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

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

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

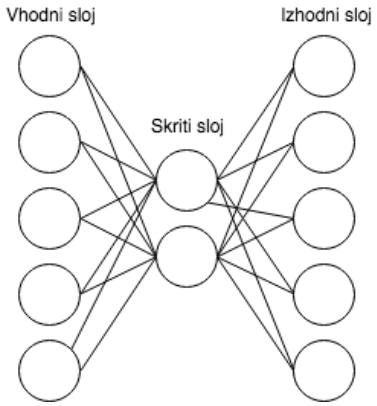
1 Sorodna dela

1.1 Autoenkoder

Ena od glavnih arhitektur na področju generativnih modelov je autoenkoder. Avtoenkoderji so sestavljeni iz dveh glavnih delov, **enkoderja** E in **dekoderja** D. Cilj enkoderja je stisniti vhodne podatke v latentno reprezentacijo manjše dimenzije, cilj dekoderja pa je iz latente reprezentacije rekonstruirati vhodne podatke. Želimo torej

$$E(x) = y \text{ in } D(y) \approx x$$

V splošnem se bo v procesu kodiranja in odkodiranja vedno zgodila izguba informacij, saj je latentni prostor manjše dimenzije kot vhodni podatek. Naprimer, sliko velikost 28x28 točk stisnemo v vektor velikost 10x1.
Nevronsko mrežo učimo tako, da želimo da je rekonstuiran podatek čim bolj podoben vhodnemu. Na prvi pogled se zdi, da je uporabnost avtoenkoderjev omejena na kompresijo podatkov, vendar obstaja velika uporabnost na področju generativnih modelov. Ker je latentni prostor manjše dimenzije, prisilimo mrežo, da v latentni prostor zakodira čim več informacije in tako ohrani najpomembnejše značilke na slikah.
V praksi je to koristno v namene razšumljanja slik, inpaintinga. V teh primerih mrežo naučimo da iz nepopolnih (manjkajočih, šumnatih) slik zgenerira čiste slike.
V najbolj enostavni obliki je autoenkoder sestavljen iz treh slojev. Vhodni sloj je polno povezav z edinim skritim slojem, ki je nato polno povezan z izhodnim slojem. V praksi je najpogostejša izboljšava osnovne arhitekture oblika z več polnopovezanimi skritimi sloji, ki se zmanjšujejo v velikosti na strani enkoderja in povečujejo na strani dekoderja. Če delamo s podatki kot so slike, lahko polnopovezane sloje zamenjamo s konvolucijskimi sloji različnih velikosti.
Dimenzije skritih slojev in latentnega vektorja so odvisne od dimenzij vhodnih podatkov ter od tipa podatkov s katerimi delamo. Parametri modela morajo biti prilagojeni našim podatkom in cilju, ki ga želimo doseči.



Slika 1: Diagram najenostavnejšega autoenkoderja

1.2 Generativne nasprotniške mreže

Generativne nasprotniške mreže so bile prvič predstavljene v članku [1], 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

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

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

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

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

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.

1.3 Variacijski autoenkoder

Eden od glavnih pristopov pri generativnem strojnem učenju je uporaba variacijskih autoenkoderjev [2]. TI so posplošitev autoenkoderja, kjer enkoder pogojimo da ustvarjeni latentni vektorji okvirno sledijo normalni porazdelitvi. Generiranje novih slik je torej samo vzorčenje iz normalne porazdelitve, z določeno srednjo vrednostjo in standardno deviacijo, ki jo dobimo iz mreže. Vedno je potreben kompromis med rekonstrukcijsko napako in prileganjem normalni porazdelitvi. Statistično gledano imamo spremenljivko z, katera generira x. Izračunali bi radi $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$. Izkaše se, da je izračun te porazdelitve v praksi problematičen, zato aproksimiramo to porazdelitev z porazdelitvijo $q(z|x)$, ki ji je podobna in jo znamo izračunati. Za mero podobnosti med dvema porazdelitvama pa uporabimo KL divergenco. V naslednjem delu predstavimo bolj natančno in formalno izpeljavo variacijskega autoenkoderja.
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) in kjer je posteriorna gostota $p_\theta(z|x) = p_\theta(z|x)p_\theta(z)/p_\theta(x)$ neobvladljiva. Ta dva primera, se velikokrat pojavljata prav v nevronskih mrežah z nelinearnimi skritimi sloji.
Pristop z variacijskimi autoenkoderji nam omogoča učinkovito ocenjevanje parametrov θ, kar nam omogoča generiranje umetnih podatkov, ki so podobni naravnim. Kadar imamo neko vrednost x ter izbrane parametre θ, nam VAE omogoča dobiti

aproksimacijo latentne spremenljivke z. To se uporablja v namene kodiranja, kjer informacije iz x zakodiramo v z. Še ena od uporabnih lastnosti pa je aproksimiranje inference spremenljivke x, kar nam omogoča uporabo v smeri razšumenja, "inpaintinga"itd.
Uvedemo prepoznavni model $q_\phi(z|x)$, ki je aproksimacija $p_\theta(z|x)$. Izpeljali bomo metodo, ki se ϕ nauči skupaj s parametri θ Neopazovano spremenljivko z, si lahko predstavljamo kot zakodirano informacijo. Zato model $q_\phi(z|x)$ imenujemo **enkoder**, saj nam glede na podatek x izračuna porazdelitev možnosti z, ki bi lahko generirale ta podatek. Analogno bomo $p_\theta(x|z)$ imenovali **dekoder**, saj nam iz z producira porazdelitev čez vrednosti x.
Izpeljemo lahko spodnjo mejo:

$$\mathcal{L}(\theta, \phi, x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]$$

ki jo želimo optimizirati glede na parametre ϕ in θ. Zaradi velike variance gradienta, je naivna Monte Carlo metoda, za ta primer neučinkovita in potrebujemo boljšo metodo.
Zanima nas cenilka za spodnjo mejo $\mathcal{L}$ in njene odvode. Upoštevajoč nekaj pogojev, ki so bolj natančno razloženi v [2], lahko reparametriziramo $\tilde{z} \sim q_\phi(z|x)$ z odvedljivo preslikavo šuma ϵ, torej $\tilde{z} = g_\phi(\epsilon, x)$, kjer velja $\epsilon \sim p(\epsilon)$. Sedaj lahko vpeljemo Monte Carlo aproksimacijo za pričakovano frednost neke funkcije $f(z)$, glede na $q_\phi(z|x)$. kot

$$\mathbb{E}_{q_\phi(z|x^{(i)})}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g_\phi(\epsilon, x^{(i)}))] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\epsilon^{(l)}, x^{(i)}))$$

, kjer je $\epsilon^{(l)} \sim p(\epsilon)$.
To tehniko apliciramo na naš problem in dobimo cenilko $\tilde{\mathcal{L}}^A(\theta, \phi; x^{(i)}) \simeq \mathcal{L}(\theta, \phi; x^{(i)})$ za katero velja

$$\tilde{\mathcal{L}}^A(\theta, \phi; x^{(i)}) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^(i), z^{(i,l)}) - \log q_\phi(z^{(i,l)}|x^{(i)})$$

kjer

$$z^{(i,l)} = g_\theta(\epsilon^{(i,l)}, x^{(i)})$$

in

$$\epsilon^{(l)} \sim p(\epsilon)$$

Velikokrat lahko $KL - divergenco$ integriramo analitično, tako da je ocena z vzorčenjem potrebna le za rekonstrukcijsko napako. $\mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)$. KL-divergenco si lahko predstavljamo kot regularizacijski člen ϕ, kar nam da drugo različico cenilke $\widetilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) \simeq \mathcal{L}(\theta, \phi; x^{(i)})$ ki je definirana kot

$$\widetilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(x^{(i)}|z^{(i,l)}))$$

in velja

$$z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)}) \text{ in } \epsilon^{(l)} \sim p(\epsilon)$$

Če Iz podatkovnem množice X z N elementi vzorčimo po M vzorcev, lahko skonstruiramo cenilko osnovano na "minibatchih":

$$\mathcal{L}(\theta, \phi; X) \simeq \widetilde{\mathcal{L}}(\theta, \phi; X^M) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\theta, \omega; x^{(i)})$$

Minibatch $X^M = \{x^{(i)}\}_{i=1}^{M}$ je naključno izbran vzorec velikosti M iz množice X. V psevdokodi je algoritem predstavljen kot

Algoritem 2 Minibatch verzija AEVB algoritma. Lahko se uporablja cenilka $\widetilde{\mathcal{L}}^A$ ali $\widetilde{\mathcal{L}}^B$

$\theta, \phi \leftarrow$ inicializacija parametrov
repeat
  $X^M \leftarrow$ Naključen minibatch M vzorcev iz X
  $\epsilon \leftarrow$ Naključni vzorec šuma iz porazdelitve $p(\epsilon)$
  $g \leftarrow \nabla_{\theta,\phi} \widetilde{\mathcal{L}}^M(\theta, \phi; X^M, \epsilon)$ (Gradient minibatch cenilke)
  $\theta, \phi \leftarrow$ Posodobimo parametre s premikom v smeri gradienta (SGD ali ADAGRAD)
until parametra (θ, ϕ) skonvergirata
return θ, ϕ

1.4 Nasprotniški autoenkoder

Nasprotniški autoenkoderji [3] so razširitev autoenkoderjev in so po svoje zasnovi precej podobni variacijskim avtoenkoderjem. Glavna razlika se pojavi v načinu zagotavljanja porazdelitve latentnega vektorja. Variacijski autoenkoder uporablja KL-divergenco kot vodilo za vodenje pravilne porazdelitve, medtem ko se pri nasprotniških avtoenkoderjih uporablja nasprotniško učenje, kjer želimo s pomočjo diskriminatorja doseči isti cilj.
Naj bo x vhodni podatek, z latentni vektor autoenkoderja in $p(z)$ porazdelitev kateri želimo da koda z sledi. Definirajmo $q(z|x)$ kot porazdelitev enkoderja, $p(x|z)$ kot porazdelitev dekoderja, $p_d(x)$ kot porazdelitev podatkov ter $p(x)$ kot porazdelitev našega modela. Enkoder definira porazdelitev $q(z)$ na latentnem vektorju kot

$$q(z) = \int_x q(z|x) p_d(x) dx$$

Nasprotniški autoenkoder je autoenkoder, ki je regulariziran z ujemanjem med $q(z)$ in $p(z)$.
Nasprotniška mreža je zadolžena za vodenje porazdelitve $q(z)$ proti $p(z)$, medtem ko je autoenkoder zaslužen za minimiziranje rekonstrukcijske napake. Velja, da je generator nasprotniške mreže tudi enkoder autoenkoderja $q(z|x)$, ki je zadolžen da prelisiči diskriminator, da ne loči med $q(z)$ in $p(z)$. Učenje vedno izvajamo v dveh delih, najprej učimo nasprotniško mrežo (diskriminator) in s tem regulariziramo porazdelitev latentnega vektorja. S tem se posodobi tudi generator mreže, ki je

hkrati enkoder autoenkoderja tako, da bolje zmede diskriminator. V drugem koraku pa učimo autoenkoder in s tem izboljšujemo rekonstrukcijsko napako.
Možnih je nekaj različnih izbir za enkoder $q(z|x)$

- **Deterministični** Predpostavimo, da je $q(z|x)$ deterministična funkcija x-a. V tem primeru je enkoder podoben enkoderju standardnega autoenkoderja in edini vir stohastičnosti (nepredvidljivosti) v $q(z)$ ostane učna množica $p_d(x)$.

- **Normalno porazdeljeni** Privzamemo, da je $q(z|x)$ normalna porazdelitev, katere srednja vrednost in varianca je dobljena iz enkoderja. Velja torej

  $$z_i \sim \mathcal{N}(\mu_i(x), \sigma_i(x))$$

  V tem primeru stohastičnost dobimo tako iz učne množice kot iz naključnosti naravne porazdelitve.

- **Univerzalni aproksimator** To si predstavljamo kot posplošitev prejšne možnosti. Cilj je, da $q(z|x)$ naučimo kot univerzalni aproksimator. Naj bo enkoder funkcija $f(x, \eta)$, ki na vhod dobi x in naključni šum η s fiksno porazdelitvijo, potem lahko vzorčimo iz katerekoli posteriorne porazdelitve $q(z|x)$, tako da evaluiramo $f(x, \eta)$ pri različnih vzorcih η. Matematično gledano, predpostavimo da velja $q(z|x, \eta) = \delta(z - f(x, \eta))$ in

  $$q(z|x) = \int_\eta q(z|x, \eta)p_\eta(\eta)d\eta \implies q(z) = \int_x \int_\eta q(z|x, \eta)p_d(x)p_\eta(\eta)d\eta dx$$

  Tu je stohastičnost v $q(z)$ dobljena tako iz učne množice kot iz šuma η na vhodu enkoderja. V tem primeru nismo več omejeni na naravno porazdelitev, ampak si lahko izberemo kakršnokoli porazdelitev želimo.

Literatura

[1] I. Goodfellow in dr., *Generative adversarial nets*, v: Advances in Neural Information Processing Systems 27 (ur. Z. Ghahramani in dr.), Curran Associates, Inc., 2014, str. 2672–2680.

[2] D. P. Kingma in M. Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114 (2013).

[3] A. Makhzani in dr., *Adversarial autoencoders*, v: International Conference on Learning Representations, 2016.