Rozpoznávanie obrazcov - 7 cvičenie
kNN a validácia

Viktor Kocur
viktor.kocur@fmph.uniba.sk

DAI FMFI UK

6.4.2020

k nearest neighbors

Základný princíp

Na príznakovom priestore $\mathbb{R}^n$ opatreným metrikou ρ určíme triedu pre príznakový vektor $\vec{x} \in \mathbb{R}^n$ tak, že nájdeme k prvkov z trénovacej množiny, ktoré sú k $\vec{x}$ najbližšie a triedu priradíme podľa toho ktorá trieda má najviac zástupcov z pomedzi k susedov.

Trénovanie?

Táto metóda nevyžaduje trénovanie, vždy sa pozeráme na celú trénovaciu množinu a hľadáme tam susedov. Táto metóda je teda pomalá.

Metriky

Definícia

Majme množinu P. Potom metrika je funkcia $\rho : P \times P \mapsto \mathbb{R}_0^+$ taká že pre $\forall p, q, r \in P$ platí:

1. $\rho(p, q) \geq 0$
2. $\rho(p, q) = 0 \Leftrightarrow p = q$
3. $\rho(p, q) = \rho(q, p)$
4. $\rho(p, r) \leq \rho(p, q) + \rho(q, r)$

Množinu s metrikou nazývame metrický priestor.

Metriky

Metriky na $\mathbb{R}^n$

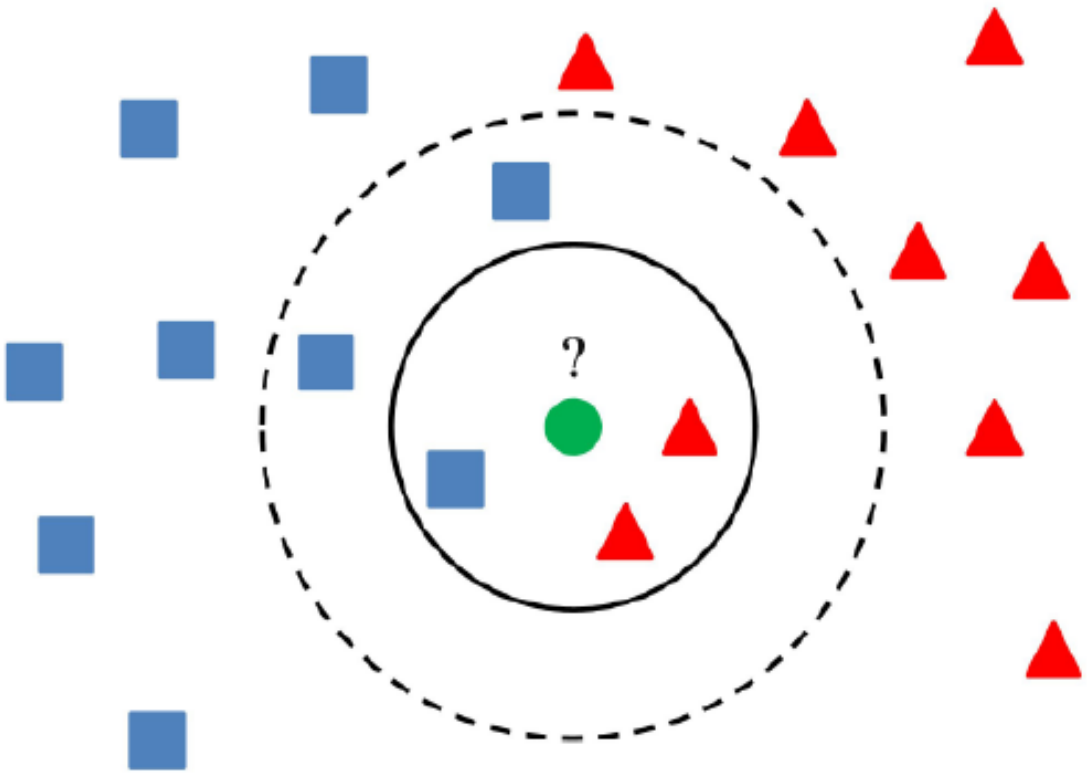
$$\rho_e(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$\rho_m(\vec{x}, \vec{y}) = \sum_{i=1}^{n}|x_i - y_i|$$

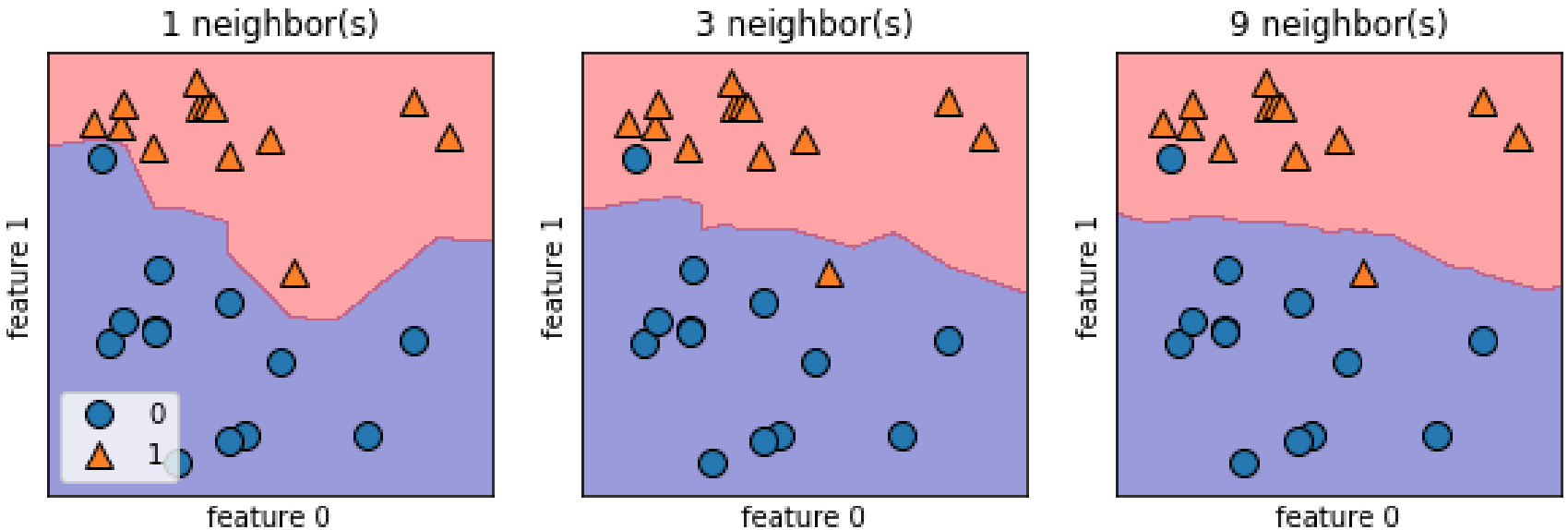
$$\rho_q(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n}(x_i - y_i)^q\right)^{\frac{1}{q}}$$

$$\rho_{max}(\vec{x}, \vec{y}) = max_i|x_i - y_i|$$

Voľba k



Voľba k



Úloha

Úloha

Vytvorte vlastnú funkciu mykNN(k, X, y, p) - ktorá vráti triedu podľa k-NN na trénovacích dátach X s triedami y pre vektor p.

Úloha

Funkciu otestujte na dátach z minulého cvičenia a fisheriris.

Úloha

fitcknn

Mdl = fitcknn(X,y) - vytvorí knn klasifikátor podobne ako na minulom cvičení

predict

Mdl.predict(x) - vráti predpoveď modelu

Properties

Zaujímavé properties sú: 'Standardize', 'Distance', 'NumNeighbors', 'NSMethod'. Pozrite si ich v helpe.

Úloha

Upravte m-file na zobrazovanie hraníc SVM klasifikátora z minulého cvičenia a zobrazte si hranice pre kNN.

Rozdelenie dát

Trénovacia množina

Doteraz sme vždy operovali s trénovacou množinou. Teda všetky dáta sme použili na nastavenie parametrov modelu.

Testovacia množina

V prípade, že chceme overiť že náš model je spoľahlivý je nutné odložiť si časť dát na testovanie. Testovacie dáta použijeme až na úplnom konci keď máme model hotový. Používame ich čisto na vyhodnotenie a nie na určenie metódy, alebo parametrov a hyperparametrov modelu.

Rozdelenie dát

Validačná množina

Keďže testovaciu množinu nepoužívame na určenie modelu, tak potrebujeme ešte jednu množinu na tento účel. Validačnú množinu používame na určenie správneho prístupu a nastavenie hyperparametrov modelu.

Rozdelenie dát

Podiely na rozdelovanie dát záležia od ich charakteru, množstva a modelu. Pri neurónových sieťach potrebujeme veľa trénovacích dát, preto je vhodné využiť split 80/10/10. Pri metódach aké sme si zatiaľ ukázali stačí aj 60/20/20. V niektorých prípadoch však je nutné isť ešte ďalej. Existujú datasety kde je split napr. 40/20/40.

Validácia - postup

Hyperparametre

Na validačnej množine určujeme hyperparametre. To sú parametre/nastavenia, ktoré menia spôsob akým sa model trénuje a ako funguje predikcia. Pre SVM je to napr. výber kernelovej funkcie a jej škály. Pre kNN je to napríklad hodnota k a výber metriky.

Validácia

Pre rôzne hyperparametre natrénujeme (v prípade kNN len vytvoríme) na trénovacej množine naše modely. Tieto potom otestujeme na validačnej množine. Použijeme na to nejakú mieru spoľahlivosti. Ideálne presnosť klasifikácie. Na základe výsledkov vyberieme hyperparametre.

Validácia - úloha

Úloha

Rozdelte si dáta z predchádzajúceho cvičenia na train/val/test s pomerom 60/20/20. A určite najlepší parameter k pre kNN klasifikátor a metriku na validačnej množine.

Pozor na dostatočnú reprezentáciu

Často sú dáta zoradené podľa triedy, alebo v nejajek inej pravidelnej forme. Je preto nutné overiť si, či je rozdelenie na train/val/test zmysluplné. Ideálne chceme rovnaký podiel tried pre každú množinu.

Vzájomná validácia

Vzájomná validácia

Ak máme málo dát tak nedelíme dáta na trénovacie a validačné. Dáta rozdelíme na n približne rovnakých podmnožín. Model vždy natrénujeme na dátach zo všetkých okrem jednej podmnožiny a otestujeme na jednej podmnožine. Toto opakujeme n krát a výsledok spriemerujeme.

Matlab

```
Mdl = fitcknn(X, y, 'NumNeighbors', k);
CVMdl = crossval(Mdl)
loss = kfoldLoss(CVMdl)
```

Vzájomná validácia

Automatické určenie hyperparametrov

Matlab pri väčšine fitc... funkcií dokáže nájsť optimálne hyperparametre sám. Ak to budete používať je dobre pozrieť sa do helpu.

Matlab

```
Mdl = fitcknn(X,Y,'OptimizeHyperparameters','auto')
```