Avtomatsko odvajanje: prvi koraki

V prejšnjem poglavju smo obravnavali tehnike zmanjšanja dimenzij, s katerimi lahko podatke prikažemo v dvodimenzionalnih kartah oziroma razsevnih diagramih. Cilj teh metod je poenostaviti podatke tako, da ohranimo njihovo strukturo in odnose, pri čemer vsaka tehnika daje prednost določenim lastnostim podatkov. Zmanjšanje dimenzij je ključno pri vizualizaciji kompleksnih podatkov, saj omogoča intuitivno predstavitev odnosov med podatkovnimi primeri.

Podrobneje smo obravnavali tri pristope: metodo glavnih komponent, večdimenzionalno lestvičenje in t-SNE. Vsaka od teh metod temelji na optimizaciji kriterijske funkcije, ki določa, kako naj bo podatkovna predstavitev v nižji dimenziji čim bolj smiselna. Pri metodi glavnih komponent je ta funkcija povezana z varianco podatkov v novih koordinatnih oseh, pri večdimenzionalnem lestvičenju z ohranjanjem razdalj med podatkovnimi primeri, pri t-SNE pa z ohranjanjem lokalnih sosedstev podatkovnih točk:

1. Cilj metode PCA je poiskati smer $\mathbf{u}_1$ v podatkovnem prostoru, v kateri imajo projekcije podatkov največjo varianco. Če podatkovno matriko X projiciramo na enotski vektor $\mathbf{u}_1$, dobimo nove koordinate podatkov v tej smeri. Varianca teh projekcij določa razpršenost podatkov v novi osi, zato želimo poiskati vektor $\mathbf{u}_1$, ki maksimizira to varianco. To vodi do sledečega optimizacijskega problema, njegova rešitevpa določi prvo glavno komponento kot smer največje variance v podatkih.

$$\max_{\mathbf{u}_1} \frac{1}{n} \sum_{i=1}^{n} (X_i \mathbf{u}_1)^2, \quad \text{pri pogoju} \quad \mathbf{u}_1^T \mathbf{u}_1 = 1$$

2. MDS minimizira razliko med razdaljami podatkovnih točk v izvirnem in znižanem prostoru. Kriterijska funkcija je vsota kvadratov razlik med izvirnimi razdaljami d_{ij} in novo dobljenimi razdaljami $\|\mathbf{z}_i - \mathbf{z}_j\|$:

$$\min_{\mathbf{z}_1, \ldots, \mathbf{z}_n} \sum_{i<j} \left(d_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|\right)^2$$

   kjer so $\mathbf{z}_i$ točke v nizkodimenzionalnem prostoru, ki jih iščemo.
3. Tehnika t-SNE ohranja sosednje odnose med točkami, zato minimizira Kullback-Leiblerjevo divergenco med verjetnostnimi porazdelitvami sosedstev v

originalnem in nizkodimenzionalnem prostoru:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_n} \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

kjer je p_{ij} verjetnost, da sta podatkovni točki i in j sosednji v visokodimenzionalnem prostoru, in q_{ij} enaka verjetnost v nizkodimenzionalnem prostoru.

Pri metodi glavnih komponent lahko rešitev poiščemo analitično, kar pomeni, da za dano kriterijsko funkcijo obstaja točno določena matematična formula, ki nam neposredno poda rešitev. Takih rešitev pa ne moremo dobiti pri večdimenzionalnem lestvičenju in t-SNE, saj njihovi kriteriji niso enostavno rešljivi. V prejšnjem poglavju smo se izognili vprašanju, kako dejansko priti do optimalne rešitve za takšne probleme. Tu je zdaj čas, da podrobneje raziščemo postopke, ki nam lahko pomagajo pri iskanju rešitev v primerih, ko analitičnega odgovora ni.

Ti postopki predvidevajo, da imamo na vhodu podatke, definiramo model, katerega struktura je matematično jasno podana, in določimo kriterijsko funkcijo, ki jo želimo optimizirati. Cilj optimizacije je poiskati tiste vrednosti parametrov modela, pri katerih kriterijska funkcija doseže svojo optimalno vrednost – običajno minimum. Pri t-SNE in večdimenzionalnem lestvičenju so parametri, ki jih iščemo, dvodimenzionalne koordinate podatkovnih točk v vložitvenem prostoru, pri drugih problemih pa so lahko uteži modelov, parametri funkcij ali druge vrednosti, ki določajo optimalno rešitev.

V tem poglavju bomo začeli postopoma, najprej na enostavnih funkcijah, saj nas bo zanimal predvsem sam postopek optimizacije. Ko bomo osvojili osnovne koncepte, bomo to znanje razširili in pokazali, da lahko s to metodo rešujemo tudi mnogo drugih problemov. Gradientni sestop ni uporaben le za zmanjševanje dimenzionalnosti, temveč tudi za gradnjo napovednih modelov, izdelavo nevronskih mrež, sisteme za priporočanje in številne druge naloge, kjer moramo optimizirati matematično funkcijo. Razumevanje tega postopka je zato ključno za nadaljnjo obravnavo sodobnih metod strojnega učenja.

Primer

Začnimo s preprostim primerom, ki nam bo pomagal intuitivno razumeti gradientni sestop. Obravnavali bomo kvadratno funkcijo:

$$f(a) = a^2 - 10a + 28$$

in poiskali takšno vrednost parametra a, pri kateri funkcija doseže minimum.

Analitična rešitev. Ker je dana funkcija kvadratna, lahko njen minimum poiščemo analitično. Kvadratne funkcije oblike

$$f(a) = c_1 a^2 + c_2 a + c_3$$

imajo ekstrem v točki, kjer je prvi odvod enak nič. Odvajamo funkcijo:

$$\frac{d}{da} f(a) = 2a - 10$$

Minimum najdemo tako, da odvod enačimo z nič:

$$2a - 10 = 0$$

Rešimo za a:

$$a = 5$$

Vrednost funkcije v tej točki je:

$$f(5) = 5^2 - 10(5) + 28 = 25 - 50 + 28 = 3$$

Torej funkcija doseže svoj minimum pri $a = 5$, kjer velja $f(5) = 3$.

Numerična rešitev. Če analitične rešitve ne bi poznali, bi lahko uporabili **gradientni sestop**, ki iterativno prilagaja vrednost parametra a, dokler ne doseže minimuma funkcije.

Gradientni sestop deluje tako, da se premikamo v smeri negativnega gradienta (odvoda funkcije), saj ta kaže proti nižjim vrednostim funkcije. Posodobitev parametra a v vsakem koraku poteka po formuli:

$$a_{t+1} = a_t - \eta \frac{d}{da} f(a_t)$$

kjer je η (grški simbol "eta") **hitrost učenja**, ki določa, kako veliki bodo koraki pri posodabljanju vrednosti parametra.

Začnimo s **začetno vrednostjo** $a_0 = 6$ in izberimo hitrost učenja $\eta = 0.1$.

1. Izračunamo odvod funkcije pri $a_0 = 6$:

$$\frac{d}{da} f(6) = 2(6) - 10 = 12 - 10 = 2$$

   Posodobimo vrednost parametra:

$$a_1 = 6 - 0.1 \times 2 = 6 - 0.2 = 5.8$$

2. Izračunamo gradient pri $a_1 = 5.8$:

$$\frac{d}{da} f(5.8) = 2(5.8) - 10 = 11.6 - 10 = 1.6$$

   Posodobimo vrednost parametra:

$$a_2 = 5.8 - 0.1 \times 1.6 = 5.8 - 0.16 = 5.64$$

3. Izračunamo gradient pri $a_2 = 5.64$:

$$\frac{d}{da} f(5.64) = 2(5.64) - 10 = 11.28 - 10 = 1.28$$

   Posodobimo vrednost parametra:

$$a_3 = 5.64 - 0.1 \times 1.28 = 5.64 - 0.128 = 5.512$$

Postopek nadaljujemo, dokler se vrednost a ne približa 5. Če bi nadaljevali, bi videli, da se vrednosti sčasoma stabilizirajo okoli $a = 5$, kar je pričakovani minimum.

Gradientni sestop

Gradientni sestop deluje tako, da v vsakem koraku oceni, kako strma je funkcija v trenutni točki, in nato prilagodi vrednost parametra v smeri nižje vrednosti. Pomembno je izbrati ustrezno hitrost učenja η – če je prevelika, lahko preskočimo minimum, če je premajhna, bo postopek zelo počasen.

V gradientnem sestopu iščemo vrednost parametra, ki minimizira kriterijsko funkcijo. Če funkcijo označimo s $f(\theta)$, kjer je θ vektor parametrov, potem gradientni sestop deluje po naslednji iterativni posodobitvi:

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

kjer je $\nabla f(\theta_t)$ gradient funkcije v trenutni točki, ki nam pove smer največjega naraščanja funkcije. Ker želimo minimizirati funkcijo, se premikamo v nasprotni smeri gradienta.

Da lahko izvedemo gradientni sestop, moramo poznati odvod funkcije po danem parametru. V primeru enega parametra je gradient enak navadnemu odvodu, v večdimenzionalnem primeru pa gre za vektor vseh delnih odvodov:

$$\nabla f(\theta) = \left(\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2}, \ldots, \frac{\partial f}{\partial \theta_n} \right)^T$$

Gradient funkcije lahko določimo na dva načina: **analitično** ali **numerično**.

Analitična rešitev temelji na simbolnem odvajanju funkcije, pri katerem neposredno izpeljemo enačbo za gradient. To je običajno bolj natančno in učinkovito, vendar je lahko pri kompleksnih funkcijah težko izvedljivo.

Alternativno lahko gradient izračunamo numerično z metodo končnih diferenc. Če želimo izračunati odvod funkcije f po parametru θ_i, lahko uporabimo aproksimacijo:

$$\frac{\partial f}{\partial \theta_i} \approx \frac{f(\theta_i + h) - f(\theta_i)}{h}$$

kjer je h majhno število (npr. $h = 10^{-5}$).

Numerični gradient ima več slabosti. Prvič, izbira vrednosti h vpliva na natančnost – če je prevelik, dobimo grobo oceno, če je premajhen, lahko pride do numeričnih napak zaradi omejene natančnosti računalniške aritmetike. Drugič, za vsak parameter moramo izračunati vrednost funkcije vsaj dvakrat, kar pomeni, da je numerični gradient računsko drag pri funkcijah z veliko parametri.

Zaradi teh slabosti se gradientni sestop običajno izvaja s simbolnim (analitičnim) gradientom, ki ga dobimo z odvajanjem kriterijske funkcije. To pomeni, da moramo že na začetku določiti enačbe za vse odvode po parametrih in jih nato uporabiti pri posodobitvah parametrov.

Kriterijske funkcije v strojnem učenju in umetni inteligenci običajno niso preproste, kot v primeru kvadratne funkcije, temveč so lahko zelo kompleksne. V primeru nevronskih mrež imamo opravka s funkcijami, ki vsebujejo več milijonov parametrov. Ročno računanje gradientov bi bilo v takšnih primerih zamudno in nepraktično, zato se uporablja avtomatično izračunavanje gradientov.

Postopek avtomatičnega izračuna gradienta temelji na metodi avtomatskega odvajanje (angl. *automatic differentiation*), ki omogoča natančno in učinkovito računanje gradientov kompleksnih funkcij brez uporabe simbolne ali numerične diferenciacije. Ta metoda je osnova za algoritme učenja v nevronskih mrežah in drugih naprednih modelih strojnega učenja.

V nadaljevanju bomo podrobneje raziskali, kako gradientni sestop deluje v večdimenzionalnem prostoru in kako lahko pospešimo njegovo delovanje s pomočjo različnih različic algoritma.

Zametek programa za avtomatsko odvajanje

Na predavanju smo postopoma, po majhnih korakih, razvili kodo za avtomatsko odvajanje. Začeli smo z enostavno funkcijo

$$L(a, b, c, d) = (ab + c)f$$

in analizirali izračun njenega gradienta v dani točki, na primer pri $a = 2$, $b = -3$, $c = 10$, $d = -2$.

Pri razvoju kode smo se močno zgledovali po izjemnem predavanju Andreja Karpathyja *The spelled-out intro to neural networks and backpropagation: building micrograd* ter uporabili pristop iz njegove knjižnice micrograd (GitHub repo).

Besedilo tega poglavja bomo po predavanju primerno dopolnili s kodo in primeri.