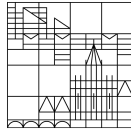


Universität  
Konstanz



Bundesamt  
für Sicherheit in der  
Informationstechnik

UNIVERSITÄT KONSTANZ  
FACHBEREICH MATHEMATIK UND STATISTIK  
&  
BUNDESAMT FÜR SICHERHEIT IN DER  
INFORMATIONSTECHNIK

MASTERARBEIT ZUM THEMA:

# Untersuchung & Entwicklung von Ansätzen zur Detektion von Poisoning-Angriffen

vorgelegt von

Lukas Schulth  
`lukas.schulth@uni.kn`

unter der Betreuung von

Erstkorrektor:

Herr Prof. Dr. Johannes Schropp  
`johannes.schropp@uni.kn`

Zweitkorrektor:

Herr Prof. Dipl.-Ing. Markus Ullmann  
`markus.ullmann@bsi.bund.de`

Herr Dr. Christian Berghoff  
`christian.berghoff@bsi.bund.de`

Herr Matthias Neu  
`matthias.neu@bsi.bund.de`

1. Oktober 2021

---

## Abstract

english

## Zusammenfassung

deutsch

TODO:

- Algorithmen aufschreiben
- Promotion anschauen von vayer [?]

**Keywords**— one, two, three, four

## Abbildungsverzeichnis

1	(Optischer) Vergleich von korruptem Datenpunkt und berechneter Heatmap. . . . .	17
2	Beispiel isometrischer metrischer Maßräume . . . . .	24
3	Monge-Abbildung . . . . .	25
4	Transportpläne im diskreten, semi-diskreten und kontinuierlichen Fall . . . . .	27
5	Visualisierung der Hausdorff-Distanz zwischen den Mengen $X$ und $Y$ in $\mathbb{R}^2$ . . . . .	36
6	Beispiel der Gromov-Wasserstein-Distanz . . . . .	37
7	Ergebnis des spektralen Clusterings unter Verwendung der Euklidischen Distanz und $k=10$ Nachbarn . . . . .	45
8	Auswahl der relevantesten Pixel (bis zu 99% der Gesamtmasse) zweier Heatmaps . . . . .	46
9	Auswahl der relevantesten Pixel (bis zu 50% der Gesamtmasse) zweier Heatmaps . . . . .	47
10	Einbettung des Baryzentrums mithilfe von Multidimensionaler Skalierung(MDS) bei der Wahl von 99% der Gesamtmasse . . . . .	48
11	Angriffserfolgsrate pro Klasse bei Clean-Label-Poisoning-Attacks für verschiedene Werte $amp$ des Amplitudenstickers in vierfacher Ausföhrung bei Abstand $d = 10$ zum Rand . . . . .	50

## Tabellenverzeichnis

1	Für einen Poisoning-Angriff interessante Klassen und die zugehörige Anzahl an Bildern. . . . .	7
2	Auswertung des Clusterings auf den rohen Bilddaten . . . . .	44
3	Vergleich von Angriffen und Verteidigungen für SPA . . . . .	44
4	Qualität der Angriffe auf das Inception v3-Netz mit Stickern Seitenlänge 2 und 3 Pixel bei unterschiedlich großen Anteilen an korrupten Daten . . . . .	49

## Listings

1	python-interner Aufbau einer BatchConv Schicht . . . . .	6
2	Verfügbare Schichten und Aktivierungsfunktionen . . . . .	17
3	Implementierte Regeln fhvilshoj . . . . .	18
4	Augmentierung beim Einlesen der Daten . . . . .	51

## Algorithmenverzeichnis

1	Berechnung der $GW_{\varepsilon}$ Baryzentren . . . . .	42
2	Sinkhorn-Algorithmus . . . . .	43

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>5</b>
<b>2</b>	<b>Neuronale Netzwerke</b>	<b>5</b>
2.0.1	CNNS	6
2.0.2	Besondere Schichten	6
2.0.3	Inception v3	6
2.0.4	VGG16	7
2.1	Datensatz	7
<b>3</b>	<b>Poisoning-Angriffe</b>	<b>8</b>
3.1	Standard Poisoning-Angriffe	8
3.2	Label-konsistente Poisoning-Angriffe	8
3.3	Bewertung von Poisoning-Angriffen	9
3.4	Verteidigungen	10
3.4.1	Referenzwert: kMeans(k=2)	10
3.4.2	Activation Clustering	10
3.5	Implementierung	11
3.5.1	Methoden zur Untersuchung, ob ein Angriff vorliegt	11
3.5.2	Entfernen von korruptierten Datenpunkten	12
3.5.3	Heatmap Clustering	13
<b>4</b>	<b>Erklärbare KI</b>	<b>13</b>
4.1	Lokale Methoden	13
4.2	Globale Methoden	13
<b>5</b>	<b>Layer-wise Relevance Propagation</b>	<b>13</b>
5.1	Idee	13
5.2	Behandlung von biases	15
5.3	Beispiel an einem kleinen Netzwerk	15
5.4	Deep Taylor Decomposition	15
5.4.1	Taylor Decomposition	16
5.4.2	Deep Taylor Decomposition	16
5.5	Verschiedene Verfahren	16
5.6	Eigenschaften	16
5.7	Behandlung besonderer Schichten	16
5.7.1	BatchNorm2D	16
5.8	LRP für Deep Neural Nets/Composite LRP	16
5.9	Verarbeitung der Heatmaps	16
5.10	Implementierungen	17
5.10.1	Tensorflow	17
5.10.2	pytorch	17
<b>6</b>	<b>Detektion von Poisoning-Angriffen basierend auf LRP</b>	<b>19</b>
6.1	Idee	19
6.2	k-means / k-means++ -Clustering	19
6.3	Spektrales Clustering	20
6.4	Anwendung auf unterschiedliche Poisoning-Angriffe	20
6.5	Verwendete Distanzen & Approximationen	21

<b>7</b>	<b>Optimal Transport</b>	<b>21</b>
7.1	Einführung . . . . .	22
7.1.1	Frage: . . . . .	22
7.1.2	Anwendungsgebiete . . . . .	22
7.2	Notation . . . . .	25
7.3	Kantorovichs Optimal Transport . . . . .	25
7.3.1	Optimaler Transport (Monge Formulierung) . . . . .	25
7.3.2	Optimaler Transport nach Kantorovich . . . . .	26
7.3.3	Metrische Eigenschaften . . . . .	27
7.3.4	Duale Formulierung . . . . .	29
7.3.5	Verfahren zur Berechnung der exakten Lösung . . . . .	29
7.3.6	Komplexitätsanalyse . . . . .	29
7.3.7	Wasserstein-Baryzentren . . . . .	29
7.4	Entropisch regularisierte Wasserstein-Distanz . . . . .	30
7.4.1	Algorithmus von Sinkhorn und Variationen . . . . .	31
7.4.2	Komplexitätsanalyse . . . . .	35
7.5	Gromov-Wasserstein-Divergenz . . . . .	35
7.5.1	Gromov-Wasserstein-Divergenz . . . . .	35
7.5.2	Gromov-Wasserstein Baryzentren . . . . .	40
7.5.3	Komplexitätsanalyse . . . . .	43
7.5.4	Implementierung . . . . .	43
<b>8</b>	<b>(Numerische) Ergebnisse/Vergleich mit anderen Verfahren</b>	<b>43</b>
8.0.1	Standard Poisoning-Angriffe . . . . .	43
8.1	Clustering auf den Rohdaten . . . . .	43
8.1.1	Label-konsistente Poisoning-Angriffe . . . . .	45
8.2	Activation Clustering . . . . .	48
8.3	Räumliche Transformationen . . . . .	48
<b>9</b>	<b>Weitere mögliche Schritte</b>	<b>49</b>
<b>10</b>	<b>Zusammenfassung und Ausblick</b>	<b>50</b>
<b>A</b>	<b>Verwendete Netzwerke</b>	<b>51</b>
A.1	Net . . . . .	51
<b>B</b>	<b>Parameter für Training und Einlesen der Daten</b>	<b>51</b>
<b>C</b>	<b>Einlesen der Daten bei AC</b>	<b>51</b>
<b>D</b>	<b>Parameter für die ausgeführten Angriffe</b>	<b>52</b>
<b>E</b>	<b>Datensätze</b>	<b>52</b>
<b>F</b>	<b>Programmcode</b>	<b>52</b>
<b>G</b>	<b>Notizen</b>	<b>52</b>

# 1 Einführung

Datengewinnung(LRP) Datenverarbeitung(kMeans, Gromov Wasserstein) Pweave<sup>1</sup>

A Complete List of All (arXiv) Adversarial Example Papers <sup>2</sup>

In sicherheitskritischen Anwendungsgebieten ist die Erklärung für das Zustandekommen einer Entscheidung genauso wichtig wie die Entscheidung selbst [?].

Clustering auf Datenpunkten direkt( 50 Prozent = raten), Clustering auf Aktivierungen gut geeigneter Netzwerkschichten. Clustering auf den Heatmaps der verdächtigen Klasse.

Clustering auf unterschiedlichen Repräsentationen der Bilder:

- Clustering direkt auf den Bildern
- Clustering auf den Activations einer Netzwerkschicht(Im Paper [?] wird die vorletzte Schicht benutzt)
- Clustering auf den Heatmaps

In Abschnitt 2 geben wir eine kurze Einführung in Neuronale Netzwerke und stellen die untersuchten Modelle vor. Abschnitt 3 führt in die unterschiedlichen Möglichkeiten eines Poisoning-Angriffs auf Neuronale Netzwerke ein. Abschnitt 4 gibt eine kurze Übersicht über den Bereich der Erklärbaren Künstlichen Intelligenz, wobei ein Beispiel eines Verfahrens, die sogenannte Layer-wise Relevanz Propagation ausführlich in Abschnitt 5 vorgestellt wird. Kern der Arbeit bildet Abschnitt 6, wo wir zu Beginn die grundlegenden Bestandteile des Algorithmus zur Detektion von Poisoning-Angriffen auf Neuronale Netzwerke erklären, bevor die experimentellen Ergebnisse in Unterabschnitt 6.4 ausführen. Ein Vergleich mit anderen Detektionsverfahren wird in Abschnitt 8 durchgeführt.

## 2 Neuronale Netzwerke

Wir betrachten ein Neuronales Netzwerk (NN), dass die Funktion  $f_{\theta} : \mathbb{R}^n \rightarrow \mathbb{R}$ , mit  $\theta = (w_{il}, b_{il})$  beschreibt. i: Schicht l: Neuron in der Schicht w: Gewichte b: Bias g: nichtlineare Aktivierungsfunktion Architektur, Modell Pre-Activations(lokal, global):  $z_{ij} = x_i * w_{ij}$   $z_j = \sum_i z_{ij} + b_j$

Vorschrift/aktivierungen:  $x_j = g(z_{ij})$

Training;testing, Validation, Forward pass, backward pass SGD erklärt im Einführungsteil von [?]

fehlende Interpretierbarkeit

ReLU in den meisten Netzwerken

Definition Klasse

Supervised vs Unsupervised

Dimensionality reduction and Visualisation Was ist die letzte/vorletzte Schicht?

vgl. Ac

Wir verwenden die Begriffe Bild und Datenpunkt äquivalent.

<sup>1</sup><https://mpastell.com/pweave/>

<sup>2</sup><https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html>

### 2.0.1 CNNs

Idee, Abstraktion, high level, low level features, bekannte Netzwerke

Ausführliche Einführung stanford Kurs [?]. Unterschied zu FC layers: It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and CONV layers

Starting with LeNet-5 [10], convolutional neural networks (CNN) have typically had a standard structure – stacked convolutional layers (optionally followed by contrast normalization and max-pooling) are followed by one or more fully-connected layers. Variants of this basic design are prevalent in the image classification literature and have yielded the best results to-date on MNIST, CIFAR and most notably on the ImageNet classification challenge [9, 21]. For larger datasets such as ImageNet, the recent trend has been to increase the number of layers [12] and layer size [21, 14], while using dropout [7] to address the problem of overfitting. [?]

Softmax am Ende für Transformation in Probabilities.

Netzwerk im Netzwerk [?, ?]

### 2.0.2 Besondere Schichten

Promotion Sebastian Lapuschkin

- *BatchConv* besteht aus

```

1  nn.Conv2d(in_channels=in_channels, out_channels=
    out_channels, **kwargs)
2  nn.BatchNorm2d(num_features=out_channels)
3  nn.ReLU()
4
```

**Listing 1:** python-interner Aufbau einer BatchConv Schicht

in genau dieser Reihenfolge Bem.: Nur für BatchNorm2d müsste man LRP implementieren, für Conv2d funktioniert das bereits.

Batch Normalization<sup>3</sup>

### 2.0.3 Inception v3

Filter, In klassischen feed forward Netzen wird Output der vorherigen layer ist input der nächsten layer

Jetzt: Inception Block: Previous layer input, 4 operations in parallel, concatenation, 1x1 conv -> lower dimension -> less computational cost

Intermediate classifiers: kommt aus multitask learning. Eigentlich eine Möglichkeit gegen vanishing gradients

<sup>3</sup><https://arxiv.org/pdf/1502.03167.pdf>

### 2.0.4 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper Very Deep Convolutional Networks for Large-Scale Image Recognition. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple  $3 \times 3$  kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GP's.

## 2.1 Datensatz

### GTSRB<sup>4</sup>

Für die Poisoning-Angriffe auf verschiedene neuronale Netzwerke benutzen wir den Datensatz German Traffic Sign Recognition Benchmark 1. Dieser besteht aus 52.001 Bildern von Verkehrsschildern aus 43 verschiedenen Kategorien der Pixelgröße  $32 \times 32$ . Etwa 75 Prozent der Bilder wird für das Training, die anderen 25 Prozent für das Testen benutzt. Der Datensatz wurde ursprünglich in einem Wettbewerb auf der International Joint Conference on Neural Networks (IJCNN) im Jahr 2011 benutzt. Die Bilder sind aus einer Videosequenz herausgeschnitten. Deshalb befinden sich in einer Klasse jeweils immer mehrere Bilder desselben Verkehrsschildes zu unterschiedlichen Zeitpunkten. Aufnahmen desselben Verkehrsschildes kommen nicht übergreifend in Training-, Validierungs- oder Testdatensatz vor. Verkehrsschild

Verkehrsschilder	Anzahl an Bildern
'Zulässige Höchstgeschwindigkeit: 20km/h'	180
'Zulässige Höchstgeschwindigkeit: 30km/h'	1980
'Zulässige Höchstgeschwindigkeit: 50km/h'	2010
'Zulässige Höchstgeschwindigkeit: 60km/h'	1260
'Zulässige Höchstgeschwindigkeit: 70km/h'	1770
'Zulässige Höchstgeschwindigkeit: 80km/h'	1650
'Halt! Vorfahrt gewähren'	690

**Tabelle 1:** Für einen Poisoning-Angriff interessante Klassen und die zugehörige Anzahl an Bildern.

In Tabelle ?? sind einige Klassen der Verkehrsschilder und deren Anzahl im Datensatz aufgelistet, die für einen Poisoning-Angriff interessant sein könnten. Die Anzahl der Schilder 'Halt! Vorfahrt gewähren'-Schilder im Trainingssatz beträgt etwa 690 Aufnahmen. Diese wurden von insgesamt nur 24 verschiedenen 'Halt! Vorfahrt gewähren'-Schildern aufgenommen. Da beim Erstellen der korrumpierten Daten auch immer das Bild aus der angegriffenen Klasse in die Zielklasse verschoben wird, wird die Anzahl der in der Ursprungs-kategorie verbleibenden Daten abhängig vom Anteil an korrumpierten Daten kleiner. Wir werden uns deshalb im

<sup>4</sup>[https://benchmark.ini.rub.de/gtsrb\\_dataset.html](https://benchmark.ini.rub.de/gtsrb_dataset.html)



Folgenden mit Angriffen auf die Klasse 'Zulässige Höchstgeschwindigkeit: 50 km/h' beschäftigen, da sie die höchste Anzahl an Daten aufweist.

### 3 Poisoning-Angriffe

Mithilfe eines manipulierten Datensatzes wird das Netzwerk manipuliert, sodass die Entscheidung des Netzwerkes abhängig von einem Auslöser ist.

**Wer wird wie angegriffen?:** Der Angreifer erstellt einen Datensatz, sodass in den Netzwerken, die auf diesem Datensatz trainiert werden eine Hintertür implementiert wird. Damit ergibt sich die Annahme, dass der Angreifer volle Kontrolle über den Datensatz hat und somit Datenpunkte entfernen oder hinzufügen kann.

Was passiert, wenn der Angreifer keinen Zugriff auf die Modell-Architektur hat? Transfer-Learning?

**Anbringen des Auslösers**

#### 3.1 Standard Poisoning-Angriffe

Wir wollen Schilder der Klasse 50kmh absichtlich falsch als 80kmh. Wir wählen diese beiden Klassen aufgrund der Größe beider Klassen(s. Aufstellung in Praktikumsbericht). Stoppschildklasse ist wohl vergleichsweise ziemlich klein.

Dazu fügen wir auf den 50er Schildern einen Sticker ein und ändern das Label auf 80. Label-Consistent Backdoor Attacks Für die Bewertung, wie erfolgreich ein Angriff war, fügen wir in jedem Bild der 50er Klasse im Testdatensatz einen Sticker ein und messen, wie groß der Anteil der 50er Schilder ist, die als 80er Schild klassifiziert werden.

**CH- und Backdoor-Artefakte:**

In [?] wird wie folgt zwischen Clever Hans- und Backdoor-Artefakten unterschieden. In beiden Fällen wird rechts oben im Bild ein grauer 3x3 Sticker eingefügt. Bei CH geschieht dies bei 25% der airplane-Klasse. Bei Backdoor-Artefakten werden 10% aller Bilder korumpiert. Im zweiten Fall wird das entsprechende Label abgeändert. Dies entspricht dann einem Standard- bzw. Clean-Label-Poisoning-Angriff. (Wie gut funktioniert der CLPA/CH hier ohne die Bilder vorher schlechter zu machen? TODO: Vergleich mit [?]). In [?], Kapitel 2.1 wird auch auf die Methode der Spektralen Signatur [?] eingegangen, die zur Detektion genutzt wird. Diese eignet sich wohl sehr gut für die Backdoor-Attacks, aber nur schlecht für die CH-Artefakte.

#### 3.2 Label-konsistente Poisoning-Angriffe

Bei den vorherigen Standard-Angriffen war es der Fall, dass das Label und das entsprechende Bild nicht mehr zusammenpassen. Ein händisches Durchsuchen des Datensatz (wenn auch sehr aufwendig) könnte damit ebenfalls zur Detektion eines Angriffs führen.

Eine deutlich schwieriger zu detektierende Art von Poisoning-Angriffen sind sogenannte Label-konsistente Angriffe, bei denen genau diese Schwachstelle eliminiert ist, d.h. Label und Bild passen wieder zueinander, während der Angriff noch immer erfolgreich funktioniert. Es ist das Ziel, ein Bild zunächst so zu modifizieren, dass es für das menschliche Auge noch immer zur entsprechenden Klasse gehört, für das

Neuronale Netzwerk aber so schwierig zu klassifizieren ist, dass sich das Netzwerk eher auf den Auslöser anstatt auf das ursprüngliche Bild verlässt. Im Anschluss wird wieder ein Auslöser eingefügt.

In [?] werden zwei Verfahren vorgestellt, die die Klassifikation einzelner Bilder erschweren. Das erste Verfahren besteht aus einer Einbettung in einen niedrig-dimensionalen Raum, das auch bei Autoencodern, etc. verwendet wird TODO.

Beim zweiten Verfahren wird ein sogenannte Projizierter Gradienten-Abstieg-Angriff durchgeführt.

Dabei wird ein Adversarialer Angriff in leicht abgewandelter Form genutzt, um das Netzwerk zu stören. Bei Adversarialen Angriffen wird ein Netzwerk im Unterschied zum Poisoning-Angriff, bei dem der Angriff während des Trainings stattfindet, nach dem Training angegriffen. Dazu wird eine natürliche Netzwerkeingabe leicht gestört, sodass diese vom Netzwerk falsch klassifiziert wird. Diese Störungen lassen sich auch von einer Architektur oder sogar einem Modell auf andere übertragen [?, ?]. Für diese Art von Angriff werden die adversarialen Angriffe und ihre leichte Übertragbarkeit auf andere Architekturen und Modelle so benutzt, dass es bereits während des Trainings zu falschen Klassifikationen kommt.

Für unser erstes trainiertes Netzwerk  $f_\theta$  mit Verlustfunktion  $\mathcal{L}$  und einem Eingabe-Paar  $(x, y)$ , konstruieren wir die modifizierte Version von  $x$  als

$$x_{adv} = \arg \max_{||x' - x||_p \leq \varepsilon} \mathcal{L}(x', y, \theta), \quad (3.1)$$

für  $p > 1$  und  $\varepsilon > 0$ . Dieses Optimierungsproblem wird mit einem Projizierten Gradienten-Verfahren [?]. Details dazu finden sich in Anhang D. Im Unterschied zu [?] ändern wir nur Bilder im Datensatz ab und fügen nicht zusätzlich zum Original  $x$  auch  $x_{adv}$  hinzu. Damit ändert sich die Anzahl an Datenpunkten durch den Angriff nicht.

Für das anschließende Einfügen des Auslöser ergeben sich die folgenden Optionen:

- Der im Standard-Angriff verwendete Sticker
- Ein Amplitudensticker: Dabei wird im rechten unteren Eck des Bildes ein
- Amplitudensticker 4 fach
- In die Mitte verschobene Amplitudensticker

RGB auf jedem Kanal in der range von [0,255] 0 entspricht weiß, 255=schwarz Da die zweite Möglichkeit als deutlich erfolgreicher angegeben wird, beschränken wir uns auf diese Angriffe basierend auf einem Projizierten Gradienten-Abstieg. amp=16,32,64,255 Wir gehen zunächst davon aus, dass der Angreifer volle Kontrolle über den Datensatz und das Netzwerk besitzt.TODO: Angriff mit einem andere Netzwerk erstellen, als das angegriffene

### 3.3 Bewertung von Poisoning-Angriffen

Wann ist ein Angriff erfolgreich?

Im Trainingsdatensatz werden im Fall des Standard-Angriffs alle Bilder mit dem Sticker versehen. Die Angriffserfolgsrate beschreibt nun den Anteil an Bildern der

attackierten Klasse, die erfolgreich falsch klassifiziert wurden.

Für die Label-konsistenten Angriffe werden im Test-Datensatz alle Bilder mit dem entsprechenden Auslöser versehen und es kann eine Erfolgsrate pro Klasse berechnet werden. Es ist zu beachten, dass für Angriffe, die mit einer reduzierten Amplitudenstärke durchgeführt werden, die Bilder im Test-Datensatz dennoch mit Auslösern mit voller Amplitudenstärke versehen werden.

### 3.4 Verteidigungen

In diesem Kapitel beschäftigen wir uns mit gängigen Methoden zur Detektion von Poisoning-Attacks und geben am Ende einen kurzen Ausblick auf die Idee für einen neuen Ansatz. Wir wollen beide Arten von Poisoning-Angriffen erfolgreich detektieren.

#### 3.4.1 Referenzwert: kMeans(k=2)

Der einfachste Ansatz, um korruptierte Datenpunkte zu erkennen, ist ein kMeans-Clustering, das direkt(bzw. nach einer Dimensionsreduktion) auf den Eingabedaten einer Klasse durchgeführt wird. Hierbei war auffällig, dass der Großteil der Daten als korruptiert klassifiziert wurde. Für die Dimensionsreduktionen FastICA und PCA ergab sich eine Genauigkeit von etwa 66%. Die FPR lag bei über 70 %, die TPR bei ca. 50%. Dieser Referenzwert w@articleszegedy2013intriguing, title=Intriguing properties of neural networks, author=Szegedy, Christian and Zaremba, Wojciech and Sutskever, Ilya and Bruna, Joan and Erhan, Dumitru and Goodfellow, Ian and Fergus, Rob, journal=arXiv preprint arXiv:1312.6199, year=2013 wurde für einen Sticker mit Seitenlänge 3 und 15% korruptierten Daten durchgeführt.

#### 3.4.2 Activation Clustering

Nehme Datensatz her Beim Standardangriff wollen wir Klasse 5 als Klasse 8 klassifizieren und fügen dazu Sticker der

Diese Idee der Verteidigung basiert auf der Annahme, dass bestimmte Schichten innerhalb des Netzwerkes die Entscheidung, dass ein Bild mit einem Auslöser falsch klassifiziert wird, sehr gut codieren. Für die Detektion der Hintertüren im Datensatz sollen nun genau diese Aktivierungen für ein Clustering herangezogen werden. Das Activation Clustering wird erstmalig in [?] vorgestellt und nutzt aufgrund experimenteller Untersuchungen stets die Aktivierungen der vorletzten Netzwerkschicht. Eine Kombination von Aktivierungen mehrere Schichten wäre ebenfalls denkbar.

Ein Angriff ist erfolgreich, wenn eine große Anzahl an Datenpunkten der Ursprungsklasse, versehen mit einem Auslöser, der Zielklasse zugeordnet werden. Im Falle eines erfolgreichen Angriffs werden korruptierte und nicht korruptierte Datenpunkte im Trainingsdatensatz derselben Klasse zugeordnet. Der Grund weshalb diese derselben Klasse zugeordnet werden, unterscheidet sich jedoch. Beim Activation Clustering wird nun angenommen, dass pro Klasse entweder korruptierte und nicht korruptierte Datenpunkte oder nur nicht korruptierte Datenpunkte existieren. Deshalb werden die Aktivierungen der letzten verdeckten Schicht des Netzwerkes aus dem Netz extrahiert, nach ihren zugehörigen Klassen der Labels segmentiert, auf 10 Dimensionen reduziert und anschließend mit Hilfe des kMeans-Algorithmus geclustert. Das kleinere Cluster wird immer als der Anteil an verdächti-

gen Datenpunkten betrachtet. Die Idee ist es, dass die korruptierten Datenpunkte, sofern welche existieren, alle in die eine und die nicht korruptierten Datenpunkte in das andere Cluster aufgeteilt werden. Sind keine korruptierten Datenpunkte vorhanden, so sollen beide Cluster ungefähr dieselbe Anzahl an Datenpunkten erhalten.

Wir werten die Qualität des Clusterings anschließend aus. Als Detektionsrate beschreiben wir die Genauigkeit des Clusterings auf den Trainingsdaten.

Im folgenden Abschnitt werden Methoden vorgestellt, mit denen das resultierende Clustering auf die Präsenz eines Angriffs untersucht werden kann. Dies ist notwendig, da in der Praxis ein Angriff zunächst erkannt und anschließend die korruptierten Datenpunkte entfernt werden müssen.

**Bemerkung 3.4.1.** *Das SPA benutzt die Idee, dass innerhalb einer Klasse bezüglich unterschiedlicher Aktivierungen klassifiziert werden kann. Beim CLPA funktioniert das nicht mehr, denn: Hier passen jetzt auch die Aktivierungen der korruptierten Bilder zur entsprechenden/untersuchten Klasse. Es ist also zu erwarten, dass das AC für CLPA nicht funktioniert.*

## 3.5 Implementierung

Wir nutzen die Python-Implementierung in sklearn mit den Standard-Werten  $n\_init = 10$  und  $max\_iter = 300$

### 3.5.1 Methoden zur Untersuchung, ob ein Angriff vorliegt

#TODO: Vergleich mit Fisher-Discriminant-Analyse/Ansatz in [?] Zur Bestimmung, ob eine Klasse korruptierte Daten enthält, kann das Ergebnis des Clusterings mit den folgenden Methoden untersucht werden:

@articleszegedy2013intriguing, title=Intriguing properties of neural networks, author=Szegedy, Christian and Zaremba, Wojciech and Sutskever, Ilya and Bruna, Joan and Erhan, Dumitru and Goodfellow, Ian and Fergus, Rob, journal=arXiv preprint arXiv:1312.6199, year=2013 Vergleich der relativen Größe: Eine Möglichkeit, korruptierte Datenpunkte zu erkennen, ist der Vergleich der relativen Größen der beiden Cluster. Laut [2] ist die relative Größe bei nicht korruptierten Klassen ca. 50 Prozent, bei korruptierten Daten und einem erfolgreichen Clustering würde die relative Größe dann dem prozentualen Anteil an korruptierten Datenpunkten entsprechen.

**Silhouette-Koeffizient:** Eine weitere Möglichkeit besteht darin, die Qualität des Clusterings mit Hilfe des Silhouette-Koeffizienten zu beschreiben. Dieser gibt an, wie gut ein Clustering zu den gegebenen Datenpunkten mit den entsprechenden Labels passt und ist wie folgt definiert: Sei das Ergebnis eines Clustering-Algorithmus mit verschiedenen Clustern gegeben. Zu einer Beobachtung  $x$  im Cluster  $A$  wird die Silhouette  $s(x) = \frac{d(B,x) - d(A,x)}{\max\{d(A,x), d(B,x)\}}$  definiert, wobei  $d(A,x) = \frac{1}{n_A - 1} \sum_{a \in A, a \neq x} d(a,x)$  dem mittleren Abstand einer Beobachtung innerhalb einer Klasse zu allen anderen Beobachtungen dieser Klasse entspricht. Dabei steht  $n_A$  für die Anzahl der Beobachtungen in Cluster  $A$ .  $d(B,x) = \min_{C \neq A} d(C,x)$  beschreibt die Distanz von  $x$

zum nächstgelegenen Cluster B. Der Silhouetten-Koeffizient  $SC$  ist nun definiert als

$$SC = \max_k \tilde{s}(k), \quad (3.2)$$

wobei  $\tilde{s}(k)$  der Mittelwert der Silhouetten aller Datenpunkte im gesamten Datensatz ist. Damit ist der Silhouettenkoeffizient ein Maß dafür, wie gut ein Clustering für eine vorher fixierte Clusteranzahl  $k$  zum Datensatz passt.

**Exklusives Retraining:** Beim exklusiven Retraining wird das neuronale Netz von Grund auf neu trainiert. Das oder die verdächtigen Cluster werden beim erneuten Training nicht benutzt. Mit Hilfe des neu trainierten Netzes werden dann anschließend die vorenthaltenen, verdächtigen Cluster klassifiziert. Falls das Cluster Aktivierungen von Datenpunkten enthält, die zum Label des Datenpunktes gehören, erwarten wir, dass die Vorhersage des Netzwerks mit dem Label übereinstimmen. Gehören die Aktivierungen eines Datenpunktes im verdächtigen Cluster jedoch zu einer anderen Klasse als die durch das Label angedeutete Klasse, so sollte das Netzwerk den Datenpunkt einer anderen Klasse zuordnen. Um nun zu entscheiden, ob ein verdächtiges Cluster korruptiert oder nicht korruptiert ist, wird wie folgt vorgegangen: Sei  $l$  die Anzahl an Vorhersagen, die zum Label des Datenpunktes passen. Sei  $p$  die größte Anzahl an Vorhersagen, die für eine weitere Klasse  $C$  sprechen, wobei  $C$  nicht die Klasse mit den Labels des zu untersuchenden Clusters ist. Der Quotient  $\frac{l}{p}$  gibt dann an, ob das Cluster korruptiert ist oder nicht: Es wird ein Schwellenwert  $T > 0$  gesetzt. Gilt  $\frac{l}{p} < T$ , wurden mehr Datenpunkte einer anderen Klasse zugeordnet und das Cluster wird als korruptiert deklariert. Umgekehrt wird das verdächtige Cluster im Fall von  $\frac{l}{p} > T$  als nicht korruptiert/sauber eingestuft. @articleszegedy2013intriguing, title=Intriguing properties of neural networks, author=Szegedy, Christian and Zaremba, Wojciech and Sutskever, Ilya and Bruna, Joan and Erhan, Dumitru and Goodfellow, Ian and Fergus, Rob, journal=arXiv preprint arXiv:1312.6199, year=2013

### 3.5.2 Entfernen von korruptierten Datenpunkten

**AC für Label-konistente Poisoning-Angriffe:** Warum funktioniert Activation-Clustering hier nur schlecht oder gar nicht?: Wenn wir einen korruptierten Trainingsdatensatz gegeben haben, gilt im Fall des Standard-Angriffs folgender Sachverhalt: Die angegriffene Klasse, die Klasse in der samples eingefügt wurden, besitzt die eine Gruppe an Bildern, die zu einer Aktivierung von einer anderen Klasse führen sollten, und die Gruppe an Bildern, die zu dieser Klasse gehören und zur Aktivierung genau dieser Klasse führen sollte.

Im Fall des Label-konsistenten Poisoning-Angriffs, werden nun keine Label mehr getauscht, d.h. Bilder von der einen in die andere Klasse verschoben. Damit können die beiden Gruppen (korruptiert, sauber) innerhalb einer Klasse nicht mehr anhand ihrer Aktivierungen unterschieden werden.

Trotzdem ergibt sich ein Ansatz daraus, dass es innerhalb dieser Klasse verschiedene „Strategien“ gibt, die zur selben Klassifikation führen. Mithilfe des kmeans-Clustering basierend auf den Heatmaps sollen genau diese Strategien ausfindig gemacht werden, um die Bilder in korruptiert und sauber zu unterteilen.

### 3.5.3 Heatmap Clustering

Im Unterschied zum Activation Clustering, bei dem die Aktivierungen der vorletzten Netzwerk-Schicht verwendet werden, ist nun hier die Idee, zu jedem Eingabebild eine Relevanzkarte zu erstellen, die für jeden Pixelpunkt angibt, wie wichtig dieser für die Klassifikation dieses Bildes ist.

Für das Erstellen/Berechnen solcher Relevanzkarten/Heatmaps existieren mehrere Methoden, die zusammengefasst dem Bereich der Erklärbaren KI zugeordnet werden. Im folgenden Kaptitel wollen wir einen kurzen Überblick über verschiedene Methoden geben.

## 4 Erklärbare KI

Erklärbarkeit vs. Interpretierbarkeit, youtube talk?!

### 4.1 Lokale Methoden

### 4.2 Globale Methoden

## 5 Layer-wise Relevance Propagation

### 5.1 Idee

Die Layer-wise Relevance Propagation (LRP) wird in [?] erstmalig vorgestellt. Die Idee besteht darin, einen Zusammenhang zwischen der Ausgabe eines Klassifikators  $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^+$  und der Eingabe  $x$  herzustellen. Dabei wird eine definiert, die über gewisse Eigenschaften eingeschränkt wird. Die Autoren bezeichnen die Herangehensweise hier selbst als heuristisch und liefern in ?? eine Verallgemeinerung des Konzepts, die gleichzeitig die mathematische Grundlage bildet.

Wir betrachten eine nicht-negative Funktion  $f : \mathbb{R}^d \rightarrow \mathbb{R}^+$ . Im Bereich der Bild-Klassifizierung ist die Eingabe  $x \in \mathbb{R}^d$  ein Bild, das wir als Menge von Pixelwerten  $x = \{x_p\}$  auffassen können. Dabei beschreibt der Index  $p$  einen genauen Pixelpunkt. Während für schwarz-weiß Bilder  $x_p \in \mathbb{R}$  gilt, gilt im Fall von RGB-Bildern  $x_p \in \mathbb{R}^3$  für die einzelnen Farbkanaäle Rot, Grün und Blau. Die Funktion  $f(x)$  ist ein Maß dafür, wie präsent ein oder mehrere Objekte in der Eingabe/im Eingabebild vorhanden sind. Ein Funktionswert  $f(x) = 0$  beschreibt die Abwesenheit. Gilt andererseits  $f(x) > 0$ , so wird die Präsenz mit einem gewissen Grad an Sicherheit oder eine gewisse Menge zum Ausdruck gebracht.

Mit Hilfe der LRP soll nun jedem Pixel  $p$  im Eingabebild eine Relevanz  $R_p(x)$  zugeordnet werden, die für jedes Pixel  $x_p$  angibt, mit welcher Größe es für das Entstehen einer Entscheidung  $f(x)$  verantwortlich ist. Die Relevanz eines jeden Pixels wird dabei in einer Heatmap  $R(x) = \{R_p(x)\}$  zusammengefasst.

Die Heatmap besitzt dieselbe Größe wie  $x$  und kann als Bild visualisiert werden. Wir definieren die folgenden Eigenschaften:

**Definition 5.1.1.** Eine Heatmap  $R(x)$  heißt konservativ, falls gilt:

$$\forall x : f(x) = \sum_p R_p(x), \quad (5.1)$$

d.h. die Summe der im Pixelraum zugeordneten Relevanz entspricht der durch das Modell erkannten Relevanz.

**Definition 5.1.2.** Eine Heatmap  $R(x)$  heißt positiv, falls gilt:

$$\forall x, p : R_p(x) \geq 0, \quad (5.2)$$

d.h. alle einzelnen Relevanzen einer Heatmap sind nicht-negativ.

Die erste Eigenschaft verlangt, dass die umverteilte Gesamtrelevanz der Relevanz entspricht, mit der ein Objekt im Eingabebild durch die Funktion  $f(x)$  erkannt wurde. Die zweite Eigenschaft beschreibt, dass keine zwei Pixel eine gegensätzliche Aussage über die Existenz eines Objektes treffen können. Beide Definitionen zusammen ergeben die Definition einer *konsistenten* Heatmap:

**Definition 5.1.3.** Eine Heatmap  $R(x)$  heißt konsistent, falls sie konservativ und positiv ist, d.h. Definition 5.1.1 und Definition 5.1.2 gelten.

Für eine konsistente Heatmap gilt dann  $(f(x) = 0 \Rightarrow R(x) = 0)$ , d.h. die Abwesenheit eines Objektes hat zwangsläufig auch die Abwesenheit jeglicher Relevanz in der Eingabe zur Folge, eine Kompensation durch positive und negative Relevanzen ist folglich nicht möglich.

**Bemerkung 5.1.4.** Die geforderten Eigenschaften an eine Heatmap definieren diese nicht eindeutig. Es sind also mehrere Abbildungen möglich, die die genannten Forderungen erfüllen. Beispiele dafür sind eine natürliche Zerlegung und Taylor-Zerlegungen [?].

Die LRP liefert nun ein Konzept, mit dem eine Zerlegung

$$f(x) = \sum_d R_d \quad (5.3)$$

bestimmt werden kann.

TODO: Summenabfolge von layer zu layer einfügen

Wir gehen nun davon aus, dass die Funktion  $f$  ein NN repräsentiert, dass aus mehreren Schichten mit mehreren Neuronen pro Schicht und dazwischengeschalteten nicht-linearen Aktivierungsfunktionen aufgebaut ist. Die erste Schicht ist die Eingabe-Schicht, bestehend aus den Pixeln eines Bildes. Die letzte Schicht ist die reellwertige Ausgabe von  $f$ . Die  $l$ -te Schicht ist durch einen Vektor  $z = (z_d^l)_{d=1}^{V(l)}$  der Dimension  $V(l)$  dargestellt. Sei also eine Relevanz  $R_d(l+1)$  für jede Dimension  $z_d^{(l+1)}$  des Vektors  $z$  in der Schicht  $l+1$  gegeben. Die Idee besteht nun darin, eine Relevanz  $R_d^{(l)}$  für jede Dimension  $z_d^{(l)}$  des Vektors  $z$  in der Schicht  $l$  zu finden, die einen Schritt näher an der Eingabeschicht liegt, sodass die folgende Abfolge von Gleichungen gilt:

$$f(x) = \dots = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l} R_d^{(l)} = \dots = \sum_d R_d^{(1)}. \quad (5.4)$$

Für diese Funktion benötigen wir eine Regel, mit der die Relevanz eines Neurons einer höheren Schicht  $R_j^{(l+1)}$  auf ein Neuron einer benachbarten, näher an der Eingabeschicht liegendes Neuron, übertragen werden kann. Die Übertragung der Relevanz zwischen zwei solchen Neuronen wird mit  $R_{i \leftarrow j}$  bezeichnet. Auch hier muss die übertragene Relevanz erhalten bleiben. Es wird also gefordert:

$$\sum_i R_{i \leftarrow j}^{(l,l+1)} = R_j^{(l+1)}. \quad (5.5)$$

D.h. die gesamte Relevanz eines Neurons der Schicht  $l+1$  verteilt sich komplett auf alle Neuronen der Schicht  $l$ . Im Falle eines linearen NN  $f(x) = \sum_i z_{ij}$  mit der Relevanz  $R_j = f(x)$  ist eine Zerlegung gegeben durch  $R_{i \leftarrow j} = z_{ij}$ . Im allgemeineren Fall ist die Neuronenaktivierung  $x_j$  eine nicht-lineare Funktion abhängig von  $z_j$ . Für die beiden Aktivierungsfunktionen  $\tanh(x)$  und  $\text{ReLU}(x)$  - beide monoton wachsend mit  $g(0) = 0$  - bieten die Vor-Aktivierungen noch immer ein sinnvolles Maß für den relativen Beitrag eines Neurons  $x_i$  zu  $R_j$  (müsste das nicht umgekehrt sein, die INdizes?!?!).

Eine erste Mögliche Relevanz-Zerlegung, basierend auf dem Verhältnis zwischen lokalen und globalen Vor-Aktivierung, ist gegeben durch:

$$R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j} \cdot R_j^{(l+1)}. \quad (5.6)$$

Für diese Relevanzen  $R_{i \leftarrow j}$  gilt die Erhaltungseigenschaft 5.4, denn:

$$\sum_i R_{i \leftarrow j}^{(l,l+1)} = R_j^{l+1} \cdot \left(1 - \frac{b_j}{z_j}\right). \quad (5.7)$$

Dabei steht der rechte Faktor für die Relevanz, die durch den Bias-Term absorbiert wird. Falls notwendig, kann die verbleibende Bias-relevanz auf jedes Neuron  $x_i$  verteilt werden(?s.Abschnitt über Biases Promotion, S.Lapuschkin).

Diese Regel wird in der Liteartur als LRP-0 bezeichnet. Ein Nachteil dieser ist, dass die Relevanzen  $R_{i \leftarrow j}$  für kleine globale Voraktivierung  $z_j$  beliebig große Werte annehmen können.

Um dies zu verhindern, wird in der LRP- $\varepsilon$ -Regel ein vorher festgelegter Parameter  $\varepsilon > 0$  eingeführt:

$$R_{i \leftarrow j}^{(l,l+1)} = \begin{cases} \frac{z_{ij}}{z_j + \varepsilon} \cdot R_j^{(l+1)}, & z_j \geq 0 \\ \frac{z_{ij}}{z_j - \varepsilon} \cdot R_j^{(l+1)}, & z_j < 0 \end{cases} \quad (5.8)$$

In [?] wird die Layer-wise Relevance Propagation erstmalig vorgestellt. Zudem wird eine Taylor Zerlegung präsentiert, die eine Approximation der LRP darstellt.

Hier<sup>5</sup> werden einige Bereiche vorgestellt, in denen LRP angewendet wurde.

## 5.2 Behandlung von biases

## 5.3 Beispiel an einem kleinen Netzwerk

## 5.4 Deep Taylor Decomposition

Mathematischer Hintergrund für LRP.LRP als Spezialfall von DTD

<sup>5</sup><https://towardsdatascience.com/indepth-layer-wise-relevance-propagation-340f95deb1ea>



### 5.4.1 Taylor Decomposition

We will assume that the function  $f(x)$  is implemented by a deep neural network, composed of multiple layers of representation, where each layer is composed of a set of neurons. Each neuron performs on its input an elementary computation consisting of a linear projection followed by a nonlinear activation function. Deep neural networks derive their high representational power from the interconnection of a large number of these neurons, each of them, realizing a small distinct subfunction.

Laut [?] ist die in [?] vorgestellte Layer-wise Relevance Propagation eher heuristisch. In diesem Paper wird nun eine solide theoretische Grundlage geliefert.

DTD liefert den mathematischen Hintergrund für LRP

Simple Taylor decomposition. Finde rootpoints, sodass Erhaltungseigenschaft erhalten bleibt.

Simple Taylor in Practice: funktioniert in der Praxis nicht wirklich. Viel Noise meistens positive Relevanz

Relevanz Propagation: Heatmaps look much cleaner

Simple Taylor:- root point hard to find -gradient shattering. Gradient loses its informative structure in big layer nets

Use Taylor Decomposition to explain LRP from layer to layer

### 5.4.2 Deep Taylor Decomposition

LRP in verschiedenen Anwendungsgebieten [?], 10.2. In diesem Paper: LRP-0 schlechter als LRP- $\epsilon$  schlechter als LRP- $\gamma$  schlechter als Composite-LRP.

## 5.5 Verschiedene Verfahren

### 5.6 Eigenschaften

Beweise in DTD Paper

- Numerische Stabilität
- Konsistenz (mit Linearer Abbildung)
- Erhaltung der Relevanz

### 5.7 Behandlung besonderer Schichten

#### 5.7.1 BatchNorm2D

### 5.8 LRP für Deep Neural Nets/Composite LRP

### 5.9 Verarbeitung der Heatmaps

Aktuell benutzte default colormap ist Option D (Viridis)<sup>6</sup>

- Wertebereich
- Interpretation
- Skalen

---

<sup>6</sup><https://bids.github.io/colormap/>



**Abbildung 1:** (Optischer) Vergleich von korrumpiertem Datenpunkt und berechneter Heatmap. Links: Verkehrsschild der Klasse 'Höchstgeschwindigkeit: 50km/h' versehen mit einem 3x3 Sticker und dem Label 'Höchstgeschwindigkeit: 80km/h'. Rechts: Zugehörige Heatmap bezüglich der Klasse 'Höchstgeschwindigkeit: 80km/h'.

Quelle: [?]

- Normalisierung

## 5.10 Implementierungen

### 5.10.1 Tensorflow

### 5.10.2 pytorch

**Allgemeines Tutorial:**<sup>7</sup>

pytorch-LRP für VGG16 wird vorgestellt.

**GiorgioML**<sup>8</sup>:

Alternative pytorch-Implementierung basierend auf Tensorflow paper.

**moboehle**<sup>9</sup>:

Der code entstand im Rahmen der Forschungsarbeit [?], in der eine Alzheimer-Feststellung aufgrund von Bilddaten(scans?) vorgenommen wird. Framework leicht anpassbar. Benutzt pytorch hooks. Unterstützte Netzwerkschichten<sup>10</sup>:

```
1 torch.nn.BatchNorm1d,
2 torch.nn.BatchNorm2d,
3 torch.nn.BatchNorm3d,
4 torch.nn.ReLU,
5 torch.nn.ELU,
6 Flatten,
7 torch.nn.Dropout,
8 torch.nn.Dropout2d,
9 torch.nn.Dropout3d,
```

<sup>7</sup><https://git.tu-berlin.de/gmontavon/lrp-tutorial>

<sup>8</sup><https://giorgiomorales.github.io/Layer-wise-Relevance-Propagation-in-Pytorch/>

<sup>9</sup><https://github.com/moboehle/Pytorch-LRP>

<sup>10</sup>[https://github.com/moboehle/Pytorch-LRP/blob/master/inverter\\_util.py](https://github.com/moboehle/Pytorch-LRP/blob/master/inverter_util.py)

```

10 torch.nn.Softmax,
11 torch.nn.LogSoftmax,
12 torch.nn.Sigmoid
13
14

```

**Listing 2:** Verfügbare Schichten und Aktivierungsfunktionen**fhvilshoj**<sup>11</sup>:

LRP für linear und Convolutional layers

- Die Klassen `torch.nn.Sequential`, `torch.nn.Linear` und `torch.nn.Conv2d` werden erweitert, um autograd für die Berechnung der Relevanzen zu berechnen.
- Ausgabe der Relevanzen von Zwischenschichten ist möglich
- : Implementierte Regeln: epsilon Regeln mit `epsilon=1e-1`, gamma-regel mit `gamma=1e-1`. alphabeta-Reagel mit `a1b0` und `a2b1`
- Netz muss hier umgeschrieben werden, sodass die Anwendung des Algorithmus möglich wird.

```

1 conv2d = {
2   "gradient":          F.conv2d,
3   "epsilon":           Conv2DEpsilon.apply,
4   "gamma":             Conv2DGamma.apply,
5   "gamma+epsilon":     Conv2DGammaEpsilon.apply,
6   "alpha1beta0":       Conv2DAlpha1Beta0.apply,
7   "alpha2beta1":       Conv2DAlpha2Beta1.apply,
8   "patternattribution": Conv2DPatternAttribution.apply,
9   "patternnet":        Conv2DPatternNet.apply,
10  }
11
12
13

```

**Listing 3:** Implementierte Regeln fhvilshoj**Zennit**:<sup>12</sup> Zennit (Zennit explains neural networks in torch)

- Modell wird mithilfe eines Canonizers so aufbereitet, dass LRP möglich wird
- Backward pass wird modifiziert, um Heatmaps zu erhalten.
- VGG- und ResNet-Beispiel

<sup>11</sup><https://github.com/fhvilshoj/TorchLRP><sup>12</sup><https://github.com/chr5tphr/zennit>

## 6 Detektion von Poisoning-Angriffen basierend auf LRP

Super Einführung in Wasserstein und OT: [?]

Computing distances between probability measures on metric spaces, or more generally between point clouds, plays an increasingly preponderant role in machine learning [SL11, MJ15, LG15, JSCG16, ACB17], statistics [FCCR16, PZ16, SR04, BGKL17] and computer vision [RTG00, BvdPPH11, SdGP+15]. A prominent example of such distances is the earth mover's distance introduced in [WPR85] (see also [RTG00]), which is a special case of Wasserstein distance, or optimal transport (OT) distance [Vil09]. While OT distances exhibit a unique ability to capture geometric features of the objects at hand, they suffer from a heavy computational cost that had been prohibitive in large scale applications until the recent introduction to the machine learning community of Sinkhorn Distances by Cuturi [Cut13]. Combined with other numerical tricks, these recent advances have enabled the treatment of large point clouds in computer graphics such as triangle meshes [SdGP+15] and high-resolution neuroimaging data [GPC15]. Sinkhorn Distances rely on the idea of entropic penalization, which has been implemented in similar problems at least since Schrödinger [Sch31, Leo14]. This powerful idea has been successfully applied to a variety of contexts not only as a statistical tool for model

How Well Do WGANs Estimate the Wasserstein Metric? [?]

### 6.1 Idee

Die Idee zur Detektion von Poisoning-Angriffen besteht aus den folgenden Schritten:

- Berechnung der Heatmaps mit Hilfe der LRP
- Berechnung einer Distanzmatrix basierend auf  $L^2$ - oder GMW-Distanz
- Spektrale Relevanzanalyse (Bestimmung der verschiedenen Cluster innerhalb einer Klasse)

**Bemerkung:** Anstatt das Clustering nur auf den Heatmaps durchzuführen, könnten die LRP-Ausgaben und/oder Aktivierungen bestimmter Netzwerkschichten hinzugenommen werden.

The theory of optimal transport generalizes that intuition in the case where, instead of moving only one item at a time, one is concerned with the problem of moving simultaneously several items (or a continuous distribution thereof) from one configuration onto another. [?]

### 6.2 k-means / k-means++ -Clustering

Beispiel-Implementierung<sup>13</sup>

Baryzentrische Koordinaten<sup>14</sup>

<sup>13</sup><https://towardsdatascience.com/k-means-implementation-in-python-and-spark-856e7eb5fe9b>

<sup>14</sup>[https://de.wikipedia.org/wiki/Baryzentrische\\_Koordinaten](https://de.wikipedia.org/wiki/Baryzentrische_Koordinaten)

### 6.3 Spektrales Clustering

Wir folgen [?]. Gegeben: Datenpunkte  $x_1, \dots, x_n$  sowie eine Größe  $s = s_{ij} \in \mathbb{R}^+$ , die einen paarweisen Zusammenhang der einzelnen Punkte beschreiben.

Ziel: Aufteilen der Punkte in verschiedene Cluster, sodass sich Punkte innerhalb eines Clusters ähnlich bezüglich  $s$  sind.

Alternative Repräsentation der Daten mithilfe eines Ähnlichkeitsgraphen  $G = (V, E)$  möglich.

Umformulierung des Clustering-Problems mithilfe des Ähnlichkeitsgraphen: Finde Partitionierung des Graphen, sodass die Kanten-Gewichte innerhalb einer Gruppe niedrig (niedriges Gesamtgewicht?) und außerhalb einer Gruppe groß sind.

Graph-Notationen:

Verschiedene Konstruktionsmöglichkeiten von Ähnlichkeitsgraphen:

- $\varepsilon$ -Nachbarschaft-Graph
- kNN-Graph
- fully connected graph

### 6.4 Anwendung auf unterschiedliche Poisoning-Angriffe

**Berechnung der Relevanzen:**

Wir berechnen die Relevanzen jedes einzelnen Eingabebildes klassenweise, d.h. besitzt eine Eingabe das Label  $y$ , so berechnen auf einem trainierten Netzwerk, für jeden Pixelwert der Eingabe, wie relevant dieser für die Ausgabe  $f(x) = y$  ist.

Wir summieren über die Farboxen des Bildes, um einzelne Relevanzen pro Pixelpunkt zu erhalten.

Für die Berechnung der Relevanzen benutzen wir eine modifizierte Version des im Rahmen von [?] entstandenen Programmcodes<sup>15</sup>.

**Vorverarbeitung der Relevanzen:**

In [?] wird anschließend ein Sum-Pooling auf die Relevanzen angewendet, um eine Dimensionsreduktion zu erhalten. Wie in [?] verzichten wir auf eine weitere Dimensionsreduktion, da wir nur relativ kleine Relevanzen der Größe  $32 \times 32$  verarbeiten.

Für  $\text{eps}=5e-2$  liegen beide barycentren identisch weit weg. Probiere nun  $\text{eps}=5e-3$

**Berechnung der Distanzen und Aufstellen einer Affinitätsmatrix:**

Wir berechnen zunächst eine Distanzmatrix, die die paarweisen Distanzen aller Heatmaps einer Klasse enthält.

<sup>15</sup><https://github.com/moboehle/Pytorch-LRP>

Für die Berechnung der euklidischen Distanz betrachten wir Heatmaps  $x, y$  der Größe  $32 \times 32$  als Elemente  $x, y \in \mathbb{R}^{32 \times 32}$ . Die Distanz lässt sich dann wie in ?? berechnen.

Die Gromov-Wasserstein-Distanz lässt sich wie in [?] angegeben berechnen.

Barycenters Definition und Vergleich zum euklidischen Raum [?]  
In einer Affinitätsmatrix oder Ähnlichkeitsmatrix sind die

### Berechnung Spektralen Einbettung:

### Dimensionsreduktion vor dem Clustering ?!

In [?] wird beispielsweise eine Dimensionsreduktion mit PCA durchgeführt.  
**k-Means-Clustering:**

**Bemerkung 6.4.1.** In [?] Kapitel '2.3. Fisher Discriminant Analysis for Clever Hans identification' wird ein Verfahren vorgestellt, mit dem verdächtige Klassen identifiziert werden können. Für diese würde man anschließend das obige Verfahren durchführen

## 6.5 Verwendete Distanzen & Approximationen

Um die Struktur innerhalb einer Klasse zu analysieren, benötigen wir eine Metrik. Anhand dieser wird abhängig von den Heatmaps einer Klasse eine Affinitätsmatrix berechnet, die dann anschließend zur Berechnung der Spektralen Einbettung als wichtigster Schritt von SpRAy verwendet wird. Wir wollen dazu die im Folgenden vorgestellten Metriken verwenden.

Wie in [?] summieren wir über die Farbkanäle, um einen einzelnen Relevanzwert pro Pixelpunkt zu erhalten. Wir benötigen also eine Metrik zur Berechnung der Distanz zwischen  $32 \times 32$  großen Heatmaps.

Wir normalisieren die Relevanzen zusätzlich auf das Intervall  $[0, 1]$ .

Die Wahl der Pixel mit 99 Prozent der Gesamtmasse und anschließende Normalisierung wird vermutlich durchgeführt, um die Bedingung Gleichung 7.1 zu erhalten.

## 7 Optimal Transport

Optimal Transport is an interesting topic which connects many fields and has interesting applications. Applications in image analysis: Interplay between geometry, probability and PDEs. Convexity, duality. Numerical optimization. Statistics and Machine Learning<sup>16</sup>

Optimal transport can deal with smooth and discrete measures and it has proved to be very useful for comparing distributions in a shared space, but with different (and even non-overlapping) supports [?]. Einführung OT Villani. OT hebt Distanzen von metrischen Räumen auf den Raum der metrischen Räume [?].

<sup>16</sup>[https://indico.cern.ch/event/845380/attachments/1915103/3241592/Dvurechensky\\_lectures.pdf](https://indico.cern.ch/event/845380/attachments/1915103/3241592/Dvurechensky_lectures.pdf)

In diesem Kapitel betrachten wir einige Konzepte aus dem Bereich Optimal Transport, um einen Distanzbegriff zu entwickeln, der im Unterschied zur pixelweisen euklidischen Distanz besser für das kMeans-Clustering geeignet sein könnte. Ausgehend von Gaspard Monge's ursprünglicher Transportprobleme-Formulierung betrachten wir die durch Kantorovich eingeführte Relaxierung dieses Problems. Die Optimale Lösung wird für die Definition der p-Wasserstein-Distanz verwendet. Für eine Verallgemeinerung auf verschiedene Grundräume wird die Gromov-Wasserstein-Distanz definiert. Damit können wir zwei Heatmaps als sogenannte metrische Maßräume auffassen. Abschließend definieren wir sogenannte Gromov-Wasserstein-Baryzentren, die im Rahmen des kMeans-Clustering als Mittelwerte fungieren.

Stimmt Gromov-Wasserstein mit Wasserstein für dieselbe Distanzfunktion (Metrik) überein? Welche Rolle spielt die Lossfunktion  $L$  bei der Definition? Ja, siehe Memoli 2011.  $L=KL$

## 7.1 Einführung

### 7.1.1 Frage:

Was ist ein registration problem? Warum brauchen wir Gromov-Wasserstein? Warum reicht nicht Wasserstein? Weil wir Matrizen unterschiedlicher Größen vergleichen? Nein. Wir benutzen ja immer dieselbe Distanz für alle Pixel-Werte. Wenn wir eine Pixel-Wolke auswählen, sind das nicht dieselben Räume, da dann gewisse Koordinaten nicht zum Raum gehören. Wenn wir aber alle Pixel als Koordinaten im Raum auswählen würden, dann würden die metrischen Räume  $(X, d_X)$  und  $(Y, d_Y)$  übereinstimmen. Liegt das an der Definition der Baryzentren? Wie würde man das mit Baryzentren auf demselben Raum machen? s. [?] Translationen und Rotationen könnte ein wichtiger Punkt sein!!! Verwendung von Grafiken aus anderen Papern?? Kann man ein anderes Netzwerk zur Detektion nutzen?

<https://arxiv.org/pdf/1705.09634.pdf> <https://arxiv.org/pdf/1412.5154.pdf> Für die Grundlagen im Bereich des Optimalen Transports zitieren wir [?] Jilla-ni2009optimal.

Entscheidend für die Numerischen Resultate ist die Arbeit von Marco Cuturi [?, ?]

[?] Remark 2.19 (Translations). zeigt dass die Unabhängigkeit gegenüber Translationen? Kann ich das auch für die Gromov-Wasserstein-Distanz zeigen?

### 7.1.2 Anwendungsgebiete

Optimal transport is also used widely in domain adaptation [?]. In domain adaptation, one has access to labeled examples from a source domain and unlabeled examples from a target domain. The goal is to predict labels for examples from the target domain. This is different from the typical train-test paradigm in machine learning, because covariate shift between the two domains is allowed.

Ein weitere Anwendung von Optimal Transport sind Wasserstein Generative Adversarial Networks (WGANs) [?]. WGANs gehören zur Familie der Modelle namens Generative Adversarial Networks (GANs) [?]. Zu einem GAN gehören zwei Modelle, von denen beide typischerweise Neuronale Netzwerke sind. Das erste Netzwerk (generator) erzeugt Datenpunkte innerhalb des Datensatzes, die so realistisch wie möglich sein sollen. Das zweite Netzwerk (discriminator network) versucht,

zwischen den realen und den erzeugten Datenpunkten zu unterscheiden. Damit kann der Trainingsprozess von GANs als ein Spiel zwischen den beiden Netzwerken betrachtet werden, bei dem der Generator den Diskriminator täuschen und der Diskriminator die korruptierten Datenpunkte aufdecken möchte.

Durch die Verwendung von WGANs anstatt GANs wurde ein stabilerer Trainingsprozess erreicht, da diese weniger anfällig für abrupte Trainingsabbrüche sind. Die ursprünglichen GANs benutzten eine Jensen-Shannon-Divergenz als Verlustfunktion, die nicht überall stetig und fast überall differenzierbar ist. Auch andere Verlustfunktion wie beispielsweise die KL-Divergenz weisen ähnliche Schwierigkeiten auf. Durch die Verwendung der Wasserstein-Verlustfunktion, basierend auf der 1-Wasserstein-Distanz, die überall stetig und fast überall differenzierbar ist, konnte der Trainingsprozess der GANs somit deutlich verbessert werden [?].

**Definition 7.1.1.**

**Definition 7.1.2** (Histogramm). *Als Histogramm (oder: Wahrscheinlichkeitsvektor) bezeichnen wir ein Element  $\mathbf{a} \in \Sigma_n$ , das zum folgenden Wahrscheinlichkeits-Simplex gehört:*

$$\Sigma_n := \{\mathbf{a} \in \mathbb{R}_+^n \mid \sum_{i=1}^n \mathbf{a}_i = 1\}$$

**Definition 7.1.3** ([?]). *Ein metrischer Maßraum ist ein Tripel  $(X, d_X, \mu_X)$  mit*

- $(X, d_X)$  ist ein kompakter metrischer Raum
- $\mu_X$  ist ein Borel-Maß auf  $X$  mit vollem Support, d.h.

$$\text{supp}[\mu_X] = X. \quad (7.1)$$

**Definition 7.1.4** (Isometrie). *Seien zwei metrische Räume  $(X, d_X)$  und  $(Y, d_Y)$  gegeben. Wir nennen die Abbildung  $\phi : X \rightarrow Y$  Isometrie, falls  $\phi$  surjektiv ist und  $d_X(x, x') = d_Y(\phi(x), \phi(x'))$  f.a.  $x, x' \in X$  gilt. Für den Fall, dass eine solche Abbildung  $\phi$  existiert, nennen wir  $X$  und  $Y$  isometrisch.*

**Definition 7.1.5** (Isomorphie). *Zwei MMR  $(X, d_X, \mu_X)$  und  $(Y, d_Y, \mu_Y)$  heißen isomorph, falls eine Maß-erhaltende Abbildung  $f : X \rightarrow Y$  existiert, die  $f_{\#}\mu_X = \mu_Y$  erfüllt.*

Mit  $\mathcal{G}_{\mathcal{W}}$  bezeichnen wir die Menge aller metrischen Maßräume.

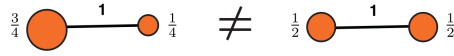
**Beispiel 7.1.6.** *Wir betrachten die beiden metrischen Maßräume  $(\{a, b\}, (\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}), \{\frac{1}{2}, \frac{1}{2}\})$  und  $(\{a', b'\}, (\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}), \{\frac{1}{4}, \frac{3}{4}\})$ . Dann sind diese Räume isometrisch, jedoch nicht isomorph, vergleiche dazu Abbildung 2*

**Definition 7.1.7** (pushforward-Maß/Bildmaß). *Seien  $X, Y$  zwei Maßräume,  $T : X \rightarrow Y$  eine messbare Abbildung und  $\mu$  ein Maß auf  $X$ . Dann ist das pushforward-Maß  $T^{\#}\mu$  ein Maß auf  $Y$ , definiert durch*

$$T^{\#}\mu(A) = \mu(T^{-1}(A)) \quad (7.2)$$

für alle  $A \subset Y$ .





**Abbildung 2:** Die beiden metrischen Maßräume sind ismetrisch, aber nicht isomorph.

Quelle: [?]

**Definition 7.1.8** (Kopplung). *Ein Wahrscheinlichkeitsmaß  $\mu$  auf  $X \times Y$  heißt Kopplung zwischen  $\mu_X$  und  $\mu_Y$ , falls  $(\mu_1)_\# \mu = \mu_X$  und  $(\mu_2)_\# \mu = \mu_Y$  gilt. Wir bezeichnen mit  $\Pi(\mu_X, \mu_Y)$  die Menge aller Kopplungen zwischen  $\mu_X$  und  $\mu_Y$ .*

Einführung W-Theorie am KIT <sup>17</sup>

**Definition 7.1.9** (Kopplungen zwischen Histogrammen). *Seien die beiden Histogramme  $p \in \Sigma_{N_1}$  und  $q \in \Sigma_{N_2}$  gegeben. Als Menge der Kopplungen zwischen beiden Histogrammen definieren wir*

$$\mathcal{C}_{p,q} := \{T \in (\mathbb{R}_+)^{N_1 \times N_2} \mid T \mathbb{K}_{N_2} = p, T^\top \mathbb{K}_{N_1} = q\},$$

wobei  $\mathbb{K} := (1, \dots, 1)^\top \in \mathbb{R}^N$  gilt.

**Definition 7.1.10** ([?]). *Sei  $X$  eine beliebige Menge. Eine Funktion  $d : X \times X \rightarrow \mathbb{R} \cup \{\infty\}$  heißt Metrik auf  $X$ , falls die folgenden Bedingungen für alle  $x, y, z \in X$  gelten:*

- (1) *Positive Definitheit:*  $d(x, y) > 0$  für  $x \neq y$  und  $d(x, y) = 0$  für  $x = y$ ,
- (2) *Symmetrie:*  $d(x, y) = d(y, x)$ ,
- (3) *Dreiecksungleichung:*  $d(x, z) \leq d(x, y) + d(y, z)$ .

Ein *metrischer Raum* ist eine Menge, versehen mit einer Metrik. Formal gesehen ist ein metrischer Raum ein Paar  $(X, d)$ , wobei  $d$  eine Metrik auf  $X$  ist. Elemente von  $X$  heißen Punkte und  $d(x, y)$  bezeichnet die Distanz zwischen  $x$  und  $y$ .

Geodesic Space <sup>18</sup>

**Definition 7.1.11** (Metrischer Maßraum). *Ein metrischer Maßraum ist ein Tripel  $(X, d_X, \mu_X)$ , wobei  $(X, d_X)$  ein metrischer Raum und  $\mu_X$  ein borelsches  $W$ -Maß auf  $X$  ist.*

**Definition 7.1.12** (Correspondance). *content...*

**Definition 7.1.13** (Kopplung). *Seien*

Seien  $(X, d_X)$  und  $(Y, d_Y)$  zwei metrische Räume. Wir betrachten im Folgenden die Abbildung

$$\Gamma_{X,Y} : (X \times Y) \times (X \times Y) \rightarrow \mathbb{R}^+, \quad (7.3)$$

gegeben durch

$$\Gamma_{X,Y}(x, y, x', y') := |d_X(x, x') - d_Y(y, y')|.$$

<sup>17</sup><https://www.math.kit.edu/stoch/~henze/media/wt-ss15-henze-handout.pdf>

<sup>18</sup><http://www.math.toronto.edu/mccann/papers/FiveLectures.pdf>

**Definition 7.1.14** (Entropischer Optimaler Transport). *Wir definieren den  $n$ -dimensionalen Zufalls-Simplex als  $\Sigma_n := \{a \in \mathbb{R}_+^n : \sum_{i=1}^n a_i = 1\}$ . Ein Element  $a \in \Sigma_n$  bezeichnen wir als Histogramm oder Zufallsvektor.*

**Definition 7.1.15** (Diskretes Maß). *Ein diskretes Maß mit Gewichtung  $\mathbf{a}$  und Punkten  $x_1, \dots, x_n \in X$  wird geschrieben als*

$$\alpha = \sum_{i=1}^n \mathbf{a}_i \delta_{x_i}, \quad (7.4)$$

wobei  $\delta_x$  das Dirac-Maß and Position  $x$  ist. Im dem Fall, dass  $\mathbf{a} \in \Sigma_n$  und  $\mathbf{a}_i > 0$  f.a.  $i$  git, sprechen wir von einem diskreten Wahrscheinlichkeitsmaß.

## 7.2 Notation

siehe: Introduction to Optimal Transport Matthew Thorpe

[http://inis.jinr.ru/sl/vol2/Ax-books/Disk\\_01/Gromov-Metric-structures-Riemann-non-Riemann.pdf](http://inis.jinr.ru/sl/vol2/Ax-books/Disk_01/Gromov-Metric-structures-Riemann-non-Riemann.pdf)

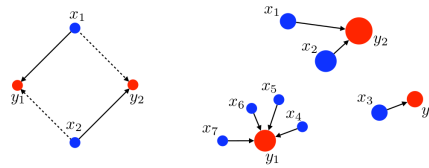
## 7.3 Kantorovichs Optimal Transport

In diesem Kapitel wollen wir einen Ähnlichkeitsbegriff für Maße einführen.

Der Bereich des Optimalen Transports beschäftigt sich mit der Frage, wie mehrerer Objekte, die sich in einer bestimmten Verteilung an einem Startpunkt befinden, auf optimale Weise zu einem Zielpunkt transportiert werden können. Dabei wird die Verteilung der Objekte als Wahrscheinlichkeitsverteilung interpretiert.

### 7.3.1 Optimaler Transport (Monge Formulierung)

Dieses Problem wurde zuerst von Gaspard Monge im Jahr 1781 formuliert. In dieser Formulierung wurde Abbildung gesucht, die jedem Punkt in der Ausgangsverteilung einen Punkt in der Zielverteilung zuordnet. Dabei wird die gesamte Masse eines Punktes in der Startverteilung an einen Punkt in der Zielverteilung verschoben. Eine solche Abbildung ist in Abbildung 3 dargestellt.



**Abbildung 3: Links:** Fehlende Eindutigkeit in der Zuordnung. Die beiden Punkte  $x_1$  und  $x_2$  können sowohl den Punkten  $y_1$  bzw.  $y_2$  zugeordnet werden, um eine zulässige Abbildung zu erhalten. **Rechts:** Die Monge-Abbildung assoziiert das blaue Maß  $\alpha$  mit dem roten Maß  $\beta$ . Dabei ist die Masse in den jeweiligen Punkten über den Flächeninhalt der Kreise dargestellt.

Quelle: [?]

Für die exakte Formulierung im diskreten Fall benötigen wir die folgende Definition.

**Definition 7.3.1** (Bildmaß). Für zwei Maßräume  $(X, \Sigma_X)$  und  $(Y, \Sigma_Y)$  und eine messbare Abbildung  $f : X \rightarrow Y$  ist das Bildmaß  $f_{\#}\mu$  auf  $(Y, \Sigma_Y)$  definiert als  $f_{\#}\mu = \mu(f^{-1}(B))$  für alle  $B \in \Sigma_Y$ .

Für den Fall zweier diskreter Maße

$$\alpha = \sum_{i=1}^n \mathbf{a}_i \delta_{x_i} \text{ und } \beta = \sum_{j=1}^m \mathbf{b}_j \delta_{y_j} \quad (7.5)$$

können wir das Monge-Problem dann wie folgt formulieren:

Gesucht ist eine Abbildung  $T : \{x_1, \dots, x_n\} \rightarrow \{y_1, \dots, y_m\}$ , die die Bedingung

$$\forall j \in \{1, \dots, m\} \mathbf{b}_j = \sum_{i: T(x_i)=y_j} \mathbf{a}_i \quad (7.6)$$

erfüllt, die wir in Kurzform als  $T_{\#}\alpha = \beta$  schreiben. Aufgrund der Positivität von  $\mathbf{b}$  ist die Abbildung notwendigerweise surjektiv. Als weitere Forderung gilt, dass die Abbildung  $T$  minimal bezüglich einer Transportkostenfunktion  $c(x, y)$  für  $x, y \in \mathbb{R} \times \mathbb{R}$  sein soll:

$$\min_T \sum_i c(x_i, t(x_i)) : T_{\#}\alpha = \beta. \quad (7.7)$$

Neben der fehlenden Eindeutigkeit einer solchen Abbildung ist auch die Existenz nicht immer gegeben.

### 7.3.2 Optimaler Transport nach Kantorovich

Das Optimal Transport Problem nach Kantorovich gehört zu den typischen Optimal Transport Problemen. Es stellt eine Relaxierung der Formulierung von Gaspard Monge dar, bei der nun auch das Aufteilen von Masse (mass splitting) zulässig ist, d.h. die Masse an einem Startpunkt, kann auf mehrer Zielpunkte abgebildet werden.

In Kantorovichs Formulierung ist eine Kopplung (oder: Transportabbildung)  $\mathbf{T}$  gesucht, die die Kosten, die bei der Verschiebung eines diskreten Maßes  $\mathbf{a}$  auf ein anderes diskretes Maß  $\mathbf{b}$  bezüglich der Kosten  $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$  entstehen, minimiert. Damit  $\mathbf{T}$  eine Transportabbildung ist, muss  $\mathbf{T} \in \Gamma(\mathbf{a}, \mathbf{b}) = \{\mathbf{T} \geq \mathbf{0}, \mathbf{T}\mathbf{1}_{n_2} = \mathbf{a}, \mathbf{T}^T \mathbf{1}_{n_1} = \mathbf{b}\}$  gelten.

Ein solcher Transportplan ist für drei verschiedene Ausgangssituationen in ?? dargestellt.

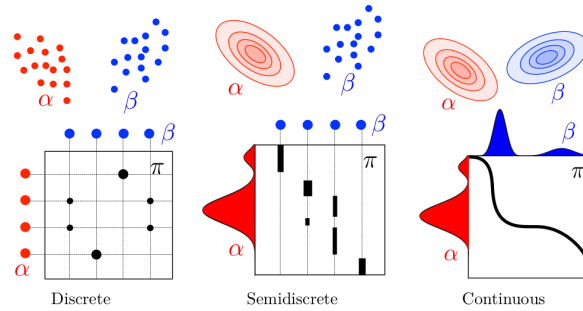
Für den Fall, dass die Grundkosten eine Metrik darstellen, ist auch die optimale Lösung des Optimal Transport Problems wieder eine Metrik [?] und definiert die *Wasserstein Distanz*. Das OT Problem ist definiert als

$$W_M(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{T} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{T}, \mathbf{M} \rangle, \quad (7.8)$$

wobei  $\langle \mathbf{T}, \mathbf{M} \rangle = \sum_{ij} t_{ij} m_{ij}$  gilt.

Die Lösung des Problems ist eine Kopplung/Kopplungsmatrix, die beschreibt, wie viel Masse von einem Punkt zum anderen Punkt fließt. In einer Kostenmatrix derselbe Größe sind die Kosten abgespeichert, um von einem zum anderen Punkt zu kommen.

Die Menge der Matrizen  $\Pi(\mathbf{a}, \mathbf{b})$  ist beschränkt und durch  $n + m$  Gleichungen gegeben und damit ein konvexes Polytop (die konvexe Hülle einer endlichen Menge



**Abbildung 4: Links:** Kopplung zwischen zwei diskreten Wahrscheinlichkeitsverteilungen. **Mitte:** Kopplung zwischen einer kontinuierlichen W-Verteilung  $\alpha$  und einer diskreten W-Verteilung  $\beta$ . **Rechts:** Kopplung im kontinuierlichen Fall.

Quelle: [?]

von Matrizen). Zudem ist die Formulierung von Kantorovich im Unterschied zu Monges Formulierung immer symmetrisch in dem Sinne, dass  $\mathbf{P} \in \Pi(\mathbf{a}, \mathbf{b})$  genau dann gilt, wenn  $\mathbf{P}^\top \in \Pi(\mathbf{b}, \mathbf{a})$ .

Kantorovichs Optimal Transport Problem lässt sich nun schreiben als

$$L_C(\mathbf{a}, \mathbf{b}) := \min_{\mathbf{P} \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle := \sum_{i,j} C_{i,j} P_{i,j} \quad (7.9)$$

Diese Problem ist ein lineares Problem. Für diese Art von Problemen ist die optimale Lösung nicht notwendigerweise eindeutig. Kantorovich gilt als Erfinder der linearen Optimierung<sup>19</sup>. Problem Gleichung 7.9 ist ein unendlich-dimensionales lineares Optimierungsproblem über einem Raum von Maßen. Falls  $(\mathcal{X}, \mathcal{Y})$  kompakt und  $c$  stetig ist, existiert immer eine Lösung.

### 7.3.3 Metrische Eigenschaften

Optimaler Transport definiert eine Distanz zwischen Histogrammen und W-Maßen, sofern die Kostenmatrix gewisse Eigenschaften erfüllt. Optimal Transport kann dabei als naheliegende Idee verstanden werden, um Distanzen zwischen Punkten auf Distanzen zwischen Histogrammen oder Maßen zu verallgemeinern.

**Definition 7.3.2** ( $p$ -Wasserstein-Distanz auf  $\Sigma_n$ ). Sei  $n = m$  und für  $p \geq 1$  gelte  $\mathbf{C} = \mathbf{D}^p = (\mathbf{D}_{i,j}^p)_{i,j} \in \mathbb{R}^{n \times n}$ , wobei  $\mathbf{D} \in \mathbb{R}_+^{n \times n}$  eine Metrik ist.

Dann definiert

$$W_p(\mathbf{a}, \mathbf{b}) := L_{\mathbf{D}^p}(\mathbf{a}, \mathbf{b})^{1/p} \quad (7.10)$$

die  $p$ -Wasserstein-Distanz auf  $\Sigma_n$

**Lemma 7.3.3.**  $W_p$  ist eine Metrik

<sup>19</sup>OTNotes\_campride.pdf

*Beweis.* Nach Voraussetzung besitzt  $\mathcal{C} = \mathcal{D}^p$  eine Nulldiagonale. Somit gilt  $W_p(\mathbf{a}, \mathbf{a}) = 0$ . Die zugehörige Transportmatrix ist  $\mathbf{P}^* = \text{diag}(\mathbf{a})$ . Aufgrund der Positivität aller Nicht-Diagonalelemente von  $\mathbf{D}^p$  gilt  $W(\mathbf{a}, \mathbf{b})$  für  $\mathbf{a} \neq \mathbf{b}$ , da in diesem Fall jede zulässige Kopplungen und damit insbesondere die optimale Kopplungen ein Nicht-Diagonalelement ungleich 0 besitzt. Die Symmetrie von  $W_p(\mathbf{a}, \mathbf{b})$  gilt wegen der Symmetrie von  $\mathbf{D}^p$ .

Für den Nachweis der Dreiecksungleichung im Fall beliebiger Maße nutzt Villani [?] das sogenannte Gluing Lemma. Im diskreten Fall ist die Konstruktion dieser geklebten Kopplung etwas einfacher. Seien  $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \Sigma_n$ . Seien  $\mathbf{P}$  und  $\mathbf{Q}$  zwei optimale Lösungen des Transportproblems zwischen  $\mathbf{a}$  und  $\mathbf{b}$  bzw.  $\mathbf{b}$  und  $\mathbf{c}$ . Wir definieren  $\tilde{\mathbf{b}}$ , wobei  $\tilde{b}_j = b_j$ , falls  $b_j > 0$ , und  $\tilde{b}_j = 1$  sonst gilt. Damit können wir

$$\mathbf{S} := \mathbf{P} \text{diag}(1/\tilde{\mathbf{b}}) \mathbf{Q} \in \mathbb{R}_+^{n \times n} \quad (7.11)$$

schreiben. Es gilt  $\mathbf{S} \in \Pi(\mathbf{a}, \mathbf{c})$  wegen

$$\mathbf{S} \mathbf{1}_n = \mathbf{P} \text{diag}(1/\tilde{\mathbf{b}}) \mathbf{Q} \mathbf{1}_n = \mathbf{P}(\mathbf{b}/\tilde{\mathbf{b}}) = \mathbf{P} \mathbf{1}_{\text{supp}[\mathbf{b}]} = \mathbf{a}, \quad (7.12)$$

wobei  $\mathbf{1}_{\text{supp}[\mathbf{b}]}$  den Vektor der Größe  $n$  bezeichnet, der Einsen an den Stellen mit Indizes  $j$  besitzt, für die auch  $b_j > 0$  gilt, und sonst aus Nullen besteht. Außerdem wurde  $\mathbf{P} \mathbf{1}_{\text{supp}[\mathbf{b}]} = \mathbf{P} \mathbf{1}_n = \mathbf{a}$  benutzt, denn es gilt notwendigerweise  $P_{i,j} = 0$  für diejenigen  $j$  mit  $b_j = 0$ . Analog folgt  $\mathbf{S}^\top \mathbf{1}_n = \mathbf{c}$ .

Damit erhalten wir

$$W_p(\mathbf{a}, \mathbf{c}) = \left( \min_{\mathbf{P} \in \Pi(\mathbf{a}, \mathbf{c})} \langle \mathbf{P}, \mathbf{D}^p \rangle \right)^{1/p} \leq \langle \mathbf{S}, \mathbf{D}^p \rangle^{1/p} \quad (7.13)$$

$$= \left( \sum_{ik} D_{ik}^p \sum_j \frac{P_{ij} Q_{jk}}{\tilde{b}_j} \right)^{1/p} \leq \left( \sum_{ijk} (D_{ij} + D_{jk})^p \frac{P_{ij} Q_{jk}}{\tilde{b}_j} \right)^{1/p} \quad (7.14)$$

$$\leq \left( \sum_{ijk} D_{ij}^p \frac{P_{ij} Q_{jk}}{\tilde{b}_j} \right)^{1/p} + \left( \sum_{ijk} D_{jk}^p \frac{P_{ij} Q_{jk}}{\tilde{b}_j} \right)^{1/p}. \quad (7.15)$$

□

Dabei haben wir in der ersten Ungleichung benutzt, dass  $\mathbf{S}$  keine optimale Lösung ist. Für die zweite Ungleichung wurde die Dreiecksungleichung für  $\mathbf{D}$  verwendet und die dritte Ungleichung folgt aus der Minkowski-Ungleichung. Nach Umstellen der beiden letzten Terme erhalten wir damit

$$\begin{aligned} W_p(\mathbf{a}, \mathbf{c}) &\leq \left( \sum_{ij} D_{ij}^p P_{ij} \sum_k \frac{Q_{jk}}{\tilde{b}_j} \right)^{1/p} + \left( \sum_{jk} D_{jk}^p Q_{jk} \sum_i \frac{P_{ij}}{\tilde{b}_j} \right)^{1/p} \\ &= \left( \sum_{ij} D_{ij}^p P_{ij} \right)^{1/p} + \left( \sum_{jk} D_{jk}^p Q_{jk} \right)^{1/p} \\ &= W_p(\mathbf{a}, \mathbf{b}) + W_p(\mathbf{b}, \mathbf{c}) \end{aligned}$$

und damit die Behauptung.

**Bemerkung 7.3.4** (Der Fall  $0 < p \leq 1$ ). Für  $0 < p \leq 1$  ist auch  $D^p$  eine Distanz. Damit ist für  $p \leq 1$   $W_p(\mathbf{a}, \mathbf{b})^p$  eine Distanz auf dem Simplex.

**Bemerkung 7.3.5.** Für  $p = 1$  ist die  $p$ -Wasserstein-Distanz auch als *Earth Movers's Distance* [?] bekannt

Wasserstein-Metrik bildet einen metrischen Raum: siehe optimal Transport\_tübingen.pdf

Das hatte ich ja eigentlich schon erwähnt, indem ich zeige, dass Wasserstein eine Metrik ist, die Konsequenz ist ja dann ein metrischer Raum. Die Menge  $X$  muss ich noch angeben:

### 7.3.4 Duale Formulierung

Kurzer Überblick hier<sup>20</sup> oder (Herleitung der Dualen Formulierung mit Beweis)<sup>21</sup>.

The Kantorovich problem (2.11) is a constrained convex minimization problem, and as such, it can be naturally paired with a so-called dual problem, which is a constrained concave maximization problem. The following fundamental proposition explains the relationship between the primal and dual problems.

Der Vollständigkeit halber geben wir auch die Definition im kontinuierlichen Fall an:

**Definition 7.3.6.** [?] Sei  $(X, d)$  ein Polnischer Raum. Sei  $p \in [1, \infty)$ . Für zwei Wahrscheinlichkeitsmaße  $\mu, \nu$  auf  $X$  ist die Wasserstein-Distanz der Ordnung  $p$  (oder kurz:  $p$ -Wasserstein-Distanz) zwischen  $\mu$  und  $\nu$  definiert durch

$$W_p(\mu, \nu) = \left( \inf_{\pi \in \Pi(\mu, \nu)} \int_X d(x, y)^p d\pi(x, y) \right)^{1/p}. \quad (7.16)$$

### 7.3.5 Verfahren zur Berechnung der exakten Lösung

- Network Simplex Algorithm
- Dual Ascent Methods
- Auction Algorithm

### 7.3.6 Komplexitätsanalyse

Im Fall  $n = m$  ist das Kantorovich-Problem und damit die Berechnung der Wasserstein-Distanz ein lineares Optimierungsproblem mit  $\mathcal{O}(n)$  linearen Bedingungen. Für die Lösung kann beispielsweise der lineare Lee-Siford-Algorithmus zur Berechnung einer Lösung in  $\tilde{\mathcal{O}}(n^{2.5})$  [?] verwendet werden, der den vorherigen Standard von  $\mathcal{O}(n^{2.5})$  [?] verbessert.

### 7.3.7 Wasserstein-Baryzentren

[?] <https://optimaltransport.github.io/slides-peyre/CourseOT.pdf> Fast Computation of Wasserstein Barycenters<sup>22</sup>

<sup>20</sup><https://arxiv.org/pdf/1609.04767.pdf>

<sup>21</sup>text

<sup>22</sup><https://arxiv.org/pdf/1310.4375.pdf>

## 7.4 Entropisch regularisierte Wasserstein-Distanz

In diesem Abschnitt betrachten wir regularisierte Version der Wasserstein-Distanz mithilfe von Entropie. Diese von Marco Cuturi vorgestellte Regularisierung [?] verhalf dem Gebiet des Computational Optimal Transport zu sehr großem Interesse. Durch die Regularisierung mit Entropie entsteht eine Kullback-Leibler-Distanz, für die Standard-Verfahren wie beispielsweise der Sinkhorn-Algorithmus verwendet werden können. Andere Arten der Regularisierung sind Group Lasso [Courty et al., 2016a] und KL, Itakura Saito,  $\beta$ -divergences, [Dessein et al., 2016].<sup>23</sup>

Die Entropie einer Matrix ist wie folgt definiert:

**Definition 7.4.1** (Entropie). Für  $T \in \mathbb{R}_+^{n \times n}$  definieren wir die Entropie als

$$H(T) := - \sum_{i,j=1}^N T_{i,j} (\log(T_{i,j}) - 1). \quad (7.17)$$

Entropic Regularization of Optimal Transport [?]

Optimal Transport: Regularization and Applications <sup>24</sup> Python Optimal Transport Toolbox<sup>25,26</sup>

Kapitel 2.1 im Paper: Vergleich von Histogrammen auf demselben metrischen Raum.

**Lemma 7.4.2.**

$$L_C(\mathbf{a}, \mathbf{b}) := \min_{P \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon H(\mathbf{P}) \quad (7.18)$$

ist ein  $\varepsilon$ -streng konvexes Problem und besitzt deshalb eine eindeutige optimale Lösung.

*Beweis.* Die Entropie  $H$  ist wegen der Hesse-Matrix  $\partial^2 H(P) = -\text{diag}(1/P_{i,j})$  und  $P_{i,j} \leq 1$  stark konkav. Durch die Multiplikation mit  $\varepsilon$  und anschließender Subtraktion wird aus dem obigen Problem ein konvexes.  $\square$

**Bemerkung 7.4.3.** Für größere  $\varepsilon$  wird mehr Entropie gefordert

**Proposition 7.4.4** (Konvergenz in  $\varepsilon$ ). Die eindeutige Lösung  $P_\varepsilon$  von Gleichung 7.18 konvergiert gegen die optimale Lösung mit maximaler Entropie innerhalb der Menge aller optimalen Lösungen des Kantorovich Problems, d.h.

$$P_\varepsilon \xrightarrow{\varepsilon \rightarrow 0} \arg \min_P \{-H(P) : P \in \Pi(\mathbf{a}, \mathbf{b}), \langle \mathbf{P}, \mathbf{C} \rangle = L_C(\mathbf{a}, \mathbf{b})\}, \quad (7.19)$$

d.h. es gilt

$$L_C^\varepsilon(\mathbf{a}, \mathbf{b}) \xrightarrow{\varepsilon \rightarrow 0} L_C(\mathbf{a}, \mathbf{b}) \quad (7.20)$$

Zudem gilt

$$P_\varepsilon \xrightarrow{\varepsilon \rightarrow \infty} \mathbf{a} \otimes \mathbf{b} = \mathbf{a} \mathbf{b}^\top = (\mathbf{a}_i \mathbf{b}_j)_{i,j}. \quad (7.21)$$

<sup>23</sup>[https://www.damtp.cam.ac.uk/research/cia/files/teaching/Optimal\\_Transport\\_Syllabus.pdf](https://www.damtp.cam.ac.uk/research/cia/files/teaching/Optimal_Transport_Syllabus.pdf)

<sup>24</sup><https://www.otra2020.com/schedule>

<sup>25</sup><https://pythonot.github.io/quickstart.html>

<sup>26</sup>[https://pythonot.github.io/auto\\_examples/gromov/plot\\_gromov.html](https://pythonot.github.io/auto_examples/gromov/plot_gromov.html)

*Beweis.* Sei eine Folge  $\varepsilon_l$  mit  $\varepsilon_l \rightarrow 0$  und  $\varepsilon_l > 0$  gegeben. Sei  $P_l$  die Lösung von Gleichung 7.18 für  $\varepsilon = \varepsilon_l$ . Da  $\Pi(a, b)$  beschränkt ist, existiert eine Teilfolge  $\varepsilon_k$  mit  $P_k \rightarrow P^*$ . Aufgrund der Abgeschlossenheit von  $\Pi(a, b)$  folgt  $P^* \in \Pi(a, b)$ .

Sei  $P \in \Pi(a, b)$  beliebig mit  $\langle C, P \rangle = L_C(a, b)$ . Aufgrund der Optimalität von  $P$  und  $P_l$  bezüglich  $L_C(a, b)$  bzw.  $L_C^{\varepsilon_l}(a, b)$  gilt

$$0 \leq \langle C, P_l \rangle - \langle C, P \rangle \leq \varepsilon_l (H(P_l) - H(P)). \quad (7.22)$$

Aufgrund der Stetigkeit von  $H$  folgt für  $l \rightarrow +\infty$  in Gleichung 7.22  $\langle C, P^* \rangle = \langle C, P \rangle$ , womit  $P^*$  ein zulässiger Punkt ist. Division durch  $\varepsilon_l$  in Gleichung 7.22 und Übergang zum Grenzwert liefert  $H(P) \leq H(P^*)$ . Folglich ist  $P^*$  eine Lösung von Gleichung 7.19.

Da die Lösung  $P_0^*$  aufgrund der strikten Konvexität von  $-H$  eindeutig ist, gilt  $P^* = P_0^*$  und die gesamte Folge konvergiert.  $\square$

**Bemerkung 7.4.5.** Gleichung 7.19 zeigt, dass die Lösung für kleine  $\varepsilon$  gegen die Optimale Transport Kopplung mit maximaler Entropie konvergiert. Im Gegensatz dazu, bedeutet Gleichung 7.21, die Lösung für große Regularisierungsparameter konvergiert gegen die Kopplung mit maximaler Entropie zwischen zwei gegebenen Randverteilungen  $\mathbf{a}$  und  $\mathbf{b}$ .

**Bemerkung 7.4.6.** A key insight is that, as  $\varepsilon$  increases, the optimal coupling becomes less and less sparse (in the sense of having entries larger than a prescribed threshold), which in turn has the effect of both accelerating computational algorithms (as we study in Abschnitt 4.2) and leading to faster statistical convergence (as shown in Abschnitt 8.5)

**Interpretation als KL Problem:** Gleichung 7.18 kann außerdem als Spezialfall eine Kullback-Leiber Minimierungsproblems betrachtet werden, bei dem eine Kopplungsmatrix  $\mathbf{P}_\varepsilon$  gesucht ist, die im Sinner einer Kullback-leibler-Divergenz möglichst nahe an einem Kernel  $\mathbf{K}$  liegt. Wir definieren die Kullback-Leibler-Divergenz zwischen Kopplungen als

$$KL(\mathbf{P}|\mathbf{K}) := \sum_{i,j} P_{i,j} \log \left( \frac{P_{i,j}}{K_{i,j}} \right) - P_{i,j} + K_{i,j}. \quad (7.23)$$

Damit ist die eindeutige Lösung  $\mathbf{P}_\varepsilon$  von Gleichung 7.18 eine Projektion des zur Kostenmatrix  $\mathbf{C}$  gehörigen Gibbs-Kernels  $\mathbf{K}_{i,j} := e^{-\frac{C_{i,j}}{\varepsilon}}$  auf  $\Pi(\mathbf{a}, \mathbf{b})$ .

Mit der obigen Definition erhalten wir

$$\mathbf{P}_\varepsilon = Proj_{\Pi(\mathbf{a}, \mathbf{b})}^{KL}(\mathbf{K}) := \arg \min_{P \in \Pi(\mathbf{a}, \mathbf{b})} KL(\mathbf{P}|\mathbf{K}). \quad (7.24)$$

#### 7.4.1 Algorithmus von Sinkhorn und Variationen

In diesem Kapitel sehen wir, dass die Lösung des regularisierten Problems eine besondere Form besitzt, die wir über  $m + n$  Variablen parametrisieren können.

Matrix Scaling + einige Theoreme und Bemerkungen zu Kullback-Leibler und mehr Info [?]



**Proposition 7.4.7.** *Die Lösung des regularisierten Problems Gleichung 7.18 besitzt die Form*

$$\forall(i, j) \in \{1, \dots, n\} \times \{1, \dots, m\} : \mathbf{P}_{i,j} = \mathbf{u}_i \mathbf{K}_{i,j} \mathbf{v}_j \quad (7.25)$$

für die beiden (unbekannten) Variablen  $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}_+^n \times \mathbb{R}_+^m$ .

*Beweis.* Wir führen für jede der beiden Nebenbedingungen die dualen Variablen  $\mathbf{f} \in \mathbb{R}^n$  und  $\mathbf{g} \in \mathbb{R}^m$  ein. Für die Lagrange-Funktion zu Gleichung 7.18 erhalten wir damit:

$$\mathcal{L}(\mathbf{P}, \mathbf{f}, \mathbf{g}) = \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon \mathbf{H}(\mathbf{P}) - \langle \mathbf{f}, \mathbf{P} \mathbf{1}_m - \mathbf{a} \rangle - \langle \mathbf{g}, \mathbf{P}^\top \mathbf{1}_n - \mathbf{b} \rangle. \quad (7.26)$$

Mit der Optimalitätsbedingung erster Ordnung ergibt sich

$$\frac{\partial \mathcal{L}(\mathbf{P}, \mathbf{f}, \mathbf{g})}{\partial \mathbf{P}_{i,j}} = \mathbf{C}_{i,j} + \varepsilon \log(\mathbf{P}_{i,j}) - \mathbf{f}_i - \mathbf{g}_j = 0, \quad (7.27)$$

womit wir für eine optimale Kopplung  $\mathbf{P}$  für das regularisierte Problem den Ausdruck  $\mathbf{P}_{i,j} = e^{\mathbf{f}_i/\varepsilon} e^{-\mathbf{C}_{i,j}/\varepsilon} e^{\mathbf{g}_j/\varepsilon}$ , der in die gewünschte Form umgeschrieben werden kann.  $\square$

**Bemerkung 7.4.8** (Algorithmus von Sinkhorn). *Die Faktorisierung der Lösung in Gleichung 7.25 können in der folgenden Matrix-Form schreiben:  $\mathbf{P} = \text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v})$ . Die beiden Variablen  $(\mathbf{u}, \mathbf{v})$  müssen deshalb die folgenden nichtlinearen Gleichungen erfüllen, die aufgrund der geforderten Massenerhaltungsbedingung in  $\Pi(\mathbf{a}, \mathbf{b})$  gelten:*

$$\text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v}) \mathbf{1}_m = \mathbf{a} \text{ und } \text{diag}(\mathbf{v}) \mathbf{K}^\top \text{diag}(\mathbf{u}) \mathbf{1}_n = \mathbf{b}. \quad (7.28)$$

Aufgrund der Beziehung  $\text{diag}(\mathbf{v}) \mathbf{1}_m = \mathbf{v}$  und selbiger Beziehung für  $\mathbf{u}$  erhalten wir die folgende Vereinfachung

$$\mathbf{u} \odot (\mathbf{K} \mathbf{v}) = \mathbf{a} \text{ und } \mathbf{v} \odot (\mathbf{K}^\top \mathbf{u}) = \mathbf{b}, \quad (7.29)$$

wobei  $\odot$  für die Element-weise Multiplikation zweier Vektoren steht. Dieses Problem ist als *Matrix Scaling Problem* [?] bekannt.

(? Sollte ich hier Bedingungen für die Lösbarkeit angeben?)

Eine Möglichkeit zur Lösung dieses Problems ist ein iteratives Vorgehen(?Verfahren), bei dem zunächst  $\mathbf{u}$  so modifiziert wird, dass die linke Seite in Gleichung 7.29 erfüllt ist, und anschließend die Modifikation von  $\mathbf{v}$  vorgenommen wird, sodass die rechte Seite in Gleichung 7.29 gilt. Mit diesen beiden Modifikationen erhalten wir den Algorithmus von Sinkhorn, der aus den beiden folgenden Updates besteht:

$$\mathbf{u}^{l+1} := \frac{\mathbf{a}}{\mathbf{K} \mathbf{v}^{(l)}} \text{ und } \mathbf{v}^{l+1} := \frac{\mathbf{b}}{\mathbf{K}^\top \mathbf{u}^{(l+1)}}, \quad (7.30)$$

wobei zu Beginn mit einem beliebigen positiven Vektor, beispielsweise  $\mathbf{v}^{(0)} = \mathbf{1}_m$  initialisiert wird und  $l$  den aktuellen Iterationsschritt bezeichnet. Die obigen Division muss ebenfalls elementweise verstanden werden.

The iterations (4.15) first appeared in [Yule, 1912, Kruithof, 1937]. They were later known as the iterative proportional fitting procedure (IPFP) Deming and Stephan [1940] and RAS [Bacharach, 1965] methods [Idel, 2016]. The proof of their convergence is attributed to Sinkhorn [?], hence the name of the algorithm.

**Bemerkung 7.4.9.** Die Lösung zum regularisierten Problem lässt sich nach den Sinkhorn-Iterationen wie folgt angeben:  $P_L = \text{diag}(U_L)K\text{diag}(v_L)$ . Das ist der optimal Transport plan für das Regularisierte Problem. Optimal Transport Caost lässt sich berechnen als:  $\langle P_L, M_{XY} \rangle = u_L^\top (K \odot M_{XY})v_L$ .

Für die globale Konvergenzanalyse der Sinkhorn-Algorithmus lässt sich ein Zusammenhang mit der Hilbertschen Projektionsmetrik benutzen, der erstmals in [?] vorgestellt wurde.

Diese ist wie folgt definiert:

**Definition 7.4.10** (Projektive Hilbert-Metrik). Seien  $u, u' \in \mathbb{R}_{+,*}^n$ . Dann ist die projektive Hilbert-Metrik  $d_{\mathcal{H}}$  definiert durch

$$d_{\mathcal{H}}(u, u') := \log \max_{i,j} \frac{u_i u'_j}{u_j u'_i}. \quad (7.31)$$

**Bemerkung 7.4.11.** It can be shown to be a distance on the projective cone  $R_{n,+,*}/\sim$ , where  $u \sim u_0$  means that  $\exists r > 0, u = ru_0$  (the vectors are equal up to rescaling, hence the name “projective”). This means that  $d_{\mathcal{H}}$  satisfies the triangular inequality and  $d_{\mathcal{H}}(u, u_0) = 0$  if and only if  $u \sim u_0$ . This is a projective version of Hilbert’s original distance on bounded open convex sets [Hilbert, 1895]. The projective cone  $R_{n,+,*}/\sim$  is a complete metric space for this distance. By a logarithmic chan

the Hilbert metric on the rays of the positive cone is isometric to the variation seminorm (it is a norm between vectors that are defined up to an additive constant)  $d_{\mathcal{H}}(u, u_0) = \log(u) - \log(u_0)$

where (4.21) var def. wobei  $\|f\|_{\text{var}} := (\max_i f_i) - (\min_i f_i)$ .

This variation seminorm is closely related to the ‘ norm since one always has  $\|f\|_{\text{var}} \leq \|f\|_{\infty}$ . If one imposes that  $f_i = 0$  for some fixed  $i$ , then a converse inequality also holds since  $\|f\|_{\infty} \leq \|f\|_{\text{var}}$ . These bounds are especially useful to analyze Sinkhorn convergence (see Remark 4.14 below), because dual variables  $f = \log(u)$  solving (4.14) are defined up to an additive constant, so that one can impose that  $f_i = 0$  for some  $i$ .

The Hilbert metric was introduced independently by [Birkhoff, 1957] and [Samelson et al., 1957]. They proved the following fundamental theorem, which shows that a positive matrix is a strict contraction on the cone of positive vectors.

**Theorem 7.4.12.** Sei  $K \in \mathbb{R}_{+,*}^{n \times m}$ . Dann gilt für  $(v, v') \in (\mathbb{R}_{+,*}^m)^2$

$$d_{\mathcal{H}}(Kv, Kv') \leq \lambda(K) d_{\mathcal{H}}(v, v'), \quad \text{mit} \quad \begin{cases} \lambda(K) := \frac{\sqrt{\nu(K)}-1}{\sqrt{\nu(K)}+1} \\ \nu(K) := \max_{i,j,k,l} \frac{K_{i,k}K_{j,l}}{K_{j,k}K_{i,l}} \end{cases} \quad (7.32)$$

Visualisierung des Theorems vgl [?],Seite 70.

Mit diesem Resultat lässt sich die Konvergenz des Verfahrens zeigen:

**Theorem 7.4.13.** Es gilt  $(u^{(l)}, v^{(l)}) \rightarrow (u^*, v^*)$  und

$$d_{\mathcal{H}}(u^{(l)}, u^*) = \mathcal{O}(\lambda(K)^{2l}), \quad d_{\mathcal{H}}(v^{(l)}, v^*) = \mathcal{O}(\lambda(K)^{2l}) \quad (7.33)$$

Es gelten außerdem die beiden folgenden Abschätzungen

$$d_{\mathcal{H}}(u^{(l)}, u^*) \leq \frac{d_{\mathcal{H}}(P^{(l)} \mathbf{1}_m, a)}{1 - \lambda(K)^2}, \quad (7.34)$$

$$d_{\mathcal{H}}(v^{(l)}, v^*) \leq \frac{d_{\mathcal{H}}(P^{(l)\top} \mathbf{1}_n, b)}{1 - \lambda(K)^2}, \quad (7.35)$$

wobei

$$P^{(l)} := \text{diag}(u^{(l)}) K \text{diag}(v^{(l)}) \quad (7.36)$$

gilt. Desweiteren gilt die Abschätzung

$$\|\log(P^{(l)}) - \log(P^*)\|_{\infty} \leq d_{\mathcal{H}}(u^{(l)}, u^*) + d_{\mathcal{H}}(v^{(l)}, v^*), \quad (7.37)$$

wobei  $P^*$  die eindeutige Lösung zu ?? ist.

*Beweis.* Für ein beliebiges Paar  $(v, v') \in (\mathbb{R}_{+,*}^m)^2$  gilt

$$d_{\mathcal{H}}(v, v') = d_{\mathcal{H}}(v/v', \mathbf{1}_m) = d_{\mathcal{H}}(\mathbf{1}_m/v, \mathbf{1}_m/v').$$

Mit dieser Beziehung und Theorem 7.4.12 erhalten wir

$$\begin{aligned} d_{\mathcal{H}}(u^{(l+1)}, u^*) &= d_{\mathcal{H}}\left(\frac{a}{Kv^{(l)}, \frac{a}{Kv^*}}\right) \\ &= d_{\mathcal{H}}(Kv^{(l)}, Kv^*) \\ &\leq \lambda(K) d_{\mathcal{H}}(v^{(l)}, v^*). \end{aligned}$$

Dies zeigt Gleichung 7.33. Mit Hilfe der Dreiecksungleichung erhalten wir

$$\begin{aligned} d_{\mathcal{H}}(u^{(l)}, u^*) &\leq d_{\mathcal{H}}(u^{(l+1)}, u^{(l)}) + d_{\mathcal{H}}(u^{(l+1)}, u^*) \\ &\leq d_{\mathcal{H}}\left(\frac{a}{Kv^{(l)}}, u^{(l)}\right) + \lambda(K)^2 d_{\mathcal{H}}(u^{(l)}, u^*) \\ &= d_{\mathcal{H}}\left(a, u^{(l)} \odot (Kv^{(l)})\right) + \lambda(K)^2 d_{\mathcal{H}}(u^{(l)}, u^*) \\ &= d_{\mathcal{H}}\left(a, P^{(l)} \mathbf{1}_m\right) + \lambda(K)^2 d_{\mathcal{H}}(u^{(l)}, u^*) \end{aligned}$$

und damit nach Division durch  $1 - \lambda(K)^2$  Gleichung 7.34. Die zweite Abschätzung, Gleichung 7.35, folgt analog. Die Gleichung 7.37 gilt nach Lemma 3 in [?].  $\square$

#### Bemerkung 7.4.14.

- Konvergenzrate?
- Berechnung der optimalen Transportpläne mit Gleichung 7.36 im Zeitschritt  $l$  und Berechnung der Transportkosten ...
- Abbruchkriterien (vgl. mit der POT Implementierung)

### 7.4.2 Komplexitätsanalyse

[?] liefert eine Komplexitätsanalyse für die Sinkhorn-Iterationen. Im Fall  $n = m$  sind für die Wahl  $\varepsilon = \frac{4 \log(n)}{\tau} \mathcal{O}(\|C\|_\infty^3 \log(n) \tau^{-3})$  Sinkhorn-Iterationen (inklusive eines Rundungsschrittes) notwendig, um eine zulässige Kopplung  $\hat{P} \in \Pi(\mathbf{a}, \mathbf{b})$  zu berechnen, die die Abschätzung  $\langle \hat{P}, C \rangle \leq L_C(\mathbf{a}, \mathbf{b} + \tau)$  zu berechnen. Somit liefert das Sinkhorn-Verfahren eine  $\tau$ -Approximation des nicht regularisierten Optimal Transport-Problems in  $\mathcal{O}(n^2 \log(n) \tau^{-3})$  Operationen. Gleichzeitig wird dort eine greedy Variante der Sinkhorn-Iterationen namens Greenkhorn präsentiert, die eine  $\tau$ -Approximation in  $\mathcal{O}(n^2 \tau^{-3})$  Operationen liefert. [?] verbessert diese auf  $\mathcal{O}(n^2 \tau^{-2})$  Operationen.

## 7.5 Gromov-Wasserstein-Divergenz

In den vorherigen Abschnitten hatten wir jeweils Wahrscheinlichkeitsmaße auf demselben metrischen Raum verglichen. In diesem Abschnitt wollen wir diesen Abstandsbegriff auf die Distanz zweier verschiedener metrischer Maßräume (mM) verallgemeinern. Dazu definieren wir die Gromov-Wasserstein-Distanz, die der Wasserstein-Distanz sehr ähnlich und eine Relaxierung der Hausdorff-Distanz ist.

### 7.5.1 Gromov-Wasserstein-Divergenz

[?] Wir definieren zunächst die beiden folgenden Distanzen:

**Definition 7.5.1** (Hausdorff-Distanz). *Sei  $\mathcal{X}$  ein metrischer Raum. Für  $X, Y \subset \mathcal{X}$  kompakt und nichtleer definieren wir die Hausdorff-Distanz  $d_H(X, Y)$  als*

$$d_H(X, Y) = \max\left\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y)\right\}. \quad (7.38)$$

Anschaulich haben zwei kompakte Teilmengen einen geringen Hausdorff-Abstand, wenn es zu jedem Element der einen Menge ein Element der anderen Menge gibt, zu dem dieses einen geringen Abstand hat. In Abbildung 5 ist die Hausdorff-Distanz zweier Mengen  $X$  und  $Y$  dargestellt.

Eine Teilmenge  $R \subset X \times Y$  ist eine Korrespondenz zwischen den Mengen  $X$  und  $Y$ , falls  $\pi_1(R) = X$  und  $\pi_2(R) = Y$ , wobei  $\pi_1 : X \times Y \rightarrow X$  und  $\pi_2 : X \times Y \rightarrow Y$  die kanonischen Projektionen sind. Sei  $\mathcal{R}(X, Y)$  die Menge aller Korrespondenzen zwischen  $X$  und  $Y$ .

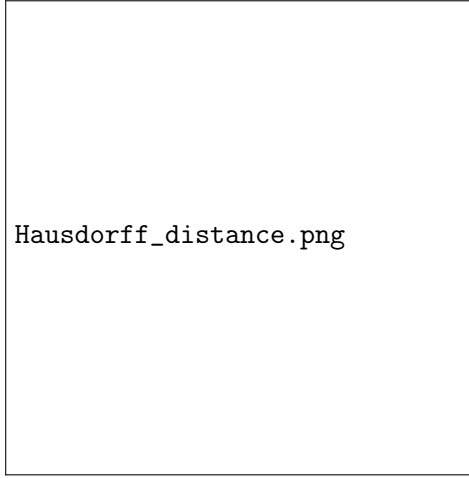
**Definition 7.5.2** (Gromov-Hausdorff-Distanz). *Für die kompakten, metrischen Räume  $(X, d_X)$  und  $(Y, d_Y)$  ist die Gromov-Hausdorff-Distanz definiert als*

$$d_{\mathcal{GH}}(X, Y) = \frac{1}{2} \inf_R \sup_{(x, y), (x', y') \in R} |d_X(x, x') - d_Y(y, y')|, \quad (7.39)$$

wobei  $R$  über ganz  $\mathcal{R}(X, Y)$  reicht.

Sei nun  $p \geq 1$ . Für zwei metrische Maßräume  $(X, d_X, \mu_X)$  und  $(Y, d_Y, \mu_Y)$  betrachten die Funktion

$$L(x, y, x', y') = |d_X(x, x') - d_Y(y, y')| \quad (7.40)$$



**Abbildung 5:** Visualisierung der Hausdorff-Distanz zwischen den Mengen  $X$  und  $Y$  in  $\mathbb{R}^2$ .

Quelle: By Rocchini - Own work, CC BY 3.0,  
<https://commons.wikimedia.org/w/index.php?curid=2918812>

und definieren die Gromov-Wasserstein-Distanz der Ordnung  $p$  [?]:

$$d_{\mathcal{GW},p}(\mu_X, \mu_Y) := \left( \inf_{\mu \in \Pi(\mu_X, \mu_Y)} J_p(\pi) \right)^{1/p}, \quad (7.41)$$

wobei

$$J_p(\pi) := \int_{X \times Y \times X \times Y} L(x, y, x', y')^p d\pi(x, y) d\pi(x', y') \quad (7.42)$$

gilt.

**Bemerkung 7.5.3.** *Mit dieser Definition erhalten wir nun eine relaxierte  $L^p$ -Version der Gromov-Hausdorff-Distanz Gleichung 7.39*

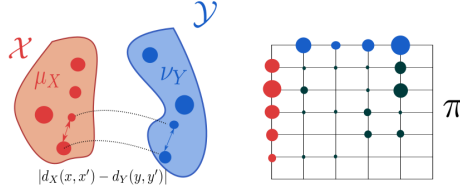
Die Gromov-Wasserstein Distanz definiert eine Metrik auf dem Raum aller metrischen Räume  $\mathcal{M}^{iso}$ . Mit dieser Definition wird der Vergleich von Maßen über unterschiedlichen Grundräumen möglich und damit auch ein Vergleich von Objekten jeglicher Art.

Wir zitieren einige Eigenschaften der Gromov-Wasserstein-Distanz:

**Theorem 7.5.4.** *[?, Thm. 5.1] Sei  $p \in [1, \infty]$  Dann gilt:*

- (a)  $d_{\mathcal{GW},p}$  definiert eine Metrik auf der Menge aller Isomorphieklassen von metrischen Maßräumen.
- (b) (Was passiert im Fall zweier Wahrscheinlichkeitsmaße auf demselben Raum?)  
 Sei  $(Z, d)$  ein kompakter metrischer Raum und  $\alpha$  und  $\beta$  zwei verschiedene Borel-Maße auf  $Z$ . Sei  $X = (Z, d, \alpha)$  und  $Y = (Z, d, \beta)$ . Dann gilt

$$d_{\mathcal{GW},p}(X, Y) \leq d_{\mathcal{W},p}^Z(\alpha, \beta). \quad (7.43)$$



**Abbildung 6:** Links: Die beiden metrischen Maßräume  $X$  und  $Y$ . Rechts: Zugehöriger Transportplan

Dieses Theorem zeigt nun, dass wir mit der Gromov-Wasserstein-Distanz einen Distanzbegriff erhalten, der invariant gegenüber Rotationen, Translationen und Permutationen ist [?].

Aufgrund genau dieser Inavrianz vermuten wir, dass sich dieser Distanzbegriff deutlich besser als eine Pixel-weise  $L_2$ -Distanz für ein CLustering auf der Grundlage von Heatmaps ist. Diese Invarianz resultiert daraus, dass wir eine Metrik auf den Isomorphie-Klassen metrischer Maßräume betrachten.

Im Folgenden werden wir die numerischen Verfahren zu Bestimmung der Gromov-Wasserstein-Distanz behandeln. Auch hier ermöglicht eine Entropie-regularisierte Version die einfache Berechnung einer Approximation wie im Fall der Wasserstein-Distanz.

Wir betrachten dazu wieder die diskreten Wahrscheinlichkeitsmaße  $\mu = \sum_{i=1}^n a_i \delta_{x_i} \in \mathcal{P}(X)$ ,  $\nu = \sum_{j=1}^m b_j \delta_{y_j} \in \mathcal{P}$  auf den Räumen  $(X, d_X)$  bzw.  $(Y, d_Y)$ . Mit  $C_1$  und  $C_2$  bezeichnen wir die Distanzmatrizen der paarweisen Distanzen aller Punkte innerhalb des entsprechenden Raumes, d.h. es gilt  $\forall(i, k) \in \{1, \dots, n\}^2 : C_1(i, k) = d_X(x_i, x_k)$  und  $\forall(j, l) \in \{1, \dots, m\}^2 : C_2(j, l) = d_Y(y_j, y_l)$ . Damit lautet das Gromov-Wasserstein Problem im diskreten Fall:

$$GW_p^p(C_1, C_2, \mathbf{a}, \mathbf{b}) = \min_{\pi \in \Pi(\mathbf{a}, \mathbf{b})} \sum_{i,j,k,l} |C_1(i, k) - C_2(j, l)| \pi_{i,j} \pi_{k,l} = \min_{\pi \in \Pi(\mathbf{a}, \mathbf{b})} \langle \mathbf{L}(C_1, C_2)^p \otimes \pi, \pi \rangle_{\mathcal{F}}, \quad (7.44)$$

wobei  $\langle \cdot, \cdot \rangle_{\mathcal{F}}$  das Matrix-Produkt  $\langle \mathbf{P}, \mathbf{Q} \rangle_{\mathcal{F}} = \text{tr}(\mathbf{Q}^\top \mathbf{P})$  bezeichnet und  $\mathbf{L}(C_1, C_2) = (|C_1(i, k) - C_2(j, l)|)_{i,j,k,l}$  und  $\otimes$  die Tensor-Matrix-Multiplikation ist.

Dieses Optimierungsproblem Gleichung 7.44 ist ein nicht-konvexes quadratisches Optimierungsproblem, dass im Allgemeinen NP-schwierig und ebenfalls schwer zu approximieren ist.

Für  $p = 2$  und damit  $\mathbf{L} = |\cdot|^2$  können wir Gleichung 7.44 schreiben als:

$$\min_{\pi \in \Pi(\mathbf{a}, \mathbf{b})} \text{tr}(\mathbf{c}_{C_1, C_2} \pi^\top) - 2\text{tr}(\mathbf{C}_1 \pi \mathbf{C}_2 \pi^\top), \quad (7.45)$$

wobei  $\text{tr}(\mathbf{c}_{C_1, C_2}) := f_1(C) \mathbf{a} \mathbf{1}_{N_2}^\top + \mathbf{1}_{N_1} b^\top f_2(\bar{C})^\top$  unabhängig von  $\pi$  ist. In Standardform erhalten wir

$$\min_{\pi \in \Pi(\mathbf{a}, \mathbf{b})} \mathbf{c}^\top \mathbf{x}(\pi) + \frac{1}{2} \mathbf{x}(\pi)^\top \mathbf{Q} \mathbf{x}(\pi), \quad (7.46)$$

wobei  $\mathbf{x}(\pi) = \text{vec}(\pi)$ ,  $\mathbf{c} = \text{vec}(\mathbf{c}_{C_1, C_2})$  und  $\mathbf{Q} = -4\mathbf{C}_1 \otimes_K \mathbf{C}_2$  und  $\otimes_K$  das

Kronecker-Matrix-Produkt für  $\mathbf{A} \in \mathbb{R}^{n \times m}$ ,  $\mathbf{B} \in \mathbb{R}^{p \times q}$ ,  $\mathbf{A} \otimes_K \mathbf{B} \in \mathbb{R}^{np \times mq}$  mit  $\mathbf{A} \otimes \mathbf{B} = (A_{i,j} B)_{i,j}$  gilt.

Gleichung 7.46 ist ein nicht konvexes quadratisches Optimierungsproblem, da die Hesse-Matrix  $\mathbf{Q}$  im Allgemeinen nicht positiv semi-definit ist (die Eigenwerte sind die Produkte der Eigenwerte der Matrizen  $\mathbf{C}_1$  und  $\mathbf{C}_2$ )

<http://people.csail.mit.edu/jsolomon/assets/gw.pdf>

Vernutzen wir nun auch hier einen Regularisierungs-Term basierend auf der Entropie erhalten wir das folgende Optimierungsproblem:

$$\min_{\pi \in \Pi(\alpha, \beta)} \langle \mathbf{L}(\mathbf{C}_1, \mathbf{C}_2)^p \otimes \pi, \pi \rangle_{\mathcal{F}} - \varepsilon H(\pi). \quad (7.47)$$

Dieses nicht-konvexe Optimierungsproblem können wir mithilfe eines projektiven Gradienten-Verfahrens lösen, bei dem sowohl für den Gradienten-Schritt als auch die Projektion die KL-Divergenz verwenden.

**Definition 7.5.5** (Tensor-Matrix-Multiplikation). Für einen Tensor  $\mathcal{L} = (\mathcal{L}_{i,j,k,l})_{i,j,k,l}$  und eine Matrix  $(T_{i,j})_{i,j}$  definieren wir die Tensor-Matrix-Multiplikation als

$$\mathcal{L} \otimes T := \left( \sum_{k,l} \mathcal{L}_{i,j,k,l} T_{k,l} \right)_{i,j}. \quad (7.48)$$

**Definition 7.5.6** (Entropie). Für  $T \in \mathbb{R}_+^{N \times N}$  definieren wir die Entropie als

$$H(T) := - \sum_{i,j=1}^N T_{i,j} (\log(T_{i,j}) - 1). \quad (7.49)$$

Warum wir GW-Distanz brauchen <https://arxiv.org/pdf/1811.02834.pdf>

Der Vergleich zwischen Ähnlichkeits- bzw. Distanzmatrizen ist schwierig, da diese die innere Struktur eines Datensatzes beschreiben, die unabhängig von Rotationen und Translationen ist. Es existiert keine kanonische Ordnung der Reihen und Spalten. Verallgemeinerung auf beliebige Matrizen  $\mathbf{C}$ , d.h. diese Distanzmatrizen müssen nicht notwendigerweise positiv sein und die Dreiecksungleichung erfüllen. Häufig verwendete Fehlerfunktionen sind die quadratische Fehlerfunktion  $L(a, b) = L_2(a, b) := \frac{1}{2}|a - b|^2$  und die Kullback-Leibler-Divergenz  $L(a, b) = KL(a|b) := a \log(a/b) - a + b$ .

Diese Definition der Gromov-Wasserstein-Distanz verallgemeinert die Version in [?], da nun beliebige Fehlerfunktionen betrachtet werden.

Mit der folgenden Proposition ergibt sich eine effiziente Berechnung von  $\mathcal{L}(\mathbf{C}, \bar{\mathbf{C}}) \otimes T$  für eine bestimmte Klasse von Verlustfunktionen  $L$ :

**Proposition 7.5.7.** Die Verlustfunktion  $L$  lasse sich schreiben als

$$L(a, b) = f_1(a) + f_2(b) - h_1(a)h_2(b) \quad (7.50)$$

für  $f_1, f_2, h_1, h_2 : \mathbb{R} \rightarrow \mathbb{R}$ . Dann gilt für  $T \in \mathcal{C}_{p,q}$ :

$$\mathcal{L}(\mathbf{C}, \bar{\mathbf{C}}) \otimes T = c_{\mathbf{C}, \bar{\mathbf{C}}} - h_1(\mathbf{C}) T h_2(\bar{\mathbf{C}})^T, \quad (7.51)$$

wobei  $c_{\mathbf{C}, \bar{\mathbf{C}}} := f_1(\mathbf{C}) p \mathbf{1}_{N_2}^T + \mathbf{1}_{N_1} q^T f_2(\bar{\mathbf{C}})^T$  unabhängig von  $T$  ist.

*Beweis.* Aufgrund von Gleichung 7.50 gilt nach der Tensor-Matrix-Multiplikation Gleichung 7.48 die Zerlegung  $\mathcal{L}(C, \bar{C}) \otimes T = A + B + C$  mit

$$\begin{aligned} A_{i,j} &= \sum_k f_1(C_{i,k}) \sum_l T_{k,l} = (f_1(C)(T\mathbf{1}))_i, \\ B_{i,j} &= \sum_l f_2(\bar{C}_{j,l}) \sum_k T_{k,l} = (f_2(\bar{C})(T^\top \mathbf{1}))_j, \\ C_{i,j} &= \sum_k h_1(C_{i,k}) \sum_l h_2(\bar{C}_{j,l}) T_{k,l}. \end{aligned}$$

Dies ist äquivalent zu  $(h_1(C))(h_1(\bar{C}T^\top)^\top)_{i,j}$ .  
(? Wie folgt daraus die Behauptung)  $\square$

**Bemerkung 7.5.8** (Verbesserte Komplexität). *Mit dem Resultat in Theorem 7.5.7 können wir  $\mathcal{L}(C, \bar{C}) \otimes T$  effizient in der Größenordnung  $\mathcal{O}(N_1^2 N_2 + N_2^2 N_1)$  mit ausschließlich Matrix/Matrix-Multiplikationen berechnen im Unterschied zur Komplexität von  $\mathcal{O}(N_1^2 N_2^2)$  für die Implementierung von Gleichung 7.48.*

**Bemerkung 7.5.9** (Spezialfälle). *Im Fall  $L = L_2$  ist die Bedingung Gleichung 7.50 für die Funktionen  $f_1(a) = a^2, f_2(b) = b^2, h_1(a) = a$  und  $h_2(b) = 2b$  erfüllt. Für  $L = KL$  sind die Funktionen  $f_1(a) = a \log(a) - a, f_2(b) = b, h_1(a) = a$  und  $h_2(b) = \log(b)$  notwendig.*

Wir erhalten damit ein nicht-konvexes Optimierungsproblem. Für dessen Lösung benutzen wir ein projiziertes Gradienten-Verfahren, bei dem sowohl die Schrittweite als auch die Projektion bezüglich der KL-Metrik berechnet werden.

Die Iterationen sind gegeben durch

$$T \leftarrow \text{Proj}_{\mathcal{C}_{p,q}}^{KL} \left( T \odot e^{-\tau(\nabla \epsilon_{C,\bar{C}}(T) - \epsilon H(T))} \right), \quad (7.52)$$

wobei die Schrittweite  $\tau > 0$  und die KL-Projektion einer beliebigen Matrix  $K$  gegeben ist durch:

$$\text{Proj}_{\mathcal{C}_{p,q}}^{KL}(K) := \arg \min_{T' \in \mathcal{C}_{p,q}} KL(T'|K). \quad (7.53)$$

**Proposition 7.5.10.** *Für den Fall  $\tau = 1/\epsilon$  erhalten wir die Iterationsvorschrift*

$$T \leftarrow \mathcal{T}(\mathcal{L}(C, \bar{C}) \otimes T, p, q). \quad (7.54)$$

*Beweis.* Nach [?] ist die Projektion in 7.52 gegeben durch die Lösung des regularisierten Transportproblems 7.18 und ist damit gegeben durch:

$$\text{Proj}_{\mathcal{C}_{p,q}}^{KL}(K) = \mathcal{T}(-\epsilon \log(K), p, q) \quad (7.55)$$

(? Wo steht das genau in dem Paper?) Es gilt außerdem

$$\nabla \epsilon_{C,\bar{C}}(T) - \epsilon H(T) = \mathcal{L}(C, \bar{C}) \otimes T + \epsilon \log(T). \quad (7.56)$$

Durch Umordnen der Terme in Gleichung 7.52 erhalten wir für  $\tau\epsilon = 1$  die angegebene Vorschrift.  $\square$



**Bemerkung 7.5.11.** Die Iterationsvorschrift 7.54 definiert einen einfachen Algorithmus, der in jedem Update von  $T$  eine Sinkhorn-Projektion benötigt.

**Bemerkung 7.5.12** (Konvergenz). Die Iterationen Gleichung 7.52 konvergieren nach [?] für  $\tau < \tau_{\max}$ . Im Allgemeinen gilt jedoch  $1/\varepsilon < \tau_{\max}$  jedoch nicht, womit die Wahl der Schrittweite  $\tau = 1$  in Theorem 7.5.10 nicht durch die Theorie abgedeckt ist. Laut [?] konvergiert die Iteration mit  $\tau = 1/\varepsilon$  jedoch in der Praxis.

Für den Fall  $L = L_2$  ergibt sich der „Softassign quadratic assignment algorithm“, [?], für welchen ein Konvergenzresultat im Fall eines konvexen Problems existiert. Da wir in unserem Fall nur positive symmetrische Matrizen  $C, C_s$  betrachten ist hier die Konvergenz gesichert.

**Bemerkung 7.5.13.** Für gewöhnlich wählt man die Schrittweite  $\tau$  im Gradientenverfahren nicht zu groß, um Konvergenz des Verfahrens zu erhalten. Da die Schrittweite  $\tau$  hier jedoch antiproportional zum Regularisierungs-Parameter  $\varepsilon$  ist, gibt es einen Trade-off zwischen Konvergenz des Verfahrens und einer Unschärfe in der optimalen Lösung.

**Bemerkung 7.5.14** (Wahl von  $\varepsilon$ ). In [?] werden verschiedene Werte für  $\varepsilon$  angegeben. Diese sind  $\varepsilon = 0.02, 0.1, 1.0$

Im zugehörigen Beispiel<sup>27</sup> von POT wird  $\varepsilon = 0.0005$  verwendet.

Für  $\varepsilon$  klein werden die Ergebnisse besser während der Rechenaufwand steigt, welche Resultate existieren bezüglich der Approximationsgüte abhängig von  $\varepsilon$ ?

## 7.5.2 Gromov-Wasserstein Baryzentren

In diesem Kapitel wollen wir uns mit dem "Mittelwert" befassen, der elementar für das kmeans-Clustering ist. Auch hier soll der Mittelwert auf die abstrakte Ebene von gewichteten Distanzmatrizen angehoben werden. Dazu definieren wir im Folgenden sogenannte Wasserstein-Baryzentren und folgen [?] für die Lösung des entstehenden Optimierungsproblems. Gute Einführung<sup>28</sup>

**Definition 7.5.15** (Gromov-Wasserstein Baryzentrum). Seien die gewichteten Distanzmatrizen  $(C_s)_{s=1}^S$ , mit  $C_s \in \mathbb{R}^{N_s \times N_s}$  und den zugehörigen Histogrammen  $(p_s)_s$  gegeben.

Dann ist das Gromov-Wasserstein Baryzentrum definiert durch

$$\min_{C \in \mathbb{R}^{N \times N}} \sum_s \lambda_s \text{GW}_\varepsilon(C, C_s, p, p_s). \quad (7.57)$$

Existenz und Eindeutigkeit von Baryzentren: siehe [?].

**Bemerkung 7.5.16** (Darstellung des Baryzentrums). Um das berechnete Baryzentrum  $C$  wieder zu visualisieren, kann  $C$  als Distanzmatrix wieder in den zwei-dimensionalen Raum eingebettet werden. Dies kann beispielsweise durch Multi-dimensionale Skalierung erreicht werden<sup>29</sup>.

<sup>27</sup>[https://pythonot.github.io/auto\\_examples/gromov/plot\\_gromov.html](https://pythonot.github.io/auto_examples/gromov/plot_gromov.html)

<sup>28</sup><https://hal.archives-ouvertes.fr/hal-00637399/document>

<sup>29</sup>[https://pythonot.github.io/auto\\_examples/gromov/plot\\_gromov\\_barycenter.html](https://pythonot.github.io/auto_examples/gromov/plot_gromov_barycenter.html)

**Bemerkung 7.5.17.** Wir gehen im Folgenden davon aus, dass sowohl die Histogramme  $(p_s)_s$  als auch das Histogramm  $p$  bekannt sind. Die Größe  $(N, N)$  des gesuchten Bary-Zentrums muss ebenfalls vorher festgelegt werden. Eine Erweiterung auf den Fall, dass auch  $p$  unbekannt sein sollte und damit als Optimierungsparameter aufgefasst wird, ist leicht möglich [?].

Wir können das Bary-Zentrum mithilfe von Kopplungen umformulieren als

$$\min_{C, (T_s)_s} \sum_s \lambda_s (\varepsilon_{C, C_s}(T_s) - \varepsilon H(T_s)), \quad (7.58)$$

unter den Nebenbedingungen:  $\forall s : T_s \in \mathcal{C}_{p, p_s} \subset \mathbb{R}_+^{N \times N_s}$ . Für den Fall, dass  $L$  bezüglich der ersten Variable konvex ist, ist dieses Problem konvex bezüglich  $C$ . Bezüglich  $(T_s)_s$  ist diese Problem quadratisch aber nicht notwendigerweise positiv.

Das Problem Gleichung 7.58 kann durch eine Block-Koordinaten-Relaxierung gelöst werden. Dabei wird iterativ abwechselnd bezüglich den Kopplungen  $(T_s)_s$  und der Metrik  $C$  minimiert.

**Minimierung bezüglich  $(T_s)_s$ .** Anhand der Umformulierung Gleichung 7.58 sehen wir, dass das Optimierungsproblem Gleichung 7.57 bezüglich  $(T_s)_s$  in  $S$  (?viele) unabhängige  $GW_\varepsilon$ -Optimierungen

$$\forall s : \min_{T_s \in \mathcal{C}_{p, p_s}} \mathcal{E}_{C, C_s}(T_s) - \varepsilon H(T_s) \quad (7.59)$$

zerfällt, die jeweils wie in Theorem 7.5.10 angegeben gelöst werden können.

**Minimierung bezüglich  $C$ .** Sei  $(T_s)_s$  gegeben. Dann lautet, die Minimierung bezüglich  $C$ :

$$\min_C \sum_s \lambda_s \langle \mathcal{L}(C, C_s) \otimes T, T \rangle. \quad (7.60)$$

Mit der folgenden Proposition erhalten wir für eine Klasse von Verlustfunktionen  $L$  eine Lösung in geschlossener Form.

**Proposition 7.5.18.** Sei  $L$  eine Verlustfunktion, die die Bedingung Gleichung 7.50 erfüllt. Sei  $f'_1/h'_1$  invertierbar.

Dann lässt sich die Lösung zu Gleichung 7.60 schreiben als:

$$C = \left( \frac{f'_1}{h'_1} \right)^{-1} \left( \frac{\sum_s \lambda_s T_s^\top h_2(C_s) T_s}{pp^\top} \right), \quad (7.61)$$

wobei die Normalisierung  $\lambda_s = 1$  gilt.

*Beweis.* Nach Theorem 7.5.7 können wir das zu minimierende Funtional schreiben als

$$\sum_s \lambda_s \langle f_1(C) p \mathbf{1}^\top + \mathbf{1} p_s^\top f_2(C_s) - h_1(C) T_s h_2(C_s)^\top, T_s \rangle. \quad (7.62)$$

Die Optimalitätsbedingung erster Ordnung lautet folglich

$$f'_1(C) \odot pp^\top = h'_1(C) \odot \sum_s \lambda_s T_s h_2(C_s) T_s^\top. \quad (7.63)$$

Durh Umstellen der Gleichung erhalten wir die angegebene Form.  $\square$

Anhand von Gleichung 7.61 wird die folgende Interpretation deutlich/möglich: Für jedes  $s \in S$  ist  $T_s^\top h_2(C_s)T_s$  eine wiedereingeordnete Matrix, wobei  $T_s$  als Optimal Transport-Kopplung von Zeilen und Spalten der Distanzmatrix  $C_s$  fungiert. Über diese Matrizen wird anschließend gemittelt, wobei die Art der Mittelung von der Verlustfunktion  $L$  abhängig ist.

Für den Fall  $L = L_2$  wird aus dem Update Gleichung 7.61 die folgende Vorschrift:

$$C \leftarrow \frac{1}{pp^\top} \sum_s \lambda_s T_s^\top C_s T_s. \quad (7.64)$$

**Proposition 7.5.19.** *Sei  $L = L_2$  und  $(C_s)_s$  positiv semi-definit f.a.  $s \in S$ . Dann sind die Iterierten  $C$  ebenfalls positiv semi-definit.*

*Beweis.* Gleichung 7.64 zeigt, dass das Update aus einer Bildung des Mittelwertes der Matrizen  $(\text{diag}(1/p)T_s^\top C_s T_s \text{diag}(1/p))_s$  besteht. Diese sind alle positiv semi-definit, da die  $(C_s)_s$  nach Voraussetzung positiv semi-definit sind.  $\square$

Für den Fall  $L = KL$  ergibt sich die folgende Verfahrensvorschrift:

$$C \leftarrow \left( \frac{1}{pp^\top} \sum_s \lambda_s T_s^\top \log(C_s) T_s \right). \quad (7.65)$$

---

**Algorithm 1** Berechnung der  $GW_{\varepsilon}$  Baryzentren

---

**Input:**  $(C_s, p_s), p$ .

**Output:**  $C$ .

Initialize  $C$ .

**repeat**

  // minimize over  $(T_s)_s$

**for**  $s = 1$  **to**  $S$  **do**

    Initialize  $T_s$ .

**repeat**

      // Compute  $c_s = \mathcal{L}(C, C_s) \otimes T_s$  using Gleichung 7.51

$c_s \leftarrow f_1(C) + f_2(C_s)^\top - h_1(C)T_s h_2(C_s)^\top$

      // Sinkhorn iterations to compute  $\mathcal{T}(c_s, p, q)$

      Initialize  $a \leftarrow \mathbf{1}$ , set  $K \leftarrow e^{-c_s/\varepsilon}$ .

**repeat**

$a \leftarrow \frac{p}{Kb}, b \leftarrow \frac{q}{K^\top a}$ .

**until** convergence

$T_s \leftarrow \text{diag}(a)K \text{diag}(b)$ .

**until** convergence

**end for**

  // Minimize over  $C$

$C \leftarrow \left( \frac{f'_1}{h'_1} \right)^{-1} \left( \frac{\sum_s \lambda_s T_s^\top h_2(C_s) T_s}{pp^\top} \right)$

**until** convergence

---

**Pseudocode.**

Häufige Verwendungen(s. Einführung hier <sup>30</sup>). zB. Shape interpolation

Wir haben in diesem Kapitel gesehen, wie mithilfe des Wasserstein-Baryzentums eine Art Mittelwert für Wahrscheinlichkeitsverteilungen definiert werden kann. Algorithm 1 details the steps of the optimization technique. Note that it makes use of three nested iterations: (i) blockwise coordinate descent on  $(T_s)$ s and  $C$ , (ii) projected gradient descent on each  $T_s$ , and (iii) Sinkhorn iterations to compute the projection.

**7.5.3 Komplexitätsanalyse****7.5.4 Implementierung**

---

**Algorithm 2** Sinkhorn-Algorithmus

---

**Input:**  $\mathbf{a}, \mathbf{b}, \mathbf{C}, \varepsilon > 0$

**Output:** Optimal Coupling  $P^*$ .

Initialize  $\mathbf{u}^{(0)}, \mathbf{v}^{(0)} = \mathbf{1}, \mathbf{K} = \exp(-\frac{\mathbf{C}}{\varepsilon})$ .

**repeat**

$\mathbf{u}^{(i)} = \mathbf{a} \oslash \mathbf{K}^\top \mathbf{v}^{(-1)}$

$\mathbf{v}^{(i)} = \mathbf{b} \oslash \mathbf{K} \mathbf{u}^{(-1)}$

**until** convergence

---

??

## 8 (Numerische) Ergebnisse/Vergleich mit anderen Verfahren

**8.0.1 Standard Poisoning-Angriffe****8.1 Clustering auf den Rohdaten**

Für das Clustering auf den Rohdaten bei einem Anteil von 15 Prozent und einer Stickergröße von 3x3 erhalten wir die Tabelle 2 in dargestellten Werte unter Verwendung der L2-Distanz. Dazu werden die Bilder als Graustufenbild eingelesen und anschließend in der Dimension auf 10 reduziert.

Wir konnten beobachten, dass die Qualität des angegriffenen Netzwerkes auf unkorruptierten Daten für alle durchgeführten Angriffe über 96 Prozent beträgt. Deshalb werden wir diese Kennzahl als Qualität des Angriffs nicht jeweils zusätzlich angeben.

**Bemerkung 8.1.1** (Die Fälle mit einer AER uner 100 Prozent). *Besonders interessant sind auch genau die Fälle, bei denen die Hintertür im Netzwerk so implementiert ist, dass nicht alle präsentierten korruptierten samples gewollt falsch klassifiziert werden. Was erwarten wir hier von unserem Detektionsalgorithmus? Wenn wir das Gromov-Wasserstein-Clustering auf den Heatmaps durchführen, können wir ja*

---

<sup>30</sup><https://arxiv.org/pdf/2102.01752.pdf>

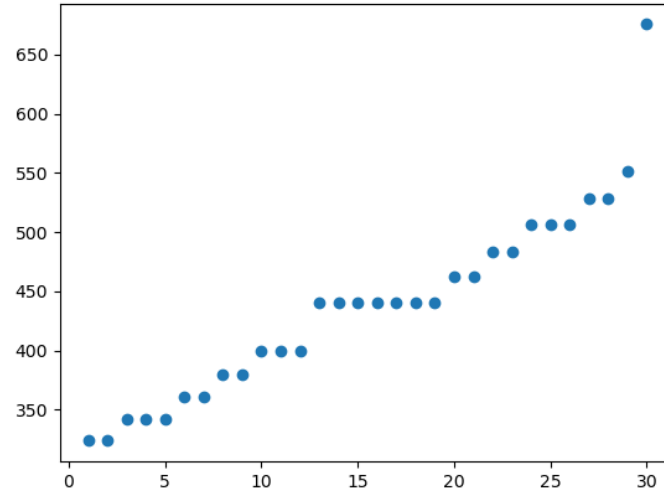
	PCA	FastICA
ACC	65.49	63.13
TPR	51.54	31.99
TNR	72.36	78.48

**Tabelle 2:** Auswertung des Clusterings auf den rohen Bilddaten bei 33 Prozent korruptierten Daten mit einem Sticker der Größe 3x3. Alle numerischen Werte sind in Prozent angegeben.

Seitenlänge s des Triggers	Prozentualer Anteil	AER	DR (HC)			DR (AC,PCA)			Clustering HC
			ACC	TPR	TNR	ACC	TPR	TNR	
s=2	0.000625	0.01333333							
	0.125	0.356							
	0.25	0.576							
	0.5	0.99867							
	1.00	0.92							
true	2.00	100.00				23.02	4.43	32.18	
true	5.00	100.00				47.44	10.34	49.39	
true	10.00	100.00	99.72	100.00	99.69	67.21	93.99	64.24	
hctrue true	15.00	100.00	100.00	100.00	100.00	87.53	97.59	85.76	iter:4(tn, fp, fn, tp): 1645,5,0,183
(eps5e-4)	0.33	1.0	98.17	96.98	98.91	99.72	99.14	100.00	(tn, fp, fn, tp): 1650,0,0,291 it3
eps0.02			100.00	100.00	100.00				(tn, fp, fn, tp): 1632,18,27,786 it5
									(tn, fp, fn, tp): 1650,0,0,813 it3
s=3	0.05	1.0							
	0.15	1.0							
s=1	33.0	100.0	100.00	100.00	100.00	98.78	97.66	99.33	
	15.0	100.0	100.00	100.00	100.00	96.29	79.04	99.33	
	10.00	100.0	100.00	100.00	100.00	71.47	87.43	69.7	
	5.00	100.0	100.00	100.00	100.00	73.4	100.00	72.0	
	2.00	100.0	100.00	100.00	100.00	58.31	100.00	57.45	
	1.00	13.87	60.77	47.06	60.91	56.99	100.00	56.55	

**Tabelle 3:** Qualität der Detektion unterschiedlich starker Angriffe mithilfe von LRP-Clustering(?) und Gromov-Wasserstein-Distanzen. Das LRP-Clustering für s=1. wurde mit eps=0.1 durchgeführt

*nur die Informationen nutzen, die das Netzwerk über die LRP liefert. Diese ist aber schon fehlerhaft/mangelhaft. Vielleicht sollte man hier dann das Clustering wieder auf den rohen Bilddaten durchführen? Welche Pixel nimmt man hier dann? Alle? Wichtig ist hier auch der Vergleich mit dem Clustering bezüglich der L2 Distanz auf den EingabeBilder.*

**Spektralanalyse:**Absolute values of first 30 eigenvalues of  $L_{sym}$  with 10-nearest neighbours**Abbildung 7:** Ergebnis des spektralen Clusterings unter Verwendung der Euklidischen Distanz und  $k=10$  Nachbarn

**Clustering(euklidisch):**  $(tn, fp, fn, tp) = (1650, 0, 386, 427)$

**Clustering(GWD):**

**Bemerkung 8.1.2.** Für größere Netzwerke ist es einfacher, erfolgreiche Angriffe zu implementieren, auch schon mit weniger korruptierten Daten.

Vermutung: Die Detektion mittels Clustering funktioniert für eine größere Anzahl an korruptierten Daten besser. Wenn wir also nur perfekte Angriffe verteidigen wollen, d.h.  $AER=100.00$  funktioniert das bei kleineren Netzwerken besser.

**8.1.1 Label-konsistente Poisoning-Angriffe**

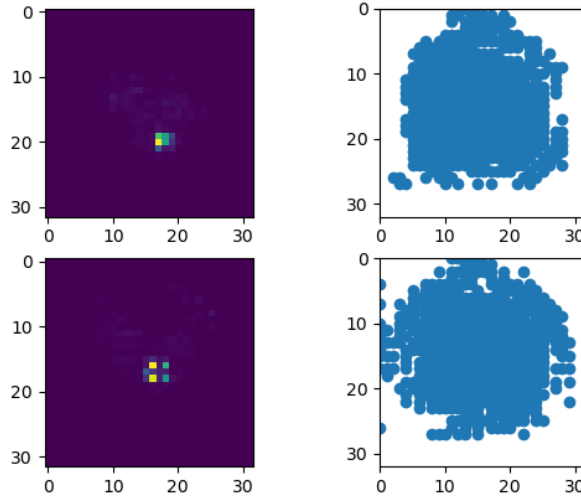
**Bemerkung 8.1.3.** Für einen einzelnen Amplitudensticker mit  $d=10$  ist das eigentlich identisch zu dem Angriff mit dem Sticker

```

0.7 0.86111111 0.85066667 0.81111111 0.66666667 nan 0.42 0.72888889 0.60888889
0.77083333 0.40757576 0.46666667 0.69855072 0.88472222 0.53333333 0.74285714
0.39333333 0.67222222 0.75128205 0.11666667 0.58888889 0.32222222 0.425 0.18
0.44444444 0.44375 0.8 0.71666667 0.56 0.86666667 0.11333333 0.3 0.71666667 0.5047619
0.23333333 0.44358974 0.63333333 0.58333333 0.46666667 0.01111111 0.5 0.36666667
0.67777778 CLP_amplitudesticker4_pp33_eps1200_d10amp32 Jetzt dasselbe noch
für nur !!!einen!!! Sticker mit derselben Amplitude am selben Ort 'CLP_amplitudesticker_pp33_eps1200_d10amp32'
[0. 0.00833333 0.00133333 0.03555556 0.01666667 nan 0. 0. 0.00222222 0. 0. 0.
0.05362319 0.00138889 0.0037037 0. 0. 0.025 0. 0. 0. 0. 0. 0. 0. 0. 0.05 0. 0.
0. 0. 0. 0. 0.00512821 0. 0. 0. 0. 0. 0. 0. ]

```

Heatmap und erzeugte Punktwolke für threshold = 0.99



**Abbildung 8:** Auswahl der relevantesten Pixel (bis zu 99% der Gesamtmasse) zweier Heatmaps

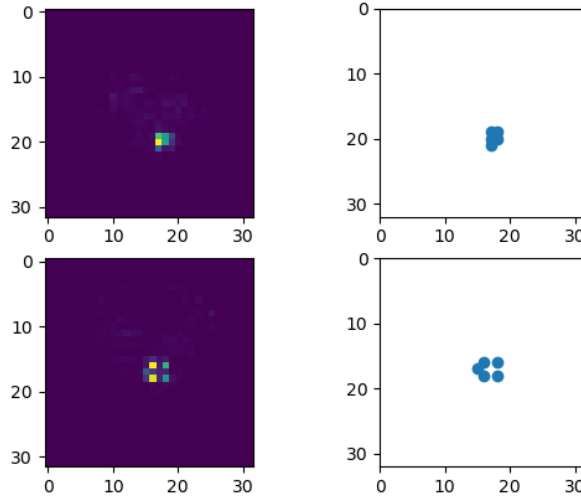
Dann noch mit amp=16 wieder 4 sticker mit d=10  $CLP_{amplitude}sticker4_p33_{eps1200_d10amp16}$   
 0.11666667 0.23055556 0.26 0.55333333 0.43333333 nan 0.23333333 0.28444444 0.11555556  
 0.07708333 0.15 0.0952381 0.4173913 0.41527778 0.40740741 0.45238095 0.01333333  
 0.35833333 0.29487179 0. 0. 0.04444444 0.05 0.01333333 0.4 0.29791667 0.13888889  
 0.23333333 0.11333333 0.3 0.06 0.02962963 0.11666667 0.65714286 0.475 0.71794872  
 0.50833333 0.43333333 0.35652174 0.45555556 0.27777778 0. 0.51111111 Accuracy  
 on test Dataset: 0.958

Jetzt:  $CLP_{amplitude}sticker4_p33_{eps1200_d10amp255}$  0.71666667 0.78333333 0.80266667  
 0.99333333 0.9469697 nan 0.96 0.97555556 0.98444444 1. 1. 0.25714286 0.98985507  
 0.86111111 0.94444444 0.99047619 0.91333333 0.51666667 0.84871795 0.86666667  
 0.52222222 0.57777778 0.75 0.67333333 0.95555556 0.725 0.98888889 0.88333333  
 0.45333333 0.82222222 0.81333333 1. 0.98333333 0.4952381 0.26666667 0.92820513  
 0.63333333 0.81666667 0.77681159 0.33333333 0.74444444 0.91666667 0.83333333  
 Accuracy on test Dataset: 0.959

$CLP_{amplitude}sticker4_p33_{dist10_a mp64}$  eps300 [0.8 0.98888889 0.99733333 1.  
 1. nan 1. 1. 0.99555556 0.97916667 0.99545455 0.8047619 1. 0.99861111 0.92222222  
 1. 0.99333333 0.98888889 0.98717949 0.93333333 0.86666667 0.93333333 0.975 0.83333333  
 0.96666667 0.93958333 1. 1. 0.98 1. 0.98 0.97777778 1. 0.92380952 0.55 0.93076923  
 0.70833333 0.85 0.82173913 1. 0.8 1. 1.

$CLP_{amplitude}sticker4_p33_{dist10_a mp32}$  0.7 0.86111111 0.85066667 0.81111111  
 0.66666667 nan 0.42 0.72888889 0.60888889 0.77083333 0.40757576 0.46666667 0.69855072  
 0.88472222 0.53333333 0.74285714 0.39333333 0.67222222 0.75128205 0.11666667  
 0.58888889 0.32222222 0.425 0.18 0.44444444 0.44375 0.8 0.71666667 0.56 0.86666667  
 0.11333333 0.3 0.71666667 0.5047619 0.23333333 0.44358974 0.63333333 0.58333333  
 0.46666667 0.01111111 0.5 0.36666667 0.67777778 Accuracy on test Dataset: 0.959

Heatmap und erzeugte Punktwolke für threshold = 0.5



**Abbildung 9:** Auswahl der relevantesten Pixel (bis zu 50% der Gesamtmasse) zweier Heatmaps

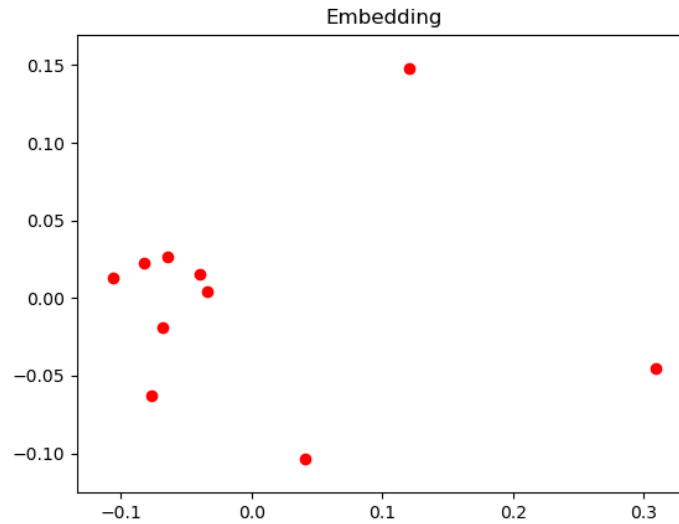
*CLP<sub>a</sub>mplitudesticker4<sub>p</sub>33<sub>d</sub>ist10<sub>a</sub>mp16* 0.11666667 0.23055556 0.26 0.55333333  
 0.43333333 nan 0.23333333 0.28444444 0.11555556 0.07708333 0.15 0.0952381 0.4173913  
 0.41527778 0.40740741 0.45238095 0.01333333 0.35833333 0.29487179 0. 0. 0.04444444  
 0.05 0.01333333 0.4 0.29791667 0.13888889 0.23333333 0.11333333 0.3 0.06 0.02962963  
 0.11666667 0.65714286 0.475 0.71794872 0.50833333 0.43333333 0.35652174 0.45555556  
 0.27777778 0. 0.51111111] Performance of poisoned net on unpoisoned training data:  
 loss on test dataset: 0.19637071904851583 Accuracy on test Dataset: 0.958

In Abbildung 11 sind die Angriffserfolgsraten pro Klasse im Fall von 33% korruptierten Daten und dem Amplitudensticker in jeder der 4 Bildecken mit dem Abstand von 10 Pixeln zum Rand dargestellt. In beiden Fällen geben wir keine AER für die Klasse 6 an, über die der Angriff stattfindet. Die mittlere Angriffserfolgsrate beträgt für  $amp = 255$  79.15%. In mehreren Klassen wird eine AER von 100% erreicht.

Für  $amp = 64$  fällt der Angriff mit einer mAER von 48.17% deutlich schwächer aus. Die maximal erreichte AER beträgt 95.33% in Klasse 10. Die minimalen AER sind 25% bzw. 0.83%. Eine weitere Reduktion der Amplitude auf  $amp = 32$  führt zu einer mAER von 6.55% und einer maximalen AER von 33%.

Für  $d=0$ ,  $amplitude=64$  und  $eps=1200$  ergibt sich kein erfolgreicher Angriff, dabei war fast alles 0, die Trigger im Testdatensatz waren auch auf 64 Jetzt für  $d=10$ , auch hier gilt  $amp=64$  im Testdatensatz:Ergebnis:





**Abbildung 10:** Einbettung des Baryzentrums mithilfe von Multidimensionaler Skalierung(MDS) bei der Wahl von 99% der Gesamtmasse

## 8.2 Activation Clustering

### 8.3 Räumliche Transformationen

- ASR ist sehr stark vom Ort des Triggers abhängig.
- Ort des Triggers kann nicht direkt geändert werden.
- Benutze Transformationen(Flipping, Scaling), um den Trigger wirkungslos zu machen.
- Somit kann die ASR während der Inferenz verringert werden. Es lässt sich aber keine Aussage darüber treffen, ob ein Angriff vorliegt

Seitenlänge des Triggers	Prozentualer Anteil	AER	GUD	num. korruptierte Daten
s=2	0.000625	0.01333333	0.962	1
	0.00125	0.356	0.964	2
	0.0025	0.576	0.97	4
	0.005	0.99867	0.962	8
	0.01	0.92	0.962	17
	0.02	1.0	0.964	34
	0.10	1.0	0.968	291
	0.15	1.0	0.968	436
	0.33	1.0	0.968	813
s=3	?	?	?	?
	0.00125	34.26	96.2	2
	0.0025	85.07	96.5	4
	0.005	1.0	96.9	8
	0.01	1.0	0.962	17
	0.05	1.0	0.966	87
	0.15	1.0	0.961	
s=1	0.33	1.0	0.966	813

**Tabelle 4:** Qualität der Angriffe auf das Inception v3-Netz mit Stickern Seitenlänge 2 und 3 Pixel bei unterschiedlich großen Anteilen an korruptierten Daten

## 9 Weitere mögliche Schritte

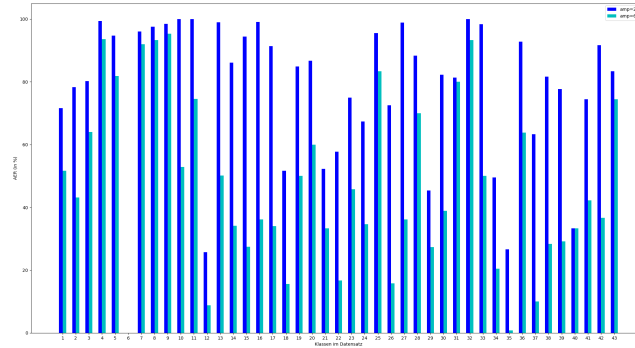
- Untersuchung der Detektionsqualität in Abhängigkeit von  $\varepsilon$
- Automatische Platzierung des Auslösers an fest gewählter Position auf dem Verkehrsschild anstatt zufälligem Platzieren in einem Fenster mit vorher festgelegter Größe. In [?] wird Faster-RCNN (F-RCNN) zur Klassifikation des LISA-Datensatzes<sup>31</sup> benutzt. Es ist die Aufgabe, die Verkehrsschilder in die 3 Superklassen Stoppschild, Geschwindigkeitsbegrenzung und Warnschild einzuteilen. Der Datensatz enthält zudem die BoundingBoxen, sodass der Auslöser genauer angebracht werden kann.
- Verbesserte Version der Layer-wise Relevance Propagation
- Untersuchung anderer Verfahren, die die Interpretierbarkeit ermöglichen, beispielsweise: VisualBackProp: efficient visualization of CNNs<sup>32</sup>
- Vergleich mit Cifar-10/Cifar-100 Datensatz<sup>3334</sup>

<sup>31</sup><http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html>

<sup>32</sup><https://arxiv.org/abs/1611.05418>

<sup>33</sup><https://www.cs.toronto.edu/~kriz/cifar.html>

<sup>34</sup><https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>



**Abbildung 11:** Angriffserfolgsrate pro Klasse bei Clean-Label-Poisoning-Attacks für verschiedene Werte  $amp$  des Amplitudenstickers in vierfacher Ausführung bei Abstand  $d = 10$  zum Rand

## 10 Zusammenfassung und Ausblick

Beobachtung: Je größer die Netzwerke sind, desto leichter lassen sich Poisoning-Angriffe realisieren.

## A Verwendete Netzwerke

### A.1 Net

Aufbau dieses Netzwerkes: 1. Inception-Modul 2. [pool1, batchConv1, pool2, batchConv2, pool3, batchConv3, pool4] 3. Drei Lineare Schichten mit ReLu und Dropout dazwischen

Im Unterschied zum offiziellen Inception Netz(v1v2v3) gibt es in dieser vereinfachten Version keinen `stem` aus convs, es geht direkt mit InceptionA los.

Wie ähnlich sind sich InceptionA(hier) und das offizielle InceptionA-Modul?

## B Parameter für Training und Einlesen der Daten

Die in [?] gewählten Parameter wären ein guter Ausgangspunkt.

Für das Einlesen der Daten benutzen wir, sofern nicht weiter angegeben die folgenden Augmentierungen:

```

1  __train_transform = transforms.Compose(
2  [
3  transforms.RandomResizedCrop((image_size, image_size),
4  scale=(0.6, 1.0)),
5  transforms.RandomRotation(degrees=15),
6  transforms.ColorJitter(brightness=0.1, contrast=0.1,
7  saturation=0.1, hue=0.1),
8  transforms.RandomAffine(15),
9  transforms.RandomGrayscale(),
10 transforms.Normalize( mean=[0.485, 0.456, 0.406],
11 std=[0.229, 0.224, 0.225]),
12 transforms.ToTensor()
13 ]
14
```

**Listing 4:** Augmentierung beim Einlesen der Daten

Die Werte von mean und std variieren für alle ausgeführten Poisoning-Angriffe. Anstatt beide jedes Mal erneut zu berechnen, verwenden wir die von pytorch angegebenen Werte <sup>35</sup>, die für die vor-trainierten Modelle empfohlen werden und auf dem Datensatz ImageNet<sup>36</sup> basieren.

Wir trainieren die Netzwerke über maximal 100 Epochen und benutzen *early stopping* mit einer *patience* = 20. Die verwendete Implementierung ist eine modifizierte Version von Bjarte Mehus Sunde <sup>37</sup>, die wiederum auf PyTorch Ignite<sup>38</sup> basiert.

## C Einlesen der Daten bei AC

Ohne Transformationen, wie den Testdatensatz.

<sup>35</sup><https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198>

<sup>36</sup><https://image-net.org/>

<sup>37</sup><https://github.com/Bjarten/early-stopping-pytorch>

<sup>38</sup>[https://github.com/pytorch/ignite/blob/master/ignite/handlers/early\\_stopping.pyt](https://github.com/pytorch/ignite/blob/master/ignite/handlers/early_stopping.pyt)

## D Parameter für die ausgeführten Angriffe

Für das projizierte Gradientenverfahren benutzen wir 10 Iterationen und eine Schrittweite von 0.015.

**Label-konsistente Poisoning-Angriffe:**

## E Datensätze

GTSRB<sup>39</sup> Datensatz Splitting (Train; Val, test)

ImageNet besteht über 14 Millionen Bildern in 100 Klassen.

## F Programmcode

Der vollständige Programmcode ist verfügbar unter <https://github.com/lukasschulth/MA-Detection-of-Poisoning-Attacks>

## G Notizen

registered spaces<sup>40</sup> Barycenters in the Wasserstein Space<sup>41</sup>

Was passiert bei der Kombination von 2 verschiedenen Triggern? Einmal keine Überlappung(d.h. 2verschiedene Trigger auf dem selben Bild) vs. auch beide Trigger auf einem Bild ist zulässig.

---

<sup>39</sup>[https://benchmark.ini.rub.de/gtsrb\\_dataset.html](https://benchmark.ini.rub.de/gtsrb_dataset.html)

<sup>40</sup><https://arxiv.org/pdf/1809.06422.pdf>

<sup>41</sup><https://arxiv.org/pdf/1809.06422.pdf>

