機械学習の心臓部：活性化関数・損失関数・最適化の連携プレー入門

1. はじめに

機械学習モデルの「学習」とは？

機械学習、特に近年注目を集めるニューラルネットワークは、まるで人間が経験から学ぶように、データの中に潜むパターンやルールを見つけ出し、未知のデータに対しても的確な予測や判断を行う能力を獲得します。この驚くべき能力の獲得プロセスが、機械学習における「学習」です。
この学習プロセスの核心には、「予測 → 評価 → 改善」という繰り返しのサイクルが存在します。モデルはまず、与えられたデータに基づいて予測を行い（予測）、その予測がどれだけ正解に近いかを評価し（評価）、評価結果をもとに、より良い予測ができるように自身の内部設定を調整します（改善）。このサイクルを何度も繰り返すことで、モデルは徐々に賢くなっていきます。この学習サイクルを力強く駆動させるエンジンとなるのが、本レポートで解説する「活性化関数」「損失関数」「最適化」という3つの重要な要素です[1]。

本レポートの目的と構成

本レポートでは、機械学習モデルが学習を行う上で不可欠なこれら3つの要素、すなわち活性化関数、損失関数、そして最適化アルゴリズムについて、それぞれの役割、代表的な種類、そしてそれらがどのように連携してモデルの学習を可能にするのかを、機械学習の学習を始めたばかりの方にも理解しやすいように解説します。専門用語は避けられない部分もありますが、その都度丁寧に説明し、簡単な例や比喩を交えながら、各要素がなぜモデルの性能向上に不可欠なのかを明らかにしていきます。

2. 活性化関数：ニューラルネットワークに表現力を与える

役割：ニューロンの発火を制御するスイッチ

ニューラルネットワークは、人間の脳神経系を模倣した数理モデルで、多数の「ニューロン」と呼ばれる計算ユニットが層状に接続された構造を持っています。入力データは、このネットワークの入力層から入り、複数の「隠れ層」を経て、最終的に出力層から予測結果として出力されます[3]。
各ニューロンは、前の層にある複数のニューロンから信号を受け取ります。これらの入力信号は、それぞれの接続の「重み」に応じて強調されたり抑制されたりした後、合計されます[3]。活性化関数は、この合計された入力値を処理し、そのニューロンが「発火」する（＝次の層に意味のある信号を送る）かどうか、そしてどのような強さの信号を送るかを決定する、いわば「スイッチ」のような役割を担っています[3]。これは、単に入力値を足したり掛けたりする線形的な処理ではなく、ニューロンの出力の仕方を決定づける非常に重要なステップです[3]。

非線形性の導入：なぜ重要なのか？

活性化関数の最も重要な役割の一つが、「非線形性」をニューラルネットワークに導入することです。では、なぜ非線形性がそれほど重要なのでしょうか？
もし活性化関数が線形関数（例えば y=ax+b のような、グラフにすると直線になる関数）だった場合、ニューラルネットワークの層をどれだけ深く積み重ねても、ネットワーク全体としては依然として一つの線形関数にしかなりません [4]。線形関数は、入力と出力の間に単純な比例関係しか表現できず、複雑な現象を捉える能力には限界があります。例えば、2つのグループのデータを分類する場合、線形関数では直線を引いて分けることしかできません [6]。
しかし、私たちが機械学習で扱いたい現実世界のデータ（例えば、写真に写っている動物の種類、人間の話し声、株価の変動など）は、ほとんどの場合、単純な直線では説明できない非常に複雑なパターンや関係性を持っています [8]。画像認識であれば、対象物の曲線的な輪郭や複雑なテクスチャを認識する必要がありますし、音声認識であれば、時間とともに変化する複雑な周波数パターンを捉える必要があります。
ここで非線形な活性化関数の出番です。非線形な活性化関数（グラフにすると曲線になる関数）をニューロンに組み込むことで、ニューラルネットワークは直線だけでなく、複雑な曲線的な関係性や、入り組んだ境界線を学習する能力を獲得します [3]。これにより、モデルの「表現力」（どれだけ複雑な関数を近似できるか）が飛躍的に向上し、画像認識や自然言語処理といった困難なタスクに取り組むことが可能になるのです [4]。もし非線形性がなければ、どんなに層を深くしても、ニューラルネットワークは本質的には単純な線形モデルと変わらず、その能力は著しく制限されてしまいます [3]。

比喩：まっすぐな道 **vs.** 曲がりくねった道

この非線形性の重要性を、道を作ることに例えてみましょう。活性化関数がない、あるいは線形な活性化関数しか使えない場合、モデルはまっすぐな道しか作れません。平坦な土地なら問題ありませんが、山や谷がある複雑な地形（データ）には対応できません。一方、非線形な活性化関数は、モデルに曲がりくねった道を作る能力を与えます。これにより、どんなに複雑な地形（データ）に対しても、適切に対応する道（モデル）を作ることができるようになるのです [3]。

比喩：積み木

別の例えとして、積み木を考えてみましょう。線形モデルは、「直線の棒」という種類の積み木しか使えません。これだけでは、単純な直線的な構造物しか作れません。しかし、非線形モデルは、「曲がる棒」のような特殊な積み木も使うことができます。これにより、曲線を含む、より複雑で自由な形（データパターン）を表現できるようになるのです [8]。

代表的な活性化関数

活性化関数には様々な種類がありますが、ここでは特に代表的なReLU関数とシグモイド関数を紹介します。

ReLU (Rectified Linear Unit) 関数

- 仕組み: ReLU関数は非常にシンプルな関数で、入力値が0以下であれば0を出力し、入力値が0より大きければその値をそのまま出力します [4]。数式で書くと f(x)=max(0,x) となります。
- 特徴:
  - 計算効率: 最大値を求めるだけの単純な計算なので、非常に高速に処理できます [4]。これは、多数のニューロンで計算を行うニューラルネットワークにとって大きな利点です。
  - 勾配消失問題の緩和: 後述するシグモイド関数が抱える「勾配消失問題」を起こしにくいという特徴があります。入力が正の領域では勾配（傾き）が常に1であるため、誤差がネットワークの深い層まで伝わりやすくなります [4]。
  - スパース性: 入力が負の場合に出力が0になるため、ネットワーク内の一部のニューロンだけが活性化（発火）する状態（スパースな活性化）が生まれやすくなります。これが、より効率的な学習や汎化性能の向上に繋がることがあります [13]。
- 短所:
  - **Dying ReLU**問題: 学習中にニューロンへの入力が常に負になってしまうと、そのニューロンの出力は常に0となり、勾配も0になってしまいます。その結果、そのニューロンの重みは更新されなくなり、学習に寄与しなくなってしまう「死んだニューロン」が発生する可能性があります [4]。
- 用途: そのシンプルさと性能の良さから、現在、ニューラルネットワークの中間層（隠れ層）で最も広く使われている活性化関数です [4]。

シグモイド (Sigmoid) 関数

- 仕組み: シグモイド関数は、入力を0から1の間の値に滑らかに変換する関数で、そのグラフはS字カーブを描きます [4]。数式では f(x)=1+e−x1 と表されます。
- 特徴:
  - 確率的出力: 出力が0から1の範囲に収まるため、その値を確率として解釈することができます。この性質から、2つのクラスのどちらに属するかを予測する「二値分類問題」の出力層でよく用いられます [4]。
  - 滑らかな微分: グラフが滑らかな曲線であるため、どの点でも微分が可能です。
- 短所:
  - 勾配消失問題: 入力値の絶対値が大きくなる（グラフの両端に近づく）と、関数の傾き（勾配）がほぼ0になってしまいます [4]。ニューラルネットワークの層が深くなると、誤差逆伝播（後述）の際に勾配がどんどん小さくなり、入力層に近い層では重みがほとんど更新されなくなる「勾配消失問題」を引き起こし、学習が停滞する大きな原因となります [4]。
  - 非ゼロ中心: 出力が常に正の値（0〜1）を取るため、出力がゼロ中心になっていません。これが学習の効率を低下させる要因となる場合があります [4]。
  - 計算コスト: 指数関数 (e−x) の計算を含むため、ReLU関数と比較して計算コストが高くなります [4]。
- 用途: 主に二値分類問題の出力層で使用されます。勾配消失問題のため、深いネットワークの中間層で使用されることは現在では少なくなっています [4]。

活性化関数の選択は文脈依存

これらの例からわかるように、「常に最適な活性化関数」というものは存在しません。ReLUは中間層の標準的な選択肢として広く使われていますが [4]、出力層の活性化関数は解きたいタスクの種類によって決まります [7]。二値分類ならシグモイド [4]、複数のクラス分類ならソフトマックス関数 [4]、数値を直接予測する回帰問題なら活性化関数を使わない（線形活性化）こともあります [6]。
さらに、ReLUのDying ReLU問題を解決するために、Leaky ReLU [11] やELU [11] といった派生的な活性化関数も提案されており、特定の状況下でより良い性能を発揮することがあります。このように、活性化関数の選択は、ネットワークの構造、データの特性、そして解きたい問題の性質を考慮して行われるべき重要な設計判断なのです。

活性化関数の比較

特徴項目	**ReLU**関数	シグモイド関数
数式	f(x)=max(0,x)	f(x)=1+e−x1
グラフ形状	0で折れ曲がる直線	S字カーブ
出力範囲	$。この計算された「ずれ」の大きさは、「損失（Loss）」や「コスト（Cost）」、「誤差（Error）」などと呼ばれます [1]。	

損失関数の値が小さいほど、モデルの予測が実際の値に近い、つまり「モデルがより正確である」ことを意味します [25]。逆に、損失の値が大きいほど、予測が大きく外れていることを示します。

なぜ必要か：学習の道しるべ

損失関数は、単にモデルの間違いの大きさを測るだけではありません。機械学習モデルの「学習（トレーニング）」とは、本質的にこの損失関数の値をできるだけ小さくする（最小化する）プロセスなのです [1]。
損失関数は、モデルがどれだけ間違っているかを具体的な数値で示すことで、学習の「目標」を設定します。そして、その数値を最小化すべき目標（目的関数）とすることで、モデルが自身の内部パラメータ（重みやバイアス）をどちらの方向に調整すれば、より正解に近づけるのか、という「道しるべ」を提供するのです [1]。後述する最適化アルゴリズムは、この道しるべに従って、損失が小さくなる方向へとモデルのパラメータを導いていきます。

代表的な損失関数

損失関数にも様々な種類があり、解きたい問題の種類（回帰か分類かなど）に応じて使い分けられます。ここでは、最も代表的な2つの損失関数、平均二乗誤差と交差エントロピー誤差について解説します。

平均二乗誤差 (Mean Squared Error - MSE / L2 Loss)

- 計算の考え方: 各データ点におけるモデルの予測値と実際の正解値の差を計算し、その差を二乗します。そして、すべてのデータ点について計算した二乗誤差を合計し、データ点の数で割って平均値を出します [21]。数式で表すと以下のようになります。 MSE=N1i=1ΣN(yi−y^i)2 ここで、N はデータ点の数、yi は i 番目のデータの実際の正解値、y^i はモデルによる予測値を表します。
- 主な用途: 連続的な数値を予測する「回帰問題」で最も一般的に用いられる損失関数です [21]。例えば、家の広さや築年数から価格を予測する、過去の気温データから明日の気温を予測する、といったタスクに適しています。
- 特徴:
    - 誤差の強調: 予測値と正解値の差を二乗するため、誤差が大きいほど損失の値は急激に大きくなります。つまり、大きな間違いに対してより厳しいペナルティを与えます [24]。
    - 外れ値への感度: 上記の性質から、データセットの中に極端に値が異なる「外れ値」が存在する場合、その影響を強く受けやすいという特徴があります [21]。モデルは、損失を小さくするために、これらの外れ値に予測を近づけようとする傾向があります [24]。
    - 数学的扱いやすさ: 数学的に微分が容易であるため、最適化アルゴリズム（特に勾配を用いる手法）との相性が良いです。

交差エントロピー誤差 (Cross-Entropy Error / Log Loss)

- 計算の考え方: モデルが予測した「確率分布」と、実際の正解の「確率分布」がどれだけ異なっているか（隔たっているか）を測る指標です [21]。分類問題では、正解のクラスに対応する確率が1で、それ以外のクラスの確率が0であるような分布を理想とします。モデルが正解クラスに対して低い確率を予測したり、逆に不正解クラスに対して高い確率を予測したりすると、交差エントロピー誤差の値は大きくなります [21]。 例えば、犬、猫、鳥の3クラス分類で、ある画像が「猫」である（正解分布: [犬0, 猫1, 鳥0]）場合に、モデルが [犬0.1, 猫0.7, 鳥0.2] と予測したとします。この場合、交差エントロピー誤差は主に正解である「猫」クラスの予測確率（0.7）に基づいて計算され、この確率が1に近いほど誤差は小さくなります。
- 主な用途: データがどのカテゴリに属するかを予測する「分類問題」で標準的に用いられる損失関数です [21]。特に、出力層で確率を出力するシグモイド関数（二値分類）やソフトマックス関数（多クラス分類）と組み合わせて使われることが多いです [1]。画像分類（写真に写っているのが犬か猫か）、スパムメール判定（メールがスパムか否か）、文章の感情分析（文章がポジティブかネガティブか）などのタスクに適しています。

- 特徴:
    - 確率のずれを評価: モデルの予測確率が、真のラベル（正解クラス）からどれだけ乖離しているかを直接的に評価するのに適しています [31]。
    - 情報理論的背景: 情報理論における「KLダイバージェンス」という概念と密接に関連しており、2つの確率分布間の差異を測る尺度として理論的な裏付けがあります [38]。

損失関数の選択はタスクの目標とデータの特性を反映する

MSEが回帰問題に、交差エントロピーが分類問題に適しているという事実は、損失関数が単なる計算式ではなく、解きたいタスクの「目標」そのものを定義していることを示しています [21]。回帰では予測値と真の値の「距離」が重要であり、MSEはその二乗距離を測ります。一方、分類では、モデルが正しいクラスをどれだけ「確信」しているかが重要であり、交差エントロピーは確率分布の「ずれ」を測ります。
さらに、MSEが外れ値に敏感であるのに対し、平均絶対誤差（MAE、誤差の絶対値の平均を取る損失関数）は外れ値の影響を受けにくいという性質があります [24]。これは、データの特性（外れ値の有無など）も損失関数の選択に影響を与えることを示唆しています。
不適切な損失関数を選択すると、モデルは間違った目標に向かって学習を進めてしまいます [39]。例えば、分類問題をMSEで学習させようとしたり、外れ値が多い回帰問題で何も考えずにMSEを使ったりすると、たとえ損失関数の値が小さくなったとしても、本来解きたかったタスクにおける性能は最適にならない可能性があります。したがって、損失関数の選択は、タスクの種類、データの特性、そして最終的に達成したい目標を慎重に考慮して行う必要があるのです [22]。

損失関数の比較

特徴項目	平均二乗誤差 **(MSE)**	交差エントロピー誤差 **(Cross-Entropy)**
主な用途	回帰問題 (連続値予測)	分類問題 (カテゴリ予測)
計算の考え方	予測値と正解値の差の二乗の平均	予測確率分布と正解分布の間の差異
主な特徴	大きな誤差をより強調、外れ値に敏感	予測確率のズレを評価、確率的な出力に適している
長所	微分が容易、数学的に扱いやすい	分類タスクの評価に適している、確率出力と相性が良い
短所	外れ値の影響を受けやすい	回帰問題には直接適用できない

4. 最適化：損失を最小化する旅

役割：最良のパラメータを見つけ出すエンジン

活性化関数がモデルに「どのような道を作れるか（表現力）」を与え、損失関数が「どこがゴールか（損失最小点）」を示すとすれば、最適化アルゴリズムは「どのようにしてそのゴールに効率的にたどり着くか」という具体的な道のり（探索戦略）を提供する役割を担います [2]。
機械学習モデル、特にニューラルネットワークは、膨大な数の調整可能なパラメータ（主にニューロン間の接続の重みと、各ニューロンのバイアス）を持っています。最適化の目的は、これらのパラメータをうまく調整することで、損失関数の値をできるだけ小さくする（最小化する）ようなパラメータの組み合わせを見つけ出すことです [1]。これは、広大で複雑な「パラメータ空間」の中から、最も損失が低い「谷底」を探す旅に例えられます。

代表的な最適化アルゴリズム

最適化アルゴリズムにも多くの種類がありますが、ここでは基本となる勾配降下法と、現在広く使われているAdamについて解説します。

勾配降下法 (Gradient Descent - GD)

- 基本的な考え方: 最適化アルゴリズムの最も基本的な考え方の一つです。損失関数を、パラメータを軸とした一種の「地形」と見なします。勾配降下法は、現在いる地点（現在のパラメータ値）から見て、最も傾斜が急な下り坂の方向（損失関数の勾配の逆方向）に少しずつ移動していくことで、最終的に地形の最も低い場所、つまり損失の最小値（谷底）に到達することを目指します [2]。
- 仕組み:
    1. まず、パラメータをランダムな値で初期化します [2]。
    2. 現在のパラメータ値を使って、損失関数を各パラメータで偏微分し、「勾配」を計算します。勾配は、各パラメータを少し動かしたときに損失がどれだけ変化するか、そしてどの方向に動かすと損失が最も増えるかを示します [2]。
    3. 計算された勾配と逆の方向に、事前に設定した「学習率（ステップ幅）」を掛けた分だけ、各パラメータの値を更新します。これにより、損失が減少する方向にパラメータが少し移動します [2]。
    4. ステップ2と3を、損失の変化が十分小さくなるか、決められた回数繰り返します [42]。
- 学習率 (Learning Rate): 勾配降下法において非常に重要な調整パラメータ（ハイパーパラ

メータ）です。これは、パラメータを更新する際に一歩あたりに進む「歩幅」を決定します [2]。学習率が小さすぎると、谷底にたどり着くまでに非常に多くのステップが必要となり、学習に時間がかかりすぎます。逆に、学習率が大きすぎると、歩幅が大きすぎて谷底を飛び越えてしまい、損失が増加したり、パラメータが発散（振動）して学習が収束しなくなったりする可能性があります [2]。適切な学習率を見つけることが、効率的な学習の鍵となります。

- 長所: アルゴリズムの考え方がシンプルで、直感的に理解しやすい点です。
- 短所:
    - 局所最適解 **(Local Minima):** 損失関数の地形が複雑で、谷底（極小値）が複数存在する場合、最初に探索を開始した場所によっては、全体の最も低い谷底（大域的最適解）ではなく、近くにある浅い谷底（局所最適解）に到達してしまい、そこで探索が終了してしまう可能性があります [2]。
    - 計算コスト: 勾配を計算するために、訓練データセットに含まれる全てのデータを使う必要があります。データセットが非常に大きい場合、1回のパラメータ更新に必要な計算量が膨大になり、学習が非常に遅くなるという問題があります [40]。
    - 鞍点・平坦な領域: 勾配がゼロになるものの最小値ではない「鞍点（Saddle Point）」や、勾配が非常に小さい「平坦な領域（Plateau）」で学習が停滞してしまうことがあります [2]。

確率的勾配降下法 (Stochastic Gradient Descent - SGD)

- **GD**の改良: 上記のGDの計算コスト問題を解決するために考案された手法です。データセット全体を使う代わりに、データセットからランダムに選び出した一部のデータ（「ミニバッチ」と呼ばれる小さなグループ、あるいは究極的には1つのデータ）だけを使って勾配を計算し、パラメータを更新します [40]。
- メリット: 1回の更新に必要な計算量が大幅に削減されるため、GDよりもはるかに高速に学習を進めることができます。また、毎回異なるデータで勾配を計算するため、更新にノイズ（ばらつき）が生じますが、このノイズが局所最適解や鞍点から抜け出す助けとなることがあります [40]。
- デメリット: 更新がデータの一部に依存するため、パラメータの更新経路が不安定になりやすく、損失がジグザグに変動（振動）することがあります [52]。学習率の調整がより重要になります。

Adam (Adaptive Moment Estimation)

- **SGD**の進化形: SGDをベースに、いくつかの改良を加えて性能を高めた最適化アルゴリズムで、現在、深層学習の分野で最も広く使われている手法の一つです [33]。Adamは、「モーメンタム」と「適応的学習率」という2つの主要なアイデアを組み合わせています。
- 特徴:
    - モーメンタム **(Momentum):** 過去のパラメータ更新の方向（移動平均）を現在の更新に加味します。これは、坂道を転がるボールが持つ「慣性（勢い）」のようなものです [40]。勾配の方向が安定している場合は更新を加速させ、勾配が頻繁に変わる場合は振動を

抑制する効果があります。これにより、SGDよりも速く谷底に向かい、浅い局所最適解を乗り越えやすくなります [40]。
    - 適応的学習率 **(Adaptive Learning Rate):** 全てのパラメータに同じ学習率を適用するのではなく、パラメータごとに個別に学習率を調整します。具体的には、過去の勾配の大きさ（二乗平均）に基づいて学習率を調整します（RMSpropやAdaGradという手法のアイデアを取り入れています） [40]。頻繁に更新される（勾配が大きい）パラメータの学習率は小さめに、あまり更新されない（勾配が小さい）パラメータの学習率は大きめに調整されます。これにより、パラメータ空間をより効率的に探索できるようになります。
- 長所**:** 多くの場合、SGDや他の最適化手法と比較して、より高速に損失が収束し、学習率などのハイパーパラメータの調整が比較的容易であるとされています [41]。そのため、多くの深層学習フレームワークでデフォルトの最適化手法として採用されています [41]。
- 短所**:** 常に最良の性能を発揮するとは限らず、問題によってはSGDにモーメンタムを加えた手法の方が最終的なモデルの汎化性能（未知のデータに対する性能）が高くなる場合もある、という研究報告もあります [52]。また、理論的な収束保証に関して議論がある場合もあります。

最適化アルゴリズムの進化：勾配降下法の限界を超えるために

基本的な勾配降下法（GD）は、理論はシンプルですが、実際の複雑な問題に適用するには、計算コスト、局所最適解、学習率の調整といった課題を抱えています [2]。SGD、モーメンタム、AdaGrad、RMSprop、そしてAdamといったアルゴリズムの発展の歴史は、これらのGDの限界を克服しようとする試みの連続でした [40]。
SGDは計算コストの問題に取り組みました [40]。モーメンタムは、慣性を導入することで収束を速め、局所最適解や平坦な領域からの脱出を助けました [40]。AdaGrad、RMSprop、そしてAdamは、パラメータごとに学習率を適応的に調整するメカニズムを導入し、より効率的な探索と収束を実現しました [40]。特にAdamは、モーメンタム（勾配の一次モーメント）と適応的学習率（勾配の二次モーメント）を組み合わせることで、多くの場合で高い性能を発揮し、広く使われるようになりました [41]。
この進化の過程は、深層学習モデルが持つような高次元で複雑な損失関数の地形を、より賢く、より速く探索するための技術が洗練されてきたことを示しています。

最適化アルゴリズムの比較

特徴項目	勾配降下法 **(GD)**	確率的勾配降下法 **(SGD)**	**Adam**
基本的な考え方	全データの勾配で一気に更新	一部データ(ミニバッチ)の勾配で頻繁に更新	モーメンタムと適応的学習率を組み合わせる
勾配計算の対象データ	データセット全体	ミニバッチ (または1データ)	ミニバッチ
学習率	固定 (手動調整)	固定 (手動調整、より慎重な調整が必要)	適応的 (パラメータ毎に自動調整)
主な長所	シンプル、理論的解析が	計算が高速、局所解脱	高速な収束、ハイパー

	容易	出の可能性	パラメータ調整が比較的容易
主な短所	計算コスト大、局所解・鞍点に弱い	更新が不安定(振動)、学習率調整がシビア	常に最適とは限らない、汎化性能で劣る場合も、メモリ消費量がやや多い

5. 連携プレー：活性化関数、損失関数、最適化の協力

これまで見てきた活性化関数、損失関数、最適化アルゴリズムは、それぞれ独立して機能するのではなく、互いに密接に連携し、一つのチームとして機械学習モデルの学習プロセスを推進します。ここでは、その連携プレーがどのように行われるのかを、ステップ・バイ・ステップで見ていきましょう。

学習プロセスの全体像

モデルの学習は、以下の3つのステップからなるサイクルを何度も繰り返すことで進行します [1]。

1. フォワードプロパゲーション **(Forward Propagation):** モデルが入力データを受け取り、予測値を計算する。
2. 損失計算 **(Loss Calculation):** モデルの予測値と実際の正解値を比較し、間違いの大きさ（損失）を計算する。
3. バックプロパゲーションとパラメータ更新 **(Backpropagation and Parameter Update):** 損失を減らすために、モデルの内部パラメータ（重みとバイアス）を調整する。

このサイクルを通じて、モデルは徐々にデータに適合し、予測精度を高めていきます。

ステップ・バイ・ステップ解説

(a) フォワードプロパゲーション **(Forward Propagation):** 予測値を計算する

1. **入力:** まず、訓練データセットから取り出された入力データ（例えば、画像ピクセル値や文章の単語ベクトルなど）が、ニューラルネットワークの入力層に与えられます [4]。
2. **層間の伝播:** データは、入力層から隠れ層、そして出力層へと、層から層へと順番に伝播していきます。
3. **ニューロンの計算:** 各層の各ニューロンでは、まず前の層のニューロンからの出力信号を受け取ります。これらの信号は、接続ごとに設定された「重み」で掛け算され、すべて合計されます。さらに、各ニューロン固有の「バイアス」と呼ばれる値が加えられます。ここまでの計算（重み付き合計＋バイアス）は「線形変換」です [3]。

4. 活性化関数の適用: 次に、この線形変換の結果が、そのニューロンに設定された活性化関数（例：中間層ならReLU）に入力されます。活性化関数は、この線形的な値を非線形的な値に変換し、それがそのニューロンの最終的な出力信号となります [3]。この非線形変換が、ネットワークに複雑なパターンを学習する能力を与えます。
5. 最終出力: この「線形変換 → 活性化関数による非線形変換」というプロセスが、隠れ層を経て出力層まで繰り返されます。出力層のニューロン（とその活性化関数、例：分類ならソフトマックス、回帰なら線形）が出力した値が、モデルの最終的な予測値となります [3]。

(b) 損失計算 **(Loss Calculation):** 予測の「間違い」を測る

1. 予測と正解の比較: フォワードプロパゲーションによって得られたモデルの予測値と、その入力データに対応する実際の正解値（訓練データに含まれるラベル）を比較します [1]。
2. 損失関数の適用: この予測値と正解値のペアを、事前に定義された損失関数（例：回帰ならMSE、分類なら交差エントロピー誤差）に入力します。
3. 損失値の算出: 損失関数は、予測と正解の間の「ずれ」を計算し、一つの数値（損失値）として出力します [1]。この損失値が、現在のモデルのパラメータ設定における「間違いの大きさ」を表します。

(c) バックプロパゲーションとパラメータ更新 **(Backpropagation and Parameter Update):** モデルを修正する

1. 目標: 計算された損失値を最小化することが目標です。そのためには、損失値に影響を与えているモデルの各パラメータ（全ての層の全ての重みとバイアス）を、損失が減少する方向に調整する必要があります [1]。
2. 勾配計算 **(**バックプロパゲーション**):** ここで最適化アルゴリズム（例：SGDやAdam）が登場します。最適化アルゴリズムは、「バックプロパゲーション（誤差逆伝播法）」と呼ばれるアルゴリズムを用いて、損失値に対する各パラメータの勾配（偏微分）を計算します [1]。バックプロパゲーションは、出力層から入力層に向かって、微分法の連鎖律（Chain Rule）を巧みに利用し、ネットワーク全体の膨大なパラメータに関する勾配を効率的に計算する手法です [25]。この勾配は、「各パラメータを少し変化させると、損失がどれだけ変化するか」を示しています。
3. パラメータ更新: 最適化アルゴリズムは、バックプロパゲーションによって計算された勾配の情報を使って、各パラメータを更新します。基本的には、勾配が示す「損失が増加する方向」とは逆の方向に、学習率でスケーリングされた分だけパラメータを移動させます [2]。これにより、損失が減少する方向にパラメータが調整されます。
4. 反復: この「フォワードプロパゲーション → 損失計算 → バックプロパゲーション → パラメータ更新」という一連のサイクルを、訓練データセット全体（またはミニバッチ）に対して何度も何度も（「エポック」と呼ばれる単位で）繰り返します [1]。この反復的なプロセスを通じて、モデルは徐々にデータからパターンを学習し、損失を最小化するようなパラメータの値へと近づいていきます。

緊密に連携するフィードバックループ

このステップ・バイ・ステップの解説から明らかなように、活性化関数、損失関数、最適化アルゴリズムは、単なる一連の処理ではなく、緊密に連携したフィードバックループを形成しています。

- 活性化関数はフォワードプロパゲーションにおいて予測の「形」を作ります [3]。
- 損失関数はその予測を評価し、「誤差信号」を生成します [22]。
- 最適化アルゴリズムは、バックプロパゲーションを通じて計算された誤差信号の勾配に基づき、次のフォワードプロパゲーションで使われるパラメータ（重みとバイアス）を更新します [1]。

このループのどこか一つでも欠けたり、不適切なものが使われたりすると（例えば、微分不可能な損失関数 [22]、タスクに合わない活性化関数、非効率な最適化アルゴリズムなど）、学習プロセス全体がうまく機能しなくなります。これら3つの要素が調和して働くことで、初めてニューラルネットワークは効果的に学習を進めることができるのです。

簡単な例や比喩を用いた説明

この複雑な連携プレーを、より直感的に理解するために、いくつかの比喩を用いて説明しましょう。

全体像の比喩：料理のレシピ調整

機械学習モデルの学習プロセス全体を、美味しい料理を作るためのレシピ調整に例えることができます。

- **モデル（パラメータ）:** 料理のレシピ（材料の分量、火加減、調理時間など）です。最初は適当なレシピから始まります。
- **活性化関数:** 食材に変化を加える「調理法」（焼く、煮る、蒸すなど）に相当します。これにより、単なる材料の混合（線形）ではない、複雑な味（非線形）が生み出されます。
- **フォワードプロパゲーション:** 現在のレシピに従って、実際に料理を作ってみるプロセスです。
- **予測値:** 出来上がった料理そのものです。
- **正解値:** 目指している理想の料理の味（例えば、有名シェフの料理の味）です。
- **損失関数:** 出来上がった料理を「味見」して、理想の味とどれだけ違うかを評価するプロセスです。「塩味が足りない」「少し甘すぎる」といった評価（損失）が得られます。
- **最適化（バックプロパゲーション）:** 味見の結果（損失）に基づいて、「どの材料の分量をどれだけ変えれば（パラメータ調整）、理想の味に近づくか」を分析し、レシピを修正するプロセスです。例えば、「塩味が足りないなら、塩を少し増やす」といった具体的な修正を行います。
- **学習:** この「料理を作る → 味見して評価 → レシピ修正」というサイクルを何度も繰り返し、徐々に理想の味の料理が作れるようにレシピを改良していくプロセス全体が、機械学習の学習に相当します。

勾配降下法の比喩:「霧の中の谷下り」

最適化アルゴリズムの代表である勾配降下法は、よく「霧の中の谷下り」に例えられます[2]。

- あなたは深い霧がかかった山の中(損失関数の複雑な地形)に立っています。目標は、山の最も低い場所である谷底(損失の最小点)にたどり着くことです。
- しかし、霧(パラメータ空間の広大さや複雑さ)のために、谷底がどちらの方向にあるのか、どれくらい遠いのかは直接見えません。
- 唯一の手がかりは、自分の足元(現在のパラメータ地点)の地面の傾斜(損失関数の勾配)だけです[2]。
- あなたは、足元で感じられる最も急な下り坂の方向(勾配の逆方向)を選んで、一歩ずつ慎重に進んでいきます[2]。
- この一歩の大きさ(学習率)が重要です。歩幅が大きすぎると、勢い余って谷底を通り過ぎて反対側の斜面を登ってしまうかもしれません。逆に歩幅が小さすぎると、谷底にたどり着くまでに非常に長い時間がかかってしまいます[2]。
- この「傾斜を確認して、下る方向に一歩進む」というプロセスを繰り返すことで、霧の中でもいずれ谷底にたどり着けるだろう、というのが勾配降下法の考え方です。

活性化関数の比喩:「脳の神経細胞の働き」

活性化関数は、私たちの脳の神経細胞(ニューロン)の働きに例えることができます[56]。

- 脳のニューロンは、他の多数のニューロンから電気信号(入力)を受け取ります。
- 受け取った信号の合計がある一定の「閾値」を超えると、そのニューロンは「発火」し、次のニューロンへと信号を伝達します。閾値に満たなければ発火しません。
- 活性化関数は、このニューロンの発火を制御する「スイッチ」のような役割や、発火した際の信号の強さを調整する役割に似ています[4]。
- 脳が単純な反射だけでなく、複雑な思考や認識を行えるのは、このようなニューロンの非線形な応答(単なる入力の合計ではない、発火するかしないか、どの程度の強さで発火するかという応答)が組み合わさっているからです。同様に、ニューラルネットワークも非線形な活性化関数を持つことで、複雑な情報処理が可能になります。

6. なぜこれらは不可欠なのか?モデル性能向上の鍵

最後に、活性化関数(特にその非線形性)、損失関数の選択、そして最適化手法が、なぜ機械学習モデルの性能向上にとって「不可欠」なのかを、それぞれの観点から掘り下げてみましょう。

活性化関数(非線形性)の不可欠性:複雑な現実世界を捉えるために

繰り返しになりますが、私たちが機械学習で解きたい問題の多くは、現実世界の複雑な現象に基づ

いています。画像の中の物体の形状、音声のパターン、自然言語の文法構造、金融市場の動きなど、これらは単純な直線的な関係では到底説明できません[8]。
もしニューラルネットワークが線形な活性化関数しか持たなければ、それは層をいくら重ねても、結局は入力と出力の関係を直線（または平面、超平面）でしか表現できないモデルになってしまいます[6]。このようなモデルでは、現実世界の複雑な非線形パターンを学習することは不可能です。
非線形な活性化関数を導入することによって初めて、ニューラルネットワークは曲線的な決定境界を描いたり、入力特徴間の複雑な相互作用をモデル化したりする能力、すなわち高い「表現力」を獲得します[3]。この非線形性こそが、ニューラルネットワーク、特に深層学習（ディープラーニング）が、従来の手法では困難だった様々な複雑なタスク（高精度な画像認識、自然な機械翻訳など）で目覚ましい成功を収めている根源的な理由なのです[10]。非線形性なくして、現代の高性能な機械学習モデルは成り立ちません。

損失関数の選択の重要性：正しい目標設定のために

損失関数は、モデルの学習プロセスにおける「羅針盤」であり、「評価基準」そのものです。モデルは、損失関数が示す値を最小化することを目指して学習を進めます[1]。したがって、どのような損失関数を選ぶかによって、モデルが何を「良い予測」と見なし、どのような間違いを避けようとするかが決まります。
解きたい問題の種類（例えば、数値を予測する回帰か、カテゴリを予測する分類か）、データの特性（例えば、外れ値が多く含まれるか）、さらにはビジネス上の要求（例えば、特定の種類の誤分類を特に避けたいか）などを考慮して、最も適切な損失関数を選択することが極めて重要です[22]。
もしタスクの性質に合わない損失関数を選んでしまうと、モデルは的外れな目標に向かって学習を進めることになります。その結果、訓練データに対する損失値は下がったとしても、実際の未知のデータに対する性能（汎化性能）は向上しない、あるいはむしろ悪化してしまう可能性すらあります[39]。例えば、本来はクラス間の確率的なずれを評価すべき分類問題で、単純な距離を測るMSEを使ってしまうと、モデルは望ましい性能を発揮できないでしょう。正しい目標設定、すなわち適切な損失関数の選択が、効果的な学習の第一歩です。

最適化手法の重要性：効率的かつ効果的な学習のために

損失関数が学習の目標（谷底）を定めたとしても、その谷底にたどり着くための道のりは平坦ではありません。特に、現代の深層学習モデルは何百万、何億という膨大な数のパラメータを持つことがあり、そのパラメータ空間（損失関数の地形）は極めて高次元で、無数の局所最適解や鞍点、平坦な領域を含む非常に複雑なものになっています[2]。
このような複雑な地形の中から、効率的に、かつできるだけ良い解（低い損失値を持つパラメータ）を見つけ出すためには、優れた探索戦略、すなわち高性能な最適化アルゴリズムが不可欠です。基本的な勾配降下法だけでは、計算時間の問題や局所最適解に陥るリスクなどから、大規模で複雑なモデルの学習は現実的ではありません[40]。
Adamのような洗練された最適化アルゴリズムは、学習率の自動調整や慣性の導入といった工夫により、単純な勾配降下法よりもはるかに速く、安定して、そして多くの場合より良い解に到達することを可能にします[40]。どの最適化手法を選択し、そのハイパーパラメータ（学習率など）をどのように設

定するかは、モデルの学習に必要な時間、最終的に到達できる性能のレベル、そして学習が途中で停滞してしまうリスクなどに直接的な影響を与えます [40]。効率的で強力な最適化手法の存在が、今日の深層学習の発展を支える重要な基盤となっているのです。

相互作用が汎化性能を決める

これら3つの要素は個々に重要ですが、最終的なモデルの成功、すなわち未知のデータに対する予測能力（汎化性能）は、これらの要素がどのように組み合わされ、相互作用するかによって決まります。

- 活性化関数は、モデルがどれだけ複雑なパターンを表現できるかの「潜在能力（キャパシティ）」を決定します [8]。しかし、能力が高すぎると、訓練データのノイズまで学習してしまい、未知のデータには対応できない「過学習」を引き起こす可能性があります。
- 損失関数は、モデルが学習すべき「目標」を定義し、訓練データのどの側面（例えば、平均的な誤差か、確率分布か）に焦点を当てるべきかを指示します [1]。適切な目標設定は、汎化に繋がる本質的なパターンを学習する上で重要です。
- 最適化アルゴリズムは、設定された目標（損失関数）を達成するために、パラメータ空間をどれだけ「効果的に探索」できるかを決定します [2]。効率的な探索は、訓練データに対して良い解を見つけるのに役立ちますが、それ自体が汎化を保証するわけではありません。

良い汎化性能を得るためには、モデルの表現能力（活性化関数）、学習の目標（損失関数）、そして学習プロセス（最適化アルゴリズム）の間に適切なバランスが必要です。例えば、表現能力が高すぎるモデル（活性化関数）を、訓練データの損失を徹底的に最小化するような設定（損失関数と最適化）で学習させると、過学習に陥りやすくなります。逆に、モデルの能力が低すぎたり、損失関数がタスクに合っていなかったり、最適化がうまく機能しなかったりすると、訓練データすら十分に学習できない「未学習」の状態になります。
したがって、これらの3つの要素を個別に理解するだけでなく、それらがどのように連携し、互いに影響し合いながらモデル全体の性能、特に汎化性能を形作っていくのかを理解することが、高性能な機械学習モデルを構築するための鍵となるのです。

7. まとめ

三位一体の重要性

本レポートでは、機械学習モデルの学習プロセスにおける3つの基本的な構成要素、すなわち活性化関数、損失関数、最適化アルゴリズムについて解説してきました。

- 活性化関数は、ニューラルネットワークに非線形性をもたらし、複雑なデータパターンを捉えるための「表現力」を与えます。
- 損失関数は、モデルの予測と実際の正解との間の「誤差」を測るものさしであり、学習が目指すべき「目標」を定義します。

- 最適化アルゴリズムは、損失関数によって定義された目標を達成するために、モデルのパラメータを効率的に調整する「探索戦略」を提供します。

これら3つの要素は、それぞれが独立して存在するのではなく、互いに密接に連携し合い、影響を与え合いながら、モデルの学習という複雑なプロセス全体を支える、まさに「三位一体」の存在であると言えます。

機械学習理解への第一歩

一見するとブラックボックスのように見える機械学習モデルも、その内部では、これらの基本的な構成要素が論理的なステップに従って連携し、機能しています。活性化関数がどのように信号を変換し、損失関数がどのように間違いを評価し、最適化アルゴリズムがどのようにパラメータを改善していくのか、その基本的な仕組みを理解することは、「モデルがどのように学習しているのか」という疑問に答えるための重要な第一歩となります。

本レポートが、機械学習の世界に足を踏み入れたばかりの方々にとって、その基本的な仕組みを理解し、さらに深い学習へと進むための一助となり、機械学習の持つ面白さや奥深さを感じるきっかけとなれば幸いです。

引用文献

1. Loss and Loss Functions for Training Deep Learning Neural ..., 4月 24, 2025にアクセス、https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/
2. Deep Learning Optimization Algorithms - neptune.ai, 4月 24, 2025にアクセス、https://neptune.ai/blog/deep-learning-optimization-algorithms
3. 【初心者】ネコでも分かる「活性化関数」ってなに？ - Zenn, 4月 24, 2025にアクセス、https://zenn.dev/nekoallergy/articles/4e224b57a97af9
4. Activation Functions in Neural Network - Analytics Vidhya, 4月 24, 2025にアクセス、https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/
5. 活性化関数の役割と代表的な活性化関数～ステップ ... - AVILEN, 4月 24, 2025にアクセス、https://avilen.co.jp/personal/knowledge-article/activation_function/
6. Activation functions in Neural Networks - GeeksforGeeks, 4月 24, 2025にアクセス、https://www.geeksforgeeks.org/activation-functions-neural-networks/
7. 深層学習で用いられる活性化関数の重要性と種類 - データサイエンスを勉強したいブログ, 4月 24, 2025にアクセス、https://datastudy.gonna.jp/activation-function/
8. 【ディープラーニング基礎⑦】線形モデルと非線形モデルとは ..., 4月 24, 2025にアクセス、https://zenn.dev/student_blog/articles/3032d5a0d659c0
9. 非線形関係とは何か？ - 統計を簡単に学ぶ, 4月 24, 2025にアクセス、https://ja.statisticseasily.com/%E7%94%A8%E8%AA%9E%E9%9B%86/%E9%9D%9

E%E7%B7%9A%E5%BD%A2%E9%96%A2%E4%BF%82%E3%81%A8%E3%81%AF%E4%BD%95%E3%81%8B%E3%82%92%E8%AA%AC%E6%98%8E%E3%81%97%E3%81%BE%E3%81%99/
10. 非線形モデルとは？機械学習における特性と活用例を徹底解説！ - DataStreet, 4月 24, 2025にアクセス、https://statistical.jp/nonlinear-models/
11. 機械学習初心者必見！なぜ活性化関数が必要なのか | AIワナビードットコム, 4月 24, 2025にアクセス、https://aiwannabe.com/why-activation-functions-are-needed-for-machine-learning/
12. 5 Essential Guide to Using Activation Function in Deep Learning - Number Analytics, 4月 24, 2025にアクセス、https://www.numberanalytics.com/blog/essential-guide-activation-function-deep-learning
13. Introduction to Activation Functions in Neural Networks - DataCamp, 4月 24, 2025にアクセス、https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks
14. statistical.jp, 4月 24, 2025にアクセス、https://statistical.jp/nonlinear-models/#:~:text=%E9%9D%9E%E7%B7%9A%E5%BD%A2%E3%83%A2%E3%83%87%E3%83%AB%E3%81%AF%E3%80%81%E6%A9%9F%E6%A2%B0%E5%AD%A6%E7%BF%92,%E3%81%8C%E5%8F%AF%E8%83%BD%E3%81%AB%E3%81%AA%E3%82%8A%E3%81%BE%E3%81%99%E3%80%82
15. 13. ニューラルネットワークの基礎 — ディープラーニング入門 - Chainer チュートリアル, 4月 24, 2025にアクセス、https://tutorials.chainer.org/ja/13_Basics_of_Neural_Networks.html
16. 【用語解説】ReLU（Rectified Linear Unit）とは？ - AILANDs, 4月 24, 2025にアクセス、https://dc-okinawa.com/ailands/relurectified-linear-unit/
17. ReLU関数とは？なぜ重要か・意味・メリット・シグモイド関数まで分かりやすく解説！深層学習における活性化関数の完全ガイド | Reinforz Insight, 4月 24, 2025にアクセス、https://reinforz.co.jp/bizmedia/6606/
18. 【図解】一撃でわかる活性化関数。わかりやすく、そして深く解説。 - すえつぐのNLP&LLM, 4月 24, 2025にアクセス、https://nlpillustration.tech/?p=733
19. Why Non-linear Activation Functions (C1W3L07) - YouTube, 4月 24, 2025にアクセス、https://m.youtube.com/watch?v=NkOv_k7r6no
20. 深層学習において重要なシグモイドやReLUなどの活性化関数の特徴を解説 - zero to one, 4月 24, 2025にアクセス、https://zero2one.jp/learningblog/deep-learning-activation-function/
21. 機械学習における損失関数の役割と代表的な損失関数 - AVILEN, 4月 24, 2025にアクセス、https://avilen.co.jp/personal/knowledge-article/nn-loss-function/
22. 損失関数とは - IBM, 4月 24, 2025にアクセス、https://www.ibm.com/jp-ja/think/topics/loss-function
23. www.ibm.com, 4月 24, 2025にアクセス、https://www.ibm.com/think/topics/loss-function#:~:text=Gather-,What%20is%20a%20loss%20function%3F,output%20of%20some%20loss%20function.
24. Linear regression: Loss | Machine Learning | Google for Developers, 4月 24, 2025

にアクセス、https://developers.google.com/machine-learning/crash-course/linear-regression/loss
25. What is Loss Function? | IBM, 4月 24, 2025にアクセス、https://www.ibm.com/think/topics/loss-function
26. Introduction to Loss Functions | DataRobot Blog, 4月 24, 2025にアクセス、https://www.datarobot.com/blog/introduction-to-loss-functions/
27. 今さら聞けない ディープラーニング入門 損失関数と最適化関数 - ZEPエンジニアリング, 4月 24, 2025にアクセス、https://www.zep.co.jp/tinoue/article/zmag_aiedge-da3/
28. Loss Functions in Machine Learning Explained | DataCamp, 4月 24, 2025にアクセス、https://www.datacamp.com/tutorial/loss-function-in-machine-learning
29. Understanding Loss Function Optimization in Modern Machine Learning - Number Analytics, 4月 24, 2025にアクセス、https://www.numberanalytics.com/blog/understanding-loss-function-optimization-machine-learning
30. 【初心者】機械学習では〇〇関数がたくさんありすぎる【Python】 - Zenn, 4月 24, 2025にアクセス、https://zenn.dev/nekoallergy/articles/machinelearning-func
31. 交差エントロピー誤差をわかりやすく説明してみる #機械学習 - Qiita, 4月 24, 2025にアクセス、https://qiita.com/kenta1984/items/59a9ef1788e6934fd962
32. 【エンジニア監修】機械学習のlossが下がらない原因と精度向上のための対処法 - 初心者用語解説 - Alibaba Cloud ドキュメントセンター, 4月 24, 2025にアクセス、https://www.alibabacloud.com/help/ja/cloud-migration-guide-for-beginners/latest/loss-function
33. 【機械学習の極意】損失関数の全貌を徹底解説!モデル精度向上への鍵を掴め, 4月 24, 2025にアクセス、https://ai-library.site/how-to/the-essence-of-machine-learning-a-comprehensive-guide-to-loss-functions-for-improving-model-accuracy/
34. 機械学習の初歩, 4月 24, 2025にアクセス、https://www.jwri.osaka-u.ac.jp/~tecd/technicalrepot20240112.pdf
35. 機械学習の損失関数をマスター！重要な5つの数式を初学者向けに解説 - Tech Teacher, 4月 24, 2025にアクセス、https://www.tech-teacher.jp/blog/loss-function/
36. 交差エントロピーと最小二乗法は、機械学習における異なる種類の問題で使用される二つの損失関数です。- note, 4月 24, 2025にアクセス、https://note.com/rodz/n/n4df44ba22927
37. Why does combining the sigmoid with the binary cross entropy loss work so well? - Reddit, 4月 24, 2025にアクセス、https://www.reddit.com/r/learnmachinelearning/comments/kvv66c/why_does_combining_the_sigmoid_with_the_binary/
38. 論文読みサポート: 深層学習における損失関数 - Qiita, 4月 24, 2025にアクセス、https://qiita.com/hiyoko1729/items/4351d7e5c6cfbef3ec1a
39. AI/MLにおける損失関数の説明 - Ultralytics, 4月 24, 2025にアクセス、https://www.ultralytics.com/ja/glossary/loss-function
40. 【初学者必見】機械学習の最適化アルゴリズムを簡単解説！！ - DS Media by Tech Teacher, 4月 24, 2025にアクセス、https://www.tech-teacher.jp/blog/algorithm-machine-learning/

41. AI/MLにおける最適化アルゴリズム | Ultralytics, 4月 24, 2025にアクセス、https://www.ultralytics.com/ja/glossary/optimization-algorithm
42. Understanding Optimization Algorithms in Machine Learning | Towards Data Science, 4月 24, 2025にアクセス、https://towardsdatascience.com/understanding-optimization-algorithms-in-machine-learning-edfdb4df766b/
43. Optimization Algorithms in Machine Learning | GeeksforGeeks, 4月 24, 2025にアクセス、https://www.geeksforgeeks.org/optimization-algorithms-in-machine-learning/
44. Optimizers in Deep Learning: A Detailed Guide - Analytics Vidhya, 4月 24, 2025にアクセス、https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
45. Loss functions and optimizers - Intro to Deep Learning, 4月 24, 2025にアクセス、https://kitchell.github.io/DeepLearningTutorial/7lossfunctionsoptimizers.html
46. 機械学習における学習率とは？最適化・損失関数についても解説 - TRYETING, 4月 24, 2025にアクセス、https://www.tryeting.jp/column/6842/
47. 機械学習のAdamやRMSPropなどの最適化手法まとめ：効率的なモデル構築の秘訣, 4月 24, 2025にアクセス、https://zero2one.jp/learningblog/machine-learning-optimization-methods/
48. 勾配法の仕組みをわかりやすく解説 - AVILEN, 4月 24, 2025にアクセス、https://avilen.co.jp/personal/knowledge-article/optimizer/
49. Understanding Optimization Algorithms in Machine Learning ..., 4月 24, 2025にアクセス、https://towardsdatascience.com/understanding-optimization-algorithms-in-machine-learning-edfdb4df766b
50. 【決定版】スーパーわかりやすい最適化アルゴリズム -損失関数から ..., 4月 24, 2025にアクセス、https://qiita.com/omiita/items/1735c1d048fe5f611f80
51. ニューラルネットワークの訓練, 4月 24, 2025にアクセス、https://ml4a.github.io/ml4a/jp/how_neural_networks_are_trained/
52. 深層学習における最適化アルゴリズムの種類と特徴 - Zenn, 4月 24, 2025にアクセス、https://zenn.dev/nakano_teppei/articles/339e084ba32075
53. 勾配降下法とは何か？図解でわかりやすくざっくり解説! | AIワナビードットコム, 4月 24, 2025にアクセス、https://aiwannabe.com/gradient-descent-explained/
54. 【イメージを使って解説】勾配降下法の問題点と改善を分かりやすく説明, 4月 24, 2025にアクセス、https://dx-consultant-fast-evolving.com/gradient-descent/
55. 【AI No.54】今更聞けない！Adamをサクッと解説 - 副業ブログ, 4月 24, 2025にアクセス、https://www.siteproducts.jp/ai/4744/
56. What is a Neural Network? - GeeksforGeeks, 4月 24, 2025にアクセス、https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/
57. A Beginner's Guide to Artificial Neural Networks - Wisdom Geek, 4月 24, 2025にアクセス、https://www.wisdomgeek.com/development/machine-learning/beginner-guide-to-artificial-neural-networks/
58. Few activation functions handling various problems - neural networks, 4月 24,

2025にアクセス、 https://datascience.stackexchange.com/questions/36646/few-activation-functions-handling-various-problems-neural-networks
59. 非線形回帰分析をわかりやすく解説｜データ分析スキル向上の秘訣 - Hakky Handbook, 4月 24, 2025にアクセス、 https://book.st-hakky.com/data-analysis/nonlinear-regression-analysis/
60. Tutorial 1: Optimization techniques - Neuromatch Academy: Deep Learning, 4月 24, 2025にアクセス、 https://deeplearning.neuromatch.io/tutorials/W1D5_Optimization/student/W1D5_Tutorial1.html
61. 機械学習と数理最適化の融合｜moai-lab公式 - note, 4月 24, 2025にアクセス、 https://note.com/moai_lab/n/nb28898e99919