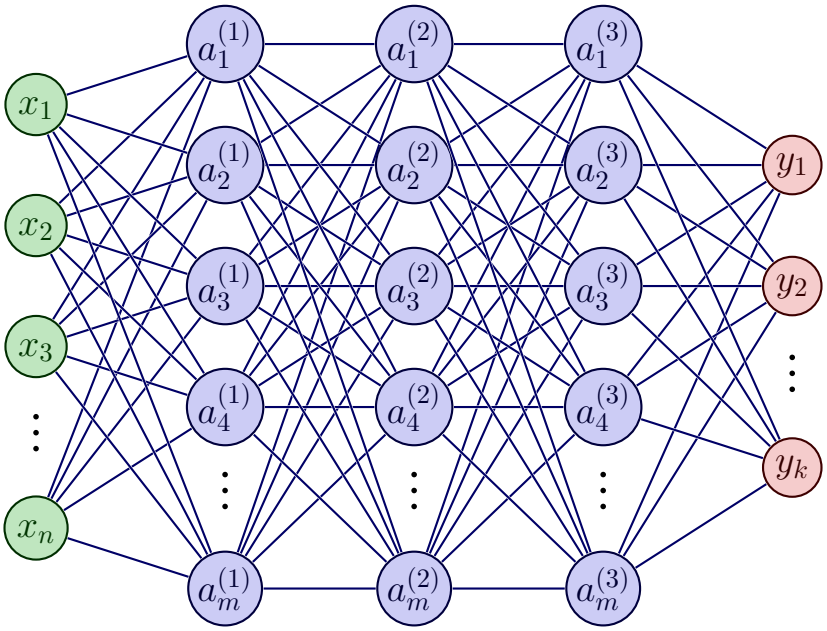
Strojové učení

Kurzy BI-ML1 / BI-ML2

Tommy Chu
chutommy@fit.cvut.cz

Fakulta informačních technologií,
České vysoké učení technické v Praze

3. února 2024



Obsah

0 Úvod

Ahoj,

tento text jsem sepsal v rámci přípravy na zkoušky BI-ML1 a BI-ML2. Jedná se o přehled klíčových konceptů, které mi osobně přišly důležité nebo zajímavé. Pro hlubší porozumění tématu doporučuji následující literaturu:

- Murphy K. P.: Probabilistic Machine Learning: An Introduction.
- Bishop Ch. M.: Pattern Recognition and Machine Learning.
- Hastie T., Tibshirani R., Friedman J.: The Elements of Statistical Learning.

Přeji příjemné studium!

TC

1 Rozhodovací stromy

Na vstup předpokládáme tabulku s N záznamy a p příznaky $X_0, \ldots, X_{p-1}$. Cílem je vytvořit rozhodovací strom který přiřadí co nejvíce vstupům co nejpřesnější hodnoty Y. Exhaustive search takového optimálního stromu je NP-úplný problém, existují však algoritmy nabízející kompromis. Pro klasifikaci ID3 a pro regresi CART nabízejí suboptimální ale dosažitelné řešení.

1.1 Algoritmus ID3

Algoritmus ID3 je hladový algoritmus pro konstrukci rozhodovacího stromu. V každém vrcholu hledá podle zadaného kritéria nejlepší test (rozhodovací pravidlo) nepoužitých příznaků, které nejlépe rozdělí data na dvě podmnožiny, které maximalizují vybrané kritérium.

Algoritmus začíná s celým datasetem a rekurzivně se zanořuje do synů, dokud nenastane zastavovací kritérium (typicky max hloubka, malý počet záznamů v listech, ...).

1.2 Kritéria pro větvení

1.2.1 Míra neuspořádanosti

Na binární množině kde p_0 a p_1 označují poměry 0 a 1, definujeme míru neuspořádanosti jako funkci p_0 splňující:

1. nezápornost na $[0, 1]$,
2. nulovost pro $p_0 = 1$ nebo $p_0 = 0$,
3. maximum nabývá v $p_0 = \frac{1}{2}$,
4. je rostoucí na $\left[0, \frac{1}{2}\right]$ a klesající na $\left[\frac{1}{2}, 1\right]$

1.2.2 Entropie

Definici míry neuspořádanosti například splňuje entropie.

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

Entropii lze definovat i pro nebinární hodnoty.

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

Entropie (pojem z teorie informace) používá dvojkový logaritmus a pracuje s jednotkou bit (Claude Shannon).

1.2.3 Gini index

Místo entropie lze použít gini index (gini impurity), které má podobné vlastnosti.

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

1.2.4 Informační zisk

Příznak pro rozdělení se volí podle informačního zisku

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

kde $\mathcal{D}_0, \mathcal{D}_1$ jsou příslušné podmnožiny $\mathcal{D}$ a t_0, t_1 poměry počtu 0 a 1 v Y.

1.3 Použití pro klasifikaci a regresi

1.3.1 Klasifikace

V případě klasifikace strom rozhoduje většinovým hlasováním v příslušném listu, do kterého záznam přísluší. Poměr výsledného prvku v listu určuje jistotu modelu při takové volbě.

1.3.2 Regrese

U regrese se rozhoduje podle průměru v příslušném listu.

Konstrukce stromu v regresní úloze probíhá podobně jako v klasifikační. Místo minimalizace míry neuspořádanosti, algoritmus dělí data tak, aby byly hodnoty v listech co nejblíže ke střední hodnotě.

Pro odhad odchylky od střední hodnoty se používá MSE (mean squared error), případně MAE (mean absolute error).

$$\mathrm{MSE}(\mathbf{Y}) = \frac{1}{N}\sum_{i=0}^{N-1}(Y_i - \bar{Y})^2 \qquad \mathrm{MAE}(\mathbf{Y}) = \frac{1}{N}\sum_{i=0}^{N-1}|Y_i - \bar{Y}|$$

Hladovému algoritmu konstrukce stromu, ve kterém se minimalizuje MSE, se říká CART (classification and regression trees).

Jako alternativu informačního zisku (ID3), používá CART

$$\mathrm{MSE}(\mathcal{D}) - t_0\,\mathrm{MSE}(\mathcal{D}_0) - t_1\,\mathrm{MSE}(\mathcal{D}_1)$$

1.4 Hyperparametry

Rozhodovací stromy mají často nízký bias a vysokou varianci. Jako hyperparametry rozhodovacího stromu můžeme volit různá ukončovací pravidla, kterými zamezit přeučení.

1. dělící kritérium
   (a) gini, entropy (klasifikace)
   (b) squared error, absolute error (u regrese)
2. max hloubka, od které algoritmus už dále nedělí
3. minimální počet dat v množině před, resp. po, dělení
4. minimální nutná hodnota informačního zisku

1.5 Shrnutí

Rozhodovací stromy jsou nenáročné na přípravu, jsou dobře interpretovatelné, jednoduché a rychlé. Jsou však nerobustní a je snadné je přeučit.

2 Metoda nejbližších sousedů kNN

Metoda nejbližších sousedů je model supervizovaného učení, které predikuje na základě nejbližších záznamů ve vícedimenzionálním prostoru trénovacích dat.

2.1 Hyperparametry

2.1.1 Počet sousedů k

Číslo k určuje počet nejbližších sousedů, ze kterých model počítá predikci. Vyšší hodnota zabraňuje přeučení.

2.1.2 Vzdálenost (metrika)

Metrika na množině $\mathcal{X}$ je funkce $d : \mathcal{X} \times \mathcal{X} \longrightarrow [0, +\infty)$ taková, že pro každé $x, y, z \in \mathcal{X}$ platí

1. pozitivní definitnost: $d(x, y) \geq 0 \wedge d(x, y) = 0$ právě tehdy když $x = y$
2. symetrie: $d(x, y) = d(y, x)$
3. trojúhelníková nerovnost: $d(x, y) + d(y, z) \geq d(x, z)$

Častými metrikami jsou Minkowského k-metriky (L_k vzdálenosti)

$$\|\boldsymbol{x} - \boldsymbol{y}\|_k = d_k(\boldsymbol{x}, \boldsymbol{y})_k = \sqrt[k]{\sum_{i=0}^{p-1} |x_i - y_i|^k}$$

Speciálně pro $k = 1$ dostáváme Manhattanskou vzdálenost

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{p-1} |x_i - y_i|$$

Pro $k = 2$ Eukleidovskou vzdálenost

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=0}^{p-1} (x_i - y_i)^2}$$

A pro $k = +\infty$ Chebyshevovu vzdálenost

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

Příklad dalších často používaných metrik je Levenshteinova editační vzdálenost dvou řetězců nebo kosinová vzdálenost dvou vektorů podle úhlu (pouze pozitivně semidefinitní), které svírají

$$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2}$$

Existují i sofistikovanější metriky jako je například Jaccardova pro množiny nebo Haversinova pro vzdálenost dvou bodů na sféře.

2.1.3 Váha sousedů

U regresních úloh může být váha sousedů uniformní (každý z k sousedů má stejný vliv na výsledek predikce)

$$\hat{y} = \frac{1}{k} \sum_{i=0}^{k-1} y_k$$

nebo může být vážený podle vzdálenosti (bližší soused je vlivnější)

$$\hat{y} = \frac{\sum_{i=0}^{k-1} w_i y_k}{\sum_{i=0}^{k-1} w_i} \quad \text{kde} \quad w_i = \frac{1}{d(\boldsymbol{x}_i, \boldsymbol{n}_i)}$$

2.2 Použití pro klasifikaci a regresi

Pro trénovací data $\boldsymbol{X} \in \mathbb{R}^{N,p}$ s vysvětlovanou proměnou $Y \in \mathbb{R}^N$ predikujeme hodnotu vysvětlované proměnné pro datový bod $\boldsymbol{x} \in \mathbb{R}^p$ hlasováním (klasifikace), případně průměrem (regrese), k nejbližších sousedů.

Model kNN má tak velmi levné “trénování", kdy pouze ukládá trénovací množinu do paměti. Predikce je ovšem výpočetně náročnější.

2.3 Normalizace dat

Kvůli rozdílným rozsahům příznaků mají rozsahy hodnot v původních datech nerovnoměrný příspěvek na vzdálenost. Tento problém se dá řešit normalizací do intervalu $[0, 1]$, případně $[-1, 1]$, nebo standardizací.

$$\text{normalizace: } x_i \leftarrow \frac{x_i - \min_x}{\max_x - \min_x} \qquad \text{standardizace: } x_i \leftarrow \frac{x_i - \bar{x}}{\sqrt{s_x^2}}$$

Normalizace není vždy přímočará. Někdy se vyplatí normalizovat jen některé příznaky, nebo do různě velkých rozsahů, čímž se přiřadí umělá váha.

2.4 Prokletí dimenzionality

Metodě kNN neprospívá vysoká dimenzionalita. S vyšší dimenzionalitou rostou vzdálenosti všech bodů a může se stát, že v případě řídkých dat je okolí, co do vzdálenosti, prázdné nebo nereprezentativní (rozdíl vzdálenosti vzdálených a blízkých bodů se zmenšuje).

Nominální data se předzpracovávají pomocí OHE, což kvůli těmto problémům není ideální. Problém představuje také přítomnost méně relevantní příznaků, protože se model chová ke všem rozměrům stejně.

Pro rostoucí počet dimenzí významně roste potřebný počet prvků pro zaplnění prostoru a reprezentativní predikce.

3 Lineární regrese

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

3.1 Model

Pro data s hodnotami příznaků $X_1, \ldots, X_p$ a hodnotou vysvětlované proměnné Y předpokládáme lineární model

$$Y = w_0 + w_1 X_1 + \ldots + w_p X_p + \varepsilon$$

kde w_i jsou neznámé koeficienty a ε představuje chybu nebo nekonzistenci výsledné hodnoty Y, kterou model nezachycuje a pro kterou platí $\mathrm{E}\,\varepsilon = 0$.

V bodě $(x_1, \ldots, x_p)^T$ tohoto modelu platí vztah

$$Y = w_0 + w_1 x_1 + \ldots + w_p x_p + \varepsilon = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon$$

kde zavádíme následující vektorovou notaci pro vstup $\boldsymbol{x}$ a vektor vah $\boldsymbol{w}$

$$\boldsymbol{x} = (1, x_1, \ldots, x_p)^T$$
$$\boldsymbol{w} = (w_0, w_1, \ldots, w_p)^T$$

3.2 Predikce

S odhadnutými váhami $\hat{\boldsymbol{w}}$ predikujeme vztahem

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

Pro skutečnou hodnoty $Y = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon$ platí z předpokladu $\mathrm{E}\,\varepsilon = 0$

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

$\hat{Y}$ je tedy bodovým odhadem EY v bodě $\boldsymbol{x}$.

3.3 Metoda nejmenších čtverců

Metoda nejmenších čtverců nalézá hodnotu $\hat{\boldsymbol{w}}$ odhadu $\boldsymbol{w}$ minimalizací následující kvadratické ztrátové funkce:

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

Pro trénovací množinou $(\boldsymbol{x}_i, Y_i), i = 1, \ldots, N$ minimalizujeme reziduální součet čtverců

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

3.3.1 Maticový zápis trénovací množiny

Zavádíme náhodné vektory $\boldsymbol{Y} = (Y_1, \ldots, Y_N)^T, \boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_N)$ a body $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ zapisujeme do řádků matice

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,p} \end{pmatrix} \in \mathbb{R}^{N,p+1}$$

Při tomto značení platí rovnice $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$ kde $\mathrm{E}\,\boldsymbol{\varepsilon} = \boldsymbol{0}$.

3.3.2 Minimalizace RSS

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

Začneme nalezením gradientu této funkce

$$\begin{aligned} \frac{\partial\, \mathrm{RSS}(\boldsymbol{w})}{\partial w_j} &= \sum_{i=1}^{N} 2(Y_i - \boldsymbol{w}^T \boldsymbol{x}_i)(-x_{i,j}) \\ \nabla\, \mathrm{RSS}(\boldsymbol{w}) &= \sum_{i=1}^{N} 2(Y_i - \boldsymbol{w}^T \boldsymbol{x}_i)(-\boldsymbol{x}_i) \\ &= -2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}) \end{aligned}$$

Položením $\nabla\, \mathrm{RSS}(\boldsymbol{w}) = \boldsymbol{0}$ obdržíme normální rovnici

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

Předpokládáme-li, že $\boldsymbol{X}^T\boldsymbol{X}$ je regulární, pak lze $\boldsymbol{w}$ odhadnout následovně

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

V případě regulární matice $\boldsymbol{X}^T\boldsymbol{X}$ je $\hat{\boldsymbol{w}}_{\mathrm{OLS}}$ jediným kritickým bodem. Z geometrické interpretace, kterou dosáhneme stejného výsledku, bude plynout, že se jedná o globální minimum $\mathrm{RSS}(\boldsymbol{w})$.

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

3.3.3 Geometrická interpretace

Minimalizace $\mathrm{RSS}(\boldsymbol{w}) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|^2$ je problém ekvivalentní minimalizaci $\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|$. Hledáme tedy $\boldsymbol{w} \in \mathbb{R}^{p+1}$ takové, že vektory $\boldsymbol{Y}$ a $\boldsymbol{X}\boldsymbol{w}$ jsou si v prostoru $\mathbb{R}^N$ co nejblíže.

$\boldsymbol{X}\boldsymbol{w}$ je lineární kombinací sloupců $\boldsymbol{X}$:

$$\boldsymbol{X}\boldsymbol{w} = \sum_{i=0}^{p} \boldsymbol{w}_i \boldsymbol{X}_{:,i} \in \langle \boldsymbol{X}_{:,0}, \boldsymbol{X}_{:,1}, \ldots, \boldsymbol{X}_{:,p} \rangle = \boldsymbol{P}$$

Hledaný bod $\boldsymbol{X}\boldsymbol{w}$ je proto nejblíže k bodu $\boldsymbol{Y}$, právě pokud je vektor $\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}$ ortogonální na podprostor $\boldsymbol{P}$:

$$\boldsymbol{X}_{:,i}^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}) = 0 \text{ pro všechna } i = 0, 1, \ldots, p$$

Což po přepsání do maticového tvaru dá opět normální rovnici

$$\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}) = \boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{0}$$

Z úvah také navíc plyne, že každé $\hat{\boldsymbol{w}}$, které splňuje normální rovnici, $\mathrm{RSS}(\boldsymbol{w})$ minimalizuje (jedná se o globální minimum).

3.4 Regularita versus lineární nezávislost sloupců matice X

Pro libovolné $\boldsymbol{s} \in \mathbb{R}^{p+1}$ platí následující posloupnost implikací:

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

Z toho vyplývá, že $\boldsymbol{X}^T\boldsymbol{X}$ je regulární právě tehdy, když $\boldsymbol{X}\boldsymbol{s} = \boldsymbol{0}$ pouze pro $\boldsymbol{s} = \boldsymbol{0}$, což platí právě tehdy, když jsou sloupce matice $\boldsymbol{X}$ lineárně nezávislé. To zřejmě nemusí vždy platit: např. když $N < p + 1$ nebo pokud je nějaký příznak lineární kombinací ostatních.

Normální rovnice má v případě regulární $\boldsymbol{X}^T\boldsymbol{X}$ právě jedno řešení.

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

V opačném případě má jich má nekonečně mnoho.

3.5 Problém kolinearity

Pokud je nějaký příznak "skoro" lineární kombinací ostatních příznaků, pak můžeme narazit na problém kolinearity. V případě silně korelovaného příznaku je příslušná proměnná do jisté míry zbytečná, protože ve svém rozměru (dimenzi) nepřidává žádnou informaci navíc a zároveň může mít ve výsledném vektoru $\hat{\boldsymbol{w}}_{\text{OLS}}$ vysoký koeficient. Jako příklad lze uvést 3-dimenzionální prostor (X_1, X_2, Y) ve kterém řešení degeneruje do skoro-přímky, kterou lze protnou rovinou více způsoby.

3.5.1 Důsledky kolinearity

Ve výsledku kolinearita způsobuje vysoký rozptyl $\hat{\boldsymbol{w}}_{\text{OLS}}$, který je tak velmi citlivý na data. Pro různé realizace stejného náhodného výběru se řešení normální rovnice může značně lišit. Vysoký rozptyl se následovně přenáší na predikce, které jsou pak méně spolehlivé.

3.5.2 Řešení problému kolinearity

Řešením by bylo odstranit příznaky, které kolinearity způsobují. To však není vždy jednoduché. S vyšším počtem příznaků je velmi obtížné takové sloupce identifikovat. Také se může stát, že kolinearita je mezi velkým počtem příznaků, které jsou dohromady klíčové pro správnou predikci. V takovém případě není ideální takové příznaky zahodit. Existují však metody, které dokážou počet příznaků snížit (odstraněním, případně nahrazením menším počtem) tak, aby byly sloupce LN.

Je také možné změnit funkci, kterou minimalizujeme, abychom měli stabilnější a jednoznačnější řešení. Typicky se přidává regulační člen, který problémy kolinearity může zmírnit (hřebenová regrese v sekci 4 nebo lasso v sekci 5.2).

3.6 Shrnutí

Lineární regrese je rezistentní vůči problémům spojené s vysokou dimenzí dat. Problém nastává, když je dat méně než příznaků nebo když jsou příznaky silně korelované.

4 Hřebenová regrese

Hřebenová nebo také L_2 regularizace, stejně jako lineární regrese, předpokládá lineární model $Y = \boldsymbol{w}^T\boldsymbol{x} + \varepsilon$. Nicméně navíc řeší problém kolinearity tím, že penalizuje vysoké hodnoty koeficientů $\boldsymbol{w}$ vyjma interceptu.

4.1 Regularizovaný reziduální součet čtverců

Hřebenová regrese tedy minimalizuje následují reziduální součet čtverců

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

s regulačním členem $\lambda \geq 0$.

Pro $\lambda = 0$ dostáváme klasickou neregularizovanou lineární regresi. Intercept se nepenalizuje, protože ten jen zajišťuje $\mathrm{E}\,\varepsilon = 0$.

4.2 Minimalizace regularizovaného RSS

Pro účely hřebenové regrese zavedeme

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

Regularizovaný RSS_λ lze vyjádřit Jako

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

Nalezneme parciální derivaci a následně gradient

$$\begin{aligned} \frac{\partial\,\mathrm{RSS}(\boldsymbol{w})}{\partial w_j} &= \sum_{i=1}^{N} 2(Y_i - \boldsymbol{w}^T\boldsymbol{x}_i)(-x_{i,j}) + 2\lambda w_j \\ \nabla\,\mathrm{RSS}(\boldsymbol{w}) &= \sum_{i=1}^{N} 2(Y_i - \boldsymbol{w}^T\boldsymbol{x}_i)(-\boldsymbol{x}_i) + 2\lambda \mathbf{I}'\boldsymbol{w} \\ &= -2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}) + 2\lambda \mathbf{I}'\boldsymbol{w} \end{aligned}$$

A položením rovno nule obdržíme ekvivalent normální rovnice

$$\begin{aligned}\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \lambda\mathbf{I}'\boldsymbol{w} &= 0\\ \boldsymbol{X}^T\boldsymbol{Y} - (\boldsymbol{X}^T\boldsymbol{X} + \lambda\mathbf{I}')\boldsymbol{w} &= 0\end{aligned}$$

Hessova matice zde vychází

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

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

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

Matice $\boldsymbol{X}^T\boldsymbol{X} + \lambda\mathbf{I}'$ je tedy vždy pozitivně definitní a regulární a řešení normální rovnice pro regulované RSS_λ je jednoznačné.

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

Predikce modelu je $\hat{Y} = \hat{\boldsymbol{w}}_\lambda^T\boldsymbol{x}$.

4.3 Modely bázových funkcí

Model lineární regrese, případně hřebenové regrese, modeluje pouze lineární funkci. Množinu příznaků lze však rozšířit jejími transformovanými variantami.

4.3.1 Bázové funkce

Pro $M \in \mathbb{N}$ zvolme M funkcí $\varphi_1, \ldots, \varphi_M$ z $\mathbb{R}^p$ do $\mathbb{R}$ reprezentující transformace. Těmto funkcím říkáme bázové funkce.

Modely bázových funkcí se od předchozích lineárních regresních modelů liší pouze tím, že místo původního $\boldsymbol{x}$ pracuje s $\boldsymbol{\varphi}(\boldsymbol{x}) \coloneqq (1, \varphi_1(\boldsymbol{x}), \ldots, \varphi_M(\boldsymbol{x}))^T$

4.3.2 Volba bázových funkcí

Časté volby bázových funkcí jsou

- $\varphi(\boldsymbol{x}) = x_i$ – původní příznaky
- $\varphi(\boldsymbol{x}) = x_i^2, x_i x_j$ – mocniny, součiny (polynomiální regrese)
- $\varphi(\boldsymbol{x}) = \log(x_i), \sqrt{x_i}, \sin(x_i)$ – nelineární transformace
- $\varphi(\boldsymbol{x}) = \mathbb{1}_A$ – indikátory (dělení prostoru)

4.3.3 Model

Model bázové funkce rozšiřuje možnosti hřebenové regrese a může jej dělat velmi mocným a sofistikovaným nástrojem.

$$\boldsymbol{x} \longrightarrow \boldsymbol{\varphi}(\boldsymbol{x}): \quad Y = \boldsymbol{w}_\lambda^T \boldsymbol{\varphi}(\boldsymbol{x}) + \varepsilon$$

Postup s transformovaným vektorem $\boldsymbol{x}' \coloneqq \boldsymbol{\varphi}(\boldsymbol{x})$ je dál zcela analogický jako u hřebenové regrese a predikce modelu je $\hat{Y} = \hat{\boldsymbol{w}}_\lambda^T \boldsymbol{\varphi}(\boldsymbol{x})$.

5 Výběr příznaků

Často je výhodné počet příznaků před trénováním modelů snížit. Proces, který zvolí nějakou výhodnou podmnožinu příznaků, je feature selection. Tento proces spadá do části předzpracování dat, konkrétně se jedná o podoblast redukce dimenzionality (dimension reduction).

Výběrem příznaků řešíme hned několik problémů najednou:

- Zahozením nerelevantních a redundantních příznaků můžeme významně zlepšit schopnost generalizace modelu (model není zatížený šumem).
- Pomáhá s prokletím dimenzionality (curse of dim.), kdy vysoká dimenze způsobuje řídkost dat a nerelevantnost sousedství.
- Zlepšuje interpretovatelnost modelu.
- Snižuje výpočetní nároky pro trénování.

5.1 Základní metody výběru příznaků

5.1.1 Filtrační metody

Filtrační metody jsou jednoduché a často nevyžadují náročné trénovaní modelů:

- Vyhodit příznaky s příliš nízkým rozptylem (jsou téměř konstantní).
- Vyhodit příznaky, které mají příliš chybějících hodnot.
- Vyhodit redundantní příznaky, které mají vysokou korelaci s jiným příznakem (jsou v datasetu již zastoupeny).
- Vyhodit příznaky, které mají s cílovou proměnnou nízkou korelaci (je dobré v kombinaci s bázovými funkcemi – samotný příznak totiž nemusí vysvětlovanou proměnnou ovlivňovat pouze lineárně).
- U binárních příznaků rozdělit data na dvě populace a provést hypotézu o rovnosti středních hodnot obou populací (dvouvýběrový t-test).
- Provést test nezávislosti mezi příznakem a vysvětlovanou proměnnou.

5.1.2 Obalové metody

Obalové metody používají pro ohodnocení příznaků pomocný model, který na příslušné kandidátní množině příznaků natrénují a pak výkon porovnají se stejným modelem natrénovaný na jiné sadě příznaků.

Pokud se jako pomocný model použije finální model, je výhodou, že se zvolí tu sadu příznaků, která dobře pracuje s vybraným modelem. V takovém případě však snadněji dochází k přeučení. Vybere-li se ale jiný model, sada příznaků může být zvolena nevýhodně pro finální model.

Pokud je kandidátních množin příznaků hodně, je tato metody výpočetně náročná, proto se často používají hladové algoritmy:

- Dopředný výběr (forward selection) začíná s prázdnou množinou a postupně přidává příznak, který v dané iteraci nejvíce zvýší výkonnost modelu.
- Zpětný výběr (backward selection) začíná se všemi příznaky a postupně odebírá ty, které nejméně sníží výkonnost modelu.
- Rekurzivní odebírání příznaků (recursive feature elimination) postupně odebírá podle vnitřního ohodnocení příznaků pomocného modelu (u norm. lineární regrese koeficienty, u stromu příslušný informační zisk)

Algoritmy běží dokud nemají požadovaný počet příznaků, případně mohou skončit i s menším počtem příznaků, pokud přidávání, resp. odstranění, příznaků nesnižuje výkonnost.

5.1.3 Vestavěné metody

Vestavěné metody (embedded methods) provedou výběr příznaků natrénováním modelu na celých datech a pak zahodí příznaky, které se naučil vůbec nepoužívat.

U lineární regrese se jedná o příznaky s koeficientem 0, u rozhodovacího stromu ty, které se nikde nepoužily, atd.

5.2 Lasso

Nejpoužívanější vestavěnou metodou je L_1 regularizovaná lineární regrese, která volí množinu příznaků s nenulovým koeficientem.

Na rozdíl od hřebenové regrese penalizuje absolutní hodnotu koeficientů. Pro $\lambda \geq 0$ minimalizuje

$$\mathrm{RSS}_\lambda^{\mathrm{Lasso}} = \sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{i=1}^{p} |w_i|$$

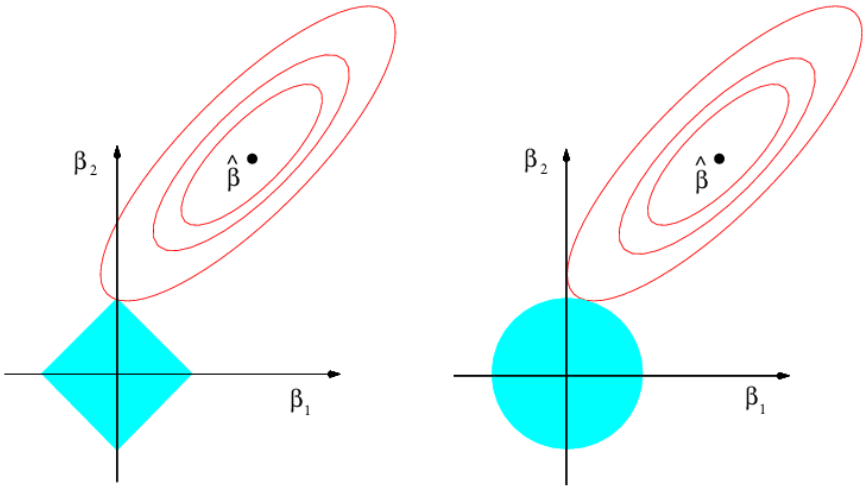
Pro $\lambda = 0$ se jedná o klasickou lineární regresi. Pro $\lambda > 0$ cílí na co nejmenší koeficienty, ale nevadí mu vysoké hodnoty.

$\mathrm{RSS}_\lambda^{\mathrm{Lasso}}$ není diferencovatelný, proto se řešení hledá iterativní metodou:

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

Výhoda modelu Lasso je, že $\hat{\boldsymbol{w}}_\lambda^{\mathrm{Lasso}}$ je řídké – hodně členů je rovno nule. S vyšší λ jsou nuly častější.

Formální důkaz tohoto tvrzení je složitý, ale pro představu lze znázornit obrázkem.



Červeně jsou vykreslené vrstevnice parabolické jámy neregularizované části $\Sigma_{i=1}^{N}(Y_i - \hat{Y}_i)^2$. Přímo na nějakých osách bude hyperkrychle vrstevnici protínat celkem často, zatímco hypersféra prakticky nikdy (pokud neleží minimum přímo na ose).

U Lasso může být nežádoucí, že v případě kolinearity má tendenci volit pouze některé z příznaků. To se projevuje jako nevýhoda především u nových dat – příznak může chybět nebo být sám o sobě zatížený nějakým šumem. Proto existuje model, elastic net, který má oba regularizační členy (L_1 i L_2) a kombinuje výhody obou přístupů:

$$\mathrm{RSS}_{\lambda_1,\lambda_2}^{\text{Elastic net}} = \sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 + \lambda_1 \sum_{i=1}^{p} |w_i| + \lambda_2 \sum_{i=1}^{p} w_i^2$$

$$\hat{\boldsymbol{w}}_{\lambda_1,\lambda_2}^{\text{Elastic net}} = \arg\min_{\boldsymbol{w}} \text{RSS}_{\lambda_1,\lambda_2}^{\text{Elastic net}}(\boldsymbol{w})$$

6 Statistické vlastnosti modelů

V modelu pro trénovací množinu $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$ je $\boldsymbol{\varepsilon}$ náhodný vektor, z čehož plyne, že $\boldsymbol{Y}$ je náhodný vektor, a tedy i $\hat{\boldsymbol{w}}_\lambda$ je náhodný vektor.

6.1 Očekávaná chyba modelu

Protože je odhad $\hat{\boldsymbol{w}}_\lambda$ náhodný vektor, můžeme pomocí kvadratické ztrátové funkce zkoumat očekávanou chybu predikce $Y = \boldsymbol{w}^T\boldsymbol{x} + \varepsilon$ pomocí $\hat{Y} = \hat{\boldsymbol{w}}_\lambda^T\boldsymbol{x}$.

6.1.1 Rozklad očekávané chyby modelu

Předpokládáme-li, nezávislost trénovací množiny s testovací množinou, tedy i nezávislost Y a $\hat{Y}$, pak lze očekávanou chybu spočíst jako

$$\mathrm{E}\, L(Y, \hat{Y}) = \mathrm{E}(Y - \hat{Y})^2 = \ldots = \operatorname{var} Y + \mathrm{E}(\hat{Y} - \mathrm{E}\, Y)^2 = \sigma^2 + \mathrm{MSE}(\hat{Y})$$

První člen odpovídá Bayesovské chybě, která je dána náhodností modelu. Druhý člen značíme $\mathrm{MSE}(\hat{Y})$ (mean squared error), který se dá dále dělit na dva členy:

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

kde $\operatorname{bias}\hat{Y}$ značí vychýlení odhadu $\hat{Y}$ a $\operatorname{var}\hat{Y}$ jeho rozptyl. Finální dekompozice očekávané chyby odhadu je

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

a skládá se z neodstranitelné Bayesovské chyby. kvadrátu vychýlení odhadu a rozptylu odhadu.

Tento rozklad je pouze teoretický, v praxi máme jen celkové MSE, které se snažíme minimalizovat.

6.2 Bias-variance tradeoff

Zpravidla s komplexitou modelu klesá bias (vychýlení), ale roste variance (rozptyl) – model se přeučuje.

U hřebenové regrese s rostoucím regulačním koeficientem klesá rozptyl (regularizace je silnější, odhad $\hat{\boldsymbol{w}}$ je stabilnější), ale zároveň roste vychýlení (koeficienty více "dusíme" a jsou systematicky podhodnocované). Takovému chování v závislosti na hyperparametrech modelu se nazývá bias-variance tradeoff.

V praxi při ladění hřebenové regresi hledáme optimální λ, při které je MSE na validační množině $(Y_i', \boldsymbol{x}_i')$ nejmenší. MSE odhadujeme

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

Při použití hřebenové regrese je rozumné příznaky standardizovat, aby byly rozsahově podobné a penalizované stejně. Příznaky X_i nahradíme

$$X_i \leftarrow \frac{X_i - \bar{X}_i}{\sqrt{s_{X_i}^2}}, \quad \text{kde} \quad \bar{X}_i = \tfrac{1}{N}\sum_{i=1}^{N}(X_i)_j \quad \text{a} \quad s_{X_i}^2 = \tfrac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X}_i)^2$$

6.3 Nestrannost odhadu v metodě nejmenších čtverců

Protože je $\hat{\boldsymbol{w}}_{\mathrm{OLS}}$ bodový odhad (statistika) můžeme zkoumat zda je nestranný. Předpokládáme-li $\mathrm{E}\,\boldsymbol{\varepsilon} = 0$, pak z linearity střední hodnoty platí

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

Dále platí

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

Z čehož plyne

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

$\hat{Y}$ je tedy nestranným bodovým odhadem EY. To znamená, že pro neregularizovanou lineární regresi platí, že je vychýlení nulové:

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

7 Logistická regrese

Logistická regrese predikuje pravděpodobnost jednotlivých hodnot vysvětlované proměnné. Pro binární klasifikaci $Y \in \{0, 1\}$, na kterou se omezíme, tedy vrací pravděpodobnost 1, $P(Y = 1) \in [0, 1]$.

7.1 Použití pro binární klasifikaci

7.1.1 Sigmoida

V modelu použijeme lineární výraz $\boldsymbol{w}^T\boldsymbol{x} = w_0 + w_1x_1 + \ldots + w_px_p$, který má však obor hodnot na celém $\mathbb{R}$. Tento výraz proto dosadíme do funkce, která je ostře rostoucí a má obor hodnot podmnožinu $[0, 1]$. V logistické regresi volíme sigmoidu

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

Jedná se o speciální případ logistické funkce

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}},$$

se supremem $L = 1$, koeficientem růstu $k = 1$ a středem $x_0 = 0$.

Sigmoida má jako definiční obor celé $\mathbb{R}$ a obor hodnot $(0, 1)$. Na celém definičním oboru je ostře rostoucí, limita pro $x \to -\infty$ je 0 a pro $x \to +\infty$ je 1. Také platí, že střed má v bodě 0: $f(0) = \frac{1}{2}$.

7.1.2 Fungování modelu logistické regrese

Binární klasifikace cílové proměnné $Y \in \{0, 1\}$ s p příznaky $X_1, \ldots, X_p$ logistická regrese provede predikcí pravděpodobnosti

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

kde $\boldsymbol{x} = (1, x_1, \ldots, x_p)$ je vektor hodnot příznaků a $\boldsymbol{w} = (w_0, \ldots, w_p)$ je vektor koeficientů. Model zvolí 1 když $\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) > 0.5$, jinak predikuje 0.

7.2 Hranice rozhodnutí

Hranice rozhodnutí je dána rovnicí

$$\mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = 0.5 \quad \Leftrightarrow \quad \boldsymbol{w}^T\boldsymbol{x} = 0 \quad \Leftrightarrow \quad w_0 + w_1x_1 + \ldots + w_px_p = 0$$

To odpovídá nadrovině v prostoru $\mathbb{R}^p$. Hranici tedy tvoří lineární varieta dimenze $p - 1$.

7.3 Logistická regrese jako MLE odhad

Vzhledem k tomu, že u logistické regrese predikujeme pravděpodobnost hodnot proměnné Y, nelze vyloženě měřit chybu takových odhadů a následně je minimalizovat jako u lineární regrese. Proto parametry $\boldsymbol{w}$ odhadujeme MLE (maximum likelihood estimation) metodou maximální věrohodností.

7.3.1 Myšlenka MLE odhadu

Pro parametry $\boldsymbol{w} = (w_0, w_1, \ldots, w_p)$ jsou pravděpodobnosti následující

$$p_1(\boldsymbol{x}; \boldsymbol{w}) = \mathrm{P}(Y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}}}$$
$$p_0(\boldsymbol{x}; \boldsymbol{w}) = \mathrm{P}(Y = 0 \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}}}$$

MLE je takový odhad parametrů, pro které je daná realizace náhodného výběru nejpravděpodobnější (má největší věrohodnost). Metoda maximální věrohodnosti formálně odhaduje hodnotu $\hat{\boldsymbol{w}}$ parametru $\boldsymbol{w}$, která maximalizuje $L(\boldsymbol{w}; \boldsymbol{X})$ na trénovacích datech $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, kde $\boldsymbol{x}_i = (1, x_{i,1}, \ldots, x_{i,p})$ jsou jednotlivé naměřené hodnoty:

$$\hat{\boldsymbol{w}} \in \left\{ \arg\max_{\boldsymbol{w} \in \mathbb{R}^{p+1}} L(\boldsymbol{w}; \boldsymbol{X}) \right\}, \quad \text{kde } L(\boldsymbol{w}; \boldsymbol{X}) = \prod_{i=1}^{N} p_{Y_i}(\boldsymbol{x}_i; \boldsymbol{w})$$

7.3.2 Sestavení optimalizační úlohy pro trénování

Pro tuto funkci se snažíme nalézt maximum (nemusí existovat). Před zderivováním se však často vyplatí věrohodnost zlogaritmovat (log-likelihood), který je na $(0, +\infty)$ prostý a ostře rostoucí a tudíž má maximum ve stejném

bodě.

$$\begin{aligned}\ell(\boldsymbol{w};\boldsymbol{X}) &= \ln L(\boldsymbol{w};\boldsymbol{X}) = \sum_{i=1}^{N} \ln p_{Y_i}(\boldsymbol{x}_i;\boldsymbol{w}) \\ &= \sum_{i=1}^{N} Y_i \ln p_1(\boldsymbol{x}_i;\boldsymbol{w}) + (1-Y_i)\ln p_0(\boldsymbol{x}_i;\boldsymbol{w}) = \dots \\ &= \sum_{i=1}^{N} \left(Y_i \boldsymbol{w}^T \boldsymbol{x}_i - \ln\left(1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}\right)\right)\end{aligned}$$

Parciální derivace a gradient vychází

$$\frac{\partial \ell(\boldsymbol{w};\boldsymbol{X})}{\partial w_j} = \sum_{i=1}^{N}\left(Y_i \boldsymbol{x}_{i,j} - \frac{e^{\boldsymbol{w}^T\boldsymbol{x}_i}\cdot \boldsymbol{x}_{i,j}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}\right) = \sum_{i=1}^{N} \boldsymbol{x}_{i,j}\left(Y_i - p_1(\boldsymbol{x};\boldsymbol{w})\right)$$

$$\nabla \ell(\boldsymbol{w};\boldsymbol{X}) = \boldsymbol{X}^T\left(\boldsymbol{Y}-\boldsymbol{P}\right), \quad \text{kde } \boldsymbol{P} = \left(p_1(\boldsymbol{x}_1;\boldsymbol{w}),\dots,p_1(\boldsymbol{x}_N;\boldsymbol{w})\right)^T$$

Náš odhad leží v bodě, kde věrohodnost nabývá maxima, což nalezneme položením gradientu nule:

$$\nabla \ell(\hat{\boldsymbol{w}};\boldsymbol{X}) = \boldsymbol{X}^T\left(\boldsymbol{Y}-\hat{\boldsymbol{P}}\right) = \boldsymbol{0}$$

Zde neexistuje explicitní řešení a je třeba jej hledat numerickými aproximativními metodami (např. vícerozměrnou Newtonovou metodou lze ukázat, že řešení konverguje k lok. maximu, které je v případě logistické regrese současně globálním maximem).

Výpočet koeficientů logistické regrese je výpočetně náročný. Bez explicitního vzorce je výsledek jen aproximace a je možné, že počítač nic nevrátí (nepodaří se mu nalézt dostatečně dobrá aproximace).

Funkce $\ell(\boldsymbol{w};\boldsymbol{X})$ také žádné maximum mít nemusí. V takovém případě se numerickou metodou pouze snažíme hledat přibližné řešení $\boldsymbol{X}^T\left(\boldsymbol{Y}-\hat{\boldsymbol{P}}\right)=\boldsymbol{0}$.

8 Ensemble metody

Ensemble metody spočívají v konstrukci velkého množství jednoduchých modelů, jejichž predikce se zkombinují do finálního rozhodnutí.

8.1 Bagging (bootstrap aggregating)

Jednou z metod konstrukce slabých podmodelů (weak learners), je trénování nad náhodným výběrem trénovací množiny. Tím se zajistí rozdílnost podmodelů, jejich pestrost a variabilita. Tato metoda se nazývá bootstrap.

8.2 Náhodné lesy

Velmi silným a mocným reprezentantem bagging metody je náhodný les, který agreguje stromy. Tyto stromy jsou zpravidla malé hloubky. Datasety vzniklé bootstrapem způsobí vyšší pestrost stromů, které jsou sami o sobě velmi citlivé a u kterých se malá změna dat často projevuje velmi rozdílnými výsledky. Stromům s rostoucí hloubkou klesá bias, ale roste variance. Kombinací stromů model v jistém smyslu rozptyl krotí. Díky této stabilitě funguje náhodný les v praxi velmi dobře.

8.2.1 Pestrost stromů

Dalším velmi důležitým krokem v konstrukci náhodného lesa je náhodný výběr příznaků při každém dělení listu během konstrukce jednotlivých stromů. Algoritmus náhodně volí nějakou podmnožinu příznaků (o velikosti např. $\sqrt{\cdot}$, log) mezi kterými hledá ten nelepší split. Silné prediktory budou stále hodně zastoupené, nicméně v případě, že se ve výběru nevyskytnou, mají šanci výslednou konstrukci ovlivnit i slabších prediktory (které se mohou nyní vyskytnout i v kořeni nebo jinde ve vyšších hladinách).

8.2.2 Predikce

Náhodný les kombinuje výsledky stromů v případě klasifikace majoritním hlasováním a v případě regrese průměrem.

8.2.3 Shrnutí

Rozhodovací stromy jsou díky kolektivnímu rozhodování velmi robustní, díky průměrování odolné vůči přeučení (s rostoucím počtem průměrovaných

hodnot klesá rozptyl a roste spolehlivost). Naopak ztrácí interpretovatelnost a trénují se znatelně déle (konstrukce stromů lze však provádět paralelně).

U náhodného lesa ladíme jako hyperparametry četnost podmodelů, maximální hloubku jednotlivých stromů, počet/poměr náhodně vybraných příznaků při větvení.

8.3 Boosting

8.3.1 Vážené hodnoty u rozhodovacího stromu

Princip vážených dat u rozhodovacího stromu je následovný: Každému prvku $\boldsymbol{x}_i$ v trénovací množině přiřadíme nezápornou váhu $w(\boldsymbol{x}_i)$ tak, že

$$\sum_{i=1}^{N} w(\boldsymbol{x}_i) = 1$$

Při učení se tak význam vah u výpočtu informačního zisku trochu mění.

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

kde $\mathcal{D}_0, \mathcal{D}_1$ jsou příslušné podmnožiny $\mathcal{D}$, jsou nyní koeficienty definované jako

$$t_0 = \frac{\sum_{\boldsymbol{x}\in\mathcal{D}_0} w(\boldsymbol{x})}{\sum_{\boldsymbol{x}\in\mathcal{D}} w(\boldsymbol{x})} \quad \text{a} \quad t_1 = \frac{\sum_{\boldsymbol{x}\in\mathcal{D}_1} w(\boldsymbol{x})}{\sum_{\boldsymbol{x}\in\mathcal{D}} w(\boldsymbol{x})}.$$

Takhle naučený strom klade větší důraz na to, aby správně predikoval datové body s vyšší váhou.

8.3.2 AdaBoost

AdaBoost podobně jako bagging konstruuje velké množství stromů, jejichž výsledky pro finální predikci kombinuje. Na rozdíl od náhodného lesa však spolu vybudované stromy souvisí. Stromy se konstruují sekvenčně, přičemž každý strom v posloupnosti se soustředí na ty datové body, ve kterých předchozí strom chyboval (zvýší jim váhu).

Konstrukce stromů V algoritmu se vyskytuje hodnota λ (learning rate), která v případě, že je menší než jedna trénování zpomaluje a zabraňuje přeučení. Pro $\lambda \geq 1$ je konstrukce významných stromů rychlejší, ale zároveň

volatilní (zběsile řeší "aktuální" problém, kterým rychleji vznikají špatně klasifikované hodnoty, na které se nesoustředil).

Algorithm AdaBoost

Require:
Trénovací data $\mathcal{D}$.
Learning rate $\lambda \geq 0$.

1: Přiřaď všem datovým bodům váhu $w_i = \frac{1}{N}$.
2: Polož $m = 1$ (indexuje stromy v posloupnosti).
3: **while** $m \leq$ `n_estimators` **do**
4: Natrénuj strom $T^{(m)}$ s váhami w_i.
5: Spočítej součet vah $e^{(m)}$ špatně klasifikovaných hodnot v $T^{(m)}$.
6: Pokud je $e^{(m)} = 0$, skonči.
7: Polož

$$\alpha^{(m)} = \lambda \log \frac{1 - e^{(m)}}{e^{(m)}}.$$

8: Špatně klasifikovaným prvkům v $T^{(m)}$ přiřaď nové váhy

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

9: Váhy znormalizuj, aby součet byl 1, a inkrementuj m.
10: **end while**
11: **return** $T^{(1)}, T^{(2)}, \ldots, T^{(m)}$

O regularizaci Regularizaci většinou doprovází nějaký tradeoff. V případě AdaBoostu se jedná o pomalejší trénování (je potřeba víc stromů), nebo v případě hřebenové regrese způsobuje vychýlení odhadu $\hat{\boldsymbol{w}}$.

Predikce Model každému stromu $T^{(m)}$ přiřadí váhu $\alpha^{(m)}$ a při klasifikaci spočte W_1 součet vah stromů, které predikují 1, a W_0 součet vah stromů, které predikují 0. Ve finále predikuje tu hodnotu, která má mezi všemi stromy vyšší celkovou váhu.

Existuje i verze AdaBoostu pro multiclass klasifikaci: AdaBoost-SAMME, a pro regresi: AdaBoost.R2. AdaBoost nemusí nutně používat stromy. Umí používat jakýkoliv model, který umí správně pracovat s váženými datovými body.

9 Evaluace modelů

Jedna z klíčových aspektů strojového učení je, aby natrénovaný model byl schopný generalizace – fungovat dobře na nových vstupech.

Při trénování můžeme natrénovat celou řadu modelů (např. ze stejné rodiny ale s různými sadami hyperparametrů). Pro to, abychom mohli modely porovnat, potřebujeme mít nějakou kvantitativní míru výkonnosti – metriku. To jakou zvolíme často záleží na charakteru problému. Někdy chceme minimalizovat drobné chyby, někdy celkovou chybovost, apod.

9.1 Ztrátová funkce

Uvažujme model natrénovaný na vstupu $\boldsymbol{X}$ s vysvětlovanou proměnnou Y. Takový model zpravidla není dokonalý a pro $\boldsymbol{X}$ predikuje nějaké $\hat{Y} \equiv \hat{Y}(\boldsymbol{X})$. Funkci L, měřící chybu této predikce, nazýváme ztrátovou funkcí (loss function).

Regrese V případě regrese je typická kvadratická ztrátová funkce (squared error):

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

případně L_1 ztrátová funkce měřící absolutní chybu (absolute error):

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

Klasifikace U binární klasifikace se často odhaduje pravděpodobnost

$$\hat{p} = \hat{P}(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}),$$

pro kterou se nabízí následující ztrátová funkce (binary cross-entropy loss function)

$$L(Y, \hat{Y}) = -Y \log \hat{p} - (1 - Y) \log(1 - \hat{p})$$

9.1.1 Trénovací chyba

Při trénování se snažíme minimalizovat průměrnou hodnotu ztrátové chyby přes všechny prvky v trénovací množině (trénovací chybu, test error):

$$\overline{\mathrm{err}}_{\mathrm{train}} = \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} L(Y_i, \hat{Y}(\boldsymbol{x}_i))$$

V případě regrese se tedy bude jednat například o:

$$\text{MSE}_{\text{train}} = \frac{1}{N}\sum_{i=1}^{N}(Y - \hat{Y})^2 \quad \text{nebo} \quad \text{MAE}_{\text{train}} = \frac{1}{N}\sum_{i=1}^{N}|Y - \hat{Y}|$$

A u binární klasifikaci o:

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

Řešení minimalizující trénovací chybu (test error) lze u některých modelů spočíst explicitně (closed-form solution), např. u lin. regrese. Pro většinu modelů to nelze a musíme jej počítat numericky iterativními metodami (např. gradientním sestupem).

9.1.2 Testovací chyba

Testovací chyba (test error) je střední chyba na novém vstupu $\boldsymbol{X}$ při dané trénovací množině $\mathcal{D}$:

$$\text{Err}_{\mathcal{D}} = \text{E}(L(Y, \hat{Y}(\boldsymbol{X})) \mid \mathcal{D})$$

kterou můžeme odhadnout

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

Nejobecnější mírou schopnosti modelu generalizovat je očekávaná testovací chyba (expected test error), která je střední hodnotou testovací chyby pro náhodný výběr trénovací množiny $\mathcal{D}$:

$$\text{Err} = \text{E}(\text{Err}_{\mathcal{D}}) = \text{E}(L(Y, \hat{Y}(\boldsymbol{X})))$$

Jedná se tedy odhad $\text{Err}_{\mathcal{D}}$ s neznámým $\mathcal{D}$.

9.2 Evaluační scénáře

Z pohledu evaluace máme dva úkoly:

- Výběr modelu - odhadnout chybu různých modelů za účelem výběru nejlepšího.
- Ohodnocení modelu - odhadnout testovací chybu finálního modelu.

9.2.1 Hold-out

Pokud máme dostatek dat, dělíme data na část

- Trénovací - k natrénování konkrétních modelů.
- Validační - k výběru nejlepší sady hyperparametrů.
- Testovací - k odhadu testovací chyby, kterou očekáváme na nových datech.

Pro získání neoptimistického odhadu výkonnosti modelu, musí být testovací část skutečně nový vstup, který nijak neovlivnil parametry ani volbu modelu.

9.2.2 k-fold cross-validation

Pokud nemáme dostatek dat, je často nerozumné je dělit na trénovací, validační a testovací. Vedlo by to na nedostatek dat pro správně natrénovaní, dobrou volbu správného modelu a spolehlivého odhadu testovací chyby.

S křížovou validací se obejdeme bez validační množiny.

Algorithm Cross-validation

Require: $2 \leq k \leq N$

1: Trénovací data $\mathcal{D}$ rozděl na k podobně velkých podmnožin $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
2: Pro $j = 1, \ldots, k$ natrénuj model s danými hyperparametry na $\mathcal{D} \setminus \mathcal{D}_j$.
3: Na $\mathcal{D}_j$ odhadni chybu modelu e_j.
4: **return** cross-validation error

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

Takto odhadneme cross-validační chybu pro všechny uvažované sady hyperparametrů a zvolíme model s nejnižší takovou chybou. Zvolený model s nejlepšími hyperparametry natrénujeme na celé množině $\mathcal{D}$.

Volba k je kvůli výpočetním nárokům obvykle malá (5, 10). Pro velmi malé datasety může však být únosná i volba $k = N$ (leave-one-out cross-validation).

Cross-validation error Cross-validation error je odhad očekávané testovací chyby Err a ne testovací chyby $\text{Err}_{\mathcal{D}}$.

Při skutečně malém datasetu je možné zvolit dvoustupňovou křížovou validaci, která si neodkládá testovací sadu, ale zanořuje se s křížovou validací o jednu stupeň níž, přičemž vnitřní cross-validační chyba se používá pro výběr modelu a vnější pro odhad očekávané chyby.

Vnější cross-validation chyba je pouze jedna ("společná") a odpovídá očekávané testovací chybě celé procedury pro výběr nejlepšího modelu, ale nejlepší model teprve musíme získat (natrénování zvoleného modelu na celé množině $\mathcal{D}$).

Finální model Ve všech metodách lze po výsledném odhadu testovací chyby přetrénovat nejlepší model na celém datasetu a pro odhad výkonnosti použít testovací chybu, resp. očekávanou testovací chybu, z předchozího kroku. U hold-out lze před evaluací na trénovacích datech ještě přetrénovat model na (trénovací + validační) množině.

9.3 Evaluace regrese

Nejčastější volba ztrátové funkce u regresních úloh je MSE (mean squared error) nebo MAE (mean absolute error):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \qquad \text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|$$

Zatímco MSE je citlivější na velká residua, MAE se spíše soustředí na celkovou chybu a odlehlým hodnotám se chová "spravedlivě". Jednotky MSE jsou v kvadrátech, proto se také často udává přeškálované RMSE, které má interpretovatelné jednotky vysvětlované proměnné:

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

Pro nezáporné hodnoty lze použít ztrátovou funkcí RMSLE, které se soustředí na relativní míru odchylek:

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

Koeficient determinace R^2 (coefficient of determination) vyjadřuje podíl

variability cílové proměnné, kterou model vysvětluje:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = \frac{\sum_{i=1}^{N}(\log Y_i - \log \hat{Y}_i)^2}{\sum_{i=1}^{N}(\log Y_i - \log \bar{Y})^2},$$

kde TSS je total sum of squares (RSS modelu, který predikuje $\bar{Y}$ pro každý datový bod).

9.4 Evaluace klasifikace

Matice záměn U klasifikace je konstrukce a interpretace ztrátových funkcí obecně problematické. Využívá se proto matice záměn:

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

(T/F - True/False; P/N - Positive/Negative)

Z matice záměn můžeme vyvodit následující míry:

		Skutečnost	
$\text{P}(\hat{Y} \mid Y)$		$Y = 1$	$Y = 0$
Predikce	$\hat{Y} = 1$	$\text{TPR} = \frac{\text{TP}}{N_+}$	$\text{FPR} = \frac{\text{FP}}{N_+}$
	$\hat{Y} = 0$	$\text{FNR} = \frac{\text{FN}}{N_-}$	$\text{TNR} = \frac{\text{TN}}{N_-}$

- True positive rate (TPR): senzitivita (recall).
- False positive rate (FPR): type I error rate.
- False negative rate (FNR): type II error rate.
- True negative rate (TNR): specificita (specificity).

Často používanou mírou je také odhad $P(Y = 1 \mid \hat{Y} = 1)$ – precision:

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

9.4.1 Evaluační míry binární klasifikace

Nejpoužívanější mírou je přesnost (accuracy) – odhad $P(Y = \hat{Y})$:

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{N}$$

Tato míra však není spolehlivá pro nevybalancovaná data, kdy modelu stačí predikovat pouze majoritní třídu. V takovém případě se používá F_1 score:

$$F_1 = \frac{2}{\text{PPV}^{-1} + \text{TPR}^{-1}},$$

kde hodnota $P(Y = 1)$ je velmi malá.

9.4.2 ROC a AUC

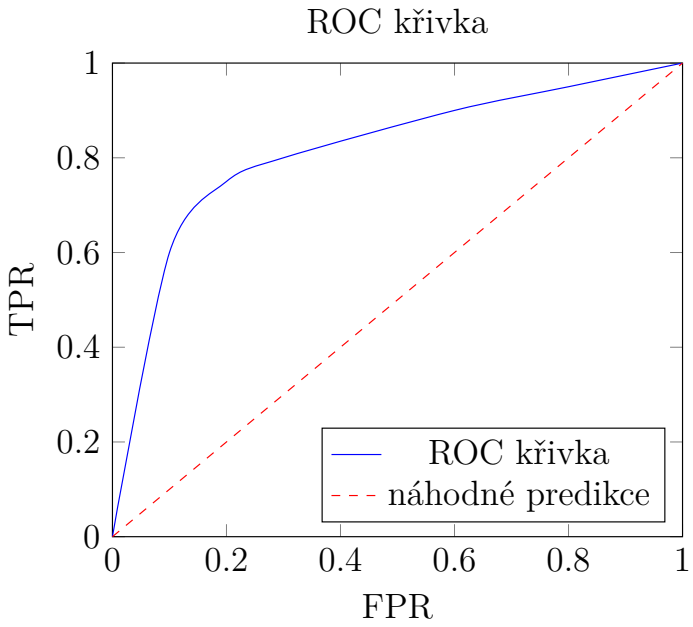
Modely binární klasifikace často odhadují $p(\boldsymbol{X}) = P(Y = 1 \mid \boldsymbol{X})$ a predikují

$$\hat{Y} = \mathbb{1}_{\hat{p}>0.5}$$

Hodnotu 0.5 však můžeme parametrizovat pomoc $\tau \in [0, 1]$ a získat tak pro různé hranice různé predikce:

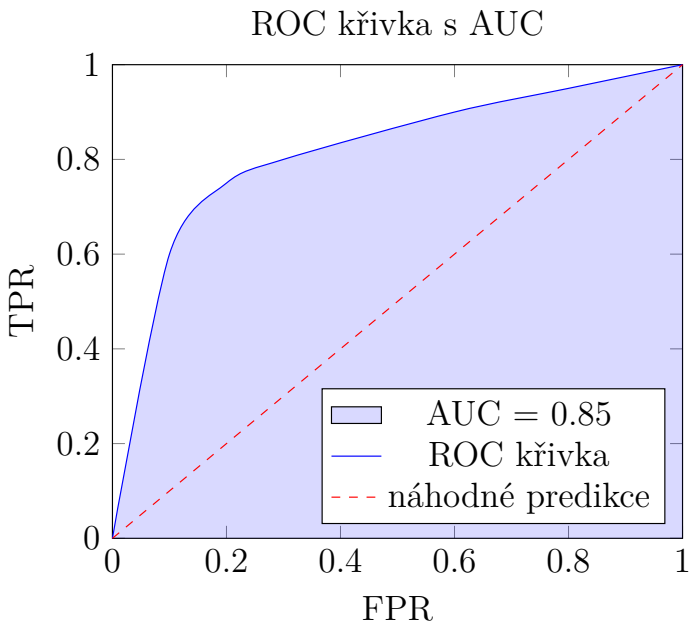
$$\hat{Y}_\tau = \mathbb{1}_{\hat{p}>\tau}$$

ROC S hodnotou τ od 0 do 1 se různě mění TPR a FPR, jejichž vztah lze vykreslit a následně dále zkoumat pomocí ROC křivky (receiver operating characteristic curve).



Pro dobrý model se křivka přimyká k levé a horní ose (strmě roste).

AUC Kvalita modelu, pro kterou máme ROC křivku, vyhodnocujeme plochou pod křivkou AUC (area under the curve). Model s náhodnými predikcemi má AUC = 0.5, dokonalý model by měl AUC = 1.



10 Nesupervizované učení

Nesupervizované učení se zabývá problematikou neoznačených dat. Cílem je tedy datům porozumět, případně odhalit nějakou vnitřní strukturu. To většinou znamená odhalit nějaké omezené oblasti v prostoru příznaků, ve kterých se data vyskytují častěji (data se zpravidla nevyskytují náhodně, často tvoří nějaké shluky). Bývají nějakým způsobem lokalizovaná (např. v nějakých méně dim. oblastech, apod.).

Obecným problémem je, že zpočátku typicky o datech nemáme žádnou informaci. Tu se snažíme různými metodami z dat extrahovat. Na rozdíl od nesupervizového učení zde není jasný postup, nebo přímočarý způsob, jak úspěšnost řešení vyhodnocovat.

Z pohledu teorie pravděpodobnosti Uvažujme realizaci náhodného vektoru $\boldsymbol{X} = (X_1, \ldots, X_p)^T$ v prostoru $\mathcal{X}$, které je v případě binárních příznaků $\mathcal{X} = \{0, 1\}^p$ a v případě spojitých příznaků $\mathcal{X} = \mathbb{R}^p$.

Porozuměním vnitřní struktuře znamená porozumění rozdělení $\boldsymbol{X}$ tak, že jsme schopni spolehlivě predikovat pravděpodobnost $\mathrm{P}(\boldsymbol{X} \in O)$, kde O je nějakou zajímavou podmnožinou $\mathcal{X}$.

Odhadujeme tedy pravděpodobnostní hustotu $f_{\boldsymbol{X}}(x_1, \ldots, x_p)$, respektive pravděpodobnostní funkci $\mathrm{P}_{\boldsymbol{X}}(X_1 = x_1, \ldots, X_p = x_p)$.

11 Shluková analýza

Shluková analýza (clustering) je metoda nesupervizovaného učení, která rozřadí data do nějakých shluků tak, aby byly blízké body v jednom shluku a vzdálené v různých. Jak přesně formalizovat takovou úlohu je problematické, proto může mít různá řešení v závislosti na tom, jak definujeme vzdálenost.

Vzdálenost Pro připomenutí, definice metriky byla zevedena v sekci 2.1.2.

Formalizace úlohy shlukování Úloha na vstupu dostane metrický prostor $\mathcal{X}$ se vzdáleností d, množinu dat $\mathcal{D} \subset \mathcal{X}$ a počet požadovaných shluků k. Na výstupu se požaduje nějaký rozklad množiny $\mathcal{D}$ na k shluků. To znamená $C_1, \ldots, C_k \subset \mathcal{D}$, kde $C_i \neq C_j$ pro všechna $i \neq j$, přičemž

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

11.1 Hierarchické shlukování

Hierarchické shlukování používá hladový aglomerativní přístup popsaný v následujícím algoritmu.

Algorithm Hierarchické shlukování

Require:
Množina dat $\mathcal{D}$ a metrika pro vzdálenost shluků.
Požadovaný počet shluků k (jinak $k = 1$).

1: Uvažuj každý bod jako jednoprvkový shluk (celkem $|\mathcal{D}|$ shluků).
2: **while** počet shluků $> k$ **do**
3: Nalezni dva nejbližší shluky.
4: Tyto dva shluky spoj do jednoho.
5: **end while**

11.1.1 Měření vzdáleností shluků

Po volbě metriky dvou bodů $d(a, b)$ je ještě nutné zvollit vzdálenost dvou shluků $D(A, B)$. Obvykle se použije jedna z následujících:

- Single linkage, který generuje dlouhé řetězce

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

- Complete linkage, který generuje kompaktní buňky

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

- Average linkage, který měří podle průměrné vzdálenosti všech bodů

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

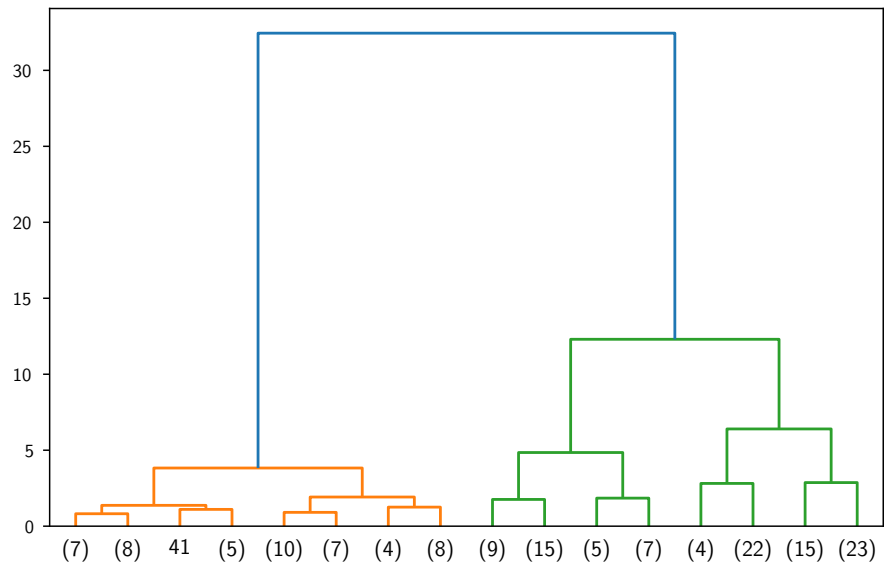
- Wardova metoda (pouze pro $\mathcal{X} = \mathbb{R}^p$), která minimalizuje nárůst vnitřního rozptylu buněk

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

  kde $\bar{\boldsymbol{x}}_S$ značí geometrický střed (centroid) shluku S.

11.1.2 Dendrogram

Celý proces aglomerativního shlukování lze reprezentovat dendrogramem. To je strom, který má v listech počáteční jednoprvkové shluky a ve vrcholech spojení. Obvykle algoritmus běží, dokud nespojí všechny shluky, v takovém případě má v kořeni finální shluk. Strom se vykresluje tak, že listy jsou ve výšce 0 a jednotlivé vrcholy ve výšce podle vzdálenosti shluků, které se v daném bodě spojily (vzdálenost, kterou musely "překonat").



Máme-li požadované k, příslušně dendrogram rozřízneme tak, aby vodorovná čára protla přesně k hran. Každá taková hrana odpovídá nějakému shluku. Případně můžeme zvolit nějakou danou výšku (přijatelnou vzdálenost shluků).

Shrnutí Výhodou Hierarchického shlukování je flexibilita. Můžeme algoritmus provést a pak následně rozhodovat o rozdělení. Při změně počtu shluků stačí jen změnit místo, ve kterém provádíme řez (struktura dendrogramu se nemění).

Nevýhodou hierarchického shlukování je výpočetní náročnost $\mathcal{O}(n^3)$, v případě single/complete linkage lze zefektivnit na $\mathcal{O}(n^2)$. Pro velké datové soubory se proto nehodí.

11.2 Algoritmus k-means

11.2.1 Shlukování jako optimalizační úloha

Ke shlukování lze přistupovat jako k optimalizační úloze, ve které minimalizujeme účelovou funkci (objective function).

Pro dané k hledáme rozklad $C = (C_1, \ldots, C_k)$ množiny dat $\boldsymbol{X}$ v metrickém prostoru $\mathcal{X} = \mathbb{R}^p$ vybavenou Eukleidovskou vzdáleností minimalizující účelovou funkci

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

Jedná se o průměrnou kvadratickou vzdálenost vnitřních bodů.

Souvislost účelové funkce s geometrickým středem Lze ukázat, že takový součet lze vyjádřit jako součet kvadrátů vzdálenosti od geometrického středu příslušných shluků.

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

11.2.2 k-means

Algoritmus k-means využívá předchozí rovnosti a jediné, s čím "manipuluje", jsou středy shluků. Nalezení globálního minima je NP-těžká úloha, proto se používá iterativní způsob, který hladově zmenšuje hodnotu účelové funkce.

Algorithm k-means

Require:

Množina dat $\mathcal{D}$ a požadovaný počet shluků k.
Počáteční rozmístění $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$.

1: **while** hodnota účelové funkce "hodně" klesá **do**
2: Roztřiď $\mathcal{D}$ do shluků

$$Ci = \{\boldsymbol{x} \in \mathcal{D} \mid i = \arg\min_j \|\boldsymbol{x} - \boldsymbol{\mu}_j\|\}.$$

3: Přepočítej středy nových clusterů

$$\boldsymbol{\mu}_i \leftarrow \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}.$$

4: **end while**

Lze ukázat, že v každém kroku algoritmu se hodnota účelové funkce zmenší, případně zůstane stejná (v průběhu algoritmu je hodnota účelové funkce nerostoucí).

Běh algoritmu skončí, pokud se žádný ze středů nepřepočítá nebo pokud je rozdíl hodnot účelové funkce mezi iteracemi dostatečně malá.

Výsledek algoritmu hodně záleží na počátečním rozmístění středů, proto se typicky algoritmus spouští víckrát s různými počátky, z nichž si následně vybere shluk s nejnižší účelovou funkcí. Existuje rozšíření algoritmu, které vybírá počátky tak, aby byly "chytře" rozmístěné (k-means++).

Volba počtu shluků k-means algoritmus dává různé výsledky pro různá k, proto je důležité jej stanovit dopředu. S rostoucím k, zřejmě hodnota účelové funkce vždy klesá (případně neroste), proto je volba často subjektivní. Jednou z metod (i když nespolehlivých) je metoda lokte (elbow method), která předpokládá to, že existuje nějaké ideální $k*$, takové, že hodnota účelové funkce s rostoucím $k < k*$ rychle klesá, a pro $k > k*$ roste méně. Zlomový bod je předpokládaný loket, který ale není vždy úplně jednoznačný či optimální. Alternativní způsob ohodnocení clusteringu je například silhouette.

11.3 Silhouette skóre

Silhouette score je jednoduchou metodou pro evaluaci shlukování pro různá porovnání nebo i pro výběr optimálního počtu shluků (pro k-means).

Uvažujme nějaké shlukování $\mathcal{D} = C_1 \cup \ldots \cup C_k$ na metrickém prostoru $\mathcal{X}$ s metrikou $d(x, y)$, kde $\forall x \in \mathcal{D} : x \in C_{j(x)}$. Pro každý bod $x \in \mathcal{D}$ se spočítá:

- průměrná vzdálenost bodu od ostatních bodů ve stejném shluku:

$$a(x) = \frac{1}{|C_{j(x)}| - 1} \sum_{y \in C_{j(x)}, y \neq x} d(x, y)$$

- průměrná vzdálenost bodu od bodů v jiném shluku:

$$d(x, C_i) = \frac{1}{|C_{j(x)}|} \sum_{y \in C_i} d(x, y)$$

- průměrná vzdálenost bodu od nejbližšího jiného shluku:

$$b(x) = \min_{i \neq j(x)} d(x, C_i)$$

Evaluace pomocí Silhouette skóre Finální skóre bodu $x \in \mathcal{D}$ se počítá následujícím vzorcem:

$$s(x) = \frac{b(x) - a(x)}{\max a(x), b(x)}$$

V případě jednoho shluku je $s(x) = 0$.

Pokud je $s(x)$ blízko 1, $b(x) \gg a(x)$, pak je bod dobře zařazen. Pokud je $s(x)$ blízko 0, $b(x) \approx a(x)$, je bod na kraji svého a sousedního shluku (také v pořádku). Je-li však $s(x)$ blízko -1, $b(x) \ll a(x)$, je bod špatně zařazen. Poznamenejme, že v takovém případě nestačí bod jen přemístit do druhého – změnilo by celé ohodnocení (mohlo by být ve výsledku shlukování ještě zhoršit).

S ohodnocením jednotlivých bodů lze nyní provést evaluaci jednotlivých clusterů i celého shlukování:

$$s(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} s(x) \qquad s = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} s(x)$$

Vyšší hodnota (blíže k 1) značí lepší shlukování (lepší celkové umístění jednotlivých bodů).