Clasificatorul *k*-Nearest Neighbors

1 Generalităţi

În problemele de învățare automată, algoritmul *K*-nearest neighbors este o metodă utilizată atât pentru clasificare cât și pentru regresie. În ambele cazuri intrarea pentru acest algoritm constă în alegerea celor mai apropiate *k* exemple de antrenament reprezentate în spațiul trăsăturilor. În cazul regresiei ieșirea reprezintă valoarea proprie pentru obiectul curent și este calculată ca și medie a valorilor celor mai apropiați *k* vecini. În cazul clasificării ieșirea reprezintă clasa de care aparține exemplul din setul de testate. Un exemplu de testare este clasificat pe baza votului majoritar a vecinilor lui, astfel acestuia i se va atribuii clasa cea mai frecventă dintre clasele celor *k* cei mai apropiați vecini. Este un algoritm de învățare supervizat care oferă o perspectivă asupra naturii datelor, acesta fiind sensibil la structura locală a datelor.

Considerăm că avem un set de *m* vectori de intrare (set de date de intrare):

$$\vec{X} = \{\vec{X}_1, \vec{X}_2, ..., \vec{X}_m\},$$

şi o mulțime de *c* clase în care sunt clasificaţi vectorii de intrare $C = \{C_1, C_2, ..., C_c\}$

Fiecare vector de intrare este reprezentat prin *n* atribute şi o clasă ţintă C:

$$\vec{X} = (x_1, x_2, ..., x_n : C_j)$$

Setul de date se împarte în 2 subseturi, unul de antrenare și unul de testare. Setul de antrenare, mai mare reprezentând 70% din datele inițiale se folosește pentru a ajuta algoritmul să stabilească regulile de clasificare. Setul de testare este folosit pentru a stabili cât de bine a învățat algoritmul respectiv.

2 Algoritmul *K*-Nearest Neighbors

Acest algoritm este un algoritm de învățare supervizată bazat pe asocieri care nu necesită o etapă de antrenare propriu-zisă. Se bazează pe învățarea prin analogie și stabilește clasa corespunzătoare unui exemplu de testare pe baza similarității acestuia cu *k* exemple, cele mai similare, din setul de date de antrenament. Cele *k* exemple luate în considerare vor stabili clasa exemplului de test pe baza votului majoritar. Fiecare exemplu de antrenament este un vector în spațiul de reprezentare al datelor și are asignat o singură etichetă (clasa, target,..). Etapa de antrenare pentru algoritmul KNN constă doar în memorarea vectorilor de trăsături și a etichetelor corespunzătoare claselor pentru exemplele de antrenament. În faza de clasificare propriu-zisă (în etapa de testare), la un element din setul de testare îi atribuim clasa corespunzătoare ca fiind cea

mai frecventă clasă dintre clasele celor *k* exemple de antrenament, cele mai apropiate de exemplul de testare. Parametrul *k* este o constată specificată de utilizator și de obicei are o valoare mică. Cea mai bună alegere a lui *k* depinde de date; în general, o valoare mare pentru *k* va reduce influența zgomotului asupra clasificării, dar va face ca zonele de separare dintre clase să fie mai puțin distincte.

Clasificatorul *k*- Nearest Neighbors poate fi văzut ca un algoritm care atribuie la cei mai apropiați *k* vecini o pondere egală cu 1/k și la restul o pondere egală cu 0.

3 Pașii algoritmului KNN:

1. Se stabilește valoarea lui *k* în raport cu numărul de exemple de antrenament pe care le avem la dispoziție.
2. Pentru fiecare exemplu din setul de testare se stabilește clasa acestuia astfel:
3. Se calculează similaritatea dintre exemplul de testare și toate exemplele avute în setul de antrenare. Pentru calculul similarității se pot folosi oricare dintre metricile de similaritate descrise mai jos.
4. Se iau primele *k* exemple dintre cele de antrenare care sunt cele mai similare cu exemplul curent de testare și pe baza lor se stabilește clasa exemplului de testare folosind votul majoritar.
5. Se verifică dacă clasificarea este sau nu corectă pe baza informațiilor deținute în fișierul de testare.
6. Atâta timp cât mai sunt exemple de testare se reia de la pasul 3.
7. Se evaluează calitatea clasificării pentru valoarea lui *k* curenta, folosind metricile externe de evaluare a algoritmilor de învățare cum ar fi acuratețea de clasificare, precizia, recall, true negative rate etc…).

4 Metode de calcul a similarității

Există mai multe metode de calcul a similarității între doi vectori. Fiecare metodă se alege în funcție de domeniu aplicabilității metodei respective. Printre cele mai cunoscute și folosite metode sunt calculul distanței Euclidiene sau a cosinusului unghiului dintre doi vectori.

Prezentăm în acest laborator doar trei metode:

1. Calculul similarității folosind distanta Euclidiană:

$d(\vec{x},\vec{x}') = \sqrt{\sum_{i=0}^{n}(x_i - x'_i)^2}$, unde *n* reprezintă nr. de trăsături caracteristice iar *x* și *x'* reprezintă cei doi vectori pentru care se calculează distanța.

2. Calculul similarității folosind distanta Manhattan:

$$d(\vec{x},\vec{x}') = \sum_{i=0}^{n}(|x_i - x'_i|)$$

3. Calculul similarității folosind cosinusul unghiului între 2 vectori (atenție pentru ca această metrică să întoarcă rezultate corecte vectori trebuie normalizați):

$$d(\vec{x},\vec{x}') = \frac{<\vec{x},\vec{x}'>}{\|\vec{x}\| \cdot \|\vec{x}'\|}, \quad unde <\vec{x},\vec{x}'> = \sum_{i=0}^{n} x_i \cdot x_i^{'} \quad si \quad \|\vec{x}\| = \sqrt{\sum_{i=0}^{n} x_i \cdot x_i}$$

5 Metode de normalizare a datelor

1. ***Normalizarea binară*** – vectorul va conține valoare „0" dacă cuvântul respectiv nu apare în document, și „1" dacă cuvântul respectiv apare în document;
2. ***Normalizarea nominală*** – se normalizează valorile vectorului în intervalul [0,1] conform cu formula:

$$TF(d,t) = \frac{n(d,t)}{\max_{\tau} n(d,t)}$$

n(d,t) este frecvența de apariției a termenului *t* în documentul *d* și $\max_{\tau} n(d,\tau)$ valoarea termenului din documentul *d* cu număr maxim de apariții;

3. ***Normalizarea suma 1*** – se normalizează valorile vectorului în intervalul [0,1] conform cu formula:

$$TF(d,t) = \frac{n(d,t)}{\sum_{i=0}^{k} n(d,t_i)}$$

n(d,t) este frecvența de apariție a termenului *t* în documentul *d* și *suma(n(d,t$_i$))* reprezintă suma tuturor termenilor din vectorul de reprezentare a documentului *d;*

4. ***Normalizarea Cornell SMART*** – valoarea ponderilor se calculează conform formulei:

$$TF(d,t) = \begin{cases} 0 & daca\, n(d,t) = 0 \\ 1 + \log_{10}(1 + \log_{10}(n(d,t))) & altfel \end{cases}$$

n(d,t) este frecvența de apariție a termenului *t* în documentul *d*. Domeniul de reprezentare a valorii ponderilor este în intervalul [0,2]. Dacă cuvântul nu apare în document valoarea este 0. Valoarea maximă este 2 pentru un număr de apariții mai mic decât 10^9, având în vedere faptul că logaritmul este în baza 10.

5. ***Normalizarea IDF*** (Invers Document Frequency) – în vectorul de intrare sunt ponderate valorile în funcție de frecvența de apariției a termenului *t* în toată colecția de documente utilizând formula:

$$IDF(t) = \log\left(\frac{N}{N_t}\right)$$

unde *t* este termenul, *N* numărul total de documente din colecție iar N_t numărul de documente din colecție care conțin termenul *t*.

6. ***Reprezentarea TF_IDF*** – valorile ponderilor din vectorul de intrare se calculează folosind formula

$$TF\_IDF = TF \cdot IDF$$

unde *TF* este calculată folosind formula de la normalizarea Cornell Smart iar *IDF* se calculează folosind formula de la normalizarea IDF.

6 Tema laborator 4

Problemă: Pornind de la fișierul rezultat la tema Selecția Trăsăturilor caracteristice să se implementeze algoritmul de învățare KNN prezentat în acest laborator folosind toate cele 3 metode de calcul a similarității și toate metodele de normalizare a datelor. Să se evalueze rezultatele obținute de acest algoritm de învățare pentru diferite valori ale parametrului *k* de intrare și diferite valori pentru normalizarea datelor și calculul similarității. Algoritmul de învățare se va evalua din punct de vedere al acurateței de clasificare, al preciziei și al recall-lului.