

損保アクチュアリー業務におけるニューラルネットワークの活用 ＜ASTIN 関連研究会＞

大同火災 川上 良一 君

あいおいニッセイ同和損害 渡辺 重男 君

【司会】 時間になりましたので、セッション C の 4、ASTIN 関連研究会による「損保アクチュアリー業務におけるニューラルネットワークの活用」を開始します。発表者は、大同火災の川上良一さん、あいおいニッセイ同和の渡辺重男さんの 2 名で、お二人ともリモートから発表します。

なお、質疑応答に関しては、Slido に投稿された質問に対して、各発表者が、それぞれの発表直後に回答することとしています。質問のある方は、質問したい事項の発表者の発表中に質問を投稿するようお願いいたします。また、時間の都合などで、回答する質問を選ばせていただく場合もありますので、あらかじめご了承ください。

それでは、川上さん、よろしく申し上げます。

2021年度 日本アクチュアリー会年次大会

損保アクチュアリー業務における ニューラルネットワークの活用

2021年11月5日

ASTIN関連研究会

大同火災海上保険 川上 良一
あいおいニッセイ同和損害保険 渡辺 重男

1

【川上】 はい。大同火災の川上と申します。

本日は、ASTIN 関連研究会で取り組みを行ったニューラルネットの研究について「損保アクチュアリー業務におけるニューラルネットワークの活用」というテーマで発表させていただきます。どうぞよろしくお願いいたします。

発表の流れと担当

1. はじめに
2. NNによるクレーム頻度モデリングの概要
3. NNのモデルの詳細

4. CANN
5. CANNによる備金見積もり
6. まとめ

担当

大同火災海上保険
川上 良一

内容

スイスアクチュアリー会のADSチュートリアルを紹介し、NNを用いた自動車保険の頻度モデルについて説明する

担当

あいおいニッセイ同和損害保険
渡辺 重男

内容

NNを用いてGLMを改善するCANNの説明を行い、備金見積もりへの応用例について紹介する

2

まず発表の流れですが、前半につきましては、私の方から、ニューラルネットによるクレーム頻度モデリングの概要について、基本的なところから説明させていただきます。後半は、あいおいニッセイ同和損保の渡辺さんから、ニューラルネットを用いてGLMを改善するCANNについて説明いただき、更に、支払備金見積もりへの応用について紹介いたします。

1. はじめに

3

本セッションの概要

- 概要
 - スイスアクチュアリー会の Actuarial Data Science Tutorials やその他海外文献に基づき、自動車保険のクレーム頻度やIBNR見積もり等、損害保険分野におけるニューラルネットワークの活用について紹介するとともに、ニューラルネットワークを用いてGLMの予測精度を改善するアプローチについても紹介する。
- 留意事項
 - 本発表では各種論文等にならない、NNをあてはめて事故頻度をモデリングやIBNR算出等をRおよびkeras (tensorflow) パッケージを用いて行っているが、Rやパッケージのバージョンの違い等により異なった結果になっている箇所もある。

4

本セッションの概要ですが、スイスアクチュアリー会の Actuarial Data Science Tutorials や、その他海外文献に基づき、自動車保険のクレーム頻度や、IBNR 見積もり等、損害保険分野におけるニューラルネットワークの活用について紹介し、ニューラルネットを用いて GLM の予測精度を改善するアプローチについても紹介いたします。

留意事項ですが、本発表については、R や keras パッケージを用いてモデリングを行っております。環境等によって多少結果が変わることが想定されます。

機械学習とNN① SOA

- Emerging Data Analytics Techniques with Actuarial Applications (2019)
 - Emergingなデータ分析についてのSOAのレポート
 - 次の手法について紹介

教師あり学習

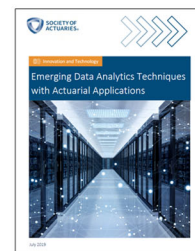
回帰とGLM、ツリー系、**NN**

教師なし学習

主成分分析、クラスター分析、遺伝的アルゴリズム、**NN**

その他

MCMC、ベイズ分析



<https://www.soa.org/resources/research-reports/2019/emerging-analytics-techniques-applications/>

5

まず、機械学習とニューラルネットの状況について、簡単に紹介させていただきます。本日の年次大会でも様々な機械学習手法が紹介されておりましたが、SOA のレポートの方でも、様々な機械学習手法が紹介されています。2019年のレポート『Emerging Data Analytics Techniques with Actuarial Applications』では、「教師あり」、「教師なし」、「その他」の分

野でのエマージングな分析手法を紹介しており、ニューラルネットが、「教師あり」と「教師なし」で紹介されています。

機械学習とNN① SOA

Section 3: Data Analytics Techniques

- 教師あり学習

Schelldorfer and Wüthrich (2019)	<ul style="list-style-type: none"> • カテゴリカルデータを扱うためNNの埋込層の使用について議論 • NNは他の教師あり手法よりも優れた性能を発揮することがある
Mendes, et al. (2017)	<ul style="list-style-type: none"> • 人工クレームデータを用いたモデル性能の研究において、NNは予測誤差が最も小さい手法の1つであることを発見
Diana, et al. (2019)	<ul style="list-style-type: none"> • クレーム予測の研究で単純なNNを調査したところ他の手法よりも性能が劣ることを発見
- 教師なし学習

Hainaut (2018)	<ul style="list-style-type: none"> • 説明変数をクラスター化し、共変量間の依存性を検出するために自己組織化マップの開発に教師なしNNを採用 • 約65,000件のオートバイ保険のデータを所有者年齢・車齢でセグメント化しクレーム頻度を説明変数に回帰させた
----------------	--

Section 4: Emerging Data Analytic Technologies

Harej, et al. (2017)	<ul style="list-style-type: none"> • 人工データを用い、個々のクレームレベルのカスケード型NNを利用した備金見積法をCL法と比較 • クレーム構造が変化する場合、NNは個々のクレームのLDFと全体的な支払備金見積の両方において、CL法よりも優れたパフォーマンスを示した
----------------------	--

6

レポートについて全部紹介することはできませんが、教師あり学習の1点めについては、ニューラルネットの埋込層というものに着目したものになっておりまして、後ほど紹介させていただきます。ニューラルネットについては、回帰、クラスター化など、幅広く用いられているという実績がございます。

機械学習とNN② 機械学習研究動向

- P&C分野のプライシングおよびリザービングの機械学習の活用について、2015年以降2020年8月までの77件の論文を調査した結果
- NNが15件（プライシング6件、リザービング9件）と一番多い

Year	Overview	Pricing	Reserving
2000-2014	3	6	4
2015	2	0	0
2016	2	1	1
2017	1	3	1
2018	5	6	3
2019	8	8	7
2020 (aug)	5	5	7

Source	Pricing	Reserving
arXiv	3	2
ASTIN	1	3
CAS	3	3
EAJ	3	1
IBIE	3	0
NAAJ	3	0
Risks	3	4
SAJ	2	3
SSRN	1	2
Other	13	4

Model	Pricing	Reserving
Neural networks	6	9
CART	3	7
Boosting	8	1
GAM/GAMLSS	6	2
Unsupervised	3	1
SVM	2	1

MDPI *Risks*, Volume 9 (2021), Machine Learning in P&C Insurance A Review for Pricing and Reserving
<https://www.mdpi.com/2227-9091/9/1/4>


7

続いて、機械学習の研究動向として、P&C分野のプライシング、リザービングについて、2015年以降2020年8月までの論文を調査した『Risks』という雑誌を紹介させていただきます。一番下赤で囲っているところですが、こちらがニューラルネットを表しておりまして、プライシングで6件、リザービングで9件、合計15件と、他の機械学習手法である決定木系の手法、

20-4

ブースティング、その他の手法に比べて発表数が多いという結果になっております。

機械学習とNN③ SAA



Actuarial Data Science
An initiative of the Swiss Association of Actuaries

■ SAA (Swiss Association of Actuaries)

■ ADS (Actuarial Data Science)

- ADSは、保険数理 (AS) とデータサイエンス (DS) の共通部分
- ADSの主な目的は、SAAの“Data Science” WGの研究と結果を関心のある人々が簡単に利用できるようにすること
- チュートリアル、レクチャーノート等多数のコンテンツを提供

■ ADS Tutorials

- アクチュアリー向けでDSのさまざまな方法を徹底的かつ簡単に紹介
- 理論と統計モデルについて説明し、コードを提供することで実行した分析を簡単に独自データでテストできる
- 次のパートで、ADS Tutorialに沿ってNNによる分析手法について説明

<https://www.actuarialdatascience.org/ADS-Tutorials/>

8

続いて、スイスアクチュアリー会の ADS Tutorials について説明いたします。ADS は、アクチュアルサイエンスとデータサイエンスの共通部分として定義されています。ADS Tutorials は、アクチュアリー向けのデータサイエンスの手法について、理論と統計モデルについて説明し、また R や Python のコードを提供し、自ら学んで生かすことができる仕組みを提供しています。

ADS Tutorials①

タイトル	概要	主なモデリング手法
<p>case study 01: French Motor Third-Party Liability Claims</p>	<p>< freMTPL2freqデータ ></p> <ul style="list-style-type: none"> ➢ フランスの自動車保険データを用い、クレーム頻度についての各モデリング手法を解説 ➢ 各モデリング手法の入門的な内容であり、予測精度の比較を行う 	<ul style="list-style-type: none"> ➢ GLM ➢ 決定木 ➢ ブースティングマシン ➢ NN
<p>case study 02: Insights from Inside Neural Networks</p>	<p>< freMTPL2freqデータ ></p> <ul style="list-style-type: none"> ➢ case study01のNNについて、より踏み込んでモデルを解説 ➢ NNのハイパーパラメータ設定方法や、モデルの改善手法について考察 	<ul style="list-style-type: none"> ➢ NN
<p>case study 03: Nesting Classical Actuarial Models into Neural Networks</p>	<p>< freMTPL2freqデータ ></p> <ul style="list-style-type: none"> ➢ NNにおける埋込層の活用 ➢ NNを用いたGLMの改善 (CANN) 	<ul style="list-style-type: none"> ➢ NN (embedding) ➢ CANN
<p>case study 04: On Boosting: Theory and Applications</p>	<ul style="list-style-type: none"> ➢ 一般的に用いられることの多い2つのブースティング手法を解説 ➢ 自動車保険金請求の予測にブースティングを用いた手法を紹介 	<ul style="list-style-type: none"> ➢ AdaBoost ➢ XGBoost
<p>case study 05: Unsupervised Learning: What is a Sports Car?</p>	<ul style="list-style-type: none"> ➢ 自動車のデータの分類を通して、各教師なし学習を解説 ➢ 教師なし学習はデータの次元を削減することを目的とした手法であり、類似の特徴を持つケースをクラスター化し、高次元のデータをグラフィカルに示す 	<ul style="list-style-type: none"> ➢ PCA ➢ Bottleneck neural network ➢ K-means clustering

9

Tutorials の case study の紹介です。case study は全部で 10 個あり、右側に主なモデリング手法を記載していますが、ニューラルネット関係が多いことが特徴となっています。case study 01 では、ニューラルネットを含む様々な手法で自動車保険の頻度分析を行っています。本日は、前半は主に case study 02 の「Insights form Inside Neural Networks」と、一部、

case study 03 に基づいております。これらは一貫して同じデータに基づいて分析が行われております。

ADS Tutorials②		
タイトル	概要	主なモデリング手法
case study 06: Lee and Carter go Machine Learning: Recurrent Neural Networks (RNN)	<ul style="list-style-type: none"> スイスの男女別のデータを用い、リカレントニューラルネットワーク (RNN: 右記2手法) で死亡率をモデル化 	<ul style="list-style-type: none"> Long Short-Term Memory Gated Recurrent Unit
case study 07: The Art of Natural Language Processing	<ul style="list-style-type: none"> テキストデータに自然言語処理 (NLP) を実施し、機械学習を使用してドキュメントの分類を実行する各種手法を解説 映画レビューのデータセットに対して各種手法を適用して分類 	<ul style="list-style-type: none"> Long Short-Term Memory Gated Recurrent Unit
case study 08: Peeking into the Black Box An Actuarial Case Study for Interpretable Machine Learning	<p>< freMTPL2freqデータ ></p> <ul style="list-style-type: none"> ブースティングツリーやNNなどのブラックボックス機械学習モデルを説明および解釈するためのツールを解説 	<ul style="list-style-type: none"> GLM NN XGBoost
case study 09: Convolutional neural network case studies	<ul style="list-style-type: none"> NNの一種である畳み込みニューラルネットワーク (CNN) の解説 CNNは画像または時系列で一般的な空間構造を見つけるのに適しており、生命保険の例としてCNNを使用し死亡率の異常を検出する方法を解説 	<ul style="list-style-type: none"> Convolutional neural network (CNN)
case study 10: LocalGLMnet: a deep learning architecture for actuaries	<ul style="list-style-type: none"> LocalGLMnet を解説。LocalGLMnet は変数選択、相互作用の解釈を提供し、変数の重要度をランク付けできる 傷害保険のデータにLocalGLMnetを適用 	<ul style="list-style-type: none"> LocalGLMnet RNN

10

Tutorials の続きはこのような形で 10 種類あるのですが、詳細については割愛させていただきます。

<h2>2. NNによるクレーム頻度モデリングの概要</h2>

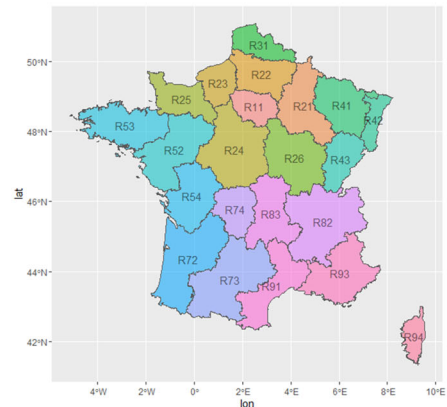
11

続いて、ニューラルネットによるクレーム頻度モデリングの概要について、説明させていただきます。

データ

- 「freMTPL2freq」データ
 - フランスの自動車第三者賠償責任 (MTPL) 保険のデータで、ある1会計年度に観測されたクレームの件数を含む
 - 全部で677,991件のデータ、項目数は12

※ 現在のRegion数とは異なる22Regionのデータ (海外県除く)



R42 Alsace	R94 Corse	R31 Nord-Pas-de-Calais
R72 Aquitaine	R43 Franche-Comté	R52 Pays de la Loire
R83 Auvergne	R23 Haute-Normandie	R22 Picardie
R25 Basse-Normandie	R11 Île-de-France	R54 Poitou-Charentes
R26 Bourgogne	R91 Languedoc-Roussillon	R93 Provence-Alpes-Côte d'Azur
R53 Bretagne	R74 Limousin	R82 Rhône-Alpes
R24 Centre	R41 Lorraine	
R21 Champagne-Ardenne	R73 Midi-Pyrénées	

12

本日使用しますデータは、「freMTPL2freq」というフランスの自動車保険のデータで、全部で 677,991 件のデータが含まれております。右側の方にフランスに地図が載っておりますが、地域データも含んだ情報となっております。

データの項目

- IDpolを除くと11項目
- ClaimNbを予測することが目的

No	Item	Type	説明
1	IDpol	num	ポリシー番号 (一意の識別子)
2	ClaimNb	num	クレーム件数
3	Exposure	num	年単位の総エクスポージャー
4	Area	Factor(6)	エリアコード (順序付きカテゴリ変数: Densityを分類)
5	VehPower	num	エンジン出力
6	VehAge	num	車齢
7	DrivAge	num	運転者の年齢
8	BonusMalus	num	Bonus-Malusの水準、50~230 (基準レベルは100)
9	VehBrand	Factor(11)	車のブランド
10	VehGas	Factor(2)	ディーゼルかガソリンか
11	Density	num	運転者居住地のkm ² あたりの人口密度
12	Region	Factor(22)	2016年以前の地域区分

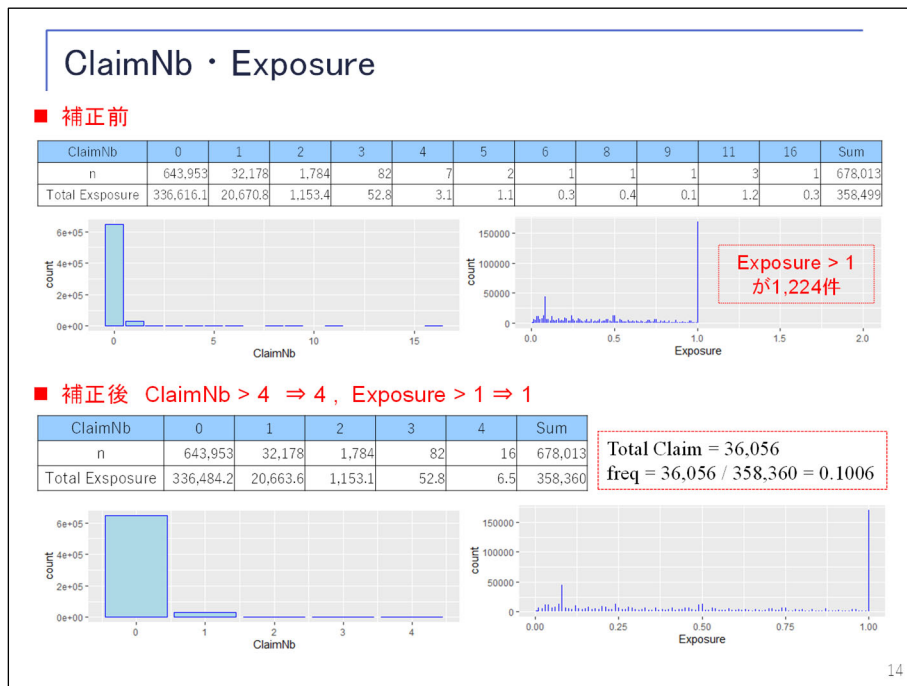
$$\text{freq} = \frac{\text{sum}(\text{ClaimNb})}{\text{sum}(\text{Exposure})}$$

13

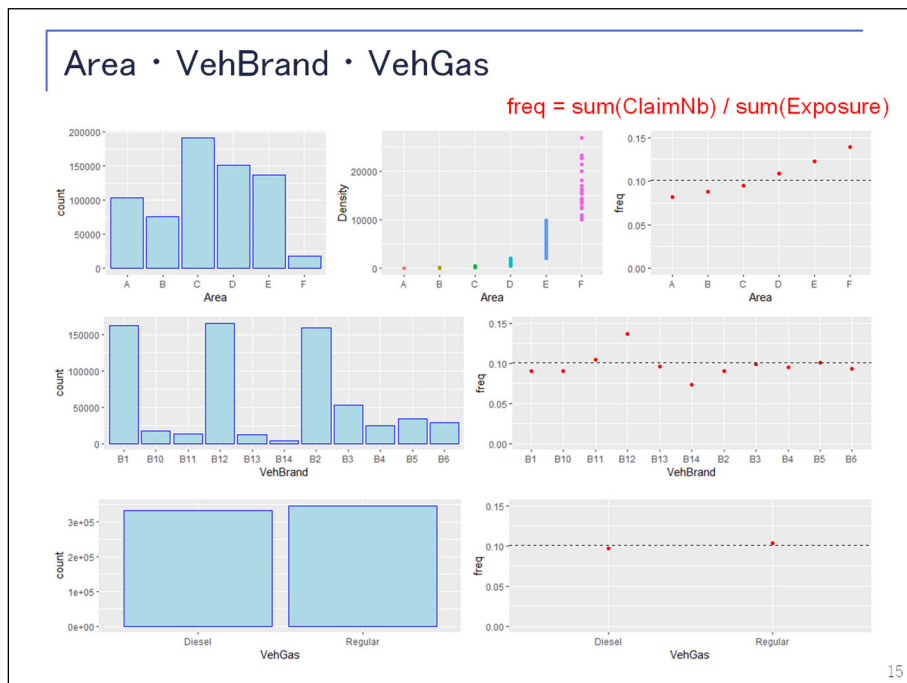
データ項目について説明させていただきます。一番上、No. 1については、「IDpol」ということで、ポリシー番号ですので、実質 11 項目あります。No. 2 が「ClaimNb」ということで、クレーム件数、こちらを予測することが目的になります。また、下の方に記載しておりますが、フリークエンシー・事故頻度として「ClaimNb」を「Exposure」で割ったものとして定義しております。

No. 4以降については「Area」、「VehPower」(エンジン出力)、「VehAge」(車齢)、「DrivAge」(運転者年齢)、「BonusMalus」、「VehBrand」(車のブランド)、「VehGas」(ディーゼルかガソリン)

ンか)、「Density」(人口密度)、「Region」(地域) というデータになっております。

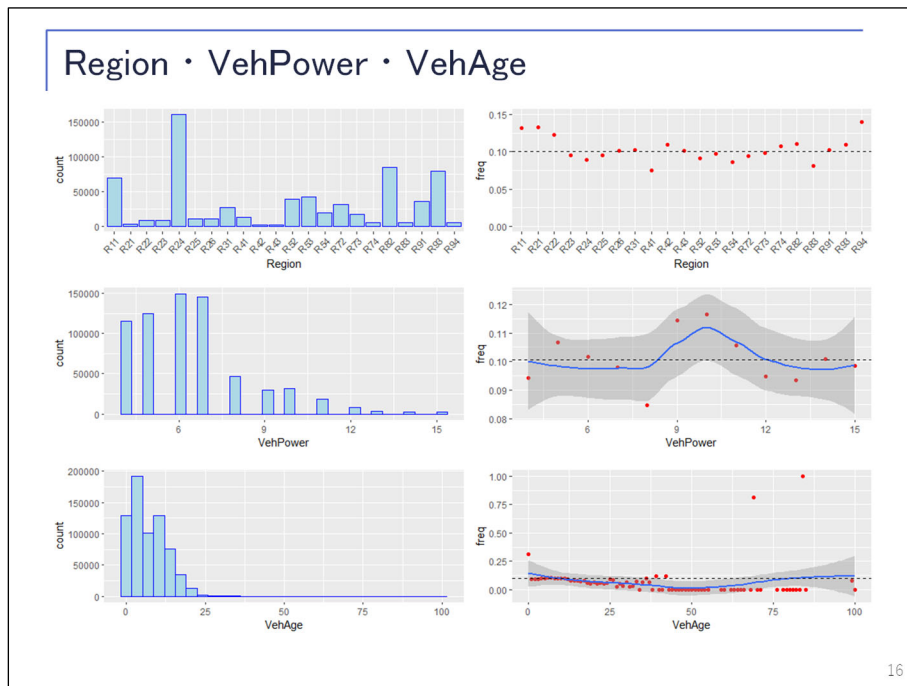


データについて、補正を行っています。ClaimNbとExposureについては、異常値と思われるデータがあるため、これを補正しています。クレーム件数は4件超というデータが含まれているのですが、こちらについては、異常値とみなし4を最大とします。また、Exposureについても、1以上のExposureが含まれるため、1を最大値とします。そうしますと、事故頻度は全体で約10%ということになります。

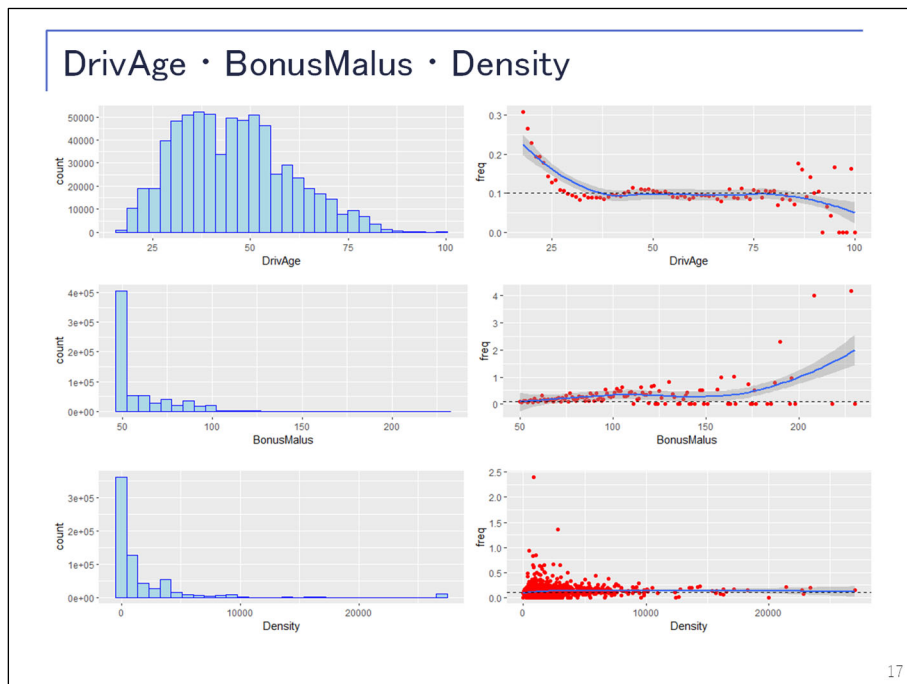


続いて、項目ごとに見ていきたいと思いますが、「Area」については、記載の通りのポートフォリオになっておりまして、「Density」によって分けております。「Density」(人口密度)が一番高いとFという区分になっており、順序付きカテゴリーのデータとなっています。続いて「VehBrand」(自動車ブランド)については、記載の通りとなっております。右の方は事故頻度

を表しておりますが、B12の事故頻度が高いことがうかがえます。続いて、燃料（VehGas）についてはそれほど差異がないことが分かります。



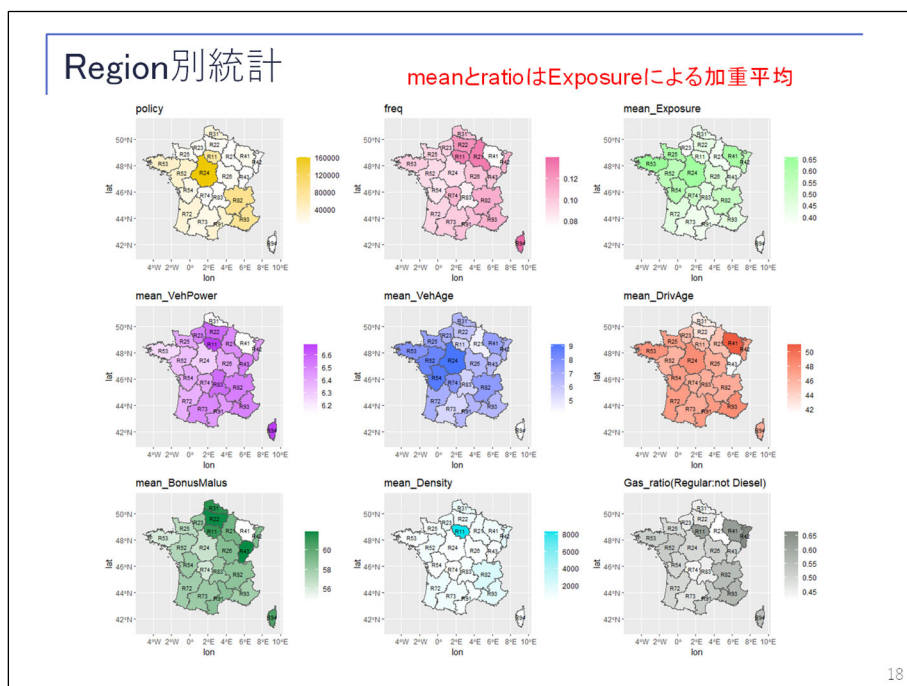
続きまして、「Region」（地域）です。こちらについても、かなり特徴が出てきており、最大で2倍程度の事故頻度格差が出ております。次の「VehPower」からは数値になりますが、こちらについては、真ん中辺りの馬力の事故頻度が高いということがうかがえます。車齢については、25年までのポートフォリオが多くなっており、少し分かりにくいのですが、新しい車ほど事故頻度が高いということがわかります。



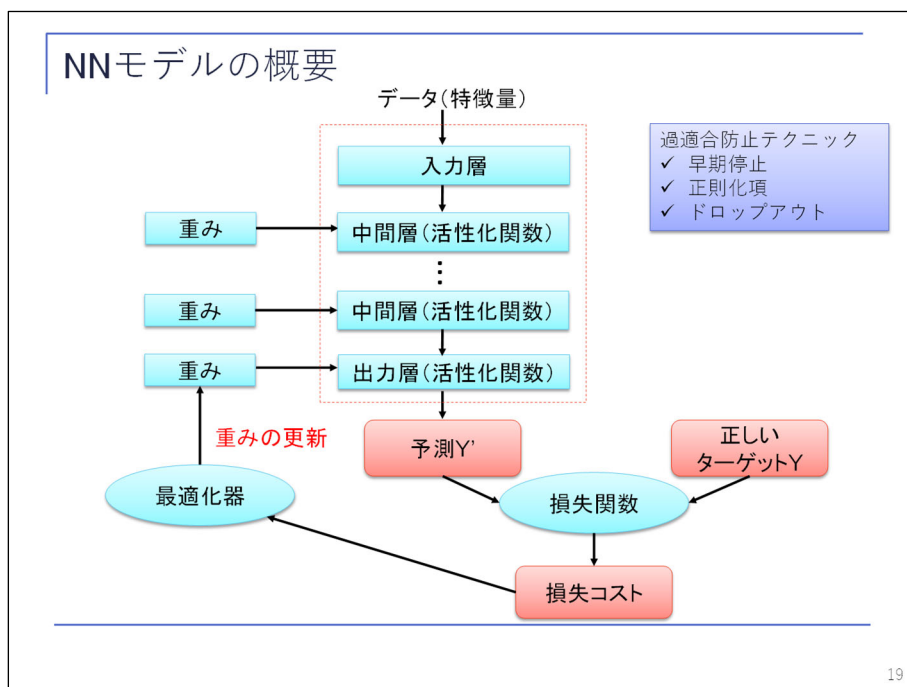
続いて、「DrivAge」です。こちらは、運転者年齢は若年層ほど事故頻度が高く、そのあとは安定していて、その後、データのばらつきというものはあるかと思いますが、高齢者になると事故頻度が低下傾向となります。続いて、「BonusMalus」は高くなるほど事故頻度が高くなる

傾向となっております。

人口密度は、このグラフでは分かりにくいのですが、先ほどの「Area」の通り、人口密度が上がると事故頻度も上昇いたします。



地域別にまとめたものがこちらになります。左上の黄色は、ポリシー数を表しております。ピンクの方はフリークエンシーを表しております。その他、「mean」で始まるものについては、各項目の平均値を表しています。このように、各項目とも、地域ごとにばらつきがあることがうかがえます。



続いて、ニューラルネットモデルの概要について説明いたします。真ん中の赤枠で囲われているところ、こちらがいわゆるモデルとなりますが、入力層、幾つかの中間層と、最後に出力層があります。また、その層には活性化関数というものが定義されています。出力層の結果が

予測値 Y' ということ、予測値を出力いたします。それに対して教師あり学習ですので、正解データ Y というものがあり、この 2 つから損失関数を基に損失コストというものを求めます。最適化器は、損失コストを最小化するようにニューラルネットのパラメータである「重み」というものを調整していきます。ニューラルネットのパラメータは、この重みとバイアスというものになっております。ニューラルネットでは、この調整を学習と言いまして、一連の流れごとに重みを更新していきます。

Rとkerasパッケージ

■ kerasパッケージ

- NNモデル構築で用いるPythonのKerasライブラリへのインターフェイス
- Kerasを用いると、Tensorflow (Googleが公開している機械学習用のライブラリ) を用いて簡単にディープラーニングのモデルを実装可能
- kerasパッケージだけでは実行不可であり、事前にAnaconda・MinicondaなどのPython開発環境の構築が必要

(参考1) <https://www.python.jp/install/anaconda/windows/install.html>

(参考2) <https://web.stanford.edu/~hastie/ISLR2/keras-instructions.html>

■ Kaggle Notebook

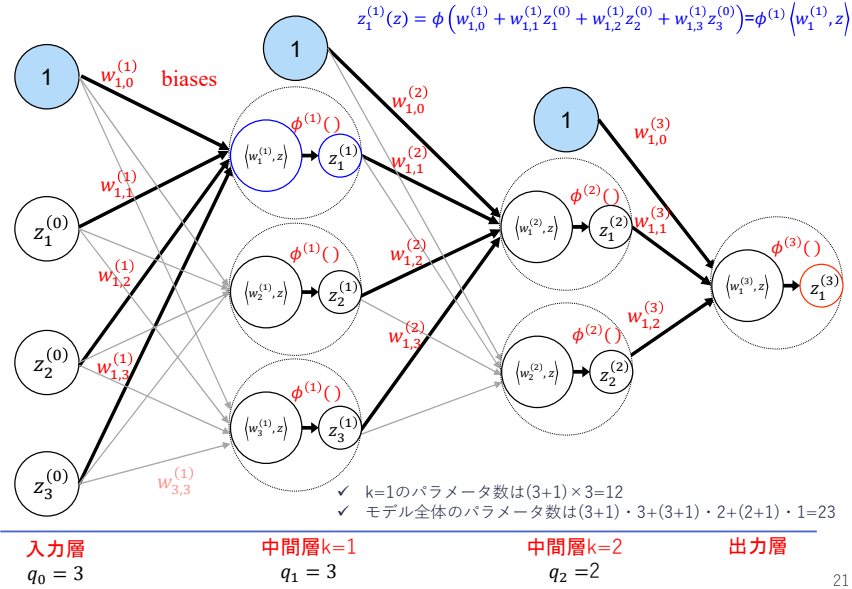
- Kaggle Notebookはクラウド環境でRが使用でき、kerasパッケージもインストール済み
- 利用規約に注意が必要：[for your own internal, personal, non-commercial use](#)
- 本発表の一部コードを下記に掲載

<https://www.kaggle.com/nn2021rkeras/nn2021rkeras>

20

ここで留意点です。本発表のニューラルネットワークは R 言語と keras パッケージを用いており、これはニューラルネットモデル構築で用いる Python の keras ライブラリのインターフェイスとなっています。keras を用いますと、Tensorflow を用いて簡単にディープラーニングが実装できます。ただし、R の keras パッケージだけでは実行不可能であり、Anaconda 等の Python 環境のインストールが必要になります。なお、クラウド環境の Kaggle Notebook というものを用いれば、すぐに R と keras が使用できます。ただし、商業利用はできませんので、青字で記載の利用規約に注意が必要となります。本発表の前半部分については、一番下に記載の URL で、Kaggle Notebook 上にコードを公開しておりますので、再現することが可能となっております。

NNモデルの概要(K=2)



21

続いて、ニューラルネットワークモデルの説明となります。一番左が入力層になっております。続いて、真ん中の方に2つありますが、こちらが中間層と呼ばれるものとなっております。最後に出力層があります。各層の一番上にある「1」というものについてはバイアス (biases) と呼ばれるもので、GLMで言うところの切片項となります。各矢印のところに w というものがありますが、こちらは重みとなっております、ニューラルネットワークのパラメータとなっております。その他の層数やニューロン数と呼ばれるものについてはハイパーパラメータとなっております。

中間層1の一番上のこの青で囲んだものの計算式を上の方に記載しておりますが。入力層の $z_1^{(0)}$ 、 $z_2^{(0)}$ に対して重みを線形結合したものに活性化関数 ϕ を掛けたものが、こちらの出力となりまして、活性化値と呼ばれるものとなっております。

NNモデルの記号の定義

no	記号	説明	備考
1	K	中間層数 (入力層と出力層は含めない)	ハイパーパラメータ
2	k	中間層の順番、 $k = 1, \dots, K$	
3	q_k	k 番目の中間層のニューロン数	ハイパーパラメータ
4	$z_j^{(k)}$	k 番目の中間層の j 番目のニューロン	
5	$z^{(k)}$	k 番目の中間層のニューロンベクトル $z^{(k)} = (z_0^{(k)}, z_1^{(k)}, \dots, z_{q_k}^{(k)})'$	
6	$w_{j,i}^{(k)}$	$(k-1)$ 番目の中間層の i 番目のニューロンから k 番目の中間層の j 番目のニューロンへの重み	切片 $w_{j,0}^{(k)}$ を biases と呼ぶ
7	$w_j^{(k)}$	k 番目の中間層の j 番目のニューロンに対する重みベクトル $w_j^{(k)} = (w_{j,0}^{(k)}, w_{j,1}^{(k)}, \dots, w_{j,q_{k-1}}^{(k)})'$	
8	$w^{(k)}$	$w^{(k)} = (w_1^{(k)}, w_2^{(k)}, \dots, w_{q_k}^{(k)})' = (w_{1,0}^{(k)}, w_{1,1}^{(k)}, \dots, w_{q_k-1,q_k}^{(k)})'$	
9	$\langle w_j^{(k)}, z \rangle$	$z_j^{(k)}$ を計算するための $(k-1)$ 番目の中間層のニューロンと $w_j^{(k)}$ の内積	
10	$\phi^{(k)}$	k 番目の中間層の活性化関数	ハイパーパラメータ

$$z_j^{(k)}(z) = \phi \left(w_{j,0}^{(k)} + \sum_{l=1}^{q_{k-1}} w_{j,l}^{(k)} z_l \right) \stackrel{\text{def}}{=} \phi(w_j^{(k)}, z),$$

22

モデル記号の定義、詳細については、割愛させていただきます。

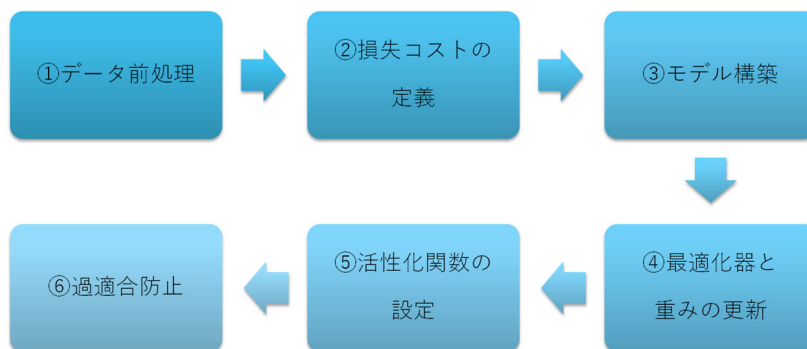
3. NNのモデルの詳細

23

続いて、モデルの詳細について説明させていただきます。

説明の流れ

- 「ADS TutorialのCase study 02(*1)」 に沿い、NNを用いた自動車保険のクレーム頻度分析について説明

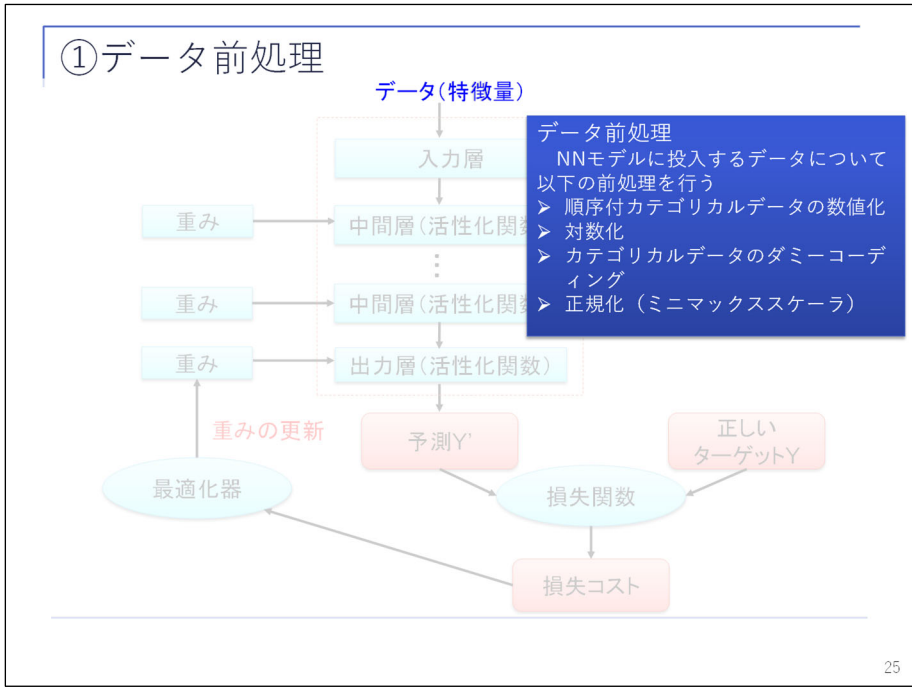


*1 Ferrario (2018)

24

ここから ADS Tutorials の case study 02 に沿って、ニューラルネットを用いた自動車保険のクレーム頻度分析について説明していきます。

まず初めにデータの前処理、続いて損失コストの定義、モデル構築、最適化器と重みの更新、活性化関数の設定、過適合防止という順に説明させていただきます。



まず、データの前処理なのですが、こちらは特にニューラルネットに限ったものではなく、機械学習で一般的に行われるデータ前処理をしていきます。

データのの前処理

- 数値化・正規化を実施

No	Item	dummy coding	MinMax Scaler	other	変換の説明	次元
1	IDpol				使用せず	-
2	ClaimNb			○	4を上限	-
3	Exposure			○	1を上限	1
4	Area		○		{A, ..., F}を{1, ..., 6}にしてMinMaxScaler	1
5	VehPower		○		そのままMinMaxScaler	1
6	VehAge		○		20を上限にしてMinMaxScaler	1
7	DrivAge		○		90を上限にしてMinMaxScaler	1
8	BonusMalus		○		150を上限にしてMinMaxScaler	1
9	VehBrand	○			dummy coding (変換後21変数)	10
10	VehGas			○	$\pm 1/2$ に変換	1
11	Density		○		対数化してMinMaxScaler	1
12	Region	○			dummy coding (変換後21次元)	21

ExposureからRegionが特徴量空間(説明変数)となり合計39次元

26

ここでは、順序付きカテゴリカルデータの数値化、対数化、カテゴリカルデータのダミーコーディング、正規化というものを行います。数値については、全体的に-1から1に収まるように変換します。また、最終的には39次元となります。

ダミーコーディング/正規化

- カテゴリカルデータのダミーコーディング
 - GLMで使われるダミーコーディングを本発表で用いる
 - NNではワンホットエンコーディングも用いられる

<VehBrandに対する例>

ダミーコーディング

	特徴量 (10次元)									
B1	0	0	0	0	0	0	0	0	0	0
B10	1	0	0	0	0	0	0	0	0	0
B11	0	1	0	0	0	0	0	0	0	0
B12	0	0	1	0	0	0	0	0	0	0
B13	0	0	0	1	0	0	0	0	0	0
B14	0	0	0	0	1	0	0	0	0	0
B2	0	0	0	0	0	1	0	0	0	0
B3	0	0	0	0	0	0	1	0	0	0
B4	0	0	0	0	0	0	0	1	0	0
B5	0	0	0	0	0	0	0	0	1	0
B6	0	0	0	0	0	0	0	0	0	1

ワンホットエンコーディング

	特徴量 (11次元)										
B1	1	0	0	0	0	0	0	0	0	0	0
B10	0	1	0	0	0	0	0	0	0	0	0
B11	0	0	1	0	0	0	0	0	0	0	0
B12	0	0	0	1	0	0	0	0	0	0	0
B13	0	0	0	0	1	0	0	0	0	0	0
B14	0	0	0	0	0	1	0	0	0	0	0
B2	0	0	0	0	0	0	1	0	0	0	0
B3	0	0	0	0	0	0	0	1	0	0	0
B4	0	0	0	0	0	0	0	0	1	0	0
B5	0	0	0	0	0	0	0	0	0	1	0
B6	0	0	0	0	0	0	0	0	0	0	1

- 正規化 (MinMaxScaler)
 - 入力層の特徴量が-1から1に収まるように調整

$$x_l \mapsto x_l^* = \frac{2(x_l - m_l)}{M_l - m_l} - 1 \in [-1, 1].$$

27

ダミーコーディングなのですが、こちらは GLM で用いられるダミーコーディングというものを今回用います。ただし、ニューラルネットにおいては、右に記載のワンホットエンコーディングもよく用いられます。続いて、数値データについては、-1 から 1 に収まるように MinMaxScaler というもので正規化をしていきます。

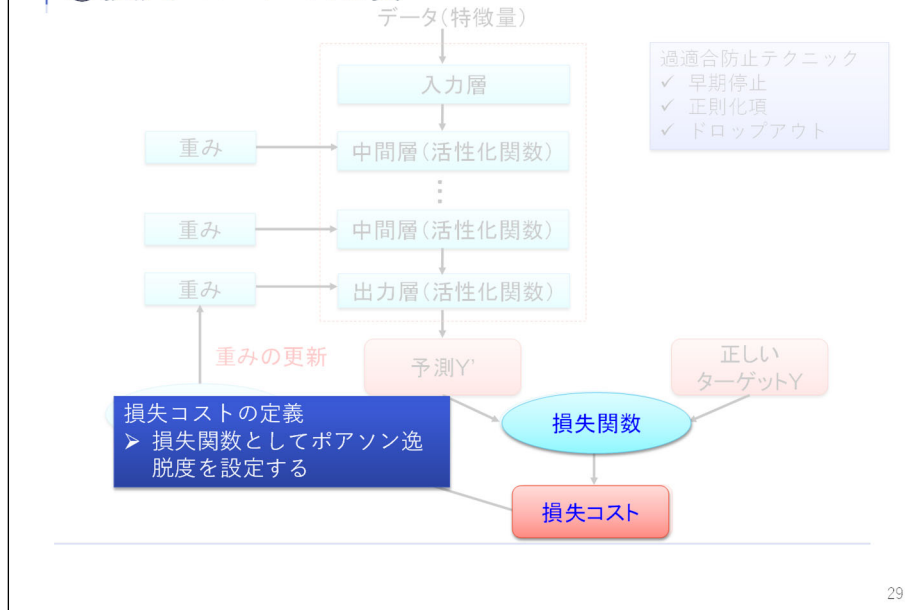
変換後特徴量 (変数) 一覧

```
'data.frame': 678013 obs. of 39 variables:
 $ AreaX : num 0.2 0.2 -0.6 -0.6 -0.6 0.6 0.6 -0.2 -0.2 -0.6 ...
 $ VehPowerX : num -0.818 -0.818 -0.636 -0.455 -0.455 ...
 $ VehAgeX : num -1 -1 -0.8 -1 -1 -0.8 -0.8 -1 -1 -1 ...
 $ DrivAgeX : num 0.0278 0.0278 -0.0556 -0.2222 -0.2222 ...
 $ BonusMalusX: num -1 -1 -1 -1 -1 -1 -1 -1 -0.64 -0.64 -1 ...
 $ Br2 : int 0 0 0 0 0 0 0 0 ...
 $ Br3 : int 0 0 0 0 0 0 0 0 ...
 $ Br4 : int 1 1 1 1 1 1 1 1 ...
 <省略>
 $ Br11 : int 0 0 0 0 0 0 0 0 ...
 $ VehGasX : num 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 ...
 $ DensityX : num 0.392 0.392 -0.218 -0.151 -0.151 ...
 $ R2 : int 0 0 0 0 0 0 0 0 ...
 $ R3 : int 0 0 1 0 0 0 0 0 ...
 $ R4 : int 0 0 0 0 0 0 0 0 ...
 <省略>
 $ R22 : int 0 0 0 0 0 0 0 0 ...
 $ Exposure : num 0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
```

28

変換後の特徴量一覧がこちらとなります。全部で約 67 万件のデータがありまして、39 の特徴量、39 次元となっております。青字の方が VehBrand のダミーコーディング、赤字のところは Region のダミーコーディングとなっております。

②損失コストの定義



続いて、損失コストについて説明いたします。

損失コストの定義

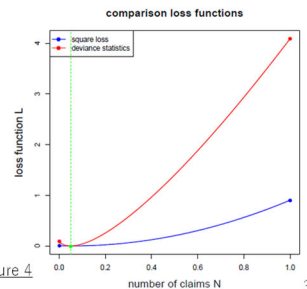
- 損失コスト
 - 真の値と予測値のズレを表し小さいほど望ましい
 - ハイパーパラメータの調整やNNの学習に用いる
- ポアソン逸脱度 (Poisson deviance loss function)
 - 損失コストとしてポアソン逸脱度を用いる (※)

$$\mathcal{L}(\mathcal{D}, \hat{\mu}(\cdot)) = \frac{1}{n} \sum_{i=1}^n 2N_i \left[\frac{\hat{\mu}(x_i, v_i)}{N_i} - 1 - \log \left(\frac{\hat{\mu}(x_i, v_i)}{N_i} \right) \right]$$

※kerasでは $\hat{\mu}(x_i, v_i)$ に依存しない箇所を省略しており、以降グラフ上では損失コストの値が異なる

$$\mathcal{L}^{\text{Keras}}(\mathcal{D}, \hat{\mu}(\cdot)) = \frac{1}{n} \sum_{i=1}^n \hat{\mu}(x_i, v_i) - N_i \log \hat{\mu}(x_i, v_i)$$

- 回帰ではMSEが用いられることも多いが、予測値が小さいカウントデータ予測の場合はポアソン逸脱度が望ましい
- 【右図】ポアソン分布仮定の下、平均値が0.05(緑)のとき、MSE(青)に比べてポアソン逸脱度(赤)の方が損失コストが高くなり感応的



ADS Case study 02, Figure 4

損失コストにつきましては、把握できる真の値とモデルにより予測した値のずれを表し、損失コストが小さいほど望ましいものになり、ニューラルネットではハイパーパラメータの調整に用います。本発表の損失コストはポアソン逸脱度を用います。定義はこちらの方に記載の通りです。なお、赤字で記載しておりますが、keras 上でもポアソン逸脱度の損失コストが定義されていますが、 $\hat{\mu}$ に依存しないところを省略しているため、以降、グラフが出てくるのですが、keras の結果の損失コストと表での損失コストの値が異なりますので、ご注意ください。

なお、回帰ではMSEが用いられることも多いかと思いますが、予測値が小さいカウントデータの予測の場合はポアソン逸脱度が望ましいことが紹介されています。右は、ポアソン分布を仮定の下、平均値0.05(5%)のとき、MSE(青)に比べてポアソン逸脱度の方がロスが高くな

っております、感応的であることを示しております。

訓練・評価・テストデータ

- データを9:1で訓練・テストデータに分割、更に訓練データの20%を評価データとする

データの種類	役割 / 損失コスト
訓練データ	重みの学習に使用 / in-sample loss
評価データ	モデルの汎用性を評価 / validation loss (ハイパーパラメータ調整目的)
テストデータ	最終的なテストに使用 / out-of-sample loss

訓練データ $|D| = 610,212$ 評価データ 20% テストデータ $|T| = 67,801$

in-sample loss $\mathcal{L}(D, \hat{\mu}(\cdot))$ out-of-sample loss $\mathcal{L}(T, \hat{\mu}(\cdot))$

ADS Case study 02, Figure 3

続いて、データにつきましては、9対1の割合で訓練データとテストデータに分割いたします。また、ニューラルネットモデルの汎用化性能を評価、すなわち過学習が起こっていないかを評価するために、訓練データの20%を評価データとします。訓練データによる損失コストをこちらの「in-sample loss」、テストデータによるロス「out-of-sample loss」と表記いたします。

③モデル構築 (shallow network)

データ(特徴量)

入力層

中間層(活性化関数)

出力層(活性化関数)

重み

重み

重みの更新

最適化器

予測 \hat{Y}

正しいターゲット Y

損失関数

損失コスト

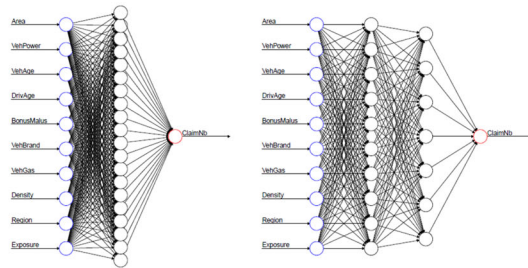
モデル
 > 中間層が1のshallow networkモデルの実装

モデル
 中間層数・ニューロン数・活性化関数

続いて、モデルについて説明いたします。ここでは、中間層が1つのシャロウ・ネットワークについて説明いたします。

レイヤーとニューロン数

- 中間層数 (K)
 - 中間層 1つをシャロウ・ネットワーク、2つ以上をディープ・ネットワークという
 - 中間層数はハイパーパラメータで任意設定
 - 層が多いほどパラメータが増え表現力が高まるが処理時間がかかり過適合しやすい
- ニューロン数
 - 各中間層に含まれるユニットの数 (ハイパーパラメータ)



ADS Case study 02, Figure 3

33

まず、中間層 1つの場合をシャロウ・ネットワークと言いまして、2つ以上の場合をディープ・ニューラルネットワークと言います。中間層はハイパーパラメータで任意設定可能となっております、層が多いほどパラメータが増え、表現が高まりますが、過適合しやすくなります。ニューロン数についてもハイパーパラメータ数となっておりますが、中間層に含まれるユニットの数です。左図は中間層 1、ニューロン数 20 のシャロウ・ネットワーク、右側については、中間層が 2つ、ニューロン数は第 1層が 10、第 2層が 7 のディープ・ネットワークとなっております。

R・Kerasを用いたshallow networkモデルの構築

- モデル構築 (K=1)

```

model%>%
  layer_dense(units=20,activation='tanh',input_shape=39)%>%
  layer_dense(units=1,activation='exponential')
  # 中間層ニューロン数 中間層活性化関数 入力層ニューロン数
  # 出力層ニューロン数 出力層活性化関数
model%>%compile(loss='poisson',optimizer='nadam')
  # 損失関数 最適化器
model%>%fit(learn.XX,learn$claimNb,epochs=100,batch_size=5000)
  # エポック数 バッチ数
    
```

- モデル確認

```

Model
Model: "sequential_169"

Layer (type)                 Output Shape                 Param #
-----
dense_339 (Dense)            (None, 20)                  800
dense_338 (Dense)            (None, 1)                   21
-----
Total params: 821
Trainable params: 821
Non-trainable params: 0
    
```

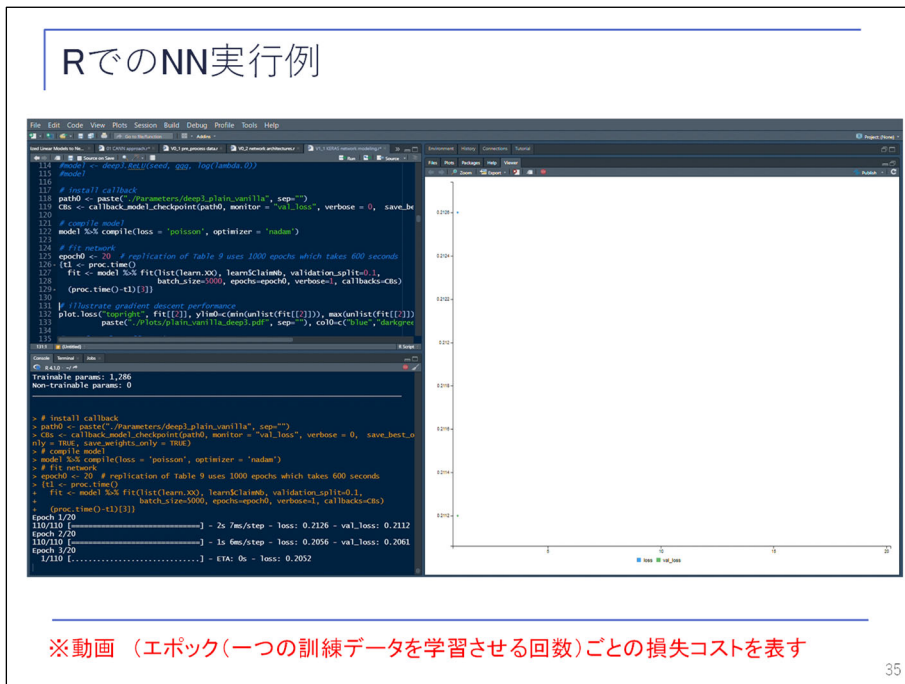
34

続いて、R、keras を用いたモデルの構築について説明いたします。

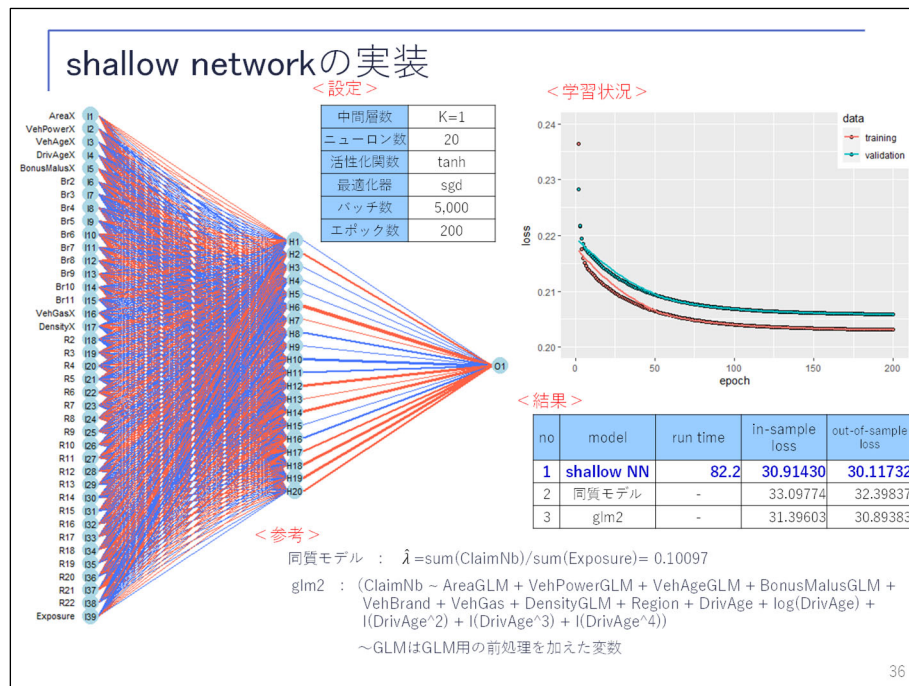
上の方に R のコードを記載しております。初めの「model」から始まり、「layer_dense」というところで中間層のニューロン数 20、あとは後ほど説明をします活性化関数、入力層のニュー

ーロン数 39 というものを設定します。2 番めの「layer_dense」というところは、出力層を定義しておりまして、ニューロン数と、活性化関数としては指数関数を設定しております。

次に、「model」の「compile」という部分がございます、ここでは、損失関数、先ほどのポアソンと、最適化器というもので「nadam」というものを設定しております。最終的に fit 関数で実行いたしますが、後ほど説明する「epochs」というものと「batch_size」というものをそれぞれ 100、5000 で設定しております。「model」についてはサマリーすることができまして、こちらでは、「model」は 821 のパラメータがあるということが分かります。



ここで、R でのニューラルネットの実行例を、ご覧いただきます。動画となっております。右の方に着目ください。横軸がエポック数となっております、縦軸が損失コストとなっております。青が訓練データによる損失コスト、緑が評価データによる損失コストを表しております、どちらもエポックが進むたびに損失コストが小さくなっていること、すなわち学習していることが分かります。



ここで、シャロウ・ネットワークの実行結果について説明いたします。設定については、記載の通りとなっております。右上の方に学習状況を記載しておりまして、赤が訓練データ、青が評価データとなっておりますが、エポックが進むごとにロスが下がっていくということが分かります。

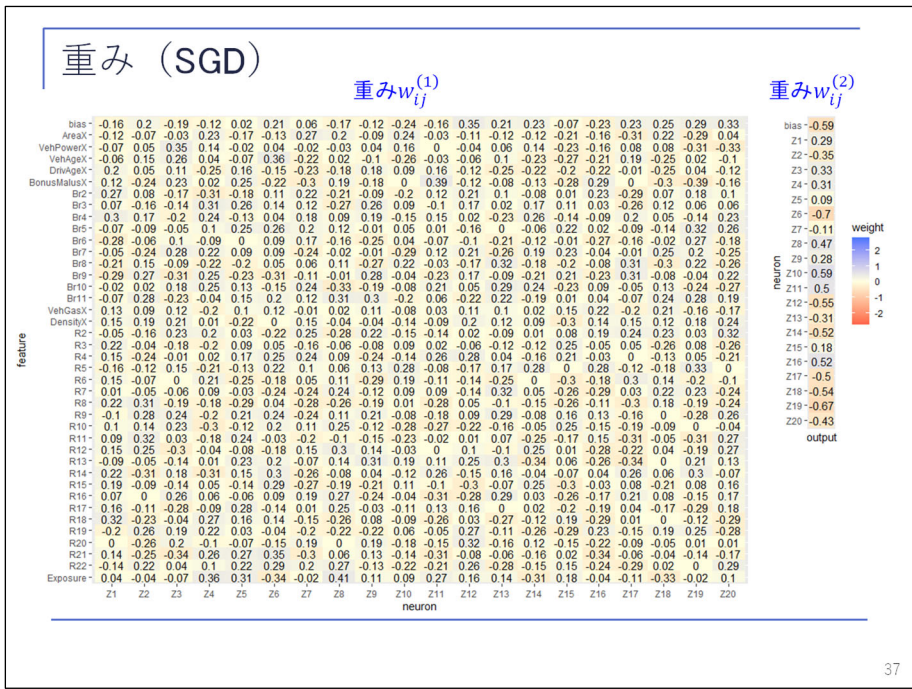
結果について、No. 1 では、このモデルの実行時間を 82.2 秒で、訓練データのロス (in-sample loss) は 30.91430、テストデータのロス (out-of-sample loss) は 30.11732 であることが分かります。なお、タイムというもの (実行時間) は PC スペックによって異なりまして、今回は、第 7 世代 COREi5、メモリ 8 メガということなので、それほどスペックの高くない PC で実行しておりまして、GPU は使っておりません。

参考として、No. 2 で条件によらず、同じポアソン・パラメータとした同質モデルのロスを記載しておりますが、out-of-sample loss は 32.39837 となっております。シャロウ・ネットワークは 7% 程度の改善につながっているというようになります。

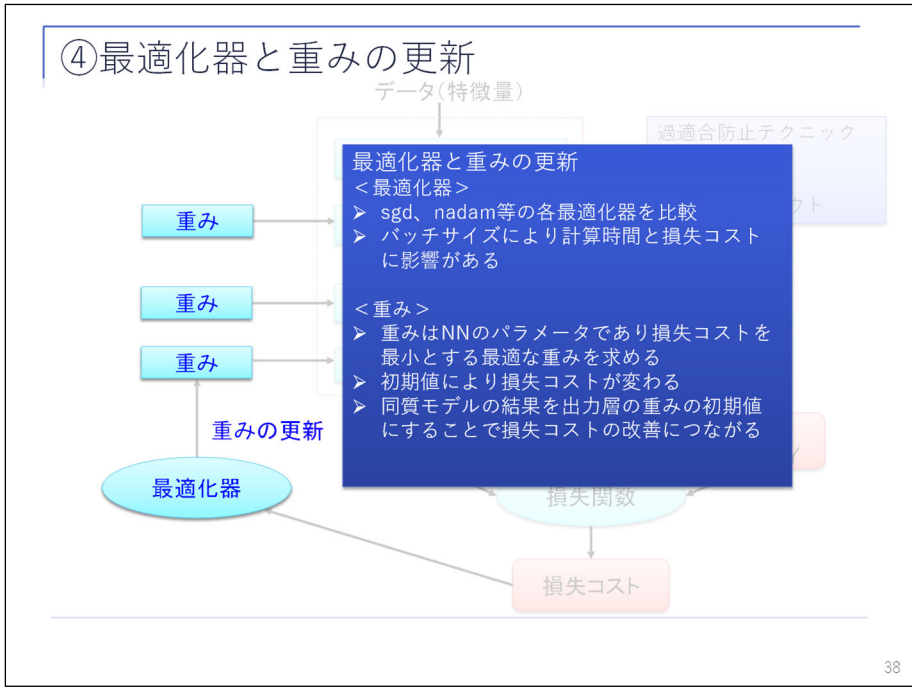
また、後ほど後半パートで出てきますが、glm2 として、GLM 用の前処理を施した特徴量で当てはめた GLM の out-of-sample loss を記載しておりますが、30.89383 となっております。GLM の算式は下方に記載しておりますが、割と複雑な GLM となっております。

なお、参考文献の最後の方の文献により、glm2 を真のモデルとした場合の out-of-sample loss は 28.228 となり、クレーム発生のランダム差に起因する誤差というもののが大半に占めるということが説明されております。このため、モデル改善の難しさがあるということをおっしゃいます。シャロウ・ネットワークについては、まだ改善の余地があるということになっております。

左側は、先ほどのモデルに対してグラフ化 (可視化) したもので、青がプラスの重み、赤がマイナスの重みを示してありまして、線の太さが絶対値を示しています。



続いて、重みについて一覧ですが、こちらのように可視化しております。左の方に入力層の特徴量と、一番上にはバイアスが記載されていまして、重みを数値と色で表したものとなっております。



続いて、最適化器について説明します。keras では数種類の最適化器を選択することができますので、これを比較していきます。

最適化器 (optimizer)

- 勾配降下法 (GDM:gradient descent methods)
 - 最適化器は損失コストを最小にするようにパラメータを更新する (学習)
 - 単に損失コストを最小にするのみならず、なるべく効率よく損失の最小化に到達するアルゴリズムが望ましい
- 最急降下法
 - 損失関数を重みの関数とし、重みベクトルの微分を用いて重みを更新
$$w_{t+1} \leftarrow w_t - \alpha \frac{\partial}{\partial w} L(w)$$

w_t : t回目のエポックの重みベクトル、 $L(w)$: 重みベクトルを引数とする損失コスト、 α : 学習率

勾配 (微分係数) と学習率 α により重みを更新

<課題>
最小値ではなく極小値に陥ったときに更新がストップする
⇒SGD等の手法により解決

39

まず、ニューラルネットでは最適化器は勾配降下法という手法がよく用いられ、損失コストを最小にするようにパラメータを更新していきますが、単に損失コストを最小にするのみならず、なるべく効率良く損失コストを最小化するアルゴリズムが望まれます。勾配降下法の一つである一番基本的な最急降下法というものを下の方で簡単に説明しておりますが、こちらは、損失関数は重みを引数とする関数として、重みベクトルの微分の結果、すなわち勾配を用いて重みを更新していきます。

左の方に図が載っておりますが、t 時点の重みが w_t となっている場合に、微分しますと計数が 0 よりも大きくなりますので、微分が 0 になる点を目指すためには、青の w_{t+1} の方に進めば

よいということが分かります。このように勾配により重みを調整していくものになります。この際に、学習率 α によって重みを更新していきます。課題といたしましては最小値ではなく、極小値に陥ったとき更新がストップするというので、右図のような状況に陥ることがあります。

SGDと派生アルゴリズム

- SGD (確率的勾配降下法)
 - ミニバッチ学習
 - 全ての標本を用いずランダムに選択した標本 (ミニバッチ) により学習
 - バッチサイズが小さいほど損失コストが小さくなる傾向があるが計算時間を要する (トレードオフ)
 - ADS Case study 02でバッチサイズ5,000を推奨のため、本発表でも5,000とした
 - モーメンタム
 - モーメンタム (慣性) を導入し振動を抑制
- Kerasで用いられる最適化器アルゴリズム

sgd	最適な学習率、モーメンタムベースの改善、ネステロフ加速、および最適なバッチを使用して、収束速度に合わせて微調整できる。「確率的」勾配とは、学習データのランダムなサブサンプル (ミニバッチ) で局所的に最適なステップを探索することを意味。
adagrad	勾配のすべての方向で異なり、勾配の方向のサイズを考慮した学習率を選択
adadelat	adagradの修正バージョンで、ハイパーパラメータへの感度を克服
rmsprop	adagradの欠点を克服するための別の手法
adam	adagradと同様に、L2ノルムによって測定された過去の勾配によって誘発されたモーメンタムに基づいて方向的に最適な学習率を検索
adamax	adamにおいて、L2ノルムに代えてL ∞ ノルムを用いたもの
nadam	adamのネステロフ加速バージョン

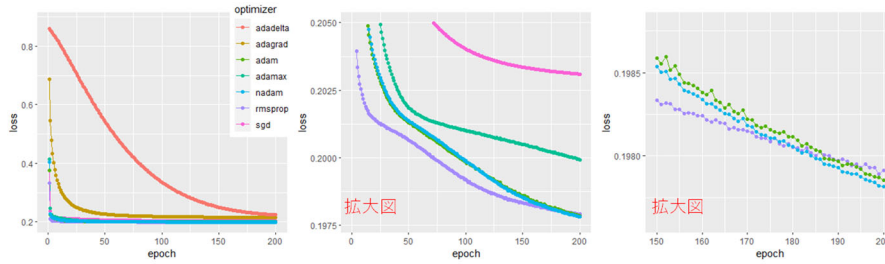
40

続いて、SGD について説明いたします。こちらは、確率的勾配法と呼ばれるものです。こちらは、先ほどのアルゴリズムを発展させたものとなっております、ミニバッチ学習というものと、モーメンタムという特性があります。まずミニバッチ学習については、全ての標本を用いず、ランダムで選択した標本で学習を行います。バッチサイズについても、いろいろと考察がありますが、case study 02の方ではバッチサイズ 5,000 を推奨しているため、本発表でも 5,000 としております。

なお、1 エポックは 5,000 件のデータで更新しているわけではなく、訓練データを 5,000 件ごと、全データ分勾配を求めて処理します。このため、仮に 5 万件のデータがあった場合は 1 エポックで 10 回分、勾配を計算していることとなります。また、モーメンタムという特性を導入することにより、重み更新の振動を抑える手法が導入されています。

keras で用いられている最適化アルゴリズムは、SGD の派生として 7 種類ほど挙げられております。

結果比較 (最適化器)



<設定>

中間層数	K=1
ニューロン数	20
活性化関数	tanh
最適化器	右記
バッチ数	5,000
エポック数	200

<結果>

no	model	run time	in-sample loss	out-of-sample loss
1	sgd	80.6	30.91430	30.11732
2	adagrad	77.9	33.26602	32.38670
3	adadelta	86.2	35.03064	34.22252
4	rmsprop	84.6	29.92512	29.36036
5	adam	83.9	29.90757	29.35680
6	adamax	79.9	30.30915	29.61529
7	nadam	83.9	29.90182	29.35056

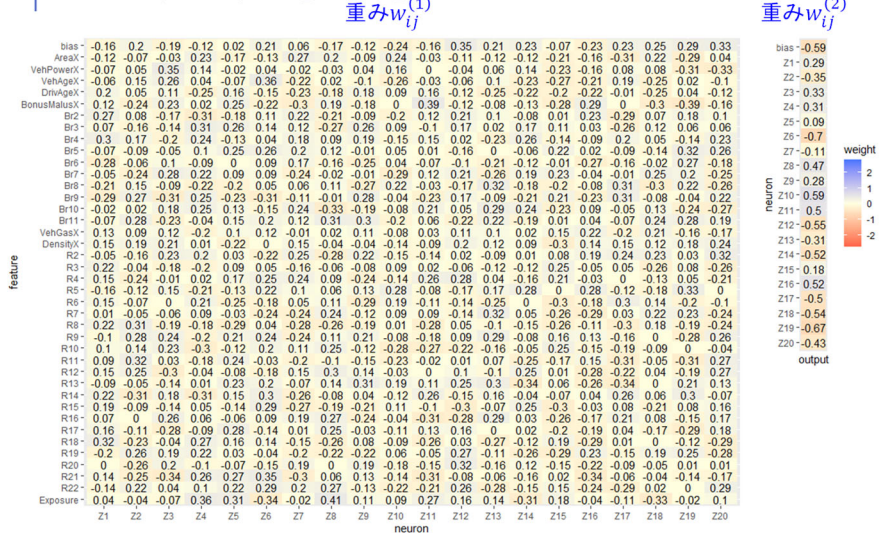
- ✓ 初めはrmspropの損失コスト低減が速いが最終的にはnadamが最良
- ✓ ADS Case study 02でもnadamが最良の結果となり、採用されているため以降nadamを採用

41

これらを比較した結果が、こちらになっております。このような形で最適化器のみ7種類で試し、ロスがどのようになったのかということを示しております。上の方にグラフを記載しております。拡大図で見ていただいたほうがいいかと思いますが、一番右上の拡大図を見ていただきますと、紫の「rmsprop」というものが良かったのですが、最終的にはNo. 7の「nadam」が良いという結果となっております。

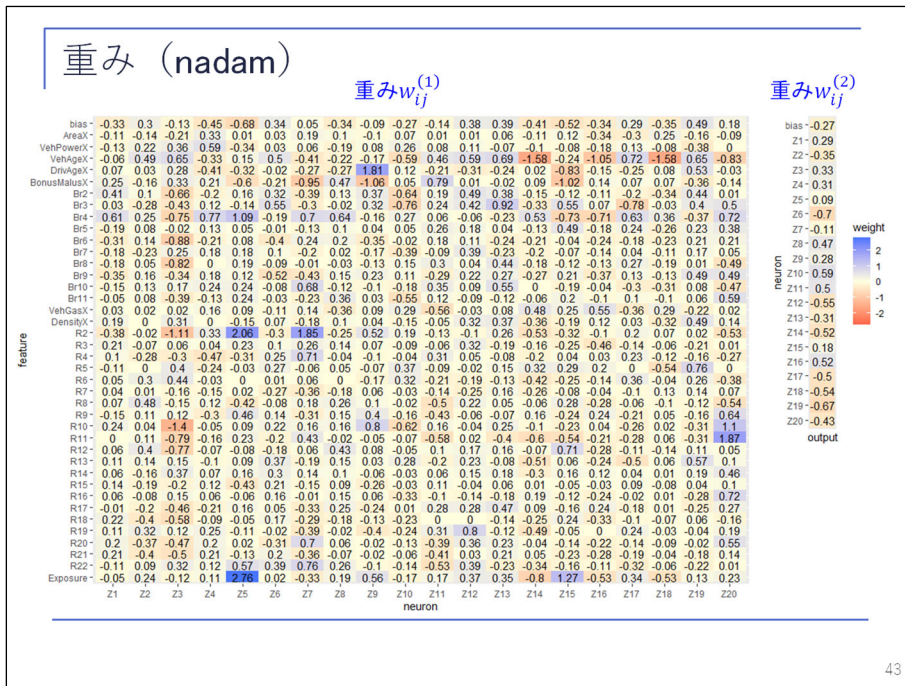
case study 02の方でも「nadam」を採用していきまして、本発表の続きでも「nadam」を採用していきます。

重み (SGD) 【再掲】



42

先ほどのSGDの重みの再掲となっております。



それに対し、今回実施した「nadam」の重みを可視化したものとなっておりますが、色が全体的に濃くなっていることが分かります。最適化器によって重みというものも変わってくるというところが分かります。

重みの初期値

- 重み初期値の違いによるランダム性
 - 重みパラメータの初期値 (ランダム) により損失コストの結果が異なる
 - 【boxplot左側】 50回乱数シードを変えて実行した結果
- 同質モデルの埋込み
 - 【boxplot右側】 出力層の重みの初期値を同質モデルの結果とすると、損失コストが改善され、更にバラつきも減少

```

mode1%>%
  layer_dense(units=20,activation='tanh',input_shape=c(ncol(learn.xx)))%>%
  layer_dense(units=1,activation='exponential',
              weights=list(array(0, dim=c(20,1)), array(log(lambda.0), dim=c(1))))
  
```

➢ boxplot左側は重みの初期値を初期設定とし、右側は出力層の重みを同質モデルの λ のlogを代入

➢ $\log(\hat{\lambda}) = -2.29290$

	no weight control		hom weight	
	in-sample loss	out-of-sample loss	in-sample loss	out-of-sample loss
max	31.19576	30.34742	30.62700	29.92692
min	30.78811	29.99068	30.57338	29.83017
mean	30.95892	30.13706	30.60060	29.86901

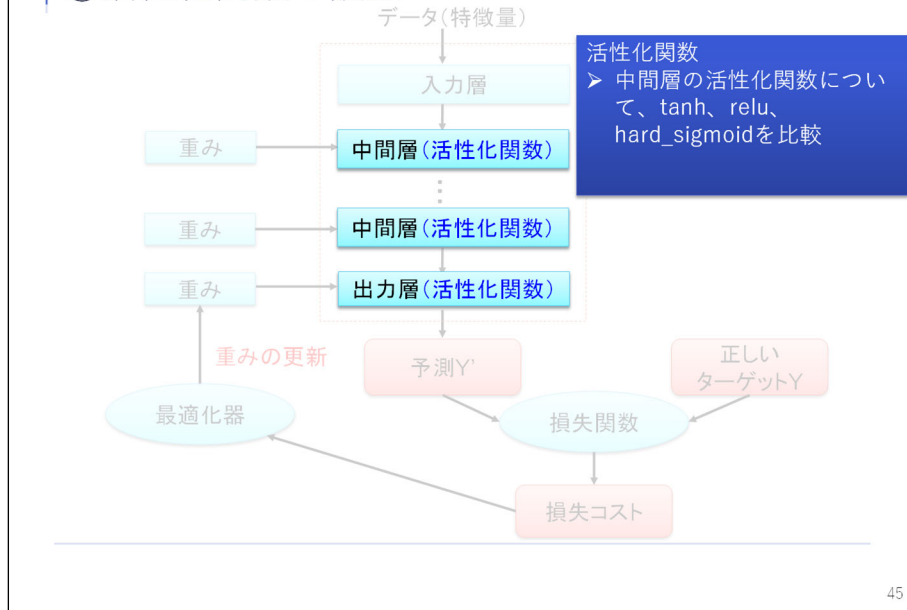
以降重みの初期値として同質モデルより得られた値を設定

続いて、重みの初期値について説明します。重みパラメータの初期値については、何も指定しないとランダムな値となり、結果的に「loss」が変わってきます。右下の boxplot 図、こちらに記載しておりますが、こちらは乱数シードを 50 回変えて予測した結果の損失コストのばらつき具合を示したものです。赤が in-sample loss で青が out-of-sample loss ですが、共にばらついております。

ここで、出力層の重みの初期値として同質モデルの $\hat{\lambda}$ の log を指定して、同じく乱数シードを 50 回変えて検証した結果が、右図となっております。こちらで見ますと、ばらつきも小さ

く、また、最終的な損失コストも小さくなっていることが分かります。重みについては、コード上で指定してあげることができます。このように初期値を指定することは非常に有用で、以降、重みの初期値として同質モデルの結果を指定します。

⑤活性化関数の設定

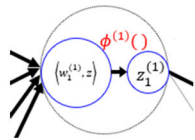


45

続いて、活性化関数について説明いたします。

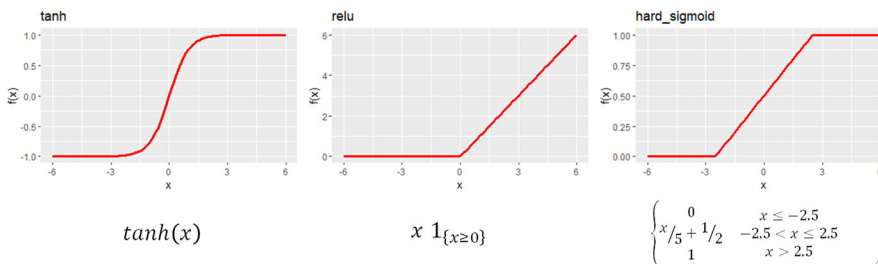
活性化関数 (activation function)

- 各中間層の入力値と重みの内積に対して、活性化関数を適用し活性値として出力
- tanh(hyperbolic tangent)、ReLU、hard sigmoidの3種類について比較を実施
- 中間層ごとに異なる活性化関数を適用することも可能だが本発表ではすべての中間層に同一の活性化関数を適用
- 出力層の活性化関数については本発表前半の部とおしてexponentialに固定



$$z_1^{(1)}(z) = \phi(w_{1,0}^{(1)} + w_{1,1}^{(1)}z_1^{(0)} + w_{1,2}^{(1)}z_2^{(0)} + w_{1,3}^{(1)}z_3^{(0)}) = \phi^{(1)}(w_1^{(1)}, z)$$

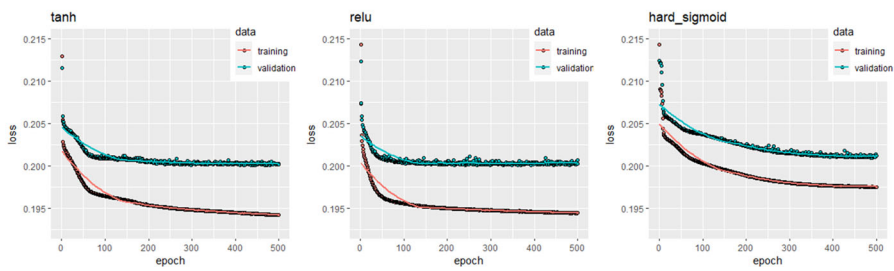
各ニューロンで前の層のニューロンの出力値と重みパラメータの内積を活性化関数 $\phi(\cdot)$ に入力して出力する



46

各中間層の入力値の重みの内積について活性化関数を適用しています。先ほどで見た ϕ のところ。活性化関数としては、本発表では「hyperbolic tangent」、「ReLU」、「hard sigmoid」という3種類について比較しております。それぞれグラフは、下の通りとなっております。また、出力層の活性化関数については指数関数に固定しております。

結果比較 (活性化関数)



<設定>

中間層数	K=3
ニューロン数	20,15,10
活性化関数	右記
最適化器	nadam
バッチ数	5,000
エポック数	500

<結果>

no	model	run time	in-sample loss	out-of-sample loss
1	tanh	336.8	29.22518	29.13526
2	relu	272.6	29.28680	29.24033
3	hard_sigmoid	498.3	29.84429	29.37261

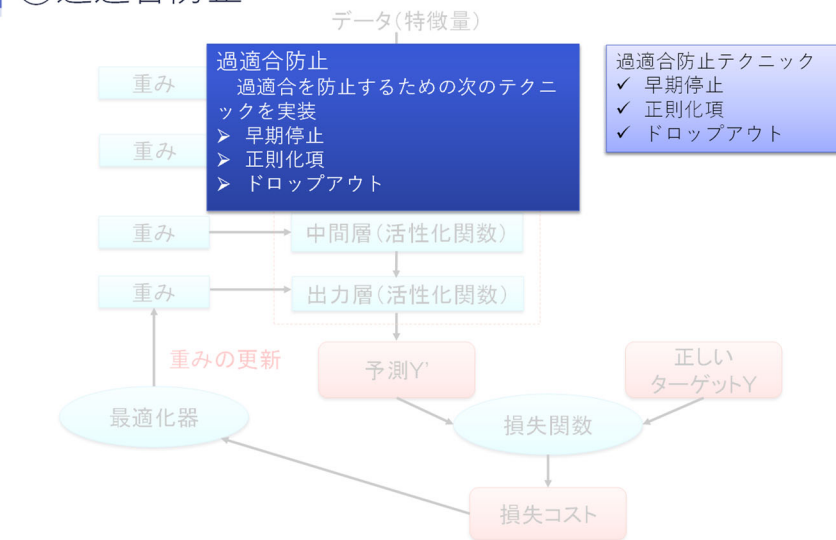
- ✓ tanhが最良の結果
- ✓ hard_sigmoidは収束も遅く、結果も他に比べて損失コストが高い
- ✓ ADS Case study 02ではtanhを推奨

※K=3のディープNNで検討

47

結果として「hyperbolic tangent」(一番左のもの)が、out-of-sample lossが一番小さく、29.13526ということで最良モデルとなっております。case study 02でも、「hyperbolic tangent」を推奨しております。

⑥過適合防止

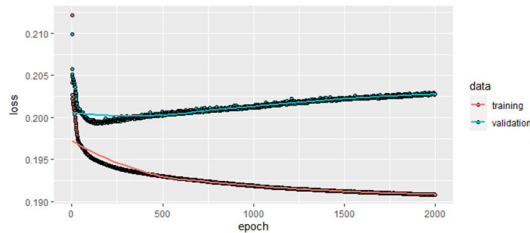


48

続いて、過適合の防止について説明いたします。

過適合と早期停止

- 過適合（オーバーフィッティング）
 - 最適化器は重みを適切に調整していくため、訓練データに対する損失コストはエポックが進行すると減少する
 - 一方、評価データの損失コストについてエポックが進むと上昇することがあり、訓練データに対して過適合となる
- 早期停止（early stopping）
 - 訓練データの任意の割合（本発表では20%）を検証データとして活用
 - kerasではコールバック機能により検証データの損失コストが一番小さいところで学習を終えることができ（早期停止）、これにより過適合を抑える



▶ 左記の例では100エポック辺りで過適合が始まっている

49

過適合（オーバーフィッティング）については、学習により訓練データに対する損失コストは、エポックが下がると減少していきませんが、一方評価データのロスについては、エポックが進むと上昇することがあります。これは下図の通りとなっております。

この状況をオーバーフィッティング、過適合と言いますが、評価データで過適合となっているということは、すなわち最終的なテストデータでも性能が悪いことが予測されます。

過適合を防止する一つの手法が、早期停止 (early stopping) となっております。左の図では、赤の訓練データが徐々に下がっているのですが、青の評価データについては、100 エポック辺りからロスが上昇していることが分かります。そのため、keras では、コールバックという機能によって、ロスが一番小さいところで学習を止めるという機能を実装していますので、これを早期停止と呼んでおります。

正則化（Regularization）

- 重みを計算する際に大きな重みに対して罰則を行うことにより、過度な重みを取らないようにする
- L1ノルムのLasso($p=1$)、L2ノルムのridge($p=2$)等がある。ridgeは重み減衰とも呼ばれる
- η はハイパーパラメータで罰則を与える強度を設定する

$$\mathcal{L}(\mathcal{D}, \hat{\mu}(\cdot); \eta) = \frac{1}{n} \sum_{i=1}^n 2N_i \left[\frac{\hat{\mu}(x_i, v_i)}{N_i} - 1 - \log \left(\frac{\hat{\mu}(x_i, v_i)}{N_i} \right) \right] + \eta \|\theta\|_p^p$$

※ θ は全ての重みパラメータの部分集合

```
output = design %>%  
  layer_dense(units=20, kernel_regularizer=regularizer_l2(0.00001),  
              activation='tanh', name='layer1') %>%  
  layer_dense(units=15, kernel_regularizer=regularizer_l2(0.00001),  
              activation='tanh', name='layer2') %>%  
  layer_dense(units=10, kernel_regularizer=regularizer_l2(0.00001),  
              activation='tanh', name='layer3') %>%  
  layer_dense(units=1, activation='exponential', name='output')
```

50

続いて正則化です。重みを計算する際に、大きな重みに対して罰則化を行うことにより、過度な重みを取らないような手法となっております。L1 ノルムの Lasso、L2 ノルムの ridge 等があります。式はこちらの通りですが、 η はハイパーパラメータで罰則を与える強度を設定するものとなっております。モデル上では ridge を使い、 η の強度を記載のとおり設定しております。

ドロップアウト (dropout)

- ドロップアウト率 $p \in [0,1)$ に対して、特定の層の出力結果のニューロンをランダムに確率 p で取り除く
- エポックのたびに取り除かれるニューロンは異なる
- ドロップアウトにより過適合を抑える効果がある

<適用前>

<適用後>

```

output = design %>%
  layer_dense(units=20, activation='tanh', name='layer1') %>%
  layer_dropout (rate = 0.05) %>%
  layer_dense(units=15, activation='tanh', name='layer2') %>%
  layer_dropout (rate = 0.05) %>%
  layer_dense(units=10, activation='tanh', name='layer3') %>%
  layer_dropout (rate = 0.05) %>%
  layer_dense(units=1, activation='exponential', name='output')
  
```

51

続いて、ドロップアウトについて説明します。ドロップアウトは、0 から 1 のドロップアウト率 p に対して、特定の層の出力結果のニューロンをランダムに確率 p で取り除きます。エポックのたびに取り除かれるニューロンが異なり、理論の説明は割愛いたしますが、過適合を抑える効果があるということが、よく知られております。

結果比較 (過適合抑制)

norm

ridge

dropout_0.01

dropout_0.05

dropout_0.10

> 下記 (es) は早期停止を表示
 > ridgeの早期停止が最良 (28.98406) だが、ridgeの最終エポック後の値 (29.05889) よりもノーマルの早期停止 (29.02806) の方が良い

<設定>

中間層数	K=3
ニューロン数	20,20,20
活性化関数	tanh
最適化器	nadam
バッチ数	5,000
エポック数	500

<結果>

no	model	run time	in-sample loss	out-of-sample loss	in-sample loss(es)	out-of-sample loss(es)
1	norm	375.9	29.07909	29.14862	29.25694	29.02806
2	ridge	392.9	29.36515	29.05889	29.35091	28.98406
3	dropout_0.01	682.6	29.13024	29.09521	29.30178	29.05422
4	dropout_0.05	627.2	29.39614	29.11966	29.42612	29.12019
5	dropout_0.10	692.9	29.58527	29.15203	29.56678	29.14627

- ✓ ridgeは効果が高く、早期停止は他の手法とも組み合わせて使え効果が高い
- ✓ ドロップアウトは一定過適合を抑えているもののそこまで損失コストが減少しない
- ✓ ADS Case study 02ではridgeやドロップアウトはあまり使わず早期停止が用いられている

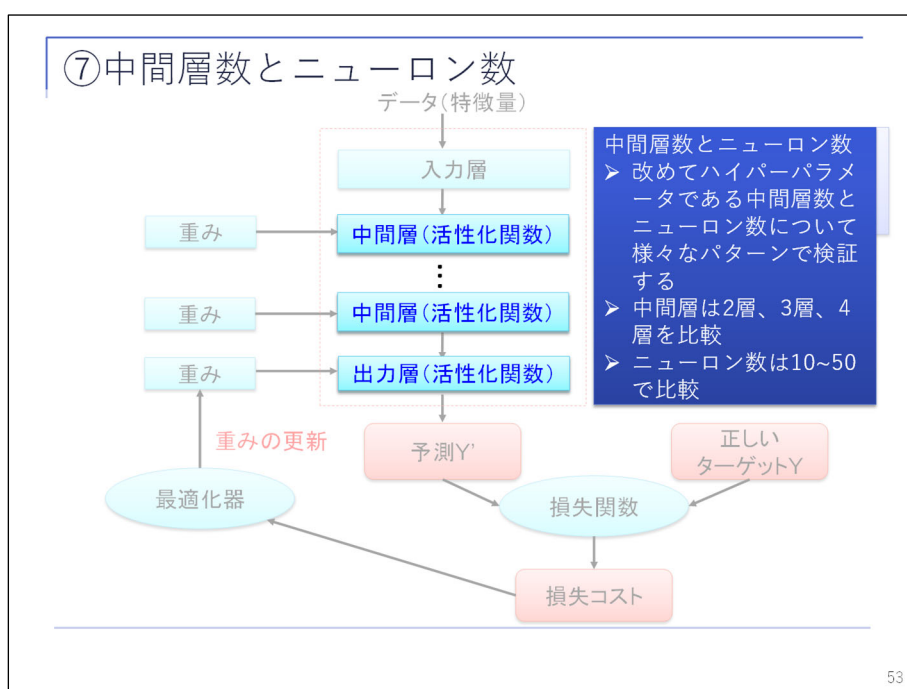
52

結果比較となっております。過適合防止の結果です。こちらのシミュレーションでは、過適

合しやすいように3層モデルで、ニューロン数が20、20、20という割と大きいニューロン数で実験しております。何もしない場合は、左上の図の通り評価データが過適合となっておりますが、他のモデルについては過適合が抑えられているということがうかがえます。

なお、ドロップアウト率については、1%、5%、10%の3パターンで行っています。また、早期停止につきましては、早めに停止するというだけです。ドロップアウトやリッジなど、他の手法と組み合わせて使うこともできます。

結果として、一番良かったものが青のridge、なおかつ、早期停止というものでしたが、赤のridgeの最終エポック29.05889というものよりも、ノーマルの早期停止29.02806というものが良かったので、早期停止というものは、かなり効果が高いということがうかがえます。ドロップアウトについては、それほどいい効果が得られておりませんでした。ADS case study 02でもridgeやドロップアウトはあまり用いられずに、早期停止を用いています。



続いて、ハイパーパラメータである中間層数とニューロン数について考察しております。

比較結果（中間層・ニューロン数）①

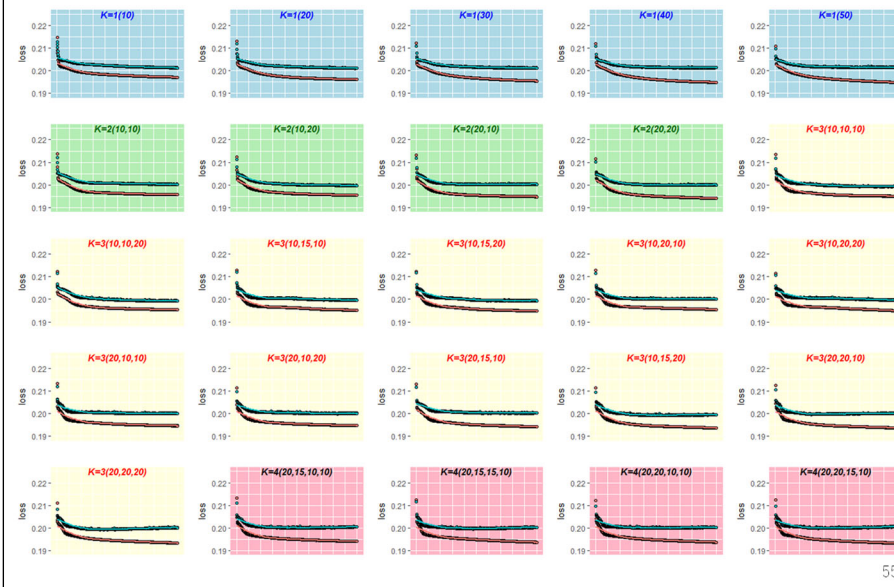
no	K	q1	q2	q3	q4	パラメータ	time	in_loss	out_loss	in_loss(es)	out_loss(es)	rank	ave_rank
1	1	10				411	298.4	29.75532	29.30460	29.76072	29.32338	23	23
2	1	20				821	316.3	29.58250	29.30864	29.58353	29.31389	22	
3	1	30				1,231	328.5	29.48289	29.29253	29.48091	29.28735	21	
4	1	40				1,641	339.1	29.39302	29.41822	29.38061	29.39558	25	
5	1	50				2,051	423.6	29.39649	29.39712	29.38458	29.37462	24	
6	2	10	10			521	273.9	29.49149	29.11103	29.49785	29.11636	16	13
7	2	10	20			641	283.5	29.41342	29.07567	29.42098	29.07399	11	
8	2	20	10			1,021	276.0	29.34589	29.15524	29.38931	29.11955	18	
9	2	20	20			1,241	292.3	29.20883	29.07036	29.32759	29.03536	6	
10	3	10	10	10		631	282.2	29.33740	29.01860	29.33901	29.00262	3	
11	3	10	10	20		751	309.6	29.38861	29.06540	29.39431	29.05027	10	9
12	3	10	15	10		736	292.5	29.37502	29.03616	29.38072	29.04044	7	
13	3	10	15	20		906	309.7	29.28924	28.94131	29.29956	28.94168	2	
14	3	10	20	10		841	286.6	29.43032	29.12131	29.44296	29.11951	17	
15	3	10	20	20		1,061	314.7	29.32569	29.02546	29.32569	29.02546	4	
16	3	20	10	10		1,131	286.3	29.29522	29.11116	29.34650	29.10098	13	
17	3	20	10	20		1,251	306.6	29.31420	29.24616	29.32595	29.18877	20	
18	3	20	15	10		1,286	300.2	29.22518	29.13526	29.25766	29.11479	15	
19	3	20	15	20		1,456	316.8	29.11494	29.04273	29.24587	28.93724	1	
20	3	20	20	10		1,441	303.0	29.15014	29.16802	29.40258	29.04102	8	
21	3	20	20	20		1,661	333.6	29.07909	29.14862	29.25694	29.02806	5	
22	4	20	15	10	10	1,396	330.6	29.21595	29.25710	29.33764	29.12375	19	14
23	4	20	15	15	10	1,526	339.5	29.14837	29.18395	29.33404	29.04946	9	
24	4	20	20	10	10	1,551	333.5	29.15322	29.21407	29.29384	29.11355	14	
25	4	20	20	15	10	1,706	339.5	29.09449	29.26403	29.32795	29.09302	12	

※これまでの結果により最適化器：nadam、活性化関数：tanh、バッチサイズ：5,000、エポック数：500で実行

54

まず、色ごとにK（中間層）が1、2、3、4というように分かれています。それとニューロンの数を多数並べて、全部で25モデルを計算して並べてみた結果となっております。ちなみに、これを全部実施する場合に約130分かかっております。最良のモデルは青で記載しておりますが、中間層3のエポック数が20、15、20というモデルとなっております。

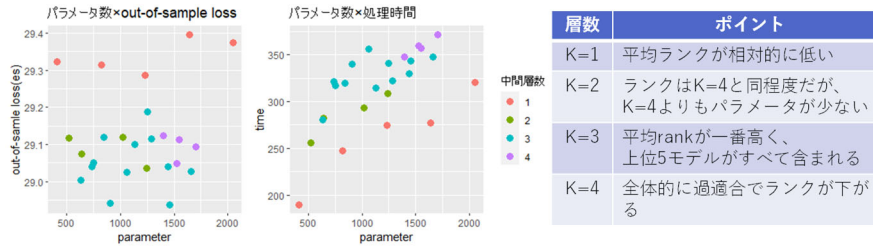
比較結果（中間層・ニューロン数）②



55

色分けで見たものが、こちらとなっておりますが、ピンクの中間層4つが過適合気味になっているということが分かります。

中間層・ニューロン数による影響考察



- ✓ 同程度のパラメータ数の場合、階層が深くなるほど処理時間を要する
- ✓ ベストモデルはK=3のニューロン数 (20,15,20)
- ✓ ADS Case study 02ではK=3の (20,15,10) がベストモデルとなり推奨

※ADS Case study 02より

- 理論的にはシャロウ・ネットワークで十分であるが、実際にはディープ・ネットワークの方が回帰問題のパフォーマンスが優れている。特に3つの中間層は、正確なモデルと高速な収束につながる。
- 初めの中間層は大きくする必要があり、小さい場合は圧縮で失われる情報が多い。(20,15,10)は情報が連続的に圧縮され、適切な戦略となる可能性がある。
- (20,10,20) はPCAと同様に機能するボトルネックネットワークと見なすことができる。

56

結果考察です。一番左上図は縦軸がロスですので、中間層が1つであると、性能があまり良くないということが分かります。ADS case study 02 から引用させていただきますが、

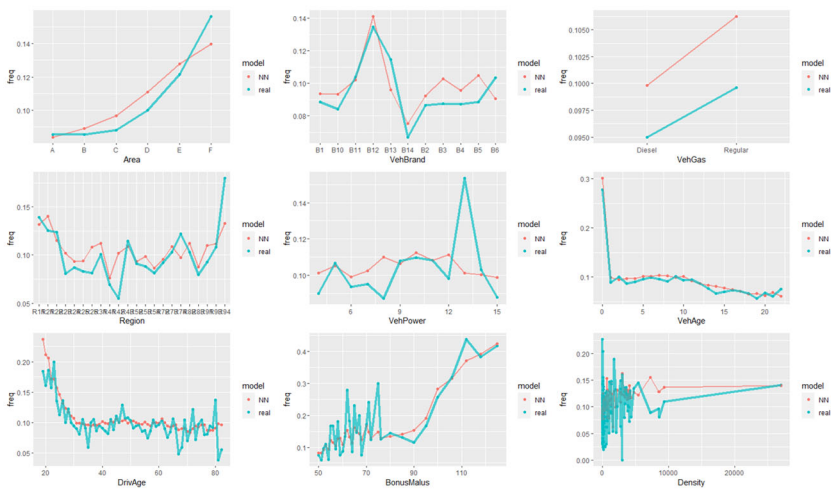
「理論的にはシャロウ・ネットワークで十分であるが、実際にはディープ・ネットワークの方が回帰のパフォーマンスが優れている。特に3つの中間層は、正確なモデルと高速な収束につながる。

初めの中間層は大きくする必要があり、小さい場合は圧縮で失われる情報が多い。20、15、10というものが連続的に圧縮され、適切な戦略である」

というように考えられております。

テストデータによる頻度比較

※ポリシー100以上区分のみ



NN : K=3(20,15,10) モデル real : 実績値

57

続いて、テストデータによる頻度の比較をしております。こちらについては、値ごとの平均事故頻度とモデルによる事故頻度を比較しております。あまりに小さい区分は分かりにくいので

で、ポリシーが 100 以上のところを表示しておりますが、ニューラルネットによって、割と全体的に傾向を捉えられているものと思われます。

その他① Blending (network) model

- Blending (network) model predictions
 - ネットワーク複雑になると、通常、複数の同等に良いモデルが得られ、最適なネットワーク構造を見つけること探すことは賢明なやり方ではない。ネットワークを固定したとしても、異なるパラメータから複数の同等に良いモデルが得られる。
 - 既に得られたモデルのブレンドにより、より良い結果が得られることがある
 - 複数レイヤーに対して実施したシミュレーションの中間層K=3の12モデル (no.10-no.21)を用いたブレンド結果は下記の通り、損失コストの値が最小となった

個別モデル(12モデル)

Blending model

個別モデルのClaimNbの予測の合算値/12

no	model	in-sample loss(es)	out-of-sample loss(es)
1	blend	29.17652	28.89612
2	K=3(20,15,20)	29.24587	28.93724

58

続いて、手法その他として「Blending」というものを紹介させていただきます。ネットワークが複雑になると、通常、同等に良いモデルというものが幾つか出てきますが、その良いモデルを探すということが賢明なやり方ではないということが紹介されております。既に得られたモデルのブレンドにより、より良い結果が得られることがあります。先ほどの中間層 3 つの 12 モデルの予測値の合算を、単純に 12 で割ったものですが、そうしますと、これまでの最良結果よりも更に小さい 28.89612 という値が出てきて、最小を更新した結果となります。

その他② 埋込層 (Embedding Layer)

- 埋込層は自然言語処理の分野でよく用いられ、カテゴリカル特徴量を任意の次元にマッピングする
- 埋込層を活用することにより、ダミーコーディングよりも次元が圧縮され、また、特徴量の関係性から追加の洞察を得られることができる
- VehBrandとRegionをそれぞれ 2 次元の埋込層に変換

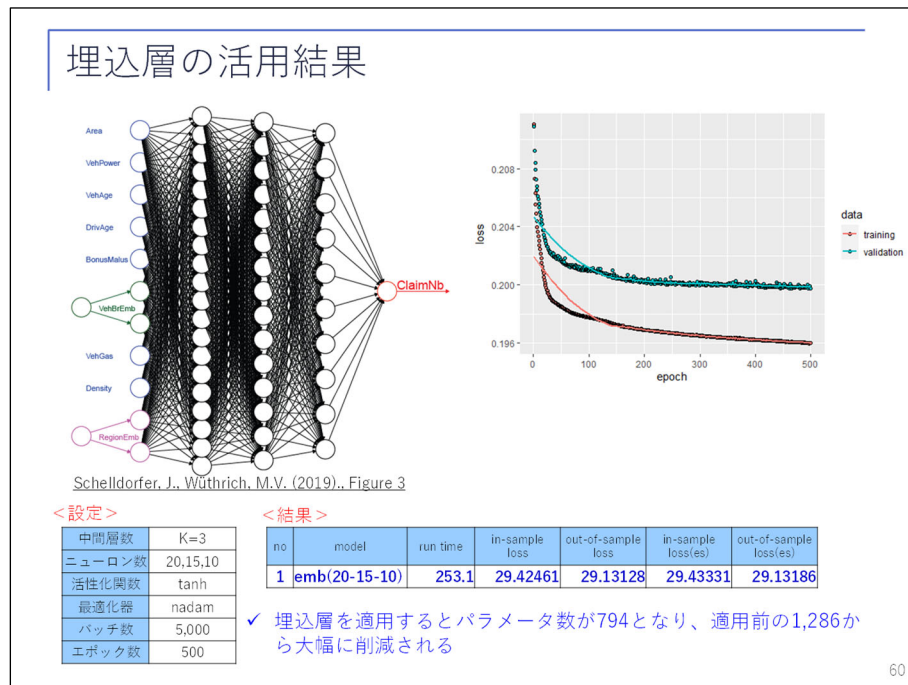
VehBrand

Region

59

続いて、埋込層について説明させていただきます。埋込層は、自然言語処理の分野でよく用

いられ、カテゴリカル特徴量を任意に次元マッピングするものとなっております。埋込層を活用することにより、ダミーコーディングよりも次元が圧縮されまして、特徴量の関係から追加の洞察を得ることができます。VehBrandとRegionについて、それぞれ2次元の埋込層に変換したものがこちらとなっておりますが、VehBrandですとB12が右上にあるように特徴が捉えられております。



続いて、モデルについては、左図となっております。入力層の特徴量が大幅に減少しております。結果を示しておりますが、残念ながら先ほどの Blending モデルよりはよろしくなかったものの、一定の成果が出ているということが分かります。

前半部分は以上で終了となりますが、ニューラルネットワークを使うことにより、予測精度の高いモデルを構築することができるということを、ご覧いただけたかと思えます。

前半の発表については、以上となります。質問をいただいているものですが。

「損失コストとしてポアソン逸脱度を使用されておりますが、当該手法を選択された理由、望ましい理由、考えられる他の手法等がありましたら、ご教授いただければ」

ということです。

損失コストの定義

- 損失コスト
 - 真の値と予測値のズレを表し小さいほど望ましい
 - ハイパーパラメータの調整やNNの学習に用いる

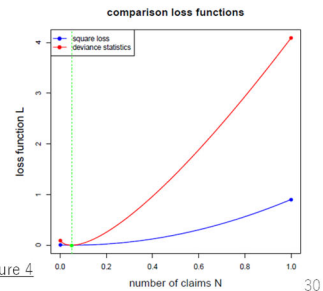
- ポアソン逸脱度 (Poisson deviance loss function)
 - 損失コストとしてポアソン逸脱度を用いる (※)

$$\mathcal{L}(\mathcal{D}, \hat{\mu}(\cdot)) = \frac{1}{n} \sum_{i=1}^n 2N_i \left[\frac{\hat{\mu}(x_i, v_i)}{N_i} - 1 - \log \left(\frac{\hat{\mu}(x_i, v_i)}{N_i} \right) \right]$$

※kerasでは $\hat{\mu}(x_i, v_i)$ に依存しない箇所を省略しており、以降グラフ上では損失コストの値が異なる

$$\mathcal{L}^{\text{Keras}}(\mathcal{D}, \hat{\mu}(\cdot)) = \frac{1}{n} \sum_{i=1}^n \hat{\mu}(x_i, v_i) - N_i \log \hat{\mu}(x_i, v_i)$$

- 回帰ではMSEが用いられることも多いが、予測値が小さいカウントデータ予測の場合はポアソン逸脱度が望ましい
- 【右図】ポアソン分布仮定の下、平均値が0.05 (緑) のとき、MSE (青) に比べてポアソン逸脱度 (赤) の方が損失コストが高くなり感応的



ADS Case study 02, Figure 4

30

こちらについては、先ほど若干説明させていただきましたが、通常ですと Mean Squared Error (MSE) など、回帰ですとそのようなものもよく使われるかと思えます。ただ、こちらの case study 02 の論文に記載があるのですが、ポアソン回帰のような頻度データで、かなり頻度が少ない場合の損失コストについては、なるべくポアソン逸脱度のようなものが望ましいとなっております。先ほどのグラフを再度説明させていただきますと、緑が事故頻度 5% の点となっており、それに対して loss function が縦軸となっております。MSE に比べて、赤のポアソン逸脱度の方のロスが高く出る。いわば、ロスに対して感応的になるため、モデルを調整しやすくなるということになります。論文においては、その他の損失コストについても多少触れておりますが、比較の結果このような頻度モデルについてはポアソン逸脱度が望ましい、ということが紹介されておりました。

前半については以上で終了させていただきます。

【司会】 川上さん、ありがとうございます。それでは渡辺さん、よろしくお願いいたします。

【渡辺】 はい。それでは始めさせていただきます。

4. CANN

(Combined Actuarial and Neural Network)

61

後半を担当いたします、あいおいニッセイ同和損保の渡辺です。よろしくお願いいたします。

前半では、ニューラルネットワークによる事故頻度分析の例を通じて、ニューラルネットワークで予測精度が高いモデルを構築できるということや、ニューラルネットワークでモデリングを行うにあたり検討する必要がある要素について、説明してきました。後半は、伝統的なアクチュアリー業務で使うにあたってどうするのか、というところを考えてみたいと思います。

伝統的アクチュアリー業務での利用

- NNにより、予測性能の高いモデルを構築できる

- 特に、複雑な問題について既存の手法よりも優れたパフォーマンスを示す
 - クレーム構造が変化するなかでの支払備金見積もりなど

- ただし、実用にあたっては以下の問題がある (*1)
 - 「最適な」モデルが一意に定まらない
 - モデルや結果の説明が難しい

*1 Wüthrich (2019)

62

前半の最初で紹介した SOA レポートにもありましたけれども、ニューラルネットワークを使うと予測性能が高いモデルを構築することができる、特にクレーム構造が変化する中での備金

の見積もりといったような複雑な問題については、既存の手法よりも優れたパフォーマンスを示すことがあると、そのようなことが言われています。ただ、実用にあたっては、いろいろと問題があり、例えば、最適なモデルが一つに決まらない、あるいは、モデルや結果の説明が難しい、そのような問題が指摘されているところです。

ここで「伝統的」とことわっているのは、保険会社の中でも、利用する分野によってはここに挙げたような問題が制約にならないこともあるためです。

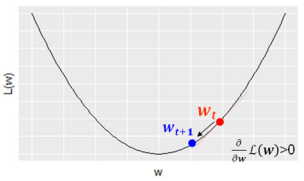
まず、「最適なモデルが一つに決まらない」というところからお話をしようと思います。

最適化器 (optimizer)

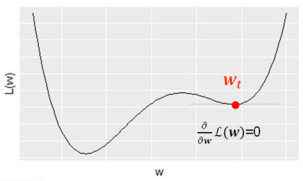
- 勾配降下法 (GDM: gradient descent methods)
 - 最適化器は損失コストを最小にするようにパラメータを更新する (学習)
 - 単に損失コストを最小にするのみならず、なるべく効率よく損失の最小化に到達するアルゴリズムが望ましい
- 最急降下法
 - 損失関数を重みの関数とし、重みベクトルの微分を用いて重みを更新

$$w_{t+1} \leftarrow w_t - \alpha \frac{\partial}{\partial w} \mathcal{L}(w)$$

w_t : t回目のエポックの重みベクトル、 $\mathcal{L}(w)$: 重みベクトルを引数とする損失コスト、 α : 学習率



勾配 (微分係数) と学習率 α により重みを更新



<課題>
最小値ではなく極小値に陥ったときに更新がストップする
⇒SGD等の手法により解決

39

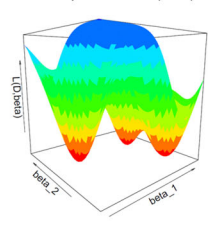
前半の 39 ページに戻ります。ここで、ニューラルネットワークの学習は損失関数を小さくするように進められる、ということをお話ししています。ここで、損失関数の値とパラメータの関係が、この左側の図のような関係にあれば、最小値を求めることは簡単です。

解の非一意性

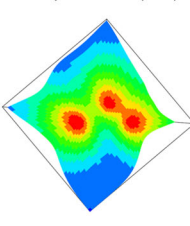
- NNの損失関数は非凸

Wüthrich (2019), Figure 4

in-sample deviance loss (view 1)



in-sample deviance loss (view 2)


- 実際には、過適合を避けるため早期停止を行う
 - 例えば上図緑色の領域で停止
 - 唯一の最適解の代わりに、無数の「十分に良い解」を得る
 - 解は初期値に依存

63

ところが、ニューラルネットワークの損失関数は、p. 63 のこの図のように、もっと複雑な関係になっています。左側の図が横から見た図で、右側が上から見た図だと思ってください。水平方向の縦横はパラメータが動く範囲で、垂直方向は損失関数の値です。赤いところは小さい値です。ニューラルネットワークの学習は、どこか適当な初期値から出発し、損失関数が小さくなるように進んでいきます。最適値に至ればいいのですが、実際には、どこから出発するかにより、最適値に至らないかもしれません。あるいは、過適合を避けるために早期停止する場合には、最適値まで行かず、例えば、この緑色のところで止まってしまうかもしれません。そうすると、ニューラルネットワークで得られる解は、唯一の最適解ではなく、無数にある「十分に良い解」の一つでしかない、どこで止まるのかは初期値によって違う、ということになります。

このように、一つの問題に対して同じような精度を持つ多数のモデルが存在するという状況を指して、「羅生門効果」と呼ぶようです。「羅生門」という名前は、黒澤映画のタイトルから来ているようで、この映画は、一つの事件について複数の証言者がそれぞれ違った証言をし、真相は結局分からないというお話です。この羅生門効果を前提として、その度合いを表す「羅生門レシオ」というモデリング手法に固有の値を定義し、これに基づいてモデル選択をするという研究もあるようです。

前半の最後の方で、複数の同程度に良いモデルをブレンドするというやり方を紹介していますが、それ以外に、対象とする問題に関する背景知識を基にして制約条件を追加する、あるいは、入力データにノイズを与えた場合のモデルの振る舞いを分析することによりモデルを絞り込んでいく、そのようなアプローチもあるかと思えます。

文字認識のように「当たればいい」というような用途ではなく、例えばプライシングといった用途に使う場合には、モデルの選択にあたり、このあとお話するような解釈手法を使って、「モデルによる予測が用途に見合っているか」というようなところの評価も必要になってくるように思えます。

機械学習の解釈手法

- 機械学習の解釈手法の分類
 - Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (*1)による分類

本質的な解釈	(正則化) GLM、GAM、決定木等、本質的に解釈可能なモデルを用いる
後付けの解釈	モデルを学習した後に解釈するための手法
モデル非依存	大域的：Permutation Importance, Partial Dependence Plot等 局所的：SHAP (SHapley Additive exPlanations) 等
モデル固有	NN固有：特徴量の可視化、敵対的サンプル、概念の抽出 等

*1 Molnar (2020)

64

次に、解釈手法についてお話します。「モデル結果の説明が難しい」といいましたが、全

く説明できないわけではなく、いろいろな手法が提案されています。冒頭に掲げている『Interpretable Machine Learning』という本は、英語版も日本語版も Web で公開されていて、大変参考になるものです。その中で解釈手法について、この表のように分類されています。

「本質的な解釈」とは、そもそも解釈しやすいような手法を使うということです。ニューラルネットワークは、下の「後付けの解釈」に頼らざるを得ないのですが、この中でも「モデルに依存しない手法」と「モデルに依存する手法」があります。以下、順に、まずモデルによらない手法についてお話をし、その後、ニューラルネットワーク固有の手法について、お話をしようと思います。

モデルによらない解釈手法 (*1)

- 例：埋込層を用いたモデル (p.60)
- Rのpackage DALEXを利用
 - NNを含む機械学習モデルの学習結果の分析が可能
 - 一般的な解釈手法の多くに対応
 - explainerと呼ばれるオブジェクトを作成

```
library(DALEX)
dal_nn <- DALEX::explain(
  model=nn.model, ← Kerasで学習したモデル
  data=test, ← テストデータ
  y=test$ClaimNb, ← 目的変数のデータ
  predict_function=predict.nn, ← modelとdataから予測値を返す関数
  residual_function=residual.pois, ← 予測値と観測値から損失を返す関数
  label="NN (20, 15, 10)"
)
```

- explainerを解釈手法に対応する関数に与えて実行

*1 Biecek, P., & Burzykowski, T. (2021)

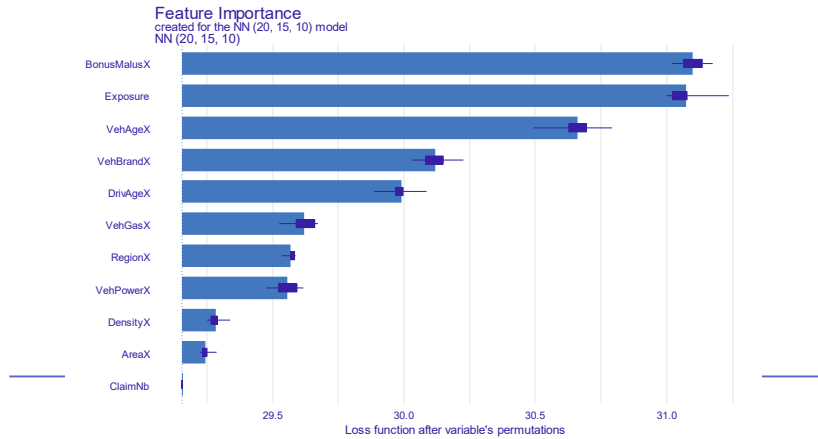
まず、モデルによらない解釈手法ですが、これ自体は、2019 年年次大会のプレゼンテーションで、AFIR 関連研究会が大変良い説明をされていたので、重ねて説明をすることは避け、ここでは、前半の最後に取り上げた埋込層のあるモデルを例にして、具体的な分析ツールやその使い方について、お話をしようと思います。

Rで分析を行うためのパッケージはいろいろとありますが、ここでは DALEX というパッケージを使っています。DALEX での分析にあたっては、ここに掲載したコードのようにして、まずモデルやデータについての情報を持つ explainer というオブジェクトを作り、これをいろいろな関数に与えてプロットを作成します。

大域的な説明 (1)

■ 特徴量の重要度 (Feature Importance)

```
fi.nn<-feature_importance(dal_nn,loss_function=loss.pois,type="raw")  
plot(fi.nn)
```



66

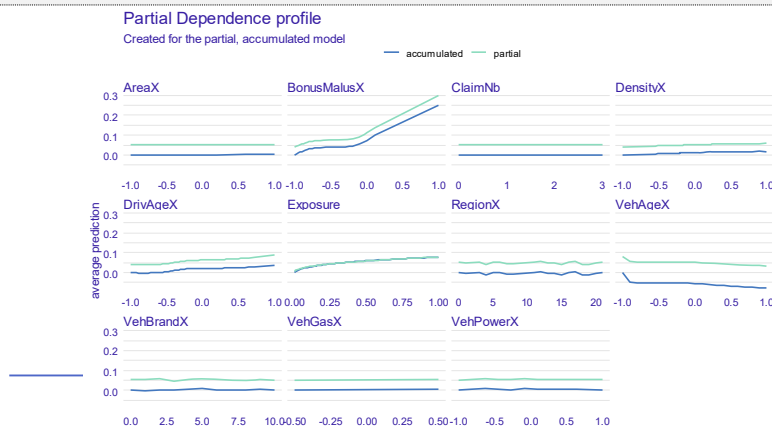
モデルによらない解釈手法には、データ全体を対象にしてモデルの振る舞いを見る「大域的な説明」と、特定のデータについて評価する「局所的な説明」がありますが、まず大域的な説明に分類される手法について紹介します。

最初のプロットは、「特徴量の重要度」です。この図から、どの特徴量が最終結果に影響を与えているのか、ということを見て取ることができます。この例では、BonusMalusX、Exposureなどが結果に大きく影響を与えている、ということが分かるかと思えます。逆に、下の方にあるDensityX（人口密度）やAreaXはそれほど影響していない、ということが分かります。

大域的な説明 (2)

■ 特徴量と目的変数の関係 (Partial Dependence, Accumulated-Local Profiles)

```
pdp.nn.i <- partial_dependence(dal_nn) : pdp.nn.i$'_label_' <- "partial"  
ale.nn.i <- accumulated_dependence(dal_nn) : ale.nn.i$'_label_' <- "accumulated"  
plot(pdp.nn.i, ale.nn.i, color="'_label_'")
```



67

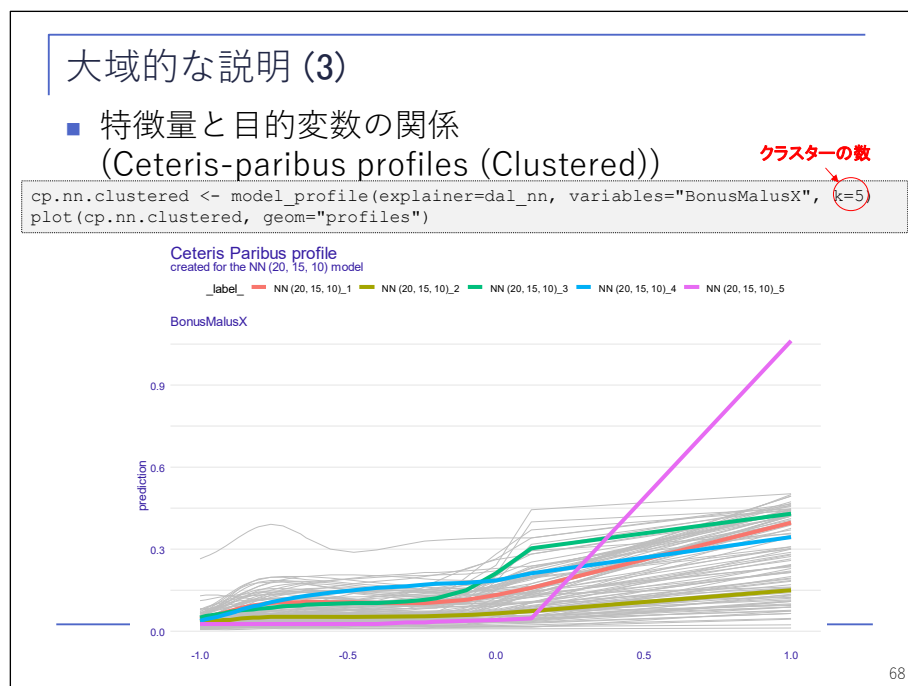
次に、それぞれの特徴量について、値の変化が結果にどのような影響を与えるのかを見ていこうと思います。ここでは2つのグラフを載せています。1つは「Partial Dependence Plot」、もう1つは「Accumulated-Local Effects Profiles」または「ALE Plot」と呼ばれているもの

です。それぞれ、薄い水色と濃い青のグラフです。大体どちらも同じような傾向を示しています。

Partial Dependence Plot は、対象とするデータの中で、特定の特徴量の値を動かした場合に予測値がどのように変化するか、ということプロットしています。ただし、動かすときに、他の特徴量との関係を考慮しませんので、例えば BonusMalusX で、非常に若いにも関わらず何年も無事故でなければ到達しないような等級に到達している人、というデータもできてしまっており、依存関係が高い特徴量があるデータセットには、うまくいかない可能性があります。

この問題を解決するものが、ALE Plot です。特徴量を取る区間ごとに区切って、その区間ごとに、近い値のデータを使って予測値の変化を求めて累積して作ります。この図では、どちらも同じような傾向となっていますが、これを見ることで、大体の傾向がつかめるかと思えます。

例えば BonusMalusX であれば、値が高ければ高いほど事故頻度は高くなり、しかも、直線の関係ではなく、途中から折れ曲がっている、という傾向が見て取れます。



次に、「Ceteris-paribus profiles」を紹介します。前のページで「Partial Dependence Plot を作る際には、データの中のある特徴量の値を動かす」というお話をしました。それをデータ全体で平均したものが Partial Dependence Plot ですが、Ceteris-paribus profiles は、平均する前のサンプルごとの値を示したものです。薄いグレーの線はそれにあたります。Ceteris-paribus は「他の条件が同じならば」という意味ですが、他の条件が同じであった場合に、特徴量を動かしたときの値の変化を示したものが、Ceteris-paribus profiles です。

グレーの線だけでは様子が分かりにくいので、これをクラスタリングして5つの区分に分けて、クラスターごとにプロットしたものも示しています。これを見ると、一つだけ飛び抜けて変なものがあります。このようなものを見ると、実は BonusMalusX は、他の変数との交互作用を考慮する必要があるのではないか、というところが見えるかと思えます。

局所的な説明 (1)

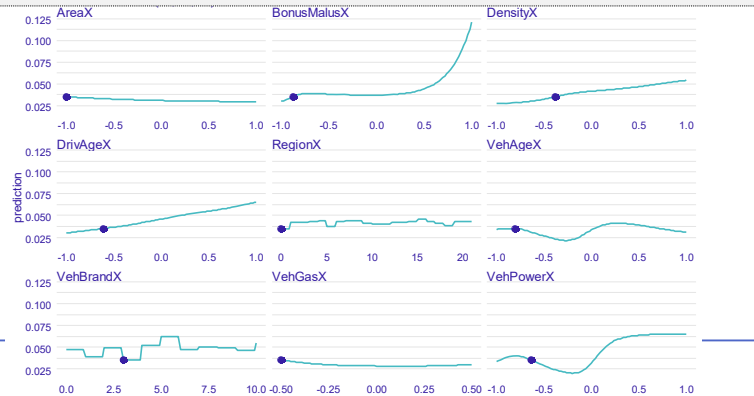
■ 特定のデータについての予測に対する説明を考える

D pol	ClaimNb	Exposure	Area	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Density	Region	予測値
42	1	0.77	A	6	2	32	56	B12	Diesel	23	R24	0.035142

■ Ceteris-paribus Profiles

テストデータの2番目のレコード

```
cp.nn <- predict_profile(explainer=dal_nn, new_observation=test[2, ])
plot(cp.nn)
```



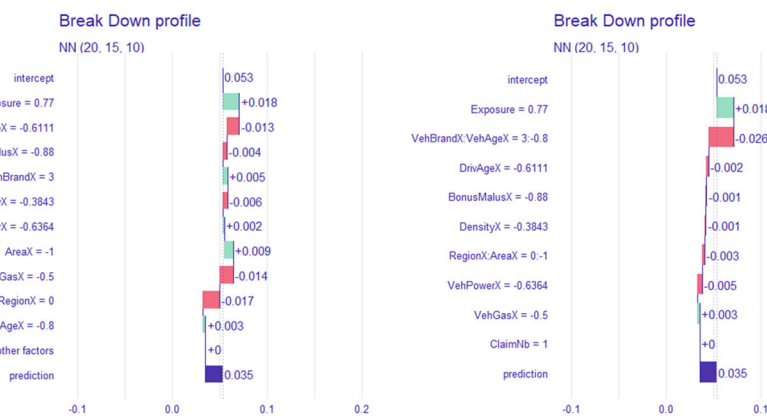
69

ここからは、局所的な説明に分類される分析手法を紹介します。例として、ID ナンバー (IDpol) 42 というデータについて見てみましょう。前のページと同じ Ceteris-paribus profiles が出てきましたが、このページの図は、前のページの薄いグレーの線のうち一つのサンプルを抜き出してきたものだと思います。●は実際のデータの値を表しており、この値が動くことによって、結果、事故頻度がどのように動くのか、ということを示しています。

局所的な説明 (2)

■ Break-down Profile

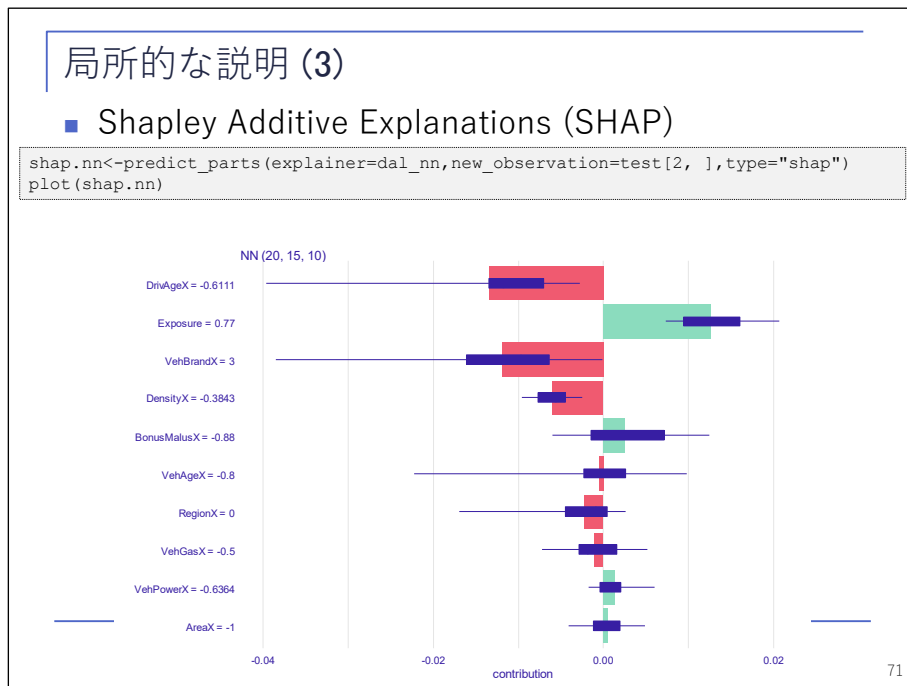
```
bd.nn <- predict_parts(explainer=dal_nn, new_observation=test[2, ], type="break_down")
bdi.nn <- predict_parts(explainer=dal_nn, new_observation=test[2, ],
  type="break_down_interactions")
plot(bd.nn) + plot(bdi.nn) # "+" を使うためには library(patchwork) が必要
```



70

続いて、「Break-down Profile」です。この図は、一番下に、このサンプルについての事故頻度の値が出ています。このモデルでは 0.035 です。一番上の切片項は全体の平均で、これに対し要因ごとにプラス・マイナスの効果を積み上げられていった結果として事故頻度が 0.035 になる、ということを示しています。左側はそれぞれの変数を単独で見た場合のグラフで、右側は交互作用を考慮した場合のグラフです。右側を見ると、上の 2 つで大半説明できている、と

いう図になっています。



局所的な説明の最後は「SHAP」です。前のページの Break-down は非常に見やすいのですが、実は、特徴量を並べる順序により見え方が変わってしまうという欠点があり、その欠点を避けるために、全体を平均して作ったあげたものが SHAP です。SHAP も Break-down と同様に、IDpol142 というサンプルについて、どの変数がどのように効いているのかということを表しています。

NN固有の解釈手法 (1) (*1)

- 出力層に注目
 - 活性化関数 バイアス 重み 最後の中間層の活性化値
$$\mathbb{E}[Y_i] = h^{-1} \left(o_i + \langle \beta^{(d+1)}, z_i \rangle \right) \quad h^{-1}(\cdot) = \exp(\cdot)$$

$$z_i = \left(z^{(d)} \circ \dots \circ z^{(1)} \right) \quad \text{説明変数}$$
- 損失関数: $\mathcal{L}(\mathcal{D}, \beta) = \frac{2}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} Y_i \left[\frac{\mathbb{E}[Y_i]}{Y_i} - 1 - \log \frac{\mathbb{E}[Y_i]}{Y_i} \right]$ (ポアソン逸脱度)

- これは以下のGLMに相当する
 - リンク関数 回帰係数 説明変数
$$\mathbb{E}[Y_i] = h^{-1} \left(o_i + \langle \beta, \mathbf{x}_i \rangle \right) \quad h^{-1}(\cdot) = \exp(\cdot)$$

$$Y_i \sim \text{Poisson}(\mathbb{E}[Y_i])$$
- 実際には x_i の代わりに前処理を加えた z_i を利用

*1 Wüthrich (2019)

72

次に、ニューラルネットワーク固有の説明手法についてお話したいと思います。いろいろな手法がありますが、ここでは特に出力層に注目したいと思います。出力層は、バイアスと重みの線形結合、すなわちネットワーク予測子を、活性化関数で変換したものが、全体のクレーム頻度の期待値 $E[Y]$ となっている、という構造になっています。

この構造は、下の GLM と似た形になっています。そう考えると、出力層で使用する z の値（最後の中間層の活性化値）が GLM の説明変数に相当すると考えることができます。

NN固有の解釈手法 (2)

- GLMとNNの違いは、特徴量の前処理をアクチュアリーが行うかアルゴリズム内で行うかの差

GLM : $x_i \xrightarrow{\text{アクチュアリーによる前処理}} z_i \xrightarrow{\text{GLM}} E[Y_i] = \exp(o_i + \langle \beta, z_i \rangle)$

NN : $x_i \xrightarrow{\text{NN(第1~K中間層による変換)}} z_i \xrightarrow{\text{NN(出力層)}} E[Y_i] = \exp(o_i + \langle \beta^{K+1}, z_i \rangle)$

- NNは中間層で $x_i \mapsto z_i$ の対応関係を学習している

図で示しますと、GLM であれば、元の説明変数に対し、アクチュアリーが何か前処理を行って適当な説明変数を作り、これを使って予測しますが、ニューラルネットワークであれば、この前処理をアルゴリズムの中で自動的に行ってくれるということです。この点に注目し、ニューラルネットワークが前処理として中間層でどのような対応関係を学習したのかを見れば、モデルの中身が分かるのではないかと、というのが、ここで紹介する手法のアイデアです。

NN固有の解釈手法 (3)

- NNの中間層で何が学習されたのか
 - 最後の中間層の活性化値に注目
 - 特徴量毎に取り得る値のそれぞれについて平均活性化値を計算

例 : DrivAge

$$\bar{z}_j(a) = \frac{1}{|\{i : \text{DrivAge}_i = a\}|} \sum_{i: \text{DrivAge}_i = a} \tilde{z}_{j,i}$$

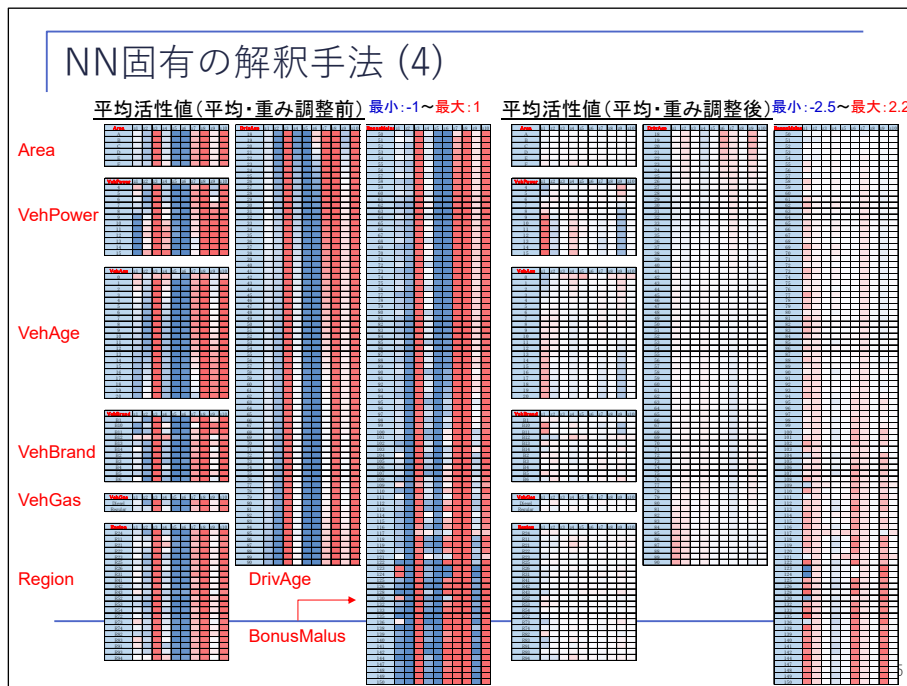
最後の中間層の活性化値
($\tilde{z}_{j,i}$)
DrivAge=aである標本についての平均

- 比較のため、特徴量毎に平均を控除し、出力層の重みを乗じる
- Model="emb (20-15-10)" (p.60) についての計算結果は次頁のとおり

次に、中間層で学習した内容を見るための方法を考えます。

ここでは、ある特徴量に注目し、特徴量の値ごとに、最後の中間層の活性化値 z が平均してどの程度の値になっているのかを調べます。

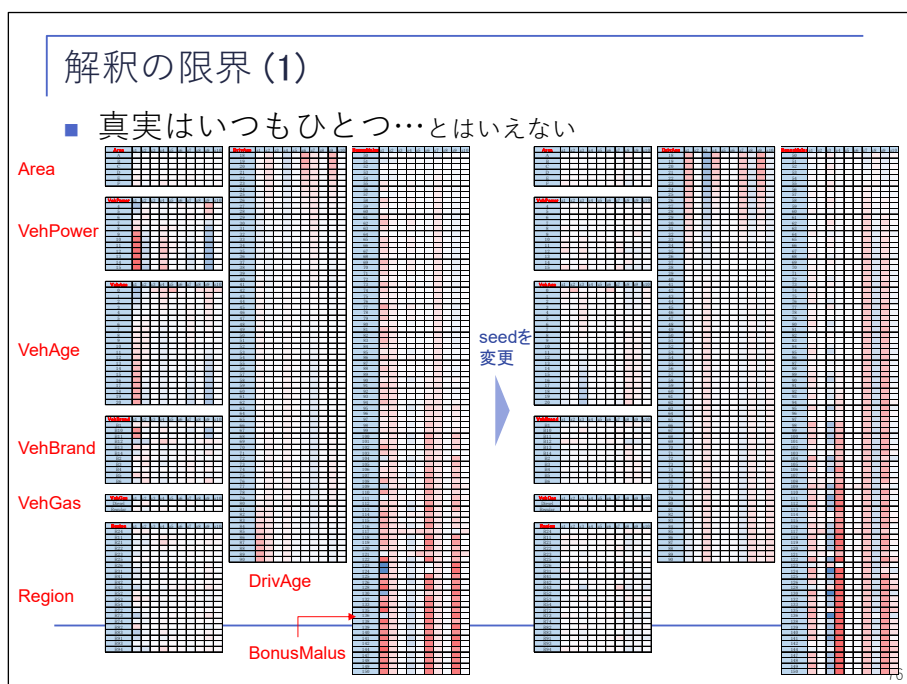
実際に計算してみた結果が、次のページの図です。



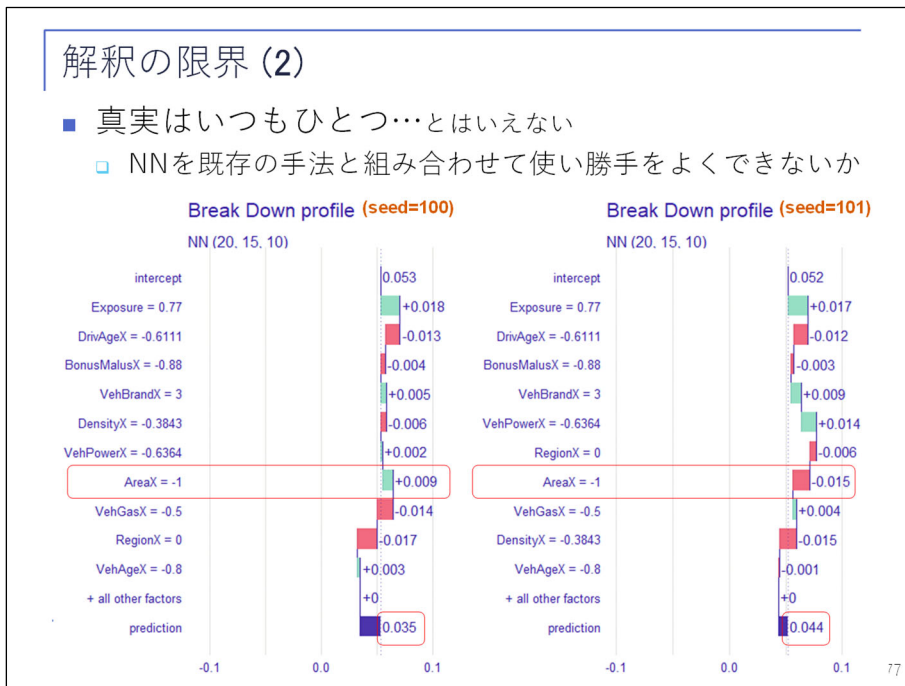
非常に細かい図ですが、左側は、前のページの平均を計算した結果です。右側は、比較のために、ニューロンごとの重みを掛けて平均を引いたものを示しています。特徴量ごとに図が分かれており、それぞれ、縦方向にその特徴量を取り得る値、横方向に 10 個のニューロンが並んでいます。青いセルが小さな値、赤いセルが大きな値、白いセルが 0 に近い値です。

例えば、右側の図の中央の列は、運転者年齢を表しています。これを見ると、左端のニューロンでは、下の方、運転者年齢が高いところに赤い色がついていますので、まず高齢部分の事故頻度の状況を表現しているのではないかと見えて取れます。

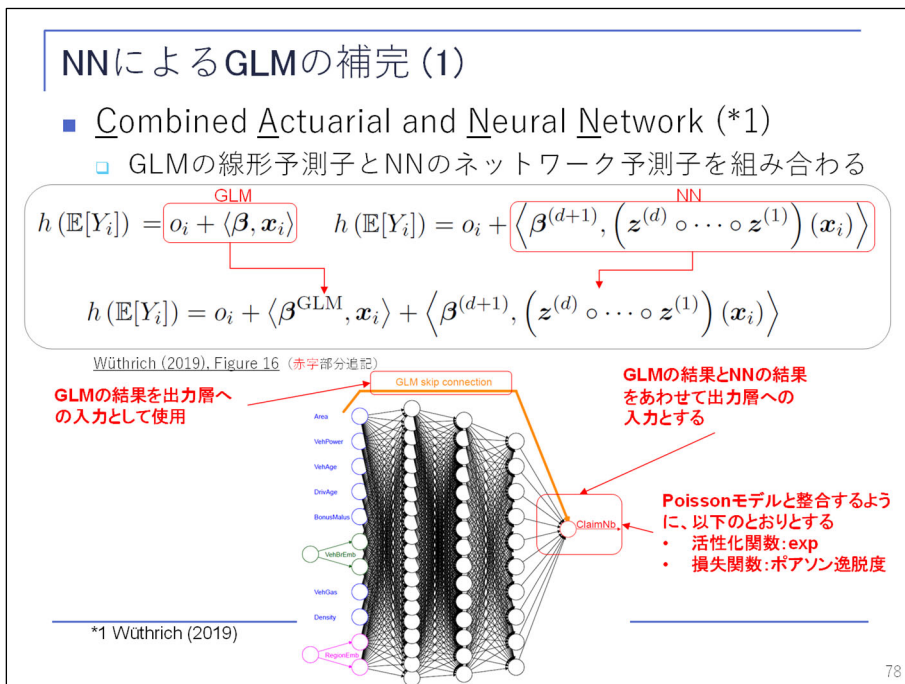
それでは「この図を見れば分かるのか」というと、実はそれほど簡単ではありません。次のページをご覧ください。



左側は、前のページ右側の図と同じものです。右側は、乱数を変えてモデルを当てはめ直して作成したものです。結構違いますね。このように、乱数の seed を変えることで、得られる説明が結構変わってしまいます。



次のページでは、乱数を変えて作成した Break Down profile を示しています。乱数を変えることによりモデルが変わったので、最後の予測値、期待値も変わっていますし、例えば「AreaX」の影響を見ると、符号も変わっています。このように乱数を変えると、結果も変わってくるし、説明も変わってきます。



これまでお話したような手法を使うことで、ある程度の説明はできるかもしれませんが、他にもいろいろな説明手法がありますが、GLMと同じような説明ができるわけではありません。だからといって「ニューラルネットワークは使えない」というわけではなく、「既存の手法

と組み合わせることで、うまく使う方法があるのではないか」ということが提案されています。その一つが、ここに示す「CANN」(Combined Actuarial and Neural Network) という手法です。GLMの線形予測子とニューラルネットワークのネットワーク予測子を組み合わせたものをニューラルネットワーク予測子にしよう、という発想です。最後の活性化関数を指数関数にし、損失関数をポアソン逸脱度にすることによって、ポアソン-log GLMと同じ構造になります。

NNによるGLMの補完(2)

- 初期値を以下のとおりとする
 - β^{GLM} : GLMによる推定結果 $\beta : 0$
 - β^{GLM} は学習対象としない(初期値のまま固定)
 - 結果として、NNのモデルは変更不要
オフセット α_i に用いるデータを「log Exposure」→「GLMの予測結果」に変更すればよい
 - CANNは、NNによるGLMのブースティングと考えられる
 - バリエーションとして、GLMの結果に対する重みを設定し学習対象とするモデルも考えられる

79

さらに、初期値としてGLMの推定結果を与えて、残りは0にします。

そうすると、損失関数の値はGLMと同じ値から出発し、改善方向にどんどん学習していきますので、ニューラルネットワークによってGLMをブースティングしているというような言い方もできるかと思います。

実際に計算した結果が、次のページの表です。

NNによるGLMの補完(3)

- MTPLデータによる結果(*1)

m odel	run tin e	bss.tran	bss.test	freq.tran	freq.test
GLM 2	15.31	31.396	30.894	10.10%	10.05%
NN (20-15-10) em b (es)	160.59	29.442	29.156	10.12%	10.03%
CANN (20-15-10) em b (es)	159.22	30.484	30.284	10.26%	10.20%

- GLMよりは精度向上
- NNよりは精度悪化
 β^{GLM} を学習対象としなかったことによる
- 予測精度がGLM対比で大幅に向上する場合、ベースとしたGLMに改善の余地があることを示唆
 - 非線形の関係
 - 交互作用

*1 Intel Core i7-6700 CPU, 16GB RAM

80

縦に3つのモデルが並んでいます。上が GLM2、前半で出てきたものです。真ん中が通常のニューラルネットワーク、一番下が CANN の結果です。これを見ると、ニューラルネットワークほど良くはないけれども、GLM よりは良い結果になっています。ニューラルネットワークよりも少し悪いのは、一部のパラメータ（GLM 部分の係数）を固定しているためです。全部動かしたら、それなりに良い結果になるかと思えます。

GLM に比べて CANN が改善しているとしたら、おそらく GLM の中で何か取りこぼしたような構造があるのではないかとということで、これを見ることにより GLM の改善につなげることのできるのではないかと考えられます。

5. CANNによる備金見積もり

81

ここからは、CANN を使ったもう一つの例として、備金の見積もりを考えてみます。

CANNによる備金見積もり(1)

- NNを用いた備金見積もりに関する研究(*1)

集約データによる備金見積もり法	ランオフトライアングルの形に集計されたデータを用いた手法 <ul style="list-style-type: none">• CANN Gabrielli et al. (2018) : 交差分類過分散ポアソンモデル Gabrielli (2020) : クレーム件数・金額の同時モデリング• リカレントニューラルネットワークを用いた手法 Kuo (2019)
個別データによる備金見積もり法	クレームごとの明細データを用いた手法

- 以下 Gabrielli et al. (2018) に沿って、CANN の備金見積もりへの応用について紹介

*1 Blier-Wong et al. (2021)

82

いろいろな見積もり方法が提案されていますが、ここでは、なじみのあるチェーンリーダーと

同様に、集約データであるトライアングルデータを使った備金見積もりについてご紹介します。

CANNによる備金見積もり(2)

- トライアングルデータの例(*1)

事故 年度	事故からの経過年数 (同一年度=0)											
	0	1	2	3	4	5	6	7	8	9	10	11
1994	9,560	4,778	1,644	942	665	502	360	287	224	176	162	135
1995	9,413	4,882	1,554	894	551	431	326	222	191	157	127	
1996	10,008	5,180	1,812	1,021	728	496	403	360	275	244		
1997	9,451	5,205	1,855	1,033	681	467	338	244	193			
1998	9,918	5,773	2,079	1,195	777	579	444	313				
1999	10,282	5,581	2,025	1,156	754	558	397					
2000	10,760	6,021	2,335	1,287	806	581						
2001	11,275	6,257	2,293	1,287	877							
2002	11,699	6,575	2,542	1,395								
2003	11,952	6,778	2,414									
2004	12,949	7,378										
2005	13,301											

- チェインラダー法によるIBNR 37,047

*1 p.87の人工データのうちLoB=1のもの

83

使うデータはこのトライアングルデータです。チェーンラダーで IBNR を予測すると 37,047 になります。

CANNによる備金見積もり(3)

- チェインラダー法と同じIBNRを与える確率論的手法
 - マックモデル(*1)
 - 過分散ポアソンモデル(*2)
- 過分散ポアソンモデル (Over-dispersed Poisson ~) はGLMであり、CANNに組み込むことができる

$$E[Y_{ij}] = \exp(\alpha_i + \beta_j) (= \mu_{ij})$$

$$\frac{Y_{ij}}{\phi} \text{ ind. } \sim \text{Poisson}(\mu_{ij}/\phi)$$

- $E[Y_{ij}]$ に積構造 $\exp(\alpha_i) \cdot \exp(\beta_j)$ を仮定しているので特に交差分類ODPモデル (cross-classified ~) という

*1 Mack (1993)
*2 Renshaw et al. (1998)

84

CANN を使うには、まず GLM でのモデリングが必要になります。チェーンラダーと同じ結果が得られるようなモデルを考えると、過分散ポアソンモデル (ODP モデル) がそれに当たります。

CANNによる備金見積もり(4)

■ cc-ODPモデルの計算例

- トライアングルデータを→の形式に変換
- i (事故年度), j (経過年数)は
カテゴリー変数とする
- GLMのあてはめ

	i	j	Y_{ij}
	1994	0	9,560
	1995	0	9,413
	1996	0	10,008
	1997	0	9,451
	1998	0	9,918
	1999	0	10,282

```
dat$i <- as.factor(dat$i); dat$j <- as.factor(dat$j)
ccODP.fit <- glm(Yij ~ i + j, data=dat, family=quasipoisson())
```

- 将来の i, j における Y_{ij} の予測

```
newYij <- data.frame(expand.grid(c(1994:2005), c(0:11)))
names(newYij) <- c("i", "j")
newYij <- newYij[newYij$i + newYij$j > 2005, ]
newYij$i <- as.factor(newYij$i); newYij$j <- as.factor(newYij$j)
Yij.pred <- predict(ccODP.fit, newdata=newYij, type="response")

> sum(Yij.pred)
[1] 37047.38
```

85

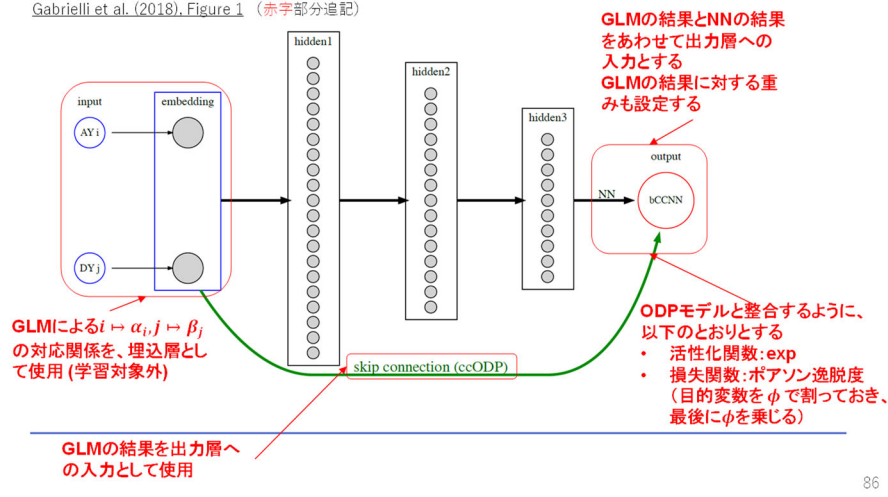
実際に ODP モデルで備金を計算すると、このようになります。まず一旦、トライアングルデータを、右上の表のような形に変換をした上で、このコードのように GLM を当てはめると、チェーンラダーの結果と一致していることが分かります。これを CANN に組み込みます。

CANNによる備金見積もり(5)

■ cc-ODPのCANNへの組み込み

(bCCNN: blended Cross-Classified Neural Network regression model)

Gabrielli et al. (2018). Figure 1 (赤字部分追記)



86

この図は、GLM をどのように CANN に組み込むのかを表したものです。左端の青い丸二つ、AY (事故年度) と DY (経過年数) がインプットデータです。これを、隣の四角、これは前半の最後に出てきた埋込層ですが、この中でそれぞれ GLM の結果にマッピングします。これを中間層へのインプットとして使い、また最後の出力層にも直接持っていき、ここで中間層の結果に直接足し込んで、最後の予測値とする、というモデルです。活性化関数や損失関数については、先にお話ししたものと同じです。このモデルを使って計算例をお示しします。

CANNによる備金見積もり(6)

■ 使用するデータ

- Individual Claims History Simulation Machine(*1)で生成
 - シミュレーションの設定はGabrielli et al. (2018)と同じ
 - KaggleのIndividual Claims History Simulation Machine Dataと同内容
- 1994~2005年発生事故について、クレームごとに発生から12年後までの各年度の支払を保有
 - 備金見積もりに使用するのは2005年までの支払のみ
- 保険種目：LoB=1~6

LoB	クレーム件数	最終損害額	単価	件数増加率	単価上昇率
1	250,040	278,854,738	1,115	1.0%	3.2%
2	250,197	289,046,068	1,155	0.5%	4.4%
3	99,969	110,128,046	1,102	5.3%	4.9%
4	249,683	434,138,451	1,739	1.0%	4.2%
5	249,298	433,835,526	1,740	1.5%	3.9%
6	99,701	177,651,537	1,782	3.1%	2.9%
計	1,198,888	1,723,654,366	1,438	1.5%	3.9%

*1 Gabrielli et al. (2018b)

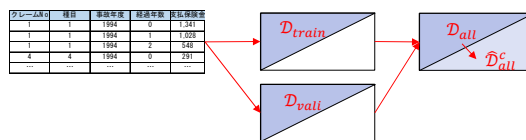
87

使用するデータは人工データです。12年間の事故年度それぞれについて、発生から12年間の支払データを持っています。つまり、トライアングルの左上だけではなく、右下も真の値が分かっているというデータです。それぞれ特徴が違う6つの種目、LoB 1から6についてのデータが含まれています。クレーム件数は120万件です。

CANNによる備金見積もり(7)

■ bCCNNによる備金見積もり

- ハイパーパラメータ設定のため、過去の支払データを訓練データ(50%)と検証データ(50%)に分割する
- まず、訓練データのみでモデルを構築し、検証データ上での損失を見ながらハイパーパラメータを決定する
 - 中間層のユニット数(q_1, q_2, q_3) = (20,15,20), 活性化関数(tanh), ドロップアウト率(10%)はGabrielli et al. (2018)の値を採用
 - エポック数のみ検証損失をもとに設定(全LoB共通で500)
- 決定したハイパーパラメータを用いて、すべての過去支払データでモデルを構築し、将来の支払を予測する

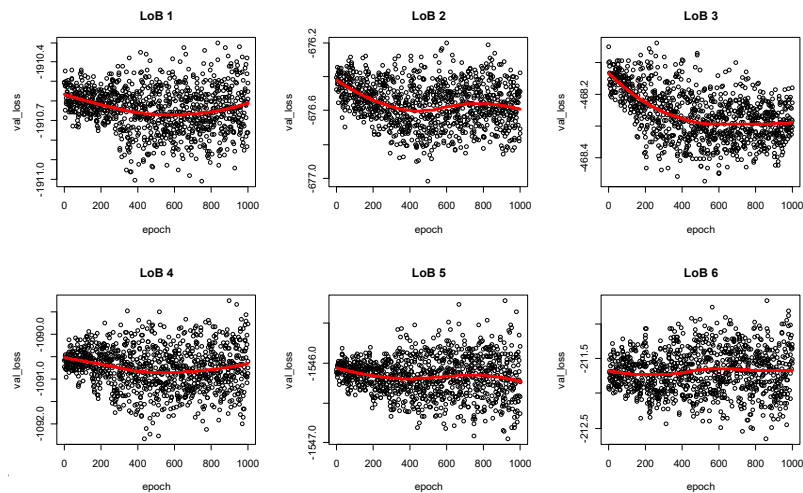


88

時間の都合で、ここは割愛させていただきます。

CANNによる備金見積もり(8)

- LoB=1~6の検証損失の推移 (赤線：Loess曲線)



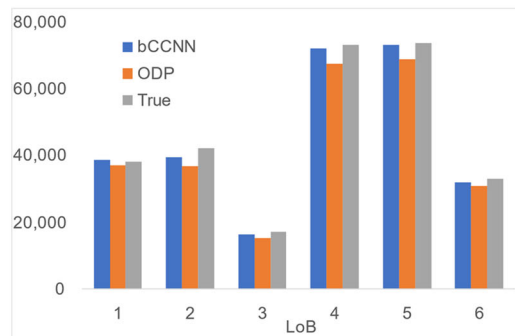
89

ニューラルネットワークでの学習の様子について、エポックごとの損失関数の値を見ると、学習が進んでもあまり損失は下がっていないように見えます。元々初期値としている ODP モデルの結果が、それなりに精度が高いものだったということです。このような図になっているため最適なエポック数を決めることは難しいのですが、LoB 1~6 の図を見比べて、共通で 500 ぐらいとしておけばいいのではないかと、ということでエポック数は 500 を採用しました。

CANNによる備金見積もり(9)

- bCCNNの結果を、ODPの結果および真のIBNR（人工データなので利用可能）と比較
 - 全てのLoBで、bCCNNの結果はODPよりも真のIBNRに近い

LoB	bCCNN	ODP	True
1	38,790 (+1.9%)	37,047 (▲2.7%)	38,084
2	39,647 (▲6.2%)	36,982 (▲2.5%)	42,269
3	16,392 (▲5.1%)	15,228 (▲1.9%)	17,278
4	72,057 (▲1.7%)	67,606 (▲7.7%)	73,285
5	73,333 (▲0.7%)	69,022 (▲6.5%)	73,847
6	32,134 (▲3.2%)	30,976 (▲6.7%)	33,208
計	272,354 (▲2.0%)	256,861 (▲7.6%)	277,972



90

結果は、ここにお示ししている通りです。左側の表では、左端の bCCNN が CANN の結果です。ODP はチェーンラダーの結果です。True は真の値です。どの種目についてもチェーンラダーの結果よりはいい結果が出ています。しかも、チェーンラダーの結果は全て一貫してマイナス方向にバイアスが掛かっているのに対し、bCCNN の結果は、一部プラスとなっているなど、バイアスが解消しているということが見て取れます。

CANNによる備金見積もり(10)

- LoB 5を例にbCCNNとODPの予測値の違いを確認
 - bCCNNは、ODPに比べ、新しい事故年度ほど支払いが遅れる傾向を反映している
 - 相対的に金額の大きい部分（経過年数が小さい部分）での精度が向上している

①bCCNN / ODP

事故年度	経過年数	0	1	2	3	4	5	6	7	8	9	10	11
1994	0	0.1%	0.6%	1.1%	1.7%	2.3%	2.9%	3.5%	4.1%	4.7%	5.3%	5.9%	6.5%
1995	0	4.9%	0.2%	2.4%	2.2%	1.7%	1.2%	0.8%	0.4%	0.1%	-0.2%	-0.5%	-0.8%
1996	0	5.1%	2.0%	4.2%	3.9%	2.9%	2.4%	1.9%	1.4%	1.0%	0.7%	0.4%	0.0%
1997	0	4.8%	-0.4%	-0.8%	-0.9%	-0.7%	-0.7%	-0.5%	-0.3%	-0.1%	0.0%	1.0%	1.5%
1998	0	0.6%	-1.3%	0.0%	1.0%	1.8%	2.3%	2.7%	3.1%	3.4%	3.6%	3.9%	4.1%
1999	0	-0.5%	-0.4%	1.6%	2.6%	3.3%	3.7%	4.1%	4.4%	4.7%	4.9%	5.0%	5.3%
2000	0	2.7%	1.3%	4.1%	5.0%	5.5%	5.8%	6.0%	6.2%	6.4%	6.5%	6.6%	6.7%
2001	0	-0.8%	-0.2%	1.9%	2.9%	3.6%	4.0%	4.3%	4.7%	4.9%	5.1%	5.3%	5.5%
2002	0	-0.6%	1.0%	2.6%	4.5%	5.0%	5.2%	5.4%	5.6%	5.9%	6.0%	6.2%	6.4%
2003	0	-0.3%	3.0%	6.7%	7.3%	7.5%	7.7%	7.8%	7.9%	7.9%	8.0%	8.0%	8.1%
2004	0	-2.1%	3.1%	7.0%	7.5%	7.8%	7.9%	8.0%	8.1%	8.1%	8.2%	8.2%	8.2%
2005	0	-1.4%	3.4%	7.9%	8.3%	8.5%	8.5%	8.5%	8.6%	8.6%	8.7%	8.7%	8.7%

②ODP / True IBNR

事故年度	経過年数	0	1	2	3	4	5	6	7	8	9	10	11
1994	0	-6.3%	0.6%	0.4%	12.3%	6.1%	14.1%	3.4%	6.4%	0.5%	2.8%	-2.9%	0.0%
1995	0	-7.6%	0.7%	11.9%	8.2%	8.2%	15.4%	11.3%	10.5%	12.7%	3.1%	-0.2%	
1996	0	-4.7%	2.3%	9.2%	11.9%	8.8%	4.0%	4.2%	-10.7%	-16.5%	-16.9%	-10.1%	-26.2%
1997	0	-2.2%	3.3%	1.9%	0.1%	-0.1%	6.2%	2.3%	-3.5%	11.6%	3.2%	-1.8%	-0.5%
1998	0	0.1%	3.5%	0.1%	-2.7%	-1.0%	11.9%	-7.9%	-6.5%	-5.9%	-16.4%	-14.2%	-1.1%
1999	0	2.0%	-0.1%	0.7%	-6.4%	-2.7%	-6.5%	-7.1%	-2.6%	-2.2%	-9.9%	-11.7%	-16.5%
2000	0	4.8%	0.1%	-6.8%	-8.2%	-6.5%	-7.6%	-17.8%	-20.1%	-23.8%	-25.4%	-30.8%	-32.7%
2001	0	1.9%	-1.6%	-0.1%	-4.6%	-5.2%	-0.6%	0.7%	2.4%	0.7%	1.1%	-15.7%	-19.6%
2002	0	1.8%	0.4%	-6.1%	-3.9%	-2.0%	-1.4%	7.7%	20.0%	10.0%	27.3%	14.5%	-9.7%
2003	0	4.5%	-2.5%	-7.9%	-9.2%	-8.7%	-4.8%	-5.4%	-6.7%	0.4%	-7.1%	-14.0%	-21.8%
2004	0	2.1%	-3.1%	-8.6%	-7.0%	-6.4%	-2.2%	-1.2%	-8.9%	-3.7%	-6.8%	-7.0%	-10.4%
2005	0	0.0%	-4.5%	-7.3%	-11.1%	-7.4%	-6.4%	-6.6%	-6.1%	-1.6%	5.8%	3.0%	-3.4%

③bCCNN / True IBNR

事故年度	経過年数	0	1	2	3	4	5	6	7	8	9	10	11
1994	0	-1.6%	-0.1%	3.8%	7.4%	2.3%	0.9%	0.1%	5.3%	5.9%	0.0%	-4.6%	-1.4%
1995	0	-3.0%	0.9%	5.1%	3.1%	3.0%	10.5%	7.0%	6.7%	9.3%	19.7%	0.6%	-2.3%
1996	0	0.2%	0.3%	4.5%	7.9%	5.6%	1.5%	2.2%	11.9%	17.6%	17.1%	-10.4%	26.2%
1997	0	1.2%	0.8%	-1.1%	-1.9%	-2.0%	6.0%	2.1%	-3.5%	2.3%	4.3%	-0.5%	11.1%
1998	0	0.7%	2.1%	0.1%	-1.7%	0.8%	-9.9%	-5.4%	-3.6%	-2.7%	-13.4%	-10.9%	3.0%
1999	0	0.5%	-0.5%	2.3%	-3.0%	0.5%	-3.0%	-3.3%	1.7%	2.4%	-5.5%	-7.3%	-12.1%
2000	0	1.9%	1.4%	-2.9%	-3.7%	-1.4%	-2.3%	-12.9%	-15.1%	-18.0%	-20.5%	-26.2%	-28.1%
2001	0	0.1%	-1.8%	1.8%	1.2%	-0.5%	3.4%	5.1%	2.2%	5.6%	6.3%	-9.1%	-15.2%
2002	0	-0.9%	1.3%	-1.7%	0.4%	2.9%	3.8%	13.7%	27.0%	16.6%	25.3%	4.8%	-3.9%
2003	0	2.1%	0.0%	-1.7%	-2.7%	-1.8%	2.5%	2.0%	0.6%	8.4%	0.3%	-7.1%	-15.5%
2004	0	0.0%	0.0%	-2.2%	-0.1%	0.8%	5.5%	6.7%	-1.5%	4.2%	0.8%	0.6%	-3.0%
2005	0	-1.4%	-1.2%	0.0%	-3.7%	0.5%	1.8%	1.4%	3.1%	6.9%	15.0%	11.9%	5.0%

もう少し細かく見てみましょう。LoB 5を例に取り、詳細を見てみます。①の図は、トライアングルのセルごとの支払保険金の予測値について、bCCNNとチェーンラダーを比べたものです。こうして見ると結果は一目瞭然で、経過年数が2以上のところでは、古い事故年度でbCCNNの方が赤くなっています。赤いところは、bCCNNの方が小さくなっているところです。青いところはbCCNNの方が大きくなっているところです。

ODPモデルでは、「経過年数ごとの効果は事故年度によらず一定だ」と見ているのですが、実際にはbCCNNで予測しているように、新しい事故年度ほど経過の長いところの支払いが大きくなるという効果を入れるべきではないか、というところが見て取れます。

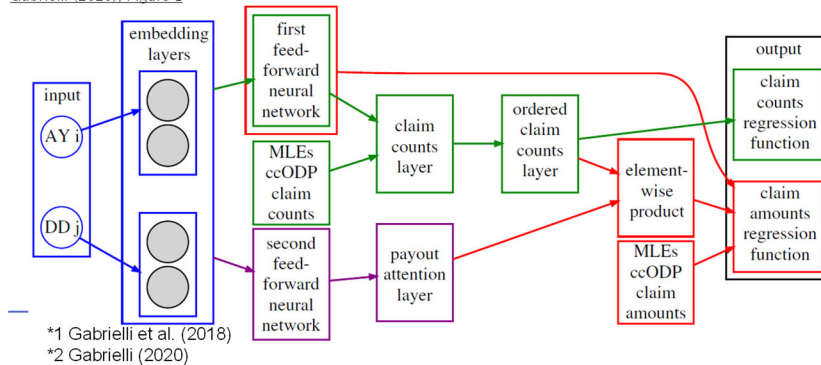
②、③は、それぞれ、ODPとbCCNNの予測結果と真の値を比較したものです。どちらも経過の長いところで真の値との差がありますが、この辺りは金額が小さいためそれほど効いてきません。影響が大きい、金額が大きい経過の短いところを見ると、bCCNNはODPに比べて色が薄く、全体に精度が良くなっていることが見えるかと思えます。

CANNによる備金見積もり(11)

■ さらなる精度向上

- 複数のポートフォリオ(LoB等)の同時モデル化 (*1)
→共通する構造を学習することで予測精度が向上
- クレーム件数の情報の利用 (*2)
NNDODP (Neural Network Double ccODP) model

Gabrielli (2020), Figure 1



92

「更に精度を良くするために、どうするか」という提案も、いろいろとされています。一つは、先ほど LoB 1~6 をばらばらに予測しましたが、「ばらばらに予測するのではなく、全部まとめて予測したらどうなるのか」ということが、同じ論文の中で提案されています。ただし、今回計算してみたところ、ばらばらに予測する場合と比べそれほど精度は変わりませんでした。

もう一つのアイデアとして、「金額だけではなく、件数の情報も使ったらどうなるのか」というモデルも提案されています。スライドで NNDODP モデルとしているものです。伝統的手法にも件数と金額の両方を使ったモデルがありますが、同様に、件数と単価それぞれについてのニューラルネットワークを作って、これを組み合わせたモデルです。

6. まとめ

93

まとめ

- NNをGLMと組み合わせることで、既存の保険数理手法より予測精度の高いモデルが得られる
 - 一方で、「最適な」モデルが一意に定まらない、モデルや結果が（GLMと同じ意味では）説明できないといった問題があり、用途によってはそのまま使用することができない
- NNが学習した結果を確認することで、見落としていた構造に気づき、既存の保険数理手法に反映できる
 - 非線形の関係、交互作用など
 - 分析者の経験や感覚に頼ってきた特徴量の前処理について、NNを活用することで説明可能性が高まるかもしれない
- アクチュアリーがNNを学ぶためのよい教材がある
 - データ分析のための道具の一つとして使えるようにしておくことは有用

94

ここまで、「ニューラルネットワークを GLM と組み合わせることで、既存の手法よりも精度が高いモデルが得られる」ということを見てきました。一方で、最適なモデルが一つに決まらない、モデルや結果が説明できない、といった問題があります。この問題は、GLM と組み合わせる、あるいは、初期値を固定することにより緩和されている面はありますが、それでも残っており、用途によっては、そのまま使うことができないこともあるかと思えます。

そのような場合でも、ニューラルネットワークの結果をそのまま使うのではなく、ニューラルネットワークが学習した結果を確認することで、見落としていた構造に気づき、既存の手法に反映するといった使い方はできるのかもしれませんが。従来、伝統的手法で分析者の経験や感覚に頼ってきた特徴量の前処理について、ニューラルネットワークを使うことにより、かえって説明可能性が高まるということもあるかもしれません。

冒頭で紹介をしたスイスアクチュアリー会の Tutorials をはじめとして、アクチュアリーがニューラルネットワークを学ぶための良い教材がそろってきていますし、使い方のアイデアもいろいろと出てきています。このようなものに加え、この発表が、「データ分析のための道具の一つとして、ニューラルネットワークを使ってみよう」と考えるきっかけになってくれれば幸いです。

以上で、私のパートの発表を終わります。ありがとうございました。

【司会】 渡辺さん、ありがとうございました。

以上で、セッション C の 4 「損保アクチュアリー業務におけるニューラルネットワークの活用」を終了します。発表された 2 名の方、ありがとうございました。

参考文献

- Biecek, P., & Burzykowski, T. (2021). Explanatory model analysis: explore, explain, and examine predictive models. CRC Press. <https://ema.drwhy.ai/>
- Blier-Wong, C., Cossette, H., Lamontagne, L., & Marceau, E. (2021). Machine learning in P&C insurance: A review for pricing and reserving. *Risks*, 9(1), 4.
- Ferrario, A., Noll, A., Wüthrich, M.V. (2018). Insights from inside neural networks. Available at SSRN 3226852.
- Gabrielli, A. (2020). A neural network boosted double overdispersed Poisson claims reserving model. *ASTIN Bulletin: The Journal of the IAA*, 50(1), 25-60.
- Gabrielli, A., Richman, R., & Wuthrich, M. V. (2018). Neural Network Embedding of the Over-Dispersed Poisson Reserving Model. Available at SSRN 3288454.
- Gabrielli, A., & V Wüthrich, M. (2018b). An individual claims history simulation machine. *Risks*, 6(2), 29.
- Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin: The Journal of the IAA*, 23(2), 213-225.
- Molnar, C. (2018). A guide for making black box models explainable.
URL: <https://christophm.github.io/interpretable-ml-book>.
和訳: <https://hacarus.github.io/interpretable-ml-book-ja/index.html>
- Renshaw, A. E., & Verrall, R. J. (1998). A stochastic model underlying the chain-ladder technique. *British Actuarial Journal*, 4(4), 903-923.
- Schelldorfer, J., Wüthrich, M.V. (2019). Nesting classical actuarial models into neural networks. Available at SSRN 3320525.
- Wüthrich, M. V. (2019). From generalized linear models to neural networks, and back. Available at SSRN 3491790.