technische universität
dortmund

Ordinale Klassifikation mit neuronalen Netzen in DSEA

Nicolai Weitkemper

14.10.2022

Inhalt

Nominale und ordinale Daten

- Nominale Daten: *Kategorien*
  - z. B. „Elektron“, „Myon“, „Tauon“
  - *Keine* natürliche Ordnung
- Ordinale Daten: *Rangfolge*
  - z. B. „schlecht“, „mittel“, „gut“
  - Natürliche Ordnung
- → Neutrinoenergien sind *ordinale* Daten (sogar *metrisch*)

Vorgängerarbeiten

- DSEA$^+$ mit ordinaler Klassifikation: ✅
  - Vorteil: Konfidenzverteilungen physikalisch sinnvoll(er)
  - → siehe Jäkel[1]
- DSEA$^+$ mit neuronalen Netzwerken: ✅
  - Vorteil: Flexibel, evtl. bessere Performance
  - → siehe Haefs[2]
- DSEA$^+$ mit neuronalen Netzwerken *und* ordinaler Klassifikation: ⏳
  - → Thema dieser Arbeit

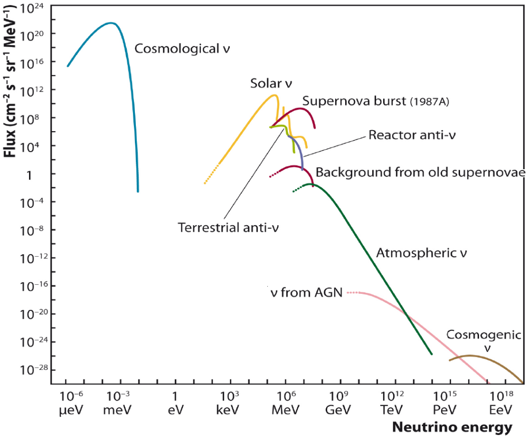
[1] Jäkel, „Ordinal Classification in DSEA".

[2] Haefs, „Lösungen inverser Probleme".

Übersicht

Neutrinos

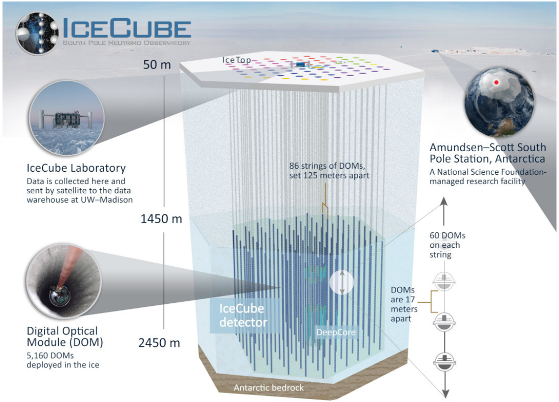
- Ungeladene, schwach wechselwirkende Elementarteilchen
- Botenteilchen
  - Unbeeinflusst von Magnetfeldern
  - Durchdringen Materie (fast) unbehindert
- Quellen:
  - Urknall (CNB)
  - Supernovae
  - Aktive Galaxienkerne
  - ...



Bildquelle: (Spiering, „Towards high-energy neutrino astronomy“)

IceCube

- Neutrino-Detektor am Südpol
- Indirekte Detektion mittels Cherenkov-Licht von Reaktionsprodukten
- Ziele:
  - Richtung / Quelle
  - Flavor
  - **Energie**
  - ...
- Energiebereich: GeV bis PeV[a]



[a] Aartsen u. a., „The IceCube Neutrino Observatory: instrumentation and online systems“.

Bildquelle: (IceCube Collaboration, *IceCube*)

Übersicht

Entfaltungsproblem I

- Unsere Messungen sind *indirekt*
  → Für eine physikalische Wahrheit $\boldsymbol{f(x)}$ können wir nur Messungen $\boldsymbol{g(y)}$ durchführen
- Fredholmsche Integralgleichung[1]:

$$\int_a^b A(x,y) f(x)\,\mathrm{d}x = g(y)$$

- Diskretisiert:

$$\boldsymbol{A}\vec{f} = \vec{g}$$

- Naive Lösung: Invertieren der Matrix $\boldsymbol{A}$:

$$\vec{f} = \boldsymbol{A}^{-1}\vec{g}$$

Entfaltungsproblem II

- Problem: $\boldsymbol{A}$ ist meist schlecht konditioniert
  → Numerisch instabil
  → Oszillationen
- Verschiedene verbesserte Ansätze:
  - Time-dependent Regularized Unfolding for Economics and Engineerings (TRUEE) / Regularized Unfolding (RUN)[1]
  - Iterative Bayesian Unfolding (IBU)[2]
  - DSEA$^+$

[1] Fredholm, „Sur une classe d'équations fonctionnelles".

[1] Milke u. a., „Solving inverse problems with the unfolding program TRUEE: Examples in astroparticle physics".

[2] D'Agostini, „A multidimensional unfolding method based on Bayes' theorem"; D'Agostini, „Improved iterative Bayesian unfolding".

DSEA$^+$ – Idee

- DSEA$^+$ = **D**ortmund **S**pectrum **E**stimation **A**lgorithm[1]
- Fasse das Entfaltungsproblem als *Klassifikationsproblem* auf
  - Diskretisiere die Zielvariable in Bins
  - Trainiere einen Klassifizierer darauf, zu einer gegebenen Messung den Energie-Bin zu finden
  - Prinzipiell eignet sich jeder Klassifizierer (z. B. Random Forests[2], NN'e, ...)
- Summiere die Konfidenzen auf, um das gesamte Energiespektrum zu erhalten
  → Besser als nur die jeweils wahrscheinlichsten Bin-Zuordnungen aufzusummieren
- Aktualisiere die Gewichte jedes Bins iterativ abhängig vom rekonstruierten Spektrum
- Ziel: Konvergenz & Unabhängigkeit vom Spektrum der Monte-Carlo-Trainingsdaten

[1] Bunse u. a., „Unification of Deconvolution Algorithms for Cherenkov Astronomy".

[2] Hymon und Ruhe, „Seasonal Variations of the Unfolded Atmospheric Neutrino Spectrum with IceCube".

DSEA$^+$ – Algorithmus

- Initialisierung
  - Nehme einen gleichverteilten Prior an: $\hat{\mathbf{f}}_i^{(0)} = \frac{1}{I} \quad \forall i$
  - Gewichtung s. u.
- Iteration
  - Training (gewichtet gemäß $w_i^{(k)}$)
  - Rekonstruiere das Test-Spektrum als Summe von Konfidenzen
  - Optionale Zwischenschritte (gekürzt):
    - (Adaptive) Schrittweite α
    - Regularisierung
  - Umgewichtung: $w_i^{(k+1)} = \frac{\hat{\mathbf{f}}_i^{(k)}}{\mathbf{f}_i^{\text{train}}}$
- Abbruch / Konvergenz:
  - Nach K Iterationen
  - Bei Unterschreiten einer χ^2-Distanz zur vorherigen Schätzung

Übersicht

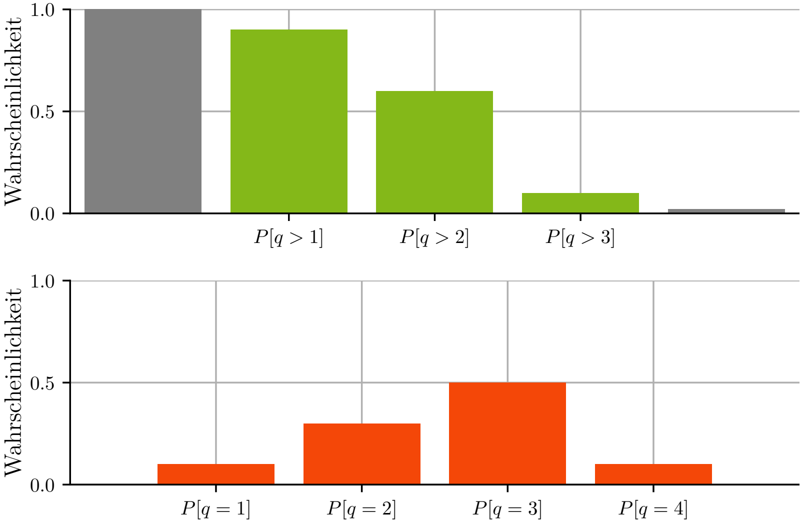
CORN – Funktionsweise

- CORN = **C**onditional **O**rdinal **R**egression for **N**eural Networks
- Unterteile die Aufgabe, den Rang-Index $q \in \{1, 2, 3, 4\}$ vorherzusagen, in binäre Teilaufgaben:
  $P[q > 1], \; P[q > 2 | q > 1], \; P[q > 3 | q > 2]$
- Berechne die unbedingten Wahrscheinlichkeiten:
  $\hat{P}[q > 2] = \hat{P}[q > 2 | q > 1] \cdot \hat{P}[q > 1]$,
  $\hat{P}[q > 3] = \hat{P}[q > 3 | q > 2] \cdot (\hat{P}[q > 2 | q > 1] \cdot \hat{P}[q > 1])$ usw.
  - Erzwingt $\hat{P}[q > 1] \geq \hat{P}[q > 2] \geq \hat{P}[q > 3] \geq \cdots$
- Vorhersage: $\hat{q} = 1 + \sum_k \mathbb{1}\left\{\hat{P}[q > k] > 0.5\right\}$

CORN – Ermittlung von Konfidenzen

- Bisher liefert CORN $P[q > 1]$, $P[q > 2]$ usw. und daraus eine Rang-Vorhersage $\hat{q}$.
- Aber wir benötigen $P[q = 1]$, $P[q = 2]$ usw.
- Unter der Annahme, dass es keine Energien außerhalb der gegebenen Bins gibt, gilt z. B. für $q \in \{1, 2, 3, 4\}$:
    - $P[q = 1] = 1 - P[q > 1]$
    - $P[q = 2] = P[q > 1] - P[q > 2]$
    - $P[q = 3] = P[q > 2] - P[q > 3]$
    - $P[q = 4] = P[q > 3]$

CORN – Beispiel



Übersicht

Datensatz

- Datensatz *11374*[1]
  - Monte-Carlo-Simulation
  - Nur „upgoing“ ν_μ
  - E^{-2}-Spektrum
  - 13 Mio. Ereignisse
    → 500 000 davon verwendet (vgl. Haefs[2])
  - 94 Features verfügbar
- Zielvariable: `MCPrimary.energy`
- 12 Features [3]
  - Selektion mit *MRMR* (Maximum Relevance — Minimum Redundancy)
  - Yeo-Johnson-Transformation (reduziert Schiefe) und Skalierung auf Standardabweichung

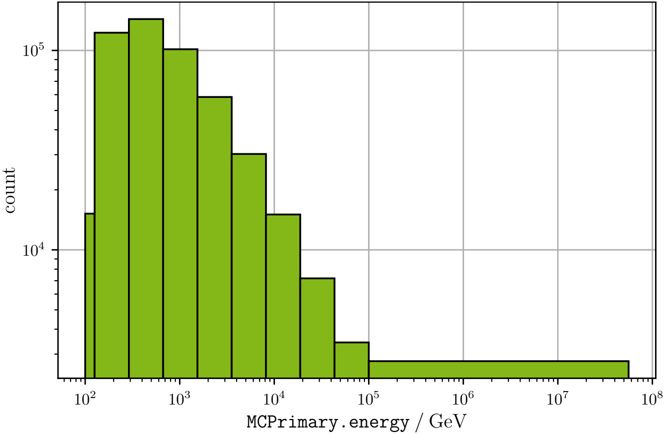
[1] IceCube Collaboration, *Dataset 11374*.

[2] Haefs, „Lösungen inverser Probleme“.

[3] übernommen aus Jäkel (Jäkel, „Ordinal Classification in DSEA“)

Bins

- 10 Bins
- Under-/Overflow-Bins
  - Untere Grenze: $\mathbf{10^{2.1}}$GeV
  - Obere Grenze: $\mathbf{10^{5}}$GeV
    (vgl. Haefs[a])



[a] Haefs, „Lösungen inverser Probleme".

Neuronales Netzwerk

- Form der Hidden Layers: **120, 240, 120, 12** fully connected (vgl. Haefs[1])
- Aktivierungsfunktion: `leaky ReLU`
- Output-Layer: von CORN bereitgestellt
  - Ergänzt um die Berechnung von Konfidenzen (statt des wahrscheinlichsten Ranges)
- Verlustfunktion: von CORN bereitgestellt
  - Ergänzt um eine Gewichtung der Beispiele
- Optimierer: Adaptive Moment Estimation (Adam)

[1] Haefs, „Lösungen inverser Probleme“.

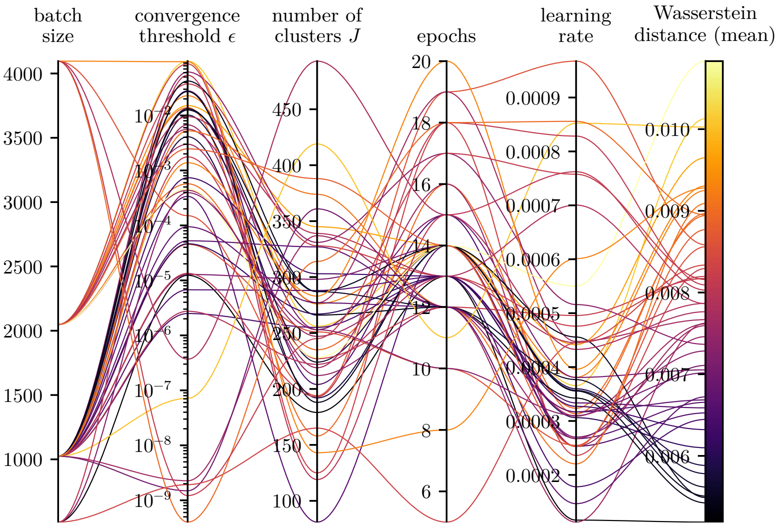
Weitere Hyperparameter

- Maximal $K = 20$ DSEA^{+}-Iterationen
  → nie erreicht
- Verwende adaptive Schrittweite [1]

[1] aus dem Python-Paket `CherenkovDeconvolution` (Bunse, *CherenkovDeconvolution.py*)

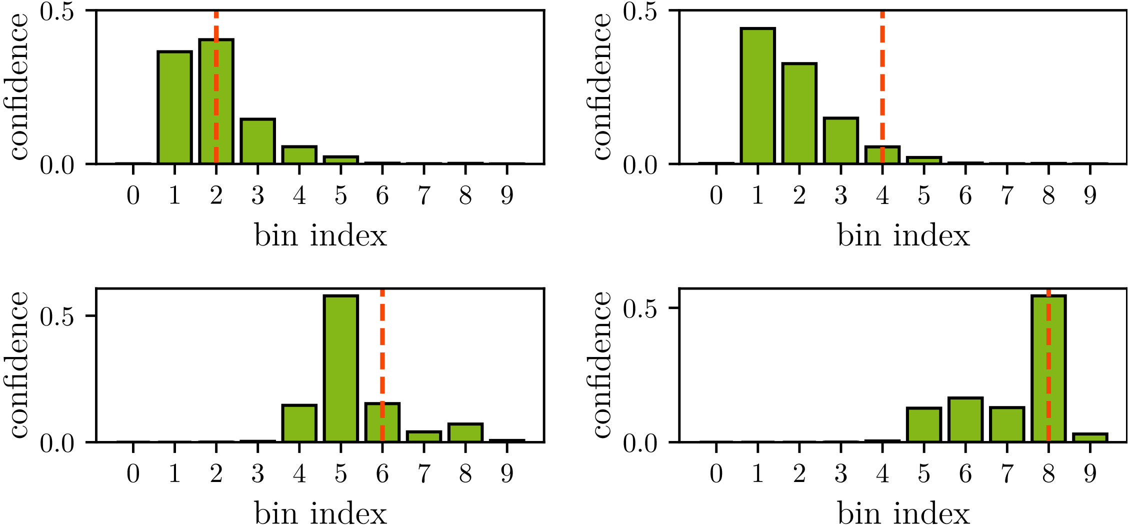
Übersicht

Bayesische Optimierung

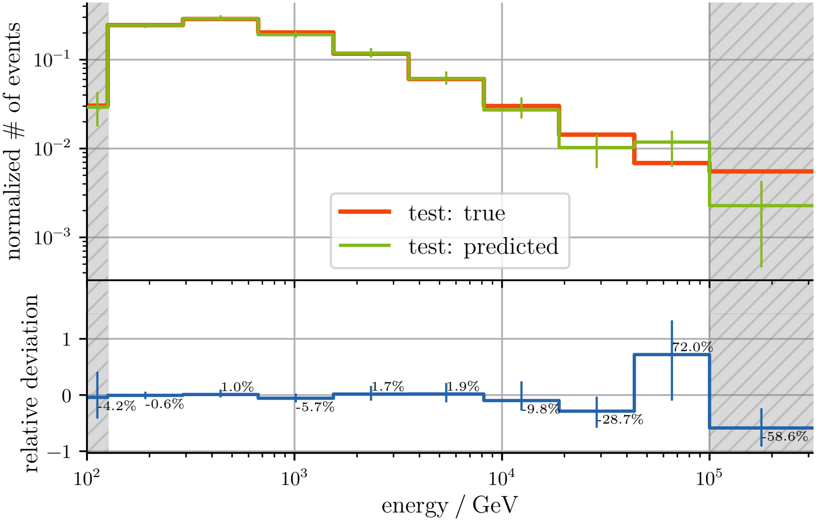


Übersicht

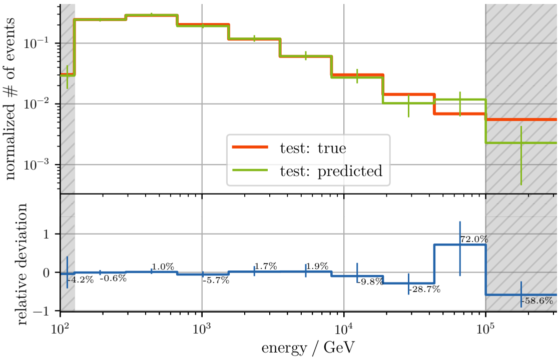
Konfidenzen für einzelne Ereignisse

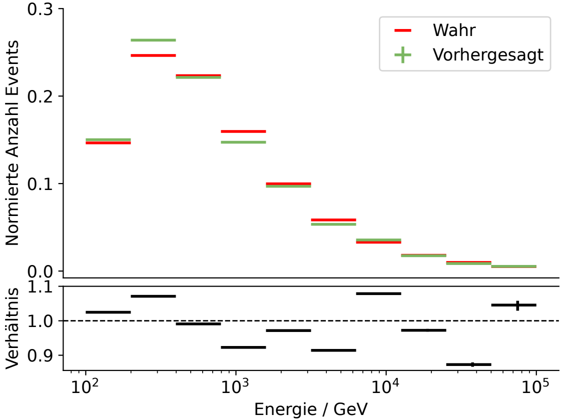


Spektrum


normalized # of events
10^{-1}
10^{-2}
10^{-3}
test: true
test: predicted
relative deviation
1
0
−1
-4.2%
-0.6%
1.0%
-5.7%
1.7%
1.9%
-9.8%
-28.7%
72.0%
-58.6%
10^2
10^3
10^4
10^5
energy / GeV

Spektrum – Vergleich mit Haefs[1]



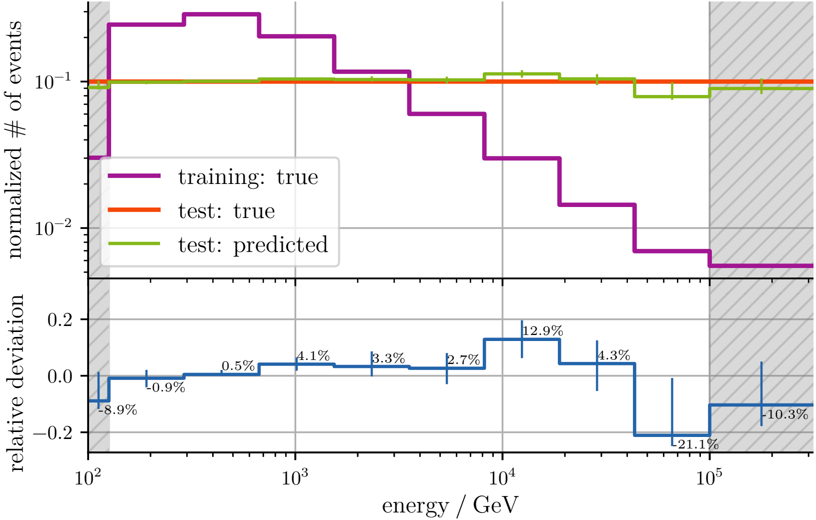


[1] Haefs, „Lösungen inverser Probleme“.

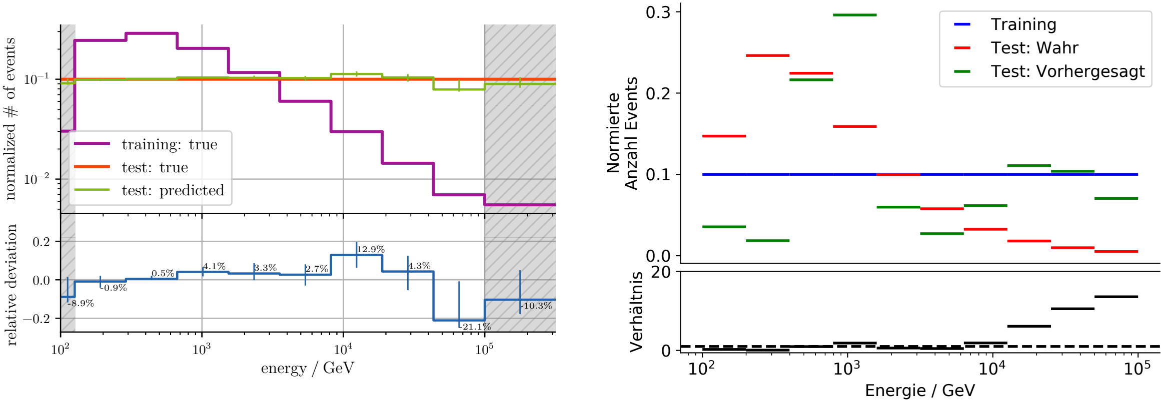
Bias

- Ein Klassifizierer könnte direkt das Trainings-Spektrum als Konfidenzen ausgeben
  → Perfekte Ergebnisse, aber nichts gelernt
  → Bias
- DSEA$^+$ soll den Bias durch (Um-)Gewichtung minimieren
- Ansatz zur Überprüfung:
  - Training auf E^{-2}-Spektrum (wie bisher)
  - Test auf einem gleichverteilten Spektrum

Bias


normalized # of events
10^{-1}
10^{-2}
training: true
test: true
test: predicted
relative deviation
0.2
0.0
−0.2
-8.9%
-0.9%
0.5%
4.1%
3.3%
2.7%
12.9%
4.3%
-21.1%
-10.3%
10^2
10^3
10^4
10^5
energy / GeV

Bias – Vergleich mit Haefs[1]



[1] Haefs, „Lösungen inverser Probleme“.

Übersicht

Zusammenfassung

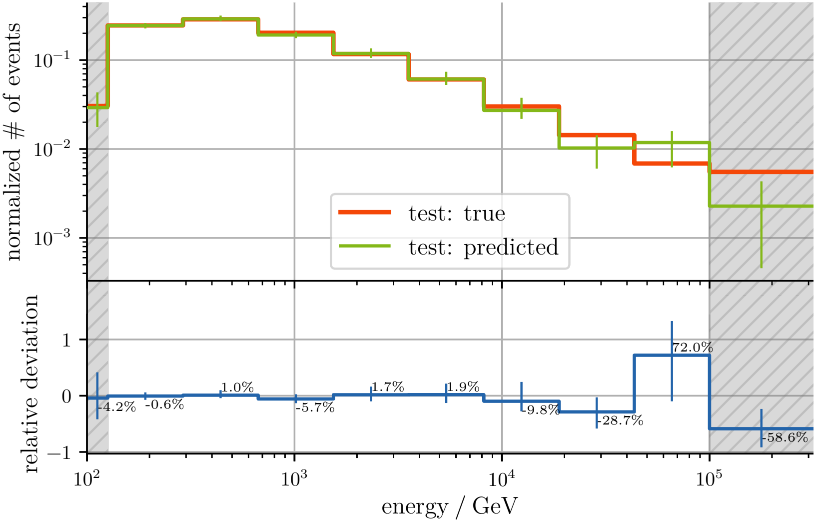
- Physikalisch sinnvolle Konfidenzverteilungen (mit Einschränkungen)
- Oszillationen in den höheren Energiebereichen
- CORN verbessert Bias drastisch (ggü. normalem NN)

Ausblick

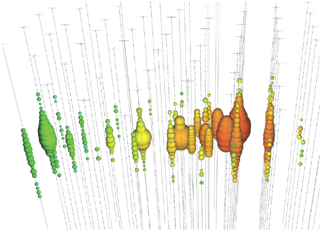
- (Explizite) Regularisierung
- Mehr (/gleichverteilte) Trainingsdaten
- Prüfung weiterer Hyperparameter, z. B. der Form des neuronalen Netzes
  → CORN schränkt nur die Form des Output-Layers ein
- Adaptiere DSEA$^+$ für metrische Daten / Regression
- Verwende Graph Neural Networks (GNNs) auf „rohen“ Daten (Zeit, Ort, Ladung)
  → Schlägt Boosted Decision Trees (BDTs) in Auflösung *und* Geschwindigkeit[1]

[1] Minh, „Reconstruction of Neutrino Events in IceCube using Graph Neural Networks“.

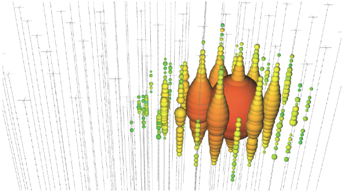
Ende des Vortrags



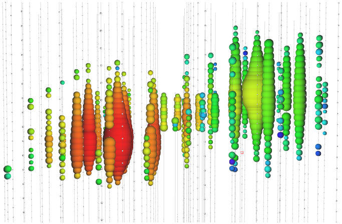
IceCube – Detektorsignaturen



Track / ν_μ



Cascade / ν_e



Double bang / ν_τ

DSEA$^+$ – Vorteile

- Beliebige Anzahl an Input-Variablen möglich
- Information über einzelne Ereignisse (/ Beiträge zum Spektrum) bleibt vollständig erhalten
  → Rekonstruktion von z. B. zeitabhängigen Spektren[1]

[1] Bunse, „DSEA Rock-Solid“.

[1] Quelle: Folien zu SMD-B

DSEA^{+} – Voraussetzungen

- Diskretisierte Energien (Bins)
- Der Klassifizierer muss Konfidenzen für jede Klasse zurückgeben
- Die Referenz-Implementierung[1] geht von einem **`sklearn`**-Klassifizierer aus

[1] Bunse, *CherenkovDeconvolution.py*.

CORN – Ermittlung von Konfidenzen – „Beweis“ I

- Es gebe 3 Energie-Klassen/Bins, genannt $\mathbf{1, 2, 3}$
- CORN’s Output-Layer gibt uns (nach Anwendung von `sigmoid` und einem kumulativen Produkt) zwei Wahrscheinlichkeiten:
  - $P[y > 1]$
  - $P[y > 2]$
- **Gesucht:** $P[y = 1], P[y = 2], P[y = 3]$

CORN – Ermittlung von Konfidenzen – „Beweis“ II

- Sei $y \in \{1, 2, 3\}$
- Dann gilt $P[y = 1] \iff \neg P[y > 1] = 1 - P[y > 1]$
-

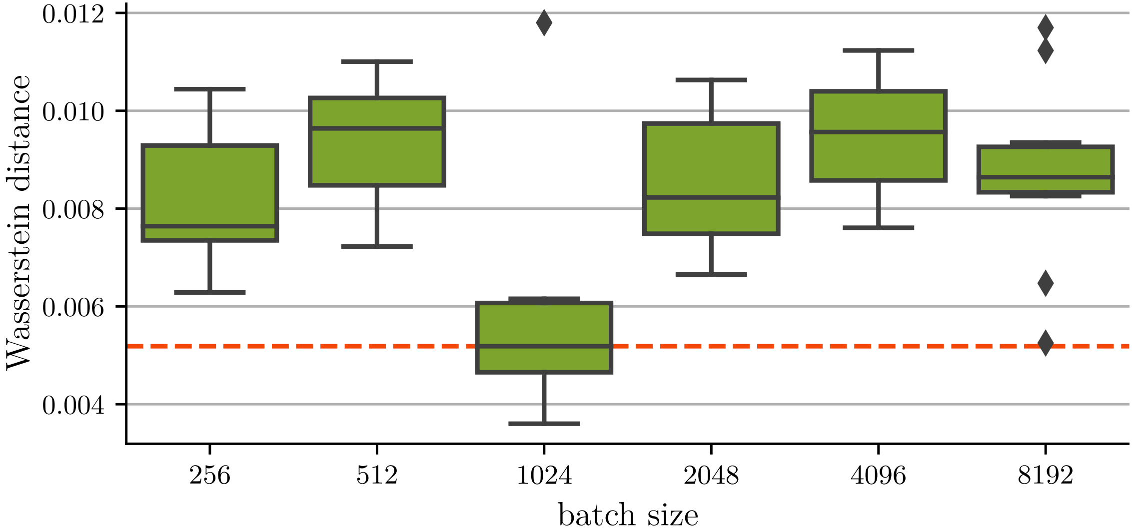
$$P[y = 2] \iff \neg P[y \neq 2] \iff \neg(P[y < 2] \lor P[y > 2]) \iff \neg(P[y = 1] \lor P[y = 3])$$
$$= 1 - ((1 - P[y > 1]) + P[y = 3])$$
$$= P[y > 1] - P[y > 2]$$

- $P[y = 3] \iff P[y > 2]$
- Fertig! ✅🎉

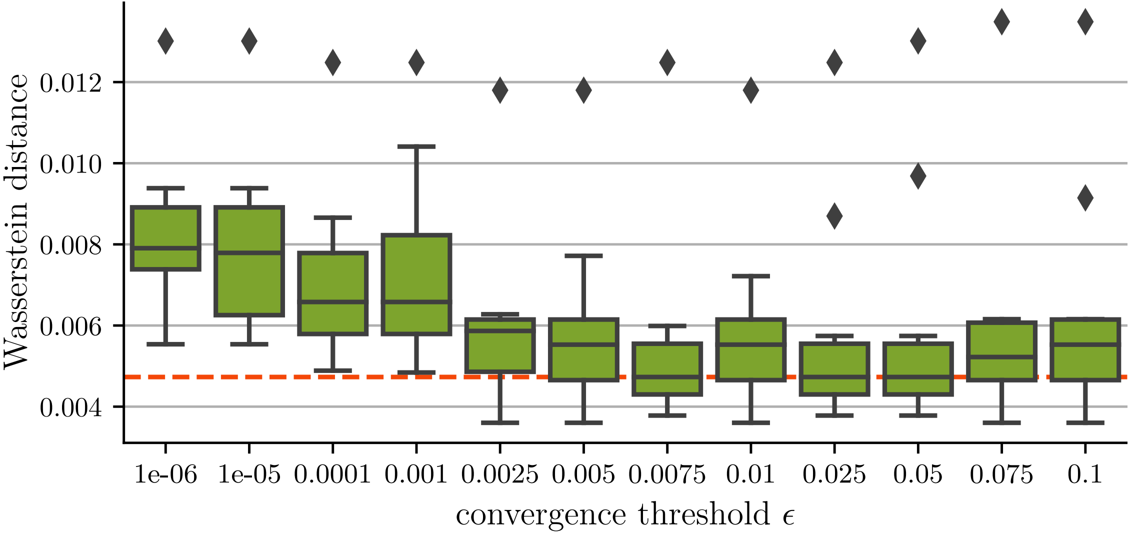
Liste von Features

- MCPrimary.energy
- SplineMPEDirectHitsICE.n_dir_doms
- VariousVariables.Cone_Angle
- SplineMPECramerRaoParams.variance_theta
- Borderness.Q_ratio_in_border
- SplineMPETruncatedEnergy_SPICEMie_BINS_MuEres.value
- SplineMPETruncatedEnergy_SPICEMie_DOMS_Neutrino.energy
- SplineMPEDirectHitsICB.n_late_doms
- Dustyness.n_doms_in_dust
- LineFitGeoSplit1Params.n_hits
- SplineMPEDirectHitsICC.dir_track_hit_distribution_smoothness
- SPEFit2GeoSplit1BayesianFitParams.logl
- SplineMPECharacteristicsIC.avg_dom_dist_q_tot_dom

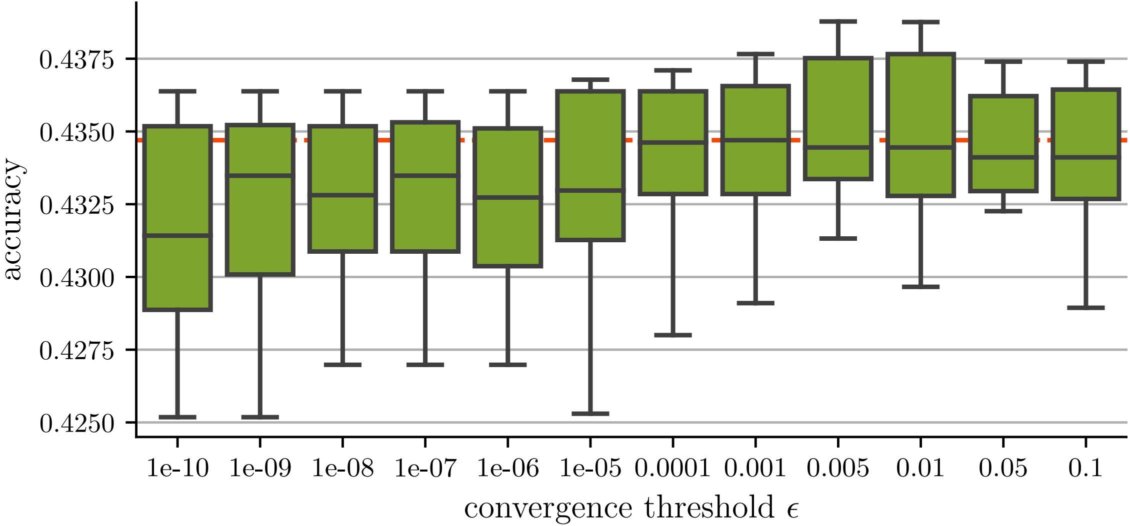
Hyperparametersuche – *batch size*



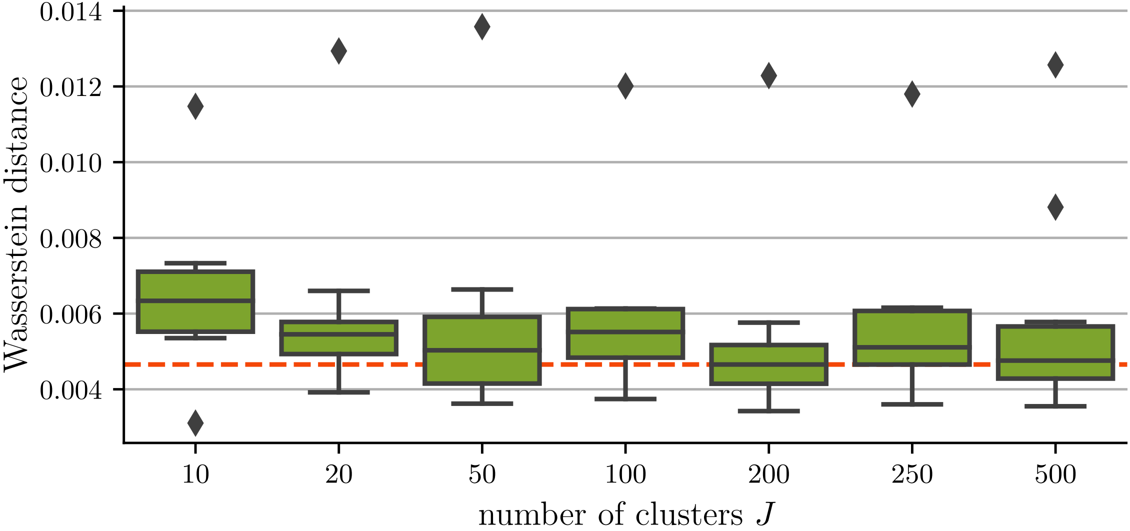
Hyperparametersuche – Konvergenzschwelle ϵ



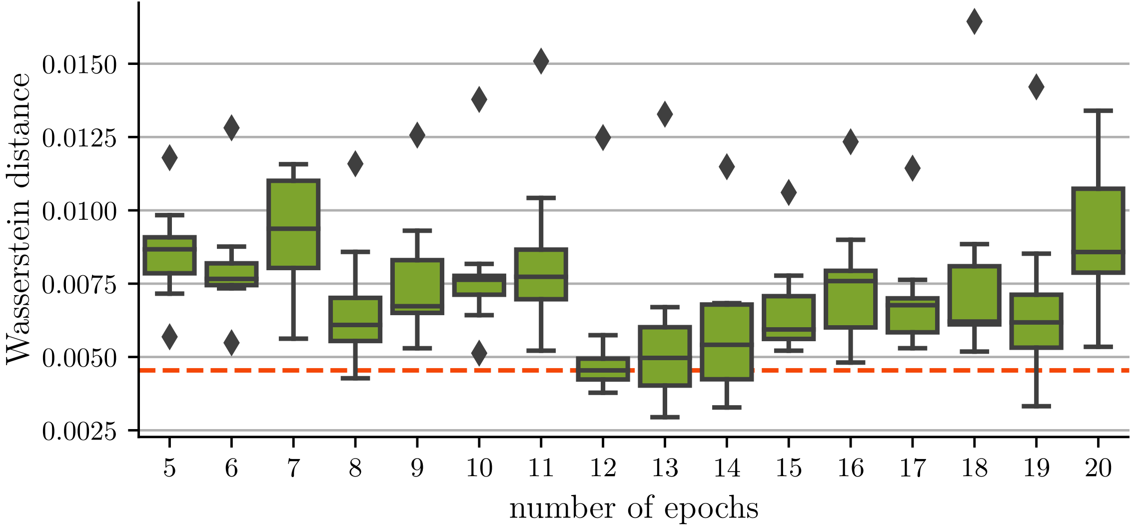
Hyperparametersuche – Konvergenzschwelle ϵ



Hyperparametersuche – Clusteranzahl J



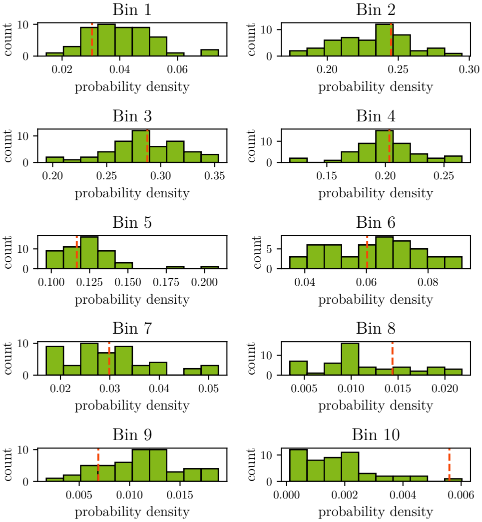
Hyperparametersuche – Epochenzahl



Bootstrapping

- Ziel: Schätze Unsicherheit der Entfaltung ab
- Vorgehen (iteriert):
  - Ziehe (mit Zurücklegen) Beispiele aus dem ursprünglichen Datensatz
  - Trainiere damit das Modell
  - Evaluiere jeweils auf den Beispiele, die nicht gezogen wurden
- Bestimme Median/Quantile

Bootstrapping – Verteilungen pro Bin



Vergleiche

- Jäkel[1]:
  - ⚠️ 12 Bins
  - ⚠️ Ohne Under-/Overflow-Bins
  - ⚠️ Ohne adaptive Schrittweite
  - Ähnliche Wasserstein-Distanz (0.0108 vs. 0.008 79)
  - Keine strikte Unimodalität der Konfidenzverteilungen
- Haefs[2]:
  - ⚠️ Ohne Under-/Overflow-Bins
  - ⚠️ Ohne adaptive Schrittweite
  - ⚠️ Bias-Test anders herum
  - ⚠️ Bootstrapping nur auf den Test-Daten
  - Etwas bessere Genauigkeit (42.7 % vs. < 39 %)
  - Etwas besseres Spektrum für Energien $< 1 \times 10^5$ GeV
  - Deutlich weniger Bias

[1] Jäkel. „Ordinal Classification in DSEA“.