卒業論文 2020 年度 (令和 02 年)

カーネル関数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)

カーネル関数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

- $\mathbb{R}$ は実数の集合
- E[x] は x の期待値
- 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。

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 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関数を用いた活性化関数を提案する。本間研究ではカーネル関数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関数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.

- カーネル関数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、カーネル法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法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と． 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。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法を導入する。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。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　勾配の消失

勾配消失問題とは、機械学習手法のひとつである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。

2.2　活性化関数

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脳のニューロンの数学的モデルに基づいた関数を応用したものにはじまる。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は、数学的には $z = g(y) = g(\sum wixi + b)$ となり、ここで z は活性化関数の出力 $g(y)$ である。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、$f(x) = x\mathbf{sigmoid}(\beta\mathbf{x})$ 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

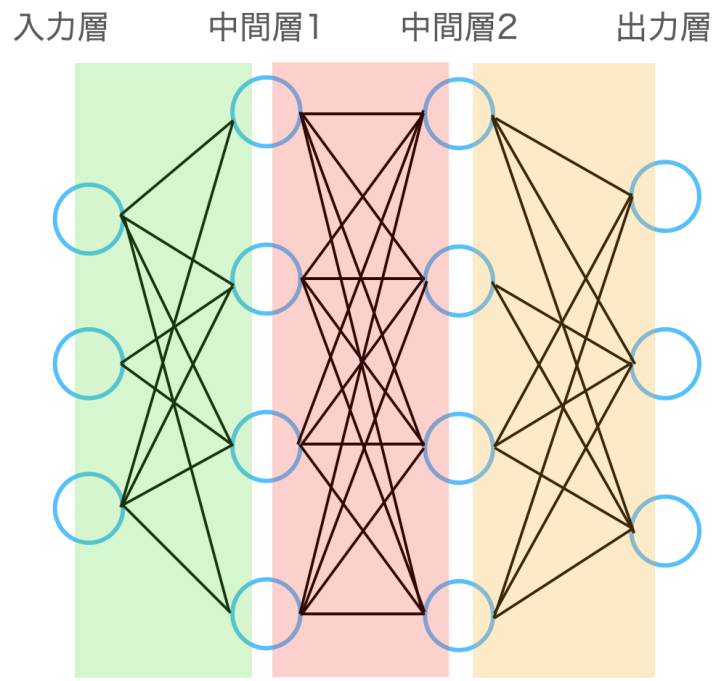

図 2.1: 活性化関数の形

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 と言うものもあげらている。 [8]

[9] [10] [3] [11] [12] [13] [14] [15] [2] [16] [7] [17] [15]

表 2.1: 活性化関数の種類

活性化関数の数式	数式
Sigmoid	sigmoid(x)
Tanh	tanh(x)
ReLU	$l = \begin{cases} 0 & \text{when x ¡ 0} \\ x & \text{when x ‘0 else} \end{cases}$
Swish	$x\mathbf{sigmoid}(\beta x)$
Mish	$x\mathbf{tanh}(softplus(x))$

2.3 汎用的な活性化関数

しかしながら、2.1 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り、あらゆるパターンにおいて最適かどうかと言う議論がされていない。また、アルバート・マルキシオ（2018）は、既存の活性化関数の中から最適な活性化関数を見出しています。しかし、その選択は、すでに知られているものから関数全体を選択している。Garrett Bingham（2020）は、内部から探索することはできなかったようです。多くのハイパーパラメータを持つ活性化関数を選択し、訓練で推定することで精度が向上します。 しかし、これも関数全体の探索には程遠い。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。

2.4 統計学における位置付け

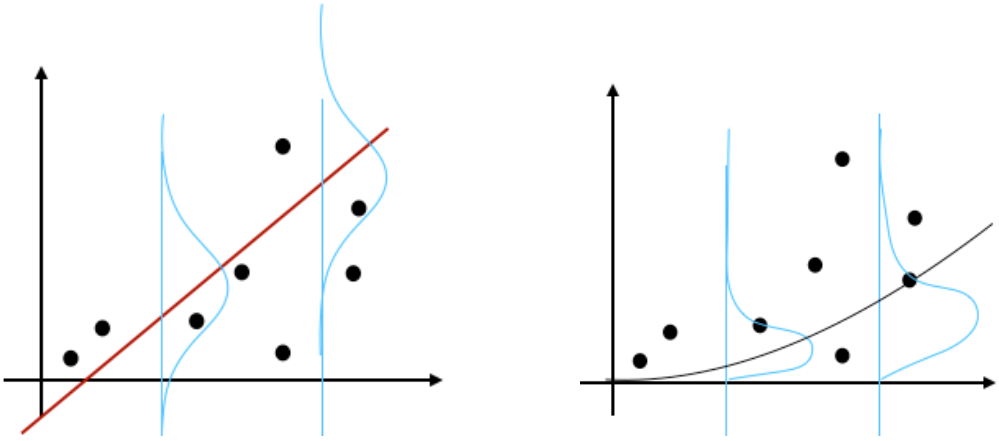

図 2.2: 活性化関数の形

一般線形化モデルとは

一方、統計学の世界に目を向けると、活性化関数と同様の概念がリンク関数 (Link Function) として一般化線形モデル (GLM) においても流用されている。説明変数を X, パラメータを w で表現し、従属変数を Y とすると、$\epsilon \sim \mathcal{G}(0, \sigma^2)$

$Y = X \cdot w + \epsilon$ 線形モデルの数式は $E[Y] = X \cdot w$ で表現することができる。しかしながら、このままでは従属変数が正規分布に従わない場合に徐々に誤差が生まれてしまう。

GLM の数式はリンク関数 G を用いて以下のように示す。

$$E[Y|X] = G^{-1}(X\dot{w}) \tag{2.1}$$

シングルインデックスモデルとは

シングルインデックスモデル (SIM) とは

GLM（Generalized Linear Model）や SIM（single index model）などの線形回帰の一般化が提案されている。

これら 2 つのモデルは、$E[Y|X] = X \cdot w$ の線形回帰モデルに、$E[Y|X] = X \cdot w$ のリンク機能を持たせたものです。統計学の世界では汎用的なリンク関数の提案として isotonic regression を応用した手法が研究されている。

一方、SIM ではリンク関数 g の推定にノンパラメトリックな手法を用いており ディープラーニングには適用されていないようです。また、SIM の経験的に推定可能な反復学習 LPAV アルゴリズムと呼ばれる L-isotron 法が Sham Kakade [13] で紹介されている。しかし、LPAV では線形相補的な位置推定法を用いてデータを並べ替えるため、ディープラーニングのような重い計算には不向きである。

isotonic regression は GLM と同様に推定する関数に単調増加性を仮定している。統計学の世界ではリンク関数自体が逆関数の考え方から導出されているため、このように単調増加性を仮定しているが、これは必要のない仮定である。

2.5 ノンパラメトリックモデルとカーネル法

カーネル法とははパターン認識において使われる手法の一つで、 判別などのアルゴリズムに組み合わせて利用するものである。

また、活性化関数に対応するリンク関数が未知の場合には、カーネル推定などのリンク関数をノンパラメトリックに推定する方法が市村（1993）によって提案されている。しかし、リンク関数をデータセット全体で推定するため、ディープラーニング的には データ数が多い場合でも有効ではない。Klein and Spady の二項選択モデル [18] も同様の手法である.

カーネル関数を用いたノンパラメトリックな方法で提案されているが、この方法での深層学習については研究されていない。

2.6 学習におけるいくつかの知識

機械学習において精度を向上させる方法は大きく分けると

2.6.1 ラーニングレート

学習率はニューラルネットにおけるハイパーパラメータの一つ。入念に調整する必要がある。

2.6.2 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。初期値は 0 や 1 で初期化する方法以外にさまざまな方法がある。

Uniform

Uniform：-1.0〜1.0 の一様乱数で初期化します

orthogonal

sparse

sparse は [15] で紹介されている。2 次元入力テンソルを疎な行列として充填し，ここで非ゼロ要素は，正規分布で埋める。

kaiming uniform

ガウス乱数に Kaiming He 提案の係数をかけて初期化する。

2.6.3 レギュラライザー

ニューラルネットワークは学習を行う際に、初期値やデータによっては重みに偏りが生まれ、勾配消失につながる可能性が高くなる。それらを解消するために、損失関数に数学的な項を加えることで学習の際の偏りを解消する手法が、実用的にも用いられている。

L2 ノルム

ippanntek

L1 ノルム

2.6.4 optimizer

この辺について

2.7 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として株式会社ソニーの Neural Network Console がある。しかしながらそれでも共通として存在する問題点はデータセットの整形はこちら側でやる必要があるということである。データ整形のプログラムは非技術者には難易度が高く機械学習を導入するにあたって、データセットに応じたさまざまなメタ的要素の取捨選択が必要となり、

第3章　提案手法

本章では提案手法について述べる.

3.1　概要

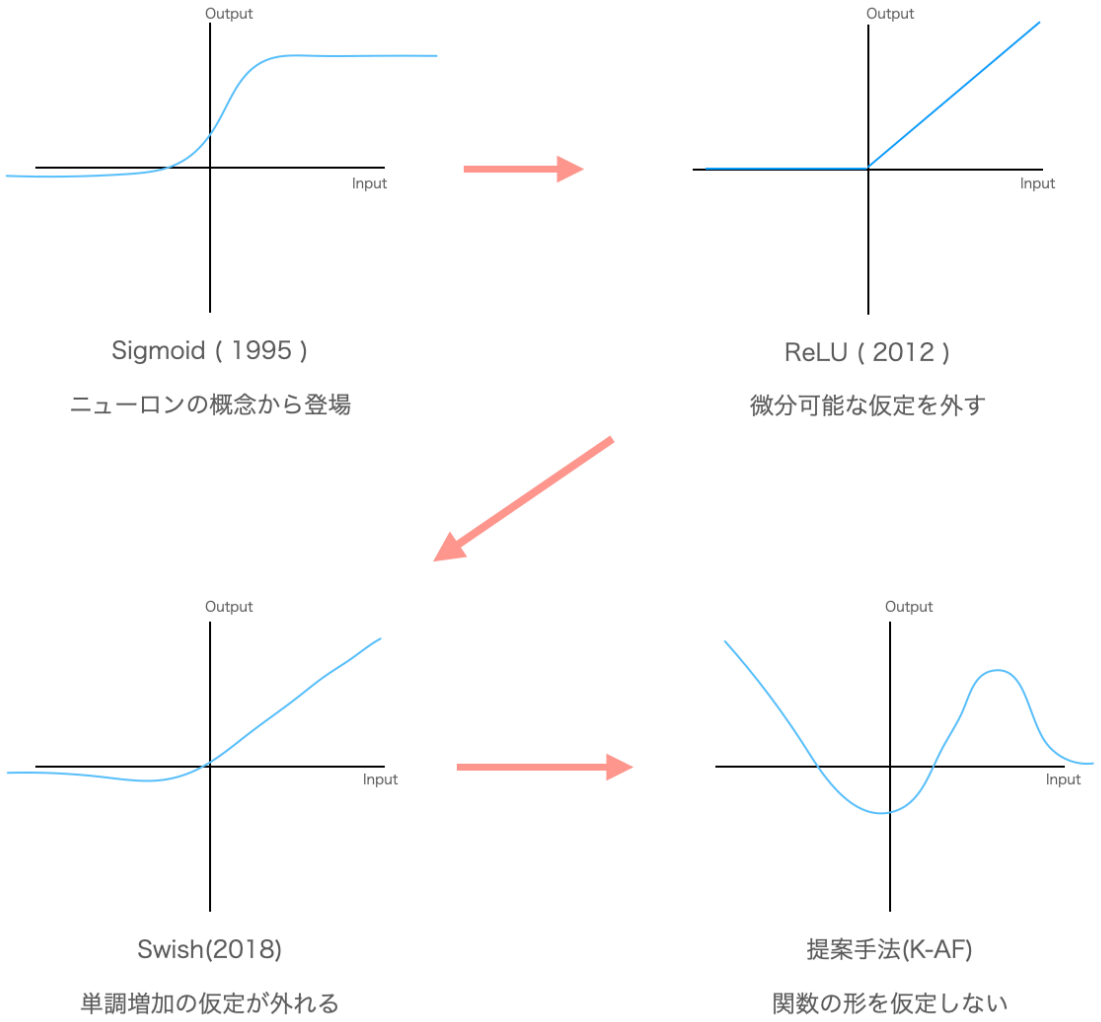

図 3.1: 活性化関数の歴史

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。最近では、中間層に ReLU

を用い、最終的な出力層にシグモイド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。ReLUやsigmoidといった関数はあくまでリンク関数の概念に相当していたこと考えると、単調増加性というのは機械学習の観点では本来必要であったものかどうか問われるものであった。統計学のSIMの法でも同様に単調増加性について仮定しているようであるが、ほんけんきゅうからはそれを外してやる。

そのため本研究ではSIMの考え方の中の単調増加性の過程を外し、本研究では、統計学の世界で使われているSIMのノンパラメトリック法を用いて、活性化関数に我々のリンク関数推定法を適用した。

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。関数の形式はカーネル関数であり、入力の出力を一つの式で表現することができる。これにより、深層学習に利用できる程度に計算コストを削減することができる。また、今回の実験では、活性化関数を既存の関数から選択するのではなく、状況に応じた関数、つまり活性化関数の形を作ることができる。そうすることで、関数全体から逆算して最適な関数を見つけることができる。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。

3.2 ノンパラメトリック

現状ではディープラーニングに活かせるようなノンパラメトリックに推定する活性化関数は研究されておらず、経験的に中間層ではRelu、最終的なアウトプット層ではデータセットに合わせてSigmoidが使われることが多い。しかしながらこれらの組み合わせは経験的であるだけではなく、データに対する人知見が事前に必要である。本研究では、統計の世界で使われていたSIMでのノンパラメトリックな手法を用いて行われていたリンク関数の推定方法を活性化関数に応用する。そうすることにより、経験的な知見による活性化関数の選択という行為を行わずともより高い精度の結果を導けるのではないかということである。関数の形式はカーネル関数を用いることで、入力に対しての出力を一つの式で表せるようにする。そうすることでディープラーニングでも使えるぐらいの少ない計算コストが実現できる。

また、この実験により状況に応じた適切か活性化関数の形を既存のものから選択するのではなく、関数全体の中から逆算できると考えている。それにより、ディープラーニングの課題であった、活性化関数の選択問題という課題も新しいアプローチで解決できると考えている。

3.3 活性化関数

本節では既存の活性化関数の問題点を具体的な事例を交えて考える。

3.4 kernel 活性関数

本論文で私が提案する活性化関数を以下の数式で表現する。

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

[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変数で表現することに成功した。

3.5 アルゴリズム

Algorithm 1 K-AF-tron

Input: data $\langle (x_i, y_i) \rangle_{i=1}^m \in \mathbb{R}^d \times [0,1]$, $u : \mathbb{R} \to [0,1]$.
$w^1 := 0$;
for $t = 1, 2, \ldots$ **do**
  $h^t(x) := u(w^t \cdot x)$;
  $w^{t+1} := w^t + \frac{1}{m} \sum_{i=1}^{m} (y_i - u(w^t \cdot x_i)) x_i$;
end for

これが実装の大まかなアルゴリズムである。詳細については appendix で述べた。

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. その後実験に用いたハイパーパラメータ, 実装における留意点について解説を行う。

今回は大きく３つの比較実験を行う。

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　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。

- 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

既存のものと比較している

ラーニングレート

初期値

初期値は

レギュラライザー

レギュラライザーは L1 ノルム, L2 ノルム等の比較ができる。

optimizer

テストデータ

各テストデータの特徴を以下の表にまとめる。データセットには出力の形式が存在し、多くの問題は「分類」もしくは「回帰」の二つに分類される。

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
iris	3	分類
MNIST	i10	分類
wine	13	分類
住宅の価格	6	回帰
健康の状態	6	回帰
膵臓癌	6	回帰

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

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

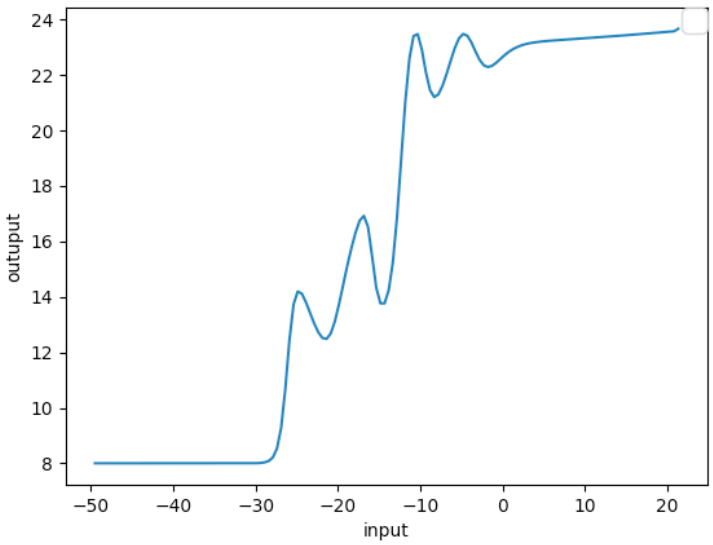

図 5.1: 活性化関数の形

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

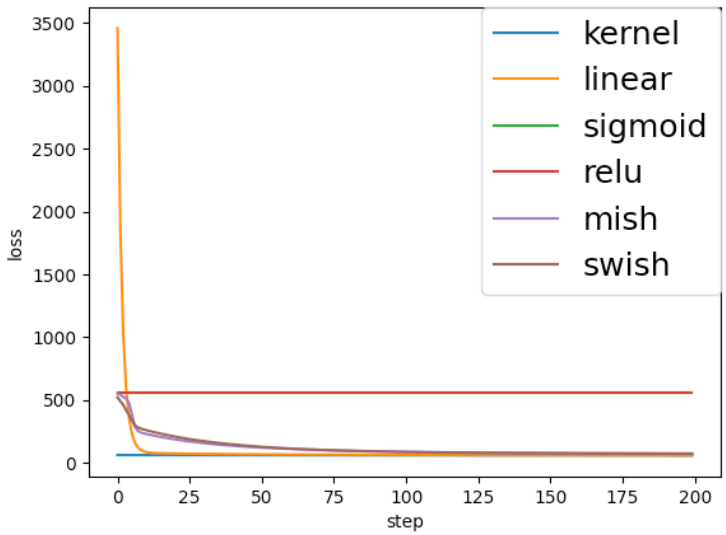

図 5.2: Loss の比較データ

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法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

以下の数式を出す

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

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 kernel activation function 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: hoge

```
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] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. https://arxiv.org/pdf/1811.03378.pdf, 2018.

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

[11] Garrett Bingham and Risto Miikkulainen. Discovering parametric activation functions. https://arxiv.org/pdf/2006.03179.pdf, 2020.

[12] 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.

[13] 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.

[14] 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.

[15] Zeljko Kereta, Timo Klock, and Valeriya Naumova. Nonlinear generalization of the monotone single index model. https://arxiv.org/pdf/1902.09024.pdf, 2019.

[16] Hanxiao Liu, Andrew Brock, Karen Simonyan, and Quoc V. Le. Evolving normalization-activation layers. https://arxiv.org/pdf/2004.02967.pdf, 2020.

[17] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. https://arxiv.org/abs/1502.01852, 2019.

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