UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
FAKULTETA ZA RAČUNALNIŠTVO IN MATEMATIKO

Interdisciplinarni študijski program računalništvo in matematika – 2. stopnja

Nejc Vesel

NASLOV VAŠEGA DELA

Magistrsko delo

Mentor: prof. dr. Peter Peer

Ljubljana, 2018

Zahvala

Neobvezno. Zahvaljujem se ...

Kazalo

Program dela **vii**

1 Sorodna dela **1**
1.1 Generativne nasprotniške mreže 1
1.2 Variacijski autoenkoder 2

Program dela

Mentor naj napiše program dela skupaj z osnovno literaturo. Na literaturo se lahko sklicuje kot [?], [?], [?], [?].

Osnovna literatura

Literatura mora biti tukaj posebej samostojno navedena (po pomembnosti) in ne le citirana. V tem razdelku literature ne oštevilčimo po svoje, ampak uporabljamo okolje itemize in ukaz plancite, saj je celotna literatura oštevilčena na koncu.

[?]

[?]

[?]

[?]

Podpis mentorja:

Naslov vašega dela

Povzetek

Tukaj napišemo povzetek vsebine. Sem sodi razlaga vsebine in ne opis tega, kako je delo organizirano.

English translation of the title

Abstract

An abstract of the work is written here. This includes a short description of the content and not the structure of your work.

Math. Subj. Class. (2010): oznake kot 74B05, 65N99, na voljo so na naslovu `http://www.ams.org/msc/msc2010.html?t=65Mxx`

Ključne besede: integracija, kompleks

Keywords: integration, complex

1 Sorodna dela

1.1 Generativne nasprotniške mreže

Generativne nasprotniške mreže so bile prvič predstavljene v članku [**?**], kjer je predstavljen algoritem tudi podkrepljen s teoretičnimi dokazi. Glavna ideja algoritma je, da pomerimo generativni model proti nasprotniku, ki določa ali je generiran rezultat podoben tistemu, ki ga želimo modelirati. Predstavljamo si lahko bitko med ponarejevalcem denarja in strokovnjakom, ki določa ali je kos denarja pristen. Želimo, da oba akterja skozi iterativni nasprotniški proces izboljšujeta drug drugega.
Rečemo, da sta oba diskriminator kot generator večslojni nevronski mreži. Matematično gledano, želimo naučiti generator G, da bo proizvajal vzorce, katerih porazdelitev je podobna porazdelitvi podatkov, katere želimo modelirati. Da bi se naučili porazdelitev generatorja p_g na podatkih x, definiramo priorni porazdelitev $p(z)$, kjer z predstavlja vektor šuma.
Preslikavo v prostor podatkov predstavimo z $G(z;\theta_g)$, kjer je G odvedljiva funkcija, ki jo predstavlja večslojna nevronska mreža in θ_g njeni parametri. Prav tako definiramo $D(x,\theta_d)$, katerega izhod je skalar $D(x)$,ki predstavlja verjetnost, da je x prišel iz podatkov in ne iz p_g (torej ni bil generiran s pomočjo generatorja). Hočemo torej, da diskriminator za vhod vedno pravilno določi ali je prišel iz množice realnih podatkov oz. ali je bil generiran s pomočjo generatorja G. To zapišemo kot:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Psevdokoda algoritma, ki uporablja gradientni spust za implementacijo našega generativnega nasprotniškega modela, je sledeča:

Algoritem 1 Učenje generativnega nasprotniškega modela s pomočjo gradientnega spusta

1: **for** št. iteracij treninga **do**
2: **for** k korakov **do**
3: Vzorči m vzorcev $\{z^{(1)},\dots,z^{(m)}\}$ iz priorne porazdelitve $p_g(z)$
4: Vzorči m vzorcev $\{x^{(1)},\dots,x^{(m)}\}$ iz porazdelitvne podatkov $p_{data}(x)$
5: Posodobi diskriminator z vzpenjanjem po gradientu

$$\nabla\theta_d \frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))]$$

6: **end for**
7: Vzorči m vzorev $\{z^{(1)},\dots,z^{(m)}\}$ iz priorne porazdelitve $p_g(z)$
8: Posodobi generator s pomočjo gradientnega spusta

$$\nabla\theta_d \frac{1}{m}\sum_{i=1} m\log(1 - D(G(z^{(i)})))$$

9: **end for**

Parameter k za število iteracij notranje zanke, določimo glede na eksperimentalne korake. Pove nam kolikokrat naredimo korak optimizacije diskriminatorja za vsak korak generatorja. Želimo, da generator in diskriminator ostajata v ravnovesju. Če eden od njiju postane preveč učinkovit, težko pride do nadaljnjega izboljšanja v kvaliteti rezultatov, kar je tudi ena glavnih slabosti tega pristopa. Na začetku učenja, ko je G slab, se v praksi lahko zgodi, da D z lahkoto zavrne vse generirane vzorce, saj so očitno različni od podatkov iz učne množice. V tem primeru se vrednost $\log(1 - D(G(z)))$ močno poveča in težko najdemo globalni minimum. Da se izognemo temu problemo, reformuliramo problem tako, da namesto minimizacije $\log(1 - D(G(z)))$ učimo G, da maksimizira $\log D(G(z))$. To nam omogoča. da na začetku učenja dobimo močnešje gradiente, ki omogoča boljše premike v smer optimalne rešitve.
Ena od možnih razširitev omenjenih v članku je razširitev na pogojni generativni model $p(x|c)$, kar dobimo tako, da dodamo pogoj c na vhod tako generatorju kot diskriminatorju.

1.2 Variacijski autoenkoder

Eden od glavnih pristopov pri generativnem strojnem učenju je uporaba variacijskih autoenkoderjev [**?**].
Predstavljajmo si množico $X = \{x^{(i)}\}_{i=n}^{N}$, ki vsebuje N neodvisnih in enakomerno porazdaljenih slučajnih spremenljivk x. Predpostavimo, da so podatki generirani s pomočjo nekega naključnega procesa, ki vključuje slučajno spremenljivko z, ki pa jo ne vidimo.
Ta proces je sestavljen iz dveh korakov

1. Vrednost $z^{(i)}$ je generirana iz predhodne porazdelitve $p_{\theta^*}(z)$

2. Vrednost x^i je generirana iz pogojne porazdelitve $p_{\theta^*}(x|z)$

Predpostavimo, da $p_{\theta^*}(z)$ in $p_{\theta^*}(x|z)$ prihajata iz družine porazdelitev $p_\theta(z)$ in $p_\theta(x|z)$ in da je njihova gostota verjetnosti povsod odvedljiva glede na parametra θ in z. V praksi so vrednosti $z^{(i)}$ ter vrednost θ^* neznane.
Zanima nas rešitev, ki deluje tudi v primeru večje podatkovne množice ter kadar je intergral marginalne verjetnosti $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$ neobladljiv (ne moremo ga odvajati oz. izračunati)