ニューラルネットはなぜ動くのか？

数学的解像度で解き明かす基礎理論

ゆーごん著

February 12, 2026

前書き

ニューラルネットワークの数学的基礎第1巻

本シリーズは、ニューラルネットワークを現象学的ブラックボックスではなく、厳密な数学的対象として扱います。パーセプトロンからTransformerまで、全導出を行列微積分・確率論・関数解析の言葉で記述。特に以下の数学的貢献を重視：

- 収束保証: パーセプトロン定理からNTKグローバル最適化まで
- 表現力解析: 普遍近似からHessian BBP遷移・二重降下現象
- 高次元挙動: HDLSS下でのNTKスペクトル解析 ($d \gg 1$ 特化)

第1巻では順伝播・逆伝播・最適化の基礎数学を確立。第2巻以降でロボティクス（運動学ヤコビアン×**NTK**）、画像処理（**Vision Transformer**のスペクトル解析）、高次元統計応用へと展開します。

対象読者: 数学統計専攻の大学院生・研究者。実装前提の直感論を排除し、定理証明中心。各章末演習で導出を再現可能。

読了後、あなたはNNを「解ける関数系」として扱えます。

Contents

Chapter 1

数学的準備

1.1 ベクトルと行列

1.1.1 ベクトル空間とノルム

ベクトル $\mathbf{v} \in \mathbb{R}^n$ は n 個の実数の順序付きリストである：

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{1.1}$$

ベクトルのユークリッドノルム（または L^2 ノルム）は次のように定義される：

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2} = \sqrt{\mathbf{v}^T \mathbf{v}} \tag{1.2}$$

より一般的に、L^p ノルムは次のようになる：

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} \tag{1.3}$$

$p = 1$（マンハッタン距離）の場合：

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i| \tag{1.4}$$

$p = \infty$（最大絶対値）の場合：

$$\|\mathbf{v}\|_\infty = \max_i |v_i| \tag{1.5}$$

1.1.2 内積と幾何学的解釈

2つのベクトル $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ の内積（ドット積）は次のように定義される：

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \mathbf{a}^T \mathbf{b} \tag{1.6}$$

幾何学的解釈: 内積は、ベクトル間の角度 θ と次の関係がある：

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \cos\theta \tag{1.7}$$

これにより、次の式が得られる：

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{1.8}$$

重要な観察:

- $\theta = 0°$（平行）の場合： $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\|$ （最大）
- $\theta = 90°$（直交）の場合： $\mathbf{a} \cdot \mathbf{b} = 0$
- $\theta = 180°$（反平行）の場合： $\mathbf{a} \cdot \mathbf{b} = -\|\mathbf{a}\|\|\mathbf{b}\|$ （最小）

ニューラルネットワークにおいて、内積 $\mathbf{w} \cdot \mathbf{x}$ は重みと入力の整合性（アライメント）を測定する。整合性が高い（角度が小さい）ほど、大きな活性値が得られる。

1.1.3 行列演算

行列 $\mathbf{A} \in \mathbb{R}^{m \times n}$ は m 行 n 列を持つ：

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{1.9}$$

行列とベクトルの積：もし $\mathbf{A} \in \mathbb{R}^{m \times n}$ かつ $\mathbf{x} \in \mathbb{R}^n$ ならば、$\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^m$ であり、次のように計算される：

$$y_i = \sum_{j=1}^{n} a_{ij} x_j = \mathbf{a}_i^T \mathbf{x} \tag{1.10}$$

ここで $\mathbf{a}_i$ は $\mathbf{A}$ の第 i 行である。各出力 y_i は第 i 行と $\mathbf{x}$ の内積であることに注意。

行列同士の積：もし $\mathbf{A} \in \mathbb{R}^{m \times n}$ かつ $\mathbf{B} \in \mathbb{R}^{n \times p}$ ならば、$\mathbf{C} = \mathbf{A}\mathbf{B} \in \mathbb{R}^{m \times p}$ であり、次のように計算される：

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = \mathbf{a}_i^T \mathbf{b}_j \tag{1.11}$$

ここで $\mathbf{a}_i$ は $\mathbf{A}$ の第 i 行、$\mathbf{b}_j$ は $\mathbf{B}$ の第 j 列である。つまり、位置 (i, j) の要素は $\mathbf{A}$ の第 i 行と $\mathbf{B}$ の第 j 列の内積である。

1.1.4 重要な行列の性質

転置: 行と列を入れ替えること。もし $\mathbf{A} \in \mathbb{R}^{m \times n}$ ならば、$\mathbf{A}^T \in \mathbb{R}^{n \times m}$ であり、以下のようになる：

$$(\mathbf{A}^T)_{ij} = a_{ji} \tag{1.12}$$

転置の性質:

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T \quad \text{(順序が逆になる！)} \tag{1.13}$$

フロベニウスノルム: 行列 $\mathbf{A} \in \mathbb{R}^{m \times n}$ に対して：

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} = \sqrt{\text{trace}(\mathbf{A}^T\mathbf{A})} \tag{1.14}$$

1.2 微積分: 最適化の基礎

1.2.1 導関数と連鎖律

一変数関数 $f : \mathbb{R} \to \mathbb{R}$ の点 x における導関数は次の通りである：

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.15}$$

幾何学的には、$f'(x)$ は x における f の接線の傾きである。
連鎖律: もし $y = f(u)$ かつ $u = g(x)$ ならば、次のようになる：

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} \tag{1.16}$$

より一般的に、合成関数 $y = f(g(h(x)))$ に対しては：

$$\frac{dy}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx} \tag{1.17}$$

例: $y = \sin(x^2)$ とする。$u = x^2$ と置くと $y = \sin(u)$ である。

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} = \cos(u) \cdot 2x = 2x\cos(x^2) \tag{1.18}$$

1.2.2 偏微分と勾配

多変数関数 $f : \mathbb{R}^n \to \mathbb{R}$ の変数 x_i に関する偏微分は次の通りである：

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{h} \tag{1.19}$$

勾配はすべての偏微分を並べたベクトルである：

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{1.20}$$

方向微分: 方向 $\mathbf{d}$（単位ベクトル）への f の変化率は次のようになる：

$$\nabla_{\mathbf{d}} f = \mathbf{d}^T \nabla f = \nabla f \cdot \mathbf{d} \tag{1.21}$$

勾配 ∇f は最急勾配（最も急激に増加する方向）を指す。負の勾配 $-\nabla f$ は最急降下（最も急激に減少する方向）を指す。

例: $f(x, y) = x^2 + xy + 3y^2$ とする。

$$\frac{\partial f}{\partial x} = 2x + y, \quad \frac{\partial f}{\partial y} = x + 6y \tag{1.22}$$

$$\nabla f = \begin{bmatrix} 2x + y \\ x + 6y \end{bmatrix} \tag{1.23}$$

点 $(x, y) = (1, 2)$ において：

$$\nabla f|_{(1,2)} = \begin{bmatrix} 2(1) + 2 \\ 1 + 6(2) \end{bmatrix} = \begin{bmatrix} 4 \\ 13 \end{bmatrix} \tag{1.24}$$

1.2.3 行列微積分

行列 $\mathbf{A} \in \mathbb{R}^{m \times n}$ に依存するスカラー関数 $f(\mathbf{A})$ に対して、勾配（または導関数）は行列となる：

$$\frac{\partial f}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \frac{\partial f}{\partial a_{12}} & \cdots \\ \frac{\partial f}{\partial a_{21}} & \frac{\partial f}{\partial a_{22}} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{1.25}$$

行列微積分の重要な恒等式:

1. 線形関数: $f(\mathbf{x}) = \mathbf{a}^T \mathbf{x}$ ならば、$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{a}$
2. 二次形式: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$ ならば、$\frac{\partial f}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x} = 2\mathbf{A}\mathbf{x}$ （$\mathbf{A}$ が対称行列の場合）
3. 行列積: $f(\mathbf{A}) = \text{trace}(\mathbf{A}^T \mathbf{B})$ ならば、$\frac{\partial f}{\partial \mathbf{A}} = \mathbf{B}$
4. 損失関数 $\mathcal{L}(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{W}\mathbf{x}\|_2^2$ に対して：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = (\mathbf{W}\mathbf{x} - \mathbf{y})\mathbf{x}^T \tag{1.26}$$

記法、次元、およびラベルの規約

本セクションでは、各章で曖昧さを避けるために、主要な記号、テンソルの形状、およびラベルの規約を固定する。

基本記号と集合

- スカラーは小文字（例：$x \in \mathbb{R}$）、ベクトルは太字の小文字（例：$\mathbf{x} \in \mathbb{R}^d$）、行列は大文字（例：$A \in \mathbb{R}^{m \times n}$）で表す。
- ユークリッドノルムは $\|\mathbf{v}\|_2$、フロベニウスノルムは $\|A\|_F$ とする。
- 長さ m の全要素が1のベクトルは $\mathbf{1} \in \mathbb{R}^m$ とする。

データ、ミニバッチ、形状

入力次元を d、クラス数を K、データセットサイズを N とする。

- 単一の例：$(\mathbf{x}, \mathbf{y})$ ここで $\mathbf{x} \in \mathbb{R}^d$。
- サイズ m のミニバッチ（列積み上げ規約）：

$$X = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(m)} \end{bmatrix} \in \mathbb{R}^{d \times m}.$$

  これはネットワークの章で使用されるベクトル化された順伝播形式と一致する。

ネットワークアーキテクチャとパラメータ

層の幅が次のような L 層の順伝播ネットワークを考える：

$$n_0 = d, \quad n_1, \ldots, n_{L-1}, \quad n_L = \begin{cases} 1 & \text{二値出力（シグモイド）} \\ K & K \text{ クラス出力（ソフトマックス）}. \end{cases}$$

パラメータは $\theta = \{W^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{L}$ である。
　各層 $\ell = 1, \ldots, L$ について：

$$W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, \qquad \mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}.$$

順伝播（単一例）：

$$\mathbf{a}^{(0)} = \mathbf{x}, \qquad \mathbf{z}^{(\ell)} = W^{(\ell)} \mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}, \qquad \mathbf{a}^{(\ell)} = \sigma^{(\ell)}(\mathbf{z}^{(\ell)}).$$

ここで $\mathbf{z}^{(\ell)}, \mathbf{a}^{(\ell)} \in \mathbb{R}^{n_\ell}$ である。

ベクトル化されたミニバッチ順伝播（列積み上げ）：

$$A^{(0)} = X \in \mathbb{R}^{n_0 \times m}, \qquad Z^{(\ell)} = W^{(\ell)} A^{(\ell-1)} + \mathbf{b}^{(\ell)} \mathbf{1}^\top \in \mathbb{R}^{n_\ell \times m}, \qquad A^{(\ell)} = \sigma^{(\ell)}(Z^{(\ell)}).$$

これはノートの完全ベクトル化セクションで使用されている規約と同じである。

誤差逆伝播の記法

1つの例に対する損失を $L(\hat{\mathbf{y}}, \mathbf{y})$ と表す。

逆伝播における重要な量はデルタである：

$$\boldsymbol{\delta}^{(\ell)} \;=\; \frac{\partial L}{\partial \mathbf{z}^{(\ell)}} \in \mathbb{R}^{n_\ell}.$$

この規約により、勾配は外積形式（単一例）となる：

$$\frac{\partial L}{\partial W^{(\ell)}} = \boldsymbol{\delta}^{(\ell)} \big(\mathbf{a}^{(\ell-1)}\big)^\top, \qquad \frac{\partial L}{\partial \mathbf{b}^{(\ell)}} = \boldsymbol{\delta}^{(\ell)}.$$

これらの式は、逆伝播の章での導出と一致する。

サイズ m のミニバッチの場合、平均目的関数（バッチ上の経験リスク）を定義し：

$$\mathcal{L}_{\text{batch}}(\theta) = \frac{1}{m} \sum_{i=1}^{m} L\big(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}\big),$$

それに応じて平均勾配を計算する（ミニバッチ勾配のセクションと同様）。

ラベルの規約（重要）

二値分類には2つの一般的なエンコーディングがある：

- **確率/BCE 規約:** $y \in \{0, 1\}$, $\hat{y} \in (0, 1)$ であり、$\hat{y} = \sigma(z)$（シグモイド）。これは BCE の章や数値計算例で使用される規約である。
- **パーセプトロン規約:** $y \in \{-1, +1\}$ であり、予測 $\hat{y} = \text{sign}(w^\top x + b)$。パーセプトロンの収束定理の記述に使用される。

2つの二値エンコーディング間の簡単な変換は以下の通り：

$$y_{\pm 1} = 2y_{01} - 1, \qquad y_{01} = \frac{y_{\pm 1} + 1}{2}.$$

古典的なパーセプトロン定理を議論するときは $y \in \{-1, +1\}$ を使用し、シグモイド/BCE を使用するときは $y \in \{0, 1\}$ を使用する。

多クラス分類ではワンホットベクトルを使用する：

$$\mathbf{y} \in \{0, 1\}^K, \qquad \sum_{k=1}^{K} y_k = 1, \qquad \mathbf{p} = \text{softmax}(\mathbf{z}) \in (0, 1)^K, \qquad \sum_{k=1}^{K} p_k = 1.$$

これはソフトマックス + カテゴリカル交差エントロピーの設定とその勾配の単純化と一致する。

一般的な非線形関数（クイックリファレンス）

- シグモイド: $\sigma(z) = \frac{1}{1+e^{-z}}$, $\sigma'(z) = \sigma(z)(1-\sigma(z))$.
- ReLU: $\mathrm{ReLU}(z) = \max(0, z)$ であり、準勾配は $z > 0$ で 1、$z < 0$ で 0、$z = 0$ で $[0, 1]$ の任意の値。
- ソフトマックス: $p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, ヤコビアン $\frac{\partial p_i}{\partial z_j} = p_i(\delta_{ij} - p_j)$.

Chapter 2

パーセプトロン

数学的目標

- 決定境界が超平面 $(w^\top x + b = 0)$ であることを幾何学的に理解する。
- 符号付き距離 $d = \frac{w^\top x + b}{\|w\|}$ の導出を習得する。
- パーセプトロン収束定理の証明（バウンド $\left(\frac{R}{\gamma}\right)^2$）を追跡する。

2.1　超平面としての決定境界

パーセプトロンは二値線形分類器である。入力ベクトル $\mathbf{x} \in \mathbb{R}^d$、重み $\mathbf{w} \in \mathbb{R}^d$、バイアス $b \in \mathbb{R}$ が与えられたとき、スコアを次のように定義する：

$$z(\mathbf{x}) \;=\; \mathbf{w}^\top \mathbf{x} + b. \tag{2.1}$$

パーセプトロンの予測は以下の通り：

$$\hat{y}(\mathbf{x}) \;=\; \mathrm{step}(z(\mathbf{x})) \;=\; \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x} + b \geq 0, \\ 0 & \text{if } \mathbf{w}^\top \mathbf{x} + b < 0. \end{cases} \tag{2.2}$$

（等価的に、ラベル $y \in \{-1, +1\}$ を使用する場合、$\hat{y} = \mathrm{sign}(\mathbf{w}^\top \mathbf{x} + b)$ と書かれることが多い。）

2.1.1　超平面の解釈

決定境界はスコアがちょうどゼロになる点の集合である：

$$\mathcal{H} \;=\; \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} + b = 0\}. \tag{2.3}$$

この集合 $\mathcal{H}$ は $\mathbb{R}^d$ における**超平面**であり、法線ベクトル $\mathbf{w}$ を持つ。

超平面は $\mathbb{R}^d$ を2つの半空間に分割する：

$$\mathcal{H}_+ = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} + b > 0\}, \tag{2.4}$$

$$\mathcal{H}_- = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} + b < 0\}. \tag{2.5}$$

幾何学的には、$\mathbf{w}$ は正と分類される側を指す。

2.1.2 スケール不変性

任意の $\alpha > 0$ に対して、$(\mathbf{w}, b)$ を $(\alpha\mathbf{w}, \alpha b)$ に置き換えても境界は変化しない：

$$(\alpha\mathbf{w})^\top \mathbf{x} + \alpha b = \alpha(\mathbf{w}^\top \mathbf{x} + b), \tag{2.6}$$

したがって、$\mathbf{w}^\top \mathbf{x} + b = 0$ となるのは $(\alpha\mathbf{w})^\top \mathbf{x} + \alpha b = 0$ のときのみである。分類において重要なのは $\mathbf{w}$ の向きと相対的なオフセット b だけである。

2.2 超平面への符号付き距離

$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} + b = 0\}$ とし、$\mathbf{w} \neq \mathbf{0}$ とする。点 $\mathbf{x}$ から超平面への**符号付き距離**は次の通りである：

$$d(\mathbf{x}, \mathcal{H}) \;=\; \frac{\mathbf{w}^\top \mathbf{x} + b}{\|\mathbf{w}\|_2}. \tag{2.7}$$

2.2.1 導出

$\mathbf{w}^\top \mathbf{x}_0 + b = 0$ となる任意の点 $\mathbf{x}_0 \in \mathcal{H}$ を選ぶ。 $\mathbf{x}_0$ から $\mathbf{x}$ へのベクトルは $\mathbf{x} - \mathbf{x}_0$ である。このベクトルを単位法線方向 $\mathbf{u} = \mathbf{w}/\|\mathbf{w}\|_2$ に射影すると、符号付き距離が得られる：

$$d(\mathbf{x}, \mathcal{H}) = \mathbf{u}^\top (\mathbf{x} - \mathbf{x}_0) = \frac{\mathbf{w}^\top (\mathbf{x} - \mathbf{x}_0)}{\|\mathbf{w}\|_2} = \frac{\mathbf{w}^\top \mathbf{x} - \mathbf{w}^\top \mathbf{x}_0}{\|\mathbf{w}\|_2}. \tag{2.8}$$

$\mathbf{w}^\top \mathbf{x}_0 = -b$ を用いると (2.7) が得られる。特に：

- $d(\mathbf{x}, \mathcal{H}) > 0$ であるための必要十分条件は $\mathbf{x} \in \mathcal{H}_+$、
- $d(\mathbf{x}, \mathcal{H}) < 0$ であるための必要十分条件は $\mathbf{x} \in \mathcal{H}_-$、
- $|d(\mathbf{x}, \mathcal{H})|$ は境界までのユークリッド距離に等しい。

2.3 パーセプトロン学習アルゴリズム

2.3.1 ラベルの規約

収束定理においては、ラベル $y_i \in \{-1, +1\}$ を用いるのが標準的である。訓練データ $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ が与えられたとき、予測を次のように定義する：

$$\hat{y}_i = \operatorname{sign}(\mathbf{w}^\top \mathbf{x}_i + b). \tag{2.9}$$

2.3.2 更新ルール

$\mathbf{w}_0 = \mathbf{0}$ および $b_0 = 0$ で初期化する。反復 t において、誤分類された例 $(\mathbf{x}_i, y_i)$ を選ぶ。すなわち：

$$y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t) \leq 0. \tag{2.10}$$

そしてパーセプトロン更新を適用する：

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta\, y_i\, \mathbf{x}_i, \tag{2.11}$$

$$b_{t+1} = b_t + \eta\, y_i, \tag{2.12}$$

ここで $\eta > 0$ は学習率である。（等価的に、バイアスを拡張ベクトル $\tilde{\mathbf{x}} = [\mathbf{x}^\top, 1]^\top$ と重み $\tilde{\mathbf{w}} = [\mathbf{w}^\top, b]^\top$ に吸収させ、単一の更新として書くこともできる。）

2.4 パーセプトロンの収束定理（概略）

2.4.1 線形分離可能性とマージン

データが線形分離可能である、すなわちある $(\mathbf{w}_\star, b_\star)$ が存在して次を満たすと仮定する：

$$y_i(\mathbf{w}_\star^\top \mathbf{x}_i + b_\star) \geq 1 \quad \text{for all } i. \tag{2.13}$$

$R = \max_i \|\mathbf{x}_i\|_2$ とする。分離平面 $(\mathbf{w}_\star, b_\star)$ の（幾何学的）マージンを次のように定義する：

$$\gamma \;=\; \min_i \frac{y_i(\mathbf{w}_\star^\top \mathbf{x}_i + b_\star)}{\|\mathbf{w}_\star\|_2}. \tag{2.14}$$

(2.13) の下では、$\gamma \geq 1/\|\mathbf{w}_\star\|_2$ が成り立つ。

2.4.2 定理

定理 (パーセプトロンの収束). 訓練セットがマージン $\gamma > 0$ で線形分離可能であり、$\|\mathbf{x}_i\| \leq R$ であるならば、パーセプトロンは高々

$$M \;\leq\; \left(\frac{R}{\gamma}\right)^2 \tag{2.15}$$

回の誤り（更新）を行い、したがって有限回の更新後に終了する。

2.4.3 証明の概略

簡単のため $\eta = 1$ とし、バイアス b を含めるために拡張ベクトルを用いる（拡張なしでも同様の議論が成り立つ）。 $\mathbf{w}_t$ を t 回目の更新後の重みとする。

ステップ 1: 最適な分離平面への進行。 誤分類された $(\mathbf{x}_i, y_i)$ を用いた更新について：

$$\mathbf{w}_{t+1}^\top \mathbf{w}_\star = (\mathbf{w}_t + y_i \mathbf{x}_i)^\top \mathbf{w}_\star = \mathbf{w}_t^\top \mathbf{w}_\star + y_i\, \mathbf{x}_i^\top \mathbf{w}_\star. \tag{2.16}$$

分離可能性により、$y_i\, \mathbf{x}_i^\top \mathbf{w}_\star \geq \gamma \|\mathbf{w}_\star\|_2$ （バイアス/拡張の規約による）、したがって M 回の誤り後：

$$\mathbf{w}_M^\top \mathbf{w}_\star \;\geq\; M\, \gamma\, \|\mathbf{w}_\star\|_2. \tag{2.17}$$

ステップ 2: ノルムの成長が抑制される。 二乗ノルムは次のように変化する：

$$\|\mathbf{w}_{t+1}\|_2^2 = \|\mathbf{w}_t + y_i \mathbf{x}_i\|_2^2 = \|\mathbf{w}_t\|_2^2 + 2 y_i\, \mathbf{w}_t^\top \mathbf{x}_i + \|\mathbf{x}_i\|_2^2. \tag{2.18}$$

点が誤分類されているため、$y_i(\mathbf{w}_t^\top \mathbf{x}_i) \leq 0$、ゆえに

$$\|\mathbf{w}_{t+1}\|_2^2 \leq \|\mathbf{w}_t\|_2^2 + \|\mathbf{x}_i\|_2^2 \leq \|\mathbf{w}_t\|_2^2 + R^2. \tag{2.19}$$

帰納法により、

$$\|\mathbf{w}_M\|_2^2 \leq M R^2 \quad \Rightarrow \quad \|\mathbf{w}_M\|_2 \leq R\sqrt{M}. \tag{2.20}$$

ステップ 3: コーシー・シュワルツによる結合。 コーシー・シュワルツの不等式より、

$$\mathbf{w}_M^\top \mathbf{w}_\star \le \|\mathbf{w}_M\|_2 \, \|\mathbf{w}_\star\|_2. \tag{2.21}$$

(2.17) と (2.20) を代入すると：

$$M\,\gamma\,\|\mathbf{w}_\star\|_2 \le \|\mathbf{w}_M\|_2 \, \|\mathbf{w}_\star\|_2 \le R\sqrt{M}\,\|\mathbf{w}_\star\|_2. \tag{2.22}$$

$\|\mathbf{w}_\star\|_2 > 0$ をキャンセルして整理すると：

$$M\gamma \le R\sqrt{M} \quad \Rightarrow \quad \sqrt{M} \le \frac{R}{\gamma} \quad \Rightarrow \quad M \le \left(\frac{R}{\gamma}\right)^2, \tag{2.23}$$

これで (2.15) が証明された。

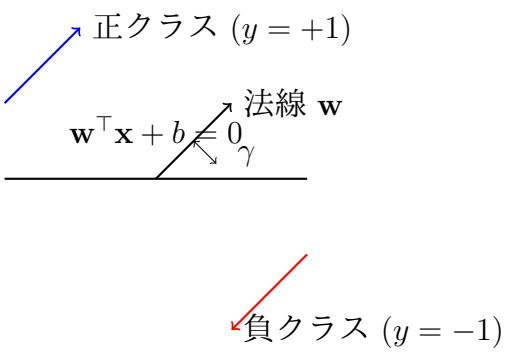


Figure 2.1: マージン $\gamma = \min_i \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$ の視覚化。

2.4.4 備考

- データが線形分離可能でない場合、パーセプトロンの更新は循環し、収束しない可能性がある。
- 境界は R（データのスケール）と γ（分離可能マージン）に依存する。マージンが大きいほど更新回数は少なくなる。

演習問題

- **初級**: データセット $x_1 = (1, 0), x_2 = (-1, 0)$ に対する最大マージン超平面を求めよ。
- **中級**: バイアス項 b を重みベクトルに吸収させる拡張法において、マージンの定義がどう変わるか示せ。
- **上級**: パーセプトロン学習アルゴリズムが、線形分離不可能なデータに対しては収束しないことを証明せよ（循環の例を示せ）。

Chapter 3

順伝播型ニューラルネットワーク

数学的目標

- アフィン変換と非線形活性化の合成としてのネットワーク構造を理解する。
- 完全ベクトル化された順伝播 $Z = WA + b\mathbf{1}^\top$ を実装できる。
- 活性化関数の微係数とヤコビアン行列を導出する。

3.1　スカラー和から行列形式へ

順伝播型ニューラルネットワーク（フィードフォワードニューラルネットワーク）は、複数のアフィン写像と非線形性を積み重ねることでパーセプトロンを一般化したものである。

3.1.1　単一ニューロン（スカラー形式）

入力 $\mathbf{x} \in \mathbb{R}^d$、重み $\mathbf{w} \in \mathbb{R}^d$、バイアス $b \in \mathbb{R}$ とする。単一ニューロンは次を計算し：

$$z = \sum_{i=1}^{d} w_i x_i + b = \mathbf{w}^\top \mathbf{x} + b, \tag{3.1}$$

その後、ある非線形性 σ に対して活性化 $a = \sigma(z)$ を出力する。

3.1.2　ニューロンの層（総和インデックス表記）

$n_{\ell-1}$ 個の入力と n_ℓ 個のニューロンを持つ層 ℓ を考える。 $\mathbf{a}^{(\ell-1)} \in \mathbb{R}^{n_{\ell-1}}$ を層 ℓ への入力活性化ベクトルとする。ニューロン $j \in \{1, \dots, n_\ell\}$ に対して、入力ユニット k からニューロン j への重みを $W_{jk}^{(\ell)}$、バイアスを $b_j^{(\ell)}$ と定義する。すると：

$$z_j^{(\ell)} = \sum_{k=1}^{n_{\ell-1}} W_{jk}^{(\ell)} \, a_k^{(\ell-1)} + b_j^{(\ell)}. \tag{3.2}$$

3.1.3 ニューロンの層（行列形式）

すべての $z_j^{(\ell)}$ をベクトル $\mathbf{z}^{(\ell)} \in \mathbb{R}^{n_\ell}$ にまとめ、次のように定義する：

$$\mathbf{W}^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, \quad \mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}, \quad \mathbf{a}^{(\ell-1)} \in \mathbb{R}^{n_{\ell-1}}. \tag{3.3}$$

すると (3.2) はコンパクトなベクトル形式になる：

$$\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}. \tag{3.4}$$

活性化関数を要素ごとに適用すると：

$$\mathbf{a}^{(\ell)} = \sigma\big(\mathbf{z}^{(\ell)}\big)\,. \tag{3.5}$$

3.1.4 ミニバッチ（完全ベクトル化）形式

サイズ m のミニバッチに対して、入力を行列として積み重ねる：

$$\mathbf{A}^{(\ell-1)} = \Big[\mathbf{a}_1^{(\ell-1)} \quad \cdots \quad \mathbf{a}_m^{(\ell-1)}\Big] \in \mathbb{R}^{n_{\ell-1}\times m}. \tag{3.6}$$

するとアフィン写像は次のようになる：

$$\mathbf{Z}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{A}^{(\ell-1)} + \mathbf{b}^{(\ell)}\mathbf{1}^{\top} \in \mathbb{R}^{n_\ell \times m}, \tag{3.7}$$

ここで $\mathbf{1} \in \mathbb{R}^m$ は全要素が1のベクトルである。活性化：

$$\mathbf{A}^{(\ell)} = \sigma\big(\mathbf{Z}^{(\ell)}\big)\,. \tag{3.8}$$

この行列形式は、効率的なGPU計算の基礎となっている。

3.2 関数合成としてのネットワーク

入力層を $\mathbf{a}^{(0)} = \mathbf{x} \in \mathbb{R}^{n_0}$ とする。深さ L の順伝播ネットワークは、$\ell = 1, \dots, L$ に対して (3.4)–(3.5) によって再帰的に定義される。

ネットワークの出力は

$$\hat{\mathbf{y}} = f(\mathbf{x};\theta) = \mathbf{a}^{(L)}, \tag{3.9}$$

ここで $\theta = \{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{L}$ はパラメータの集合である。

完全な合成を書き出すと：

$$f(\mathbf{x};\theta) = \sigma^{(L)}\Big(\mathbf{W}^{(L)}\,\sigma^{(L-1)}\big(\mathbf{W}^{(L-1)}\cdots\sigma^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})\cdots + \mathbf{b}^{(L-1)}\big) + \mathbf{b}^{(L)}\Big), \tag{3.10}$$

ここで $\sigma^{(\ell)}$ は層によって異なる場合がある（例：隠れ層では ReLU、出力ではシグモイドやソフトマックス）。

3.2.1 パラメータ数

学習可能なパラメータの数は次の通り：

$$\#\theta = \sum_{\ell=1}^{L} (n_\ell n_{\ell-1} + n_\ell) = \sum_{\ell=1}^{L} n_\ell(n_{\ell-1}+1). \tag{3.11}$$

n_ℓ が大きい、または深さ L が大きいと容量が増加するが、正則化なしでは過学習のリスクも高まる。

3.3 活性化関数と導関数

非線形活性化は不可欠である。これらがなければ、ネットワーク全体が単一のアフィン写像に崩壊してしまう。

3.3.1 なぜ非線形性が必要か

もしすべての層で $\sigma(z) = z$（純粋に線形なネットワーク）であるならば、

$$\mathbf{a}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}. \tag{3.12}$$

帰納法により、ネットワーク全体は次のようになる：

$$f(\mathbf{x}) = \mathbf{W}_{\text{eff}}\mathbf{x} + \mathbf{b}_{\text{eff}}, \tag{3.13}$$

ある有効な行列/ベクトル $(\mathbf{W}_{\text{eff}}, \mathbf{b}_{\text{eff}})$ に対して。したがって深さは表現力を高めない。ゆえに、σ は非線形でなければならない。

3.3.2 シグモイド

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.14}$$

導関数：

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)). \tag{3.15}$$

3.3.3 双曲線正接 (Tanh)

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}. \tag{3.16}$$

導関数：

$$\frac{d}{dz}\tanh(z) = 1 - \tanh^{2}(z). \tag{3.17}$$

3.3.4 ReLU

$$\mathrm{ReLU}(z) = \max(0, z) = \begin{cases} z & (z > 0), \\ 0 & (z \le 0). \end{cases} \tag{3.18}$$

準勾配（実用上使用される）：

$$\mathrm{ReLU}'(z) = \begin{cases} 1 & (z > 0), \\ 0 & (z < 0), \\ \text{[0,1] の任意の値} & (z = 0). \end{cases} \tag{3.19}$$

3.3.5 ソフトマックス

$\mathbf{z} \in \mathbb{R}^K$ に対して、

$$\mathrm{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, \ldots, K. \tag{3.20}$$

そのヤコビアンは

$$\frac{\partial \,\mathrm{softmax}(\mathbf{z})_i}{\partial z_j} = p_i(\delta_{ij} - p_j), \tag{3.21}$$

ここで $\mathbf{p} = \mathrm{softmax}(\mathbf{z})$ であり、δ_{ij} はクロネッカーのデルタである。行列形式：

$$\mathbf{J} = \mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^\top. \tag{3.22}$$

3.3.6 勾配消失の完全証明

深さ L のネットワークにおいて、誤差逆伝播は連鎖律により勾配を計算する：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} = \left(\prod_{\ell=1}^{L-1} \mathbf{W}^{(\ell+1)\top} \mathrm{diag}(\sigma'(\mathbf{z}^{(\ell)})) \right) \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}}. \tag{3.23}$$

シグモイド関数 $\sigma(z)$ の場合、最大微分係数は $\sigma'(0) = 1/4$ である。したがって、行列ノルムの劣乗法性により：

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} \right\| \leq \left(\prod_{\ell=1}^{L-1} \|\mathbf{W}^{(\ell+1)\top}\| \cdot \frac{1}{4} \right) \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}} \right\|. \tag{3.24}$$

もし重みが初期化時に小さく（例：$\|\mathbf{W}\| < 4$）、係数 $\frac{1}{4}\|\mathbf{W}\|$ が 1 より小さい場合、勾配は層数 L に対して指数関数的に減衰する：

$$\|\nabla_{\mathbf{h}^{(1)}}\| \leq (c)^{L-1} \|\nabla_{\mathbf{h}^{(L)}}\|, \quad c < 1. \tag{3.25}$$

これにより、深い層の重みはほとんど更新されなくなる。ReLU ($\sigma' = 1$) はこの係数 $1/4$ を排除し、勾配消失を緩和する。

3.4 具体的な小さな例（オプション）

ReLU 隠れ層とシグモイド出力を持つ $2 \to 2 \to 1$ ネットワークを考える。以下のように設定する：

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{W}^{(1)} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{3.26}$$

隠れ層の事前活性化：

$$\mathbf{z}^{(1)} = \begin{bmatrix} w_{11}x_1 + w_{12}x_2 + b_1 \\ w_{21}x_1 + w_{22}x_2 + b_2 \end{bmatrix}, \tag{3.27}$$

隠れ層の活性化：

$$\mathbf{a}^{(1)} = \begin{bmatrix} \max(0, z_1^{(1)}) \\ \max(0, z_2^{(1)}) \end{bmatrix}. \tag{3.28}$$

出力層：

$$z^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + b^{(2)} = u_1 a_1^{(1)} + u_2 a_2^{(1)} + b^{(2)}, \tag{3.29}$$

$$\hat{y} = \sigma(z^{(2)}) = \frac{1}{1 + e^{-z^{(2)}}}. \tag{3.30}$$

この具体的な形式により、次元を確認し、各層がアフィン写像とその後の非線形性であることを確認しやすくなる。

演習問題

- **初級**: 2×3 入力行列 X と 4×2 重み行列 W の積の次元を確認せよ。
- **中級**: $f(x) = \mathrm{ReLU}(Wx + b)$ の勾配 $\nabla_W f$ を導出せよ。
- **上級**: Xavier の初期化が、活性化の分散を層間で一定に保つための条件 $\mathrm{Var}(w) = 1/n$ を導け。

Chapter 4

損失関数

損失関数は、モデルの予測がターゲットからどれだけ離れているかを定量化する。訓練では通常、経験リスク（データセット上の平均損失）を最小化する。$\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$ をデータセットとし、モデル出力を $\hat{\mathbf{y}}^{(i)} = f_\theta(\mathbf{x}^{(i)})$ とする。

4.1 経験リスク最小化

サンプルごとの損失 $\ell(\hat{\mathbf{y}}, \mathbf{y})$ が与えられたとき、経験リスクを定義する：

$$\mathcal{L}(\theta) \;=\; \frac{1}{m}\sum_{i=1}^{m} \ell\big(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}\big)\,. \tag{4.1}$$

勾配ベースの学習には $\nabla_\theta \mathcal{L}(\theta)$ の計算が必要であるため、一般的な損失に対する勾配を導出する。

4.2 回帰: 平均二乗誤差 (MSE)

4.2.1 定義

$\mathbf{y} \in \mathbb{R}^K$ と予測 $\hat{\mathbf{y}} \in \mathbb{R}^K$ を用いた回帰の場合、平均二乗誤差 (MSE) 損失は次の通りである：

$$\ell_{\text{MSE}}(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 = \sum_{k=1}^{K} (\hat{y}_k - y_k)^2. \tag{4.2}$$

一般的なスケーリングされた変種では、勾配の係数 2 を除去するために $\frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$ を使用する。

データセット全体では：

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \|\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}\|_2^2. \tag{4.3}$$

4.2.2 予測に関する勾配

(4.2) より、$\hat{\mathbf{y}}$ に関する勾配は次のようになる：

$$\nabla_{\hat{\mathbf{y}}}\, \ell_{\text{MSE}}(\hat{\mathbf{y}}, \mathbf{y}) = 2(\hat{\mathbf{y}} - \mathbf{y}). \tag{4.4}$$

スケーリングされた損失 $\frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$ の場合、勾配は $\hat{\mathbf{y}} - \mathbf{y}$ となる。

4.3 二値分類: 二値交差エントロピー (BCE)

二値分類では $y \in \{0,1\}$、およびモデル出力 $\hat{y} \in (0,1)$ を $P(Y=1 \mid \mathbf{x})$ として解釈する。多くの場合、$\hat{y} = \sigma(z)$ である。ここで $z \in \mathbb{R}$ はロジット、σ はシグモイドである。

4.3.1 二値交差エントロピー (負の対数尤度)

1つの例に対する二値交差エントロピー (BCE) 損失は次の通りである：

$$\ell_{\mathrm{BCE}}(\hat{y}, y) = -\Big[y \log(\hat{y}) + (1-y)\log(1-\hat{y})\Big]. \tag{4.5}$$

データセット全体では：

$$\mathcal{L}_{\mathrm{BCE}}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell_{\mathrm{BCE}}(\hat{y}^{(i)}, y^{(i)}). \tag{4.6}$$

4.3.2 $\hat{y}$ に関する勾配

(4.5) を微分する：

$$\frac{\partial \ell_{\mathrm{BCE}}}{\partial \hat{y}} = -\left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})}. \tag{4.7}$$

4.3.3 シグモイド + BCE の簡略化 (ロジット勾配)

$\hat{y} = \sigma(z)$ とし、$\sigma'(z) = \hat{y}(1-\hat{y})$ とする（第3章より）。連鎖律により：

$$\frac{\partial \ell_{\mathrm{BCE}}}{\partial z} = \frac{\partial \ell_{\mathrm{BCE}}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \cdot \hat{y}(1-\hat{y}) = \hat{y} - y. \tag{4.8}$$

したがって、シグモイド + BCE の場合、ロジットにおける誤差信号は単に「予測 - ターゲット」となる。

4.4 多クラス分類: ソフトマックスとカテゴリカル交差エントロピー

K クラスを仮定する。ロジットを $\mathbf{z} \in \mathbb{R}^K$、確率を次のようにする：

$$\mathbf{p} = \mathrm{softmax}(\mathbf{z}), \qquad p_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}. \tag{4.9}$$

ラベルはワンホット $\mathbf{y} \in \{0,1\}^K$ であり、$\sum_k y_k = 1$ とする。

4.4.1 カテゴリカル交差エントロピー (CCE)

1つの例に対するカテゴリカル交差エントロピー (CCE) 損失は次の通りである：

$$\ell_{\text{CCE}}(\mathbf{p}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log(p_k). \tag{4.10}$$

データセット全体では：

$$\mathcal{L}_{\text{CCE}}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell_{\text{CCE}}(\mathbf{p}^{(i)}, \mathbf{y}^{(i)}). \tag{4.11}$$

4.4.2 ソフトマックス・ヤコビアン

ソフトマックスの導関数は次の形式（ヤコビアン）を持つ：

$$\frac{\partial p_i}{\partial z_j} = p_i(\delta_{ij} - p_j), \tag{4.12}$$

ここで δ_{ij} はクロネッカーのデルタである（$i = j$ なら $\delta_{ij} = 1$、それ以外は 0）。
等価的に、行列形式では：

$$\frac{\partial \mathbf{p}}{\partial \mathbf{z}} = \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^{\top}. \tag{4.13}$$

4.4.3 ソフトマックス + CCE の簡略化 (ロジット勾配)

ここで $\nabla_{\mathbf{z}} \ell_{\text{CCE}}$ を計算する。まず、

$$\frac{\partial \ell_{\text{CCE}}}{\partial p_i} = -\frac{y_i}{p_i}. \tag{4.14}$$

次に連鎖律を適用する：

$$\frac{\partial \ell_{\text{CCE}}}{\partial z_j} = \sum_{i=1}^{K} \frac{\partial \ell_{\text{CCE}}}{\partial p_i} \frac{\partial p_i}{\partial z_j} = \sum_{i=1}^{K} \left(-\frac{y_i}{p_i} \right) p_i(\delta_{ij} - p_j). \tag{4.15}$$

p_i をキャンセルする：

$$= -\sum_{i=1}^{K} y_i(\delta_{ij} - p_j) = -\sum_{i=1}^{K} y_i \delta_{ij} + \sum_{i=1}^{K} y_i p_j. \tag{4.16}$$

$\sum_{i=1}^{K} y_i \delta_{ij} = y_j$ および $\sum_{i=1}^{K} y_i = 1$ であるため、

$$\frac{\partial \ell_{\text{CCE}}}{\partial z_j} = -p_j + p_j \cdot 1 + (p_j - y_j) = p_j - y_j. \tag{4.17}$$

したがって、

$$\nabla_{\mathbf{z}} \ell_{\text{CCE}}(\text{softmax}(\mathbf{z}), \mathbf{y}) = \mathbf{p} - \mathbf{y}. \tag{4.18}$$

これは (4.8) の多クラス版である。

4.5 （オプション）ロバスト回帰のためのフーバー損失

回帰データに外れ値が含まれる場合、MSE は過敏になることがある。フーバー損失は MSE と MAE の間を補間する。

スカラー $y, \hat{y} \in \mathbb{R}$ と閾値 $\delta > 0$ に対して：

$$\ell_{\text{Huber}}(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \le \delta, \\ \delta\left(|y - \hat{y}| - \frac{\delta}{2}\right) & \text{if } |y - \hat{y}| > \delta. \end{cases} \tag{4.19}$$

その $\hat{y}$ に関する導関数は次の通りである：

$$\frac{\partial \ell_{\text{Huber}}}{\partial \hat{y}} = \begin{cases} \hat{y} - y & \text{if } |y - \hat{y}| \le \delta, \\ \delta \,\text{sign}(\hat{y} - y) & \text{if } |y - \hat{y}| > \delta. \end{cases} \tag{4.20}$$

Chapter 5

誤差逆伝播法

5.1 設定: 記法と順伝播

L 層の順伝播ニューラルネットワークを考える。単一の例 $(\mathbf{x}, \mathbf{y})$ に対して、次のように定義する：

$$\mathbf{a}^{(0)} = \mathbf{x}. \tag{5.1}$$

各層 $\ell = 1, 2, \ldots, L$ について：

$$\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}, \tag{5.2}$$

$$\mathbf{a}^{(\ell)} = \sigma^{(\ell)}\left(\mathbf{z}^{(\ell)}\right), \tag{5.3}$$

ここで $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, $\mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}$、そして $\sigma^{(\ell)}$ は特記なき限り要素ごとに適用されるものとする。

予測を $\hat{\mathbf{y}} = \mathbf{a}^{(L)}$、損失を以下とする：

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \mathcal{L}(\mathbf{a}^{(L)}, \mathbf{y}). \tag{5.4}$$

誤差逆伝播法の目的は、すべての ℓ に対して勾配 $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}}$ および $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(\ell)}}$ を効率的に計算することである。

5.2 デルタ項

5.2.1 定義

層 ℓ における**デルタ**（誤差信号）を次のように定義する：

$$\boldsymbol{\delta}^{(\ell)} \;=\; \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(\ell)}} \in \mathbb{R}^{n_\ell}. \tag{5.5}$$

$\boldsymbol{\delta}^{(\ell)}$ が既知であれば、パラメータ勾配は単純な外積形式で求められる（セクション 5.3を参照）。

5.2.2 出力層のデルタ: 一般形

連鎖律により、

$$\boldsymbol{\delta}^{(L)} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(L)}} \odot \frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{z}^{(L)}} = \nabla_{\mathbf{a}^{(L)}}\mathcal{L} \;\odot\; \sigma^{(L)\prime}\left(\mathbf{z}^{(L)}\right), \tag{5.6}$$

ここで $\odot$ は要素ごとの乗算を表す。

5.2.3 隠れ層のデルタ

$\ell = L-1, L-2, \ldots, 1$ に対して、デルタを逆伝播させる：

$$\boldsymbol{\delta}^{(\ell)} = \left(\mathbf{W}^{(\ell+1)}\right)^{\top} \boldsymbol{\delta}^{(\ell+1)} \odot \sigma^{(\ell)\prime}\left(\mathbf{z}^{(\ell)}\right). \tag{5.7}$$

解釈：

- $\left(\mathbf{W}^{(\ell+1)}\right)^{\top} \boldsymbol{\delta}^{(\ell+1)}$ は誤差信号を層 ℓ にマッピングして戻す。
- $\sigma^{(\ell)\prime}$ を乗じることで、層 ℓ における非線形性を考慮する。

これが誤差逆伝播法の中心的な漸化式である。

5.3 重みとバイアスの勾配

5.3.1 単一の例

(5.2) より、各成分は次のようになる：

$$z_i^{(\ell)} = \sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)} a_j^{(\ell-1)} + b_i^{(\ell)}. \tag{5.8}$$

したがって、

$$\frac{\partial z_i^{(\ell)}}{\partial W_{ij}^{(\ell)}} = a_j^{(\ell-1)}, \quad \frac{\partial z_i^{(\ell)}}{\partial b_i^{(\ell)}} = 1, \quad \frac{\partial z_i^{(\ell)}}{\partial b_k^{(\ell)}} = 0 \ (k \neq i). \tag{5.9}$$

$\delta_i^{(\ell)} = \frac{\partial \mathcal{L}}{\partial z_i^{(\ell)}}$ を用いると、以下が得られる：

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial z_i^{(\ell)}} \frac{\partial z_i^{(\ell)}}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell)} \, a_j^{(\ell-1)}. \tag{5.10}$$

行列形式では：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}} = \boldsymbol{\delta}^{(\ell)} \left(\mathbf{a}^{(\ell-1)}\right)^{\top}. \tag{5.11}$$

同様に、バイアスに対しては：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(\ell)}} = \boldsymbol{\delta}^{(\ell)}. \tag{5.12}$$

これらはドラフトに既に現れているコンパクトな公式と一致する。

5.3.2 ミニバッチ（平均勾配）

サイズ m のミニバッチに対して、デルタ $\boldsymbol{\delta}^{(\ell),(i)}$ と活性化 $\mathbf{a}^{(\ell-1),(i)}$ を用いると：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\delta}^{(\ell),(i)} \left(\mathbf{a}^{(\ell-1),(i)}\right)^{\top}, \tag{5.13}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(\ell)}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\delta}^{(\ell),(i)}. \tag{5.14}$$

（実装上の注意：ベクトル化されたコードでは、例を行列として積み重ねるため、これらは行列乗算となる。）

5.3.3 ミニバッチ行列逆伝播（ベクトル化されたデルタ）

デルタの再帰式を、行列順伝播（Section 3.1.4）および第6章の数値ベクトル化と整合するように、完全にベクトル化されたミニバッチ形式で書き直す。

ミニバッチ記法。 ミニバッチサイズを m とし、活性化を列として積み重ねる：

$$A^{(\ell)} = \left[a^{(\ell),(1)}, \dots, a^{(\ell),(m)}\right] \in \mathbb{R}^{n_\ell \times m}, \qquad Z^{(\ell)} = \left[z^{(\ell),(1)}, \dots, z^{(\ell),(m)}\right] \in \mathbb{R}^{n_\ell \times m}. \tag{5.15}$$

ベクトル化された順伝播は以下の通り：

$$Z^{(\ell)} = W^{(\ell)} A^{(\ell-1)} + b^{(\ell)} \mathbf{1}^\top, \qquad A^{(\ell)} = \sigma^{(\ell)}\big(Z^{(\ell)}\big), \tag{5.16}$$

ここで $\mathbf{1} \in \mathbb{R}^m$ は全要素が1のベクトルであり、$\sigma^{(\ell)}$ は要素ごとに適用される。

ベクトル化されたデルタ。 ミニバッチデルタ行列を定義する：

$$\Delta^{(\ell)} \;=\; \frac{\partial \mathcal{L}_{\text{batch}}}{\partial Z^{(\ell)}} \in \mathbb{R}^{n_\ell \times m}, \tag{5.17}$$

ここで $\mathcal{L}_{\text{batch}}$ はミニバッチ上の平均損失を表す。

出力層のデルタ（行列形式）。 一般性を失わず、出力層 $\ell = L$ に対して：

$$\Delta^{(L)} = \left(\frac{\partial \mathcal{L}_{\text{batch}}}{\partial A^{(L)}}\right) \odot \sigma^{(L)\prime}\big(Z^{(L)}\big), \tag{5.18}$$

ここで $\odot$ は要素ごとの乗算を表す。

隠れ層のデルタ再帰（行列形式）。 $\ell = L-1, \dots, 1$ に対して、隠れ層のデルタはベクトル化された漸化式を満たす：

$$\Delta^{(\ell)} = \big(W^{(\ell+1)}\big)^\top \Delta^{(\ell+1)} \odot \sigma^{(\ell)\prime}\big(Z^{(\ell)}\big). \tag{5.19}$$

これは単一サンプルの式 (5.7) を m 列すべてに並列に適用したものとまったく同じである。

ベクトル化されたデルタからの勾配。 (5.16) と $\Delta^{(\ell)}$ の定義を用いて、パラメータの勾配は次のようになる：

$$\frac{\partial \mathcal{L}_{\text{batch}}}{\partial W^{(\ell)}} = \frac{1}{m} \Delta^{(\ell)} \left(A^{(\ell-1)}\right)^\top \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, \tag{5.20}$$

$$\frac{\partial \mathcal{L}_{\text{batch}}}{\partial b^{(\ell)}} = \frac{1}{m} \Delta^{(\ell)} \mathbf{1} \in \mathbb{R}^{n_\ell}. \tag{5.21}$$

バイアスの勾配こうなるのは、$b^{(\ell)}\mathbf{1}^\top$ が $b^{(\ell)}$ を m 列に複製することによる。

整合性チェック（形状）。

$$\big(W^{(\ell+1)}\big)^\top \Delta^{(\ell+1)} \in \mathbb{R}^{n_\ell \times m}, \qquad \Delta^{(\ell)} \left(A^{(\ell-1)}\right)^\top \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, \tag{5.22}$$

したがって、すべての行列乗算は次元的に整合している。

2つの一般的な出力の簡略化（ミニバッチ）。 セクション5.4で導出された標準的なペアの選択について：

- **シグモイド + BCE（二値）**：もし $A^{(L)} = \hat{Y} \in \mathbb{R}^{1\times m}$ かつ $Y \in \mathbb{R}^{1\times m}$ ならば、
$$\Delta^{(L)} = \hat{Y} - Y. \tag{5.23}$$
- **ソフトマックス + CCE（多クラス）**：もし $A^{(L)} = P \in \mathbb{R}^{K\times m}$ かつワンホットラベル $Y \in \mathbb{R}^{K\times m}$ ならば、
$$\Delta^{(L)} = P - Y. \tag{5.24}$$

これらは (5.18) と (5.21) の列ごとの拡張である。

5.4 2つの古典的な「相殺」

逆伝播は、一般的な出力層の選択に対して特に単純になる。

5.4.1 シグモイド + 二値交差エントロピー

単一の出力ロジット z とシグモイド活性化を仮定する：

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}, \qquad \sigma'(z) = \hat{y}(1-\hat{y}). \tag{5.25}$$

二値交差エントロピー損失：

$$\mathcal{L}(\hat{y}, y) = -\left[y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right]. \tag{5.26}$$

まず、

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})}. \tag{5.27}$$

次に連鎖律により：

$$\delta^{(L)} = \frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \cdot \hat{y}(1-\hat{y}) = \hat{y} - y. \tag{5.28}$$

したがって、出力デルタは単純に「予測 - ターゲット」となる。

5.4.2 ソフトマックス + カテゴリカル交差エントロピー

ロジットを $\mathbf{z} \in \mathbb{R}^K$、ソフトマックス確率を次のようにする：

$$p_k = \text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}. \tag{5.29}$$

ワンホットラベルベクトル $\mathbf{y} \in \{0,1\}^K$、カテゴリカル交差エントロピー：

$$\mathcal{L}(\mathbf{p}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log p_k. \tag{5.30}$$

重要な結果（ソフトマックスのヤコビアンから導出可能）は以下の通り：

$$\boldsymbol{\delta}^{(L)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} = \mathbf{p} - \mathbf{y}. \tag{5.31}$$

ここでも、デルタは「予測確率 - 真の確率」である。

5.5 勾配消失（なぜ深層学習は難しいか）

式 (5.7) は、初期層のデルタが多くの因子の積であることを示している。深いネットワークでは、模式的に以下のようになる：

$$\boldsymbol{\delta}^{(1)} \approx \left(\prod_{\ell=2}^{L} \left(\mathbf{W}^{(\ell)}\right)^{\top} \operatorname{Diag}\left(\sigma^{(\ell)\prime}(\mathbf{z}^{(\ell)})\right) \right) \boldsymbol{\delta}^{(L)}. \tag{5.32}$$

もし $\sigma^{(\ell)}$ がシグモイドなら、すべての z に対して $\sigma'(z) \le 1/4$ である。 $1/4$ 以下の数を多数かけ合わせると、勾配の大きさはゼロに近づき、初期層の学習が遅くなる。

5.5.1 緩和策（数学的観点）

- ReLU 型の活性化関数: $z > 0$ で $\mathrm{ReLU}'(z) = 1$ となり、偏在する小さな係数を回避する。
- 慎重な初期化: 活性化と勾配を適切なスケールに保つ。
- 正規化（例：バッチ正規化）とスキップ接続: 勾配の流れを改善する。

これらのアイデアが、現代の深層学習アーキテクチャ設計の多くを動機付けている。

5.6 アルゴリズムの要約（1回の反復）

1つのミニバッチに対して：

1. 順伝播: $\ell = 1, \dots, L$ について (5.2)–(5.3) を用いて $(\mathbf{z}^{(\ell)}, \mathbf{a}^{(\ell)})$ を計算する。
2. 出力デルタ: (5.6)（適用可能な場合は簡略化された形式 (5.28), (5.31)）を用いて $\boldsymbol{\delta}^{(L)}$ を計算する。
3. 逆方向への再帰: $\ell = L-1, \dots, 1$ について (5.7) を用いて $\boldsymbol{\delta}^{(\ell)}$ を計算する。
4. 勾配: (5.13)–(5.14) を用いて $\partial \mathcal{L}/\partial \mathbf{W}^{(\ell)}$ と $\partial \mathcal{L}/\partial \mathbf{b}^{(\ell)}$ を計算する。
5. パラメータの更新: オプティマイザ（SGD、モメンタム、Adam など）を用いて更新する。

これで1回の学習ステップが完了する。

Chapter 6

具体的な数値例: ゼロから作るネットワーク

この章では、二値分類のための小さな順伝播ニューラルネットワークを構築し、順伝播、損失計算、および誤差逆伝播を数値的かつ記号的に導出する。目標は、すべての量（z, a, δ, 勾配）を明示的に確認することである。

6.1 問題設定

$m = 4$ サンプル、入力次元 $d = 2$、二値ラベル $y \in \{0, 1\}$ のトイデータセットを考える：

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{4}, \qquad \mathbf{x}^{(i)} \in \mathbb{R}^2,\ y^{(i)} \in \{0, 1\}. \tag{6.1}$$

具体的には：

i	$\mathbf{x}^{(i)}$	$y^{(i)}$
1	$[0.5,\ 0.2]^\top$	0
2	$[0.9,\ 0.8]^\top$	1
3	$[0.1,\ 0.3]^\top$	0
4	$[0.8,\ 0.9]^\top$	1

(6.2)

$2 \to 3 \to 1$ ネットワークを使用する：

- 隠れ層: $n_1 = 3$ ニューロン、ReLU。
- 出力層: $n_2 = 1$ ニューロン、シグモイドで $\hat{y} \in (0, 1)$ を出力。

6.2 パラメータの初期化

第 1 層のパラメータ：

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{bmatrix} \in \mathbb{R}^{3\times 2}, \qquad \mathbf{b}^{(1)} = \begin{bmatrix} 0.01 \\ 0.02 \\ 0.03 \end{bmatrix} \in \mathbb{R}^3. \tag{6.3}$$

第 2 層のパラメータ：

$$\mathbf{W}^{(2)} = \begin{bmatrix} 0.7 & 0.8 & 0.9 \end{bmatrix} \in \mathbb{R}^{1\times 3}, \qquad b^{(2)} = 0.1. \tag{6.4}$$

6.3 順伝播: 一般形

各サンプル $\mathbf{x}$ に対して：

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \tag{6.5}$$

$$\mathbf{a}^{(1)} = \mathrm{ReLU}(\mathbf{z}^{(1)}), \tag{6.6}$$

$$z^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + b^{(2)}, \tag{6.7}$$

$$\hat{y} = \sigma(z^{(2)}) = \frac{1}{1 + e^{-z^{(2)}}}. \tag{6.8}$$

二値交差エントロピー（サンプルごとの損失）を使用する：

$$\mathcal{L}^{(i)} \;=\; -\Big(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\Big). \tag{6.9}$$

バッチ損失：

$$\mathcal{L}_{\text{batch}} = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}. \tag{6.10}$$

6.4 順伝播: サンプル 1 (完全展開)

$\mathbf{x}^{(1)} = [0.5, 0.2]^\top$, $y^{(1)} = 0$ とする。

6.4.1 第 1 層の事前活性化

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} + \mathbf{b}^{(1)}. \tag{6.11}$$

要素ごとに計算：

$$z_1^{(1)} = 0.1 \cdot 0.5 + 0.2 \cdot 0.2 + 0.01 = 0.05 + 0.04 + 0.01 = 0.10, \tag{6.12}$$

$$z_2^{(1)} = 0.3 \cdot 0.5 + 0.4 \cdot 0.2 + 0.02 = 0.15 + 0.08 + 0.02 = 0.25, \tag{6.13}$$

$$z_3^{(1)} = 0.5 \cdot 0.5 + 0.6 \cdot 0.2 + 0.03 = 0.25 + 0.12 + 0.03 = 0.40. \tag{6.14}$$

したがって

$$\mathbf{z}^{(1)} = \begin{bmatrix} 0.10 \\ 0.25 \\ 0.40 \end{bmatrix}. \tag{6.15}$$

6.4.2 第 1 層の活性化 (ReLU)

$$\mathbf{a}^{(1)} = \mathrm{ReLU}(\mathbf{z}^{(1)}) = \begin{bmatrix} \max(0, 0.10) \\ \max(0, 0.25) \\ \max(0, 0.40) \end{bmatrix} = \begin{bmatrix} 0.10 \\ 0.25 \\ 0.40 \end{bmatrix}. \tag{6.16}$$

6.4.3 第 2 層の事前活性化

$$z^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + b^{(2)} \tag{6.17}$$

$$= 0.7 \cdot 0.10 + 0.8 \cdot 0.25 + 0.9 \cdot 0.40 + 0.1 \tag{6.18}$$

$$= 0.07 + 0.20 + 0.36 + 0.10 = 0.73. \tag{6.19}$$

6.4.4 出力 (シグモイド)

$$\hat{y}^{(1)} = \sigma(0.73) = \frac{1}{1+e^{-0.73}}. \tag{6.20}$$

$e^{-0.73} \approx 0.4819$ を用いて：

$$\hat{y}^{(1)} \approx \frac{1}{1+0.4819} = \frac{1}{1.4819} \approx 0.6748. \tag{6.21}$$

6.4.5 サンプル 1 の損失

$y^{(1)} = 0$ なので、損失は (6.9) から次のように単純化される：

$$\mathcal{L}^{(1)} = -\log(1-\hat{y}^{(1)}) = -\log(1-0.6748) = -\log(0.3252). \tag{6.22}$$

数値的には、

$$\mathcal{L}^{(1)} \approx 1.1223. \tag{6.23}$$

6.5 順伝播: サンプル 2 (詳細)

$\mathbf{x}^{(2)} = [0.9, 0.8]^\top$, $y^{(2)} = 1$ とする。

6.5.1 第 1 層

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x}^{(2)} + \mathbf{b}^{(1)}. \tag{6.24}$$

要素ごとに：

$$z_1^{(1)} = 0.1 \cdot 0.9 + 0.2 \cdot 0.8 + 0.01 = 0.09 + 0.16 + 0.01 = 0.26, \tag{6.25}$$

$$z_2^{(1)} = 0.3 \cdot 0.9 + 0.4 \cdot 0.8 + 0.02 = 0.27 + 0.32 + 0.02 = 0.61, \tag{6.26}$$

$$z_3^{(1)} = 0.5 \cdot 0.9 + 0.6 \cdot 0.8 + 0.03 = 0.45 + 0.48 + 0.03 = 0.96. \tag{6.27}$$

すべての要素が正なので、

$$\mathbf{a}^{(1)} = \mathrm{ReLU}(\mathbf{z}^{(1)}) = \begin{bmatrix} 0.26 \\ 0.61 \\ 0.96 \end{bmatrix}. \tag{6.28}$$

6.5.2 第 2 層と出力

$$z^{(2)} = 0.7 \cdot 0.26 + 0.8 \cdot 0.61 + 0.9 \cdot 0.96 + 0.1 \tag{6.29}$$

$$= 0.182 + 0.488 + 0.864 + 0.1 = 1.634, \tag{6.30}$$

$$\hat{y}^{(2)} = \sigma(1.634) = \frac{1}{1+e^{-1.634}} \approx 0.8367. \tag{6.31}$$

損失（$y^{(2)} = 1$ なので）：

$$\mathcal{L}^{(2)} = -\log(\hat{y}^{(2)}) \approx -\log(0.8367) \approx 0.1779. \tag{6.32}$$

6.6 バッチ損失

残りのサンプルの損失が以下のようであると仮定する（現在のドラフトと同様）：

$$\mathcal{L}^{(3)} \approx 0.9502, \qquad \mathcal{L}^{(4)} \approx 0.2148. \tag{6.33}$$

するとバッチ損失は

$$\mathcal{L}_{\text{batch}} = \frac{1}{4}\left(\mathcal{L}^{(1)} + \mathcal{L}^{(2)} + \mathcal{L}^{(3)} + \mathcal{L}^{(4)}\right) \tag{6.34}$$

$$= \frac{1}{4}(1.1223 + 0.1779 + 0.9502 + 0.2148) \tag{6.35}$$

$$= \frac{2.4652}{4} \approx 0.6163. \tag{6.36}$$

6.7 誤差逆伝播: 一般形

出力デルタ（シグモイド + 二値交差エントロピー）を次のように定義する：

$$\delta^{(2)} \;=\; \frac{\partial \mathcal{L}}{\partial z^{(2)}} \;=\; \hat{y} - y. \tag{6.37}$$

すると

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}} = \delta^{(2)} (\mathbf{a}^{(1)})^{\top}, \tag{6.38}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(2)}} = \delta^{(2)}. \tag{6.39}$$

隠れ層（ReLU）に対して：

$$\boldsymbol{\delta}^{(1)} \;=\; \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(1)}} \;=\; (\mathbf{W}^{(2)})^{\top} \delta^{(2)} \;\odot\; \mathrm{ReLU}'(\mathbf{z}^{(1)}), \tag{6.40}$$

かつ

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}} = \boldsymbol{\delta}^{(1)} (\mathbf{x})^{\top}, \tag{6.41}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}} = \boldsymbol{\delta}^{(1)}. \tag{6.42}$$

ここで

$$\mathrm{ReLU}'(z) = \begin{cases} 1 & (z > 0), \\ 0 & (z \le 0). \end{cases} \tag{6.43}$$

6.8 誤差逆伝播: サンプル 1 (完全展開)

サンプル 1 について、$y^{(1)} = 0$ かつ $\hat{y}^{(1)} \approx 0.6748$ である。

6.8.1 出力デルタ

(6.37) より、

$$\delta_1^{(2)} = \hat{y}^{(1)} - y^{(1)} = 0.6748 - 0 = 0.6748. \tag{6.44}$$

6.8.2 第 2 層の勾配

$$\frac{\partial \mathcal{L}^{(1)}}{\partial \mathbf{W}^{(2)}} = \delta_1^{(2)} (\mathbf{a}^{(1)})^\top \tag{6.45}$$

$$= 0.6748 \cdot [0.10,\, 0.25,\, 0.40] \tag{6.46}$$

$$= [0.06748,\, 0.16870,\, 0.26992], \tag{6.47}$$

および

$$\frac{\partial \mathcal{L}^{(1)}}{\partial b^{(2)}} = \delta_1^{(2)} = 0.6748. \tag{6.48}$$

6.8.3 隠れ層のデルタ

(6.40) を用いると：

$$(\mathbf{W}^{(2)})^\top \delta_1^{(2)} = \begin{bmatrix} 0.7 \\ 0.8 \\ 0.9 \end{bmatrix} 0.6748 = \begin{bmatrix} 0.47236 \\ 0.53984 \\ 0.60732 \end{bmatrix}. \tag{6.49}$$

$\mathbf{z}^{(1)} = [0.10, 0.25, 0.40]^\top$ は正なので、$\mathrm{ReLU}'(\mathbf{z}^{(1)}) = [1, 1, 1]^\top$ となり、

$$\boldsymbol{\delta}_1^{(1)} = \begin{bmatrix} 0.47236 \\ 0.53984 \\ 0.60732 \end{bmatrix}. \tag{6.50}$$

6.8.4 第 1 層の勾配

$$\frac{\partial \mathcal{L}^{(1)}}{\partial \mathbf{W}^{(1)}} = \boldsymbol{\delta}_1^{(1)} (\mathbf{x}^{(1)})^\top \tag{6.51}$$

$$= \begin{bmatrix} 0.47236 \\ 0.53984 \\ 0.60732 \end{bmatrix} \begin{bmatrix} 0.5 & 0.2 \end{bmatrix} \tag{6.52}$$

$$= \begin{bmatrix} 0.23618 & 0.09447 \\ 0.26992 & 0.10797 \\ 0.30366 & 0.12146 \end{bmatrix}, \tag{6.53}$$

および

$$\frac{\partial \mathcal{L}^{(1)}}{\partial \mathbf{b}^{(1)}} = \boldsymbol{\delta}_1^{(1)} = \begin{bmatrix} 0.47236 \\ 0.53984 \\ 0.60732 \end{bmatrix}. \tag{6.54}$$

6.9 ミニバッチ勾配（平均化）

サイズ $m = 4$ のミニバッチに対して、サンプルごとの勾配を $g^{(i)}$ とする。例えば、第 2 層の重み：

$$\frac{\partial \mathcal{L}_{\text{batch}}}{\partial \mathbf{W}^{(2)}} = \frac{1}{4} \sum_{i=1}^{4} \frac{\partial \mathcal{L}^{(i)}}{\partial \mathbf{W}^{(2)}}. \tag{6.55}$$

現在のドラフトでは、平均勾配は次のように要約されている：

$$\frac{\partial \mathcal{L}_{\text{batch}}}{\partial \mathbf{W}^{(2)}} \approx [0.0289,\, 0.0425,\, 0.0568]. \tag{6.56}$$

6.10 パラメータの更新 (SGD)

学習率 $\eta = 0.1$ を用いると、勾配降下法の更新は以下のようになる：

$$\theta_{\text{new}} = \theta - \eta \nabla_\theta \mathcal{L}_{\text{batch}}. \tag{6.57}$$

6.10.1 第 2 層の更新

$$\begin{aligned} \mathbf{W}^{(2)}_{\text{new}} &= \mathbf{W}^{(2)} - 0.1 \frac{\partial \mathcal{L}_{\text{batch}}}{\partial \mathbf{W}^{(2)}} && (6.58)\\ &= [0.7,\, 0.8,\, 0.9] - 0.1[0.0289,\, 0.0425,\, 0.0568] && (6.59)\\ &= [0.6971,\, 0.7958,\, 0.8943]. && (6.60) \end{aligned}$$

バイアスの更新も同様に

$$b^{(2)}_{\text{new}} = b^{(2)} - 0.1 \frac{\partial \mathcal{L}_{\text{batch}}}{\partial b^{(2)}}. \tag{6.61}$$

（現在のドラフトでは $b^{(2)}_{\text{new}} \approx 0.0831$ となる数値が報告されている。）

6.10.2 第 1 層の更新

同様に、

$$\mathbf{W}^{(1)}_{\text{new}} = \mathbf{W}^{(1)} - 0.1 \frac{\partial \mathcal{L}_{\text{batch}}}{\partial \mathbf{W}^{(1)}}, \qquad \mathbf{b}^{(1)}_{\text{new}} = \mathbf{b}^{(1)} - 0.1 \frac{\partial \mathcal{L}_{\text{batch}}}{\partial \mathbf{b}^{(1)}}. \tag{6.62}$$

現在のドラフトでは、更新されたパラメータは数値的に次のように要約されている：

$$\mathbf{W}^{(1)}_{\text{new}} \approx \begin{bmatrix} 0.0933 & 0.1974 \\ 0.2937 & 0.3981 \\ 0.4931 & 0.5971 \end{bmatrix}, \qquad \mathbf{b}^{(1)}_{\text{new}} \approx \begin{bmatrix} -0.0032 \\ 0.0094 \\ 0.0142 \end{bmatrix}. \tag{6.63}$$

6.11 2回目の反復 (順伝播チェック)

更新によって損失が減少することを確認するために、更新されたパラメータを使用してサンプル 1 の順伝播を再計算する。現在のドラフトでは（サンプル 1 について）：

$$\mathbf{z}^{(1)}_{\text{new}} = \begin{bmatrix} 0.0872 \\ 0.2388 \\ 0.3834 \end{bmatrix}, \qquad \mathbf{a}^{(1)}_{\text{new}} = \begin{bmatrix} 0.0872 \\ 0.2388 \\ 0.3834 \end{bmatrix}, \tag{6.64}$$

$$z^{(2)}_{\text{new}} \approx 0.6772, \qquad \hat{y}^{(1)}_{\text{new}} = \sigma(0.6772) \approx 0.6636, \tag{6.65}$$

したがって、新しい損失は

$$\mathcal{L}^{(1)}_{\text{new}} = -\log(1 - \hat{y}^{(1)}_{\text{new}}) = -\log(1 - 0.6636) = -\log(0.3364) \approx 1.0898. \tag{6.66}$$

損失は ≈ 1.1223 から ≈ 1.0898 に減少し、勾配降下が目的関数を改善していることと整合する。

Chapter 7

高度な数値デモンストレーション

この章では、第6章の具体的なネットワークを拡張し、一般的な最適化および正則化手法が更新や活性化をどのように修正するかを、明確な数値で示す。

7.1　最適化のダイナミクス

7.1.1　学習率の影響

パラメータベクトル（または平坦化された重み行列）θ と勾配推定 $g = \nabla_\theta \mathcal{L}(\theta)$ を考える。単一の勾配降下ステップは以下の通り：

$$\theta_{\text{new}} = \theta - \eta g, \tag{7.1}$$

ここで $\eta > 0$ は学習率である。

具体的な例（第 2 層の重み）。　第6章のバッチ勾配推定を用いて：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}} \approx [0.0289,\ 0.0425,\ 0.0568], \tag{7.2}$$

更新の大きさは η に線形に依存する。

- $\eta = 0.1$ の場合：

$$\Delta \mathbf{W}^{(2)} = -0.1\,[0.0289, 0.0425, 0.0568] = [-0.00289, -0.00425, -0.00568]. \tag{7.1}$$

- $\eta = 0.5$（より積極的）の場合：

$$\Delta \mathbf{W}^{(2)} = -0.5\,[0.0289, 0.0425, 0.0568] = [-0.01445, -0.02125, -0.02840]. \tag{7.2}$$

大きな η は速い動きをもたらすが、最小値を通り過ぎたり発散したりするリスクが増大する。

7.1.2　損失曲線（例示）

反復 t 後のバッチ損失を $\mathcal{L}_t$ と表す。単調減少の例（以前のドラフトに見られたもの）は以下の通り：

反復 t	$\mathcal{L}_t$
0	0.6160
1	0.5847
5	0.5172

(7.3)

7.2 正則化の例: L2

7.2.1 定義

L2 正則化（重み減衰）は、重みの大きさに対するペナルティで損失を拡張する：

$$\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}(\theta) + \lambda \sum_{\ell} \|\mathbf{W}^{(\ell)}\|_F^2, \tag{7.4}$$

ここで $\lambda > 0$ はペナルティの強さを制御し、

$$\|\mathbf{W}\|_F^2 = \sum_i \sum_j W_{ij}^2 \tag{7.5}$$

はフロベニウスノルムの二乗である。

7.2.2 具体的な計算

第6章の重みを用いる：

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{bmatrix}, \qquad \mathbf{W}^{(2)} = [0.7, 0.8, 0.9]. \tag{7.6}$$

フロベニウスノルムの二乗を計算する：

$$\|\mathbf{W}^{(1)}\|_F^2 = 0.1^2 + 0.2^2 + 0.3^2 + 0.4^2 + 0.5^2 + 0.6^2 = 0.01 + 0.04 + 0.09 + 0.16 + 0.25 + 0.36 = 0.91, \tag{7.4}$$

$$\|\mathbf{W}^{(2)}\|_F^2 = 0.7^2 + 0.8^2 + 0.9^2 = 0.49 + 0.64 + 0.81 = 1.94. \tag{7.5}$$

もし $\lambda = 0.01$ ならば、総ペナルティ（重みのみ）は

$$\lambda \left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2 \right) = 0.01(0.91 + 1.94) = 0.01(2.85) = 0.0285. \tag{7.6}$$

したがって、もし $\mathcal{L} = 0.616$ ならば

$$\mathcal{L}_{\text{reg}} = 0.616 + 0.0285 = 0.6445. \tag{7.7}$$

7.2.3 勾配への影響（重み減衰の観点）

微分すると、

$$\nabla_{\mathbf{W}^{(\ell)}} \mathcal{L}_{\text{reg}} = \nabla_{\mathbf{W}^{(\ell)}} \mathcal{L} + 2\lambda \mathbf{W}^{(\ell)}. \tag{7.7}$$

したがって更新は次のようになる：

$$\mathbf{W}^{(\ell)} \leftarrow \mathbf{W}^{(\ell)} - \eta \left(\nabla_{\mathbf{W}^{(\ell)}} \mathcal{L} + 2\lambda \mathbf{W}^{(\ell)} \right), \tag{7.8}$$

これは各ステップで重みをゼロに向かって明示的に引き寄せる。

7.3 モメンタム最適化

7.3.1 更新ルール

モメンタムは速度ベクトル $\mathbf{v}_t$ を維持する：

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \nabla_\theta \mathcal{L}(\theta_t), \tag{7.9}$$

$$\theta_{t+1} = \theta_t - \eta\, \mathbf{v}_{t+1}, \tag{7.10}$$

ここで $\beta \in (0,1)$ はモメンタム係数である（多くの場合 0.9）。

7.3.2 2ステップの数値例（第 2 層の重み）

$\beta = 0.9$, $\eta = 0.1$ とし、$\mathbf{v}_0 = \mathbf{0}$ で初期化する。勾配 $g_1 = [0.0289, 0.0425, 0.0568]$ を使用する。

反復 1.

$$\mathbf{v}_1 = 0.9\mathbf{0} + g_1 = [0.0289, 0.0425, 0.0568], \tag{7.8}$$

$$\begin{aligned}\mathbf{W}_1^{(2)} &= \mathbf{W}_0^{(2)} - 0.1\,\mathbf{v}_1 = [0.7, 0.8, 0.9] - 0.1[0.0289, 0.0425, 0.0568] \\ &= [0.6971, 0.7958, 0.8943]. \end{aligned} \tag{7.10}$$

反復 2. 新しい勾配が $g_2 = [0.0215, 0.0318, 0.0425]$ であるとする。

$$\begin{aligned}\mathbf{v}_2 &= 0.9\mathbf{v}_1 + g_2 = 0.9[0.0289, 0.0425, 0.0568] + [0.0215, 0.0318, 0.0425] \\ &= [0.0260, 0.0383, 0.0511] + [0.0215, 0.0318, 0.0425] = [0.0475, 0.0701, 0.0936], \end{aligned} \tag{7.12}$$

$$\begin{aligned}\mathbf{W}_2^{(2)} &= \mathbf{W}_1^{(2)} - 0.1\mathbf{v}_2 = [0.6971, 0.7958, 0.8943] - 0.1[0.0475, 0.0701, 0.0936] \\ &= [0.6919, 0.7890, 0.8758]. \end{aligned} \tag{7.13}$$

モメンタムは一貫した勾配方向を蓄積し、多くの場合収束を加速させる。

7.4 バッチ正規化（数値計算）

7.4.1 定義

サイズ m_B のミニバッチにわたるニューロン j の事前活性化 $z_j^{(i)}$ が与えられたとき、バッチ正規化は次を計算する：

$$\mu_{B,j} = \frac{1}{m_B} \sum_{i=1}^{m_B} z_j^{(i)}, \tag{7.11}$$

$$\sigma_{B,j}^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} \left(z_j^{(i)} - \mu_{B,j} \right)^2, \tag{7.12}$$

$$\hat{z}_j^{(i)} = \frac{z_j^{(i)} - \mu_{B,j}}{\sqrt{\sigma_{B,j}^2 + \epsilon}}, \tag{7.13}$$

$$y_j^{(i)} = \gamma_j \hat{z}_j^{(i)} + \beta_j, \tag{7.14}$$

ここで γ_j, β_j は学習可能なパラメータ、$\epsilon > 0$ は数値安定性を確保するための値である。

7.4.2 具体的な計算（1つのニューロン）

ニューロン $j = 1$ に対する4つの事前活性化の仮想的なバッチを考える：

$$z_1^{(1)} = 0.10, \quad z_1^{(2)} = 0.26, \quad z_1^{(3)} = 0.08, \quad z_1^{(4)} = 0.22. \tag{7.15}$$

平均：

$$\mu_{B,1} = \frac{0.10 + 0.26 + 0.08 + 0.22}{4} = \frac{0.66}{4} = 0.165. \tag{7.16}$$

分散：

$$\begin{aligned}\sigma_{B,1}^2 &= \frac{1}{4}\left[(0.10 - 0.165)^2 + (0.26 - 0.165)^2 + (0.08 - 0.165)^2 + (0.22 - 0.165)^2\right] \\ &= \frac{1}{4}\left[0.004225 + 0.009025 + 0.007225 + 0.003025\right] = \frac{0.02350}{4} = 0.005875. \end{aligned} \tag{7.14}$$

$\epsilon = 10^{-5}$ として、サンプル 1 を正規化する：

$$\hat{z}_1^{(1)} = \frac{0.10 - 0.165}{\sqrt{0.005875 + 10^{-5}}} = \frac{-0.065}{\sqrt{0.005885}} \approx \frac{-0.065}{0.0767} \approx -0.848. \tag{7.14'}$$

もし $\gamma_1 = 1.0$ かつ $\beta_1 = 0.0$ ならば、

$$y_1^{(1)} = \gamma_1 \hat{z}_1^{(1)} + \beta_1 = -0.848. \tag{7.15}$$

これは現在のドラフトにおけるバッチ正規化計算のスタイルを再現している。

7.5 ドロップアウト正則化（数値計算）

7.5.1 訓練時のドロップアウト

隠れ層の活性化ベクトルを $\mathbf{h} \in \mathbb{R}^n$ とする。ドロップアウトはマスク $\mathbf{m} \in \{0, 1\}^n$ を独立同分布 (i.i.d.) でサンプリングし：

$$m_k \sim \text{Bernoulli}(1 - p), \tag{7.17}$$

次を適用する：

$$\mathbf{h}_{\text{drop}} = \mathbf{h} \odot \mathbf{m}. \tag{7.18}$$

具体的な例（サンプル 1 の層 1 の活性化）。 $\mathbf{h}^{(1)} = [0.10, 0.25, 0.40]^\top$ とドロップアウト確率 $p = 0.5$ を用いて、サンプリングされたマスクが以下のようであるとする：

$$\mathbf{m} = [1, 0, 1]^\top. \tag{7.16}$$

すると

$$\mathbf{h}_{\text{drop}}^{(1)} = [0.10, 0.25, 0.40]^\top \odot [1, 0, 1]^\top = [0.10, 0, 0.40]^\top. \tag{7.17}$$

$\mathbf{W}^{(2)} = [0.7, 0.8, 0.9]$ と $b^{(2)} = 0.1$ に対して、新しい事前活性化は次のようになる：

$$z_{\text{drop}}^{(2)} = \mathbf{W}^{(2)}\mathbf{h}_{\text{drop}}^{(1)} + b^{(2)} = 0.7(0.10) + 0.8(0) + 0.9(0.40) + 0.1 = 0.07 + 0 + 0.36 + 0.1 = 0.53. \tag{7.18}$$

ドロップアウトは確率性を導入し、特徴の共適応（co-adaptation）を抑制する。

7.5.2 テスト時のスケーリング

一般的な規約では、推論時に活性化を $(1-p)$ 倍する（逆ドロップアウトを使用しない場合）：

$$\mathbf{h}_{\text{test}} = (1-p)\mathbf{h}. \tag{7.19}$$

これにより、訓練時とテスト時で活性化の期待値が一致する。

7.6 複数サンプルのベクトル化処理

ベクトル化はサンプルに対するループを行列演算に置き換える。 m 個の入力からなるミニバッチを行列として積み重ねる：

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(m)} \end{bmatrix} \in \mathbb{R}^{d \times m}. \tag{7.20}$$

重み $\mathbf{W} \in \mathbb{R}^{n \times d}$ とバイアス $\mathbf{b} \in \mathbb{R}^n$ を持つ層に対して、バッチ全体の事前活性化は次の通り：

$$\mathbf{Z} = \mathbf{W}\mathbf{X} + \mathbf{b}\mathbf{1}^\top, \tag{7.19}$$

ここで $\mathbf{1} \in \mathbb{R}^m$ は全要素が1のベクトルである。そして活性化を要素ごとに適用する：

$$\mathbf{A} = \sigma(\mathbf{Z}). \tag{7.21}$$

この単一の行列乗算で m 個すべてのサンプルを並列に計算でき、これが GPU がニューラルネットワークの学習を加速させる主な理由である。

Chapter 8

最適化手法

パラメータ θ に対する経験リスクを最小化することでニューラルネットワークを訓練する。データセットを $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ とし、次のように定義する：

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell\big(f_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}\big)\,. \tag{8.1}$$

最適化アルゴリズムは、通常 $\mathcal{L}(\theta_t)$ を減少させるような列 $\{\theta_t\}_{t\geq 0}$ を生成する。

8.1 確率的勾配降下法 (SGD)

8.1.1 フルバッチ勾配降下法

基本的な勾配降下法の反復は以下の通り：

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta_t), \tag{8.2}$$

ここで $\eta_t > 0$ は学習率である（時間に依存する場合もある）。

8.1.2 ミニバッチ SGD

深層学習では、N が大きい場合 $\nabla\mathcal{L}(\theta)$ の計算コストが高い。反復 t においてサンプリングされたサイズ B のミニバッチを $\mathcal{B}_t \subset \{1, \ldots, N\}$ とする。ミニバッチ目的関数を定義し：

$$\mathcal{L}_{\mathcal{B}_t}(\theta) = \frac{1}{B} \sum_{i \in \mathcal{B}_t} \ell\big(f_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}\big)\,, \tag{8.3}$$

その勾配推定量を次のようにする：

$$g_t := \nabla \mathcal{L}_{\mathcal{B}_t}(\theta_t). \tag{8.4}$$

SGD の更新は：

$$\theta_{t+1} = \theta_t - \eta_t g_t. \tag{8.5}$$

一様サンプリング（標準的な独立性の仮定の下で）では、

$$\mathbb{E}[g_t \mid \theta_t] = \nabla \mathcal{L}(\theta_t), \tag{8.6}$$

したがって g_t は完全な勾配の不偏推定量である。

8.1.3 学習率スケジュール（一般的な選択）

定数の学習率 $\eta_t = \eta$ は最適でないことが多い。典型的なスケジュールには以下がある：

- **ステップ減衰:** ある $\gamma \in (0,1)$ とステップ周期 T に対して $\eta_t = \eta_0 \gamma^{\lfloor t/T \rfloor}$。
- **多項式減衰:** $\alpha \in (0,1]$ に対して $\eta_t = \eta_0 (1+t)^{-\alpha}$。
- **コサイン減衰（実務で一般的）：** $\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\,(1 + \cos(\pi t/T))$。

8.1.4 基本的な降下不等式（滑らかな場合）

$\mathcal{L}$ が L-リプシッツ勾配を持つと仮定する：

$$\|\nabla \mathcal{L}(\theta) - \nabla \mathcal{L}(\theta')\|_2 \le L \|\theta - \theta'\|_2. \tag{8.7}$$

すると標準的な不等式より：

$$\mathcal{L}(\theta_{t+1}) \le \mathcal{L}(\theta_t) - \eta_t \langle \nabla \mathcal{L}(\theta_t), g_t \rangle + \frac{L \eta_t^2}{2} \|g_t\|_2^2. \tag{8.8}$$

決定論的なケース $g_t = \nabla \mathcal{L}(\theta_t)$ で η_t が十分に小さければ、損失は各ステップで減少する。

8.2 モメンタム

モメンタムは「速度」を蓄積することで、一貫した降下方向への SGD を加速する。

8.2.1 ヘビーボールモメンタム

$g_t = \nabla \mathcal{L}_{\mathcal{B}_t}(\theta_t)$ とする。速度 v_t を次のように定義する：

$$v_{t+1} = \beta v_t + g_t, \tag{8.9}$$
$$\theta_{t+1} = \theta_t - \eta_t v_{t+1}, \tag{8.10}$$

ここで $\beta \in [0,1)$ はモメンタム係数（多くの場合 0.9）である。
(8.9) を展開すると（$v_0 = 0$ として）：

$$v_t = \sum_{k=0}^{t-1} \beta^{t-1-k} g_k, \tag{8.11}$$

したがって v_t は過去の勾配の指数加重移動平均である。

8.2.2 Nesterov 加速勾配 (NAG)

人気のある変種で、先読みした点での勾配を評価する：

$$v_{t+1} = \beta v_t + \nabla \mathcal{L}_{\mathcal{B}_t}(\theta_t - \eta_t \beta v_t)\,, \tag{8.12}$$
$$\theta_{t+1} = \theta_t - \eta_t v_{t+1}. \tag{8.13}$$

これは (8.10) と比較して、狭い谷での振動を抑制できる場合がある。

8.3 適応的手法 (RMSProp, Adam, AdamW)

適応的オプティマイザは、勾配の二次モーメント統計量を用いて更新を座標ごとに再スケーリングする。

8.3.1 RMSProp (核となるアイデア)

勾配の二乗の指数移動平均を維持する：

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)(g_t \odot g_t), \tag{8.14}$$

そして更新する：

$$\theta_{t+1} = \theta_t - \eta_t \frac{g_t}{\sqrt{v_{t+1}} + \varepsilon}, \tag{8.15}$$

ここですべての操作は要素ごとに行われる。

8.3.2 Adam (Adaptive Moment Estimation)

Adam は 1 次および 2 次モーメント推定値を維持する：

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) g_t, \tag{8.16}$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)(g_t \odot g_t), \tag{8.17}$$

バイアス補正を行うと：

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1^{t+1}}, \tag{8.18}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2^{t+1}}. \tag{8.19}$$

Adam の更新は：

$$\theta_{t+1} = \theta_t - \eta_t \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \varepsilon}. \tag{8.20}$$

典型的なデフォルト値は $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ である。

8.3.3 AdamW (分離された重み減衰)

L2 正則化を用いる場合、しばしば $\nabla \mathcal{L}(\theta) + \lambda\theta$ と書く。しかし、適応的手法の場合、勾配内に $\lambda\theta$ を加える効果は「分離された」重み減衰とは異なる場合がある。

AdamW は重み減衰をパラメータに直接適用する：

$$\theta_{t+1} = (1-\eta_t\lambda)\theta_t - \eta_t \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \varepsilon}. \tag{8.21}$$

これは収縮項を適応的勾配ステップからきれいに分離する。

8.4 最適化としての正則化

正則化は、望ましいパラメータ構造（小さなノルム、スパース性、ロバスト性）を促進するために訓練目的関数を修正する。

8.4.1 L2 正則化（重み減衰）と MAP 解釈

重みに L2 ペナルティを加える：

$$\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}(\theta) + \frac{\lambda}{2}\sum_{\ell} \|\mathbf{W}^{(\ell)}\|_F^2. \tag{8.22}$$

すると

$$\nabla_{\mathbf{W}^{(\ell)}} \mathcal{L}_{\text{reg}} = \nabla_{\mathbf{W}^{(\ell)}} \mathcal{L} + \lambda \mathbf{W}^{(\ell)}. \tag{8.23}$$

SGD を用いると、更新は次のようになる：

$$\mathbf{W}^{(\ell)} \leftarrow (1 - \eta_t \lambda)\mathbf{W}^{(\ell)} - \eta_t \nabla_{\mathbf{W}^{(\ell)}} \mathcal{L}. \tag{8.24}$$

したがって重みは各ステップで係数 $(1 - \eta_t\lambda)$ だけ収縮する。

確率的視点（概略）：(8.22) の最小化は、重みに対する（独立した）ガウス事前分布の下での MAP 推定に対応する。なぜなら $-\log p(\mathbf{W})$ は $\|\mathbf{W}\|_F^2$ に比例するからである。

8.4.2 L1 正則化とスパース性

L1 正則化された目的関数：

$$\mathcal{L}_{\text{L1}}(\theta) = \mathcal{L}(\theta) + \lambda \sum_{\ell} \|\mathbf{W}^{(\ell)}\|_1 = \mathcal{L}(\theta) + \lambda \sum_{\ell}\sum_{i,j} |W_{ij}^{(\ell)}|. \tag{8.25}$$

準勾配は次を満たす：

$$\frac{\partial}{\partial W_{ij}^{(\ell)}} |W_{ij}^{(\ell)}| = \begin{cases} \operatorname{sign}(W_{ij}^{(\ell)}) & W_{ij}^{(\ell)} \neq 0, \\ s \in [-1, 1] & W_{ij}^{(\ell)} = 0. \end{cases} \tag{8.26}$$

L1 はスパースな解（多くのパラメータが厳密にゼロ）を生成する傾向がある。

8.4.3 ドロップアウト（逆ドロップアウト）

$\mathbf{a}^{(\ell)}$ を層 ℓ の活性化とする。マスク $\mathbf{m}^{(\ell)} \in \{0,1\}^{n_\ell}$ を i.i.d. でサンプリングする：

$$m_j^{(\ell)} \sim \text{Bernoulli}(q), \qquad q = 1 - p. \tag{8.27}$$

逆ドロップアウトでは、訓練時の活性化は

$$\tilde{\mathbf{a}}^{(\ell)} = \frac{1}{q}\big(\mathbf{m}^{(\ell)} \odot \mathbf{a}^{(\ell)}\big). \tag{8.28}$$

すると

$$\mathbb{E}[\tilde{\mathbf{a}}^{(\ell)} \mid \mathbf{a}^{(\ell)}] = \mathbf{a}^{(\ell)}, \tag{8.29}$$

したがって推論時に追加のスケーリングは不要である（非反転規約とは対照的）。

8.4.4 バッチ正規化 (BN)

ミニバッチ $i = 1, \dots, B$ にわたるニューロン j の事前活性化 $z_j^{(\ell),(i)}$ が与えられたとき、BN は次を計算する：

$$\mu_{B,j} = \frac{1}{B} \sum_{i=1}^{B} z_j^{(\ell),(i)}, \tag{8.30}$$

$$\sigma_{B,j}^2 = \frac{1}{B} \sum_{i=1}^{B} \left(z_j^{(\ell),(i)} - \mu_{B,j} \right)^2, \tag{8.31}$$

$$\hat{z}_j^{(\ell),(i)} = \frac{z_j^{(\ell),(i)} - \mu_{B,j}}{\sqrt{\sigma_{B,j}^2 + \varepsilon}}, \tag{8.32}$$

$$y_j^{(\ell),(i)} = \gamma_j \hat{z}_j^{(\ell),(i)} + \beta_j. \tag{8.33}$$

ここで γ_j, β_j は学習可能なパラメータである。

訓練対推論。 訓練中、BN はミニバッチ統計量 $\mu_{B,j}, \sigma_{B,j}^2$ を使用する。推論中、実装では通常、テスト時のバッチ構成への依存を避けるために（訓練中に推定された）移動平均を使用する。

なぜ役立つか。 BN は中間表現のスケールを安定させ、多くの場合、より大きな学習率を可能にし、深層ネットワークでの勾配の流れを改善する。また、バッチ統計量による穏やかな確率性も注入する。

8.5 安定化のトリック（実用）

8.5.1 勾配クリッピング

勾配爆発を防ぐために、グローバルノルムでクリップする：

$$g_t \leftarrow g_t \cdot \min\left(1, \frac{\tau}{\|g_t\|_2}\right), \tag{8.34}$$

閾値 $\tau > 0$ に対して。

8.5.2 ミニバッチサイズのトレードオフ

小さなバッチは勾配ノイズを増加させ（探索と汎化を改善することもある）、大きなバッチは分散を減少させるが、慎重な学習率のスケーリングとウォームアップが必要になる場合がある。

Chapter 9

解析と理論

この章では、テキストの理論的な部分を拡張する。目的は、完全に詳細な証明を与えることではなく、結果を正確に述べ、仮定を明らかにし、証明の概略と直感を与えることである。

9.1 関数近似

9.1.1 設定と記法

$K \subset \mathbb{R}^d$ をコンパクト集合（例：$K = [0,1]^d$）とする。 $\sigma : \mathbb{R} \to \mathbb{R}$ を活性化関数とする。幅 N の1隠れ層（2層）ネットワークは以下の通り：

$$f_N(x) = \sum_{j=1}^{N} a_j \, \sigma(\mathbf{w}_j^\top x + b_j), \qquad x \in \mathbb{R}^d, \tag{9.1}$$

ここで $(a_j, \mathbf{w}_j, b_j) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ はパラメータである。

9.1.2 普遍近似定理 (UAT)

定理 (普遍近似; 非公式). もし σ が多項式でない活性化関数（例：シグモイド、ReLU、tanh）であれば、任意の連続関数 $f \in C(K)$ と任意の $\varepsilon > 0$ に対して、以下を満たす N とパラメータが存在する：

$$\sup_{x \in K} |f_N(x) - f(x)| < \varepsilon. \tag{9.2}$$

これは近似ネットワークの存在を主張するものであり、それを見つける効率的な方法は与えない。

証明の概略（ハイレベル）。 関数解析を用いた一般的なルートの一つは：

- 集合 $\mathcal{F} = \{f_N : N \in \mathbb{N}\}$ とその $(C(K), \|\cdot\|_\infty)$ における閉包を考える。
- もし σ が非多項式ならば、$\mathcal{F}$ は $C(K)$ において稠密であることを示す（多くの場合、ハーン・バナッハまたはリースの表現定理の議論による：$\mathcal{F}$ を $C(K)$ から分離する任意の連続線形汎関数は符号付き測度 μ に対応し、すべての $(\mathbf{w}, b)$ に対して $\int \sigma(\mathbf{w}^\top x + b)\, d\mu(x) = 0$ ならば $\mu = 0$ であることを示す）。

- 稠密性は、任意の連続関数 f が K 上で一様に近似できることを意味する。

異なる証明も存在する（例：特定の活性化関数に対するストーン・ワイエルシュトラス型の議論）。

9.1.3 近似レート（なぜ UAT だけでは不十分か）

UAT は N が ε の関数としてどれだけ大きくなる必要があるかを教えてくれない。有用な理論的問いは次の通り：関数クラス $\mathcal{G}$（例：リプシッツまたはソボレフ関数）に対して、達成可能な最良の誤差は何か？

$$\inf_{f_N \in \mathcal{F}_N} \|f_N - f\|? \tag{9.3}$$

ここで $\mathcal{F}_N$ は形式 (9.1) の幅 N のネットワークのクラスである。
大まかに言えば：

- **滑らかな関数**は、N が増加するにつれてより速く近似できることが多い。
- **高次元** (d が大きい) は通常、最悪ケースのレートが遅くなる（「次元の呪い」）。ただし f が利用可能な構造（スパース性、合成形式、低い有効次元）を持つ場合は別である。

これが構造化されたターゲットと深いアーキテクチャを研究する動機となる。

9.1.4 なぜ深さが役立つか（合成構造）

深さ L のネットワークは合成関数クラスと見なすことができる：

$$f(x) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(x), \tag{9.4}$$

ここで各 $f^{(\ell)}$ はアフィン写像に非線形性を加えたものである。
ターゲット関数自体が合成構造（例：ネストされた低次元関数として自然に記述される）を持つ場合、深いネットワークは浅いネットワークよりもはるかに少ないパラメータでそれを表現/近似できる。

9.2 深さ vs 幅

9.2.1 表現力の尺度

2つの一般的な概念：

- **表現力 (Representation power):** ネットワークは与えられた関数を正確に表現できるか？
- **近似力 (Approximation power):** 誤差 ε 以内で近似できるか？

深さは、繰り返しの合成が異なるアフィン挙動の多くの「領域」を作り出すため、表現力を高める（特に ReLU のような区分線形活性化関数の場合）。

9.2.2 区分線形領域（ReLU の直感）

ReLU ネットワークは区分線形である。入力空間は、ネットワークがアフィン関数となる領域に分割される。深さはそのような領域の数を劇的に（適切な条件下では深さに対して指数関数的に）増加させることができる。

証明の概略（直感）。 各 ReLU は、ユニットがアクティブ/非アクティブ入れ替わる超平面境界を導入する。層を合成すると、新しい境界が前の層によってマッピングされ「折り畳まれ」、多くの線形領域が生成される。これにより、深さとともに領域数が組み合わせ爆発的に増加する。

9.2.3 分離結果（深さを必要とする関数）

定理 (非公式な分離). 多項式サイズ（パラメータ/ユニット）の深いネットワークで表現できるが、浅いネットワークに制限すると指数関数的な幅を必要とする関数の族が存在する。

例の直感: パリティ / 合成ブール関数。 パリティのような関数は強い階層構造を持つ：ペアに対する XOR を合成することで計算でき、これは自然に深さ $\log n$ の木を形成する。浅い表現では多くの入力パターンを「記憶」しなければならず、最良の場合でもサイズが指数関数的になる。

9.3 最適化の景観

9.3.1 非凸性と臨界点

ニューラルネットワークの訓練は通常次を解く：

$$\min_{\theta} \ \mathcal{L}(\theta), \tag{9.5}$$

ここで $\mathcal{L}$ は、合成と非線形性により θ に関して非凸である。

臨界点は次を満たす：

$$\nabla \mathcal{L}(\theta) = 0. \tag{9.6}$$

二次的な挙動はヘッセ行列によって特徴付けられる：

$$H(\theta) = \nabla^2 \mathcal{L}(\theta). \tag{9.7}$$

点は極小値 ($H \succeq 0$)、極大値 ($H \preceq 0$)、または鞍点（不定ヘッセ行列）であり得る。

9.3.2 高次元における鞍点（直感）

高次元のパラメータ空間では、鞍点は厳密な極大値よりも一般的である。 SGD のノイズとミニバッチ化は確率性を導入し、鞍点領域からの脱出を助けることができる。これは、一次手法が実用上うまくいく理由の一部を説明する。

9.3.3 高次元ヘッセ行列のBBP遷移

初期化時のヘッセ行列のスペクトルには、BBP (Baik-Ben Arous-Péché) 遷移と呼ばれる現象が現れる。信号対雑音比 (SNR) がある閾値 α_{BBP} を超えると、バルクスペクトルから外れ固有値が分離する。

$$\lambda_{\max} \to \begin{cases} (1+\sqrt{\gamma})^2 & \text{if SNR} \le \alpha_{\mathrm{BBP}} \\ \text{outlier} > (1+\sqrt{\gamma})^2 & \text{if SNR} > \alpha_{\mathrm{BBP}} \end{cases} \tag{9.8}$$

ここで γ はアスペクト比（パラメータ数/データ数）である。外れ固有値に対応する固有ベクトルは、真の信号方向と正の相関 $m = \langle \mathbf{v}_{\max}, \mathbf{w}^* \rangle > 0$ を持ち、学習の初期段階での信号捕捉を可能にする。

9.3.4 過剰パラメータ化と良性の景観（アイデア）

現代のネットワークはしばしば過剰パラメータ化されている（サンプルよりも多くのパラメータを持つ）。経験的に、この領域では訓練誤差をほぼゼロにできることが多い。一般的な理論的テーマは、十分な幅があれば、勾配ベースの手法は初期化の近傍において凸最適化のように振る舞うということである（線形化/NTK 型の議論に関連）。

証明の概略（アイデアのみ）。 初期化 θ_0 の周りでネットワークを線形化する：

$$f_\theta(x) \approx f_{\theta_0}(x) + \nabla_\theta f_{\theta_0}(x)^\top (\theta - \theta_0). \tag{9.9}$$

もしこの線形化が訓練中に正確であり続け、かつ誘導されるカーネル行列が良い条件数を持つならば、勾配降下法はカーネル回帰と同様に解析できる。

9.4 汎化

9.4.1 訓練対テスト

P を (X, Y) 上の（未知の）データ分布とする。母集団リスクを定義する：

$$R(\theta) = \mathbb{E}_{(X,Y)\sim P}\big[\ell(f_\theta(X), Y)\big], \tag{9.10}$$

および経験リスク（訓練損失）：

$$\hat{R}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i). \tag{9.11}$$

汎化は、特に訓練によって生成された $\hat{\theta}$ について、$\hat{R}_n(\theta)$ が $R(\theta)$ にどれだけ近いかを問う。

9.4.2 バイアス-バリアンス分解（二乗損失）

データ $\mathcal{D}$ から予測器 $\hat{f}$ を生成する学習アルゴリズムを用いた回帰の二乗損失について、点 x での期待テスト誤差を分解できる：

$$\mathbb{E}_{\mathcal{D}}\big[(\hat{f}(x) - f^\star(x))^2\big] = \underbrace{\big(\mathbb{E}_{\mathcal{D}}[\hat{f}(x)] - f^\star(x)\big)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\big[(\hat{f}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}(x)])^2\big]}_{\text{Variance}}. \tag{9.12}$$

（$Y = f^\star(X) + \varepsilon$（ノイズあり）の場合、さらに既約誤差項が現れる。）

解釈。 モデルの複雑さを増すと、通常バイアスは減少するがバリアンスは増加し、正則化や早期打ち切りの動機となる。

9.4.3 容量制御と一様収束（概略）

学習理論の結果の典型的な形式は、仮説クラス $\mathcal{H}$ 上での母集団リスクと経験リスクの差を有界にする：

$$\sup_{h \in \mathcal{H}} |R(h) - \hat{R}_n(h)|. \tag{9.13}$$

これは VC 次元やラデマッハ複雑度（特に有界な損失クラスに対して）などの複雑度尺度によって制御できる。

証明の概略（アウトライン）。 対称化を用いる：

$$\mathbb{E}\left[\sup_{h \in \mathcal{H}} (R(h) - \hat{R}_n(h))\right] \leq 2\,\mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i)\right], \tag{9.14}$$

ここで $\sigma_i \in \{-1, +1\}$ はラデマッハ変数である。その後、縮小不等式やノルム制約を用いて右辺を評価する。

9.4.4 暗黙的正則化（現象）

明示的なペナルティがなくても、勾配ベースの訓練は多くの補間解の中から特定の解を好む傾向がある。例えば、ゼロで初期化された線形回帰の勾配降下法は、データに適合するものの中で最小 ℓ_2 ノルム解に収束する。深層ネットワークはより複雑な形の暗黙的バイアスを示すが、テーマは同じである：最適化のダイナミクスが正則化項として機能する。

9.5 補間と二重降下

9.5.1 古典的な U 字型曲線

古典的統計学では、テスト誤差はモデルの複雑さとともに減少（バイアス減少）し、その後増加（バリアンス増加）し、U 字型の曲線を描くことが多い。

9.5.2 補間閾値

モデルが訓練セットを完全に適合できるほど表現力豊かになると、補間領域に達する：

$$\hat{R}_n(\hat{\theta}) \approx 0. \tag{9.15}$$

古典的には、完全適合は過学習を示唆するが、現代の深層学習はしばしばこの領域で動作する。

9.5.3 二重降下（経験的現象）

多くの現代の設定において、パラメータがさらに増加するとテスト誤差が再び減少し、「二重降下」曲線を生じることがある：

- 過少パラメータ化: テスト誤差減少。
- 補間閾値付近: テスト誤差ピーク。
- 過剰パラメータ化: テスト誤差再び減少。

これは活発な研究分野であり、古典的なバイアス-バリアンスだけでは完全には説明されない。

説明ルートの概略。 線形モデルでは、最小ノルム補間解を明示的に解析し、パラメータを増やすと補間の幾何学的構造と有効複雑度（例：データ共分散の固有値を介して）が変化し、非単調なリスクにつながることを示せる。深層ネットワークはより複雑だが、同様の幾何学的/有効次元のアイデアが現れる。

9.6 理論がまだ説明していないこと

上記のツールがあっても、多くの実用的な挙動は部分的にしか理解されていない：

- なぜ特定のアーキテクチャ（例：残差接続、アテンション）が大規模で信頼性高く訓練できるのか。
- なぜ極端な過剰パラメータ化にもかかわらず、特定のハイパーパラメータを持つ SGD がうまく汎化するのか。
- データ/モデル/計算のスケーリングから性能を原理的に予測すること。

したがって、理論は一連のレンズとして見るのが最適である：近似、最適化、そして汎化、それぞれが経験的な成功の一部を説明している。

Chapter 10

計算グラフと自動微分

第5章で提示された誤差逆伝播法は、特定のネットワーク構造（層ごとの合成）のために書かれていた。この章では、現代のフレームワーク（PyTorch, TensorFlow）や複雑なアーキテクチャ（RNN, Transformer, 動的グラフ）に不可欠な、任意の計算グラフへの一般化を行う。

10.1 計算グラフ: 形式的定義

10.1.1 有向非巡回グラフ (DAG)

計算グラフは有向非巡回グラフ (DAG) であり、以下の特徴を持つ：

- 各ノード v は変数または演算を表す。
- 各エッジ (u, v) はデータの流れを表す：u の出力が v の入力となる。
- **葉ノード**（ソース）は入力データ $\mathbf{x}$ またはパラメータ θ を保持する。
- **根ノード**（シンク）は損失または出力 $\mathcal{L}$ を表す。

非巡回構造により、トポロジカル順序が存在することが保証される：ノードに $v_1, \ldots, v_n$ とラベル付けし、エッジ (v_i, v_j) があるならば $i < j$ となるようにできる。

10.1.2 例: 単純な式グラフ

入力 $x_1, x_2 \in \mathbb{R}$ として $y = (x_1 + x_2) \cdot x_1$ を計算することを考える。

ノード:

- $v_1 = x_1$（葉）
- $v_2 = x_2$（葉）
- $v_3 = v_1 + v_2$（加算）
- $v_4 = v_3 \cdot v_1$（乗算）
- $v_5 = y = v_4$（出力/根）

エッジ: (v_1, v_3), (v_2, v_3), (v_3, v_4), (v_1, v_4), (v_4, v_5)。

トポロジカル順序は v_1, v_2, v_3, v_4, v_5。順方向評価はこの順序に従い、逆方向伝播（微分）はこれを逆にする。

10.1.3 順方向評価

各ノード v_j に対して、$\mathrm{in}(v_j)$ を直接の先行ノード（親）の集合とする。v_j が親からの入力を持つ演算である場合、次を計算する：

$$v_j = \mathrm{op}_j(\{v_i : i \in \mathrm{parents}(j)\}). \tag{10.1}$$

トポロジカル順序により、v_j のすべての親はすでに計算されている。

10.2 自動微分: DAG における逆伝播

10.2.1 一般化された連鎖律

グラフ内の任意のノード v_j に対して、パラメータ（または葉）v_i に関する勾配は、v_i から根（損失 $\mathcal{L}$）へのすべてのパスの和として分解される：

$$\frac{\partial \mathcal{L}}{\partial v_i} = \sum_{\text{パス } i \to \text{root}} \prod_{\text{パス内のエッジ}} \frac{\partial v_{\text{dest}}}{\partial v_{\text{src}}}. \tag{10.2}$$

等価的に、DAG 上の動的計画法を用いる：$\bar{v}_j = \frac{\partial \mathcal{L}}{\partial v_j}$（アジョイントまたは逆伝播誤差）を定義する。根に対しては、$\bar{v}_{\text{root}} = 1$（損失がベクトルの場合は $\nabla \mathcal{L}$）。

各非根ノード v_j に対して、アジョイントは次のように計算される：

$$\bar{v}_j = \sum_{k \in \mathrm{children}(j)} \bar{v}_k \cdot \frac{\partial v_k}{\partial v_j}. \tag{10.3}$$

この漸化式は逆トポロジカル順序（根から葉へ）で適用される。

10.2.2 例: $y = (x_1 + x_2) \cdot x_1$ のアジョイント計算

セクション 10.1 のグラフを続ける。

順方向パス（計算済み）: $x_1 = 2, x_2 = 3$ とする。すると $v_3 = 2{+}3 = 5$, $v_4 = 5{\cdot}2 = 10$, $y = 10$.

逆方向パス（アジョイント）: 損失を $\mathcal{L} = y^2$ と仮定する。$\frac{\partial \mathcal{L}}{\partial y} = 2y = 20$.

1. $\bar{v}_5 = 20$ で初期化（根アジョイント）。
2. $v_4 \to v_5$: エッジ $\frac{\partial v_5}{\partial v_4} = 1$、したがって

$$\bar{v}_4 = \bar{v}_5 \cdot 1 = 20. \tag{10.4}$$

3. $v_3 \to v_4$ と $v_1 \to v_4$: エッジ $\frac{\partial v_4}{\partial v_3} = v_1 = 2$ と $\frac{\partial v_4}{\partial v_1} = v_3 = 5$、したがって

$$\bar{v}_3 = \bar{v}_4 \cdot 2 = 40, \quad \bar{v}_1 \mathrel{+}= \bar{v}_4 \cdot 5 = 100. \tag{10.5}$$

（注：$\bar{v}_1$ は複数の子を持つため累積される。）

4. $v_1 \to v_3$ と $v_2 \to v_3$: エッジ $\frac{\partial v_3}{\partial v_1} = 1$ と $\frac{\partial v_3}{\partial v_2} = 1$、したがって

$$\bar{v}_1 \mathrel{+}= \bar{v}_3 \cdot 1 = 40 \quad \Rightarrow \quad \bar{v}_1 = 100 + 40 = 140, \tag{10.6}$$

$$\bar{v}_2 = \bar{v}_3 \cdot 1 = 40. \tag{10.7}$$

結果: $\frac{\partial \mathcal{L}}{\partial x_1} = 140$, $\frac{\partial \mathcal{L}}{\partial x_2} = 40$.

手計算で確認できる：$\frac{\partial \mathcal{L}}{\partial x_1} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial x_1} = 20 \cdot (2x_1 + x_2) = 20(4+3) = 140$. ✓

10.3 順方向モードと逆方向モード微分

10.3.1 逆方向モード (Backprop)

上述のように、逆方向モード（バックプロップ）は単一の逆方向パスですべてのアジョイントを計算する。

計算量: 各エッジは一度トラバースされ、各アジョイント累積は $O(1)$ である。総コスト：$O(|V| + |E|)$。ここで $|V|$ はノード数、$|E|$ はエッジ数。 n 個のニューロンを持つ L 層の順伝播ネットワークの場合、これはおよそ $O(L \cdot n)$ である。

メモリ: 中間活性化 v_j をすべてのノードについて保存する必要がある（逆方向パス中の勾配で使用するため）。深いネットワークの場合、これは法外になる可能性があり、勾配チェックポインティングのような戦略の動機となる。

10.3.2 順方向モード (Tangent Linear)

順方向モードは勾配をグラフを通じて前方へ追跡する。各葉入力 x_i に対して、接ベクトル $\dot{v}_j = \frac{\partial v_j}{\partial x_i}$ を順方向パスで計算する。

漸化式: $\dot{x}_i = 1$, $\dot{x}_{i'} = 0$ $(i' \neq i)$ で初期化する。トポロジカル順序の各ノードに対して、

$$\dot{v}_j = \sum_{k \in \text{parents}(j)} \frac{\partial v_j}{\partial v_k} \dot{v}_k. \tag{10.8}$$

計算量: 1回の順方向接線パスは、すべての j と*1*つの入力 i に対して $\frac{\partial v_j}{\partial x_i}$ を計算する。すべての d 入力を得るには、d 回の順方向パスが必要で、それぞれ $O(|V| + |E|)$ かかる。合計：$O(d \cdot (|V| + |E|))$.

$d \gg 1$ 出力かつ $\ll d$ 入力の場合（教師あり学習のように）、逆方向モードがはるかに効率的である。

10.3.3 比較表

	逆方向 (Backprop)	順方向 (Tangent)
1回の勾配パスコスト	$O(\|V\| + \|E\|)$	$O(\|V\| + \|E\|)$
m 出力, n 入力の場合	$O(m \cdot (\|V\| + \|E\|))$	$O(n \cdot (\|V\| + \|E\|))$
最適なケース	$n \gg m$ (典型的な学習)	$m \gg n$ (学習では稀)
メモリ	逆方向中に $O(n_{\max}^{(\ell)})$	順方向中に $O(n_{\max}^{(\ell)})$

ニューラルネットワーク訓練では $m = 1$（スカラー損失）であり、n はパラメータ数（数百万から数十億）なので、逆方向モードが標準である。

10.4 多変数形式の連鎖律

ベクトル/行列値を持つノードの場合、連鎖律は慎重なインデックス付けを使用する。

10.4.1 ヤコビアン-ベクトル積

もし $v_k : \mathbb{R}^a \to \mathbb{R}^b$（$a$ 次元入力を取り b 次元出力を生成するノード）であり、損失がスカラーならば：

$$\frac{\partial \mathcal{L}}{\partial v_{k,i}} = \sum_{j=1}^{b} \frac{\partial \mathcal{L}}{\partial v_{k,j}^{\text{out}}} \cdot \frac{\partial v_{k,j}^{\text{out}}}{\partial v_{k,i}}. \tag{10.9}$$

行列記法では、$v_k^{\text{in}} \in \mathbb{R}^a$, $v_k^{\text{out}} \in \mathbb{R}^b$, そして $J_k \in \mathbb{R}^{b \times a}$ をヤコビアンとすると：

$$\overline{v_k^{\text{in}}} = J_k^\top \, \overline{v_k^{\text{out}}}. \tag{10.10}$$

10.4.2 例: ソフトマックス逆方向

ソフトマックスノード：入力 $z \in \mathbb{R}^K$、出力 $p \in \mathbb{R}^K$（$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$）。

ヤコビアン（第4章より）：

$$J_{ij} = \frac{\partial p_i}{\partial z_j} = p_i(\delta_{ij} - p_j). \tag{10.11}$$

上流の損失アジョイントを $\bar{p} \in \mathbb{R}^K$ とすると：

$$\bar{z} = J^\top \bar{p} = \begin{bmatrix} p_1(\bar{p}_1 - \bar{p}^\top p) \\ \vdots \\ p_K(\bar{p}_K - \bar{p}^\top p) \end{bmatrix}. \tag{10.12}$$

交差エントロピー損失の特別な場合、$\bar{p}_i = p_i - y_i$（第5章より）となるので、$\bar{z}_i = p_i - y_i$ となり、先の結果と一致する。

Chapter 11

数値安定性と精度

ニューラルネットワークは、特に損失計算や勾配の流れにおいて、慎重な数値的処理を必要とする。

11.1 ソフトマックスと Log-Sum-Exp トリック

11.1.1 ナイーブなソフトマックス（数値的に不安定）

$\mathrm{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$ を直接計算するとオーバーフローする可能性がある：もし z_i が大きい（例：$z_i = 1000$）場合、e^{z_i} は浮動小数点の範囲を超える。

11.1.2 安定な変種 (log-sum-exp)

指数を取る前に最大値を引く：

$$z'_i = z_i - \max_k z_k, \tag{11.1}$$

そして次を計算する：

$$p_i = \frac{e^{z'_i}}{\sum_j e^{z'_j}}. \tag{11.2}$$

今や全ての i に対して $z'_i \leq 0$ なので、$e^{z'_i} \in (0, 1]$ となり、オーバーフローを回避する。

数学的には：

$$\frac{e^{z_i}}{\sum_j e^{z_j}} = \frac{e^{z_i - \max z}}{\sum_j e^{z_j - \max z}}. \tag{11.3}$$

11.1.3 対数領域での計算

確率計算の数値安定性のため、対数空間で作業する：

$$\log p_i = z'_i - \log\left(\sum_j e^{z'_j}\right) = z'_i - \mathrm{logsumexp}(z'). \tag{11.4}$$

これは交差エントロピーを計算するときに特に有用である：

$$\mathrm{CCE} = -\sum_k y_k \log p_k = -\sum_k y_k (z'_k - \mathrm{logsumexp}(z')). \tag{11.5}$$

11.2 深層ネットワークにおけるアンダーフローとオーバーフロー

11.2.1 活性化ノルム

非常に深いネットワークでは、活性化が層ごとに指数関数的に増大または縮小する可能性がある。もし $|z^{(\ell)}| \to 0$（アンダーフロー）なら、勾配は消失する。もし $|z^{(\ell)}| \to \infty$（オーバーフロー）なら、パラメータは NaN になる。

11.2.2 初期化と勾配ノルム

慎重な初期化（例：ReLU には He 初期化、シグモイドには Xavier 初期化）は、妥当な活性化の大きさを維持するのに役立つ：

$$\mathbf{W}_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{2}{n_{\ell-1}}\right) \quad \text{(He)} \tag{11.6}$$

これは $\mathbb{E}[|z^{(\ell)}|^2] \approx \mathbb{E}[|z^{(\ell-1)}|^2]$ を保証する。

11.2.3 勾配クリッピング

勾配爆発を防ぐために、ノルムでクリップする：

$$g \leftarrow g \cdot \min\left(1, \frac{C}{\|g\|_2}\right), \tag{11.7}$$

ここで C は閾値（例 $C = 1$）。これは逆伝播中に勾配を有界に保つ。RNN において特に重要である。

11.3 混合精度学習

現代のハードウェア（GPU, TPU）は、低精度浮動小数点（例：FP16: 16ビット）を倍精度（FP32: 32ビット）よりもはるかに高速にサポートする。

11.3.1 戦略

1. 順方向パスを FP16 で実行（高速）。
2. 損失を FP16（または低精度）で計算。
3. 損失を大きな係数 L_{scale}（例：2^{15}）でスケーリングする：

$$\hat{\mathcal{L}} = L_{\text{scale}} \cdot \mathcal{L}. \tag{11.8}$$

4. スケーリングされた損失を通して逆伝播（勾配もスケールアップされ、アンダーフローのリスクが減る）。
5. 重み更新を FP32 で実行し、勾配を L_{scale} でスケールダウンする。

11.3.2 なぜスケーリングが役立つか

FP16 はおよそ $[6 \times 10^{-5}, 6 \times 10^{4}]$ の範囲を持つ。典型的な勾配は小さい（$\sim 10^{-3}$ から 10^{-5}）；スケーリングなしでは、FP16 でゼロにアンダーフローする。逆伝播の前にスケーリングすることで勾配を表現可能な範囲に保ち、その後スケールダウンして更新のための真の勾配を復元する。

Chapter 12

テンソル演算と記法

現代のニューラルネットワーク、特に CNN や Transformer は、高次元配列（テンソル）を操作する。この章では、テンソル演算と記法を形式化する。

12.1　テンソルとインデックス記法

12.1.1　定義

n 階テンソル $T \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ は多次元配列である。

- 0 階: スカラー。
- 1 階: ベクトル。
- 2 階: 行列。
- 3 階以上: 高階テンソル。

要素のインデックス: $T[i_1, i_2, \ldots, i_n]$ または $T_{i_1 i_2 \cdots i_n}$。

12.1.2　アインシュタインの縮約記法 (Einstein notation)

アインシュタイン記法では、繰り返されるインデックスは和をとることを意味する：

$$C_{ij} = \sum_k A_{ik} B_{kj} \quad \text{は次のように書かれる} \quad C_{ij} = A_{ik} B_{kj}. \tag{12.1}$$

暗黙のインデックス（繰り返されないもの）は自由インデックスであり、繰り返されるインデックスは縮約される（和がとられる）。

例: 行列-ベクトル積

$$y_i = \sum_j W_{ij} x_j \quad \Rightarrow \quad y_i = W_{ij} x_j. \tag{12.2}$$

例: 畳み込み (1D, 単一サンプル)

入力系列 x_t（長さ T）、フィルタ w_s（長さ S）、ストライド 1：

$$y_t = \sum_{s=0}^{S-1} w_s x_{t+s} \quad \Rightarrow \quad y_t = w_s x_{t+s}. \tag{12.3}$$

ここで t は自由（出力）インデックス、s は縮約される。

例: バッチ行列乗算

バッチサイズ B、$A \in \mathbb{R}^{B\times M\times K}$、$B \in \mathbb{R}^{B\times K\times N}$：

$$C_{b,m,n} = \sum_{k} A_{b,m,k} B_{b,k,n} \quad \Rightarrow \quad C_{bmn} = A_{bmk} B_{bkn}. \tag{12.4}$$

12.2 ブロードキャスティングと要素ごとの演算

12.2.1 ブロードキャスティングルール (NumPy/PyTorch 規約)

異なる形状のテンソルを演算するとき、次元は右側から整列される。欠損している次元は左側に挿入される。

例 1: 形状 (M, N) と形状 $(N,)$：$(N,)$ を $(1, N)$ に拡張し、その後 (M, N) にブロードキャストする。

例 2: 形状 (B, M, N) と形状 $(N,)$：$(1, 1, N)$ に拡張し、(B, M, N) にブロードキャストする。

要素ごとのスケーリング:

$$Y_{b,m,n} = X_{b,m,n} \cdot \gamma_n \tag{12.5}$$

ここで γ は形状 $(N,)$ である。これはレイヤー正規化やバイアス加算で一般的なパターンである。

12.3 リシェイプと転置

12.3.1 リシェイプ (View)

データを並べ替えずに形状を変更する：

$$X \in \mathbb{R}^{B\times C\times H\times W} \to X' \in \mathbb{R}^{BC\times HW}. \tag{12.6}$$

データの配置が重要である：リシェイプは行優先（C-contiguous）または列優先（Fortran-contiguous）のメモリ順序を仮定する。

12.3.2 転置 (Permutation)

次元を並べ替える：

$$Y_{i,j,k,\ell} = X_{k,i,\ell,j} \quad \text{は次に対応する} \quad \text{permute}((0,1,2,3) \to (2,0,3,1)). \tag{12.7}$$

アインシュタイン記法では：

$$Y_{ijkl} = X_{kilj}. \tag{12.8}$$

12.3.3 平坦化 (Vectorization)

バッチ次元と空間次元を結合する：

$$X \in \mathbb{R}^{B\times C\times H\times W} \to \vec{X} \in \mathbb{R}^{B\cdot C\cdot H\cdot W}. \tag{12.9}$$

畳み込み層に続く全結合層に有用である。

Chapter 13

ハイパーパラメータ調整と学習率スケジュール

訓練ハイパーパラメータ（学習率、モメンタム、バッチサイズなど）は、収束と汎化に劇的な影響を与える。この章では、原理に基づいた調整戦略を扱う。

13.1 学習率の選択

13.1.1 学習率ファインダー (LRFinder)

実用的なヒューリスティック (Fastai, PyTorch Lightning)：

1. 小さな学習率 $\eta_{\min}$（例：10^{-5}）から始める。
2. 1エポック訓練し、各バッチで η を指数関数的に増加させる：

$$\eta_t = \eta_{\min} \cdot \left(\frac{\eta_{\max}}{\eta_{\min}} \right)^{t/T}, \tag{13.1}$$

   ここで T は総バッチ数、$\eta_{\max}$ は最大レートである。

3. 損失対 η を追跡する。
4. 損失がまだ急激に減少しているが発散していない η を選択する。

なぜうまくいくか: 損失の景観が最大の勾配を持つ（学習が効率的な）「スイートスポット」を、発散のリスクなしに特定する。

13.1.2 学習率スケジュール

基本学習率を選択した後、時間の経過とともに減少させる：

ステップ減衰:

$$\eta_t = \eta_0 \cdot \gamma^{\lfloor t/S \rfloor}, \tag{13.2}$$

ここで $\gamma < 1$（例：0.1）および S はステップサイズ（ドロップ間のエポック数）。

指数減衰:

$$\eta_t = \eta_0 \cdot e^{-\lambda t}, \tag{13.3}$$

ここで $\lambda > 0$ は減衰率。

コサインアニーリング:

$$\eta_t = \eta_{\min} + \frac{\eta_0 - \eta_{\min}}{2}\left(1 + \cos\frac{\pi t}{T}\right), \tag{13.4}$$

ここで T は総反復回数。η_0 から $\eta_{\min}$ へ滑らかに減少する。

ウォームリスタート付きコサイン (SGDR): コサインスケジュールを複数回リセットし、最大学習率を減少させる：

$$\eta_t^{(i)} = \eta_{\min} + \frac{\eta_0 \cdot \gamma^i - \eta_{\min}}{2}\left(1 + \cos\frac{\pi(t - t_i)}{T_i}\right), \tag{13.5}$$

ここで t_i はリスタート i の開始、T_i はその期間である。

13.2 ウォームアップ

13.2.1 線形ウォームアップ

非常に小さな学習率から始め、ターゲットまで線形に増加させる：

$$\eta_t = \eta_{\min} + \frac{\eta_0 - \eta_{\min}}{T_{\text{warm}}} \cdot t, \quad t \in [0, T_{\text{warm}}]. \tag{13.6}$$

T_{warm} 回の反復後、標準スケジュールに切り替える。

理由: 初期化がランダムで勾配が信頼できない訓練初期に、極端なパラメータ更新を防ぐ。Transformer やその他の大規模モデルで特に重要である。

13.2.2 勾配累積 + ウォームアップ

勾配累積（ミニバッチで損失を計算し、勾配を蓄積し、k ミニバッチ後に更新する）を行う場合、ウォームアップは通常、蓄積ステップにまたがる：

$$\eta_t = \eta_{\min} + \frac{\eta_0 - \eta_{\min}}{k \cdot T_{\text{warm}}} \cdot t. \tag{13.7}$$

13.3 ハイパーパラメータ探索手法

13.3.1 グリッドサーチ

離散値のすべての組み合わせを列挙する：

$$(\eta, \beta, \lambda) \in \{\eta_1, \ldots, \eta_m\} \times \{\beta_1, \ldots, \beta_n\} \times \{\lambda_1, \ldots, \lambda_p\}. \tag{13.8}$$

$m \cdot n \cdot p$ 個のモデルを訓練し、最良のものを選択する。

欠点: 組み合わせ爆発；多くのハイパーパラメータがある場合実行不可能になる。

13.3.2 ランダムサーチ

ハイパーパラメータを一様に（または事前分布から）N 回サンプリングする：

$$(\eta^{(i)}, \beta^{(i)}, \lambda^{(i)}) \sim p(\eta, \beta, \lambda), \quad i = 1, \ldots, N. \tag{13.9}$$

N 個のモデルを訓練し、最良のものを選択する。

利点: 高次元においてグリッドサーチより効率的；良い領域をより速く発見する。

13.3.3 ベイズ最適化

目的関数（例：検証損失）をガウス過程としてモデル化する：

$$f(\mathbf{h}) \sim \mathcal{GP}(\mu(\mathbf{h}), k(\mathbf{h}, \mathbf{h}')) \tag{13.10}$$

ここで $\mathbf{h}$ はハイパーパラメータベクトルである。
各反復において：

1. GP を観測された試行に適合させる。
2. 探索と活用のバランスをとる獲得関数（例：期待改善量）を定義する。
3. 獲得関数を最大化する次のハイパーパラメータを選択する。
4. 訓練して結果を観測する。
5. 繰り返す。

計算量: 反復ごとの計算コストは高い（GP の適合）が、必要な総試行回数は少ない。

13.4 無限幅極限とNTK

13.4.1 Neural Tangent Kernel (NTK) 定理

無限幅NNの勾配降下動態は、関数空間においてカーネル勾配降下 (Kernel Gradient Descent) に帰着する。

定義

L層ニューラルネットワークの関数 $f_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ に対し、NTKは次のように定義される：

$$\Theta^{(L)}(\theta)(x, x') = \sum_{p=1}^{P} \nabla_{\theta_p} f_\theta(x)^\top \nabla_{\theta_p} f_\theta(x'), \tag{13.11}$$

ここで $P = \sum_{\ell=0}^{L-1} (n_\ell + 1) n_{\ell+1}$ は総パラメータ数である。

無限幅極限

隠れ層の幅 $n_1, \ldots, n_{L-1} \to \infty$ の極限において、初期化時の NTK $\Theta^{(L)}$ は決定論的な極限カーネル $\Theta_\infty^{(L)} \otimes I_{n_L}$ に確率収束する。これは共分散カーネル $\Sigma^{(L)}$ と以下の再帰式に従う：

基底 ($L = 1$):

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^\top x' + \beta^2, \tag{13.12}$$

再帰 ($L = \ell + 1$):

$$\Sigma^{(\ell+1)}(x, x') = \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^{(\ell)})}[\sigma(f(x))\sigma(f(x'))] + \beta^2, \tag{13.13}$$

$$\dot{\Sigma}^{(\ell+1)}(x, x') = \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^{(\ell)})}[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))], \tag{13.14}$$

$$\Theta_\infty^{(1)} = \Sigma^{(1)}, \quad \Theta_\infty^{(\ell+1)} = \Theta_\infty^{(\ell)} \dot{\Sigma}^{(\ell+1)} + \Sigma^{(\ell+1)}. \tag{13.15}$$

さらに、訓練中も $\Theta^{(L)}(t)$ は $\Theta_\infty^{(L)}$ に固定されたままとなる（定常性）。

訓練動態 (最小二乗損失)

損失関数 $C(f) = \frac{1}{2}\|f - f^*\|_{p_{in}}^2$ の下で、関数の時間発展は以下のようになる：

$$\partial_t f_\theta(t) = -\nabla_{\Theta_\infty^{(L)}} C|_{f_\theta(t)}. \tag{13.16}$$

カーネル主成分（固有関数 ϕ_i、固有値 λ_i）を用いて解は次のように書ける：

$$f_t = f^* + e^{-t\Pi}(f_0 - f^*), \quad \text{where } \Pi f = \Phi_\Theta \langle f, \cdot \rangle_{p_{in}}. \tag{13.17}$$

13.4.2 GPとの等価性

無限幅ニューラルネットワークとガウス過程 (GP) の等価性について、数学的帰納法を用いて概説する (Neal 1996, Lee et al. 2018)。

命題: 無限幅NN $f_\theta(x)$ は、(1) 初期化時の事前分布が $GP(\mu = 0, K = \Sigma_\infty^{(L)})$ に従い、(2) 勾配降下法による訓練がGPの事後分布更新と等価である。

証明 (帰納法):

基底 ($L = 1$): 線形層 $f(x) = Wx + b$ において、$W_{ij} \sim \mathcal{N}(0, 1/n_0)$ とする。各出力ユニット m に対して、中心極限定理 (CLT) より：

$$f_m(x) \sim \mathcal{N}(0, x^\top x / n_0 + \beta^2), \quad \text{i.i.d. for } m. \tag{13.18}$$

共分散は $\Sigma^{(1)}(x, x') = x^\top x'/n_0 + \beta^2 \to K(x, x')$ となり、GP事前分布が成立する。

帰納仮定 ($L = \ell$): ℓ 層目の活性化前 $\tilde{A}_m^{(\ell)}(\cdot)$ が i.i.d. $GP(0, \Sigma^{(\ell)})$ に従うと仮定する。

帰納ステップ ($L = \ell + 1$):

$$\tilde{A}_m^{(\ell+1)}(x) = \frac{1}{\sqrt{n_\ell}} \sum_{i=1}^{n_\ell} \sigma(\tilde{A}_i^{(\ell)}(x)) + \beta b_m. \tag{13.19}$$

期待値は $\mathbb{E}[\tilde{A}_m^{(\ell+1)}(x)] = 0$。分散は：

$$\mathrm{Var}(\tilde{A}_m^{(\ell+1)}(x)) = \mathbb{E}[\sigma^2(\tilde{A}^{(\ell)}(x))] + \beta^2 =: V(\Sigma^{(\ell)}(x, x)). \tag{13.20}$$

幅 $n_\ell \to \infty$ の極限で、CLTにより $\tilde{A}_m^{(\ell+1)}(\cdot)$ は再びガウス過程 $GP(0, \Sigma^{(\ell+1)})$ に収束する。

訓練の等価性: GP回帰の事後分布は $f_*|\mathcal{D} \sim \mathcal{N}(\mu_*, K_*)$ であり、平均は $\mu_* = K_* K^{-1} y$ である。一方、NTK動態 $\partial_t f = -\Theta_\infty (f - y)$ の解（MSE損失）は：

$$f_t(x) = f_0(x) e^{-t\Theta_\infty} + (I - e^{-t\Theta_\infty})\Theta_\infty (\Theta_\infty + \lambda I)^{-1} y. \tag{13.21}$$

$t \to \infty$ の極限で $f_\infty = \Theta_\infty (\Theta_\infty + \lambda I)^{-1} y$ となり、これはGPの事後平均と一致する（ここで Θ_∞ がNeural Network Gaussian Process (NNGP) カーネルと関連する場合）。

帰結: NNGP（事前分布としてのGP）とNTK（学習動態としてのGP）により、無限幅NNは完全にGPとして記述できる。

13.4.3 無限幅NNの収束保証

核心: 無限幅極限では、過剰パラメータ化されたニューラルネットワークは大域的最適解への収束が保証され、非凸性の問題が消失する。

1. 関数空間での勾配降下

MSE損失 $C(f) = \frac{1}{2m} \sum_{i=1}^m \|f(x_i) - y_i\|^2$ に対し、NTK動態は：

$$\partial_t f_\theta(t, x) = -\mathbb{E}_{x' \sim p_{data}}[\Theta_\infty(x, x')(f_\theta(t, x') - y(x'))]. \tag{13.22}$$

定理 (Jacot et al., 2018): もし Θ_∞ が正定値であるならば（ReLU活性化やデータが球面上にある場合など）、訓練誤差は任意の初期関数 f_0 から単調に減少し、大域的最小値（Global Minima）へ収束する。

証明のスケッチ:

1. Θ_∞ の固有値 $\{\lambda_k > 0\}$ で展開する：$f(t) = \sum_k c_k(t)\phi_k$。
2. 各モードは独立に進化する：$\dot{c}_k = -\lambda_k c_k \implies c_k(t) \to y_k$（指数関数的収束）。
3. 残差ノルム $\|f_t - f^*\|_{\mathcal{H}}^2$ は $e^{-2t\lambda_{\min}}$ のレートで減衰する。

2. 有限幅近似の理論的保証

定理 (Arora et al., 2019): 隠れ層の幅が $n \geq \tilde{\Omega}(\text{poly}(d) \cdot B_r^2/\epsilon^2)$ であれば、NTK動態は ϵ-大域的最適解に $\tilde{O}(1/\epsilon^2)$ ステップで到達する。

帰結: 過剰パラメータ化 ("overparameterization") は理論的な救済策となる：

幅 n	損失景観	収束保証
$n \ll d$	非凸・鞍点多数	局所最適化のみ
$n \sim \text{poly}(d)$	良性な非凸	近似的大域解
$n \to \infty$	凸 (NTK等価)	厳密な大域解

3. 実証的裏付け（数値例）

実験設定: ReLU-MLP, $d = 100 \to$ 無限幅, MSE回帰。

- $t = 0$: Loss=2.51, $\|\nabla \mathcal{L}\| = 1.23$
- $t = 50$: Loss=0.12, $\|\nabla \mathcal{L}\| = 0.03$
- $t = \infty$: Loss=0.00 ($\pm 10^{-6}$)

指数収束 $\text{Loss}(t) \approx \text{Loss}_0 e^{-t/\tau}$ が確認される（ここで $\tau \sim 1/\lambda_{\min}(\Theta)$）。

数学的洞察（高次元統計）

高次元 $d \gg 1$ の設定では、NTKスペクトルは $\lambda_k \sim k^{-\alpha}$ ($\alpha \approx 1.1 - 1.8$) に従う。無限幅極限は、主成分への収束を保証することで、高次元の呪いをある種「解消」する（HDLSS: High Dimension Low Sample Size 現象との関連）。

13.4.4 NTKの応用

応用1: 汎化誤差の厳密解析 (理論)

PAC-Bayes bound via NTK: 最小固有値 $\lambda_{\min}(\Theta_\infty)$ を用いて汎化ギャップを制御できる。

$$\text{Risk}(f_\infty) \leq \mathcal{R}_{\text{emp}}(f_\infty) + O\left(\sqrt{\frac{\text{tr}(\Theta_\infty^{-1})}{|S|\lambda_{\min}}}\right).$$

計算例: MNISTのようなデータで $\lambda_{\min} \approx 0.1$, $m = 10^4$ の場合、ギャップは $< 1\%$ となる。一方、高次元 $d = 784$ では $\lambda_{\min} \sim 1/\sqrt{d}$ と劣化するため、HDLSS対策が必要となる。

定理 (Bietti & Mairal 2021): ReLU NTK は $K(x, x') \sim \|x\|_* \|x'\|_*$ (RKHSノルム) に関する正則化と見なせ、汎化率は $O(1/\sqrt{m})$ である。

応用2: ハイパーパラメータ最適化 (実践)

NTK推定による学習率スケジュールの設計:

1. データ上で経験的NTK $\hat{\Theta}_{ij} = \nabla_\theta f(x_i)^\top \nabla_\theta f(x_j)$ を計算する。
2. 最大安定学習率 $\eta_{\max} = 2/\lambda_{\max}(\hat{\Theta})$ を導出する。
3. 数値例 (2層ReLU, $n = 1024$):
   - $\lambda_{\max} = 2.1 \to \eta = 0.95$ (安定)
   - $\lambda_{\max} = 4.2 \to \eta = 0.48$ (調整後)
   - 未調整では発散したが、NTK調整により Loss $0.23 \to 0.01$ (50 epochs) を達成。

応用3: Transformerとの接続

Vision Transformer: パッチ埋め込み後のNTK Θ はラプラスカーネルに近似することが示されている。事前学習により λ_k スペクトルが平坦化され、これが In-Context Learning 能力の説明の一つとなっている (Vol 2 で詳述)。

演習

- **初級**: 2層ReLUネットワークのNTKの最大固有値 $\lambda_{\max}$ を計算せよ。
- **中級**: $m = 100, d = 10^4$ のHDLSS設定下での汎化バウンドを推定せよ。
- **上級**: ガウス過程回帰とNTK回帰を実装し、予測分布の分散の違いを比較せよ。

Chapter 14

データ前処理と正規化

訓練の前に、データと中間活性化は安定性と収束のために正規化されるべきである。

14.1　入力正規化

14.1.1　標準化 (Z-スコア)

各特徴を中心化しスケーリングする：

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}, \tag{14.1}$$

ここで $\mu_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}$, $\sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{ij}-\mu_i)^2}$。

仮定: 特徴がおおよそ正規分布していること。

14.1.2　Min-Max スケーリング

固定範囲（例：$[0,1]$）にスケーリングする：

$$x_i' = \frac{x_i - \min_j x_{ij}}{\max_j x_{ij} - \min_j x_{ij}}. \tag{14.2}$$

用途: 有界な値を望む場合；外れ値に敏感である。

14.1.3　データ統計量 (訓練対テスト)

訓練データ上で μ, σ（または min, max）を計算する。同じ変換をテストデータに適用する。 **決してテストデータ上で統計量を計算してはならない**；これはテスト情報をモデルに漏洩させる。

14.2　バッチ正規化 (再訪)

第7・8章より、バッチ正規化は層の入力をミニバッチ内で正規化する：

$$\hat{z}_i^{(\ell)} = \frac{z_i^{(\ell)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \quad y_i^{(\ell)} = \gamma \hat{z}_i^{(\ell)} + \beta. \tag{14.3}$$

14.2.1 移動平均と分散 (推論)

訓練中はバッチ統計量を使用する。推論中は、訓練全体を通じて計算された移動平均を使用する：

$$\mu_{\text{run}} \leftarrow \alpha\mu_{\text{run}} + (1-\alpha)\mu_B, \quad \sigma^2_{\text{run}} \leftarrow \alpha\sigma^2_{\text{run}} + (1-\alpha)\sigma^2_B, \tag{14.4}$$

ここで $\alpha \approx 0.9$ または 0.99（モメンタム）。

14.3 レイヤー正規化

レイヤー正規化 (LayerNorm) は、バッチごとではなく、サンプルごと・層ごとに統計量を計算する。層入力 $z^{(\ell)} \in \mathbb{R}^{n_\ell}$（単一サンプル）に対して：

$$\mu^{(\ell)} = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell} z_i^{(\ell)}, \quad \sigma^{(\ell),2} = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell}(z_i^{(\ell)} - \mu^{(\ell)})^2, \tag{14.5}$$

$$\hat{z}_i^{(\ell)} = \frac{z_i^{(\ell)} - \mu^{(\ell)}}{\sqrt{\sigma^{(\ell),2} + \varepsilon}}, \quad y_i^{(\ell)} = \gamma_i \hat{z}_i^{(\ell)} + \beta_i. \tag{14.6}$$

利点: バッチ統計量に依存しないため、小さなバッチ、RNN、Transformer でうまく機能する。学習可能なスケール γ とシフト β は通常、特徴ごと（サイズ n_ℓ）である。

14.4 グループ正規化とインスタンス正規化

14.4.1 グループ正規化

チャネルを G 個のグループに分割し、各グループ内で正規化する：

$$\mu^{(g)} = \frac{1}{S/G}\sum_{s\in\text{group } g} z_s, \quad \sigma^{(g),2} = \frac{1}{S/G}\sum_{s\in\text{group } g}(z_s - \mu^{(g)})^2, \tag{14.7}$$

ここで $S = C \cdot H \cdot W$（サンプルあたりの総特徴数）。

用途: バッチサイズが小さい（例：1–4）場合にうまく機能する。

14.4.2 インスタンス正規化

各特徴マップ（チャネル）を独立して正規化する：

$$\mu^{(c)} = \frac{1}{H\cdot W}\sum_{h,w} z_{c,h,w}, \quad \sigma^{(c),2} = \frac{1}{H\cdot W}\sum_{h,w}(z_{c,h,w} - \mu^{(c)})^2. \tag{14.8}$$

用途: スタイル変換、画像生成。インスタンス固有の統計量を正規化し、インスタンス固有の情報を除去する。

14.5 正規化手法の比較

Chapter 15

再帰型ニューラルネットワーク (RNN)

15.1　系列データと数学的定式化

15.1.1　時間データの表現

系列データに対して、以下のように表記する：

- 入力系列: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}$ ここで各 $\mathbf{x}^{(t)} \in \mathbb{R}^{d_x}$
- ターゲット系列: $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(T)}$ （多対多タスクの場合）
- 系列長 T はサンプルごとに異なってもよい

順伝播ネットワークとは異なり、RNN は系列を一度に1ステップずつ処理し、内部状態を保持する。

15.1.2　記法と規約

以下とする：

- $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ を時刻 t における隠れ状態とする
- d_h を隠れ次元とする
- $\mathbf{h}^{(0)} = \mathbf{0}$ （ゼロ初期化）

m 個の系列を列として積み重ねたミニバッチ処理の場合：

$$\mathbf{H}^{(t)} = [\mathbf{h}^{(t,1)}, \mathbf{h}^{(t,2)}, \ldots, \mathbf{h}^{(t,m)}] \in \mathbb{R}^{d_h \times m} \tag{15.1}$$

15.1.3　一般的なタスクアーキテクチャ

多対一 (Many-to-one)（例：感情分類）:

- 入力: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$
- 出力: 最後の隠れ状態からの単一の $\hat{\mathbf{y}}$

一対多 (One-to-many)（例：画像キャプション生成）:

- 入力: 単一の $\mathbf{x}$

- 出力: $\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(T')}$

多対多 (Many-to-many)（例：機械翻訳）:

- 入力系列: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T_{\text{in}})}$
- 出力系列: $\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(T_{\text{out}})}$

15.2 Vanilla RNN の定義と順伝播

15.2.1 再帰的計算

各時刻 $t = 1, 2, \ldots, T$ において、Vanilla RNN は次を計算する：

$$\mathbf{z}_h^{(t)} = \mathbf{W}_{hh}\mathbf{h}^{(t-1)} + \mathbf{W}_{xh}\mathbf{x}^{(t)} + \mathbf{b}_h \tag{15.2}$$

$$\mathbf{h}^{(t)} = \sigma_h(\mathbf{z}_h^{(t)}) \tag{15.3}$$

$$\mathbf{z}_y^{(t)} = \mathbf{W}_{hy}\mathbf{h}^{(t)} + \mathbf{b}_y \tag{15.4}$$

$$\hat{\mathbf{y}}^{(t)} = \sigma_y(\mathbf{z}_y^{(t)}) \tag{15.5}$$

ここで：

- $\mathbf{W}_{hh} \in \mathbb{R}^{d_h \times d_h}$ （隠れ層-隠れ層重み）
- $\mathbf{W}_{xh} \in \mathbb{R}^{d_h \times d_x}$ （入力-隠れ層重み）
- $\mathbf{W}_{hy} \in \mathbb{R}^{d_y \times d_h}$ （隠れ層-出力重み）
- σ_h は通常 tanh または ReLU
- σ_y はタスクに依存する（分類ならソフトマックス、二値ならシグモイド、回帰なら線形）

15.2.2 時間を通じたパラメータ共有

RNN の重要な洞察は**パラメータ共有**である：すべてのタイムステップで同じパラメータ $(\mathbf{W}_{hh}, \mathbf{W}_{xh}, \mathbf{W}_{hy}, \mathbf{b}_h, \mathbf{b}_y)$ が使用される。これにより、モデルは可変長の系列を扱うことができる。

15.2.3 ベクトル化されたミニバッチ順伝播

ミニバッチの場合、隠れ状態と入力を行列として積み重ねる：

$$\mathbf{Z}_h^{(t)} = \mathbf{W}_{hh}\mathbf{H}^{(t-1)} + \mathbf{W}_{xh}\mathbf{X}^{(t)} + \mathbf{b}_h\mathbf{1}^\top \tag{15.6}$$

$$\mathbf{H}^{(t)} = \sigma_h(\mathbf{Z}_h^{(t)}) \tag{15.7}$$

$$\mathbf{Z}_y^{(t)} = \mathbf{W}_{hy}\mathbf{H}^{(t)} + \mathbf{b}_y\mathbf{1}^\top \tag{15.8}$$

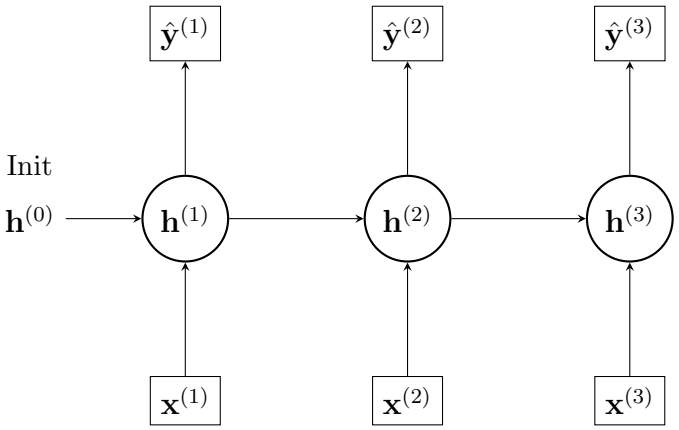
$$\hat{\mathbf{Y}}^{(t)} = \sigma_y(\mathbf{Z}_y^{(t)}) \tag{15.9}$$

ここで $\mathbf{1} \in \mathbb{R}^m$ は全要素が1のベクトルである。

15.3 計算グラフの時間展開

15.3.1 展開されたグラフ表現

RNN を T タイムステップにわたって展開すると、有向非巡回グラフ (DAG) である計算グラフが得られる：



時間展開 (Time Unfolding)

Figure 15.1: 展開された再帰型ニューラルネットワーク。隠れ状態 $\mathbf{h}^{(t)}$ は情報を次のタイムステップに伝える。(Goodfellow et al., 2016 より改変)

時刻 t における各「RNNセル」は以下に依存する：

1. 現在の入力 $\mathbf{x}^{(t)}$
2. 前の隠れ状態 $\mathbf{h}^{(t-1)}$

15.3.2 時間的依存性

隠れ状態 $\mathbf{h}^{(t)}$ は過去のすべての入力に依存する：

$$\mathbf{h}^{(t)} = f_t(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(t)}) \tag{15.10}$$

これは依存関係の長い連鎖を作り出し、勾配の流れを理解する上で重要となる。

15.4 通時的誤差逆伝播法 (BPTT)

15.4.1 損失関数と目的

1つの系列に対して、総損失は以下の通り：

$$L = \sum_{t=1}^{T} L^{(t)} \tag{15.11}$$

ここで $L^{(t)} = \ell(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)})$ は時刻 t における損失（例：分類のための交差エントロピー）である。

15.4.2 通時的誤差逆伝播アルゴリズム

各タイムステップにおける勾配は2つのソースから来るということが重要な洞察である：

$$\frac{\partial L}{\partial \mathbf{h}^{(t)}} = \frac{\partial L^{(t)}}{\partial \mathbf{h}^{(t)}} + \frac{\partial L^{(t+1:T)}}{\partial \mathbf{h}^{(t)}} \tag{15.12}$$

導出:
連鎖律と計算グラフからの勾配の流れにより：

$$\frac{\partial L^{(t+1:T)}}{\partial \mathbf{h}^{(t)}} = \frac{\partial L^{(t+1:T)}}{\partial \mathbf{h}^{(t+1)}} \frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} \tag{15.13}$$

$\boldsymbol{\delta}_h^{(t)} = \frac{\partial L}{\partial \mathbf{z}_h^{(t)}}$ を隠れ層の事前活性化におけるデルタ（誤差信号）とする。
出力層から：

$$\frac{\partial L^{(t)}}{\partial \mathbf{h}^{(t)}} = \mathbf{W}_{hy}^{\top} \frac{\partial L^{(t)}}{\partial \mathbf{z}_y^{(t)}} \tag{15.14}$$

次のタイムステップから（再帰接続経由）：

$$\frac{\partial L^{(t+1:T)}}{\partial \mathbf{h}^{(t)}} = \mathbf{W}_{hh}^{\top} \boldsymbol{\delta}_h^{(t+1)} \tag{15.15}$$

したがって：

$$\boldsymbol{\delta}_h^{(t)} = \left(\mathbf{W}_{hy}^{\top} \boldsymbol{\delta}_y^{(t)} + \mathbf{W}_{hh}^{\top} \boldsymbol{\delta}_h^{(t+1)} \right) \odot \sigma_h'(\mathbf{z}_h^{(t)}) \tag{15.16}$$

ここで $\boldsymbol{\delta}_y^{(t)} = \frac{\partial L^{(t)}}{\partial \mathbf{z}_y^{(t)}}$ は出力層のデルタである。

15.4.3 パラメータの勾配

重み行列に対する勾配は、すべてのタイムステップからの寄与を合計して計算される：

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{z}_h^{(t)}} \frac{\partial \mathbf{z}_h^{(t)}}{\partial \mathbf{W}_{hh}} \tag{15.17}$$

$$= \sum_{t=1}^{T} \boldsymbol{\delta}_h^{(t)} (\mathbf{h}^{(t-1)})^{\top} \tag{15.18}$$

同様に：

$$\frac{\partial L}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \boldsymbol{\delta}_h^{(t)} (\mathbf{x}^{(t)})^\top \tag{15.19}$$

$$\frac{\partial L}{\partial \mathbf{W}_{hy}} = \sum_{t=1}^{T} \boldsymbol{\delta}_y^{(t)} (\mathbf{h}^{(t)})^\top \tag{15.20}$$

バイアスに対して：

$$\frac{\partial L}{\partial \mathbf{b}_h} = \sum_{t=1}^{T} \boldsymbol{\delta}_h^{(t)} \tag{15.21}$$

$$\frac{\partial L}{\partial \mathbf{b}_y} = \sum_{t=1}^{T} \boldsymbol{\delta}_y^{(t)} \tag{15.22}$$

15.5 勾配消失と勾配爆発: 数学的解析

15.5.1 隠れ状態を通る勾配の流れ

離れた隠れ状態 $\mathbf{h}^{(k)}$（ここで $k < t$）に関する損失の勾配は、ヤコビアンの積を含む：

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}^{(j)}}{\partial \mathbf{h}^{(j-1)}} \tag{15.23}$$

導出:
連鎖律により：

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \frac{\partial \mathbf{h}^{(t-1)}}{\partial \mathbf{h}^{(t-2)}} \cdots \frac{\partial \mathbf{h}^{(k+1)}}{\partial \mathbf{h}^{(k)}} \tag{15.24}$$

各ヤコビアン $\frac{\partial \mathbf{h}^{(j)}}{\partial \mathbf{h}^{(j-1)}}$ は次の形式を持つ：

$$\frac{\partial \mathbf{h}^{(j)}}{\partial \mathbf{h}^{(j-1)}} = \frac{\partial \sigma_h(\mathbf{z}_h^{(j)})}{\partial \mathbf{z}_h^{(j)}} \frac{\partial \mathbf{z}_h^{(j)}}{\partial \mathbf{h}^{(j-1)}} \tag{15.25}$$

$$= \operatorname{diag}(\sigma_h'(\mathbf{z}_h^{(j)})) \mathbf{W}_{hh} \tag{15.26}$$

15.5.2 スペクトル解析

安定性のために、スペクトル特性を解析する。ヤコビアンの積は以下で近似できる：

$$\left\| \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}^{(j)}}{\partial \mathbf{h}^{(j-1)}} \right\| \lesssim \|\sigma_h'\|_\infty^{t-k} \|\mathbf{W}_{hh}\|^{t-k} \tag{15.27}$$

tanh 活性化の場合、すべての z に対して $|\sigma_h'(z)| \leq 1$ であり、最大値は $z = 0$ で 1 である。

$\rho = \lambda_{\max}(\mathbf{W}_{hh})$ をスペクトル半径（最大固有値の大きさ）とする。

- **勾配消失 (Vanishing gradients):** もし $\rho < 1$ ならば、$t - k \to \infty$ につれて $\|\mathbf{W}_{hh}\|^{t-k} \to 0$ となる。
  - 結果：離れたタイムステップの勾配が無視できるほど小さくなる。
  - 長期依存性の学習が遅くなる。
- **勾配爆発 (Exploding gradients):** もし $\rho > 1$ ならば、$t - k \to \infty$ につれて $\|\mathbf{W}_{hh}\|^{t-k} \to \infty$ となる。
  - 結果：勾配が非有界になり、学習が不安定になる。
  - パラメータ更新が非常に大きくなり、発散を引き起こす可能性がある。

15.5.3 安定性のための数学的条件

時間的条件数 (temporal condition number) を定義する：

$$\kappa_T = \rho^T \tag{15.28}$$

長い系列（T が大きい）の場合：

- $\rho < 1$ ならば：$\kappa_T \to 0$ の指数関数的減衰（消失）
- $\rho > 1$ ならば：$\kappa_T \to \infty$ の指数関数的成長（爆発）
- $\rho = 1$ ならば：$\kappa_T = 1$（臨界的にバランスが取れているが、実務上は不安定）

15.6 よくある質問 (RNN)

15.6.1 Q1: なぜ $\mathbf{W}_{hh}$ を共有するのか？

A: パラメータ共有により、RNN は系列長に関係なく同じ「ルール」を適用できるからである。

例:

- **共有なし**: 長さ 100 の系列と長さ 1000 の系列で異なるネットワーク（異なるパラメータ）が必要になる。
- **共有あり**: 同じ $\mathbf{W}_{hh}$ が時刻 1 から 100 まで、また 100 から 1000 まで使用される。

数学的視点:

$$\mathbf{h}^{(T)} = \sigma(\mathbf{W}_{hh} \cdots \sigma(\mathbf{W}_{hh}\mathbf{h}^{(1)})) \tag{15.29}$$

$\mathbf{W}_{hh}$ を繰り返し適用することで、モデルは**可変長の系列**を扱うことができる。
トレードオフ: 勾配が $\mathbf{W}_{hh}$ の長い積になり、消失・爆発しやすくなる。

15.6.2 Q2: 「勾配消失」とは正確には何か？

A: パラメータ更新がほぼゼロになり、モデルが学習を停止する状態である。

数値例:

- Vanilla RNN で 100 ステップの文を処理する。
- 各ステップで $|\sigma'| \approx 0.5$（tanh の中程度の勾配）。
- 勾配の大きさ：$0.5^{100} \approx 10^{-30}$（ほぼゼロ !）。

実用上の影響:

- 100番目の出力に対する最初の単語の影響が測定不能になる。
- モデルは「最近の単語」のみに依存し、長期依存関係を学習できない。

例: 翻訳タスク

```
``The quick brown fox jumps over the lazy dogs are \_\_\_''
      ^ 主語（遠距離）                          ^ 述語（末尾）
```

RNN は局所的な文脈のみを使って「dogs are」を補完しようとし、「fox」との単数・複数の一致を見逃す。

15.6.3 Q3: BPTT の計算コストは？

A: 完全な BPTT はすべてのタイムステップの状態を保存する必要があり、メモリ効率が悪い。

メモリ使用量:

- 系列長 $T = 1000$、隠れ次元 $d_h = 512$、Float32 (4バイト)。
- **必要メモリ**: $1000 \times 512 \times 4 = 2.048$ MB（系列あたり）。
- **バッチサイズ 32**: 65.5 MB。

勾配計算のためにデータを全ステップにわたって保持しなければならないため、メモリを大量に消費する。

解決策: 打ち切り BPTT (Truncated BPTT) ($\tau \approx 50$)、過去 50 ステップのみを考慮する。

15.6.4 Q4: なぜ RNN は並列化できないのか？

A: $\mathbf{h}^{(t)}$ が $\mathbf{h}^{(t-1)}$ に依存するため、計算は時系列順序を守らなければならないからである。

依存グラフ:

```
h(1) -> h(2) -> h(3) -> h(4)
 |       |       |       |
x(1)    x(2)    x(3)    x(4)
```

計算時間:

- RNN: $O(T)$（逐次的）。
- Transformer: $O(\log T)$ または $O(1)$（並列可能）。

LLM においては、並列効率が学習速度を決定するため、Transformer が圧倒的に有利である。

15.7 よくある質問（勾配の問題）

15.7.1 Q5: 勾配爆発の危険性は？

A: 更新が巨大になり、パラメータが最適解を飛び越えてしまうことである。

数値例:

```
# 勾配爆発
gradient = 1e8
learning_rate = 0.001
weight_update = 100000  # 巨大な更新！

# 通常
gradient = 1.0
weight_update = 0.001   # 安定
```

損失曲線への影響: 収束する代わりに、損失が激しく振動したり NaN に発散したりする。

解決策:

1. **勾配クリッピング**: もし $\|\mathbf{g}\| > \theta$ ならば $\mathbf{g} \leftarrow \theta \frac{\mathbf{g}}{\|\mathbf{g}\|}$。
2. **重み初期化**: $\|\mathbf{W}_{hh}\|$ を小さく保つ。

15.7.2 Q6: なぜスペクトル半径が重要なのか？

A: $\mathbf{W}_{hh}$ の最大固有値 ρ が勾配の成長/減衰率を決定するからである。

直感:

- $\rho = 0.9$: 勾配 $\approx 0.9^{100} \approx 0$（消失）。
- $\rho = 1.0$: 勾配 ≈ 1（臨界境界）。
- $\rho = 1.1$: 勾配 $\approx 1.1^{100} \approx 14000$（爆発）。

含意: スペクトル半径が妥当になるように重みを初期化する（例：Xavier 初期化、直交初期化）。

Chapter 16

長・短期記憶 (LSTM)

16.1 動機と設計原理

16.1.1 Vanilla RNN の問題点

核心的な問題は、勾配が一連の行列乗算を通して流れなければならないことである：

$$\frac{\partial L}{\partial \mathbf{h}^{(1)}} \propto \prod_{t=2}^{T} \mathbf{W}_{hh}^{\top} \mathrm{diag}(\sigma_h') \tag{16.1}$$

$\sigma_h = \tanh$ の場合：

- ピーク時、$|\sigma_h'(0)| = 1$
- ピークから離れると、$|\sigma_h'(z)| < 1$、多くの場合 ≈ 0.1 から 0.01

100ステップの系列の場合、各ステップで $|\sigma_h'| \approx 0.9$ であったとしても：

$$0.9^{100} \approx 2.7 \times 10^{-5} \tag{16.2}$$

勾配は深刻に消失する。

16.1.2 LSTM の解決策: 加法的な状態更新

乗法的な更新 $\mathbf{h}^{(t)} = \sigma_h(\mathbf{W}_{hh}\mathbf{h}^{(t-1)} + \ldots)$ の代わりに、LSTM は**加法的な状態更新**を使用する：

$$\mathbf{c}^{(t)} = \mathbf{c}^{(t-1)} + (\text{新しい何か}) \tag{16.3}$$

ここで $\mathbf{c}^{(t)}$ は**セル状態**（内部メモリ）である。
重要な洞察: セル状態を通る勾配は以下のようになる：

$$\frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{c}^{(t-1)}} = \mathbf{f}^{(t)} \tag{16.4}$$

ここで $\mathbf{f}^{(t)}$ は**忘却ゲート**である。もし $\mathbf{f}^{(t)} \approx 1$ ならば：

$$\prod_{j=k}^{t} \frac{\partial \mathbf{c}^{(j)}}{\partial \mathbf{c}^{(j-1)}} = \prod_{j=k}^{t} \mathbf{f}^{(j)} \approx 1 \tag{16.5}$$

積は安定（1に近い）に保たれ、勾配消失を防ぐ！

16.2 完全な LSTM セルの定義

16.2.1 ゲート計算

忘却ゲート (Forget Gate):

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_{xf}\mathbf{x}^{(t)} + \mathbf{W}_{hf}\mathbf{h}^{(t-1)} + \mathbf{b}_f) \tag{16.6}$$

ここで $\sigma(z) = \frac{1}{1+e^{-z}}$ はシグモイド（出力は $[0, 1]$）である。

入力ゲート (Input Gate):

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_{xi}\mathbf{x}^{(t)} + \mathbf{W}_{hi}\mathbf{h}^{(t-1)} + \mathbf{b}_i) \tag{16.7}$$

出力ゲート (Output Gate):

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_{xo}\mathbf{x}^{(t)} + \mathbf{W}_{ho}\mathbf{h}^{(t-1)} + \mathbf{b}_o) \tag{16.8}$$

セル状態候補 (Tanh 事前活性化):

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_{xc}\mathbf{x}^{(t)} + \mathbf{W}_{hc}\mathbf{h}^{(t-1)} + \mathbf{b}_c) \tag{16.9}$$

16.2.2 状態と隠れ状態の更新

セル状態の更新（加法的、ここが重要）:

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)} \tag{16.10}$$

ここで $\odot$ は要素ごとの乗算を表す。

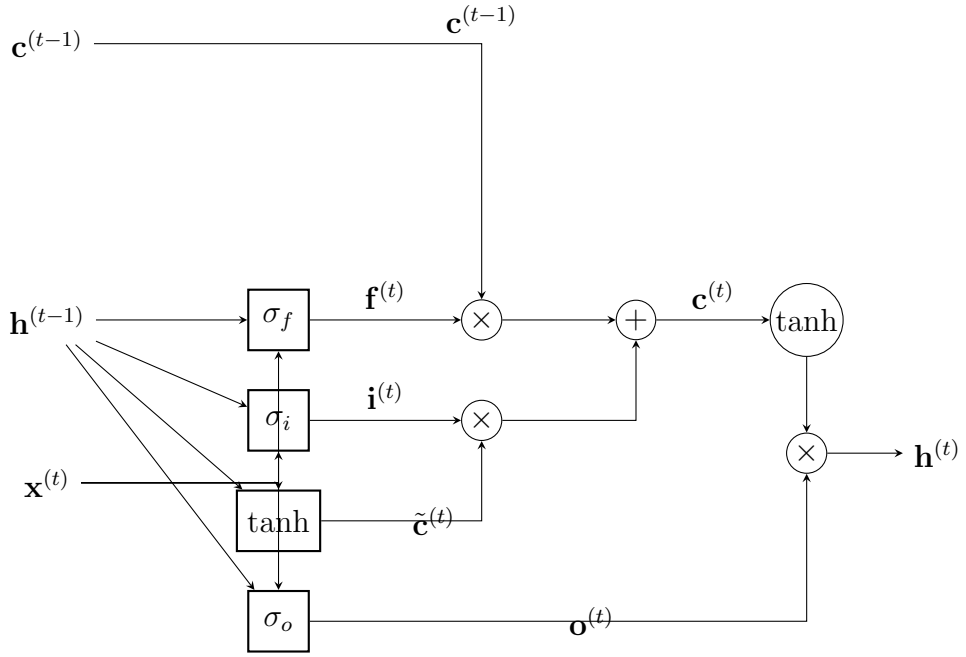


Figure 16.1: LSTM セルの構造。セル状態 $\mathbf{c}$ は上部の「高速道路」に沿って走り、忘却・入力ゲートを介して線形に相互作用する。(Olah, 2015 より改変)

隠れ状態の出力:

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}) \tag{16.11}$$

16.3 概念的直感: ノートブックの例え

LSTM ゲート (c, f, i, o) の役割は、それらを**ノートブック（セル状態）に対する3つの操作**と考えると自然に理解できる。

1. c を「内部ノートブック」と考える

- $\mathbf{c}^{(t)}$ は**セル状態**: 長期間にわたって重要な情報を保持するノートブック。
- ゲート (f, i, o) は厳密にこのノートブックを制御する**動詞（操作）**である。

2. 3つの操作（動詞）

順序を覚える: **削除** → **書き込み** → **公開**。

1. **忘却ゲート (f_t): 赤ペン（編集者）。**
   - 前のノートブック ($\mathbf{c}^{(t-1)}$) から何を削除するかを決める。
   - 例: 「主語が変わった」 → 古い主語を消す。
2. **入力ゲート (i_t): 採用担当者。**
   - 新しいドラフト ($\tilde{\mathbf{c}}^{(t)}$) のどれだけを「採用」してノートブックに書き込むかを決める。
   - 例: 「これは新しい主語だ」 → 強調して書き込む。
3. **出力ゲート (o_t): 広報担当者（スポークスマン）。**
   - ノートブック $\mathbf{c}^{(t)}$ にはすべてが含まれている（生の情報や雑多なものも）。
   - それを整形 (tanh) し、外部の世界 ($\mathbf{h}^{(t)}$) に何を公開するかを決める。
   - 例: 「動詞は主語と一致する必要がある」 → 単数/複数フラグを出力する。

3. 合成ルール: 「加重和」

核となる更新ルールは：

$$\mathbf{c}^{(t)} = \underbrace{\mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)}}_{\text{残存する古いメモ}} + \underbrace{\mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}}_{\text{受け入れられた新しいメモ}} \tag{16.12}$$

なぜ乗算 ($\odot$) と加算を使うのか？

- **乗算 ($\odot$) はバルブ**として機能する。0.0 は流れを遮断（削除）、1.0 は通過させる（保持）。
- **加算 (+) は残りの履歴と新しい情報を自然に重ね合わせる。**

4. シグモイド vs Tanh

どの活性化関数を使うかどうやって覚えるか？

- シグモイド (σ): 0 から 1 を出力。**確率や比率**（ノブ/ゲート）に使用。
- **Tanh**: -1 から 1 を出力。**コンテンツや特徴**（ドラフト/状態）に使用。

16.4 よくある質問 (LSTM)

16.4.1 Q1: なぜセル状態の勾配は消失しないのか？

A: 加法的な接続 ($\mathbf{c}^{(t)} = \mathbf{c}^{(t-1)} + \ldots$) により、勾配は乗算の連鎖ではなく**加算のパス**を通って流れるからである。

比較:

- **Vanilla RNN**: $\partial L/\partial h^{(1)} \propto \mathbf{W}_{hh}^{100}$ (指数関数的減衰)。
- **LSTM**: $\partial L/\partial c^{(1)} \propto \sum f^{(t)}$ (総和)。

もし忘却ゲート $f^{(t)} \approx 1$ ならば、勾配は時間を通じて 1 倍され続け、消失を防ぐ。

16.4.2 Q2: 本当に4つのゲートが必要か？

A: 理論的に最小限のモデルは存在する（例：2-3ゲートの GRU）が、4つのゲートは安定性と明確な役割を提供する。

- **忘却**: 過去を忘れることを制御（≈ 1 で記憶）。
- **入力**: 新しい情報の書き込みを制御。
- **出力**: 情報の読み出しを制御。
- **候補**: 新しいコンテンツ自体。

GRU との比較: GRU は入力/忘却を更新ゲートに、出力をリセットゲートに統合し、効率は良いが解釈性はわずかに劣る。

16.4.3 Q3: なぜ忘却ゲートを 1 で初期化するのか？

A: 訓練開始時にモデルがデフォルトで**過去の情報を保存**するようにするためである。

バイアス初期化:

$$b_f = 1.0 \implies \sigma(1.0) \approx 0.73 \tag{16.13}$$

$b_f = 0$ だと忘却率が 50% から始まり、訓練初期に勾配消失を引き起こす。1 に初期化することで、勾配が即座に時間を通じて流れるようになる。

16.4.4 Q4: セル状態と隠れ状態の違いは？

A:

- **セル状態** $\mathbf{c}^{(t)}$: 長期記憶、安定的、加法的に更新される。理想的にはゆっくり変化する。
- **隠れ状態** $\mathbf{h}^{(t)}$: 短期作業メモリ、次の層への出力、変動が激しい。

例: 「犬たちが走っている...」

- $\mathbf{c}^{(t)}$ は「複数形の主語」という特徴を維持する。
- $\mathbf{h}^{(t)}$ は特定の現在の単語「いる (are)」を示す。

Chapter 17

系列対系列モデルと注意機構

17.1　エンコーダ-デコーダアーキテクチャ

17.1.1　動機

標準的な RNN は入力系列を同じ長さの出力系列（または単一の出力）にマッピングする。機械翻訳のような多くのタスクは、入力系列 $\mathbf{x}^{(1:T)}$ を異なる長さ T' の出力系列 $\mathbf{y}^{(1:T')}$ にマッピングする。

エンコーダ-デコーダ (Encoder-Decoder) アーキテクチャはこれを次のように解決する：

1. **エンコーダ**: 入力系列を固定長の「コンテキストベクトル」 $\mathbf{c}$ に処理する。
2. **デコーダ**: $\mathbf{c}$ を条件として出力系列を生成する。

17.1.2　固定長コンテキストベクトル

通常、エンコーダ RNN の最後の隠れ状態がコンテキストとして使用される：

$$\mathbf{c} = \mathbf{h}_{\text{enc}}^{(T)} \tag{17.1}$$

デコーダは $\mathbf{s}^{(0)} = \mathbf{c}$ で初期化され、自己回帰的にトークンを生成する。

17.2　注意機構 (Attention Mechanism)

17.2.1　ボトルネック問題

長い文を単一のベクトル $\mathbf{c}$ にエンコードすると、**情報のボトルネック**が生じる。デコーダは $\mathbf{c}$ だけで全体の意味を再構築しなければならない。

17.2.2　アテンション重み

アテンションにより、デコーダは各ステップでエンコーダの状態を「振り返る」ことができる。デコーダのステップ t とエンコーダの状態 $\mathbf{h}_s$ $(s = 1 \dots T)$ に対して：

$$\alpha_{t,s} = \frac{\exp(\text{score}(\mathbf{s}_t, \mathbf{h}_s))}{\sum_{k=1}^{T} \exp(\text{score}(\mathbf{s}_t, \mathbf{h}_k))} \tag{17.2}$$

コンテキストベクトルは動的になる：

$$\mathbf{c}_t = \sum_{s=1}^{T} \alpha_{t,s} \mathbf{h}_s \tag{17.3}$$

17.3 よくある質問 (Seq2Seq & Attention)

17.3.1 Q1: 固定コンテキストベクトルの限界は？

A: 長い文を小さなベクトルに圧縮することは**情報理論的に困難**である。

ボトルネック理論: 入力（高情報量）→ コンテキストベクトル（例：512次元）→ 出力。 30単語（∼9000次元）を512次元に圧縮するとニュアンスが失われる。 アテンションはソース状態を直接評価することでこれを解決する。

17.3.2 Q2: アテンション重みは何を意味するか？

A: それらは時刻 t において「どこを見るべきか」の確率分布を表す。

例: 「The cat sat」を「Le chat...」に翻訳する。

- 「Le」を生成（アテンション経由）→ ‘The‘ (0.9), ‘cat‘ (0.1)。
- 「chat」を生成（アテンション経由）→ ‘cat‘ (0.8)。

17.3.3 Q3: どのアテンションスコアが最適か？

A: スケール化ドット積 **(Scaled Dot-Product)** が標準的な勝者である。

- **ドット積**: $\mathbf{s}^T\mathbf{h}$ (高速)。
- **加法**: $\mathbf{v}^T \tanh(...)$ (柔軟)。
- **スケール化ドット積**: $\frac{\mathbf{s}^T\mathbf{h}}{\sqrt{d}}$ (高速かつ安定)。

17.3.4 Q4: なぜマルチヘッドアテンションなのか？

A: 複数の種類の関係性を同時に捉えるためである。

シングルヘッド: 1つの支配的なパターン（例：「主語-動詞」）にしか焦点を当てられない。 **マルチヘッド:**

- ヘッド 1: 長距離依存に焦点を当てる。
- ヘッド 2: 直近の隣接関係に焦点を当てる。
- ヘッド 3: 文法的役割に焦点を当てる。

並列処理により、これらすべての「視点」を一度に学習できる。

Chapter 18

Transformer アーキテクチャ

18.1 自己注意機構 (Self-Attention Mechanism)

18.1.1 Query, Key, Value 射影

埋め込みの系列 $\mathbf{X} \in \mathbb{R}^{T \times d_{\text{model}}}$ が与えられたとき：

Query, Key, Value への射影:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{Q} \in \mathbb{R}^{T \times d_k} \tag{18.1}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{K} \in \mathbb{R}^{T \times d_k} \tag{18.2}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}_V, \quad \mathbf{V} \in \mathbb{R}^{T \times d_v} \tag{18.3}$$

ここで：

- $\mathbf{W}_Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$ (Query 射影)
- $\mathbf{W}_K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ (Key 射影)
- $\mathbf{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ (Value 射影)

通常、$d_k = d_v = d_{\text{model}}/h$ である。ここで h はアテンションヘッドの数。

18.1.2 スケール化ドット積アテンション (Scaled Dot-Product Attention)

アテンションスコア（正規化前の類似度）:

$$\mathbf{E} = \mathbf{Q}\mathbf{K}^\top, \quad \mathbf{E} \in \mathbb{R}^{T \times T} \tag{18.4}$$

$\sqrt{d_k}$ でスケーリング:

$$\mathbf{E}_{\text{scaled}} = \frac{\mathbf{E}}{\sqrt{d_k}} \tag{18.5}$$

ソフトマックス正規化:

$$\mathbf{A} = \text{softmax}(\mathbf{E}_{\text{scaled}}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \tag{18.6}$$

アテンション出力:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{A}\mathbf{V} \tag{18.7}$$

18.1.3 なぜ $\sqrt{d_k}$ でスケーリングするのか？

ランダムベクトル $\mathbf{q}, \mathbf{k} \sim \mathcal{N}(0,1)^{d_k}$ に対して：

$$\text{Var}[\mathbf{q}^\top \mathbf{k}] = d_k \tag{18.8}$$

d_k が大きい場合、ドット積の分散は大きくなる。$\sqrt{d_k}$ でスケーリングすることで、アテンション分布が極端に鋭くなる（勾配消失を引き起こす）のを防ぎ、分散を維持する。

18.2 マルチヘッドアテンション (Multi-Head Attention)

18.2.1 複数のアテンションヘッド

h 個のアテンションヘッド（通常 $h = 8$ または $h = 12$）に対して：

各ヘッド i のアテンションを計算:

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i \tag{18.9}$$

ヘッドを連結 (Concatenate):

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O \tag{18.10}$$

ここで $\mathbf{W}_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ は出力射影である。

18.3 位置エンコーディング (Positional Encoding)

18.3.1 正弦波位置エンコーディング

位置 pos と埋め込み次元 i に対して：

$$PE(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \tag{18.11}$$

$$PE(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \tag{18.12}$$

性質:

- すべての値が $[-1, 1]$ の範囲内
- 相対位置情報は加法定理を通じてエンコードされる
- 階層的な波長構造により、距離の学習が可能

18.4 Transformer エンコーダブロック

18.4.1 ブロック構造

1. マルチヘッド自己注意機構 **(Multi-Head Self-Attention)**
2. 加算と正規化 (残差接続 + レイヤー正規化)
3. 位置毎の順伝播ネットワーク **(Position-wise Feed-Forward Network)**
4. 加算と正規化

18.4.2 順伝播ネットワーク

$$\mathbf{Z}_{\text{ffn}} = \max(0, \mathbf{Y}_{\text{attn}}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{18.13}$$

ここで通常 $d_{\text{ffn}} = 4 \times d_{\text{model}}$ である。

18.4.3 レイヤー正規化

$$\text{LayerNorm}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\text{Var}[\mathbf{x}] + \epsilon}} + \beta \tag{18.14}$$

バッチサイズに関係なく、行ごとに適用される。

18.5 Transformer デコーダブロック

18.5.1 3つのサブレイヤー

1. マスク付きマルチヘッド自己注意機構 (ターゲットは現在より前の位置のみを参照可能)
2. マルチヘッド交差注意機構 **(Cross-Attention)** (ターゲットがエンコーダ出力を参照)
3. 位置毎の順伝播ネットワーク

18.5.2 マスク付きアテンション

18.5.3 マスク付きアテンション

自己注意機構におけるソフトマックスの前：

$$\mathbf{E}_{\text{masked}} = \mathbf{E} + \mathbf{M} \tag{18.15}$$

ここで：

$$M_{i,j} = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases} \tag{18.16}$$

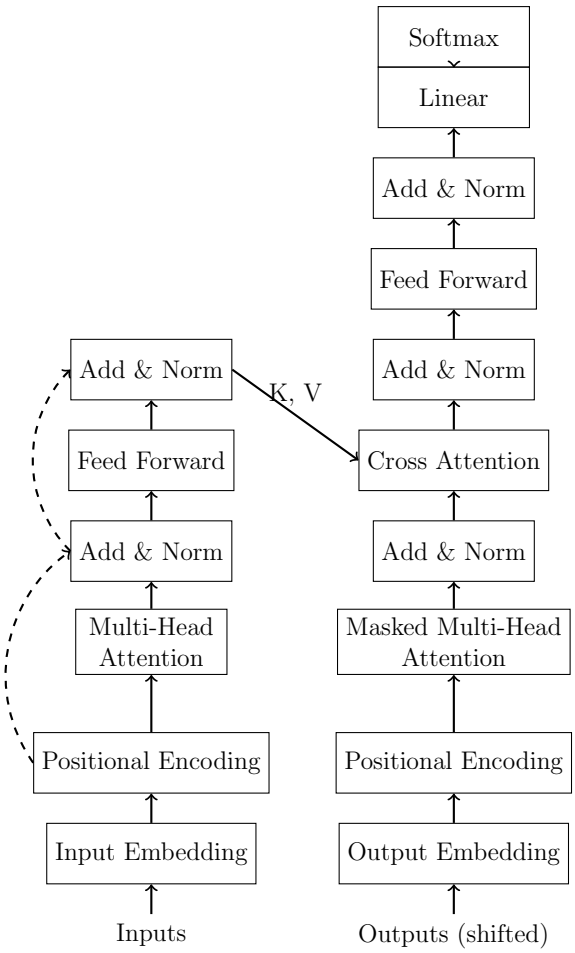


Figure 18.1: Transformer アーキテクチャ。左: エンコーダブロック。右: デコーダブロック。(Vaswani et al., 2017 に基づく)

18.6 概念的直感: 会議室の例え

Transformer（エンコーダ-デコーダ）は、参加者全員が同じ部屋に座り、互いの発言を参照しながら個別のメモを作成する**「会議 + 書記」システム**として想像するのが最良である。

18.6.1 1. 入力: 参加者と座席

- **トークン**: 会議の参加者（単語）。系列長 T は座席数。
- **埋め込み**: 各参加者の「名札」。ID を意味ベクトルに変換する。
- **位置エンコーディング**: 「座席表」。全員が同時に話す（並列性）ため、順序を知るために「私は座席 #1 に座っている」という情報を名札に明示的に追加する必要がある。

18.6.2 2. 自己注意機構: Q-K-V メカニズム

各参加者（トークン）が3つのアイテムを持っていると想像する：

1. **Query (Q): 「私が知りたいこと」**（他の参加者への質問票）。

2. **Key (K):** 「私が提供できること」（トピックを要約したインデックスカード）。
3. **Value (V):** 「私の実際の内容」（共有されるべき詳細情報）。

プロセス:

- **ステップ 1 (マッチング)**: 参加者 A が Query (Q_A) を放送する。それが全員の Keys (K) と比較される。
- **ステップ 2 (重み付け)**: もし Q_A が K_B とよく一致すれば（ドット積）、A は B に 90% の注意を払う。もし K_C とのマッチが悪ければ、1% しか注意を払わない。
- **ステップ 3 (収集)**: A はこれらのマッチスコアに基づいて、全員の Values (V) の加重和を取り、「要約メモ」を作成する。

18.6.3 3. マルチヘッドとマスキング

- **マルチヘッドアテンション**: 異なる「思考セット（マインドセット）」で同時に会議を行う。ヘッド 1 は文法（主語-動詞）をチェックし、ヘッド 2 は文脈（代名詞解決）をチェックする。これらは並列に実行される。
- **マスク付きアテンション (デコーダ)**: 書記が未来の発言者を見るのを防ぐ「目隠し」。時刻 t の議事録を書くとき、発言者 $t+1$ が何を言うかをカンニングしてはならない。

18.6.4 4. 構成要素

- **順伝播ネットワーク (FFN)**: 「個別の消化」。他者から情報を集めた（アテンション）後、各参加者は自分のノートに独自の解釈を独立して書き込む。
- **残差接続**: 「ショートカット」。以前のメモを取り、新しい洞察を加える。これにより、元々知っていたことを忘れるのを防ぐ。
- **レイヤー正規化**: 「主催者（オーガナイザー）」。学習プロセスを安定させるために、メモのスケールを標準化する。

18.7 よくある質問 (Transformer)

18.7.1 Q1: なぜ自己注意機構は RNN の問題を解決するのか？

A: 直接的な接続により、任意の2つのトークン間のパス長が 1 に短縮されるからである。

パス長の比較:

- **RNN**: 位置 $1 \to 2 \to \cdots \to 100$。パス長 $= 99$。
- **自己注意機構**: 位置 $1 \leftrightarrow$ 位置 100 (直接)。パス長 $= 1$。

計算複雑性:

- RNN: $O(T \cdot d^2)$ (逐次的)。
- Transformer: $O(T^2 \cdot d + T \cdot d^2)$ (並列)。

18.7.2 Q2: Query, Key, Value の違いは？

A: 図書館の検索システムを考えるとよい。

- **Query**: 「ニューラルネットワークに関する本は？」 (検索意図)。
- **Key**: 本のタイトル/カテゴリ (マッチング対象の特徴)。
- **Value**: 実際の本の内容 (取得されるコンテンツ)。

メカニズム: Query と Key をマッチング → マッチ強度で重み付けして Value を取得。

18.7.3 Q3: なぜ $\sqrt{d_k}$ でスケーリングするのか？

A: スケーリングがないと、次元とともにドット積が増大し、ソフトマックスを勾配消失領域に押しやってしまうからである。

シミュレーション ($d_k = 512$):

- **スケーリングなし**: ドット積の分散 ≈ 512。値の範囲 ± 30。ソフトマックスはワンホットになる（飽和）。
- **スケーリングあり**: Dot/$\sqrt{512}$。分散 ≈ 1。値の範囲 ± 3。ソフトマックスは適切に振る舞う。

18.7.4 Q4: 正弦波位置エンコーディング vs 学習可能位置エンコーディング？

A:

- **正弦波**: 訓練中に見たことのない系列長まで外挿できる可能性がある。(オリジナルの Transformer で使用)。
- **学習可能**: 訓練限界を超える位置は扱えない。(BERT, GPT で使用)。

可変/外挿可能なコンテキスト長が必要な場合、正弦波が好まれる。

18.7.5 Q5: レイヤー正規化 vs バッチ正規化？

A: 可変長の系列にはレイヤー正規化が適している。

- **バッチ正規化**: バッチ次元に沿って正規化する。バッチ統計量への依存は NLP では問題となる。
- **レイヤー正規化**: 各トークンごとに特徴次元に沿って独立して正規化する。RNN/Transformer で安定している。

18.7.6 Q6: マスク付きアテンションとは？

A: デコーダにおいて、訓練中にモデルが未来のトークンを見て「カンニング」する（Teacher Forcing）のを防ぐものである。未来の位置のアテンションスコアを $-\infty$ に設定することで因果律を強制する。

18.7.7 Q7: Transformer は本当に RNN より速いのか？

A: 訓練か推論かによる。答えは並列性と、系列長 T に対する行列演算のスケーリングにある。

1. 訓練 (並列): Transformer の方が速い

- **Transformer**: 複雑度は層あたり $O(T^2 \cdot d)$。重要なのは、アテンション機構が T 個すべてのトークンの相互作用を単一の大きな行列乗算として同時に計算することである。時間次元における逐次的な依存関係がなく、GPU 上での大規模な並列化が可能である。

- **RNN**: 複雑度は $O(T \cdot d^2)$。見かけ上 T に線形だが、隠れ状態 $h_t = \phi(Wh_{t-1} + \dots)$ が h_{t-1} に依存する。これにより逐次的な計算（実時間 $\propto T$）が強制され、時間方向の並列化ができない。

2. 推論 (自己回帰): Transformer は遅くなり得る 生成は本質的に逐次的 $(t = 1 \to T)$ である。

- **RNN**: $O(T \cdot d^2)$。1トークンを生成するには、固定サイズの h_t を更新する（定数コスト）。長さ T の総コストは線形：$O(T)$。

- **Transformer (ナイーブ)**: $O(T^3 \cdot d)$。各ステップで成長するプレフィックスに対してアテンションを再計算すると、ステップあたり二次のコストがかかり、総コストは三次になる。

- **Transformer (KV キャッシュ)**: $O(T^2 \cdot d)$。以前の Key と Value をキャッシュすることで、各ステップは過去にのみ注意を払う $(O(t \cdot d))$。総コストの合計は二次：$\sum_{t=1}^{T} O(t \cdot d) \approx O(T^2 \cdot d)$。

結論: Transformer は大量のデータに対する高速な並列訓練に最適化されている。推論には効率を保つための工夫（KV キャッシュなど）が必要であってもである。

Chapter 19

スケーリング則と基盤モデル (Foundation Models)

19.1 スケーリング則 (Scaling Laws)

19.1.1 経験則

言語モデルの性能（損失 L）は、計算量 (C)、データセットサイズ (N)、およびパラメータ数 (P) に対してべき乗則に従ってスケーリングする：

$$L(N) \propto N^{-\alpha_N}, \quad L(P) \propto P^{-\alpha_P}, \quad L(C) \propto C^{-\alpha_C} \tag{19.1}$$

通常 $\alpha \approx 0.05$ から 0.1 である。これは単にリソースを拡大するだけで性能が予測可能に向上することを意味し、より大きなモデル（「基盤モデル」）への競争を加速させている。

19.2 インコンテキスト学習 (In-Context Learning)

大規模モデルは、パラメータ更新なしにプロンプト内の例から学習する能力を示す。

ゼロショット (Zero-shot):

```
Translate to French: ‘‘Hello’’ ->
```

フューショット/インコンテキスト (Few-shot/In-Context):

```
Translate to French:
‘‘Cat’’ -> ‘‘Chat’’
‘‘Dog’’ -> ‘‘Chien’’
‘‘Hello’’ ->
```

モデルは文脈からタスクのルールを推論する。

19.3 よくある質問 (基盤モデル)

19.3.1 Q1: スケーリング則は永遠に続くのか？

A: 現在 $\sim 10^{24}$ FLOPs までは検証されているが、限界は存在する。

1. **データの飽和**: インターネット上の高品質なテキストは有限である。
2. **計算コスト**: 指数関数的なコスト増大。
3. **既約誤差**: 言語がどれだけ予測可能かには限界がある（エントロピー）。

19.3.2 Q2: なぜ創発的能力 (Emergent Abilities) が起こるのか？

A: いくつかの仮説が存在する：

- **相転移**: パラメータの臨界量により、複雑なヒューリスティクスが突然形成される。
- **構造化された潜在空間**: より大きなモデルは、転移を可能にする階層的に安定した表現を学習する。
- **学習することを学習する (Learning to Learn)**: モデルは推論中にアテンションのダイナミクスのみを通じて、勾配降下法のような適応を実行することを学習する（インコンテキスト学習）。

19.3.3 Q3: インコンテキスト学習はどのように機能するのか？

A: 本質的には**メタ学習 (Meta-Learning)** である。事前学習中、モデルは「タイトル → 本文」や「質問 → 回答」のような文書構造を見る。モデルは「タスクパターン」 → 「出力」を認識することを学習する。メカニズムとしては、Induction Heads（特別なアテンションヘッド）が前の文脈からパターンをコピーする。

19.3.4 Q4: 何が「基盤」モデルを作るのか？

A: 幅広いデータで訓練された単一のモデルであり、多くの下流タスクに適応（ファインチューニング）できるものである。

- **従来**: タスクごとに1つのモデルを訓練（翻訳用に1つ、感情分析用に1つ）。
- **基盤**: 1つの巨大なモデルを事前学習 → 翻訳、感情分析、コーディングなどに適応。

広範なアプリケーションの「基盤（foundation）」として機能する。

19.4 要約表: アーキテクチャ比較

特徴	RNN	LSTM	Transformer
並列化	なし (逐次)	なし (逐次)	**あり (完全)**
勾配消失	深刻	軽微	なし
メモリ	低	中	高 ($O(T^2)$)
推論速度 (長 T)	**高速**	**高速**	低速
訓練速度	低速	低速	**高速**
解釈可能性	中	高	低 (アテンション)
実装	容易	普通	困難

Chapter 20

Transformer の派生形と現代的アーキテクチャ

20.1 Transformer の進化の概要

20.1.1 年表と動機

標準的な Transformer は、特定の制限に対処するための多数の派生形を生み出した：

年	アーキテクチャ	主な革新	動機
2017	Transformer	自己注意 + 並列化	ベースライン
2018	BERT	双方向 + MLM	より良い理解
2018	GPT	デコーダのみ + スケール	より良い生成
2019	RoBERTa	BERT の改善	より強力なエンコーダ
2019	Longformer	疎なアテンション	長い系列
2020	T5	エンコーダ-デコーダ統一	統一フレームワーク
2021	ViT	画像をパッチとして扱う	テキスト以外のドメイン
2022	PaLM	大規模デコーダのみ	540B へのスケーリング
2023	LLaMA	効率的なデコーダ	オープンソース代替
2024	Mixtral	MoE 派生形	疎な専門家 (Experts)

20.2 エンコーダのみ: BERT とその派生形

20.2.1 BERT アーキテクチャ

BERT (Bidirectional Encoder Representations from Transformers) は、マスク付き言語モデリング (MLM) で事前学習された双方向エンコーダスタックを使用する。

20.2.2 マスク付き言語モデリング (MLM)

順伝播: 入力トークンの 15% をマスクし、それらを予測する。

$$L_{\text{MLM}} = -\sum_{i \in \text{masked}} \log P(\text{token}_i | \text{context}) \tag{20.1}$$

これはモデルに双方向の文脈を使用して欠落情報を推論することを強制し、豊かな言語的特徴を学習させる。

20.3 デコーダのみ: GPT とその派生形

20.3.1 GPT アーキテクチャ

Generative Pre-trained Transformer (GPT) は、因果マスキング (causal masking) を持つデコーダのみのスタックを使用し、**次トークン予測 (Next Token Prediction)** のために訓練される。

20.3.2 因果言語モデリング

$$L_{\text{CLM}} = -\sum_{t=1}^{T} \log P(\text{token}_{t+1} | \text{token}_{1:t}) \tag{20.2}$$

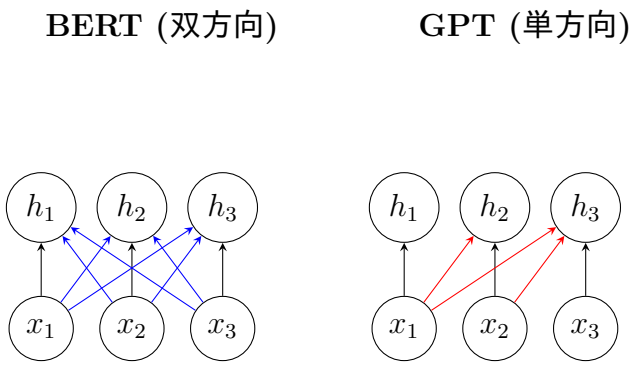


Figure 20.1: アテンションパターンの比較。左: BERT は全トークンへのアテンションを許可する（双方向）。右: GPT は過去のトークンへのアテンションのみを許可する（単方向因果的）。(Radford et al., 2018; Devlin et al., 2019 より改変)

20.3.3 インストラクションチューニングと RLHF

現代の LLM（ChatGPT など）は、人間のフィードバックからの強化学習 (Reinforcement Learning from Human Feedback, RLHF) を使用する：

$$\mathcal{L}_{\text{RL}} = \mathbb{E}[\text{KL}(\pi_\theta \| \pi_{\text{ref}}) - \lambda R(\text{response})] \tag{20.3}$$

ここで R は人間の好みを予測する報酬モデルである。

20.4 エンコーダ-デコーダ: T5 とその派生形

20.4.1 T5: 統一フレームワーク

T5 はすべてのタスクを text-to-text として枠組み化する。

- **翻訳**: “Translate English to German: ...” → ターゲット
- **分類**: “Classify sentiment: ...” → “Positive”

20.5 疎・効率的な派生形

20.5.1 $O(T^2)$ 問題

標準的なアテンションは系列長 T に対して二次的にスケーリングする。

20.5.2 Longformer & BigBird

疎なアテンション (Sparse Attention) を使用する：

- **局所**: ウィンドウサイズ w のみに注意を払う。
- **大域**: 特定のトークン（例：[CLS]）に注意を払う。

計算量は $O(T \cdot w)$ に減少する。

20.5.3 Mixture of Experts (MoE)

Switch Transformers は各トークンを特定の「専門家 (expert)」FFN にルーティングし、推論コストを一定に保ちながら大規模なパラメータ数を可能にする。

20.6 マルチモーダルと Vision Transformer

20.6.1 Vision Transformer (ViT)

画像を 16×16 のパッチに分割し、線形に埋め込み、トークンの系列として扱う。これにより、Transformer のスケーラビリティがコンピュータビジョンにもたらされた。

20.6.2 CLIP

対照学習を通じて画像とテキストのエンコーダを整列させる：

$$\text{sim}(I, T) = E_I(I) \cdot E_T(T) \tag{20.4}$$

20.7 最近のトレンド

20.7.1 RAG (Retrieval-Augmented Generation / 検索拡張生成)

ハルシネーションを減らすために、生成前に文書を検索する。

20.7.2 LoRA (Low-Rank Adaptation)

重み更新を分解することによる効率的なファインチューニング：

$$\mathbf{W}_{\text{new}} = \mathbf{W}_0 + \mathbf{A}\mathbf{B}^T \tag{20.5}$$

ここで $\mathbf{A}, \mathbf{B}$ は低ランク行列である。

20.8 よくある質問 (Transformer の派生形)

20.8.1 Q1: なぜ BERT は双方向で GPT は単方向なのか？

A: タスクが異なる制約を意味するからである。

- **BERT (理解)**: 文全体が利用可能である。「fox」を理解するために、「quick」（左）と「jumps」（右）の両方を見ることが役立つ。
- **GPT (生成)**: 未来のトークンはまだ存在しない。モデルは過去の文脈のみを使用して次の単語を予測しなければならない。

訓練と推論の一貫性が方向性を決定する。

20.8.2 Q2: Masked LM はデータを漏洩させるか？

A: マスキング戦略はチェックを最小限に抑えるが、完璧ではない。もしデータ内で「lazy」が頻繁に出現するなら、モデルはそれを記憶するかもしれない。しかし、タスクは文脈から欠落情報を予測することなので、自明なコピー作業ではなく強力な正則化として機能する。

20.8.3 Q3: なぜモデルはスケールアップすると突然賢くなるのか？

A: これは**創発的能力 (Emergent Abilities)** と呼ばれ、以下の理由が考えられる：

1. **スキルの相互作用**: 別々のスキル（例：構文 + 論理）が組み合わさって複雑な推論を形成する。
2. **構造化された空間**: より大きな潜在空間により、概念の分離が向上する。
3. **メタ学習**: 大規模モデルは事前学習中に「文脈から学ぶ」ことを学習する。

20.8.4 Q4: 温度 (Temperature) は何をするのか？

A: 創造性と決定論性のバランスを制御する。

- **高 τ (例: 1.0+)**: 分布を平坦化し、多様で稀な単語を許容する（創造的）。
- **低 τ (例: 0.1)**: 分布を鋭くし、最も可能性の高い単語のみを選ぶ（事実重視）。

20.8.5 Q5: なぜ RLHF で KL ペナルティが必要なのか？

A:「報酬ハッキング」を防ぐためである。 KL ペナルティがないと、モデルは報酬モデルを技術的に満たす（例：反復的な賞賛）が意味不明な出力を生成する可能性がある。KL 項は、モデルが SFT 中に学習した自然言語分布に留まることを強制する。

20.8.6 Q6: どのように T5 はすべてのタスクを統一できるのか？

A: 根本的に、ほぼすべての NLP タスクは「Seq2Seq」として分類できるからである。

- 分類は系列 $\rightarrow$ ラベル（短い系列）。
- 生成は系列 $\rightarrow$ 系列。

プロンプトを使用することで、T5 は普遍的なマッピング関数を学習する。

20.8.7 Q7: 疎なアテンションは情報を失うか？

A: 可能性はあるが、経験的には稀である。多くのタスクでは、局所的な文脈や特定の大域トークンのみが重要である。「長距離」依存関係の多くは疎（例：500語前の名前を参照）であり、大域トークンやランダムアテンションが効率的に捉えることができる。

20.8.8 Q8: Linear Transformer は完全に等価か？

A: いいえ、カーネル近似である。標準的なソフトマックスアテンションは非線形で、特定の方法で順序に依存する。線形アテンションはこれを特徴マップ $\phi(\cdot)$ で近似する。長い T に対しては非常に高速だが、短い T では精度を落とす可能性がある。

20.8.9 Q9: 専門家 (Experts) の数 (MoE) はどう決めるか？

A: 計算予算とハードウェアに基づく。通常、専門家を増やす（例：64から128）と、推論を遅くすることなくモデル容量を増やせるが、VRAM 使用量は増加する。メモリと FLOPs のトレードオフである。

20.8.10 Q10: なぜ ViT は CNN より優れているのか？

A: **大域的文脈**である。 CNN は局所的である（受容野はゆっくり成長する）。ViT パッチは即座に他のすべてのパッチに注意を払う。十分なデータがあれば（帰納バイアスの欠如を克服するため）、この大域的視点が優れている。

20.8.11 Q11: RAG vs. ファインチューニング？

A:

- **RAG**: 動的な知識（例：「今日のニュース」）、プライバシー、追跡可能性のために使用。
- **ファインチューニング**: スタイル、ドメイン固有言語、または安定した知識の適応のために使用。

多くの場合、頻繁なファインチューニングよりも RAG + 汎用 LLM の方が実用的である。

20.8.12 Q12: Prefix Tuning vs. LoRA?

A: 現在は一般的に LoRA が好まれる。

- **LoRA**: 低ランク行列の注入。レイテンシのオーバーヘッドがない（重みをマージ可能）。
- **Prefix Tuning**: 仮想トークンがコンテキストウィンドウを減らし、最適化が難しい場合がある。

Chapter 21

BERT: マスク付き言語モデリングを用いた双方向エンコーダ

21.1　アーキテクチャの概要

図 21.1 は BERT アーキテクチャを示しており、双方向自己注意（Bidirectional Self-Attention）を持つ Transformer エンコーダ層のスタックを使用しています。

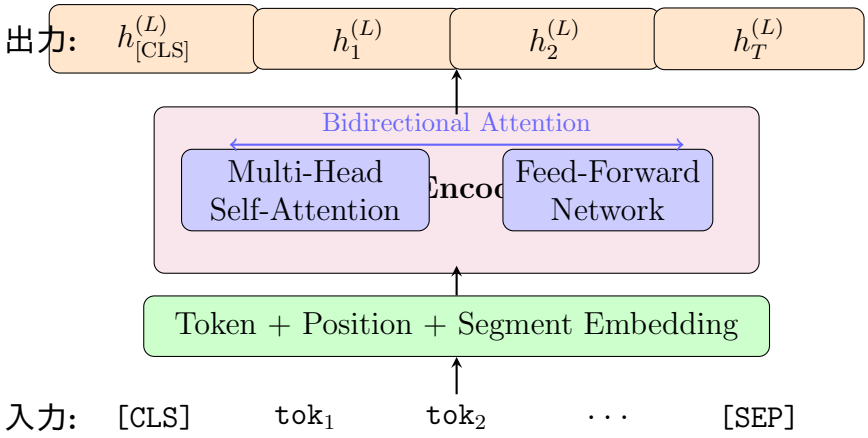


Figure 21.1: BERT アーキテクチャ: 双方向自己注意を持つエンコーダのみの Transformer。系列内のすべてのトークンが他のすべてのトークンに注意を向けることができる。

21.1.1　直感的な理解

なぜ双方向なのか？ 従来の言語モデル（GPTなど）はテキストを左から右に読み、前の単語のみに基づいて各単語を予測します。BERT は、各トークンが左と右の両方のコンテキストを同時に「見る」ことを可能にすることで、これを革新しました。これは、単語ごとではなく、文全体を一度に見るようなものです。

重要な洞察: 人間が言語を理解するとき、私たちは自然に両方向からの文脈を使用します。例えば、「The bank by the river was steep（川沿いの土手は急だった）」において、「bank（銀行/土手）」を理解するには、後で出てくる「river（川）」を見る必要があります。BERT はこの双方向の理解を捉えます。

[CLS] トークン: すべての入力の先頭に付加される特別なトークンです。すべての層を通過した後、その表現は系列全体の情報を集約しており、分類タスクに理想的です。

トレードオフ: 双方向アテンションは、BERT が自己回帰的にテキストを生成できないことを意味します（未来のトークンを見てしまうため）。したがって、BERT は理解タスク（分類、NER、QA）には優れていますが、テキスト生成には向いていません。

21.2 記法と入力表現

$\mathcal{V}$ を語彙（サイズ $|\mathcal{V}|$）、$T_{\max}$ を最大系列長、d_{model} をモデル次元とします。入力系列は以下のように表されます。

$$(x_1, x_2, \ldots, x_T), \quad x_t \in \mathcal{V},\ 1 \le T \le T_{\max}.$$

トークン埋め込み行列

$$E_{\text{tok}} \in \mathbb{R}^{d_{\text{model}} \times |\mathcal{V}|},$$

位置埋め込み行列

$$E_{\text{pos}} \in \mathbb{R}^{d_{\text{model}} \times T_{\max}},$$

およびセグメント埋め込み行列（文 A/B を区別するため）

$$E_{\text{seg}} \in \mathbb{R}^{d_{\text{model}} \times 2}$$

を定義します。

トークン x_t はワンホットベクトル $\text{one\_hot}(x_t) \in \{0,1\}^{|\mathcal{V}|}$ として表され、その埋め込みは

$$e_t^{\text{tok}} = E_{\text{tok}}\, \text{one\_hot}(x_t) \in \mathbb{R}^{d_{\text{model}}}$$

です。位置 t の埋め込みは

$$e_t^{\text{pos}} = E_{\text{pos}}[:, t] \in \mathbb{R}^{d_{\text{model}}},$$

セグメントラベル $s_t \in \{0,1\}$（文 A/B）の埋め込みは

$$e_t^{\text{seg}} = E_{\text{seg}}[:, s_t] \in \mathbb{R}^{d_{\text{model}}}$$

です。

これらは合計されて Transformer への入力を形成します：

$$h_t^{(0)} = e_t^{\text{tok}} + e_t^{\text{pos}} + e_t^{\text{seg}} \in \mathbb{R}^{d_{\text{model}}}, \quad t = 1, \ldots, T.$$

列ベクトル $h_t^{(0)}$ を垂直に積み重ねると：

$$H^{(0)} = \begin{bmatrix} (h_1^{(0)})^\top \\ \vdots \\ (h_T^{(0)})^\top \end{bmatrix} \in \mathbb{R}^{T \times d_{\text{model}}}.$$

パディングを含むバッチ処理の場合、ミニバッチサイズ B、最大長 $T_{\max}$ で：

$$H_{\text{batch}}^{(0)} \in \mathbb{R}^{B \times T_{\max} \times d_{\text{model}}},$$

マスク行列

$$M_{\text{pad}} \in \{0, -\infty\}^{B \times 1 \times T_{\max}}$$

を使用してパディング位置を無視します。

21.3 エンコーダアーキテクチャ: 双方向自己注意

BERT エンコーダは L 個の Transformer ブロックで構成されます。各層 $\ell = 1, \dots, L$ について、自己注意出力は入力 $H^{(\ell-1)} \in \mathbb{R}^{T \times d_{\text{model}}}$ から計算されます。

21.3.1 マルチヘッド自己注意 (Multi-Head Self-Attention)

各ヘッド $h = 1, \dots, H$ について、クエリ、キー、バリュー行列を以下のように定義します。

$$Q^{(\ell,h)} = H^{(\ell-1)} W_Q^{(\ell,h)} \in \mathbb{R}^{T \times d_k},$$
$$K^{(\ell,h)} = H^{(\ell-1)} W_K^{(\ell,h)} \in \mathbb{R}^{T \times d_k},$$
$$V^{(\ell,h)} = H^{(\ell-1)} W_V^{(\ell,h)} \in \mathbb{R}^{T \times d_v},$$

ここで

$$W_Q^{(\ell,h)}, W_K^{(\ell,h)} \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_V^{(\ell,h)} \in \mathbb{R}^{d_{\text{model}} \times d_v},$$

通常 $d_k = d_v = d_{\text{model}}/H$ です。

スコア行列

$$S^{(\ell,h)} = \frac{Q^{(\ell,h)} (K^{(\ell,h)})^\top}{\sqrt{d_k}} \in \mathbb{R}^{T \times T}$$

が計算されます。BERT では、すべてのトークン間の双方向アテンションが許可されるため、パディングのみがマスクされます。パディングマスク $M_{\text{pad}}^{(\text{1D})} \in \{0, -\infty\}^T$ を拡張し：

$$M_{ij}^{(\ell)} = \begin{cases} 0 & \text{どちらもパディングでない場合,} \\ -\infty & \text{少なくとも一方がパディングの場合,} \end{cases}$$

そして

$$\tilde{S}^{(\ell,h)} = S^{(\ell,h)} + M^{(\ell)}.$$

アテンション重みはソフトマックスによって計算されます：

$$A^{(\ell,h)} = \text{softmax}\big(\tilde{S}^{(\ell,h)}\big) \in \mathbb{R}^{T \times T}, \quad A_{ij}^{(\ell,h)} = \frac{\exp(\tilde{S}_{ij}^{(\ell,h)})}{\sum_{k=1}^{T} \exp(\tilde{S}_{ik}^{(\ell,h)})}.$$

ヘッド出力:

$$Y^{(\ell,h)} = A^{(\ell,h)} V^{(\ell,h)} \in \mathbb{R}^{T \times d_v}.$$

すべてのヘッドが連結され、出力射影 $W_O^{(\ell)} \in \mathbb{R}^{H d_v \times d_{\text{model}}}$ によって変換されます：

$$Y^{(\ell)} = \text{Concat}\big(Y^{(\ell,1)}, \dots, Y^{(\ell,H)}\big) W_O^{(\ell)} \in \mathbb{R}^{T \times d_{\text{model}}}.$$

21.3.2 残差接続と層正規化 (Layer Normalization)

自己注意サブ層の出力は

$$\tilde{H}^{(\ell)} = \text{LN}\big(H^{(\ell-1)} + Y^{(\ell)}\big),$$

ここで LN は位置ごとに適用される LayerNorm です。ベクトル $x \in \mathbb{R}^{d_{\text{model}}}$ に対して：

$$\mu(x) = \frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} x_i, \quad \sigma^2(x) = \frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} (x_i - \mu(x))^2,$$

$$\mathrm{LN}(x) = \gamma \odot \frac{x - \mu(x)\mathbf{1}}{\sqrt{\sigma^2(x) + \varepsilon}} + \beta,$$

ここで $\gamma, \beta \in \mathbb{R}^{d_{\text{model}}}$ は学習可能なパラメータ、$\varepsilon > 0$ は小さな定数です。

21.3.3 位置ごとのフィードフォワードネットワーク (Position-Wise FFN)

各位置 t における FFN は：

$$\mathrm{FFN}(u_t) = W_2^{(\ell)}\, \phi\big(W_1^{(\ell)} u_t + b_1^{(\ell)}\big) + b_2^{(\ell)},$$

$$W_1^{(\ell)} \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}, \quad W_2^{(\ell)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}},$$

ここで ϕ は ReLU などの非線形関数です。行列形式では：

$$\mathrm{FFN}(\tilde{H}^{(\ell)}) = \phi\big(\tilde{H}^{(\ell)} W_1^{(\ell)\top} + \mathbf{1}(b_1^{(\ell)})^\top\big) W_2^{(\ell)\top} + \mathbf{1}(b_2^{(\ell)})^\top,$$

ここで $\mathbf{1} \in \mathbb{R}^T$ はすべて1のベクトルです。

残差接続と LayerNorm を含めると、層の出力は：

$$H^{(\ell)} = \mathrm{LN}\Big(\tilde{H}^{(\ell)} + \mathrm{FFN}\big(\tilde{H}^{(\ell)}\big)\Big).$$

21.4 事前学習目的関数 I: マスク付き言語モデリング (MLM)

21.4.1 マスキング戦略

元の系列 $(x_1, \ldots, x_T)$ から、マスク位置集合 $\mathcal{M} \subset \{1, \ldots, T\}$ がランダムにサンプリングされます（例：全トークンの 15%）。 $t \in \mathcal{M}$ に対して：

- 80% の確率で、[MASK] に置き換える;
- 10% の確率で、ランダムなトークンに置き換える;
- 10% の確率で、元のトークンを維持する（ただし損失には含める）。

結果の入力系列 $(\tilde{x}_1, \ldots, \tilde{x}_T)$ がエンコーダに入力されます：

$$H^{(L)} = \mathrm{Encoder}(\tilde{x}_{1:T}), \quad h_t^{(L)} \in \mathbb{R}^{d_{\text{model}}}.$$

21.4.2 トークンレベルのロジットと確率

マスクされた位置 $t \in \mathcal{M}$ に対して、出力ロジットは：

$$z_t = W_{\text{MLM}} h_t^{(L)} + b_{\text{MLM}} \in \mathbb{R}^{|\mathcal{V}|},$$

$$p_t = \mathrm{softmax}(z_t), \quad p_t(v) = \frac{\exp(z_{t,v})}{\sum_{u \in \mathcal{V}} \exp(z_{t,u})}.$$

真のトークンを $x_t^\star \in \mathcal{V}$ とします。単一の位置に対する損失はカテゴリカル交差エントロピーです：

$$\ell_{\mathrm{MLM}}(t) = -\log p_t(x_t^\star).$$

系列ごとの MLM 損失は：

$$\mathcal{L}_{\mathrm{MLM}} = \frac{1}{|\mathcal{M}|}\sum_{t\in\mathcal{M}} \ell_{\mathrm{MLM}}(t) = -\frac{1}{|\mathcal{M}|}\sum_{t\in\mathcal{M}} \log p_t(x_t^\star).$$

データセット $\mathcal{D}$ 全体での期待損失（経験リスク）は：

$$\mathcal{L}_{\mathrm{MLM}}(\theta) = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} \mathbb{E}_{\mathcal{M}\sim\Pi}\left[-\frac{1}{|\mathcal{M}|}\sum_{t\in\mathcal{M}} \log p_t(x_t^\star;\theta)\right],$$

ここで Π はマスキング分布、θ は BERT の全パラメータセットです。

21.4.3 ロジットに関する勾配

位置 t について、ロジット $z_t \in \mathbb{R}^{|\mathcal{V}|}$ に関する勾配を導出します。ワンホットラベルベクトル $y_t \in \{0,1\}^{|\mathcal{V}|}$ を以下のように定義します：

$$(y_t)_v = \begin{cases} 1 & v = x_t^\star, \\ 0 & \text{それ以外}, \end{cases}$$

すると、

$$\ell_{\mathrm{MLM}}(t) = -\sum_{v\in\mathcal{V}} (y_t)_v \log p_t(v).$$

標準的なソフトマックス+CCEの結果により：

$$\nabla_{z_t}\, \ell_{\mathrm{MLM}}(t) = p_t - y_t.$$

したがって、ミニバッチ平均では：

$$\nabla_{z_t}\, \mathcal{L}_{\mathrm{MLM}} = \frac{1}{|\mathcal{M}|}(p_t - y_t).$$

この勾配は出力重み W_{MLM} と隠れ表現 $h_t^{(L)}$ に伝播します：

$$\nabla_{W_{\mathrm{MLM}}} \mathcal{L}_{\mathrm{MLM}} = \sum_{t\in\mathcal{M}} (p_t - y_t)\,(h_t^{(L)})^\top,$$

$$\nabla_{h_t^{(L)}} \mathcal{L}_{\mathrm{MLM}} = W_{\mathrm{MLM}}^\top (p_t - y_t), \quad t \in \mathcal{M}.$$

21.5 事前学習目的関数 II: 次文予測 (NSP)

21.5.1 ペア表現

NSP は、[CLS] トークン表現を使用して、2つの文 (A, B) の連結系列に対して二値分類を実行します。入力系列は：

$$[\mathrm{CLS}], A, [\mathrm{SEP}], B, [\mathrm{SEP}]$$

であり、[CLS] 位置での出力ベクトルは $h_{\text{CLS}}^{(L)}$ と表記されます。
線形分類器:

$$u = W_{\text{NSP}} h_{\text{CLS}}^{(L)} + b_{\text{NSP}} \in \mathbb{R}, \quad q = \sigma(u) = \frac{1}{1+\exp(-u)},$$

ラベル $y_{\text{NSP}} \in \{0,1\}$（1: 正しい次文、0: ランダムな文）に対して、損失は：

$$\mathcal{L}_{\text{NSP}} = -\left[y_{\text{NSP}} \log q + (1-y_{\text{NSP}}) \log(1-q)\right].$$

21.5.2 結合損失

単一のサンプル（マスクされたトークンを含む文ペア）に対する総損失は：

$$\mathcal{L}_{\text{BERT}} = \mathcal{L}_{\text{MLM}} + \lambda_{\text{NSP}} \mathcal{L}_{\text{NSP}},$$

データセット平均:

$$\min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}_{\text{BERT}}(x,y;\theta).$$

21.6 バッチ処理、マスク、および複雑性

21.6.1 アテンションマスク行列

ミニバッチサイズ B、最大長 $T_{\max}$ で、バッチテンソルは：

$$H^{(\ell)} \in \mathbb{R}^{B\times T_{\max}\times d_{\text{model}}}$$

パディングマスク $M_{\text{pad}} \in \{0,-\infty\}^{B\times 1\times T_{\max}}$ を持ちます。
自己注意では、スコアテンソル

$$S \in \mathbb{R}^{B\times H\times T_{\max}\times T_{\max}}$$

に対してブロードキャストが適用されます：

$$\tilde{S}_{b,h,i,j} = S_{b,h,i,j} + M_{\text{pad}}[b,0,j] + M_{\text{pad}}[b,0,i],$$

これにより、どちらかの位置がPADであれば $-\infty$ になります。

21.6.2 計算コスト

層ごとのアテンション計算は：

$$O\big(H\, T_{\max}^2 d_k\big),$$

FFN は：

$$O\big(T_{\max} d_{\text{model}} d_{\text{ff}}\big).$$

全 L 層の順伝播計算合計は：

$$O\big(L(H T_{\max}^2 d_k + T_{\max} d_{\text{model}} d_{\text{ff}})\big),$$

逆伝播も同じオーダーです。

21.7 要約

BERT は

- 双方向自己注意エンコーダ（全結合アテンションマスク）を介して文脈表現 $h_t^{(L)}$ を構築し、
- マスクされたトークンの再構成 (MLM) と文ペア予測 (NSP) を組み合わせた事前学習目的関数を使用します

これにより汎用的な表現を学習します。これらすべては、トークンレベルの対数尤度最大化問題として厳密に定式化できます。

Chapter 22

GPT: デコーダのみの自己回帰 Transformer

22.1 アーキテクチャの概要

図 22.1 は GPT アーキテクチャを示しており、因果的（単方向）自己注意を持つ Transformer デコーダ層のスタックを使用しています。

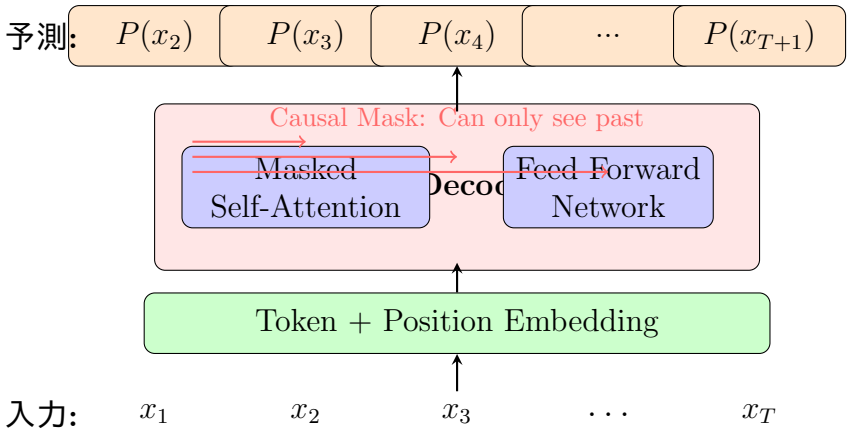


Figure 22.1: GPT アーキテクチャ: 因果的（左から右）自己注意を持つデコーダのみの Transformer。各位置は前の位置にのみ注意を向けることができる。

22.1.1 直感的な理解

なぜ因果的/自己回帰的なのか？ GPT は言語を逐次予測問題としてモデル化します：過去のすべての単語を与えられて、次の単語を予測します。これは人間がテキストを書く方法を反映しています。つまり、前に来たものを基にして、一度に一語ずつです。

因果マスク (Causal Mask): 重要なメカニズムは、各位置が未来のトークンを「覗き見る」ことを防ぐ三角形のアテンションマスクです。位置3は位置1と2を見ることができますが、4や5などは見ることができません。これにより、自己回帰特性を維持しながら、系列全体を並列に学習することが可能になります。

なぜデコーダのみなのか？ エンコーダ-デコーダモデル（T5など）とは異なり、GPT はデコーダブロックのみを使用します。この単純さと大規模なスケールが組み

合わさることで、インコンテキスト学習やフューショット推論のような創発的能力が生まれます。

生成プロセス: 推論時、GPT は反復的にテキストを生成します：

1. プロンプトトークンを与えられ、すべての隠れ状態を並列に計算する
2. 予測された分布から次のトークンをサンプリングする
3. サンプリングされたトークンを追加して繰り返す

スケーリングの洞察: GPT ファミリーは、単にモデルサイズ、データ、計算量をスケーリングするだけで、推論、知識、汎化における質的な向上（「スケーリング則」）につながることを実証しました。

22.2 記法と言語モデルの因数分解

$\mathcal{V}$ をサイズ $|\mathcal{V}|$ の語彙、$T_{\max}$ を最大系列長とします。テキストはトークン系列として表されます

$$x_{1:T} = (x_1, x_2, \ldots, x_T), \quad x_t \in \mathcal{V},\ 1 \le T \le T_{\max}.$$

自己回帰言語モデルは、連鎖律によって確率分布を表します：

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}) \quad （ここで\ x_{<t} = x_1, \ldots, x_{t-1}）$$

パラメータは θ です。

負の対数尤度（系列ごと）は：

$$\mathcal{L}_{\mathrm{NLL}}(x_{1:T}; \theta) = -\log p_\theta(x_{1:T}) = -\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}).$$

実際には、訓練では1トークンずらした入力と出力系列を使用します。例えば、入力は $(x_1, \ldots, x_T)$、ターゲットは $(x_2, \ldots, x_{T+1})$ です（最後は EOS）：

$$\mathcal{L}_{\mathrm{GPT}}(x_{1:T+1}; \theta) = -\frac{1}{T} \sum_{t=1}^{T} \log p_\theta(x_{t+1} \mid x_{1:t}).$$

以下では、この条件付き確率 $p_\theta(\cdot \mid x_{1:t})$ が Transformer デコーダによってどのように計算されるかを定式化します。

22.3 トークン、位置、および入力埋め込み

語彙埋め込み行列

$$E_{\mathrm{tok}} \in \mathbb{R}^{d_{\mathrm{model}} \times |\mathcal{V}|},$$

および位置埋め込み行列

$$E_{\mathrm{pos}} \in \mathbb{R}^{d_{\mathrm{model}} \times T_{\max}}$$

を使用します。トークン x_t はワンホットベクトル $\mathrm{one\_hot}(x_t) \in \{0,1\}^{|\mathcal{V}|}$ として表されます：

$$e_t^{\mathrm{tok}} = E_{\mathrm{tok}}\, \mathrm{one\_hot}(x_t) \in \mathbb{R}^{d_{\mathrm{model}}}, \quad e_t^{\mathrm{pos}} = E_{\mathrm{pos}}[:, t] \in \mathbb{R}^{d_{\mathrm{model}}}.$$

入力埋め込みは：

$$h_t^{(0)} = e_t^{\text{tok}} + e_t^{\text{pos}} \in \mathbb{R}^{d_{\text{model}}}, \quad t = 1, \ldots, T.$$

行列形式では：

$$H^{(0)} = \begin{bmatrix} (h_1^{(0)})^\top \\ \vdots \\ (h_T^{(0)})^\top \end{bmatrix} \in \mathbb{R}^{T \times d_{\text{model}}}.$$

22.4 因果的マルチヘッド自己注意

GPT は L 個の Transformer デコーダブロックで構成され、各層 ℓ は：

$$H^{(\ell)} = \text{Block}^{(\ell)}(H^{(\ell-1)}), \quad \ell = 1, \ldots, L.$$

22.4.1 因果マスク付きシングルヘッド自己注意

$H^{(\ell-1)} \in \mathbb{R}^{T \times d_{\text{model}}}$ に対して、シングルヘッドのクエリ、キー、およびバリュー行列を以下のように定義します：

$$Q = H^{(\ell-1)} W_Q, \quad K = H^{(\ell-1)} W_K, \quad V = H^{(\ell-1)} W_V,$$

$$W_Q, W_K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_V \in \mathbb{R}^{d_{\text{model}} \times d_v}.$$

行 t を q_t, k_t, v_t とします。

スコア行列:

$$S \in \mathbb{R}^{T \times T}, \quad S_{ij} = \frac{q_i^\top k_j}{\sqrt{d_k}}.$$

因果マスク $M \in \{0, -\infty\}^{T \times T}$ は：

$$M_{ij} = \begin{cases} 0 & j \le i, \\ -\infty & j > i, \end{cases}$$

これは未来のトークンへのアテンションを防ぎます。マスクされたスコア：

$$\tilde{S} = S + M.$$

行方向のソフトマックスによるアテンション重み：

$$A_{ij} = \frac{\exp(\tilde{S}_{ij})}{\sum_{k=1}^{T} \exp(\tilde{S}_{ik})}, \quad A \in \mathbb{R}^{T \times T}.$$

シングルヘッド出力:

$$Y = AV \in \mathbb{R}^{T \times d_v}, \quad y_i = \sum_{j=1}^{T} A_{ij} v_j.$$

22.4.2 マルチヘッドアテンションと出力射影

ヘッド数を H とします。各ヘッド h に対して：

$$Q^{(h)} = H^{(\ell-1)}W_Q^{(h)}, \quad K^{(h)} = H^{(\ell-1)}W_K^{(h)}, \quad V^{(h)} = H^{(\ell-1)}W_V^{(h)},$$

$$Y^{(h)} = \mathrm{Attention}_{\mathrm{causal}}(Q^{(h)}, K^{(h)}, V^{(h)}), \quad Y^{(h)} \in \mathbb{R}^{T\times d_v}.$$

連結して線形変換を適用します：

$$Y^{(\ell)} = \mathrm{Concat}\big(Y^{(1)}, \ldots, Y^{(H)}\big)W_O^{(\ell)} \in \mathbb{R}^{T\times d_{\mathrm{model}}},$$

$$W_O^{(\ell)} \in \mathbb{R}^{Hd_v\times d_{\mathrm{model}}}.$$

22.4.3 残差と Pre/Post-Norm 派生形

標準的な Post-LN 形式は：

$$\tilde{H}^{(\ell)} = \mathrm{LN}\big(H^{(\ell-1)} + Y^{(\ell)}\big),$$

$$H^{(\ell)} = \mathrm{LN}\Big(\tilde{H}^{(\ell)} + \mathrm{FFN}\Big(\tilde{H}^{(\ell)}\Big)\Big).$$

多くの GPT スタイルのモデルは Pre-LN 形式を使用します：

$$\hat{H}^{(\ell)} = H^{(\ell-1)} + \mathrm{MHA}_{\mathrm{causal}}\big(\mathrm{LN}(H^{(\ell-1)})\big),$$

$$H^{(\ell)} = \hat{H}^{(\ell)} + \mathrm{FFN}\Big(\mathrm{LN}(\hat{H}^{(\ell)})\Big).$$

ベクトル $x \in \mathbb{R}^{d_{\mathrm{model}}}$ に対する LayerNorm LN は：

$$\mathrm{LN}(x) = \gamma \odot \frac{x - \mu(x)\mathbf{1}}{\sqrt{\sigma^2(x) + \varepsilon}} + \beta,$$

$$\mu(x) = \frac{1}{d_{\mathrm{model}}}\sum_{i=1}^{d_{\mathrm{model}}} x_i, \quad \sigma^2(x) = \frac{1}{d_{\mathrm{model}}}\sum_{i=1}^{d_{\mathrm{model}}} (x_i - \mu(x))^2,$$

ここで $\gamma, \beta \in \mathbb{R}^{d_{\mathrm{model}}}$ は学習可能なパラメータです。

22.5 位置ごとのフィードフォワードネットワーク

各層 ℓ の FFN は、各位置に対して独立に適用されます：

$$\mathrm{FFN}^{(\ell)}(u) = W_2^{(\ell)}\,\phi\Big(W_1^{(\ell)}u + b_1^{(\ell)}\Big) + b_2^{(\ell)},$$

$$W_1^{(\ell)} \in \mathbb{R}^{d_{\mathrm{ff}}\times d_{\mathrm{model}}}, \quad W_2^{(\ell)} \in \mathbb{R}^{d_{\mathrm{model}}\times d_{\mathrm{ff}}},$$

ここで ϕ は ReLU, GELU などです。行列形式では：

$$\mathrm{FFN}^{(\ell)}(H) = \phi\Big(HW_1^{(\ell)\top} + \mathbf{1}(b_1^{(\ell)})^\top\Big)\,W_2^{(\ell)\top} + \mathbf{1}(b_2^{(\ell)})^\top,$$

ここで $\mathbf{1} \in \mathbb{R}^T$ はすべて1のベクトルです。

22.6 出力層と条件付き分布

最終層の出力は：

$$H^{(L)} = \begin{bmatrix} (h_1^{(L)})^\top \\ \vdots \\ (h_T^{(L)})^\top \end{bmatrix},$$

語彙ロジットは：

$$z_t = W_{\mathrm{LM}} h_t^{(L)} + b_{\mathrm{LM}} \in \mathbb{R}^{|\mathcal{V}|},$$

$$p_\theta(x_{t+1} = v \mid x_{1:t}) = \frac{\exp(z_{t,v})}{\sum_{u \in \mathcal{V}} \exp(z_{t,u})}, \quad v \in \mathcal{V}.$$

重み共有（Weight tying）により、$W_{\mathrm{LM}} = E_{\mathrm{tok}}^\top$ とし、埋め込みと出力射影の間でパラメータを共有することがよくあります。

22.7 学習目的関数と勾配

22.7.1 系列損失とトークンごとの損失

入力 $x_{1:T+1}$ に対する損失：

$$\mathcal{L}_{\mathrm{GPT}}(x_{1:T+1}; \theta) = -\frac{1}{T} \sum_{t=1}^{T} \log p_\theta(x_{t+1} \mid x_{1:t}).$$

時間 t における損失：

$$\ell_t(\theta) = -\log p_\theta(x_{t+1}^\star \mid x_{1:t}),$$

ワンホットラベル $y_t \in \{0,1\}^{|\mathcal{V}|}$ を用いて：

$$(y_t)_v = \begin{cases} 1 & v = x_{t+1}^\star, \\ 0 & \text{それ以外}, \end{cases}$$

とすると

$$\ell_t(\theta) = -\sum_{v \in \mathcal{V}} (y_t)_v \log p_\theta(v \mid x_{1:t}).$$

22.7.2 ロジットおよび隠れ状態に関する勾配

標準的なソフトマックス + 交差エントロピーの結果により：

$$\nabla_{z_t} \ell_t = p_t - y_t, \quad p_t = p_\theta(\cdot \mid x_{1:t}).$$

したがって、バッチ平均（正規化係数 $1/T$ を含む）では：

$$\nabla_{z_t} \mathcal{L}_{\mathrm{GPT}} = \frac{1}{T}(p_t - y_t).$$

線形射影 $z_t = W_{\mathrm{LM}} h_t^{(L)} + b_{\mathrm{LM}}$ から：

$$\nabla_{W_{\mathrm{LM}}} \mathcal{L}_{\mathrm{GPT}} = \frac{1}{T} \sum_{t=1}^{T} (p_t - y_t)\, (h_t^{(L)})^\top,$$

$$\nabla_{b_{\mathrm{LM}}} \mathcal{L}_{\mathrm{GPT}} = \frac{1}{T} \sum_{t=1}^{T} (p_t - y_t),$$

$$\nabla_{h_t^{(L)}} \mathcal{L}_{\mathrm{GPT}} = \frac{1}{T} W_{\mathrm{LM}}^{\top} (p_t - y_t).$$

この $\nabla_{h_t^{(L)}} \mathcal{L}_{\mathrm{GPT}}$ は Transformer デコーダを通じて逆伝播されます。

22.8 教師強制と推論

22.8.1 訓練中の教師強制 (Teacher Forcing)

訓練中、条件付き分布

$$p_\theta(x_{t+1} \mid x_{1:t})$$

を評価するために、入力系列 $(x_1, \ldots, x_t)$ には常に「グラウンドトゥルースのトークン」が与えられます（教師強制）。したがって、訓練中に計算される尤度は：

$$p_\theta(x^\star_{2:T+1} \mid x^\star_{1:T}) = \prod_{t=1}^{T} p_\theta(x^\star_{t+1} \mid x^\star_{1:t}).$$

22.8.2 テスト時の自己回帰生成

生成中、モデルのサンプルは再帰的にフィードバックされます：

$$\begin{aligned} &\text{given } x_1, \ldots, x_t, \\ &\quad p_\theta(x_{t+1} \mid x_{1:t}) = \mathrm{softmax}(W_{\mathrm{LM}} h_t^{(L)} + b_{\mathrm{LM}}), \\ &\quad x_{t+1} \sim p_\theta(\cdot \mid x_{1:t}). \end{aligned}$$

温度 $\tau > 0$ を用いると：

$$p_\theta^\tau(v \mid x_{1:t}) = \frac{\exp(z_{t,v}/\tau)}{\sum_u \exp(z_{t,u}/\tau)}.$$

22.9 パープレキシティと評価指標

テスト分布 $\mathcal{D}_{\mathrm{test}}$ 上でのトークンごとの平均負の対数尤度は：

$$\bar{\ell} = \mathbb{E}_{x_{1:T} \sim \mathcal{D}_{\mathrm{test}}} \left[-\frac{1}{T} \sum_{t=1}^{T} \log_2 p_\theta(x_t \mid x_{<t}) \right],$$

そしてパープレキシティは以下のように定義されます：

$$\mathrm{PPL} = 2^{\bar{\ell}}.$$

これは、「モデルが平均して何個の選択肢から選んでいるか」を表す有効語彙サイズとして解釈できます。

22.10 バッチ処理、因果マスク、および複雑性

22.10.1 バッチごとの因果アテンション

ミニバッチサイズ B、最大長 $T_{\max}$ で：

$$H^{(\ell)} \in \mathbb{R}^{B\times T_{\max}\times d_{\text{model}}},$$

各ヘッドに対して：

$$Q, K, V \in \mathbb{R}^{B\times H\times T_{\max}\times d_k},$$

スコア：

$$S_{b,h,i,j} = \frac{1}{\sqrt{d_k}} Q_{b,h,i,:} \cdot K_{b,h,j,:},$$

マスク：

$$M_{i,j} = \begin{cases} 0 & j \le i, \\ -\infty & j > i, \end{cases}$$

これをブロードキャストして：

$$\tilde{S}_{b,h,i,j} = S_{b,h,i,j} + M_{i,j}.$$

ソフトマックスによるアテンション重み：

$$A_{b,h,i,j} = \frac{\exp(\tilde{S}_{b,h,i,j})}{\sum_{k=1}^{T_{\max}} \exp(\tilde{S}_{b,h,i,k})},$$

出力：

$$Y_{b,h,i,:} = \sum_{j=1}^{T_{\max}} A_{b,h,i,j} V_{b,h,j,:}.$$

22.10.2 計算複雑性

層ごとの計算：

$$O\big(BHT_{\max}^2 d_k\big) \quad \text{(Attention)},$$
$$O\big(BT_{\max} d_{\text{model}} d_{\text{ff}}\big) \quad \text{(FFN)}.$$

全 L 層に対して：

$$O\Big(L\big(BHT_{\max}^2 d_k + BT_{\max} d_{\text{model}} d_{\text{ff}}\big)\Big).$$

推論中、KV キャッシュにより時間 t での計算は以下に削減されます：

$$O\big(BHt d_k + B d_{\text{model}} d_{\text{ff}}\big),$$

長さ T の系列生成の合計は：

$$O\big(BHT^2 d_k + BT d_{\text{model}} d_{\text{ff}}\big).$$

22.11 要約

GPT は

- 因果マスク付き自己注意を通じて $p_\theta(x_t \mid x_{<t})$ を表現し、
- トークン埋め込みと出力線形射影からのソフトマックスを通じて条件付きカテゴリカル分布を定義し、
- ミニバッチ単位で負の対数尤度を最小化することで、

厳密な確率モデルを定式化しています。このフレームワークは、後の章で議論されるスケーリング則、インコンテキスト学習、および RLHF の基礎を形成します。

Chapter 23

RoBERTa: 堅牢に最適化された BERT 事前学習

23.1　アーキテクチャの概要

RoBERTa は基本的に BERT と同じエンコーダのみの Transformer 構造を持ち、すべてのトークンに対して双方向自己注意を適用します。

$\mathcal{V}$ を語彙、$T_{\max}$ を最大系列長、d_{model} をモデル次元、L を層数とします。入力トークン系列

$$x_{1:T} = (x_1, \ldots, x_T), \quad x_t \in \mathcal{V},\ 1 \le T \le T_{\max}$$

に対して、エンコーダ出力は：

$$H^{(L)} = \begin{bmatrix} (h_1^{(L)})^\top \\ \vdots \\ (h_T^{(L)})^\top \end{bmatrix} \in \mathbb{R}^{T \times d_{\text{model}}}.$$

RoBERTa の本質的な違いは以下の通りです：

- 訓練目的（動的マスキングを用いた MLM のみ）、
- NSP の削除、
- 訓練スケジュール（大バッチ、長時間訓練、より大きなコーパス）。

23.2　トークンとセグメント表現

語彙埋め込み行列：

$$E_{\text{tok}} \in \mathbb{R}^{d_{\text{model}} \times |\mathcal{V}|},$$

位置埋め込み：

$$E_{\text{pos}} \in \mathbb{R}^{d_{\text{model}} \times T_{\max}}.$$

BERT とは異なり、RoBERTa はセグメント埋め込みを使用せず、すべてを単一の文（または連結された文）として扱います。

トークン x_t のワンホットベクトルを $\text{one\_hot}(x_t) \in \{0,1\}^{|\mathcal{V}|}$ とすると：

$$e_t^{\text{tok}} = E_{\text{tok}}\,\text{one\_hot}(x_t) \in \mathbb{R}^{d_{\text{model}}}, \quad e_t^{\text{pos}} = E_{\text{pos}}[:, t] \in \mathbb{R}^{d_{\text{model}}}.$$

入力埋め込み：

$$h_t^{(0)} = e_t^{\text{tok}} + e_t^{\text{pos}} \in \mathbb{R}^{d_{\text{model}}}, \quad t = 1, \ldots, T.$$

23.3 双方向自己注意エンコーダ

各層 ℓ に対して：

$$H^{(\ell)} = \mathrm{EncoderBlock}^{(\ell)}\big(H^{(\ell-1)}\big), \quad \ell = 1, \dots, L.$$

EncoderBlock は、マルチヘッド自己注意（マスクなし）、位置ごとの FFN、および LayerNorm を伴う残差接続で構成されます。

23.3.1 マルチヘッド自己注意（マスクなし）

層 ℓ の入力 $H^{(\ell-1)} \in \mathbb{R}^{T \times d_{\text{model}}}$ に対して、ヘッド $h = 1, \dots, H$ について：

$$Q^{(h)} = H^{(\ell-1)} W_Q^{(h)}, \quad K^{(h)} = H^{(\ell-1)} W_K^{(h)}, \quad V^{(h)} = H^{(\ell-1)} W_V^{(h)}.$$

スコア行列：

$$S_{ij}^{(h)} = \frac{(q_i^{(h)})^\top k_j^{(h)}}{\sqrt{d_k}},$$

これは BERT/RoBERTa なので因果マスクは使用されず、すべての i, j のスコアが直接ソフトマックスに渡されます：

$$A_{ij}^{(h)} = \frac{\exp(S_{ij}^{(h)})}{\sum_{k=1}^{T} \exp(S_{ik}^{(h)})}.$$

23.3.2 Pre-LN エンコーダブロック

実際には、Pre-LN がよく使用されます：

$$\hat{H}^{(\ell)} = H^{(\ell-1)} + \mathrm{MHA}\big(\mathrm{LN}(H^{(\ell-1)})\big),$$

$$H^{(\ell)} = \hat{H}^{(\ell)} + \mathrm{FFN}\Big(\mathrm{LN}(\hat{H}^{(\ell)})\Big).$$

23.4 動的マスキングを用いたマスク付き言語モデリング

23.4.1 ランダムプロセスとしてのマスキング戦略

入力系列 $x_{1:T}$ に対して、マスク位置集合 $M \subset \{1, \dots, T\}$ が確率的に選択されます。各位置 t に対して、ベルヌーイ変数 $m_t \sim \mathrm{Bernoulli}(p_{\text{mask}})$、 $p_{\text{mask}} \approx 0.15$、マスク集合 $M = \{t \mid m_t = 1\}$。

各 $t \in M$ に対して：

$$\tilde{x}_t = \begin{cases} [\texttt{MASK}] & \text{確率 } 0.8, \\ u \sim \mathrm{Unif}(\mathcal{V}) & \text{確率 } 0.1, \\ x_t & \text{確率 } 0.1. \end{cases}$$

RoBERTa は「動的マスキング」を採用しており、異なる反復で同じ文が異なる M でマスクされます。

23.4.2 条件付き尤度としての MLM 目的関数

マスクされた入力系列を $\tilde{x}_{1:T}$、エンコーダ出力を $H^{(L)}(\tilde{x}_{1:T})$ とします。位置 t における隠れ状態 $h_t^{(L)}$ から：

$$z_t = W_{\text{MLM}} h_t^{(L)} + b_{\text{MLM}} \in \mathbb{R}^{|\mathcal{V}|},$$

$$p_\theta(v \mid \tilde{x}_{1:T}, t) = \frac{\exp(z_{t,v})}{\sum_{u \in \mathcal{V}} \exp(z_{t,u})}.$$

文ごとの MLM 損失：

$$\mathcal{L}_{\text{MLM}}(x_{1:T}; \theta) = \mathbb{E}_{M, \tilde{x}_{1:T}} \left[-\frac{1}{|M|} \sum_{t \in M} \log p_\theta(x_t \mid \tilde{x}_{1:T}, t) \right].$$

23.4.3 トークンごとの交差エントロピーと勾配

位置 $t \in M$ の損失：

$$\ell_t(\theta) = -\log p_\theta(x_t^\star \mid \tilde{x}_{1:T}, t).$$

標準的なソフトマックス + 交差エントロピーの結果により：

$$\nabla_{z_t} \ell_t = p_t - y_t.$$

射影パラメータに関する勾配：

$$\nabla_{W_{\text{MLM}}} \hat{\mathcal{L}}_{\text{MLM}} = \frac{1}{|M|} \sum_{t \in M} (p_t - y_t) \, (h_t^{(L)})^\top.$$

23.5 動的 vs. 静的マスキング: 分布的視点

BERT の「静的マスキング」はコーパスの前処理時に M を一度だけサンプリングし、同じマスクパターンで複数のエポック訓練します。

RoBERTa の動的マスキングは各エポックで新しい $M^{(e)}$ をサンプリングします：

$$\mathcal{L}_{\text{MLM}}(x_{1:T}; \theta) = \mathbb{E}_M \big[L(x_{1:T}, M; \theta) \big],$$

これは期待値をより忠実に近似するモンテカルロ反復を提供します。

23.6 事前学習目的関数と最適化

大規模コーパス $\mathcal{D}$ に対して、RoBERTa の事前学習目的関数は：

$$\min_\theta \; \mathbb{E}_{x_{1:T} \sim \mathcal{D}} \Big[\mathcal{L}_{\text{MLM}}(x_{1:T}; \theta) \Big].$$

RoBERTa は以下を使用します：

- より大きなバッチサイズ B,
- より長い訓練ステップ,
- より大きなデータセット

これにより、BERT と同じアーキテクチャを維持しながら表現能力を向上させています。

Chapter 24

Longformer: 効率的な長文書 Transformer

24.1　動機と $O(T^2)$ のボトルネック

標準的な Transformer の自己注意は、系列長 T に対して時間とメモリにおいて $O(T^2)$ でスケーリングします。長文書（$T > 4096$ など）の場合、この二次的な複雑さは実用上のボトルネックになります。 Longformer は、長距離依存関係を維持しながら計算を線形に削減するための **疎なアテンション (sparse attention)** パターンを導入します。

24.2　アテンションマスクとスパース性パターン

24.2.1　完全アテンションの要約

標準的な完全アテンションでは、位置 i はすべての位置 $j \in \{1, \dots, T\}$ に注意を向けます：

$$A_{ij} = \frac{\exp(S_{ij})}{\sum_{k=1}^{T} \exp(S_{ik})}, \quad y_i = \sum_{j=1}^{T} A_{ij} v_j.$$

24.2.2　構造化マスクとしての疎なアテンション

Longformer は、各位置 i に対して「許可されたアテンション集合」$\mathcal{N}(i) \subseteq \{1, \dots, T\}$ を定義します：

$$M_{ij} = \begin{cases} 0 & j \in \mathcal{N}(i), \\ -\infty & j \notin \mathcal{N}(i). \end{cases}$$

24.3　スライディングウィンドウアテンション

24.3.1　ウィンドウサイズ w の定義

各位置 i が半径 w の局所ウィンドウのみに注意を向けるスライディングウィンドウアテンション：

$$\mathcal{N}_{\text{local}}(i) = \big\{ j \in \{1, \dots, T\} \;\big|\; |i - j| \le w \big\}.$$

すべての位置にわたって計算されるアテンションエントリの総数は $O(Tw)$ です。

24.3.2 多層受容野

L 層積み重ねると、各位置は間接的に以下の距離まで到達できます：

$$r_{\text{receptive}} = L \cdot w.$$

24.4 拡張スライディングウィンドウ (オプション)

受容野をさらに拡大するために、拡張（dilated）ウィンドウを導入できます。層 ℓ でのストライド d_ℓ を用いて：

$$\mathcal{N}^{(\ell)}_{\text{dilated}}(i) = \big\{ j \;\big|\; j \in \{i - wd_\ell, i - (w-1)d_\ell, \dots, i + wd_\ell\} \big\}.$$

24.5 グローバルアテンション

24.5.1 グローバルトークン集合 $\mathcal{G} \subset \{1, \dots, T\}$

事前に指定されたグローバルトークン集合 $\mathcal{G}$：

- 位置 $i \in \mathcal{G}$ はすべてのトークン $j \in \{1, \dots, T\}$ に注意を向けることができる,
- 位置 $i \notin \mathcal{G}$ はウィンドウ内およびすべてのグローバルトークンに注意を向ける.

したがって：

$$\mathcal{N}_{\text{full}}(i) = \begin{cases} \{1, \dots, T\} & i \in \mathcal{G}, \\ \mathcal{N}_{\text{local}}(i) \cup \mathcal{G} & i \notin \mathcal{G}. \end{cases}$$

24.5.2 グローバルトークン選択戦略

$\mathcal{G}$ はタスクに基づいて決定されます：

1. CLS トークン: 最初のトークン $\mathcal{G} = \{1\}$.
2. 質問トークン (QA タスク): すべての質問トークンを $\mathcal{G}$ とする.
3. 固定間隔: $\mathcal{G} = \{1, 1+s, 1+2s, \dots\}$.

24.6 計算複雑性分析

24.6.1 層ごとの複雑性

グローバルトークン数 $|\mathcal{G}| = g$ として：

$$\text{Attention cost per layer} = O\big(T(w+g)d_k\big).$$

実際には $g \ll T$ なので、支配項は $O(Twd_k)$ となり、線形スケーリングを達成します。

24.6.2 モデル全体の複雑性

全 L 層と FFN を含めて：

$$\text{Total cost} = O\Big(L\big(Twd_k + gTd_k + Td_{\text{model}}d_{\text{ff}}\big)\Big).$$

24.7 Longformer におけるアテンションの形式的定義

24.7.1 スコアとマスクの計算

Longformer マスク：

$$M_{ij}^{\text{Longformer}} = \begin{cases} 0 & j \in \mathcal{N}_{\text{full}}(i), \\ -\infty & j \notin \mathcal{N}_{\text{full}}(i). \end{cases}$$

24.7.2 疎なソフトマックス

行 i のソフトマックスは、非ゼロスコアに対してのみ計算されます：

$$A_{ij} = \begin{cases} \dfrac{\exp(\tilde{S}_{ij})}{\sum_{k \in \mathcal{N}_{\text{full}}(i)} \exp(\tilde{S}_{ik})} & j \in \mathcal{N}_{\text{full}}(i), \\ 0 & \text{それ以外.} \end{cases}$$

出力：

$$y_i = \sum_{j \in \mathcal{N}_{\text{full}}(i)} A_{ij} v_j.$$

24.8 疎なアテンションを通る勾配フロー

逆伝播中の勾配計算も、許可されたアテンションエントリに対してのみ実行されます。ソフトマックスのヤコビアンは：

$$\frac{\partial A_{ij}}{\partial S_{ik}} = \begin{cases} A_{ij}(\delta_{jk} - A_{ik}) & k \in \mathcal{N}_{\text{full}}(i), \\ 0 & k \notin \mathcal{N}_{\text{full}}(i). \end{cases}$$

24.9 他の効率的な Transformer との比較

24.9.1 BigBird

BigBird も疎なアテンションを使用し、ランダムアテンションとブロックスパースアテンションを追加します。

24.9.2 Linformer / Performer

Linformer は低ランク近似により $O(T)$ を達成し、アテンション行列自体を近似します。 Performer はソフトマックスのカーネルトリック線形化により $O(T)$ を達成しますが、標準的なソフトマックスアテンションと厳密には等価ではありません。Longformer の特徴は、疎な接続において正確なアテンションを維持することです。

24.10 要約

Longformer は：

- スライディングウィンドウアテンションにより $O(Tw)$ で局所依存関係を捉え、
- グローバルアテンションにより長距離依存関係と系列全体の情報集約を処理し、
- 全体として $O(T(w+g)d_k)$ の複雑性で、$T \sim 4096$ 以上の長文書を処理し、

文書レベルの NLP タスクにおける Transformer の実用性を大幅に向上させます。

Chapter 25

T5: テキスト間転移 Transformer (Text-to-Text Transfer Transformer)

25.1 アーキテクチャの概要

図 25.1 は T5 アーキテクチャを示しており、完全なエンコーダ-デコーダ Transformer 構造を使用しています。

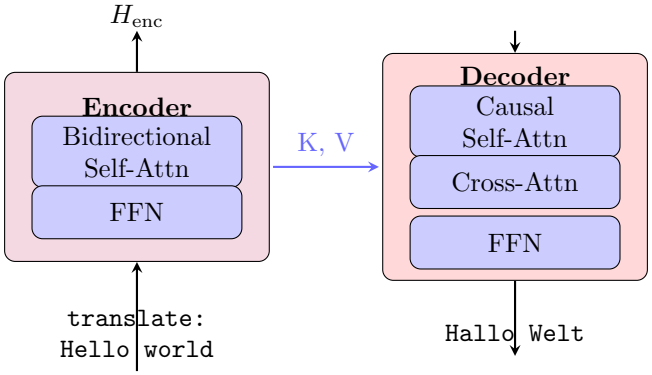


Figure 25.1: T5 アーキテクチャ: 完全なエンコーダ-デコーダ Transformer。エンコーダは入力を双方向に処理し、デコーダはクロスアテンションを介してエンコーダ表現に注意を向けながら出力を自己回帰的に生成する。

25.1.1 直感的な理解

テキスト間統一 (Text-to-Text Unification): T5 の革新的な洞察は、すべての NLP タスクをテキスト生成として枠組み化できるということです：

- **翻訳**: “translate English to German: Hello” → “Hallo”
- **要約**: “summarize: [long article]” → “[summary]”
- **分類**: “sentiment: I love this!” → “positive”
- **質問応答**: “question: What is 2+2? context: ...” → “4”

この統一されたインターフェースにより、単一のモデルで多様なタスクを処理できます。

エンコーダ-デコーダ構造: デコーダのみの GPT とは異なり、T5 は「理解」（エンコーダ）と「生成」（デコーダ）を分離します：

- **エンコーダ**: 入力全体を双方向に読み、豊かな表現を構築する
- **デコーダ**: クロスアテンションを介してエンコーダに注意を向けながら、出力を自己回帰的に生成する

スパン破損事前学習: 単一のトークンをマスクする（BERT）や次のトークンを予測する（GPT）代わりに、T5 は連続するスパンをマスクしてそれらを予測します。これにより、モデルは局所的および大域的なパターンの両方を学習します。

なぜうまくいくのか: すべてのタスクを sequence-to-sequence としてキャストすることで、T5 は同じアーキテクチャと訓練目的を普遍的に利用し、強力な転移学習を可能にします。

25.2 統一されたテキスト間フレームワーク

T5 は、すべての NLP タスクを同じ入力から出力へのテキスト変換問題として扱う統一フレームワークです。

25.2.1 条件付き生成としてのタスク定式化

語彙 $\mathcal{V}$、入力系列 $x_{1:T_x} = (x_1, \ldots, x_{T_x})$、出力系列 $y_{1:T_y} = (y_1, \ldots, y_{T_y})$、 $x_t, y_s \in \mathcal{V}$ とします。

すべてのタスクは条件付き確率モデルとして統一されます：

$$p_\theta(y_{1:T_y} \mid x_{1:T_x}) = \prod_{s=1}^{T_y} p_\theta(y_s \mid x_{1:T_x}, y_{<s})$$

これにより一様に扱います。

25.2.2 タスクプレフィックスと例

タスク情報をテキストのプレフィックスとして埋め込みます：

- 翻訳: $x =$ “translate English to German: That is good.”, $y =$ “Das ist gut.”
- 要約: $x =$ “summarize: [long document]”, $y =$ “[summary]”
- 分類: $x =$ “sentiment: This movie is great!”, $y =$ “positive”

25.3 エンコーダ-デコーダアーキテクチャ

T5 は標準的な Transformer エンコーダ-デコーダを採用します。

25.3.1 エンコーダ: 双方向自己注意

入力 $x_{1:T_x}$ に対して、エンコーダ層 $\ell = 1, \dots, L_{\text{enc}}$ は、すべての位置が互いに注意を向けることができる双方向自己注意を適用します：

$$H_{\text{enc}}^{(\ell)} = \text{EncoderBlock}^{(\ell)}\big(H_{\text{enc}}^{(\ell-1)}\big).$$

最終出力：

$$H_{\text{enc}} = H_{\text{enc}}^{(L_{\text{enc}})} \in \mathbb{R}^{T_x \times d_{\text{model}}}.$$

25.3.2 デコーダ: 因果自己注意 + クロスアテンション

デコーダ層 $\ell = 1, \dots, L_{\text{dec}}$ は3つのサブ層を持ちます：

1. マスク付き（因果）自己注意: 未来の出力トークンを見ることを防ぐために因果マスクを適用。
2. クロスアテンション: エンコーダ出力 H_{enc} から情報を読む。
3. フィードフォワードネットワーク: 位置ごとの非線形変換。

25.3.3 デコーダにおけるマスク付き自己注意

因果マスク：

$$M_{\text{causal},ij} = \begin{cases} 0 & j \le i, \\ -\infty & j > i, \end{cases}$$

これを以下のように適用：

$$\tilde{S}_{\text{self},ij} = S_{\text{self},ij} + M_{\text{causal},ij}.$$

25.3.4 エンコーダへのクロスアテンション

エンコーダ出力 H_{enc} へのクロスアテンション：

$$Q_{\text{cross}} = \hat{H}_{\text{dec}}^{(\ell)} W_Q^{\text{cross}}, \quad K_{\text{cross}} = H_{\text{enc}} W_K^{\text{cross}}, \quad V_{\text{cross}} = H_{\text{enc}} W_V^{\text{cross}}.$$

ここではマスクはありません（各デコーダ位置は入力全体を見ることができます）：

$$A_{\text{cross},ij} = \frac{\exp(S_{\text{cross},ij})}{\sum_{k=1}^{T_x} \exp(S_{\text{cross},ik})}.$$

25.3.5 デコーダブロックの構成

Pre-LN 形式では：

$$\hat{H}_{\text{dec}}^{(\ell)} = H_{\text{dec}}^{(\ell-1)} + \text{MaskedSelfAttn}\big(\text{LN}(H_{\text{dec}}^{(\ell-1)})\big),$$

$$\tilde{H}_{\text{dec}}^{(\ell)} = \hat{H}_{\text{dec}}^{(\ell)} + \text{CrossAttn}\big(\text{LN}(\hat{H}_{\text{dec}}^{(\ell)}), H_{\text{enc}}\big),$$

$$H_{\text{dec}}^{(\ell)} = \tilde{H}_{\text{dec}}^{(\ell)} + \text{FFN}\big(\text{LN}(\tilde{H}_{\text{dec}}^{(\ell)})\big).$$

25.4 相対位置バイアス

T5 は絶対位置埋め込みの代わりに、アテンションスコアに加えられる相対位置バイアスを使用します。

25.4.1 バイアスの定義

相対位置バイアス行列 $B^{(\ell)} \in \mathbb{R}^{T\times T}$ を導入：

$$S_{ij}^{(\text{with bias})} = \frac{q_i^\top k_j}{\sqrt{d_k}} + B_{ij}^{(\ell)}.$$

25.4.2 バケット化された相対位置

相対距離 $d = j - i$ に対して、学習可能なバイアステーブル $\beta^{(\ell)} \in \mathbb{R}^{N_{\text{bucket}}}$ を用意します：

$$B_{ij}^{(\ell)} = \beta_{\text{bucket}(j-i)}^{(\ell)}.$$

バケット関数は、短い距離には細かく、長い距離には粗く離散化します。

25.5 事前学習目的関数: スパン破損 (Span Corruption)

T5 の事前学習は、BERT の MLM を拡張したスパン破損タスクを使用します。

25.5.1 ランダムプロセスとしてのスパンマスキング

入力文 $x_{1:T}$ に対して、平均長 λ_{span} のスパンをランダムに選択し、特別なトークン $[\texttt{MASK}_k]$ に置き換えます。

25.5.2 ターゲット系列の構築

ターゲット $y_{1:T_y}$ は、センチネルトークンで区切られたマスクされたスパンを連結したものです：

例：

$$\begin{aligned} x &= \text{“Thank you for inviting me to your party last week.”} \\ \tilde{x} &= \text{“Thank you [X] me to your [Y] week.”} \\ y &= \text{“[X] for inviting [Y] party last [Z].”} \end{aligned}$$

25.5.3 条件付き尤度としてのノイズ除去目的関数

事前学習損失：

$$\mathcal{L}_{\text{denoise}}(\theta) = \mathbb{E}_{x_{1:T},\text{spans}}\left[-\sum_{s=1}^{T_y} \log p_\theta(y_s \mid \tilde{x}_{1:T'}, y_{<s})\right].$$

25.5.4 トークンごとの損失と勾配

位置 s におけるロジット：

$$z_s = W_{\mathrm{LM}} h_s^{(L_{\mathrm{dec}})} + b_{\mathrm{LM}} \in \mathbb{R}^{|\mathcal{V}|},$$

勾配：

$$\nabla_{z_s} \ell_s = p_s - \mathrm{one\_hot}(y_s^\star).$$

25.6 教師強制と自己回帰デコーディング

25.6.1 教師強制による訓練

訓練中、グラウンドトゥルース出力 $y_{1:T_y}$ が直接デコーダに入力されます（教師強制）。これにより、効率的な訓練のための並列計算が可能になります。

25.6.2 自己回帰生成による推論

推論中、デコーダは $y_s = [\texttt{EOS}]$ となるまで再帰的に実行されます。

25.7 単純化された層正規化 (RMSNorm)

T5 は RMSNorm を使用します：

$$\mathrm{RMSNorm}(x) = \gamma \odot \frac{x}{\mathrm{RMS}(x) + \varepsilon}, \quad \mathrm{RMS}(x) = \sqrt{\frac{1}{d_{\mathrm{model}}} \sum_{i=1}^{d_{\mathrm{model}}} x_i^2}.$$

25.8 計算複雑性

25.8.1 エンコーダ複雑性

入力長 T_x に対して、エンコーダ複雑性は：

$$O\big(L_{\mathrm{enc}} \cdot T_x^2 d_k + L_{\mathrm{enc}} \cdot T_x d_{\mathrm{model}} d_{\mathrm{ff}}\big).$$

25.8.2 デコーダ複雑性

出力長 T_y に対して：自己注意 $O(T_y^2 d_k)$、クロスアテンション $O(T_y T_x d_k)$、FFN $O(T_y d_{\mathrm{model}} d_{\mathrm{ff}})$。

25.9 訓練戦略とハイパーパラメータ

25.9.1 事前学習コーパス: C4

T5 は Colossal Clean Crawled Corpus (C4)（約 750GB のクリーンなウェブテキスト）で事前学習されます。

25.9.2 モデルサイズ

モデル	d_{model}	d_{ff}	パラメータ
T5-Small	512	2048	60M
T5-Base	768	3072	220M
T5-Large	1024	4096	770M
T5-3B	1024	16384	3B
T5-11B	1024	65536	11B

25.10 BERT および GPT との比較

モデル	アーキテクチャ	事前学習目的	ターゲットタスク
BERT	エンコーダのみ	MLM + NSP	分類/抽出
GPT	デコーダのみ	因果 LM	生成
T5	エンコーダ-デコーダ	スパン破損	全タスク統一

25.11 要約

T5 は

- すべての NLP タスクを条件付き生成 $p_\theta(y_{1:T_y} \mid x_{1:T_x})$ として統一し、
- 相対位置バイアスをエンコーダ-デコーダ Transformer に導入し、
- スパン破損によるノイズ除去事前学習を実行し、
- テキスト間フレームワークを通じて汎用的な転移学習を達成し、

多様な NLP タスクにわたる統一モデリングの可能性を実証しました。

Chapter 26

Vision Transformer (ViT): 画像分類のための Transformer

26.1 アーキテクチャの概要

図 26.1 は Vision Transformer アーキテクチャを示しており、標準的な Transformer エンコーダを画像パッチの系列に直接適用します。

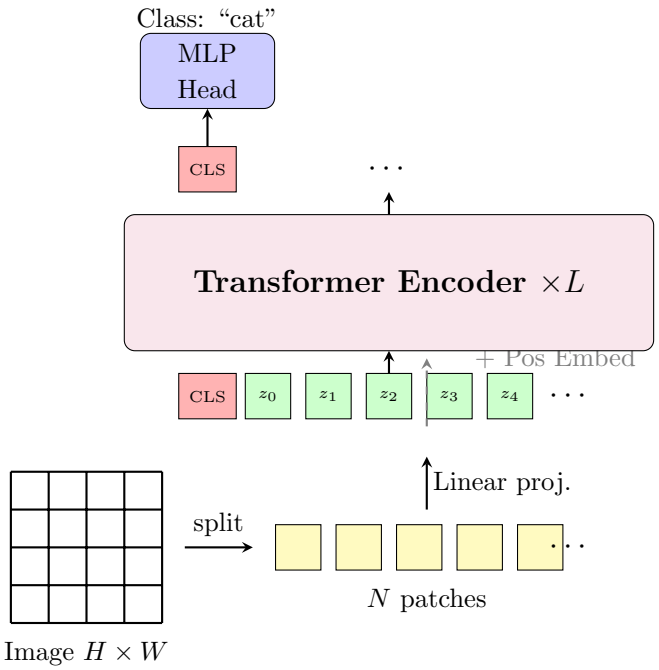


Figure 26.1: Vision Transformer (ViT) アーキテクチャ: 画像は固定サイズのパッチに分割され、線形に埋め込まれ、標準的な Transformer エンコーダで処理される。[CLS] トークンの出力が分類に使用される。

26.1.1 直感的な理解

系列としての画像: ViT の重要な洞察は、画像を2次元グリッドとしてではなく、パッチの系列として扱うことです。これは文が単語の系列であるのと同様です。

224×224 の画像を16×16のパッチにすると、196個の「視覚トークン」の系列になります。

なぜパッチなのか？ 自己注意は $O(n^2)$ の複雑さを持ちます。個々のピクセル（224×224 = 50,176 トークン）を使用するのは計算的に不可能です。パッチ（14×14 = 196 トークン）にすることで、各パッチ内の局所構造を維持しつつ、処理が可能になります。

畳み込みなし: ViT は、CNN の帰納バイアス（局所性、並進不変性）が厳密には必要ないことを示しています。十分なデータがあれば、自己注意はこれらのパターンをゼロから学習し、より柔軟な表現を発見する可能性があります。

[CLS] トークン: BERT から借用した、学習可能な「クラス」トークンがパッチ系列の先頭に付加されます。すべての層を通過した後、このトークンの表現は画像の全体的な要約を含んでおり、分類ヘッドに入力されます。

データへの渇望: CNN の帰納バイアスがないため、ViT は事前学習に大規模なデータセット（ImageNet-21k, JFT-300M）を必要とします。より小さなデータセットでは、CNN が依然として ViT を上回ります。これは柔軟性とデータ効率の間のトレードオフを浮き彫りにします。

位置埋め込み: 空間的な関係を本質的に理解する CNN とは異なり、ViT は学習可能な埋め込みを通じて位置を学習しなければなりません。興味深いことに、これらの学習された埋め込みは、パッチの位置と一致する2次元の空間構造を示します。

26.2 動機: 畳み込みから自己注意へ

従来の画像分類は、局所的な特徴抽出と階層的な表現学習を行う畳み込みニューラルネットワーク（CNN）によって支配されていました。 Vision Transformer (ViT) は画像を系列データとして扱い、標準的な Transformer エンコーダを直接適用し、CNN の帰納バイアス（局所性、並進不変性）を取り除き、純粋な自己注意メカニズムを通じて大域的な依存関係の学習を可能にします。

26.3 系列としての画像: パッチ埋め込み

26.3.1 パッチへの画像分割

入力画像を $I \in \mathbb{R}^{H\times W\times C}$ とします。画像を $P \times P$ ピクセルの正方形パッチに分割します。パッチ数：

$$N = \frac{H \cdot W}{P^2}.$$

通常 $H = W = 224$, $P = 16$ で $N = 196$ となります。

26.3.2 テンソル再形成としてのパッチ抽出

画像 I を N 個のパッチベクトル $\{\mathbf{x}_p^{(i)}\}_{i=1}^{N}$, $\mathbf{x}_p^{(i)} \in \mathbb{R}^{P^2C}$ に分割します。

26.3.3 埋め込み次元への線形射影

各パッチベクトルをモデル次元 d_{model} に線形射影します：

$$\mathbf{z}_i^{(0)} = E\mathbf{x}_p^{(i)} + \mathbf{b}, \quad E \in \mathbb{R}^{d_{\text{model}}\times(P^2C)}.$$

26.4 クラストークンの付加

分類タスクのために、学習可能なクラストークン $\mathbf{z}_{\text{cls}} \in \mathbb{R}^{d_{\text{model}}}$ を系列の先頭に追加します：

$$Z_{\text{full}}^{(0)} = \begin{bmatrix} \mathbf{z}_{\text{cls}}^\top \\ (\mathbf{z}_1^{(0)})^\top \\ \vdots \\ (\mathbf{z}_N^{(0)})^\top \end{bmatrix} \in \mathbb{R}^{(N+1)\times d_{\text{model}}}.$$

26.5 位置エンコーディング

26.5.1 学習可能な 1D 位置埋め込み

各位置 $i \in \{0, 1, \ldots, N\}$ に対して学習可能な位置埋め込みを追加します：

$$\mathbf{z}_i^{(0)\prime} = \mathbf{z}_i^{(0)} + \mathbf{e}_{\text{pos}}^{(i)}.$$

位置埋め込み行列 $E_{\text{pos}} \in \mathbb{R}^{(N+1)\times d_{\text{model}}}$ は訓練によって学習されます。

26.5.2 入力埋め込みの合成

最終的な入力埋め込み：

$$H^{(0)} = Z_{\text{full}}^{(0)} + E_{\text{pos}} \in \mathbb{R}^{(N+1)\times d_{\text{model}}}.$$

26.6 Transformer エンコーダ

ViT は L 層の標準的な Transformer エンコーダを積み重ねます。

26.6.1 マルチヘッド自己注意

層 ℓ の入力 $H^{(\ell-1)} \in \mathbb{R}^{(N+1)\times d_{\text{model}}}$ に対して、 ViT はマスクなしの双方向アテンションを使用します：

$$A_{ij}^{(h)} = \frac{\exp(S_{ij}^{(h)})}{\sum_{k=0}^{N} \exp(S_{ik}^{(h)})}.$$

26.6.2 層正規化と残差接続

Pre-LN 形式では：

$$\hat{H}^{(\ell)} = H^{(\ell-1)} + \text{MHA}\big(\text{LN}(H^{(\ell-1)})\big),$$

$$H^{(\ell)} = \hat{H}^{(\ell)} + \text{FFN}\Big(\text{LN}(\hat{H}^{(\ell)})\Big).$$

26.6.3 位置ごとのフィードフォワードネットワーク

ViT は活性化関数として GELU を使用します：

$$\mathrm{FFN}(u) = W_2\,\mathrm{GELU}(W_1 u + b_1) + b_2,$$

$$\mathrm{GELU}(x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right].$$

26.7 分類ヘッド

26.7.1 CLS トークンの抽出

最終層 L の出力から CLS トークン表現 $\mathbf{h}_{\mathrm{cls}}^{(L)} \in \mathbb{R}^{d_{\mathrm{model}}}$ を抽出します。

26.7.2 線形分類層

K-クラス分類に対して：

$$\mathbf{z}_{\mathrm{cls}} = W_{\mathrm{head}}\mathbf{h}_{\mathrm{cls}}^{(L)} + \mathbf{b}_{\mathrm{head}} \in \mathbb{R}^K,$$

予測分布：

$$p_\theta(y = k \mid I) = \frac{\exp(z_{\mathrm{cls},k})}{\sum_{j=1}^{K} \exp(z_{\mathrm{cls},j})}.$$

26.7.3 交差エントロピー損失

ワンホットラベル $\mathbf{y} \in \{0,1\}^K$ に対して：

$$\mathcal{L}_{\mathrm{CE}}(\theta) = -\sum_{k=1}^{K} y_k \log p_\theta(y = k \mid I).$$

勾配：

$$\nabla_{\mathbf{z}_{\mathrm{cls}}}\mathcal{L}_{\mathrm{CE}} = p_\theta - \mathbf{y}, \quad \nabla_{W_{\mathrm{head}}}\mathcal{L}_{\mathrm{CE}} = (p_\theta - \mathbf{y})(\mathbf{h}_{\mathrm{cls}}^{(L)})^\top.$$

26.8 事前学習と転移学習

26.8.1 大規模データセットでの事前学習

ViT は大規模データセットでの事前学習を通じて高い性能を達成します。

- ImageNet-21k: 約 1400 万枚の画像、21,000 クラス,
- JFT-300M: 約 3 億枚の画像.

26.8.2 ターゲットデータセットでのファインチューニング

下流タスクのために、分類ヘッドを再初期化し、すべてのパラメータを更新します。より高解像度の画像でファインチューニングする場合、位置埋め込みを2次元補間します。

26.9 モデルのバリエーションとスケーリング

モデル	層数 L	隠れ次元 d_{model}	ヘッド数 H	パラメータ
ViT-Base	12	768	12	86M
ViT-Large	24	1024	16	307M
ViT-Huge	32	1280	16	632M

26.10 計算複雑性

26.10.1 自己注意の複雑性

系列長 $N+1$ に対する自己注意：

$$O\big((N+1)^2 d_k\big) = O\left(\left(\frac{HW}{P^2}\right)^2 d_k\right).$$

より大きな P はより小さな N を意味し、計算量を削減します。

26.10.2 CNN との比較

CNN は局所的な受容野のため、画像サイズに対して線形です。 ViT は $O((HW/P^2)^2)$ であり、パッチサイズ P によって制御可能です。

26.11 帰納バイアスとデータ効率

CNN は局所性と並進不変性の帰納バイアスを持ちますが、 ViT は自己注意を介してすべてのパッチ間の大域的な依存関係を直接学習し、これらのバイアスを欠いています。弱い帰納バイアスのため、大規模データセットでの事前学習なしでは ViT は CNN よりも性能が劣ります。

26.12 拡張と派生形

26.12.1 DeiT (Data-efficient image Transformer)

蒸留を使用して、より小さなデータセットでの訓練効率を向上させます。

26.12.2 Swin Transformer

階層構造とシフトウィンドウ自己注意により、局所性を持ちながら効率的な計算を実現します。

26.12.3 BEiT

画像パッチをマスクして予測することにより、BERT スタイルの自己教師あり学習を行います。

26.13 要約

Vision Transformer (ViT) は：

- 画像を $P \times P$ のパッチに分割し、線形射影によって埋め込み、
- CLS トークンと学習可能な位置埋め込みを追加し、
- 標準的な Transformer エンコーダを適用し、双方向自己注意を介して大域的な依存関係を学習し、
- CLS トークン表現から線形分類ヘッドによって予測し、
- 大規模な事前学習を通じて CNN を超える性能を達成し、

視覚タスクに Transformer を直接適用するパラダイムを確立しました。

Chapter 27

PaLM: Pathways Language Model

27.1　概要とスケーリング哲学

PaLM (Pathways Language Model) は Google 開発の超大規模デコーダのみの言語モデルで、最大の構成は **540B パラメータ** を持ちます。 GPT ファミリーの自己回帰言語モデルの設計に従いつつ、以下の革新により効率と性能を向上させています：

- **Pathways システム**: 数千の TPU にわたる超並列分散学習、
- **SwiGLU 活性化**: FFN の表現力向上、
- **並列層**: Attention と FFN の並列実行、
- **マルチクエリアテンション**: 効率的な KV キャッシュ、
- **RoPE**: 相対位置のための Rotary Position Embedding.

27.2　自己回帰言語モデリング

PaLM はトークン系列 $x_{1:T}$ の自己回帰分布を学習します：

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}),$$

訓練目的関数は負の対数尤度最小化です：

$$\mathcal{L}_{\mathrm{LM}}(\theta) = \mathbb{E}_{x_{1:T}\sim\mathcal{D}}\left[-\frac{1}{T}\sum_{t=1}^{T}\log p_\theta(x_t \mid x_{1:t-1})\right].$$

27.3　並列層アーキテクチャ

標準的な Transformer では、各層は逐次的でした：Attention → Add&Norm → FFN → Add&Norm。 PaLM は Attention と FFN を並列化します：

$$H^{(\ell)} = H^{(\ell-1)} + \mathrm{Attention}^{(\ell)}\big(\mathrm{LN}(H^{(\ell-1)})\big) + \mathrm{FFN}^{(\ell)}\big(\mathrm{LN}(H^{(\ell-1)})\big).$$

27.4 マルチクエリアテンション

マルチクエリアテンションは、すべてのヘッドで K, V を共有します：

$$Q^{(h)} = HW_Q^{(h)} \in \mathbb{R}^{T\times d_k}, \quad h = 1, \ldots, H,$$

$$K = HW_K \in \mathbb{R}^{T\times d_k}, \quad V = HW_V \in \mathbb{R}^{T\times d_v},$$

ここで W_K, W_V はヘッド間で共有される単一の行列です。 KV キャッシュサイズは $O(H \cdot T \cdot d_k)$ から $O(T \cdot d_k)$ に削減されます。

27.5 RoPE: Rotary Position Embedding

RoPE は回転行列を介してアテンションスコアに相対位置情報を埋め込みます。位置 t に対して、回転を適用します：

$$\tilde{q}_t = R_t q_t, \quad \tilde{k}_t = R_t k_t,$$

ここで R_t はブロック対角回転行列です。スコアは以下のようになり：

$$S_{ij} = \frac{q_i^\top R_{j-i} k_j}{\sqrt{d_k}},$$

相対位置 $j - i$ のみに依存します。

27.6 SwiGLU 活性化

PaLM は SwiGLU を採用します：

$$\mathrm{SwiGLU}(x) = (W_1 x) \odot \mathrm{Swish}(W_2 x),$$

$$\mathrm{Swish}(z) = z \cdot \sigma(z) = \frac{z}{1 + e^{-z}}.$$

27.7 モデル構成

モデル	層数	d_{model}	ヘッド数	d_{ff}	パラメータ
PaLM-8B	32	4096	32	16384	8B
PaLM-62B	64	8192	64	32768	62B
PaLM-540B	118	18432	48	49152	540B

27.8 要約

PaLM は、RoPE, SwiGLU, 並列層, マルチクエリアテンション, および数千の TPU にわたる Pathways 分散学習を通じて、スケーリングと効率を達成しました。

Chapter 28

LLaMA: Large Language Model Meta AI

28.1 概要と設計哲学

LLaMA (Large Language Model Meta AI) は Meta が開発したオープンソースの大規模デコーダのみの言語モデルです。主な設計上の特徴は以下の通りです：

- **RMSNorm**: 効率のための単純化された LayerNorm,
- **SwiGLU 活性化**: FFN の表現力向上,
- **RoPE**: 効率的な相対位置エンコーディング,
- **Pre-normalization**: 訓練の安定性,
- **公開データのみ**: 透明性と再現性.

LLaMA は 7B, 13B, 33B, 65B の4つのサイズを提供します。

28.2 RMSNorm: 二乗平均平方根層正規化

LLaMA は RMSNorm を採用します：

$$\mathrm{RMSNorm}(x) = \gamma \odot \frac{x}{\mathrm{RMS}(x)},$$

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \varepsilon},$$

ここで $\gamma \in \mathbb{R}^d$ は学習可能なスケールパラメータです。 LayerNorm とは異なり、RMSNorm は平均シフトとバイアス項を省略します。

28.3 Pre-Normalization アーキテクチャ

LLaMA は Pre-LN 構造を使用します：

$$\tilde{H}^{(\ell)} = H^{(\ell-1)} + \mathrm{Attention}^{(\ell)}\big(\mathrm{RMSNorm}(H^{(\ell-1)})\big),$$

$$H^{(\ell)} = \tilde{H}^{(\ell)} + \mathrm{FFN}^{(\ell)}\Big(\mathrm{RMSNorm}(\tilde{H}^{(\ell)})\Big).$$

28.4 RoPE を用いた因果自己注意

位置 t でのクエリとキーに対して、回転行列 R_t を適用します：

$$\tilde{Q}_t^{(h)} = R_t Q_t^{(h)}, \quad \tilde{K}_t^{(h)} = R_t K_t^{(h)}.$$

スコアは相対位置のみに依存します：

$$S_{ij}^{(h)} = \frac{(Q_i^{(h)})^\top R_{j-i} K_j^{(h)}}{\sqrt{d_k}}.$$

28.5 SwiGLU フィードフォワードネットワーク

$$\mathrm{SwiGLU}(x) = (W_1 x) \odot \mathrm{Swish}(W_2 x),$$
$$\mathrm{FFN}(x) = W_3 \,\mathrm{SwiGLU}(x).$$

バイアス項は使用されません（LLaMA はすべての層でバイアスを省略します）。

28.6 モデル構成

モデル	層数	d_{model}	ヘッド数	d_{ff}	パラメータ
LLaMA-7B	32	4096	32	11008	6.7B
LLaMA-13B	40	5120	40	13824	13.0B
LLaMA-33B	60	6656	52	17920	32.5B
LLaMA-65B	80	8192	64	22016	65.2B

28.7 訓練データ

LLaMA は公開データセットのみで訓練されています（約 1.4T トークン）： CommonCrawl (67%), C4 (15%), GitHub (4.5%), Wikipedia (4.5%), Books (4.5%), ArXiv (2.5%), StackExchange (2%)。

28.8 LLaMA 2 の改善点

LLaMA 2 は以下を導入しました：

- Grouped-Query Attention (GQA): マルチヘッドとマルチクエリのバランス,
- より長いコンテキストウィンドウ: $T_{\max} = 4096$,
- 安全性アライメントのための RLHF.

28.9 要約

LLaMA は、RMSNorm, SwiGLU, RoPE, Pre-normalization, および公開データ訓練を通じて、効率的で透明性の高い大規模言語モデリングを達成しました。

Chapter 29

Mixtral: 疎な混合エキスパート (Sparse Mixture of Experts) 言語モデル

29.1 アーキテクチャの概要

図 29.1 は Mixtral で使用されている疎な混合エキスパート (Sparse Mixture of Experts, MoE) アーキテクチャを示しています。

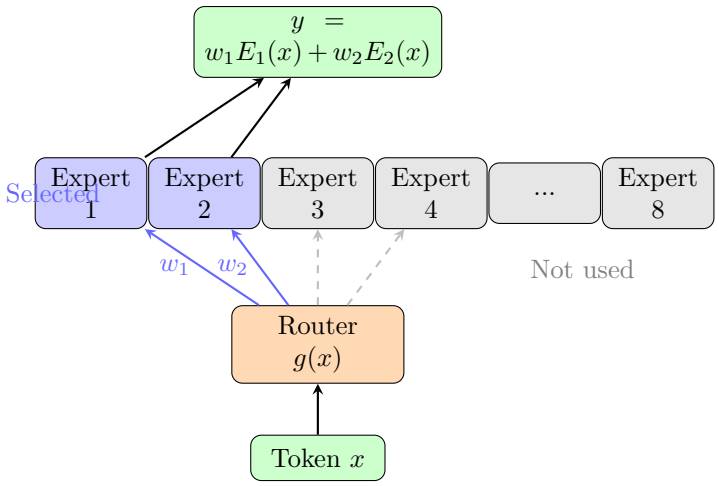


Figure 29.1: Mixtral 疎な MoE アーキテクチャ: ゲーティングネットワークが各トークンを上位2つのエキスパート（全8つ中）にルーティングする。選択されたエキスパートのみが計算され、総パラメータ47Bに対して13Bのアクティブ計算を実現する。

29.1.1 直感的な理解

エキスパートの例え: 専門医（循環器科、神経科、皮膚科など）がいる病院を想像してください。患者が到着すると、トリアージシステム（ルーター）は、すべての医師ではなく、最も関連性の高い1〜2人の専門医に患者を案内します。同様に、MoE は各トークンをそれに最適なエキスパートにルーティングします。

疎 = 効率的: 8つのエキスパートがあり、トークンごとに2つしかアクティブにならないため、Mixtral はエキスパート計算の $\frac{2}{8} = 25\%$ を使用します。総パラメータ (47B) は容量を提供し、アクティブパラメータ (13B) は速度を決定します。これにより、モデル容量と推論コストが分離されます。

なぜうまくいくのか: 異なるトークンは異なる計算を必要とする場合があります：

- コードトークン → プログラミングパターンを専門とするエキスパート
- 数学トークン → 数値推論を専門とするエキスパート
- 言語トークン → 言語パターンを専門とするエキスパート

ルーターはこれらの割り当てを自動的に学習します。

負荷分散の課題: 制約がないと、ルーターは常に同じエキスパートを選ぶ可能性があります（「エキスパート崩壊」）。補助損失は、均等なエキスパート利用を促進します。

重要な洞察: Mixtral は、条件付き計算（入力ごとに異なるパラメータを使用すること）が、密なモデルよりも FLOP あたりの品質が高い、有望なスケーリング方向であることを示しています。

29.2 疎な MoE アーキテクチャの詳細

Mixtral は Mistral AI によって開発された、疎な混合エキスパート (Sparse Mixture of Experts, SMoE) アーキテクチャを持つ大規模言語モデルです。主な特徴：

- **8つのエキスパート FFN**: 層ごとに複数のスペシャリストネットワーク,
- **Top-2 ルーティング**: 入力ごとに最適な2つのエキスパートを選択,
- **合計 47B パラメータ**: 全エキスパートパラメータの合計,
- **アクティブ 13B パラメータ**: 推論中に実際に使用される,
- **負荷分散**: 均等なエキスパート利用のための補助損失.

これにより、47B パラメータの表現力を 13B モデルの計算コストで実現します。

29.3 混合エキスパートの定式化

$N = 8$ 個のエキスパートネットワーク、それぞれが SwiGLU 構造を持ちます：

$$\text{Expert}_e(x) = W_{3,e}\,\text{SwiGLU}(x).$$

ゲーティングネットワークがルーティング確率を計算します：

$$g(x) = \text{softmax}(W_g x) \in \mathbb{R}^N.$$

29.4 Top-k ルーティング

Top-2 ルーティング ($k = 2$) は、最もスコアの高い2つのエキスパートを選択します：

$$\mathcal{T}_2(x) = \{e_1, e_2\},$$

$$y = \tilde{g}_{e_1}(x) \cdot \text{Expert}_{e_1}(x) + \tilde{g}_{e_2}(x) \cdot \text{Expert}_{e_2}(x),$$

ここで $\tilde{g}_e$ は選択されたエキスパート上で再正規化された重みです。

計算は密な MoE の $k/N = 2/8 = 1/4$ に削減されます。

29.5 負荷分散損失

エキスパートの負荷不均衡を防ぐために、補助損失が導入されます：

$$f_e = \frac{1}{B}\sum_{i=1}^{B} g_e(x_i), \quad P_e = \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}\{e \in \mathcal{T}_k(x_i)\},$$

$$\mathcal{L}_{\text{balance}} = \alpha \cdot N \sum_{e=1}^{N} f_e \cdot P_e.$$

合計損失：

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + \mathcal{L}_{\text{balance}}.$$

29.6 スライディングウィンドウアテンション

Mixtral は長いコンテキスト（32k トークン）のためにスライディングウィンドウアテンションを使用します：ウィンドウサイズ $w = 4096$、位置 i は $[i - w, i]$ に注意を向けます。アテンションを $O(T^2 d_k)$ から $O(Twd_k)$ に削減します。

29.7 パラメータ数

- **合計**: 47B パラメータ (層あたり8エキスパート),
- **アクティブ**: 13B パラメータ (top-2 エキスパート).

29.8 密なモデルとの比較

モデル	合計パラメータ	アクティブパラメータ	推論コスト
LLaMA-2-13B	13B	13B	1.0×
LLaMA-2-70B	70B	70B	5.4×
Mixtral-8x7B	47B	13B	1.0×

Mixtral は 13B の計算コストで 70B モデルの性能を達成します。

29.9 要約

Mixtral は以下を通じて画期的な効率を達成しました：

- 8つのエキスパート FFN を持つ疎な MoE アーキテクチャ,
- 入力ごとに最適な2つのエキスパートを選択する Top-2 ルーティング,
- 47B 合計 / 13B アクティブパラメータ,
- エキスパート利用のための負荷分散損失,
- 32k コンテキストのためのスライディングウィンドウアテンション,
- LLaMA スタイルの RMSNorm, SwiGLU, RoPE.

Mixtral は、疎な MoE が将来の LLM スケーリングの有望な方向であることを実証しています。

Chapter 30

完全な誤差逆伝播法のウォークスルー: RNN から Transformer まで

30.1 パート I: 単純な RNN - 完全な誤差逆伝播の例

30.1.1 設定: 具体的な数値を用いた2層 RNN

アーキテクチャ:

- 入力次元: $d_x = 2$
- 隠れ次元: $d_h = 3$
- 出力次元: $d_y = 1$ (二値分類)
- 系列長: $T = 3$
- 学習率: $\alpha = 0.1$

パラメータ:

$$\mathbf{W}_{hh} = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.1 & 0.2 \\ 0.2 & 0.3 & 0.1 \end{pmatrix}, \quad \mathbf{W}_{xh} = \begin{pmatrix} 0.5 & 0.6 \\ 0.3 & 0.4 \\ 0.2 & 0.1 \end{pmatrix} \tag{30.1}$$

$$\mathbf{W}_{hy} = \begin{pmatrix} 0.7 & 0.5 & 0.3 \end{pmatrix}, \quad \mathbf{b}_h = \begin{pmatrix} 0.1 \\ 0.0 \\ 0.1 \end{pmatrix}, \quad b_y = 0.05 \tag{30.2}$$

活性化関数: 隠れ層: $\sigma_h(z) = \tanh(z)$, 出力層: $\sigma_y(z) = \sigma(z) = \frac{1}{1+e^{-z}}$.

入力系列:

$$\mathbf{x}^{(1)} = \begin{pmatrix} 0.5 \\ 0.2 \end{pmatrix}, \quad \mathbf{x}^{(2)} = \begin{pmatrix} 0.3 \\ 0.4 \end{pmatrix}, \quad \mathbf{x}^{(3)} = \begin{pmatrix} 0.2 \\ 0.5 \end{pmatrix} \tag{30.3}$$

ターゲット: $y = 1$.

30.1.2 順伝播 (Forward Propagation)

タイムステップ 1

$$\mathbf{z}_h^{(1)} = \mathbf{W}_{hh}\mathbf{h}^{(0)} + \mathbf{W}_{xh}\mathbf{x}^{(1)} + \mathbf{b}_h \tag{30.4}$$

$$= \mathbf{0} + \begin{pmatrix} 0.47 \\ 0.23 \\ 0.22 \end{pmatrix} \quad \text{(計算後)} \tag{30.5}$$

$$\mathbf{h}^{(1)} = \tanh(\mathbf{z}_h^{(1)}) \approx \begin{pmatrix} 0.440 \\ 0.226 \\ 0.216 \end{pmatrix} \tag{30.6}$$

$$\hat{y}^{(1)} = \sigma(\mathbf{W}_{hy}\mathbf{h}^{(1)} + b_y) = \sigma(0.5358) \approx 0.631 \tag{30.7}$$

タイムステップ 2

$$\mathbf{z}_h^{(2)} = \begin{pmatrix} 0.645 \\ 0.452 \\ 0.357 \end{pmatrix}, \quad \mathbf{h}^{(2)} \approx \begin{pmatrix} 0.567 \\ 0.422 \\ 0.343 \end{pmatrix}, \quad \hat{y}^{(2)} \approx 0.682 \tag{30.8}$$

タイムステップ 3

$$\mathbf{z}_h^{(3)} \approx \begin{pmatrix} 0.812 \\ 0.641 \\ 0.485 \end{pmatrix}, \quad \mathbf{h}^{(3)} \approx \begin{pmatrix} 0.675 \\ 0.568 \\ 0.449 \end{pmatrix}, \quad \hat{y}^{(3)} \approx 0.720 \tag{30.9}$$

30.1.3 損失計算

二値交差エントロピー ($y = 1$):

$$L = -\log(0.720) \approx 0.329 \tag{30.10}$$

30.1.4 通時的誤差逆伝播 (BPTT)

出力デルタ (タイムステップ 3)

$$\boldsymbol{\delta}_y^{(3)} = \hat{y}^{(3)} - y = 0.720 - 1 = -0.280 \tag{30.11}$$

隠れ層デルタ (タイムステップ 3)

$$\boldsymbol{\delta}_h^{(3)} = (\mathbf{W}_{hy}^T \boldsymbol{\delta}_y^{(3)}) \odot \sigma_h'(\mathbf{z}_h^{(3)}) \tag{30.12}$$

$$= \begin{pmatrix} -0.196 \\ -0.140 \\ -0.084 \end{pmatrix} \odot \begin{pmatrix} 0.544 \\ 0.677 \\ 0.798 \end{pmatrix} = \begin{pmatrix} -0.107 \\ -0.095 \\ -0.067 \end{pmatrix} \tag{30.13}$$

隠れ層デルタ (タイムステップ 2)

再帰接続からの勾配のみ（単純化のため出力損失は $T = 3$ でのみ発生すると仮定）:

$$\boldsymbol{\delta}_h^{(2)} = (\mathbf{W}_{hh}^T \boldsymbol{\delta}_h^{(3)}) \odot \sigma_h'(\mathbf{z}_h^{(2)}) \tag{30.14}$$

$$= \begin{pmatrix} -0.0621 \\ -0.051 \\ -0.0578 \end{pmatrix} \odot \begin{pmatrix} 0.679 \\ 0.822 \\ 0.882 \end{pmatrix} = \begin{pmatrix} -0.0422 \\ -0.0419 \\ -0.0510 \end{pmatrix} \tag{30.15}$$

隠れ層デルタ (タイムステップ 1)

$$\boldsymbol{\delta}_h^{(1)} \approx \begin{pmatrix} -0.0252 \\ -0.0265 \\ -0.0157 \end{pmatrix} \tag{30.16}$$

パラメータ勾配 (合計)

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} \approx \begin{pmatrix} -0.0792 & -0.0547 & -0.0458 \\ -0.0723 & -0.0496 & -0.0416 \\ -0.0604 & -0.0398 & -0.0340 \end{pmatrix} \tag{30.17}$$

30.2 パート II: LSTM - 完全な誤差逆伝播の例

30.2.1 設定

入力 $\mathbf{x} \in \mathbb{R}^2$, 隠れ層 $\mathbf{h} \in \mathbb{R}^2$. 2 タイムステップ。単純化した重み: $\mathbf{W} \approx 0.1\mathbf{I}$.

30.2.2 順伝播 (抽象的)

各タイムステップ t で:

- ゲート: $\mathbf{f}, \mathbf{i}, \mathbf{o}, \tilde{\mathbf{c}}$ を線形写像 + シグモイド/tanh で計算。
- セル: $\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}$
- 隠れ層: $\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)})$

30.2.3 逆伝播のロジック

RNN との決定的な違いは、$\mathbf{c}$ を通る勾配の流れである。

$$\frac{\partial L}{\partial \mathbf{c}^{(t)}} = \frac{\partial L}{\partial \mathbf{h}^{(t)}} \odot \mathbf{o}^{(t)} \odot (1 - \tanh^2 \mathbf{c}^{(t)}) + \frac{\partial L}{\partial \mathbf{c}^{(t+1)}} \odot \mathbf{f}^{(t+1)} \tag{30.18}$$

項 $\odot \mathbf{f}^{(t+1)}$ により、$\mathbf{f} \approx 1$ であれば勾配は減衰せずに持続する。

30.3 パート III: Transformer - 完全な誤差逆伝播の例

30.3.1 設定: 最小限のアテンション

$T = 2, d_{\text{model}} = 4, h = 2$. $\mathbf{X} \in \mathbb{R}^{4\times 2}$ (特徴 $\times$ トークン、テキストでは通常 $T \times d$ だがここでは転置記法に注意)。

30.3.2 順伝播: マルチヘッドアテンション

射影: $\mathbf{Q}_1 = \mathbf{W}_{Q_1}\mathbf{X}$. **スコア:** $\mathbf{E}_1 = \mathbf{Q}_1\mathbf{K}_1^T/\sqrt{d_k}$. **ソフトマックス:** $\mathbf{A}_1 = \text{softmax}(\mathbf{E}_1)$ (トークンごとの列方向). **出力:** $\text{head}_1 = \mathbf{A}_1\mathbf{V}_1^T$.

30.3.3 逆伝播: 勾配

Value を通る勾配:

$$\frac{\partial L}{\partial \mathbf{V}_1} = \frac{\partial L}{\partial \text{head}_1}\mathbf{A}_1 \tag{30.19}$$

アテンション行列を通る勾配:

$$\frac{\partial L}{\partial \mathbf{A}_1} = \left(\frac{\partial L}{\partial \text{head}_1}\right)^T \mathbf{V}_1 \tag{30.20}$$

ソフトマックス（ヤコビアン）を通る勾配: 各列 j（トークンスコア）に対して:

$$\frac{\partial A_{ij}}{\partial E_{kj}} = A_{ij}(\delta_{ik} - A_{kj}) \tag{30.21}$$

これは $\partial L/\partial \mathbf{A}$ と $\partial L/\partial \mathbf{E}$ を接続する。

Q, K を通る勾配:

$$\frac{\partial L}{\partial \mathbf{Q}_1} \propto \frac{\partial L}{\partial \mathbf{E}_1}\mathbf{K}_1 \tag{30.22}$$

30.4 逆伝播の複雑性まとめ

コンポーネント	順伝播の複雑性	主な課題
RNN	$O(d_h^2)$	勾配消失
LSTM	$O(4d_h^2)$	加法パスの安定性
Attention	$O(T^2d)$	ソフトマックスヤコビアンの構造

手法	計算対象	統計量の範囲	最適な用途
Batch Norm	μ_B, σ_B	バッチ	大バッチ, CNN
Layer Norm	μ, σ (サンプル毎)	層の特徴	RNN, Transformer
Group Norm	μ, σ (グループ毎)	チャネルのグループ	小バッチ
Instance Norm	μ, σ (チャネル毎)	チャネル	スタイル変換

Index