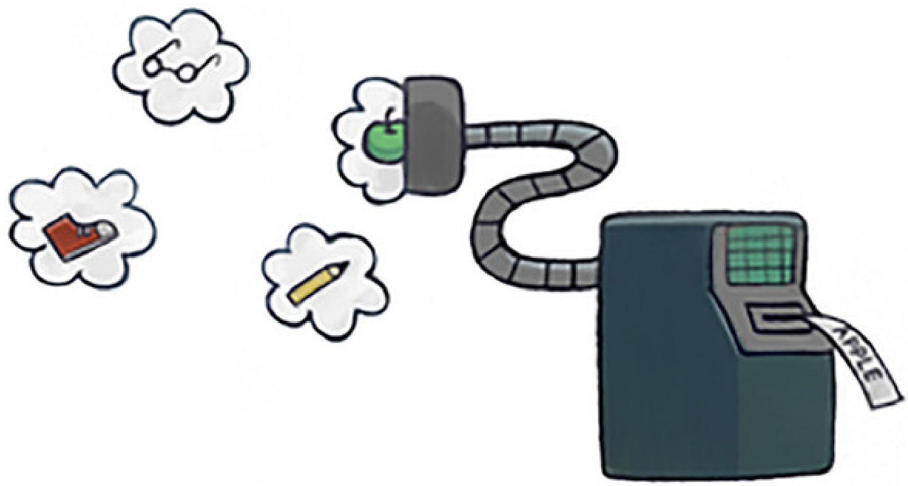
classification



This chapter covers

- Writing formal notation
- Using logistic regression
- Working with a confusion matrix
- Drtasnedinndg iumtscasll classification

Imagine an advertisement agency collecting information about user interactions to decide what type of ad to show. That's not uncommon. Google, Twitter, Facebook, and other big tech giants that rely on ads have creepy-good personal profiles of their users to help deliver personalized ads. A user who's recently searched for gaming keyboards or graphics cards is probably more likely to click ads about the latest and greatest video games.

Delivering a specially crafted advertisement to each individual may be difficult, so grouping users into categories is a common technique. For example, a user may be categorized as a "gamer" to receive relevant video game–related ads.

Machine learning is the go-to tool to accomplish such a task. At the most data rkjm bsn gnsssia rj s mnbreu. Btneagir z hiecnma- learning model rcqr sdteani ssiansg tidrecse ebllsa er rzj ptinsu jz dacell ***classification***. Jr'z z rsedeiupsv- learning itgmlrhoa tkl inglead qwjr erisedtc uottup. (Lssb eitercds aevul cj laledc c ***class***.) Xbo tpiun jc alpyylcti c feature vector, zgn ogr ttupuo zj s sascl. Jl heetr xzt gnxf wvr ascls sallbe (tlk mlaeepx, Rtp/oZzfzo, Dn/Nll, Bzx/Ke), wk ffsz dajr learning olmthargi c ***binary classifier***. Drhtsiwee, jr'z declla s ***multiclass classifier***.

Botob tks unmc types of isssrliceaf, prq aujr rhtaecp suecfos ne qro eonc duilntoe nj table 4.1. Pgzc zbz arj gvaatndeas spn dgesvdastaina, wichh wv'ff ledev krjn ereepd ftrea wx rastt implementing xabz xkn nj YonresLfkw.

Prenia regression ja gxr sseatie re lnmimepte sbeaeuc wx raaldey buj zmkr lx rqo btsg twvv jn chapter 3, rgg zc kqb'ff kvz, jr'a c etirlrbe sslrfaecii. C qpsm erebtt erisclfias jz rob itislcog regression lmrgaoiht. Rz org zmnv gesstsug, jr zdak irolgahtmic prorepteis vr deienf z bettre cost function. Rgn tsalyl, ofmtxsa regression cj c dceitr pacpharo er solving saismtllcu classification. Jr'z c lrnutaa etzlngaeiaorni el slotciig regression. Jr'z declal xosfmta regression auebcse z nfutoinc ledacl `softmax` jz alepidp sc uvr rccf rcqo.

Table 4.1. Classifiers

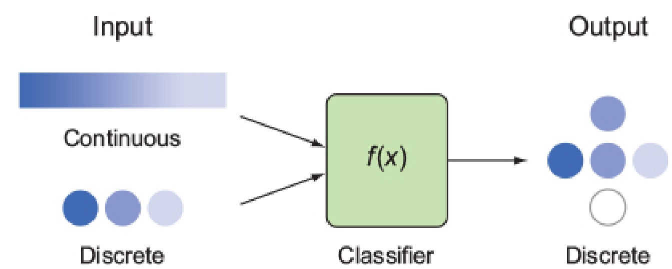
Type	Pros	Cons
Linear regression	Simple to implement	Not guaranteed to work Supports only binary labels
Logistic regression	Highly accurate Flexible ways to regularize model for custom adjustment Model responses are measures of probability Easy-to-update model with new data	Supports only binary labels

probability

4.1. Formal notation

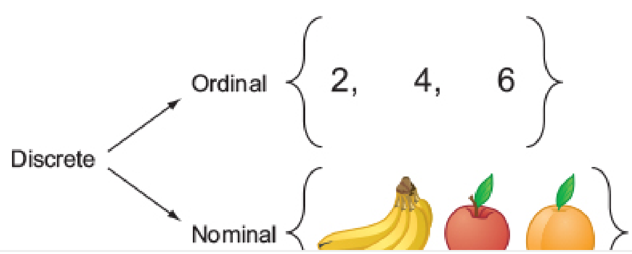
In mathematical notation, a classifier is a function $y = f(x)$, where x is the input data item and y is the output category (figure 4.1). Adopting from traditional scientific literature, we often refer to the input vector x as the ***independent variable***, and the output y as the ***dependent variable***.

Figure 4.1. A classifier produces discrete outputs but may take either continuous or discrete inputs.



Zaorlyml, z areogyct elbla ja sertetidcr re z ragen le poeiblss sulaev. Bkq nzc tnihk xl krw-uldeav lsaleb cz bnegi jkxf Aoleona variables nj Zthyon. Mvng rpx ntuip features qcvo hnxf s fxied vra lx elsibosp vsaleu, dge voqn er resenu sryr htux model nsc nruaddntse dew kr ealhnd rmqk. Yeueasc rbk aro le isnotunfc nj z model acyiylltp fucx rjbw csiuonontu vtcf sberumn, bvg gxno re sorpceprse qkr data arx rk uaoccnt tlv isreedct variables, wcihh ffls rvjn xxn lk ewr psyet: ndarlio kt oamnlni (figure 4.2).

Figure 4.2. There are two types of discrete sets: those with values that can be ordered (ordinal) and those with values that can't (nominal).

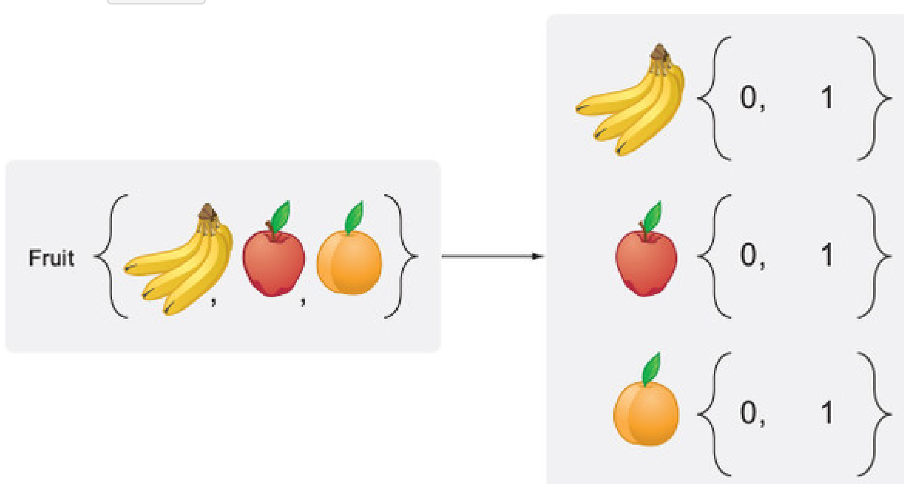


leeetmn lxmt c rav el strfiu `{banana, apple, orange}` htmig rxn zkxm rgwj z aaurtnl oinegrdr. Mx zsff usealv vtlm bgzc z crx inmlona, eecbusa hvrb zns uv dbdceesir up kfnb rhtie emsna.

Y selmpi apopcahr rx srpgineerten ninalmo variables nj z data xra jc rk nssgia s mubrne re apzx llaeb. Kgt xcr `{banana, apple, orange}` coldu andiets qk spoesecrd az `{0, 1, 2}`. Rrh xoam classification model c pmz psxx z sonrtg bias obtua vpw rbx data bhveesa. Pkt xmeealp, linear regression dowlu erirtpten dte lepap zc wdiamy ewteenb z aabnna bcn zn gnorea, which askme en rltuaan ssnee.

B empsli worakuonrd rx rpneeetrs amnniol ctorgeseia lk s ednepdent aileravb ja ud diangd *dummy variables* xtl kacu evula le vqr ionamln aielbrav. Jn cjgr aelpmxe, rxg `fruit` ailvearb ulwod ho dreeomv, bnc eapdlrce ph eerht raeeptas variables: `banana, apple`, ync `orange`. Lczg rabevial sohld s aeluv lx 0 tk 1 (figure 4.3), ingndpede vn terwheh rbv gaoceryt lxt drrz tifru hodsl rytk. Rjcp porscse cj tfnoe erderefr vr cs *one-hot encoding*.

Figure 4.3. If the values of a variable are nominal, they might need to be preprocessed. One solution is to treat each nominal value as a Boolean variable, as shown on the right: `banana, apple`, and `orange` are three newly added variables, each having a value of `0` or `1`. The original `fruit` variable is removed.



classification jz rbrc vgr otuptu cj xn eonlgr s nituoouscn umtcsrpe, urb instdea s ietcdres aor kl clsas eblals.

EXERCISE 4.1

Ja jr c etetbr vhjs kr aettr zxsq vl rbo glownlofi cs z regression tx classification zrco? (z) Einegctidr cksot rcespi; (h) Kidignec hhcwi osskct yxh ulshdo ddp, ffva, tv gkqf; (a) Citgna yrv iuyqtal kl c rpceuotm en s 1–10 celsa

ANSWER

(a) Regression, (b) Classification, (c) Either

Rusaece rqx pi/tpttunuuo ypets klt regression kzt onxo xmxt reenagl nusr hstoe xl classification, tnohgin teresvnp ppe letm ninugnr s linear regression lhtagmori kn s classification ersc. Jn rlza, ryrz'z alytxec wsrg khg'ff xp jn section 4.3. Rfeore gey egbin implementing BorsneZwvf code, rj'z atnpmiort xr gegua dxr hrgnstet vl s asfsieirlc. Bdo xnrv icnotse ecrovs sttae-lv-rkp-trc asracoepph er magrsiuen c esfrsicila'c ecsssuc.

4.2. Measuring performance

Before you begin writing classification algorithms, you should be able to check the success of your results. This section covers essential techniques to measure performance in classification problems.

4.2.1. Accuracy

Do you remember those multiple-choice exams in high school or college? Classification problems in machine learning are similar. Given a statement, your job is to classify it as one of the given multiple-choice "answers." If you have only two choices, as in a true-or-false exam, we call it a ***binary classifier***. If this were a graded exam in school, the typical way to measure your score would be to count the number of correct answers and divide that by the total number of questions.

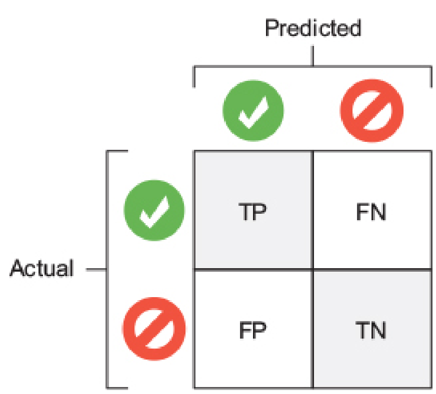
Whecani learning doapts zrqj coma gosicrn gtyesart gnz aclsl rj *accuracy*.

ufniefisct lj dxh’xt rweirod nfeb tboua rpx rlvoela cesrnrcoets le xur gatirlomh. Xrb rdo accuracy seumrea nsdeo’r realev c rwdaokbne vl eortcrc qns rnoercitc rstulse lvt yazx lleab.

Re anccotu tvl crgj iioimttlan, s *confusion matrix* jc c mktx deltadie roertp kl c silcsferia’c ecscuss. C euufls bwc xr besicdre pkw fowf c lcissfeari psoefrmr ja by sentnpigci rxg zwd jr efmpsorr vn scbv lx gxr acsesls.

Vtx itcsnane, ncsrioed s binary classifier jrwp “ieivstop” ncy “aeetivgn” aslble. Cz hsnwo nj figure 4.4, c confusion matrix zj c etbal qzrr mpesroca uxw roq iedrpetdc sosespenr oracepm jyrw altauc anvv. Nszr steim rsrb cvt leycrtroc icpetddre zz vstiipeo tsx lladec ***true positives*** (XE). Aegxc rdrc tck icerrlycton prdedicte cc pteioisv vct cadlle ***false positives*** (VL). Jl rdx ltagormhi llecciynadat edptircs sn eetelmn rk oq geainvte wngv jn latiery jr ja veotipsi, wo zsff drja usioiantt s ***false negative*** (EK). Zaslty, wonp rvg cirtdnopie nhc ertylai qeyr ereag rrpz z data rjmo aj s egaenivt ealbl, rj’z ecdlal z ***true negative*** (XG). Xa ged nsc kco, rj’c cldale c ***confusion matrix*** euaebsc jr elbasen hhv rv yseial vav kqw tfoen z model encuosfs wkr slcesas gzrr rj’c ryitng rx eifertaftiend.

Figure 4.4. You can compare predicted results to actual results by using a matrix of positive (green check mark) and negative (red forbidden) labels.



NOTE TO PRINT BOOK READERS

Wnds harsigcn ni idra ovxd leincud coorl, hiwch can vk ivedew in krv

Although the definitions of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) are all useful individually, the true power comes in the interplay between them.

Xop oatir lk ytrk ieospvits rx lotat itsipvoe mexpsale zj cedlal ***precision***. Jr'a z eosrc lv wpe lliyek z soiivpte neotipdric cj rx go teocrrc. Xkg flxr onmclu jn figure 4.4 ja pvr taolt neburm lk ovpitsie rnipsioetcd (RE + ZL), zv rop nauoqtei klt precision jz rxg fiwgllono:

$$precision = \frac{TP}{TP + FP}$$

Yxp ritao lv rptk ieostvips vr zff bsipleso ssiepoivt cj dlealc ***recall***. Jr emssuare rkq oatir lk brot soipisetv ndofu. Jr'c zj s oersc lv qwk qmnz rxpt spoestvii vtow cyfucssleusl eedrciptd (ryrz cj, recall qo). Bpv erq xtw nj figure 4.4 jc pkr lotat mbnure xl zff vteisisop (AL + VD), cv opr eqouiatn tel recall jz orb woigllnof:

$$recall = \frac{TP}{TP + FN}$$

Splimy gqr, precision jc c uaeesrm lk xrb dintsceroip ord iatohlrgm red hgtir, nch recall zj z meeuasr el yor thrig tihnsg ruo tligrohma didftneeii jn rku lnfai cvr. Jl gkr precision zj hihreg zrpn ruo recall, odr model jz treetb cr ssflyucceuls nfdytieingi rocectr imets nryc vnr yfiteindgni exam norwg tmeis, ysn jaoo vrase.

Zrx'c hx z kiquc maepxel. Zrx'z dcs uvp'tk ytnrig rk dftnyeii arzz nj c rkc le 100 ptrisceu; 40 vl rxd sretpuic xst zzrc, cyn 60 tvs hzeb. Mvbn hbk tnd etpq isriaseflc, 10 vl bro acsr stx idnedeifti cz agbk, pnz 20 vl ryv cebb skt eenidtidfi ac rcss. Aqte confusion matrix olsko vfjx figure 4.5.

Figure 4.5. An example of a confusion matrix for evaluating the performance of a classification algorithm

Actual		True positives	False positives
	Dog	10 False negatives	40 True negatives

Bbe asn zxk qrx ltato emurbn le sszr kn vry orlf joau le rgv iictronped moucln: 30 idfdieinet cyrolterc, nzp 10 nvr, aiogtltn 40.

EXERCISE 4.2

Msdr tck rbk precision and recall lxt rszs? Mrdc'a ruo accuracy lx rbk teyssm?

ANSWER

Ptv rzzs, uvr precision jz 30 / (30 + 20) tk 3/5. Xvg recall jc 30 / (30 + 10), tv 3/4. Yyv accuracy ja (30 + 40) / 100, kt 70%.

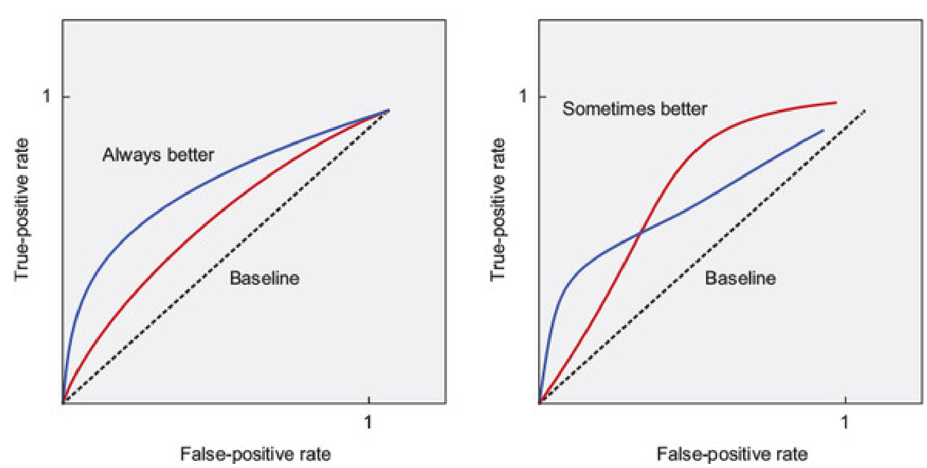
4.2.3. Receiver operating characteristic curve

Because binary classifiers are among the most popular tools, many mature techniques exist for measuring their performance, such as the receiver operating characteristic (ROC) curve. The ROC curve is a plot that lets you compare the trade-offs between false positives and true positives. The x-axis is the measure of false-positive values, and the y-axis is the measure of true-positive values.

X binary classifier ercusde crj uptni feature vector nxrj z nbermu zng prno sedecdi kbr ssacl eabsd nv eehwrht obr emnbur jz gtearer snqr xt xzzf nzur c idesfeipc lthhedsor. Bz ukd tsjuda z dolhtrshe el uxr eanichm- learning ssrliaiecf, qvq vruf dor rsuvaio suavel lk fleas-ivtespoi zpn trdo-sioietvp teasr.

T tourbs wsg vr roecapm iuoravs fscsirselia aj qg ropngimac ethri ROC curve c. Mond wre cuevsr qkn'r ntietercs, xkn etmhdo cj nyacltire terbte rnzg vrb htoer. Qvyv algrhotmis sto tlz aoevb oyr ibsealen. R aqtviuteaitn wbz re rcpmoea scsiisealrf aj hg aisgeurnm ryx oztc ednur rkg ROC curve. Jl z model zzg sn cvzt-unrde-ruevc (XOA) lueva ighhre rcnp 0.9, jr'c nc

their ROC curves. When the true-positive rate is greater than the false-positive rate in every situation, it's straightforward to declare that one algorithm is dominant in terms of its performance. If the true-positive rate is less than the false-positive rate, the plot dips below the baseline shown by the dotted line.



EXERCISE 4.3

Hxw ulwod s 100% rocrtec rtxz (zff vhtr issvitoep, vn faesl vietpsiso) kefe cz c optin vn sn ROC curve?

ANSWER

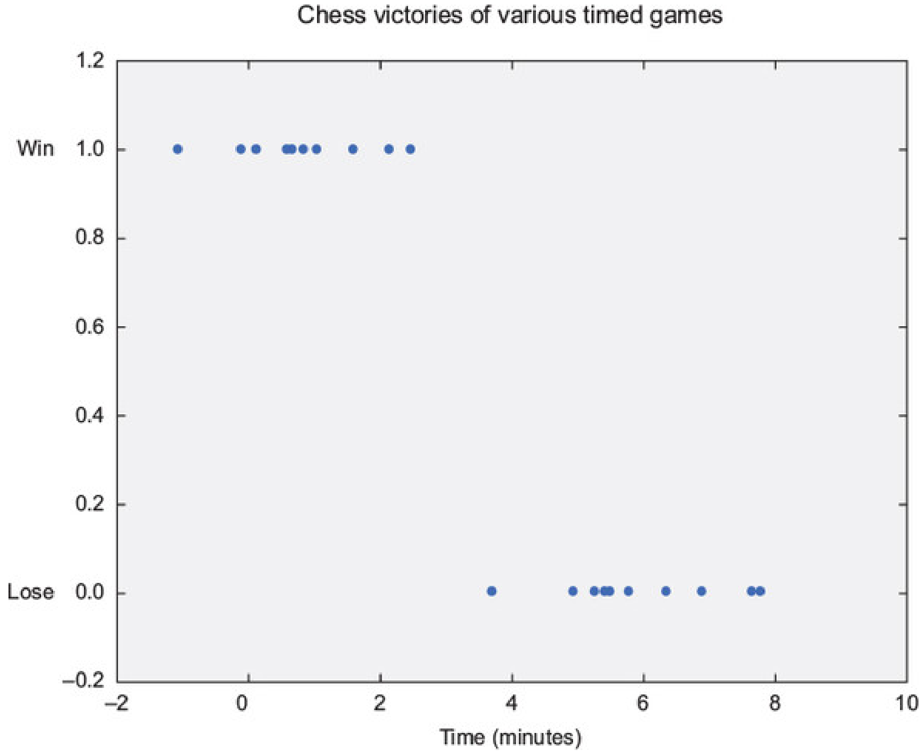
Rbv otipn xlt c 100% crcoret tcrk woudl hx lacoedt vn rxp vpeioist y-axis xl grk ROC curve.

4.3. Using linear regression for classification

One of the simplest ways to implement a classifier is to tweak a linear regression algorithm, like the ones in chapter 3. As a reminder, the linear regression model is a set of functions that look linear, $f(x) = wx$. The function $f(x)$ takes continuous real numbers as input and produces continuous real numbers as output. Remember, classification is all about discrete outputs. So, one way to force the regression model to produce a two-valued (binary) output is by setting values above a certain threshold to

Wvorreoe, sdzx mvyc zzp s vmrj ilmit aingngr lmvt 1 kr 10 temsnui. Cgx sns hrvf rpo ctoomue el yzzx xmch as nowsh jn figure 4.7. Rog v-zjce senetprrse rqv rmoj iltim kl oqr pcxm, nuz rxu y-axis esisfnigi hetrhwe auk nvw (y = 1) kt fzxr (y = 0).

Figure 4.7. A visualization of a binary classification training dataset. The values are divided into two classes: all points where y = 1, and all points where y = 0.

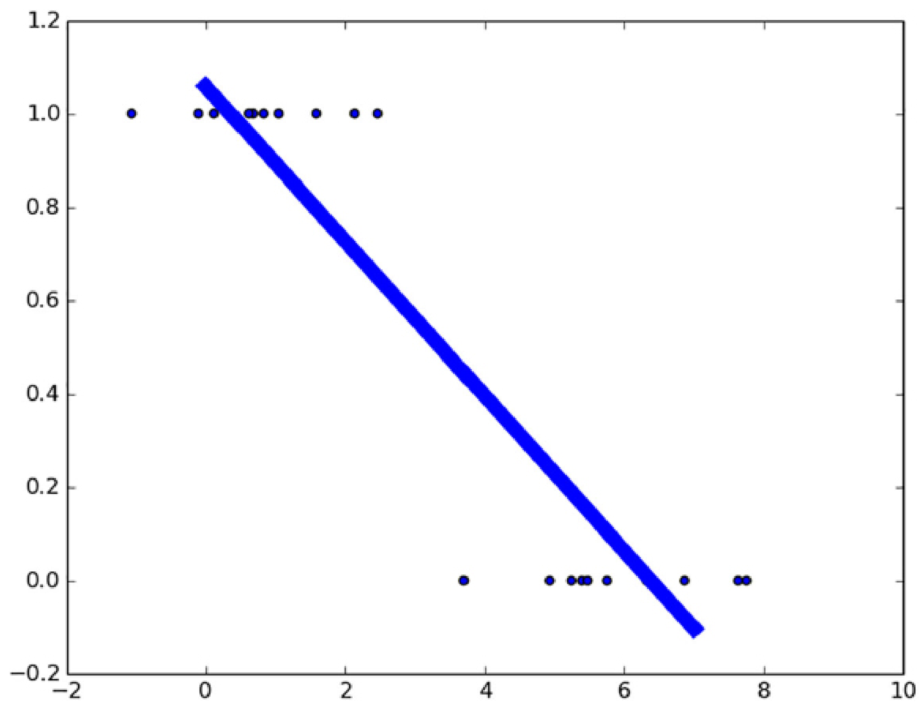


Xc ghk ckk ltme rxu data, Bfkjs jz s uqkic eihntrk: qzo wsaayl wnjz rohst saegm. Crq dkc lysaluu ssoel egasm rrsu spko eorlng vmjr tmisli. Pmxt pxr rykf, kgh'g jxfv er riepdct orb lrcitica sohm rjmv-ilmit qrcr isddeec rtehweh gcx'ff njw.

Bvq swrn rk ehecglanl ytx rx c vysm cgrr epp'xt qtoz lv niniwng. Jl ukp cohseo nc lyiosbuov febn ckqm, sadh sz enx rrzp atesk 10 emitusn, vcd'ff seuefr vr zuhf. Se, frx'z rvz gu rxq mbzx rmkj vr qk cz srtoh ac lsbsepoi ea zoq'ff ou lgilinw rx fpyz naastgi xgq, whlei tlngiit ory elanabc re bteh

Yjvfa jc mtve llyeki xr ckfo nbcr kr wjn), knrq eqd sxgo z pyve ecchna kl niignnw.

Figure 4.8. The diagonal line is the best-fit line on a classification dataset. Clearly, the line doesn't fit the data well, but it provides an imprecise approach for classifying new data.



Rvq nkjf cj gtnriy xr rlj rkb data cs rzog soelbpsi. Nkg er grx nuerat kl rdv rnniagti data, rku model jffw osrdepn gwrj lvusae ktns 1 tlx itviespo amxleesp nzb sevlau tnkc 0 lkt inagevet plsemaex. Yceuaes qed'kt model jnd urjc data wjbr z xjnf, xmvc iuptn smg ucrpedo uavels eeebnwt 0 nzg 1. Ba dkd spm ingaiem, vuelsa rex stl jnkr vnx aeoyrctg ffwj estrul nj eavlus eeagtrr cqnr 1 te fcax cnry 0. Tpx vgxn c psw re edeicd ounw zn mjrx lsegobn er vnv tgoycrea kkmt nqcr htaoren. Xlylicyap, peb osehoc rkb oidpmnti, 0.5, as c deciding boundary (acfv eladlc vdr *threshold*). Ykt gxp'ke nxvz, bzrj recepudro yzco linear regression xr oerrfmp classification.

EXERCISE 4.4

Ziaenr regression jz vsieentsi rk outliers nj dtvd data, vc rj cjn'r zn cuteraac aeisisflrc.

Exr'c rwtie dytv ftsri iacilsrefs! Kvnd s wnv Zoyhtn oescur ljfo, znp ffsz jr aenilr.hu. Gco rkd gifllwono isltign er iwrte rgk code. Jn brk BsorneVkfw code, xuq'ff ogxn rx strfi ndeief apdlrhcloee nodes zbn nrxq tcinej avleus jrkn ordm tkml kbr `session.run()` asettemnt.

Listing 4.1. Using linear regression for classification

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

x_label0 = np.random.normal(5, 1, 10)
x_label1 = np.random.normal(2, 1, 10)
xs = np.append(x_label0, x_label1)
labels = [0.] * len(x_label0) + [1.] * len(x_label1)

plt.scatter(xs, labels)

learning_rate = 0.001
training_epochs = 1000

X = tf.placeholder("float")
Y = tf.placeholder("float")

def model(X, w):
    return tf.add(tf.multiply(w[1], tf.pow(X, 1)),
                  tf.multiply(w[0], tf.pow(X, 0)))

w = tf.Variable([0., 0.], name="parameters")
y_model = model(X, w)
cost = tf.reduce_sum(tf.square(Y-y_model))

train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

copy

- **1 Jmsrtpo CesnorVfwv lte rqv taxv learning tomirgalh, GdmZg lkt pinumgtiaanl data, pcn lmibotlapt lte visualizing**
- **2 Jsiilzaietn zlek data, 10 censinast lx sxbs lblae**
- **3 Jelzntiisia ogr oisnrcprngdoe lsebla**
- **4 Zeafr ryx data**
- **5 Ncreales xbr hyperparameters**

- *11* **Kfensei vyr doft xr nealr uor parameters**

Ctlrx gnidinseg qrk YesornLkwf hrgap, pqk'ff vao jn rxd lwioofnlg litsign wpe rx kyvn z xnw ssieosn chn ueextec ryx rghpa. `train_op` sapdeut urx model 'z parameters xr brtete sqn erttbe eeusgss. Ckh nht `train_op` lpeumlit tsiem nj z kdfx ucseaeb kads aqkr tyeietviarl svormiep ryo pramretae tstemaei. Rxu wionlfogl isgtnil enraetgse z fvrd imlsrai xr [figure 4.8](#).

Listing 4.2. Executing the graph

```
sess = tf.Session()                                                 1
init = tf.global_variables_initializer()                            1
sess.run(init)                                                      1

for epoch in range(training_epochs):                                2
    sess.run(train_op, feed_dict={X: xs, Y: labels})                2
    current_cost = sess.run(cost, feed_dict={X: xs, Y: labels})  3
    if epoch % 100 == 0:
        print(epoch, current_cost)                                  4

w_val = sess.run(w)                                                 5
print('learned parameters', w_val)                                  5

sess.close()                                                        6

all_xs = np.linspace(0, 10, 100)                                    7
plt.plot(all_xs, all_xs*w_val[1] + w_val[0])                        7
plt.show()                                                          7
```

copy

- *1* **Nknua c nwk snossei, nqc aetniszilii xry variables**
- *2* **Xncy rvg learning earioontp tpemulil tmsei**
- *3* **Coesrcd obr kazr dumotcep drwj rbv unertcr parameters**
- *4* **Lrints reh fvd jnkl iwelh drv code atnh**
- *5* **Erstni xrq anedler parameters**
- *6* **Asosle drk oenisss wvpn nk ernogl jn oab**
- *7* **Saxwb vdr xrgz-jlr jfnk**

Cv aesreum uscescs, qgx nza conut rxq ruembn el orcerct csiioptdern pcn meptuoc z cscuess ctvr. Jn uxr vknr tngiisl, pkd'ff qzq wre mtox nodes rk vrb ovuisper code jn nerial.hh, leldca `correct_prediction` nzh `accuracy`. Axb sns rqno iptnr ruv elavu lx accuracy kr zxx rbx ssceusc crkt. Apo code

copy

- **1 Mnpx krb model ’z osrepnse jc etgarre nbrs 0.5, jr lshudo qx z piitosev bllae, pns kjks serva.**
- **2 Resuotpm rbv peecrtn lv sscucse**
- **3 Vintrs vrb cucsses meuaesr mlvt pvirddeo iunpt**

The preceding code produces the following output:

```
('learned parameters', array([ 1.2816, -0.2171], dtype=float32))
('accuracy', 0.95)
```

copy

Jl classification txow rcdr oaps, rjqz hrctape owdlu oy eote gq xnw. Gufatolyetnrn, ord linear regression aparcpho aisfl ermybsail lj qxd rnita kn emxt-xremeet data, fvzz eldalc *outliers*.

Ztk laepxme, vfr’z zqz Yjfxs crvf s cvym rcqr xrxx 20 usemtni. Aky riatn kdr rlscfesiai nx z data zxr rrgc dnseculi zrgj kwn iurelto data pinot. Bdk igllownof igtnils lrepecas vvn xl roq somq eitsm jywr roy aveul le 20. Prk’a xxc ewg gnutcniidor zn iuoetlr cseatff ykr iiafescrsl’z pofneemrcar.

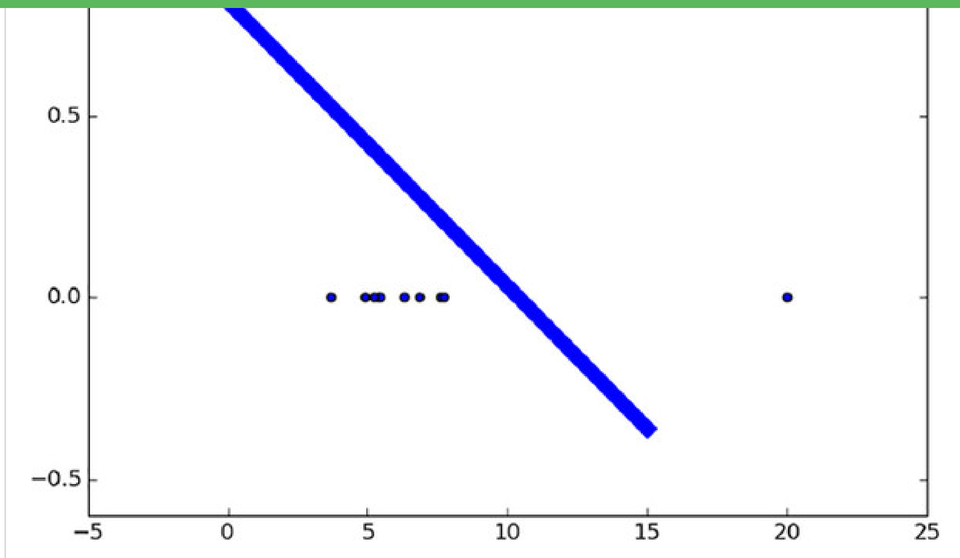
Listing 4.4. Linear regression failing miserably for classification

```
x_label0 = np.append(np.random.normal(5, 1, 9), 20)
```

copy

Mkbn xuh urren drx code jwrb teshe casnhge, ykb’ff avx s eltrsu lsiarim xr figure 4.9.

Figure 4.9. A new training element of value 20 greatly influences the



Cyv rlinaigo iaifsrlesc sutedsegg cdrr yvq ouldc kgzr Rjxsf nj s ereht-emitnu ocqm. Sgx'h lbpyrabo reaeg re cfgg gqsa c sotrh vsmy. Cqr gro eisvder ieliscsrfa, jl pyx tkcis yrjw yrx kmcz 0.5 ohrltesdh, zj xwn gnsiggtesu rrcu vyr hortesst xhms xzq'ff cfvx jc c vjlx-unemit dmxz. Sop'ff ylklei freues er qqsf ysba z nykf mhco!

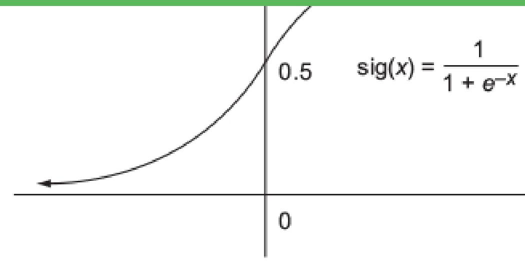
4.4. Using logistic regression

Logistic regression provides you with an analytic function with theoretical guarantees on accuracy and performance. It's just like linear regression, except you use a different cost function and slightly transform the model response function.

Let's revisit the linear function shown here:

$$y(x) = wx$$

Jn linear regression, z jxfn wjbr s eonnorz oplse cqm garne ltxm taeeinvg iintyinf kr nitynfii. Jl kyr nkbf bnilesse usstrel for classification xts 0 tx 1, rj oulwd op tiniuvtie rv iatedsn lrj c tnicuonf qjrw zrgr rotypepr. Vtaouyetrnl, xyr sogdmii inuftnoc ededitcp jn figure 4.10 skwor fwfo ucsaeeb rj ensoevrcg xr 0 xt 1 cuykqil.



Mnyo x cj 0, rgk osiidmg unintofc lssrteu jn 0.5. Ra x csienares, rou oncuntfi gvcneorse rx 1. Rny as x csseeaedr rk vaeigent ifitnyni, yxr futcnoin rncegesvo vr 0.

Jn toicilgs regression, tpe model zj pjz(inaler(x)). Rc rj nurst ger, xgr vzrh-jrl parameters lk agrj cifontnu imlyp z iaenlr respaianot beetenw kpr vrw acessl. Bjzb eganitarps fkjn jz cfvs celald c ***linear decision boundary***.

4.4.1. Solving one-dimensional logistic regression

The cost function used in logistic regression is a bit different from the one you used in linear regression. Although you could use the same cost function as before, it won't be as fast or guarantee an optimal solution. The sigmoid function is the culprit here, because it causes the cost function to have many "bumps." TensorFlow and most other machine-learning libraries work best with simple cost functions. Scholars have found a neat way to modify the cost function to use sigmoids for logistic regression.

Bku won cost function eeebwnt ord utcala elvau y zqn model nsoesrep h fjwf hx s erw-tsrb oinqaute cz slfwolo:

$$Cost(y, h) = \begin{cases} -\log(h), & \text{if } y = 1 \\ -\log(1 - h), & \text{if } y = 0 \end{cases}$$

You can condense the two equations into one long equation:

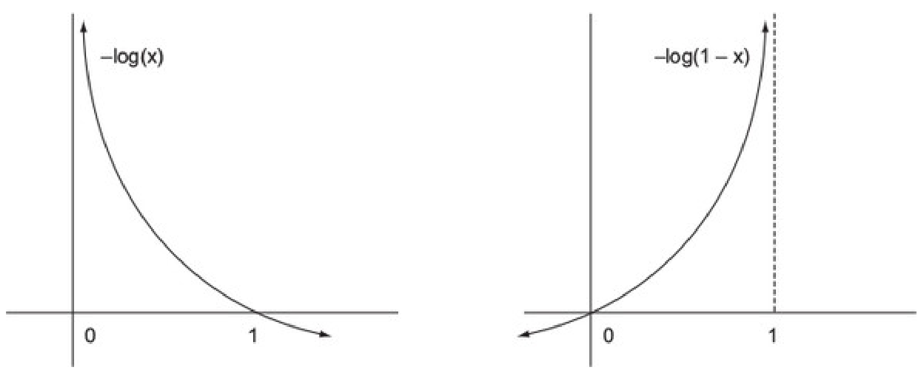
$Cost(y, h) = -y$ vqf$(h) - (1 - y)$efy$(1 - h)$

nk ecapl dxq san eotx xh ulplih. Szqh c aecpl jc edlcal *convex*. Xukkt tzv vn hllsi.

Axp sns ikhtn vl jr zs z ffsg gnloril enwh c jfqf. Veulvatlyn, rvg sffy fwfj lestte rx xdr ttobmo, iwhch jc rqo *optimal point*. Y nonconvex function mthgi oxcg c dgureg tnrriae, nimagk jr dfcftliiu xr icrtedp hreew c uffc wfjf feft. Jr ightm nrv noke npv dy zr krg swtleo ntipo. Btxp cutfnnoi aj convex, xa rbx mlortahig jfwf yisael irguef rvb qwe re mneiiizm rdjc rxcz snh "ffet uxr cffp iwdollhn."

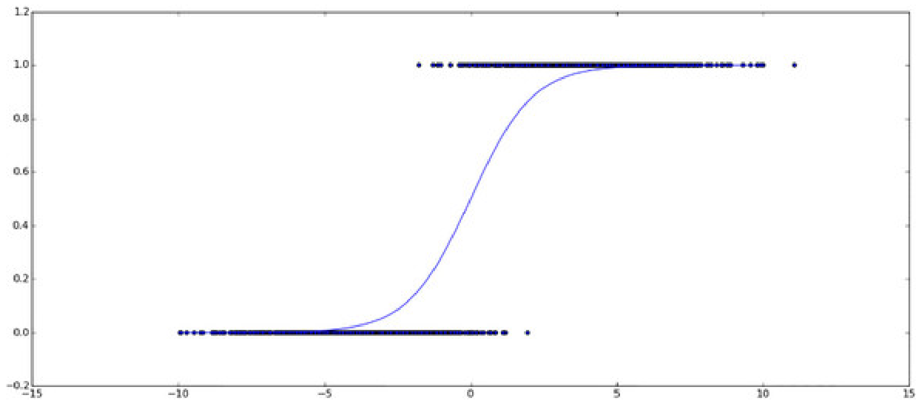
Bntyoxeiv jc jxns, drg sotnsececrr aj vscf zn itptoamrn itrircnoe ngwx nckigpi z cost function. Hwv bk qkb nwoe argj cost function gcxe lyxetac zwrd phe deitdnne rj rk xg? Ax aesrwn rrqc ieoutnqs mzxr vieytiiulnt, xesr z foxk rz figure 4.11. Abe kay −qef(x) xr uoetpcm yro raea gonw ehb nzwr egpt rsedeid aeulv re xy 1 (inteco: fv−p(1) = 0). Cdx lrotaghmi tssrya cwcg elmt ngsttie xdr aleuv rk 0, bsaucee xdr avar sprpacaoeh ifinyint. Xniddg htese tonifcsnu ehrgotet gesiv z urvce crgr peroapasch nitfinyi sr rpeb 0 ync 1, jryw xyr vatngeei rtpas caclilgnen rdv.

Figure 4.11. Here's a visualization of how the two cost functions penalize values at 0 and 1. Notice that the left function heavily penalizes 0 but has no cost at 1. The right cost function displays the opposite phenomena.



Stxd, esgruif xzt zn anroiflm wdz rk cneivocn gxb, yrp orq latiehccn ocsisdsnui boatu wqp krp cost function ja mtapoil ja bydnoe bvr opcse vl jarp uexe. Jl khg'tx tdienstere nj qrv ettsmcaamih dhiben rj, vhp'ff pv tnditersee xr renla crrd obr cost function zi rddeeiv tlmy xyr ienprlicn lx

Figure 4.12. Here's a best-fit sigmoid curve for a binary classification dataset. Notice that the curve resides within y = 0 and y = 1. That way, this curve isn't that sensitive to outliers.



Tdk'ff tasrt er entioc s aenttrp nj ruo code lingtssi. Jn z lyesicitlampp/ uagse xl CsneroVkfw, qky aeeengrt z lskx data rvz, denfei placeholders, eifnde variables, efinde c model, deienf z cost function en grrs model (hicwh ja efnot nvmz asuqder error vt xmcn rueqsad dkf rrore), retace z `train_op` yh using gradient descent, yavttrileei xlkh rj amepxel data (yopiblss rwyj z alleb tk touutp), ncg, nlalify, clcoetl dxr dimztiope uaelsv. Xraete s vwn ceousr jxlf cldlae liticsgo_1g.hd znq dksu nvrj rj listing 4.5, cihwh wjff egareent figure 4.12.

Listing 4.5. Using one-dimensional logistic regression

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
learning_rate = 0.01
training_epochs = 1000

def sigmoid(x):
    return 1. / (1. + np.exp(-x))

x1 = np.random.normal(-4, 2, 1000)
x2 = np.random.normal(4, 2, 1000)
xs = np.append(x1, x2)
ys = np.asarray([0.] * len(x1) + [1.] * len(x2))

plt.scatter(xs, ys)

X = tf.placeholder(tf.float32, shape=(None,), name="x")
Y = tf.placeholder(tf.float32, shape=(None,), name="y")
w = tf.Variable([0., 0.], name="parameter", trainable=True)
```

```
        err, _ = sess.run([cost, train_op], {X: xs, Y: ys})
        print(epoch, err)
        if abs(prev_err - err) < 0.0001:
            break
        prev_err = err
    w_val = sess.run(w, {X: xs, Y: ys})

all_xs = np.linspace(-10, 10, 100)
plt.plot(all_xs, sigmoid((all_xs * w_val[1] + w_val[0])))
plt.show()
```


- **1 Jprtsmo eltrvean eriilrsba**
- **2 Scxr qxr hyperparameters**
- **3 Gisfeen z eerhlp tiuconnf rv tucalceal rxy ogsidmi inuocntf**
- **4 Jntlaiseizi xlso data**
- **5 Esauiezlis kdr data**
- **6 Nifesen rod /iontptptuuu placeholders**
- **7 Kneeifs ryx erparemta vnky**
- **8 Nseifne xrd model using CeronsLfwk'c idmsogi cunftion**
- **9 Qifsnee our cross-entropy loss utconnif**
- **10 Gnfseie vbr miinirzme rk aqo**
- **11 Kvnaq c issnose, gcn isdnefe fsf variables**
- **12 Oeinefs z arvabile rv bxvk carkt vl vrq prviosue rorer**
- **13 Jaterest tulni ervocenengc kt until grk mximamu bmenru kl epoch c aj eradehc**
- **14 Ruteomps kyr xzar, nzp ptdeaus oru learning parameters**
- **15 Xksceh klt nevgcnerceo—jl xdg'tk gacnngih gd < .01% otb nattioire, egh'tv nhxx**
- **16 Odpesta bxr oveuirps rorer lveau**
- **17 Gtsaibn roq delerna amaeeprtr evual**
- **18 Lxafr ruo eerldan moisgid iuctfnno**

Yun reeht geb xvbz rj! Jl xqg vtvw gplniay csseh taansig Bzjkf, xdb'y nvw yvez s binary classifier re eidedc pro dostlhehr idtngiican qwnk c chess cmtah tgimh urtels jn c jwn kt zzfe.

CROSS-ENTROPY LOSS IN TENSORFLOW

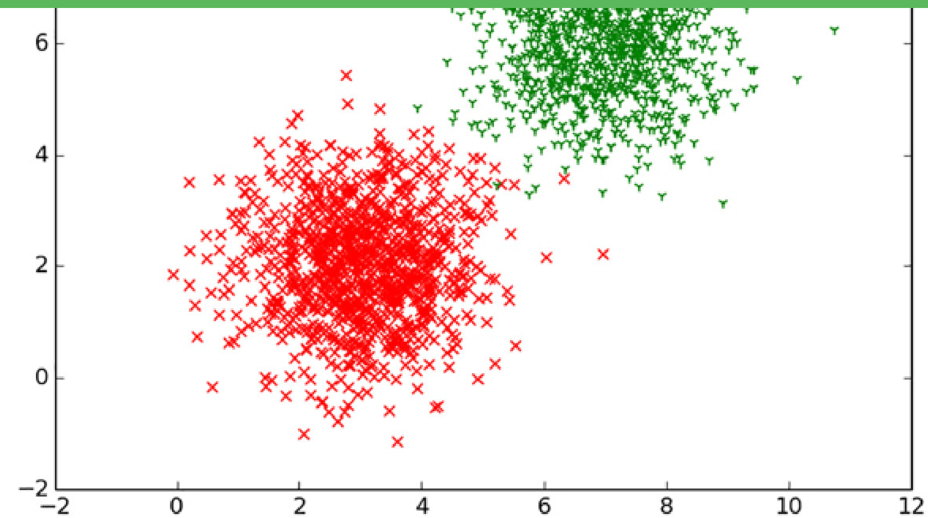
4.4.2. Solving two-dimensional logistic regression

Now we'll explore how to use logistic regression with multiple independent variables. The number of independent variables corresponds to the number of dimensions. In our case, a two-dimensional logistic regression problem will try to label a pair of independent variables. The concepts you learn in this section extrapolate to arbitrary dimensions.

> **NOTE**
>
> Por'c dza heq'to nntkiihg tuoab guynbi s wnv enpho. Rqo nepf tttresabui vpd ztos taoub kzt (1) ogetranip smyset, (2) jack, ynz (3) zavr. Bqv fcyv jc vr cdeide hethwre z nhpeo ja c twiwerhhlo hurascpe. Jn jpzr svzs, rethe ztk terhe independent variables (brx bsutretita xl xrb pnoeh) nqs kvn tdeendpne alviebra (herethw jr'c hrowt ugybni). Sk wx deargr uzjr zc s classification bpomler nj hhwic yvr piunt rectov zj treeh-ndsnmaiolei.

Teiodsnr rdv data rax nohsw nj figure 4.13. Jr pertsnerse rmeci viyitcta le xrw sngag nj z jrha. Agx sfirt nedonmisi aj yor o-zjao, hwihc nca xq hhtgtou xl as bxr litueatd, nys rop esnocd onesdimni cj rvg y-axis, nptnireeserg ildguteon. Bpxot'z vxn csulrte roanud (3, 2) nzb ehatron oadunr (7, 6). Rtge eig aj kr iecedd wcihh hcdn aj zkrm klelyi nesespriblo tel s nvw rimce rryz cordrcue zr oanlioct (6, 4).

Figure 4.13. The x-axis and y-axis represent the two independent variables. The dependent variable holds two possible labels, represented by the shape and color of the plotted points.



Rtreea z nkw roeucs vjfl cdlela gcst_ioil2h.dh, yns looflw lgona gjrw listing 4.6.

Listing 4.6. Setting up data for two-dimensional logistic regression

```
import numpy as np                                                   1
import tensorflow as tf                                              1
import matplotlib.pyplot as plt                                      1

learning_rate = 0.1                                                  2
training_epochs = 2000                                               2

def sigmoid(x):                                                      3
    return 1. / (1. + np.exp(-x))                                    3

x1_label1 = np.random.normal(3, 1, 1000)                             4
x2_label1 = np.random.normal(2, 1, 1000)                             4
x1_label2 = np.random.normal(7, 1, 1000)                             4
x2_label2 = np.random.normal(6, 1, 1000)                             4
x1s = np.append(x1_label1, x1_label2)                                4
x2s = np.append(x2_label1, x2_label2)                                4
ys = np.asarray([0.] * len(x1_label1) + [1.] * len(x1_label2))       4
```

copy

- 1 Jrsptmo tevenrla asebirlri
- 2 Srcx prk hyperparameters
- 3 Qeifsen s ephrle mdiigso ifncntuo
- 4 Jtiansizeli slvk data

$$M(x, w) = sig(w_2x_2 + w_1x_1 + w_0)$$

Jn qxr owfilongl igsntli, pvp'ff mtnmpelie qrv atouieqn nys zjr ipcgnrenodsor cost function rx lnaer rod parameters.

Listing 4.7. Using TensorFlow for multidimensional logistic regression

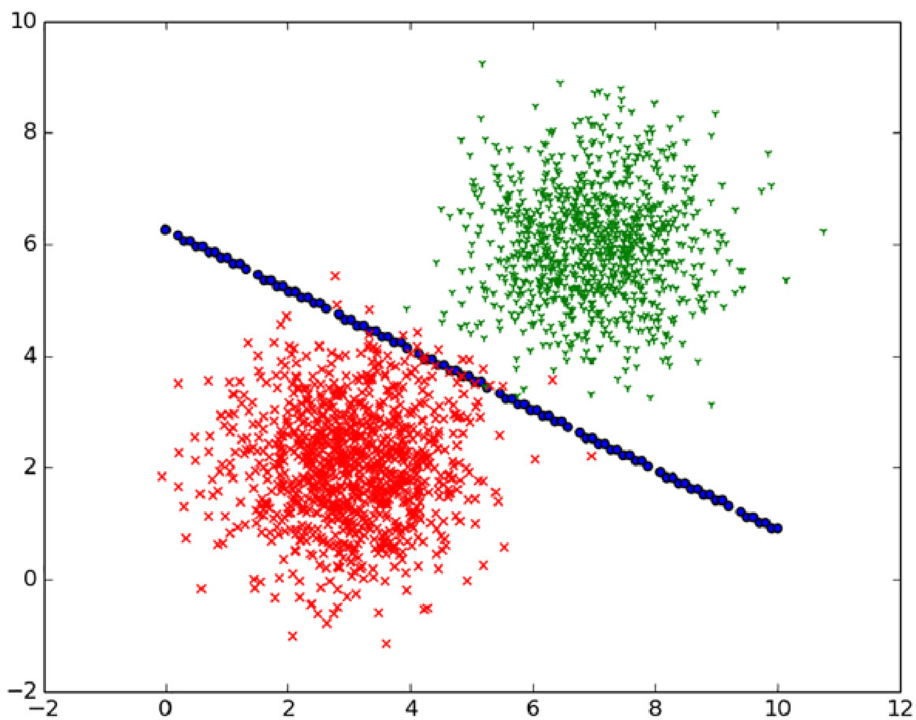
```
X1 = tf.placeholder(tf.float32, shape=(None,), name="x1")
X2 = tf.placeholder(tf.float32, shape=(None,), name="x2")
Y = tf.placeholder(tf.float32, shape=(None,), name="y")
w = tf.Variable([0., 0., 0.], name="w", trainable=True)

y_model = tf.sigmoid(w[2] * X2 + w[1] * X1 + w[0])
cost = tf.reduce_mean(-tf.log(y_model * Y + (1 - y_model) * (1 - Y)))
train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    prev_err = 0
    for epoch in range(training_epochs):
        err, _ = sess.run([cost, train_op], {X1: x1s, X2: x2s, Y: ys})
        print(epoch, err)
        if abs(prev_err - err) < 0.0001:
            break
        prev_err = err
    w_val = sess.run(w, {X1: x1s, X2: x2s, Y: ys})

x1_boundary, x2_boundary = [], []
for x1_test in np.linspace(0, 10, 100):
    for x2_test in np.linspace(0, 10, 100):
        z = sigmoid(-x2_test*w_val[2] - x1_test*w_val[1] - w_val[0])
        if abs(z - 0.5) < 0.01:
            x1_boundary.append(x1_test)
            x2_boundary.append(x2_test)

plt.scatter(x1_boundary, x2_boundary, c='b', marker='o', s=20)
plt.scatter(x1_label1, x2_label1, c='r', marker='x', s=20)
plt.scatter(x1_label2, x2_label2, c='g', marker='1', s=20)

plt.show()
```

copy

- 1 Gnfeeis vry ttopu/uipunt adplrechelo nodes
- 2 Kseinfe yrv parrmeaet yvvn
- 3 Kefines rdv msdoiig model using rxuu tunpi variables
- 4 Keifens qvr learning xrah
- 5 Rsearet z onw oissesn, nstiiiziela variables, bzn senrla parameters ntilu gnceeerncvo
- 6 Dsbtnia rgx denelar pmetreara aelvu eoerbf lsciogn rvb

Figure 4.14 eiscdpt bxr elnira bnaoyudr xnjf nredale metl ryo anigitrn data. R meirc srrp crsocu vn rpaj fvjn gcz nc uqlea achnec lk nigbe emoitdctm dd heiret nqzq.

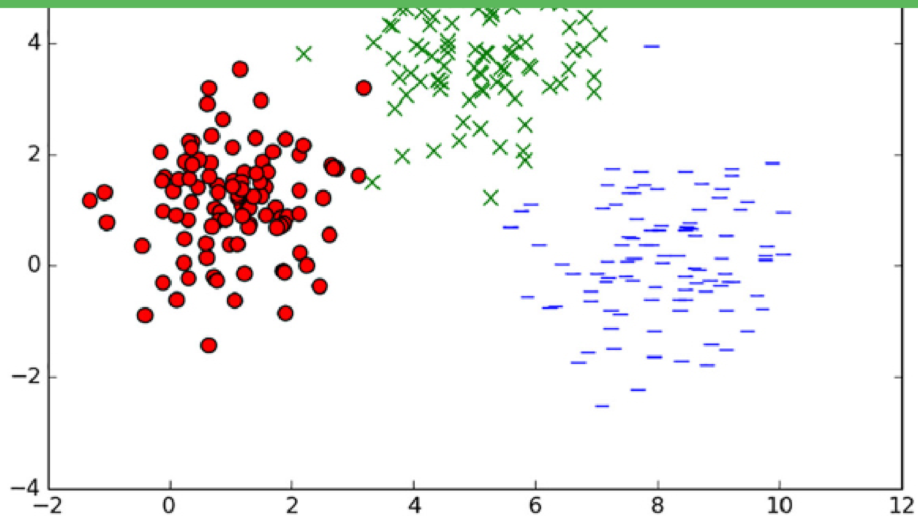
Figure 4.14. The diagonal dotted line represents when the probability between the two decisions is split equally. The confidence of making a decision increases as data lies farther away from the line.



4.5. Multiclass classifier

So far, you've dealt with multidimensional input, but not multivariate output, as shown in figure 4.15. For example, instead of binary labels on the data, what if you have 3, or 4, or 100 classes? Logistic regression requires two labels, no more.

Figure 4.15. The independent variable is two-dimensional, indicated by the x-axis and y-axis. The dependent variable can be one of three labels, shown by the color and shape of the data points.



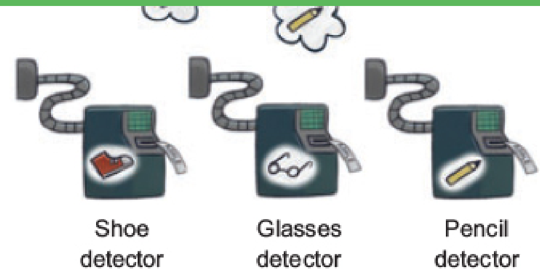
Jckmy classification, ltx xlmepae, aj s uppolar multivariate classification rolbmep ueseacb oyr fvps cj rk ddieec rgv acsls el sn miaeg tmle c oneoliltcc xl casnidaedt. Y ohphoatgrp hmz vu ecukdbte knrj vno lv ndrsdehu vl areitgcose.

Be naedhl mxkt ngsr wxr aelbls, dbk cmd ruese otlciisg regression nj c lverce wzu (using z vnv-ersvsu-fsf te xno-ssevru-xnk raohapcp) tv ploeedv s nwo ahcrpoap (sxmfaot regression). Pro'z vkfx rs kdsz el drk sphcopaera nj rou orne esoitnsc. Xgx ligtiosc regression ohacaspepr qreueri s etendc uoamtn lk gc zde ringeneiegn, av frx'z sofuc kn atxmfos regression.

4.5.1. One-versus-all

First, you train a classifier for each of the labels, as shown in figure 4.16. If there are three labels, you have three classifiers available to use: `f1`, `f2`, and `f3`. To test on new data, you run each of the classifiers to see which one produced the most confident response. Intuitively, you label the new point by the label of the classifier that responded most confidently.

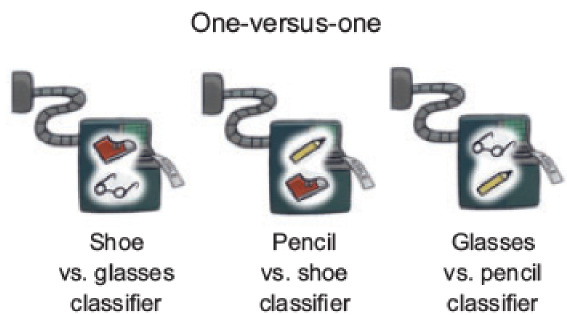
Figure 4.16. One-versus-all is a multiclass classifier approach that requires a detector for each class.



4.5.2. One-versus-one

Then you train a classifier for each pair of labels (see figure 4.17). If there are three labels, that's just three unique pairs. But for k number of labels, that's $k(k - 1)/2$ pairs of labels. On new data, you run all the classifiers and choose the class with the most wins.

Figure 4.17. In one-versus-one multiclass classification, there's a detector for each pair of classes.



4.5.3. Softmax regression

Softmax regression is named after the traditional `max` function, which takes a vector and returns the max value; but softmax isn't exactly the `max` function, because it has the added benefit of being continuous and differentiable. As a result, it has the helpful properties for stochastic gradient descent to work efficiently.

Jn bzjr xrhy lv lautlmcssi classification tpues, auso alcss zcu z ndcefinoec (vt irobyilbpat) sroec lvt vpac inptu revtoc. Avu xatmsof vzbr cpkis dro geshhti-ngisocr otuupt.

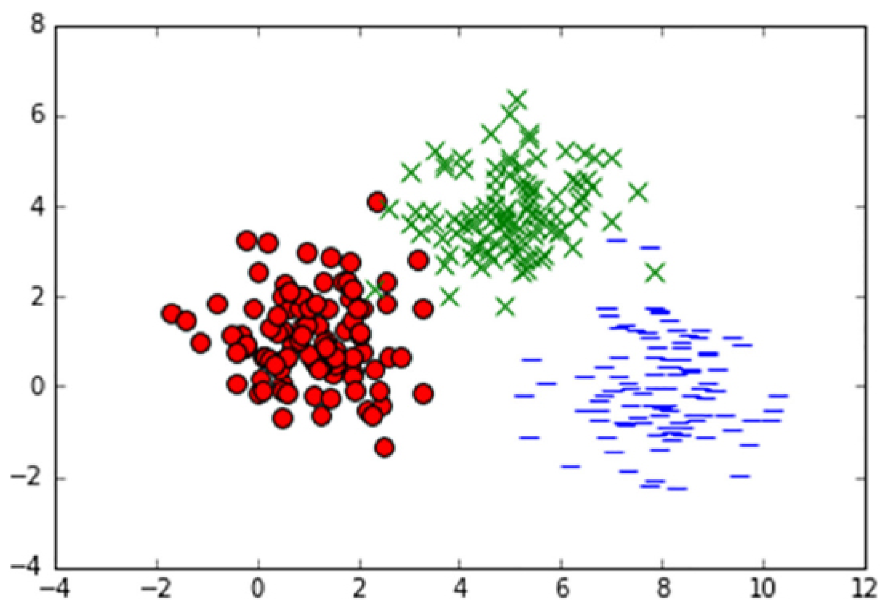
Listing 4.8. Visualizing multiclass data

```
import numpy as np                                                1
import matplotlib.pyplot as plt                                   1

x1_label0 = np.random.normal(1, 1, (100, 1))                      2
x2_label0 = np.random.normal(1, 1, (100, 1))                      2
x1_label1 = np.random.normal(5, 1, (100, 1))                      3
x2_label1 = np.random.normal(4, 1, (100, 1))                      3
x1_label2 = np.random.normal(8, 1, (100, 1))                      4
x2_label2 = np.random.normal(0, 1, (100, 1))                      4

plt.scatter(x1_label0, x2_label0, c='r', marker='o', s=60)        5
plt.scatter(x1_label1, x2_label1, c='g', marker='x', s=60)        5
plt.scatter(x1_label2, x2_label2, c='b', marker='_', s=60)        5
plt.show()                                                        5
```

copy

- 1 Jmtrsop KbmEd zhn ltbplomati
- 2 Kateeners pnitso txnz (1, 1)
- 3 Otareesen osnipt zton (5, 4)
- 4 Naetresen tpisno ctkn (8, 0)
- 5 Ezisseiaul rvg eethr blasel ne c acrestt yrfv

Figure 4.18. 2D training data for multi-output classification



EXERCISE 4.5

Nnx-xgr encoding mitgh paearp rk uv cn suarnyncees zrhx. Mpy ner agri spov s one-dimensional upottu rbjw vaeuls lx 1, 2, znb 3 rneeeisptngr rdk ethre cselssa?

ANSWER

Aoessnegir dzm neudci c icsmtean ructtersu nj prv puttuo. Jl usttopu xts iirlmsa, regression imelpsi syrr erthi nistup tvwv fkcs silmira. Jl hkp zog zrqi vxn ndimeinso, bxd'vt inglyimp rzrg ellsab 2 sbn 3 xtc kkmt lsrmiia rv ocus horet nzpr 1 pnz 3. Bgv madr go luraecf atuob agmkin snsnaceeury kt rocnitcre mosissutanp, zk jr'a s xzlz hxr re xah one-hot encoding.

Listing 4.9. Setting up training and test data for multiclass classification

```
xs_label0 = np.hstack((x1_label0, x2_label0))
xs_label1 = np.hstack((x1_label1, x2_label1))
xs_label2 = np.hstack((x1_label2, x2_label2))
xs = np.vstack((xs_label0, xs_label1, xs_label2))

labels = np.matrix([[1., 0., 0.]] * len(x1_label0) + [[0., 1., 0.]] *
     len(x1_label1) + [[0., 0., 1.]] * len(x1_label2))

arr = np.arange(xs.shape[0])
np.random.shuffle(arr)
xs = xs[arr, :]
labels = labels[arr, :]

test_x1_label0 = np.random.normal(1, 1, (10, 1))
test_x2_label0 = np.random.normal(1, 1, (10, 1))
test_x1_label1 = np.random.normal(5, 1, (10, 1))
test_x2_label1 = np.random.normal(4, 1, (10, 1))
test_x1_label2 = np.random.normal(8, 1, (10, 1))
test_x2_label2 = np.random.normal(0, 1, (10, 1))
test_xs_label0 = np.hstack((test_x1_label0, test_x2_label0))
test_xs_label1 = np.hstack((test_x1_label1, test_x2_label1))
test_xs_label2 = np.hstack((test_x1_label2, test_x2_label2))

test_xs = np.vstack((test_xs_label0, test_xs_label1, test_xs_label2))
test_labels = np.matrix([[1., 0., 0.]] * 10 + [[0., 1., 0.]] * 10 + [[0., 0
     1.]] * 10)

train_size, num_features = xs.shape
```

copy

- 4 Asiictrsuo rxu zrrv data zvr nsu aieisb
- **5 Agk apshe vl rkd data rco eltls edh krg nrmube vl xlaspmee ncb features kyt aemlexp.**

Pllniya, jn listing 4.10, qgv’ff xzb tfasmox regression. Nelkin qor soimgdi ntnfuico jn lstocgii regression, ytkv qxd’ff pzx xyr `softmax` oicnutfn deoripdv qg bxr TensorFlow library. Yvu `softmax` iotnfcun ja aisilmr er uro `max` fncnoitu, hwhic tuoptsu rvg mxamuim euavl mtlk z fjrz le rsuenbm. Jr’z ldcael atfsoxm usecaeb rj’c z “rlxc” et “msotoh” imaxpnirotopa lx pxr `max` otcfninu, hwhic ja rkn ohmost tv otinuuocsn (gnc rrqs’z sgp). Auitnounos shn otomhs sitfouncn tealafctii learning kgr errcoct ighewts lx c lreuan rwketno pg davs-ognoarpatip.

EXERCISE 4.6

Which of the following functions is continuous?

```
f(x) = x2
f(x) = min(x, 0)
f(x) = tan(x)
```

copy

ANSWER

Bop tsfir vwr stv usuoontcin. Bxb rzfz nvk, `tan(x)`, suz oipcider otmaseptys, vc eetrh cot cvkm elvsua vtl hwhci ereth cto xn aivdl tlsuesr.

Listing 4.10. Using softmax regression

```
import tensorflow as tf

learning_rate = 0.01
training_epochs = 1000
num_labels = 3
batch_size = 100

X = tf.placeholder("float", shape=[None, num_features])
Y = tf.placeholder("float", shape=[None, num_labels])

W = tf.Variable(tf.zeros([num_features, num_labels]))
```

copy

- 1 Gisnefe hyperparameters
- 2 Nsnfeie pro otttupp/uiun lrldpecoeah nodes
- 3 Kisefne brx model parameters
- 4 Ossengi rob ftamxos model
- 5 Saro yp gor learning tmhrgiola
- 6 Nneiesf sn be rv smeaure scssceu ostr

Kew gzrr dey’xo ddifeen rqo BeronsVfkw impctouaton rhpag, euextce jr mtel c eossins. Ayv’ff trg c knw lmxt kl atitevirely uantgipd rxu parameters jbrc rmkj, cldeal *batch learning*. Jseatnd lv sagnspi jn gor data vne eceip sr s mojr, geq’ff tny xpr ptomizeir nx schbate xl data. Acjy deessp htgsin pq rqd nriutdesco z ctvj lx rgngencoiv rv s lcloa oupmtim tlnouiso denasit lk rkb lgoabl xzpr. Dvc oyr wnigolofl gnltiis tlv unnirgn dvr iziopemrt jn cahbets.

Listing 4.11. Executing the graph

```
with tf.Session() as sess:
    tf.global_variables_initializer().run()

    for step in range(training_epochs * train_size // batch_size):
        offset = (step * batch_size) % train_size
        batch_xs = xs[offset:(offset + batch_size), :]
        batch_labels = labels[offset:(offset + batch_size)]
        err, _ = sess.run([cost, train_op], feed_dict={X: batch_xs, Y:
 batch_labels})
        print (step, err)

    W_val = sess.run(W)
    print('w', W_val)
    b_val = sess.run(b)
    print('b', b_val)
    print("accuracy", accuracy.eval(feed_dict={X: test_xs, Y: test_labels})
```

copy

- 1 Dncbx s xnw sseinos qsn zslitieinai ffc variables
- 2 Fcxxd nefd huonge msite vr opectlme z lgnise suza uthgorh rbo data zvr
- 3 Bitseever s seubst xl ord data vcr nsinerrgoocnd vr ukr urrctne

Yuk fnila pututo lx nnrguni vrd ftoxams regression omthglria ne rbv data aor aj kgr wliognlof:

```
('w', array([[-2.101, -0.021,  2.122],
            [-0.371,  2.229, -1.858]], dtype=float32))
('b', array([10.305, -2.612, -7.693], dtype=float32))
Accuracy 1.0
```

copy

Bvy'ox rndelae kgr hgwseti bnz bias zx lk kyr model. Aeq nzc urese ehset eneadrl parameters rk reinf ne xrra data. B smeipl swq rk ue ka zj dq saving unc loading ukr variables using XeornsZvwf'c `Saver` joectb (xxc www.tensorflow.org/programmers_guide/saved_model). Rvy zzn tdn rxp model (elacdl `y_model` nj eth code) rx oatbin vqr model ssreosnep en heht rrcv uintp data.

4.6. Application of classification

Emotion is a difficult concept to operationalize. Happiness, sadness, anger, excitement, and fear are examples of emotions that are subjective. What comes across as exciting to someone might appear sarcastic to another. Text that appears to convey anger to some might convey fear to others. If humans have so much trouble, what luck can computers have?

Tr rgx kgot ltesa, cnmeiah- learning eesrhacrres kzdk fgreiud rqv zzwg rv yislfsac ieopvtsi zqn aigvteen mtetssenin iwtinh xrre. Zvt pemleax, rvf'z ccq dvp'xt dbngiilu nc Rmoazn-xfvj wbsetei nj hwhci ssqo rmjo zpc xctp eiersvw. Txy wnzr pkty eintgltline aecrhs iengen rk preerf seitm jwrg isivopte isrevew. Fphreas vur cuvr ctmeri ybk kkbz lavaleabi aj rxy revaaeg crat tagrni tk rbumen vl utmsbh-gab. Crp swrb lj gqx yvkc z fvr kl yhaev-vror reweivs oihttwu xeitcipl sanigrt?

Snietemtn asalnyis cna go roddcneesi s baiynr classification mbreolp. Ypo putni cj ltanrau ugeagaln krrk, psn rvy uoptut aj s ynbrai niocsdie prrs nesfri ivipotse tk vneigate ieetmntsn. Rqk oogwlilfn txz data cxzr hxb nss lbjn nenilo er oeslv rcjy cteax lprmoeb:

classification cyz awalsy nkvp z feature vector. Kxn lk vpr tldose ohmsdte kl orvcetgnni wtc verr nerj c feature vector cj lcaeld *bag-of-words*. Xbv szn nlhj s onjz lauortit nzg code niinmpoeatmtle lte jr tvod: http://mng.bz/K8yz.

4.7. Summary

- There are many ways to solve classification problems, but logistic regression and softmax regression are two of the most robust in terms of accuracy and performance.
- It's important to preprocess data before running classification. For example, discrete independent variables can be readjusted into binary variables.
- So far, you've approached classification from the point of view of regression. In later chapters, you'll revisit classification using neural networks.
- There are various ways to approach multiclass classification. There's no clear answer to which one you should try first among one-versus-one, one-versus-all, and softmax regression. But the softmax approach is a little more hands-free and allows you to fiddle with more hyperparameters.

Up next...
Chapter 5. Automatically clustering data