STRETNUTIE SO ZADÁVATEĽOM – prepis audia

Stretnutie: 4.10.2019
Čas: 10:00

Projekt je zameraný na rekonštrukciu top kvarkových párov.

VYSVETLENIE

Je to proces vo fyzike kedy vznikajú 2 častice, ktoré sa volajú top kvark. Častice sa následne rozpadnú. Je tam nejaký rozpadový diagram. Jeden top kvark sa rozpadne na 3 častice, čiže máme **aspoň** 6 častíc.
Prečo **aspoň** 6 a nie práve 6?
Zo šiestich častíc, môžu byť niektoré kvarky, ktoré vyžarujú nejaké ďalšie častice, ktoré sa volajú gluóny. Tieto častice sa nemerajú, ale meria sa odozva v detektore.
V detektore potom nevidieť jednotlivé častice , ale napr. pre kvarky vidíme JETY – spŕšky nejakých častíc.
Detektor meria týchto 6 objektov -> potrebujeme ich spárovať so šiestimi časticami, na ktoré sa rozpadli dva kvarky.
Toto párovanie sa tam robí na základe toho, že z objektov, ktoré nameriame, rekonštruujeme nejaké vlastnosti top kvarkov napr. hmotnosť a to pomáha pri určovaní, či sme to dobre spárovali. Vďaka Monte Carlo simulácii to vieme takto otestovať (či je to naozaj dobre spárované). Vieme ohodnotiť nejaké kombinácie alebo permutácie spárovania na základe vlastností top kvarkov a na základe Monte Carlo simulácií to vieme naozaj overiť.
V detektore vieme teda merať vznik JETOV, jeho vlastnosti teda **uhol1, uhol2, energia a hybnosť.**
OTÁZKA: Ktorý JET patrí ktorému KVARKU ?
My **dostaneme** len **zoznam** JETOV(jet1, jet2, jet3...) a ich vlastností, ale priradenie ku kvarkom nevieme určiť.

RIEŠENIE:

1. metóda – KLFitter

Máme nejaký algoritmus. Kód presne urobí to, že zoberie permutáciu, zráta pravdepodobnosť, zoberie druhú permutáciu, zráta pravdepodobnosť ... a potom nám povie, že táto permutácia, toto zoradenie napr.: 1, 2, 4, 3 je správne.
Bežne sa to robí metódou, ktorá ráta likelihood, čo je funkcia, ktorá pre danú permutáciu vyhodí pravdepodobnosť, že to sedí (využíva sa tam fyzikálna informácia(nie je to potrebné vedieť)).
(Táto metóda existuje. Je to nakódené v C++ na githube (zadávateľ poskytne daný kód))

2. metóda - Neurónové siete

Vieme pomocou nejakých simulácií zistiť ako to je správne. Nebudeme rátať likelihood z nejakých fyzikálnych veličín, ale nasimulujeme si to tak, že vieme čo je a čo nie je správne. Teda **máme zoznam**, ktoré permutácie vyzerajú správne a ktoré nie.
Pointou tejto metódy je natrénovať neurónové siete tým, že zadáme napr.: JET1 má takéto uhly, energiu... a nech sa neurónová sieť naučí rozpoznávať tie, kedy je to správne voči tým kedy to správne nie je.

HLAVNÁ POINTA (Základná myšlienka)

- Natrénovať neurónovú sieť
- Po natrénovaní použiť NS reálne, teda zrátať, v koľkých prípadoch sme to naozaj trafili dobre
- Urobiť INTERFACE kde sa tieto veci budú spúšťať a testovať, aby sme to vedeli ľahko vyhodnocovať

(budeme mať na vstupe jednu zrážku (nazývame EVENT), týchto zrážok je nasimulovaných veľmi veľa, a pre jednu zrážku budeme mať zoznam JETOV)
(Súbory s nasimulovanými dátami budú dodané, súbory sú vo formáte ROOT)
(Dá sa to čítať v C++ a v Pythone (je na to urobený interface aj v C++ aj v Python))
(vlastnosti (údaje) sú jednoduchý formát – float)
(File (ROOT formát) – rôzne premenné, je to tabuľka)
(riadky sú Eventy (zrážky), stĺpce – premenné)

WEBOVSKÝ INTERFACE

Používateľ si vyberie, ktorú z dvoch metód chce využívať.

Buď **prvá** -> KLFitter -> Kinematic Likelihood Fitter (nakódená na githube)
KLFitter bude mať možnosť iba validovať.

Alebo **druhá** -> Neurónové siete
Táto metóda má 2 kroky – Natrénovať a Validovať

TRÉNOVANIE NEURÓNOVEJ SIETE

Používateľ si môže vybrať -> ak chce NS, dostane FILE (ROOT formát), kde bude natrénovanie, teda nejaký signál kde to vyzerá správne -> permutácie, a takisto aj ako to vyzerá nesprávne (rovnaké údaje, rovnaké typy premenných aj v jednom aj v druhom)

Všetky údaje sa vložia do KERASU. Musíme si vybrať aký chceme FILE a aj aké chceme premenné (v nastaveniach v KERASE sa dá kliknúť aké chceme premenné, počet neurónov...)
Ak máme vybraté všetky tieto veci, vygeneruje to CONFIG, ktorý bude human readable.
Je jedno v akom to bude formáte (json...).
Spustí sa samotné trénovanie (musí to prečítať rootovský FILE, takže by bolo lepšie ak to bude napísané v Pythone pretože aj KERAS je v Pythone).
Spustí sa trénovanie NS. Zavolá KERAS (zavolá jeho trénovanie).
Ak to zbehne, tak KERAS sám by mal ponúkať nejaké ploty (obrázky) či to skovergovalo to trénovanie -> základné obrázky, ktoré sa zobrazia na webe, aby sme zistili či je to správne alebo nie.
Tieto údaje by sa mali ukladať niekde v **DATABÁZE** -> aby sme mohli porovnávať, nastavenie ktoré sme na začiatku zvolili s výsledkami a pod.

VALIDÁCIA

- Spoločný krok pre KLFitter a Neurónové siete

Takisto pôjdeme cez FILE, musíme zbehnúť permutácie vhodiť do NS, to nám povie, ktorá permutácia je správna a ten samotný FILE má uložené, že ktorý je správny napr. index 4 je ten správny pre tento JET a pod. – porovná či sa to trafilo.

Výstup z validácie musí byť **len tabuľka** – koľkokrát to trafilo správne, ako dlho to bežalo(real time, CPU time)
Takisto ako natrénovanie sa aj validácia **uloží do databázy.**

To isté aj pre KLFitter

KLFitter – funkčný kód, ktorý sa používa v časticovej fyzike

Z informatického hľadiska je zle nakódený. Pomalé veľmi neefektívne kroky čo sa týka permutácií. V tomto kóde sa urobí tabuľka permutácií potom sa maže a pod. Je to neefektívne pri predstave, že už výpočty pri 6! trvajú dlho a ak by sme to mali volať milión krát bolo by to takisto veľmi zdĺhavé..

ZRÝCHLIŤ EXISTUJÚJI KÓD!

Na githube sú poskytnuté nejaké nápady a myšlienky ako to zrýchliť.

TECHNICKÉ VECI

- Trénovanie trvá veľmi dlho – hodiny, dni
- Teda ak to spustíme musí to niečo **monitorovať**!
- Bude poskytnutý testovací FILE, ktorý zbehne okamžite aby sme technicky zistili či to je dobre.
- Určite nie výstup GUI! – robí sa to na serveroch – najlepšie **príkazový riadok**
- Všetko by malo byť textový výstup
- Vytvorí to preto CONFIG, ktorý je textový súbor