卒業論文

機械学習を用いた宇宙線電子スペクトルのカットオフエネルギーの予測

早稲田大学
先進理工学部 物理学科
平本亮
Motz Holger Martin 研究室

2025 年 10 月 22 日

概要

概要 desu
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要
概要

目次

図目次

表目次

1 序章

1.1 宇宙線

宇宙線とは、宇宙から飛来する高エネルギーの荷電粒子のことである。現在では宇宙線の定義は広く、宇宙線と呼ばれるものの中には高エネルギー γ 線のような電磁波やニュートリノのような中性粒子も含まれるが、本稿では原点に立ち返って高エネルギー荷電粒子のことを指すこととする。宇宙線のうち、9 割は陽子であり、その他の重い原子核や電子などが残りの割合を占める。宇宙線のエネルギーは非常に広範囲にわたり、数 MeV から 10^{20}eV を超えるものまで観測されている。宇宙線はシンクロトロン放射や逆コンプトン散乱によってエネルギー損失が生じ、エネルギースペクトルは指数関数的に減少する。宇宙線のエネルギースペクトルは現在までに観測された宇宙線のうち、最も高いエネルギーを持つものは 1991 年に観測された 3.2×10^{20}eV の宇宙線である。

宇宙線のような荷電粒子は銀河内を伝播するときに銀河磁場によって進行方向が曲げられ、拡散的に運動する。そのため、宇宙線の到来方向を観測しても起源を特定できないことが多い。故に宇宙線の起源や加速機構については、候補はあるものの未だ解明されていないことも多い。

2 CALET

3 宇宙線電子

宇宙線電子 (Cosmic Ray Electron) は、宇宙線のうち、電子や陽電子のことを指す。宇宙線電子のエネルギーは MeV から TeV スケールである。宇宙線電子の起源は候補がいくつかあるが、特に TeV 領域の宇宙線電子の起源は超新星残骸 (Super Nova Remnant:SNR) であると考えられている。また宇宙線電子は質量が小さく、シンクロトロン放射や逆コンプトン散乱によるエネルギー損失が大きいので、TeV 領域の宇宙線電子の伝播距離は $\lesssim$1kpc、伝播年数は $\lesssim 10^5$ 年までに制限される。したがって TeV 領域の宇宙線電子の起源は近傍の超新星残骸に絞られる。

3.1 加速機構

宇宙線の加速機構についても依然はっきりとしたものは分かっていないが、現在の有力な加速機構は衝撃波加速 (First Order Fermi Acceleration) である。超新星爆発によって生じた衝撃波が

3.2 伝播機構

4 機械学習

本研究ではシミュレーションによって得られた宇宙線電子のデータをニューラルネットワークを用いて学習し、得られたスペクトルデータからカットオフエネルギーを予測することを目的とする。機械学習 (Machine Learning) は一般に、人工知能 (Artificial Intelligence:AI) にデータを学習させ、未知のデータに対して予測をする技術である。さらに機械学習は正解データを用いて学習を進める教師あり学習と、正解データなしで探索的に学習を進める教師なし学習に分けられる。本研究では教師あり学習を、さらに言えば深層学習 (Deep Learning) を用いる。

4.1 機械学習の仕組み

機械学習 (以下では教師あり学習のみを指す) の大まかな構造は、入力→変換→出力である。はじめこの変換はランダムに設定されているが。入力データを出力データに変換した後、出力データと正解データの誤差 (Loss) を損失関数 (Loss Function) によって評価し、誤差を小さくするように変換を更新する。この変換の更新のことを “学習”(Training) と呼ぶ。

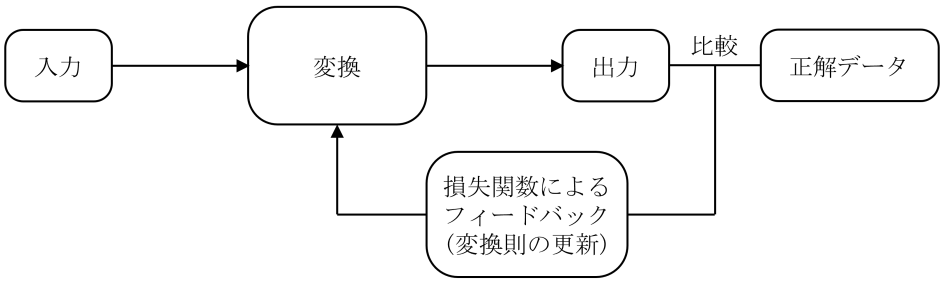

図 4.1 機械学習の流れ

この “学習” を繰り返すことで次第に出力が正解に近づいていき、学習用の入力データ (トレーニングデータ) 以外の未知の入力データに対しても正解に近い出力を返すことができる。

4.2 機械学習の構造

前節では機械学習における変換の構造については明らかにしていない。ここではその構造についてを例示的に説明する。まず、入力データは一般にテンソル (Tensor) で表現される。実際には 1 階のテンソルであるベクトルを用いることが多い。同様に、出力データもテンソルで表現される。それぞれのテンソルは入力層 (Input Layer)、出力層 (Output Layer) と呼ばれる。各層の要素 (成分) のことをノード (Node) と呼び、層間のノードの結合は重み (Weight) で関係づけられる。重みは結合の “強さ” と表現されることもある。一つの結合に対し、前の層のノードの値と重みの積が次の層のノードとなる。一つのノードに、前の層のノードが複数結合するときは、結合する前の層の各ノードと重みの積の総和が次の層のノードとなる。例えば、図 4.2 におけるノード M_1 の値は

$$M_1 = w_{11}I_1 + w_{12}I_2 + w_{13}I_3 + w_{14}I_4 \tag{4.1}$$

となる。ここで I_i は入力層のノード、w_{ij} はノード I_i とノード M_j の結合の重みである。

このような層間のノードの結合の構造をニューラルネットワーク (Neural Network) と呼ぶ。図 4.2 のように前の層のすべてのノードが自身のノードと結合している層を全結合層 (Fully Connected Layer) という。

機械学習における変換はこのような重み付き和と活性化関数 (Activation Function) と呼ばれる非線形変換で構成される。普遍近似定理 (Universal Approximation Theorem) より、重み付き和ととある条件を満たす非線形関数と十分な数のノードがあれば、任意の変換を近似的に生成することができることが知られている。重み付き和と活性化関数による非線形変換、および変換後のノードをまとめて層 (Layer) と呼ぶ。出力層やのちに述べる中間層はこれに該当する。入力層はこれに該当しないが、出力層や中間層と同じように層として扱うことが多い。

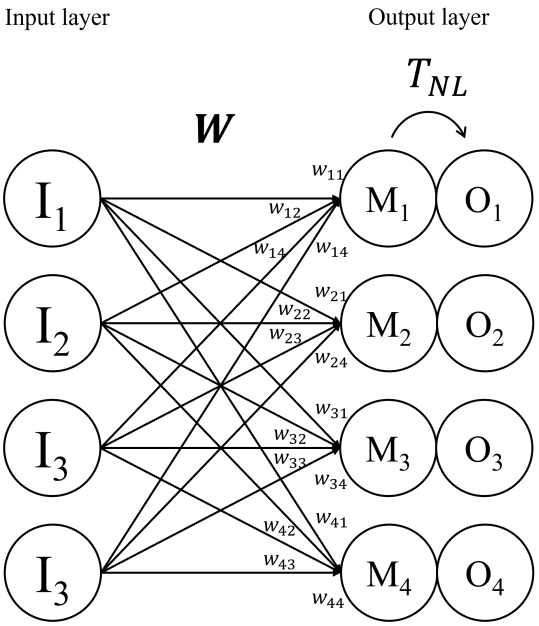

図 4.2　全結合層によるネットワーク構造

図 4.2 のように入力層と出力層のみでニューラルネットワークが構成される場合、出力層の i 番目のノード M_i は式 4.1 をより一般化して

$$M_i = \sum_j w_{ij} I_j \tag{4.2}$$

と書ける。ここで n 次元ベクトル $\boldsymbol{M}$ の第 i 成分を M_i、m 次元ベクトル $\boldsymbol{I}$ の第 j 成分を I_j とすると、式 4.2 は

$$(W)_{ij} = w_{ij} \tag{4.3}$$

と定義される $n \times m$ の重み行列 W を用いて,

$$\boldsymbol{M} = W\boldsymbol{I} \tag{4.4}$$

と書ける。式 4.4 は重み付き和を行列で表現したものである。さらに、活性化関数となる非線形変換を T_{NL} とすると、最終的な出力層のノードのベクトル表示 $\boldsymbol{O}$ は

$$\boldsymbol{O} = T_{NL}(\boldsymbol{M}) = T_{NL} W \boldsymbol{I} \tag{4.5}$$

と書ける。

4.3 深層学習

前節では機械学習における変換の構造について述べ、入力層と出力層のみの単層のニューラルネットワークの例を扱った。ここではさらに深層学習について述べる。深層学習は重み付き和と非線形変換で構成される層が複数存在するようなニューラルネットワークを用いた機械学習である。入力層と出力層の間に追加された層はすべて中間層 (Hidden Layer) に分類される。

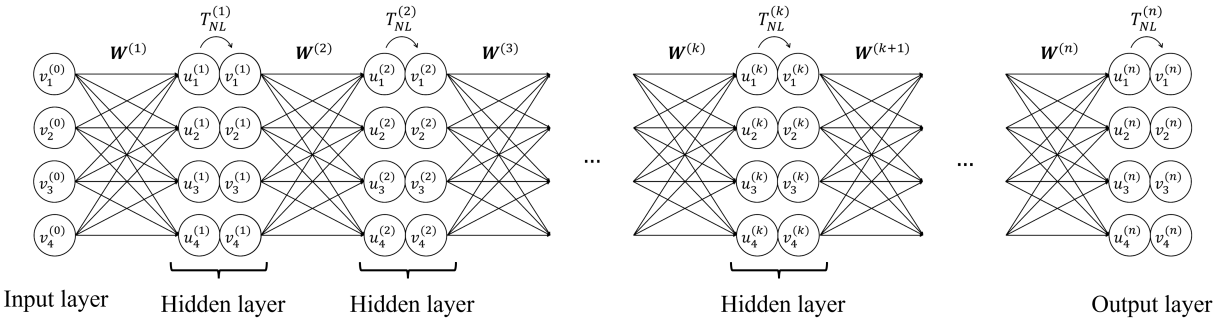

図 4.3 深層学習におけるニューラルネットワークの構造

深層学習では層を複数重ねることで、入力から出力までの変換をより複雑にすることができ、複雑なデータの分類、予測が可能になる。理論上は、普遍近似定理より単層のみでも任意の変換を近似的に生成することができるが、必要なノードの数が膨大になってしまい、計算に時間がかかってしまう。一方で、層を重ねることで、より少ないノード数で複雑な変換を近似できる。

深層学習による中間層のノードをベクトル表示すると、図 4.3 のようにひとつ前の中間層のノードからの伝搬は 4.2 節と同様に行列表示を用いて

$$\boldsymbol{v}^{(k)} = T_{NL}^{(k)}(\boldsymbol{u}^{(k)}) = T_{NL}^{(k)} W^{(k)} \boldsymbol{v}^{(k-1)} \tag{4.6}$$

と書ける。ここで $\boldsymbol{v}^{(k)}$ は k 番目の中間層の非線形変換後のノードのベクトル表示、$\boldsymbol{u}^{(k)}$ は k 番目の中間層の重み付き和後のノードのベクトル表示、$W^{(k)}$ は k 番目の中間層の重み行列、$T_{NL}^{(k)}$ は k 番目の中間層の非線形変換である。さらに、入力層を 0 番目の層、ベクトル表示を $\boldsymbol{I} = \boldsymbol{v}^{(0)}$、出力層を n 番目の層とすることで、出力層を含み、入力層を除いて n 個の層で構成されたニューラルネットワークの出力層のノードのベクトル表示 $\boldsymbol{O}$ は

$$\boldsymbol{O} = \boldsymbol{v}^{(n)} = T_{NL}^{(n)} W^{(n)} T_{NL}^{(n-1)} W^{(n-1)} \cdots T_{NL}^{(k)} W^{(k)} \cdots T_{NL}^{(1)} W^{(1)} \boldsymbol{v}^{(0)} \tag{4.7}$$

と書ける。

4.4 損失関数と最適化

損失関数 (Loss Function) による重みの更新は。機械学習において最も核心的な部分である。一般に損失関数は出力データと正解データの誤差を表す関数であり、この損失関数を小さくすることで “学習” を進め、正解率を上げる。したがって “学習を” を進めることは損失関数を最小にするような最適化問題を解くことに他ならない。この最適化問題は一般に勾配降下法 (Gradient Descent Method) を用いて解かれる。

4.4.1 勾配降下法

損失関数は出力データ $\boldsymbol{O}$ と正解データ $\hat{\boldsymbol{O}}$ を用いて $L(\boldsymbol{O}, \hat{\boldsymbol{O}})$ と表される。出力データ $\boldsymbol{O}$ は重み $W^{(k)}$ に依存するので、損失関数 L も重み $W^{(k)}$ に依存する。したがって、損失関数 L の重み $W^{(k)}$ 方向の勾配 $\frac{\partial L}{\partial W^{(k)}}$ を計算でき、勾配の逆方向、つまり L が小さくなる方向に重みを平行移動させればよい。つまり、現在の重みを $W^{(k)}_{old}$、学習率 (Learning Rate) を η とすると、新たに更新された重み $W^{(k)}_{new}$ は

$$W^{(k)}_{new} = W^{(k)}_{old} - \eta \left. \frac{\partial L}{\partial W^{(k)}} \right|_{W^{(k)}=W^{(k)}_{old}} \tag{4.8}$$

と書ける。これを逐次的に繰り返すことで損失関数 L を小さくし、“学習” を進める。学習率 (Learning Rate)η は一回の更新でどれだけ重みを変化させるかを決定するパラメータである。学習率 η が大きすぎると、損失関数が最小にとならずに発散してしまう。逆に、学習率 η が小さすぎると極小に陥ってしまい、真にの最小値に到達できないことがある。

4.4.2 誤差逆伝播法

重みを更新するために損失関数の重み方向の勾配を計算する必要がある。この勾配の計算には誤差逆伝播法 (Back Propagation Method) が用いられる。誤差逆伝播法は入力→出力とは逆方向に勾配を計算し、重みを更新する方法である。まず、出力層のにつながる重み $W^{(n)}$ 方向の損失関数 $L(\boldsymbol{O}, \hat{\boldsymbol{O}})$ の勾配は $\boldsymbol{O} = T_{NL} W^{(n)} \boldsymbol{v}^{(n-1)}$ より、

$$\frac{\partial L}{\partial W^{(n)}} = \frac{\partial L}{\partial \boldsymbol{O}} \cdot \frac{\partial \boldsymbol{O}}{\partial W^{(n)}} \tag{4.9}$$

と書ける。