Maschinelles Lernen

Vorlesung 4

Cross-Validation

Was passiert, wenn man das Test-Dataset auf verschiedene Arten auswählt?

Was sollte im Idealfall passieren?

In der Praxis: k = 5 .. 10

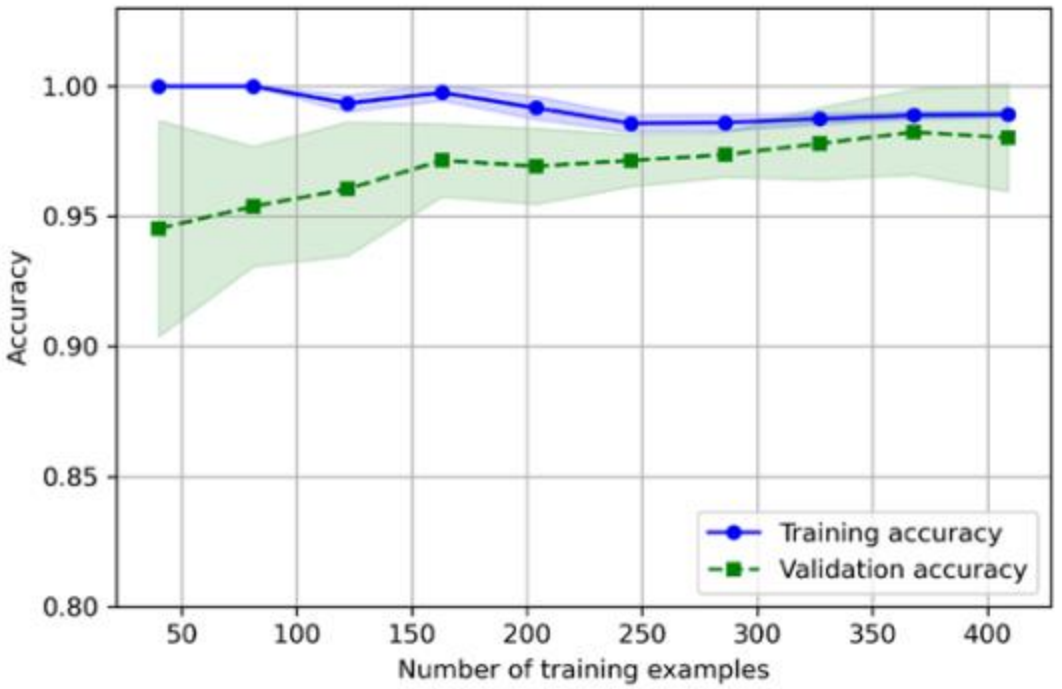
Mit 10 Folds kann man besser das Bias-Variance-Tradeoff bewerten, und die L1, bzw L2 Regularisierungen anpassen

Lern- und Validierungskurve

- Maß für Over- bzw Underfitting

Lernkurve: Zeigt wie sich die Metriken des Modells für eine vorgegebene Anzahl von Datenpunkten entwickeln

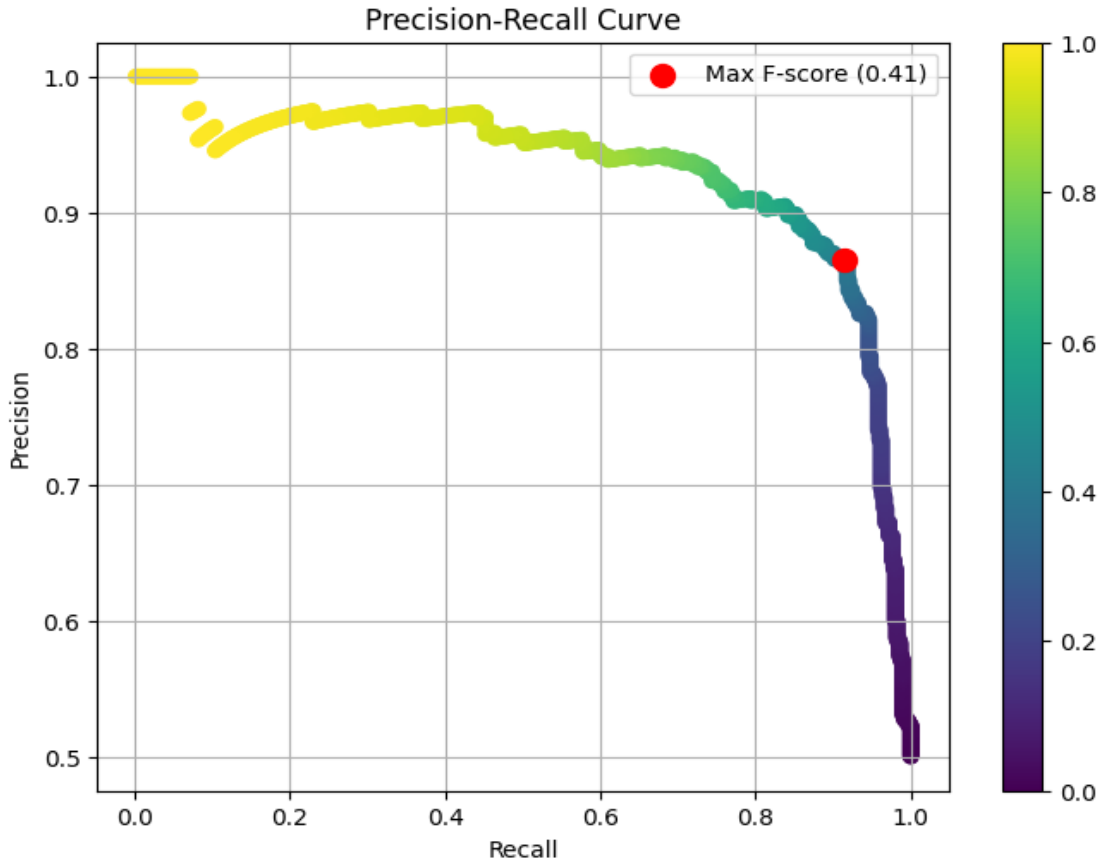
Validierungskurve: Dasselbe für einen gegebenen Modellhyerparameter



Quelle: Raschka (2019)

Precision-recall curve

- Modell gibt Wahrscheinlichkeiten aus:
  - 0: gehört zu Klasse A
  - 1: gehört zu Klasse B
  - 0.3: zu welcher Klasse gehört es?
- Man kann die Schwelle ändern, der default-Wert ist 0.5
- Nützlich wenn man unbalancierte Daten hat oder wenn man eine von Precision und Recall optimieren muss



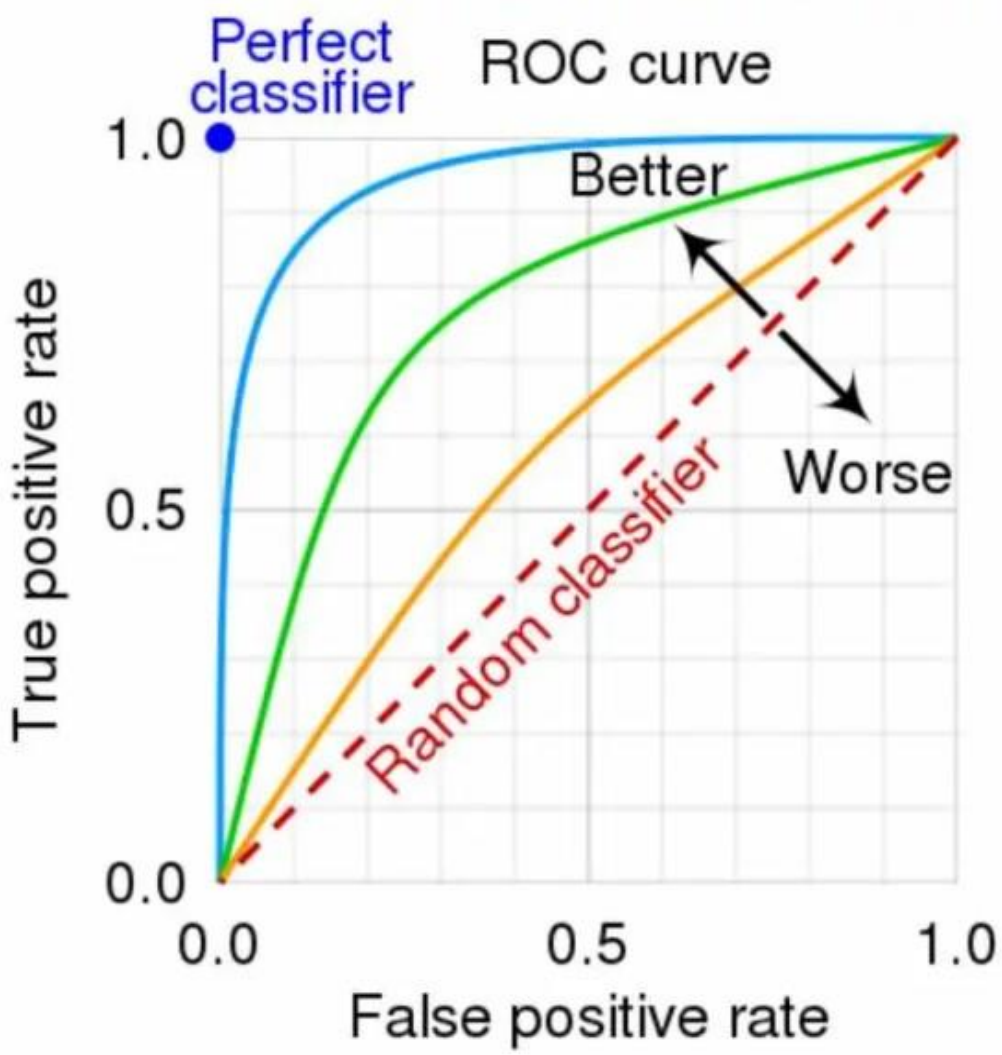
ROC (Receiver operating characteristic) Kurve

AUC (area under curve):

- im Idealfall 1
- wenn das Modell rätselt: 0.5
- link

Bemerkungen:

- Für Multiklasse Klassifikationen: 1 vs all
- Alle sklearn-Metriken sind auch in pytorch verfügbar, Syntax ist meistens komplexer
- Der Roc ist ein riesiger legendärer Vogel aus der orientalischen Mythologie



Ungleichgewicht bei den Klassen

- Tretet in der Praxis sehr oft auf – denke an praktische Beispiele
- recall, f1 und auc-roc betrachten – warum?

Ungleichgewicht bei den Klassen
Mögliche Lösungen

- Anpassen der precision-recall-Schwelle
- Stärkere Strafe für falsches Predicten der kleinen Klasse

in der Praxis am meisten benutzt

- Ensemble Learning: gesamte kleine Klasse mit einer Untermenge der großen Klasse

sehr kräftige Methode, braucht aber viel Rechenkraft, große Wahrscheinlichkeit von Overfitting

- Resampling der kleineren Klasse
- Undersampling der größeren Klasse (falls genügend Daten vorhanden)
- Generieren von synthetischen Daten
- Erhalten von zusätzlichen Daten für die kleine Klasse

in der Praxis selten möglich 😢

Die Wissenschaft der seltenen Events ist ein riesiges Feld im Bereich Statistik, mathematische Modellierung, Machine Learning - genügend Stoff für ein ganzes Semester!

Ungleichgewicht bei den Daten
SMOTE

Synthetic Minority Oversampling Technique

Für Punkte aus der kleineren Klasse:

- Man wählt für einen gegebenen x_i einen Punkt x_j aus den k nächsten Nachbarpunkte
- Man berechnet $x_{new} = x_i + \lambda(x_j - x_i)$

- Es gibt zahlreiche Variationen, für alle Arten von Datentypen
- Meistens für Experimentieren und PoCs
- Zu vermeiden in stark reglementierten Bereichen (Banking, Versicherung, Medical)

Metriken für Regression

MSE = mean squared error (siehe V1)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

R2 = MSE, aber normalisiert auf [0, 1]:

- 0: wenn die Prediction für jeden Punkt der Durchschnittswert ist
- 1: Perfektes Fit
- Kann sehr selten negativ sein

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

Intuitiv: wie viel von der Varianz der predicted Zielvariable kann das Modell durch die Varianz der Features erklären

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Bemerkung: Man kann baumbasierte Methoden auch für Regression, mit einer vorgegebenen Verlustfunktion, benutzen (link)

Where,

$\hat{y}$ – *predicted value of y*

$\bar{y}$ – *mean value of y*

Datentransformationen

- Immer **VOR** dem Skalieren!

- Nicht notwendig für baumbasierte Methoden – warum?
- Nicht notwendig für neuronale Netze für Text- und Bilderverarbeitung – wird automatisch beim Konvertieren dieser Daten in numerischer Form macht
- Notwendig für alle anderen Fälle

Datentransformationen

Logarithmieren von Features

- Verzerrte Verteilung, "long tail"
- Positive Werte
- Sehr oft in der Praxis verwendet
- Achtung wenn andere Features aus dem logarithmierten Feature berechnet werden!!!!

Andere Transformationen

- Nischenfälle
- z.B. Yeo-Johnson transform, Erweiterung der Logarithmierung falls es negative Werte gibt oder die Daten sehr stark verzerrt sind

Optimierung der Hyperparameter

Hyperparameter von Hand optimieren ist langwierig und nicht immer optimal

Man kann heuristisch einschätzen, welche Hyperparameter man optimieren soll, und welcher der Raum der Suche sein sollte

Man wählt eine Zielmetrik, für die man optimiert

Man macht es gemeinsam mit cross validation, wegen Overfitting

Optimierung der Hyperparameter
Grid Search

Einfachste Methode: Grid Search

1. Man braucht keine Voraussetzungen über die Hyperparameter
2. Man wählt die Hyperparameter von Interesse aus
3. Man wählt Werte für jeden Hyperparameter aus

```
{reg_lambda: [0, 0.01, 0.1, 1, 5], # l2 regularization
learning rate: [0.005, 0.02, 0.1],
subsample: [0.6, 0.85, 1]}
```

4. Modell wird mit allen Wertekombinationen aus dem kartesischen Produkt trainiert, und die Metriken werden gespeichert
5. Man geht die Wohnung putzen und ein Kaffee trinken bis es fertig ist…

Optimierung der Hyperparameter Randomized Search

- Nicht alle möglichen Kombinationen von Hyperparameter werden ausprobiert
- Man kann selbst den Rechenaufwand begrenzen

Successive halving: man benutzt nur eine Untermenge des Trainingsdatensatzes, man verzichtet gleich auf Hyperparameterwertekombinationen, die schlechte Metriken ergeben

Optimierung der Hyperparameter
Bayesian Optimization

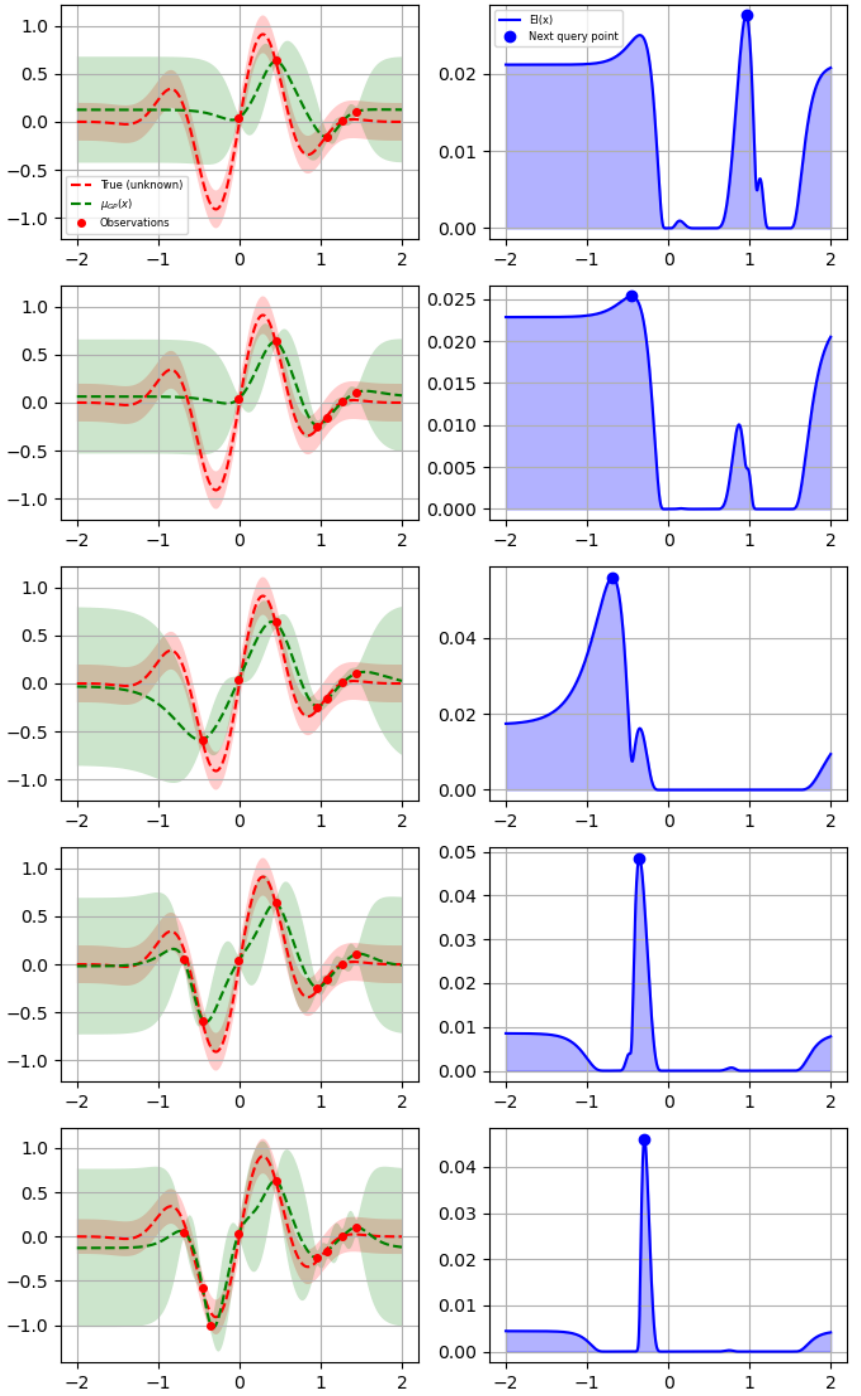
Idee: man betrachtet die Zielmetrik als eine Funktion der Hyperparameter

$$v = f(x)$$

Bewertung in jedem Punkt ist sehr teuer → Heuristiken

Bewertung in wenigen beliebigen Punkten

- davon ausgehend Aufbau einer Wahrscheinlichkeitsverteilung / eines Surrogatmodells, welches der nächste beste Punkt zu bewerten ist
- weiteres Aktualisieren der Schätzung auf Grund vom tatsächlichen Ergebnis
- Stoppbedingung: Anzahl von Iterationen oder Wertunterschied



Optimierung der Hyperparameter

- Manchmal: Gradient descent ist möglich

- In der Praxis ist es meistens besser, mit den Features zu arbeiten
- Hyperparameter tuning: letzter Schritt in der Optimierung
- Für große Modelle: immer langwierig, unabhängig von der Methode

Warum mehrere Algorithmen studieren?

David Wolpert, “The Lack of A Priori Distinctions Between Learning Algorithms,” Neural Computation 8, no. 7 (1996): 1341–1390.

"No free lunch" - theorem

Formeller Beweis, dass es keinen absolut besten Modell für alle Aufgaben und Datensätze gibt, kein "one size fits all".

In der Praxis: keine vollständigen Daten und Verständnis, keine unendliche Zeit und Rechenkraft. Welche ist die passende Vorgehensweise für die jeweilige Aufgabe?

Tendenzen in der Industrie

Aufgabe	Modell	Bemerkungen
Tabellarische oder strukturierte Daten	Boosted Trees, Lineare / logistische Regressionsmodelle	Bei Weitem der am meisten verbreitete Fall, wegen Erklärbarkeit, Leichtheit und Geschwindigkeit
Textverarbeitung	Transformers	
Bilderverarbeitung	CNNs	Tendenz ist, Fine-tuning und Anpassungen von vortrainierten Modellen zu benutzen
Zeitbasierte Daten, predictions	ARIMA & Co, LSTM, Boosted Trees	
Clustern, Dimensionsreduktion	k-means, PCA	Für höchst nichtlineare Datensätze: autoencoders
Empfehlungssysteme	Collaborative filtering, nearest neighbours	

Tendenzen in der Industrie

LLMs

- RAG (retrieval augmented generation): Erreicherung mit vorgegebenen Daten
- Aufbau von Chatbots und Texte low / no code
- Benutzt für Suche und Organisieren von Informationen in großen unstrukturierten Datenmengen
- Validierung: man läßt zwei LLMs miteinander "sprechen"

MLOps, Data Engineering

- Provisionierung der Infrastruktur, Anbindung an Datenquellen
- Management von Modellen
- CI / CD
- Integrierung als API

Im Allgemeinen

- Hybride Systeme