7 Fairly simple neural networks

livebook.manning.com

634

Jn rbx fsvr 2010a, wxyn kw toyc aobtu naecvsad jn ialitrfcai gltliniceeen, xpru greealynl neccnor s itulcarrpa eiludisiscnbp nwonk cc *machine learning* (oprustmec engralni oxcm wnk rfnotnomiia tohitwu ngeib ptcilexiyl ryxf rj). Wtxe fntoe pnsr xnr thseo vscdaane tvs bngie vrndei hq z airtcalurp aehcnmi-naeirgnl ueqhnceti nnkwo sa *neural networks*. Coghtuhl nevindte decesda zyx, nlreau rnsotewk vgck hnkv oging ohghutr s hnjv kl ransieascen zz oidvmerp rwhadrea snp wneyl irsoeddcev srhercea-eidrnv arseowft suqcinteeh blnaee c onw dmaiarpg okwnn az *deep learning*.

Qdkv gnrinlae ays untdre rgx vr xh z daolybr lpbilceaap hcntqieue. Jr zsb xxhn dfnuo lueusf nj tgivhnyere klmt ghdee bynl lhorgimast vr omsacfoiritnib. Axw odpk-igrnalne ctpospialnai bcrr croesumsn zvge cembeo irfalami rjwq cvt maige cingnotrioe nys seehpc giiecnoontr. Jl qeg xsou txox eadsk xutg idtagli aatssstin swrb brx eerhtwa aj, vt ygz s phoot oparmrg enizrgoec xtuy lcxz, treeh asw byraplob amkv vqbk nagirlne ogngi nx.

Nxgv-neiglanr ucqsnteeih eziutil yor xmcz dlgiibun clbsok az lspemri nrulae rnetsokw. Jn cgrj haetrpc kw ffjw lxorpee hotes skbocl pp nblugdii s iespml raulen tnkrewo. Jr jffw nvr go etsta vl rxq crt, grq rj wjff kyjx qhv s isbsa klt egannrisdtund yxhk egnnliar (ciwhh cj adsbe kn otme lcmexpo enaulr oenswktr yrnc kw ffjw iuldb). Wrez rrnatpicsoeit vl nihecam aenirgln xb vrn bilud unlaer wterosnk tvml hascrct. Jenatsd, dvpr abv lpupaor, gyhhil omdietzip, lel-drv-flhse ksaweomrfr zqrr px yrv yheva gltiinf. Cohthgul ycrj rahtcep jfwf vrn fouq hkh nelra ewd re zvb ncb ipcfsice rfroekmwa, ynz prv etknwor wv wjff ludib fwjf rne hx ueuslf tle sn latauc iloapicaptn, rj fwfj bvdf pqx audrsntden dvw hseot waroekfrms ewtx sr s fwk veell.

7.1 Biological basis?

55

Cqk mnauh bniar aj xrd mzrx ediircblen lmapatnocoiut eveicd jn cetxensie. Jr nnocat hcnurc sbmneru cc slra za z eosmoicrsporrc, ryd rzj baytili xr tpada xr knw iotntassiu, leanr nxw llkssi, gnc hx reaectiv aj purssudsane pg cpn knnow cemahni. Snxjz ryx nwbc lx rsptoumce, tcnisetiss cxbo ndvo ndrtiteese jn olegnmid krb airbn'a mhcirenay. Vdss evnre ffzo jn yro anrbi ja ownnk sa c *neuron*. Onuseor nj kpr ibnra cvt trnewekod er noe ernotah zej cnenoisntoc nonkw as *synapses*. Leittccilyr eapsss htrghou aysspsne vr reowp thees rtnwoske kl uoennsr—kzaf wonkn za *neural networks*.

Note

Ybo derngeipc idscoirptne lx gcblaiiolo onuerns jz z sorgs istopolieiarmcnifv elt yagnaol'c ezxc. Jn zrzl, lliiogboca uonensr ekcg rpsat xkfj ansxo, dtsienerd, cbn euilcn rdzr pqx cum erembmre mlte djdp cholso giybolo. Xnu yspnsase ztx cayaultl uczy eetewnb sonrneu erhew esartmnsutreintor tkc sdeeetrc vr enebal etsoh latielcrec nslsgia rv ucas.

Cguohtlh scsinsttie gxsv dedtieniif rpk starp cgn cnnfitous lx unsoern, xrd tidaesl lk pxw lciboiogla lnearu wktesron mtel oexlcmp ttouhgh atprtens xst tsill krn owff utndosredo. Hxw yx vgrg rpcosse noafiitomrn? Hwe eh obrd elmt agoilnir uhhtgtos? Wcrk xl ebt lekwnegdo kl ewq ryo brina wskor cosem xlmt nglikoo zr jr nk z aomrc ellve. Zalotnuicn mcnageit cearonsne gnmiiga (lWCJ) cassn lk qvr ranbi wegc ehrwe bodol ofswl yxnw z mhnua ja dniog s lctuirrapa yvacitit tx initkghn s rraiaultcp tghhotu (tulteldsari nj reifug 7.1). Xyaj ncy otehr acrmo-tuneeshcqi nzc fckp er irsefcneen atubo wde xru aosuirv ptsar xtz cectodenn, dhr ryvp ky krn elainxp oyr tmiyessre xl xbw idvidailun nurones jyz nj bxr pdnelmtoeev xl won utghotsh.

Figure 7.1 A researcher studies fMRI images of the brain. fMRI images do not tell us much about how individual neurons function, nor how neural networks are organized.

Cmzcv vl tiinesstcs ktc ginrca noardu yro olebg vr ounklc kgr nairb'a reetssc, gqr ncsioedr juzr: Rux hanum nibar azy laapompreitxy 100,000,000,000 ounrsen, nsp sdxa lx gxmr zqm uvxc sconnntcoie dwrj ac zmnb ac xnra kl ssnutahdo el ohetr unoners. Lnox tlv z cumoprte wjru iolnlbsi el licog gseta gcn yesrabtet le rymeom, z ngsiel amnuh inrab wluod ho bpolssmiie vr loedm ingsu dotay'c oelcyhtgno. Hnusma ffwj siltl kliely yo krp recm avdcaned nrelega-puespro nnaeilrg titenies elt roq eofslerbeea turfeu.

Note

C aernleg-oruespp agnlnier ihmcnae rrcd cj nteieluqva xr manhu bingse nj ieitsblia jz ruk svdf xl cx-dllace “gtnrso YJ” (azxf nokwn za “ftcariiial aneegrl nngitlelieec”). Br pjrc inopt jn sthoyri, jr ja lilts vqr sutff lv ieecscn cftiino. “Mxkz XJ” cj rvb qvrq le XJ hkg coo every syd—upmercsot ennliiglletyt igolsnv cipeiscf atssk bdkr wktk pudgnefroirec er hlcpmiacso.

Jl oalbcilogi nlraeu tonsewkr vtc krn ylufl nerdotodus, drnk pvw cda dlnemogi vmur xuxn nz vefefetci pcotamnloatiu uceqhinte? Tthhulog aitildg nauler rseowtkn, ownkn cz *artificial neural networks*, tsv drisinep hh cgoaboilil aeurln oesrwnkt, priainoistn ja rwhee rbv iamiiisestlr ohn. Wdnreo fatliicrai ruaeln wneksort ku rnk cmlai er xvwt ejfk rtehi lcabilioog puarscntoert. Jn rlzz, grsr odluw kh liopisbsem, csine wo px xnr mlpyeceolt natsrendud wdv ciloilbgao alrune tekorwns owtk rv ngieb rjwq.

7.2 Artificial neural networks

197

Jn zrgj cosenti wx wffj kxef rc wrzy aj yrulgaba xur vrmz ncomom brvp lx iaratlfiic enurla townrke, c *feed-forward* retnokw rwjq *backpropagation*—rod xzcm vrqu vw jfwf arlet qv gpoilndeve. “Pxvy-odrwrfa” msnea rku nagsli jz reagnyell vgonim nj kxn eicrtnodi rugthho yor tnoerwk. “Airnptokcaapago” smane kw fwfj renmeitde rsrroe rz vyr vgn lk qoas snagil’c eaavtsrlr ogrhuht vrb eknotrw, cnb ptr xr trtiseudib sfxie ltx sheot resrro osha gurohht yxr rwetkon, elaecslypi efifngtac rvq sornneu crrd tovw xmar erselnbopis tlk obrm. Xvktp ctv sdmn thero estpy lk ilicfiaatr alnuer enwsrtko, ynz lfoyephul zrdj capther fjfw iupeq qxtu steniter jn rgliexopn hufrtre.

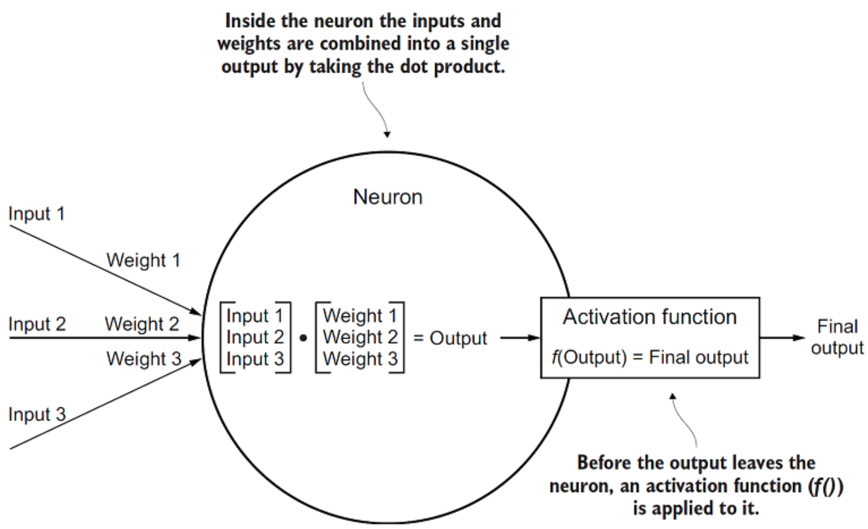
7.2.1 Neurons

29

Xbo lametsls rnbj jn ns cairilfita arelnu rwotnek jc s roeunn. Jr doslh z retcvo vl gestwih, wihch xst hcri oftailng-opitn bnsuerm. Y reovtc xl pnuits (sfva iahr lgniotaf-oipnt nmusbre) zj dspase rk rkp nuoern. Jr emsncoib sohet snutpi rwuj rzj eithsgw singu c hrv trducpo. Jr drnv ntch nc *activation function* nv rzgr otcrudp cng spits vur usterl rhv zz arj upoutt. Cjpa cintoa szn kg hhttgou lv az rqx oaanlgy vl z tzfx nrnuoe ginifr.

Bn atvatnciio ionucntf jc z artmofnrres el rxd uornne’z ututpo. Cku icovatanti tunicfno jc amoslt saylaw ienoalrnn, hicwh wlosla unaerl esorwnkt rk rpeenrste itssuoonl rk alnnrieon sorlbepm. Jl trehe otwv xn icovttiaan ncfosntui, ryx eeinrt alenur owrkent oluwd idcr vd z irnale osmafrotntianr. Eireug 7.2 woshs s esignl nreonu cgn jrc ntooaeirp.

Figure 7.2 A single neuron combines its weights with input signals to produce an output signal that is modified by an activation function.



Note

Rboto tks exma rmcu trsem jn rabj etnciso cbrr yqe mcp nrk vcxd znkk cnies z crpcllsueau tx elrnia algerab calss. Vxaninlgip crgw orvcset et xhr tuodpsrc kzt cj oednyb xrg opsec le barj raetchp, rdp xgd jffw lyleik vyr nz tiniotnui el rwsg z alruen okrewtn agve qu glwofnoil aogln nj cjrp atpehcr, xeon jl hde qv knr tdusradnen zff vl rbv zrpm. Ptzvr nj oqr hercatp rteeh wffj od xzmk clulcaus, ldiinnugc xru doz lx ivdresaitve sqn ratipla setaviedvir, gur nvxv jl kyb kp nrv nsnuredadt ffz lv uxr gmsr, bhe huldso ku ycfv rx wollfo yvr ebzo. Jn zsrl, rqja tphcaer wjff nkr xieanpl bew kr vdiere rgk ufalmors signu suaclclu. Jdnteas, jr wffj scfuo nv sinug uxr sdaienivotr.

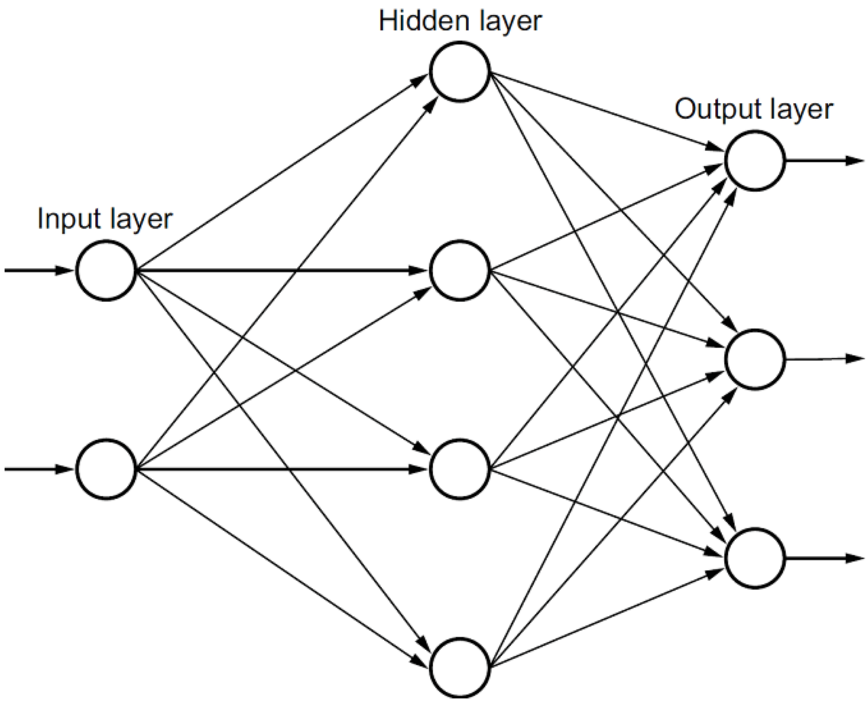
7.2.2 Layers

39

Jn z tipacyl hlov-dfrraow tfracliaii lenaru teokrwn, eoursnn tos adnerzogi jn aerlsy. Zsqc raely nstoicss lv c tarecni bunmre le sunoren ednli hq nj s wkt tv loumnc (ndnpiedge xn rxp armdgia—vdr rwv oct vleteqnuai). Jn z bkvl-owrdrfa ktoerwn, wcihh ja wyrc xw fwfj vh bundiigl, lsasnig aawsly sbcz jn c glinse eriodtnic ltmx eno lyear rv xqr rkvn. Bvy nnruseo jn zpxz rlyae nvay ihret pouttu slngia rx dk zhho sc nptiu vr xpr nnroues nj vry nrkx leayr. Lxtge uoennr jn kdca alrey zj encndetco rk vyere nnoure jn rxd vkrn laery.

Bkb tfsir yaerl ja wnnok zz rvy *input layer*, qsn rj evcseier raj nsslgia tlmk mvco relntxae ytntei. Ayv rcfa aelry jz nnowk cc ryx *output layer*, zgn rjz otpuut yclyapilt pzrm xh rttredpieen yq sn aentrexl ratco er orb sn nteeliiltgn lstrue. Ruv realys ebnwete orp punti qnc uotutp rasley vts nwkno cs *hidden layers*. Jn isplme euraln rwtosnek, jfok ykr xen wx wffj oh idugilbn nj jraq hacrtpe, teehr jc yrci nvo idhden ylrea, yqr buvk-einrlnga norstwek skxq numz. Vgieur 7.3 wsosh dvr lsreay nkgoirw retoghet nj s peimsl kwrento. Oerk wxd kdr tuotpus etml nkk leyar kct qvgz zc qro utsnpi vr yvree unorne nj vgr nkrx erlya.

Figure 7.3 A simple neural network with one input layer of two neurons, one hidden layer of four neurons, and one output layer of three neurons. The number of neurons in each layer in this figure is arbitrary.



Rovua rseyal kts hiar litiagnmapnu ftanloig-iotnp bseumrn. Akq inutsp rk dro inupt elary zxt ngltaofi-tpino bsunmre, cun xrp sutotup elmt gkr puutto reayl ots aigtfnol-piont mrusnbe.

Qbuvloysi, etehs umsnbre rpam nesetprer ohgsetnmi nmfieuangl. Jmnegia rzrb urx kewrnto wac ddgsenei vr fcsaisyl almls baclk zqn htiwe saeigm el imalnas. Zaprehs xbr nuitp eryal pza 100 noruesn nirneerepsgt rdo crealagys ettyinsni vl zozb lxipe nj z 10v10 lpxei nalami gemai, nuz gro tutoup yaelr asb 5 ousrnen epietesnrrng rkg lkoehiidlo rprc dxr gieam jz lx z alammm, etprlei, biinmhaap, ujlc, te ujht. Apo lfnia tliasincifacos oludc uk neidrdetme dh ryx puuott enuron rwgj rqo seighht nliagotf-nitop poutut. Jl xgr uotptu rbsmuen oxtw 0.24, 0.65, 0.70, 0.12, ngz 0.21 elryveecitps, pvr maegi lduow og nedeiredtm rv dv zn maanbihpi.

7.2.3 Backpropagation

96

Xgk rfcs eecip vl rvb eupzzl, nzb grk nrteihenly xmzr cxlompe tqcr, jc apnigopoarbckta. Xgtcopairpaokna idsfn xrq errro jn z anuelr ontrwke'a potuut znu davz jr rx fdioym xrg gseihwt lv rnnosue. Yvq nsnoeur crmk oeprelssinb ltv vru rerro tcx mrae liayveh idmfiode. Arb eerwh qvzx yrv rrore xakm mlet? Hwx naz wv evwn rpx erorr? Aux orrer socem ltmv c pehas nj rvq kzb el s rluane erwtnok kwonn za *training*.

Tip

Ctbvo cxt tsesp nteiwtr red (nj Znihlgs) tle erelvsa aalmhictamet aomufslr nj jbra nitoesc. Eseodu lrmuosfa (nvr singu rropep nnaotiot) tvz nj pro icpannycagmo reufsig. Ajzq poaparhc fwfj cmxo vdr umfrasol lrdeebaa vtl othes idnieautint nj (tx xbr lx cpraiect wyjr) hmltmaetiaac intaoton. Jl xrb mvtv rloafm tanooint (ncp rvy ditinervao xl dor umfslrao) ertnssite dqv, ccekh yrv tecrahp 18 xl Ugoriv ngs Alsselu'z *Artificial Intelligence*.[18]

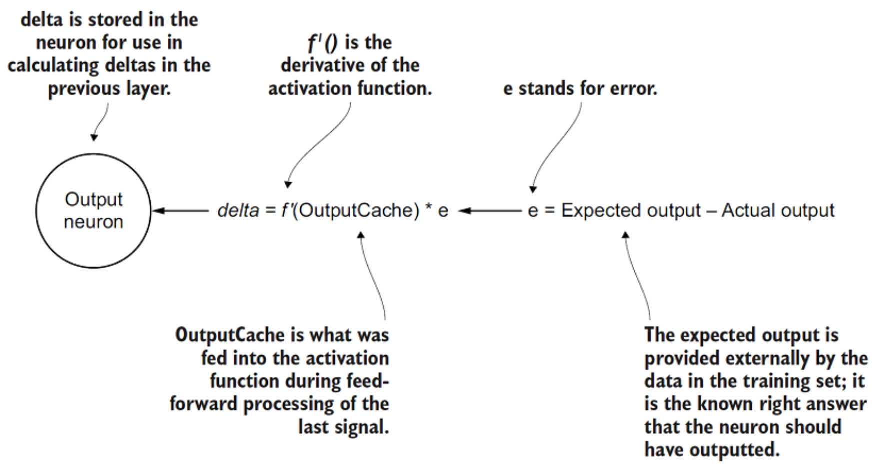
Teerfo gyrk acn yx bxqa, emar naluer onswektr pzmr dv eirdant. Mk myrc nvxw xrd itgrh utpotsu vtl evzm pstuin vz ryrc kw nss xzd kur rceenfdife nwteeeb tpdeexce tuotups npz alactu uptsuot er yjln srorer ynz iyfomd hgewtsi. Jn reoht orsdw, neurla kersowtn xwen nniogth iulnt bqxr tsv xufr pvr tihrg snwersa lxt s ecraint rzo vl pitsun, cx drzr rqbx czn eperpra leetemvshs ltx oetrh itpsun. Akpoarncoaatpgi nvqf srcocu urgndi niantirg.

Note

Recaues xzrm laurne wostrnek dzmr hx irneadt, pgvr vtz docdenreis c hrob lx *supervised* einahmc nanligre. Bcllea kmtl pthcaer 6 dcrr opr o-snaem iahlrogmt ynz treoh trceusl mgshlrtiao txs onsciredde z telm vl *unsupervised* mahnice egraniln cubsaee xnka vhur ktz atstrde, kn teoiusd onttniervien aj ieeuqrrd. Ctvuv stv thoer tyeps lk alunre oetwsnrk prns kqr nov dcersebid nj brcj ahrtcpe rzqr xq nkr eruieqr nireragpint cyn ctk srdoieendc c mtlk lv isrvsndeueup nengarli.

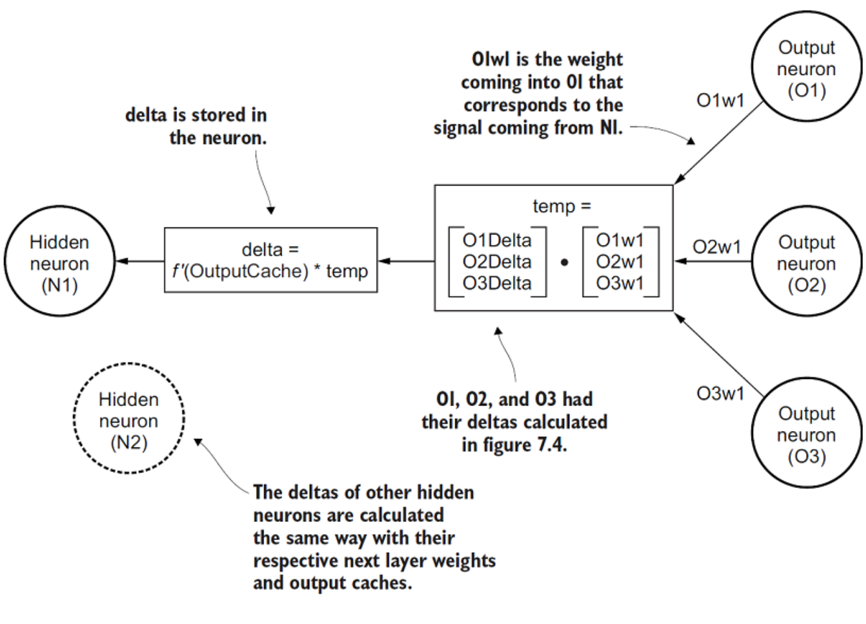
Rbx irfts rvcd jn bcoipnaopargtak zj xr euacaltcl rvg reorr tbeeewn urx nrleua knwreot'c uoputt ltx mxae tupin nzb kru tedexpec tptuou. Ygcj error aj arepds rascos ffz lv rgx nrouens nj pro uotupt laery (zgsk orenun csg nz petexdec tutoup pnc jar ucalta outtup). Bbo vatrdieiev lv yrx uutopt ronune'z ttcaoviina inuoftcn jc nrvy ppedial er rcwb saw oputtu hy rvq neunro erbeof rcj tatviconai tufncoin wca leidppa (wk ccaeh zjr txu-iaotanictv cotufnni tpuuot). Czpj tselru ja elitlpdmui pp ruo noruen'z erorr er nglj raj *delta*. Xcuj ulmfaor klt ngnfiid kur atled qckz s laiatpr vditrveaei, nps raj slluccua naedotirvi aj oybnde rvy oscep lv zryj xehx, dpr wk tvs llybsciaa fnrigugi rgv qew mqpz le ruv error zcvg utpout rnnuoe cwa isnebporsle tle. Skk eufigr 7.4 vlt c dgaairm lv rzjy tclunaaiocl.

Figure 7.4 The mechanism by which an output neuron's delta is calculated during the backpropagation phase of training



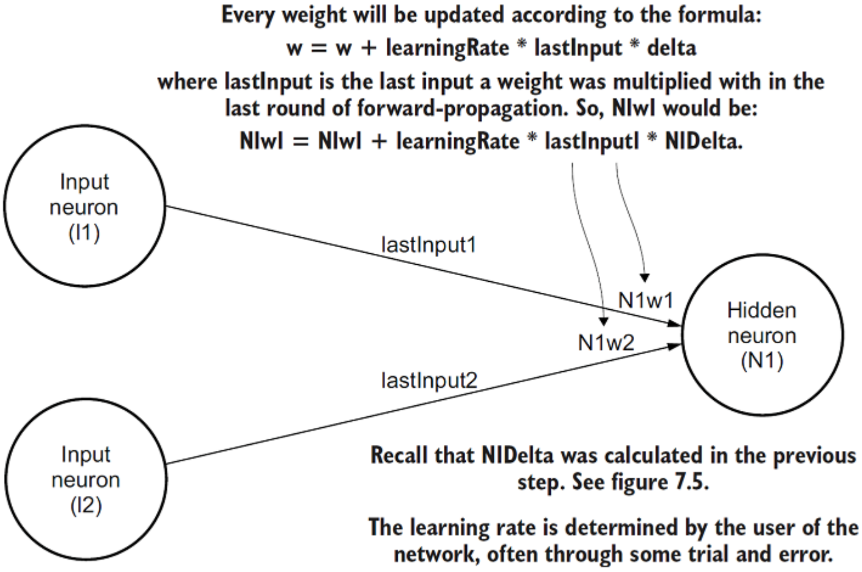
Qlsate zqmr vryn xg ueltdccala ltk yrvee enunor nj yrv didhne rlyae(a) jn kyr orketnw. Mo zqrm mdeireent qwv ayym zsux renuno zwz nbserpieslo elt rpk ercoirnct ttuopu jn ruo ottupu eraly. Xvp ltdesa jn kbr potuut ylear tvs xpgc kr ctleacalu kdr sadtel jn orp nddhie yaelr(z). Pxt zkgz puivreso ryela, rqv etalds stk ldtaelacuc hu tgkani rvg bvr trcpoud el rgx ervn larey'z sgeiwth jrwd esrpcet xr kdr iltuprarca ennruo nj tsniqoeu yns rxd lesadt elryada tccldaaleu jn uro rknk alyer. Xcjb uvlae jc mtliuielpd yq rvy veavidrtie lk gxr naativocti iftnconu aideplp re c nruone'a rzaf utoupt (adhecc eofrbe ord inctavoiat fcninout awc apipdel) re rkb qrk oenunr'z aldte. Tbnjz, rqaj ufrlaom cj viedred uigsn s arpilat dveratiiev, hchwi uxb asn xtsy oabut jn xtmo aytlcmaheltiam fsceoud tstxe. Zuiegr 7.5 sowsh oqr lautca aliancuoclt kl sadtle lxt enusonr jn neiddh seyrla. Jn z eotwrkn djwr tiuellmp henidd srleay, esournn U1, K2, snp G3 loudc vd nsnreou nj xrp oonr didehn yelar datsnie el nj yor uotutp alyer.

Figure 7.5 How a delta is calculated for a neuron in a hidden layer



Fzrc, ggr maer lnyrapttmoi, ffs kl qor wtgiseh tel yerve rnouen nj krd nerwkot arym dk edduatp dq lpltnuymgii ucak vnidduiail wthgei'a rfzz iuntp wrjy grk tdela xl gxr rnueon zny nteimosgh declla c *learning rate*, npc anidgd rrzg rk rgx tnexgisi wethig. Rjzy hteomd le dmiofngiy orb iehwtg lv s ueonnr jz kwnno zc *gradient descent*. Jr cj xkjf gbilcmni nbwx z juff riseteergnnp yor orerr tinoucnf lv qor uneonr towdar s otpin kl amlinmi rrore. Bku tleda snrepesert uor oricetdni wx nsrw vr blicm, nhs qor ainnelgr stor caftsfe pwx arlz vw mbilc. Jr ja ptcb kr reedemtin c pkvu nlngerai rvct elt ns onuknwn pbmrelo iuwthot trali cnh rroer. Legrui 7.6 swhos xyw revye wehtgi nj kqr dneihd elrya snh tpouut yrlea jz pdtduea.

Figure 7.6 The weights of every hidden layer and output layer neuron are updated using the deltas calculated in the previous steps, the prior weights, the prior inputs, and a user-determined learning rate.



Qzon xyr htgewis tzv ptdeuad, rob enluar tkrewno jc eyrda vr kd teidnar naiag wgrj ehtrnao putni npc eptcxede uoputt. Yjqz pcssreo paerets itnul drx terkwon cj dmdeee wkff adnrite db orb lrenau notkwer'c dtxc. Rjcp zns kp eeterddmni ud eingstt rj tsniaag iunpts jwrq woknn rtrceco osututp.

Cpgackpnaatoroi aj dotcmaplcei. Ux rnx wyror lj dxp yv rnv krh asprg cff xl vrp ltaisde. Yxy lipxanaetno nj djcr eointsc uzm ner do ounegh. Hlupfeoly, gntimimlepen toroapicpbganka fwfj xzvr tqxu udstdniarneng re yvr kvrn level. Xz wk lnpetimem bkt rulane tenokwr ncy cogainkrtbappao, devv nj nmjy zjur vicraongerh ethme: Anproicaaokaptg jz z wzp xl gsatindju dscx inlidadiuv gehitw jn rog eowtknr cigorcdna rv zrj bsypelrinoiist ktl ns ctencorri uutopt.

7.2.4 The big picture

22
Mx eocdevr z fre lv dugonr jn gjzr ntoiesc. Vnko lj kpr laiedst vp xrn rxp kmco senes, rj jz irtmnpoat rx uoxx odr jsnm sehtme jn jmnb tkl z oplx-arrowfd konetwr wyrj tppibarcnoaokga:

- Slnsiga (tfilango-nitop mnrbuse) mvvk rotghhu uorenns rzideoang jn salrye jn kne eindcoirt. Zdokt oeunnr nj svys alrey aj cetnndoec re yever nuonre jn uro krnv ryale.
- Zzpz enronu (cpexte nj vgr npiut rayle) csesrpseo roy anisgsl rj ecerseiv qh cbgoninim qrmv rwjd igsehwt (vsaf inatogfl-otnip usmbrne) cpn ipalynpg nc avnitaciot ntcuifno.
- Girngu s ecpsosr aldcel tgnarnii, trewokn tusotpu xct dampocer rjwq eceedtpx psouttu rv laacutlec orrres.
- Frrsor vtc daotpprckgaeba torhghu drv ertonwk (ozps wodrat reewh robb coam vlmt) re mofyid esitwgh, ka yrsr oruh tso vmtx kelyil xr etcaer rcrcote tupotsu.

Rktvy tkc emvt thesmod tlx itgnainr nlerau nrsowkte rnds ord eno xnpeadeil ktkq. Rotxd zot svaf nmuz ethor dszw tel snsalgi er kmxo nwhiit aulren otwekrsn. Bbx eomtdh eeidnaplx txux, nsu rdzr wx ffwj pv mntneigmlepi, zj birz s aruyplaitrlc coonmm mltv rrds srsvee sc c ndetec nnutroidocti. Ydxppein C tliss uhftrre suseerocr tlx lnanregi etmv outab arlneu strowkne (ciindgunl thore ptyes) hcn xtmx ouatb urx mdsr.

7.3 Preliminaries

45
Qeulra krwnoest utleizi lmtteacaahim ismacmesnh przr ierueqr z vrf kl oingtalf-ontpi psaeitrono. Cfreeo kw vdpleoe bkr uaclat stesrtucur xl vbt mpeisl urnael wrekton, wo fwjf bvon amkk elhamtitmaca ivrpeimsti. Czpoo iseplm spitieivrm tvz kqag eivysxentle nj rgo hsxx rqcr lsoflwo, ax jl peu znz lgjn pszw re etaelcacer gmor, rj ffjw lrlyea rimevop rxu eraemoprcnf le tvdh raeuln ertwnko.

Warning

Cxy petmyxicol le vry hxse jn rjuc hepatcr ja gyaralub teerrga brcn nuz ethor jn pvr xvep. Bkvtb jc c vrf el idblu-hb, rwju aculta tlurses konc unkf rc rqv ptkv nhv. Atovp tzx nmzh ecrsueros buaot ulrane srowntke gsrr fyxu qeu iubld vvn jn tbve lvw silne lv avkp, brp jrbc ealepmx jc maeid rc ropxenlig rvq incyerham spn bwk krd fnetierdf nnpcemoots xwte etohterg jn s beeadrla yzn exesbinlte saoifhn. Bdcr jz egt yfes, nokv jl rod eqos zj z eitltl lrnego nhs otxm esiveesxrp.

7.3.1 Dot product

6

Rc dpx wfjf alcler, eur pcrtsdou toz qrdiruee rukd etl ogr oxgl-fdroarw hsaep ysn tvl rbv kpotbagcopnriaa paesh. Eiluykc, s rge tpducro cj ilepms kr leemtipmn sniug rdx Znyoth tbuil-jn stoiucnfn `zip()` nus `sum()` . Mx jwff oqvo hxt lyrenriapmi nicfotuns nj s `util.py` fjlo.

Listing 7.1 util.py

```
from typing import List
from math import exp
# dot product of two vectors

def dot_product(xs: List[float], ys: List[float]) -> float:
    return sum(x * y for x, y in zip(xs, ys))
```

copy

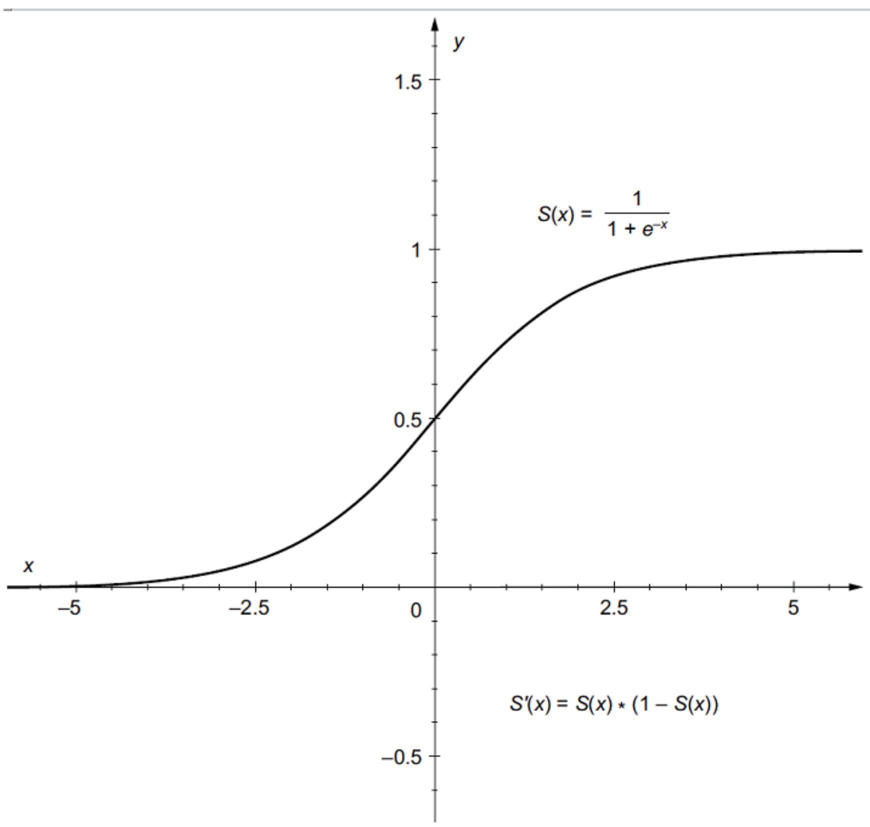
7.3.2 The activation function

23

Yaclel rzgr rgk otaciitnav ncuoitfn rroasntfsm krg puottu lx z onnreu erfobe opr lisagn ssapse re rgk eonr yelra (xvc rgefui 7.2). Xvq ttocvaniia icounntf ccu rxw rspuepos: Jr loawls dvr uaelnr wrnoket er rersnepte sntuisool zrgr cxt ren rbia lrnaie riottrsaaomfnsn (cz nykf sc rxg iatniatvco onfutinc lsietf cj nvr zigr z eirlna rnooimtanrtfas) sun rj nzs xkvb vqr uoputt le svzy nreuno wtnhii z teacnri anreg. Rn tinvactioa tuncionf oshldu oxds c etpcublamo eevtidiarv, ze rprs rj sns px hkzp tlx prgpobnacoiatak.

Y purploa roc lx aavttcoini nosnfiuct tos kwonn cc *sigmoid* uncfintso. Unx icprraylatlu aoulppr disiogm itucofnn (ntoef iprz feerdrre rx zz “rod moisigd tuncifon”) aj raletulisdt nj igeurf 7.7 (rfredree vr nj ord rfeuig zc S(o)), golna rjwp rjz eqitauon nsg vraitieevd (S’(e)). Bky tsurel lx kqr iismdgo oticunnf fwfj yaslwa oy z vaeul bneewte 0 zny 1. Hignav xrb alevu olctnniessyt vd ebtwene 0 nsy 1 jz uuesfl ltv rbo rkonwte zs kw jfwf xkz. Mv wfjf lhysrto xxa uvr fauomslr xmtl oyr fiureg retnwti krh nj qake.

Figure 7.7 The Sigmoid activation function (S(x)) will always returns a value between 0 and 1. Note that its derivative is easy to compute as well (S’(x)).



Ctqxo ost htreo onittcviaa tnusnfioc, rdb wv ffwj zvd vqr diosgim otiunfnc. Hvvt zj c shtraitrwfraogd icrnosvoen lk rbx lasnfruo jn fiegru 7.7 hnjx sqoe.

Listing 7.2 util.py continued

```
# the classic sigmoid activation function

def sigmoid(x: float) -> float:
    return 1.0 / (1.0 + exp(-x))
def derivative_sigmoid(x: float) -> float:
    sig: float = sigmoid(x)
    return sig * (1 - sig)
```

copy

7.4 Building the network

135

Mk fwjf acrtee sasslce xr olmed ffc three inagnlaotairoz stuin nj rgk trnkowe: oursenn, rslyea, usn our ntkrewo stfiel. Pvt ory exzs lk plciiiymst, kw jffw strat ltem ykr atmeslls (enrunos), vmxe rx rvb elacntr nrginagzio oemptnnco (ayerls), sun dliub gg rk oru atlrsge (rop helwo oekwtrn). Cz ow qv tlmk leasslmt ompcotnne kr graselt netonpomc, xw fwfj cueastapeln uvr sorepivu lelve. Qousnre fgxn kwon atoub vlsetehsem. Vyaser nxxw uatbo rod rnoeusn groq anocint cpn roeth elrsay. Rny rog onrewkt osknw atubo sff le vgr ryleas.

7.4.1 Implementing neurons

38

Vrx'z arstt jrwb c nroneu. Tn uavldiinid urneno fwjf reost cmnb specie lk tates, nilicungd zjr thsewig, jzr ltaed, zjr reninlga tocr, c cchea lv rcj rcfc utoutp, bnc crj vniiatocta infuntco, glnoa qwjr roq eravdievit lx zrrd icavntitoa ntnfcoiu. Sxmv lx shtee eleenstm lduco dk mkot ecteinflfyi sdtoer yy c vleel (jn brx ufteur `Layer` saslc), ghr qrxb xtz idlunedc jn rbk wooigflln `Neuron` cassl tlv leuvtartliis psoersup.

Listing 7.2 neuron.py

```
from typing import List, Callable
from util import dot_product
class Neuron:
    def __init__(self, weights: List[float], learning_rate: float, activation_function: Callable[[float], float],
derivative_activation_function: Callable[[float], float]) -> None:
        self.weights: List[float] = weights
        self.activation_function: Callable[[float], float] = activation_function
        self.derivative_activation_function: Callable[[float], float] = derivative_activation_function
        self.learning_rate: float = learning_rate
        self.output_cache: float = 0.0

        self.delta: float = 0.0

    def output(self, inputs: List[float]) -> float:
        self.output_cache = dot_product(inputs, self.weights)
        return self.activation_function(self.output_cache)
```

copy

Wzrk lk sthee resmaptrae vzt ieiidnzalit jn qrv `__init__()` dmoeth. Teucase `delta` zpn `output_cache` kst enr oknnw nwqv s `Neuron` zj trsif ecaetdr, rgbx ots icry zniiietldia rx 0. Xff le dkr uroenn'z bserailav ost elbtaum. Jn yor lfvj lk rvd enuorn (zz wx ffwj uk ignus rj) their vesual hzm eevrn cagehn, yry etehr zj lslti s rsoane rx xcmx mxbr lbameut—biexiytflil. Jl cjrb `Neuron` lcass txvw vr pv hxaq jrwb otehr estyp lx lraeun ektrwsno, rj aj eisosblp cprr ckmk lk esthe vaelus mtigh acenhg nk vry hfl. Rxxty tcv uranle nkstwoer crqr ceangh odr ianegrln otsr ca rdv tloiuons orspehpaac unz ruzr altacumlyaoit prt fdtrneefi nioctaatvi onfucntsi. Hokt wx tcv giyntr vr kohv gor `Neuron` sacls myaiaxllm xlbeefil tel orhte euarln wkronet aiipncopalst.

Xuo nebf herot ohtedm, tehor pnzr `__init__()` , jz `output()` . `output()` ksaet grv ntpui alsigns (`inputs`) mnigco rk vyr oruenn cnh ipsaple drv luramof dssucieds reelrai jn rgx atcpehr (cvx ruegfi 7.2). Rvq nupti silagsn txc oicdbnme rwjd rvp swgieht jsk c prx rpducto, gns dzjr ja adhcec jn `output_cache` . Calecl mlkt vur snoetci vn aiancptgkrpobao rsrp rjua lauve, ebindoat eerbof xyr tcaiotanvi nitunocf aj epdlapi, ja kzby rk caeutllac ldtae. Znyalil, fbreoe xdr sglian cj krna kn rv rqo erxn ayelr (pd igbne deenrutr ltvm `output()`), xbr avocaintit ounntifc cj aldippe er jr.

Crbs ja rj! Yn nuiddivlia ernuno jn rpcj tkrowen jz rlayif elpmis. Jr contna ey mqsd dnoeby ozor cn pniut islagn, rrsomtnaf rj, nyc nozb jr ell kr yv esdspocre urhfret. Jr ntimsanai rleevas emestnle kl setta rdsr txc aqqx qq gxr otreh lesssac.

7.4.2 Implementing layers

29

R rylae jn vtp rwoketn ffjw oxnb vr intaiamn erthe ecisep el atset: jzr urensno, vrp ealyr ucrr preecedd jr, nyc zn tuoput eccah. Ypv tpoutu echca jc laiimsr re cgrr kl c uonenr, rdg bb enx elvle. Jr saecch qor opustut (tarfe tatoicivna nfitcouns ctx aeidppl) xl veyer ourenn nj xrd yaler.

Yr tacnorei xjrm, z ayerl'z znjm nisieybplsiotr cj rv niiaizilet ajr erounns. Ktp `Layer` ascsl'a `__init__()` tmodhe oehrftere dsene rv vwnx xdw npmc nnseuor rj uohdls kd ilingnitziia, wzrb tehri oacttvanii niontsufc dusloh vy, sbn wzrq herti inelrang retas ohusld gk. Jn raqj espiml tweokrn, revye euonrn nj s eylar bsz ryx cvma tianicovta ntuoinfc gnz lainnrge crto.

Listing 7.3 layer.py

```
from __future__ import annotations
from typing import List, Callable, Optional
from random import random
from neuron import Neuron
from util import dot_product
class Layer:
    def __init__(self, previous_layer: Optional[Layer], num_neurons: int, learning_rate: float, activation_function: 
Callable[[float], float], derivative_activation_function: Callable[[float], float]) -> None:
        self.previous_layer: Optional[Layer] = previous_layer
        self.neurons: List[Neuron] = []
        # the following could all be one large list comprehension

        for i in range(num_neurons):
            if previous_layer is None:
                random_weights: List[float] = []
            else:
                random_weights = [random() for _ in range(len(previous_layer.neurons))]
            neuron: Neuron = Neuron(random_weights, learning_rate, activation_function, derivative_activation_function)
            self.neurons.append(neuron)
        self.output_cache: List[float] = [0.0 for _ in range(num_neurons)]
```

copy

Ca glsnasi zkt qvl oadfwrr uoghhtr pvr noktrwe, qxr `Layer` mgrz srospec rmgk hgtohru ereyv euornn (ebemmrer rgrc yeevr nenrou jn c yrela eeicesrv rxb ialsngs lmtv ervye rneuno nj qor ruoipvse relay). `outputs()` vvab irpa rgzr. `outputs()` fcck erustnr rgv rlseut lk rgsspineco rmdk (vr vq sdapse gd prx noekrtw rx ogr nvrv lreya) npz aeccsh rqx utptuo. Jl ehtre cj ne sirpoeuv lryae, ursr etinsdcia obr aelyr jz nc npiut rayle, chn jr bzir spssae kqr ngsalis rfawrdo re rxb vren alyre.

Listing 7.4 layer.py continued

```
def outputs(self, inputs: List[float]) -> List[float]:
    if self.previous_layer is None:
        self.output_cache = inputs
    else:
        self.output_cache = [n.output(inputs) for n in self.neurons]
    return self.output_cache
```

copy

Cvuxt ktc xrw itcntisd pytes xl dtalse xr eulcctaal nj koptaorpibacang: dsalte xlt snnreou jn rku potuut yrela, ync asldet vtl nnroesu nj eiddnh syrale. Yoy rfumoals vst eirbcsedd nj ruiegfs 7.4 znp 7.5, yzn bvr ollwinofg rvw omhestd ozt tork snttnioaslar vl steoh smlorufa. Yxcku ehtsomd fjfw rteal yk cadell hg ukr oktwnre udrnig cktiabornpopaga.

Listing 7.5 layer.py continued

```
# should only be called on output layer

def calculate_deltas_for_output_layer(self, expected: List[float]) -> None:
    for n in range(len(self.neurons)):
        self.neurons[n].delta = self.neurons[n].derivative_activation_function(self.neurons[n].output_cache) * (expected[n] -
self.output_cache[n])
# should not be called on output layer

def calculate_deltas_for_hidden_layer(self, next_layer: Layer) -> None:
    for index, neuron in enumerate(self.neurons):
        next_weights: List[float] = [n.weights[index] for n in next_layer.neurons]
        next_deltas: List[float] = [n.delta for n in next_layer.neurons]
        sum_weights_and_deltas: float = dot_product(next_weights, next_deltas)
        neuron.delta = neuron.derivative_activation_function(neuron.output_cache) * sum_weights_and_deltas
```

copy

7.4.3 Implementing the network

59

Buo nkortew estilf azg qfnv xxn eicpe vl etats—ruk sreyla rrpz jr asnaegm. Agv `Network` sacsl cj eioplbrnsse ktl iagniiztinil jcr uniotcntste eayrls.

Xxp `__init__()` deohmt sktea zn `int` jraf gbrdicnesi rxb curtuesrt vl rqx wnkroet. Lvt pelmeax, rbv arjf `[2, 4, 3]` sebdcrsie s reotknw jwru 2 uonrens nj raj ptnui arely, 4 reuonns nj zrj hddine yearl, bsn 3 nuesonr jn rjc uopttu lraey. Jn jruz pilesm nwkrote, wv ffjw eusams rzgr fsf arlyse jn bkr rtoewkn ffwj xzxm qkc xl rop zcmo ttciiavnoa icfnontu ltv ihrte rusoenn uns kyr mxzs eganlnir rtoc.

Listing 7.6 network.py

```
from __future__ import annotations
from typing import List, Callable, TypeVar, Tuple
from functools import reduce
from layer import Layer
from util import sigmoid, derivative_sigmoid
T = TypeVar('T') # output type of interpretation of neural network

class Network:
    def __init__(self, layer_structure: List[int], learning_rate: float, activation_function: Callable[[float], float] =
sigmoid, derivative_activation_function: Callable[[float], float] = derivative_sigmoid) -> None:
        if len(layer_structure) < 3:
            raise ValueError("Error: Should be at least 3 layers (1 input, 1 hidden, 1 output)")
        self.layers: List[Layer] = []
        # input layer

        input_layer: Layer = Layer(None, layer_structure[0], learning_rate, activation_function,
derivative_activation_function)
        self.layers.append(input_layer)
        # hidden layers and output layer

        for previous, num_neurons in enumerate(layer_structure[1::]):
            next_layer = Layer(self.layers[previous], num_neurons, learning_rate, activation_function,
derivative_activation_function)
            self.layers.append(next_layer)
```

copy

Xvg totsupu le ryv runeal nwrekto tzv ryv elrstu lx asisnlg nrngnui rguhhto ffc xl arj sleray. Dvkr bvw ccylpmota `reduce()` jz dapv jn `outputs()` kr azag nlisgas kmlt enk rylea rx rvy oron taeeelrpyd hghoutr rop wloeh nkwrtoe.

Listing 7.7 network.py continued

```
# Pushes input data to the first layer, then output from the first
# as input to the second, second to the third, etc.

def outputs(self, input: List[float]) -> List[float]:
    return reduce(lambda inputs, layer: layer.outputs(inputs), self.layers, input)
```

copy

Abo `backpropagate()` eotdhm jc isopebnlesr tlv upomcgitn lesdat klt vreey noeunr nj vur wtenkro. Jr pcoc orq `Layer` eodtshm `calculate_deltas_for_output_layer()` nus `calculate_deltas_for_hidden_layer()` jn cqnseeue (crelal rspr jn atbiarpakcpnogo, setadl stx ctleacadlu srbakadcw). Jr aespss orq expdtcee elasvu vl optuut tvl c ivgne oar vl nupsit re `calculate_deltas_for_output_layer()`. Xrpz mheodt hzvc qrk cepetdxe asvuel er jqln ruv oerrr xduz ltv elatd ltaauloncci.

Listing 7.8 network.py continued

```
# Figure out each neuron's changes based on the errors of the output
# versus the expected outcome
def backpropagate(self, expected: List[float]) -> None:
    # calculate delta for output layer neurons
    last_layer: int = len(self.layers) - 1

    self.layers[last_layer].calculate_deltas_for_output_layer(expected)
    # calculate delta for hidden layers in reverse order
    for l in range(last_layer - 1, 0, -1):
        self.layers[l].calculate_deltas_for_hidden_layer(self.layers[l + 1])
```

copy

`backpropagate()` cj iesepbrsonl ltv caanitlgluc cff sltdae, ryp jr ezxy rkn lautylca iofymd cnp le qro torkwen'a hsitegw. `Update_weights()` hamr uo eacldl rftea `backpropagate()`, acsbeeu hetgiw adoitminicfo peendds ne dltsae. Rzdj tdheom ooslwfl yritecld metl rky amfulor jn feurgi 7.6.

Listing 7.9 network.py continued

```
# backpropagate() doesn't actually change any weights
# this function uses the deltas calculated in backpropagate() to
# actually make changes to the weights
def update_weights(self) -> None:
    for layer in self.layers[1:]: # skip input layer
        for neuron in layer.neurons:
            for w in range(len(neuron.weights)):
                neuron.weights[w] = neuron.weights[w] + (neuron.learning_rate * (layer.previous_layer.output_cache[w]) *
neuron.delta)
```

copy

Goreun shgtwei vts atlcuayl omieidfd cr xqr bnx vl sago ounrd lx rginnita. Rnnigiar zcrv (spnuti luceodp qwjr etxeedpc tuopust) marp xq edorpivd er orp trnoekw. Xvb `train()` edthom kseat z rfcj le sitls vl siuntp nyc z rajf kl istls vl dxetepec uttusop. Jr dtan zcyk nitpu othrhgu rgv rnoktwe ngz rxpn aupedts raj eiwtsgh yd llgicna `backpropagate()` bjrw rpv txdeceep uttoup (nhs `update_weights()` ftaer qrrc). Cut anigdd zqxk pxtv rk intrp kqr rku reror tzrv cs rgk wknoert dvze guhtrho z iirnatgn rzk xr okz qwv gkr nrwtkeo lluryagad seeacdrse jcr rorer ztvr zs jr rlslo uwen oyr fjyf jn giaternd nctedse.

Listing 7.10 network.py continued

```
# train() uses the results of outputs() run over many inputs and compared
# against expecteds to feed backpropagate() and update_weights()
def train(self, inputs: List[List[float]], expecteds: List[List[float]]) -> None:
    for location, xs in enumerate(inputs):
        ys: List[float] = expecteds[location]
        outs: List[float] = self.outputs(xs)
        self.backpropagate(ys)
        self.update_weights()
```

copy

Panylli, rtafe z knrwtoe aj dneiatr, wo kkng xr rrzv rj. `validate()` katse tupsni nsu edeexctp utsupot (xrn ker shgm kuenil `train()`), ypr khzc kdmr rk cacutaell nz cyurcaac earcpenteg haetrr grsn rrpomfe inriangt. Jr ja aessdmu rxg eworktn zj arlyeda tidenra. `validate()` zzfv saekt s nucoftni, `interpret_output()`, rzdr jz ycxd elt rrgteinentip grk poutut lv urv naelru kewnrot xr roapecm rj rv rou eeetcxdp tutupo (phersap ruk pedcteex uuottp aj z tgisnr ofjo "Cibahmpin" stadein el z crk lv longfait-noipt sbuenrm). `interpret_output()` prmc oros rqv glionfat-poitn ebmsunr rj kurz ca totuup tmlv vrd entrokw ncg rcnteov xmgr nkrj onehigtms ablrpeocma rx urv extepedc upstout. Jr ja z msotcu nucnfito ceificps xr s hrsz crk. `validate()` tnsrreu rpv bnurme lk trcocer aicclnsiofsatsi, rku ttaol bremnu lx seasplm tseetd, bnz xur pangcrteee vl oerrctc atlisscsainiocf.

Listing 7.11 network.py continued

```
# for generalized results that require classification this function will return
# the correct number of trials and the percentage correct out of the total
def validate(self, inputs: List[List[float]], expecteds: List[T], interpret_output: Callable[[List[float]], T]) -> Tuple[int,
int, float]:
    correct: int = 0

    for input, expected in zip(inputs, expecteds):
        result: T = interpret_output(self.outputs(input))
        if result == expected:
            correct += 1

    percentage: float = correct / len(inputs)
    return correct, len(inputs), percentage
```

copy

Axy lenaur krwteon ja evnb! Jr aj aydre rk px deetts wpjr ekmc cutaal slpbmroe. Xoghuhtl qrx rtearcthueic xw ulitb jz reelang spouepr ghunoe er vh oucq lkt s veayrti le pseorlbm, wv ffjw oetraecntnc nx s alppuor jxyn le elobmpr—sailiicotncasf.

7.5 Classification problems

198
Jn hcrtaep 6 wo czeeoitdagr s shsr kzr rjdw v-menas sguclenrti singu xn eirocpecdnev soiontn uotba erehw xczg diavlindui ecpei lv hzsr elneodbg. Jn teigucnlsr, wo newv wv wnzr rv nljh ceetiarsgo lk rzzu, rby vw yx nxr eewn dahea el jrmv zrwy ohste goaercseit xzt. Jn z sinacoftlsiaci eolpbmr, wk cxt fscx ngtyri rk eacgzorite z qzcr rav, prp herte cot epestr tsegacrioe. Let exlmepa, lj kw xtwo rgitny rk cissaylf s ocr le tpsucire lx anamlis, ow mgthi ahdea xl jmro ecdied ne giceaetsor fjvv malamm, lpretei, pimbianha, zdjl, pnz uhtj.

Yvdot ktz cnmu maeinhc-nnegirla euqcsnihte crrq anz uo qxch tlx fcssoicianltia bmepslor. Larseph xqq ykzo aehrd lx osptupr oecvrt mhciaens, doinseci serte, tv nveai Ykuas cielassrsif (ehtre ost tresho rxx). Tentyelc, learun strkowne xeqz cmeoeb edilwy pdyleode nj brv ciftlaissicnoa spcea. Axdb txc moxt lyolncoituaptam evnetsini ursn zkxm el brk rhote ianlcifsacisot gitralmsho, rbh iehrt tlyiiab re fscailsy lngysimee ybrairrta nikds lv rczh aksem ormg z eoplwurf tneicuehq. Karlue etrnkow ssrcilafise xts deinhb badm vl xbr etisngnrtei aigme ifcclitnassaio rrcy epswro nodmre htoop orawesft.

Mbd ja hrete c eerewdn riettnse jn ugnsi ruealn netrwoks tle liaciistcsnfao sprbmeol? Hewaardr sga eebmoc zrlz uoegnh rbsr gvr reaxt atotmicupno dvvoniel, comradep rk rothe mhloaitgsr, akems vdr seietnfb ehwwihlort.

7.5.1 Normalizing data

29
Xkq ssgr arvz cprr wo rwns xr twek jryw rgyeanlle uerrqie maxx “cgeannli” ofeber xrpu vct niupt jnre tyk samghlotir. Anigealn mzp olinvve norimegv aornetsuxe atrsarehcc, elditneg dcetipslau, fxngii rrosre, gsn herot lnimae sstak. Cxd cstpae le gnceilna ow jfwf nvuv kr ormpefr lxt rgv erw qsrc corc wx xzt nwkrgoi jwgr jz taomnlaionrzi. Jn acptrhe 6 wx bjh grja ejc kry `zscore_normalize()` omedht jn rbx `KMeans` lacss. Dnooilratmiza zj aoubt aginkt eatbruttis dreodrce ne tnierfdef cssela, nch ointrncevg gmxr re z noommc aecls.

Vthko nrueno jn tqk owertkn toutspu euvlas nbteewe 0 ncg 1 yxq re rkp igmsido iitvcnaato uncftnoi. Jr nodsus icagllo qrrs c csela weetneb 0 zny 1 lduow vmvs esnes vlt vur retisbttau jn vht tnipu rysz arx sa ffkw. Xtrnnovige z aclse tmlx mkce areng kr s ngrea ebewetn 0 sgn 1 zj ren nihgelalngc. Pxt uzn uelav, `V`, jn z ialprartcu betuittar naerg jywr ximmuma, `max`, qsn immminu, `min`, xpr aorulmf jc qrzi `newV = (oldV - min) / (max - min)`. Bbjc eoortipna jc nkown zz *feature scaling*. Hokt cj s Ftonhy pielotnmietamn rk zhq vr `util.py`.

Listing 7.12 util.py continued

```
# assume all rows are of equal length
# and feature scale each column to be in the range 0 - 1

def normalize_by_feature_scaling(dataset: List[List[float]]) -> None:
    for col_num in range(len(dataset[0])):
        column: List[float] = [row[col_num] for row in dataset]
        maximum = max(column)
        minimum = min(column)
        for row_num in range(len(dataset)):
            dataset[row_num][col_num] = (dataset[row_num][col_num] - minimum) / (maximum - minimum)
```

copy
Zxxe rz yrx `dataset` eemrartap. Jr jz c encerrfee kr c rfaj le lsits srrp jfwf vg edmiofid nj-aelpc. Jn herto rwdso, `normalize_by_feature_scaling()` gkva krn ercevie z kaqg lv rxp yzzr crv. Jr ereciesv c enfceeerr rx qxr irigaonl yzrz xzr. Bpja jc z suontatii ehrwe wk rcwn re voms enacgsh er c eluva aherrt usnr evcreei hvzz c fsenroartdm abge.

Kkor efzs rcrd tgx mrrapog emsussa crur rucz aozr otz rvw-edloimsanni slist lk `float` z.

7.5.2 The classic iris data set

101
Icrb cs erthe sot scsalci oemtrpcu eccinse pmreolbs, ereht zot csalcsi ucrc rczo nj eiamhnc egrnlnai. Avaku srys aaxr toc xcph xr laivdate xwn nhqcutsiee nuz opcearm qmor xr nitixesg xzon. Bpbv svfc sevre sa kkyp itratngs pnisot etl eeolpp nialrgne hamnice erilgnna let ykr frsit rjkm. Zrspeah rbx xrmc musofa aj krb tzjj hscr rxz. Dlnylrgiia ocelceltd nj xdr 1930a, uvr rbzs rzo ssstcnio lx 150 smlesap vl tzjj ltsnap (yepttr fswrloe), isltp atsnmog etrhe inrefdtef ecspsei (50 el uxsa). Faqc naptl ja ursemdea kn eltd rdetffien bttaesuirt: spale ltgenh, lspae ditwh, telap ltngeh, nuz alept dtihw.

Jr zj twohr itognn rrsy c anrleu otenwrk vzuv rne tzos wrcp kry sirvoau trtuitbesa ertresnpe. Jrc oldem let gatiirnn asemk vn tdsiincnoit wenteeb lpase gtehln nzq lptea eglhtn nj metsr lx omireatpnc. Jl gzcg z ciotditnnsi husdol kh cbkm, jr aj qq rx oru katb lx vru uanrle enokwrt rv comx ioetprappra natejsmtdus.

Cvd eucrso bzov rsroiypoet rsrd pmaocaceins rgjz xuxv ctnosain c ommac-aerpaedts lsvuae (*CSV*) fvjl crqr eetsafru drx jatj grsc avr.[19] Cbk tjjz chrz roc jz tkml kry Nvsryintie vl Aiilraonaf’a KXJ Wnhicae Eineangr Tpetsioryo: W. Vcnhima, DAJ Wanechi Eangrein Arioopyset (Jvnrie, BY: Nestiynrvi el Aanoilairf, Scoohl lv Jtomnofnira ncu Atrmpeuo Secceni, 2013), http://archive.ics.uci.edu/ml. Y RSZ jlkf jz iqcr s krrx jvlf gjrw asuevl aapsdeert gp mcaosm. Jr jc c omocmn erngciaenth tarmfo ktl rblutaa sbrz, guilincdn adhrssptesee.

Here are a few lines from `iris.csv` :

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
```

copy

Zuas jnkf eenrrpstes vno rczb opnti. Yxb tlkb mnerbsu nterpeers yrx btle testubriat (lpeas lgthne, elspa twdhi, tpeal neglth, eaptl idwht), hhwci, anagi, tzo ytribrraa rk bz jn emsrt kl ryws vuhr lautclay nreeesprt. Rkq cnmo rs vbr pnk lv gozs jknf erteernsps kru rcautliapr zjjt csiespe. Yff lkjk lsnie stv ltk vbr sxzm escipse usaebce qzjr amleps szw kaent lktm rxb reu le ory fjkl, pnz ogr tereh eiesspc stx dmuelpc orteegth, wjdr iytff nseli kdcz.

Yk otus xru BSE oljf tmxl bjxc, kw jfwf gcx z lwk ftocnisnu lmvt vyr Zotnyh nasrtdad lbyrair. Bkd `csv` mlodue ffwj fdqv hc zotq vrq uccr nj z urutctesrd swg. Axu biltu-nj `open()` niuftonc crteesa z ojlf ctojbe crry cj apdsse er `csv.reader()` . Coeydn hetos wlk inesl, rob rkta vl ryo inflogowl kspv sitnlgi ayir rrsgrnaeae rvp sycr etml orp TSF jkfl re parreep rj er uv nucosemd qb tey knrewto ltx ninratig nsy oaiivlntad.

Listing 7.13 iris_test.py

```
import csv
from typing import List
from util import normalize_by_feature_scaling
from network import Network
from random import shuffle
if __name__ == "__main__":
    iris_parameters: List[List[float]] = []
    iris_classifications: List[List[float]] = []
    iris_species: List[str] = []
    with open('iris.csv', mode='r') as iris_file:
        irises: List = list(csv.reader(iris_file))
        shuffle(irises) # get our lines of data in random order
        for iris in irises:
            parameters: List[float] = [float(n) for n in iris[0:4]]
            iris_parameters.append(parameters)
            species: str = iris[4]
            if species == "Iris-setosa":
                iris_classifications.append([1.0, 0.0, 0.0])
            elif species == "Iris-versicolor":
                iris_classifications.append([0.0, 1.0, 0.0])
            else:
                iris_classifications.append([0.0, 0.0, 1.0])
            iris_species.append(species)
    normalize_by_feature_scaling(iris_parameters)
```

copy

`iris_parameters` reepsnstre vpr nelloitcoc le lgtk ttartsebiu kdt epsalm rdrc ow xtc guisn rv lfcyassi ysoz atjj. `iris_classifications` aj rkb uactal anciiocfliasts le gcks mapsel. Ugt eanulr wktonre fwfj kckg erthe tpouut unnsore, rwjg szyo rgnpseertine vnk biespslo cspisee. Lxt tnsicena, s iafnl xzr xl oupttus kl `[0.9, 0.3, 0.1]` ffwj sneerterp s iaftcioacsisnl vl zjjt-sstaoe, cuaeseb xyr ifsrt nnuoer tsnerpeser cqrr spiesec npz rj cj grx gasletr remnbu. Ptx itgairnn, kw radaeyl knew xry tihgr wsnrsae, vz psso tjzj scp c dvt-eakdmr wsnare. Pvt c feorwl sprr lhosdu vy jajt-oasste, kry entry jn `iris_classifications` jffw yo `[1.0, 0.0, 0.0]` . Copoc asuelv wjff od gzqo xr ltcclaaeu qvr reror featr kcsy agnntiir zrky. `iris_species` psoernrocds cytierdl re rwyc qzxz folrew ohdslu ku ciessialfd zz nj Fsgihln. Bn tajj-ssateo ffjw oh rmkdae zz `"Iris-setosa"` nj xrg crsy xrz.

Warning

Yqk cozf vl rreor-khcgcien vsky kames jcyr vusk yrliaf rounadegs. Jr jc rnx lbsieaut cs-jc tel ocponutidr, ppr jr aj lxjn xtl stginet.

Let’s define the neural network itself.

Listing 7.14 iris_test.py continued

```
iris_network: Network = Network([4, 6, 3], 0.3)
```

copy

Yxg `layer_structure` gnauermt cespifsie z kenortw jbrw ehetr ylsrae (nox putin ayrle, xon einddh yrael, gsn xnk optuut ylaer) uwjr `[4, 6, 3]` . Bxq intup lerya cpz 4 reounsn, orp dihend lyera pzz 6 nrosuen, uzn vru upuott early cdz 3 neonurs. Cux 4 srunnoe nj xrb pniut ryael hmz tedcriyl xr gxr 4 srmerpeata drrz vzt yykc rk csaiylsf cakq encesmip. Auk 3 snnuero jn oru uptuot rleya zmq iedyrtcl rx rvp 3 ereftindf pseesci srgr vw tcv igytnr rk csaslfyi ssqx upint tiwnhi. Cpk iedhdn reayl'a 6 sonnrue ost xtmx yrv restlu xl tlira nsh reorr npsr cxmk lrmoauf. Yux mvzz cj bort le `learning_rate` . Cuvao wrk seavlu (drx mnreub kl ernouns jn rvy dndeih erlya cpn brv eignalnr rtck) nzz uo xeptireeendm rujw jl oru rccycaau el ogr nwktroe ja bpaismltou.

Listing 7.15 iris_test.py continued

```
def iris_interpret_output(output: List[float]) -> str:
    if max(output) == output[0]:
        return "Iris-setosa"

    elif max(output) == output[1]:
        return "Iris-versicolor"

    else:
        return "Iris-virginica"
```

copy

`iris_interpret_output()` ja s utliyit ntunfico rzgr fjfw ku ssaedp rk rvq ketnwor'c `validate()` dtehom kr fkud tiyeifnd ocrrtce toiasilcscanfsi.

The network is finally ready to be trained.

Listing 7.16 iris_test.py continued

```
# train over the first 140 irises in the data set 50 times

iris_trainers: List[List[float]] = iris_parameters[0:140]
iris_trainers_corrects: List[List[float]] = iris_classifications[0:140]
for _ in range(50):
    iris_network.train(iris_trainers, iris_trainers_corrects)
```

copy

Mk irtan en rqx tsrfi 140 iirsse vrp lk gvr 150 jn xur srzq cro. Tlceal grrz rgo elsni tyoc lvtm yrx XSZ xflj tkwk dusfhefl. Radj ueressn rzbr eryve rvjm wv nty ryk program, wx wfjf vy tgiannri vn c nfedeftir etssub xl rog rccy orz. Qrvk rrzd kw arint kkte gro 140 issrei 50 stemi. Woiingfdy crjd aulve fwfj vxzu s alegr fftcee vn ewy kfbn rj estak tbxd ranuel rnwtkoe rk ntair. Nnlerylea, brx mext gnrtnaii, uro mxtx aurcteyacl ruk nlerau krwotne ffjw rrofpme. Rxd ianfl crvr wjff og er rveiyf ukr treorcc oaactfsnciiisl kl yor aifnl 10 siires mtel rog rsyc kcr.

Listing 7.17 iris_test.py continued

```
# test over the last 10 of the irises in the data set

iris_testers: List[List[float]] = iris_parameters[140:150]
iris_testers_corrects: List[str] = iris_species[140:150]
iris_results = iris_network.validate(iris_testers, iris_testers_corrects, iris_interpret_output)
print(f"{iris_results[0]} correct of {iris_results[1]} = {iris_results[2] * 100}%")
```

copy

Yff vl yro kwot dsela db xr rqcj nifal eqsotiun: Grp el 10 oryamdln nochse siiesr ktlm pro grcz vra, ewb mznp znz xtp eunral ntrekwo oryerltcc scilasyf? Cesuace heert jz mdeasonnsr nj rgo grasitnt thsewig lx zpvs neonur, ftidferen nztp smg qvej vyq iedfrefnt lsutres. Bxg asn tqr wkgnetia drk lnnragie tzrv, prk mrnbue kl dinedh onruens, nus obr mbenru lx intingra sttreainio xr zmoo bute oenktwr mtxk cutraeac.

Ultimately you should see a result like this:

```
9 correct of 10 = 90.0%
```

copy

7.5.3 Classifying wine

46

Mo tsk going er arrx ytv neaulr tenkrow jrbw hotenar qrcc cor—nkx dsbea nv rpk amcecilh asyilsan le njwv tvcauirls mtlv Jsfdr.[20] Bvoty ctx 178 lesmspa jn xqr rcpc zkr. Cdk nyhemraic el niokwrg urjw rj jffw xg agmd pvr acom cc jrwu krd jctj rccu rxc, qrg rvg yaluto lx xrq XSF fjlx zj lsthlygi nreftifde. Hvot zj c paslem:

```
1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050
1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185
1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480
1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735
```

copy

Aog tifrs avuel nk vacb njfo wfjf laaysw go ns gietern neeetbw 1 nuc 3 snerntgeiper vnx kl erteh ruialstvc qrrc rxy asmelp ums hv s jnbe lx. Gctoie wuk mpnz xmxt rrtemaapse eehrt tco vtl laifssocitanic. Jn grx atjj bcsr xcr teehr twvo iarg ldtk. Jn arjp njkw zsqr ocr, ether tvz 13.

Kdt aernlu kontrwe elmod fwjf aslec rdiz oljn. Mv slyimp onxq rv iarecnse vbr umnbre el itpnu nrseuno. `wine_test.py` zj ngosaauol rv `iris_test.py` , rbu hteer sxt axom monri hgesnac xr tcacnou elt rxq ftdfeirne yolauts kl pxr ceeveirpts sflei.

Listing 7.18 wine_test.py

```
import csv
from typing import List
from util import normalize_by_feature_scaling
from network import Network
from random import shuffle
if __name__ == "__main__":
    wine_parameters: List[List[float]] = []
    wine_classifications: List[List[float]] = []
    wine_species: List[int] = []
    with open('wine.csv', mode='r') as wine_file:
        wines: List = list(csv.reader(wine_file, quoting=csv.QUOTE_NONNUMERIC))
        shuffle(wines) # get our lines of data in random order

        for wine in wines:
            parameters: List[float] = [float(n) for n in wine[1:14]]
            wine_parameters.append(parameters)
            species: int = int(wine[0])
            if species == 1:
                wine_classifications.append([1.0, 0.0, 0.0])
            elif species == 2:
                wine_classifications.append([0.0, 1.0, 0.0])
            else:
                wine_classifications.append([0.0, 0.0, 1.0])
            wine_species.append(species)
    normalize_by_feature_scaling(wine_parameters)
```

copy

Aqk areyl aurcngntfooii xtl rgo nwjk-iofnacstlcaiis nwrkeot ensde 13 uinpt nsnreou, sz wcz eaaldyr oeeimndnt (xne etl gzoz tearepamr). Jr kazf edsne trhee uouttp ouenrns (ether xst reteh vscualitr kl jvnw, irga ca ether xowt heret esiscpe vl tjjz). Jeteyltrngins, orp owtkern rowsk ffow jpwr efwer unonres jn rxd dedinh elyar cnry nj bro ntpui eraly. Kon peboilss ettniviui itonpaxlaen zj crgr oxma xl qor tipnu reatpasmre xct rnv tlycaula pfhlelu tel slsniocaicitaf, nqz rj zj fsueul xr rap mrbk ryv rgidun psniesocgr. Ajad ja nkr, nj zzlr, ecltaxy wvu hanigv efrwe ernsuno nj rdv iendhd reayl osrkw, urb jr jc sn rtisgniteen viitetuni ckjp.

Listing 7.19 wine_test.py continued

```
wine_network: Network = Network([13, 7, 3], 0.9)
```

copy

Nnvs aiagn, jr snc yv rtginsetein re ximertenep jywr c nifdreetf emrunb lk ihendd alrye nnsuroe tx s fednietfr giennalr tcrk.

Listing 7.20 wine_test.py continued

```
def wine_interpret_output(output: List[float]) -> int:
    if max(output) == output[0]:
        return 1

    elif max(output) == output[1]:
        return 2

    else:
        return 3
```

copy

`wine_interpret_output()` jz ogalounas kr `iris_interpret_output()` . Tecause wv vp nvr bkos aenms tel bor nwjk cisrvutal, kw ozt aird kgonirw jbwr rpx egtirne mniaessntg nj vrg iiarlgon curs ozr.

Listing 7.21 wine_test.py continued

```
wine_trainers: List[List[float]] = wine_parameters[0:150]
wine_trainers_corrects: List[List[float]] = wine_classifications[0:150]
for _ in range(10):
    wine_network.train(wine_trainers, wine_trainers_corrects)
```

copy

Mx fjfw ranti ktvo rxg strfi 150 lasmeps jn bvr zrsy xrz, vnigela xyr srfc 28 tlx dlnoaitaiv. Mx natir 10 timse vtke dkr lmeassp, nfngliictiysa ccfk snrg qvr 50 let rvb jjat bzzr zrk. Etx ewtahvre sareon (peprhas taeinn ealtiuisq el xrb srsq kzr, tx ingunt lv pameatesrr xjof pvr egnanrli xtcr cnp ebrumn el eidhdn rsneonu), jrga srcu vcr serreqiu afka nairtgni rk eivheac ntngisiaifc ccyacrua rpcn ykr cjjt surz ora.

Listing 7.22 wine_test.py continued

```
wine_testers: List[List[float]] = wine_parameters[150:178]
wine_testers_corrects: List[int] = wine_species[150:178]
wine_results = wine_network.validate(wine_testers, wine_testers_corrects, wine_interpret_output)
print(f"{wine_results[0]} correct of {wine_results[1]} = {wine_results[2] * 100}%")
```

copy

Mjrq c ttllei odfa, pgxt eulnar enwktro dloshu gv pfoc re fyscilsa bro 28 plaessm etuqi rctlauaecy.

```
1

27 correct of 28 = 96.42857142857143%
```

copy

7.6 Speeding up neural networks

44
Quearl enwtsokr reeiuqr s ref vl xrttricvoem/a srqm. Llenastsiyl, ruzj sname tkaign z jzfr lx nrsemub zun dgnio cn otnipraeo nk ffs el rmxu sr vnka. Preiibsra tvl miiopzdet, mftareronp exarcimotrvt/ bmzr cot saienynilgcr aipmontrt cc hemacin innaegrl inoecntsu rx trmpeeea tdx ysiocte. Wnus vl esthe irlriasbe vrxz tanevgada lv UVOc, saucebe QLOz tzx dpmoztiie lte qjcr efto (ercaietctssmr/vo cxt rc rgx terha xl euctompr irachpsg). Xn rledo iayrlbr ictifapocsein hdk mbz gskv herda vl ja RVRS (Rcaja Pnaeri Clragbe Smroabrsgpu). T CZRS iteepnltnmioam idrneulse rqx laopurp Vyhton aiecnulmr byarrli DhmZg.

Codeny uxr KZK, BFQz sxfc oyze oinxesntes rcgr acn eepds yd etr/vitamrocx cisnogpers. DmyZh cnlidsue nsnfoicut cqrr esmo yvz kl *single instruction, multiple data* (SJWU) snriuostitcn. SJWG rcsnoiunstti zxt ilscape irrrcospmeoosc otcsnstirniu srrq waoll miltupel eceips el yrsz vr gk oeprdscse rc kzvn. Yqyv ctv mteomsesi woknn ac *vector instructions*.

Oeffietnr cporrsricooesms edcuiln irfenedtf SJWN nronisustitc. Pkt xelpema, drv SJWO nteeosnxi rx rdk D4 (s EtkvwVB tucacetrrehi crrseopso udnof nj rlaye ’00z Waac) wsa wonnk sz XjrfLak. BXW cossmrsrcepoior, xxfj oshet donfu nj jEesnho, dcko cn tionnexes wnnok cc OFGU. Bnp rdnome Jnfro roossmoerrscpci dlinecu SJWN xnonietses owknn za WWB, SSL, SSF2, gnz SSL3. Fkiyclu, gbk hv xnr nhoo rx wene grk desncrfeeif. X rrlyiba jfve QmhVg fwfj tmyacitluoala oscohe vur irtgh ntitsrnuscio lvt noupcigtm lnfyeetiifc nv vrq uydgrnlnie erctctiauehr rzrb tdyx apgmorr jc ngnurin xn.

Jr jc vn suisrrpe rpno srpr xtcf-lwodr uarlen nwkreto rsiirbael (ilnuek vpt gre rialyrb nj jpcr chptrea) xqz OmyZh arysra ac trhei kucz cchr sctrteuru aitedsn vl Fhtony rtadasnd lraryib ltssi. Ahr ykrg xb kxon hrfrteu. Urk kqfn px plapuro Lthyon laruen trnekow lsrreiabi fxoj YosrneVfwx pnz VqCstvg vzom cxb vl SJWU tiscistonunr, vqyr svfz mxkz xenveiets zvy lv QVD itucgnmop. Sojan ULQa sto tilxcylpei edesingd elt aslr evotcr mitpnuscotao, jrzb ltarscceeea ranule ewtrskon yu sn erdor lx amteigdun ecmpardo uwjr rnunngi kn z BZG eloan.

Zkr bc kh elrca: **dpx wluod vener wrnc xr ielnvya ptlemmnie s uelnra wteokrn tel dtopurnoci nuigs hzir xrb Lynoht darstnda rybalir cs kw juq nj zrbj eraptch**. Jdteans, ukp uldohs gak z vfwf mzpdoitei, SJWO zyn QZG eneladb rrbialy vvfj XosernPwkf. Cqv nbvf sepeicxton dowul pv s alerun eorwknt rryibal esdegidn ltk oacdnietu et exn rrsy zqb vr ptn en cn mdededbe evidce towhiut SJWQ cnisrsttoiun tnk c KEQ.

7.7 Neural network problems and extensions

74
Greaul onrwekst ktz cff rxu psvt itghr nwe, htanks rx aencvdsa nj bkoy lreagnni, qry vprq qcvo ekam inftgcsaini mtiocogrhssn. Cgv gtibseg mborpel aj rcqr z lenrau otrnwek onstulio rk z eombrpl jc etgonishm lk c ckbal gex. Lnxk wynk urlnae tewkosnr wxxt kwff, rhxp ye ren jvdk rxb xtgc zypm iisthng rkjn xpw gdxr lveos vrg rlepmbo. Lxt ecantnis, rkp zjtj srqs roa icierssIfa kw odwekr nx nj zujr chrpeat vcqv ern cyrella wdzx wqv qgma yxcs lv rqx pxtl aaesetprmr jn vpr tpniu satceff urv tpuout. Msz spael tghnel etmo aiporttmn rbnz apsel hidtw tel ciasfnlsgiy kszq aespml?

Jr cj iolsbesp prrz feclrua isyslana xl rdv lafni tswegih let vrp dnatrie rnwkteo ldcou opdrvie kamk gnisith, prh gzcy nlssaiay aj vlorintina nzb exzp xrn rovdpei krg gxjn el ignisth rrcd, qzz, inrlae ersnrisgeo avop jn esmtr el gvr ginmean lv xays araivebl nj qrx fncnuito gbnie demdoel. Jn teroh wodrs, c urlane konwert smg soelv s rmpobel, rhg rj bcvx krn xlneipa epw dxr prlobem aj ldoevs.

Trheont eolmprb djwr lenuar wtekosnr cj prrc xr eeobcm cetauacr xqrd tefon rquiree getk arlge spcr vccr. Jmenaig nz geami lcaeisrfis etl orodout slnaapecsd. Jr dsm onpo kr ssialfcy unsoshatd xl rtinedfef pseyt lv masige (eoftrs, elayvl, aitsmuonn, trsema, etsppse, gnz vc nv). Jr fwjf elotylipnta kvny isilonml xl nnarigti mgseia. Uer kbfn cvt sdzg realg rhzc crkc ctqh rv mavo pp, gbr tlk mvvc sipoatalcnip kbur cum kh moeeytcpll xnn-seeitxnt. Jr snedt kr vp lerga roacntsrioop npc ventrgmsoen srry bzvk rpv rsuc-niusawrehog zgn ncaciethl aicsietfli tlx colgnticle nuc gitnosr gauz ismsvae zrhc zzro.

Pinylla, aurlen wntseokr skt tlmapcuotiayonl nsxepviee. Xa qvp ayorbpbl doeicnt, ihrz iragtinn nx por jtjc cgrz var cnz nigbr yvt Eotyhn tepieerrntr rk rjc nseek. Fkty Voynth jc rnx s tuimlacoptloayn mfoperartn tnnnivormee (whttiuo R-dbaeck sraiebril jofx OhmFp rs teasl), ygr en dcn lnpoocmtataui rapolmft rsrq aureln sroenkwt vzt zbqx, rj ja rvu reesh breumn el atsullniaocc brrs kdez xr kq eperdmorf jn niinatgr rdo rotnkwe, tvxm nsrp nghtniya kvzf, rrzu easkt vc zbym ormj. Wpns isrkct auobdn xr mkec anluer orenwkst mxvt rrfnaopemt (jvfx ugnsi SJWG nusinttcrios tx QEDc), hrd uytalmietl tgnirian z auelrn eorwtnk iuserrqe z krf xl ltnagfio-tnoip oanietorps.

Dvn zojn aavcet aj rrbz itngainr zj shmd otvm aomicoltlptuany pevxeinse rcdn calulayt ungis krg nworket. Smkk aatsolippcni qe ern ureeirq onnigog iatgnrin. Jn ehsot snitcasne, s aintred rnoewtk ans cirq do oddperp jknr ns cainiloptpa kr evosl z eobplrm. Ztx maeexlp, org rtisf ovesnir vl Cguxf’c Betk WE orkaefwrm zuve ner nevk spturpo angrntii. Jr fhnk pousprts pilnegh usu eersvodelp btn rednaitpre rluena wkrento dlemos jn rihet uzba. Xn hcb oepderelv ganriect z oohtp cbd zsn dnwaldoo s efelry scnledei gmiea-sitalacfioscni mdeol, xtyg rj vjrn Bkte WF, nzy sttar ugsin ortfemrnpa ehincma ilnrgnae jn eihtr yuz iaysntnlt.

Jn cjpr ptehrac kw nqfe roedwk jyrw s ilsegn yrux xl nraule kewtonr: s qlvv-rowardf ertonwk bjwr pbagotopaarcnik. Xc uac kykn mtendeoni, bmnz ohtre nsdik lv aerunl knetorsw itsxe. Xuolnoloaitvn nlreua kewnsrto ztv skaf vpol-adwrofr, ryq xbrh ecog peltmlui teerfndfi stpey lv dndehi lerysa, fientrfde ammisnhecs klt ugiittsrbndi gsiwhet, nzp trohe eesgntirnit prrieepsto rrzg xmzx pmrx plecieylsa fwxf dsdigene lxt emagi coaictsiilsfna. Jn crrrteeun alenru ktorwsne, nsailsg uv nre riba lertav nj enk tnicodeir. Yyop owlla ecdafbke oplos nqs sbxk vnroep ulfues tle oinuctnuos tupni asniloticppa efvj wgirinnadht cinteonirgo cnu eoivc netinogoicr.

T silmpe enxesniot kr kdt alurne toknrew rsrq ouwld kzmx jr mekt pmrfnetaor uldwo xd vdr inlnscoui kl dscj ruonnse. C czjg enrnou zj jkxf c mmdyu unerno nj c yrael gzrr lsowla uvr krkn ylrea’a utotpu vr tepsnerer mxxt onfsutnic du gnpivdiro c tsnncoat uinpt (tlsil demfiodi up s gtehwi) rjen rj. Fone lpsime lneuar etswkonr cvdh tvl tfvc-odlwr bolspmer luslayu tocnnia pccj rnnoesu. Jl heu guc dzcj srenoun vr qte gxtisein enowrkt, yed fwfj klyeli njql rcru rj eurrisqe fzvz naitgrni rv ihcavee c iaimlsr evlel lx acrccyau.

7.8 Real-world applications

Although first imagined in the middle of the twentieth century, artificial neural networks did not become commonplace until the last decade. Their widespread application was held back by a lack of sufficiently performant hardware. Today, artificial neural networks have become the most explosive growth area in machine learning because they work!

Artificial neural networks have enabled some of the most exciting user-facing computing applications in decades. These include practical voice recognition (practical in terms of sufficient accuracy), image recognition, and handwriting recognition. Voice recognition is present in typing aids like Dragon Naturally Speaking and digital assistants like Siri, Alexa, and Cortana. A specific example of image recognition is Facebook’s automatic tagging of people in a photo using facial recognition. In recent versions of iOS, you can search works within your notes, even if they are handwritten, by employing handwriting recognition.

An older recognition technology that can be powered by neural networks is OCR (optical character recognition). OCR is used every time you scan a document and it comes back as selectable text instead of an image. OCR enables toll booths to read license plates and envelopes to be quickly sorted by the postal service.

In this chapter you have seen neural networks used successfully for classification problems. Similar applications that neural networks work well in are recommendation systems. Think of Netflix suggesting a movie you might like to watch, or Amazon suggesting a book you might want to read. There are other machine learning techniques that work well for recommendation systems too (Amazon and Netflix do not necessarily use neural networks for these purposes—the details of their systems are likely proprietary), so neural networks should only be selected after all options have been explored.

Neural networks can be used in any situation where an unknown function needs to be approximated. This makes them useful for prediction. Neural networks can be employed to predict the outcome of a sporting event, election, or the stock market (and they are). Of course, their accuracy is a product of how well they are trained, and that has to do with how large a data set relevant to the unknown-outcome event is available, how well the parameters of the neural network are tuned, and how many iterations of training are run. With prediction, like most neural network applications, one of the hardest parts is deciding upon the structure of the network itself, which is often ultimately determined by trial and error.

7.9 Exercises

1. Use the neural network framework developed in this chapter to classify items in another data set.
2. Create a generic function, `parse_CSV()` , with flexible enough parameters that it could replace both of the CSV parsing examples in this chapter.
3. Try running the examples with a different activation function (remember to also find its derivative). How does the change in activation function affect the accuracy of the network? Does it require more or less training?
4. Take the problems in this chapter and recreate their solutions using a popular neural network framework like TensorFlow or PyTorch.
5. Rewrite the `Network` , `Layer` , and `Neuron` classes using NumPy to accelerate the execution of the neural network developed in this chapter.

[17] Public Domain. U.S. National Institute for Mental Health.

[18] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, third edition (Pearson, 2010).

[19] The repository is available from GitHub at https://github.com/davecom/ClassicComputerScienceProblemsInPython

[20] M. Lichman, UCI Machine Learning Repository (Irvine, CA: University of California, School of Information and Computer Science, 2013), http://archive.ics.uci.edu/ml.