Studien zur Klassifikation der Events des ANTARES Neutrinoteleskops mit neuronalen Netzen

Horst Laschinsky
Physikalisches Institut I
Universität Erlangen-Nürnberg

horst.laschinsky@physik.uni-erlangen.de

Motivation

Daten des ANTARES Neutrinoteleskops lassen sich in verschiedene Ereignisklassen unterteilen:

- Myonen
- Schauer
  - Elektromagnetisch
  - Hadronisch
  - Doppelschauer (ν_τ)
- Untergrund

Für verschiedene Ereignistypen sind verschiendene Rekonstruktionsalgorithmen nötig.

Motivation

Daten des ANTARES Neutrinoteleskops lassen sich in verschiedene Ereignisklassen unterteilen:

- Myonen
- Schauer
  - Elektromagnetisch
  - Hadronisch
  - Doppelschauer (ν_τ)
- Untergrund

Für verschiedene Ereignistypen sind verschiendene Rekonstruktionsalgorithmen nötig.
$\Rightarrow$ Kenntnis des Ereignistyps ermöglicht die gezielte Auswahl des richtigen Algorithmus.

Motivation

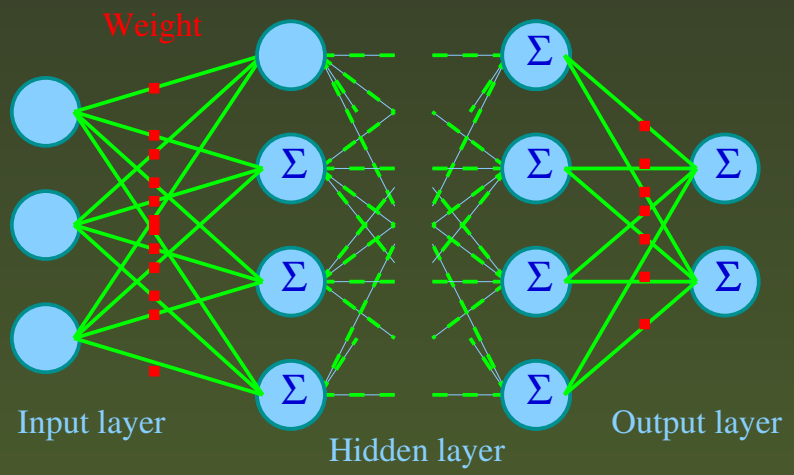
Daten des ANTARES Neutrinoteleskops lassen sich in verschiedene Ereignisklassen unterteilen:

- Myonen
- Schauer
  - Elektromagnetisch
  - Hadronisch
  - Doppelschauer (ν_τ)
- Untergrund

Für verschiedene Ereignistypen sind verschiendene Rekonstruktionsalgorithmen nötig.
$\Rightarrow$ Kenntnis des Ereignistyps ermöglicht die gezielte Auswahl des richtigen Algorithmus. Idee: Klassifikation der Ereignisse mit Hilfe von neuronalen Netzen.

Neuronale Netze - Überblick

- Muster liegt an der Eingangsschicht an
- Muster propagiert von einem Knoten zu den Knoten der nächsten Schicht und wird dabei gewichtet
- Knoten addiert die Werte auf, und schickt Signal an die Knoten der nächsten Schicht, falls ein bestimmtes Kriterium erfüllt ist.



- Anpassung der Gewichte anhand Soll-Ist-Differenz an der Ausgangsschicht ($\Rightarrow$ Backpropagation)
- Neues Muster wird an die Eingangsschicht angelegt
- Wiederholung bis Soll-Ist-Differenz für alle Muster unterhalb einer Toleranzgrenze

Auswertung von Events

Naiver Ansatz:

- Eingang: Ort (PMT) und Zeit (in ns)
- Ausgang: Ein Knoten pro Ereignisklasse

Auswertung von Events

Naiver Ansatz:

- Eingang: Ort (PMT) und Zeit (in ns)
- Ausgang: Ein Knoten pro Ereignisklasse

Aber:

- Rechenzeit $\propto \sum_i^{N_{\text{Layer}}-1} N_{\text{Knoten},i} \cdot N_{\text{Knoten},i+1}$

Auswertung von Events

Naiver Ansatz:

- Eingang: Ort (PMT) und Zeit (in ns)
- Ausgang: Ein Knoten pro Ereignisklasse

Aber:

- Rechenzeit $\propto \sum_i^{N_{\text{Layer}}-1} N_{\text{Knoten},i} \cdot N_{\text{Knoten},i+1}$
- 900 Photomultiplier, Zeitauflösung $\approx$ 1ns, typisches Event $\approx$ 5μs. $\Rightarrow$ 4.5 Mio Eingangsknoten

Auswertung von Events

Naiver Ansatz:

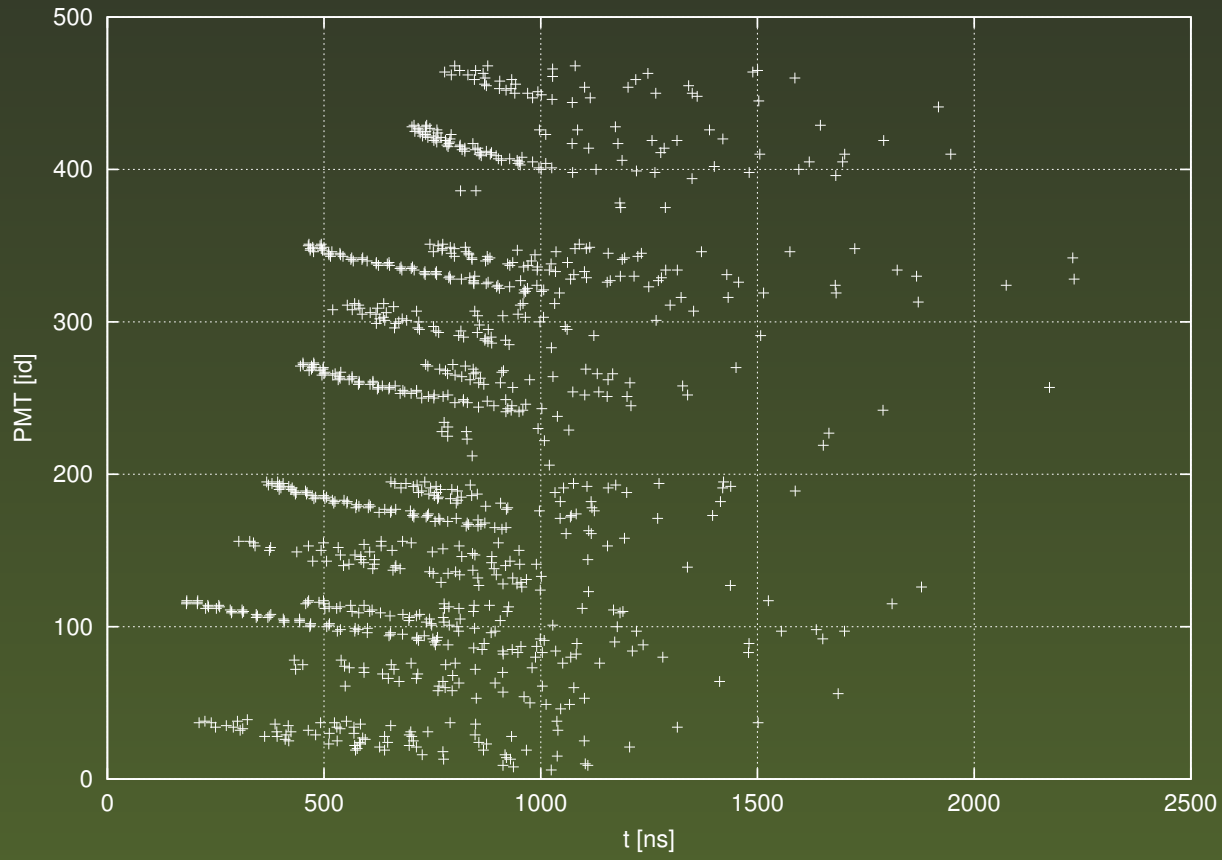
- Eingang: Ort (PMT) und Zeit (in ns)
- Ausgang: Ein Knoten pro Ereignisklasse

Aber:

- Rechenzeit $\propto \sum_i^{N_{\text{Layer}}-1} N_{\text{Knoten},i} \cdot N_{\text{Knoten},i+1}$
- 900 Photomultiplier, Zeitauflösung $\approx$ 1ns, typisches Event $\approx$ 5μs. $\Rightarrow$ 4.5 Mio Eingangsknoten
- Zuviel für heutige Computer. $\Rightarrow$ Kompression nötig

Auswertung von Events

Wie sehen die Muster überhaupt aus?



Idee: Charakteristische Muster $\Rightarrow$ Mustererkennungsalgorithmen aus der Bildbearbeitung könnten helfen.

Auswertung von Events

Es gibt eigentlich mehrere Grafiken,

Auswertung von Events

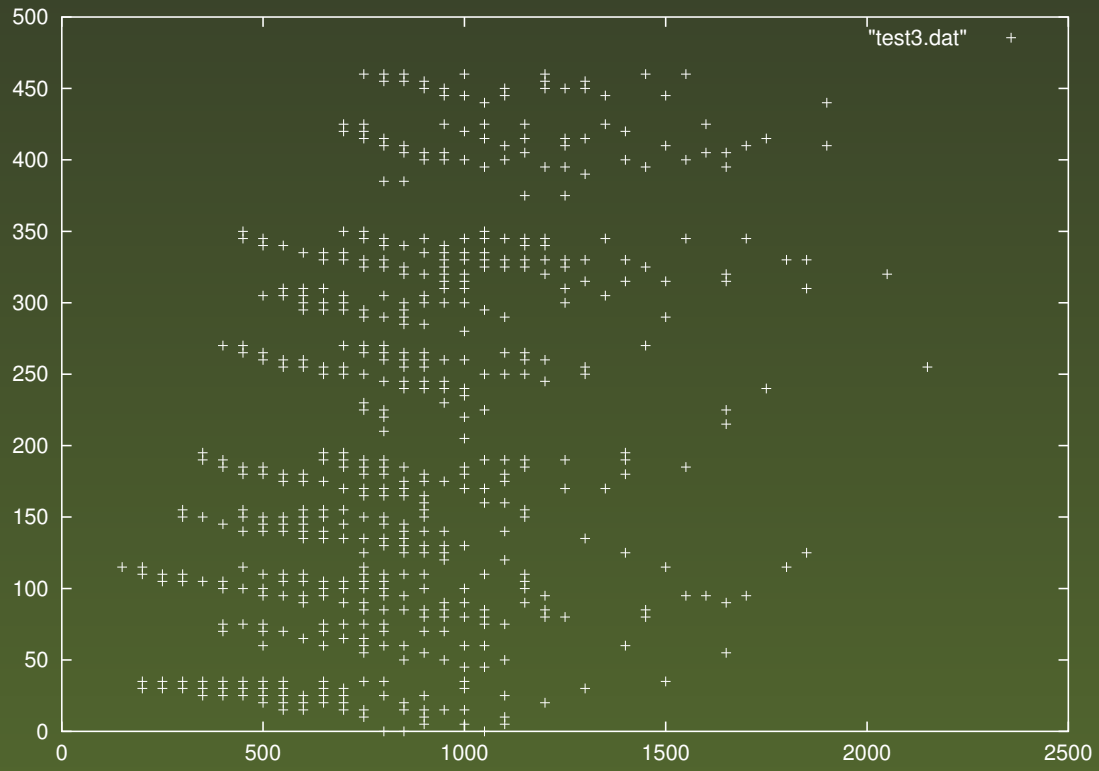
Es gibt eigentlich mehrere Grafiken, aber...

```
bash-2.05b$ ls -l
total 10
lrwxr-xr-x   1 htlaschi  htlaschi    29 Oct 10 23:18 data -> /automount/a8c/htla
schi/data/
drwxr-xr-x   2 htlaschi  htlaschi   512 Sep  4  2003 simul
drwxr-xr-x  13 htlaschi  htlaschi   512 Aug 24 14:29 src
drwxr-xr-x   6 htlaschi  htlaschi   512 Nov 24  2003 tex
drwxr-xr-x   2 htlaschi  htlaschi  1024 Jun 26  2003 tmp
drwxr-xr-x   2 htlaschi  htlaschi   512 Jun 24  2003 tools
bash-2.05b$ ls -l data/
ls: data/: No such file or directory
bash-2.05b$
```

...hier gibts kein Internet, und ich bin einfach unfähig...

Auswertung von Events

1. Ansatz: Genaue (Raum-Zeit) Position ist wichtig für Track-Rekonstruktion, aber nicht für Ereignisklassifiation. $\Rightarrow$ Mittelung über Raum-Zeit-Punkte möglich.



Reduktion um ca. den Faktor 800, jedoch immer noch zuviel.

Auswertung von Events

2. Ansatz: Standardverfahren Kompression durch *Principal Component Analysis* (PCA).

- Gegeben: ein Satz von N Mustern aus M Datenpunkten
- Betrachte ein Muster als M-Dimenstionalen Vektor p
- Berechne Durchschnittsvektor $\hat{p} = \frac{1}{N} \sum_{i=1}^{N} p_i$
- Berechne Kovarianz-Matrix $A = \frac{1}{N} \sum_{i=1}^{N} (p_i - \hat{p})(p_i - \hat{p})^T$
- Betrachte die Eigenvektoren von A als neue Basisvektoren
- Rekonstruiere p aus den Eigenvektoren, verwende dabei jedoch nur diejenigen K (K$\ll$M, hier K=50) Stück mit den höchsten Eigenwerten
- Verwende die K Koeffizienten als neue Eingangsdaten für das neuronale Netz.

$\Rightarrow$ Reduktion der Daten von 1500 Punkten auf 50 Punkte.

Ausgangsschicht

- Pro Ereignissklasse ein Knoten
- Ausgangswert zwischen 0 und 1 $\hat{=}$ Wahrscheinlichkeit der dazugehörigen Ereignisklasse.

Bisher verwendet: Monte-Carlo-Simulationen von 4 verschiedenen Klassen $\Rightarrow$ 4 Ausgangsknoten:

- EM charged current
- EM neutral current
- Myon charged current
- Myon charged current, contained Event

Der Algorithmus

- Kompression durch Mittelung üeber Raum-Zeit-Koordinaten
- Weitere Kompression durch PCA
- Training des Netzes

Erste Ergebnisse

```
k: 0 run: 1 event: 190 0.607198 0.419914 0.389648 0.426990
k: 0 run: 1 event: 14 0.389701 0.419915 0.426952 0.607192
k: 0 run: 1 event: 79 0.610513 0.419914 0.418328 0.389701
k: 0 run: 1 event: 69 0.389472 0.607947 0.420273 0.426938
k: 0 run: 1 event: 135 0.426992 0.573620 0.419832 0.418331
k: 0 run: 1 event: 47 0.418317 0.392737 0.607958 0.420271
k: 0 run: 1 event: 210 0.608112 0.426216 0.389643 0.419830
k: 0 run: 1 event: 98 0.389636 0.418493 0.426952 0.607954
k: 0 run: 1 event: 109 0.426955 0.420238 0.607195 0.389648
k: 0 run: 1 event: 56 0.418328 0.419840 0.573798 0.426951
k: 0 run: 1 event: 31 0.420271 0.607957 0.392717 0.418326
```

Sieht gut aus...

Erste Ergebnisse

```
k: 0 run: 1 event: 190 0.607198 0.419914 0.389648 0.426990
k: 0 run: 1 event: 14 0.389701 0.419915 0.426952 0.607192
k: 0 run: 1 event: 79 0.610513 0.419914 0.418328 0.389701
k: 0 run: 1 event: 69 0.389472 0.607947 0.420273 0.426938
k: 0 run: 1 event: 135 0.426992 0.573620 0.419832 0.418331
k: 0 run: 1 event: 47 0.418317 0.392737 0.607958 0.420271
k: 0 run: 1 event: 210 0.608112 0.426216 0.389643 0.419830
k: 0 run: 1 event: 98 0.389636 0.418493 0.426952 0.607954
k: 0 run: 1 event: 109 0.426955 0.420238 0.607195 0.389648
k: 0 run: 1 event: 56 0.418328 0.419840 0.573798 0.426951
k: 0 run: 1 event: 31 0.420271 0.607957 0.392717 0.418326
```

Sieht gut aus... bis ein Fehler im Algorithmus entdeckt wurde...

Erste Ergebnisse

Nach der Bereinigung des Fehlers

```
k: 0 run: 6 event: 103 0.420271 0.389632 0.419930 0.610309
k: 0 run: 6 event: 50 0.419831 0.426953 0.607946 0.389484
k: 0 run: 6 event: 213 0.419932 0.607197 0.389647 0.426989
k: 0 run: 6 event: 167 0.419910 0.389700 0.426951 0.607192
k: 0 run: 6 event: 216 0.419914 0.426938 0.418328 0.573799
k: 0 run: 6 event: 208 0.419911 0.418328 0.420271 0.581507
k: 0 run: 6 event: 227 0.419911 0.420270 0.419829 0.579767
k: 0 run: 6 event: 232 0.419911 0.419829 0.419930 0.580162
k: 0 run: 6 event: 81 0.419911 0.419930 0.419907 0.580073
k: 0 run: 6 event: 212 0.419911 0.607949 0.419912 0.392051
k: 0 run: 6 event: 183 0.419912 0.389643 0.419911 0.610357
Segmentation fault (core dumped)
bash-2.05b$
```

Well, nobody’s perfect...

Ausblick

- Fehler suchen... :-)

Ausblick

- Fehler suchen... :-)
- Das ganze mit Untergrundsimulation

Ausblick

- Fehler suchen... :-)
- Das ganze mit Untergrundsimulation
- Performance-Analyse

Ausblick

- Fehler suchen... :-)
- Das ganze mit Untergrundsimulation
- Performance-Analyse
- Performance verbessern (Inline-Assembler, LAPACK o.ä.)

Ausblick

- Fehler suchen... :-)
- Das ganze mit Untergrundsimulation
- Performance-Analyse
- Performance verbessern (Inline-Assembler, LAPACK o.ä.)
- Anpassung auf ROOT-Dateiformat

Ausblick

- Fehler suchen... :-)
- Das ganze mit Untergrundsimulation
- Performance-Analyse
- Performance verbessern (Inline-Assembler, LAPACK o.ä.)
- Anpassung auf ROOT-Dateiformat
- evtl. Ausbau zur Echtzeit-Analyse

Ausblick

- Fehler suchen... :-)
- Das ganze mit Untergrundsimulation
- Performance-Analyse
- Performance verbessern (Inline-Assembler, LAPACK o.ä.)
- Anpassung auf ROOT-Dateiformat
- evtl. Ausbau zur Echtzeit-Analyse
- Dankefein fürs zuhören!