PnC Reinsurance Modeling Using NumPy and TensorFlow

Pauli Rämö, Roland Schmid

16 July 2018

Mirai Solutions - www.mirai-solutions.com

Problem Description

- Insurance companies require detailed insights into risks arising from claim losses in order to determine adequate **reinsurance strategies**.
- To model rare events incurring large losses, large-scale simulations are required to obtain stable risk estimates and other statistics
  - Expected Shortfall (ES / CVaR)
  - Value at Risk (VaR)
  - Statistics at high-resolution business unit and reinsurance contract level

Simulation of Reinsurance Contracts

- We simulate aggregate large loss data through convolution of frequency and severity distributions
  - Frequencies from a Poisson distribution
  - Severities from a Pareto distribution
- The simulation creates a matrix of gross losses
  - 1'000'000 Monte-Carlo simulations
  - 100+ nodes (business units and line of business)
- We model two kinds of reinsurance contract types and apply them to all simulations
  - Excess of Loss
  - Surplus share

Reinsurance Contracts - Excess of Loss

- Level 2 contracts apply on top of level 1 contracts, i.e. they apply to losses net of level 1 reinsurance rather than gross
- **Node** is defined as the combination of BU and LoB
- Excerpt of a realistic portfolio of reinsurance treaties:

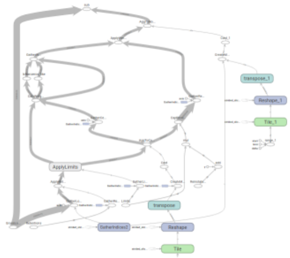
Level	Contract.No	CedingUnit	CedingLoB	Retention	Limit	Reinstatement
1	GB1	London-BU	Property	500000	250000	2
1	GB1	London-BU	Property	750000	250000	2
1	GB1	London-BU	Property	1000000	500000	1
1	GB2	London-BU	Marine/Aviation	500000	250000	2
1	GB2	London-BU	Marine/Aviation	750000	250000	2
1	GB2	London-BU	Marine/Aviation	1000000	500000	1

Using TensorFlow for Reinsurance Contract Modeling

- As a case study, we implement the reinsurance models with NumPy and TensorFlow in Python
    - Does it add value to use TensorFlow instead of standard NumPy?
- Google's TensorFlow is a framework designed for big data analytics, particularly in machine learning
    - Computational graphs & lazy execution
        - pre-optimization of code execution
    - High performance
    - IT framework with active development community
    - Visualization and profiling tools
    - CPU / GPU / TPU & Google Cloud support
- On top of machine learning, TensorFlow can be used for any computation task suitable to tensor mathematics

TensorFlow: Session Setup

```
# Initialize variables
grossLossesTF   = tf.placeholder('float64', grossLosses.shape, 'GrossLosses')
limitsTF        = tf.placeholder('float64', limits.shape, 'Limits')
retentionsTF    = tf.placeholder('float64', retentions.shape, 'Retentions')
reinstatementTF = tf.placeholder('float64', reinstatement.shape, 'Reinstatement')

# Define calculation graph
netLossesTF = tf_calculate_netLosses(grossLossesTF, limitsTF, retentionsTF, reinstatementTF)

# Run TensorFlow session
sess        = tf.Session()
run_options = tf.RunOptions(trace_level=tf.RunOptions.FULL_TRACE)
run_md      = tf.RunMetadata()
writer      = tf.summary.FileWriter('logreins', sess.graph)

netLossesFromTF = sess.run(netLossesTF, feed_dict={
    grossLossesTF        : grossLosses,
    limitsTF             : limits,
    retentionsTF         : retentions,
    reinstatementTF      : reinstatement
}, options=run_options, run_metadata=run_md)

writer.close(); sess.close()

# Define two-level graph by nesting
netLosses2TF = tf_calculate_netLosses(
                    tf_calculate_netLosses(
                        grossLossesTF, limitsTF, retentionsTF, reinstatementTF
                    ), limitsTF, retentionsTF, reinstatementTF)
```

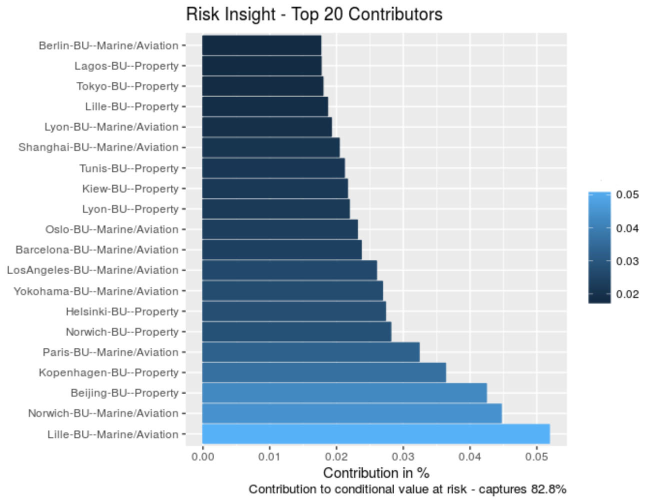


NumPy / TensorFlow: Loops / Tensors

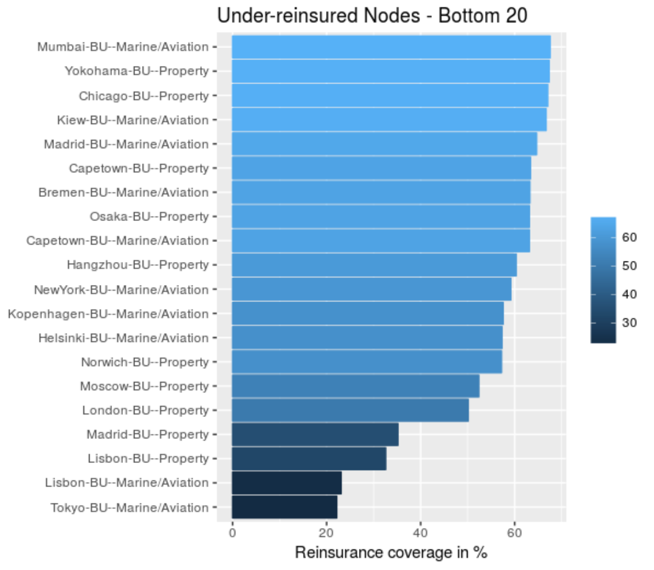
```
# NumPy code using a for-loop over contracts
def calculate_netLosses(grossLosses, contracts, contractMap, limits, retentions):

    for i in range(contracts.size):                                                      # loop over contracts
        losses      = grossLosses[:, contractNodeMapBool[i,]]                             # losses for contract
        retained    = np.maximum(np.minimum(losses - retentions[i], limits[i]), 0.)      # main formula
        retainedtot = retained.sum(axis=1, keepdims=False)                               # sum over nodes
        recovered   = np.minimum(retainedtot, maxrecovery[i])                            # reinstatement limit
        weights     = retained / retainedtot                                             # weight
        ceded       = weights * recovered                                                # weighted coverage
        cededLosses[:, contractMap[i,]] += ceded                                         # sum up

    netLosses = grossLosses - cededLosses # final net losses
    return netLosses
```

```
# TensorFlow code using tensors
def tf_calculate_netLosses(grossLossesTF, contractMapTF, limitsTF, retentionsTF):

    # tensor indices
    idcs = tf.where(contractMapTF)
    idxc = tf.reshape(idcs[:, 0], [tf.shape(idcs)[0]])

    # map from nodes to contracts
    map1 = tf.equal(idxc, tf.transpose(tf.reshape(tf.tile(tf.range(start=0,
             limit=contractMapTF.shape[0], dtype=tf.int64), [tf.shape(idcs)[0]]
         ), shape=[tf.shape(idcs)[0], tf.shape(contractMapTF)[0]])))
    # display debug
    map1 = tf.Print(map1, [map1], summarize=100, message="map1: ")

    # calculate retained
    losseslev = tf.gather(grossLossesTF, colsc, axis=1)
    retlev    = tf.gather(retentionsTF, idxc)
    retainlev = tf.clip_by_value(
        tf.subtract(losseslev, retlev, name='ApplyRetention'),
        clip_value_min=ZERO,
        clip_value_max=limlev,
        name='ApplyLimits'
    )
    retained = tf.matmul(retainlev, tf.cast(map1, dtype=tf.float64),
                transpose_b=True, b_is_sparse=True, name='AggToContracts')
```

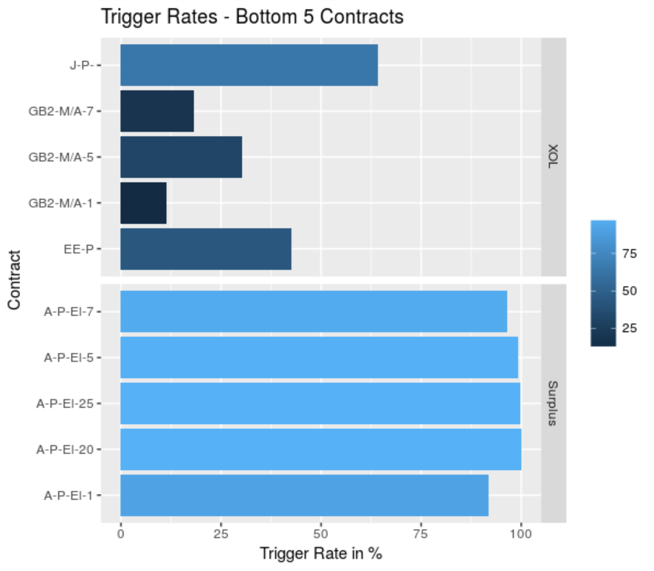
Results - Loss Contributions



Results - Under-reinsured Nodes



Results - Trigger Rate of Contracts



Results - Execution Times on Google Cloud Platform

- Ubuntu 16.04 LTS
- 1 GPU: NVIDIA Tesla P100 - 16GiB of HBM2 memory
- 2 virtual CPUs (Intel Sandy Bridge) - 48GiB of RAM

code	CPU	GPU	TPU
NumPy	115s	—	—
TensorFlow	5.8s	1.3s	0.?!s

- 1'000'000 simulations for 76 XoL-affected nodes

Discussion

- TensorFlow can be used for reinsurance contract modeling
    - **PRO**: All TensorFlow utilities readily available
        - TensorBoard, GPU / TPU / Google Cloud support
    - **PRO**: Increased performance compared to NumPy (GPU)
    - **CON**: More difficult to program
        - No interactive line-by-line programming style
        - Requires switching to "tensor-mode" mindset
- Many factors make the choice of approach a case-by-case decision. In particular, detailed problem modeling aspects:
    - Desired simulation sizes
    - Number and complexity of coverages (reinsurance contracts)
    - Potential need for granular high-resolution insight
    - Need for extensive scenario, sensitivity, or uncertainty analysis

Thank You

- Any questions or comments?

- pauli.ramo@mirai-solutions.com
- www.mirai-solutions.com