WWU
MÜNSTER

Behavioral Context Recognition

Praktikum Mustererkennung II

wissen.leben
WWU Münster

Daniel Beckmann, Thomas Poschadel, Tony Prange, Joschka Strüber

30. Juli 2019

Aufbau

1. Was wir bisher gemacht haben
2. Probleme und offene Fragen
3. Pläne für die Zukunft

1. Was wir bisher gemacht haben

2. Probleme und offene Fragen

3. Pläne für die Zukunft

Kennenlernen des Datensatzes und Benutzererkennung

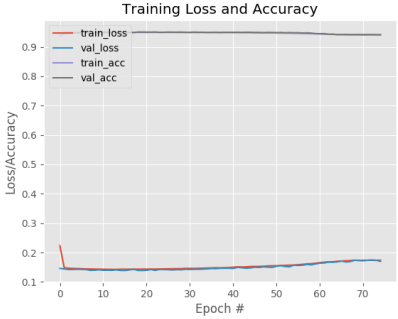
- Durcharbeiten des i-Python Notebooks von Vaizman
- Auslesen der Daten aus den Dateien und Verwendung der UUID als Label
- Benutzererkennung zunächst mit Random Forests, später mit XGBoost
- Sehr gute Ergebnisse: F1-Score von 0,999 bis 1,0

Klassifizierung mit Tensorflow

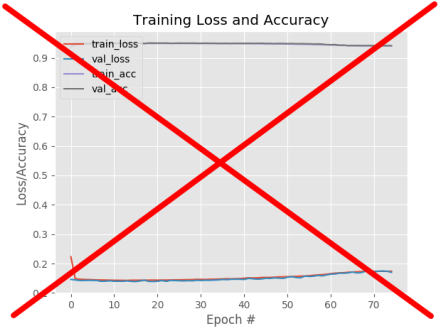
- Erstellung eines ersten Netzes
- Training auf dem gesamten Datensatz
- Erste Versuche der Multi-Label-Klassifizierung

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 64)	14464
activation_1 (Activation)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2080
activation_2 (Activation)	(None, 32)	0
dropout_1 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 64)	2112
activation_3 (Activation)	(None, 64)	0
dropout_2 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 512)	33280
activation_4 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 51)	26163
activation_5 (Activation)	(None, 51)	0

Klassifizierung mit Tensorflow - Erste Ergebnisse



Klassifizierung mit Tensorflow - Erste Ergebnisse



- Manuelle Verifikation deutet wesentlich schlechtere Resultate an
- Erste Klassifizierung möglich

Probleme:

- Gewichtung der NaN-Labels

Klassifizierung mit Tensorflow - Nächste Schritte

- Finden einer geeigneten Verlustfunktion
- Multi-Label-Evaluation
- Verwenden von Gewichten

Klassifizierung mit XGBoost

- Bibliothek für GPU-unterstützte und verteilte Berechnung von *Gradient Boosted Trees*

Vorteile:

- liefert gute Ergebnisse für tabulare Daten
- Scikit-learn API vorhanden → Verwendung der Scikit-learn Infrastruktur problemlos möglich, insbesondere *OneVsRestClassifier*
- gute Interpretierbarkeit → Bibliothek kann berechnete Bäume und *Feature Importances* als Plots ausgeben

Nachteile:

- hoher Berechnungsaufwand, insbesondere für Multilabel-Klassifizierung (Training von 51 Modellen)
- Hyperparametersuche schwierig

Hyperparametertuning mit Bayesian Optimization

- Training auf ganzem Datensatz dauert zu lange und ist auf GPU nicht möglich → für Hyperparametersuche Beschränkung auf jeweils fünf zufällig gewählte Attribute
- Suche guter Startpunkte für Hyperparameter mit *Randomized Search CV*
- Verfeinerung der Parameter mit *Bayesian Optimization*
  - Maximierung einer unbekannten Funktion durch Interpolation dieser anhand bekannter Startwerte und statistisch sinnvoll gewählten weiteren Parametersätzen
- Mögliches Problem: optimale Hyperparameter unterscheiden ggf. sich für verschiedene Label

1. Was wir bisher gemacht haben

2. Probleme und offene Fragen

3. Pläne für die Zukunft

Learning Rate und N_Estimators

- letzte Boosting-Runde in der Regel nicht die beste → verwende *Early Stopping*
- Parameterempfehlung für *Learning Rate*: 2 bis $10 \div \mathrm{n\_estimators}$
- Problem bei uns: beste Iteration ist immer die letzte, auch bei deutlich höheren Lernraten

NaN-Werte in den Labeln

- viele der Label des Datensatzes sind weder True noch False, sondern NaN (also fehlend)
- wir setzen fehlende Label aktuell auf False → Problem wird durch einige falsche Label schwerer
- in Vaizman et al. (2017) wurden NaN-Label bei Training und Testen ignoriert - wir sollen vermutlich genauso vorgehen?

Aufteilung Trainings- und Testdaten

- Vaizman et al. teilen die Trainings- und Testdaten auf Basis der einzelnen Benutzer auf → beim Testen werden nur Daten von zuvor unbekannten Personen betrachtet
- andere Möglichkeit: zufällige Aufteilung aller Daten von allen Personen → deutlich leichteres Problem
- Frage: welche Aufgabe sollen wir genau lösen?

1. Was wir bisher gemacht haben

2. Probleme und offene Fragen

3. Pläne für die Zukunft

Pläne und Ideen

- Verwendung der bisherigen Classifier und spätere Entscheidung welcher besser funktioniert
- Aufbereitung des Datensatzes
  - *Maximal Correlation Embeddings* [Li et al. 19], um fehlende Label in den Trainingsdaten zu ersetzen
  - Rekonstruktion fehlender Sensordaten mit einem *Adversarial Autoencoder* [Saeed et al. 18]
- Auslesen der *Feature Importances* und Entscheidungsbäume pro Label bei XGBoost (jeweils 51 Stück)