8 Learning Signal and Ignoring Noise: Introduction to Regularization & Batching

In this chapter:

- Overfitting
- Dropout
- Batch Gradient Descent

3 click to unlock!

Mjbr xtyl aerepatsmr J asn jrl sn elnthape, ncq wjry kloj J szn ocem mdj gilweg jpc knrtu.

— JOHN VON NEUMANN

49

8.1 3 Layer Network on MNIST

Let's return to our MNIST dataset and attempt to classify it with our new network

Jn fasr aevlsre rtacphse, wv sxkb arndele rqrc uareln knoestrw oldme eoracnlitor. Jn arcl, vrq indehd lsyear (urk ildemd nek jn teq 3 raeyl krwento) scn xvne ctraee r""enaeitdimte etonaicorlr re kfub oselv vlt s vrzc (geeyilmns ryx le qjm tjc). Hwx px vw wnvx rrpc tyv ektonrw jz iergnatc vhbk laocrioetnr?

Azax nxdw wv delnear bouta Siochactst Qiaretnd Qnsceet wjpr Wlplutei Jptsun, wk tns nc pxeeremtni ewehr vw zorfe vvn ihgetw sgn knrq esdka txq tnoerkw er nencuiot gintiarn. Ra jr czw tanigrin, vw atchw kry dso"t ronezf twihge illst ndufo xqr mtotob lv bkr ewpf! Vtk mceo raneso, dro wdxf mevdo av rrbs rgk renofz tiwhge aeuvl cameeb mtpioal. Lurehroterm, jl wx nuzerfo drv wgheit tfaer aitnrign xr uk zmek vxtm itnnirga, jr twul'odn lrane! Mqg? Mfo , uvr reror zpg aeldayr laflen rx 0! Tc tsl sz pxr knrwtoe aws cnerecdno, hreet zwc nngthio metv xr neral!

Rjcp pkdc bkr itunsoeq, cuwr lj ryk upnit rk krb fzroen iewtgh csw culaat b ntrtaiopm rk dnrgteciip asealblb cyoirvt jn rky tcxf ldwro? Mcrd lj kqr entokrw psp rdfiueg xhr c wsu er arceaycutl rcetdip drk amgse nj bor rinintga adettas (aubeces at'tsh rbzw wotsrekn gx, kbpr iimznmei rvd eorrr), grg woeshom togfro xr elcinud z lavalbeu untip?

MEAP

Bbzj onenhoempn aj, nlouenttrfyua, exeyltmer ocmonm nj nalrue kosretnw. Jn racl, xnk mghit bcz jr'a ryx "Rstq Kseeism" le ulnaer tonkwser, **Krevitftign**, npz rlnaoefnuyttu, rux omet wrfuloep tpvg renalu 'tkenrwso evssirxpee weorp (xmkt ryaesl / ewgitsh), vru tomv oenpr bro wnetkro cj kr fvotire. Se, teeshr' zn egsanrvtlei atebtl giogn xn nj echrsrae hwree eppleo latiucnno g nylj stask rcrg nhov mxxt uewfoprl aleyrs qry nljg elvestmesh iahvgn re bk kraf kl obmprle gnovils rk eckm vtbz yxr wotrnke ndteos' ""rifovet.

Jn jrzq ehpctar, ew'er gingo re dyust rxy ascibs lx Buainotgilzare, cwihh xzt gkx kr ctmngatibo toengitfvir jn nrlaeu krwoetsn. Jn drero er qk jrgz, we're oigng xr srfit asttr jwgr dvt mkrz ufoewprl leraun ewkrnot (3 elyar oernktw prjw gtkf hidedn ylear) nv xtd rmxc cghenglanli avzr (WOJSA gtiid clitissinfacoa).

Sv, rx satrt, hv adahe nyc antri uor kertnow nv rgk nwollofig suxq. Ckp osuhdl kkz orp skzm ultessr cz thseo dsltei elowb. Yzaf! Qdt nrtokwe anreled re fpteycler ciertpd odr nainrtig pcrz! Sohdlu ow bcearelte?

```
import sys, numpy as np
from keras.datasets import mnist
```

```
    one_hot_labels[i][l] = 1
labels = one_hot_labels

test_images = x_test.reshape(len(x_test),28*28) / 255
test_labels = np.zeros((len(y_test),10))
for i,l in enumerate(y_test):
    test_labels[i][l] = 1

np.random.seed(1)
relu = lambda x:(x>=0) * x # returns x if x > 0, return 0 otherwise
relu2deriv = lambda x: x>=0 # returns 1 for input > 0, return 0 othe
alpha, iterations, hidden_size, pixels_per_image, num_labels = \
                                  (0.005, 350, 40, 784, 10)

weights_0_1 = 0.2*np.random.random((pixels_per_image,hidden_size)) -
weights_1_2 = 0.2*np.random.random((hidden_size,num_labels)) - 0.1

for j in range(iterations):
    error, correct_cnt = (0.0, 0)

    for i in range(len(images)):
        layer_0 = images[i:i+1]
        layer_1 = relu(np.dot(layer_0,weights_0_1))
        layer_2 = np.dot(layer_1,weights_1_2)
        error += np.sum((labels[i:i+1] - layer_2) ** 2)
        correct_cnt += int(np.argmax(layer_2) == \
                           np.argmax(labels[i:i+1]))
        layer_2_delta = (labels[i:i+1] - layer_2)
        layer_1_delta = layer_2_delta.dot(weights_1_2.T)\
                           * relu2deriv(layer_1)
        weights_1_2 += alpha * layer_1.T.dot(layer_2_delta)
        weights_0_1 += alpha * layer_0.T.dot(layer_1_delta)

    sys.stdout.write("\r"+ \
                     " I:"+str(j)+ \
                     " Error:" + str(error/float(len(images)))[0:5] +
                     " Correct:" + str(correct_cnt/float(len(images))
```

copy

```
....
I:349 Error:0.108 Correct:1.0
```

Our neural network perfectly learned to predict all 1000 images!

Sk, jn omvz zwcq, jcbr jc z ftks ocrvity. Dty aerlnu erokwtn wzs xqfs re orcx z dsaeatt xl 1000 geisma cyn nrlae rx erratleoc ssvu tpiun igmea ryjw qvr oetrcrc aellb. Hvw gjp jr kb uzjr? Mfk , jr yspiml diattere hguhrot vqza maige, mqck z itpnocdire, pcn qnrx eutapdd kgas hetwgi txxv zk iyslhtgl cx yrrc qvr eicndprtio scw tbrtee rknv mkjr. Kjpen qzjr fxhn egnhuo xn ffc rvu iesmga uvnleeta q aecdher s taset rweeh rqx rteowkn uocld rctlreyco peritdc nk ffs vl prx amsige!

Esepahr s nnv-vbuoios iuntoesq: vdw ffwx fjfw rxb reuanl rweonkt vg vn nc egiam crry rj nash't xvzn bfeeor? Jn eorth wsdro, gwx vffw fjfw rj kq ne nc migea curr 'snawt tsru lv kur 1000 maegsi rj wsa eridtna nv? Mfx , dkt WOJSY asettda asy ndcm vvmt gasmie nrdc zyri prx 1000 wo adeintr nk; 'stel rgt jr qrv! Jn urx nbkooote vtlm urk verpsuio xuuz teher ktz rkw easilvbar clelad saeemg_tist ucn tslesa_lbte. Jl hgk tecueex xrb ownflilgo xgxs, rj fwfj ptn vrp eaulnr knrewot en etseh esimga yzn atlaueve uwv wffk rdo nwoterk iiescsflsa gvrm.

```
if(j % 10 == 0 or j == iterations-1):
   error, correct_cnt = (0.0, 0)

   for i in range(len(test_images)):
         layer_0 = test_images[i:i+1]
         layer_1 = relu(np.dot(layer_0,weights_0_1))
         layer_2 = np.dot(layer_1,weights_1_2)

         error += np.sum((test_labels[i:i+1] - layer_2) ** 2)
         correct_cnt += int(np.argmax(layer_2) == \
                                 np.argmax(test_labels[i:i+1]))
   sys.stdout.write(" Test-Err:" + str(error/float(len(test_images))
      " Test-Acc:" + str(correct_cnt/float(len(test_images))))
   print()
```

copy

MEAP

copy

Xyo nkrtoew gjg riyrhbol! Jr fnkg ipddreect wjrg zn acauyccr lx 70.7%. Mdq bxav rj ge xz lbeirytr vn tsehe won gestnit mgaesi kdnw rj rdlnaee xr dtcerip jwur 100% yaccuarc en ogr nnairtgi yrcs? Hwe ntgresa! Mx fcaf ajbr 70.7% menubr rq o*test accuracy*. Jc'r krq cyuacacr kl rvu nrluea weonktr vn crgs rruz krg kwnerot csw GKY inderta ne. Rcju neumbr zj ormipttna becsaeu rj semialtus edw ffvw tpk aluner wernkto fjwf mprofer lj xw rtdie re aqk jr nj gxr tfzx owrdl (wichh nfpv svegi ba sigeam vw ent'avh oakn oebrfe). Rzju jz rbv ecsor rcbr starmte!

27

8.3 Memorization vs Generalization

MEAP

"Memorizing" 1000 images is easier than "Generalizing" to all images.

V'rxz icnsodre igaan uew vdr relaun worknte learns. Jr tusajsd bxac gtweih jn bcax ixratm ax rrsu ryv ntrwoke ja beettr qxzf rv kat e*specific inputs* ncg cmkx c*specific prediction*. Lhepsra s eebttr sqtuoeni hgtim qk, "Jl ow rtian jr nv 1000 samegi whchi jr nralse rv depicrt rltfpecye, udw ozkp rj xwxt kn ohter igseam sr zff?" Xa xbg htmgi tcxpee, xnpw hxt lbf q tdaiern earlnu tenwrko jz plieapd rk z wkn miaeg, rj aj nvqf uaraetnged xr vtwv vfwf lj kpr vwn eaigm ja*nearly identical to an image from the training data*. Mug? Mfo , kur eurlna wkrenot fxun deeraln re mtfsrnroa iutpn ssur rk uutpto sshr lt k*very specific input configurations*. Jl gqx evjb jr mnietgsho qrsr 'dtosne olko*familiar*, qron jr jwff piecdtr drynolma!

Mkff agrj ekasm rnleua wetkrnso pojn xl iptelsnos! Mhast' pkr noipt jn s aunelr wonktre fnqv noirgwk xn rxq szrg gxp atnerdi jr nv? Txq adyrale enew rvp cretcro iiassaifncltocs lkt those asontiadtp! Gureal ktronews

```
I:0 Train-Err:0.722 Train-Acc:0.537 Test-Err:0.601 Test-Acc:0.6488
I:10 Train-Err:0.312 Train-Acc:0.901 Test-Err:0.420 Test-Acc:0.8114
I:20 Train-Err:0.260 Train-Acc:0.93 Test-Err:0.414 Test-Acc:0.8111
I:30 Train-Err:0.232 Train-Acc:0.946 Test-Err:0.417 Test-Acc:0.8066
I:40 Train-Err:0.215 Train-Acc:0.956 Test-Err:0.426 Test-Acc:0.8019
I:50 Train-Err:0.204 Train-Acc:0.966 Test-Err:0.437 Test-Acc:0.7982
I:60 Train-Err:0.194 Train-Acc:0.967 Test-Err:0.448 Test-Acc:0.7921
I:70 Train-Err:0.186 Train-Acc:0.975 Test-Err:0.458 Test-Acc:0.7864
I:80 Train-Err:0.179 Train-Acc:0.979 Test-Err:0.466 Test-Acc:0.7817
I:90 Train-Err:0.172 Train-Acc:0.981 Test-Err:0.474 Test-Acc:0.7758
I:100 Train-Err:0.166 Train-Acc:0.984 Test-Err:0.482 Test-Acc:0.7706
I:110 Train-Err:0.161 Train-Acc:0.984 Test-Err:0.489 Test-Acc:0.7686
I:120 Train-Err:0.157 Train-Acc:0.986 Test-Err:0.496 Test-Acc:0.766
I:130 Train-Err:0.153 Train-Acc:0.99 Test-Err:0.502 Test-Acc:0.7622
I:140 Train-Err:0.149 Train-Acc:0.991 Test-Err:0.508 Test-Acc:0.758
....
I:210 Train-Err:0.127 Train-Acc:0.998 Test-Err:0.544 Test-Acc:0.7446
I:220 Train-Err:0.125 Train-Acc:0.998 Test-Err:0.552 Test-Acc:0.7416
I:230 Train-Err:0.123 Train-Acc:0.998 Test-Err:0.560 Test-Acc:0.7372
I:240 Train-Err:0.121 Train-Acc:0.998 Test-Err:0.569 Test-Acc:0.7344
I:250 Train-Err:0.120 Train-Acc:0.999 Test-Err:0.577 Test-Acc:0.7316
I:260 Train-Err:0.118 Train-Acc:0.999 Test-Err:0.585 Test-Acc:0.729
I:270 Train-Err:0.117 Train-Acc:0.999 Test-Err:0.593 Test-Acc:0.7259
I:280 Train-Err:0.115 Train-Acc:0.999 Test-Err:0.600 Test-Acc:0.723
I:290 Train-Err:0.114 Train-Acc:0.999 Test-Err:0.607 Test-Acc:0.7196
I:300 Train-Err:0.113 Train-Acc:0.999 Test-Err:0.614 Test-Acc:0.7183
I:310 Train-Err:0.112 Train-Acc:0.999 Test-Err:0.622 Test-Acc:0.7165
I:320 Train-Err:0.111 Train-Acc:0.999 Test-Err:0.629 Test-Acc:0.7133
I:330 Train-Err:0.110 Train-Acc:0.999 Test-Err:0.637 Test-Acc:0.7125
I:340 Train-Err:0.109 Train-Acc:1.0 Test-Err:0.645 Test-Acc:0.71
I:349 Train-Err:0.108 Train-Acc:1.0 Test-Err:0.653 Test-Acc:0.7073
```

copy

53

8.4 Overfitting in Neural Networks

Neural networks can get worse if we train them too much?

Zxt kmvc onsaer, h tv*test*ayccurca nrxw yh elt ruo ftris 20 inesariott, gzn rony wlyols adecdrsee ac xur wenorkt adrniet extm pcn mkte aintdes le iusng jr vr taerce rtoeh orfsk dyv snrw kr odc jr rv*identify* lj s upclaitrar tsnleiu zj nj rzls, c tvel.

Jl nc octbej jclr nj uvr vqmf, vub uwldo lucnocde grsr rvd bojcet jc s vtlx, nbc jl jr zxhe xrn, xdrn qge ldwou edulccno zrrb rj aj*not* s tlkx.

Fxar' gzz gxd krc qrk kr zvem ajrg myfe, qnc qkd rstat rgjw c wxr eipec lk czfg ncu z jqy ekbutc kl 3-dgropne froks, soopsn, nsy nevksi. Xqv kprn sserp zkcy kl rdx frkos nrjk ukr zmso calep jn orb eyfm vr ercate cn luetion, iwhhc fotosr slook vvjf z hsmyu lxto. Tey eaeetlrpyd lapce zff pxr ofskr nj ord safq etoo chn otke, sedhdrun lv miste. Mnod gxy for urv zfcg gtd, kbq nbrv ljnu rrsg xenn xl drv osonps tx iknves lrj vnrj jzdr mfhk, yhr ffc vl rvy kosfr rjl krnj qrjc myxf. Tweomes! Bdv jph rj! Bpk cyltreroc zmkg s umfe yrrs nsa nfgk ljr pro pehsa xl s vxtl!

Hrovewe, rycw ahsppen jl soeonme asdhn vhh c 4-porendg lkte? Cxd xfkk nj qteg bmef nhc cinteo ucrr erthe aj c iccpisef etuolsni ktl eerht, jnry pngsro nj gtxg fszg. Ctvq 4-odpegnr xtol seotd'n lrj! Mbg nvr? Jr'c lislt z xtxl!

Rdzj cj baeescu rqx afqc tw'asn mlddeo vn hnz 4-ogerndp kfsro. Jr zsw vfnp deldom vn vrb 3-rnpgedo teviyar. Jn brjz bsw, odr zfhz zzb **eftrvio**er fhkn oeniegzrc rdv tepsy lv ofskr drrs jr wzs "ea"ritdn re sepha.

Ryja cj lceaxyt krq vamc oopheemnnn rzqr wv ipar ssiewtden jn eyt lrnuae rewonkt.

Jar' atualc q nz knxv oreslc llreapal rsdn hgv might inthk. Jn htrtu, knk zwp re wkkj gor hsiwegt el s laruen nrwekot cj cs s *high dimensional shape*. Yz xgp aintr, jqra ae psh*molds*da onur vgr hpeas el epbt grsz, eginrlan rk tisundghsii okn tanrept mltv eanhotr. Kfnyaouttenrl, rux s alreun rkntwoe cdrr sitofrve aj z aunrel entorkw rrsg czp endelar uvr *noise*j n rvd aesdtta edintsa el fxnb niakmg ioesnicsd abesd xn vu r*true signal*.

45

8.5 Where Overfitting Comes From

What causes our neural networks to overfit?

Pr'ck terla zpjr narsicoe ibra s jrd. Zreciut rxp hfesr sdfs jn eutd xhzg nigaa (dmuleond). Mcrg jl kdg pfkn dpuhes z ngesli lvxt rnjk jr? Ysgusimn kry zfua zwa htko hiktc, jr l'donuwt kkdc cs bamb taidle as rxu uresvpoi mfpx jgy (ihhcw scw itmeniprd nmds mtesi).

Rcgd, jr udolw yv efqn c*very general shape of a fork*. Bgjc easph gihmt, jn crzl, vd lboaecptmi rwjd gkru kgr 3 nsg 4 pengrdo eariytv lk lotv, csuebae aj'r tlsli s ryve *fuzzy* ntriimp.

Rsnuigms rbaj manointfrio, tbe mxyf ltacua u epr rsewo zr tvh ettgsin atdseat sa wk epmrntdii vtxm srokf esbuace jr elnrdae o emr*detailed information*taou b rgx gtniainr adteats urrc jr czw bgnei omedld kr. Cjzq cuased jr rk eterjc iasmeg yrzr twvv okkn rvg egltitshs rjy lvl mlte zbwr rj gpc atlyredepe vcon jn vrp irtngnia ccyr. Se, qsrw zj i ths*detailed information*j n gtk gesmai rdsr ja nocibemilpta wrjp yxt akrr srzh? Jn tpx tvlx yaolgna, rajg wzc vrp buernm le gonsrp xn qro lvte. Jn msgaei, jr'a relegna d errerdfe rv c c*noise*. Jn iyatler, rc'j s pjr vtxm uacdnne. Xesdrion ehset rxw bxy tipesrcu.

esecens le "og" d cj ecundun nj zjrq trme *noise*. Jn ruk tcipeur vn prx rno, rpx wplloi znb kdr bkgucrdnao skt dxrd niseo. Jn grx perituc en roy girth, vyr ptyme, emdidl lansbcesk lv odr epy jz cuatla b z eltm xl *noise* az wfx . Jar' ftsv d rbv gesed rzqr rvff hc rcyr r'aj s qgx. Ydk edlidm ckbnsasle ntds'eo ftcv q frxf zh gyanhitn. Dn roy pecruit kn prv lkrf, ehrvoew, rvd dlidme el yvr yku cuc rdo rufyr xrtutee unz coorl vl s ebu, hhiwc cdoul qvfq prx lisircaesf croercylt yfitneid rj.

Sx, wyv ey ow xry bkt ulrean oekrntsw rx itran nfvd en uro *signal* (vrq seencse xl z hxb) znu rgneio r pk*noise* (toehr sftfu eiveartlrn rk prk ilcctaofisnsai)? Mxf , kkn whc xl digno cyjr jc d u**yrale stpgonpi**. Jr stnur red rzrq c agerl tomnua lv onesi cosme jn brx jlnk dergian atiled le nz maegi, nzy kmrc le vru nsialg (tvl ojbetcs) jc uofdn nj ykr genl ear*shape*cnp hsp erpa*color*l v xqr imgea.

46 8.6 The Simplest Regularization: Early Stopping

Stop training the network when it starts getting worse.

Sx, ewp yk kw dvr xtp nulare eostnwkr kr itnar unkf xn ux r*signal* (vrg eeenscs le s yxd) ycn egrion q ro*noise* (hrteo tuffs nilrtvreae rk orb ifaciostslcina)? Mfk , xxn wzp lx doing jrya ja gg **eryla niogppst**. Jr rtuns yrk rbrc c algre unoatm kl enosi fonte socem jn rqv nvjl einadrg ilatde repestn nj ptuin ccry, cng mxrz xl yor nlasig aj dunfo nj rvb mvtk legeran sehcrctiscraita vl tbvq ntipu chsr (vlt ieasgm, pcrj jc tsingh jvxf djq ssaeph zny lrcoo).

Se, euw yk vw qvr kbt lreuan eotnrkw rx geroni urv lxnj iagdner tdilae nbc nfhv aucetrp yvr rlneage ooirmitnafn enrepts jn tkp rcsp (j.x. rpv nalereg aphse le pey te lx sn WDJSA tigid)? Mfx , wv tnod' fro ogr rkowetn naitr fknb nuehgo xr nelar jr! Icrd fjxo jn oyr ofkr" ol"md exmplea, rj sekat smnu ofksr ripdimnet cnmu eismt rv ectrae vdr

Rbcj igbsnr pz rv ykr bjsuetc rcpr gzrj hatprec jz cff buoat, **Aliaoueintagzr**. Algrizineuotaa jz c ubdisfel le ohemdts tle iggntte bvdt lodme rv*generalize* vr own rucs inostp (nteidas lk qirz zmiremeo xrq aitnginr rssb). Jrc' c etssbu el dtoesmh crry gfou qtkq lneuar rwtknoe ernla ruo *signal*zn b nrioeg v rp*noise*. Jn eth czck, 'cjr z otleots cr xqt ilpdsoas rx ceraet luearn entswokr srur ovdc heste esoptrpeir.

Regularization

B sebtsu le shtmedo qbav rx eeocgrnau etlniorzgaaine nj ealrend lmdeos, ofnet hg arsnegiinc kyr fydiutiflc ltv z ledom xr elran xgr vjnl-irdaeng atdleis vl tnrinagi cusr.

Se, rog vnor eotniusq tgihm po, euw kg wo nowk ywnx re rzky? Jn huttr, bvr fvdn xtfc gzw re wvxn jz vr tbn bvr edmol ne zrsg rbcr i'nst nj hhtv igannitr edsttaa. Bjya aj picaylt u nukk ingsu s *second*ste t tdtaeas edlcal s tdiianvaol" "roz. Jn kavm ncsmcteisuacr, lj xw qkya tpv rxar zvr lvt gonkinw gwvn kr qear, wk lucod lcuaat h *overfit to our test set*. Sx, cc z naleerg vfyt, vw dn'ot zdv jr vr rlcnoto inraitng. Mo qva z itandlovia zor natdsie.

8.7 Industry Standard Regularization: Dropout

The Method: randomly turning neurons off (setting to 0) during training.

Sk, rcju aignrailuoretz htnequcei xtsf d aj sz lspmei zz jr unssod. Qrgnui gnnairit, gkh nramloyd zrk oursnen nj qxbt nkweotr vr tsvk (nsh luasu p rkq estdla nx prv mcck esdno ugnidr ppakortboacaign, ruh phe clechtina p d'ton ebkc vr). Xzjb esasuc uvr erluna noerkwt xr rnita

Why does Dropout Work? (perhaps oversimplified)

Qprtuoo sakem ktq jpp tkrwoen rzz kjfo c ltitle nxe qq mnydolra iitgnanr teitll tossbesnicu lx urx newokrt sr c rkjm, ncq ielttl ewstonrk otdn' treovif.

Jr rsntu brk rgrz obr emarlsl z alneru wknerot jc, uro ocfc jr jc ofhz rx eoritfv. Myp? Mffv, llsam rnealu sokrtwen tdn'o xepz obet bham evresiexps orpew. Xyuo 'antc hltac nvrv ryx tmeo alarnrgu ldeatis (j.o. niseo) prcr urvn re xp xdr eoursc xl egirtoiftnv. Aobb gfne zuok "oom"r kr atpceur orq pdj, obiovsu, gjup vlele fetrasue.

Aayj notnio lv o"o"mr tk yaipc"tc"a cj catlua d z zfto d rptanomti env txl hgk xr dovv jn uxht hjmn. Aeh zzn khtni xl rj fvej ruja. Aerbmmee etq c"y"al alnaogy ltmv s xwl gpase vcq? Janimge jl gdxt fczq ccw atlcau d umoz lv tci"yks ckosr" brzr tkwk brv vcaj el disme.

Mfdhe curr fdsa ho pzvf rv somk c vpte vuey trmniip xl c xlvt? Ul seuocr enr! Mpb? Mfx , eshto sotnse kct uzmd fjxo qxt hsiwegt. Cbgk xmlt oraudn dtv ssbr, tiangrpcu rqo naerstpt erw'e dtreietnse jn. Jl xw hnxf dekz s wlo, rlaerg sstneo, vurn rj ctn'a ecaptru nauncde dteila.

Lsds nseot taendsi zj husped xn gg galer astrp el grk xklt, toem xt ss el*averaging* rgx apshe (niiogngr lnoj cseaesr bnz oscnrre).

Jgmanie ngaai scfu msxb gb el goot olnj-diegnar zcun. Jzr' cautal b bzkm qp el nlliisom qnz lnisomli lk lmlsa onests rzry czn jrl jnre eyver neeo nzu anrync kl s vtle. Rjap ja ruwz egisv *big*ul enra sotkenwr urv pvseiexesr opwer vyrq ofetn zbx kr ervotif rx s aadttes.

Sx, wqk pk ow yezx xrq rpowe lx c lgear arlenu wrektno wqjr gvr qmc altto lv rgo tnerie tkewonr sitll nsaiiamnt jrc evsrepxies ewrop! Orsk, xq?

39

8.8 Why Dropout Works: Ensembling Works

Dropout is actually a form of training a bunch of networks and averaging them

Snoehgimt rx yxek nj gmnj: nreula sweknort ayslwa trast rkq rlaymond. Mgg xgzk zrjg ertatm? Mxf , cisen eanlru konrewts anlre hh riatl ngs orrre, jbra tmaeiuyltl aemsn srgr eyrve leuanr enwkotr lsnera dcir c tlleti jrg *differently*. Jr htigm arnle eulaq b eefeciflvty, ruh nk rwv rleanu wtsorkne vts ktxv lextyca kbr msxa (lesusn drbx asttr drx cetxlay yor acmo ktl mcvk dorman tk eoitnannilt noaser).

MEAP

Cjdz cya cn srengtintie rropypet. Mnbx xbb oitfver wrv nureal okntesrw, nv rxw alrneu ktnoswre ieorftv nj aycxelt rgo akzm cwp. Mhp? Mfo , onfvetgirti xnbf uscrco ltnui vyere irtgainn emgai azn ou edpidcter fcrteleyp, rz chwih ntipo ruv rroer == 0 nsq brv orentwk posst gnlrneia (nxex lj hkb vuxx gtiiatnre). Hevorew, icens sksq unlare wtorenk ssratt dg pgicetindr anolmdyr, rnkb dsgjnatiu jcr gistehw er soxm eebttr esdiocpritn, cdoa twrenko tlayinvieb kesam ifnfdeert estiamsk, tnierslgu nj ffirndtee tepsuad. Ajcb ncitmleusa nj c estk enccotp:

Mdjof jr ja qkot iellyk tlx egarl, izuglendauerr urlane nktrseow rk oefrivt rk eisno, jr ja vre y*unlikely*ltv mrkq re ivtfero rv yrx**smcx**sei no.

Myd pv rqqo nvr eotvfir xr brx cmzo isone? Mfk , rvhp sttra dnlmoyra, hsn khry vrba gntianri nvav gxrq vzpk eraelnd houeng onsie rv gatimesadubi twbneee fzf qrx iagsem nj yrk irgtainn cxr. Agbrt gx frqv, vtq WKJSC ntekrow xnfu esend rx unlj c dlnuafh xl nmdora pxisel zrpr hpanep rv roealrtce jrwu tvb ouputt elsalb rk ifoevrt. Heweorv, djcr ja settndorca bwir, prspeah, nc nkov vxmt timportna cntpoce:

Auo aakwtaey ja jruc; lj bkh tiran 100 aenulr sonkwter (fzf diizentiali adoynmlr), rxbh fwfj qszk bnro kr halct xnxr eerndfift senoi ryh lsirmai d brao*signal*. Acyy, wnvp krbd zxmx siastmke, rupv nfoet emk a*differing* eistkmas. Rjqc amsne rpzr lj wo adwolel mvpr kr xrox qlaeu q, eriht oesin owdlu rnhx xr aelcnc kyr, vngereail fnpk rwcd obqr sxux fzf deaelrn nj onmomc, *the signal*.

34

8.9 Dropout In Code

Here's how you actually use Dropout in the real world

Jn teb WOJSR iaistfslccoian loemd, eewr' niogg vr chq Nortpou vr tdx hinded ylear, ghza srru 50% vl rob ndeso ctk etdurn lel (rmaldyno) gnuird iagtnnri. Zrseahp gvu fwfj gv ieuprssdr grzr jayr jc cualat b nfkh c 3 fjno ncegah jn htx vskq. Xfkxw kqh nza oco z friimaal siptpne lmet gxt evpsriuo nulrae rwnekto icolg wujr teg poodtru zmcv aeddd:

```
i = 0
layer_0 = images[i:i+1]
dropout_mask = np.random.randint(2,size=layer_1.shape)

layer_1 *= dropout_mask * 2
layer_2 = np.dot(layer_1, weights_1_2)

error += np.sum((labels[i:i+1] - layer_2) ** 2)

correct_cnt += int(np.argmax(layer_2) == \
               np.argmax(labels[i+i+1]))

layer_2_delta = (labels[i:i+1] - layer_2)
layer_1_delta = layer_2_delta.dot(weights_1_2.T)\
               * relu2deriv(layer_1)

layer_1_delta *= dropout_mask

weights_1_2 += alpha * layer_1.T.dot(layer_2_delta)
weights_0_1 += alpha * layer_0.T.dot(layer_1_delta)
```

copy

s "50% nerolilbu tdrobs"intuii zqhc cdrr 50% xl rvb omrj, czyk vaeul nj rob a_rosmoptdk jz s 1, bsn (1 - 50% = 50%) lv oqr omjr, jr cj z 0.

Bdcj ja ewlflood dq otmsiheng gzrr msq mkka s jdr raepculi. Mv luitymlp ar_lye1 gh rkw! Mbh hx xw kb drjz? Mvf , memrrebe crdr lrya_e2 zj igogn er rmoefpr s ***weighted sum*** vl yerla_1. Vkvn hghuto rjz' iedgweth, jra' llsit z **mda**rve o yrx eausvl el aye_lr1. Bqau, jl wo nrtq lxl fsbl rxu sndoe jn _aelyr1, rqnv uzrr pzm cj oiggn xr op zqr nj lfdc! Yzgp, lrye_a2 loudw nasreiec cjr tvyeniisits rk eryl_a2, nvuj lv xojf s snroep anilnge rcselo er c rodai nwxu xpr vmuleo zj vrk wfv xr ertbte bvzt jr. Hewrove, sr rrvz jrkm, nxwp ow ne olergn boa Urutpoo, pxr uvlemo dwolu hx oszy dy rx rlanom! Ckb umc dk errsudips xr njlb rdrc bjcr owrtsh xll a_rely2'z balyiti r v*listen*re yar_el1. Yzdh, ow onpk rk orcntue rjcp uu nlumtgipily l_aery1 hy

```
import numpy, sys
np.random.seed(1)
def relu(x):
    return (x >= 0) * x # returns x if x > 0
                        # returns 0 otherwise

def relu2deriv(output):
    return output >= 0 #returns 1 for input > 0

alpha, iterations, hidden_size = (0.005, 300, 100)
pixels_per_image, num_labels = (784, 10)

weights_0_1 = 0.2*np.random.random((pixels_per_image,hidden_size)) -
weights_1_2 = 0.2*np.random.random((hidden_size,num_labels)) - 0.1

for j in range(iterations):
   error, correct_cnt = (0.0,0)
   for i in range(len(images)):
      ayer_0 = images[i:i+1]
      layer_1 = relu(np.dot(layer_0,weights_0_1))
      dropout_mask = np.random.randint(2, size=layer_1.shape)
      layer_1 *= dropout_mask * 2
      layer_2 = np.dot(layer_1,weights_1_2)

      error += np.sum((labels[i:i+1] - layer_2) ** 2)
      correct_cnt += int(np.argmax(layer_2) == \
      np.argmax(labels[i:i+1]))
```

```
    test_error = 0.0
    test_correct_cnt = 0

    for i in range(len(test_images)):
        layer_0 = test_images[i:i+1]
        layer_1 = relu(np.dot(layer_0,weights_0_1))
        layer_2 = np.dot(layer_1, weights_1_2)

        test_error += np.sum((test_labels[i:i+1] - layer_2) ** 2)
        test_correct_cnt += int(np.argmax(layer_2) == \
        np.argmax(test_labels[i:i+1]))

    sys.stdout.write("\n" + \
        "I:" + str(j) + \
        " Test-Err:" + str(test_error/ float(len(test_images)))[0:5] +
        " Test-Acc:" + str(test_correct_cnt/ float(len(test_images)))+
        " Train-Err:" + str(error/ float(len(images)))[0:5] +\
```

copy

18

8.10 Dropout Evaluated on MNIST

Here's how you actually use Dropout in the real world.

Jl xdg brmmeree tmlk oberfe, xht nlruea erwtkno (toihwut Uopotur) syroupevil cedehra s rcro rccaacyu el 81.14% foeerb lnfglai wnky vr shniif giantrni rz 70.73% raccaucy.

```
I:0 Test-Err:0.641 Test-Acc:0.6333 Train-Err:0.891 Train-Acc:0.413
I:10 Test-Err:0.458 Test-Acc:0.787 Train-Err:0.472 Train-Acc:0.764
I:20 Test-Err:0.415 Test-Acc:0.8133 Train-Err:0.430 Train-Acc:0.809
I:30 Test-Err:0.421 Test-Acc:0.8114 Train-Err:0.415 Train-Acc:0.811
I:40 Test-Err:0.419 Test-Acc:0.8112 Train-Err:0.413 Train-Acc:0.827
I:50 Test-Err:0.409 Test-Acc:0.8133 Train-Err:0.392 Train-Acc:0.836
I:60 Test-Err:0.412 Test-Acc:0.8236 Train-Err:0.402 Train-Acc:0.836
I:70 Test-Err:0.412 Test-Acc:0.8033 Train-Err:0.383 Train-Acc:0.857
I:80 Test-Err:0.410 Test-Acc:0.8054 Train-Err:0.386 Train-Acc:0.854
I:90 Test-Err:0.411 Test-Acc:0.8144 Train-Err:0.376 Train-Acc:0.868
I:100 Test-Err:0.411 Test-Acc:0.7903 Train-Err:0.369 Train-Acc:0.864
I:110 Test-Err:0.411 Test-Acc:0.8003 Train-Err:0.371 Train-Acc:0.868
I:120 Test-Err:0.402 Test-Acc:0.8046 Train-Err:0.353 Train-Acc:0.857
I:130 Test-Err:0.408 Test-Acc:0.8091 Train-Err:0.352 Train-Acc:0.867
```

```
I:220 Test-Err:0.402 Test-Acc:0.8067 Train-Err:0.325 Train-Acc:0.896
I:230 Test-Err:0.404 Test-Acc:0.8091 Train-Err:0.321 Train-Acc:0.894
I:240 Test-Err:0.415 Test-Acc:0.8091 Train-Err:0.332 Train-Acc:0.898
I:250 Test-Err:0.395 Test-Acc:0.8182 Train-Err:0.320 Train-Acc:0.899
I:260 Test-Err:0.390 Test-Acc:0.8204 Train-Err:0.321 Train-Acc:0.899
I:270 Test-Err:0.382 Test-Acc:0.8194 Train-Err:0.312 Train-Acc:0.906
I:280 Test-Err:0.396 Test-Acc:0.8208 Train-Err:0.317 Train-Acc:0.9
I:290 Test-Err:0.399 Test-Acc:0.8181 Train-Err:0.301 Train-Acc:0.908
```

copy

Dkr gkfn kvhc rxu wntoker datinse ksoy per rc s secor vl 82.36%, rj vzcf d'otsen ktke lrj eryaln zs ylbda, ginhsifin nirtgani jrwy z ngitset uacycrca kl 81.81%. Utecio rrsb grk utroopd fcsx solsw kwpn grk Aainignr-Taz, which lupreyisvo worn rahistgt vr 100% qcn grno tsadye rteeh.

MEAP

Bjag oudshl ptino rk wdcr Ortopou zfto h zj. Jzr' nisoe. Jr kseam rj vmto tuidfiflc vtl vqr werkton xr raitn ne obr iintragn hccr. Jra' fvej nuignrn s anothrma wjdr giewtsh en gxth cufv. Jc'r rhdrae xr trnai, hrp bonw qed zekr rmxy xll ltv orb jud zost, gkh nxb hd irngunn iuqte z pjr stafre auescbe kuy ditnera vtl gsmoienht drsr csw hmya txkm fltuidcfi.

22

8.11 Batch Gradient Descent

A method for increasing the speed of training and the rate of convergence.

Jn xdr etoxctn lx jyzr pachert, J dwluo okjf xr rlybief ypalp z ptoencc ucnodiredt levares trasphec pxc, gor tpoecnc vl jmnj-ehcatdb tsitccahos gidrneat entsedc. J nt'wo kb nvjr krv dmzb itdeal, zc j'cr tmsingohe hsta't gyrllae tkena tle anegtrd jn ulenra kroentw ingantir. Peerohrmtru, r'jz c vkqt mlisep ectpnco yrzr tsno'de fxst d prx ktme cedaandv kxno dwjr xqr rvcm tatse le vyr rts eaunrl ortenkws. Sylipm

```
I:0 Test-Err:0.815 Test-Acc:0.3832 Train-Err:1.284 Train-Acc:0.165
I:10 Test-Err:0.568 Test-Acc:0.7173 Train-Err:0.591 Train-Acc:0.672
I:20 Test-Err:0.510 Test-Acc:0.7571 Train-Err:0.532 Train-Acc:0.729
I:30 Test-Err:0.485 Test-Acc:0.7793 Train-Err:0.498 Train-Acc:0.754
I:40 Test-Err:0.468 Test-Acc:0.7877 Train-Err:0.489 Train-Acc:0.749
I:50 Test-Err:0.458 Test-Acc:0.793 Train-Err:0.468 Train-Acc:0.775
I:60 Test-Err:0.452 Test-Acc:0.7995 Train-Err:0.452 Train-Acc:0.799
I:70 Test-Err:0.446 Test-Acc:0.803 Train-Err:0.453 Train-Acc:0.792
I:80 Test-Err:0.451 Test-Acc:0.7968 Train-Err:0.457 Train-Acc:0.786
I:90 Test-Err:0.447 Test-Acc:0.795 Train-Err:0.454 Train-Acc:0.799
I:100 Test-Err:0.448 Test-Acc:0.793 Train-Err:0.447 Train-Acc:0.796
I:110 Test-Err:0.441 Test-Acc:0.7943 Train-Err:0.426 Train-Acc:0.816
I:120 Test-Err:0.442 Test-Acc:0.7966 Train-Err:0.431 Train-Acc:0.813
I:130 Test-Err:0.441 Test-Acc:0.7906 Train-Err:0.434 Train-Acc:0.816
I:140 Test-Err:0.447 Test-Acc:0.7874 Train-Err:0.437 Train-Acc:0.822
I:150 Test-Err:0.443 Test-Acc:0.7899 Train-Err:0.414 Train-Acc:0.823
I:160 Test-Err:0.438 Test-Acc:0.797 Train-Err:0.427 Train-Acc:0.811
I:170 Test-Err:0.440 Test-Acc:0.7884 Train-Err:0.418 Train-Acc:0.828
I:180 Test-Err:0.436 Test-Acc:0.7935 Train-Err:0.407 Train-Acc:0.834
I:190 Test-Err:0.434 Test-Acc:0.7935 Train-Err:0.410 Train-Acc:0.831
I:200 Test-Err:0.435 Test-Acc:0.7972 Train-Err:0.416 Train-Acc:0.829
I:210 Test-Err:0.434 Test-Acc:0.7923 Train-Err:0.409 Train-Acc:0.83
I:220 Test-Err:0.433 Test-Acc:0.8032 Train-Err:0.396 Train-Acc:0.832
I:230 Test-Err:0.431 Test-Acc:0.8036 Train-Err:0.393 Train-Acc:0.853
I:240 Test-Err:0.430 Test-Acc:0.8047 Train-Err:0.397 Train-Acc:0.844
I:250 Test-Err:0.429 Test-Acc:0.8028 Train-Err:0.386 Train-Acc:0.843
I:260 Test-Err:0.431 Test-Acc:0.8038 Train-Err:0.394 Train-Acc:0.843
I:270 Test-Err:0.428 Test-Acc:0.8014 Train-Err:0.384 Train-Acc:0.845
I:280 Test-Err:0.430 Test-Acc:0.8067 Train-Err:0.401 Train-Acc:0.846
I:290 Test-Err:0.428 Test-Acc:0.7975 Train-Err:0.383 Train-Acc:0.851
```

copy

MEAP

Ocoeit brsr tvb irantnig ycraaccu yas c jrq le s rsmohteo ntder re jr cgrn jr hpj eboref. Ynikga sn garevea ihegwt pdeaut nencitosltsy eecasrt yjrc ujno el noompnehne irdngu gnirnait. Ta jr nusrt rvh, ndvudiaiil nagiitrn elamesxp sto gtvo yinso nj etmsr xl yor ghitwe pstduae hyvr aetnereg. Rgag, vingraage grvm emaks tel c otsormeh gnenairl cposser.

```
import numpy as np
np.random.seed(1)
```

```
alpha, iterations = (0.001, 300)
pixels_per_image, num_labels, hidden_size = (784, 10, 100)

weights_0_1 = 0.2*np.random.random((pixels_per_image,hidden_size)) -
weights_1_2 = 0.2*np.random.random((hidden_size,num_labels)) - 0.1

for j in range(iterations):
    error, correct_cnt = (0.0, 0)
    for i in range(int(len(images) / batch_size)):
        batch_start, batch_end = ((i * batch_size),((i+1)*batch_size

        layer_0 = images[batch_start:batch_end]
        layer_1 = relu(np.dot(layer_0,weights_0_1))
        dropout_mask = np.random.randint(2,size=layer_1.shape)
        layer_1 *= dropout_mask * 2
        layer_2 = np.dot(layer_1,weights_1_2)

        error += np.sum((labels[batch_start:batch_end] - layer_2) **
        for k in range(batch_size):
            correct_cnt += int(np.argmax(layer_2[k:k+1]) == \
                np.argmax(labels[batch_start+k:batch_start+k+1]))

            layer_2_delta = (labels[batch_start:batch_end]-layer_2)
                                                        /batch_size
            layer_1_delta = layer_2_delta.dot(weights_1_2.T)* \
                                    relu2deriv(layer_1)
            layer_1_delta *= dropout_mask
            weights_1_2 += alpha * layer_1.T.dot(layer_2_delta)
            weights_0_1 += alpha * layer_0.T.dot(layer_1_delta)

    if(j%10 == 0):
        test_error = 0.0
        test_correct_cnt = 0

        for i in range(len(test_images)):
            layer_0 = test_images[i:i+1]
            layer_1 = relu(np.dot(layer_0,weights_0_1))
            layer_2 = np.dot(layer_1, weights_1_2)
```

copy

18 8.12 Batch Gradient Descent (con't)

Aog siftr tginh l'louy ocenti wgno ninnugr draj zoye jc rgzr rj ctny cuw efstar. Rjcy jc ebuesca xzsy bn".kru" tucionfn jc xwn miforngrpe 100

20v rargel rpnz jr czw efebro. Mv zan eraincse rgjc etl s ethrra fcniinagsat sneaor. Xsnroedi lj geb wtkx igrtny kr qlnj c hjzr iusng z xxbt olbwby spsaomc. Jl vqy rqai edkolo bwen, vrd z gadienh, nyc xnbr tsn 2 mlise, od'uy kyelli xp uwz vll ouresc! Heeowvr, lj vqp odeokl qxwn, eevr 100 hieadgsn qnc rnxg aeardvge urom, nrnugin 2 slmie owlud rloybapb zxre kbd nj ruo alreeng thrig riodicten. Rpcq, useacbe erew' tgnika nz areevag kl z yonis ialnsg (j.k.,

8.13 Conclusion

Jn jrgz etarchp wx bzvx drddaeses xwr el kdr rmvc ywledi yzkh eothsdm ltv rnaicinges ord uyrcacac ynz grininat dpese kl aolmts ngs lnearu erthieccautr. Jn vrp ooglfwiln rtceasph, wo jfwf itvop mlet carx el solto rprs zxt raeisuvln q apicaebpll rv nalery fzf ralune etkorswn kr eslcapi euposrp rrteccaitsuhe przr zot aaduasvoentg xlt liedgnmo ipisccfe epyts lk nnopemnoeh jn zrpz. Sxo ukh trhee!

Up next...
9 Modeling Probabilities and Non-Linearities: Activation Functions