Metodologji praktike për Deep Learning

kapitull i plotë

zgjodhi dhe përktheu

Ridvan Bunjaku

maj 2017

Përmbajtja

Për publikimin

Ky publikim është përkthim i kapitullit 'Practical Methodology' nga libri "Deep Learning". Fillon me një përshkrim të metodologjisë së rekomanduar nga autorët dhe të procesit praktik të dizajnit. Pastaj shpjegohen hapat e procesit më detalisht. Në fund përshkruhet aplikimi i metodologjisë në shembullin real: sistemin e transkriptimit të numrave të adresave në Google Street View.

Publikimi mund të konsiderohet vazhdim i publikimit paraprak "Shembuj ilustrues nga Deep Learning" që është përmbledhje hyrëse informative.

Bibliografia

[1] I. Goodfellow, Y. Bengio dhe A. Courville, Deep Learning, MIT Press, 2016.

http://www.deeplearningbook.org

Falënderim

I falënderoj autorët për lejimin e publikimit:

"Ridvan,

I have no objection to you publishing online and for free a translation of parts of our book. For commercial ventures, it's another story and you would need to contact MIT Press.

-- Yoshua"

Licencimi

Online

Ky publikim mund të gjendet online në:

https://rbunjaku.wordpress.com/

Metodologjia praktike

Aplikimi i suksesshëm i teknikave të deep learning kërkon më shumë se vetëm njohuri të mirë të asaj se çfarë algoritme ekzistojnë dhe të principeve që e shpjegojnë se si funksionojnë ato. Një praktikuesi të mirë të machine learning i duhet të dijë edhe si ta zgjedhë një algoritëm për një aplikim të caktuar dhe si ta monitorojë dhe t'i përgjigjet feedback-ut që merret nga eksperimentet në mënyrë që ta përmirësojë një sistem të machine learning. Gjatë zhvillimit të përditshëm të sistemeve të machine learning, praktikuesit duhet të vendosin a të mbledhin më shumë shënime, a ta rrisin apo ta zvogëlojnë kapacitetin e modelit, ta përmirësojnë konkluzionin e përafërt (approximate inference) në një model, apo t'i rregullojnë defektet e implementimit softuerik të modelit (debug). Të gjitha këto operacione në rastin më të mirë marrin kohë për t'i provuar, prandaj është me rëndësi që, në vend të hamendësimit të verbër, të jeni të aftë ta përcaktoni drejtimin e duhur të veprimit.

Pjesa më e madhe e këtij libri flet për modelet e ndryshme të machine learning, algoritmet trajnuese, dhe funksionet objektive (objective functions të njohura edhe si loss functions, vër. e përkthyesit). Kjo mund ta japë përshtypjen se përbërësi më i rëndësishëm në të qenurit ekspert i machine learning është ta dini një mori të gjerë të teknikave të machine learning dhe të jeni të mirë në llojet e ndryshme të matematikës. Në praktikë, zakonisht personi mund t'ia dalë shumë më mirë me një aplikim korrekt të një algoritmi të rëndomtë se sa duke e aplikuar në mënyrë josistematike një algoritëm të mjegullt. Aplikimi korrekt i një algoritmi varet nga zotërimi i një metodologjie goxha të thjeshtë. Shumë nga rekomandimet në këtë kapitull janë adaptuar nga Ng (2015).

Ne e rekomandojmë procesin vijues praktik të dizajnit:

- Përcaktoni qëllimet tuaja—çfarë metrike e gabimit të përdoret, dhe vlerën tuaj target (cak) për këtë metrikë të gabimit. Këto qëllime dhe metrika të gabimit duhet të orientohen prej problemit të cilin po synon ta zgjidhë aplikacioni.
- Sa më parë etablojeni një proces funksional fillim-e-mbarim, duke e përfshirë edhe vlerësimin e metrikave të duhura të performansës.
- Aranzhojeni (instrumentojeni) sistemin mirë për t'i përcaktuar pikat e bllokimit (bottlenecks) në performansë. Diagnozoni se cilat komponenta po performojnë më keq se që pritet dhe a është kjo për shkak të mbipërputhjes (overfitting), nënpërputhjes (underfitting), apo ndonjë defekti në shënime apo softuer.
- Bëni ndryshime inkrementale (rritëse) siç është mbledhja e shënimeve të reja, rregullimi i hiperparametrave, apo ndryshimi i algoritmeve, bazuar në gjetjet specifike nga aranzhimi juaj.

Si shembull që funksionon, do ta përdorim sistemin e transkriptimit të numrave të adresave në Street View (Street View address number transcription system, Goodfellow *et al.* 2014d). Qëllimi i këtij aplikacioni është të shtohen ndërtesa në Google Maps. Veturat Street View i fotografojnë ndërtesat dhe i regjistrojnë koordinatat GPS të asociuara me secilën fotografi. Një rrjet konvolucional e njeh numrin e adresës në secilën fotografi, duke e lejuar bazën e shënimeve të Google Maps ta shtojë atë adresë në lokacionin korrekt. Storja se si është zhvilluar ky aplikacion e jep një shembull se si të ndiqet metodologjia e dizajnit të cilën e përkrahim.

Tani e përshkruajmë secilin nga hapat në këtë proces.

11.1 Metrikat e performansës

Përcaktimi i qëllimeve tuaja, në terma të asaj se cilën metrikë të gabimit ta përdorni, është hap i parë i nevojshëm sepse metrika juaj e gabimit do t'i drejtojë të gjitha veprimet tuaja të ardhshme. Do të duhej të keni edhe një ide se çfarë niveli të performansës dëshironi.

Mbani mend se për shumicën e aplikimeve, është e pamundur të arrihet gabim zero absolut. Gabimi i Bayes-it e definon normën minimale të gabimit (minimum error rate) të cilën mund të shpresoni ta arrini, madje edhe nëse keni shënime të pafundme të trajnimit dhe mund ta riktheni shpërndarjen e vërtetë të probabilitetit. Kjo sepse veçoritë tuaja hyrëse mund të mos përmbajnë informacion të plotë për ndryshoren dalëse, apo sepse sistemi mund të jetë brendësisht stohastik. Po ashtu do të jeni të kufizuar edhe nga posedimi i një sasie të fundme të shënimeve trajnuese.

Sasia e shënimeve trajnuese mund të jetë e kufizuar për një mori arsyesh. Kur qëllimi i juaj është ta ndërtoni produktin apo shërbimin më të mirë të mundshëm të botës reale, zakonisht mund të mblidhni më shumë shënime por duhet ta përcaktoni vlerën e zvogëlimit të mëtejmë të gabimit dhe ta peshoni këtë ndaj kostos së mbledhjes së më shumë shënimeve. Mbledhja e shënimeve mund të kërkojë kohë, para, apo vuajtje njerëzore (për shembull, nëse procesi i juaj i mbledhjes së shënimeve përfshin teste invazive mjekësore). Kur qëllimi juaj është t'i përgjigjeni një pyetjeje shkencore për atë se cili algoritëm performon më mirë në një benchmark të fiksuar, specifikimi i benchmark-ut zakonisht e përcakton setin e trajnimit dhe nuk ju lejohet të mblidhni më shumë shënime.

Si mund ta përcaktojë personi një nivel të arsyeshëm të pritshëm të performansës? Zakonisht, në aranzhim akademik, e kemi ndonjë vlerësim të normës së gabimit që është i arritshëm bazuar në rezultatet e benchmark-ave të publikuara më parë. Në aranzhimin e botës reale, e kemi një ide për normën e gabimit që nevojitet që një aplikacion të jetë i sigurt, efektiv për kosto, apo tërheqës për klientët. Pasi ta keni përcaktuar normën tuaj të dëshiruar realiste të gabimit, vendimet tuaja të dizajnit do të drejtohen prej arritjes së kësaj norme të gabimit.

Një konsideratë tjetër e rëndësishme përveç vlerës target të metrikës së performansës është zgjedhja se cilën metrikë ta përdorni. Për ta matur efektivitetin e një aplikimi të plotë që përfshin komponenta të machine learning mund të përdoren disa metrika të ndryshme të performansës. Këto metrika të performansës janë zakonisht të ndryshme nga funksioni i kostos që përdoret për ta trajnuar modelin. Siç përshkruhet në seksionin 5.1.2 (në libër), është e zakonshme të matet saktësia (accuracy), apo në mënyrë ekuivalente, norma e gabimit, e një sistemi.

Mirëpo, shumë aplikime kërkojnë metrika më të avansuara.

Nganjëherë është shumë më e kushtueshme ta bëni një lloj të gabimit se sa një tjetër. Për shembull, një sistem i detektimit të spam-it të email-ave mund t'i bëjë dy lloje të gabimeve: ta klasifikojë në mënyrë jokorrekte një mesazh legjitim si spam, dhe ta lejojë në mënyrë jokorrekte një mesazh spam në inbox. Është shumë më keq të bllokohet një mesazh legjitim se sa të lejohet të kalojë një mesazh i diskutueshëm. Në vend të matjes së normës së gabimit të një klasifikuesi të spam-it, ne mund të duam ta masim ndonjë

formë të kostos totale, ku kostoja e bllokimit të mesazheve legjitime është më e lartë se kostoja e lejimit të mesazheve spam.

Nganjëherë duam ta trajnojmë një klasifikues binar që synohet ta detektojë ndonjë ngjarje të rrallë. Për shembull, do të mund ta dizajnonim një test mjekësor për një sëmundje të rrallë. Supozojmë se vetëm një në një milion njerëz e ka këtë sëmundje. Lehtë mund të arrijmë saktësi 99.9999% në task-un e detektimit, thjesht duke e hard-koduar klasifikuesin që gjithmonë të raportojë se sëmundja mungon. Qartë, saktësia është mënyrë e dobët për ta karakterizuar perfomansën e një sistemi të tillë. Një mënyrë për ta zgjidhur këtë problem është që në vend të kësaj ta masim **precizitetin** (precision) dhe **recall**-in (rikthimi). Preciziteti është thyesa (pjesa) e detektimeve të raportuara nga modeli ndaj atyre që ishin korrekte, ndërsa recall-i është thyesa e ngjarjeve të vërteta ndaj atyre që ishin detektuar. Një detektor që thotë se askush nuk e ka sëmundjen do të arrinte precizitet perfekt, mirëpo recall zero. Një detektor që thotë se secili e ka sëmundjen do të arrinte recall perfekt, mirëpo precizitet të barabartë me përqindjen e njeërzve që e kanë sëmundjen (0.0001% në shembullin tonë të një sëmundjeje ku vetëm një person në një milion e ka). Kur përdoret preciziteti dhe recall-i, është e zakonshme të vizatohet një **lakore PR** (PR curve), ku preciziteti është në boshtin y dhe recall-i në boshtin x. Klasifikuesi e prodhon një rezultat (score) që është më i lartë nëse ka ndodhur ngjarja që duhet të detektohet. Për shembull, një rrjet feedforward i dizajnuar ta detektojë një sëmundje jep si rezultat dalës $\hat{y} = P(y = 1 \mid \boldsymbol{x})$, duke e vlerësuar gjasën që një person, rezultatet mjekësore të të cilit janë përshkruar nga veçoritë $\boldsymbol{x}$, e ka sëmundjen. Ne zgjedhim ta raportojmë një detektim sa herë që ky rezultat e tejkalon ndonjë prag. Duke e ndryshuar pragun, mund ta tregtojmë precizitetin me recall. Në shumë raste, performansën e klasifikuesit duam ta përmbledhim me një numër në vend se me një lakore. Për ta bërë këtë, mund ta konvertojmë prezicizetin i dhe recall-in r në një **F-score** (rezultat F) të dhënë me

$$F = \frac{2pr}{p + r}. \tag{11.1}$$

Një opsion tjetër është ta raportojmë sipërfaqen totale që është nën lakoren PR.

Në disa aplikime është e mundur që sistemi machine learning të refuzojë të marrë vendim. Kjo është e dobishme kur algoritmi machine learning mund ta vlerësojë se sa konfident duhet të jetë për një vendim, sidomos nëse një vendim i gabuar mund të jetë i dëmshëm dhe nëse një operator njerëzor është i aftë ta marrë kontrollin kohë pas kohe. Sistemi i transkriptimit Street View e ofron një shembull të kësaj situate. Task-u (detyra) është të transkriptohet numri i adresës nga një fotografi ashtu që të asociohet lokacioni ku është marrë foto me adresën korrekte në hartë. Meqë vlera e hartës degradohet në mënyrë të konsiderueshme nëse harta është e pasaktë, është me rëndësi të shtohet një adresë vetëm nëse transkriptimi është korrekt. Nëse sistemi machine learning mendon se ai ka më pak gjasa sesa një qenie njerëzore që ta nxjerrë transkriptimin korrekt, atëherë drejtimi më i mirë i veprimit është të lejohet një njeri që ta transkriptojë foton në vend të sistemit. Natyrisht, sistemi machine learning është i dobishëm vetëm nëse është i aftë që ta zvogëlojë dramatikisht sasinë e fotove që duhet t'i procesojnë operatorët njerëz. Një metrikë natyrale e performansës për ta përdorur në këtë situatë është **mbulimi** (coverage). Mbulimi është pjesa (thyesa) e shembujve për të cilët sistemi machine learning është i aftë ta prodhojë një përgjigje. Ka mundësi të tregtohet mbulimi me saktësi. Personi

mund të fitojë gjithmonë 100% saktësi duke mos e procesuar asnjë shembull, mirëpo kjo e zvogëlon mbulimin në 0%. Për task-un Street View, qëllimi për projektin ishte të arrihet saktësi e transkriptimit në nivel njerëzor duke mbajtur mbulim 95%. Performansa e nivelit njerëzor në këtë task është 98% saktësi.

Janë të mundura edhe shumë metrika të tjera. Për shembull, mund t'i masim normat e click-through, të mbledhim anketa të kënaqësisë së përdoruesve, e kështu me radhë. Shumë fusha të specializuara të aplikimit kanë edhe kritere specifike të aplikimit.

E rëndësishme është të përcaktohet se cilën metrikë të performansës ta përmirësojmë para kohe, e pastaj të përqendrohemi në përmirësimin e kësaj metrike. Pa qëllime të definuara qartë, mund të jetë e vështirë të thuhet se a bëjnë progres apo jo ndryshimet në një sistem machine learning.

11.2 Modelet default baseline

Pas zgjedhjes së metrikave të performansës dhe qëllimeve, hapi tjetër në çdo aplikim praktik është të etablohet sa më parë një sistem i arsyeshëm fillim-e-mbarim. Në këtë pjesë, ofrojmë rekomandime për atë se cilat algoritme të përdoren si qasje e parë baseline në situata të ndryshme. Mbani mend se kërkimi në deep learning përparon shpejt, prandaj me gjasë së shpejti pas këtij shkrimi do të ketë në dispozicion algoritme më të mira default.

Varësisht nga kompleksiteti i problemit tuaj, ju madje mund të doni të filloni pa përdorur deep learning fare. Nëse problemi i juaj ka shans të zgjidhet thjesht duke i zgjedhur disa pesha lineare në mënyrë korrekte, mund të doni të filloni me një model të thjeshtë statistik si regresioni logjistik.

Nëse e dini se problemi i juaj bie në një kategori "AI-komplete" siç është object recognition (njohja e objektit), speech recognition (njohja e të folurit), machine translation (përkthimi nga makina), e kështu me radhë, atë herë me gjasë do të bëni mirë të filloni me një model të duhur të deep learning.

Së pari, zgjedheni kategorinë e përgjithshme të modelit bazuar në strukturën e shënimeve tuaja. Nëse doni të bëni supervised learning (mësim të mbikëqyrur) me vektorë me madhësi fikse si hyrje, përdoreni një rrjet feedforward me shtresa plotësisht të lidhura. Nëse hyrja ka strukturë të njohur topologjike (për shembull, nëse hyrja është imazh), përdoreni një rrjet konvolucional. Në këto raste, duhet të filloni duke e përdorur ndonjë lloj të piecewise linear unit (njësisë lineare në nivel të pjesëve) (ReLU-të apo përgjithësimet e tyre si Leaky ReLU-të, PreLu-të dhe maxout). Nëse hyrja dhe dalja juaj është sekuencë, atëherë përdoreni një gated recurrent net (LSTM apo GRU).

Një zgjedhje e arsyeshme e algoritmit të optimizimit është SGD (Stochastic Gradient Descent, ulja stohastike e gradientit) me momentum, me një normë dobësuese të mësimit (skema të njohura të dobësimit që performojnë më mirë apo më keq në probleme të ndryshme përfshijnë dobësim linear deri kur të arrihet një normë e fiksuar minimale e mësimit, dobësimin eksponencial, apo zvogëlimin e normës së mësimit për një faktor prej 2-10 sa herë që gabimi i validimit arrin plato – ku nuk ka rritje apo ngritje). Një

alternativë shumë e arsyeshme është Adam. Normalizimi i batch-it mund të ketë efekt dramatik në performansën e optimizmit, sidomos për rrjetet konvolucionale dhe rrjetet me jolinearitete sigmoidale. Megjithëse është e arsyeshme ta heqim normalizimin e batch-it nga baseline-i i parë, ai duhet të futet shpejt nëse optimizimi duket të jetë problematik.

Përveç në rastet kur seti juaj i trajnimit përmban dhjetëra miliona shembuj apo më shumë, duhet të përfshini disa forma të buta të regularizimit që nga fillimi. Early stopping (ndalja e hershme) duhet të përdoret pothuaj universalisht. Dropout (braktisja) është regularizues i shkëlqyeshëm që është lehtë të implementohet dhe kompatibil me shumë modele dhe algoritme të trajnimit. Edhe normalizimi i batch-it nganjëherë e zvogëlon gabimin e përgjithësimit dhe lejon të hiqet droput-it, për shkak të zhurmës në estimimin (vlerësimin) e statistikës që përdoret për ta normalizuar secilën variabël.

Nëse task-u juaj është i ngjashëm me një task tjetër që është studiuar gjerësisht, me gjasë do të bëni mirë që së pari ta kopjoni modelin dhe algoritmin që tashmë dihet se performon më së miri në task-un e studiuar më herët. Madje mund të doni ta kopjoni edhe një model të trajnuar nga ai task. Për shembull, është e zakonshme të përdoren features (veçoritë) nga një rrjet konvolucional i trajnuar në ImageNet për të zgjidhur task-a të tjerë të computer vision (shikimit kompjuterik) (Girshick *et al.,* 2015).

Një pyetje e zakonshme është a të fillojmë duke përdorur unsupervised learning (mësim të pambikëqyrur), që përshkruhet tutje në pjesën III (në libër). Kjo është disi specifike për fushën. Disa fusha, siç është natural language processing (procesimi i gjuhës natyrale), dihet se përfitojnë jashtëzakonisht nga teknikat e unsupervised learning siç është learning unsupervised word embeddings. Në fusha të tjera, siç është computer vision, teknikat aktuale të mësimit të pambikëqyrur nuk sjellin benefit, pos në aranzhim gjysmë-të-mbikëqyrur, ku numri i shembujve të etiketuar është shumë i vogël (Kingma *et al.*, 2014; Rasmus *et al.*, 2015). Nëse aplikimi juaj është në një kontekst ku unsupervised learning dihet të jetë i rëndësishëm, atëherë përfshijeni atë në baseline-in tuaj të parë fillim-e-mbarim. Përndryshe, mësimin e pambikëqyrur (unsupervised learning) përdoreni në tentimin tuaj të parë vetëm nëse task-u që doni ta zgjidhni është i pambikëqyrur. Gjithmonë mund të provoni ta shtoni mësimin e pambikëqyrur më vonë nëse vëreni se baseline-i i juaj fillestar mbipërputhet (overfits).

11.3 Përcaktimi se a të mbledhim ende shënime

Pasi të jetë etabluar sistemi i parë fillim-e-mbarim, është koha ta masim performansën e algoritmit dhe ta përcaktojmë se si ta përmirësojmë atë. Shumë fillestarë në machine learning joshen të bëjnë përmirësime duke provuar shumë algoritme të ndryshme. Mirëpo, shpesh është më mirë të mbledhim më shumë shënime se sa ta përmirësojmë algoritmin e mësimit.

Si vendos personi se a të mbledhë më shumë shënime? Së pari, përcaktojeni se a është e pranueshme performansa në setin e trajnimit. Nëse performansa në setin e trajnimit është e dobët, algoritmi i mësimit nuk po i përdor shënimet e trajnimit që tashmë janë në dispozicion, prandaj nuk ka arsye të mbledhim më shumë shënime. Në vend të kësaj, provoni ta rritni madhësinë e modelit duke shtuar më shumë shtresa apo duke i shtuar

njësi të fshehura secilës shtresë. Po ashtu, provoni ta përmirësoni algoritmin e mësimit, për shembull duke e rregulluar hiperparametrin e normës së mësimit. Nëse modelet e mëdha dhe algoritmet e optimizimit të rregulluara me kujdes nuk punojnë mirë, atëherë problemi mund të jetë *cilësia* e shënimeve trajnuese. Shënimet mund të jenë tepër të zhurmshme apo mund të mos i përfshijnë hyrjet e duhura që nevojiten për t'i parashikuar daljet e dëshiruara. Kjo sugjeron të fillohet përsëri nga fillimi, duke mbledhur shënime më të pastra apo duke e mbledhur një set më të pastër të veçorive.

Nëse performansa në setin e trajnimit është e pranueshme, atëherë mateni performansën në setin testues. Nëse edhe performansa në setin testues është e pranueshme, atëherë nuk ka mbetur asgjë për t'u bërë. Nëse performansa e setit testues është shumë më e keqe se sa performansa e setit trajnues, atëherë mbledhja e më shumë shënimeve është njëra nga zgjidhjet më efektive. Konsideratat kryesore janë kostoja dhe fizibiliteti apo realizueshmëria e mbledhjes së më shumë shënimeve, kostoja dhe fizibiliteti i zvogëlimit të gabimit të testimit me mënyra tjera, dhe sasia e shënimeve që pritet të jenë të nevojshme për ta përmirësuar dukshëm performansën e setit testues. Në kompani të mëdha të internetit me miliona apo miliarda përdorues, është e realizueshme të mblidhen dataseta të mëdhenj, dhe shpenzimi i kësaj mund të jetë në mënyrë të konsiderueshme më pak se sa alternativat tjera, prandaj përgjigja është pothuajse gjithmonë të mblidhen më shumë shënime. Për shembull, zhvillimi i dataset-ave të etiketuar të mëdhenj ishte njëri ndër faktorët më të rëndësishëm në zgjidhjen e njohjes së objekteve. Në kontekste të tjera, siç janë aplikimet mjekësore, mund të jetë e kushtueshme apo e parealizueshme të mblidhen më shumë shënime. Një alternativë e thjeshtë ndaj mbledhjes së më shumë shënimeve është të zvogëlohet madhësia e modelit apo të përmirësohet regularizimi, duke i rregulluar hiperparametrat siç janë koeficientët e dobësimit të peshave (weight decay coefficients), apo duke shtuar strategji të regularizimit siç është dropout (braktisja). Nëse e gjeni se gap-i (hendeku) ndërmjet performansës së trajnimit dhe të testit është ende i papranueshëm edhe pas rregullimit të hiperparametrave të regularizimit, atëherë është e këshillueshme mbledhja e më shumë shënimeve.

Kur vendosni se a të mblidhni më shumë shënime, nevojitet edhe të vendosni se sa të mblidhni. Është e dobishme të vizatoni lakore që e tregojnë lidhjen ndërmjet madhësisë së setit të trajnimit dhe gabimit të përgjithësimit, si në figurën 5.4. Duke ekstrapoluar lakore të tilla, personi mund ta parashikojë se sa shënime shtesë të trajnimit do të nevojiteshin për ta arritur një nivel të caktuar të performansës. Zakonisht, shtimi i një pjese të vogël të numrit total të shembujve nuk do të ketë ndikim të dukshëm në gabimin e përgjithësimit. Prandaj rekomandohet të eksperimentohet me madhësi të setit të trajnimit në shkallë logaritmike, për shembull duke e dyfishuar numrin e shembujve ndërmjet eksperimenteve të njëpasnjëshme.

Nëse mbledhja e shumë më shumë shënimeve nuk është e realizueshme, e vetmja mënyrë tjetër për ta përmirësuar gabimin e përgjithësimit është të përmirësohet vetë algoritmi i mësimit. Kjo kalon në domen të hulumtimit dhe jo në domen të këshillës për praktikuesit aplikues.

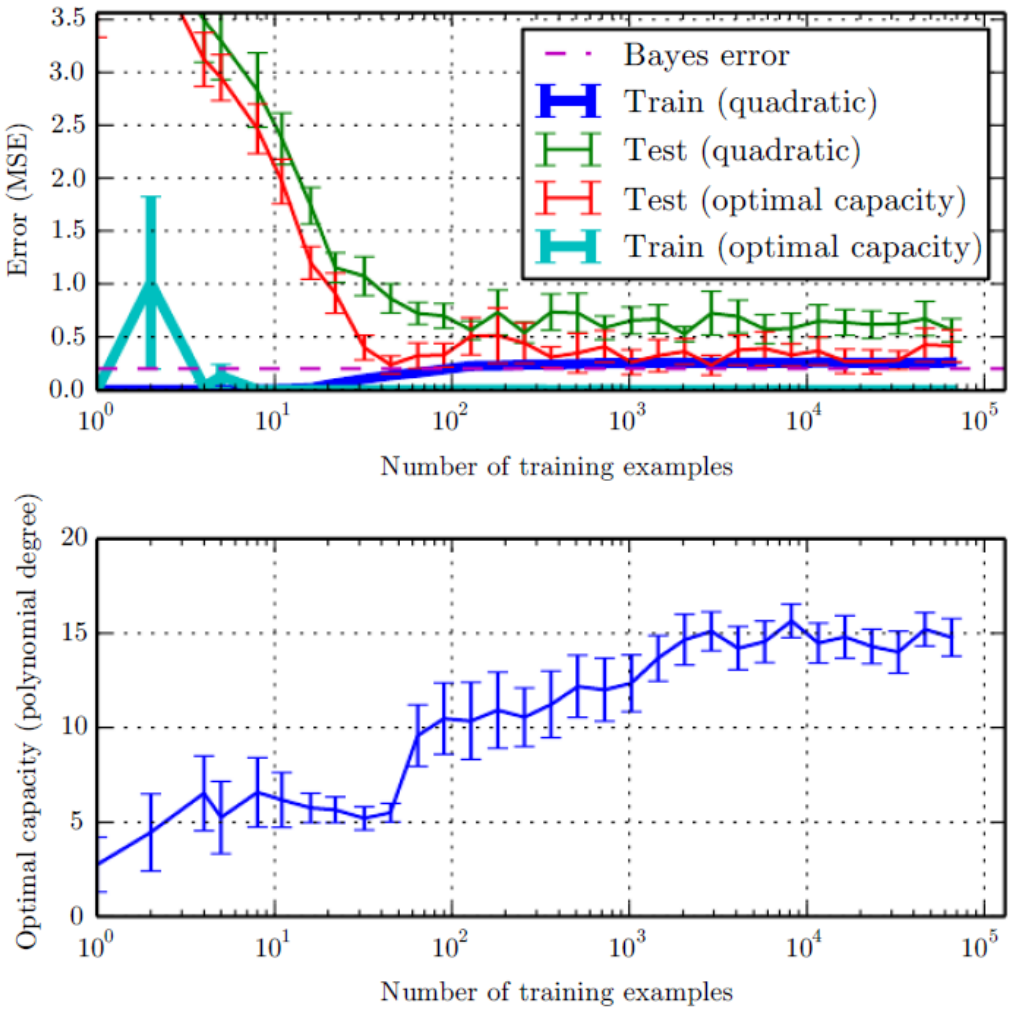

Figura 5.4 (nga kap. 5): Efekti i madhësisë së datasetit të trajnimit në gabimin e trajnimit dhe të testit, si dhe në kapacitetin optimal të modelit. E konstruktuam një problem të regresionit sintetik duke u bazuar në shtimin e një sasie të butë të zhurmës një polinomi të shkallës-5, e prodhuam një set të vetëm të testit dhe pastaj i prodhuam disa madhësi të ndryshme të setit të trajnimit. Për secilën madhësi, i gjeneruam 40 sete të ndryshme të trajnimit ashtu që t'i vizatojmë shiritat e gabimit që tregojnë intervale të konfidencës 95 përqind.
(Lart) MSE (Mean Squared Error - Gabimi 'mesatarja e katrorëve') në setin e trajnimit dhe të testimit për dy modele të ndryshme: një model kuadratik, dhe një model me shkallë të zgjedhur për ta minimizuar gabimin e testit. Të dyjat janë përputhur (fit) në formë të mbyllur. Për modelin kuadratik, gabimi i trajnimit rritet gjersa madhësia e setit të trajnimit rritet. Kjo sepse datasetat më të mëdhenj janë më të vështirë për t'i përputhur. Njëkohësisht, gabimi i testit zvogëlohet, sepse më pak hipoteza jokorrekte janë konsistente me shënimet trajnuese. Modeli kuadratik nuk ka mjaft kapacitet ta zgjidhë task-un, prandaj gabimi i tij i testit asimptoton në një vlerë të lartë. Gabimi i testit në kapacitetin optimal asimptoton në gabimin e Bayes-it. Gabimi i trajnimit mund të bie nën gabimin e Bayes-it, për shkak të aftësisë së algoritmit të trajnimit që të memorizojë instanca specifike të setit trajnues. Gjersa madhësia e trajnimit rritet në pafund, gabimi i trajnimit i çfarëdo modeli me kapacitet të fiksuar (këtu, modeli kuadratik) duhet të ngritet të paktën deri te gabimi i Bayes-it.
(Poshtë) Gjersa rritet madhësia e setit trajnues, kapaciteti optimal (i treguar këtu si shkalla e regresorit polinomial optimal) rritet. Kapaciteti optimal bëhet plato (rrafshnaltë) pas arritjes së kompleksitetit të mjaftueshëm për ta zgjidhur task-un.

11.4 Zgjedhja e hiperparametrave

Shumica e algoritmeve në deep learning vijnë me shumë hiperparametra që kontrollojnë shumë aspekte të sjelljes së algoritmit. Disa nga këta hiperparametra ndikojnë në kohën dhe koston memorike të ekzekutimit të algoritmit. Disa nga ta ndikojnë në cilësinë e modelit që fitohet nga procesi i trajnimit dhe aftësia e tij për të nxjerrë rezultate korrekte kur lëshohet në hyrje të reja.

Janë dy qasje themelore për zgjedhjen e këtyre hiperparametrave: zgjedhja e tyre manualisht dhe zgjedhja e tyre automatikisht. Zgjedhja e hiperparametrave manualisht kërkon kuptim të asaj që e bëjnë hiperparametrat dhe si arrijnë përgjithësim të mirë modelet e machine learning. Algoritmet me zgjedhje automatike të hiperparametrave e zvogëlojnë shumë nevojën për t'i kuptuar këto ide, mirëpo ato janë shpesh shumë më të kushtueshme për llogaritje.

11.4.1 Rregullimi manual i hiperparametrave

Për t'i caktuar hiperparametrat manualisht, personi duhet ta kuptojë lidhjen ndërmjet hiperparametrave, gabimit të trajnimit, gabimit të përgjithësimit dhe resurseve llogaritëse (memorja dhe koha e ekzekutimit). Kjo domethënë etablim i një baze të fortë mbi idetë themeltare në lidhje me kapacitetin efektiv të një algoritmi të mësimit nga kapitulli 5 (në libër).

Zakonisht qëllimi i kërkimit manual të hiperparametrit është të gjendet gabimi më i ulët i përgjithësimit nën kushtet e ndonjë buxheti të kohës së ekzekutimit dhe të memorjes. Këtu nuk e diskutojmë se si të përcaktohet ndikimi i hiperparametrave të ndryshëm në kohë të ekzekutimit dhe në memorje sepse kjo varet shumë nga platforma.

Qëllimi primar i kërkimit manual të hiperparametrave është të rregullohet kapaciteti efektiv i modelit për t'u përputhur me kompleksitetin e task-ut. Kapaciteti efektiv kufizohet nga tre faktorë: kapaciteti përfaqësues i modelit, aftësia e algoritmit të mësimit për ta minimizuar me sukses funksionin e kostos që përdoret për ta trajnuar modelin, dhe shkalla në të cilën funksioni i kostos dhe procedura e trajnimit e regularizojnë (rregullojnë) modelin. Një model me më shumë shtresa dhe më shumë njësi të fshehura për shtresë ka kapacitet më të lartë përfaqësues—ai ka kapacitet të përfaqësojë funksione më të komplikuara. Por ai nuk mund t'i mësojë patjetër të gjitha ato funksione, nëse algoritmi i trajnimit nuk mund të zbulojë se funksione të caktuara bëjnë punë të mirë të minimizimit të kostos së trajnimit, apo nëse termat e regularizimit siç është dobësimi i peshave i pengojnë disa nga ato funksione.

Gabimi i përgjithësimit zakonisht e ndjek një lakore në formë U-je kur vizatohet si funksion i njërit nga hiperparametrat, si në figurën 5.3.. Në njërin ekstrem, vlera e hiperparametrit korrespondon me kapacitetin e ulët, dhe gabimi i përgjithësimit është i lartë sepse gabimi i trajnimit është i lartë. Ky është regjimi i nënpërputhjes. Në ekstremin tjetër, vlera e hiperparametrit i korrespondon kapacitetit të lartë, dhe gabimi i përgjithësimit është i lartë sepse gap-i ndërmjet gabimit të trajnimit dhe të testit është i lartë. Diku në mes qëndron kapaciteti optimal i modelit, që e arrin gabimin më të ulët të

mundshëm të përgjithësimit, duke ia shtuar një gap të mesëm të përgjithësimit një sasie të mesme të gabimit të trajnimit.

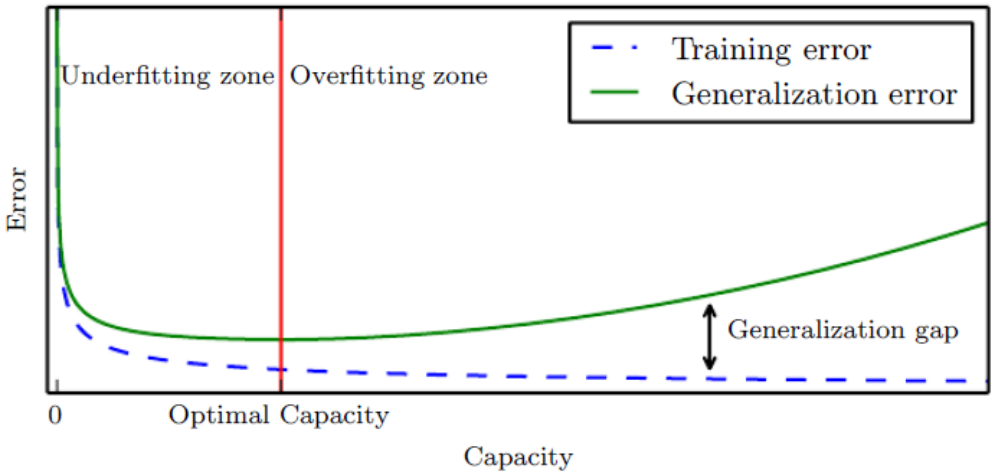

Figura 5.3 (nga kapitulli 5): Relacioni tipik ndërmjet kapacitetit dhe gabimit. Gabimi i trajnimit dhe i testit sillen ndryshe. Në skajin e majtë të grafit, edhe gabimi i trajnimit edhe gabimi i përgjithësimit janë të lartë. Ky është **regjimi i nënpërputhjes** (underfitting regime). Gjersa e rrisim kapacitetin, gabimi i trajnimit zvogëlohet, mirëpo hendeku ndërmjet gabimit të trajnimit dhe të përgjithësimit rritet. Përfundimisht, madhësia e këtij hendeku e mbipeshon zvogëlimin në gabimin e trajnimit, dhe hyjmë në **regjimin e mbipërputhjes** (overfitting regime), ku kapaciteti është tepër i madh, mbi **kapacitetin optimal** (optimal capacity).

Për disa hiperparametra, mbipërputhja ndodh kur vlera e hiperparametrit është e madhe. Numri i njësive të fshehura në një shtresë është një shembull i tillë, sepse rritja e numrit të njësive të fshehura e rrit kapacitetin e modelit. Për disa hiperparametra, mbipërputhja ndodh kur vlera e hiperparametrit është e vogël. Për shembull, koeficienti më i vogël i lejueshëm i dobësimit të peshës me vlerë zero i korrespondon kapacitetit më të madh efektiv të algoritmit të mësimit.

Jo çdo hiperparametër do të jetë i aftë ta eksplorojë lakoren e plotë në formë të U-së. Shumë hiperparametra janë diskretë, siç është numri i njësive në një shtresë apo numri i pjesëve lineare në një njësi maxout, prandaj është e mundur të vizitohen vetëm disa pika nëpër lakore. Disa hiperparametra janë binarë. Zakonisht këta hiperparametra janë ndërprerës (switch-a) që e specifikojnë se a të përdoret apo jo ndonjë komponentë opsionale e algoritmit të mësimit, siç është hapi i paraprocesimit që i normalizon veçoritë hyrëse duke e zbritur mesataren e tyre dhe duke pjesëtuar me devijimin e tyre standard. Këta hiperparametra mund t'i eksplorojnë vetëm dy pika në lakore. Hiperparametrat tjerë kanë ndonjë vlerë minimale apo maksimale që i pengon ata nga eksplorimi i ndonjë pjese të lakores. Për shembull, koeficienti i dobësimit minimal të peshave është zero. Kjo domethënë se nëse modeli po nënpërputhet kur dobësimi i peshave është zero, ne nuk mund të hyjmë në regjionin nënpërputhës duke e ndryshuar koeficientin e dobësimit të peshave. Me fjalë të tjera, disa hiperparametra mund ta zbresin vetëm kapacitetin.

Norma e mësimit (learning rate) është mbase hiperparametri më i rëndësishëm. Nëse keni kohë ta rregulloni vetëm një hiperparametër, rregullojeni normën e mësimit. Ajo e kontrollon kapacitetin efektiv të modelit në një mënyrë më të komplikuar se sa hiperparametrat e tjerë—kapaciteti efektiv i modelit është më i larti kur norma e mësimit

është *korrekte* për problemin e optimizimit, jo kur norma e mësimit është veçanërisht e madhe apo veçanërisht e vogël. Norma e mësimit ka lakore në formë të U-së për gabimin e *trajnimit*, e ilustruar në figurën 11.1. Kur norma e mësimit është tepër e madhe, ulja e gradientit mund ta rrisë paqëllim në vend sa ta ulë gabimin e trajnimit. Në rastin e idealizuar kuadratik, kjo ndodh nëse norma e mësimit është të paktën dy herë më e madhe se sa vlera e saj optimale (LeCun *et al.*, 1998a). Kur norma e mësimit është tepër e vogël, trajnimi jo vetëm që është më i ngadalshëm, por mund të bëhet i ngecur përgjithmonë me një gabim të lartë të trajnimit. Ky efekt është kuptuar dobët (nuk do të ndodhte për një funksion konveks të humbjes).

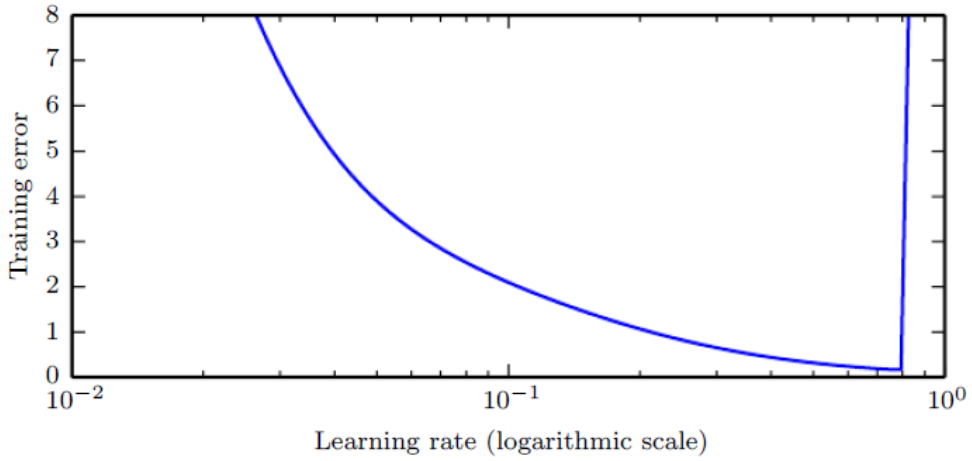

Figura 11.1: Relacioni tipik ndërmjet normës së mësimit dhe gabimit të trajnimit. Vërejeni rritjen e ashpër në gabim kur mësimi është përtej mbi një vlerë optimale. Kjo është për një kohë të fiksuar të trajnimit, meqë një normë më e vogël e mësimit nganjëherë mundet vetëm ta zvogëlojë trajnimin për një faktor proporcional me zvogëlimin e normës së mësimit. Gabimi i përgjithësimit mund ta ndjekë këtë lakore apo të komplikohet nga efektet e regularizimit që dalin nga të pasurit e normave tepër të mëdha apo tepër të vogla, meqë optimizimi i dobët mundet, deri në një shkallë, ta zvogëlojë apo ta parandalojë mbipërputhjen, dhe madje pikat me gabim ekuivalent të trajnimit mund të kenë gabim të ndryshëm të përgjithësimit.

Rregullimi i parametrave të tjerë (jo norma e mësimit) kërkon monitorim edhe të gabimit të trajnimit edhe të testit për ta diagnozuar se a po mbipërputhet apo po nënpërputhet modeli juaj, e pastaj rregullim të kapacitetit të tij përkatësisht.

Nëse gabimi juaj në setin e trajnimit është më i lartë se sa norma juaj e synuar e gabimit (target error rate), ju nuk keni zgjedhje pos ta rrisni kapacitetin. Nëse nuk po e përdorni regularizimin dhe jeni konfidentë se algoritmi juaj i optimizimit po performon në mënyrë korrekte, atëherë duhet t'i shtoni më shumë shtresa rrjetit tuaj apo të shtoni më shumë njësi të fshehura. Fatkeqësisht, kjo i rrit kostot llogaritëse të shoqëruara me modelin.

Nëse gabimi juaj në setin e testit është më i lartë se sa norma juaj e synuar e gabimit, tani ju mund t'i ndërmerrni dy lloje të veprimeve. Gabimi i testit është shuma e gabimit të trajnimit dhe gap-it ndërmjet gabimit të trajnimit dhe të testit. Gabimi optimal i testit gjendet duke i balansuar-tregtuar këto sasi. Rrjetet neurale zakonisht performojnë më së miri kur gabimi i trajnimit është shumë i ulët (dhe pra, kur kapaciteti është i lartë) dhe gabimi i testit është në radhë të parë i drejtuar prej gap-it ndërmjet gabimit të trajnimit

dhe të testit. Qëllimi juaj është ta zvogëloni këtë gap pa e rritur gabimin e trajnimit më shpejt se sa që zvogëlohet gap-i. Për ta zvogëluar gap-in, ndryshoni hiperparametrat e regularizimit për ta zvogëluar kapacitetin e modelit efektiv, si p.sh. duke shtuar dropout apo dobësim të peshave. Zakonisht performansa më e mirë vjen nga një model i madh që është i regularizuar mirë, për shembull duke përdorur dropout.

Shumica e hiperparametrave mund të caktohen duke arsyetuar për atë se a e rrisin apo e zvogëlojnë ata kapacitetin e modelit. Disa shembuj janë përfshirë në Tabelën 11.1.

Kur i rregulloni hiperparametrat manualisht, mos e humbni pamjen nga qëllimi i juaj përfundimtar: performansa e mirë në setin e testit. Shtimi i regularizimit është vetëm një mënyrë për ta arritur këtë qëllim. Përderisa keni gabim të ulët të trajnimit, ju mund ta zvogëloni gjithmonë gabimin e përgjithësimit duke mbledhur më shumë shënime trajnuese. Mënyra brute force (e vrazhdtë) për ta garantuar suksesin praktikisht është ta rrisni vazhdimisht kapacitetin e modelit dhe madhësinë e setit të trajnimit deri kur të zgjidhet task-u. Kjo qasje natyrisht e rrit koston llogaritëse të trajnimit dhe konkludimit, prandaj është e realizueshme vetëm kur janë dhënë resurse te duhura. Në princip, kjo qasje do të mund të dështonte për shkak të vështirësive të optimizimit, por për shumë probleme optimizimi nuk duket të jetë barrierë domethënëse, nën kushtin që modeli është zgjedhur si duhet.

11.4.2 Algoritmet e optimizimit me hiperparametra automatikë

Algoritmi ideal i mësimit vetëm e merr një dataset dhe e nxjerr në dalje një funksion, pa kërkuar rregullim me dorë të hiperparametrave. Popullariteti i disa algoritmeve të mësimit siç është regresioni logjistik dhe SVM-të buron pjesërisht nga aftësia e tyre për të performuar mirë me vetëm një apo dy hiperparametra të rregulluar. Rrjetet neurale nganjëherë mund të performojnë mirë me vetëm një numër të vogël të hiperparametrave të rregulluar. Rregullimi manual i hiperparametrave mund të funksionojë shumë mirë kur përdoruesi e ka një pikë të mirë startuese, siç është ajo e përcaktuar nga të tjerët që kanë punuar në llojin e njëjtë të aplikimit dhe arkitekturës, apo kur përdoruesi ka përvojë me muaj apo vite në eksplorimin e vlerave hiperparametrike për rrjete neurale të aplikuara në task-a të ngjashëm. Mirëpo, për shumë aplikime, këto pika startuese nuk janë në dispozicion. Në këto raste, algoritmet e automatizuara mund të gjejnë vlera të dobishme të hiperparametrave.

Nëse mendojmë për mënyrën në të cilën përdoruesi i një algoritmi që mëson kërkon vlera të mira të hiperparametrave, e kuptojmë se po ndodh një optimizim: po provojmë ta gjejmë një vlerë të hiperparametrave që e optimizon një funksion objektiv, siç është gabimi i validimit, nganjëherë nën kufizime (siç është buxheti për kohën e trajnimit, memorjen apo kohën e njohjes). Prandaj është e mundur, në princip, të zhvillohen algoritme të **optimizimit të hiperparametrave** që e mbështjellin një algoritëm të mësimit dhe i zgjedhin hiperparametrat e tij, duke i fshehur kështu nga përdoruesi hiperparametrat e algoritmit të mësimit. Fatkeqësisht, algoritmet e optimizimit të hiperparametrave shpesh i kanë hiperparametrat e vet, siç është rangu i vlerave që duhet të eksplorohen për secilin nga hiperparametrat e algoritmit të mësimit. Mirëpo, këta hiperparametra sekondarë janë më të lehtë për t'u zgjedhur, në sensin që performansa e pranueshme mund të arrihet në një rang të gjerë të task-ave duke përdorur hiperparametra të njëjtë sekondarë për të gjithë task-at.

Hiperparametri	**E rrit kapacitetin kur...**	**Arsyeja**	**Paralajmërime**
Numri i njësive të fshehura	rritet	Rritja e numrit të njësive të fshehura e rrit kapacitetin përfaqësues të modelit.	Rritja e numrit të njësive të fshehura e rrit edhe kohën edhe koston e memorjes praktikisht të secilit operacion në model.
Norma e mësimit (learning rate)	rregullohet optimalisht	Një normë e pasaktë e mësimit, qoftë tepër e lartë apo tepër e ulët, rezulton në një model me kapacitet të ulët efektiv për shkak të dëmtimit të optimizimit	
Convolution kernel width (gjerësia e kernel-it të konvolucionit)	rritet	Rritja e gjerësisë së kernel-it e rrit numrin e parametrave në model	Një kernel (thelb) më i gjerë rezulton në një dimension dalës më të ngushtë, duke e zvogëluar kapacitetin e modelit përveç nëse e përdorni implicit zero padding (shtimin artificial implicit të zerove) për ta zvogëluar këtë efekt. Kernel-ët më të gjerë kërkojnë më shumë memorje për ruajtjen e parametrave dhe e rrisin kohën e ekzekutimit, por një dalje më e ngushtë e zvogëlon koston e memorjes.
Implicit zero padding (shtimi artificial implicit i zerove)	rritet	Shtimi i zerove implicite para konvolucionit e mban të madhe madhësinë e përfaqësimit	Koha dhe kostoja memorike të rritura për shumicën e operacioneve
Weight decay coefficient (koeficienti i dobësimit të peshave)	zvogëlohet	Zvogëlimi i dobësimit të peshave i liron parametrat e modelit të bëhen më të mëdhenj	
Dropout rate (norma e braktisjes)	zvogëlohet	Largimi i njësive më rrallë, shpesh iu jep njësive më shumë shanse të "konspirojnë" me njëra tjetrën për ta përputhur setin e trajnimit	

Tabela 11.1: Efekti i hiperparametrave të ndryshëm në kapacitetin e modelit.

11.4.3 Grid search

Kur janë tre apo më pak hiperparametra, praktika e zakonshme është të bëjmë **grid search** (kërkim në grid, tabelë). Për secilin hiperparametër, përdoruesi e zgjedh një grup të vogël të fundëm të vlerave për t'i eksploruar. Algoritmi grid search pastaj e trajnon një model për secilin specifikim të përbashkët të vlerave hiperparametrike në Produkin Kartezian të grupit të vlerave për secilin hiperparametër individual. Eksperimenti që e jep gabimin më të mirë të grupit të validimit pastaj zgjedhet si ai që i ka gjetur hiperparametrat më të mirë. Shiheni anën e majtë të figurës 11.2 për një ilustrim të një grid-i të vlerave hiperparametrike.

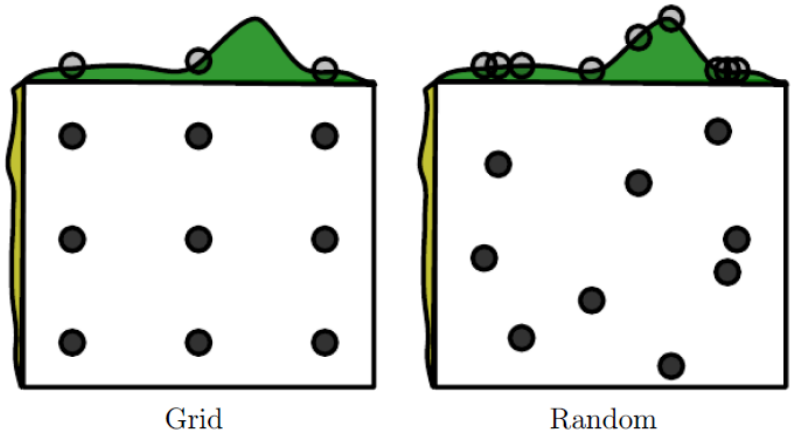

Figura 11.2: Krahasimi i grid search (kërkimit në grid) dhe random search (kërkimit të rastësishëm). Për qëllime të ilustrimit i shfaqim dy hiperparametra mirëpo zakonisht jemi të interesuar të kemi shumë më shumë.
(Majtas) Për të bërë kërkim në grid, e japim një bashkësi të vlerave për secilin hiperparametër. Algoritmi i kërkimit e ekzekuton trajnimin për secilin aranzhim të hiperparametrave të bashkuar në produktin kartezian të këtyre bashkësive.
(Djathtas) Për të bërë kërkim të rastit, e ofrojmë një shpërndarje të gjasës në konfigurimet e hiperparametrave të bashkuar. Zakonisht shumica nga këta hiperparametra janë të pavarur nga njëri-tjetri. Zgjedhjet më të shpeshta për shpërndarje të një hiperparametri të vetëm përfshijnë uniformen dhe log-uniformen (për të mostruar nga një shpërndarje log-uniforme, merreni exp të një mostre nga një shpërndarje uniforme). Algoritmi i kërkimit pastaj i mostron rastësisht konfigurimet e hiperparametrave të bashkuar dhe e ekzekuton trajnimin me secilin nga ta. Edhe kërkimi në grid edhe kërkimi i rastësishëm e vlerësojnë gabimin e bashkësisë së validimit dhe e kthejnë konfigurimin më të mirë. Figura e ilustron rastin tipik kur vetëm disa hiperparametra kanë ndikim domethënës në rezultat. Kërkimi në grid e harxhon një sasi të llogaritjes që është eksponenciale me numrin e hiperparametrave jo-ndikues, ndërsa kërkimi i rastësishëm e teston një vlerë unike të secilit hiperparametër ndikues në pothuajse secilin tentim. Figura e reprodukuar me leje nga Bergstra and Bengio (2012).

Si duhet të zgjedhen listat e vlerave në të cilat do të kërkohet? Në rastin e hiperparametrave numerikë (të renditur), zgjedhet me kujdes elementi më i vogël dhe më i madh i secilës listë, bazuar në përvojën paraprake me eksperimente të ngjashme, për t'u siguruar se vlera optimale ka shumë gjasa të jetë në rangun e zgjedhur. Zakonisht, një kërkim në grid përfshin zgjedhje të vlerave afërsisht në një *shkallë logaritmike*, p.sh. një

normë e mësimit e marrur brenda bashkësisë {.1, .01, 10^{-3}, 10^{-4}, 10^{-5}}, apo një numër i njësive të fshehura i marrur me bashkësinë {50, 100, 200, 500, 1000, 2000}.

Kërkimi në grid zakonisht performon më së miri kur performohet në mënyrë të përsëritur. Për shembull, supozojmë se e kemi ekzekutuar një kërkim në grid ndaj një hiperparametri α duke i përdorur vlerat {-1, 0, 1}. Nëse vlera më e mirë e gjetur është 1, atëherë ne e kemi nënvlerësuar rangun në të cilin qëndron α më e mirë dhe do të duhej ta zhvendosnim grid-in dhe ta ekzekutojmë edhe një kërkim me α në, për shembull, {1, 2, 3}. Nëse gjejmë se vlera më e mirë e α është 0, atëherë mund të duam ta përmirësojmë vlerësimin tonë duke zoom-uar dhe duke e ekzekutar një kërkim në grid ndaj { −.1, 0, .1}.

Problemi evident me kërkimin në grid është se kostoja e tij llogaritëse rritet eksponencialisht me numrin e hiperparametrave. Nëse janë m hiperparametra, secili që i merr të shumtën n vlera, atëherë numri i kërkuar i tentimeve të trajnimit dhe evaluimit rritet si $O(n^m)$. Tentimet mund të ekzekutohen paralelisht dhe të shfrytëzohet paralelizmi i lirshëm (ku nuk ka pothuaj fare nevojë për komunikim ndërmjet makinave të ndryshme që e bëjnë kërkimin). Fatkeqësisht, për shkak të kostos eksponenciale të kërkimit në grid, madje edhe paralelizimi mund të mos ofrojë madhësi të kënaqshme të kërkimit.

11.4.4 Random search

Fatmirësisht, është një alternativë ndaj kërkimit në grid që është po aq e thjeshtë për ta programuar, më e përshtatshme për ta përdorur, dhe konvergjon shumë më shpejt në vlera të mira të hiperparametrave: random search (kërkimi i rastësishëm) (Bergstra and Bengio, 2012).

Një random search zhvillohet si vijon. Së pari e definojmë një shpërndarje margjinale për secilin hiperparametër, p.sh., një Bernuli apo multinuli për hiperparametrat binarë apo diskretë, apo një shpërndarje uniforme në një shkallë-logaritmike për hiperparametrat pozitivë me vlera reale. Për shembull,

$$\texttt{log\_learning\_rate} \sim u(-1, -5) \tag{11.2}$$

$$\texttt{learning\_rate} = 10^{\texttt{log\_learning\_rate}}. \tag{11.3}$$

Ku $u(a, b)$ e shënon një mostër të shpërndarjes uniforme në intervalin (a, b). Ngjashëm, `log_number_of_hidden_units` mund të mostrohet nga $u(\log(50), \log(2000))$.

Për dallim nga rasti i grid search, personi *nuk duhet t'i diskretojë* apo t'i ndajë në bin-a (kova) vlerat e hiperparametrave. Kjo lejon që personi ta eksplorojë një grup më të madh të vlerave, dhe nuk shkakton kosto llogaritëse shtesë. Në fakt, siç është ilustruar në figurën 11.2, një random search mund të jetë eksponencialisht më efikas se një grid search, kur ka disa hiperparametra që nuk ndikojnë fuqishëm në matjen e performansës. Kjo është studiuar gjerësisht në Bergstra and Bengio (2012), të cilët gjetën se random search e zvogëlon gabimin e setit të validimit shumë më shpejt se grid search-i, në terma të numrit të tentimeve të ekzekutuara nga secila metodë.

Si edhe me grid search, personi shpesh mund të dëshirojë të ekzekutojë versione të përsëritura të random search-it, për ta përmirësuar kërkimin bazuar në rezultatet e ekzekutimit të parë.

Arsyeja kryesore pse random search gjen zgjidhje të mira më shpejt se grid search është sepse nuk ka ekzekutime eksperimentale të çuara dëm, për dallim nga rasti i grid search-it, kur dy vlerat e një hiperparametri (atëherë kur janë dhënë vlerat e hiperparametrave të tjerë) do ta jepnin rezultatin e njëjtë. Në rastin e grid search-it, hiperparametrat e tjerë do t'i kishin vlerat e njëjta për këto dy ekzekutime, ndërsa me random search, ata zakonisht do të kishin vlera të ndryshme. Prandaj nëse ndryshimi ndërmjet dy vlerave nuk bën dallim të madh margjinal në terma të gabimit të setit të validimit, grid search do t'i përsërisë pa nevojë dy eksperimente ekuivalente ndërsa random search do t'i japë prapë dy eksplorime të pavarura të hiperparametrave të tjerë.

11.4.5 Optimizimi i hiperparametrave i bazuar në model

Kërkimi i hiperparametrave të mirë mund të paraqitet si problem i optimizimit. Variablat e vendimit janë hiperparametrat. Kostoja që do të optimizohet është gabimi i setit të validimit që rezulton nga trajnimi duke i përdorur këta hiperparametra. Në aranzhime të thjeshtuara ku është e realizueshme të llogaritet gradienti i ndonjë mase të diferencueshme të gabimit në setin e validimit në lidhje me hiperparametrat, mund ta ndjekim këtë gradient (Bengio *et al.*, 1999; Bengio, 2000; Maclaurin *et al.*, 2015). Fatkeqësisht, në shumicën e aranzhimeve praktike, ky gradient është i padisponueshëm, ose për shkak të kostos së tij të lartë të llogaritjes dhe të memorjes, ose për shkak të hiperparametrave që kanë ndërveprime brendësisht të padiferencueshme me gabimin e setit të validimit, si në rastin e hiperparametrave me vlera diskrete.

Për ta kompensuar këtë mungesë të gradientit, mund ta ndërtojmë një model të gabimit të setit të validimit, pastaj të propozojmë hamendje të reja të hiperparametrave duke e bërë optimizimin brenda këtij modeli. Shumica e algoritmeve të bazuara në modele për kërkim të hiperparametrave e përdorin një model të regresionit Bayesian për ta vlerësuar edhe vlerën e pritur të gabimit të setit të validimit për secilin hiperparametër edhe pasigurinë përreth kësaj pritjeje. Kështu optimizimi e përfshin një balansim (tradeoff) ndërmjet eksplorimit (propozimit të hiperparametrave për të cilët ka pasiguri të lartë, që mund të çojë në një përmirësim të madh por mund edhe të performojë dobët) dhe eksploatimit (propozimit të hiperparametrave për të cilët modeli është konfident se do të performojnë po aq mirë sa cilëtdo hiperparametra që i ka parë deri tani—zakonisht hiperparametra që janë shumë të ngjashëm me ata që i ka parë më herët). Qasjet bashkëkohore ndaj optimizimit të hiperparametrave përfshijnë Spearmint (Snoek *et al.*, 2012), TPE (Bergstra *et al.*, 2011) dhe SMAC (Hutter *et al.*, 2011).

Aktualisht, nuk mund ta rekomandojmë qartë optimizimin Bayesian të hiperparametrave si vegël të etabluar për të arritur rezultate më të mira të deep learning apo për t'i siguruar ato rezultate me më pak përpjekje. Optimizimi Bayesian i hiperparametrave nganjëherë performon në mënyrë të krahasueshme me ekspertët njerëzorë, nganjëherë më mirë, mirëpo dështon në mënyrë katastrofike në probleme të tjera. Mund t'ia vlejë të provojmë ta shohim se a funksionon në një problem të caktuar por ai nuk është ende mjaft i pjekur apo i besueshëm. Thënë këtë, optimizimi i hiperparametrave është fushë e rëndësishme e kërkimit që, ndërsa shpesh drejtohet në radhë të parë nga nevojat e deep learning, mban potencial për t'i sjellë dobi jo vetëm tërë fushës së machine learning por edhe disciplinës së inxhnieringut në përgjithësi.

Një mangësi e zakonshme për shumicën e algoritmeve të optimizimit të hiperparametrave me më shumë sofistikim se sa random search është se ata kërkojnë që një eksperiment trajnues të ekzekutohet deri në përfundim para se të jenë të aftë të nxjerrin ndonjë informacion nga eksperimenti. Kjo është shumë më pak efikase—në sensin se sa shumë informacion mund të grumbullohet pak-nga-pak herët në eksperiment—se sa kërkimi manual nga një praktikues njeri, meqë personi zakonisht mund ta dijë herët nëse ndonjë set i hiperparametrave është plotësisht patologjik. Swersky *et al.* (2014) e kanë paraqitur një version të hershëm të një algoritmi që e mirëmban një set të shumë eksperimenteve. Në pika të ndryshme kohore, algoritmi i optimizimit të hiperparametrave mund të zgjedhë ta nisë një eksperiment të ri, ta "ngrijë" një eksperiment që po ekzekutohet që nuk është premtues, apo ta "shkrijë" dhe ta vazhdojë një eksperiment që ishte ngrirë më herët por që tani duket premtues kur është dhënë më shumë informacion.

11.5 Strategji të debug-imit

Kur një sistem i machine learning performon dobët, zakonisht është e vështirë të thuhet se a është performansa e dobët në brendësi të vetë algoritmit apo a ka bug (gabim, defekt) në implementimin e algoritmit. Sistemet machine learning janë të vështira për t'u debug-uar për një mori të arsyeve.

Në shumicën e rasteve, nuk e dimë a priori se cila është sjellja e synuar e algoritmit. Në fakt, e tërë poenta e përdorimit të machine learning është se ai do të mësojë sjellje të dobishme të cilat nuk ishim të aftë t'i specifikonim vetë. Nëse e trajnojmë një rrjet neural në një task *të ri* të klasifikimit dhe ai arrin gabim të testit 5%, ne nuk kemi mënyrë të thjeshtë ta dimë se a është kjo sjellja e pritur apo sjellje nën-optimale.

Një vështirësi e mëtejme është se shumica e modeleve të machine learning kanë shumë pjesë ku secila është adaptive. Nëse një pjesë prishet, pjesët e tjera mund të adaptohen dhe prapë të arrijnë performansë të pranueshme në vija të trasha. Për shembull, supozojmë se po e trajnojmë një rrjet neural me disa shtresa të parametrizuara nga peshat $\boldsymbol{W}$ dhe bias-et $\boldsymbol{b}$. Supozojmë tutje se e kemi implementuar manualisht rregullin e uljes së gradientit për secilin parametër veçmas, dhe e kemi bërë një gabim në përditësimin për bias-et:

$$\boldsymbol{b} \leftarrow \boldsymbol{b} - \alpha \tag{11.4}$$

ku α është norma e mësimit (learning rate). Ky përditësim i pasaktë nuk e përdor gradientin fare. Ai shkakton që bias-et të bëhen vazhdimisht negative përgjatë mësimit, që qartë nuk është implementim korrekt i asnjë algoritmi të arsyeshëm të mësimit. Mirëpo bug-u mund të mos jetë i dukshëm vetëm nga kontrollimi i rezultatit dalës të modelit. Varësisht nga shpërndarja e hyrjes, peshat mund të jenë të afta të adaptohen për t'i kompensuar bias-et negative.

Shumica e strategjive të debug-imit për rrjetet neurale janë të dizajnuara për t'iu shmangur njërës apo të dy këtyre vështirësive. Ose e dizajnojmë një rast që është kaq i

thjeshtë sa që sjellja korrekte në fakt mund të parashikohet, ose e dizajnojmë një test që e ushtron një pjesë të implementimit të rrjetit neural në izolim.

Disa teste të rëndësishme të debug-imit përfshijnë:

Vizualizojeni modelin në veprim: Kur e trajnoni një model për t'i detektuar objektet në imazhe, shihni disa imazhe me detektimet e propozuara nga modeli të shfaqura në mënyrë të mbivendosur në imazh. Kur e trajnoni një model gjenerativ të të folurit, dëgjoni disa nga mostrat e të folurit që i prodhon. Kjo mund të duket evidente, mirëpo është lehtë të biem në praktikën vetëm të shikimit të matjeve sasiore të performansës siç është saktësia apo log-likelihood. Vëzhgimi direkt i modelit të machine learning duke e kryer task-un e vet do të ndihmojë të përcaktojmë se a duken të arsyeshëm numrat sasiorë të performansës që i arrin. Bug-at e evaluimit (vlerësimit) mund të jenë një ndër bug-at më rrënues sepse mund të ju çorientojnë në besimin se sistemi i juaj është duke performuar mirë kur nuk është.

Vizualizoni gabimet më të këqija: Shumica e modeleve janë të afta të nxjerrin si rezultat ndonjë lloj matje të konfidencës për task-un që e kryejnë. Për shembull, klasifikuesit e bazuar në një shtresë dalëse softmax ia caktojnë një gjasë secilës klasë. Kështu, gjasa që ishtë caktuar klasës më të pritur e jep një vlerësim të konfidencës që e ka modeli në vendimin e vet të klasifikimit. Zakonisht, trajnimi me gjasë maksimale rezulton që këto vlera të jenë mbivlerësime në vend se gjasa të sakta të parashikimit korrekt, mirëpo ato janë në njëfarë mënyre të dobishme në sensin që shembujt që në fakt kanë më pak gjasë të etiketohen në mënyrë korrekte pranojnë gjasa më të vogla nën këtë model. Duke i shikuar shembujt e setit të trajnimit që janë më të vështirët për t'i modeluar korrektësisht, personi shpesh mund të zbulojë probleme në mënyrën se si janë paraprocesuar apo etiketuar shënimet. Për shembull, sistemi i transkriptimit Street View fillimisht e kishte një problem ku sistemi i detektimit të numrit të adresës do ta prente imazhin tepër ngusht dhe do t'i hiqte disa nga shifrat. Rrjeti i transkriptimit pastaj caktonte gjasë shumë të ulët të përgjigjes së saktë në këto imazhe. Radhitja e imazheve për t'i identifikuar gabimet më konfidente tregoi se ishte një problem sistematik me prerjen. Modifikimi i sistemit të detektimit për të prerë imazhe shumë më të gjera rezultoi në performansë shumë më të mirë të sistemit të tërësishëm, edhe pse rrjetit të transkriptimit i nevojitej të jetë i aftë të procesojë variacion më të madh në pozitë dhe shkallë të numrave të adresave.

Rezonimi për softuerin duke e përdorur gabimin e trajnimit dhe të testit: Shpesh është vështirë të përcaktohet se a është implementuar në mënyrë korrekte softueri i aplikimit. Disa të dhëna mund të merren nga gabimi i trajnimit dhe i testit. Nëse gabimi i trajnimit është i ulët por gabimi i testit është i lartë, atëherë me gjasë që procedura e trajnimit punon në mënyrë korrekte, dhe modeli po mbipërputhet për arsye algoritmike themeltare. Një mundësi alternative është se gabimi i testit matet në mënyrë jokorrekte për shkak të një problemi me ruajtjen e modelit pas trajnimit dhe pastaj ringarkimit të tij për evaluimin e setit të testit, apo nëse shënimet testuese janë përgatitur ndryshe nga shënimet trajnuese. Nëse edhe gabimi i trajnimit edhe i testit janë të lartë, atëherë është vështirë të përcaktohet se a ka defekt softuerik apo po mbipërputhet modeli për shkak të arsyeve algoritmike themeltare. Ky skenar kërkon teste të mëtutjeshme, të përshkruara në vijim.

Përputheni një dataset të vockël: Nëse keni gabim të madh në setin e trajnimit, përcaktojeni se a është kjo për shkak të nënpërputhjes së vërtetë apo për shkak të një

defekti softuerik. Zakonisht madje edhe modelet e vogla mund të garantohen të jenë të afta ta përputhin një dataset mjaft të vogël. Për shembull, një dataset i klasifikimit me vetëm një shembull mund të përputhet vetëm duke i caktuar në mënyrë korrekte bias-et e shtresës dalëse. Zakonisht nëse nuk mund ta trajnoni një klasifikues për ta etiketuar në mënyrë korrekte një shembull të vetëm, një autoencoder për ta riprodhuar me sukses një shembull të vetëm me besnikëri të lartë, apo një model gjenerativ për të emetuar në mënyrë konsistente mostra që i ngjajnë një shembulli të vetëm, atëherë është një defekt softuerik që e pengon optimizimin e suksesshëm në setin e trajnimit. Ky test mund të zgjerohet në një dataset të vogël me disa shembuj.

Krahasoni derivatet back-propagated (të prapa-propaguara) me derivatet numerike: Nëse jeni duke përdorur software framework që kërkon t'i implementoni llogaritjet tuaja të gradientit, apo nëse jeni duke e shtuar një operacion të ri në librarinë e diferencimit dhe duhet ta definoni metodën e saj `bprop`, atëherë një burim i shpeshtë i gabimit është që kjo shprehje e gradientit të implementohet në mënyrë jokorrekte. Një mënyrë për ta verifikuar se a janë derivatet korrekte është të krahasohen derivatet e llogaritura nga implementimi juaj i diferencimit automatik me derivatet e llogaritura nga **diferencat jozero** (finite differences). Meqë

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}, \tag{11.5}$$

mund ta përafrojmë derivatin duke e përdorur një ϵ të vogël, jozero:

$$f'(x) \approx \frac{f(x+\epsilon) - f(x)}{\epsilon}. \tag{11.6}$$

Mund ta përmirësojmë saktësinë e përafrimit duke e përdorur **diferencën e qendërzuar** (centered difference):

$$f'(x) \approx \frac{f(x+\frac{1}{2}\epsilon) - f(x-\frac{1}{2}\epsilon)}{\epsilon}. \tag{11.7}$$

Madhësia e shmangies ϵ duhet të zgjedhet të jetë mjaft e madhe për t'u siguruar që shmangia nuk rrumbullakësohet tepër nga llogaritjet numerike me precizitet jozero.

Zakonisht, do të duam ta testojmë gradientin apo Jacobian-in e një funksioni me vlera vektoriale $g : \mathbb{R}^m \to \mathbb{R}^n$. Fatkeqësisht, diferencimi jozero na lejon ta nxjerrim vetëm një derivat përnjëherë. Ose mund ta bëjmë diferencimin mn herë për t'i evaluuar (llogaritur) të gjitha derivatet e pjesshme të g, ose mund ta aplikojmë testin në një funksion të ri që përdor projeksione të rastësishme edhe në hyrje edhe në dalje të g-së. Për shembull, mund ta aplikojmë testin tonë të implementimit të derivateve në $f(x)$ ku $f(x) = \boldsymbol{u}^\top g(\boldsymbol{v}x)$, ku $\boldsymbol{u}$ dhe $\boldsymbol{v}$ janë vektorë të zgjedhur rastësisht. Llogaritja e $f'(x)$ në mënyrë korrekte kërkon të jemi të aftë të propagojmë prapa nëpër g në mënyrë korrekte, por është efikase të bëhet me diferenca jozero sepse f e ka vetëm një hyrje të vetme dhe një dalje të vetme. Zakonisht është ide e mirë që ky test të përsëritet për më shumë se një vlerë të $\boldsymbol{u}$-së dhe $\boldsymbol{v}$-së për ta zvogëluar gjasën që testi të mos i vërejë gabimet që janë ortogonale me projeksionin e rastësishëm.

Nëse personi ka qasje në llogaritjen numerike në numra kompleksë, atëherë është një mënyrë shumë efikase për ta estimuar numerikisht gradientin duke i përdorur numrat kompleksë si hyrje të funksionit (Squire and Trapp, 1998). Metoda bazohet në observimin se

$$f(x + i\epsilon) = f(x) + i\epsilon f'(x) + O(\epsilon^2) \tag{11.8}$$

$$\text{real}(f(x + i\epsilon)) = f(x) + O(\epsilon^2), \quad \text{imag}(\frac{f(x + i\epsilon)}{\epsilon}) = f'(x) + O(\epsilon^2), \tag{11.9}$$

Ku i = √-1. Për dallim nga rasti i mësipërm me vlera reale, këtu nuk ka efekt të anulimit për shkak të marrjes së diferencave ndërmjet vlerës së *f*-së në pika të ndryshme. Kjo lejon përdorim të vlerave të vockla të ϵ-it si $\epsilon = 10^{-150}$, që e bëjnë gabimin $O(\epsilon^2)$ të parëndësishëm për të gjitha qëllimet praktike.

Monitoroni histogramet e aktivizimeve dhe të gradientit: Shpesh është e dobishme të vizualizohen statistikat e aktivizimeve dhe të gradientëve të rrjeteve neurale, të mbledhura gjatë një sasie të madhe të iterimeve trajnuese (mbase një epokë). Vlera para-aktivizuese e njësive të fshehura mund të na tregojë se a saturojnë (ngopen) njësitë, apo sa shpesh saturojnë ato. Për shembull, për korrigjuesit (rectifiers), sa herë janë të ndalur? A ka njësi që janë gjithmonë të ndalura? Për njësitë tanh, mesatarja e vlerës absolute të para-aktivizimeve na tregon se sa është njësia e saturuar. Në një rrjet të thellë ku gradientët e propaguar rriten shpejt apo dobësohen shpejt, optimizimi mund të jetë i frenuar. Në fund, është e dobishme të krahasohet magnituda (madhësia absolute) e gradientëve të parametrave me magnitudën e vetë parametrave. Siç është sugjeruar nga Bottou (2015), do të donim që magnituda e përditësimeve (update-ave) të parametrave ndaj një minibatch-i (minigrupi) të paraqesë diçka si 1% të magnitudës së parametrit, jo 50% apo 0.001% (që do t'i bënte parametrat të lëvizin tepër ngadalë). Mund të ndodhë që disa grupe të parametrave janë duke lëvizur në tempo të mirë ndërsa të tjerët kanë ngecur. Kur shënimet janë të rralla (si në gjuhën natyrale), disa parametra mund të përditësohen shumë rrallë, dhe kjo duhet të mbahet mend kur monitorohet evoluimi i tyre.

Në fund, shumë algoritme të deep learning e ofrojnë njëfarë lloj garancie për rezultatet e prodhuara në secilin hap. Për shembull, në pjesën III (në libër), do t'i shohim disa algoritme të përafërta të konkludimit që punojnë duke përdorur zgjidhje algjebrike në probleme të optimizimit. Zakonisht këto mund të debug-ohen duke e testuar secilën nga garancitë e tyre. Disa garanci që ofrohen nga disa algoritme të optimizimit thonë që funksioni objektiv nuk do të rritet kurrë pas një hapi të algoritmit, se gradienti në lidhje me ndonjë nëngrup të variablave do të jetë zero pas secilit hap të algoritmit, dhe se gradienti në lidhje me të gjitha variablat do të jetë zero në konvergjencë. Zakonisht për shkak të gabimit të rrumbullakësimit, këto kushte nuk do të qëndrojnë saktësisht në një kompjuter digjital, prandaj testi i debug-imit duhet të përfshijë ndonjë parametër të tolerancës.

11.6 Shembull: Njohja e numrit shumëshifror

Për ta ofruar një përshkrim fillim-e-mbarim se si të aplikohet në praktikë metodologjia jonë e dizajnit, po e prezentojmë një shpjegim të shkurtër të sistemit të transkriptimit Street View, nga pikëpamja e dizajnit të komponentave të deep learning. Qartë, shumë komponenta të tjera të sistemit të plotë, siç janë veturat e Street View, infrastruktura e bazës së shënimeve, e kështu me radhë, ishin të rëndësisë së lartë.

Nga pikëpamja e task-ut të machine learning, procesi filloi me mbledhjen e shënimeve. Veturat mblidhnin shënime të papunuara dhe operatorët njerëzorë ofronin etiketa. Task-u i transkriptimit ishte paraprirë nga një sasi domethënëse e kurimit (shoshitjes) së datasetit, duke përfshirë edhe teknika të tjera të machine learning për t'i *detektuar* numrat e shtëpive para transkriptimit të tyre.

Projekti i transkriptimit nisi me një zgjedhje të metrikave të performansës dhe vlerave të dëshiruara për këto metrika. Një princip i rëndësishëm i përgjithshëm është t'i përshtatet zgjedhja e metrikës qëllimit biznesor për projektin. Meqë hartat janë të dobishme vetëm nëse kanë saktësi të lartë, ishte me rëndësi të caktohej një kërkesë e lartë e saktësisë për këtë projekt. Specifikisht, qëllimi ishte të arrihej nivel njerëzor, saktësi 98%. Ky nivel i saktësisë mund të mos jetë gjithmonë i realizueshëm. Për ta arritur këtë nivel të saktësisë, sistemi i transkriptimit Street View e sakrifikon mbulimin. Kështu mbulimi u bë metrika kryesore e performansës e optimizuar gjatë projektit, me saktësinë të mbajtur në 98%. Gjersa rrjeti konvolucional përmirësohej, u bë e mundur të zvogëlohet pragu i konfidencës nën të cilin rrjeti refuzon ta transkriptojë hyrjen, përfundimisht duke e kapërcyer qëllimin e mbulimit prej 95%.

Pas zgjedhjes së qëllimeve sasiore, hapi tjetër në metodologjinë tonë të rekomanduar është të etablohet shpejt një sistem i mençur baseline. Për task-a të vision-it (shikimit), kjo domethënë një rrjet konvolucionar me rectified linear units (njësi të korrigjuara lineare). Projekti i transkriptimit nisi me një model të tillë. Në atë kohë, nuk ishte e zakonshme për një rrjet konvolucional të nxirrte si rezultat një sekuencë të parashikimeve. Për të nisur me baseline-in më të thjeshtë të mundshëm, implementimi i parë i shtresës dalëse të modelit përbëhej nga n njësi të ndryshme softmax për ta parashikuar një sekuencë prej n karakterëve. Këto njësi softmax ishin trajnuar saktësisht në mënyrën e njëjtë sikur task-u të ishte klasifikim, ku secila njësi softmax trajnohej pavarësisht.

Metodologjia jonë e rekomanduar është të përmirësohet iterativisht baseline-i dhe të testohet se a po bën secili ndryshim ndonjë përmirësim. Ndryshimi i parë në sistemin e transkriptimit Street View u motivua nga një kuptim teorik i metrikës së mbulimit dhe të strukturës së shënimeve. Specifikisht, rrjeti refuzon ta klasifikojë një hyrje $\boldsymbol{x}$ kurdo që gjasa e sekuencës dalëse $p(\boldsymbol{y} \mid \boldsymbol{x}) < t$ për ndonjë prag t. Fillimisht, definicioni i $p(\boldsymbol{y} \mid \boldsymbol{x})$ ishte ad-hoc, bazuar thjesht në shumëzimin e përbashkët të të gjitha daljeve softmax. Kjo e motivoi zhvillimin e një shtrese të specializuar dalëse dhe të një funksioni të kostos që në fakt e llogariste një principled log-likelihood (gjasë-log parimore). Kjo qasje ia lejonte mekanizmit të refuzimit të shembujve të funksionojë në mënyrë shumë më efektive.

Në këtë pikë, mbulimi ishte ende nën 90%, dhe nuk kishte probleme të dukshme teorike me qasjen. Metodologjia jonë prandaj sugjeron të instrumentohet (aranzhohet) performansa e setit trajnues dhe testues ashtu që të përcaktohet se a është problemi

nënpërputhja apo mbipërputhja. Në këtë rast, gabimi i setit të trajnimit dhe të testit ishin afërsisht identikë. Në të vërtetë, arsyeja kryesore që ky projekt vazhdoi kaq lëmueshëm ishte disponueshmëria e një dataseti me dhjetëra miliona shembuj të etiketuar. Meqë gabimi i trajnimit dhe i testimit ishin kaq të ngjashëm, kjo sugjeronte se problemi ishte ose për shkak të nënpërputhjes ose për shkak të ndonjë problemi me shënime të trajnimit. Njëra nga strategjitë e debug-imit që e rekomandojmë është të vizualizohen gabimet më të këqija të modelit. Në këtë rast, kjo do të thoshte vizualizimin e transkriptimeve jokorrekte të setit të trajnimit të cilave modeli ua jipte konfidencën më të lartë. Këto dolën të përbëheshin kryesisht nga shembuj ku imazhi hyrës ishte prerë tepër ngusht, me disa shifra të adresës të hequra nga operacioni i prerjes. Për shembull, një foto e një adrese "1849" do të mund të prehej tepër ngusht, duke mbetur e dukshme vetëm "849". Ky problem do të kishte mundur të zgjidhej duke kaluar javë të tëra në përmirësimin e saktësisë së sistemit të detektimit të numrave të adresave që ishte përgjegjës për përcaktimin e regjioneve të prerjes. Në vend të kësaj, ekipi e mori një vendim shumë më praktik, thjesht ta zgjerojë gjerësinë e regjionit të prerjes të jetë sistematikisht më i gjerë se sa që parashikonte sistemi i detektimit të numrit të adresës. Ky ndryshim i vetëm i shtoi dhjetë pikë përqindjeje në mbulimin e sistemit të transkriptimit.

Në fund, disa nga pikët e fundit të performansës erdhën nga rregullimi i hiperparametrave. Kjo kryesisht konsistoi në bërjen e modelit më të madh gjersa mbaheshin disa kufizime në koston e tij llogaritëse. Meqë gabimi i trajnimit dhe i testit mbetën afërsisht të barabartë, ishte gjithmonë e qartë se çfarëdo të meta të performansës ishin për shkak të nënpërputhjes, si dhe për shkak të disa problemeve të vogla të mbetura me vetë datasetin.

Përgjithësisht, projekti i transkriptimit ishte sukses i madh, dhe lejoi që qindra miliona adresa të transkriptohen edhe më shpejt edhe me kosto më të ulët se që do të ishte e mundur përmes mundit njerëzor.

Shpresojmë se principet e dizajnit të përshkruara në këtë kapitull do të çojnë në shumë suksese të tjera të ngjashme.