ニューラルネットワーク (Neural Network)

基本構造

ニューラルネットワークは, 入力層, 隠れ層, 出力層の 3 つの層で構成されている. 入力層は, 特徴量を受け取ってデータをネットワークに渡す. 隠れ層は, データの変換と特徴量の抽出が行われ, モデルの精度を向上させる. 出力層は, ネットワークの最終的な予測値が出力される. 各層におけるノード (ユニット) は, 入力を重み付き和で計算される. 活性化関数を適応して, 次の層に出力される.

ユニットが 1 つの場合

1 つのユニットにおける入力と出力は以下のように計算される:

$$u = \sum_{i=1}^{n} w_i x_i + b$$
$$z = f(u)$$

- w_i: 各入力に対する重み
- x_i: 各入力データ
- b: バイアス
- u: ユニット入力
- z: ユニット出力
- $f(u)$: 活性化関数

ユニットが複数の場合

多層ニューラルネットワークについては以下のように計算される:

$$\mathbf{u}^{(\ell)} = \mathbf{W}^{(\ell-1)}\mathbf{z}^{(\ell-1)} + \mathbf{b}^{(\ell-1)}$$
$$\mathbf{z}^{(\ell)} = f_\ell(\mathbf{u}^{(\ell)})$$

- $\mathbf{W}^{(\ell)}$: ℓ 層のパラメータ行列
- $\mathbf{b}^{(\ell)}$: ℓ 層のバイアス
- $\mathbf{u}^{(\ell)}$: ℓ 層へのユニット入力
- $\mathbf{z}^{(\ell)}$: ℓ 層からのユニット出力
- f_ℓ: ℓ 層における活性化関数

順伝播

順伝播とは, ニューラルネットワークにおいて入力層から出力層への方向を順方向として入力変数を変換し, 予測値を計算する. 各ノードでの入力, パラメータ, バイアスの掛け合わせによる線形変換と, 活性化関数による非線形変換を繰り返して, 最後に出力層で結果を得る.

活性化関数

活性化関数とは, ニューラルネットワークの各ノードでの出力値を返す関数である. 活性化関数に非線形な関数を導入することで, ニューラルネットワークが複雑な関数を学習可能になる. 代表的な活性化関数として, 恒等写像, ReLU(Rectified Linear Unit), ロジスティック (シグモイド) 関数, ソフトマックス関数が挙げられる. 恒等写像は, 主に回帰問題の出力層で用いられる活性化関数で, 入力値をそのまま返す関数. ReLU は, 隠れ層でよく用いられる活性化関数で, 入力値が負のときは 0 を返し, 正のときはそのままの値を返す関数. ReLU は勾配消失問題の対策の 1 つとして挙げられる. シグモイド関数は 2 値分類の出力層で使用され, 実数の入力を $(0, 1)$ の範囲に圧縮する関数で, 出力は確率として解釈できる. 勾配消失問題が発生しやすいため, 隠れ層ではあまり使われない. ソフトマックス関数は多クラス分類の出力層で用いられる活性化関数で, 入力ベクトルを確率ベクトルに変換する. 出力ベクトルの各要素の総和は 1 となる. ベクトルのサイズが 2 の場合はシグモイド関数とみなせる.

恒等写像

回帰の出力層に用いられる.

$$u = \sum_{i=1}^{n} w_i x_i + b$$
$$f(u) = u$$

- w_i: 各入力に対する重み
- x_i: 各入力データ
- b: バイアス
- u: ユニット入力
- $f(u)$: 活性化関数

ReLU(Rectified Linear Unit)

隠れ層に用いられる.

$$u = \sum_{i=1}^{n} w_i x_i + b$$
$$f(u) = max\{0, u\}$$

ロジスティック (シグモイド) 関数

2 値分類の出力層に用いられる.

$$u = \sum_{i=1}^{n} w_i x_i + b$$
$$f(u) = \frac{1}{1 + \exp(-u)}$$

以下の性質を持つ:

- $\forall u \in \mathbb{R}, 0 < f(u) < 1$
- $\lim_{u \to \infty} f(u) = 1$
- $\lim_{u \to -\infty} f(u) = 0$
- $f(0) = 0.5$

ソフトマックス関数

多クラス分類の出力層に用いられる.

$$\mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b}$$
$$f_k(\mathbf{u}) = \frac{\exp(u_k)}{\sum_{\ell=1}^{K} \exp(u_\ell)}$$

- $\mathbf{x}$: 入力
- $\mathbf{W}$: パラメータ行列
- $\mathbf{b}$: バイアス
- $\mathbf{u}$: ユニット入力
- f_k: ベクトルの k 番目に対応する活性化関数

以下の性質を持つ:

- $\forall u \in \mathbb{R}, 0 < f_k(\mathbf{u}) < 1$
- $\sum_{k=1}^{K} f_k(\mathbf{u}) = 1$
- $K = 2$ のとき, シグモイド関数と一致する

$K = 2$ の場合, 2 つのクラスに対応する出力 u_1, u_2 に対して:

$$f_1 = \frac{\exp(u_1)}{\exp(u_1) + \exp(u_2)} = \frac{1}{1 + \exp(-(u_1 - u_2))}$$

これはシグモイド関数の形と一致する.

多層 NN における学習

ニューラルネットワークの学習は, 入力データ $\mathbf{x}_n$ とその正解ラベル $\mathbf{y}_n$ に基づいて行われる. 学習の目的は, 出力層の予測値 $\mathbf{z}^{(L)}$ が正解ラベル $\mathbf{y}_n$ に近づくように, 損失関数 $L(W)$ を最小化することである. この最適化の過程で, 各層のパラメータ $\mathbf{W}^{(\ell)}$ が更新されることを学習と呼ぶ. 損失関数はニューラルネットワークのタスクによって異なり, 主なタスクとして回帰, 2 値分類, 多クラス分類が挙げられる. 回帰では, 予測値とラベルの間の二乗誤差を最小化する損失関数が使用される. 2 値分類では, ラベルが 1 である確率を最大化するための最大尤度推定に基づいた損失関

数が使用される. 多クラス分類では, 各ラベルが正解である確率を最大化する交差エントロピー損失が使用される.

最小二乗誤差関数

$$L(W) = \frac{1}{N}\sum_{n=1}^{N} \|y_{(n)} - z_{(n)}^{(L)}\|_2^2$$

最大尤度関数

$$L(W) = \sum_{n=1}^{N} \log(1 + \exp(-y_{(n)} z_{(n)}^{(L)}))$$

交差エントロピー関数

$$L(W) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{(n)k} \log(z_{(n)k}^{(L)})$$

ニューラルネットワークの利点と課題

ニューラルネットワークの利点は, 複雑で非線形なデータ構造を学習する能力が高い. 例えば, 画像認識や音声処理などをモデル化できる. また, 入力データから自動的に特徴を抽出する能力を持っている. よって, 人が行う特徴量抽出のコストを軽減できる.

ニューラルネットワークの課題は, 訓練に多くのデータとマシンパワーが必要であること. また, 学習されたパラメータはブラックボックス的な性質があり, 解釈性の低さがある. さらに, データ量が増えると訓練データに対する精度が向上する反面, 未知のデータに対して予測精度が低下する (過学習) リスクも高まる.

過学習と対策

過学習 (Overfitting) は, ニューラルネットワークの学習において, モデルが訓練データに過度に適応してしまい, 未知のデータに対する汎化性能が低下する現象のことである. 過学習を防ぐために, いくつかの対策が提案されている. まず, 正則化 (Regularization) は, モデルの複雑さを抑えるための手法であり, 損失関数にパラメータの絶対値の上昇を抑えるようなペナルティ項を追加することでパラメータの大きさを制限する. 一般的な正則化手法として, パラメータの絶対値の和を制約する L1 正則化や, 二乗和を制約する L2 正則化が挙げられます. つぎに, ドロップアウト (Dropout) は, 学習中にランダムにユニットを非活性化する手法であり, 特定のユニットやパラメータに依存することを防ぐ. この手法は, モデルの汎化性能を向上させる効果がある. さらに, 交差検証 (Crossvalidation) は, 訓練データを複数の部分集合に分割し, これらを交互に訓練データとテストデータとして使用することで, モデルの汎化性能を検証する手法である. この手法により, モデルが特定のデータに過剰適応していないかを評価することができる. 最後に, バッチ正則化 (Batch Normalization) は, ミニバッチ単位でデータを平均 0, 標準偏差 1 に正規化する手法である. バッチ正則化より, 各ミニバッチのスケールがまとめられ, ミニバッチのランダムな選出の影響を受けづらくなる.

chatGPT による試験対策問題

ニューラルネットワークは, 人工知能や機械学習の分野で広く使用されるモデルであり, 人間の脳の神経構造を模倣したものである. その基本構成は層 (入力層, 中間層, 出力層) とノード (ニューロン) から成り立っている. 以下の問いに答えなさい.

問 1

ニューラルネットワークにおいて, 中間層 (隠れ層) が多層化されることによって, モデルの性能がどのように変化するかを説明しなさい. また, 過学習が発生するリスクとその対処法についても述べなさい.

(回答) 隠れ層が多層化することで, より複雑な特徴量の抽出が可能になり, 訓練データに対する精度が向上する. しかし, 訓練データに過度に適応して, 過学習するリスクも高まる. 対策として, 損失関数にパラメータの上昇を抑えるようなペナルティ項の追加 (L1 正則化, L2 正則化) をする. 交差検証による複数の仮の未知データに対する検証. 学習中にランダムなノードを非活性化させる (ドロップアウト) が挙げられる.

問 2

勾配消失問題 (Gradient Vanishing Problem) は, 深層ニューラルネットワークの学習において重要な課題の一つである. この問題が発生する理由と, それを解決するために提案された方法を 2 つ挙げ, それぞれについて簡潔に説明しなさい.

(回答) 勾配消失問題とは, 誤差逆伝播の際に層が深いニューラルネットワークについて勾配がほぼ 0 になり, 学習が上手くいかなくなる問題. 誤差逆伝播では, 出力から入力に向かって勾配を乗算していくので, この時に勾配の値が小さくなるような活性化関数 (シグモイド関数など) を用いていると, 勾配消失問題が発生しやすい. 対策として, 以下の 2 つが挙げられる. 1 つ目は, 活性化関数に ReLU 関数を採用する.

$$max\{u, 0\}, (u \text{ はユニット入力})$$

この関数の勾配は $u \geq 0$ のとき 1, $u < 0$ のとき 0 をとるので勾配が減少しない. 2 つ目は, ネットワーク構造を工夫する. 入力を層をまたいで直接次の層に伝えるスキップ接続 (ショートカット接続) を用いることで, 勾配が深い層まで効果的に伝播されるように設計する. スキップ接続により, 深いネットワークでも勾配が消失することなく, 安定して学習を進めることが可能になる.

問 3

以下の図は, 単純なニューラルネットワークの構造を示している. このネットワークにおいて, 順伝播 (Forward Propagation) と誤差逆伝播法 (Backpropagation) の計算手順を説明しなさい. 特に, 重みの更新に関する具体的な数式を示すこと.

(図を挿入: 入力層 3 ノード, 中間層 2 ノード, 出力層 1 ノードの単純な構造)

(回答) 順伝播

入力層から中間層への順伝播では, 各ノードで重み付き入力と活性化関数を計算し, 中間層の出力 $\mathbf{h}$ を得る. 中間層から出力層への順伝播では, $\mathbf{h}$ を基に出力層の出力 y を計算する. この y がネットワーク全体の出力となる.

入力層から中間層への伝播:
入力を $\mathbf{x} = (x_1, x_2, x_3)^\top$, 各ノードの重みを $\mathbf{w}_1, \mathbf{w}_2$, バイアスを b_1, b_2, 活性化関数を $f(x)$ とする.
ノード 1 の入力：

$$u_1 = \mathbf{w}_1^\top \mathbf{x} + b_1$$

ノード 1 の出力：

$$h_1 = f(u_1)$$

ノード 2 の入力：

$$u_2 = \mathbf{w}_2^\top \mathbf{x} + b_2$$

ノード 2 の出力：

$$h_2 = f(u_2)$$

中間層の出力をまとめて $\mathbf{h} = (h_1, h_2)^\top$ とする.
中間層から出力層への伝播:
中間層の出力 $\mathbf{h}$ を出力層に伝播させ, 出力層の重みを $\mathbf{w}_3$, バイアスを b_3 とする.
出力層の入力：

$$u_3 = \mathbf{w}_3^\top \mathbf{h} + b_3$$

出力層の出力：

$$y = f(u_3)$$

(回答) 誤差逆伝播

誤差逆伝播では, 順伝播で求めた $\hat{y}$ とラベル y を使って, 損失 $L(\mathbf{W})$ を求める. $\mathbf{W}$ はバイアスを含めたパラメータ行列. 出力層から逆方向に勾配を計算し, 各層の重みに対する勾配を求める (チェーンルールを使用). 勾配降下法よって, 損失関数を最小化するように (勾配が 0 になるように) パラメータを更新する. 勾配降下法の更新式は以下の通り. 学習率を η とする.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial L(\mathbf{W})}{\partial w_{ij}}$$

問 4

ニューラルネットワークにおいて, 活性化関数 (Activation Function) は重要な役割を果たしている. 以下の活性化関数について, それぞれの特性と使用例を述べなさい.

1. シグモイド関数
2. ReLU(Rectified Linear Unit)
3. ソフトマックス関数

(回答)
シグモイド関数:

$$f(u) = \frac{1}{1 + exp(-u)}$$

特性は, 任意の実数を $(0, 1)$ 区間に圧縮する. 出力は確率として解釈できる. $f(0) = 0.5$ となる. 勾配消失が発生しやすい問題がある. 使用例としては, 2 クラス分類のの出力層に用いられる.

ReLU(Rectified Linear Unit):

$$f(u) = max\{0, u\}$$

特性は, 入力が負の値のときは 0, 正の値の時はそのままを返す関数. 勾配消失問題の対策の 1 つとして挙げれらる. 使用例としては, 隠れ層に用いられる.

ソフトマックス関数:

$$f_k(\mathbf{u}) = \frac{exp(u_k)}{\sum_{\ell=1}^{K} exp(u_\ell)}$$

特性は, ベクトルの各要素を $(0, 1)$ 区間に圧縮する. 出力は確率のベクトルとして解釈できる. ベクトルの各要素の総和は 1 となる. $K = 2$ ときはシグモイド関数と同じ挙動になる. 使用例としては, 多クラス分類の出力層に用いられる.