Sisteme de recunoaşterea formelor – Lab 9

Clasificatorul Bayes Naiv

1. Obiective

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

2. Fudamente teoretice

Clasificatorul Bayes Naiv aplică regula Bayes şi presupune independenţa trăsăturilor pentru a calcula probabilităţile posterioare. Clasa cu probabilitate posterioară maximă va fi aleasă în momentul clasificării.

Datorită presupunerii de independenţă clasificatorul poate lucra cu un număr arbitrar de trăsături. Pornim de la vectorul aleator cu d componente $\boldsymbol{x} = \{x_1, x_2, \dots, x_d\}$. Dorim să estimăm probabilitatea posterioară pentru clasele $C = \{c_1, c_2, \dots, c_J\}$. Într-un limbaj mai familiar, valorile din $\boldsymbol{x}$ sunt trăsăturile care determina clasa instanţei. 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 funcţia de likelihood sau verosimilitate; iar $P(C)$ este priorul sau probabilitatea a priori. Bayes Naiv presupune că trăsăturile condiţionate de clasă sunt independente, deci putem să descompunem likelihood-ul într-un produs în următorul mod:

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

şi să rescriem probabilitatea posterioară ca:

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

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

$$c^* = argmax_j p(c_j|\boldsymbol{x})$$

Deşi presupunera de independenţa nu este în general valabilă aceasta simplifică problema de clasificare în mod dramatic fiindcă permite calcularea probabilităţilor separat pentru fiecare clasă. În esenţă, se reduce problema estimării unei densităţi de probabilitate multidimensionale la mai multe probleme unidimensionale. Densitatea de probabilitate

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

3. Detalii de implementare

3.1. Datasetul MNIST

Vom utiliza un benchmark standard pentru recunoaşterea cifrelor. Datasetul MNIST a fost asamblat de Yann LeCun din mai multe seturi. Partea de antrenare conţine 60000 imagini cu cifre scrise de mână de aproximativ 250 de persoane. Setul de test conţine 10000 instanţe. Distribuţia cifrelor este aproape uniformă. Pentru mai multe detalii accesaţi linkul [2]. Vom utiliza distribuţii binomiale pentru a modela likelihoodul. O distribuţie binomială specifică probabilităţile pentru două evenimente discrete şi complementare.

3.2. Algoritmul de antrenare

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

Priorul pentru clasa *i* se calculează ca şi fracţia instanţelor aparţinând clasei *i* dintre toate instanţele de antrenare:

$$P(c = i) = {}^{n_i}/_{n}$$

Funcţia de likelihood pentru ca trăsătura *j* să fie egală cu 255 condiţionată de clasa *c* este egală cu numărul de instanţe din clasa *c* care au trăsătura *j* egală cu 255 împărţit la numărul total de instanţe din clasa *c*:

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

Likelihood-ul pentru a avea o trăsătură egală 0 este se poate calcula utilizând valoarea anterioră:

$$p\left(x_j = 0 \middle| c = i\right) = 1 - p\left(x_j = 255 \middle| c = i\right)$$

Pentru a evita înmulţirea cu 0 la produsul final trebuie să asigurăm cu likelihood-urile 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:

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

3.3. Algoritmul de clasificare

Odată ce am calculat valorile pentru prior-uri şi likelihood-uri putem realiza clasificarea. Deoarece valorile probabilităţilor sunt cuprinse între 0 şi 1 produsul lor va fi un număr foarte mic. Pentru a evita probleme de precizie numerică se va lucra cu logaritmul posteriorului. Fie *T* vectorul de trăsături pentru imaginea de test, care sunt valorile binarizate în acelaş mod ca şi la imaginile antrenare. Logaritmul posteriorului pentru fiecare clasă se poate calcula ca şi:

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

Deoarece ordinea probabilităţilor posterioare nu se schimbă din cauza logaritmului clasificatorul va selecta clasa cu posteriorul maxim, la fel ca şi 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++;
}
```

Priorul este un vector de *C x 1*:

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

Likelihood-ul este o matrice de *C x d* (memorăm doar likelihoodul pentru valoarea 255):

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

Sugestie de antet pentru funcţia de clasificare:

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

4. Lucrare practică

1. Încărcaţi imaginile din setul de antrenare şi aplicaţi operaţia de binarizare. Salvaţi valorile în matricea de trăsături. Salvaţi clasa instanţelor în vectorul de etichete *y*. Pentru varianta initială utilizaţi doar primele 100 de imagini din clasele 0 şi 1.
2. Implementaţi algoritmul de antrenare.
2.1. Calculaţi şi salvaţi priorul pentru fiecare clasă.
2.2. Calculaţi şi salvaţi likelihood-ul pentru fiecare clasă şi fiecare trăsătură. Aplicaţi netezirea Laplace pentru a evita valorile nule.
3. Implementaţi clasificatorul Bayes Naiv pe o imagine necunoscută.
4. La clasificare afişaţi log-probabilităţile posterioare pentru fiecare clasă. Opţional, convertiţi aceste valori în probabilităţi.
5. Evaluaţi clasificatorul pe setul de test. Calculaţi si afişaţi matricea de confuzie şi eroarea de clasificare. Eroarea este egală cu numărul de instanţe clasificate greşit supra numărul total de instanţe.
6. Antrenaţi şi evaluaţi pe setul întreg utilizând toate clasele.

5. Referinţe

[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).