# Assignment 4 – Praktikum Smart Data Analytics, Group 4

## Task 1

### Benchmark-Methode

Für die Durchführung der Evaluation haben wir das Causeme-Example-Skript von der Causeme-Plattform angepasst. Das Skript führt eine ausgewählte Methode für Kausale Inferenz auf allen Daten aus einem Benchmark-Datensatz aus und erzeugt aus den Ergebnissen eine Datei zum Upload auf die Causeme-Plattform.

Das causeme-runner.py Skript integriert dabei PCMCI aus dem Packet TIGRAMITE sowie TCDF. Die Auswahl von anzuwendendem Algorithmus, Benchmark-Gruppe und Datensatz erfolgt per Kommandozeilenparameter.

Das Skript ist für den Einsatz in der SDIL-Batch-Umgebung optimiert (es kann jedoch auch lokal ausgeführt werden und verwendet dann automatisch relative statt absoluter Pfade). Die auszuführenden Experimente werden in der Datei experimente.txt eingetragen, HTCondor erzeugt dann pro Experiment einen Job. Um den Fortschritt während der Bearbeitung der (meist 200) einzelnen Teildatensätzen zu verfolgen, werden im Ordner progress Dateien angelegt, deren Dateinamen dem aktuellen Datensatz entsprechen.

**Wichtig**: Vor dem Einsatz des Skripts müssen in allen Dateien die absoluten Pfade angepasst werden (notwendig aufgrund des HTCondor Systems).

### Egebnisse

Im Folgenden werden die Methoden PCMCI, TCDF und Transferentropie betrachtet, um kausale Inferenzen in den Causeme-Datensätzen zu erkennen. Hierbei wurden zunächst generelle Performance Analysen mit verschiedensten Experiment Datensätzen durchgeführt. Darauf basierend wurden die Gruppen **nongauss VAR** und **TestWeath** für einen detaillierteren Vergleich der Methoden ausgewählt.

Um ein Gefühl für die Performance und Laufzeiten der einzelnen Methoden zu erhalten, wurden die Methoden zunächst mit ihren Standardparametern auf einzelnen Datensätzen in den Gruppen linear VAR, linear VAR aggregated, nongauss VAR, TestCLIM und TestWeath angewandt. Hierbei wurde bereits deutlich, dass die PCMCI Methode auf den linearen Datensätzen (linear var, linear var aggregated) mit einer Durchschnitts-AUC von 0.95 und einer Durchschnitts False-Positive-Rate von 0.06[[1]](#footnote-2) sehr gute Resultate erzielt. Auf den restlichen Datensätzen (nongauss, river-runoff, TestClim, TestWeath) erzielten wir mit der PCMCI Methode eine Durchschnitts-AUC von 0.87.

Bei den ersten Durchläufen der PCMCI Methode, aber auch der TCDF Methode, mussten wir feststellen, dass die Durchführung eines Experiments vor allem bei den höherdimensionalen Datensätze unter Umständen länger als 12h in der SDIL-Umgebung benötigt. Unter Berücksichtigung der Vorgabe, dass die gewählten Methoden für alle Datensätze innerhalb einer Gruppe durchlaufen müssen, haben wir uns im Folgenden dazu entschieden die zwei Gruppen **nongauss VAR** und **TestWeath** zu verwenden.

Die PCMCI Methode erzielte für die TestWeath Gruppe eine Durschnitss-AUC von 0.83, bei einer Durchschnitts-False-Positive-Rate von 0.1 (Figure 1). Die Kausalbeziehungen in der nongauss Gruppe wurden von der PCMCI Methode nochmal etwas besser erkannt (Durschn.AUC: 0.91, Durchschn.FPR: 0.06).

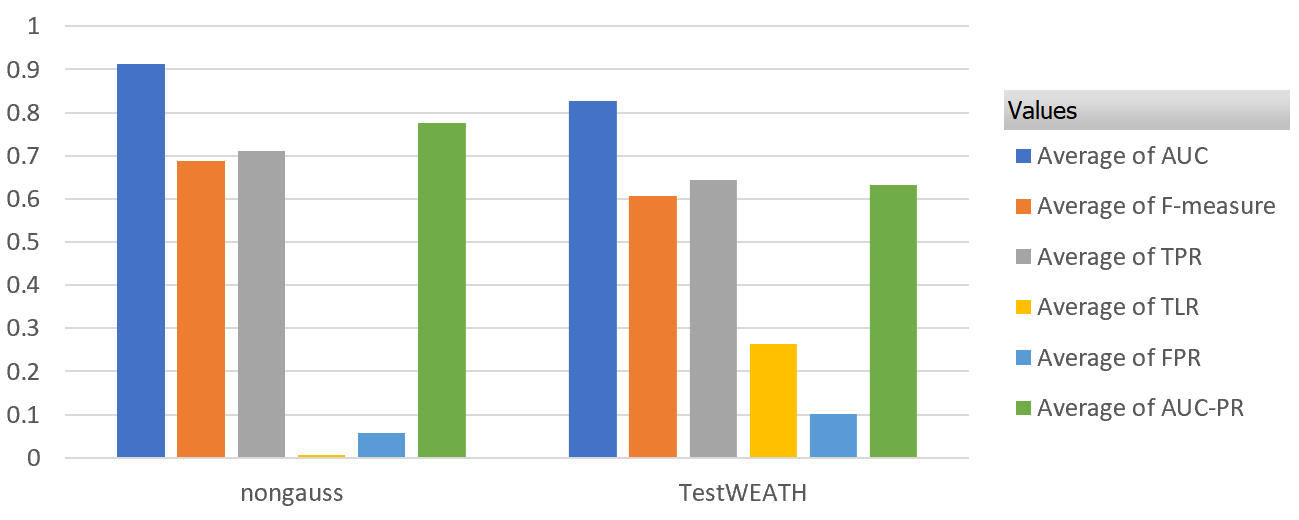


Figure 1: PCMCI Performance

Die nongauss Gruppe bietet sich zudem augrund der höheren Anzahl an Experimenten (12) an, um die Standardabdweichungen der Scores für die PCMCI Methode zu betrachten (Stdev.FPR = 0.01; Stdev.TPR = 0.14).

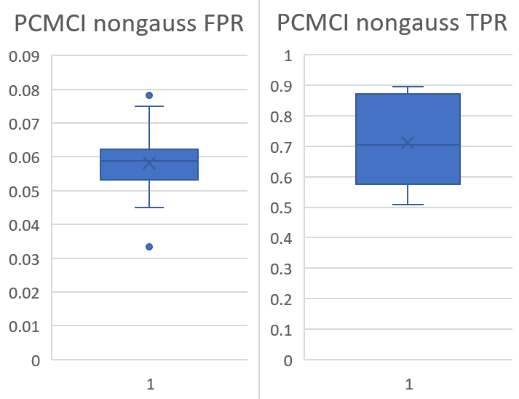


Figure 2:Boxplots der FPR und TPR Scores der PCMCI Methode für die nongauss Gruppe

Die Transferentropie wurde als weiteres Werkzeug für die Analyse von Kausalbeziehungen in den **nongauss** und **TestWeath** Gruppen verwendet. Dafür wurde das R Paket “RTransferEntropy” angewandt. Die Transferentropie erzielte mit einer Durchschnitts-AUC von 0.63 und einer Durchschnitts-AUC-PR von 0.36 schlechtere Werte als unsere ursprünglich betrachtete PCMCI Methode. Darüberhinaus bietet das “RTransferEntropy” Paket keine bereits implementierte Funktion, um sich die paarweisen P-Werte ausgeben zu lassen.

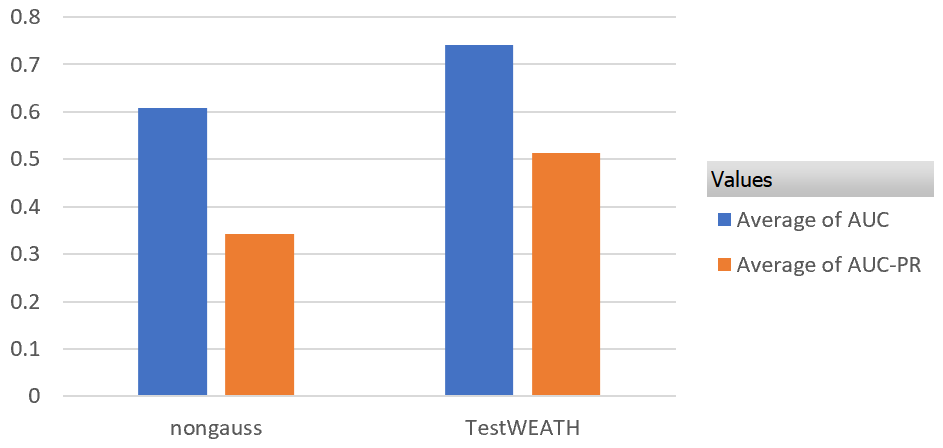


Figure 3: Transferentropie Performance

Die TCDF Methode hingegen hat mit einer Durchschnitts-AUC von 0.51 keine guten Resultate erzielen können (Figure 4).

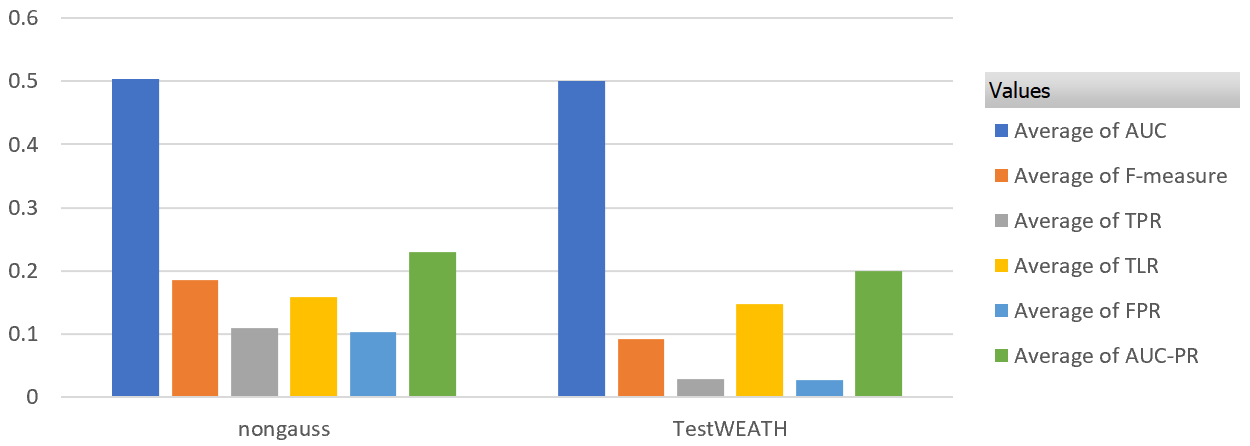


Figure 4: TCDF Performance

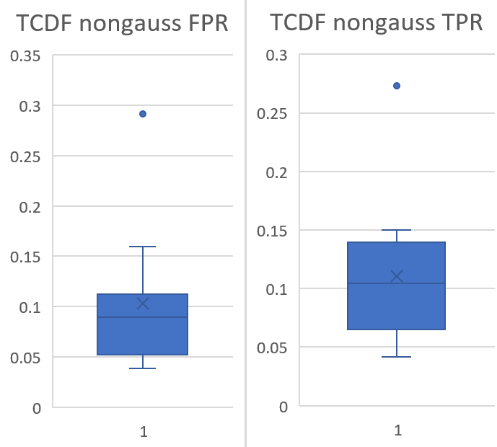


Figure 5: Boxplots der FPR und TPR Scores der TCDF Methode für die nongauss Gruppe

### Hyperparameter

Daher haben wir im nächsten Schritt versucht, die Performance von TCDF über die Wahl der Hyperparameter zu verbessern. Um möglichst robuste Hyperparameter zu erhalten, haben wir zur Evaluation hier wieder den Mix aus einzelnen Experimenten aus mehreren unterschiedlichen Benchmark-Gruppen angewendet und versucht, diesen auf jeweils 5 verschiedenen Hyperparameter-Settings auszuführen. Da jedoch mit veränderten Hyperparametern die TCDF-Methode zum Teil noch einmal deutlich länger benötigt, konnten wir nicht alle Settings auf allen Datensätzen evaluieren (aus diesem Grund haben wir für die Hyperparametersuche auch generell nur „kleine“ Datensätze verwendet). Nachfolgend exemplarisch einige Resultate, die vollständigen Ergebnisse befinden sich in der Ergebnisübersicht:

linear-VAR\_N-3\_T-150:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Setting | Time | AUC | F-Score | FPR | TPR |
| setting\_0 | 8,7968 | 0.5333 | 0.4918 | 0.2717 | 0.3383 |
| setting\_1 | 15,1052 | 0.5375 | 0.5076 | 0.3033 | 0.3783 |
| setting\_2 | 43,5577 | 0.5308 | 0.4935 | 0.2917 | 0.3533 |
| setting\_3 | 41,7921 | 0.5317 | 0.4816 | 0.25 | 0.3133 |
| setting\_4 | 8,1738 | 0.5358 | 0.5018 | 0.2917 | 0.3633 |

linear-VAR\_aggregated\_N-3\_T-300\_agg-3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Setting | Time | AUC | F-Score | FPR | TPR |
| setting\_0 | 6,3644 | 0.5192 | 0.4721 | 0.2883 | 0.3267 |
| setting\_1 | 10,2512 | 0.5017 | 0.4698 | 0.37 | 0.3733 |
| setting\_2 | 31,8925 | 0.5217 | 0.473 | 0.2783 | 0.3217 |
| setting\_3 | 31,1702 | 0.5017 | 0.4457 | 0.3033 | 0.3067 |
| setting\_4 | 6,4375 | 0.5175 | 0.4913 | 0.36 | 0.395 |

River-Runoff:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Setting | Time | AUC | F-Score | FPR | TPR |
| setting\_0 | 91,102 | 0.5455 | 0.3333 | 0.0 | 0.0909 |
| setting\_1 | 212,4243 | 0.6322 | 0.5556 | 0.0083 | 0.2727 |
| setting\_2 | 595,2961 | 0.6736 | 0.5714 | 0.0165 | 0.3636 |
| setting\_3 | 500,9142 | 0.5909 | 0.5263 | 0.0 | 0.1818 |
| setting\_4 | 92,5449 | 0.6322 | 0.5556 | 0.0083 | 0.2727 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| setting\_nr | layers | epochs | lr | significane |
| 0 | 0 | 1000 | 0.01 | 0.8 |
| 1 | 1 | 1000 | 0.01 | 0.8 |
| 2 | 0 | 5000 | 0.01 | 0.8 |
| 3 | 0 | 5000 | 0.001 | 0.8 |
| 4 | 0 | 1000 | 0.01 | 0.9 |

Bei Betrachtung der vollständigen Ergebnisse wird deutlich, dass Setting 1 in fast allen Fällen eine Verbesserung der Performance bei Vertretbarer höherer Rechenzeit erzielt. Dieses Setting kann daher als unser Robustestes Setting bezeichnet werden. Das Hinzufügen eines zusätzlichen Layers lohnt sich daher in vielen Fällen.

Ein wichtiger Hyperparameter ist außerdem die Kernel-Größe. Diese bestimmt, welcher maximale Zeitliche Versatz erkannt werden kann und ist nicht robust, sondern muss auf die jeweiligen Datensätze individuell angepasst werden.

Aufgrund von fehlender Rechenzeit konnten wir die obigen vollständigen Benchmark-Datensätze leider nicht mit dem optimierten Hyperparameter-Setting wiederholen (die noch einmal deutlich rechenintensiver sind als das Standard-Setting).

PCMCI benötigt als Hyperparameter ebenfalls den maximalen zeitlichen Versatz. Außerdem muss ein Verfahren für den Independence-Test gewählt werden. Hier haben wir nur mit dem ParCorr-Test gearbeitet, der eigentlich nur für lineare Daten gedacht ist, aber auch auf den von uns verwendeten nichtlinearen Datensätzen gute bis akzeptable Ergebnisse erzielte.   
Die weiteren Tests (GPDC, CMIknn) sind noch einmal deutlich rechenintensiver und waren daher mit der uns verfügbaren Zeit und Ressourcen nicht sinnvoll auf den Benchmark-Gruppen ausführbar.

### Schlussfolgerung

Wir können schlussfolgern, dass PCMCI mit unseren Benchmarks auf allen betrachteten Metriken (F-Measure, TPR, etc.) die beiden anderen Methoden outperformed. Zwar ist die Runtime für die Transferentropie deutlich schneller als die der PCMCI oder der TCDF Methode, sie bietet allerdings aufgrund der fehlenden p-Wert Funktion nicht den gleichen Informationsgehalt wie die TCDF oder die PCMCI Methode als Vergleich an.

Wir haben durch den Vergleich der Scores auf den verschiedenen nongauss-Datensätze erkennen können, dass die PCMCI Methode allerdings auch variierende Performance aufweist, insbesondere bei höheren Dimensionen der Datensätze und mit Noise (Siehe Figure 2).

Unter den betrachteten Methoden gibt es daher keine, die durchgehend und zuverlässig gute Ergebnisse liefert. Es ist daher wichtig, auf einem praktischen Problem unterschiedliche Methoden auszuprobieren und die Ergebnisse idealerweise mit Domänenwissen zu validieren.

1. Die zugrundeliegenden Ergebnisse der einzelnen Durchläufe können unter folgendem Link aufgerufen werden: <https://git.scc.kit.edu/ubelj/psda-group-4-assignment-4/-/tree/master/causeme> [↑](#footnote-ref-2)