6.4 アーキテクチャ設計

6.4.1 普遍近似特性と深度

線形モデルは, 行列計算によって特徴から出力にマッピングし, 定義上, 線形関数のみを表すことができる. 線形モデルを適用すると, 多くの損失関数が凸最適化問題を引き起こすため, トレーニングが容易であるという利点がある. 残念ながら, 非線形関数を学習したい場合がよくある.

一見すると, 非線形関数を学習するには, 学習したい非線形性の種類に特化したモデルファミリを設計する必要があると推測される. 幸いなことに, 隠れ層を持つフィードフォワードネットワークは, 普遍的な近似フレームワークを提供する. 具体的には, ユニバーサル近似定理 (Hornik ら, 1989 年; Cybenko, 1989 年) は, 線形出力層と任意の「圧縮」活性化関数 (ロジスティックシグモイド活性化関数など) を持つ少なくとも 1 つの隠れ層を持つフィードフォワードネットワークが, ネットワークに十分な隠れユニットが与えられていれば, 任意のボレル可測関数を 1 つの有限次元空間から別の有限次元空間に任意の非ゼロ量の誤差で近似できると述べている. フィードフォワードネットワークの導関数も, 関数の導関数を任意に適切に近似できる (Hornik ら, 1990 年). ボレル可測性の概念は, この本の範囲外である; ここでの目的には, $\mathbb{R}^n$ の閉じた有界部分集合上の任意の連続関数はボレル測定可能であり, それゆえニューラルネットワークで近似できることが言えれば十分である. ニューラルネットワークは, 任意の有限次元離散空間から別の離散空間への任意の関数マッピングを近似することもできる. 元の定理は, 非常に負の引数と非常に正の引数の両方で飽和する活性化関数を持つユニットの観点から最初に述べられたが, 現在一般的に使用される正規化線形ユニットを含む, より広いクラスの活性化関数に対しても証明されている (Leshno ら, 1993 年).

ユニバーサル近似定理とは, 学習しようとしている関数が何であれ, 大規模な MLP でその関数が表現できることがわかっていることを意味する. ただし, トレーニングアルゴリズムがその関数を学習できるという保証はない. MLP で関数を表現できたとしても, 2 つの異なる理由で学習が失敗する可能性がある. まず, トレーニングに使用される最適化アルゴリズムが, 目的の関数に対応するパラメータの値を見つけられない可能性がある. 次に, トレーニングアルゴリズムが過学習のために誤った関数を選択する可能性がある. 5.21 節で述べたように「ノーフリーランチ」定理は, 普遍的に優れた学習アルゴリズムが存在しないことを示している. フィードフォワードネットワークは, 関数が与えられた場合, その関数を近似するフィードフォワードネットワークが存在するという意味で, 関数を表現するための普遍的なシステムを提供する. 特定のサンプルのサンプルのトレーニングセットを調べ, トレーニングセットにないポイントに一般化される関数を選択するための普遍的な手順は存在しない.

ユニバーサル近似定理によれば, 必要な精度を達成できるほど大きなネットワークが存在するが, この定理ではこのネットワークがどの程度の大きさになるかは述べられていない. Barron(1993 年) は, 幅広いクラスの関数を近似するために必要な単層ネットワークのサイズについて, いくつかの制限を示している. 残念ながら, 最悪の場合, (区別する必要がある各入力構成に対応する 1 つの隠れ層を含む可能性がある) 指数関数の数の隠れユニットが必要になる可能性がある. これはバイナリの場合に最も簡単に理解できる: ベクトル $\boldsymbol{v} \in \{0,1\}^n$ 上の可能なバイナリ関数の数は 2^{2^n} であり, そのような関数の 1 つを選択するには 2^n ビットが必要であり, 一般的に $\mathcal{O}(2^n)$ の自由度が必要になる.

要約すると, 単一層のフィードフォワードネットワークはあらゆる関数を表現するのに十分であるが, 層が実現不可能なほど大きく正しく学習および一般化できない可能性がある. 多くの場合, より深いモデルを使用すると, 目的の関数を表現するために必要なユニットの数を減らし, 一般化エラーの量を減らすことができる.

ある値 d よりも大きい深さを持つアーキテクチャによって効率的に近似できる関数のファミリが存在するが, 深さが d 以下に制限されている場合は, はるかに大きなモデルが必要になる. 多くの場合, 浅いモデルに必要な隠れユニットの数は n の指数関数である. このような結果は, 機械学習に使用される連続微分可能ニューラルネットワークに似ていないモデルで最初に証明されたが, その後, これらのモデルに拡張された. 最初の結果は, 論理ゲートの回路に関するものでした (Håstad, 1986 年). その後の研究で, これらの結果は, 非負の重みを持つ線形閾値ユニットに拡張され (Håstad および Goldman, 1991 年; Hajnal ら, 1993 年), その後, 連続値の活性

化を持つネットワークに拡張された (Maass, 1992 年; Maass ら, 1994 年). 現代のニューラルネットワークの多くは, 整流線形ユニットを使用している. Leshno ら (1993 年) は, 整流線形ユニットを含む幅広い非多項式活性化関数のファミリを持つ浅いネットワークが普遍的な近似特性を持つことを実証したが, これらの結果は深さや効率の問題には触れておらず, 単に十分に広い整流ネットワークが任意の関数を表現できることを規定している. Pascanu(2013 年, b) と Montufar(2014 年) は, 深い整流器ネットで表現できる関数は, 浅い (1 つの隠れ層) ネットワークで指数関数的な数の隠れユニットを必要とする可能性があることを示した. より正確には, 彼らは, (整流器非線形性または最大出力ユニットから取得可能な) 区分線形ネットワークがネットワークの深さに対して指数関数的な数の領域を持つ関数を表現できることを示した. 図 6.5 は, 絶対値整流化を備えたネットワークが, ある隠れユニット上で計算された関数の, その隠れユニットの入力に対する鏡像を作成する方法を示している. 各隠れユニットは, 鏡像応答 (絶対値非線形性の両側) を作成するために入力空間を折りたたむ場所を指定する. これらの折りたたみ操作を組み合わせることで, あらゆる種類の規則的な (繰り返しなどの) パターンをキャプチャできる, 指数関数的に多数の区分線形領域が得られる.

図 6.5: Pascanu ら (2014 年, a) および Montufar ら (2014 年) によって正式に示された, より深い整流器ネットワークの指数関数的利点の直感的かつきか幾何学的な説明.
(左) 絶対値整流ユニットは, 入力の全てのミラーポイントのペアに対して同じ出力を持つ. ミラー対称軸は, ユニットの重みとバイアスによって定義される超平面によって与えられる. そのユニット (緑の決定面) 上で計算される関数は, その対称軸を横切るより単純なパターンの鏡像になる. (中央) 関数は, 対称軸の周りのスペースを折りたたむことによって取得できる. (右) 別の繰り返しパターンを (別の下流ユニットによって) 最初のパターンの上に折りたたむと, 別の対称性が得られる (これで 4 回繰り返され 2 つの隠れ層がある).

より正確には, Montufar ら (2014 年) の主定理は, d 入力, 深さ l, 隠れ層あたり n ユニットの深層整流器ネットワークによって切り取れられる線形領域の数が,

$$O\left(\binom{n}{d}^{d(l-1)} n^d\right), \tag{6.42}$$

と述べている. つまり, 深さ l の指数関数である. ユニットあたり k このフィルタを持つ最大出力ネットワークの場合, 線形領域の数は,

$$O\left(k^{l-1} + d\right). \tag{6.43}$$

もちろん, 機械学習 (特に AI) のアプリケーションで学習したい種類の関数が, そのような特性を共有しているという保証はない.

統計的な理由から, ディープモデルを選択する場合もある. 特定の機械学習アルゴリズムを選択する時はいつでも, アルゴリズムが学習すべき関数の種類に関する一連の事前の考え方を暗黙的に表明している. 深いモデルを選択することは, 学習したい関数にはいくつかのより単純な関数の合成が含まれるべきという非常に一般的な考え方を意味する. これは, 表現学習の観点からは, 学習の問題が, 他のより単純な変動の根本的要因で説明できる変動の根本的要因のセットを発見することであると信じていると解釈できる. 別の解釈として, 深層アーキテクチャの使用は, 学習したい関数が複数のステップで構成されるコンピュータプログラムであり, 各ステップで前のステップの出力が利用されるという考え方を表現していると解釈することができる. これらの中間出力は必ずしも変動の要因ではないが, ネットワークが内部処理を整理するために使用するカウンタやポインタに類似している可能性がある. 経験的に, 深さが増すと様々なタスクにおいて良い汎化が得られるとされる (Bengio ら, 2007 年; Erthan ら, 2009 年; Bengio, 2009 年; Mesnil ら, 2011 年; Ciresan ら, 2012 年; Krizhevsky ら, 2012 年; Sermanet ら, 2013 年; Farabet ら, 2013 年; Couprie ら, 2013 年; Kahou ら, 2013 年; Goodfellow ら, 2014

年 d; Szegedy ら, 2014 年 a). これらの実験結果の例については, 図 6.6 と図 6.7 を参照. これは, 深層アーキテクチャを使用すると, モデルが学習する関数の空間に対する有用な事前分布が実際に表現されることを示唆している.

図 6.6: 住所の写真から複数桁の数字を転記する場合, ネットワークが深いほど一般化が優れていることを示す実験結果.
Goodfellow ら (2014 年, d) のデータ. テストセットの精度は, 深さが増すにつれて一貫して向上する. モデルサイズを他の方法で増加しても, 同じ効果が得られないことを示すコントロール実験については, 図 6.7 を参照.

図 6.7: より深いモデルの方が性能が優れている傾向がある.
これは, 単にモデルが大きいからというだけではない. Goodfellow ら (2014 年, d) のこの実験は, 畳み込みネットワークの層のパラメータ数を深さを増やさずに増やしても, テストセットの性能を向上させるのにそれほど効果的でないことを示す. 凡例は, 各曲線を作成するために使用されたネットワークの深さと, 曲線が畳み込み層または完全接続層のサイズの変動を表しているかどうかを示している. この状況では, 浅いモデルは 2,000 万程度のパラメータで過剰適合するが, 深いモデルは 6,000 万を超えるパラメータで恩恵を受けることができる. これは, 深いモデルを使用することで, モデルが学習できる関数の空間に対して有用な優先順位が示されることを示唆している. 具体的には, 関数は共に構成された多くのより単純な関数で構成されるべきであるという考え方を表す. これは, より単純な表現 (例えば, エッジで定義されたコーナ) で構成される表現を学習するか, あるいは, 連続的に依存するステップ (例えば, 最初にオブジェクトのセットを見つけ, 次にそれらを互いにセグメント化し, 最後にそれらを認識する) を含むプログラムを学習することになる.

+———————————————-+
要約
線形モデルは, 特徴から出力へのマッピングを行う際に行列計算を使用し, その結果として線形関数を表現することができる. このようなモデルを用いると, 多くの場合凸最適化問題を解くことになるため, トレーニングが比較的容易になる. しかし, 現実には学習したい関数が非線形である場合も存在する.
一般的には, 非線形関数を学習するためには, その関数が持つ非線形の特性に合わせたモデルファミリを設計する必要があるとされる. ただし, 隠れ層を持つフィードフォワードネットワークは, 幅広い関数を近似するための汎用的なフレームワークを提供する. 具体的には, ユニバーサル近似定理は, 線形出力層と任意の「圧縮」活性化関数を持つ少なくとも 1 つの隠れ層を持つフィードフォワードネットワークが, ネットワークに十分な隠れユニットが与えられている場合に任意のボレル加速関数を任意の非ゼロ量の誤差で近似できることを示している. つまり, この定理によれば, 十分な大きさの MLP であれば, ほぼどんな関数でも近似できるということである.
+———————————————-+

6.4.2 その他のアーキテクチャ上の考慮事項

これまで, ニューラルネットワークは単純な層の連鎖であり, ネットワークの深さと各層の幅が主な考慮事項であると説明してきた. 実際には, ニューラルネットワークははるかに多様性を示す.
特定のタスク向けに, 多くのニューラルネットワークアーキテクチャが開発されている. 畳み込みネットワークと呼ばれるコンピュータビジョン用の特殊なアーキテクチャについては, 第 9 章で説明する. フィードフォ

ワードネットワークは, 第 10 章で説明するシーケンス処理用の再帰型ニューラルネットワークに一般化することもできる. この再帰型ニューラルネットワークには, 独自のアーキテクチャ上の考慮事項がある.

一般的には, 層同士が鎖状に接続される必要はないが, これが最も一般的な手法である. 多くのアーキテクチャでは, 主要な鎖状構造を構築し, レイヤ i からレイヤ $i+2$ またはそれ以上の層へのスキップ接続などの追加のアーキテクチャ機能を付け足す. これらのスキップ接続により, 勾配が出力層から入力に近い層へと流れやすくなる.

アーキテクチャ設計のもう 1 つの重要な考慮事項は, 層のペアをどのように接続するかである. 行列 W による線形変換で記述されるデフォルトのニューラルネットワーク層では, 全ての入力ユニットが全ての出力ユニットに接続される. 以降の章で説明する多くの特赦なネットワークでは接続が少なく, 入力層の各ユニットは出力層のユニットの小さなサブセットにのみ接続される. 接続数を減らすこれらの戦略により, パラメータの数とネットワークの評価に必要な計算量が減るが, 多くの場合問題に大きく依存する. 例えば, 第 9 章で説明する畳み込みネットワークでは, コンピュータビジョンの問題に非常に効果的な特殊な疎な接続パターンが使用される. この章では, 一般的なニューラルネットワークのアーキテクチャに関して, より具体的なアドバイスを提供することは困難である. 以降の章では, 様々なアプリケーションドメインでうまく機能することがわかっている特定のアーキテクチャ戦略について説明する.

+———————————————————-+
要約
ニューラルネットワークは単純な層の連鎖ではなく, 様々なアーキテクチャが存在する. 例えば, 特定のタスク向けに畳み込みネットワークなどの特殊なアーキテクチャが開発されている. また, 層同士を鎖状に接続する必要はなく, スキップ接続などの追加の機能が用いられることもある. これにより勾配の流れが改善される. 層の接続パターンや構造は, 特定の問題やアプリケーションに応じて調整される.
+———————————————————-+