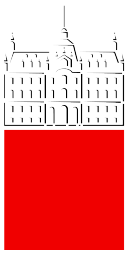
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Seminar - 3. letnik, I. stopnja

Uporaba metod strojnega učenja pri analizi podatkov na LHC

Avtor: Jan Gavranovič

Mentor: prof. dr. Borut Paul Kerševan

Ljubljana, 2020

Povzetek

Strojno učenje z razvojem strojne opreme in same optimalnosti algoritmov postaja vse bolj pomembno orodje v eksperimentalni fiziki delcev, v kateri ne manjka podatkov, na katerih se lahko algoritmi učijo. V seminarju naprej opišem delovanje LHC-ja, ter nato še fiziko, ki jo raziskuje. Sledi uvod v strojno učenje, ter predstavitev globokega učenja preko globokih usmerjenih nevronskih mrež (DNN). Na koncu predstavim še konkreten zgled analize na detektorju ATLAS, ki obravnava nastanek parov Higgsovih bozonov v dileptonskem razpadnem kanalu $WWbb$ s pomočjo nevronskih mrež.

Kazalo

1 Uvod v LHC

Veliki hadronski trkalnik (LHC) je pospeševalnik namenjen za trkanje žarkov protonov z energijo $\sqrt{s} = 13$ TeV v težiščnem sistemu, trka pa lahko tudi težke svinčeve ione [32]. LHC je največji pospeševalnik na svetu, tako po velikosti, kot po doseganju najvišjih energij. Na pospeševalniku je več detektorjev. Največji je ATLAS, takoj za njim pa je CMS [8, 12]. Detektorja raziskujeta procese, ki potekajo pri trkih med protoni v LHC-ju. Namenjena sta raziskovanju napovedi Standardnega modela, ki opisuje naše trenutno razumevanje gradnikov snovi in interakcij med njimi, ter tudi njegovim razširitvam. Velik izziv pri eksperimentih v fiziki osnovnih delcev je analiza in interpretacija velikih količin podatkov, ki se zabeležijo pri trkih. Analizo otežuje majhno razmerje med signalom in šumom (znani in neznani procesi), kompleksnost, visoka dimenzionalnost ter število podatkov, ki prihajajo iz detektorja. Eno od najmočnejših orodij za tako analizo je strojno učenje, ki samodejno prepoznava vzorce in povezave iz velikega števila podatkov. Poleg pospeševalnika in detektorja so pomembni še sistemi opisani v nadaljevanju.

1.1 Prožilni sistem

Trkajoči žarki, sestavljeni iz velikega števila protonov, se v LHC-ju zaletavajo med sabo s frekvenco do 40 MHz. Vsak trk lahko proizvede veliko število novih delcev, ki jih detektor ATLAS zaznava preko $150 \cdot 10^6$ senzorjev. Pri tem nastane vsako sekundo 1 PB podatkov, česar pa ni mogoče shraniti. Veliko število podatkov je potrebno zaradi tega, ker so trki, pri katerih nastanejo zanimivi produkti, zelo redki. ATLAS uporablja napredni prožilni sistem (*trigger*), ki pove, katere dogodke (katere trke protonov) je vredno shraniti. Sestavljen je iz dveh stopenj. Prva stopnja temelji na posebni elektroniki narejeni prav v ta namen. Druga stopnja uporablja računalniško analizo za še dodatno znižanje frekvenc in števila podatkov. Po obeh stopnjah na koncu ostane ~ 1000 zanimivih dogodkov vsako sekundo, ki se shranijo za nadaljnjo analizo s hitrostmi prenosa podatkov ~ 1 GB na sekundo.

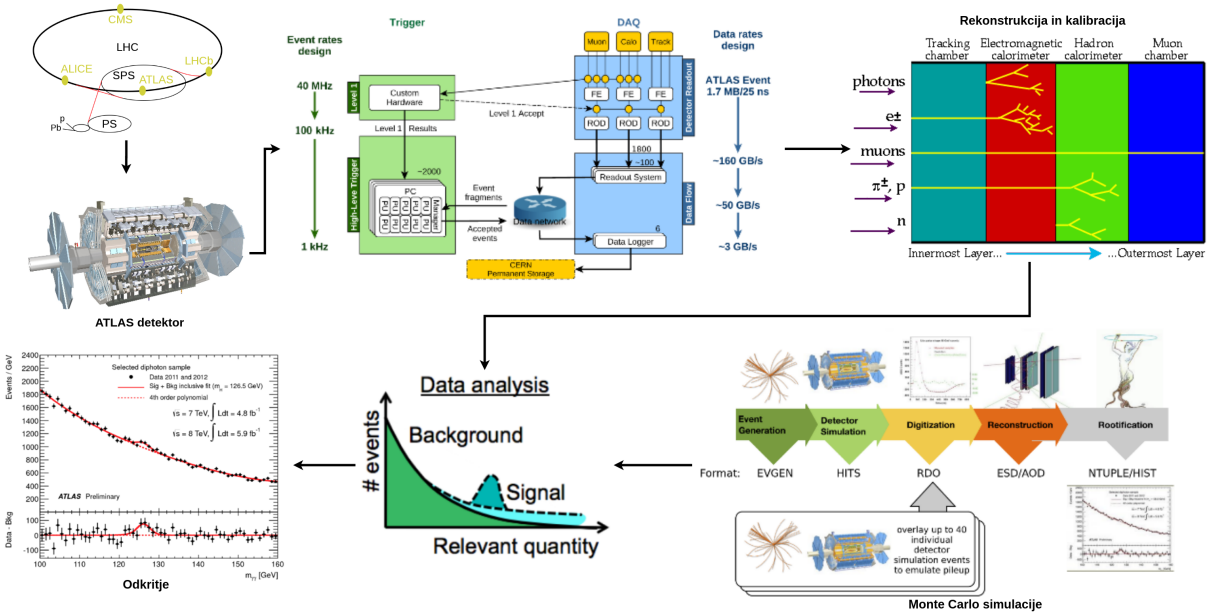
1.2 Rekonstrukcija

V detektorju se zanimivi procesi dogajajo na majhnih časovnih skalah. Higgsov bozon na primer razpade v približno 10^{-22} s. Za opazovanje so tako uporabni razpadni produkti začetnih delcev, iz katerih sklepamo na delec, iz katerega so nastali. Boljše znanje o lastnostih (tip delca, energija, smer, gibalna količina, ...) razpadnih produktov pomeni tudi boljšo rekonstrukcijo začetnega delca. Pri rekonstrukciji kinematičnih količin delcev, ki nastanejo pri trkih, se uporabljajo napredne računske metode, tudi strojno učenje.

1.3 Simulacije

Pomemben korak pri analizi podatkov iz eksperimentov je primerjava teh vrednosti s teoretično napovedanimi preko simulacij. Simulira se fizikalni proces, kot tudi odziv detektorja. Procesi so naključne narave zaradi česar se uporabljajo Monte Carlo simulacije. Take vrste simulacij so računsko izredno zahtevne. Simulacija odziva detektorja pri enem trku protona s protonom lahko traja tudi do nekaj minut.

Potek analize podatkov vse od zajema le-teh v detektorju ATLAS prikazuje slika 1.



Slika 1: Pretok in procesiranje podatkov, povzeto po [22].

2 Fizika na LHC-ju

2.1 Higgsov bozon

Z odkritjem Higgsovega bozona na detektorjih ATLAS in CMS leta 2012 [14, 15] je bil najden še zadnji delec, ki ga predvideva Standardni model [33]. V Standardnem modelu Higgsov mehanizem pojasni, kako bozoni, ki so nosilci šibke interakcije, dobijo maso preko zloma elektrošibke simetrije (*EWSB*) [3, 6]. Brez tega mehanizma bi bili vsi delci, pa tudi bozoni brez mase, kar pa se ne ujema z meritvami, ki dajo za mase bozonov W^- in W^+ vrednost ≈ 80 GeV/c^2 in za maso bozona Z^0 vrednost ≈ 90 GeV/c^2. Problem reši Higgsovo polje, ki povzroči spontan zlom elektrošibke simetrije. Zlom simetrije sproži Higgsov mehanizem, ki povzroči maso bozonov in drugih delcev s katerimi polje interagira. Teorija napove obstoj Higgsovega bozona, ki je kvant tega polja (fluktuacija polja) in je masiven delec. Higgsov bozon je sam svoj antidelec, naboja, ter barvnega naboja, ter ima spin 0 (skalarni bozon). Standardni model ne napove njegove mase, izmerjena vrednost z zgoraj omenjenima detektorjema pa je 125.18 ± 0.16 GeV/c^2.

V zgodnjem vesolju, 10^{-12} s po velikem poku, je imelo Higgsovo polje zaradi visokih temperatur povprečno vrednost 0 povsod (slika 2 levo). V tem času sta bili elektromagnetna sila, katere prenašalci so fotoni in šibka sila združeni v eno samo, elektrošibko. Obstajali so torej štirje brez masni delci elektrošibke sile. Pri nadaljnjem širjenju vesolja je prišlo do razcepitve elektrošibke sile na elektromagnetno in na šibko. Elektrošibka simetrija je bila zlomljena. Bozoni so dobili maso preko Higgsovega polja, foton, ki ne interagira s tem poljem pa je ostal brez mase. Higgsovo polje dobi pod kritično temperaturo T_C ne ničelno povprečno vrednost v celotnem prostoru, tudi v vakuumu. Vrednosti se reče vakuumska pričakovana vrednost (*vacuum expectation value*) in je za Higgsovo polje po celotnem prostoru enaka. To se lahko zgodi zaradi točno določenega potenciala katerega pričakovana vrednost ni več v središču (v ničli povprečne vrednosti polja, slika 2 desno). Za primerjavo, polja drugih elementarnih delcev se stalno spreminjajo zaradi

kvantnih fluktuacij, ampak imajo povprečno vrednost v vakuumu enako nič. Masa delca je odvisna od moči interakcije med tem delcem in Higgsovim poljem.

Maso Higgsovega bozona, ki je fluktuacija Higgsovega polja okoli ravnovesne vrednosti H_{vac}, definira Higgsov potencial $V(H)$. Polje H razstavimo na dva dela: na del h, ki predstavlja delec polja (Higgsov bozon) in niha okrog H_{vac} ter na konstanten del $v/\sqrt{2}$ kot

$$H = \frac{h+v}{\sqrt{2}} \,. \tag{2.1}$$

Potencial Higgsovega bozona je podan z enačbo:

$$V(h) = \frac{1}{4}\lambda h^4 + \lambda v h^3 + \lambda v^2 h^2 \,. \tag{2.2}$$

Zadnji člen ustreza masi delca $m_h^2 = 2\lambda v^2$ druga dva člena pa sklapljanju Higgsovega bozona samim s seboj. Potencial je tako [6, 26]:

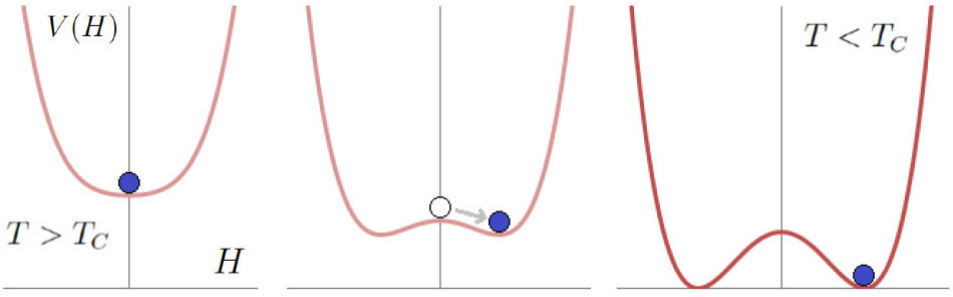
$$V(h) = \frac{1}{2}m_h^2 h^2 + \lambda v h^3 + \frac{1}{4}\lambda h^4 \,, \tag{2.3}$$

kjer je λ sklopitvena konstanta. Preko meritve mase m_h se lahko izračuna tudi vakuumska pričakovana vrednost Higgsovega polja, ki znaša [4]

$$v \approx 246.22 \text{ GeV} \,. \tag{2.4}$$

Izračunamo lahko tudi sklopitveno konstanto, ki je

$$\lambda = \frac{m_h^2}{2v^2} \approx 0.13 \,. \tag{2.5}$$



Slika 2: Zlom simetrije v potencialu Higgsovega polja (fazni prehod). Na začetku vesolja (levo) in danes (desno). Na ordinatni osi je potencial Higgsovega polja. Na abscisni osi je povprečna pričakovana vrednost vakuuma Higgsovega polja. Eksperimenti opazujejo pozicijo ter ukrivljenost tega minimuma. Potencial ima obliko mehiškega klobuka [3].

2.2 Nastanek para Higgsovih bozonov

Eksperimenti na LHC-ju so do zdaj izmerili samo masni člen potenciala 2.3. Členi povezani s sklopitvijo Higgsovega bozona s samim s sabo določajo obliko potenciala in s tem tudi naravo zloma elektrošibke simetrije. Ena od pomembnih nalog detektorjev LHC-ja je natančna meritev sklopitvene konstante preko identifikacije procesov, ki so občutljivi

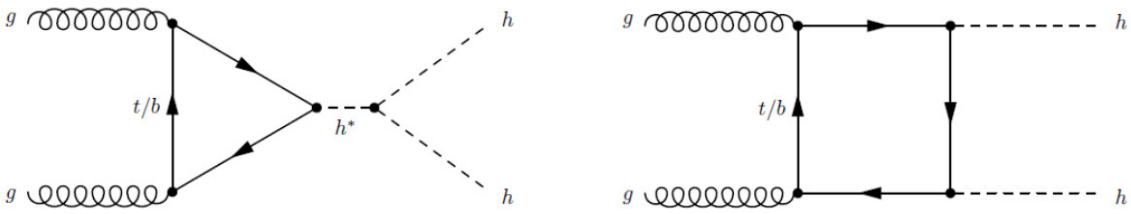
na ta parameter. Kakršnokoli znatno odstopanje meritev od teorije bi vodilo do odkritja fizike onstran Standardnega modela, ki jo pogosto imenujemo "nova fizika".

Drugi člen v enačbi 2.3 opisuje vozlišča s tremi Higgsovimi bozoni, tretji člen pa vozlišča s štirimi Higgsovimi bozoni. Parameter λ v drugem členu se imenuje trilinearni sklopitveni parameter.

Standardni model napoveduje ne resonančno produkcijo parov Higgsovih bozonov pri trkih protonov. Proton je sestavljen iz treh valenčnih kvarkov, množice parov kvark-antikvark ter iz gluonov. Glavni način nastajanja parov je preko procesa fuzije dveh gluonov (*gg-fusion*), ki lahko poteče pri trkih protonov. V Standardnem modelu je možnih več različnih procesov fuzije gluonov, ki ustvarijo pare Higgsovih bozonov preko zank težkih kvarkov. Prvi tak proces opisuje "trikotni diagram"(slika 3a), drugega pa "škatlasti diagram"(slika 3b). Samo pri prvem procesu lahko izmerimo parameter λ preko vozlišča treh Higgsovih bozonov. Drugi proces pa ne predstavlja samo ozadja, pač pa tudi destruktivno interferira s prvim. Nastajanje parov Higgsovih bozonov ima majhen sipalni presek in ga spremlja veliko procesov ozadja, zaradi česar bo potrebno veliko število podatkov za dovolj natančno analizo.

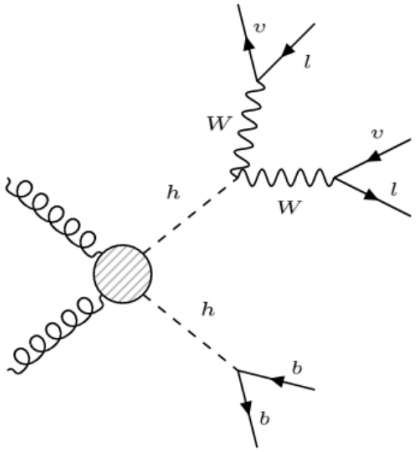
V nadaljevanju opišem analizo [6, 7] iskanja tvorbe parov Higgsovih bozonov pri katerem ta par razpade v $b\bar{b}$ ter W^+W^-, kjer nabiti šibki bozon $W^\pm$ nadalje razpade leptonsko v $l\nu$ (slika 4).

Pred opisom te analize povem še nekaj o strojnem učenju ter nevronskih mrežah, ki jih je analiza uporabila za razločevanje med ozadjem in procesom hh.



(a) (b)

Slika 3: Možna Feynmanova diagrama nastanka hh.



Slika 4: Feynmanov diagram hh razpada v $WWb\bar{b} \to l\nu l\nu b\bar{b}$. Krog predstavlja možne vmesne procese preko fuzije gluonov (slika 3).

3 Strojno učenje

Strojno učenje je veja umetne inteligence, ki se ukvarja z zasnovo in razvojem algoritmov, ki lahko izboljšajo svoje vedenje na podlagi empiričnih podatkov. Podatki so predstavljeni v obliki primerov, ki ponazarjajo povezave med opaženimi spremenljivkami. Glavni cilj strojnega učenja je samodejno učenje prepoznavanja vzorcev in sprejemanje inteligentnih odločitev. S strojnim učenjem se lotimo nalog, ki so pretežke, da bi jih rešili s fiksnim programiranjem. Primer takšnega problema je prepoznavanje slik in govora. Bolj natančna definicija algoritemskega učenja:

Računalniški program se uči iz izkušenj E glede na nalogo T in na neko mero uspešnosti P, če se uspešnost za T, ki jo meri P poveča z izkušnjami E.

Algoritme strojnega učenja lahko razdelimo v dve širši skupini. Razdeljeni sta glede na to, koliko nadzora dobi algoritem med samim postopkom učenja. Pri nadzorovanem učenju se algoritem uči na primerih. Vhodni podatki vključujejo tudi oznake, ki povejo pravilen odgovor. Od algoritma na koncu pričakujemo, da bo pravilno označil nove podatke, ki jih nismo uporabili pri učenju. Najbolj tipična primera sta klasifikacija (diskretna kategorizacija podatkov) in regresija (modeliranje, prilagajanje).

Pri nenadzorovanem učenju algoritmu ne podamo oznak in od njega pričakujemo, da sam najde podobnosti in povezave med vhodnimi podatki. Primer je grupiranje (postopek razvrščanja predmetov v razrede in podrazrede).

V tem seminarju sem se omejil na nadzorovano učenje, ter kot primer le-tega vzel nevronske mreže.

3.1 Nadzorovano učenje

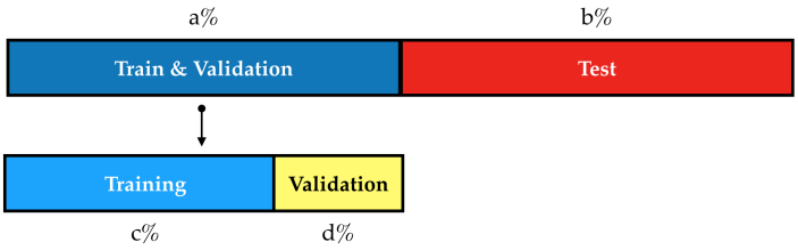
Problem nadzorovanega učenja je sestavljen iz:

- množice podatkov $\mathcal{D}_{data} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ sestavljene iz meritev $\boldsymbol{x}_i$ in ciljnih vrednosti $\boldsymbol{y}_i$ vzorčenih iz verjetnostne porazdelitve $p_{\text{data}}(y|\boldsymbol{x};\boldsymbol{\theta})$, kjer je $\{\theta_i\}$ množica parametrov modela.
- funkcije izgube $L(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{\theta})$, ki omogoča presojo o tem, kako dobro deluje model na vektorju $\boldsymbol{y}$.
- optimizacijskega algoritma.
- učnega algoritma, ki vrne funkcijo $h : \boldsymbol{x} \to \boldsymbol{y}$. Funkcija $h(\boldsymbol{x};\boldsymbol{\theta})$ je napoved modela, ki minimizira funkcijo izgube.

3.2 Množica podatkov

Preden začnemo s strojnim učenjem vedno naključno razdelimo $\mathcal{D}_{data}$ na učno množico $\mathcal{D}_{train}$ in testno množico $\mathcal{D}_{test}$ (npr. v razmerju 0.7:0.3). Pri tem predpostavimo, da so primeri v obeh množicah neodvisni in porazdeljeni po enaki porazdelitvi p_{data}. $\mathcal{D}_{train}$ uporabljamo pri učenju, $\mathcal{D}_{test}$ pa pri analizi rezultatov. Množico $\mathcal{D}_{train}$ uporabimo pri minimizaciji funkcije izgube na modelu z izračunom $\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{Y}_{train}, \boldsymbol{X}_{train}; \boldsymbol{\theta})$. Model ovrednotimo preko napake $L(\boldsymbol{Y}_{test}, \boldsymbol{X}_{test}; \hat{\boldsymbol{\theta}})$. V vrsticah matrike $\boldsymbol{X}$ so meritve $\boldsymbol{x}$, stolpci pa predstavljajo različne količine. V matriki $\boldsymbol{Y}$ so ciljne vrednosti, v primeru skalarne ciljne vrednosti je to vektor. Veljati mora, da je napaka izračunana na testni

množici (*out-of-sample error*) večja od napake na učni množici (*in-sample error*), torej $E_{out} \geq E_{in}$. Zaradi kompleksnosti sistemov ponavadi ne poznamo točne funkcijske zveze, ki nam da najboljše napovedi, zaradi česar uporabimo več različnih modelov, ki jih nato primerjamo med seboj. Pri takem pristopu na začetku razdelimo podatke v tri množice in sicer na učno, testno in še validacijsko množico (npr. v razmerju 0.5:0.25:0.25). Testno množico uporabimo za primerjavo med modeli, validacijsko pa na koncu za oceno izbranega modela. Takim vrstam postopkov pravimo *cross-validation*.



Slika 5: Razdelitev podatkov, ki so na voljo za strojno učenje.

3.3 Funkcija cene

Naj bo $p_{\text{model}}(y|\boldsymbol{x};\boldsymbol{\theta})$ verjetnostna porazdelitev, ki ocenjuje pravo a neznano porazdelitev $p_{\text{data}}(y|\boldsymbol{x};\boldsymbol{\theta})$. Cilj je oceniti pogojno verjetnost $p_{\text{data}}(y|\boldsymbol{x};\boldsymbol{\theta})$, ki napove y pri danem $\boldsymbol{x}$. Tak problem tvori osnovo večine algoritmov nadzorovanega strojnega učenja. Po metodi maksimalne zanesljivosti optimalno cenilko za parameter $\boldsymbol{\theta}$ izračunamo kot [1]

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{N} p_{\text{model}}(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{\text{model}}(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}) \,. \tag{3.1}$$

Produktu verjetnostih gostot

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N} p_{\text{model}}(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}) \tag{3.2}$$

pravimo funkcija zanesljivosti. Definiramo še njen logaritem (*log-likelihood*), ki spremeni produkt v vrsto in ne spremeni ekstrema funkcije, ker je logaritem monotona funkcija. Zgornji zapis je ekvivalenten

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{N} \log p_{\text{model}}(y_i|\boldsymbol{x}_i;\boldsymbol{\theta}) = 0 \tag{3.3}$$

za izračun parametrov $\hat{\theta}_j$.

V strojnem učenju se definira funkcija cene (*cost function*), ki je

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{\text{data}}} L(y, h(\boldsymbol{x};\boldsymbol{\theta})) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, h(\boldsymbol{x}_i;\boldsymbol{\theta})) \,, \tag{3.4}$$

kjer je L definirana za vsak primer, $\hat{p}_{\text{data}}$ empirična porazdelitev definirana na $\mathcal{D}_{train}$ iz katere so vzorčeni podatki $(\boldsymbol{x}_i, y_i)$, h napovedana vrednost pri vhodnem podatku $\boldsymbol{x}$,

[1] Vsi log so mišljeni z osnovo e.

y ciljna vrednost ter $\mathbb{E}$ pričakovana vrednost. Količini se reče tudi empirično tveganje [2, 19]. Maksimizacija v 3.1 je ekvivalentna maksimizaciji količine 3.4. Cilj metod strojnega učenja je iskanje različnih parametrov, uteži ter struktur, ki minimizirajo tako definirano funkcijo.

Želeli bi pričakovano vrednost funkcije izgube definirano na porazdelitvi p_{data},

$$J^*(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}} L(y, h(\boldsymbol{x};\boldsymbol{\theta}))\ , \tag{3.5}$$

namesto pričakovane vrednosti na končni učni množici. Tako definirani funkciji se reče tveganje. Problem je v tem, da ne poznamo porazdelitve p_{data} in ne moremo izračunati 3.5, zaradi česar se uporablja približek s povprečjem 3.4.

Pogosto se za funkcijo izgube uporablja križna entropija (*cross-entropy*). Za poljubni verjetnostni porazdelitvi $p(x)$ in $q(x)$ jo napišemo kot

$$H(p,q) = -\mathbb{E}_{x\sim p}[\log q] = -\sum_i p(x_i)\log q(x_i)\ , \tag{3.6}$$

kjer vsota teče po vseh podatkih $\{x_i\}$. V teoriji strojnega učenja je definirana med porazdelitvama $\hat{p}_{\text{data}}$ in p_{model}. Maksimizacija funkcije zanesljivosti je ekvivalentna minimizaciji križne entropije [19].

3.4 Optimizacijski algoritmi

Poiskati moramo vrednosti $\boldsymbol{\theta}$, ki minimizirajo funkcijo cene. Razredu metod, ki minimizirajo $J(\boldsymbol{\theta})$ pravimo *gradient descent (GD)*. Ideja je podobna kot pri Newtonovi metodi 3.7. Iterativno spreminjamo parametre $\boldsymbol{\theta}$ v smeri v kateri je gradient funkcije cene največji in negativen. Algoritem zaključimo, ko najdemo lokalni minimum. Težave se pojavijo, ker imamo zapletene funkcije v večih dimenzijah, ki niso konveksne in imajo veliko lokalnih minimumov. Poleg tega pa sploh ne poznamo prave funkcije, ampak samo njeno aproksimacijo, ki smo jo dobili iz podatkov.

Pri osnovni verziji algoritma GD začnemo z $\boldsymbol{\theta}_0$ in iterativno posodabljamo:

$$\begin{aligned} \boldsymbol{v_i} &= \eta\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_i), \\ \boldsymbol{\theta}_{i+1} &= \boldsymbol{\theta}_i - \boldsymbol{v}_i, \end{aligned} \tag{GD}$$

kjer je $\boldsymbol{v}$ korak in η velikost tega koraka (*learning rate*). Konvergenca algoritma je odvisna od izbire velikosti η (slika 6).

Metodo primerjamo z Newton–Raphsonovo, ki je metoda drugega reda in se glasi:

$$\begin{aligned} \boldsymbol{v_i} &= H^{-1}(\theta_i)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_i), \\ \boldsymbol{\theta}_{i+1} &= \boldsymbol{\theta}_i - \boldsymbol{v}_i, \end{aligned} \tag{3.7}$$

kjer je $H(\theta)$ Hessova matrika drugih odvodov. Opazimo, da ta metoda avtomatsko posodablja korak preko matrike H. Na žalost pa moramo pri tej metodi poznati tako matriko drugih odvodov, kot njen inverz, zaradi česar je računsko potratna.

GD zahteva izračun

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} L(y_i, \boldsymbol{x}_i; \boldsymbol{\theta})\ . \tag{3.8}$$

Cena take operacije je $O(N)$, ker računamo gradient na vsakem primeru iz množice podatkov. Za velike N (veliko število podatkov) traja izračun enega koraka predolgo. Algoritem pohitrimo tako, da podatke iz $\mathcal{D}_{\text{data}}$ razdelimo v skupine (*mini-batches*) velikosti M

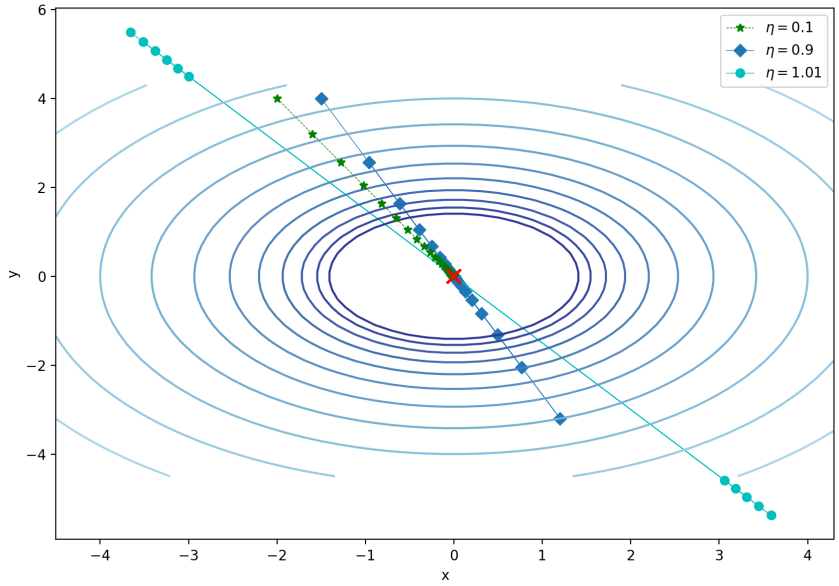
$$B = \{(\boldsymbol{X}^{\{1\}}, y^{\{1\}}), (\boldsymbol{X}^{\{2\}}, y^{\{2\}}), ..., (\boldsymbol{X}^{\{K\}}, y^{\{K\}})\}\,, \tag{MB}$$

ki jih je vseh skupaj $K = N/M$. Število elementov M v eni skupini B_k je ponavadi majhno v primerjavi z N, ter se ne spreminja z dodajanjem novih primerov. Približek za gradient je tako

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} L(y_i, \boldsymbol{x}_i; \boldsymbol{\theta})\,, \tag{3.9}$$

kjer v vsakem koraku porabimo eno skupino B_k. Iteraciji čez vse podatke N, torej čez vse skupine B_k, pravimo epoha (*epoch*) ali obdobje.

GD se ponavadi ne uporablja zaradi počasne konvergence in drugih težav. Uporabljajo pa se njegove nadgradnje kot so Adagrad, RMSprop, Adam,... [29].



Slika 6: Konvergenca algoritma GD glede na η v kvadratičnem potencialu. Pri $\eta = 0.1$ konvergira enakomerno, pri $\eta = 0.9$ oscilira ampak še vedno konvergira, pri $\eta = 1.01$ pa divergira (trajektorija gre stran od minimuma).

4 Nevronske mreže

Nevronske mreže so vrsta nadzorovanega strojnega učenja. Najbolj pogosto se uporabljajo pri nalogah klasifikacije. Velikokrat se jih obravnava kot "*black box*"katerega delovanje je skrito pred uporabnikom. Ta del seminarja je namenjen osvetlitvi delovanja globokih usmerjenih nevronskih mrež, ki predstavljajo eno od najbolj popularnih orodij statističnega odločanja. Pomembne so tudi za nadaljnjo obravnavo bolj kompleksnih algoritmov, kot so rekurzivne in konvolucijske nevronske mreže, ki predstavljajo njihovo nadgradnjo.

Nedavni uspeh strojnega učenja izvira predvsem iz uporabe velikih količin podatkov za učenje globokih nevronskih mrež, ki so sestavljene iz velike količine nelinearnih plasti (*deep learning*). Globlje mreže lahko predstavijo kompleksne funkcije bolj učinkovito kot plitve (*shallow*) mreže, vendar je njihovo učenje težje. Napredki v strojni opremi in izboljšave algoritmov v zadnjih letih so omogočili hitrejše in bolj učinkovito učenje velikih mrež, zaradi česar je njihova uporaba na vseh področjih močno poskočila.

V eksperimentalni fiziki ponavadi ne manjka podatkov, ki bi bili uporabni za učenje nevronskih mrež. Primer so že omenjene Monte Carlo simulacije, ki lahko generirajo poljubno število primerov s pravilnimi oznakami, na katerih se lahko učijo nevronske mreže.

4.1 Globoke usmerjene nevronske mreže

Cilj algoritma (*deep feedforward neural network*) je aproksimacija neznane funkcije f^*. Primer take funkcije je klasifikacijska funkcija $y = f^*(\boldsymbol{x})$, ki pripiše vhodnemu podatku $\boldsymbol{x}$ določeno kategorijo y. Usmerjena nevronska mreža definira preslikavo $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ in se nauči vrednosti parametrov $\boldsymbol{\theta}$, ki kar najbolje opišejo f^*. Usmerjene nevronske mreže, torej take brez povratnih povezav, predstavimo z usmerjenim acilkičnim (brez zank) grafom, ki pove kako so funkcije povezane med seboj. Funkcija f je sestavljena iz večih med seboj povezanih funkcij, na primer $f(\boldsymbol{x}) = f^{(1)}(f^{(2)}(f^{(3)}(\boldsymbol{x})))$. V tem primeru so $f^{(1)}$ prva plast, $f^{(2)}$ skrita plast, ter $f^{(3)}$ izhodna plast. Skrite plasti omogočajo učenje nelinearnih povezav. Dolžina te verige funkcij se nanaša na globino modela, iz česar tudi pride ime globoke nevronske mreže (slika 7).

Nevronsko mrežo si lahko predstavljamo kot transformacijo, ki za vhodni podatek vzame vektor neke dimenzije, ter vrne skalar ali pa transformiran vektor ne nujno iste dimenzije.

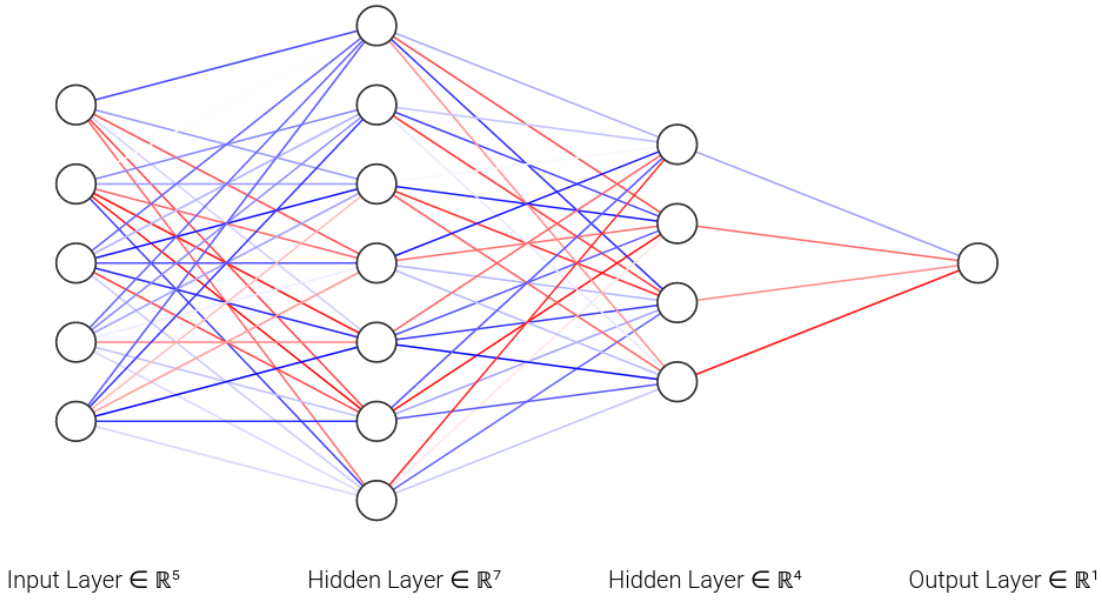
4.2 Nevron

Osnovna enota nevronske mreže je nevron, ki sprejme poljuben vektor lastnosti $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ in vrne skalar $a(\boldsymbol{x})$. Mreža sestoji iz množice takih nevronov, ki so zloženi po plasteh (slika 7). Funkcija a je sestavljena iz linearne operacije, ki uteži vhodne vrednosti ter iz nelinearne transformacije σ, ki je enaka za vse nevrone in ji rečemo aktivacijska funkcija. Linearna funkcija ima obliko skalarnega produkta

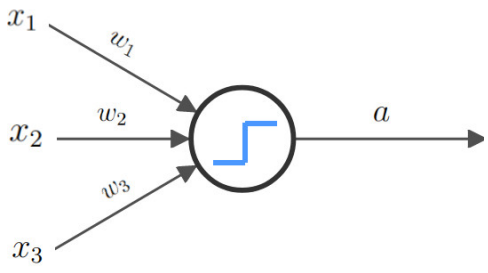
$$z = \boldsymbol{w} \cdot \boldsymbol{x} + b\,, \tag{4.1}$$

kjer so $\boldsymbol{w} = (w_1, w_2, ..., w_d)$ uteži ter b pristranskost (*bias*) nevrona. Tako je izhodna funkcija posameznega nevrona $a = \sigma(z)$ (slika 8).

Različne izbire aktivacijskih funkcij vodijo do različnih računskih sposobnosti nevronov. V preteklosti se je za aktivacijsko funkcijo največ uporabljalo Heavisideovo funkcijo,



Slika 7: Primer nevronske mreže z dvema skritima plastema. Barve predstavljajo velikost posamezne uteži.



Slika 8: Perceptron.

hiperbolični tangens ter logistično/sigmoidno (v fiziki poznana kot Fermijeva) funkcijo (slika 9 zgornja vrstica). Nevronom, ki uporabljajo stopnico kot aktivacijsko funkcijo pravimo perceptroni in so bili razviti že v 60 letih. Izhod perceptrona je

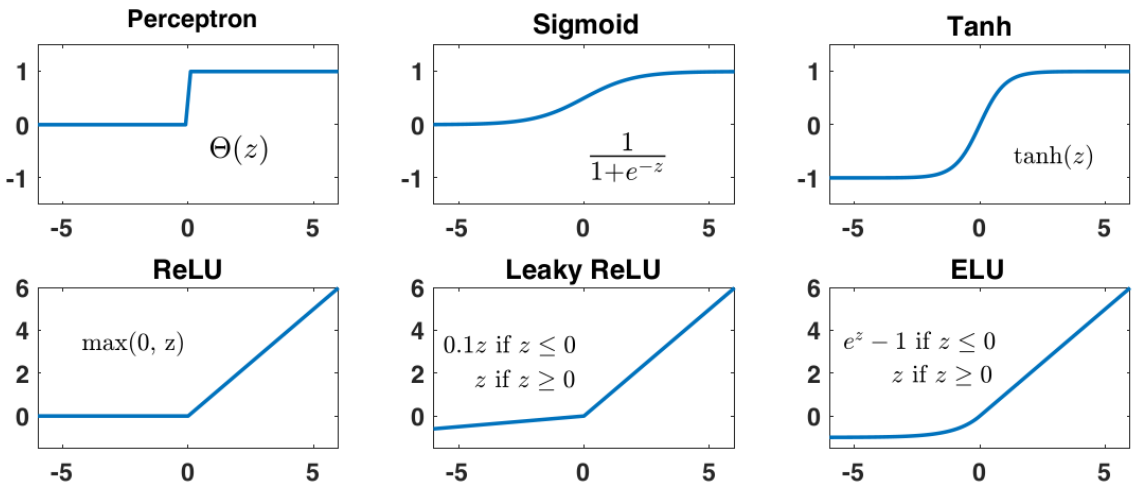
$$a = \begin{cases} 0 & \text{če} \quad \boldsymbol{w} \cdot \boldsymbol{x} \leq \text{aktivacijski prag} \\ 1 & \text{če} \quad \boldsymbol{w} \cdot \boldsymbol{x} > \text{aktivacijski prag} \end{cases}$$

pri čemer definiramo bias kot $b \equiv -$aktivacijski prag. Večji bias pomeni, da se bo perceptron prej aktiviral (1 na izhodu). Tako definiran perceptron se obnaša enako kot logična vrata NAND. Problem mreže perceptronov je, da lahko majhna sprememba uteži ali biasa povzroči velike spremembe (izhod gre takoj iz 0 na 1). Problem rešimo z uporabo funkcije, ki na izhodu nevrona da zvezno vrednost $a \in [0, 1]$ ali $a \in [-1, 1]$. Primer take funkcije je logistična/sigmoidna

$$\sigma(z) = \frac{1}{1 + e^{-z}} \,. \tag{4.2}$$

Danes se uporabljajo večinoma funkcije tipa ReLU (*rectified linear units*). Razlog je v tem, da pri učenju nevronske mreže uporabljamo metode GD, pri katerih je potreben

izračun odvoda teh funkcij. Pri velikih vrednostih uteži in biasa pride do nasičenja odvoda, $\partial\sigma/\partial z \to 0$ pri $z \gg 1$, za funkciji kot sta sigmoidna in tanh (*vanishing gradients*). To dejstvo otežuje učenje nevronske mreže zaradi česar uporabljamo funkcije, ki se ne nasičijo pri velikih vhodnih vrednostih.



Slika 9: Primeri aktivacijskih funkcij. V drugi vrstici so funkcije pri katerih ne pride do nasičenja [5].

4.3 Usmerjena propagacija

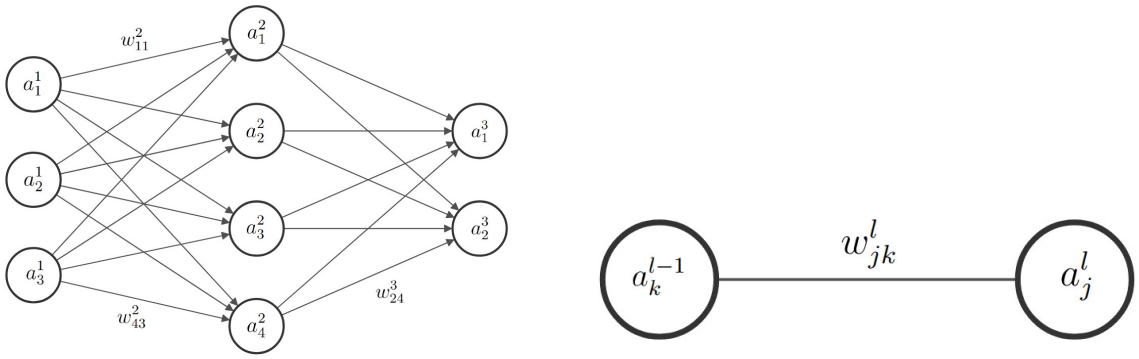
Naj bo w^l_{jk} utež povezav med k-tim nevronom v plasti $l-1$ in j-tim nevronom v plasti l (slika 10b). Označimo še a^l_j aktivacijo j-tega nevrona v plasti l ter enako še njegov bias b^l_j. Aktivacija j-tega nevrona v l-ti plasti je tako povezana z aktivacijami v $l-1$ plasti kot

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) = \sigma(z^l_j)\,, \tag{4.3}$$

kjer vsota teče po vseh nevronih v plasti $l-1$. Enačbo napišemo vektorsko tako, da definiramo matriko uteži W^l za plast l z elementi w^l_{jk}, vektor aktivacij v l-ti plasti $\boldsymbol{a}^l$ in še $\boldsymbol{b}^l$. Enačba v vektorski obliki je tako

$$\boldsymbol{a}^l = \sigma(W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l) \ = \sigma(\boldsymbol{z}^l), \tag{F}$$

kjer funkcija σ deluje na vsak element vektorja posebej. Definirali smo še $\boldsymbol{z}^l$, ki je vektor uteženih vhodnih vrednosti v plasti l. Enačba F opiše povezavo med aktivacijami v plasti l in $l-1$, pravimo ji usmerjena propagacija (*feedforward*), ker računa vrednosti aktivacij naprej po nevronski mreži.



(a) Povezave med nevroni.

(b) Dva sosednja nevrona

Slika 10: Oznake nevronske mreže.

4.4 Povratna propagacija

Ideja učenja nevronske mreže je enaka kot pri drugih algoritmih nadzorovanega strojnega učenja. Cilj je izračunati najboljše uteži in biase, ki imajo najmanjšo napako pri posplošitvi na še ne videne primere. Parametre dobimo preko minimizacije funkcije cene s pomočjo GD. Razlika je le v tem, da imamo sedaj več plasti, zaradi katerih je računanje gradienta bolj zapleteno. Zanima nas torej kako učinkovito izračunati

$$\frac{\partial J}{\partial w^l_{jk}} \quad \text{in} \quad \frac{\partial J}{\partial b^l_j} \tag{4.4}$$

za celotno mrežo. Izraz podaja hitrost spreminjanja funkcije cene, ko spreminjamo uteži (ali biase). Za funkcijo izgube lahko uporabimo [2]

$$J(a^L) = \frac{1}{2}\sum_j (y_j - a^L_j)^2 \,, \tag{MSE}$$

kjer j teče po nevronih končne plasti. Funkcija je indirektno odvisna od vseh plasti preko enačbe F. MSE je nenegativna ter ima vrednost blizu 0, če $y \approx a$, kar je natanko to kar želimo od funkcije izgube.

Vpeljemo vmesno količino [3]

$$\Delta^l_j = \frac{\partial J}{\partial z^l_j} \tag{4.5}$$

preko katere bomo izračunali 4.4. Predstavlja napako j-tega nevrona v plasti l. Povratna propagacija sestavljajo 4 osnovne enačbe preko katerih izračunamo 4.5 ter gradient funkcije izgube 3.4. Enačbe izpeljemo s pomočjo verižnega pravila za funkcije večih spremenljivk.

Enačbo za napako v plasti L dobimo z uporabo verižnega pravila na 4.5 glede na izhodne aktivacije kot

$$\Delta^L_j = \sum_k \frac{\partial J}{\partial a^L_k}\frac{\partial a^L_k}{\partial z^L_j} = \sum_k \frac{\partial J}{\partial a^L_k}\frac{\partial \sigma(z^L_k)}{\sigma z^L_j} = \frac{\partial J}{\partial a^L_j}\sigma'(z^L_j) \,,$$

[2] Ponavadi se ne dela razlike med funkcijo izgube in cene. Vsaki minimizacijski funkciji se reče *cost/loss/error/objective/energy function.*

[3] Ekvivalenten algoritem dobimo če vzamemo $\partial J/\partial a^l_j$.

kjer upoštevamo, da je aktivacija a_k^L ne ničelna le za $k = j$. Enačbo napišemo vektorsko

$$\boldsymbol{\Delta}^L = \nabla_a J \odot \sigma'(\boldsymbol{z}^L)\,, \tag{B1}$$

kjer je $\nabla_a J$ vektor z elementi $\partial J/\partial a_j^L$. Produkt $\odot$ je mišljen po komponentah (*element-wise*).
Potrebujemo propagacijo napake med dvema sosednjima plastema l in $l+1$. Želimo napisati Δ_j^l z Δ_k^{l+1}. Ponovno uporabimo verižno pravilo

$$\Delta_j^l = \frac{\partial J}{\partial z_j^l} = \sum_k \frac{\partial J}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \Delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}\,. \tag{4.6}$$

Iz F vidimo

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}\,,$$

odvajamo po z_j^l ter dobimo

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l)\,.$$

Vstavimo v enačbo 4.6 in tako

$$\Delta_j^l = \sum_k w_{kj}^{l+1} \Delta_k^{l+1} \sigma'(z_j^l)\,,$$

kar lahko napišemo vektorsko kot

$$\boldsymbol{\Delta}^l = ((W^{l+1})^T \boldsymbol{\Delta}^{l+1}) \odot \sigma'(z^l)\,. \tag{B2}$$

Napako glede na bias izračunamo iz

$$\Delta_j^l = \frac{\partial J}{\partial z_j^l} = \frac{\partial J}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l}\,.$$

Upoštevamo $\frac{\partial b_j^l}{\partial z_j^l} = 1$ po 4.3 in dobimo

$$\Delta_j^l = \frac{\partial J}{\partial b_j^l}\,. \tag{B3}$$

Potrebujemo še enačbo za spremembo funkcije izgube glede na uteži. Odvajamo funkcijo izgube po utežeh

$$\frac{\partial J}{\partial w_{jk}^l} = \frac{\partial J}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l}.$$

Odvajamo še 4.3 po w_{jk}^l

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = \sum_k a_k^{l-1}$$

upoštevamo 4.5 ter izpeljemo zadnjo enačbo

$$\frac{\partial J}{\partial w_{jk}^l} = \Delta_j^l a_k^{l-1}\,. \tag{B4}$$

Enačbe B1, B2, B3 in B4 podajajo način izračuna gradienta funkcije izgube na učinkovit način. Celoten algoritem za učenje nevronske mreže z uporabo MB, usmerjene propagacije, povratne propagacije in GD je tako (algoritem 1):

Algoritem 1: Učenje nevronske mreže

Data: Podatki $\boldsymbol{X}$ in oznake $\boldsymbol{y}$ razdeljeni v skupine.
Result: Optimalne uteži W^l in biase b^l nevronske mreže.
Naključno inicializiraj vse uteži W^l in biase b^l po vseh plasteh;
for *epoch* **do**
  for *mini-batch* B_k **do**
    for *primer* $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ *iz* B_k **do**
      1. Nastavi prvo aktivacijo a^1.
      2. Izvedi usmerjeno propagacijo F po plasteh $l = 2, 3, ..., L$.
      3. Izračunaj napako v zadnji plasti $\boldsymbol{\Delta}^L$ po B1.
      4. Propagiraj napake $\boldsymbol{\Delta}^l$ po plasteh $l = L, L-1, ..., 2$ z B2.
    end
    Uporabi optimizacijski algoritem npr. GD ter za vse $l = L, L-1, ..., 2$ posodobi uteži in biase z uporabo B3 in B4 kot $W^l \to W^l - \frac{\eta}{M} \sum_{i=1}^{M} \boldsymbol{\Delta}^{i,l} (\boldsymbol{a}^{i,l-1})^T$ in $b^l \to b^l - \frac{\eta}{M} \sum_{i=1}^{M} \boldsymbol{\Delta}^{i,l}$.
  end
end

Odvode funkcije cene bi lahko enostavno računali tudi z dvostransko diferenco

$$\frac{\partial J}{\partial w_j} \approx \frac{J(\boldsymbol{w} + \epsilon \boldsymbol{e}_j) - J(\boldsymbol{w} - \epsilon \boldsymbol{e}_j)}{2\epsilon} , \tag{4.7}$$

kjer so vse uteži zapisane kot $\boldsymbol{w} = (w_1, w_2, ...)$, $\boldsymbol{e}_j$ je enotski vektor v smeri j ter $\epsilon > 0$. Tak pristop računanja odvoda je sicer enostaven, ampak časovno zahteven. Formula zahteva izračun $J(\boldsymbol{w} \pm \epsilon \boldsymbol{e}_j)$ za vsako utež w_j. Ta postopek je sicer počasen, vendar pa je uporaben za preverjanje pravilnosti izračunanih gradientov s povratno propagacijo.

Kljub hitremu načinu izračuna gradienta funkcije izgube pa je učenje globokih nevronskih mrež z veliko plastmi in parametri še vedno počasno. Napredki v tehnologiji, predvsem masovna uporaba GPU-jev, so v zadnjem desetletju povzročili velik porast uporabe nevronskih mrež na problemih z velikim številom podatkov (*big-data*), tako v znanosti, kot v komercialnih aplikacijah.

4.5 Klasifikacija z uporabo nevronskih mrež

Nevronske mreže lahko uporabimo za regresijo, ciljna vrednost $y \in \mathbb{R}$. V zadnjo plast postavimo linearne nevrone brez aktivacijske funkcije, torej $a_j^L = z_j^L \in \mathbb{R}$ namesto sigmoidnih iz 4.2. V takem primeru lahko za funkcijo izgube uporabimo MSE. Izračunajmo B4 za zadnjo plast ter za funkcijo izgube J vzemimo MSE

$$\frac{\partial J}{\partial w_{jk}^L} = \frac{\partial J}{\partial z_j^L} a_k^{L-1} = a_k^{L-1} (a_j^L - y_j) \sigma'(z_j^L) .$$

Člen $\sigma'(z_j^L)$ povzroči upočasnjevanje učenja, ko se izhodni nevron nasiči. V primeru linearnih nevronov je zadnji člen enak 1 zaradi česar postane MSE primerna funkcija izgube. V tem primeru za napako B1 v zadnji plasti dobimo

$$\boldsymbol{\Delta}^L = \boldsymbol{a}^L - \boldsymbol{y} . \tag{4.8}$$

Ponavadi pa imamo na voljo več razredov med katerimi mora izbrati nevronska mreža. Klasičen primer je klasifikacija na roke napisanih številk (slika 11a). Za ta primer bi imela prva plast 28x28 nevronov (velikost slike) ter izhodna plast 10 nevronov (številke 0 do 9). Pravilno številko (ciljno vrednost) opiše vektor dolžine 10 samih ničel in ene enice (*one-hot vector*) na mestu, ki označuje pravilen razred. Izkaže se, da uporaba MSE ni primerna za takšne naloge, saj je učenje prepočasno. Kvadratično izgubo zamenjamo s križno entropijo [4], ki se zapiše za nevronsko mrežo kot

$$J(a^L) = -\sum_j y_j \log a_j^L + (1 - y_j) \log(1 - a_j^L) \, . \tag{4.9}$$

Izpeljemo jo lahko kot negativen logaritem Bernoullijeve porazdelitve [11, 17] ali pa kot križno entropijo med empirično porazdelitvijo $\hat{p}_{\text{data}} = \{\hat{y}, \hat{y}-1\}$ in modelsko porazdelitvijo $p_{\text{model}} = \{y, y-1\}$ po enačbi 3.6 [1]. Enačbo 4.9 interpretiramo kot vsoto križnih entropij vsakega nevrona v končni plasti, kjer je aktivacija vsakega nevrona dvodelna diskretna verjetnostna porazdelitev, torej a ali $1 - a$ [28]. Kot vidimo ima tako definirana funkcija podobne lastnosti kot MSE. Napaka v zadnji plasti je po B1

$$\Delta_j^L = \frac{\sigma(z_j^L) - y_j}{\sigma(z_j^L)(1 - \sigma(z_j^L))} \sigma'(z_j^L) \, .$$

Če vzamemo sigmoidne nevrone z aktivacijsko funkcijo 4.2 dobimo za napako 4.8 iz česar sledi, da je križna entropija primerna funkcija za klasifikacijske naloge.

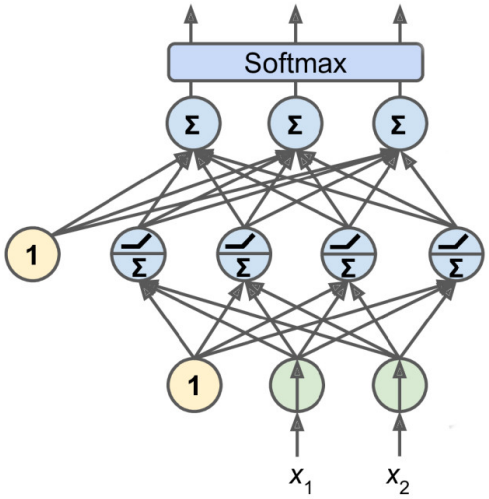
Problema klasifikacije se lahko lotimo tudi malo drugače. In sicer tako, da konstruiramo plast *softmax* (slika 11b). Namesto sigmoidnih funkcij v zadnji plasti uporabimo funkcijo softmax na z_j^L. Aktivacije z uporabo te funkcije so

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \, , \tag{4.10}$$

kjer v imenovalcu seštevamo po vseh nevronih v izhodni plasti. Funkcija softmax ima lepo lastnost, da je $\sum_j a_j^L = 1$, zaradi tega se ji reče tudi normalizirani eksponent. Izhodi nevronske mreže so tako na intervalu [0, 1] in se seštejejo v 1. Izhod iz plasti softmax torej predstavlja verjetnostno porazdelitev. Aktivacijo a_j^L interpretiramo kot verjetnost, da je pravilen razred j.

[4] Vsaka funkcija izgube sestavljena iz $-\log$ je križna entropija med empirično porazdelitvijo in modelsko porazdelitvijo [19]. Ponavadi je s *cross-entropy* mišljena funkcija 4.9.

(a) Številke iz MNIST podatkovne baze. Posamezna slika številke je velika 28x28 slikovnih točk.



(b) Softmax plast na izhodu.

Slika 11: Primer klasifikacije v več razredov 11a in shematski prikaz izhodne softmax plasti 11b [18].

5 Iskanje procesa hh z uporabo nevronskih mrež

Analiza [6, 7], ki sem jo vzel za zgled, se je osredotočila na iskanje ne-resonančnih parov Higgsovih bozonov za primer ko prvi razpade kot $h \to b\bar{b}$ ter drugi preko $h \to WW^*$ [5], kjer nabiti šibki bozoni nadalje razpadejo leptonsko (slika 4). Pri analizi je bilo uporabljenih 139 fb^{-1} [6] podatkov trkov pp pri $\sqrt{s} = 13$ TeV zbranih na detektorju ATLAS med junijem 2015 in oktobrom 2018. Procesi ozadja iz Standardnega modela so bili ocenjeni preko Monte Carlo simulacij. Na enak način je bil simuliran tudi signal, ki ga bi pričakovali po napovedih Standardnega modela pri tem procesu.

Analiza je tipa MVA (*multivariate analysis*). Takšna analiza združi informacije iz več spremenljivk dogodka (*feature vector* $\boldsymbol{x}$) v eno spremenljivko ali v vektor nižje dimenzije. Spremenljivko MVA ali vektor se potem uporabi za napoved verjetnosti ali je dogodek signal ali pa pripada ozadju. Pri analizi so za algoritem MVA uporabili nevronsko mrežo, ki klasificira dogodke v različne razrede, ki predstavljajo verjetnosti za signal ali ozadje. Cilj analize je bil določiti zgornjo mejo sipalnega preseka za $pp \to hh$ proces ter postaviti meje intervala za sklopitveno konstanto λ .

5.1 Arhitektura in učenje nevronske mreže

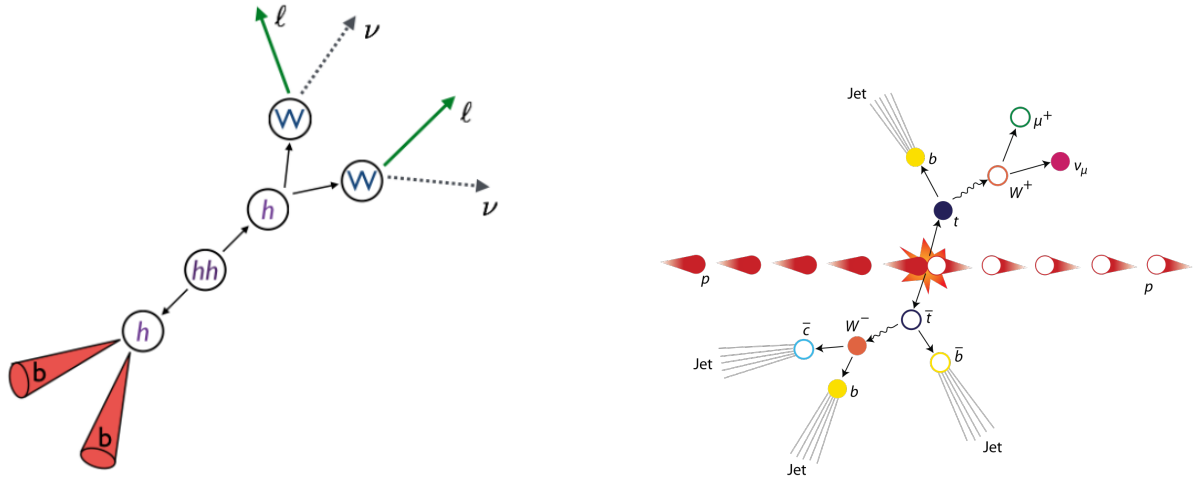
Pri analizi so uporabili več-izhodni klasifikacijski algoritem, torej takšnega, ki ne vrne samo vrednosti za verjetnosti pripada/ne pripada procesu, ampak vrne tudi razred procesa. Primer takega algoritma so že opisane nevronske mreže. Uporabili so klasifikacijo v štiri razrede (oznake):

1. Ne-resonančna tvorba hh
2. Tvorba kvarkov top: $t\bar{t}$ in Wt
3. Drell-Yan proces z lahkimi leptoni: $Z/\gamma^* \to ll$
4. Drell-Yan proces z leptoni tau: $Z/\gamma^* \to \tau\tau$

Prvi razred predstavlja signal (slika 12a), drugi trije pa predstavljajo več kot 80% vsega ozadja v območju analize (slika 12b). Za vhodne podatke nevronske mreže so uporabili 35 količin, ki vključujejo nekatere enostavne, pa tudi kompleksnejše količine. Primer enostavne količine je transverzalna gibalna količina (gibalna količina v ravnini pravokotni na žarek protonov), primer kompleksne pa je porazdelitev invariantne mase izračunane iz enostavnih količin. Nevronsko mrežo si lahko predstavljamo kot način kako pridemo iz višje dimenzije, sestavljene iz enostavnih količin, v nižjo dimenzijo bolj kompleksnih spremenljivk. Za učenje nevronske mreže so bili uporabljeni simulirani dogodki iz Standardnega modela, ki so imeli tudi pravilno oznako kateremu procesu pripadajo (1. do 4.).

[5] Možna sta še razpada v ZZ^* ter $\tau^+\tau^-$ z verjetnostima 6% in 1%, ki ju analiza tudi upošteva. Eden od bozonov je vedno virtualen (*off shell*), kar označuje *.

[6] Luminoznost je definirana kot $L = \frac{\#\text{trkov}}{\text{cm}^2\text{s}}$ in je za LHC trenutno $\sim 10^{34}$ cm^{-2}s^{-1}. Vrednost 139 fb^{-1} se nanaša na integrirano luminoznost $L = \int L\text{d}t$, ki pove število trkov (zbranih podatkov) v časovnem intervalu in je merilo za zmogljivost pospeševalnika [31].

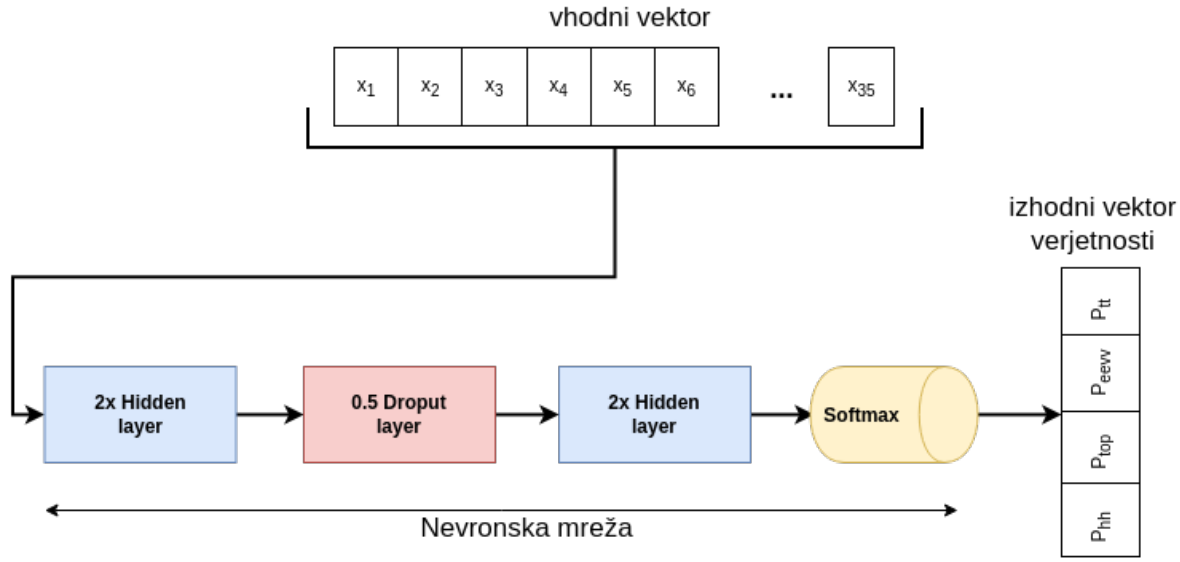


(a) Signalni proces. (b) Primer ozadja, razpad $t\bar{t}$.

Slika 12: Shematični prikaz procesov signala in ozadja pri analizi.

Uporabljena nevronska mreža (slika 13) je imela 5 skritih plasti ter na koncu še plast s softmax transformacijo. Kot vhodni podatek dobi vektor spremenljivk, na izhodu pa vrne štiri dimenzionalen vektor, katerega elementi predstavljajo verjetnosti določenih procesov.

V sredinski plasti je bila uporabljena regularizacija *dropout*. Dropout plast nevronov je skrita plast v kateri pri usmerjeni propagaciji naključno izbrišemo neko število nevronov (50% v tem primeru). Pri vsaki iteraciji čez nov mini-batch se izbriše nova skupina naključno izbranih nevronov. Postopek pomaga zmanjšati *overfit* [7] nevronske mreže, s tem, da poskuša zmanjšati število prostih parametrov. Za aktivacijske funkcije je bila uporabljena ReLU funkcija, za funkcijo izgube križna entropija ter predpis Adam [23] za optimizacijski algoritem. Parametri mreže so bili na začetku naključno inicializirani po normalni porazdelitvi z $\mu = 0$ in $\sigma = \sqrt{1/35}$.

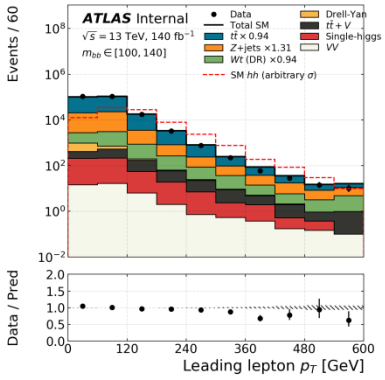


Slika 13: Primer arhitekture nevronske mreže uporabljene pri iskanju signala hh.

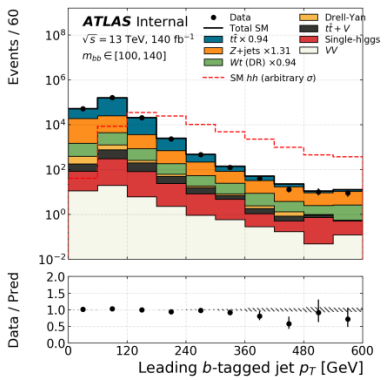
[7] Problem je ekvivalenten prilagajanju polinoma podatkom. Polinom prevelike stopnje na celotnem območju opiše podatke slabše od polinoma nižje stopnje.

Na slikah 14a, 14b, 14c [8] so porazdelitve treh enostavnih količin uporabljenih pri učenju nevronske mreže. Porazdelitvi invariantih mas leptonov in b kvarkov prikazujeta sliki 14d in 14e. Na sliki 14f pa je količina sestavljena iz vsote velikosti transverzalnih gibalnih količin sistema leptonov in sistema b kvarkov, torej kompleksna količina.

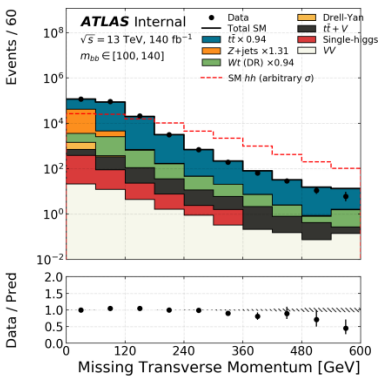
Z uporabo posameznih količin (slika 14) ne moremo ločiti med signalom in ozadjem, zaradi prevelikega prekrivanja. Problem je torej v ločljivosti med signalom in ozadjem. Nevronska mreža omogoča sestavljanje optimalne kompleksne spremenljivke za ločevanje procesov, kar je nujno potrebno pri tej analizi.



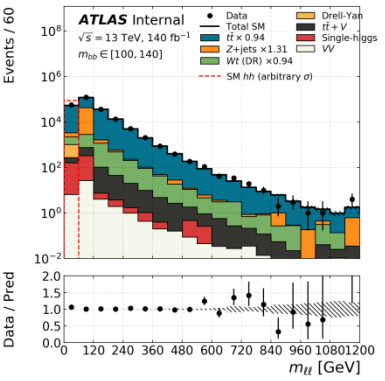
(a) Transverzalna gibalna količina leptona.



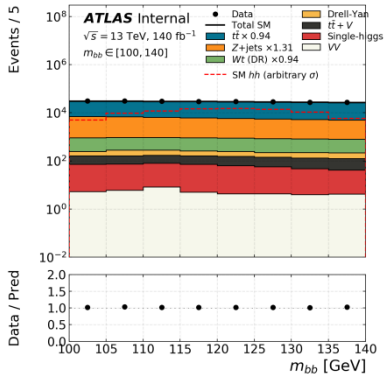
(b) Transverzalna gibalna količina b pljuska.



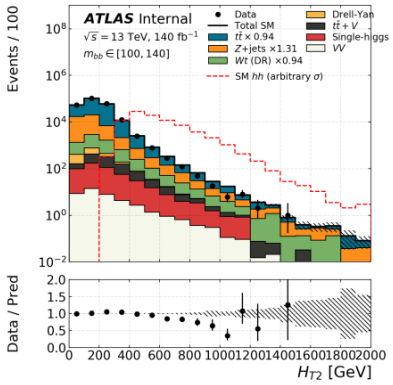
(c) Manjkajoča transverzalna gibalna količina.



(d) Invariantna masa leptonov.



(e) Invariantna masa b kvarkov.

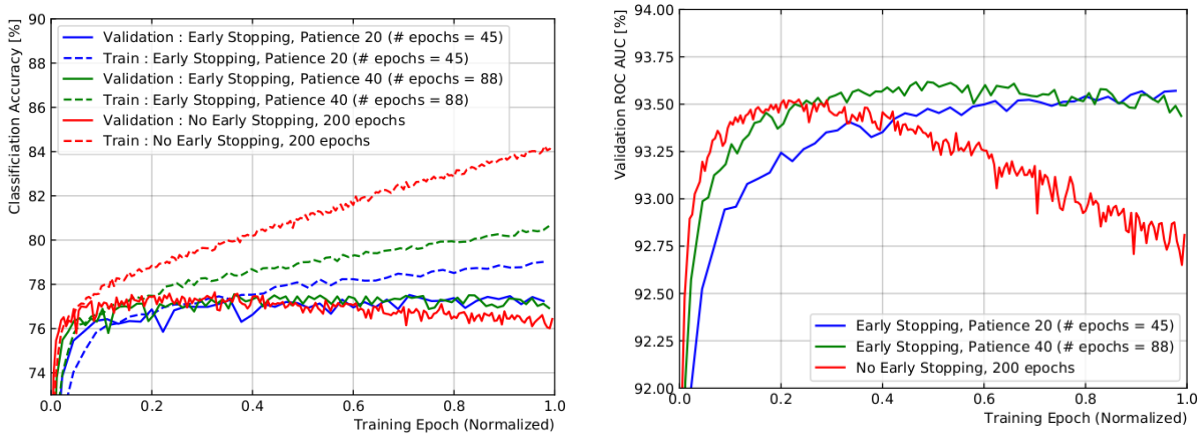


(f) Sestavljena količina.

Slika 14: Porazdelitve nekaterih vhodnih kinematičnih spremenljivk nevronske mreže za proces $WWb\bar{b} \to l\nu l\nu b\bar{b}$ (slika 12a). Izbrani so dogodki za katere: število b pljuskov ≥ 2 in $m_{bb} \in [100, 140]$ GeV. V prvi vrstici so enostavne količine. V drugi vrstici pa so sestavljene količine. Signal ima poljuben sipalni presek in je povečan zaradi vidljivosti [6].

[8] Količina se nanaša na gibalno količino, ki jo detektor ne zazna. Pričakujemo jo zaradi ohranitve energije. Razlog zanjo so delci, ki jih detektor ne zazna (npr. nevtrini).

Rezultat učinkovitosti nevronske mreže pri klasifikaciji še ne videnih primerov (to so lahko dejanske meritve) prikazuje slika 15. Slika 15a kaže natančnost klasifikacije v odvisnosti od števila korakov algoritma (kolikokrat vidi vse podatke) na učni ter validacijski množici. ROC (*receiver operating characteristic*) krivulja pove povezavo med pravilno klasificiranimi pozitivnimi primeri (*true positive rate*) ter nepravilno klasificiranimi pozitivnimi primeri (*false positive rate*). AUC (*area under the curve*) pove ploščino pod to krivuljo, ki je merilo za to kako dobra je klasifikacija (slika 15b). Iz grafov je razvidno, da je učenje nevronske mreže efektivno samo do približno 45 epoh, nad to mejo pa se lahko pojavi prekomerna prilagojenost nevronske mreže podatkom pri učenju (*over-training*).



(a) Klasifikacijska natančnost.

(b) ROC AUC odvisnost.

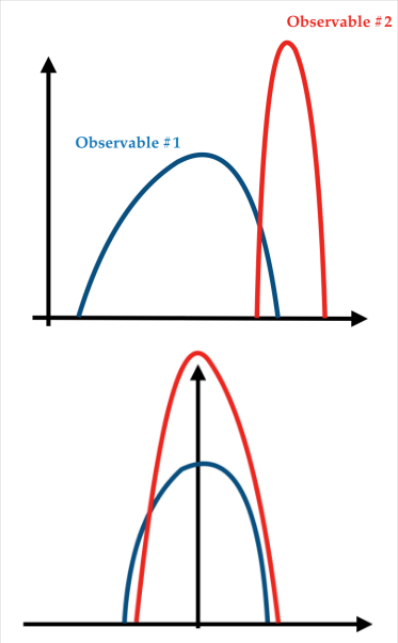
Slika 15: Klasifikacijska napaka nevronske mreže pri opisani analizi [6].

5.2 Delitev podatkov in preprocesiranje

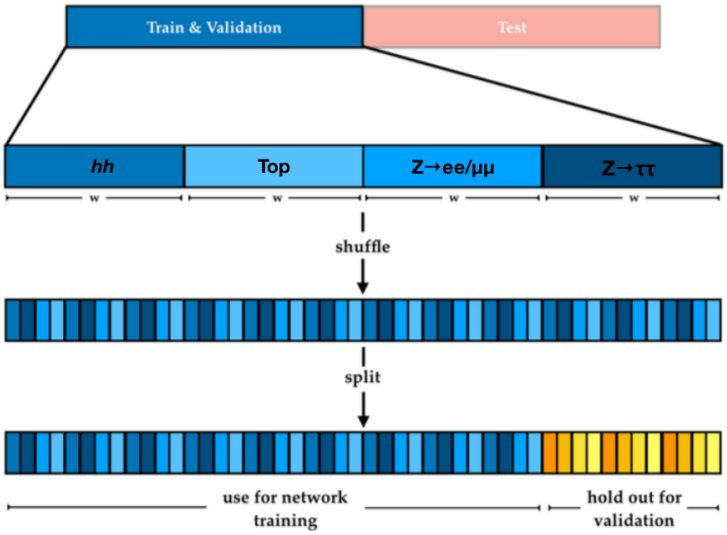
Med učenjem dobi nevronska mreža podatke štirih Monte Carlo procesov, ki ustrezajo procesom signala in ozadja. V simulaciji je bilo generiranih enako število teh štirih procesov zato, da je nevronska mreža procesirala enako število vseh procesov. Polovica vseh je bila uporabljena za testiranje, druga polovica pa je bila naključno premešana in uporabljena v procesu učenja (slika 16b). Validacijska množica je bila uporabljena za določanje zaustavitve učenja (slika 15).

Pred učenjem nevronske mreže je potrebno podatke še nekoliko obdelati (*preprocessing*). Dogodki, ki jih nevronska mreža procesira, so morali imeti vsaj 1 b pljusk [9] ter invariantno maso para leptonov $m_{ll} > 20$ GeV. Poleg tega je bila na podatkih narejena še standardizacija. Postopek vključuje premik vsakega primera za povprečje ter množenje z obratno vrednostjo variance vzorca. Oboje izračunano na celotni množici te spremenljivke (slika 16a). Tak ali podoben postopek reskaliranja podatkov se vedno izvede v strojnem učenju, saj omogoča učinkovitejše delovanje optimizacijskega algoritma, kot je GD. Enak postopek je seveda potrebno izvesti tudi na dejanskih podatkih, ko že imamo naučeno nevronsko mrežo.

[9] Pljuski (*jets*) so eksperimentalna značilnost kvarkov in gluonov, ki nastanejo pri visoko energijskih trkih delcev (npr. protonov v LHC-ju). Kvarki in gluoni nosijo barvni naboj, zaradi česar jih ne opazimo samih. Namesto posameznih kvarkov in gluonov opazimo barvno nevtralne hadrone (sestavljeni iz kvarkov in gluonov) zbrane v skupke, ki letijo ven iz detektorja.



(a) Premik spremenljivke: $x \to \frac{x-\mu}{\sigma^2}$.



(b) Delitev podatkov.

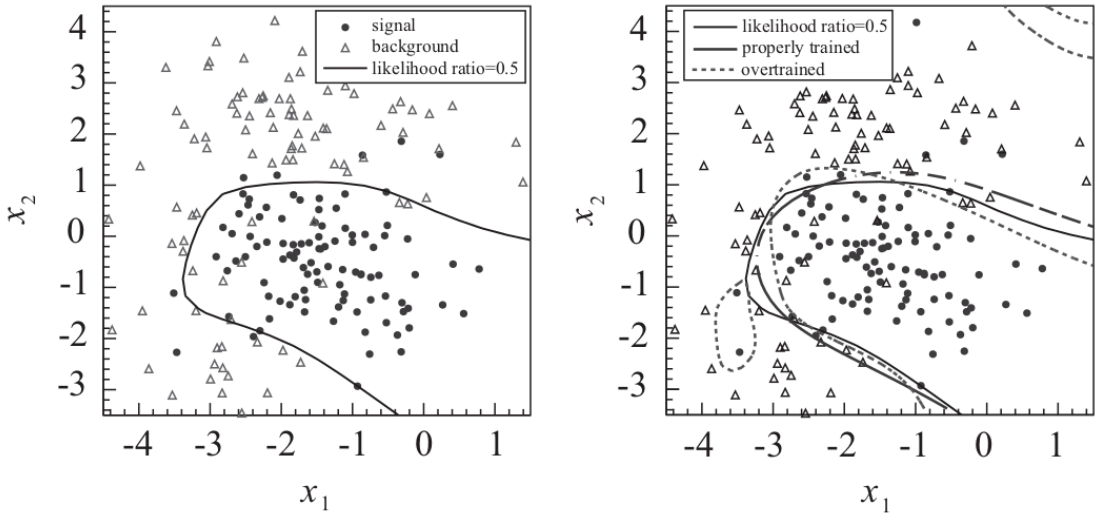
Slika 16: Preprocesiranje podatkov, ki so na voljo.

5.3 Izbira območja signala

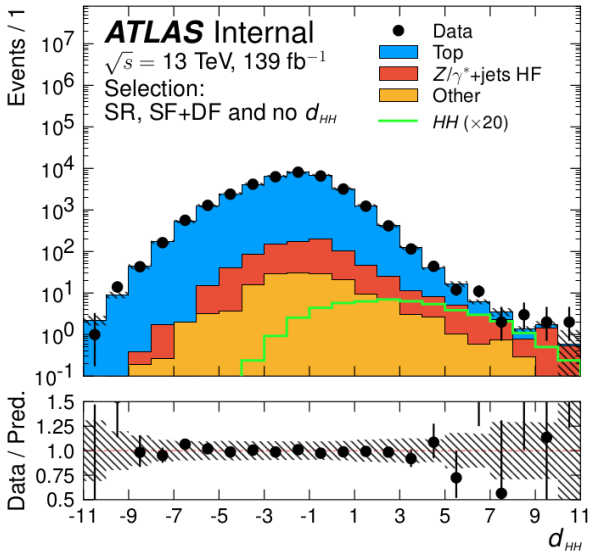
Nevronska mreža vrne verjetnosti za štiri procese: $[p_{hh}, p_{top}, p_{Zsf}, p_{Z\tau\tau}]$. Na sliki 17 je primer, kako nevronska mreža postavi mejo med signalom in ozadjem za dve spremenljivki, pri zahtevi za razmerje verjetnosti 0.5. Iz tega rezultata nevronske mreže analiza sestavi diskriminanto iz razmerij verjetnosti

$$d_{hh} = \ln\left(\frac{p_{hh}}{p_{top} + p_{Zsf} + p_{Z\tau\tau}}\right) , \tag{5.1}$$

ki vsebuje informacijo o zmožnosti nevronske mreže, da loči med štirimi procesi. Območje signala je definirano glede na primerno vrednost d_{hh}, torej na omejitvi vrednosti (rezanju) diskriminante d_{hh}. Takemu pristopu se reče "*cut and count*"analiza. Analiza nadalje razdeli območje glede na simetrijo okusa v procesih (SF in DF). Definirajo območje z istim okusom ($ee + \mu\mu$) in območje z različnim okusom ($e\mu + \mu e$) glede na vrsto leptonov, ki se razlikujeta po vrednosti d_{hh}. Poleg tega pa nadalje omejijo število b pljuskov na ≥ 2, $m_{bb} \in [110, 140]$ GeV ter $m_{ll} < 60$ GeV. Po določitvi zanimivega območja preko d_{hh} se analiza nadaljuje z oceno ozadja v tem območju, določanjem eksperimentalnih ter modelskih sistematskih napak. Slika 18 prikazuje število dogodkov signala (HH) ter število dogodkov ozadja (vse ostalo) v odvisnosti od vrednosti diskriminante d_{hh} na celotnem področju. Opazen ni noben presežek nad pričakovanim ozadjem Standardnega modela.



Slika 17: Odločitvena meja, z zahtevo razmerja verjetnosti (*likelihood ratio*) 0.5, med signalom in ozadjem v dvodimenzionalnem primeru (x_1 in x_2 sta dve od vhodnih spremenljivk), ki jo proizvede nevronska mreža. Desno je prikaz meje ne-optimalno naučene mreže [10].

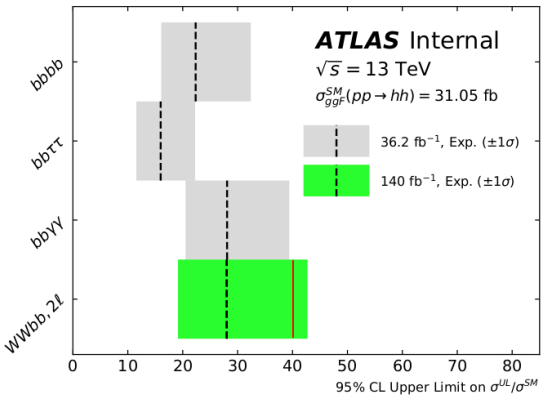


Slika 18: Porazdelitev diskriminante d_{hh} v celotnem območju. Napoved za porazdelitev procesov signala je množena s faktorjem 20 za boljšo vidljivost [7].

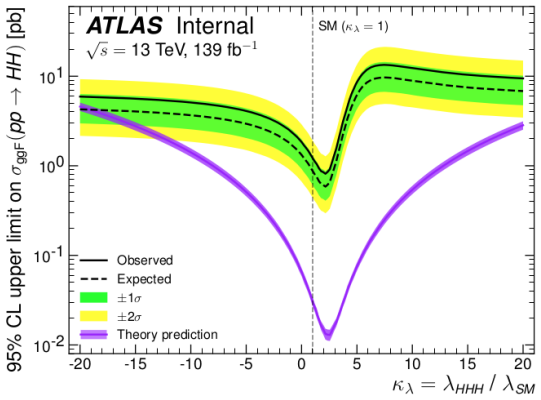
5.4 Rezultati analize

Za določitev sipalnega preseka $hh \to b\bar{b}l\nu l\nu$ je bilo narejeno kinematično prilagajanje napovedi za signal in ozadje na izmerjene podatke. Analiza določi s stopnjo zaupanja (*confidence level*, CL) 95% zgornjo mejo (*upper limit*, UL) sipalnega preseka tvorbe para hh ter nato primerja tako določen sipalni presek σ s teoretičnim, ki je $\sigma^{\mathrm{SM}} = 31.05$ fb. Na sliki 19a je prikazan rezultat za razmerje $\sigma^{\mathrm{UL}}/\sigma^{\mathrm{SM}}$, kjer σ^{UL} označuje izmerjeno zgornjo mejo (UL) za presek. Izmerjena zgornja meja je za faktor 29^{+14}_{-9} višja od preseka napovedanega po Standardnem modelu. Za boljšo omejitev (nižjo zgornjo mejo) bo potrebnih več podatkov, še boljši postopki analize in kombinacija z drugimi analizami, kot so nekatere prikazane na sliki 19a.

Pri analizi so izračunali tudi sipalni presek pri različnih vrednostih $\kappa_\lambda = \lambda_{HHH}/\lambda^{SM}$, kjer je $\lambda_{HHH} = \lambda$ sklopitvena konstanta opisana v poglavju 2. Izračun je bil narejen, da bi preverili možnosti teorij izven Standardnega modela v katerih $\kappa_\lambda \neq 1$. Rezultat prikazuje slika 19b. Dovoljena vrednost κ_λ pri stopnji zaupanja 95% leži na intervalu $[-19, > 20]$. V primerjavi z drugimi podobnimi raziskavami [13] je rezultat ocene intervala precej šibek. Razlog je v tem, da se analiza omeji na ozko območje faznega prostora izbranega glede na d_{hh} v katerem ni območji z ne-standardnimi κ_λ scenariji.



(a) Sipalni presek izračunan pri opisani analizi s statističnimi in eksperimentalnimi napakami (zeleno) ter primerjava z drugimi analizami v različnih kanalih (sivo). Z rdečo črto je prikazana izmerjena vrednost.



(b) Izmerjena zgornja meja sipalnega preseka s stopnjo zaupanja 95% pri različnih vrednostih razmerja med λ_{HHH} in napovedano vrednostjo Standardnega modela λ^{SM}. Prikazana je tudi teoretična napoved pri kateri $\kappa_\lambda \neq 1$. Vertikalna črta prikazuje vrednost po Standardnem modelu, torej 1.

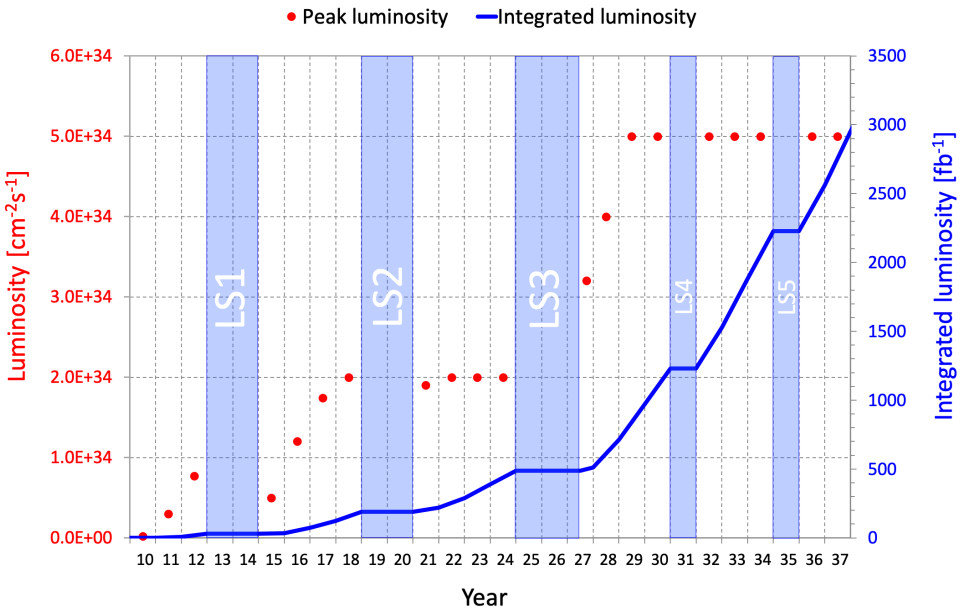
Slika 19: Rezultati opisane analize [6, 7].

6 Zaključek

V zadnjih nekaj letih je napredek strojnega učenja omogočil razvoj orodij, ki lahko izboljšajo ter pohitrijo analizo podatkov v fiziki visokih energij. V prihodnosti je pričakovati še nadaljnjo rast zmogljivosti računalniške opreme ter nadaljnji razvoj strategij učenja, kar bo še povečalo uporabnost različnih algoritmov.

LHC se nadgrajuje še naprej (slika 20). Njegova največja nadgradnja HL-LHC (*High Luminosity Large Hadron Collider*) bo dosegla integrirano luminoznost za faktor tudi do 20 krat večjo od današnje. Nadgradnja bo prinesla nove izzive zaradi velikega števila dogodkov ter s tem povezane velike količine zahtevnih podatkov, ki jih bo potrebno obdelati in analizirati. Uspeh pri iskanju procesov nove fizike bo zaradi tega omejen tudi z zmogljivostjo računalniških virov ter učinkovitostjo algoritmov.

Strojno učenje uporabljeno v fiziki delcev obljublja izboljšave na področjih kot so algoritmi za analizo in rekonstrukcijo, identifikacija delcev, kalibracija detektorja, pohitritev računalniško zahtevnih simulacij, prepoznavanje vzorcev, takojšnja analiza podatkov v prožilnem sistemu ter še kaj.



Slika 20: (HL-)LHC napoved glede na nominalne parametre [21].

Literatura

[1] URL: https://en.wikipedia.org/wiki/Cross_entropy.

[2] URL: https://en.wikipedia.org/wiki/Empirical_risk_minimization.

[3] URL: https://en.wikipedia.org/wiki/Higgs_boson.

[4] URL: https://en.wikipedia.org/wiki/Vacuum_expectation_value.

[5] Pankaj Mehta et al. *A high-bias, low-variance introduction to Machine Learning for physicists*. 2018. arXiv: 1803.08823 [physics.comp-ph].

[6] D. Antrim, A.S. Mete, J. Olsson in A. Taffard. *Search for Higgs boson pair production in the dileptonic WWbb channel in pp collisions at $\sqrt{s}$ = 13 TeV*. ATLAS Note ANA-HDBS-2018-33-INT1. 2019.

[7] D. Antrim, A.S. Mete, J. Olsson in A. Taffard. *Search for non-resonant Higgs boson pair production in the $b\bar{b}l\nu l\nu$ final state with the ATLAS detector in pp collisions at $\sqrt{s}$ = 13 TeV*. ATLAS Paper Draft HDBS-2018-33. 2019.

[8] *ATLAS Experiment*. URL: https://atlas.cern/discover/experiment.

[9] Pierre Baldi, Peter Sadowski in Daniel Whiteson. *Searching for Exotic Particles in High-Energy Physics with Deep Learning*. 2014. arXiv: 1402.4735 [hep-ph].

[10] Olaf Behnke, Kevin Kröninger, Grégory Schott in Thomas Schörner-Sadenius. *Data Analysis in High Energy Physics: A Practical Guide to Statistical Methods*. 2013.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] *CMS detector*. URL: https://cms.cern/detector.

[13] The ATLAS Collaboration. *Combination of searches for Higgs boson pairs in pp collisions at $\sqrt{s}$ = 13 TeV with the ATLAS detector*. ATLAS Paper Draft HDBS-2018-58. 2018.

[14] The ATLAS Collaboration. *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*. 2012. arXiv: 1207.7214 [hep-ex].

[15] The CMS Collaboration. *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*. 2012. arXiv: 1207.7235 [hep-ex].

[16] Jernej Debevc. *Seminar I: Event simulation using machine learning at the ATLAS detector*. 2019.

[17] Nando de Freitas. *Machine Learning*. 2013. URL: https://www.youtube.com/playlist?list=PLE6Wd9FR--EdyJ5lbFl8UuGjecvVw66F6.

[18] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2nd Edition*. O'Reilly, 2019.

[19] Ian Goodfellow, Yoshua Bengio in Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[20] Dan Guest, Kyle Cranmer in Daniel Whiteson. *Deep Learning and its Application to LHC Physics*. 2018. arXiv: 1806.11484 [hep-ex].

[21] *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report*. URL: https://inspirehep.net/literature/1635816.

[22] Borut Paul Kerševan. *Something about new Physics Searches at LHC*. FMF Colloquium, November 2016.

[23] Diederik P. Kingma in Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[24] *Machine Learning in High Energy Physics Community White Paper*. 2019. arXiv: 1807.02876 [physics.comp-ph].

[25] Miha Mali. *Seminar I: Jet Imaging with Deep Neural Networks*. 2019.

[26] Ivan Melo. *Higgs potential and fundamental physics*. 2019. arXiv: 1911.08893 [physics.gen-ph].

[27] Andrew Ng. *Machine Learning*. 2014. URL: https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN.

[28] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/.

[29] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. arXiv: 1609.04747 [cs.LG].

[30] Grant Sanderson. *Neural networks*. 2017. URL: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi.

[31] *Taking a closer look at LHC*. URL: https://www.lhc-closer.es/taking_a_closer_look_at_lhc/1.home.

[32] *The Large Hadron Collider*. URL: https://home.cern/science/accelerators/large-hadron-collider.

[33] *The Standard Model*. URL: https://home.cern/science/physics/standard-model.