

Strojové učení

Studijní materiál ke státnicím (S3)

Jakub Klesa

Pokryté předměty: NAIL121 (Seminář dobývání znalostí), NPFL054 (Úvod do SU v R), NPFL129 (Úvod do SU v Pythonu), NPGR035 (SU v počítačovém vidění)

Obsah

Úvod	2
1 Učení s učitelem a evaluace modelů	3
1.1 Úlohy učení s učitelem	3
1.2 Ohodnocení modelu: testovací data, křížová validace, maximum likelihood	3
1.3 Evaluační metriky pro klasifikaci	4
1.4 Evaluační metriky pro regresi	8
1.5 Naivní Bayesův klasifikátor	8
1.6 Praktický postup: od dat k modelu	9
2 Overfitting a regularizace	10
2.1 Generalizační chyba, underfitting a overfitting	10
2.2 Kapacita modelu	11
2.3 Regularizace	12
2.4 Prokletí dimenzionality	13
3 Učení založené na příkladech: k nejbližších sousedů	15
3.1 Princip: líný, neparametrický model	15
3.2 Klasifikace: většinové hlasování	16
3.3 Regrese: (vážený) průměr sousedů	16
3.4 Vzdálenostní metriky a normalizace příznaků	16
3.5 Volba k : vliv na hranici, overfitting, křížová validace	17
3.6 Výpočetní cena predikce	17
3.7 Řešený příklad	18
4 Lineární regrese	19
4.1 Model a chybová funkce	19
4.2 Analytické řešení: metoda nejmenších čtverců	20
4.3 Trénování pomocí SGD	22
4.4 OLS vs. SGD a krátce o nelinearitě	24
5 Logistická regrese	24
5.1 Binární logistická regrese	25
5.2 Ztrátová funkce, gradient a SGD	26
5.3 Klasifikace do více tříd: softmax	27
5.4 Řešené příklady	28

6	Rozhodovací stromy	30
6.1	Algoritmus učení: hladově shora dolů (rozděl a panuj)	30
6.2	Kritéria větvení	31
6.3	Prořezávání (pruning)	33
6.4	Řešené příklady	34
7	Metoda podpůrných vektorů (SVM)	39
7.1	Lineárně separabilní třídy: maximální margin	39
7.2	Lineárně neseperabilní třídy: soft margin	41
7.3	Jádrové funkce: kernel trick	42
7.4	Klasifikace do více tříd	44
7.5	Řešený příklad	44
8	Kombinace více modelů	46
8.1	Bagging (bootstrap agregace)	46
8.2	Boosting a algoritmus AdaBoost	47
8.3	Random forest (náhodný les)	48
8.4	Řešené příklady	49
9	Statistické testy	51
9.1	Rámec testování hypotéz	51
9.2	Studentův t-test	52
9.3	Chí-kvadrát test dobré shody	54
10	Učení bez učitele: shlukování a redukce dimenze	56
10.1	Shlukování algoritmem k -means	56
10.2	Hierarchické shlukování	60
10.3	Redukce dimenze: analýza hlavních komponent (PCA)	61
10.4	Měkké shlukování: EM a směs Gaussovců	63
	Shrnutí	64
	Zdroje	67

Úvod

Tento materiál pokrývá okruh **S3 – Strojové učení**, druhý ze dvou okruhů specializace UI-SU. Výklad stojí především na kurzu **NPFL129 Úvod do strojového učení v Pythonu** (Milan Straka, výuka Jindřich Libovický; ÚFAL MFF, zdrojový repozitář <https://github.com/ufal/npfl129>). Téma *metody podpůrných vektorů* (SVM) čerpá z ročníku, kdy bylo v osnově NPFL129 (jádrové funkce a SVM, přednášky 6–7), *statistické testy* (t-test, χ^2) navazují na okruh **M5** (Pravděpodobnost a statistika). Strojové učení je probíráno i v úvodním kurzu UI (okruh **S1**, sekce *Strojové učení*) na úrovni „základní principy“; *tento* okruh jde do hloubky a na S1 jen navazuje (kde stačí přehled, odkazujeme tam).

Role důkazů. Požadavky okruhu jmenují pojmy, modely, algoritmy a metody evaluace, ne formální důkazy. Těžiště je v definici pojmu, v popisu a aplikaci algoritmu na konkrétních datech (matice záměn a metriky, stavba rozhodovacího stromu podle informačního zisku, iterace k -means, krok gradientního sestupu, odhad koeficientů regrese, statistický test) a ve zdůvodnění. Výzvy „odvoďte“ a „ukážte“ se v zařazených otázkách týkají *výpočtu*: derivace ztrátové funkce logistické regrese (sekce 5) a explicitního řešení nejmenších čtverců (sekce 4). Věty a vlastnosti (např. konvergence k -means, optimum nejmenších čtverců) proto uvádíme se zněním, intuicí a krátkým *odvozením* tam, kde pomáhá pochopení.

Do výkladu je zařazeno všech 35 reálných otázek SZS k okruhu z let 2022–2026 jako řešené příklady; jejich originální zadání (a grafy oříznuté z PDF) jsou i v samostatném dokumentu *Příklady otázek*. Použité zdroje jsou uvedeny na konci.

Značení. *Vstup* (vektor příznaků) je $\mathbf{x} \in \mathbb{R}^D$, jednotlivé příznaky $x_1, \dots, x_D$. *Trénovací množina* má N příkladů; *matice dat* $\mathbf{X} \in \mathbb{R}^{N \times D}$ má příklad $\mathbf{x}_i$ na i -tém řádku. *Cíl* (target) je t – reálné číslo u regrese ($t \in \mathbb{R}$), třída u klasifikace ($t \in \{0, 1, \dots, K-1\}$, případně $t \in \{0, 1\}$ binárně); vektor cílů $\mathbf{t} \in \mathbb{R}^N$. *Predikci* modelu značíme $y(\mathbf{x})$ nebo $\hat{t}$. *Váhy* lineárního modelu jsou $\mathbf{w} \in \mathbb{R}^D$, *bias* b (ve statistickém značení regrese koeficienty $\beta_0, \beta_1, \dots$); skalární součin $\mathbf{w}^\top \mathbf{x} = \sum_i w_i x_i$. *Rychlost učení* (krok) je α , *regularizační síla* λ (nebo C u SVM). $\sigma(z) = 1/(1 + e^{-z})$ je *logistická* (sigmoidní) funkce. Logaritmus $\log$ bez základu v entropii je $\log_2$ (entropie v bitech). E je střední hodnota, Var rozptyl, $\|\cdot\|$ eukleidovská (L^2) norma.

1 Učení s učitelem a evaluace modelů

Tato sekce zavádí společný rámec celého okruhu: co je *učení s učitelem*, jak rozlišujeme *klasifikaci* a *regresi*, a především **jak model ohodnotíme** – testovacími daty, křížovou validací a principem maximum likelihood estimation, a jakými *metrikami* měříme jeho kvalitu. Konkrétní modely (regrese, stromy, SVM, ...) přijdou v dalších sekcích; evaluace je společná všem.

Poznámka. [Vztah k S1] Okruh S1 (sekce *Strojové učení*) podává učení s učitelem na úrovni „základní principy“: druhy učení, klasifikace vs. regrese, Ockhamova břitva. Zde to nezdvoujeme – jen krátce zrekapitulujeme a rovnou jdeme do hloubky, kterou S1 výslovně přenechává okruhu S3 (evaluační metriky, křížová validace).

1.1 Úlohy učení s učitelem

Definice – Učení s učitelem, klasifikace, regrese

Při **učení s učitelem** máme *trénovací množinu* N příkladů $(\mathbf{x}_i, t_i)$, kde $\mathbf{x}_i \in \mathbb{R}^D$ je vstup (vektor příznaků) a t_i je *cíl* (požadovaný výstup, label). Hledáme funkci (model) y , která dobře předpovídá cíl i pro *nové, neviděné* vstupy. Podle typu cíle:

- **regrese** – cíl je reálné číslo, $t \in \mathbb{R}$ (cena, teplota, výnos);
- **klasifikace** – cíl je jedna z K tříd, $t \in \{0, 1, \dots, K-1\}$ (spam/ham, druh kytky). Model může vrátit jen třídu, nebo celé *rozdělení pravděpodobností* tříd.

Intuice. Příklady chápeme jako nezávislé vzorky z neznámého *rozdělení generujícího data*. Cílem není zapamatovat trénovací množinu (to je pouhá *optimalizace*), ale **generalizovat** – dobře předpovídat data ze stejného rozdělení, která model neviděl. Optimalizace na trénovacích datech je jen prostředek; tím, co nás zajímá, je chyba na nezávislých datech. Celá tato sekce je o tom, jak takovou chybu *pochtivě změřit*.

1.2 Ohodnocení modelu: testovací data, křížová validace, maximum likelihood

Trénovací, validační a testovací množina

Chybu na neviděných datech nelze měřit na trénovací množině (model je na ni „nalaďen“ a chybu podhodnotí). Data proto dělíme:

- **trénovací** – na nich se učí parametry modelu (váhy);
- **validační** (development) – na nich ladíme *hyperparametry* (např. stupeň polynomu, sílu regularizace, k u k -NN) a rozhodujeme o *včasném zastavení*; viz sekce 2;
- **testovací** – použijí se *jen jednou na konci* k odhadu generalizační chyby. Kdo ladí podle testovacích dat, klame sám sebe.

Definice — k -násobná křížová validace

Dat je vždy málo na to, vyčlenit velkou validační množinu. **k -násobná křížová validace** (k -fold cross-validation) data rozdělí na k stejných dílů (*foldů*). Postupně bere každý díl jako validační a zbylých $k - 1$ dílů jako trénovací; natrénuje a změří chybu. Výsledný odhad je *průměr* k takto získaných chyb. Využije tak na validaci postupně *všechna* data, ale model nikdy netestuje na datech, na nichž se učil.

běh 1	valid.	trén.	trén.	trén.	trén.
běh 2	trén.	valid.	trén.	trén.	trén.
běh 3	trén.	trén.	valid.	trén.	trén.
běh 4	trén.	trén.	trén.	valid.	trén.
běh 5	trén.	trén.	trén.	trén.	valid.

5-násobná křížová validace: data jsou v 5 dílech, v každém z 5 běhů slouží jiný díl jako validační (žlutě) a zbylé čtyři k trénování (modře). Odhad chyby je průměrem chyb z pěti běhů. Krajní případ $k = N$ (každý příklad sám fold) se nazývá *leave-one-out*.

Maximum likelihood estimation

Mnoho modelů (lineární i logistická regrese, naivní Bayes) se učí *pravděpodobnostně*: model definuje pravděpodobnost dat při daných parametrech a my hledáme parametry, které pozorovaná data činí *nejvíce pravděpodobnými*.

Definice — Maximum likelihood estimation (MLE)

Mějme model s parametry $\mathbf{w}$, který datům přiřazuje pravděpodobnost (věrohodnost) $p(\text{data} \mid \mathbf{w})$. Při nezávislých příkladech se faktorizuje, $p(\text{data} \mid \mathbf{w}) = \prod_{i=1}^N p(t_i \mid \mathbf{x}_i; \mathbf{w})$. **Maximum likelihood estimation** (maximálně věrohodný odhad) parametrů je

$$\mathbf{w}_{\text{MLE}} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^N p(t_i \mid \mathbf{x}_i; \mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^N (-\log p(t_i \mid \mathbf{x}_i; \mathbf{w})).$$

Intuice. Logaritmus mění součin na součet a je rostoucí, takže maximum nezmění; navíc se lépe počítá a derivuje. Veličina $-\log p(t_i \mid \mathbf{x}_i; \mathbf{w})$ je *negative log-likelihood* (NLL, záporná logaritmická věrohodnost) a slouží přímo jako **ztrátová funkce**: minimalizovat NLL = maximalizovat věrohodnost. Pro normálně rozdělený šum dává MLE přesně *nejmenší čtverce* (sekce 4), pro klasifikaci *křížovou entropii* (sekce 5). Maximum likelihood tak propojuje „intuitivní“ ztrátové funkce s pravděpodobnostním modelem.

1.3 Evaluační metriky pro klasifikaci

Základem evaluace binárního klasifikátoru je **matice záměn** (confusion matrix): tabulka, kolik příkladů které skutečné třídy bylo predikováno do které třídy.

		predikce	
		neg	poz
skutečnost	poz	FN false negative	TP true positive
	neg	TN true negative	FP false positive

Matice záměn binárního klasifikátoru. Zeleně správné predikce (TP, TN), červeně chyby (FP = falešný poplach, FN = přehlédnutí). Z těchto čtyř čísel se počítají všechny níže uvedené metriky. Pozitivní třída je obvykle ta „zajímavá“ (nemoc, podvod, anomálie).

☰ Definice – Metriky binární klasifikace

Označme TP, TN, FP, FN počty příkladů v matici záměn. Pak:

- **accuracy** (úspěšnost): podíl správně klasifikovaných,

$$\text{acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

- **precision** (přesnost): podíl skutečně pozitivních mezi predikovanými pozitivními,

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

- **recall** (úplnost), též *sensitivity* a *true positive rate* (TPR): podíl zachycených ze skutečně pozitivních,

$$\text{recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

- **specificity** (specifická), též *true negative rate* (TNR): podíl zachycených ze skutečně negativních,

$$\text{specificity} = \text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}.$$

- **F1** (F1 skóre): harmonický průměr precision a recall,

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = \frac{2 \text{ TP}}{2 \text{ TP} + \text{FP} + \text{FN}}.$$

💡 **Intuice.** *Precision a recall* jsou v napětí: agresivnější klasifikátor (více predikcí „pozitivní“) zvýší recall, ale sníží precision, a naopak. F1 je vyvažuje jedním číslem; harmonický průměr (na rozdíl od aritmetického) je nízký, je-li *kterákoli* z obou nízká. Falešný poplach (FP) a přehlédnutí (FN) mívají *různě velké* náklady (přehlédnutý nádor vs. zbytečná biopsie), proto neexistuje jedna „správná“ metrika – volíme podle úlohy.

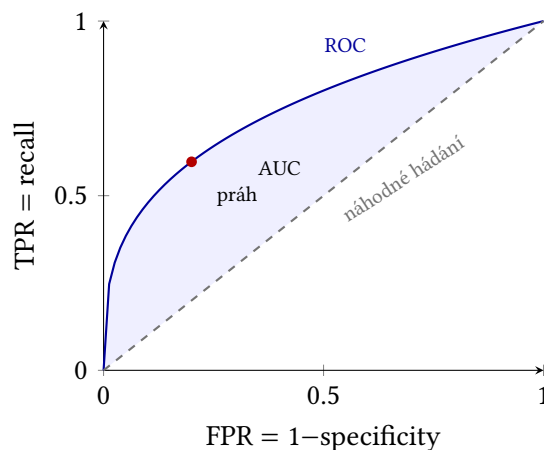
📌 **Poznámka.** [Pozor na nevyvážené třídy] U silně nevyvážených dat (např. 1 % podvodů) je *accuracy* *zavádějící*: triviální klasifikátor „vše negativní“ má accuracy 99 %, a přitom nezachytí jediný podvod. Tam dává smysl sledovat *precision*, *recall*, *F1* nebo *AUC*, ne *accuracy*.

ROC křivka a AUC

Klasifikátor obvykle vrací *skóre* (např. pravděpodobnost pozitivní třídy) a třídu přiřadí podle *prahu*. Posouváním prahu měníme kompromis mezi TPR a FPR.

☰ Definice – ROC křivka, AUC

ROC křivka (receiver operating characteristic) vynáší pro všechny prahy bod (FPR, TPR), kde $\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{specificity}$. Diagonála $\text{TPR} = \text{FPR}$ odpovídá náhodnému hádání, levý horní roh (0, 1) dokonalému klasifikátoru. **AUC** (area under curve) je plocha pod ROC křivkou; rovná se pravděpodobnosti, že náhodný pozitivní příklad dostane vyšší skóre než náhodný negativní. AUC = 1 je dokonalý, AUC = 0,5 náhodný klasifikátor.



ROC křivka (modře): každý bod odpovídá jednomu prahu rozhodování. Čím blíž levému hornímu rohu (0, 1), tím lepší. Čárkovaná diagonála je náhodné hádání; plocha pod křivkou (AUC) měří kvalitu nezávisle na volbě prahu.

Řešené příklady: matice záměn a metriky

Příklad – SZZ léto 2022 č. 11: Matice konfuze a specifita

(Reálná otázka SZZ: léto 2022, č. 11 – Evaluace binárního klasifikátoru)

Zadání. Testovací množina 1000 příkladů, 50 % negativních. Klasifikátor má sensitivity 60 % a accuracy 70 %. (1) Napište kompletní matici konfuze. (2) Vypočtěte specifitu.

Řešení. Je $P = 500$ pozitivních a $N = 500$ negativních. Sensitivity (recall) = $TP/P = 0,6 \Rightarrow TP = 300$, $FN = 200$. Accuracy = $(TP + TN)/1000 = 0,7 \Rightarrow TP + TN = 700$, odtud $TN = 400$ a $FP = N - TN = 100$.

	predikováno pozitivní	predikováno negativní
skutečně pozitivní	TP = 300	FN = 200
skutečně negativní	FP = 100	TN = 400

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{400}{500} = \boxed{0,8} = 80\%.$$

Příklad – SZZ podzim 2024 č. 30: Evaluace z accuracy a specifity

(Reálná otázka SZZ: podzim 2024, č. 30 – Evaluace binárního klasifikátoru)

Zadání. Testovací množina má 1000 příkladů; klasifikátor 400 označil jako pozitivní a 600 jako negativní, s accuracy 70 % a specifitu 80 %. Doplňte matici záměn a spočtěte precision, recall a F1.

Řešení. Z predikcí: $TP + FP = 400$, $TN + FN = 600$. Z accuracy $TP + TN = 0,7 \cdot 1000 = 700$. Specificity $\frac{TN}{TN+FP} = 0,8$. Soustavu vyřeší

$$TN = 400, \quad FN = 200, \quad TP = 300, \quad FP = 100$$

(kontrola: $TN/(TN + FP) = 400/500 = 0,8$ ✓, $(TP + TN)/1000 = 0,7$ ✓). Pak

$$\text{precision} = \frac{300}{400} = \boxed{0,75}, \quad \text{recall} = \frac{300}{500} = \boxed{0,6}, \quad F_1 = \frac{2 \cdot 0,75 \cdot 0,6}{0,75 + 0,6} = \boxed{0,6}.$$

Příklad – SZZ jaro 2023 č. 7: Matice záměn ze sensitivity a specificity

(Reálná otázka SZZ: jaro 2023, č. 7 – Evaluace binárního klasifikátoru)

Zadání. Klasifikátor má sensitivity 0,8 a specificity 0,9; testováno na 10 000 příkladech, z nichž 500 je pozitivních. (1) Spočítejte matici záměn. (2) Vstup je predikován jako pozitivní – jaká je pravděpodobnost, že je skutečně pozitivní? (3) Jaká je accuracy?

Řešení. (1) Pozitivních je 500, negativních 9500. Sensitivity (recall) = $TP/500 = 0,8 \Rightarrow TP = 400$, $FN = 100$. Specificity = $TN/9500 = 0,9 \Rightarrow TN = 8550$, $FP = 950$.

$$TP = 400, FN = 100, FP = 950, TN = 8550.$$

(2) Hledaná pravděpodobnost je právě *precision*: $precision = \frac{400}{400+950} = \frac{400}{1350} \approx \boxed{0,296}$. (Ač má klasifikátor vysokou sensitivity, kvůli vzácnosti pozitivní třídy je jen ~30 % predikovaných pozitivních skutečně pozitivních – ukázka, jak nevyváženost sráží precision.) (3) $acc = \frac{400+8550}{10000} = \boxed{0,895}$.

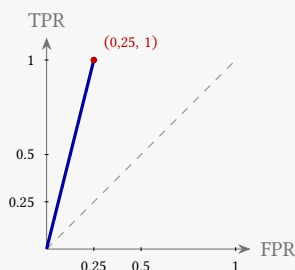
Příklad – SZZ léto 2023 č. 6: ROC a precision v jednom bodě

(Reálná otázka SZZ: léto 2023, č. 6 – Testování binárního klasifikátoru)

Zadání. Testovací sada má 100 pozitivních a 400 negativních příkladů. (a) ROC křivka prochází bodem $TPR = FPR = 20\%$; spočítejte precision v tomto bodě. (b) Jiný klasifikátor má konstantní precision = 50 %; znázorněte jeho ROC křivku.

Řešení. (a) $TPR = 0,2 \Rightarrow TP = 0,2 \cdot 100 = 20$; $FPR = 0,2 \Rightarrow FP = 0,2 \cdot 400 = 80$. Precision = $\frac{20}{20+80} = \boxed{0,2}$.

(b) Konstantní precision $\frac{TP}{TP+FP} = \frac{1}{2}$ znamená $FP = TP$ při každém prahu. Protože $TPR = \frac{TP}{100}$ a $FPR = \frac{FP}{400} = \frac{TP}{400}$, platí $FPR = \frac{1}{4}TPR$, tj. $TPR = 4 \cdot FPR$. ROC je tedy úsečka z (0, 0) do $(FPR, TPR) = (0,25, 1)$ (při $TPR = 1$ je $FP = TP = 100$, $FPR = 100/400 = 0,25$).



ROC křivka klasifikátoru s konstantní precision 50 % na datech 100 : 400: úsečka $TPR = 4 FPR$ od počátku do (0,25, 1). Leží nad diagonálou (lepší než náhoda).

Příklad – SZZ podzim 2025 č. 28: Dva nástroje na detekci anomálií

(Reálná otázka SZZ: podzim 2025, č. 28 – Evaluace binárního klasifikátoru)

Zadání. 10 000 instancí, z nichž 1 % (100) jsou anomálie (pozitivní), 9 900 normálních. Nástroj 1: accuracy 99,5 %, zachytí 90 % anomálií. Nástroj 2: zachytí 81 % anomálií, 10 % nahlášených je falešně pozitivních. Spočítejte matice záměn a metriky a rozhodněte, který nástroj zvolit.

Řešení.

Nástroj 1. Recall 0,9 $\Rightarrow TP = 90$, $FN = 10$. Správně klasifikováno $0,995 \cdot 10000 = 9950$, z toho $TN = 9950 - 90 = 9860$, takže $FP = 9900 - 9860 = 40$.

$$precision = \frac{90}{130} \approx 0,69, \quad recall = 0,9, \quad F_1 \approx 0,78, \quad acc = 0,995.$$

Nástroj 2. Recall 0,81 $\Rightarrow TP = 81$, $FN = 19$. „10 % nahlášených falešně pozitivních“ znamená

precision = 0,9 = $\frac{TP}{TP+FP}$, odkud $FP = \frac{81}{9} = 9$, $TN = 9900 - 9 = 9891$.

precision = 0,9, recall = 0,81, $F_1 \approx 0,85$, acc = 0,9972.

Závěr. Nástroj 2 je lepší ve všech metrikách kromě recall (0,81 vs. 0,9). Hlásí mnohem méně falešných poplachů (FP 9 vs. 40), takže neobtěžuje správce. Naopak nástroj 1 odhalí víc anomálií; jsou-li kritické, může se i přes nižší precision vyplatit. Jednoznačná odpověď neexistuje, rozhoduje cena FP vs. FN. V původním PDF jsou u nástroje 2 drobné nekonzistence (zaměněný vzorec precision, „80 ze 100“ místo 81 a hodnota FP označená jako FN); zde uvádíme spočítanou konzistentní variantu, viz též poznámka ve fragmentu v příklady/.

Příklad — SZZ podzim 2023 č. 31: Metriky a detekce podvodů

(Reálná otázka SZZ: podzim 2023, č. 31 – Evaluační metriky)

Zadání (zkráceně). (1) Definujte accuracy, recall, precision, F1. (2) Detekce podvodných plateb, kde z milionů jen stovky jsou podvody – jaké metriky (ne)použít? (3) Systém A: recall 0,99, precision 0,2; systém B: recall 0,9, precision 0,95. Co znamenají a který zvolit?

Řešení. (1) Viz definice výše. (2) Data jsou silně nevyvážená, proto *accuracy je nevhodná* (triviální „vše legitimní“ má accuracy $\approx 100\%$). Vhodné jsou *precision a recall* (případně F1 nebo AUC), které se soustředí na vzácnou pozitivní třídu. (3) Systém A odhalí 99 % podvodů, ale jen 20 % jeho hlášení jsou skutečné podvody (oproti B zhruba 5× víc hlášení k prošetření: $0,99/0,2 \approx 4,95$ vs. $0,9/0,95 \approx 0,95$ na jeden podvod). Systém B odhalí 90 % podvodů a 95 % hlášení sedí. Pokud nás trápí náklady na prošetřování a obtěžování zákazníků, volíme B; pokud je prioritní zachytit i poslední podvod, volíme A. Jednoznačná odpověď neexistuje, rozhoduje argumentace (cena FP vs. FN).

1.4 Evaluační metriky pro regresi

U regrese měříme *velikost chyby* predikce $\hat{t}_i = y(\mathbf{x}_i)$ vůči cíli t_i .

Definice — Metriky regrese

$$MSE = \frac{1}{N} \sum_i (t_i - \hat{t}_i)^2, \quad RMSE = \sqrt{MSE}, \quad MAE = \frac{1}{N} \sum_i |t_i - \hat{t}_i|.$$

RMSE je ve stejných jednotkách jako cíl; oproti MAE *víc penalizuje velké chyby* (kvadrát). Koefficient determinace $R^2 = 1 - \frac{\sum_i (t_i - \hat{t}_i)^2}{\sum_i (t_i - \bar{t})^2}$ udává podíl rozptylu cíle vysvětlený modelem ($R^2 = 1$ dokonale, $R^2 = 0$ jako predikce průměrem).

1.5 Naivní Bayesův klasifikátor

i Poznámka. [Nad rámec požadavků] Naivní Bayes není v požadavcích jmenován, ale je to standardní pravděpodobnostní klasifikátor probíraný v NPFL129 (vedle k -NN) a objevuje se v reálné SZZ otázce; uvádíme ho proto jako řešený příklad klasifikace.

Definice — Naivní Bayesův klasifikátor

Pro vstup s příznaky $x_1, \dots, x_D$ přiřadíme třídu c podle *aposteriorní* pravděpodobnosti (Bayesova věta):

$$P(c \mid x_1, \dots, x_D) = \frac{P(c) P(x_1, \dots, x_D \mid c)}{P(x_1, \dots, x_D)} \propto P(c) P(x_1, \dots, x_D \mid c).$$

Naivní předpoklad je *podmíněná nezávislost* příznaků při dané třídě, takže sdruženou věrohodnost rozložíme na součin:

$$\hat{c} = \underset{c}{\operatorname{argmax}} P(c) \prod_{j=1}^D P(x_j | c).$$

Pravděpodobnosti $P(c)$ (apriorní třídy) a $P(x_j | c)$ odhadneme z trénovacích dat relativními četnostmi (MLE). Jmenovatel $P(x_1, \dots, x_D)$ je stejný pro všechny třídy, takže pro výběr maxima stačí čítatel.

Intuice. „Naivnost“ je v předpokladu, že příznaky jsou při dané třídě *nezávislé* – to skoro nikdy přesně neplatí (např. déšť a oblíbená přednáška spolu mohou korelovat), ale klasifikátor i tak často dobře funguje a potřebuje málo dat. Bez tohoto předpokladu bychom museli odhadovat celé sdružené rozdělení $P(x_1, \dots, x_D | c)$, což je exponenciálně mnoho parametrů.

Příklad – SZZ jaro 2023 č. 16: Naivní Bayes – půjde student do školy?

(Reálná otázka SZZ: jaro 2023, č. 16 – Naivní Bayesovský klasifikátor)

Zadání. Z 13 pozorování (atributy *Vstanu*, *Oblíbená přednáška?*, *Počasí*; cíl *Jdu do školy*) určete naivním Bayesem, zda student půjde do školy, jestliže *vstane pozdě, nemá oblíbenou přednášku a prší*.

Den	Vstanu	Oblíbená přednáška?	Počasí	Jdu do školy
1	Brzo	Ne	Slunce	Ne
2	Normálně	Ne	Déšť	Ne
3	Brzo	Ne	Slunce	Ano
4	Pozdě	Ne	Slunce	Ano
5	Pozdě	Ano	Slunce	Ano
6	Pozdě	Ano	Déšť	Ne
7	Brzo	Ano	Déšť	Ano
8	Normálně	Ne	Slunce	Ne
9	Normálně	Ano	Slunce	Ano
10	Pozdě	Ano	Slunce	Ano
11	Normálně	Ano	Déšť	Ano
12	Brzo	Ne	Déšť	Ano
13	Brzo	Ano	Slunce	Ano

Řešení. Z 13 dnů jde do školy 9× (Ano), nejde 4× (Ne): $P(\text{Ano}) = \frac{9}{13}$, $P(\text{Ne}) = \frac{4}{13}$. Spočteme podmíněné četnosti potřebných hodnot:

$$P(\text{Pozdě} | \text{Ano}) = \frac{3}{9}, P(\text{obl=Ne} | \text{Ano}) = \frac{3}{9}, P(\text{Děšť} | \text{Ano}) = \frac{3}{9};$$

$$P(\text{Pozdě} | \text{Ne}) = \frac{1}{4}, P(\text{obl=Ne} | \text{Ne}) = \frac{3}{4}, P(\text{Děšť} | \text{Ne}) = \frac{2}{4}.$$

Skóre (čítatel) pro obě třídy:

$$\text{Ano} : \frac{9}{13} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{39} \approx 0,0256, \quad \text{Ne} : \frac{4}{13} \cdot \frac{1}{4} \cdot \frac{3}{4} \cdot \frac{2}{4} = \frac{3}{104} \approx 0,0288.$$

Větší je skóre třídy *Ne*, takže naivní Bayes predikuje, že student **do školy nepůjde** (po normalizaci $P(\text{Ne} | \text{data}) \approx 53\%$, $P(\text{Ano} | \text{data}) \approx 47\%$ – těsné). Zajímavé je, že přestože „chodí do školy“ je apriorně mnohem častější, kombinace tří příznaků (pozdě, bez oblíbené přednášky, déšť) převáží ve prospěch „nejde“.

1.6 Praktický postup: od dat k modelu

Úloha nemusí mířit na jeden algoritmus, ale na *celý postup*: rozpoznat typ úlohy, zvolit model, předzpracovat data, vybrat metriku. Následující reálná otázka SZZ takový postup procvičuje – nejde o samostatné

téma požadavků, ale o syntézu pojmů této sekce (a modelů z dalších sekcí).

Příklad — SZZ podzim 2025 č. 29: Predikce z tabulárních dat

(Reálná otázka SZZ: podzim 2025, č. 29 – Predikce pomocí tabulárních dat)

Zadání (zkráceně). Zemědělský dataset (pH, srážky, teplota, typ hnojiva, odrůda, velikost farmy, zavlažování, nadm. výška) má předpovídat výnos (t/ha). (1) Typ úlohy? (2) Vhodný model a proč? (3) Předzpracování a zacházení s typy příznaků. (4) Evaluační metriky?

Řešení. (1) *Regrese* s učitelem – cíl (výnos) je spojitě číslo a máme trénovací data s cílovými hodnotami. (2) Pro tabulární data s mixem číselných i kategorických příznaků jsou typicky nejlepší *stromové ansámby* – *random forest* nebo *gradient boosting* (XGBoost/LightGBM): zvládnou nelinearity i interakce, nevyžadují škálování (viz sekce 8). Lineární regrese nebo MLP fungují s pečlivějším předzpracováním. (3) *Kategorické* příznaky (typ hnojiva, odrůda, zavlažování) zakódovat *one-hot*; *číselné* (pH, srážky, ...) zkontrolovat odlehlé hodnoty a u modelů citlivých na škálu *standardizovat*; chybějící hodnoty doplnit (průměr/medián, resp. modus). (4) MAE, (R)MSE, případně R^2 .

2 Overfitting a regularizace

Sekce 1 ukázala, *jak* chybu modelu poctivě měřit. Tato sekce vysvětluje, *proč* se trénovací a generalizační chyba rozcházejí: model může být na úlohu příliš slabý (*underfitting*, podučení) nebo příliš silný a zapamatoval si i šum (*overfitting*, přeučení). Zavedeme *kapacitu modelu* jako škálu mezi těmito extrémy, ukážeme, jak ji ovládat *regularizací* (L^2 , L^1) a *early stopping*, a na závěr rozebereme *prokletí dimenzionality* – proč je vysoká dimenze příznaků sama o sobě zdrojem overfittingu.

Poznámka. [Vztah k S1] Okruh S1 zmiňuje overfitting jen v souvislosti s *Ockhamovou břitvou* (preferuj nejjednodušší hypotézu konzistentní s daty). Zde to rozvádíme do hloubky, kterou S1 výslovně přenechává S3: kapacita modelu, generalizační chyba, regularizační techniky a prokletí dimenzionality. Konkrétní aplikace L^2 na lineární regresi je v sekci 4.

2.1 Generalizační chyba, underfitting a overfitting

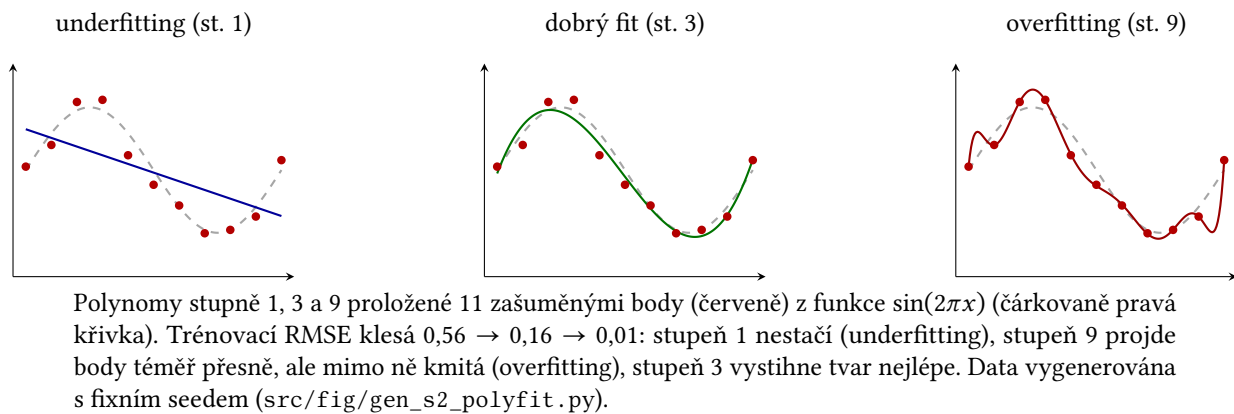
Definice — Trénovací a generalizační chyba

Trénovací chyba je chyba modelu na trénovací množině, na které se učil. **Generalizační chyba** (chyba zobecnění, též *testovací chyba*) je *očekávaná* chyba na nových, neviděných příkladech ze stejného rozdělení generujícího data; odhadujeme ji chybou na nezávislé testovací množině (sekce 1). Cílem učení je malá generalizační chyba, ne malá trénovací chyba.

Definice — Underfitting a overfitting

Underfitting (podučení) nastane, když model nedokáže zachytit ani zákonitost v trénovacích datech: trénovací i generalizační chyba jsou velké. **Overfitting** (přeučení) nastane, když se model „naučí“ i náhodný šum trénovací množiny: trénovací chyba je malá, ale generalizační chyba velká (mezera mezi nimi je velká).

Intuice. Klasická ukázka je prokládání zašuměných bodů polynomem (obrázek níže). *Přímka* (stupeň 1) je na sinusovku příliš tuhá – nastává underfitting. Polynom *vysokého stupně* (stupeň 9, deset koeficientů na 11 bodů) projde body téměř přesně (trénovací chyba skoro nulová), ale mezi body divoce kmitá: zapamatoval si šum, ne tvar (overfitting). *Střední stupeň* (zde 3) zachytí tvar a šum ignoruje – nejlépe generalizuje. Overfittovaný model je jako student, který nabífováním odpoví na cvičné otázky doslova, ale u nové otázky selže.

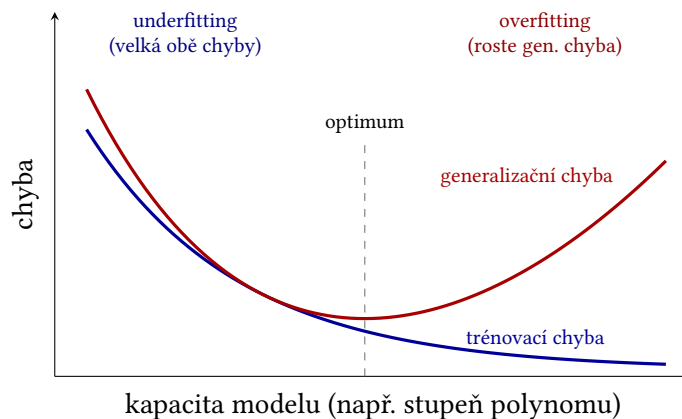


2.2 Kapacita modelu

Definice – Kapacita modelu

Kapacita modelu je neformálně jeho schopnost vyjádřit *složité* funkce – jak bohatá je třída funkcí, mezi nimiž učení vybírá. Rozlišujeme *reprezentační kapacitu* (jaké funkce model vůbec umí vyjádřit, např. polynomy do stupně M) a *efektivní kapacitu* (kterých z nich učící algoritmus reálně dosáhne, případně omezenou regularizací). U polynomiální regrese roste kapacita se stupněm M ; obecně s počtem volných parametrů.

Intuice. Kapacita je „regulátor“ mezi underfittingem a overfittingem. *Malá* kapacita $\Rightarrow$ underfitting (model je příliš tuhý). *Velká* kapacita $\Rightarrow$ riziko overfittingu (model je tak ohebný, že se přizpůsobí i šumu). Optimum je uprostřed. Kapacitu lze měnit nejen volbou modelu (stupeň polynomu, hloubka stromu, počet sousedů k), ale i *množstvím dat*: víc trénovacích příkladů relativní kapacitu snižuje (model už nemá dost volnosti zapamatovat si každý bod), takže více dat samo o sobě overfitting tlumí.



Schematická závislost chyby na kapacitě modelu. Trénovací chyba (modře) s rostoucí kapacitou monotónně klesá. Generalizační chyba (červeně) má tvar U : nejdřív klesá (mizí underfitting), pak roste (model overfittuje). Optimální kapacita je v minimu červené křivky; vlevo od něj je oblast underfittingu, vpravo overfittingu. Mezera mezi křivkami je míra overfittingu.

2.3 Regularizace

Definice – Regularizace

Regularizace je jakákoli úprava učicího algoritmu, jejímž záměrem je *snížit generalizační chybu* (ne nutně trénovací). Typicky omezuje efektivní kapacitu modelu, takže ten musí dát přednost „jednodušším“ řešením a hůře se přizpůsobí šumu.

Intuice. Omezení kapacity zvenčí (menší stupeň polynomu) je hrubé – skáče po celých číslech a rovnou škrtá celé příznaky. Regularizace dovoluje *plynule* škálovat, jak moc model „smí“ přizpůsobit váhy datům: necháme bohatou třídu funkcí, ale přidáme do ztrátové funkce *trest za složitost*. Nejčastěji trestáme *velikost vah* – malé váhy znamenají hladší, méně rozkmitanou funkci.

L^2 a L^1 regularizace

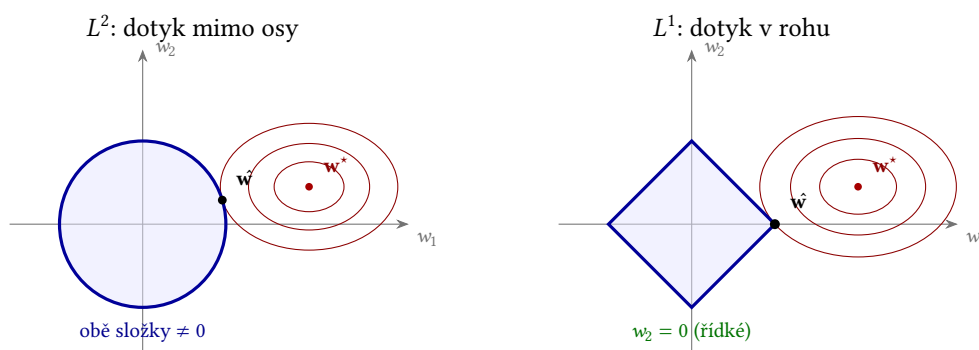
Definice – L^2 a L^1 regularizace

K původní ztrátové funkci $E_0(\mathbf{w})$ (např. součtu čtverců) přidáme penalizační člen úměrný normě vah:

$$E_{L^2}(\mathbf{w}) = E_0(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 = E_0(\mathbf{w}) + \frac{\lambda}{2} \sum_j w_j^2, \quad E_{L^1}(\mathbf{w}) = E_0(\mathbf{w}) + \lambda \|\mathbf{w}\|_1 = E_0(\mathbf{w}) + \lambda \sum_j |w_j|.$$

L^2 **regularizace** (též *weight decay*, hřebenová / ridge regrese) trestá kvadrát eukleidovské normy; L^1 **regularizace** (lasso) trestá součet absolutních hodnot. *Regularizační síla* $\lambda \geq 0$ je hyperparametr: $\lambda = 0$ je bez regularizace, velké λ tlačí váhy k nule. Penalizace se obvykle *nevztahuje na bias* b (přes bias nelze overfittovat).

Intuice. [L^2 vyhlazuje, L^1 řídne] Obě penalizace táhnou váhy k nule, ale jinak. L^2 zmenšuje *všechny* váhy úměrně jejich velikosti – velké srazí hodně, malé téměř nechá; vede k mnoha *malým, ale nenulovým* vahám, tedy k hladkému modelu (gradient lineární predikce vůči vstupu jsou přímo váhy, $\nabla_{\mathbf{x}} \mathbf{w}^T \mathbf{x} = \mathbf{w}$, takže malé váhy = malé změny výstupu). L^1 naopak tlačí část vah *přesně na nulu* – provádí tak *výběr příznaků*: výsledný model je *řídý* (sparse), používá jen podmnožinu příznaků. Proč rozdíl, ukazuje geometrie níže.



Geometrie regularizace. Červené elipsy jsou *kontury* (vrstevnice) původní ztráty se středem v nepenalizovaném optimu $\mathbf{w}_{\text{MLE}}^*$ (maximum likelihood estimation, odhad maximální věrohodnosti). Regularizace nutí řešení $\hat{\mathbf{w}}$ dovnitř modré omezující oblasti; optimum leží tam, kde se nejmenší kontura oblasti *dotkne*. Vlevo: L^2 oblast je kruh (hladký), dotyk nastane v obecném bodě s oběma složkami nenulovými. Vpravo: L^1 oblast je kosočtverec s *rohy* na osách; elipsa se ho typicky dotkne v rohu, kde $w_2 = 0$ – proto L^1 vynuluje váhy a dává řídý řešení.

Poznámka. λ a stupeň polynomu M jsou *hyperparametry* – učicí algoritmus je sám nenastaví, určujeme je zvenčí. Konkrétní dopad L^2 na uzavřené řešení lineární regrese (přechod $\mathbf{X}^T \mathbf{X} \rightarrow \mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}$, vždy invertibilní) a na krok SGD rozebíráme u modelu, kde se to počítá – v sekci 4.

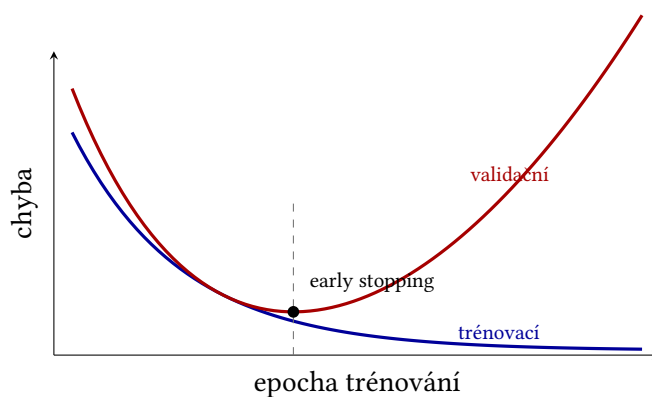
Early stopping

U iterativně trénovaných modelů (gradientní sestup) lze kapacitu omezit i *časem*: čím déle trénujeme, tím víc se model přizpůsobuje datům včetně šumu.

Definice – Early stopping

Early stopping (včasné zastavení) sleduje během trénování chybu na *validační* množině a trénink ukončí (resp. použije uložený model) ve chvíli, kdy validační chyba přestane klesat a začne růst, i když trénovací chyba dál klesá. Počet kroků (epoch) je tak vlastně hyperparametr laděný na validačních datech.

Intuice. Trénovací chyba u dostatečně kapacitního modelu klesá monotónně až k nule, ale validační chyba má opět tvar U : nejdřív klesá (model se učí zákonitost), pak roste (začíná se učit šum). Bod obratu je optimální okamžik zastavení – model v něm má nejlepší generalizaci. Early stopping je oblíbené, protože je levné (nevyžaduje volbu λ ani opakované trénování) a funguje jako forma regularizace: omezuje, jak daleko se váhy od počátku stačí pohnout. (Příbuznou technikou je *dropout* u neuronových sítí, mimo rozsah tohoto okruhu.)



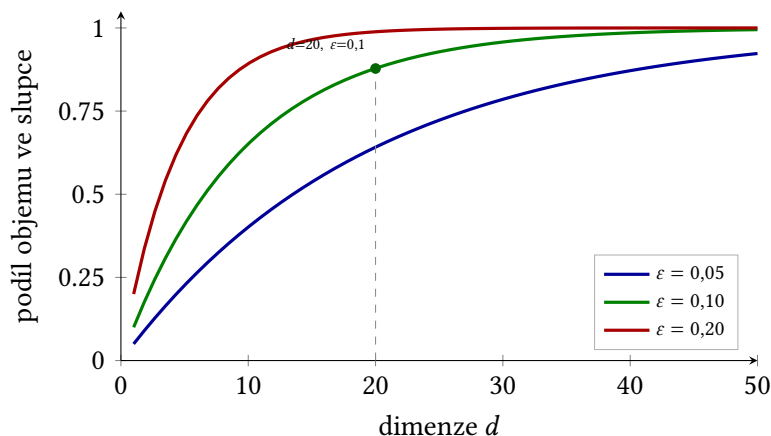
Early stopping. Trénovací chyba (modře) klesá k nule, validační chyba (červeně) nejdřív klesá, pak roste – model začíná overfittovat. V minimu validační chyby (černá tečka) trénink zastavíme; další epochy už generalizaci jen zhoršují.

2.4 Prokletí dimenzionality

Definice – Prokletí dimenzionality

Prokletí dimenzionality (curse of dimensionality) je souhrn jevů, kvůli nimž se s rostoucím počtem příznaků D (dimenzí vstupu) učení dramaticky ztěžuje: objem prostoru roste exponenciálně, takže fixní množina dat je ve vysoké dimenzi *řídka* (vzdálené, prázdné okolí každého bodu), a vzdálenosti mezi body se vzájemně *připodobňují*. K pokrytí prostoru se stejnou hustotou bodů je třeba *exponenciálně* mnoho příkladů v D .

Intuice. Tři tváře téhož jevu. (1) *Hustota*: 10 bodů rovnoměrně pokryje úsečku $[0, 1]$, ale $[0, 1]^{10}$ jich vyžaduje 10^{10} , aby byla mřížka stejně hustá – jinak jsou data ve vysoké dimenzi téměř všude prázdná. (2) *Koncentrace objemu*: skoro celý objem vysokodimenzionální krychle (či koule) leží těsně u povrchu, takže „typický“ bod je u kraje, ne uvnitř. (3) *Koncentrace vzdáleností*: poměr nejbližšího a nejvzdálenějšího souseda se blíží jedné, takže pojem „nejbližší soused“ ztrácí smysl – to přímo dopadá na k -NN (sekce 3). Důsledek pro učení: víc příznaků zvyšuje kapacitu, ale i potřebu dat; nadbytečné příznaky proto raději odstraňujeme (výběr příznaků, L^1 , PCA).



Koncentrace objemu: podíl objemu jednotkové krychle $[0, 1]^d$, který leží ve vnější slupce tloušťky ε (mimo vnitřní krychli $[\varepsilon, 1 - \varepsilon]^d$), je $1 - (1 - 2\varepsilon)^d$; zde kreslíme zjednodušeně podíl mimo zmenšenou krychli $[0, 1 - \varepsilon]^d$, tj. $1 - (1 - \varepsilon)^d$. Pro $\varepsilon = 0,1$ a $d = 20$ vychází $1 - 0,9^{20} \approx 0,878$, tj. skoro 88 % objemu je v tenké slupce u povrchu; pro $d = 50$ je to $1 - 0,9^{50} \approx 0,995$. Funkce je analytická, bez šumu.

Příklad – Ilustrace: L^1 vynuluje váhu, L^2 ne

Zadání. Mějme model se dvěma vahami, kde nepenalizované optimum ztráty je $\mathbf{w}^* = (2, 0,5)$ a kontury ztráty jsou kružnice $\|\mathbf{w} - \mathbf{w}^*\|^2 = \text{konst.}$ (izotropní). Omezíme řešení podmínkou $\|\mathbf{w}\| \leq 1$ (L^2), resp. $\|\mathbf{w}\|_1 \leq 1$ (L^1). Které řešení je řidké?

Řešení. Při *kruhových* konturách je penalizované řešení průmět $\mathbf{w}^*$ na omezující oblast (nejbližší bod oblasti k $\mathbf{w}^*$). L^2 (kruh poloměru 1): nejbližší bod je ve směru $\mathbf{w}^*$, tj. $\hat{\mathbf{w}} = \mathbf{w}^* / \|\mathbf{w}^*\| = (2, 0,5) / \sqrt{4,25} \approx (0,970, 0,243)$ – obě složky nenulové. L^1 (kosoúhelník $|w_1| + |w_2| \leq 1$): nejbližší bod ke vzdálenému $\mathbf{w}^* = (2, 0,5)$ je roh $(1, 0)$ (ověření: vzdálenost $\mathbf{w}^*$ od $(1, 0)$ je $\sqrt{1 + 0,25} \approx 1,118$ a každý jiný bod kosoúhelníku je dál), takže $\hat{\mathbf{w}} = (1, 0)$ – druhá váha je přesně nula. L^1 tedy příznak 2 vyřadí (řidké řešení), L^2 ho jen zmenší. To je kvantitativní ukázka geometrie z obrázku výše.

Příklad – SZZ podzim 2022 č. 18: L^2 regularizace, porovnání modelů

(Reálná otázka SZZ: podzim 2022, č. 18 – Přeučení a regularizace)

Zadání. Trénujeme lineární model pro regresi. (1) Definujte střední čtvercovou chybu (MSE) a upravenou chybovou funkci pro L^2 regularizaci. (2) Dva lineární modely pro data se čtyřmi atributy,

$$M_1(\mathbf{x}) = x_1 - 2x_2 + 4x_3 - x_4 + 3, \quad M_2(\mathbf{x}) = x_2 - 5x_3 + x_4 - 1,$$

mají na daných datech stejnou MSE. Který má menší hodnotu L^2 -regularizované chyby? (3) Uveďte alespoň jeden další způsob, jak zlepšit generalizaci.

Řešení. (1) Střední čtvercová chyba je

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2,$$

kde $\hat{y}_i$ je predikce a y_i skutečná hodnota. Při L^2 regularizaci přidáme k MSE penalizaci za velikost

vah,

$$\text{MSE} + \lambda \sum_j w_j^2, \quad \lambda > 0,$$

přičemž absolutní člen (bias) se *neregularizuje*.

(2) Regularizační člen závisí jen na vahách (bez biasu). Pro M_1 jsou váhy $(1, -2, 4, -1)$, takže $\sum_j w_j^2 = 1 + 4 + 16 + 1 = 22$; pro M_2 jsou váhy $(0, 1, -5, 1)$, takže $\sum_j w_j^2 = 0 + 1 + 25 + 1 = 27$. Při stejné MSE má menší regularizovanou chybu model M_1 ($22 < 27$).

Pozn. Kdyby se regularizoval i bias, vyšlo by M_1 : $22 + 3^2 = 31$ a M_2 : $27 + (-1)^2 = 28$, a vyhrál by M_2 . Právě proto je podstatné, že se bias neregularizuje.

(3) Generalizaci lze zlepšit i více trénovacími daty, výběrem/redukci příznaků, předčasným zastavením (early stopping), dropoutem (u neuronových sítí), kombinováním modelů (ensembling), L^1 regularizací nebo laděním hyperparametrů (např. λ) křížovou validací.

3 Učení založené na příkladech: k nejbližších sousedů

Předchozí dvě sekce zavedly evaluaci a overfitting obecně; teď přichází *první konkrétní model*. **k nejbližších sousedů** (k -NN, k -nearest neighbors) je nejjednodušší příklad *učení založeného na příkladech* (instance-based learning): predikci nového vstupu složí přímo z *uložených trénovacích příkladů, které jsou mu nejpodobnější*. Žádný globální model dat se nestaví.

i Poznámka. [Vztah k S1] Okruh S1 představuje k -NN jako *typický neparametrický model* (drží data místo vah) na úrovni přehledu. Zde to nezdujujeme – princip jen krátce zrekapitulujeme a rovnou jdeme do hloubky, kterou S1 přenechává S3: váhování sousedů, volba metriky a normalizace, volba k křížovou validací a výpočetní cena predikce.

3.1 Princip: líný, neparametrický model

Definice – k nejbližších sousedů

Mějme trénovací množinu $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$ a *metriku* (funkci vzdálenosti) $d(\cdot, \cdot)$ na vstupním prostoru. Pro nový (dotazovaný) vstup $\mathbf{x}_q$:

1. spočti vzdálenosti $d(\mathbf{x}_q, \mathbf{x}_i)$ ke *všem* trénovacím příkladům a vyber k příkladů s nejmenší vzdáleností – jeho *k nejbližších sousedů*;
2. predikci $y(\mathbf{x}_q)$ slož z cílů těchto sousedů: u **klasifikace** většinovým hlasováním, u **regrese** (váženým) průměrem (viz níže).

💡 Intuice. k -NN je *líný* (lazy) model: **trénování je triviální – jen se uloží celá trénovací množina**, žádné parametry se neoptimalizují. Veškerá práce se odkládá na okamžik predikce (proto „líný“). Je to zároveň model *neparametrický*: nepředpokládá pevný tvar rozhodovací funkce s daným počtem vah; „složitost“ hypotézy roste s počtem dat (čím víc příkladů, tím jemnější hranice umí popsat). Předpoklad za metodou je prostý: *podobné vstupy mají podobné cíle*, tedy lokální hladkost cílové funkce.

Algoritmus – k -NN: trénování a predikce

Trénování($\{(\mathbf{x}_i, t_i)\}$):

ulož celou trénovací množinu

▷ nic víc – líné učení

Predikce($\mathbf{x}_q, k, d$):

pro každý trénovací příklad $i = 1, \dots, N$:

spočti $d_i \leftarrow d(\mathbf{x}_q, \mathbf{x}_i)$
 $S \leftarrow$ indexy k nejmenších d_i
klasifikace: vrať nejčastější třídu mezi $\{t_i : i \in S\}$
regrese: vrať (vážený) průměr $\{t_i : i \in S\}$

▷ vzdálenost k dotazu
 ▷ k nejbližších sousedů
 ▷ remízy libovolně

3.2 Klasifikace: většinové hlasování

U klasifikace do K tříd se nový vstup zařadí do **nejčastější třídy** mezi jeho k sousedy (*majority voting*). Remízy (např. 2 : 2 při $k = 4$) se řeší libovolně – typicky volbou liché k u dvou tříd, náhodně, nebo podle bližšího souseda.

Intuice. Hlasování lze chápat jako lokální odhad pravděpodobností tříd: podíl sousedů třídy c je odhad $P(c | \mathbf{x}_q)$ v okolí dotazu, a vybíráme nejpravděpodobnější třídu. Při $k = 1$ dostaneme přímo třídu nejbližšího příkladu; rozhodovací hranice je pak *Voroného diagram* trénovacích bodů (každý bod „vlastní“ okolí složené z míst, kde je nejbližší).

Poznámka. [Vážené hlasování] Sousedy lze *vážít* podle blízkosti: bližší soused má větší hlas. Běžné volby vah w_i jsou *uniformní* ($w_i = 1$, prostá většina), *inverzní* ($w_i = 1/d_i$, váha úměrná převrácené vzdálenosti) nebo *softmax* záporných vzdáleností. Predikovaná třída je pak $\operatorname{argmax}_c \sum_{i \in S, t_i=c} w_i$ (váhovaný součet hlasů). Vážení zjemňuje vliv vzdálenějších, méně relevantních sousedů.

3.3 Regrese: (vážený) průměr sousedů

U regrese je cíl reálné číslo a predikcí je **průměr cílů** k sousedů. S vahami w_i (uniformní $\Rightarrow$ prostý průměr):

$$y(\mathbf{x}_q) = \sum_{i \in S} \frac{w_i}{\sum_{j \in S} w_j} t_i.$$

Pro klasifikaci téhož vzorce: cíle t_i zakódujeme *one-hot* (vektor s jedničkou na pozici třídy) a vážené zprůměrujeme, čímž dostaneme rozdělení pravděpodobností tříd $\mathbf{t} = \sum_{i \in S} \frac{w_i}{\sum_j w_j} \mathbf{t}_i$; predikovaná třída je $\operatorname{argmax}_c t_c$. Uniformní váhy a argmax pak dají právě většinové hlasování.

3.4 Vzdálenostní metriky a normalizace příznaků

Výsledek k -NN zcela stojí na tom, *jak měříme blízkost*. Pro $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ je standardní volbou **Minkowského** (L^p) vzdálenost

$$d_p(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^D |x_i - y_i|^p \right)^{1/p},$$

jejímiž speciálními případy jsou *Manhattanská* ($p = 1$, součet absolutních rozdílů), *Eukleidovská* ($p = 2$, „vzdušnou čarou“) a v limitě *Čebyševova* ($p \rightarrow \infty$, $\max_i |x_i - y_i|$). Nejběžnější je Euklid ($p = 2$).

Intuice. [Proč normalizovat příznaky] Metriky L^p sčítají rozdíly přes všechny příznaky, takže příznak s *větším rozsahem hodnot* (např. příjem 0–100 000 vs. věk 0–100) vzdálenost zcela ovládne a ostatní příznaky se přehlasují. Před k -NN proto příznaky **normalizujeme** na srovnatelný rozsah: buď *standardizací* (každý příznak na nulový průměr a jednotkový rozptyl, $x' = (x - \mu)/\sigma$), nebo *min-max škálováním* na $[0, 1]$. Bez normalizace by „nejbližší soused“ znamenal pouze „nejbližší v jednom dominantním příznaku“.

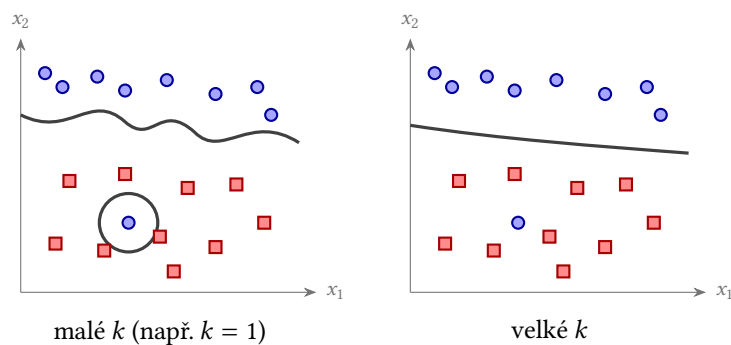
Poznámka. [Prokletí dimenzionality] Ve vysoké dimenzi přestává pojem „blízký soused“ dávat smysl: vzdálenosti k nejbližšímu a nejvzdálenějšímu bodu se k sobě přibližují, takže všechny příklady jsou zhruba stejně daleko a hlasování ztrácí výpovědní hodnotu. k -NN je proto na vysokodimen-

ziónální data citlivý (*prokletí dimenzionality*, podrobně sekce 2); pomáhá redukce dimenze (PCA, sekce 10) nebo výběr příznaků.

3.5 Volba k : vliv na hranici, overfitting, křížová validace

Počet sousedů k je **hyperparametr** – neučí se z dat, volíme ho zvenčí. Řídí kompromis mezi *overfittingem* (přeučení, velký rozptyl) a *underfittingem* (podučení, velké vychýlení); viz sekce 2:

- **malé k** (v krajnosti $k = 1$): predikce závisí na jednom či pár nejbližších bodech, je velmi citlivá na šum a odlehlé příklady. Rozhodovací hranice je *členitá*, „roztřepená“, s ostrůvky kolem jednotlivých bodů – model má *velký rozptyl* a *overfittuje* (trénovací chyba u $k = 1$ je nulová, ale generalizace špatná);
- **velké k** : hlasuje hodně sousedů, predikce je vyhlazená a odolná vůči šumu, ale hranice je *hladká až příliš* a ztrácí lokální detail – model má *velké vychýlení* a *underfittuje*. V krajnosti $k = N$ vrací vždy globálně nejčastější třídu (resp. celkový průměr) bez ohledu na vstup.



Vliv k na rozhodovací hranici k -NN (dvě třídy: ● modré, ■ červené). Při *malém k* (vlevo) je hranice členitá a kolem osamělého šumového bodu vznikne ostrůvek – model *overfittuje*. Při *velkém k* (vpravo) je hranice hladká, šumový bod je sousedy přehlasován; přílišné k však ztrácí lokální detail (*underfitting*). Optimální k leží mezi.

Intuice. [Jak vybrat k] Vhodné k nehádáme – vybereme ho **křížovou validací** (sekce 1): pro kandidátní hodnoty $k = 1, 3, 5, \dots$ změříme k -násobnou křížovou validací chybu a zvolíme k s nejlepším *validačním skóre*. Trénovací chyba k volbě *nestačí* (u $k = 1$ je vždy nulová, přesto *overfittuje*). U dvou tříd se volí *liché k* , aby se předešlo remízám v hlasování. Závislost validační chyby na k má typický tvar písmene U: zprvu klesá (mizí *overfitting*), pak roste (*underfitting*).

3.6 Výpočetní cena predikce

Líné učení má svou cenu na druhé straně. **Naivní predikce je drahá**: pro každý dotaz se spočítá vzdálenost ke *všem* N trénovacím bodům (D -rozměrným), tedy $\mathcal{O}(ND)$ na výpočet vzdáleností plus výběr k nejmenších ($\mathcal{O}(N \log k)$ haldou, nebo $\mathcal{O}(N \log N)$ tříděním). Trénování je $\mathcal{O}(1)$ (jen uložení), ale predikce roste lineárně s velikostí trénovací množiny – opak parametrických modelů, kde je drahé trénování a levná predikce.

Poznámka. [Zrychlení vyhledávání] Predikci lze urychlit prostorovými datovými strukturami, které předpočítají organizaci bodů: **k -d stromy** (dotaz v průměru v logaritmickém čase pro nízkou dimenzi, ale ve vysoké dimenzi degradují), *ball trees*, *R-stromy* apod. Ve vysoké dimenzi nebo na obrovských datech se používá *přibližné* hledání nejbližších sousedů (ANN). Tyto struktury jsou nad rámec požadavků; podstatné je, že naivní predikce je $\mathcal{O}(ND)$ na dotaz a že existují stromové struktury pro zrychlení.

3.7 Řešený příklad

Příklad — SZZ jaro 2025 č. 28: k nejbližších sousedů

(Reálná otázka SZZ: jaro 2025, č. 28 – k nejbližších sousedů)

Zadání. (1) Popište metodu k -NN pro klasifikaci i regresi; jak probíhá trénování a predikce? (2) Data se dvěma atributy se klasifikují do dvou tříd ($\circ$, $\times$) podle obrázku. Pro $k = 3$ klasifikujte nový vstup $(3, 2)$. Změnila by se klasifikace při jiném k ? (3) Jak nastavit vhodné k ?

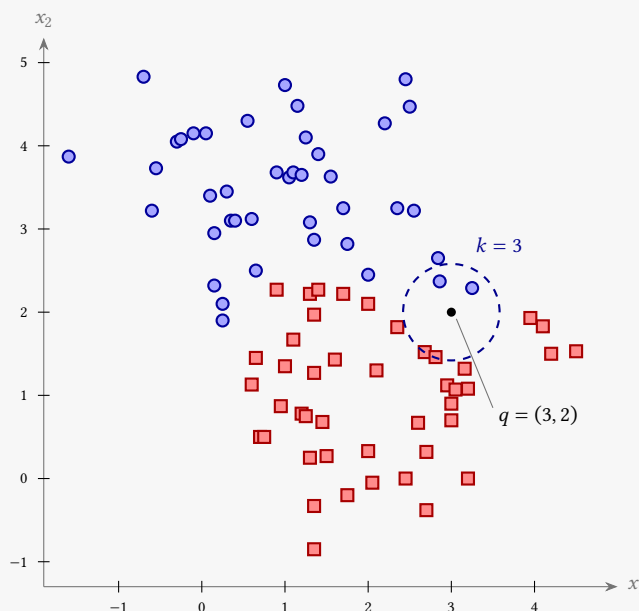
Řešení.

(1) Při **trénování** se pouze uloží celá trénovací množina (líné učení). Při **predikci** se k dotazu najde k nejbližších uložených příkladů (zvolenou metrikou, typicky Euklid); pro **klasifikaci** se vrátí jejich nejčastější třída (většinové hlasování), pro **regresi** (vážený) průměr jejich cílů.

(2) Dotaz $(3, 2)$ leží na rozhraní obou tříd (viz obrázek). Dva nejbližší body jsou *kroužky* ($\circ$) těsně nad ním, $(3,25; 2,29)$ a $(2,86; 2,37)$ ve vzdálenostech $\approx 0,38$ a $0,40$. Třetím sousedem je *křížek*: dva křížky pod dotazem, $(2,81; 1,46)$ a $(2,68; 1,52)$, leží skoro stejně daleko ($\approx 0,58$), takže nezáleží na tom, který z nich je třetí. Hlasování $2 : 1$ klasifikuje vstup jako $\circ$.

Při *změně* k se výsledek může změnit: dotaz je blízko hranice a hned za dvěma nejbližšími kroužky následují křížky. Pro $k = 5$ rozhoduje téměř remíza vzdáleností: pátý a šestý soused, kroužek $(2,84; 2,65)$ a křížek $(2,35; 1,82)$, jsou oba ve vzdálenosti $\approx 0,67$ a rozdíl je pod přesností odečtu z obrázku (poměr hlasů $3 : 2$ nebo $2 : 3$). Pro $k = 7$ jsou sousedé jednoznačně $3 \times \circ$, $4 \times \times$ (sedmý je křížek ve vzdálenosti $\approx 0,70$, osmý až $\approx 0,89$), takže hlasování přejde na $\times$. Větší k tu „dosáhne“ do husté oblasti křížků pod dotazem a klasifikaci překllopí – ukázka citlivosti na volbu k u bodu blízko rozhodovací hranice.

(3) Vhodné k zvolíme **křížovou validací**: pro $k = 1, 3, 5, \dots$ změříme křížově validovanou chybu a vybereme k s nejlepším validačním skóre. U dvou tříd volíme liché k kvůli remízám. Malé k hrozí overfittingem (citlivost na šum), velké k underfittingem (hranice příliš hladká).



Data SZZ jaro 2025 č. 28 (rekonstrukce z obrázku zadání, okolí dotazu odečteno přesně): třída $\circ$ nahore vlevo, třída $\times$ dole vpravo. Čárkovaná kružnice kolem dotazu $q = (3, 2)$ má poloměr $\approx 0,58$: uvnitř jsou dva $\circ$ (vzdálenosti $\approx 0,38$ a $0,40$), kružnice prochází dvěma $\times$ ve vzdálenosti $\approx 0,58$, z nichž jeden je třetím sousedem. Hlasování $2 : 1$ klasifikuje q jako $\circ$. Při větším k (kolem $k = 7$) se do okolí dostanou křížky pod dotazem a klasifikace se překllopí na $\times$.

(Reálná otázka SZZ: podzim 2022, č. 20 – k nejbližších sousedů)

Zadání. (1) Popište fungování k -NN pro klasifikaci i regresi – jak probíhá trénování a predikce? (2) Pro data níže (příznaky A_1, A_2 , třída $C \in \{0, 1\}$) ohodnoťte k -NN s $k = 3$ a *Manhattanskou* (L1) vzdáleností nový vstup (5, 5). (3) Změní se odpověď při jiném k ?

A_1	A_2	C
0	3	0
2	2	0
3	0	0
4	2	0
6	6	1
4	5	1
1	1	1

Řešení. (1) k -NN je líný (instance-based) model: trénování pouze uloží trénovací data. Predikce nového bodu najde jeho k nejbližších sousedů podle zvolené metriky; pro *klasifikaci* vrátí jejich většinovou třídu, pro *regresi* (vážený) průměr jejich cílů.

(2) Manhattanské vzdálenosti bodu (5, 5) od dat (seřazeno):

bod	L1 vzdálenost
(4, 5)	1
(6, 6)	2
(4, 2)	4
(2, 2)	6
(0, 3)	7
(3, 0)	7
(1, 1)	8

Tři nejbližší jsou (4, 5) s $C=1$, (6, 6) s $C=1$ a (4, 2) s $C=0$, tedy třídy $\{1, 1, 0\}$. Pro $k = 3$ je (5, 5) klasifikován jako třída 1.

(3) Ano, odpověď se mění. Pro $k = 5$ je pět nejbližších (4, 5): 1, (6, 6): 1, (4, 2): 0, (2, 2): 0, (0, 3): 0, tedy $\{1, 1, 0, 0, 0\}$, většina 0 (a stejně tak $k = 7$). Na pátém místě je sice remíza vzdáleností ((0, 3) i (3, 0) mají vzdálenost 7), oba body ale patří do třídy 0, takže na výsledku nezáleží. Bod leží blízko rozhraní tříd, proto je citlivý na volbu k .

4 Lineární regrese

Lineární regrese je nejjednodušší regresní model: predikuje cíl jako *lineární kombinaci* příznaků. Přesto je klíčová – má **analytické (uzavřené) řešení** metodou nejmenších čtverců, dá se trénovat i **gradientním sestupem** (SGD) a na jejím příkladu se nejlépe ukáže L^2 regularizace. Okruh S1 ji zmiňuje přehledově; zde rozebíráme obojí řešení do hloubky, kterou S1 přenechává tomuto okruhu.

4.1 Model a chybová funkce

Definice – Model lineární regrese

Pro vstup $\mathbf{x} \in \mathbb{R}^D$ predikuje **lineární regrese**

$$y(\mathbf{x}; \mathbf{w}, b) = w_1 x_1 + w_2 x_2 + \dots + w_D x_D + b = \mathbf{x}^\top \mathbf{w} + b.$$

Vektor $\mathbf{w} \in \mathbb{R}^D$ jsou *váhy*, b je *bias* (posun, absolutní člen). Ve statistickém značení regrese se píše jako koeficienty $\beta_1, \dots, \beta_D$ (váhy) a β_0 (bias); jde o totéž.

Intuice. [Bias jako další váha (padding jedničkou)] S biasem se nepohodlně počítá zvlášť. Trik: každý vstup **rozšíříme o konstantní příznak** 1, tj. $\mathbf{x} \mapsto (x_1, \dots, x_D, 1)$, a bias se stane poslední váhou. Pak píšeme jednotně $y(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ a „váhy“ už zahrnují i bias. V maticovém zápisu to znamená přidat do datové matice $\mathbf{X}$ *sloupec jedniček*. Těto úmluvy se držíme v celé sekci.

Definice – Chybová funkce: MSE a součet čtverců

Máme trénovací data $\mathbf{x}_1, \dots, \mathbf{x}_N$ s cíli $t_1, \dots, t_N$. Kvalitu vah měříme **střední kvadratickou chybou** (mean squared error)

$$\text{MSE}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y(\mathbf{x}_i; \mathbf{w}) - t_i)^2.$$

Často se místo ní minimalizuje **součet čtverců** (sum of squares)

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y(\mathbf{x}_i; \mathbf{w}) - t_i)^2.$$

Obě mají *stejné minimum* (MSE má před součtem faktor $\frac{1}{N}$, součet čtverců faktor $\frac{1}{2}$ – liší se tedy jen kladnou multiplikativní konstantou), ale součet čtverců se příjemněji derivuje – faktor $\frac{1}{2}$ se vykrátí proti dvojce z derivace kvadrátu.

Intuice. Hledáme přímku (obecně nadrovinu), která prochází mraku bodů „co nejbliž“ – tak, aby *součet čtverců svislých vzdáleností* bodů od přímky byl nejmenší. Svislá vzdálenost $\hat{t}_i - t_i$ se nazývá *reziduum*. Kvadrát (místo absolutní hodnoty) penalizuje velké odchylky silněji a činí funkci hladkou a konvexní, takže každé lokální minimum je globální (a jediné, je-li $\mathbf{X}^\top \mathbf{X}$ regulární, viz níže). Vztah k věrohodnosti: pro normálně rozdělený šum kolem přímky dává metoda nejmenších čtverců přesně maximum likelihood estimation (MLE, maximálně věrohodný odhad; viz sekce 1).

4.2 Analytické řešení: metoda nejmenších čtverců

Vyjádříme součet čtverců maticově. Datová matice $\mathbf{X} \in \mathbb{R}^{N \times D}$ má příklad $\mathbf{x}_i^\top$ na i -tém řádku (po paddingu jedničkou má D sloupců včetně biasu), $\mathbf{t} \in \mathbb{R}^N$ je vektor cílů. Pak $(\mathbf{X}\mathbf{w})_i = \mathbf{x}_i^\top \mathbf{w}$ a

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w} - t_i)^2 = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2.$$

★ Věta – Normální rovnice nejmenších čtverců

Minimum součtu čtverců $E(\mathbf{w}) = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2$ splňuje **normální rovnice**

$$\mathbf{X}^\top \mathbf{X} \mathbf{w} = \mathbf{X}^\top \mathbf{t}.$$

Je-li matice $\mathbf{X}^\top \mathbf{X}$ (rozměru $D \times D$) regulární, má řešení tvar

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}.$$

💡 Intuice. [Odvození přes nulový gradient (Fermat)] Toto *není* formální důkaz, jen výpočet – ukázka, jak se ke vzorci dojde. Funkce E je hladká a konvexní, takže její minimum je tam, kde je gradient nulový (Fermatova věta o vnitřním extrémě, vícerozměrná verze: $\nabla_{\mathbf{w}} E = \mathbf{0}$). Krok po kroku:

1. **Parciální derivace podle jedné váhy w_j .** $Z E = \frac{1}{2} \sum_i (\mathbf{x}_i^\top \mathbf{w} - t_i)^2$ řetězovým pravidlem

$$\frac{\partial E}{\partial w_j} = \frac{1}{2} \sum_{i=1}^N 2(\mathbf{x}_i^\top \mathbf{w} - t_i) x_{ij} = \sum_{i=1}^N x_{ij}(\mathbf{x}_i^\top \mathbf{w} - t_i).$$

2. **Položíme rovno nule pro všechna j .** Suma $\sum_i x_{ij}(\dots)$ je skalární součin j -tého sloupce $\mathbf{X}$ s vektorem $(\mathbf{X}\mathbf{w} - \mathbf{t})$, tj. $\mathbf{X}_{*,j}^\top (\mathbf{X}\mathbf{w} - \mathbf{t}) = 0$.
3. **Sebereme rovnice přes všechna j do matice.** Dostaneme $\mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{t}) = \mathbf{0}$, tj. $\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{t}$.
4. **Invertujeme** (je-li $\mathbf{X}^\top \mathbf{X}$ regulární): $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$.

Že jde o minimum (ne maximum/sedlo), plyne z konvexity E : její Hessián $\mathbf{X}^\top \mathbf{X}$ je pozitivně semidefinitní.

📌 Poznámka. [Singularita a SVD] Matice $\mathbf{X}^\top \mathbf{X}$ je singulární (neinvertibilní), pokud sloupce $\mathbf{X}$ jsou lineárně závislé – typicky když je příznaků víc než příkladů ($D > N$) nebo jsou některé příznaky kolineární. Pak normální rovnice nemá jednoznačné řešení. Místo přímé inverze se používá **SVD** (singulární rozklad) $\mathbf{X}$, který spolehlivě najde řešení (přes pseudoinverzi) i v singulárním a numericky špatně podmíněném případě; je to i standardní postup uvnitř knihovných funkcí (`numpy.linalg.lstsq`). Alternativně singularitu odstraní L^2 regularizace (níže).

⚙️ Algoritmus – Analytické řešení lineární regrese

Vstup: data $\mathbf{X} \in \mathbb{R}^{N \times D}$ (se sloupcem jedniček), cíle $\mathbf{t} \in \mathbb{R}^N$.

Výstup: váhy $\mathbf{w}$ minimalizující MSE.

1. Sestav $\mathbf{X}^\top \mathbf{X}$ a $\mathbf{X}^\top \mathbf{t}$.
2. Je-li $\mathbf{X}^\top \mathbf{X}$ regulární: $\mathbf{w} \leftarrow (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$.

▷ $D \times D$ matice, D -vektor

▷ jinak řeš přes SVD

Složitost $\mathcal{O}(ND^2)$ (sestavení $\mathbf{X}^\top \mathbf{X}$) za předpokladu $N \geq D$.

📌 Příklad – SZZ léto 2024 č. 31: Metoda nejmenších čtverců

(Reálná otázka SZZ: léto 2024, č. 31 – Metoda nejmenších čtverců)

Zadání. Data s jedním číselným atributem x a cílem y : (1, 1), (2, 3), (3, 2). (1) Popište lineární model a metodu nejmenších čtverců. (2) Odhadněte koeficienty modelu. (3) Spočítejte střední kvadratickou chybu modelu.

Řešení. (1) Model je přímka $\hat{y} = \beta_0 + \beta_1 x$ (β_0 posun, β_1 směrnice); koeficienty odhadneme tak, aby součet čtverců reziduí $\sum_i (\beta_0 + \beta_1 x_i - y_i)^2$ byl nejmenší (nulový gradient $\Rightarrow$ normální rovnice).

(2) Pro jednu proměnnou stačí uzavřené vzorce s průměry $\bar{x} = 2$, $\bar{y} = 2$:

$$\beta_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{(-1)(-1) + 0 \cdot 1 + 1 \cdot 0}{(-1)^2 + 0^2 + 1^2} = \frac{1}{2}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x} = 2 - \frac{1}{2} \cdot 2 = 1.$$

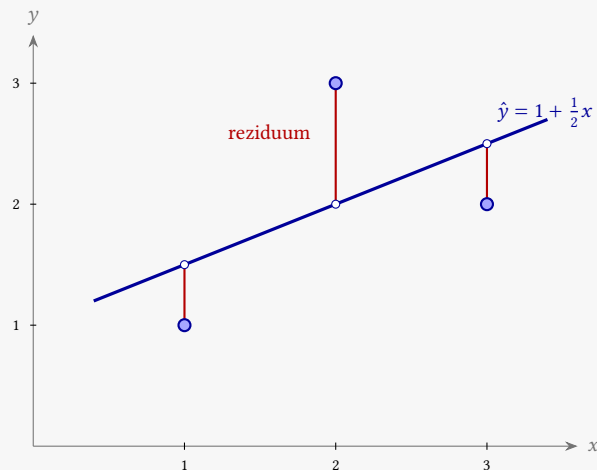
Tedy $\hat{y} = 1 + \frac{1}{2}x$ (totéž dá maticově $(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$ se sloupcem jedniček).

(3) Predikce $\hat{y} = (1,5; 2; 2,5)$, rezidua $y - \hat{y} = (-0,5; 1; -0,5)$. Pak

$$\text{MSE} = \frac{1}{3}((-0,5)^2 + 1^2 + (-0,5)^2) = \frac{0,25 + 1 + 0,25}{3} = \frac{1,5}{3} = \boxed{0,5}.$$

(Ověřeno `numpy.linalg.lstsq` dá $(\beta_1, \beta_0) = (0,5; 1)$, $\text{MSE} = 0,5$, součet čtverců reziduí = 1,5.)

Geometricky: přímka prokládá tři body s minimálním součtem čtverců svislých úseček (reziduí).



Proložení nejmenšími čtverci dat $(1, 1), (2, 3), (3, 2)$ přímkou $\hat{y} = 1 + \frac{1}{2}x$. Plné body jsou data, prázdné kroužky predikce na přímce, červené svislé úsečky jsou rezidua $y_i - \hat{y}_i = (-0,5; 1; -0,5)$. Metoda minimalizuje součet jejich čtverců $(= 1,5)$.

Příklad – SZZ podzim 2025 č. 30: Lineární regrese a L2 (centrovaná data)

(Reálná otázka SZZ: podzim 2025, č. 30 – Lineární regrese, L2 regularizace)

Zadání. Data (x, y) : $(-1, 1), (0, 3), (1, 2)$. (1) Popište model a MNČ. (2) Spočítejte OLS koeficienty (bez regularizace). (3) Popište cílovou funkci s L^2 a určete *směr* změny koeficientu u x oproti OLS.

Řešení. (1) Model $f(x) = \beta_0 + \beta_1 x$, MNČ minimalizuje $\sum_i (f(x_i) - y_i)^2$, řešení $\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ se sloupcem jedniček.

(2) Data jsou *centrovaná* ($\bar{x} = 0$), takže $\mathbf{X}^T \mathbf{X}$ vyjde diagonální a výpočet je krátký. Se sloupcem jedniček $\mathbf{X} = \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$:

$$\mathbf{X}^T \mathbf{X} = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}, \quad \mathbf{X}^T \mathbf{y} = \begin{pmatrix} 1+3+2 \\ -1+3+2 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \end{pmatrix}, \quad \beta = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 6 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1/2 \end{pmatrix}.$$

Tedy $\beta_0 = 2, \beta_1 = \frac{1}{2}$, model $\hat{y} = 2 + \frac{1}{2}x$. (Ověřeno numpy: $\beta = (2; 0,5)$.)

(3) L^2 minimalizuje $\sum_i (f(x_i) - y_i)^2 + \lambda \beta_1^2$ ($\lambda > 0$, bias β_0 se nepenalizuje). Penalizace táhne β_1 k nule, takže $|\beta_1|$ *klesne* (přímka se zploští), znaménko zůstane kladné. Intercept β_0 se zde nezmění (data jsou centrovaná, $\bar{x} = 0$); obecně se měnit může.

4.3 Trénování pomocí SGD

Analytické řešení vyžaduje invertovat $D \times D$ matici, což je pro velké D pomalé a paměťově náročné. Iterativní alternativa je **gradientní sestup**: opakovaně se posouváme proti gradientu chyby.

Definice – Gradientní sestup, SGD, minibatch

Minimalizujeme-li chybu $E(\mathbf{w})$, **gradientní sestup** opakuje krok

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} E(\mathbf{w}),$$

kde $\alpha > 0$ je **rychlost učení** (learning rate, délka kroku). Podle toho, z kolika příkladů gradient odhadujeme:

- **(dávkový) gradientní sestup** – z *celé* trénovací množiny (přesný gradient, drahý krok);
- **stochastický gradientní sestup (SGD)** – z *jednoho náhodného* příkladu (nestranný, ale zašuměný odhad gradientu, levný krok);

- **minibatch SGD** – kompromis: z náhodné podmnožiny B o B příkladech.

Jeden průchod všemi trénovacími daty (rozdělenými na minibatche náhodnou permutací) se nazývá **epocha**.

Pro lineární regresi s chybou $\frac{1}{2}(\mathbf{x}^\top \mathbf{w} - t)^2$ (a volitelně L^2 členem $\frac{\lambda}{2}\|\mathbf{w}\|^2$) je gradient přes minibatch

$$\nabla_{\mathbf{w}} E(\mathbf{w}) \approx \frac{1}{|B|} \sum_{i \in B} (\mathbf{x}_i^\top \mathbf{w} - t_i) \mathbf{x}_i + \lambda \mathbf{w}.$$

Reziduum $(\mathbf{x}_i^\top \mathbf{w} - t_i)$ škáluje příspěvek vstupu $\mathbf{x}_i$; L^2 člen přidá $\lambda \mathbf{w}$ (tlak vah k nule, *weight decay*); složka odpovídající biasu se v praxi obvykle nepenalizuje.

⚙️ Algoritmus – Lineární regrese minibatch SGD

Vstup: data $(\mathbf{X}, \mathbf{t})$, rychlost učení $\alpha > 0$, síla L^2 $\lambda \geq 0$.

Výstup: váhy $\mathbf{w}$ (přibližně) minimalizující regularizovanou MSE.

1. $\mathbf{w} \leftarrow \mathbf{0}$ (nebo náhodně). ▷ inicializace
2. Opakuj do konvergence (přes epochy):
 - (a) navzorkuj minibatch indexů B ; ▷ náhodná permutace $\rightarrow$ chunky velikosti $B = \text{epocha}$
 - (b) $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{|B|} \sum_{i \in B} (\mathbf{x}_i^\top \mathbf{w} - t_i) \mathbf{x}_i - \alpha \lambda \mathbf{w}$. ▷ krok proti gradientu

i Poznámka. [Konvergence a normalizace] Součet čtverců je *konvexní* funkce vah, takže gradientní sestup s dostatečně malým krokem konverguje ke globálnímu optimu (pro konvexní hladkou ztrátu; přesněji SGD konverguje skoro jistě, klesá-li krok dostatečně, ale ne moc rychle – podmínky $\sum_i \alpha_i = \infty$, $\sum_i \alpha_i^2 < \infty$). Příznaky v různých škálách potřebují různé kroky; proto se před SGD obvykle **normalizují** – min–max škálováním do $[0, 1]$, nebo *standardizací* $x' = \frac{x - \hat{\mu}}{\hat{\sigma}}$ (nulový průměr, jednotkový rozptyl). L^2 regularizaci a výběr λ /stupně polynomu coby hyperparametrů řešíme obecně v sekci 2.

📌 Příklad – SZZ jaro 2024 č. 27: Krok SGD s L^2 regularizací

(Reálná otázka SZZ: jaro 2024, č. 27 – Lineární regrese a regularizace)

Zadání. Model $y = \beta_1 x_1 + \beta_2 x_2 + \beta_0$, data $\{(0, 2; 3), (2, 3; 9), (1, 1; 2), (2, 0; 2)\}$ (řádky $x_1, x_2; y$). Učící konstanta $\alpha = 0,1$, L^2 síla $\lambda = 0,5$. (1) Definujte chybovou funkci (součet čtverců + L^2). (2) Z odhadu $\beta_1 = 1, \beta_2 = 3, \beta_0 = -1$ proveďte *jeden* krok gradientní metody.

Řešení. (1) $E = \sum_{i=1}^N (y_i - \hat{y}_i)^2 + \lambda \sum_{j \geq 1} \beta_j^2$, kde $\hat{y}_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_0$; bias β_0 se do regularizace nezahrnuje.

(2) Gradient počítáme jako *součet přes všechny čtyři vzory* (zde dávkový krok). Pro tuto formu chyby (bez faktoru $\frac{1}{2}$) je $\frac{\partial E}{\partial \beta_0} = \sum_i -2(y_i - \hat{y}_i)$, $\frac{\partial E}{\partial \beta_j} = \sum_i -2(y_i - \hat{y}_i)x_{ij} + 2\lambda\beta_j$. Spočteme predikce a rezidua $r_i = \hat{y}_i - y_i$:

i	x_1	x_2	y	$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \beta_0$	$r = \hat{y} - y$
1	0	2	3	$0 + 6 - 1 = 5$	2
2	2	3	9	$2 + 9 - 1 = 10$	1
3	1	1	2	$1 + 3 - 1 = 3$	1
4	2	0	2	$2 + 0 - 1 = 1$	-1

Derivace součtu čtverců E_{RSS} (nepenalizovaná část chyby; pozor: $-2(y - \hat{y}) = 2r$, sčítáme příspěvky $2r_i$, resp. $2r_i x_{ij}$):

$$\frac{\partial E_{\text{RSS}}}{\partial \beta_0} = 2(2+1+1-1) = 6, \quad \frac{\partial E_{\text{RSS}}}{\partial \beta_1} = 2(2 \cdot 0 + 1 \cdot 2 + 1 \cdot 1 - 1 \cdot 2) = 2, \quad \frac{\partial E_{\text{RSS}}}{\partial \beta_2} = 2(2 \cdot 2 + 1 \cdot 3 + 1 \cdot 1 - 1 \cdot 0) = 16.$$

Regularizační člen přidá $\frac{\partial}{\partial \beta_j} \lambda \beta_j^2 = 2\lambda \beta_j = 2 \cdot 0,5 \cdot \beta_j = \beta_j$ (jen k vahám, ne k β_0): $+\beta_1 = 1$, $+\beta_2 = 3$.
Celkové gradienty:

$$g_0 = 6, \quad g_1 = 2 + 1 = 3, \quad g_2 = 16 + 3 = 19.$$

Krok $\beta \leftarrow \beta - \alpha g$ s $\alpha = 0,1$:

$$\beta_0 = -1 - 0,1 \cdot 6 = -1,6, \quad \beta_1 = 1 - 0,1 \cdot 3 = 0,7, \quad \beta_2 = 3 - 0,1 \cdot 19 = 1,1.$$

(Ověřeno numpy: $g = (6; 3; 19)$, nové $\beta = (-1,6; 0,7; 1,1)$ – shoduje se s oficiálním náčrtem.)

4.4 OLS vs. SGD a krátce o nelinearitě

Příklad – SZZ léto 2025 č. 30: Lineární regrese – analytický vs. SGD

(Reálná otázka SZZ: léto 2025, č. 30 – Lineární regrese)

Zadání (zkráceně). (1) Zformulujte model a chybovou funkci. (2) Jak získat optimum *explicitně*? (3) Zformulujte optimalizaci pomocí SGD. (4) Výhody a nevýhody obou, zejména z hlediska kontroly chyby na validačních datech.

Řešení. (1) $y(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ (bias paddingem 1), minimalizujeme součet čtverců $\frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2$. (2) *Analytický*: položíme gradient roven nule, dostaneme normální rovnice $\mathbf{X}^\top \mathbf{X} \mathbf{w} = \mathbf{X}^\top \mathbf{t}$ a řešíme přes inverzi $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$, resp. přes SVD při singularitě. (3) *SGD*: opakuj $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (\mathbf{x}_i^\top \mathbf{w} - t_i) \mathbf{x}_i$ po minibatchích/epochách.

(4) **Analytické řešení**: přesné, v jednom kroku, bez ladění; ale inverze $D \times D$ matice je drahá a paměťově náročná pro velké D , a přesné optimum na trénovacích datech může způsobit *overfitting* (přeučení). **SGD**: škáluje na velká data a málo paměti, snadno přidá regularizaci – a hlavně *můžeme v průběhu sledovat chybu na validační množině* a zastavit, než nastane *overfitting* (**early stopping**); naopak vyžaduje ladit α , počet epoch a konverguje jen přibližně. Stručně: *malé D a potřeba přesnosti $\rightarrow$ analytický; velká data a kontrola overfittingu $\rightarrow$ SGD.*

i Poznámka. [Polynomiální příznaky: most k nelinearitě] Lineární regrese modeluje jen přímky/nadroviny, ale „lineární“ se vztahuje k *vahám*, ne ke vstupu. Rozšíříme-li vstup x o mocniny, příznaky $(x^0, x^1, \dots, x^M)$, predikce $w_0 + w_1 x + \dots + w_M x^M$ je polynom stupně M – model zůstává lineární ve $\mathbf{w}$, takže platí všechny vzorce výše. Vyšší M zvyšuje kapacitu (riziko overfittingu, viz sekce 2). Obecněji se konstruují *příznaky* (feature engineering), např. one-hot kódování kategorických hodnot.

5 Logistická regrese

Logistická regrese je přes svůj název *klasifikační* model: lineární funkci vstupu protáhne *logistickou* (*sigmoidní*) funkcí a výstup čte jako *pravděpodobnost* třídy. Je to jedna z nejpoužívanějších klasifikačních technik a zároveň nejjednodušší *pravděpodobnostní* klasifikátor trénovaný *maximum likelihood estimation* (MLE, maximální věrohodnost; sekce 1). V této sekci ji zavedeme pro binární klasifikaci, odvodíme její ztrátovou funkci i gradient a zobecníme ji softmaxem na více tříd.

i Poznámka. [Vztah k S1] Okruh S1 (sekce *Strojové učení*) zmiňuje logistickou regresi přehledově jako „změkčený tvrdý práh“ s pravděpodobnostním výstupem. Zde to **nezdvojujeme** – jdeme do hloubky, kterou S1 přenechává S3: parametrizace přes sigmoid, ztráta jako negativní log-věrohodnost, odvození gradientu a SGD, multiclass softmax.

5.1 Binární logistická regrese

Od tvrdého prahu k sigmoidě

Lineární klasifikátor počítá skóre $\bar{y}(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x} + b$ a třídu přiřadí podle znaménka (prahu). *Perceptron* (historicky nejstarší algoritmus, Rosenblatt 1958) používá *tvrdý* práh $\text{sign}(\bar{y})$: vrací jen třídu $\{-1, +1\}$, na lineárně neseparabilních datech nekonverguje a nedává pravděpodobnosti. Logistická regrese tvrdý práh *změkčí* – skóre $\bar{y}$ protáhne **sigmoidou** a výstup interpretuje jako $P(C_1 | \mathbf{x})$.

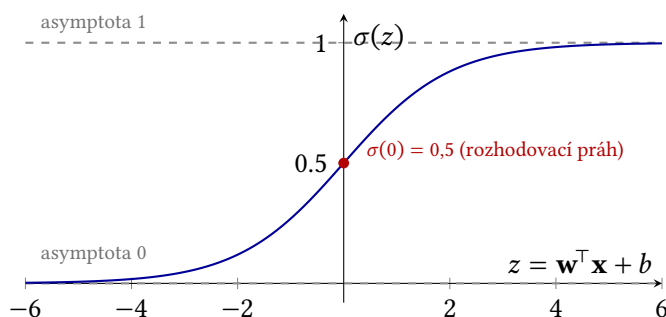
Definice – Sigmoidní (logistická) funkce

Sigmoida je funkce

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Má hodnoty v otevřeném intervalu $(0, 1)$, je rostoucí, $\sigma(0) = \frac{1}{2}$, $\sigma(-z) = 1 - \sigma(z)$ (středová symetrie kolem bodu $(0, \frac{1}{2})$) a asymptoticky $\sigma(z) \rightarrow 1$ pro $z \rightarrow +\infty$, $\sigma(z) \rightarrow 0$ pro $z \rightarrow -\infty$. Její derivace je

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)), \quad \sigma'(0) = \frac{1}{4}.$$



Sigmoida $\sigma(z) = 1/(1 + e^{-z})$. Hladce přechází z 0 do 1, hodnotu 0,5 nabývá v $z = 0$ (rozhodovací práh) a k asymptotám 0 a 1 se blíží pro velká záporná, resp. kladná z . Záporné z tedy znamená pravděpodobnost třídy C_1 pod 50 %, kladné nad 50 %.

Definice – Model binární logistické regrese

Pro binární klasifikaci ($t \in \{0, 1\}$) logistická regrese modeluje podmíněnou pravděpodobnost pozitivní třídy C_1 pomocí lineární funkce protažené sigmoidou. Označíme *lineární část* $\bar{y}(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ a celkový výstup

$$y(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}), \quad P(C_1 | \mathbf{x}) = y(\mathbf{x}; \mathbf{w}), \quad P(C_0 | \mathbf{x}) = 1 - y(\mathbf{x}; \mathbf{w}).$$

Předikce. Vstup zařadíme do třídy C_1 , je-li $y(\mathbf{x}; \mathbf{w}) \geq 0,5$ (rozhodovací práh 0,5), jinak do C_0 . Protože $\sigma(z) \geq \frac{1}{2} \iff z \geq 0$, je rozhodnutí ekvivalentní podmínce $\mathbf{w}^\top \mathbf{x} \geq 0$.

Intuice. [Bias jako práh; lineární rozhodovací hranice] *Bias* b (absorbovaný do $\mathbf{w}$ přidáním stálého příznaku $x_0 = 1$, jak děláme dál) hraje roli trénovatelného prahu, proto stačí porovnávat skóre s nulou. **Rozhodovací hranice** je množina bodů s $y = 0,5$, tj. $\mathbf{w}^\top \mathbf{x} = 0$ – to je *nadrovina* (přímka v 2D), takže logistická regrese je *lineární* klasifikátor. Sigmoida mění jen *tvar* přechodu mezi třídami (jak rychle pravděpodobnost roste přes hranici), ne jeho *polohu*.

Poznámka. [Logit jako log-šance] Inverze sigmoidy dává pro $p = P(C_1 | \mathbf{x})$ lineární část jako *logit*:

$$\bar{y}(\mathbf{x}; \mathbf{w}) = \log \frac{p}{1 - p} = \log \frac{P(C_1 | \mathbf{x})}{P(C_0 | \mathbf{x})}.$$

Lineární skóre tak má význam *logaritmu šance* (log-odds) mezi oběma třídami: $\bar{y} > 0$ znamená šanci ve prospěch C_1 , $\bar{y} < 0$ ve prospěch C_0 .

5.2 Ztrátová funkce, gradient a SGD

Ztráta = negativní log-věrohodnost (křížová entropie)

Logistická regrese se trénuje *maximum likelihood estimation* (MLE) (sekce 1). Výstup $y(\mathbf{x}; \mathbf{w})$ je přímo pravděpodobnost třídy C_1 , takže cíl t pochází z *Bernoulliho* rozdělení s parametrem $\varphi = y$: $p(t | \mathbf{x}; \mathbf{w}) = y^t(1 - y)^{1-t}$. Minimalizujeme *negativní logaritmicou věrohodnost* (NLL); pro minibatch N příkladů je ztráta

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N -\log p(C_i | \mathbf{x}_i; \mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \left(-t_i \log y(\mathbf{x}_i; \mathbf{w}) - (1 - t_i) \log (1 - y(\mathbf{x}_i; \mathbf{w})) \right).$$

Definice – Binární křížová entropie

Pro jeden příklad s výstupem $y = \sigma(\mathbf{w}^\top \mathbf{x})$ a cílem $t \in \{0, 1\}$ je ztráta (binární křížová entropie, = NLL Bernoulliho modelu)

$$E = -t \log y - (1 - t) \log(1 - y).$$

Intuice. Pro $t = 1$ se z ní stane $-\log y$ (chceme $y \rightarrow 1$), pro $t = 0$ je to $-\log(1 - y)$ (chceme $y \rightarrow 0$). Jistá *a správná* predikce dá ztrátu 0; jistá *a chybná* predikce ($y \rightarrow 0$ při $t = 1$) ztrátu $+\infty$ – model je za sebevědomý omyl tvrdě penalizován. Název „křížová entropie“ plyne z obecného rámce MLE: NLL = křížová entropie mezi empirickým rozdělením cílů a modelem (sekce 1). Pozn.: kvadratická (MSE) ztráta se zde *nepoužívá* – vede k nekonvexnímu problému a mizejícím gradientům v saturaci sigmoidy.

Odvození gradientu

Toto odvození žádá otázka SZZ jaro 2025 (č. 29, část 2). Provedeme derivaci krok po kroku pro *jeden* příklad; značme $z = \mathbf{w}^\top \mathbf{x}$, takže $y = \sigma(z)$ a $E = -t \log \sigma(z) - (1 - t) \log (1 - \sigma(z))$. Cílem je $\nabla_{\mathbf{w}} E$.

★ Věta – Gradient ztráty logistické regrese

Pro jeden příklad $(\mathbf{x}, t)$ a ztrátu $E = -t \log \sigma(\mathbf{w}^\top \mathbf{x}) - (1 - t) \log (1 - \sigma(\mathbf{w}^\top \mathbf{x}))$ platí

$$\frac{\partial E}{\partial z} = \sigma(z) - t, \quad \text{a tedy} \quad \boxed{\nabla_{\mathbf{w}} E = (\sigma(\mathbf{w}^\top \mathbf{x}) - t) \mathbf{x}}.$$

Odvození (krok po kroku).

(1) **Derivace sigmoidy.** Z $\sigma(z) = (1 + e^{-z})^{-1}$ řetízovým pravidlem

$$\sigma'(z) = -(1 + e^{-z})^{-2} \cdot (-e^{-z}) = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}} = \sigma(z)(1 - \sigma(z)),$$

kde poslední rovnost využívá $\frac{e^{-z}}{1 + e^{-z}} = \frac{(1 + e^{-z}) - 1}{1 + e^{-z}} = 1 - \sigma(z)$.

(2) **Derivace ztráty podle z .** Použijeme $\frac{\partial}{\partial z} \log \sigma = \frac{\sigma'}{\sigma}$ a $\frac{\partial}{\partial z} \log(1 - \sigma) = \frac{-\sigma'}{1 - \sigma}$:

$$\frac{\partial E}{\partial z} = -t \cdot \frac{\sigma'(z)}{\sigma(z)} - (1 - t) \cdot \frac{-\sigma'(z)}{1 - \sigma(z)} = -t \frac{\sigma'(z)}{\sigma(z)} + (1 - t) \frac{\sigma'(z)}{1 - \sigma(z)}.$$

(3) **Dosazení** $\sigma' = \sigma(1 - \sigma)$. V prvním zlomku se zkrátí σ , ve druhém $(1 - \sigma)$:

$$\frac{\partial E}{\partial z} = -t(1 - \sigma(z)) + (1 - t)\sigma(z).$$

(4) **Roznásobení a zjednodušení.**

$$\frac{\partial E}{\partial z} = -t + t\sigma(z) + \sigma(z) - t\sigma(z) = \sigma(z) - t.$$

Členy $\pm t\sigma(z)$ se vyruší; zbývá rozdíl predikce a cíle.

(5) **Přechod ke gradientu podle $\mathbf{w}$.** Protože $z = \mathbf{w}^\top \mathbf{x}$, je $\nabla_{\mathbf{w}} z = \mathbf{x}$. Řetízkovým pravidlem

$$\nabla_{\mathbf{w}} E = \frac{\partial E}{\partial z} \nabla_{\mathbf{w}} z = (\sigma(z) - t) \mathbf{x} = (y(\mathbf{x}; \mathbf{w}) - t) \mathbf{x}.$$

□

💡 Intuice. [Elegantní tvar a jednotnost s lineární regresí] Gradient má pozoruhodně jednoduchý tvar (*chyba predikce*) $\times$ (*vstup*): $(y - t)\mathbf{x}$. Je-li predikce přesná ($y = t$), gradient je nulový a váhy se nemění; čím větší chyba, tím větší krok, a to ve směru příznaků $\mathbf{x}$. Stejný tvar $(y - t)\mathbf{x}$ má gradient lineární regrese s MSE ztrátou (sekce 4) i softmaxové regrese (níže) – to není náhoda, ale důsledek zobecněných lineárních modelů: vhodná aktivace (identita / sigmoida / softmax) s odpovídajícím rozdělením (normální / Bernoulli / kategorické) a NLL ztrátou vždy vede k témuž tvaru gradientu.

Stochastický gradientní sestup

Minimum ztráty nemá logistická regrese (na rozdíl od lineární) v uzavřeném tvaru, hledáme ho proto *gradientním sestupem*. V praxi *stochastickým* (SGD): místo plného gradientu přes všechna data použijeme odhad z jednoho příkladu nebo malé dávky (minibatch).

⚙️ Algoritmus – Trénování logistické regrese (minibatch SGD)

Vstup: data ($\mathbf{X} \in \mathbb{R}^{N \times D}$, $\mathbf{t} \in \{0, 1\}^N$), rychlost učení $\alpha > 0$.

Výstup: váhy $\mathbf{w}$.

1. $\mathbf{w} \leftarrow \mathbf{0}$ (nebo malé náhodné hodnoty)

▷ inicializace

2. **opakuji** do konvergence (nebo vyčerpání trpělivosti) pro minibatch $\mathcal{B}$:

a) $\mathbf{g} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (\sigma(\mathbf{w}^\top \mathbf{x}_i) - t_i) \mathbf{x}_i$

▷ průměrný gradient na dávce

b) $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{g}$

▷ krok proti gradientu

Pro jeden příklad (čistý SGD, $|\mathcal{B}| = 1$) je krok přímo

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha (\sigma(\mathbf{w}^\top \mathbf{x}) - t) \mathbf{x}.$$

💡 Poznámka. [Prevence overfittingu] Vše, co platí o *příznamech* a L^2 *regularizaci* obecně, platí i zde (sekce 2). Overfittingu (přeučení) při SGD bráníme zejména:

- L^2 **regularizaci** – ke ztrátě přidáme $\frac{\lambda}{2} \|\mathbf{w}\|^2$, čímž do kroku přibude člen $-\alpha \lambda \mathbf{w}$ (váhy se v každém kroku mírně „smršťují“ k nule);
- **early stopping** (včasné zastavení) – sledujeme *validační* chybu a trénování ukončíme, jakmile začne růst (model se přestává učit signál a začíná se učit šum).

Pomáhá i rozumný *learning-rate schedule* (postupné snižování α). Podrobnosti a obrázek validační vs. trénovací chyby viz sekce 2.

5.3 Klasifikace do více tříd: softmax

Pro $K > 2$ tříd zobecníme model dvěma kroky: vygenerujeme K lineárních výstupů (každý se svými vahami) a sigmoidu nahradíme **softmaxem**, který K čísel převede na rozdělení pravděpodobností přes třídy.

Definice – Multinomiální logistická regrese (softmax)

Pro K tříd máme matici vah $\mathbf{W} \in \mathbb{R}^{D \times K}$ (sloupec na třídu). Lineární část je vektor $\bar{\mathbf{y}}(\mathbf{x}; \mathbf{W}) = \mathbf{x}^\top \mathbf{W} \in \mathbb{R}^K$ (jeho i -tá složka je skóre třídy C_i). *Softmax* z něj dělá pravděpodobnosti:

$$P(C_i | \mathbf{x}; \mathbf{W}) = \text{softmax}(\bar{\mathbf{y}})_i = \frac{e^{\bar{y}_i}}{\sum_{j=0}^{K-1} e^{\bar{y}_j}}.$$

Výsledné hodnoty jsou v $(0, 1)$ a sčítají se na 1. Klasifikujeme do třídy s *nejvyšší* pravděpodobností, $\hat{c} = \text{argmax}_i P(C_i | \mathbf{x}; \mathbf{W})$. Model se též nazývá *softmax regrese* nebo *maximum entropy* klasifikátor.

 **Intuice.** [Softmax zobecňuje sigmoidu] Softmax je „měkký argmax“: zvýrazní největší skóre, ale zachová pravděpodobnostní rozdělení (nejen tvrdé vítězství). Pro $K = 2$ se redukuje na sigmoidu, $\sigma(z) = \text{softmax}([z, 0])_0 = \frac{e^z}{e^z + e^0} = \frac{1}{1 + e^{-z}}$. Exponenciála je vždy kladná, dělení normalizuje na součet 1. Softmax je invariantní vůči přičtení konstanty ke všem skóre ($\text{softmax}(\bar{\mathbf{y}} + c) = \text{softmax}(\bar{\mathbf{y}})$), takže model je mírně *přeurčený* – pro K tříd by stačilo $K - 1$ sad vah.

 **Poznámka.** [Ztráta a trénování – analogicky binárnímu případu] Cíl $t \in \{0, 1, \dots, K - 1\}$ reprezentujeme *one-hot* vektorem $\mathbf{1}_t$ (1 na pozici správné třídy, jinak 0). Cíl pochází z *kategorického* rozdělení, ztráta je opět $\text{NLL} = (\text{kategorická})$ křížová entropie:

$$E(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N -\log P(C_{t_i} | \mathbf{x}_i; \mathbf{W}).$$

Trénuje se stejným minibatch SGD jako binární případ. Gradient ztráty podle vah třídy má opět tvar $(chyba) \times (vstup)$,

$$\nabla_{\mathbf{W}} E = \mathbf{x} (\mathbf{y}(\mathbf{x}) - \mathbf{1}_t)^\top \in \mathbb{R}^{D \times K} \quad (\text{pro jeden příklad; sloupec třídy } i \text{ je } (y_i - [t=i]) \mathbf{x}),$$

kde $\mathbf{y}(\mathbf{x}) = \text{softmax}(\mathbf{x}^\top \mathbf{W})$ je vektor predikovaných pravděpodobností. Odvození je analogické binárnímu případu: derivace křížové entropie podle skóre softmaxu vyjde $\partial E / \partial \bar{\mathbf{y}} = \mathbf{y}(\mathbf{x}) - \mathbf{1}_t$. Rozhodovací oblasti softmaxu jsou *konvexní* (a tedy souvislé): hranice mezi dvojicemi tříd jsou lineární.

5.4 Řešené příklady

Příklad – SZZ léto 2022 č. 10: Predikční funkce logistické regrese

(Reálná otázka SZZ: léto 2022, č. 10 – Logistická regrese)

Zadání. (1) Napište funkci, kterou predikuje model logistické regrese. (2) Jakých hodnot tato funkce nabývá a jaká je jejich interpretace?

Řešení. (1) Model spočítá lineární kombinaci příznaků $z = \mathbf{w}^\top \mathbf{x} + b$ a aplikuje na ni logistickou (sigmoidní) funkci $\sigma(z) = \frac{1}{1 + e^{-z}}$, takže predikuje

$$\hat{y} = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}.$$

(2) Hodnoty leží v intervalu $(0, 1)$ a interpretují se jako pravděpodobnost příslušnosti k pozitivní třídě, $P(y=1 | \mathbf{x})$. Vlastní klasifikaci dostaneme prahováním (typicky práh 0,5: třída 1 pro $\hat{y} \geq 0,5$, jinak třída 0).

Příklad – SZZ jaro 2025 č. 29: Logistická regrese pro binární klasifikaci

(Reálná otázka SZZ: jaro 2025, č. 29 – Logistická regrese pro binární klasifikaci)

Zadání. Dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ s cíli $\mathbf{t} = (t_1, \dots, t_N)$. (1) Jak použít logistickou regresi pro binární klasifikaci? Jak vypadá predikce? Jak nezávisle odhadnout chybu na neviděných datech? (2) Jak je definována ztrátová funkce? *Odvoďte její derivaci a ukažte SGD.* (3) Jak při SGD zabráníte overfittingu?

Řešení.

(1) **Model a predikce.** Logistická regrese modeluje $P(C_1 | \mathbf{x}) = y(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x})$ s $\sigma(z) = 1/(1+e^{-z})$ a $P(C_0 | \mathbf{x}) = 1 - y$. *Predikce:* třída 1, je-li $y \geq 0,5$ (ekvivalentně $\mathbf{w}^\top \mathbf{x} \geq 0$), jinak třída 0. *Odhad chyby na neviděných datech:* data rozdělíme na *trénovací* (učení vah), *validační* (ladění hyperparametrů, např. λ , rozhodnutí o early stopping) a *testovací* (jediné finální měření). Při málu dat použijeme k -násobnou křížovou validaci (sekce 1).

(2) **Ztráta, derivace, SGD.** Ztráta je negativní log-věrohodnost (Bernoulli), pro minibatch

$$E(\mathbf{w}) = \frac{1}{N} \sum_i -\log p(C_{t_i} | \mathbf{x}_i; \mathbf{w}), \quad \text{pro 1 vzor } E = -t \log \sigma(z) - (1-t) \log (1 - \sigma(z)), \quad z = \mathbf{w}^\top \mathbf{x}.$$

Derivaci provedeme přes $\sigma'(z) = \sigma(z)(1 - \sigma(z))$:

$$\frac{\partial E}{\partial z} = -t \frac{\sigma'}{\sigma} + (1-t) \frac{\sigma'}{1-\sigma} = -t(1-\sigma) + (1-t)\sigma = -t + t\sigma + \sigma - t\sigma = \sigma(z) - t,$$

a protože $\nabla_{\mathbf{w}} z = \mathbf{x}$, je $\nabla_{\mathbf{w}} E = (\sigma(\mathbf{w}^\top \mathbf{x}) - t)\mathbf{x}$ (plné odvození viz věta výše). *SGD:* inicializuj $\mathbf{w}$, opakuj $\mathbf{w} \leftarrow \mathbf{w} - \alpha(\sigma(\mathbf{w}^\top \mathbf{x}_i) - t_i)\mathbf{x}_i$ (resp. průměr přes minibatch).

(3) **Prevence overfittingu.** L^2 regularizace (člen $\frac{\lambda}{2} \|\mathbf{w}\|^2$ ke ztrátě, ve kroku $-\alpha \lambda \mathbf{w}$), *early stopping* (zastavit, když roste validační chyba) a rozumný learning-rate schedule (sekce 2).

Příklad – SZZ podzim 2024 č. 29: Logistická regrese (model, výstup, multiclass)

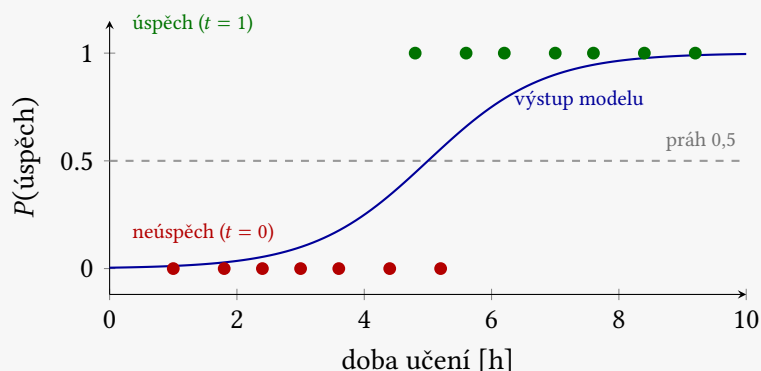
(Reálná otázka SZZ: podzim 2024, č. 29 – Logistická regrese)

Zadání. Model má předpovídat pravděpodobnost úspěchu u zkoušky podle doby učení; data (úspěch 1 / neúspěch 0 vs. doba učení) jsou na obrázku. (1) Popište model logistické regrese: jak se spočítá výstup, jak se trénují parametry, jaká je ztrátová funkce? (2) Zakreslete přibližně výstup natrénovaného modelu (nepočítejte). (3) Jak by se model lišil při klasifikaci do více než dvou tříd (např. predikce známky)?

Řešení.

(1) **Model.** Jeden příznak x = doba učení (plus bias). Výstup $y(x; \mathbf{w}) = \sigma(w_1 x + w_0)$, čtený jako $P(\text{úspěch} | x)$; klasifikace prahem 0,5. *Trénink:* maximum likelihood estimation (MLE) přes SGD, minimalizujeme *negativní log-věrohodnost* (NLL, binární křížovou entropii) $E = -t \log y - (1-t) \log(1-y)$ s gradientem $(y-t)\mathbf{x}$; krok $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{g}$.

(2) **Výstup modelu.** Sigmoidní křivka rostoucí s dobou učení: u krátké doby blízko 0, u dlouhé blízko 1, hodnotu 0,5 (rozhodovací práh) protíná zhruba tam, kde se v datech mísí neúspěchy a úspěchy.



Logistická regrese na 1D datech (doba učení → úspěch). Body leží na $y = 0$ (neúspěchy, krátká doba) a $y = 1$ (úspěchy, dlouhá doba); modrá sigmoida je výstup modelu $P(\text{úspěch} | x)$. Práh 0,5 protíná v pásmu, kde se obě třídy překrývají – to je rozhodovací hranice. Data jsou schematická.

(3) **Více tříd.** Místo jednoho výstupu se sigmoidou bychom měli K výstupů (sada vah na třídu) a sigmoidu nahradili *softmaxem* $P(C_i | \mathbf{x}) = e^{\tilde{y}_i} / \sum_j e^{\tilde{y}_j}$, který dá rozdělení pravděpodobností přes K tříd (sčítá se na 1). Predikce = $\text{argmax}_i P(C_i | \mathbf{x})$, ztráta = kategorická křížová entropie, trénink opět SGD. (Pro predikci *známky* by navíc dávalo smysl využít její *uspořádanost* – ordinální regrese – ale to už je nad rámec.)

6 Rozhodovací stromy

Rozhodovací strom (decision tree) je model, který rozdělí prostor vstupů na *obdélníkové oblasti* (osově zarovnané řezy) a v každé oblasti predikuje jednu hodnotu. Inference je triviální: od kořene následujeme větvičí pravidla v uzlech, až dorazíme do *listu*, jehož hodnotu vrátíme. Stromy zvládají klasifikaci i regresi, mix číselných i kategorických příznaků, nevyžadují škálování dat a jsou dobře *interpretovatelné* – proto jsou populární jako stavební kámen ansámblů (random forest, boosting; viz sekce 8).

i Poznámka. [Vztah k S1] Okruh S1 (sekce *Strojové učení*) už rozhodovací stromy zavádí na úrovni „základní principy“: definice (vnitřní uzly testují atribut, listy dávají rozhodnutí), hladová konstrukce „rozděl a panuj“ a kritérium *informačního zisku* s booleovskou entropií $B(q)$ a příkladem AIMA (restaurace, Patrons vs. Type). Zde to **nezdvojujeme** – entropii a zisk jen krátce zrekapitulujeme s konzistentním značením a jdeme do hloubky, kterou S1 výslovně přenechává S3: *Gini index* jako alternativní kritérium, *regresní stromy* (rozptylové/MSE kritérium), *souvislost kritérií se ztrátovou funkcí* a především **prořezávání** (pruning).

6.1 Algoritmus učení: hladově shora dolů (rozděl a panuj)

Nalézt globálně nejmenší strom konzistentní s daty je NP-těžké; v praxi se proto staví **hladově**: začneme jediným listem se všemi příklady a opakovaně *rozdělíme* list na rozhodovací uzel se dvěma (obecně více) potomky tak, aby rozdělení co nejvíc *zlepšilo homogenitu* skupin. Hladová volba se nikdy nereviduje (greedy). Označme $I_{\mathcal{T}}$ množinu indexů trénovacích příkladů v uzlu $\mathcal{T}$.

Definice – Kritérium uzlu a větvení

Kritérium $c(\mathcal{T})$ měří *nehomogenitu* (nečistotu, impurity) příkladů v uzlu $\mathcal{T}$ – kolik nejistoty o cíli v uzlu zbývá. Při rozdělení uzlu $\mathcal{T}$ na potomky $\mathcal{T}_L$ a $\mathcal{T}_R$ hledáme *příznak* a jeho *práh* (resp. hodnotu), které nejvíc sníží celkové kritérium, tj. minimalizují

$$c(\mathcal{T}_L) + c(\mathcal{T}_R) - c(\mathcal{T}) \leq 0.$$

Kritéria se obvykle volí *vážená velikostí uzlu*, $c(\mathcal{T}) = |I_{\mathcal{T}}| \cdot (\text{nečistota})$, takže rozdíl výše je přímo *informační zisk* (až na znaménko a násobek $|I_{\mathcal{T}}|$).

🔧 Algoritmus – Učení rozhodovacího stromu (greedy, rozděl a panuj)

BUILD-TREE(uzel $\mathcal{T}$ s příklady $I_{\mathcal{T}}$):

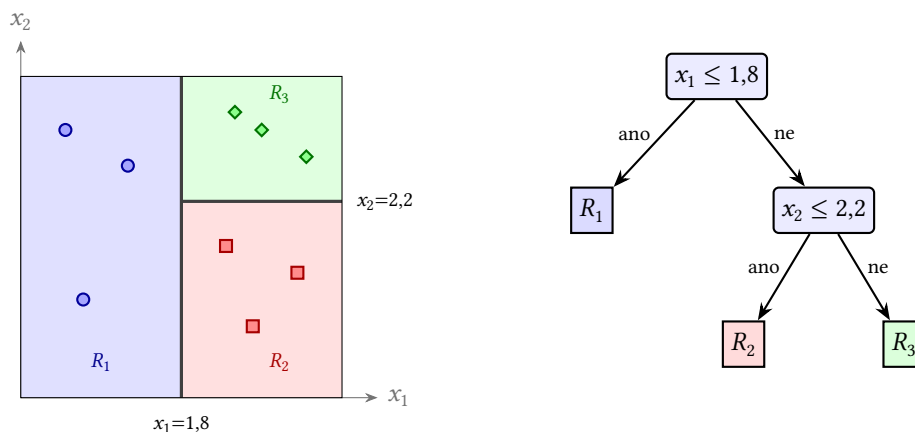
```

if zastavovací podmínka( $\mathcal{T}$ ): return list s predikcí z  $I_{\mathcal{T}}$       ▷ čistý uzel / hloubka / min.
příkladů
  ( $j^*, v^*$ )  $\leftarrow \operatorname{argmin}_{j,v} [c(\mathcal{T}_L^{j,v}) + c(\mathcal{T}_R^{j,v})]$       ▷ for přes příznaky  $j$  a jejich hodnoty  $v$ 
  rozděl  $I_{\mathcal{T}}$  testem „ $x_{j^*} \leq v^*$ “ na  $\mathcal{T}_L, \mathcal{T}_R$ 
  BUILD-TREE( $\mathcal{T}_L$ ); BUILD-TREE( $\mathcal{T}_R$ )
  return rozhodovací uzel s testem ( $j^*, v^*$ ) a podstromy  $\mathcal{T}_L, \mathcal{T}_R$ 

```

💡 **Intuice.** Hledání nejlepšího řezu je dvojitého cyklus: *přes všechny příznaky a přes všechny prahy* (u číselného příznaku stačí hodnoty mezi sousedními seřazenými hodnotami, u kategorického jednotlivé hodnoty). Pro každý kandidát spočítáme kritérium obou potomků a vybereme řez s nejnižším součtem. Predikce listu je *nejčastější třída* (klasifikace), *rozdělení tříd* (pravděpodobnostní výstup) nebo *průměr cílů* (regrese). Rozhodovací uzly tak postupně *dělí prostor osově zarovnanými řezy* na obdélníky (viz obrázek níže).

📌 **Poznámka.** [Zastavovací heuristiky a způsob stavby] Bez omezení strom roste, dokud nejsou listy čisté (overfitting (přeučení) – viz prořezávání). Běžné *heuristiky*: maximální *hloubka*; minimální počet příkladů nutný k dělení uzlu; maximální počet listů. Strom se obvykle staví buď *do hloubky* (rekurzivně dělíme každý list, dokud platí omezení), nebo – je-li dán limit počtu listů – dělíme vždy ten list, který nabízí největší pokles kritéria. Strom *neomezené* velikosti je neparametrický model (roste s daty); s pevným limitem listů je parametrický.



Rozhodovací strom dělí prostor příznaků *osově zarovnanými řezy* na obdélníkové oblasti, každá oblast je jeden list (jedna predikovaná třída). Vlevo dva řezy ($x_1=1,8$ svisle, pak $x_2=2,2$ vodorovně vpravo) rozdělují rovinu na tři oblasti R_1, R_2, R_3 ; vpravo strom, který tyto řezy generuje. Hranice je tedy vždy po částech kolmá na osy – strom neumí šikmou hranici jedním řezem (potřeboval by „schody“).

6.2 Kritéria větvení

Otázka „který řez je nejlepší“ se redukuje na „jak měřit nečistotu uzlu“. Pro *klasifikaci* se používají dvě kritéria – **entropie** (informační zisk) a **Gini index** – pro *regresi* **rozptylové (MSE) kritérium**. Označme $p_{\mathcal{T}}(k)$ podíl příkladů třídy k v uzlu $\mathcal{T}$.

Entropie a informační zisk (rekapitulace z S1)

Definice – Entropie, booleovská entropie, informační zisk

Entropie rozdělení tříd v uzlu (v bitech, $\log = \log_2$) je

$$H(\mathbf{p}_{\mathcal{T}}) = - \sum_{k: p_{\mathcal{T}}(k) > 0} p_{\mathcal{T}}(k) \log_2 p_{\mathcal{T}}(k).$$

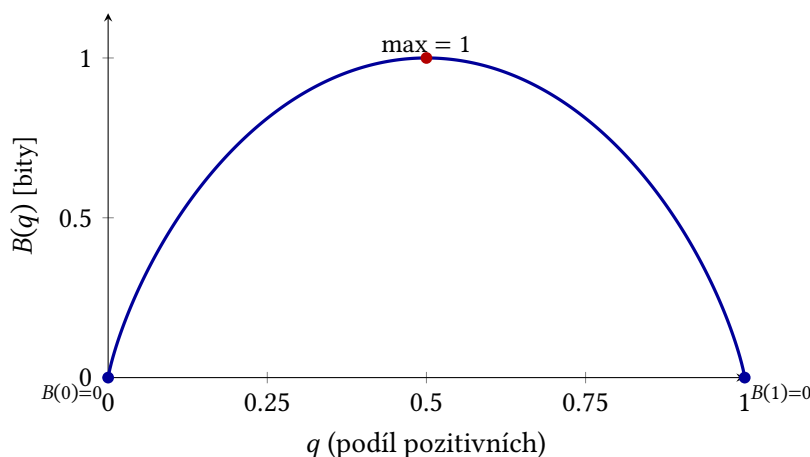
Pro binární cíl (podíl pozitivních q) píšeme *booleovskou entropii*

$$B(q) = -q \log_2 q - (1-q) \log_2 (1-q), \quad B(0) = B(1) = 0, \quad B\left(\frac{1}{2}\right) = 1.$$

Pro množinu s p pozitivními a n negativními příklady je entropie $B\left(\frac{p}{p+n}\right)$. Atribut A rozdělí příklady do skupin; *zbytková* (vážená) entropie je $\text{Remainder}(A) = \sum_k \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$ a **informační zisk**

$$\text{Gain}(A) = B\left(\frac{p}{p+n}\right) - \text{Remainder}(A).$$

V uzlu vybíráme atribut (řez) s *největším* ziskem.



Booleovská entropie $B(q)$: je *konkávní*, nulová v čistém uzlu ($q = 0$ nebo $q = 1$) a maximální (1 bit) při vyrovnaném poměru $q = \frac{1}{2}$. Čím nečistější uzel, tím vyšší entropie; dělení vybíráme tak, aby vážená entropie potomků byla co nejnižší. Hodnota v $q = \frac{1}{3}$ (i $\frac{2}{3}$) je $B\left(\frac{1}{3}\right) \approx 0,918$.

Gini index

Definice – Gini index (Gini impurity)

Gini index uzlu $\mathcal{T}$ měří pravděpodobnost, že náhodně vybraný prvek bude *špatně* označen, kdybychom mu náhodně přiřadili třídu podle rozdělení $\mathbf{p}_{\mathcal{T}}$:

$$\text{Gini}(\mathcal{T}) = \sum_k p_{\mathcal{T}}(k) (1 - p_{\mathcal{T}}(k)) = 1 - \sum_k p_{\mathcal{T}}(k)^2.$$

Pro binární cíl s podílem q pozitivních je $\text{Gini} = 2q(1-q)$ (maximum $\frac{1}{2}$ v $q = \frac{1}{2}$, 0 v $q \in \{0, 1\}$). Stejně jako u entropie hledáme řez minimalizující *vážený* Gini potomků $\sum \frac{|I_{\mathcal{T}_L}|}{|I_{\mathcal{T}}|} \text{Gini}(\mathcal{T}_L)$.

💡 **Intuice.** Gini i entropie se chovají velmi podobně: obě jsou 0 v čistém uzlu, maximální u vyrovnaného a jsou *konkávní* (proto dělení nikdy nezhorší kritérium). Liší se jen mírně tvarem a Gini nepotřebuje logaritmus, je tedy *levnější na výpočet* – proto je výchozí v `scikit-learn`. V praxi vedou skoro vždy na *stejný strom* (jak uvidíme u klasifikačních SZZ otázek níže). Vztah binárních verzí je $Gini = 2q(1 - q)$ vs. $B(q)$; obě křivky mají stejný tvar „kopce“ mezi 0 a 1.

Regresní stromy: rozptylové (MSE) kritérium

Je-li cíl reálné číslo, predikuje list *průměr* cílů svých příkladů $\hat{t}_{\mathcal{J}} = \frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} t_i$. Nečistotu měříme *součtem kvadrátů odchylek* od tohoto průměru.

📖 Definice – Kritérium regresního stromu

$$c_{SE}(\mathcal{J}) = \sum_{i \in \mathcal{J}} (t_i - \hat{t}_{\mathcal{J}})^2, \quad \hat{t}_{\mathcal{J}} = \frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} t_i.$$

Je to $|\mathcal{J}|$ -násobek *rozptylu* cílů v uzlu. Řez vybíráme tak, aby $c_{SE}(\mathcal{T}_L) + c_{SE}(\mathcal{T}_R)$ byl co nejmenší (největší pokles rozptylu).

💡 **Intuice.** [Kritérium plyne ze ztrátové funkce] Není náhoda, že regrese používá kvadratickou chybu a klasifikace entropii/Gini. Vezmeme-li *ztrátovou funkci* modelu a do ní dosadíme optimální parametr listu (ten, který ztrátu minimalizuje), dostaneme „minimální dosažitelnou ztrátu“ daného listu – a právě ta slouží jako kritérium. Pro list:

- dosazení optima do *kvadratické ztráty* dá c_{SE} (regrese);
- dosazení optima do *záporné log-věrohodnosti* (NLL, sekce 1) dá $|\mathcal{J}| \cdot H(\mathbf{p}_{\mathcal{J}})$, tj. *entropické kritérium*;
- dosazení optima do *kvadratické ztráty* na binárním cíli $t \in \{0, 1\}$ dá $|\mathcal{J}| \cdot p_{\mathcal{J}}(1 - p_{\mathcal{J}})$, tj. (až na faktor 2) *Gini*; s dvousložkovým one-hot cílem vyjde přesně $|\mathcal{J}| \cdot Gini$.

Dělením listu chceme tuto minimální dosažitelnou ztrátu snížit, proto je *minimální ztráta uzlu rovnou kritériem větvení*.

6.3 Prořezávání (pruning)

Strom rostlý do čistých listů snadno **overfituje** (přeučí se): dokonale klasifikuje trénovací data (každý šum dostane vlastní list), ale špatně generalizuje. **Prořezávání** (pruning) zjednodušuje strom, aby lépe generalizoval – buď ho rovnou nenechá přerůst, nebo přerostlý strom zpětně zařízne.

📖 Definice – Pre-pruning vs. post-pruning

- **Pre-pruning** (early stopping): zastavíme dělení *už při stavbě*, jakmile přestane být užitečné – omezením hloubky, minimálním počtem příkladů v uzlu, minimálním poklesem kritéria, nebo statistickým testem (χ^2 , sekce 9), zda je zisk významný. Levné, ale *krátkozraké*: může zastavit příliš brzy (řez, který sám nepomůže, by mohl umožnit dobré řezy o úroveň níž).
- **Post-pruning**: necháme strom plně přerůst a pak ho *zpětně* zjednodušujeme – nahrazujeme podstromy listy. Dražší, ale spolehlivější (vidíme celý strom). Dvě hlavní varianty níže.

📖 Definice – Redukce chyby pomocí validačních dat (reduced-error pruning)

Máme samostatnou **validační množinu**. Procházíme uzly (typicky zdola nahoru) a uzel nahradíme *listem* (s většinou třídu jeho trénovacích příkladů) vždy, když to *nezhorší* (nebo *zlepší*) chybu na validačních datech. Opakujeme, dokud lze. Validační data strom při stavbě neviděl; větve, jejichž odstranění validační chybu nezhorší, tedy pravděpodobně zachycovaly spíš šum trénovacích dat než obecnou zákonitost.

Definice – Cost-complexity pruning (weakest-link pruning, CART)

Zavedeme *penalizaci za velikost* stromu. Pro podstrom T definujeme cenu

$$R_\alpha(T) = R(T) + \alpha |\sim T|,$$

kde $R(T)$ je trénovací chyba (např. počet či podíl chybně klasifikovaných příkladů), $|\sim T|$ počet listů a $\alpha \geq 0$ *cena za list*. Pro uzel t a jeho podstrom T_t je *efektivní* α

$$\alpha_{\text{eff}}(t) = \frac{R(t) - R(T_t)}{|\sim T_t| - 1},$$

kde $R(t)$ je chyba, kdyby se t stal listem. Iterativně ořezáváme uzel s *nejmenším* α_{eff} (nejlevnější zjednodušení na list), čímž vzniká *posloupnost* stromů pro rostoucí α ; nejlepší α (kompromis accuracy vs. velikost) se vybere křížovou validací.

 **Intuice.** Cost-complexity pruning je regularizace stromu: α je „pokuta“ za každý list. Při $\alpha = 0$ zůstává plný strom; jak α roste, vyplácí se obětovat listy, které zlepšují trénovací chybu jen málo. α_{eff} je „cena za list ušetřený sloučením“ – ořezáváme tam, kde je nejlevnější. Reduced-error pruning je intuitivnější (přímo se ptá validačních dat), cost-complexity dává hladkou škálu složitosti laditelnou křížovou validací. Obě řeší totéž: *vyměnit kus trénovací accuracy za jednodušší, lépe generalizující strom* (Ockhamova břitva).

 **Poznámka.** [Stromy jako stavební kámen ansámblů] Jeden strom má vysoký *rozptyl* (drobná změna dat může dát úplně jiný strom). Proto se kombinují do ansámblů: *random forest* (bagging stromů na bootstrapových vzorcích s náhodnou podmnožinou příznaků v každém uzlu) a *gradient boosting* (postupné stromy opravující chybu). To je téma sekce 8; zde stromy bereme jako samostatný model.

6.4 Řešené příklady

Následují všechny čtyři reálné SZZ otázky k rozhodovacím stromům (tři klasifikační, jedna regresní), krokované. Postavené stromy jsou překresleny do TikZ; číselné hodnoty (entropie, zisky, struktura) jsme ověřili nezávisle v Pythonu (scikit-learn DecisionTreeClassifier i ruční výpočet).

Příklad – SZZ léto 2023 č. 7: Výběr kořene a mez 100 % accuracy

(Reálná otázka SZZ: léto 2023, č. 7 – Rozhodovací stromy)

Zadání. Trénujeme strom na datech níže se dvěma kategorickými atributy x_1, x_2 a cílem y . (1) Popište algoritmus a kritéria větvení. (2) Podle kterého atributu se větví *kořen*? Proč? (3) Lze data klasifikovat se 100 % accuracy (přesností)? Vysvětlete.

x_1	x_2	y
0	0	0
1	0	1
0	1	1
2	1	0
0	0	1
1	1	1
2	0	0
0	1	0

Řešení. (1) Hladově shora dolů (rozděl a panuj): v každém uzlu zvol atribut s *největším informačním*

ziskem (resp. nejmenším váženým Gini), rozděl příklady podle jeho hodnot a rekurzivně pokračuj, dokud nejsou listy čisté nebo nedojdou atributy (pak většinová třída).

(2) Z 8 příkladů je $4 \times y=0$ a $4 \times y=1$, takže entropie kořene je $B(\frac{4}{8}) = B(\frac{1}{2}) = 1$ bit. Spočteme zisk obou atributů.

Atribut x_1 (hodnoty 0, 1, 2) rozdělí příklady:

$$x_1=0 : (2, 2) \rightarrow B(\frac{1}{2}) = 1, \quad x_1=1 : (0, 2) \rightarrow 0, \quad x_1=2 : (2, 0) \rightarrow 0.$$

(Zde (a, b) značí a příkladů s $y=0$ a b s $y=1$.) Zbytek a zisk:

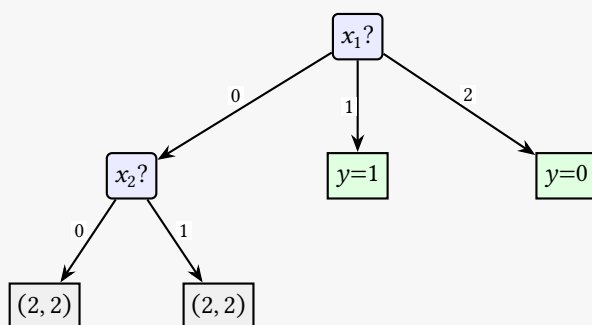
$$\text{Remainder}(x_1) = \frac{4}{8} \cdot 1 + \frac{2}{8} \cdot 0 + \frac{2}{8} \cdot 0 = \frac{1}{2}, \quad \text{Gain}(x_1) = 1 - \frac{1}{2} = 0,5.$$

Atribut x_2 (hodnoty 0, 1): obě skupiny $x_2=0$ i $x_2=1$ mají poměr (2, 2), tedy $B(\frac{1}{2}) = 1$:

$$\text{Remainder}(x_2) = \frac{4}{8} \cdot 1 + \frac{4}{8} \cdot 1 = 1, \quad \text{Gain}(x_2) = 1 - 1 = 0.$$

Kořen se větví podle x_1 (vyšší zisk; x_2 samo o sobě nerozliší vůbec nic). Větev $x_1=1$ a $x_1=2$ jsou rovnou čisté listy.

(3) **Ne.** V datech jsou *shodné vstupy s různým cílem*: řádky $(x_1, x_2) = (0, 0)$ se objeví s $y=0$ i $y=1$, stejně $(0, 1)$ s $y=1$ i $y=0$. Žádný model rozhodující jen podle (x_1, x_2) tyto dva páry nerozliší – ať strom roste jakkoli hluboko, ve větvi $x_1=0$ zbudou čtyři příklady $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 1)$, $(0, 1, 0)$, které mají jen dvě různé kombinace vstupů, ale obě třídy. Maximální dosažitelná accuracy je tedy $< 100\%$ (konkrétně $6/8$: ve větvi $x_1=0$ predikuje list většinu, dvě protichůdná pozorování zůstanou špatně).



Strom pro léto 2023: kořen x_1 . Větev $x_1=1$ a $x_1=2$ jsou čisté listy. Větev $x_1=0$ se sice dá ještě dělit podle x_2 , ale obě podskupiny zůstanou poměr (2, 2) – nerozdělitelný šum se shodnými vstupy a různými cíli. Proto 100 % accuracy není dosažitelná.

Příklad – SZZ podzim 2024 č. 31: Stavba stromu, kořen A, prořezávání validačními daty

(Reálná otázka SZZ: podzim 2024, č. 31 – Rozhodovací strom)

Zadání. Data se třemi atributy A, B, C a cílem $Y \in \{n, y\}$. (1) Postavte plný strom (bez omezení). (2) Pro atribut v kořeni popište výpočet kritéria. (3) Proveďte prořezávání validačními daty.

Trénovací:

	A	B	C	Y
0	1	2	6	n
1	1	2	6	n
2	3	2	8	y
3	3	4	6	y
4	3	4	8	y
5	3	4	8	n
6	3	4	8	n
7	3	4	6	y

Validační:

	A	B	C	Y
8	1	2	8	n
9	3	4	6	y
10	3	4	6	y
11	3	4	8	y

Řešení. (1)+(2) **Kořen.** Z 8 příkladů jsou $4 \times n$ a $4 \times y$, entropie kořene $B(\frac{1}{2}) = 1$. Spočteme zisk řezů (kontrola jednotlivých prahů):

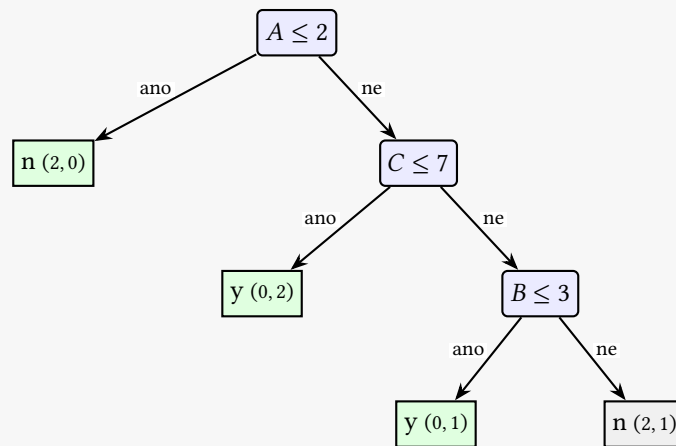
$$A \leq 2 \text{ (tj. } A=1 \text{ vs. } A=3) : (2, 0) | (2, 4) \Rightarrow \text{Remainder} = \frac{2}{8} \cdot 0 + \frac{6}{8} B(\frac{2}{6}) = \frac{3}{4} \cdot 0,918 \approx 0,689,$$

$$\text{Gain} \approx 1 - 0,689 = \boxed{0,311};$$

$$B \leq 2 : (2, 1) | (2, 3) \Rightarrow \text{Gain} \approx 0,049;$$

$$C \leq 6 : (2, 2) | (2, 2) \Rightarrow \text{Gain} = 0.$$

Největší zisk má $\boxed{A}$ (oficiální náčrt uvádí $\approx 0,32$ kvůli zaokrouhlení tabulky logaritmu). Větev $A \leq 2$ je čistý list (n). V pravé větvi $A > 2$ (6 příkladů, 2 n, 4 y, $B(\frac{2}{6}) \approx 0,918$) dělíme dál podle $C \leq 7$: vlevo (0, 2) čistý list y, vpravo $C > 7$ má (2, 2), kde rozdělí $B \leq 3$ na čistý list y (0, 1) a list (2, 1). Strom vyjde *stejný pro entropii i Gini* (ověřeno).



Plný strom (podzim 2024), totožný pro entropii i Gini. Dvojice (n, y) udává počty tříd v uzlu. Kořen A, pak v pravé větvi C, pak B. List (2, 1) není čistý (zbylé příklady mají shodné A, B, C, ale různé Y), predikuje většinu n.

(3) **Prořezávání validačními daty (reduced-error pruning).** Spustíme 4 validační příklady plným stromem:

id	A	B	C	Y	plný strom	po ořezu $A > 2 \rightarrow y$
8	1	2	8	n	n ✓	n ✓
9	3	4	6	y	y ✓	y ✓
10	3	4	6	y	y ✓	y ✓
11	3	4	8	y	n ✗	y ✓

Plný strom dělá na validaci 1 chybu (id 11: prochází $A > 2, C > 7, B > 3$ do listu n, ale skutečnost je y). Nahradíme-li celý *pravý podstrom* (uzel C, tj. vše pod $A > 2$) *listem* s většinovou trénovací třídou y (4 y vs. 2 n), validační chyba klesne na $\boxed{0}$. Prořezání tedy chybu *zlepšilo* – ořez přijmeme. (Postupem

zdola nahoru dojdeme ke stejnému stromu: nejdřív uzel $B \leq 3$, jehož trénovací příklady mají poměr $(2, 2)$ – ať remízu rozhodneme jakkoli, validační chyba se nezhorší, takže se ořízne; pak uzel $C \leq 7$ nahrazený listem y dá validační chybu 0, takže se ořízne také.) Strom se zmenší na $A \leq 2 \rightarrow n$, jinak y . (Alternativně lze použít cost-complexity pruning.)

Příklad – SZZ jaro 2026 č. 30: Zadaný kořen D , kritérium pravého podstromu, prořezávání

(Reálná otázka SZZ: jaro 2026, č. 30 – Rozhodovací strom)

Zadání. (a) Postavte strom, do kořene zvolte atribut D ; volby zdůvodněte, pro *pravý podstrom* D spočítejte kritérium pro všechny atributy. (b) Vysvětlete prořezávání, uveďte neprořezaný a prořezaný strom a kritérium prořezávání. Sloupec *id* se k učení nepoužívá.

id	A	B	C	D	Y
0	3	4	8	6	0
1	1	2	8	6	0
2	1	2	8	3	1
3	3	2	8	6	1
4	1	4	6	6	1
5	3	2	8	6	0
6	3	4	6	3	0
7	3	4	8	3	1

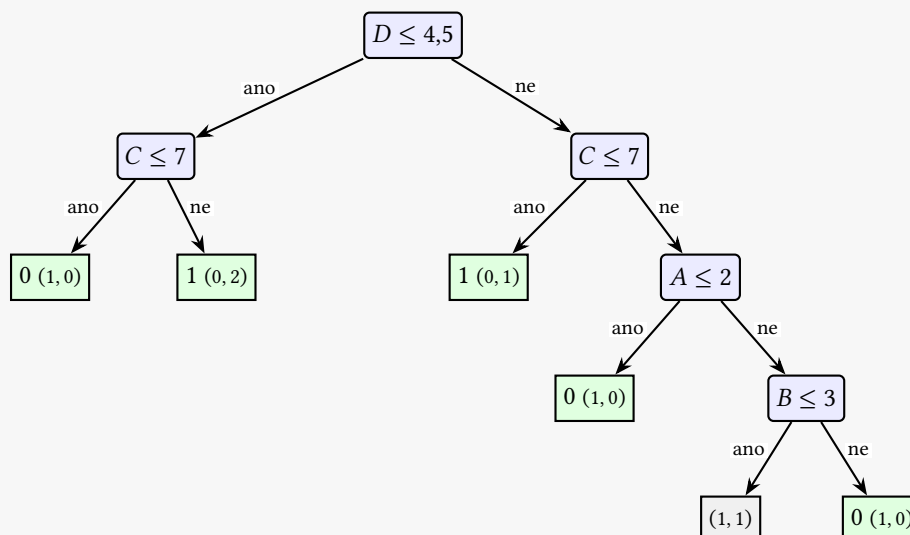
Řešení. (a) Kořen je zadán: $D \leq 4,5$ (tj. $D=3$ vlevo, $D=6$ vpravo). Kořen má $4 \times 0, 4 \times 1$, entropie 1. *Levý podstrom* $D=3$ (id 2, 6, 7, cíle 1, 0, 1, tj. poměr $(1, 2)$): dělení podle $C \leq 7$ dá vlevo $C=6$ jediný příklad id 6 (list 0), vpravo $C=8$ příklady id 2, 7 (oba $Y=1$, čistý list 1). Levý podstrom je tedy hotový. *Pravý podstrom* $D=6$ (id 0, 1, 3, 4, 5, cíle 0, 0, 1, 1, 0, poměr $(3, 2)$, $B(\frac{2}{5}) \approx 0,971$) – spočítáme *vážené entropie po dělení* pro všechny atributy:

$$A \leq 2 : (1, 1)_{n_2} | (2, 1)_{n_3} \Rightarrow \frac{2}{5} \cdot 1 + \frac{3}{5} B(\frac{1}{3}) \approx 0,4 + 0,6 \cdot 0,918 \approx \boxed{0,951};$$

$$B \leq 3 : (2, 1)_{n_3} | (1, 1)_{n_2} \Rightarrow \frac{3}{5} B(\frac{1}{3}) + \frac{2}{5} \cdot 1 \approx \boxed{0,951};$$

$$C \leq 7 : (0, 1)_{n_1} | (3, 1)_{n_4} \Rightarrow \frac{1}{5} \cdot 0 + \frac{4}{5} B(\frac{1}{4}) \approx \frac{4}{5} \cdot 0,811 \approx \boxed{0,649}.$$

Nejnižší vážená entropie (a tedy *největší zisk*, $\approx 0,971 - 0,649 = 0,322$) má **C**. Vlevo $C=6$ čistý list 1 (id 4), vpravo $C=8$ má id 0, 1, 3, 5, poměr $(3, 1)$; dělí se podle $A \leq 2$ (id 1 list 0; stejný zisk by dal i $B \leq 3$) a $A > 2$ (id 0, 3, 5, poměr $(2, 1)$), kde řez $B \leq 3$ dá id 3, 5 (poměr $(1, 1)$, nerozdělitelný šum) a $B > 3$ čistý list 0 (id 0). Strom dává stejné predikce pro entropii i Gini (ověřeno); nakreslené pořadí uzlů odpovídá entropii.



Neprořezaný strom do maximální hloubky (jaro 2026), shodný pro entropii i Gini. Dvojice (n_0, n_1) udává počty tříd. Pravá větev $D=6$ se dělí podle C , pak A , pak B ; list $(1, 1)$ je nerozdělitelný (shodné vstupy id 3 a 5 s různým Y).

(b) **Prořezávání** odstraňuje overfitting: na trénovacích datech je chyba malá, ale na nových větší než u jednoduššího stromu. Nahradíme uzel $A \leq 2$ (celý pravý spodní podstrom) *listem*: jeho trénovací příklady id 0, 1, 3, 5 mají poměr $(3, 1)$, většina je 0. Plný podstrom stejně jeden příklad (id 3 vs. 5) klasifikuje špatně (list $(1, 1)$), takže náhrada listem 0 chybu *nezhorší* a strom zjednoduší (méně listů). Prořezaný strom: $D \leq 4,5 \rightarrow (C \leq 7? 0 : 1)$; jinak $C \leq 7 \rightarrow 1, C > 7 \rightarrow 0$.

$$\text{Kritérium ořezu (cost-complexity): } \alpha_{\text{eff}}(t) = \frac{R(t) - R(T_t)}{|\sim T_t| - 1},$$

ořezává se uzel s nejmenším α_{eff} , dokud je pod prahem; $R(t)$ je chyba uzlu jako listu, $R(T_t)$ chyba podstromu, $|\sim T_t|$ počet jeho listů. (Alternativně reduced-error pruning podle validačních dat.)

Příklad – SZZ podzim 2022 č. 19: Regresní strom – výběr atributu

(Reálná otázka SZZ: podzim 2022, č. 19 – Rozhodovací stromy)

Zadání. (1) Definujte rozhodovací strom a popište algoritmus jeho trénování. (2) Uveďte kritérium pro výběr atributu pro *klasifikaci* a pro *regresi*. (3) Pro data níže (kategorické atributy A_1, A_2 , číselný cíl C) určete, podle kterého atributu se dělí kořen pro *regresi*, a proč.

A_1	A_2	C
0	0	5
0	0	3
0	1	4
1	0	6
1	1	2
1	1	4

Řešení. (1) Rozhodovací strom je stromový model: vnitřní uzly testují atribut, větve odpovídají jeho hodnotám a listy nesou predikci (třidu nebo číslo). Trénuje se hladově shora dolů (rozděl a panuj): v každém uzlu se vybere atribut podle zvoleného kritéria, data se podle něj rozdělí a rekurzivně se pokračuje, dokud nejsou listy čisté, nedojdou atributy nebo není splněna zastavovací podmínka.

(2) *Klasifikace*: maximalizace informačního zisku (pokles entropie) nebo minimalizace Gini indexu. *Regrese*: minimalizace vážené sumy čtvercových odchylek (SSE) v potomcích, ekvivalentně maximalizace redukce rozptylu.

(3) Celkový průměr je $\bar{C} = 4$, SSE rodiče = $\sum (C_i - 4)^2 = 1 + 1 + 0 + 4 + 4 + 0 = 10$.

- Dělení podle A_1 : $A_1=0 \rightarrow \{5, 3, 4\}$, průměr 4, SSE 2; $A_1=1 \rightarrow \{6, 2, 4\}$, průměr 4, SSE 8. Celkem $2 + 8 = 10$, redukce 0.
- Dělení podle A_2 : $A_2=0 \rightarrow \{5, 3, 6\}$, průměr $\approx 4,67$, SSE $\approx 4,67$; $A_2=1 \rightarrow \{4, 2, 4\}$, průměr $\approx 3,33$, SSE $\approx 2,67$. Celkem $\approx 7,33$, redukce $\approx 2,67$.

V kořeni proto dělíme podle A_2 (větší redukce rozptylu). Atribut A_1 nedává žádnou redukci, protože obě jeho větve mají stejný průměr 4 jako rodič.

i Poznámka. [Co si odnést – rozhodovací stromy]

- *Inference* = sleduj testy do listu. *Učení* = hladově shora dolů (rozděl a panuj): v každém uzlu řez (příznak + práh) minimalizující kritérium potomků.
- *Kritéria*: klasifikace – **entropie**/informační zisk $\text{Gain} = B(\frac{p}{p+n}) - \text{Remainder}(A)$ nebo **Gini** $= 1 - \sum_k p_k^2$ (levnější, prakticky stejný strom); regrese – **rozptyl/MSE** $c_{SE} = \sum (t_i - \hat{t})^2$. Kritérium plyne z minimální dosažitelné ztráty listu.
- *Prořezávání*: **pre-pruning** (zastav brzy: hloubka, min. příkladů, χ^2) vs. **post-pruning** (přerost a zařízní): **reduced-error** (ořež, co nezhorší validační chybu) a **cost-complexity** (penalizace α za list, ořež podle nejmenšího α_{eff} , α ladí křížová validace).
- *Mez stromu*: shodné vstupy s různým cílem nelze rozdělit (léto 2023: 100 % accuracy nedosažitelná). Jeden strom má vysoký rozptyl $\Rightarrow$ ansámby (sekce 8).

7 Metoda podpurných vektorů (SVM)

Metoda podpurných vektorů (support vector machines, SVM) je lineární klasifikátor, který mezi všemi oddělovacími nadrovinami vybere tu s *největším marginem* (margin, odstup od dat). Ve spojení s *kernel trikem* zvládne i nelineárně oddělitelná data a patří k nejúspěšnějším klasifikátorům „z krabice“. Tato sekce jde do hloubky, kterou S1 (sekce *Strojové učení*) výslovně přenechává S3.

i Poznámka. [Vztah k S1] Okruh S1 zavádí SVM *přehledově*: maximální margin, podpurné vektory a jádrovou transformaci jako hlavní myšlenku, s dvoupanelovým obrázkem. Zde to nezdujujeme. Stavíme na tom a jdeme do detailu: jak přesně vypadá optimalizační úloha (QP), proč závisí jen na skalárních součinech, soft margin a hinge loss, konkrétní jádra a klasifikace do více tříd.

7.1 Lineárně separabilní třídy: maximální margin

Mějme binární klasifikaci s daty $(\mathbf{x}_i, t_i)$, kde cíle značíme **symetricky** $t_i \in \{-1, +1\}$ (ne $\{0, 1\}$ jako u logistické regrese – u SVM se to vyplatí). Lineární model je

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \quad \text{predikce} = \text{sign } y(\mathbf{x}).$$

Rovnice $\mathbf{w}^\top \mathbf{x} + b = 0$ určuje *oddělovací nadrovinu* (v rovině přímku); vektor $\mathbf{w}$ je na ni kolmý. Předpokládejme, že data jsou **lineárně separabilní**, tj. existuje nadrovina, která bezchybně oddělí obě třídy. Takových nadrovin je obecně nekonečně mnoho; SVM mezi nimi volí jednu konkrétní.

Definice – Margin, podpurné vektory

Margin (margin, šíře odstupů) oddělovací nadroviny je vzdálenost od nadroviny k nejbližšímu trénovacímu bodu (na obě strany). **Maximálně marginový (max-margin) klasifikátor** je nadrovina s *největším možným marginem*. Trénovací body ležící *nejblíže* k nadrovině (přesně na marginu) se nazývají **podpurné vektory** (support vectors); právě ony nadrovinu určují – ostatní body lze odebrat, aniž se hranice změní.

💡 **Intuice.** Mezi mnoha přímkami, které data oddělí, vybere SVM tu „nejodvážnější“ – s co největší rezervou na obě strany. Široký margin zlepšuje *generalizaci*: malé posuny dat hranici nepřekročí, takže model je robustní vůči šumu. Tomuto principu se říká *maximalizace marginu* a lze ho chápat jako jistou formu regularizace.

Od marginu k optimalizační úloze

Vzdálenost bodu $\mathbf{x}_i$ od nadroviny je $\frac{|y(\mathbf{x}_i)|}{\|\mathbf{w}\|}$. Klasifikuje-li model všechny body správně, je $t_i y(\mathbf{x}_i) > 0$, takže $|y(\mathbf{x}_i)| = t_i y(\mathbf{x}_i)$ a vzdálenost je $\frac{t_i y(\mathbf{x}_i)}{\|\mathbf{w}\|}$. Chceme maximalizovat margin, tj. vzdálenost *nejbližšího* bodu:

$$\operatorname{argmax}_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|} \min_i [t_i(\mathbf{w}^\top \mathbf{x}_i + b)].$$

Takto se úloha optimalizuje špatně. Pomůže pozorování, že model je *invariantní vůči přeskálování*: vynásobíme-li $\mathbf{w}$ i b stejnou konstantou, nadrovina (ani znaménko y) se nezmění. Můžeme proto škálu „ukotvit“ tak, aby pro nejbližší body platilo $t_i y(\mathbf{x}_i) = 1$. Pak pro *všechny* body platí $t_i y(\mathbf{x}_i) \geq 1$ a maximalizace $1/\|\mathbf{w}\|$ je totéž co minimalizace $\frac{1}{2}\|\mathbf{w}\|^2$.

★ Věta – Primární úloha hard-margin SVM (kvadratické programování)

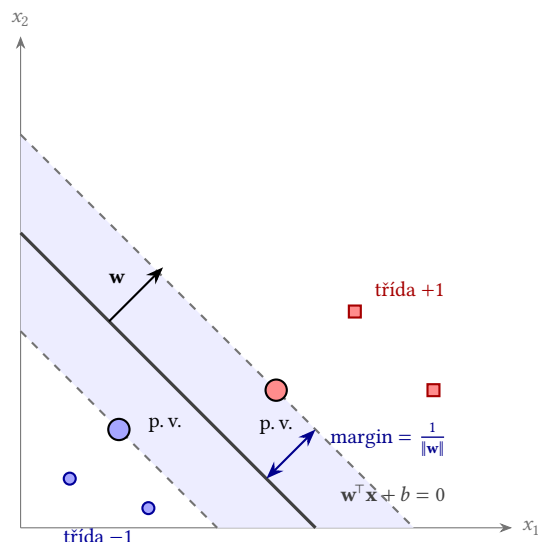
Pro lineárně separabilní data je max-margin nadrovina řešením úlohy

$$\min_{\mathbf{w}, b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{za podmínky} \quad t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \text{ pro všechna } i.$$

Účelová funkce je *kvadratická* a podmínky *lineární* (afinní) v $\mathbf{w}, b$, jde tedy o úlohu **kvadratického programování** (QP). Účelová funkce je navíc *konvexní*, takže optimum je jednoznačné (globální). Při této normalizaci je margin roven

$$\text{margin} = \frac{1}{\|\mathbf{w}\|}, \quad \text{tj. celková šíře pásu mezi marginy} = \frac{2}{\|\mathbf{w}\|}.$$

💡 **Intuice.** [Proč $2/\|\mathbf{w}\|$] Body s $t_i y(\mathbf{x}_i) = 1$ leží na dvou rovnoběžných nadrovinách $\mathbf{w}^\top \mathbf{x} + b = +1$ a $\mathbf{w}^\top \mathbf{x} + b = -1$ (kladný a záporný margin). Vzdálenost mezi dvěma rovnoběžkami $\mathbf{w}^\top \mathbf{x} + b = \pm 1$ je $\frac{2}{\|\mathbf{w}\|}$. Minimalizovat $\|\mathbf{w}\|$ tedy znamená *rozšiřovat* pás mezi marginy.



Max-margin separátor: plná čára je rozhodovací hranice $\mathbf{w}^T \mathbf{x} + b = 0$, čárkované čáry jsou dva marginy $\mathbf{w}^T \mathbf{x} + b = \pm 1$. Zakroužkované body leží přesně na marginu – to jsou *podpůrné vektory* (p. v.), které hranici určují. Vektor $\mathbf{w}$ je na hranici kolmý; šíře pásu je $2/\|\mathbf{w}\|$.

Duální tvar a role skalárních součinů

Úloha s nerovnostními podmínkami se řeší přes *Lagrangeovy multiplikátory* $a_i \geq 0$ (jeden na každou podmínku; podmínky optimality KKT – Karush–Kuhn–Tucker). Plnou derivaci vynecháváme; klíčové je, jak vypadá výsledek. Po dosazení optimálních $\mathbf{w} = \sum_i a_i t_i \mathbf{x}_i$ vznikne ekvivalentní **duální úloha**

$$\max_{\mathbf{a}} \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j t_i t_j \underbrace{\mathbf{x}_i^T \mathbf{x}_j}_{\text{skalární součin}} \quad \text{za podmíněk} \quad a_i \geq 0, \quad \sum_i a_i t_i = 0.$$

💡 **Intuice.** [Dvě zásadní pozorování]

- **Závislost jen na skalárních součinech.** V duální úloze se data objevují *výhradně* přes skalární součiny $\mathbf{x}_i^T \mathbf{x}_j$. To je brána ke kernel triku (níže): stačí umět počítat skalární součiny, samotné vektory nepotřebujeme.
- **Řídkost: jen podpůrné vektory.** Z podmínek KKT plyne $a_i (t_i y(\mathbf{x}_i) - 1) = 0$ pro každé i . Tedy buď je bod přesně na marginu ($t_i y(\mathbf{x}_i) = 1$), nebo $a_i = 0$. Body s $a_i = 0$ se v predikci $y(\mathbf{x}) = \sum_i a_i t_i \mathbf{x}_i^T \mathbf{x} + b$ neuplatní. Stačí tedy uchovat jen *podpůrné vektory* (body s $a_i > 0$). SVM je sice neparametrický model (počet podpůrných vektorů roste s daty), ale paměťově úsporný.

7.2 Lineárně neseparabilní třídy: soft margin

Reálná data bývají *zašuměná* a oddělit je beze zbytku přímkou nelze (hard-margin úloha pak nemá řešení). **Soft margin** (měkký margin) povolí, aby některé body byly *uvnitř* marginu, nebo i na *špatné straně* hranice, ale za pokutu.

Definice – Slack proměnné, soft-margin SVM

Pro každý bod zavedeme **slack proměnnou** $\xi_i \geq 0$ měřící porušení marginu. Podmínku $t_i y(\mathbf{x}_i) \geq 1$ uvolníme na $t_i y(\mathbf{x}_i) \geq 1 - \xi_i$. Optimalizujeme

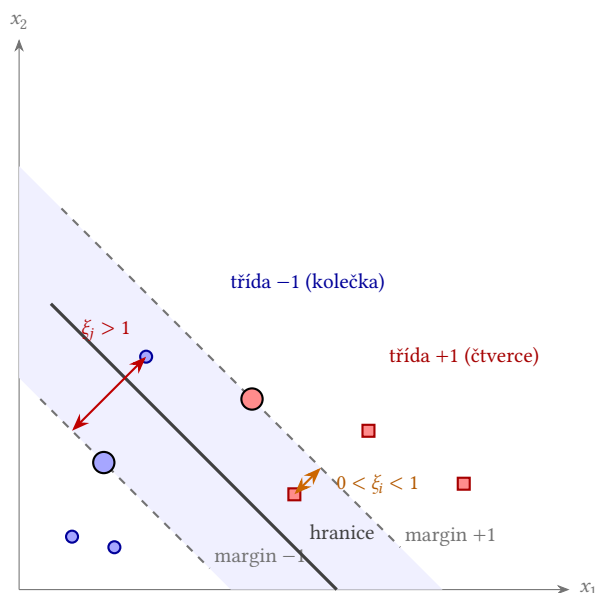
$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \quad \text{za podmíněk} \quad t_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Hodnota slacku rozlišuje polohu bodu: $\xi_i = 0$ bod *vně* marginu (správně, s rezervou), $0 < \xi_i < 1$ bod *uvnitř* marginu (správně, ale bez rezervy), $\xi_i = 1$ bod *na* hranici a $\xi_i > 1$ bod *na špatné straně* hranice (chyba klasifikace).

💡 **Intuice.** [Význam konstanty C] Konstanta $C > 0$ je *kompromis* mezi šíří marginu a chybami: penalizuje součet slacků $\sum_i \xi_i$.

- **Velké C** – chyby jsou drahé, model se snaží klasifikovat vše správně (úzký margin, blíž k hard-margin; riziko *overfittingu*, tj. přeučení).
- **Malé C** – chyby jsou levné, model upřednostní široký margin a toleruje porušení (více *regularizace*, hladší hranice).

C je hyperparametr; ladí se křížovou validací (viz sekce 1).



Soft margin: většina bodů leží správně vně marginu ($\xi = 0$, zakroužkované jsou podpůrné vektory přesně na marginu). Oranžová úsečka – kladný bod uvnitř pásu na správné straně hranice ($0 < \xi_i < 1$). Červená úsečka – bod záporné třídy zabloudil na kladnou stranu hranice ($\xi_j > 1$, chyba). Slack ξ_i je (škálovaná) kolmá vzdálenost od příslušného marginu k bodu; jeho součet $\sum_i \xi_i$ pokutuje konstanta C .

Hinge loss: SVM jako regularizovaná ztráta

Z podmínek lze slack vyjádřit explicitně: $\xi_i = \max(0, 1 - t_i y(\mathbf{x}_i))$ (je-li bod správně s rezervou, je $1 - t_i y \leq 0$ a slack je 0; jinak je kladný). Dosazením dostaneme *bezpodmínkovou* formulaci přes **hinge loss** (závěsovou ztrátu):

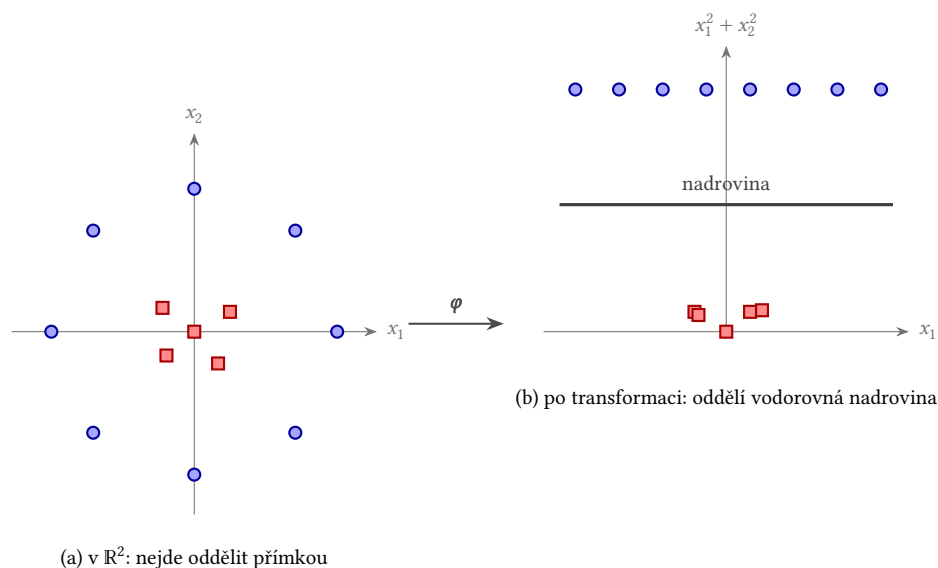
$$\mathcal{L}_{\text{hinge}}(t, y) = \max(0, 1 - t y), \quad \min_{\mathbf{w}, b} \underbrace{C \sum_i \mathcal{L}_{\text{hinge}}(t_i, y(\mathbf{x}_i))}_{\text{datová ztráta}} + \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{\text{regularizace}}.$$

💡 **Intuice.** Hinge loss penalizuje bod, dokud není správně klasifikován s rezervou ($t_i y \geq 1$); za touto hranicí je ztráta nulová (proto „závěs“). To je rozdíl oproti logistické regresi (sekce 5), jejíž ztráta $-\log \sigma(ty)$ klesá k nule jen asymptoticky a odměňuje i body daleko za hranicí. Formulace ukazuje, že C je **inverzní síla regularizace**: $C \rightarrow \infty$ znamená žádnou regularizaci (hard-margin), $C \rightarrow 0$ ignoruje data. SVM tak zapadá do téhož rámce „datová ztráta + regularizace“ jako lineární a logistická regrese; liší se jen volbou ztrátové funkce.

7.3 Jádrové funkce: kernel trick

Lineární SVM zvládne jen lineárně (téměř) oddělitelná data. Trik, jak ho zmocnit pro nelineární hranice: data nejprve *transformovat* zobrazením $\phi: \mathbb{R}^D \rightarrow \mathbb{R}^F$ do prostoru vyšší *dimenze*, kde se stanou lineárně separabilní. Lineární hranice v novém prostoru odpovídá *zakřivené* hranici v původním.

Intuice. [Proč vyšší dimenze pomáhá] Body, které v rovině nejdou oddělit přímkou (např. jedna třída uvnitř kruhu, druhá kolem), se přidáním vhodných příznaků (např. x_1^2 , x_2^2 , $x_1 x_2$) v prostoru vyšší dimenze „rozestoupí“ tak, že je oddělí nadrovinou. Vzdálenost od počátku ($x_1^2 + x_2^2$) je nový příznak, podle nějž se vnitřní a vnější kruh liší.



Kernel trik. (a) Vnitřní třída (červené čtverce, malý shluk kolem počátku) a vnější třída (modrá kolečka rovnoměrně na kružnici) nejdou v rovině oddělit přímkou. (b) Po transformaci do prostoru vyšší dimenze, kde je novým příznakem vzdálenost od počátku $x_1^2 + x_2^2$, se třídy rozdělí a oddělí obyčejnou (vodorovnou) nadrovinou; ta v původní rovině odpovídá kružnici.

Transformace do vysoké (i nekonečné) dimenze by byla výpočetně drahá. Tady se uplatní, že duální SVM závisí na datech *jen přes skalární součiny*: stačí umět spočítat $\varphi(\mathbf{x})^\top \varphi(\mathbf{x}')$, transformaci samotnou nikdy nepotřebujeme.

Definice – Jádro, kernel trick

Jádro (kernel) odpovídající transformaci φ je funkce

$$K(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \varphi(\mathbf{x})^\top \varphi(\mathbf{x}').$$

Kernel trik: v duální úloze i v predikci $y(\mathbf{x}) = \sum_i a_i t_i K(\mathbf{x}, \mathbf{x}_i) + b$ všude nahradíme skalární součin $\mathbf{x}_i^\top \mathbf{x}_j$ jádrem $K(\mathbf{x}_i, \mathbf{x}_j)$. Pracujeme tak ve vysokodimenzionálním prostoru, *aniž bychom φ kdy explicitně vyčíslili*.

Definice – Nejčastější jádra

- **Polynomiální jádro** stupně (nejvýše) d :

$$K(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x}^\top \mathbf{x}' + 1)^d.$$

Odpovídá transformaci na všechny součiny až d vstupních příznaků (interakce do stupně d). Pro $d = 2$ a $\gamma = 1$ je $\varphi(x_1, x_2) = (1, \sqrt{2} x_1, \sqrt{2} x_2, x_1^2, \sqrt{2} x_1 x_2, x_2^2)$.

- **Gaussovo (RBF) jádro** (radial basis function):

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2), \quad \gamma > 0.$$

Odpovídá skalárnímu součinu v prostoru *nekonečné* dimenze (kombinaci polynomiálních jader všech stupňů).

Intuice. [Kdy jádra pomohou a jak se chová RBF] Jádra pomohou, když hranice mezi třídami *není lineární*. RBF jádro je funkcí *vzdálenosti*: $K(\mathbf{x}, \mathbf{x}')$ je velké pro blízké body a klesá k nule pro vzdálené. SVM s RBF tak vlastně váží „svědectví“ trénovacích bodů podle vzdálenosti – je to chytřejší varianta k -NN (sekce 3), která bere v úvahu všechny body, každý vážený podobností. Parametr γ řídí *dosah*: velké γ = úzké špičky kolem bodů (riziko *overfittingu*), malé γ = široký vliv (hladší hranice). Na *lineárně oddělitelných* datech jádro nepomůže – naopak může vést k *overfittingu*; tam stačí lineární SVM.

7.4 Klasifikace do více tříd

SVM je ze své podstaty *binární*. Pro $K > 2$ tříd se skládá z více binárních klasifikátorů dvěma standardními způsoby.

Definice – One-vs-rest a one-vs-one

- **One-versus-rest** (jeden proti zbytku, OvR): natrénuje se K binárních klasifikátorů; i -tý odděluje třídu i od *všech ostatních*. Při predikci vybereme třídu, jejíž klasifikátor dá nejvyšší skóre (vyžaduje *srovnatelná* skóre, ideálně kalibrované pravděpodobnosti – což SVM přímo nedává).
- **One-versus-one** (každý s každým, OvO): natrénuje se $\binom{K}{2}$ klasifikátorů, jeden pro každou dvojici tříd (i, j) . Při predikci každý hlasuje pro jednu ze své dvojice a vítězí třída s *nejvíce hlasy*. (Tuto variantu používá pro SVM např. knihovna `libsvm/scikit-learn`.)

Intuice. [Kompromis OvR vs. OvO] OvR trénuje jen K klasifikátorů (méně modelů), ale každý na *nevyvážených* datech (jedna třída proti $K - 1$) a vyžaduje porovnatelná skóre. OvO trénuje $\binom{K}{2} = \frac{K(K-1)}{2}$ klasifikátorů (kvadraticky mnoho), zato každý na malé, vyvážené dvojici tříd; proto se hodí pro SVM, jehož trénink škáluje nadlineárně s počtem bodů. Nevýhoda hlasování: vznikají *nerozhodné* oblasti, kde víc tříd dostane stejně hlasů.

Poznámka. [Nad rámec požadavků] Jen pro úplnost a bez detailu: duální úlohu SVM v praxi řeší algoritmus **SMO** (Sequential Minimal Optimization), což je souřadnicový sestup optimalizující vždy dvojici multiplikátorů a_i, a_j najednou (kvůli podmínce $\sum_i a_i t_i = 0$). SVM lze upravit i na *regresi* (SVR) s ε -necitlivou ztrátou. Tyto detaily požadavky nevyžadují.

7.5 Řešený příklad

Příklad – SZZ léto 2024 č. 30: Metoda podpůrných vektorů (SVM)

(Reálná otázka SZZ: léto 2024, č. 30 – Metoda podpůrných vektorů (SVM))

Zadání. Data se dvěma příznaky, dvě třídy. Třída A (křížky, $t = +1$): (1, 1), (2, 2), (2, 0). Třída B (kolečka, $t = -1$): (0, 0), (1, 0), (0, 1). (1) Popište, jak fungují a trénují se lineární SVM pro lineárně separabilní data. (2) Zakreslete přibližnou rozhodovací hranici lineárního SVM. (3) Jak ovlivní hranici přidání bodu (3, 3) do třídy A? (4) Pomohlo by na těchto datech RBF jádro? Nakreslete dataset, kde RBF pomohou.

Řešení.

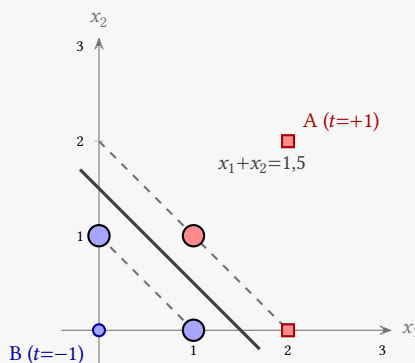
(1) *Princip.* Viz výklad výše: SVM hledá oddělovací přímku s *maximálním marginem*, tj. řeší QP úlohu $\min \frac{1}{2} \|\mathbf{w}\|^2$ za podmínek $t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$. Hranici určují *podpůrné vektory* (body s $a_i > 0$); ty leží přesně na marginu, obráceně to platit nemusí.

(2) *Hranice.* Geometrie dat: třída B je v „levém dolním“ rohu, třída A nad ní. Nejbližší body třídy A leží na přímce $x_1 + x_2 = 2$, nejbližší body třídy B na $x_1 + x_2 = 1$; separátor kolmý na směr (1, 1) uprostřed mezi nimi je zároveň maximálně marginový. Lepší být nemůže: bod (1, 1) třídy A má od úsečky (1, 0)–(0, 1), která leží v konvexním obalu třídy B, vzdálenost $1/\sqrt{2}$, a každá oddělovací přímka

musí ležet mezi nimi, takže žádná nemá margin větší než $1/(2\sqrt{2})$. Tento separátor ho dosahuje (ověřeno i řešičem, viz níže). Je to přímka

$$x_1 + x_2 = 1,5, \quad \text{tj. } \mathbf{w} = (1, 1), \quad b = -1,5$$

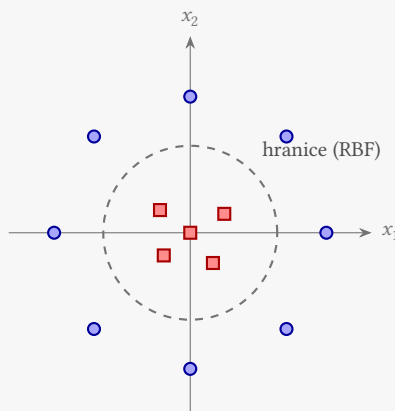
(v kanonické normalizaci $\mathbf{w} = (2, 2)$, $b = -3$, aby nejbližší body měly $|y| = 1$). Na marginu leží $(1, 1)$ a $(2, 0)$ z A a $(1, 0)$, $(0, 1)$ z B. Jako **podpůrné vektory** (s $a_i > 0$) vrátí řešič $(1, 1)$, $(1, 0)$ a $(0, 1)$; bod $(2, 0)$ je na marginu také, ale k určení optima už není potřeba (degenerovaný případ). Při kanonické normalizaci $\mathbf{w} = (2, 2)$ je $\|\mathbf{w}\| = 2\sqrt{2}$, takže margin (vzdálenost separátoru k nejbližšímu bodu) je $\frac{1}{\|\mathbf{w}\|} = \frac{1}{2\sqrt{2}} \approx 0,354$ a celková šíře pásu mezi marginy $\frac{2}{\|\mathbf{w}\|} = \frac{1}{\sqrt{2}} \approx 0,707$.



Skutečný max-margin separátor pro data ze zadání (ověřeno sklearn): plná čára $x_1 + x_2 = 1,5$, čárkované marginy $x_1 + x_2 = 2$ (strana A) a $x_1 + x_2 = 1$ (strana B). Zakroužkované podpůrné vektory: $(1, 1)$ z A a $(1, 0)$, $(0, 1)$ z B. Body $(2, 2)$ a $(0, 0)$ leží daleko a hranici neovlivňují; $(2, 0)$ leží sice na marginu, ale optimum je určeno už třemi vektory.

(3) *Přidání bodu $(3, 3)$ do A.* Bod $(3, 3)$ leží *hluboko* na správné straně (pro něj $x_1 + x_2 = 6 \gg 2$), daleko od marginu. Není to podpůrný vektor, jeho odebrání či přidání hranici nezmění: **separátor i margin zůstanou stejné** ($\mathbf{w} = (1, 1)$, $b = -1,5$). To je klíčová vlastnost SVM – hranici určují jen body na marginu, ne vzdálené body (na rozdíl třeba od lineární regrese, kterou by odlehlý bod „přitáhl“). (Ověřeno numericky: po přidání $(3, 3)$ vrací sklearn též separátor a $(3, 3)$ mezi podpůrnými vektory není.)

(4) *RBF na těchto datech.* Data jsou lineárně separabilní (právě jsme našli přímku), takže RBF jádro by zde **nepomohlo** – spíš by hrozil *overfitting* (přeučení). RBF se vyplatí, až když lineární hranice nestačí. Příklad takových dat – *soustředné kruhy*:



Dataset, kde RBF jádro pomáhá: jedna třída uvnitř (čtverce, malý shluk kolem počátku), druhá v prstenci kolem (kolečka rovnoměrně na kružnici). Žádná přímka je neoddelí, RBF jádro ano, kruhovou hranicí. (Na vygenerovaných datech tohoto typu dosáhne lineární SVM jen $\approx 63\%$ správnosti, RBF 100% ; ověřeno sklearn.)

i Poznámka. [Numerické ověření] Skutečný separátor, podpůrné vektory, neměnnost při přidání (3, 3) i rozdíl lineární vs. RBF na kruzích jsou ověřeny skriptem `src/fig/gen_s7_svm_verify.py` (`sklearn.svm.SVC`). Výsledek: $\mathbf{w} = (2, 2)$, $b = -3$ (kanonicky), margin $1/(2\sqrt{2}) \approx 0,354$ (šíře pásu $1/\sqrt{2}$), podpůrné vektory (1, 1), (1, 0), (0, 1); lineární SVM na soustředných kruzích 63 %, RBF 100 %.

8 Kombinace více modelů

Jeden model má svá omezení: lineární modely nezachytí nelinearity, jeden hluboký rozhodovací strom má vysoký *rozptyl* (drobná změna dat dá úplně jiný strom; viz sekce 6). **Kombinace více modelů** (ensembling) tyto slabiny obchází tím, že více modelů necháme rozhodovat společně. Tato sekce pokrývá tři klasické přístupy: **bagging** (paralelní modely na bootstrapových vzorcích, snižuje rozptyl), **boosting** (sekvenční modely opravující chyby předchůdců) a jejich nejznámější zástupce – **random forest** (náhodný les), tj. bagging rozhodovacích stromů s náhodným výběrem příznaků.

💡 Intuice. Proč vůbec kombinovat? Představme si výbor expertů: pokud každý dělá chyby *jinde* (jejich chyby nejsou korelované), pak hlasováním nebo průměrem se chyby navzájem vyruší a výbor je lepší než kterýkoli jednotlivý expert. Naopak když všichni dělají *tytéž* chyby, kombinace nepomůže. Celé umění ensemblingu je vyrobit modely, které jsou jednotlivě slušné, ale jejich chyby spolu *nesouvisí*.

Definice – Kombinace modelů: hlasování a průměrování

Mějme M natrénovaných modelů $y_1, \dots, y_M$. Jejich predikce zkombinujeme do jediné:

- **hard voting** (tvrdé hlasování) – u klasifikace vrátíme třídu, kterou predikuje *nejvíce* modelů (většina);
- **soft voting** (měkké hlasování / averaging) – zprůměrujeme *rozdělení pravděpodobností* tříd vracená modely a vybereme třídu s nejvyšší průměrnou pravděpodobností; u regrese prostě zprůměrujeme číselné predikce.

8.1 Bagging (bootstrap agregace)

Definice – Bootstrap, bagging

Bootstrapový vzorek trénovací množiny o N příkladech vznikne tak, že z ní vylosujeme N příkladů s *opakováním* (s vracením). Vzniklá množina má stejnou velikost N , ale některé příklady obsahuje vícekrát a jiné vůbec. **Bagging** (*bootstrap aggregating*) natrénuje M kopií téhož modelu, každou na vlastním nezávislém bootstrapovém vzorku, a jejich predikce *agreguje* – u regrese průměrem, u klasifikace hlasováním.

💡 Intuice. Losování s opakováním vyrobí M *podobných, ale ne identických* trénovacích množin, takže vzniklé modely jsou různé. Bagging hlavně **snižuje rozptyl** (variance) odhadu, aniž zvýší vychýlení (bias): je proto ideální pro modely s *nízkým vychýlením a vysokým rozptylem* – typicky hluboké rozhodovací stromy. Zhruba $1/e \approx 37\%$ příkladů se do daného bootstrapového vzorku nedostane (tzv. *out-of-bag* příklady); na nich lze model „zdarma“ validovat.

i Poznámka. [Proč právě bootstrap] Pro neuronové sítě často stačí trénovat modely s různou náhodnou inicializací: ztráta má mnoho lokálních minim, takže modely vyjdou dost odlišné i na stejných datech. U modelů s *konvexní* ztrátovou funkcí (lineární či logistická regrese) ale učení na týchž datech konverguje k témuž optimu bez ohledu na náhodnost (inicializaci, pořadí příkladů). Různé modely tu dostaneme jen změnou samotných trénovacích dat – a to dělá bagging.

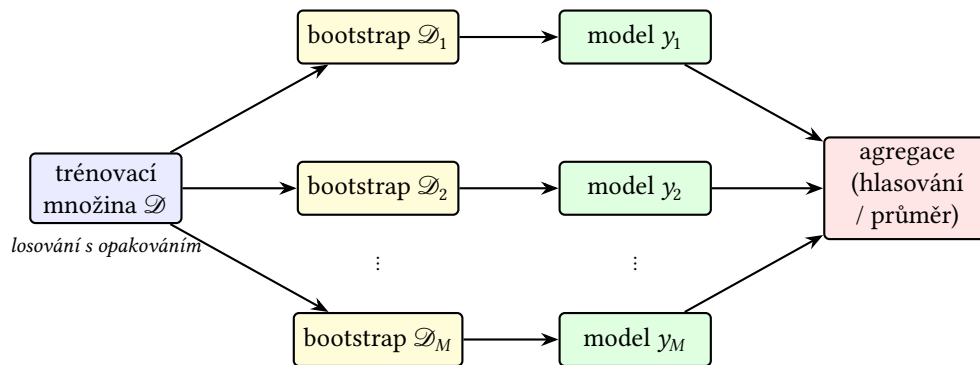


Schéma baggingu. Z jediné trénovací množiny $\mathcal{D}$ vzniká M bootstrapových vzorků (losování N příkladů s *opakováním*). Na každém se natrénuje samostatný model; jejich predikce se na konci agregují (hlasováním u klasifikace, průměrem u regrese). Modely jsou trénované *paralelně* a *nezávisle*.

Proč průměrování snižuje chybu

★ Věta – Redukce chyby průměrováním (nekorelované chyby)

Označme chybu i -tého modelu na příkladu $\mathbf{x}$ jako $\varepsilon_i(\mathbf{x}) = y_i(\mathbf{x}) - t$. Mají-li chyby *nulovou střední hodnotu* a jsou *nekorelované* ($\mathbb{E}[\varepsilon_i \varepsilon_j] = 0$ pro $i \neq j$), pak střední kvadratická chyba průměru M modelů je

$$\mathbb{E}\left[\left(\frac{1}{M} \sum_i \varepsilon_i(\mathbf{x})\right)^2\right] = \frac{1}{M} \mathbb{E}\left[\frac{1}{M} \sum_i \varepsilon_i^2(\mathbf{x})\right],$$

tj. $\frac{1}{M}$ -násobek průměrné chyby jednotlivých modelů.

💡 **Intuice.** Roznásobíme-li $(\sum_i \varepsilon_i)^2 = \sum_{i,j} \varepsilon_i \varepsilon_j$ a vezmeme střední hodnotu, smíšené členy $\mathbb{E}[\varepsilon_i \varepsilon_j]$ ($i \neq j$) vypadnou (nekorelovanost) a zůstane jen M diagonálních členů $\mathbb{E}[\varepsilon_i^2]$; faktor $\frac{1}{M^2}$ před sumou pak dá výsledný $\frac{1}{M}$. Toto je *ideální* mez: chyba klesá až M -krát. V praxi jsou chyby *korelované* (modely se učí z překrývajících se dat), takže zlepšení je menší. Předpoklad nekorelovanosti je tedy klíčový – proto se bagging snaží modely co nejvíc „rozhodit“.

8.2 Boosting a algoritmus AdaBoost

📖 Definice – Boosting

Boosting buduje silný klasifikátor jako *vážený součet* mnoha *slabých* klasifikátorů (např. mělkých stromů), trénovaných **sekvenčně**. Po každém kole se zvýší *váhy* těch trénovacích příkladů, které dosavadní kombinace klasifikuje *špatně*, takže další slabý klasifikátor se zaměří právě na ně.

💡 **Intuice.** Zatímco bagging trénuje modely *paralelně a nezávisle* (cíl: snížit rozptyl), boosting je trénuje *postupně a závisle* – každý nový model „dohání“ chyby předchozích (cíl: snížit vychýlení). Slabý klasifikátor stačí lepší než náhoda (chybovost $< \frac{1}{2}$); jejich vážená kombinace může být libovolně přesná. Metafora: tým, kde se každý nováček soustředí na úlohy, které tým dosud nezvládal.

⚙️ Algoritmus – AdaBoost (binární klasifikace, třídy $t_i \in \{-1, +1\}$)

Vstup: trénovací data $(\mathbf{x}_i, t_i)$, $i = 1, \dots, N$; počet kol T .

Inicializuj váhy příkladů rovnoměrně: $w_i^{(1)} = \frac{1}{N}$.

▷ na začátku stejně důležité

pro $m = 1, \dots, T$:

1. Natrénuj slabý klasifikátor $h_m(\mathbf{x}) \in \{-1, +1\}$ na datech s váhami $w^{(m)}$.

2. Spočti váženou chybu $\varepsilon_m = \sum_i: h_m(\mathbf{x}_i) \neq t_i w_i^{(m)}$. ▷ podíl chybně klasifikované váhy
 3. Spočti váhu klasifikátoru $\alpha_m = \frac{1}{2} \ln \frac{1 - \varepsilon_m}{\varepsilon_m}$. ▷ čím menší chyba, tím větší vliv
 4. Převáž příklady: $w_i^{(m+1)} \propto w_i^{(m)} e^{-\alpha_m t_i h_m(\mathbf{x}_i)}$ a normalizuj na součet 1. ▷ špatně klasifikované získají větší váhu
- Výstup: $H(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^T \alpha_m h_m(\mathbf{x})\right)$. ▷ vážené hlasování slabých klasifikátorů

Intuice. V kroku 4 je $t_i h_m(\mathbf{x}_i) = +1$ pro správně a -1 pro špatně klasifikovaný příklad, takže $e^{-\alpha_m t_i h_m}$ je < 1 pro správně (váha klesne) a > 1 pro chybně (váha vzroste). Příklady, na kterých model selhává, tak v dalším kole „křičí“ hlasitěji. Klasifikátor s malou chybou ε_m dostane velké α_m (velký hlas); klasifikátor s $\varepsilon_m = \frac{1}{2}$ (náhoda) dostane $\alpha_m = 0$ (žádný hlas).

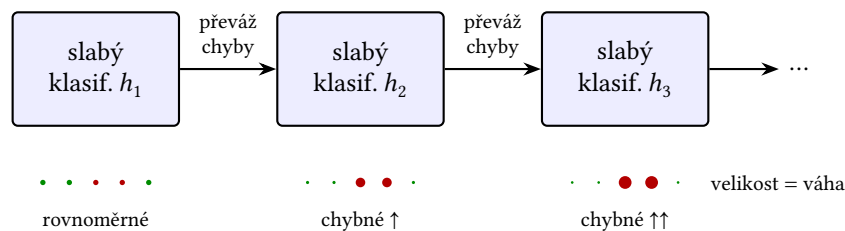


Schéma boostingu. Slabé klasifikátory se trénují *sekvenčně*; po každém kole vzroste váha (velikost bodu) příkladů, které dosud klasifikujeme špatně (červeně), takže další klasifikátor se zaměří na ně. Finální predikce je *vážené hlasování* všech slabých klasifikátorů.

Poznámka. [Gradientní boosting a XGBoost (jen koncept)] Modernější varianta, **gradientní boosting** (GBDT), místo převažování příkladů trénuje každý nový strom tak, aby predikoval *záporný gradient ztráty* dosavadní kombinace (tj. „opravoval rezidua“ předchozích stromů); přidává se s malým koeficientem (learning rate). Efektivní implementace **XGBoost** a **LightGBM** (využívají i druhý řád ztráty a chytré dělení uzlů) patří k nejsilnějším modelům pro *tabulární data*. Detailní matematiku (Newtonův krok, Hessián) zde nerozebíráme; podstatná je myšlenka „sekvenční stromy opravující chybu předchůdců“.

8.3 Random forest (náhodný les)

Definice – Random forest

Random forest (náhodný les) je *bagging rozhodovacích stromů* s jednou přidanou náhodností: při hledání nejlepšího řezu v *každém uzlu* se uvažuje jen **náhodná podmnožina příznaků** (ne všechny). Predikce random forest vznikne *hlasováním* stromů (klasifikace), resp. *průměrem* (regrese).

Algoritmus – Trénink random forest

Vstup: data $\mathcal{D}$ o N příkladech s D příznaky; počet stromů M ; velikost podmnožiny příznaků k (typicky $k \approx \sqrt{D}$ u klasifikace).

pro $m = 1, \dots, M$:

1. Vytvoř bootstrapový vzorek $\mathcal{D}_m$ (losuj N příkladů s opakováním). ▷ bagging
2. Natrénuj rozhodovací strom na $\mathcal{D}_m$, ale v *každém uzlu* vyber nejlepší řez jen z *náhodných k příznaků* (čerstvá podmnožina pro každý uzel). ▷ náhodné příznaky
3. Strom *neprořezávej* (necháváme přerůst – rozptyl srovná až agregace).

Predikce: hlasování (klasifikace) / průměr (regrese) přes všech M stromů.

 **Intuice.** Proč navíc *náhodné příznaky*? Samotný bagging stromů by dal stromy stále dost podobné – pokud existuje jeden velmi silný příznak, vybraly by ho téměř všechny stromy v kořeni a jejich chyby by zůstaly korelované. Omezení na náhodnou podmnožinu příznaků nutí různé stromy „dívat se jinak“, čímž *dekoreluje* jejich chyby; podle věty výše to je přesně to, co dělá agregaci účinnou. (Ještě radikálnější jsou *extra trees*: nehledají ani nejlepší hodnotu řezu, volí ji náhodně z rozsahu příznaku.)

 **Poznámka.** [Random forest vs. jeden strom – výhody a nevýhody] **Jeden strom:** rychlý trénink, snadná *interpretace* (cestu k rozhodnutí lze přechít), ale vysoký *rozptyl* a sklon k overfittingu. **Random forest:** výrazně *přesnější a stabilnější* (lépe generalizuje, odolný vůči overfittingu), zvládá vysoký počet příznaků, dává „zdarma“ odhad chyby (out-of-bag) i důležitost příznaků. Cena: *ztráta interpretovatelnosti* (stovky stromů už člověk nepřečte), pomalejší a paměťově náročnější trénink i inference. Random forest a gradientní boosting stromů patří k nejlepším modelům pro tabulární data (viz příklad v sekci 1).

8.4 Řešené příklady

Příklad – SZZ jaro 2023 č. 6: Učení pomocí kolektivních prediktorů

(Reálná otázka SZZ: jaro 2023, č. 6 – Učení pomocí kolektivních prediktorů)

Zadání. (1) Vysvětlíte princip trénování metodami *bagging* a *boosting*. (2) Popište trénovací algoritmus *random forest* (náhodných lesů). (3) Uveďte konkrétní příklad algoritmu typu *boosting* a stručně vysvětlíte jeho hlavní myšlenku.

Řešení. (1) **Bagging** trénuje M kopií modelu *paralelně*, každou na vlastním *bootstrapovém* vzorku (losování N příkladů s opakováním); predikce agreguje hlasováním/průměrem. Snižuje *rozptyl* (vhodný pro nestabilní modely, např. hluboké stromy). **Boosting** trénuje slabé klasifikátory *sekvencně*; po každém kole zvýší *váhu* chybně klasifikovaných příkladů, takže další model se zaměří na ně. Finální model je vážené hlasování. Snižuje hlavně *vychýlení*.

(2) **Random forest** = bagging rozhodovacích stromů + náhodná podmnožina příznaků v každém uzlu. Pro $m = 1, \dots, M$: vytvoř bootstrapový vzorek, natrénuj na něm (nezprořezaný) strom, ale v každém uzlu hledej nejlepší řez jen z náhodně zvolené podmnožiny příznaků (typicky $\sqrt{D}$). Predikce: hlasování stromů. Dvojí náhodnost (data + příznaky) *dekoreluje* stromy.

(3) **AdaBoost**. Slabé klasifikátory $h_m \in \{-1, +1\}$ se trénují postupně na vážených datech; klasifikátor s váženou chybou ϵ_m dostane hlas $\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$ a váhy chybně klasifikovaných příkladů se vynásobí e^{α_m} (vzrostou). Výsledek $H(\mathbf{x}) = \text{sign} \left(\sum_m \alpha_m h_m(\mathbf{x}) \right)$. Hlavní myšlenka: *každé další kolo se soustředí na to, co dosavadní kombinace ještě nezvládá*.

Příklad – SZZ podzim 2023 č. 30: Kombinace více modelů

(Reálná otázka SZZ: podzim 2023, č. 30 – Kombinace více modelů)

Zadání. (1) Popište myšlenky baggingu a boostingu; jeden boosting algoritmus podrobněji. (2) Jak funguje a jak se trénuje random forest? (3) Porovnejte rozhodovací stromy a random forest (výhody/nevýhody).

Řešení. (1) Viz definice baggingu a boostingu výše a algoritmus **AdaBoost** v předchozím příkladu (slabé klasifikátory, převažování chyb, vážené hlasování α_m).

(2) Viz algoritmus tréninku random forest výše: každý strom na bootstrapovém vzorku, v každém uzlu řez jen z náhodné podmnožiny příznaků; výsledek hlasováním stromů.

(3) **Stromy:** interpretovatelné, rychlé, ale vysoký rozptyl a overfitting. **Random forest:** přesnější, stabilnější, lépe generalizuje, OOB odhad chyby a důležitost příznaků zdarma; cena je ztráta interpretovatelnosti a vyšší výpočetní nároky. (Viz poznámka „random forest vs. strom“ výše.)

(Reálná otázka SZZ: jaro 2026, č. 29 – Kombinace modelů)

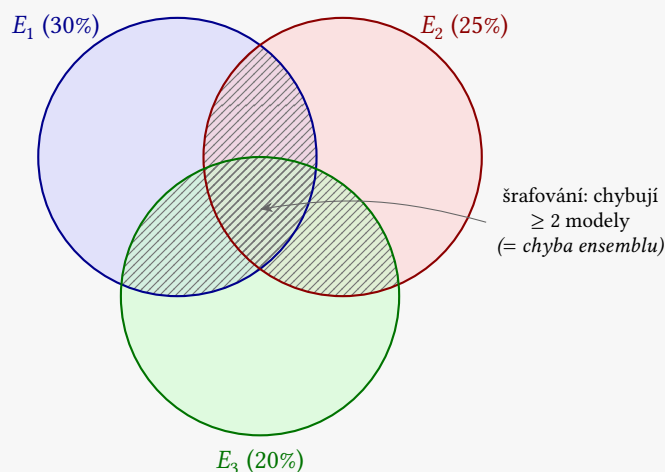
Zadáni. (1) Stručně popište myšlenky baggingu a boostingu a uveďte modely, které je používají. (2) Tři modely M_1, M_2, M_3 pro binární klasifikaci mají accuracy 0,70, 0,75, 0,80. Vytvoříme ensemble většinovým hlasováním (finální odpověď je ta častější ze tří). Za jakých podmínek bude mít ensemble vyšší accuracy než nejlepší model? Jaká je nejlepší dosažitelná accuracy ensemble? Kdy bude naopak horší?

Řešení. (1) **Bagging** – paralelní modely na bootstrapových vzorcích, agregace hlasováním/průměrem; příklad: *random forest*. **Boosting** – sekvenční modely, převažování chyb; příklad: *AdaBoost*, *XGBoost* (gradientní boosting).

(2) Klíč je uvažovat **chybové množiny** $E_i = \{\text{vzorky, které } M_i \text{ klasifikuje špatně}\}$. Z přesností: $|E_1| = 30\%$, $|E_2| = 25\%$, $|E_3| = 20\%$ z testovací množiny. Při většinovém hlasování tří modelů je vzorek klasifikován **špatně**, právě když chybují alespoň dva modely, tj. vzorek leží alespoň ve dvou z množin E_1, E_2, E_3 . Chyba ensemble je tedy

$$\text{err}_{\text{ens}} = |\text{vzorky pokryté alespoň dvěma z } E_1, E_2, E_3|.$$

Vše dál plyne z toho, jak se chybové množiny překrývají.



Chybové množiny tří modelů. Vzorek je ensemblem klasifikován špatně, právě když leží alespoň ve dvou množinách (vyšrafovaná oblast tvořená třemi párovými průniky). Disjunktní množiny (žádný překryv) $\Rightarrow$ ensemble se nikdy nemýlí; velké překryvy $\Rightarrow$ ensemble se mýlí často.

Nejlepší dosažitelná accuracy. Mají-li modely *disjunktní* chybové množiny (každý chybuje jinde), žádný vzorek neleží ve dvou množinách, takže $\text{err}_{\text{ens}} = 0$, tj. *accuracy* 100 %. To je možné, protože $30\% + 25\% + 20\% = 75\% \leq 100\%$ se do testovací množiny vejde disjunktně.

nejlepší dosažitelná accuracy ensemble = 100 % (disjunktní chyby).

Kdy je ensemble lepší než nejlepší model. Nejlepší model M_3 má chybu 20 %. Ensemble je lepší, právě když $\text{err}_{\text{ens}} < 20\%$, tj. když se chybové množiny překrývají jen málo (chyby modelů jsou málo korelované). Stačí tedy, aby modely chybovaly převážně na *různých* vzorcích.

Kdy je ensemble horší. Při *silně korelovaných* (překrývajících se) chybách:

- je-li $E_2 \subseteq E_1$ (kdykoli chybuje M_2 , chybuje i M_1), je na celém E_2 chyba ≥ 2 , takže $\text{err}_{\text{ens}} \geq 25\%$, tj. *accuracy nejvýše* 0,75;
- je-li navíc i zbytek $E_1 \setminus E_2$ pokryt E_3 , je celé E_1 pokryté dvěma množinami, $\text{err}_{\text{ens}} \geq 30\%$, tj. *accuracy nejvýše* 0,70 (ensemble není lepší ani než nejhorší model).

Vůbec *nejhorší* dosažitelná hodnota (maximální překryv při daných velikostech) je $1 - \frac{0,30+0,25+0,20}{2} = 0,625$: každý vzorek, na kterém ensemble chybí, spotřebuje aspoň dvě chyby jednotlivých modelů, a těch je dohromady 75 %. Mez se dosáhne např. při $|E_1 \cap E_2| = 17,5\%$, $|E_1 \cap E_3| = 12,5\%$, $|E_2 \cap E_3| = 7,5\%$ (trojný průnik prázdný): aspoň dvěma množinami je pokryto 37,5 % vzorků a accuracy klesne na 0,625. (Mají-li chyby být celá procenta, tj. $N = 100$, je nejhorší 0,63, např. při 17/12/8 %.) Ensemble tedy může být horší než i *nejhorší jednotlivý model*, pokud modely dělají tytéž chyby.

Závěr. Většinové hlasování pomáhá jen tehdy, jsou-li chyby modelů málo korelované: při *nezávislých* chybách chybí právě dva modely s pravděpodobností $0,3 \cdot 0,25 \cdot 0,8 + 0,3 \cdot 0,75 \cdot 0,2 + 0,7 \cdot 0,25 \cdot 0,2 = 0,14$ a všechny tři s 0,015, takže accuracy ensmbly je $0,845 > 0,80$; při *disjunktních* chybových množinách až 100 %. Korelované chyby kombinaci uškodí. (Přesnosti 0,70/0,75/0,80 samy o sobě výsledek neurčují – rozhoduje překryv chybových množin. Ověřeno vyčerpávajícím prohledáním rozložení chyb, src/fig/gen_s8_ensemble.py.)

9 Statistické testy

Statistický test rozhoduje *ano/ne* otázku o datech: liší se průměrná hmotnost sušenek od deklarované? Liší se počty zákazníků mezi dny? Tato látka patří do **pravděpodobnosti a statistiky** (okruh M5, předmět NMAI059), ne do strojového učení v užším smyslu; požadavky S3 ji ale jmenují, protože se ve strojovém učení používá k ověření, zda je rozdíl mezi dvěma modely (nebo mezi pozorováním a předpokladem) *statisticky významný*, nebo jen náhodný šum. Zde proto zopakujeme obecný rámec testování hypotéz (podrobně viz M5) a rozebereme tři testy z požadavků: *jednovýběrový* a *dvouvýběrový Studentův t-test* a *chí-kvadrát test dobré shody*.

i Poznámka. [Rozsah] Držíme se přesně požadavků: u chí-kvadrát testu jen *test dobré shody* (ne test nezávislosti v kontingenční tabulce), u t-testu jednovýběrovou a dvouvýběrovou variantu. Resamplingové testy (bootstrap, permutační test) z NPFL129 (přednáška 12) ani ANOVA do požadavků nepatří, neuvádíme je.

9.1 Rámec testování hypotéz

Připomeňme z M5 stavební kameny každého testu. Cíl je jediný: rozhodnout, zda data dostatečně silně *odporují* výchozímu předpokladu.

Definice – Hypotézy, testová statistika, kritický obor

- **Nulová hypotéza** H_0 : výchozí, konzervativní tvrzení („nic zajímavého se neděje“ – sušenky váží přesně tolik, kolik mají; zákazníci přicházejí každý den stejně).
- **Alternativní hypotéza** H_1 : tvrzení, které přijmeme, pokud H_0 zamítneme (hmotnost se liší; rozdělení neodpovídá).
- **Testová statistika** T : číslo spočtené z dat, jehož rozdělení za *platnosti* H_0 známe (zde t -rozdělení nebo χ^2 -rozdělení).
- **Hladina významnosti** α (typicky 0,05): maximální přípustná pravděpodobnost *chyby 1. druhu*, tj. zamítnutí H_0 , ačkoli platí.
- **Kritická hodnota** c a **kritický obor** W : hranice a množina hodnot T , pro které H_0 zamítáme. W se volí tak, aby $P(T \in W \mid H_0) = \alpha$.

💡 Intuice. Postup je vždy stejný: (1) formuluj H_0 a H_1 ; (2) zvol α ; (3) spočti testovou statistiku T z dat; (4) urči kritickou hodnotu z příslušného rozdělení (z tabulek nebo softwaru); (5) je-li T v kritickém oboru (zde T „příliš velké“), **zamítni** H_0 , jinak ji *nezamítej*. Logika je důkazem sporem: předpokládáme H_0 a ptáme se, jak nepravděpodobná jsou pak pozorovaná data. Pokud velmi (statistika padne do ocasu), H_0 zavrhneme.

i Poznámka. [p-hodnota] Ekvivalentní pohled: **p-hodnota** je pravděpodobnost (za H_0), že bychom dostali výsledek *aspoň tak extrémní* jako pozorovaný. Rozhodovací pravidlo $T \in W$ je totéž co $p \leq \alpha$. P-hodnota navíc říká, *jak silně* data H_0 porušují. Pozor: p-hodnota *není* pravděpodobnost, že H_0 platí. Rámec p-hodnoty viz M5.

9.2 Studentův t-test

T-test porovnává *střední hodnoty* u normálně rozdělených dat, když **neznáme rozptyl** a musíme ho odhadnout z výběru. Právě nahrazení neznámého σ výběrovou směrodatnou odchylkou s vede místo normálního rozdělení na *t-rozdělení* (Studentovo) s ν *stupni volnosti*: je symetrické kolem nuly, ale má těžší ocasy než $N(0, 1)$; s rostoucím ν se k $N(0, 1)$ blíží (pro $\nu \geq 30$ jsou prakticky k nerozeznání).

Jednovýběrový t-test

Definice — Jednovýběrový t-test

Máme výběr $X_1, \dots, X_n$ z normálního rozdělení s neznámou střední hodnotou μ a neznámým rozptylem. Testujeme, zda se μ rovná dané konstantě μ_0 :

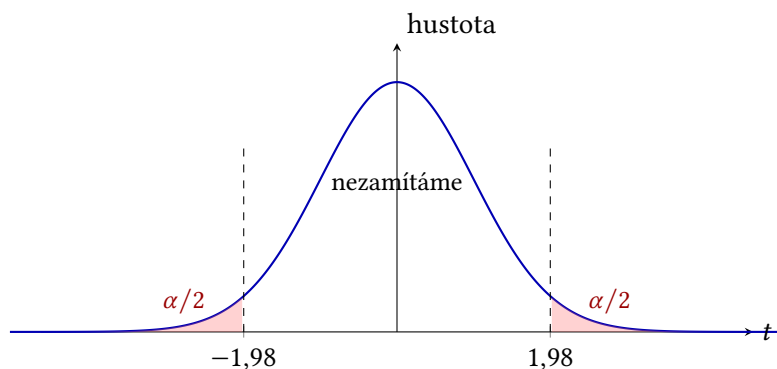
$$H_0 : \mu = \mu_0, \quad H_1 : \mu \neq \mu_0 \quad (\text{oboustranně}).$$

Testová statistika je

$$t = \frac{\bar{X} - \mu_0}{s/\sqrt{n}}, \quad \bar{X} = \frac{1}{n} \sum_{i=1}^n X_i, \quad s^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2,$$

kde $\bar{X}$ je výběrový průměr a s výběrová směrodatná odchylka. Za H_0 má t **Studentovo t-rozdělení s $n - 1$ stupni volnosti**. H_0 na hladině α zamítáme, pokud $|t| \geq t_{n-1, 1-\alpha/2}$ (kritická hodnota oboustranného testu).

💡 Intuice. Čitatel $\bar{X} - \mu_0$ měří, jak daleko je pozorovaný průměr od testované hodnoty; jmenovatel $s/\sqrt{n}$ je *směrodatná chyba průměru*, tedy typický rozptyl $\bar{X}$. Statistika t tak vyjadřuje rozdíl *v jednotkách jeho vlastní nejistoty*: $t = 10$ znamená, že pozorovaný průměr je deset směrodatných chyb od μ_0 – to je za H_0 extrémně nepravděpodobné. Jednostrannou variantu ($H_1 : \mu > \mu_0$, resp. $<$) použijeme, je-li předem jasný směr odchylky; pak je kritická hodnota $t_{n-1, 1-\alpha}$ a kritický obor jen jeden ocas.



Hustota t -rozdělení s 99 stupni volnosti (oboustranný t -test, $n = 100$). Při $\alpha = 0,05$ leží kritický obor v obou ocasech za $\pm t_{99, 0,975} \approx \pm 1,98$ (**červeně**, dohromady míra 0,05). Pro 99 stupňů volnosti je rozdělení prakticky nerozeznatelné od $N(0, 1)$ (kritická hodnota 1,98 vs. $z_{0,975} = 1,96$). Hustota generována scipy (src/fig/gen_s9_dist.py).

Dvouvýběrový t -test

Definice – Dvouvýběrový t -test

Máme *dva nezávislé* výběry $X_1, \dots, X_{n_1}$ a $Y_1, \dots, Y_{n_2}$ z normálních rozdělení se středními hodnotami μ_1, μ_2 . Testujeme rovnost středních hodnot:

$$H_0 : \mu_1 = \mu_2, \quad H_1 : \mu_1 \neq \mu_2.$$

Při *shodných rozptylech* obou skupin se testová statistika spočte ze *sdrúženého* odhadu rozptylu s_p^2 :

$$t = \frac{\bar{X} - \bar{Y}}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}, \quad s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2},$$

kde $\bar{X}, \bar{Y}$ jsou výběrové průměry a s_1^2, s_2^2 výběrové rozptyly. Za H_0 má t Studentovo rozdělení s $n_1 + n_2 - 2$ stupni volnosti; H_0 zamítáme při $|t| \geq t_{n_1+n_2-2, 1-\alpha/2}$.

Intuice. Myšlenka je stejná jako u jednovýběrového testu, jen místo „průměr vs. konstanta“ porovnáváme „průměr vs. průměr“: čitatel je rozdíl průměrů $\bar{X} - \bar{Y}$, jmenovatel jeho směrodatná chyba. Sdrúžený rozptyl s_p^2 je vážený průměr rozptylů obou skupin (váhami jsou stupně volnosti) – nejlepší společný odhad společného σ^2 . Nelze-li předpokládat shodné rozptyly, používá se *Welchova* varianta (jiný jmenovatel a stupně volnosti); princip rozhodování zůstává.

Poznámka. [Párový vs. nepárový] Jsou-li data *párová* (např. stejní pacienti před a po léčbě), počítáme rozdíly $D_i = X_i - Y_i$ a aplikujeme *jednovýběrový* t -test na $H_0 : \mu_D = 0$. Výše uvedený dvouvýběrový test předpokládá *nezávislé* skupiny.

Příklad – Dvouvýběrový t -test – ilustrace

Zadání. Dvě skupiny studentů (každá $n = 8$) psaly stejný test. Skupina A měla průměr 80 a výběrovou směrodatnou odchylku 4, skupina B průměr 74 a odchylku 4. Liší se průměry statisticky významně na hladině $\alpha = 0,05$ (předpokládáme normalitu a shodné rozptyly)?

Řešení. Sdrúžený rozptyl $s_p^2 = \frac{7 \cdot 16 + 7 \cdot 16}{14} = 16$, tedy $s_p = 4$. Testová statistika

$$t = \frac{80 - 74}{4 \sqrt{\frac{1}{8} + \frac{1}{8}}} = \frac{6}{4 \cdot 0,5} = \boxed{3,0}, \quad \text{df} = n_1 + n_2 - 2 = 14.$$

Kritická hodnota $t_{14, 0,975} \approx 2,14$. Protože $|3,0| \geq 2,14$, H_0 **zamítáme** – skupiny se liší.

Příklad – SZZ jaro 2024 č. 26: Jednovýběrový t-test (sušenky)

(Reálná otázka SZZ: jaro 2024, č. 26 – Statistické testy – t-test)

Zadání. Výrobce deklaruje hmotnost sušenek 100 g. Z kontroly: $n = 100$ sušenek, průměrná hmotnost $\bar{X} = 102$ g, výběrová směrodatná odchylka $s = 2$ g. T-testem rozhodněte, zda se průměrná hmotnost liší od deklarovaných 100 g. (1) Hypotézy. (2) Testová statistika a její rozdělení. (3) Rozhodnutí, je-li kritická hodnota na $\alpha = 0,05$ přibližně 2.

Řešení.

(1) *Hypotézy.* Jednovýběrový t-test proti deklarované hodnotě $\mu_0 = 100$:

$$H_0 : \mu = 100, \quad H_1 : \mu \neq 100 \quad (\text{oboustranný}).$$

(2) *Testová statistika.* Dosadíme do vzorce $t = (\bar{X} - \mu_0)/(s/\sqrt{n})$:

$$t = \frac{102 - 100}{2/\sqrt{100}} = \frac{2}{2/10} = \frac{2}{0,2} = \boxed{10}.$$

Za H_0 má t -rozdělení s $n - 1 = \boxed{99}$ stupni volnosti.

(3) *Rozhodnutí.* Kritická hodnota je $t_{99, 0,975} \approx 2$ (přesně 1,984). Protože $|t| = 10 \gg 2$, statistika leží hluboko v kritickém oboru, H_0 **zamítáme**. Průměrná hmotnost sušenek se na hladině $\alpha = 0,05$ statisticky významně liší od deklarovaných 100 g (p-hodnota je prakticky nulová).

9.3 Chí-kvadrát test dobré shody

Zatímco t-test pracuje se *spojitými* daty a jejich průměrem, **chí-kvadrát test dobré shody** (goodness-of-fit) pracuje s *kategoriálními* daty: máme pozorované počty (četnosti) v několika kategoriích a ptáme se, zda odpovídají nějakému teoretickému rozdělení.

Definice – Chí-kvadrát test dobré shody

Pozorování je rozděleno do k kategorií s *pozorovanými četnostmi* $O_1, \dots, O_k$ (celkem $n = \sum_i O_i$). Za H_0 jsou teoretické pravděpodobnosti kategorií $p_1, \dots, p_k$ ($\sum p_i = 1$), což dává *očekávané četnosti* $E_i = n p_i$:

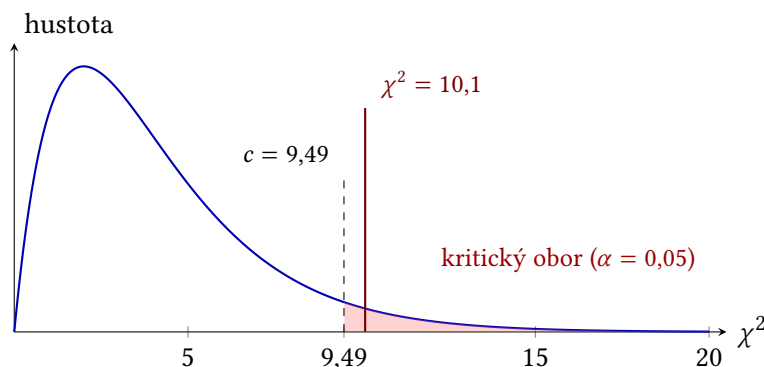
$$H_0 : \text{pozorované četnosti odpovídají očekávaným } (E_i = n p_i), \quad H_1 : \text{neodpovídají.}$$

Testová statistika je

$$\chi^2 = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i}.$$

Za H_0 má pro dostatečně velkou n přibližně χ^2 -rozdělení s $k - 1$ **stupni volnosti**. H_0 zamítáme, pokud $\chi^2 \geq \chi_{k-1, 1-\alpha}^2$ (test je vždy *jednostranný* – velká hodnota svědčí proti shodě).

 **Intuice.** Každý sčítanec $(O_i - E_i)^2/E_i$ měří, jak moc se pozorovaná četnost v kategorii odchýlila od očekávané, *relativně* k očekávané (dělení E_i normuje měřítko). Když data sedí, jsou rozdíly malé a χ^2 je malé; velký nesoulad v kterékoli kategorii statistiku nafoukne. Stupňů volnosti je $k - 1$, ne k , protože jedna vazba $\sum_i O_i = n$ ubírá jeden stupeň volnosti (poslední četnost je dopočitatelná z ostatních). Pravidlo: aproximace χ^2 -rozdělením je spolehlivá, jen když jsou očekávané četnosti dost velké (obvykle $E_i \geq 5$).



Hustota χ^2 -rozdělení se 4 stupni volnosti (pět kategorií, $k - 1 = 4$). Kritický obor na hladině $\alpha = 0,05$ je pravý ocas za kritickou hodnotou $c = \chi_{4,0,95}^2 \approx 9,49$ (červeně, míra 0,05). Pozorovaná statistika $\chi^2 = 10,1$ leží v kritickém oboru, takže H_0 zamítáme. Rozdělení je nesymetrické a kladné (statistika je součet čtverců). Hustota generována scipy (src/fig/gen_s9_dist.py).

Příklad – SZZ podzim 2023 č. 29: Chí-kvadrát test dobré shody (zákazníci)

(Reálná otázka SZZ: podzim 2023, č. 29 – chí-kvadrát test (test dobré shody))

Zadání. (1) Popište χ^2 test dobré shody: hypotézy, testová statistika, její rozdělení. (2) Obchod měl za pět pracovních dní celkem 100 zákazníků; po dnech (po–pá) postupně 25, 30, 15, 14, 16. Rozhodněte χ^2 testem, zda se počty mezi dny statisticky významně liší. *Nápověda:* kritické hodnoty na $\alpha = 0,05$ jsou 7,8 (df = 3), 9,5 (df = 4), 11,1 (df = 5), 12,6 (df = 6).

Řešení.

(1) *Popis testu.* H_0 : pozorované četnosti odpovídají očekávaným; H_1 : neodpovídají. Statistika $\chi^2 = \sum_i (O_i - E_i)^2 / E_i$ má za H_0 rozdělení χ^2 s $k - 1$ stupni volnosti, kde k je počet kategorií. Velká hodnota svědčí proti H_0 .

(2) *Výpočet.* „Zákazníků je každý den stejně“: 5 kategorií, $n = 100$, takže za H_0 je $p_i = \frac{1}{5}$ a očekávaná četnost $E_i = np_i = 100 \cdot \frac{1}{5} = 20$ pro každý den. Dosadíme pozorované četnosti:

$$\chi^2 = \frac{(25 - 20)^2}{20} + \frac{(30 - 20)^2}{20} + \frac{(15 - 20)^2}{20} + \frac{(14 - 20)^2}{20} + \frac{(16 - 20)^2}{20}.$$

Čitatele jsou $5^2, 10^2, (-5)^2, (-6)^2, (-4)^2 = 25, 100, 25, 36, 16$, součet 202, tedy

$$\chi^2 = \frac{25 + 100 + 25 + 36 + 16}{20} = \frac{202}{20} = \boxed{10,1}.$$

Stupně volnosti: $k - 1 = 5 - 1 = 4$, kritická hodnota z nápovědy $c = \chi_{4,0,95}^2 = 9,5$.

Rozhodnutí. $\chi^2 = 10,1 > 9,5 = c$, statistika padla do kritického oboru, H_0 **zamítáme**. Počty zákazníků v jednotlivých dnech se na hladině $\alpha = 0,05$ statisticky významně liší (rozdíl není jen náhodný šum; p-hodnota $\approx 0,039 < 0,05$). Největší příspěvek ke statistice dává úterý (30 zákazníků, příspěvek $100/20 = 5$ z celkových 10,1), ostatní dny přispívají nejvýše 1,8 (čtvrtek).

i Poznámka. [Když test nezamítá] Pozor na interpretaci: *nezamítnutí* H_0 neznamená, že H_0 „platí“ – jen že data nemají dost síly ji vyvrátit. U obou příkladů by stačil výrazně menší rozdíl (sušenky 100,3 g, vyrovnanější počty zákazníků) k tomu, aby test nezamítl, ač H_0 doslova neplatí. Statistická významnost také není totéž co *praktická* významnost: u obrovského vzorku zamítneme H_0 i při zanedbatelném rozdílu.

10 Učení bez učitele: shlukování a redukce dimenze

Dosud měla každá data *cíl* t (label) – učili jsme se s učitelem. Při **učení bez učitele** máme jen vstupy $\mathbf{x}_1, \dots, \mathbf{x}_N$ bez cílů a hledáme v nich *strukturu*: přirozené skupiny (*shlukování*) nebo nízkodimenzionální popis (*redukce dimenze*). Není „správná odpověď“ k porovnání, takže evaluace je vnitřní (jak kompaktní jsou shluky, kolik rozptylu zachováme).

i Poznámka. [Vztah k S1 a M2] S1 (sekce *Strojové učení*) zmiňuje EM jako Bayesovské učení se skrytými proměnnými a poznamenává, že k -means je jeho speciální případ s „tvrdým“ přiřazením. Zde shlukování rozvádíme do hloubky (algoritmus, kritérium, konvergence) a EM dořešíme na konkrétní SZZ úloze. *Redukce dimenze* přes PCA stojí na vlastních číslech a vlastních vektorech symetrické matice (okruh **M2**); odtud bereme spektrální rozklad jako hotový nástroj.

10.1 Shlukování algoritmem k -means

Definice – Úloha shlukování a kritérium k -means

Mějme N bodů $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^D$ a zadaný počet shluků K . Hledáme *centroidy* (středů) $\mu_1, \dots, \mu_K \in \mathbb{R}^D$ a přiřazení každého bodu k jednomu shluku, popsané indikátory $z_{i,k} \in \{0, 1\}$ ($z_{i,k} = 1$, právě když bod $\mathbf{x}_i$ patří do shluku k ; pro každé i je právě jedno $z_{i,k} = 1$). Minimalizujeme **kritérium** (součet čtverců vzdáleností bodů od jejich centroidu)

$$J(\{z_{i,k}\}, \{\mu_k\}) = \sum_{i=1}^N \sum_{k=1}^K z_{i,k} \|\mathbf{x}_i - \mu_k\|^2.$$

💡 Intuice. J je celková „rozptýlenost“ shluků: čím menší, tím kompaktnější jsou shluky kolem svých středů. Najít globální minimum přes *všechna* přiřazení je NP-těžké (exponenciálně mnoho rozdělení), proto k -means minimalizuje J *střídavě*: drží středů a optimalizuje přiřazení, pak drží přiřazení a optimalizuje středů. To je přesně struktura EM (sekce 10.4).

Algoritmus – k -means (Lloydův algoritmus)

Vstup: body $\mathbf{x}_1, \dots, \mathbf{x}_N$, počet shluků K .

1. Inicializace. Zvol K počátečních centroidů $\mu_1, \dots, \mu_K$

▷ náhodně z dat / k -means++

2. opakuj až do konvergence (přiřazení se nemění) nebo do limitu iterací:

(a) **Krok přiřazení.** Každý bod přiřaď k *nejbližšímu* centroidu:

▷ drží středů, mění z

$$z_{i,k} = \begin{cases} 1 & k = \operatorname{argmin}_j \|\mathbf{x}_i - \mu_j\|^2, \\ 0 & \text{jinak.} \end{cases}$$

(b) **Krok přepočtu.** Každý centroid přepočti jako *průměr* bodů svého shluku: ▷ drží z , mění středů

$$\mu_k = \frac{\sum_i z_{i,k} \mathbf{x}_i}{\sum_i z_{i,k}}.$$

Výstup: přiřazení $z_{i,k}$ a centroidy μ_k .

💡 Intuice. Oba kroky jsou *optimální pro svoji polovinu*. (a) Při pevných středech je J minimalizováno tím, že každý bod jde k nejbližšímu středů (jiné přiřazení by jen zvětšilo jeho člen $\|\mathbf{x}_i - \mu_k\|^2$). (b) Při pevném přiřazení je $\sum_i z_{i,k} \|\mathbf{x}_i - \mu\|^2$ jako funkce μ minimalizována *průměrem* bodů shluku – derivace podle μ je $-2 \sum_i z_{i,k} (\mathbf{x}_i - \mu) = 0$, odkud $\mu = \bar{\mathbf{x}}$ shluku. Proto se každý krok počítá tak, jak se počítá.

Inicializace: náhodně vs. k-means++

Náhodná inicializace volí K počátečních středů jako náhodné body z dat. Riziko: dva středy padnou do téhož hustého shluku a jeden přirozený shluk zůstane bez středu. **k-means++** volí středy „rozprostřené“: první náhodně, každý další s pravděpodobností *úměrnou čtverci vzdálenosti* od nejbližšího už zvoleného středu. Vzdálené (dosud nepokryté) oblasti tak mají větší šanci dostat střed. Dokazatelně dává v očekávání $\mathcal{O}(\log K)$ -aproximaci optimálního J .

Poznámka. [Praxe] Protože k -means konverguje jen k *lokálnímu* minimu (viz níže), spouští se obvykle *vícekrát* s různými inicializacemi a vybere se běh s *nejmenším* J (scikit-learn má ve výchozím stavu `n_init=10`).

Konvergence k lokálnímu optimu

★ Věta — k -means konverguje (k lokálnímu minimu)

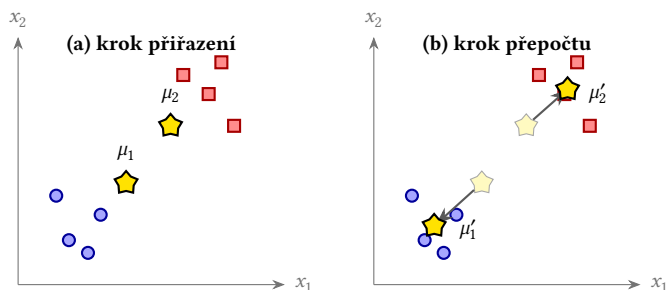
Algoritmus k -means se po konečně mnoha iteracích zastaví; kritérium J při tom neroste a výsledek je *lokální* (ne nutně globální) minimum.

Intuice. [Argument konvergence – dotažený] Tvrzení stojí na dvou pozorováních a jedné konečnosti:

1. **Žádný krok J nezvýší.** Krok přiřazení (a) přiřadí každý bod k nejbližšímu středu, takže jeho příspěvek $\|\mathbf{x}_i - \mu_k\|^2$ klesne nebo zůstane – J neroste. Krok přepočtu (b) nahradí každý střed průměrem svého shluku, který *minimalizuje* $\sum_i z_{i,k} \|\mathbf{x}_i - \mu\|^2$ (viz intuice výše) – J opět neroste. Posloupnost hodnot J je tedy *nerostoucí* a zdola omezená nulou.
2. **Konečně mnoho stavů.** J je plně určeno *přiřazením* $\{z_{i,k}\}$ (středy jsou pak dané průměry). Různých přiřazení N bodů do K shluků je konečně mnoho ($\leq K^N$).
3. **Závěr.** Algoritmus prochází přiřazeními tak, že J neroste; kdyby se přiřazení změnilo, J *ostře* klesne (jinak jsme v minimu a iterace se zastaví). Ostře klesající hodnota na konečné množině stavů nemůže pokračovat donekonečna – žádný stav se nezopakuje. Po konečně mnoha krocích se přiřazení přestane měnit, tehdy se zastavíme.

Že jde jen o *lokální* minimum, plyne z toho, že střídavá minimalizace optimalizuje vždy jen jednu skupinu proměnných; výsledek je „nejlepší v okolí“, ne globálně. *Proto* je výsledek citlivý na inicializaci a spouští se *vícekrát* (viz výše).

Poznámka. [Pozor na prázdnou frázi] Říct „ k -means konverguje, protože J klesá“ je *neúplné*: klesání samo ještě nezaručuje *zastavení* (klesat lze i nekonečně, $1/n \rightarrow 0$). Klíč je *konečnost* počtu přiřazení – proto se posloupnost monotónně klesajících hodnot na konečné množině stavů musí ustálit.



Jedna iterace k -means. (a) Krok přiřazení: každý bod (modrý kruh / červený čtverec) jde k bližšímu z centroidů (žluté hvězdy μ_1, μ_2). (b) Krok přepočtu: každý centroid se posune do těžiště (průměru) bodů svého shluku – šipka vede od staré polohy k nové μ'_k . Kroky se střídají, dokud se přiřazení nemění.

Řešené příklady: *k*-means

Příklad – SZZ léto 2022 č. 12: K-means – první iterace

(Reálná otázka SZZ: léto 2022, č. 12 – Shlukování (*K-means*))

Zadání. Šest bodů s příznaky A_1, A_2 : 1(2, 4), 2(1, 3), 3(0, 4), 4(5, 2), 5(6, 2), 6(4, 0); $K = 2$, počáteční náhodné přiřazení $C_1 = \{2, 3, 4\}$, $C_2 = \{1, 5, 6\}$. (1) Spočítejte centroidy obou shluků. (2) Přiřadte každý bod k nejbližšímu centroidu (euklidovsky) a uveďte nové shluky.

Řešení. (1) Centroid je průměr bodů shluku:

$$c_1 = \left(\frac{1+0+5}{3}, \frac{3+4+2}{3} \right) = (2, 3), \quad c_2 = \left(\frac{2+6+4}{3}, \frac{4+2+0}{3} \right) = (4, 2).$$

(2) Pro každý bod porovnáme čtverce vzdáleností k $c_1 = (2, 3)$ a $c_2 = (4, 2)$:

bod	$\ \cdot - c_1\ ^2$	$\ \cdot - c_2\ ^2$	shluk
1 (2, 4)	1	8	C_1
2 (1, 3)	1	10	C_1
3 (0, 4)	5	20	C_1
4 (5, 2)	10	1	C_2
5 (6, 2)	17	4	C_2
6 (4, 0)	13	4	C_2

Nové shluky: $C_1 = \{1, 2, 3\}$, $C_2 = \{4, 5, 6\}$.

Příklad – SZZ léto 2023 č. 8: *k*-means iterace + hierarchické shlukování

(Reálná otázka SZZ: léto 2023, č. 8 – Shlukování)

Zadání. Data (1, 1), (2, 1), (3, 2), (4, 3), (4, 2), $k = 2$, počáteční středy $\mu_1 = (1, 1)$, $\mu_2 = (3, 2)$. (1) Nové středy po jedné iteraci? (2) Co v dalších iteracích? (3) Popište hierarchické shlukování a jeho výsledek na těchto datech.

Řešení. (1) *Krok přiřazení* – ke každému bodu spočteme vzdálenost k oběma středům (stačí porovnat čtverce):

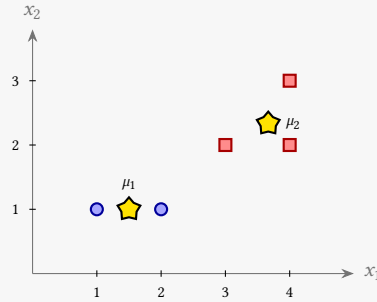
bod	$\ \cdot - \mu_1\ $	$\ \cdot - \mu_2\ $	shluk
(1, 1)	0	$\sqrt{5}$	1
(2, 1)	1	$\sqrt{2}$	1
(3, 2)	$\sqrt{5}$	0	2
(4, 3)	$\sqrt{13}$	$\sqrt{2}$	2
(4, 2)	$\sqrt{10}$	1	2

Shluk 1 = {(1, 1), (2, 1)}, shluk 2 = {(3, 2), (4, 3), (4, 2)}. *Krok přepočtu* (průměry):

$$\mu_1 = \left(\frac{1+2}{2}, \frac{1+1}{2} \right) = \boxed{(1,5, \ 1)}, \quad \mu_2 = \left(\frac{3+4+4}{3}, \frac{2+3+2}{3} \right) = \boxed{\left(\frac{11}{3}, \ \frac{7}{3} \right)}.$$

(2) Opakované přiřazení s novými středy dá *totéž* rozdělení (body (1, 1), (2, 1) zůstanou u μ_1 , zbytek u μ_2), takže se středy už **nemění** – **algoritmus zkonvergoval** (po jedné iteraci).

(3) *Aglomerativní* shlukování (viz 10.2): začni s pěti jednobodovými shluky a opakovaně slučuj dva nejbližší. Při *single linkage* (vzdálenost nejbližších bodů) mají tři dvojice vzdálenost 1: (1, 1)–(2, 1), (3, 2)–(4, 2) a (4, 2)–(4, 3). Ve výšce 1 tak vzniknou skupiny {(1, 1), (2, 1)} a {(3, 2), (4, 2), (4, 3)} (při postupném slučování po jedné dvojici v libovolném pořadí), které se nakonec spojí ve výšce $\sqrt{2}$ (vzdálenost (2, 1)–(3, 2)). Řez dendrogramu na dvě skupiny dá *stejně dva shluky jako k-means*: {(1, 1), (2, 1)} a {(3, 2), (4, 2), (4, 3)}.



Data úlohy léto 2023. k -means s $k = 2$ konverguje na shluky $\{(1, 1), (2, 1)\}$ a $\{(3, 2), (4, 2), (4, 3)\}$ s centroidy $(1,5, 1)$ a $(\frac{11}{3}, \frac{7}{3})$ (žluté hvězdy). Single-linkage aglomerativní shlukování dává řezem na dvě skupiny totéž rozdělení.

Příklad — SZZ jaro 2024 č. 28: k -means se třemi shluky

(Reálná otázka SZZ: jaro 2024, č. 28 – Shlukování)

Zadání. Body $A(-2, 1)$, $B(-1, -1)$, $C(3, -2)$, $D(4, -1)$, $E(3, 3)$, $F(4, 4)$, $G(-3, 0)$, $H(4, 2)$, $I(4, -2)$, $J(6, 1)$; $k = 3$, počáteční středy A , F , J . (1) Hlavní kroky k -means. (2) Rozdělení a nové středy po první iteraci (euklidovská vzdálenost). (3) Změní se středy dál?

Řešení. (1) Inicializuj K středů; pak opakuj *přiřazení* (každý bod k nejbližšímu středu) a *přepočít* (střed = průměr bodů shluku), dokud se přiřazení mění.

(2) *Krok přiřazení* ke středům $A(-2, 1)$, $F(4, 4)$, $J(6, 1)$. Vzdálenosti levých bodů jsou nejmenší k A , horních pravých k F , dolních/krajně pravých k J . Např. $C(3, -2)$: k A je $\sqrt{25 + 9} = \sqrt{34}$, k F je $\sqrt{1 + 36} = \sqrt{37}$, k J je $\sqrt{9 + 9} = \sqrt{18}$ – nejbližší J ; $H(4, 2)$: k F je $\sqrt{0 + 4} = 2$, k J je $\sqrt{4 + 1} = \sqrt{5}$ – nejbližší F . Výsledek:

$$\text{shluk } A : \{A, B, G\}, \quad \text{shluk } F : \{E, F, H\}, \quad \text{shluk } J : \{C, D, I, J\}.$$

Krok přepočtu (průměry):

$$\mu_A = \left(\frac{-2-1-3}{3}, \frac{1-1+0}{3} \right) = \boxed{(-2, 0)}, \quad \mu_F = \left(\frac{3+4+4}{3}, \frac{3+4+2}{3} \right) = \boxed{\left(\frac{11}{3}, 3 \right)},$$

$$\mu_J = \left(\frac{3+4+4+6}{4}, \frac{-2-1-2+1}{4} \right) = \boxed{\left(\frac{17}{4}, -1 \right)}.$$

(3) Opakované přiřazení k novým středům dá *totéž* rozdělení, takže se středy už **nemění** – algoritmus zkonvergoval.

Příklad — SZZ léto 2025 č. 31: kritérium a lokální optimum

(Reálná otázka SZZ: léto 2025, č. 31 – Shlukování (k -means))

Zadání. (1) Popište k -means (volba a úprava středů, ukončení). (2) Jaké kritérium optimalizuje? (3) Na datech se třemi zjevnými shluky k -means ($k = 3$) nenašel optimální shluky – proč a jak zvýšit šanci na lepší výsledek?

Řešení. (1) Středy se volí (náhodně z dat / k -means++); pak se střídá přiřazení bodů k nejbližšímu středu a přepočít středů na průměr shluku; končí se, když se přiřazení přestane měnit (nebo po daném počtu iterací). (2) Minimalizuje $J = \sum_i \sum_k z_{i,k} \|\mathbf{x}_i - \mu_k\|^2$ (součet čtverců vzdáleností bodů od jejich centroidu).

(3) k -means konverguje jen k *lokálnímu* minimu J , závislému na inicializaci. Při nešťastné inicializaci se některý přirozený shluk rozdělí mezi dva středy, zatímco jiný přirozený shluk vlastní střed nedostane. Na obrázku ze zadání si velký levý dolní shluk rozdělily značky „čtvereček“ a „kolečko“, přičemž kolečka zároveň pokrývá malý horní prostřední shluk; oficiální náčrt to popisuje tak, že

prostřední shluk (*mnohem menší* než zbylé dva) byl přiřazen k části levého. Takové rozdělení je lokálně stabilní (žádný bod nechce přejít k jinému středu), ale má vyšší J než pravé tři shluky. **Náprava:** spustit k -means *vícekrát* z různých inicializací a vzít běh s nejmenším J ; použít k -means++ inicializaci, která rozprostře počáteční středy.

Příklad – SZZ jaro 2026 č. 28: k -means a argument konvergence

(Reálná otázka SZZ: jaro 2026, č. 28 – Shlukování pomocí K -means)

Zadáni. (1) Popište k -means, k čemu slouží, jeho kroky a inicializaci. (2) Jakou ztrátovou funkci optimalizuje? (3) Konverguje? Zformulujte argument a uveďte praktické důsledky.

Řešení. (1) k -means je metoda *učení bez učitele* pro rozdělení dat do K shluků podle geometrické podobnosti. Inicializuje K centroidů (náhodně z dat nebo k -means++: první náhodně, další úměrně čtverci vzdálenosti od nejbližšího už zvoleného) a opakuje přiřazení (bod $\rightarrow$ nejbližší centroid) a přepočít (centroid = průměr shluku), dokud se přiřazení mění. (2) $J = \sum_i \sum_k z_{i,k} \|\mathbf{x}_i - \mu_k\|^2$. (3) **Ano, konverguje k lokálnímu minimu.** Argument: oba kroky J *nezvýší* (přiřazení k nejbližšímu středu i přepočít na průměr jsou každý optimální pro svoji proměnnou); J je zdola omezené nulou; a hodnota J je určena *přiřazením*, kterých je jen *konečně mnoho* ($\leq K^N$). Nerostoucí posloupnost, která při každé skutečné změně přiřazení ostře klesá, se na konečné množině stavů musí po konečně mnoha krocích ustálit – tehdy se přiřazení nemění a zastavíme. **Praktické důsledky:** výsledek je jen *lokální* optimum a závisí na inicializaci, proto se algoritmus spouští *vícekrát* a bere se nejlepší (nejmenší J); inicializace k -means++ kvalitu zlepšuje. (Konvergence je obvykle rychlá – pár iterací.)

10.2 Hierarchické shlukování

k -means vyžaduje předem zvolené K a dává „placaté“ rozdělení. **Aglomerativní (zdola-nahoru) hierarchické shlukování** žádné K předem nepotřebuje: vybuduje celou *hierarchii* vnořených shluků a rozhodnutí o počtu skupin nechá až na konec (řez dendrogramu).

Algoritmus – Aglomerativní hierarchické shlukování

Vstup: body $\mathbf{x}_1, \dots, \mathbf{x}_N$, míra vzdálenosti shluků (*linkage*).

1. Začni s N shluky, každý bod sám jeden shluk.

▷ *singletons*

2. **opakuj**, dokud nezbude jeden shluk:

- (a) najdi dvojici shluků s *nejmenší* vzdáleností (dle zvoleného linkage);
- (b) sluč je do jednoho a zaznamenej výšku spojení = jejich vzdálenost.

Výstup: *dendrogram* – strom slučování s výškami.

Definice – Linkage: vzdálenost dvou shluků $\mathcal{A}, \mathcal{B}$

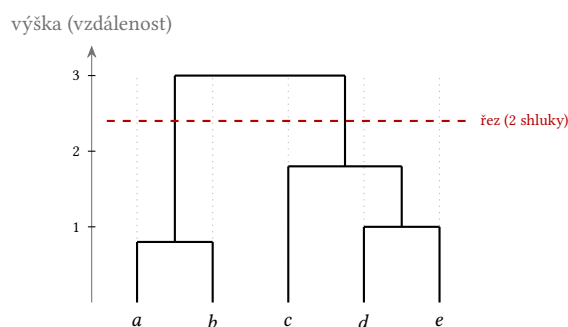
single (nejbližší soused) : $d(\mathcal{A}, \mathcal{B}) = \min_{a \in \mathcal{A}, b \in \mathcal{B}} \|a - b\|;$

complete (nejvzdálenější) : $d(\mathcal{A}, \mathcal{B}) = \max_{a \in \mathcal{A}, b \in \mathcal{B}} \|a - b\|;$

average (průměrná) : $d(\mathcal{A}, \mathcal{B}) = \frac{1}{|\mathcal{A}||\mathcal{B}|} \sum_{a,b} \|a - b\|;$

Ward : slučuj dvojici, která *nejméně* zvýší vnitroshlukový rozptyl.

💡 **Intuice.** Volba linkage mění *tvar* shluků: *single* umí natáhnout shluk podél řetězce blízkých bodů (riziko „řetězení“ – dva oddělené shluky spojené můstkem splynou), *complete* preferuje kompaktní kulaté shluky, *average* je kompromis. *Ward* minimalizuje růst rozptylu – dává podobně kompaktní shluky jako *k-means* a je výchozí volbou např. ve *scikit-learn*. **Dendrogram** je strom, jehož listy jsou body a výška spojení je vzdálenost slučovaných shluků. *Vodorovný řez* v určité výšce rozpadne strom na shluky: čím níž řežeme, tím *víc* shluků. Počet shluků tak volíme až dodatečně, čtením z dendrogramu.



Dendrogram aglomerativního shlukování pěti bodů. Nejdřív se spojí blízké páry (a, b) a (d, e) , pak se c přidá ke skupině $\{d, e\}$, nakonec splynou obě skupiny. Výška vodorovné spojnice = vzdálenost slučovaných shluků. Vodorovný řez (červeně) protne dvě svislé čáry, tedy dělí data na **dva** shluky $\{a, b\}$ a $\{c, d, e\}$; nižší řez by dal víc shluků.

❗ **Poznámka.** [Hustotní shlukování – jen zmínka] Existují i *hustotní* metody (DBSCAN): shluk = souvislá oblast s dostatečnou hustotou bodů, určenou parametry ϵ (poloměr) a minPts ; body v řídkých oblastech zůstanou jako šum. Umí najít shluky libovolného tvaru a nepotřebuje předem K , ale v požadavcích S3 jmenováno není (uvádíme jednou větou pro úplnost).

10.3 Redukce dimenze: analýza hlavních komponent (PCA)

Prokletí dimenzionality (sekce 2) i potřeba vizualizace motivují *redukci dimenze*: nahradit D příznaků menším počtem $M < D$ nových příznaků, které zachovají co nejvíc informace. **PCA** (analýza hlavních komponent) je lineární metoda, která nové osy volí ve směrech *největšího rozptylu* dat.

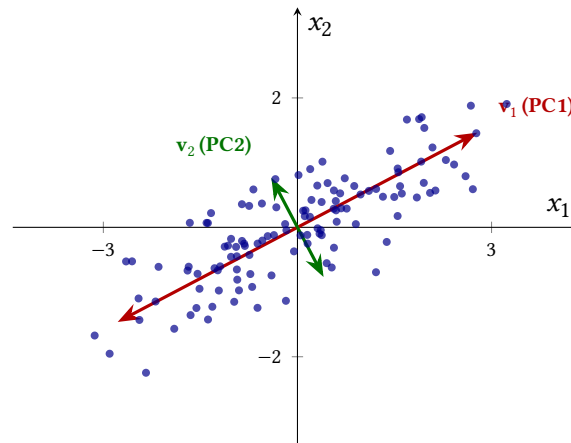
📖 Definice – PCA: hlavní komponenty

Mějme matici dat $\mathbf{X} \in \mathbb{R}^{N \times D}$ (příklad na řádku). PCA:

1. **Centrování:** odečti průměr, $\tilde{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{x}}$ (každý sloupec má střední hodnotu 0).
2. **Kovarianční matice:** $\mathbf{S} = \frac{1}{N} \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \in \mathbb{R}^{D \times D}$ (symetrická, pozitivně semidefinitní).
3. **Spektrální rozklad:** najdi vlastní čísla $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_D \geq 0$ a ortonormální vlastní vektory $\mathbf{v}_1, \dots, \mathbf{v}_D$ matice $\mathbf{S}$. *Hlavní komponenty* jsou vlastní vektory $\mathbf{v}_k$, seřazené sestupně podle λ_k .
4. **Projekce:** nové souřadnice bodu jsou jeho průměty na prvních M komponent, $\mathbf{z}_i = (\mathbf{v}_1^\top \tilde{\mathbf{x}}_i, \dots, \mathbf{v}_M^\top \tilde{\mathbf{x}}_i) \in \mathbb{R}^M$.

💡 **Intuice.** Vlastní číslo λ_k je přesně *rozptyl dat ve směru* $\mathbf{v}_k$ (kovarianční matice měří rozptyl, její vlastní směry jsou hlavní osy „mraku“ dat). První hlavní komponenta $\mathbf{v}_1$ míří tam, kde jsou data nejvíc rozprostřená; $\mathbf{v}_2$ je na ni kolmá s druhým největším rozptylem atd. Zahodit komponenty s malým λ znamená zahodit směry, v nichž se data skoro nemění – ztratíme *nejméně* informace (v smyslu rozptylu). *Kolik komponent vzít?* Podle *podílu zachyceného rozptylu* $\frac{\lambda_1 + \dots + \lambda_M}{\lambda_1 + \dots + \lambda_D}$ (např. tolik M , aby zachoval 95 %), nebo z „lomu“ v grafu seřazených λ (scree plot).

i Poznámka. [Souvislost se SVD a M2] PCA je vlastní rozklad symetrické pozitivně semidefinitní matice $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ (M2: taková matice má reálná nezáporná vlastní čísla a ortonormální vlastní vektory). Ekvivalentně lze hlavní komponenty získat z *singulárního rozkladu* $\tilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ (sloupce $\mathbf{V}$ jsou tytéž vlastní vektory, $\lambda_k = \sigma_k^2/N$); v praxi se počítá přes SVD, numericky stabilnější. Pro jen pár prvních komponent stačí *power iteration* (opakované násobení $\mathbf{S}$ s normalizací konverguje k dominantnímu vlastnímu vektoru). Detail SVD a Eckart-Youngovu větu (nejlepší nízkohodnotová aproximace) zde nerozvádíme – jen zmiňujeme.



PCA na protáhlém 2D mraku (centrovaná data). První hlavní komponenta $\mathbf{v}_1$ (červeně) míří směrem *největšího rozptylu*, druhá $\mathbf{v}_2$ (zeleně) je na ni kolmá s menším rozptylem. Zde $\mathbf{v}_1$ zachycuje přes 90 % rozptylu, takže projekci na samotnou $\mathbf{v}_1$ ($M = 1$) ztratíme jen málo informace. Komponenty jsou vlastní vektory kovarianční matice, seřazené dle vlastních čísel = rozptylů.

Řešený příklad: PCA a vliv na klasifikaci

Příklad – SZZ léto 2024 č. 29: směry komponent a PCA před k -NN

(Reálná otázka SZZ: léto 2024, č. 29 – PCA analýza)

Zadání. Data se dvěma atributy ze tří tříd (obrázek – tři protáhlé, vzájemně oddělené mraky podél „antidiagonály“, tj. směru klesající přímkou). (1) Jak funguje PCA, jak spočítat hlavní komponenty? (2) Bez počítání odhadněte směry obou hlavních komponent. (3) k -NN po redukci PCA na 1D – jaká accuracy (přesnost) oproti původním datům? Vysvětlete.

Řešení. (1) Viz výklad výše: centruj data, sestav kovarianční matici, její vlastní vektory (seřazené dle vlastních čísel) jsou hlavní komponenty; projekci na prvních M z nich snížíš dimenzi.

(2) Rozptyl dat je *největší podél antidiagonály* (mraky se táhnou a zároveň jsou rozmístěny ve směru klesající přímkou, přes celý graf). **1. hlavní komponenta $\mathbf{v}_1$ tedy míří podél této antidiagonály** (směr $\approx (1, -1)$, klesající), **2. komponenta $\mathbf{v}_2$ je na ni kolmá** (směr $\approx (1, 1)$, stoupající). Obě se zakreslí jako šipky z těžiště dat.

(3) **Accuracy klesne** (na původních 2D datech je k -NN téměř bezchybný, po projekci na 1D se třídy $+$ a $\bullet$ z velké části překryjí). Vysvětlení: PCA je *bez učitele* – maximalizuje rozptyl, **ne** oddělitelnost tříd. Projekce na $\mathbf{v}_1$ (antidiagonála) sice zachová nejvíc rozptylu, ale *stlačí dohromady* třídy, které se liší hlavně v *kolmém* (málorozptylovém) směru $\mathbf{v}_2$ – jejich průměty na $\mathbf{v}_1$ se překryjí a k -NN je splete. Redukce na 1D vždy zahodí jednu souřadnici, takže může smazat právě tu informaci, která třídy dělila. (Projekce na málorozptylovou $\mathbf{v}_2$ by tu nebyla lepší: v ní se zase překryjí třídy $\times$ a $\bullet$, jejichž součty souřadnic $x_1 + x_2$ jsou podobné – ukázka, že „více rozptylu“ $\neq$ „lepší pro klasifikaci“.)

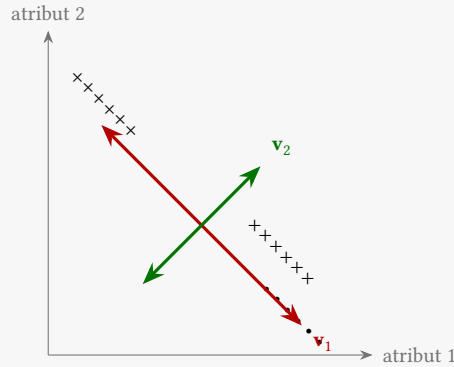


Schéma dat úlohy léto 2024: tři protáhlé třídy (x, +, •) podél antidiagonály. Odhadnuté hlavní komponenty: $\mathbf{v}_1$ (červeně) podél směru největšího rozptylu (klesající antidiagonála), $\mathbf{v}_2$ (zeleně) kolmá. Projekce na samotnou $\mathbf{v}_1$ překryje třídy + a • (liši se kolmo), proto k -NN po redukci na 1D ztrácí accuracy.

10.4 Měkké shlukování: EM a směs Gaussovců

i Poznámka. [Zařazení] EM a směs Gaussovců nejsou v požadavcích S3 jmenovány přímo, ale jde o *pravděpodobnostní zobecnění k -means* (měkké místo tvrdého přiřazení) a okruh má na ně reálnou SZZ otázku; uvádíme je proto stručně a vyřešíme otázku. Obecný princip EM (E-krok/M-krok pro modely se skrytými proměnnými) je vyložen v S1.

k -means přiřazuje bod *tvrdě* k jedinému shluku. **Směs Gaussovců** (Gaussian mixture) modeluje data jako směs K normálních rozdělání a přiřazuje *měkce*: každý bod patří do shluku k s pravděpodobností (zodpovědností) $r_{i,k} \in [0, 1]$, $\sum_k r_{i,k} = 1$. Navíc dovolí shlukům *různou velikost i elipsovitý tvar* (vlastní kovarianční matici), kdežto k -means implicitně předpokládá kulové shluky podobné velikosti. Parametry (váhy, středy, kovariance) se učí **EM algoritmem**:

- **E-krok:** při současných parametrech spočti zodpovědnosti $r_{i,k} = P(\text{shluk } k \mid \mathbf{x}_i)$ (Bayesova věta).
- **M-krok:** přepočti parametry jako *vážené* (zodpovědnostmi) odhady – vážené průměry a kovariance.

k -means je limitní případ EM se sférickými Gaussovcí nekonečně malého rozptylu, kde zodpovědnosti zkolabují na 0/1 (tvrdé přiřazení).

Příklad — SZZ léto 2025 č. 32: EM na bayesovské síti

(Reálná otázka SZZ: léto 2025, č. 32 – EM algoritmus)

Zadání. Bayesovská síť: skrytá $Cluster \in \{1, 2\}$ je rodičem $Color \in \{\text{blue, green}\}$ a $Holes \in \{\text{yes, no}\}$ (mezi $Color$ a $Holes$ hrana není). (1) Popište EM. (2) Určete parametry, inicializujte z $\{0, 4, 0, 6\}$. (3) Proveďte jednu EM aktualizaci jednoho parametru na datech (řádky $Color, Holes, Cluster$ chybí).

Řešení. (1) EM odhaduje parametry modelu se *skrytou* proměnnou (zde $Cluster$) střídáním E-kroku (spočti očekávané hodnoty / zodpovědnosti skryté proměnné při daných parametrech) a M-kroku (přepočti parametry maximalizací věrohodnosti na „doplňených“ datech), dokud parametry nezkonvergují.

(2) Model má 5 nezávislých parametrů; inicializujeme z $\{0, 4, 0, 6\}$ tak, aby podmínky pro $Cluster=1$ a $Cluster=2$ nebyly stejné:

$$\begin{aligned} \theta = P(Cluster=1) = 0,6, \quad \theta_{B1} = P(\text{blue} \mid 1) = 0,6, \quad \theta_{B2} = P(\text{blue} \mid 2) = 0,4, \\ \theta_{H1} = P(\text{yes} \mid 1) = 0,6, \quad \theta_{H2} = P(\text{yes} \mid 2) = 0,4. \end{aligned}$$

(3) Aktualizujeme nejjednodušší parametr $\theta = P(Cluster=1)$. *E-krok* – pro každý řádek spočti zodpovědnost $P(Cluster=1 \mid \text{data})$ Bayesovou větou (faktorizace dle sítě: $P(C, \text{col}, \text{hol}) = P(C)P(\text{col} \mid C)P(\text{hol} \mid C)$).

C) $P(\text{hol} | C)$:

$$P(C=1 | \text{col}, \text{hol}) = \frac{\theta P(\text{col} | 1)P(\text{hol} | 1)}{\theta P(\text{col} | 1)P(\text{hol} | 1) + (1 - \theta) P(\text{col} | 2)P(\text{hol} | 2)}.$$

Po řádcích (čitatel = $\theta P(\text{col} | 1)P(\text{hol} | 1)$, jmenovatel = $\theta P(\text{col} | 1)P(\text{hol} | 1) + (1 - \theta) P(\text{col} | 2)P(\text{hol} | 2)$) – blue,no: $0,6 \cdot 0,6 \cdot 0,4 = 0,144$ vs. $0,4 \cdot 0,4 \cdot 0,6 = 0,096$; blue,yes: $0,6 \cdot 0,6 \cdot 0,6 = 0,216$ vs. $0,4 \cdot 0,4 \cdot 0,4 = 0,064$; green,yes: $0,6 \cdot 0,4 \cdot 0,6 = 0,144$ vs. $0,4 \cdot 0,6 \cdot 0,4 = 0,096$:

Color	Holes	$P(C=1 \text{data})$
blue	no	$\frac{0,144}{0,144 + 0,096} = 0,6$
blue	yes	$\frac{0,216}{0,216 + 0,064} \approx 0,77$
blue	no	$0,6$
green	yes	$\frac{0,144}{0,144 + 0,096} = 0,6$
green	yes	$0,6$

M-krok – nové θ je průměr zodpovědností (očekávaný podíl příkladů ve shluku 1):

$$\theta^{\text{nové}} = \frac{0,6 + 0,77 + 0,6 + 0,6 + 0,6}{5} = \frac{3,17}{5} = \boxed{0,634}.$$

(Číslo přesně: $\frac{4 \cdot 0,6 + 0,7714}{5} = 0,6343$.) Ostatní parametry by se aktualizovaly analogicky – váženými četnostmi $\sum_j r_j [\dots]$ místo tvrdých počtů.

Shrnutí sekce

Shlukování (k-means) minimalizuje $J = \sum_i \sum_k z_{i,k} \|\mathbf{x}_i - \mu_k\|^2$ střídáním přiřazení (bod $\rightarrow$ nejbližší střed) a přepočtu (střed = průměr shluku); konverguje k *lokálnímu* minimu (oba kroky J nezvyšují, přiřazení je konečně mnoho), proto se spouští vícekrát / s k-means++. **Hierarchické** (aglomerativní) shlukování buduje dendrogram slučováním nejbližších shluků (linkage single/complete/average/Ward); počet shluků určí řez dendrogramu. **PCA** centruje data, vezme vlastní vektory kovarianční matice (hlavní komponenty seřazené dle vlastních čísel = rozptylů) a projektuje na prvních M – pozor, maximalizuje rozptyl, ne oddělitelnost tříd. **EM / směs Gaussovců** je měkké, pravděpodobnostní zobecnění k-means.

Shrnutí

i Poznámka. Klíčové vlastnosti a vzorce (přehled). Stručný výčet klíčových definic, vět a vzorců okruhu S3. Anglická terminologie tam, kde čeština není běžná. Zkratky: TP/TN/FP/FN = true/false positive/negative; D = počet příznaků; N = počet příkladů; $\mathbf{w}$ = váhy; α = krok SGD; λ = regularizační síla; $\hat{y}$ = predikce modelu.

Evaluace klasifikátoru.

- Matice záměn:** řádky = skutečná třída, sloupce = predikovaná; TP (správně pozitivní), TN (správně negativní), FP (falešně pozitivní), FN (falešně negativní).
- accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$. Pro nevyvážené třídy **nevhodná**; používej precision/recall/F1 nebo AUC.
- precision = $\frac{TP}{TP + FP}$; recall = sensitivity = TPR = $\frac{TP}{TP + FN}$; $F_1 = \frac{2 TP}{2 TP + FP + FN} = \frac{2 \cdot \text{prec} \cdot \text{recall}}{\text{prec} + \text{recall}}$.

- specificity = TNR = $\frac{TN}{TN + FP}$; FPR = 1 - specificity. **ROC**: TPR vs. FPR při plynulém posouvání prahu; **AUC** = plocha pod ROC (0,5 = náhoda, 1,0 = perfektní).
- **Regresní metriky**: MSE = $\frac{1}{N} \sum_i (y_i - t_i)^2$; RMSE = $\sqrt{\text{MSE}}$; MAE = $\frac{1}{N} \sum_i |y_i - t_i|$.
- **Křížová validace** (k -fold): data dělí na k foldů; každý slouží jednou jako validační, ostatní jako trénovací; průměr k chyb = odhad generalizační chyby. Leave-one-out: $k = N$.
- **MLE**: $\mathbf{w}_{\text{MLE}} = \text{argmin}_{\mathbf{w}} \sum_i (-\log p(t_i | \mathbf{x}_i; \mathbf{w}))$; pro normální šum $\Rightarrow$ MSE, pro binární klasifikaci $\Rightarrow$ cross-entropy.
- **Naivní Bayes**: $\hat{c} = \text{argmax}_c P(c) \prod_j P(x_j | c)$; předpoklad podmíněné nezávislosti příznaků dané třídou; $P(c)$ a $P(x_j | c)$ odhadujeme z dat četnostmi.

Overfitting, regularizace a prokletí dimenzionality.

- **Overfitting**: model se naučí šum; trénovací chyba malá, generalizační velká. **Underfitting**: model slabý; obě chyby velké. *U-křivka*: generalizační chyba má minimum pro optimální kapacitu.
- **L2 regularizace** (ridge, Tikhonov): $E(\mathbf{w}) = E_0(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$; penalizuje velké váhy, dává hladké řešení; analyticky pro lin. regresi: $\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$.
- **L1 regularizace** (LASSO): $E(\mathbf{w}) = E_0(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$; tlačí váhy přesně na nulu $\Rightarrow$ řídká řešení (feature selection). Geometricky: L1 koule (v 2D kosočtverec) má rohy na osách, L2 koule je hladká – proto L1 řešení typicky padne do rohu (nulová souřadnice).
- **Early stopping**: sleduj validační chybu; zastav trénování ve chvíli, kdy začne na validaci růst (model se začíná přeučovat). Hyperparametr je počet trpělivostních epoch (*patience*).
- **Prokletí dimenzionality**: objem prostoru roste exponenciálně s D – pokrýt $[0, 1]^D$ mřížkou s krokem r vyžaduje $N \propto r^{-D}$ příkladů; objem se koncentruje u povrchu; vzdálenosti ztrácejí kontrast (k -NN degraduje); nutná regularizace nebo redukce dimenze.

k-NN (k nejblížešších sousedů).

- **Líný/neparametrický**: trénování = uložení dat; inference = najdi k nejblížešších sousedů. *Klasifikace* = nejčastější třída, *regrese* = průměr cílů k sousedů.
- Metrika: L^p (L^1 manhattanská, L^2 euklidovská). Nutná **normalizace příznaků** (různé škály deformují vzdálenosti). Volba k křížovou validací: malé $k \Rightarrow$ overfitting, velké $k \Rightarrow$ underfitting.

Lineární regrese.

- Model: $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$; minimalizujeme MSE = $\frac{1}{N} \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2$ (ekvivalentně součet čtverců).
- **Normální rovnice** (analytické řešení): $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$; s L2: $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$. Přesné, ale pomalé ($\mathcal{O}(ND^2 + D^3)$), neumí early stopping.
- **SGD krok** (jeden příklad $(\mathbf{x}, t)$): $\mathbf{w} \leftarrow \mathbf{w} - \alpha[(\mathbf{w}^T \mathbf{x} - t) \mathbf{x} + \lambda \mathbf{w}]$; bias bez L2: $b \leftarrow b - \alpha(\hat{y} - t)$. Škálovatelné, umí early stopping.

Logistická regrese.

- **Binární**: $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b) = P(C_1 | \mathbf{x})$; $\sigma(z) = \frac{1}{1+e^{-z}} \in (0, 1)$; prahování na 0,5; $P(C_0) = 1 - \hat{y}$.
- **Cross-entropy ztráta** (NLL Bernoulliho): $E = -\sum_i [t_i \log \hat{y}_i + (1 - t_i) \log(1 - \hat{y}_i)]$.
- **Gradient/SGD krok**: $\nabla_{\mathbf{w}} E_i = (\hat{y}_i - t_i) \mathbf{x}_i$; $\mathbf{w} \leftarrow \mathbf{w} - \alpha(\hat{y} - t) \mathbf{x}$ (tvar identický s lin. regresí, liší se jen $\hat{y}$).
- **Multiclass (softmax)**: K tříd, K vektorů vah $\mathbf{w}_k$; $p_k = \frac{e^{\mathbf{w}_k^T \mathbf{x}}}{\sum_j e^{\mathbf{w}_j^T \mathbf{x}}}$; ztráta = categorical cross-entropy – $\sum_i \log p_{t_i}(\mathbf{x}_i)$.

Rozhodovací stromy.

- **Stavba**: hladově shora dolů (greedy, rozděl a panuj); v každém uzlu vyber řez $(j^*, v^*) = \text{argmin}_{j,v} [c(\mathcal{T}_L) + c(\mathcal{T}_R)]$; rekurze do zastavovací podmínky (max. hloubka, min. příkladů). Predikce listu: nejčastější třída (klas.) nebo průměr cílů (reg.).
- **Entropie**: $H(S) = -\sum_k p_k \log_2 p_k$ (bity; $0 \log 0 := 0$; čistý uzel $\Rightarrow H = 0$).
- **Informační zisk**: $\text{IG} = H(\mathcal{T}) - \frac{|\mathcal{T}_L|}{|\mathcal{T}|} H(\mathcal{T}_L) - \frac{|\mathcal{T}_R|}{|\mathcal{T}|} H(\mathcal{T}_R)$; volíme řez s maximálním IG.
- **Gini index**: $\text{Gini}(S) = \sum_k p_k(1 - p_k) = 1 - \sum_k p_k^2$; rychlejší výpočet, podobné výsledky jako

entropie; pro regresní stromy používáme vážené MSE (rozptyl cílů v uzlu).

- **Prořezávání** (pruning): neomezený strom overfittuje. *Post-pruning* (cost-complexity): $C_\alpha(T) = \text{chyba}(T) + \alpha \cdot |\text{listy}(T)|$; ladí se α na validaci. *Pre-pruning*: max. hloubka, min. příkladů v uzlu.

SVM (metoda podpůrných vektorů).

- **Hard-margin** (lineárně separabilní): $\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$ za $t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$; cílové hodnoty $t_i \in \{-1, +1\}$; margin = $\frac{1}{\|\mathbf{w}\|}$, šíře pásu = $\frac{2}{\|\mathbf{w}\|}$. **Podpůrné vektory**: trénovací body na hranici pásu ($t_i \cdot y(\mathbf{x}_i) = 1$).
- **Soft-margin** (neseperabilní): $\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i$ za $t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$. C velké $\Rightarrow$ menší margin, méně klasifikačních chyb; C malé $\Rightarrow$ větší margin, více chyb.
- **Kernel trick**: nahraď $\mathbf{x}_i^\top \mathbf{x}_j$ jádrem $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j)$ – výpočet K bez explicitní φ . **RBF**: $K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2}$. **Polynomiální**: $(1 + \mathbf{x}^\top \mathbf{x}')^d$.
- **Multiclass**: one-vs-rest – K binárních SVM, predikce = třída s nejvyšším $y_k(\mathbf{x})$; one-vs-one – $\binom{K}{2}$ SVM pro dvojice tříd, predikce hlasováním (1 libsvm).

Kombinace více modelů (ensemble).

- **Bagging** (bootstrap aggregating): M kopií modelu, každá na vlastním bootstrapovém vzorku (N příkladů s opakováním); predikce průměrem (reg.) nebo hlasováním (klas.). Snižuje rozptyl; out-of-bag příklady ($\approx 37\%$) $\Rightarrow$ zdarma validace.
- **Boosting**: sekvenční – každý model se zaměřil na příklady, které předchozí chybovaly. **AdaBoost**: hlas slabého klasifikátoru $\alpha_m = \frac{1}{2} \ln \frac{1 - \varepsilon_m}{\varepsilon_m}$; špatně klasifikovaným příkladům zvýší váhy w_i ; finální $\hat{y} = \text{sign}(\sum_m \alpha_m h_m(\mathbf{x}))$.
- **Random forest**: bagging rozhodovacích stromů + v každém uzlu nejlepší řez z náhodné podmnožiny příznaků ($\approx \sqrt{D}$, typicky u klasifikace); stromy se nechávají přerůst; snižuje korelaci chyb stromů.
- Ensemble zlepšil pouze při **nekorelovaných chybách** modelů; jsou-li chyby totožné, kombinace nepomůže.

Statistické testy.

- **Jednovýběrový t-test**: $H_0 : \mu = \mu_0$; testová statistika $t = \frac{\bar{X} - \mu_0}{s/\sqrt{n}}$ (s = výběrová směrodatná odchylka); pod H_0 platí $t \sim t_{n-1}$. Zamítáme H_0 , padne-li $|t|$ do kritického oboru (tj. $|t| > t_{n-1, 1-\alpha/2}$, nebo $p\text{-value} < \alpha$).
- **Dvouvýběrový t-test**: $H_0 : \mu_1 = \mu_2$; při shodných rozptylech $t = \frac{\bar{X} - \bar{Y}}{s_p \sqrt{1/n_1 + 1/n_2}}$, $\text{df} = n_1 + n_2 - 2$, kde $s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1 + n_2 - 2}$; Welchova varianta (nestejné rozptyly): jiný jmenovatel a df.
- χ^2 **test dobré shody**: H_0 : pozorované četnosti odpovídají očekávaným; $\chi^2 = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i}$; pod H_0 platí $\chi^2 \sim \chi_{k-1}^2$ (k kategorií). Zamítáme, je-li $\chi^2 >$ kritická hodnota (nebo $p < \alpha$).

Učení bez učitele: shlukování a redukce dimenze.

- **k-means** (Lloydův algoritmus): střídá *přiřazení* (každý bod k nejbližšímu centroidu) a *přepočty* (centroid = průměr shluku); kritérium $J = \sum_i \sum_k z_{i,k} \|\mathbf{x}_i - \mu_k\|^2$. Konverguje k *lokálnímu* minimu (J v žádném kroku neroste; konečně mnoho přiřazení $\Rightarrow$ zastaví se). Spouštět vícekrát, vzít nejmenší J .
- **k-means++**: první střed náhodně; každý další s pravděpodobností $\propto$ čtverci vzdálenosti od nejbližšího dosud zvoleného – lepší inicializace, $\mathcal{O}(\log K)$ aproximace optima v očekávání.
- **Hierarchické shlukování** (aglomerativní, zdola nahoru): začni se N singletony; opakovaně slučuj dvojici s nejmenší vzdáleností; výsledkem je **dendrogram** – vodorovný řez určuje počet shluků. Linkage: *single* (min vzdálenost; riziko řetězení), *complete* (max vzdálenost; kompaktní shluky), *average*, *Ward* (min nárůst vnitroshlukového rozptylu; výchozí ve scikit-learn).
- **PCA** (analýza hlavních komponent): (1) centruj data $\tilde{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{x}}$; (2) kovarianční matice $\mathbf{S} =$

$\frac{1}{N} \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$; (3) vlastní vektory $\mathbf{v}_1, \dots, \mathbf{v}_D$ seřazené dle $\lambda_1 \geq \dots \geq \lambda_D \geq 0$ jsou hlavní komponenty; (4) projekce: $\mathbf{z}_i = (\mathbf{v}_1^\top \tilde{\mathbf{x}}_i, \dots, \mathbf{v}_M^\top \tilde{\mathbf{x}}_i) \in \mathbb{R}^M$. Vlastní číslo λ_k = rozptyl dat ve směru $\mathbf{v}_k$; zachovaný podíl = $\frac{\sum_{k=1}^M \lambda_k}{\sum_{k=1}^D \lambda_k}$. PCA je bez učitele – maximalizuje rozptyl, *ne* oddělitelnost tříd; po projekci může accuracy k -NN klesnout.

Zdroje

- M. Straka, J. Libovický: slidy přednášek předmětu NPFL129 *Úvod do strojového učení v Pythonu* (Introduction to Machine Learning with Python), ÚFAL MFF UK, ročník 2024/25 (primární zdroj výkladu; téma SVM z ročníku 2022/23, kdy bylo součástí osnovy).
<https://ufal.mff.cuni.cz/courses/npfl129>
- M. Straka a kol.: zdrojový repozitář kurzu NPFL129 (Markdown zdroj slidů a generátory obrázků, včetně archivních ročníků).
<https://github.com/ufal/npfl129>
- R. Šámal: *NMAI059 Pravděpodobnost a statistika 1*, skripta k přednášce, MFF UK, verze 14. 6. 2025 (rámeček testování hypotéz, t-test a χ^2 test; podrobně v okruhu M5).
<https://iuuk.mff.cuni.cz/~samal/vyuka/2425/PSt1/skripta.pdf>
- MFF UK: požadavky ke státní závěrečné zkoušce (Informatika, Bc.) a zadání otázek z minulých termínů.
<https://www.mff.cuni.cz/cs/studenti/bakalarske-studium/statni-zaverecne-zkousky/bakalarske-statni-zkousky-studijniho-programu-informatika>