BI-ML1.21 přednáška 1

Daniel Vašata

FIT ČVUT

25. 9. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 25. září 2024 14:24.

Vyučující předmětu

- **Daniel Vašata** (přednáší, cvičí a garantuje)
- **Ivo Petr** (cvičí)
- **Ondřej Tichý** (cvičí)

Podmínky získání zápočtu

- **Zápočet bude postaven na vypracovávání domácích úkolů.**
- Úkoly budou během semestru **dva**, každý za **max. 25 bodů**.
- Domácí úkoly budete vypracovávat v jazyce Python ve formátu Jupyter notebook (`.ipynb`).
- Zadání a podrobné instrukce k vypracování a odevzdání najdete na stránkách předmětu:

  courses.fit.cvut.cz/BI-ML1/

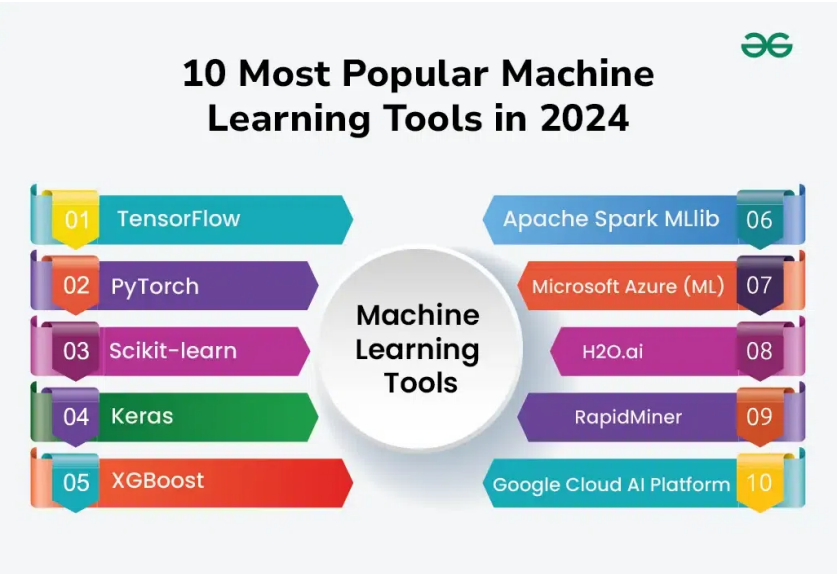
- **K získání zápočtu je třeba získat alespoň 25 bodů z 50.**

Podmínky složení zkoušky

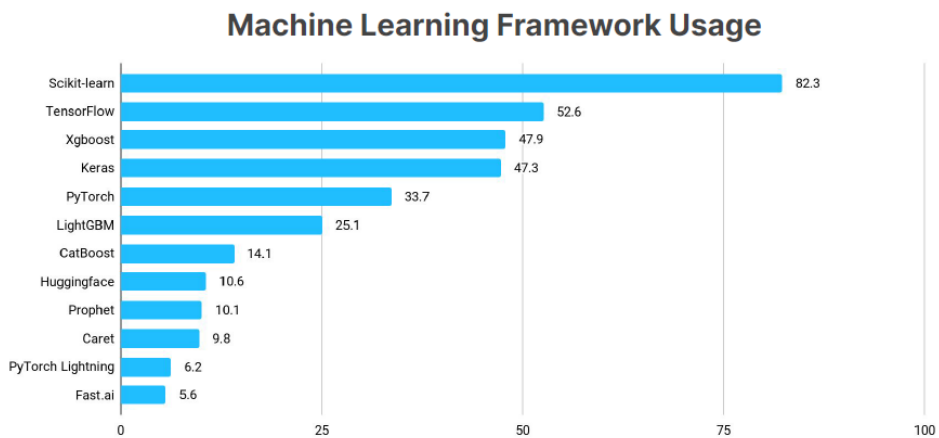
- Zkouška bude **ústní**.
- Každý student dostane **dvě otázky z předem zveřejněného seznamu**.
- Z každé otázky můžete získat **až 25 bodů**.
- Celkem tedy můžete ze zkoušky získat **až 50 bodů**. Není žádný minimální nutný počet bodů, který je třeba ze zkoušky získat.
- Pokud student/ka u zkoušky prokáže zásadní neznalost, může zkoušející použít právo veta a zkoušku ukončit jako neúspěšnou.
- V případě úspěchu u zkoušky se výsledná známka odvodí ze součtu bodů ze semestru a ze zkoušky.

O čem to všechno vlastně bude?

- Ve vší obecnosti: budeme se učit, jak z dat získat informace.
- V mnoha případech to bude znamenat naučit se úspěšně predikovat nějakou veličinu na základě známých charakteristik.
- Budeme se ale také učit nějakým způsobem se v datech pouze vyznat, bez ohledu na potenciální predikce.
- V tomto předmětu budeme používat jazyk **Python** a zejména balíčky, které se pro práci s daty používají.



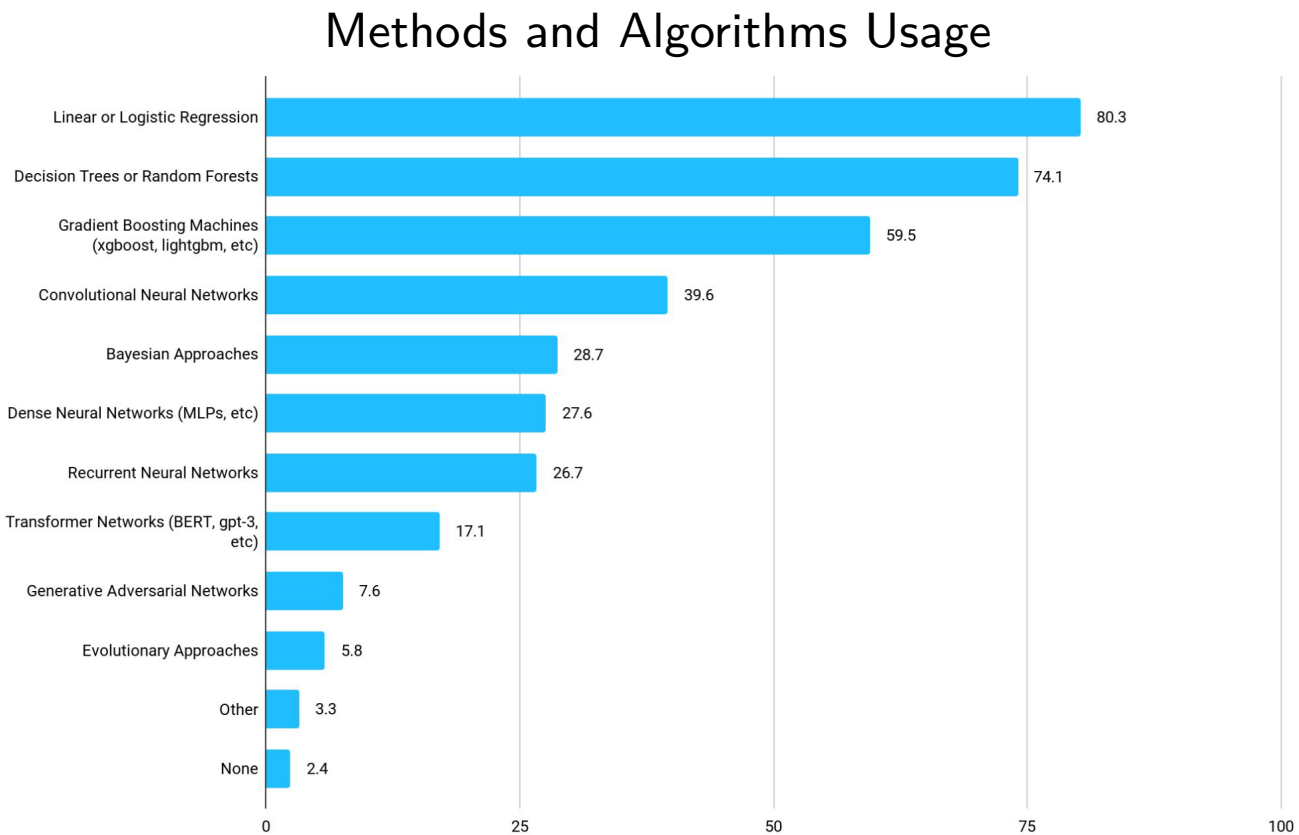
Převzato z geeksforgeeks.



Z výzkumu kaggle.com 2021.

Metody a algoritmy

- Cílem tohoto kurzu je naučit Vás základy dané problematiky, a tedy se zaměříme především na základní metody a algoritmy a na řešení problémů spojených s jejich použitím.
- Ovšem i úplně základní metody stále patří k nejpoužívanějším v praxi!



Z výzkumu kaggle.com 2021.

Z čeho se učit

- Studijní materiály tohoto předmětu by Vám měly stačit pro jeho absolvování: jsou to prezentace k přednáškám (vč. jejich handout formátu), Jupyter notebooky ke cvičením a svým způsobem i úkoly.
- Videa z (letošních i dřívějších) přednášek.
- Probírané metody patří ke klasické látce a lze k nim najít nepřeberné množství videí a textů.
- Pěkným zdrojem je dokumentace ke knihovně `scikit-learn`, kterou budeme pro aplikování probíraných modelů především používat.
- Užitečným zdrojem (nejen) zajímavých datasetů je server www.kaggle.com. Tam najdete uživateli vytvořené příklady použití různých metod, tipy a triky, soutěže (i o docela slušné peníze) atp.

Co už byste měli umět (1/2)

- Předpokládáme, že už umíte programovat, ale nepředpokládáme znalost Pythonu.
- Naučit se základy Pythonu, hlavně specializovaných knihoven, je součást tohoto kurzu.
- Předpokládáme znalost lineární algebry (*BI-LA1* a *BI-LA2*) a matematické analýzy (*BI-MA1* a souběžné *BI-MA2*), neb tabulky jsou matice a strojově učit znamená optimalizovat!
- Také předpokládáme, že průběžně studujete předmět *BI-PST: Pravděpodobnost a statistika*.
- Budeme Vám muset říci alespoň trochu o hledání extrémů funkcí více proměnných (viz *BI-MA2*) a pár dalších věcí.

Co už byste měli umět (2/2)

Machine Learning in a nutshell

[zdroj: xkcd.com/1838/]

Navazující a související předměty (1/3)

BI-ML2 **Strojové učení 2**: Navazuje přímo na tento předmět, neuronové sítě, posilované učení a další techniky.
(letní semestr).

?I-SZ **Seminář znalostního inženýrství**: V rámci tohoto předmětu budete zpracovávat (Vámi) vybrané téma pod vedením jednoho z vyučujících či hostů.
(Magda Friedjungová, Pavel Kordík a Rodrigo Alves, letní i zimní semestr).

BI-SVZ **Strojové vidění a zpracování obrazu**: Práce s kamerovými systémy a zpracování obrazu z takových zařízení. Odehrává se v Improlabu.
(doc. Marcel Jiřina a Improlab, zimní a letní semestr).

BI-ZUM **Základy umělé inteligence**: Obecnější zaměření na umělou inteligenci, agentní systémy.
(prof. Pavel Surynek, letní semestr).

BI-VIZ **Vizualizace dat**: Jak vizualizovat data, nástroje i postupy.
(Magda Friedjungová, zimní semestr).

BI-PRS **Praktická statistika**: Statistika nad reálnými daty, nástroj R.
(doc. Kamil Dedecius a Petr Novák, letní semestr).

Navazující a související předměty (2/3)

NI-PDD **Předzpracování dat**: Zaměřeno na přípravu dat pro různé metody. Tato fáze je svým způsobem nejdůležitější a má největší vliv na kvalitu výsledku.
(doc. Marcel Jiřina, letní semestr, povinný pro ZI).

NI-UMI **Umělá inteligence**: Navazuje a rozšiřuje předmět *BI-ZUM*.
(prof. Pavel Surynek, letní semestr, povinný pro ZI).

NI-ADM **Algoritmy data miningu** Nejbližší příbuzný *BI-ML1*, na který přímo navazuje. Zaměření je podobné, témata ovšem pokročilejší.
(doc. Pavel Kordík, Rodrigo Alves a Daniel Vašata, zimní semestr, povinný pro ZI)

NI-MVI **Metody výpočetní inteligence** Je zaměřený především na moderní použití neuronových sítí.
(doc. Pavel Kordík a spol, letní semestr, povinný pro ZI)

Navazující a související předměty (3/3)

NI-SCR **Statistická analýza časových řad**: Předmět je zaměřen na práci s daty, které mají formát časové řady.
(doc. Kamil Dedecius, zimní semestr, povinný pro ZI).

NI-BML **Bayesovské metody ve strojovém učení**: Praktické využití základních metod bayesovského modelování v dynamicky se rozvíjející oblasti machine learningu.
(doc. Kamil Dedecius a Ondřej Tichý, letní semestr).

NI-GNN **Grafové neuronové sítě**: Umělá inteligence pro práci s grafovými daty.
(Miroslav Čepek, letní semestr).

NI-EDW **Podnikové datové sklady**: Prakticky zaměřený předmět o návrhu, architektuře a budování datových skladů.
(organizuje M. Friedjungová, letní semestr).

NIE-PML **Personalised ML**: Cílem kurzu je naučit studenty vytvářet modely na základě vlastností a chování jednotlivých entit (uživatelů).
(Rodrigo Alves, zimní semestr)

... ...

BI-ML1.21 přednáška 2

Daniel Vašata

FIT ČVUT

25. 9. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 9. října 2024 15:59.

Co bude v dnešní přednášce

- základní informace, úvodní příklad
- problém klasifikace
- jak fungují rozhodovací stromy
- jak se budují rozhodovací stromy
- hyperparametry a testovací, trénovací a validační data
- jak na spojité příznaky
- jak na spojitou vysvětlovanou proměnnou

Příklad: prenatální (1/4)

- První *datový model* se Vás pravděpodobně týkal ještě v prenatálním období Vašeho života.
- Při odhadování porodní váhy plodu pomocí ultrazvuku (tzv. SONO) se používá mj. následující model:

Model pro porodní váhu (Shephard et al., 1982)

$$\log_{10}(\text{eFW}) = -1.749 + 0.017 \cdot \text{BPD} + 0.005 \cdot \text{AC} - \frac{2.646}{1000} \cdot (\text{BPD} \cdot \text{AC}),$$

eFW = odhadovaná hmotnost při narození, BPD = biparietal diameter (příčný průměr hlavy), AC = abdominal circumference (obvod břicha).

- Pro nás to bude **lineární model bázových funkcí** (7. přednáška) což je rozšíření **lineárního regresního modelu** (5. přednáška), kde je
  - porodní váha eFW resp. její logaritmus je tzv. **vysvětlovaná proměnná** (angl. **target variable**),
  - BPD a AC jsou pak tzv. **příznaky** (angl. a často i česky **features**),
  - čísla $1.749, 0.017$ atd. jsou takzvané **regresní koeficienty** (příp. váhy), obecně **parametry modelu**.

Příklad: prenatální (2/4)

Model pro porodní váhu (Shephard et al., 1982)

$$\log_{10}(\text{eFW}) = -1.749 + 0.017 \cdot \text{BPD} + 0.005 \cdot \text{AC} - \frac{2.646}{1000} \cdot (\text{BPD} \cdot \text{AC}),$$

- **Jak se tento model používá?**
  1. Na ultrazvuku se změříte veličiny BPD a AC,
  2. získaná čísla dosadíte do pravé strany vzorce,
  3. výsledek je odhad hodnoty $\log_{10}(\text{eFW})$ a tedy i kýžené porodní váhy eFW.

- **Kde se vzala ta divná čísla jako -1.749 apod.?**
  To se právě budeme učit v tomto kurzu: Jsou to parametry zvoleného modelu a ty se vždy nějakým způsobem odhadují na základě dostupných dat, to je tzv. **učení modelu**.

Příklad: prenatální (3/4)

Jak asi autoři došli právě k tomuto modelu:

1. Z nějakého důvodu věřili, že příznaky BPD a AC jsou pro porodní váhu důležité. Nejspíše ale zkoušeli i jiné, ale ty buď nepřinášely velké zlepšení nebo se daly BPD a AC plně nahradit.

2. Pomocí různých testovacích procedur si jako model zvolili lineární regresní model s třemi příznaky (poslední příznak je spočten jako funkce prvních dvou) a čtyřmi koeficienty:

$$\log_{10}(\text{eFW}) = w_0 + w_1 \cdot \text{BPD} + w_2 \cdot \text{AC} + w_3 \cdot (\text{BPD} \cdot \text{AC}).$$

3. Předchozí fázi, kdy hledáme „tvar“ modelu, se říká **ladění hyperparametrů** (angl. **hyperparameter tuning**).

4. Koeficienty w_i pak odhadli z dat o již narozených dětech, u kterých měli přesnou porodní váhu i prenatálně naměřené hodnoty BPD a AC.

Příklad: prenatální (4/4)

Zmíněná data, na kterých se model učil, mohla vypadat nějak takto:

kid_id	FW	BPD	AC
1	číslo	číslo	číslo
2	číslo	číslo	číslo
⋮	⋮	⋮	⋮

Pro zajímavost (detaily v 5. přednášce):

- Označme sloupec pod FW jako vektor $\boldsymbol{Y}$
- a vytvořme matici $\mathbf{X}$ o čtyřech sloupcích, kde v každém řádku je vektor $(1, \text{BPD}, \text{AC}, \text{BPD} \cdot \text{AC})$.
- Nejpoužívanější metoda pro výpočet koeficientů w_i je **metoda nejmenších čtverců**, ta nám říká, že

$$(w_0, w_1, w_2, w_3)^T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

- Tento vzorec jsme získali vyřešením jistého optimalizačního problému (a.k.a. *hledání extrémů*), což je velice častý případ: **učení modelu = optimalizace**!

Supervizované vs. nesupervizované učení

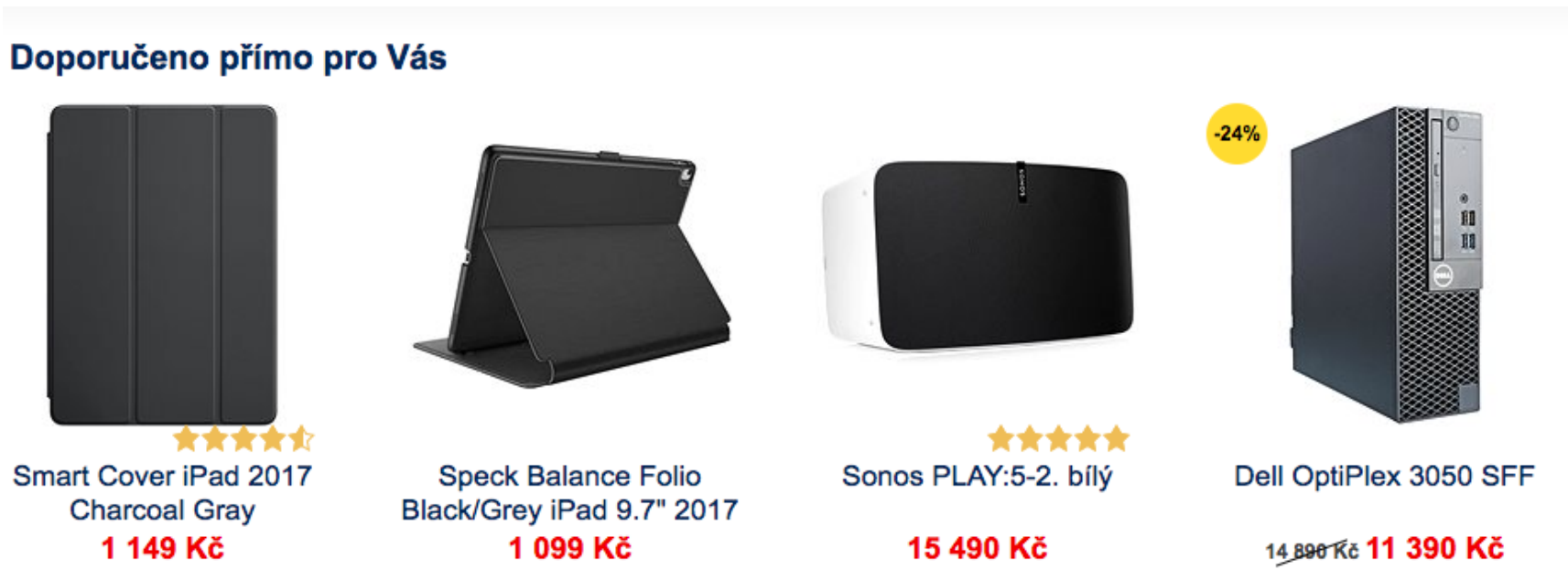
- Předchozí *prenatální* příklad je typickou ukázkou **supervizovaného učení** (učení s učitelem, angl. **supervized learning**).
- Tím „učitelem“ jsou zde známé hodnoty porodních vah u dětí, což je veličina, kterou se snažíme pomocí modelu *predikovat* resp. pochopit, na čem závisí.
- Někdy takovou veličinu ale ani nemáme a prostě se v datech pokoušíme nějak vyznat a najít jejich skrytou strukturu.
- Takovým problémům se říká **nesupervizované učení** (učení bez učitele) a typickým příkladem je **shlukování** dat (téma 11. přednášky).

Příklady nesupervizovaného učení

- Problém shlukování je velice obvyklý v praxi.
- Pokud máte například e-shop (nebo banku, nebo telefonního operátora), chcete se vyznat ve svých zákaznících, o kterých máte nasbíraná různá data (tzv. *segmentace zákazníků*).
- Můžete tak hledat např. podmnožinu „nejlepších“ zákazníků, kterým má cenu věnovat speciální péči. Nebo naopak skupinu, která potřebuje k polepšení pomoci nějakou reklamní akcí (cílení reklamy je velký byznys).
- Do nesupervizovaného učení také (obvykle) spadá i **detekce anomálií** (angl. **anomaly detection**).
- Např. banka se snaží najít podezřelé transakce (fraud detection, ochrana proti zneužití karty, atp.).

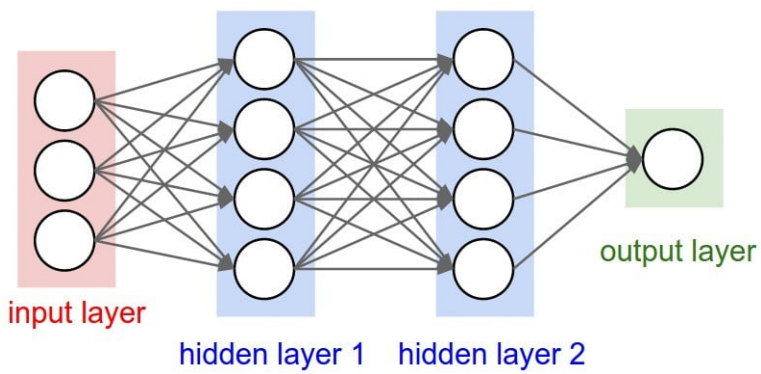
Další příklady: doporučovací systémy

- Dalším příkladem problému řešeného pomocí zkoumání dat je tzv. **doporučování** (angl. **recommendation**).
- Například: vlastníte-li e-shop (příp. internetový časopis, iTunes, Netflix atp.), snažíte se na základě dat o zákaznících a zejména zákazníkovi, který právě prohlíží Vaše stránky, odhadnout, co by si tak mohl ještě chtít koupit (přečíst, podívat, poslechnout) a to mu ukázat.



Další příklady: data bez jasných příznaků

- Data ale vždy nemusí mít formu tabulky. Může se jednat o obrázky, videa, časové řady, dlouhé texty atp., ze kterých je těžké získat pro modely příznaky.
- V takovém případě si musíte dát práci a nějaké příznaky z dat vydolovat (tzv. **feature extraction**).
- Nebo použijete algoritmy a metody, které si příznaky vytvářejí samy automaticky.
- Mezi takové metody patří (čím dál populárnější a mocnější) umělé **neuronové sítě** (angl. artificial **neural networks**, ANN)
- O neuronových sítích budeme mluvit v v BI-ML2. Používají se k všemožným úkolům (překlady, detekce objektů v obrázku videu, hraní GO, shlukování, detekci anomálií, . . .).



Co je problém klasifikace

- Supervizované učení: Snažíme se zjistit, jak vysvětlovanou proměnnou Y ovlivňují příznaky $X_1, \ldots, X_p$, hledáme tedy nějaký funkční vztah tak, aby „co nejvíce platilo“
$$Y \approx f(X_1, X_1, \ldots, X_p).$$
- Funkce f nemusí být nutně podobná funkcím, které znáte z analýzy. Např. v této přednášce to bude strom ϟ.
- Tvar hledané funkce často ovlivňuje to, jakých hodnot může nabývat vysvětlovaná proměnná Y:
  - Může-li nabývat jen několik málo hodnot, mluvíme o problému **klasifikace** (angl. **classification**). Sem spadá např. určení, jestli pacient má/nemá nemoc, jaké písmeno je (ručně) napsáno na obrázku, atp.
  - Může-li nabývat tolika hodnot, že je rozumnější ji považovat za *spojitou*, mluvíme o problému **regrese** (angl. **regression**).
- **Rozhodovací stromy** (angl. **decision trees**) lze použít pro oba typy problému: my začneme tím klasifikačním.

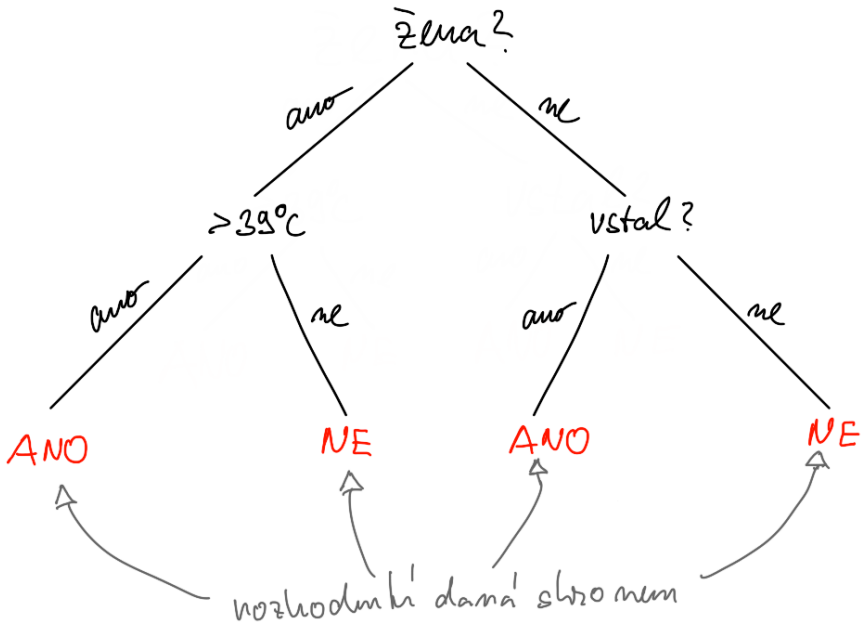
Ukázka použití rozhodovacího stromu (1/6)

- Velmi často je klasifikační problém **binární**, kdy proměnná Y může mít jen dvě hodnoty.
- My si použití stromu ukážeme na (vymyšlených) datech a problému určování, jestli pacient má či nemá závažnou nemoc známou jako „rýmička“.
- Příznaky budou pro jednoduchost také binární: Pohlaví (žena/muž), horečka ($> 39°\mathrm{C}/\leq 39°\mathrm{C}$) a to, jestli daný člověk zvládl/nezvládl vstát z postele.
- Ukážeme si dva rozhodovací stromy a porovnáme si, jak je který z nich dobrým modelem následujících dat:

rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?
ano	muž	ne	ne
ne	žena	ano	ano
ne	muž	ne	ano
ano	žena	ano	ne

Ukázka použití rozhodovacího stromu (2/6)

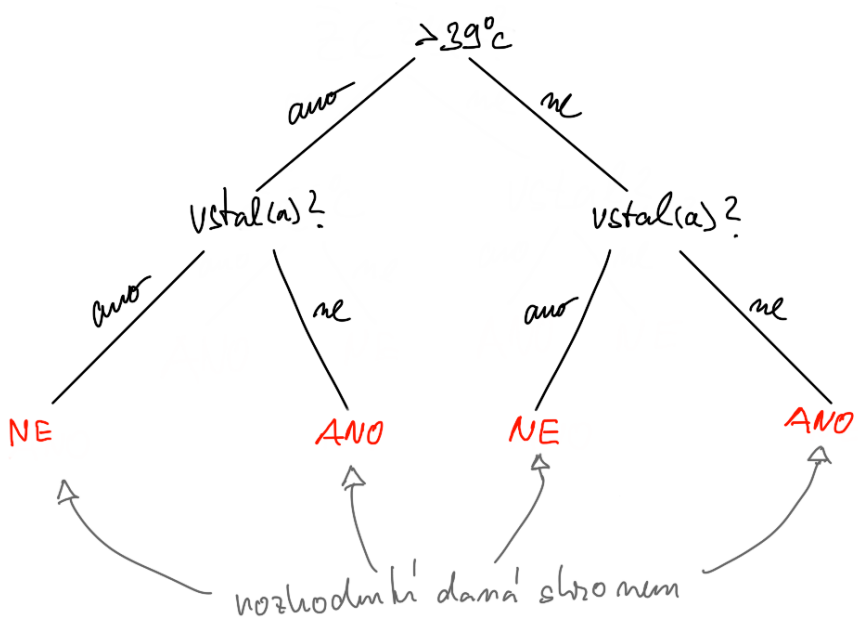
Strom 1:



rýmička	pohlaví	$> 39°C$	vstal(a)?	co říká strom
ano	muž	ne	ne	ne
ne	žena	ano	ano	ano
ne	muž	ne	ano	ano
ano	žena	ano	ne	ano

Ukázka použití rozhodovacího stromu (3/6)

Strom 2:



rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?	co říká strom
ano	muž	ne	ne	ano
ne	žena	ano	ano	ne
ne	muž	ne	ano	ne
ano	žena	ano	ne	ano

Ukázka použití rozhodovacího stromu (4/6)

- Strom 1 dává špatný výsledek pro tři řádky, strom 2 dává správné výsledky pro všechny řádky.
- Strom 2 je tedy, zdá se, lepší. Je však toto dostatečné zdůvodnění?
- My ve skutečnosti chceme vědět, jak často se strom trefí **pro všechna možná data**.
- Což je trochu smůla, protože všechna možná data nikdy nemáme. Máme většinou jen „jednu tabulku“ dat a ta nám musí stačit jak pro vytvoření (neboli naučení) stromu, tak i pro ověření toho, jak je dobrý.
- Jak se to dělá, si ukážeme později.

Ukázka použití rozhodovacího stromu (5/6)

- Strom 1 i strom 2 jsou stromy hloubky 2 a mají tedy 4 listy.
- Kdybychom tedy vytvořili strom hloubky 3, měl by 8 listů. Přesně tolik je ale taky možných kombinací hodnot tří příznaků! Na natrénování takového stromu ale nemáme dostatek dat.
- Strom hloubky tři by se mýlil pouze v případě, že by hodnoty všech příznaků byly stejné, ale hodnota vysvětlované proměnné by byla jiná (např. kdyby jeden pacient byl chlapík s angínou):

pause

rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?
⋮	⋮	⋮	⋮
ano	muž	ano	ne
ne	muž	ano	ne
⋮	⋮	⋮	⋮

- V takovém případě by se mýlil libovolný strom a dokonce i jakákoli funkce příznaků (viz definici funkce).

Ukázka použití rozhodovacího stromu (6/6)

- Skutečným model rýmičky je, jak známo, toto: rýmička nastává právě když

$$(\text{žena} \ \wedge (> 39^\circ) \wedge \ \text{nevstala}) \vee (\text{muž} \ \wedge ((> 39^\circ) \vee \ \text{nevstal}))$$

- Takový model ale není postižitelný stromem hloubky dva!

Konstrukce stromu: základní úkoly

Postupně si ukážeme, jak se řeší následující problémy:

- Máme-li data s příznaky i s hodnotami vysvětlované proměnné, jak zkonstruuji rozhodovací strom, který data co nejlépe modeluje?
- Jak poznám, že můj vytvořený strom není jen dobrým modelem dat, která mám, ale bude dobře fungovat i pro data jiná?
- Jak poznám, jakou mám zvolit hloubku stromu příp. jiné jeho parametry?
- Jak si poradit s nebinárními, nebo dokonce se spojitými parametry?

Konstrukce stromu: formulace úlohy

- Na vstupu máme N řádkovou tabulku s hodnotami pro binární vysvětlovanou proměnnou a p binárních příznaků $X_1, \ldots, X_p$.
- Cílem je vytvořit strom zadané hloubky k, který správně přiřadí hodnotu Y co nejvíce řádkům z tabulky.
- Jak to vyřešit? Vyzkoušejme všechny stromy a pro každý změříme podíl správně určených a je hotovo!
- Nebo je to snad problém?
- Stromů hloubky 1 je p, stromů hloubky 2 je $p \cdot (p-1)^2$, stromů hloubky 3 je „fakt hodně“, ...
- Konstrukce hrubou silou je neprůchozí kvůli počtu možných stromů, ve skutečnosti je konstrukce optimálního stromu **NP-úplný** problém (viz [Hyafil, Rivest, (1976)]).

Konstrukce stromu: hladový ID3 algoritmus

- Pro konstrukci stromů se používá **hladový algoritmus** označovaný jako **ID3** (resp. jeho rafinovanější verze **C4.5** a **C5** vše od **Johna Rosse Quinlana**).
- Tyto algoritmy pro danou množinu dat vybírají jeden (ze zatím nepoužitých) příznaků, který rozdělí data na dvě části tak, že vzniklé rozdělení maximalizuje vybrané kritérium (viz dále).
- Daná množina dat je tak rozdělena na dvě části a na každou zvlášť je pak aplikován stejný postup, jehož výsledkem jsou další dva příznaky, použité jako kritérium a rozdělení na čtyři podmnožiny dat.
- Takto se postupuje, dokud nenastane nějaké zastavovací kritérium (maximální hloubka stromu, na listech stromu už je málo dat, atp.).
- Jak zvolit to kritérium? Používají se dvě, o obou si řekneme později.

Konstrukce stromu: ukázková data

Zkusme zkonstruovat strom pro následující binární data s třemi příznaky:

id	Y	X_1	X_2	X_3
0	1	1	0	0
1	1	0	1	1
2	1	1	0	0
3	1	1	1	1
4	0	0	0	1
5	0	0	1	0
6	0	0	0	1
7	0	1	1	0

? Jaký příznak použít jako první k rozdělení dat?

- Příznak X_1 rozdělí data na dvě části[1] $\{1_0, 1_2, 1_3, 0_7\}$ a $\{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_2 rozdělí data na dvě části $\{1_1, 1_3, 0_5, 0_7\}$ a $\{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_3 rozdělí data na dvě části $\{1_1, 1_3, 0_4, 0_6\}$ a $\{1_0, 1_2, 0_5, 0_7\}$.

[1] Uvádíme hodnotu Y a jako dolní index `id` přísl. řádku.

Konstrukce stromu: kritérium volby příznaku

- Příznak X_1 rozdělí data na dvě části $\{1_0, 1_2, 1_3, 0_7\}$ a $\{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_2 rozdělí data na dvě části $\{1_1, 1_3, 0_5, 0_7\}$ a $\{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_3 rozdělí data na dvě části $\{1_1, 1_3, 0_4, 0_6\}$ a $\{1_0, 1_2, 0_5, 0_7\}$.

? Který příznak je lepší a jak to změřit?

- Na vstupu jsou data $\{1_0, 1_1, 1_2, 1_3, 0_4, 0_5, 0_6, 0_7\}$, kde jsou hodnoty 0 a 1 zastoupeny rovnoměrně.
- Ideální by byl příznak, který data rozdělí na dvě skupiny, jednu se samými 1 a druhou s 0. Takový ale nemáme k dispozici.
- Příznaky X_2 a X_3 jsou ale opačný extrém: Data rozdělí na dva kusy, kde jsou opět 0 a 1 zastoupeny přesně napůl.
- Příznak X_1 pak představuje zřejmě nejlepší volbu, neb data alespoň trochu více „uspořádá“: z rovnoměrného zastoupení k poměru 1 ku 3.
- Jak tuto uspořádanost resp. neuspořádanost měřit?

Míra **ne**uspořádanosti

- Máme množinu $\mathcal{D}$ nul a jedniček (nebo i více hodnot) a chceme nějak změřit, jak moc je uspořádaná.
- Co by taková míra měla splňovat? Označme p_0 a p_1 poměry počtu 0 resp. 1 v množině (tj. $p_0 + p_1 = 1$).
  1. Míra by měla být nezáporná (z technických důvodů).
  2. Pokud jsou v množině např. samé nuly (tj. $p_0 = 1$), měla by být neuspořádanost nulová.
  3. Měla by být maximální, pokud jsou počty nul a jedniček stejné, tj. když $p_0 = p_1 = \frac{1}{2}$.
  4. Měla by to být rostoucí funkce p_0 na intervalu $[0, \frac{1}{2}]$ a klesající na intervalu $[\frac{1}{2}, 1]$.
- Taková funkce **měřící neuspořádanost** existuje a říká se jí **entropie**:

$$H(\mathcal{D}) = -p_0 \log p_0 - p_1 \log p_1 = -p_0 \log p_0 - (1 - p_0) \log(1 - p_0).$$

- Pro nebinární případ s k různými hodnotami je to analogické:

$$H(\mathcal{D}) = -\sum_{i=0}^{k-1} p_i \log p_i.$$

Entropie: poznámky

- Formálněji se budete entropii věnovat v předmětu *NI-VSM: Vybrané statistické metody*, kde si zavedete entropii *náhodné veličiny*. Nám ale stačí předchozí zavedení[2].
- Ve vzorci pro entropii budeme používat dvojkový logaritmus. V takovém případě se jednotce entropie říká **bit**.
- Entropie množiny našich vstupních dat $\mathcal{D} = \{1_0, 1_1, 1_2, 1_3, 0_4, 0_5, 0_6, 0_7\}$ je rovna 1, neboť $p_0 = \frac{1}{2}$, a tedy

$$H(\mathcal{D}) = -\frac{1}{2}\log\frac{1}{2} - \left(1-\frac{1}{2}\right)\log\left(1-\frac{1}{2}\right) = -2\frac{1}{2}\log\frac{1}{2} = -\log\frac{1}{2} = 1.$$

- Entropie množiny dat $\mathcal{D}_1 = \{1_0, 1_1, 1_3, 0_7\}$ je rovna ($p_0 = \frac{1}{4}$)

$$H(\mathcal{D}_1) = -\frac{1}{4}\log\frac{1}{4} - \left(1-\frac{1}{4}\right)\log\left(1-\frac{1}{4}\right) = 0.811278124459133 28\ldots$$

[2] To, co počítáme, není přesně řečeno entropie, ale její odhad na základě dat. Skutečné pravděpodobnosti p_i totiž vlastně neznáme a používáme jen jejich odhad.

Konstrukce stromu: informační zisk

Vraťme se k otázce, který příznak použít pro rozdělení množiny dat $\mathcal{D} = \{1_0, 1_1, 1_2, 1_3, 0_4, 0_5, 0_6, 0_7\}$:

- Příznak X_1: $\mathcal{D}_1 = \{1_0, 1_2, 1_3, 0_7\}$ a $\mathcal{D}_0 = \{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_2: $\mathcal{D}_1 = \{1_1, 1_3, 0_5, 0_7\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_3: $\mathcal{D}_1 = \{1_1, 1_3, 0_4, 0_6\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_5, 0_7\}$.

Chceme vybrat příznak, který rozdělením dat nejvíce sníží neuspořádanost!

Toto snížení se určuje tzv. **informačním ziskem** (angl. **information gain**), který je definován jaké „entropie $\mathcal{D}$ mínus vážený součet entropií $\mathcal{D}_0$ a $\mathcal{D}_1$“.

Formálně:

$$IG(\mathcal{D}, X_i) = H(\mathcal{D}) - t_0 H(\mathcal{D}_0) - t_1 H(\mathcal{D}_1)$$

kde $\mathcal{D}_0$ a $\mathcal{D}_1$ jsou podmnožiny dat $\mathcal{D}$, pro které $X_i = 0$ resp. $X_i = 1$, a t_i je podíl počtu prvků v $\mathcal{D}_i$ a $\mathcal{D}$, neboli $t_i = \frac{\#\mathcal{D}_i}{\#\mathcal{D}}$.

Konstrukce stromu pro ukázková data (1/4)

Spočítejme informační zisk pro naše tři příznaky:

- Příznak X_1: $\mathcal{D}_1 = \{1_0, 1_2, 1_3, 0_7\}$ a $\mathcal{D}_0 = \{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_2: $\mathcal{D}_1 = \{1_1, 1_3, 0_5, 0_7\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_3: $\mathcal{D}_1 = \{1_1, 1_3, 0_4, 0_6\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_5, 0_7\}$.

- Už víme, že $H(\mathcal{D}) = 1$.
- Pro X_1 dostáváme $H(\mathcal{D}_0) = H(\mathcal{D}_1) = 0.811278124459133 8$ a tedy

$$IG(\mathcal{D}, X_1) = H(\mathcal{D}) - \frac{1}{2}H(\mathcal{D}_0) - \frac{1}{2}H(\mathcal{D}_1) = 1 - 0.811 = 0.189.$$

- Pro X_1 s X_2 postupujeme podobně. Pro ně platí $H(\mathcal{D}_0) = H(\mathcal{D}_1) = 1$ a tedy

$$IG(\mathcal{D}, X_2) = IG(\mathcal{D}, X_3) = H(\mathcal{D}) - \frac{1}{2}H(\mathcal{D}_0) - \frac{1}{2}H(\mathcal{D}_1) = 1 - 1 = 0.$$

Vítězem hladového souboje je tedy příznak X_1.

Konstrukce stromu pro ukázková data (2/4)

- Příznak X_1: $\mathcal{D}_1 = \{1_0, 1_2, 1_3, 0_7\}$ a $\mathcal{D}_0 = \{1_1, 0_4, 0_5, 0_6\}$.
- Nyní aplikujeme stejný postup dvakrát: na $\mathcal{D}_1$ a na $\mathcal{D}_0$. Nemá už ale smysl dělit data podle X_1, takže budeme zkoušet jen X_2 a X_3.
- Data $\mathcal{D}_1$ příznak X_2 rozdělí na $\mathcal{D}_{11} = \{1_3, 0_7\}$ a $\mathcal{D}_{10} = \{1_0, 1_2\}$, informační zisk je tedy

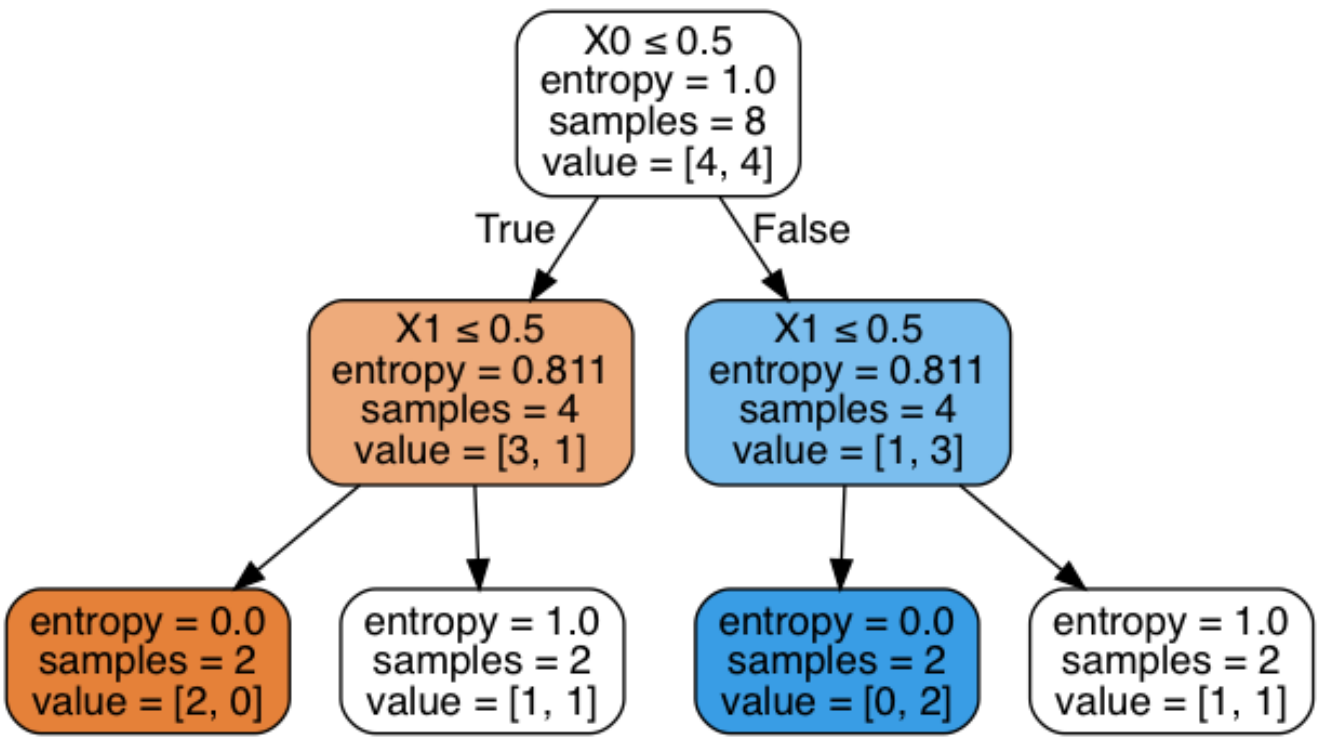
$$IG(\mathcal{D}_1, X_2) = H(\mathcal{D}_1) - \frac{1}{2}H(\mathcal{D}_{11}) - \frac{1}{2}H(\mathcal{D}_{10}) = 0.811 - \frac{1}{2}1 - \frac{1}{2}0 = 0.311\,.$$

- Pro příznak X_3 vyjde informační zisk nižší (zhruba 0.123, spočítejte si), takže hladový souboj vyhrává příznak X_2.

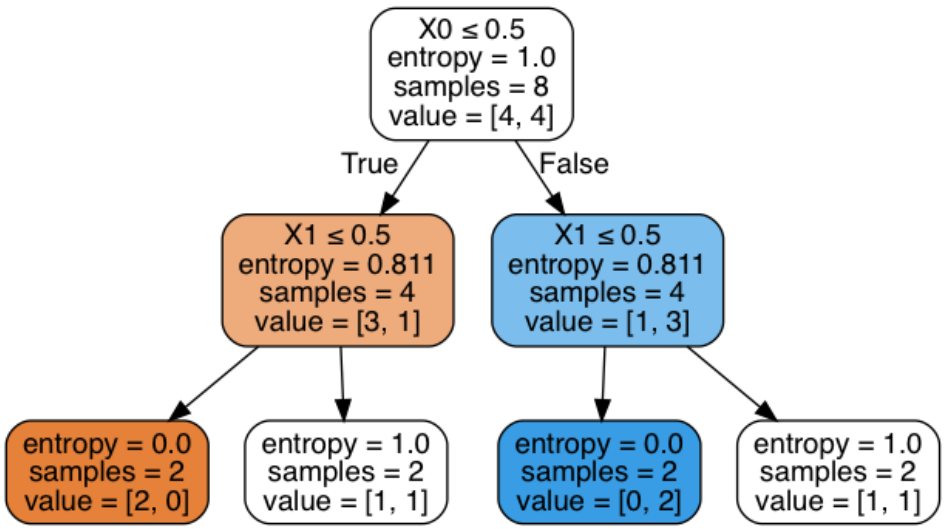
Konstrukce stromu pro ukázková data (3/4)

Takto přesně postupuje i implementace konstruování rozhodovacích stromů v knihovně `sklearn` (s tím, že příznaky jsou v Pythonu indexovány od 0):

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='entropy', max_depth=2)
dt.fit(X,Y)
```



Konstrukce stromu pro ukázková data (4/4)



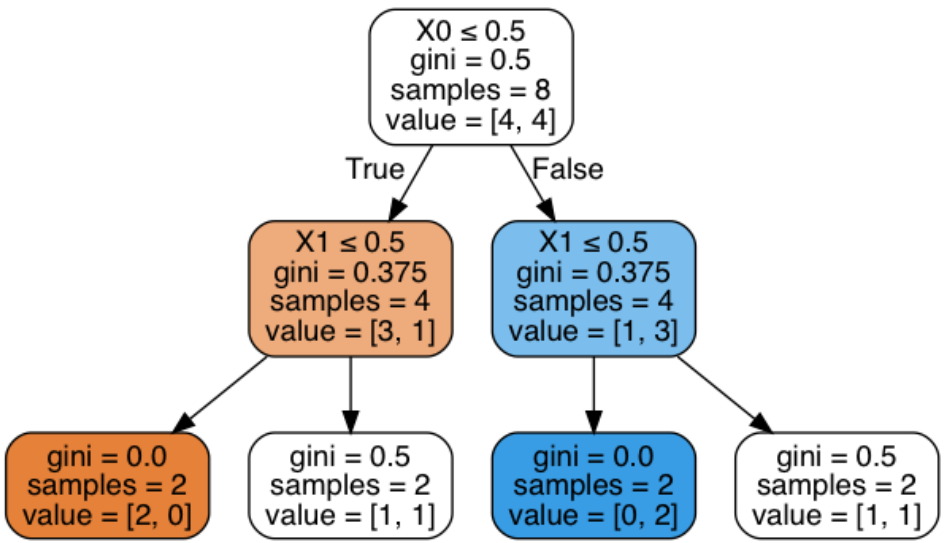
- Strom jsme zkonstruovali, ale ještě jsme k jeho jednotlivým listům nepřiřadili rozhodnutí, jestli pro daný list má být výsledek $Y = 1$ nebo $Y = 0$.
- To se dělá prostým hlasováním daty, které do daného listu „propadnou“: převažují-li nuly (první list zleva), je rozhodnutí $Y = 0$, převažují-li jedničky (třetí list), je to $Y = 1$ a při shodě (druhý a čtvrtý) přiřadíme rozhodnutí náhodně.

Gini index

- Namísto entropie se také používá **Gini index** (angl. **Gini impurity**), pro množinu $\mathcal{D}$ s k různými hodnotami:

$$GI(\mathcal{D}) = 1 - \sum_{0}^{k-1} p_i^2 = \sum_{0}^{k-1} p_i(1 - p_i).$$

- Gini index má podobné vlastnosti jako Entropie. Je to jakási míra toho, že nově přidaný prvek bude špatně klasifikován.
- Jinak ale vše funguje stejně, jen se nahradí $H(\mathcal{D})$ výrazem $GI(\mathcal{D})$.
- Pro naše data to při použití Gini indexu dopadne stejně (blbě):



Hladový $\neq$ optimální

- Získaný rozhodovací strom nebyl bezchybným modelem, neboť vždy u dvou dat rozhodl špatně.
- Jak jsme viděli dříve, je to někdy nevyhnutelná situace, řešitelná pouze tím, že se použije strom s větší hloubkou.
- V tomto případě to tak ale není, neboť existuje strom hloubky dva, který daných osm dat modeluje perfektně.
- **Tento optimální strom ale není dosažitelný hladovým algoritmem!** Přitom je celkem očividně daný podmínkou $Y = 1 \Leftrightarrow (X_2 = X_3)$.

id	Y	X_1	X_2	X_3
0	1	1	0	0
1	1	0	1	1
2	1	1	0	0
3	1	1	1	1
4	0	0	0	1
5	0	0	1	0
6	0	0	0	1
7	0	1	1	0

Konstrukce rozhodovacího stromu: shrnutí

- Konstrukce optimálního stromu je NP-úplný problém, a proto se v praxi používají hladové strategie (algoritmy ID3, C4.5 a C5 od Johna Rosse Quinlana), které ale často najdou *suboptimální* řešení.

- Hladový algoritmus funguje rekurzivně: Pro danou množinu dat najde příznak, který tuto množinu rozdělí tak, aby bylo dosaženo maximálního možného *informačního zisku* (z entropie příp. Gini indexu). Stejný postup se pak opakovaně aplikuje na podmnožiny vzniklé tímto rozdělením.

- Takto se data dělí na menší a menší podmnožiny, dokud nenastane **ukončovací podmínka**, kterou si uživatel zvolil (max. hloubka stromu, minimální počet dat v množině, minimální nutná hodnota informačního zisku, atp., viz cvičení).

Jak je můj strom dobrý?

Nyní se dostáváme k druhé otázce: **Jak poznám, že můj vytvořený strom není jen dobrým modelem dat, která mám, ale bude dobře fungovat i pro data jiná?**

- Chceme-li rozhodovat, jak je model dobrý, potřebujeme nějakou objektivní vyčíslitelnou míru jeho kvality.
- Volba této míry je důležitou součástí celého procesu hledání modelu. Míry se obvykle velmi liší pro problémy klasifikace a regrese.
- My se nyní věnujeme klasifikaci a vystačíme si s přirozenou mírou **klasifikační přesnosti** (angl. **classification accuracy**), která prostě měří poměr správně klasifikovaných, tedy je rovna číslu (příp. procentu)

$$\frac{\text{počet správně klasifikovaných dat}}{\text{počet všech dat}}.$$

- Např. rozhodovací strom zkonstruovaný algoritmem ID3 pro množinu osmi dat se pletl ve dvou případech, a tedy jeho přesnost byla $\frac{6}{8} = 0.75 = 75\ \%$.

Jak je můj strom dobrý?

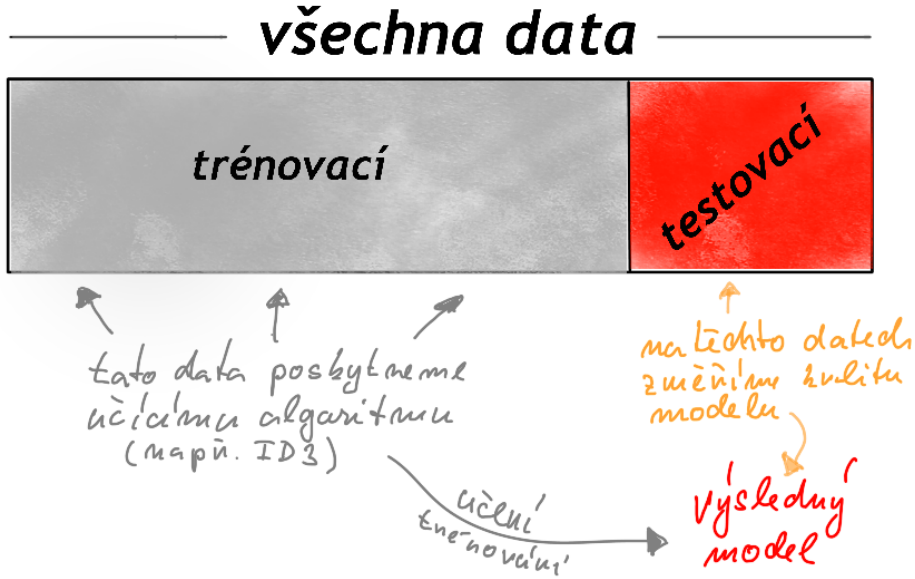
- Mohlo by se zdát, že máme vyřešeno: prostě zkonstruujeme strom, který má pro naše data nejvyšší přesnost.
- S tímto přístupem bychom ale narazili, neb by vedl k hlubokým stromům, které mají na listech třeba jen jeden datový bod.
- Platí totiž, že čím je strom hlubší, tím má lepší přesnost!
- My vlastně nechceme maximalizovat přesnost na našich datech, ale **na všech možných datech**, která ovšem nemáme.

? Jak to vyřešit?

- Uděláme to tak, že naše data rozdělíme na dva kusy: na jednom (tom větším), model naučíme a na tom druhém kusu změříme přesnost. **Tak dostaneme spolehlivější odhad toho, jak se bude náš model chovat pro data, na kterých se neučil.**

Trénovací a testovací data

- Těm dvěma kusům dat se obvykle říká **trénovací** a **testovací data** (angl. **train** a **test set**).
- Chybovosti modelu (pro nás nepřesnost = 1 - přesnost) na těchto množinách dat se pak adekvátně říká **trénovací** a **testovací chyba** (angl. **train** a **test error**).



Přeučení modelu (1/2)

- V případě stromů zjevně platí, že čím hlubší strom učíme, tím dostaneme menší trénovací chybu. Spolehlivější měrou kvality je však testovací chyba, neb tou trénovací si lžeme do kapsy: Měříme, jak dobře jsme *přizpůsobili* strom dostupným datům, nikoli jak dobře jsme odhadli skrytý model za těmito daty schovaným (pokud tedy na takový model věříte).

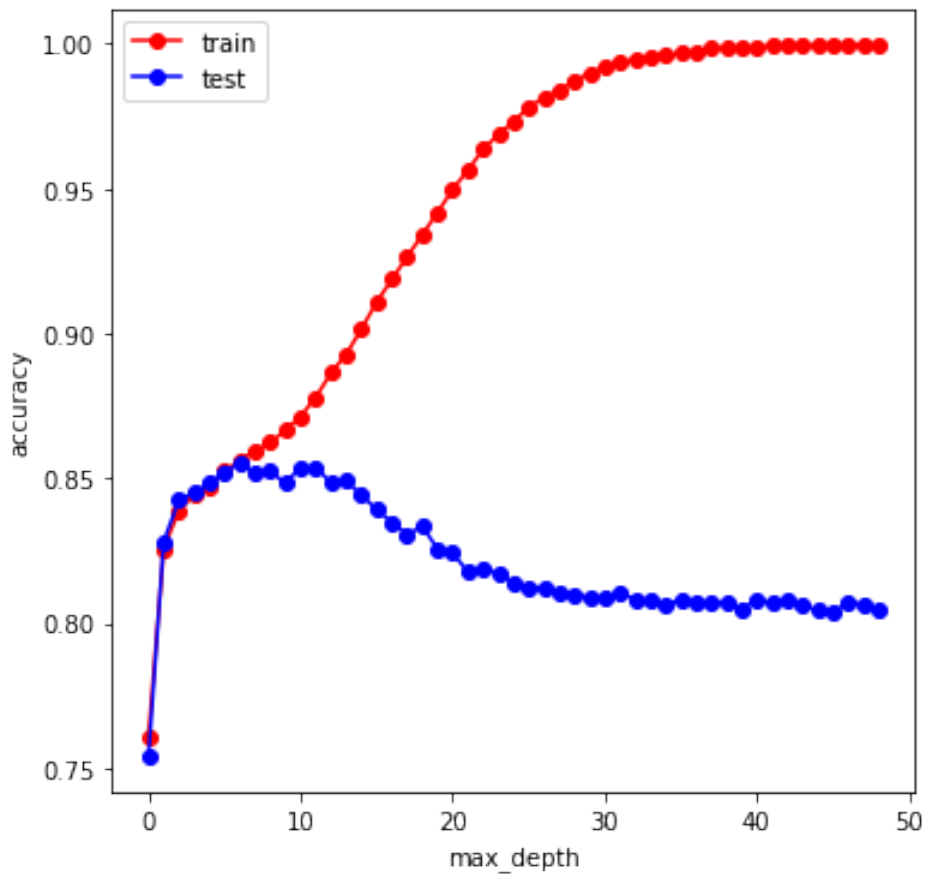
- Tomuto přílišnému přizpůsobení trénovacím datům se říká **přeučení** modelu (angl. **overfitting**).

- Obvykle platí, že čím složitější model (v našem příp. čím hlubší strom), tím nižší trénovací chyba.

- Testovací chyba se však chová jinak: Nejdříve se zvětšováním složitosti modelu klesá, ale v jistý okamžik začne růst. Najít tento bod zlomu je úkolem celého procesu budování modelu.

Přeučení modelu (2/2)

- Následující obrázek ukazuje typický vývoj trénovací a testovací chyby v závislosti na hloubce stromu (`max_depth`).
- Z tohoto obrázku je vidět, že nejrozumnější volba parametru `max_depth` by byla 6.



Ladění hyperparametrů (1/3)

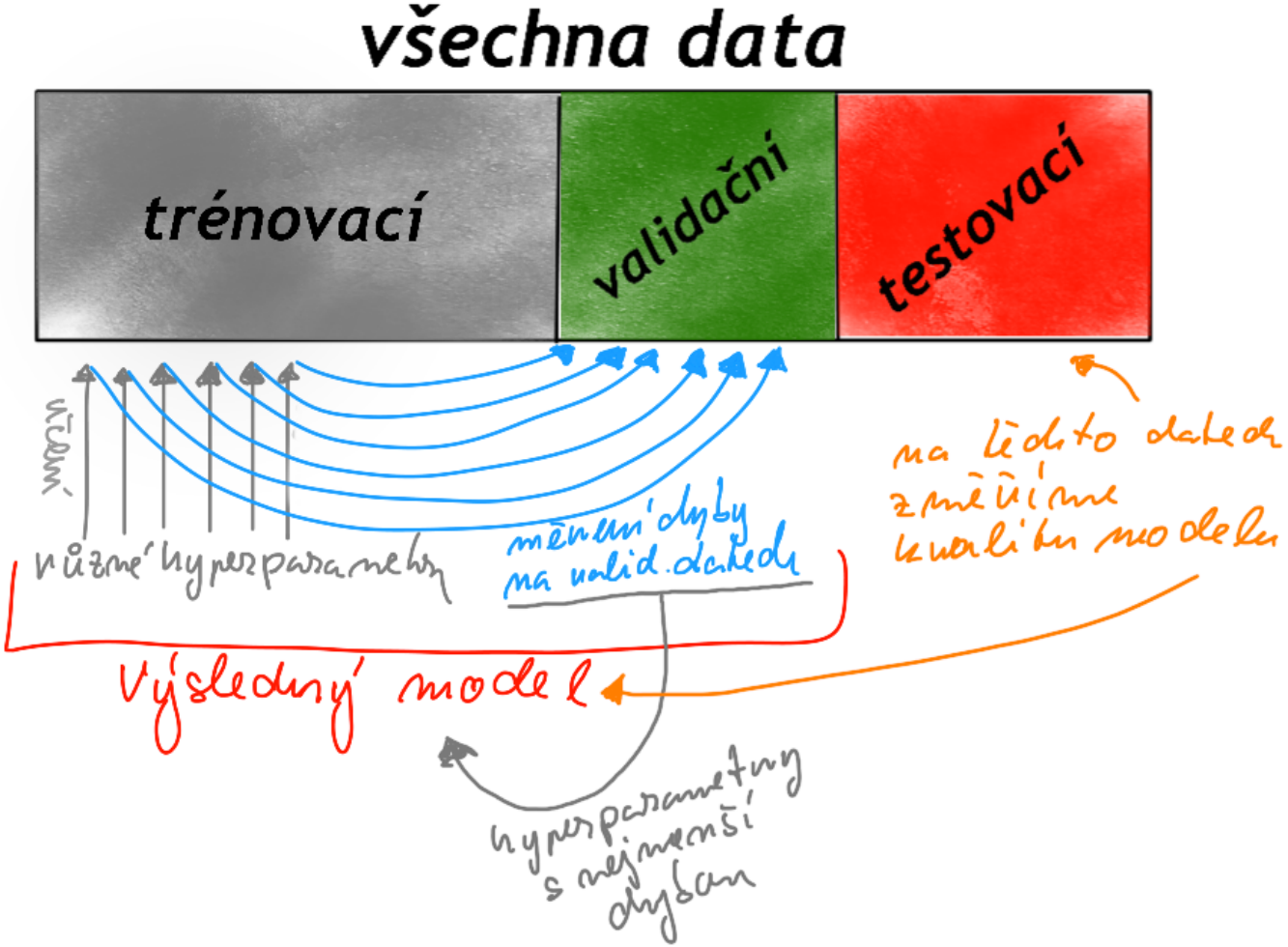
Teď se dostáváme ke třetí otázce: **Jak poznám, jakou mám zvolit hloubku stromu příp. jiné jeho parametry?**

- Mohlo by se zdát rozumné postupovat takto:
  1. Data si rozdělíme na trénovací a testovací.
  2. Pro různé hodnoty hloubky stromu (parametr `max_depth`) naučíme rozhodovací strom na trénovacích datech
  3. a pro každou hloubku stromu změříme přesnost (classification accuracy) na testovacích datech.
  4. Vyberme tu hloubku, která dává nejmenší testovací chybu.
  5. Tuto hodnotu vezmeme také jako odhad chyby na reálných datech, a tedy jako objektivní míru kvality modelu.
- Je někde problém? Ano, je!
- Výsledná testovací chyba bude zpravidla příliš optimistickým odhadem skutečnosti! Výsledný model (konkrétně parametr hloubka stromu) byl vybrán na základě těchto dat a model je tedy těmto datům přizpůsobený.
- Takto bychom porušili zásadu, že **před získáním finálního modelu nesaháme na testovací data**, která je nutná pro to, aby testovací chyba byla rozumným odhadem skutečné chyby modelu.

Ladění hyperparametrů (2/3)

- Jak se s tímto problémem vypořádat?
- Obvykle se to dělá tak, že se data nerozdělí na dvě, ale na tři podmnožiny: trénovací, **validační** (angl. **validation**) a testovací:
  1. Pro různé hodnoty hloubky stromu `max_depth` naučíme rozhodovací strom
  2. a změříme jeho chybu (přesnost) na validačních datech.
  3. Jako optimální hodnotu vybereme tu s nejmenší chybou (tj. s nejvyšší přesností).
  4. Pro finální model pak změříme chybu na testovacích datech, které dosud ležely ladem, a ta je pak rozumným odhadem chyby modelu.
- Parametrům, jako je `max_depth`, které určují *tvar* nebo *komplexitu* modelu, se říká **hyperparametry modelu**.
- Určování těchto hyperparametrů pomocí validačních dat se říká **ladění** (angl. **tuning**).
- Tento hyperparametr nemusí být jeden, ale může jich být více. Pokud jsou spojité, může být výpočetně složité jich otestovat *reprezentativní* množství a je třeba dělat kompromisy.

Ladění hyperparametrů (3/3)



Ladění hyperparametrů: poznámky (1/2)

- Pro představu, jaké hyperparametry jsou dostupné pro rozhodovací stromy v knihovně `sklearn`:

```
Init signature: DecisionTreeClassifier(criterion='gini', splitter='best', max_de
pth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
 max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decreas
e=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

- Význam těchto parametrů si vysvětlíte (a ukážete) na cvičení.
- Dělení dat na trénovací/validační/testovací podmnožiny je dobré dělat náhodně. Tj. nevybrat první půlku dat jako trénovací, další čtvrtinu jako validační a zbytek vzít jako testovací.
- V pořadí dat by mohla být nějaká zákonitost, která by znamenala, že trénovací a testovací data nebudou reprezentativní.
- Proto se postupuje tak, že se data vybírají náhodně. V `sklearn` je na to metoda:

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.25, random_state=33)
```

Ladění hyperparametrů: poznámky (2/2)

- Neexistuje žádný optimální poměr velikosti trénovací/validační/testovací množiny dat.
- Obvyklé poměry jsou takové, že 20 % dat vezmeme jako testovací množinu a ze zbytku vezmeme 20 % jako validační množinu. Co zbude, jsou trénovací data.
- Místo 20 % lze použít 25 %, nebo 30 %.
- Lze použít i sofistikovanější strategie, např. měnit velikost validační množiny a koukat se, co to dělá.
- Často používanou metodou je také **křížová validace**, o které budeme mluvit na 4. přednášce.

Vícehodnotové příznaky: one-hot encoding

- Používané algoritmy rozhodovacích stromů a jejich implementace (vč. té v `sklearn`) umí konstruovat pouze binární stromy.
- Je tedy otázka, jak si poradit s příznaky, které mají více než dvě různé hodnoty.
- V případě příznaků, které nabývají pouze několika možných hodnot, lze tento problém lze vyřešit pomocí tzv. **one-hot encoding**, kdy se jeden příznak s n různými hodnotami nahradí n binárními indikátory těchto různých hodnot (tzv. **dummy variables**).
- Např. v našem příkladu s rýmičkou bychom mohli mít tři možnosti: vstal(a), nevstal(a), spadl(a) z postele. Abychom toto převedli do binárních příznaků pomoci dummy příznaků, museli bychom zavést tři nové binární příznaky *vstal(a) / nevstal(a) / spadl(a)*:

původní příznak	$\rightarrow$ *dummy příznaky* $\rightarrow$	vstal	nevstal	spadl
vstal	$\longrightarrow$	1	0	0
nevstal	$\longrightarrow$	0	1	0
spadl	$\longrightarrow$	0	0	1

Nominální a ordinální příznaky

- Metoda *one-hot encoding* má své nevýhody:
  - Zvyšuje výrazně počet příznaků (dimenzi datasetu), což speciálně u rozhodovacích stromů může výrazně podporovat přeučení.
  - Z datasetu sice nemizí žádná informace, ale vzhledem k tomu, že algoritmus pro konstrukci stromů není optimální (jen „hladový“), může se zvýšení počtu příznaků projevit na kvalitě modelu.
  - Není vhodná pro všechny druhy příznaků s diskrétními hodnotami, zejm. ne pro ty, kde je mezi jednotlivými hodnotami nějaké pořadí.

- Příznaky s konečným počtem možných hodnot, které ale nemají nějaký číselný kvantitativní význam, nazýváme **kategorické** (angl. **categorical**) a jsou v zásadě dvou druhů:

  Nominální — Mezi hodnotami není žádné pořadí, např. *místo narození*, *fakulta*, *pohlaví*, atp.

  Ordinální — Mezi hodnotami je nějaké přirozené uspořádání, např. *vzdělání (základní* $<$ *střední* $<$ *s maturitou* $<$ *univerzitní)* atp.

- Metodu *one-hot encoding* je tedy lepší používat pouze pro nominální příznaky. Ordinální je lepší pouze reprezentovat číselně (očíslováním podle uspořádání) a nechat tak, jak jsou. `Sklearn` s nimi pak může pracovat jako se spojitými příznaky, což dává větší smysl (viz dále).

Spojité příznaky

- V datech ale většinou nemáme pouze kategorické (nominální nebo ordinální) příznaky. Často se v nich objevují příznaky jako *věk*, *cena*, *teplota*, *tlak* atp., které jsou kvantitativní a pracujeme s nimi jako se **spojitými** (angl. **continuous**) příznaky.

- V rozhodovacích stromech mohou k větvení sloužit i tyto příznaky. Pravidlo pro (binární) větvení může být např. věk ≤ 30, které opět množinu dat rozdělí na dvě části.

- Oproti binárním příznakům má smysl, aby se spojité proměnné objevily ve stromu vícekrát. Rozdělíme-li data pravidlem věk ≤ 30, dává smysl „starší“ část dat dělit znovu pravidlem věk ≤ 60. U pravidla s binárním příznakem pohlaví $=$ žena toto smysl nedávalo.

Spojité příznaky a algoritmus

- Algoritmus pro binární příznak $X \in \{0, 1\}$ fungoval tak, že spočítal *informační zisk* rozdělení dat $\mathcal{D}$ podle pravidla $X = 0$ a buď toto dělení použil, nebo si vybral jiný příznak s vyšším informačním ziskem.

- Spojitý příznak X s hodnotami např. z intervalu $[0, 100]$ musí fungovat jinak, neboť pravidel tvaru $X \leq d$ existuje vlastně nekonečno (pro všechna možná $0 < d \leq 100$).

- V základním nastavení algoritmus funguje tak, že ale vlastně všechna dělení ve tvaru $X \leq d$ vyzkouší a vybere to s nejvyšším informačním ziskem. Funguje to takto:
  1. Projdi všechny možné hodnoty příznaku X v právě dělené množině dat $\mathcal{D}$ a setřiď je podle velikosti: $x_1 < \cdots < x_\ell$.
  2. Vyzkoušej rozdělení dat podle pravidla $X \leq (x_{i-1} + x_i)/2$ pro všechna $i = 2, \ldots, \ell$ a pro každé takové rozdělení $\mathcal{D}$ spočítej informační zisk.
  3. Jako nejlepší pravidlo dělení podle příznaku X vezmi to s největším spočítaným informačním ziskem.

Spojité příznaky a algoritmus: příklad

Představme si, že máme data $\mathcal{D}$ s osmi řádky s příznakem `věk`, který má pro řádky 1 až 8 tyto hodnoty:

$$13_1, 35_2, 15_3, 65_4, 35_5, 15_6, 19_7, 19_8,$$

přičemž binární vysvětlovaná proměnná Y má pro tyto řádky hodnoty

$$0_1, 1_2, 0_3, 0_4, 1_5, 0_6, 1_7, 1_8.$$

Entropie $\mathcal{D}$ je tedy maximální, tj. $H(\mathcal{D}) = 1$.

Dle předchozího postupujeme tak, že možné hodnoty X seřadíme: $13, 15, 19, 35, 65$, a vyzkoušíme všechna následující rozdělení.

pravidlo	$\mathcal{D}_L$	$\mathcal{D}_R$	informační zisk
$X \leq 14$	0_1	$1_2, 0_3, 0_4, 1_5, 0_6, 1_7, 1_8$	0.14
$X \leq 17$	$0_1, 0_3, 0_6$	$1_2, 0_4, 1_5, 1_7, 1_8$	0.55
$X \leq 27$	$0_1, 0_3, 0_6, 1_7, 1_8$	$1_2, 0_4, 1_5$	0.05
$X \leq 50$	$0_1, 1_2, 0_3, 1_5, 0_6, 1_7, 1_8$	0_4	0.14

Pokud se tedy nenajde jiný příznak dávající lepší informační zisk, použije algoritmus pro rozdělení dat $\mathcal{D}$ pravidlo $X \leq 17$.

Poznámky

- Pokud má spojitý příznak příliš mnoho hodnot, obvykle se postupuje tak, že se uvažuje pravidlo $X \leq (x_{i-1} + x_i)/2$ například pouze pro každé desáté x_i. Příp. se náhodně vyzkouší daný počet hodnot.

- Implementace v `sklearn` se chová vlastně ke všem příznakům jako ke spojitým.

- Dovede si tak celkem dobře poradit s ordinálními příznaky.

- V případě rovnosti informačního zisku si `sklearn.tree.DecisionTreeClassifier` vybírá náhodně. Algoritmus se tak může chovat nedeterministicky, přestože k tomu není žádný viditelný důvod.

Rozhodovací stromy pro regresi

Pokud bychom chtěli stejně jako s diskrétní pracovat se **spojitou vysvětlovanou proměnnou**, musíme vyřešit dva problémy:

- Jak bude naučený strom určovat *predikovanou* hodnotu vysvětlované proměnné?
  - V případě diskrétní proměnné se rozhodovalo hlasováním v rámci listu: Skončila-li cesta stromem pro datový bod v listu, ve kterém z trénovacích dat skončilo nejvíce bodů s hodnotou $Y = y$, rozhodnutí daného listu bylo y.
  - V případě spojité proměnné Y (např. věk, teplota, cena, doba trvání, ...) se může snadno stát, že každý z trénovacích bodů má jinou hodnotu.
- Jaké kritérium použijeme v hladovém algoritmu pro výběr nejlepšího příznaku k dělení?
  - V případě diskrétní proměnné jsme používali *entropii* resp. *gini index*.
  - Ty jsou ale v případě spojité proměnné nepoužitelné (rozmyslete si, jak by to vypadalo, zejm. jaké hodnoty by měla čísla p_i).
  - Pro spojitou vysvětlovanou proměnnou budeme muset najít jiné kritérium.

Jak strom určuje své rozhodnutí

- Předpokládejme, že už máme naučený strom a chceme najít predikovanou hodnotu Y pro nějaký datový bod s příznaky $X_1, \ldots, X_p$.
- Pomocí rozhodovacích pravidel v daném stromu a hodnot příznaků $X_1, \ldots, X_p$ se dostaneme do listu, kde při učení „skončilo“ šest datových bodů z trénovacího množiny dat.
- Předpokládejme, že vysvětlovaná proměnná Y pro těchto šest bodů měla hodnoty

$$\{10, 15, 20, 25, 30, 35\}.$$

- Jaké má být tedy rozhodnutí stromu pro body, které skončí v tomto listu?
- Obvykle se bere **průměr hodnot z listu**, v našem případě tedy 22.5.

Co použít místo entropie?

- V případě klasifikace se stromy rozhodovaly pomocí „hlasování“ v rámci jednotlivých listů. Minimalizace entropie měla zaručit, aby toto hlasování mělo co nejjednoznačnějšího vítěze.
- V případě regrese se strom rozhoduje tak, že v rámci listu spočítá průměr.
- Naší snahou by tedy mělo být, aby **hodnoty vysvětlované v rámci listu byly co nejblíže střední hodnoty**. Jak toto měřit?
- Známou mírou „odchylky od střední hodnoty“ je **MSE = mean squared error**, což je skoro stejná veličina jako (výběrový) **rozptyl** (angl. **sample variance**).

Co použít místo entropie? MSE!

- Představme si, že máme data s hodnotami vysvětlované proměnné

$$\{Y_1 = 10, Y_2 = 15, Y_3 = 20, Y_4 = 25, Y_5 = 30, Y_6 = 35\}.$$

- Průměr (tj. odhad střední hodnoty) této množiny je $\overline{Y} = 22.5$.
- Odhad MSE je číslo $\mathsf{MSE}(\mathbf{Y}) = (Y_1, \ldots, Y_N)$, což není nic jiného, než aritmetický průměr čtverců vzdáleností od průměru:

$$\mathsf{MSE}(\mathbf{Y}) = \frac{1}{N} \sum_{j=}^{N} (Y_j - \overline{Y})^2$$

pro naše data tedy

$$\mathsf{MSE}((10, 15, 20, 25, 30, 35)) = \frac{1}{6}((10 - 22.5)^2 + (15 - 22.5)^2 +$$
$$+(20 - 22.5)^2 + (25 - 22.5)^2 + (30 - 22.5)^2 + (35 - 22.5)^2) = 72.9.$$

Rozhodovací strom pro regresi

- Hladový algoritmus, který se používá při regresi, volí vrcholy rozhodovacího stromu tak, aby se minimalizovalo MSE.

- Funguje stejně jako algoritmus představený dříve: kvalitu rozdělení množiny $\mathcal{D}$ na podmnožiny $\mathcal{D}_L$ a $\mathcal{D}_R$ ale namísto entropie

$$H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R)$$

používá MSE

$$\text{MSE}(\mathcal{D}) - t_L \text{MSE}(\mathcal{D}_L) - t_R \text{MSE}(\mathcal{D}_R),$$

kde $t_L = \frac{\#\mathcal{D}_L}{\#\mathcal{D}}$ a $t_R = \frac{\#\mathcal{D}_R}{\#\mathcal{D}}$ a $\text{MSE}(\mathcal{D})$ je MSE spočítané pro hodnoty vysvětlované proměnné Y pro všechny body z $\mathcal{D}$.

Poznámky

- Místo MSE lze používat také MAE = Mean Absolute Error:

$$\mathrm{MAE}(\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \overline{Y}|,$$

  který místo čtverce vzdálenosti používá absolutní hodnotu.
- Rozhodovací stromy mají mnoho výhod:
  - Nenáročnost na přípravu dat: poradí si s kategorickými i spojitými příznaky, s chybějícími hodnotami, atp.
  - Jsou jednoduché a srozumitelné a učení je relativně rychlé.
  - Jsou dobře interpretovatelné a jejich rozhodnutí lze snadno rozklíčovat.
- Ale mají samozřejmě i nevýhody:
  - Jsou *nerobustní*: i drobná změna v trénovacích datech může znamenat zásadní změnu struktury výsledného stromu.
  - Většina implementací podporuje pouze binární stromy.
  - Najít optimální strom je NP-úplný problém.
  - Je snadné rozhodovací stromy přeučit.
- V praxi se typicky používají různá vylepšení námi uvedeného algoritmu konstrukce stromu pro regresi a klasifikaci jako např. **CART = Classification and Regression Trees** nebo **C4.5**, které obsahují prořezávání, pokročilejší zastavovací kritéria atd.

BI-ML1.21 přednáška 3

Daniel Vašata

FIT ČVUT

9. 10. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 9. října 2024 16:02.

Co bude v dnešní přednášce

- metoda nejbližších sousedů (KNearestNeighbors, kNN)
- různé míry vzdáleností
- kNN a normalizace dat
- prokletí dimenzionality

kNN: základní myšlenka

- Řešíme problém *supervizovaného učení*, kdy tedy chceme na základě p příznaků $X_1, \ldots, X_p$ predikovat hodnotu vysvětlované proměnné Y.
- Pro zjednodušení značení poskládáme příznaky do vektoru $\boldsymbol{X} = (X_1, \ldots, X_p)^T$, který budeme chápat jako náhodný vektor, a jednu jeho konkrétní realizaci budeme značit $\boldsymbol{x}$.
- Dále budeme jako $\mathcal{X}$ značit množinu obsahující všechny možné hodnoty příznaků, tj. $\boldsymbol{x} \in \mathcal{X}$, přičemž typicky $\mathcal{X} = \mathbb{R}^p$.
- Máme tedy trénovací množinu tvořenou N dvojicemi $(\boldsymbol{x}_1, Y_1), \ldots, (\boldsymbol{x}_N, Y_N)$.
- Základní (a velice jednoduchá) myšlenka kNN je následující:
  - Chceme predikovat hodnotu vysvětlované proměnné pro datový bod $\boldsymbol{x} \in \mathcal{X}$.
  - V trénovacích datech najdeme k bodů (k je zadaný hyperparametr), které mají od $\boldsymbol{x}$ nejmenší vzdálenost.
  - Predikci pak založíme na známých hodnotách vysvětlované proměnné pro těchto k bodů.
  - Jedná-li se o regresi (spojité Y), bereme průměr z hodnot pro tyto body.
  - Jedná-li se o klasifikaci, bereme nejčastější hodnotu mezi těmito body.

Nejbližší soused: jak ho najít

Klíčový je pojem vzdálenosti (resp. metriky):

Definice

Vzdálenost nebo také **metrika** na množině $\mathcal{X}$ je funkce $d : \mathcal{X} \times \mathcal{X} \to [0, +\infty)$ taková, že pro každé $x, y, z \in \mathcal{X}$ platí

i) $d(x, y) \geq 0$, a $d(x, y) = 0$ právě tehdy když $x = y$ - pozitivní definitnost,

ii) $d(x, y) = d(y, x)$ - symetrie,

iii) $d(x, y) \leq d(x, z) + d(z, y)$ - trojúhelníková nerovnost.

- Nejběžnější je volba **Eukleidovské vzdálenosti** nebo také L_2 vzdálenosti:

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 = d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2},$$

  pro dva body $\boldsymbol{x} = (x_1, \ldots, x_p)$ a $\boldsymbol{y} = (y_1, \ldots, y_p)$ z $\mathbb{R}^p$.

- Později si ukážeme i jiné možnosti.

kNN: učení vs. predikování

- U rozhodovacích stromů byla výpočetně náročná fáze učení, ale predikce už se konstruovaly velmi snadno a rychle (šlo jen o průchod nehlubokým stromem).
- Toto platí skoro o všech metodách pro supervizované učení: řádově více je náročné učení modelu než výpočet predikcí.
- U kNN je tomu naopak! Učení totiž vlastně neprobíhá: **trénovací data jsou sama o sobě naučeným modelem**.
- Co je naopak výpočetně náročné je predikce, tedy hledání nejbližších sousedů pro daný bod: vyžaduje to průchod všemi trénovacími daty a naměření vzdálenosti od každého trénovacího bodu.
- Hledání lze zrychlit tím, že si data jistým způsobem indexujeme (obvykle do jistého vyhledávacího stromu), potom se situace „znormální“ a predikování se zrychlí na úkor učení se (tj. tvorby indexu trénovacích dat).

kNN: hyperparametry

- Implementace kNN v `scikit-learn` je dostupná v balíčcích KNeighborsRegressor a KNeighborsClassifier.

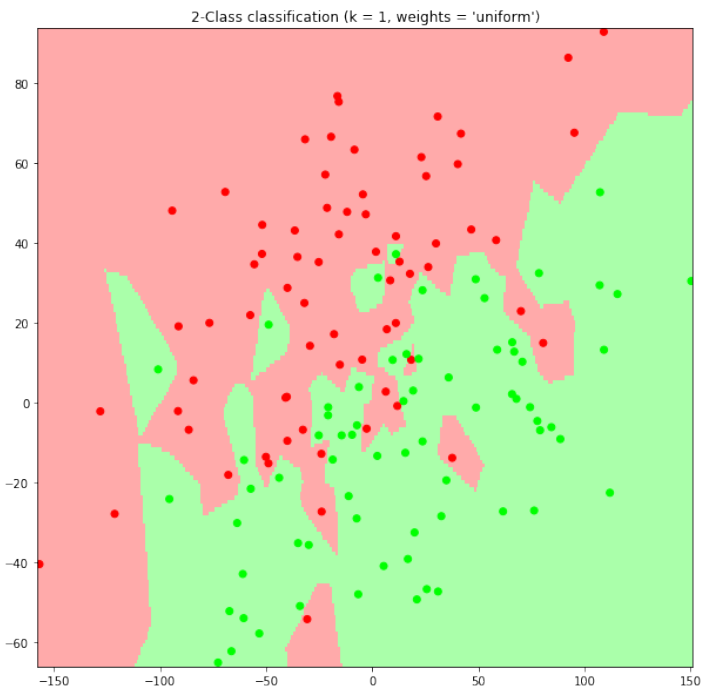
- V případě regrese i klasifikace jsou pro kNN smysluplné tři hyperparametry:
  - Číslo k určující počet hledaných nejbližších sousedů (`n_neighbors`).
  - Použitá vzdálenost; obecně funkce vracejícím dvěma datovým bodům číslo (`metric`).
  - Váhy nejbližších sousedů určující „sílu jejich hlasu“ při predikci (`weights`).

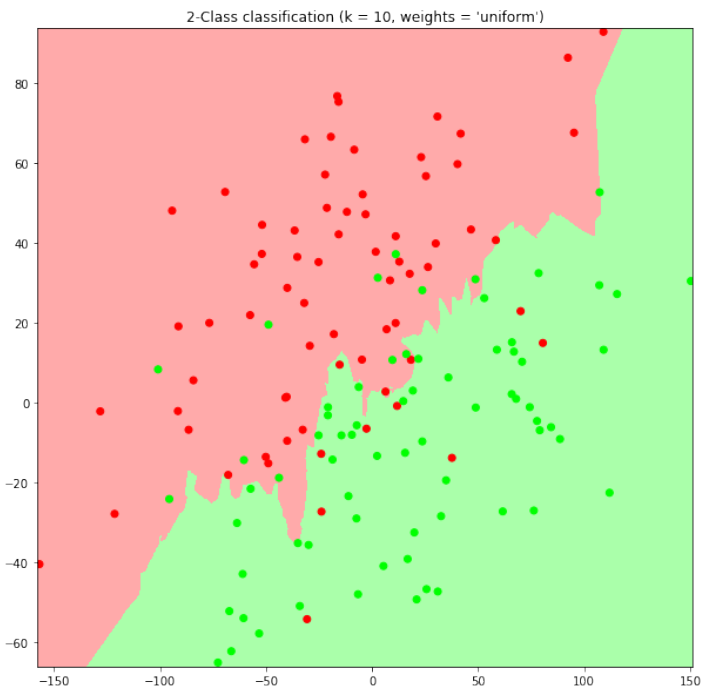
- Postupně si tyto tři hyperparametry projdeme.

- Implementace kNN v `scikit-learn` nabízí ještě další parametry, ale ty se už týkají spíše způsobu výpočtu a tvorby případného indexu, takže neovlivňují přímo tvar modelu a nejedná se tedy striktně řečeno o hyperparametry.

kNN: hyperparametr k (počet sousedů)

- Většinu modelů lze přeučit (tzv. *overfitting*). Např. u stromů to šlo snadno, stačilo vytvořit příliš hluboký strom s velkým množstvím listů.
- U kNN lze přeučení zabránit **zvýšením počtu sousedů**, které mají vliv na predikci.
- Jak vypadá přeučený kNN lze vidět na obrázku níže vlevo, kde je výsledek binární klasifikace pro $k = 1$, vpravo jsou vidět tatáž data pro $k = 10$.





kNN: míra vzdálenosti jako hyperparametr (1/2)

- Předpokládejme, že všechny naše příznaky jsou čísla; ideálně, že se jedná o spojité numerické příznaky.
- Potom lze jako vzdálenost vzít jakoukoli metriku definovanou na $\mathbb{R}^p$ (kde p je počet příznaků).
- Nejobvyklejší volbou jsou tzv. L_q metriky (také Minkovského q-metriky příp. q-normy), kde $q = 1, 2, \ldots$ Tyto jsou také přímo podporované v `scikit-learn`.
- Vzdálenost dvou bodů $\boldsymbol{x} = (x_1, \ldots, x_p)$ a $\boldsymbol{y} = (y_1, \ldots, y_p)$ z $\mathbb{R}^p$ dané L_q metrikou je rovna

$$\|\boldsymbol{x} - \boldsymbol{y}\|_q = d_q(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[q]{\sum_{i=1}^{p} |x_i - y_i|^q},$$

  speciálně pro $q = 1$ dostáváme

$$\|\boldsymbol{x} - \boldsymbol{y}\|_1 = d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{p} |x_i - y_i|.$$

- Eukleidova vzdálenost tak odpovídá $q = 2$, Manhattanská vzdálenost pak $q = 1$.

kNN: míra vzdálenosti jako hyperparametr (2/2)

- Vzdálenost si můžeme ale definovat v podstatě libovolně, jde jen o to, aby byla schopná vrátit jednoznačně určené číslo pro dva datové body (tj. dva vektory příznaků).
- Lze také definovat i vzdálenosti, které se v každé dimenzi chovají jinak: např. jedná-li se o příznak spojitý numerický, přispěje do vzdálenosti absolutní hodnotou rozdílu, jedná-li se např. o jméno, lze použít Levenshteinovu vzdálenost, příp. lze na podmnožinu příznaků použít třeba cosinovou vzdálenost
$$\frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2}.$$
- Volba sofistikované míry vzdálenosti může z jednoduchého kNN modelu udělat model složitý (a třeba i mocný). Obvykle také významně vzroste počet hyperparametrů, které určují, jak přesně daná vzdálenost vypadá (zejm. jakou váhu mají jednotlivé složky).
- Více se lze dočíst například zde: Similarity Measures for Categorical Data: A Comparative Evaluation (2008).

kNN: hyperparametr `weights`

- Představme si, že řešíme regresní problém a že jsme pro daný bod $\boldsymbol{x}$ našli nejbližší sousedy $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ s hodnotami vysvětlované proměnné $y_1, \ldots, y_k$.
- Predikci pro $\boldsymbol{x}$ můžeme zvolit klasicky jako „průměr nejbližších sousedů“, tedy

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i.$$

- Často se ale volí vážený průměr

$$\hat{y} = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i},$$

  kde w_i jsou nezáporné váhy jednotlivých bodů.
- Tyto váhy se obvykle volí tak, že *klesají se vzdáleností*; např. v `scikit-learn` je možné zvolit `weights=distance`, které nastaví

$$w_i = \frac{1}{d(\boldsymbol{x}, \boldsymbol{x}_i)}.$$

Různé váhy stejných příznaků

- Na rozdíl od rozhodovacích stromů je metoda kNN náročná na přípravu dat a je také citlivá na typy jednotlivých příznaků.
- Předpokládejme prozatím, že všechny naše příznaky jsou spojité numerické.
- Představme si, že dva z příznaků jsou dva rozměry půdorysu bazénu (např. určujeme prodejní cenu domu): x_1 je uveden v centimetrech a x_2 v metrech.
- Nyní předpokládejme, že máme dva domy se čtvercovými bazény o straně 5 a 6 metrů. Jaký bude příspěvek příznaků x_1 a x_2 do Eukleidovské vzdálenosti příslušných dvou datových bodů?

$$(500 - 600)^2 + (5 - 6)^2 = 100^2 + 1 = 10001.$$

- Přestože se tedy tyto příznaky liší o stejnou vzdálenost, změna v x_2 je při měření vzdálenosti datových bodů (reprezentujících domy) naprosto zanedbatelná vůči změnám v x_1.

Často nedosažitelná souměřitelnost příznaků

- Předchozí příklad s rozměry bazénů lze snadno vyřešit: budeme oba rozměry měřit v metrech. Ale většinou to není tak jednoduché.
- U domu můžeme mít například příznaky určující
  - plochu v bazénu v metrech čtverečních,
  - velikost koupelny také v metrech čtverečních,
  - velikost televize danou úhlopříčkou v palcích,
  - počet oken.
- Co s takovým případem? Je řešením převést plochu televize na metry čtvereční? Je zvetšení kuchyně o 2 čtvereční metry to samé, jako stejné zvětšení bazénu? A co teprve počet oken?
- V některých případech jsou příznaky jen těžko porovnatelné a v podstatě nelze najít nějakou univerzální míru.
- Toto lze složitě řešit pomocí sofistikovaných metrik plných hyperparametrů anebo jednoduše (ale často naivně) pomocí **normalizace dat**.

Normalizace dat (1/2)

- Jednoduchou metodou, jak zabránit nejextrémnějším úletům způsobeným pestrou škálou příznaků a jejich významů je normalizace každého příznaku do intevalu $[0, 1]$, která se nazývá **min-max normalizace**.
- Postup je následující: pro daný příznak najdeme jeho minimální a maximální hodnotu v trénovacích datech: $\min_x, \max_x$ a pak hodnotu x_i tohoto příznaku pro i-tý datový bod nahradíme

$$x_i \leftarrow \frac{x_i - \min_x}{\max_x - \min_x}.$$

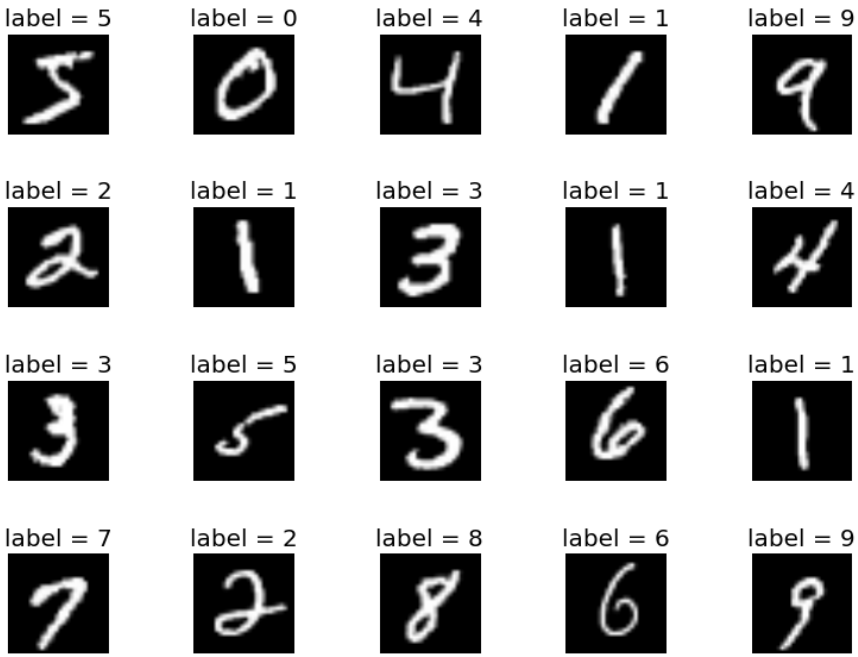
- Tím docílíme toho, že všechny hodnoty budou z intervalu $[0, 1]$.
- Změny v jednotlivých příznacích se pak vlastně porovnávají s maximálním možným rozdílem hodnot v trénovacích datech a tím se docílí jisté omezené souměřitelnosti.
- Další, často používanou normalizací je **standardizace**:

$$x_i \leftarrow \frac{x_i - \bar{x}}{\sqrt{s_x^2}},$$

kde $\bar{x} = \frac{1}{n}\sum_i x_i$ je výběrový průměr a $s_x^2 = \frac{1}{n-1}\sum_i (x_i - \bar{x})^2$ je výběrový rozptyl daného příznaku.

Normalizace dat (2/2)

- Předchozí postup *normalizace po příznacích* ale rozhodně není univerzální.
- Uvažujme známý MNIST dataset s rukou psanými číslicemi zachycenými na černobílých fotkách o rozměrech 28×28 pixelů se stupněm šedi 0 až 255.



- Datový bod tedy obsahuje $28 \times 28 = 784$ příznaků s hodnotami od 0 (černá) do 255 (bílá).
- Je v tomto případě normalizace po příznacích vhodná?

Normalizace dat (2/2)

- Vhodná moc není, jednak kvůli příznakům (pixelům), které jsou ve všech obrázcích nulové (černé), ale i kvůli těm, které jsou na pár obrázcích trochu šedivé.
- Zde vlastně není normalizace vůbec nutná, neboť všechny příznaky jsou co do měřítka a významu totožné.
- Lze použít normalizaci do intervalu $[0, 1]$, ovšem nikoli po příznacích:

$$x_i \leftarrow \frac{x_i - \min_X}{\max_X - \min_X},$$

  kde $\min_X$ a $\max_X$ jsou maxima nikoli v daném sloupci (příznaku), ale ve všech sloupcích.
- Pro MNIST data je $\min_X = 0$ a $\max_X = 255$.

Obecně je normalizace dat velké a složité téma, ke kterému neexistuje nějaký univerzální správný přístup.[1]

[1] Jak to tak chodí.

kNN a nominální příznaky

- kNN se bez použití speciálních metrik neumí dobře vypořádat s nominálními příznaky.
- Jako příklad uvažujme příznak u domu (zase dům prodáváme), který určuje číslo části Prahy (1 až 22).
- Zde očividně nemá cenu měřit velikost číselného rozdílu.
- Lze buď modifikovat metriku tak, aby za tento příznak vracela nulu, pokud má pro dva datové body stejnou hodnotu, jinak aby vracela jedničku. Rozlišovala by tak jenom dva stavy: shodná/různá hodnota příznaku.
- Podobný (ale ne úplně stejný, to si rozmyslete) výsledek dostaneme, pokud použijeme *one-hot encoding* a *dummy příznaky* (kterých bude 22).
- Pro ordinární příznaky už může použití klasických číselných rozdílů dávat smysl, ale obecně je to problematické.

Prokletí dimenzionality

- **Prokletí dimenzionality** (angl. **the curse of dimensionality**), je pojem, který odkazuje na některé problémy objevující se v případě vysokého počtu příznaků, kdy jsou datové body prvky mnohadimenzionálního prostoru.
- Tyto problémy hrají svoji roli v mnoha metodách a proto jsou **redukce dimenzionality** a výběr příznaků (angl. **feature selection**) jedním ze zásadních témat při zpracování dat.
- S kNN jsou spojeny zejména dva efekty způsobené vysokou dimenzí dat:
  - Data se zvyšováním dimenze řídnou a navzájem se vzdalují. Pro zachování stejné hustoty pro vyšší dimenzi by bylo nutné řádově navýšit počet datových bodů, což není obvykle možné.
  - S rostoucí dimenzí se pro klasické metriky zmenšují rozdíly mezi vzdálenými a blízkými body.
- První efekt si demonstrujeme na příkladu.

Prokletí dimenzionality: řídnutí bodů

Představme si, že máme d-dimenzionální jednotkovou krychli (hyperkrychli), tedy oblast

$$[0,1] \times [0,1] \times \cdots \times [0,1] \subset \mathbb{R}^d$$

a v ní máme náhodně rozhozeno (dle uniformního rozdělení) 1000 bodů.
Otázka: Jak velkou musíme mít „podkrychli“, aby obsahovala v průměru 10 bodů (tj. pokrývala jednu setinu objemu)?

- Pro jednodimenzionální krychli (tj. úsečku) je to úsečka o délce 0,01.
- Pro dvoudimenzionální čtverec je to čtverec o straně 0,1.
- Ve třídimenzionální prostoru je to krychle o hraně a, kde $a^3 = 1/100$, tedy

$$a = \sqrt[3]{\frac{1}{100}} \approx 0,215$$

- V dimenzi d je to pak krychle o straně

$$a = \sqrt[d]{\frac{1}{100}}$$

  pro $d = 10$ je $a \approx 0,63$ a pro $d = 50$ je $a \approx 0,91$.
- Z pohledu strojového učení to znamená, že hustota trénovacích bodů s rostoucí dimenzí klesá. Body jsou od sebe velmi vzdálené.

BI-ML1.21 přednáška 4

Daniel Vašata

FIT ČVUT

16. 10. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 16. října 2024 07:39.

Co bude v dnešní přednášce

- Evaluace modelů obecně
- Připomenutí trénování, validace a testování
- Křížová validace
- Evaluace regrese
- Evaluace klasifikace

Úvod do evaluace modelů

- Jednou z hlavních výzev strojového učení je, aby natrénovaný model dokázal dobře fungovat i na nových vstupech, které doposud neviděl.
- Této schopnosti říkáme schopnost **generalizace**.
- V typickém scénáři máme několik kandidátů na finální model a potřebujeme mezi nimi vybrat ten nejlepší.
- K tomuto je třeba mít nějakou kvantitativní míru výkonnosti modelu.
- Zaměřme se na problematiku měření výkonnosti modelů v rámci supervizovaného učení.
- V tomto případě se může zdá volba míry přímočará (přesnost pro klasifikaci, MSE pro regresi), ale ve skutečnosti to není tak jednoduché a je třeba správně zvolit metriku, která odpovídá žádoucímu chování modelu.
- Například u regresní úlohy můžeme chtít preferovat model, který nedělá velké chyby, ale poměrně často dělá chyby menší. Anebo můžeme preferovat model, který vetšinou nedělá ani malé chyby, ale občas udělá hodně velkou.

Ztrátová funkce

- Uvažujme vysvětlovanou proměnnou Y a vektor příznaků (vstupů) $\boldsymbol{X} = (X_1, \ldots, X_p)^T$.
- Model predikuje $\hat{Y} \equiv \hat{Y}(\boldsymbol{X})$, které je funkcí $\boldsymbol{X}$.
- Chybu predikce Y pomocí $\hat{Y}$ obecně měříme pomocí tzv. **ztrátové funkce** (angl. **loss function**) L.
- Ztrátová funkce vhodným způsobem měří, jak dobře daný model predikuje konkrétní hodnotu.
- Typickou volbou v případě regresní úlohy je kvadratická ztrátová funkce měřící **kvadratickou chybu** (angl. **squared error**)

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2$$

nebo L_1 ztrátová funkce měřící **absolutní chybu** (angl. **absolute error**)

$$L(Y, \hat{Y}) = |Y - \hat{Y}|.$$

- Kvadratickou chybu využívá třeba lineární regrese.

Ztrátová funkce binární klasifikace

- U binární klasifikace model často odhaduje pravděpodobnost
$$\hat{p} = \hat{\mathrm{P}}(Y = 1 | \boldsymbol{X} = \boldsymbol{x}).$$
- Poznamenejme, že to umí například rozhodovací strom, který může pro daný list, do kterého padne $\boldsymbol{x}$, vrátit $\hat{p}$ jako relativní počet reprezentantů třídy 1 z trénovací množiny v tom listu.
- Finální predikce vysvětlované proměnné je potom
$$\hat{Y} = \begin{cases} 1 & \text{když } \hat{p} > 1/2, \\ 0 & \text{jinak.} \end{cases}$$
- Typická ztrátová funkce, která se v takovém případě používá je (angl. **binary cross-entropy loss**)
$$L\big(Y, \hat{p}\big) = -Y \log \hat{p} - (1 - Y) \log(1 - \hat{p}).$$
- To reálně znamená
$$L\big(1, \hat{p}\big) = -\log \hat{p} \quad \text{a} \quad L\big(0, \hat{p}\big) = -\log(1 - \hat{p}).$$
- Jak uvidíme později v semestru, tato ztrátová funkce odpovídá tomu, co se děje u logistické regrese.

Trénovací chyba

- Při učení se snažíme minimalizovat chybu predikce měřenou pomocí průměrné hodnoty ztrátové funkce L na trénovacích datech

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} L\big(Y_i, \hat{Y}(\boldsymbol{x}_i)\big),$$

  kde trénovací množina je N dvojic $(Y_i, \boldsymbol{x}_i)$.
- Tato průměrná hodnota se nazývá **trénovací chyba** (angl. **training error**) a občas se značí jako $\overline{\mathrm{err}}_{\mathsf{train}}$. Někdy se jí také říká trénovací ztráta (angl. **training loss**).
- U regresní úlohy, použití kvadratické ztrátové funkce vede k minimalizaci **střední kvadratické chyby** (angl. **mean squared error**)

$$\mathrm{MSE}_{\mathsf{train}} = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2,$$

  což je ekvivalentní minimalizaci RSS v lineární regresi.

Trénovací chyba pokračování

- Použití L_1 ztrátové funkce vede na minimalizaci **střední absolutní chyby** (angl. **mean absolute error**)

$$\mathrm{MAE}_{\mathsf{train}} = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|.$$

- U binární klasifikace, použití předchozí ztrátové funkce vede k minimalizaci binární relativní entropie (angl. **binary cross-entropy**)

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\Big[Y_i \log \hat{p}(\boldsymbol{x}_i) + (1 - Y_i)\log(1 - \hat{p}(\boldsymbol{x}_i))\Big].$$

- Časem uvidíme, že až na mínus před sumou se jedná přesně o logaritmus věrohodnostní funkce, který budeme maximalizovat u logistické regrese. S mínusem a minimalizací se tedy jedná o totožnou úlohu.

Učení modelu

- Během procesu učení modelu se zafixovanými hyperparametry se snažíme najít hodnoty parametrů, které minimalizují trénovací chybu.
- U některých modelů, jako např. lineární regrese, lze toto řešení najít explicitně.
- Pro většinu modelů to ale nelze a používají se různé iterativní metody (typicky založené na gradientním sestupu), které se snaží nalézt lokální minimum.
- Jakmile je model natrénován, zajímá nás jeho schopnost generalizace.
- Podívejme se nejprve na odhadování výkonnosti modelu na nových datech pomocí ztrátové funkce. Pozdeji si ukážeme i jiné míry měřící výkonnost, resp. schopnost generalizace.

Testovací chyba

- **Testovací chyba** (angl. **test error**), také někdy nazývaná **generalizační chyba** (angl. **generalization error**), je definována jako střední hodnota chyby na novém vstupu $\boldsymbol{X}$ podmíněná danými trénovacími daty,

$$\mathrm{Err}_{\mathcal{D}} = \mathrm{E}\left(L\Big(Y, \hat{Y}(\boldsymbol{X})\Big)|\mathcal{D}\right),$$

  kde $\mathcal{D} = \Big((Y_1, \boldsymbol{x}_1), \ldots, (Y_N, \boldsymbol{x}_N)\Big)$ značí trénovací data.

- Testovací chybu můžeme odhadnout výběrovým průměrem

$$\overline{\mathrm{err}}_{\mathsf{test}} = \frac{1}{N_{\mathsf{test}}} \sum_{i=1}^{N_{\mathsf{test}}} L\Big(Y_i, \hat{Y}(\boldsymbol{x}_i)\Big),$$

  změřeným na testovací datech $\Big((Y_1, \boldsymbol{x}_1), \ldots, (Y_{N_{\mathsf{test}}}, \boldsymbol{x}_{N_{\mathsf{test}}})\Big)$, které byla získány nezávisle na trénovacích datech $\mathcal{D}$.

- Poznamenejme, že $\overline{\mathrm{err}}_{\mathsf{test}}$ podmíněno trénovacími daty je nestranný odhad $\mathrm{Err}_{\mathcal{D}}$, t.j.

$$\mathrm{E}\left(\overline{\mathrm{err}}_{\mathsf{test}}\,|\mathcal{D}\right) = \mathrm{Err}_{\mathcal{D}}\,.$$

- Nejobecnější mírou schopnosti modelu generalizovat je **očekávaná testovací chyba** nebo taky **očekávaná chyba predikce** (angl. **expected test error**, **expected prediction error**) definovaná jako střední hodnota testovací chyby vzhledem k náhodnému výběru trénovací množiny

$$\mathrm{Err} = \mathrm{E}\left(\,\mathrm{Err}_{\mathcal{D}}\,\right) = \mathrm{E}\,L\Big(Y, \hat{Y}(\boldsymbol{X})\Big).$$

Vyhodnocovací scénáře

Chceme-li natrénovat model a pak odhadnout jeho testovací chybu, musíme mít k dispozici trénovací data a nezávislá testovací data.

- Kromě toho v mnoha případech máme více kandidátů na finální model a potřebujeme z nich vybrat ten nejlepší.
- Typicky se jedná o situaci, kdy máme celou třídu modelů závisejících na nějaké sadě hyperparametrů a chceme najít jejich hodnoty tak, aby testovací chyba finálně vybraného modelu byla co nejmenší.
- Z pohledu evaluace tak máme dva různé úkoly:
  - Výběr modelu - odhadnout výkonnost různých modelů za účelem výběru nejlepšího.
  - Ohodnocení modelu - odhadnout testovací chybu finálního modelu.
- Protože výběr modelu vlastně spadá do procesu trénování (vybereme model, který se nejlépe přizpůsobí datům) nesmíme použít stejná data pro výběr finálního modelu i pro ohodnocení tohoto finálního modelu.

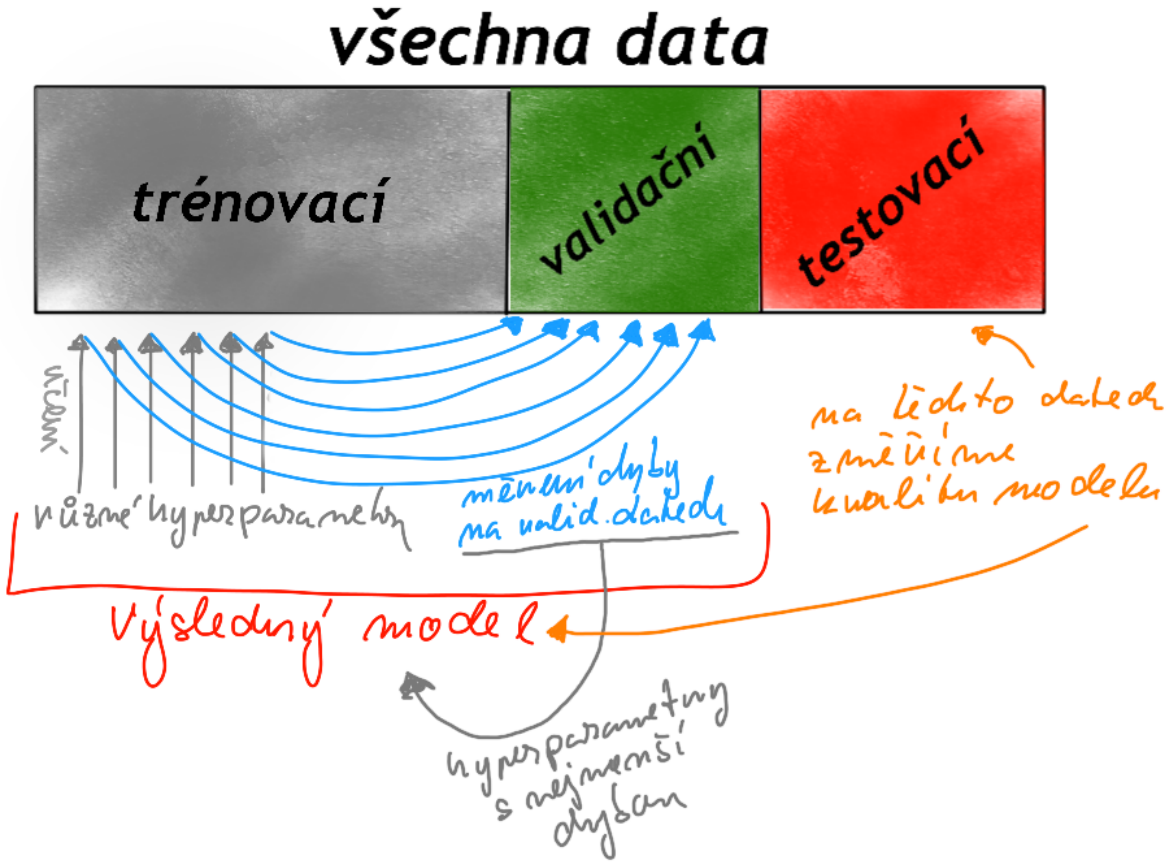
Trénování, validace a testování

Když máme **dostatek dat**, rozdělíme vstupní data na tři části: **trénovací** a **validační** a **testovací**.

- **Trénovací část** použijeme pro trénování konkrétních modelů se zafixovanými hyperparametry.
- **Validační část** použijeme k ohodnocení daného modelu a porovnání s ostatními modely. Takto vybereme nejlepší sady hyperparametrů a případně porovnáme nejlepší modely z různých tříd. Finální model tedy vybíráme na základě validační množiny.
- **Testovací část** použijeme až v závěrečné fázi, když už máme vybraný a natrénovaný finální model. Tím získáme odhad testovací chyby, kterou můžeme očekávat na nových datech.

Tento způsob práce s testovací množnou se nazývá **hold-out**.

Trénování, validace a testování - obrázek



Trénování, validace a testování - chyby

- V některých případech se objevuje nesprávné použití těchto tří částí datasetu.
- Validační část je (správně) použita k ladění hyperparametrů v rámci jedné třídy modelů.
- Potom jsou ale modely s nejlepšímy hyperparametry z různých tříd (např. KNN a rozhodovací strom) porovnávány (špatně) na základě testovacích dat!
- Protože **výběr nejlepšího modelu je součást trénování**, finální model je potom vybrán na základě všech tří částí datasetu.
- Tudíž je finální ohodnocení na testovacích datech příliš optimistické.
- Testovací množinu musíme vždy oddělit co nejdříve (ještě před předzpracováním) a držet ji striktně bokem pouze pro finální evaluaci.

Trénování, validace a testování - rizika

- Evaluace pomocí validačních nebo testovacích dat dává rozumné výsledky pouze pokud trénovací, validační a testovací data pochází ze stejného rozdělení a když tam nejsou žádné procesně vnesené odlišnosti (např. že by data byla nějakým způsobem uměle uspořádaná a před rozdělení by se neprovedla náhodná permutace).
- V mnoha aplikacích prediktivního modelování se systém, který sledujeme a chceme predikovat, v čase vyvíjí (je tzv. „nestacionární“). To může přinést systematické odlišnosti mezi trénovacími a reálnými daty v budoucnosti.
- Příkladem může být predikce cen na burze, kdy data za minulých 5 let typicky nebudou odpovídat současnosti.
- Pokud očekáváme takovéto chování, je dobré, aby testovací (a případně i validační) data správně reflektovala chronologické řazení a příslušné odhady chyb tak mohli reálně ukazovat tento posun v čase. V takovém případě tedy data před rozdělením typicky nepermutujeme.
- U vícekrokového modelování (kdy např. doplňujeme chybějící hodnoty nějakým modelem, vybíráme příznaky atd.) musíme validační a testovací data oddělit již na začátku a pak na ně aplikovat všechny tyto kroky „naučené“ na trénovacích datech.

Křížová validace

- Když nemáme dostatek dat, nebývá rozumné dělit je na trénovací, validační a testovací část.
- Uvažujme nejprve scénář, kdy si testovací data dopředu oddělíme a jde nám pouze trénování a výběr nejlepšího modelu.
- V takovém případě můžeme k výběru nejlepšího modelu použít techniku **křížové validace** (angl. **cross-validation**).
- Základní metodou je tzv. k-násobná křížová validace (angl. k-**fold cross-validation**), kde $2 \leq k \leq N$:
  - Trénovací data $\mathcal{D}$ náhodně rozdělíme na k *podobně* velkých částí $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
  - Pro každé $j = 1, \ldots, k$ model s danými hodnotami hyperparametrů natrénujeme na datech z množiny $\left(\cup_{i=1}^{k} \mathcal{D}_i\right) \setminus \mathcal{D}_j$.

Trénovací $\mathcal{D}_1$	Testovací $\mathcal{D}_2$	Trénovací $\mathcal{D}_3$	Trénovací $\mathcal{D}_4$	Trénovací $\mathcal{D}_5$

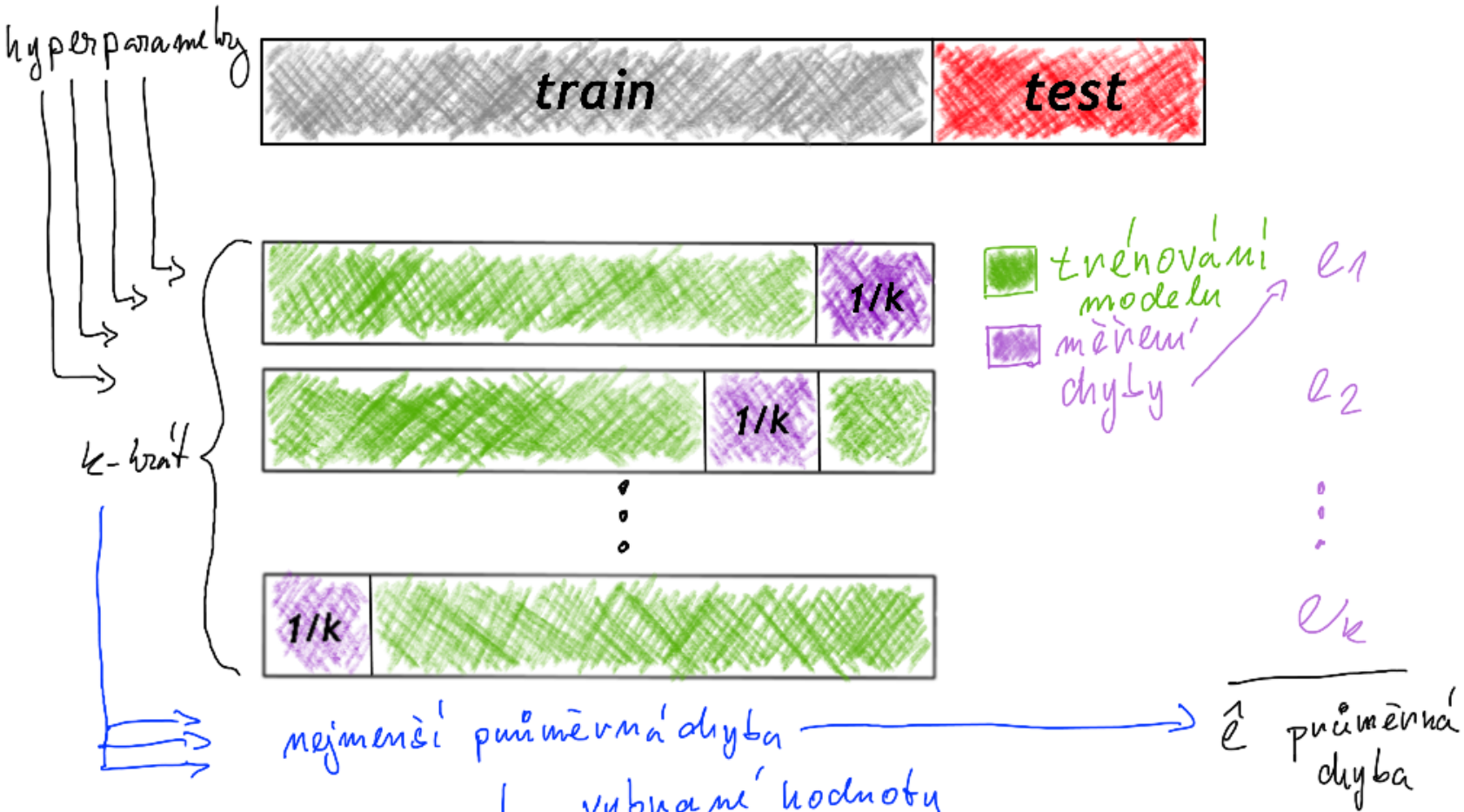
  - Na množině $\mathcal{D}_j$ odhadneme jeho chybu jako e_j.
  - Nakonec vrátíme průměrnou „cross-validační“ chybu

$$\hat{e} = \frac{1}{k} \sum_{i=1}^{k} e_i.$$

Křížová validace - pokračování

- Tento proces zopakujeme pro všechny zkoumané hodnoty hyperparametrů a na závěr vybereme jako nejlepší volbu ty hodnoty, které vedly k nejmenší cross-validační chybě.
- Abychom pak získali finální natrénovaný model (ten v tuhle chvíli nemáme), pro vybrané hodnoty hyperparametrů model znovu natrénujeme na celé trénovací množině $\mathcal{D}$!
- Typické volby k jsou 5 až 10.
- V extrémním případě, kdy se k rovná počtu trénovacích dat, mluvíme o **leave-one-out cross-validation**: trénuje se na celé trénovací množině bez jediného bodu, na kterém se pak měří výsledná chyba.

Křížová validace: jak to funguje (obrázek)



Křížová validace - diskuse

- Křížová validace může být neúnosně výpočetně náročná a nemůže tak vždy nahradit strategii s jedinou validační množinou.
- Pro fixní model je cross-validační chyba odhadem očekávané testovací chyby Err a nikoliv testovací chyby $\mathrm{Err}_{\mathcal{D}}$, což může být matoucí.
- Když máme opravdu málo dat a nechceme ani oddělovat testovací množinu, můžeme křížovou validaci použít dvoustupňově a současně tak vybrat nejlepší model i odhadnout jeho výkonnost pomocí očekávané testovací chyby. V tomto případě vnitřní křížovou validaci používáme na výběr nejlepšího modelu a vnější na odhad očekávané chyby.
- Výsledná vnější cross-validační chyba ale odpovídá očekávané chybě celé procedury pro výběr nejlepšího modelu, nikoliv chybě konkrétního modelu (který nám z toho ani nevypadne) a my tu proceduru pak musíme na celých datech provést a získat tak nejlepší model pro predikci (vizte [Cawley, Talbot (2010)]).

Strategie přípravy finálního modelu

- Při využití testovací množiny (hold-out přístup) získáme odhad testovací chyby nějakého nejlepšího modelu, který máme v tu chvíli i natrénovaný.
- U dvoustupňové křížové validace, získáme odhad očekávané testovací chyby procedury pro výběr nejlepšího modelu, ale nejlepší model teprve musíme získat.
- V tomto případě a i v tom prvním, pokud nemáme moc dat, má smysl sestrojit finální model úplně odznova a využít k tomu celá data.
  - Při hold-out přístupu s validační množinou tedy rozdělíme celý dataset pouze na 2 části - trénovací a validační. Různé modely (hyperparametry) pak trénujeme na trénovací části a pomocí validační části vybereme nejlepší model (hyperparametry). Dokonce tento model můžeme ještě znovu natrénovat na celých datech.
  - Při hold-out přístupu s křížovou validací nebo při dvoustupňové křížové validaci vezmeme celý dataset a pomocí křížové validace vybereme nejlepší model (hyperparametry). Ten pak na celých datech natrénujeme.
- Ve žádném ze scénářů si neodložíme data pro měření výkonnosti finálního modelu.
- Jako odhad této výkonnosti použijeme odhad (očekávané) testovací chyby z předchozího kroku (hold-out s validační množinou, s křížovou validací nebo dvoustupňová křížová validace)

Vyhodnocování regrese (1/2)

Nejobvyklejší volba kvadratické chyby jako ztrátové funkce vede na evaluaci pomocí **střední kvadratické chyby** (angl. **mean squared error**)

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2.$$

Tato míra penalizuje především velké odchylky a je velmi citlivá na odlehlé hodnoty.

Podívejme se na další vyhodnocovací míry, které se soustředí i na jiné aspekty predikce:

- **Root mean squared error** - odpovídá nelineárně přeškálovanému MSE:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}.$$

  Má stejné vlastnosti, akorát jednotky jsou stejné, jako jednotky vysvětlované proměnné.

Vyhodnocování regrese (2/2)

- **Root mean squared logarithmic error** - pro nezáporné hodnoty vysvětlované proměnné:

$$\mathrm{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log Y_i - \log \hat{Y}_i)^2}$$

  Soustředí se na relativní míru odchylek - tj. pro malé hodnoty řeší i malé odchylky, pro velké hodnoty pouze ty velké. Je méně citlivá na odlehlé hodnoty.

- **Mean absolute error** - odchylky se skládají lineárně:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|.$$

  Méně citlivá k odlehlým hodnotám než MSE.

- **Koeficient determinace** - R^2 (koeficient „R kvadrát“) vyjadřuje, jaký podíl variability cílové proměnné model vysvětluje:

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{SST}},$$

  kde

$$\mathrm{RSS} = \sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \quad \text{a} \quad \mathrm{SST} = \sum_{i=1}^{N}(Y_i - \bar{Y})^2.$$

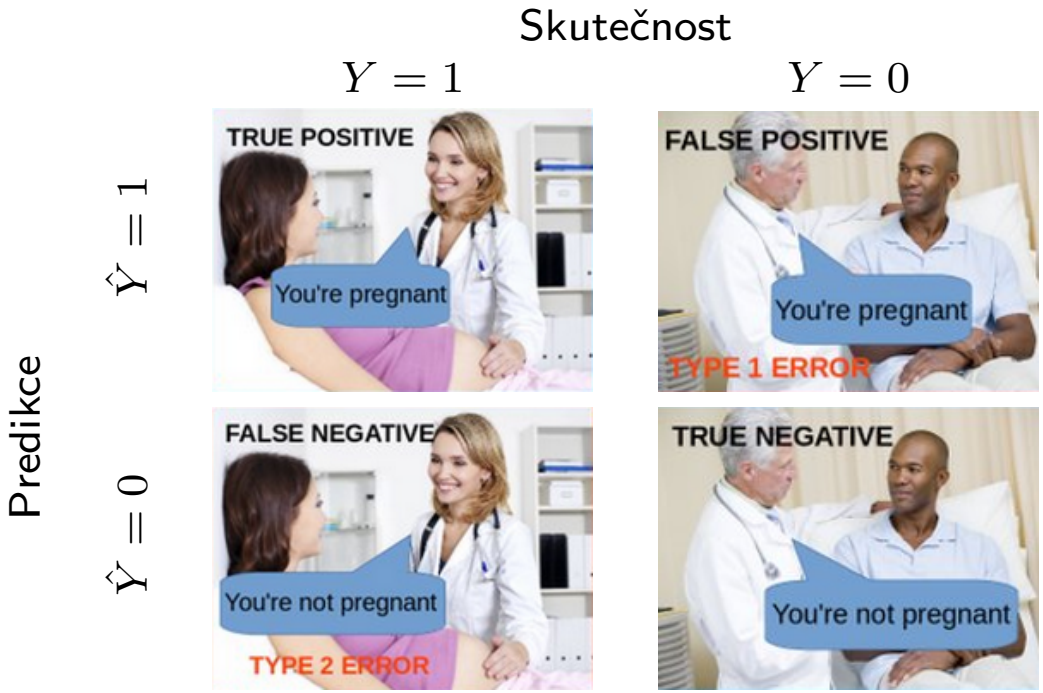
  (RSS - residual sum of squares, SST - total sum of squares)

Evaluace klasifikace - úvod

Při klasifikaci se chyba měřená pomocí ztrátové funkce pro rozumnou evaluaci příliš nehodí. Typicky se totiž jedná o relativní entropii, jejíž číselné hodnoty jsou těžko interpretovatelné.

Nejčastěji se vyhodnocování klasifikačních modelů provádí pomocí měr odvozených z tzv. **matice záměn** (angl. **confusion matrix**), což je matice četností různých predikovaných hodnot $\hat{Y}_i$ proti různým skutečným hodnotám Y_i.

Zaměřme se pouze na binární klasifikaci.



[zdroj: univ-angers.fr]

Matice záměn

Matice záměn spolu s řádkovými a sloupcovými součty:

		Skutečnost		
		$Y = 1$	$Y = 0$	$\sum$
Predikce	$\hat{Y} = 1$	TP	FP	$\hat{N}_+ = \mathrm{TP} + \mathrm{FP}$
	$\hat{Y} = 0$	FN	TN	$\hat{N}_- = \mathrm{FN} + \mathrm{TN}$
	$\sum$	$N_+ = \mathrm{TP} + \mathrm{FN}$	$N_- = \mathrm{FP} + \mathrm{TN}$	$N = \mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}$

V matici záměn jsou následující četnosti:

- **True positive** - TP - kolikrát model správně predikoval $Y = 1$.
- **False positive** - FP - kolikrát model špatně predikoval $Y = 0$.
- **False negative** - FN - kolikrát model špatně predikoval $Y = 1$.
- **True negative** - TN - kolikrát model správně predikoval $Y = 0$.

Řádkovými součty $\hat{N}_+$, resp. $\hat{N}_-$ jsou počty bodů, kde model predikoval 1, resp 0.

Sloupcovými součty N_+, resp. N_- jsou počty bodů ve třídě 1, resp 0.

Ideální predikce vede k tomu, že FN i FP jsou rovny 0.

Matice záměn

Z matice záměn můžeme odvodit následující míry, které odpovídají odhadům podmíněných pravděpodobností $\mathrm{P}(\hat{Y} = \hat{y}|Y = y)$:

	$Y = 1$	$Y = 0$
$\hat{Y} = 1$	$\mathrm{TPR} = \frac{\mathrm{TP}}{N_+}$	$\mathrm{FPR} = \frac{\mathrm{FP}}{N_-}$
$\hat{Y} = 0$	$\mathrm{FNR} = \frac{\mathrm{FN}}{N_+}$	$\mathrm{TNR} = \frac{\mathrm{TN}}{N_-}$

Pro tyto míry se používá několik různých označení:

- **True positive rate** (TPR) se také nazývá **sensitivita** nebo **recall** nebo **hit rate**.
- **False positive rate** (FPR) se také nazývá **false alarm rate** nebo **type I error rate**.
- **False negative rate** (FNR) se také nazývá **miss rate** nebo **type II error rate**.
- **True negative rate** (TNR) se také nazývá **specificita** nebo **selektivita**.

Kromě toho se někdy používají i odhady obrácených pravděpodobností $\mathrm{P}(Y = y|\hat{Y} = \hat{y})$.

Konkrétně především **precision** nebo také **positive predictive value**, což je odhad $\mathrm{P}(Y = 1|\hat{Y} = 1)$:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\hat{N}_+}.$$

Nepoužívanější evaluační míry binární klasifikace

Podívejme se nyní na dvě důležité evaluační míry, které můžeme odvodit z matice záměn.

- **Přesnost** - odhad $\mathrm{P}(\hat{Y} = Y)$:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{N}.$$

  Jedná se o suveréně nejpoužívanější míru.

- Přesnost není příliš vhodná, pro nevybalancované datasety, kde $\mathrm{P}(Y = 1)$ nebo $\mathrm{P}(Y = 0)$ je velmi malé. V takovém případě bude přesnost vysoká, jakmile model zvládne správně predikovat majoritní třídu.

- **F1 score** - harmonický průměr precision $\mathrm{P}(Y = 1|\hat{Y} = 1)$ a recall $\mathrm{P}(\hat{Y} = 1|Y = 1)$

$$F_1 = \frac{2}{1/\mathrm{PPV} + 1/\mathrm{TPR}} = 2\frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}}.$$

  Tato míra je užitečná především pro nevybalancované datasety, kde $\mathrm{P}(Y = 1)$ je velmi malá.

Finální predikce v binární klasifikaci

- Podívejme se na úlohu binární klasifikace, kdy model v bodě $\boldsymbol{X}$ odhaduje pravděpodobnost $p(\boldsymbol{X}) = \mathrm{P}(Y = 1|\boldsymbol{X})$.
- Při klasickém přístupu (např. v logistické regresi) se predikce Y získá porovnáním této pravděpodobnosti s číslem $1/2$

$$\hat{Y} = \mathbb{1}_{\hat{p}(\boldsymbol{X})>0.5}$$

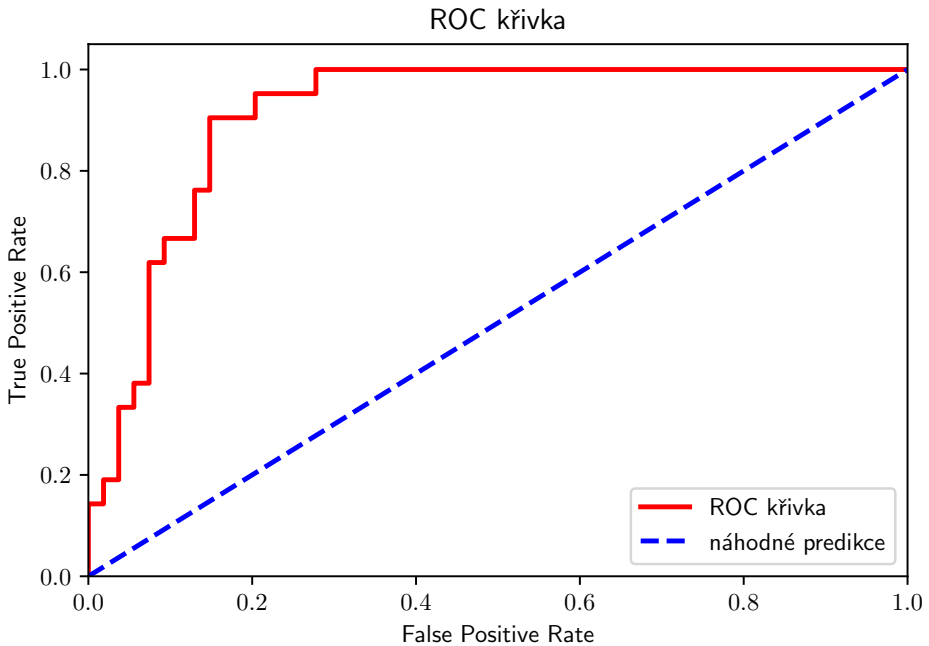
  tj. pokud je pravděpodobnost větší než 0.5 predikujeme třídu 1, v opačném případě predikujeme třídu 0.
- Tento způsob znamená, že predikujeme třídu s vyšší pravděpodobností.
- Ve skutečnosti ale můžeme zobecnit toto rozhodovací pravidlo zavedením nového hyperparametru $\tau \in [0, 1]$ a modifikací predikce tak, že

$$\hat{Y}_\tau \equiv \hat{Y}_\tau(\boldsymbol{X}) = \mathbb{1}_{\hat{p}(\boldsymbol{X})>\tau}.$$

- Platí tedy $\hat{Y} = \hat{Y}_{0.5}$.
- Pro každou hodnotu τ získáme trochu jiné predikce. Vždy ale platí, že pokud $\hat{Y}_\tau = 0$, potom určitě $\hat{Y}_{\tau'} = 0$ pro každé $\tau' > \tau$.
- V každém konkrétním bodě $\boldsymbol{X}$ tudíž platí, že predikce $\hat{Y}_\tau$ v závislosti na rostoucí hodnotě τ začíná na 1 pro $\tau = 0$ (kromě málo častého případu $\hat{p}(\boldsymbol{X}) = 0$) a poté v nějaké konkrétní hodnotě $\tau \in (0, 1)$ přeskočí do 0 a tam zůstane až do $\tau = 1$.

ROC křivka

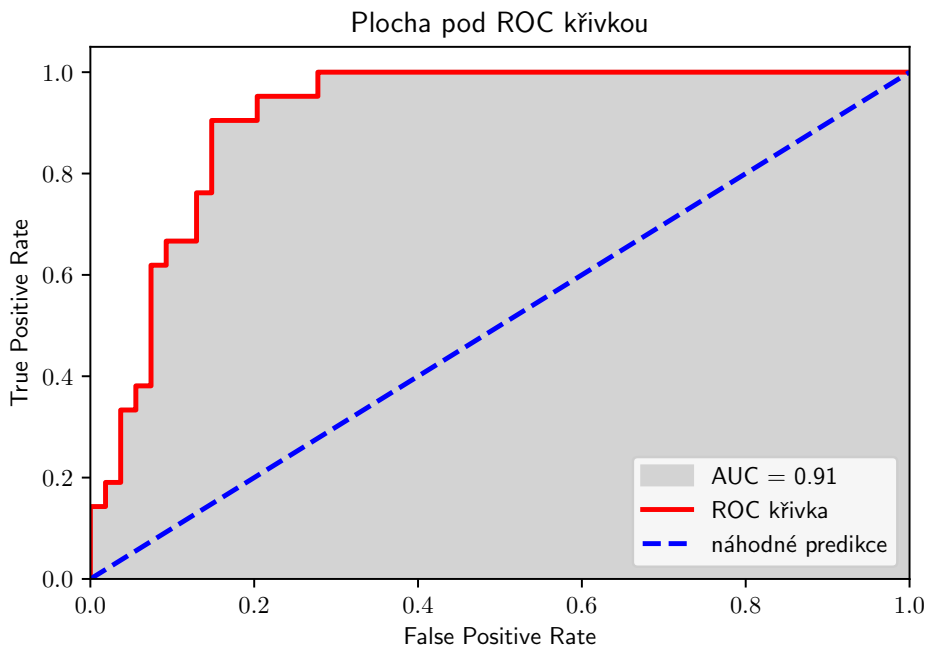
- Podívejme se nyní na chování true positive rate (TPR) a false positive rate (FPR) v závislosti na τ.
- Z předchozího slajdu plyne, že jsou obě neklesající funkce od τ.
- Graf TPR_τ versus FPR_τ jakožto implicitní funkce τ se nazývá **receiver operating characteristic** nebo také **ROC křivka**.



- Pro dobrý model, bude graf strmě stoupat k levému hornímu rohu a poté již jen velmi pomalu stoupat do pravého horního rohu, kde musí končit.
- Přímka na diagonále odpovídá náhodné predikci, kdy $\mathrm{TPR}\tau \doteq \mathrm{FPR}_\tau$ pro každé τ.

AUC - plocha pod ROC křivkou

- Kvalita modelu, pro který máme ROC křivku se obvykle vyhodnocuje jedním číslem, které znamená **plochu pod křivkou** (angl. **area under curve**) a značí se **AUC**.
- Model s náhodnými predikcemi bude mít plochu pod křivkou 0.5.
- AUC dokonalého modelu bude rovno 1.
- Obvykle má většina modelů AUC mezi 0.5 a 1.



BI-ML1.21 přednáška 5

Daniel Vašata

FIT ČVUT

23. 10. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 30. října 2024 15:48.

Co bude v dnešní přednášce

- Představení modelu lineární regrese
- Úvod do problematiky hledání extrémů funkce více proměnných
- Odhad parametrů lineárního modelu pomocí metody nejmenších čtverců
- Diskuse výsledků

Motivační příklad

- Chceme prodat nemovitost (řekněme v Praze) a netušíme, za kolik ji máme potenciálním zájemcům nabídnout.
- Zároveň si nechceme platit žádného „realitního“ odborníka, který by nám se stanovením ceny poradil.
- Zkusíme tedy udělat vlastní průzkum realitního trhu a vytvořit model, který nám pomůže cenu stanovit.
- Napíšeme skript, který stáhne data z realitních serverů a uloží je v nějaké strukturované podobě (ideálně tabulce či více tabulkách).
- Pro jednoduchost uvažujme, že budeme znát u každé nabídky toto:
  - Y – cenu, za kterou se prodává,
  - X_1 – užitnou plochu,
  - X_2 – počet místností,
  - X_3 – vzdálenost od nejbližší zastávky metra.
- Jako vysvětlovanou proměnnou Y jsme označili veličinu, kterou pro naši nemovitost neznáme a chceme ji predikovat.
- Příznaky $X_1, \ldots X_3$ označují veličiny, které pro naši nemovitost známe a o kterých věříme, že cenu Y ovlivňují.

Formalizace úlohy

Obecně tedy chceme na základě p příznaků $X_1, \ldots, X_p$ predikovat hodnotu vysvětlované proměnné Y.

V modelu lineární regrese předpokládáme lineární závislost vysvětlované proměnné na hodnotách příznaků.

Jelikož nedoufáme, že tato závislost je perfektní v tom smyslu, že pro stejné hodnoty $x_1, \ldots, x_p$ příznaků $X_1, \ldots, X_p$ dostaneme vždy stejnou hodnotu vysvětlované proměnné Y, modelujeme tuto závislost následovně:

$$Y = w_1 x_1 + \ldots + w_p x_p + \varepsilon,$$

kde $w_1, \ldots, w_p$ jsou nějaké neznámé koeficienty a ε je náhodná veličina.

Poznámky:

- Veličina ε odpovídá části Y, která je nevysvětlitelná pomocí hodnot příznaků a je tedy z našeho pohledu náhodná.
- Do náhodné veličiny ε se tak „schovají“ vlivy, které neznáme nebo cíleně nezahrnujeme do našeho modelu (např. stáří budovy, počet koupelen, počet oken) ale např. i chyby, nekonzistence dat a jiné podivnosti v měření příznaků.

Model lineární regrese

Obvykle ještě oddělujeme střední hodnotu náhodných vlivů a dostáváme tak:

Model lineární regrese

Hodnota vysvětlované proměnné Y v bodě $(x_1, \ldots, x_p)^T$ je

$$Y = w_0 + w_1 x_1 + \ldots + w_p x_p + \varepsilon,$$

kde $\mathrm{E}\,\varepsilon = 0$.

- Koeficient w_0 se nazývá **intercept** a odpovídá (očekávané) výchozí hodnotě Y při nulových příznacích.
- Zavedeme-li nový konstantní příznak $X_0 = x_0 = 1$ a vektorové značení

  $$\boldsymbol{x} = (x_0, x_1, \ldots, x_p)^T \quad \text{a} \quad \boldsymbol{w} = (w_0, w_1, \ldots, w_p)^T,$$

  můžeme zkráceně psát

  $$Y = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon.$$

- Vektor $\boldsymbol{w} = (w_0, w_1, \ldots, w_p)^T$ koeficientů také někdy nazýváme vektor vah.

Predikce v modelu lineární regrese

Předpokládejme nyní, že už máme odhad $\hat{\boldsymbol{w}}$ vektoru koeficientů $\boldsymbol{w}$.

Hodnotu Y v konkrétním bodě $\boldsymbol{x}$ predikujeme vztahem

$$\hat{Y} = \hat{\boldsymbol{w}}^T \boldsymbol{x} = \hat{w}_0 + \hat{w}_1 x_1 + \dots \hat{w}_p x_p.$$

Skutečná hodnota Y v bodě $\boldsymbol{x}$ je přitom určena vztahem

$$Y = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon$$

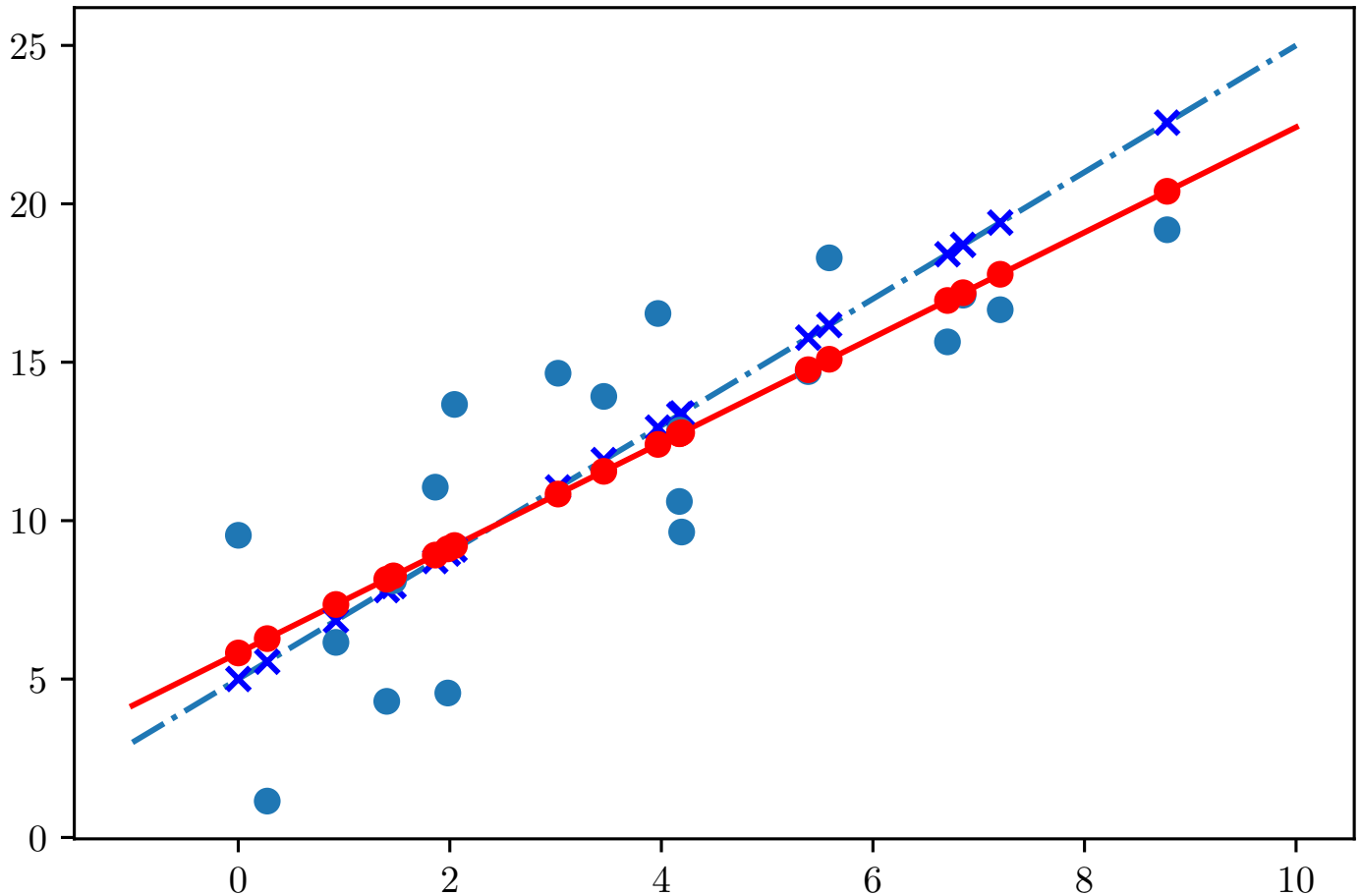
a je tedy náhodnou veličinou.

Z předpokladu $\mathrm{E}\,\varepsilon = 0$ plyne, že

$$\mathrm{E}\,Y = \boldsymbol{w}^T \boldsymbol{x}$$

a $\hat{Y}$ je tedy vlastně bodovým odhadem střední hodnoty $\mathrm{E}\,Y$ v bodě $\boldsymbol{x}$.

Vizualizace modelu lineární regrese



Modré body jsou body trénovací množiny. Červené body jsou predikce. Modré křížky odpovídají středním hodnotám bodů trénovací množiny, $(\boldsymbol{x}_i, \mathrm{E}\, Y_i)$. Modrá čerchovaná čára je skutečná regresní přímka daná rovnicí $y = \boldsymbol{w}^T \boldsymbol{x}$ a červená čára je přímka $\hat{y} = \hat{\boldsymbol{w}}^T \boldsymbol{x}$ určující naše predikce.

Měření chybovosti predikce pomocí ztrátové funkce

Zaměřme se nyní na problematiku odhadu vektoru parametrů modelu $\boldsymbol{w}$.

Následující úvahy jsou obecně platné pro supervizované učení nějakého modelu s parametry.

- Naším cílem je najít takovou hodnotu $\boldsymbol{w}$, aby chyba modelu byla co nejmenší.
- Tuto hodnotu pak použijeme jako odhad $\hat{\boldsymbol{w}}$.
- K tomu musíme specifikovat, co se myslí chybou modelu a v jakém smyslu má být nejmenší.
- Chybu modelu nejčastěji měříme pomocí nějaké nezáporné funkce $L : \mathbb{R}^2 \to \mathbb{R}$, nazývané **ztrátová funkce** (angl. **loss function**), kterou aplikujeme na skutečnou hodnotou proměnné Y a odpovídající predikci $\hat{Y}$.
- Obvyklou volbou v případě spojité vysvětlované veličiny bývá **kvadratická ztrátová funkce**,

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2.$$

Metoda nejmenších čtverců

- Velikost chyby modelu v bodě $\boldsymbol{x}$ je tedy $L(Y, \hat{Y})$, kde Y je skutečná hodnota vysvětlované Y v bodě $\boldsymbol{x}$ a $\hat{Y} = \boldsymbol{w}^T \boldsymbol{x}$ je predikce v bodě $\boldsymbol{x}$.
- Otázkou je, v jakém bodě $\boldsymbol{x}$ bychom měli hodnotu $L(Y, \hat{Y})$ vyhodnocovat a následně minimalizovat vzhledem k $\boldsymbol{w}$.
- Zřejmě to musí být bod $\boldsymbol{x}$ z trénovací množiny, protože jinak bychom neznali skutečnou hodnotu Y v tomto bodě.
- Abychom co nejvíc využili trénovací data, budeme minimalizovat součet chyb přes všechny body trénovací množiny, tj. přes všechny dvojice $(\boldsymbol{x}_i, Y_i)$ pro $i = 1, \ldots, N$.
- Součet chyb přes všechny tyto body pro kvadratickou ztrátovou funkci je

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} L(Y_i, \boldsymbol{w}^T \boldsymbol{x}_i) = \sum_{i=1}^{N} (Y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

  a nazýváme ho **reziduální součet čtverců** (angl. **residual sum of squares**).
- Minimalizací tohoto výrazu získáme odhad $\hat{\boldsymbol{w}}$. Tento postup se nazývá **metoda nejmenších čtverců** (angl. the method of **least squares**).

Parciální derivace

Protože $\boldsymbol{w}$ je vektor, minimalizace $\mathrm{RSS}(\boldsymbol{w})$ spadá do problematiky minimalizace funkce více proměnných.

Jak si nyní ukážeme, postupuje se analogicky jako v případě funkce jedné proměnné.

Definice

Buď $f : \mathbb{R}^d \to \mathbb{R}$ funkce d proměnných. **Parciální derivaci** funkce $f(x_1, \ldots, x_d)$ podle proměnné x_i v bodě $\boldsymbol{a} = (a_1, \ldots, a_d) \in \mathbb{R}^d$ definujeme jako derivaci funkce $g(x_i) = f(a_1, \ldots, a_{i-1}, x_i, a_{i+1} \ldots, a_d)$ v bodě a_i a značíme

$$\partial_{x_i} f(\boldsymbol{a}) \quad \text{nebo} \quad \frac{\partial f}{\partial x_i}(\boldsymbol{a}).$$

Označením $\partial_{x_i} f$ nebo $\frac{\partial f}{\partial x_i}$ pak myslíme funkci, která každému bodu, kde je konečná, přiřadí hodnotu parciální derivace podle x_i v tomto bodě.

Parciální derivace je stejná jako ta „obyčejná“, akorát ostatní proměnné bereme jako konstanty.

Platí např. $\partial_x (x + y) = 1$ nebo $\partial_y \sin(y + xy) = (1 + x)\cos(y + xy)$.

Gradient funkce

Definice

Buď $f : \mathbb{R}^d \to \mathbb{R}$ funkce d proměnných, která má v bodě $\boldsymbol{a} \in \mathbb{R}^d$ konečné všechny parciální derivace. **Gradient** funkce f v bodě $\boldsymbol{a}$ definujeme jako vektor

$$\nabla f(\boldsymbol{a}) = \left(\frac{\partial f}{\partial x_1}(\boldsymbol{a}), \ldots, \frac{\partial f}{\partial x_d}(\boldsymbol{a}) \right).$$

Označením ∇f pak myslíme gradient funkce jakožto zobrazení, které každému bodu, kde to lze, přiřadí gradient v tomto bodě.

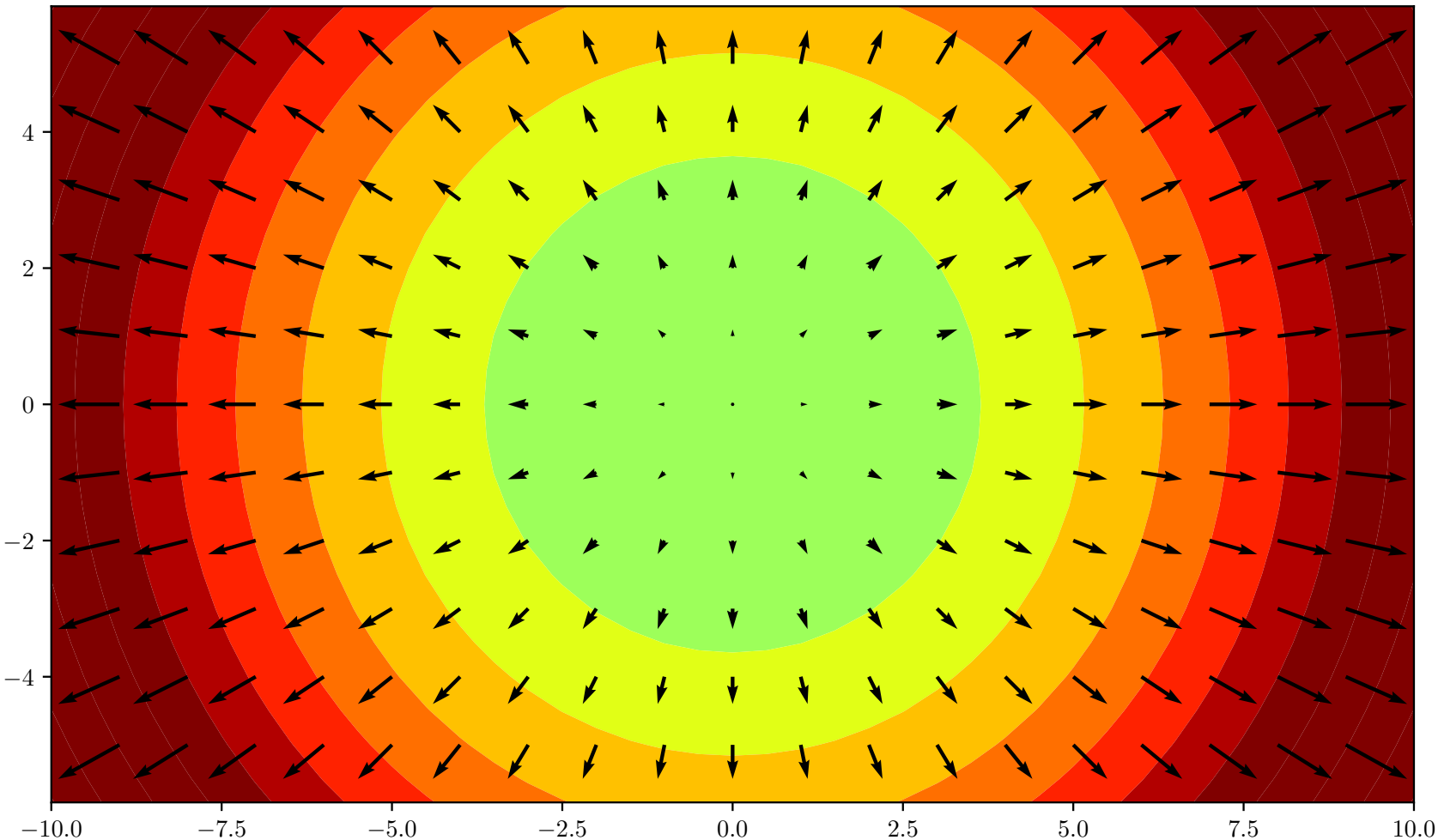
Gradient v bodě je tedy vektor z $\mathbb{R}^d$, jehož složky jsou jednotlivé parciální derivace.

Uvažujme funkci $f(x, y) = x^2 + y^2$, která odpovídá parabolické jámě s minimem v počátku $\mathbf{0}$. Platí $\nabla f = (2x, 2y)$.

Nejdůležitější vlastností gradientu[1] je, že ukazuje směr maximálního růstu funkce v daném bodě. Z toho mimo jiné plyne, že je gradient vždy kolmý na vrstevnici procházející daným bodem.

[1] Za dodatečného předpokladu spojitostí všech jeho složek na okolí daného bodu.

Vizualizace gradientu



Hessova matice

Analogicky k jednorozměrnému případu platí, že má-li funkce v nějakém bodě lokální extrém a gradient zde existuje, musí být nulový.

K získání postačující podmínky nám ještě zbývá sestrojit analog druhé derivace.

Definice

Buď $f : \mathbb{R}^d \to \mathbb{R}$ funkce d proměnných. **Hessovu matici** funkce f v bodě $\boldsymbol{a} \in \mathbb{R}^d$ definujeme jako

$$\mathbf{H}_f(\boldsymbol{a}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\boldsymbol{a}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(\boldsymbol{a}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1}(\boldsymbol{a}) & \cdots & \frac{\partial^2 f}{\partial x_d^2}(\boldsymbol{a}) \end{pmatrix},$$

kde $\frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{a}) = \left(\frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right)\right)(\boldsymbol{a})$ značí druhou parciální derivaci podle x_j a x_i.

Je to tedy matice druhých parciálních derivací podle všech proměnných.

Postačující podmínka pro existenci lokálního extrému

Věta

Buď $f : \mathbb{R}^d \to \mathbb{R}$ funkce d proměnných a bod $\boldsymbol{x}^* \in \mathbb{R}^d$ takový, že $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$ a f má spojité všechny druhé parciální derivace na nějakém okolí bodu $\boldsymbol{x}^*$.

- *Jestliže*
$$\boldsymbol{s}^T \mathbf{H}_f(\boldsymbol{x}^*)\boldsymbol{s} > 0, \quad \text{pro každé} \quad \boldsymbol{s} \in \mathbb{R}^d, \boldsymbol{s} \neq \mathbf{0},$$
nabývá funkce f v bodě $\boldsymbol{x}^*$ ostrého lokálního minima.
- *Jestliže pro každé $\boldsymbol{x}$ z nějakého okolí bodu $\boldsymbol{x}^*$*
$$\boldsymbol{s}^T \mathbf{H}_f(\boldsymbol{x})\boldsymbol{s} \geq 0, \quad \text{pro každé} \quad \boldsymbol{s} \in \mathbb{R}^d,$$
nabývá funkce f v bodě $\boldsymbol{x}^*$ neostrého lokálního minima.

Tyto vlastnosti se nazývají **pozitivní definitnost** resp. **pozitivní semi-definitnost** Hessovy matice $\mathbf{H}_f$ v bodě $\boldsymbol{x}^*$ resp. $\boldsymbol{x}$.

Pro funkci $f(x, y) = x^2 + y^2$ je řešením $\nabla f = (2x, 2y) = \mathbf{0}$ bod $\boldsymbol{x}^* = \mathbf{0}$ a Hessova matice je

$$\mathbf{H}_f(\boldsymbol{x}^*) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = 2\mathbf{I}.$$

Protože $\boldsymbol{s}^T 2\mathbf{I}\boldsymbol{s} = 2\boldsymbol{s}^T\boldsymbol{s} = 2\|\boldsymbol{s}\|^2 > 0$ pro každé $\boldsymbol{s} \neq \mathbf{0}$, nastává v bodě $\boldsymbol{x}^* = \mathbf{0}$ ostré lokální minimum. To pro náš paraboloid skutečně platí.

Přepis modelu trénovacích dat

Než aplikujeme předchozí metodu na minimalizaci RSS, přepišme si celkový model pro naše trénovací data do maticového tvaru.

Trénovací data považujeme za náhodný výběr z uvažovaného lineárního modelu provedený v různých bodech $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.

Máme tedy N párů $(Y_i, \boldsymbol{x}_i)$, kde $Y_i = \boldsymbol{w}^T \boldsymbol{x}_i + \varepsilon_i$.

Zaveďme náhodné vektory $\boldsymbol{Y} = (Y_1, \ldots, Y_N)^T$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_N)^T$ a body $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ zapišme v řádcích do matice $\mathbf{X} \in \mathbb{R}^{N,p+1}$,

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & x_{1;2} & \cdots & x_{1;p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & x_{N;2} & \cdots & x_{N;p} \end{pmatrix}.$$

Při tomto značení můžeme celkový model trénovací množiny zapsat jako

$$\boldsymbol{Y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{\varepsilon},$$

kde $\mathrm{E}\,\boldsymbol{\varepsilon} = (\mathrm{E}\,\varepsilon_1, \ldots, \mathrm{E}\,\varepsilon_N)^T = \mathbf{0}$.

Minimalizace RSS (1/3)

RSS můžeme vyjádřit jako

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} (Y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2.$$

Aplikujme předchozí teorii na minimalizaci $\mathrm{RSS}(\boldsymbol{w})$. Začneme parciálními derivacemi podle $w_0, \ldots, w_p$,

$$\frac{\partial \mathrm{RSS}}{\partial w_j} = \sum_{i=1}^{N} 2(Y_i - \boldsymbol{w}^T \boldsymbol{x}_i)(-x_{i;j}).$$

Pro gradient tedy dostáváme

$$\nabla \mathrm{RSS} = -\sum_{i=1}^{N} 2(Y_i - \boldsymbol{w}^T \boldsymbol{x}_i)\boldsymbol{x}_i = -2\mathbf{X}^T(\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}).$$

Položíme-li $\nabla \mathrm{RSS} = \mathbf{0}$, získáme tzv. **normální rovnici** (angl. **normal equation**),

$$\mathbf{X}^T\boldsymbol{Y} - \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{0}.$$

Minimalizace RSS (2/3)

Při výpočtu Hessovy matice použijeme

$$\frac{\partial^2 \mathrm{RSS}}{\partial w_k \partial w_j} = \sum_{i=1}^{N} 2(-x_{i;k})(-x_{i;j}).$$

Hessova matice je tedy

$$\mathbf{H}_{\mathrm{RSS}}(\boldsymbol{w}) = 2\mathbf{X}^T\mathbf{X}$$

bez ohledu na konkrétní hodnotu $\boldsymbol{w}$.

Dále pro každé $\boldsymbol{s} \in \mathbb{R}^{p+1}$ platí

$$\boldsymbol{s}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{s} = (\mathbf{X}\boldsymbol{s})^T(\mathbf{X}\boldsymbol{s}) = \|\mathbf{X}\boldsymbol{s}\|^2 \geq 0.$$

Hessova matice $\mathbf{H}_{\mathrm{RSS}}(\boldsymbol{w})$ je tedy vždy pozitivně semi-definitní.

Podle předchozí věty proto nastává neostré lokální minimum v jakémkoliv bodě $\boldsymbol{w}$, který řeší normální rovnici

$$\mathbf{X}^T\boldsymbol{Y} - \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{0}.$$

Minimalizace RSS (3/3)

Předpokládejme nyní, že $\mathbf{X}^T\mathbf{X}$ je **regulární matice**.

Normální rovnice

$$\mathbf{X}^T\boldsymbol{Y} - \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{X}^T\boldsymbol{Y}$$

má potom jednoznačné řešení

$$\hat{\boldsymbol{w}}_{\mathsf{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y},$$

kde značení vychází z anglického názvu **ordinary least squares** solution.

Pro matici $\mathbf{X}$ a libovolný vektor $\boldsymbol{s} \in \mathbb{R}^{p+1}$ snadno vidíme řetěz implikací

$$\mathbf{X}\boldsymbol{s} = \mathbf{0} \;\Rightarrow\; \mathbf{X}^T\mathbf{X}\boldsymbol{s} = \mathbf{0} \;\Rightarrow\; \boldsymbol{s}^T\mathbf{X}^T\mathbf{X}\boldsymbol{s} = 0 \;\Rightarrow\; \|\mathbf{X}\boldsymbol{s}\|^2 = 0 \;\Rightarrow\; \mathbf{X}\boldsymbol{s} = \mathbf{0}.$$

Z regularity $\mathbf{X}^T\mathbf{X}$ tedy plyne, že pro nenulové $\boldsymbol{s}$ nikdy nemůže platit $\boldsymbol{s}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{s} = 0$, a tedy Hessova matice je pozitivně definitní.

To znamená, že $\hat{\boldsymbol{w}}_{\mathsf{OLS}}$ je bodem ostrého lokálního minima.

Jak uvidíme později, v tomto bodě RSS nabývá dokonce globálního minima.

Shrnutí metody nejmenších čtverců

- Model pro vysvětlovanou proměnnou Y v bodě $\boldsymbol{x}$ je $Y = \boldsymbol{w}^T\boldsymbol{x} + \varepsilon$.
- Model pro trénovací množinu je $\boldsymbol{Y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$.
- Při trénování minimalizujeme residuální součet čtverců

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N}(Y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2.$$

- Za předpokladu, že je matice $\mathbf{X}^T\mathbf{X}$ regulární, existuje jediné řešení minimalizující $\mathrm{RSS}(\boldsymbol{w})$,

$$\hat{\boldsymbol{w}}_{\mathsf{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

- Predikce $\hat{Y}$ v bodě $\boldsymbol{x}$ je potom

$$\hat{Y} = \hat{\boldsymbol{w}}_{\mathsf{OLS}}^T\boldsymbol{x} = \boldsymbol{x}^T\hat{\boldsymbol{w}}_{\mathsf{OLS}} = \boldsymbol{x}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

Různé poznámky

- Lineární regrese je krásným příkladem diskriminativní metody, kdy přímo odhadujeme $\mathrm{P}(Y|\boldsymbol{X} = \boldsymbol{x})$. Resp. přímo $\mathrm{E}(Y|\boldsymbol{X} = \boldsymbol{x})$.
- Lineární regrese je ve vztahu ke problémům dimenzionality poměrně rezistentní.
- Důvodem je, že se jedná o parametrickou metodu. V ideálním případě nám tedy pro p příznaků $+1$ intercept může k určení přesného modelu stačit přesně $p + 1$ bodů trénovací množiny.
- Problémy nastávají, když jsou v důsledku malé trénovací množiny nebo špatných příznaků, které jsou např. silně korelované, sloupce matice $\mathbf{X}$ (skoro) lineárně závislé.
- Potom již nelze snadno provést inverzi matice $\mathbf{X}^T\mathbf{X}$ a případně je numericky nestabilní.
- V důsledku se tento problém typicky projeví přeučením modelu, které znamená, že se model příliš přizpůsobí trénovací množině a nebude schopen dobře predikovat nové body.

BI-ML1.21 přednáška 6

Daniel Vašata

FIT ČVUT

30. 10. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 30. října 2024 15:56.

Co bude v dnešní přednášce

- Připomenutí lineární regrese
- Geometrická interpretace metody nejmenších čtverců
- Problém lineárně závislých sloupců

Lineární model

- Model pro vysvětlovanou proměnnou Y v bodě $\boldsymbol{x}$ je

$$Y = \boldsymbol{w}^T\boldsymbol{x} + \varepsilon = \boldsymbol{x}^T\boldsymbol{w} + \varepsilon = w_0 + w_1x_1 + \dots w_px_p + \varepsilon.$$

- Model pro trénovací množinu tvořenou N páry $(Y_i, \boldsymbol{x}_i)$ je $Y_i = \boldsymbol{x}_i^T\boldsymbol{w} + \varepsilon_i$.
- Dohromady při značení $\boldsymbol{x}_i = (1, x_{i;1}, \dots, x_{i;p})^T$ tedy

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & x_{1;2} & \cdots & x_{1;p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & x_{N;2} & \cdots & x_{N;p} \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_p \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}.$$

- Toto zapisujeme maticově jako $\boldsymbol{Y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$, kde

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & \cdots & x_{1;p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & \cdots & x_{N;p} \end{pmatrix}, \quad \boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}.$$

- Naměřené hodnoty jednotlivých příznaků $X_1, \dots, X_p$ spolu s přidaným umělým příznakem $X_0 = 1$ jsou tedy uvedeny ve sloupcích matice $\mathbf{X}$.

Metoda nejmenších čtverců

- Při trénování minimalizujeme residuální součet čtverců

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} (Y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2.$$

- Minimum je určeno řešením **normální rovnice**

$$\mathbf{X}^T \boldsymbol{Y} - \mathbf{X}^T \mathbf{X} \boldsymbol{w} = \mathbf{0},$$

  která odpovídá podmínce $\nabla \mathrm{RSS}(\boldsymbol{w}) = \mathbf{0}$.

- Za předpokladu, že je matice $\mathbf{X}^T\mathbf{X}$ regulární, existuje jediné řešení minimalizující $\mathrm{RSS}(\boldsymbol{w})$,

$$\hat{\boldsymbol{w}}_{\mathrm{OLS}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{Y}.$$

- Predikce v bodě $\boldsymbol{x}$ je potom $\hat{Y} = \boldsymbol{x}^T \hat{\boldsymbol{w}}_{\mathrm{OLS}}$.

Geometrická interpretace metody nejmenších čtverců (1/3)

- Minimalizace $\mathrm{RSS}(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2$ je ekvivalentní minimalizaci $\|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|$.
- To znamená, že pro optimální $\boldsymbol{w}$ je Eukleidovská vzdálenost bodů $\boldsymbol{Y}$ a $\mathbf{X}\boldsymbol{w}$ v prostoru $\mathbb{R}^N$ nejmenší možná.
- Označíme-li i-tý sloupec matice $\mathbf{X}$ jako $\mathbf{X}_{\bullet i}$, můžeme si všimnout, že

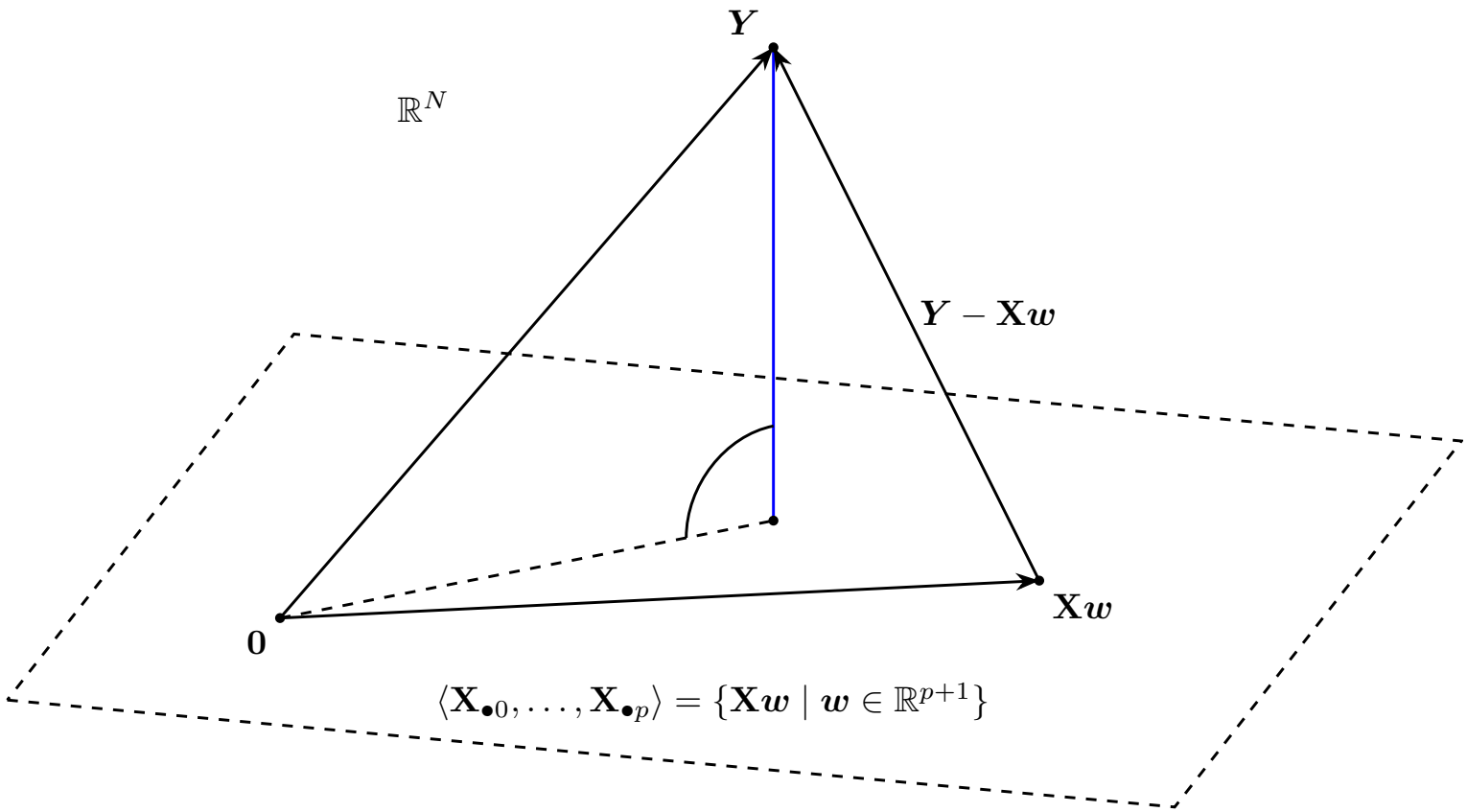
$$\mathbf{X}\boldsymbol{w} = w_0\mathbf{X}_{\bullet 0} + w_1\mathbf{X}_{\bullet 1} + \ldots + w_p\mathbf{X}_{\bullet p}.$$

- Vektor $\mathbf{X}\boldsymbol{w}$ je lineární kombinací sloupců matice $\mathbf{X}$ s koeficienty $w_0, \ldots, w_p$.
- Leží tedy v lineárním podprostoru prostoru $\mathbb{R}^N$, který je lineárním obalem $p+1$ sloupců $\mathbf{X}_{\bullet 0}, \ldots, \mathbf{X}_{\bullet p}$.
- Pro různé hodnoty $\boldsymbol{w}$ pak vektor $\mathbf{X}\boldsymbol{w}$ celý tento podprostor pokrývá, tj.

$$\langle \mathbf{X}_{\bullet 0}, \ldots, \mathbf{X}_{\bullet p} \rangle = \{\mathbf{X}\boldsymbol{w} \mid \boldsymbol{w} \in \mathbb{R}^{p+1}\}.$$

Geometrická interpretace metody nejmenších čtverců (2/3)

- Chceme-li minimalizovat vzdálenost $\boldsymbol{Y}$ a $\mathbf{X}\boldsymbol{w}$, hledáme bod $\mathbf{X}\boldsymbol{w}$ v podprostoru sloupců matice $\mathbf{X}$, který je k $\boldsymbol{Y}$ nejblíže.
- Bod $\mathbf{X}\boldsymbol{w}$ je k bodu $\boldsymbol{Y}$ nejblíže, jestliže je vektor $\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}$ na ten podprostor kolmý (modrý vektor na obrázku).



Geometrická interpretace metody nejmenších čtverců (3/3)

- Bod $\mathbf{X}\boldsymbol{w}$ je k bodu $\boldsymbol{Y}$ nejblíže, jestliže je vektor $\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}$ na ten podprostor kolmý.
- To znamená, že je kolmý na všechny vektory $\mathbf{X}_{\bullet 0}, \ldots, \mathbf{X}_{\bullet p}$, které ho generují:

$$(\mathbf{X}_{\bullet i})^T(\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}) = 0 \quad \text{pro všechny} \quad i = 0, \ldots, p.$$

- To lze maticově zapsat jako

$$\mathbf{X}^T(\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}) = \mathbf{0} \quad \text{a tedy} \quad \mathbf{X}^T\boldsymbol{Y} - \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{0}.$$

- Získali jsme tím starou známou normální rovnici a tedy stejné řešení.
- Z výše uvedených geometrických úvah navíc plyne, že pro jakékoliv řešení $\boldsymbol{w}$ normální rovnice je $\|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|$, a tedy i $\mathrm{RSS}(\boldsymbol{w})$, nejmenší možné.
- Jakékoliv řešení normální rovnice tedy dává globální minimum.

Regularita versus lineární nezávislost sloupců (1/3)

- Normální rovnice má jednoznačné řešení, pokud je $\mathbf{X}^T\mathbf{X}$ regulární.
- Pojďme si odvodit, jak to souvisí s lineární nezávislostí sloupců matice $\mathbf{X}$.
- Je-li $\mathbf{X}_{\bullet i}$ i-tý sloupec matice $\mathbf{X}$, platí, že vektor

$$\mathbf{X}\boldsymbol{s} = s_0\mathbf{X}_{\bullet 0} + s_1\mathbf{X}_{\bullet 1} + \ldots + s_p\mathbf{X}_{\bullet p}$$

  je lineární kombinací sloupců matice $\mathbf{X}$ s koeficienty danými složkami $\boldsymbol{s}$.

- Matice $\mathbf{X}$ má lineárně nezávislé sloupce, právě když je $\mathbf{X}\boldsymbol{s} = \mathbf{0}$ pouze pro $\boldsymbol{s} = \mathbf{0}$.

- Obecně pro matici $\mathbf{X}$ a libovolný vektor $\boldsymbol{s} \in \mathbb{R}^{p+1}$ platí

$$\mathbf{X}\boldsymbol{s} = \mathbf{0} \;\Rightarrow\; \mathbf{X}^T\mathbf{X}\boldsymbol{s} = \mathbf{0} \;\Rightarrow\; \boldsymbol{s}^T\mathbf{X}^T\mathbf{X}\boldsymbol{s} = 0 \;\Rightarrow\; \|\mathbf{X}\boldsymbol{s}\|^2 = 0 \;\Rightarrow\; \mathbf{X}\boldsymbol{s} = \mathbf{0}.$$

- Z toho plyne, že je $\mathbf{X}^T\mathbf{X}$ je regulární, právě když jsou sloupce matice $\mathbf{X}$ lineárně nezávislé.

Regularita versus lineární nezávislost sloupců (2/3)

- Problém zcela určitě nastává, pokud $N < p + 1$. Pak totiž v N rozměrném prostoru $\mathbb{R}^N$ nemůže existovat $p + 1$ lineárně nezávislých vektorů a tak ani sloupce $\mathbf{X}_{\bullet 0}, \ldots, \mathbf{X}_{\bullet p}$ matice $\mathbf{X}$ nemohou být lineárně nezávislé.

- I v situaci $N \geq p + 1$ se ale může stát, že sloupce $\mathbf{X}_{\bullet 0}, \ldots, \mathbf{X}_{\bullet p}$ nebudou lineárně nezávislé.

- Může to být například tím, že přímo jednotlivé příznaky jsou lineárně závislé a tedy jeden z nich je lineární kombinací ostatních.

- V takovém případě nepomůže ani libovolně vysoké N a sloupce matice $\mathbf{X}$ budou lineárně závislé vždy.

Regularita versus lineární nezávislost sloupců (3/3)

- Podívejme se, co to znamená pro řešení úlohy minimalizace $\mathrm{RSS}(\boldsymbol{w})$.
- Normální rovnice $\mathbf{X}^T\boldsymbol{Y} - \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{0}$ je z pohledu složek vektoru $\boldsymbol{w}$ soustava $p+1$ lineárních rovnic o $p+1$ neznámých.
- Tato soustava má vždy alespoň jedno řešení. Pokud jsou sloupce matice $\mathbf{X}$ lineárně nezávislé, je $\mathbf{X}^T\mathbf{X}$ regulární, řešení je právě jedno a to

$$\hat{\boldsymbol{w}}_{\mathsf{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

Regularita versus lineární nezávislost sloupců (4/4)

- V opačném případě existuje nekonečně mnoho řešení tak, že pro každé dvě řešení $\boldsymbol{w}$ a $\boldsymbol{w}'$ platí $\mathbf{X}^T\mathbf{X}(\boldsymbol{w} - \boldsymbol{w}') = \mathbf{0}$.

- To, jak víme z předchozích úvah, implikuje $\mathbf{X}(\boldsymbol{w} - \boldsymbol{w}') = \mathbf{0}$.

- Pro každou dvojici řešení tedy platí

$$\begin{aligned}\mathrm{RSS}(\boldsymbol{w}) &= \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w} + \mathbf{X}\boldsymbol{w}' - \mathbf{X}\boldsymbol{w}'\|^2 \\ &= \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}' - \mathbf{X}(\boldsymbol{w} - \boldsymbol{w}')\|^2 = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}'\|^2 = \mathrm{RSS}(\boldsymbol{w}').\end{aligned}$$

- Všechny řešení tudíž odpovídají stejné hodnotě RSS, která je, jak jsme již zmínili, globálním minimem, které je v tomto případě neostré.

Řešení při lineární závislých sloupcích

- Otázkou je, jak nějaké řešení získat, když nemůžeme invertovat matici $\mathbf{X}^T\mathbf{X}$ a následně použít vzoreček $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}$.
- Funkce `LinearRegression` z balíčku `scikit-learn` si s takovými případy (většinou) poradí a řešení vrátí.[1]
- Pokud $\mathbf{X}^T\mathbf{X}$ není regulární, vrátí (v rámci numerických možností) takový vektor $\hat{\boldsymbol{w}}$, který řeší normální rovnici a zároveň má mezi všemi řešeními nejmenší normu $\|\hat{\boldsymbol{w}}\|$.
- Jen pro zajímavost uveďme, že toto řešení lze zapsat ve tvaru

$$\hat{\boldsymbol{w}} = (\mathbf{X}^T\mathbf{X})^{+}\mathbf{X}^T\boldsymbol{Y},$$

kde $(\mathbf{X}^T\mathbf{X})^{+}$ je takzvaná Moorova-Penroseova pseudoinverzní matice k matici $\mathbf{X}^T\mathbf{X}$.

[1] Jen pro zajímavost uveďme, že uvnitř se využívá funkce `dgelsd` z knihovny LAPACK.

Problém kolinearity

- Problémem nejsou pouze případy, kdy jsou sloupce matice $\boldsymbol{X}$ lineárně závislé, ale úplně stačí, když jsou „skoro“ lineárně závislé.
- V obou těchto případech mluvíme o problému **kolinearity** (angl. **collinearity**).
- Myslíme tím tedy, že existují lineární kombinace sloupců, které dávají téměř nulové vektory, zatímco jiné lineární kombinace vrací mnohem větší vektory, tj.

$$\|\mathbf{X}\boldsymbol{u}\| \gg \|\mathbf{X}\boldsymbol{v}\| \doteq 0 \quad \text{pro nějaké} \quad \|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1.$$

- V takovém případě sice inverze $\mathbf{X}^T\mathbf{X}$ teoreticky existuje, ale prakticky je její výpočet numericky problematický.
- Především - a to je to hlavní jádro problému - je získaný odhad $\hat{\boldsymbol{w}}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}$ velmi citlivý na malé nevhodné změny $\boldsymbol{Y}$.

Důsledky kolinearity

- Především - a to je to hlavní jádro problému - je získaný odhad $\hat{\boldsymbol{w}}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}$ velmi citlivý na malé nevhodné změny $\boldsymbol{Y}$ nebo $\mathbf{X}$.
- To znamená, že kdybychom náhodný výběr trénovací množiny zopakovali, může se hodnota odhadu $\hat{\boldsymbol{w}}_{\text{OLS}}$ radikálně změnit.
- Z pravděpodobnostního pohledu lze ukázat, že $\hat{\boldsymbol{w}}_{\text{OLS}}$ má potom v jistých směrech velký rozptyl.
- Toto se pochopitelně přenáší i na predikce $\hat{Y}$, které pak mají v některých bodech velký rozptyl, což znamená, že jim nemůžeme příliš důvěřovat.

Řešení problému kolinearity

Pokud narazíme na problém kolinearity, máme v zásadě tři možnosti:

- Přigenerovat další data nebo odebrat existující a doufat, že se problém vyřeší.

  To se ale nestane, pokud jsou samotné příznaky (skoro) lineárně závislé.

- Snížit počet příznaků. To znamená vyhození některých příznaků, případně nahrazení příznaků menším počtem nových, které již nebudou lineárně závislé.

  Jednou z metod redukce počtu příznaků, kdy ty existující nahrazujeme menším počtem jejich lineárních kombinací, se budeme zabývat v příštím semestru.

- Změnit funkci, kterou minimalizujeme, abychom měli jednoznačné a stabilní řešení.

  Typicky provedeme tak zvanou regularizaci, kdy k RSS přidáme **regularizační člen**, který problémy kolinearity odstraní nebo alespoň dostatečně zmírní.

BI-ML1.21 přednáška 7

Daniel Vašata

FIT ČVUT

6. 11. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 6. listopadu 2024 14:28.

Co bude v dnešní přednášce

- Hřebenová regrese
- Vztah vychýlení a rozptylu
- Modely bázových funkcí

Hřebenová regrese - úvod

Hřebenová regrese (angl. **ridge regression**) [Hoerl, Kennard (1970)] nebo taky L_2 regularizace se k problému kolinearity staví zavedením penalizačního členu úměrného kvadrátu normy vektoru koeficientů $\boldsymbol{w}$ s vynecháním interceptu.

Minimalizujeme tedy **regularizovaný reziduální součet čtverců**

$$\mathrm{RSS}_\lambda(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \sum_{i=1}^{p} w_i^2,$$

který závisí na parametru $\lambda \geq 0$.

- Pro $\lambda = 0$ dostáváme $\mathrm{RSS}_0(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w})$ a máme tedy obyčejnou metodu nejmenších čtverců.
- Pro $\lambda > 0$ je vidět, že v minimu se bude cílit na takové vektory $\boldsymbol{w}$, které mají co nejmenší složky.
- Hodnotu w_0 interceptu nijak nepenalizujeme. Jedná se pouze o vertikální posun, který zajišťuje předpoklad $\mathrm{E}\,\varepsilon = 0$ modelu a je tedy vhodné ho neomezovat.
- Stále platí, že model pro Y v bodě $\boldsymbol{x}$ je $Y = w_0 + w_1 x_1 + \ldots w_p x_p + \varepsilon$.

Hřebenová regrese - normální rovnice

Zavedeme-li matici

$$\mathbf{I}' = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{p+1,p+1},$$

můžeme psát

$$\mathrm{RSS}_\lambda(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \sum_{i=1}^{p} w_i^2 = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \boldsymbol{w}^T \mathbf{I}' \boldsymbol{w}.$$

Gradient je

$$\nabla \mathrm{RSS}_\lambda(\boldsymbol{w}) = -2\mathbf{X}^T(\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}) + 2\lambda \mathbf{I}' \boldsymbol{w}.$$

Ekvivalent **normální rovnice** je tedy

$$\mathbf{X}^T \boldsymbol{Y} - \mathbf{X}^T \mathbf{X} \boldsymbol{w} - \lambda \mathbf{I}' \boldsymbol{w} = \mathbf{0}.$$

Hřebenová regrese - odhad parametrů

Hessova matice je dále

$$\mathbf{H}_{\mathrm{RSS}_\lambda}(\boldsymbol{w}) = 2\mathbf{X}^T\mathbf{X} + 2\lambda\mathbf{I}' = 2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}').$$

Pro každé $\boldsymbol{s} \in \mathbb{R}^{p+1}, \boldsymbol{s} \neq \mathbf{0}$ a $\lambda > 0$ platí

$$\boldsymbol{s}^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}')\boldsymbol{s} = (\mathbf{X}\boldsymbol{s})^T(\mathbf{X}\boldsymbol{s}) + \lambda\boldsymbol{s}^T\mathbf{I}'\boldsymbol{s} = \|\mathbf{X}\boldsymbol{s}\|^2 + \lambda\sum_{i=1}^{p} s_i^2 > 0,$$

protože pro $\boldsymbol{s} = (s_0, 0, \ldots, 0)^T \neq \mathbf{0}$ máme $\mathbf{X}\boldsymbol{s} = (s_0, \ldots, s_0)^T \neq \mathbf{0}$.

Hessova matice je tedy pozitivně definitní a matice $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}'$ je regulární.

Pro $\lambda > 0$ tak vždy existuje jednoznačné řešení normální rovnice

$$\hat{\boldsymbol{w}}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}')^{-1}\mathbf{X}^T\boldsymbol{Y}$$

a odpovídá globálnímu minimu RSS_λ.

Predikce v bodě $\boldsymbol{x}$ je potom opět $\hat{Y} = \boldsymbol{x}^T\hat{\boldsymbol{w}}_\lambda$.

Očekávaná chyba modelu

V modelu pro trénovací množinu, $\boldsymbol{Y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$, je $\boldsymbol{\varepsilon}$ náhodný vektor, pro který předpokládáme $\mathrm{E}\,\boldsymbol{\varepsilon} = \mathbf{0}$.

Z toho plyne, že i $\boldsymbol{Y}$ je náhodný vektor a tedy i odhad vektoru parametrů $\hat{\boldsymbol{w}}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}')^{-1}\mathbf{X}^T\boldsymbol{Y}$ je náhodný vektor.

Uvažujme nyní nějaký pevný bod $\boldsymbol{x} = (1, x_1, \ldots, x_p) \in \mathbb{R}^{p+1}$ a zkoumejme očekávanou chybu měřenou pomocí kvadratické ztrátové funkce při predikci $Y = \boldsymbol{x}^T\boldsymbol{w} + \varepsilon$ pomocí $\hat{Y} = \boldsymbol{x}^T\hat{\boldsymbol{w}}_\lambda$.

Budeme předpokládat nezávislost trénovacích a testovacích dat, tj. nezávislost $\boldsymbol{Y}$ a Y, a v důsledku tedy nezávislost $\hat{Y}$ a Y.

Z toho plyne

$$\begin{aligned}\mathrm{E}\left((Y - \mathrm{E}\,Y)(\mathrm{E}\,Y - \hat{Y})\right) &= \mathrm{E}\left(Y(\mathrm{E}\,Y) - (Y\hat{Y}) - (\mathrm{E}\,Y)^2 + (\mathrm{E}\,Y)\hat{Y}\right)\\ &= (\mathrm{E}\,Y)^2 - \mathrm{E}(Y\hat{Y}) - (\mathrm{E}\,Y)^2 + \mathrm{E}\,Y\,\mathrm{E}\,\hat{Y}\\ &= -\mathrm{E}(Y\hat{Y}) + \mathrm{E}\,Y\,\mathrm{E}\,\hat{Y} = 0.\end{aligned}$$

Rozklad očekávané chyby modelu

Pro očekávanou chybu tedy platí

$$\begin{aligned}\mathrm{E}\, L(Y,\hat{Y}) &= \mathrm{E}(Y-\hat{Y})^2 = \mathrm{E}(Y-\mathrm{E}\,Y+\mathrm{E}\,Y-\hat{Y})^2 \\ &= \mathrm{E}(Y-\mathrm{E}\,Y)^2 + 2\,\mathrm{E}\left((Y-\mathrm{E}\,Y)(\mathrm{E}\,Y-\hat{Y})\right) + \mathrm{E}(\hat{Y}-\mathrm{E}\,Y)^2 \\ &= \mathrm{E}(Y-\mathrm{E}\,Y)^2 + \mathrm{E}(\hat{Y}-\mathrm{E}\,Y)^2.\end{aligned}$$

Označíme-li $\mathrm{var}\, Y = \mathrm{var}\,\varepsilon = \sigma^2$ dostáváme

$$\mathrm{E}\, L(Y,\hat{Y}) = \sigma^2 + \mathrm{E}(\hat{Y}-\mathrm{E}\,Y)^2.$$

První člen odpovídá neodstranitelné chybě, která je dána náhodností v modelu. Tato chyba se nazývá Bayesovská (**Bayes error**).

Druhý člen se značí $\mathrm{MSE}(\hat{Y})$ a nazývá střední kvadratická chyba odhadu $\hat{Y}$ parametru $\mathrm{E}\,Y$ (angl. **mean squared error**).

Rozklad očekávané chyby modelu

Pro MSE($\hat{Y}$) dále platí:

$$\begin{aligned}
\mathrm{MSE}(\hat{Y}) &= \mathrm{E}(\hat{Y} - \mathrm{E}\,Y)^2 = \mathrm{E}(\mathrm{E}\,\hat{Y} - \mathrm{E}\,Y + \hat{Y} - \mathrm{E}\,\hat{Y})^2 \\
&= \mathrm{E}(\mathrm{E}\,\hat{Y} - \mathrm{E}\,Y)^2 + \mathrm{E}(\hat{Y} - \mathrm{E}\,\hat{Y})^2 + 2\,\mathrm{E}(\hat{Y} - \mathrm{E}\,\hat{Y})(\mathrm{E}\,\hat{Y} - \mathrm{E}\,Y) \\
&= (\mathrm{E}\,\hat{Y} - \mathrm{E}\,Y)^2 + \mathrm{E}(\hat{Y} - \mathrm{E}\,\hat{Y})^2 + 2 \cdot 0 \cdot (\mathrm{E}\,\hat{Y} - \mathrm{E}\,Y) \\
&= (\mathrm{E}\,\hat{Y} - \mathrm{E}\,Y)^2 + \mathrm{var}\,\hat{Y} = (\mathrm{bias}\,\hat{Y})^2 + \mathrm{var}\,\hat{Y},
\end{aligned}$$

kde $\mathrm{bias}\,\hat{Y} = \mathrm{E}\,\hat{Y} - \mathrm{E}\,Y$ značí vychýlení odhadu (angl. bias).

Dohromady tedy máme finální dekompozici očekávané chyby jako

$$\mathrm{E}\,L(Y, \hat{Y}) = \sigma^2 + (\mathrm{bias}\,\hat{Y})^2 + \mathrm{var}\,\hat{Y}.$$

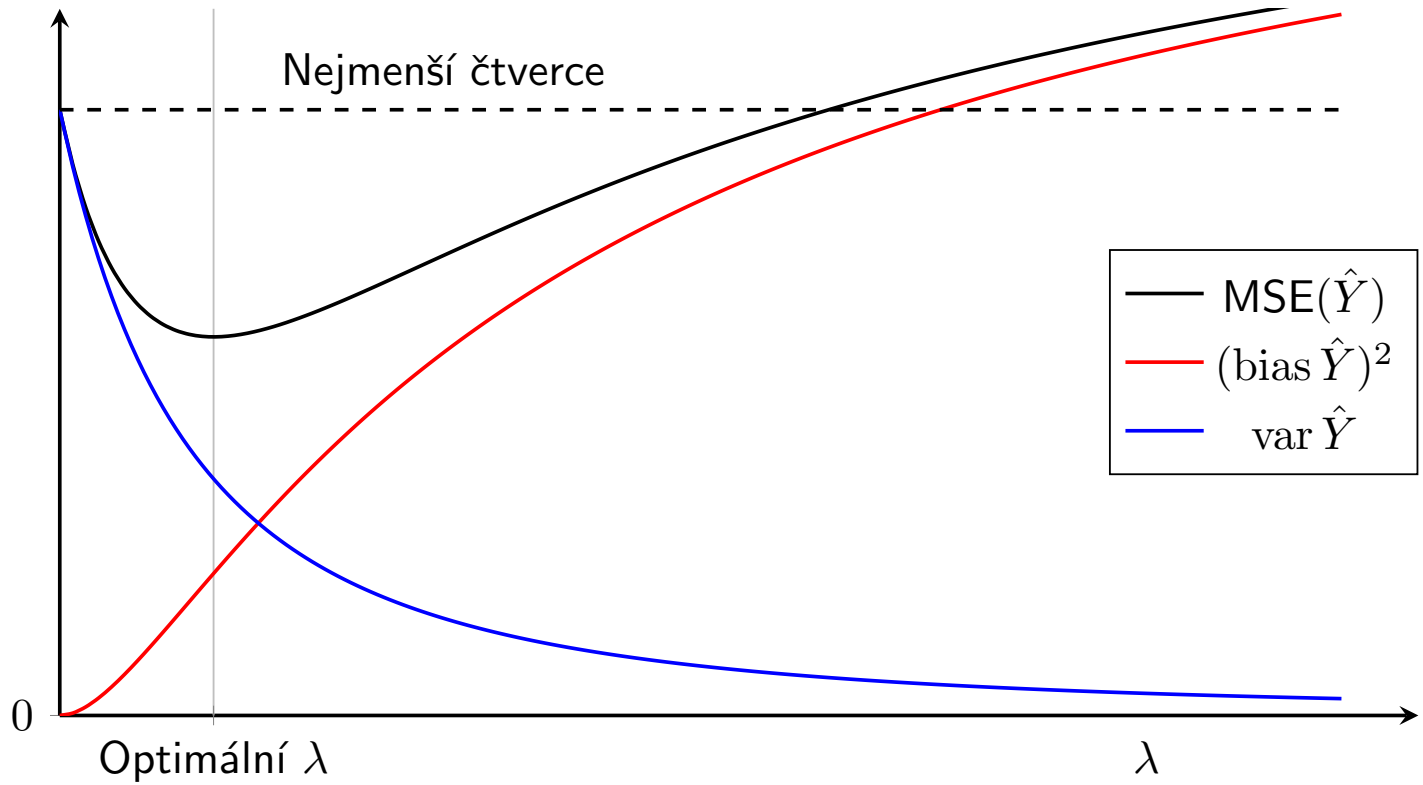
Očekávaná chyba modelu je tedy součtem **neodstranitelné chyby**, kvadrátu **vychýlení** odhadu a **rozptylu** odhadu.

Vztah vychýlení a rozptylu

U hřebenové regrese lze ukázat, že (hodně zjednodušeně) platí

$$(\operatorname{bias} \hat{Y})^2 \sim \left(1 - \frac{1}{1+\lambda}\right)^2 \quad \text{a} \quad \operatorname{var} \hat{Y} \sim \left(\frac{1}{1+\lambda}\right)^2.$$

To znamená, že s rostoucím λ vychýlení roste a rozptyl klesá. Takovéto chování v závislosti na hyperparametrech modelu je typické a nazývá se **bias-variance tradeoff**.



Různé poznámky

- Hledáme tedy optimální hodnotu parametru λ, pro kterou je chyba modelu nejmenší.
- Obvykle se snažíme minimalizovat odhad MSE validační množiny dat případně odhad MSE pomocí cross-validace. Odhad MSE se pro validační množinu $(Y_i', \boldsymbol{x}_i')$ velikosti n počítá jako

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i' - \boldsymbol{x}_i'^T\hat{\boldsymbol{w}}_\lambda\right)^2.$$

- Při použití hřebenové regrese bývá obvyklé nejprve jednotlivé příznaky standardizovat, aby se staly rozsahově porovnatelné a tedy, aby byly penalizovány všechny stejně. Tj. místo příznaku X_i použijeme příznak

$$X_i' = \frac{X_i - \bar{X}_i}{\sqrt{s_{X_i}^2}}, \quad \text{kde} \quad \bar{X}_i = \frac{1}{N}\sum_{j=1}^{N} x_{j;i} \quad \text{a} \quad s_{X_i}^2 = \frac{1}{N-1}\sum_{j=1}^{N}(x_{j;i} - \bar{X}_i)^2$$

- Existují i další možnosti regularizace jako např. $\lambda\sum_{i=1}^{p}|w_i|$ (Lasso).

Modely bázových funkcí (1/2)

- Doposud jsme uvažovali pouze jednoduchý lineární model ve tvaru

$$Y = \boldsymbol{x}^T \boldsymbol{w} + \varepsilon.$$

- Principiálně jsme tak schopni modelovat pouze lineární funkci ve vstupních proměnných. Ukažme si, jak rozšířit naše možnosti za obzor linearity.
- Základní rozšíření spočívá v nahrazení původních příznaků jejich transformovanými variantami.
- Pro $M \in \mathbb{N}$ vezměme M funkcí $\varphi_1, \ldots, \varphi_M$ z $\mathbb{R}^p$ do $\mathbb{R}$ reprezentujících transformace X a nazvěme je **bázové funkce** (angl. **basis functions**).
- K těmto funkcím přidáme $\phi_0(\boldsymbol{x}) = 1$ a poskládáme je do vektorové funkce $\boldsymbol{\varphi} : \mathbb{R}^p \to \mathbb{R}^{M+1}$ vztahem $\boldsymbol{\varphi}(\boldsymbol{x}) = (1, \varphi_1(\boldsymbol{x}), \ldots, \varphi_M(\boldsymbol{x}))^T$.
- Jako model vztahu Y a $\boldsymbol{x}$ budeme uvažovat lineární model

$$Y = \sum_{j=0}^{M} w_j \varphi_j(\boldsymbol{x}) + \varepsilon = \boldsymbol{\varphi}(\boldsymbol{x})^T \boldsymbol{w} + \varepsilon,$$

- Další postup je nyní zcela analogický jako před tím.

Modely bázových funkcí (2/2)

- Mějme tedy trénovací množinu jako náhodný výběr z výše uvedeného modelu určený N páry typu $(Y_i, \boldsymbol{x}_i)$.
- Maticově můžeme zapsat model pro trénovací data jako $\boldsymbol{Y} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\varepsilon}$, kde

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\varphi}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{\varphi}(\boldsymbol{x}_N)^T \end{pmatrix} = \begin{pmatrix} 1 & \phi_1(\boldsymbol{x}_1) & \cdots & \phi_M(\boldsymbol{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\boldsymbol{x}_N) & \cdots & \phi_M(\boldsymbol{x}_N) \end{pmatrix}, \boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix}, \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}.$$

- Obecně budeme minimalizovat (pro $\lambda = 0$ máme metodu nejmenších čtverců)

$$\mathrm{RSS}_\lambda(\boldsymbol{w}) = \|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 + \lambda \boldsymbol{w}^T \mathbf{I}' \boldsymbol{w}.$$

- Řešením je (pro $\lambda = 0$ značíme také $\hat{\boldsymbol{w}}_{\mathrm{OLS}}$)

$$\hat{\boldsymbol{w}}_\lambda = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \mathbf{I}'\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{Y}.$$

- Predikce hodnoty Y v bodě $\boldsymbol{x}$ je potom určena vztahem

$$\hat{Y} = \boldsymbol{\varphi}(\boldsymbol{x})^T \hat{\boldsymbol{w}}_\lambda.$$

Bázové funkce

Mezi obvyklé volby bázových funkcí patří:

- $\varphi(\boldsymbol{x}) = x_i$ – přímo jednotlivé příznaky.
- $\varphi(\boldsymbol{x}) = x_i^2$, $\varphi(\boldsymbol{x}) = x_k x_\ell$ – mocniny příznaků a jejich různé součiny, odpovídá polynomiální regresi.
- $\varphi(\boldsymbol{x}) = \log(x_i), \sqrt{x_i}, \sin(x_i)$ atd. – nelineární transformace jednotlivých příznaků.
- $\varphi(\boldsymbol{x}) = \mathbb{1}_{(a,b)}(x_i)$, kde $\mathbb{1}_A(x) = 1$ pokud $x \in A$ a $\mathbb{1}_A(x) = 0$ pokud $x \notin A$ – indikátory množin. Umožňují rozdělení prostoru příznaků na kousky a následné modelování v každém kousku zvlášť.
- $\varphi(\boldsymbol{x}) = h(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$, kde $\boldsymbol{x}_i$ je i-tý trénovací bod a h je nějaká funkce – tzv. **radiální bázové funkce** centrované v bodech trénovací množiny.

Pokud nemáme žádné speciální znalosti o systému, který modelujeme, typicky na počátku volíme velké množství bázových funkcí a používáme hřebenovou regresi, případně jinou formu regularizace.

Nestrannost metody nejmenších čtverců

Jak jsme již zmínili, odhad vektoru parametrů $\hat{\boldsymbol{w}}_\lambda = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}$ je náhodný vektor.

Je to tedy **bodový odhad** vektoru parametrů $\boldsymbol{w}$ a je příkladem tzv. statistiky[1].

Věta

Odhad $\hat{\boldsymbol{w}}_{OLS}$ získaný metodou nejmenších čtverců je za předpokladu $\mathrm{E}\,\boldsymbol{\varepsilon} = \mathbf{0}$ ***nestranný****, tj. $\mathrm{E}\,\hat{\boldsymbol{w}}_{OLS} = \boldsymbol{w}$.*

Důkaz.

Z linearity střední hodnoty plyne:

$$\mathrm{E}\,\boldsymbol{Y} = \mathrm{E}(\mathbf{X}\boldsymbol{w} + \boldsymbol{\varepsilon}) = \mathbf{X}\boldsymbol{w} + \mathrm{E}\,\boldsymbol{\varepsilon} = \mathbf{X}\boldsymbol{w}.$$

Dále platí:

$$\begin{aligned}\mathrm{E}\,\hat{\boldsymbol{w}}_{\mathrm{OLS}} &= \mathrm{E}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{Y} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\,\mathrm{E}\,\boldsymbol{Y} \\ &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{w} = \boldsymbol{w}.\end{aligned}$$

□

[1] Funkce od náhodného výběru (v našem případě trénovací množiny), viz BI-PST.

Nestrannost metody nejmenších čtverců

Z nestrannosti odhadu vektoru parametrů $\hat{\boldsymbol{w}}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}$ dále plyne nestrannost odhadu $\hat{Y}$ v bodě $\boldsymbol{x}$.

K tomu je třeba si uvědomit, že pomocí $\hat{Y} = \boldsymbol{x}^T\hat{\boldsymbol{w}}_{\text{OLS}}$ se sice snažíme predikovat skutečnou hodnotu Y, ale ze statistického pohledu se jedná o bodový odhad střední hodnoty $\mathrm{E}\,Y = \boldsymbol{x}^T\boldsymbol{w} + \mathrm{E}\,\varepsilon = \boldsymbol{x}^T\boldsymbol{w}$.

Z předchozí věty tak plyne

$$\mathrm{E}\,\hat{Y} = \mathrm{E}\,\boldsymbol{x}^T\hat{\boldsymbol{w}}_{\text{OLS}} = \boldsymbol{x}^T\,\mathrm{E}\,\hat{\boldsymbol{w}}_{\text{OLS}} = \boldsymbol{x}^T\boldsymbol{w} = \mathrm{E}\,Y$$

a $\hat{Y}$ je tedy **nestranným odhadem** $\mathrm{E}\,Y$.

To samozřejmě znamená, že

$$\mathrm{bias}\,\hat{Y} = \mathrm{E}\,\hat{Y} - \mathrm{E}\,Y = 0,$$

t.j. vychýlení je 0.

BI-ML1.21 přednáška 8

Daniel Vašata

FIT ČVUT

13. 11. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 13. listopadu 2024 16:01.

Co bude v dnešní přednášce

- Připomenutí problému klasifikace
- Základní myšlenka logistické regrese
- Maximálně věrohodný odhad: myšlenka
- Výpočet odhadu

Úvodní poznámky

- Přestože se metoda **Logistická regrese** jmenuje tak, jak se jmenuje, je to metoda určená pro *klasifikaci*.
- Pro připomenutí: V případě klasifikace vysvětlovaná proměnná Y může nabývat jen několika málo hodnot.
- **My se omezíme na binární klasifikaci**, kdy má Y hodnotu buď 0 nebo 1.
- Rozdíl mezi regresí (spojitá vysvětlovaná proměnná) a klasifikací se projevuje zejména v tom, jaký je tvar hledané závislosti mezi příznaky $X_1, X_2, \ldots, X_p$ a vysvětlovanou proměnnou.
- Zatímco pro regresi může mít (ale často nemá) tato závislost vztah připomínající klasické funkce známé z analýzy, u funkcí, jejichž obor hodnot je dvouprvková množina, se musí použít nějaký „trik“.

Připomenutí známých metod

Připomeňme si metody, které už známe:

- Rozhodovací stromy (zvládají klasifikaci i regresi): Funkční závislost je komplikovaná a daná průchodem daným stromem. V případě regrese má výsledná funkce tvar po (velmi malých) částech konstantní funkce.

- KNN (zvládá klasifikaci i regresi): Funkční závislost je opět velice komplikovaná a nemá žádný explicitní tvar. Rozhodnutí o hodnotě Y je velice „lokální“.

- Lineární regrese (jen pro regresi): Jedná se o parametrickou metodu, kde se hledá závislost v zadaném tvaru „lineární kombinace příznaků“

$$Y \approx w_0 + w_1 x_1 + \cdots + w_p x_p,$$

kde x_i jsou nějaké konkrétní hodnoty příznaků X_i a w_i jsou neznámé koeficienty.

Příklad (rýmička)

- Uvažujme trochu modifikovaný příklad „rýmičkového“ příkladu z druhé přednášky o stromech:
  - Vysvětlovaná proměnná Y má dvě hodnoty: $Y = 1$ značí, že má daná osoba rýmičku, $Y = 0$ značí, že jí nemá.
  - X_1 je binární příznak pohlaví (1 = žena, 0 = muž).
  - X_2 je numerický příznak věku (v celých letech).
  - X_3 je teplota ve stupních Celsia.

- Souvislost logistické regrese a lineární regrese spočívá v tom, že i u logistické regrese se rozhodnutí konstruuje pomocí lineární kombinace příznaků, tedy v našem případě výrazu

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3.$$

- Jak ale donutit tento výraz, aby z něho padalo rozhodnutí o tom, jestli je $Y = 1$ nebo $Y = 0$?

Myšlenka logistické regrese

- Logistická regrese stojí na triku, který z diskrétního problému dělá problém spojitý: **Namísto hodnoty vysvětlované proměnné $Y \in \{0, 1\}$ se snažíme predikovat pravděpodobnost, že Y má hodnotu 1, tj. číslo $\mathrm{P}(Y = 1)$ z intervalu $[0, 1]$.**
- Přesněji řečeno: hledáme *funkční předpis*, který pro dané hodnoty x_i příznaků X_i a dané hodnoty koeficientů w_i, vrátí číslo z intervalu $[0, 1]$, které bude odhadem pravděpodobnosti, že daná osoba má rýmičku $(Y = 1)$.
- Tuto získanou pravděpodobnost budeme značit následovně:

$$\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}),$$

  čímž je vyjádřeno to, že je závislá na hodnotách příznaků $\boldsymbol{x} = (1, x_1, x_2, x_3)$ a koeficientů $\boldsymbol{w} = (w_0, w_1, w_2, w_3)$.
- Jelikož součet pravděpodobností, že někdo má a nemá rýmičku musí být jedna, platí

$$\mathrm{P}(Y = 0 \mid \boldsymbol{x}, \boldsymbol{w}) = 1 - \mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w})$$

  a nám tedy opravdu stačí najít pouze model pro $\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w})$.

Sigmoida

- Přešli jsme tedy od funkce s oborem hodnot $\{0, 1\}$ k funkci se spojitým oborem hodnot $[0, 1]$.
- Jak ale donutit lineární výraz

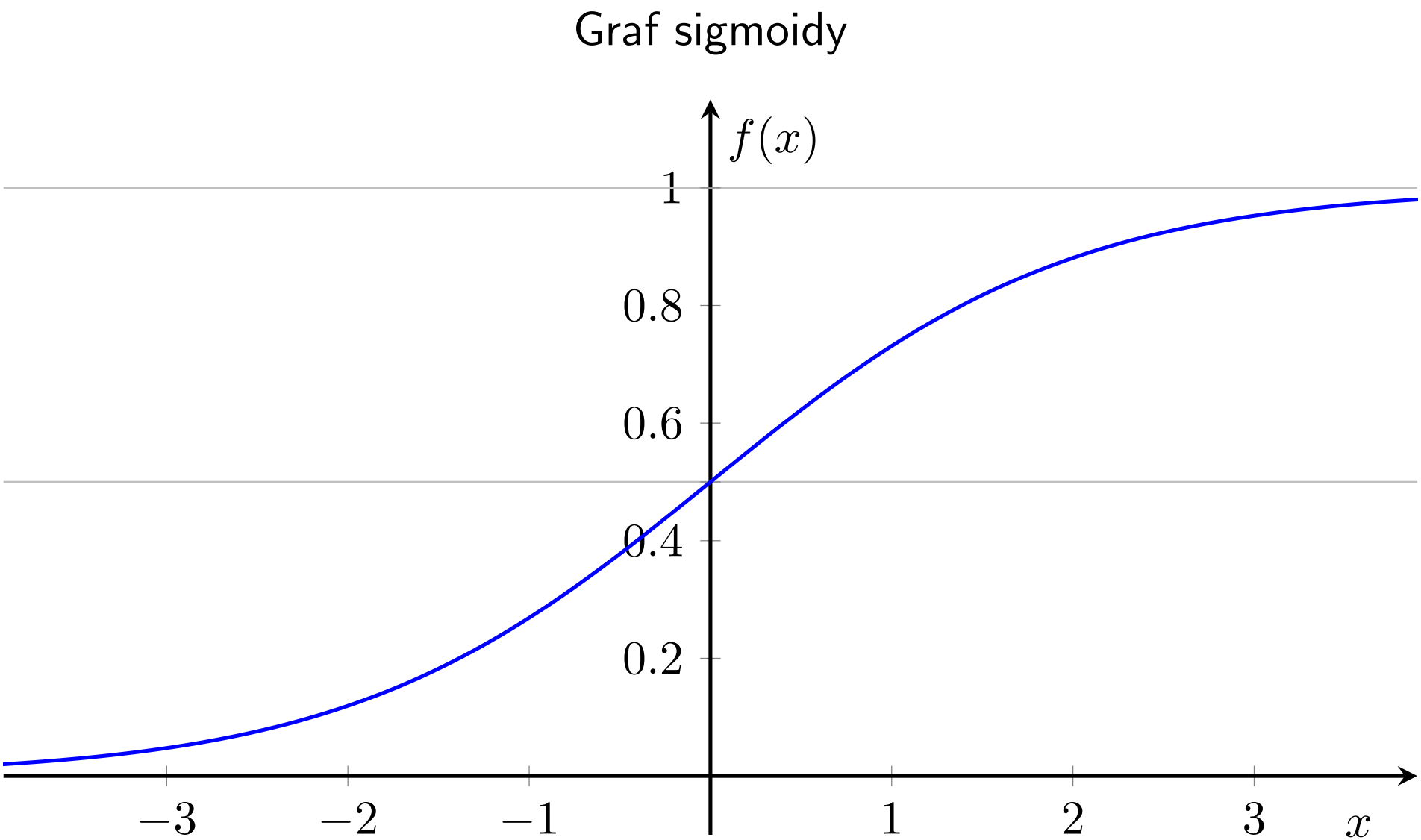
$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3,$$

aby neutekl z tohoto intervalu?

- Zde přichází na řadu druhý trik: **Číslo $w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$ dosadíme do vhodně zvolené funkce, jejíž obor hodnot je podmnožinou intervalu $[0, 1]$.** Tím zajistíme, že pro jakékoli hodnoty koeficientů i příznaků dostaneme číslo smysluplně vyjadřující pravděpodobnost.
- Obvyklou volbou této funkce je **sigmoida** (angl. **sigmoid function**) (což je speciální případ *logistické funkce*):

$$f(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}.$$

Sigmoida



Sigmoida: vlastnosti

$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}.$$

- Definičním oborem je celá množina $\mathbb{R}$, za x tedy můžeme dosadit cokoli; my budeme dosazovat číslo $w_0 + w_1x_1 + w_2x_2 + w_3x_3$.
- Oborem hodnot je interval $(0, 1)$, nikdy se nám tedy nestane, že by pravděpodobnost $Y = 1$ byla jedna (jistý jev) nebo nula (nemožný jev).
- Funkce je ostře rostoucí na $\mathbb{R}$ a tedy prostá. Inverzní funkcí je

$$f^{-1}(x) = \ln \frac{x}{1-x}.$$

- Limity pro $x \to -\infty$ a $x \to +\infty$ jsou 0 resp. 1, platí $f(0) = \frac{1}{2}$ a funkce $f(x) - \frac{1}{2}$ je lichá.
- Číslo $f(x)$ bude pro nás pravděpodobnost jevu $Y = 1$, opačný jev $Y = 0$ tak bude mít pravděpodobnost

$$1 - f(x) = \frac{1}{1+e^x}.$$

Jak to tedy bude celé fungovat?

- Uvažujme náš rýmičkový příklad se třemi příznaky. Z technických důvodů přidáme nultý příznak X_0, který bude mít vždy hodnotu $x_0 = 1$ (viz pojem *intercept* z minulých přednášek).
- Předpokládejme dále, že koeficienty mají tyto hodnoty:

$$\boldsymbol{w} = (w_0, w_1, w_2, w_3) = (0.1, -0.3, -0.2, 0.2),$$

  zatím ponechme stranou, jak se koeficienty hledají.
- Předpokládejme, že máme 35letého muže, který má teplotu 37.2 stupně Celsia. Tj. příslušný vektor s hodnotami příznaků je

$$\boldsymbol{x} = (x_0, x_1, x_2, x_3) = (1, 0, 35, 37.2).$$

- Pravděpodobnost toho, že má rýmičku, dostaneme výpočtem výrazu

$$\boldsymbol{w}^T \boldsymbol{x} = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0.54$$

  a jeho dosazením do sigmoidy

$$\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{e^{0.54}}{1 + e^{0.54}} = 0.631812.$$

  Odhad pravděpodobnosti rýmičky je tedy přes 63 %, tedy je rozumnější rozhodnout, že rýmičku daná osoba má.

Logistická regrese: popis modelu

Jak tedy vypadá model logistické regrese pro binární klasifikaci:

- Máme binární vysvětlovanou proměnnou Y s hodnotami 0 a 1 a p příznaků $X_1, X_2, \ldots, X_p$ s konstantním $X_0 = 1$.
- Hledáme model pro odhad pravděpodobnosti, který pro dané hodnoty $\boldsymbol{x} = (x_0, x_1, \ldots, x_p)$ a pro koeficienty $\boldsymbol{w} = (w_0, w_1, \ldots, w_p)$ má tvar

$$\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}.$$

- Predikce se pak dělá následovně: Pro daná data $\boldsymbol{x}$ se spočítá odhad pravděpodobnosti $\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w})$. Je-li tento odhad větší než $\frac{1}{2}$, rozhodneme se pro $Y = 1$, je-li menší nebo roven $\frac{1}{2}$, pak pro $Y = 0$.

Logistická regrese: hranice rozhodnutí

- Jelikož máme p číselných příznaků, je každý datový bod vlastně bodem v prostoru $\mathbb{R}^p$.
- Máme-li pevně zvolené parametry modelu $\boldsymbol{w}$, můžeme (teoreticky) pro každý bod spočítat hodnotu

$$\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}.$$

  a rozhodnout, jestli je větší než 0.5 ($Y = 1$) nebo menší ($Y = 0$).
- Jak vypadají tyto dvě podmnožiny $\mathbb{R}^p$?
- Nebo jinými slovy: Jak vypadá hranice mezi těmito množinami daná rovnicí

$$\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}} = \frac{1}{2}\,?$$

- Z toho, co jsme si řekli o sigmoidě, víme, že řešením této rovnice je

$$\boldsymbol{w}^T \boldsymbol{x} = w_0 + w_1 x_1 + \cdots + w_p x_p = 0,$$

  což není nic jiného, než **nadrovina** (angl. **hyperplane**) v prostoru $\mathbb{R}^p$.
- Jazykem lineární algebry je to lineární varieta dimenze $p - 1$.

Logistická regrese: hranice rozhodnutí

- Ukázali jsme si, jak model logistické regrese vypadá a funguje, ale zatím nevíme, jak se učí, tj. jak se volí hodnota parametrů $\boldsymbol{w}$ na základě trénovacích dat, u kterých známe vedle příznaků i hodnoty Y.
- U modelů, kde odhadujeme přímo hodnotu proměnné Y, se obvykle postupuje tak, že se rozhodneme pro nějakou míru chyby a parametry vybíráme tak, abychom tuto chybu minimalizovali. Vzpomeňme na přednášku o lineární regresi.
- U logistické regrese ale odhadujeme pravděpodobnosti hodnot proměnné Y; měřit chybu je tedy těžké.
- Musíme postupovat jinak!
- Pro znalé statistiky by stačilo říci, že parametry $\boldsymbol{w}$ odhadneme metodou MLE (**maximálně věrohodný odhad**, angl. **maximum likelihood estimate**).
- Jelikož ale znalost této metody zatím předpokládat nemůžeme, ukážeme si myšlenku MLE na jednoduchém příkladě.

MLE odhad pro házení mincí (1/4)

- Zapomeňme chvilku na logistickou regresi a uvažujme následující příklad.
- Máme minci, kterou házíme a zaznamenáváme si, co nám padlo: 1 = hlava, 0 = orel. Označme si tuto náhodnou veličinu jako Y.
- Máme důvod si myslet, že mince není férová, tedy že je možné, že $\mathrm{P}(Y = 1) \neq \frac{1}{2}$.
- Označme pravděpodobnost $\mathrm{P}(Y = 1) = p$. Chceme na základě trénovacích dat nějak odhadnout hodnotu p.
- Hodíme desetkrát mincí a v nějakém pořadí nám padne sedmkrát $Y = 1$ a třikrát $Y = 0$.
- Naučit model v tomto případě znamená určit hodnotu p, protože to je jediný parametr. Jak na to?

MLE odhad pro házení mincí (2/4)

- Pro každou hodnotu p umíme spočítat, s jakou pravděpodobností nám při deseti hodech padnou naše trénovací data. Např. pro férovou minci s $p = \frac{1}{2}$ je to[1]

$$\left(\frac{1}{2}\right)^7 \cdot \left(\frac{1}{2}\right)^3 = 0.0009765625.$$

- Například pro $p = 0.6$ je ale tato pravděpodobnost vyšší:

$$0.6^7(1 - 0.6)^3 = 0.0017915904.$$

- Z toho důvodu bereme $p = 0.6$ jako lepší model našich trénovacích dat než $p = 0.5$: Pro $p = 0.6$ jsou totiž **trénovací data pravděpodobnější**!
- **Odhad metodou MLE pak odpovídá hodnotě p, pro která jsou trénovací data nejpravděpodobnější možná!**
- Jedná se tedy o optimalizaci (maximalizujeme pravděpodobnost), v tomto případě funkce jedné proměnné $p \in [0, 1]$, která udává pravděpodobnost trénovacích dat:

$$p^7(1 - p)^3.$$

[1] Viz Bernoulliho a binomické rozdělení v BI-PST.

MLE odhad pro házení mincí (3/4)

- Hledáme tedy maximum funkce[2]

$$L(p) = p^7(1-p)^3$$

  na intervalu $[0, 1]$.

- Jedná se o reálnou funkci jedné proměnné, takže můžeme funkci klasicky zderivovat a najít nulové body.
- Funkce L je vlastně polynom desátého stupně v součinovém tvaru, takže by derivace byla celkem pracná. Proto použijeme obvyklý trik a funkci zlogaritmujeme.
- Jelikož je logaritmus ostře rostoucí funkce, mají funkce

$$L(p) = p^7(1-p)^3 \quad \text{a} \quad \ell(p) = \ln p^7(1-p)^3 = 7\ln p + 3\ln(1-p)$$

  extrémy ve stejných bodech.

[2]Sice je to pravděpodobnost, ale obvykle se značí písmenem L od slova likelihood (česky věrohodnost). Více viz BI-PST.

MLE odhad pro házení mincí (4/4)

- Derivace funkce $\ell(p)$ je ale mnohem „hezčí“ funkce:

$$\ell'(p) = \frac{7}{p} - \frac{3}{1-p}.$$

- Snadno spočítáme, že jediný nulový bod derivace je

$$\hat{p}_{\mathsf{MLE}} = \frac{7}{10}.$$

- Pro tuto hodnotu p je pravděpodobnost trénovacích dat

$$0.7^7(1-0.7)^3 = 0.0022235661,$$

  což je maximum možného, pro žádné jiné p není tato pravděpodobnost vyšší.

- Tato vlastnost dává volbě $p = 0.7$ důvěryhodnost.

MLE odhad pro logistickou regresi (1/6)

- S logistickou regresí je to velmi podobné jako pro minci: Máme také jen dvě hodnoty $Y = 1$ a $Y = 0$.
- Máme také daný vzorec pro výpočet pravděpodobnosti; ten ovšem nezávisí na jediném parametru ale na $p + 1$ parametrech $w_0, w_1, \ldots, w_p$. Označme pro úsporu místa

$$p_1(\boldsymbol{x}, \boldsymbol{w}) = \mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}$$

a

$$p_0(\boldsymbol{x}, \boldsymbol{w}) = \mathrm{P}(Y = 0 \mid \boldsymbol{x}, \boldsymbol{w}) = 1 - \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}} = \frac{1}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}.$$

- Máme-li i-tý datový bod s hodnotou vysvětlované proměnné Y_i a s hodnotami příznaků $\boldsymbol{x}_i = (x_0, x_1, x_2, \ldots, x_p)$, lze pro zadané hodnoty parametrů $\boldsymbol{w}$ označit pravděpodobnost tohoto datového bodu jako

$$p_{Y_i}(\boldsymbol{x}_i, \boldsymbol{w}).$$

MLE odhad pro logistickou regresi (2/6)

- Předpokládejme, že máme N bodů v trénovacích datech, každý trénovací bod sestává z vysvětlované proměnné Y_i a hodnot příznaků

$$\boldsymbol{x}_i = (x_{i;0}, x_{i;1}, \ldots, x_{i;p}),$$

  kde $i = 1, \ldots, N$ a $x_{i;0} = 1$ (intercept).
- Tyto hodnoty zapíšeme do vektoru $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_N)^T \in \mathbb{R}^N$ a do matice

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & x_{1;2} & \cdots & x_{1;p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & x_{N;2} & \cdots & x_{N;p} \end{pmatrix}.$$

- S tímto značením můžeme konečně napsat, jaká je pravděpodobnost konkrétních trénovacích dat při parametrech $\boldsymbol{w}$. Předpokládáme (většinou oprávněně), že jednotlivé datové body jsou navzájem nezávislé a pravděpodobnost tak lze napsat jako součin pravděpodobností jednotlivých bodů:

$$L(\boldsymbol{w}) = \prod_{i=1}^{N} p_{Y_i}(\boldsymbol{x}_i, \boldsymbol{w}).$$

MLE odhad pro logistickou regresi (3/6)

- Snažíme se tedy maximalizovat funkci nazývanou **věrohodnostní funkce** (angl. **likelihood function**)

$$L(\boldsymbol{w}) = \prod_{i=1}^{N} p_{Y_i}(\boldsymbol{x}_i, \boldsymbol{w}),$$

  což je reálná funkce $p+1$ proměnných $\mathbb{R}^{p+1} \to \mathbb{R}$ vyjadřující **pravděpodobnost trénovacích dat** pro danou hodnotu parametrů $\boldsymbol{w}$.

MLE odhad pro logistickou regresi (4/6)

- Podobně jako u hodu mincí budeme derivovat místo součinu pravděpodobností logaritmus (tj. logaritmus věrohodnostní funkce):

$$\ell(\boldsymbol{w}) = \ln L(\boldsymbol{w}) = \sum_{i=1}^{N} \ln p_{Y_i}(\boldsymbol{x}_i, \boldsymbol{w}) =$$

$$= \sum_{i=1}^{N} \left(Y_i \ln p_1(\boldsymbol{x}_i, \boldsymbol{w}) + (1 - Y_i) \ln p_0(\boldsymbol{x}_i, \boldsymbol{w})\right) =$$

$$= \sum_{i=1}^{N} \left(Y_i \ln \left(\frac{e^{\boldsymbol{w}^T \boldsymbol{x}_i}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}} \right) + (1 - Y_i) \ln \left(\frac{1}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}} \right) \right) =$$

$$= \{\text{trocha čarování s logaritmem}\} =$$

$$= \sum_{i=1}^{N} \left(Y_i \boldsymbol{w}^T \boldsymbol{x}_i - \ln \left(1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i} \right) \right).$$

MLE odhad pro logistickou regresi (5/6)

- Hledáme tedy maximum funkce

$$\ell(\boldsymbol{w}) = \sum_{i=1}^{N} \left(Y_i \boldsymbol{w}^T \boldsymbol{x}_i - \ln\left(1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}\right)\right).$$

- Stejně jako v případě lineární regrese, kde se minimalizoval součet čtverců residuí $\mathrm{RSS}(\boldsymbol{w})$, se postupuje tak, že se najde gradient, tj. vektor složený z parciálních derivací podle všech proměnných $w_0, w_1, \ldots, w_p$:

$$\frac{\partial \ell}{\partial w_j}(\boldsymbol{w}) = \sum_{i=1}^{N} x_{i;j}\big(Y_i - p_1(\boldsymbol{x}_i, \boldsymbol{w})\big), \quad j = 0, 1, \ldots, p.$$

- Pomocí maticového násobení lze pak gradient napsat ve tvaru

$$\nabla \ell(\boldsymbol{w}) = \mathbf{X}^T(\boldsymbol{Y} - \boldsymbol{P}), \quad \text{kde } \boldsymbol{P} = (p_1(\boldsymbol{x}_1, \boldsymbol{w}), p_1(\boldsymbol{x}_2, \boldsymbol{w}), \ldots, p_1(\boldsymbol{x}_N, \boldsymbol{w}))^T.$$

MLE odhad pro logistickou regresi (6/6)

- Teorie říká, že maximum bychom měli nalézt mezi řešeními rovnice „gradient se rovná nule“, tedy
$$\nabla \ell(\boldsymbol{w}) = \mathbf{X}^T(\boldsymbol{Y} - \boldsymbol{P}) = 0.$$

- Tato rovnice nevypadá nijak strašně, dokud si neuvědomíme, co se skrývá pod nevinným označeným $\boldsymbol{P}$: vektor plný sigmoid se všemi těmi exponenciálami.

- **Na rozdíl od lineární regrese neumíme najít explicitní řešení**, tedy neexistuje vzorec, do kterého bychom dosadili trénovací data a vypadly by nám z něho hodnoty koeficientů $\boldsymbol{w}$.

- **Je tedy nutné použít numerické aproximativní metody**.

- Používají se buď vícerozměrná verze Newtonovy metody, nebo gradientní vzestup: v obou případech se konstruuje posloupnost $\boldsymbol{w}^{(0)}, \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots$ o které lze předpokládat, že konverguje k nějakému lokálnímu maximu[3].

[3] Pro logistickou regresi lze ukázat, že pokud existuje lok. maximum, je jediné a je to hledané globální maximum.

Různé poznámky

- Logistická regrese je přímým použitím metod parametrické statistiky, o kterých budete mluvit podrobně v BI-PST[4].
- Celá metoda stojí na předpokladu, že se chování dat dá zachytit ve tvaru daném sigmoidou jakožto funkcí $w_0 + w_1 x_1 + \cdots + w_p x_p$.
- Díky parametrům $w_0, w_1, w_2, \ldots, w_p$ má tento model poměrně dost volnosti, ale jestli tato volnost stačí k tomu, aby se model mohl dostatečně přiblížit ke skutečnosti, je důležitá a těžko zodpověditelná otázka, kterou je třeba mít na paměti.
- Logistická regrese je výpočetně náročná a příslušný odhad není dán explicitně: To co nám vrátí počítač je tedy aproximace a je i možné, že nám nevrátí nic.
- Stejně jako u regrese a jiných modelů můžeme možnosti modelu značně rozšířit, použijeme-li i odvozené příznaky od příznaků dostupných v datech (druhé a vyšší mocniny, součiny více různých příznaků, atp.).
- Funkce $\ell(\boldsymbol{w})$ nemusí mít maximum! Rozmyslete si, že to platí vždy, když jsou třídy lineárně separabilní, tj. když existuje $\boldsymbol{w}$, tak, že pro všechny body z třídy 1 ($Y_i = 1$) platí $\boldsymbol{w}^T \boldsymbol{x}_i > 0$ a pro všechny body z třídy 0 ($Y_i = 0$) platí $\boldsymbol{w}^T \boldsymbol{x}_i < 0$.

[4] Tak dávejte pozor!

Metoda gradientního vzestupu

- K maximalizaci $\ell(\boldsymbol{w})$ a potažmo k trénovaní logistické regrese můžeme použít metody gradientního vzestupu (angl. **gradient ascent**).
- V takovém případě začneme s nějakou počáteční hodnotou $\boldsymbol{w}^{(0)}$ a pomocí rekurentního vztahu

$$\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)} + \alpha \cdot \nabla \ell\left(\boldsymbol{w}^{(i)}\right) = \boldsymbol{w}^{(i)} + \alpha \cdot \mathbf{X}^T \left(\boldsymbol{Y} - \boldsymbol{P}\left(\boldsymbol{w}^{(i)}\right)\right)$$

  postupně konstruujeme posloupnost vektorů o které doufáme, že konverguje k argumentu globálního maxima $\ell(\boldsymbol{w})$, tj. ke kýženému odhadu $\hat{\boldsymbol{w}}$.
- Protože gradient ukazuje směrem nejvyššího růstu, děláme kroky ve směru nejvyššího růstu.
- Koeficient α je tzv. učící parametr (angl. **learning rate**) a může záviset na i (v tzv. adaptivní verzi).
- I v případě, že globální maximum $\ell(\boldsymbol{w})$ existuje, konvergence může být velmi pomalá a pro nevhodné α ani konvergovat nemusí.

BI-ML1.21 přednáška 9

Daniel Vašata

FIT ČVUT

20. 11. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 20. listopadu 2024 14:28.

Co bude v dnešní přednášce

- Ensemble metody – základní myšlenka
- Bagging – náhodné lesy
- Boosting – AdaBoost

Ensemble metody: základní myšlenka

- Základní myšlenka spočívá v tom, že namísto jednoho modelu (např. rozhodovacího stromu) použijeme **více modelů** a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.
- My si ukážeme dva nejobvyklejší způsoby: **bagging** (bootstrap aggregating) a **boosting**.
- Každý z těchto dvou přístupů si ilustrujeme na nejznámějších reprezentantech, v obou případech spočívajících ve skládání rozhodovacích stromů, které už známe.
- Tito dva reprezentanti jsou: **náhodný les** (angl. **Random Forest**) pro bagging a **AdaBoost** (Adaptive Boosting) pro boosting.

Náhodný les pro klasifikaci: základní myšlenka

Pro jednoduchost předpokládejme, že máme binární klasifikační problém, tj. rozhodujeme jestli $Y = 0$ nebo $Y = 1$.

1. Ze vstupního trénovacího datasetu $\mathcal{D}$ vytvoříme n datasetů $\mathcal{D}_1, \ldots, \mathcal{D}_n$ obvykle stejně velkých jako $\mathcal{D}$ (příp. mohou být i větší či menší) pomocí metody **bootstrap**, neboli pomocí výběru *s opakováním*.

2. Na každém datasetu $\mathcal{D}_i$ naučíme rozhodovací strom tak, jak jsme si ukázali v přednášce o rozhodovacích stromech.

   Typicky pro trénování tohoto stromu použijeme pouze omezenou náhodně vzatou podmnožinu příznaků. Strom může, ale nemusí být málo hluboký, klidně hloubky dva nebo tři (někdy se používá i hloubka jedna).

   Tyto stromy označíme $T_1, \ldots, T_n$ a budeme je nazývat **podmodely** (angl. base learners).

3. Při predikci daný datový bod $\boldsymbol{x}$ proženeme všemi stromy $T_1, \ldots, T_n$ a od každého z nich si uložíme predikci $\hat{Y}_1, \ldots, \hat{Y}_n$.

4. Všechny tyto stromy $T_1, \ldots, T_n$ tvoří **náhodný les** a jeho finální rozhodnutí o hodnotě Y je dané většinovým rozhodnutím stromů, je-li např. v množině $\{\hat{Y}_1, \ldots, \hat{Y}_n\}$ více jedniček než nul, je predikce náhodného lesa $\hat{Y} = 1$.

Bootstrap

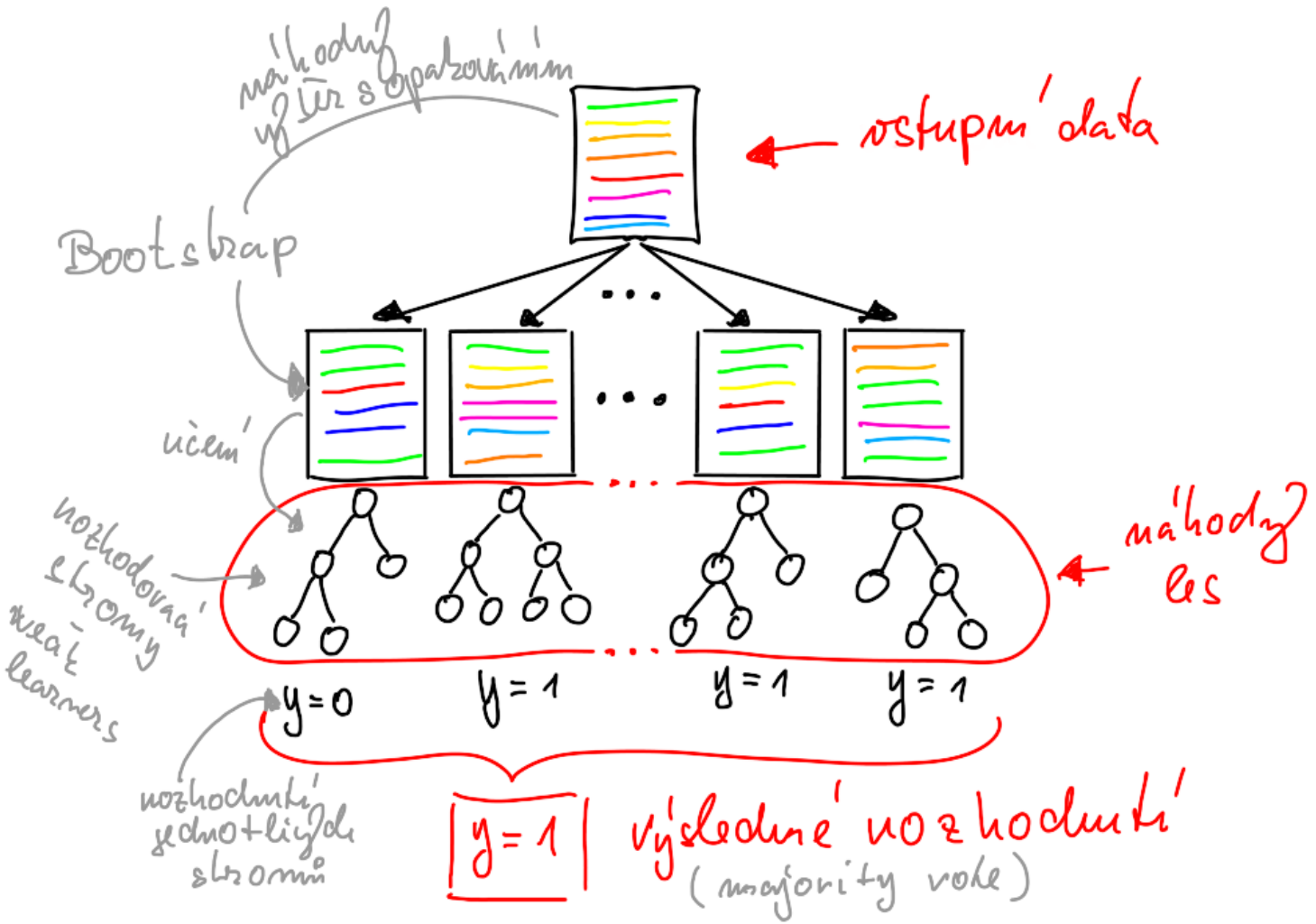
Ukážeme si, jak funguje *bootstrap* na jednoduchém příkladu a našem rýmičkovém datasetu:

id	rýmička	pohlaví	$> 39°$C	vstal(a)?
1	ano	muž	ne	ne
2	ne	žena	ano	ano
3	ne	muž	ne	ano
4	ano	žena	ano	ne

Chceme-li vytvořit „bootstrapem“ dataset velikosti pět, pětkrát si **náhodně vybereme řádek s tabulky** s tím, že se řádky v našem výběru **mohou opakovat**. Vybereme-li např. řádky s `id` 1,4,3,3,1, dostaneme dataset

id	rýmička	pohlaví	$> 39°$C	vstal(a)?
1	ano	muž	ne	ne
4	ano	žena	ano	ne
3	ne	muž	ne	ano
3	ne	muž	ne	ano
1	ano	muž	ne	ne

Náhodný les pro klasifikaci: základní myšlenka na obrázku



Predikce náhodného lesa

- V případě regresního problému (spojité Y) se postupuje opět analogicky jako u regresního rozhodovacího stromu: predikce náhodného lesa se bere jako průměr z predikcí jednotlivých stromů lesa:

$$\hat{Y} = \frac{1}{n} \sum_{i=1}^{n} \hat{Y}_i.$$

- Implementace `RandomForestClassifier` ve `scikit-learn` pracuje na místo predikcí tříd s predikcemi pravděpodobností tříd od jednotlivých podmodelů (relativní četnosti tříd v listech).
- Tj. např. u binární klasifikace, pokud označíme jako $\hat{p}_i = \hat{\mathrm{P}}(Y = 1 | T_i, \boldsymbol{X} = \boldsymbol{x})$ predikci pravděpodobnosti příslušnosti bodu $\boldsymbol{x}$ ke třídě 1 podmodelu T_i, pak finální predikovaná pravděpodobnost třídy 1 pro náhodný strom je určena průměrem:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} \hat{p}_i.$$

- Finální predikce vysvětlované proměnné Y je pak určena obvyklým porovnáním této pravděpodobnosti s číslem $1/2$ jako

$$\hat{Y} = \begin{cases} 1 & \text{pro} \quad \hat{p} > 0.5, \\ 0 & \text{jinak.} \end{cases}$$

Základní hyperparametry

Základními hyparametry metod `RandomForestClassifier` a `RandomForestRegressor` v `sklearn` jsou:

- `n_estimators`: určuje počet stromů v náhodném lese,
- `max_depth`: určuje maximální hloubku stromů v lese (typicky spíše nižší hodnota),
- `max_features`: počet (náhodně vybraných) příznaků, ze kterých si hladový algoritmus vybírá ten, podle kterého bude v aktuálním kroku data „větvit“. Defaultně se volí hodnota $\sqrt{p}$, tj. odmocnina počtu příznaků,
- je možné nastavovat i další obvyklé hyperparametry rozhodovacích stromů jakožto podmodelů.

Poznámky

- U baggingu, který skládá rozhodnutí z více podmodelů, je vhodné, aby jednotlivé podmodely nebyly stejné, ale naopak co nejpestřejší.
- Toho se jak docílí nějakým druhem randomizace; v případě náhodných lesů je tento náhodný prvek daný bootstrap metodou pro generování jednotlivých trénovacích datasetů a také hodnotou parametru `max_features`.
- Jak víme, rozhodovací stromy jsou velice citlivé na změny v trénovacích datech (mají velký rozptyl), a tak často stačí odlišnost datasetů daná bootstrapem, abychom získávali velice odlišné rozhodovací stromy.
- Tím, že při baggingu počítáme průměry predikcí těchto rozdílných podmodelů snažíme se redukovat rozptyl (a povětšinou se to úspěšně daří).
- Přestože mohou být jednotlivé stromy samy o sobě slabé modely (podmodelům v ensemble metodách se proto často říká **weak learners**), jejich kolektivní rozhodování dává až překvapivě dobré výsledky.
- Náhodné lesy jsou na rozdíl od rozhodovacích stromů velice robustní a poměrně odolné vůči přeučení. Bohužel ale částečně ztrácejí jejich jednoduchost a snadnou interpretovatelnost.

Rozhodovací stromy a `sample_weight` (1/2)

- Abychom mohli vysvětlit, jak funguje boosting a konkrétně algoritmus AdaBoost, potřebujeme pochopit použití vážených dat u rozhodovacích stromů.

- V řeči `sklearn` se jedná o použití hyperparametru `sample_weight` v metodě `.fit(X,y,sample_weight)` tříd `DecisionTreeClassifier` a `DecisionTreeRegresor`.

- Pokud máme v trénovací množině N datových bodů (řádků v tabulce, poloangl. „samplů“), je proměnná `sample_weight` pole $w_1, \ldots, w_N$ nezáporných vah jednotlivých bodů.

- Tyto váhy určují, jak moc je který bod „důležitý“. Defaultní hodnotou je $w_i = 1$ pro $i = 1, \ldots, N$. V některých případech bývají znormované na jedničku, ale nemusí to platit.

Rozhodovací stromy a `sample_weight` (2/2)

- Při učení stromu se váhy projeví v kroku, ve kterém se počítá informační zisk

$$H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R)$$

  kde $t_L = \frac{\#\mathcal{D}_L}{\#\mathcal{D}}$ a $t_R = \frac{\#\mathcal{D}_R}{\#\mathcal{D}}$.

- Konkrétně se váženým způsobem napočítají jak entropie $H(\mathcal{D})$, $H(\mathcal{D}_L)$ a $H(\mathcal{D}_R)$, tak hodnoty t_L a t_R.

  Např. pro výpočet entropie $H(\mathcal{D}_L)$ se v případě binární klasifikace použije odhad pravděpodobnosti třídy 1 určený součtem vah bodů ve třídě 1, které spadají do $\mathcal{D}_L$, poděleným sumou vah všech bodů v $\mathcal{D}_L$.

  Dále např. hodnota t_L je rovna součtu vah bodů, které spadají do $\mathcal{D}_L$, poděleném sumou vah všech bodů z $\mathcal{D}$.

- Jsou-li tedy v $\mathcal{D}_L$ body z řádků s indexy $3, 10, 15, 25$ a odpovídajícími třídami $0, 1, 1, 1$ a v $\mathcal{D}$ body s indexy $3, 10, 15, 17, 21, 25$, dostáváme:

$$p_1 = \frac{w_{10} + w_{15} + w_{25}}{w_3 + w_{10} + w_{15} + w_{25}} \quad \text{spolu s} \quad H(\mathcal{D}_L) = -p_1 \log p_1 - (1 - p_1) \log(1 - p_1)$$

  a

$$t_L = \frac{w_3 + w_{10} + w_{15} + w_{25}}{w_3 + w_{10} + w_{15} + w_{17} + w_{21} + w_{25}}$$

Výsledkem použití vah je to, že strom se učí tak, aby správně predikoval zejména datové body s vyšší vahou. (Toto si rozmyslete!)

AdaBoost (Adaptive Boosting): základní myšlenka

- Stejně jako při baggingu konstruujeme více podmodelů (opět uvažujeme rozhodovací stromy, i když AdaBoost může používat i jiné typy modelů) a finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.
- Na rozdíl od baggingu ale při **boostingu** nejsou tyto modely nezávislé, ale jsou seřazené a každý další je ovlivněn těmi předchozími.
- Tento vliv je při AdaBoostu realizován pomocí vah datových bodů: Při konstrukci n-tého stromu je zvýšena váha těm bodům, které předchozí $(n-1)$-tý strom klasifikoval špatně.
- Takovýmito změnami vah je zajištěno, že se další model soustředí více na ty datové body, se kterými si předchozí modely neporadily.
- Přibližně takto funguje boosting obecně, my si dále ukážeme konkrétní implementaci této myšlenky známou jako algoritmus **AdaBoost** [Freund, Schapire (1997)].

AdaBoost (Adaptive Boosting): popis algoritmu (1/3)

- Na začátku máme dataset $\mathcal{D}$ s N datovými body.
- Pro jednoduchost opět uvažujeme binární klasifikaci. Existují ale i modifikace algoritmu pro více než dvě třídy a i pro regresi.
- Počet zkonstruovaných stromů je zadán uživatelem v hyperparametru `n_estimators`.

AdaBoost:

1. Nastavme váhy rovnoměrně, tedy $w_i = \frac{1}{N}$ a položme $m = 1$.
2. Pokud $m \leq$ `n_estimators`, naučme strom $T^{(m)}$ na datech $\mathcal{D}$ s váhami w_i.
3. Do proměnné $e^{(m)}$ uložme součet vah těch bodů z $\mathcal{D}$, které jsou špatně klasifikované stromem $T^{(m)}$.
4. Pokud je $e^{(m)} = 0$ skončeme, všechna data jsou klasifikována správně.

AdaBoost (Adaptive Boosting): popis algoritmu (2/3)

5. Položme

$$\alpha^{(m)} = \texttt{learning\_rate} \cdot \log \frac{1 - e^{(m)}}{e^{(m)}},$$

   kde `learning_rate` je hyperparametr zadaný uživatelem; používá se ke zpomalení trénování a k zabránění přeučení (je-li menší než jedna).

6. Pro stromem $T^{(m)}$ špatně klasifikované body nastavme nové váhy[1]

$$w_i \leftarrow w_i \exp(\alpha^{(m)}).$$

7. Znormalizujme váhy tak, aby jejich součet byl jedna.

8. Zvětšíme m o jedna a vraťme se do bodu 2.

Výsledkem algoritmu je tedy až `n_estimators` rozhodovacích stromů $T^{(1)}, T^{(2)}, \ldots$

[1] Výraz $\exp(x)$ značí staré známé e^x. Často se to v literatuře zapisuje takto, tak si zvykejte ;).

AdaBoost (Adaptive Boosting): popis algoritmu (3/3)

Abychom určili rozhodnutí tohoto modelu pro nějaký datový bod $\boldsymbol{x}$, postupujeme následovně:

1. Každému stromu $T^{(m)}$ přiřadíme váhu danou číslem $\alpha^{(m)}$ z kroku 5.

2. Sečti váhy $\alpha^{(m)}$ všech stromů, které pro $\boldsymbol{x}$ predikují $Y = 1$ a to samé udělej pro stromy predikující $Y = 0$.

3. Rozhodni se pro tu z možností, pro kterou je součet vah vyšší.

Poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá *AdaBoost.R2* [Drucker (1997)].
- AdaBoost nemusí nutně používat rozhodovací stromy; je možné použít jakýkoli model, který umí pracovat s parametrem `sample_weight`.
- V `sklearn` implementaci jsou rozhodovací stromy (hloubky 3) výchozí volba (parametr `base_estimator`).
- Parametr `learning_rate` je obvykle zaváděn u většiny ensemble metod spadajících do kategorie Boostingu.
- Jedná se o tzv. **regularizaci**: čím je `learning_rate` nižší, tím je model odolnější vůči přeučení. Nevýhodou je, že je pak obvykle nutné zvýšit počet stromů (parametr `n_estimators`).
- Při boostingu dochází k postupnému korigování chyb předchozích modelů, což vede na redukci vychýlení (bias). Podmodely se typicky konstruují jako slabé modely (weak learners) - tj. např. jako poměrně mělké stromy.

BI-ML1.21 přednáška 10

Daniel Vašata

FIT ČVUT

27. 11. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 27. listopadu 2024 10:24.

Co bude v dnešní přednášce

- Výběr příznaků obecně
- Filtrovací metody
- Obalové metody
- Vestavěné metody - Lasso

Redukce počtu vysvětlujících proměnných

- V dnešní přednášce se budeme věnovat metodám, které mají za úkol snížit počet vysvětlujících proměnných.
- V zásadě jde o to, vybrat ze všech dostupných příznaků (které mohly být i extra napočítány) jistou podmnožinu, kterou použijeme pro trénování a predikci modelu.
- Obecně v této situaci mluvíme o **výběru příznaků** (angl. **feature selection** nebo také **subset selection**).
- Problematika výběru příznaků spadá do části předzpracování dat. Dneska se podíváme jenom na některé základní metody a podrobněji se touto problematikou budete zabývat v magisterském kurzu **NI-PDD**.
- Z jistého pohledu se jedná o podoblast redukce dimenzionality (angl. dimensionality reduction). Do té ale spadají především metody, které napočítávají (lineárně nebo nelineárně) kompletně nové příznaky. Proto se redukce dimenzionality často uvádí jako separátní oblast.

Účel výběru příznaků

Výběr příznaků může být výhodný z mnoha různých důvodů. Uveďme si některé z nich.

- Zahozením nerelevantních a redundantních příznaků můžeme významně zlepšit schopnost generalizace modelu.
  - Příznaky, které jsou nerelevantní a nesouvisí s vysvětlovanou proměnnou, škodí typicky i modelům, které by si s nimi teoreticky dokázali poradit - jako např. lineární regrese (může tam použít koeficient 0) nebo rozhodovací strom (nepoužije tento příznak do žádného dělícího kritéria).
  - Redundantní příznaky jsou takové, které nesou stejnou informaci. Kromě toho, že zbytečně zvyšují dimenzi mohou některým modelům vyloženě škodit, jako např. problém kolinearity u lineární regrese.
- Pomáhá s prokletím dimenzionality. Jak víme, tak pro některé modely je velká dimenze prostoru příznaků problematická (např. KNN).
- Zlepšuje vysvětlitelnost modelu. U modelů, který využívají menší počet příznaků bývá jednodušší porozumět tomu, na základě čeho se rozhodují.
- Snižuje výpočetní nároky pro trénování a použití výsledného modelu.

Filtrační metody

Filtrační metody (angl. **filter methods**) mohou být supervizované i nesupervizované.

Typicky koukají, jak hodně informace může být v daném příznaku a případně jak ta informace souvisí s vysvětlovanou proměnnou.

Pokud nemáme nebo nechceme využít vysvětlovanou proměnnou, můžeme:

- Vyhodit příznaky, které mají příliš nízký rozptyl a jsou tedy téměř konstantní.
- Vyhodit příznaky, které mají příliš chybějících hodnot.
- Vyhodit některé z příznaků, které spolu hodně korelují a jsou tedy redundantní.

Pokud chceme využít vysvětlovanou proměnnou, můžeme:

- Vyhodit příznaky, které mají nízkou korelaci s vysvětlovanou proměnnou.
- U binárních příznaků rozdělit vysvětlovanou na dvě populace a udělat test hypotézy o rovnosti středních hodnot (dvouvýběrový t-test). Pokud vyjde velká p hodnota, můžeme tento příznak vyhodit.
- U diskrétních příznaků i diskrétní vysvětlované proměnné udělat test hypotézy o nezávislosti těchto dvou diskrétních veličin (např. chí kvadrát test nezávislosti). Opět, pokud vyjde velká p hodnota, je tento příznak kandidátem na vyhození.

Obalové metody (1/2)

- **Obalové metody** (angl. **wrapper methods**) využívají nějaký pomocný model pro ohodnocování podmnožin příznaků.
- Cílem je vybrat takovou podmnožinu, pro kterou je výkonnost modelu největší.
- Pro každou kandidátní podmnožinu se model natrénuje a změří jeho výkonnost na validační množině (nebo křížovou validací).
- Jako finální podmnožina příznaků je použita taková, pro kterou je model nejvýkonnější.
- Výhodou obalových metod je, že preferují podmnožiny příznaků, které jsou pro daný model výhodné.
- To je ale zároveň nevýhoda, protože může snadno dojít k přeučení a také protože takto vybrané podmnožiny nemusí být vhodné pro jiné modely (např. pro ten finální, který chceme použít).
- Nevýhodou obalových metod je jejich výpočetní náročnost. I při použití jednoduchého pomocného modelu (jako např. lineární regrese) může být výpočetní čas značný.

Obalové metody (2/2)

Z důvodů výpočetní náročnosti nejčastěji používané metody vybranou množinu příznaků konstruují iterativně nějakým „hladovým“ přístupem.

- **Dopředný výběr** (angl. **forward selection**) - začínáme s prázdnou množinou příznaků a postupně po jednom přidáváme vždy tak, aby ten jeden přidaný nejvíce zlepšil výkonnost modelu v dané iteraci.

  Skončíme když máme požadovaný počet příznaků, případně když už nedochází ke zlepšování výkonnosti.

- **Zpětný výběr** (angl. **backward selection**) - začneme se všemi příznaky a postupně po jednom odebíráme tak, aby ten odebraný příznak vedl na nejvýkonnější model v dané iteraci.

  Skončíme když máme požadovaný počet příznaků, případně když už nedochází ke zlepšování výkonnosti.

- Dopředný i zpětný výběr jsou v knihovně `sklearn` implementovány pomocí `sklearn.feature_selection.SequentialFeatureSelector`.

Obalové metody (3/3)

- **Rekurzivní odebírání příznaků** (angl. **recursive feature elimination**) - probíhá podobně jako zpětný výběr, ale model se využívá trochu jiným způsobem.

  Konkrétně se k vybírání příznaků používá vnitřní ohodnocení důležitosti jednotlivých příznaků modelem (který toho tedy musí být schopen).

  U lineární (logistické) regrese to je například absolutní hodnota koeficientu u daného příznaku. U rozhodovacího stromu to může být informační zisk daného příznaku.

  Nejprve model natrénujeme pro všechny příznaky. Potom najdeme příznak, který má nejmenší důležitost a ten vyhodíme. Pak model znovu natrénujeme na redukované množině a postup opakujeme.

- Rekurzivní odebírání příznaků je v knihovně `sklearn` implementováno pomocí `sklearn.feature_selection.RFE`.

Vestavěné metody

- **Vestavěné metody** (angl. **embedded methods**) využívají model, který se trénuje pouze jednou na celých datech a při tom implicitně provede výběr příznaků.
- Tento implicitní výběr se projeví tak, že se model naučí některé příznaky vůbec nevyužívat.
- Např. u lineární regrese jsou příslušné koeficienty odhadnuty jako 0.
- Modelem, který je nejpoužívanější, je L_1 regularizovaná lineární regrese (Lasso). V případě klasifikace to je pak L_1 regularizovaná logistická regrese.
- Jako vybraná podmnožina příznaků se vezmou příznaky, u který jsou příslušné koeficienty nenulové.

Model lineární regrese

Nejprve si připomeňme model lineární regrese.

- Model pro vysvětlovanou proměnnou Y v bodě $\boldsymbol{x}$ je

$$Y = \boldsymbol{w}^T\boldsymbol{x} + \varepsilon = \boldsymbol{x}^T\boldsymbol{w} + \varepsilon = w_0 + w_1x_1 + \dots w_px_p + \varepsilon.$$

- Model pro trénovací množinu tvořenou N páry $(Y_i, \boldsymbol{x}_i)$ je $Y_i = \boldsymbol{x}_i^T\boldsymbol{w} + \varepsilon_i$.
- Toto zapisujeme maticově jako $\boldsymbol{Y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$, kde

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & \cdots & x_{1;p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & \cdots & x_{N;p} \end{pmatrix}, \quad \boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}.$$

- Při trénování metodou nejmenších čtverců minimalizujeme residuální součet čtverců

$$\mathrm{RSS}(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2.$$

- Za předpokladu, že je matice $\mathbf{X}^T\mathbf{X}$ regulární, existuje jediné řešení minimalizující $\mathrm{RSS}(\boldsymbol{w})$,

$$\hat{\boldsymbol{w}}^{\mathrm{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

Lasso - úvod

Lasso (zkr. angl. **Least Absolute Shrinkage and Selection Operator**) [Tibshirani (1996)] nebo taky L_1 regularizace zavádí penalizační člen úměrný součtu absolutních hodnot složek vektoru koeficientů $\boldsymbol{w}$ s vynecháním interceptu.

Minimalizujeme tedy **regularizovaný reziduální součet čtverců**

$$\mathrm{RSS}_\lambda^{\mathsf{Lasso}}(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \sum_{i=1}^{p} |w_i|,$$

který závisí na parametru $\lambda \geq 0$.

- Lasso jakožto jiná forma regularizace sdílí některé obecné vlastnosti s modelem hřebenové regrese.
- Pro $\lambda = 0$ dostáváme $\mathrm{RSS}_0^{\mathsf{Lasso}}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w})$ a máme tedy obyčejnou metodu nejmenších čtverců.
- Pro $\lambda > 0$ je vidět, že v minimu se bude cílit na takové vektory $\boldsymbol{w}$, které mají co nejmenší složky.
- Hodnotu w_0 interceptu nijak nepenalizujeme. Jedná se pouze o vertikální posun, který zajišťuje předpoklad $\mathrm{E}\,\varepsilon = 0$ modelu a je tedy vhodné ho neomezovat.

Lasso - pokračování

- Na rozdíl od modelu hřebenové regrese není regularizační člen $\sum_{i=1}^{p} |w_i|$ diferencovatelný v bodech kde $w_j = 0$.
- V tomto případě není možné nalézt explicitní řešení a existují pouze iterativní metody, které najdou řešení

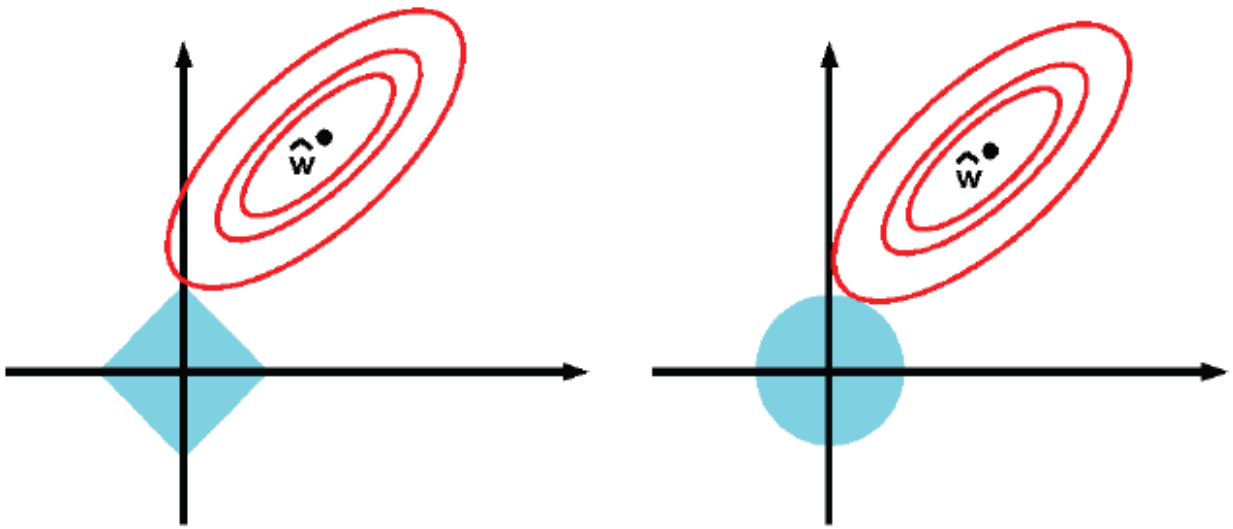
$$\hat{\boldsymbol{w}}_\lambda^{\text{Lasso}} = \arg\min_{\boldsymbol{w}} \text{RSS}_\lambda^{\text{Lasso}}(\boldsymbol{w}).$$

- Výhoda modelu Lasso je, že řešení je tzv. **řídké** (angl. **sparse solution**).
- To znamená, že odhady $\hat{\boldsymbol{w}}_{\lambda;j}^{\text{Lasso}}$ některých složek $\boldsymbol{w}$ jsou rovny 0.

  Samozřejmě tím častěji, čím je λ větší.
- Tím se lasso zásadně liší od hřebenové regrese, kde tento efekt v podstatě nikdy nenastane.
- Formálně to dokázat není jednoduché (používá se teorie subgradientů), ale ukážeme si alespoň přesvědčivou ilustraci.

Lasso - ilustrace fungování

Podívejme se na „vrstevnice“ funkcí, které se snažíme minimalizovat u lasso (vlevo) a hřebenové regrese (vpravo).



Červeně jsou znázorněny vrstevnice odpovídající neregularizované části $\|Y - \mathbf{X}\boldsymbol{w}\|^2$, což je v zásadě parabolická jáma.

Modře je pak znázorněna oblast, která odpovídá regularizačnímu členu.

Vidíme, že u lasso existuje velká šance, že se vrstevnice dotýká některého z rohů, což znamená, že je jedna ze složek $\boldsymbol{w}$ rovna 0.

U hřebenové regrese se to samozřejmě také může stát, ale evidentně to je velmi nepravděpodobné.

Lasso a hřebenová regrese - vázané optimalizace

Fakt z předchozí ilustrace může být ještě podpořen následující ekvivaletní formulací. Lze ukázat, že úloha minimalizace

$$\mathrm{RSS}_\lambda^{\mathsf{Lasso}}(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \sum_{i=1}^{p} |w_i|,$$

je ekvivalentní úloze vázané minimalizace

$$\mathrm{RSS}(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2,$$

za podmínky

$$\sum_{i=1}^{p} |w_i| \leq B, \quad \text{pro nějaké } B > 0.$$

Analogicky pro hřebenovou regresi platí, že minimalizace

$$\mathrm{RSS}_\lambda^{\mathsf{Ridge}}(\boldsymbol{w}) = \|\boldsymbol{Y} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \sum_{i=1}^{p} w_i^2,$$

je ekvivalentní úloze vázané minimalizace $\mathrm{RSS}(\boldsymbol{w})$ za podmínky

$$\sum_{i=1}^{p} w_i^2 \leq B, \quad \text{pro nějaké } B > 0.$$

Porovnání nejmenších čtverců, hřebenové regrese a lasso

Pro porovnání metod regularizace se zaměřme na situaci, kdy jsou příznaky ortonormální. Tj. když $\mathbf{X}^T\mathbf{X} = \mathbf{I}$.

- Metoda nejmenších čtverců v takovém případě vychází

$$\hat{\boldsymbol{w}}^{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y} = \mathbf{X}^T\boldsymbol{Y}.$$

- Hřebenová regrese vychází

$$\hat{\boldsymbol{w}}_\lambda^{\text{Ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}')^{-1}\mathbf{X}^T\boldsymbol{Y} = (\mathbf{I} + \lambda\mathbf{I}')^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

Protože

$$(\mathbf{I} + \lambda\mathbf{I}')^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \frac{1}{1+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{1+\lambda} \end{pmatrix},$$

platí pro jednotlivé složky

$$\hat{\boldsymbol{w}}_{\lambda;0}^{\text{Ridge}} = \hat{\boldsymbol{w}}_0^{\text{OLS}} \quad \text{a} \quad \hat{\boldsymbol{w}}_{\lambda;j}^{\text{Ridge}} = \frac{1}{1+\lambda}\hat{\boldsymbol{w}}_j^{\text{OLS}}.$$

- Pro Lasso lze ukázat

$$\hat{\boldsymbol{w}}_{\lambda;0}^{\text{Lasso}} = \hat{\boldsymbol{w}}_0^{\text{OLS}} \quad \text{a} \quad \hat{\boldsymbol{w}}_{\lambda;j}^{\text{Lasso}} = \operatorname{sgn}\left(\hat{\boldsymbol{w}}_j^{\text{OLS}}\right) \max\left(0, |\hat{\boldsymbol{w}}_j^{\text{OLS}}| - \tfrac{\lambda}{2}\right).$$

Tomuto se říká soft thresholding.

Závěrečné poznámky

- Lasso je v knihovně `sklearn` implementováno pomocí `sklearn.linear_model.Lasso`.
- Při využití pro výběr příznaků se dá použít implementace `sklearn.feature_selection.SelectFromModel`.
- U logistické regrese bychom regularizační člen přidali k mínus logaritmu věrohodnostní funkce (k binární relativní entropii) a pak bychom prováděli minimalizaci.
- Jednou z nevýhod lasso proti hřebenové regresi je, že v případě kolineárních příznaků má tendenci vybrat pouze některé z nich. To se v některých případech při predikci na nových datech ukazuje jako jistá nevýhoda oproti hřebenové regresi, která typicky využije všechny příznaky.
- Existuje tedy také model, který obsahuje oba způsoby regularizace.
- Tomuto modelu se říká **elastic net** a kombinuje výhody obou přístupů.

BI-ML1.21 přednáška 11

Daniel Vašata

FIT ČVUT

4. 12. 2024

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 27. listopadu 2024 10:31.

Co bude v dnešní přednášce

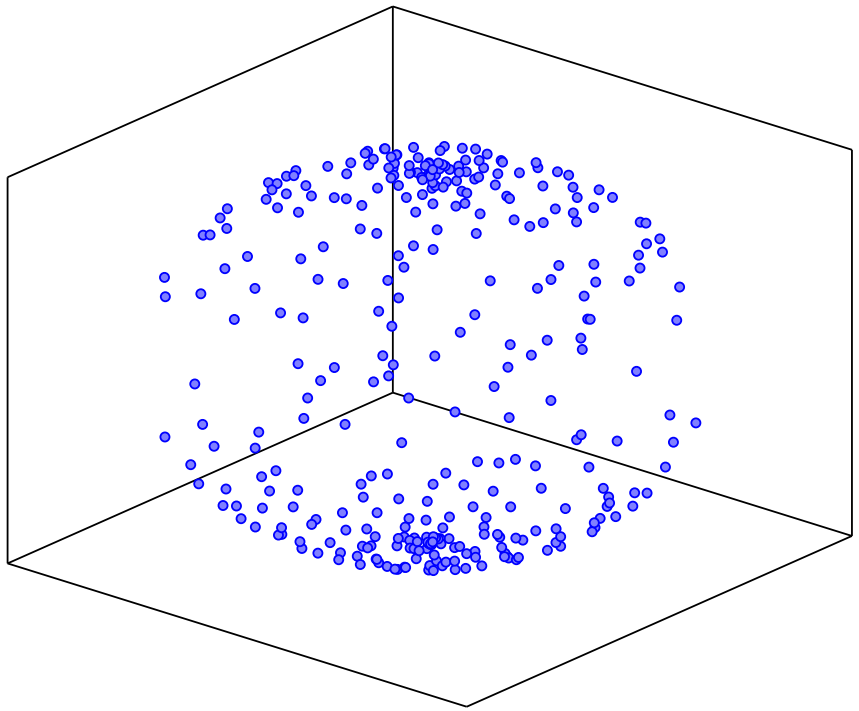
- Principy nesupervizovaného učení.
- Úvod do shlukové analýzy (clusterové analýzy).
- Hierarchické shlukování.
- Nehierarchické shlukování – algoritmus k-means.

Nesupervizované učení

- Nastává v situaci, kdy data nemáme nikterak označena. Tj. nemáme žádnou veličinu, kterou bychom u trénovacích dat znali a snažili se ji naučit predikovat.
- Cílem nesupervizovaného učení je porozumět struktuře dat pouze na základě jich samotných. To znamená bez nějakého vnějšího vodítka.
- Proto se nesupervizovanému učení také říká **učení bez učitele**.
- Porozuměním zde typicky myslíme nalezení co „nejmenších“ oblastí v prostoru příznaků, kde se data vyskytují nejčastěji.
- Obvykle totiž platí, že se naměřená skutečná data nevyskytují v celém prostoru stejně pravděpodobně, ale bývají více či méně lokalizována - tvoří nějaké shluky, vyskytují se v méně-dimenzionálních oblastech atd.
- Porozumění této lokalizaci přináší důležitou informaci o vnitřní struktuře dat!

Nesupervizované učení

Uvažujme data rozložená v třírozměrném prostoru podle následujícího obrázku.



Podaří-li se nám zjistit, že jsou všechny tyto body na kouli, můžeme jejich polohu popsat pomocí sférických souřadnic a získat tak ekvivalentní reprezentaci pomocí dvou spojitých příznaků (tzv. redukce dimenzionality).

Navíc je možné detekovat, že hustota bodů na pólech je vyšší než hustota bodů na rovníku.

Obecným problémem nesupervizovaného učení je, že vůbec není jasné, jak bychom měli vyhodnocovat úspěšnost získaného porozumění.

To je velký rozdíl oproti supervizovanému učení, kde je možné kvalitu naučeného modelu vyhodnocovat mnoha víceméně rovnocennými způsoby (např. přesností u klasifikace).

Nesupervizované učení z pohledu teorie pravděpodobnosti

Uvažujme situaci, kdy naše data obsahují p příznaků a označme $\mathcal{X}$ prostor ve kterém se nacházejí možné výsledky[1].

- Pro p binárních příznaků volíme $\mathcal{X} = \{0, 1\}^p$.
- Pro p spojitých příznaků typicky volíme $\mathcal{X} = \mathbb{R}^p$.

Z pohledu teorie pravděpodobnosti a statistiky (viz BI-PST) chápeme pozorovaná data jako **realizace náhodného vektoru** $\boldsymbol{X} = (X_1, \ldots, X_p)^T$.

Porozumění vnitřní struktuře pak znamená porozumění rozdělení $\boldsymbol{X}$. Chceme tedy získat **odhad pravděpodobnosti** $\mathrm{P}(\boldsymbol{X} \in O)$ pro každou (rozumnou) podmnožinu $O \subset \mathcal{X}$.

- V případě spojitých příznaků to ekvivalentně znamená odhadnout sdruženou **hustotu pravděpodobnosti** $f_{\boldsymbol{X}}(\boldsymbol{x}) \equiv f_{\boldsymbol{X}}(x_1, \ldots, x_p)$.
- V případě diskrétních příznaků to ekvivalentně znamená odhadnout sdruženou **pravděpodobnostní funkci** $p_{\boldsymbol{X}}(\boldsymbol{x}) \equiv \mathrm{P}(X_1 = x_1, \ldots, X_p = x_p)$.

Soustředíme se přitom zejména na nalezení oblastí v prostoru $\mathcal{X}$, které mají velkou pravděpodobnost a jsou při tom „co nejmenší“.

[1] Volba $\mathcal{X}$ není jednoznačná a je součástí volby modelu.

Úvod do shlukové analýzy

Jednou ze základních metod nesupervizovaného učení je **shlukování** nazývané také clusterování (angl. **clustering**).

Naším cílem je roztřídit data do skupin, které budeme nazývat **shluky**, tak, že platí dva přirozené požadavky:

- blízké body budou ve stejném shluku a
- vzdálené body budou v různých shlucích.

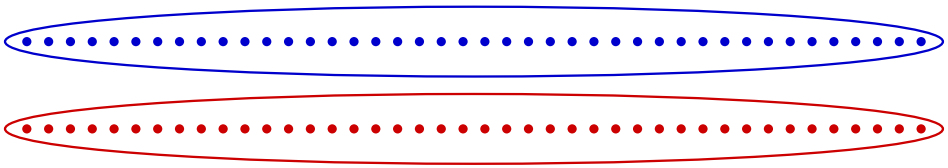
Tento popis je ovšem hodně vágní a není vůbec jasné, jak ho zformalizovat.

Jak uvidíme na následujícím slajdu, jedním z problémů je fakt, že tyto intuitivní požadavky obecně nejsou kompatibilní.

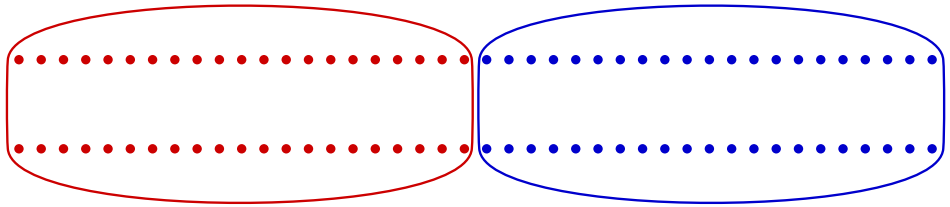
Dalším problémem je již dříve zmiňovaná neexistence hodnotícího kritéria kvality nalezeného shlukování.

V důsledku absence jednoznačnosti tudíž existuje více způsobů formalizace a následně také mnoho různých shlukovacích algoritmů, které mohou na stejných datech dávat velmi rozdílné výsledky.

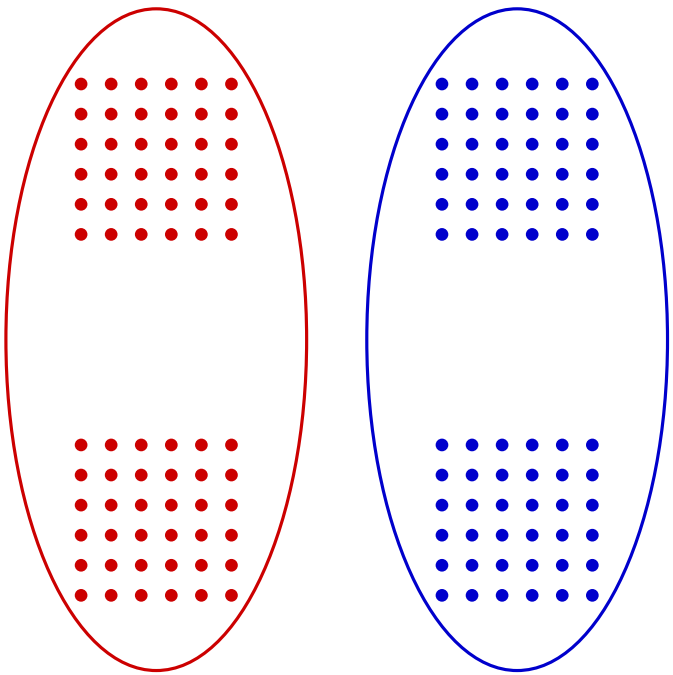
Shluková analýza - znázornění nejasných situací

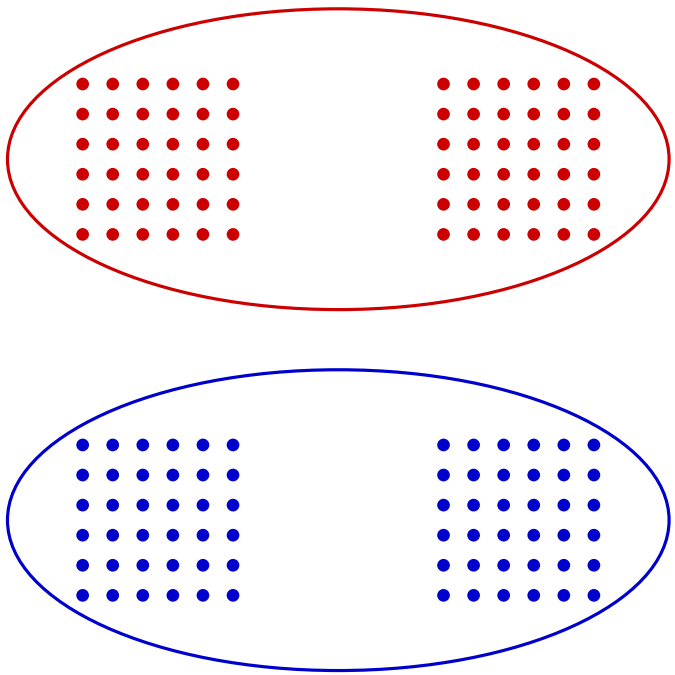


Shlukování, které se snaží udržet blízké body ve stejných shlucích.



Shlukování, které se snaží nemít v jednotlivých shlucích příliš vzdálené body.





Obě shlukování jsou stejně legitimní a není jasné, které vybrat.

Vzdálenost

Klíčovým pojmem, na kterém stojí shlukování, je vzdálenost, kterou jsme si zaváděli na třetí přednášce.

Definice

Vzdálenost nebo také **metrika** na množině $\mathcal{X}$ je funkce $d : \mathcal{X} \times \mathcal{X} \to [0, +\infty)$ taková, že pro každé $x, y, z \in \mathcal{X}$ platí

i) $d(x, y) \geq 0$, a $d(x, y) = 0$ právě tehdy když $x = y$ - pozitivní definitnost,

ii) $d(x, y) = d(y, x)$ - symetrie,

iii) $d(x, y) \leq d(x, z) + d(z, y)$ - trojúhelníková nerovnost.

Dvojice $(\mathcal{X}, d)$ se potom nazývá **metrický prostor**.

Poznámky k vlastnostem vzdálenosti:

i) Vzdálenost různých bodů je vždy kladná. Nulová je pouze pro stejné body.

ii) Vzdálenost bodu x od bodu y je stejná jako vzdálenost y od x.

iii) Přímá vzdálenost mezi dvěma body je vždy menší nebo rovna vzdálenosti přes nějaký třetí bod.

Vzdálenost - příklady

Často používané vzdálenosti na $\mathbb{R}^p$:

- **Eukleidovská vzdálenost** nebo také L_2 vzdálenost,

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}.$$

- **Manhattanská vzdálenost** nebo také L_1 vzdálenost,

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{p} |x_i - y_i|.$$

- **Čebyševova vzdálenost** nebo také L_∞ vzdálenost,

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_i |x_i - y_i|.$$

Příkladem vzdálenosti na množině řetězců je pak **Levenštejnova vzdálenost** (angl. **Levenshtein distance**) definovaná jako minimální počet jednoznakových operací (vkládání, mazání, nahrazení), které převedou jeden řetězec na druhý.

Vstupy a výstupy shlukování

Než se zaměříme na dva konkrétní případy shlukovacích algoritmů, ujasněme si, co je vlastně vstupem a co výstupem shlukování.

Vstupy

- Metrický prostor $\mathcal{X}$ se vzdáleností d.
- Množina dat $\mathcal{D} \subset \mathcal{X}$.
- Obvykle také požadovaný počet shluků k.

Výstupy

- Rozklad množiny dat na jednotlivé shluky. To jest $C = (C_1, \ldots, C_k)$, kde $C_i \subset \mathcal{D}$ pro každé i a $C_i \cap C_j = \emptyset$ pro každé $i \neq j$, přičemž

$$\mathcal{D} = \bigcup_{i=1}^{k} C_i.$$

- Bod $x \in \mathcal{D}$ je tedy v i-tém shluku, jestliže $x \in C_i$.
- U hierarchického shlukování může být finálním výstupem dendrogram jakožto grafické znázornění hierarchické struktury shlukování, které podrobně představíme za chvíli.

Hierarchické shlukování

Začněme zcela přirozeným hladovým aglomerativním přístupem. Označme N počet prvků množiny dat $\mathcal{D}$.

- Na začátku uvažujme každý bod jako jeden shluk. Tj. máme právě N shluků.
- Nyní budeme opakovat následující kroky:
  - Najdeme dva shluky, které jsou k sobě nejblíže.
  - Tyto dva shluky spojíme do nového shluku.

Po $N - 1$ opakováních tedy skončíme s jediný velkým shlukem, ve kterém jsou všechny body.

- K tomu, abychom mohli tento postup provést, je třeba stanovit způsob měření vzdálenosti dvou shluků.
- Má-li být navíc výstupem shlukování, musíme určit nějaké zastavovací kritérium.
- To bývá nejčastěji počet shluků k, případně limitní hodnota vzdálenosti shluků, nad kterou už nebudeme shluky spojovat.[2]

[2] Vzdálenost spojovaných shluků v každém kroku roste (resp. neklesá).

Měření vzdálenosti shluků

Vstupem do shlukování je vzdálenost dvojice bodů $d(x, y)$. K měření vzdálenosti dvojic shluků $D(A, B)$ se obvykle používá jeden z následujících způsobů:

Metoda nejbližšího souseda (angl. single linkage) - generuje dlouhé řetězce

$$D(A, B) = \min_{x \in A, y \in B} d(x, y).$$

Metoda nejvzdálenějšího souseda (angl. complete linkage) - generuje kompaktní shluky

$$D(A, B) = \max_{x \in A, y \in B} d(x, y).$$

Párová vzdálenost (angl. average linkage) - kompromis předchozích dvou

$$D(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y).$$

Wardova metoda - v $\mathbb{R}^p$ - velmi účinná, minimalizuje nárust vnitřního rozptylu

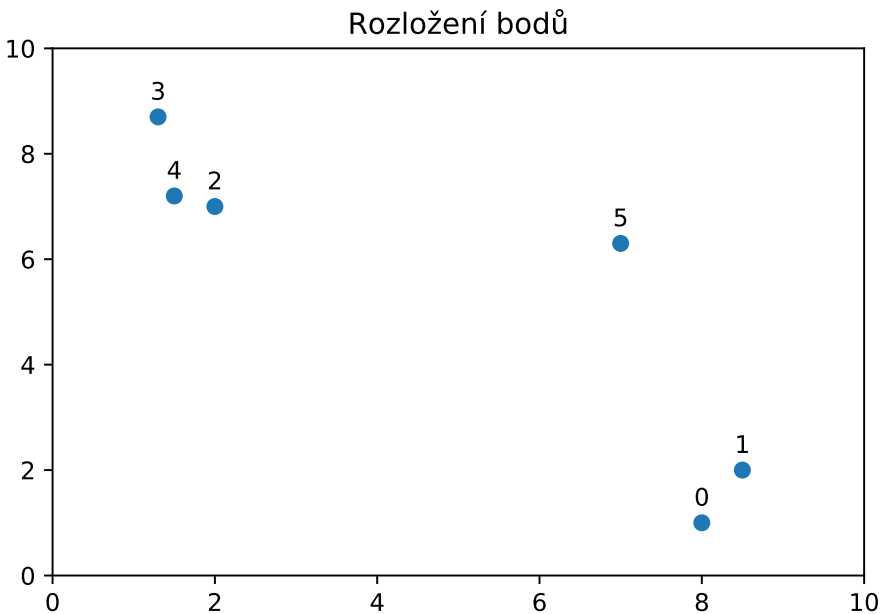
$$D(A, B) = \sum_{\boldsymbol{x} \in A \cup B} \|\boldsymbol{x} - \bar{\boldsymbol{x}}_{A \cup B}\|^2 - \sum_{\boldsymbol{x} \in A} \|\boldsymbol{x} - \bar{x}_A\|^2 - \sum_{\boldsymbol{x} \in B} \|\boldsymbol{x} - \bar{x}_B\|^2,$$

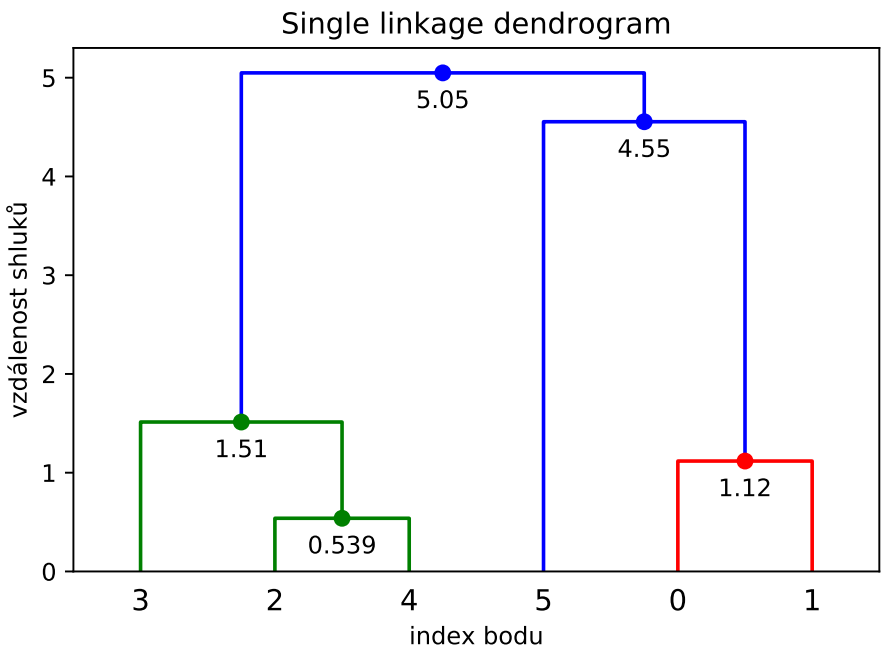
kde $\bar{\boldsymbol{x}}_A = \frac{1}{|A|} \sum_{\boldsymbol{x} \in A} \boldsymbol{x}$ je geometrický střed množiny A a $\bar{\boldsymbol{x}}_B$, $\bar{\boldsymbol{x}}_{A \cup B}$ analogicky.

Vizualizace hierarchického shlukování pomocí dendrogramu

Celý proces aglomerativního shlukování můžeme reprezentovat a graficky znázornit pomocí **dendrogramu**.

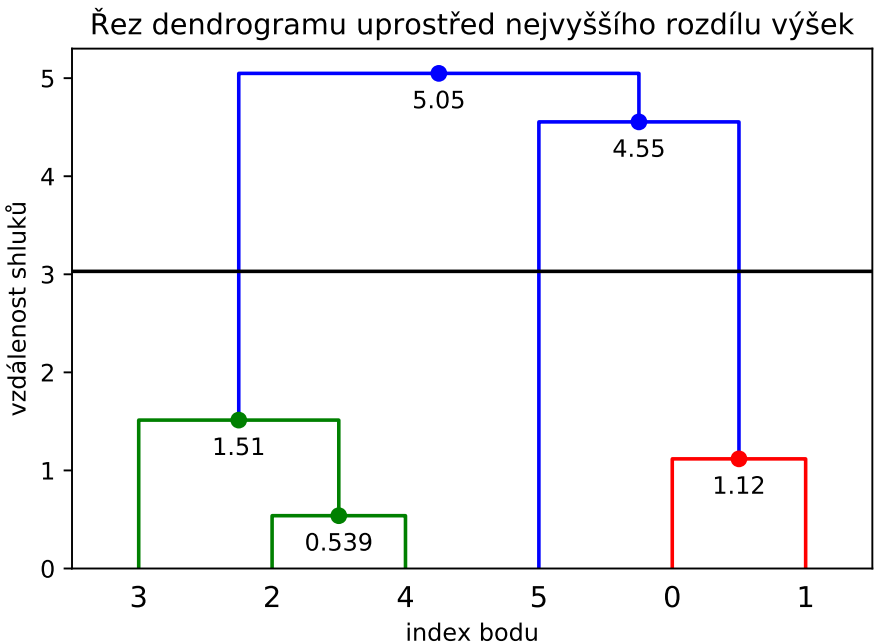
- Jedná se o strom, jehož vrcholy představují shluky, které při běhu vznikly.
- V listech jsou počáteční jednoprvkové shluky a kořen reprezentuje finální shluk všech bodů.
- Strom je navíc nakreslený tak, že jsou všechny listy ve výšce 0 a výška ostatních vrcholů odpovídá vzdálenosti podřazených shluků, které se v tomto vrcholu spojily.

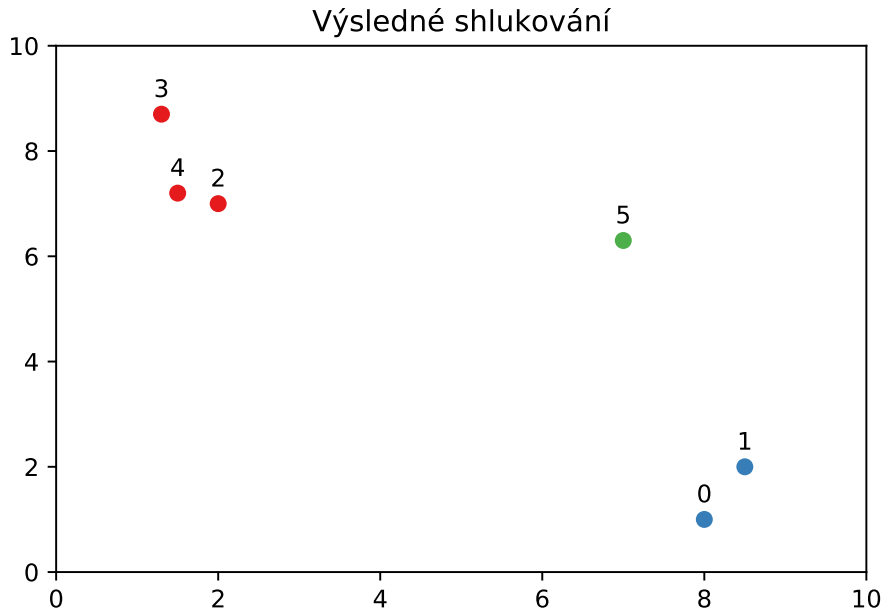




Jak získáme shlukování z dendrogramu

- Známe-li požadovaný počet k shluků, „rozřízneme“ dendrogram mezi k-tým a $(k-1)$-tým nejvyšším vrcholem. Hrany procházející říznutím pak odpovídají výsledným shlukům.
- Nebo můžeme stanovit limitní hranici vzdálenosti shluků a rozříznout dendrogram v dané výšce.
- Případně můžeme určovat místo rozříznutí a tím potažmo i počet shluků pomocí rozdílů výšek sousedních vrcholů.





Finální poznámky k hierarchickému shlukování

- Ukázali jsme si aglomerativní hierarchické shlukování.
- Existuje také divizní hierarchické shlukování při kterém naopak vycházíme z jednoho shluku a opakovaně provádíme dělení.
- Výhodou hierarchického shlukování je možnost získat ucelený pohled pomocí dendrogramu a teprve na jeho základě vybírat vhodný počet shluků.
- Další výhodou je hierarchická struktura. Při zvýšení počtu shluků o jedničku dojde pouze k rozdělení některého ze stávajících, přičemž ostatní se nemění.
- Kvalita vzniklého dendrogramu z pohledu zachování párových vzdáleností originálních dat může například být měřena pomocí cophenetic correlation.
- Hierarchické shlukování se příliš nehodí pro velké datové soubory, protože je výpočetně náročné.
  - V jednom kroku aglomerativního algoritmu musíme provést $m(m-1)/2$ porovnání, kde m je aktuální počet shluků. Složitost je tedy $\mathcal{O}(N^3)$. V případě single nebo complete linkage je možné se dostat na $\mathcal{O}(N^2)$.
  - V jednom kroku divizního algoritmu musíme provést 2^{m-1} porovnání, kde m je velikost děleného shluku, což implikuje složitost $\mathcal{O}(2^N)$.

Formulace shlukování jako optimalizační úlohy

- Oblíbeným přístupem ke shlukování je formulace ve tvaru optimalizační úlohy.
- Při tomto přístupu je třeba definovat **účelovou funkci** (angl. **objective function**), která nějakým způsobem ohodnocuje daný rozklad množiny na jednotlivé shluky.
- Cílem je pak nalézt takový rozklad, který účelovou funkci minimalizuje.
- Nyní se budeme zabývat situací, kdy pro dané k hledáme takový rozklad $C = (C_1, \ldots, C_k)$ na prostoru $\mathcal{X} = \mathbb{R}^p$ vybaveném Eukleidovskou vzdáleností, který minimalizuje účelovou funkci

$$G(C) = \sum_{i=1}^{k} \frac{1}{2|C_i|} \sum_{\boldsymbol{x},\boldsymbol{y} \in C_i} \|\boldsymbol{x} - \boldsymbol{y}\|^2.$$

- V účelové funkci se pro každý shluk sečtou průměrné kvadráty vzdáleností všech bodů daného shluku od jeho ostatních bodů.
- Ukažme si, že tuto účelovou funkci můžeme elegantně vyjádřit pomocí součtů kvadrátů vzdáleností od geometrických středů (těžišť) příslušných shluků.

Souvislost účelové funkce s geometrickým středem

Tvrzení

Pro konečnou množinu bodů $A \subset \mathbb{R}^p$ platí

$$\frac{1}{2|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} \|\boldsymbol{x}-\boldsymbol{y}\|^2 = \sum_{\boldsymbol{x} \in A} \|\boldsymbol{x}-\bar{\boldsymbol{x}}\|^2 = \min_{\boldsymbol{\mu} \in \mathbb{R}^p} \sum_{\boldsymbol{x} \in A} \|\boldsymbol{x}-\boldsymbol{\mu}\|^2,$$

*kde $\bar{\boldsymbol{x}} = \frac{1}{|A|} \sum_{\boldsymbol{x} \in A} \boldsymbol{x}$ je **geometrický střed** (angl. **centroid**) množiny A.*

Důkaz.

Pro každé $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$ platí $\|\boldsymbol{a}-\boldsymbol{b}\|^2 = (\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{b}) = \|\boldsymbol{a}\|^2 - 2\boldsymbol{a}^T\boldsymbol{b} + \|\boldsymbol{b}\|^2$, protože $\boldsymbol{b}^T\boldsymbol{a} = \boldsymbol{a}^T\boldsymbol{b}$. Pro $\boldsymbol{a} = \boldsymbol{x}-\boldsymbol{\mu}$ a $\boldsymbol{b} = \boldsymbol{y}-\boldsymbol{\mu}$ z toho plyne

$$\frac{1}{2|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} \|\boldsymbol{x}-\boldsymbol{y}\|^2 = \frac{1}{2|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} \|\boldsymbol{x}-\boldsymbol{\mu}\|^2 + \frac{1}{2|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} \|\boldsymbol{y}-\boldsymbol{\mu}\|^2 - \frac{1}{|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} (\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{y}-\boldsymbol{\mu}).$$

Poslední člen upravíme:

$$\frac{1}{|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} (\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{y}-\boldsymbol{\mu}) = \frac{1}{|A|} \sum_{\boldsymbol{x} \in A} (\boldsymbol{x}-\boldsymbol{\mu})^T \sum_{\boldsymbol{y} \in A} (\boldsymbol{y}-\boldsymbol{\mu}) = \frac{1}{|A|} \Big\| \sum_{\boldsymbol{x} \in A} (\boldsymbol{x}-\boldsymbol{\mu}) \Big\|^2.$$

Poslední člen je tedy vždy nezáporný. Navíc je rovný nule právě, když $\boldsymbol{\mu} = \bar{\boldsymbol{x}}$, což plyne z definice $\bar{\boldsymbol{x}}$. Prohodíme-li ve druhém členu $\boldsymbol{x}$ a $\boldsymbol{y}$ dostaneme první člen a po vyščítání přes $\boldsymbol{y}$ dostaneme

$$\frac{1}{2|A|} \sum_{\boldsymbol{x},\boldsymbol{y} \in A} \|\boldsymbol{x}-\boldsymbol{y}\|^2 \leq \sum_{\boldsymbol{x} \in A} \|\boldsymbol{x}-\boldsymbol{\mu}\|^2, \quad \text{s rovností pouze pokud } \boldsymbol{\mu} = \bar{\boldsymbol{x}}. \qquad \square$$

Algoritmus k-means - úvodní představení

- Problém nalezení globálního minima uvedené účelové funkce je výpočetně obtížný, ve skutečnosti **NP-těžký** (viz [Aloise et al, (2009)]).
- Představíme si heuristický iterativní algoritmus nazývaný **algoritmus k-means**, který bude v každém kroku zmenšovat hodnotu účelové funkce a konvergovat tak k jejímu lokálnímu minimu.

Algoritmus k-means - slovní popis

Nejprve zvolíme k středových bodů.
Iterativně opakujeme:

i. Vytvoříme shluky odpovídající středovým bodům tak, že pro každý bod $\boldsymbol{x}$ najdeme k němu nejbližší středový bod a podle něj $\boldsymbol{x}$ zařadíme do shluku, který tomuto středovému bodu odpovídá.

ii. Spočítáme nové středové body jako geometrické středy těchto shluků.

Algoritmus k-means - důkaz lokální optimalizace

Na základě předchozího tvrzení můžeme účelovou funkci vyjádřit jako

$$G(C) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \bar{\boldsymbol{x}}_i\|^2,$$

kde $\bar{\boldsymbol{x}}_i$ je geometrický střed i-tého shluku.

Zafixujme nyní $\boldsymbol{\mu}_i = \bar{\boldsymbol{x}}_i$. Vytvořme nové shluky $\tilde{C} = \{\tilde{C}_1, \ldots, \tilde{C}_k\}$, tak že bod $\boldsymbol{x}$ přesuneme do takového shluku $\tilde{C}_i$, ve kterém je vzdálenost $\|\boldsymbol{x} - \boldsymbol{\mu}_i\|$ nejmenší.

Tím zcela jistě dojde ke zmenšení součtů kvadrátů vzdáleností,

$$\sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \tilde{C}_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \leq \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2.$$

Z druhé rovnosti předchozího tvrzení pak plyne, že

$$G(\tilde{C}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \tilde{C}_i} \|\boldsymbol{x} - \bar{\tilde{\boldsymbol{x}}}_i\|^2 \leq \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \tilde{C}_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2,$$

kde $\bar{\tilde{\boldsymbol{x}}}_i$ je geometrický střed shluku $\tilde{C}_i$.
Dohromady tedy máme $G(\tilde{C}) \leq G(C)$.Tento postup můžeme opakovat a postupně tak klesat k nějakému lokálnímu minimu účelové funkce.

Algoritmus k-means - formalizace

Algoritmus k-means

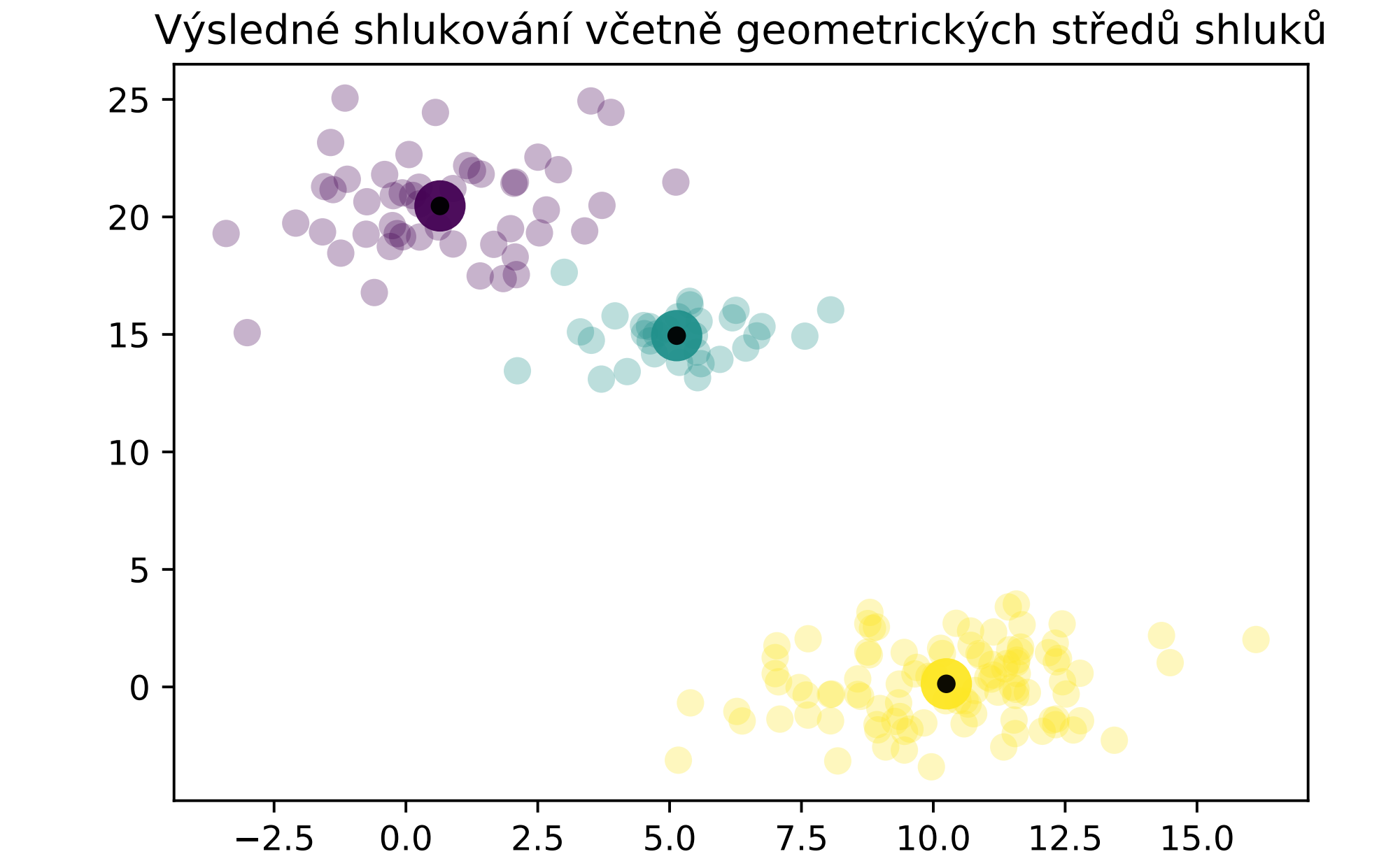
Inicializace: počáteční rozmístění k středových bodů $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$.
Iterativní část:

i. Roztřídíme body do shluků: $C_i = \left\{\boldsymbol{x} \in \mathcal{D} \mid i = \arg\min_j \|\boldsymbol{x} - \boldsymbol{\mu}_j\|\right\}$.

ii. Přepočítáme body $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ jako geometrické středy těchto shluků: $\boldsymbol{\mu}_i \leftarrow \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$.

- Z předhozích úvah plyne, že v každé iteraci má účelová funkce stejnou nebo nižší hodnotu.
- Běh algoritmu zastavíme, jakmile je změna hodnoty účelové funkce mezi jednotlivými iteracemi dostatečně malá.
- Výsledek algoritmu významně závisí na incializační části.
- Obvykle jsou počáteční středové body generovány náhodně (např. náhodným výběrem z dat), existují ale i „chytřejší metody“ jako např. k-means++ (viz [David, Vassilvitskii, (2007)]).
- Algoritmus následně opakovaně spouštěn. Jako finální pak bereme výsledek běhu s nejnižší hodnotou účelové funkce výsledku.

Algoritmus k-means - ukázka běhu

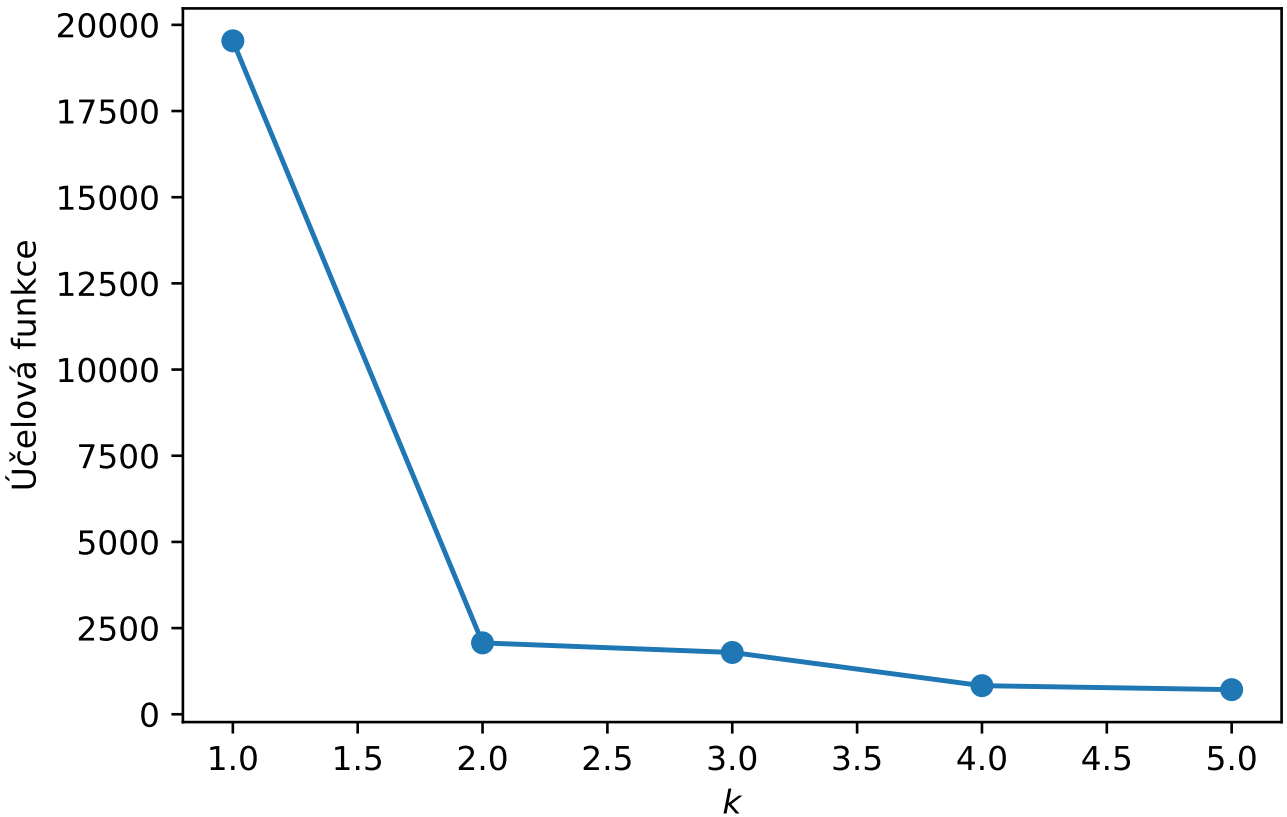


Algoritmus k-means - problematika volby k

- Na rozdíl od hierarchického shlukování, kde můžeme počet shluků určovat až na základě dendrogramu, je u algoritmu k-means nezbytné stanovit k dopředu.
- Bohužel neexistuje žádný univerzální způsob, jak počet shluků určit automaticky.
- Podívejme se tedy alespoň na jeden z používaných heuristických přístupů.
- Je-li k^* optimální počet shluků, lze očekávat, že pro $k < k^*$ bude účelová funkce s měnícím se k klesat hodně. Pro $k \geq k^*$ lze naopak očekávat zmenšení poklesu účelové funkce.
- Optimální k tedy můžeme detekovat jako hodnotu, pro kterou se mění pokles účelové funkce z hodně prudkého na méně prudký, hledat tzv. **loket** (angl. **elbow**). Je to ale hodně subjektivní a pro některá data v podstatě nepoužitelné.
- Existují i jiné metody, viz např. silhouette.

Algoritmus k-means - problematika volby k

Metoda lokte často nedává optimální výsledky, jak je pro předchozí data vidět na následujícím obrázku:



Zde bychom nejspíš určili jako optimální $k = 2$. Přitom ve skutečnosti byla data vygenerována jako směs tří různých dvourozměrných normálních rozdělení.

BI-ML1.21 přednáška 12

Daniel Vašata

FIT ČVUT

11. 12. 2024

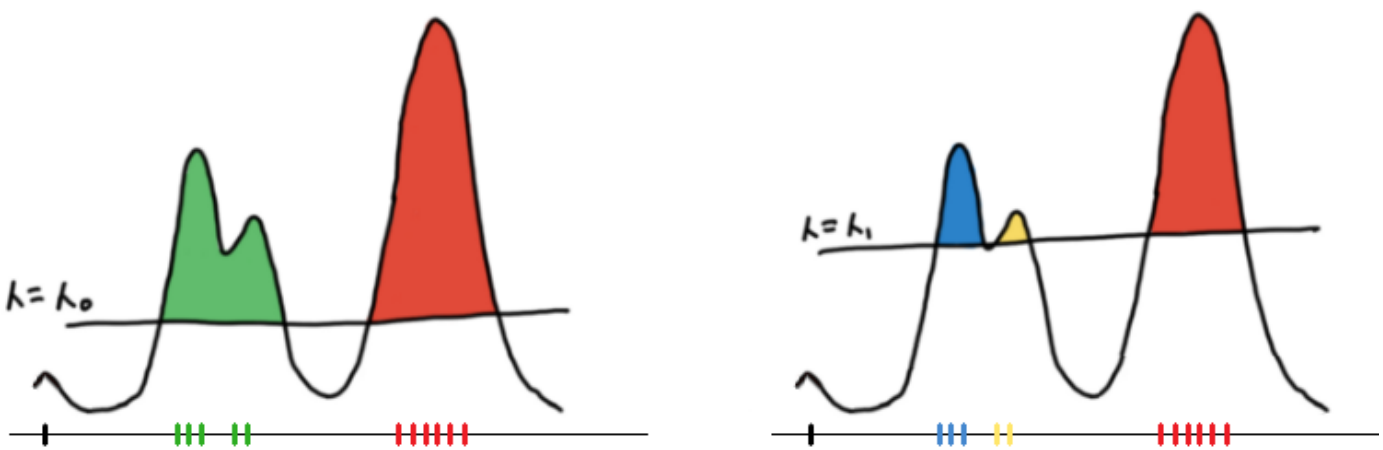
Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 18. prosince 2024 14:20.

Co bude v dnešní přednášce

- Shlukování pomocí hustoty
- DBSCAN
- Evaluace shlukování pomocí Silhouette skóre
- Asociační pravidla

Úvod do shlukování pomocí hustoty

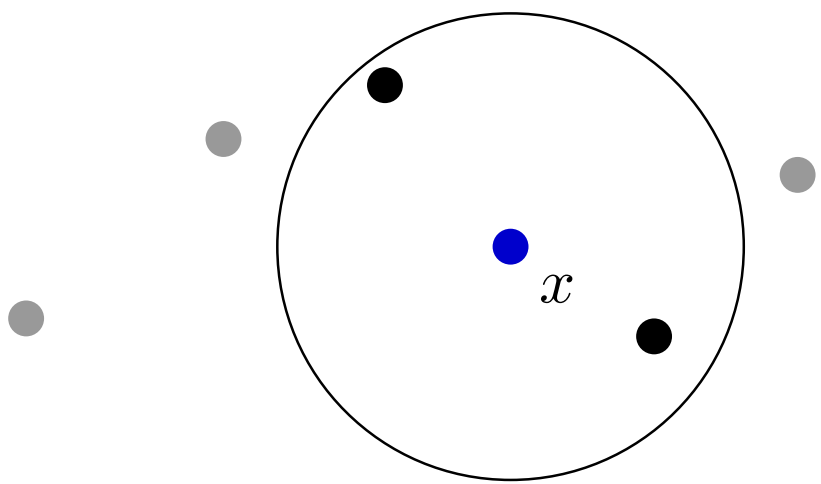
- V dnešní přednášce se zaměříme na shlukování, které využívá odhad hustoty rozdělení dat na prostoru $\mathcal{X}$ možných hodnot příznaků.
- Z pravděpodobnostního pohledu chápeme pozorovaná data jako realizace náhodného vektoru $\boldsymbol{X} = (X_1, \ldots, X_p)^T$.
- Pokud budeme mít nějakým způsobem odhadnutou hustotu pravděpodobnosti $f_{\boldsymbol{X}}(\boldsymbol{x}) \equiv f_{\boldsymbol{X}}(x_1, \ldots, x_p)$ (resp. něco, co jí je úměrné), můžeme ji využít k získání shlukování včetně identifikace bodů, které do žádných shluků nepatří.
- Shluky můžeme získat jako souvislé oblasti, ve kterých odhad hustoty překročí nějakou zvolenou hranici.
- Body mimo tyto oblasti pak označíme jako šum.



DBSCAN - úvodní pojmy

- Jedním s aktuálně nejpoužívanějších algoritmů shlukování, který je implicitně založen na principu odhadu hustoty, je algoritmus **DBSCAN**, což je zkratka angl. **density-based spatial clustering of applications with noise**, [Ester et al. (1996)].
- Připravme si nyní základní pojmy, na kterých DBSCAN stojí.
- Uvažujme metrický prostor $\mathcal{X}$ s metrikou $d(x, y)$ ze kterého pochází dataset $\mathcal{D}$ a parametry $\varepsilon > 0$ a MinPts $\in \mathbb{N}^+$.
- Definujme ε **okolí** bodu x v $\mathcal{D}$ (angl. ε-neighborhood) jako množinu

$$N_\varepsilon(x) = \{y \in \mathcal{D} | d(x, y) \leq \varepsilon\}.$$



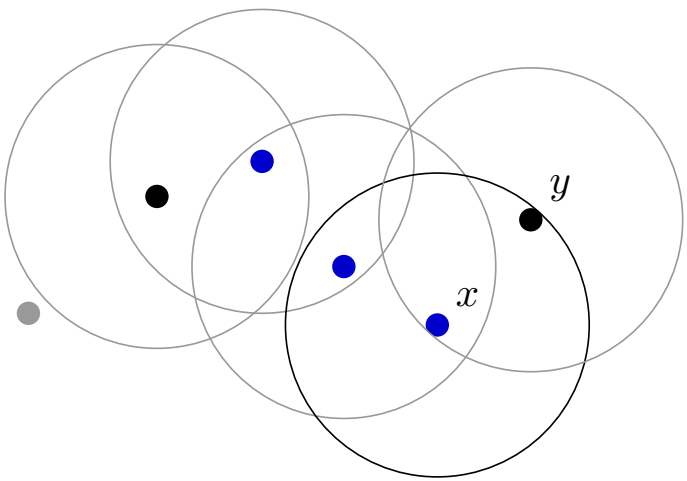
Znázornění ε okolí.

DBSCAN - klíčové body, přímá dosažitelnost

- Bod $x \in \mathcal{D}$ je **klíčový bod** (angl. **core point**), jestliže v jeho ε okolí v $\mathcal{D}$ je alespoň MinPts bodů,

$$|N_\varepsilon(x)| \geq \text{MinPts}.$$

- Bod $y \in \mathcal{D}$ je **přímo dosažitelný** (angl. **directly density-reachable**) z bodu $x \in \mathcal{D}$, jestliže x je klíčový bod a $y \in N_\varepsilon(x)$.
- Relace přímé dosažitelnosti je symetrická pro dvojici klíčových bodů, je ale nesymetrická pro tzv. **okrajový bod** (angl. **border point**), což je bod, který není klíčový, ale je přímo dosažitelný z klíčového bodu.

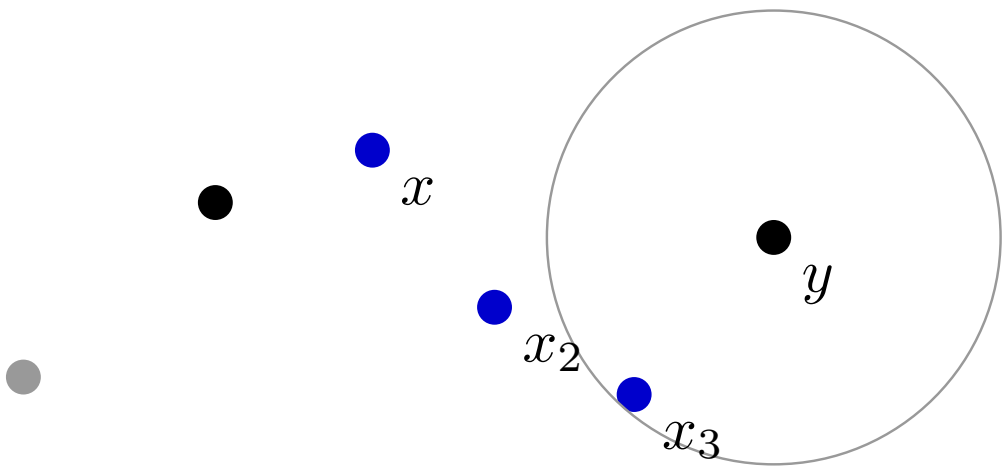


Pro MinPts $= 3$ je bod y přímo dosažitelný z x. Všechny klíčové body jsou modré, všechny okrajové body jsou černé.

DBSCAN - dosažitelnost

- Bod $y \in \mathcal{D}$ je **dosažitelný** (angl. **density-reachable**) z bodu $x \in \mathcal{D}$, pokud existuje posloupnost $x_1, x_2, \ldots, x_n \in \mathcal{D}$ bodů tak, že $x_1 = x$, $x_n = y$ a pro každé $i = 1, \ldots, n-1$ je x_{i+1} přímo dosažitelný z bodu x_i.

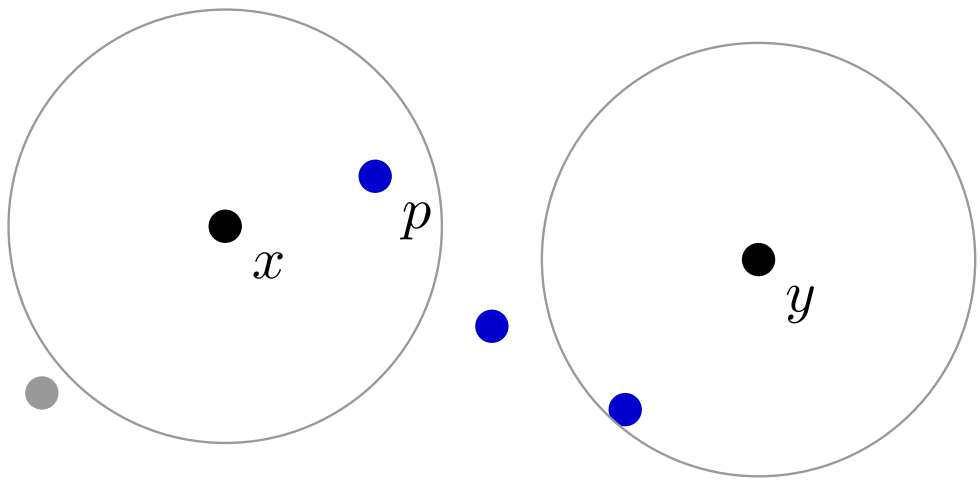
- Z toho plyne, že všechny body po cestě kromě posledního musí být klíčové body.



Pro MinPts $= 3$ je bod y dosažitelný z bodu x.

DBSCAN - spojenost

- Bod $y \in \mathcal{D}$ je **spojený** (angl. **density-connected**) s bodem $x \in \mathcal{D}$, jestliže existuje (klíčový bod) $p \in \mathcal{D}$, tak, že x i y jsou dosažitelné z bodu p.
- Relace spojenosti je zjevně symetrická. Pro klíčové body je také tranzitivní.
- Jestliže jsou dva body spojené a jeden z nich je klíčový bod, tak ten druhý je z toho prvního dosažitelný.



Pro MinPts $= 3$ je bod y spojený s bodem x.

DBSCAN - shluky, šum

Shluk nyní definujeme jako maximální množinu spojených bodů.

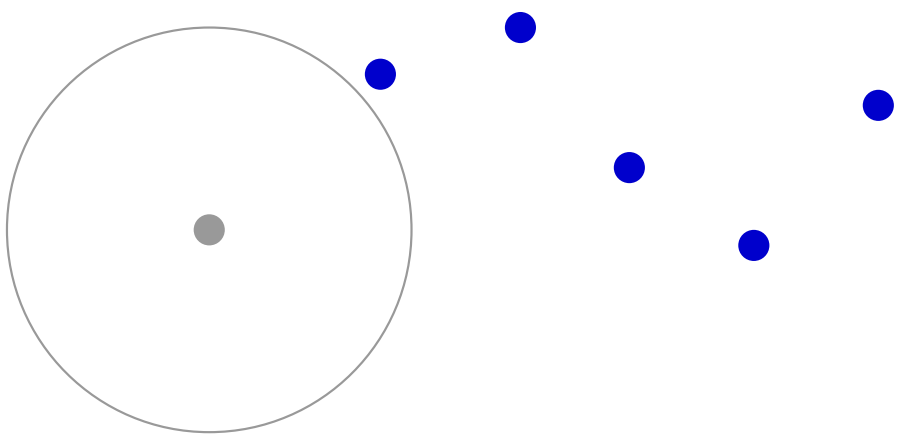
Definice

Shluk (angl. **cluster**) C je podmnožina $\mathcal{D}$ obsahující alespoň jeden klíčový bod tak, že:

- Pro každé $x, y \in \mathcal{D}$ platí, že když $x \in C$ a y je dosažitelný z x, pak $y \in C$ (maximalita).
- Pro každé $x, y \in C$ je x spojený s y (souvislost).

Označme $C_1, \ldots, C_k$ množinu všech shluků v $\mathcal{D}$ (vzhledem k ε a k). Množinu N bodů z $\mathcal{D}$, které nejsou v žádném ze shluků nazýváme **šumem** (angl. **noise**),

$$N = \mathcal{D} \setminus \bigcup_{i=1}^{k} C_i$$



Body ve shluku jsou modré, bod šumu je šedivý.

Poznámky k definici shluků

- Shluk C vždy obsahuje nějaký klíčový bod $p \in C$, ze kterého jsou dosažitelné všechny body v jeho ε okolí, kterých je alespoň MinPts.

  Právě jsme tedy ukázali, že každý shluk obsahuje alespoň MinPts bodů.

- Každý bod ve shluku C je spojený se všemi klíčovými body v C a tedy je z libovolného klíčového bodu dosažitelný. Shluk je tedy tvořen všemi body, které jsou dosažitelné z libovolného klíčového bodu v C.

- To nám dává návod, jak může algoritmus tvorby shluků fungovat. Najdeme klíčový bod a vytvoříme k němu shluk jako množinu všech z něho dosažitelných bodů (které ještě nejsou v jiném shluku).

Abstraktní popis algoritmu

Nyní si abstraktním způsobem popišme běh algoritmu DBSCAN.

Algoritmus DBSCAN

Nalezení klíčových bodů: Spočítáme ε okolí každého bodu a identifikujeme klíčové body.

Vytvoření zárodků shluků: Spojíme sousední (přímo dosažitelné) klíčové body do shluků.

Pro každý bod, který není klíčový:

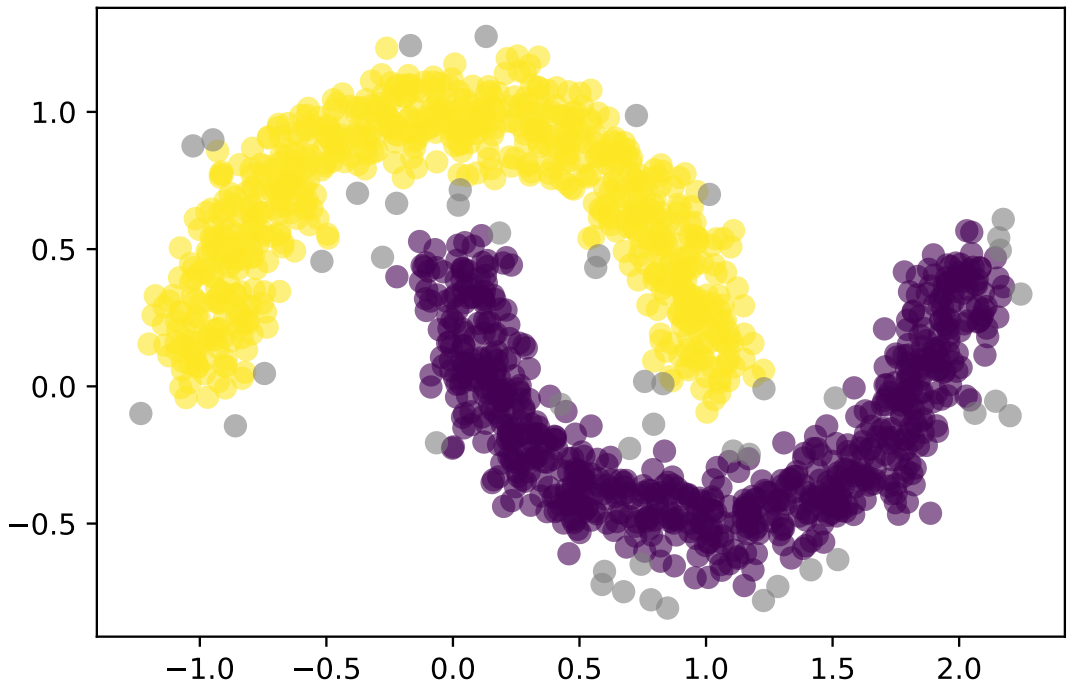
- Přidáme do shluku podle klíčového bodu v jeho okolí.
- Pokud takový neexistuje, přidáme mezi šum.

- Okrajový bod, který má ve svém ε okolí klíčové body z různých (zárodků) shluků spadne do prvního z těchto shluků, ke kterému se algoritmus dostane.
- Finálním výsledkem v takovém případě budou shluky, které nesplňují podmínku maximality v předchozí definici, protože okrajový bod bude pouze v jednom shluku a nikoliv ve všech, kde by dle maximality měl být.
- Lze ukázat ([Schubert et al. (2017)]), že složitost v nejhorším případě je $O(n^2)$. V mnoha reálných situacích se ale lze dostat na $O(n \log n)$.

DBSCAN - volba parametrů

- Zaměřme se nyní krátce na volbu parametrů algoritmu DBSCAN.
- Ukazuje se, že parametr MinPts je mnohem méně důležitý než parametr ε. Obvykle dobrou volbou jsou hodnoty okolo $4 - 6$ (někdy bývá doporučováno $2 \dots p$, kde p je počet příznaků).
- Pro parametr ε bývá doporučováno volit co nejmenší hodnotu s tím, že lze například brát průměrnou vzdálenost bodů k jejich nejbližšímu $(2 \cdot p - 1)$tému sousedovi.
- Také můžeme sledovat velikost šumu. Uvádí se, že obvykle by poměr šumu měl být mezi 1% a 30%.
- Dále je dobré sledovat velikost největšího shluku. Pokud jeho velikost překračuje 50% velikosti datasetu, bývá vhodné zmenšit ε, případně použít nějaký pokročilejší algoritmus (např. HDBSCAN).
- Více detailů k volbě parametrů najdete v [Schubert et al. (2017)] nebo v [Sander et al. (1998)].

DBSCAN - ukázka a poznámky



Znázornění shluků a šumu.

- Mezi hlavní výhody DBSCAN (a obecně metod založených na hustotě) patří možnost nalezení nekonvexních shluků.
- Další výhodou je snížená citlivost na odlehlé hodnoty - které jsou označeny jako šum.

Evaluace pomocí Silhouette skóre (1/3)

- Vraťme se nyní k obecné úloze shlukování a ukažme si jednoduchou a používanou metodu pro evaluaci shlukování pomocí metody **Silhouette** [Rousseeuw, P. (1987)], kterou lze využít i k určení optimálního počtu shluků.
- Uvažujme shlukování $\mathcal{D} = C_1 \cup \ldots C_k$ na metrickém prostoru $\mathcal{X}$ s metrikou $d(x, y)$ a pro libovolný bod $x \in \mathcal{D}$ označme $j(x)$ index shluku do kterého x patří, tj. $x \in C_{j(x)}$.
- Pro bod $x \in \mathcal{D}$ nyní:
  - Spočteme $a(x)$ jako průměrnou vzdálenost bodu x od všech ostatních bodů ve stejném shluku (vnitřní rozdílnost, angl. within dissimilarity)

$$a(x) = \frac{1}{|C_{j(x)}| - 1} \sum_{y \in C_{j(x)}, y \neq x} d(x, y).$$

  - Pro každý další shluk C_i, $i \neq j(x)$ spočteme průměrnou vzdálenost bodu x od všech bodů v tomto shluku

$$d(x, C_i) = \frac{1}{|C_i|} \sum_{y \in C_i} d(x, y)$$

  - Spočteme $b(x)$ jako minimum z těchto průměrných vzdáleností od ostatních shluků (sousední rozdílnost, angl. between dissimilarity)

$$b(x) = \min_{i \neq j(x)} d(x, C_i)$$

Evaluace pomocí Silhouette skóre (2/3)

Finální silhouette skóre bodu $x \in \mathcal{D}$ získáme vztahem

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}.$$

- Jestliže máme pouze jeden shluk, položíme $s(x) = 0$.
- Z definice vidíme, že vždy platí

$$-1 \leq s(x) \leq 1.$$

- Hodnota $s(x)$ blízká 1 znamená, že vnitřní rozdílnost $a(x)$ je mnohem menší než sousední rozdílnost $b(x)$, což značí, že ten bod je dobře zatříděn.
- Hodnota $s(x)$ okolo 0 znamená, že $a(x)$ je podobně velké jako $b(x)$ a tedy bod x je někde na okraji svého shluku a sousední shluk je blízko. Čili by ten bod klidně mohl patřit i do toho druhého shluku.
- Hodnota $s(x)$ blízká -1 znamená, že vnitřní rozdílnost $a(x)$ je mnohem větší než sousední rozdílnost $b(x)$ a tedy bod x by měl spíše patřit do toho sousedního shluku než do toho svého. Je tedy špatně přiřazen.

Evaluace pomocí Silhouette skóre (3/3)

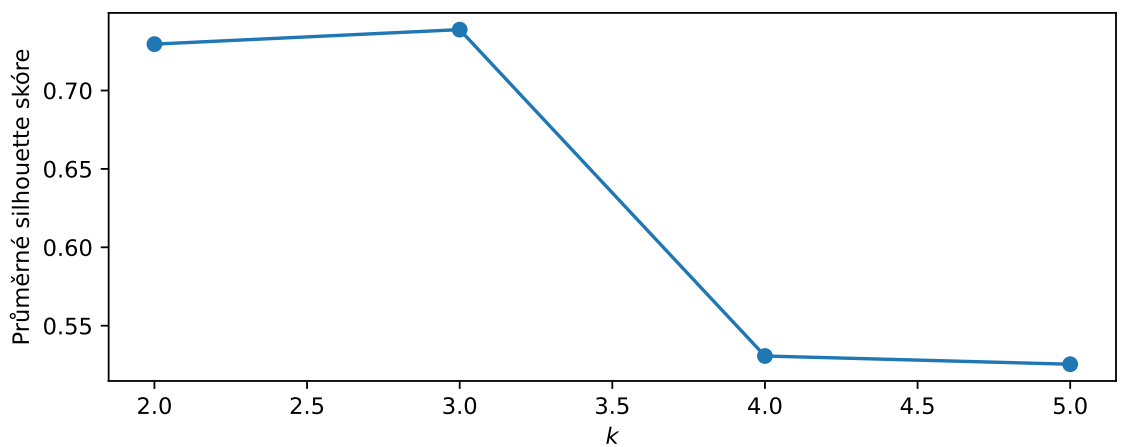
- Z ohodnocení $s(x)$ každého z bodů x můžeme získat jednak průměrné skóre s_i pro každý shluk C_i zvlášť

$$s_i = \frac{1}{|C_i|} \sum_{x \in C_i} s(x),$$

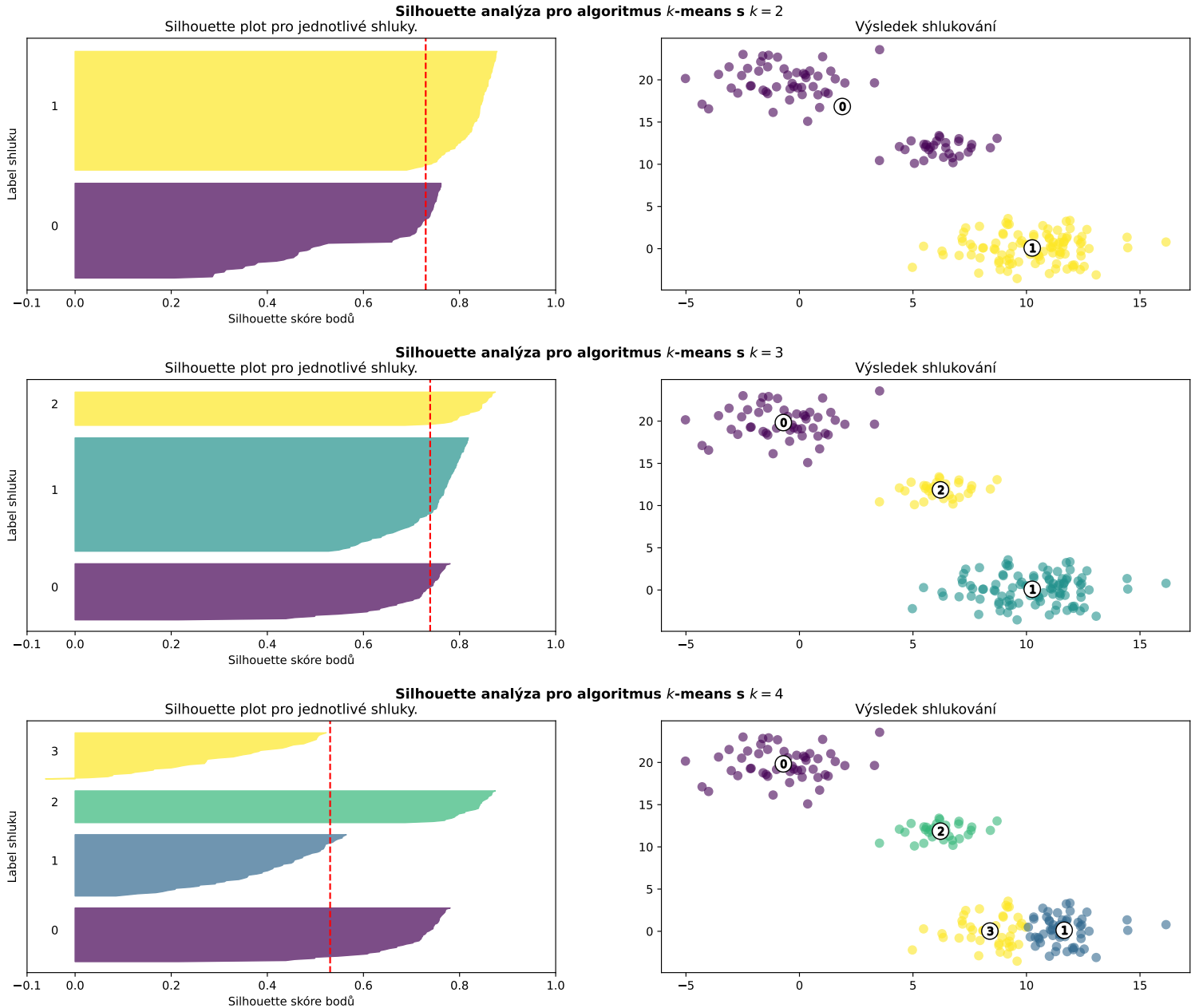
- a pak také průměrné skóre s pro celé shlukování

$$s = \frac{1}{|D|} \sum_{x \in \mathcal{D}} s(x).$$

- Čím vyšší dostaneme hodnotu, tím jsou body ve shlucích správněji umístěné a tedy je celé shlukování lepší.
- Porovnání skóre pro různé počty shluků můžeme použít k nalezení vhodného počtu shluků jako hodnoty, při které je skóre s maximální.



Znázornění silhouette skóre



Analýza asociačních pravidel

- **Analýza asociačních pravidel** je jedna ze standardních a oblíbených metod pro dobývání znalostí z komerčních databází.
- Obecným cílem je nalézt společné hodnoty příznaků $\boldsymbol{X} = (X_1, \ldots, X_p)^T$, které se v databázi nejčastěji vyskytují.
- V zásadě jde přesně o jeden z hlavních cílů nesupervizovaného učení, kdy chceme nalézt oblasti prostoru, kde se data vyskytují s velkou pravděpodobností.
- V nejčastěji používaném zjednodušení se zabýváme pouze oblastmi, které jsou ve tvaru kartézského součinu pro jednotlivé příznaky, tj. chceme, aby pravděpodobnost

$$\mathrm{P}\left(\cap_{j=1}^{p}(X_j \in s_j)\right),$$

kde s_j je podmnožina hodnot příznaku X_j, byla relativně velká.

- Průnik $\cap_{j=1}^{p}(X_j \in s_j)$ se v takovém případě nazývá **konjunktivní pravidlo** (angl. **conjunctive rule**).

Analýza nákupního košíku

- Analýza asociačních pravidel je nejčastěji aplikována v případě binárních příznaků, $X_j \in \{0, 1\}$, kdy se pak nazývá **analýza nákupního košíku** (angl. **market basket analysis**).
- Pro oblast, kterou hledáme ve tvaru kartézského součinu se v tomto případě používá ještě další omezení, a to, že s_j je buď jednoprvková množina $\{1\}$ nebo všechny možnosti daného příznaku X_j (v tu chvíli příznak vypadne).
- Ekvivalentně tedy hledáme množinu indexů $\mathcal{K} \subset \{1, \ldots, p\}$ tak, že

$$\mathrm{P}\left(\cap_{j\in\mathcal{K}}(X_j = 1)\right) = \mathrm{P}\left(\prod_{j\in\mathcal{K}} X_j = 1\right)$$

  je relativně velká.
- Množina $\mathcal{K}$ se pak nazývá **množina položek** (angl. **item set**).
- Relativní velikost položek v datasetu, které danou množinu položek obsahují se značí $T(\mathcal{K})$ a nazývá **podpora** (angl. **support**) množiny položek $\mathcal{K}$ a odpovídá odhadu výše uvedené pravděpodobnosti:

$$T(\mathcal{K}) = \hat{\mathrm{P}}\left(\cap_{j\in\mathcal{K}}(X_j = 1)\right) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j\in\mathcal{K}} x_{i;j}.$$

Asociační pravidla (1/2)

- Při provádění analýzy nákupního košíku hledáme všechny množiny položek, pro které je podpora větší než nějaká zvolená mez t:

$$\{\mathcal{K}_\ell | T(\mathcal{K}_\ell) > t\}.$$

- K nalezení řešení se používá efektivní algoritmus (případně jeho novější vylepšení), který se nazývá Apriori algoritmus ([Agrawal, Srikant (1994)]).
- Pro každou množinu položek $\mathcal{K}$, kterou takto získáme, dále hledáme vhodné rozložení na dvě disjunktní podmnožiny A a B, $A \cup B = \mathcal{K}$, které budeme nazývat **asociační pravidlo** (angl. **association rule**) a značit

$$A \Rightarrow B.$$

- První položka A asociačního pravidla se nazývá **předpoklad** (angl. **antecedent**) a druhá položka B se nazývá **závěr** (angl. **consequent**).
- **Podpora** $T(A \Rightarrow B)$ pravidla $A \Rightarrow B$ je definována jako podpora sjednocení $\mathcal{K} = A \cup B$.
- **Spolehlivost** (angl. **confidence**) $C(A \Rightarrow B)$ pravidla $A \Rightarrow B$ je definována jako podpora pravidla podělená podporou předpokladu A:

$$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)},$$

což odpovídá odhadu podmíněné pravděpodobnosti $\mathrm{P}(B|A)$.

Asociační pravidla (2/2)

- Asociační pravidla jsou volena tak, aby spolehlivost byla větší než nějaká zvolená mez c.
- Finálním výstupem asociační analýzy pravidel je množina asociačních pravidel, které splňují
$$T(A \Rightarrow B) > t \quad \text{a} \quad C(A \Rightarrow B) > c.$$
- U nalezených asociačních pravidel se dále často měří **zdvih** (angl. **lift**) definovaný jako
$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)},$$
který odpovídá odhadu podílu $\mathrm{P}(B|A)/\mathrm{P}(B)$ což znamená, kolikanásobně je větší $P(B|A)$ oproti $P(B)$.
- Někdy se také měří **pokrytí** (angl. **coverage**) definované jako
$$\text{Coverage}(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(B)},$$
což odpovídá odhadu podmíněné pravděpodobnosti $\mathrm{P}(A|B)$.
- Pokrytí tedy indikuje, jak často lze závěr vyložit coby důsledek předpokladu.

Asociační pravidla - příklady

Klasickým příkladem asociačního pravidla je

$$\{\texttt{párky}\} \Rightarrow \{\texttt{hořčice, chléb}\}.$$

- Podpora 0.06 znamená, že v celém datasetu se trojice položek $\{\texttt{párky, hořčice, chléb}\}$ vyskytuje v 6% případů.
- Pokud se `párky` vyskytují v datasetu v 8% případů, bude spolehlivost $0.06/0.08 = 0.75$, což znamená, že když si zákazník koupil `párky`, v 75% případů si koupil také `horčici` a `chléb`.
- Pokud je dvojice `horčice` a `chléb` v datasetu v 15% případů, zdvih bude $0.75/0.15 = 5$.
- Pokrytí v takovém případě bude $0.06/0.15 = 0.4$. Čili ve 40% případů lze `horčici` a `chléb` uvažovat jako důsledek koupě `párků`.

Nevýhoda omezení na podporu je, že pravidla s velkou hodnotou spolehlivosti i zdvihu ale s nízkou podporou, jako např.

$$\{\texttt{doutník}\} \Rightarrow \{\texttt{rum}\},$$

nebudou nalezeny.