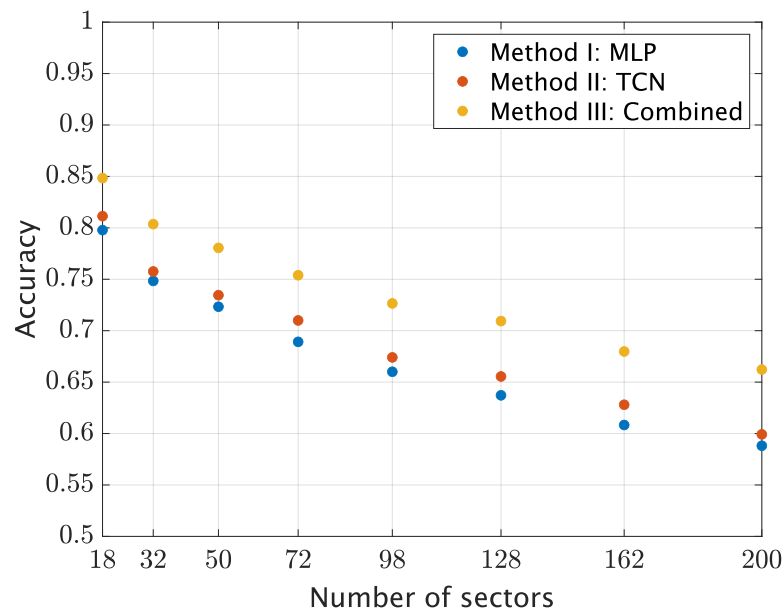


図 7.3: 分割方法 A で 3 台の検出器による各手法の正解率

7.1.2 分割方法 B: HEALPix

3 台の検出器で, HEALPix で分割した場合の分割数に対する正解率は図 7.4 のようになった. この分割方法でも分割方法 A と同様に手法 3, 2, 1 の順に正解率が高い. 分割方法 A と分割方法 B での比較を図 7.5 に示した. 分割方法 A の方がわずかに正解率が高いため, これ以降は分割方法 A を用いた.

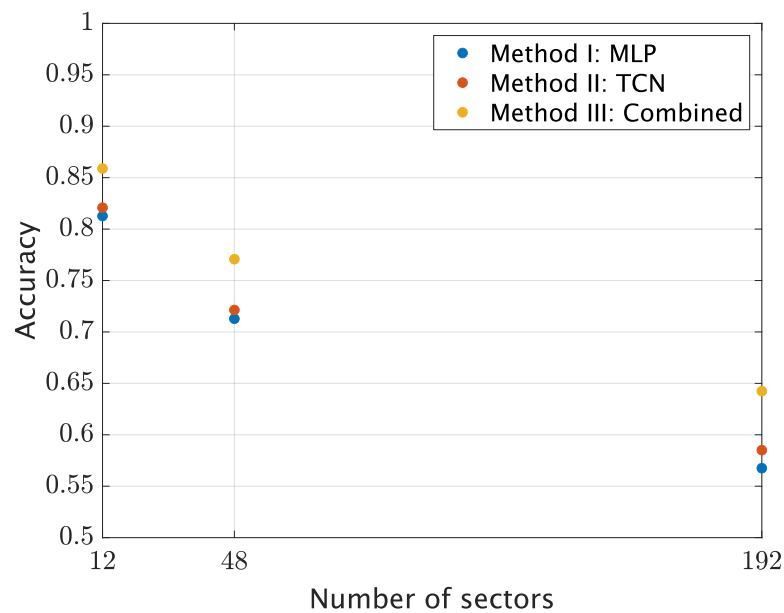


図 7.4: 分割方法 B で 3 台の検出器による各手法の正解率

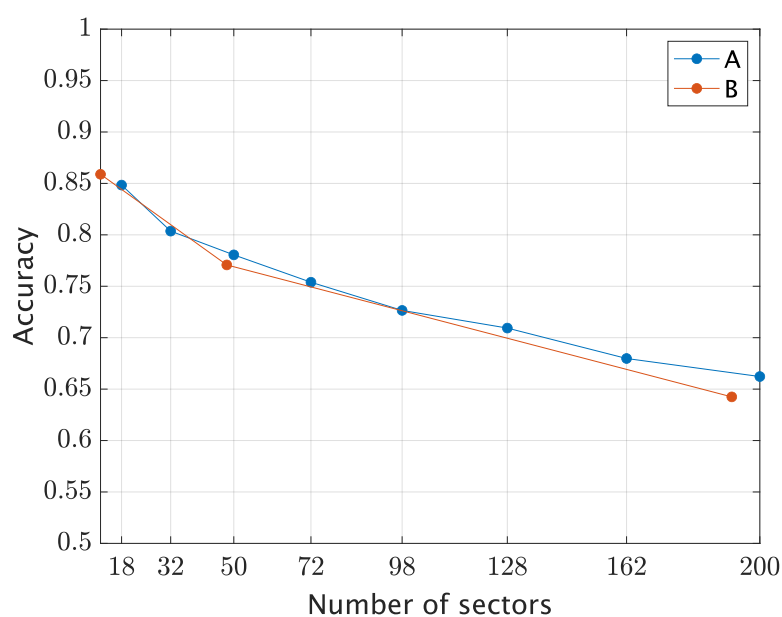


図 7.5: 各分割方法での 3 台の検出器による正解率

7.2 4 台の検出器による推定結果

分割方法 A で手法 3 を用いた場合の, LIGO H1, LIGO L1, Virgo による正解率と KAGRA を加えた場合の正解率は図 7.6 のようになった. KAGRA を加えると 6~10% ほど精度が改善されている. 特に分割数が増えるほどその改善率は大きくなっていることが読み取れる.

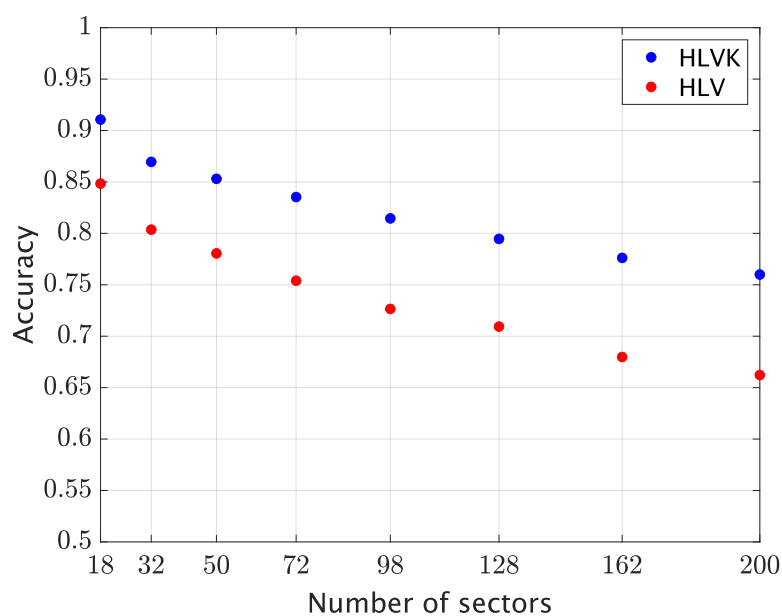


図 7.6: 分割方法 A, 推定手法 3 での 3 台の検出器の正解率と 4 台の検出器の正解率の比較

4 台の検出器で 32 分割したときの混同行列をヒートマップで表したものが図 7.7 である。ここではラベルごとのサンプル数が異なるため、行方向の和が 1 になるように規格化している。対角線上は正解したデータを表し、それ以外は間違えたデータを表す。対角線にあるデータの割合が最も大きい、その対角線の 1 つ隣の直線と 8 つ隣の直線にあるデータの割合も大きいことが分かる。前者は RA 方向の隣のセクタとの間違い、後者は Dec 方向の隣のセクタとの間違いを表している。したがって、ほとんどのデータは正しいセクタまたはその隣のセクタから到来してきたと予測されており、ニューラルネットワークが到来方向の特徴を正しく学習できていることが分かる。

次に SNR と正解率の関係を調べた。3 台の場合と 4 台の場合の 200 分割のときの各 SNR に対する正解率は図 7.8 のようになった。SNR が小さい、ノイズの大きなデータに対してはこの手法は良い正解率を得られないことが分かる。また Dec に対する正解率は図 7.9 のようになった。Dec の大きいセクタは立体角が小さいため、正解率は悪い。しかし、そのようなセクタに対して KAGRA による改善度が大きくなっている。

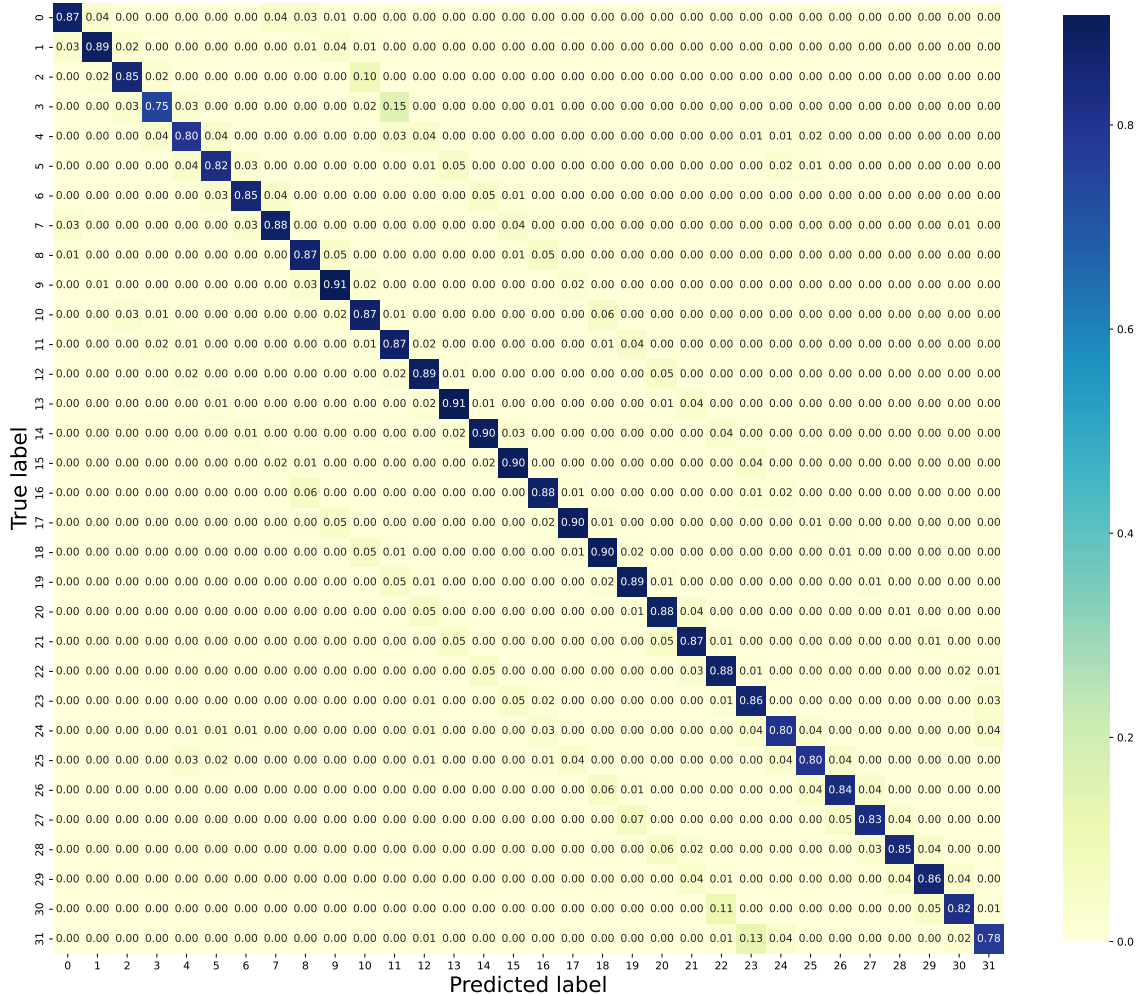


図 7.7: 32 分割で 4 台の検出器を用いたときの混同行列

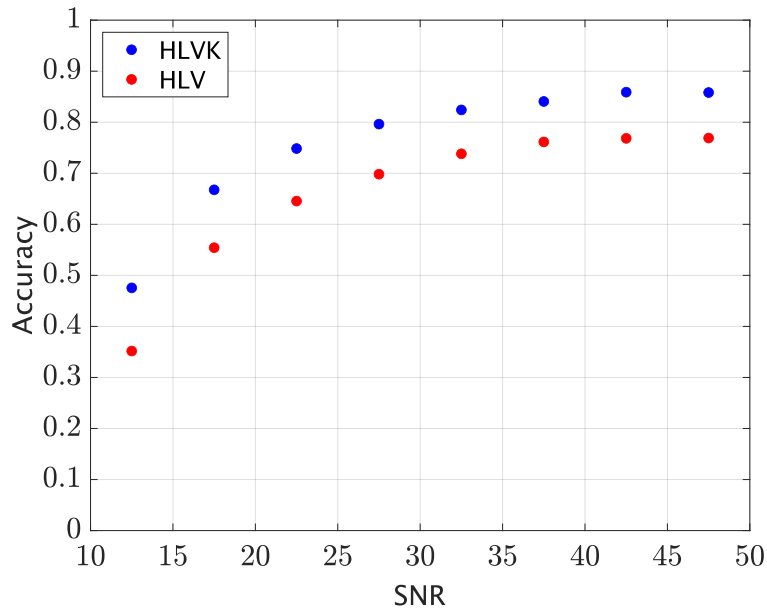


図 7.8: SNR に対する正解率

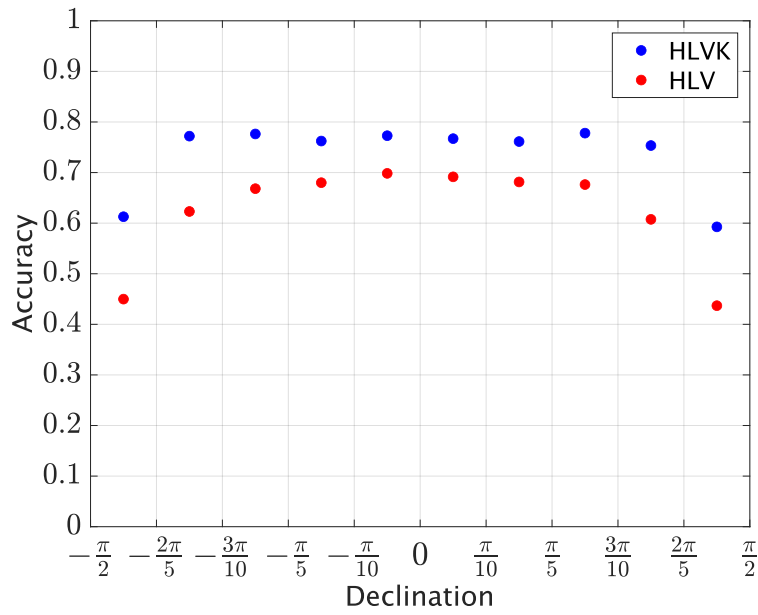


図 7.9: Dec に対する正解率

7.3 GW170814 データでの検証

GW170814 [36] は LIGO H1, LIGO L1, Virgo の 3 台の検出器で同時観測された初めての重力波である。本研究の手法の性能を検証するためにこの重力波を用いた。実データは GWOSC (Gravitational Wave Open Science Center) [81] からダウンロードすることができるが、本研究のニューラルネットワークの学習に用いた訓練データは各検出器の設計感度を用

いており，これは GW170814 が観測された当時の検出器の感度とは大きく異なる．したがって実データではなく，GW170814 のパラメータを用いてシミュレーションした重力波に，訓練データと同様に各検出器の設計感度から生成されたガウシアンノイズを加えた．用いたパラメータは表 7.1 に示した．この表に書かれていないパラメータについてはランダムに選んだ．また，検出器は LIGO H1, LIGO L1, Virgo の 3 台とした．

表 7.1: GW170814 のパラメータ

Mass1	$30.5M_{\odot}$
Mass2	$25.3M_{\odot}$
Right ascension	$03^{\text{h}}11^{\text{m}}$
Declination	$-44^{\circ}57^{\text{m}}$
Network SNR	18.3

ここでは角度分解能を上げるために 200 分割した後にさらに 4 分割するモデルを学習した．本来は 200 セクタのそれぞれに対して 4 分割するモデルを学習すべきであるが，本研究では 200 セクタのうち GW170814 データに対して予測されたセクタを 4 分割するモデルのみを学習した．

GW170814 データに対して，手法 3 で 200 分割のモデルを用いて予測した結果，42 番目のセクタと予測された．これは Dec が $-3\pi/10$ から $-\pi/5$ ，RA が $\pi/5$ から $3\pi/10$ であり，図 7.10 の赤色の部分を表す．

次に 42 番目のセクタを 4 分割するモデルを学習した．この範囲のデータを 24 万個用意し，16 万個で訓練し，4 万個はバリデーションデータとし，残りの 4 万個でテストした．テストデータによる推定の正解率を表 7.2 に示した．正解率が最も高い手法 3 を用いて GW170814 データに対して推定を行った．

GW170814 データを用いた結果，図 7.11 の赤色のセクタに分類された．また，図にデータ

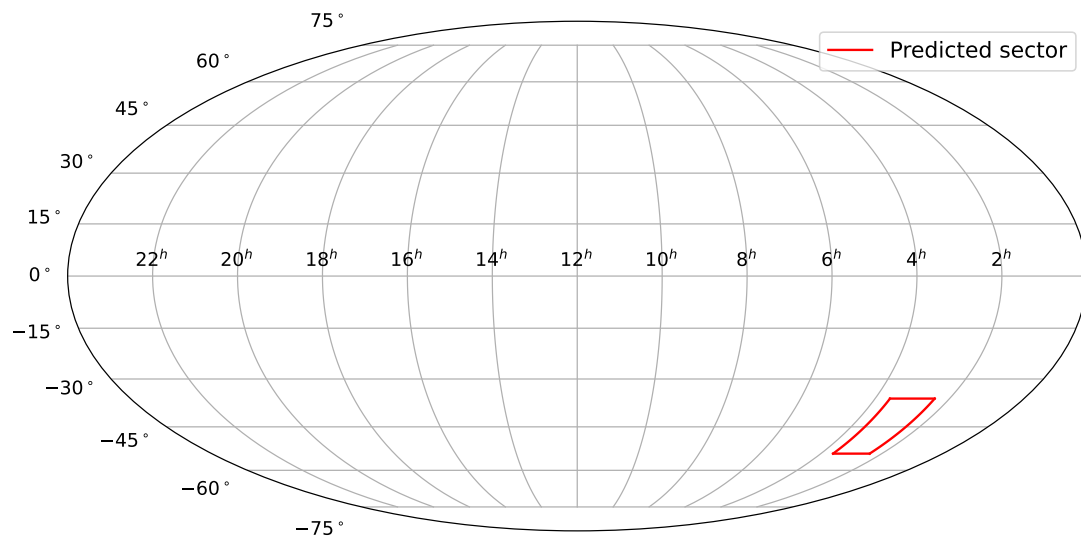


図 7.10: GW170814 データに対する 200 分割のモデルの予測結果

表 7.2: 200 分割後に 4 分割するモデルの正解率

手法 1	0.8244
手法 2	0.8761
手法 3	0.8812

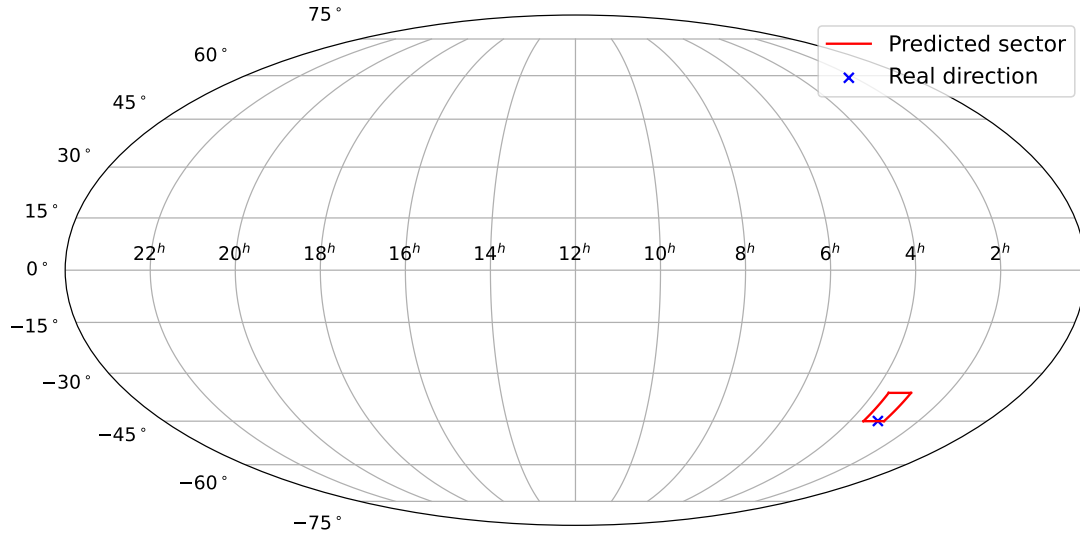


図 7.11: 200 分割後さらに 4 分割するモデルによる GW170814 データの推定結果

の生成に用いた正しい方向を示した。これはちょうどセクタの境界部のように見えるが、実際は赤で示したセクタ内に含まれており、正しいセクタに分類できたことになる。このセクタは Dec が $-\pi/4$ から $-\pi/5$ 、RA が $\pi/4$ から $3\pi/10$ であり、立体角は 52.6 deg^2 である。

本研究の手法を用いた結果、GW170814 を模倣したデータで推定時間 0.568 秒で 52.6 deg^2 のセクタに正しく分類することができた。実際に GW170814 データに対して行われた低遅延での到来方向推定の 90% 信頼区間は LIGO の 2 台では 1160 deg^2 であり、Virgo での観測により 100 deg^2 に絞られた [36]。実データの非定常なノイズの影響や検出器の感度の違いなどによってこれらと比較することはできないが、本研究の手法は十分速い時間で到来方向推定ができることを確かめられた。

第 8 章

結論

本研究では連星ブラックホール合体からの重力波の到来方向推定を行った。先行研究を含む 3 つの手法を試した結果、先行研究による手法と新たに用いた TCN を組み合わせることにより先行研究より精度が 5~8% 向上した。次にこの手法を用いて LIGO と Virgo の 3 台の検出器のデータを用いた場合と、KAGRA を含む 4 台の検出器のデータを用いた場合の精度の比較を行い、KAGRA が加わることによって推定精度は上がり、特に分割数が大きい場合や Declination が大きい方向での精度の向上が大きいことが分かった。最後に GW170814 のパラメータを用いてシミュレーションした波形でモデルの検証を行った。ここでは 200 分割後にさらに 4 分割するモデルで推定を行い、1 秒未満で 52.6 deg^2 のセクタに正しく分類することができた。

7.2 節で述べたように本研究の手法では SNR が小さい重力波に対してはあまり良い精度で推定ができない。このようなノイズの大きな重力波に対する精度を向上させるためにノイズ除去の機械学習モデルと到来方向推定を組み合わせる方法が考えられる。また、非定常なノイズが含まれる実データに対しての推定精度を上げるために、ニューラルネットワークの構造や訓練データをさらに工夫する必要がある。さらに、本研究では分類のみを行ったが、BAYESTAR による到来方向推定と精度や時間を比較するために 90% 信頼区間を構成することを今後の課題としたい。

謝辞

多くの方々のご支援があって本論文を完成させることができました。この場で感謝の気持ちを述べさせていただきます。

指導教員の宗宮健太郎先生は、本研究を進めるにあたってアドバイスや指摘をたくさん頂きました。また、投稿論文の執筆や学会発表などの機会を与えて頂き、研究生活1年目からたくさんの貴重な経験をすることができました。

東京都市大学総合研究所の高橋弘毅先生は、ミーティングで本研究についてのアドバイスをたくさん頂きました。また、データ解析の論文や教科書をたくさん紹介していただき、大変勉強になりました。

原田健一先生は研究テーマは異なりますが、干渉計について丁寧に教えて頂き、光学実験にも興味を持つことができました。

博士2年の小田部荘達さん、修士2年の阿部誉さん、栗林誠さん、立原浩輝さんにはゼミや発表練習の際に鋭い質問や指摘をして頂きました。また、先輩方のスライドや発表の仕方などをたくさん参考にさせて頂きました。修士1年の鈴木海堂さんと鈴木孝典さんは、輪講の内容や院試についての質問に優しく丁寧に答えて頂きました。同じく修士1年のHou Yilunさんは時間のかかるデータ生成を手伝って頂きました。また、Yilunさんとの議論によって自分の間違いをいくつか発見することができました。同期の竹口浩太郎君とは輪講と一緒に議論したり、授業の課題を協力し合ったりしたのがとても助かりました。

他にも物理学系の先生方や、一緒に自主ゼミをしてきた物理学系の友人のおかげで楽しく物理を学ぶことができました。ありがとうございました。最後に、大学で自由に学ぶ環境を与えてくれた両親に感謝申し上げます。

参考文献

- [1] A. Einstein, Näherungsweise Integration der Feldgleichungen der Gravitation, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin) , 688 (1916).
- [2] A. Einstein, Über Gravitationswellen, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin) , 154 (1918).
- [3] J. Aasi *et al.* (LIGO Scientific Collaboration), Advanced LIGO, [Classical and Quantum Gravity](#) **32**, 074001 (2015).
- [4] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Observation of Gravitational Waves from a Binary Black Hole Merger, [Phys. Rev. Lett.](#) **116**, 061102 (2016).
- [5] F. Acernese *et al.*, Advanced Virgo: a second-generation interferometric gravitational wave detector, [Classical and Quantum Gravity](#) **32**, 024001 (2014).
- [6] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral, [Phys. Rev. Lett.](#) **119**, 161101 (2017).
- [7] B. P. Abbott *et al.*, Multi-messenger Observations of a Binary Neutron Star Merger, [The Astrophysical Journal](#) **848**, L12 (2017).
- [8] R. Abbott *et al.* (LIGO Scientific Collaboration, Virgo Collaboration and KAGRA Collaboration), GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Part of the Third Observing Run, [arXiv:2111.03606 \[gr-qc\]](#) .
- [9] T. Akutsu *et al.* (The KAGRA Collaboration), KAGRA: 2.5 generation interferometric gravitational wave detector, [Nature Astronomy](#) **3**, 35–40 (2019).
- [10] H. Grote, The status of GEO 600, [Classical and Quantum Gravity](#) **25**, 114043 (2008).
- [11] A. Nitz *et al.*, [gwastro/pycbc: Release v2.0.1 of PyCBC](#).
- [12] D. George and E. A. Huerta, Deep neural networks to enable real-time multimessenger astrophysics, [Phys. Rev. D](#) **97**, 044039 (2018).
- [13] P. Astone, P. Cerdá-Durán, I. Di Palma, M. Drago, F. Muciaccia, C. Palomba, and F. Ricci, New method to observe gravitational waves emitted by core collapse supernovae, [Phys. Rev. D](#) **98**, 122002 (2018).
- [14] A. Iess, E. Cuoco, F. Morawski, and J. Powell, Core-Collapse supernova gravitational-

- wave search and deep learning classification, [Machine Learning: Science and Technology](#) **1**, 025014 (2020).
- [15] M. L. Chan, I. S. Heng, and C. Messenger, Detection and classification of supernova gravitational wave signals: A deep learning approach, [Phys. Rev. D](#) **102**, 043022 (2020).
- [16] M. López, I. Di Palma, M. Drago, P. Cerdá-Durán, and F. Ricci, Deep learning for core-collapse supernova detection, [Phys. Rev. D](#) **103**, 063011 (2021).
- [17] C. Dreissigacker, R. Sharma, C. Messenger, R. Zhao, and R. Prix, Deep-learning continuous gravitational waves, [Phys. Rev. D](#) **100**, 044009 (2019).
- [18] C. Dreissigacker and R. Prix, Deep-learning continuous gravitational waves: Multiple detectors and realistic noise, [Phys. Rev. D](#) **102**, 022005 (2020).
- [19] B. Beheshtipour and M. A. Papa, Deep learning for clustering of continuous gravitational wave candidates, [Phys. Rev. D](#) **101**, 064009 (2020).
- [20] B. Beheshtipour and M. A. Papa, Deep learning for clustering of continuous gravitational wave candidates. II. Identification of low-SNR candidates, [Phys. Rev. D](#) **103**, 064027 (2021).
- [21] T. S. Yamamoto and T. Tanaka, Use of an excess power method and a convolutional neural network in an all-sky search for continuous gravitational waves, [Phys. Rev. D](#) **103**, 084049 (2021).
- [22] B. Allen and J. D. Romano, Detecting a stochastic background of gravitational radiation: Signal processing strategies and sensitivities, [Phys. Rev. D](#) **59**, 102001 (1999).
- [23] A. J. K. Chua and M. Vallisneri, Learning Bayesian Posteriors with Neural Networks for Gravitational-Wave Inference, [Phys. Rev. Lett.](#) **124**, 041102 (2020).
- [24] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy, [Nature Physics](#) (2021).
- [25] W. Wei and E. Huerta, Gravitational wave denoising of binary black hole mergers with deep learning, [Physics Letters B](#) **800**, 135081 (2020).
- [26] C. Chatterjee, L. Wen, F. Diakogiannis, and K. Vinsen, Extraction of binary black hole gravitational wave signals from detector data using deep learning, [Phys. Rev. D](#) **104**, 064046 (2021).
- [27] M. Zevin *et al.*, Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science, [Classical and Quantum Gravity](#) **34**, 064003 (2017).
- [28] S. Bahaadini, V. Noroozi, N. Rohani, S. Coughlin, M. Zevin, J. Smith, V. Kalogera, and A. Katsaggelos, Machine learning for Gravity Spy: Glitch classification and dataset, [Information Sciences](#) **444**, 172 (2018).
- [29] S. Soni *et al.*, Discovering features in gravitational-wave data through detector characterization, citizen science and machine learning, [Classical and Quantum Gravity](#)

- [38](#), 195016 (2021).
- [30] S. Bahaadini, N. Rohani, S. Coughlin, M. Zevin, V. Kalogera, and A. K. Katsaggelos, Deep multi-view models for glitch classification, in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017) pp. 2931–2935.
 - [31] Y. Sakai, Y. Itoh, P. Jung, K. Kokeyama, C. Kozakai, K. T. Nakahira, S. Oshino, Y. Shikano, H. Takahashi, T. Uchiyama, G. Ueshima, T. Washimi, T. Yamamoto, and T. Yokozawa, Unsupervised Learning Architecture for Classifying the Transient Noise of Interferometric Gravitational-wave Detectors, [arXiv:2111.10053 \[gr-qc\]](#) .
 - [32] C. Chatterjee, L. Wen, K. Vinsen, M. Kovalam, and A. Datta, Using deep learning to localize gravitational wave sources, *Phys. Rev. D* **100**, 103025 (2019).
 - [33] L. P. Singer and L. R. Price, Rapid Bayesian position reconstruction for gravitational-wave transients, *Phys. Rev. D* **93**, 024013 (2016).
 - [34] Y. Liu, Detection and Localization of Gravitational Waves Using Convolutional Neural Networks, [Master's thesis](#), Tokyo Institute of Technology (2020).
 - [35] S. Bai, J. Z. Kolter, and V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, [arXiv:1803.01271 \[cs.LG\]](#) .
 - [36] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), GW170814: A Three-Detector Observation of Gravitational Waves from a Binary Black Hole Coalescence, *Phys. Rev. Lett.* **119**, 141101 (2017).
 - [37] M. Maggiore, *Gravitational waves: Volume 1: Theory and experiments*, Vol. 1 (Oxford university press, 2008).
 - [38] R. A. Isaacson, Gravitational Radiation in the Limit of High Frequency. I. The Linear Approximation and Geometrical Optics, *Phys. Rev.* **166**, 1263 (1968).
 - [39] R. A. Isaacson, Gravitational Radiation in the Limit of High Frequency. II. Nonlinear Terms and the Effective Stress Tensor, *Phys. Rev.* **166**, 1272 (1968).
 - [40] R. Abbott *et al.*, Observation of Gravitational Waves from Two Neutron Star–Black Hole Coalescences, *The Astrophysical Journal Letters* **915**, L5 (2021).
 - [41] W. G. Anderson, P. R. Brady, J. D. E. Creighton, and E. E. Flanagan, Excess power statistic for detection of burst sources of gravitational radiation, *Phys. Rev. D* **63**, 042003 (2001).
 - [42] S. Chatterji, A. Lazzarini, L. Stein, P. J. Sutton, A. Searle, and M. Tinto, Coherent network analysis technique for discriminating gravitational-wave bursts from instrumental noise, *Phys. Rev. D* **74**, 082005 (2006).
 - [43] *Latest estimated sensitivity of KAGRA (v201708)*, Tech. Rep. T1707038-v9 (JGW Document, 2017).
 - [44] D. Shoemaker, R. Schilling, L. Schnupp, W. Winkler, K. Maischberger, and A. Rüdiger, Noise behavior of the Garching 30-meter prototype gravitational-wave detector, *Phys. Rev. D* **38**, 423 (1988).
 - [45] J. D. Creighton and W. G. Anderson, *Gravitational-wave physics and astronomy: An*

- introduction to theory, experiment and data analysis* (John Wiley & Sons, 2012).
- [46] J. Neyman and E. S. Pearson, On the Problem of the Most Efficient Tests of Statistical Hypotheses, [Philosophical Transactions of the Royal Society of London Series A](#) **231**, 289 (1933).
 - [47] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of State Calculations by Fast Computing Machines, [The Journal of Chemical Physics](#) **21**, 1087 (1953).
 - [48] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, [Biometrika](#) **57**, 97 (1970).
 - [49] J. Skilling, Nested sampling for general Bayesian computation, [Bayesian Analysis](#) **1**, 833 (2006).
 - [50] J. Veitch *et al.*, Parameter estimation for compact binaries with ground-based gravitational-wave observations using the LALInference software library, [Phys. Rev. D](#) **91**, 042003 (2015).
 - [51] S. Fairhurst, Triangulation of gravitational wave sources with a network of detectors, [New Journal of Physics](#) **11**, 123006 (2009).
 - [52] P. Welch, The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms, [IEEE Transactions on Audio and Electroacoustics](#) **15**, 70 (1967).
 - [53] B. P. Abbott *et al.*, A guide to LIGO–Virgo detector noise and extraction of transient gravitational-wave signals, [Classical and Quantum Gravity](#) **37**, 055002 (2020).
 - [54] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen, The Modern Mathematics of Deep Learning, [arXiv:2105.04026 \[cs.LG\]](#) .
 - [55] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, USA, 2016) <http://www.deeplearningbook.org>.
 - [56] H. Robbins and S. Monro, A Stochastic Approximation Method, [The Annals of Mathematical Statistics](#) **22**, 400 (1951).
 - [57] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, On the importance of initialization and momentum in deep learning, in [Proceedings of the 30th International Conference on Machine Learning](#), Proceedings of Machine Learning Research, Vol. 28, edited by S. Dasgupta and D. McAllester (PMLR, Atlanta, Georgia, USA, 2013) pp. 1139–1147.
 - [58] J. Duchi, E. Hazan, and Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, [Journal of Machine Learning Research](#) **12**, 2121 (2011).
 - [59] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, [arXiv:1412.6980 \[cs.LG\]](#) .
 - [60] S. Ruder, An overview of gradient descent optimization algorithms, [arXiv:1609.04747 \[cs.LG\]](#) .
 - [61] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in [Proceedings of the 32nd International Con-](#)

- ference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 37, edited by F. Bach and D. Blei (PMLR, Lille, France, 2015) pp. 448–456.
- [62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research* **15**, 1929 (2014).
 - [63] S. Hochreiter and J. Schmidhuber, Long Short-Term Memory, *Neural Computation* **9**, 1735 (1997).
 - [64] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, [arXiv:1412.3555 \[cs.NE\]](#) .
 - [65] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society, Los Alamitos, CA, USA, 2016) pp. 770–778.
 - [66] A. Nitz *et al.*, [gwastro/pycbc: Release v2.0.1 of PyCBC](#).
 - [67] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Convolutional neural networks: A magic bullet for gravitational-wave detection?, *Phys. Rev. D* **100**, 063015 (2019).
 - [68] T. Gebhard and N. Kilbertus, [timothygebhard/ggwd: Version 1.0](#).
 - [69] A. Buonanno and T. Damour, Effective one-body approach to general relativistic two-body dynamics, *Phys. Rev. D* **59**, 084006 (1999).
 - [70] A. Bohé *et al.*, Improved effective-one-body model of spinning, nonprecessing binary black holes for the era of gravitational-wave astrophysics with advanced detectors, *Phys. Rev. D* **95**, 044028 (2017).
 - [71] *Updated Advanced LIGO sensitivity design curve*, Tech. Rep. T1800044-v5 (LIGO Document, 2018).
 - [72] F. Acernese *et al.*, Advanced Virgo: a second-generation interferometric gravitational wave detector, *Classical and Quantum Gravity* **32**, 024001 (2014).
 - [73] A. Manzotti and A. Dietz, Prospects for early localization of gravitational-wave signals from compact binary coalescences with advanced detectors, [arXiv:1202.4031 \[gr-qc\]](#) .
 - [74] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere, *The Astrophysical Journal* **622**, 759 (2005).
 - [75] A. Zonca, L. Singer, D. Lenz, M. Reinecke, C. Rosset, E. Hivon, and K. Gorski, healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in Python, *Journal of Open Source Software* **4**, 1298 (2019).
 - [76] P. Virtanen *et al.* (SciPy 1.0 Contributors), SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, *Nature Methods* **17**, 261 (2020).
 - [77] A. Paszke *et al.*, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wal-

- lach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019) pp. 8024–8035.
- [78] K. He, X. Zhang, S. Ren, and J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, in *2015 IEEE International Conference on Computer Vision (ICCV)* (2015) pp. 1026–1034.
- [79] L. Breiman, Bagging Predictors, *Machine Learning* **24**, 123 (1996).
- [80] D. H. Wolpert, Stacked generalization, *Neural Networks* **5**, 241 (1992).
- [81] R. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Open data from the first and second observing runs of Advanced LIGO and Advanced Virgo, *SoftwareX* **13**, 100658 (2021).

付録 A

開発環境

本研究の開発環境を以下にまとめる.

ハードウェア

- CPU: AMD Ryzen 7 3700X
- GPU: NVIDIA GeForce RTX 3070
- Memory: 16GB

ソフトウェア

- OS: Ubuntu 20.04.3 LTS
- Python: 3.8.10
- CUDA: 11.1
- cuDNN: 8.0.5
- SciPy: 1.7.3
- NumPy: 1.19.5
- pandas: 1.3.4
- torch: 1.9.0+cu111
- torchaudio: 0.9.0
- torchvision: 0.10.0+cu111
- PyCBC: 1.18.3
- healpy: 1.15.2

付録 B

ソースコード

本研究で使⽤したソースコードを以下に示す。ソースコード [B.1](#): train.py は学習，ソースコード [B.2](#): predict.py は推論に⽤いるコードである。ソースコード [B.3](#): preprocess.py には train.py と predict.py 内で⽤いられている関数が含まれている。例えば 4 台の検出器で⼿法 1 の学習を⾏う際は

```
python train.py \  
--hdf_file a.hdf b.hdf \  
--method 1 \  
--use_KAGRA \  
--training_size 200000 \  
--batch_size 4000 \  
--epoch 200 \  
--seed 42 \  

```

を実⾏する。

ソースコード B.1: train.py

```
1  import argparse  
2  import os  
3  import random  
4  
5  import h5py  
6  import numpy as np  
7  import pandas as pd  
8  import scipy.signal  
9  import scipy.stats  
10 import torch  
11 import torch.nn as nn  
12 import torch.nn.functional as F  
13 import torch.optim as optim  
14 import torchaudio  
15 import torchvision  
16 from torch.nn import BatchNorm1d  
17 from torch.utils.data import DataLoader  
18
```

```

19 from preprocess import (create_array_from_hdf, create_dataframe,
20                          create_labels_from_hdf, create_train_dataloader,
21                          drop_KAGRA, fix_seed)
22
23
24 def get_arguments():
25
26     parser = argparse.ArgumentParser()
27
28     parser.add_argument("--hdf_file", nargs='*', required=True)
29     parser.add_argument("--csv_file")
30     parser.add_argument("--save_csv_path")
31     parser.add_argument("--HEALPix", action='store_true', help='default: False')
32     parser.add_argument("--method", type=int, default=1, help="MLP: 1, TCN:2")
33     parser.add_argument("--use_KAGRA", action='store_true', help='default: False')
34     parser.add_argument("--batch_size", type=int, default=512)
35     parser.add_argument("--seed", type=int, default=42)
36     parser.add_argument("--epoch", type=int, default=200)
37     parser.add_argument("--learning_rate", type=float, default=0.001)
38     parser.add_argument("--dropout", type=float, default=0.0)
39     parser.add_argument("--training_size", type=int, default=200000)
40     parser.add_argument("--weights_directory", default='weights')
41
42     args = parser.parse_args()
43     return args
44
45 def train_model(net, dataloaders_dict, criterion, \
46                optimizer, scheduler, num_epochs, n_sectors, weights_path):
47     train_loss, train_acc, val_loss, val_acc = [], [], [], []
48     max_acc = 0
49
50     for epoch in range(num_epochs):
51         print('Epoch {}/{}'.format(epoch+1, num_epochs), end='\t')
52         for phase in ['train', 'val']:
53             if phase == 'train':
54                 net.train()
55             else:
56                 net.eval()
57
58             epoch_loss, epoch_corrects = 0.0, 0.0
59             for inputs, labels in dataloaders_dict[phase]:
60                 inputs = inputs.to(device)
61                 labels = labels.to(device)
62
63                 optimizer.zero_grad()
64                 with torch.set_grad_enabled(phase == 'train'):
65                     outputs = net(inputs)
66                     loss = criterion(outputs, labels)
67                     _, preds = torch.max(outputs, 1)
68
69                 if phase == 'train':
70                     loss.backward()
71                     optimizer.step()
72
73             epoch_loss += loss.item() * inputs.size(0)
74             epoch_corrects += torch.sum(preds == labels.data)
75

```

```

76         epoch_loss = epoch_loss/len(dataloaders_dict[phase].dataset)
77         epoch_acc = \
78         (epoch_corrects.double()/len(dataloaders_dict[phase].dataset)).item()
79
80         if phase=='train':
81             train_loss.append(round(epoch_loss,5))
82             train_acc.append(round(epoch_acc,5))
83             print('train loss: {:.4f} acc: {:.4f}'.format(epoch_loss, epoch_acc),
84                   end='\t')
85         else:
86             val_loss.append(round(epoch_loss,5))
87             val_acc.append(round(epoch_acc,5))
88             print('val loss: {:.4f} acc: {:.4f}'.format(epoch_loss, epoch_acc))
89
90             if epoch_acc > max_acc:
91                 max_acc = epoch_acc
92                 torch.save(net.state_dict(), weights_path)
93                 print('Saved the weights.')
94
95         scheduler.step(epoch_loss)
96         history = {'train_loss': train_loss, 'train_acc': train_acc,
97                  'val_loss': val_loss, 'val_acc': val_acc}
98         return history
99
100 class TemporalBlock(nn.Module):
101     def __init__(self, n_inputs, n_outputs, kernel_size, \
102                 stride, dilation, padding, dropout):
103         super(TemporalBlock, self).__init__()
104
105         self.conv1 = nn.Conv1d(
106             n_inputs, n_outputs, kernel_size, stride, int(padding/2), dilation
107         )
108         self.bn1 = nn.BatchNorm1d(64, eps=1e-03, momentum=0.01)
109         self.prelu1 = nn.PReLU()
110         self.dropout1 = nn.Dropout(dropout)
111
112         self.conv2 = nn.Conv1d(
113             n_outputs, n_outputs, kernel_size, stride, int(padding/2), dilation
114         )
115         self.bn2 = nn.BatchNorm1d(64, eps=1e-03, momentum=0.01)
116         self.prelu2 = nn.PReLU()
117         self.dropout2 = nn.Dropout(dropout)
118
119         self.net = nn.Sequential(
120             self.conv1, self.bn1, self.prelu1, self.dropout1,
121             self.conv2, self.bn2, self.prelu2, self.dropout2
122         )
123
124         if n_inputs != n_outputs:
125             self.downsample = nn.Conv1d(n_inputs, n_outputs, 1)
126         else:
127             self.downsample = None
128         self.prelu = nn.PReLU()
129         self.init_weights()
130
131     def init_weights(self):
132         torch.nn.init.kaiming_normal_(self.conv1.weight)

```

```

133         torch.nn.init.kaiming_normal_(self.conv2.weight)
134         if self.downsample is not None:
135             torch.nn.init.kaiming_normal_(self.downsample.weight)
136
137     def forward(self, x):
138         out = self.net(x)
139         res = x if self.downsample is None else self.downsample(x)
140         return self.prelu(out + res)
141
142
143 class TemporalConvNet(nn.Module):
144     def __init__(self, num_inputs, num_channels, kernel_size, dropout):
145         super(TemporalConvNet, self).__init__()
146         layers = []
147         num_levels = len(num_channels)
148         for i in range(num_levels):
149             dilation_size = 2 ** i
150             in_channels = num_inputs if i == 0 else num_channels[i-1]
151             out_channels = num_channels[i]
152             layers += \
153                 [TemporalBlock(in_channels, out_channels, kernel_size, \
154                             stride=1, dilation=dilation_size, \
155                             padding=(kernel_size-1)*dilation_size, dropout=dropout)
156                ]
157
158             self.network = nn.Sequential(*layers)
159
160     def forward(self, x):
161         return self.network(x)
162
163 class TCN(nn.Module):
164     def __init__(self, input_size, output_size, \
165                 num_channels, kernel_size, dropout):
166         super(TCN, self).__init__()
167         self.tcn = TemporalConvNet(input_size, num_channels, kernel_size, dropout)
168         self.linear = nn.Linear(num_channels[-1], output_size)
169
170     def forward(self, x):
171         x = self.tcn(x)
172         x = self.linear(x[:, :, -1])
173         return x
174
175 class MLP(nn.Module):
176     def __init__(self, n_sectors, use_KAGRA):
177         super(MLP, self).__init__()
178         self.fc1 = nn.Linear(42, 512) if use_KAGRA else nn.Linear(21, 512)
179         self.fc2 = nn.Linear(512, 256)
180         self.fc3 = nn.Linear(256, 256)
181         self.fc4 = nn.Linear(256, n_sectors)
182         self.dropout1 = nn.Dropout(p=0.2)
183         self.prelu1 = nn.PReLU()
184         self.prelu2 = nn.PReLU()
185         self.prelu3 = nn.PReLU()
186
187         self.net = nn.Sequential(
188             self.fc1, self.prelu1, self.dropout1,
189             self.fc2, self.prelu2, self.dropout1,

```



```

190         self.fc3, self.prelu3, self.dropout1,
191         self.fc4
192     )
193
194     def forward(self, x):
195         x = self.net(x)
196         return x
197
198 if __name__ == '__main__':
199     device = 'cuda' if torch.cuda.is_available() else 'cpu'
200     print('Using', device)
201
202     args = get_arguments()
203
204     fix_seed(args.seed)
205     input_size=4 if args.use_KAGRA else 3
206     detector = 'HLVK' if args.use_KAGRA else 'HLV'
207     model = 'MLP' if args.method==1 else 'TCN'
208
209     print('Start preprocessing')
210
211     if args.method==1:
212         X = np.array([])
213         if args.csv_file:
214             df = pd.read_csv(args.csv_file)
215         else:
216             create_dataframe(args.hdf_file)
217
218         if args.save_csv_path:
219             os.makedirs(os.path.dirname(args.save_csv_path), exist_ok=True)
220             df.to_csv(args.save_csv_path, index=None)
221         if not args.use_KAGRA: df = drop_KAGRA(df)
222
223     elif args.method==2:
224         X = create_array_from_hdf(args.hdf_file, args.use_KAGRA).transpose(0,2,1)
225         df = pd.DataFrame([])
226     else:
227         raise RuntimeError('Choose method 1 or 2.')
228
229     if not args.HEALPix:
230         n_sectors_list = [2*i*i for i in range(3,11)]
231     else:
232         n_sectors_list = [12*(4**i) for i in range(3)]
233
234     print('Start training')
235     for n_sectors in n_sectors_list:
236         print('#'*30)
237         print('Number of sectors:', n_sectors)
238         print('#'*30)
239         os.makedirs(args.weights_directory, exist_ok=True)
240         weights_path = f'{args.weights_directory}/{model}_{detector}_{n_sectors}.pth'
241         labels = create_labels_from_hdf(args.hdf_file, n_sectors)
242         if args.method==1:
243             net = MLP(n_sectors, args.use_KAGRA)
244         else:
245             net = TCN(
246                 input_size=input_size, output_size=n_sectors, \

```

```

247         num_channels=[64]*7, kernel_size=3, dropout=args.dropout
248     )
249
250     dataloaders_dict = create_train_dataloader(
251         model=model, df=df, arr=X, labels=labels, n_sectors=n_sectors, \
252         batch_size=args.batch_size, training_size = args.training_size
253     )
254     net.to(device)
255     criterion = nn.CrossEntropyLoss()
256     optimizer = optim.Adam(net.parameters(), lr=args.learning_rate)
257     scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min')
258
259     history = train_model(net, dataloaders_dict, criterion, optimizer, \
260                           scheduler, args.epoch, n_sectors, weights_path
261                           )

```

ソースコード B.2: predict.py

```

1  import argparse
2  import copy
3  import os
4  import random
5
6  import h5py
7  import numpy as np
8  import pandas as pd
9  import scipy.signal
10 import scipy.stats
11 import torch
12 import torch.nn as nn
13 import torch.nn.functional as F
14 import torch.optim as optim
15 import torchaudio
16 import torchvision
17 from scipy.stats import rankdata
18 from torch.nn import BatchNorm1d
19 from torch.utils.data import DataLoader
20
21 from preprocess import (create_array_from_hdf, create_dataframe,
22                         create_labels_from_hdf, create_test_dataloader,
23                         drop_KAGRA, fix_seed)
24 from train import MLP, TCN, TemporalBlock, TemporalConvNet
25
26
27 def get_arguments():
28
29     parser = argparse.ArgumentParser()
30
31     parser.add_argument("--hdf_file", nargs='*', required=True)
32     parser.add_argument("--csv_file")
33     parser.add_argument("--save_csv_path")
34     parser.add_argument("--HEALPix", action='store_true', help='default: False')
35     parser.add_argument("--use_KAGRA", action='store_true', help='default: False')
36     parser.add_argument("--seed", type=int, default=42)
37     parser.add_argument("--method", type=int, default=1, help="MLP:1, TCN:2, Combined:3")

```

```

38     parser.add_argument("--weights_directory", default='weights')
39     args = parser.parse_args()
40
41     return args
42
43 def predict_TCN(test_dataloader, labels, n_sectors, \
44                 input_size, use_KAGRA, weights_path, device, method):
45     net = TCN(
46         input_size=input_size, output_size=n_sectors,
47         num_channels=[64]*7, kernel_size=3, dropout=0
48     )
49
50     net.to(device)
51     net.load_state_dict(torch.load(weights_path, map_location=device))
52     net.eval()
53     y_pred = np.zeros((0,n_sectors))
54     corrects = 0
55     for inputs, labels in test_dataloader:
56         inputs = inputs.to(device)
57         labels = labels.to(device)
58         outputs = net(inputs)
59         _, preds = torch.max(outputs, 1)
60
61         if method != 3:
62             corrects += torch.sum(preds == labels.data)
63
64     y_pred = np.concatenate([y_pred, np.array(outputs.to('cpu').detach())], axis=0)
65
66     if method==3:
67         return y_pred
68     else:
69         accuracy = round(((corrects.double() / len(test_dataloader.dataset)).item(), 5)
70         return y_pred, accuracy
71
72 def predict_MLP(X_test, labels, n_sectors, use_KAGRA, weights_path, method):
73     net = MLP(n_sectors, use_KAGRA)
74     net.load_state_dict(torch.load(weights_path, map_location='cpu'))
75     net.eval()
76
77     y_pred = net(torch.from_numpy(X_test).float())
78     _, y_pred_max = torch.max(y_pred, 1)
79     y_pred_max = np.array(y_pred_max)
80
81     if method == 3:
82         return y_pred.detach().numpy()
83     else:
84         accuracy = sum(labels == y_pred_max) / len(X_test)
85
86         return y_pred.detach().numpy(), accuracy
87
88 if __name__ == '__main__':
89     device = 'cuda' if torch.cuda.is_available() else 'cpu'
90
91     args = get_arguments()
92     fix_seed(args.seed)
93     input_size=4 if args.use_KAGRA else 3
94     detector = 'HLVK' if args.use_KAGRA else 'HLV'

```

```

95     model = 'MLP' if args.method==1 else 'TCN'
96
97     if args.method==1 or args.method==3:
98         if args.csv_file:
99             df = pd.read_csv(args.csv_file)
100         else:
101             create_dataframe(args.hdf_file)
102
103         if args.save_csv_path:
104             os.makedirs(os.path.dirname(args.save_csv_path), exist_ok=True)
105             df.to_csv(args.save_csv_path, index=None)
106         if not args.use_KAGRA:
107             df = drop_KAGRA(df)
108             df_ = df.drop(['dec', 'ra', 'snr'], axis=1).values
109     if args.method==2 or args.method==3:
110         X = create_array_from_hdf(args.hdf_file, args.use_KAGRA).transpose(0,2,1)
111
112
113     accs = []
114     print('Start prediction')
115
116     if not args.HEALPix:
117         n_sectors_list = [2*i*i for i in range(3,11)]
118     else:
119         n_sectors_list = [12*(4**i) for i in range(3)]
120
121     for n_sectors in n_sectors_list:
122         weights_path = f'{args.weights_directory}/{model}_{detector}_{n_sectors}.pth'
123         labels = create_labels_from_hdf(args.hdf_file, n_sectors, args.HEALPix)
124
125         if args.method==1:
126             y_pred, accuracy = predict_MLP(
127                 df_, labels, n_sectors, args.use_KAGRA, weights_path, args.method
128             )
129         elif args.method==2:
130             test_dataloader = create_test_dataloader(X, labels, n_sectors, 200)
131             y_pred, accuracy = predict_TCN(
132                 test_dataloader, labels, n_sectors, input_size, \
133                 args.use_KAGRA, weights_path, device, args.method
134             )
135         elif args.method==3:
136             weights_path = f'{args.weights_directory}/MLP_{detector}_{n_sectors}.pth'
137             y_pred1 = predict_MLP(df_, labels, n_sectors, args.use_KAGRA, \
138                                 weights_path, args.method
139                                 )
140
141             weights_path = f'{args.weights_directory}/TCN_{detector}_{n_sectors}.pth'
142             test_dataloader = create_test_dataloader(X, labels, n_sectors, 200)
143             y_pred2 = predict_TCN(
144                 test_dataloader, labels, n_sectors, input_size, args.use_KAGRA, \
145                 weights_path, device, args.method
146             )
147
148         r = 0.6
149         y_pred = r*y_pred1 + (1-r)*y_pred2
150         y_pred_max = np.argmax(y_pred, axis=1)
151

```

```

152         accuracy = sum(labels == y_pred_max) / len(labels)
153
154     else:
155         raise RuntimeError('Choose method 1, 2, or 3.')
156
157     accs.append(accuracy)
158
159     print('n_sectors:', n_sectors, 'accuracy:', accuracy)
160
161     if len(labels)==1 and args.method==3:
162         print('true sector =', labels[0], \
163             'predicted sector =', y_pred_max, y_pred[0][y_pred_max])
164     print(accs)

```

ソースコード B.3: preprocess.py

```

1  import random
2
3  import h5py
4  import healpy as hp
5  import numpy as np
6  import pandas as pd
7  import scipy.signal
8  import scipy.stats
9  import torch
10 import torch.nn as nn
11 import torch.nn.functional as F
12 import torch.optim as optim
13 import torchaudio
14 import torchvision
15 from torch.utils.data import DataLoader
16
17
18 def fix_seed(seed=42):
19     random.seed(seed)
20     np.random.seed(seed)
21     torch.manual_seed(seed)
22     torch.cuda.manual_seed(seed)
23
24 def create_dataframe(hdf_paths):
25     M, dec, ra, snr = np.zeros((0, 42)), [], [], []
26     for n, hdf_path in enumerate(hdf_paths):
27         with h5py.File(hdf_path, 'r') as f:
28             h1_strain = f['injection_samples']['h1_strain'][()]
29             l1_strain = f['injection_samples']['l1_strain'][()]
30             v1_strain = f['injection_samples']['v1_strain'][()]
31             k1_strain = f['injection_samples']['k1_strain'][()]
32             dec_i = f['injection_parameters']['dec'][()]
33             ra_i = f['injection_parameters']['ra'][()]
34             snr_i = f['injection_parameters']['injection_snr'][()]
35
36         dec += list(dec_i)
37         ra += list(ra_i)
38         snr += list(snr_i)
39

```

```

40     l = len(h1_strain)
41     strains = {0:h1_strain, 1:l1_strain, 2:v1_strain, 3:k1_strain}
42
43     # Take mean 0
44     for i in range(l):
45         for j in range(4):
46             strains[j][i] -= np.mean(strains[j][i])
47
48     hilbert_h1 = scipy.signal.hilbert(h1_strain)
49     hilbert_l1 = scipy.signal.hilbert(l1_strain)
50     hilbert_v1 = scipy.signal.hilbert(v1_strain)
51     hilbert_k1 = scipy.signal.hilbert(k1_strain)
52     hilberts = {0:hilbert_h1, 1:hilbert_l1, 2:hilbert_v1, 3:hilbert_k1}
53
54     # Initializing input features matrix
55     features = np.zeros((l,42))
56
57     for i in range(l):
58         features_i = []
59
60         for d1 in range(4):
61             # for (v)(vi)
62             coal_time1 = np.argmax(np.abs(strains[d1][i]))
63             amp1 = strains[d1][i][coal_time1]
64             phase1 = hilberts[d1][i][coal_time1].imag
65
66             for d2 in range(d1+1,4):
67                 # (i) Arrival time delays of signals
68                 # (ii) Maximum cross-correlation values of signals
69                 corr = np.correlate(strains[d1][i], strains[d2][i], 'full')
70                 corr_argmax = np.argmax(abs(corr))
71
72                 features_i.append(corr_argmax-len(strains[d1][i])+1)
73                 features_i.append(corr[corr_argmax])
74
75                 # (iii) Arrival time delays of analytic signal
76                 # (iv) Maximum cross-correlation values of analytic signals
77                 corr_h = np.correlate(hilberts[d1][i], hilberts[d2][i], 'full')
78                 corr_h = np.abs(corr_h)
79                 corr_h_argmax = np.argmax(corr_h)
80
81                 features_i.append(corr_h_argmax()-len(strains[d1][i])+1)
82                 features_i.append(corr_h[corr_h_argmax])
83
84                 # (v) Ratios of average instantaneous amplitudes around merger
85                 # (vi) Phase lags around merger
86                 coal_time2 = np.argmax(np.abs(strains[d2][i]))
87                 amp2 = strains[d2][i][coal_time2]
88                 phase2 = hilberts[d2][i][coal_time2].imag
89
90                 features_i.append(amp1/amp2)
91                 features_i.append(phase1-phase2)
92
93                 # (vii) Complex Correlation Coefficients
94                 ccc = scipy.stats.pearsonr(strains[d1][i], strains[d2][i])[0]
95                 features_i.append(ccc)
96

```

```

97         features[i] = features_i
98
99     dec, ra, snr = np.array(dec), np.array(ra), np.array(snr)
100
101     # Standarization
102     if l>1:
103         for i in range(features.shape[1]):
104             features[:,i] -= np.mean(features[:,i])
105             features[:,i] /= np.std(features[:,i])
106
107     columns = ["Delay HL", "Max-Corr HL", "Delay-Ana HL", "Max-Corr-Ana HL",
108               "Amp-Ratio HL", "Phase-lag HL", "Corr-Coeff HL",
109               "Delay HV", "Max-Corr HV", "Delay-Ana HV", "Max-Corr-Ana HV",
110               "Amp-Ratio HV", "Phase-lag HV", "Corr-Coeff HV",
111               "Delay HK", "Max-Corr HK", "Delay-Ana HK", "Max-Corr-Ana HK",
112               "Amp-Ratio HK", "Phase-lag HK", "Corr-Coeff HK",
113               "Delay LV", "Max-Corr LV", "Delay-Ana LV", "Max-Corr-Ana LV",
114               "Amp-Ratio LV", "Phase-lag LV", "Corr-Coeff LV",
115               "Delay LK", "Max-Corr LK", "Delay-Ana LK", "Max-Corr-Ana LK",
116               "Amp-Ratio LK", "Phase-lag LK", "Corr-Coeff LK",
117               "Delay VK", "Max-Corr VK", "Delay-Ana VK", "Max-Corr-Ana VK",
118               "Amp-Ratio VK", "Phase-lag VK", "Corr-Coeff VK"
119           ]
120     df = pd.DataFrame(features, columns=columns)
121     df['dec'] = dec
122     df['ra'] = ra
123     df['snr'] = snr
124
125     return df
126
127 def drop_KAGRA(df):
128     df = df.drop(["Delay HK", "Delay LK", "Delay VK",
129                  "Max-Corr HK", "Max-Corr LK", "Max-Corr VK",
130                  "Delay-Ana HK", "Delay-Ana LK", "Delay-Ana VK",
131                  "Max-Corr-Ana HK", "Max-Corr-Ana LK", "Max-Corr-Ana VK",
132                  "Amp-Ratio HK", "Amp-Ratio LK", "Amp-Ratio VK",
133                  "Phase-lag HK", "Phase-lag LK", "Phase-lag VK",
134                  "Corr-Coeff HK", "Corr-Coeff LK", "Corr-Coeff VK"
135                  ], axis=1)
136     return df
137
138 def create_train_dataloader(model, df, arr, labels, n_sectors, training_size, batch_size):
139     if (model=='MLP' and training_size >= len(df)) \
140         or (model=='TCN' and training_size >= len(arr)):
141         raise RuntimeError('Training size must be smaller than the number of samples.')
142
143     if model=='MLP':
144         df_ = df.drop(['dec', 'ra', 'snr'], axis=1)
145         X_train, X_val = df_[:training_size].values, df_[training_size:len(labels)].values
146     else:
147         X_train, X_val = arr[:training_size], arr[training_size:len(labels)]
148
149     y_train, y_val = labels[:training_size], labels[training_size:len(labels)]
150
151     X_train, X_val = torch.from_numpy(X_train).float(), torch.from_numpy(X_val).float()
152     y_train, y_val = torch.from_numpy(y_train).long(), torch.from_numpy(y_val).long()
153

```

```

154     train_dataset = torch.utils.data.TensorDataset(X_train, y_train)
155     val_dataset = torch.utils.data.TensorDataset(X_val, y_val)
156
157     train_dataloader = torch.utils.data.DataLoader(
158         train_dataset, batch_size, shuffle=True, num_workers=2
159     )
160     val_dataloader = torch.utils.data.DataLoader(
161         val_dataset, batch_size, shuffle=False, num_workers=2
162     )
163     dataloaders_dict = {"train": train_dataloader, "val": val_dataloader}
164
165     return dataloaders_dict
166
167 def create_labels_from_hdf(hdf_paths, n_sectors, HEALPix=False):
168     dec, ra, snr = [], [], []
169     for i, hdf_path in enumerate(hdf_paths):
170         with h5py.File(hdf_path, 'r') as f:
171             dec_i = f['injection_parameters']['dec'][:]
172             ra_i = f['injection_parameters']['ra'][:]
173             snr_i = f['injection_parameters']['injection_snr'][:]
174
175             dec += list(dec_i)
176             ra += list(ra_i)
177             snr += list(snr_i)
178     dec, ra, snr = np.array(dec), np.array(ra), np.array(snr)
179
180     if not HEALPix:
181         delta = np.pi/int((n_sectors/2)**(0.5))
182         dec_label = ((dec+np.pi/2)/delta).astype(int)
183         ra_label = ((ra/delta)).astype(int)
184         labels = 2*int((n_sectors/2)**(0.5))*dec_label+ra_label
185     else:
186         Nside = int((n_sectors/12)**0.5)
187         labels = []
188         for i in range(len(dec)):
189             labels.append(hp.pixelfunc.ang2pix(Nside, dec[i]+np.pi/2, ra[i], nest=True))
190         labels = np.array(labels)
191     return labels
192
193 def create_array_from_hdf(input_paths, use_KAGRA):
194     h1_strain, l1_strain = np.zeros((0,512)), np.zeros((0,512))
195     v1_strain, k1_strain = np.zeros((0,512)), np.zeros((0,512))
196
197     for i, input_path in enumerate(input_paths):
198         with h5py.File(input_path, 'r') as f:
199             h1_strain_i = f['injection_samples']['h1_strain'][:]
200             l1_strain_i = f['injection_samples']['l1_strain'][:]
201             v1_strain_i = f['injection_samples']['v1_strain'][:]
202             k1_strain_i = f['injection_samples']['k1_strain'][:]
203
204             h1_strain = np.concatenate([h1_strain, h1_strain_i])
205             l1_strain = np.concatenate([l1_strain, l1_strain_i])
206             v1_strain = np.concatenate([v1_strain, v1_strain_i])
207             k1_strain = np.concatenate([k1_strain, k1_strain_i])
208
209     l = len(h1_strain)
210     strains = {0:h1_strain, 1:l1_strain, 2:v1_strain, 3:k1_strain}

```



```
211
212     # Take mean 0 and normalize
213     for i in range(1):
214         for j in range(4):
215             strains[j][i] -= np.mean(strains[j][i])
216             strains[j][i] /= (np.abs(strains[j][i])).max()
217
218     if use_KAGRA:
219         return np.dstack([h1_strain, l1_strain, v1_strain, k1_strain])
220     else:
221         return np.dstack([h1_strain, l1_strain, v1_strain])
222
223 def create_test_dataloader(arr, labels, n_sectors, batch_size):
224     X_test = torch.from_numpy(arr).float()
225     y_test = torch.from_numpy(labels).long()
226     test_dataset = torch.utils.data.TensorDataset(X_test, y_test)
227     test_dataloader = torch.utils.data.DataLoader(
228         test_dataset, batch_size=batch_size, shuffle=False, num_workers=2
229     )
230     return test_dataloader
```