

Duboko učenje 1

Uvodno predavanje

Siniša Šegvić

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

SADRŽAJ UVODNOG PREDAVANJA

- **Motivacija** za duboko učenje:
 - strojno učenje i umjetna inteligencija
 - kompozitni podatci (sastoje se od dijelova)
- **O predmetu:**
 - glavne teme
 - način održavanja nastave, razdioba bodova, literatura
- **Pregled osnova **strojnog učenja**:**
 - osnovni pojmovi i tehnike
 - primjeri algoritama
- **Pregled suvremenih izazova i prednosti **dubokog učenja****

MOTIVACIJA: STROJNO UČENJE

- **Strojno učenje:** proučava algoritme čija funkcionalnost nije unaprijed isprogramirana nego je zadana primjerima za učenje:
 - jedan od središnjih problema **umjetne inteligencije**
 - zanimljivi su i hibridni algoritmi koji se dijelom uče a dijelom su ožičeni
- **Umjetna inteligencija:** proučava izradu strojeva koji "misle":
 - zadatci koji su laki za ljude, a vrlo teški za računala
- Primjer: program koji u ovim slikama pronađe kravu
 - inteligentno ponašanje lakše je naučiti nego konstruirati.



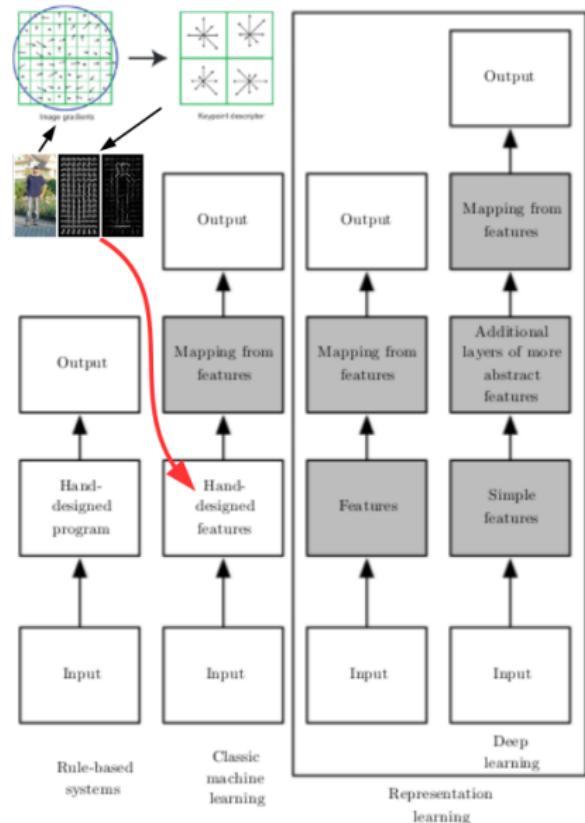
MOTIVACIJA: UMJETNA INTELIGENCIJA

Odnos AI vs ML:

- rani pristupi: nema učenja
- klasični pristupi: naučiti plitki model na konstruiranim značajkama
- reprezentacijski pristupi: naučiti i značajke i model
- duboki pristupi: odjednom naučiti (engl. end to end) niz transformacija

Učenje postaje sve važnije!

Podatkovne reprezentacije isto!



[goodfellow16]

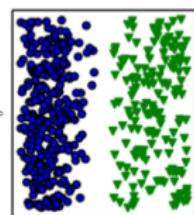
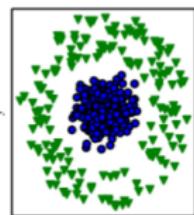
MOTIVACIJA: PODATKOVNE REPREZENTACIJE

Složenost obrade često ovisi o reprezentaciji podataka:

- za dani zadatak, neke reprezentacije mogu biti pogodnije od drugih
- MCMLXXI + XIX vs 1971 + 19?
- polarna reprezentacija problema na slici desno može se klasificirati plitkim modelom

Ponekad značajke nije lako konstruirati ručno

- Grci i Rimljani u više od 1000 godina civilizacije nisu uspjeli izumiti položajni brojevni sustav
- to je značajno otežalo razvoj matematike
 - MCMLXXI + XXIX = ?
 - MXXIV : LXIV = ?
- ⇒ naučene reprezentacije su zanimljive!



[goodfellow16]

MOTIVACIJA: PRIMJER

Kako razlikovati slike goveda od slika bizona?



[image-net.org]

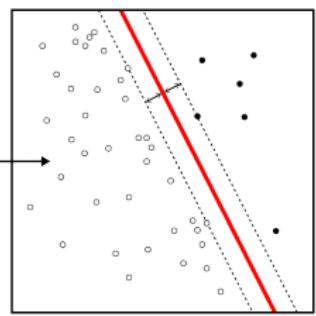
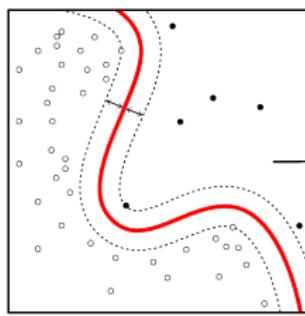
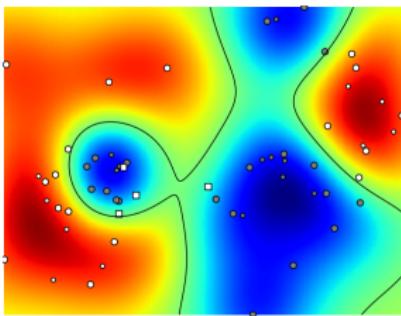
Težina klasifikacije jako ovisi o reprezentaciji podataka

- životinje bi bilo lako razlikovati kad bi neki magični algoritam pretvorio sliku u binarni vektor: [krzno?, grba?, divljina?, ...]
- većina bizona bi bili: [DA, DA, DA, ...]
- većina goveda bi bila: [NE, NE, NE, ...]

Najbolje: **istovremeno** učiti i reprezentaciju i klasifikacijsko pravilo!

MOTIVACIJA: PLITKI MODELI

- Dominantni pristup strojnom učenju 1990-2006: ručno konstruirane značajke i plitki klasifikatori s konveksnim gubitkom
 - SVM, logistička regresija, generalizirani linearni modeli.
- U to vrijeme, prednosti plitkih modela bile su jasne:
 - konvergencija učenja je garantirana i brza
 - jezgreni trik osigurava ogroman reprezentacijski potencijal
 - kompetitivna uspješnost raspoznavanja u praksi



[Wikipedia]

- Ipak, ti postupci ne mogu razlikovati krave od bizona...

MOTIVACIJA: DUBOKO UČENJE

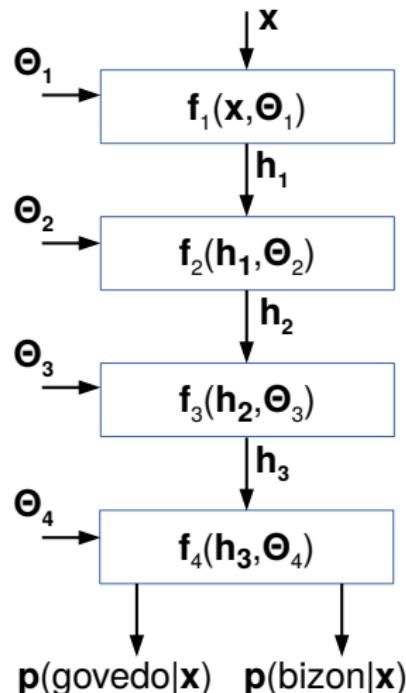
Duboki model: slijed naučenih nelinearnih transformacija.

Zašto duboko učenje nije bilo popularno?

- nema garancije uspjeha učenja
 - nekonveksni gubitak → lokalni minimumi
- nisu mogli nadmašiti stanje tehnike

Zašto je duboko učenje postalo uspješno?

- nove tehnike modeliranja i učenja
- veliki skupovi podataka ($n=10^6$)
- velika procesna moć (TFLOPS)
 - cuda, cuDNN, OpenBLAS, OpenMP



Primjene: razumijevanje slika, jezika i govora, bioinformatika itd.

MOTIVACIJA: NO FREE LUNCH

Duboki modeli nisu "radili" kako treba jer nismo imali:

- dovoljno jake strojeve da dočekamo konvergenciju
- dovoljno podataka da ih naučimo
- tehnike koje pospješuju konvergenciju

Međutim, to ne pojašnjava zašto bi duboki modeli bolje generalizirali od drugih modela velikog kapaciteta (kSVM, stabla, ...)

- algoritme učenja ne možemo vrednovati neovisno o stvarnim podatcima: no free lunch theorem [domingos12cacm]

Uspješnost na neviđenim podatcima ovisi o **pristranosti** algoritma:

- prepostavke modela, gubitka ili optimizacijske metode
- pristranost odgovara podatcima \Rightarrow dobra generalizacija
- logistička regresija: izvrstan odabir za linearne razdvojive razrede

MOTIVACIJA: KOMPOZITNI PODATCI

Kompozitna struktura: temeljna pretpostavka dubokih modela

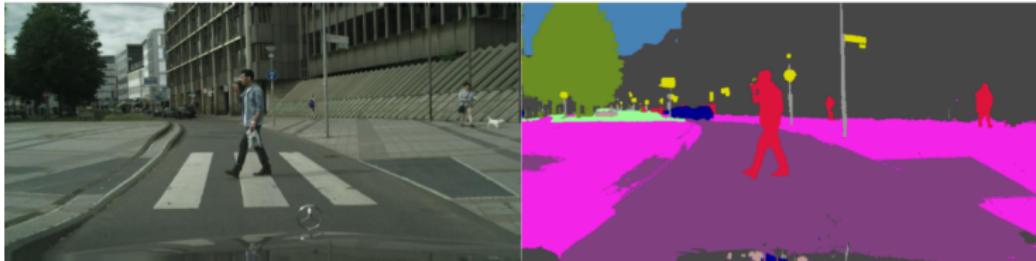
- podatkovna reprezentacija na razini n gradi se obradom reprezentacije na razini n-1
- to svojstvo dubokih modela dobro odgovara podatcima u mnogim teškim zadatcima
 - npr. auto ima kotače, kotači imaju naplatke, naplatci imaju rupice
 - slova čine riječi, riječi sintagme, a sintagme rečenice.



[zeiler14eccv]

MOTIVACIJA: DUBOKO UČENJE - PRIMJENE

- Primjene:
 - računalni vid: klasifikacija slika, lokalizacija objekata, raspoznavanje simbola, semantička segmentacija



[kreso17iccvw]

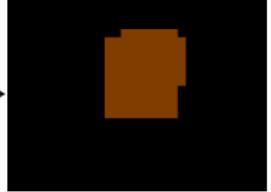
- obrada prirodnog jezika: analiza sentimenta, automatsko prevođenje, generiranje odgovora...
- pretraživanje informacija: učenje rangiranja
- raspoznavanje govora

MOTIVACIJA: JOŠ JEDAN PRIMJER

- Neka su gornje slike zadane kao negativi te donje kao pozitivi:



- Kako biste pronašli gdje su objekti u prethodno neviđenim slikama?



[krapac16gcpr]

O PREDMETU: PLAN

Što ćemo proučavati?

- Osnove unaprijednog dubokog učenja:
 - potpuno povezani modeli
(prva laboratorijska vježba)
 - diskriminativni konvolucijski modeli
(druga laboratorijska vježba)
 - optimizacijske metode
 - regularizacija
- povratni modeli
(treća laboratorijska vježba)
- metrička ugrađivanja
(četvrta laboratorijska vježba)

O PREDMETU: LITERATURA

- Deep Learning. Ian Goodfellow, Yoshua Bengio and Aaron Courville. MIT Press 2016.
- Jan Šnajder, Bojana Dalbelo Bašić. Strojno učenje. FER, Zagreb.
- Neural Networks and Deep Learning. Michael Nielsen. Determination press 2015.
- Deep Learning: Foundations and Concepts. Christopher M. Bishop and Hugh Bishop. Springer 2024.
- Understanding Deep Learning. Simon JD Prince. MIT Press 2023.
- PyTorch tutorials and documentation
 - <https://pytorch.org/tutorials/>
 - <https://pytorch.org/docs/stable/index.html>

O PREDMETU: PREDZNANJE

Potrebno predznanje za praćenje kolegija:

- linearna algebra (§2.1 - §2.11) i teorija vjerojatnosti (§3.1 - §3.11)
- analiza vektorskih funkcija više varijabli (§4.3.1)
- osnovni pojmovi i tehnike strojnog učenja (§5.1 - §5.11)
- osnove programskog jezika Python

Nulta laboratorijska vježba: instrument za provjeru/sticanje predznanja

- možete (trebate!) je odmah riješiti
- ako ne riješite nultu vježbu - prva vježba će vam biti preteška
- gradivo svih laboratorijskih vježbi uči će u međuispit i ispit

O PREDMETU: LABORATORIJ

Predviđjeli smo četiri laboratorijske vježbe:

0. vektorska algebra, plitki modeli, Python i numpy
1. potpuno povezani unaprijedni modeli
2. konvolucijski modeli
3. povratni modeli
4. metrička ugrađivanja

Vježbe se rade kod kuće, a predaju u terminu vježbi

- računalni zahtjevi: Python, Numpy, Scipy, Matplotlib i PyTorch
- krajem semestra: specijalni termin za nadoknadu jedne vježbe

O PREDMETU: LABORATORIJ (2)

Laboratorijske vježbe (Python) su središnji dio predmeta:

- izravnih 20% bodova $4 \times (2.5 + 2.5)$
- barem 20% zadataka na međuispitima i ispitima.

Potrebno je dobiti barem 50% laboratorijskih bodova za izaći na ispit.

Nulta vježba ne donosi bodove i ne provjeravamo je.

O PREDMETU: UVJETI, BODOVI

Aktivnosti: predavanja, vježbe, međuispit, završni ispit, klasični ispit

Kalendar nastave:

kraj ožujka: L1

sredina travnja: L2

kraj travnja: MI

početak lipnja: L3

sredina lipnja: L4

kraj lipnja: ZI

početak srpnja: KI

Kontinuirana provjera:

laboratorij: 10 (A) + 10 (B)

ispiti: 40, 40

preduvjet: 50% laboratorijskih

Klasični ispit:

preduvjet: 50% laboratorijskih

Mogućnost dobivanja **bonus bodova** za: korisne sugestije, prijedloge problemskih zadataka ili vježbi, seminare

- javiti se e-mailom bilo kojem nositelju ili asistentu

Burza grupa: tjedan-dva prije odgovarajuće vježbe (pratite oglase!).

Ocjenjivanje. 2: 50%, 3: 63%, 4: 76%, 5: 89%.

O PREDMETU: PREDAVAČI I ASISTENTI

- predavanja:
 - Siniša Šegvić
 - (Marko Čupić)
 - Marin Oršić
- laboratorijske vježbe:
 - Marin Oršić
 - Ivan Sabolić
 - Anja Delić
 - Ivan Martinović

PREGLED STROJNOG UČENJA: OSNOVNI POJMOVI

Strojno učenje proučava oblikovanje postupaka čija se uspješnost poboljšava s iskustvom

Algoritam strojnog učenja obuhvaća sljedeća dva postupka:

- postupak obrade podataka (**model**), npr. klasificiranje slike
 - nužno zadati mjeru **performanse** (uspješnost) ili mjeru **pogreške**
 - ◊ npr. točnost klasifikacije na **ispitnom skupu** $\{x_i, y_i\}$
 - ◊ npr. broj krivo klasificiranih ispitnih primjera
 - razlikujemo **empirijsku** i **generalizacijsku** uspješnost
- postupak za **optimiranje** slobodnih parametara postupka obrade na podatcima za učenje npr. gradijentni spust
 - nužno zadati **skup za učenje** $\{x_i, y_i\}$
 - ◊ mora biti uzorkovan iz iste distribucije kao i ispitni skup!
 - nužno zadati kriterij optimizacije (**gubitak**)
 - ◊ npr. negativna log-izglednost parametara na skupu za učenje

PREGLED STROJNOG UČENJA: DEFINICIJE

Algoritam strojnog učenja definiran je:

- **modelom**: postupak obrade sa slobodnim parametrima
- **gubitkom**: formalizacija (anti-)dobrote parametara modela
- **metodom optimizacije**: način pronalaženja parametara koji minimiziraju gubitak.

S obzirom na kvalitetu podataka za učenje, razlikujemo:

- **nadzirano** učenje: svaki podatak označen željenim izlazom
 - tipični zadatci: **klasifikacija, regresija**
- **nenadzirano** učenje: dostupni su samo podatci
 - tipični zadatci: **estimiranje gustoće, generiranje podataka.**
- **podržano** (ojačano) učenje: dostupna je povratna veza o kvaliteti međudjelovanja s okolinom.

PREGLED STROJNOG UČENJA: DEFINICIJE (2)

Izglednost parametara modela $\text{lik}(\theta)$:

- mjeri koliko dobro model objašnjava opažene podatke
- odgovara vjerojatnosti koju model dodjeljuje skupu za učenje

$$\begin{aligned}\text{lik}(\theta) &= P(\{y_i\} | \{\mathbf{x}_i\}, \theta) \\ &= \prod_{i=1}^N P(y_i | \mathbf{x}_i, \theta)\end{aligned}$$

Često razmatramo **log-izglednost** (prikladnija za diferenciranje):

- jednako rangira modele (logaritam monotono raste),
- pogodna za dosljednu formulaciju gubitka (NLL, MLE)

$$-\log \text{lik}(\theta) = -\sum_{i=1}^N \log P(y_i | \mathbf{x}_i, \theta)$$

PREGLED STROJNOG UČENJA: PRIMJER

Logistička regresija: nadzirana klasifikacija u više razreda

- model vraća aposteriornu distribuciju vjerojatnosti razreda:

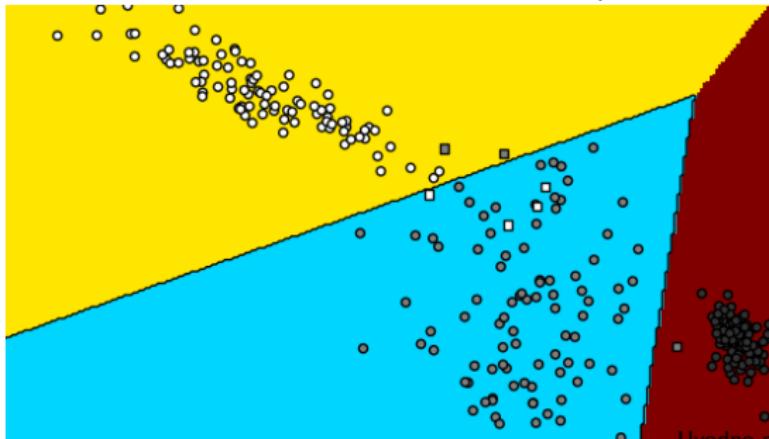
$$P(Y | \mathbf{x}) = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b}), \text{softmax}(\mathbf{s}) = [e^{s_j} / \sum_k e^{s_k}]^\top.$$

- gubitak (negativna log-izglednost modela na skupu za učenje):

$$\mathcal{L}(\mathbf{W}, \mathbf{b} | \mathbf{Y}, \mathbf{X}) = - \sum_i \log P(Y = y_i | \mathbf{x}_i)$$

- optimizacijski postupak (gradijentni spust):

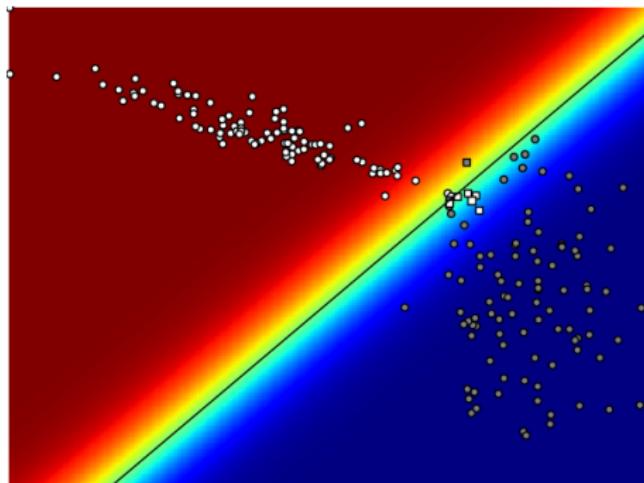
$$\mathbf{b}_{i+1} = \mathbf{b}_i - \delta \cdot \nabla_{\mathbf{b}_i} \mathcal{L} = \mathbf{b}_i - \delta \cdot \left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_i} \right)^\top$$



PREGLED STROJNOG UČENJA: PRIMJER 2

Stroj s potpornim vektorima (nadzirana binarna klasifikacija):

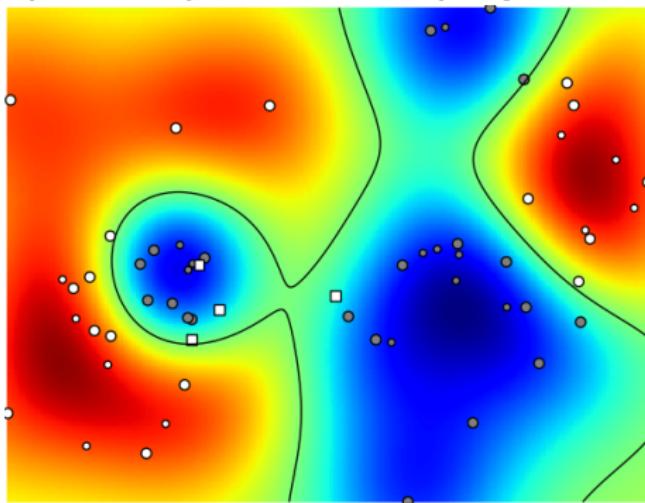
- model (pričinost podatka): $f(\mathbf{x}) = \begin{cases} c_0 & \text{ako } \mathbf{w}^\top \mathbf{x} + b < 0 \\ c_1 & \text{ako } \mathbf{w}^\top \mathbf{x} + b > 0 \end{cases}$
- gubitak (ukupna povreda margin plus regularizacija):
$$\mathcal{L}(\mathbf{w}, b \mid \mathbf{Y}, \mathbf{X}) = \lambda \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^n \max(0, 1 - (-1)^{\llbracket y_i \neq c_1 \rrbracket} (\mathbf{w}^\top \mathbf{x}_i + b))$$
- optimizacija: kvadratno programiranje ili gradijentni spust



PREGLED STROJNOG UČENJA: PRIMJER 3

Stroj s potpornim vektorima \mathbf{x}_j , težinama α_j i jezgrenom funkcijom k :

- model: $f(\mathbf{x}) = \begin{cases} c_0 & \text{ako } \sum_j (-1)^{\llbracket y_j \neq c_1 \rrbracket} \cdot \alpha_j \cdot k(\mathbf{x}_j, \mathbf{x}) + b < 0 \\ c_1 & \text{ako } \sum_j (-1)^{\llbracket y_j \neq c_1 \rrbracket} \cdot \alpha_j \cdot k(\mathbf{x}_j, \mathbf{x}) + b > 0 \end{cases}$
- gubitak (regularizacija + prijestup margine): $\mathcal{L}(\boldsymbol{\alpha}, b) = h(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top) + \frac{1}{n} \sum_{i=1}^n \max \left(0, 1 - (-1)^{\llbracket y_i \neq c_1 \rrbracket} (\sum_j (-1)^{\llbracket y_j \neq c_1 \rrbracket} \alpha_j \cdot k(\mathbf{x}_j, \mathbf{x}_i) + b) \right)$
- optimizacija: kvadratno programiranje ili gradijentni spust



PREGLED STROJNOG UČENJA: POSREDNA OPTIMIZACIJA

Specifičnost strojnog učenja: uspješnost optimiramo **posredno**

- optimizacijska metoda **ne "vidi"** gubitak na ispitnom skupu
 - ali pretp. da empirijska distribucija odgovara generativnoj distribuciji
 - tj. podatke za učenje i ispitivanje generira isti slučajni proces
- gubitak se često ne može poistovijetiti s empirijskom pogreškom
 - tipično zato što formulacija pogreške nije derivabilna
 - tada je gubitak **zamjena** (engl. proxy) za empirijsku pogrešku
 - međutim, dobro definiran zamjenski gubitak (ML) može poboljšavati generalizaciju i nakon što empirijska greška padne na nulu!

PREGLED STROJNOG UČENJA: KAPACITET

Kapacitet je osnovno svojstvo modela:

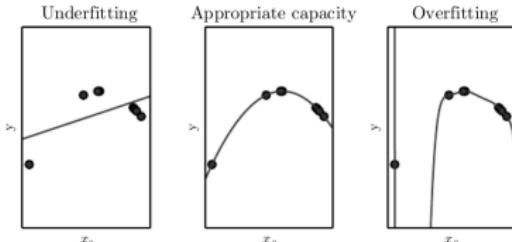
- opisuje sposobnost prilagođavanja podatcima
- najčešće je proporcionalan broju stupnjeva slobode modela
- formalno može se mjeriti brojem podataka (VC dimenzija) koje model može **razbiti** (tj. objasniti uz proizvoljno označavanje)

Modeli malog kapaciteta skloni su **podnaučenosti**:

- ne postoji skup parametara koji može objasniti skup za učenje.

Modeli velikog kapaciteta skloni su **prenaučenosti**:

- previše skupova parametara objašnjavaju skup za učenje.



[goodfellow16]

Uvodno predavanje → Pregled strojnog učenja (7) 27/43

PREGLED STROJNOG UČENJA: UTJECAJ PODATAKA

Proučit ćemo kako empirijska i generalizacijska uspješnost naučenog modela ovise o veličini skupa za učenje

Razmotrit ćemo seriju eksperimenata vezanih uz skalarnu regresiju $y = f_{\theta}(x)$ koji su prikazani na sl. 5.4 knjige [goodfellow16book]

Podatci:

- nezavisna varijabla x : slučajno uzorkovana na konačnom intervalu
- zavisna varijabla y : polinom 5. stupnja + umjereni šum

Modeli:

- polinom drugog stupnja (uče se tri parametra, potkapacitiran)
- polinom optimalnog kapaciteta (n određen iscrpnom validacijom)

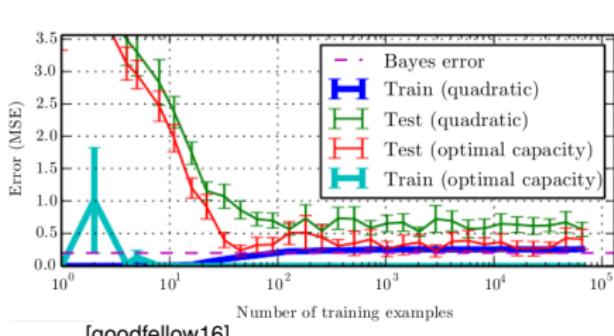
PREGLED STROJNOG UČENJA: UTJECAJ PODATAKA (2)

Definirajmo **neizbjježnu** (Bayesovu) pogrešku(ljubičasto):

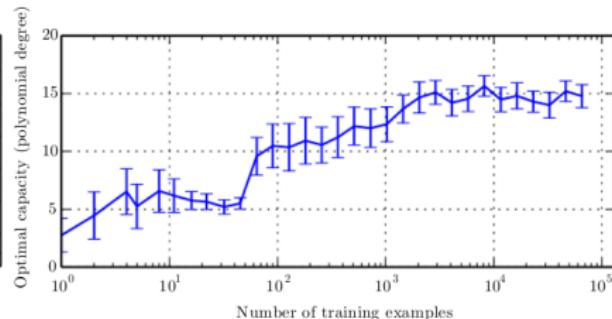
- javlja se zbog stohastičke prirode podataka ili šuma u označavanju

Kako skup za učenje raste, algoritam optimalnog kapaciteta teži:

- zanemarivoj empirijskoj grešci (svijetlo plavo)
- generalizacijskoj grešci (crveno) koja je bolja nego kod potkapacitiranog modela (zeleno) a konvergira **neizbjježnoj** (ljubičasto)



[goodfellow16]



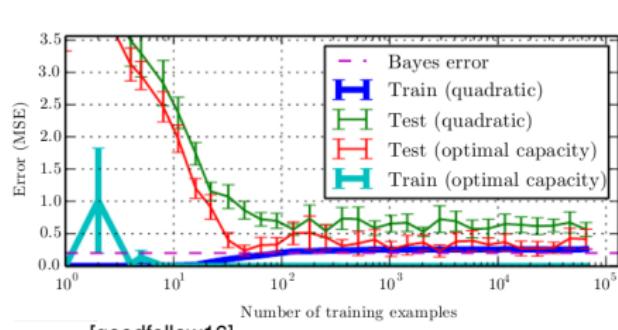
PREGLED STROJNOG UČENJA: UTJECAJ PODATAKA (3)

Kako skup za učenje raste, potkapacitirani algoritam teži:

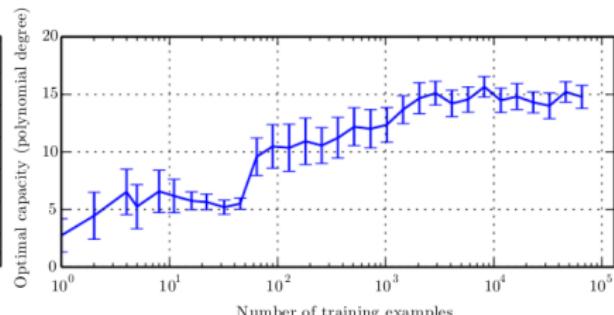
- zamjetnoj empirijskoj grešci (tamno plavo)
- generalizacijskoj grešci (zeleno) koja je veća od optimalne (crveno)

Model s viškom kapaciteta može biti bolji od modela koji zrcali složenost procesa koji je generirao podatke ($n=5$ + šum, desno)

Kad imamo malo podataka za učenje, algoritam s prilagođenim kapacitetom može biti lošiji od potkapacitiranog algoritma (desno)



[goodfellow16]



PREGLED STROJNOG UČENJA: LEX PARSIMONAE

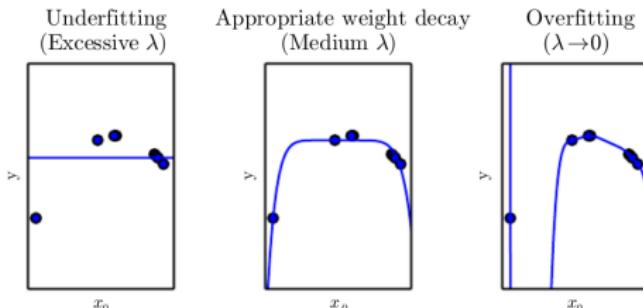
Ograničavanje kapaciteta modela nije jedini način usklađivanja algoritma s pristranošću podataka.

Drugi način je zadržati visoki kapacitet ali uvesti preferenciju prema **jednostavnijim** modelima modificiranjem **gubitka**.

Ako promatramo regresiju, jedan način regulariziranja gubitka bio bi:

$$J(\mathbf{w}) = \lambda \mathbf{w}^\top \mathbf{w} + \sum_i (\sum_j w_j x_i^j + b - y_i)^2.$$

Algoritam sada apriorno preferira rješenja u kojima promjene ulaza vode na blage promjene odluke modela.



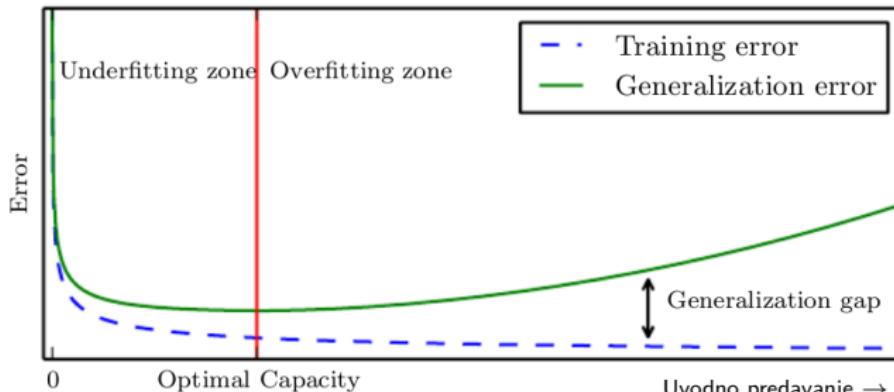
[goodfellow16]

PREGLED STROJNOG UČENJA: REGULARIZACIJA

Svaku modifikaciju koja namjerava **poboljšati generalizaciju** bez smanjivanja empirijske pogreške nazivamo **regularizacijom** algoritma.

Regularizacija se može primijeniti na sve dijelove algoritma:

- gubitak: kažnjavanje norme vektora parametara
- optimizacijsku metodu: rano zaustavljanje učenja
- podatke: rastresanje, zašumljivanje oznaka
- model: složenost preslikavanja, vezivanje parametara



PREGLED STROJNOG UČENJA: STATISTIČKI POGLED

Podnaučenost i prenaučenost možemo pojasniti ako generalizacijsku pogrešku rastavimo na doprinose pristranosti i varijance.

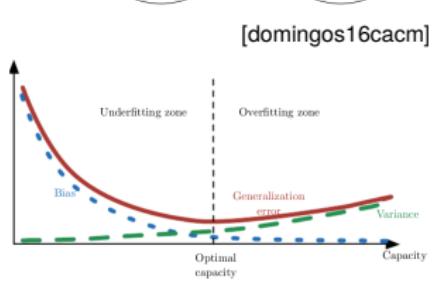
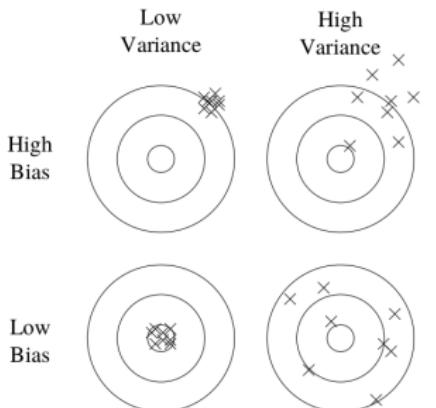
Pristranost algoritma: ugrađena sklonost prema nekim rješenjima.

Varijanca algoritma: sklonost da rezultat jako varira o ulaznim podatcima.

Regularizacija algoritma smanjuje varijancu i povećava pristranost.

Za najbolje rezultate mjeru regularizacije treba prilagoditi podatcima:

- izbjegći i lošu pristranost i pretjeranu varijancu.



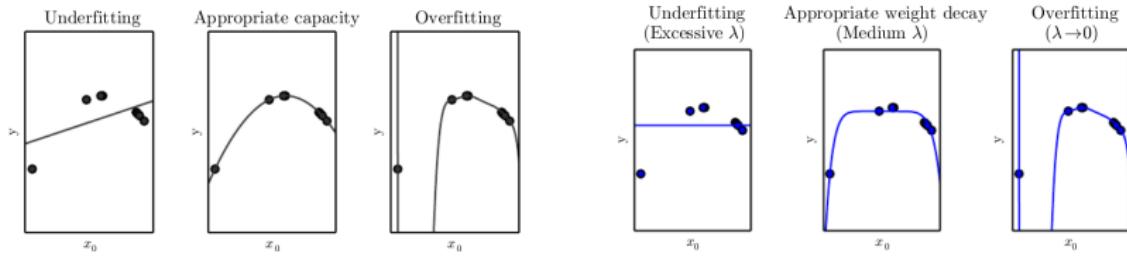
[goodfellow16]

PREGLED STROJNOG UČENJA: HIPERPARAMETRI

Većina algoritama ima tzv. **hiperparametre** koji reguliraju njihovo ponašanje, a nisu obuhvaćeni optimizacijom

Primjeri: kapacitet modela, faktor kazne parametarske norme, korak optimizacije, broj epoha učenja...

Odabir takvih parametara najčešće provodimo iscrpnim ili slučajnim pretraživanjem prostora parametara na **validacijskom skupu**



[goodfellow16]

PREGLED STROJNOG UČENJA: GUBITAK

Najintuitivniji gubitak je srednja kvadratna pogreška:

$$J(\theta) = \sum_i (\text{model}(\mathbf{x}_i) - y_i)^2$$

Taj gubitak nije prikladan za probabilističku klasifikaciju jer ignorira da model vraća vjerojatnosnu distribuciju

- npr. $d_{L2}^2([1,0,0], [0.2, 0.4, 0.4]) < d_{L2}^2([1,0,0], [0.2, 0.0, 0.8])$

Dosljednija formulacija gubitka u probabilističkom okruženju jest negativna log-izglednost parametara modela:

$$J(\theta) = -\frac{1}{N} \sum_i \log P_{\text{model}}(y_i | \mathbf{x}_i, \theta).$$

Može se pokazati da je negativna log-izglednost specijalni slučaj unakrsne entropije (ekvivalent KL divergencije).

Ako modeliramo regresijsko odstupanje Gaussovom razdiobom, negativna log-izglednost se svodi na **kvadratni gubitak**.

PREGLED STROJNOG UČENJA: SGD

Jedna od trenutno najpopularnijih optimizacijskih metoda u strojnom učenju jest **stohastički gradijentni spust**

Gradijentni spust po negativnoj log-izglednosti mora izračunati:

$$\nabla_{\theta} J(\theta) = \frac{1}{N} \sum_i^N \nabla_{\theta} L(\mathbf{x}_i, y_i, \theta).$$

Kod teških problema optimizacija sporo napreduje jer: $N \cdot \dim(\theta) \sim 10^{12}$

Problem rješavamo odvajanjem podataka u manje **grupe** (engl. batch):

$$\nabla_{\theta} J(\theta) = \frac{1}{N'} \sum_i^{N'} \nabla_{\theta} L(\mathbf{x}_i, y_i, \theta).$$

- korak optimizacije izvodi se u vremenu $O(N' \cdot \dim(\theta))$, $N' \ll N$
- grupe se slučajno formiraju nakon svake epohe (stohastika!)
- u velikim modelima ($\dim(\theta) \sim 10^6$): brže od metoda višeg reda
- koristi se i kod velikih plitkih modela!
- SVM s jezgrom: prostor $\sim O(N^2)$, vrijeme $\sim O(N^3)$.

PREMA DUBOKOM UČENJU: IZAZOV

Problemi UI-razine razmatraju složene podatke ($D=\dim(\mathbf{x}_i) \sim 10^5$)

Zbog toga broj svih mogućih podataka iznosi barem $O(2^D)$

Ako pristranost klasifikatora ne pogađa pristranost podataka,
moramo imati predstavnika u svakoj hiper-kocki prostora podataka

- u *pesimističnom* slučaju treba nam $O(2^D)$ primjera za učenje!
- oblik **prokletstva dimenzionalnosti** (curse of dimensionality)



[goodfellow16]

PREMA DUBOKOM UČENJU: KLASIČNI ODGOVOR

Klasični klasifikatori pristupaju tom problemu pod pretpostavkom **glatkoće** (ili **lokalne konstantnosti**) modela:

rezultat modela ne bi se smio "mnogo" mijenjati unutar kompaktne regije prostora podataka.

Ovakvu pristranost implementiraju k-NN, jezgrene metode, stabla

- svi ti pristupi zahtijevaju $O(n)$ primjera za učenje za razlikovanje $O(n)$ regija u prostoru podataka
- takvi pristupi ne mogu funkcionirati kad je $n = 2^{10^5}$

Možemo izvesti sljedeće zaključke:

- glatkoća je OK, ali ne može se nositi s porastom dimenzionalnosti
- trebaju nam **pristrani** modeli prilagođeni **stvarnim podatcima**.

PREMA DUBOKOM UČENJU: KOMPOZITNI PODATCI

Temeljna pretpostavka **dubokih modela**: podatci su generirani rekurzivnom **kompozicijom dijelova**

- osoba ima glavu, glava ima lice, lice ima oči, oko ima šarenicu

Potencijal za nezavisno učenje značajki nižih razina:

- osoba s plavim očima i crnom kosom može doprinijeti prepoznavanju osoba s plavim očima i crvenom kosom.

Duboke modele možemo izraziti manjim brojem značajki

- npr. naučiti xor_n kao zbroj minterma: $f_n^{\text{exp}}(\mathbf{b}) = \sum_{j=1}^{2^n} w_j \cdot [\![\mathbf{m}_j = \mathbf{b}]\!]$
 - O(2ⁿ) binarnih značajki (i parametara)
- npr. naučiti xor_n kao kompoziciju *dvoulaznih* logičkih funkcija f_i :
 - $f_n^{\text{lin}}(\mathbf{b}) = f_1(b_1, f_2(b_2, \dots, f_{n-1}(b_{n-1}, b_n) \dots))$
 - O(4n) binarnih značajki (i parametara)

Ovakav pristup može se suprotstaviti prokletstvu dimenzionalnosti.

PREMA DUBOKOM UČENJU: KOMPOZITNI PODATCI (2)

Kompozitna struktura: važan oblik pristranosti dubokih modela

- podatkovna reprezentacija na razini n gradi se od reprezentacija na razini n-1
- to svojstvo dubokih modela dobro odgovara podatcima u mnogim teškim zadatcima
 - npr. auto ima kotače, kotači imaju naplatke, naplatci imaju rupice
 - slova čine riječi, riječi sintagme, a sintagme rečenice.



PREMA DUBOKOM UČENJU: UČENJE MNOGOSTRUKOSTI

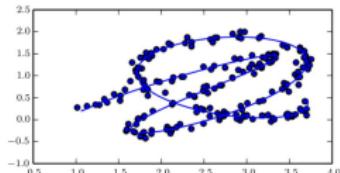
Mnogostrukost: povezani skup $\{x_i\} \in \mathbb{R}^n$ koji se lokalno može aproksimirati skupom $\{x'_i\} \in \mathbb{R}^m$, $m \ll n$ (lijevo!).

Duboki modeli pretpostavljaju da podatci stanuju na niskodimenzionalnoj mnogostruktosti

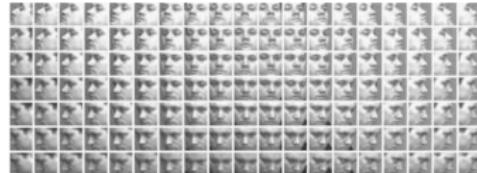
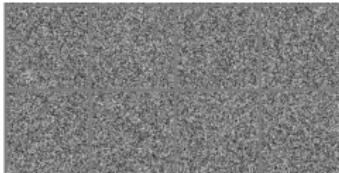
- model se specijalizira na dijelove podatkovnog prostora s velikom gustoćom vjerojatnosti podataka

Koliko je ta pretpostavka zadovoljena u praksi?

- većina svih mogućih ulaznih vektora nisu valjani podatci (sredina!)
- možemo zamisliti skup transformacija koje definiraju obrise mnogostrukosti: svjetlina, kontrast, rotacija (desno!) itd.



[goodfellow16]



PREMA DUBOKOM UČENJU: ZAKLJUČAK

Duboki modeli su **pristrani**: bolje se mogu prilagoditi podatcima koji se sastoje od dijelova

- ima ih smisla koristiti kad su podatci **kompozitni**
- inače, bolje rezultate mogli bi dati plitki modeli

Duboki modeli su **skalabilni**, mogu raditi s:

- visokodimenzionalnim podatcima ($D=10^5$)
- ogromnim skupovima za učenje ($N=10^6$)
- ogromnim brojem parametara ($\dim(\theta)=10^9$)

Zbog toga su duboki modeli **metoda izbora** kod mnogih problema koje danas svrstavamo u UI

ZAHVALA

Ova predavanja proizšla su iz istraživanja koje je finansirala Hrvatska zaklada za znanost projektom I-2433-2014 MultiCLoD.



<http://multiclod.zemris.fer.hr>

Unaprijedni duboki modeli

Josip Krapac i Siniša Šegvić

- Što su unaprijedni duboki modeli?
- Funkcija gubitka i izlazni slojevi.
- Aktivacijske funkcije u skrivenim slojevima.
- Univerzalna aproksimacija: dubina je važna.
- Backprop: izračun gradijenta kompozicije funkcije prosljeđivanjem greške unazad.

- Što su unaprijedni duboki modeli?
- Funkcija gubitka i izlazni slojevi.
- Aktivacijske funkcije u skrivenim slojevima.
- Univerzalna aproksimacija: dubina je važna.
- Backprop: izračun gradijenta kompozicije funkcije prosljeđivanjem greške unazad.

Što su unaprijedni duboki modeli?

Duboka unaprijedna mreža (eng. deep feedforward network)

- osnovna formulacija dubokog modela
- mreža: sastoji se od većeg broja jednostavnijih premreženih funkcija
- osnovni oblik odgovara sljedu afinih transformacija s nelinearnom aktivacijom

Cilj:

- aproksimirati funkciju $y = f^*(x)$ parametarskim modelom $\hat{y} = f(x, \Theta)$.
- model f mapira ulaz x u prediktirani izlaz \hat{y}
- parametre Θ združeno učimo na podatcima s kraja na kraju

Što su unaprijedni duboki modeli?

Detalji cilja:

- funkcija $y = f^*(x)$ opisuje **stvarni odnos** između ulaza x i izlaza y
- naš model $\hat{y} = f(x, \Theta)$ **aproksimira** stvarnu funkciju.
- želimo pronaći skup parametara Θ^* koji daje najbolju aproksimaciju funkcije $f^*(x) \approx f(x, \Theta^*)$.
- **problem:** ne znamo kako funkcija $f^*(x)$ izgleda za svaki x ; znamo samo kako f izgleda na ograničenom skupu za učenje $\{(x_i, y_i)\}_{i=1}^N$.
- stalo nam je do ispravne **generalizacije**
- izbor modela mora biti prilagođen podatcima (usp. no free lunch theorem)

Što su unaprijedni duboki modeli?

Osnovna svojstva dubokih modela:

- informacija struji od ulaza prema izlazu, nema petlji.
- mogu se predstaviti kompozicijom jednostavnijih funkcija:
$$f(\mathbf{x}, \Theta) = o(f_L(f_{L-1}(\cdots(f_1(\mathbf{x}, \Theta_1)), \cdots), \Theta_{L-1}), \Theta_L)),$$
- jednostavne funkcije f_i nazivamo slojevima
- svaki sloj ima točno jednu nelinearnu aktivaciju
- dubina modela (L): broj slojeva

Što su unaprijedni duboki modeli?

Model možemo izraziti i preko pomoćnih varijabli h_L, h_{L-1}, \dots, h_1

$$h_1 = f_1(x, \Theta_1)$$

⋮

$$h_{L-1} = f_{L-1}(h_{L-2}, \Theta_{L-1})$$

$$h_L = f_L(h_{L-1}, \Theta_L)$$

$$f(x, \Theta) = o(h_L)$$

pomoćne varijable zovemo **skrivenim** ili **latentnim značajkama**

širina l -tog sloja: dimenzija značajki u l -tom sloju: $h^l \in \mathbb{R}^{d_l}$.

Samo su ulaz x i izlaz y specificirani, model ima slobodu da iskoristi **skrivene slojeve** na način koji osigurava najbolju aproksimaciju funkcije.

Što su unaprijedni duboki modeli?

Osnovni oblik: lanac potpuno povezanih slojeva

- svaka f_i modelira elementarnu nelinearnu transformaciju: afino preslikavanje i nelinearnu aktivaciju σ :

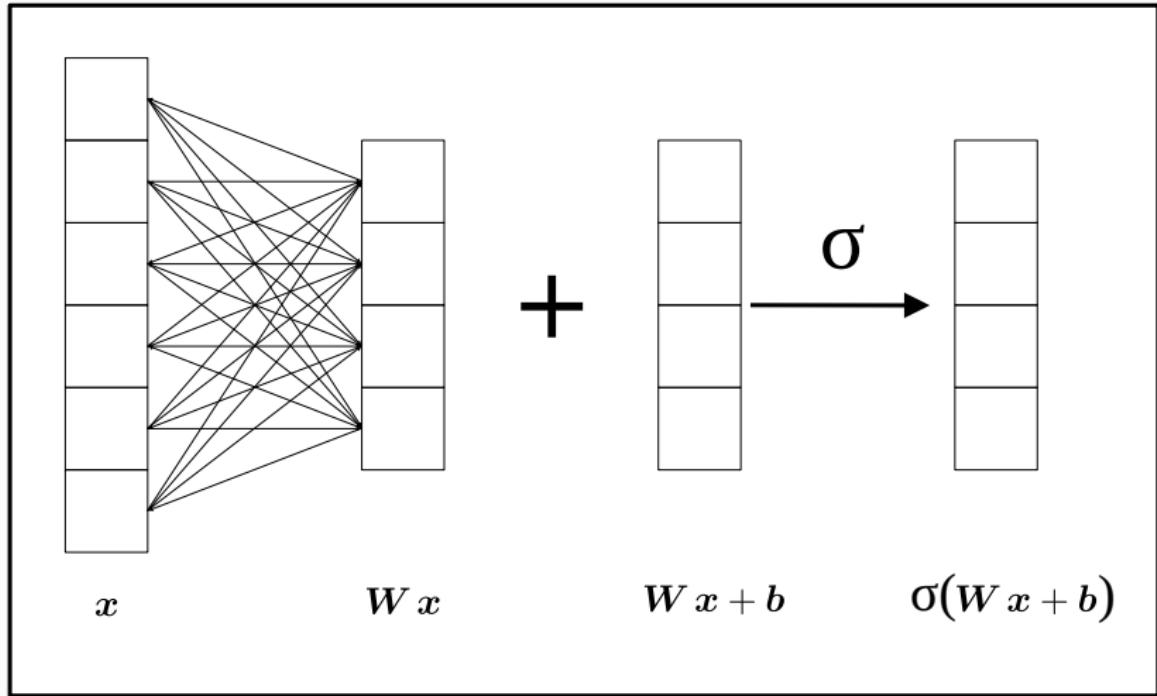
$$f_k(h_{k-1}) = \sigma(W_k h_{k-1} + b_k)$$

- Potpuno povezane slojeve moramo dobro upoznati:
 - osnova za složenije slojeve (npr. konvolucijske)
 - građevne jedinice složenijih arhitektura (npr. pažnja)

Drugi nazivi:

- (unaprijedni, duboki) **potpuno povezani model** (s afinim slojevima) (*eng. fully connected*)
- višeslojni perceptron (*eng. multi-layer perceptron*)
- (unaprijedna) umjetna neuronska mreža

Potpuno povezani sloj



$$f(x; W, b) = \sigma(W \cdot x + b)$$

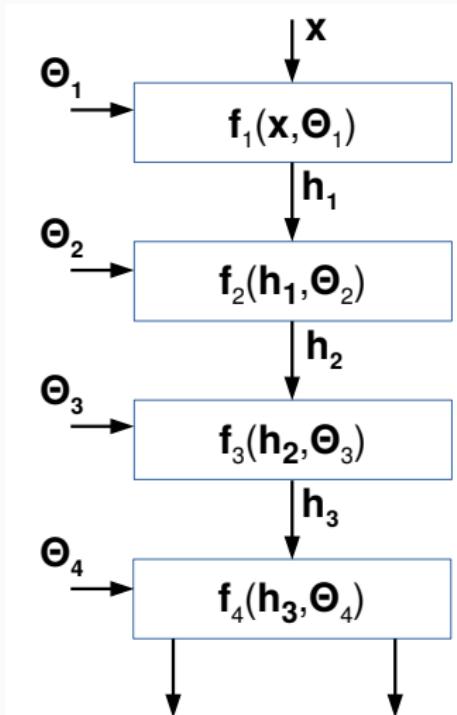
$$\sigma(s)_i = \sigma(s_i)$$

Potpuno povezani model

Zadatak: odrediti strukturu, jednadžbe i ukupan broj parametara potpuno povezanog modela za 2D podatke. ako znamo da su dimenzije slojeva: 5, 10, 5, 2.

Potpuno povezani model

Zadatak: odrediti strukturu, jednadžbe i ukupan broj parametara potpuno povezanog modela za 2D podatke. ako znamo da su dimenzije slojeva: 5, 10, 5, 2.



Što su unaprijedni duboki modeli?

Odnos prema umjetnim neuronskim mrežama:

- umjetne neuronske mreže proučavaju algoritme strojnog učenja koji su inspirirani ranim modelima ljudskog mozga
- s druge strane, duboko učenje proučava modele koji dobro generaliziraju na stvarnim podatcima

Linearni i nelinearni modeli

Pitanje: koliko dubok treba biti potpuno povezani model?

Zavodljiva ideja: $L=1$!

$$f(x, \Theta = (w, b)) = \sigma(w^\top x + b)$$

- **prednost:** uz uobičajene funkcije gubitka vodi na konveksnu optimizaciju
- **prednost:** garantirana konvergencija
- **nedostatak:** naš svijet nije linearan.

Linearni i nelinearni modeli

Ako jednoslojni model nije opcija, što nam preostaje?

- **Rješenje:** originalne značajke x mapirati nelinearnom funkcijom $\Phi(x)$ u drugi prostor u kojem klasifikaciju obavljamo linearnim modelom

$$f(x, \Phi, \Theta = (w, b)) = \Phi(x)^\top w + b$$

- Tri su dominantna načina konstrukcije funkcije $\Phi(x)$:
 - upotrijebiti generičku funkciju $\Phi(x)$,
 - ručno dizajnirati funkciju $\Phi(x)$,
 - naučiti funkciju $\Phi(x|\Theta_\Phi)$.

Generička funkcija za mapiranje značajki

Primjer: jezgrene funkcije.

- npr. RBF funkcija $k(x, \cdot)$ implicitno preslikava ulaz u točku $\Phi(x)$ beskonačno-dimenzionalnog prostora.

Problem: takve funkcije prepostavljaju lokalnu glatkoću (*eng. local smoothness prior*)

- nažalost, lokalna glatkoća nije dovoljno dobra kad je $\dim(x)=10^5$

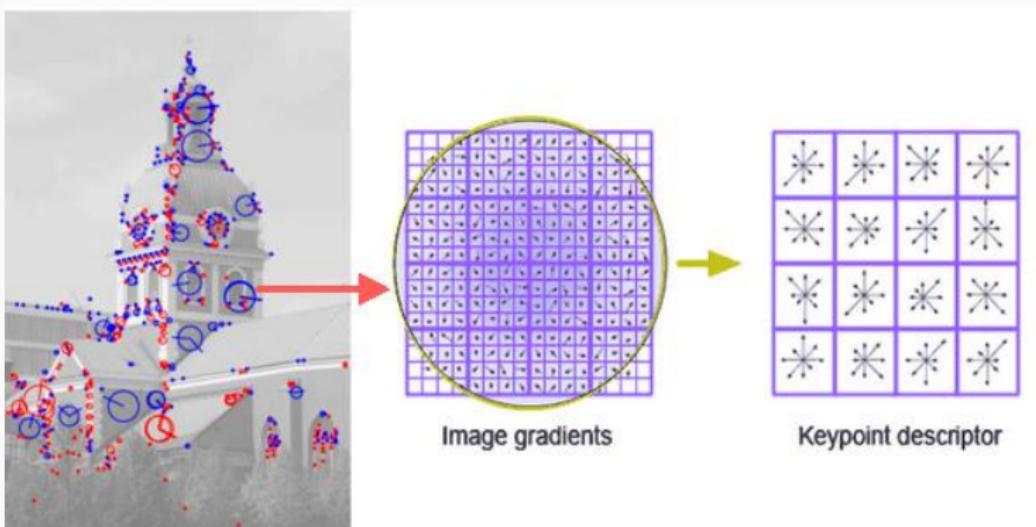


Ručni dizajn značajki

Primjer: SIFT deskriptor u računalnom vidu, vreće riječi u obradi prirodnog jezika, MFCC deskriptori u obradi govora.

Problem: zahtjeva domensko znanje, dugotrajan proces

Problem: (danas znamo) ograničeni rezultati



Jedino preostalo: naučiti funkciju $\Phi(x|\Theta_\Phi)$, s parametrima Θ_Φ

Možemo probati slojeve učiti odvojeno: prvo značajke Θ_Φ (npr. nенадзирano), a tek onda klasifikator w, b

- to bi radilo bolje od linearog modela
- ali nije dobro skaliralo u stvarnim eksperimentima s više od dva sloja

Ostaje samo jedna opcija: učiti duboki model [s kraja na kraj](#):

- spregnuto učenje $\Theta = (w, b) \cup \Theta_\Phi$

Učenje dubokog modela s kraja na kraj

Prednosti u odnosu na generičke funkcije i ručni dizajn:

- zadajemo *klasu* funkcija $\Phi(x|\Theta_\Phi)$ umjesto specifične funkcije $\Phi(x)$
- klasa funkcija je određena strukturom modela.
- možemo imati proizvoljno mnogo slojeva (ili skoro)

Nedostatak u odnosu na generičke funkcije i ručni dizajn:

- optimizacijski problem više nije konveksan
- nema garancije za globalnu konvergenciju učenja

Učenje dubokog modela s kraja na kraj

U praksi se međutim pokazuje da nekonveksni gubitak nije problem u visokodimenzionalnom prostoru

Duboki modeli najprikladniji za podatke koji su generirani kombinacijom faktora, npr. lice se sastoji od usta, očiju, nosa...

Ako takvi faktori postoje, njihova nezavisna obrada može osigurati efikasnu reprezentaciju regija ulaznog prostora.

Duboki modeli mogu biti eksponencijalno učinkovitiji [delalleau11nips] od modela koji prepostavljaju da se predikcija glatko mijenja u okolini podataka za učenje:

- plitki modeli, prototipovi (k-NN), jezgrene funkcije

Učenje dubokog modela s kraja na kraj

Primjer funkcije koja točke ravnine preslikava u RGB boju

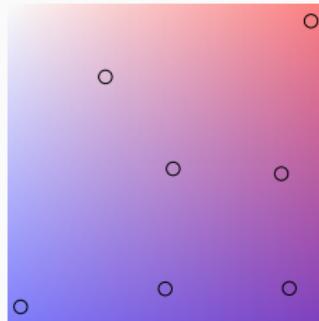
- zadan je sljedeći skup za učenje



Učenje dubokog modela s kraja na kraj

Primjer funkcije koja točke ravnine preslikava u RGB boju

- zadan je sljedeći skup za učenje
- intuitivno je jasno da generalizaciju nije lako postići

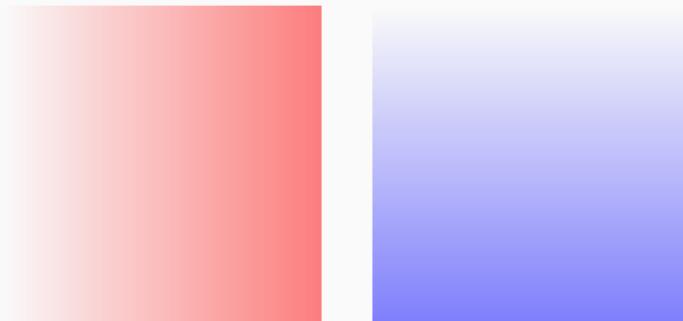


Učenje dubokog modela s kraja na kraj

Primjer funkcije koja točke ravnine preslikava u RGB boju

- zadan je sljedeći skup za učenje
- intuitivno je jasno da generalizaciju nije lako postići
- međutim, problem postaje lakši ako model izrazimo aditivnom kombinacijom dvaju nezavisnih 1D modela

$$f(x, y) = (f_R(x) + f_B(y))/2$$



Primjer: učenje funkcije XOR

- Promotrimo sljedeću funkciju dvije binarne varijable:
$$f^*(x) = (x_0 \wedge \overline{x_1}) \vee (\overline{x_0} \wedge x_1).$$
- pokušajmo naučiti linearni model koji aproksimira f^* :
$$f(x, \Theta = (w, b)) = w^\top x + b$$
- želimo naći takve $\Theta^* = (w^*, b^*)$ da kvadratna greška predikcije bude minimalna:

$$J_{\text{MSE}}(Y, f(X, \Theta)) = \frac{1}{4} \sum_{i=1}^4 (f(x_i, \Theta) - y_i)^2$$

- Vidjet ćemo kasnije da takva funkcija gubitka nije osobito dobar izbor za klasifikacijske probleme, ali ovdje je pogodna jer rješenje možemo dobiti eksplizitno.

Primjer: učenje funkcije XOR

Označimo:

$$w' = [w_1, w_2, b]^\top, \quad X' = \begin{bmatrix} x_{11} = 0, x_{12} = 0, 1 \\ x_{21} = 0, x_{22} = 1, 1 \\ x_{31} = 1, x_{32} = 0, 1 \\ x_{41} = 1, x_{42} = 1, 1 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 = 0 \\ y_2 = 1 \\ y_3 = 1 \\ y_4 = 0 \end{bmatrix}$$

Izrazimo gubitak u pogodnom obliku (za pravilo ulančavanja):

$$J_{\text{MSE}}(y, X', w') = \frac{1}{N} \|X'w' - y\|_2^2 = \frac{\mathbf{q}^\top \mathbf{q}}{N}, \quad \mathbf{q} = X'w' - y$$

Sada možemo izračunati gradijent:

$$\begin{aligned} \nabla_{w'} J_{\text{MSE}}(y, X', w')^\top &= \frac{\partial J}{\partial w} = \frac{\partial J}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial w} \\ &= \frac{2 \cdot \mathbf{q}^\top}{N} \cdot X' = \frac{2}{N} (X'w' - y)^\top X' \end{aligned}$$

Primjer: učenje funkcije XOR

Tražimo minimum funkcije J:

$$\nabla_{w'} J = 0 \rightarrow w' = (X'^\top X')^{-1} X'^\top y$$

Rješenje: $w^* = 0, b^* = 0.5$ (??!)

Zaključak: linearni model ne može riješiti XOR problem.

- Minski and Papert objavili su ovaj zaključak u knjizi: Perceptrons: An Introduction to Computational Geometry
- ovo se smatralo ograničenjem svih pristupa učenju i pridonijelo je prvoj zimi umjetne inteligencije (1974.-1980.)
- algoritam backprop izumio je Seppo Linnainmaa u okviru svog magisterija (1970)

Primjer: učenje funkcije XOR

- Uvedimo dodatni **nelinearni** sloj: mora biti nelinearan, inače bi cijeli model bio (opet) linearan.
- Uobičajeno se danas koristi zglobnica kao nelinearnost: $g(x) = \text{ReLU}(x) = \max(0, x)$.
- Nelinearnost djeluje na svaki element vektora odvojeno: $g(\mathbf{x})_i = g(x_i)$
- Sada možemo postaviti nelinearni model:

$$f(\mathbf{x}, \Theta) = \mathbf{w}_2^\top \mathbf{h} + b_2,$$

$$\mathbf{h} = g(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}_1)$$

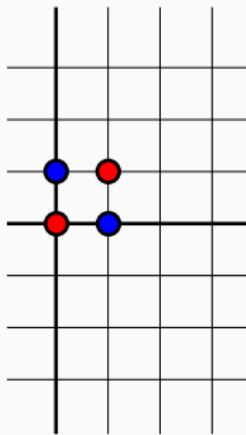
- \mathbf{h} : vektor (naučenih) značajki u skrivenom sloju
- $\mathbf{W}_1, \mathbf{b}_1$: naučeni parametri za računanje značajki iz podataka

Primjer: učenje funkcije XOR

Rješenje : $W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, W_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, b_2 = 0.$

$$f(x_0, x_1) = 1 \cdot \max(x_0 + x_1, 0) - 2 \cdot \max(x_0 + x_1 - 1, 0) + 0$$

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

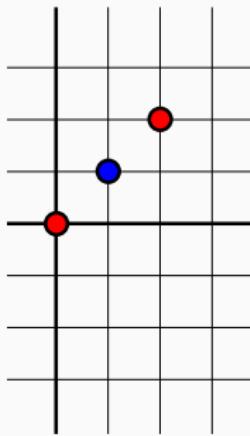


Primjer: učenje funkcije XOR

Rješenje : $W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, W_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, b_2 = 0.$

$$f(x_0, x_1) = 1 \cdot \max(x_0 + x_1, 0) - 2 \cdot \max(x_0 + x_1 - 1, 0) + 0$$

$$W_1 X = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

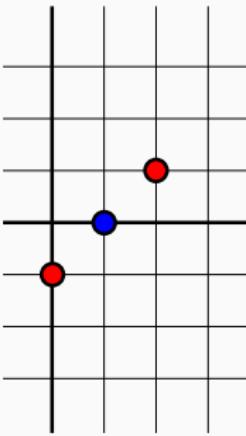


Primjer: učenje funkcije XOR

Rješenje : $W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, W_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, b_2 = 0.$

$$f(x_0, x_1) = 1 \cdot \max(x_0 + x_1, 0) - 2 \cdot \max(x_0 + x_1 - 1, 0) + 0$$

$$W_1 X + b_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

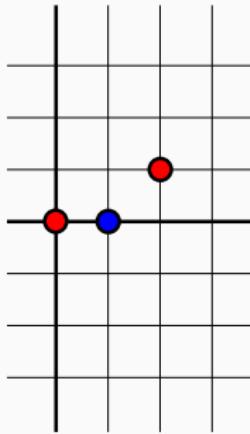


Primjer: učenje funkcije XOR

Rješenje : $W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, W_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, b_2 = 0.$

$$f(x_0, x_1) = 1 \cdot \max(x_0 + x_1, 0) - 2 \cdot \max(x_0 + x_1 - 1, 0) + 0$$

$$\text{ReLU}(W_1 X + b_1) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$



Primjer: učenje funkcije XOR

Pokazali smo konačno rješenje, ali nismo pokazali kako se do njega dolazi.

U pravilu se za određivanje parametara dubokih modela koriste postupci optimizacije temeljeni na gradijentu funkcije gubitka.

Prikazano rješenje je globalni minimum funkcije gubitka,

- u generalnom slučaju gradijentni spust dovodi u lokalni minimum, ako se radi o ne-konveksnom gubitku
- u tom slučaju rješenje ovisi o inicijalizaciji.

- Što su unaprijedni duboki modeli?
- **Funkcija gubitka i izlazni slojevi.**
- Aktivacijske funkcije u skrivenim slojevima.
- Univerzalna aproksimacija: dubina je važna.
- Backprop: izračun gradijenta kompozicije funkcije prosljeđivanjem greške unazad.

Učenje parametara modela – minimizacija empirijskog rizika

- Učenje modela: pronalaženje parametara Θ^* za koje je empirijski rizik minimalan:

$$J(X, Y, \Theta) = \frac{1}{N} \sum_{i=1}^N \ell(y_i, f(x_i, \Theta)) + \lambda \Omega(\Theta)$$

$$\Theta^* = \arg \min_{\Theta} J(X, Y, \Theta)$$

- Funkcija gubitka $\ell(y, \hat{y})$ odražava naše “razočaranje” razlikom između predikcije modela \hat{y} i stvarne vrijednosti y .
- Regularizator $\Omega(\Theta)$ kažnjava neke vrijednosti parametara modela.

0-1 gubitak

$$\ell_{01}(y, \hat{y}) = \begin{cases} 0, & \text{ako } y = \hat{y} \\ 1, & \text{inače} \end{cases}$$

- Ovaj gubitak nije derivabilan, pa je minimizacija $J(\mathbf{X}, \mathbf{Y}, \boldsymbol{\Theta})$ teška: kombinatorna optimizacija

Funkcija gubitka

Pokušajmo ne biti isključivi, oblikujmo model koji prediktira distribuciju preko mogućih izlaza $P(\hat{y}|x; \Theta)$

Definirajmo gubitak kao negativnu log-izglednost:

$$\ell_{\text{MLE}}(Y, \hat{Y}) = - \sum \log P(\hat{y} = y|x; \Theta)$$

- za regresiju prepostavljamo normalnu (Gaussovu) razdiobu, s jediničnom kovarijacijskom matricom:

$$P(\hat{y} = y|x; \Theta) = \mathcal{N}(y|\mu = f(x, \Theta), \Sigma = I)$$

- za klasifikaciju prepostavljamo kategoričku razdiobu (ili “generaliziranu Bernoullijevu razdiobu”):

$$P(\hat{y} = y|x; \Theta) = f_y(x, \Theta)$$

Negativna log-izglednost kao funkcija gubitka (2)

Negativna log izglednost kategoričke razdiobe odgovara unakrsnoj entropiji između distribucije stvarnih oznaka y i distribucije izlaza našeg modela $P(\hat{y}|x; \Theta)$.

Možemo formulirati i determinističko predviđanje uključivanjem Diracove δ -distribucije (ovo dovodi do gubitka 0 - 1)

Negativna log-izglednost kao funkcija gubitka

- Prednost prikaza funkcije gubitka preko negativne log-izglednosti je **općenitost**: nema potrebe dizajnirati funkcije gubitka specifične za svaki pojedinačni model.
 - definiramo distribuciju preko izlaza: $P(\hat{y}|x; \Theta)$
 - funkcija gubitka je: $\ell_{\text{MLE}}(y, \hat{y}) = -\log P(\hat{y} = y|x; \Theta)$
- Za regresiju dobivamo srednju kvadratnu pogrešku:

$$\ell_{\text{MSE}}(y, \hat{y}) = (y - \hat{y})^2$$

- Za klasifikaciju dobivamo:

$$\ell_{\text{CE}}(y, \hat{y}) = -\log f_y(x, \Theta)$$

- Obje ove funkcije gubitka su derivabilne.
⇒ mogu se učiti gradijentnim spustom

Klasifikacija: soft-max

Klasifikator mora vratiti kategoričku razdiobu preko C razreda:

- $f_i(x, \Theta) \in [0, 1] \quad \forall i, \sum_i^C f_i(x, \Theta) = 1$

Promotrimo značajke u zadnjem sloju modela:

$$z = h^L = W_L^\top h^{L-1} + b_L$$

Tada distribuciju možemo dobiti funkcijom softmax:

$$f_i(x, \Theta) = P(\hat{y} = i | x; \Theta) = \text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- eksponenciranje garantira $f_i(x, \Theta) > 0 \quad \forall i$
- nazivnik garantira: $\sum_i^C f_i(x, \Theta) = 1$

Klasifikacija: softmax (objašnjenje)

Reinterpretirajmo logite \mathbf{z} kao logaritmiranu nenormaliziranu združenu gustoću podataka i razreda:

$$z_y = \log \text{const} \cdot P(\hat{y} = y, \mathbf{x}; \Theta) .$$

Odatle lako slijedi da softmaks odgovara aposteriornoj vjerojatnosti [grathwohl20iclr]:

$$\sum_k e^{z_k} = P(\mathbf{x}; \Theta) \quad (\text{marginalizacija preko } \hat{y}) ,$$

$$\begin{aligned} \text{softmax}(\mathbf{z})_y &= \frac{e^{z_j}}{\sum_k e^{z_k}} \\ &= \frac{\text{const} \cdot P(\hat{y} = y, \mathbf{x}; \Theta)}{\text{const} \cdot P(\mathbf{x}; \Theta)} \\ &= P(\hat{y} = y | \mathbf{x}; \Theta) \quad (\text{Bayesov poučak}) . \end{aligned}$$

Klasifikacija: stabilnost log-softmaka

Izrazimo negativnu log-izglednost kao funkciju logita $z = f_\theta(x)$ i indeksa točnog razreda y :

$$\ell_{\text{NLL}}(z, y) = -\log \text{softmax}_y(z)$$

- problem: ako softmaks **podlije** (ode u nulu), gubitak $\rightarrow \infty$
- kada se to može dogoditi?

Stabilnija formulacija log-softmaka:

$$\begin{aligned}\ell_{\text{NLL}}(z, y) &= -\log \text{softmax}_y(z) \\ &= -\log \frac{e^{z_y}}{\sum_k e^{z_k}} = \log \sum_k e^{z_k} - z_y\end{aligned}$$

- ova izvedba osjetljiva je na **preljev** softmaka
- kako to možemo izbjegići?

Klasifikacija: soft-max kao izlazni sloj

Stabilna negativna log izglednost (y : indeks točnog razreda):

$$\ell_{\text{NLL}}(\mathbf{z}, y) = -\log \text{softmax}_y(\mathbf{z}) = \log \sum_k e^{z_k} - z_y$$

Možemo izvesti sljedeće intuitivne zaključke:

- Kada model točno predviđa ($\max_j z_j = z_y$) gubitak je ≈ 0 .
- Kada model griješi, tj. kada je $\max_j z_j \neq z_y$, na gubitak najviše utječe najaktivnija (najjača) netočna predikcija.
- Takvo ponašanje je vrlo slično 0-1 gubitku: negativna log-izglednost je gornja ograda 0-1 gubitka.

Za domaći rad dokazati:

$$\frac{d\ell_{\text{CE}}(\hat{y}, \text{softmax}(\mathbf{z}))}{dz_k} = \text{softmax}(\mathbf{z})_k - \llbracket \hat{y} = k \rrbracket$$

Svojstva softmaka

Invarijantnost na dodavanje konstante:

$$\text{softmax}(z) = \text{softmax}(z + c) = \text{softmax}\left(z - \max_j z_j\right)$$

Bolje ime (koje se nažalost nije uvriježilo): **softargmax**

"Pravi softmax" izračunali bismo kao log-sum-exp:

$$\text{LSE}(\mathbf{z}) = \log \sum_i e^{z_i} = \max(\mathbf{z}) + \log \sum_i e^{z_i - \max(\mathbf{z})}$$

Parametrizacije softmaxa

Iako je izlaz softmaxa C -dimenzionalan, postoji samo $C - 1$ stupnjeva slobode (izlaz je distribucija)

To znači da jedan od ulaza možemo fiksirati (npr. postaviti na 0) bez smanjenja općenitosti modela.

U praksi nema razlike između te dvije varijante, a implementacija je jednostavnija kada je ulaz C dimenzionalan.

Binarna klasifikacija: sigmoida kao izlazni sloj

Ako je $C = 2$ onda:

$$\begin{aligned} P(\hat{y} = 1|x) &= \text{softmax}(z)_1 = \frac{\exp(z_1)}{\exp(z_0) + \exp(z_1)} \\ &= \frac{1}{1 + \exp(z_0 - z_1)} \end{aligned}$$

Ako postavimo $z_0 = 0$ dobivamo:

$$P(\hat{y} = 1|x) = \sigma(z_1)$$

- ⇒ soft-max je poopćenje sigmoide na slučaj $C > 2$ klase
- ⇒ kategorička razdioba je poopćenje Bernoullijeve razdiobe na slučaj $C > 2$ ishoda.

Za domaći rad dokazati: $\frac{d\ell_{\text{CE}}(y, \sigma(z))}{dz} = \sigma(z) - y$

Srednja kvadratna pogreška za klasifikacijski gubitak?

Zašto negativna (log-)izglednost kao funkcija gubitka za klasifikaciju? Zašto ne srednja kvadratna pogreška?

$$\ell_{\text{MSE}}(y, \sigma(z)) = (y - \sigma(z))^2$$

Pogledajmo gradijent funkcije gubitka s obzirom na značajke u zadnjem sloju:

$$\frac{\partial \ell_{\text{MSE}}(y, \sigma(z))}{\partial z} = 2(\sigma(z) - y)(1 - \sigma(z))\sigma(z)$$

Kada je sigmoida u zasićenju ($z \gg 0$ ili $z \ll 0$) gradijent funkcije gubitka je malen, bez obzira da li je $\sigma(z)$ blizu y ili ne: model ne može naučiti iz takvog primjera.

Srednja kvadratna pogreška za klasifikacijski gubitak?

Glavni nedostatak MSE-a jest ignoriranje strukture vjerojatnosne distribucije; to se najbolje vidi u višerazrednom slučaju

Npr. pretpostavimo da imamo podatke \mathbf{x}_1 i \mathbf{x}_2 koji bi se trebali klasificirati u drugi razred:

$$\mathbf{Y}_1^{OH} = \mathbf{Y}_2^{OH} = [0, 0, 1]$$

Pretpostavimo dalje da za njih dobivamo sljedeće distribucije:

$$P(\mathbf{Y}|\mathbf{x}_1) = [0.8, 0, 0.2], P(\mathbf{Y}|\mathbf{x}_2) = [0.4, 0.4, 0.2]$$

Gubitci su različiti iako su predikcije jednako pogrešne:

$$L_{MSE}(\mathbf{x}_1, \mathbf{Y}_1^{OH}) = 1.28, L_{MSE}(\mathbf{x}_2, \mathbf{Y}_2^{OH}) = 0.96$$

Za klasifikaciju, MLE je bolji gubitak od MSE.

- Što su unaprijedni duboki modeli?
- Funkcija gubitka i izlazni slojevi.
- Aktivacijske funkcije u skrivenim slojevima.
- Univerzalna aproksimacija: dubina je važna.
- Backprop: izračun gradijenta kompozicije funkcije prosljeđivanjem greške unazad.

Primjer: učenje funkcije XOR

Uveli smo bili dodatni **nelinearni** sloj: mora biti nelinearan, inače bi cijeli model bio (opet) linearan.

Uobičajeno se danas koristi zblobnica kao nelinearnost:
 $g(x) = \text{ReLU}(x) = \max(0, x)$.

Nelinearnost djeluje na svaki element vektora odvojeno:
 $g(\mathbf{x})_i = g(x_i)$

Nelinearnost se zove **aktivacijska funkcija**.

Zglobnica kao aktivacijska funkcija

Zglobnica (*eng. rectified linear unit*):

$$g(x) = \text{ReLU}(x) = \max(0, x).$$

Prednosti:

- u aktivnom stanju propušta signal unaprijed i gradijent unatrag
- podržava propagiranje gradijente s obzirom na ulazne i izlazne aktivacije

"Nedostatak" 1: gradijent funkcije nije definiran za $x = 0$

- u implementaciji definiramo gradijent da bude jednak ili gradijentu s lijeva (0) ili gradijentu s desna (1).

Zglobnica kao aktivacijska funkcija

Nedostatak 2: u neaktivnom stanju ne propušta signal unaprijed i gradijent unatrag, ali:

- postoje bijektivne generalizacije zglobnice:
 - Leaky ReLU: $g(x, \alpha) = \max(0, x) + \alpha \min(0, x)$.
 - Soft Plus: $g(x) = \log(1 + e^x)$.
- normalizacija po grupi (*eng. batch normalization*) implicira srednju vrijednost 0 i varijancu 1
 - ⇒ u svakoj iteraciji učenja i u svakoj značajki imamo 50% aktivnih aktivacija

Nedostatak 3: srednja vrijednost aktivacija nije 0

- situaciju ponovo spašava normalizacija po grupi

Druge korištene aktivacijske funkcije

Sigmoida $\sigma(x) = (1 + \exp(-x))^{-1}$

- guši gradijent kada uđe u zasićenje (učenje prestaje, nestajući gradijent, eng. vanishing gradient),
- nema ih više u unaprijednim modelima (ima iznimki)

Tangens hiperbolni $\tanh(x) = \frac{\exp(2x)-1}{\exp(2x)+1}$ obično se ponaša bolje od sigmoide budući da sliči identitetu u dijelu oko $x = 0$, to osigurava jednostavan transfer gradijenata unatrag

- veza između \tanh i σ : $\tanh(x) = 2\sigma(2x) - 1$.

Eksp.-linearna funkcija (ELU): $f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1); & \text{for } x < 0 \\ x; & \text{for } x \geq 0 \end{cases}$

Generalno: bilo koja nelinearna funkcija je u redu

- Što su unaprijedni duboki modeli?
- Funkcija gubitka i izlazni slojevi.
- Aktivacijske funkcije u skrivenim slojevima.
- **Univerzalna aproksimacija: dubina je važna.**
- Backprop: izračun gradijenta kompozicije funkcije prosljeđivanjem greške unazad.

Teorem o univerzalnoj aproksimaciji

Teorem: model s linearним излазним слојем и најмање једним скривеним слојем са неполиномном активацијском функцијом може апроксимирати било коју Borel мјерљиву функцију из једног коначно-димензионалног простора у други са произволно маленом грешком (већом од 0), ако модел има довољно активација у скривеном слоју (или скривеним слојевима).

Sвака непрекинута функција дефинирана на затвореном и ограниченој подскупу \mathbb{R}^n је Borel мјерљива функција

Не требамо прilagođavati aktivacijske funkcije: довољно је да имамо један скривени слој

Teorem o univerzalnoj aproksimaciji (caveat 1)

Teorem o reprezentacijskom kapacitetu dubokih modela: kad bi funkcija f^* bila poznata onda bi je mogli aproksimirati proizvoljno dobro.

Međutim, funkcija nije poznata, dostupni su samo podaci za učenje $(\mathcal{X}, \mathcal{Y})$.

Ovaj teorem ne govori ništa o tome da li f^* možemo naučiti iz $(\mathcal{X}, \mathcal{Y})$.

Teorem samo kaže da dovoljno "nabildani" model može naštrebati skup za učenje

Teorem o univerzalnoj aproksimaciji (caveat 2)

Teorem ne specificira koliko aktivacija trebamo za postići zadanu grešku aproksimacije, ali postoji gornja ograda (eng. *upper bound*)

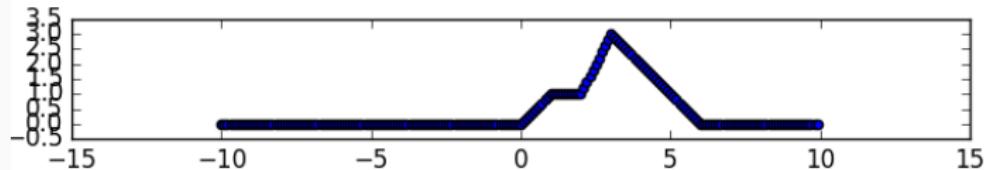
- u najgorem slučaju trebamo eksponencijalno veliki broj značajki u srednjem sloju:

$$\dim(\mathbf{h}) \sim O(a^{\dim(x)})$$

- svaka od tih značajki odgovara konfiguraciji ulazâ koja se mapira u izlaz koji moramo razlikovati od drugih izlaza.
 - za naučiti logičku funkciju n varijabli trebamo izračunati $O(2^n)$ minterma

Teorem o univerzalnoj aproksimaciji

Zadatak: konstruirati dvorazinski afini model aktiviran zglobnicom za aproksimaciju sljedeće funkcije $\mathbb{R} \rightarrow \mathbb{R}$:



Rješenje:

```
h10 = np.maximum(X-0, 0)
h11 = np.maximum(X-1, 0)
h12 = np.maximum(X-2, 0)
h13 = np.maximum(X-3, 0)
h14 = np.maximum(X-6, 0)
h21 = 1*h10 - 1*h11 + 2*h12 - 3*h13 + 1*h14
```

Zašto onda uopće duboki modeli? Efikasnost prikaza funkcije.

Duboki modeli mogu smanjiti potreban broj aktivacija u skrivenim slojevima za prikaz funkcije

- postoje funkcije koje se mogu efikasno predstaviti dubokim modelima.

Efikasnost dubokih modela u odnosu na plitke (samo jedan skriveni sloj) može biti čak eksponencijalna u broju potrebnih aktivacija

Zašto onda uopće duboki modeli? Efikasnost prikaza funkcije.

Modeli koje koriste zglobnicu kao aktivacijsku funkciju definiraju po dijelovima linearne funkcije nad regijama ulaznog prostora

- broj tih regija je mjera fleksibilnosti (kapaciteta) modela.
- duboki modeli imaju eksponencijalno više regija od plitkih modela s istim brojem aktivacija.

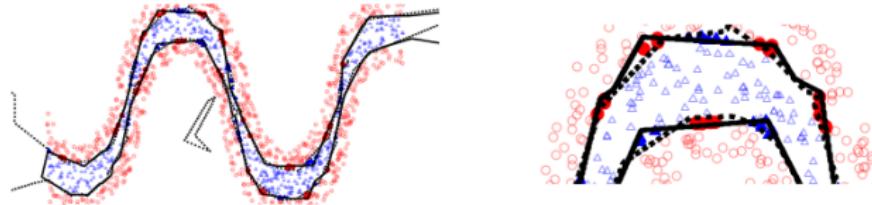
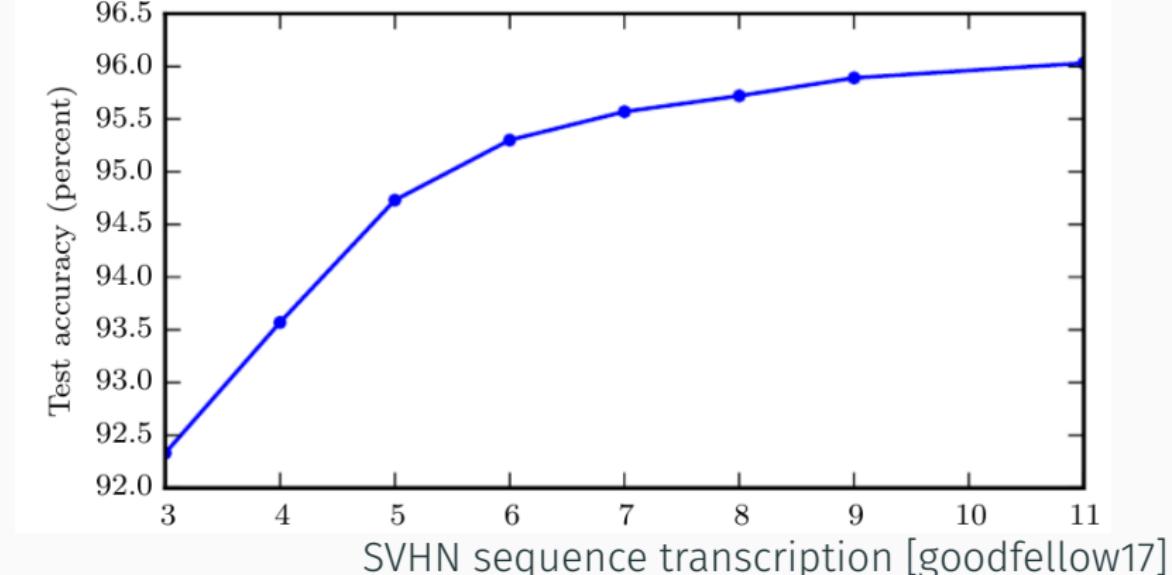


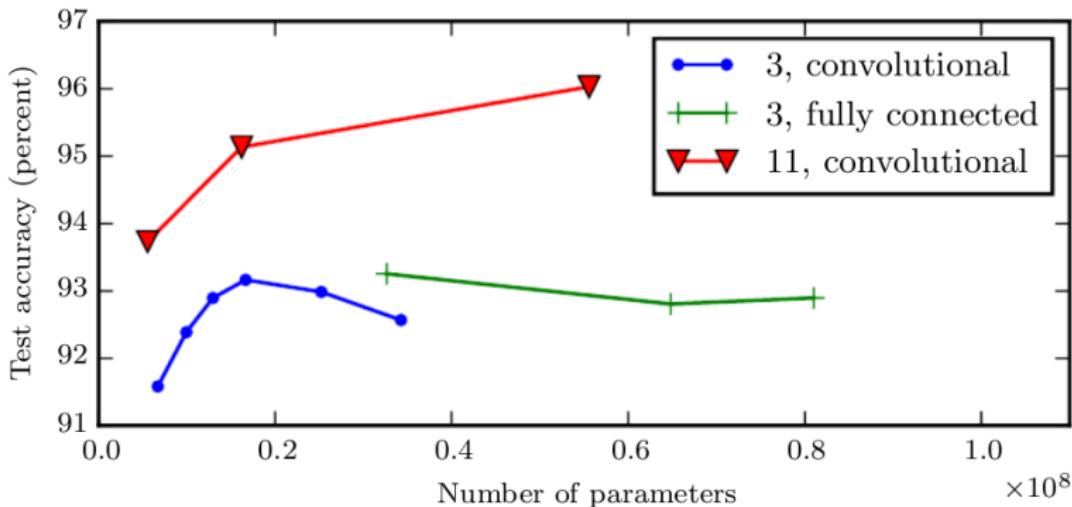
Figure 1: Binary classification using a shallow model with 20 hidden units (solid line) and a deep model with two layers of 10 units each (dashed line). The right panel shows a close-up of the left panel. Filled markers indicate errors made by the shallow model.



Empirijski rezultati pokazuju da modeli za klasifikaciju slika postižu bolje rezultate kad povećamo dubinu

- x: dubina modela, y: točnost klasifikacije

povećanjem širine modela dobivamo značajno manja poboljšanja...



Duboki konvolucijski modeli za klasifikaciju slika generaliziraju bolje od plitkih (SVHN sequence transcription [goodfellow17])

- x: broj parametara, y: točnost klasifikacije

Produbljenje uvodi pristranost koja pogoduje generalizaciji!

- plitki modeli se prenauče već na $2e7$ parametara
- duboki modeli dobro generaliziraju i sa $6e7$ parametara

- Što su unaprijedni duboki modeli?
- Funkcija gubitka i izlazni slojevi.
- Aktivacijske funkcije u skrivenim slojevima.
- Univerzalna aproksimacija: dubina je važna.
- Backprop: izračun gradijenta kompozicije funkcije prosljeđivanjem greške unazad.

Nadzirano učenje modela

Prolaz unaprijed: računanje izlaza modela $\hat{y} = f(x, \Theta)$ i funkcije gubitka $J(y, \hat{y}) = J(y, f(x, \Theta))$

Prolaz unatrag: računanje gradijenta funkcije gubitka s obzirom na parametre modela $\nabla_{\Theta} J(y, f(x, \Theta)) = (\frac{\partial J(y, f(x, \Theta))}{\partial \Theta})^{\top}$

Optimizacijski postupak: tipično varijanta gradijentnog spusta

- $\Theta' = \Theta - \delta \cdot \nabla_{\Theta} J(y, f(x, \Theta))$
- o tome ćemo pričati detaljnije neki drugi put.

Prosljeđivanje greške unatrag (*eng. backprop*) je jednostavan i računski nezahtjevan način računanja gradijenta kompozicije funkcija.

Derivacija kompozicije funkcija ulančavanjem gradijenata

Pravilo ulančavanja (eng. *chain rule*): recept za deriviranje kompozicije funkcija za koje su derivacije poznate

Za skalarne funkcije, npr. $y = g(x)$, $z = f(y) = f(g(x))$, imamo:

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx} = \frac{df(y)}{dy} \frac{dg(x)}{dx}$$

Za vektorske funkcije, $\mathbf{y} = g(\mathbf{x})$, $z = f(\mathbf{y}) = f(g(\mathbf{x}))$, dobivamo:

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \quad \text{ili} \quad \nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^T \nabla_{\mathbf{y}} z.$$

- $\frac{\partial z}{\partial \mathbf{y}}$ i $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ su Jakobijani dimenzija $1 \times n$ odnosno $n \times m$;
- $\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$

Takov umnožak provodimo za svaki računski korak modela.

Backprop: rekurzivna primjena ulančavanja

Promotrimo duboki model f parametriziran s Θ koji podatke x preslikava u predikcije \hat{y} :

$$\hat{y} = f(x, \Theta)$$

Gradijent gubitka s obzirom na parametre l -tog sloja tada je:

$$\begin{aligned}\frac{\partial \mathcal{L}(y, \hat{y})}{\partial \Theta^l} &= \frac{\partial \mathcal{L}(y, \hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h^l} \frac{\partial h^l}{\partial h^{l-1}} \cdots \frac{\partial h^l}{\partial \Theta^l} \\ &= \frac{\partial \mathcal{L}(y, \hat{y})}{\partial h^l} \frac{\partial f^l(h^{l-1}, \Theta^l)}{\partial h^{l-1}} \cdots \frac{\partial f^{l+1}(h^l, \Theta^{l+1})}{\partial h^l} \frac{\partial f^l(h^{l-1}, \Theta^l)}{\partial \Theta^l}\end{aligned}$$

Za svaki sloj trebamo izračunati parcijalnu derivaciju izlaza:

- s obzirom na parametre (ako postoje) $\frac{\partial f^l(h^{l-1}, \Theta^l)}{\partial \Theta^l}$
- s obzirom na ulaz $\frac{\partial f^l(h^{l-1}, \Theta^l)}{\partial h^{l-1}}$ (samo ako nismo gotovi)
- **problem:** Θ^l može biti matrica (potpuno povezani sloj)
- **problem:** h^l i Θ^l mogu biti tenzori 4. reda (konv. sloj)

Gradijenti po tenzorima višeg reda (>1)

Gradijente po tenzorima *mogemo* računati kao i za vektore

- prvo odredimo gradijente za vektorizirani tenzor,
- te ih na kraju presložimo u početni oblik

Prepostavimo: $X \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \cdots \mathbb{R}^{m_M}$, $Y \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \cdots \mathbb{R}^{n_N}$

- Tada je $\frac{\partial \text{vec}(Y)}{\partial \text{vec}(X)}$ Jakobijan dimenzija $(n_1 n_2 \cdots n_N) \times (m_1 m_2 \cdots m_M)$.
- backprop je i u ovom slučaju samo množenje Jakobijskog:

$$\frac{\partial z}{\partial \text{vec}(X)} = \frac{\partial z}{\partial \text{vec}(Y)} \frac{\partial \text{vec}(Y)}{\partial \text{vec}(X)}$$

Backprop za parametre potpuno povezanog sloja

U praksi, gradijente gubitka ipak nećemo računati po vektoriziranim težinama jer postoje efikasniji pristupi.

Razmotrimo potpuno povezani sloj ($\mathbf{h}_{k-1} \in \mathbb{R}^{D_{k-1}}$, $\mathbf{s}_k \in \mathbb{R}^{D_k}$):

$$\mathbf{s}_k = \mathbf{W}_k \cdot \mathbf{h}_{k-1} + \mathbf{b}_k .$$

Gradijente po vektoriziranim parametrima možemo izračunati prema osnovnom receptu u $O(D_k^2 \cdot D_{k-1})$:

$$\frac{\partial \mathcal{L}}{\partial \text{vec}(\mathbf{W}_k)} = \frac{\partial \mathcal{L}}{\partial \mathbf{s}_k} \cdot \frac{\partial \mathbf{s}_k}{\partial \text{vec}(\mathbf{W}_k)} .$$

Backprop za parametre potpuno povezanog sloja

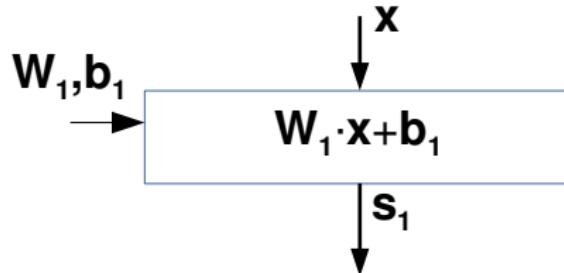
Umjesto toga, taj gradijent radije čemo računati prema receptu iz uputa za laboratorijske vježbe u $O(D_k \cdot D_{k-1})$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_k} := \left(\frac{\partial \mathcal{L}}{\partial w_{kij}} \right)_{D_k \times D_{k-1}} = \left[\frac{\partial \mathcal{L}}{\partial \mathbf{s}_k} \right]^\top \cdot \mathbf{h}_{k-1}^\top$$

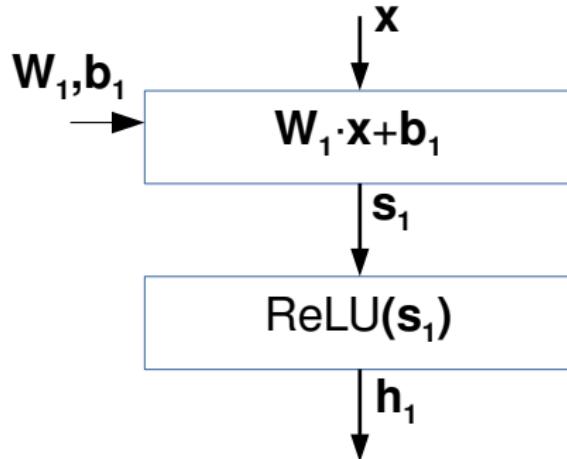
Domaći rad:

- pokazati da prikazani pristupi ekvivalentni (isti rezultat)
- razumjeti kako određujemo njihovu složenost

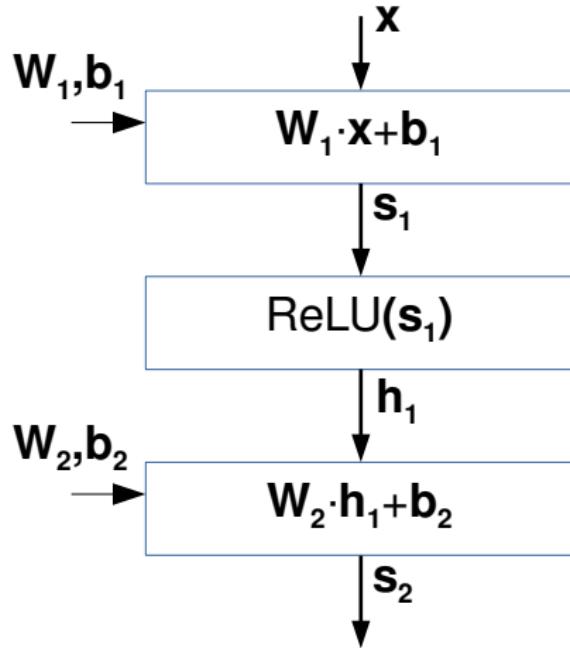
Primjer: unaprijedni prolaz



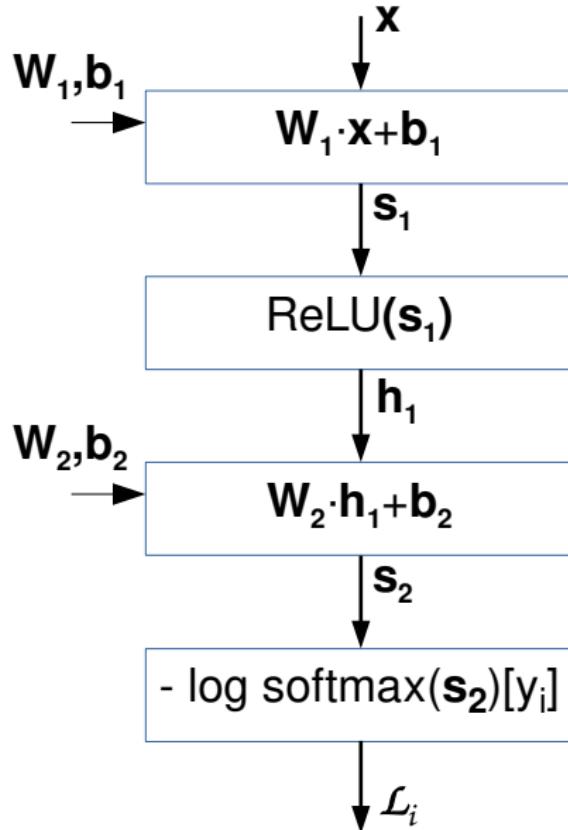
Primjer: unaprijedni prolaz



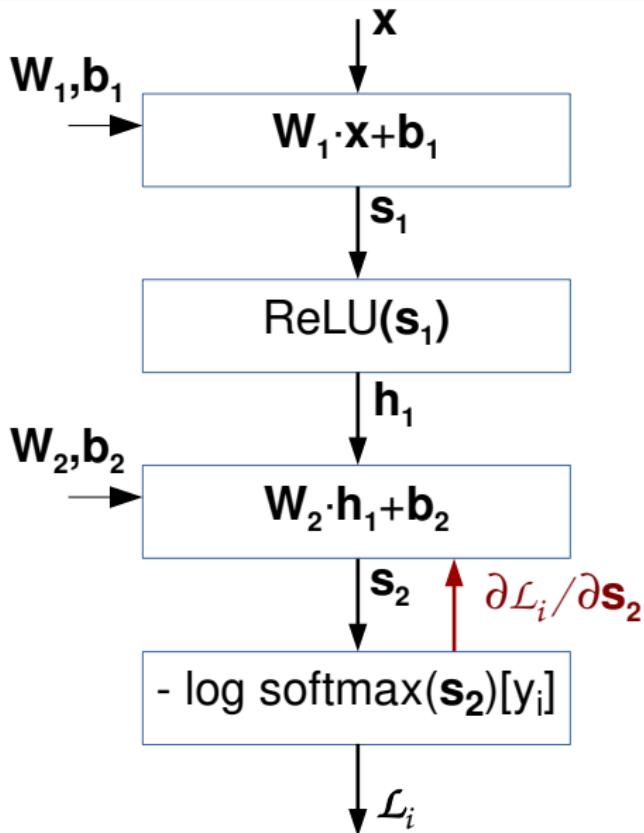
Primjer: unaprijedni prolaz



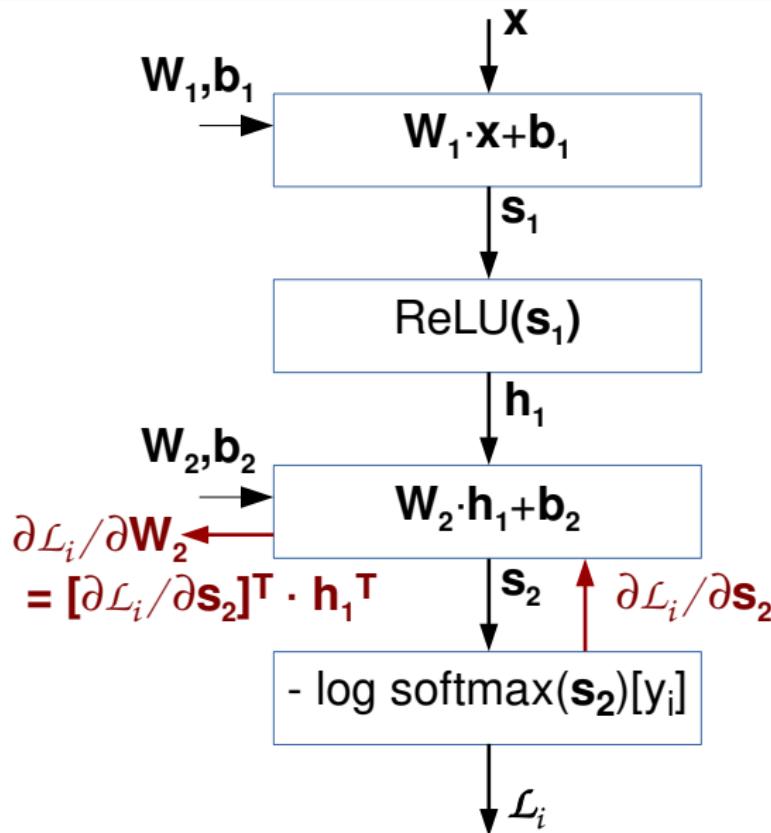
Primjer: unaprijedni prolaz



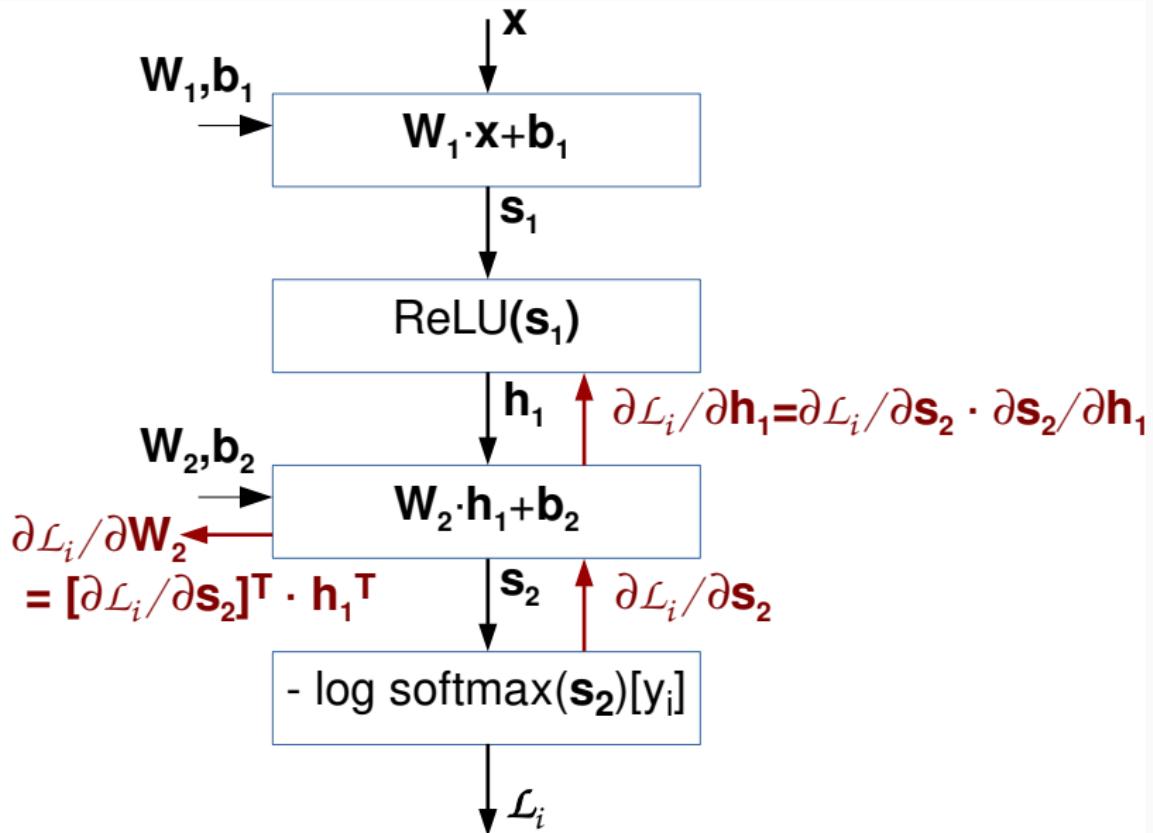
Primjer: unatražni prolaz (=dinamičko programiranje)



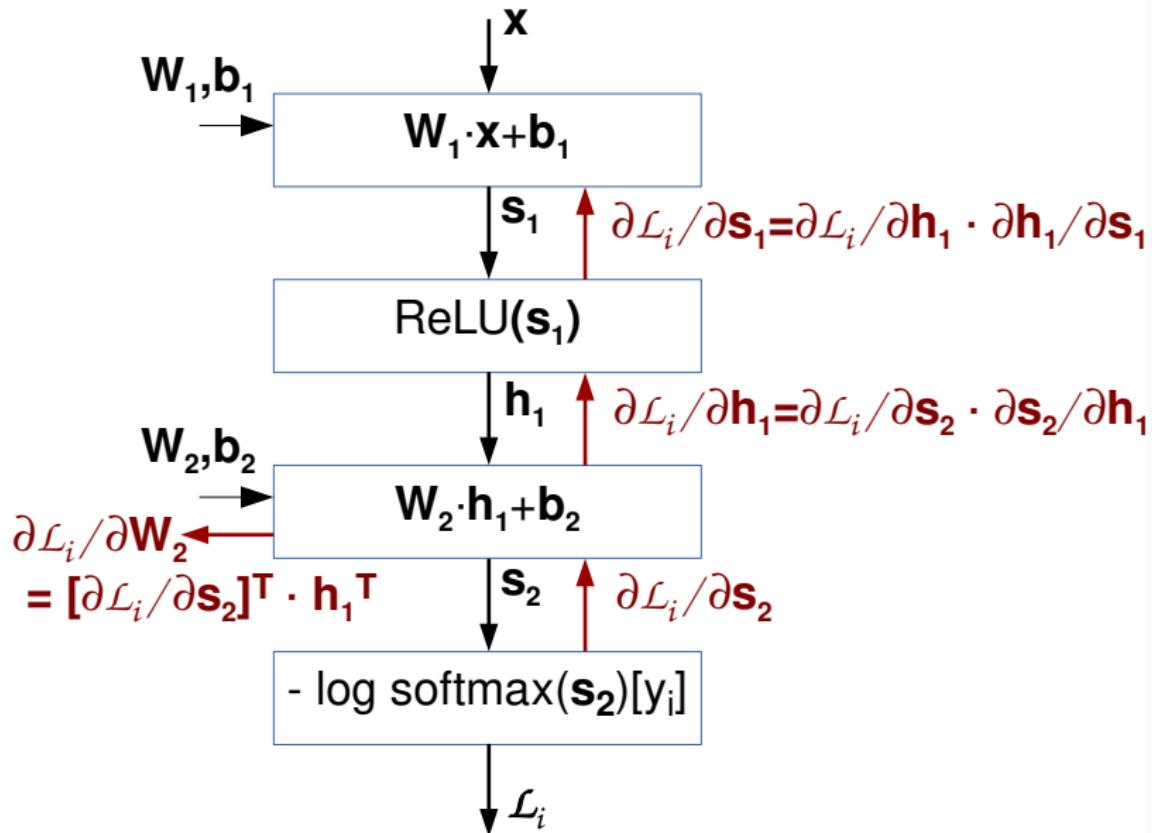
Primjer: unatražni prolaz (=dinamičko programiranje)



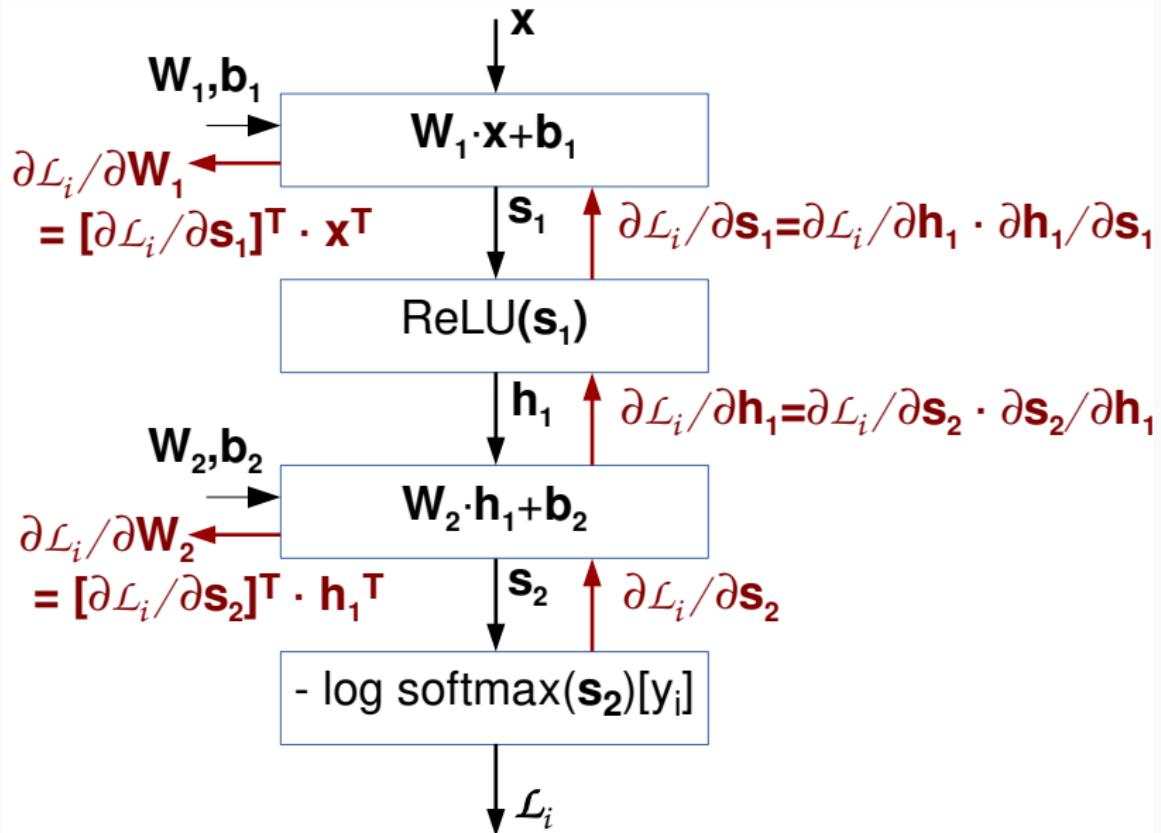
Primjer: unatražni prolaz (=dinamičko programiranje)



Primjer: unatražni prolaz (=dinamičko programiranje)



Primjer: unatražni prolaz (=dinamičko programiranje)



Automatsko računanje gradijenata

Kako bismo proveli automatsko računanje gradijenata, duboki model predstavljamo računskim grafom

Korijeni grafa predstavljaju ulazne podatke, oznake, parametre i hiperparametre

Svi ostali čvorovi predstavljaju operacije: diferencijabilne funkcije jedne ili više varijabli.

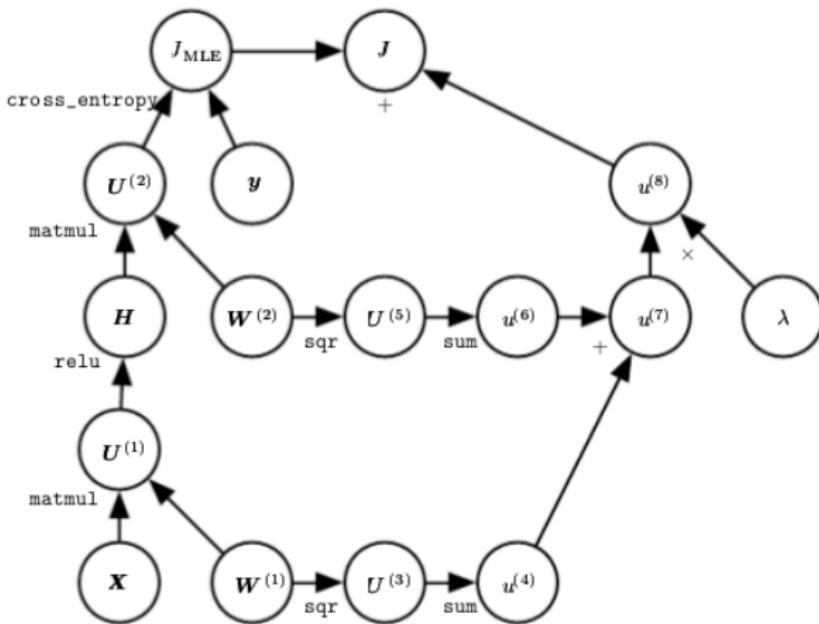
Operacije vraćaju samo jedan izlaz (radi jednostavnosti)

- izlaz može biti skalar, vektor, matrica ili tenzor
- stoga ovo ograničenje ne smanjuje općenitost.

Primjer: klasifikacijski model s dva potpuno povezana sloja i L2 regularizacijom

- radi jednostavnosti izostaviti ćemo pomake **b**

Prikaz dubokog modela računskim grafom



PyTorch hello world

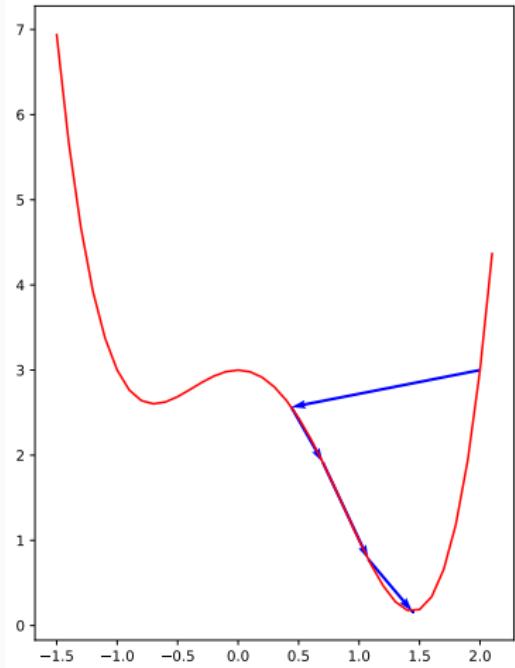
```
import torch

step = 0.13

x = torch.tensor(2.0,
                 requires_grad=True)

for i in range(100):
    y = x**4 - x**3 - 2*x**2 + 3
    y.backward()
    print(x, y, x.grad)

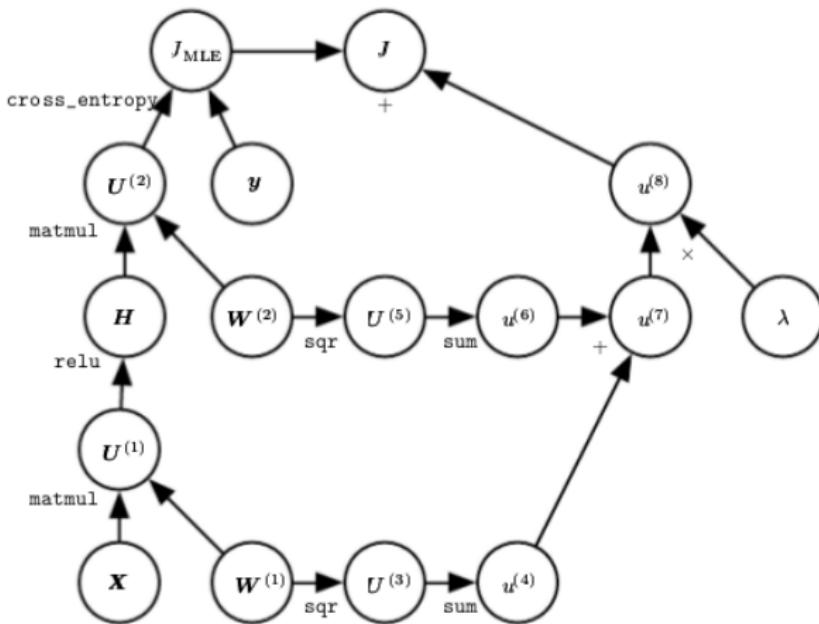
    x.data = x - step * x.grad
    x.grad.zero_()
```



Glavni sastojak: unatražno automatsko differenciranje

Domaći rad: riješiti $x^5 - x^4 - x = -1$ gradijentnim spustom

Vratimo se na naš računski graf



Odgovarajući kôd pod PyTorchem

```
import torch, torch.nn.functional as F

# roots: input, label, parameters, hiperparameter
x = torch.tensor([1.,1.])
y = torch.tensor(0.)
W1 = torch.tensor([[0.5,0], [0,1]], requires_grad=True)
W2 = torch.tensor([1.,0.], requires_grad=True)
lambda1 = torch.tensor(0.01)

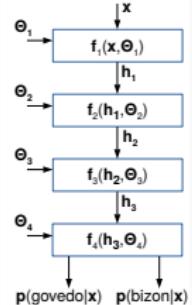
# model
h1 = torch.relu(W1 @ x)
JMLE = F.binary_cross_entropy_with_logits(W2 @ h1, y)
J = JMLE + lambda1 * (W1.pow(2).sum() + W2.pow(2).sum())

# ask autograd to compute the gradients
J.backward()
print(W1.grad)
```

Konvolucijski modeli

Josip Krapac i Siniša Šegvić

Motivacija: razlučiti goveda od bizona



Duboki model ima šansu naučiti značajke koje reagiraju na dijelove objekata, npr: [grba?, mali rogovi?, divljina?, ...]

- bizoni: [DA, DA, DA, ...], goveda: [NE, NE, NE, ...]

Potpuno povezani modeli su u opasnosti da nauče šum jer:

- translatirana slika potpuno različita od originala
- ključne značajke određene lokalnim susjedstvima
- model treba odvojeno naučiti svaku translaciju

Pregled

- Što su konvolucijski modeli?
- Što je konvolucija?
- Zašto konvolucija?
- Sažimanje (*engl. pooling*) i nadopunjavanje.

Što su konvolucijski modeli?

Modeli specijalizirani za podatke s topologijom rešetke

- topologija: oblik strukture definiran relacijom susjedstva

Tipični primjeri:

- vremenski slijed (1D), slika (2D), volumen (3D)

Jednostavna definicija: konvolucijski model ima najmanje jedan konvolucijski sloj umjesto potpuno povezanog sloja.

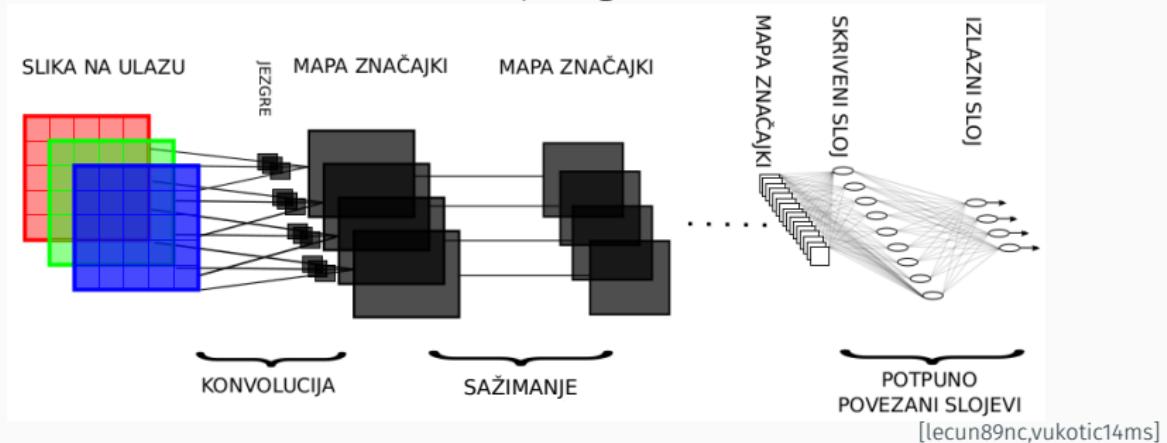
- pored konvolucijskih slojeva u pravilu koristimo slojeve sažimanja i aktivacijske funkcije (ReLU)

Pogledajmo značenje riječi convoluted (nomen est omen):

- extremely complex and difficult to follow
- intricately folded, twisted, or coiled

Što su konvolucijski modeli?

Klasična struktura konvolucijskog modela (LeNet-5):



- konvolucijski slojevi transformiraju tenzore trećeg reda:
 - dvije prostorne, jedna "semantička" dimenzija
 - mi ćemo prvo prepostaviti da je semantička dimenzija 1
- transformacije su lokalne: "pikseli" izlaza ovise o lokalnom susjedstvu piksela ulaza
- težine su tenzori četvrtog reda (!)

Što je to konvolucija?

Konvoluciju definiramo kao skalarni produkt jedne funkcije s obzirom na posmagnutu i reflektiranu drugu funkciju:

$$h(t) = (w * x)(t) = \int_{\mathcal{D}(w)} w(\tau)x(t - \tau)d\tau$$

U strojnom učenju, pod konvolucijom najčešće podrazumijevamo unakrsnu korelaciju:

$$h(t) = (w \star x)(t) = \int_{\mathcal{D}(w)} w(\tau)x(t + \tau)d\tau$$

Konvolucija (odnosno unakrsna korelacija) nam je zanimljiva kao diferencijabilna operacija sa slobodnim parametrima

Jednadžbe pokazuju da jezgru w možemo koristiti za ekstrakciju lokalnih značajki iz signala x

Što je to konvolucija?

Primjer: praćenje svemirskog broda laserskim senzorom koji daje izlaz $x(t)$, poziciju broda trenutku t .

- mjerenja su pokvarena šumom, želimo dobiti usrednjenu predikciju h

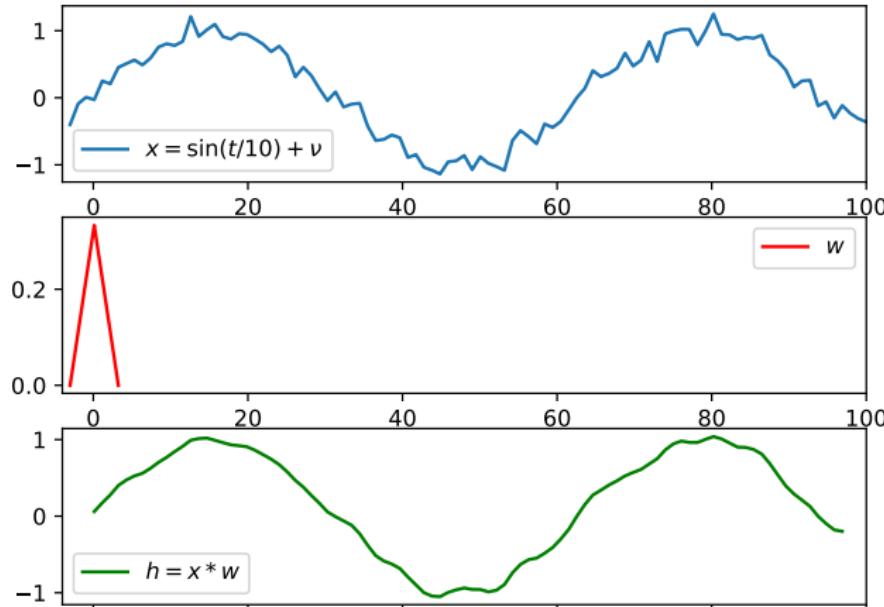
Parametrizirajmo postupak usrednjavanja funkcijom $w(\tau)$

- $w(\tau)$ kazuje koliki je doprinos mjerenja $x(t + \tau)$ filtriranom izlazu $h(t)$.

Filtrirani izlaz dobivamo skalarnim produktom funkcije w s posmakenutim signalom x :

$$h(t) = (w \star x)(t) = \int_{-\infty}^{\infty} w(\tau)x(t + \tau)d\tau$$

Što je to konvolucija?



$$h(t) = (w \star x)(t) = \int_{\mathcal{D}(w)} w(\tau)x(t+\tau)d\tau$$

Što je konvolucija?

Usrednjavanje signala postigli smo unakrsnom korelacijom ulaza $x(t)$ s prikladnom funkcijom $w(t)$:

$$h(t) = w(t) \star x(t)$$

U kontekstu "konvolucijskih" modela:

- funkcija x (argument) je **ulaz**,
- funkcija w (argument) je **jezgra**, (slobodni parametri)
- funkcija h (rezultat) se naziva **mapa značajki**.

Što je konvolucija?

U diskretnom slučaju umjesto integrala koristimo zbroj:

$$h(t) = (w \star x)(t) = \sum_{\tau=-\infty}^{\infty} w(\tau)x(t+\tau)$$

Prepostavljamo da je domena ulaza i jezgre konačan skup, tj. da su funkcije x i w izvan domene jednake 0:

$$h(t) = (w \star x)(t) = \sum_{\tau=\tau_{\min}}^{\tau_{\max}} w(\tau)x(t+\tau)$$

U primjenama je su x i w obično višedimenzionalne funkcije, tj. $\mathbf{x}(t), \mathbf{w}(t) \in \mathbb{R}^d$.

Što je konvolucija?

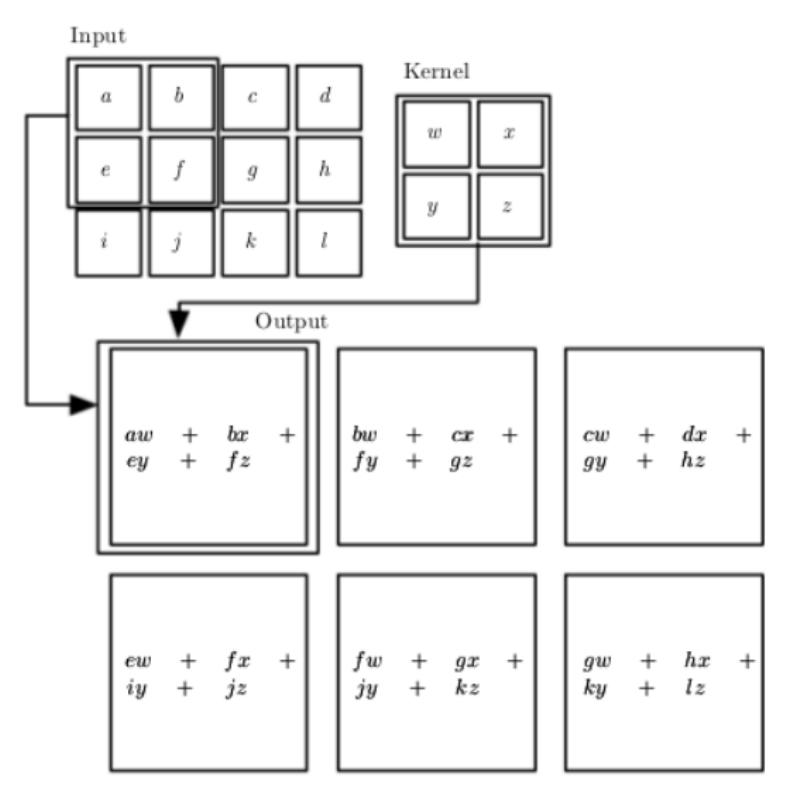
Korelacija se može primjenjivati kroz više dimenzija, npr. ako je argument ulaza dvodimenzionalan, kao u slučaju slika:

$$S(i, j) = (K \star I)(i, j) = \sum_{m=m_{\min}}^{m_{\max}} \sum_{n=n_{\min}}^{n_{\max}} K(m, n) \cdot I(i + m, j + n)$$

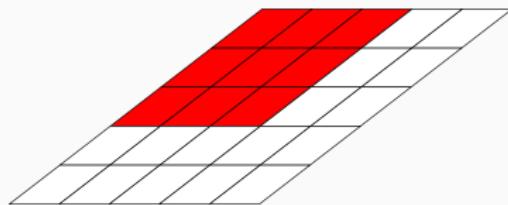
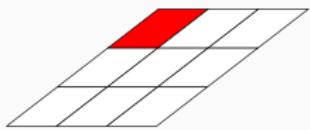
- rasponi min i max određeni vrijednostima na kojima su I i K definirani (tj. $\neq 0$).

Jezgra (3×3 - 7×7) je tipično manja od slike (MPx)

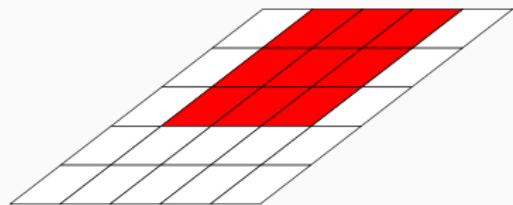
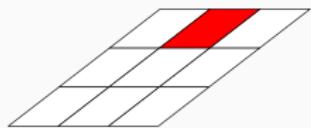
Što je to konvolucija?



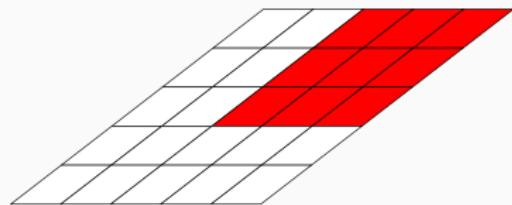
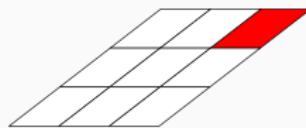
Što je to konvolucija?



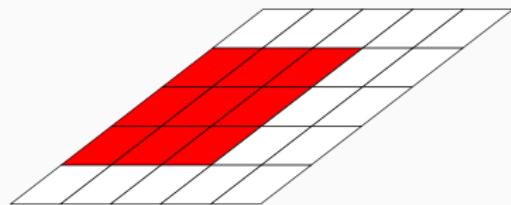
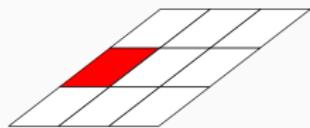
Što je to konvolucija?



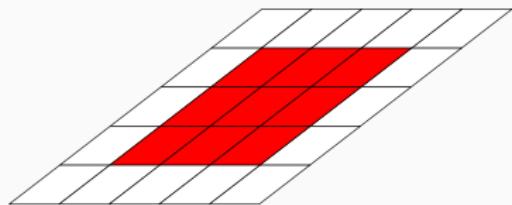
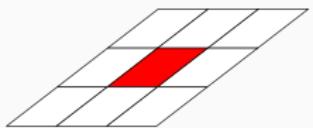
Što je to konvolucija?



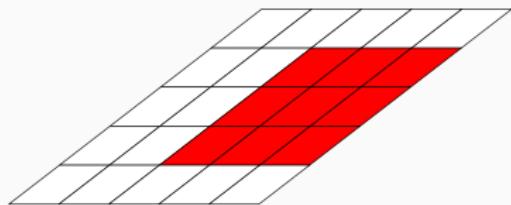
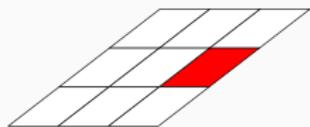
Što je to konvolucija?



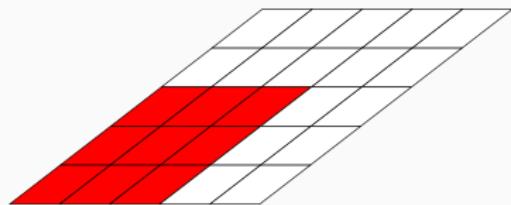
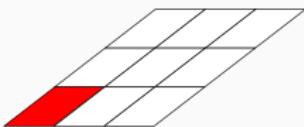
Što je to konvolucija?



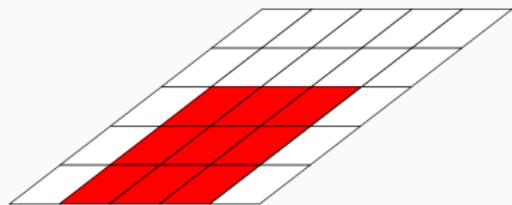
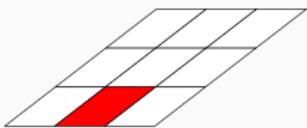
Što je to konvolucija?



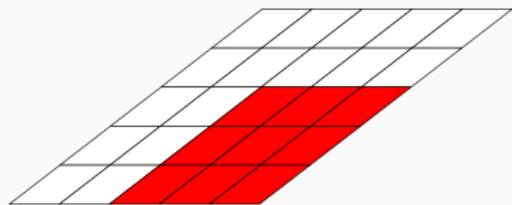
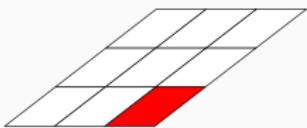
Što je to konvolucija?



Što je to konvolucija?

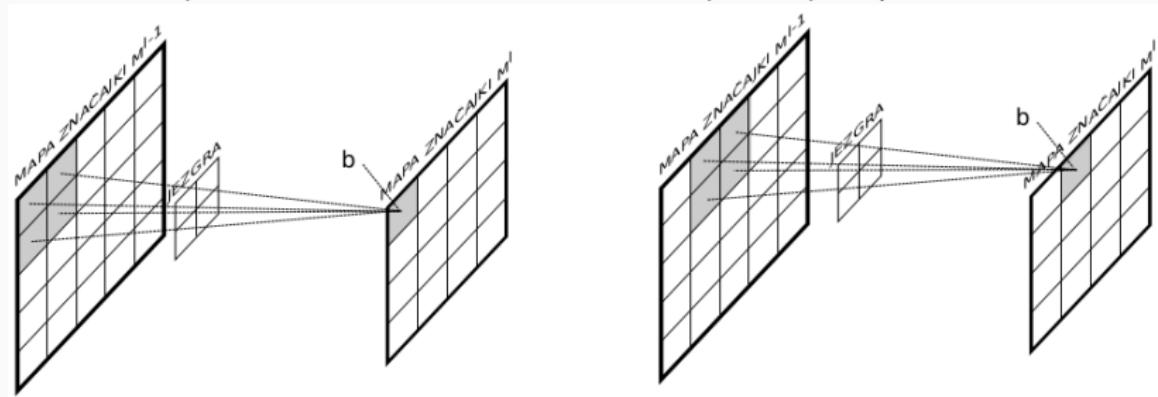


Što je to konvolucija?



Što je to konvolucija?

Konvolucija modelira lokalne interakcije i dijeli parametre:



Veza između ulaza i izlaza je linearna, ali:

- elementi izlaza M^l ovise o **lokalnom** susjedstvu ulaza M^{l-1}
- svi elementi mape značajki se računaju uz pomoć **istog** (dijeljenog) skupa parametara

Zašto konvolucija?

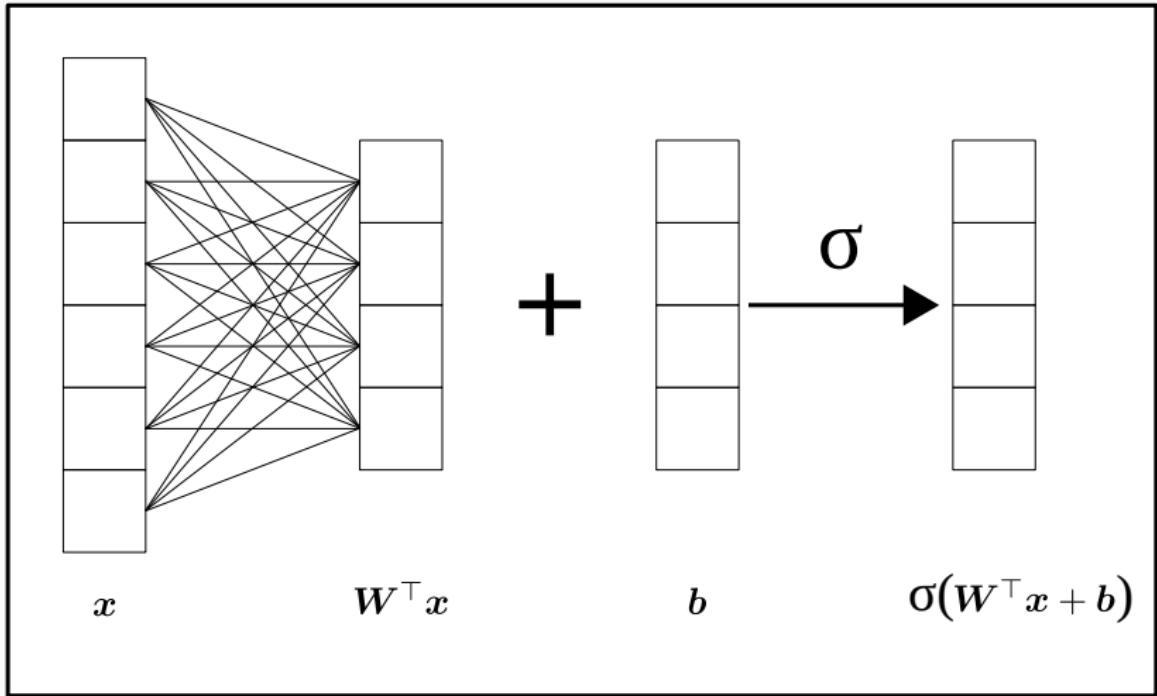
Konvolucija je slična potpuno povezanom sloju, ali postoje razlike:

- možemo modelirati samo lokalne interakcije.
- dijeljenje parametara → izlazna reprezentacija je **ekvivariantna** s obzirom na pomak.

Svaka konvolucija može se predstaviti odgovarajućom afinom transformacijom:

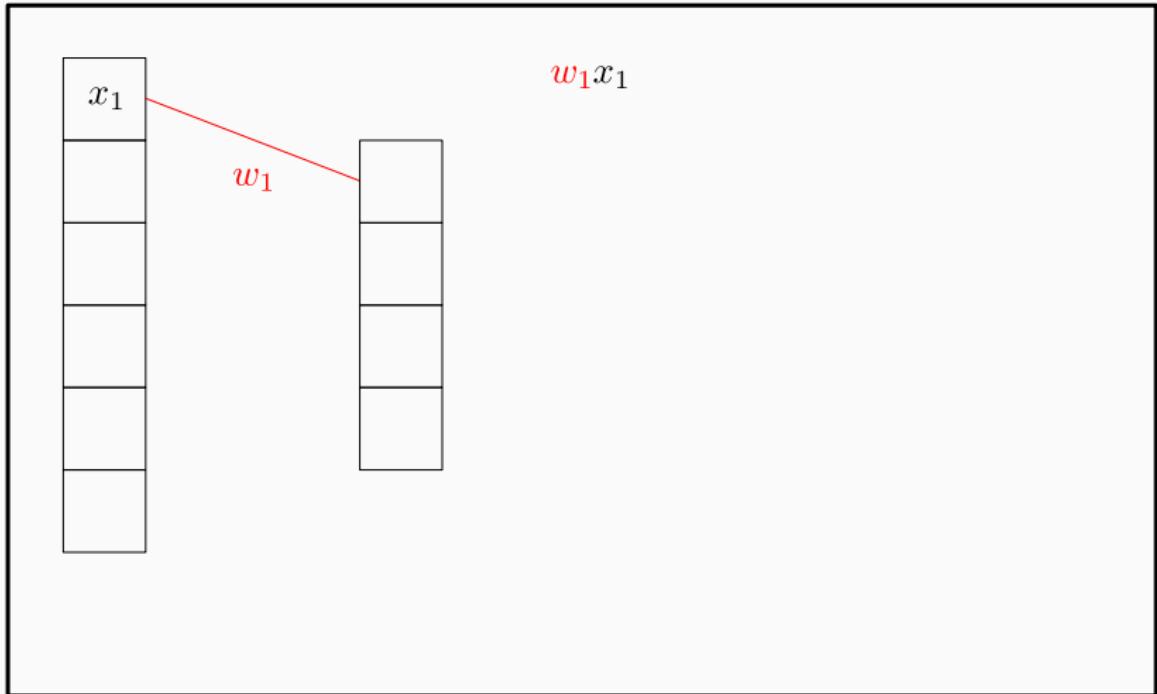
- konvolucijski sloj je regularizirana specijalizacija potpuno povezanog sloja.

Potpuno povezani sloj

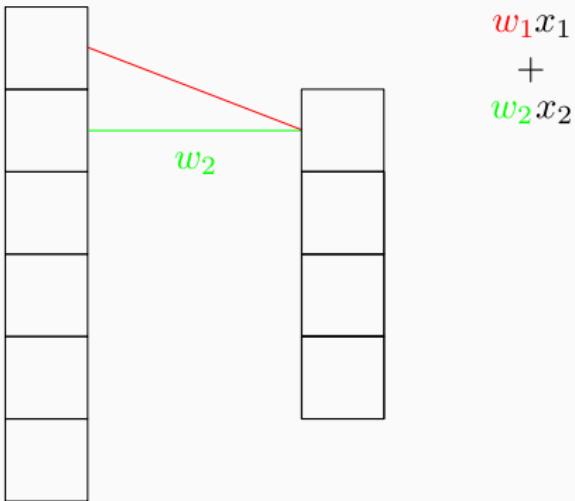


$$f(x, \Theta = (W, b)) = Wx + b$$

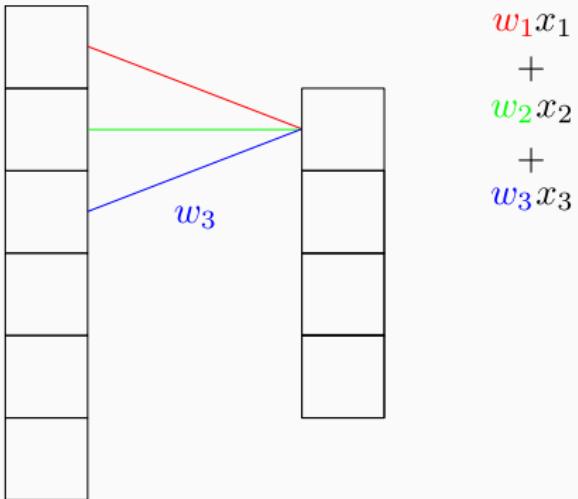
Konvolucijski sloj (linearni dio)



Konvolucijski sloj (linearni dio)



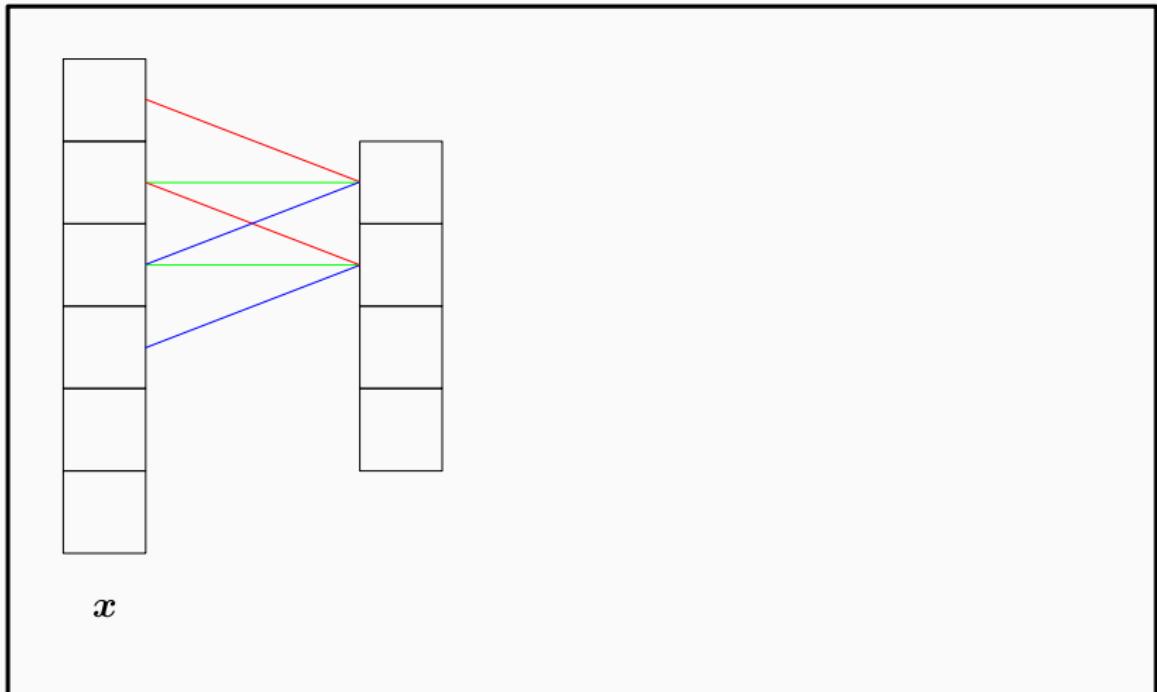
Konvolucijski sloj (linearni dio)



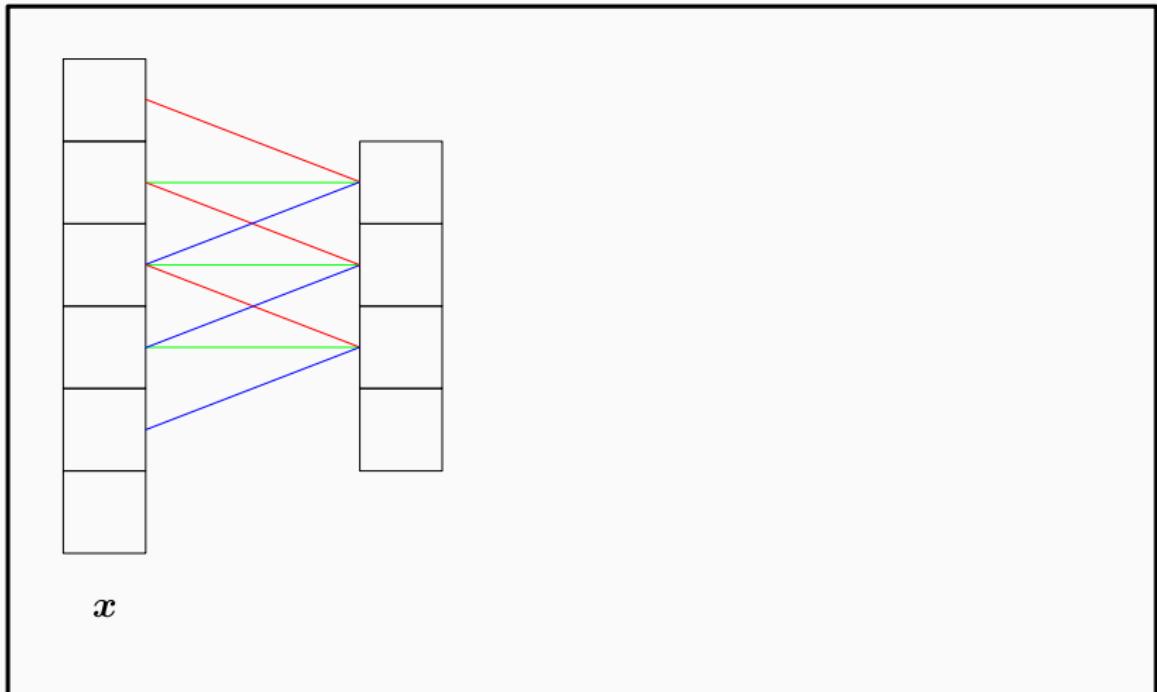
Konvolucijski sloj (linearni dio)



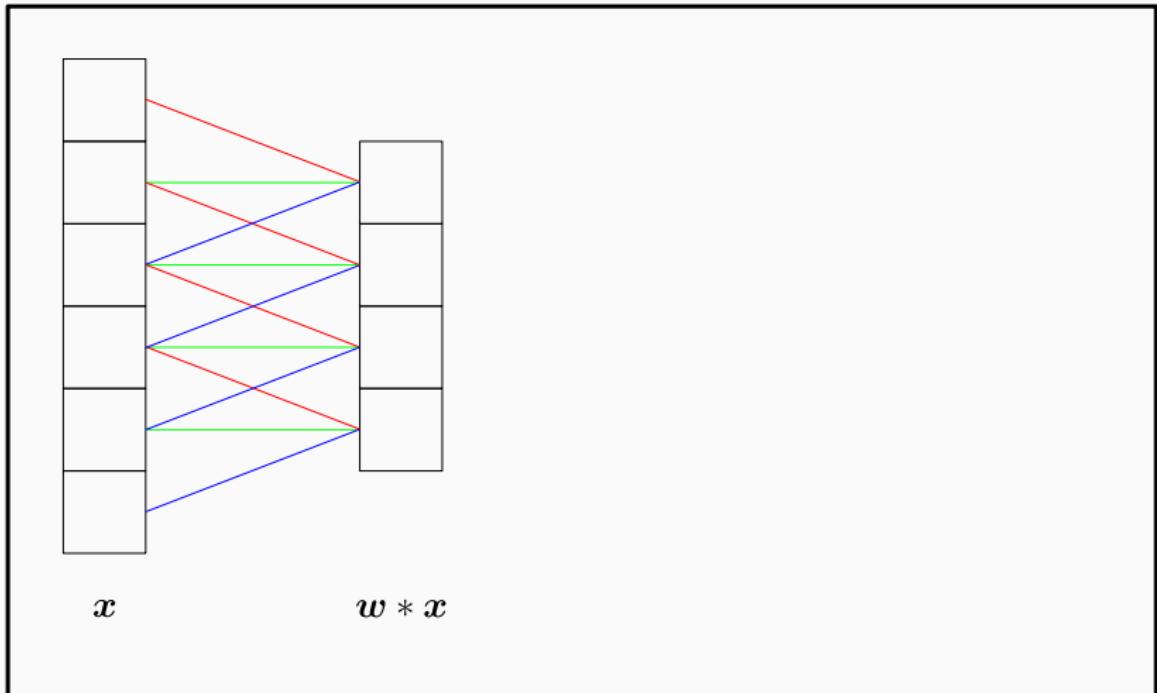
Konvolucijski sloj (linearni dio)



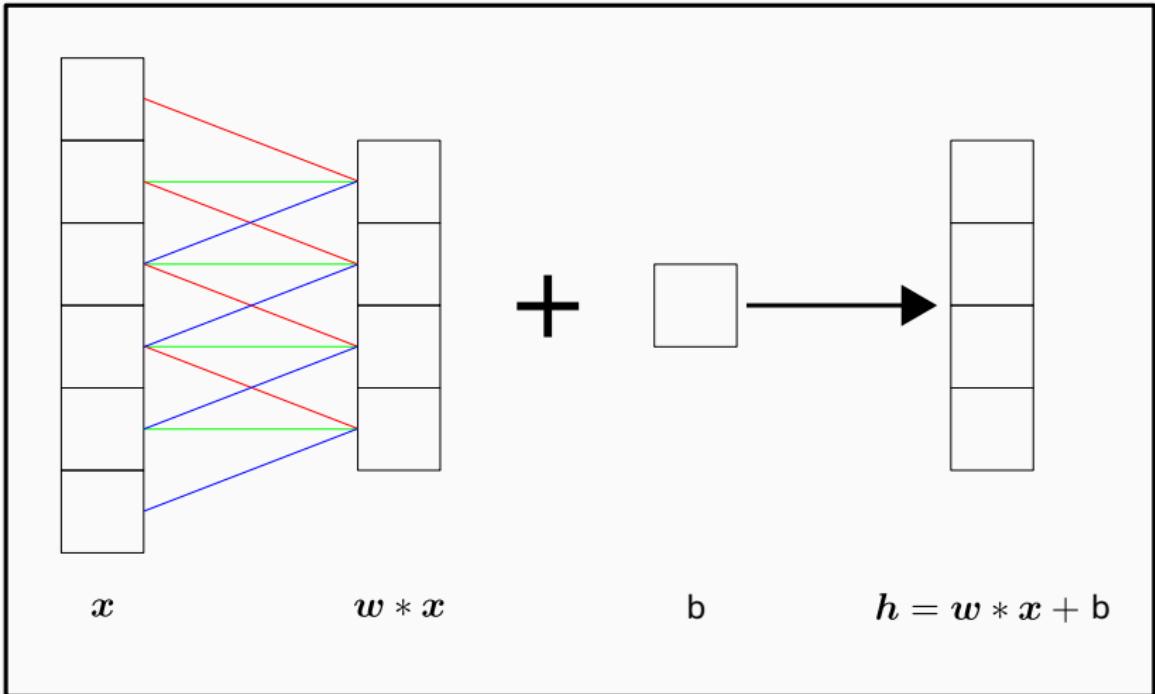
Konvolucijski sloj (linearni dio)



Konvolucijski sloj (linearni dio)



Konvolucijski sloj (linearni dio)

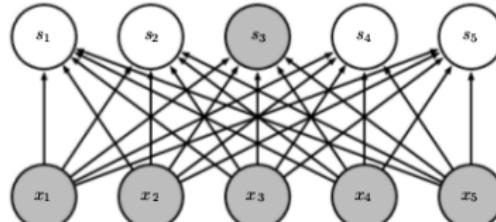
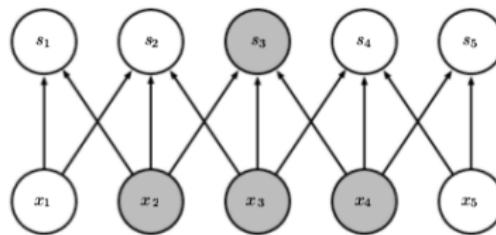


$$f(x, \Theta = (w, b)) = w * x + b \cdot \mathbf{1}$$

Lokalne interakcije

Usporedba konvolucije s potpuno povezanim slojem:

- konvolucijske aktivacije (gore) "vide" samo mali broj ulaza
- aktivacije potpuno povezanog sloja (dolje) "vide" sve ulaze
- konvolucijski sloj ima manje veza i manje parametara.

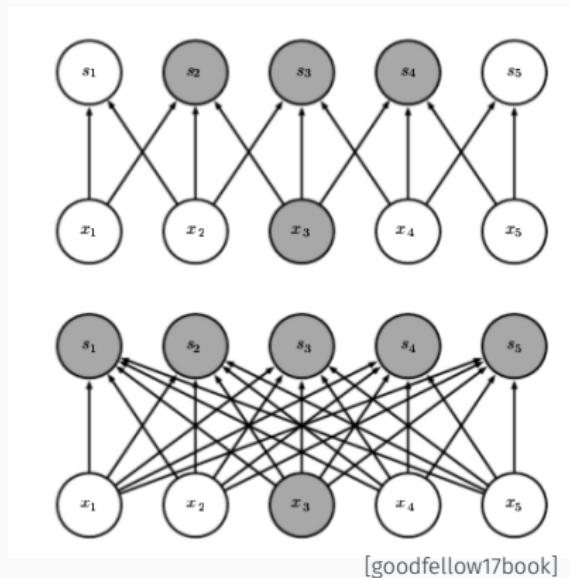


[goodfellow17book]

Lokalne interakcije (2)

Usporedba konvolucije s potpuno povezanim slojem (2):

- ulazi konvolucije (gore) utječu na samo mali broj izlaza
- ova spoznaja sugerira da ćemo širenje gradijenata unatrag također izražavati konvolucijom



Lokalne interakcije (3)

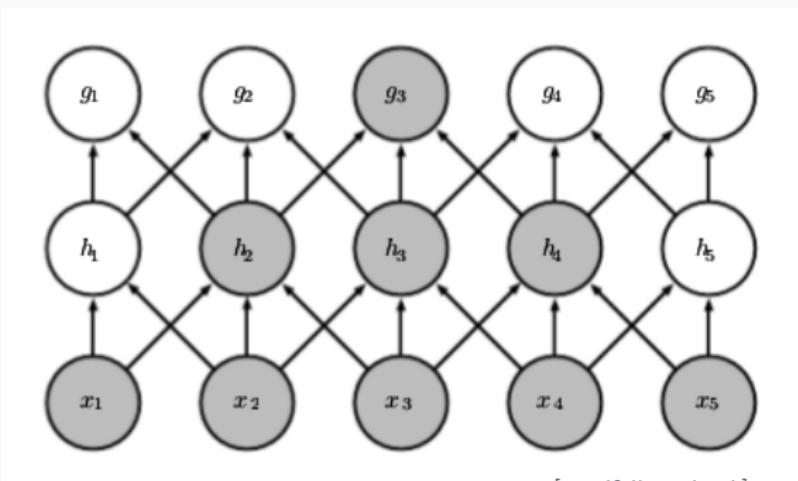
Prednosti konvolucije pred potpuno povezanim slojem:

- brža evaluacija $O(m \cdot n)$ vs $O(k \cdot n)$
 - k širina jezgre
 - m dimenzija ulazne latentne reprezentacije
 - $k \ll m.$
- manji model: $k \cdot n$ vs $m \cdot n$ parametara
 - ovdje razmatramo samo lokalnost, a zanemarujemo dijeljenje parametara.
- manje parametara za naučiti, s istom količinom označenih podataka.

Lokalne interakcije (4)

Ipak, značajke dubokog konvolucijskog modela **indirektno** mogu modelirati interakciju velike regije ulaznih značajki:

- **receptivno polje** značajke: skup svih elemenata ulaznog sloja koje mogu utjecati na tu značajku.
- receptivno polje konvolucijskih aktivacija raste s dubinom.



[goodfellow17book]

Dijeljenje parametara (ili povezivanje težina)

Sve izlazne značajke računaju se s obzirom na isti skup težina:

- skup težina za računanje značajke h_{00} isti je kao i skup težina za računanje značajke $h_{kl} \forall k, l$

Umjesto da učimo odvojen skup parametara za svaki od n izlaza, svi izlazi dijele jedan te isti skup parametara:

- više signala za učenje
- manja opasnost da će model biti prenaučen

Dijeljenje parametara (2)

Prednosti pred potpuno povezanim slojem:

- još manje parametara ($m \cdot n$ vs k): bolja statistička efikasnost modela.
- računska složenost evaluacije: ista kao i za model koji ima samo lokalne interakcije ali ne dijeli težine $O(nk)$.

Ekvivariantnost na pomak

$f(x)$ je **ekvivariantna** s obzirom na g ako vrijedi:

$$f(g(x)) = g(f(x))$$

Konvolucija (f) je ekvivariantna s obzirom na pomak (g):

- izlaz konvolucije prikazuje prostornu mapu značajki ulaznog tenzora (vremenskog slijeda, matrice, volumena)
- ako pomaknemo ulaz, pomaknut će se i mapa značajki.
- konvolucijski modeli su prikladni za slike, govor, jezik, bioinformatiku, ...

Konvolucija nije ekvivariantna s obzirom na neke druge transformacije ulaza, npr. skaliranje ili rotaciju.

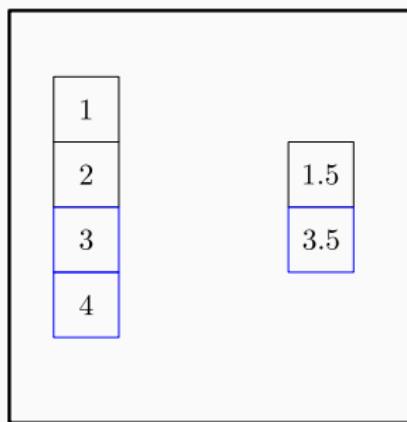
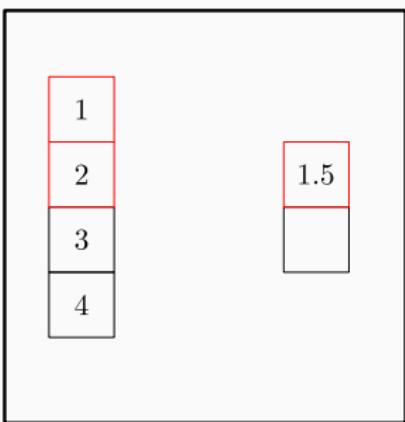
Sloj sažimanja

Funkcija sažimanja (*eng. pooling function*) mapira skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu.

Obično se računa statistički pokazatelj ulaznih značajki, npr. srednja ili maksimalna vrijednost.

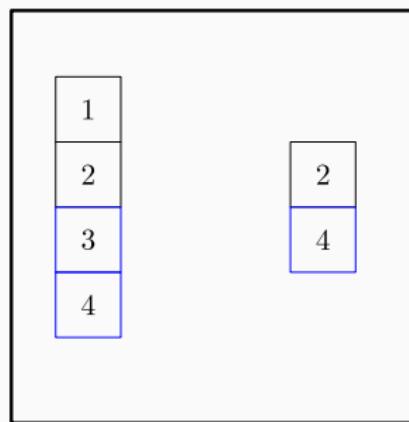
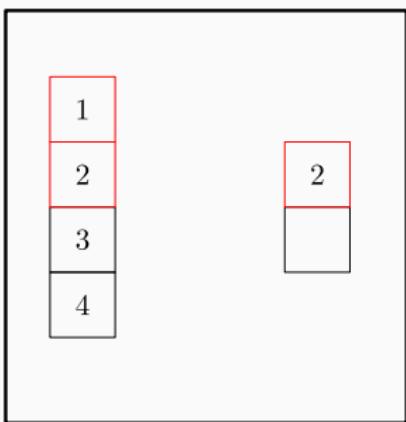
Sažimanje maksimalnom i srednjom vrijednošću

Sažimanje srednjom vrijednošću



Sažimanje maksimalnom i srednjom vrijednošću

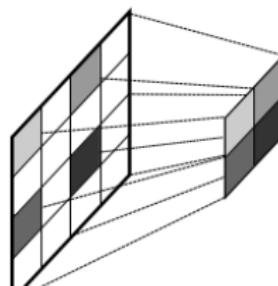
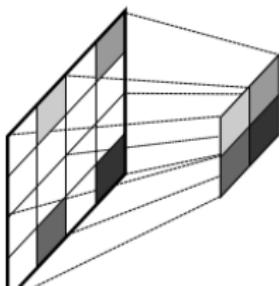
Sažimanje maksimalnom vrijednošću



Sloj sažimanja: motivacija

Povećanje invarijantnosti na pomak

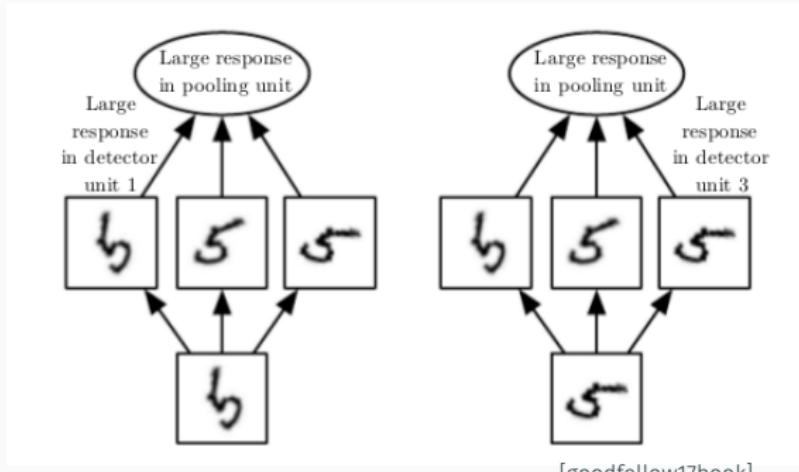
- $f(x)$ je invarijantna s obzirom na g ako: $f(g(x)) = f(x)$
- posebno korisno ako je za raspoznavanje važno detektirati prisutnost koncepta nego lokaciju
 - npr. kod detekcije lica: pomak očiju u odnosu na nos varira od osobe do osobe
- veličina regije sažimanja regulira dozu invarijantnosti: veća regija \rightarrow invarijantnost na veće pomake
 - npr. kod kategorizacije slika: objekt koji definira razred može biti bilo gdje u slici



Sloj sažimanja: motivacija (2)

Sažimanje se može provesti ne samo preko susjednih značajki iste mape nego i preko različitih mapa.

Tada model može naučiti invarijantnost i na druge transformacije.

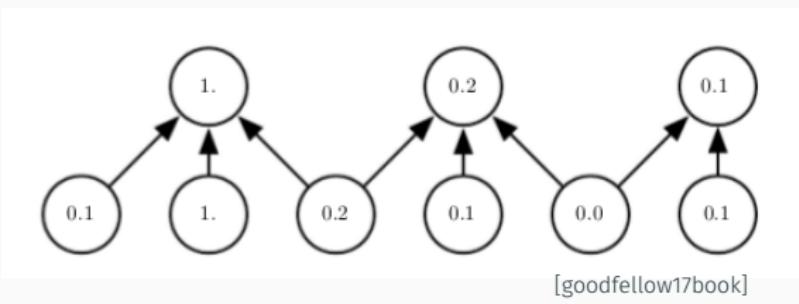


[goodfellow17book]

Sloj sažimanja: lokalna primjena

Sažimanje tipično provodimo na nepreklapajućim oknima $k \times k$

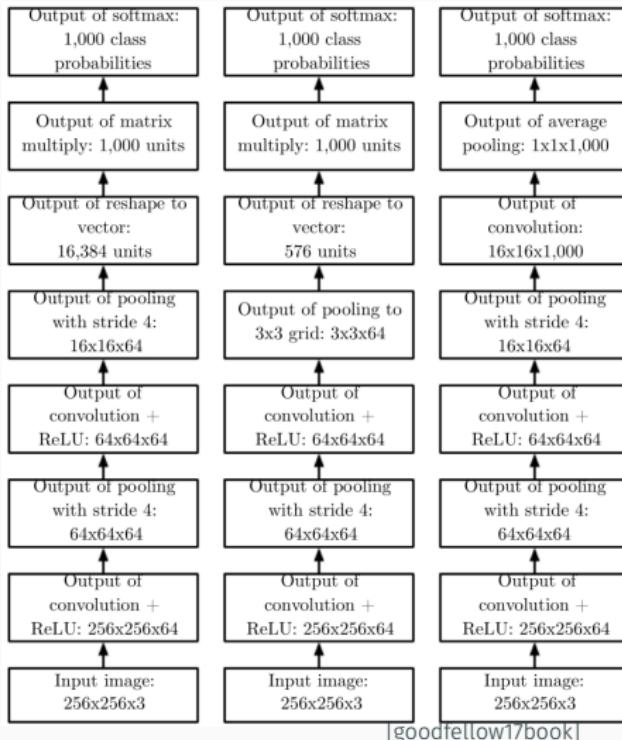
- mapa značajki se podijeli na regije
- svaka regija se sažima u jednu značajku nepromijenjene semantičke dimenzionalnosti
- izlazni korak k : mapa značajki se smanjuje k puta
 - najčešće: $k = 2$, izlaz $(H/2, W/2)$
 - ponekad ciljamo izlaz $q \times q$: $(k_h, k_w) = (H/q, W/q)$
 - moderne modele tipično zaključujemo **globalnim sažimanjem**: $(k_h, k_w) = (H, W)$, izlaz 1×1 .



Sloj sažimanja: primjena preko rešetke

Ponekad veličina regije sažimanja nije fiksna

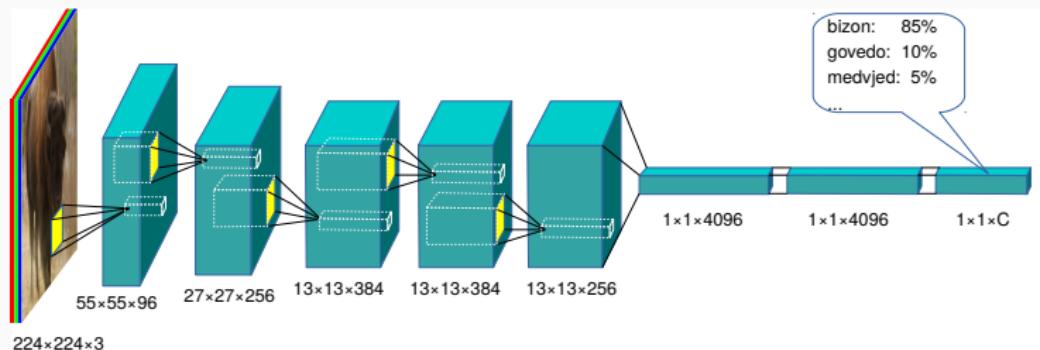
- time omogućavamo procesiranje ulaza različitih veličina



Sloj sažimanja: globalna primjena

Globalno sažimanje koristimo kada latentnu mapu značajki prevodimo u simboličku kategoriju:

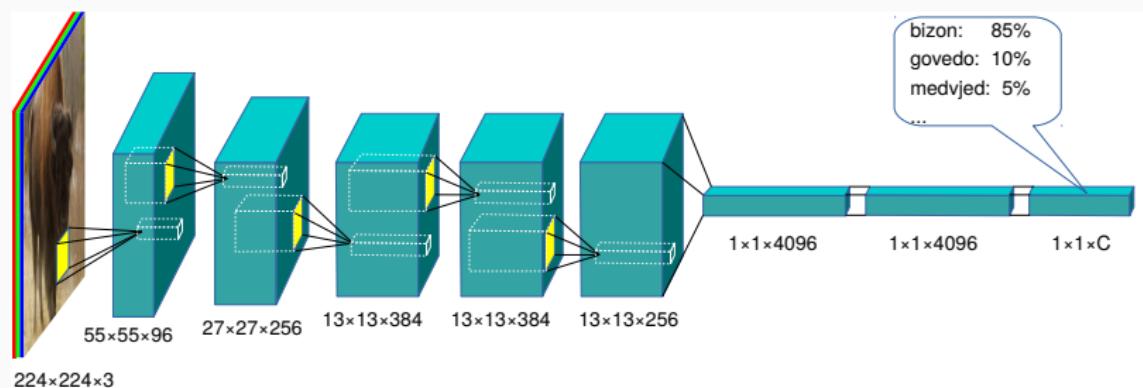
- tijesto za pizzu trebamo premjesiti u tijesto za francuz
- javlja se u najdesnjem stupcu prethodne stranice



Sloj sažimanja: pitanja

Što se zbiva s receptivnim poljem značajki dobivenih sažimanjem (pretp. izlazni korak k)?

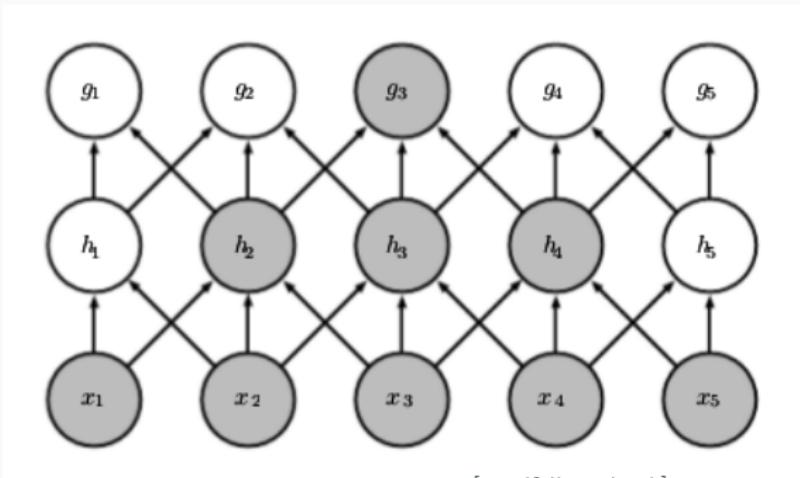
kako sažimanje utječe na broj parametara modela?



Receptivno polje

Učinak konvolucije s jezgrom veličine k:

- uvećanje receptivnog polja za k-1 (ako nije bilo sažimanja)



[goodfellow17book]

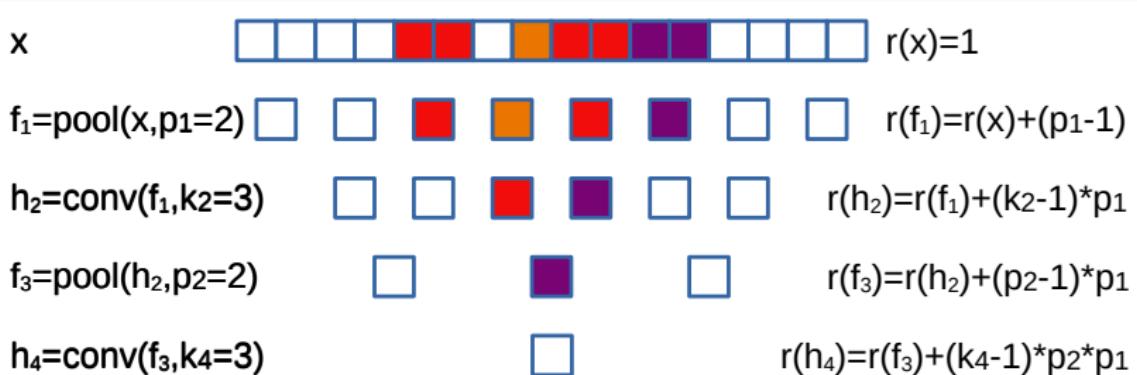
Učinak poduzorkovanja s jezgrom veličine k i korakom k:

- uvećanje svog receptivnog polja za k-1
- umnažanje receptivnog doprinosa svih sljedbenika k puta!

Receptivno polje (2)

Veličinu receptivnog polja možemo odrediti analiziranjem modela unaprijed sloj po sloj:

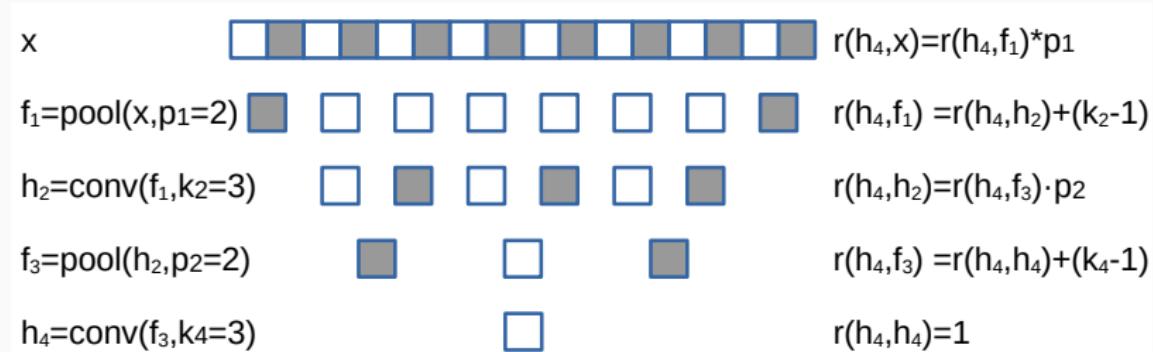
- boje označavaju ukupno receptivno polje odgovarajućih aktivacija
- moramo pamtiti ukupni faktor poduzorkovanja
- tim faktorom množimo receptivni doprinos sloja: $k - 1$



Receptivno polje (3)

Veličinu receptivnog polja možemo odrediti i analizom unatrag (pretpostavljamo da model raste prema prednjoj strani):

- sive aktivacije označavaju uvećanje receptivnog polja u odnosu na sljedeći sloj
- prednost: ne moramo pamtitи ukupni faktor poduzorkovanja (teže je pogriješiti)



Konvolucija i sažimanje: pristranost

Postavljanjem konvolucija i sažimanja u model unosimo sljedeće oblike pristranosti:

- konvolucija: interakcije su lokalne s obzirom na topologiju
→ prepostavljamo topologiju podataka
- konvolucija: predikcija je ekvivariantna s pomakom
- sažimanje: predikcija je invarijantna na male pomake

Te prepostavke povećavaju pristranost i smanjuju varijancu

- teorija: to može dovesti do podnaučenosti i bolje generalizacije
- praksa: nema podnaučenosti, konvolucijski modeli bolje generaliziraju od potpuno povezanih

Konvolucija i sažimanje: pristranost (2)

Konvolucija je kao potpuno povezani sloj u kojem smo težine izvan područja jezgre dekretom postavili na 0:

- gubitak dobivenog modela biti će veći od gubitka odgovarajućeg modela sa slobodnim težinama

Ako konvolucijski sloj dovodi do podnaučenosti (loših rezultata na skupu za učenje):

- možda nisu samo lokalne interakcije važne → povećati veličinu receptivnog polja

Ako sloj sažimanja dovodi do podnaučenosti:

- zadatak ovisi o preciznim lokacijama značajki → smanjiti veličinu regije sažimanja

Nadopunjavanje (eng. padding)

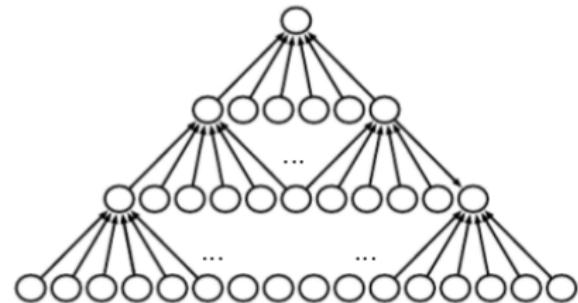
Bez nadopune: reprezentacija se smanjuje s dubinom

- za ulaz veličine m izlaz je $m - k + 1$, gdje je k veličina jezgre
- nedostatak: elementi na rubovima manje utječu na izlaz od elemenata u sredini.
- dubina ograničena veličinom ulaza i konvolucijske jezgre
- u programskim bibliotekama ovakvu konvoluciju označavamo s "**VALID**".

Nadopuna nulama na rubovima: neograničena dubina mreže

- ulaz iste veličine kao i izlaz pod uvjetom da se doda $k - 1$ nula na rubove
- primjer: za $k = 5$ dodamo po dvije 0 sa sve četiri strane (lijево, desno, gore, dolje)
- ovakve konvolucije označavamo s "**SAME**"

Nadopunjavanje (primjer)



Višekanalna 2d konvolucija

Proširujemo definiciju operatora \star prema slici i zadržavamo sintaksu:

$$\mathbf{q} = \mathbf{w} \star \mathbf{p}$$

Izlaz $\mathbf{q}^{(g)}$ zbraja konvolucije odgovarajućih kriški ulaza i g-te jezgre:

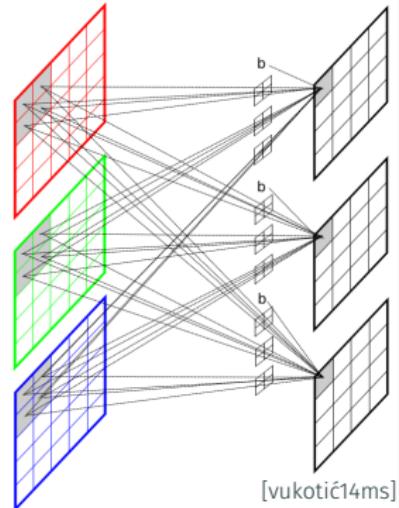
$$\mathbf{q}^{(g)} = \sum_f \mathbf{w}^{(g,f)} \star \mathbf{p}^{(f)}$$

Ista operacija može se izraziti i kroz umnoške skalarja:

$$q_{ij}^{(g)} = \sum_{fuv} p_{i-o_k+u, j-o_k+v}^{(f)} \cdot w_{uv}^{(g,f)}$$

Konvolucije često imaju i pomak: $\mathbf{q} = \mathbf{p} \star \mathbf{w} + \text{broadcast}(\mathbf{b})$

- komponente pomaka odgovaraju kanalima izlaza: $\mathbf{b} = [b_g]$



Konvolucija: zadatak

Razmatramo klasifikacijski konvolucijski model za sive slike dimenzija 28×28 . Arhitektura modela je:

- dva konvolucijska sloja bez nadopunjavanja: jezgra 5×5 s pomakom; aktivacija ReLU; sažimanje maksimumom 2×2 ; korak 2;
 - prvi sloj: 16 kanala, drugi sloj: 32 kanala;
- potpuno povezani sloj dimenzije 512 s pomakom + ReLU;
- potpuno povezani sloj $D=10$ s pomakom + softmaks.

Zadatci:

1. Odredite dimenzijske aktivacije, broj parametara te veličinu receptivnog polja u svim slojevima.
2. Natipkajte implementaciju u PyTorchu s metodama `__init__`, `fwd`, i `loss`.

Konvolucija: zadatak 2

Natipkajte svoju implementaciju 1D konvolucije pod Numpyjem (jedna for-petlja, samo unaprijedni prolaz).

```
import numpy as np

def conv1d_my(vector, kernel):
    n = vector.shape[0]
    k = kernel.shape[0]
    out = np.zeros((n-k+1,), dtype=np.float32)
    kernel = np.flip(kernel)
    for i in range(n-k+1):
        out[i] = np.sum(vector[i:i+k] * kernel)
    return out
```

Konvolucija: zadatak 3

Usporedite svoju implementaciju konvolucije s odgovarajućim implementacijama iz torcha i scipyja.

```
import torch
from torch.nn.functional import conv1d as conv1d_torch
from scipy.ndimage import convolve1d as conv1d_scipy

x = np.array([2.0]*3 + [6.0]*4)
w = np.array([-1.0, 1.0])
print(conv1d_my(x,w))
print(conv1d_scipy(x, w, mode='nearest'))
print(conv1d_torch(torch.tensor(x).reshape([1,1,-1]),
                   torch.tensor(w).reshape([1,1,-1])).numpy().squeeze())

# [ 0.  0. -4.  0.  0.  0.]
# [ 0.  0. -4.  0.  0.  0.]
# [ 0.  0.  4.  0.  0.  0.]
```

Duboko učenje

Unutražni prolaz kroz konvolucije

Siniša Šegvić

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

SADRŽAJ

- ponavljanje: afini slojevi, širenje gradijenata unatrag
- gradijenti parametara 1D konvolucije
 - jezgra dimenzije 1
 - jezgra dimenzije k
- gradijenti parametara 2D konvolucije
 - jezgra dimenzija $k \times k$
 - zadatak s unutrašnjim prolazom kroz 2D konvoluciju
- 2D konvolucija nad tenzorima trećeg reda
- učenje konvolucije s korakom
- efikasna implementacija 2D konvolucije

UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz



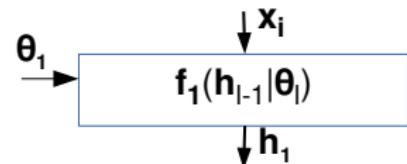
- ulaz: podatak x , izlaz: $\text{softmax}(\mathbf{h}_L)$

UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

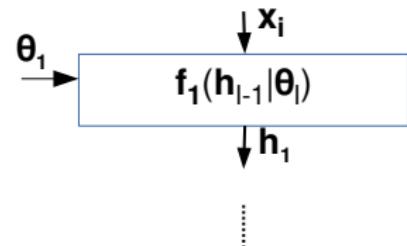


UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

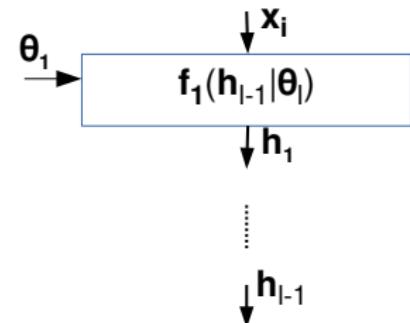


UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

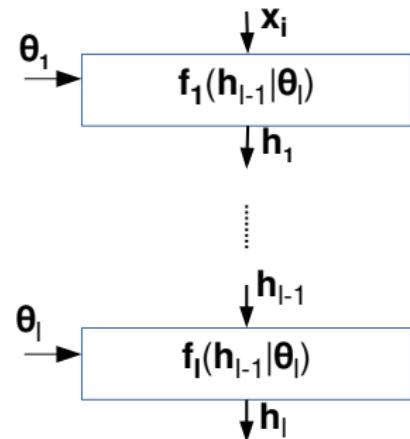


UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

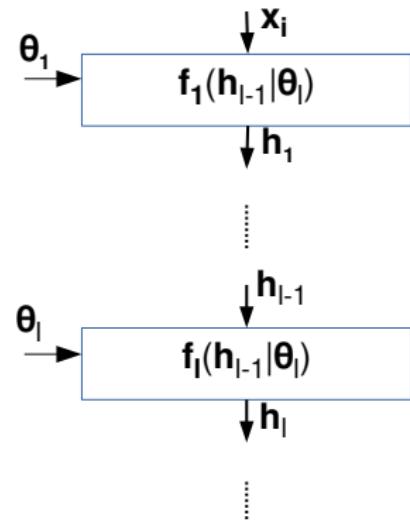


UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

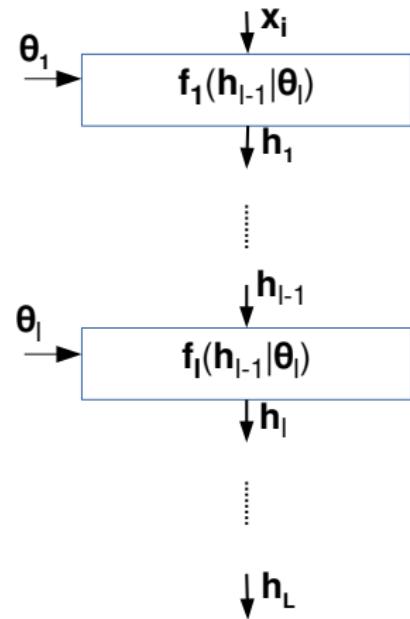


UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$



UNATRAŽNI PROLAZ: DUBOKI MODEL

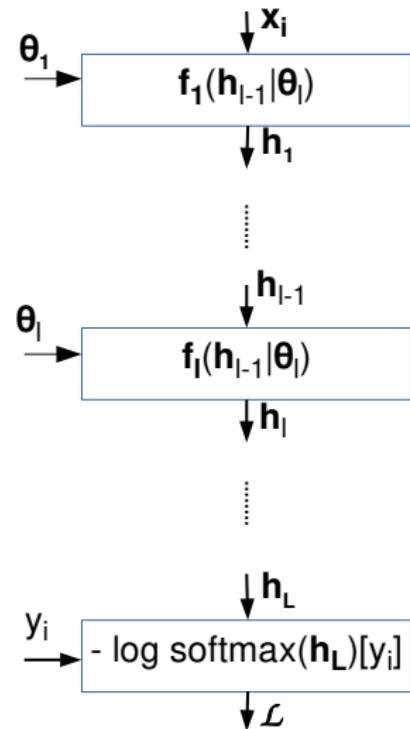
Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$



UNATRAŽNI PROLAZ: DUBOKI MODEL

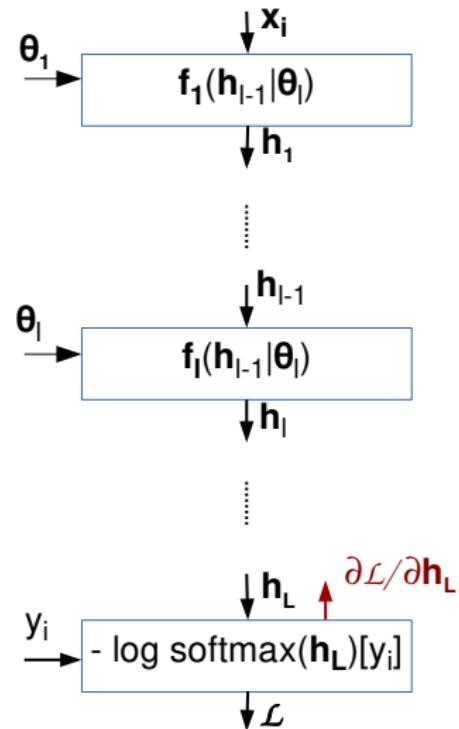
Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$



UNATRAŽNI PROLAZ: DUBOKI MODEL

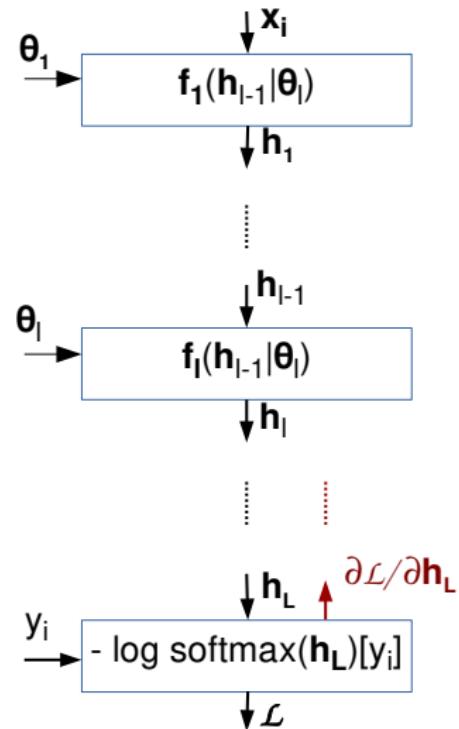
Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$



UNATRAŽNI PROLAZ: DUBOKI MODEL

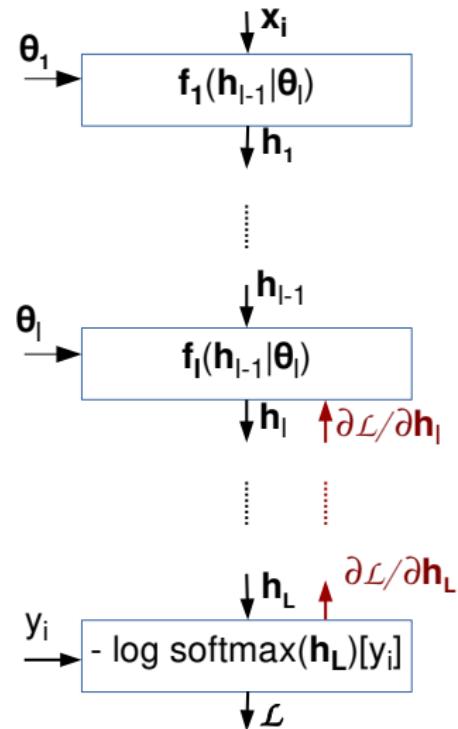
Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$



UNATRAŽNI PROLAZ: DUBOKI MODEL

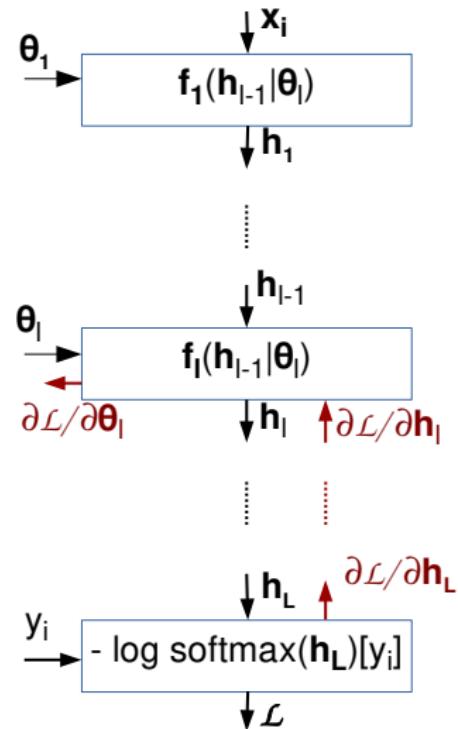
Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$



UNATRAŽNI PROLAZ: DUBOKI MODEL

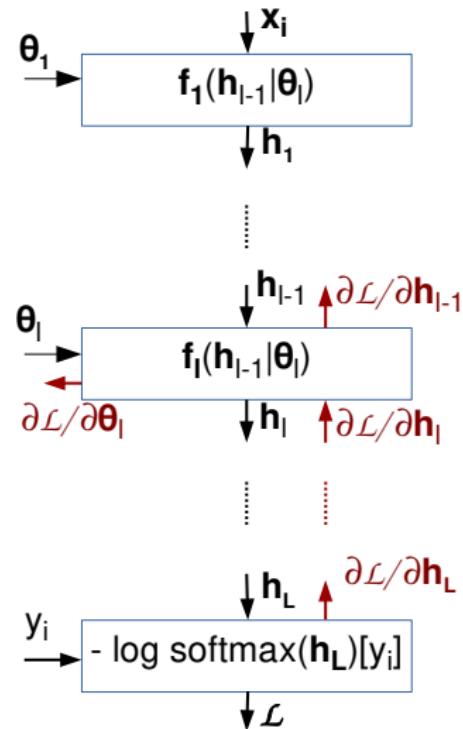
Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(h_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$



UNATRAŽNI PROLAZ: DUBOKI MODEL

Duboki model: kompozicija parametarskih nelinearnih transformacija

Unaprijedni prolaz

- ulaz: podatak x , izlaz: $\text{softmax}(\mathbf{h}_L)$

Unatražni prolaz

- ulaz: predikcija, oznaka, aktivacije
- izlaz: gradijenti parametara $\partial \mathcal{L} / \partial \theta_l$

Definicija l -tog sloja:

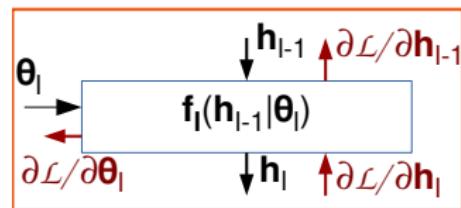
- unaprijedni prolaz: $\mathbf{h}_l = f_l(\mathbf{h}_{l-1} | \boldsymbol{\theta}_l)$

- određivanje gradijenata ulaza:

$$\partial \mathcal{L} / \partial \mathbf{h}_{l-1} = \partial \mathcal{L} / \partial \mathbf{h}_l \cdot \partial \mathbf{h}_l / \partial \mathbf{h}_{l-1}$$

- određivanje gradijenata parametara:

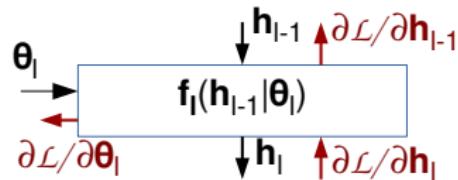
$$\partial \mathcal{L} / \partial \boldsymbol{\theta}_l = \partial \mathcal{L} / \partial \mathbf{h}_l \cdot \partial \mathbf{h}_l / \partial \boldsymbol{\theta}_l$$



UNATRAŽNI PROLAZ: DUBOKI MODEL, DETALJI

Pogledajmo dimenzije Jakobijana $\partial \mathcal{L} / \partial \mathbf{h}_l$:

- $\partial \mathcal{L} / \partial \mathbf{h}_l$: isto kao i \mathbf{h}_l^\top
 - potpuno povezani sloj: $1 \times D_l$
 - konvolucijski sloj (Torch): $[n, c, h, w]$



Dimenzije Jakobijana $\partial \mathcal{L} / \partial \mathbf{W}_l = (\partial \mathcal{L} / \partial \mathbf{W}_{luv})_{u,v=1,1}^{H,W}$:

- $\partial \mathcal{L} / \partial \mathbf{W}_l$: isto kao i \mathbf{W}_l
 - potpuno povezani sloj: $D_l \times D_{l-1}$
 - jednostavan konvolucijski sloj: $k \times k$
 - konvolucijski sloj (Torch): $[c_{\text{out}}, c_{\text{in}}, k, k]$
- $\partial \mathcal{L} / \partial \text{vec}(\mathbf{W}_l)$: isto kao i $\text{vec}(\mathbf{W}_l)^\top$
 - potpuno povezani sloj: $1 \times D_l \cdot D_{l-1}$
 - jednostavan konvolucijski sloj: $1 \times k \cdot k$

UNATRAŽNI PROLAZ: DUBOKI MODEL, DETALJI (2)

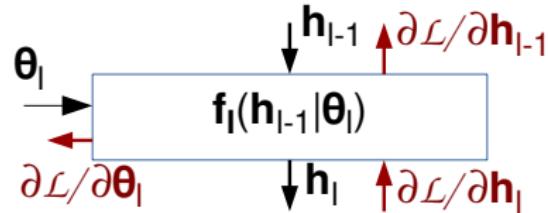
Dimenzije Jakobijana $\partial \mathbf{h}_l / \partial \mathbf{h}_{l-1}$:

- u potpuno povezanom sloju:

- gusta matrica $D_l \times D_{l-1}$

- u konvolucijskom sloju:

- rijetki tenzor reda 4 (odnosno 6)
 - nepraktično odrediti ga izravno!



Dimenzije Jakobijana $\partial \mathbf{h}_l / \partial \mathbf{W}_l$:

- u potpuno povezanom sloju:

- $\partial \mathbf{h}_l / \partial \mathbf{W}_l$: rijetki tenzor trećeg reda

- $\partial \mathbf{h}_l / \partial \text{vec}(\mathbf{W}_l)$: rijetka matrica $D_l \times D_l \cdot D_{l-1}$

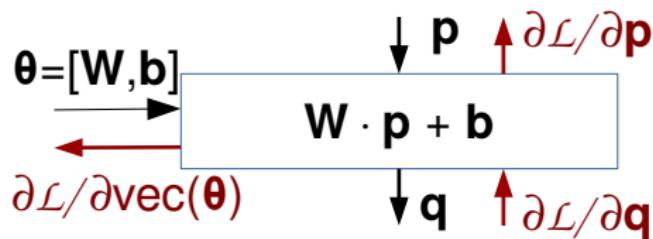
- u praksi koristimo postupak koji uzima u obzir rijetkost $\partial \mathbf{h}_l / \partial \mathbf{W}_l$

- u konvolucijskom sloju: gusti tenzor reda 4 (odnosno 7)

- vidjet ćemo da komponente tog tenszora odgovaraju posmagnutom ulazu sloja $\partial \mathbf{h}_l / \partial \mathbf{W}_{luv} = \text{shift}(\mathbf{h}_{l-1}, u - o_k, v - o_k)$, $o_k = \lfloor k/2 \rfloor + 1$

UNATRAŽNI PROLAZ: POTPUNO POVEZANI SLOJ

Unaprijedni prolaz: množenje matricom plus pomak



Gradijenti ulaza: $\partial \mathcal{L} / \partial p = \partial \mathcal{L} / \partial q \cdot \partial q / \partial p = \partial \mathcal{L} / \partial q \cdot W$

Gradijenti vektoriziranih parametara:

$$\partial \mathcal{L} / \partial \text{vec}(W) = \partial \mathcal{L} / \partial q \cdot \partial q / \partial \text{vec}(W)$$

Problem: dimenzije $\partial q / \partial \text{vec}(W)$

- Jakobijan dimenzija $\dim(q) \times \dim(q) \cdot \dim(p)$
- npr. za MNIST: $784 \cdot 10^2$ množenja po podatku

UNATRAŽNI PROLAZ: POTPUNO POVEZANI SLOJ (DETALJI)

Jakobijan $\partial \mathbf{q} / \partial \mathbf{W}$ je rijedak zbog specifične strukture ovisnosti:

- svaki izlaz q_j ovisi samo o jednom retku težina - $\mathbf{W}_{j,:}$,
- s druge strane, $\partial \mathbf{q} / \partial \mathbf{p}$ je gust jer svaki q_j ovisi o svakom p_i

Ideja 1: iskoristiti strukturu za efikasniji izračun gradijenata parametara

$$\partial \mathcal{L} / \partial \mathbf{W}_{j,:} = \partial \mathcal{L} / \partial q_j \cdot \partial q_j / \partial \mathbf{W}_{j,:} = \partial \mathcal{L} / \partial q_j \cdot \mathbf{p}^\top$$

Ideja 2: izračunati gradijente svih redaka vanjskim produktom

$$\partial \mathcal{L} / \partial \mathbf{W} = [\partial \mathcal{L} / \partial \mathbf{q}]^\top \cdot \mathbf{p}^\top$$

U literaturi se nađe i sljedeći izraz (veza s pravilom ulančavanja?):

$$\partial \mathcal{L} / \partial \mathbf{W}^\top = \mathbf{p} \cdot \partial \mathcal{L} / \partial \mathbf{q}$$

Npr. za MNIST: $784 \cdot 10$ množenja po podatku

- $10 \times$ brže nego prije
- više od 10 dimenzija u skrivenom sloju \Rightarrow veće ubrzanje

UNATRAŽNI PROLAZ: POTPUNO POVEZANI SLOJ, GRUPA

Gubitak grupe je prosječan gubitak preko svih podataka:

$$\partial \mathcal{L} / \partial \mathbf{W} = \frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}}$$

Ideja 3: pozbrajati doprinose podataka grupe matričnim množenjem

$$\partial \mathcal{L}_i / \partial \mathbf{W} = [\partial \mathcal{L}_i / \partial \mathbf{q}]^\top \cdot \mathbf{p}_i^\top$$

\Rightarrow

$$\partial \mathcal{L} / \partial \mathbf{W} = \frac{1}{N} \cdot \mathbf{G} \cdot \mathbf{P}, \text{ gdje su:}$$

$$\mathbf{G}_{:,i} = [\partial \mathcal{L}_i / \partial \mathbf{q}]^\top$$

$$\mathbf{P}_{i,:} = \mathbf{p}_i^\top$$

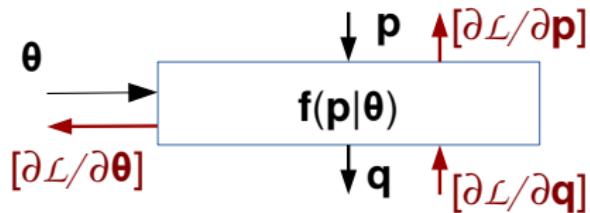
To implicira sljedeću proceduru:

- unaprijedni prolaz pamti ulazne aktivacije svih podataka $\mathbf{P} = \{\mathbf{p}_i^\top\}$
- backprop računa sve gradijente po izlazu sloja $\mathbf{G} = \{\partial \mathcal{L}_i / \partial \mathbf{q}\}^\top$
- gradijenti gubitka grupe po parametrima su: $\frac{1}{N} \cdot \mathbf{G} \cdot \mathbf{P}$

UNATRAŽNI PROLAZ: SUČELJE SLOJEVA

Vidimo da sveki sloj možemo opisati sljedećim sučeljem:

```
class Layer: # ...
    def fwd_pass(self, p):
        self.p=p
        q=f(p, self.theta)
        return q
    def bwd_pass(self, dq):
        dp = g(dq, self.theta)
        self.dtheta = h(dq, self.p)
        return dp
```



Svaki sloj mora pamtiti:

- parametre (potrebni za gradijente po ulazu sloja)
- ulazne aktivacije (potrebne za gradijente po parametrima)
- gradijente parametara (potrebni za učenje)

UNATRAŽNI PROLAZ: ALGORITAM

Sad smo spremni skicirati općeniti backprop:

- ovaj algoritam pokrećemo pozivom `loss.backward()` u torchu
- više o unatražnim automatskim gradijentima: [gavranovic18sem]

```
def backprop(L, x,y):  
    # forward pass  
    q=x # input  
    for l in L:  
        q=l.fwd_pass(q)  
    # backward pass, q=logits  
    dq=grad_loss(q, y)  
    for l in reversed(L):  
        dq=l.bwd_pass(dq)
```

U ovu formulaciju mogu se uklopiti svi slojevi koji:

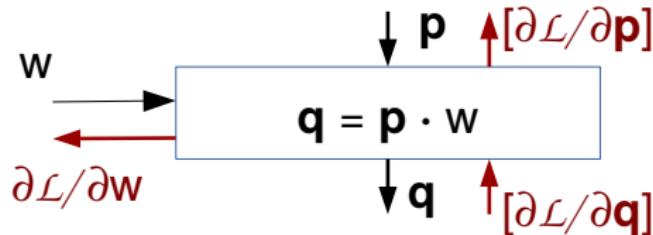
- zadovoljavaju sučelje Layer
- imaju dimenzije ulaza i izlaza kompatibilne sa susjedima

SADRŽAJ

- ponavljanje: unatražno učenje, potpuno povezani slojevi
- **gradijenti parametara 1D konvolucije**
 - jezgra dimenzije 1
 - jezgra dimenzije k
- gradijenti parametara 2D konvolucije
 - jezgra dimenzija $k \times k$
 - zadatak s unutrašnjim prolazom kroz 2D konvoluciju
- 2D konvolucija nad tenzorima trećeg reda
- učenje konvolucije s korakom
- efikasna implementacija 2D konvolucije

VEKTORI: SKALARNA JEZGRA, JEDNADŽBE

Unaprijedni prolaz: $\mathbf{q} = w \cdot \mathbf{p}$



Gradijenti po ulazu:

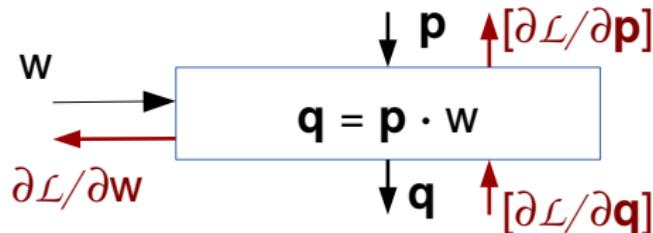
$$\begin{aligned}\partial \mathcal{L} / \partial \mathbf{p} &= \partial \mathcal{L} / \partial \mathbf{q} \cdot \partial \mathbf{q} / \partial \mathbf{p} \\ &= \partial \mathcal{L} / \partial \mathbf{q} \cdot (w \cdot \mathbf{I}) \\ &= w \cdot \partial \mathcal{L} / \partial \mathbf{q}\end{aligned}$$

Gradijenti po parametrima:

$$\begin{aligned}\partial \mathcal{L} / \partial w &= \partial \mathcal{L} / \partial \mathbf{q} \cdot \partial \mathbf{q} / \partial w \\ &= \partial \mathcal{L} / \partial \mathbf{q} \cdot \mathbf{p}\end{aligned}$$

VEKTORI: SKALARNA JEZGRA, JEDNADŽBE (2)

Unaprijedni prolaz: $q = w \cdot p$



Gradijenti po ulazu: $\frac{\partial \mathcal{L}}{\partial p} = w \cdot \frac{\partial \mathcal{L}}{\partial q}$

Gradijenti po parametrima: $\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial q} \cdot p$

Kakve razlike možemo očekivati u "pravoj" konvoluciji?

- gradijenti po ulazu biti će složeniji: q_i ovisi o susjedstvu $\mathcal{N}(p_i)$
- imat ćemo više gradijenata po parametrima, ali oni će biti jednako složeni kao i ovdje

VEKTORI: SKALARNA JEZGRA, KÔD

Sloj kojeg smo upravo definirali možemo opisati sljedećim kodom:

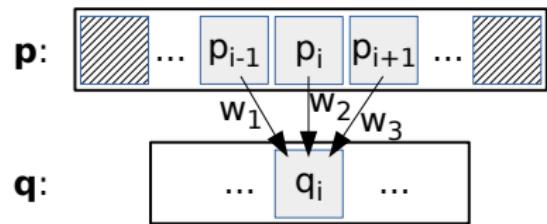
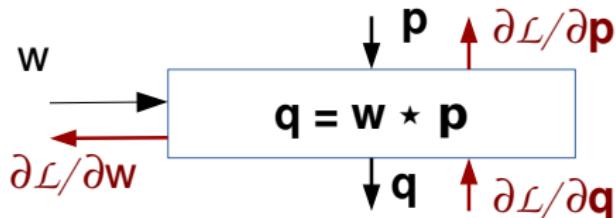
```
class LayerConv1x1: #...
    def fwd_pass(self, p):
        self.p=p
        return self.w*self.p
    def bwd_pass(self,dq):
        dp = self.w * dq
        self.dw = np.dot(dq,self.p)
        return dp
```

Kao i ranije, sloj pamti aktivacije, parametre i gradijente po parametrima

Sloj mora dodatno omogućiti:

- pristup parametru i njegovom gradijentu
- postavljanje parametra (inicijalizacija, učenje)

VEKTORI: JEZGRA DIMENZIJE K, NAPRIJED



Kod 1D konvolucije i-ta izlazna aktivacija odgovara **skalarnom produktu** isječka (crop) ulaza i jezgre:

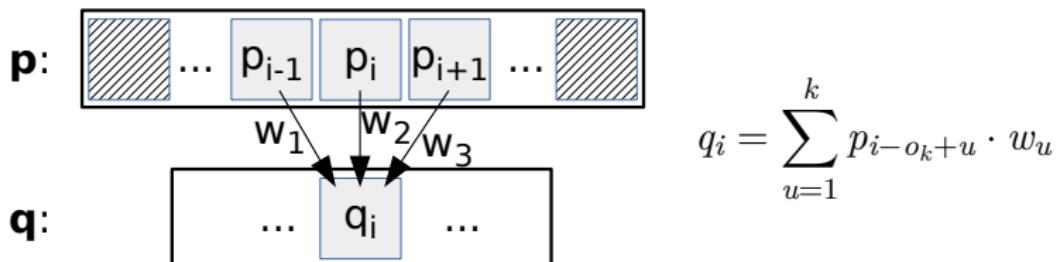
$$q_i = \text{crop}_k(\mathbf{p}, i)^\top \cdot \mathbf{w}$$

Primjerice, za $k=3$, $i=5$ imali bismo: $q_5 = p_4 \cdot w_1 + p_5 \cdot w_2 + p_6 \cdot w_3$

Općenita jednadžba i-te aktivacije ($o_k = \lfloor k/2 \rfloor + 1$, $o_3=2$):

$$q_i = \text{crop}_k(\mathbf{p}, i)^\top \cdot \mathbf{w} = \sum_{u=1}^k p_{i-o_k+u} \cdot w_u$$

VEKTORI: JEZGRA DIMENZIJE K, NATRAG W (1)



Za gradijent po parametru w_u postavljamo pitanje: kako q_i ovisi o w_u ?

$$\partial q_i / \partial w_u = p_{i-o_k+u}$$

Traženi gradijent je skalarni produkt gradijenata po izlazu i posmknutog (shift) ulaza ("ilegalne pristupe" rješava nadopunjavanje):

$$\begin{aligned}\partial \mathcal{L} / \partial w_u &= \sum_i \partial \mathcal{L} / \partial q_i \cdot \partial q_i / \partial w_u = \sum_i \partial \mathcal{L} / \partial q_i \cdot p_{i-o_k+u} \\ &= \partial \mathcal{L} / \partial \mathbf{q} \cdot \text{shift}(\mathbf{p}, -o_k + u)\end{aligned}$$

VEKTORI: JEZGRA DIMENZIJE K, NATRAG W (2)

Raspišimo dobivene gradijente po parametrima za k=3:

$$\partial \mathcal{L} / \partial w_1 = \sum_i \partial \mathcal{L} / \partial q_i \cdot p_{i-1} = \partial \mathcal{L} / \partial \mathbf{q} \cdot \text{shift}(\mathbf{p}, -1)$$

$$\partial \mathcal{L} / \partial w_2 = \sum_i \partial \mathcal{L} / \partial q_i \cdot p_i = \partial \mathcal{L} / \partial \mathbf{q} \cdot \text{shift}(\mathbf{p}, 0)$$

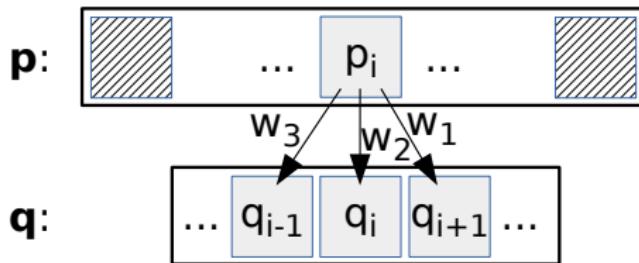
$$\partial \mathcal{L} / \partial w_3 = \sum_i \partial \mathcal{L} / \partial q_i \cdot p_{i+1} = \partial \mathcal{L} / \partial \mathbf{q} \cdot \text{shift}(\mathbf{p}, 1)$$

Vidimo da se gradijenti svih parametara mogu dobiti unakrsnom korelacijom nadopunjenoj (pad) ulaza s gradijentima po izlazu:

$$\partial \mathcal{L} / \partial \mathbf{w} = \partial \mathcal{L} / \partial \mathbf{q} \star \text{pad}(\mathbf{p}, \lfloor k/2 \rfloor)$$

VEKTORI: JEZGRA DIMENZIJE K, NATRAG P (1)

Za gradijente po ulazu sloja ključno je pitanje: koji q_r ovise o p_i i kako?



$$\frac{\partial q_{i-1}}{\partial p_i} = w_3$$

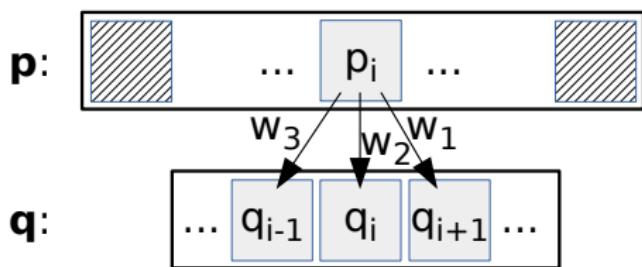
$$\frac{\partial q_{i+1}}{\partial p_i} = w_1$$

$$\frac{\partial q_{i+s}}{\partial p_i} = w_{o_k-s} = w_{2-s}$$

Fokusirajmo se na p_i kao na slici ($o_k = \lfloor k/2 \rfloor + 1$, $o_3=2$):

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_i} &= \sum_r \frac{\partial \mathcal{L}}{\partial q_r} \cdot \frac{\partial q_r}{\partial p_i} \\&= \sum_{s=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \frac{\partial \mathcal{L}}{\partial q_{i+s}} \cdot \frac{\partial q_{i+s}}{\partial p_i} \\&= \sum_{s=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \frac{\partial \mathcal{L}}{\partial q_{i+s}} \cdot w_{o_k-s}\end{aligned}$$

VEKTORI: JEZGRA DIMENZIJE K, NATRAG P (2)



$$\frac{\partial q_{i-1}}{\partial p_i} = w_3$$
$$\frac{\partial q_{i+1}}{\partial p_i} = w_1$$
$$\frac{\partial q_{i+s}}{\partial p_i} = w_{2-s}$$

$$\frac{\partial \mathcal{L}}{\partial p_i} = \sum_{s=-1}^1 \frac{\partial \mathcal{L}}{\partial q_{i+s}} \cdot w_{o_k-s}$$

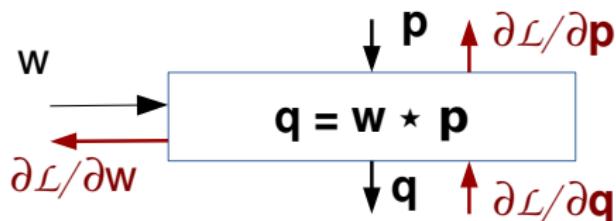
Vidimo da se gradijent po i -tom ulazu dobiva **skalarnim produktom** isječka (crop) gradijenata po izlazu i zrcaljene (flip) jezgre:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \text{sum}(\text{crop}_k(\frac{\partial \mathcal{L}}{\partial \mathbf{q}}, i) \odot \text{flip}(\mathbf{w}))$$

Gradijente po svim ulazima dobivamo **unakrsnom korelacijom** nadopunjениh (pad) gradijenata po izlazu i zrcaljene (flip) jezgre:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}} = \text{flip}(\mathbf{w}) \star \text{pad}(\frac{\partial \mathcal{L}}{\partial \mathbf{q}}, \lfloor k/2 \rfloor)$$

VEKTORI: JEZGRA DIMENZIJE K, SAŽETAK



Unaprijedni prolaz:

$$q_i = \text{crop}_k(\mathbf{p}, i)^\top \cdot \mathbf{w}$$

$$\mathbf{q} = \mathbf{w} * \mathbf{p}$$

Unatražni prolaz po parametrima:

$$\partial \mathcal{L} / \partial \mathbf{w} = \partial \mathcal{L} / \partial \mathbf{q} * \text{pad}(\mathbf{p}, \lfloor k/2 \rfloor)$$

Unatražni prolaz po ulazima:

$$\partial \mathcal{L} / \partial \mathbf{p} = \text{flip}(\mathbf{w}) * \text{pad}(\partial \mathcal{L} / \partial \mathbf{q}, \lfloor k/2 \rfloor)$$

VEKTORI: KOD

```
def my_conv1d(data, kernel):
    data_pad = torch.nn.functional.pad(data,(1,1), 'constant',0)
    return torch.nn.functional.conv1d(
        data_pad.reshape([1,1,-1]),
        kernel.reshape([1,1,-1])).squeeze()
```

D, K = 7, 3

```
w = torch.tensor(np.random.randn(K), requires_grad=True)
p = torch.tensor(np.random.randn(D), requires_grad=True)
q = my_conv1d(p, w)
torch.sum(q).backward()
```

```
dLdq = torch.ones(D, dtype=torch.float64) # dL/dq_j=1 !
dLdp = my_conv1d(dLdq, torch.flip(w, (0,)))
dLdw = my_conv1d(p, dLdq)
```

```
print(dLdw, w.grad)
print(dLdp, p.grad)
```

VEKTORI: ZGLOBNICA

Unaprijedni prolaz nezavisno primjenjuje zglobnicu u svakom ulazu:

$$q_i = \max(0, p_i)$$

Unatražni prolaz (svaki q_i ovisi samo o p_i):

$$\partial \mathcal{L} / \partial \mathbf{p} = \partial \mathcal{L} / \partial \mathbf{q} \cdot \partial \mathbf{q} / \partial \mathbf{p}$$

$$= \partial \mathcal{L} / \partial \mathbf{q} \cdot \begin{bmatrix} [\![p_1 > 0]\!] & & & \\ & [\![p_2 > 0]\!] & & \\ & & \ddots & \\ & & & [\![p_n > 0]\!]\end{bmatrix}$$
$$= \partial \mathcal{L} / \partial \mathbf{q} \odot [\![\mathbf{p} > 0]\!]$$

Posljednji izraz primjenljiv je i na tensore višeg reda.

VEKTORI: GLOBALNO SAŽIMANJE MAKSIMUMOM

Globalno sažimanje maksimumom reducira ulaz u skalar:

$$q = \text{maxpool}(\mathbf{p})$$

Unutrašnji prolaz izražavamo funkcijom onehot

□ vrijedi npr $\text{onehot}^4(2) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$

$$\begin{aligned}\partial \mathcal{L} / \partial \mathbf{p} &= \partial \mathcal{L} / \partial q \cdot \partial q / \partial \mathbf{p} \\ &= \partial \mathcal{L} / \partial q \cdot [\llbracket \arg \max(\mathbf{p}) = 1 \rrbracket, \llbracket \arg \max(\mathbf{p}) = 2 \rrbracket, \dots \\ &\quad \dots, \llbracket \arg \max(\mathbf{p}) = \dim(\mathbf{p}) \rrbracket] \\ &= \partial \mathcal{L} / \partial q \cdot \text{onehot}^n(\arg \max \mathbf{p})\end{aligned}$$

Posljednji izraz primjenljiv je i na tenzore višeg reda, samo tada umjesto n navodimo $\text{shape}(\mathbf{p})$

VEKTORI: SAŽIMANJE S JEZGROM K I KORAKOM K

Sažimanje maxpool_k zasebno reducira nepreklapajuće **regije** ulaza veličine k :

$$\mathbf{q} = \text{maxpool}_k(\mathbf{p}) .$$

Ulez se poduzorkuje $k \times$: $\dim(\mathbf{p}) = N \Rightarrow \dim(\mathbf{q}) = \lceil N/k \rceil$.

Primjer:

$$\text{maxpool}_2 \left(\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \right) = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}$$

VEKTORI: SAŽIMANJE S JEZGROM K I KORAKOM K (NATRAG)

Unatražni prolaz izražavamo funkcijom $\text{embed}_k^n(\mathbf{x}, p)$:

- ulaz: \mathbf{x} , $\dim(\mathbf{x}) = k$,
- izlaz: \mathbf{x}' , $\dim(\mathbf{x}') = n$, takav da $\mathbf{x}'_{[p:p+k]} = \mathbf{x}$
- vrijedi: $\text{onehot}_1^n(p) = \text{embed}_1^n(1, p)$

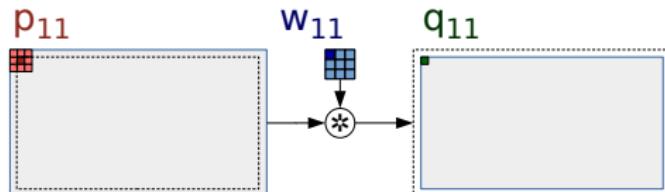
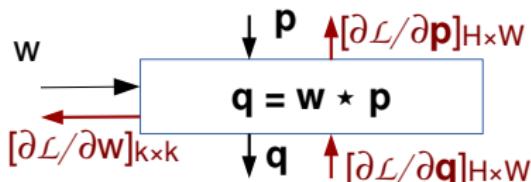
Unatražni prolaz širi $\partial \mathcal{L} / \partial q_i$ kao kod globalnog sažimanja (onehot_1^n) te rezultat **ugrađuje** u $\partial \mathcal{L} / \partial \mathbf{p}$ primjenom funkcije embed_k^n :

$$\begin{aligned}\partial \mathcal{L} / \partial \mathbf{p} &= \sum_{i=0}^{\lfloor N/k \rfloor - 1} \partial \mathcal{L} / \partial q_i \cdot \text{embed}_k^N(\partial q_i / \partial \mathbf{p}_{[k \cdot i : k \cdot i + k]}, k \cdot i) \\ &= \sum_{i=0}^{\lfloor N/k \rfloor - 1} \partial \mathcal{L} / \partial q_i \cdot \text{embed}_k^N(\text{onehot}_1^k(\arg \max \mathbf{p}_{[k \cdot i : k \cdot i + k]}), k \cdot i)\end{aligned}$$

SADRŽAJ

- ponavljanje: unatražno učenje, potpuno povezani slojevi
- gradijenti parametara 1D konvolucije
 - jezgra dimenzije 1
 - jezgra dimenzije k
- **gradijenti parametara 2D konvolucije**
 - jezgra dimenzija $k \times k$
 - zadatak s unutrašnjim prolazom kroz 2D konvoluciju
- 2D konvolucija nad tenzorima trećeg reda
- učenje konvolucije s korakom
- efikasna implementacija 2D konvolucije

MATRICE: JEZGRA KxK, NAPRIJED



Izlazne aktivacije 2D konvolucije odgovaraju **skalarnom produktu pravokutnog isječka ulaza** (crop) i jezgre:

$$q_{ij} = \text{sum}(\text{crop}_{k \times k}(\mathbf{p}, i, j) \odot \mathbf{w})$$

Raspišimo jednadžbu jedne aktivacije uz $o_k = \lfloor k/2 \rfloor + 1$:

$$q_{ij} = \text{sum}(\text{crop}_{k \times k}(\mathbf{p}, i, j) \odot \mathbf{w}) = \sum_{uv} \mathbf{p}_{i-o_k+u, j-o_k+v} \cdot \mathbf{w}_{uv} .$$

Npr. za $i=j=1$ i $k=3$ ($o_k=2$):

$$\mathbf{q}_{11} = \sum_{uv=1}^3 \mathbf{p}_{u-1, v-1} \cdot \mathbf{w}_{uv}$$

MATRICE: KXK, NATRAG

Dimenziije tenzora koji sudjeluju u unatražnom prolazu:

- ulaz (gradijent po izlazu): $\partial \mathcal{L} / \partial \mathbf{q}$ - $H \times W$
- izlaz (gradijent po ulazu): $\partial \mathcal{L} / \partial \mathbf{p}$ - $H \times W$
 - pretp: unaprijedni prolaz koristio nadopunjavanje
- izlaz (gradijent po parametrima): $[\partial \mathcal{L} / \partial \mathbf{w}]$ - $(k \times k)$

Specifičnost: izlazi \mathbf{q} ne ovise o svim ulazima \mathbf{p}

Opći oblik gradijenata po jednom pikselu ulaza:

$$\partial \mathcal{L} / \partial p_{ij} = \partial \mathcal{L} / \partial \text{vec}(\mathbf{q}) \cdot \partial \text{vec}(\mathbf{q}) / \partial p_{ij} = \sum_r \partial \mathcal{L} / \partial q_r \cdot \partial q_r / \partial p_{ij}$$

Ključna pitanja za gradijente po ulazima:

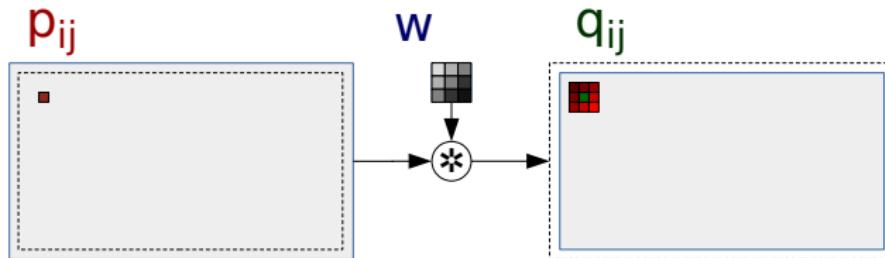
- koji q_r -ovi ovise o p_{ij} ?
- za koje r je $\partial q_r / \partial p_{ij} \neq 0$?

MATRICE: KXK, NATRAG, ULAZI

Prepostavimo da je u cijelom p -u upaljen samo jedan piksel: p_{ij}

Tada će u cijelom q biti upaljeni pikseli:

$$q_{i+s,j+t} = p_{ij} \cdot w_{o_k-s,o_k-t}, \text{ gdje su } s, t \in -\lfloor k/2 \rfloor .. \lfloor k/2 \rfloor$$



Označeni pikseli tenzora q : "zona utjecaja" piksela p_{ij}

- susjedstvo veličine $k \times k$ čije je središte q_{ij}
- za te piksele je $\partial q_r / \partial p_{ij} \neq 0!$
- vrijedi: $\partial q_{i+s,j+t} / \partial p_{ij} = w_{o_k-s,o_k-t}$
- donji desni element susjedstva - gornji lijevi element jezgre:
 - $\partial q_{i+1,j+1} / \partial p_{ij} = w_{1,1}$, uz $k=3$

MATRICE: KXK, NATRAG, ULAZI (2)

Što se zbiva kada upalimo sve aktivacije p?

- aktivacije q odgovaraju zbroju doprinosa svih p_{ij}
- to znači da gradijenti $\partial q_{i+s,j+t} / \partial p_{ij}$ ostaju isti.

Sada možemo zapisati gradijente po ulazu:

$$\begin{aligned}\partial \mathcal{L} / \partial p_{ij} &= \sum_{s,t} \partial \mathcal{L} / \partial q_{i+s,j+t} \cdot \partial q_{i+s,j+t} / \partial p_{ij} \\ &= \sum_{s,t} \partial \mathcal{L} / \partial q_{i+s,j+t} \cdot w_{o_k-s, o_k-t}\end{aligned}$$

Oni odgovaraju **skalarnom produktu** pravokutnog isječka (crop) gradijenata po izlazu i zrcaljene (flip2d) jezgre (kao i u 1D):

$$\partial \mathcal{L} / \partial p_{ij} = \text{sum}(\text{crop}_{k \times k}(\partial \mathcal{L} / \partial \mathbf{q}, i, j) \odot \text{flip2d}(\mathbf{w}))$$

MATRICE: KXK, NATRAG, ULAZI (3)

Vidjeli smo da se gradijenti po ulazu dobivaju klizanjem zrcaljene jezgre po gradijentima po izlazu:

$$\partial \mathcal{L} / \partial p_{ij} = \text{sum}(\text{crop}_{k \times k}(\partial \mathcal{L} / \partial \mathbf{q}, i, j) \odot \text{flip2d}(\mathbf{w}))$$

Sada je jasno da se gradijenti po ulazu dobivaju unakrsnom korelacijom gradijenta po izlazu sa zrcaljenom jezgrom $\text{flip2d}(\mathbf{w})$.

Ako želimo dobiti gradijente u svim pikselima ulaza, gradijente po izlazu moramo nadopuniti s $\lfloor k/2 \rfloor$ nula sa svake strane:

$$\partial \mathcal{L} / \partial \mathbf{p} = \text{flip2d}(\mathbf{w}) \star \text{pad}(\partial \mathcal{L} / \partial \mathbf{q}, \lfloor k/2 \rfloor)$$

Ako u unaprijednom prolazu nismo koristili nadopunjavanje, onda moramo nadopuniti s $k-1$ nula (sa svake strane)

MATRICE: KXK, TRANSPONIRANA KONVOLUCIJA

Posljednju operaciju nazivamo **transponiranom konvolucijom**

- konvoluirani tenzor se nadopuni $\lfloor k/2 \rfloor$ nula
- konvolucijska jezgra se zrcali po obje osi
- stariji naziv: dekonvolucija; bolje: unatražna konvolucija

Uvodimo novi operator: $\mathbf{w} \star^T \mathbf{x} \triangleq \text{flip2d}(\mathbf{w}) \star \text{pad}(\mathbf{x}, \lfloor k/2 \rfloor)$

Kasnije ćemo upoznati dodatna svojstva transponirane konvolucije

- kod tenzora trećeg reda, jezgru treba zrcaliti samo po prostornim osima
- može se koristiti i u unaprijednom prolazu

MATRICE: K×K, NATRAG, PARAMETRI

Ako ugasimo sve elemente jezgre osim w_{uv} , $u, v \in 1..k$, dobivamo:

$$q_{ij} = p_{i-o_k+u, j-o_k+v} \cdot w_{uv}, \forall i, j$$

Vidimo da parametri utječu na sve izlaze \rightarrow potrebno uzeti u obzir sve lokacije izlaznog tenzora:

$$\begin{aligned}\partial \mathcal{L} / \partial w_{uv} &= \sum_{ij} \partial \mathcal{L} / \partial q_{ij} \cdot \partial q_{ij} / \partial w_{uv} \\ &= \sum_{ij} \partial \mathcal{L} / \partial q_{ij} \cdot p_{i-o_k+u, j-o_k+v}\end{aligned}$$

Ovaj rezultat možemo interpretirati kao redukciju Hadamardovog (elementwise) produkta gradijenata izlaza i posmagnutih ulaza:

$$\begin{aligned}\partial \mathcal{L} / \partial w_{uv} &= \sum_{ij} \partial \mathcal{L} / \partial q_{ij} \cdot p_{i-o_k+u, j-o_k+v} \\ &= \text{sum}(\text{shift}(\mathbf{p}, -o_k + u, -o_k + v) \odot \partial \mathcal{L} / \partial \mathbf{q})\end{aligned}$$

MATRICE: K×K, NATRAG, PARAMETRI (2)

Kada upalimo sve piksele jezgre w :

- aktivacije q odgovaraju zbroju doprinosa svih w_{uv}
- to znači da gradijenti $\partial q_{ij} / \partial w_{uv}$ ostaju isti

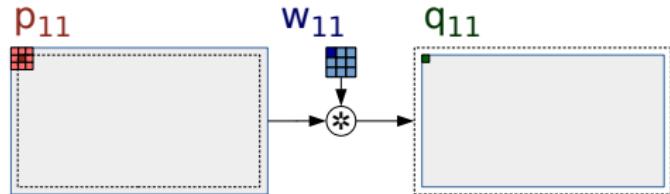
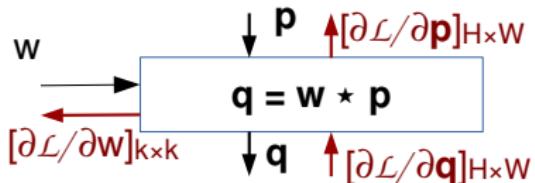
Gradijente po svim težinama sada možemo dobiti unakrsnom korelacijom gradijenta po izlazu s nadopunjениm ulazom:

$$\partial \mathcal{L} / \partial \mathbf{w} = \partial \mathcal{L} / \partial \mathbf{q} \star \text{pad}(\mathbf{p}, \lfloor k/2 \rfloor)$$

Prethodna jednadžba vrijedi ako izlaz i ulaz imaju istu rezoluciju zbog nadopunjavanja u unaprijednom prolazu.

Ako unaprijedni prolaz nije koristio nadopunjavanje, onda i gradijente po parametrima možemo izračunati bez nadopunjavanja.

MATRICE: $K \times K$, NATRAG, SAŽETAK



Unaprijedni prolaz:

$$q_{ij} = \text{sum}(\text{crop}_{k \times k}(\mathbf{p}, i, j) \odot \mathbf{w})$$

$$\mathbf{q} = \mathbf{w} * \mathbf{p}$$

Unatražni prolaz po parametrima:

$$\partial \mathcal{L} / \partial \mathbf{w} = \partial \mathcal{L} / \partial \mathbf{q} * \text{pad}(\mathbf{p}, \lfloor k/2 \rfloor)$$

Unatražni prolaz po ulazima:

$$\partial \mathcal{L} / \partial \mathbf{p} = \text{flip2d}(\mathbf{w}) * \text{pad}(\partial \mathcal{L} / \partial \mathbf{q}, \lfloor k/2 \rfloor)$$

MATRICE: ZADATAK

Razmatramo konvolucijski model za klasificiranje jednokanalne slike:

- konvolucija 3×3 bez pomaka i nadopunjavanja s aktivacijom ReLU
 - dvije izlazne mape značajki računaju jezgre w_1 i w_2
- globalno sažimanje maksimumom
- potpuno povezani sloj bez pomaka (W), softmaks

Inicijalizacija:

- $w_{121} = -1$, $w_{122} = 1$, svi ostali elementi w_1 su 0
- $w_{232} = -1$, $w_{222} = 1$, svi ostali elementi w_2 su 0
- $W_{11} = W_{22} = 1$, svi ostali elementi W su 0

Na ulazu je 4×4 matrica x , $x_{22} = 1$, $x_{33} = 1$, svi ostali elementi su 0

Odredite gradijente negativne log-izglednosti uz $y=2$

MATRICE: RJEŠENJE

```
import torch
import numpy as np

x = np.zeros([4,4])
x[1,1] = x[2,2] = 1
x = x.reshape([1,*x.shape])
x = torch.tensor(x, requires_grad=True)

w1 = np.zeros([3,3])
w1[1,0], w1[1,1] = -1,1
w1 = w1.reshape([1,1,*w1.shape])
w1 = torch.tensor(w1, requires_grad=True)

w2 = np.zeros([3,3])
w2[2,1], w2[1,1] = -1,1
w2 = w2.reshape([1,1,*w2.shape])
w2 = torch.tensor(w2, requires_grad=True)

W = torch.tensor(np.eye(2), requires_grad=True)
```

MATRICE: RJEŠENJE (2)

```
f1 = torch.nn.functional.conv2d(x, w1)
f1r = torch.nn.functional.relu(f1)
f1m = torch.nn.functional.max_pool2d(f1r, [2,2])

f2 = torch.nn.functional.conv2d(x, w2)
f2r = torch.nn.functional.relu(f2)
f2m = torch.nn.functional.max_pool2d(f2r, [2,2])

h = torch.concat([f1m, f2m]).squeeze()
s = W @ h

for t in [f1, f2, h, s]: t.retain_grad()
L = torch.nn.functional.cross_entropy(s, torch.tensor(1))
L.backward()

for t in [s, h, W, f1, f2, w1, w2]:
    print(t.data, t.grad.data, sep='\n', end='\n\n')
```

MATRICE: RJEŠENJE (3)

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, y = 2$$

$$\frac{\partial L}{\partial \mathbf{w}_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

$$\frac{\partial L}{\partial \mathbf{w}_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.5 & 0 \\ 0 & 0 & -0.5 \end{bmatrix}$$

Interpretacija: nakon koraka optimizacije grana 1 će davati slabiji a grana 2 jači odziv.

SADRŽAJ

- ponavljanje: unatražno učenje, potpuno povezani slojevi
- gradijenti parametara 1D konvolucije
 - jezgra dimenzije 1
 - jezgra dimenzije k
- gradijenti parametara 2D konvolucije
 - jezgra dimenzija $k \times k$
 - zadatak s unutrašnjim prolazom kroz 2D konvoluciju
- **2D konvolucija nad tensorima trećeg reda**
 - učenje konvolucije s korakom
 - efikasna implementacija 2D konvolucije

VIŠEKANALNI 2D SLUČAJ: NAPRIJED

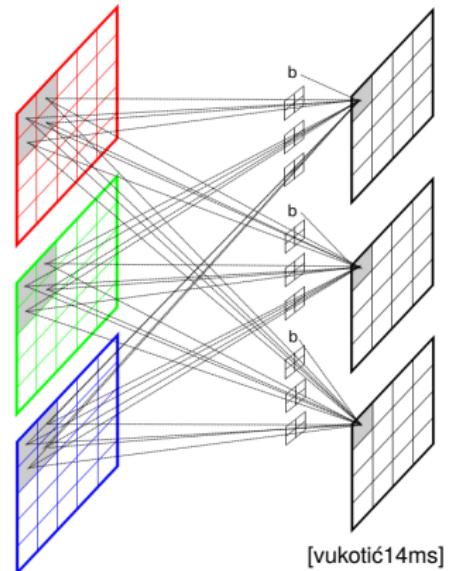
Zadržavamo standardnu sintaksu:

$$\mathbf{q} = \mathbf{w} * \mathbf{p} + \text{broadcast}(\mathbf{b})$$

Izlaz $\mathbf{q}^{(g)}$ zbraja konvolucije odgovarajućih kriški ulaza i g-te konvolucijske jezgre:

$$\mathbf{q}^{(g)} = \sum_f \mathbf{w}^{(g,f)} * \mathbf{p}^{(f)}$$

$$q_{ij}^{(g)} = \sum_{fuv} p_{i-o_k+u, j-o_k+v}^{(f)} \cdot w_{uv}^{(g,f)}$$



Gradijenti pomaka odgovaraju zbroju gradijenata izlaza:

$$\partial \mathcal{L} / \partial b_g = \sum_{ij} \partial \mathcal{L} / \partial q_{ij}^{(g)} \cdot \partial q_{ij}^{(g)} / \partial b_g = \sum_{ij} \partial \mathcal{L} / \partial q_{ij}^{(g)}$$

VIŠEKANALNI 2D SLUČAJ: NATRAG, ULAZI

Ulas $\mathbf{p}^{(f)}$ utječe na svaku krišku $\mathbf{q}^{(g)}$ preko konvolucije \mathbf{s} $\mathbf{w}^{(g,f)}$

Stoga, gradijenti po f-toj kriški ulaza zbrajaju gradijente iz svih $\mathbf{q}^{(g)}$:

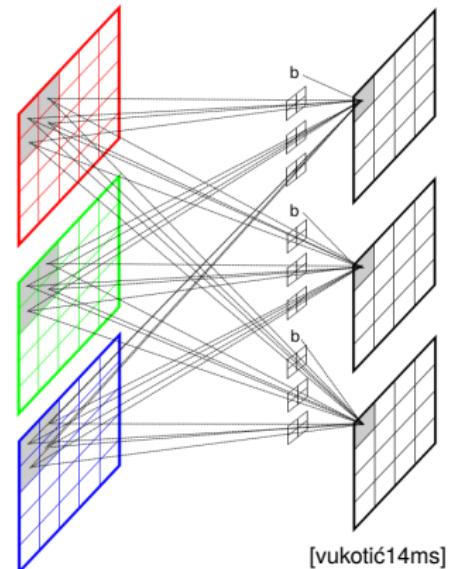
$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}_{ij}^{(f)}} = \sum_g \sum_{s,t} \frac{\partial \mathcal{L}}{\partial \mathbf{q}_{i+s,j+t}^{(g)}} \cdot w_{o_k-s, o_k-t}^{(g,f)}$$

Kao i ranije, te gradijente možemo izraziti konvolucijom sa zrcaljenom jezgrom:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}^{(f)}} = \sum_g \text{flip2d}(\mathbf{w}^{(g,f)}) \star \text{pad}(\frac{\partial \mathcal{L}}{\partial \mathbf{q}^{(g)}}, \lfloor k/2 \rfloor)$$

Posljednja operacija odgovara **transponiranoj konvoluciji** za 3D tenzore:

$$\mathbf{w} \star^\top \mathbf{x} \triangleq \sum_g \text{flip2d}(\mathbf{w}^{(g,f)}) \star \text{pad}(\mathbf{x}^{(g)}, \lfloor k/2 \rfloor)$$



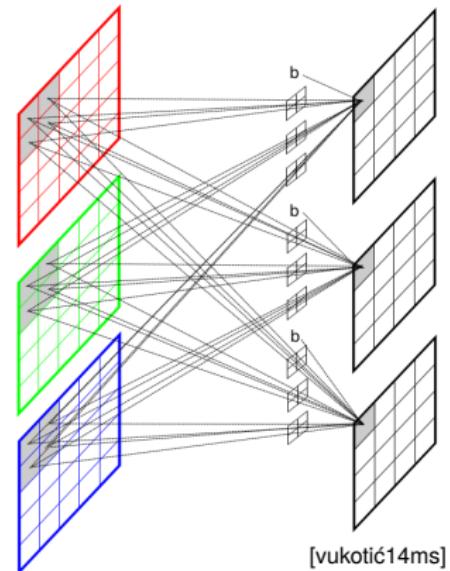
[vukotić14ms]

VIŠEKANALNI 2D SLUČAJ: NATRAG, PARAMETRI

Izlazni tenzor zbraja doprinose odgovarajućih kriški jezgre i ulaza

- f-ta kriška g-te jezgre utječe na g-tu krišku izlaza konvolucijom sa f-tom kriškom ulaza

Zato gradijente po parametrima računamo odvojeno za svaku krišku jezgre



[vukotić14ms]

Konačni izraz je potpuno isti kao i kod 2D tenzora:

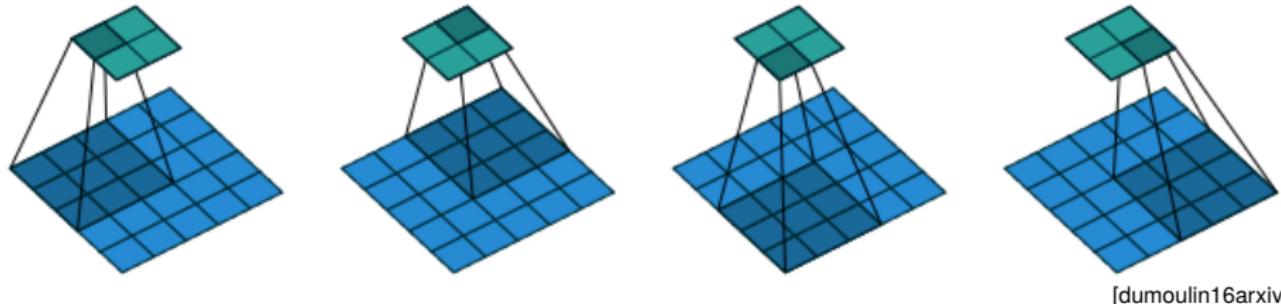
$$\partial \mathcal{L} / \partial \mathbf{w}^{(g,f)} = \partial \mathcal{L} / \partial \mathbf{q}^{(g)} * \text{pad}(\mathbf{p}^{(f)}, \lfloor k/2 \rfloor)$$

SADRŽAJ

- ponavljanje: unatražno učenje, potpuno povezani slojevi
- gradijenti parametara 1D konvolucije
 - jezgra dimenzije 1
 - jezgra dimenzije k
- gradijenti parametara 2D konvolucije
 - jezgra dimenzija $k \times k$
 - zadatak s unutrašnjim prolazom kroz 2D konvoluciju
- 2D konvolucija nad tenzorima trećeg reda
- **učenje konvolucije s korakom**
- efikasna implementacija 2D konvolucije

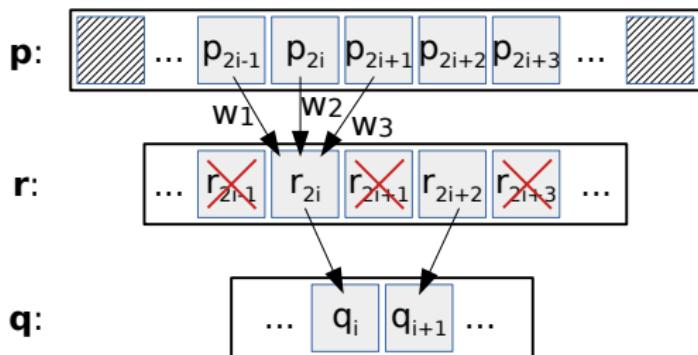
KORAK: NAPRIJED

Konvolucija s izlaznim korakom s ostavlja svaki s -ti element izlaza



[dumoulin16arxiv]

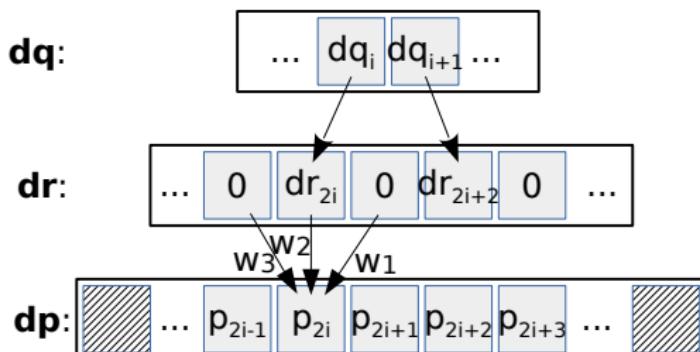
Kao da smo nakon standardne konvolucije proveli poduzorkovanje $\times s$:



KORAK: NATRAG ULANČAVANJEM

Vraćamo se kroz konvoluciju i poduzorkovanje $\times s$:

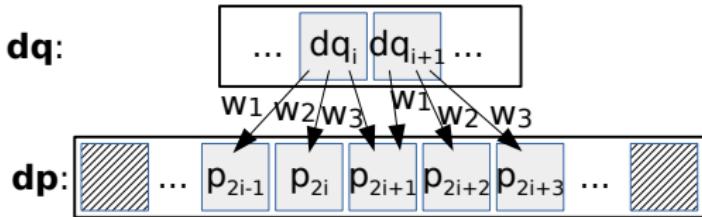
- nakon što gradijenti prođu kroz poduzorkovanje, gradijent svakog preskočenog elementa izlaza konvolucije postaje nula
- nastavljamo normalnim unutrašnjim prolazom kroz konvoluciju.
- efektivno, receptivno polje transformacije $\partial \mathcal{L} / \partial q \rightarrow \partial \mathcal{L} / \partial p$ za $s \times s$ je manje od receptivnog polja konvolucije s istom jezgrom
 - zato ovu operaciju neki zovu frakcijska konvolucija (Theano)



KORAK: NATRAG IZRAVNO

Množenje s nulama možemo izbjegići akumuliranjem doprinosa izlaza:

- višestruko pisanje na istu lokaciju je loše za paralelnu izvedbu, ali ovdje može biti metoda izbora



Ovakva operacija ne može se izraziti konvolucijom

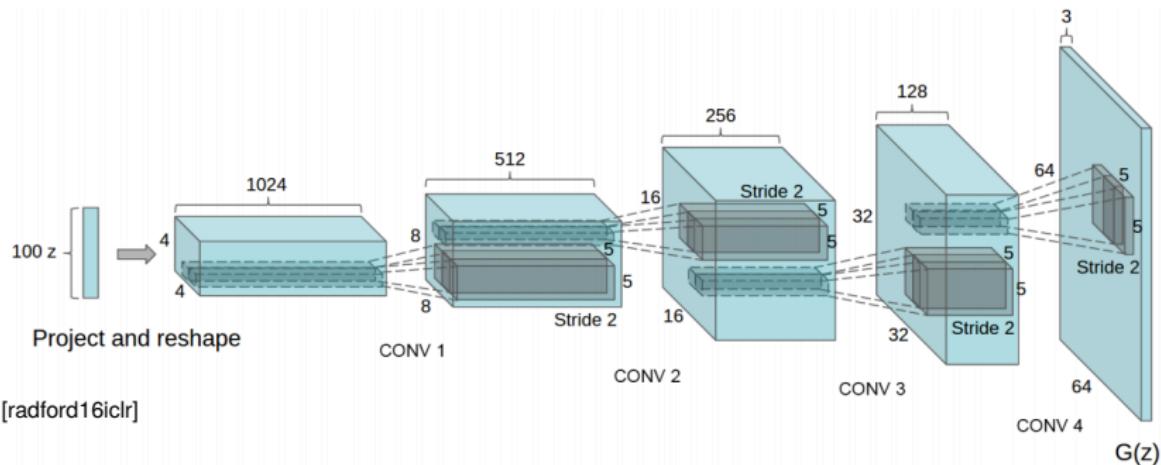
- ⇒ izvodimo je posebnim algoritmom

$$\begin{aligned}\partial \mathcal{L} / \partial \mathbf{p}^{(f)} &= \sum_{ij} \sum_g \partial \mathcal{L} / \partial q_{ij}^{(g)} \cdot \partial q_{ij}^{(g)} / \partial \mathbf{p}^{(f)} \\ &= \sum_{ij=0,0}^{\text{shape}(\mathbf{q})} \sum_g \text{embed}_{k \times k}^{H \times W} (\partial \mathcal{L} / \partial q_{ij}^{(g)} \cdot \mathbf{w}^{(g,f)}, [s \cdot i, s \cdot j])\end{aligned}$$

KORAK: NATRAG IZRAVNO (2)

Transponirana konvolucija s korakom u unaprijednom prolazu:

- standardni štos za povećati rezoluciju reprezentacije
- za raspoznavanje malih slika (npr. CIFAR)
- kod generativnih modela s niskodimenzionalnim latentnim prostorom (npr. DCGAN, VAE)



KORAK: ALTERNATIVA

Rezoluciju latentne reprezentacije možemo povećati i bilinearnim naduzorkovanjem nakon čega tipično ide normalna konvolucija:

- učinak je vrlo sličan transponiranoj konvoluciji s korakom
- razlika: rupe u naduzorkovanom ulazu pune se intepoliranim značajkama (umjesto nulama)
- veća složenost ali ista brzina u praksi
- nama je takav postupak radio bolje [kreso17iccvw]

SADRŽAJ

- ponavljanje: unatražno učenje, potpuno povezani slojevi
- gradijenti parametara 1D konvolucije
 - jezgra dimenzije 1
 - jezgra dimenzije k
- gradijenti parametara 2D konvolucije
 - jezgra dimenzija $k \times k$
 - zadatak s unutrašnjim prolazom kroz 2D konvoluciju
- učenje konvolucije s korakom
- 2D konvolucija nad tenzorima trećeg reda
- **efikasna implementacija 2D konvolucije**

IMPLEMENTACIJA: UVOD

Ideja: predstaviti konvoluciju kao matrični umnožak

- kapitalizirati GEMM za efikasnu evaluaciju

Dvije izvedbe ideje:

- izravnati ulazne i izlazne aktivacije (loša ideja)
- izravnati težine konvolucijske jezgre (dobra ideja)

Novije implementacije koriste Winogradov konvolucijski algoritam

- ukupni broj množenja više nego dvostruko manji
- Nervana [lavin15arxiv], cudnn v5.1+

IMPLEMENTACIJA: RAVNANJE AKTIVACIJA

$$\mathbf{q}_{3 \times 3} = \mathbf{w}_{3 \times 3} * \mathbf{p}_{5 \times 5}$$

$$\text{vec}(\mathbf{q}_{3 \times 3}) = \text{doubly_circ}(\mathbf{w}_{3 \times 3}) \cdot \text{vec}(\mathbf{p}_{5 \times 5})$$

$$\begin{bmatrix} q_{11} \\ q_{12} \\ q_{13} \\ q_{21} \\ q_{22} \\ q_{23} \\ q_{31} \\ q_{32} \\ q_{33} \end{bmatrix} = \left[\begin{array}{cccccccccccccccccccccccccccc} w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & w_{11} & w_{12} & w_{13} & . & . & w_{21} & w_{22} & w_{23} & . & . & w_{31} & w_{32} & w_{33} & . & . & . & . & . & . & . \end{array} \right] \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{15} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{25} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \\ p_{35} \\ p_{41} \\ p_{42} \\ p_{43} \\ p_{44} \\ p_{51} \\ p_{52} \\ p_{53} \\ p_{54} \\ p_{55} \end{bmatrix}$$

Vremenska složenost: $O(W^2 H^2)$ vs $O(WHk^2)$

IMPLEMENTACIJA: RAVNANJE JEZGRI

Ideja: presložiti ulaz da konvolucija odgovara matričnom množenju:

$$\text{vec}(\mathbf{q}) = \text{im2row}(\mathbf{p}) \cdot \text{vec}(\mathbf{w})$$

Na slici dolje X odgovara p , dok X^{ROWS} odgovara $\text{im2row}(p)$:

x

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

X ROWS

x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}
x_{12}	x_{13}	x_{14}	x_{22}	x_{23}	x_{24}	x_{32}	x_{33}	x_{34}
x_{13}	x_{14}	x_{15}	x_{23}	x_{24}	x_{25}	x_{33}	x_{34}	x_{35}
x_{33}	x_{34}	x_{35}	x_{43}	x_{44}	x_{45}	x_{53}	x_{54}	x_{55}

Vremenska složenost: $O(WHk^2)$ vs $O(WHk^2)$

Prostorna složenost: $O(WHk^2)$ vs $O(WH)$

IMPLEMENTACIJA: RAVNANJE JEZGRI

Ideja: presložiti ulaz da konvolucija odgovara matričnom množenju:

$$\text{vec}(\mathbf{q}) = \text{im2row}(\mathbf{p}) \cdot \text{vec}(\mathbf{w})$$

Na slici dolje \mathbf{X} odgovara \mathbf{p} , dok \mathbf{X}^{ROWS} odgovara $\text{im2row}(\mathbf{p})$:

\mathbf{X}

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

\mathbf{X}^{ROWS}

x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}
x_{12}	x_{13}	x_{14}	x_{22}	x_{23}	x_{24}	x_{32}	x_{33}	x_{34}
x_{13}	x_{14}	x_{15}	x_{23}	x_{24}	x_{25}	x_{33}	x_{34}	x_{35}
x_{33}	x_{34}	x_{35}	x_{43}	x_{44}	x_{45}	x_{53}	x_{54}	x_{55}

Vremenska složenost: $O(WHk^2)$ vs $O(WHk^2)$

Prostorna složenost: $O(WHk^2)$ vs $O(WH)$

IMPLEMENTACIJA: RAVNANJE JEZGRI

Ideja: presložiti ulaz da konvolucija odgovara matričnom množenju:

$$\text{vec}(\mathbf{q}) = \text{im2row}(\mathbf{p}) \cdot \text{vec}(\mathbf{w})$$

Na slici dolje \mathbf{X} odgovara \mathbf{p} , dok \mathbf{X}^{ROWS} odgovara $\text{im2row}(\mathbf{p})$:

\mathbf{X}

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

\mathbf{X}^{ROWS}

x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}
x_{12}	x_{13}	x_{14}	x_{22}	x_{23}	x_{24}	x_{32}	x_{33}	x_{34}
x_{13}	x_{14}	x_{15}	x_{23}	x_{24}	x_{25}	x_{33}	x_{34}	x_{35}
x_{33}	x_{34}	x_{35}	x_{43}	x_{44}	x_{45}	x_{53}	x_{54}	x_{55}

Vremenska složenost: $O(WHk^2)$ vs $O(WHk^2)$

Prostorna složenost: $O(WHk^2)$ vs $O(WH)$

IMPLEMENTACIJA: RAVNANJE JEZGRI

Ideja: presložiti ulaz da konvolucija odgovara matričnom množenju:

$$\text{vec}(\mathbf{q}) = \text{im2row}(\mathbf{p}) \cdot \text{vec}(\mathbf{w})$$

Na slici dolje \mathbf{X} odgovara \mathbf{p} , dok \mathbf{X}^{ROWS} odgovara $\text{im2row}(\mathbf{p})$:

\mathbf{X}

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

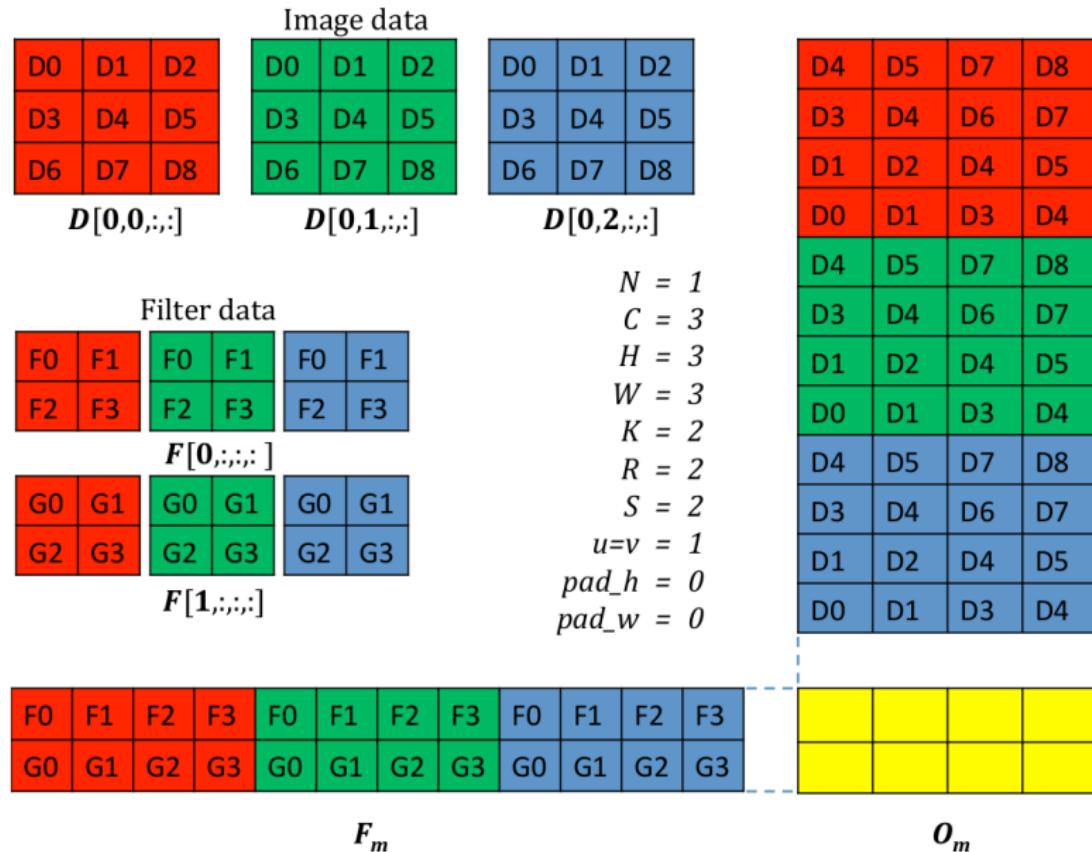
\mathbf{X}^{ROWS}

x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}
x_{12}	x_{13}	x_{14}	x_{22}	x_{23}	x_{24}	x_{32}	x_{33}	x_{34}
x_{13}	x_{14}	x_{15}	x_{23}	x_{24}	x_{25}	x_{33}	x_{34}	x_{35}
x_{33}	x_{34}	x_{35}	x_{43}	x_{44}	x_{45}	x_{53}	x_{54}	x_{55}

Vremenska složenost: $O(WHk^2)$ vs $O(WHk^2)$

Prostorna složenost: $O(WHk^2)$ vs $O(WH)$

IMPLEMENTACIJA: RAVNANJE JEZGRI (TENZORI 3. REDA)



IMPLEMENTACIJA: NATRAG - PARAMETRI

Podsjetimo se, konvoluciju smo izrazili matričnim produktom:

$$\text{vec}(\mathbf{q}) = \text{im2row}(\mathbf{p}) \cdot \text{vec}(\mathbf{w})$$

Odatle se jasno vidi da mora biti:

$$\frac{\partial \text{vec}(\mathbf{q})}{\partial \text{vec}(\mathbf{w})} = \text{im2row}(\mathbf{p})$$

Konačno, gradijente funkcije gubitka s obzirom na parametre (konvolucijsku jezgru) računamo kao:

$$\begin{aligned}\partial \mathcal{L} / \partial \text{vec}(\mathbf{w}) &= \partial \mathcal{L} / \partial \text{vec}(\mathbf{q}) \cdot \partial \text{vec}(\mathbf{q}) / \partial \text{vec}(\mathbf{w}) \\ &= \partial \mathcal{L} / \partial \text{vec}(\mathbf{q}) \cdot \text{im2row}(\mathbf{p})\end{aligned}$$

IMPLEMENTACIJA: NATRAG - ULAZI

Podsjetimo se kako izgledaju gradijenti po ulazu:

$$\partial \mathcal{L} / \partial \mathbf{p} = \text{flip2d}(\mathbf{w}) \star \text{pad}(\partial \mathcal{L} / \partial \mathbf{q}, \lfloor k/2 \rfloor)$$

Gradijente izlaza treba presložiti u konvolucijsku matricu $\mathbf{G}_q^{\text{ROWS}}$:

$$\mathbf{G}_q^{\text{ROWS}} = \text{im2row}(\text{pad}(\partial \mathcal{L} / \partial \mathbf{q}, \lfloor k/2 \rfloor))$$

Konvoluciju sada možemo provesti matričnim umnoškom:

$$\mathbf{G}_p^{\text{VEC}} = \mathbf{G}_q^{\text{ROWS}} \cdot \text{vec}(\text{flip2d}(\mathbf{w}))$$

Konačni rezultat dobivamo preoblikovanjem $\mathbf{G}_p^{\text{VEC}}$:

$$\partial \mathcal{L} / \partial \mathbf{p} = \text{reshape}(\mathbf{G}_p^{\text{VEC}}, [H, W])$$

Ako smo u unaprijednom prolazu imali izlazni korak:

- $\mathbf{G}_q^{\text{ROWS}}$ je rijetka matrica
- bolju performansu postiže izravna transponirana konvolucija

ZAHVALA

Ova predavanja proizšla su iz istraživanja koje je finansirala Hrvatska zaklada za znanost projektom I-2433-2014 MultiCLoD.



<http://multiclod.zemris.fer.hr>