Informatii ML

Kernel liniar

Kernel-ul este o funcție care transformă datele de intrare într-un alt spațiu dimensional pentru a face clasificarea mai ușoară în spațiul transformării. SVM este un algoritm de clasificare care încearcă să găsească hiperplanul care separă datele de intrare ale claselor diferite în acest spațiu transformării.
În cazul în care este specificat kernel-ul liniar, SVM va căuta un hiperplan linear de separare între cele două clase, adică o linie (în cazul bidimensional) sau un plan (în cazul tridimensional) care poate separa datele de intrare ale claselor diferite. Astfel, pentru datele de intrare care pot fi separabile prin hiperplane liniare, SVM cu kernel liniar poate fi o alegere bună pentru clasificare.

ReLU

Funcția de activare ReLU (Rectified Linear Unit) este una dintre cele mai utilizate funcții de activare în rețelele neuronale. Ea este definită astfel:

f(x) = max(0, x)

În această funcție, x este valoarea de intrare, iar f(x) este valoarea de ieșire. Funcția ReLU produce ieșiri de 0 pentru toate valorile negative de intrare și produce valoarea intrării pentru toate valorile pozitive de intrare.
Funcția ReLU este populară în rețelele neuronale datorită câtorva avantaje importante:

1. Eficiența computațională - funcția ReLU este foarte rapidă de calculat datorită simplității sale.
2. Sparsitate - funcția ReLU produce valori de ieșire zero pentru toate intrările negative, ceea ce poate ajuta la obținerea unor modele de rețea neurală mai puțin dense și mai eficiente.
3. Non-liniaritate - deși funcția ReLU este o funcție simplă, ea poate introduce non-liniaritate în modelele de rețea neurală și poate ajuta la învățarea de caracteristici mai complexe.

În general, funcția ReLU este o alegere populară pentru funcția de activare în rețelele neuronale datorită performanțelor sale bune în multe aplicații diferite.

Sigmoid

Funcția sigmoid este o funcție de activare comună folosită în rețelele neuronale. Aceasta este o funcție sigmoidală (curba S), care este definită astfel:

f(x) = 1 / (1 + exp(-x)) **exp(-x)=1/e^x**

În această funcție, x este valoarea de intrare, iar f(x) este valoarea de ieșire. Funcția sigmoid produce valori de ieșire între 0 și 1, ceea ce o face utilă în probleme de clasificare binară.

Prin utilizarea funcției sigmoid ca funcție de activare, o rețea neurală poate învăța să atribuie un scor de probabilitate pentru fiecare clasă posibilă. Dacă scorul de probabilitate este mai mare decât un prag prestabilit, rețeaua va prezice clasa pozitivă, în caz contrar, va prezice clasa negativă.
Funcția sigmoid a fost larg utilizată în trecut în rețelele neuronale, dar în prezent este mai puțin întâlnită datorită unor probleme cu gradientul său. Aceasta poate provoca probleme de gradient vanish (gradient care se apropie de 0) atunci când se utilizează în rețele adânci, ceea ce poate încetini sau chiar împiedica procesul de învățare. În schimb, funcții de activare non-saturante precum ReLU sau Leaky ReLU sunt preferate în prezent pentru rețelele neuronale adânci.

Cu cât x este mai mare, cu atât exp(-x) se apropie de zero, iar valoarea funcției sigmoid se apropie de 1. Cu cât x este mai mic, cu atât exp(-x) se apropie de 1, iar valoarea funcției sigmoid se apropie de 0. Astfel, funcția sigmoid poate fi utilizată pentru a atribui un scor de probabilitate între 0 și 1 pentru fiecare clasă posibilă în problemele de clasificare binară.

Interpretarea datelor de la CNN

Accuracy: 0.8717647058823529
Recall: 0.4669260700389105
F1-score: 0.5240174672489083
Aceste date reprezintă metrici de evaluare a performanței unui model de rețea neuronală convoluțională (CNN) pe setul de date de testare.

- Accuracy (acuratețe) se referă la proporția de exemple clasificate corect din totalul de exemple de testare. Acesta este unul dintre cei mai comuni indicatori de performanță a modelului, dar poate fi înșelător în cazul în care există dezechilibre semnificative între clasele de exemple.
- Recall (rechemare) sau Sensitivity (sensibilitate) se referă la proporția de exemple pozitive adevărate (exemplu de clasă pozitivă clasificat corect) din totalul de exemple pozitive (exemplu de clasă pozitivă reală). Aceasta poate fi utilă în cazurile în care este important să se detecteze toate exemplele pozitive.
- F1-score este o măsură a performanței care combină precision și recall. Este media armonică a precision-ului și recall-ului și oferă o valoare de evaluare a performanței care ține cont de ambele aspecte ale clasificării. Valoarea F1-score este cuprinsă între 0 și 1, cu 1 reprezentând performanța perfectă și 0 reprezentând performanța cea mai slabă.

În interpretarea acestor valori, se poate spune că modelul a atins o acuratețe de 0,871 (sau 87,1%), ceea ce înseamnă că a clasificat corect aproximativ 87% din exemplele din setul de date de testare. Recall-ul de 0,467 (sau 46,7%) arată că modelul a detectat aproximativ jumătate dintre exemplele pozitive, în timp ce F1-score-ul de 0,524 (sau 52,4%) indică o performanță generală bună, echilibrând precision-ul și recall-ul.

-> media armonica? Ea se calculează astfel:

Media armonică = n / (1/x1 + 1/x2 + ... + 1/xn)

unde:

- n este numărul de elemente în setul de date
- x1, x2, ..., xn sunt valorile setului de date

Putem vedea că formula este invers proporțională cu suma inverselor valorilor din setul de date. Astfel, valorile mai mici din set au o pondere mai mare în calcularea mediei armonice decât cele mai mari, ceea ce poate fi util în evaluarea performanței în cazurile în care contează mai mult obținerea de rezultate bune pentru valori mici decât pentru cele mari.

Model.compile()

Aceasta primește trei argumente:

- **loss:** = Funcția de pierdere pe care o va folosi modelul în timpul antrenării.
  este utilizată pentru a evalua cât de bine performează modelul în timpul antrenării
  diferența dintre predicția făcută de model și adevărata valoare țintă.

- **optimizer:** Algoritmul de optimizare folosit pentru a actualiza parametrii modelului în timpul antrenării. Scopul algoritmului de optimizare este să găsească valorile parametrilor care minimizează funcția de pierdere.
- **metrics:** O listă de metrici pe care modelul le va calcula și le va afișa în timpul antrenării și evaluării modelului. (f1, loss, …) - din pacate am luat decat acuratetea

loss= Binary crossentropy este o funcție de pierdere care se bazează pe logaritmul șanselor (logarithm of odds) și care măsoară diferența dintre distribuția de probabilitate a clasei prezise de model și distribuția reală a claselor.
penalizează modelul pentru predicțiile incorecte, ducând astfel la o mai bună convergență a modelului.
este folosită pentru că este optimizată pentru probleme de clasificare binară, adică atunci când avem două clase posibile.

optimizer=Adam : este un algoritm de optimizare ce se bazează pe optimizarea adaptivă a ratei de învățare (adaptive learning rate optimization). Acesta ajustează automat rata de învățare în timpul antrenării în funcție de gradientul mediu al ponderilor rețelei, ceea ce permite o mai bună convergență a modelului și o reducere a timpului necesar pentru antrenare. În comparație cu alți algoritmi de optimizare, cum ar fi gradientul descendent stocastic sau AdaGrad, Adam este considerat a fi mai eficient întrucât poate îmbunătăți rapid convergența la minimul global. În plus, acesta a demonstrat

o performanță mai bună decât alte algoritmi de optimizare în multe aplicații de rețele neuronale.

Recall?

Recall/ "sensibilitate" sau "acoperire", este o măsură a performanței unui model de clasificare, care exprimă proporția de cazuri pozitive din setul de date care au fost corect identificate de model. **Mai specific, recall-ul reprezintă numărul de cazuri pozitive corect clasificate împărțit la numărul total de cazuri pozitive din setul de date.**
În formă matematică, recall-ul poate fi calculat folosind următoarea formulă:

recall = true positives / (true positives + false negatives)

unde:

- true positives reprezintă numărul de cazuri pozitive corect clasificate de model
- false negatives reprezintă numărul de cazuri pozitive clasificate greșit ca negative de model.

Pentru a interpreta recall-ul, trebuie să ținem cont de specificitate (specificity) și de alte măsuri de performanță ale modelului. Un recall ridicat indică faptul că modelul are o capacitate bună de a identifica cazurile pozitive, dar acest lucru poate fi însoțit de o capacitate mai slabă de a identifica cazurile negative. De aceea, este important să analizăm toate măsurile de performanță ale modelului pentru a evalua eficacitatea generală a acestuia.

(despre) Matricea de confuzie

= metodă de evaluare a performanței unui model de clasificare,
reprezintă numărul de predicții corecte și greșite realizate de model

Această matrice este o reprezentare tabulară a numărului de cazuri din fiecare clasă adevărată și din fiecare clasă prezisă de model.
Matricea de confuzie este construită comparând etichetele reale (y_test) cu predicțiile modelului (y_pred) și numărând câte cazuri se încadrează în fiecare din cele patru categorii posibile:

- True Positive (TP): numărul de cazuri pozitive corect identificate de model
- False Positive (FP): numărul de cazuri negative greșit identificate ca pozitive de model
- False Negative (FN): numărul de cazuri pozitive greșit identificate ca negative de model
- True Negative (TN): numărul de cazuri negative corect identificate de model

Matricea de confuzie poate fi reprezentată sub forma unei matrice pătratice de dimensiunea numărului de clase prezise de model, în care **elementul (i,j) reprezintă numărul de cazuri care aparțin clasei i și au fost clasificate ca aparținând clasei j de către model**. !!!

=> Prezentat trasaturi / modele utilizate: 0.25 puncte per metoda (maxim 0.5 puncte daca s-au folosit mai mult de doua metode). Se urmareste corectitudinea implementarii si completitudinea documentatiei.
=> Tabele / Figuri pentru hyperparameter tuning: 0.25 puncte per metoda (maxim 0.5 puncte daca s-au folosit mai mult de doua metode).
=> Acuratete; precision si recall pe fiecare clasa; matrici de confuzie: 0.25 puncte per metoda (maxim 0.5 puncte daca s-au folosit mai mult de doua metode).
=> Doua intrebari despre metodele utilizate: 0.25 puncte per raspuns. Daca se prezinta o singura metoda, a doua intrebare poate fi despre orice metoda prezentata la laborator.

Acest cod reprezintă construirea unei rețele neuronale convoluționale (CNN) în biblioteca Keras, folosind modelul secvențial.

Rețeaua neuronală este formată din următoarele straturi:

1. Straturi convoluționale - acestea aplică filtre pe imaginea de intrare pentru a detecta diverse caracteristici, în acest caz avem trei straturi convoluționale cu 16, 32 și 64 de filtre fiecare, fiecare cu dimensiunea filtrului de 3x3.
2. Straturi de pooling - acestea reduc dimensiunea imaginii și numărul de parametri, făcând rețeaua mai eficientă. Folosim MaxPooling2D cu o dimensiune de 2x2.
3. Stratul flatten - acest strat transformă ieșirea de la straturile convoluționale și pooling într-un vector unidimensional.
4. Stratul dens - acesta este un strat complet conectat, cu 512 neuroni și funcția de activare ReLU.
5. Stratul de ieșire - este un strat dens, cu un singur neuron și funcția de activare sigmoid, folosit pentru clasificare binară.

Această rețea este proiectată să preia imagini de dimensiune 224x224x3 și să le clasifice în două clase diferite, pe baza valorii de ieșire a stratului de ieșire.