BI-VZD přednáška 5

Alexander Kovalenko

FIT ČVUT

14. 03. 2022

Autoři: Karel Klouda, Juan Pablo Maldonado Lopez, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 14. března 2022 09:25.

Co bude v dnešní přednášce

- metoda nejbližších sousedů (KNearestNeighbors, kNN)
- různé míry vzdáleností
- kNN a normalizace dat
- prokletí dimenzionality (the curse of dimensionality)
- křížová validace

kNN: základní myšlenka

- Řešíme problém *supervizovaného učení*, máme tedy trénovací data $X \in \mathbb{R}^{N,p}$ se známými hodnotami vysvětlované proměnné $Y \in \mathbb{R}^N$.

- Základní (a velice jednoduchá) myšlenka kNN je následující:
  - Chceme predikovat hodnotu vysvětlované proměnné pro datový bod $\boldsymbol{x} \in \mathbb{R}^p$.
  - V trénovacích datech najdeme k bodů (k je zadaný hyperparametr), které mají od $\boldsymbol{x}$ nejmenší vzdálenost.
  - Predikci pak založíme na známých hodnotách vysvětlované proměnné pro těchto k bodů.
  - Jedná-li se o regresi (spojité Y), bereme průměr z hodnot pro tyto body.
  - Jedná-li se o klasifikaci, bereme nejčastější hodnotu mezi těmito body.

Nejbližší soused: jak ho najít

Klíčový je pojem vzdálenosti (resp. metriky), který jsme již ale probírali na minulé přednášce:

Definice

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

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

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

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

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

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 = d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=0}^{p-1} (x_i - y_i)^2},$$

  pro dva body $\boldsymbol{x} = (x_0, x_1, \ldots, x_{p-1})$ a $\boldsymbol{y} = (y_0, y_1, \ldots, y_{p-1})$ z $\mathbb{R}^p$.
- Poději si ukážeme i jiné možnosti.

kNN: učení vs. predikování

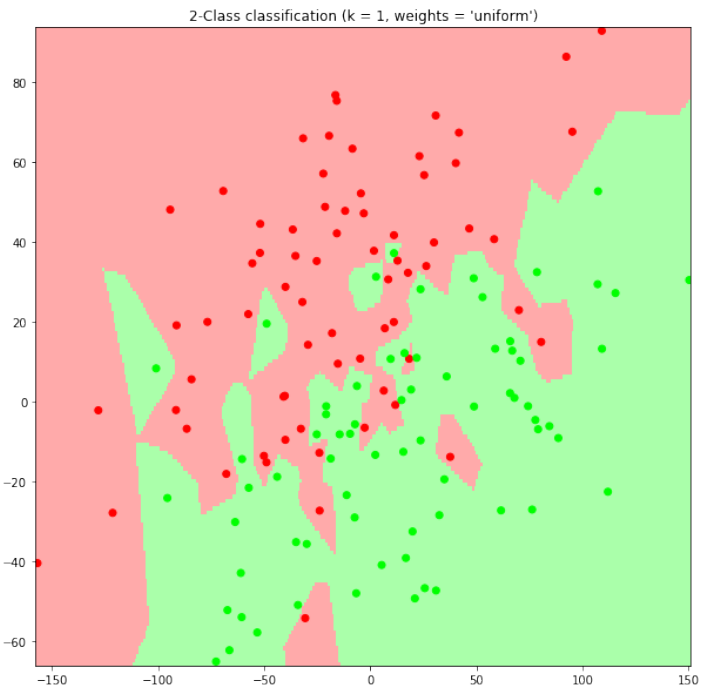
- U rozhodovacích stromů byla výpočetně náročná fáze učení, ale predikce už se konstruovaly velmi snadno a rychle (šlo jen o průchod nehlubokým stromem).
- Toto platí skoro o všech metodách pro supervizované učení: řádově více je náročné učení modelu než výpočet predikcí.
- U kNN je tomu naopak! Učení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

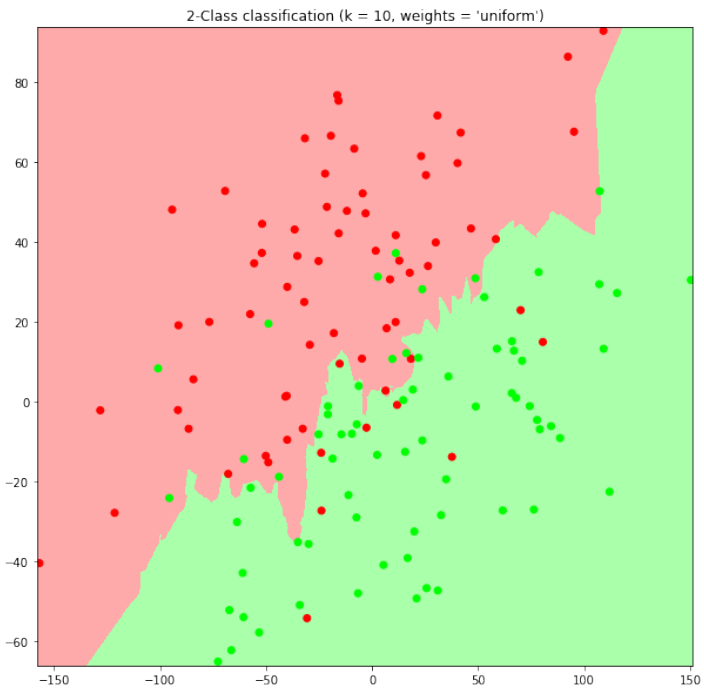
kNN: hyperparametry

- Implementace kNN v `scikit-learn` je dostupná v balíčcích KNeighborsRegressor a KNeighborsClassifier.
- V případě regrese i klasifikace jsou pro kNN smysluplné tři hyperparametry:
  - Číslo k určující počet hledaných nejbližších sousedů (`n_neighbors`).
  - Použitá vzdálenost; obecně funkce vracejícím dvěma datovým bodům číslo (`metric`).
  - Váhy nejbližších sousedů určující „sílu jejich hlasu“ při predikci (`weights`).
- Postupně si tyto tři hyperparametry projdeme.
- Implementace kNN v `scikit-learn` nabízí ještě další parametry, ale ty se už týkají spíše způsobu výpočtu a tvorby případného indexu, takže neovlivňují přímo tvar modelu a nejedná se tedy striktně řečeno o hyperparametry.

kNN: hyperparametr k (počet sousedů)

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





kNN: míra vzdálenosti jako hyperparametr (1/2)

- Předpokládejme, že všechny naše příznaky jsou čísla; ideálně, že se jedná o spojité numerické příznaky.
- Potom lze jako vzdálenost vzít jakoukoli metriku definovanou na $\mathbb{R}^p$ (kde p je počet příznaků).
- Nejobvyklejší volbou jsou tzv. L_k metriky (také Minkovského k-metriky příp. k-normy), kde $k = 1, 2, \ldots$ Tyto jsou také přímo podporované v `scikit-learn`.
- Vzdálenost dvou bodů $\boldsymbol{x} = (x_0, x_1, \ldots, x_{p-1})$ a $\boldsymbol{y} = (y_0, y_1, \ldots, y_{p-1})$ z $\mathbb{R}^p$ dané L_k metrikou je rovna

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

  speciálně pro $k = 1$ dostáváme

$$\|\boldsymbol{x} - \boldsymbol{y}\|_1 = d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{p-1} |x_i - y_i|.$$

- Euklidova vzdálenost tak odpovídá $k = 2$, Manhattanská vzdálenost pak $k = 1$.

kNN: míra vzdálenosti jako hyperparametr (2/2)

- Vzdálenost si můžeme ale definovat v podstatě libovolně, jde jen o to, aby byla schopná vrátit jednoznačně určené číslo pro dva datové body (tj. dva vektory příznaků).
- Lze také definovat i vzdálenosti, které se v každé dimenzi chovají jinak: např. jedná-li se o příznak spojitý numerický, přispěje do vzdálenosti absolutní hodnotou rozdílu, jedná-li se např. o jméno, lze použít Levenshteinovu vzdálenost, příp. lze na podmnožinu příznaků použít třeba cosinovou vzdálenost

$$\frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2}.$$

- Volba sofistikované míry vzdálenosti může z jednoduchého kNN modelu udělat model složitý (a třeba i mocný). Obvykle také významně vzroste počet hyperparametrů, které určují, jak přesně daná vzdálenost vypadá (zejm. jakou váhu mají jednotlivé složky).
- Více se lze dočíst například zde: Similarity Measures for Categorical Data: A Comparative Evaluation (2008).

kNN: hyperparametr `weights`

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

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

- Často se ale volí vážený průměr

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

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

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

Různé váhy stejných příznaků

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

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

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

Často nedosažitelná souměřitelnost příznaků

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

Normalizace dat (1/2)

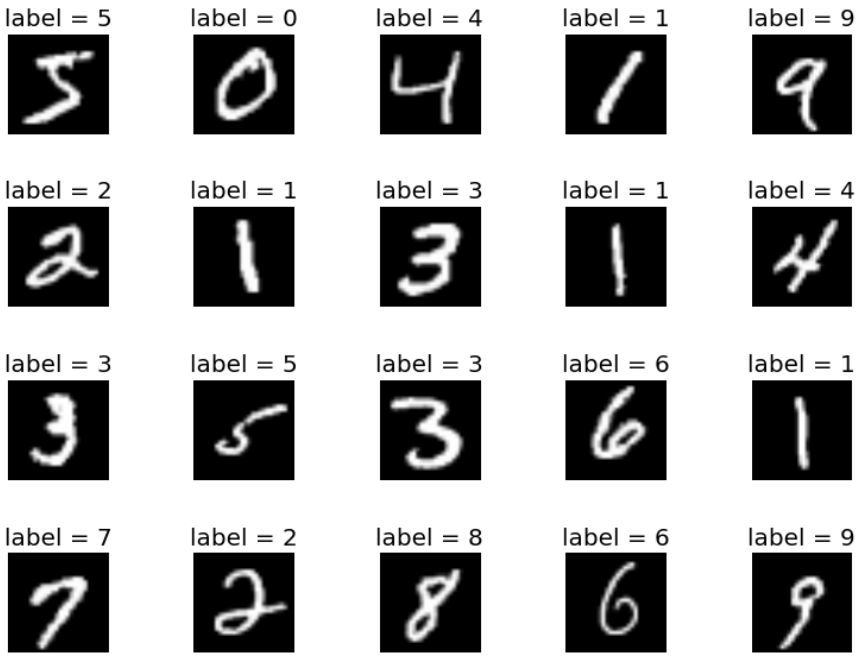
- Jednoduchou metodou, jak zabránit nejextrémnějším úletům způsobeným pestrou škálou příznaků a jejich významů je **normalizace každého příznaku** do intevalu $[0, 1]$.
- Postup je následující: pro každý příznak najdeme jeho minimální a maximální hodnotu v trénovacích datech: $\min_x, \max_x$ a pak hodnotu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

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

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

Normalizace dat (2/2)

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



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

Normalizace dat (2/2)

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

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

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

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

[1] Jak to tak chodí.

kNN a nominální příznaky

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

Prokletí dimenzionality

- Prokletí dimenzionality (angl. **the curse of dimensionality**), je pojem, který odkazuje na některé problémy objevující se v případě vysokého počtu příznaků, kdy jsou datové body prvky mnohadimenzionálního prostoru.
- Tyto problémy hrají svoji roli v mnoha metodách a proto jsou **redukce dimenzionality** a výběr příznaků (angl. **feature selection**) jedním ze zásadních témat při zpracování dat (budeme se jim věnovat v 10. přednášce a v jiných předmětech).
- S kNN jsou spojeny zejména dva efekty způsobené vysokou dimenzí dat:
  - Data se zvyšováním dimenzí řídnou a navzájem se vzdalují. Pro zachování stejné hustoty pro vyšší dimenzi by bylo nutné řádově navýšit počet datových bodů, což není obvykle možné.
  - S rostoucí dimenzí se pro klasické metriky zmenšují rozdíly mezi vzdálenými a blízkými body.
- První efekt si demonstrujeme na příkladu. Tomuto tématu se budeme věnovat také na cvičení.

Prokletí dimenzionality: řídnutí bodů

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

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

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

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

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

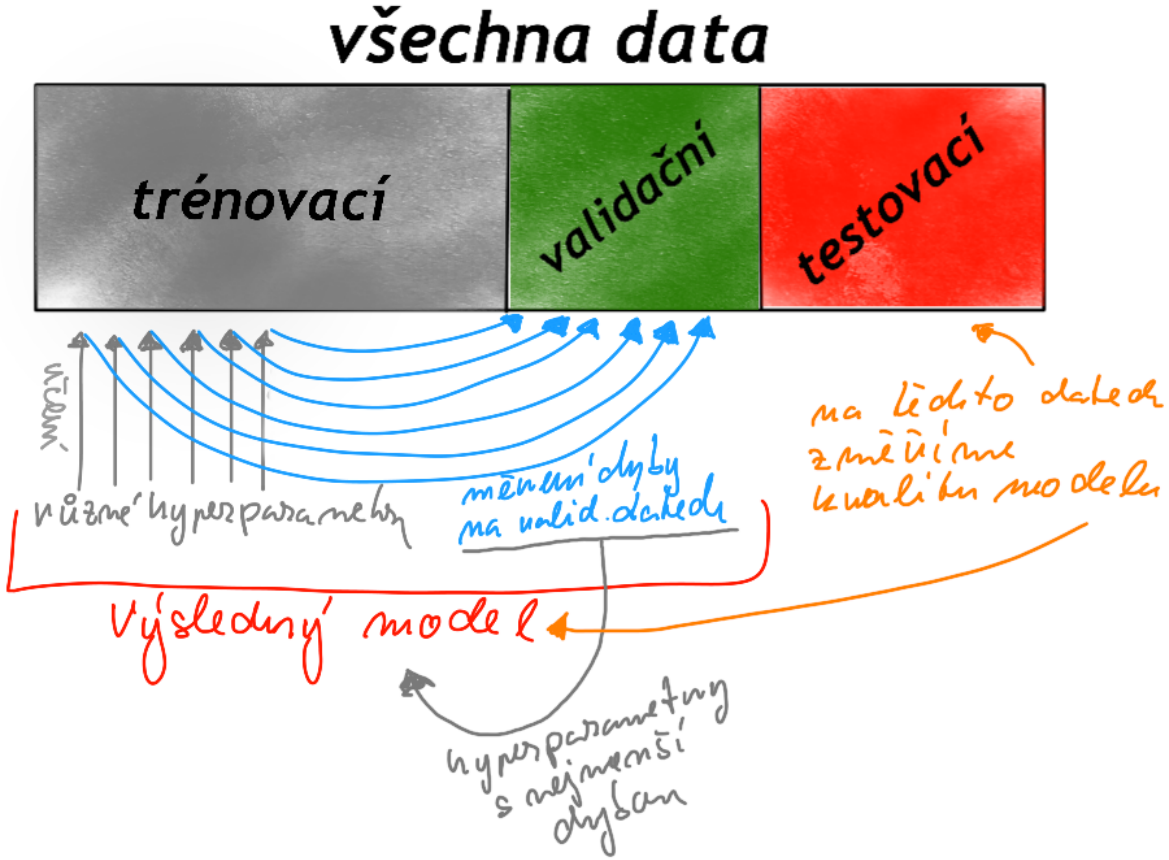
- V dimenzi d je to pak krychle o straně

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

pro $d = 10$ je $a \approx 0,63$ a pro $d = 50$ je $a \approx 0,91$.

Ladění hyperparametrů: připomenutí

Dříve jsme si řekli, že pro správné ladění hyperparametrů je dobré si rozdělit data na trénovací, testovací a validační množinu. Křížová validace je lehce odlišnou strategií, jak řešit tento problém.



Cross-validace: jak to funguje

- Data (náhodně) rozdělíme na trénovací a testovací, tak jak jsme popisovali dříve (např. v poměru 1 ku 4 či 1 ku 5).
- Používáme-li tzv. k-**fold Cross-Validation**, kde k je alespoň 2 a nejvýše rovno počtu bodů v trénovací množině, rozdělíme si (náhodně) trénovací množinu na k *podobně* velkých částí, které označíme $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$.
- Model (rozhodovací strom, kNN, atp.) pro dané hodnoty hyperparametrů postupně pro $j = 1, 2, \ldots, k$ natrénujeme na datech z množiny

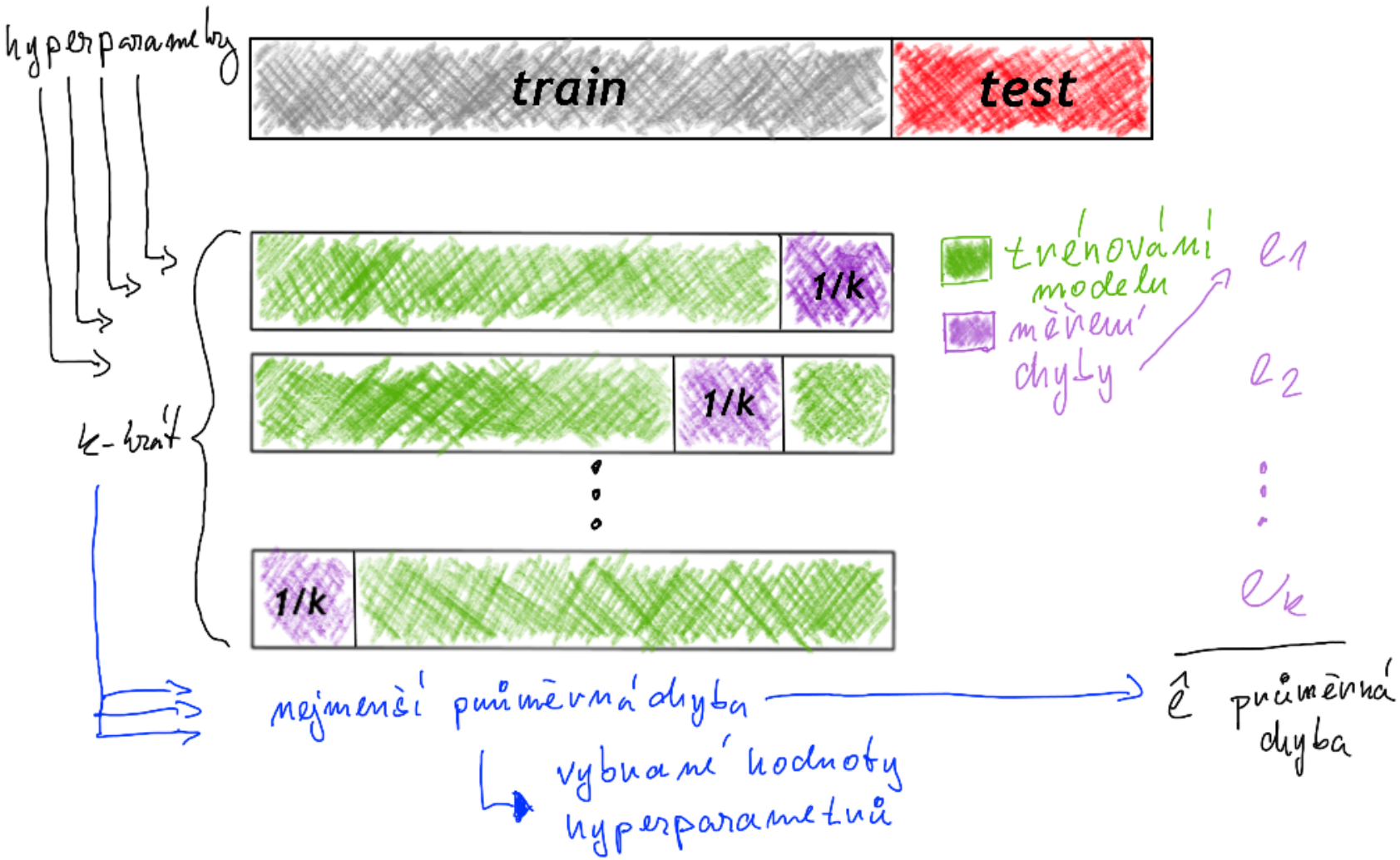
$$\left(\cup_{i=1}^{k} \mathcal{D}_i\right) \setminus \mathcal{D}_j$$

  a změříme chybu (MSE, klasifikační (ne)přesnost, atp.) na množině $\mathcal{D}_j$.
- Chybu pro každé j uložíme jako e_j a následně vrátíme průměrnou „cross-validační“ chybu

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

- Toto uděláme pro všechny zkoumané hodnoty hyperparametrů a na závěr vybereme jako nejlepší volbu ty hodnoty, které vedly k nejmenší cross-validační chybě.

Cross-validace: jak to funguje (obrázek)



Různé poznámky

- Křížová validace může být neúnosně výpočetně náročná a nemůže tak vždy nahradit strategii s jedinou validační množinou.
- V extrémním případě, kdy se k rovná počtu trénovacích dat, mluvíme o **Leave-one-out cross-validation**: trénuje se na celé trénovací množině bez jediného bodu, na kterém se pak měří výsledná chyba.
- Pro zrychlení vyhledávání nejbližších sousedů se data indexují: nejpoužívanější metody jsou `kD-trees` a `ball-trees`.
- Indexování zpomaluje fázi učení a zrychluje predikování. V `scikit-learn` je volba indexování umožněna parametrem `algorithm`, který může mít hodnoty `auto, ball_tree, kd_tree, brute`.
- V případě klasifikace umožňuje kNN od mnoha jiných (parametrických) metod vytvoření velice nepravidelné rozhodovací hranice.
- Na cvičení si ukážeme, jak lze kNN použít pro doplňování chybějících hodnot.