卒業論文 2020 年度 (令和 02 年)

カーネル密度推定を用いた汎用的な活性化関数

村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・武田 合同研究プロジェクト

慶應義塾大学 環境情報学部
神保和行

卒業論文要旨 - 2020 年度 (令和 02 年度)

カーネル密度推定を用いた汎用的な活性化関数

近年機械学習では、ニューラルネットワークにおける活性化関数として、シグモイド関数や ReLU 関数などが一般的に用いられてきた。活性化関数は、その種類や問題に応じて最適な活性化関数を経験則に基づいて調整していた。一方，統計学の分野では，リンク関数が未知の場合には，カーネル関数を用いてノンパラメトリックに推定するという手法が推定されている。そこで本論文は事前に関数の形を仮定しないカーネル関数を用いた活性化関数を提案する。さらに、実際のデータセットを用いて、ニューラルネットワークの出力層を本論文で提案する手法に置き換えることにより従来の活性化関数と同等かそれ以上の精度で予測できること示した。

キーワード:
1. ディープラーニング, 2. 活性化関数, 3. ノンパラメトリック, 4. カーネル関数

慶應義塾大学 環境情報学部
神保和行

Abstract of Bachelor's Thesis - Academic Year 2020

Generic Activation Functions Using Kernel Functions

In recent years, in machine learning, sigmoid functions and ReLU functions have been commonly used as activation functions in neural networks. The optimal activation function was adjusted empirically according to the type of activation function and the problem. On the other hand, in the field of statistics, when the link function is unknown, the method of nonparametric estimation using kernel functions has been estimated. Therefore, this paper proposes an activation function using a kernel function that does not assume the form of the function in advance. Furthermore, by replacing the output layer of the neural network with the method proposed in this paper using a real data set, it is shown that the prediction accuracy is as good as or better than the conventional activation function.

Keywords :
1. Deep lerning, 2. Activation Function, 3. Non parametric, 4. Kernel Function

Keio University Faculty of Environment and Information Studies
Kazuyuki Jimbo

目 次

図 目 次

表 目 次

記号

- １次元の値は通常の字体 x を用い流。ベクトルや行列など、複数の値を内部に持っていることを強調したい場合は x や X などの太字を用いる。
- $\mathbb{R}$ は実数の集合
- $\mathrm{E}_{\mathrm{x}}[\mathrm{x}]$ は x の期待値で $\int_{\mathrm{R}} x\mathrm{P}(x)dx$ を表現する。 ここで $\mathrm{P}(x)$ は x の分布
- e は自然対数の底で、指数関数は $\exp(x) = e^x$ のように表す
- $\mathcal{G}$ はガウス分布を表現する。
- 分布 $p(x)$ からサンプルを得ることを $x \sim p(x)$ と表記します。
- データセットの集合を $\mathcal{D}$ で表現しその要素を d_i 、その部分集合は $\mathcal{D}_i$ で表現する。

第1章　序論

本章ではまず, 本研究を取り巻く社会の背景について述べる. そして本研究の解決する課題及び課題を解決する意義, 解決するための手法を提示する. 最後に本論文の構成を外観し, 序論を締める.

1.1　はじめに

背景

近年、画像認識や音声認識といった分野で機械学習の中でもディープラーニングと呼ばれる分野が急速発展し、自動運転やスマートスピーカーといったプロダクトとして人々の日常の中にも応用され始めている。機械学習はプログラミングを容易にする Pytorch や Tensorflow と呼ばれるライブラリの開発も積極的に行われており、非エンジニアにも使いやすいソニーの Neural Network Console などといったツールなども登場している。これらは機械学習におけるニューラルネットワークの構築を容易にするだけではなく、学習アルゴリズム自体も抽象化され複雑な数式を理解せずとも使用できるようになっている。ニューラルネットワークを構築する重要な要素の一つに活性化関数がある。この活性化関数には、シグモイド関数や ReLU 関数 [1] などの一般的に用いられてきた。そしてこれらは問題やデータに応じて最適な活性化関数を経験則に基づいて調整していた。また、ニューラルネットワークでは特に出力層のアウトプットを考慮した活性化関数の組み合わせを採択することが多く、例えば Resnet50 では、問題が分類ということを考慮され、中間層は全て ReLU、出力層はシグモイドなどといった組み合わせが使用されている。

一方，統計学の分野では，リンク関数が未知の場合には，ノンパラメトリックな手法に基づきモデルを推論する幾つかの手法が考案されている。カーネル密度推定を用いてノンパラメトリックに推定するという手法が Ichimura [2] によって提案されている。

課題・手法

ニューラルネットワーク構築の際の課題として、問題に応じた最適な活性化関数が未だわかっていないことが挙げられる。より良い精度を出すために適切な活性化関数を導くことができると良い。特に出力層に使う活性化関数はデータセットや問題の種類を意識する必要があり、初学者にとっての構築を難しくする。そこで本論文は事前に関数の形を指定しないカーネル密度推定を用いた活性化関数を提案する。本間研究ではカーネル密度推定とトレーニングデータのいくつかの点を用いて活性化関数の推論をするアルゴリズムを構

築する。統計学の世界で用いられていた、ノンパラメトリックなリンク関数の推論では、トレーニングデータの全てを使用する必要があったが、ディープラーニングでの応用を考え計算時間を現実的なものにするため、使用するデータ点を可変にすることが可能なアルゴリズムを提案した。

本論文では以下の課題を解決した。

- 出力層に使う活性化関数を状況に応じた適切な形に変わる汎用的な関数を導き、高い精度を出せるようにした。
- そのような関数を使うことで、データセットの形等の専門的な知識を理解しなくて済むようになり、ニューラルネットの構築を容易にした。

貢献

今後本研究で提案するカーネル密度推定を用いた活性化関数を Kernel AF と表記する。Kernel AF は実際のデータセットを用いて、ニューラルネットワークの出力層を本論文で提案する方法に置き換えることにより従来の活性化関数と同等かそれ以上の精度で予測できること示した。本研究における主な貢献を以下にまとめる.

- カーネル密度推定を用いた汎用的な活性化関数で実用的なものを完成させた。
- Kernel AF はさまざまなデータセットにおいて既存の活性化関数によりより良い精度を出すことを達成した。
- Kernel AF はいくつかのデータセットでは高い学習率でも安定した学習精度を出すことに成功した。
- Kernel AF を用いることによりデータセットに応じて出力層の活性化関数の形は従来のものではないことを示した。
- Kernel AF が勾配消失しないための条件を探求した。

1.2 本論文の構成

本論文における以降の構成は次の通りである.

2 章では，ノンパラメトリックモデル、カーネル密度推定などといった本研究へとつながる背景の解説し、これらの手法における課題を洗い出す。 3 章では，本研究におけるカーネル密度推定を用いた活性化関数についての解説を行い、提案手法の解説の詳細を述べる。 4 章では， 3 章で述べた手法の実装及び、実装における留意点を述べる。 5 章では，2 章で求められた課題に対しての評価を行い，考察する． 6 章では、実験の結果に対する考察を行い, 本研究を行う上で浮上した提案手法の限界を示し, 今後の研究方針についてまとめる

第2章　背景

本章では本研究の背景について述べる．まず機械学習における活性化関数のの役割について明確にする。また、統計学において活性化関数に相当する概念がどのように応用されてきたか述べる。活性化関数について概説し、現在の機械学習における活性化関数の抱える問題点を明らかにする。次に、活性化関数の他に、ニューラルネットワークにおける精度を向上させるいくつかの構成要素について述べる。本研究の問題点の解決に必要な、ノンパラメトリックモデルとその具体例であるカーネル密度推定を導入する。最後に、実社会において機械学習を行う上での問題点や課題を述べ、本研究が取り組むべき課題を明確にする。

2.1　活性化関数

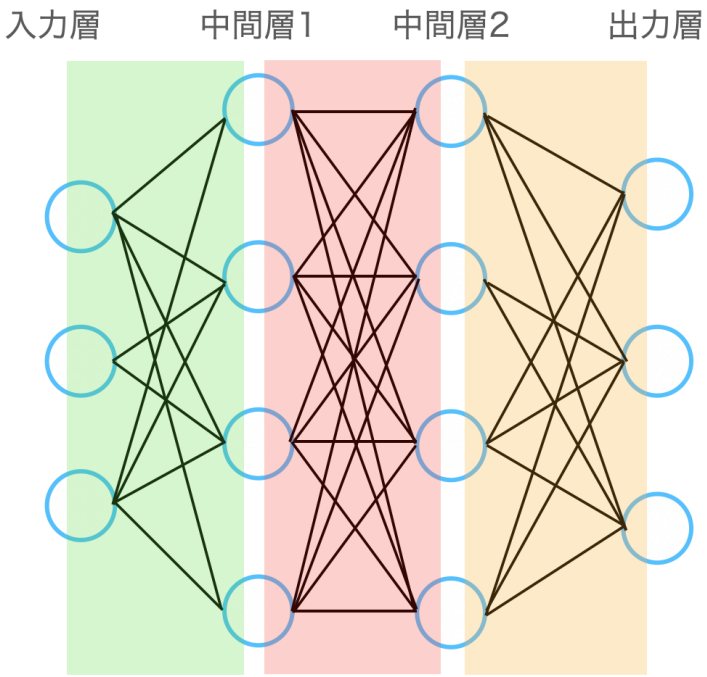

図 2.1: 活性化関数の形

また、概要を述べるにあたってニューラルネットワークの用語を定義する。活性化関数は図の黄色で囲った活性化関数を出力層の活性化関数、赤色部分を中間層の活性化関数、緑色部分を入力そうの活性化関数と呼ぶことにする。

ディープラーニングの活性化関数に関する最近の研究はまだ多く、様々な実験が 行われている。 [3] 活性化関数の歴史はシグモイドという統計学的にも馴染み深いロジスティック回帰のモデルから始まった。ニューラルネットワークの層に活性化関数を適用する過程を以下に示す。w_i を重み、x_i は入力の値、b はバイアス、z は出力、g を活性化関数とした時 $z = g(y) = g(\sum w_i x_i + b)$ このように用いられる。その後 TanH や ReLU などといったより計算に適した活性化関数が発見されてきた。特に ReLU に関しては現在のディープラーニングなどの深層ニューラルネットワークにおいても未だ応用されており、実用的にもその有用性が示されていることがわかる。

長年にわたり、性能を向上させ、ReLU の欠点に対処する多くの活性化関数が提案されてきたが、その中には Leaky ReLU [4]、ELU [5]、SELU [6] などが含まれる。Swish [7] は、$f(x) = x\mathbf{sigmoid}(\beta x)$ と定義できるが、よりロバストな活性化関数であることが証明され、ReLU と比較して結果が大幅に改善された。活性化関数はそれまで単調増加な関数が使われることが多かったが、Swish で単調増加である必要なく、汎用的に精度が向上することがわかった。またそのような活性化関数の例として Mish [8] と言うものもあげらている。

表 2.1: 活性化関数の種類

活性化関数の数式	数式
Sigmoid	$\dfrac{1}{1+\exp(x)}$
TanH	$\tanh(\mathrm{x})$
ReLU	$l = \begin{cases} 0 & \text{when } x < 0 \\ x & \text{when } x \geq 0 \text{ else} \end{cases}$
Swish	$x\mathbf{sigmoid}(\beta x)$
Mish	$x\mathbf{tanh}(\log(1+\exp(x)))$

2.1.1 勾配の消失

勾配消失問題とは、ニューラルネットワークの設計において、勾配が消失することで学習が進まなくなる技術的な問題のことである。Sigmoid などの場合、勾配が 0 に近い領域

が存在するため、勾配消失に陥ると重みがほぼ修正されなくなる。多層ニューラルネットワークでは一カ所でも勾配が 0 に近い層が存在すると、それより下層の勾配も全て 0 に近くなることが知られている。このため、層数が増えるほど学習が難しくなっていた。現在一般的に使われている ReLU は勾配消失に陥りづらいと言うところが精度向上につながっている。

2.2 統計学における位置付け

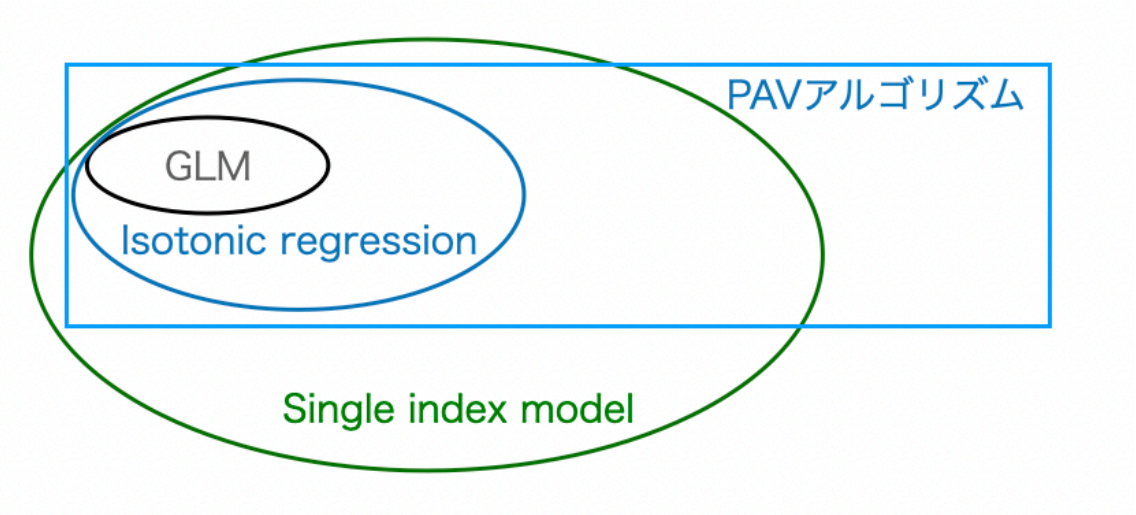

図 2.2: 機械学習と統計学の繋がり

活性化関数の概念は統計学におけるリンク関数と呼ばれるモデルの汎用性を高める動きに始まり、機械学習へと応用されている。本項目ではその理解に必要な知識を述べていく。

2.2.1 一般化線形モデルとは

説明変数を X, パラメータを W で表現し、従属変数を Y、誤差を $\epsilon \sim \mathcal{G}(0, \sigma^2)$ で表現すると、一般線形モデルは以下の式で表現することができる。

$$Y = X \cdot W + \epsilon \tag{2.1}$$

またこれは Y の期待値を使って表現すると上記は

$$\begin{aligned} \mathrm{E}[Y] &= \mathrm{E}[X \cdot W + \epsilon] && (2.2)\\ \mathrm{E}[Y] &= \mathrm{E}[X \cdot W] + \mathrm{E}[\epsilon] && (2.3)\\ \mathrm{E}[Y] &= X \cdot W && (2.4) \end{aligned}$$

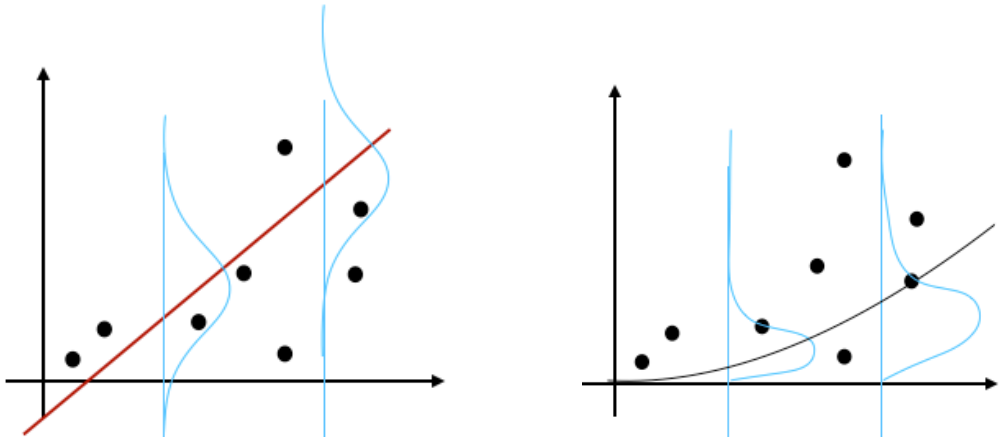

図 2.3: 一般化線形モデルの必要性

である。しかしながら、上記の式の展開では ϵ が正規分布に従うことを想定した、すなわち従属変数 Y がガウス分布に従うことを仮定したが、実際は図 2.3 のように、誤差の分布にガウス分布を仮定すると、正確さが失われることがある。そこで、従属変数をある関数 G で変換してからモデル化することでモデルの正確さが向上する。すなわち、$\mathrm{E}[Y|X] = X\dot{W}$ に対して $G(\mathrm{E}[Y|X]) = X\dot{W}$ となるような G を取り入流。またこの G の逆関数 G^{-1} をリンク関数と呼ぶ。一般線形モデルに対して、リンク関数を加えた式を以下に記す。

$$E[Y|X] = G^{-1}(X\dot{W}) \tag{2.5}$$

一般に誤差構造が決まれば、リンク関数も自動的に決まる。ガウス分布の場合のリンク関数は $G(U) = U$ である。これらの結果は G^{-1} を単調増加な任意の関数に置き換えることでさまざまなモデルを表現することが可能になる。

2.2.2 シグモイド関数とロジスティック回帰

$$G(Y) = X\dot{W} \tag{2.6}$$

とした時、

$$G = \log\Big(\frac{y}{1-y}\Big) \tag{2.7}$$

とする。一般的にこれはロジット関数と呼ばれる。これを左辺が y になるように変形すると,

$$\log\Big(\frac{y}{1-y}\Big) = z \tag{2.8}$$

$$y = \frac{1}{1+\exp(-z)} \tag{2.9}$$

右辺を z を関数にすると

$$g(z) = \frac{1}{1 + \exp(-z)} \tag{2.10}$$

となり、これはシグモイド関数である。g(z) はロジスティック関数でもあり、

$$y = \frac{1}{1 + \exp(X\dot{W})} \tag{2.11}$$

より、ロジスティック回帰であることも示される。

これらにより、ニューラルネットワークに出てくるシグモイど関数が潜在的に統計学の世界でも出てくることがわかる。

2.2.3 ノンパラメトリックモデルとカーネル密度推定

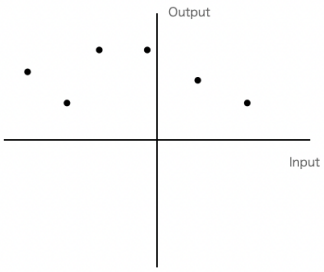

図 2.4: データ点がまばらに存在する。

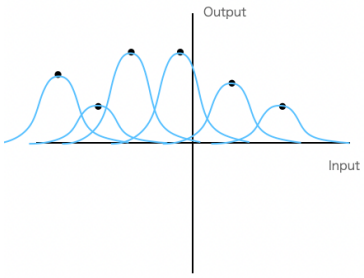

図 2.5: カーネル関数、今回はガウス関数でその周辺ごと近似する。

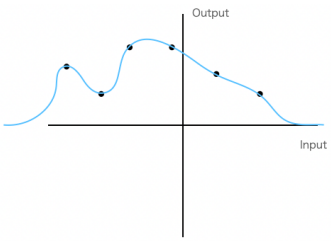

図 2.6: 点の周辺のカーネル関数を
足し合わせた時にできる関数

2.2.4 ノンパラメトリックとカーネル密度推定

統計学において、パラメータで表現されるモデルや確率分布を使用するものをパラメトリックな手法として分類するが、パラメータを使用せずモデルを表現する手法をノンパラメトリック手法という。ノンパラメトリックを代表する手法の一つにカーネル密度推定と呼ばれる手法がある。 [9] これは、ある母集団のデータが与えられたとき、カーネル関数を用いてその関数を推定する手法である。カーネル関数とは、与えられた領域内で積分した時に 1 となり、対称性を持つものとしてイメージして良い。カーネル関数の代表例としてガウス関数があげられる。

K をカーネル関数 $u \in R$ とした時、カーネル関数の定義は以下である。

- $\int_{-\inf}^{+\inf} K(u)du = 1$
- $K(-u) = K(u)$

この時、カーネル密度推定法とは、x_n をデータ、推定すべき関数を f カーネル関数を K バンド幅を h としたとき、以下の式で表現することができる。

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K\big(\frac{x - x_i}{h}\big) \tag{2.12}$$

2.4 のようにまばらに存在するデータ点の周辺に、2.5 のようにカーネル関数をおき、任意のバンド幅で足し合わせ近似していくようなものである。

2.2.5 セミパラメトリックモデルとシングルインデックスモデル

統計学の世界では、セミパラメトリックモデルというノンパラメトリックな手法とセミパラメトリックな手法を組み合わせた手法が存在する。その中の一つの代表的な手法の中にシングルインデックスモデルと呼ばれる手法が存在する。シングルインデックスモデル (SIM) とは、未知の関数 g、従属変数 Y、説明変数 X、パラメータ W、誤差項 ϵ と置いた時、以下のように表される式である。

$$Y = g(X\dot{W}) + \epsilon \tag{2.13}$$

SIM は未知の関数 g を推定しながらパラメータ W を求めていく問題に帰着されるため、ノンパラメトリックとパラメトリックが混ざった手法であるセミパラメトリックモデルとして表現される理由である。この g は、一般化線形モデルのリンク関数 G^{-1} をさらに一般化した単調増加性を無くしたモデルだと考えることができる。SIM の有名なモデルの一つに isotonic regression と呼ばれるものがある。

isotonic regression は単調増加性とう制約を仮定した SIM の一つで、化学分野や経済分野に応用されている。

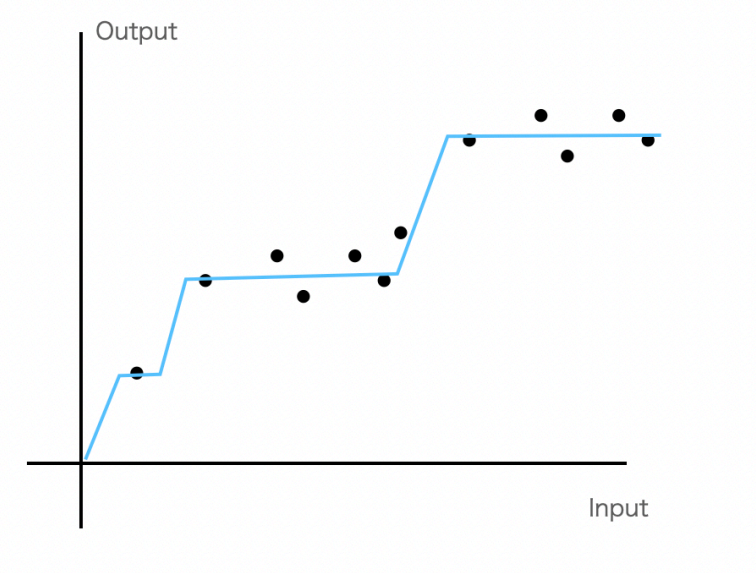

図 2.7: シングルインデックスモデルの例の一つの isotonic regression

2.2.6 セミパラメトリックモデルと機械学習学習

セミパラメトリックにモデルを推定する手法は統計学では Ichimura [2] から始まり、PAV アルゴリズムとして Adam Tauman Kalai [10] によりその有用性が確かめられた。

Ichimura の手法

SIM の未知の関数を leave one out 法を用いたカーネル密度推定法と最尤推定により導く試みは Ichimura [2] や Klein [11] によって提案され始めた。

Ichimura の手法は SingleIndexModel のノンパラメトリック関数を以下の式で近似する手法である。

$$G(X_i w) = \frac{\sum_{i \neq j} K\left(\frac{X_j w - X_i w}{h}\right) Y_j}{\sum_{i \neq j} K\left(\frac{X_j w - X_i w}{h}\right)} \tag{2.14}$$

ここで、K はカーネル関数である。$i \neq j$ とすることにより、x_i を入力した時の値が y_i へと過剰適合しないようにするためである。

PAV アルゴリズム

SIM や isotonic regression を機械学習に応用する試みは Adam Tauman Kalai [10] の PAV アルゴリズムと呼ばれる手法で、分類問題の応用へと繋がった。isotonic regression 自体は回帰問題として発明された手法であったが、これによりアルゴリズム的に分類問

題がセミパラメトリックな手法を用いて解くことが可能であることが発見された。この手法をベースに Sham Kakade [12] や Ravi Ganti [13] などによってより高速で汎用的な isotonic regression を応用したセミパラメトリックモデルの分類問題の解法のアルゴリズムが導かれた。

2.3 勾配法と学習における知識

機械学習の問題の多くは学習の際の最適なパラメータを探索する。最適なパラメータとは損失関数が最小値を撮る時の値のことである。勾配法とは関数の勾配方向に閾値を移動させることで、関数の最小値を見つける方法のことである。特にニューラルネットにおいては最小値を見つけるために、勾配法がよく用いられる。損失関数を E, w_i を i ステップ目のパラメータとした時、勾配法を数式で表すと以下のようになる。

$$w_{i+1} = w_i - \mu \frac{\partial E}{\partial w} \tag{2.15}$$

複雑な損失関数を最小化させるためのテクニカルな手法として、学習率、初期パラメータ、正則化などと言ったものが挙げられる。本項ではこれらについて必要な概念を述べる。

2.3.1 ラーニングレート（学習率）

ラーニングレートとはハイパーパラメータの一つで、式 (2.15) の μ にそうとする部分でらう。ラーニングレートは大きいほど収束の可能性は小さくなり、小さいほど学習が遅くなる。活性化関数の性能評価の一つに、大きな学習率に対しての収束性能のがあげられる。

2.3.2 重み初期値

ニューラルネットワークの学習効率は、重みの初期値によって大きく変わることが知られている。例えば初期値を全て 0 に固定すると、逆誤差伝播の影響で重みが均一になることにより、重みを多く持つ意味がなくなる。この問題を解消するために、学習のテクニックとして以下のような手法がある。

Xavier initializer

Xavier [14] によって提唱された重みの初期化手法の一つで、現在最もニューラルネットワークの学習で利用されてる手法である。ニューラルネットワークの i 層の重みの数を n_i、重みを w_i、一様分布を U で表現した時

$$w_i \sim U\big[-\frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}\big] \tag{2.16}$$

で初期化する手法である。これにより初期化すると、重みの分布が広がりを持つことが知られている。活性化関数が linear、sigmoid、tanh の時に有効に働く。

kaiming uniform

Kaiming He [15] によって提案された初期化の手法で、以下の分布から重みをサンプリングする。

$$w_i \sim U\big[-\frac{\sqrt{6}}{\sqrt{n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_{i+1}}}\big] \tag{2.17}$$

主に ReLU に有効になる。

2.3.3 レギュラライザー（正則化）

ニューラルネットワークは訓練データを過剰に学習すると未知データへの予測精度が落ちることがある。これはモデルが訓練データに対して過剰に学習したため、はずれ値やノイズまで学習してしまうことが問題であると考えることができる。このような現象を過学習というが、この過学習を防ぐためにパラメータに対して一種の罰則をかけるようなことが一般的に行われている。

損失関数を $E(w)$ とした時に、最適化する関数を $E(w)$ の代わりに以下の式をも使う。

$$E(w) + \lambda \frac{1}{p}\|w\|_p^p \tag{2.18}$$

ここで w はパラメータベクトルで　$\|\cdot\|$ は L1 ノルム (p=1) や L2 ノルム (p=2) などである。λ はハイパーパラメータである。

L1 ノルム

L1 正則化は余分なパラメータ（説明変数）を省くことを目的とした手法である。2.18 の式を考えると、モデルに必要ないパラメータを 0 にすることが損失関数の最小化につながることがわかる。この結果は、主に次元の圧縮などに対して応用することも可能である。

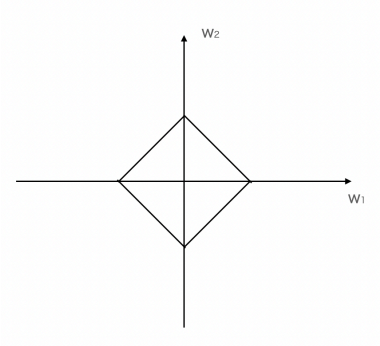

図 2.8: パラメータが二つの時の L2 ノルムのイメージ図。パラメータが二つある時、その合計値 $(\sum_i w_i)$ が 1 の点を取ると、一つのパラメータを 0 にすることが最も大きくなる。

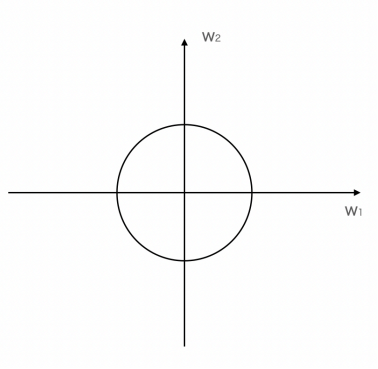

図 2.9: パラメータが二つの時の L2 ノルムのイメージ図。パラメータの合計値その合計値 $(\sum_i w_i)$ が 1 の時は $w_1 = w_2 = 0.5$ の時が最も値が小さくなるので、より高次元で考えるとこの値を大きくするのは均一的な重み分布であることが望ましい。

L2 ノルム

L2 ノルムの値を $L2$、L1 ノルムの値を $L1$、またパラメータの合計値 $(\sum_i w_i)$ が等しい場合においては

$$L2 < L1 \tag{2.19}$$

であることがわかる。これは L1 ノルム同様に一つのパラメータを 0 に対することの影響度が関数全体に対して小さいことを表していると考えられる。以上により L2 ノルムの最小化は、均一的にパラメータを小さくすることで、最小化を図ることができる。以上により L2 ノルムの罰則を足した合わせた損失関数により最小化したニューラルネットワークはパラメータ数が多いため、”表現力に優れている”と表現することが可能である。これは過学習の回避に使うことができる。

2.3.4 データセットの選択

ニューラルネットワークの最適化の際に学習データセットをどのように使うかによって性能が大きく変わることが知られている。データセットの集合を D 大きく分けて以下の三つの方法があることが知られている。

バッチ学習は安定した学習が行えるものの、の欠点は大きく以下の二つがある。

- 損失関数の形が変わらないため、最適化の手法によっては学習が停滞してしまう。

表 2.2: 実験のデータセットの名称

扱うデータ	名称	説明
$D_i \subseteq D$	ミニバッチ学習	データを部分的にランダムで取り出して学習を行う
$d_i \in D$	オンライン学習	データを一つ取り出して学習を行う
D	バッチ学習	全てのデータを用いて学習する

- Kernel AF はさまざまなデータセットにおいて既存の活性化関数によりより良い精度を出すことを達成した。

また、オンライン学習は局所界に陥りにくいというメリットがあるものの、以下欠点が存在する。

- 最初より最後のデータに過剰に適合してしまう。
- 外れ値にも反応しやすいため、パラメータの収束が不安定になる。

以上により一般的に両者の欠点を抑えたミニバッチ学習が実務では使用されることが多い。

2.3.5 optimizer

ニューラルネットワークの学習の目的は、損失関すの値をできるだけ小さくするパラメータを見つけることである。これは最適なパラメータを見つける問題であり、その問題を解くことを最適化という。

2.15 で表現した手法も最適化の一つである。これは確率的勾配効果法 (SGD) と言って単純な方法であるが、パラメータ空間を闇雲に探すよりは遥かに効率的な方法である。しかしながら、SGD 以外にもパラメータをよりよく最適化する方法は多く研究されている。

SGD

SGD は (2.15) でも記したように以下の式の形で一般的に広く知れ渡り、実装が簡単な手法として認知されている。

$$\mathrm{W} \leftarrow \mathrm{W} - \mu \frac{\partial E}{\partial \mathrm{W}} \tag{2.20}$$

$$\tag{2.21}$$

ここで、W は各パラメータ w_i をベクトルで表現したものである。

しかしながら数式からもわかるように SGD には欠点とした以下の二つが広く認知されている。

- 勾配が 0 の点では学習が進まなくなる。
- 勾配の方向が本来の最小値では ない方向を指していないことがある。

これらの理由により近年では実用的には使用されていない。

Momentum

モーメンタム (Momentum) [16] は SGD の勾配の方向が本来の最小値ではないという考えから、物理の法則を応用するような形で生まれた勾配法の一つである。

Momentum という手法は、数式で次のように表される。

$$\mathrm{v} \leftarrow \alpha \mathrm{v} - \mu \frac{\partial E}{\partial \mathrm{W}} \tag{2.22}$$

$$\mathrm{W} \leftarrow \mathrm{W} + \mathrm{v} \tag{2.23}$$

ここで新しく v という変数が登場する。これは一つ前の勾配の速度のようなものを記録しており、勾配が急なところでは大きな値になり、小さなところでは値が小さくなる。これにより SGD に比べると更新するときの”ジグザグ度合い”のようなものが軽減され、学習が安定し高速化することが知られている。

AdaGrad

ニューラルネットワークの学習では学習係数の値が重要になる。これを初めは大きく学習し、次第に小さ学習する、学習係数の減衰 (learning rate decay) という方法がよく使われる。これを発展させた方法に AdaGrad [16] というものがある。AdaGrad は以下の数式で表現できる。

$$\mathrm{h} \leftarrow \alpha \mathrm{h} + \frac{\partial E}{\partial \mathrm{W}} \odot \frac{\partial E}{\partial \mathrm{W}} \tag{2.24}$$

$$\mathrm{W} \leftarrow \mathrm{W} - \mu \frac{1}{\sqrt{\mathrm{h}}} \frac{\partial E}{\mathrm{W}} \tag{2.25}$$

ここで $\odot$ は行列の要素ごとの掛け算を意味する。パラメータ更新の際に $\frac{1}{\sqrt{\mathrm{h}}}$ を乗算することで、学習スケールを調整するという手法である。これにより、よく動いた学習パラメータは次第に小さくなる。

Adam

物理的なテクニックを応用する Momentum と学習係数を調整する AdaGrad を掛け合わせたせるのが AdamAdam [17] である。機械学習の世界では最も頻繁に用いられる。

2.4 実社会における学習の問題点

ディープラーニングを GUI で簡易的に扱えるツールは様々な企業が積極的に開発しており、50 以上のツールがあることが確認されている。しかしながらこれらのツールの共通として存在する問題点は画像分類に強いなどといった特化型となっており、それぞれの問題に応じて使い分ける必要が出てくるということである。機械学習では問題は大きく分けて回帰と分類の二つが存在するが、初学者にはそのような問題に応じたモデルの構築や学習を各ツールと状況に応じて使い分けることは非常に困難である。この問題点

2.5 汎用的な活性化関数

ReLU や Swish, Mish といった**??**であげたような活性化関数は、どれも実験的に精度が向上すると言う理由で選択したものである。そのため、あらゆるパターンにおいて最適かどうかは未知数である。Alberto Marchisio [18] が提案した手法では、このような問題を解決するため、既存の活性化関数の中から最適な活性化関数を見出し精度を向上させる方法を見出した。

しかし、この方法にも欠点があり既知の活性化関数が問題に応じた適切な活性化関数かどうか判断する方法は存在しない。Garrett Bingham [19] は進化的アルゴリズムを用いて x^2 や $sin(x)$ といった原始的関数を組み合わせ最適な活性化関数を見つけることが提唱されているが、計算量が重く、関数全体の空間を探査できるかどうかは原始的関数の組み合わせに依存してしまう。より良い活性化関数を選択して精度を向上させ、パラメータ数を減らすことは、より良いモデルを学習・発見するための重要な課題である。

2.6 本研究が取り組むべき課題

以上のような背景の中で活性化関数は、今もなお汎用的で精度が向上するような手法が模索されている。またこれまで用いてきた活性化関数が、問題に対して最適だったか確認する手立てが求められている。本研究ではそれを踏まえて以下の 2 つの課題に取り組む。

- 状況に応じた活性化関数が自動で推論でき、既存のものより安定的で精度がよくなること。
- 関数全体から活性化関数が推定でき、扱っていた活性化関数との差分を確認できるようにすること。

第3章　提案手法

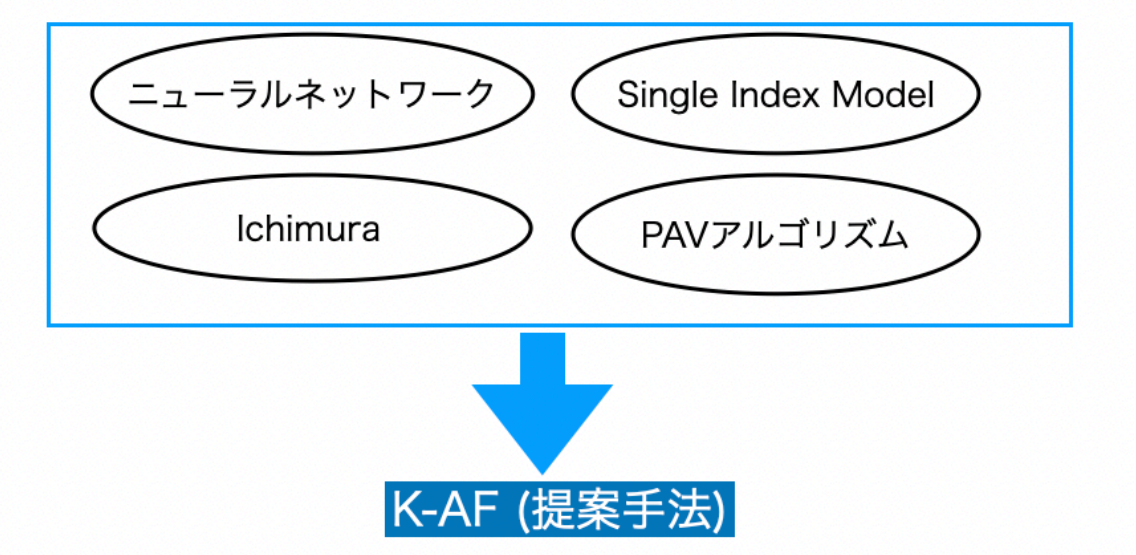

図 3.1: 提案手法

本章では背景で述べた課題を整理しつつ、それらを解決する提案手法の位置付けについて述べる。3.1 では既存の活性化関数の動向を踏まえながら本提案手法へとつながる過程を述べ流。3.2 では本研究の研究的な位置付けについて解説する。3.3 では K-AF の数式について解説し、活性化関数として使用できることを示す。**??**では K-AF のアルゴリズムについて概説し、実装の理解を深める。

3.1　活性化関数と背景

現在、深層学習に利用できる活性化関数が研究されている。特に近年では、中間層に ReLU を用い出力層に Sigmoid 関数を用いた組み合わせがよく用いられている。しかし、これらの組み合わせは経験的なものだけでなく、データに対する人間の知識が事前に必要とされる。

また、本研究で提案する活性化関数は Swish や Mish などからの単調増加性の仮定を外した点に着目した。Sigmoid はロジスティック回帰から生まれたものであり、ReLU 自体も実験的に精度がいいと導出されただけのものであった。それらの単調増加性という性質は機械学習の観点では本来必要ではないと考えることもできる。図 3.2 のように、活性化

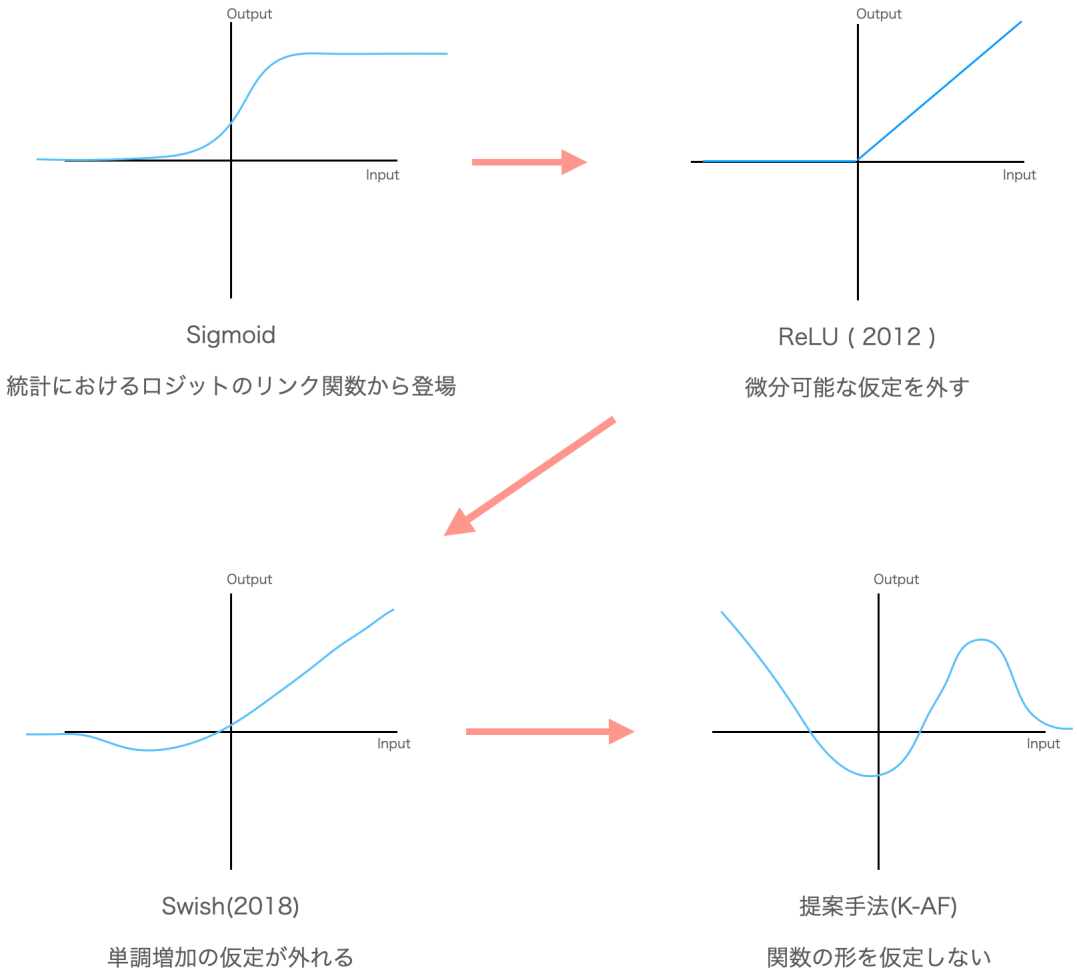

図 3.2: 活性化関数の歴史

関数は Swish や Mish などといった形へと進化する中で、単調増加性を仮定する必要がなくなった。isotonic regression 同様に単調増加性について仮定しているようであるが、本研究からはその仮定を外す。そうすることで、より精度の高い結果を導き出すことができると予想される。関数の推定はカーネル密度推定で行う。これにより活性化関数を既存の関数から選択するのではなく、関数空間全体から活性化関数を推定することができる。これにより、これまでディープラーニングで課題とされてきた活性化関数の選択の問題を解決することが可能となり、新たなアプローチが可能になることが予想される。また、これまで選択してきた活性化関数が実験的に正しいかどうか判定することも可能であると考えられる。

3.2 提案手法の位置付け

本研究の提案手法の位置付けをまとめた図を 3.1 に示した。統計学の世界で SingleIndexModel における Ichimura の手法を多次元化し、学習アルゴリズムに PAV アルゴリズ

ムを組み合わせた手法をニューラルネットワークの活性化関数に応用する。

3.3 K-AF

本論文で私が提案する活性化関数を数式 3.1 で表現する。

$$G(X_i w, X^{calc}, Y^{calc}) = \frac{\sum_{i \neq j} K\left(\frac{X_j^{calc} w - X_i w}{h_{calc}}\right) Y_j^{calc}}{\sum_{i \neq j} K\left(\frac{X_j^{calc} w - X_i w}{h_{calc}}\right)} \tag{3.1}$$

X_j^{calc} 及び Y_j^{calc} は計算用に用いるデータ点である。現在の機械学習における問題の多くは学習に用いるデータセットが非常に大きい。そのため、 [2] ではデータセットの数だけで表現していたが、一部を省略することにより少ないデータ点で表現する。少ないデータ点で記述することは、活性化関数の形の単純化と計算量を減らすことにも直結する。

詳細な数式の導出は appendixA.1 で述べた。

3.3.1 バンド幅推定

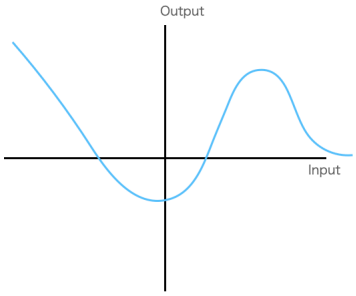

図 3.3: バンド幅が大きいと、K-AF で表現できる活性化関数の数は減る。

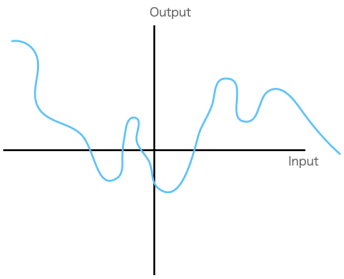

図 3.4: バンド幅が小さいと、K-AF で表現できる活性化関数の形は大きくなる。

カーネル密度推定は関数の形状を推定するにあたってバンド幅の大きさが非常に重要になる。バンド幅は大きいほど関数の自由度が減り、小さいほど自由度が大きくなることが容易に推定できる。

3.4 の方が表現の幅が広いが、勾配消失の問題や過学習の問題が考えらる。本実験ではこのバンド幅も学習のパラメータに含めることで最適なバンド幅を推定できるようにした。

3.4 アルゴリズム

3.5 al

K-AF のアルゴリズムの概要を 1 に記述した。

入力の次元を d　出力の次元を e とする。m をミニバッチのサイズ、w^t を t ステップ目の重みのパラメータとする。また、x のデータセットの集合を D_x　y のデータセットの集合を D_y、それらから n 個サンプリングすることを $Y^{calc} \sim_n D_y$ と記述する。E を目的関数とした時、以下のアルゴリズムで最適化を表現することができる。

Algorithm 1 K-AF

Input: data $\langle (x_i, y_i) \rangle_{i=1}^{m} \in \mathbb{R}^d \times \mathbb{R}^e$, $G : \mathbb{R} \to [0, 1]$.
$X^{calc} \sim_n D(X)$;
$Y^{calc} \sim_n D(Y)$;
for $t = 1, 2, \ldots$ **do**
　$g^t(x) := G(X \cdot w^t, X^{calc}, Y^{calc})$;
　$\min_w \mathrm{E}(w) = \frac{1}{2} \sum_m (y_i - g^t(x_i))^2$
end for

- データセット D_x,D_y から任意個数の X^{calc} と Y^{calc} から取り出す。
- 現状のパラメータ w, D_x, D_y を用いて、リンク関数 g^t の計算を行う。
- そのリンク関数を用いて y の値との最小二乗誤差を取り w に対して最適化を行う。

python での実装については appendixB に記述した。

第4章　実装

本章では本研究における実装環境, 提案手法の実装, 提案手法の評価に用いるデータセットについて述べる. 4.1 では本研究における実験のための実行環境及び事前知識について述べる。4.2 では K-AF の性能を既存の活性化関数と比較する実験を行う。4.6 では各データセットにおいての活性化関数の形を調査する実験をこなう。4.7 では、K-AF の性能を最も引き出す可能性がある、学習の設定を調査する。

4.1　実装環境

本研究において利用した実装環境を Table 4.1 に示す. 提案手法の実装は Pytorch 及を用いた. PyTorch, Chainer は計算グラフの自動微分ライブラリであり, 深層ニューラルネットワークの研究や開発にも用いられる. Pytorch を用いた理由は実装コストが低く研究領域に従事できるところにある。

表 4.1: 本研究の実行環境

ソフトウェア	バージョン
Python	3.6.2 or above
CPU	intel core i7
Tensorflow	2.1.0-rc0
PyTorch	6

4.2　実験 1

活性化関数の性能の比較実験のために、以下の項目を変えながら実験する。

- ラーニングレート
- 初期値、
- レギュラライザー (l1 ノルムなど)
- optimizer
- テストデータ

比較用の活性化関数には以下を用いる

- ReLU
- Sigmoid
- Linear
- TanH
- Mish
- Swish
- K-AF(本手法)

4.2.1 比較データ

他の活性化関数と適当に比較するために、以下の条件を比較して実験を行う。

表 4.2: 実験のデータセットの名称

データセット名	出力層	出力の形式	中間層の数
iris	3	分類	10
MNIST	i10	分類	10
wine	13	分類	10
住宅の価格	6	回帰	10
健康の状態	6	回帰	10
膵臓癌	6	回帰	10

表 4.3: 実験のデータセットの名称

比較実験	出力層	出力の形式	中間層の数
iris	3	分類	10
MNIST	i10	分類	10

4.2.2 実験 1

4.3 実装手法

各データセットにおける中間層は以下のパラメータで固定した。

4.4 活性化関数

4.5 実装における留意点

本研究における提案手法を実装する際に留意する必要のある点を述べる. 一つは勾配が消失してしまった場合の処理である。

4.6 実験 2

２つ目の実験では推論した活性化関数の形を観測し、既存の活性化関数との違いを定性評価を行う

4.6.1 比較データ

4.7 実験 3

K-AF における学習の難点である、勾配消失について定量評価を行う。

4.7.1 比較データ

第5章　評価

本章では，提案システムの評価に大きく二つ述べる。一つは他の活性化関数との比較。もう一つは、

5.1　実験内容

実験 1

ニューラルネット、データセット 、中間層、学習率 、勾配アルゴリズム、イニシャライザ、ステップ数 、ノーマライザーを変更していくつか実験した。

実験 3

勾配が消失する条件について実験した。

5.2　評価内容

5.2.1　既存の活性化関数との比較実験

実験 1

以下の条件で実験した。

5.3　まとめ

K-AF の精度、勾配消失の有無、実用性の観点から先行研究との比較を行った. 実験結果を踏まえ, 第 6 章で考察を行う.

表 5.1: 実験１の実行条件

設定	パラメータ
データセット	ボストンの土地の価格
入力層 の次元	10
中間層 の次元	40
出力層 の次元	13
学習率	0.000001
勾配アルゴリズム	SGD
イニシャライザ	kaiming normal
ステップ数	200
ノーマライザー	なし

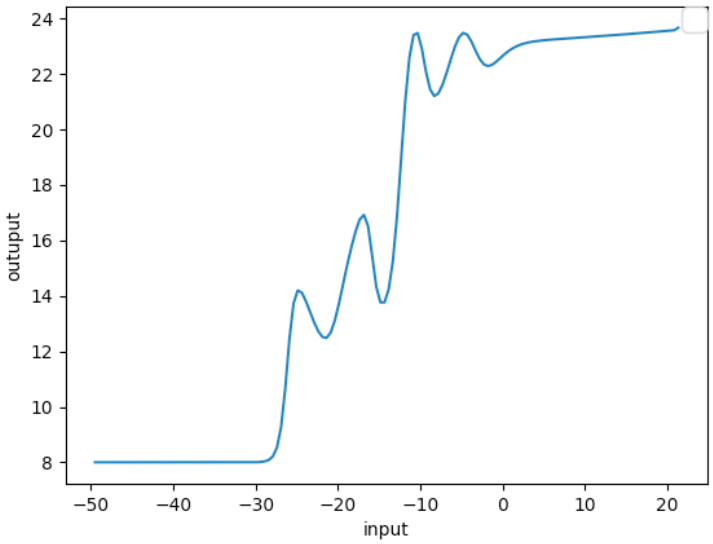

図 5.1: 活性化関数の形

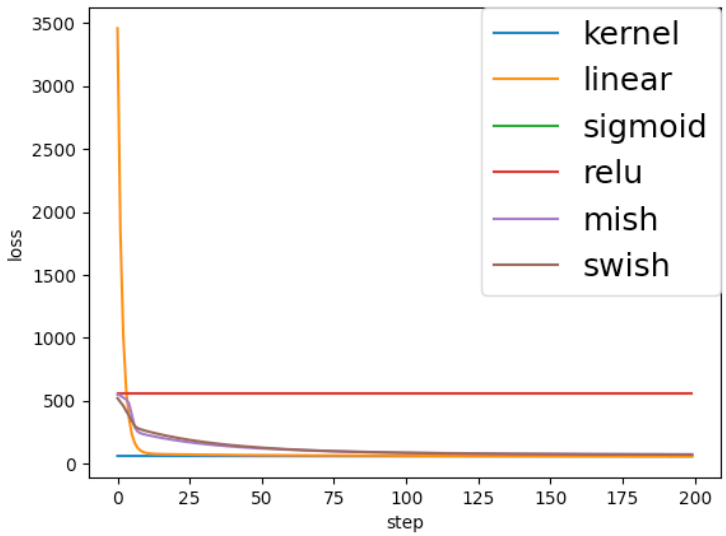

図 5.2: Loss の比較データ

表 5.2: 実験 1 の結果まとめ

活性化関数	loss
K-AF	0.0
Sigmoid	-22.0
Linear	0.0
ReLU	-22.0
Swish	0.0
Mish	-1.0

第6章　結論

本章では，実験の結果に対する考察を行い, 提案手法の利点と限界について述べ, 今後の課題, 方針を示す.

6.1　本研究のまとめ

カーネルを使った汎用的な関数でニューラルネットの最終層を置き換えることで、実際に精度の向上を図ることができた。

この実験を通して、ReLU やシグモイドと同等かそれ以上の結果が得られていることがわかります。重要な結果は、データセットの形状がわからなくても、K-AF の形状が Sigmoid に近いことである。また、決定木の実験によく使われるワインデータセットでは、式による単純な分類が難しいとされていますが、Sigmoid などよりも良い結果が得られています。また、シグモイドは学習の仕方によっては特異点にはまってしまうこともありますが カーネルはこれを回避することができました。これらの結果により、ブラックボックス化された活性化関数選択問題の解決に近づいたのではないでしょうか。

6.2　本研究の課題

スカラー値が大きなデータセットにおいては、その推論の精度が低下するだけではなく、勾配が消失して計算の継続が難しくなることがある。これらを解消するために、適切なニューラルネットの構成をより一層研究するだけではなく、それらが起こる原因を探求する必要がある。mata,

6.3　将来的な展望

本研究ではカーネル密度推定を用いて機械学習における学習精度の向上を目指した。提案手法が幅広いデータセットにおいて有益な結果をしますことを実験により明らかにし、それが実用的なデータでも応用可能であることを示した。活性化関数を汎用的に推論するという論文は未だ少なく研究分野として今後非常に注目すべきであると考えている。 ベイズ深層生成モデルの振る舞いを実験的に示した. 今後は浅いニューラルネットワークだけではなく、自動運転などの産業分野においても有用なモデルへの応用、また、形を変える汎用的な活性化関数の代表として初学者や非エンジニアが扱いやすい道具として応用

されることを望んでいる。本研究における提案手法をより効率的で使いやすいものにすることで深層学習とベイズの融合的アプローチに関する諸研究, 機械学習の応用分野に対してさらなる貢献ができることを望む

付 録A　ガウス分布とカーネル関数

A.1　カーネル活性化関数の導出

本研究で実際に使用したアルゴリズムに用いた数式実際に導出する。Ichimura [2] の手法を用いてまずは以下の式に変換する。

$$G(X_i w) = \frac{\sum_{i \neq j} K\left(\frac{X_j w - X_i w}{h}\right) Y_j}{\sum_{i \neq j} K\left(\frac{X_j w - X_i w}{h}\right)} \tag{A.1}$$

ここで、バンド幅 $K\left(\frac{X_j w - X_i w}{h}\right) \approx K\left(\frac{X_j^{calc} w - X_i w}{h_{calc}}\right)$ となるような h_{calc} を見つけることで、全てのデータ点を使わなくとも X_{calc} を用いて A.1 を近似することができる。

これにより A.1 は以下の式に直すことができる。

$$G(X_i w) \approx \frac{\sum_{i \neq j} K\left(\frac{X_j^{calc} w - X_i w}{h_{calc}}\right) Y_j^{calc}}{\sum_{i \neq j} K\left(\frac{X_j^{calc} w - X_i w}{h_{calc}}\right)} \tag{A.2}$$

付 録B　カーネル活性化関数の実装

B.1　クラス

中間層が一つの K-AF の計算を考慮した実装クラスを以下に示す。

プログラム B.1: Pytorch を用いた K-AF の計算用のクラス

```
1  class Net(nn.Module):
2
3      def __init__(self, Y, calc_Y, X, calc_X, settings):
4          super(Net, self).__init__()
5
6          self.fc1 = nn.Linear(DATA_INPUT_LENGTH, DATA_MID_LENGTH
               , bias=False)
7          self.fc2 = nn.Linear(DATA_MID_LENGTH,
               DATA_OUTPUT_LENGTH, bias=False)
8          # leave_ont_outのために事前に入力と出力をセットしておく
9          self.Y = Y
10         self.calc_Y = calc_Y
11         self.calc = False
12         # バンド幅も推定する
13         self.h = nn.Parameter(torch.tensor(1.5, requires_grad=
               True))
14         self.h_middle = torch.tensor(1.0)
15
16         self.last_layer_result = []
17         self.sigmoid = nn.Sigmoid()
18
19     # kernel推定量の計算
20     def kernel(self, Zw):
21         numerator = 0
22         denominator = 0
23         result = []
24         for j, x_j in enumerate(self.train_X):
25
26             Xw = self.fc2(F.relu(self.fc1(x_j)))
27             tmp = gauss((Xw - Zw) / self.h)
28
29             tmp[j] = 0
30             denominator += tmp
31             numerator += tmp * self.Y[j]
32
33         g = numerator/denominator
34         return g
35
36
37     def forward(self, x):
```

```
38
39          xw = F.relu(self.fc1(x))
40          xw = self.fc2(xw)
41
42          y = self.kernel(xw)
43
44          return y
```

謝辞

本論文の執筆にあたり，ご指導頂いた慶應義塾大学環境情報学部村井純博士，同学部教授中村修博士，同学部教授楠本博之博士，同学部准教授高汐一紀博士，同学部教授三次仁博士，同学部准教授植原啓介博士，同学部准教授中澤仁博士，同学部準教授 Rodney D. Van Meter III 博士，同学部教授武田圭史博士，同大学政策・メディア研究科特任准教授鈴木茂哉博士，同大学政策・メディア研究科特任准教授佐藤 雅明博士，同大学 SFC 研究所上席所員斉藤賢爾博士に感謝致します.

特に斉藤氏には重ねて感謝致します．研究活動を通して技術的視点，社会的視点等の様々な視点から私の研究に対して助言を頂き，深い思考と学びを経験させて頂くことができました．これらの経験は私の人生において人・学ぶ者として、素敵な財産として残りました．博士の指導なしには、卒業論文を執筆することは出来ませんでした.

徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・武田合同研究プロジェクトに所属している学部生，大学院生，卒業生の皆様に感謝致します．研究会に所属する多くの方々が各々の分野・研究で奮闘している姿を見て学んだことが私の研究生活をより充実したものとさせました.

異なる分野同士が触れ合い，学び合う環境に出会えたことを嬉しく感じます．また，NECO 研究グループとして多くの意見・発想・知見を与えてくださった，慶應義塾大学政策メディア・研究科 阿部涼介氏，卒業生 菅藤佑太氏，在校生 島津翔太氏，宮本眺氏，松本三月氏，梶原留衣氏，渡辺聡紀氏，木内啓介氏，後藤悠太氏，倉重健氏，九鬼嘉隆氏，内田渓太氏，山本哲平氏，吉開拓人氏，金城奈菜海氏，長田琉羽里氏，前田大輔氏に感謝致します.

皆様には，私の研究に対する多くの助言や発想を頂いただけでなく，研究活動における学びを経験させて頂きました．多くの出会いと学びの環境である SFC に感謝致します．多様な学問領域に触れ，学生同士で議論し思考することが出来ました．幸せで素敵な時間でした.

最後に，これまで私を育て，見守り，学びの機会を与えて頂いた，父 良昭氏，母 ちや子氏，兄 良行氏 に感謝致します.

参考文献

[1] Xavier. Glorot and Antoine. Bordes. Deep sparse rectifier neural networks. http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf, 2011.

[2] Hidehiko Ichimura, Wolfgang Hardle, and Peter Hall. Optimal smoothing in single index model. https://projecteuclid.org/download/pdf_1/euclid.aos/1176349020, 1993.

[3] M. N. Favorskaya and V. V. Andreev. The study of activation functions in deep learning for pedestrian detection and tracking. https://www.researchgate.net/publication/332975597_THE_STUDY_OF_ACTIVATION_FUNCTIONS_IN_DEEP_LEARNING_FOR_PEDESTRIAN_DETECTION_AND_TRACKING, 2019.

[4] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. https://arxiv.org/abs/1505.00853, 2015.

[5] Djork-Arne Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning byexponential linear units (elus). https://arxiv.org/pdf/1511.07289.pdf, 2016.

[6] Günter Klambauer, Thomas Unterthiner, and Andreas Mayr. Self-normalizing neural networks. https://arxiv.org/pdf/1706.02515.pdf, 2017.

[7] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. https://arxiv.org/pdf/1710.05941.pdf, 2017.

[8] Diganta. Misra. Deep sparse rectifier neural networks. https://arxiv.org/pdf/1908.08681.pdf, 2019.

[9] Richard O. Duda and Peter E. Hart. Pattern classification and scene analysis, 1973.

[10] Adam Tauman Kalai and Ravi Sastry. The isotron algorithm: High-dimensional isotonic regression. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.414.453&rep=rep1&type=pdf, 2008.

[11] Roger W. Klein and Richard H. Spady. An efficient semiparametric estimator for binary response models. https://www.jstor.org/stable/2951556?seq=1, 1993.

[12] Sham Kakade, Adam Tauman Kalai, Varun Kanade, and Ohad Shamir. Efficient learning of generalized linear and single index models with isotonic regression. https://arxiv.org/pdf/1104.2018.pdf, 2011.

[13] Ravi Ganti, Nikhil Rao, Rebecca M. Willett, and Robert Nowak. Learning single index models in high dimensions. https://arxiv.org/pdf/1506.08910.pdf, 2015.

[14] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. http://proceedings.mlr.press/v9/glorot10a.html, 2010.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. https://arxiv.org/abs/1502.01852, 2015.

[16] Ning Qian. On the momentum term in gradient descent learning algorithms. neural networks :the official journal of the international neural network society, 12(1):145–151, 1999. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.57.5612&rep=rep1&type=pdf, 1999.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980, 2014.

[18] Alberto Marchisio, Muhammad Abdullah Hanif, Semeen Rehman, Maurizio Martina, and Muhammad Shafique. A methodology for automatic selection of activation functions to design hybrid deep neural networks. https://arxiv.org/pdf/1811.03980.pdf, 2018.

[19] Garrett Bingham, William Macke, and Risto Miikkulainen. Evolutionary optimization of deep learning activation functions. https://arxiv.org/pdf/2002.07224.pdf, 2020.