Institute
and Faculty
of Actuaries

Neural network models
(Especially in claims reserving)

Jacky Poon

25 April 2023

About this presentation

- Neural network models have found successful applications in wide-ranging fields from computer vision to natural language and generative text. Can it find a place in claims reserving?
- In this live presentation, we will develop a reserving model step-by-step: starting with a simple chain ladder, we gradually introduce incremental improvements, to culminate with a probabilistic, mixture density, neural network on individual claims.
- This is a technical session, including both introductory and novel concepts, conducted by a member of the Machine Learning in Reserving Working Party (MLRWP). It should be of interest to actuaries at all levels in ML, especially those keen on the practical implementation of ML in the reserving process.
- Attendees will have access to an accompanying Python notebook with full workings for further reading which allows them to fully replicate the models.

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

What is the potential with neural networks?

Advantages:

- Residual networks generalize GLMs
- Used in state-of-the-art models for e.g. image, text, audio transcriptions
- Transformers quite powerful for sequence data generally
- Entity embeddings can effectively model categorical variables
- Output probability distributions with mixture density

Issues to consider:

- In time series, simpler models often perform as well as neural networks ("Are Transformers Effective for Time Series Forecasting?" A. Zeng et al)
- With tabular-only data, gradient boosted decision trees are often easy to calibrate to a good result.
- Random initialization may lead to variance in model predictions
- Complexity vs simpler models

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

About the author

- Member of Machine Learning in Reserving Working Party (with IFoA)
- Head of Finance at nib Travel
  - Experience in pricing & analytics
- Convenor for the Young Data Analytics Working Group (with Actuaries Institute in Australia)
  - Newsletter, podcast, articles, events
  - Check out our “Actuaries’ Analytical Cookbook”! https://actuariesinstitute.github.io/cookbook/docs/index.html

Institute and Faculty of Actuaries

This session’s journey

- Start with a chain ladder model
- Incremental changes
- Working model at each step
- Finish with probabilistic neural network model
- Simple simulated dataset
- Code available

Chain Ladder
↓
Aggregated GLM
↓
Individual GLM
↓
Neural Network
↓
Probabilistic Model

First, some background and observations…

Institute and Faculty of Actuaries

Institute and Faculty of Actuaries

Part 1: Chain Ladder

Loss data

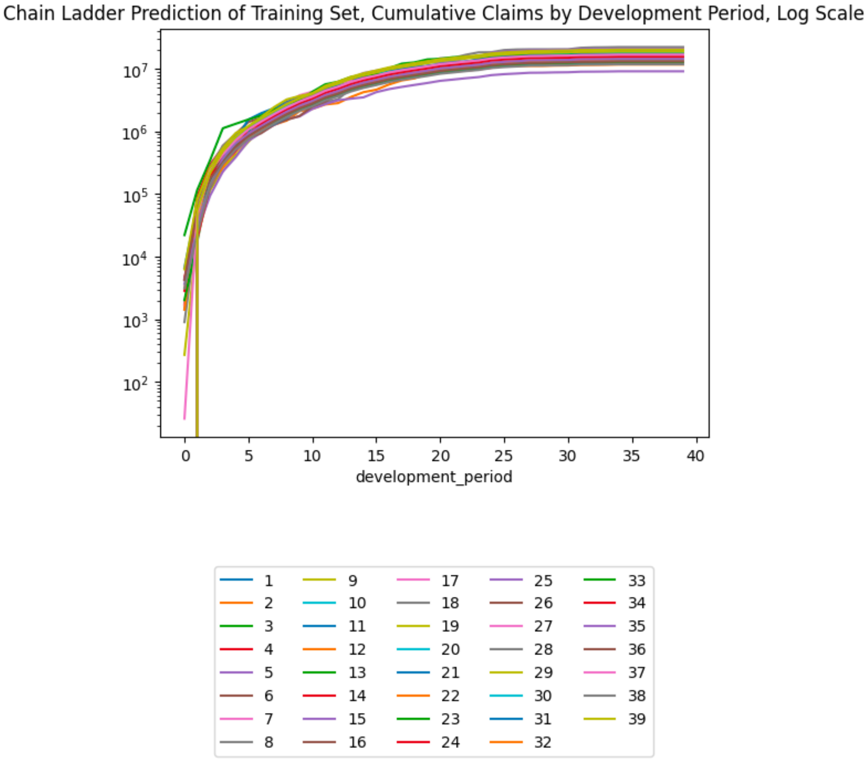
- Few examples of publicly available, detailed, real world data.
- Code for this presentation is fully available, so using simulated data.
- Five datasets from using a simulated package, SPLICE.
- Includes payments and reserves, but not exposures.
- Different behaviour for large vs attritional claims.

- "**Scenario 1**: simple, homogeneous claims experience, with zero inflation.
- **Scenario 2**: slightly more complex than 1, with dependence of notification delay and settlement delay on claim size, and 2% p.a. base inflation.
- **Scenario 3**: steady increase in claim processing speed over occurrence periods (i.e. steady decline in settlement delays).
- **Scenario 4**: inflation shock at time 30 (from 0% to 10% p.a.).
- **Scenario 5**: default distributional models, with complex dependence structures (e.g. dependence of settlement delay on claim occurrence period)."

From https://github.com/agi-lab/SPLICE/tree/main/datasets

Institute and Faculty of Actuaries

Chain ladder



- CAS has a chain ladder package (https://chainladder-python.readthedocs.io/en/latest/intro.html) but chain ladder is mechanically quite simple
- We fit a chain ladder using dataframes (Pandas + Numpy)
- So now, we have a basic reserving model – what's next?

Institute and Faculty of Actuaries

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Institute and Faculty of Actuaries

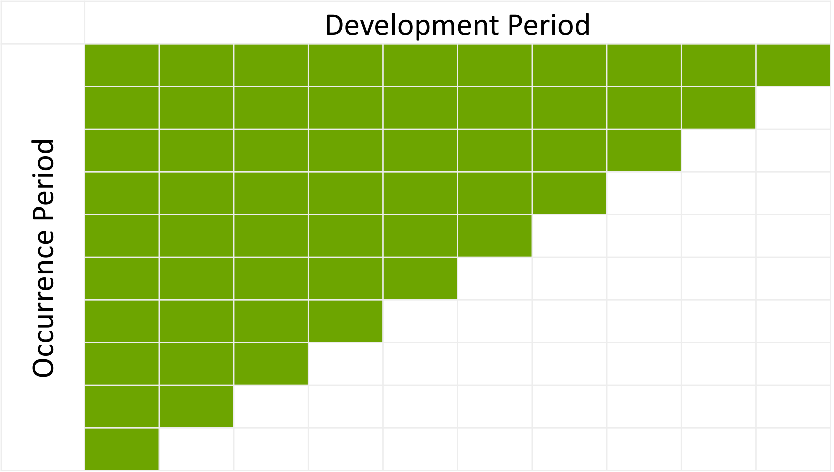
Part 2: Fitting GLMs

Using neural network packages (so that logic can be expanded to neural networks)

Observation 1: Chain ladder is a GLM

There is a GLM form of chain ladder:

- Log link, over-dispersed Poisson
- *Incremental Payments ~ Occurrence Period + Development Period*
- Occurrence Period and Development Period are one-hot encoded (1-0 flags for occurrence and development period = n, n=1...N)
- See https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/foundations/python-glms/

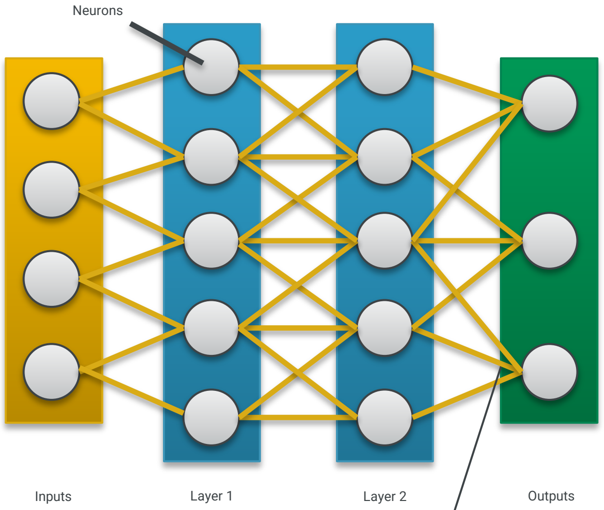


Institute and Faculty of Actuaries

Observation 2: A linear model is a neural network

(With no hidden layers)

- A feedforward neural network is a type of neural network where information flows in one direction, from the input layer through one or more hidden layers to the output layer.
- Each layer in a feedforward neural network consists of a set of nodes, or neurons, that perform a **linear transformation of their inputs** followed by a non-linear activation function.
- The linear transformation performed by each neuron is similar to that of a linear model: the inputs are multiplied by a set of weights, and a bias term is added to the result.
- The activation function applied to the output of each neuron introduces non-linearity into the model, allowing it to learn complex relationships between the input and output variables.
- The outputs are a linear transform of the final hidden layer.
- **Consequently, a feedforward network can be considered a linear model of features, being the final hidden layer.**



The final step is linear!

Institute and Faculty of Actuaries

Observation 2: A linear model is a neural network

With no hidden layers

```python
# One hidden layer (simplified example)
class FeedForwardNet(nn.Module):
    def __init__(self, n_input, n_hidden, n_output):
        super(FeedForwardNet, self).__init__()
        self.hidden = nn.Linear(n_input, n_hidden)
        self.linear = nn.Linear(n_hidden, n_output)

    def forward(self, x):
        x = self.hidden(x)
        x = F.relu(x)
        x = self.linear(x)
        return x
```

```python
# No hidden layers = LM
class LinearModel(nn.Module):
    def __init__(self, n_input, n_output):
        super(LinearModel, self).__init__()

        self.linear = nn.Linear(n_input, n_output)

    def forward(self, x):


        x = self.linear(x)
        return x
```

Gradient descent methods can be used to fit instead of iterated reweighted least squares – minimize normal loss to maximise likelihood

Institute and Faculty of Actuaries

Observation 3: A GLM is also a neural network

```python
# Linear model
class LinearModel(nn.Module):
    def __init__(self, n_input, n_output):
        super(LinearModel, self).__init__()

        self.linear = nn.Linear(n_input, n_output)

    def forward(self, x):


        x = self.linear(x)
        return x
```

```python
# Add an exponential activation at the end
class LogLinkGLM(nn.Module):
    def __init__(self, n_input, n_output):
        super(LogLinkGLM, self).__init__()

        self.linear = nn.Linear(n_input, n_output)

    def forward(self, x):


        x = self.linear(x)
        return torch.exp(x)  # log(Y) = XB -> Y = exp(XB)
```

Add exponential activation to convert from a linear model to a GLM with log link
Poisson loss function can be minimized to fit the GLM

Institute and Faculty of Actuaries

Chain Ladder in a GLM in a Neural Network Package

Returning to our claims data:

- Use Pytorch, a popular, intuitive neural network package,
- Use with scikit-learn, a popular Python machine learning package.
- Suggestions for ensuring models train to convergence to mechanical chain ladder estimates:
  - High number of epochs with gradient descent using Adam
  - Include a bias and set initial value based on the mean
- Output is the same as the mechanical CL.

```
GLM_CL_agg = Pipeline(
    steps=[
        ("keep", ColumnKeeper(["occurrence_period",
        "development_period"])),
        ('one_hot',
        OneHotEncoder(sparse_output=False)), #
        OneHot to get one factor per
        ("model", TabularNetRegressor(LogLinkGLM,
        has_bias=True, max_iter=10000,
        max_lr=0.10))
    ]
)
GLM_CL_agg.fit(
    triangle_train,
    triangle_train.loc[:, ["payment_size"]]
)
```

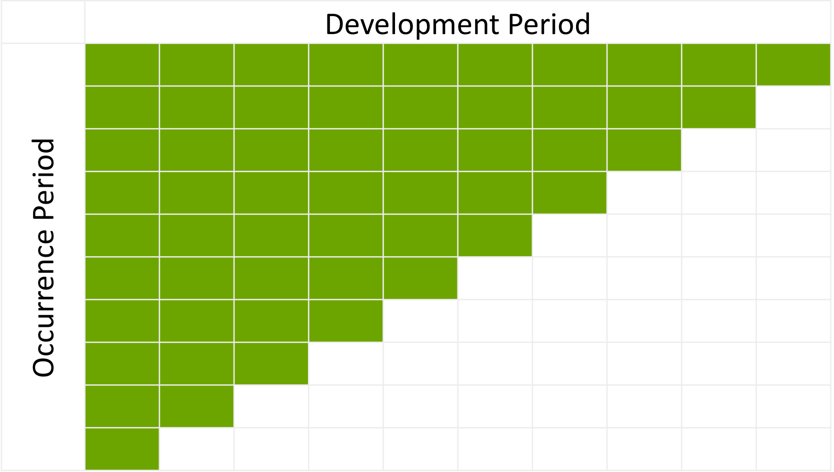
Institute and Faculty of Actuaries

Institute and Faculty of Actuaries

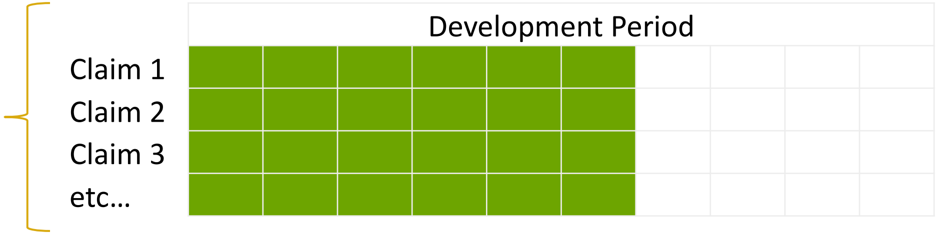
Modelling individual claims data

Expanding to individual data – one approach

Chain ladder GLM: record per occurrence period x development period



"Zoom in"



Instead, use one record per:
claim number x development period

→ Per claim projection (IBNER)

Institute and Faculty of Actuaries

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Tabular data format

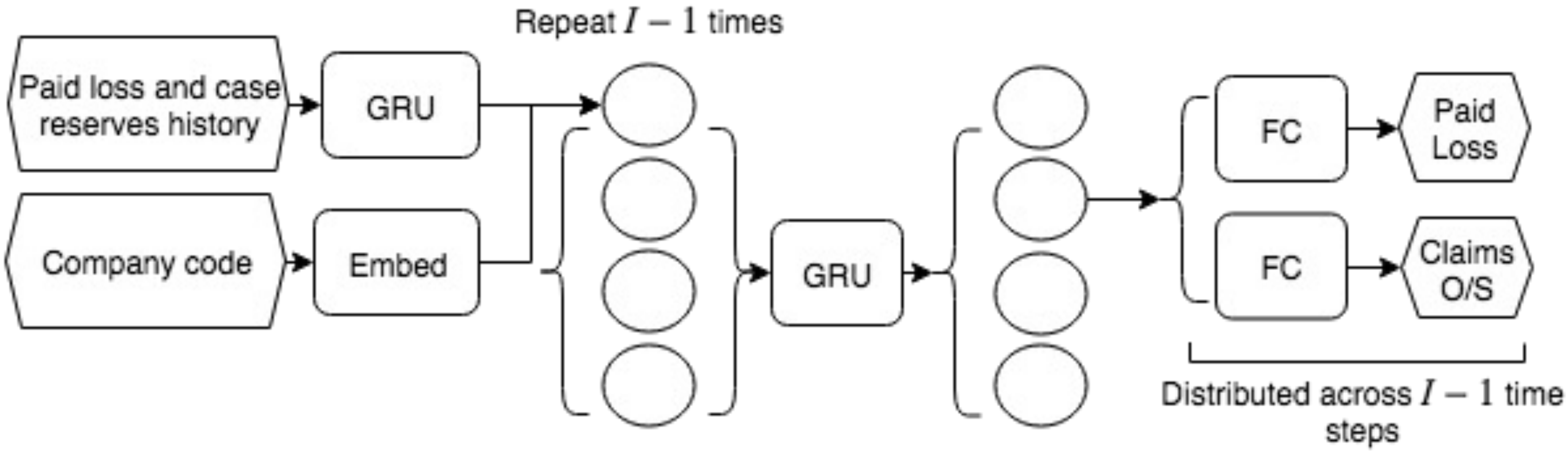
- Tabular approaches – convert the claims data to a tabular format
- Simplest example (right):
  - X: Accident, Development and Calendar Periods
  - y: Payment in the Acc/Dev period.
- More advanced example: "Penalising Unexplainability in Neural Networks for Predicting Payments per Claim Incurred" (Poon 2019)

Accident Period	Dev. Period	Cal. Period	Claims Paid
1	1	1	$xx
1	2	2	$xx
1	3	3	$xx
1	4	4	$xx
2	1	2	$xx
2	2	3	$xx
2	3	4	$xx
2	4	5	$xx
…	…	…	

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Bonus: Not in scope of the current presentation, but worth mentioning

Sequence-based neural networks



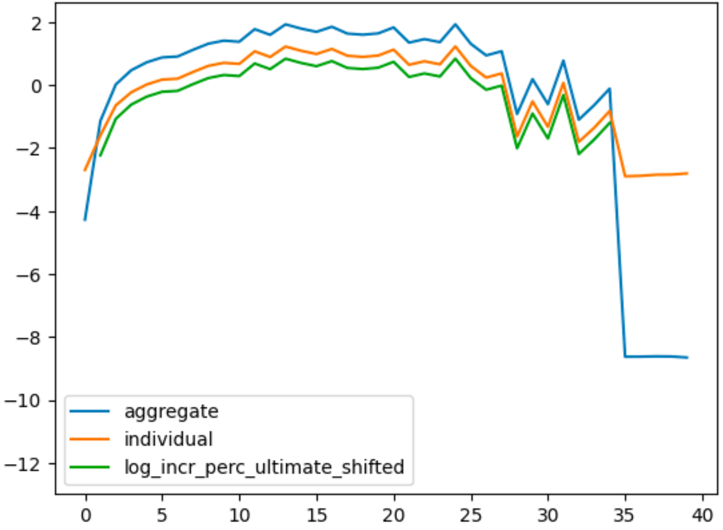
Source: “DeepTriangle: A Deep Learning Approach to Loss Reserving” (Kuo 2018)

- Sequence approaches are also viable
- DeepTriangles uses Gated Recurrent Units

Institute and Faculty of Actuaries

GLMs on individual claims data

Results are, unsurprisingly, similar across the chain ladder, CL-GLM and individual GLM:



Reasons to consider GLMs for claims reserving

GLMs can incorporate:

- Splines
- Features for mix effects
- Features for seasonality
- Transparent results

Institute and Faculty of Actuaries

Institute and Faculty of Actuaries

Individual neural networks

Transitioning from GLMs to Residual Networks

```
# GLM
class LogLinkGLM(nn.Module):
    def __init__(self, n_input, n_output):
        super(LogLinkGLM, self).__init__()


        self.linear = nn.Linear(n_input, n_output)


    def forward(self, x):


        x = self.linear(x)
        return torch.exp(x)  # log(Y) = XB -> Y = exp(XB)
```

```
# “Log Link” ResNet
class LogLinkResNet(nn.Module):
    def __init__(self, n_input, n_hidden, n_output):
        super(LogLinkResNet, self).__init__()
        self.hidden = nn.Linear(n_input, n_hidden)
        self.neural = nn.Linear(n_hidden, n_output)
        self.linear = nn.Linear(n_input, n_output)


    def forward(self, x):
        h = F.elu(self.hidden(x))
        x = self.linear(x) + self.neural(h)
        return torch.exp(x)  # log(Y) = XB -> Y = exp(XB)
```

A residual network is similar to a linear model with additional non-linear deep learning features. By including the exponential transformation at the final step, results become similar to a log-link GLM.

Institute and Faculty of Actuaries

Tips and tricks for neural networks for claims data

- Initialisation strategy:
  - Bias: Set to mean(log(y)) to converge faster
  - Weights: Use zeroes for final layer for stability (see FixUp Initialisation)
- Batch size – data is sparse so
  - As high as possible (we used the full dataset)
- Optimiser – using AdamW
- Architecture:
  - Neural networks are flexible and the structure can be varied to needs.

Institute and Faculty of Actuaries

Custom “SplineNet” Architecture

- We test out our “SplineNet” design:
  - Split inputs into individual features
  - For each feature, fit a hidden layer on just that feature as a one-way “spline”
  - Fit an interaction hidden layer on all inputs as per a residual network but
  - Hide the interaction layer behind a “gate” weight, which is initialized in an “off” state

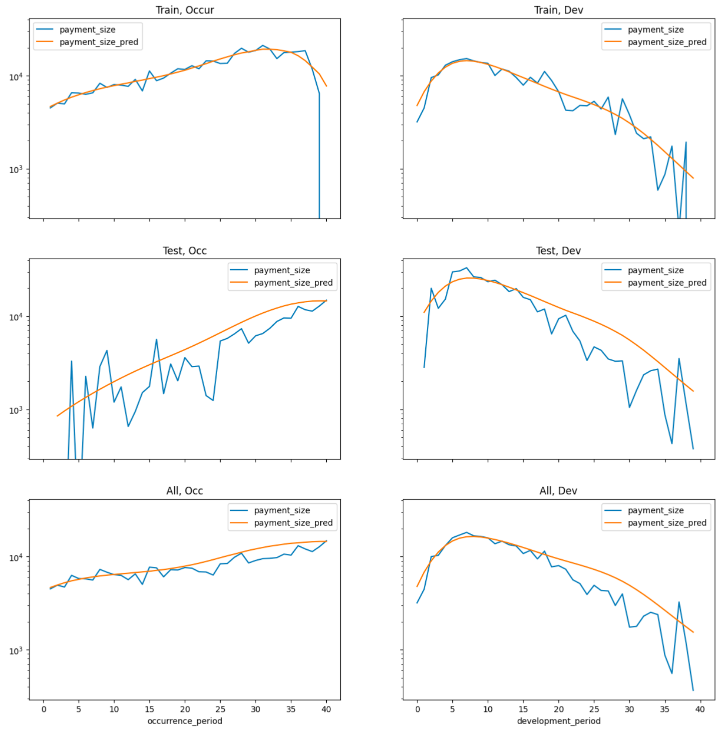
```
# The forward function defines how you get y from X.
def forward(self, x):
    # Apply one-ways
    chunks = torch.split(x, [1 for i in range(0,
    self.n_input)], dim=1)
    splines = torch.cat([self.oneways[i](chunks[i]) for
    i in range(0, self.n_input)], dim=1)

    # Sigmoid gate
    interact_gate = torch.sigmoid(self.interactions)

    splines_out = self.oneway_linear(F.elu(splines)) *
    (1 - interact_gate)
    interact_out =
    self.linear(F.elu(self.hidden(self.dropout(x)))) *
    (interact_gate)
    # Add ResNet style
    return self.inverse_of_link_fn(splines_out +
    interact_out)
```


Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Institute and Faculty of Actuaries

Comparison

GLMs on individual claims data

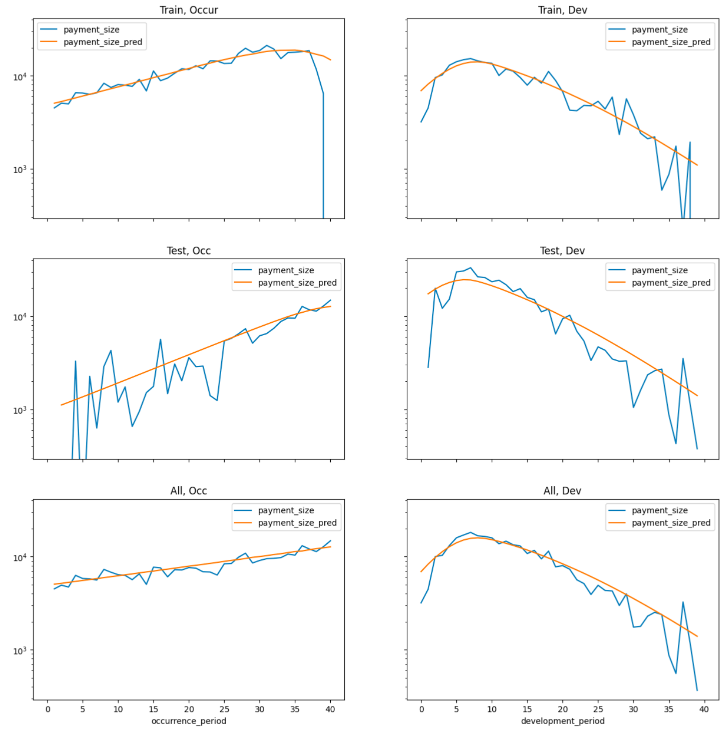


- Example: replacing one-hot encoding of each period with splines
- Results smooth over noisy data
- Picks up on major trends, but not fully capturing the curve

Institute and Faculty of Actuaries

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

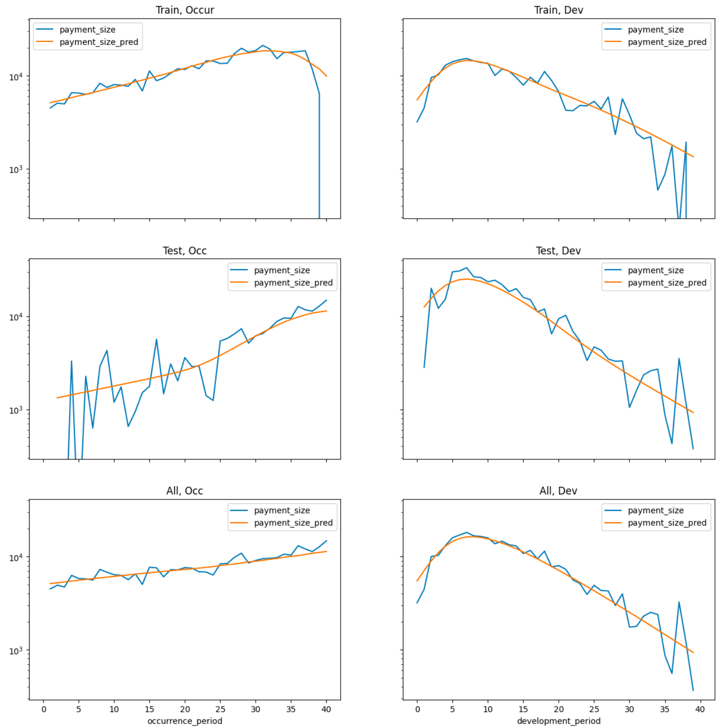
Residual Network



- Residual network
- Using only occurrence and development periods only (but on individual data)
- Picks up major trends
- This run looks ok, but some tendency to overfit.

Institute and Faculty of Actuaries

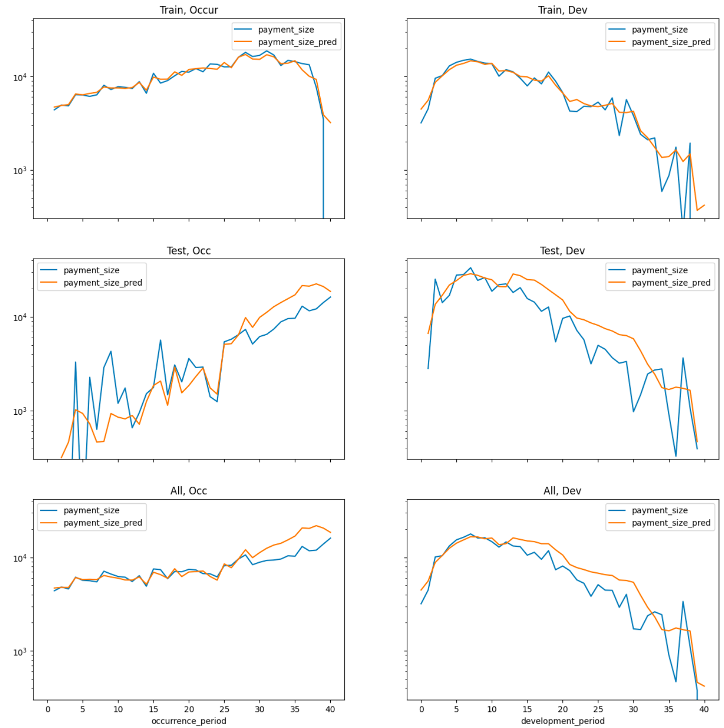
Customising the structure



- “SplineNet”, our customized design
- Using only occurrence and development periods only (but on individual data)
- Fits trends slightly better

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Gradient Boosting Benchmark



- Our models capture claim data in a tabular format.
- Gradient boosted decision trees often perform well on tabular data.
- Plug-in replacement to GLM with non-linear capability.
- Decision trees models may see more step changes
- Does not fit that closely in this instance

Institute and Faculty of Actuaries

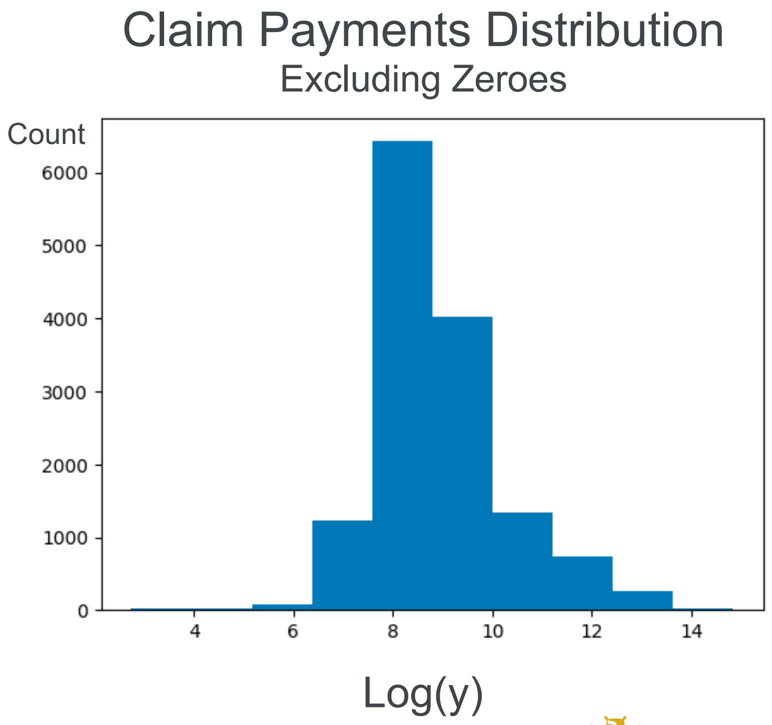
Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Institute
and Faculty
of Actuaries

Probabilistic neural networks

Probabilistic models

- Output a distribution – capturing the variability of the data, not just a point estimate
- Our payment data has many records with zero payments a given period for a given claim
- Excluding the zeroes - distribution is skewed, looks normal only after log-scaling



Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

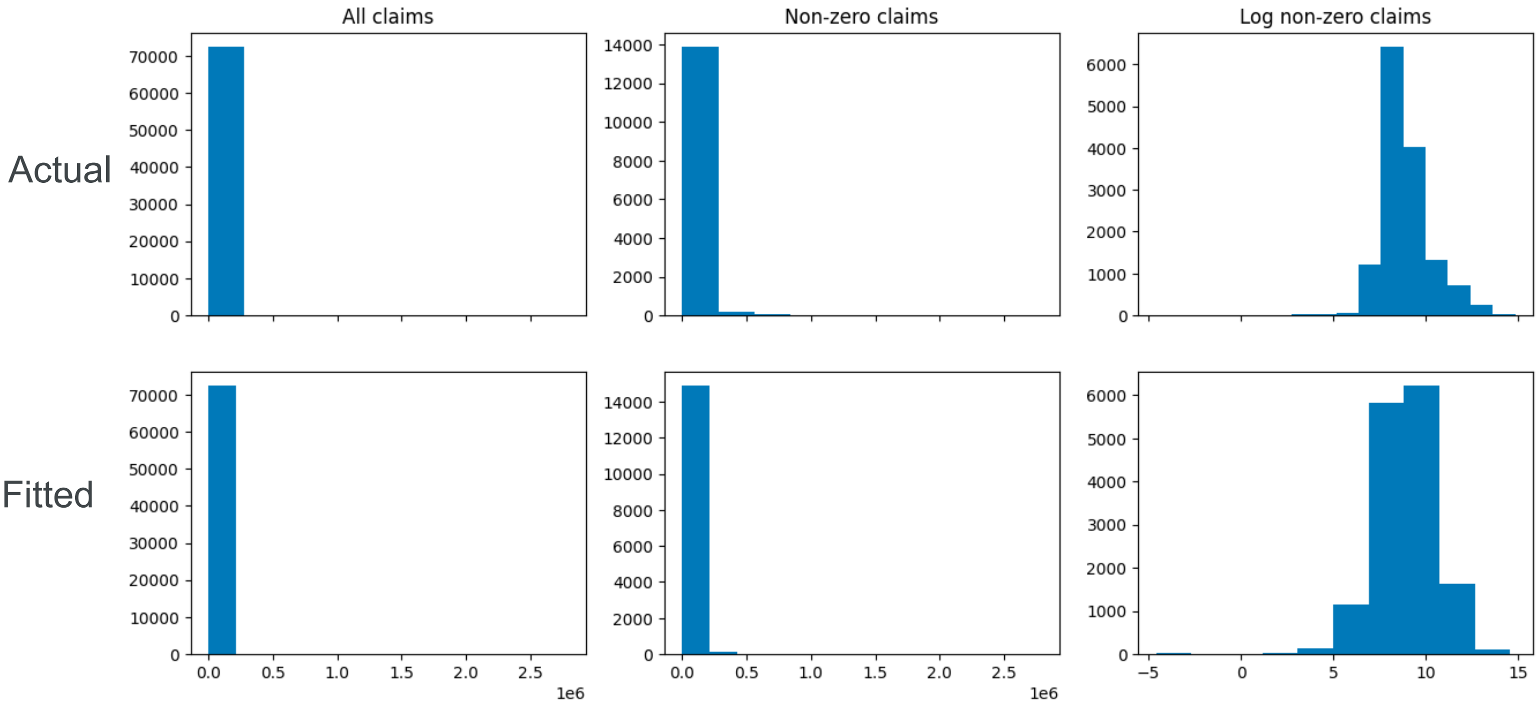
Lognormal Mixture Density Network

- Output variable modelled as the weighted sum of log-normal distributions.
- α is weight of each distribution
- μ and σ is lognormal's μ and σ
- Mean is $\sum_{k=1}^{n} a_k \cdot e^{(\mu_k + \frac{\sigma_k^2}{2})}$
- Some tricks to ensure numerical stability (details in notebook)

```
SMALL = 1e-7

def log_mdn_loss_fn(y_dists, y):
    y = torch.log(y + SMALL) # log(y) ~ Normal
    alpha, mu, sigma = y_dists
    m = torch.distributions.Normal(loc=mu, scale=sigma) # Normal
    loss = -torch.logsumexp(m.log_prob(y) + torch.log(alpha + 1e-15), dim=-1)
    return torch.mean(loss) # Average over dataset
```

Institute and Faculty of Actuaries

Distributions: Actual vs fitted



Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

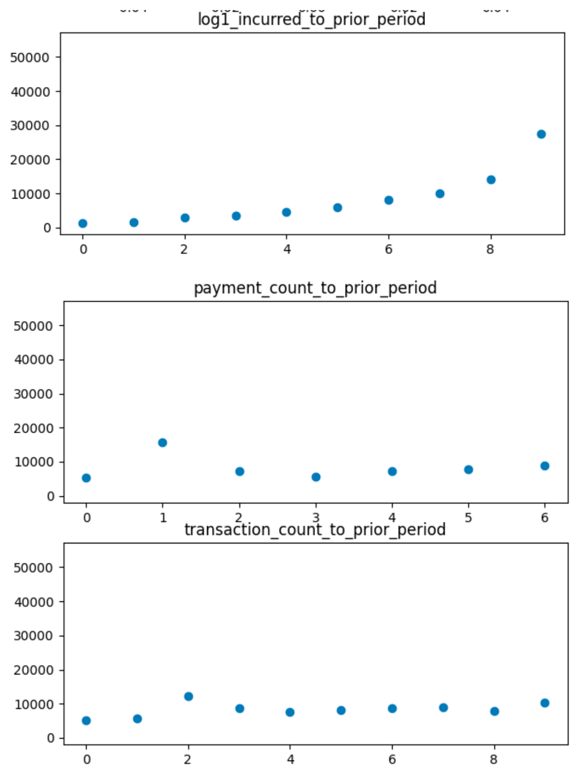
Institute and Faculty of Actuaries

Institute and Faculty of Actuaries

Improving the model?

Using detailed features

- Key advantage of using individual data is for utilizing claim level information
- In real life scenarios, this may include peril coding, claim descriptions, policy information.
- In our simulated data, mostly limited to features engineered from claims history.



Institute and Faculty of Actuaries

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Cross-Validation and Hyperparameter Search

- Hyperparameter search – find best model parameters:
  - Model size: Neurons in hidden layer
  - Regularisation: Lasso penalty, weight decay, dropout
- "Rolling origin" cross validation
- Claim history feature engineering

Institute and Faculty of Actuaries

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Summary:

- Individual, granular models can be valuable in some circumstances
- Neural networks can effectively model trends in claims data:
  - Reflect trends
  - Potential to use detailed claims information
  - Probabilistic output
- Link: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/

Dataset 5 Leaderboard

Met hod	Outstanding Claims Liability	Period Level MSE	Period Level Absolute Error	Total OCL Absolute Error	Total OCL Absolute Percent Error	
1	True Ultimate	415,208,224.0	0.0	0.0	0.0	0.0
7	D/O SplineNet	377,212,928.0	17,353,778.0	72,897,200.0	37,995,252.0	9.2
8	D/O SplineMDN	377,212,928.0	17,353,778.0	72,897,200.0	37,995,252.0	9.2
6	D/O ResNet	456,777,888.0	16,902,854.0	75,666,032.0	41,569,676.0	10.0
10	Detailed ResNet	478,930,661.1	41,708,402.1	157,432,251.7	63,722,452.3	15.3
11	Detailed SplineNet	483,915,060.2	42,700,794.0	161,107,670.6	68,706,851.4	16.5
12	Detailed SplineMDN	483,915,060.2	42,700,794.0	161,107,670.6	68,706,851.4	16.5
9	D/O SplineNet CV	548,784,064.0	31,238,146.0	146,933,376.0	133,575,840.0	32.2
2	Chain Ladder	553,335,232.0	43,015,804.0	174,280,848.0	138,127,024.0	33.3
3	GLM Chain Ladder	553,337,792.0	43,016,100.0	174,282,560.0	138,129,600.0	33.3
13	Detailed GBM	563,046,264.0	54,976,179.2	193,207,856.1	147,838,055.2	35.6
5	GLM Spline	565,708,352.0	35,842,056.0	164,710,784.0	150,500,096.0	36.2
4	GLM Individual	574,341,184.0	42,662,360.0	174,327,264.0	159,132,944.0	38.3
0	Paid to Date	0.0	102,023,488.0	415,208,224.0	415,208,224.0	100.0

NN's do well for dataset 5: detailed features not leading to stronger predictions.

Institute and Faculty of Actuaries

Questions

Comments

The views expressed in this presentation are those of invited contributors and not necessarily those of the IFoA. The IFoA do not endorse any of the views stated, nor any claims or representations made in this presentation and accept no responsibility or liability to any person for loss or damage suffered as a consequence of their placing reliance upon any view, claim or representation made in this presentation.

The information and expressions of opinion contained in this publication are not intended to be a comprehensive study, nor to provide actuarial advice or advice of any nature and should not be treated as a substitute for specific advice concerning individual situations. On no account may any part of this presentation be reproduced without the written permission of the authors.

Institute and Faculty of Actuaries

Link to code: https://institute-and-faculty-of-actuaries.github.io/mlr-blog/post/research/chain_ladder_to_individual_mdn/