

支払備金の見積もりにおける機械学習・深層学習的手法の適用と 従来手法との比較 (2018 ASTIN WP 参加報告)

アクサ生命保険株式会社 平松 雄司

【司会】 時間となりましたので、セッションB、「支払備金の見積もりにおける機械学習・深層学習的手法の適用と従来手法との比較 (2018 ASTIN WP 参加報告)」を開始します。発表者は、アクサ生命の平松さんです。それでは、よろしくお願いします。

支払備金の見積もりにおける 機械学習・深層学習的手法の適用と 従来手法との比較 (ASTIN WP 参加報告)

ASTIN 関連研究会 活動内容のご紹介

2018 / 11 / 09
ASTIN 関連研究会

アクサ生命保険 平松 雄司



【平松】 よろしくお願いたします。アクサ生命の平松と申します。本日は、お忙しい中お集まりいただき、ありがとうございます。本発表では、昨年末から今年初めにかけて参加いたしました、ASTIN 国際ワーキングパーティーの参加報告をさせていただきます。こちらが、本日のアジェンダとなります。

目次

1. ASTIN WP について
2. 本研究の目的と概要
3. 各種ツール
4. 検討項目
5. 担当項目
6. 検討内容詳細
7. 今後の展望
8. ASTIN 関連研究会
活動内容のご紹介



2

ASTIN WP の参加報告に加えまして、最後に、ASTIN 関連研究会の活動内容ですね。そちらの方も軽くさせていただきますが、1時間程おつきあいいただけましたらと思います。

Machine Learning and Traditional Methods Synergy in Non-Life Reserving @ ICA2018



4

最初に、ASTIN WP について、ご説明をさせていただきます。今年の6月に、4年に1度開催されます国際アクチュアリー会議が、ベルリンにて行われました。ASTIN WP は、開催期間中の月曜日の午後のセッションにおきまして、「Machine Learning and Traditional Methods Synergy in Non-Life Reserving」と題しまして、支払準備金の見積もりにおける機械学習と深層学習的手法の適用に関する検討内容を発表いたしました。この取り組みは、2014年に発足し、教師あり・教師なし学習に関しまして検討いたしましたワ

ーキングパーティーや、2016年に発足し、ニューラルネットワークを支払備金の見積りに適用する、そのような検討をいたしましたワーキングパーティーに続くものとなっております。

ワーキングパーティーの方なのですが、メンバーは、各国から参加した総勢34名で構成されておりました。日本からは、私を含めまして3人、ASTIN 関連研究会より参加いたしました。コミュニケーションツールは、skype や slack。また、プログラムコードはR。そして、そちらのプログラムコードの管理は、GitHub を利用いたしました。後ほど、こちらの各ツールに関しまして、簡単にご紹介をさせていただきます。また、これは少し雑感めいたもので恐縮なのですが、日本は、GMT+9 ですので、skype のミーティング時間が夜中の2時、3時ということで、非常に大変でした。

本研究の目的と概要

1. **アクチュアリーは、
チェーンラダー法やボーンヒュッターファーガソン法といった
「伝統的な支払備金の見積り手法」を今日まで使用してきている。**
2. **これらの伝統的な手法は、
集積レベル（個々のクレーム毎のレベルではなく）で、
支払備金の見積りに一定の成功を収めてきた。**
3. **しかし、一方で、伝統的な手法は、
クレームデータに内在する特性やその複雑な相互作用
（契約者毎の特性、契約内容などに起因するもの）
を考慮することは難しかった。**
4. **昨今、ビジネスの世界で成功を収めている
機械学習・深層学習的手法を支払備金の見積りに適用した場合、
伝統的手法と比較して、どのような利点・欠点があるかを検証する。**



7

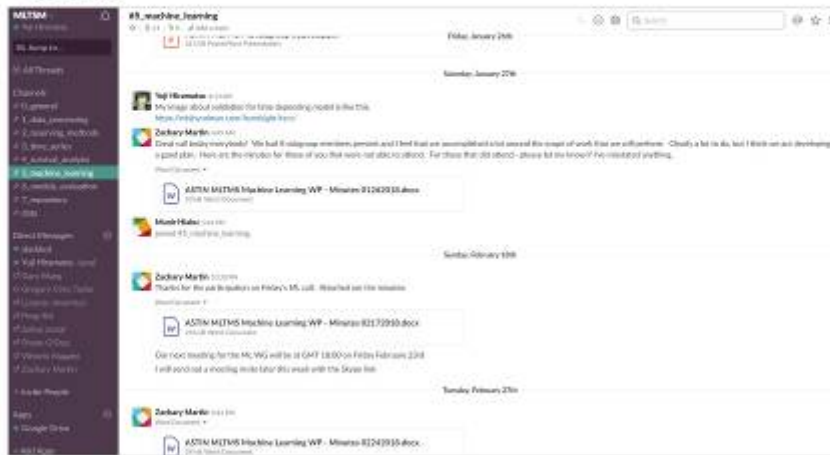
それでは、次に、本ワーキングパーティーの研究目的と概要に関しまして、ご説明をさせていただきます。アクチュアリーなのですけれども、今日まで、支払備金の見積り手法として、チェーンラダー法やボーンヒュッターファーガソン法などの手法を伝統的に使用いたしております。これらの伝統的な手法は、ここでは集積レベルと申し上げておりますが、基本的にはトライアングルデータの形にしたものを指しております。そのような集積レベルでの支払備金の見積りにおいて、一定の成功を収めてきていると思えます。

しかし、一方で、伝統的な手法は、個々のクレームデータに内在しておりますような特性や、複雑な相互作用といったものを考慮することは非常に難しいという欠点がございます。そこで、本ワーキングパーティーでは、他の業界、例えばIT業界のウェブマーケティングなどの分野で成功を収めております機械学習・深層学習的手法を支払備金の見積りに適用した場合、伝統的手法と比較して、どのような利点・欠点があるかを検証いたしました。

各種ツール



時差の関係もあり、基本的なコミュニケーションは slack にて。



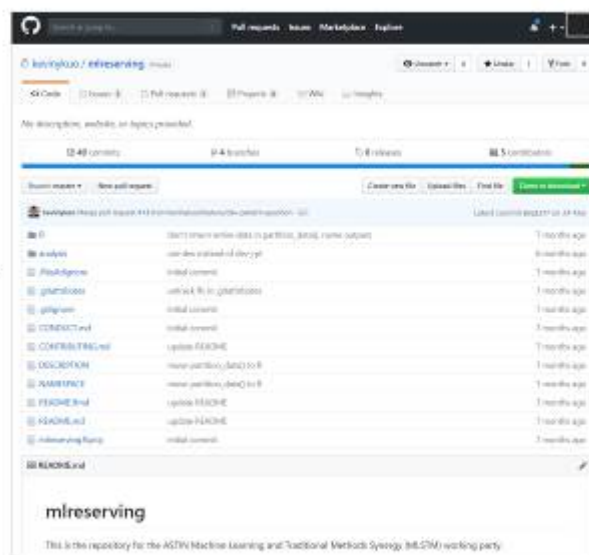
9

それでは、詳細な検討内容に入る前に、先ほど少しご紹介いたしました、本ワーキングパーティーで使用いたしました各種ツールのお話を、簡単にさせていただきます。まず、slack ですが、現在、日本でも多くのビジネスで使用されるようになってきましたコミュニケーションツールです。主に IT 関連の会社等で、社内プロジェクトで頻繁に使用されているとは思いますが、一部の保険会社の皆様方のもとでも使われているものかと思えます。本ワーキングパーティーも、slack において、メンバー間のやり取りなどを、意思疎通を行っておりました。

各種ツール



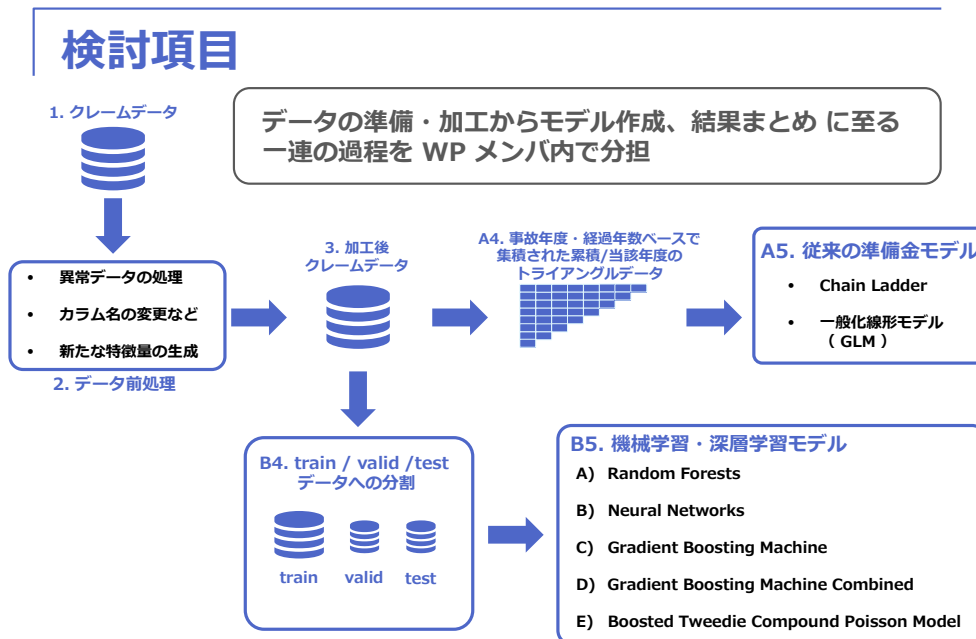
code の一意性を保つために、github にてコードを管理



10

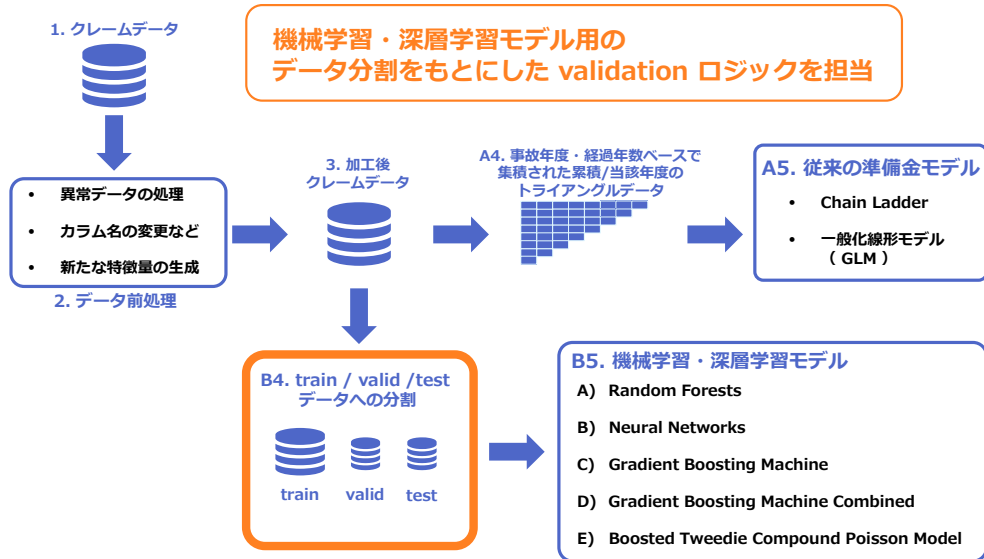
また、R のプログラムコードですが、コードの一意性ですね。つまり、何が最新なのか、バージョ

ンが古いコードは何なのか。そのようなところをしっかりと管理するために、GitHub でコード管理をいたしておりました。残念ながら、こちらは現在もプライベート管理状態ですので、一般公開はしていないということで、そちらをご承知いただけましたらと思います。



では、本発表の本題であります、検討内容の説明に入っていきたいと思ひます。本ワーキングパーティーのアウトプットとして目指しておりましたものは、従来の準備金計算モデルと、機械学習・深層学習的手法を適用した準備計算モデルの比較でした。そのため、検討項目は、データの選定から、データの前処理ですね。そちらを経たあと、従来のモデルおよび機械学習・深層学習モデルの構築、そして、最終的に結果の比較。そのような一連の分析過程を網羅したものとなりました。本発表では、その分析過程を最初から最後まで、一気通貫した形でご報告させていただきます。そのため、容量が多いので、非常に駆け足なところもあるとは思ひますが、ご承知いただけましたらと思ひます。

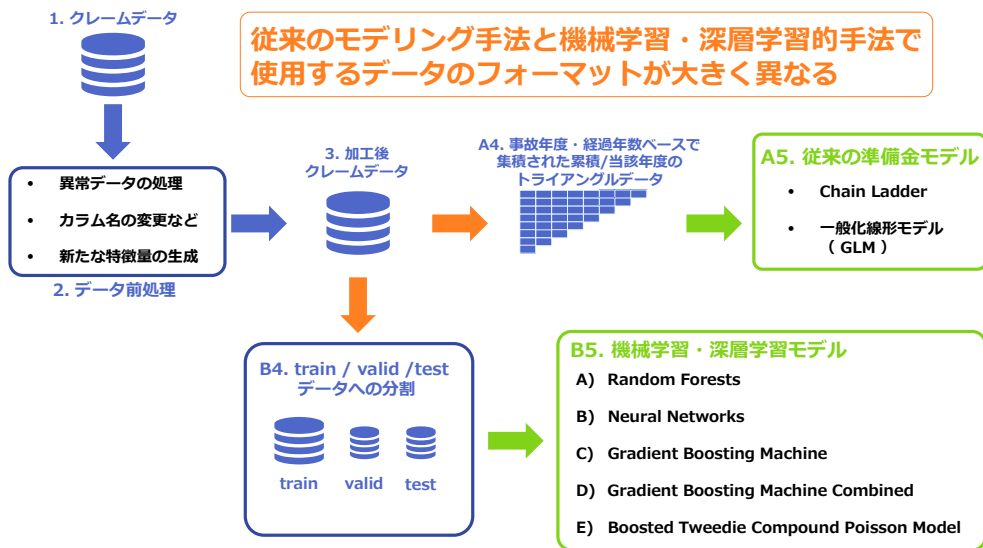
担当項目



14

今、お話しいたしましたとおり、一連の分析過程を 34 名のメンバー間で分担いたしましたでしたが、私の担当項目は、先ほどからお見せいたしております、このパイプラインと申しますか、一連のフローなのですが、こちらの部分になります。私が担当しましたものは、主に機械学習・深層学習モデル用の validation の枠組みの設計と、基礎コードの構築でした。こちらの validation という言葉は、お聞きになられない方も多数いらっしゃると思いますので、後ほどご説明をさせていただきます。また、本ワーキングパーティーの分析過程は分担制でしたので、私の担当部分以外で非常に細かいところになってきますと、なぜそうなったのかなどの経緯も含めまして、あまり定かではない部分もあるため、説明が少しあやふやな部分はあるかもしれませんが、ご承知いただけましたらと思います。

検討内容詳細



16

それでは、検討内容の詳細の説明に入らせていただきます。こちらは、先ほどからお見せいたしております、検討内容の一連の流れを図示したパイプラインなのですが、分かりやすく明滅させていると思いますが、従来の準備金モデルと機械学習・深層学習モデルで使用するデータの形態は、非常に大きく異なります。従来の準備金モデルで使用しておりますデータは、事故年度・経過年数別に集積されたトライアングルデータの形態を取ります。一方で、個々のクレームデータごとの粒度になっているものが、機械学習・深層学習モデルとなります。こちらのデータは、validation用にですね、train、valid、testと、3つにデータを分割して使用することになります。

1. クレームデータ

- Swiss Re 提供の「専門職業人賠償責任保険」のデータ
- データの取得期間は、1994年～2016年（但し、契約年は2009年まで。以降はrun offのみ。）
- 各クレーム毎の粒度のデータ
- 16個の変数の名前、型、詳細は以下表の通り

ASTIN WP 2018 REPORT より

Variable	Type	Description
Paid YY	Numerical	Payment of the year YY (incremental)
O/S YY	Numerical	Case reserve (outstanding) of the year YY (cumulative)
Incurred YY	Numerical	Incurred amount of the year YY (cumulative)
UWY	Discrete	Underwriting year
DoL	Discrete	Accident year (Date of Loss)
DoN	Discrete	Reporting year (Date of Notification)
Settlement Year	Discrete	Settlement year for claims that are already closed
Year of Birth	Discrete	Birth date of the claimant
Insured	Categorical	Anonymized code of insured entity
Loss Number	Categorical	Univocal number of claim
Open / Close	Categorical	Status of claim (1 for open and 2 for close)
Age Group	Categorical	Grouping claimants in age classes
Insured Profession	Categorical	Groups of claimants' profession
Incident Grouping	Categorical	Subcategory related to insured field
Case specialty	Categorical	Description of Claim Event
Event	Categorical	Flag for difficult type of losses (expensive and long-tail claims)

金額に関連した項目

時期に関連した項目

個人やクレームの属性に関連した項目



18

それでは、一連のプロセスを、最初から説明に入らせていただきたいと思います。まず元データに関して、ご説明をいたします。元データなのですけれども、Swiss Re 様から、ご厚意で、専門職業人賠償責任保険のデータを提供いただきました。こちらは、私も聞き慣れなかったのですけれども、基本的には労災補償の一種とお考えいただければと思います。データの取得期間は、1994年から2016年までとなります。また、契約は、2009年度までにされたものとなっていますので、以降はrun off ですね。経過のみを観察したデータとなっています。データはクレームごとの粒度となっておりまして、スライド内の表にありますような変数を持っております。主に経過年数ごとの支払保険金、発生保険金、普通備金。また、Underwriting year などの、時期に関連した項目ですね。タイミングですね。そして、3つ目が、契約者の年齢や、クレームの性質などに関連した項目。以上、大別すると3つから成っています。

1. クレームデータ

元データの一部

ユニークキー

Insured	Loss Number	UWY	DoL	DoH	Settlement Year	Open/Closed	Year of Birth	Age Group	Case Specialty	Insured Profession	Incident Grouping	Event
4	40738	2007	2007	2008	2010	2		2	46	91	4	2
4	27646	2007	2008	2008		2		2	32	151	3	2
4	27647	2007	2008	2008		2		2	32	151	3	2
4	27648	2006	2007	2008		1		2	32	119	3	2
4	27649	2007	2007	2008		2		2	32	62	3	2
4	27650	2005	2005	2008		1	2005	4	41	53	4	2
4	35426	2007	2007	2008		1	2008	5	41	53	4	1
4	27651	2004	2004	2008		2		2	32	151	3	2
4	27658	2007	2008	2008		2		2	32	119	3	2
4	27653	2006	2007	2008		2		2	16	29	4	2

PAID94	O/S94	Incurred94	PAID95	O/S95	Incurred95	PAID14	O/S14	Incurred14	PAID15	O/S15	Incurred15	PAID16	O/S16	Incurred16
0	0	0	0	0	0	0	-162	33443	0	-162	33443	0	-162	33443
0	0	0	0	0	0	0	0	14037.5	0	0	14037.5	0	0	14037.5
0	0	0	0	0	0	0	0	346.5	0	0	346.5	0	0	346.5
0	0	0	0	0	0	0	-2176.25	6250	0	-2176.25	6250	0	-2176.25	6250
0	0	0	0	0	0	0	-79.5	2623.25	0	-79.5	2623.25	0	-79.5	2623.25
0	0	0	0	0	0	0	37500	37500	0	37500	37500	0	37500	37500
0	0	0	0	0	0	0	125000	125000	0	125000	125000	877.5	124122.5	125000
0	0	0	0	0	0	0	0	2841.25	0	0	2841.25	0	0	2841.25
0	0	0	0	0	0	0	15.75	3896.5	0	15.75	3896.5	0	15.75	3896.5
0	0	0	0	0	0	0	-4.25	370.75	0	-4.25	370.75	0	-4.25	370.75

各クレームを表す `Loss Number` に対して、
 カラム（横）方向に、
 経過年度別に、支払保険金の額、発生保険金の額、普通備金の額が並んでいる



次に、実際のデータの一例をごらんいただきたいと思います。元データなのですけれども、各クレームのユニークキーである Loss Number というものが、カラムの名前として元々ございまして、それに対して横方向に、経過年度別に支払保険金の額、発生保険金の額、普通備金の額が並んでいます。このようなテーブルデータの形を、「ワイドフォーマット」と呼びます。データ点数は、このワイドフォーマットで約5万点ほどでした。後ほどお話をいたしますが、こちらのテーブルを、各クレームに対して経過年数ごとに横だったものを、縦に、行方向に分解した形でありまして「ロングフォーマット」という形式にいたします。その場合ですと、当然データ点数は増えるのですけれども、61万点ほどのデータ点数となります。

2. データ前処理

- Long format への変換
- 例外データの処理
- カラム名の変更など
→ 分かりやすい名前へ
- 新たな特徴量の生成
→ calendar_year
→ report_lag
→ initial_case_reserve
→ ...

以降、説明変数と同義の
特徴量という表現を使用いたします

```
library(stringr)
library(dplyr)
reshape_and_filter_data <- function(data) {
  # Long format
  dplyr::gather(Transactions, value, NAIS04.Incurred) %>%
  # Creating calendar_year and transaction_year
  dplyr::mutate(calendar_year = as.integer(abs(Transactions[-2,])),
               Transaction = abs(Transactions[-2,])) %>%
  # Formatting calendar_year
  dplyr::mutate(calendar_year = 1994 + (calendar_year > 18) * calendar_year,
               calendar_year = NA) %>%
  # Removing NA's
  dplyr::mutate(drop = (calendar_year == NA) %>%
  # Feature selection
  dplyr::rename(
    transaction_type = Transaction,
    claim_number = "Loss Number",
    status = "Open/Closed",
    accident_year = NA,
    notification_year = DOP,
    underwriting_year = DAY,
    age_group = "Age Group",
    settlement_year = "Settlement Year",
    year_of_birth = "Year of Birth",
    case_sociality = "Case Sociality",
    insured_profession = "Insured Profession",
    incident_grouping = "Incident Grouping",
    event = "Event",
    year = "Year"
  )
}

generate_features <- function(data) {
  data %>%
  dplyr::mutate(status = factor(status),
               age_group = factor(age_group),
               case_sociality = factor(case_sociality),
               insured_profession = factor(insured_profession),
               incident_grouping = factor(incident_grouping),
               event = factor(event)) %>%
  dplyr::filter(calendar_year == notification_year) %>%
  # Incremental loss is set to zero
  dplyr::mutate(incr_paid = paid) %>%
  dplyr::rename(claim_number = ID) %>%
  dplyr::arrange_by(claim_number) %>%
  # Incremental incurred is calculated as observed - incurred from previous
  dplyr::mutate(incr_incurred = incurred - lag(incurred, default = 0)) %>%
  # Note: system is default state is open and using open status
  # If prior case ending reserve is 0 then the claim is closed
  # Note: if there is a non-zero
  # (non zero incremental underwritten, the claim status is also set to open
  # Note: if there is a non-zero
  dplyr::mutate(claim_status_target = ifelse(
    incr_paid() == 0 && incr_incurred() == 0, "Closed",
    (abs(lag(paid)) != 0 | abs(incr_paid) != 0 | abs(incr_incurred) != 0), "Open"
  ))
}
```



では、今、お話いたしましたようなワイドフォーマット、ロングフォーマットも含めまして、データの前処理に関して、ご説明をいたします。データの前処理なのですけれども、先ほどお話いたしましたように、列方向に経過年数ごとの支払保険金、発生保険金、普通備金が並んでいるワイドフォーマットから、行方向に経過年数ごとにそれらが並んでいるようなロングフォーマットへと変換いたします。これは、同じクレームでも、経過年数分のデータ点数を持つということを意味いたしますので、非常にデータ点数が増えることとなります。また、例外データの処理も行っております。例えば、1994年よりも前にクレームが発生していて、いまだにクローズしていないものを除外したり、クレームの報告がクレームの発生よりもなぜか前になっているような、多分、何かのミスだとは思うのですけれども、そのような異常なデータを除外したりしています。

また、前処理で一番重要になってきますものが、新たな特徴量の生成です。会場の皆様の中には、特徴量という言葉が耳慣れない方もいらっしゃると思うのですけれども、データサイエンスの分野では非常に一般的な言葉でして、基本的には説明変数と同義、同じ意味だとお考えいただければと思います。今後は、本発表ではこちらの言葉を使用させていただきます。

また、新たに作成します特徴量なのですけれども、こちらのスライドにありますように、calendar_yearやreport_lagなど、簡単に四則演算で求められるような特徴量を作ることとなります。これは、後ほど少し言及できるかもしれないのですけれども、基本的に機械学習や深層学習のモデルは、簡単な四則演算で作れる特徴量をなかなかモデル内部で作りにくい、自分たちで表現しにくいところがありまして、そのため、前もって前処理の段階でそのようなところを作ることとなります。このようにして作成した特徴量は、後ほどご説明させていただきますが、いろいろな機械学習・深層学習のモデルで使うこととなります。

3. 加工後クレームデータ

例：1つのクレームデータ (claim_number = 1000)

claim_number	calendar_year	accident_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year
1000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
1000	2001	2001	2001	2001	2001	2001	2001	2001	2001	2001	2001	2001
1000	2002	2002	2002	2002	2002	2002	2002	2002	2002	2002	2002	2002
1000	2003	2003	2003	2003	2003	2003	2003	2003	2003	2003	2003	2003
1000	2004	2004	2004	2004	2004	2004	2004	2004	2004	2004	2004	2004
1000	2005	2005	2005	2005	2005	2005	2005	2005	2005	2005	2005	2005
1000	2006	2006	2006	2006	2006	2006	2006	2006	2006	2006	2006	2006
1000	2007	2007	2007	2007	2007	2007	2007	2007	2007	2007	2007	2007
1000	2008	2008	2008	2008	2008	2008	2008	2008	2008	2008	2008	2008
1000	2009	2009	2009	2009	2009	2009	2009	2009	2009	2009	2009	2009
1000	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
1000	2011	2011	2011	2011	2011	2011	2011	2011	2011	2011	2011	2011
1000	2012	2012	2012	2012	2012	2012	2012	2012	2012	2012	2012	2012
1000	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
1000	2014	2014	2014	2014	2014	2014	2014	2014	2014	2014	2014	2014
1000	2015	2015	2015	2015	2015	2015	2015	2015	2015	2015	2015	2015
1000	2016	2016	2016	2016	2016	2016	2016	2016	2016	2016	2016	2016

claim_number	calendar_year	accident_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year	calendar_year
1000	2000	0	0	0	0	0	0	0	0	0	0	0
1000	2001	1	12000	0	12000	0	0	0	0	0	0	0
1000	2002	1	12000	0	12000	0	0	0	0	0	0	0
1000	2003	1	12000	0	12000	0	0	0	0	0	0	0
1000	2004	1	12000	0	12000	0	0	0	0	0	0	0
1000	2005	1	12000	0	12000	0	0	0	0	0	0	0
1000	2006	1	11815	0	11815	0	0	0	0	0	0	0
1000	2007	1	11815	0	11815	0	0	0	0	0	0	0
1000	2008	1	11815	0	11815	0	0	0	0	0	0	0
1000	2009	1	11815	0	11815	0	0	0	0	0	0	0
1000	2010	1	11815	0	11815	0	0	0	0	0	0	0
1000	2011	1	11815	0	11815	0	0	0	0	0	0	0
1000	2012	1	11815	0	11815	0	0	0	0	0	0	0
1000	2013	1	11815	0	11815	0	0	0	0	0	0	0
1000	2014	1	11815	0	11815	0	0	0	0	0	0	0
1000	2015	1	11815	0	11815	0	0	0	0	0	0	0
1000	2016	1	11815	0	11815	0	0	0	0	0	0	0

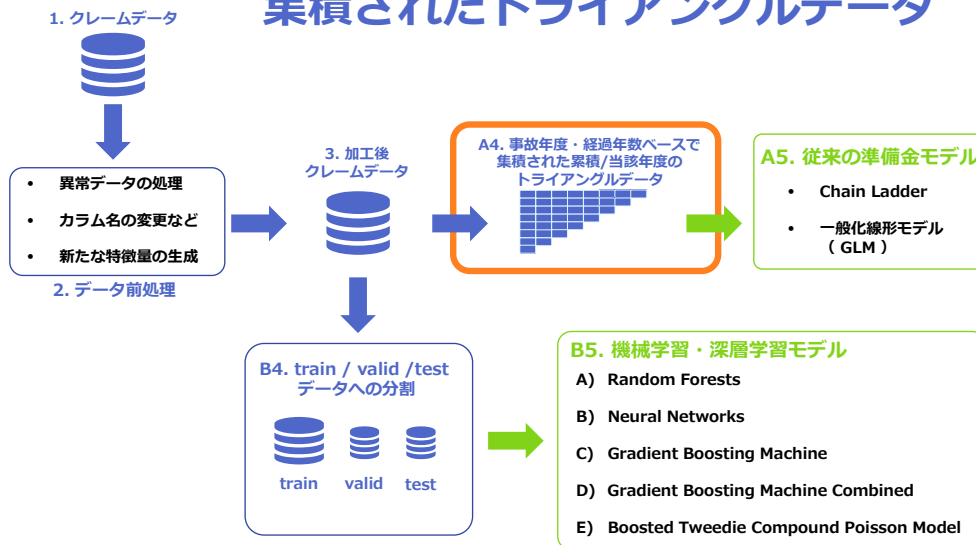
行方向に calendar_year 別に各クレームの履歴が並ぶ Long format



以上のプロセスを経まして、加工されたクレームデータができあがります。ここでは、そのデータの一部をごらんいただきます。こちらがデータの一例なのですが、クレームデータのユニークキーであります claim_number が、1000 ですね。少し見にくいかもしれませんが、左上が claim_number なのですが、全部 1000 だと思います。その 1000 のデータだけを取り出してきて、表示したものになります。先ほどからご説明させていただいておりますが、行方向に、経過年数ごとにこちらのクレームの累計が並ぶデータとなっています。

そして、左下の緑枠で囲んでいる、少し見にくいかもしれませんが、こちらですね。こちらが accident_year に対して calendar_year を示しているのですが、calendar_year が 1000 のものは accident_year が 2000 年なのですが、calendar_year が、2000 年から 2001 年、2002 年と、2016 年まで並んでいるということで、ロングフォーマットにきちんとなっていることがお分かりいただけるかと思えます。

A4. 事故年度・経過年数ベースで集積されたトライアングルデータ



それでは、データの前処理が終わりましたので、これから、従来の伝統的手法、機械学習・深層学習的手法の説明に移らせていただきたいと思います。まず、従来の準備金モデルに関しまして、こちらは、チェーンラダーに代表されますように、トライアングルデータを使用いたします。これまでデータの加工を通して得られておりますものは、ロングフォーマットのデータですので、そのデータをチェーンラダー等に適した形に集計をいたします。

A4. 事故年度・経過年数ベースで集積されたトライアングルデータ

R package `ChainLadder`

- Triangle data の作成 (右図)
- Long format から Triangle への変換が容易
- CL, Mack, GLMなどのモデリングが可能
- グラフ描画メソッド

developers
 Alessandro Carrato,
 Fabio Concina,
 Markus Gesmann,
 Dan Murphy,
 Mario Wuthrich,
 Wayne Zhang

<https://github.com/mages/ChainLadder>

```
library(ChainLadder)
library(tidy)
library(dplyr)
library(ggplot2)
library(string)

package loading

data <- read_csv("../input/preprocessed_data.csv")

long.data <- data %>%
  group_by(incident_year, dev) %>%
  summarise(inc_loss = sum(incurred, na.rm = TRUE),
            paid_loss_incr = sum(paid, na.rm = TRUE)) %>%
  ungroup() %>%
  right_join(
    expand_grid(
      incident_year = 1994:2010,
      dev = seq(1, 23, 1)
    )
  ) %>%
  group_by(incident_year) %>%
  mutate(paid_loss_cum = cumsum(paid_loss_incr)) %>%
  ungroup()

Wide format から Long format への変換

tri.data <- long.data %>%
  as_triangle(
    origin = 'incident_year',
    dev = 'dev',
    value = 'inc_loss'
  )

Long format から Triangle への変換
```

トライアングルデータを作成するに当たって、使っていたものはRなのですが、Rパッケージで「ChainLadder」という非常に有用なパッケージがございます。こちらを使用することで、ロングフォー

マットからトライアングルフォーマットへと簡単に変換することができます。また、こちらのパッケージなのですが、チェーンラダー、Mack モデル、GLM といった各種モデリング手法も、パッケージの中にそのような関数がございますので、使用することができ、従来のモデリング手法に関しては、非常に充実したパッケージとなっています。開発者ですけれども、Wuthrich など、有名なりザービング・アクチュアリーですね。そのような方々が開発されているので、パッケージとしての信頼度も、それなりに高いのではないかと個人的に考えています。

右側の図にありますものは、コードの全文ですね。一例ではなく、全文です。一番上段でパッケージをロードして、二段目でワイドフォーマットからロングフォーマットに変換する。そして、最後、ロングフォーマットからトライアングルデータへの変換は、チェーンラダーパッケージの「as.triangle」という関数を使うと一発でできるということで、ほとんど苦勞することなく、このようなデータテーブルの集計ができます。

A4. 事故年度・経過年数ベースで集積されたトライアングルデータ

Cumulative incurred claim amounts (累計発生保険金)

AY : 事故年度
DEV : 経過年数

	DEV	0	1	2	3	4	5	6	7	8	9	10
1994	3,349,291	12,636,268	15,528,994	16,853,432	19,486,003	23,638,743	26,049,777	22,302,587	24,397,688	24,933,717	24,911,070	
1995	15,419,283	33,114,401	38,965,892	44,346,883	47,827,439	50,605,938	51,443,543	49,901,125	50,610,693	53,132,498	55,163,766	
1996	17,050,090	32,717,244	37,631,768	43,438,143	53,308,928	59,875,376	61,934,993	55,356,013	55,174,437	54,449,055	56,455,203	
1997	18,109,382	36,311,893	42,193,245	47,942,155	54,385,836	62,809,117	64,505,637	57,849,342	59,040,249	61,069,922	63,718,647	
1998	23,224,197	40,547,829	49,915,629	53,369,040	57,961,887	63,625,294	65,452,004	54,825,999	57,955,603	60,559,700	75,246,730	
1999	20,662,821	41,786,198	50,543,856	56,933,371	64,536,369	69,340,364	66,099,963	59,456,547	64,982,799	74,388,405	85,785,796	
2000	19,961,458	41,954,468	52,016,559	56,266,583	60,314,760	61,464,735	61,332,185	60,911,821	72,758,312	80,898,584	80,758,390	
2001	18,756,606	45,065,326	57,924,080	67,158,163	72,664,016	80,639,724	81,425,794	82,361,209	96,238,127	99,601,436	96,311,717	
2002	20,804,971	46,534,770	58,164,792	67,168,870	75,388,954	81,217,320	83,228,525	94,574,915	101,665,869	100,099,948	91,286,509	
2003	23,272,575	46,129,306	55,251,952	67,031,626	72,819,485	85,460,733	95,738,649	101,964,701	97,287,524	84,590,684	86,237,834	
2004	21,393,949	48,047,041	62,660,447	73,738,929	93,583,614	111,397,008	110,585,142	101,843,285	95,818,006	96,621,687	97,787,807	
2005	13,296,154	35,928,219	49,225,566	63,527,170	80,880,226	91,624,723	85,446,968	77,265,401	85,015,842	88,176,759	91,468,746	
2006	22,256,213	48,097,250	62,743,804	83,357,713	96,147,273	104,089,468	87,485,357	86,172,299	86,916,292	91,663,741	92,572,661	
2007	19,586,963	47,169,049	66,896,340	76,261,682	84,225,171	85,973,038	79,791,545	79,610,555	89,264,285	92,468,291	NA	
2008	23,441,702	61,042,168	73,074,376	83,852,295	83,094,805	83,207,042	79,449,123	84,834,047	89,292,714	NA	NA	
2009	17,193,212	38,802,433	50,937,746	48,829,445	58,044,114	58,744,756	62,895,252	69,006,711	NA	NA	NA	
2010	1,136,121	2,133,857	1,612,655	4,906,603	6,539,149	6,406,980	7,270,584	NA	NA	NA	NA	

	11	12	13	14	15	16	17	18	19	20	21	22
24,621,797	25,022,897	24,387,234	45,165,897	25,602,134	26,600,867	27,041,658	25,066,399	24,432,428	25,594,678	25,406,514	25,439,338	
54,100,332	57,263,664	59,974,035	65,135,092	65,429,517	62,472,685	59,231,158	61,085,905	62,813,008	64,619,050	64,916,166	NA	
59,352,533	66,542,690	67,687,083	70,334,891	72,742,675	66,534,324	72,092,968	74,596,410	77,414,724	77,469,194	NA	NA	
65,514,138	70,388,924	75,164,139	77,160,609	71,599,617	74,604,751	78,832,768	81,194,716	80,992,013	NA	NA	NA	
80,495,913	83,517,518	82,614,866	74,757,578	75,258,337	75,124,198	82,043,819	84,125,463	NA	NA	NA	NA	
87,895,067	83,004,246	78,453,449	82,044,480	83,606,990	88,541,498	89,111,012	NA	NA	NA	NA	NA	
80,152,740	77,500,087	78,051,510	79,068,949	84,718,535	85,134,396	NA	NA	NA	NA	NA	NA	
87,811,466	90,407,519	93,950,595	102,675,617	104,518,066	NA	NA	NA	NA	NA	NA	NA	
92,826,509	94,825,982	102,458,998	105,439,493	NA	NA	NA	NA	NA	NA	NA	NA	
85,764,419	93,047,109	95,710,615	NA	NA	NA	NA	NA	NA	NA	NA	NA	
104,403,799	106,014,692	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
93,564,492	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	



そのようにできたものが、こちらは累計支払保険金のトライアングルデータなのですが、エクセルにペーストしただけなのですが、このような形で集計ができます。単位は、アメリカドルとなっています。観察データがあるものは、最終 calendar_year は 2016 年なのですが、見切れてしまっているので上と下に分割していますが、ご了承ください。この対角上が、calendar_year 2016 年に該当するという形になります。縦軸・横軸は、事故年度・経過年数という形になります。そして、先ほどが累計支払保険金ですが、同じように、累計発生保険金のトライアングルデータも簡単に作ることができます。

A4. 事故年度・経過年数ベースで集積されたトライアングルデータ

Incremental paid claim amounts (当該年度 支払保険金)

dev	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
1994	55,946	210,544	585,513	1,099,405	1,919,244	3,051,971	4,539,725	6,326,256	8,361,881	10,591,500	12,959,100	15,519,675	18,228,225	21,040,750	23,914,250	26,807,750	29,789,250	32,828,750	35,886,250	38,941,750	42,004,250	45,073,750	48,150,250	51,233,750
1995	217,300	760,800	1,564,100	2,458,200	3,489,100	4,694,800	6,019,300	7,419,600	8,849,700	10,365,400	11,922,700	13,587,800	15,336,600	17,145,100	19,000,300	20,900,300	22,843,000	24,817,500	26,813,000	28,820,500	30,840,000	32,871,500	34,915,000	36,970,500
1996	249,900	766,400	1,500,200	2,268,500	3,095,200	3,994,500	4,980,400	6,067,900	7,262,100	8,569,100	9,984,000	11,502,700	13,121,200	14,835,500	16,640,600	18,533,500	20,511,200	22,570,700	24,709,000	26,914,200	29,184,500	31,518,000	33,913,500	36,370,000
1997	284,800	820,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
1998	324,800	850,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
1999	369,800	880,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2000	419,800	910,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2001	474,800	940,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2002	534,800	970,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2003	599,800	1,000,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2004	670,800	1,030,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2005	747,800	1,060,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2006	831,800	1,090,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2007	923,800	1,120,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2008	1,024,800	1,150,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2009	1,136,800	1,180,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2010	1,261,800	1,210,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2011	1,400,800	1,240,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2012	1,554,800	1,270,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2013	1,726,800	1,300,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2014	1,917,800	1,330,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2015	2,128,800	1,360,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2016	2,361,800	1,390,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2017	2,617,800	1,420,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2018	2,898,800	1,450,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2019	3,207,800	1,480,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200
2020	3,545,800	1,510,800	1,488,000	2,235,200	3,004,500	3,829,800	4,736,100	5,748,400	6,881,700	8,159,000	9,595,300	11,195,600	12,964,900	14,898,200	16,990,500	19,237,800	21,637,100	24,185,400	26,880,700	29,720,000	32,701,300	35,822,600	39,083,900	42,475,200

Incremental incurred claim amounts (当該年度 発生保険金)

dev	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
1994	55,946	210,544	585,513	1,099,405	1,919,244	3,051,971	4,539,725	6,326,256	8,361,881	10,591,500	12,959,100	15,519,675	18,228,225	21,040,750	23,914,250	26,807,750	29,789,250	32,828,750	35,886,250	38,941,750	42,004,250	45,073,750	48,150,250	51,233,750
1995	154,920	1,085,110	2,180,280	3,258,550	4,369,820	5,524,090	6,730,360	7,984,630	9,288,900	10,643,170	12,047,440	13,501,710	15,005,980	16,560,250	18,164,520	19,818,790	21,523,060	23,277,330	25,081,600	26,935,870	28,840,140	30,794,410	32,798,680	34,852,950
1996	170,900	1,180,190	2,340,380	3,480,570	4,710,760	6,030,950	7,451,140	8,971,330	10,591,520	12,311,710	14,131,900	16,052,090	18,072,280	20,192,470	22,412,660	24,732,850	27,153,040	29,673,230	32,293,420	35,013,610	37,833,800	40,754,000	43,774,200	46,894,400
1997	189,900	1,290,190	2,520,380	3,740,570	5,060,760																			

た結果が、こちらのグラフとなります。最終累計支払保険金と最終発生保険金ですね。ごめんなさい、当該年度の支払保険金ですね。2009年度までの契約データが対象ですので、支払保険金は、2009年を過ぎますと、2012年以降くらいから減少傾向にあります。一方で、発生保険金ですけれども、トライアングルデータでもマイナスの場所があるとお話しさせていただきましたが、そちらを受けて、非常に変動幅の大きい、不安定なカーブを描きます。こちらは、後ほどスライドの最後の方のモデルの検証結果でご説明させていただきますのですが、支払保険金は安定した傾向にあって、モデリングがしやすい一方で、発生保険金は、ごらんのとおり不安定ですので、モデリングはそれほど簡単ではないデータになっています。

A5. 従来の準備金モデル

Chain Ladder (チェーンラダー法)

1. 累計 (支払 / 発生) 保険金に基づいた LDF (ロスディベロップメントファクター) を計算し、将来・最終累計 (支払 / 発生) 保険金を算出。
2. 各経過年数毎の LDF を算出するにあたり、AY 2010 年のトライアングルデータは、2009 年の written premium のうち 2010 年に earned となった契約に関するものであり、データ数が少ないため、計算対象から除外。
3. LDF の計算対象セルを、異常な値を除くように選択。



31

それでは、従来の準備金モデルの説明に入りたいと思います。本ワーキングパーティーでは、チェーンラダーと一般化線形モデルの2つを検討いたしました。まずはチェーンラダーに関しまして、簡単にご説明させていただきます。チェーンラダーに関しましては、トライアングルデータの累計支払・発生保険金に対しまして、ロスディベロップメントファクターを作り、それを基に、将来および最終累計支払・発生保険金を算出いたしました。

また、各経過年数ごとのロスディベロップメントファクターを算出するにあたりまして、事故年度が2010年度のデータは除外いたしております。その理由といたしましては、契約自体が2009年度までのものしかなかったため、2010年度に事故となったものは2009年度契約のもので、データ点数が非常に少なく、信頼性に欠けるものであったためということになります。また、同時に、これは非常に主観的なもので、いろいろ議論はあると思うのですが、ロスディベロップメントファクターが非常に異常な値となっていたものは、分析者の方で幾つか除外をいたしております。

では、次のページで、トライアングルデータに対して、どのセルをモデルのフィッティングに使用して、どのセルを予測したかをごらんいただきたいと思います。こちらの考え方は、後々機械学習・深層学習のモデルと比較する際にバックテストを行います。そちらにおいて非常に重要な考え方となってきます。

A5. 従来の準備金モデル

モデルのフィッティングと評価手法

CY 2016

Ultimate Loss

dev	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Estimated Loss
1994	62,916	3,242,110	9,825,894	2,028,119	4,216,119	6,411,827	7,079,367	11,581,186	17,070,145	13,489,658	16,929,951	16,781,180	17,029,969	18,327,600	19,093,126	20,279,471	22,077,907	22,889,761	21,178,292	21,009,112	21,009,112	21,009,112	1,328,760
1995	217,700	1,008,637	1,962,762	5,539,379	7,325,999	13,027,613	18,306,361	22,613,967	26,254,335	30,257,687	30,966,086	40,027,741	41,978,969	44,984,960	46,686,287	48,073,688	50,589,239	51,136,306	55,647,361	56,471,294	56,471,294	56,471,294	5,948,447
1996	506,960	1,940,467	3,191,783	5,241,452	7,263,268	8,404,949	12,009,669	16,008,787	22,612,249	29,193,213	32,053,620	36,413,011	44,964,414	46,812,391	51,902,400	54,146,004	60,986,628	62,181,923	65,766,349	66,086,969	66,086,969	66,086,969	5,776,690
1997	295,898	927,324	2,473,538	5,173,598	7,241,624	10,222,896	15,182,328	19,746,656	26,689,979	32,038,830	36,581,132	42,362,228	46,843,884	50,028,721	53,849,932	56,974,118	62,897,967	64,567,065	70,292,313	71,029,313	71,029,313	71,029,313	8,234,123
1998	273,884	1,386,224	2,443,138	5,096,868	6,368,787	11,387,958	15,386,849	20,046,262	25,268,218	30,476,817	36,214,486	38,878,428	39,699,227	57,048,861	62,545,883	63,786,167	65,157,940	66,086,969	66,086,969	66,086,969	66,086,969	66,086,969	78,277,228
1999	634,584	1,350,067	3,058,717	6,345,642	9,503,195	14,441,688	20,214,201	24,267,058	31,261,385	36,648,905	44,427,084	49,754,466	52,526,413	52,570,370	68,754,085	69,997,106	73,005,477	75,421,791	78,236,704	80,852,163	83,194,443	85,937,649	87,372,381
2000	424,734	1,142,023	3,317,161	6,449,612	10,308,137	13,946,420	17,893,621	21,822,024	31,246,044	41,161,088	47,229,268	56,020,249	62,249,356	69,332,100	71,972,390	74,771,628	77,762,978	80,485,106	83,079,368	85,482,236	87,748,631	89,794,181	10,560,488
2001	471,612	1,066,081	3,038,391	6,665,975	13,306,124	19,316,266	26,466,649	36,618,014	41,861,494	53,613,220	61,776,362	76,907,247	84,966,479	98,631,028	106,232,355	107,628,789	109,269,921	111,159,944	113,176,868	115,409,264	117,841,629	120,469,948	23,263,694
2002	701,296	2,231,124	6,096,110	12,369,762	19,286,174	27,336,907	37,482,021	50,261,114	61,169,740	79,775,327	87,715,242	106,966,789	119,775,327	146,266,479	157,322,036	177,923,921	182,041,921	183,509,192	185,443,872	189,941,773	192,803,724	195,499,158	17,726,225
2003	782,845	2,212,022	5,772,127	12,511,029	19,826,260	27,394,862	39,969,287	56,513,052	69,268,185	91,254,813	93,926,749	120,269,789	132,629,789	159,269,789	177,923,921	182,041,921	183,509,192	185,443,872	189,941,773	192,803,724	195,499,158	197,941,629	200,818,579
2004	526,627	1,370,467	3,008,518	6,021,538	12,801,702	21,408,280	29,683,115	40,008,320	52,576,790	61,967,190	71,611,121	86,026,064	97,026,218	116,026,218	127,026,218	146,026,218	157,026,218	168,026,218	179,026,218	190,026,218	201,026,218	212,026,218	42,898,493
2005	812,119	2,246,467	5,061,713	11,334,302	19,487,400	27,348,853	40,378,945	48,238,562	61,272,550	72,501,748	79,237,878	91,624,466	97,926,979	121,729,066	130,224,719	139,449,614	149,403,253	159,988,900	171,513,846	183,688,260	196,941,773	211,295,286	127,612,627
2006	476,100	1,212,068	3,076,879	6,162,789	13,821,993	24,481,287	38,378,945	55,826,132	77,269,790	99,216,910	121,164,110	143,112,310	165,060,510	187,008,710	208,956,910	230,905,110	252,853,310	274,801,510	296,749,710	318,697,910	340,646,110	362,594,310	112,026,460
2007	526,864	1,841,235	5,043,662	10,865,470	22,217,028	38,744,750	58,094,227	80,266,610	98,279,989	115,237,368	131,194,747	147,152,126	163,109,505	179,066,884	195,024,263	210,981,642	226,939,021	242,896,400	258,853,779	274,811,158	290,768,537	306,725,916	48,824,765
2008	476,100	1,212,068	3,076,879	6,162,789	13,821,993	24,481,287	38,378,945	55,826,132	77,269,790	99,216,910	121,164,110	143,112,310	165,060,510	187,008,710	208,956,910	230,905,110	252,853,310	274,801,510	296,749,710	318,697,910	340,646,110	362,594,310	112,026,460
2009	477,577	1,458,611	4,028,528	12,265,395	22,023,262	32,541,630	39,719,813	47,251,511	55,200,979	63,418,278	71,867,746	80,527,214	89,376,682	98,406,150	107,615,618	117,004,086	126,572,554	136,321,022	146,249,490	156,357,958	166,646,426	177,114,894	156,216,000
2010	572,067	1,661,438	4,120,817	8,618,396	16,488,400	27,195,904	40,607,908	56,819,912	75,821,916	97,523,920	121,825,924	143,727,928	163,229,932	180,331,936	196,033,940	211,335,944	227,237,948	242,839,952	258,141,956	273,143,960	287,845,964	302,247,968	83,961,244
2011	623,008	1,961,381	4,922,538	9,723,460	17,046,464	27,269,468	40,607,908	56,819,912	75,821,916	97,523,920	121,825,924	143,727,928	163,229,932	180,331,936	196,033,940	211,335,944	227,237,948	242,839,952	258,141,956	273,143,960	287,845,964	302,247,968	316,111,184

- 白色セルと黄色セルは、モデルのフィッティングに使用するデータ
- 黄色セルは、Calendar Year (CY) = 2016 時点のセルに該当
- 緑色セルが予測すべきセル



本ワーキングパーティーでは、Calendar Year (CY) によって、フィッティングに使用するセルと予測すべきセルを分けておきます。例えば、こちらのスライドの図ですと、CYが2016年までのデータをフィッティング用のセル、それ以降を予測すべきセルとした場合のものになります。白色と黄色のセルが、フィッティングに使用しますデータです。緑色の文字のセルが、予測すべきセルになります。これはもう予測後のものなので、埋まってしまっているのですけれども、そこが予測すべきセルになります。黄色のセルは、対角上にあることから、CYが2016年のものと分かりいただけるかと思えます。こちらは全て、先ほどお話いたしましたとおり、累計ベースのデータで、こちらは支払保険金のものになります。同様に、発生保険金に関してもモデリングをすることになります。

以上のように、2016年から2006年にかけて、CYを基準としまして、トライアングルデータをフィッティングと予測用に分けまして予測してみた結果が、次のページのようにになります。ここでは、モデルの当てはまり具合、バックテストに関してはご紹介しません。後ほど、最後の機械学習・深層学習のモデルと併せて、ご説明をさせていただきます。

A5. 従来の準備金モデル

評価結果

- 最終累計支払保険金
- 最終累計発生保険金



- **Paid CL**
2012年のspikeを除けば、安定した見積もり額となっている。
- **Incurred CL**
LDFが2008年までは増加しているが、その後、2011年まで大きく減少するため、大きく変動している。



34

こちらが、チェーンラダーの最終累計支払保険金と最終累計発生保険金を図示したものととなります。このグラフのx軸、「Valuation year」とありますが、こちらが、トライアングルデータのセルをフィッティング用と予測用に分ける時の基準となっています、CYのことです。「Calendar year」と書けばよかったのですが、すみません。こちらのグラフから読み取れますが、支払保険金の最終累計は、2012年にちょっとしたスパイクがあるのですが、そちらを除けば、大体収束していき、安定した見積もりとなっています。

一方で発生保険金は、非常に大きく変動しています。これが非常にモデリングしにくい理由になるのですが、実績値として、2010年以降に発生保険金が大きくマイナスとなる年が、前のスライドにあったと思います。それらを考慮したロスディベロップメントファクターを使って計算すると、このように非常にボラティリティーの激しいカーブになってしまいます。以上が、チェーンラダーの結果の説明になります。

A5. 従来の準備金モデル

GLM（一般化線形モデル）

1. 当該年度（支払 / 発生）保険金（incremental）を、GLM で予測する目的変数とした。
2. 当該年度支払保険金の GLM には、累積分布関数のあてはまりの良さから、log リンク関数とガンマ分布を採用。
3. 当該年度発生保険金の GLM には、負の実績値が存在したため、恒等リンク関数と正規分布を採用。

```
library(ChainLadder)
library(tidyverse)
library(ggplot2)
library(stringr)

load('./input/tri_paid.RData')
load('./input/tri_inc.RData')
glm.paid <- glmReserve(triangle = tri.paid, var.power = 2, link.power = 0)
glm.inc <- glmReserve(triangle = tri.inc, var.power = 0, link.power = 1)
```



35

次に、GLM に関して説明をさせていただきます。GLM の場合は、チェインラダー法と異なり、当該年度の支払保険金・発生保険金を予測する形となります。先ほどが Cumulative なのに対して、今回は Incremental でやるという形になります。支払保険金の分布には、log リンク関数とガンマ分布を採用いたしました。一方で発生保険金の分布には、先ほどからご覧いただいておりますが、マイナスの値ですね。負の値がございますので、恒等リンク関数と正規分布を採用いたしました。下の図は、R のパッケージ、先ほどご紹介した ChainLadder です。そちらを使用いたしまして、incremental なトライアングルデータに対して GLM を適用するコードとなります。ものの 2 行で終わってしまうのです。非常に楽です。簡素なコードでできるということになります。

A5. 従来の準備金モデル

当該年度支払保険金 $C_{i,j}$ の場合

		dev																				
		0	1	...	j	...	21	22														
AY	1994																					
	1995																					
	...																					
	2009																					
	2010																					

- 事故年度と経過年数をリンク関数の説明変数とした GLM

$$E(C_{i,j}) = \mu_{i,j}$$

$$\ln(\mu_{i,j}) = \mu + \alpha_i + \beta_j$$

α_i : 事故年度 i に対応する係数
 β_j : 経過年数 j に対応する係数



GLM におけるリンク関数のご説明を、もう少しさせていただきます。本検討では、個別クレームごとに GLM を組み立てているわけではなく、トライアングルデータに対してやっておりますので、集積されてしまっているわけですね。ですので、集積された値が持つ特徴量といえますか、そのようなものは非常に限られておりまして、結局、アクシデントイヤーとディベロップメントイヤー、つまり経過年数ですね。そちらを2つの説明変数として、リンク関数とひもづけています。

A5. 従来の準備金モデル

CY 2016

Ultimate Loss

dev	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
AY	1994	65,945	131,110	193,694	2,030,374	4,792,346	6,413,007	6,976,787	11,961,166	15,361,145	15,983,076	16,258,111	16,751,180	17,038,699	17,351,400	18,081,200	20,055,200	20,974,872	22,037,627	23,189,927	23,934,122	24,202,021	24,202,021	24,202,021	
	1995	11,760	1,208,637	1,962,762	5,839,279	7,326,899	13,024,579	15,897,613	18,000,301	22,612,967	28,254,305	30,297,697	30,860,886	40,077,741	41,978,969	44,964,580	46,086,267	48,973,688	50,296,019	51,336,360	55,647,381	56,477,294	57,026,100	58,274,749	
	1996	60,060	1,241,407	2,191,763	5,241,403	7,625,289	9,454,833	11,068,699	15,000,387	22,822,249	28,262,353	30,952,028	40,101,233	44,904,474	46,823,293	51,308,403	54,160,084	58,905,676	61,793,053	65,926,794	69,495,669	70,929,251	70,929,251	70,929,251	
	1997	59,880	82,534	2,473,338	5,179,058	7,241,824	10,222,896	15,102,328	18,748,050	25,689,770	32,028,820	38,955,132	42,382,228	48,843,884	50,028,721	53,846,532	58,874,118	62,897,887	64,567,065	70,293,183	73,211,221	74,232,459	76,151,300	76,151,300	
	1998	17,886	1,380,338	1,896,868	6,438,787	11,887,068	14,386,849	20,996,862	26,786,216	38,476,817	48,436,480	48,436,480	58,699,027	57,048,081	60,694,003	65,786,147	67,597,940	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661
	1999	64,584	2,350,087	3,556,717	6,349,562	9,920,195	14,443,688	20,234,501	24,207,258	31,781,385	38,646,905	44,073,004	49,764,466	57,256,433	62,076,210	68,754,085	69,601,106	73,920,427	76,921,111	81,247,311	82,115,819	85,215,819	86,476,111	86,476,111	
	2000	40,724	1,742,021	3,517,861	6,495,812	10,304,137	13,749,420	17,695,487	22,022,094	31,766,044	41,161,488	47,302,669	50,201,340	60,976,076	67,244,556	69,932,266	73,072,261	77,620,000	77,620,000	81,781,888	84,392,477	86,292,425	87,512,689	89,032,462	
	2001	67,462	1,266,288	3,908,349	9,499,576	13,920,894	19,391,086	24,946,046	30,693,014	41,663,283	51,613,200	61,372,267	65,571,747	74,466,479	78,020,024	80,232,105	85,022,000	88,000,000	90,771,280	92,332,888	94,947,864	96,986,664	98,647,864	100,000,000	
	2002	69,286	2,211,124	4,096,116	12,260,782	16,230,749	23,866,174	32,398,907	40,400,114	51,169,740	71,096,399	79,775,327	82,715,242	97,322,024	102,000,000	105,000,000	108,000,000	112,000,000	116,000,000	120,000,000	124,000,000	128,000,000	132,000,000	136,000,000	
	2003	60,940	2,212,021	4,172,127	7,651,020	10,910,913	15,024,806	20,024,806	25,024,806	31,024,806	38,024,806	45,024,806	52,024,806	60,024,806	68,024,806	76,024,806	84,024,806	92,024,806	100,024,806	108,024,806	116,024,806	124,024,806	132,024,806	140,024,806	
	2004	61,110	2,246,487	4,261,713	11,314,362	15,487,058	21,548,810	28,786,945	40,236,262	51,737,503	72,501,748	79,107,875	83,224,466	97,828,045	98,818,810	102,883,820	108,943,503	112,374,850	119,442,717	123,332,051	128,162,288	132,076,115	133,389,864	133,389,864	
	2005	60,642	1,811,867	3,286,318	8,026,528	12,881,737	17,428,113	22,324,367	28,511,021	35,916,216	44,561,748	48,207,881	50,000,000	58,248,248	62,848,248	66,878,248	70,328,248	74,298,248	78,678,248	83,468,248	88,668,248	94,268,248	100,268,248	106,668,248	
	2006	60,844	1,841,215	3,543,362	10,868,470	22,117,020	28,783,219	36,648,700	46,028,217	57,028,217	69,513,624	79,513,624	86,262,418	92,246,415	98,277,124	104,177,504	110,138,283	116,162,629	122,248,540	128,384,540	134,570,540	140,806,540	147,092,540	153,428,540	
	2007	60,920	2,222,021	4,107,470	9,767,708	13,832,053	23,294,188	30,522,475	40,243,012	52,208,770	66,448,247	82,028,247	92,228,247	98,228,247	103,228,247	108,228,247	113,228,247	118,228,247	123,228,247	128,228,247	133,228,247	138,228,247	143,228,247	148,228,247	
	2008	67,370	1,904,631	4,400,518	12,968,202	21,026,266	32,464,663	39,719,013	47,251,111	57,028,217	69,421,754	74,893,088	78,968,728	80,879,472	81,808,668	82,869,472	83,966,668	85,099,868	86,268,668	87,473,668	88,714,668	89,992,668	91,307,668	92,658,668	
	2009	67,920	1,958,861	4,310,617	9,610,390	14,696,438	23,337,975	29,400,259	37,028,217	47,307,259	59,818,189	69,812,478	66,167,774	71,232,631	71,052,410	69,961,749	68,038,057	68,111,118	61,543,589	67,134,898	72,889,720	78,889,720	84,889,720	90,889,720	
	2010	66,201	1,768,211	4,022,208	9,721,257	13,276,008	18,604,965	24,008,217	29,008,217	34,008,217	39,008,217	44,008,217	49,008,217	54,008,217	59,008,217	64,008,217	69,008,217	74,008,217	79,008,217	84,008,217	89,008,217	94,008,217	99,008,217	104,008,217	

CY 2006

Ultimate Loss

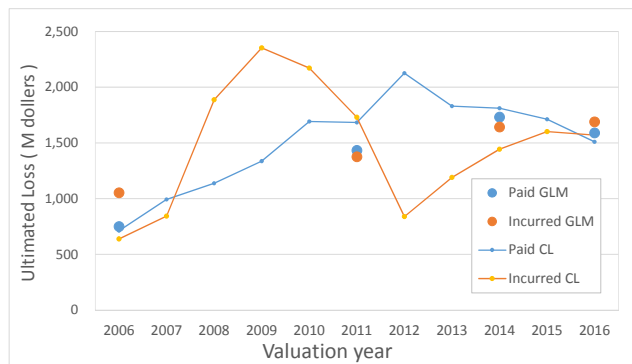
dev	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
AY	1994	65,945	131,110	193,694	2,030,374	4,792,346	6,413,007	6,976,787	11,961,166	15,361,145	15,983,076	16,258,111	16,751,180	17,038,699	17,351,400	18,081,200	20,055,200	20,974,872	22,037,627	23,189,927	23,934,122	24,202,021	24,202,021	24,202,021
	1995	11,760	1,208,637	1,962,762	5,839,279	7,326,899	13,024,579	15,897,613	18,000,301	22,612,967	28,254,305	30,297,697	30,860,886	40,077,741	41,978,969	44,964,580	46,086,267	48,973,688	50,296,019	51,336,360	55,647,381	56,477,294	57,026,100	58,274,749
	1996	60,060	1,241,407	2,191,763	5,241,403	7,625,289	9,454,833	11,068,699	15,000,387	22,822,249	28,262,353	30,952,028	40,101,233	44,904,474	46,823,293	51,308,403	54,160,084	58,905,676	61,793,053	65,926,794	69,495,669	70,929,251	70,929,251	70,929,251
	1997	59,880	82,534	2,473,338	5,179,058	7,241,824	10,222,896	15,102,328	18,748,050	25,689,770	32,028,820	38,955,132	42,382,228	48,843,884	50,028,721	53,846,532	58,874,118	62,897,887	64,567,065	70,293,183	73,211,221	74,232,459	76,151,300	76,151,300
	1998	17,886	1,380,338	1,896,868	6,438,787	11,887,068	14,386,849	20,996,862	26,786,216	38,476,817	48,436,480	48,436,480	58,699,027	57,048,081	60,694,003	65,786,147	67,597,940	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661	70,951,661
	1999	64,584	2,350,087	3,556,717	6,349,562	9,920,195	14,443,688	20,234,501	24,207,258	31,781,385	38,646,905	44,073,004	49,764,466	57,256,433	62,076,210	68,754,085	69,601,106	73,920,427	76,921,111	81,247,311	82,115,819	85,215,819	86,476,111	86,476,111
	2000	40,724	1,742,021	3,517,861	6,495,812	10,304,137	13,749,420	17,695,487	22,022,094	31,766,044	41,161,488	47,302,669	50,201,340	60,976,076	67,244,556	69,932,266	73,072,261	77,620,000	77,620,000	81,781,888	84,392,477	86,292,425	87,512,689	89,032,462
	2001	67,462	1,266,288	3,908,349	9,499,576	13,920,894	19,391,086	24,946,046	30,693,014	41,663,283	51,613,200	61,372,267	65,571,747	74,466,479	78,020,024	80,232,105	85,022,000	88,000,000	90,771,280	92,332,888	94,947,864	96,986,664	98,647,864	100,000,000
	2002	69,286	2,211,124	4,096,116	12,260,782	16,230,749	23,866,174	32,398,907	40,400,114	51,169,740	71,096,399	79,775,327	82,715,242	97,322,024	102,000,000	105,000,000	108,000,000	112,000,000	116,000,000	120,000,000	124,000,000	128,000,000	132,000,000	136,000,000
	2003	60,940	2,212,021	4,172,127	7,651,020	10,910,913	15,024,806	20,024,806	25,024,806	31,024,806	38,024,806	45,024,806	52,024,806	60,024,806	68,024,806	76,024,806	84,024,806	92,024,806	100,024,806	108,024,806	116,024,806	124,024,806	132,024,806	140,024,806
	2004	61,110	2,246,487	4,261,713	11,314,362	15,487,058	21,548,810	28,786,945	40,236,262	51,737,503	72,501,748	79,107,875	83,224,466	97,828,045	98,818,810	102,883,820	108,943,503	112,374,850	119,442,717	123,332,051	128,162,288	132,076,115	133,389,864	133,389,864
	2005	60,642																						

年、全部のCYに対してバックテストのような検討を行ったわけではなくて、2006年、2011年、2014年、2016年と、4つの年度だけ選んでやる形になっています。これは、先ほどチェーンラダーの分析結果をお見せしたと思うのですが、その年度以外だとかなり予測がぶれぶれだということから、比較的安定した年度を恣意的に選んでいます。

A5. 従来の準備金モデル

評価結果

- 最終累計支払保険金
- 最終累計発生保険金



- 本 WP では、GLM の計算に関しては、valuation year が、**2006, 2011, 2014 及び 2016 年**の場合のみ実施。
(後述する機械学習・深層学習的手法においても同様)
- 4 種の valuation year において、CL と全体的な傾向は似通っている。



38

次に、チェーンラダーとの比較結果をごらんいただこうと思います。こちらが、チェーンラダーの結果と GLM の結果をまとめたものとなります。両モデルの最終累計支払保険金と最終累計発生保険金ですが、両モデルの2006年、2011年、2014年、2016年。こちらに関しましては、GLM もチェーンラダーも、比較的似通ったような傾向にあることがごらんいただけるかと思います。

本スライドまでが、従来手法の説明となります。これ以降は、機械学習・深層学習的手法の説明に移らせていただきます。まずはじめに、これらのモデルで共通に利用いたしますフレームワーク、validation ですね。先ほどからお話ししておりました。そちらに関しまして、説明をさせていただきます。

B. 機械学習・深層学習モデルの枠組み

データの粒度

- ・ トライアングルに集積したデータを使用する伝統的手法と異なり、機械学習・深層学習を適用したモデルを作成する場合のデータは、「各年度の各クレーム毎」のものとなる。
- ・ また、目的変数は当該年度「支払/発生」保険金 (incremental)

Year	Claim_Count	IBNR	IBNR	IBNR	IBNR	IBNR	IBNR	IBNR	IBNR	IBNR	IBNR	IBNR
2000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2001	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2002	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2003	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2004	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2005	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2006	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2007	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2008	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2009	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2010	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2011	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2012	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2013	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2014	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2015	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2016	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2017	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2018	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2019	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2020	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

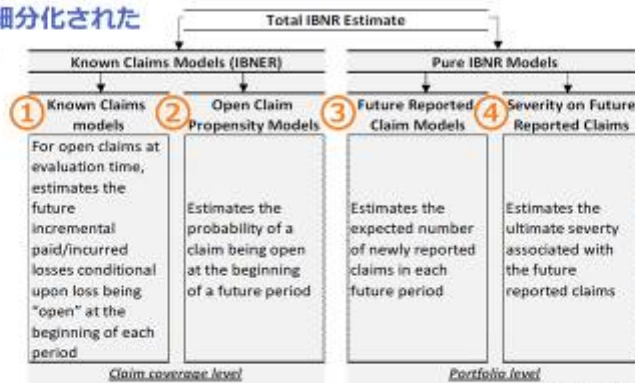
データの粒度は伝統的手法で使用するものと比較すると細かい



ごめんなさい。その前に、データの粒度に関してご説明させていただきます。使用するデータの粒度なのですが、先ほどご説明いたしましたとおり、トライアングルデータとは異なります。クレームごと、かつ経過年数ごとの、粒度の細かいものとなります。また、機械学習・深層学習モデルが予測する対象としますのは、GLMと同様に、累計ベースではなく、incremental な当該年度の支払保険金・発生保険金となります。

B. 機械学習・深層学習モデルの枠組み

4 つに細分化されたモデル



③④ IBNR に対しては、機械学習・深層学習を適用していないため、本報告では記載しません。

- ・ 既報告損害に対する支払備金 (IBNR) と 既発生未報告損害に対する支払備金 (IBNR) に分けてモデリングを行い、最後に統合する。

- ①② IBNR の場合、「クレーム請求が終了しているか、していないかという確率モデル」と、「終了していないという条件つきでの支払、発生保険金の回帰モデル」とで推定を行う
- ③④ IBNR の場合、「クレーム請求数」と「支払、発生保険金の回帰モデル」をもとに推定を行う



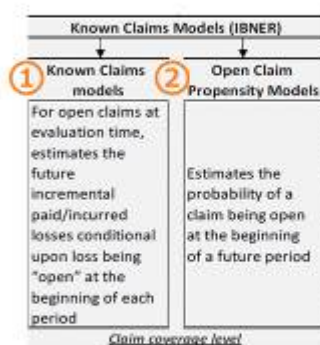
では、先ほど言いかけましたフレームワークに関して、ご説明をさせていただきます。支払備金のモデルなのですが、2つに大きく分けてモデリングすることになります。1つ目が、既に報告済みのク

クレームに対する支払備金、IBNER に対するモデルとなります。このモデルを、本ワーキングパーティーでは「Known Claims Models」と呼称しておりました。こちらを、次のスライド以降でご説明いたしますが、KCM および OPM というモデルに分けることとなります。KCM は、クレーム請求が終了していないという条件付きでの支払保険金・発生保険金の値を回帰するモデルになっておりまして、一方で OPM は、クレーム請求が終了しているか、終了していないかという確率を表すモデルとなります。

2つ目は、発生はしているのですが、未報告のクレームに対する支払備金、IBNER に対するモデルとなります。こちらも少し変わった呼び方で、一般的なのでしょうか？ピュア IBNR と呼んでおりまして、クレーム数とクレーム損害額を基にした Frequency-Severity model、FD 法のモデルとなります。こちらに関しましては、ワーキングパーティーもなかなか締め切りまでの時間があまりなく厳しかったものですから、機械学習・深層学習の手法を適用しておりません。ですので、本発表では、こちらの報告は割愛させていただきます。とは言いまして、最終的な予測値を算出する場合には、以上の2つのモデルの予測値を合算する形となります。

B. 機械学習・深層学習モデルの枠組み

既報告損害に対する支払備金 (IBNER)



KCM (Known Claims Models)

- ・ クレーム請求が終了していないという条件付きモデル
- ・ 将来における当該年度の支払/発生保険金を算出する
- ・ 回帰問題 として取り扱う

OPM (Open Propensity Models)

- ・ クレーム請求が終了していない確率を算出するモデル
- ・ 前年度の支払備金が0であれば、終了していると見なす
- ・ 分類問題 として取り扱う

**KCM 及び OPM に対して、
機械学習・深層学習的手法を適用**



42

こちらのスライドでは、もう少し詳しく、既報告損害に対します支払備金の予測モデルの説明をさせていただきますと思います。先ほどの KCM に関しましては、クレーム請求が終了していないという条件付きのもとで、将来における当該年度の支払/発生保険金を算出することとなります。発生保険金そのものを予測するモデルですので、KCM は回帰問題となります。

一方で OPM に関しましてですけれども、こちらは、クレーム請求が終了しているか、していないかという確率を算出するモデルですので、二値の分類問題として取り扱うこととなります。ただ、教師データとして使います実績値として、クレーム請求が終了しているか、していないかというものに関しましては、前年度の備金がゼロであれば終了していると思なすというルールを設けて、データ上で判断いたしております。こちらに関しましては、当然例外もあるかとは思うのですが、例えばオープンするなど、いろいろなケースもあると思うのですが、データから得られている情報が限られておりますため、そのような形で判断いたしております。

B. 機械学習・深層学習モデルの枠組み

i : 事故年度
 j : 報告遅延年数
 N_{ij} : 事故年度 i 、報告遅延年数 j におけるクレームの数
 \mathcal{F}_t : t 時点で判明している情報
 v : 各クレーム
 $C_{ij|k}^{(v)}$: 事故年度 i 、報告遅延年数 j 、クレーム v の 経過年数 k における (支払/発生) 保険金

事故年度 i の t 年度における最終累計 (支払/発生) 保険金は、

$$\sum_{j=0}^{t-i} \sum_{v=1}^{N_{ij}} \left(\sum_{k=0}^{t-(i+j)} C_{ij|k}^{(v)} + \sum_{k>t-(i+j)} E \left[C_{ij|k}^{(v)} \mid \mathcal{F}_t \right] \right)$$

とかける。(Wuthrich 2016)

■ ここで、KCM 及び OPM の考え方にそって上式を拡張した、既発生クレームの最終累計 (支払/発生) 保険金は、

$x_{ij|k}^{(v)}$: 事故年度 i 、報告遅延年数 j 、クレーム v の 経過年数 k におけるクレームの特徴量
 $OPM_{j+k} \left(x_{ij|k}^{(v)} \right)$: 事故年度 i 、報告遅延年数 j 、経過年数 k のクレーム v の請求が終了していない確率
 $KCM_{j+k} \left(x_{ij|k}^{(v)} \right)$: 事故年度 i 、報告遅延年数 j 、経過年数 k のクレーム v の請求が終了していない条件付き $i+j+k$ 年度におけるクレームの期待 (支払/発生) 保険金

事故年度 i の t 年度における最終累計 (支払/発生) 保険金は、

$$\sum_{j=0}^{t-i} \sum_{v=1}^{N_{ij}} \left(\sum_{k=0}^{t-(i+j)} C_{ij|k}^{(v)} + \sum_{k>t-(i+j)} KCM_{j+k} \left(x_{ij|k}^{(v)} \right) \times \frac{OPM_{j+k} \left(x_{ij|k}^{(v)} \right)}{OPM_{j+k} \left(x_{ij|k=j}^{(v)} \right)} \right)$$

とかける。(Golfin 2017)



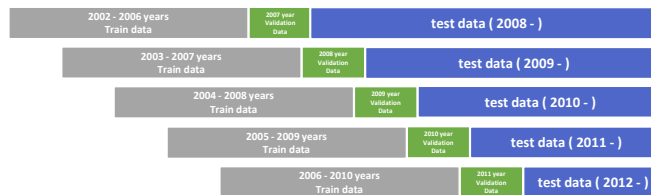
以上の KCM と OPM の考え方を数式を交えて説明しているものが、こちらのスライドとなります。時間の関係上、詳しく数式の中身を追うことはできませんが、基本的に、こちらのスライドの上部ですね。青枠内にありますようなノーテーションと設定のもとで、事故年度 i の最終累計支払/発生保険金は、スライド中段のような数式で書くことができます。括弧の中の左側の項が、実績値、すなわち既に分かっている値、右側の項が予測部に対応する項となっています。

ここまでは、特に KCM・OPM の枠組みを一切入れたものにはなっていませんが、前のスライドでご説明いたしました KCM・OPM の考え方を盛り込んで拡張いたしますと、スライド下段のような式にすることができます。下段の式におけます青色の部分ですね。見にくくて申し訳ございません。そちらの方は、 t 年度以降の将来予測値に対応する部分となります。数式の形から、クレーム請求が終了していないという条件のもとで KCM による保険金額が計算されるモデルであることが、雰囲気でお分かりいただければと思います。時間の関係上、数式の説明はこれ以上はできないのですけれども、興味のある方は、参照論文等をご紹介しますので、おっしゃっていただければと思います。

B4. データの分割

機械学習・深層学習モデルにおけるハイパーパラメータの決定

- 正則化 GLM における正則化パラメータのように、機械学習・深層学習モデルは各々のモデル固有のパラメータをもつ
→ 「ハイパーパラメータ」と呼ぶ
- test data 以外のデータを train data と validation data に分割し、train data で fitting を行ったモデルが、validation data において精度がでるような、ハイパーパラメータを探索する
- 時系列性のある支払備金の予測モデルのような場合は、例えば、下図のように、年度 (CY) で test data, train data, validation data を分割する



45

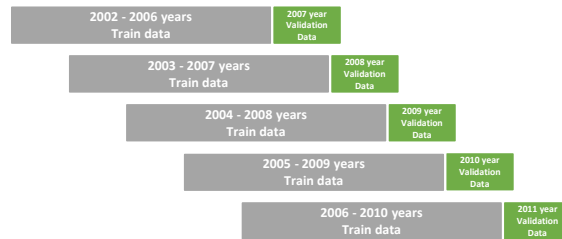
それでは、機械学習・深層学習のモデルの詳細に関して、説明をいたします。最初に、先ほどから validation と申し上げておりますが、なぜデータを分割する必要があるのかということに関しまして、ご説明をさせていただきます。このスライドにも記載させていただいておりますとおり、機械学習・深層学習モデルは、ハイパーパラメータというパラメータがございます。そちらを決定するためには、データを分割する必要があります。

聞き慣れない方には、「ハイパーパラメータって何だろう」と思われる方もいるとは思いますが、例えば、正則化 GLM の場合、罰則項に対応する GLM の正則化パラメータをどうしても決めなければいけない、どのパラメータがいいかを判断するとき、このような枠組みが必要になるとお考えいただければと思います。正則化 GLM のハイパーパラメータの数は、1つや2つなのですが、機械学習・深層学習の場合は、これらのハイパーパラメータの数が非常に多くなります。最低でも、3つから4つ程度あります。それらを決定したうえで、改めてモデルのフィッティング、予測を行うこととなります。

実際にどのようにやるかということなのですが、予測に使用するデータ、これは、先ほどトライアングルで出てきた緑色のセルに該当するようなイメージですね。そちらを test data と読んでおいて、test data 以外のデータを、更に train data、validation data と呼ばれるデータに分割いたします。下の図が、それに対応しております。分割のしかたなのですが、今回のように時系列性のあるデータの場合は、CY で分割することになります。ハイパーパラメータの決定に必要なものは、実際のところ、train data と validation data の2つのみになります。test data は予測する場所ですので。

B4. データの分割

本検討におけるハイパーパラメータの決定



- 5つの validation data における評価指標（後述）の平均値がベストとなるようなハイパーパラメータを各モデルにて探索し採用
- KCM と OPM は、それぞれ異なるハイパーパラメータを持ちうる
- KCM は、 $RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$ を、
OPM は、AUC（受信者感度曲線の下側面積）を、
モデル精度を判断する評価指標として採用



46

次のページで、実際にどのように分割したかをご紹介します。本ワーキングパーティーでは、5とおりのデータの分割を行いました。それぞれで train data のモデルのフィッティングを行っています。フィッティングを行いましたモデルの性能は、では、どのように評価するか。それを train data で行っては、意味がないですね。train data の目的値に合わせて行っていますので。そのため、モデルの性能は validation data で評価いたします。評価した性能の結果は5つ分ありますので、5つの平均値がベストとなるようなハイパーパラメータを、各モデルにおいて探索することになります。

探索の行い方ですけれども、幾つか手法がございます。例えば、一定の範囲内でハイパーパラメータの取りうる値の組み合わせを全点探索する、グリッドサーチというものがありますけれども、非常に計算コストが高いです。ですが、本ワーキングパーティーでは、そちらを採用いたしました。後ほど、各モデルのハイパーパラメータの探索範囲を、各モデルの説明をしながらご紹介いたします。また、KCM と OPM は、異なるハイパーパラメータを持つことを許した形となっております。KCM は回帰モデルですので、精度評価には RMSE を、OPM は分類モデルですので AUC を、モデル精度を判断する評価指標として採用いたしました。

B4. データの分割

データ粒度は、各クレーム毎だが、データの分割範囲をトライアングルデータ上で示すと・・・

黄色のセルは train data
 緑色のセルは validation data
 緑文字のセルは test data

2002 - 2006 years Train data													2007 year Validation Data									
年	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1999	15,650	19,110	940,080	2,038,179	4,251,197	4,782,260	8,427,287	17,975,307	11,982,126	14,311,101	14,481,029	16,966,111	16,711,167	17,975,229	18,957,467	19,538,229	17,159,947	17,438,221	17,518,653	17,685,000	17,699,382	17,699,382

2006 - 2010 years Train data													2011 year Validation Data									
年	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1999	15,650	19,110	940,080	2,038,179	4,251,197	4,782,260	8,427,287	17,975,307	11,982,126	14,311,101	14,481,029	16,966,111	16,711,167	17,975,229	18,957,467	19,538,229	17,159,947	17,438,221	17,518,653	17,685,000	17,699,382	17,699,382

トライアングルデータ上で、これまでお話いたしましたデータの分割のしかたを見てみます。そうしますと、先ほどのスライドの最初の分割パターンですと、このスライドの上図のような形になります。黄色のセルが train data、緑色のセルが validation data ということ、1年分なのですね、validation data は。そのような形でデータの分割をすることになります。

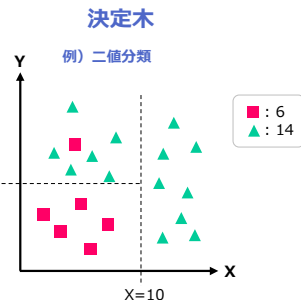
それでは、機械学習・深層学習モデルの説明に、いよいよ入っていききたいと思います。一旦、各種アルゴリズムの説明をひととおりいたします。お時間の関係がありますので、比較的駆け足になってしまうと。

B5. 機械学習・深層学習モデル

機械学習・深層学習のモデルトレンド

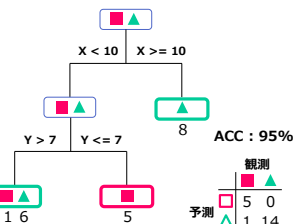
1. 決定木系のアルゴリズム

- Random Forest
- Boosting Machine



2. ニューラルネットワーク系のアルゴリズム

- DNN
- CNN
- RNN / LSTM / GRU



本ワーキングパーティーで使用しました機械学習・深層学習のモデルは、大きく分けまして、決定木系の機械学習アルゴリズムと、ニューラルネットワークという深層学習系のアルゴリズムとなります。昨今のデータサイエンスにおけるモデルのトレンド、はやりなのですけれども、決定木系のアルゴリズムが非常にはやっております。例えば、Random Forest や Boosting Machine など、そのあたりが非常にはやっております。ニューラルネットワーク系のアルゴリズムですと、ディープニューラルネットワーク、畳み込みニューラルネットワーク、再帰型ニューラルネットワークなどが、よく利用されております。本ワーキングパーティーで実際に使用しましたものは、こちらのアルゴリズムとなっています。赤枠で囲った部分ですね。後ほどニューラルネットワークに関しましては、ベーシックな説明を簡単にさせていただきますので、ここでは決定木系のアルゴリズムの基となる決定木に関して、右図で簡単にご説明をさせていただきます。

ここで例として出しておりますのは、三角と四角の2つのクラスが、X と Y という値で決まっているようなデータを、線形のアルゴリズムでうまく分類モデルを作って分割しようといいたしますと、難しいことがお分かりいただけるかと思えます。一方で、決定木系ですと、右下の図にありますように、最初に X が 10 以上か、それより小さいか、次に Y が 7 より大きいのか、それ以下かという分け方をいいたしますと、少し恣意的な例ではありますが、正解率 95% ということで、うまく三角と四角の2つのクラスを分類できるようになります。つまり、従来の線形モデルではうまく分類できなかったものが、決定木系の非線形アルゴリズムでは非常にきれいに分類できるということで、決定木系アルゴリズムのパワフルなところが、かいま見えるかと思えます。

B5. 機械学習・深層学習モデル

A) Random Forests (Breiman 2001)

Bagging (アンサンブル学習) に基づくアルゴリズム

X から y を予測 (分類・回帰) する問題に対して、データを M 通りサンプリング (ブートストラップ) し、各サンプリングデータに対して別々の決定木でモデリングをする。

決定木の各分岐点において、全特徴量の中から一定数の特徴量をランダムに選択することで、M 個ある決定木の予測値間の相関を低減し、予測値の分散を改善させている。これによって、汎化性能が向上している。

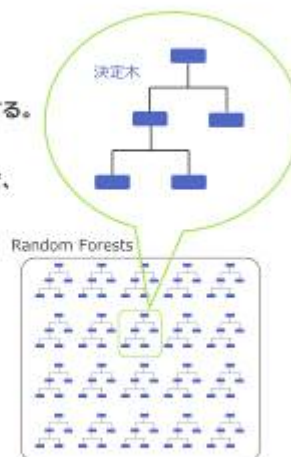
予測値を $y_M(X)$ とすると、

$$y_M(X) = \frac{1}{M} \sum_{m=1}^M z_m(X)$$

X : 選択された特徴量ベクトル

M : モデルの数 (ブートストラップの回数)

$z_m(X)$: 各データにおける決定木の予測関数



それでは、最初に、Random Forest に関して説明をいたします。こちらは、決定木を幾つも使用したものでして、「Bagging」と呼ばれるアンサンブル学習に基づいたアルゴリズムとなります。アンサンブル学習という言葉をお聞きになったことがあるかと思えますけれども、アンサンブル学習には Bagging 以外にも、Boosting というものもございます。

特徴としては 2 点ほどございます。1 つ目はモデルのアルゴリズムに関してです。この手法ではデータ

をM通り、ブートストラップでサンプリングします。各サンプリングデータに対して、それぞれの決定木でフィッティングする。そして、予測を行うと。M通り予測を行うので、木がM個できるわけですね。2つ目のポイントですけれども、木を分割するときに、全特徴量を使うのではなく、一定数の特徴量をランダムに選択するという点がポイントとなります。なぜこのようなことをやるのか、理論は今日は追いかけないのですけれども、これらを行うことで予測値の分散が低減されまして、モデルの期待損失を改善させることができます。下式にありますように、予測値は、全決定木の平均値となります。

B5. 機械学習・深層学習モデル

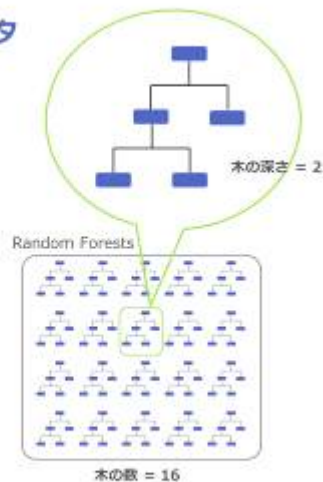
Random Forests における ハイパーパラメータ

1. 木の数
2. 木の最大深さ
3. 木の分割時における変数の選択数
4. ブートストラップ時におけるサンプルデータの数

これらを validation data におけるモデル精度をもとに決定する

KCM は、RMSE $(= \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2})$ を、

OPM は、AUC (受信者検度曲線の下側面積) を、モデル精度を判断する評価指標として採用した。



51

データの分割のしかたでお話しいたしました、機械学習特有のハイパーパラメータですけれども、Random Forest の場合、重要なものとして4つ挙げられます。1つ目が木の数です。2つ目は、それぞれの木が許される最大の深さ。例えば、右図ですと「深さ=2」と書いています。3つ目が、先ほどお話しいたしました、木の分割時における変数の選択数です。最後は、ブートストラップ時におけるサンプリングデータの数。全体のどれくらいをサンプリングするのかというところになります。基本的には、一番目の木の数は多ければ多いほどいいのですが、計算時間との兼ね合いもございますので、十分に精度が出て、収束するような木の数を選択することになります。

B5. 機械学習・深層学習モデル

Random Forests の R package

h2o

- 機械学習・深層学習アルゴリズム 各種が使用可能
- Java base で高速に動作



```
install.packages("h2o")
library(h2o)

train_km <- data_I(km)
filter(train_km, status_target == 1) & (dev_pct != 1) %>%
  sample() %>%
  select(
    #
    "calendar_year",
    "incident_year",
    "initial_claim_reserve",
    "year_of_birth",
    "dev_report",
    "insured_profession",
    "case_specialty",
    "report_lag",
    "sex",
    "m1_lower_age"
  )

gm_km <-
  glm(
    km$y ~ sample_n(
      expand_grid(nTrees = c(50, 100, 150, 200),
                 sample_rate = c(0.3, 0.4, 0.5, 0.6, 0.7, 0.8),
                 n2oCore = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40),
                 max_depth = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40)
                ),
    size = 40070
  )

system.time(
  model_rf_km <- h2o.randomForest(x = 2:mp,
                                 y = km$y,
                                 training.frame = train_km_sub,
                                 sample.rate = gm_kom$sample_rate[1],
                                 categorical.encoding = "AUTO",
                                 ntree = gm_kom$ntree[1],
                                 n2oCore = gm_kom$n2oCore[1],
                                 max_depth = gm_kom$max_depth[1],
                                 seed = 13350)
)
```



使用いたしましたRのパッケージですけれども、h2oのRandom Forestパッケージを使用いたしました。h2oパッケージは、Random Forestだけに限らない非常に包括的なパッケージでして、他にもいろいろなモデルが使用可能です。また、バックエンドがJava baseですので、かなり高速にR上でも動作します。右図がコードの一部を示したものになりますが、上から、パッケージの読み込み、使用している特徴量の一覧となります。そして、その下が探索したハイパーパラメータのレンジです。グリッドサーチをするためのグリッドを表しています。最後がモデル構築部となります。これが一連の流れとなります。結果に関しましては、このスライドの部分では説明いたしません。最後に他のモデルとあわせてご説明させていただきます。

B5. 機械学習・深層学習モデル

C) Gradient Boosting Machine (Friedman 2001) (勾配 boosting)

Boosting に基づくアルゴリズム

X から y を予測 (分類・回帰) する問題に対して、決定木を M 個分、逐次的に学習させる。

例えば、 i ($= 2, 3, \dots, M$) 番目の木を学習させる際、 $i-1$ 番目の木までの予測結果の誤差を補正するように、 i 番目の木の構造を決定する。

「勾配 boosting」の場合は、誤差関数の負の勾配方向に沿うように、決定木の構造を fit させる。予測値を $y_M(X)$ とすると、

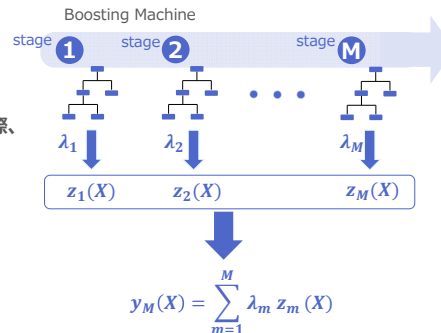
$$y_M(X) = \sum_{m=1}^M \lambda_m z_m(X)$$

X : 選択された特徴量ベクトル

M : モデル (木) の数 (boosting の stage 数)

$z_m(X)$: 各 boosting の stage (1 ~ M) における決定木の予測関数

$\lambda_m(X)$: 各 boosting の stage (1 ~ M) における決定木の重み (学習速度)



53

続きまして、こちらはデータサイエンスのアルゴリズムの中でも大本命なのですが、Gradient Boosting Machine というアルゴリズムに関して、簡単にご説明させていただきます。Random Forest と対称的なアルゴリズムとして挙げられるものです。Boosting という概念に基づいております。Random Forest は、木の数分、並列的に学習を行って、最後に平均を取るということを行ったの対しまして、Boosting アルゴリズムは、逐次的に、直列に学習を行います。

どのようなことかという、右図をごらんいただければイメージがつかうかと思いますが、ステージ1、ステージ2、ステージMとあります。M個分の木を学習させることはさせるのですが、最初の1個目の木の学習が終わったら、次の2個目に行くということで、シーケンシャルに続いているわけですね。では、どのように学習するのかというお話になってくるのですが、例えば、 i 番目の決定木の学習を行う場合、 $i-1$ 番目までの木の全てで構成される予測値、予測結果の誤差を、補正するように学習を行うこととなります。ですから、前の木までの出力が、非常に重要なわけですね。

使っております Gradient Boosting Machine は、「Gradient」とありますので、勾配なのですが、勾配 boosting と呼ばれます。どのようなことかといいますと、勾配 Boosting は、 i 番目の木の出力を決める際に、誤差関数の負の勾配方向に沿うように、 i 番目の決定木の構造を決めることとなります。スライドの左下にあります数式が、予測値を表すものとなります。Random Forest のように平均値にはなっておらず、各木の出力がある決められた重みで、加重平均のような形で計算されていることが分かります。こちらの重みの決め方までは、本日は説明いたしません。

B5. 機械学習・深層学習モデル

Gradient Boosting Machine における ハイパーパラメータ

1. 木の数 (boosting の stage 数)

2. 木の最大深さ

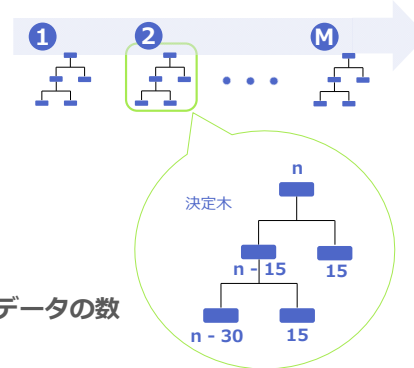
3. 終端ノードにおけるデータ点数の数

4. 各 boosting stage におけるサンプルデータの数

5. 学習速度 (λ)

KCM は、RMSE ($= \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$) を、

OPM は、AUC (受信者感度曲線の下側面積) を、モデル精度を判断する評価指標として採用した。



各終端ノードにおけるデータ点数の数は、15, 15, n - 30



54

勾配 boosting のハイパーパラメータですけれども、重要なものを、スライドに5つほど挙げさせていただきました。どれも重要なのですが、特に木の深さは重要です。他には、木の終端ノードにおけるデータの点数など、このあたりも重要になります。

B5. 機械学習・深層学習モデル

Gradient Boosting Machine の R package

gbm

• 開発終了済みであり、安定版

gbm

Overview:

The gbm package makes it easy for generalized boosted models. It implements regression, classification, and distribution-free survival regression. It includes regression methods for both square and non-square distributions. It also includes methods for handling missing data.

Installation:

```
install.packages("gbm")
# or
library(devtools)
install_github("gbm-developers/gbm")
```

<https://github.com/gbm-developers/gbm>

```
require(tidyverse)
require(gbm)
require(data.table)
require(MASS)

data_validation_kov <- data_validation_kov %>%
  select(
    c(
      "hospital_cost_reporter",
      "year_of_birth",
      "sex_report",
      "trauma_profession",
      "trauma_specialty",
      "report_lag",
      "event"
    )
  )

# Fit a model
gb <- gbm.grid(
  n.trees = 4000,
  interaction.depth = c(2, 3, 5, 10),
  n.minobsinnode = c(5, 8, 1),
  shrinkage = c(0.05, 0.1),
  bag.fraction = c(0.5, 0.8)
)

set.seed(12345)
fit_kov_gbm <- gbm(data_model_kov[, colnames(fit_kov_gbm) %in%
  colnames(fit_kov_gbm)],
  data_validation_kov,
  distribution = "gaussian",
  n.trees = gbm.trees(fit_kov_gbm),
  interaction.depth = gbm.interaction.depth(fit_kov_gbm),
  n.minobsinnode = gbm.minobsinnode(fit_kov_gbm),
  shrinkage = gbm.shrinkage(fit_kov_gbm),
  bag.fraction = gbm.bag.fraction(fit_kov_gbm),
  verbose = 1,
  verboseIT = 1)
```



55

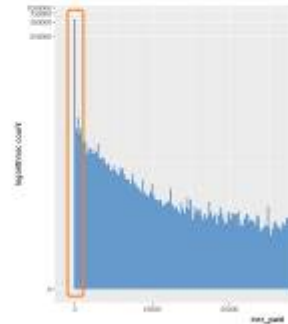
そして、勾配 boosting の R のパッケージですけれども、gbm というパッケージを、本ワーキングパーティーでは使用いたしました。このパッケージは、かなり前からあるもので、かつ、既に開発は終了しています。ですから、今は使う人はあまりいないのかもしれませんが、本ワーキングパーティーでは採用しております。右図が、R のコードの一例です。先ほどと同じような一連の流れを図示したものとなっています。

ます。

B5. 機械学習・深層学習モデル

D) Gradient Boosting Machine Combined

- 前述の Gradient Boosting Machine を「KCM / OPM に分けずに統合した形」でモデリングを行う。
- 回帰問題として取り扱う形になる。
- クレームデータの大部分は、既に close していて 支払・損害発生 が無く、ゼロ点過剰な分布 (Zero Inflated) である。そのため、本モデルは支払備金のモデルとしてはあまり良いものではない。後述のモデル性能の比較にて、本モデルが他モデルと比較して劣る点を述べる。



次に、こちらの Gradient Boosting Machine の Combined というものがあるのですが、こちらは何かと申しますと、これまでのモデルでは KCM と OPM はそれぞれ別々にモデリングいたしますが、別々にモデリングせずに、統合した形でモデリングを行うものになります。ただ、結果から申しますと、あまり良いモデルではありません。なぜかといいますと、右下の図にありますように、支払保険金や発生保険金は、ゼロ点過剰な分布なわけですね。最近の機械学習のモデルといえども、なかなかこのような分布を表現することは難しいということで、後ほど他のモデルと合わせてご説明いたしますが、あまり良い結果には至りませんでした。

B5. 機械学習・深層学習モデル

E) Boosted Tweedie Compound Poisson Model (Yang et al. 2016)

- Tweedie 分布と GBM を組み合わせたモデル

- Tweedie 分布は、

$$\text{Var}(y) = \frac{\phi}{\omega} \cdot E[y]^p$$
$$y \sim \text{Tweedie}(\mu, \phi/\omega, p)$$

と表せる。ここで、 ϕ は分散パラメータ、 ω は エクスポージャを表す。

p は乗法パラメータであり、

$p = 0$	正規分布	
$p = 1$	ポアソン分布	
$p = 2$	ガンマ分布	
$1 < p < 2$	複合ポアソンガンマ分布	となる。

- 本 WP では、
KCM は $p = 2$ すなわち ガンマ分布 を、
OPM は $p = 1.001$ すなわち 「ほぼ」ポアソン分布 を採用した。
- その他の設定は、前述の GBM と同様。



57

ということで、残り 2 個ほどですね、モデルは。そのうちの 1 つが、Boosted Tweedie Compound Poisson Model となります。これは、Tweedie 分布をお聞きになった方も中にはいらっしゃるかとは思いますが、勾配 boosting と Tweedie 分布を組み合わせたモデルとなります。Tweedie 分布は、指数分布族の特別な形であるということをご存じの方も多いと思うのですが、本スライドの中程で示されるような特徴を持っています。そちらのモデルをどのように使ったかといいますと、GLM の枠組みでリンク関数は当然使うと思うのですが、リンク関数の線形表現されていた部分に、この木の回帰モデルの出力を置き換えるという形になっています。これはなかなかいいモデルでして、機械学習と従来のパラメトリックなモデルのいいとこ取りのようなイメージを私は持っています。

本ワーキングパーティーでは、乗法パラメータ、 p ですね。中程に「 p 」と書いてありますけれども、そちらも、正規分布からポアソン分布まで、いろいろな表現は当然できるのですけれども、KCM の場合は $p = 2$ 、すなわちガンマ分布としています。OPM は $p = 1$ が使えないので、1.001 ということで、ほぼポアソン分布であるような分布を採用いたしました。

B5. 機械学習・深層学習モデル

Boosted Tweedie Compound Poisson Model の R package

TDboost

- あまり major な package ではないが、本 WP の本モデル担当者はこちらの package を使用
- 最近の GBM package (xgboost 等) は、tweedie 分布を指定できる。(TDboost package は開発終了)



```

requires(tidyverse)
requires(TDboost)
requires(data.table)
requires(Metrics)

data_validation_kom <- data_validation_kom %>%
  select(
    "initial_case_reserve",
    "year_of_birth",
    "dev_rpt",
    "insured_profession",
    "case_specialty",
    "report_lag",
    "event"
  )

# Grid search
gs <- expand_grid(
  n_trees = c(50),
  interaction_depth = c(12, 13, 15),
  n_minobsinnode = c(15, 4, 8),
  shrinkage = c(0.49, 0.1),
  bag.fraction = c(0.5, 0.6)
)

set.seed(12345)
bst_kom_2006 <- TDboost(data_model_komied_liner_paid ~ .,
  data_predict_kom,
  distribution = list(name="EDM", alpha=0),
  n_trees = gs$n_trees,
  interaction_depth = gs$interaction_depth,
  n_minobsinnode = gs$n_minobsinnode,
  shrinkage = gs$shrinkage,
  bag.fraction = gs$bag.fraction,
  cv.folds = 5,
  verbose=1
  )
  
```

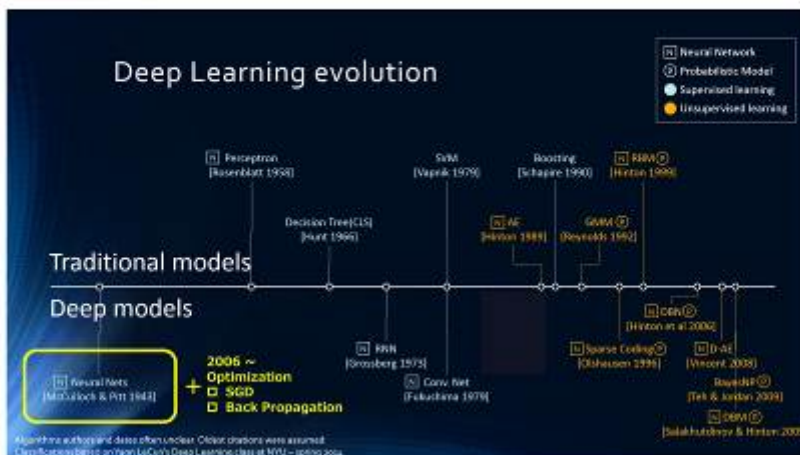


こちらが、使用いたしました R のパッケージとなります。あまりメジャーなパッケージではないのですが、TDboost というパッケージを利用いたしました。近年、データサイエンティストがよく使用するパッケージで xgboost というものがあるのですが、そちらでもこのような Tweedie 分布は指定できますので、あえてこれを使う必要はないかなと思う面もございます。R のコードの一部に関しては、右図のようになります。

B5. 機械学習・深層学習モデル

B) Neural Networks

深層学習分野における最もベーシックなモデルを本 WP で検討



最後のモデルといたしまして、ニューラルネットワークのモデルに関して、簡単にご説明をさせていただきます。こちらが、ニューラルネットワークの技術を時系列で表したものになるのですが、少し

見にくいですね。すみません。本日ご紹介しますニューラルネットワークは、最もベーシックなモデルとなります。最近ですと、画像処理、自然言語処理の分野で、畳み込みニューラルネットワークや再帰型ニューラルネットワークなどの研究開発が盛んなのですが、今から遡ること 1943 年頃に、アニメーションで示している部分ですが、既にニューラルネットワークの基礎理論が確立されています。ただ、計算機の制約や、ネットワークの学習、具体的には中で最適化される重みを求めるということなのですけれども、その辺がうまくできないということで、問題を抱えておりました。ただ、2006 年頃から、確率的勾配降下法や、Back Propagation といった技術の開発とともに実用化のめどが立ったということで、昨今、非常に話題となっている分野になります。

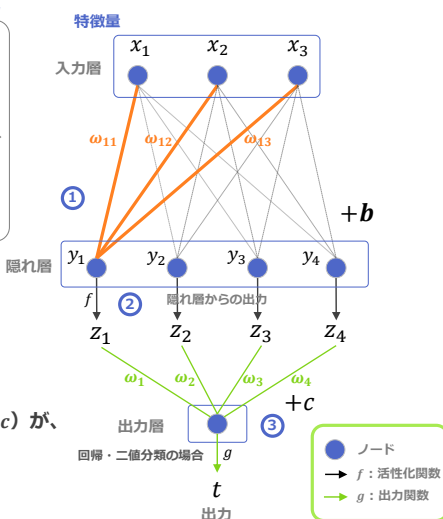
B5. 機械学習・深層学習モデル

代表的な Neural Networks の構造

$$\begin{aligned}
 \textcircled{1} \quad & \begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \vdots & \vdots & \vdots \\ \omega_{41} & \dots & \omega_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix} \\
 \textcircled{2} \quad & \begin{bmatrix} z_1 \\ \vdots \\ z_4 \end{bmatrix} = f \left(\begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} \right) \\
 \textcircled{3} \quad & t = g \left(\begin{bmatrix} \omega_1 & \dots & \omega_4 \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_4 \end{bmatrix} + c \right)
 \end{aligned}$$

演算の流れ ↓

- 最もシンプルな構造は、**入力層 / 隠れ層 / 出力層 の 3 層**からなるもの
- 原理的には、**十分な数のノードがあれば、どのような関数でも表せる (万能近似器)**
- 各層のノード間をつなぐ**重みやバイアス (ω, b, c) が、fitting で最適化される値**
- **重みの最適化手法に関しては、未だに新たな提案がされ続けている**



60

こちらが、代表的なニューラルネットワークの構造となります。基本的には、3層のニューラルネットワーク構造が代表的なものとなります。入力層と隠れ層、出力層という3つの層から成り立っておりまして、入力層は、説明変数、特徴量に対応する層となります。入力された特徴量が出力層までどのように伝播していくかを表したものが、左上の数式となっております。興味のある方は、追いかけていただければと思います。

ポイントといたしましては、3点ございます。隠れ層からの出力は、活性化関数と呼ばれる非線形な関数で変換を受けるという点が1つ目です。2つ目が、訓練データで、各層のノード間をつなぐ重み、 w と表記していますが、あとは、バイアス b ですね、それらが最適化されて求められるという点になります。そして、3つ目ですけれども、出力層からの出力は、出力関数によって、解くべき問題に適した出力構造に変換されるというものになります。例えば回帰問題でしたら、それに対応したスカラー値を表す単一ノードに、3つ以上のクラス分類でしたら、出力ノードは出力ベクトルに対応したクラス数分のものとなります。GLMは、指数分布族などの限られた分布を表現いたしますが、一方、このような構造を持つニューラルネットワークは、十分な数のノードがあれば、どのような関数でも表現が可能です。そのため、ニューラルネットワークは「万能近似器」とも呼ばれます。

B5. 機械学習・深層学習モデル

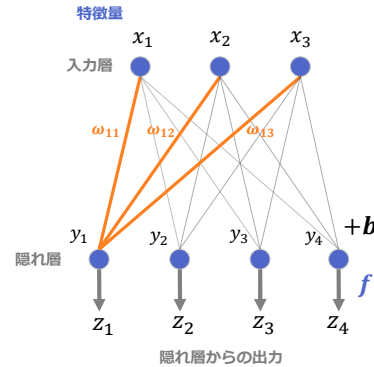
活性化関数

$$\textcircled{1} \begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \vdots & \ddots & \vdots \\ \omega_{41} & \cdots & \omega_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix}$$

隠れ層への入力 重み行列 特徴量 バイアス

$$\textcircled{2} \begin{bmatrix} z_1 \\ \vdots \\ z_4 \end{bmatrix} = f \left(\begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} \right)$$

隠れ層からの出力 f : 活性化関数



- 活性化関数は、隠れ層からの出力を演算する場合は、非線形な関数を使用する。非線形な関数は、本 WP で使用したもの以外にも多数のものが提案されている。
- 出力関数の場合は、回帰モデルであれば、線形である恒等関数を使用し、分類モデルであれば、sigmoid や softmax などを使用する。



61

次に、入力層と隠れ層の演算をもう少しだけ詳しく見てみたいと思います。特徴量が x_1 から x_3 とありますが、入力層の各ノードにそれぞれが対応いたします。その場合、左上の数式①のように、入力層と隠れ層のノード間をつなぐ重みによって、隠れ層の各ノードを表す値へと変換を受けます。この重みとバイアスが、最適化されるべき値です。そして、数式②のように、非線形な活性化関数によって、隠れ層の出力となるような変換を受けます。仮に活性化関数が非線形ではなく線形であった場合、計算してみると分かるのですが、ネットワーク全体として、ただの重回帰モデルとなってしまいます。ですから、活性化関数は、非線形であることが重要です。

B5. 機械学習・深層学習モデル

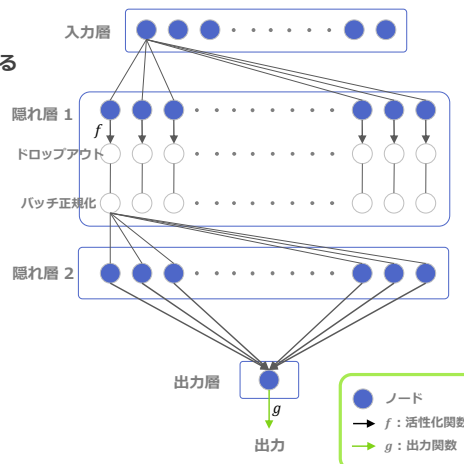
本 WP で使用した Neural Networks の構造

- 本 WP では、入力層、隠れ層 1・2、出力層の 4層からなるネットワークを、KCM 及び OPM に適用
- 隠れ層1 のノード数は、500
隠れ層2 のノード数は、250

- KCM
隠れ層の活性化関数は ReLU
出力関数は linear
- OPM
隠れ層の活性化関数は tanh
出力関数は sigmoid

- ドロップアウト層のドロップ率は 10%

- バッチ正規化の位置が通常と異なる

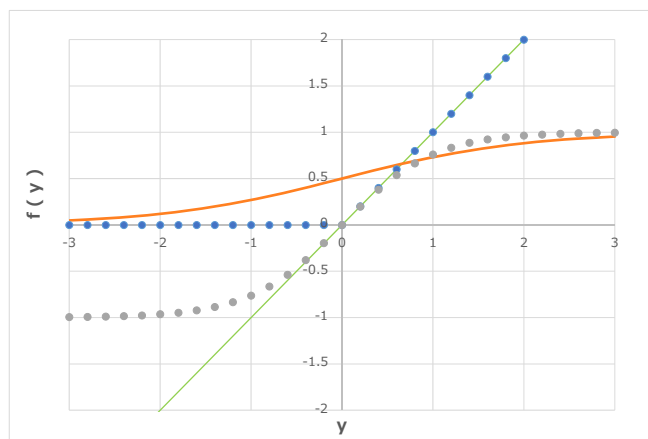


62

それでは、実際に使ったネットワークの構造を紹介いたします。先ほどの例とは異なりまして、隠れ層は1つ増えております。全部で4つの層からなるネットワークとなります。次のページ以降で、こちらに関しまして、説明をさせていただきたいと思っております。主に活性化関数、ドロップアウト、バッチ正規化、以上の3つの技術に関して説明させていただきます。

B5. 機械学習・深層学習モデル

本 WP で使用した 活性化 / 出力 関数



活性化関数

- **ReLU (KCM)**
 $f(y) = \max(y, 0)$
- **tanh (OPM)**
 $f(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$

出力関数

- **linear (KCM)**
 $f(y) = y$
- **sigmoid (OPM)**
 $f(y) = \frac{1}{1 + e^{-y}}$



63

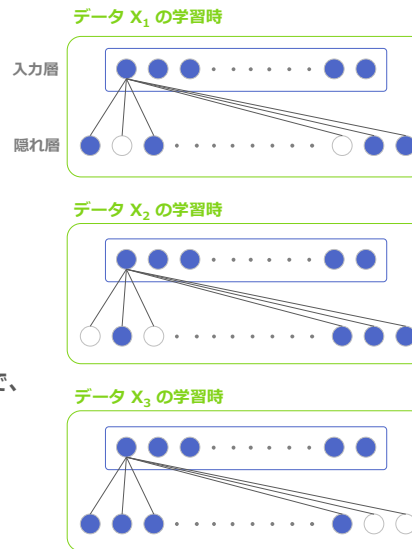
最初に活性化関数なのですが、KCMの場合は、隠れ層の活性化関数はReLU関数、これは「レル」と読みます。そして、出力関数はlinear、ただの恒等関数ですね。そちらを使用いたしました。OPMの場合は、隠れ層の活性化関数はtanh（タンジェントハイパボリック）、出力関数はsigmoid。これは、二値分類だからですね。そちらを使用いたしました。活性化関数には他にもいろいろな種類がありますが、ハイパーパラメータと同様に、validationにおけるモデル精度を判断して選択することになります。

B5. 機械学習・深層学習モデル

ドロップアウト

- 隠れ層のノードのうち、いくつかを使用せずに学習を行う。
- 一般的には、ドロップ率は 0.5 付近を使用するが、validation を通して最適化する。
- ドロップアウトは、データ毎に、いくつかの特徴量を使用せず、残りを使用するというイメージに近い。
- Random Forest と同様に、使用する特徴量をランダムで選択することで、高い汎化性能をもたせているとみなすことができる。

**ドロップアウトは、
擬似的に、
アンサンブル学習を行っている。**



ASTIN
ACCREDITED



ICASA
BERLIN 2018

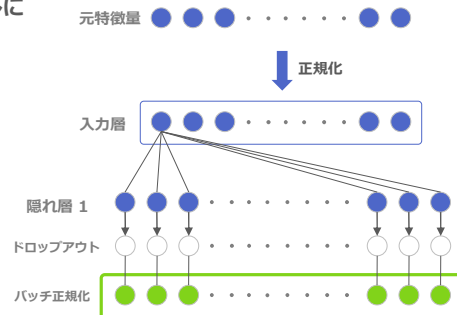
64

次に、ドロップアウトに関して説明いたします。こちらは、比較的最近出てきました技術ではあるのですが、学習の際に、隠れ層のノードをある一定の割合で使用しないで、モデルの汎化性能を上げる手法になります。汎化性能と言っているのは、train data のモデルに対して過剰に適合しないようにする。つまり、未知のデータが入ってきたときに、未知のデータに対しても一定の精度をきちんと出せるようにするということになります。使用しない割合は、これも結局 validation ロジックを通して適した値を見つけることとなりますが、大体 50%前後が目安となります。Random Forest の説明を思い出していただけるといいかと思うのですが、ドロップアウトは結局何をやっているかという、一定の割合でノードを使わないようにしているので、ノードに対応したメタな特徴量が伝播していくのを一定の割合で塞いでいるわけですね。ですから、Random Forest でありましたように、変数を一部選択するような機能があると思います。ということで、汎化性能の向上に寄与するというイメージがつくと思います。

B5. 機械学習・深層学習モデル

バッチ正規化

- 深層学習は重みを最適化で求めることになるが、最適化のアルゴリズムの関係上、重みの初期値や入力する特徴量のスケールに最適化の良し悪しが大きく影響を受ける。
- 特徴量のスケールを均一にするため、入力する特徴量を必ず正規化するが、伝播するにつれて、各ノードにおける値のスケールは大きく異なってしまう。
- そこで、主に隠れ層間に「バッチ正規化」と呼ばれる処理を挿入し、スケールを揃える。
- 通常は、活性化関数の直前にバッチ正規化をいれるが、本 WP では、ドロップアウトの直後に使用している。



65

最後に、バッチ正規化に関して説明をさせていただきます。学習を通して最適化されるのは、ノード間の重みとバイアスでした。使用している最適化のアルゴリズムに依存はいたしますけれども、重みの初期値や特徴量のスケールに最適化の良し悪しが大きく依存されてしまいます。そのため、入力する特徴量は、基本的には正規化をしたりして均一な値にスケールリングするのですが、入力層から出力層に向かって隠れ層をどんどん進むに従いまして、重み係数をいろいろと掛けたあとの各ノードの値は、ノード間でばらついてしまいます。非常に大きな値もあれば、小さな値もありますので、ニューラルネットワークでは、バッチ正規化と呼ばれる処理を、一般的には活性化関数の直前に入れることになります。本ワーキングパーティーではドロップアウトの直後に入れておりまして、この点は、実は少し「どうなのかな」ということはあるのですが、いずれにいたしましても、バッチ正規化を入れていることは確かです。

B5. 機械学習・深層学習モデル

バッチ正規化

- 実際の学習時には、データ 1 つ毎に重みを最適化するのではなく、一定数のデータに対して重みを最適化する。これを「バッチ（ミニバッチ）学習」と呼ぶ。

まず、 m 個のデータからなるミニバッチ $\beta = \{x_1, x_2, x_3, \dots, x_m\}$ があつた場合、ミニバッチの平均 μ_β および分散 σ_β^2 は、

$$\mu_\beta = \frac{1}{m} \sum_{i=1}^m x_i$$
$$\sigma_\beta^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_\beta)^2$$

と記述できる。

これに対して、バッチ正規化は、ミニバッチ内の各データ x_i を下記のように変換する。

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$$
$$y_i = \gamma \hat{x}_i + \beta$$

ここで、 γ と β は、バッチ正規化にともなうモデルパラメータであり、最適化の対象である。



66

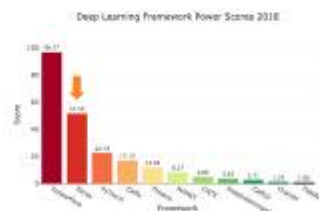
次のページで、バッチ正規化のロジックを、もう少しだけ詳しく見てみたいと思います。そもそも「バッチ」という言葉は何なのかということですが、ニューラルネットワークの学習時に、データを1つずつ使用してネットワークの学習をするのではなく、一定数のデータをまとめて学習させる「バッチ学習」という言葉から来ています。基本的には、ニューラルネットワークのモデルを学習させる時は、バッチ学習をすることになります。本スライドの数式の説明では、データが m 個あつたとして、バッチ正規化の説明を書かせていただいております。これは非常にシンプルな、標準化のようなものですね。ここで行っていることは、 m 個のサンプルに対して、その平均・分散を計算して標準化するということになります。

B5. 機械学習・深層学習モデル

Neural Networks の R package `Keras` (TensorFlow backend)



- TensorFlow は、google が開発した深層学習ライブラリ
- TensorFlow のままだとコードが書きにくいので、それを助ける wrapper package が、Keras



```
require(tidyverse)
require(Matrix)
require(keras)
require(tensorflow)

train_kcm <- data_2_0_5
filter((claim_status_target == 1) & (dev_rpt == (1-min_dev))) %>%
  as_group() %>%
  select(
    c(
      "dev",
      "claim_number",
      "claim_year",
      "accident_year",
      "initial_claim_reserve",
      "age_group",
      "dev_rpt",
      "insured_profession",
      "case_specialty",
      "incident_creating",
      "report_lag",
      "report",
      "adj_incr_poid"
    )
  )

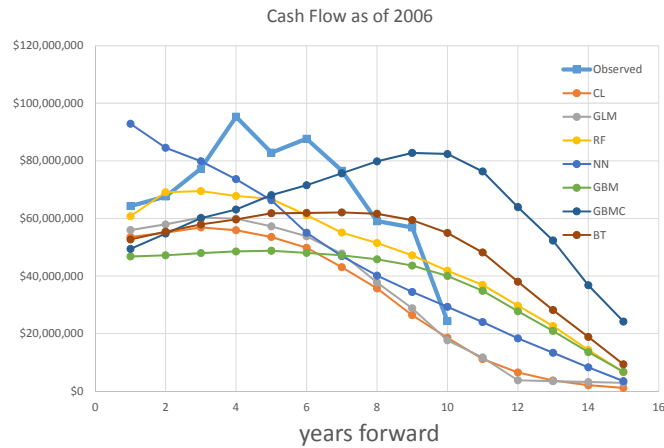
# model orig
model_kcm_orig <- keras_model_sequential()
model_kcm_orig %>%
  # layer 1
  layer_dense(units = 500,
              input_shape = c(length(x_train_kcm),),
              activation = 'relu') %>%
  layer_batch_normalization() %>%
  # layer 2
  layer_dense(units = 300,
              activation = 'relu') %>%
  layer_batch_normalization() %>%
  # output
  layer_dense(units = 1, activation = 'linear')

summary(model_kcm_orig)
```



伝統的手法と機械学習・深層学習の計算結果比較

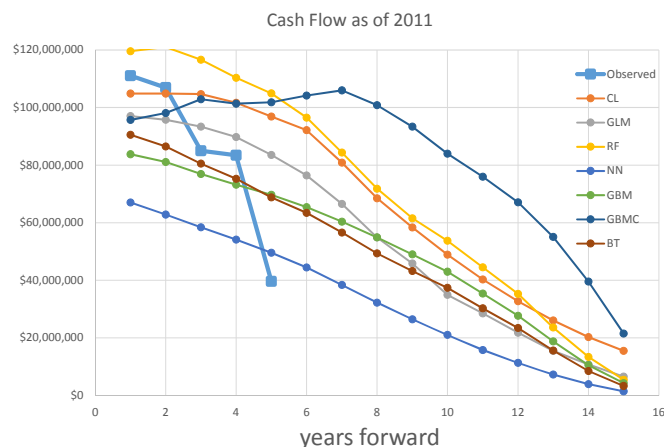
年間支払保険金



69

こちらが、2006 年を評価年度といたしました、年間支払保険金の実績値と、それに対する各モデルの予測値をプロットしたものとなります。2006 年度が評価年度ですので、1 年後は 2007 年度の支払保険金の値を、2 年後は 2008 年度の支払保険金の値を指すこととなります。Gradient Boosting Machine Combined の予測値が、他のモデルと異なっていることが読み取れると思います。また、一方で RMSE の観点では、Random Forest と Boosted Tweedie が、比較的良好な結果を示しています。

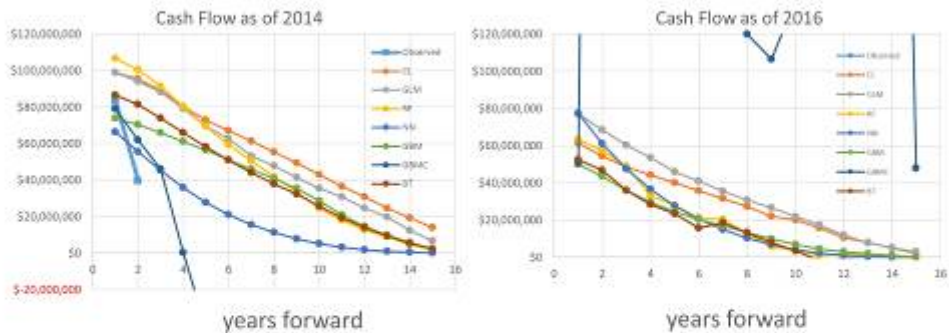
伝統的手法と機械学習・深層学習の計算結果比較



70

次に、こちらが同様に 2011 年度のものとなります。2006 年度と同様に、Gradient Boosting Machine Combined が、少し変なカーブを描いているということになります。

伝統的手法と機械学習・深層学習の計算結果比較



GBMC の年間支払額が大きく変動しており、ゼロ点過剰な分布である支払保険金に対して、良いモデルでないことが伺える。



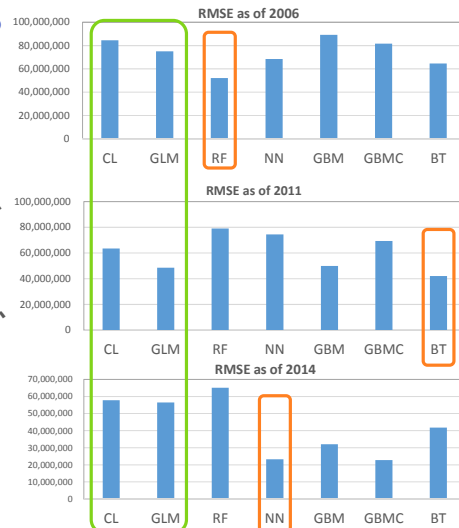
こちらは、評価年度が 2014 年度と 2016 年度のものになりますが、2016 年度のは、予測値に該当する観察データがございませんので、観察データがプロット上に存在いたしません。これらの 2 つの評価年度ですと、Gradient Boosting Machine Combined model の年間支払保険金が、プロットに収まらないくらい大きく変動してしまっているということで、ゼロ点過剰な分布に対しては、良いモデルではないということが示唆されます。

伝統的手法と機械学習・深層学習の計算結果比較

評価年度 2006, 2011, 2014年における各モデルの RMSE を比較

$$RMSE_m = \sqrt{\frac{\sum_{j=m+1}^{2016} (I_j^m - I_j)^2}{n}} \quad m \in \{2006, 2011, 2014\}$$

- 総合的にみると、年間支払保険金に関しては、Boosted Tweedie が優れている。
- GBMC の 2014 年における RMSE が低いが、2016 年以降の予測値が、前述の通り不安定である。



こちらが、観察データがございます評価年度の 2006 年、2011 年、2014 年における、各モデルの RMSE を比較したものとなります。右図において緑色で囲みましたものが伝統的な従来の手法で、オレンジ色で囲

みましたものが、各評価年度で最も優れていたモデルとなります。全年度の RMSE を取る形ですと、Boosted Tweedie が、かなり前のページであれなのですけれども、最も優れているという結果になります。

伝統的手法と機械学習・深層学習の計算結果比較

各評価指標における「Best model」と「Worst model」

		トライアングルにおける 各セルの誤差			年間（支払 / 発生）保険金			将来（支払 / 発生）保険金		
		2006	2011	2014	2006	2011	2014	2006	2011	2014
Paid	Best model	NN	GBM	NN	RF	TD Boost	NN	NN	NN	NN
	Worst model	GBM Combined	chain ladder	GBM Combined	GBM	RF	RF	GBM Combined	RF	RF
Incurred	Best model	GLM	GBM	NN	GLM	chain ladder	GLM	GBM Combined	GBM	GBM
	Worst model	NN	GBM Combined	TDBoost	NN	TDBoost	TDBoost	NN	GLM	TDBoost

ASTIN WP 2018 REPORT

- ・ 評価指標や目的変数（Paid / Incurred）によって、Best / Worst model は異なる。
- ・ 支払保険金では NN が、発生保険金では GBM が良い傾向にある。
- ・ 一方で、CL は安定した見積もりをだす傾向にある。



73

それでは、最後に、全評価指標を支払保険金・発生保険金に関しましてまとめたものを、ご覧いただきます。こちらが、ASTIN WP 2018 のレポートからそのまま表を取ってきたものになるのですが、各評価指標や、発生保険金もしくは支払保険金に関してかという観点で、ベスト、ワーストのモデルが大きく異なります。アニメーションで明滅している部分が、先ほどお見せしたのになります。全体的な傾向としましては、支払保険金ではニューラルネットワークが、発生保険金では勾配 boosting が良い傾向にあります。ただ、一方で、伝統的手法でありますチェインラダーなどは、どの評価指標、評価年度でも、安定した傾向にあるということが言えると思います。

伝統的手法と機械学習・深層学習の計算結果比較

	伝統的手法	機械学習・深層学習的手法
長所	<ul style="list-style-type: none"> 手法としてシンプルで、全体的な傾向の把握に有用 計算コストが低い volatile なデータに対しても、ロバストな結果を示す 	<ul style="list-style-type: none"> クレーム毎のデータ粒度に対してのモデル構築が容易 非線形な特性を考慮したモデル構築が容易 stable なデータに対しては、伝統的手法を上回る精度をだしやすい
短所	<ul style="list-style-type: none"> 非線形な特性を考慮したモデル構築が行にくい トライアングルデータとして、集積されたデータに対してのモデル構築をするため、クレーム毎の特性を考慮できない 	<ul style="list-style-type: none"> 計算コストが高く、ハイパーパラメータの探索も必要 volatile なデータの場合、過学習する恐れがある Neural Networks などのモデルの場合、データ量が豊富である必要がある



74

以上を踏まえますと、このような比較結果となります。伝統的手法は、シンプルで計算コストが低い点が、まず挙げられます。また、今回の発生保険金のようなボラティリティーが高いデータでも、安定して推定を行える傾向がある点が挙げられます。一方で機械学習・深層学習的手法は、クレームごとのデータの粒度に対するモデル構築が容易な点。また、非線形性を考慮したモデリングもできる点。あとは、今回の支払保険金のような、比較的データが安定している場合ですね。そちらの場合は、従来手法を上回るような精度を示すモデルが多いという点が挙げられます。1点注意なのですが、ニューラルネットワークのようなモデルの場合は、データ点数が十分ないと、なかなかうまく学習できなかったりという欠点がありますので、そこは注意をする必要があると思います。

7. 今後の展望

1. 予測すべき対象と評価指標によって、最良のモデルは異なっていた。

No Free Lunch Theorem :
there is no model that is always better than all other models,
one should try them all and then understand
which one perform better on the underlying dataset.

機械学習・深層学習のモデルの予測値を組み合わせることで、
よりロバストで高精度な予測値を得る。

2. 伝統的なモデルと機械学習・深層学習的手法によるモデルを組み合わせる。



76

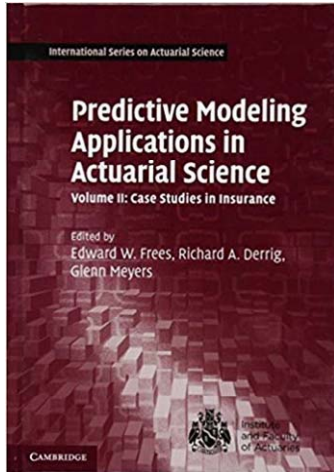
こちらのワーキングパーティーの今後の展開、展望ですけれども、予測すべき対象と評価指標によって最良のモデルは異なっておりましたので、各モデルの予測値を組み合わせる、つまりアンサンブルですね。それを行うことで、よりロバストで高精度な予測値を得る取り組みが考えられます。緑色で示させていただいておりますものは、「No Free Lunch Theorem」というものでして、簡単に申し上げますと、全ての問題に対して常にベストの予測値を返すようなモデルは存在しないということになります。また、二番目ですけれども、機械学習・深層学習モデルどうしのアンサンブルに限らず、ロバストであった伝統的な手法と、機械学習・深層学習の予測を合わせることで、いいとこ取りの高精度な予測値が出せるのではないかと考えられます。

ASTIN関連研究会の活動内容のご紹介（2018年度）

データサイエンス関連書籍の翻訳

Predictive Modeling Applications in Actuarial Science

Volume II: Case Studies in Insurance (International Series on Actuarial Science)



『保険数理における予測モデリングの応用 第II巻』

- ✓ CASの“Predictive Modeling for Actuaries Book Project”で執筆された書籍
- ✓ 原書はアクチュアリー向けに書かれた全2巻からなるシリーズの第2巻で、予測モデリングの基礎を扱ったもの
- ✓ 昨年度（2017年度）、本研究会にて“第1巻”の翻訳作業を行い、eラーニングシステム上で公開
- ✓ 第1巻は線形モデルをベースに予測モデリングの要素（パラメータ推定、変数選択、モデル検証など）について説明
- ✓ 第2巻はより実務向けの内容で、一般化線形モデル（GLM）の応用から、クラスタリングやランダムフォレストなどの手法を用いたケーススタディも記載
 - * 第1章はEDA（探索的データ分析）などの演習問題が充実！（HP上でRコードや解説が入手可能）



以上でワーキングパーティーのご説明は終わりにさせていただきますが、最後に、私が委員を務めさせていただいておりますASTIN関連研究会の取り組みに関して、ご報告をさせていただきます。現在、ASTIN関連研究会では、アクチュアリーの数理領域におけるデータサイエンス関連の書籍の翻訳を行っております。『Predictive Modeling Applications in Actuarial Science Volume II』と題しまして、CASのPredictive Modeling for Actuaries Book Projectで執筆された書籍となりまして、原書はアクチュアリー向けに書かれ、全2巻から成るシリーズの第2巻になります。昨年度、本研究会から第1巻の翻訳作業を行い、eラーニングシステム上で公開しております。

第1巻は、線形モデルをベースに、予測モデリングの要素を、パラメータ推定等について説明しておりますが、第2巻は、より実務的な内容となっております。一般化線形モデルの応用から、クラスタリングやランダムフォレスト、決定木ベースの手法などを用いたケーススタディも記載しております。

ASTIN関連研究会の活動内容のご紹介（2018年度）
 データサイエンス関連書籍の翻訳（続き）

章	内容
1	一般化線形モデル (GLM) による純保険料モデリング ←演習問題が充実！
2	一般化線形モデル (GLM) の保険データへの適用：頻度・損害規模モデルと純保険料モデル
3	クレーム予測モデルとしての一般化線形モデル (GLM)
4	損害保険の料率算定フレームワーク：一般化線形モデル (GLM) の先
5	団体健康保険の料率算定におけるマルチレベルモデルの利用：エジプト市場からのケーススタディ
6	損害保険のプライシングにおけるクラスタリング
7	不審なデータに対する2つの教師なし学習手法の適用：PRIDITとランダムフォレスト
8	有限時間ホライゾン上の支払備金の予測分布推定
9	有限混合モデルと労働者補償保険の大口損害の回帰分析
10	予測モデリングを利用したクレーム増大管理のフレームワーク
11	運転状況連動型自動車保険のための予測モデリング



詳しい章ごとの内容は、こちらとなります。演習問題も充実しておりますので、豊富な内容となっております。現在、2019年の3月完成に向け、翻訳チーム一同で鋭意作業中でございます。翻訳が完成した後は、是非お手にとっていただければと思います。以上、ご清聴ありがとうございました。

【司会】 会場からの質問を受け付けたいところではありますが、終了の時刻となりましたので、以上をもちまして、セッションB、「支払備金の見積もりにおける機械学習・深層学習的手法の適用と従来手法との比較（2018 ASTIN WP 参加報告）」を終了します。平松さんに、いま一度大きな拍手をお願いします。