Linearna regresija

Predstavljajmo si, da izvajamo fizikalni poskus. Imamo vzmet, na kateri merimo raztezek v odvisnosti od sile, s katero napenjamo vzmet (na vzmet obešamo uteži z znano težo in merimo raztezek).

[1]

F [N]	1	2	3	4	5
x [cm]	10	20	35	55	80

Velja Hookov zakon, $F = kx$, torej sta x in F med seboj linearno odvisna. Poskušali bomo najti premico, ki se bo najbolje prilegala danim točkam v ravnini.

Če imamo 2 točki $T_1(x_1, y_1), T_2(x_2, y_2)$, lahko brez težav najdemo premico, ki gre točno skozi niju. Če pa je točk več, ni nujno, da obstaja premica, ki gre skozi vse točke. Iščemo taka a in b, da se bo premica kar najbolje prilegala danim točkam. Če gre premica skozi neko točko $T_i(x_i, y_i)$, potem velja:

$$0 = ax_i + b - y_i$$

Ker pa naša premica ne poteka direktno skozi vse točke pride do napake, ki jo bomo v točki T_i označili z ε_i.

$$\varepsilon_i = ax_i + b - y_i$$

Napaka je lahko pozitivna ali negativna, odvisno ali točka leži pod ali nad premico. Želimo zmanjšati velikost vseh napak, ne glede na to, ali so pozitivne ali negativne. (Lahko bi preprosto sešteli absolutne vrednosti napak, ampak pozneje funkcije ne bi mogli odvajati.) Zato seštejemo kvadrate vseh napak.

$$\varepsilon = \sum_{i=1}^{N} \varepsilon_i^2$$

[1] Podatki pridobljeni iz: `http://www.clemson.edu/ces/phoenix/labs/124/shm/`

$$\varepsilon = \sum_{i=1}^{N} (ax_i + b - y_i)^2$$

Naš cilj je, da minimiziramo to napako. Napako bomo minimizirali s pomočjo gradientnega spusta.

Gradientni spust

Z gradientnim spustom iščemo minimum ali pa maksimum neke funkcije, to pomeni kje ima funkcija največjo oziroma najmanjšo vrednost. Gradientni spust je zelo primitivna metoda kjer se preprosto premikamo v smer kamor funkcija narašča oziroma pada.

To si lahko predstavljamo s pomočjo slepega človeka, ki želi priti na vrh Golovca. Ko bo naš človek, poimenujmo ga Lojze, na neki točki hriba, ne bo videl v katero smer mora hoditi (ker je slep), čutil pa bo kako strm je hrib in v katero smer hrib narašca, v katero pa pada.. Na začetku, ko je hrib bolj strm, bo Lojze naredil večji korak, ko pa se bo približeval vrhu in bo hrib postajal manj strm, pa bo delal manjše korake, ker se lahko drugače zgodi, da bo prestopil vrh in se znašel na drugi strani Golovca.

Z gradientnim spustom se torej premikamo v smer, v katero funkcija pada. Smer v katero funkcija pada oziroma narašča nam predstavlja gradient te funkcije. Preden pa nadaljujemo z gradientom, si moramo pogledatni nekaj več o funkcijah.

Malo o funkcijah

Celo življenje smo zapisavoli funkcije kot $f(x)$. Ta zapis nam predstavlja funkcijo, ki je odvisna od parametra x. To pomeni, da nam za vsak dani x priredi novo številko. Če želimo narisati graf take funkcije, uporabimo 2D koordinatni sistem in nanj narišemo točke, ki nam jih da funkcija, to pomeni urejene pare $(x, f(x))$.

Ker pa smo v svobodni državi, nam ni prepovedano, da bi bila funkcija odvisna od več parametrov. Funkcijo odvisno od dveh parametrov bi zapisali kot $f(x, y)$. Taki funkciji podamo 2 številki, x in y, vrne pa nam eno številko. Če želimo narisati graf take funkcije potrebujemo 3D koordinatni sistem. Za vsako točko $T(x, y)$, bi nam taka funkcija priredila novo številko, ki bi v tem koordinatnem sistemu predstavljala višino, to je vrednost na z osi. Primer take funkcije je:

$$f(x, y) = \frac{\sin\sqrt{x^2 + y^2}}{\sqrt{x^2 + y^2}}$$

$\sqrt{x^2 + y^2}$ predstavlja oddaljenost točke, ki jo podamo funkciji od izhodišča.

Slika 1: 3D funkcija

Funkcija je seveda lahko odvisna tudo od več kot dveh spremenljivk. Takih funkcij pa na našo veliko žalost ne moremo narisati, ker bi že za funkcijo odvisno od treh spremenljivk, potrebovali 4 dimenzionalni prostor.

Glede na to da smo še vedno v svobodni državi, se lahko pri parametrih funkcije poigramo tudi z njihovim poimenovanjem. Nikjer ne piše, da ne smemo na mesto x uporabljati a in da naša funkcija ne sme izgledati kot $f(a)$. To seveda velja tudi za funkcije, ki vzamejo ve"parametrov, tako da lahko namest $f(x, y)$ pišemo $f(a, b)$. Funkcijo katere graf je narisan od zgoraj, bi lahko potemtakem zapisali kot:

$$f(a, b) = \frac{\sin\sqrt{a^2 + b^2}}{\sqrt{a^2 + b^2}}$$

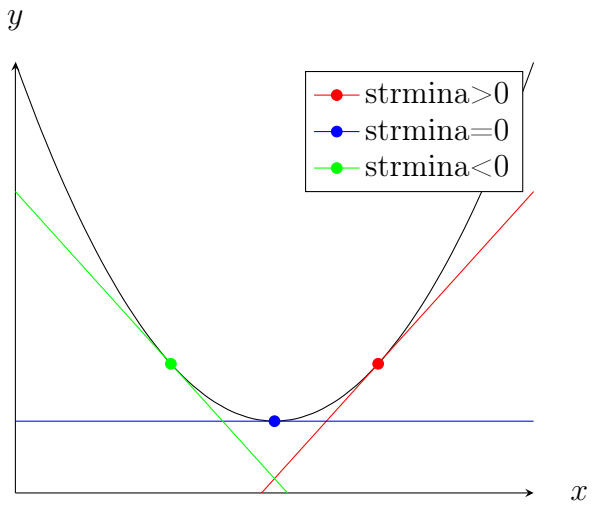
Definirali smo že napako premice za dane točke. Ker sedaj vemo, da je funkcija odvisna od večih spremenljivk in da ni nujno da sta ti spremenljivki x in y, lahko ugotovimo, da je naša napaka funkcija, ki je odvisna od spremenljivk a in b. Drugače seveda ne more biti, ker imamo že podano skupino točk, to pomeni x in y, za katere želimo najti premico, ki se jim prilega, tako da vse kar lahko spreminjamo v naši funkciji napake sta parametra a in b.

Odvod

Kot smo že omenili, nam gradient predstavlja strmino funkcije. Preden se lahko naprej pogovarjamo o gradientnem spust, potrebujemo definirati strmino funkcije.

Odvod neke funkcije $f(x)$ je strmina te funkcije v točki x. Odvod je pozitiven če funkcija narašca in negativen če funkcija pada. Večji kot je odvod, bolj strma je funkcija v dani točki. Odvod funkcije $f(x)$, ki ga označimo z $f'(x)$, nam torej pove, kakšen je koeficient tangente na graf fukncije $f(x)$ v točki x.

Slika 2: Odvod



Delni odvod

Povedali smo, da je funkcija lahko odvisna od večih parametrov. Tako funkcijo moramo odvajati po vsaki spremenljivki posebej, kar pomeni da potrebujemo funkcijo $f(x, y)$ odvajati po x in po y. S tem dobimo dva odvoda. Odvod po x nam pove kako funkcija narašča oziroma pada če spreminjamo x parameter, odvod po y pa nam seveda pove kako funkcija narašča oziroma pada po y parametru. Takemu odvodu rečemo delni odvod, zapišemo pa ga kot $\frac{\partial f(x,y)}{\partial x}$ za delni odvod po x in $\frac{\partial f(x,y)}{\partial y}$ za delni odvod po y.

Za trenutek se vrnimo na začetni problem, ki nas je pripeljal do sem. Želelimo najti tako premico, ki se najbolj prilega vsem danim točkam. Definirali smo že skupno napako in ugotovili da je naša napaka funkcija, ki je odvisna od spremenjivk a in b. Sedaj lahko poračunamo, kako strma je naša napaka za dani a in kako strma je naša napaka za dani b. Drugače povedano, izračunamo lahko delni odvod napake po a in delni odvod napake po b.

$$\frac{\partial \varepsilon}{\partial a} = \sum_{i=1}^{N} 2\frac{\partial(ax_i + b - y_i)}{\partial a} = 2\sum_{i=1}^{N}(ax_i + b - y_i)x_i$$

$$\frac{\partial \varepsilon}{\partial b} = \sum_{i=1}^{N} 2\frac{\partial(ax_i + b - y_i)}{\partial b} = 2\sum_{i=1}^{N}(ax_i + b - y_i) \cdot 1 = 2\sum_{i=1}^{N}(ax_i + b - y_i)$$

Gradient

Gradient neke funkcije je vektor, ki kaže v smer naraščanja te funkcije. Gradient funkcije f označimo z ∇f. Pri funkciji z enim parametrom ($f(x)$), ima gradient te funkcije samo eno komponento, to je odvod funkcije $f(x)$. To zapišemo kot:

$$\nabla f = \begin{pmatrix} f' \end{pmatrix}$$

Če je funkcija odvisna od dveh spremenljivk, je vsaka komponenta gradienta en delni odvod. To pomeni da nam prva komponenta gradienta pove kam narašča funkcija in kako strma je glede na prvi parameter, druga komponenta gradienta pa nam pove kam narašča funkcija in kako strma je glede na drugi parameter funkcije.

Za lažjo predstavo si oglejmo gradient funkcije $f(x, y)$. Prva komponenta tega gradienta nam bo povedala strmino funkcije v smeri x osi, druga komponenta tega gradienta pa nam bo povedala strmino funkcije v smeri y osi.

V splošnem lahko gradient funkcije f, ki je odvisna od n parametrov zapišemo kot:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Gradient naše napake pa bi izgledal takole:

$$\nabla \varepsilon = \begin{pmatrix} \frac{\partial \varepsilon}{\partial a} & \frac{\partial \varepsilon}{\partial b} \end{pmatrix}$$

Prilagajanje premice

Sedaj ko razumemo kaj je gradient, se lahko vrnemo na naš začetni problem, to je najti tako premico, ki se najbolj prilega N točkam. Na tem primeru bomo razložili tudi gradientni spust.

Najprej se spomnimo kaj sploh želimo narediti. Naša funkcija je $\varepsilon_i = ax_i + b - y_i$, kjer ε_i predstavlja napako te funkcije v točki $T_i(x_i, y_i)$. Skupno napako smo definirali kot: $\varepsilon = \sum_{i=1}^{N} \varepsilon_i^2$ in jo tudi odvajali. Definirali smo tudi gradient naše funkcije.

Vemo torej v katero smer naša funkcija napake narašča. Sedaj se lahko premaknemo v nasprotno smer, to pomeni tja, kamor naša funkcija pada. To da se premaknemo v neko smer, pomeni da spremenimo naš a in b tako, da bo napaka manjša. Parameter a zmanjšamo za vrednost, prve komponenta

našega gradienta, ker je ta komponenta delni odvod napake po a in nam pove, kako se napaka spreminja, če spreminjamo a. Parameter b pa seveda zmanjšamo za vrednost druge komponente našega gradienta.

Pa razpišimo te spremembe. Spremembo a lahko napišemo kot:

$$\Delta a = -(\nabla \varepsilon)_1 \cdot \lambda$$

Minus uporabimo zato, ker se premikamo v smer kamor funkcija pada, to pomeni nasprotno od tega kamor kaže gradient. λ je konstanta, ki nam predstavlja kako hitro spreminjamo naš parameter. λ določimo mi. O točnem pomenu λ se bomo pogovirli nekoliko kasneje.

Spremembo b pa lahko zapišemo kot:

$$\Delta b = -(\nabla \varepsilon)_2 \cdot \lambda$$

V te dve enačbi lahko sedaj vstavimo delna odvodi, ki smo jih naračunali prej in dobimo:

$$\Delta a = -\sum_{i=1}^{N} (ax_i + b - y_i)x_i \cdot \lambda$$

$$\Delta b = -\sum_{i=1}^{N} (ax_i + b - y_i) \cdot \lambda$$

Pri obeh enačbah smo se lahko znebili 2, ker lahko povečamo našo konstanto λ in dobimo enak rezultat.

Sedaj ko vemo za koliko moramo spremeniti naša parametra a in b, jih seveda tudi spremenimo. Naša nova parametra sta torej:

$$a = a + \Delta a$$

$$b = b + \Delta b$$

Z vsem tem delom smo en korak bližje našim želenim vrednostim. Ta postopek moramo ponoviti še velikokrat, zato izkoristimo računalnik. Večkrat kot ponovimo postopek, bolj natančen bo naš rezultat. Temu spreminajnju parametrov v smer gradienta pravimo gradientni spust.

Natančnost našega rezultata je odvisna od λ (velikost premika) in začetnih vrednost spremenljivk a in b. Z manjšo velikostjo premika bo rezultat bolj natančen, vendar bomo morali postopek večkrat ponoviti. Ker s postopkom iščemo samo lokalne minimume, nam začetna vrednost spremenljivk določi kateri minimum bomo našli.

Linerana regresija elipse

Seveda se nam ni treba omejiti samo na prilagajanje premice točkam. Če imamo na primer točke na krožnici nekega planeta, bomo tem točkam prilagodili elipso. Recimo da imamo N točk krožnice nekega planeta $T_i(x_i, y_i)$. Podobno kot pri prejšnjem primeru, bomo tudi tokrat iskali funkcijo, ki se točkam najbolj prilega. Tokrat bomo namesto premice iskali elipso, saj se elipsa seveda bolj prilega krožnici planeta kot pa premica.

Ker je elipsa stožnica, obomo za funkcijo, ki jo prilagajamo vzeli splošno enačbo stožnic:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

Tako kot prej se nam bo pojavila napaka, ki jo označimo z ε.

$$\varepsilon_i = Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F$$

Za razliko od premice, je oblika stožnice odvisna od šestih parametrov namesto dveh. To so: A, B, C, D, E in F. Zato bomo torej spreminjali teh šest parametrov. To pomeni da je naša napaka funkcija, ki je odvisnah od šestih parametrov.

Podobno kot pri premici najprej definiramo skupno napako kot vsoto kvadratov vseh napak:

$$\varepsilon = \sum_{i=1}^{N} \varepsilon_i^2$$

$$\varepsilon = \sum_{i=1}^{N} (Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)^2$$

Enako kot pri premici moramo našo napako delno odvajati po spremenjivkah od katerih je naša napaka odvisna. To pomeni da potrebujemo izračunati delni odvod napake po A, B, C, D, E in F. Delni odvodi za enačbo stožnic so:

$$\frac{\partial \varepsilon}{\partial A} = \sum_{i=1}^{N} 2(Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)(x_i^2)$$

$$\frac{\partial \varepsilon}{\partial B} = \sum_{i=1}^{N} 2(Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)(x_iy_i)$$

$$\frac{\partial \varepsilon}{\partial C} = \sum_{i=1}^{N} 2(Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)(y_i^2)$$

$$\frac{\partial \varepsilon}{\partial D} = \sum_{i=1}^{N} 2(Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)(x_i)$$

$$\frac{\partial \varepsilon}{\partial E} = \sum_{i=1}^{N} 2(Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)(y_i)$$

$$\frac{\partial \varepsilon}{\partial F} = \sum_{i=1}^{N} 2(Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F)$$

Da najdemo najboljše parametre, pri katerih bo funkcija imela najmanjšo napako, bomo ponovno uporabili gradientni spust. Za razliko od gradientnega spusti pri eni premici, bomo tokrat spreminjali šest parametrov. Naš gradient je torej:

$$\nabla\varepsilon = \left(\frac{\partial\varepsilon}{\partial A} \quad \frac{\partial\varepsilon}{\partial B} \quad \frac{\partial\varepsilon}{\partial C} \quad \frac{\partial\varepsilon}{\partial D} \quad \frac{\partial\varepsilon}{\partial E} \quad \frac{\partial\varepsilon}{\partial F}\right)$$

To pomeni da so naše spremembe parametrov:

$$\Delta A = -(\nabla\varepsilon)_1 \cdot \lambda$$

$$\Delta B = -(\nabla\varepsilon)_2 \cdot \lambda$$

$$\Delta C = -(\nabla\varepsilon)_3 \cdot \lambda$$

$$\Delta D = -(\nabla\varepsilon)_4 \cdot \lambda$$

$$\Delta E = -(\nabla\varepsilon)_5 \cdot \lambda$$

$$\Delta F = -(\nabla \varepsilon)_6 \cdot \lambda$$

Iz česar sledi, da so naši novi parametri podobno kot pri premici:

$$A = A + \Delta A$$

$$B = B + \Delta B$$

$$C = C + \Delta C$$

$$D = D + \Delta D$$

$$E = E + \Delta E$$

$$F = F + \Delta F$$

Pravilnost našega rezultata je seveda odvisna od tega koliko ponovitev bomo naredili, kolikšna bo naša λ in kakšne začetne vrednosti A, B, C, D, E in F smo si izbrali.

Logistična regresija

Glavna razlika med logistično in linearno regesijo je, da pri linearni regresiji določamo zvezno spremenljivko (y je odvisen od x), medtem ko nam pri logistični regresiji model vrne kakšna je verjetnost, da vhodni podatki sodijo v določeno kategorijo.

To si lahko predstavljamo kot funkcijo, ki vrne več parametrov. Linearna regresija nam predstavlja funkcijo, ki vrne več parametrov in vrne eno številko oziroma parameter. Logistična regresija pa je funkcija, ki vzame več parametrov in vrne več parametrov, to je številk.

Logistična regresija se uporablja za kalsifikacijo. Eden izmed takih primerov je prepoznavanje števil na sliki. Če se lotimo prepoznavanja enomestnih števil, bi nam logistična regresija vrnila 10 parametrov oziroma števil. Vsako izmed teh števil nam predstavlja verjetnost da je na sliki določena cifra. Prvo število nam recimo predstavlja verjetnost, da je na sliki 0, drugo da je na sliki 1, tretje da je na sliki 2 in tako naprej.

Nevronska mreža

Logistično regresijo pa lahko zakompliciramo še nekoliko bolj. Števila, ki nam jih vrne logistična regresija lahko vzamemo kot vhodne parametre na novi logistični regresiji. In potem lahko to ponovimo še enkrat, dvakrat ali pa celo n-krat. Takemu sistemu pravimo nevronska mreža.

Dodatno

Ekliptični koordinatni sistem

Eden izmed koordinatnih sistemov za določanje lege nebesnih teles. Lokacijo določimo s tremi podatki. Ker so koordinate definirane glede na ravnino zemljine orbite, je z koordinata za Zemljo vedno enaka 0.

l longtituda, ekliptična dolžina, od 0° do 360°.

b latituda, ekliptična širina od $-90°$ do 90°.

r razdalja

V kartezične koordinate podatke pretvorimo po formulah:

$$x = r \cos b \cos l$$

$$y = r \cos b \sin l$$

$$z = r \sin b$$

Ker tiri vseh planetov ležijo v (skoraj) isti ravnini, se bomo naredili fizike in z koordianto zanemarili.