

Sisteme de recunoașterea formelor – Lab 9

Clasificatorul Bayes Naiv

1. Obiective

În această sesiune vom studia clasificatorul Bayes Naiv. Vom aplica acest clasificator pe o problemă de recunoașterea cifrelor scrise de mână.

2. Fundamente teoretice

Clasificatorul Bayes Naiv aplică regula Bayes și presupune independența trăsăturilor pentru a calcula probabilitățile posterioare. Clasa cu probabilitate posterioară maximă va fi aleasă în momentul clasificării.

Datorită presupunerii de independență clasificatorul poate lucra cu un număr arbitrar de trăsături. Pornim de la vectorul aleator cu d componente $\mathbf{x} = \{x_1, x_2, \dots, x_d\}$. Dorim să estimăm probabilitatea posterioară pentru clasele $C = \{c_1, c_2, \dots, c_j\}$. Într-un limbaj mai familiar, valorile din \mathbf{x} sunt trăsăturile care determina clasa instanței. Aplicând regula lui Bayes putem scrie:

$$p(c|x_1, x_2, \dots, x_d) \propto P(c)p(x_1, x_2, \dots, x_d|c)$$

unde $p(c|x_1, x_2, \dots, x_d)$ este probabilitatea posterioară, mai precis, probabilitatea clasei c dat fiind valorile trăsăturilor; $p(x_1, x_2, \dots, x_d|c)$ este funcția de verosimilitate (likelihood function); iar $P(C)$ este probabilitatea a priori. Bayes Naiv presupune că trăsăturile condiționate de clasă sunt independente, deci putem să descompunem verosimilitatea într-un produs în următorul mod:

$$p(\mathbf{x}|c) = \prod_{k=1}^d p(x_k|c)$$

și să rescriem probabilitatea posterioară ca:

$$p(c|\mathbf{x}) \propto P(C) \prod_{k=1}^d p(x_k|c)$$

Utilizând regula Bayes de mai sus putem să implementăm un clasificator. Acesta va returna clasa cu probabilitate posterioară maximă:

$$c^* = \operatorname{argmax}_j p(c_j|\mathbf{x})$$

Deși presupunerea de independență nu este în general valabilă aceasta simplifică problema de clasificare în mod dramatic fiindcă permite calcularea probabilităților separat pentru fiecare clasă. În esență, se reduce problema estimării unei densități de probabilitate multidimensionale la mai multe probleme unidimensionale. Densitatea de probabilitate

poate avea forme diferite precum: distribuție normală, log-normală, gamma, Poisson sau variante discrete ale acestora.

3. Detalii de implementare

3.1. Datasetul MNIST

Vom utiliza un benchmark standard pentru recunoașterea cifrelor. Datasetul MNIST a fost asamblat de Yann LeCun din mai multe seturi. Partea de antrenare conține 60000 imagini cu cifre scrise de mână de aproximativ 250 de persoane. Setul de test conține 10000 instanțe. Distribuția cifrelor este aproape uniformă. Pentru mai multe detalii accesați linkul [2]. Vom utiliza distribuții binomiale pentru a modela verosimilitatea. O distribuție binomială specifică probabilitățile pentru două evenimente discrete și complementare.

3.2. Algoritmul de antrenare

Fie X matricea cu trăsături pentru setul de antrenare. În acest caz X va conține pe fiecare linie valorile binarizate ale unei imagini din setul de antrenare. Se stochează 0 sau 255 în funcție dacă pixelul pe poziția respectivă este mai mare sau egal decât un prag fixat la 128. X are dimensiunea de $n \times d$, unde n este numărul de instanțe de antrenare, iar $d = 28 \times 28$ este mărimea unei imagini. Vectorul de etichete y are dimensiunea $n \times 1$.

Probabilitatea *a priori* pentru clasa i se calculează ca și fracția instanțelor aparținând clasei i dintre toate instanțele de antrenare:

$$P(c = i) = n_i/n$$

Funcția de verosimilitate pentru ca trăsătura x_j să fie egală cu 255 condiționată de clasa c este egală cu numărul de instanțe din clasa c care au trăsătura x_j egală cu 255 împărțit la numărul total de instanțe din clasa c :

$$p(x_j = 255|c = i) = \frac{\text{count}(X_{kj} = 255 \wedge y_k = i)}{n_i}$$

Funcția de verosimilitatea pentru ca trăsătura x_j să fie egală cu 0, se poate deduce utilizând valoarea anterioară, astfel:

$$p(x_j = 0|c = i) = 1 - p(x_j = 255|c = i)$$

Pentru a evita înmulțirea cu 0 la produsul final trebuie să ne asigurăm că verosimilitățile nu au valoarea 0. Putem rezolva această problemă prin fixarea tuturor valorilor sub un prag de 10^{-5} la 10^{-5} . O alternativă practică este utilizarea netezirii Laplace, unde $|C|$ este numărul de clase:

$$p(x_j = 255|c = i) = \frac{\text{count}(X_{kj} = 255 \wedge y_k = i) + 1}{n_i + |C|}$$

3.3. Algoritmul de clasificare

Odată ce am calculat valorile pentru probabilitățile *a priori* și funcțiile de verosimilitate, putem efectua clasificarea. Deoarece valorile probabilităților sunt cuprinse între 0 și 1 produsul lor va fi un număr foarte mic. Pentru a evita probleme de precizie numerică se va lucra cu logaritmul probabilităților posterioare. Fie T vectorul de trăsături pentru imaginea de test, care sunt valorile binarizate în același mod ca și la imaginile antrenare. Logaritmul probabilității posterioare pentru fiecare clasă se poate calcula ca și:

$$\log(p(C = i|T)) \propto \log(P(C = i)) + \sum_{j=1}^d \log(p(x_j = T_j|C = i))$$

Deoarece ordinea probabilităților posterioare nu se schimbă din cauza logaritmului clasificatorului va selecta clasa cu probabilitatea posterioară maximă, la fel ca și în teoria prezentată.

Pentru a încărca primele 100 de imagini din clasa c :

```
char fname[256];
int c = 1;
int index = 0;
while(index<100){
    sprintf(fname, "train/%d/%06d.png", c, index);
    Mat img = imread(fname, 0);
    if (img.cols==0) break;
    //process img
    index++;
}
```

Probabilitatea *a priori* este un vector de $C \times 1$:

```
const int C = 3; //number of classes
Mat priors(C,1,CV_64FC1);
```

Verosimilitatea este o matrice de $C \times d$ (memorăm doar verosimilitatea pentru valoarea 255):

```
const int d = 28*28;
Mat likelihood(C,d,CV_64FC1);
```

Sugestie de antet pentru funcția de clasificare:

```
int classifyBayes(Mat img, Mat priors, Mat likelihood);
```

4. Lucrare practică

1. Încărcați imaginile din setul de antrenare și aplicați operația de binarizare. Salvați valorile în matricea de trăsături. Salvați clasa instanțelor în vectorul de etichete y . Pentru varianta inițială utilizați doar primele 100 de imagini din clasele 0 și 1.
2. Implementați algoritmul de antrenare.
 - 2.1. Calculați și salvați probabilitatea *a priori* pentru fiecare clasă.
 - 2.2. Calculați și salvați verosimilitatea pentru fiecare clasă și fiecare trăsătură. Aplicați netezirea Laplace pentru a evita valorile nule.
3. Aplicați clasificatorul Bayes Naiv pe o imagine necunoscută.
4. La clasificare afișați log-probabilitățile posterioare pentru fiecare clasă. Opțional, convertiți aceste valori în probabilități.
5. Evaluați clasificatorul pe setul de test. Calculați și afișați matricea de confuzie și eroarea de clasificare. Eroarea este egală cu numărul de instanțe clasificate greșit per numărul total de instanțe.
6. Antrenați și evaluați pe setul întreg utilizând toate clasele.

5. Referințe

- [1] *Electronic Statistics Textbook* – <http://www.statsoft.com/textbook/stnaiveb.html>
- [2] LeCun, Yann, Corinna Cortes, and Christopher JC Burges. "The MNIST database."
URL <http://yann.lecun.com/exdb/mnist> (1998).