機械学習基礎

赤穂 昭太郎

2025 年 4 月 7 日

第 6 章

ニューラルネットワーク

6.1　ニューラルネットワークの基礎

6.1.1　単一ニューロンモデルと線形識別の関係

ニューラルネットワークの基本構成要素は、**単一のニューロン（ユニット）**である。これは入力ベクトル $\boldsymbol{x} \in \mathbb{R}^d$ に対して、次のような処理を行う関数である：

$$z = \boldsymbol{w}^\top \boldsymbol{x} + b, \qquad y = \phi(z)$$

ここで：

- $\boldsymbol{w} \in \mathbb{R}^d$ は重みベクトル
- $b \in \mathbb{R}$ はバイアス項（しばしば $\boldsymbol{x}$ に定数 1 を追加して内積に吸収する）
- $\phi(\cdot)$ は活性化関数（activation function）

1. 線形識別モデルとの対応

活性化関数として**ステップ関数**（単位ステップ）を用いた場合、出力 y は以下のようになる：

$$y = \begin{cases} 1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

これは、データを線形識別面 $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ によって分類する**線形識別器**に一致する。特に、古典的な**パーセプトロンアルゴリズム**ではこの形式が用いられていた。

2. ロジスティック回帰との関係

ステップ関数の代わりに、**シグモイド関数**を活性化関数として用いると、出力は次のようになる：

$$\phi(z) = \sigma(z) = \frac{1}{1 + e^{-z}}, \qquad y = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

この出力 y は $[0, 1]$ の値をとる連続関数であり、**クラス 1 に属する確率**と解釈することができる。

すなわち、このモデルはそのまま**ロジスティック回帰モデル**に一致する。

■最尤推定との関係　ロジスティック回帰では、データ $(\boldsymbol{x}_i, y_i)$ に対してベルヌーイ分布を仮定し、パラメータ $\boldsymbol{w}, b$ を最尤推定する：

$$\ell(\boldsymbol{w}, b) = \sum_{i=1}^{n} \left[y_i \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b) + (1 - y_i) \log(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b)) \right]$$

この対数尤度関数を最大化することが、学習の目的となる。

3. まとめ

単一ニューロンモデルは、活性化関数の選択に応じて、以下のような従来の機械学習モデルと対応する：

- ステップ関数：線形識別器（パーセプトロン）
- シグモイド関数：ロジスティック回帰
- 恒等関数（$\phi(z) = z$）：線形回帰

このように、**単一ニューロンは、古典的な機械学習モデルを統一的に表現できる基本構造**となっており、これを多層構造に拡張することでニューラルネットワークが構成される。

パーセプトロンの誤り訂正学習と収束定理

線形識別を行う単一ニューロンモデルに対して、パーセプトロンアルゴリズムは非常に単純な学習則に基づく。2 クラス分類問題を想定し、ラベル $y_i \in \{-1, +1\}$、特徴ベクトル $\boldsymbol{x}_i \in \mathbb{R}^d$ に対し、分類関数は以下のように定義される：

$$f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

パーセプトロンの学習則は、誤分類があった場合にのみ、以下のようにパラメータを更新する：

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta y_i \boldsymbol{x}_i, \qquad b_{t+1} = b_t + \eta y_i$$

ここで $\eta > 0$ は学習率である。これは「**誤ったときにのみ修正する**」という誤り訂正学習（error-correction learning）であり、非常に単純で局所的な学習則である。

■パーセプトロンの収束定理　このアルゴリズムは、以下の条件のもとで有限回の更新で停止する（収束する）ことが知られている。

定理 1（パーセプトロンの収束定理）　トレーニングデータ $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ が線形分離可能であると仮定する。このとき、パーセプトロンアルゴリズムは、**有限回の更新で誤分類がなくなるパラメータ $\boldsymbol{w}$ に到達する**。

証明では、真の分離超平面との内積の増加と、重みベクトルのノルムの増加に上界があることを用いて、更新回数に上限があることを示す（詳細な導出は後述の補足で可能）。

—

神経科学との関係：小脳におけるパーセプトロンモデル

パーセプトロンは元来、**生物の神経細胞（ニューロン）の振る舞い**にヒントを得て設計されたモデルである。

特に、Marr（1969）、Albus（1971）らによって提唱された **小脳パーセプトロン説（Marr – Albus theory）** は、小脳が誤り訂正学習に基づくパーセプトロンのよう

な学習装置であるという仮説である。

■小脳パーセプトロン説の要点

- 小脳のプルキンエ細胞（Purkinje cell）が出力ニューロンとして機能し、多数の平行線維からの入力を統合している。
- 誤差情報は下オリーブ核から登上線維（climbing fiber）を通じて伝えられ、**誤りが生じたときのみシナプス重みが修正される**。
- この仕組みはパーセプトロンの誤り訂正学習と非常によく一致する。

■意義　この理論は、機械学習のアルゴリズムと神経系の機能が数学的に結びつく初期の例のひとつであり、ニューラルネットワーク研究のインスピレーションにもなった。

—

まとめ

- パーセプトロンは、誤り訂正に基づく単純な学習ルールを持ち、線形分離可能なデータに対しては必ず収束する。
- このアルゴリズムは、神経科学においても小脳の学習メカニズムと類似しているとされ、生物学的にも意義深い。
- この単純な構造は多層パーセプトロンや深層学習への足がかりとなっている。

6.1.2　多層パーセプトロンと誤差逆伝播法

1. フィードフォワードニューラルネットワーク（多層パーセプトロン, MLP）

多層パーセプトロン（MLP）は、複数の隠れ層をもつ**フィードフォワード型のニューラルネットワーク**である。各層の出力が次の層の入力となり、ループ構造は持たない。

L 層のネットワークでは、入力 $\boldsymbol{x} \in \mathbb{R}^{d_0}$ に対して、各層 $l = 1, \ldots, L$ で以下の演算を行う：

$$\boldsymbol{a}^{(l)} = \boldsymbol{W}^{(l)} \boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)}, \qquad \boldsymbol{h}^{(l)} = \phi^{(l)}(\boldsymbol{a}^{(l)})$$

ここで：

- $\boldsymbol{h}^{(0)} = \boldsymbol{x}$（入力）
- $\boldsymbol{W}^{(l)}$：l 層の重み行列（形状：$d_l \times d_{l-1}$）
- $\boldsymbol{b}^{(l)}$：バイアスベクトル（形状：d_l）
- $\phi^{(l)}$：l 層の活性化関数（例：ReLU, sigmoid, tanh）

出力層の活性化関数は、回帰では恒等関数、分類では softmax やシグモイドなどが用いられる。

2. 誤差関数と学習目標

教師あり学習では、出力 $\boldsymbol{h}^{(L)}$ と正解ラベル $\boldsymbol{y}$ との誤差を測る損失関数 $\mathcal{L}(\boldsymbol{h}^{(L)}, \boldsymbol{y})$ を定義し、パラメータ $\{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}_{l=1}^{L}$ を最適化する：

$$\min \ \mathcal{L}(\boldsymbol{h}^{(L)}, \boldsymbol{y})$$

3. 誤差逆伝播法（Backpropagation）

最急降下法に基づき、損失関数の勾配 $\nabla_{\boldsymbol{W}^{(l)}}, \nabla_{\boldsymbol{b}^{(l)}}$ を効率よく計算するためのアルゴリズムが **誤差逆伝播法（backpropagation）** である。

以下、出力層から順に誤差信号を計算していく。

■(1) 出力層の誤差信号　出力層 L に対して、誤差項（デルタ）は：

$$\boldsymbol{\delta}^{(L)} := \nabla_{\boldsymbol{a}^{(L)}} \mathcal{L} = \nabla_{\boldsymbol{h}^{(L)}} \mathcal{L} \circ \phi^{(L)\prime}(\boldsymbol{a}^{(L)})$$

ここで $\circ$ は要素ごとの積（Hadamard 積）を表す。

■(2) 隠れ層の誤差信号（逆伝播）　各隠れ層 $l = L-1, \ldots, 1$ に対して：

$$\boldsymbol{\delta}^{(l)} := \left((\boldsymbol{W}^{(l+1)})^\top \boldsymbol{\delta}^{(l+1)} \right) \circ \phi^{(l)\prime}(\boldsymbol{a}^{(l)})$$

■(3) パラメータの勾配　それぞれの層において：

$$\nabla_{\boldsymbol{W}^{(l)}}\mathcal{L} = \boldsymbol{\delta}^{(l)}(\boldsymbol{h}^{(l-1)})^\top, \qquad \nabla_{\boldsymbol{b}^{(l)}}\mathcal{L} = \boldsymbol{\delta}^{(l)}$$

このように、出力層から入力層に向かって誤差項を逐次計算することで、効率よくすべてのパラメータの勾配を得ることができる。

4. まとめ

- 多層パーセプトロンは、層を重ねて複雑な非線形写像を学習可能なモデルである。
- 誤差逆伝播法により、全パラメータの勾配を効率的に計算できる。
- このアルゴリズムは、勾配降下法やその変種（SGD, Adam など）と組み合わせて使用される。

誤差逆伝播法と自動微分：深層学習フレームワークでの実装

■誤差逆伝播法と自動微分の関係　誤差逆伝播法（Backpropagation）は、**多層関数の合成における連鎖律（chain rule）**に基づいて、損失関数の勾配を効率的に計算する手法である。

このアルゴリズムは、以下の構造を持つ関数に対して適用される：

$$y = f_L \circ f_{L-1} \circ \cdots \circ f_1(\boldsymbol{x})$$

このとき、連鎖律により導関数は次のように書ける：

$$\frac{dy}{d\boldsymbol{x}} = \frac{df_L}{df_{L-1}} \cdot \frac{df_{L-1}}{df_{L-2}} \cdots \frac{df_1}{d\boldsymbol{x}}$$

この逐次的な微分伝播の過程が、誤差逆伝播法における「誤差信号の逆方向の伝播」に対応する。

この考え方は、より一般的には**自動微分（automatic differentiation, autodiff）**と呼ばれる枠組みに含まれる。

■自動微分の 2 つのモード　自動微分には主に 2 つの方式が存在する：

- **順方向モード（forward mode）**：入力の方向に沿って微分を伝播

- **逆方向モード（reverse mode）**：出力から逆方向に微分を伝播（= Backpropagation）

多くのパラメータを持つスカラー出力関数（損失関数）に対しては、逆方向モードが非常に効率的であり、深層学習における勾配計算はこれに該当する。

深層学習フレームワークにおける誤差逆伝播の実装

現代の深層学習ライブラリ（例：PyTorch, TensorFlow）は、**誤差逆伝播法を自動微分の一部として抽象化し、効率的に実装している**。

■計算グラフの構築 ニューラルネットワークの計算は、演算の合成として**計算グラフ（computational graph）**として表現される。

- 各ノード：演算（加算、行列積、活性化関数など）
- 各エッジ：中間変数

このグラフを構築することで、誤差逆伝播は連鎖律に従って自動的に適用される。

■動的 vs 静的計算グラフ

- **PyTorch, Chainer（動的計算グラフ）**：演算が行われるたびにグラフが構築される（define-by-run）。柔軟でデバッグが容易。
- **TensorFlow（旧来の静的計算グラフ）**：グラフをあらかじめ定義し、それをセッションで実行（define-then-run）。最近は Eager execution により動的にも対応。

■自動微分エンジンの動作 1. フォワードパス：計算グラフに従って損失までを計算し、中間結果を記録 2. バックワードパス：グラフを逆にたどりながら、各演算に対応する導関数を適用し、チェインルールにより勾配を伝播 3. パラメータの更新：得られた勾配を用いて勾配降下法等でパラメータを更新

これにより、研究者や開発者は、**明示的に誤差逆伝播を実装することなく、モデル設計と損失定義だけで学習が可能**となっている。

まとめ

- 誤差逆伝播法は、逆モード自動微分の特別なケースと解釈できる。
- 現在の深層学習フレームワークでは、計算グラフに基づく自動微分により誤差逆伝播が自動化されている。
- 動的計算グラフ（PyTorch）と静的計算グラフ（TensorFlow）にはそれぞれ利点がある。

6.1.3　活性化関数と勾配消失・爆発問題

1. 活性化関数の役割と選択

活性化関数は、ニューラルネットワークの非線形性を導入するための関数であり、表現力を大幅に拡張する鍵となる。深層ニューラルネットワークでは、活性化関数の選択が学習の安定性と収束に大きく影響する。

2. 勾配消失・爆発問題とは

深いネットワークにおいて、誤差逆伝播法では、層をまたいで勾配が連鎖律により次々と乗算されて伝播する。その際、活性化関数の導関数が小さいと勾配が急速に減衰し、逆に大きすぎると爆発することがある：

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(l)}} \propto \prod_{k=l+1}^{L} \phi'(\boldsymbol{a}^{(k)})$$

- $\phi'(z) \approx 0$：**勾配消失**
- $\phi'(z) \gg 1$：**勾配爆発**

3. 代表的な活性化関数とその性質

■(1) シグモイド関数

$$\phi(z) = \frac{1}{1+e^{-z}}, \qquad \phi'(z) = \phi(z)(1-\phi(z))$$

- 出力範囲が $[0, 1]$

- $\phi'(z) \leq 0.25$ と小さい
- $|z|$ が大きくなると $\phi'(z) \to 0$ となり、**勾配消失**が起きやすい

■(2) 双曲線正接関数（tanh）

$$\phi(z) = \tanh(z), \qquad \phi'(z) = 1 - \tanh^2(z)$$

- 出力範囲が $[-1, 1]$ に対称
- 中心付近で活性が強く、シグモイドよりは若干良い
- それでも深層では依然として**勾配消失**の傾向がある

■(3) Rectified Linear Unit（ReLU）

$$\phi(z) = \max(0, z), \qquad \phi'(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$$

- 出力が正の領域で線形、導関数が 1 で一定
- 伝播される勾配が一定で、**勾配消失を回避しやすい**
- $z \leq 0$ の領域では勾配がゼロになるため、**ニューロンが死ぬ（dead neuron）**問題がある

■(4) Leaky ReLU, Parametric ReLU

$$\phi(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases}$$

- $z \leq 0$ でも微小な勾配を許容することで、dead neuron を回避
- $\alpha > 0$ はハイパーパラメータ（PReLU では学習可能）

■(5) Swish 関数（ReLU と Sigmoid のハイブリッド）

$$\phi(z) = z \cdot \sigma(z)$$

- 滑らかで微分可能な関数
- 近年では ImageNet などでも良好な結果を示す

4. 勾配消失・爆発への対策

- 活性化関数の選択（ReLU 系）
- 重みの初期化法の工夫（Xavier 初期化、He 初期化）
- バッチ正規化（Batch Normalization）などによる内部共変量シフトの抑制
- 残差接続（ResNet）などによる直接伝播の導入

まとめ

- 活性化関数の導関数の大きさが勾配のスケーリングに大きく影響する。
- ReLU とその変種は、勾配消失問題への有効な対策として深層学習で広く採用されている。
- モデルの深さが増すにつれ、活性化関数だけでなく、重み初期化や正規化手法との併用が重要になる。

6.1.4　深層学習における標準的な構成要素（テクニック）

深層ニューラルネットワークの性能と安定性を向上させるために、多くの補助的な手法が提案されてきた。ここでは、現代の深層学習で標準的に用いられている代表的な構成要素について、その定義と目的を簡単にまとめる。

1. Dropout

定義： 学習時に、各層の出力ユニットを一定の確率でランダムに無効化（ゼロ化）する手法。

$$\tilde{h}_i = \begin{cases} 0 & \text{with probability } p \\ h_i/(1-p) & \text{otherwise} \end{cases}$$

目的： ニューロン間の共適応（co-adaptation）を防ぎ、過学習を抑制する。

効果： 暗黙的なアンサンブル効果。テスト時には全ユニットを使用するが、学習時のスケーリングで出力の分布を一致させる。

2. Batch Normalization（バッチ正規化）

定義： 各ミニバッチ内で、層ごとの出力を平均 0、分散 1 に正規化する操作。

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad y_i = \gamma \hat{x}_i + \beta$$

ここで μ_B, σ_B^2 はミニバッチ内の平均と分散、γ, β は学習可能なスケーリング係数とシフト。

目的： 層をまたぐ入力分布の変化（内部共変量シフト）を緩和し、学習を安定化させる。

効果： より高い学習率の使用が可能になり、学習の収束が速くなる。

3. Layer Normalization（レイヤ正規化）

定義： ミニバッチではなく、**各サンプル内のニューロン出力全体**に対して平均・分散で正規化を行う。

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad y_i = \gamma \hat{x}_i + \beta$$

ここで μ, σ^2 は同一サンプル内での統計量。

目的： 再帰型やトランスフォーマー型など、ミニバッチの統計が使いづらい構造での正規化。

効果： バッチサイズに依存せずに安定した学習が可能。

4. Residual Connection（残差接続）

定義： ネットワークの出力を次の層に渡す際、**恒等写像**を加える接続。

$$\boldsymbol{h}^{(l+1)} = \boldsymbol{h}^{(l)} + \mathcal{F}(\boldsymbol{h}^{(l)})$$

目的： 勾配消失の抑制、深層ネットワークの学習容易化。

効果： ResNet などの超深層モデルでも安定した学習が可能になる。

5. Weight Decay（L2 正則化）

定義： 損失関数に重みノルムの二乗を加える正則化。

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \lambda \|\boldsymbol{w}\|^2$$

目的： 過学習の防止。大きすぎる重みを抑制することでモデルの複雑性を制御。

6. Early Stopping（早期終了）

定義： 検証誤差が一定回数以上改善しなかった場合に、学習を打ち切る。

目的： 過学習を防ぎ、最適なモデルを保存する。

補足： 検証誤差の最小値を更新した時点のモデルを保存する運用が一般的。

まとめ

- Dropout：過学習防止のためのランダムな無効化
- Batch / Layer Normalization：学習の安定化と高速化
- Residual Connection：深層モデルにおける勾配の伝播性向上
- Weight Decay / Early Stopping：過学習を防止する正則化的手法

6.1.5　ボルツマンマシンと深層学習における役割

ボルツマンマシン（Boltzmann Machine）は、**相互結合型ニューラルネットワーク**の一種であり、確率的なエネルギーモデルとして定義される。活性状態が確率変数として定義され、ネットワーク全体で定義された**エネルギー関数**に基づくボルツマン分布を仮定する。

1. 基本構造とエネルギー関数

ニューロン（ユニット）を二値変数 $x_i \in \{0, 1\}$ で表し、全体を $\boldsymbol{x} = (x_1, \ldots, x_n)$ とする。重み付き相互作用とバイアスに基づき、エネルギー関数を次のように定義する：

$$E(\boldsymbol{x}) = -\sum_{i<j} w_{ij} x_i x_j - \sum_i b_i x_i$$

このとき、状態 $\boldsymbol{x}$ に対する確率分布は、**ボルツマン分布（ギブス分布）**として与え

られる：

$$P(\boldsymbol{x}) = \frac{1}{Z}\exp(-E(\boldsymbol{x}))$$

ここで Z は正規化定数（分配関数）：

$$Z = \sum_{\boldsymbol{x}} \exp(-E(\boldsymbol{x}))$$

2. 統計的意味づけと指数型分布族

上記の確率分布は、指数型分布族の形式に一致する：

$$P(\boldsymbol{x}) = \exp\left(\sum_{i<j} \theta_{ij} x_i x_j + \sum_i \theta_i x_i - A(\boldsymbol{\theta})\right)$$

ここで：

- $\theta_{ij} = w_{ij}$：相互作用項のパラメータ
- $\theta_i = b_i$：バイアス項
- $A(\boldsymbol{\theta}) = \log Z$：対数正規化項（対数分配関数）

この分布におけるパラメータ推定は、最尤推定や最大エントロピー原理に基づく自然なモデル選択に対応する。

3. パラメータ学習と MCMC の必要性

データセット $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ に対して、対数尤度は：

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log P(\boldsymbol{x}^{(n)}) = \sum_{n=1}^{N} \left[-E(\boldsymbol{x}^{(n)}) - \log Z\right]$$

これを $\boldsymbol{\theta}$ に関して最大化するには、勾配を計算する必要がある。エネルギー関数 $E(\boldsymbol{x})$ の微分は容易だが、Z の勾配は難しい：

$$\frac{\partial \log Z}{\partial \theta_{ij}} = \mathbb{E}_{P(\boldsymbol{x})}[x_i x_j]$$

この期待値は全ての $\boldsymbol{x}$ に渡る和であり、**指数的に大きな状態空間**を含むため、一般に解析的に求められない。

したがって、パラメータ学習には **MCMC（マルコフ連鎖モンテカルロ）法**による近似サンプリングが必要となる（ギブスサンプリングなど）。

4. 隠れ変数を含む場合と Restricted Boltzmann Machine (RBM)

実用的には、可視変数 $\boldsymbol{v}$ と隠れ変数 $\boldsymbol{h}$ を分け、学習を簡単化した構造が使われる。

■RBM（制限付きボルツマンマシン）　RBM は、以下のような**可視層と隠れ層の間にのみ結合を持つ二層構造**である：

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

この構造により、条件付き分布が簡単な形式になる：

$$P(h_j = 1 \mid \boldsymbol{v}) = \sigma\left(\sum_i w_{ij} v_i + c_j\right)$$

$$P(v_i = 1 \mid \boldsymbol{h}) = \sigma\left(\sum_j w_{ij} h_j + b_i\right)$$

可視層・隠れ層での条件付き独立性により、MCMC の更新が効率的になる（交互ギブスサンプリング）。

5. 深層学習における位置づけ：事前学習と DBN

RBM は、**深層信念ネットワーク（Deep Belief Network, DBN）**や**深層オートエンコーダー**の構築における**層ごとの事前学習（pretraining）**に用いられる。

- 各層を RBM として順に訓練し、表現を下から上へと抽出
- 最終的に得られた特徴に対して識別器（softmax など）を訓練
- 全体を微調整（fine-tuning）することで性能向上

この事前学習の流れは、初期の深層学習の成功を支えた重要なアイディアであり、現在でも表現学習や生成モデルの文脈で位置づけられている。

6.1.6　連想記憶モデルと通信理論との関連

1. Amari – Hopfield モデルの定義と基本性質

Hopfield（1982）および Amari（1977）らによって提案された相互結合型ニューラルネットワークは、**連想記憶モデル（associative memory model）**として知られ、部分的な入力から完全な記憶を再生する能力を持つ。

■構成：　ニューロンの状態 $x_i \in \{-1, +1\}$（2 値）とし、対称結合重み $w_{ij} = w_{ji}$ に基づくエネルギー関数を定義する：

$$E(\boldsymbol{x}) = -\frac{1}{2}\sum_{i,j} w_{ij} x_i x_j + \sum_i \theta_i x_i$$

ここで θ_i はしきい値（バイアス項）である。

■ダイナミクス（更新則）：　非同期でニューロンを 1 つずつ以下の規則で更新する：

$$x_i(t+1) = \mathrm{sgn}\left(\sum_j w_{ij} x_j(t) - \theta_i\right)$$

この更新はエネルギー $E(\boldsymbol{x})$ を単調減少させるため、**離散的な局所最小点**へと収束する。

2. Hebb 則による学習と記憶容量

記憶させたいパターン $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^P$, $\boldsymbol{\xi}^\mu \in \{-1, +1\}^N$ に対して、Hebb 型の学習則を用いる：

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu, \quad w_{ii} = 0$$

このとき、各パターン $\boldsymbol{\xi}^\mu$ はエネルギー極小点となる傾向がある（ただし厳密には保証されない）。

■記憶容量：　パターン数 P を増やすと、誤った記憶（スピンガラス状態）が出現し始める。

- 統計力学的手法（平均場近似・レプリカ法）により、ノイズなしで安定に記憶できるパターン数は：

$$P_c \approx 0.138N$$

- これを **記憶容量**と呼び、ニューロン数 N に比例することが示された。
- 容量以上の記憶を学習させると、エネルギー極小点が乱雑化し、想起能力が著しく低下する。

3. CDMA 通信との数学的類似性

Hopfield 型ネットワークと **CDMA（Code Division Multiple Access）通信**との間には、数理的に本質的な共通点が存在する。

■CDMA モデル：　P 人のユーザーが N ビットの符号 $\boldsymbol{s}^\mu \in \{-1, +1\}^N$ を持ち、それを共有チャネルで送信する。

$$\boldsymbol{r} = \sum_{\mu=1}^{P} a_\mu \boldsymbol{s}^\mu + \boldsymbol{n}$$

ここで $\boldsymbol{r}$ は受信信号、a_μ は送信ビット、$\boldsymbol{n}$ はノイズ。

■推定問題との関係：

- ユーザー間の符号の重なりが $\boldsymbol{s}^\mu \cdot \boldsymbol{s}^\nu$ によって決まり、他ユーザーが干渉となる（多重アクセス干渉）。
- これは Hopfield モデルの w_{ij} に相当し、各ユーザーを記憶パターンに見立てることで、**推定問題は連想記憶モデルの想起と等価**になる。

■統計力学的解析の適用：

- これらの推定問題も、平均場理論やレプリカ法を通じて記憶容量と同様の解析が可能。
- たとえば、ユーザー数 P が多くなると、ビット推定精度が急激に悪化する相転移現象が起きる。

まとめ

- Hopfield 型連想記憶モデルは、対称結合とエネルギー最小化に基づく記憶・想起モデルである。
- Hebb 則により、記憶容量は $0.138N$ 程度とされる。
- CDMA 通信における符号間干渉の推定問題と、記憶の想起問題は数学的に類似しており、統計物理的解析が両者に共通して応用可能である。

6.1.7　Elman ネットワークから LSTM への発展と現在の位置づけ

系列データの処理を目的としたニューラルネットワークは、**相互結合型（再帰型）ネットワーク**として設計され、時間的依存関係を内部状態に記憶する機構を持つ。

1. Elman ネットワーク（1990）

Elman（1990）は、以下の構造を持つ単純な再帰型ネットワークを提案した：

$$\begin{aligned}\boldsymbol{h}_t &= \sigma(W_{xh}\boldsymbol{x}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)\\ \boldsymbol{y}_t &= \phi(W_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y)\end{aligned}$$

ここで：

- $\boldsymbol{x}_t$ は時刻 t の入力、$\boldsymbol{h}_t$ は隠れ状態、$\boldsymbol{y}_t$ は出力
- W_{xh}, W_{hh}, W_{hy} は学習可能な重み行列
- σ, ϕ は非線形関数（例：tanh, softmax など）

このネットワークは、過去の状態 $\boldsymbol{h}_{t-1}$ を内部に保持することで時系列情報を記憶できる。

■課題：　長期依存関係（long-term dependency）を学習する際、**勾配消失・勾配爆発**が発生し、学習が困難になる。

2. Long Short-Term Memory（LSTM, 1997）

Hochreiter and Schmidhuber（1997）によって提案された LSTM は、再帰構造にゲート機構を導入し、長期依存関係の学習を可能にした。

■LSTM セルの構造：

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \quad \text{（忘却ゲート）} \\
\boldsymbol{i}_t &= \sigma(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad \text{（入力ゲート）} \\
\boldsymbol{o}_t &= \sigma(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad \text{（出力ゲート）} \\
\tilde{\boldsymbol{c}}_t &= \tanh(W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \quad \text{（候補セル状態）} \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \quad \text{（セル状態）} \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \quad \text{（出力状態）}
\end{aligned}
$$

- セル状態 $\boldsymbol{c}_t$ は時間的に加算的に伝搬されるため、**勾配が消えにくく**長期記憶が可能となる。
- ゲートにより、重要な情報の選択的記憶・忘却が可能。

3. GRU（Gated Recurrent Unit, 2014）

Cho et al.（2014）は、LSTM の簡略版である GRU を提案。更新ゲートとリセットゲートの 2 つに統一された構造を持ち、より軽量で学習が安定する場合もある。

4. 現在の位置づけと発展的応用

- RNN/LSTM は、音声認識、言語モデル、時系列予測などで実用的に利用されてきた。
- しかし、Transformer（2017〜）の登場により、長距離依存を並列に学習可能な構造が主流となった。
- 現在では、LSTM は中小規模の時系列処理やハードウェア制約下での使用、あるいは制御系などリアルタイム応答の分野で引き続き重要である。

■まとめ：

- Elman ネットは再帰的構造の基本形であり、時系列処理の原型を提供した。
- LSTM は勾配消失を克服し、長期記憶可能なモデルとして広く普及。
- 現代では Transformer が主流だが、再帰型ネットは依然として多くの分野で有効である。

6.1.8　Neural Tangent Kernel（NTK）とその役割

深層ニューラルネットワークは一般に非線形かつ多数のパラメータを持つ複雑な関数であり、その理論的挙動を解析することは困難である。こうした中で、**ネットワーク幅を無限大に近づけた極限**において、学習挙動が解析的に記述できることが近年明らかになり、**Neural Tangent Kernel（NTK）理論**と呼ばれる。

1. ニューラルネットワークの勾配表現と NTK の定義

パラメータ $\boldsymbol{\theta}$ をもつニューラルネットワーク $f(\boldsymbol{x};\boldsymbol{\theta})$ を考える。このとき、入力 $\boldsymbol{x}$ に対する出力のパラメータ勾配は：

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}) \in \mathbb{R}^{|\boldsymbol{\theta}|}$$

この勾配ベクトルの内積により定まるカーネルを、**ニューラル・タンジェント・カーネル**（NTK）と呼ぶ：

$$K_{\mathrm{NTK}}(\boldsymbol{x},\boldsymbol{x}') := \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})^{\top} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}';\boldsymbol{\theta})$$

このカーネルは、ネットワークのパラメータ空間における「動きやすさ」を表すものであり、**パラメータ最適化による出力変化の幾何学的構造**を記述する。

2. 無限幅極限と線形化近似

ネットワークの各層の幅を無限大にすると、以下のような事実が知られている（Jacot et al., 2018）：

- ランダム初期化時点の NTK は確率的に収束する（確定的な関数になる）。
- 勾配降下法による学習は、初期点での線形モデルと同様に振る舞う。

- 学習中の NTK は一定とみなして良く、以下の線形 ODE で出力が変化する：

$$f_t(\boldsymbol{x}) = f_0(\boldsymbol{x}) - K_{\mathrm{NTK}}(\boldsymbol{x}, \cdot) \cdot \left(I - e^{-\eta K_{\mathrm{NTK}} t}\right) \boldsymbol{y}$$

これは、**学習がカーネル回帰に対応する**という深い意味を持つ。

3. カーネル法・ガウス過程との関係

無限幅ニューラルネットワークは、以下の 2 つの側面をもつ：

- 出力の事前分布（random initialization）⇒ **ガウス過程**（Neural Network GP）
- 出力の変化（学習）⇒ **カーネル回帰**（NTK）

つまり、**ベイズ的立場ではガウス過程として解釈され、最適化的立場ではカーネル回帰として解釈される**という二重の性質を持つ。

4. NTK 理論の役割と意義

- 深層ニューラルネットワークの学習挙動を解析的に扱える数少ない理論的手法。
- 勾配降下法による学習が、ある意味で「線形モデル」として解釈できる。
- 過剰パラメータ化（overparameterization）下でも安定に学習が進む理由の一端を説明。
- 正則化項を持たないにもかかわらず、**自然な一般化性能**が得られる理由に関する理論的議論の足掛かりとなっている。

まとめ

- NTK はニューラルネットワークの勾配空間に基づくカーネルであり、無限幅極限で一定値に収束する。
- この極限では学習がカーネル回帰と等価となり、ニューラルネットワークの出力は解析的に記述できる。
- NTK 理論は、深層学習の理論的理解に重要な道具を提供している。

6.1.9 二重降下現象とその数理的理解

1. 二重降下とは何か？

伝統的な統計学では、モデルの複雑度（パラメータ数）が増えると、訓練誤差は減少するが汎化誤差（テスト誤差）は**U 字型**のように振る舞うとされる（バイアス・バリアンスのトレードオフ）。しかし深層学習では、以下のような非直感的な挙動が観測される：

- モデル容量が訓練データにちょうど一致する境界（**補間境界, interpolation threshold**）では、テスト誤差がピークを持つ。
- しかしそれを超えて**さらにパラメータ数を増やすと、テスト誤差が再び低下する**という現象が起こる。

このようなテスト誤差の曲線は、従来の U 字型とは異なり、「**二重に落ちる（double descent）**」という特徴的な形状を持つ。

2. 二重降下の数理的モデル：線形回帰における例

単純な線形回帰を考える：

$$\boldsymbol{y} = X\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I)$$

$X \in \mathbb{R}^{n \times p}$ が説明変数行列で、p（特徴量数）が n（データ数）に等しい、またはそれを超える場合、最小二乗法は以下のように変化する：

- $p < n$：正則な解が存在し、古典的な推定理論が成り立つ。- $p = n$：最小二乗解がちょうど訓練データを補間するため、ノイズ $\boldsymbol{\varepsilon}$ も完全に学習してしまい、**テスト誤差が爆発**する。- $p > n$：最小ノルム解（例：$\hat{\boldsymbol{\beta}} = X^\dagger \boldsymbol{y}$）が唯一存在し、過剰パラメータにもかかわらず、テスト誤差が再び減少する。

■最小ノルム解の一般化誤差： Belkin et al. (2019) によれば、過剰パラメータ化領域での最小ノルム解は、次の理由で一般化能力を持つ：

- 解が訓練誤差をゼロにする制約の下でノルム最小（正則化的効果）。

- 特徴空間における最も平滑な解が選ばれる（ベイズ的 prior に近い構造）。

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta} \text{ s.t. } X\boldsymbol{\beta}=\boldsymbol{y}} \|\boldsymbol{\beta}\|^2$$

3. 深層学習における解釈：暗黙的な正則化

ニューラルネットワークにおいても、以下のような類似の構造があると考えられている：

- ネットワークは巨大な表現能力を持つが、**学習アルゴリズム（例：SGD）自体が「平滑な解」や「最小ノルム解」を暗黙的に選ぶ**。
- NTK 理論では、無限幅ネットワークの学習がカーネル回帰と等価になり、やはり最小ノルム解を与える。
- よって、過剰パラメータ化しても悪い解に陥るとは限らず、むしろ「良い解」へと導かれる場合がある。

4. まとめ

- 二重降下現象は、補間境界を越えた領域でテスト誤差が再び減少する現象。
- 線形回帰や NTK 理論では、最小ノルム解が過剰パラメータ化領域で良好な一般化性能を持つ。
- 深層学習では、暗黙の正則化と学習ダイナミクスによって、複雑なモデルでも安定した学習と一般化が実現されている。

補足：二重降下が生じる条件

二重降下現象は、すべての学習状況で起こるわけではない。以下のような条件や要因の下で、特に顕著に観測されることが知られている。

■1. モデル容量と補間境界　二重降下の鍵は、モデルが「補間可能」な領域（interpolation regime）を通過することである。

- 補間境界とは：学習データを完全に（訓練誤差ゼロで）再現できる最小のモデ

ル容量。

- 補間境界に達する直前では、過学習によりテスト誤差が増大する（バリアンス爆発）。
- しかし、補間境界を超えると、暗黙の正則化効果によってテスト誤差が再び減少する。

■2. ノイズの存在

- 出力に対するノイズ（$y = f(x) + \varepsilon$）がある場合、モデルがノイズまで学習してしまうと、汎化性能が急激に悪化する。
- 特に、補間境界ではこのノイズも忠実にフィッティングされるため、テスト誤差が最大化しやすい。

■3. モデル構造と学習アルゴリズムのバイアス

- 学習アルゴリズム（例：SGD）には、**暗黙のバイアス**（implicit bias）が存在し、同じ補間解の中でも「滑らか」なものを好む傾向がある。
- ニューラルネットワークでは、過剰パラメータ化によってこの暗黙の正則化効果が強く働く。

■4. モデルの柔軟性（nonlinearity）と初期化

- 線形モデルでも二重降下は観測されるが、深層ニューラルネットワークではより顕著。
- 幅の大きなネットワークでは、初期化と活性化関数の選択が学習挙動に影響を与える。

■5. データサイズとデータ多様性

- 固定されたデータ数 n に対して、特徴数 p やパラメータ数が増加していくときに現れる現象。

- 多様性の低い（冗長な）データでは、補間が容易で、二重降下がより観測されやすい。

まとめ

二重降下が顕在化するためには、以下のような条件が組み合わさる必要がある：

- モデル容量が補間可能性を越えて十分に大きいこと
- データにノイズや非決定性が含まれていること
- 学習アルゴリズムが暗黙的に滑らかな解を選ぶ傾向を持つこと

これらの要素が揃ったとき、モデルは「過学習による誤差増大」を一度経験した後、再び汎化性能を回復し、二重降下という形で誤差曲線が現れる。

6.2　畳み込みニューラルネットワーク（CNN）とその発展

6.2.1　1. 生物視覚にヒントを得た CNN の起源：Neocognitron

畳み込みニューラルネットワーク（CNN）は、生物の視覚系における受容野構造にヒントを得たモデルであり、その起源は Fukushima（1980）による **Neocognitron** に遡る。

- Neocognitron は、階層的な **S 細胞（特徴抽出）**と **C 細胞（位置不変性）**からなる構造を持ち、視覚パターンを認識する。
- フィルタは教師なしで獲得され、位置のずれに不変な表現を階層的に学習する点が特徴的。
- これは後の CNN における **畳み込み層（convolutional layer）**と **プーリング層（pooling layer）**の原型である。

6.2.2　2. CNN の基本構造

現代の CNN は、以下のような構造からなる：

- **畳み込み層（Convolutional Layer）**：局所受容野でフィルタを適用し、空間的特徴を抽出
- **活性化関数（Activation）**：ReLU などの非線形変換
- **プーリング層（Pooling Layer）**：空間的な縮小（例：max pooling）
- **全結合層（Fully Connected Layer）**：高次特徴を分類器に接続

6.2.3 3. CNN のブレイクスルー：AlexNet

CNN の飛躍的な発展の契機となったのが、Krizhevsky et al. による **AlexNet**（2012）である。

- ILSVRC（ImageNet Large Scale Visual Recognition Challenge）にて従来を圧倒する精度を記録。
- 主な特徴：
  - 大規模画像データセット（ImageNet）の利用
  - GPU による並列学習の導入
  - ReLU 活性化関数の採用
  - Dropout による過学習抑制

この成功により、CNN は画像認識を中心とする多くの応用で標準技術となった。

6.2.4 4. CNN の代表的な発展モデル

(1) U-Net：医療画像やセグメンテーションの標準モデル

U-Net（Ronneberger et al., 2015）は、画像のピクセル単位での分類（セグメンテーション）に特化した構造である。

- 特徴：
  - 畳み込み＋プーリングによるエンコーダ（縮小）
  - アップサンプリングによるデコーダ（拡大）
  - エンコーダの中間層とデコーダを **skip connection** で接続し、細部情報

を伝搬

- 医用画像分野を中心に、微細な境界を含むセグメンテーションで高精度を達成。

(2) ResNet：深層化の限界を打破する残差ネットワーク

ResNet（He et al., 2015）は、ネットワークを非常に深くしても学習が安定する構造として登場。

- **Residual Block** によって、以下のように恒等写像を通す：

$$\boldsymbol{h}^{(l+1)} = \boldsymbol{h}^{(l)} + \mathcal{F}(\boldsymbol{h}^{(l)})$$

- 勾配の流れを阻害せずに深いネットワークの学習が可能に（例：100 層以上）。
- ImageNet をはじめとする多くのベンチマークで性能向上。

6.2.5　5. その他の応用と変種

CNN は画像認識以外にも多くの分野に応用され、以下のような変種が提案されている：

- **Dilated CNN**：畳み込みの受容野を拡張（音声・時系列に応用）
- **MobileNet, EfficientNet**：軽量で省メモリなモデル（モバイル向け）
- **Vision Transformer (ViT)**：CNN の代替として登場したトランスフォーマーベースの画像モデル（CNN とのハイブリッド構造も）

6.2.6　まとめ

- CNN は視覚系に着想を得た構造であり、Neocognitron が原点となる。
- AlexNet により大規模画像認識への有効性が実証され、標準技術となった。
- U-Net や ResNet は、セグメンテーションや深層化における課題を解決する構造であり、多くの分野で利用されている。

6.3 Attention の原理と CNN から Transformer への発展

6.3.1 1. Attention の基本原理

Attention は、**入力情報の中で重要な部分に焦点を当てる**ためのメカニズムであり、自然言語処理や画像処理をはじめとする多くのタスクで広く使われている。

■基本構造（スコアに基づく重み付き和） Attention メカニズムは、**クエリ $\boldsymbol{q}$**、**キー $\boldsymbol{k}_i$**、**バリュー $\boldsymbol{v}_i$** の 3 種類のベクトルに基づいて定義される。

$$\text{Attention}(\boldsymbol{q}, \{\boldsymbol{k}_i, \boldsymbol{v}_i\}) = \sum_i \alpha_i \boldsymbol{v}_i$$

ここで、重み α_i は $\boldsymbol{q}$ と $\boldsymbol{k}_i$ の類似度に基づく softmax 正規化された重み：

$$\alpha_i = \frac{\exp(\text{score}(\boldsymbol{q}, \boldsymbol{k}_i))}{\sum_j \exp(\text{score}(\boldsymbol{q}, \boldsymbol{k}_j))}$$

多くの場合、スコア関数には内積スコアを用いる：

$$\text{score}(\boldsymbol{q}, \boldsymbol{k}_i) = \frac{\boldsymbol{q}^\top \boldsymbol{k}_i}{\sqrt{d}}$$

これにより、Attention は以下のように計算される：

$$\text{Attention}(\boldsymbol{q}, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

ここで Q, K, V はそれぞれ複数のクエリ・キー・バリューをまとめた行列である。

■有効性の直感的理解 Attention の利点は以下のように要約される：

- 異なる入力位置（時系列・画像領域など）間の動的な関係性をモデル化できる。
- 固定の畳み込みカーネルとは異なり、文脈に応じて重要な部分を動的に重みづけできる。
- 計算は並列化しやすく、勾配も安定して流れる。

6.3.2　2. CNN との統合：Self-Attention in Vision

畳み込みは局所的特徴抽出に優れるが、長距離依存関係の捕捉が困難という弱点がある。これに対して Attention は以下のような補完的な役割を果たす：

- CNN による特徴抽出後、各特徴マップに Self-Attention を適用することで、**位置に依存しないグローバルな関係性**を表現できる。
- これにより、**空間的に離れた領域間の相互作用（例：物体の一貫性や文脈）**をモデル化可能。
- 代表的な応用例：Non-local Neural Network（Wang et al., 2018）

6.3.3　3. 完全な置き換え：Transformer アーキテクチャへの展開

Vaswani et al. (2017) による **Transformer** は、Attention のみで系列モデリングを行う構造であり、以下のような構成を持つ：

- 自己 Attention（Self-Attention）層によって、入力系列内の全位置間の依存関係を一括して学習。
- 畳み込みや再帰構造を一切用いず、すべて行列演算で並列化可能。
- 残差接続・Layer Normalization により深い構造の安定な学習が可能。

■画像処理への応用：Vision Transformer (ViT)

- 入力画像を小さなパッチに分割し、それぞれを系列として Transformer に入力する。
- パッチ間の関係性を Self-Attention で学習し、画像全体の理解に利用。
- 十分なデータがある場合、CNN と同等以上の性能を示す。

6.3.4 まとめ

- Attention は、入力間の関係性を動的に学習する柔軟なメカニズムであり、従来の CNN の限界を補う。
- CNN + Attention のハイブリッド構造により、ローカルとグローバルの両方の特徴を効率的に捉えられる。
- 完全な Attention ベースの構造である Transformer は、自然言語処理にとどまらず、画像処理など幅広い応用に展開されている。

6.3.5 Transformer における Self-Attention の数式的表現

Transformer では、入力系列 $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times d_{\text{model}}}$ に対して、自己注意（self-attention）機構を通じて出力系列 $\boldsymbol{Z} \in \mathbb{R}^{n \times d_{\text{model}}}$ を生成する。ここで、n は系列長、d_{model} は特徴次元である。

1. クエリ・キー・バリューの生成

まず、入力 $\boldsymbol{X}$ に対して線形変換を行い、クエリ、キー、バリューを計算する：

$$Q = \boldsymbol{X} W_Q, \quad K = \boldsymbol{X} W_K, \quad V = \boldsymbol{X} W_V$$

ここで：

- $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ はそれぞれクエリ、キー、バリューの変換行列。
- 通常は $d_k = d_v = d_{\text{model}}/h$（$h$ はヘッド数）とする。

2. Attention スコアの計算と softmax 正規化

各位置間の関連度（注意スコア）は、クエリとキーの内積に基づいて計算され、softmax 関数で正規化される：

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \quad \in \mathbb{R}^{n \times n}$$

この行列 A は、各トークンが他のトークンにどの程度注意を払うかを表す。

3. バリューの重み付き和による出力の生成

出力は、注意スコアを用いてバリューの加重平均を計算することで得られる：

$$Z = AV \quad \in \mathbb{R}^{n \times d_v}$$

これが 1 つのヘッドの出力である。

4. マルチヘッド Attention の統合

h 個のヘッドを並行して処理し、それらを結合して最終出力を得る：

$$\mathrm{MultiHead}(X) = \mathrm{Concat}(Z_1, \ldots, Z_h) W_O$$

ここで：

- 各 $Z_i = \mathrm{Attention}(XW_Q^{(i)}, XW_K^{(i)}, XW_V^{(i)})$
- $W_O \in \mathbb{R}^{hd_v \times d_{\mathrm{model}}}$ は最終出力の変換行列

5. Transformer ブロックの全体構成（1 層）

Self-Attention に加えて、以下の構成で 1 層の Transformer Encoder が構成される：

$$\boldsymbol{X}' = \mathrm{LayerNorm}(X + \mathrm{MultiHead}(X)) \quad \text{（残差接続 + 正規化）}$$
$$\boldsymbol{Z} = \mathrm{LayerNorm}(\boldsymbol{X}' + \mathrm{FFN}(\boldsymbol{X}')) \quad \text{（全結合 + 正規化）}$$

ここで FFN は位置ごとの前処理ネットワーク（通常 2 層の MLP）である：

$$\mathrm{FFN}(\boldsymbol{x}) = \mathrm{ReLU}(\boldsymbol{x}W_1 + b_1)W_2 + b_2$$

6.4 Transformer を用いた大規模言語モデルの構成法：BERT と GPT を中心に

6.4.1 1. 背景と基本構成

Transformer は、自然言語処理（NLP）における系列データの処理を目的として設計され、並列計算の容易さと長距離依存関係の学習能力により、BERT や GPT といった大規模言語モデル（LLM: Large Language Models）の中核として採用されている。

LLM は、次のような構成要素から成る：

- **アーキテクチャ**：Transformer Encoder（BERT）または Transformer Decoder（GPT）
- **学習課題**：マスク予測（BERT）または自己回帰的予測（GPT）
- **学習データ**：大規模なウェブ・書籍・Wikipedia などから収集
- **目的**：汎用的な文脈理解／生成能力の獲得

6.4.2 2. BERT：双方向的文脈理解モデル

■構成： Transformer Encoder を用いた双方向の言語理解モデル。学習課題は主に以下の 2 つ：

- **Masked Language Modeling (MLM)**：入力文中の一部トークンを `[MASK]` に置き換え、それを予測する。

$$\text{Loss}_{\text{MLM}} = -\sum_{\text{masked } i} \log P(x_i \mid \boldsymbol{x}_{\setminus i})$$

- **Next Sentence Prediction (NSP)**：2 つの文が連続しているか否かを判定するバイナリ分類。

■特徴と効果：

- 双方向の文脈情報を用いた深い意味理解が可能。
- 文分類や質問応答、文の類似性評価などに有効。

6.4.3　3. GPT：自己回帰型の生成モデル

■構成：　Transformer Decoder を用い、自己回帰的な言語生成を行う。

- **Causal Masked Language Modeling**：トークン x_i の予測は、過去のトークン $x_1, ..., x_{i-1}$ のみに依存。

$$\text{Loss}_{\text{AR}} = -\sum_{i=1}^{n} \log P(x_i \mid x_1, ..., x_{i-1})$$

■特徴と効果：

- 単一モデルでテキスト生成・要約・翻訳など幅広いタスクに対応。
- 大規模事前学習の後、少数の事例または指示（in-context learning）で下流タスクに適応。

6.4.4　4. データ・学習・スケーリング

■学習データ：　Wikipedia, BooksCorpus, WebText, Common Crawl など、数百億〜数兆トークン規模のデータが使用される。

■学習戦略とスケーリング：

- パラメータ数：BERT-base（110M）〜 GPT-3（175B）〜 GPT-4（非公開）
- スケーリング法：分散学習（データ並列・モデル並列）、混合精度学習（FP16）、optimizer（AdamW）
- 学習時間：数千 GPU を用いた数週間規模の事前学習

6.4.5 5. 本質的な工夫と応用の違い

■BERT の本質：

- 双方向文脈の理解に基づく汎用エンコーダ
- ファインチューニングによる下流適応が基本

■GPT の本質：

- 自然なテキスト生成に最適化されたデコーダ型モデル
- プロンプト設計や few-shot により適応性が高い
- RLHF（人間のフィードバックによる強化学習）などでさらに性能向上

6.4.6 まとめ

- Transformer は大規模言語モデルの基盤として、自己注意機構による柔軟な系列処理を可能にした。
- BERT は文脈理解、GPT は生成能力に優れ、それぞれ異なる設計思想と適用領域を持つ。
- モデルスケーリングと事前学習の進化により、少数例学習や対話型 AI への応用が急速に進展している。

6.4.7 自然言語におけるトークンの定義と実装上の工夫

自然言語処理（NLP）では、文章を機械で扱いやすい単位に分割することが基本であり、その単位を「トークン（token)」と呼ぶ。トークンは、言語モデルの入力・出力の基本単位である。

1. トークンの定義

トークンとは、文や文書を構成する「意味的または構文的な最小単位」として扱うもので、以下のような種類がある：

- **単語トークン（word token）**：空白や句読点で区切られる語（例：”I”, ”like”, ”apples”）
- **サブワードトークン（subword token）**：単語をさらに細分化した部分語（例：”play”, ”##ing”）
- **文字トークン（character token）**：1 文字ずつ分割（例：”H”, ”e”, ”l”, ”l”, ”o”）
- **バイトペアトークン（byte token）**：バイト列として処理（Unicode 対応など）

大規模言語モデル（LLM）では、意味の保持と語彙の圧縮のバランスのために、**サブワード分割**が主流となっている。

2. サブワード分割の手法

■Byte-Pair Encoding (BPE)

- 頻出の文字ペアを反復的に結合し、新たな語彙として登録する。
- 未知語に強く、語彙サイズを制限できる。
- 例： `"play"` + `"ing"` → `"playing"` → `"play"`, `"##ing"`

■WordPiece（BERT）や SentencePiece（GPT）

- 言語モデルの性能向上を目的に、確率的なスコアに基づいて語彙を学習。
- SentencePiece は空白を含めた文字列全体を対象とし、Unicode ベースのトークン化が可能。
- 空白も明示的なトークン（例：▁）として扱うことで、前処理を簡素化。

3. トークンとモデル入力の関係

- Transformer ベースの言語モデルは、各トークンを整数 ID に変換し、埋め込みベクトルとして処理する。
- トークン数（系列長）には上限があり、通常は $n = 512$〜4096 程度。
- トークン化の粒度によって、入力文の長さ（トークン数）は大きく変動する。

4. モデルによるトークン粒度の違い：例

- **”playing”** → BERT: `["play", "##ing"]`, GPT: `["play", "ing"]`
- **”こんにちは”** → SentencePiece: `["▁こん", "にちは"]`
- **”@”**（絵文字）→ GPT 系モデルでは単独のトークンに対応することも多い

まとめ

- トークンは、自然言語をモデルが扱うための離散的な最小単位。
- BPE や SentencePiece に基づくサブワード分割が大規模言語モデルの標準。
- トークン化は、モデルの語彙設計・圧縮効率・未知語処理に大きく関係する。

6.5　生成 AI の全体像と生成モデルの分類

6.5.1　1. 生成 AI とは何か

生成 AI（Generative AI）は、**人間のように意味のある新しいデータを生成する能力を持つ人工知能技術**であり、与えられた分布に基づいて、画像・音声・文章・動画などの新しいデータサンプルを合成することを目的とする。

■定義：　生成 AI は、**観測データの分布** $p(\boldsymbol{x})$ を近似的に学習し、その分布からのサンプリングや条件付き生成を行う手法の総称である。

6.5.2　2. 主な応用分野

生成 AI は、以下のような多様な応用分野に利用されている：

- **画像生成**：人物画像、風景、イラストなどの自動生成（例：DALL-E、Midjourney）
- **自然言語生成**：文章の自動生成、要約、翻訳、対話（例：GPT）
- **音声合成**：音声読み上げ、音声模倣（例：WaveNet, VALL-E）
- **音楽・動画生成**：作曲やアニメーションなどの時系列生成
- **データ補完・ノイズ除去**：画像修復、欠損データの補完

6.5.3　3. 生成モデルの主な分類

生成モデルには、主に以下のようなカテゴリが存在する：

- **敵対的生成ネットワーク（GAN）**：判別器と生成器を対立させることで分布を近似
- **拡散モデル（Diffusion Model）**：ノイズの付加と除去の過程でサンプリング
- **正規化フロー（Normalizing Flow）**：可逆変換により密度関数と生成を両立
- **変分オートエンコーダ（VAE）**：潜在変数モデルと変分推論の組み合わせ
- **自己回帰モデル（Autoregressive Models）**：系列的な条件付き生成（例：PixelCNN, GPT）

6.5.4　4. 尤度の扱いによる分類

生成モデルは、学習における**尤度の扱い**に応じて以下のように分類できる：

- **明示的尤度モデル**：データ分布 $p(\boldsymbol{x})$ を明示的に定義し、尤度最大化によって学習（VAE, Flow）
- **暗黙的分布モデル**：$p(\boldsymbol{x})$ の形は定義せず、サンプル生成の仕組みのみを定義（GAN, 拡散モデル）

6.5.5　5. 潜在変数モデルの視点

多くの生成モデルは、**潜在変数 $\boldsymbol{z}$** を導入し、次のような構造をとる：

$$\boldsymbol{z} \sim p(\boldsymbol{z}), \quad \boldsymbol{x} \sim p(\boldsymbol{x} \mid \boldsymbol{z})$$

この形式は、人間が想像や構築を行う際の「抽象的な表現」→「具体的なデータ」の変換と類似しており、以下のモデルに共通する：

- VAE：潜在変数に対して事後分布を近似
- GAN：潜在変数からデータを生成、判別器で評価
- Flow：潜在空間とデータ空間を双方向に可逆変換

6.5.6　まとめ

- 生成 AI は、データ分布を学習して新たなデータを生成する能力を持つ AI 技術である。
- 生成モデルは、尤度の有無・潜在変数の構造・学習戦略により多様な分類がある。
- 次節以降では、代表的なモデルである VAE, GAN、拡散モデル、フローモデルの構造と特性を詳しく解説する。

6.5.7　変分オートエンコーダ（VAE）の構造と学習法

VAE（Variational Autoencoder）は、生成モデルとオートエンコーダの構造を統合した潜在変数モデルであり、**ニューラルネットワークによって変分ベイズ推論を近似的に実行する**手法である（Kingma and Welling, 2013）。

1. 潜在変数モデルとしての定式化

観測データ $\boldsymbol{x}$ に対して、潜在変数 $\boldsymbol{z}$ を導入した生成モデルを考える：

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{z}) p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$$

ここで：

- $p_\theta(\boldsymbol{z})$ は潜在変数の事前分布（通常は標準正規分布 $\mathcal{N}(\mathbf{0}, I)$）
- $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ は生成分布（Decoder）がニューラルネットで表現される

真の事後分布 $p_\theta(\boldsymbol{z} \mid \boldsymbol{x})$ は非可解なことが多いため、近似事後分布 $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$（Encoder）を導入し、変分推論を行う。

2. 変分下限（ELBO）の最大化

対数尤度 $\log p_\theta(\boldsymbol{x})$ に対して、変分下限（Evidence Lower Bound, ELBO）を最大化する：

$$\log p_\theta(\boldsymbol{x}) \geq \mathcal{L}(\theta, \phi; \boldsymbol{x}) := \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})] - D_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p_\theta(\boldsymbol{z}))$$

ここで：

- 第 1 項は再構成誤差（Decoder が $\boldsymbol{x}$ を復元する性能）
- 第 2 項は事後分布の事前分布への KL 距離（潜在空間の正則化）

3. 学習の実装：再パラメータ化トリック

変分分布を：

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x})))$$

のように仮定すると、$\boldsymbol{z}$ は以下のようにサンプリングできる：

$$\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$$

この「**再パラメータ化トリック**」によって、サンプリングを通じた勾配計算が可能となる。

4. アーキテクチャ構成

VAE は以下の 2 つのニューラルネットワークから構成される：

- **Encoder（認識モデル）**：$\boldsymbol{x} \mapsto (\boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\sigma}^2_\phi(\boldsymbol{x}))$ を出力し、潜在変数の分布を近似
- **Decoder（生成モデル）**：$\boldsymbol{z} \mapsto \hat{\boldsymbol{x}}$ を出力し、潜在空間から観測空間への写像を学習

5. 特徴と応用

- 明示的な尤度に基づく安定な学習
- 潜在空間に滑らかで意味のある表現が得られる
- 解像度や生成品質はやや低めだが、構造制御や補間に優れる
- 応用：異常検知、潜在空間探索、画像生成、ベイズ的表現学習など

まとめ

- VAE は変分ベイズとニューラルネットの統合によって潜在変数モデルを効率的に学習する。
- 潜在空間の分布を正則化しつつ、生成分布を再構成誤差として最適化する。
- GAN や拡散モデルのような高精度生成には劣るが、理論的安定性と可視化可能性に優れる。

6.5.8 敵対的生成ネットワーク（GAN）とその特徴

GAN（Generative Adversarial Network）は、Goodfellow ら（2014）により提案された生成モデルであり、**生成器と識別器の敵対的学習によってデータ分布を近似する**。

1. GAN の基本構造

GAN は以下の 2 つのネットワークから構成される：

- **生成器** $G_\theta(\boldsymbol{z})$：潜在変数 $\boldsymbol{z} \sim p(\boldsymbol{z})$（通常は標準正規分布）から観測データ $\boldsymbol{x}$ に似たサンプル $\hat{\boldsymbol{x}}$ を生成する。
- **識別器** $D_\phi(\boldsymbol{x})$：入力 $\boldsymbol{x}$ が本物データか生成データかを確率的に判別する。

2. 学習目標（ミニマックス最適化）

生成器と識別器は次のような敵対的な損失関数に基づいて学習される：

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- 識別器 D は本物と偽物を区別しようとし、
- 生成器 G は識別器を騙すようにデータを生成する。

理想的には $p_G(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$ となり、生成分布が真のデータ分布と一致する。

3. GAN の利点と課題

■利点：

- 尤度を明示的に定義せず、非常に高精度なサンプルが得られる（特に画像生成）

■課題：

- **学習の不安定性**（振動・モード崩壊）
- 勾配消失・敵対的訓練の困難さ

6.5.9　VAE との比較

- **VAE**：尤度を定義し、変分推論により安定した学習。生成画像はややぼやけやすい。

- **GAN**：尤度を使わず、敵対的損失によりリアルな生成を実現。学習は不安定。

特徴	VAE	GAN
生成方式	潜在変数から復元	潜在変数から直接生成
尤度モデル	明示的に定義	明示せず
学習安定性	高い	不安定（調整が必要）
生成品質	滑らかだがやや粗い	高品質な画像
潜在空間の意味解釈	容易（連続性あり）	難しい（非連続的）

表 6.1　VAE と GAN の比較

6.5.10　VAE-GAN：統合モデル

VAE と GAN を統合し、両者の利点を活かすモデルとして **VAE-GAN**（Larsen et al., 2015）が提案されている。

- **Encoder**：$\boldsymbol{x} \mapsto q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$（VAE）
- **Decoder / Generator**：$\boldsymbol{z} \mapsto \hat{\boldsymbol{x}}$（VAE + GAN）
- **Discriminator**：$\boldsymbol{x} \mapsto$ real/fake（GAN）

■学習損失：　再構成誤差に加えて、識別器の特徴空間での再構成誤差と敵対的損失を組み合わせる：

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda_1 D_{\text{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \| p(\boldsymbol{z})) + \lambda_2 \mathcal{L}_{\text{GAN}}$$

これにより、**潜在空間の構造性**と**生成画像の高品質化**を両立させることができる。

まとめ

- GAN は高精度なサンプル生成に優れるが、学習が難しい。
- VAE は安定性と潜在空間の解釈性に優れるが、生成品質はやや劣る。
- VAE-GAN は両者の強みを統合し、構造性と品質を両立させた生成モデルで

ある。

6.5.11　拡散モデル：基本構造と生成制御

1. 基本的な考え方

拡散モデル（Diffusion Model）は、**ランダムノイズから段階的にデータを生成する確率的モデル**である。大まかに以下の 2 つの過程を用いる：

- **順方向過程（forward process）**：データ $\boldsymbol{x}_0$ に徐々にガウスノイズを加えていき、完全なノイズ $\boldsymbol{x}_T$ に変換する。

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t \mid \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t I)$$

- **逆方向過程（reverse process）**：ノイズから元のデータを再構成する生成過程。これを学習モデル p_θ で近似：

$$p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t))$$

これにより、初期ノイズから段階的に画像などの構造を回復する「ノイズ除去の逐次過程」が実現される。

2. 学習アルゴリズム（DDPM：Denoising Diffusion Probabilistic Model）

Ho et al. (2020) による DDPM では、以下のように生成器を訓練する：

■目標：　ノイズ付加の逆過程を模倣するようなネットワーク $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ を学習

■損失関数（MSE 形式）　：

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\epsilon}, t}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

ここで：

- $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$：真のノイズ
- $\bar{\alpha}_t$：累積ノイズスケール（例：$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$）
- ネットワークは $\boldsymbol{x}_t$ から元のノイズ $\boldsymbol{\epsilon}$ を予測

3. 拡散過程と物理学との関係

拡散モデルの順方向過程は、物理学における拡散方程式（熱方程式）と等価な確率過程に対応する：

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = \nabla_{\boldsymbol{x}} \cdot (D \nabla_{\boldsymbol{x}} p(\boldsymbol{x}, t))$$

また、スコアベース生成（Score-Based Generative Model）では、**スコア関数** $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ を推定し、SDE（確率微分方程式）や ODE（常微分方程式）により逆過程を記述する。

4. 条件付き生成と context の導入

拡散モデルは、言語やクラスラベルなどの「文脈（context）」を入力として追加することで、条件付き生成が可能である。

■代表的な方法：

- **クラス条件付き生成**：クラスラベル y を入力に追加：

$$\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, y)$$

- **言語条件付き（Text-to-Image）**：テキストから得られる文脈ベクトル $\boldsymbol{c}$ を追加（CLIP などの埋め込み）：

$$\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c})$$

- **Classifier-Free Guidance**（CFG）：条件付きと非条件付きの予測を補間：

$$\tilde{\boldsymbol{\epsilon}} = (1 + w)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}) - w\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$$

ここで w は文脈の強度を制御するパラメータ。

■応用例：

- DALL-E 2、Stable Diffusion、Imagen などはテキスト条件付きの拡散モデル
- 構造制御・画像変換・音声合成などにも応用可能

まとめ

- 拡散モデルは、ノイズから段階的にデータを復元する生成手法であり、高品質なサンプルが得られる。
- 学習には逐次的なノイズ除去の過程を模倣するニューラルネットワークを用いる。
- 拡散方程式・スコアベース生成など物理学と密接に関係する。
- 条件付き拡散により、言語やラベルを用いた制御された生成が実現されている。

6.5.12　フローモデルと拡散モデルの関係性

1. フローモデルの基本的枠組み

フローモデル（Normalizing Flow）は、**単純な確率分布からの可逆変換により複雑なデータ分布を構築**する生成モデルである。

■**基本構造：**　可逆な写像 $\boldsymbol{x} = f_\theta(\boldsymbol{z})$ を学習し、逆写像 $\boldsymbol{z} = f_\theta^{-1}(\boldsymbol{x})$ も計算可能な構造を持つ。

$$\boldsymbol{z} \sim p(\boldsymbol{z}) \quad \Rightarrow \quad \boldsymbol{x} = f_\theta(\boldsymbol{z}) \sim p_\theta(\boldsymbol{x})$$

■密度の変換：

$$p_\theta(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right| = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f_\theta(\boldsymbol{z})}{\partial \boldsymbol{z}} \right)^{-1} \right|$$

ヤコビアン行列の行列式を通じて、複雑な密度関数も正確に計算可能となる。

2. 学習：尤度最大化

観測データ $\boldsymbol{x}$ に対して、対数尤度 $\log p_\theta(\boldsymbol{x})$ を最大化することでパラメータを学習：

$$\log p_\theta(\boldsymbol{x}) = \log p(f_\theta^{-1}(\boldsymbol{x})) + \log \left| \det \left(\frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right|$$

このため、**密度推定と生成が同時に可能**な利点を持つ。

3. 拡散モデルとの比較

特徴	フローモデル	拡散モデル
生成過程	可逆写像（直接）	ノイズ除去の逐次過程
尤度計算	明示的（正確）	暗黙的（または近似）
学習速度	高速（並列可）	遅い（逐次処理）
生成品質	中程度	非常に高い（画像など）
連続性・潜在空間操作	優れる	優れる
逆写像の可用性	常に可能	モデルによる

表 6.2　フローモデルと拡散モデルの比較

■共通点と関係性：

- どちらも「単純な分布」→「複雑な分布」への変換を行う点で共通
- 拡散モデルも可逆過程（SDE, ODE）として解釈可能であり、**連続的なフロー**としての視点で理解できる
- Song et al. によるスコアベース生成では、拡散過程を時間連続な流れ（continuous normalizing flow）とみなす

4. フローモデルの代表例と応用

- **Real NVP（Dinh et al., 2016）**：可逆なアフィン変換を段階的に積み重ねた構造。Jacobian が三角行列となり、効率的に計算可能。
- **Glow（Kingma and Dhariwal, 2018）**：Real NVP を改良し、学習安定性と視覚品質を向上。画像生成で有名。
- **FFJORD（Grathwohl et al., 2019）**：ODE ベースの continuous flow による可逆モデル。拡散モデルとの数理的な親和性が高い。
- **応用分野**：異常検知、分子生成、音声モデリング、可逆なデータ圧縮など

まとめ

- フローモデルは、可逆関数の学習によって密度関数と生成モデルを両立させる明示的生成モデルである。
- 拡散モデルと比較すると、密度計算が可能で高速だが、画像品質などはやや劣ることもある。
- 拡散モデルとフローモデルの境界は理論的には連続しており、ODE やスコアベースの視点で接続可能である。

6.5.13　CLIP：言語と画像の統合表現学習

CLIP（Contrastive Language-Image Pretraining）は、OpenAI によって提案された**言語と画像を共通の表現空間で結びつけるための自己教師あり学習モデル**である（Radford et al., 2021）。

1. 基本的な構造と目的

CLIP は、大規模な画像と対応するテキストのペア $(\boldsymbol{x}^{(i)}, \boldsymbol{t}^{(i)})$ を用いて、画像とテキストの意味的な整合性を高めるように共同学習される。

- **画像エンコーダ**：$f_{\text{img}}(\boldsymbol{x})$（ResNet や ViT）
- **テキストエンコーダ**：$f_{\text{text}}(\boldsymbol{t})$（Transformer ベース）

両方の出力は同じ次元のベクトル空間に射影され、**内積に基づく類似度**を最大化するように訓練される。

2. 対照学習による訓練（Contrastive Loss）

与えられたバッチ内の N 組の画像・テキストペアに対して、各画像ベクトル $\boldsymbol{v}_i$ とテキストベクトル $\boldsymbol{u}_i$ の類似度を定義：

$$s_{ij} = \frac{\boldsymbol{v}_i^\top \boldsymbol{u}_j}{\|\boldsymbol{v}_i\| \|\boldsymbol{u}_j\|}$$

損失関数として、以下の双方向のクロスエントロピー損失を用いる：

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left[-\log\frac{\exp(s_{ii}/\tau)}{\sum_{j=1}^{N}\exp(s_{ij}/\tau)} - \log\frac{\exp(s_{ii}/\tau)}{\sum_{j=1}^{N}\exp(s_{ji}/\tau)}\right]$$

ここで τ はスケーリング温度パラメータである。

3. CLIP の特徴

- **自己教師あり学習**：ラベル付きデータを用いず、インターネット上の画像とキャプションの組から大規模に事前学習。
- **共通埋め込み空間**：異なるモダリティ（画像・テキスト）を同じ空間に埋め込むことで、検索・比較・分類が可能に。
- **ゼロショット学習能力**：学習後の CLIP モデルは、明示的なファインチューニングなしで、自然言語を用いた分類や検索に応用できる。

4. 応用例

- **ゼロショット画像分類**：例：「犬」「猫」などのラベルを自然言語で記述し、それとの類似度で分類。
- **画像検索・テキスト検索**：テキストクエリで画像を検索、またはその逆。
- **テキスト条件付き生成**：拡散モデルや GAN における条件ベクトルとして使用（例：DALL-E 2、Stable Diffusion）

まとめ

- CLIP は、画像と言語を共通の意味空間で扱う自己教師ありモデルであり、大規模なテキスト-画像対によって学習される。
- テキストを使った直感的なインターフェースを実現し、多くのマルチモーダル生成モデルにおいて基盤技術として用いられている。
- 学習済み CLIP は強力なゼロショット認識器としても利用可能であり、生成 AI との接続にも適している。

6.5.14　CLIP と正準相関分析（CCA）の関係

CLIP は、画像とテキストといった異なるモダリティのデータを、共通の意味空間に埋め込むモデルである。その目的は、**画像ベクトルとテキストベクトルの意味的一致を最大化すること**であり、この点において古典的な手法である**正準相関分析（CCA）**と類似した目的を持つ。

1. 正準相関分析（CCA）の概要

CCA は、2 つの変数集合（例えば $\boldsymbol{x} \in \mathbb{R}^{d_x}, \boldsymbol{y} \in \mathbb{R}^{d_y}$）に対して、それぞれ線形写像 $w_x^\top \boldsymbol{x}, w_y^\top \boldsymbol{y}$ を求め、**その相関係数を最大化する**問題である：

$$\max_{w_x, w_y} \quad \mathrm{corr}(w_x^\top \boldsymbol{x}, w_y^\top \boldsymbol{y})$$

$$= \frac{w_x^\top \Sigma_{xy} w_y}{\sqrt{w_x^\top \Sigma_{xx} w_x}\sqrt{w_y^\top \Sigma_{yy} w_y}}$$

- Σ_{xy} は $\boldsymbol{x}$ と $\boldsymbol{y}$ の共分散行列
- w_x, w_y は線形射影ベクトル
- 最終的に、複数の「正準変数対」を求めることができる

2. CLIP と CCA の共通点と相違点

■共通点：

- どちらも、**2 つの異なるモダリティのベクトルを共通の空間に写像**することが目的
- 埋め込み空間において、対応関係にあるベクトル同士の**類似性（相関）を最大化**しようとする

■相違点：

- CCA は**線形写像**によって相関を最大化するのに対し、CLIP は**深層ニューラ**

ルネットワークによる非線形写像を用いる

- CLIP は、**softmax を用いた対照学習**によって、画像とテキストのペアを識別的に学習する点が特徴的
- CLIP では、**クロスエントロピー損失に基づく分類タスク**として学習され、CCA とは目的関数が異なる

3. 深層学習における CCA 的手法との統合

CLIP のような手法は、しばしば「**ディープ CCA**」と比較される。これは、深層ニューラルネットワークを用いて非線形変換を学習し、埋め込み空間での相関を最大化する手法である。

CLIP は相関最大化ではなく、**ペアごとの識別**に焦点を当てているが、目的は類似しており、「分類による相関誘導」とも解釈できる。

まとめ

- CLIP は、正準相関分析と同様に、異なるモダリティ間の意味的な整合性を学習するモデルである。
- CCA は線形・相関最大化、CLIP は非線形・識別的類似度最大化という違いがあるが、**共通の空間で対応ベクトルを近づけるという目的は共通**している。
- CLIP は、現代的なディープラーニング技術に基づいた「拡張された CCA 的手法」と位置付けることができる。

6.6　強化学習と深層強化学習の基本構造

6.6.1　1. 強化学習の枠組み

強化学習（Reinforcement Learning）は、**環境とエージェントの相互作用を通じて、報酬を最大化する方策を学習する**枠組みである。主に次のようなマルコフ決定過程（MDP: Markov Decision Process）に基づいて定式化される。

■マルコフ決定過程の構成要素：

- 状態空間 S
- 行動空間 A
- 状態遷移確率 $P(s' \mid s, a)$
- 即時報酬関数 $r(s, a)$
- 割引率 $\gamma \in [0, 1)$

エージェントは状態 s_t を観測し、方策 $\pi(a \mid s)$ に従って行動 a_t を選び、報酬 r_t を受け取りながら環境を移動していく。

2. 強化学習の学習目標

エージェントは、**累積報酬の期待値（割引報酬）**を最大化する方策を学習する：

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

3. 価値関数と Q 関数

- 状態価値関数：$V^\pi(s) = \mathbb{E}_\pi[\sum_t \gamma^t r_t \mid s_0 = s]$
- 状態行動価値関数：$Q^\pi(s, a) = \mathbb{E}_\pi[\sum_t \gamma^t r_t \mid s_0 = s, a_0 = a]$

4. 基本的な強化学習アルゴリズム

- **動的計画法（DP）**：モデルが既知のときに利用
- **モンテカルロ法（MC）**：サンプルに基づいて価値推定
- **TD 学習（Temporal Difference）**：更新に 1 ステップ先を利用
- **Q 学習（Q-Learning）**：

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

6.6.2　5. 深層強化学習の導入

従来の強化学習では、状態や行動空間が小さい場合に有効であったが、**高次元の状態（画像など）**では困難であった。深層強化学習（Deep RL）は、**深層ニューラルネッ**

トワークを用いて価値関数や方策関数を近似することで、この課題を解決する。

6. Deep Q Network（DQN）

DQN（Deep Q-Network）は、Q 学習の Q 関数を深層ニューラルネットワーク $Q_\theta(s, a)$ によって近似する手法である（Mnih et al., 2015）。

■損失関数：

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')} \left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a) \right)^2 \right]$$

■工夫点：

- **経験再生（Experience Replay）**：過去の遷移をランダムにバッファからサンプル
- **固定ターゲットネットワーク**：ターゲット Q 値を安定化するために更新を遅らせたネットワーク θ^- を使用

7. 方策ベースの深層強化学習（Policy Gradient）

価値関数を用いず、方策 $\pi_\theta(a \mid s)$ を直接学習する手法もある。報酬の勾配に従って方策を更新：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s, a) \right]$$

8. 代表的な深層強化学習アルゴリズム

- **DQN**：離散行動空間での価値関数近似
- **DDPG（Deep Deterministic Policy Gradient）**：連続行動空間、actor-critic 構造
- **A3C（Asynchronous Advantage Actor-Critic）**：複数エージェントによる分散学習
- **PPO（Proximal Policy Optimization）**：安定な方策更新を目指した最新手法

まとめ

- 強化学習は、報酬を最大化するような方策を探索する問題設定であり、MDP に基づく。
- 深層強化学習は、深層ネットワークを用いて価値関数や方策関数を高次元状態で近似可能にした。
- DQN、A3C、PPO などの多様なアルゴリズムにより、ロボティクスやゲーム、自然言語処理などへの応用が広がっている。

6.6.3　囲碁 AI と深層強化学習：AlphaGo における応用

1. 背景と課題

囲碁は盤面の状態数が極めて多く、従来の探索ベースの手法（α β 探索など）では処理困難であった。状態空間の爆発的増加に加え、評価関数の設計が難しいことが、人間を超える AI 実現の障壁であった。

2. AlphaGo における深層強化学習の導入

DeepMind によって開発された AlphaGo（2016）は、**深層学習と強化学習、モンテカルロ木探索（MCTS）**を統合した画期的な囲碁 AI である。

■AlphaGo の主要構成：

- **ポリシーネットワーク**（$\pi_\theta(a \mid s)$）：局面 s において次の手 a を予測。人間の棋譜から教師あり学習（supervised learning）で訓練後、自己対局により強化学習。
- **バリューネットワーク**（$v_\phi(s)$）：局面 s の勝率を推定。自己対局の結果を報酬として学習。
- **モンテカルロ木探索（MCTS）**：上記ネットワークに基づき、局面を多数シミュレーションしながら最善手を選択。

■強化学習の活用：　AlphaGo では、**自己対局による強化学習**により、ポリシーとバリューネットワークの精度を継続的に向上させた。これにより、人間レベルを超える戦略が学習された。

3. AlphaZero への発展

AlphaGo の後継である AlphaZero（2017）は、以下の点でさらに一般化された：

- **人間の棋譜を使用せず、完全な自己対局によって学習**（教師なし強化学習）
- 同一のニューラルネットワークがポリシーと価値を同時に出力：

$$f_\theta(s) = (\pi_\theta(\cdot \mid s), v_\theta(s))$$

- モンテカルロ木探索と深層ネットを完全統合し、囲碁だけでなく将棋やチェスにも適用可能な汎用アルゴリズムを実現

4. 深層強化学習との関連性と意義

- AlphaGo/AlphaZero は、**深層強化学習の応用例として最も象徴的な成功例**である。
- 探索（MCTS）と方策学習（ポリシーネット）、価値学習（バリューネット）を統合したフレームワークは、さまざまな戦略課題に応用可能である。
- 自己対局による経験から学習する構造は、モデルベース強化学習とモデルフリー強化学習のハイブリッドにも通じる。

まとめ

- 囲碁 AI における AlphaGo は、深層学習と強化学習を統合した先駆的モデルである。
- AlphaZero により、教師なしで戦略を習得可能な汎用強化学習フレームワークが構築された。
- 深層強化学習の実世界応用において、探索と学習の統合は今後も重要なテーマである。

6.7　AlphaFold2：タンパク質構造予測における深層学習の革新

6.7.1　1. バイオインフォマティクスにおける背景

生体内のタンパク質は、アミノ酸配列（一次構造）に基づいて、特有の三次元構造（立体構造）をとる。この構造はタンパク質の機能にとって決定的に重要である。しかし、実験的な構造決定（X 線結晶構造解析、NMR、クライオ電顕など）は高コストかつ時間がかかるため、**アミノ酸配列から立体構造を予測する計算手法**が求められてきた。

6.7.2　2. 従来の手法と課題

従来のタンパク質構造予測法は主に次のように分類される：

- **ホモロジーモデリング**：既知構造と配列類似性の高いテンプレートを利用
- **アブイニシオ法**：物理エネルギーに基づく構造シミュレーション
- **共進化情報の利用**：多配列アラインメント（MSA）からペアワイズ相関を推定

これらは一部で成功を収めたが、一般的には精度と計算コストのバランスに課題があった。

6.7.3　3. AlphaFold2 の革新

DeepMind による AlphaFold2（2020）は、深層学習を大規模に活用し、**アミノ酸配列と共進化情報から直接三次元構造を予測**する画期的なモデルである。

AlphaFold2 の特徴：

- 進化情報を活用した**複数配列アラインメント（MSA）**とテンプレート構造を組み合わせて入力
- Transformer 系ネットワークを応用した **Evoformer モジュール** による情報統合

- 原子座標を直接出力する **Structure module**
- 自己知識蒸留による大規模学習と多段階反復処理（recycling）

6.7.4 4. AlphaFold2 のアーキテクチャ全体像

1. **入力処理**：
   - タンパク質一次配列
   - 多配列アラインメント（MSA）：配列間の共進化関係を抽出
   - 既知の類似構造（テンプレート）情報
2. **Evoformer モジュール**：Transformer に類似した構造で、MSA とペア表現（residue pair）を交互に情報伝搬。アテンション機構により残基間相互作用や進化パターンをモデル化。
3. **Structure Module**：予測されたペア表現をもとに、**残基ごとの原子の 3D 座標を直接予測**する。座標予測後は幾何的拘束条件（bond length/angle）に従って微調整。
4. **Recycling（再利用機構）**：予測構造を再びネットワークに入力して繰り返し精度を高めるループ処理。

6.7.5 5. AlphaFold2 の意義と応用

- **構造生物学のパラダイムシフト**：実験なしに高精度の構造予測が可能
- **全生物種の構造データベース構築**（AlphaFold DB）
- 創薬、機能解析、酵素設計、疾患研究などへの応用が急速に広がっている

関連技術との統合

- CLUSTAL、HHblits などによる MSA 構築
- 自己知識蒸留（self-distillation）による教師信号の生成
- グラフニューラルネットワークに基づく改良（例：RoseTTAFold）

まとめ

- AlphaFold2 は、進化情報、構造テンプレート、深層学習を統合した、これまでにない精度のタンパク質構造予測システムである。
- 従来の物理ベースや相関ベースの手法を凌駕し、多くの生物学的研究にインパクトを与えている。

参考文献

[1] G. James, D. Witten, T. Hastie, R. Tibshirani “An Introduction to Statistical Learning, Second Edition”, Springer 2023

[2] C.M. Bishop, “Pattern recognition and machine learning”, Springer 2006 (ビショップ：パターン認識と機械学習（上下），丸善)

[3] 萩原，入門 統計的回帰とモデル選択，共立出版 2022

[4] 赤穂，カーネル多変量解析，岩波書店 2008

[5] 青嶋，矢田，高次元の統計学，共立出版 2019

[6] S. Watanabe, M. Opper, Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. Journal of machine learning research, 11(12) 2010