Bachelorarbeit

Evaluation of Machine Learning Models for *Leakage Detection* in Water Distribution Networks

Julian Hendrik Freiherr Bock von Wülfingen

Universität Bielefeld
Technische Fakultät
AG Machine Learning

Matrikelnummer: 3860703
Erstgutachter: André Artelt
Zweitgutachter: Johannes Brinkrolf
Abgabe: 10.07.2022

Contents

Chapter 1

Einleitung

Wasserverteilungssysteme (engl. Water Distribution Networks, WDNs) gehören zu den wichtigsten Infrastrukturen einer Gesellschaft, in welcher der Anspruch besteht, Trinkwasser in ausreichender Menge und Qualität auf die Haushalte zu verteilen. Da der Hauptteil dieser Netzwerke jedoch unter der Erde vergraben liegt, sind WDNs Anfällig für plötzlich auftretende sowie schleichende Schäden, die häufig ohne direkt sichtbare Indikatoren auftreten können. Das Wasser kann sich hierbei verunreinigen und geht in Massen verloren. Weltweit beträgt der jährliche Wasserverlust, welcher vor allem durch solche Lecks vorangetrieben wird, 126 Milliarden Kubikmeter. Auch wenn es in Deutschland nur 5% des gesamten Wasseraufkommens sind, kann es in manchen Extremfällen bis über 50% gehen. Der finanzielle Schaden allein beträgt weltweit knapp 40 Milliarden USD jährlich [1]. Doch nicht nur aus finanzieller Sicht ist dies dramatisch. In Zeiten des Klimawandels muss auf eine effiziente Nutzung endlicher Ressourcen Acht gegeben werden.

Das Erkennen von Lecks muss demnach schnell und akkurat sein. Alleine das Freilegen der Rohrleitungen kostet viel Zeit [Quelle], weswegen die Rate fälschlicher Erkennung niedrig gehalten werden sollte. Auch wenn manche Lecks einfach zu Erkennen sind, beispielsweise durch sichtbar austretendes Wasser oder spürbaren Druckverlust, ist in den meisten Fällen der Wasserverlust unterirdisch. Um solche Arten zu erkennen, können Sensoren installiert werden, die den aktuellen Druck in den Rohren messen. Da Drucksensoren ebenso einen finanziellen Aufwand bedeuten, haben [Q1] und [Q2] eine Methode entwickelt, ein optimales Sensoren-Placement zu ermöglichen. Thema dieser Bachelorarbeit ist es, die Sensordaten zu analysieren und auf deren Basis potentielle Lecks in WDNs zu erkennen.

Hierfür ist die Arbeit in drei Kapitel aufgeteilt: Zuerst wird auf die theoretischen Grundlagen der Anomaliedetektion in WDNs sowie ausgewählter Modelle und Ideen des Maschinellen Lernens eingegangen. In diesem Zuge wird auch untersucht, welche Metriken zur Evaluation von Detektions-Modellen geeignet sind. Wie zuvor erwähnt, ist dies kein triviales Thema, da neben der Erkennungsrate auch auf eine niedrige Fehlalarm-Quote sowie eine kurze Zeit vom

Auftreten der Lecks bis zu ihrem Erkennen geachtet werden muss.

Weiterführend wird auf die Methodik eingegangen. Hierfür wird zunächst der Datensatz analysiert gefolgt von der Implementation des Erkennungsmodells. Grundlage für ein solches Modell bilden sogenannte digitale Zwillinge, welche ein digitale Repräsentationen realer Abläufe sind; Die Sensorwerte werden simuliert und mit den echten Werten verglichen. Eine zu hohe Diskrepanz kann dann als Problemfall gemeldet werden. In der Vergangenheit wurden diese simulierte Netzwerke häufig als hydraulische Systeme gelöst, also einer Menge an spezifisch auf das WDN angepassten, mathematisch-physikalischen Gleichungen. Diese fordern jedoch in der Modellierung hohe Expertise und besitzen während der Kalibrierung eine starke Komplexität durch viele Freiheitsgrade. Somit werden häufiger Methoden aus dem Bereich des Maschinellen Lernens verwendet, welche die Gesetze der Hydraulik nicht kennen müssen und anhand großer, problem-spezifischer Datenmengen selbst lernen.

Zuletzt werden die Ergebnisse der Methoden vorgestellt und anschließend diskutiert. Hierbei wird auf die Probleme lokaler Optima sowie die Realisierbarkeit der Modelle eingegangen.

Chapter 2

Grundlagen

2.1 Maschinelles Lernen

Eingrenzung

Das umfassende Feld des Maschinellen Lernens (ML) lässt sich grob in zwei Arten unterteilen; Supervised und Unsupervised Learning. In beiden Fällen wird eine Funktion $f : X \to Y$ gelernt, jedoch besitzen die Trainingsdaten für Unsupervised ML keine Informationen für die zu prognostizierende Variable $y \in Y$. Dies kann beispielsweise interessant sein für eine Clustering-Aufgabe, in welcher der Datensatz in zwei oder mehrere Gruppen aufgeteilt werden soll und vorher nichts über diese Verteilung bekannt ist. Die Leck-Detektion benötigt jedoch weitere Informationen; Denn im Gegensatz zu Unsupervised ML ist beim Supervised ML diese sogenannte Ground Truth (GT) bekannt und das jeweilige Problem lässt sich anhand der Definition des Bildbereichs weiter eingrenzen: Die für die Arbeit wichtigen Arten sind hier die binäre Klassifikation und die Regression.

Bei der binären Klassifikation soll jeder Datenpunkt $x \in X$ einer von zwei Klassen $y \in \{0, 1\}$ zugewiesen werden. Auf unser Problem der Leck-Detektion angewendet, hieße das eine Klassifikation eines Zeitpunkts in Klasse 1, es existiert irgendwo in dem WDN ein Leck, oder Klasse 0, es existiert keins. Für die Regression wird jedem Datenpunkt ein reeller Wert $y \in \mathbb{R}$ zugewiesen. Wie dieses Konzept übernommen werden kann, ist in Kapitel ? weiter beschrieben.

Modelle des Maschinellen Lernens

Für jeden Anwendungsbereich gibt es eine Vielzahl an verschiedenen Algorithmen, welche eine gegebene Aufgabe lösen können. Die in dieser Arbeit benutzten Modelle werden im Folgenden erklärt. Hierbei ist wichtig zwischen Parametern und Hyperparametern zu unterscheiden. Ersteres sind dabei die Variablen und Gewichte, die während des Lernprozesses optimiert oder gefunden werden. Die

sogenannten Hyperparameter sind Einstellungen, welche vor dem Training konfiguriert werden und je nach Einstellung andere Ergebnisse liefern.

- **k-Nearest Neighbors (kNN)** ist ein einfaches Modell, bei dem ein neuer Datenpunkt nach der Mehrheit seiner Nachbarn (x_i, y_i) in einem n-dimensionalen Koordinatensystem klassifiziert wird. Der Hyperparameter k bestimmt hierbei wie groß die zu betrachtende Nachbarschaft ist. Sei $\mathcal{N}(\hat{x}, \mathcal{D}, k)$ die Menge der k am nächsten zu $\hat{x}$ liegenden Punkte aus $\mathcal{D}$, dann ist $P(\hat{x} = c|\hat{x}, \mathcal{D}, k) = \frac{1}{k}|\{x_i|x_i \in \mathcal{N}(\hat{x}, \mathcal{D}, k), y_i = c\}|$ der Anteil der Klasse c in der Nachbarschaft. Die finale Klasse des neuen Datenpunktes $\hat{x}$ ist dann gegeben durch die am meisten vertretene Klasse:

$$f_{kNN}(\hat{x}) = \underset{c \in \{1..C\}}{\operatorname{argmax}} P(\hat{x} = c|\hat{x}, \mathcal{D}, k) \tag{2.1}$$

  Zusätzlich lässt sich mit dem Hyperparameter der Gewichtung einstellen, ob Punkte, die sich im Raum weiter weg befinden, weniger stark gewichtet werden.

- **Multi-Layer Perceptron (MLP)** versucht, ein künstliches neuronales Netz zu erschaffen, indem es mehrere Perzeptronen aneinanderreiht. Ein Perzeptronen steht hier für ein Neuron, welches mehrere Eingabewerte hat und daraus einen Ausgabewert erstellt. Dafür wird jeder Eingabewert mit einem speziell dafür gelernten Gewicht multipliziert und anschließend wird alles aufsummiert[1]. Als letztes wird eine sogenannte Aktivierungsfunktion auf die Summe angewendet, was dann die Ausgabe des Perzeptron bildet. Die Aktivierungsfunktionen, welche in dieser Arbeit betrachtet worden sind, sind die logistische Funktion, der Hyperbeltangens und der Rectified Linear Unit (ReLU):

$$\text{logistic}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$\tanh(x) = 1 - \frac{2}{e^{2x} + 1} \tag{2.3}$$

$$\text{relu}(x) = \max(0, x) \tag{2.4}$$

  Die Ausgabe eines Perzeptrons ist also

$$f(x) = f_{activation}(\sum_i (w_i x_i) + \theta), \tag{2.5}$$

  wobei $f_{activation}$ die Aktivierungsfunktion und w die gelernten Gewichte sind. Ein MLP ist dann eine Ansammlung vieler einzelner Perzeptronen in möglicherweise mehreren Schichten. Dafür besteht eine Schichte aus mehreren, in parallel arbeitenden Perzeptronen, deren Eingabe die Ausgabe aus der vorherigen Schicht darstellt. Ihre Ausgabe ist dann wieder

[1] TODO +theta

Eingabe der folgenden Schicht. Abbildung ? zeigt so eine Struktur. Dabei ist die Anzahl der Schichten sowie die Menge an Perzeptronen pro Schicht ein Hyperparameter.

Die einzelnen Gewichte lernen kann dieses Modell indem es die Datenpunkt durch das (untrainierte) Netz schickt und die Vorhersage mit der GT vergleicht. In einem Verfahren, was sich Backtracking nennt, werden im Grunde die Gewichte, welcher zu einer falschen Entscheidung führten geschwächt, während Gewichte, die eine richtige Entscheidung herbeiführen konnten, verstärkt werden. Diese Technik lässt sich durch weitere Hyperparameter, welche zum Beispiel die Rate der Verstärkung beeinflussen, optimiert werden.

- **Linear Regression (LR)** ist eine Regressionsmethode, die versucht, eine Gerade im n-dimensionalen Raum zu schaffen, welche die Abstände zu den bekannten Datenpunkten minimiert. Der Wert eines neuen Datenpunktes wird nun durch die parametrische Funktion

$$f(\hat{x}) = \sum_i w_i x_i - \theta \tag{2.6}$$

prognostiziert. Um die optimalen Gewichte zu finden nutzt LR den Mean Squared Error (MSE), welcher gegeben ist als

$$MSE = \frac{1}{n} \sum_i (y_i - f(x_i))^2 \tag{2.7}$$

und die durchschnittliche, quadrierte Abweichung der Punkte von der Linie beschreibt. Die finalen Gewichte sind nun $w = \text{argmin}_{\tilde{w}} MSE$.

- **Ridge (L2)** und **Lasso (L1)** Regression sind Erweiterungen der LR indem sie einen Regularisierungsterm zum MSE hinzufügen, welcher Einfluss auf die optimalen Gewichte hat. Für L2 werden die einzelnen Gewichte quadriert und mit einem Hyperparameter α[2] multipliziert. Die optimalen Gewichte werden damit berechnet:

$$w = \text{argmin}_{\tilde{w}} MSE + \alpha ||\tilde{w}||_2^2. \tag{2.8}$$

Dadurch werden zu hohe Gewichte und damit Overfitting, was in Kapitel ? weiter erklärt wird, bestraft. L1 nutzt die absoluten Werte der Gewichte:

$$w = \text{argmin}_{\tilde{w}} MSE + \alpha ||\tilde{w}||_1. \tag{2.9}$$

Damit kann einfacher mit Dimensionen umgegangen werden, die nur wenig Einfluss auf das Endergebnis haben.

Das korrekte Setzen der Hyperparameter gehört zu dem Fachbereich des Fine Tunings von Modellen und kann gravierende Effekte auf das Endergebnis haben. Methoden um diese Aufgabe anzugehen, werden in den nächsten Kapiteln beschrieben.

[2] In der Literatur auch häufig λ genannt.

2.2 Evaluation und Metriken

Um Einschätzen zu können, wie gut ein Modell mit gegeben Hyperparametern und ausgewählten Daten das Problem löst wird eine Metrik benötigt. Eine solche ist definiert als Funktion, welche die wahren und die prognostizierten Werte annimmt und einen einzigen Wert[3] als Indikator, wie gut in einer gewissen Disziplin abgeschnitten wurde, zurückgibt. Die meisten Metriken, die in dieser Arbeit verwendet werden, können aus der Konfusionsmatrix berechnet werden. Für ein binäres Klassifikationsproblem hat sie den folgenden Aufbau:

		Echtes Label	
		1	0
Modell Ausgabe	1	TP	FP
	0	FN	TN

Auf der Diagonalen stehen TP (True Positive) für die Menge an positiven Fällen, die auch als solche erkannt wurden, und TN (True Negative) für alle negative Fälle, die ebenfalls richtig vom Modell erkannt wurden. Die anderen Werte stehen für keine Übereinstimmung der Werte; FN (False Negative) steht für ein fälschlicherweise als negativ klassifizierten Datenpunkte und FP (False Positive) für die Punkte, an denen das Modell ohne Grund positiv angeschlagen hat. Tabelle ? zeigt ausgewählte Metriken, die in dieser Arbeit benutzt werden und was sie in diesem Kontext bedeuten.

Metrik	Berechnung	Bedeutung
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Wie oft lag der Algorithmus richtig?
Recall (Sensitivität)	$\frac{TP}{TP+FN}$	Wie gut wurden echte Lecks erkannt?
Specificity	$\frac{TN}{TN+FP}$	Wie gut wurde 'alles ok' erkannt?
Precision	$\frac{TP}{TP+FP}$	Wie viele erkannte lecks waren auch wirklich Lecks?
Detection Time	Mittelwert vom Auftreten eines Lecks bis zur Erkennung	Wie viele Zeiteinheiten dauerte es bis zum erkennen?

Hierbei ist wichtig anzumerken, dass auch wenn die Accuracy alle Werte der Konfusionsmatrix in die Bewertung mit einfließen lässt, nur als grobe Einschätzung genutzt werden sollte, da sie keine Begründung liefert, wie die anderen Metriken. Zudem kann sie auf unbalancierten Datensätzen ein falsches Bild vermitteln[4].

[3] typischerweise zwischen 0 und 1

[4] TODO bsp. 99%

Um die Metriken erheben zu können, muss man jedoch definieren, welche Datenpunkte hierfür genutzt werden und welche nicht. Das nächste Kapitel gibt Aufschluss darüber, wie dies bewerkstelligt werden kann.

2.3 Hyperparameter Tuning

Das Ergebnis eines Machine Learning Modells hängt stark von den gewählten Hyperparametern ab. Diese zu optimieren gehört zu den Kernaufgaben im Lebenszyklus des Modellierungsprozesses, für die jedoch eine Verlässliche Aussage der Metriken wichtig ist. Man sollte davon absehen, auf den gleichen Daten zu testen, auf denen man bereits trainiert hat. Dabei kann es nämlich unbemerkt zu Overfitting kommen, was bedeutet, dass das Modell nicht genug generalisiert, sondern die Eigenheiten des Datensatzes, wie zum Beispiel Rauschen, lernt. Dadurch funktioniert es schlechter auf nicht gesehenen Daten. Weiter sind Metriken nicht mehr so aussagekräftig, wenn nur bereits Bekanntes abgefragt wird[5]. Die Test- und Trainingsdaten sollten also disjunkt sein.

Train-Test-Split

Beim Train-Test-Split wird die Datenmenge in zwei Gruppen aufgeteilt: Das Train-Set und das Test-Set. Typischerweise beträgt die größe des Test-Sets zwischen 20% und 33% der gesamten Daten. Da bei diesem Split auf neuen Daten getestet wird, zwingt es einen dazu, das Modell so zu gestalten, dass es gut generalisieren kann. Bei einer unglücklichen Wahl des Test-Sets, kann es jedoch immer noch passieren, dass die Metriken einen falschen Eindruck hinterlassen[6].

Cross-Validation

Die Lösung dieses Problems ist das wiederholte Anwenden eines Train-Test-Splits. Bei der sogenannten k-Fold Cross-Validation wird der Datensatz in k gleich große Mengen unterteilt. Mit dieser Aufteilung wird dann k-Mal der Prozess des Trainierens und Testens gestartet, wobei jedes mal ein anderes der k Sets als Test-Set gewählt wird. Abbildung ? visualisiert dieses Vorgehen. Die Ergebnisse der k Durchläufe werden anschließend gemittelt und zeigen damit eine weitaus robustere Einschätzung der Güte auf. Das k liegt hier typischerweise zwischen 5 und 10[7].

[5] Mit $k = 1$ hätte kNN einen Fehler von 0, falls auf den gleichen Daten getestet und trainiert wird.

[6] TODO 33% class1 in train, 0 in test

[7] Es gibt einen Spezialfall, indem k gleich der größe des Datensatzes ist. Eine solche Einstellung nennt sich Leave-One-Out Cross-Validation, bei welcher immer auf allen Datenpunkten mit Ausnahme eines einzigen Punktes trainiert wird. Dies ist jedoch nicht Thema dieser Arbeit.

Grid-Search

Mit dieser Technik können nun verlässlich die Hyperparameter analysiert werden. Mit der Methode des Exhaustive Grid-Search lassen sich alle Hyperparameter testen, indem das kartesisches Produkt der möglichen Einstellungen gebildet und auf allen Kombinationen mittels Cross-Validation trainiert und getestet wird. So wird für kNN einmal mit uniformer Gewichtung der Nachbarn jede Einstellung für k getestet und einmal mit Distanz-bedingter Gewichtung. Das Ergebnis zeigt nun die optimale Einstellung für die Gewichtsfunktion und das k an. Ebenso können weitere Relationen ausgelesen werden; Beispielsweise könnte für ein höheres k die Distanz-bedingter Gewichtung besser funktionieren, für kleinere k jedoch eher die uniforme Gewichtung.

2.4 Anomaliedetektion

Für die Detektion von Lecks in WDNs gibt es mehrere Ansätze, welche sich in aktive und passive Strategien einteilen lassen.

- Die **aktiven Verfahren**, auch Hardware-basierte Verfahren, sind Strategien, die mittels spezialisierter Technik aktiv nach Brüchen in den Rohren suchen. Hierfür könnten zum Beispiel Schallgeneratoren oder Kameras benutzt werden. Dabei steht jeder Suchvorgang in der Regel für sich und bildet keinen Verlauf ab. Zudem sind aktive Verfahren sowohl Zeit- als auch Ressourcenintensiv.
- Dahingegen sind **passive Verfahren**, oder auch Modell-basierte Verfahren, darauf ausgelegt ein kontinuierliches Bild über den aktuellen Zustand des Netzwerkes zu geben und bei Anomalien Alarm zu schlagen. Hierfür wird die bereits erwähnte Technik der digitalen Zwillinge angewendet, indem das reale Netz, welches durch eine Menge an Sensoren durchgehend überwacht wird, durch ein virtuell simuliertes Netzwerk erweitert wird. Mit der Annahme, dass die Simulation den Normalzustand, also eine Leckfreie Version des Netzwerkes, abbildet, können aus zu hohen Differenzen Anomalien abgeleitet werden. Die Simulation kann auf zwei Wege erstellt werden. Die Hydraulic Model Based Verfahren nutzen spezifisch kalibrierte, hydraulische Gleichungen. Sie berücksichtigen alles von der genauen Struktur und Höhenlage über Material bis hin zu Rohrdicken und Alter. Die resultierende Menge an Gleichungen ist daher sehr komplex und bei sensibel gegenüber Ungenauigkeit in der Beschreibung. Auf der anderen Seite stehen die Hydraulic Measurement Based (oder auch datengetriebenen) Verfahren. Diese nutzen maschinelles Lernen um mittels Langzeitdaten des Netzwerkes Vorhersagen zu treffen. Dadurch ist kein Vorwissen über die hydraulischen Eigenschaften des Netzwerkes nötig, da das notwendige Wissen von den Algorithmen gelernt wird.

In dieser Arbeit geht es um die datengetriebenen, also die passiven, auf vorangegangenen Messwerten basierten, Verfahren. Ein solches Modell funktioniert

in zwei Schritten. Im ersten Schritt wird das digitale Netzwerk simuliert. Hierfür wird für jeden echten Sensor ein digitaler Sensor erstellt. Dieser schätzt den eigenen Druckwert anhand der aktuellen Druckwerte aller anderen Sensoren. Möglicherweise können auch direkt vorangegangene Messwerte in diese Vorhersage mit eingebaut werden. Wird dies für jeden Sensor gemacht, entsteht ein digitales Netzwerk, bei dem jeder digitale Sensor kein Wissen über seinen realen Wert hat. Im nächsten Schritt wird dann zuerst die Differenz der Netzwerke berechnet. Ist momentan kein Leck im WDN, so sollten diese Differenzen klein sein, während Lecks zu höheren Differenzen führen. Hier gilt es nun, einen Threshold zu finden, ab dem eine Differenz zu hoch ist, um im Normalbereich zu liegen. Ist dieser abhängig von dem jeweiligen Sensor, so stellt sich ebenso die Frage, bei wie vielen Sensoren eine Differenz als 'zu hoch' gelten muss, damit das gesamte Modell Alarm schlägt.

Wie in Kapitel ? schon angemerkt, unterscheidet sich die Güte eines Modells auch darin, welche Metrik benutzt wird. So folgt aus dem finanziellen und ökologischen Schaden eines Lecks, dass potentielle Lecks früh erkannt werden sollen; Eine niedrige Detection Time und hohe Sensitivität (bzw. Recall) ist gefordert. Während das eigentliche Reparieren eines Rohrs in relativ geringer Zeit absolviert werden kann, ist das, was drum herum getan werden muss umso Zeit- und Ressourcenintensiv. Vom Absperren der Straße über das Schließen der Ventile bis hin zum Aufgraben des Bodens bis zum Rohr können viele Stunden vergehen [TODO Quelle], die im Anschluss wieder rückgängig gemacht werden müssen. Eine hohe Rate an falschen Alarmierungen, also eine niedrige Präzision, sollte dementsprechend auch vermieden werden. Verknüpft mit anderen Detektionsverfahren, wie zum Beispiel der aktiven Suche zum Überprüfen potentieller Lecks, kann der Wert auf eine optimale Präzision jedoch auch wieder geschmälert werden[8].

[8] Da diese Verfahren jedoch wie besagt auch aufwändig sind, sollte dennoch auf die Präzision geachtet werden.