Universität Potsdam

Mathematisch-Naturwissenschaftliche Fakultät

Exam: M. Streicher, September 24, 2024, Potsdam

Intelligent Data Analysis - Medical lasers

Problem Setting

Input data D:

$D = \{(x_i, y_i)\}_{i=1}^{200}$,

where $x_i = [x_{i1}, x_{i2}, \ldots, x_{i60}]$ is the frequency time series of laser i and $y_i \in \{-1,1\}$ the binary label.

$y = 1$, Suitable for sale

$y = -1$, Not suitable for sale

Model:

$f_\theta : \mathbb{R}^{60} \rightarrow \{1, -1\}$, where θ is the model parameter vector.

Das Modell ordnet einem **Eingangsvektor** aus dem **60-dimensionalen reellen Raum** einen Wert aus der Menge {1,−1} zu, wobei θ der Parametervektor des Modells ist.

Problem Setting

Task:

Binary classification problem

Type of learning:

Supervised Learning

Goal:

Classify each laser as belonging to one of two classes,
1 or -1, based on the frequency behavior over time.

Data analysis

Input data D:

$$D = \{(x_i, y_i)\}_{i=1}^{200}$$

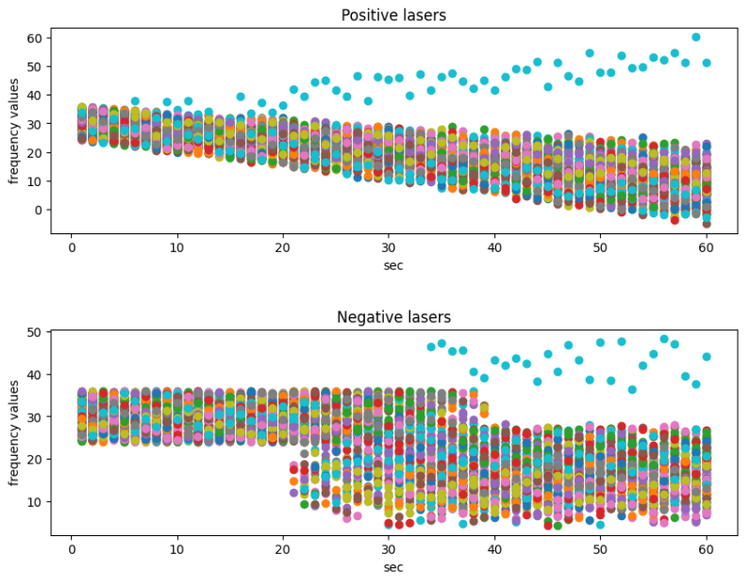
Label distribution:

$$D_{-1} = \{(x_i, y_i) \mid y_i = -1\} = 100$$

$$D_1 = \{(x_i, y_i) \mid y_i = 1\} = 100$$

Missing data:

No missing data found.

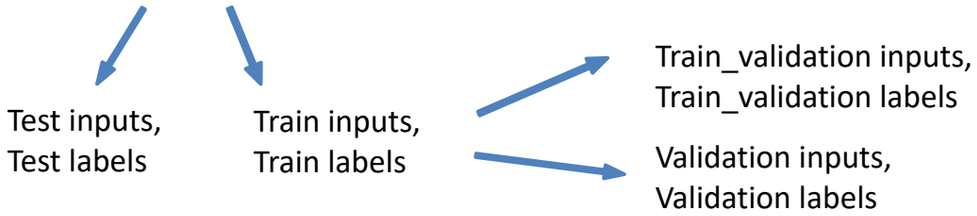


50/50 —> Baseline für das Modell ist eine Accuracy von 50% d.h. entweder alle richtig oder alle falsch

Evaluation Protocol

<u>Train and Test:</u>

BaseDataset(Dataset) #pytorch ⇄ data_split_indices.pkl #pickle



<u>Tripple Cross Validation:</u>

ParameterGrid() #sklearn.model_selection

Models

Linear Classifier, DTW Kernel, Polynomial Kernel, RBF Kernel

Linear Classifier

Model:

$$argmin_\theta \sum_{i=1}^{n} l(max(0,1 - y_i f_\theta(x_i)), y_i) + \lambda \Omega_2(\theta), \text{ with } f_\theta(x_i) = x_i * \theta_i$$

ERM using Gradient Descent Method:

$$\nabla_\theta L(\theta) = \begin{cases} \frac{2\lambda}{n}\theta, & \text{if } 1 - y_i(\mathbf{x}_i \cdot \theta) \leq 0 \\ -y_i \mathbf{x}_i + \frac{2\lambda}{n}\theta, & \text{if } 1 - y_i(\mathbf{x}_i \cdot \theta) > 0 \end{cases}$$

Feature Engineering:

Feature I: R^2

Feature II: Maximal difference to a subsequent measuring point.

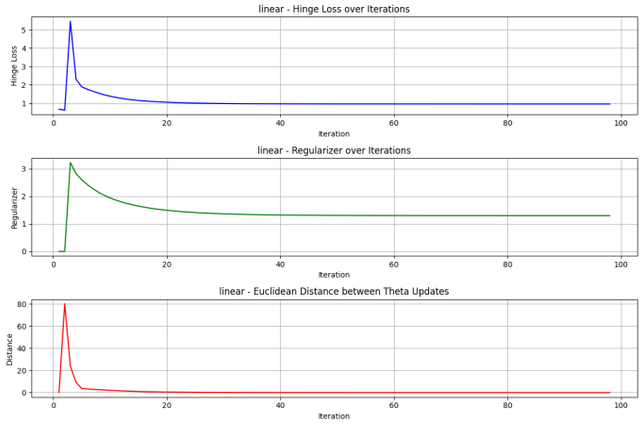
* Minimierungsproblem
* Loss-Funktion und Rehgularisierer
* Im Falle des SVMs, was wir hier sehen: Hinge Loss und L2-Regularisierer d.h. Squared Euclidean Norm
* Hier Linear Classifier d.h. Lineare Entscheidungsfunktion
* Das Ganze ist eine Empirical Risk Minimization und ich habe hier das Gradient Descent Verfahren angewendet.
* D.h. Wir bilden in jedem Iterationsschritt den Gradienten —> Siehe Folie
* !-yi(xiTheta) = Margin —> Je nach Margin wird der entsprechende Gradient benutzt
* Gradient: Vector —> Länge == Anzahl der Instanzen
* UpdateSchritt: theta = theta_old-alpha*gradient
* Alpha würde bestimmt hier mit: alpha = decay**iteration

Für lineares Modell benötigt: Feature Engineering
1. R squared
2. Maximale différent zwischen aufeinanderfolgenden Messpunkten

Warum habe ich das gemacht?
* Neugierde
* Testen meines Gradient Descent Verfahren
* Ergebnis Vergleich mit dem linearen classifier von sklearn

Linear Classifier

```
Best parameters found:
{'alpha_0': 0.001, 'decay': 0.9, 'epsilon': 0.0001, 'lambda_value': 0.001}
Best score: 0.925
```



* Vorher: GridParameter Search gemacht
* Parameter siehe Folie

Alpha_0 =Anfängliche Schrittweite im Gradient Descent
Decay = Wird nach der Iteration benutzt, um die Schrittweite einzustellen mit alpha = decay**gradient
Epsilon = Threshold für Euclidean Distance zwischen dem alten und neuen Theta
Lambda = Trade-Off zwischen Loss-Funktion und Regularisierer

Best Score = Accuracy

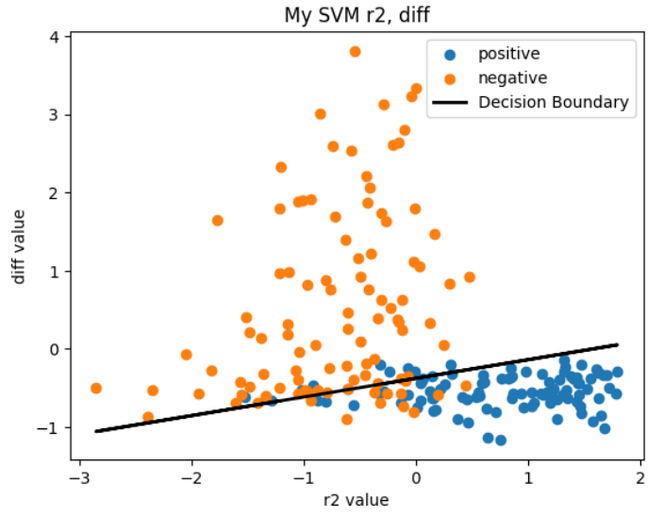
Warum verlaufen die Kurven so?
Beide Features wurden standardisiert mit **z-score Normalisierung**.

D.h.
MW 0, SD 1
-> Daten haben kleine Werte, was dazu führt, dass die zu Beginn berechneten Fehler klein sein können.

Hinge Loss bestraft, wie sehr der vorhergesagte Wert vom Ziel abweicht. Dies bedeutet, wenn theta klein ist, dann treten kleine Fehler auf d.h. keine großer Hinge Loss.

Regulariser oft direct proportional zum Regularisierer d.h. ebenfalls kleiner Wert erwartet

Linear Classifier



DTW Kernel

$$k_{\mathrm{DTW}}(x, x') = e^{-\lambda d\mathrm{DTW}^{(x,x';d)}}$$

$$d_{DTW}(x, x'; d) = \gamma(\,|\,x\,|\,,|\,x'\,|\,)$$

$$\gamma(i,j) = \begin{cases} d(x_i, x'_j) + \min\left(\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\right) & (1 \le i \le |x|, 1 \le j \le |x'|), \\ \infty & i = 0 \vee j = 0, \\ 0 & (i,j) = (0,0)\,. \end{cases}$$

DTW Kernel:

Dynamic Time Warping - Bestimmungen Ähnlichkeit zwischen zwei zeitlichen Sequenzen misst, auch wenn sie in der Geschwindigkeit variieren.

* Euclidean Distance
* Dynamic Programming

DTW Kernel:

$$k_{\mathrm{DTW}}(x, x') = e^{-\lambda d \mathrm{DTW}(x,x';d)}$$

Polynomial Kernel:

$$k_{poly}(x, x') = (\alpha * x^T x' + c)^d$$

RBF Kernel:

$$k_{RBF}(x, x') = e^{-\lambda * \|x - x'\|^2}$$

Polynomial Kernel:

* Erkennung von nicht-linearen Mustern
* Alpha - Beeinflusst, wie stark Interaktionen zwischen den Merkmalen skaliert wird
* D.h.: Wird skaliert, bevor das Polynom angewendet wird
* Großes alpha verstärkt die Interaktion von Merkmalen —> Wahrscheinlichkeit ein hohes Polynom zu bekommen steigt —> Kompliziertere Entscheidungsgrenze
* Kleines alpha vermindert die Interaktion von Merkmalen —> Wahrscheinlichkeit ein kleines Polynom zu bekommen steigt —> Flachere Entscheidungsgrenze
* D - Grad des Polynoms
* C - Offset
* Generell hohes Risiko für den Overfit —> Happy Face Kurve

RBF Kernel:

* Radial Basisi Function Kernel
* Er misst die Ähnlichkeit zwischen zwei Datenpunkten in einem unendlichen Raum
* Generel geeignet für kreisförmige oder radiale Datenverteilungen geeignet
* Gamma ist freizuzählender Parameter -> Bestimmt wie weit der Einfluss eines einzelnen Trainingsbeispiel reicht —> Höhere gamma Werte führen zu engeren Entscheidungsgrenzen
* Gefahr der Überanpassung gegeben

Results

Linear Classifier, DTW Kernel, Polynomial Kernel, RBF Kernel

	DTW	Polynomial	RBF
Alpha	0,001	0,1	0,001
Decay	0,1	0,1	0,9
Epsilon	0,0001	0,0001	0,0001
Lambda	10	0,1	1
Kernel specific		C = 0 Degree = 5 Alpha_Poly = 1	Gamma = 0,001
Accuracy	**0,975**	**0,675**	**0,975**

* Parameter haben zwischen durch auch geschwankt d.h. sind nicht unbedingt feste Werte -> Beispielhaft

Allgemein:
* Kleines Epsilon d.h. kleinen Threshold für die Distance zwischen alten und neuem Theta
* Lambda zeigt recht unterschiedliche Werte
* Hohes Lambda - Starker Einfluss vom Regularisierer bsp. DTW , Wenger Einfluss bei Polynomial

Kernel specific:

A - Polynomial
* Sehen nach sehr umoptimalen Werten aus, Warum?
* Alpha recht hoch d.h. höherer degree wahrscheinlich
* Macht sich bemerkbar mit Degree von 5
* C = null zeigt, dass der Regularisierungsterm im Kernel ausgeschaltet ist
* Naiver Gedanke: Modell kann zu stark angepasst sein, bzw. eine zu komplexe Entscheidungsfunktion rein theoretisch haben
* Zeigt sich in der Accuracy von 0,675
* Parameter wurden aber mit Triple Cross Validation bestimmt
* Somit ist die Wahrscheinlichkeit einer Überanpassung minimiert
* Accuracy war auch über die Ganze Tripple Cross Validation nicht gut
* D.h. Das Modell passt nicht gut zu den Daten
* D.h. Daten haben keine polykomische Beziehung zueinander

B - RBF
* Kleiner Wert für Gamma
* D.h.: Einfluss eines Trainingsbeispiels reicht weit und das Modell lernt eher eine glatte Entscheidungsgrenze
* Spiegelt sich auch in der Accuracy wieder von 0,975
* Genau wie bei dem DTW Kernel

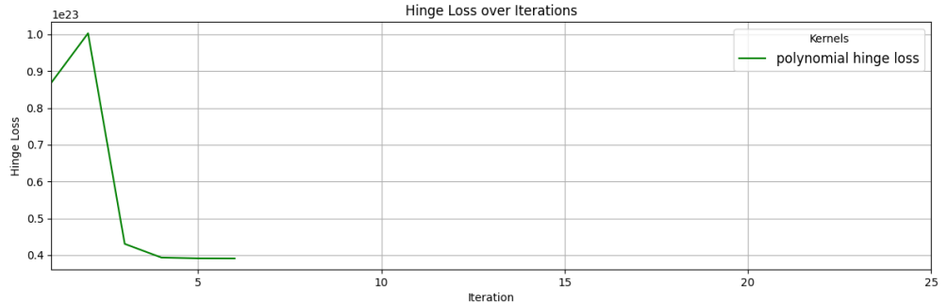
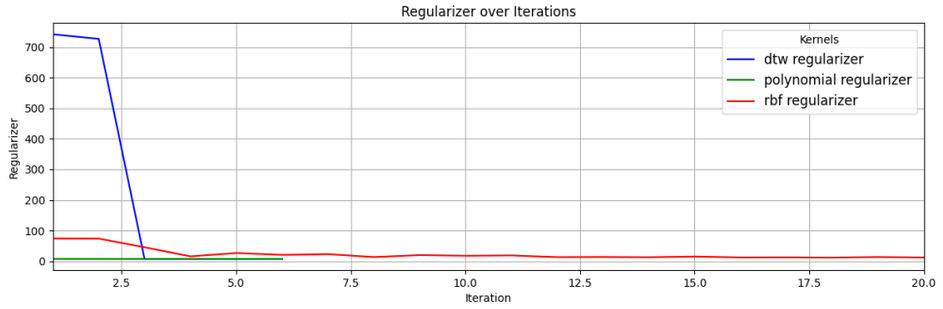
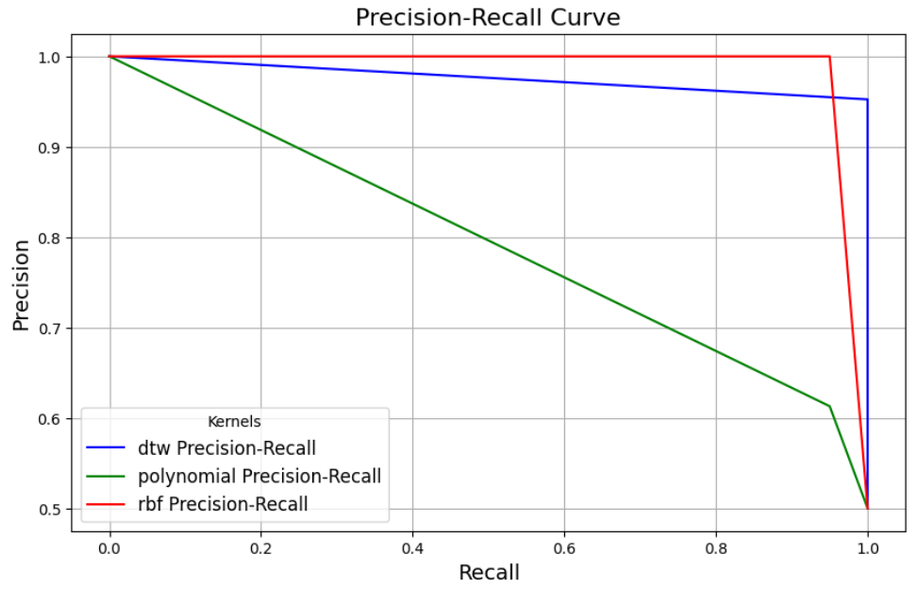


Diagram erklären + RBF zeigt das beste Verhalten.

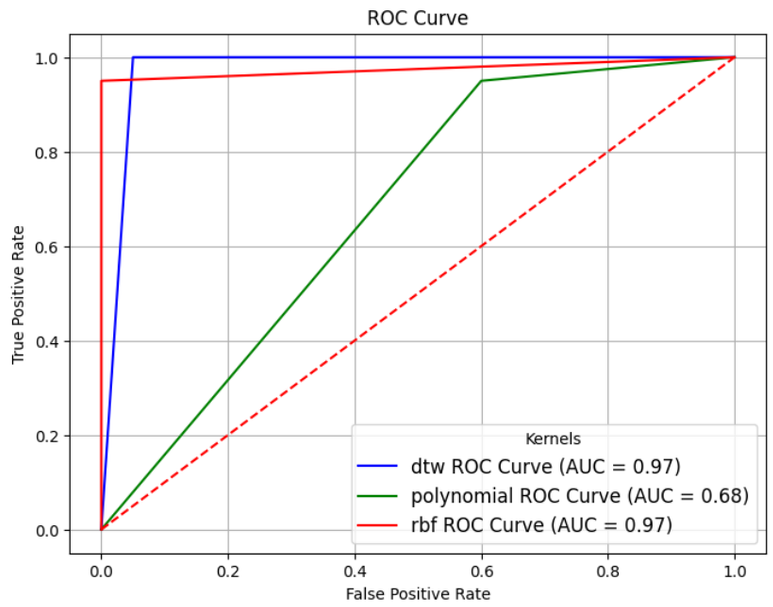


* DTW Regularisierer am Anfang sehr hoch, aber nach schnellen Anpassungen schnell niedrig
* RBF Regularisierer sehr konstant d.h. RBF benötigt weniger Anpassung der Gewichte —> Deutet auf eine stabile Optimierung gin
* Polynomial hat schon seit dem Anfang schlechte Werte d.h. kaum einen Einfluss auf die Gewichte



Präzision - Wie viele der positiven Vorhersagen sind korrekt —> Das Modell produziert wenig falsch Positive
Recall - Sensitivität - Wie viele der tatsächlich positiven Fälle wurden erkannt -> Wenig falsch negative Vorhersagen

Ziel: Keine Falsch Positive Laser verkaufen d.h. Präzision wichtiger => RBF



Präzision gegen die Falsch Positiv Rate

* Falsch Positive Rate soll gegen null gehen d.h. RBF führt

Conclusion

Linear Classifier, DTW Kernel, Polynomial Kernel, RBF Kernel

Conclusion

- **Polynomial Kernel does not fit the data**
- **Lowest accuracy** with 0.675
- **Followed by the linear model** with 0.95
- **Positive**: Hardly any risk of overfitting
- **Negative**: Requires preprocessing of the data
- **Best kernels with the same accuracy of 0.975**: DTW, RBF
- **RBF shows significantly better precision** in the Precision-Recall curve and ROC

Conclusion: RBF would be the model of choice.


Universität Potsdam

Sources

[1] Image cover slide: rbb, 2016, Retrieved from: https://www.rbb-online.de/content/dam/rbb/rbb/fernsehen/rbb_praxis_bilder/2016/05/25/COLOURBOX8440750.jpg.jpg/size=966x543.jpg

[2] Image question slide: Gekonnt wirken, Retrieved from: https://www.gekonnt-wirken.de/wp-content/uploads/2018/07/AdobeStock_496887170-scaled-82625_1080x675.jpeg

[3] Source Code: https://github.com/MarStreicher/IDA_Laser/tree/main/