A gépi tanulás alapjai

Csapó Ádám Balázs

Tartalomjegyzék

1. fejezet

Bevezető

Forrásmegjelölés

A fejezetben számos magyarázat és példa az alábbi angol nyelvű referenciákból származik: [1, 2, 4, 6].

1.1. Mivel foglalkozik a gépi tanulás?

A hétköznapi szóhasználatból kiindulva a tanulás' szónak sokféle jelentése, konnotációja lehet – magában foglalhatja pl. a lexikális tudás elsajátítását, vagy akár a gyakorlat útján szerzett tapasztalatszerzést, és az implicit-explicit skálán is sokféle árnyalata lehet. Most tekintsünk el ettől, és a gépi tanuláshoz tekintsünk egy általánosabb és formálisabb megfogalmazást.

A gépi tanulás abból indul ki, hogy van néhány „példányunk" – amelyek lehetnek tárgyak, események vagy akár folyamatok valamilyen reprezentációi – és amelyek egyúttal példák is (vagy ellenpéldák) valamilyen kategóriára. Ezeket a példányokat, akár több kategóriára nézve is, ***tanítóhalmaznak*** nevezzük. A cél, hogy ha később látunk egy újabb példányt, amely a tanítóhalmazban nem szerepel, akkor a tanuló módszerünk meg tudja mondani, hogy az milyen kategóriába tartozik. Ezt nevezzük ***általánosításnak***.

A gépi tanulás által néhány tipikusan megválaszolt kérdés lehet, hogy:

- Szék vagy asztal van a képen?
- Szerepel-e emberi arc a képen?
- Holnap délelőtt várhatóan erősödik majd vagy gyengül a Forint, és mennyivel?
- Mennyi idős az a felhasználó, aki ezt a Youtube videot nézi?
- ...

A kérdések megválaszolásához szükségünk lesz valamilyen „szabályra” (ez lesz az ún. ***hipotézis***), ami a gyakorlatban (általánosítás esetén) is jól működik!

1.1.1. A gépi tanulás algoritmikus oldala

A hipotézis „típusa” sem mindegy: mik azok a lehetséges „szabály-formák”, amik szóba jöhetnek? Például a szabály-forma állhat:

- Csak konjunkciókból („A és B és nem C és D”)
- Csak diszjunkciókból („A vagy B vagy nem C”)
- Diszjunktív normálformákból („(A és B) vagy (C és nem D)”)
- Fuzzy szabályokból („A kicsit langyos és B magas”)
- Grafikus modellekből („Az adatokat jól leírja ez a 10 darab rejtett változó, az alábbi módon” + egy irányított vagy irányítatlan gráf)

A hipotézis „típusát” szokás ***modellnek*** is nevezni. Modelltől függően más és más módon fogjuk a konkrét hipotézist megkeresni.

Fentiek alapján a tanulómódszerek különbözhetnek az alábbi *algoritmikus* kérdésekben:

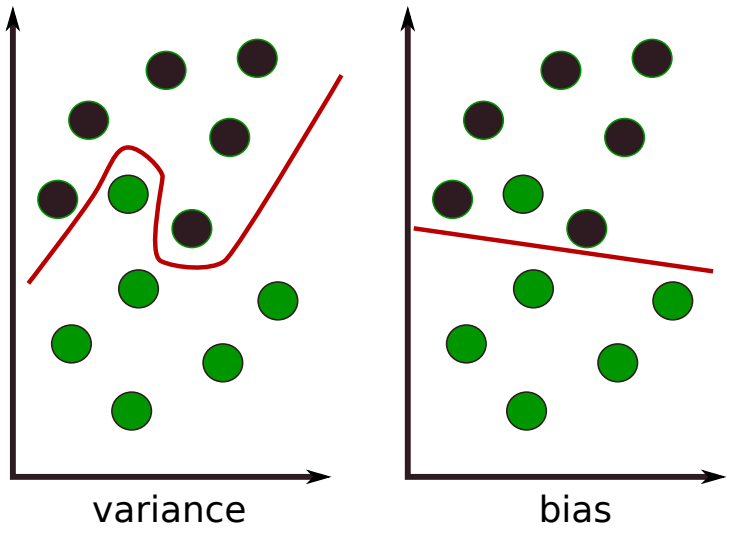
- a tanítóminták kiválasztásának módjában (felügyelt / nem felügyelt? pre-processzált? hogyan mintavételezett?)
- a választott modellben (generatív / diszkriminatív? parametrikus / nem parametrikus?)
- a konkrét hipotézis megkeresésének módjában

1.1.2. A gépi tanulás statisztikai oldala

Ezek *algoritmikus* kérdések voltak, de a tanulásnak vannak *statisztikai* vonatkozásai is, ugyanis mi történik, ha például a világ megváltozik – magyarán, ha hirtelen érvénytelenné válnak a megtanult hipotézisek? Még az is előfordulhat, hogy a világ maga sem determinisztikus... (erről jó, ha a fizikusok és a filozófusok vitatkoznak, és ez mindenki számára fontos, de a gépi tanulás gyakorlatában az a célunk, hogy a helyes választól függetlenül praktikus megoldásokhoz jussunk.)

A megtanult modell általánosítás során mutatott hibái nagyvonalakban az alábbi hiba-típusokra bonthatóak:

- **„Variancia típusú”** hibák: ezek a típusú hibák arra utalnak, hogy valamilyen szempontból a tanítóhalmaz mintái nem voltak megfelelőek (rossz példányokra tanultunk):
  - Kevés példányon tanultunk, kevés volt az adat
  - A tanuláshoz használt példányok zajosak voltak



1.1. ábra. Variancia és torzítás közötti kompromisszum jellemzése. A nagy bias-szal rendelkező modellek „merevek”, nem követik le pontosan a tanítóhalmaz mintáit, viszont általánosítás céljából esettől függően jobbak lehetnek (amennyiben a tanítóadatokban levő variancia nagy része egyébként is zaj). Fordítva, a magas varianciájú modellek hipotézise nagyon erősen függ a tanítóadatoktól, ezért ha azok zajjal terheltek (vagy más okból kifolyólag nem reprezentatívak), a modell szuboptimális lehet.

  - A tanuláshoz használt példányok nem voltak reprezentatívak (torzított mintavételezés, ritka példányok, ld. *black swan paradox*)

- **„Bias típusú”** hibák: ezek a típusú hibák arra utalnak, hogy a választott modell nem volt alkalmas az adott problémára
  - A „*No-Free-Lunch tételek*” éppen ezt mondják ki: nem létezik olyan modell, ami minden problémára jól működik (Wolpert, 1996)
  - „*All models are wrong, but some models are useful*” (G. Box, 1987)

A tévedés lehetősége összefügg a variancia-torzítás dilemmával (**variance-bias dilemma**): Ha az összes tanítómintát meg akarjuk tanulni, magas lehet a variancia (ami egyenértékű azzal, hogy ha más tanítómintákat választottunk volna, nagyon más lenne a kapott modell, magyarán: a zajokra „tanultunk rá”).

Ha viszont egyszerűbb modellhez ragaszkodunk, magas lehet a „torzítás” (hiába gyűjtünk akár több adatot, a modell merev marad, nem képes azokat „felszívni”).

Ezt a kommpromisszumot mutatja be a 1.1. ábra. A későbbiekben (1.4. fejezet) mindezt matematikailag precízebben is levezetjük.

1.2. A gépi tanulás matematikai alapjai

Milyen matematikai eszközökre lesz szükségünk a gépi tanulásban? Röviden összefoglalva az alábbi eszközök kerülnek majd szóba:

- Analízis: leginkább többváltozós függvények parciális deriválására lesz szükségünk a gradiens alapú optimalizációhoz
- Valószínűségelmélet alapfogalmai
- (Minimális) statisztikai alapismeretek

1.2.1. A probabilisztikus megközelítés motivációja

A probabilisztikus megközelítés azért nagyon fontos, mert bármit is mond egy modell, a tévedés valószínűsége sohasem lesz 0. Ugyanis:

- Akármit is tanulunk, a cél úgyis az általánosítás
- Előfordulhat, hogy rossz tanítómintákra tanultunk (variancia típusú hibák), pl. azok:
  - számban kevesek, nem reprezentatívak
  - nem tartalmaznak ritka mintákat, amikre nem „gondoltunk”
  - nem elég pontos a mintavételezésünk, ...
- Előfordulhat az is, hogy nem a megfelelő modellt (hipotézis-osztályt) használjuk (bias típusú hibák)

Cél, hogy ezt a valószínűséget (nemcsak tanuláskor, hanem főleg általánosításkor!) statisztikai módszerekkel mégis leszorítsuk. Magyarán: a hiba (általánosan: költségfüggvény) várható értékét szeretnénk csökkenteni!

Éppen ezért célszerű probabilisztikus (valószínűségi) megoldásokat használnunk!

A probabilisztikus megközelítés további előnye, hogy a számunkra ismeretlen tényezőket a „szőnyeg alá söpörhetjük”. Például: $P(\text{a rendszer meghibásodik}) = 0.005$. Ebben mint gyűjtő-kategóriában benne lehet az is, hogy egyidőben földrengés és atomtámadás van, és ezért nem helytálló a modellünk; magyarán a probabilisztikus megközelítés rugalmasan teret enged az empirikus megfogalmazásoknak.

Ha pedig a kimenet is probabilisztikus – de konkrét válaszra van szükség – megtehetjük, hogy visszaadjuk azt a választ, amelyikben a legbiztosabbak vagyunk. Erre sokféle módszer létezik, különböző számítási igényekkel – ld. később, pl.:

- Maximum likelihood estimation (MLE): azt a modellt / kimenetet választjuk, amely az adatokkal a leginkább összhangban áll (az adatok valószínűsége ennek feltételezésével a legnagyobb)
- Maximum a posteriori (MAP): azt a modellt / kimenetet választjuk, amely a leginkább összhangban áll mind az előzetes feltételezésünkkel (prior), mind a látott tanítóadatok kombinációjával
- Markov Chain Monte Carlo (MCMC), Metropolis-Hastings (MH) és hasonló szimuláció alapú módszerek: ha a fenti valószínűségek nem könnyen számszerűsíthetőek (főleg a normalizálás

nehézsége miatt), választhatjuk azt az utat is, hogy a lehetséges modellek szerint sokszor generálunk hipotetikus adatokat, és azok hasonlóságát vizsgáljuk meg a meglévő adatokhoz viszonyítva. Ez a hasonlóság adhat egy mértéket arra vonatkozóan, hogy melyik modell mennyire megfelelő.

Másrészről viszont, ha valamit részeredményként tovább használunk, jobb továbbra is valószínűségekkel dolgozni.

1.2.2. A valószínűség formális definíciója

A valószínűség a mindennapokban bizonyosságunk mértékét fejezi ki egy bizonytalan esemény bekövetkeztében. Például: „*Kis valószínűséggel esni fog az eső*". A valószínűség-elmélet ezt az intuitív definíciót formális alapokra helyezi, annak érdekében, hogy ha például új tudásra teszünk szert, a számon tartott valószínűségeket megfelelő műveletekkel frissíteni tudjuk.

Mielőtt rátérnénk a valószínűségek reprezentálására, meg kell mondanunk, milyen események fölött értelmezzük őket egyáltalán. Néhány tipikus példa:

- Kockadobás lehetséges eredményei felett
- Lóverseny lehetséges eredményei felett
- Lehetséges időjárási viszonyok felett Budapesten
- Egy gép meghibásodásainak lehetséges okai felett

Valószínűségi változók

Egy ***változó*** olyan tulajdonság vagy mérés, ami egy értéket vesz fel több lehetséges érték közül. A lehetséges értékek terét Ω-val jelöljük, és a változó ***értékkészletének*** nevezzük. Például:

- Dobókocka esetén: $\Omega = \{1, 2, 3, 4, 5, 6\}$
- Lóverseny esetén sokkal nagyobb kombinatorikus tér: $\Omega = \{\text{befutások összes lehetséges sorrendje}\}$

Egy változó ***valószínűségi változó***, ha az értékkészlet elemeihez valószínűségek is tartoznak.

Véletlen események

Egy vagy több valószínűségi változó értékeiből ún. ***véletlen eseményeket*** képezhetünk a halmaz-unió és halmaz-metszet, vagy halmaz-negált műveletek segítségével. A véletlen eseményeket a konvenció szerint α-val jelölhetjük. Például:

- Dobókocka esetén egy lehetséges esemény, hogy páratlan számot dobunk. Ekkor: $\alpha = \{1, 3, 5\}$ (diszjunkció, azaz logikai unió / vagy)

- Lóversenyen: „Kincsem nyer” – minden olyan kimenetelt jelenti, ahol Kincsem az első (diszjunkció, azaz logikai unió / vagy)
- Vagy $\alpha = \{$első dobásra 5-öst dobok, és holnap hamburgert ebédelek$\}$ (konjunkció, azaz logikai metszet / és)

A véletlen események definiálásakor implicit módon feltételeztük azt is, hogy létezik egy *mérhető eseményekből* álló $\mathcal{S}$ halmaz, mely az összes lehetséges véletlen eseményt tartalmazza. Egy valószínűségi változó esetén ez tulajdonképpen Ω hatványhalmaza (**vagy** annak valamilyen részhalmaza). Több valószínűségi változó esetén beszélhetünk az értékkészletek hatványhalmazainak keresztszorzatáról is.

Mindazonáltal, a valószínűségelmélet az $\mathcal{S}$ eseménytérre **megköveteli az alábbi tulajdonságok teljesülését** is:

- S tartalmazza az üres eseményt és a triviális eseményt is ($\{\}$ és Ω)
- S zárt az unioképzés alatt: $\alpha, \beta \in S \Rightarrow \alpha \cup \beta \in S$
- S zárt a komplemensképzés alatt (így a többi Bool-műveletre is teljesül a zártság): $\alpha \in S \Rightarrow \Omega - \alpha \in S$

Vagyis a dobókockás példánál maradva:

- Ha a páratlan számú eseményt belevesszük $\mathcal{S}$-be, akkor a páros számú eseményt is bele kell – mert ha az egyik valószínűségéről tudunk beszélni, akkor a negáltról is tudnunk kell
- Ahogy azokat az eseményeket is, amikor egy számot se kapunk (üres esemény), illetve amikor bármelyiket (triviális esemény)

Az így definiált $\mathcal{S}$ **eseménytér** egyes eseményeihez is (származtatott, de ettől még) valószínűségeket tudunk társítani.

Valószínűségi eloszlások

Egy $(\Omega, \mathcal{S})$ felett értelmezett P **valószínűségi eloszlás** az $\mathcal{S}$-ben levő események és valós értékek közötti leképezést határoz meg az alábbi feltételek mellett:

- Minden valószínűség nemnegatív: $\forall \alpha \in \mathcal{S} : P(\alpha) \geq 0$
- A triviális esemény valószínűsége 1: $P(\Omega) = 1$
- Nem átfedő unio: $\alpha, \beta \in \mathcal{S}, \alpha \cap \beta = \emptyset \Rightarrow P(\alpha \cup \beta) = P(\alpha) + P(\beta)$

Ebből a 3 axiomából sok minden levezethető. Például az üres halmaz valószínűsége 0, mert nincs közös eleme omegával, ezért:

$$
\begin{aligned}
&1 = P(\Omega) = P(\Omega \cup \emptyset) = \\
&P(\Omega) + P(\emptyset) \Rightarrow \\
&P(\emptyset) = 0
\end{aligned}
\tag{1.1}
$$

Ugyanígy, két esemény uniója és metszete (általános esetben):

$$
\begin{aligned}
&P(\alpha \cup \beta) = P(\alpha) + P(\beta - (\alpha \cap \beta)) \\
&P(\beta) = P(\beta - (\alpha \cap \beta)) + P(\alpha \cap \beta) \Rightarrow \\
&P(\alpha \cup \beta) = P(\alpha) + P(\beta) - P(\alpha \cap \beta) \\
&P(\alpha \cap \beta) = P(\beta) + P(\alpha) - P(\alpha \cup \beta)
\end{aligned}
\tag{1.2}
$$

Mivel bármely α esemény valószínűsége felírható α és β illetve α és $\neg\beta$ uniójaként, ezért:

$$
P(\alpha) = P(\alpha, \beta) + P(\alpha, \neg\beta) \tag{1.3}
$$

Ezt általánosítva megkapjuk a **„law of total probability"-t (azaz magyarul a teljes valószínűség törvényét)**. Amikor van I darab β_i esemény – melyek közül az egyik és csak az egyik biztosan teljesül – akkor:

$$
P(\alpha) = \sum_{i=1}^{I} P(\alpha, \beta_i) \tag{1.4}
$$

Ezt a műveletet a teljes valósznűség partíciója feletti **marginalizálásnak** is nevezzük. (Ez mindössze az előző eredmény általánosítása).

Ha például az a kérdés, hogy amikor kétszer dobok egy kockát, mi a valószínűsége hogy összesen 11-et kapok, akkor összegezhető:

- P(az első dobás 1 és 11-et kapok) + ... + P(az első dobás 6 és 11-et kapok)
- Az első dobás muszáj, hogy valami legyen 1 és 6 között, így a teljesül az a feltétel, hogy ami felett marginalizálunk, a biztos esemény egy partíciója.

Ezt az elvet alkalmazzuk általánosan akkor is, amikor a tanuláshoz használt modell parametrikus. Tegyük fel, hogy a tanítóvektoraink egy $\mathbf{X}$ mátrixban megtalálhatóak, és minimalizálni akarjuk az azokra adott $\hat{y}$ kimenetek várható hibáját. Ehhez figyelembe kell vennünk hogy a modell θ paraméterei sem fixek – így, feltéve hogy a kapott és kívánt kimenet közötti „távolságot" a $\mathcal{L}$ függvény adja meg, a várható hiba:

$$
E(\text{hiba}|\mathbf{X}) = \int \mathcal{L}(y, \hat{y}) p(y|\mathbf{x}) dy = \int \mathcal{L}(y, \hat{y}) \left(\int p(y|\theta, \mathbf{x}) p(\theta|\mathbf{x}) d\theta \right) dy \tag{1.5}
$$

A valószínűség értelmezései

Intuitívan $P(\alpha)$ megmondja, hogy mennyire vagyunk biztosak benne, hogy α be fog következni. Ha $P(\alpha) = 1$, biztosak vagyunk benne, hogy be fog; ha viszont 0, akkor biztos hogy nem.

Ez a magyarázat ugyanakkor nem mondja meg, hogy mit jelentenek ezek a számok. Két gyakori értelmezés:

- Frekventista: milyen gyakran fordul elő az esemény (ha végtelen sokszor megismételjük)
  - Tapintható értelmezést jelent, könnyen értjük fizikai rendszereknél is, hogy mit jelent (pl. kockadobásnál)
  - Ugyanakkor arra nem jó, hogy „holnap esni fog", mivel nem tudjuk megismételni a holnapot végtelenszer
- Szubjektivista: mennyire hiszek benne szubjektívan
  - 50% a valószínűsége, hogy holnap esni fog (úgy gondoljuk, hogy ugyanannyira valószínű hogy igen, mint hogy nem)
  - Itt viszont az a gond, hogy senki sem tudja objektíven megmondani, mit jelent a szubjektív hit. Mi tart vissza attól, hogy az eső valószínűséget 0.8-ra tegyem, annak a valószínűséget pedig, hogy nem esik, 0.5-re? Semmi!
  - Ezen kívül a hitem változhat bizonyos dolgokban, függetlenül attól, hogy maguk a dolgok nem változtak.

Mivel mindkét értelmezés ugyanahhoz a matematikához vezet, szerencsére túl sokat nem kell foglalkoznunk azzal kérdéssel, hogy melyik a mérvadó. Viszont különböző helyzetekben eltérő értelmezés lesz hasznos.

1.2.3. Feltételes valószínűség

A gépi tanulásban gyakran alkalmazott „Bayesi" megközelítés inkább a szubjektivista megközelítést tükrözi. Pont ezért központi eleme a feltételes valószínűség: Mennyire fogok x dologban szubjektíven hinni, ha tudom, hogy y esemény fennáll?

Példaként vegyünk egy eloszlást a Gépi tanulás című tárgyat hallgató diákok felett. Tegyük fel, hogy a diákok intelligenciája és záró osztályzata a fő valószínűségi változóink. Legyenek a következő eseményeink:

$$\begin{aligned} \alpha &: \text{egy random diák jegye} = \text{5-ös} \\ \beta &: \text{egy random diák intelligenciája} = \text{kimagasló} \end{aligned} \tag{1.6}$$

Ezekhez az eseményekhez valószínűségek tartozhatnak, amik megmondják hogy mennyire hiszünk abban, hogy egy tetszőlegesen kiválasztott diák jegye 5-ös, vagy hogy kimagaslóan intelligens.

De mi történik, ha egy diákról már tudjuk, hogy 5-ös a jegye? Ekkor természetesen változhat a szubjektív megítélésünk az intelligenciájáról (és fordítva is, ha megtudjuk hogy intelligens)!

Ez egy nagyon gyakori kérdés-típus, és a ***feltételes valószínűség*** segít annak megválaszolásában, hogy α ismeretében β valószínűsége miként változik:

$$P(\beta|\alpha) = \frac{P(\alpha \cap \beta)}{P(\alpha)} \tag{1.7}$$

Bayesi szempontból az egyenlet bal oldala „alapabb" mennyiség, mint az együttes valószínűség – *vagyis az emberi tudás szerkezetéhez jobban illeszthető.* Miért?

Visszaszorozva látszik ugyan, hogy az együttes valószínűség hogyan néz ki:

$$P(\alpha \cap \beta) = P(\beta|\alpha)P(\alpha) \tag{1.8}$$

. . . de az előző változat kihangsúlyozza, hogy:

- Ha megtudom hogy α igaz, akkor nem hihetek kevésbé abban, hogy β igaz, mint amennyire korábban $\alpha \cap \beta$ bekövetkeztében hittem
- Ráadásul, minél inkább meglepő az α esemény, a két hiedelem közti különbség annál nagyobb lesz

A Bayesi értelmezés segít a feltételes függetlenség megértésében is:

- α és β feltételesen függetlenek, ha nincs hatással az egyik valószínűségére az, ha tudom, hogy a másik teljesül (ez egyébként szimmetrikus reláció):

$$\alpha \perp \beta \Leftrightarrow P(\alpha|\beta) = P(\alpha) \tag{1.9}$$

Ugyanakkor frekventista nézőpontból pedig a képlet helyességét jól meg lehet érteni:

- Az első egyenlet azt nézi, hogy az összes olyan esethez, melyekben α teljesül, hogyan aránylik az összes olyan eset, ahol α és β is teljesül. Ez az arány adja meg annak a valószínűségét, hogy ha α igaz, akkor β is az lesz.
- A második azt mondja, hogy az együttes valószínűség azt jelenti, hogy megnézzük az egyik esemény valószínűséget, és (ezen az eseményen belül) megszorozzuk azon esetek arányával, amikor a másik esemény is teljesül. Ezt az esetet szokták láncszabálynak is nevezni.

$$P(\beta|\alpha) = \frac{P(\alpha \cap \beta)}{P(\alpha)} \tag{1.10}$$

$$P(\alpha \cap \beta) = P(\beta|\alpha)P(\alpha) \tag{1.11}$$

Például: ha tudom, hogy a diákok 10%-a kiemelten intelligens, és azt is tudom, hogy a diákok 15%-a kap 5-öst, ezzel a két számmal sok mindent nem csinálhatok, mert nem tudom, milyen átfedés van a két halmaz között.

De ha azt is tudom, hogy ez a 15% úgy aránylik, hogy a 6% kiemelten intelligens diák, 9% pedig nem, akkor:

$$\begin{aligned} P(\text{5-ös}|\text{kiem. intell.}) &= \frac{P(\text{kiem. intell. és 5-ös})}{P(\text{kiem. intell.})} = 0.06/0.1 = 0.6 \\ P(\text{kiem. intell.}|\text{5-ös}) &= \frac{P(\text{kiem. intell. és 5-ös})}{P(\text{5-ös})} = 0.06/0.15 = 0.4 \end{aligned} \tag{1.12}$$

1.2.4. Bayes-tétel

Ha pedig a láncszabályt kétféleképpen felírjuk, kijön a Bayes-tétel:

$$\begin{aligned} P(H,e) &= P(H|e)P(e) \\ P(H,e) &= P(e|H)P(H) \\ \Rightarrow P(H|e) &= \frac{P(e|H)P(H)}{P(e)} \end{aligned} \tag{1.13}$$

Itt a jelölések nem véletlenek: H a hipotézis, e pedig az evidencia (bizonyíték). A tétel azért hasznos, mert az egyenlet bal oldalát közvetlenül általában nehéz számolni.

A jobb oldalát viszont annál könnyebb. Pl., ha H = *a betegnek tüdőgyulladása van*, e = *a beteg sápadt és köhög*. Sokkal könnyebb megmondani, hogy „ha tüdőgyulladása lenne, mekkora eséllyel látnánk ezeket a tüneteket", mint fordítva. Azt is tudjuk, hogy a tüdőgyulladás, illetve a sárgaság/köhögés milyen önmagukban mennyire gyakori jelenségek.

Ez a fajta következtetés („tudásfrissítés") olyan gyakori, hogy a tagoknak külön nevük is van:

- Az egyenlet bal oldala a ***posterior valószínűség***
- Jobb oldalon a számláló első tagja a **(likelihood) valószínűség** – ez olyan, mint egy „hihetőségi" mérce
- Jobb oldalon a számláló második tagja a ***előzetes (prior) valószínűség***

$$\Rightarrow P(H|e) = \frac{P(e|H)P(H)}{P(e)} \tag{1.14}$$

Látjuk, hogy a posteriort úgy kapjuk, hogy összeszorozzuk a priort a likelihood-dal (és elosztjuk egy olyan taggal, ami minden hipotézis felett egyforma; magyarán a nevező csak skálázást végez és arról gondoskodik, hogy az összes lehetséges posterior hipotézis összege 1 legyen).

Ha kezdetben minden hipotézisben egyformán hiszünk, $P(H)$ értéke minden esetben ugyanaz, így a likelihood és a posterior ugyanazt jelentik, általános esetben ez azonban nincs így. Sajnos mivel a posterior pontos értékét csak normalizálással tudjuk kiszámolni, és a gyakorlati feladatokban általában számtalan lehetséges hipotézis (vagy magasabb szinten: modell) közül kell majd választanunk, ezért sok esetben akár szimulációra alapuló közelítő megoldásokat kell alkalmazni (pl. Monte Carlo módszerek).

1.2.5. Várható érték

Egy X valószínűségi változó fölött értelmezett $g(x)$ függvény **várható értéke**:

$$E_P[g] = \sum_x g(x)P(X = x) \tag{1.15}$$

Ugyanez feltételes esetben is definiálható (pl. feltéve, hogy tudom, hogy Y=y, mi g várható értéke?):

$$E_P[g|y] = \sum_x g(x)P(X = x|Y = y) \tag{1.16}$$

Ha a változók értékkészlete folytonos, akkor szumma helyett integrálni kell.

Fontos az is, hogy a képletből adódóan az összeg várható értéke egyenlő a várható értékek összegével – ugyanis a $P(X = x)$ valószínűséget az összeg fölött „beszorozva" a szumma két azonos formájú tagra bontható.

1.2.6. Variancia és kovariancia

Kitüntetett jelentőségű az $(X-E(X))^2$ függvény várható értéke, amit **varianciának (szórásnégyzetnek)** nevezünk.

$$\begin{aligned} Var(X) =& E[(X - E(X))^2] = E[X^2 - 2X * E[X] + E[X]^2] = \\ & E[X^2] - 2E[X * E[X]] + E[E[X]^2] = \underline{E[X^2] - E[X]^2} \end{aligned} \tag{1.17}$$

Ugyanígy két változó esetén az $(X - E(X)) * (Y - E(Y))$ függvény várható értéke a **kovariancia**:

$$\begin{aligned} Cov(X,Y) = & E[(X - E(X)) * (Y - E(Y))] = \\ & E[XY] - E(E[X]Y) - E(XE[Y]) + E(E(X)E(Y)) = \\ & E[XY] - 2E[X]E[Y] + E[X]E[Y] = \\ & \underline{E[XY] - E[X]E[Y]} \end{aligned} \tag{1.18}$$

Amennyiben X és Y függetlenek egymástól, a szorzatuk várható értéke egyenlő lesz a várható értékük szorzatával, így a kovariancia 0.

Lineáris függvény varianciája

Nézzük most meg, hogy mi X lineáris függvényének varianciája?

$$\begin{aligned} Var(aX+b) &= E[(aX+b)^2] - E[aX+b]^2 = \\ &E[a^2X^2] + E[2abX] + E[b^2] - (aE[X]+b)^2 = \\ &a^2E[X^2] + \cancel{2abE[X]} + \cancel{b^2} - a^2E[X]^2 - \cancel{2abE[X]} - \cancel{b^2} = \\ &a^2(E[X^2] - E[X]^2) = \underline{a^2Var(X)} \end{aligned} \tag{1.19}$$

Éppen ezért gyakran használjuk a variancia gyökét. Ez a **szórás**, vagy angolul **standard deviation**. Jelölése X valószínűségi változó esetén σ_X.

Lineáris függvény kovarianciája

Nézzük ugyanígy a lineáris függvény kovarianciáját!

$$\begin{aligned} Cov(aX+b, cY+d) &= \\ &E[(aX+b-E[aX+b])(cY+d-E[cY+d])] = \\ &E[(aX-aE[X])(cY-cE[Y])] = \\ &E[acXY - acXE[Y] - acYE[X] + acE[X]E[Y]] = \\ &acE[XY] - 2acE[X]E[Y] + acE[X]E[Y] = \\ &ac(E[XY] - E[X]E[Y]) = \underline{acCov(X,Y)} \end{aligned} \tag{1.20}$$

Vagyis hasonlót kapunk, mint a variancia lineáris transzformációja esetén, csak most a^2 helyett ac jelenik meg, mint együttható.

Transzlációra pedig ugyanúgy érzéketlen a kovariancia is, mint a variancia! (ráadásul itt a két változót különböző mértékben eltolhatjuk).

Két valószínűségi változó összegének varianciája

Nézzük most meg, hogyan alakul két valószínűségi változó, X és Y összegének varianciája:

$$\begin{aligned} Var(X+Y) &= E[(X+Y-E(X+Y))^2] = \\ &E[((X-E(X)) + (Y-E(Y))^2] = \\ &E[(X-E(X))^2] + E[(Y-E(Y))^2] + 2 * E[(X-E(X))(Y-E(Y))] = \\ &\underline{Var(X) + Var(Y) + 2 * Cov(X,Y)} \end{aligned} \tag{1.21}$$

Amennyiben X és Y függetlenek egymástól, a kovarianciájuk 0 lesz és így az összegük varianciája egyenlő lesz a varianciájuk összegével.

1.2.7. Markov- és Chebyshev-egyenlőtlenség

A legritkább esetben fog X olyan értéket felvenni, ami a szórásnak többszörösére van a várható értéktől. A szórás így egy normalizált távolságmérték a várható értéktől.

Erről a Chebyshev-egyenlőtlenség „gondoskodik". Először vegyünk egy olyan X valószínűségi változót, ami csak pozitív lehet. Ekkor:

$$\begin{aligned} t * P(X \geq t) = t \int_{x=t}^{\infty} P(x) = \int_{x=t}^{\infty} tP(x) \leq \\ \leq \int_{x=t}^{\infty} xP(x) \leq \int_{x=0}^{\infty} xP(x) = E[X] \\ \Rightarrow P(X \geq t) \leq \frac{E[X]}{t} \end{aligned} \tag{1.22}$$

Ez az ún. Markov-egyenlőtlenség. Ahogy t egyre nagyobb, ez a valószínűség egyre kisebb lesz.

Ha most X helyébe az $(X-E[X])^2$ valószínűségi változót írjuk, ami kétségtelenül biztos hogy pozitív, akkor megkapjuk a Chebyshev-egyenlőtlenséget:

$$\begin{aligned} P((X - E[X])^2 \geq k^2\sigma_X^2) \leq \frac{E[(X - E[X])^2]}{k^2\sigma_X^2} = \frac{1}{k^2} \\ \Rightarrow P(|X - E[X]| \geq k\sigma_X) \leq \frac{1}{k^2} \end{aligned} \tag{1.23}$$

A két sor bal oldala ugyanaz, mert ha az alsó teljesül, akkor a felső is, és fordítva. Ez az eredmény azt jelenti például, hogy kevesebb, mint 25% annak az esélye, hogy X mintavételezésekor távolabb leszünk a várható értéktől, mint a szórás kétszerese.

1.2.8. Korrelációs együttható

Két változó **korrelációs együtthatóját** úgy kapjuk, hogy kovarianciájukat elosztjuk a két változó szórásának szorzatával:

$$\rho(X, Y) = \frac{Cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - E[X]^2}\sqrt{E[Y^2] - E[Y]^2}} \tag{1.24}$$

Vegyük észre, hogy:

- Ha a két változó független, a számlálóban lévő kovariancia értéke 0, így a korrelációjuk is 0 lesz.
- Másrészről a korreláció abszolút értéke sose lesz nagyobb, mint 1:

$$|Cov(X,Y)| \leq \sqrt{Var(X)Var(Y)} \tag{1.25}$$

Ugyanis:

$$\begin{aligned} 0 \leq Var(tXY) = t^2 Var(X) + Var(Y) - 2tCov(X,Y) \\ \Rightarrow f(t) = t^2 Var(X) - 2tCov(X,Y) + Var(Y) \geq 0 \end{aligned} \tag{1.26}$$

Mármost, ez csak akkor lehet t értékétől függetlenül pozitív, ha felfelé nyíló „vidám" paraboláról van szó, amely legfeljebb egy (de az is lehet, hogy 0) helyen metszi a t tengelyt. Ez ekvivalens azzal, hogy a diszkrimináns kisebb, vagy egyenlő 0-nál:

$$\begin{aligned} b^2 - 4ac \leq 0 \\ 4Cov^2(X,Y) - 4Var(X)Var(Y) \leq 0 \\ Cov^2(X,Y) \leq Var(X)Var(Y) \end{aligned} \tag{1.27}$$

ahonnan be is láthatjuk az eredeti állítást.

Ha most X-et és Y-t 0-ra „helyezzük" (kivonjuk belőlük a várható értékeket), sem a két változó varianciája, sem a kovarianciájuk nem fog változni! Ezért ha ezt megtesszük, a korrelációs együttható egyszerűsíthető:

$$\begin{aligned} \rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{Cov(X - E[X], Y - E[Y])}{\sqrt{Var(X - E[X])Var(Y - E[Y])}} = \\ \frac{Cov(\tilde{X},\tilde{Y})}{\sqrt{Var(\tilde{X})}\sqrt{Var(\tilde{Y})}} = \frac{E[\tilde{X}\tilde{Y}] - E[\tilde{X}]E[\tilde{Y}]}{\sqrt{E[\tilde{X}^2] - E[\tilde{X}]^2}\sqrt{E[\tilde{Y}^2] - E[\tilde{Y}]^2}} = \\ \frac{E[\tilde{X}\tilde{Y}]}{\sqrt{E[\tilde{X}^2]E[\tilde{Y}^2]}} \end{aligned} \tag{1.28}$$

Ráadásul a skálázásra sem érzékeny a korreláció. Ugyanis:

$$\rho(aX, bY) = \frac{Cov(aX, bY)}{\sigma_{aX}\sigma_{bY}} = \frac{abCov(x,y)}{\sqrt{a^2 Var(X)}\sqrt{b^2 Var(Y)}} = \rho(X,Y) \tag{1.29}$$

Ez azt jelenti, hogy két változó szórása is normalizálható (ilyenkor egy konstanssal – ami a szórás – osztunk): a korreláció ettől nem fog változni. A gyakorlati megvalósításokban sokszor fogunk élni ezzel a lehetőséggel: a tanítóadatok dimenzióinak átlagát 0-ra állítjuk be, szórásukat pedig 1-re.

Ezzel a közöttük levő lényegi összefüggéseket (mint korreláció) nem befolyásoljuk. Ami viszont jó, hogy a tanulóalgoritmusaink így (alapból) azonos súllyal veszik majd figyelembe az adathalmaz különböző dimenzióban olvasható értékeket.

1.3. A tanulás fajtái

1.3.1. Felügyelt tanulás

Felügyelt tanulás esetén jelöljük az adott $\mathbf{x}$ bemenetre adható lehetséges válaszok feletti eloszlást $P_{\mathcal{X}}(y|\mathbf{x})$-el! Legyen továbbá:

- $\mathbf{x}$ a tanítóminta vektor-reprezentációja
- y a kategória / címke (bináris osztályozás esetén y lehet 0 vagy 1)
- $\mathcal{X}$ a tanítóminták halmaza
- A feltétel azt jelzi, hogy adott y érték valószínűsége függ a mintától, és függ a tanítóhalmaztól. Implicit módon függ a θ választott modelltől is (de ezt sokszor nem jelöljük)

Ekkor például a maximum a posteriori (MAP) jelentése:

$$\hat{y} = argmax_{c=1}^{C} P_{\mathcal{X}}(y = c|\mathbf{x}) \tag{1.30}$$

Rengeteg alkalmazás, például a spam-szűrés, arc-felismerés, stb. felügyelt tanulási problémára vezethető vissza.

Ha C értéktartománya diszkrét, ***osztályozásról*** beszélünk. Ha folytonos, ***regresszióról*** (egyébként a kettő ugyanaz).

1.3.2. Nem felügyelt tanulás

Nem felügyelt tanulás esetén ezzel szemben a keresett eloszlás $P_{\mathcal{X}}(\mathbf{x})$. Legyen továbbá:

- $\mathbf{x}$ a tanítóminta vektor-reprezentációja
- $\mathcal{X}$ a tanítóminták halmaza

Nem felügyelt tanulás esetén nem tudjuk, hogy mi a helyes válasz, ezért a cél inkább a tanítómintákban levő „struktúra" felfedezése (**knowledge discovery**)

Ez is nagyon fontos. Sőt! Nem felügyelt tanulással lehetséges például:

- Adatok automatikus klaszterezése (eleinte persze azt sem tudjuk, hány klaszter van, ez a felhasznált modelltől függ)
  - A csillagászatban pl. új csillagtípusokat fedezhetnek fel nem felügyelt tanulással
  - Az üzleti ágazatokban a vásárló-típusok klaszterezésével hatékonyabb reklám-tevékenység lehetséges.

Lehetséges továbbá:

- Látens tényezők felfedezése (dimenzió-redukció).
  - Például arcképes fényképek magas dimenzionalitásúak, de lehetséges, hogy a képek közötti különbséget nagyban leírja néhány paraméter – mint pl. a megvilágítás, az ember pozíciója a képen, illetve maga az hogy melyik az az ember, akiről a kép készült (különösen akkor, ha csak néhány emberről található sok kép az adatbázisban)
  - Szöveges dokumentumok elemzésekor a dokumentum témája (sport, jog, politika...) látens tényező lehet, ami nagyban befolyásolhatja a szóhasználatot, a stílust.
- Gráf struktúra felfedezése – például ha a gráf szerkezete valamilyen adatok közötti korrelációt tükröz.
  - Például: milyen változóktól függ leginkább, hogy mikor lesz közlekedési dugó az M1-en? Ezek a változók mitől függnek leginkább? Stb.
- Hiányos mátrixok kitöltése
  - Pl. hiányzik egy kép pixeleinek egy része, töltsük ki!
  - Pl. Netflix: jön egy új felhasználó, tudjuk hogy A és B filmet szerette, melyik filmeket szeretheti még?

A nem felügyelt tanulás sokszor nehezebb, mint a felügyelt, mert a megtanulandó eloszlásban egy vektor és nem egy skalár található, magyarán a nem felügyelt esetben egy sokváltozós valószínűségi eloszlást kell megtanulni!

Ennek ellenére fontos, mert vélhetően mi magunk is gyakran implicite, nem felügyelt módon tanuljuk meg az élet alapvető összefüggéseit. Geoff Hinton, a mesterséges neurális hálók egyik úttörője mondta például, hogy:

> „*When we're learning to see, nobody's telling us what the right answers are – we just look. Every so often, your mother says that's a dog', but that's very little information. You'd be lucky if you got a few bits of information – even one bit per second – that way. The brain's visual system has* 10^{14} *neural connections. And you only live for* 10^9 *seconds. So it's no use learning one bit per second. You need more like* 10^5 *bits per second. And there's only one place you can get that much information: from the input itself*"

Nem véletlen, hogy egyre népszerűbb, és egyre több okos megoldás születik a nem felügyelt tanulás problémáira.

1.3.3. Generatív és diszkriminatív tanulás

Ha egy osztályozó valójában a bemenet és kimenet együttes $P(y, \mathbf{x})$ valószínűségét tanulja meg, akkor marginalizálással meg tudjuk határozni a prior ($P(y)$) értékét, a kettő hányadosaként pedig

a likelihood ($P(\mathbf{x}|y)$) tagot is. Ez utóbbi eloszlásból mintavételezve gyakorlatilag a címkék alapján tudunk valószínű bemeneteket is generálni, ezért ezt a fajta osztályozót **generatív osztályozónak** is szokás nevezni.

Ezzel szemben, ha az osztályozó „csak" azt tanulja meg, hogy a kimenet hogyan függ a bemenettől ($P(y|\mathbf{x})$), akkor a címkéket vagy modelleket megkülönbözteti ugyan, de nem tud szintetikus bemeneteket generálni, ezért **diszkriminatív** modellnek nevezzük.

Általában a generatív osztályozók rugalmasabbak, ugyanis a diszkriminatív modellek nem tudnak hiányos adatokkal mit kezdeni (nem tudnak szintetikus adatot generálni), és sokszor lassabban is illeszthetőek az adatokra, mivel ez sokszor egy bonyolultabb optimalizációs probléma megoldását igényli.

Bizonyos esetekben azonban a diszkriminatív osztályozók preferáltak – például mert az adatok preprocesszálására kevésbé érzékenyek. Ha pl. $\mathbf{x}$ helyett $\Phi(\mathbf{x})$-et használunk, egy generatív modell alapvető feltételezései (például bizonyos esetekben hogy az adatok normál-eloszlásúak) sérülhetnek, míg egy diszkriminatív modell, amely csak a kategóriák közötti határmezsgyével foglalkozik, sok esetben kevésbé lesz érzékeny az ilyen transzformációkra.

Sokszor jobban is kalibráltak a kimeneti valószínűségek, kevésbé extrémek a diszkriminatív, mint a generatív esetben, ugyanis egy generatív modell gyakrabban képes adatokkal alá nem támasztott működést produkálni, ha a paramétertér egy olyan részéből generál bemeneteket, amely az adathalmazban alulreprezentált.

1.3.4. Parametrikus és nemparametrikus tanulás

Fontos kérdés, hogy ahogy egyre több tanítómintát használok, ugyanúgy fix számú paraméterem lesz-e, vagy egyre nőni fog az eltárolandó paraméterek száma?

A korábbi esetben a modell **parametrikus**, és a tanulás célja a modell paramétereinek olyan beállítása, amely az adatokhoz illeszkedik.

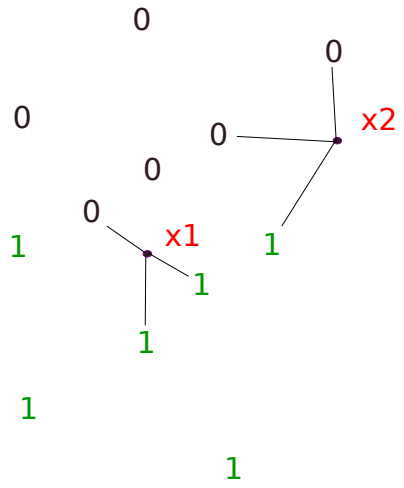
Az utóbbi esetben a modell **nem parametrikus**, és a cél inkább az eddig látott összes tanítóminta hatékony „tárolása".

A parametrikus modellek tanítása/használata gyorsabb, viszont sokszor erős feltételezést tükröznek az adatokról! A nem parametrikus modellek rugalmasabbak, de nagy adathalmazok esetén könnyen kezelhetetlenné válnak.

1.3.5. Példák

Fentiek alapján rengeteg féle modell és rengeteg féle algoritmus létezik. Néhány példa:

- K nearest neighbors: felügyelt, diszkriminatív, nem parametrikus
- Lineáris regresszió: felügyelt, diszkriminatív, parametrikus
- Logisztikus regresszió: felügyelt, diszkriminatív, parametrikus
- Naív Bayes osztályozás: felügyelt, generatív, parametrikus



1.2. ábra. K nearest neighbors módszer vizualizációja

- Kernel Density Estimation (KDE): nem felügyelt, generatív, nem-parametrikus
- Energia alapú modellek, mint pl. a Korlátozott Boltzmann-gépek (Restricted Boltzmann Machine, RBM): nem felügyelt, generatív, fix modellnél parametrikus

Ezek közül most néhányat nézzünk át!

Példa: K nearest neighbors (KNN) módszer és a dimenzionalitás átka

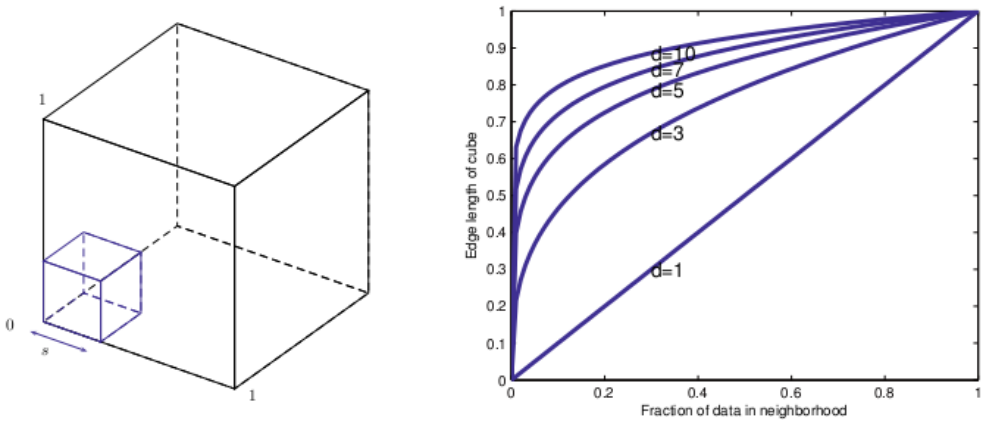
A **K nearest neighbor (KNN) osztályozás**: döntés az $\mathbf{x}$ bemenethez legközelebb eső K darab szomszéd alapján.

Megszámolhatjuk, hogy adott ponthoz melyik a K darab legközelebbi osztály, és az arányok valamilyen valószínűség-becslést adhatnak. Formálisan:

$$P_{\mathcal{X}}(y = c|\mathbf{x}, K) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x}, \mathcal{X})} \mathbb{I}(y_i = c) \tag{1.31}$$

Például az ábrán x_1 66%-os valószínűséggel az 1-es kategóriába tartozik, x_2 ugyanígy a 0-ásba.

Ezt a módszert **memória-alapú** tanulásnak, vagy **eset-alapú** tanulásnak is szokás nevezni. A módszer nem-parametrikus, hiszen az összes tanítómintát meg kell jegyeznünk. Ugyanakkor a módszer



1.3. ábra. Dimenzionalitás átka szemléltetése

diszkriminatív is, hiszen tipikus pontokat nem tudunk a modell alapján generálni, a modell nem egy együttes eloszlást fejez ki, egyszerűen csak új példák kategorizálására alkalmas.

Ha $K = 1$, ún. Voronoi-tesszelációról beszélhetünk. Ekkor egy $V(\mathbf{x}_i)$ partícióhoz azok, és csak azok a pontok tartoznak, amelyek $\mathbf{x}_i$-hez vannak a legközelebb

A memória-alapú osztályozók legnagyobb problémája, hogy magas dimenzionalitású pontok esetén nagyon nem hatékonyak. Hogy miért nem, azt a „***dimenzionalitás átka***" magyarázza, amely a gépi tanulásban egy alapvető fogalom.

Tegyük fel, hogy van egy D-dimenziós egységkockánk, és hogy olyan mintákat kell osztályoznunk, amelyek az egységkockában egyenletes eloszlás szerint helyezkednek el. Ezek modellezéséhez egy adott $\mathbf{x}$ pont körül olyan kockát (hiperkockát) növesztünk, hogy a tanítópontok valamilyen f hányada benne legyen (magyarán: ennyi közeli példával szeretnénk többségi szavazást csinálni).

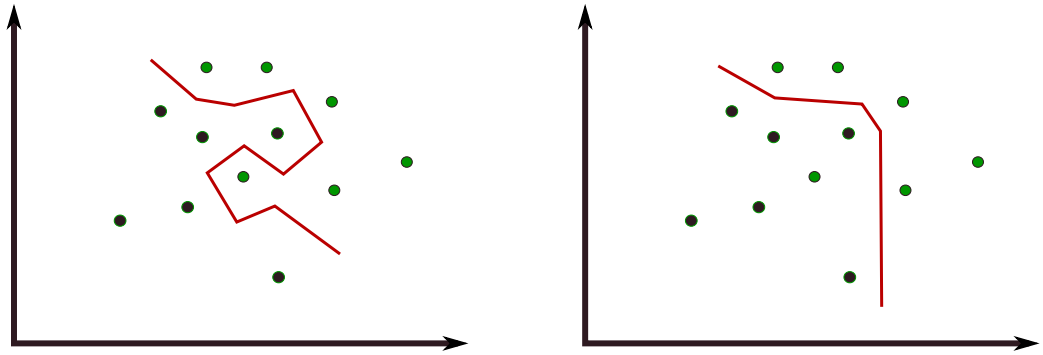
Mivel a minták eloszlása egyenletes, a kocka várható él-hossza $e_D(f) = f^{1/D}$ lesz. Például, ha két-dimenziós térben vagyunk, $D = 2$. Ha f értéke pedig 0.5, a négyzetek él-hosszának várható értéke $\sqrt{2}/2$ lesz.

Ugyanígy, ha $D = 10$ – tehát 10-dimenziós térben vagyunk – és az adatok 10%-a alapján szeretnénk döntést hozni, a tanítóminták körül minden dimenzióban az egységkocka 80%-át le kell fednünk! De ha csak az adatok 1%-ára támaszkodunk, akkor is $e_{10}(0.01) = 0.63$, ami nagyon nagy szám! Ez a módszer már nem annyira „lokális", hiába is nevezzük „nearest neighbor" alapúnak (ld. még 1.3. ábrát, amely ugyanezt szemlélteti)!

Fentiek alapján kijelenthető, hogy a nem-parametrikus megoldások nemcsak az eltárolandó adat-mennyiség miatt nehezen kezelhetőek, hanem azért is, mert sokdimenziós terekben az euklidészi értelemben vett „közelség" vagy „hasonlóság" fogalma nem úgy működik, mint ahogy intuitíve elvárhatnánk.

Vizsgáljuk még meg, hogyan függ az eredmény K-tól? Ez megint a variancia-bias dilemma!

- Ha K kicsi, nagyon szabdalt lesz a megtanult vektortér → magas variancia



1.4. ábra. Ha K nagy, akkor erősebben torzított modellt kapunk. Ha K kicsi, a nagyobb variancia felé mozdul el a modell.

- Ha K nagy, végül az egész tanítóhalmaz fölött egy maximumot veszünk → magas a bias, és nagyon sima lesz a megtanult határvonal.

Példa: A regressziós modellek mint parametrikus modellek

Parametrikus modellek esetén fix számú paraméter van (függetlenül a tanító halmaz méretétől), ezek értékét kell megtanulni. Ilyenkor eleve feltételezzük, hogy a parametrikus modell tényleg jó (**inductive bias**)

Gyakori példa a lineáris regresszió. A regresszió azt jelenti, hogy a kategóriák értéke folytonos. Lineáris regresszió esetén feltételezzük, hogy lineáris kapcsolat van $\mathbf{x}$ dimenziói és y értékek között (a gyakorlatban ez nem gond, mert $\mathbf{x}$-ben származtatott nemlineáris elemek is szerepelhetnek):

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon \tag{1.32}$$

ahol ϵ a fennmaradó hiba (miután sem a modell, sem a mintavételezés nem lesz feltétlenül tökéletes)

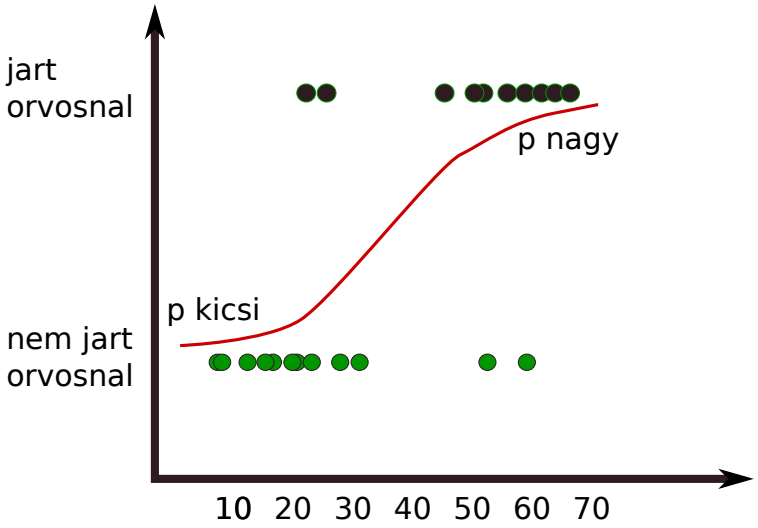
Másfajta regressziók is léteznek. Például fix számú kategória esetén igen népszerű a logisztikus regresszió (mivel fix a kategóriák száma, ez igazából osztályozás, de a lineáris regresszióhoz való hasonlósága miatt így nevezik)

A „szigmoid" szó azt jelenti, hogy „S-alakú", és a szigmoid (logisztikus) függvény így néz ki:

$$sigm(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}} \tag{1.33}$$

Ha x a végtelenhez tart, a függvény értéke 1-hez tart. Ha x a mínusz végtelenhez tart, a függvény értéke a 0-hoz tart. A két véglet között a függvény S-alakú.

Logisztikus regresszió esetén egy ilyen függvényt keresünk, melyet valószínűség-eloszlásként értelmezünk. Pl. két kategória esetén:



1.5. ábra. Példa logisztikus regresszióra.

$$P(y = 1|\mathbf{x}, \mathbf{w}) = sigm(\mathbf{w}^T\mathbf{x}) \tag{1.34}$$

A feladat ilyenkor megfelelő $\mathbf{w}$ paramétereket megtalálni. Például az 1.5. ábra megmondja, hogy adott életkorban milyen valószínűséggel megyünk el legalább egyszer orvoshoz (ez egy fiktív adatok alapján készült példa).

De nézzük, milyen megfontolásból alkalmazhatjuk a szigmoid függvényt regressziós célra – elvégre tetszőleges másmilyen összefüggést is el lehetne képzelni $\mathbf{w}$, $\mathbf{x}$ és y között! Osztályozásnál kézenfekvő ötlet lehetne például a lineáris osztályozás:

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x} \tag{1.35}$$

Ezzel két probléma van:

- A kimenet csak 0 és 1 közé eshet, de a lineáris függvény értékkészlete korlát nélküli
- Legtöbb valós példánál bizonyos szint fölött a bemenet azonos mértékű módosítása a kimeneti valószínűségen relatíve kisebb hatást fog gyakorolni.

A következő ötlet: legyen a valószínűség logaritmusa lineáris!

$$P(y = 1|\mathbf{x}, \mathbf{w}) = e^{\mathbf{w}^T\mathbf{x}} \tag{1.36}$$

. . . csakhogy az exponenciális függvény kimenete felfelé nem korlátos, így a probléma hasonló (csak az alsó korlátot értük el, hogy 0 legyen)

Ezért az ötletet kicsit módosítjuk: ami két kategória esetén lineáris lesz, az $\log P(y|\mathbf{x}, \mathbf{w})/(1 - P(y|\mathbf{x}, \mathbf{w}))$. Ez a hányados (amit **odds**-nak is nevezünk) bármilyen 0 és ∞ közötti értéket felvehet! Ekkor:

$$
\begin{aligned}
\log \frac{P(y|\mathbf{x},\mathbf{w})}{1-P(y|\mathbf{x},\mathbf{w})} &= \mathbf{w}^T\mathbf{x} \\
P(y|\mathbf{x},\mathbf{w}) &= (1-P(y|\mathbf{x},\mathbf{w}))e^{\mathbf{w}^T\mathbf{x}} \\
P(y|\mathbf{x},\mathbf{w})(1+e^{\mathbf{w}^T\mathbf{x}}) &= e^{\mathbf{w}^T\mathbf{x}} \\
P(y|\mathbf{x},\mathbf{w}) &= \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} = sgm(\mathbf{w}^T\mathbf{x})
\end{aligned}
\tag{1.37}
$$

Ez pont jól jön ki, mert a sigmoid függvény értékkészlete éppen 0 és 1 közötti! Ráadásul ez a modell nagyon jól interpretálható.

Tegyük fel pl., hogy $\mathbf{x}$ kétdimenziós, jelentése pedig hogy hány cigarettát szív el valaki egy nap, és hány percet fut egy nap; tegyük fel hogy ez alapján szeretnénk a tüdőrák valószínűségét megmondani. Ha a kapott $\mathbf{w}$ értéke $(1.3, -1.1)$, akkor minden újabb cigaretta elszívásával a tüdőrákhoz társított odds $e^{1.3}$-szorosára növekszik. Ugyanis:

$$
\begin{aligned}
\log \frac{P(y)}{1-P(y)} &= \mathbf{w}^T\mathbf{x} \\
\frac{P(y)}{1-P(y)} &= e^{1.3x_1-1.1x_2} \\
\frac{P(y)}{1-P(y)} &= e^{1.3x_1}e^{-1.1x_2}
\end{aligned}
\tag{1.38}
$$

1.4. Variancia-Bias Dilemma

A variancia és bias közötti kompromisszum matematikailag is levezethető az alábbi módon.

Tegyük fel, hogy az adataink egy $\mathbf{P}$ eloszlásból származnak. A bemenetek és kimenetek tehát így néznek ki: $(\mathbf{x}_i, y_i) \sim \mathbf{P}$.

Tegyük fel továbbá, hogy az $\mathcal{X}$ adathalmazunk N darab ilyen mintavételezett be-kimeneti párból áll.

Feltéve most, hogy regressziós problémán dolgozunk (magyarán: a kimeneti érték egy folytonos változó), az $y = f(x)$ modellünk hibáját az elvárt és a kapott kimenet valamilyen távolsága alapján definiálhatjuk. Hogyan tehetjük ezt meg?

Attól függően, hogy miként gondolkodunk a helyzetről, kétféle megközelítés is létezhet:

1. Ha az összes lehetséges input értéket vesszük alapul, a hiba kifejezhető így – feltéve, hogy a négyzetes hibát alkalmazzuk: $Error = \mathbf{E}_{\mathbf{P}}\left[(y_i - f(\mathbf{x}_i))^2\right]$. Ezt nevezhetjük **modell-független hibának**

2. Ha a hibával a mintavételezett pontokra helyeznénk a hangsúlyt, amely alapján a modellünket betanítottuk (más lehetséges mintavételezett pontokkal, és így más végső modellekkel összehasonlítva), a hibát így is írhatnánk: $Error = \mathbf{E}_{\mathcal{X}}\left[(y - f_{\mathcal{X}}(\mathbf{x}))^2\right]$. Ezt nevezhetjük **tanítómintafüggetlen hibának (TMFH)**.

Az első esetben az összes lehetséges input-output pár fölött vesszük a zárójelben levő kifejezés súlyozott átlagát, miközben az f modellt fixnek tekintjük; míg a második esetben egyetlen be-kimenet párunk van, de a súlyozott átlagot az összes lehetséges modell fölött nézzük.

A TMFH-t átlakítva az alábbiakra juthatunk:

$$\begin{aligned} TMFH &= \mathbf{E}_{\mathcal{X}}\left[(y - f_{\mathcal{X}}(\mathbf{x}))^2\right] \\ &= \sum_{i=1}^{I} p_i \left[y - f_{\mathcal{X}_i}(\mathbf{x})\right]^2 \end{aligned} \tag{1.39}$$

ahol $f(\mathbf{x})$ a betanított modell kimenete az adott bemenetre, amit $\hat{y}$-ként is jelölünk majd. Ezzel szemben y az elvárt kimenet, p_i pedig az i-edik, $\mathcal{X}_i$-vel jelölt adathalmaz valószínűségét jelöli, feltéve, hogy I különböző adathalmazon taníthatnánk be a modellt. (Az alábbiakban az i-t lehagyhatjuk a $\mathcal{X}_i$ jelölésből, tudván, hogy az adathalmaz sem fix).

Alakítsuk át a fenti képletet, hogy mélyebb megértésre juthassunk:

$$\begin{aligned} TMFH &= \sum_{i=1}^{I} p_i \left[y - \hat{y}\right]^2 \\ &= \sum_{i=1}^{I} p_i \left[y - E(\hat{y}) + E(\hat{y}) - \hat{y}\right]^2 \end{aligned} \tag{1.40}$$

Vegyük észre, hogy a tagokhoz hozzáadott és a belőlük kivont érték egy konstans, skalár érték. Most láthatjuk, hogy egy $(a+b)^2$ alakú kifejezésünk van, így a levezetés az alábbiak szerint folytatható:

$$\begin{aligned} TMFH &= \sum_{i=1}^{I} p_i \left[y - E(\hat{y}) + E(\hat{y}) - \hat{y}\right]^2 \\ &= \sum_{i=1}^{I} p_i \left[y - E(\hat{y})\right]^2 + \sum_{i=1}^{I} p_i \left[E(\hat{y}) - \hat{y}\right]^2 + 2\sum_{i=1}^{I} p_i (y - E(\hat{y}))(E(\hat{y}) - \hat{y}) \end{aligned} \tag{1.41}$$

(Ezt megtehettük, hiszen az összeg várható értéke egyenlő a várható értékek összegével!)

Mit jelent ez a 3 tagú összeg?

1.4.1. Bias tag

Az első tag az elvárt kimenet és a kapott kimenet várható értéke négyzetes különbségének várható értéke (erre az adott bemenetre). De milyen eloszlás fölött vesszük ezt a várható értéket? Természetesen a lehetséges tanítóhalmazok eloszlása fölött. Itt azonban meg kell jegyezni, hogy a kifejezés egyáltalán nem függ a tanítóhalmaztól! Ugyanis:

- y nem függ a tanítóhalmaztól, ez csak egy random módon választott be-kimeneti pár kimeneti változója, és
- $E(\hat{y})$ tulajdonképpen egy konstans érték a tanítóhalmaz szempontjából – ez az összes tanítóhalmazból származó összes lehetséges modell felett vett várható kimenet.

Éppen ezért az első tag ekvivalens ezzel:

$$[y - E(\hat{y})]^2 \tag{1.42}$$

...ami maga a négyzetes bias tag. Egyszerűen megfogalmazva, ez a tag az elvárt kimenet és a modellek fölötti átlag kimenet közötti különbséget fejezi ki.

1.4.2. Variancia tag

A második tag ellenben valóban egy véletlen változó várható értéke, a felhasznált tanítóhalmaz függvényében. Ebben az esetben arra vagyunk kíváncsiak, hogy mi a kapott kimenet szórásnégyzete a lehetséges modellek fölött. Ez az érték akkor lesz nagy, ha különböző, egyaránt valószínű tanítóhalmazokkal betanított modellekre nagyon eltérő kimenetet kapunk.

1.4.3. A harmadik tag értéke 0

Végül az utolsó tag 0-ával egyenlő. Emlékezzünk vissza, hogy az eső tényező egy konstans. Ami ezután megmarad (miután az első tagot kivittük a szumma elé), az egy valószínűségi változó és várható értéke közötti különbség várható értéke, ami éppen 0.

1.4.4. Mit jelent mindez?

A fentiek alapján a modell-független hibát csak úgy tudjuk csökkenteni (a modell befolyásolásával), ha vagy a bias-t, vagy a varianciát lecsökkentjük (vagy mindkettőt).

Ez utóbbi azonban nem lehetséges. Gondoljuk át, hogy mit jelentene, ha mindegyiket egyszerre tudnánk csökkenteni!

A variancia csökkentése azt jelentené, hogy függetlenül a felhasznált tanítóhalmaztól, alig lenne variáció a modell kimenetében (adott bemenetre). Extrém esetben a 0 variancia azt jelentené, hogy mindegyik tanítóhalmazra/modellre ugyanazt a kimenetet kapnánk – a teljesítmény így egyáltalán nem függne a tanítóhalmaztól!

Ebben az esetben viszont nem lenne módunk arra, hogy az adatokkal bármit javítsunk a modell teljesítményén! Ez pedig azt mutatná, hogy a modellünk nem veszi figyelembe a tanítóhalmazt, vagyis magas a torzítása (bias).

Más szempontból azt is mondhatjuk, hogy a bias és variancia a következőket mondják:

„Én, a bias, le szeretném magamat csökkenteni, ezért 100%-ig az adatokra figyelek oda, hogy megfelelő tanítóhalmazzal a lehető legjobban megközelítsem az adatok eloszlását. Nincsenek nekem fixa ideáim, csak az adatokra szeretnék figyelni!”

Ezzel szemben a variancia azt mondja:

„Én, a variancia le szeretném magamat csökkenteni, ezért olyan viselkedést szeretnék produkálni, ami nem annyira függ az adatoktól. Tulajdonképpen az adatoktól függetlenül ugyanúgy szeretnék viselkedni”

Magyarán, általában véve azok a mechanizmusok, amelyek a bias-t csökkentenék, éppen a variancia csökkentése ellen dolgoznak – és fordítva. Ezért beszéhetünk variancia-bias dilemmáról.

1.5. Modellek kiértékelése: tipikus költségfüggvények

Azt a lépést, amikor gyakorlati szempontból megnézzük, hogy egy modell tanításához használt mely attribútumok vagy jellemzők a legmegfelelőbbek az adott feladat szempontjából, **feature engineering**-nek szokás nevezni. Egy data scientist („adatkutató”) idejének $80-90\%$-át is kitöltheti ez a lépés! Ehhez viszont tudni kell, hogyan értékeljük ki egy adott modell teljesítményét.

1.5.1. Legkisebb négyzetes hiba

A variancia-bias tárgyalásánál láthattuk, hogy egy jelöltként felmerülő modell teljesítményét folytonos kimenet esetén jellemezhetjük a legkisebb négyzetes hibával (MSE), vagy annak négyzetgyökével (RMSE):

$$MSE = \left(\frac{1}{M}\right) \sum_{m=1}^{M} (y^{(m)} - f(x^{(m)}))^2 \tag{1.43}$$

ahol f modellünket szeretnénk kiértékelni, y az elvárt kimenet értéke az M különböző adatpont esetén, ahol m 1-től M-ig terjed.

1.5.2. Hibák átlagos abszolútértéke

Egy másik alternatíva a hibák átlagos abszolútértéke (MAE):

$$MAE = \left(\frac{1}{M}\right) \sum_{m=1}^{M} |y^{(m)} - f(x^{(m)})| \tag{1.44}$$

Ha viszont a kimenet nem folytonos hanem kategorikus, a félre-osztályozásra (misclassification) más mértéket kell találnunk. Itt mérvadó lehet például a misclassification rate (az összes adatpont fölött összegezve):

$$MCR_{\mathcal{X}}(f) = \left(\frac{1}{M}\right) \sum_{m=1}^{M} I_{y^{(m)} \neq f(x^{(m)})} \tag{1.45}$$

1.5.3. Keresztentrópia

További népszerű lehetőség az ún. ***keresztentrópia (cross entropy loss)*** használata:

$$CrossEntropy_{\mathcal{X}}(f) = -\left(\frac{1}{M}\right) \sum_{m=1}^{M} \sum_{m=c}^{C} y_c^{(m)} \log f(x_c^{(m)}) \tag{1.46}$$

amennyiben C különböző kategóriánk van, és $y_c^{(m)}$ az m-edik bemenethez tartozó kimeneti vektor c. eleme. Itt minden kimenet egy C-elemű vektor, melynek elemei ideálisan az adott kategória valószínűségeként értelmezhetőek. Ez az értelmezés megmagyarázza a keresztentrópia működését is: azon dimenziók valószínűségeit „súlyozzuk", melyek tanítópontonként a helyes kimenethez tartoznak, maga a loss érték pedig a kapott kimenet logaritmusának mínusz 1-szerese (ami 0-ához közeli valószínűségek esetében a végtelenhez tart, 1-hez közeli valószínűségek esetében pedig a 0-ához közelít).

1.5.4. Gini impuritás

Kategorikus osztályozók esetében az ún. **Gini-féle impuritás** (***Gini impurity***) is hasznos tud lenni. Az e mögötti alapelvek a következők:

- Ha feltesszük, hogy random osztályozónk van és tetszőleges pontot választunk a tanítóhalmazból, feltehetjük azt a kérdést, hogy mekkora lesz az esélye, hogy ezt a pontot félreosztályozzuk?
- Az erre adott válasz tulajdonképpen a szó szoros értelmében nem egy konkrét osztályozóról szól, hanem az adatok eloszlásáról (ezért impuritás)... ugyanakkor, amikor arra a kérdésre keressük a választ, hogy az adatok mely dimenziója mentén milyen értéknél vagy értékeknél bontsunk egy adathalmazt részkategóriákra, az újonnan (feltételesen) kapott partíciók puritás-vizsgálatához ez egy hasznos metrika tud lenni!

Nézzük át ezért a Gini-impuritás tulajdonságait közelebbről. Tegyük fel most is, hogy C-féle címke van. legyen f_c azon pontok aránya (az összes tanítópont között), amelyek az c kategoriához tartoznak; és legyen $f_{c'|c}$ a c kategóriájú tanítópontoknak az az aránya, amely pontokat c helyett az osztályozó c' kategóriába sorol. Ekkor a Gini-impuritás:

$$I_G(f) = \sum_{c=1}^{C} f_c \sum_{c'=1, c' \neq c}^{C} f_{c'|c} = \sum_{c=1}^{C} f_c (1 - f_{c'=c|c}) \tag{1.47}$$

Ehhez a magyarázat: az első szummában minden kategóriára végignézzük, hogy a random pont milyen eséllyel fog az adott, c kategóriába tartozni. Ezzel a valószínűséggel megszorozzuk azt a

valószínűséget, hogy az adott pontot félreosztályozza a random osztályozó. Láthatjuk azt is, hogy a második szumma átalakítható az 1-mínuszos tagra, mivel ha a $c' = c$ esetet is figyelembe vennénk, akkor éppen a teljes eseményt (1 valószínűség) kapnánk.

Vegyük észre azonban, hogy az $f_{c'=c|c}$ éppen azoknak a c-kategóriájú pontoknak az aránya, amelyeket a véletlen osztályozó éppen a c kategóriába osztályoz. Ez egy 0 és 1 közötti valószínűség lesz, ami azt jelenti, hogy a szumma tagjainak értéke 0 és f_c közötti lesz, vagyis a teljes impuritás sohasem lehet nagyobb, mint 1! Ezen kívül az összes tag pozitív, ezért az impuritás végső soron 0 és 1 közé esik. Minél kisebb az impuritás, annál homogénebbek az adatok (amelyek fölött még egy random osztályozó is jól teljesítene).

Éppen ezért a Gini-féle impuritás jól használható annak eldöntésére, hogy pl. egy döntési fában egy-egy csomópont mennyire végez 'hasznos' munkát (ld. később)

1.5.5. Információnyereség (Kullback-Leibler divergencia)

A teljes osztályozónkról többet mond az ún. *information gain* (más néven: ***Kullback-Leibler (KL)-divergencia***). Ehhez definiálnunk kell az *entrópia* fogalmát (eloszlásokra):

$$I_E(\mathcal{X}) = -\sum_{c=1}^{C} f_c \log_2 f_c \tag{1.48}$$

Ha f_c értéke 1, az adott tag 0 lesz... és hasonlóan adódik kb. ha f_c értéke 0-hoz közelít (pont 0 nem lehet a logaritmus miatt).

Beszélhetünk feltételes entrópiáról is: ilyenkor további feltételezésekkel élhetünk a c kategóriába tartozó egyedekkel kapcsolatban (például, hogy egy T predikátum igaz rájuk):

$$I_E(\mathcal{X}|T(x)) = -\sum_{c=1}^{C} f_c \log_2 \frac{f_{c\&T}}{f_c} \tag{1.49}$$

Az entrópia önmagában csak az adathalmaz torzításait jellemzi a teljesen uniform eloszláshoz képest, az osztályozót magát nem. Az *information gain* (vagy *KL-divergencia*) ezzel szemben figyelembe veszi a helyesen kategorizált adatpontokat is:

$$\begin{aligned}
&IG(f) = I_E(\mathcal{X}) - I_E(\mathcal{X}|correct) = \\
&= -\sum_{c=1}^{C} f_c \log_2 f_c + \sum_{c=1}^{C} f_c \log_2 \left(\frac{pr(\text{in } c \text{ and correct})}{pr(\text{in } c)} \right) = -\sum_{c=1}^{C} f_c \log_2 f_c + \\
&+ \sum_{c=1}^{C} f_c \log_2 \left(\underbrace{\frac{\cancel{|\text{in c}|}}{\cancel{|\text{all points}|}} \times \frac{|\text{correct in c}|}{\cancel{|\text{in c}|}}}_{\text{szamlalo}} \times \underbrace{\frac{\cancel{|\text{all points}|}}{|\text{in c}|}}_{\text{nevezo reciproka}} \right)
\end{aligned} \tag{1.50}$$

```
>>> import math
>>> def IG(cat_freqs, corr_freqs):
...     igval = 0.0
...     for inx, elem in enumerate(cat_freqs):
...             subval = cat_freqs[inx] * (math.log(cat_freqs[inx]) / math.log(2))
...             addval = cat_freqs[inx] * (math.log(corr_freqs[inx]) / math.log(2))
...             igval = igval - subval + addval
...     return igval
...
>>> cat_freqs = [0.3, 0.4, 0.1, 0.2]
>>> corr_freqs = [1.0, 0.3, 1.0, 1.0]
>>> IG(cat_freqs, corr_freqs)
1.1516531070045328
>>> corr_freqs = [1.0, 0.9, 1.0, 1.0]
>>> IG(cat_freqs, corr_freqs)
1.7856381072929954
>>> corr_freqs = [1.0, 0.9, 0.3, 1.0]
>>> IG(cat_freqs, corr_freqs)
1.6119415478763748
```

1.6. ábra. Példa az információ-nyereségre.

Vegyük észre, hogy minél nagyobb azon pontok aránya c kategórián belül, amiket jól osztályozunk, annál többet adunk hozzá (vagyis: annál kevesebbet vonunk ki) az eredeti értékhez / értékből.

Erre mutat példát a 1.6. ábra. Látjuk, hogy ha kevesebbet tévedünk, nagyobb lesz a kinyert információ. Ha pedig ugyanannyit tévedünk kevésbé fajsúlyos, mint a fajsúlyosabb minták esetén, az eredményben szintén meglátszik: kisebb lesz a kinyert információ, ha fontosabb kategóriát tévesztünk ugyanannyit.

1.5.6. Receiver operating characteristic és Area under curve metrikák

Ahogy korábban is említettük, hasznos, ha az osztályozó kimenete valószínűséget is társít a kimenethez. Sokszor azonban a tévedés iránya sem mindegy: nevezetesen, hogy „hamis pozitívat", vagy „hamis negatívat" ad vissza a modell.

Ha pl. az a feladat, hogy egy kártyás vásárlásnál megmondjuk hogy éppen csalás történik-e, mindkét döntésnek van költsége:

- Ha csalást jelez a rendszer, közbe kell lépni, és a vásárló esetleg mérges lesz
- Ha azt mondjuk, hogy nincs csalás, ezzel is lehet hogy, hogy tévedünk és kárt okozunk.

Ennek szemléltetéséhez hasznos absztrakció a ***konfúziós mátrix*** (más néven *confusion matrix* vagy *contingency matrix*). Ld. például a 1.7. ábrát!

Mi történik, ha itt módosítjuk a döntési küszöböt? Nézzük előbb az extrém példákat:

- Ha a küszöb 0%, minden linkre azt fogjuk mondani hogy a user rákattint. Az összes link a bal oszlopban lenne. 17 TP (true positive) lenne, és 118 FP (false positive)
- Ez akkor lenne jó döntés, ha nem lenne költsége a FP-nak és nem lenne haszna a TN-nak (true negative)

	Predicted Class		
Actual Class		Positive (Click)	Negative (Not Click)
	Positive (Click)	True Positive 10	False Negative 7
	Negative (Not-click)	False Positive 22	True Negative 96

1.7. ábra. Példa a konfúziós mátrixra a [1] könyvből, egy olyan modell esetén, amelyik megpróbálja megjósolni, hogy egy webes linkre rákattint-e a felhasználó, vagy sem.

- ... 100%-nál hasonló logikával okoskodhatunk, csak a fordított esetet kapnánk.

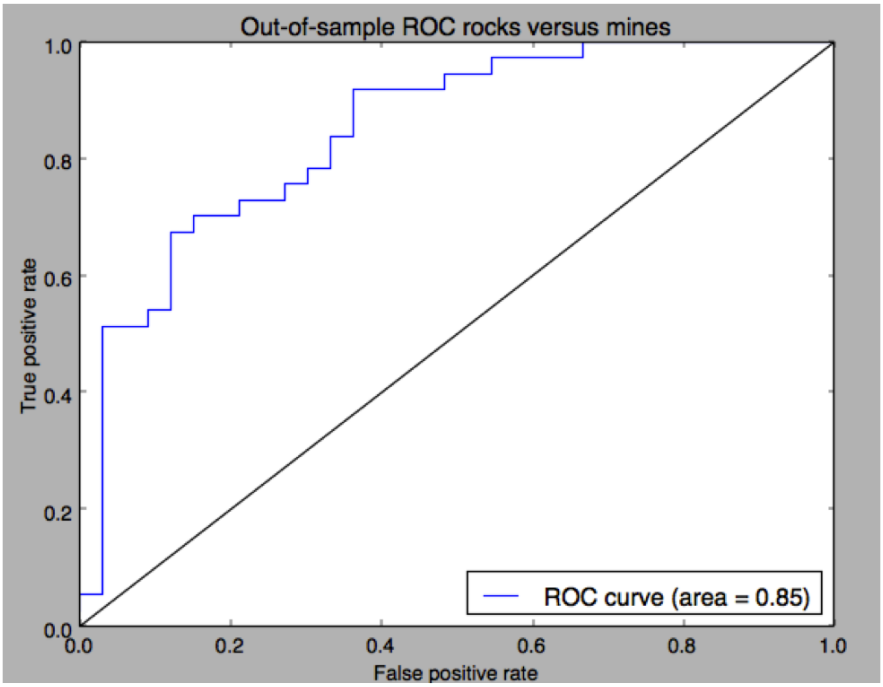
Az extrém példák a gyakorlatban nyilvánvalóan nem jók, ennél megfelelőbb küszöbértéket kell választanunk - azonban, hogy ez mennyi lesz, a feladattól is függ. Mindenesetre a küszöbérték módosításának hatásait jól jellemzi az ún. ***Receiver operating characteristic (ROC)***, vagy ROC-görbe, amelyre példát a 1.8. ábrán láthatunk.

Az elnevezés, hogy ROC, az eredeti alkalmazásból származik — amikor el kellett dönteni, hogy van-e vagy nincs a levegőben ellenséges repülőgép. A görbe maga a helyes pozitívok rátáját ábrázolja a hamis pozitívok rátájával szemben. Ezek azt fejezik ki, hogy ha egy pozitív (vagy negatív) mintát veszünk, mennyire valószínű, hogy jót (illetve hogy rosszat) mondunk rá:

$$\begin{aligned} TPR &= \frac{TP}{TP+FN} = \frac{TP}{P} \\ FPR &= \frac{FP}{FP+TN} = \frac{FP}{N} \end{aligned} \tag{1.51}$$

Magának a görbének a megértéséhez hasznos a következő gondolatkísérlet:

- Ha nagyon kicsi a döntési küszöb, minden mintára azt mondjuk, hogy pozitív $\rightarrow$ nem lesz FN és TN $\rightarrow TPR = 1$ és $FPR = 1$
- Ha maximális a döntési küszöb, minden mintára azt mondjuk, hogy negatív $\rightarrow$ nem lesz TP és FP $\rightarrow TPR = 0$ és $FPR = 0$
- Vagyis ha maximális vagy minimális a küszöb, az x-y tengely bal alsó illetve jobb felső sarkában leszünk.
- Tegyük most fel, hogy van egy nem nulla döntési küszöbünk, és ezt infinitezimálisan csökkentjük. Ekkor:
  - Több mintát fogunk pozitívnak mondani, mint korábban $\rightarrow$ *TP és/vagy FP nőni fog*



1.8. ábra. Példa a Receiver operating characteristic görbére a [1] könyvből.

– A nevezők viszont fixen maradnak → TPR és/vagy FPR nőni fog (csökkenni viszont egyik sem tud!)

Az ideális az lenne, ha mindig a bal felső sarokban lennénk. Ez azonban a gyakorlatban nem ilyen egyszerű:

- A TPR más néven a modell *érzékenysége* (sensitivity), FPR pedig 1 mínusz a *specifikussága* (specificity).
- Ha az érzékenységet növelnénk, azt fix modellnél csak úgy tehetjük meg, ha a küszöböt lejjebb visszük. Ezzel egyidőben viszont nem tudjuk csökkenteni FPR-t (vagyis növelni a specifikusságot)

Ennek megfelelően a ROC-görbe egy jobbra növekvő cikkcakk-ként jelenik meg.

1.6. Modell szelekció

Hogyha a modellünket a fentebb ismeretett módszerekkel már ki tudjuk értékelni, akkor az a kérdés, hogy több modell közül melyiket válasszuk ki a későbbi felhasználásra?

Ezt a kérdést nem feltétlenül könnyű megválaszolni, hiszen bennünket nem elsősorban a tanítóhalmazon mért teljesítmény, hanem az általánosítás képessége fog érdekelni. Az ún. out-of-sample esetben pedig (általában) természetesen romlani fog a teljesítmény, mert nem azokra a mintákra hangoltuk rá a modellt. Mindez összefügg a variancia-torzítás dilemmával is. Ha ugyanis az összes tanítómintát

„memorizáljuk”, azzal a varianciát növeljük, ami túltanulást eredményezhet. Ha viszont nagyon egyszerű modellt választunk, amely nem rendelkezik kellő számú paraméterrel vagy kellő kifejezőerővel az adatokban rejlő mintázatok lefedésére, akkor magas bias-unk lesz, ez pedig extrém esetben sem a tanulásnak, sem az általánosításnak nem kedvez. Az e kettő közötti átmenetben lesz várhatóan egy olyan pont, ahol a betanított modell a tanítóadatokra is elég jó performanciát nyújt, és az általánosító képessége sem romlik még el nagyon.

1.6.1. Train, validate, test

Ahhoz, hogy ezt a pontot megtaláljuk, és szimulálni tudjuk az általánosítókélpességet, a gépi tanulásban általában az elkülönített adatokon való tesztelés módszerét alkalmazzuk. Összesen 3-féle adathalmazt szoktunk megkülönböztetni:

1. Tanítóadatok: ezeken az adatokon tanítjuk be az összes modellt és a tanítás során az ezekre kapott hibát próbáljuk meg csökkenteni;
2. Validációs adatok: ezeket az adatokat nem használjuk fel a tanításhoz, de minden epoch (tanítási fázis) végén kiértékeljük rajtuk a modell(jeink) teljesítményét. Ez segíthet annak eldöntésében, hogy melyik modellünk a legjobb (feltéve hogy több is van, más hiperparaméterekkel vagy akár teljesen más architektúrával is);
3. Teszt adatok: miután, és csak miután kiválasztottuk a legjobb modellt, amivel dolgozni kívánunk, a modellt a soha nem látott teszt adatokon is kiértékeljük. A performanciát ez alapján a teszt alapján szabad csak mások felé jelenteni / publikálni, ugyanis ezek az adatok még a modell szelekcióban sem játszottak szerepet.

1.6.2. Keresztvalidáció

Abban az esetben, ha nem áll rendelkezésre kellő mennyiségű adat ahhoz, hogy külön validációs és teszthalmazt is kijelöljünk (nem maradna kellő tanítóadat a tanuláshoz), alkalmazhatjuk az ***n-szeres keresztvalidáció*** (angolul: ***n-fold cross-validation*** módszerét!

Ehhez a tanítóhalmazt n szeletre bontjuk, és n különböző iterációban mindig ezek közül az egyik szeletet használjuk validálásra, a többit tanításra. Így lesz n iterációnk. A végén a hibamértékeket érdemes átlagolni. Fontos azonban, hogy a szeleteket úgy válasszuk meg, hogy ne torzuljon az adatok statisztikája (pl. a teszthalmazban mindig legyen mindenféle példa, ehhez ld. a *stratified sampling* módszerét).

Végső soron így a validációs halmazt és a tanítóhalmazt eggyé olvasztjuk, így több tanítóadathoz jutunk összességében, azonban az adatok egy – mindig más – szeletét fenntartunk a validálásra. Ha ezek alapján kiválasztottuk a legjobb modellt / legjobb hiperparamétereket, utána az összes (tanító- és validációs) adaton újra taníthatjuk, viszont ez után már csak a tesztadatok maradnak a modell kiértékelésére, performanciájának publikálására.

2. fejezet

Lineáris regresszió és általánosításai

Forrásmegjelölés

A fejezetben számos magyarázat és példa az alábbi angol nyelvű referenciákból származik: [1, 3].

Általánosan a lineáris regresszió egyszerű módszert ad a felügyelt tanulásra (azon belül a predikcióra és inferenciára is):

- Predikció célja: mondjuk meg y-t $\mathbf{x}$ alapján (bemenetből kimenet)
- Inferencia célja: inverz kérdés, azaz mondjuk meg mi volt a bemenet ($\mathbf{x}$) a kimenet (y) alapján. Ez több további igényt is magával von:
  - magyarázzuk meg, hogy y melyik x_i bemenettől függ leginkább, melyiktől másodsorban, stb.
  - határozzuk meg, hogyan kellene $\mathbf{x}$ vektort módosítanunk, hogy y valamilyen módon változzon (pl. kétszer akkora legyen)
  - mutassuk meg, milyen interakciók vannak a paraméterek között

A lineáris regresszió alapváltozata több, mint 200 éve ismert, és rengeteg modernebb megközelítés létezik. Ugyanakkor hasznos tudnunk róla, egyrészt mert a témakörben megjelenő szempontok más módszereknél is vissza-visszatérő szempontok lesznek, másrészt pedig azért, mert a lineáris regresszió korszerűbb, továbbfejlesztett változataira sokszor ma is (feladattól függően) hatékony módszerként tekinthetünk.

A fejezetben először az eredeti elgondolásokon megyünk végig, majd rátérünk a lineáris regresszió kiterjesztéseként tekinthető korszerűbb módszerekre is. A megvizsgálandó kérdéseket egy példán keresztül mutatjuk be, amely a [3] könyvből származik. A példa szerint tegyük fel, hogy egy hirdetőcég megkér minket, hogy tervezzük meg marketing portfolióját. A létrehozandó modell kimenete a *sales* változó, amely a cég termékéből fix idő alatt eladott mennyiségét jelenti. A bemenő paraméterek pedig lehetnek a *tv*, *radio*, és *newspaper* változók (ezekben a médiumokban lehet hirdetni,

és a változók értéke a hirdetésekre elköltött pénzmennyiséget tükrözi). A létrehozandó modell által megválaszolandó fő kérdés: hol hirdessünk, hogy a legtöbbet tudjuk eladni?

Néhány további kérdés, amely szükségszerűen felmerül:

- Van-e egyáltalán kapcsolat a hirdetési büdzsé és az eladott mennyiség között?
- Ha van kapcsolat, milyen erős?
  - Ha csak nagyon nagy hibával tudjuk megjósolni az eladott mennyiséget, akkor gyenge a kapcsolat
- Ha van kapcsolat, lineáris-e egyáltalán, vagy valamilyen bonyolultabb összefüggést érdemesebb keresni?
- Melyik médiumok mennyire járulnak hozzá leginkább a sikerhez?
  - Ha x médiumra 1 dollárral többet költünk, mennyivel nő az eladott termékmennyiség?
  - Mennyire pontosan lehet ezt egyáltalán megmondani?
- Vannak-e szinergiák (interakciók) a médiumok között?
  - Pl. lehet hogy 50.000-50.000 tv-ben és rádióban elköltött USD többet produkál, mint ha csak egyikbe vagy másikba fektetnénk bele 100.000 dollárt

Amint a fejezetben látni fogjuk, a lineáris regresszió segítségével mindegyik kérdés legalább valamilyen bizonyossággal megválaszolható.

2.1. Mitől (nem feltétlenül) lineáris a lineáris regresszió?

Fontos mindjárt az elején leszögezni, hogy az itt bemutatott módszer neve ugyan „lineáris regresszió", de ettől még előfordulhat, hogy az x_i bemenetek mögött nemlineáris összefüggések állnak és így ezeket a regresszió során fel is használjuk (például egy konkrét bemenet jelentése lehet $sebesség^2$ vagy $\cos(\text{sebesség})$). A lineáris regresszió lényege, hogy ha a bemenő vektort fixnek tekintjük (függetlenül attól, hogy a vektor elemei között akár nemlineáris tagok is szerepelhetnek), akkor a vektor elemei között próbálunk lineáris összefüggést felállítani:

$$y \approx \omega^T \mathbf{x} \tag{2.1}$$

2.2. Jelölések

A továbbiak során az alábbi jelöléseket használjuk:

- M – a tanításhoz használt pontok száma (a pontok indexálása m-mel történik, ami 1-től M-ig fut)

- D – a lineáris modell dimenziószáma. A dimenziók számozása 1-től D-ig fut (általános esetben d-vel jelöljük).
- Létezik egy 0-ás számú dimenzió is, ami egy konstans torzítást jelent.
- Skalárok esetén – vagy ha az indexálás épp nem érdekes – akkor a tanítópontot alsó indexben levő m-mel is jelölhetjük.
- $\mathcal{X}$ - a tanítóhalmazt szükség esetén továbbra is így jelöljük

Ekkor a lineáris modell (megjegyzés: itt az m-edik tanítópontról volt szó):

$$y^{(m)} \approx \omega_0 + \omega_1 x_1^{(m)} + \ldots \omega_D x_D^{(m)} \tag{2.2}$$

2.3. Egyszerű lineáris regresszió (OLS) egyváltozós esetben

Az egyszerű lineáris regressziót angolul *ordinary least squares*-nek (OLS) is szokás nevezni. Először nézzük azt az esetet, amikor egyetlen x_1 prediktor változónk van. A keresett összefüggés:

$$y \approx \omega_0 + \omega_1 x_1 \tag{2.3}$$

Azt is mondhatjuk, hogy y-t x_1-re regresszáljuk. Például x_1 jelentheti a **tv**-hirdetések büdzséjét, y pedig a **sales** változót. A modell paraméterei, ω_0 és ω_1 egy karteziánus egyenes tengelymetsző pontját és meredekségét határozzák meg. Mielőtt bármit meg tudnánk jósolni, ezeknek értéket kell találni M darab tanítópont alapján.

Az a kérdés, hogy egy paraméterbeállítás mennyire illeszkedik egy adott tanítóhalmazhoz, sokféleképpen megválaszolható. Gyakori módszer a legkisebb négyzetek kritériuma. Ehhez legyen az m-edik pontra adott predikció:

$$\hat{y}^{(m)} = \hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)} \tag{2.4}$$

ahol a kalapok azt jelölik, hogy nem feltétlenül ez a helyes kimenet és a legjobb paraméter-értékek. Ekkor az m-edik **reziduális** (maradék, hiba):

$$e_m = y^{(m)} - \hat{y}^{(m)} \tag{2.5}$$

És az RSS (residual sum of squares):

$$\begin{aligned} RSS &= e_1^2 + e_2^2 + \ldots + e_M^2 = \\ &= (y^{(1)} - \hat{\omega}_0 - \hat{\omega}_1 x_1^{(1)})^2 + \ldots + (y^{(M)} - \hat{\omega}_0 - \hat{\omega}_1 x_1^{(M)})^2 = \\ &= \sum_{m=1}^{M} (y^{(m)} - \hat{\omega}_0 - \hat{\omega}_1 x_1^{(m)})^2 \end{aligned} \tag{2.6}$$

Ahhoz, hogy ezt minimalizáljuk, a deriváltakat kell 0-val egyenlővé tenni (a paraméterek parciális deriváltját). Egyrészt:

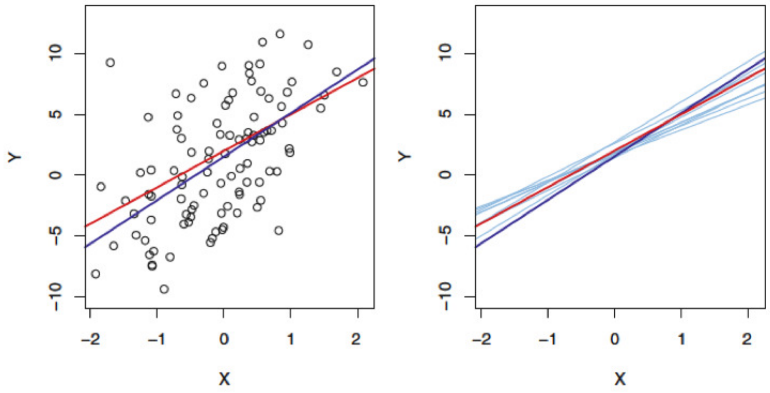
$$\begin{aligned}
\frac{\partial}{\partial \hat{\omega}_0} RSS = \frac{\partial}{\partial \hat{\omega}_0} \sum_{m=1}^{M} \left[y^{(m)} - (\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)}) \right]^2 = 0 \\
-2 \sum_{m=1}^{M} \left[y^{(m)} - (\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)}) \right] = 0 \\
M\hat{\omega}_0 + \sum_{m=1}^{M} x_1^{(m)} \hat{\omega}_1 = \sum_{m=1}^{M} y^{(m)} \\
M\hat{\omega}_0 + \hat{\omega}_1 \sum_{m=1}^{M} x_1^{(m)} = \sum_{m=1}^{M} y^{(m)} \\
\hat{\omega}_0 + E[x_1]\hat{\omega}_1 = E[y]
\end{aligned} \tag{2.7}$$

Másrészt:

$$\begin{aligned}
\frac{\partial}{\partial \hat{\omega}_1} RSS = \frac{\partial}{\partial \hat{\omega}_1} \sum_{m=1}^{M} \left[y^{(m)} - (\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)}) \right]^2 = 0 \\
\frac{\partial}{\partial \hat{\omega}_1} \sum_{m=1}^{M} \left[y^{(m)2} - 2y^{(m)}(\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)}) + (\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)})^2 \right] = 0 \\
\sum_{m=1}^{M} \left[-2y^{(m)} x_1^{(m)} + 2(\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)}) x_1^{(m)}) \right] = 0 \\
-2 \sum_{m=1}^{M} x_1^{(m)} \left[y^{(m)} - (\hat{\omega}_0 + \hat{\omega}_1 x_1^{(m)}) \right] = 0 \\
\sum_{m=1}^{M} x_1^{(m)} \hat{\omega}_0 + \sum_{m=1}^{M} x_1^{(m)\ 2} \hat{\omega}_1 = \sum_{m=1}^{M} x_1^{(m)} y^{(m)} \\
E[x_1]\hat{\omega}_0 + E[x_1^2]\hat{\omega}_1 = E[x_1 y]
\end{aligned} \tag{2.8}$$

Ha a két kék egyenlet közül az elsőt megszorozzuk $E[x_1]$-el, majd a másodikból kivonjuk, akkor az alábbi összefüggéshez jutunk:

$$\begin{aligned}
&E[x_1]\hat{\omega}_0 + E[x_1]E[x_1]\hat{\omega}_1 = E[x_1]E[y] \\
&E[x_1]\hat{\omega}_0 + E[x_1^2]\hat{\omega}_1 = E[x_1 y] \\
&\hline \\
&(E[x_1^2] - E[x_1]^2)\hat{\omega}_1 = E[x_1 y] - E[x_1]E[y] \\
&\hat{\omega}_1 = \frac{\sum (x_1^{(m)} - E[x_1])(y^{(m)} - E[y])}{\sum (x_1^{(m)} - E[x_1])^2}
\end{aligned} \tag{2.9}$$



2.1. ábra. A valódi regressziós egyenes és a legkisebb hibanégyzetes modellek kimenetei közötti különbséget mutatja be ez a [3] könyvből származó ábra.

ahol az utolsó lépésben kihasználtuk a variancia és kovariancia alternatív definícióit.

A számlálóban a kovariancia van, a nevezőben pedig x varianciája (illetve ezek M-szeresei, de M kiesik). A második kékből egyenletből pedig ω_0 is számítható.

2.3.1. Az együtthatók pontossága és az összefüggés létezése

Az eredeti képlet így nézett ki:

$$y \approx \omega_0 + \omega_1 x_1 \tag{2.10}$$

Ez azért approximatív, mert hozzáadódik egy nulla várható értékű random hiba is. Vagyis a teljes modell:

$$y = \omega_0 + \omega_1 x + \epsilon \tag{2.11}$$

A hiba tükrözi a modell helyességét (nem biztos hogy tényleg lineáris, sem pedig hogy csak egy változó határozza meg); illetve az esetleges mérési hibákat. Ez a modell tehát egy olyan populációra vonatkozó **regressziós egyenest (RE)** ad, ami a legjobb közelítést jelenti az x_1 és y közötti valódi összefüggésre. Ezt a valódi összefüggést azonban a gyakorlatban nem ismerjük (a konkrét paramétereket illetően), ezért kénytelenek vagyunk az előbb tárgyalt legkisebb hibanégyzetes modell-t használni, amely a **legkisebb négyzetes egyenest (LNE)** meghatározza:

$$\hat{y} = \hat{\omega}_0 + \hat{\omega}_1 x \tag{2.12}$$

A [3] könyvből származó, 2.1. ábrán egy szimulált példa látható. A regressziós egyenes pirossal, a kapott legkisebb hibanégyzetes modell(ek) pedig a tanítóminta függvényében különbözőek lehetnek – ezeket a kék árnyalataival szemlélteti az ábra szerepel(nek). Az eredeti regressziós egyenest a következő modell adja: $y = 2 + 3x + \epsilon$

A valódi modellt soha nem ismerjük: ehelyett pontokat kell mintavételeznünk, és azokra lefuttatnunk a regressziót. Ha ezt sokszor megtesszük, mindig picit más eredményt kapunk, noha a tényleges regressziós egyenes nem változik (pl. mert a tényleges regressziós egyenes alapján legenerálunk 10.000 pontot, de a hiba miatt azok nem egy egyenesre esnek és mindig más pontokkal tanítunk). Az eredményül kapott legkisebb hibanégyzetes modellek közötti eltérést a variancia jellemzi.

Eleinte furcsának tűnhet, hogy beszélhetünk egy tényleges modellről (RE) és egy közelítésről is (LNE), de ez a kettősség minden statisztikai módszer sajátja. Ha egy populáció átlagára vagyunk kíváncsiak:

- A tényleges átlagot szinte sosem ismerjük
- Helyette mintákat veszünk, és azokat átlagoljuk. Ha kellően sok mintát vettünk, bízhatunk benne, hogy egy precízebb becsléshez jutunk, amely sem lefelé, sem felfelé nem torzít.

Ugyanez igaz a legkisebb négyzetek módszerére is: ha sokszor megismételjük, reményeink szerint nem lesz szisztematikus torzítás. Ezt látjuk a jobb oldali részén a 2.1. ábrának.

Ezek után természetesen merülhet fel a kérdés: mennyire ad jó becslést a mintapontok átlaga a ténylegesre? (Jelen esetben a mintapontok az együtthatók lennének, ezeket „mintavételezzük" sok kísérlettel). Ennek megállapításához hasznos az alábbi képlet, amely akkor igaz, ha a mintavételezett pontok nem korrelálnak egymással:

$$Var(\hat{\mu}) = SD(\hat{\mu})^2 = \frac{\sigma^2}{M} \tag{2.13}$$

Kihasználva ugyanis, hogy az átlagok egymástól függetlenek (nem korrelálnak egymással):

$$\begin{aligned} &Var(\mu_1 + \mu_2 + \ldots + \mu_M) = Var(\text{összeg}) = M\sigma^2 \text{ , és} \\ &Var(\frac{\text{összeg}}{M}) = Var(\frac{1}{M}\text{összeg}) = \left(\frac{1}{M}\right)^2 Var(\text{összeg}) = \\ &\frac{1}{M^2} M\sigma = \frac{\sigma^2}{M} \end{aligned} \tag{2.14}$$

Ez mutatja, hogy ha $2x$-es szorzóval szeretnénk csökkenteni az átlagban való bizonytalanságunkat, akkor $4x$ annyi mintát kell vennünk, ugyanis:

$$SD(\hat{\mu}) = \sqrt{(Var(\hat{\mu}))} = \frac{\sigma}{\sqrt{M}} \tag{2.15}$$

Gyakorlati alkalmazásokban viszont nem ismerjük szigmát (a teljes populáció szórását), ezért a sample standard deviation-nel (azaz az s-ként jelölt mintaszórással) dolgozunk. Így kapjuk meg SD helyett SE-t (standard error):

$$SE(\hat{\mu}) = \frac{s}{\sqrt{M}} \tag{2.16}$$

Ha feltesszük, hogy az adatok normál-eloszlásból származnak, még ez is hasznos. Pl. 95%-os konfidencia-szinttel az átlag alsó és felső határa: $\hat{\mu}_{bound} = \hat{\mu} \pm (SE * 1.96)$

Mindezt felhasználhatjuk arra is, hogy a lineáris regresszió együtthatóiban való bizonytalanságunkat csökkentsük. Ehhez többször megismételjük (több mintán) a kísérletet, és a szórást elosztjuk a minták számának gyökével. Ez a standard error, amit több kísérlettel csökkenthetünk.

Van hogy fontos is csökkenteni. Például ha az a kérdés, hogy létezik-e valójában összefüggés x_1 és y között, adott esetben hiába is néznénk meg, hogy ω_1 különbözik-e 0-tól. Ami sokkal inkább lényeges, hogy mennyivel kellene, hogy 0-ától különbözzön? Ha ω_1 standard errorja kicsi (vagyis meglehetősen biztosak vagyunk az egész populációra vett értékében), akkor kis értékét is elfogadhatjuk nem 0-nak. De ha bizonytalanok vagyunk az együttható értékében, akkor kérdés hogy egy 0.0001-es együttható-érték mit jelent egyáltalán?

A gyakorlatban az ún. t-statisztikát számítjuk ki, ami megmondja, hogy az approximációnk a standard error hányszorosával tér el 0-tól:

$$t = \frac{\hat{\omega}_1 - 0}{SE(\hat{\omega}_1)} \tag{2.17}$$

Hogy t-nek mennyinek kell lenni ahhoz, hogy azt mondjuk hogy nem 0 az együttható, az attól is függ hogy hány mintát vettünk, illetve hogy mekkora konfidenciával (pl. 95%, vagy 99%) szeretnénk választ adni. Pl. ha 30 mintánk van, akkor ezekhez a konfidenciákhoz $t = 2$ illetve $t = 2.75$ a megfelelő t-érték.

2.3.2. A modell pontossága

Amint tudjuk, hogy létezik kapcsolat x_1 és y között (a null-hipotézist elvetettük), az is kérdés hogy mennyire illeszkednek az adatok a modellhez? Eddig csak azt tudtuk, hogy van kapcsolat: de hogy az gyenge vagy erős, azt még nem.

A modell erősségének jellemzésére kétféle statisztikát szokás használni: a **residual standard error (RSE)** és az **R2 statisztikát**. Emlékezzünk vissza, hogy a regressziós modell inheresen nem lehet pontos, mert van egy ϵ hiba. Ezért még akkor se lenne 100%-os az illeszkedés, ha a pontos együtthatókat ismernénk.

A residual standard error (RSE)

Lényegében a regressziós modell ϵ hibájának szórása – ha feltesszük hogy a legkisebb hibanégyzetes modellünk a lehető legtökéletesebb:

$$RSE = \sqrt{\frac{1}{M-2}RSS} = \sqrt{\frac{1}{M-2}\sum_{m=1}^{M}(y^{(m)} - \hat{y}^{(m)})^2} \tag{2.18}$$

A szórástól ez annyiban különbözik, hogy M helyett $M-2$-vel osztunk. Például, ha a hirdetéssel kapcsolatos feladatban RSE értéke 3.200 lenne, akkor az azt jelentené, hogy:

- Ha pontosan ismerjük ω_0 és ω_1 együtthatókat, akkor is átlagban legalább ± 3.200-zal tér el az előrejelzésünk a valójában eladott termékek számától
- Ez a legoptimistább becslést adja, és a hibát eleve minimalizáltuk $\Rightarrow$ valóban megmutatja, hogy mennyire pontos a modellünk.

Mivel az RSE a kimeneti érték egységeiben ad választ, ezért nincs univerzális értelmezése: 3.200 eladott vagy nem eladott rágógumi sok vagy kevés? (attól függ, hogy amúgy 10 ezres vagy 10 milliós nagyságrendben szoktunk-e eladni).

Az R2 statisztika

Az R2 statisztika ehelyett egy arányt fejez ki (0 és 1 között), ami független y mértékegységétől. A következő képlettel számítjuk:

$$\begin{aligned} R^2 =& \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} = \\ & = 1 - \frac{\sum_{m=1}^{M} (y^{(m)} - \hat{y}^{(m)})^2}{(y^{(m)} - \bar{y})^2} \end{aligned} \tag{2.19}$$

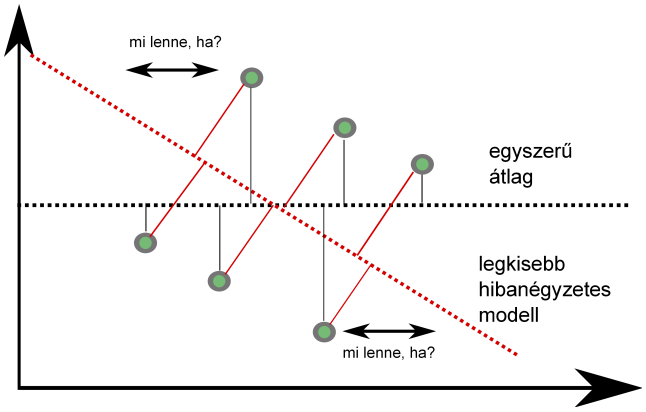
ahol TSS az ún. *total sum of squares*, ami azt mondja meg, hogy *önmagukban milyen a tanítópontok kimeneteinek variabilitása az átlagkimenet körül* (átlagcímke vs m-edik címke). RSS pedig azt mondja meg, hogy a *modellünk predikciójának mi az aktuális kimenetek körüli variabilitása* (m-edik predikció vs m-edik címke). Ezek közül fontos, hogy egyrészt egyik sem lehet kisebb, mint 0. Ha az első törtet nézzük, látszik hogy R2 értéke ráadásul maximum 1. Ugyanakkor 0-nál se lehet kisebb, mert RSS nem haladhatja meg TSS-t (ha mégis így lenne, akkor már eleve rosszul tanítottuk volna be a modellt, mert ha a regresszió után nagyobb a hibánk szórásnégyzete, mint az átlaghoz képesti szórásnégyzet, akkor célszerűbb lett volna mindjárt fixen a tanítóhalmaz átlagát visszaadni bármilyen bemenetre). Tehát indirekt úton beláttuk, hogy nem minimalizálta a modell a tanítási hibát sem!

Tulajdonképpen R2 azt mondja meg, hogy a megoldásunk mekkora hányadát „magyarázza meg" az eredeti variabilitásnak.

- Ha R2 közel van 1-hez, akkor a variabilitás nagy részét megmagyaráztuk (RSS ugyanis azt jelenti, hogy a regresszió után mennyi variabilitást hagytunk megmagyarázatlanul)
- Ha közel van 0-hoz, akkor RSS majdnem akkora volt, mint TSS $\rightarrow$ nem magyaráztunk meg semmit, mert ennyi erővel átlagolhattuk is volna a tanítópontokat.

A 2.2 ábra alapján látjuk, hogy az R2 statisztika is sokat elmond a probléma inherens hibájáról.

- Ezen az ábrán pl. nem létezik olyan legkisebb hibanégyzetes modell, ami jobb lenne.
- Ha ezeket a pontokat valóban regressziós modell alapján generáltuk, akkor eleve nagy volt az ϵ hiba szórása.



2.2. ábra. Az R2 statisztika értelmezése. A „mi lenne, ha?" felvetés a bal felső és jobb alsó adatpontnál arra vonatkozik, hogy egyrészről ha ezek a pontok vízszintesen közelebb kerülnének a piros egyeneshez (a tanult modellhez), a modell pontossága nagyobb lenne, azonban a total sum of squares (TSS) változatlan maradna. Másrészről viszont ha távolabb kerülnének a piros egyenestől – ugyanezen a vízszintes tengely mentén – akkor a modellünk pontossága romlana, miközben a TSS szintén változatlan maradna. Ahogy levezettük, annyira nem romolhat a modellünk, hogy a TSS-nél is rosszabb legyen, hiszen ekkor a modellünk a TSS-sel esne egybe. Viszont ez a feltételezés, hogy ti. a modellünk a lehetőségekhez mérten a legjobban illeszkedik a tanítópontokra, azt eredményezi, hogy ennek a kiértékelése a lineáris regresszió, mint kategória, alkalmasságáról is szól.

Az R2 statisztikát mégis értelmezhetjük úgy is, mint egy választ arra a kérdésre, hogy alkalmas-e maga a lineáris modell az adatok modellezésére? Ha minimalizáltuk a hibánkat, és így is elég kis R2-t kapunk, akkor könnyen lehet, hogy bonyolultabb modell után kell keresgélnünk.

2.4. Egyszerű lineáris regresszió (OLS) többváltozós esetben

Az eddigiek többdimenziós esetre is általánosíthatóak. Ekkor a modell:

$$y^{(m)} \approx \omega_0 + \omega_1 x_1^{(m)} + \ldots \omega_D x_D^{(m)} \tag{2.20}$$

Itt is lesz egy ϵ fennmaradó hiba, ezért beszélhetünk approximációról. ϵ továbbra sem konstans, hanem mondjuk egy normál eloszlású valószínűségi változó, μ várható értékkel és valamilyen σ szórással. Ezért az y kimenet is normál-eloszlású:

$$p(y|\mathbf{x}) = \mathcal{N}(\omega^T \mathbf{x} + \mu, \sigma^2) \tag{2.21}$$

(ω_0-át nem kell külön kiírni ha $\mathbf{x}$ vektor elejére beírunk egy 1-est. Ugyanígy μ is eltüntethető)

A hibával (μ, σ) megint nem tudunk sok mindent csinálni, a modell pontatlanságából illetve a mérési hibákból adódik. A legjobb súlyok megkereshetőek az ún. *maximum likelihood estimation* módszerrel (maximalizáljuk a látott minták előfordulásának valószínűségét, feltéve hogy a modell alkalmas egyáltalán a minták generálására). Nézzük, hogyan!

Amit keresünk az egy valószínűséget maximalizáló súlyvektor:

$$\hat{\omega} = arg\max_{\omega} \log p(\mathbf{y}|\mathbf{X}) = arg\max_{\omega} \log \prod_{m} p(y^{(m)}|\mathbf{x}^{(m)}) \tag{2.22}$$

A szorzat azért működik mert a tanítómintákat elvben függetlenül mintavételezzük, így az összes látott dolog maximum valószínűsége külön a valószínűségek szorzatának maximuma. Szorzat logaritmusa pedig a logaritmusok összege, vagyis:

$$\hat{\omega} = arg\max_{\omega} \sum_{m} \log p(y^{(m)}|\mathbf{x}^{(m)}) \tag{2.23}$$

Helyettesítsük be ide a normál-eloszlást!

$$\hat{\omega} = arg\max_{\omega} \sum_{m} \log \left(\frac{1}{2\pi\sigma^2}\right)^{1/2} exp\left(-\frac{1}{2\sigma^2}(y^{(m)} - \omega^T\mathbf{x}^{(m)})^2\right) \tag{2.24}$$

Tovább egyszerűsíthetünk, mert szorzat logaritmusa = a logaritmusok összegével, így amit maximalizálni kell:

$$\sum_{m} \left(\log \left(\frac{1}{2\pi\sigma^2}\right)^{1/2} + \log exp\left(-\frac{1}{2\sigma^2}(y^{(m)} - \omega^T\mathbf{x}^{(m)})^2\right)\right) \tag{2.25}$$

Ezek után: kitevős tag logaritmusából a kitevő „lejön”, log exp x pedig maga x:

$$\begin{aligned}
&\sum_{m} \left(\log \left(\frac{1}{2\pi\sigma^2}\right)^{1/2} + \log exp\left(-\frac{1}{2\sigma^2}(y^{(m)} - \omega^T\mathbf{x}^{(m)})^2\right)\right) = \\
&\frac{M}{2}\log\frac{1}{2\pi\sigma^2} - \sum_{m}\frac{1}{2\sigma^2}(y^{(m)} - \omega^T\mathbf{x}^{(m)})^2 = \\
&\frac{-M}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\omega)^T(\mathbf{y} - \mathbf{X}\omega)
\end{aligned} \tag{2.26}$$

Itt az összegzés elhagyása érdekében áttértünk egy mátrixos jelölésre: $\mathbf{X}$ m-edik sora az m-edik tanítóminta (ezt szorozzuk az összes súllyal). Ezt a kifejezést akarjuk maximalizálni, ami ugyanazt jelenti, mintha elhagynánk az előjeleket és azt minimalizálnánk. Az első tag viszont független a súlyvektortól, ezért csak a másodikat kell nézni. Ha a deriváltat 0-val tesszük egyenlővé:

$$\begin{aligned}
\frac{\partial}{\partial\omega}\left(\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\omega)^T(\mathbf{y} - \mathbf{X}\omega)\right) &= 0 \\
\frac{\partial}{\partial\omega}(\mathbf{y} - \mathbf{X}\omega)^T(\mathbf{y} - \mathbf{X}\omega) &= 0 \\
\frac{\partial}{\partial\omega}\left[\omega^T(\mathbf{X}^T\mathbf{X})\omega - 2\omega^T(\mathbf{X}^T\mathbf{y}) + \mathbf{y}^T\mathbf{y}\right] &= 0 \\
2(\mathbf{X}^T\mathbf{X})\omega &= 2(\mathbf{X}^T\mathbf{y}) \\
\hat{\omega} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}
\end{aligned} \tag{2.27}$$

2.4.1. Együtthatók pontossága és az összefüggés létezése

A kapott LSE modellben meg kell néznünk, hogy a súlyvektor paraméterei 0-ák-e (persze megint több kísérlet átlagolásával és statisztikai próbával!)

$$y \approx \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 \tag{2.28}$$

Most azonban kicsit bonyolultabb a statisztikai próba, mint egyváltozós esetben. Ennek az az oka, hogy minden próba bizonytalansággal jár, itt pedig nemcsak kettő, hanem jóval több együtthatónk is lehet (az egyiknél előbb-utóbb pusztán a véletlenszerűség miatt tévedhetnénk!). Emlékezzünk vissza, hogy korábban a t-statisztikát így számítottuk:

$$t = \frac{\hat{\omega} - 0}{SE(\omega)} = \frac{\hat{\omega}\sqrt{M}}{s} \tag{2.29}$$

ahol s a minták szórása (sample standard deviation) volt; és a kapott t értéket egy megfelelő statisztikai táblázatból kiolvasva lehetett megmondani, hogy adott konfidencia-szinthez (mint 95% vagy 99%) elég nagy volt-e.

Most viszont nem arra vagyunk kíváncsiak, hogy igaz-e hogy $\omega_d = 0$, hanem arra, hogy igaz-e, hogy:

$$\omega_1 = \ldots = \omega_d = \ldots = \omega_D = 0 \tag{2.30}$$

Ennek megválaszolására pedig az ún. F-teszt alkalmas. Az F-statisztika az alábbiak szerint számítható:

$$F = \frac{(TSS - RSS)/M}{RSS/(M - D - 1)} \tag{2.31}$$

ahol M továbbra is a minták, D pedig a dimenziók száma. Érdekesség, hogy a nevező negatív szám lesz – vagy esetenként 0, ha nincsen legalább annyi megfigyelés, mint dimenzió. Például kétdimenziós esetben $D = 1$, ezért legalább 3 pontra van szükség (ellenkező esetben alulhatározott a feladat)

Emlékezzünk vissza, hogy RSS/TSS azt mondja meg, mekkora a megmagyarázatlanul hagyott variabilitás aránya. Itt az inverzét nézzük, vagyis jó esetben egy „nagy" számot kapunk, majd kivonunk belőle 1-et... majd ezt megszorozzuk egy $M - D - 1$ osztva D taggal. Ez utóbbi is általában egy nagy szám, mert általában $M >> D$, vagyis F nagyon nagy is lehet. Ezen kívül mivel TSS/RSS mindig legalább 1, F nem lehet 0-nál sem kisebb (feltéve persze ha a dimenzionalitáshoz mérten van elég megfigyelésünk). A képletből az is látszik, hogy több megfigyelés (nagyobb M) nagyobb F-statisztikát is tud eredményezni, feltéve hogy ugyanolyan „jók" vagyunk a reziduális hiba tekintetében.

Ha nincs semmilyen kapcsolat a változók és a kimenet között (mindegyik együttható 0), akkor RSS alig lesz kisebb, mint TSS és F kicsi szám lesz. Ha F jóval nagyobb, mint 1, akkor valamelyik tényező mindenképpen jelentős. De mi történik, ha F értéke közel van 1-hez? Ilyenkor az ún. F-próbát kell elvégezni.

Egy másik alternatíva, hogy többféle modell-változtatot kipróbálva – akár tesztadatokon is a teljesítményt ellenőrizve – megvizsgáljuk, hogy egyik vagy másik együttható „kinullázása" hatással van-e a teljesítményre – ld. később a 2.5.1. fejezetben.

2.4.2. Modell pontossága

Ugyanazokat a mértékeket amiket korábban – mint RSE és R2 – többdimenziós esetben is alkalmazhatjuk. Az eltérés annyi, hogy RSE esetében a skálázáskor D-t is figyelembe kell venni, ezért $M-2$ helyett $M-D-1$ szerepel benne:

$$RSE = \sqrt{\frac{1}{M-D-1}RSS} = \sqrt{\frac{1}{M-D-1}\sum_{m=1}^{M}(y^{(m)}-\hat{y}^{(m)})^2} \tag{2.32}$$

2.5. Inferencia

Mindezek alapján térjünk vissza a korábbi példára: Tegyük fel, hogy egy hirdetőcég megkér minket, tervezzük meg a portfoliójukat

- A kimenet a *sales* változó
- Bemenetek: *tv*, *radio*, és *newspaper* (ezekben lehet hirdetni)

Többek között az alábbi kérdések érdekelnek bennünket:

- Van-e, és milyen erős a lineáris kapcsolat a bemenetek és kimenet között?
- Hol hirdessünk hogy a legtöbbet tudjuk eladni?
- Vannak-e szinergiák (interakciók) a médiumok között?

2.5.1. Együtthatók fontossága

A következő kérdés: melyek a fontos változók? Nem mindig jó válasz, hogy vegyük a legnagyobb együtthatójút, mert:

- Lehet hogy az így kapott legjobb modell is önmagában kritikán aluli teljesítményt adna (ha a többi nem lenne)
- Lehetnek nagyon hasonló együtthatók is, illetve nem biztos hogy a változók akár a tanítóhalmazban is normalizáltak.

Az ideális az lenne, ha az összes lehetséges modellt kipróbálhatnánk. Pl. ha van D darab paraméterünk, akkor egy vagy több paraméter kihagyásával újabb regressziós modellt kapunk. Sajnos azonban ezen modellek száma 2^D, ami kis D-nél is rengeteg! Ezért ún. heurisztikákat szokás alkalmazni.

Az egyik népszerű heurisztika az ún. **forward selection**: kezdetben csak konstans eltolás van, és ehhez veszünk hozzá a D paraméterből 1-et. Megnézzük, melyik adja a legkisebb RSS-t, és azt

tartjuk meg... ezt folytatjuk, és minden körben egy újabb paramétert veszünk hozzá a korábbiakhoz, addig amíg javul az RSS. Ez nyilvánvalóan egyszerűsítés, hiszen ha egy változót már hozzávettünk a megoldáshoz, többé nem vesszük el így azokat az eseteket nem vesszük figyelembe, amikor egy kezdetben lényegtelennek tűnő bemenet egy másikkal kombinálva már igen hatékony.

Egy másik lehetőség a **backward selection**: az összes paraméterrel kezdünk, és mindig egyet-egyet elveszünk a legkevésbé szignifikánsak közül. Sajnos ha nincsenek normalizálva az adathalmazban a változók, akkor a szignifikancia csak próbákkal határozható meg, ezért a backward selection csak körülményesebben alkalmazható.

Egy harmadik módszer lehet az előző kettő kombinációja, amikor forward selectionnel indítunk, de időnként el is veszünk a korábban bevett paraméterek közül és újra megvizsgáljuk azok alkalmasságát.

2.5.2. Prognózis

Tegyük fel hogy kész a rendszer, mondani kell valamit a jövőről. Ha x, y és z összegeket költünk a hirdetésekre különböző médiumokban, mennyi lesz az eladott termékek mennyisége? A választ konfidencia-intervallumokkal szokás megadni, pontos becslést több okból sem tudunk adni:

- Lehet hogy nem pont lineáris a modell (inherens hiba)
- Ha van is pontosan lineáris összefüggés, mérési hibák torzíthatják a mintákat
- Nem a teljes populáción, hanem csak egy mintán tanítunk, ezért is lehet hibás a modell alapján kapott prognózis

2.5.3. Szinergiák

Végezetül: hogyan tudjuk megnézni, hogy van-e szinergia két változó között? Két változó szinergiájának vizsgálatakor készíthetünk egy új, a kettő szorzatából álló bemeneti komponenst:

$$\begin{aligned} y &= \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 (x_1 x_2) + \epsilon = \\ &\omega_0 + \underbrace{(\omega_1 + \omega_3 x_2)}_{\omega_1'} x_1 + \omega_2 x_2 + \epsilon \end{aligned} \tag{2.33}$$

Megnézhetjük, hogy az így kapott modell erősebb-e az eredetinél. Ha igen, akkor határozott előnye van annak, hogy x_1 hatását úgy modelleztük, hogy x_2 növekedésével nő. Pl. elképzelhető, hogy az újsághirdetések hatékonyabbak, ha mellettük rádióban is hirdetünk.

2.6. Lineáris regresszió kiterjesztései

A lineáris regresszió alapesetben hajlamos a túltanulásra (nagy különbséget eredményez a training és test error között). Alapból ugyanis minimálisra próbálja szorítani a négyzetes hibák összegét, ami nem biztos hogy ideális. Ha kb. annyi vagy kevesebb adatunk van, mint szabadságfok, nem nagyon tudunk jót mondani. Például:

- Két darab két-dimenziós pontunk van, ez alapján adnánk egyenest → alulspecifikált probléma!
- Gének és betegségek közötti összefüggésekre vagyunk kíváncsiak. Csakhogy az emberi genóm 20.000 génből áll → 20.000 szabadságfok, amihez sose lesz elég adatunk...

A továbbiakban bemutatott néhány kiterjesztés lényege, hogy:

- Lineáris regressziót csinálunk, de bizonyos megoldásokat (főleg a túl bonyolultakat!), ilyen vagy olyan módon elkerülünk
- A lényeg, hogy szisztematikusan csökkentsük a szabadságfokokat (pl. csak az 5 legjelentősebb gént vegyük figyelembe)

2.6.1. Forward stepwise regression

Ha visszaemlékszünk, nagyon leegyszerűítve az ordinary least squares (OLS) lényege, hogy az együtthatók mátrixát transzformáljuk-invertáljuk:

$$\begin{aligned} &\mathbf{X}\omega = \mathbf{y} \\ &\mathbf{X}^T\mathbf{X}\omega = \mathbf{X}^T\mathbf{y} \\ &\omega = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{2.34}$$

A célunk most az, hogy ne csak egy modellt adjunk, hanem több, eltérő komplexitásút. Például lehetséges, hogy minden lépésben csak *n_col* számú oszlopot használunk fel *D*-ből. Kiválasztjuk az összes oszlopból azt az *n_col* méretű részhalmazt, amire a **legjobb out-of-sample errort** kapjuk. Ez egy *n_col* bonyolultságú modellt eredményez. Ezt a módszert *n_col* növelésével lehet folytatni

A probléma csak az, hogy fix *n_col* értéknél exponenciális számú variációt végig kell nézni. Pl. ha 10 attribútumunk van (10 oszlop **X**-ben), ennek már 1000 fölötti különböző részhalmaza van. Ha minden lehetséges *n_col*-t veszünk 1-től 10-ig, minden egyes részhalmazt is valamikor, pontosan egyszer figyelembe kell vennünk! A **forward stepwise regression** ezen úgy segít, hogy a korábbi lépésből a legjobbat megtartja, és mindig csak 1 oszlopot vesz hozzá a korábbiakban már meglévőkhöz.

2.6.2. Ridge regression

A **ridge regression (RR)** egy másik módszer az OLS módosítására, amely révén a bonyolultság szabályozható és a túltanulás elkerülhető. RR-nél a minimalizálandó OLS kifejezést olyan büntetőtaggal egészítjük ki, amelyik a túlzottan bonyolult megoldásokat rontja:

$$\omega^* = \arg\min_{\omega} \left[\frac{1}{M} \sum_{m=1}^{M} \left(y^{(m)} - \omega^T \mathbf{x}^{(m)} \right)^2 + \alpha \omega^T \omega \right] \tag{2.35}$$

A büntetőtag akkor lesz nagy (akkor nehezebb minimalizálni), ha sok olyan tényező van a súlyvektorban, amelyik 0-tól nagyon különbözik, tehát a RR a bonyolult megoldásokat ilyen módon bünteti! Az α

együttható szabályozásával állítható a büntetőtag szerepének mértéke, így közvetetten a megoldások bonyolultsága.

Ugyanezek a módszerek klasszifikációs problémára is minden további nélkül alkalmazhatóak (csak a hiba mércéje lesz más, pl. ROC és különösen AUC). Ráadásul az AUC értelmezését több-osztályos klasszifikációra is kiterjesztették (Hand & Till, 2001).

Az eddigi két módszer további ötleteket is szült a szakirodalomban: ezek a **büntetéses lineáris regressziós módszerek** (penalized linear regression methods) – amikor ún. **regularizáló tagot** (regularizer) adunk hozzá a minimalizálandó kifejezéshez. A ridge regression is már ennek az osztálynak egy példája, hiszen azt segíti elő, hogy sok együttható értéke kicsi legyen.

2.6.3. LASSO regression

A ridge regression lényegében ugyanaz, mint amikor attribútumokat elhagyunk, mert ennek hatása hasonló ahhoz, mint amikor sok együttható értéke kicsi. Egy másik módszer az ún. LASSO (Least Absolute Shrinkage and Selection Operator). Ez a megoldás a Manhattan-hosszt, vagyis L1-normát használja:

$$\omega^* = \arg\min_{\omega} \left[\frac{1}{M} \sum_{m=1}^{M} \left(y^{(m)} - \omega^T \mathbf{x}^{(m)} \right)^2 + \alpha \|\omega\|_{\mathcal{L}\mathbf{1}} \right] \tag{2.36}$$

Mit csinál másképp a két módszer? Röviden: RR sok kis értékű együtthatót, a LASSO pedig inkább a sok 0 értékűt preferálja. A különbséget az okozza, hogy a két esetben a büntetések ekvivalens felületeinek (iso-felületeinek) paramétertérbeli topológiája más, ahogy az 2.3 ábra mutatja.

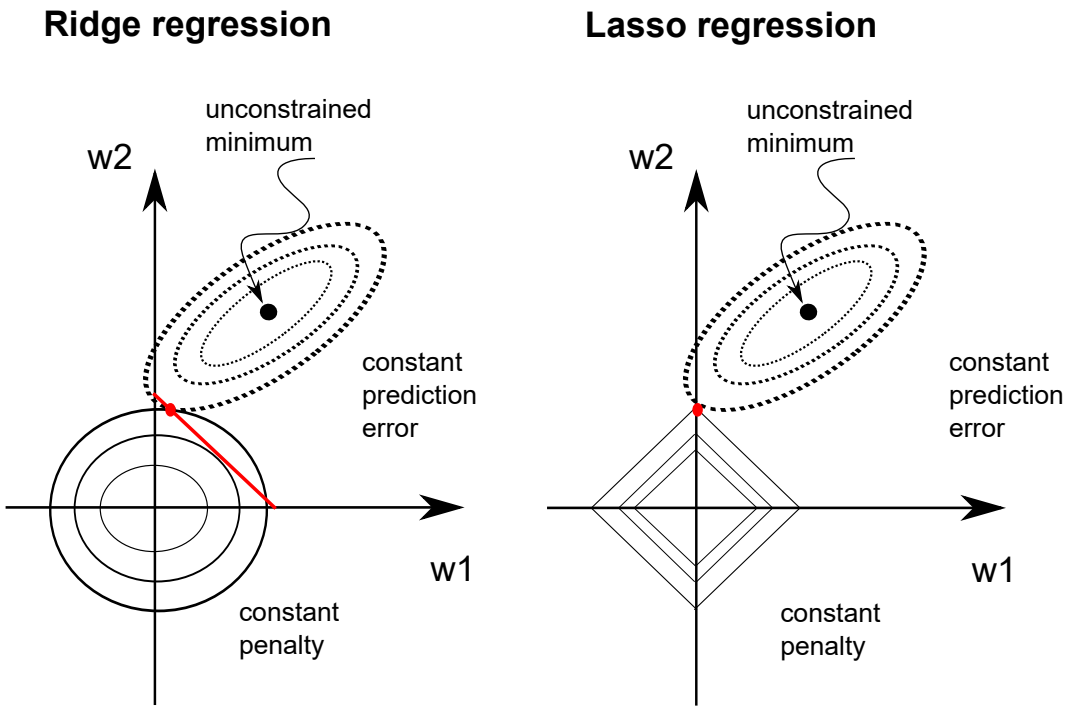
2.6.4. ElasticNet

Az ElasticNet megfogalmazás a Ridge és LASSO büntetőtényezőt ötvözi. Az ElasticNet-ben két hangolható paraméter van:

- A büntetőtagok össz-szerepe (λ)
- Ezen szerepen belül a ridge és lasso büntetés megoszlása (α)

A kérdés most már csak az, hogyan lehetne ezt az optimalizációs feladatot megoldani? Első ötletünk az lehetne, hogy deriváljuk a költségfüggvényt. Ha a költségfüggvény konvex, és csak egy minimum van → rögtön egyenlővé tehetjük 0-val és megoldhatjuk a megfelelő súlyvektorra. Ha a költségfüggvény bonyolultabb, vagy nem deriválható mindenütt → a lokális derivált (ha létezik) töredékét kivonjuk az aktuális súlyvektorból, és inkrementálisan javítjuk.

A sima ridge regressziót meg lehet oldani analitikusan is (zárt képlettel). Ugyanis a hiba (penalized residual sum of squares):



2.3. ábra. Ridge és LASSO regression közötti különbségek. LASSO esetben a minimális hiba pontja nem lehet ugyanaz, mint a Ridge regression esetében, ugyanis mivel ezen pont rombuszként történő kiegészítésének sarokpontja már az iso-felület belsejében van, ezért ez a sarokpont kisebb teljes hibát eredményezne (ld. még [1], amelyben szereplő ábra reprodukálásaként jött létre ez az ábra).

$$\begin{aligned} PRSS(\omega) &= \sum_{m=1}^{M}(y^{(m)} - \omega^T \mathbf{x}^{(m)})^2 + \alpha \sum_{d=1}^{D} \omega_d^2 = \\ &= (\mathbf{y} - \mathbf{X}\omega)^T(\mathbf{y} - \mathbf{X}\omega) + \alpha\|\omega\|_{\mathcal{L}2}^2 \end{aligned} \tag{2.37}$$

majd annak deriváltja 0-val egyenlővé téve:

$$\begin{aligned} \frac{\partial}{\partial\omega} PRSS(\omega) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\omega) + 2\alpha\omega = 0 \\ \Rightarrow \mathbf{X}^T\mathbf{y} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})\hat{\omega} \\ \hat{\omega} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{2.38}$$

A kapott kifejezése hasonlít az OLS megoldására, de van benne még egy paraméter.

Általánosabb esetben (mint Lasso, vagy ElasticNet) viszont **nem létezik zárt alak a megoldásra** (az abszolútérték-függvény ugyanis nem differenciálható)! Ilyenkor valamilyen iteratív optimalizációs eljárást kell használni. A **gradient descent** az a módszer, amikor mindig a derivált (gradiens) irányában (annak -1-szerese irányában) módosítjuk az együtthatókat:

$$\theta^{t+1} = \theta^t - \gamma\nabla_\theta \text{ , ahol } 0 < \gamma \leq 1 \tag{2.39}$$

Ezzel a naív módszerrel azonban több probléma is lehet:

- 1. Ha a gradiens kicsi, jó volna nagyobbakat lépni, mint amikor nagy. De a módszer alapból pont fordítva csinálja.
- 2. Ugyenez dimenziók között is igaz: ha egy hosszú "völgyben" vagyunk, inkább kellene a nagy gradiens mentén kicsit lépni, a kis gradiens mentén pedig nagyot.

Ezzel a témakörrel egy külön tudományterület foglalkozik (nem-konvex optimalizáció). Mivel azonban elég gyakori a büntetéses lineáris tanulás, specializált algoritmusok is léteznek. Két ilyen algoritmus a LARS és a Glmnet, melyek gyorsak és hatékonyak.

Optimalizáció megoldása LARS módszerrel

LARS = Least-Angle Regression. Kidolgozói Efron, Hastie, Johnstone és Tibshirani. Felfogható úgy is, mint a Forward Stepwise Algorithm (FSR) finomítása. Az FSR olyasmi volt, hogy:

- Kezdetben minden $\omega_d = 0$
- Minden lépésben kiszámítjuk az aktuális hibát, és...
- ... megkeressük azt a változót, ami ezt a hibát a legjobban „megmagyarázza"

Ehhez képest a LARS esetében:

- Kezdetben minden $\omega_d = 0$
- Minden lépésben megkeressük azt a változót, amelyik értéke a leginkább korrelál a hibával, és pozitív korreláció esetén az együtthatóját kicsit megnöveljük; negatív esetén kicsit csökkentjük

Ennek magyarázata, hogy hiba = elvárt_kimenet - kapott_kimenet. Ha egy változó ezzel korrelál:

- Pozitívan: amikor a változó értéke nő, a hiba is nőni fog (a kapott kimenet egyre kisebb lesz, mint kéne) $\rightarrow$ *ha azt akarjuk hogy a hiba konstans maradjon a változótól függetlenül, ezt a kimenet-csökkentést ellensúlyozandó növeljük az együtthatót*
- Negatívan: amikor a változó értéke nő, a hiba mértéke csökkenni fog (a kapott kimenet egyre nagyobb lesz) $\rightarrow$ *ha azt akarjuk hogy a hiba konstans maradjon a változótól függetlenül, ezt a kimenet-növekedést ellensúlyozandó csökkentjük az együtthatót*
- Így elérhetjük, hogy amint a súlyok egyre jobbak, az adott változó hibával vett korrelációja csökkenni fog, és más változók válnak fontossá.

Példa. Az egyenlet, amit keresünk $\omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$. Három tanítópont van, az alábbi $\mathbf{x}$ vektorokkal:

$$\begin{aligned} 1.2\omega_1 + \omega_2 + 2\omega_3 &= 9.3 \\ 2\omega_1 + 2\omega_2 + 2.2\omega_3 &= 12.8 \\ \omega_1 - \omega_2 + 0.3\omega_3 &= -0.2 \end{aligned} \tag{2.40}$$

```python
import math
import numpy

x1s_orig = [1.2, 2, 1]
x2s_orig = [1, 2, -1]
x3s_orig = [2, 2.2, 0.3]
ys_orig = [9.3, 12.8, -0.2]

def scale(xs):
    m = numpy.mean(xs)
    s = numpy.std(xs)
    retxs = [(x-m)/float(s) for x in xs]
    return retxs

x1s = scale(x1s_orig)
x2s = scale(x2s_orig)
x3s = scale(x3s_orig)
ys = scale(ys_orig)
```

```python
def compute_errors(w1, w2, w3):
    errs = [ys[i] - (x1s[i]*w1 + x2s[i]*w2 + x3s[i]*w3) for i in range(3)]
    corrs = [xes[i][0]*errs[0] + xes[i][1]*errs[1] + xes[i][2]*errs[2] for i in range(3)]
    return (errs, corrs)

def stop(errs):
    if all([errs[i] < 0.001 for i in range(len(errs))]):
        return True
    else:
        last_three_errs_norm = [lte - numpy.mean(last_three_errs) for lte in last_three_errs]
        if all([last_three_errs_norm[i] < 0.001 for i in range(len(last_three_errs_norm))]):
            print('last three errors normed: %r' % last_three_errs_norm)
            return True
    return False

def sgn(x):
    return x/abs(x)

ws = [0,0,0]
alpha = 0.05
errs = [1,1,1]
last_three_errs = [10, 20, 30]


print('weights\t\t\t\terrors\t\t\t\tcorrelations')

while True:
    errs, corrs = compute_errors(*ws)
    abscorrs = [abs(i) for i in corrs]
    inxmax = abscorrs.index(max(abscorrs))
    if stop(errs):
        break
    ws[inxmax] = ws[inxmax] + sgn(corrs[inxmax])*alpha
    last_three_errs = last_three_errs[1:] + [sum([err**2 for err in errs])]
    print('%.2f %.2f %.2f\t\t%.2f %.2f %.2f\t\t\t%.2f %.2f %.2f' % (ws[0], ws[1], ws[2], errs[0], errs[1], errs[2], corrs[0], corrs[1], corrs
```

```
weights                 errors                  correlations
0.00 0.05 0.00          0.36 1.00 -1.37         2.49 2.99 2.96
0.00 0.10 0.00          0.35 0.95 -1.30         2.36 2.84 2.81
0.00 0.15 0.00          0.34 0.89 -1.23         2.23 2.69 2.67
0.00 0.20 0.00          0.32 0.84 -1.17         2.10 2.54 2.52
0.00 0.25 0.00          0.31 0.79 -1.10         1.97 2.39 2.37
0.00 0.30 0.00          0.30 0.73 -1.03         1.84 2.24 2.23
0.00 0.35 0.00          0.28 0.68 -0.96         1.71 2.09 2.08
0.00 0.40 0.00          0.27 0.63 -0.90         1.58 1.94 1.94
0.00 0.45 0.00          0.26 0.57 -0.83         1.45 1.79 1.79
0.00 0.45 0.05          0.24 0.52 -0.76         1.32 1.64 1.65
0.00 0.50 0.05          0.21 0.48 -0.69         1.21 1.50 1.50
0.00 0.50 0.10          0.20 0.43 -0.63         1.08 1.35 1.35
0.00 0.55 0.10          0.17 0.38 -0.56         0.97 1.20 1.20
0.00 0.55 0.15          0.16 0.33 -0.49         0.84 1.05 1.05
0.00 0.60 0.15          0.13 0.29 -0.42         0.73 0.91 0.90
0.00 0.60 0.20          0.12 0.24 -0.35         0.60 0.76 0.76
0.00 0.65 0.20          0.09 0.20 -0.28         0.49 0.61 0.61
0.00 0.65 0.25          0.07 0.14 -0.22         0.36 0.46 0.46
0.00 0.70 0.25          0.04 0.10 -0.14         0.25 0.31 0.31
0.00 0.70 0.30          0.03 0.05 -0.08         0.12 0.16 0.17
0.00 0.75 0.30          0.00 0.01 -0.01         0.02 0.02 0.02
```

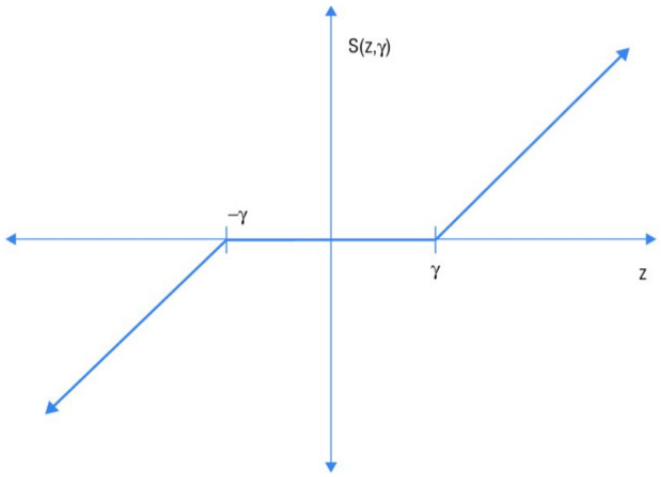
2.4. ábra. Példa LARS működésére Python nyelven.

Ezeket előbb oszloponként normalizáljuk, majd jöhetnek a számítások:

$$\begin{aligned} -0.46\omega_1 + 0.27\omega_2 + 0.59\omega_3 &= 0.36 \\ 1.39\omega_1 + 1.07\omega_2 + 0.82\omega_3 &= 1 \\ -0.93\omega_1 - 1.34\omega_2 - 1.41\omega_3 &= -1.37 \end{aligned} \tag{2.41}$$

Ezt követően jöhetnek a számítások, melyeket az 2.4. ábrán láthatunk.

Összességében a LARS kicsit mást csinál, mint az eddigi módszerek, de a gyakorlatban olyan mint a LASSO. Az együtthatók növekedése apró mértékben történik, de mindig azt az együtthatót módosítjuk, amelyik a legtöbbet mondja a hibáról $\rightarrow$ sok kis együttható lesz.



2.5. ábra. Soft threshold függvény.

Optimalizáció megoldása Glmnet módszerrel

A Glmnet algoritmust Friedman *et al* dolgozták ki a Stanfordon. Emlékezzünk vissza, hogy az ElasticNet modell miből áll: egy λ paraméterből, amelyik a büntetőtag relatív erősségét határozza meg; és α paraméterből, amelyik ha 0, csak Ridge, ha pedig 1, akkor csak LASSO regressziót használunk.

A Glmnet fő képlete, amit frissítéshez használunk miközben λ értékét folyamatosan csökkentjük:

$$\begin{aligned} &\omega_d \leftarrow \frac{S\left(\frac{1}{M}\sum_{m=1}^{M} x_d^{(m)} r_m, \lambda\alpha\right)}{1+\lambda(1-\alpha)} \\ &\text{ahol } r_m \text{ az ún. partial residual és} \\ &S(a,b) \text{ a soft-thresholding operátor, melyre:} \\ &S(a,b) = \begin{cases} a-b & \text{, ha } a>0 \text{ és } b<|a| \\ a+b & \text{, ha } a<0 \text{ és } b<|a| \\ 0 & \text{, ha } b\geq|a| \end{cases} \end{aligned} \tag{2.42}$$

Ez a soft-thresholding operátor ezt jelenti (2.5. ábra)

Vagyis:

- 1. Ahogy lambda értékét csökkentjük, a büntetőtag súlyát csökkentjük és egyre komplexebb megoldásokat kapunk
- 2. Eközben a soft-threshold op. második paramétere egyre kisebb lesz → egyre inkább kapunk 0-tól eltérő frissítéseket az $S()$ függvény kimeneteként.
- 3. Ekkor szerepet kap a reziduálissal (hibával) való korrelációja az egyes attribútumoknak (mint LARS-nál!) Itt azonban a korreláció hatása átmegy ezen a soft-threshold op.-on és az összes súlyt módosítjuk.

Hogy jön ez ki? (vázlat). Minimalizálni kell:

$$\min_{\omega}\left[\frac{1}{2M}\sum_{m=1}^{M}(y^{(m)}-\omega^T\mathbf{x}^{(m)})^2+\lambda((1-\alpha)\frac{1}{2}\|\omega\|_{\mathcal{L}2}^2+\alpha\|\omega\|_{\mathcal{L}1})\right] \tag{2.43}$$

Ehhez nézzük a gradienst. Ha a d-edik súly pozitív:

$$\begin{aligned}\frac{\partial}{\partial\omega_d}&=-\frac{1}{M}\sum_{m=1}^{M}x_d^{(m)}(y^{(m)}-\omega^T\mathbf{x}^{(m)})+\lambda(1-\alpha)\omega_d+\lambda\alpha=0\\ \Rightarrow\omega_d&=\frac{\frac{1}{M}\sum\limits_{m=1}^{M}x_d^{(m)}r_m-\lambda\alpha}{\lambda(1-\alpha)}\end{aligned} \tag{2.44}$$

Ha a nevezőhöz 1-et hozzáadunk, az annak szól hogy eleve approximációról beszélünk, és kisebb lépésekben frissítünk. Vagyis:

$$\Delta\omega_d\propto\frac{1}{M}\sum_{m=1}^{M}x_d^{(m)}r_m-\lambda\alpha \tag{2.45}$$

Ha a d-edik súly negatív, az utolsó előjel megváltozik (ugyanis a súly növelésével a minimalizálandó képlet utolsó, abszolút értékes tagja nem nőne, hanem csökkenne):

$$\begin{aligned}&\frac{\partial}{\partial\omega_d}=-\frac{1}{M}\sum_{m=1}^{M}x_d^{(m)}(y^{(m)}-\omega^T\mathbf{x}^{(m)})+\lambda(1-\alpha)\omega_d-\lambda\alpha\Rightarrow\\ &\ldots\\ &\Delta\omega_d\approx\frac{1}{M}\sum_{m=1}^{M}x_d^{(m)}r_m+\lambda\alpha\end{aligned} \tag{2.46}$$

A gyakorlatban λ kezdeti értéke akkora, hogy az összes súly értéke 0 legyen. Ehhez azt kell kiszámítani, hogy melyik oszlop értéke korrelál leginkább a hibával amikor minden súly 0, és ehhez kell illeszteni λ értékét (hogy $\lambda\alpha$ nagyobb legyen ennél a korrelációnál). Ezt követően általában olyan szorzóval csökkentjük lambdát, hogy a 100. iterációban legyen kb. 0.001-szerese az értéke az eredetinek. Ez szorzó kb. 0.93. De ha a tanulás nem konvergál, akkor közelebb kell vinni 1-hez.

2.7. Mikor használjunk (büntetéses) lineáris regressziót, és mikor inkább más módszert?

A büntetéses lineáris regresszió előnyei:

- Rendkívül gyors tanítási és kiértékelési fázis
- Pontos információ a változók fontossági sorrendjéről
- Sokféle problémán jó teljesítmény: főleg amikor több oszlop van mint sor, vagy az attribútum mátrix ritka
- Sokszor kötelező lineáris modellt adni. Pl. egy biztosítási szerződésbe nem írhatunk bele egy ezer döntési fából álló szabályt, vagy egy 10-rétegű neurális háló paramétereit!

Ebből következően: mikor lehet jobb más módszer?

- Ha a probléma eleve nagyon bonyolult, vagy nehezen fejezhető ki a lehetséges megoldás szabályokkal (pl. gépi látás: hogyan ismerjünk fel egy széket egy fényképen – melyik pixel-érték lesz jobban mérvadó, mint a többi?)
- Ha rengeteg adatunk van és hozzá képest kevés változó
- Látni fogjuk (később), hogy a döntési fák külön jók a változók közötti összefüggések felderítésében (pl. A és C attribútum együtt fontosabb, mint ha fontosságukat külön-külön összegeznénk) $\rightarrow$ ez hasznos info akkor is, ha végül lineáris modellt használunk!

3. fejezet

Együttes módszerek

Forrásmegjelölés

A fejezetben számos magyarázat és példa az alábbi angol nyelvű referenciákból származik: [1].

Az együttes módszerek mögötti alapötlet, hogy több modell együtt jobb eredményt adhat, mint 1 – legalábbis, amennyiben a modellek függetlenek egymástól!

Például, ha egy osztályozó az esetek 55%-ában jó eredményt ad (két kategória között), azt gyenge osztályozónak nevezhetjük. Ha azonban egy ilyen osztályozóból van 99 különböző modellünk, és ezeknek a tévedései függetlenek egymástól, akkor a helyes döntés valószínűsége felmegy 84%-ra!

Ugyanis:

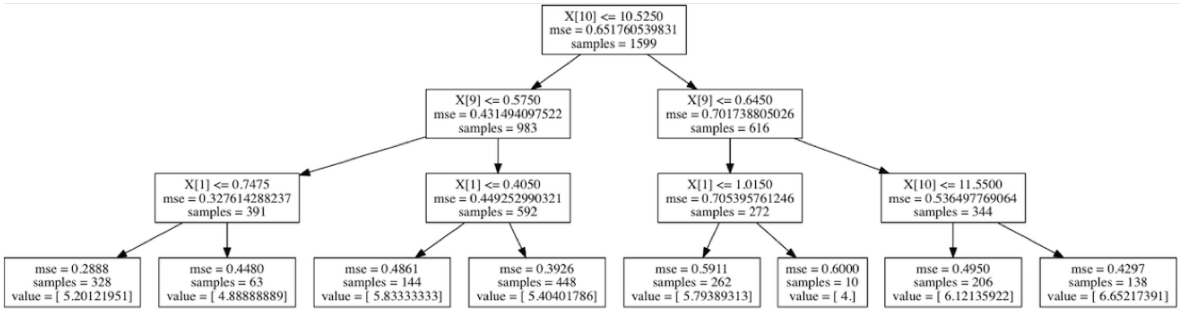
$$\begin{aligned} p(\text{hibás}) &= \sum_{i=50}^{99} \binom{99}{i} 0.45^i \times 0.55^{(99-i)} = \\ &= \sum_{i=50}^{99} \frac{99! \times 0.45^i \times 0.55^{(99-i)}}{i!(99-i)!} \approx 0.156 \end{aligned} \tag{3.1}$$

Persze ehhez feltételeztük, hogy egymástól függetlenül hibáznak a modellek (nincs valami közös, implicit feltételezés mögöttük). Ha nem így lenne, rosszabb lenne a helyzet, mert minden egyes modellre, amelyik téved, egyre nagyobb lenne a valószínűsége, hogy a többi is.

A kérdés tehát, hogy hogyan kaphatunk sok, egymástól lényegesen különböző modellt?

Erre az egyik lehetőség, hogy többféle algoritmust használunk (pl. lineáris regressziót, SVM, kNN, bináris döntési fákat, ...). Ez azonban költséges lenne, mert minden esetben specializált modellekre lenne szükség, és így biztosan nem tudnánk annyira sok modellt létrehozni (pl. többszázezret nem tudnánk összegyűjteni).

Egy másik lehetőség ezért, hogy egy olyan algoritmust használunk, ami sok független modellt képes generálni. A meglepő, hogy ilyen modell könnyen készíthető és ez a gyakorlatban is hatékony.



3.1. ábra. A [1] könyvből származó ábra a bináris döntési fák bemutatására. A fát a gyökértől a levelek felé kell olvasni; minden csomópontjánál egy döntési elágazás található valamely dimenzió valamilyen küszöbértéke szerint.

3.1. Alaposztályozók és felsőszintű algoritmusok

Legyen tehát két algoritmusunk, hierarchiában:

- 1. alaposztályozó (base learner): egy gépi tanuló algoritmus, amellyel sokszázezer modellt készítünk. Ilyen lehet például a bináris döntési fa modell
- 2. felsőszintű algoritmus: az alaposztályozó paramétereit úgy módosítja, hogy a kapott modellek függetlenek legyenek egymástól. Tipikus példák: zsákolás (bagging), gyorsítás (boosting), véletlen erdők (random forests)

Szigorúan véve a véletlen erdők módszere egy felsőszintű algoritmusnak és a bináris döntési fák egy speciális kombinációja.

Alaposztályozóként szinte bármilyen, paraméterezhető algoritmus használható, de a leggyakoribb a bináris döntési fák módszere. Bináris döntési fára mutat példát a 3.1. ábra.

Egy szinten eltérő változók szerint is lehet elágazási döntést hozni, azonban a fő kérdés, hogy hogyan lehet egy bináris döntési fát betanítani?

3.2. Döntési fák tanítása

A taításhoz a fa minden szintjén, minden csomópont esetén ún. *split point*-okat (elágazási pontokat) kell választani. Ehhez tudni kell, hogy melyik dimenzióban és melyik küszöbérték szerint ágazzunk el.

3.2.1. Folytonos kimenetekre való tanítás

Az alapötlet: folytonos kimeneteknél mindegyik dimenzióra végignézzük a négyzetes hibaösszeget, és azt választjuk, amelyiknél a legjobban minimalizálható a hiba. Ha adott dimenzióban a tanítóhalmazokban nincs ismétlődő érték, akkor N különböző tanítópont esetén a lehetséges küszöbértékek száma N-1 lehet.

Ennek a módszernek azonban hátulütői is vannak:

- ahogy nő a dimenziók száma, kezelhetetlenné válik
- ráadásul nagy adathalmazokban a lehetséges töréspontok nagyon közel is eshetnek egymáshoz

A fentiek okán gyakran approximatív és párhuzamosítható algoritmusokat használnak a gyakorlatban.

A továbblépéshez azonban vizsgáljuk meg azt is, mi történik, ahogy nő a fa mélysége?

Ilyenkor az egyes szintekre egyre kevesebb pont jut – kifogyhatnak, mire a végére jutunk! Általában ezért beállítható, mi az a legkisebb számú adatpont, amit már nem osztunk ketté.

3.2.2. Kategorikus kimenetekre való tanítás

Osztályozásnál hasonló módszert követhetünk, csak a hibametrika lesz más (nem MSE). Lehet például:

- Két osztály esetén AUC
- Több osztály esetén:
  - félreosztályozások rátája (misclassification rate)
  - Gini impurity measure
  - Information gain
  - AUC általánosítása (Hand & Till, 2001)

. . . melyek közül többféle módszer megtalálható az 1.5 fejezetben.

3.3. Tipikus felsőszintű algoritmusok

Felsőszintű algoritmusokra azért van szükség, hogy a bináris döntési fa (azaz BDF, vagy más alaposztályozó) paramétereit (vagy a bemenő adathalmazt) módosítva sok, egymástól független módon hibázó BDF jöjjön ki.

Néhány gyakori módszer, amelyek az 1990-es évektől kezdődően alakultak ki:

- Bagging (bootstrap aggregation)
- Gradiens alapú gyorsítás (gradient boosting)
- Véletlen erdők

3.3.1. Bootstrap aggregation

A ***Bagging („Bootstrap aggregation")*** feltalálója Leo Breiman. Ebben a módszerben mindig egy-egy „boostrap sample"-t generálunk a tanítóadatok mintavételezésével (visszahelyettesítéssel – *with replacement*: ilyenkor ugyanabból az adatpontból több példány is elfogadott

A bagging módszere gyakorlatilag bármilyen alaposztályozóval használható, mivel csak a tanítóadatok mintavételezéséről szól.

Azonban ha a lehetséges hibák fajtáit nézzük, könnyen belátható, hogy ezzel csak a variancia típusú hibát fogjuk tudni csökkenteni, hiszen maguk az alaposztályozók fixek, és a bagging módszere csak a tanítóhalmazok számának növelésével foglalkozik. Ha a kimeneteket átlagoljuk, a varianciát csökkenteni tudjuk – hiszen a variancia definíció szerint is éppen azt mutatja meg, hogy „ha más tanítómintáink lettek volna...", mennyire lenne más az eredmény.

Összefoglalóan a bagging nagyon egyszerű módszer, ezért bevezetőnek jó... de ennél szofisztikáltabb megoldásak is léteznek.

3.3.2. Gradient boosting

A ***Gradient boosting*** Jerome Friedman módszere (akit az ElasticNet-ből is ismerünk). A módszer lényege:

- Kategorizálás esetén: különböző címkékhez különböző osztályozókat társítunk → majd ezeket kombináljuk
- Regresszió esetén: az újabb osztályozókat mindig a korábbi osztályozók hibáira tanítjuk rá
- **A lényeg mindkét esetben: „szakértők gyűjteményét" hozzuk létre.** Mindegyik alaposztályozó egyvalamiben legyen jó, de abban tényleg.

Gradient boosting esetén a keletkezett különböző „szavazatokat" össze kell adni. Regresszió esetén ezt általában úgy csináljuk, hogy a legelső modell predikcióit (pontosabban: annak egy ϵ_1 töredékét!) kivonjuk az elvárt kimenetből, és a fennmaradó értékeket kell a következő modellel prediktálni. A következő modell kiértékelésénél már az $epsilon_1 * modell_1 + epsilon_2 * modell_2$ predikciót vonjuk le az eredetileg elvárt kimenetekből, és ezt így folytatjuk, miközben ϵ_i értéke egyre csökken. Mindez gyakorlatilag egyfajta gradiens-csökkentésként is felfogható.

Érdekesség, hogy – ellentétben a bagginggel – *a gradiens alapú gyorsítás (meg a véletlen erdők is, ld. később) csökkenteni tudja a bias típusú hibákat is.* Ugyanis ahogy az adathalmaz jellemző részein csökken a hiba, egyre jobban „oda fog figyelni" a szélekre is – mivel mindig a hibára tanul rá. Továbbá, ellentétben a bagginggel, itt figyelembe vesszük a kumulatív hibát is. Az újabb modellek ezen hiba leszorítását célozzák.

3.3.3. Véletlen erdők (Random forests)

A ***Véletlen erdők (Random forests, RF)*** Leo Breiman és Adele Cutler algoritmusa. A módszer lényege, hogy:

- Kicsit mint bagging esetén, sampling-with-replacementtel működik
- Ugyanakkor további random elem, hogy nemcsak a tanítópontoknak használja csak egy részét, hanem az attribútumoknak is csak egy részhalmazát veszi alapul minden modell esetében

Ez alapján nem mindegyik fa testesít meg azonos komplexitást → ez is a bias csökkentésének irányába hat (mint gradient boosting esetében a hibák felülreprezentálása).

3.4. Összefoglalás

Utolsó megjegyzés: noha az ensemble (együttes) tanulás lassabb mint a büntetéses regressziók, fontos benyomásokat adhatnak a változók közötti interekciók megértéséhez.

Egyrészt a fa mint alaposztályozó könnyen össze tud keverni több különböző dimenzióban levő döntéseket.

Másrészt a véletlen erdők módszere pl. minden alaposztályozóhoz kicsit más attribútum-kombinációt alkalmaz, ezért az egyes modellek hatékonysága külön-külön is sok információt adhat az interakcióban levő attribútumokról.

4. fejezet

Neurális hálók alapjai

4.1. Neurális modellek motivációja

Neurális hálókkal már a mesterséges intelligencia kezdeti korszakaiban is sokan foglalkoztak. A tématerület fontos inspirációját adta Warren McCullogh és Walter Pitts korai munkássága, akik már a 40-es években megmutatták, hogy bármely 0-adrendű (propozícionális) logikai kifejezés ekvivalens módon leírható egy egyszerű processzáló elemekből álló hálóval, és hogy ezek a hálók elvben egy univerzális Turing-gép számítási kapacitásával rendelkeznek [5, 8]. A 20. század későbbi periódusaiban különböző mértékben, de egyre vonzóbbá vált az az elképzelés, hogy egyszerű „sejtek" súlyozott összekötésével, és a sejtek aktivációinak súlyokon keresztüli továbbterjesztésével egy az emberi agy működéséhez hasonló szerkezethez juthatunk.

Az egyik legkorábbi és a maga idejében leginkább befolyásos neurális háló modell megalkotása Frank Rosenblatt nevéhez fűződik, aki 1958-ban dolgozta ki a **perceptron modellt** és annak tanulási algoritmusát [7]. A perceptron modell eredeti alkalmazási területe a képfelismerés volt (első hardverimplementációja egy 400 fotocellából álló bemeneti perifériával rendelkezett), egyes kutatók azonban a mesterséges intelligencia egészére alapvető módon kiható újító megoldásként tekintettek rá. Rosenblatt megmutatta, hogy perceptronok hálózata alkalmas bármilyen binárisan kifejezhető mintázatok osztályozására – e szerint nem volt pl. elvi akadálya, hogy egy perceptron-háló különböző binárisan kódolt fényképeket aszerint osztályozzon, hogy szerpel-e rajtuk valamilyen tárgy, vagy sem. Ugyanakkor abban az időben még nem volt ismert olyan általános algoritmus, amely a perceptron-hálók tanítására képes is lett volna.

1960-ban az ún. *Widrow-Hoff szabály* (más néven delta-szabály) megjelenésével tovább fejlődött ugyan a mesterséges neuronok (azaz: külön-külön egy-egy perceptron) tanításának elmélete [9], a fő probléma azonban továbbra is fennállt: nem létezett olyan ismert módszer, amely perceptron-hálózatban alkalmas lett volna a bemenetek és a rejtett asszociatív egységek közötti súlyok beállítására is, ezért a gyakorlatban csak olyan perceptron-hálókat lehetett alkalmazni, melyeknek mindössze egy bemeneti és egy kimeneti rétege volt – többrétegű hálót továbbra sem lehetett hatékonyan tanítani.

Az utolsó szög a perceptronok koporsójában Minsky és Papert híres „*Perceptrons*" c. könyve volt, amelyben a szerzők megmutatták hogy a többrétegűség követelménye márpedig fontos, továbbá bizonyos Boole-függvények reprezentálására a többrétegű perceptron-hálók is csak abban az eset-

ben képesek, ha a köztes rétegben létezik olyan sejt, amelyik mindegyik bemenettel (nem-nulla) összeköttetésben áll. Az elmélet szerint tehát a perceptron-hálókkal két probléma is adódott:

1. Sok esetben elengedhetetlen, hogy legyen bennük rejtett réteg, annak tanítására azonban továbbra sem létezett hatékony módszer.
2. Bizonyos függvények modellezéséhez olyan globális összeköttetésekre volt szükségük, mint amilyenek az emberi agyban kevéssé valószínű, hogy léteznek, és ellentmondott a kutatók intuícióinak, miszerint egy elegáns megoldás csakis lokális kötéseken keresztül működhet.

Ezen felismerés következtében a 70-es évek a neurális hálók háttérbe szorulását hozták. Azon kevesek, akik még hittek a neurális hálók létjogosultságában, más, elsősorban *asszociatív memória-modellekkel* kezdtek el foglalkozni, melyeket alkalmasnak véltek általános és specifikus információk közötti kapcsolatok modellezésére. Az elgondolás szerint ez a képesség fontos ahhoz, hogy például hézagos információk alapján is előhívhatóak legyenek valamilyen korábban eltárolt mintázatok – mindez az emberi agy működéséről szerzett ismeretekkel is összecsengett. További fontos elv, amely ebben a korszakban alakult ki, az ún. *versengés elve* volt, amely szerint a sejtek párhuzamosan lezajló dinamikája idővel felerősíti az aktivitásbeli különbségeket, ezáltal kiszűrve a zajok és egyéb ellentmondások hatását. Ebből a korszakból befolyásos háló-típusok például a Grossberg-féle **adaptív rezonancia elmélet** (angolul: adaptive resonance theory) és az arra támaszkodó modellek, a Kohonen-féle **önszerveződő térképek** (angolul: self-organizing maps), illetve a Hopfield által kidolgozott autoasszociatív **Hopfield-hálók**.

A neurális hálók igazi áttörése az 1980-as évek közepétől következett be. Rumelhart, Hinton és Williams ekkor javasolták a **backpropagation-algoritmust**, amely többrétegű neurális hálókban is alkalmazható akár a rejtett rétegbeli sejtek súlyainak hangolására is. *A módszer lényege, hogy a kimeneten kapott számszerűsíthető hiba alapján minden hálóban levő paramétert olyan mértékben frissítünk, amilyen mértékben hozzájárult a hibához.*

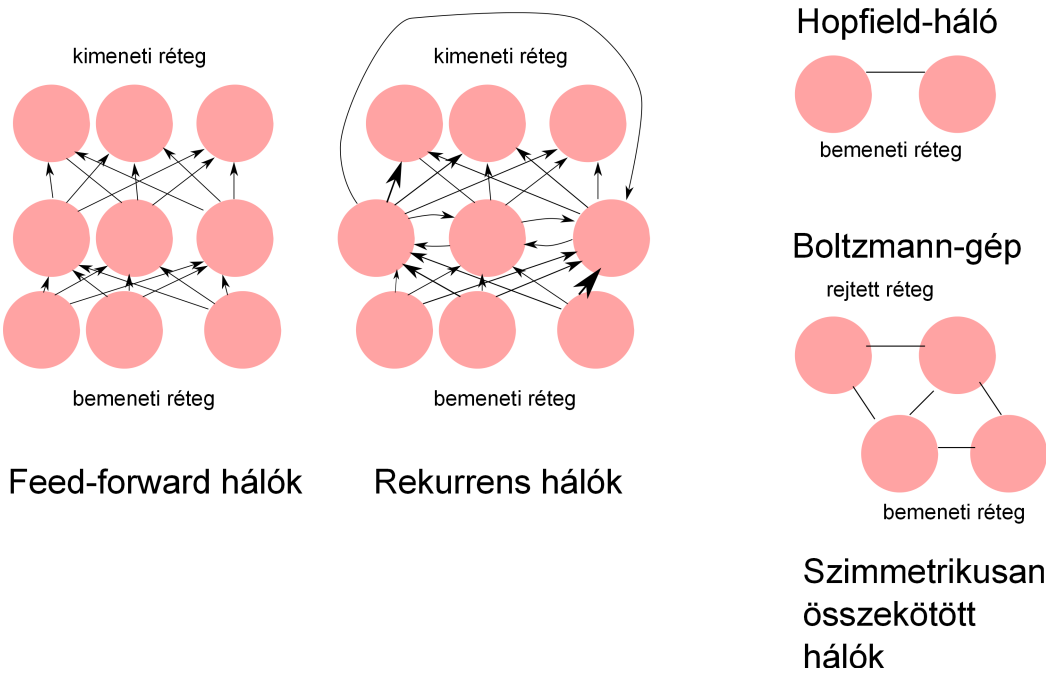
Később újszerű és hatékony módszerek születtek az autoasszociatív és rekurrens hálók tanítására is, így a korábban szerteágazó részterületek napjainkra összekapcsolódtak és elképesztő mértékű fejlődést tettek lehetővé mind a felügyelt, nem-felügyelt és megerősítéses gépi tanulásban.

A fejezeben először áttekintjük a főbb neurális-háló architektúrákat (*feed-forward hálók*, *rekurrens hálók* és *szimmetrikusan összekötött hálók*). Ezt követően bemutatjuk a főbb neuron-típusokat, melyek mindhárom architektúrában előfordulnak. A fejezetben bővebben a három architektúra közül a *feed-forward hálók* tanítását tekintjük át. Végül kis kitérőt teszünk a logisztikus regresszió bemutatására, amely nevében regressziós eljárás ugyan, de fontos párhumazokat tartalmaz a feed-forward neurális hálók témaköréhez.

4.2. Főbb neurális háló architektúrák

Három fő architektúrát különböztetünk meg, melyek a 4.1. ábrán láthatóak. Az architektúrák jellemzői az alábbiak szerint foglalhatóak össze:

- A **feed-forward hálók** két vagy több sejtréteg összekötésével jönnek létre olyan módon, hogy a kötések irányítottak és nincsenek bennük hurkok vagy körök (magyarán ha egy sejttől el-

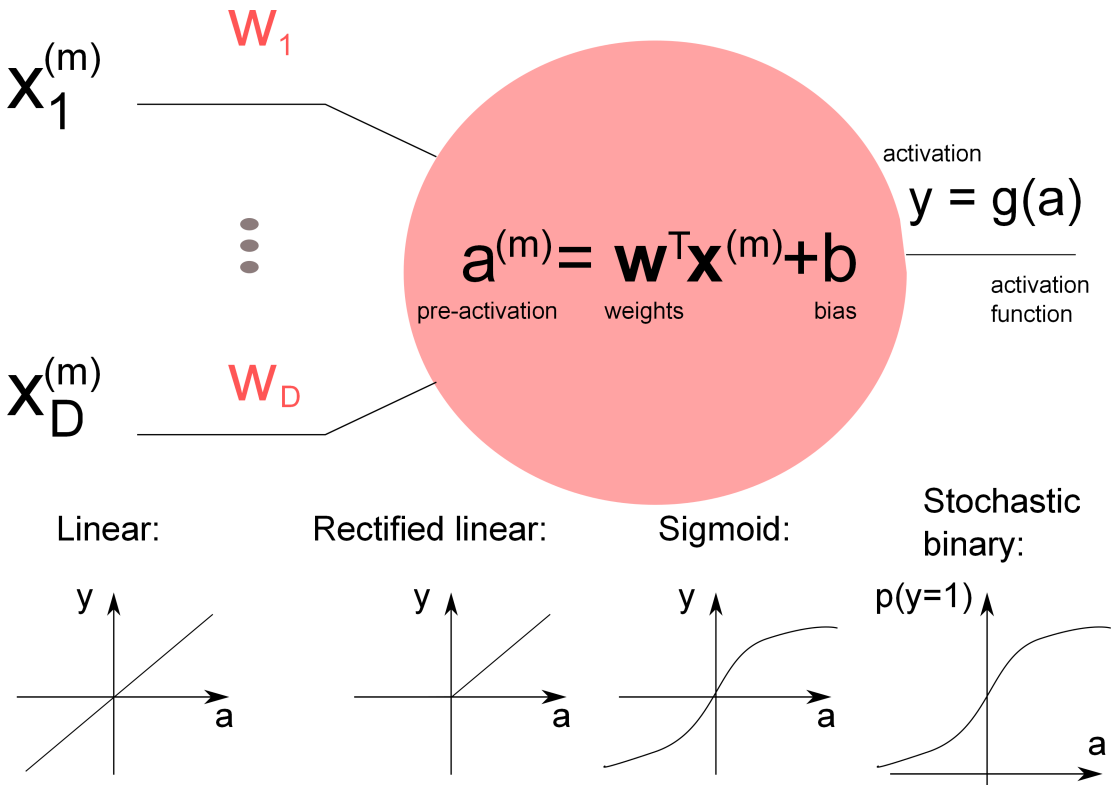


4.1. ábra. Neurális hálók típusai.

indulva az élek mentén több lépcsőn keresztül „bolyongunk”, nem állhat elő olyan eset hogy az eredeti sejthez visszaérünk). Minden rétegből kiinduló kötések egy későbbi rétegbe kell, hogy mutassanak. Elvileg nincs megtiltva, hogy egy sejtből kiinduló kötés egy olyan sejtbe mutasson, amely egy jóval fentebbi rétegben foglal helyet, azonban legtöbb esetben minden él szomszédos rétegek között húzódik. Feed-forward hálók esetében külön beszélhetünk *bementi rétegről*, *kimeneti rétegről*, és egy vagy több *köztes rétegről*. Ha egy hálóban több köztes réteg szerepel, *mély hálónak* nevezzük.

- A **rekurrens hálók** hasonlítanak a feed-forward hálókra amennyiben szintén irányított kötéseket tartalmaznak, azonban a kötések struktúrájában hurkok, körök is megengedettek. Éppen a körök jelenléte miatt a rekurrens hálók dinamikája bonyolult lehet és általánosságban nehéz őket tanítani. Napjainkban azonban már több olyan rekurrens modell ismert, amely speciális esetként jól tanítható és igen hasznos pl. időbeni mintázatok modellezésére. Rekurrens hálók esetében nincs értelme kettőnél több rétegű hálókról beszélni, ugyanis egy ilyen többrégetú háló mindössze olyan speciális esete lenne a kétrétegű rekurrens hálóknak, melyben bizonyos rejtett sejtek között nincs kapcsolat (a rejtett sejtek nem alkotnak teljes gráfot).

- A **szimmetrikusan összekötött** hálók a rekurrens hálók olyan változatai, melyben a kötések kétirányúak. Hopfield és társai vették észre, hogy ezen hálók hatékonyabban taníthatóak, mint a rekurrens hálók, mivel modellező erejük korlátozottabb, továbbá viselkedésüket egy globálisan definiálható „energia-függvény” vezérli. A rejtett réteg nélküli szimmetrikusan összekötött hálókat szokás *Hopfield-hálónak* nevezni, a rejtett réteggel rendelkezőket pedig *Boltzmann-gépeknek*.



4.2. ábra. Egyszerű neuron modellje.

4.3. Lineáris, szigmoid és sztochasztikus neuronok

Korábbi jelölésünket megtartva nézzünk egy olyan neuront, amelyik D-dimenziós $\mathbf{x}^{(m)}$ tanítóminták ($m = 1..M$) alapján számítja $y^{(m)}$ kimenetét. Nagy általánosságban a neuron működését a 4.2. ábra mutatja be. Az ábrán látható egyrészt, hogy a sejt ún. preaktivációja egy bemeneti vektor és egy súlyvektor skaláris szorzatából valamint egy torzításból (ún. bias termből) áll össze. A formalizmusok egyszerűsítése érdekében gyakran a bias termet nem jelöljük külön, hanem egy $D+1$. súlyként képzeljük el, amelyhez tartozó bemenet mindig 1.

Az ábra bemutatja azt is, hogy az ún. aktivációs (kimeneti) függvény alkazatának függvényében eltérő tulajdonságú neuronokhoz juthatunk. Ha a g aktivációs függvény:

- lineáris, pl. $y = k_1 a + k_2$ akkor **lineáris neuronról** beszélhetünk;
- korrigált lineáris, vagyis alakja:

$$y = \begin{cases} 0, \text{ ha } a < k_2 \\ k_1(a - k_2) \text{ különben} \end{cases} \tag{4.1}$$

akkor **rectified lineáris (RELU = rectified linear unit)** neuronról beszélhetünk. Mivel a felhasznált sejtek sokszor eleve tartalmaznak torzítást, ezért k_2 értéke gyakran 0. Ha feltételezzük továbbá hogy az aktivációs függvény meredeksége 1, akkor a következő szokásos leíráshoz jutunk: $y(a) = max(0, a)$;

- s-alakú (szigmoid), pl.

$$y = \frac{1}{1 + exp(-a)}, \text{ vagy}$$
$$y = tanh(a) = \frac{exp(a) - exp(-a)}{exp(a) + exp(-a)} = \frac{exp(2a) - 1}{exp(2a) + 1} \tag{4.2}$$

  akkor **szigmoid neuronról** beszélhetünk. A szigmoid függvények közkedvelt aktivációs függvények, mert jól modellezik azt, amikor egy sejt karakterisztikája a lényeges bemeneteknél közel lineáris, az azoktól eltérő bemeneteknél pedig szaturáló. A két fentebb említett függvény közül az első az ún. **logisztikus függvény**, melynek értékkészlete 0 és 1 közötti, ezért valószínűségként is könnyen értelmezhető;

- valószínűségként értelmezett (0 és 1 közötti értékkészlettel rendelkező) függvény, mely valószínűségnek megfelelően a kimenet 0 vagy 1, akkor **sztochasztikus bináris neuronról** beszélhetünk.

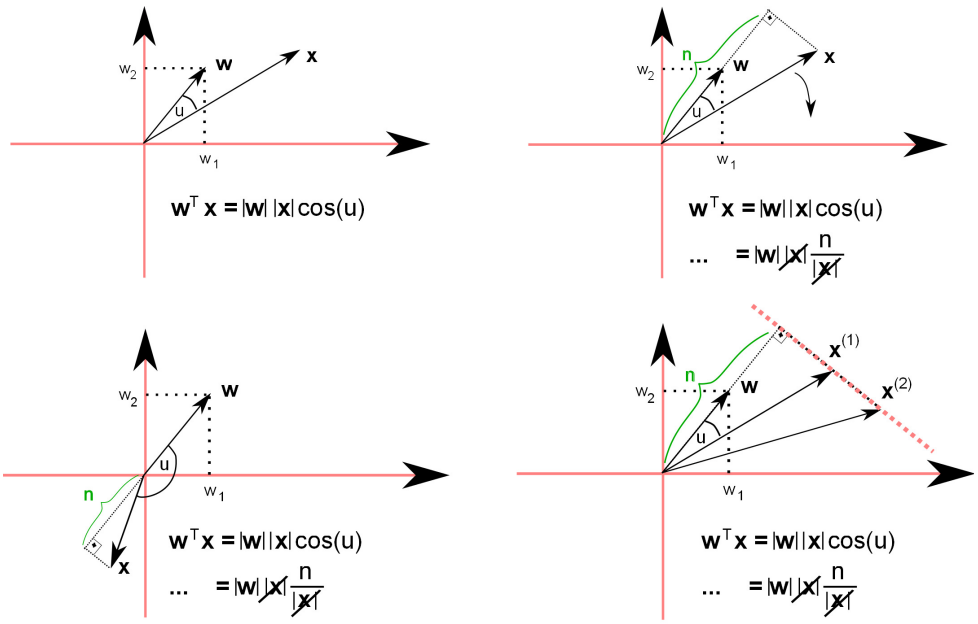
Abban az esetben, ha egy neurális háló kimeneti rétegében valamilyen kategorikus válaszra van szükségünk (például hogy egy fényképen kutya vagy macska vagy valamilyen más állat látható), gyakran alkalmazzuk az ún. **softmax aktivációs függvényt**. Ez a függvény az összes kimeneti sejt preaktivációjának exponenciális függvényét skálázza $0 - 1$ közötti tartományra. Ha K darab kimeneti sejtünk van, akkor a softmax függvény általánosan:

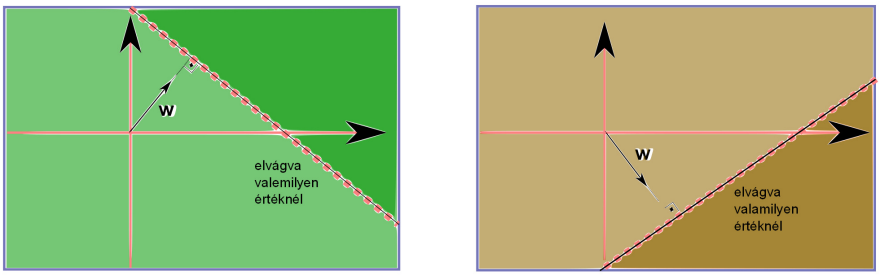
$$y_k = softmax(\mathbf{a})_k = \frac{exp(a_k)}{\sum_{k'=1}^{K} exp(a_{k'})} \tag{4.3}$$

ahol a nevezőben egy normalizáló kifejezés szerepel, melynek segítségével az összes lehetséges számláló fölött összegezve valószínűségként értelmezett kimeneteket kapunk. Ha például arról kell döntést hoznunk, hogy kutya vagy macska szerepel egy fényképben, $K = 2$ és a_1 illetve a_2 a megfelelő kimeneti sejtek preaktivációi.

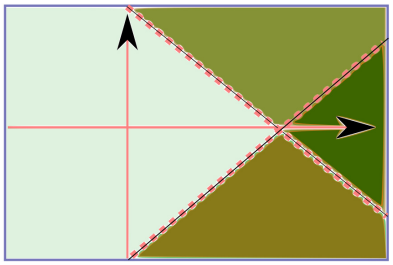
4.4. Neuronok és neurális hálók kapacitása

Egyetlen neuron egyetlen lineáris döntésre képes (mint látni fogjuk, ez akkor is igaz, ha az aktivációs függvénye nemlineáris de bemenete szerint monoton növekvő – a fentebb ismertett aktivációs függvények pedig ilyenek). Ennek belátását – legalábbis két-dimenziós esetben, ami több dimenzióra is általánosítható – a 4.3 ábra segíti. Az ábrán először a bemeneti vektor és súlyvektor skalárszorzatát a vektorok hosszának és a bezárt szög koszinuszának függvényében fejezzük ki. Egyszerűsítés után egy olyan kifejezést kapunk, amely csak a súlyvektor hosszától és egy olyan n-nel jelölt előjeles valós számtól függ, amelyik azt mondja meg hogy mennyivel kell a súlyvektort ahhoz megszoroznunk, hogy éppen egy olyan derékszögű háromszög oldalát kapjuk meg, melynek átfogója éppen a bemeneti vektor. A második sor bal oldali részábráján látszik, hogy ha a bezárt szög meghaladja a 90 fokot (és a 180 fokot nem), n előjele negatív lesz. Mindazonáltal a skalárszorzat értéke minden olyan bemeneti vektor esetén konstans, amelyhez tartozó derékszögű háromszög oldalát a súlyvektor ugyanolyan





tehát: egy neuron csak ilyesmire képes....



több neuron kombinálásával nemlineáris döntéseket is hozhatunk...

4.3. ábra. Neuronok és neurális hálók kapacitása

mértékű meghosszabbításával kapjuk. Például a második sor jobb oldali ábráján $\mathbf{x}^{(1)}$ és $\mathbf{x}^{(2)}$ egyazon $\mathbf{w}$ súlyvektorral azonos skalárszorzatot adnak.

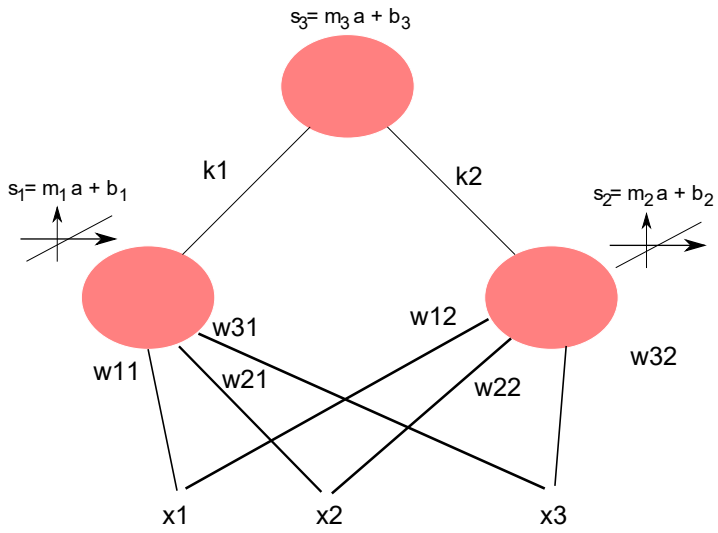
Az ábra alsó felén látszik, hogy ha egy ilyen skalárszorzatot akár nemlináris, de monoton növekvő aktivációs függvénnyel transzformálunk, majd egy döntési küszöbnél elvágunk, összességében lineáris döntést hozunk. Több neuron kombinációjával azonban összességében nemlineáris döntés is könnyedén hozható. Ha például a harmadik sor bal oldali sejtjének $0-1$ (bal-jobb) kimenetét 5-tel, a jobb oldali sejtjének $0-1$ (bal-jobb) kimenetét pedig 10-zel szorozzuk majd a kettőt összegezzük, akkor a legalsó sejt által partícionált területek súlyai óramutató irányába balról elindulva 0, 5, 15 és 10 lesznek. Ha ekkor a döntés küszöbét 2-nél húzzuk meg, akkor éppen a bal oldali világoszöld területet választjuk el a többitől egy nemlineáris határolófelület mentén.

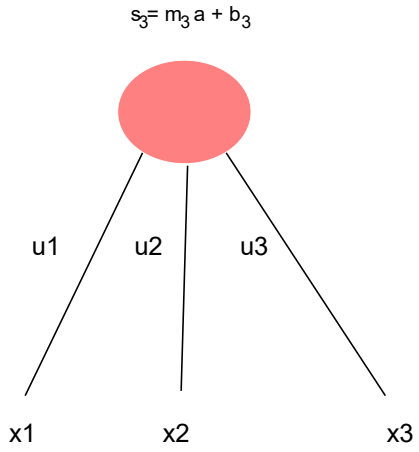
Fontos belátni, hogy ahhoz hogy többrétegű neurális háló nagyobb kapacitással rendelkezzen, mint egy rejtett réteggel nem rendelkező háló, a rejtett réteg aktivációs függvényei között kell, hogy nemlineáris függvény is szerepeljen. A 4.4. ábrán látható két neurális háló például ekvivalens. Ahhoz, hogy a rejtett neuronok többlet-kapacitást adjanak a hálóhoz, legalább az egyiknek nemlineáris aktivációs függvénnyel kell rendelkeznie (akkor is, ha ez a nemlinearitás pusztán küszöbölést és bináris kimenetet jelent, mint a 4.3. ábra alján látható példában).

4.5. Feedforward hálók tanulása backpropagation algoritmussal

A klasszikus feed-forward neurális hálók tanítását kezdetektől fogva felügyelt módon végezték. Ilyenkor minden $\mathbf{x}_m$ tanítómintához tartozik egy elvárt $\mathbf{y}_m$ kimenet, amely egy költségfüggvény segítségével összehasonlítható a kapott $\hat{\mathbf{y}}_m$ kimenettel. Ezt követően a hálóban szereplő sejtek közötti súlyokat és a sejtek bias termjeit azzal arányos mértékben szokás módosítani, amilyen mértékben ezen paraméterek a hibát befolyásolják. Ha pl. egy w_{ij} súly apró növelése jobban csökkenti a kimeneten mért hibát, mint egy másik súly ugyanilyen mértékű növelése, akkor inkább a w_{ij} súly erősségét célszerű növelni. Formálisan ezt úgy fejezzük ki, hogy a paramétereket a kimeneti hiba paraméterek szerinti gradiensével ellentétes irányba módosítjuk.

Az alkalmazott költségfüggvény alakja gyakran összefügg a kimeneti aktivációs függvény alakjával; eltérő aktivációs függvények esetén más és más költségfüggvényt célszerű használni. További szempont a neurális háló regularizációjának kérdése: ha az adatok mennyiségének megfelelő bonyolultságú modellt szeretnénk használni és ezt a bonyolultságot a súlyok nagyságának korlátozásával szeretnénk elérni, ez további additív tagokat eredményezhet a költségfüggvényben (minél nagyobbak a súlyok, annál nagyobb a megoldás költsége). Összességében a költségfüggvény általában analitikusan nem deriválható (vagy praktikusan nem deriválható mert túl sok paramétere van), ennek megfelelően a függvény globális minimuma (derivált $= 0$) nem határozható meg; továbbá legtöbbször a költségfüggvény nem lesz konvex tulajdonságú sem (ez azt jelenti, hogy ha bármely két pontját egyenessel összekötjük, nem lesz minden esetben igaz, hogy az összekötő egyenes minden pontja a függvényen vagy a fölött helyezkedik el; magyarán a függvény felületén 'hullámok' találhatóak és nem sima). Ez a két tény együttessen azt eredményezi, hogy a gradiens minimalizálásakor általában apránkénti csökkentéssel érdemes a megoldás felé haladni, ugyanakkor ügyelni kell arra is, hogy a keresés közben ne 'ragadjunk benne' az előbb említett hullámok által képzett lokális minimumokban. Az erre irányuló iteratív folyamatot **sztochasztikus gradiens-csökkentésnek** szokás nevezni, amely matematikailag a *nem-konvex optimalizálás* témaköréhez tartozik (ezzel a névvel jelezzük, hogy a költségfüggvény általában nem konvex).





$$u1 = k1\, m_1 w11 + k2\, m_2\, w12 + (b_1 / 3\ x1) + (b_2 / 3\ x1)$$

$$u2 = k1\, m_1 w21 + k2\, m_2\, w22 + (b_1 / 3\ x2) + (b_2 / 3\ x2)$$

$$u3 = k1\, m_1 w31 + k2\, m_2\, w32 + (b_1 / 3\ x3) + (b_2 / 3\ x3)$$

4.4. ábra. Ha az összes rejtett neuron aktivációs függvénye lineáris, nincs hozzáadott értékük

4.5.1. Aktivációs függvények deriváltja

Ebben az alfejezetben levezetjük néhány jellegzetes aktivációs függvény deriváltját.

Logisztikus függvény deriváltja.

$$\begin{aligned}\frac{\partial}{\partial x} sgm(x) = \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} = \frac{\partial}{\partial x}(1+e^{-x})^{-1} = -(1+e^{-x})^{-2}\cdot -1\cdot e^{-x} = \\ = \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}} = sgm(x)\cdot\frac{1+e^{-x}-1}{1+e^{-x}} = sgm(x)\,[1-sgm(x)]\end{aligned} \tag{4.4}$$

Softmax függvény deriváltja. A softmax függvény kétféle deriváltjára is kíváncsiak lehetünk: amikor azon sejt preaktivációja szerint deriválunk, amelyhez tartozó kimeneti softmax komponenst

deriváljuk, illetve amikor egy másik (szintén kimeneti) sejt preaktivációja szerint deriválunk (ez utóbbi is kihatással van a derivált függvényre, hiszen a normalizálásba beleszól!). A két esetet mutatja be az alábbi két levezetés:

$$\begin{aligned}\frac{\partial}{\partial a_k} softmax(\mathbf{a})_k &= \frac{\partial}{\partial a_k}\frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}} = \\ &= \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}} - \frac{e^{a_k}}{\left(\sum_{k'=1}^{K} e^{a_{k'}}\right)^2}\cdot e^{a_k} = \\ &= softmax(\mathbf{a})_k - softmax(\mathbf{a})_k^2 = softmax(\mathbf{a})_k \cdot [1 - softmax(\mathbf{a})_k]\end{aligned} \tag{4.5}$$

ahol felhasználtuk, hogy:

$$\left(\frac{f}{g}\right)' = \left(f \cdot \frac{1}{g}\right)' = f' \cdot \frac{1}{g} + f \cdot -\frac{1}{g^2}\cdot g' = \frac{f'}{g} - \frac{f}{g^2}\cdot g' \tag{4.6}$$

ugyanígy ha más preaktiváció szerint deriválunk:

$$\begin{aligned}\frac{\partial}{\partial a_u} softmax(\mathbf{a})_k &= \frac{\partial}{\partial a_u}\frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}} = \\ &= 0 - \frac{e^{a_k}}{\left(\sum_{k'=1}^{K} e^{a_{k'}}\right)^2}\cdot e^{a_u} = \\ &= -softmax(\mathbf{a})_k \cdot softmax(\mathbf{a})_u\end{aligned} \tag{4.7}$$

4.5.2. Költségfüggvények típusai

Elöljáróban megjegyezzük, hogy ha a kimeneti sejt aktivációs függvénye szigmoid, és nagyon extrém (1-hez vagy 0-ához közeli) y_i kimenetet kapunk, a hiba deriváltja is elenyészően kicsi lesz, hiszen ekkor a szigmoid függénynek értékkészletének a „közel 0 meredekségű" részénél tartózkodunk (a derivált pedig $y_i(1-y_i)$, mely két tag közül az egyik biztosan 0-ához közeli). Ezen a problémán semmilyen költségfüggvény nem tud segíteni, így a kimeneti rétegben a szigmoid függvényt mint költségfüggvényt érdemes kerülni.

Regresszió esetén a kimeneten célszerűen valamilyen lineáris aktivációs függvényt érdemes használni, hiszen ilyenkor a háló kimenete valós számként értelmezendő.

Ezzel szemben ha kategorizációs feladaton dolgozunk, a hálót célszerű úgy felépíteni, hogy a kimeneti réteg sejtjei egymást kizáró kategóriát reprezentálnak, azzal a megfontolással, hogy a kimeneteket valószínűségként értelmezve megállapítható az adott bemenet kategóriája, illetve az ezen megállapításhoz kapcsolódó bizonytalanságunk is. Ilyenkor az összes kimenet összege mindenkor 1 kell, hogy legyen; továbbá az egyik kimenetet módosítani csak a többivel összefüggésben lehet. Mindezen finomságokat egy szigmoid aktivációs függvényeket használó kimeneti réteg nem képes reprezentálni; a softmax aktivációs függvény viszont igen. Ezért kategorikus kimenetek esetén érdemes a softmax aktivációt használni.

Az eddigi fejtegetést mindenesetre az motiválja, hogy különböző feladatokhoz eltérő kimeneti aktivációs függvény illik, és az értelmezés (valamint számítási kényelem) az ideális költségfüggvény alakját is befolyásolja.

Regresszió kontextusában a **négyzetes hiba** kézenfekvő költségfüggvény:

$$E_{sqerror} = (y - \hat{y})^2 \tag{4.8}$$

, ahol y az elvárt, $\hat{y}$ pedig a kapott kimenet.

Amennyiben azonban kategorizációs feladatról van szó, és a fentieknek megfelelően softmax aktivációs réteget használunk, akkor a **keresztentrópia** alkalmasabb költségfüggvény. A keresztentrópia alakja egyetlen adatpontra a következő:

$$E_{crossentropy} = -\sum_i y_i \log \hat{y}_i \tag{4.9}$$

, ahol y_i a kimeneti réteg i. sejtjére vonatkozik. A hiba értelmezése a következő: ha az i. sejt elvárt y_i kimenete (vagyis az i. kategória valószínűsége) közel van 1-hez, akkor az összegben a $-log\hat{y}_i$ tag jelentős súllyal fog szerepelni, továbbá $\hat{y}_i$ értéke minél kisebb 1-nél (és minél inkább a 0-ához közelít), a negatív logaritmusa annál nagyobb lesz. Magyarán a keresztentrópia a kategóriák elvárt valószínűsége alapján számított súlyozott összege olyan tagoknak, melyek értéke minél kisebb, költségük annál nagyobb.

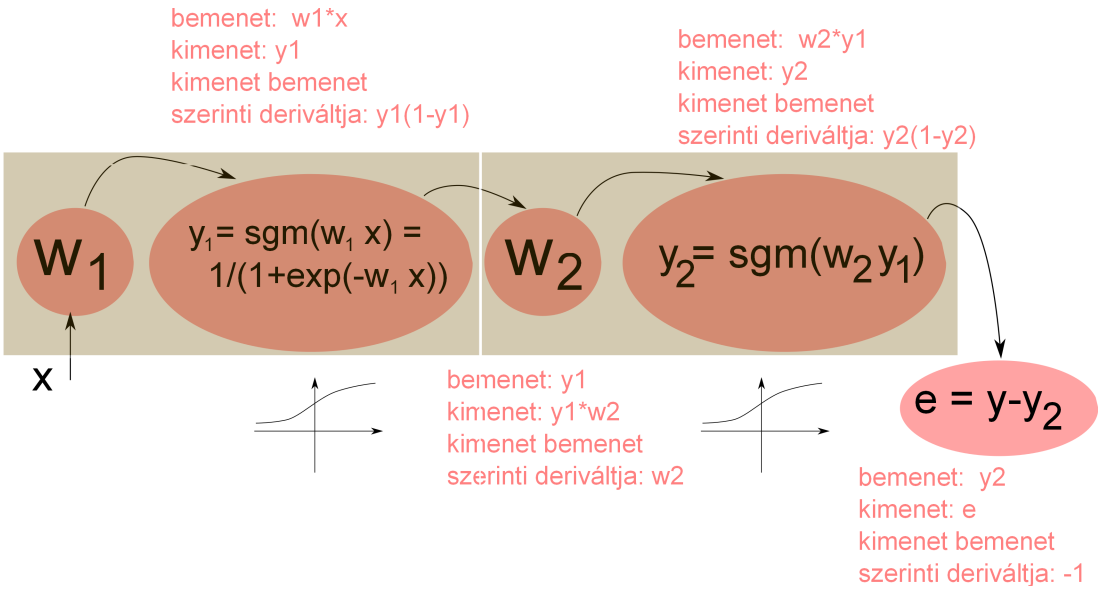
A keresztentrópia költségfüggvény preaktiváció szerinti deriváltja – softmax aktivációs réteget feltételezve – az alábbiak szerint alakul:

$$\begin{aligned}
\frac{\partial E_{crossentropy}}{\partial \mathbf{a}_k} &= \sum_i \frac{\partial E_{crossentropy}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \mathbf{a}_k} = \\
&= \frac{-y_k}{\hat{y}_k} \hat{y}_k (1 - \hat{y}_k) + \sum_{i=1, i \neq k}^{I} \frac{-y_i}{\hat{y}_i} - \hat{y}_i \hat{y}_k = \\
&= -y_k + y_k \hat{y}_k + \hat{y}_k \underbrace{\sum_{i=1, i \neq k}^{I} y_i}_{1 - y_k} = \\
&= \hat{y}_k - y_k
\end{aligned} \tag{4.10}$$

ahol az első tag annak a speciális esetnek felel meg, amikor $i = k$. Látható, hogy az eredményül kapott kifejezés könnyen számítható, ezért is hasznos a softmax aktivációt és a keresztentrópiát, mint költségfüggvényt együtt alkalmazni.

4.5.3. Backpropagation algoritmus

A backpropagation algoritmus bemutatásához elsőként tekintsünk egy leegyszerűsített neurális hálót, amely két sejt soros láncolatából áll (ld. 4.5. ábra). Az ábrán két sejt működését látjuk kiterítve:



4.5. ábra. Minimális példa backpropagation módszerének bemutatására.

mindegyik egy skalár bemenetet vár, amelyet egyetlen súllyal összeszoroz, majd az így kapott preaktivációt a logisztikus függvényen keresztül transzformálva juttatja el a kimenetéhez. Végül a hibát az elvárt y és kapott y_2 érték különbségeként fejezzük ki.

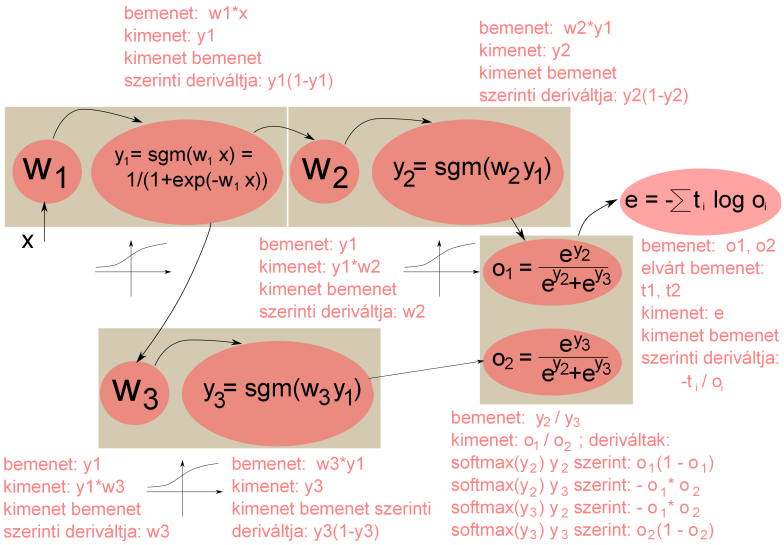
Egyenletekkel kifejezve a kimenet és a hiba az alábbi módon számítható:

$$\begin{aligned} y1 &= \frac{1}{1+exp(-w_1 x)} = sgm(w_1 x) \\ y2 &= \frac{1}{1+exp(-w_2 y_1)} = sgm(w_2 y_1) \\ e &= y - y_2 \end{aligned} \tag{4.11}$$

ahonnan a keresett parciális deriváltak a láncszabályon keresztül:

$$\begin{aligned} \frac{\partial}{\partial w_2} e &= \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial w_2} = -y_2(1-y_2)y_1 \\ \frac{\partial}{\partial w_1} e &= \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial y_1} \cdot \frac{\partial y_1}{\partial w_1} = -y_2(1-y_2)w_2 y_1(1-y_1)x \end{aligned} \tag{4.12}$$

A két eredmény alakjából és összehasonlításából látszik egyrészt, hogy a w_1 súly szerinti deriválásnál több olyan tag szerepel a szorzatban, amely már a w_1 súly szerinti deriválásnál is szerepelt. Ez a már meglevő eredmény tulajdonképpen a háló kimenethez közelebbi részei felől visszaterjesztett hibagradiens, amit változtatás nélkül tovább szorzhatunk a láncszabály szerint. Megfigyelhető továbbá, hogy mindig amikor egy adott paraméter szerinti deriváltra vagyunk kíváncsiak, a kimenet felől érkező gradienst össze kell szoroznunk a paraméter bemenetével:



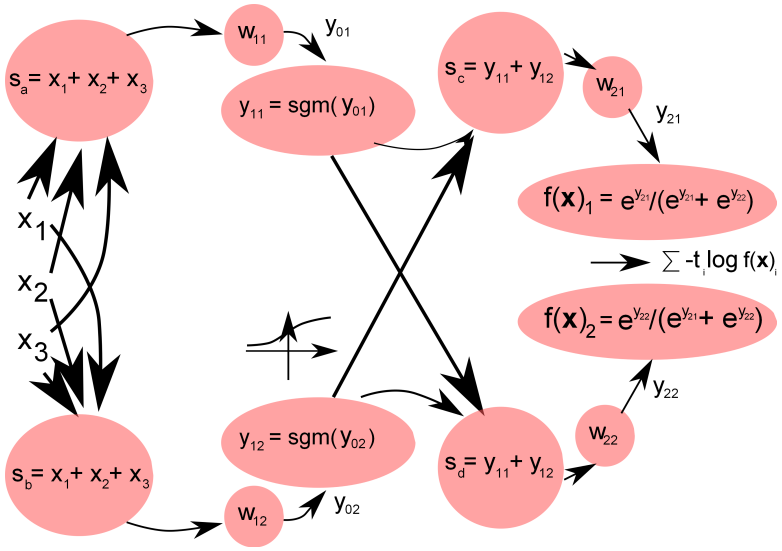
4.6. ábra. Kicsit összetettebb példa backpropagation módszerének bemutatására.

$$\begin{aligned}\frac{\partial}{\partial w_2}e &= backrop_1 \cdot \mathbf{bemenet} = -y_2(1-y_2)y_1 \\ \frac{\partial}{\partial w_1}e &= backrop_1 \cdot backprop_2 \cdot \mathbf{bemenet} = -y_2(1-y_2)w_2y_1(1-y_1)\mathbf{x}\end{aligned} \tag{4.13}$$

Ez az elv jól kiolvasható a 4.5 ábrából is. Látható, hogy minden számítási egységnél csak az kell, hogy érdekeljen bennünket, hogy az egység kimenetének mi a bemenete szerinti deriváltja. Ezt a deriváltat kell a kimenethez közelebbi részekről érkező backpropagation taggal összeszorozni. Amennyiben egy paramétert útközben módosítanánk is, akkor a szorzat továbbterjesztése mellett a visszafele beérkező backpropagation tagot meg kell szorozni a paraméter „bemenetével" is, melynek eredményeképpen megkapjuk a kimenet (hiba) paraméter szerinti deriváltját.

Bonyolultabb hálóknál a backpropagation elve ugyanez, azzal a különbséggel hogy ha egy adott paraméter a kimeneti hibát több útvonalon is befolyásolja, akkor a különböző útvonalakon érkező visszaterjesztett gradienseket összegezni kell, majd úgy a bemenettel szorozni illetve az összeget az aktuális lokális gradienssel összeszorozva továbbítani a háló bemeneti rétegei felé.

Például a 4.6. ábrán látható hálóban a kimeneti keresztentrópia függvény értékét a w_1 súly négy (fenti és lenti, azon belül pedig o_1-en és o_2-n keresztüli) útvonalon is befolyásolja. Ezért a hiba paraméter szerinti deriváltja a láncsszabály alapján a következőképpen néz ki:



4.7. ábra. Még összetettebb példa backpropagation módszerének bemutatására.

$$
\begin{aligned}
\frac{\partial e}{\partial w_1} &= \left[\frac{\partial e}{\partial o_1}\cdot\frac{\partial o_1}{\partial y_2}\cdot\frac{\partial y_2}{\partial y_1} + \frac{\partial e}{\partial o_2}\cdot\frac{\partial o_2}{\partial y_3}\cdot\frac{\partial y_3}{\partial y_1} + \frac{\partial e}{\partial o_1}\cdot\frac{\partial o_1}{\partial y_3}\cdot\frac{\partial y_3}{\partial y_1} + \frac{\partial e}{\partial o_2}\cdot\frac{\partial o_2}{\partial y_2}\cdot\frac{\partial y_2}{\partial y_1}\right]\cdot\frac{\partial y_1}{\partial w_1} = \\
&= \left[\frac{-t_1}{o_1}\cdot(o_1(1-o_1)\cdot y_2(1-y_2)\cdot w_2 - o_1\cdot o_2\cdot y_3(1-y_3)\cdot w_3)\right]\cdot\frac{\partial y_1}{\partial w_1} + \\
&\quad \left[\frac{-t_2}{o_2}\cdot(o_2(1-o_2)\cdot y_3(1-y_3)\cdot w_3 - o_1\cdot o_2\cdot y_2(1-y_2)\cdot w_2)\right]\cdot\frac{\partial y_1}{\partial w_1} = \\
&= \left[w_2\cdot y_2(1-y_2)\left(-t_1 + \underbrace{t_1o_1 + t_2o_1}_{o_1}\right)\right]\cdot\frac{\partial y_1}{\partial w_1} + \\
&= \left[w_3\cdot y_3(1-y_3)\left(-t_2 + \underbrace{t_2o_2 + t_1o_2}_{o_2}\right)\right]\cdot\frac{\partial y_1}{\partial w_1} = \\
&= \begin{bmatrix} w_2 \\ w_3 \end{bmatrix}^T \begin{bmatrix} (o_1-t_1)y_2(1-y_2) \\ (o_2-t_2)y_3(1-y_3) \end{bmatrix} y_1(1-y_1)\cdot x
\end{aligned}
\tag{4.14}
$$

Egy még összetettebb (nemkonvencionális struktúrájú) példa a 4.7. ábrán látható. Ha a hálóban a kimeneti negatív log-probabilitás (keresztentrópia) hiábját deriváljuk a w_{11} paraméter szerint, akkor az alábbi számításokat kell elvégeznünk:

$$
\begin{aligned}
\frac{\partial e}{\partial w_{11}} &= \left[\frac{\partial e}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial y_{11}} + \frac{\partial e}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial y_{11}}\right] \cdot \frac{\partial y_{11}}{\partial w_{11}} = \\
&= \left[\sum \frac{-t_i}{f(\mathbf{x})_i} \cdot \frac{\partial f(\mathbf{x})_i}{\partial y_{21}} \cdot w_{21} + \sum \frac{-t_i}{f(\mathbf{x})_i} \cdot \frac{\partial f(\mathbf{x})_i}{\partial y_{22}} \cdot w_{22}\right] \cdot y_{11}(1-y_{11})s_a = \\
&= \left[w_{21}\left(\frac{-t_1}{f(\mathbf{x})_1} \cdot f(\mathbf{x})_1(1-f(\mathbf{x})_1) + \frac{-t_2}{f(\mathbf{x})_2} \cdot -f(\mathbf{x})_1 f(\mathbf{x})_2\right) + w_{22}\left(\ldots\right)\right] \cdot y_{11}(1-y_{11})s_a = \\
&= \left[w_{21}\left(-t_1 + \underbrace{(t_1+t_2)}_{1} \cdot f(\mathbf{x})_1\right) + w_{22}\left(\underbrace{(t_1+t_2)}_{1} \cdot f(\mathbf{x})_2 - t_2\right)\right] \cdot y_{11}(1-y_{11})s_a = \\
&= \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix}^T (f(\mathbf{x}) - \mathbf{t}) \cdot y_{11}(1-y_{11})s_a
\end{aligned}
\tag{4.15}
$$

4.5.4. A backpropagation algoritmus vezérlése

Ahhoz, hogy mindez használható legyen, szükségünk van néhány további dontés meghozatalára, nevezetesen hogy:

- milyen menetrend szerint optimalizáljunk, vagyis milyen sorrendben, hányszor tekintsük a tanítómintákat és hogyan frissítsük a súlyokat (tanítómintánként? több tanítóminta felett átlagolva? az összes felett átlagolva?)
- hogyan kerüljük el a túltanulást és általánosítsunk jól?

Ezen kérdések megválaszolásakor több lehetőségünk van, melyeket a következő alfejezetekben mutatunk be.

4.6. Sztochasztikus gradiens-csökkentés optimalizációja

Arra a kérdésre, hogy milyen menetrend szerint frissítsük a súlyokat, többféle válaszunk lehet:

- **online** esetben tanítómintaként frissítjük a súlyokat a deriváltaknak megfelelően
- **full batch** esetben az összes tanítómintára együttesen frissítjük a súlyokat a deriváltak átlagának megfelelően
- **minibatch** esetben a teljes tanítóhalmazt partícionáljuk, sorra vesszük a partíciókat és a bennük szereplő tanítómintákon kiszámított deriváltakat átlagolva partíciónként frissítjük a súlyokat

Ezek közül a megoldások közül az online módszer általában elég pazarló, a full batch viszont azért lehet szuboptimális, mert ha az adathalmaz redundanciát tartalmaz, többször számoljuk ugyanazt és

az átlag nem változik. A két megoldás övözéseként általában a minibatch módszer javasolt. Ilyenkor még kapóra is jön, hogy több különböző bemeneti mintára kell a gradienst kiszámítani, ugyanis ha a háló közben nem változik, mindez hatékony mátrix-műveletekkel GPU-n gyorsan megoldható.

További szempont a kiszámított súlyok skálázásának kérdése, ugyanis általában a deriváltak szerint kiszámított frissítési értéknek csak egy töredékét adjuk hozzá a súlyokhoz (elkerülendő a zajos minták extrém hatásait). A skálázást illetően szintén többféle lehetőség adódik, melyek egyike-másika még kombinálható is:

- **fix, globális tanulási ráta** használata
- **adaptív, globális tanulási ráta** használata
- **súlyonként különböző tanulási ráta** használata. Főleg az *eltűnő gradiensek (vanishing gradients)* jelensége – miszerint a sorozatosan visszaterjesztett gradiens a bemenethez közelítve egyre kisebb, majd a végén elenyésző lesz – motiválhatja ezt a választást. Egy lehetséges módszer, hogy kezdetben minden paraméter gradienséhez tartozó lokális szorzó 1, és amikor a következő gradiens előjele megegyezik a mindenkori aktuálissal, az erősséghez hozzáadunk egy δ konstanst; amikor viszont nem egyezik meg akkor szorozzuk $1-\delta$ konstanssal. Az így kapott lokális tanulási rátákat szorozzuk a globális tanulási rátával

Fentieken kívül a legmeredekebb (gradiens) irány követése helyett mehetünk valamilyen módosított, a végső cél szempontjából optimálisabbnak ígérkező irányba is. Képzeljük el, hogy egy hosszú szzakadékban' vagyunk a hibafelületen, a szakadék mélyén pedig egy szinte vízszintes lejtő visz a minimum pont felé. Ilyenkor a szakadék falainak irányába nagy a gradiens, a szinte vízszintes lejtő irányába viszont kicsi – pont fordítva, mint ahogy ideális lenne, hiszen inkább a szakadék alja felé mennénk tovább, mint hogy a szakadék falai mentén cikkcakkozzunk! Ezért ha a paraméter-vektort a gradiensek méretének megfelelően frissítenénk, bizonyos esetekben csak nagyon lassan konvergálnánk, célszerűbb lenne azok reciprokával számolni.

Ezzel együtt ez csak bizonyos esetekben van így, nem mondható, hogy mindig így lenne, hiszen előfordulhat hogy éppen egy nagyon meredek út visz a célhoz. A példánál maradva, a probléma inkább a szakadék falaival van, hogy az egymást követő frissítéseknél a derivált falak irányába mutató komponense lényegében előjelet vált. A probléma megoldására többek között az alábbi lehetőségek kínálkoznak:

- **momentum módszer** alkalmazása: a súlyvektor irányát nem a gradienssel, hanem annak egy mozgó átlagával módosítjuk. Képzeljünk el egy labdát, ami egy bonyolult felület minimuma felé gurul. Kezdetben a labda a tényleges gradiens mentén gurul, azonban ahogy nő a sebessége, a momentuma révén korlátozottan 'emlékszik' a korábbi irányokra. A frissítés képlete tehát konceptuálisan egy sebességvektor karbantartásával az alábbi módon alakul:

$$\begin{aligned}\Delta\mathbf{w}(t) &= \alpha\Delta\mathbf{w}(t-1) - \epsilon\frac{\partial E}{\partial \mathbf{w}}(t) = \\ &= \alpha\left(\alpha\Delta\mathbf{w}(t-2) - \epsilon\frac{\partial E}{\partial \mathbf{w}}(t-1)\right) - \epsilon\frac{\partial E}{\partial \mathbf{w}}(t) \\ \Delta\mathbf{w}(0) &= -\epsilon\frac{\partial E}{\partial \mathbf{w}}(0)\end{aligned} \tag{4.16}$$

A momentum módszernek létezik egy Nesterov által 1983-ban javasolt módosított változata is. A **Nesterov módszer** használatakor nem az történik, hogy először kiszámítjuk a gradiens irányát, majd egy nagyot lépünk abba az irányba (ahogy a momentum módszerben tesszük), hanem először a korábbi gradiens irányába teszünk egy kis lépést, majd az új állapotban számítjuk a következő gradienst és lépünk el az általa mutatott irányba. Informálisan azt is mondhatjuk, hogy okosabb dolog egy hibát azután kijavítani, miután már elkövettük, mintha már előre javítani probálnánk, amikor a pontos mértékét és irányát még nem is ismerjük. Formálisan pedig mindez annyit jelent, hogy a frissítés képlete az alábbiak szerint alakul:

$$\begin{aligned} \Delta \mathbf{w}(T) &= -\epsilon \frac{\partial E}{\partial \mathbf{w}}(T) \\ \Delta \mathbf{w}(t) &= -(1+\alpha)\epsilon \frac{\partial E}{\partial \mathbf{w}}(t) \text{if t ¡ T} \end{aligned} \tag{4.17}$$

. . . csak az utolsó frissítéskor nem számít a momentum-tag (α)

- **rmsprop módszer**: az a megfigyelés motiválja, hogy amikor sokféle méretű gradiens van a hálóban, nehéz megfelelő globális tanulási rátát választani; ezért célszerű a gradienseket az azok normájának mozgó átlagával normalizálni:

$$\Delta \mathbf{w}(t) = \frac{-\epsilon \frac{\partial E}{\partial \mathbf{w}(t)}}{0.9 \frac{\partial E}{\partial \mathbf{w}}^T (t-1) \frac{\partial E}{\partial \mathbf{w}}(t-1) + 0.1 \frac{\partial E}{\partial \mathbf{w}}^T (t) \frac{\partial E}{\partial \mathbf{w}}(t)} \tag{4.18}$$

4.7. Neurális hálók regularizációja

A modellek betanításához használt adatokat a be- és kimenetek közötti regularitások bemutatásához használjuk. Az adatok azonban kétféle hibát is tartalmazhatnak:

- A mérések lehetnek *zajosak* (ez általában a kisebbik gond)
- Az adatokban lehetnek *mintavételezési hibák* – pusztán attól, hogy az összes lehetséges adatnak egy szűk halmazát tekintjük, előfordulhat hogy olyan szabályszerűségek is megfigyelhetőek bennük amik általánosságban nem érvényesek.

A mintavételezési hibák komoly általánosításbeli romláshoz vezethetnek, hiszen a modell pusztán az adatok alapján nem tudja megmondani, mi a valódi és mi a mintavételezésből adódó szabályszerűség. A probléma kiküszöböléséhez neurális hálók esetében többféle javaslat született:

- **súlyok büntetése**: hátrányban részesítjük azokat a konfigurációkat, melyekben sok nagyértékű súly van. Többféle változat lehetséges: büntethetjük a súlyokat egyenként, vagy büntethetjük a teljes súlyvektor magas normáját. Ez utóbbi mellett sok érv szól: könnyebb egy hatékony

értéket választani, és nem töltünk felesleges időt az olyan súlyok alacsony értéken tartásával, amelyek egyébként sem nyomnak sokat a latban (ha például egy másik súly is mutat ugyanazon sejtbe, melynek értéke nagyságrendekkel nagyobb).

- **súlyok megosztása**: weight sharing esetben kikötjük, hogy bizonyos súlyok értéke ugyanaz kell, hogy legyen (ekkor úgy tanítjuk őket, hogy a frissítéshez kiszámított tagokat átlagoljuk)
- **korai megállás**: nem várjuk meg, amíg a háló a tanítóadatokra optimálisan teljesít. Ettől azt reméljük, hogy cserébe jó kompromisszumot érünk el az általánosításhoz.
- **modellek átlagolása**: több modellt tanítunk be és a kimenetüket átlagoljuk. Ha a modellek egymástól függetlenül hibáznak (a hibáik nem korrelálnak) akkor általánosságban csökkentjük a teljes modell varianciáját.
- **Bayes-alapú tanítás**: a bonyolultabb modelleket mindaddig nem tekintjük rosszabbnak a többinél, ameddig nincsen növekvő bizonyítékünk ellenük. Ezért fix architektúrát használunk de többféle súlyvektorral, melyek hatását egy előzetes hiedelem szerint súlyozottan átlagoljuk
- **dropout**: minden frissítési körben néhány random módon kiválasztott sejt aktivitását kinullázzuk, ezzel egyfajta kiszámíthatatlan zajt viszünk a modell viselkedésébe és a nem kinullázott sejtek működésükben nem támaszkodhatnak fixen a többi működésére (mindegyik sejt remélhetőleg valami független, 'hasznos' dolgot fog csinálni)
- **generatív előtanítás**: a generative pre-training módszere nem felügyelt módon hoz létre olyan köztes reprezentációkat, melyek több rétegen keresztül kerülnek legenerálásra, így magukban hordozzák az eredeti adathalmaz valamilyen desztillált tulajdonságait.

4.8. Példa: logisztikus regresszió visszavezetése neurális hálókra

Látszólag messziről indítunk. Kategorikus kimenetek esetén hasznos módszer a (büntetéses) logisztikus regresszió, amely tulajdonképpen nem is regressziós módszer, azonban nevében mégis szerepel a „regresszió" szó mert a lineáris regresszió „unokatestvérének" tekinthető.

4.8.1. Logisztikus regresszió két osztály esetén

Logisztikus regresszió esetén egy szigmoid-függvényt keresünk, amelyet valószínűség-eloszlásként értelmezhetünk egy lineáris bemenet felett. Például két kategória esetén:

$$p(y = 1|\mathbf{x}, \omega) = sgm(\omega^T \mathbf{x}) \tag{4.19}$$

A feladat ilyenkor olyan ω súlyokat találni, amelyek által generált kimenet jól illeszthető a problémához.

Fontos megérteni, hogy mi motiválja ezt a modellt. Osztályozásnál kézenfekvő ötlet lehetne a lineáris osztályozás is:

$$P(y = 1|\mathbf{x}, \omega) = \omega^T \mathbf{x} \tag{4.20}$$

Ezzel két probléma van: egyrészt a kimenet csak 0 és 1 közé eshet, de a lineáris függvény értékkészlete korlát nélküli; másrészt pedig legtöbb valós példánál bizonyos szint fölött a bemenet azonos mértékű módosítása a kimeneti valószínűségen relatíve kisebb hatást ér el.

Ezért a következő ötlet, hogy legyen a valószínűség logaritmusa lineáris. Ekkor

$$P(y = 1|\mathbf{x}, \omega) = e^{\omega^T \mathbf{x}} \tag{4.21}$$

Csakhogy az exponenciális függvény kimenete felfelé nem korlátos, így a probléma hasonló (csak az alsó korlátot értük el, hogy 0 legyen). Ezért az ötletet kicsit módosítjuk: ami két kategória esetén lineáris lesz, az a *log odds*-ként is ismert hányados: $\log P(y|\mathbf{x}, \omega)/(1 - P(y|\mathbf{x}, \omega))$. Az odds bármilyen 0 és ∞ közötti értéket felvehet, ezért nem probléma, ha ezt tekintjük lineáris függvénynek.

Ezek után csak meg kell néznünk, mit jelent ez maga a valószínűség szempontjából:

$$\begin{aligned} \log \frac{P(y|\mathbf{x}, \omega)}{1 - P(y|\mathbf{x}, \omega)} &= \omega^T \mathbf{x} \\ P(y|\mathbf{x}, \omega) &= (1 - P(y|\mathbf{x}, \omega)) e^{\omega^T \mathbf{x}} \\ P(y|\mathbf{x}, \omega)(1 + e^{\omega^T \mathbf{x}}) &= e^{\omega^T \mathbf{x}} \\ P(y|\mathbf{x}, \omega) &= \frac{1}{1 + e^{-\omega^T \mathbf{x}}} = sgm(\omega^T \mathbf{x}) \end{aligned} \tag{4.22}$$

Ahogy sejthettük, éppen a logisztikus függvényt kaptuk vissza. Ez a függvény értékkészletét tekintve is alkalmas, és teljesíti a nemlinearitásra vonatkozóan megfogalmazott igényünket is. Ugyanakkor a log odds fogalmán keresztül azt is láthattuk, hogy ez a modell nagyon jól interpretálható.

Tegyük fel pl., hogy $\mathbf{x}$ kétdimenziós, jelentése pedig hogy hány cigarettát szív el valaki egy nap, és hány percet fut egy nap; tegyük fel hogy ez alapján szeretnénk azt a valószínűséget megmondani, hogy az illetőnek tüdőrákos megbetegedése lesz. Ha a kapott ω értéke $(1.3, -1.1)$, akkor minden újabb cigaretta elszívásával a tüdőrákhoz társított odds $e^{1.3}$-szorosára növekszik. Ugyanis:

$$\begin{aligned} \log \frac{P(y)}{1 - P(y)} &= \omega^T \mathbf{x} \\ \frac{P(y)}{1 - P(y)} &= e^{1.3x_1 - 1.1x_2} \\ \frac{P(y)}{1 - P(y)} &= e^{1.3x_1} e^{-1.1x_2} \end{aligned} \tag{4.23}$$

vagyis ha például x_1 értéke 1-ről 2-re nő, akkor éppen egy újabb $e^{1.3}$ kerül bele a képletbe, mint szorzótényező.

4.8.2. Logisztikus regresszió több osztály esetén

Fentiek általánosíthatóak több kategóriára is. Ehhez elég ha K különböző osztály esetén $K - 1$ különböző logisztikus regresszort állítunk fel:

$$
\begin{aligned}
\log \frac{P(y=1|\mathbf{x},\omega)}{1-P(y=K|\mathbf{x},\omega)} &= {\omega_\mathbf{1}}^T\mathbf{x} \\
\log \frac{P(y=2|\mathbf{x},\omega)}{1-P(y=K|\mathbf{x},\omega)} &= {\omega_\mathbf{2}}^T\mathbf{x} \\
&\dots \\
\log \frac{P(y=K-1|\mathbf{x},\omega)}{1-P(y=K|\mathbf{x},\omega)} &= {\omega_{\mathbf{K-1}}}^T\mathbf{x}
\end{aligned}
\tag{4.24}
$$

Az egyenletek átrendezésével azt kapjuk, hogy:

$$
\begin{aligned}
P(y=1|\mathbf{x},\omega) &= \left[1-P(y=K|\mathbf{x},\omega)\right] e^{{\omega_\mathbf{1}}^T\mathbf{x}} \propto e^{{\omega_\mathbf{1}}^T\mathbf{x}} \\
P(y=2|\mathbf{x},\omega) &= \left[(1-P(y=K|\mathbf{x},\omega)\right] e^{{\omega_\mathbf{2}}^T\mathbf{x}} \propto e^{{\omega_\mathbf{2}}^T\mathbf{x}} \\
&\dots \\
P(y=K-1|\mathbf{x},\omega) &= \left[(1-P(y=K|\mathbf{x},\omega)\right] e^{{\omega_{\mathbf{K-1}}}^T\mathbf{x}} \propto e^{{\omega_{\mathbf{K-1}}}^T\mathbf{x}}
\end{aligned}
\tag{4.25}
$$

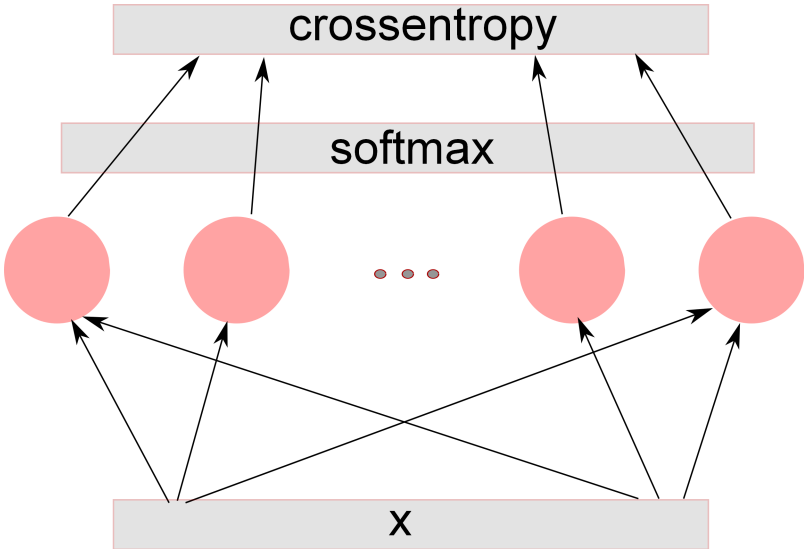
Mivel az összes valószínűség összege 1 kell, hogy legyen, így adódik, hogy:

$$
\begin{aligned}
P(y=1|\mathbf{x},\omega) &= \frac{e^{{\omega_\mathbf{1}}^T\mathbf{x}}}{\alpha+\sum_{k=1}^{K-1} e^{{\omega_\mathbf{k}}^T\mathbf{x}}} \\
P(y=2|\mathbf{x},\omega) &= \frac{e^{{\omega_\mathbf{2}}^T\mathbf{x}}}{\alpha+\sum_{k=1}^{K-1} e^{{\omega_\mathbf{k}}^T\mathbf{x}}} \\
&\dots \\
P(y=K-1|\mathbf{x},\omega) &= \frac{e^{{\omega_{\mathbf{K-1}}}^T\mathbf{x}}}{\alpha+\sum_{k=1}^{K-1} e^{{\omega_\mathbf{k}}^T\mathbf{x}}} \\
P(y=K|\mathbf{x},\omega) &= \frac{\alpha}{\alpha+\sum_{k=1}^{K-1} e^{{\omega_\mathbf{k}}^T\mathbf{x}}}
\end{aligned}
\tag{4.26}
$$

, ahol α lényegében bármilyen pozitív szám lehet – gyakran 1-re szokás állítani. Konkrét értékének azonban nincs jelentősége, hiszen értékének megfelelően a tanulás eredményeképpen kapott ω vektor is skálázódhat.

4.8.3. Logisztikus regresszió megoldása

A logisztikus regresszió megoldását egy olyan ω vektor formájában keressük, amely az adatokhoz jól illeszkedő sigmoid-függvényt eredményez.



4.8. ábra. Logisztikus regresszió neurális háló megfeleltetése.

A feladat megoldására első ötletünk lehet, hogy az ún. (log)-likelihood értéket maximalizáljuk! Ez azt jelenti, hogy a sigmoid-függvényt úgy állítjuk be, hogy az adatok 'összesített' valószínűsége minél magasabb legyen.

Mit értünk 'összesített valószínűség' alatt? Ha van M darab megfigyelésünk, és ezek függetlenek egymástól, akkor a valószínűségeiket összeszorozhatjuk (így jön ki a likelihood). Vagyis, ha bináris kimenet esetén $p(\mathbf{x}^{(m)})$ a valószínűsége, hogy az m. bemenetre adott kimenet 1, akkor a likelihood:

$$\mathcal{L}(\omega) = \prod_{m=1}^{M} p(\mathbf{x}^{(m)})^{y^{(m)}} (1 - p(\mathbf{x}^{(m)}))^{(1-y^{(m)})} \tag{4.27}$$

(attól függően, hogy az m. kimenet 0 vagy 1, az egyik valószínűséget a 0.-onra emeljük). Ennek a kifejezésnek az egyszerűség kedvéért vesszük a logaritmusát (annak érdekében, hogy a szorzatok összegekként legyenek kifejezhetőek), majd kicsit átalakítjuk. Felhasználva azt is, hogy $y^{(m)}$ csak 0 vagy 1 lehet:

$$\begin{aligned}
&\log \mathcal{L}(\omega) = \log \prod_{m=1}^{M} p(\mathbf{x}^{(m)})^{y^{(m)}} (1 - p(\mathbf{x}^{(m)}))^{(1-y^{(m)})} = \\
&= \sum_{m=1}^{M} y^{(m)} \log p(\mathbf{x}^{(m)}) + (1 - y^{(m)}) \log(1 - p(\mathbf{x}^{(m)})) \\
&= \sum_{m=1}^{M} y^{(m)} \log(1 + e^{-\omega^T \mathbf{x}^{(m)}})^{-1} + (1 - y^{(m)}) \log(1 + e^{\omega^T \mathbf{x}^{(m)}})^{-1} \\
&= \sum_{m=1}^{M} \log(1 + e^{(1-2y^{(m)})\omega^T \mathbf{x}^{(m)}})^{-1}
\end{aligned} \tag{4.28}$$

Az eredményül kapott kifejezés egy ún. „transzcendentális függvény", mely deriváltjának nincs zárt formában megoldása, hiába is akarnánk hogy deriváljuk és a deriváltat 0-val tegyük egyenlővé. Ilyenkor ugyanis megtalálható lenne a képletben ω és e^{ω}, végül pedig nem lehetne kifejezni ω-t. Innen látszik, hogy a korábbiaknak megfelelően valamilyen iteratív-approximációs módszerre van szükség a logisztikus regresszió megoldásához, ez azonban a fejezetben bemutatott sztochasztikus gradiens módszerek ismeretében nem jelent problémát.

Ugyanakkor a fenti levezetésből azt is láthattuk, hogy a logisztikus regressziót általánosan meg tudjuk oldani, ha a korábban ismertetett keresztentrópia-függvényt próbáljuk minimalizálni. Több változó esetén ugyanis a log likelihood:

$$\begin{aligned} \log \mathcal{L}(\omega) &= \log \prod_{m=1}^{M} \prod_{k=1}^{K} p(y^{(m)} = k)^{I\{y^{(m)}=k\}} = \\ &= \sum_{m=1}^{M} \sum_{k=1}^{K} I\{y^{(m)} = k\} log \left[p(y^{(m)} = k) \right] \end{aligned} \tag{4.29}$$

Az egész felállás pedig interpretálható úgy, mint egy egyrétegű háló, melynek kimenetén az egyes kategóriák valószínűsége szerepel (tehát softmax kimenetet használunk), és amely tanításához a keresztentrópia függvényt használjuk, mint költségfüggvényt (ld. 4.8. ábra):

$$\log \mathcal{L}(\omega) = \sum_{m=1}^{M} \sum_{k=1}^{K} y_k^{(m)} log \left(\hat{y}_k^{(m)} \right) \tag{4.30}$$

Ezzel megmutattuk, hogy a logisztikus regresszió szó szerint egyenértékű egy speciális (nagyon egyszerű) feed-forward neurális háló tanításával.

5. fejezet

Grafikus modellek

Forrásmegjelölés

A fejezetben számos magyarázat és példa az alábbi angol nyelvű referenciákból származik: [4].

Tanuláskor érdemes probabilisztikus megközelítéseket használni, hiszen több okból is kijelenthető, hogy a tévedések valószínűsége sohasem 0:

- a tanulást követően mindig általánosítanunk kell, ami inheresen nem egzakt feladat;
- variancia típusú hibák: elképzelhető hogy rossz tanítóhalmazzal tanultunk (ritka példányokat nélkülözte az adathalmaz, vagy a minták nem voltak reprezentatívak vagy túl zajosak voltak)
- bias típusú hibák: elképzelhető hogy a modell, amit választottunk, a feladatra önmagában nem alkalmas
- végül lehetséges, hogy a világ maga sem determinisztikus, ezért maga a problémafelvetés is olyan, hogy nem lehet egzakt szabályosságokat megállapítani

A grafikus modellek kapcsolatot teremtenek a valószínűségelmélet és a gráfelmélet között. Előnyük, hogy vizuálisan is szemléletessé teszik a probabilisztikus modellek összefüggéseit, ráadásul nemcsak az összefüggések reprezentálására, hanem következtetésre és tanulásra is használhatóak.

5.1. Miért fontosak a grafikus modellek?

Tegyük fel, hogy számításainkhoz fel szeretnénk használni néhány (diszkretizált) valószínűségi eloszlást. A helyzet nem túl bíztató: N bináris bemenet esetén a teljes eloszlás megadásához 2^N valószínűséget kell eltárolnunk (pontosabban $2^N - 1$-et, hiszen az összes valószínűség összege 1 kell hogy legyen, így az utolsó érték a többi alapján meghatározható). Ez mindenesetre egy hatalmas szám, figyelembe véve hogy a legalapvetőbb Moricka-példáktól eltekintve akár több tíz, vagy többszáz bináris bemenetünk is lehet.

A probléma azonban nemcsak ebben áll. Tegyük fel, hogy valahogy sikerült reprezentálnunk egy N-dimenziós valószínűségi eloszlást, és segítségével következtetéseket szeretnénk végezni. Egy tipikus következtetési feladat során a bemeneti változók egy részhalmaza fölötti valószínűségekre vagyunk kíváncsiak. Ha a részhalmazba tartozó változóknak van egy fix értéke, és arra vagyunk kíváncsiak, mi ennek az értékkombinációnak a valószínűsége, akkor az összes olyan teljes valószínűséget összegeznünk kell, amelyben a többi (részhalmazban nem szereplő) változó értéke is szerepel minden lehetséges kombinációban! Formálisan:

$$p(x_{\mathcal{I}_1}, \ldots, x_{\mathcal{I}_k}) = \sum_{j_1} \cdots \sum_{j_l} p(x_{\mathcal{I}_1}, \ldots x_{\mathcal{I}_k}, X_{\mathcal{J}_1} = j_1, \ldots X_{\mathcal{J}_l} = j_l) \tag{5.1}$$

ahol az eredeti eloszlás változóinak száma $N = k + l$, $\mathcal{I}$ a bennünket érdeklő részhalmaz, $\mathcal{J}$ pedig a részhalmaz komplemense. Ha például $N = 100$, azaz 100 bemeneti változónk van és annak a valószínűségére vagyunk kíváncsiak, hogy az első változó értéke 1, akkor az alábbi számítást kell elvégeznünk:

$$p(x_1 = 1) = \sum_{x_2} \sum_{x_3} \cdots \sum_{x_{100}} p(X_1 = 1, X_2 = x_2, X_3 = x_3, \ldots X_{100} = x_{100}) \tag{5.2}$$

Látható, hogy ez a számítás is 2^{99} tag összegzését jelenti akkor, ha a változók binárisak (többértékű változók esetén a helyzet még rosszabb lenne), ami kezelhetetlen költség. Fontos azonban figyelembe venni, hogy a gyakorlatban sokszor ennél egyszerűbb a helyzet, ugyanis bizonyos feltételes függetlenségi viszonyok kihasználásával a számítási igény csökkenthető. Ha például tudjuk, hogy X_1, X_2 és X_3 valószínűsége független $X_{50}, \ldots X_{100}$ változók értékétől abban az esetben, ha $X_1, \ldots X_{49}$ értékeit már ismerjük, akkor a számítandó kifejezés az alábbiak szerint egyszerűsödik:

$$\begin{aligned}
p(x_1 = 1) &= \sum_{x_2} \sum_{x_3} \cdots \sum_{x_{100}} p(X_1 = 1, X_2 = x_2, X_3 = x_3, \ldots X_{100} = x_{100}) = \\
&= \sum_{x_2} \sum_{x_3} \cdots \sum_{x_{100}} p(X_1 = 1 | X_2 = x_2, X_3 = x_3, \ldots X_{49} = x_{49}, \cancel{X_{50} = x_{50}, \ldots X_{100} = x_{100}}) \cdot \\
&\quad p(X_2 = x_2 | X_3 = x_3, \ldots X_{49} = x_{49}, \cancel{X_{50} = x_{50}, \ldots X_{100} = x_{100}}) \cdot \\
&\quad p(X_3 = x_3 | X_4 = x_4, \ldots X_{49} = x_{49}, \cancel{X_{50} = x_{50}, \ldots X_{100} = x_{100}}) \cdot \\
&\quad p(X_4 = x_4, \ldots X_{100} = x_{100}) = \\
&= \sum_{x_2} \cdots \sum_{x_{49}} p(X_1 = 1 | X_2 = x_2, X_3 = x_3, \ldots X_{49} = x_{49}) \cdot \\
&\quad p(X_2 = x_2 | X_3 = x_3, \ldots X_{49} = x_{49}) \cdot \\
&\quad p(X_3 = x_3 | X_4 = x_4, \ldots X_{49} = x_{49}) \cdot \\
&\quad \underbrace{\sum_{x_{50}} \cdots \sum_{x_{100}} p(X_4 = x_4, \ldots X_{100} = x_{100})}_{c}
\end{aligned} \tag{5.3}$$

Mennyi így a megtakarításunk? Egyfajta *dinamikus programozással*[1] a c-vel jelölt kifejezés részben külön számítható (2^{51} tag összegzésével, csak ezt 2^{46} alkalommal kell megtennünk amikor az $X_4 \dots X_{49}$ változók valamelyikének módosul az értéke, vagyis összesen 2^{97} művelet elvégzésével), ezt követően pedig a maradék 2^{48} tagot kell összegeznünk. Fontos, hogy ezek is nagy számok ugyan, de a $2^{97} + 2^{48}$ nagyságrendileg kisebb szám, mint a 2^{99}. Továbbá, mint ahogy gyakorlati példákon keresztül látni fogjuk, a gyakorlatban ennél nagyobb megtakarítások is lehetségesek (jelen esetben sem esett szó az első három változó melletti többi változó közötti feltételes függetlenségekről).

Fentiek alapján sejthetjük, hogy a grafikus modellek hatékony reprezentációt adhatnak sokváltozós valószínűségi eloszlásokra és a velük történő számításokra. Összességében három szempont szerint beszélhetünk a grafikus modellek előnyeiről, ezeket vesszük sorra a szakasz maradék részében.

5.1.1. Reprezentációs hatékonyság

A grafikus modellek előnye, hogy *deklaratív reprezentációt* adnak a tudás tárolására. A deklaratív, vagy kijelentő reprezentációk azért előnyösebbek sokszor, mint az imperatív reprezentációk, mert így a felhasznált reprezentáció független attól, hogy azt milyen algoritmusokon keresztül értelmezzük, vagy hogy egyáltalán milyen tématerületen belül dolgozunk. A deklaratív reprezentációk:

1. Alkalmazási területtől függetlenül alkalmasak tudás tárolására
2. Anélkül finomíthatóak / módosíthatóak, hogy a rajtuk operáló következtető- és tanuló-algoritmusok módosulnának

A grafikus modelleknek további előnye reprezentációs szempontból, hogy jó az interpretálhatóságuk. Ez azt jelenti, hogy ha egy a tématerületen jártas ember ránéz egy grafikus modellre, sokszor azonnal észreveszi, ha valami nem stimmel rajta. Ezen kívül bizonyos kérdéseket könnyen megfogalmazhat, pl: „vizsgáljuk meg, mi történne ha ezt a változót vagy valószínűséget itt így módosítanám", vagy éppen „mi történt volna, ha ezt a változót itt nem így állítottuk volna be?" (ez utóbbi az ún. *counterfactual* eset).

Reprezentációs szempontból fontos végezetül, hogy a grafikus modellek sokféle finomsági szinten teszik lehetővé a problémák reprezentálását. Ha szeretnénk, könnyedén a „szőnyeg alá söpörhetünk" bizonyos bennünket nem érdeklő részleteket, így például azt hogy valami közelebbről nem meghatározott „várlatlan" dolog történik.

5.1.2. Következtetési hatékonyság

Mint ahogy korábban említettük, a következtetés grafikus modellek esetén azt jelenti, hogy egy teljes eloszlás reprezentációja alapján meg szeretnénk mondani, hogy a változók egy részhalmazának valamilyen együttes érték-kombinációja milyen valószínűséggel fordulhat elő. Az ilyen kérdések megválaszolására a grafikus modellek kifejezetten alkalmasak, mint ahogy a szakasz bevezetőjében láthattuk.

[1] Ez a R. Bellman által javasolt elnevezés annyit jelent, hogy bizonyos részeredmények sokszor történő újraszámítása helyett a köztes eredményeket 'okosan' eltárolva (angolul: *memoization*) kezelhetetlen számítások kezelhetővé válnak.

5.1.3. Illeszkedés tanulási paradigmákhoz

Végül fontos megemlíteni, hogy a grafikus modellek gyakran hatékonyan tanulhatóak. Ilyenkor a modellt nem szakértő hozza létre, hanem azt akár teljesen automatizáltan, akár szakértő bevonásával hibrid módon hozzuk létre. A modell szelekció elmélete is kifejezetten jól alkalmazható grafikus modellekre, vagyis ha van 2 vagy több alternatívánk, hatékonyan kiválasztható az a modell, amelyik az adatainkhoz bizonyos szempontból leginkább illeszkedik.

5.1.4. Grafikus modellek alkalmazási területei

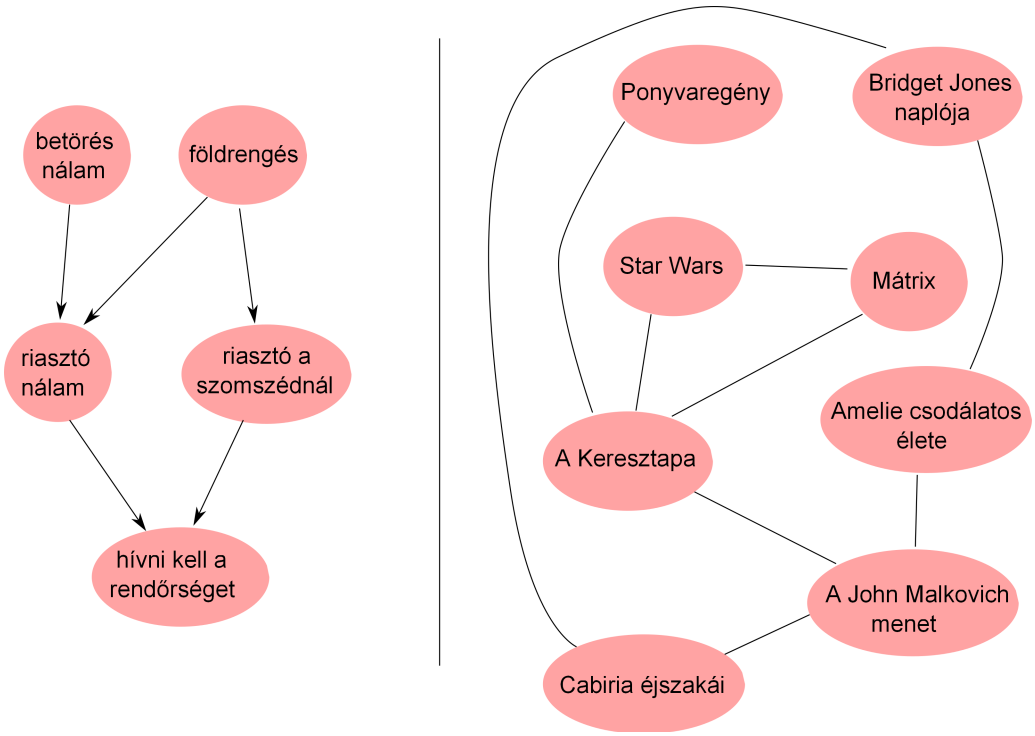
Mivel a fent tárgyalt módon a grafikus modellek segítenek szétválasztani a tárgyterület problémareprezentációjának sajátosságait a felhasznált következtető- és tanulóalgoritmusoktól, kifejezetten alkalmasak a magasszintű (szimbolikus) tudás hasznosítására. Segítségükkel számos sikeres alkalmazás valósult meg pl. az orvosi diagnosztika, műszaki hibaelemzés, vagy éppen a genetikai szabályosságok vizsgálatának területén. Ezen túlmenően a terület egyre inkább kezd összefondódni a szubszimbolikus (pl. neurális) módszerekkel is, például képek szegmentálása és zajszűrése területén, ahol egy-egy képpont vagy képrész valamilyen szürkeskálájú vagy színértékkel rendelkező valószínűségi változónak tekinthető (ilyenkor a szomszédos területek erősebb kölcsönhatásban vannak egymással, mint az egymástól távolabbra eső területek, ami jól modellezhető valószínűségi háló struktúrájában).

5.2. Grafikus modellek típusai

Nagyvonalakban az irányított és irányítatlan grafikus modelleket szokás megkülönböztetni. Ezek a hálók más jellegű információk kódolására alkalmasak, ezért másfajta inferencia-algoritmusokat is szokás rájuk alkalmazni.

Az **irányított grafikus modellek** nevükből adódóan irányított éleket tartalmaznak, melyeknek van forrás- és célcsomópontjuk. Egy ilyen él azt hivatott reprezentálni, hogy a forrás- és célcsomópont között ok-okozati összefüggés *lehetséges*. Ez a feltételes módbeli megfogalmazás fontos: pusztán egy él megléte nem feltétlenül jelenti azt, hogy az a konkrét eloszlás amelyet a háló modellez valóban olyan is, hogy a célcsomópontbeli érték függ a forráscsomópontbeli értéktől. Ezért majd látni fogjuk, hogy ezek a hálók nem is annyira az ok-okozati összefüggések reprezentálásra alkalmasak, hanem sokkal inkább a feltételes függetlenségek reprezentálására, melyek bizonyos szabályok betartásával kiolvashatóak belőlük.

Irányított grafikus modellre egy példát mutat a 5.1. ábra bal oldala. A modell azt mutatja be, milyen okok vezethetnek arra, hogy valakinek a házában, illetve a szomszédja házában megszólal a riasztó, és ezen keresztül hogy mikor érdemes az illetőnek értesítenie a rendőrséget. Az ábra jól szemlélteti, hogy egy betörés az illető riasztójának megszólalását okozhatja, míg egy földrengés mindkét riasztó megszólalását okozhatja. Továbbá kihatással lehet arra, hogy az illető végül kihívja-e a rendőröket vagy sem, hogy megszólalt-e a riasztója (pozitív értelemben), illetve hogy a szomszéd riasztója megszólalt-e (negatív értelemben). Természetesen olyan is előfordulhat, hogy a betörők egyre ügyesebbek és már előre meg tudják hekkelni a riasztórendszert, vagy hogy az illető minden apró zajra kihívja a rendőrséget, ilyenkor az élek mentén nem feltétlenül figyelhető meg ok-okozati összefüggés. A fő hangsúly azonban azon van, hogy ez is egy modell, amit a megfigyelések vagy megerősítenek, vagy nem (ez utóbbi esetben a modellt célszerű bővíteni, pl. „a betörő ügyes„ illetve



5.1. ábra. Példa irányított és irányítatlan grafikus modellre.

„az illető zsémbes öregasszony” című változók felvételével). Egy ilyen modellen működő következtető-algoritmus megmondhatná például, hogy ha pusztán annyit tudunk hogy valakinek megszólalt a riasztója, mekkora valószínűséggel fogja kihívni a rendőröket; vagy, hogy ha nem vagyunk otthon és értesülünk hogy megszólalt a riasztónk, majd felhívjuk a szomszédunkat hogy nála megszólalt-e a riasztó, akkor a kapott válasz függvényében hogyan érdemes cselekednünk. Tanulás esetén pedig automatizáltan vagy félig automatizáltan finomítanánk a modellt.

Az **irányítatlan grafikus modellek** nevükből adódóan irány nélküli (vagy kétirányú, ami ezzel ekvivalens) éleket tartalmaznak. Egy ilyen él azt hivatatott reprezentálni, hogy az általa összekötött csomópontok értékei együtt szoktak változni, vagyis valamilyen erősségű pozitív vagy negatív korreláció létezik közöttük. Egy ilyen grafikus modell reprezentációs képessége merőben más, mint az irányított grafikus modelleké: hogy a kauzáció és a korreláció mennyire mást jelentenek, azt jól szemlélteti Tyler Vigen humoros oldala (http://tylervigen.com/spurious-correlations). Egy példa: vajon pusztán az a tény, hogy nyáron több a vízbe fulladásos baleset, mint télen, és hogy ugyanígy nyáron több fagylaltot esznek az emberek, mint télen jelentheti-e azt, hogy a fagylaltevés vízbe fulladásos halált okozhat? Nem, a kettő között nincs összefüggés: ehelyett azt mondhatjuk, hogy létezik egy közös rejtett tényező, vagyis az évszak, ami döntően befolyásolja azt is, hogy sok fagylalt fogy, és azt is, hogy megnő a vízbe fulladásos balesetek száma. Általánosságban amikor két változó (X_1 és X_2) között korrelációt fedezünk fel, annak több oka is lehetséges:

- X_1 magával vonja (okozza) X_2-őt
- X_2 magával vonja (okozza) X_1-et
- Létezik egy rejtett X_3 tényező, amely együttesen magával vonja X_1-et és X_2-őt is

- Csak a véletlen műve, hogy korrelációt figyeltünk meg és az is elképzelhető, hogy újabb mérések azt nem támasztanák alá

Irányítatlan grafikus modellre egy példát mutat a 5.1. ábra jobb oldala. A modell egy Móricka-példa arra vonatkozóan, milyen adatbázisokat használhat pl. a Netflix vagy hasonló filmekre vonatkozó ajánlórendszer. Az élek ebben a gráfban azt mutatják be, mennyire jelenthető ki hogy egy tetszőleges felhasználó esetében két film szeretete vagy nem szeretete mennyire mozoghat együtt vagy ellentétesen. A példa alapján úgy tűnhet, hogy például akik a Star Warst szeretik (vagy nem) azok gyakran a Mátrixot is szeretik (vagy nem); illetve az is hogy aki a Cabiria éjszakáihoz hasonló művészfilmek kedvelője, esetleg elítélheti a Bridget Jones naplója típusú filmek által kínált könnyed szórakozást (ilyen negatív korreláció esetén használhatunk majd negatív éleket). Ami lényeges, hogy ez a modell is egyfajta tapasztalati tudást ír le, csak nem ok-okozati összefüggésekkel mint a korábbi irányított modell, hanem gyengébb korrelációs összefüggéseken keresztül. Ezzel együtt persze az irányítatlan modellekre is léteznek (másfajta) következtető- és tanulóalgoritmusok, melyek segítségével hézagos információk alapján megállapíthatnánk, hogy egy felhasználónak melyik filmet ajánljuk leginkább, vagy sok visszajelzés alapján automatizáltan is finomíthatnánk a modellt.

5.3. Bayes hiedelemhálók

Bayes-hiedelemháló (vagy rövidebben: Bayes-háló) minden olyan felbontása egy p valószínűségi eloszlásnak, melynek alakja:

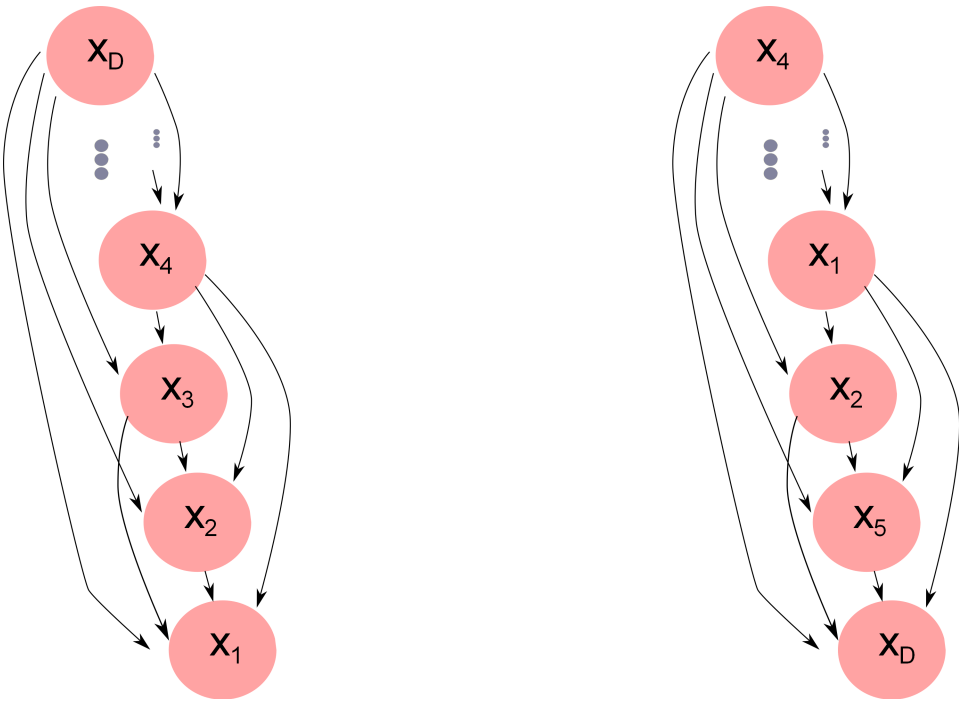
$$p(x_1, \ldots, x_D) = \prod_{d=1}^{D} p(x_d | pa(x_d)) \tag{5.4}$$

ahol $pa(x)$ az x változó *szülőváltozóit* (parents) jelenti. Látható, hogy a definíció szerint a felbontás olyan szorzatként kell, hogy kifejezhető legyen, amelyben:

- pont annyi tényező van, ahány valószínűségi változó
- mindegyik tényező más és más változóról szól
- mindegyik tényező egy feltételes valószínűség, mely feltételében az adott változó szülőváltozói találhatóak

A „szülőváltozó" elnevezés nem véletlen, ugyanis ez a leírás egy-az-egyben megfeleltethető egy irányított gráfnak, melyben D különböző csomópont szerepel és mindegyik x_d csomópontba azon $pa(x_d)$ csomópontokból érkeznek élek, amelyek az x_d-hez tartozó tényező feltételében szerepelnek.

Fontos látni, hogy a láncszabály segítségével minden teljes eloszlás felbontható így. Ehhez annyit kell tennünk, hogy a változókat valamilyen sorrendben kiemeljük és a maradék (még nem kiemelt) változót tekintjük az adott változó szülőváltozóinak. Vagyis bármely $\pi = (\pi_1, \pi_2, \ldots \pi_D)$ sorrendezés esetén:



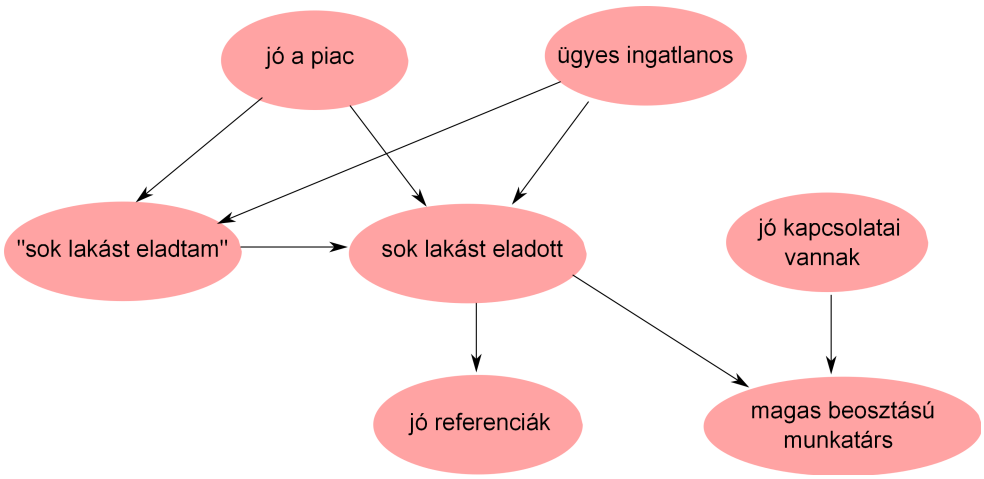
5.2. ábra. Kétféle Bayes-háló teljes eloszlás láncszabály útján történő naív felbontása alapján.

$$
\begin{aligned}
p(x_1, \ldots, x_D) &= p(x_{\pi_1} | x_{\pi_2}, \ldots, x_{\pi_D}) p(x_{\pi_2}, \ldots, x_{\pi_D}) = \\
&= p(x_{\pi_1} | x_{\pi_2}, \ldots, x_{\pi_D}) p(x_{\pi_2} | x_{\pi_3}, \ldots, x_{\pi_D}) p(x_{\pi_3}, \ldots, x_{\pi_D}) = \\
&= \ldots \\
&= \prod_{d=1}^{D} p(x_{\pi_d} | x_{\pi_{d+1}}, \ldots, x_{\pi_D})
\end{aligned}
\tag{5.5}
$$

Ekkor alapesetben a 5.2 ábrán látható irányított modellt kapjuk. Látható azonban, hogy a π sorrendezéstől függően más lesz a változók viszonya is. Ez egyrészt azért lényeges, mert az ábra bal oldalán szereplő hálóban úgy tekintjük, hogy az x_D változónak jut primér szerep és annak értéke befolyásolja a többit, míg a jobb oldali hálóban az x_4 változónak jut ugyanez a szerep, míg x_D a többi változó állapotának következtében kap(hat) értéket (általában). Előfordulhat tudjuk, hogy egy-egy tényezőben „karcsúsítani" tudjuk a feltételben levő változók számosságát, hiszen mint ahogy a bevezetőben láthattuk, lehetnek olyan változók, amik bizonyos más változók ismeretében feltételesen függetlenek. Ahhoz viszont, hogy a tényezőket a lehető leginkább karcsúsítani tudjuk, lényeges szempont hogy a kezdeti π sorrendezés optimális legyen.

Ha pl. tudjuk is, hogy x_4 feltételesen független x_2 értékétől x_1 ismeretében, a 5.2 ábra bal oldali hálójában ezt nem tudjuk kihasználni, hiszen nincsen olyan tényező a felbontásban, amely x_4-ről (vagy x_2-ről) szólna és a feltételében szerepelne x_2 (vagy x_4) és x_1 is. A jobb oldali hálóban ezzel szemben létezik egy $p(x_2|x_1, \ldots, x_4)$ tényező, melynek feltételéből x_4 tehát kihúzható.

Ezek az egyszerűsítések azért lényegesek, mert később a számításokat megkönnyítik. Általánosságban azonban minden lehetséges sorrendezés végigpróbálása nélkül eldönthetetlen kérdés, hogy melyik



5.3. ábra. Példa érvelési típusok szemléltetéséhez.

sorrendezés az optimális; ezért fontos a szakértői tudás, amely alapján megmondhatjuk hogy melyek azok az elsődleges változók amelyek a többire primér módon kihatnak.

5.3.1. Érvelés típusai hiedelemhálókban

Ezt a kérdést igyekszünk példán keresztül megválaszolni. Tegyük fel, hogy szeretnénk eladni a lakásunkat, ehhez pedig célszerű ügyes ingatlanközvetítőt választani. Azonban ahogy a 5.3. ábra is mutatja, ebben a kérdésben közvetlenül nehezen tudunk döntést hozni. Segítségünkre szolgál azonban, hogy hallhatjuk az ingatlanos bemutatkozását, láthatjuk az ingatlanos ajánlásait, valamint a cégen belüli beosztását. Ekkor az alábbiak szerint háromféle érvelési típust különböztethetünk meg.

Kauzális (okok alapján történő) érvelés

Tegyük fel, hogy az ingatlanos azt mondja magáról, hogy amióta a bizniszben utazik, rengeteg lakást adott el. Ezt vagy elhisszük, vagy nem, mindenesetre a modellünk szerint (5.3. ábra) mutat él az ingatlanos nyilatkozatától a „sok lakást eladott” nevű változóba. Ez azt jelenti, hogy a nyilatkozata mindenképpen befolyásolja abba vetett hitünket, hogy sok (vagy éppen kevés, ha nem nyilatkozik ilyet) ingatlant adott el.

Ezt a fajta, okok felől okozatok felé történő érvelést értelemszerűen **kauzális érvelésnek** nevezzük.

Evidenciális (okozatok alapján történő) érvelés

Tegyük fel, hogy az ingatlanos ezek után mutat egy weboldalt, ahol legalább 15 referencia található a legkülönbözőbb hátterű ügyfelekről, melyekben az ingatlanos munkájával kapcsolatos elégedettségüknek adnak hangot. Ez egyfajta bizonyíték arra vonatkozóan, hogy az ingatlanos tényleg jól végzi a dolgot. Láthatjuk, hogy ennek megfelelően a modellben található él a „sok lakást eladott” és „jó referenciái vannak” változók között. Persze ez az él a korábbi felől az utóbbi felé mutat, és nem fordítva, de ez érveléskor nem jelent problémát, a jó referenciák egyfajta bizonyítékként szolgálnak, tehát a hatások

iránya az éleken fordított irányba is terjedhet. Ez a bizonyíték tehát azt eredményezi, hogy akkor is jobban hiszünk benne, hogy az ingatlanos sok lakást eladott, ha az adott szava nem is teljesen győzött meg bennünket.

Ezt a fajta, okozatok felől okok felé történő érvelést **evidenciális érvelésnek** nevezzük.

Interkauzális (okok közötti) érvelés

Van azonban egy probléma. Találkozunk több ismerőssel, akiktől megtudjuk, hogy az elmúlt két évben nagyon pörgött az ingatlanpiac, és boldog-boldogtalan lakást vásárolt. Hogyan változtatja ez meg az ingatlanosról alkotott képünket? Természetesen nem tagadható, hogy szakmailag eredményes emberről van szó, ugyanakkor némiképpen árnyalhatja a képet, ha megtudjuk hogy esetleg könnyű dolga is volt. Ahogy a modellben látható, az ingatlanos nyilatkozatát tükröző változó, valamint a „sok lakást eladott" nevű változó felé fentebbi irányból két él is érkezik: egyrészt a „jó a piac" feliratú változótól, másrészt az „ügyes ingatlanos" feliratú változótól. Minket ez utóbbi változó valószínűsége érdekel (már csak azért is, mert ha most rosszabb a piac mint az elmúlt időszakban volt, akkor ez létfontosságú információ), azonban a példa alapján is látható, hogy a két ok fennállása egymásra is tud hatni.

Ezt követően tegyük fel hogy újabb összefüggésre derül fény: némi kutakodás után rájövünk, hogy az ingatlanos vezetékneve kötőjeles, és a kötőjel után ugyanaz a név szerepel, mint ami a cég tulajdonosának neve. A cégvezér lányát vette el feleségül! Ezek után a „jó kapcsolatai vannak" nevű változó valószínűsége megnő és kevéssé lepődünk meg azon, hogy magas beosztásban dolgozik a cégnél. Viszont ez úgyszólván ki is magyarázza a „sok ingatlant eladott" nevű változót: nem volt feltétlenül szükség arra, hogy sok ingatlant eladjon ahhoz, hogy jó pozícióba kerüljön.

Ezt a fajta okok közötti érvelést **interkauzális érvelésnek** vagy más szóval **kimagyarázásnak** nevezzük. Fontos, hogy ezt a fajta érvelést csak az okozat megfigyelését követően alkalmazhatjuk. Például ha nem tudunk semmit arról, hogy az ingatlanos sok vagy kevés ingatlat adott el, akkor éppenséggel független annak a valószínűsége hogy jó a piac attól, hogy az adott ingatlanos ügyes-e. Hogy megértsük, miért van ez így, képzeljük el hogy jó meg rossz piaci viszonyok között találkozunk egy random ingatlanossal. A két esetben annak a valószínűsége, hogy az ingatlanos inherensen ügyes, nem lehet más, hiszen random módon választottuk ki az összes ingatlanos közül, a piaci viszonyok pedig sem a választásunkra, sem pedig az ingatlanos képességeire vonatkozóan nincs hatással.

5.3.2. Mit fejeznek ki a Bayes-hálók?

Hogy mit fejez ki egy Bayes-háló, azt az ún. **Bayes-háló feltételezés** mondja ki: *minden változó feltételesen független a belőle nem leszármazó változóktól, ha adottak a szülei.* Ez az állítás mindenesetre igaz, de segítségével nehezen lehet függetlenségi viszonyokról általánosságban érvelni. Mi történik például akkor, ha olyan változók közötti viszonyokra vagyunk kíváncsiak, amelyek egymásból származnak?

Noha a Bayes-feltételezésről fontos tudni, éppen a felmerülő kérdés miatt célszerű általánosabban a fentebb ismertetett érvelési fajtákból kiindulni. Láthattuk, hogy a hatások élek mentén és élekkel ellentétes irányba is terjedhetnek. Mi szükség van akkor irányított élekre?

A kérdésre az interkauzális érvelés ad választ, illetve a kimagyarázás jelenségén keresztül számunkra nyilvánvalóvá lett tény, miszerint nem mindegy hogy egy változó értékét megfigyeljük, vagy sem. Amikor két változó között lehetséges hatásokat keresünk, akkor olyan irányítatlan utakat keresünk közöttük, melyeken a hatás terjedni tud. De vegyük észre, hogy pont fordított esetben tud a hatás tovaterjedni, amikor ún. „V-alakzatokkal" ($A \rightarrow B \leftarrow C$ formájú alakzatokkal) találkozunk, mint amikor nem.Ez azt jelenti, hogy:

- $A \rightarrow B \rightarrow C$ alkazat esetében B változó megfigyelésével A és C függetlenek lesznek egymástól, hiszen C valószínűségének megállapítását A értéke közvetlenül már nem befolyásolja (elég hogy B-t ismerem); és fordítva;
- $A \rightarrow B \leftarrow C$ alakzat esetében ellenben B változó megfigyelésével a korábban független A és C változók viszont már nem lesznek egymástól függetlenek

A példánál maradva, ha nem figyeljük meg, hogy az ingatlanos sok ingatlant adott-e el, akkor független marad egymástól, hogy mennyire jó a piac, és hogy ő mennyire ügyes. A hatás ezen az útvonalon csak akkor tud terjedni, ha megfigyeljük a közös okozatot. Illetőleg még akkor is, ha az okozat valamely leszármazottját, vagy éppen egyéb szülőjét megfigyeljük! Ilyenkor ugyanis áttételesen tudunk vele kapcsolatos következtetésre jutni, az evidenciális és kauzális következtetési módokon keresztül!

Ezzel szemben ha három változó A, B és C között $A \rightarrow B \rightarrow C$ vagy $A \leftarrow B \rightarrow C$ irányba mennek élek, az semmi ilyesmit nem jelent, hanem inkább azt, hogy A és C függetlenek B ismeretében.

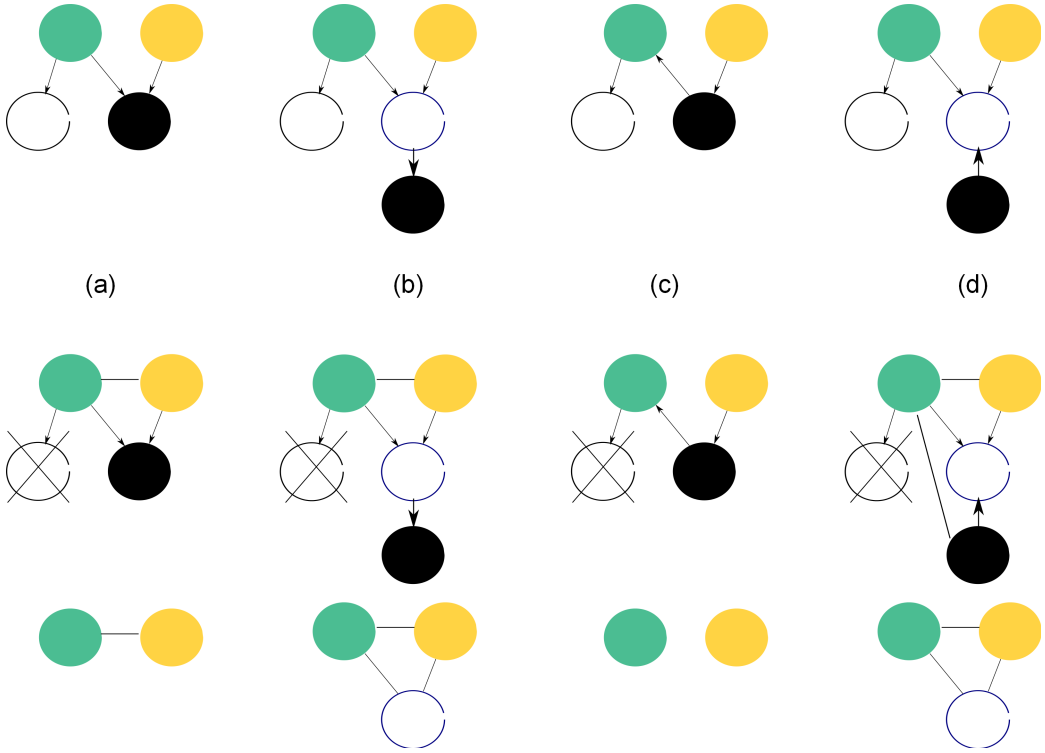
Összességében: *Bayes-hálókban egy valószínűségi változók közötti útvonal* ***aktív****, ha a rajta szereplő összes ($A \rightarrow B \leftarrow C$ alakú) „V-alakzat" aktiválva van, és a rajta szereplő összes többi változó értékét nem figyeljük meg. Egy „V-alakzat" akkor van aktiválva, ha a középen levő (B) valószínűségi változó, vagy annak valamely egyéb, az útvonalon nem szereplő szülőjének vagy leszármazottjának értékét megfigyeljük.* Azt mondjuk, hogy $\mathcal{B}$ *változó-halmaz* ***d-szeparálja*** *az $\mathcal{A}$ és $\mathcal{C}$ változóhalmazokat, ha $\mathcal{B}$ megfigyelését követően nem létezik aktív út $\mathcal{A}$ és $\mathcal{C}$ között.*

Fontos, hogy a d-szeparáció tulajdonképpen a klasszikus értelemben vett feltételes függetlenséget jelenti.

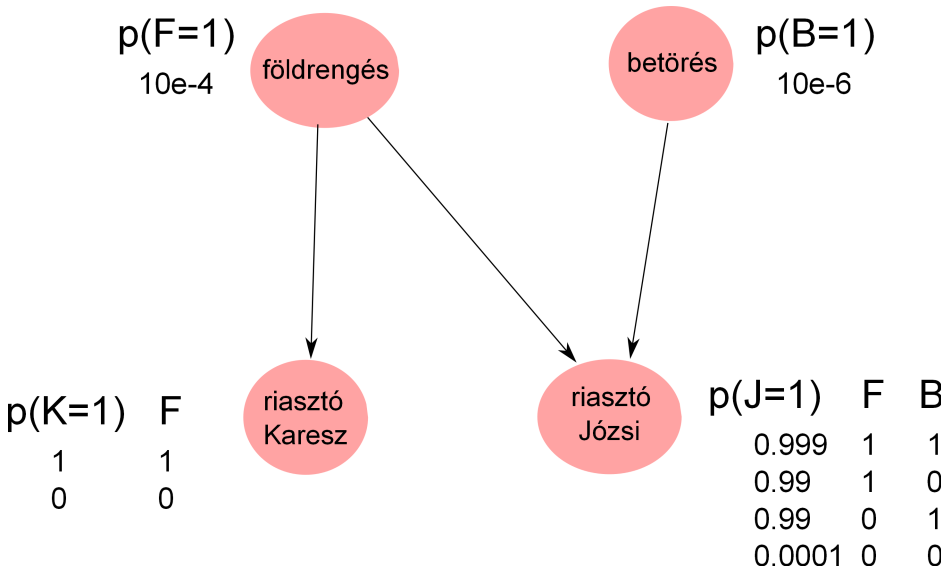
D-szeparáció megállapítása moralizációs algoritmussal

Általános esetben nehéz ránézésre megmondani, hogy két változó-halmaz egy harmadik ismeretében d-szeparációban áll-e. Ez egyrészt azért igaz, mert az aktív utak irányítatlanok ezért nem feltétlenül kell, hogy a Bayes-háló élei az út irányába mutassanak; másrészt pedig azért, mert az a $\mathcal{B}$ halmazbeli csomópont, amely egy utat aktívvá tesz, valamely úton lévő csomópont bármely távoli leszármazottja is lehet. Ezt szemlélteti a 5.4. ábra felső sora, ahol az a és b esetekben a zöld és sárga csomópontok ugyanúgy függő viszonyban vannak a fekete csomópont ismeretében, illetve ahol az is látszik, hogy d esetben mindez nem változik, függetlenül attól hogy a fekete csomópontot érintő él iránya megfordult. Létezik azonban egy négylépéses algoritmus, ami könnyedén alkalmazható, és amelyet az ábrában is felhasználunk a függőségek / függetlenségek megítélésére:

1. Ősháló megkonstruálása: az összes olyan csomópontot a beléjük menő és belőlük induló élekkel együtt töröljük a hálóból, amelyek nem tartoznak az $\mathcal{A}$, $\mathcal{B}$ vagy $\mathcal{C}$ halmazokba és nem ősük valamely ezen halmazokba tartozó változónak.



5.4. ábra. Példák d-szeparáció alapú függetlenségre moralizációs algoritmus alkalmazásával. A d-szeparáció eredeti definíciójából kiindulva is látszik, hogy az a, c és d esetekben a zöld és sárga csomópontok között egyetlen út létezik, melyet a fekete csomópont aktívvá tesz. A c esetben azonban hiába van rajta a két csomópont közötti útvonalon fekete csomópont, ugyanis nincsen a fekete csomópontnak az útvonal mentén legalább két szülője, tehát nem V-alakzatban helyezkedik el ezért a megfigyelése éppen hogy össszefüggővé teszi a szomszédait.



5.5. ábra. Számszerű élda kimagyarázás szemléltetéséhez.

2. Moralizáció: azokat a csomópontokat, melyeknek van közös gyermeke de nincsen közöttük él, összekötjük egy irányítatlan éllel

3. Diszorientáció: minden fennmaradt élnek töröljük az irányát.

4. Adottak törlése: éleikkel együtt töröljük azon csomópontokat a gráfból, melyek a $\mathcal{B}$ halmazban levő változókhoz tartoznak

Ezt követően ha maradt irányított út az $\mathcal{A}$ halmazbeli és a $\mathcal{C}$ halmazbeli csomópontok között, akkor feltételesen függetlenek a $\mathcal{B}$-beli csomópontoktól.

5.3.3. Példa számolással

Klasszikus számolásos példát mutatunk be, a 5.5 ábra szerint. Tegyük fel, hogy Józsi a munkahelyén értesül róla, hogy megszólalt a riasztója ($J = 1$). Nem tudja, hogy betörés volt nála ($B = 1?$), vagy csak földrengés okozta ($F = 1?$)... ezért aggódva felhívja Kareszt, aki azt mondja, nála is megszólalt a riasztó ($K = 1$). Ezek után Józsi megnyugszik: a földrengés (vagy bármilyen közös ok ami mindkét riasztó megszólalását okozhatja) kimagyarázza a betörés lehetőségét. Más szóval: ahhoz, hogy Józsi megmagyarázza, miért szólalt meg a riasztója, már nincs szüksége arra, hogy betörést feltételezzen.

Hogy néz ez ki formálisan? Az alábbi felbontást érdemes használni (az ok-okozati viszonyok miatt így logikus, azonban természetesen grafikusan egyszerűbb elindulni, ahogy a 5.5 ábrán tettük):

$$\begin{aligned} p(F,B,K,J) &= p(J|K,F,B)p(K,F,B) = \\ &= p(J|K,F,B)p(K|F,B)p(F|B)p(B) = \\ &= p(J|F,B)p(K|F)p(F)p(B) \end{aligned} \tag{5.6}$$

A végén egyrészt kiderült, hogy Józsi riasztója nem függ Karesz riasztójának megszólalásától ha már F változót ismerjük, vagyis tudjuk hogy volt-e földrengés vagy sem (ez azért fontos, mert természetesen a két riasztó össze is függhet, ha mindkettőt földrengés okozza, de ha már tudjuk hogy volt-e földrengés vagy sem, egymástól függetlenül válaszolhatunk arra a két kérdésre, hogy egyik vagy másik riasztó megszólalásának mi a valószínűsége). Nem pont ugyanígy de hasonló módon a földrengés (F) is független a betöréstől (B) a tekinteben, hogy a világról alkotott ismereteink szerint az utóbbi nem okozhatja a korábbit; ahogy Karesz riasztójának megszólalása (K) sem függhet attól, hogy Józsihoz betörtek-e (B). Ezt a három tényt felhasználva jutunk el a felbontás végső formájához, vagyis a 5.5. ábrához.

Ha most valószínűségekkel szeretnénk dolgozni, minden csomóponthoz meg kell adnunk, hogy a benne levő változó értékeinek mi a valószínűsége a szülőváltozók különböző értékkombinációi esetén. Látható, hogy ehhez a felbontásnak köszönhetően 15 helyett 8 értéket kell csak megadnunk. Ezt követően, ha arra vagyunk kíváncsiak, mi a valószínűsége annak hogy Józsihoz betörtek, ha megszólalt a riasztója, azt kapjuk hogy:

$$
\begin{aligned}
p(B|J) = \frac{p(B,J)}{p(J)} &= \frac{\sum\limits_{K,F} p(J|F,B)p(K|F)p(F)p(B)}{\sum\limits_{K,F,B} p(J|F,B)p(K|F)p(F)p(B)} = \\
&= \frac{p(B)\sum\limits_{F} p(J|F,B)p(F)\sum\limits_{K} p(K|F)}{\sum\limits_{F,B} p(J|F,B)p(F)p(B)\sum\limits_{K} p(K|F)} = \\
&= \frac{p(B)\sum\limits_{F} p(J|F,B)p(F)}{\sum\limits_{F,B} p(J|F,B)p(F)p(B)} =
\end{aligned}
\tag{5.7}
$$

Itt azt használtuk ki, hogy a szummák átrendezhetőek annak függvényében, hogy mely tényezők függnek a futó paramétertől és melyek nem. A végén a K értékei fölött összegzett feltételes valószínűség össze biztosan 1, hiszen valószínűség.

Ha a számlálót $B = 1$-re és $B = 0$-ra is kiszámítjuk, a nevezőt nem kell kiszámítanunk hiszen a kettő eredmény összege 1 kell, hogy legyen (csak skáláznunk kell). Ez alapján:

$$
\begin{aligned}
numerator_0 &= p(B=0)\sum_F p(J|F,B=0)p(F) = \\
&= 9.99999 * 10^{-1} \cdot \left(\cdot 10^{-4} \cdot 9.999 \cdot 10^{-1} + \cdot 0.99 \cdot 10^{-4}\right) = \\
&= 9.99999 * 10^{-1} \cdot \left(9.999 \cdot 10^{-5} + 9.99 \cdot 10^{-5}\right) = 9.99999 * 10^{-1} \cdot 1.9989 \cdot 10^{-4} = 1.9989 \cdot 10^{4}
\end{aligned}
$$

$$
\begin{aligned}
numerator_1 &= p(B=1)\sum_F p(J|F,B=1)p(F) = \\
&= 10^{-6} \cdot (\cdot 0.99 \cdot 0.0001 + \cdot 0.999 \cdot 0.9999) = \\
&= 10^{-6} \cdot 0.99 = 9.9 \cdot 10^{-7}
\end{aligned}
\tag{5.8}
$$

Ez alapján:

$$P(B=1|J=1)=\frac{9.9\cdot 10^{-7}}{9.9\cdot 10^{-7}+1.99\cdot 10^{-4}}=0.00495\approx 0.5\% \tag{5.9}$$

Ehhez képest, ha azt is tudjuk, hogy Karesz riasztója megszólalt, a számítások az alábbiak szerint módosulnak:

$$\begin{aligned}
numerator_0 &= \sum_F p(J|F,B=0)p(K=1|F)p(F)p(B=0) = \\
&= p(B=0)\sum_F p(K=1|F)p(J|F,B=0)p(F) = \\
&= 9.99999*10^{-1}\cdot\left(\cancel{0\cdot 10^{-4}\cdot 9.999\cdot 10^{-1}}+1\cdot 0.99\cdot 10^{-4}\right) = \\
&= 9.99999*10^{-1}\cdot 9.9\cdot 10^{-4}=9.9\cdot 10^{-4}
\end{aligned} \tag{5.10}$$

$$\begin{aligned}
numerator_1 &= \sum_F p(J|F,B=1)p(K=1|F)p(F)p(B=1) = \\
&= p(B=1)\sum_F p(K=1|F)p(J|F,B=1)p(F) = \\
&= 10^{-6}\cdot\left(\cancel{0\cdot 0.99\cdot 0.0001}+1\cdot 0.999\cdot 0.9999\right) = \\
&= 10^{-6}\cdot 0.99=9.9\cdot 10^{-7}
\end{aligned}$$

Ez alapján pedig:

$$P(B=1|J=1,K=1)=\frac{9.9\cdot 10^{-7}}{9.9\cdot 10^{-7}+9.9\cdot 10^{-4}}=0.00099\approx 0.1\% \tag{5.11}$$

Másszóval a kimagyarázás jelensége ötödére csökkentette annak a valószínűségét, hogy Józsihoz betörtek.

5.3.4. Következtetés Bayes-hálókban

Érdekes módon be lehet bizonyítani, hogy tetszőleges Bayes-hálóban való következtetés bonyolultsága NP-teljes!

Ha ugyanis meg tudnánk tenni, tetszőleges 3-SAT kifejezéshez konstruálható lenne olyan gráf, melyben a 3-SAT kérdés eldöntése megfelelne egy olyan kérdéssel, hogy $p(x_n)>0$?

Az se segít rajtunk, ha megengedett az approximáció (pl. ha azt mondjuk, hogy adott tűréshatáron belül kell csak megmondani, mennyi lesz vminek a valószínűsége)

Mégis rengeteg olyan számunkra érdekes eset van, ahol még a pontos inferencia sem probléma. Más esetekben meg léteznek olyan approximációs módszerek, amik működhetnek.

Következtetés változó-eliminációval

Képzeljünk el egy $A \to B \to C \to D$ alakú hiedelem-hálót. Tegyük fel, hogy feladatunk $P(B)$ eloszlás meghatározása. Ekkor, mivel B csak A-tól függ(het):

$$P(B) = \sum_a P(a)P(B|a) \tag{5.12}$$

Itt az összes érték rendelkezésre áll $P(a)$-k és $P(b|a)$-k formájában. Ha A változó k-értékű, B változó pedig m-értékű, akkor adott b-hez k darab szorzást kell végezni, és $k-1$ összegzést. Mivel m különböző b-érték lehet, ezért ez összesen $m \times (2k-1) = O(k \times m)$ a művelet bonyolultsága.

Tfh most az a feladat, hogy $P(C)$ eloszlást meghatározzuk! Ehhez:

$$P(C) = \sum_b P(b)P(C|b) \tag{5.13}$$

Itt az összes $P(c|b)$ érték rendelkezésre áll. $P(b)$-t közvetlenül nem ismerjük ugyan, de az előbb kiszámoltuk! Vegyük észre, hogy a háló struktúrája elengedhetetlen ahhoz, hogy ez működjön. Ha pl. C szülője lett volna A is, akkor nem lett volna elég az, hogy P(b)-ket korábban kiszámoltuk. Vegyük észre azt is, hogy egy eloszlás kiszámításakor nem egy számot, hanem egy egész eloszlást számítunk ki. Ez fontos!

Általánosságban, ha van egy $X_1, \ldots, X_n$ láncunk, és mindegyik változó lehetséges értékeinek száma k, akkor:

$$P(X_{i+1}) = \sum_{x_i} P(x_i)P(X_{i+1}|x_i) \tag{5.14}$$

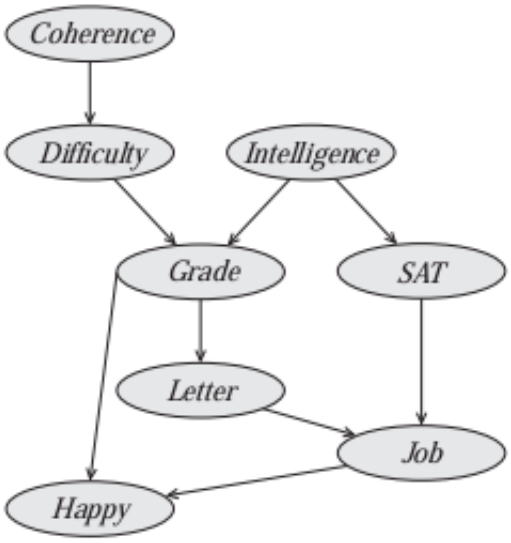
kiszámítása $O(k^2)$ bonyolultságú. Mivel n változó van, ez összesen $O(nk^2)$. Ezzel szemben, ha a teljes eloszlást akartuk volna kifejezni – és abból elvégezni a következtetést – akkor k^n különböző változókombinációhoz kellett volna valószínűséget legenerálni.

Nézzünk egy példát általános jelölésmóddal. Tfh célunk a Job változó valószínűségének meghatározása ($P(J)$). A gráf szerint a teljes valószínűséget kifejezetjük:

$$\begin{aligned} P(C,D,I,G,S,L,J,H) =& P(C)P(D|C)P(I)P(G|I,D)P(S|I) \\ & P(L|G)P(J|L,S)P(H|G,J) \end{aligned} \tag{5.15}$$

Ezt a képletet ún. faktoros tényezőkre bonthatjuk. A faktorok azt fejezik ki, hogy melyik tagban melyik változóról beszélünk, és hogy melyik tag összességében mely változóktól függ:

$$\begin{aligned} P(C,D,I,G,S,L,J,H) =& P(C)P(D|C)P(I)P(G|I,D)P(S|I) \\ & P(L|G)P(J|L,S)P(H|G,J) \\ =& \Phi_C(C)\Phi_D(D,C)\Phi_I(I)\Phi_G(G,I,D)\Phi_S(S,I) \\ & \Phi_L(L,G)\Phi_J(J,L,S)\Phi_H(H,G,J) \end{aligned} \tag{5.16}$$



5.6. ábra. Példa következtetésre (az ábra és a példa is a [4] könyvből származik).

Ha egy változót eliminálunk, az összes olyan tagot össze kell gyűjteni, ahol a paraméterekben szerepel az adott változó. Pl. C-t így eliminálhatjuk:

$$\begin{aligned} \Psi_1(C,D) &= \Phi_C(C) \cdot \Phi_D(D,C) \\ \tau_1(D) &= \sum_C \Psi_1 \end{aligned} \tag{5.17}$$

Ugyanígy folytathatjuk. D eliminálása ($\Phi_D(D,C)$-t már eliminálтuk, de van helyette $\tau_1(D)$ ami meg függ D-től):

$$\begin{aligned} \Psi_2(G,I,D) &= \Phi_G(G,I,D) \cdot \tau_1(D) \\ \tau_2(G,I) &= \sum_D \Psi_2(G,I,D) \end{aligned} \tag{5.18}$$

I eliminálása:

$$\begin{aligned} \Psi_3(G,I,S) &= \Phi_I(I) \cdot \Phi_S(S,I) \cdot \tau_2(G,I) \\ \tau_3(G,S) &= \sum_I \Psi_3(G,I,S) \end{aligned} \tag{5.19}$$

H eliminálása:

$$\begin{aligned} \Psi_4(G,J,H) &= \Phi_H(H,G,J) \\ \tau_4(G,J) &= \sum_H \Psi_4(G,J,H) \end{aligned} \tag{5.20}$$

G eliminálása:

$$\begin{aligned}\Psi_5(G,J,L,S) &= \tau_4(G,J)\cdot\tau_3(G,S)\cdot\Phi_L(L,G)\\ \tau_5(J,L,S) &= \sum_G \Psi_5(G,J,L,S)\end{aligned} \tag{5.21}$$

S eliminálása:

$$\begin{aligned}\Psi_6(J,L,S) &= \tau_5(J,L,S)\cdot\Phi_J(J,L,S)\\ \tau_6(J,L) &= \sum_S \Psi_6(J,L,S)\end{aligned} \tag{5.22}$$

L eliminálása:

$$\begin{aligned}\Psi_7(J,L) &= \tau_6(J,L)\\ \tau_7(J) &= \sum_L \Psi_7(J,L)\end{aligned} \tag{5.23}$$

Ennek eredményeképpen megkapjuk J valószínűségét. A változók persze tetszőleges sorrendben eliminálhatóak, de nem mindig ugyanannyi a költség!

Kérdés: mi a faktorok jelentése? Sokszor úgy néznek ki, mint egy változóra (vagy változóhalmazra) vonatkozó marginalizált valószínűségek.

Például amikor C-t eliminálják, $\tau_D(D)$ tulajdonképpen D valószínűsége

Jó tudni, hogy ez nincs mindig így. Például ha adva van egy ilyen Bayes háló:

$$P(X,A,B,C) = P(X)\cdot P(A|X)\cdot P(B|A)\cdot P(C|B,X) \tag{5.24}$$

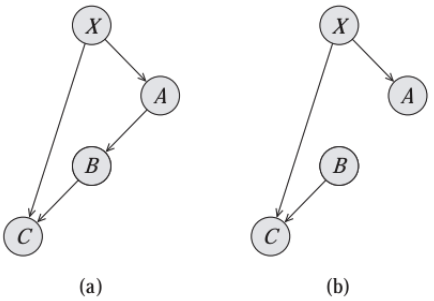
és eliminálják X-et, azt kapjuk, hogy:

$$\tau(A,B,C) = \sum_X P(X)\cdot P(A|X)\cdot P(C|B,X) \tag{5.25}$$

Ez pedig semmilyen valószínűségnek nem felel meg.

$$\tau(A,B,C) = \sum_X P(X)\cdot P(A|X)\cdot P(C|B,X) \tag{5.26}$$

B nem szerepel egyik tag bal oldalán sem, ezért nem is lehetne olyan valószínűség, amiben B benne van bal oldalon. Úgy tűnhet, mintha $P(A,C|B)$-ről lenne szó, de nem. Ehhez az kellene, hogy A független legyen B-től! (csak akkor lehetne a fenti jobb oldali felbontás igaz... vagyis, ha bal helyett jobb):



5.7. ábra. Tényezők szemantikája (az ábra és a példa is a [4] könyvből származik).

$$P(X, A, B, C) = P(X) \cdot P(A|X) \cdot P(C|B, X) \tag{5.27}$$

Mi történik, ha néhány változót megfigyelünk, és úgy szeretnénk következtetni? Tfh látjuk, hogy adott hallgató intelligens, de nem boldog. Ekkor mi a valószínűsége annak, hogy jó állása van? Ehhez elég $P(J, i = 1, h = 0)$ valószínűséget kiszámítanunk. Mivel a következő képletben:

$$P(J|i = 1, h = 0) = \frac{P(J, i = 1, h = 0)}{P(i = 1, h = 0)} \tag{5.28}$$

a nevező konstans J-ben, ezért a nevezőt nem kell kiszámítani! Elég, ha csak a végén normalizáljuk az összes valószínűséget, hogy 1 legyen az összegük. A teljes valószínűséget pedig úgy kapjuk meg, hogy minden változót eliminálunk, kivéve J-t, I-t és H-t.

Nézzük a költséget! Tfh n random változónk van és kezdetben m tényezőnk. Bayes-hálóban egyébként pont $n = m$ lesz mindig, mert változónként van egy tényező: $P(X_i|pa(X_i))$.

Minden lépésben kiválasztunk egy X_i változót, és minden faktort összeszorzunk, amelyben az a változó szerepel. Ekkor megkapjuk Ψ_i-t. Ezt követően „kisummázzuk" a változót a nagy faktorból, és megkapjuk τ_i faktort. Ebben és az összes többi faktorban már nem szerepel X_i. Legyen N_i a Ψ_i faktorban levő bejegyzések száma. Pl. ha az összes változó bináris, és a faktor úgy néz ki hogy $\Psi_i(A, B, C)$, akkor... 8 bejegyzés van. Legyen $N_{max} = \max_i N_i$.

Hány szorzást kell elvégezni?

Az m darab Φ_i tényező mindegyikével egyszer szorzunk. Ennek a költsége legfeljebb N_i, mivel Ψ_i egyetlen bejegyzéséhez Φ_i egyetlen bejegyzésével kell szoroznunk. Pl. itt $N_3 = 8$, ezért 8 darab szorzást kell elvégezni:

$$\Psi_3(G, I, S) = \Phi_I(I) \cdot \Phi_S(S, I) \cdot \tau_2(G, I) \tag{5.29}$$

Összesen $n + m$ faktor lehet, amit be kell szorozni (mert nemcsak Φ-k, hanem τ-k is vannak). Ezért a szorzás költsége $O((n + m)N_{max})$.

Összeadások: minden egyes Ψ esetén egy változót eliminálunk, $N_i - 1$ összeadással (az összes bejegyzést összeadjuk). Ezért az összeadások költsége $O(nN_{max})$

A költség összesen $O(mN_{max})$.

N_{max} azonban exponenciális is lehet a változók számában!

Látszik, hogy minden azon múlik, hogy a köztes Ψ_i tagokban hány változó van. Ha sok, az rossz. Ez pedig az eliminálás sorrendjétől függ.

Sajnos az is NP-teljes probléma, hogy eldöntsük, mi a jó sorrend a változók eliminálására. Léteznek azonban heurisztikák.

Pl. ha pár változót már eliminántunk, célszerű azokkal folytatni, amelyek szomszédjai között a legtöbb eliminált változó van. Miért? Mert az eliminált változó már nem lesz a bejegyzésben!

6. fejezet

Energia alapú modellek – A mély tanulás kezdetei

A mély tanulás napjainkban a gépi tanulás (és általában véve a mesterséges intelligencia) egyik, ha nem a legsikeresebb ágának számít. Ennek az ágnak a kialakulása azonban nem volt triviális, még ha a jövőbeni hatékonyságát illetően végig szilárd intuíciók is kísérték a fejlődését.

Ebben a fejezetben a mély tanulás megvalósításának igényét a nem felügyelt tanulás irányából vezetjük le. Ezt követően áttekintünk néhány alapvető modellt, amelyek létrejötte és elméleti-gyakorlati fejlődése a mély tanulás korai fázisára jelentős mértékben hatott. Jóllehet a tudomány mai állása szerint az újabb módszerek és modellek (pl. a reziduális kapcsolatok, vagy a transformer hálók használata) felülírta ezeknek a korai modelleknek a szerepét, kiindulópontként mindenképpen érdemes ezekkel a korábbi modellekkel is megismerkedni.

6.1. Miért kísérleteznek hosszú ideje a mély tanulással?

Mielőtt a mély tanulás mikéntjének jobban utána néznénk, fontos arról is szót ejteni, hogy pontosan mit jelent ez a fogalom, és miért kezdtek el egyáltalán a kutatók a mély tanulással foglalkozni.

A többrégetű mesterséges neurális hálók egyik alapelve, hogy minden rejtett réteg egy „elosztott reprezentációnak” felel meg. Fontos felismerés, hogy a rejtett rétegek elemei nem kölcsönösen kizáróak (egyszerre több is lehet aktív), ezért mindegyik elem egy külön feature-t jelenthet, amelyek adott bemenet hatására nem kizárólagosan vannak jelen.

Ez a fajta szubszimbolikus működés több szempontból is nagyon vonzó már régóta a kutatók és a mesterséges intelligencia rendszereket fejlesztők számára.

Az intuíció egyrészt az, hogy az emberi agy működése is hasonló módon sok rétegből tevődik össze, ha például az LGN, V1, V2, V4, PIT és AIT agyterületek közötti kapcsolatokat nézzük.

Ennél azonban méginkább kézzel fogható az a felismerés, hogy bizonyos függvényeket sokkal nagyobb hatékonysággal tudunk reprezentálni, ha nemcsak egy reprezentációs réteget használhatunk.

Fentiek miatt régóta cél, hogy minél több rétegből álló mesterséges neurális hálókat építsenek, és hogy ezeknek a rétegei továbbá automatikusan tanulják meg a felhasznált reprezentációkat az adatok alapján (ne kelljen a súlyokat kézzel beállítani, ami ma már teljesen világos, hogy nem optimális). A sok rétegből álló hálókat mély hálóknak, ezek tanulását pedig mély tanulásnak nevezzük.

6.2. Nem felügyelt tanulás neurális hálókban

A nem felügyelt tanulás jellemzője, hogy csak az $\mathbf{x}^{(m)}$ tanítómintákat használjuk, címkék nincsenek. Célunk ekkor a jelentőséggel bíró feature-ök automatikus kinyerése, annak kihasználása mellett, hogy rengeteg címkézetlen adat áll rendelkezésre (jóval több, mint a felügyelt tanulás esetében).

A fő kérdés azonban, hogy nem felügyelt tanulás esetén mi alapján tanulhatunk egyáltalán, milyen „költséget" kellene, hogy minimalizáljunk? Itt megjegyezzük, hogy a költség ez esetben csak a bemeneti adatoktól függhet!

Az alapötlet, hogy a következő regularizáló tagot használjuk: $-log\ p(\mathbf{x}^{(m)})$. Ez azt eredményezi, hogy egy olyan modellt tanulunk meg, amely az adathalmaz elemei között a gyakrabban szerpelő (jellemzőbb) példákhoz tulajdonít relatívan magasabb valószínűséget.

Ez ugyanakkor azt is jelenti, hogy óhatatlanul *generatív tanulásról* beszélhetünk.

6.2.1. Miért generatív a nem felügyelt tanulás?

Ha visszaemlékezünk, diszkriminatív tanulás esetén amit optimalizálunk, az a $-log\ p(\mathbf{y}|\mathbf{X})$ kifejezés. Ez azt jelentette, hogy ha adott bemenetre olyan kimenetet adott a modell, amely az adathalmaz alapján nagyon valószínű, akkor annak a valószínűsége közel lesz 1-hez, aminek a mínusz logaritmusa alig haladja meg a 0-át. A neurális háló ilyenkor tehát azt tanulja meg, hogy adott bemenet esetén milyen címke valószínű.

Generatív tanulás esetén viszont amit optimalizálunk, az a $-log\ p(\mathbf{y}, \mathbf{X})$ kifejezés. Ez azt jelenti, hogy ha a modell a teljes eloszlást jól tanulja meg, akkor többször fog olyan mintákat „generálni", amelyek valószínűsége 1-hez közeli (és arányaiban kevesebbszer olyanokat, amelyeknek nem). Ezt a kifejezést tovább alakítva:

$$-log\ p(\mathbf{y}, \mathbf{X}) = -log\ p(\mathbf{y}|\mathbf{X})\ p(\mathbf{X}) = -log\ p(\mathbf{y}|\mathbf{X}) - log\ p(\mathbf{X})$$

Itt az utolsó tag olyan, mint egy regularizáló tag (a diszkriminatív esethez képest!). Ez a tag azt mondja, hogy lehetőleg olyan $\mathbf{X}$-eket kellene alapul vennünk, amik eleve gyakoriak. És ez a tag ugyanaz a tag, mint amit a nem-felügyelt tanulás költségfüggvényeként az előző oldalon javasoltunk.

Megmutatható, hogy ha egy modell helyes (az adatokat is olyan modell generálta, még ha nem is pontosan azonos paraméterekkel), akkor a generatív tanulás általánosítás szempontjából jobb lesz. Ha azonban egy modell nem helyes, az adathalmaz méretétől függ, hogy melyik módszer a jobb:

- Kevés adat: generatív jobb
- Sok adat: diszkriminatív jobb

Ezzel kapcsolatban további részletek Ng és Jordan „On Discriminative vs. Generative Classifiers" c. írásában találhatóak.

6.3. Boltzmann-gépek

Az egyszerűség kedvéért kezdetben használjunk egyetlen rejtett réteget – egy „látható" (*visible*) és egy „rejtett" (*hidden* réteggel rendelkező hálót (később megnézzük, hozzá tudunk-e adni további rétegeket). Ez alapján az első modellünk a Boltzmann-gép (BG).

A Boltzmann-gépek parametrizációja a Boltzmann-eloszlás szerint írható le:

$$p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z_\theta} e^{-\epsilon_\theta(\mathbf{x},\mathbf{h})}$$

ahol az energia függvény és a θ paraméterek:

$$\epsilon_\theta(\mathbf{x}, \mathbf{h}) = -\frac{1}{2}\mathbf{x}^T\mathbf{U}\mathbf{x} - \frac{1}{2}\mathbf{h}^T\mathbf{Y}\mathbf{h} - \mathbf{x}^T\mathbf{W}\mathbf{h} - \mathbf{b}^T\mathbf{x} - \mathbf{c}^T\mathbf{h}$$
$$\theta = \{U, Y, W, b, c\}$$

$\mathbf{x}$ és $\mathbf{h}$ vektorok pedig binárisak.

Láthatjuk, hogy a háló megadásának módja lényegében egy ún. faktoros megfogalmazásnak felel meg:

$$\begin{aligned} p(\mathbf{x}, \mathbf{h}) &= \frac{1}{Z_\theta} e^{-\epsilon_\theta(\mathbf{x},\mathbf{h})} \\ &= \frac{1}{Z} exp(\frac{1}{2}\mathbf{x}^T\mathbf{U}\mathbf{x} + \frac{1}{2}\mathbf{h}^T\mathbf{Y}\mathbf{h} + \mathbf{x}^T\mathbf{W}\mathbf{h} + \mathbf{b}^T\mathbf{x} + \mathbf{c}^T\mathbf{h}) = \\ &= \frac{1}{Z} e^{\frac{1}{2}\mathbf{x}^T\mathbf{U}\mathbf{x}} \cdot e^{\frac{1}{2}\mathbf{h}^T\mathbf{Y}\mathbf{h}} \cdot e^{\mathbf{x}^T\mathbf{W}\mathbf{h}} \cdot e^{\mathbf{b}^T\mathbf{x}} \cdot e^{\mathbf{c}^T\mathbf{h}} \end{aligned}$$

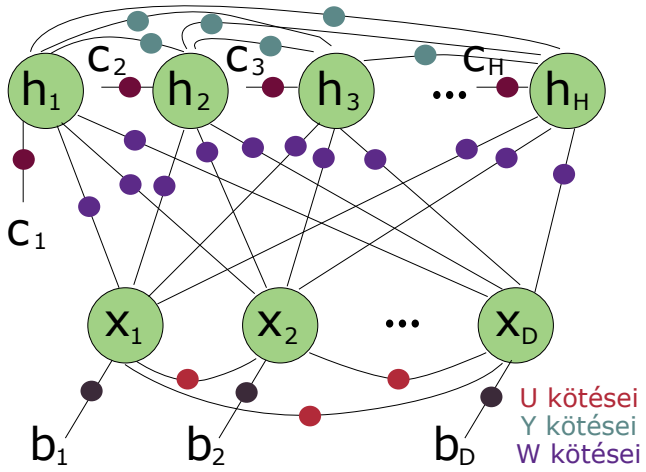
A faktor-gráf a 6.1. ábrán látható.

Mivel a kitevőben negatív energia szerepel, ezért minél kisebb az energia (akár pozitív, akár negatív: minél kisebb), annál valószínűbb a konfiguráció. Az energia a változók és a paraméterek illeszkedésétől függ. partíciófüggvény pedig:

$$Z_\theta = \sum_{x_1=0}^{x_1=1} \cdots \sum_{x_D=0}^{x_D=1} \sum_{h_1=0}^{h_1=1} \cdots \sum_{h_H=0}^{h_H=1} e^{-\epsilon_\theta(\mathbf{x},\mathbf{h})}$$

Ezt a Z_θ partíciófüggvényt sajnos nagyon költséges kiszámítani: az összes lehetséges érték-kombinációra ki kell számítani a háló energiáját ($p(\mathbf{x}, \mathbf{h})$-k összege ugyanis az összes lehetséges értékkombinációra kell, hogy 1 legyen).

Példaként tekintsük a 6.2. ábrán látható Boltzmann-gépet! Mivel a Boltzmann-gép általában véve is egy irányítatlan gráf, az élek súlyai korrelációt fejeznek ki. Ennek a Boltzmann-gépnek az állapotvalószínűségei:



6.1. ábra. Boltzmann-gépek parametrizációja.

$$
\begin{aligned}
p(0,0,0) &= (1/Z)e^{-(-0-0-0-0-0)} = 1/Z \approx 0.12 \\
p(0,0,1) &= (1/Z)e^{-(-0-0-0-0.2-0)} \approx 1.22/Z \approx 0.14 \\
p(0,1,0) &= (1/Z)e^{-(-0-0-0-0-0)} = 1/Z \approx 0.12 \\
p(0,1,1) &= (1/Z)e^{-(-0-0-0-0.2-0)} \approx 1.22/Z \approx 0.14
\end{aligned}
$$

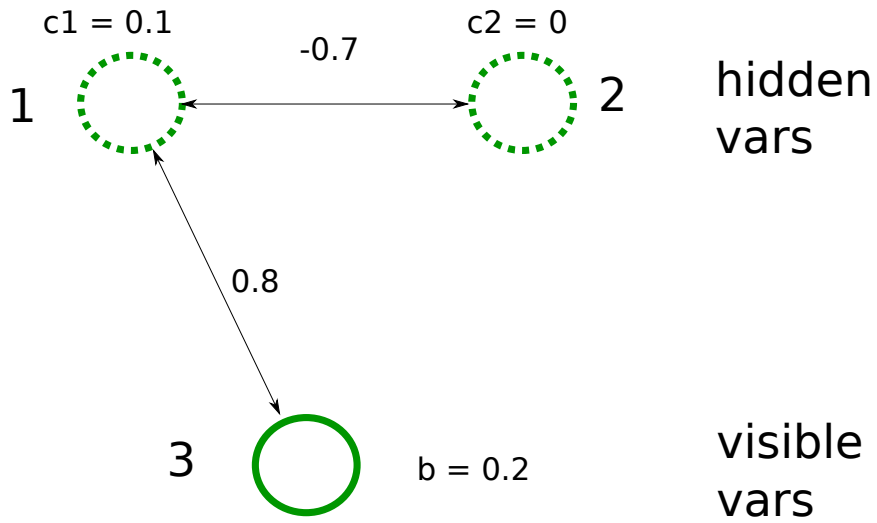
Továbbá...

$$
\begin{aligned}
p(1,0,0) &= (1/Z)e^{-(-0-0-0-0-0.1)} \approx 1.1/Z \approx 0.13 \\
p(1,0,1) &= (1/Z)e^{-(-0-0-0.8-0.2-0.1)} \approx 3/Z \approx 0.35 \\
p(1,1,0) &= (1/Z)e^{-(-0+0.7/2-0-0-0.1)} \approx 0.03/Z \approx 0 \\
p(1,1,1) &= (1/Z)e^{-(-0+0.7/2-0.8-0.2-0.1)} \approx 0.09/Z \approx 0.01
\end{aligned}
$$

6.3.1. Mintavételezés Boltzmann-gépből

Hogyan mintavételezhetünk általánosságban egy tetszőleges változó értékéből (a többi ismeretében)? Vegyük észre, hogy az i. sejt aktivációja csak egy bias tényezőtől és a bemenetek súlyozott összegétől függ (ezt eddig is tudtuk a faktor-gráfból)...a kimeneti eloszlás pedig definíció szerint legyen az aktiváció logisztikus függvénye szerinti:

$$
\begin{aligned}
a_i &= b_i + \sum_j s_j w_{ji} \\
p(s_i = 1) &= \frac{1}{1 + e^{-a_i}}
\end{aligned}
$$



6.2. ábra. Példa egyszerű Boltzmann-gépre.

Ez általánosan működik a látott és rejtett változók esetén is. Pl. rejtett változó esetén:

$$p(h_i = 1|\mathbf{x}, \mathbf{h}_{\setminus i}) = \frac{exp(\ldots)/Z}{\sum\limits_{h'_i=0..1} exp(\ldots)/Z} =$$

$$= \frac{\overbrace{exp(\mathbf{x}^T\mathbf{W}_{col-i} + 0.5 \cdot \mathbf{Y}_{row-i}\mathbf{h} + c_i)}^{a} \cdot k}{k \cdot \sum\limits_{h'_i=0..1} exp(\mathbf{x}^T\mathbf{W}_{col-i}h'_i + 0.5 \cdot h'_i\mathbf{Y}_{row-i}\mathbf{h} + h'_i c_i)} =$$

$$= \frac{ak}{k(1+a)} = \frac{1}{1+a^{-1}}$$

A levezetésben a éppen a változó súlyozott aktivitását jelöli (k pedig az attól független részt). Éppen a szigmoid-függvény jön ki!

6.3.2. Boltzmann-gépek tanítása

Tanuláshoz szükség lesz az ún. posterior eloszlásból – azaz $p(\mathbf{h}|\mathbf{x}^{(m)})$-ből – történő mintavételezésre

Ha ugyanis $p(\mathbf{h}|\mathbf{x})$-ből tudnék mintát venni, abból le tudnék generálni egy olyan $\mathbf{x}'$-t, mely esetén az $\epsilon(\mathbf{x}', \mathbf{h})$ energia kisebb lenne (ehhez elég a szigmoid függvényt használni a rejtett vektor ismeretében). Ezt követően a delta-szabály segítségével frissíteni tudnám a súlyokat olyan módon, hogy ezt az energia-különbséget csökkentsem... vagyis hogy $\mathbf{x}'$ legközelebb jobban hasonlítson x-re!

Az h_j rejtett változót és az i-edik bemenetet összekötő súlyra:

$$-\frac{\partial}{\partial w_{ji}}(\epsilon(\mathbf{x},\mathbf{h})-\epsilon(\mathbf{x}',\mathbf{h}))^2 = -2\underbrace{[\epsilon(\mathbf{x},\mathbf{h})-\epsilon(\mathbf{x}',\mathbf{h})]}_{k}\frac{\partial}{\partial w_{ji}}k =$$

$$= -2k\frac{\partial}{\partial w_{ji}}(-\mathbf{x}^T\mathbf{W}\mathbf{h}+\mathbf{x}'^T\mathbf{W}\mathbf{h}) = -2k\cdot(x_i'-x_i)h_j$$

$$\Rightarrow w_{ji} = w_{ji}+\gamma\cdot h_j(x_i-x_i')$$

Mivel a modell nem irányított, ezért egymástól függetlenül mintavételezhetünk a posterior eloszlás különböző változóiból a látható változók ismeretében (megj.: nem lesz a „kimagyarázással" probléma).

Összefüggések persze továbbra is lehetnek h_i változók között is (még ha nem is ok-okozatiak), ezért h_i értéke nemcsak $\mathbf{x}$-től, hanem a többi h_j-től is függ:

$$p(h_i|\mathbf{x}) = \sum_{h_1=0}^{h_1=1}\cdots\sum_{h_{i-1}=0}^{h_{i-1}=1}\sum_{h_{i+1}=0}^{h_{i+1}=1}\cdots\sum_{h_H=0}^{h_H=1}p(h_i|\mathbf{x},\mathbf{h}_{\setminus i})$$

Ezt azonban sajnos megintcsak igen bonyolult lesz kiszámítani... Az összegzendő tagok száma exponenciálisan függ a rejtett változók számától! Ezért inkább vegyünk egy újabb, némiképp módosított modellt – a Korlátozott Boltzmann-gép modellt!

6.4. Korlátozott Boltzmann-gépek (RBM-ek)

A klasszikus Boltzmann-égpek esetében alapvető probléma, hogy a rejtett réteg változói között lehetnek direkt kötések is, ezért nem elég hogy adott rejtett változóval összefüggnek a látható paraméterek: a rejtett változók egymással is összefüggnek!

Ezért célszerű az ún. *korlátozott Boltzmann-gépet* használni (Restricted Boltzmann Machine, RBM) a Boltzmann-gép helyett (ha egyáltalán ilyen modellel dolgozunk). Az RBM páros gráf, és hasonló a klasszikus Boltzmann-géphez, azzal a különbséggel, hogy az energiafüggvényben $U = Y = \underline{0}$.

Mivel a rejtett változók között nincsenek kötések, a feltételes valószínűségek számítása sokkal egyszerűbb:

$$p(\mathbf{h}|\mathbf{x}) = p(\mathbf{x},\mathbf{h})/\sum_{\mathbf{h}'}p(\mathbf{x},\mathbf{h}')$$

$$= \frac{exp(\mathbf{x}^T\mathbf{W}\mathbf{h}+\mathbf{b}^T\mathbf{x}+\mathbf{c}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^H}exp(\mathbf{x}^T\mathbf{W}\mathbf{h}'+\mathbf{b}^T\mathbf{x}+\mathbf{c}^T\mathbf{h}')/Z}$$

A kitevőkben ha összeg van, az felbontható a tagok szorzatára. Utána, mivel $\mathbf{b}^T\mathbf{x}$ nem függ attól, ami felett összegzünk, ezért kiemelhető. Z-vel is egyszerűsíthetünk:

$$p(\mathbf{h}|\mathbf{x}) = \frac{exp(\mathbf{x}^T\mathbf{W}\mathbf{h})exp(\mathbf{c}^T\mathbf{h})}{\sum\limits_{\mathbf{h}'\in\{0,1\}^H} exp(\mathbf{x}^T\mathbf{W}\mathbf{h}' + \mathbf{c}^T\mathbf{h}')} =$$

$$= \frac{exp(\mathbf{x}^T\mathbf{W}\mathbf{h})exp(\mathbf{c}^T\mathbf{h})}{\sum\limits_{h'_1\in\{0,1\}} \cdots \sum\limits_{h'_H\in\{0,1\}} \prod\limits_j exp(c_j h'_j + \sum_i x_i w_{ij} h'_j)}$$

Továbbá vegyük észre, hogy mindegyik összegzéstől csak az egyik exponenciális tag függ (pl. csak $j = H$ esetén függ a szorzattényező az utolsó összegzéstől), így a többi sorra kiemelhető:

$$p(\mathbf{h}|\mathbf{x}) = \frac{exp(\mathbf{x}^T\mathbf{W}\mathbf{h})exp(\mathbf{c}^T\mathbf{h})}{\left(\sum\limits_{h'_1\in\{0,1\}} exp(c_1 h'_1 + \sum_i x_i w_{i1} h'_1)\right) \cdots \left(\sum\limits_{h'_H\in\{0,1\}} exp(\ldots)\right)} =$$

$$= \frac{exp(\mathbf{x}^T\mathbf{W}\mathbf{h})exp(\mathbf{c}^T\mathbf{h})}{\prod\limits_j \sum\limits_{h'_j\in\{0,1\}} exp(c_j h'_j + \sum_i x_i w_{ij} h'_j)} =$$

$$= \frac{exp(\mathbf{x}^T\mathbf{W}\mathbf{h})exp(\mathbf{c}^T\mathbf{h})}{\prod\limits_j (1 + exp(c_j + \sum_i x_i w_{ij}))} = \prod_j \frac{\overbrace{exp((\mathbf{x}^T\mathbf{W})_j h_j + c_j h_j)}^{\text{mint nevezőben csak nem összegzünk } h_j\text{-re}}}{1 + exp((\mathbf{x}^T\mathbf{W})_j + c_j)}$$

A szorzat j-edik tényezője éppen azt a súlyozott összeget tartalmazza, amelyik a j-edik rejtett sejt aktivációja (plusz a bias tényező) – amit az előbb is megmutattunk hogy egy változó valószínűsége.

Kis átalakítással az is látszik, hogy ez a valószínűség egy szigmoid-függvény (ezt mondtuk az előbb is), és hogy könnyen tudunk a posteriorból mintavételezni:

$$(h_j = 1|\mathbf{x}) = \frac{exp(\mathbf{x}^T\mathbf{W})_j + c'_j)}{1 + exp(\mathbf{x}^T\mathbf{W})_j + c_j)} = sgm\left(\sum_i x_i w_{ij} + c_j\right)$$

Vagyis: a szorzat tényezői éppen $p(h_j = 1|\mathbf{x})$ tagok. Innen látszik, hogy az egyes rejtett paraméterek tényleg függetlenek egymástól:

$$p(\mathbf{h}|\mathbf{x}) = \prod_j p(h_j|\mathbf{x})$$

6.4.1. RBM-ek tanítása

Hogyan taníthatunk egy RBM-et? Azt mondtuk, hogy a látott vektorok valószínűségét szeretnénk megtanulni. Ezért, ha valamit sokat látunk, úgy kell beállítani a modell paramétereit, hogy a rejtett változók inkább azt a vektort generálják le (hogy együttes energia kicsi legyen).

Egy kézenfekvő ötlet, hogy mivel van sok tanítómintánk, azt szeretnénk ha a tanítóminták valószínűségeinek szorzata nagy lenne. Egy tanítóminta valószínűsége azonban függ az aktuálisan aktív rejtett paraméterektől is (hiszen a háló probabilisztikus, és h vektort is mintavételezéssel kapjuk):

$$p(\mathbf{x}^{(m)}) = \sum_{\mathbf{h}'} p(\mathbf{x}^{(m)}, \mathbf{h}') = \sum_{\mathbf{h}'} \frac{e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')}}{Z}$$

Amit keresünk:

$$\arg\max_{\theta} log \prod_{m=1}^{M} p(\mathbf{x}^{(m)})$$

Mivel szorzat logaritmusa megegyezik logaritmusok összegével, ez egyenértékű az alábbival:

$$\arg\max_{\theta} \sum_{m=1}^{M} log\ p(\mathbf{x}^{(m)}) =$$
$$\arg\max_{\theta} \sum_{m=1}^{M} \left(log \sum_{\mathbf{h}'} \frac{e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')}}{Z} \right) =$$

(Z nem fugg h-tol, ezert kiemelheto)

$$\arg\max_{\theta} \sum_{m=1}^{M} \left(log\ \frac{1}{Z} \sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')} \right) =$$

Ahhoz, hogy ezt maximalizáljuk, elég tagonként maximalizálni (vagy: a mínusz egyszeresüket minimalizálni). Ehhez: keressük meg egy mínusz egyszeres tagnak adott w_{ij} súly szerinti deriváltját! A végén a deriváltak átlaga szerint módosíthatunk.

$$\frac{\partial - log\ p(\mathbf{x}^{(m)})}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} - \left(log \sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')} - log\ Z \right)$$

$$= \frac{\partial}{\partial w_{ij}} \left[-log \sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')} + log\ Z \right]$$

(log(x) derivaltja 1/x, aztan a belso fv-eket (x) is derivalni kell)

(exp(x) fv derivaltja exp(x), aztan a belso fv-t (x) is derivalni kell)

$$= -\frac{1}{\sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')}} \frac{\partial}{\partial w_{ij}} \sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')} + \frac{1}{Z} \frac{\partial}{\partial w_{ij}} Z$$

(jelen esetben $-\epsilon(\mathbf{x}, \mathbf{h}')$ derivaltja $x_i h'_j$, hiszen ez lesz w_{ij} szorzotenyezoje az energiafv-ben)

$$= -\frac{1}{\sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')}} \sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')} x_i^{(m)} h'_j + \frac{1}{Z} \frac{\partial}{\partial w_{ij}} Z$$

$$= -\frac{1}{Z \cdot p(\mathbf{x}^{(m)})} \sum_{\mathbf{h}'} e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')} x_i^{(m)} h'_j + \frac{1}{Z} \frac{\partial}{\partial w_{ij}} Z$$

$$= -\sum_{\mathbf{h}'} \frac{e^{-\epsilon(\mathbf{x}^{(m)}, \mathbf{h}')}}{Z \cdot p(\mathbf{x}^{(m)})} x_i^{(m)} h'_j + \frac{1}{Z} \frac{\partial}{\partial w_{ij}} \sum_{\mathbf{x}, \mathbf{h}'} e^{-\epsilon(\mathbf{x}, \mathbf{h}')}$$

$$= -\sum_{\mathbf{h}'} \frac{p(\mathbf{x}^{(m)}, \mathbf{h}')}{p(\mathbf{x}^{(m)})} x_i^{(m)} h'_j + \frac{1}{Z} \sum_{\mathbf{x}, \mathbf{h}'} e^{-\epsilon(\mathbf{x}, \mathbf{h}')} x_i h'_j$$

$$= -\sum_{\mathbf{h}'} p(\mathbf{h}' | \mathbf{x}^{(m)}) x_i^{(t)} h'_j + \sum_{\mathbf{x}, \mathbf{h}'} \underbrace{\frac{e^{-\epsilon(\mathbf{x}, \mathbf{h}')}}{Z}}_{p(\mathbf{x}, \mathbf{h}')} x_i h'_j$$

$$= -E_{\mathbf{h}} \left[x_i^{(t)} h_j | \mathbf{x}^{(m)} \right] + E_{\mathbf{x}, \mathbf{h}} \left[x_i h_j \right]$$

Ennek a $-\gamma$-szorosát adjuk hozzá w_{ij}-hez.

A képlet első fele azt jelenti, hogy adott x_i és h_j közötti kötés egyrészt erősödik, ha ezek a változók (átlagban, az M darab tanítómintára nézve) egyszerre aktívak a tanítómintákban és a tanítómintákhoz tartozó leginkább valószínű rejtett vektorokban.

Másrészt ugyanezen kötés gyengül akkor, ha adott x_i és h_j gyakran egyszerre aktív azokban az esetekben is, amikor nemcsak a tanítómintákra (hanem az összes lehetséges $\mathbf{x}, \mathbf{h}$ fölött) nézzük.

Ez utóbbi azt jelenti, hogy ha adott x_i és h_j minden zagyvaságra (nemcsak a tanítómintákra) egyszerre aktív, akkor a közöttük levő kötés nem igazán alkalmas arra, hogy jellegzetes minták tanulására használjuk → inkább gyengíteni kell a kötést.

Ugyanez általánosan:

$$\Delta\theta = E_{\mathbf{h}} \left[\frac{\partial E(\mathbf{x}^{(m)}, \mathbf{h})}{\partial \theta} | \mathbf{x}^{(m)} \right] - E_{\mathbf{x}, \mathbf{h}} \left[\frac{\partial E(\mathbf{x}^{(m)}, \mathbf{h})}{\partial \theta} \right]$$

Az első tag az ún. **pozitív fázis**, és viszonylag könnyű számítani. A második tag az ún. **negatív fázis**, amit sajnos nehéz számítani, mert az összes lehetséges bemenet fölött kell integrálni / szummázni. Ezért approximálni szokás.

Például a pozitív fázis a $\mathbf{W}$ mátrixra:

$$\begin{aligned} E_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(m)},\mathbf{h})}{\partial w_{jk}}|\mathbf{x}^{(m)}\right] &= E_{\mathbf{h}}\left[-h_j x_k^{(m)}|\mathbf{x}^{(m)}\right] = \\ &= \sum_{h_j\in\{0,1\}} -h_j x_k^{(m)} p(h_j|\mathbf{x}^{(m)}) = \\ &= -x_k^{(m)} p(h_j=1|\mathbf{x}^{(m)}) = \\ &= -x_k^{(m)} \cdot sgm(b_i + \mathbf{W}_j \mathbf{x}^{(m)}) \end{aligned}$$

A negatív fázis kellemetlenebb: ahhoz, hogy $p(\mathbf{x},\mathbf{h})$ valószínűségeket kiszámítsuk, szükségünk van Z ismeretére is, ami iszonyatosan sok számítást igényel minden egyes frissítéskor!

Geoff Hinton ötlete 2002-ben, hogy a fenti nehézségek miatt a teljes eloszlás figyelembe vétele helyett vegyünk egy approximált vektorpárt. A képlet első tagját reprezentálja $\mathbf{x}^{(m)}$ és egyetlen, hozzá tartozó $\mathbf{h}$ vektor. A képlet második tagját pedig reprezentálja egyetlen általános $\tilde{\mathbf{x}}$ és $\tilde{\mathbf{h}}$ vektor, amik nem azt tükrözik, hogy a háló jelenlegi állapotában általában „miben hisz", hanem hogy „miben hisz jobban" (mint a tanító-vektorpár).

Ez azt jelenti, hogy a negatív fázisban nem a bemeneti és rejtett vektor különböző értékeinek együttes valószínűségei felett összegzünk, hanem egyetlen egy „átlagos" vektorpárt használunk, amit 1 valószínűségűnek tekintünk. Az eredmények azt mutatják, hogy a gyakorlatban ez is jól működik!

Az approximatív vektorpárt pedig az ún. **Markov Chain Monte Carlo** sampling segítségével kaphatjuk meg. Elvben végtelen hosszú ideig (gyakorlatban: stabil állapotig, vagy még addig sem) felváltva veszünk mintákat a látható és a rejtett rétegből. Ezt az elgondolást ***kontrasztív divergenciának*** nevezzük (ld. 6.3. ábra).

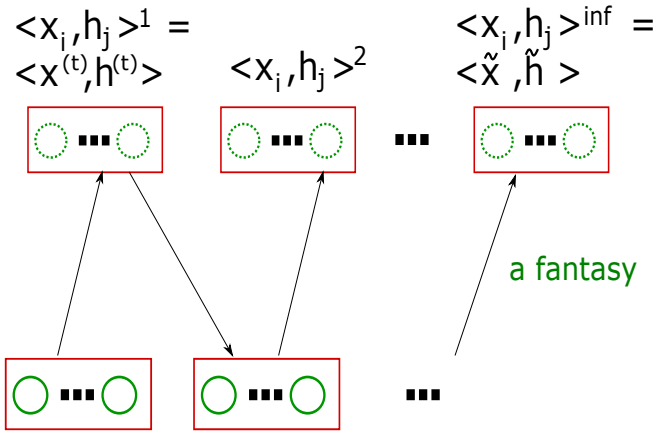
Ha nem mindig előről kezdjük az iterálást, hanem a legutóbbi approximált vektortól, akkor a módszert **perzisztens kontrasztív divergenciának** nevezzük. Ez általában jobb, ha nem csak köztes reprezentációt keresünk, hanem osztályozni is szeretnénk.

Ehhez a tanítópontok (átlagának) helyén csökkentjük a modell energiáját, az általánosnak vélt helyen viszont növeljük (ld. 6.4. ábra). A frissítés pedig:

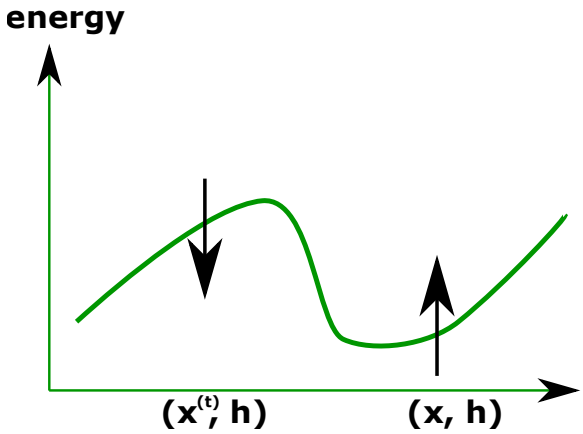
$$\Delta w_{jk} = x_k^1 h_j^1 - x_k^\infty h_j^\infty$$

Mivel ez túl időigényes („végtelen" iteráció) a gyakorlati hasznosításhoz, jó kompromisszum, ha a láncot csak fix hosszon futtatjuk. Speciális eset, amikor csak egyetlen lépést teszünk meg (ha csak reprezentációt keresünk további tanuláshoz, és nem mindjárt tanulni szeretnénk, ez is jó empirikus eredményeket ad). A frissítés ekkor:

$$\Delta w_{jk} = x_k^1 h_j^1 - x_k^2 h_j^2$$



6.3. ábra. Markov-Chain Monte Carlo mintavételezés a kontrasztív divergenciához.



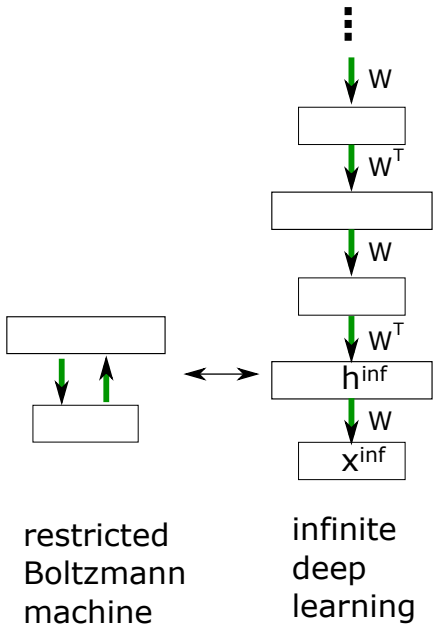
6.4. ábra. Az ábra a modell energia-konfigurációjának változását mutatja be konceptuálisan kontrasztív divergencia alapú tanulás esetén.

Mindennek van egy jópofa (informális) szemléltetése. Abból indulunk ki (ld. 6.5 ábra), hogy az RBM tulajdonképpen ekvivalens egy végtelen hosszú, irányított modellel. Az *explaining away* problémája itt azért nem gond, mert ha x^∞ megfigyelése antikorrelálja is h^∞ változóit, azok egy szinttel feljebb közös okkal rendelkeznek, ami viszont korrelálja őket (a két hatás remélhetőleg kioltja egymást!). Ha most a végtelen hosszú hálóban szeretném megtanulni w_{ij} súlyokat, erre a delta-szabály szerint a preszinaptikus aktivitást kell megszorozni a kívánt és a kapott poszt-szintaptikus aktivitás különbségével.

$$\Delta w_{ij} \propto h_j(x_i - \hat{x}_i)$$

A 0. szinttől lefelé indulunk el. x^1 szinten x_i^0-át szeretnénk visszakapni, de ez nem mindig sikerül, mert h-kat csak mintavételezni tudjuk (az egész háló probabilisztikus).

$$\Delta w_{ij} \propto h_j^0(x_i^0 - x_i^1)$$



6.5. ábra. Az RBM kiterített változata egyfajta „végtelen” mélytanulásnak feleltethető meg.

Másrészt a W súlyok újra és újra előfordulnak, vagyis lentebbi szinteken is korrigálni kell őket a hibák deriváltjával. A végleges korrekció összegszerű:

$$\Delta w_{ij} \propto h_j^0(x_i^0 - x_i^1) + \\ x_i^1(h_j^0 - h_j^1) + \cdots + h_j^{\infty-1}(x_i^{\infty-1} - x_i^\infty)$$

A köztes tagok ebből mind kiesnek! Végül ezt kapjuk:

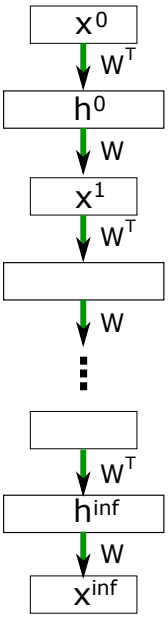
$$\Delta w_{ij} \propto x_i^0 h_j^0 - x_i^\infty h_j^\infty$$

Ez ugyanaz, mint amit korábban írtunk! A fix lépéses változat ezzel szemben azt feltételezi, hogy a lánc idővel stabilizálódik, és a deriváltak közel nullák lesznek. Ez valóban nem irracionális feltételezés, és akár kiindulópontja lehet egy adaptív eljárásnak is. Egyébként mint mondtuk, a (perzisztens) kontrasztív divergencia empirikusan nagyon jó. Ezekből a megfontolásokból kiindulva jöhet majd a következő lépés: többrétegű RBM tanítása (minél több réteg, annál jobb reprezentáció!).

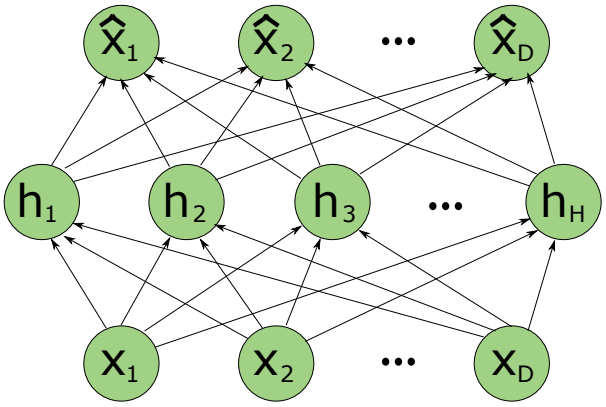
6.5. Autoenkóderek

Az autoenkóderek alternatív módszert jelentenek köztes reprezentáció elérésére.

Az autoenkóderek feed-forward hálók, amelyeket osztálycímke helyett a bemenetük reprodukálására tanítunk (ld. 6.7. ábra).



6.6. ábra. Az RBM tanítása: miért működik a kontrasztív divergencia?



6.7. ábra. Autoenkóder felépítése

Gyakori választás az aktivációkat illetően:

$$\mathbf{h}(\mathbf{x}) = g(a(\mathbf{x})) = sgm(\mathbf{b} + \mathbf{W}\mathbf{x})$$
$$\hat{\mathbf{x}}(\mathbf{h}) = o(\hat{a}(\mathbf{x})) = sgm(\mathbf{c} + \mathbf{W}^*\mathbf{x}) = sgm(\mathbf{c} + \mathbf{W}^T\mathbf{x})$$

Mivel mindkét rétegben azonos súlymátrix van, tanításkor minden gradiens két gradiens összegéből fog állni. Egy lehetséges költségfüggvény a *keresztentrópia*:

$$l(\underbrace{f(\mathbf{x})}_{\hat{\mathbf{x}}}, \mathbf{x}) = -\sum_{d=1}^{D} x_d \; log(\hat{x}_d) + (1 - x_d) \; log(1 - \hat{x}_d) \tag{6.1}$$

Ha $x_d = 1$, akkor az adott komponens $-log(\hat{x}_d)$ lenne. Ha ezt minimalizáljuk, akkor $log(\hat{x}_d)$-t maximalizálnánk, vagyis a kimenetet 1 felé löknénk. Ellenkező esetben fordítva, 0 felé löknénk.

Ha a kimenet egyenlő a bemenettel, akkor lesz minimális a költségfüggvény. Akkor is működik, ha nemcsak bináris bemenetek vannak, viszont ekkor a minimum nem 0-nál lesz.

Tanításkor használhatjuk a korábbi gradiens-módszert. Ellenben az is látszik a jelenlegi hálón, hogy túl sok érdekes dolgot nem fog csinálni. Ugyanis:

- Ha a köztes réteg kisebb, mint a bemeneti ($H < D$), a háló alulvezérelt (undercomplete), és tömöríteni fog... viszont csak a tanítómintákra tudja ezt megtenni (általánosítani nem fog, és random mintákra is rossz lesz)
- Ha a köztes réteg nagyobb, mint a bemeneti ($H > D$), a háló túlvezérelt (overcomplete). Ilyenkor nincs szükség kompresszálásra. Azt is megteheti, hogy a bemenetet az első D rejtett változóba bemásolja. De nincs garancia arra, hogy a köztes réteg bármit elárul a bemenetek struktúrájáról – pedig az eredeti cél éppen ez lenne!

6.5.1. Denoising autoencoder

Egy lehetséges megoldás az ún. ***denoising autoencoder*** használata. Az alapötlet, hogy olyan reprezentációt keresünk, amely a zajra érzéketlen. A bemeneti réteg és a rejtett réteg közé beteszünk egy szintén D-dimenziójú réteget, amit $\tilde{\mathbf{x}}$-szel jelölünk, és ami egy kis zajt ad hozzá az eredeti bemenethez. Ekkor a rekonstrukció $\tilde{\mathbf{x}}$ szerint történik, de a költséget az eredeti bemenethez képest nézzük!

Ebben az esetben az autoenkóder már nem „csalhat" úgy, hogy lemásolja a bemenetét.

6.5.2. Contractive autoencoder

Egy másik megoldást adhat a ***contractive autoencoder***. Ebben az alapötlet, hogy a költségfüggvényhez hozzá adunk egy büntetőtényezőt, amivel az érdektelen megoldásokat el tudjuk kerülni. Például, ha azt szeretnénk, hogy az autoencoder csak a tanítóminták variabilitását képezze le, más variabilitást nem:

$$l(\hat{\mathbf{x}}, \mathbf{x}) = -\sum_{d=1}^{D} x_d \; log(\hat{x}_d) + (1 - x_d) \; log(1 - \hat{x}_d) + \\ + \lambda \sum_j \sum_k \left(\frac{\partial \mathbf{h}(\mathbf{x}^{(\mathbf{m})})_{\mathbf{j}}}{\partial \mathbf{x}_k^{(m)}} \right)^2$$

Ha a parciális derivált 0, a bemenetből mint nyers információból biztos, hogy semmit se tartunk meg – mivel ez azt jelenti, hogy a j-edik rejtett változó nem változik meg a k-adik bemenet kicsi módosításával.

6.5.3. Denoising és contractive autoencoderek értékelése

A gyakorlatban a zajtalanító és kontraktáló autoenkóderek is jól működnek.

A denoising autoencoder a következő szempontok szerint értékelhető:

- Pozitívum, hogy csak pár új sort kell az eredeti programkódhoz hozzáadni
- Pozitívum, hogy nem kell a köztes réteg bemeneti réteg szerinti, páronkénti deriváljtait (Jacobi-mátrix) kiszámítani
- Pozitívum, hogy nem kell a Jacobi-mátrix szerint gradiens-csökkentést csinálni

A contractive autoencoder előnye ezzel szemben, hogy determinisztikus gradienssel tanítható (mivel nincsen zajréteg), ezért több lehetőség van a párhuzamosításra / optimalizációra.

6.6. RBM és Autoencoder alapú mély tanulás

Az előzőekben számos példát láthattunk feed-forward neurális hálóra. A gyakorlatban sokáig az volt megfigyelhető, hogy a legritkábban „nyertek" a kutatók azzal, amikor több mint 1 rejtett réteg volt és a backpropagationt használták. Ez meglepő volt, hiszen az ellenkezőjét várták volna sokan!

A mély hálók tanításának nehézségével kapcsolatban két fő hipotézis merült fel a szakirodalomban:

- Ezek közül az egyik hipotézis, hogy „alultanulunk": sok réteg használata mellett a kimenet felől visszaküldött gradiens „eltűnik" (vanishing gradient problem). Különösen amikor szigmoid függvényeket használunk a kimeneteken, a szaturáló sejtek kimeneteinek bemenetük szerinti deriváltja közel 0. Mindeközben mégis cél, hogy bizonyos sejtek szaturáljanak, hogy az aktivációs függvények nemlineáris részei is kihatással legyenek a múködésre.
- A másik hipotézis, hogy éppen ellenkezőleg, „túltanulunk", mivel minél mélyebb egy háló, annál több paramétere van és annál komplexebb függvény-osztályt ír le. Ez a magas variancia és alacsony bias tipikus esete.

A fentiek miatt két lehetséges regularizációs módszer terjedt el a mély tanulás korai fázisaiban.

6.6.1. Nem felügyelt előtanítás

Ezek egyike a *nem felügyelt előtanítás* (unsupervised pre-training). Ennek alapötlete, hogy inicializáljuk a rejtett változók paramétereit nem felügyelt tanulással. Ezzel kényszerítjük a hálót, hogy az állapottér olyan szegmensébe kerüljön, ahol remélhetőleg olyan lokális minimumok vannak, amik nem

tanulnak túl. Mindeközben arra is kényszerítjük a hálót, hogy látens struktúrát (is) reprezentáljon (ne csak a kimeneteket tanulja meg) – például azt, hogy egy adott kép miért egy írott karakter, és nem egy random mintavétel.

Az egyik legnépszerűbb módszer: *greedy layer-wise pre-training* (mohó, rétegenkénti előtanítás). A módszer négy alapelve:

1. Rétegenként vesszük és tanítjuk meg az újabb és újabb rejtett rétegeket, valamilyen nem felügyelt kritérium szerint
2. Minden újabb réteg hozzáadásakor lekötjük az előzőleg tanított rejtett rétegek paramétereit, és csak a legújabbat módosítjuk.
3. Az eggyel korábbi réteget mindig az aktuális réteg „feature-extraktorának" tekintjük
4. Utolsó lépés: finomhangolás. Hozzáadunk egy kimeneti réteget, és az egész hálót backpropagationnel betanítjuk.

Ekkor a rétegek jelentése:

- **Első réteg:** rejtett feature-ök, amik gyakoribbak a tanítóbemenetekben, mint a random bemenetekben
- **Második réteg:** rejtett feature-ök kombinációi, amik gyakoribbak, mint a random rejtett feature-ök
- **Harmadik réteg:** ... feature-ök kombinációinak kombinációi ...
- Finomhangoláskor már alig lesznek változások, csak néhány főbb helyen.

Ha rengeteg jelöletlen, címkézetlen adatunk van, még jól is járunk mert kihasználjuk ezek rendelkezésre állását.

6.6.2. Példa: többszintes RBM

A 6.8. ábra szerint „fagyasszuk" be a legalul megtanított RBM súlyait, és ehhez vegyünk hozzá egy újabb RBM-et.

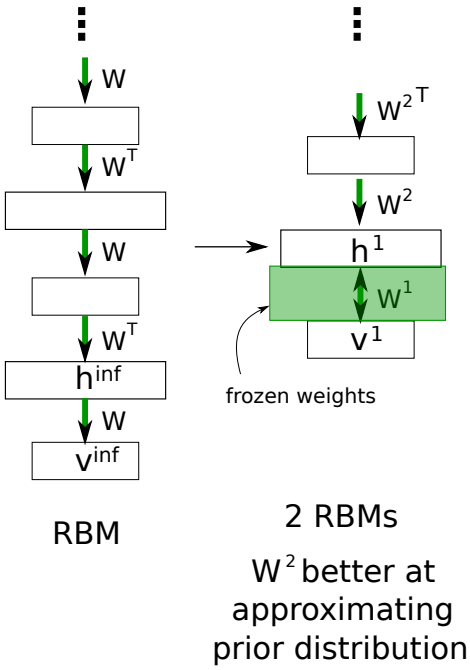
Ennek a második RBM-nek az lesz a feladata, hogy h változókat jobban le tudja írni, mint a korábbi végtelen háló.

A szabad paraméterek számát minden fázis után megnöveljük. Mivel az eredeti mély háló végtelen mélységű, a maradék is az lesz, csak legalul véges számú súlymátrixot lefixálunk.

Az eredmény: W^2 jobban tudja modellezni h^1 régetet, mint amennyire W^1 tudta volna (hiszen W^1 feladata inkább x^1 modellezése volt!)

Erre az alapelvre többféle módszer épült a 2000-es és 2010-es években:

- Stacked RBMs: Hinton, Teh, Osindero (2006)



6.8. ábra. Unsupervised greedy layer-wise pre-training

  - Hinton: „To recognize shapes, first learn to generate images”

- Stacked Autoencoders: Bengio, Lamblin, Popovici, Larochelle (2007)
- Általánosítás ritka autoenkóderekre: Ranzato, Poultney, Chopra, LeCun (2007)
- . . . és még rengeteg

Hugo Larochelle példája (youtube sorozat, 7-es rész 4. video) megmutatja, hogy az előtanítás olyan, mint a regularizálás: nagy kapacitás esetén kevésbé tanul túl, kis kapacitás esetén pedig alultanulást eredményez

6.6.3. Dropout training

E másik módszer a regularizációra az ún. dropout training („törléses tanulás”). Az alap ötlet: a finomhangolási fázisban minden tanítómintánál „csonkítsuk meg” a hálót – átlagban a rejtett sejtek felét töröljük (állítsuk a kimenetüket 0-ra).

Az ilyenkor lejátszódó folyamat kicsit olyan, mint a zajtalanító autoenkódereknél, csak itt a rejtett és nem pedig a bemeneti vagy kimeneti rétegekre teszünk zajt. Az eredmény az, hogy nem hagyjuk a rejtett rétegbeli sejteket egymáshoz adaptálódni (less co-adaptation). Ezzel elérjük, hogy a rejtett sejtek azokra a feature-ökre fókuszálnak, amelyek általánosságban jók (akkor is, ha egy szomszédjuk éppen átmenetileg hiányzik.

Mindez egy kicsit megváltoztatja a backpropagation analitikus képleteit, de nem számottevően (csak figyelembe kell venni a maszkot).

6.7. Kitekintés a napjainkra vonatkozóan

Napjainkra sok minden megváltozott a mély tanulás elméletével kapcsolatban: a leghatékonyabb modellek már nem a stacked RBM-ek illetve autoencoder-ek, és a backpropagation mély hálóknál is rendkívül hatékonyan működik.

A fejlődés mögött számos intuitív módon kikísérletezett új elméleti konstrukció és mérnöki bravúr áll. Fontos szerephez jutottak az utóbbi években például a reziduális hálók (melynek alapmechanizmusai az ún. skip connection-ök – az olyan kötések, amelyek bizonyos köztes régetegeket „átugornak" és a kimenetükhöz hozzáadódnak), illetve a transformer architektúrában is jelen lévő figyelmi mechanizmusok.

Ezen eredmények összefoglalása egy újabb fejezetet igényelne, megismerésük azonban mindenképpen javasolt.

Irodalomjegyzék

[1] Michael Bowles. *Machine learning in Python: essential techniques for predictive analysis*. John Wiley & Sons, 2015.

[2] Pedro Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015.

[3] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.

[4] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[5] W.S. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[6] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[7] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65:386–408, 1958.

[8] D. Vernon, G. Metta, and G. Sandini. A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents. *IEEE Transactions on Evolutionary Computation*, 11(2):151–180, 2007.

[9] B. Widrow and M.E. Hoff. Adaptive Switching Circuits. In *1960 IRE WESCON Convention*, pages 96–104, 1960.