Sisteme de recunoaşterea formelor – Lab 1

Metoda celor mai mici pătrate

1. *Obiective*

Acest laborator introduce librăria OpenCV care va fi folosită pentru procesarea imaginilor. Se doreşte potirivirea unei linii la o mulţime de puncte 2D. Se va folosi metoda celor mai mici pătrate. Se prezintă atât o soluţie iterativă cât şi o formulă directă pentru mai multe modele.

2. *Fundamente teoretice*

Se dă o mulţime de puncte bidimensionale de forma (x_i, y_i) unde $i = \{1,2, \dots, n\}$. Sarcina voastră este să găsiţi ecuaţia liniei care se potriveşte cel mai bine la aceste puncte. Vom utiliza regresie liniară (metoda celor mai mici pătrate). Mulţimea de puncte este considerată ca mulţime de antranare şi se doreşte potrivirea unui model liniar pe date.

Model 1

La orice metodă de potrivire primul pas constă în stabilirea modelului. La început vom folosi un model liniar care conţine un termen pentru pantă şi un termen liber. Exprimăm componenta *y* în funcţie de *x* folosind funcţia:

$$f(x) = \theta_0 + \theta_1 x$$

De obicei acest model este folosit pentru rezolvarea problemei. Însă această reprezentare nu poate să modeleze linii verticale. Totuşi vom porni de la acest model simplu. Se poate forma un vector care va conţine toţi parametri $\boldsymbol{\theta} = [\theta_0, \theta_1]^{\mathrm{T}}$ (termenul liber şi coeficientul lui *x*).

Metoda curentă adopta o funcţie de cost care sumează erorile pătrate dintre estimatorul nostru şi valorile originale. Modelul ideal va fi obţinut când funcţia de cost atinge minimul global:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

De ce pătratic? Putem motiva această alegere prin presupunerea că eroarea în date urmează o distribuţie gaussiană. O observaţie importantă este că această funcţie penalizează erorile doar în direcţia *y* şi nu foloseşte distanţele punctelor la dreaptă. Pentru a minimiza funcţia de cost vom calcula derivatele parţiale:

$$\frac{\partial}{\partial \theta_0} J(\theta) = \sum_{i=1}^{n}(f(x_i) - y_i)$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = \sum_{i=1}^{n}(f(x_i) - y_i)\, x_i$$

Funcţia de cost atinge minimul global când derivatele parţiale sunt nule. O metodă generală pentru a găsi soluţia este *gradient descent*. Deorece gradientul arată direcţia în care funcţia creşte cel mai mult făcând un pas în direcţia opusă valoarea funcţiei descreşte. Dacă facem mai multe iteraţii şi controlăm mărimea paşilor vom atinge minimul global. Deoarece funcţia de cost este pătrătică există un singur minimim global care va fi găsit de această abordare.

La început se aleg valori aleatoare dar diferite de 0 pentru parametri. Apoi se calculează gradientul în punctul curent:

$$\nabla J(\boldsymbol{\theta}) = \left[\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0}, \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1}\right]^T$$

Şi apoi se aplică următoarea regulă până la convergenţă:

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta}),$$

unde α este rata de învăţare şi este aleasă astfel încât funcţia de cost să descrească la fiecare iteraţie. Când norma gradientului devine suficient de mică algoritmul se opreşte.

Această metodă este potrivită atunci când este dificil să găsim rădăcinile sistemului de ecuaţii. Pentru modelul curent se poate determina foarte uşor soluţia finală. Ecuaţiile pentru derivatele parţiale egale cu 0 se aduc la următoarea formă:

$$\begin{cases} \theta_0 n + \theta_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ \theta_0 \sum_{i=1}^{n} x_i + \theta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}$$

care este un sistem de ecuaţii liniare în două necunoscute şi poate fi rezolvat. Se obţin valorile:

$$\begin{cases} \theta_1 = \dfrac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \\ \theta_0 = \dfrac{1}{n}\left(\sum_{i=1}^{n} y_i - \theta_1 \sum_{i=1}^{n} x_i\right) \end{cases}$$

Ecuaţiile normale - O soluţie generală sub formă vectorială

În majoritatea cazurilor sistemul de ecuaţii se poate formula sub formă vectorială. Fie matricea A formată din liniile $[1\ x_i]$ şi fie vectorul coloana $\boldsymbol{b}$ care conţine toate valorile y_i. Folosind aceste notaţii se doreşte minimizarea normei:

$$\|A\boldsymbol{\theta} - \boldsymbol{b}\|^2 = (A\boldsymbol{\theta} - \boldsymbol{b})^T (A\boldsymbol{\theta} - \boldsymbol{b})$$

Această formulă se generalizează foarte uşor pentru mai multe dimensiuni. În acest caz soluţia poate fi obţinută prin formula:

$$\boldsymbol{\theta}_{opt} = (A^T A)^{-1} A^T \boldsymbol{b}$$

Pentru mai multe detalii şi demonstraţie consultaţi [1].

Model 2

Adoptăm un alt model pentru a rezolva problemele amintite în partea anterioră. Acest model este capabil să trăteze toate cazurile cu succes. Considerăm o parametrizare a unei linii de forma:

$$xcos(\beta) + ysin(\beta) = \rho$$

Aceasta descrie o linie cu normală unitară de formă $[cos(\beta), sin(\beta)]$ care se află la o distanţă ρ de la origine.

Scriem acum funcţia de cost care va fi suma distanţelor la pătrat de la fiecare punct la dreaptă:

$$J(\beta,\rho) = \frac{1}{2}\sum_{i=1}^{n}(x_i cos(\beta) + y_i sin(\beta) - \rho)^2$$

Observăm că aceasta reprezintă eroarea adevărată pe care trebuie să minimizăm şi că funcţia de cost pentru modelul 1 măsoară doar discrepanţa pe axa y.

Derivatele parţiale au următoarea formă:

$$\frac{\partial J}{\partial \beta} = \sum_{i=1}^{n}(x_i cos(\beta) + y_i sin(\beta) - \rho)\big(-x_i sin(\beta) + y_i cos(\beta)\big)$$

$$\frac{\partial J}{\partial \rho} = -\sum_{i=1}^{n}(x_i cos(\beta) + y_i sin(\beta) - \rho)$$

Şi în acest caz putem să obţinem o soluţie directă, însă este mult mai dificil să rezolvăm sistemul. Formulele pentru cei doi parametri sunt:

$$\beta = -\frac{1}{2}atan2\left(2\sum_{i=1}^{n}x_i y_i - \frac{2}{n}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i, \sum_{i=1}^{n}(y_i^2 - x_i^2) + \frac{1}{n}\left(\sum_{i=1}^{n}x_i\right)^2 - \frac{1}{n}\left(\sum_{i=1}^{n}y_i\right)^2\right)$$

$$\rho = \frac{1}{n}\left(\cos(\beta)\sum_{i=1}^{n}x_i + sin(\beta)\sum_{i=1}^{n}y_i\right)$$

Model 3

Există încă o posibilitate care oferă o rezolvare generală. Dacă folosim o parametrizare cu 3 parametri:

$$ax + by + c = 0$$

Această parametrizare tratează corect liniile verticale fiindcă acestea se modelează cu *b=0*. Funcţia de cost se defineşte ca:

$$J(a,b,c) = \frac{1}{2}\sum_{i=1}^{n}(ax_i + by_i + c)^2$$

care poate fi scris vectorial sub forma unei norme care trebuie minimizat:

$$J(a,b,c) = (A\boldsymbol{\theta})^T A\boldsymbol{\theta}$$

unde *A* este o matrice cu *nx3* elemente, fiecare linie conţine $(x_i, y_i, 1)$ şi $\boldsymbol{\theta} = [a,b,c]^T$ este vectorul de parametri (un vector coloană). Observăm ca funcţia de cost diferă faţă de cea definită la partea cu *ecuaţia normală*.

Utilizarea unui model cu 3 parametri are două consecinţe relevante: se poate modela orice linie dar pentru o linie avem mai multe parametrizări echivalente. Pentru a rezolva problema ambiguităţii impunem restricţia ca $\boldsymbol{\theta}$ să aibă normă unitară. Soluţia la această problemă utilizaează descompunerea cu valori singulare (Singular Value Decomposition SVD). Descompunem A în trei matrici:

$$A = USV$$

unde U şi V sunt ortogonale şi S conţine valori nenule doar pe diagonală (valori singulare). Soluţia se citeşte ca şi ultima coloana din matricea V care corespunde la valoarea singulară cea mai mică. Pentru mai multe detalii consultaţi [2].

$$\theta_{opt} = V(:, n)$$

3. Exexmple de rezultate

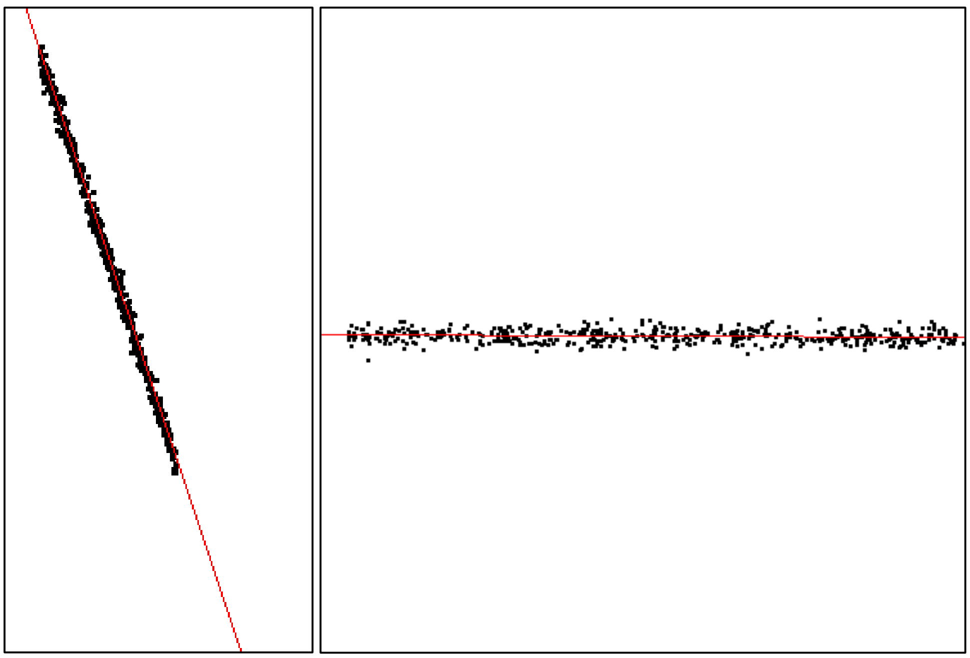

Figura 1 – Rezultate obţinute folosind modelul 2 pe datele din points1 şi points2

4. Consideraţii practice

Citire din fişier:

```
FILE* f = fopen("filename.txt","r");
float x,y;
fscanf(f,"%f%f", &x,&y);
fclose(f);
```

Crearea unei imagini color:

```
Mat img(height, width, CV_8UC3); //8bit unsigned 3 channel
```

Accesarea pixelului de pe rândul i şi coloana j:

```
Vec3b pixel = img.at<Vec3b>(i,j); //byte vector with 3
elements
```

Modificarea pixelului de la poziţia (i,j):

```
img.at<Vec3b>(i,j)[0] = 255; //blue
img.at<Vec3b>(i,j)[1] = 255; //green
img.at<Vec3b>(i,j)[2] = 255; //red
```

Desenarea unei linii care trece prin 2 puncte:

```
line(img, Point(x1, y1), Point(x2, y2), Scalar(B,G,R));
```

Afişarea imaginii:

```
imshow("title", img);
waitKey();
```

5. Activitate practică

1. Citiţi datele de intrare din fişierele ataşate. Prima linie conţine numărul de puncte. Liniile următoare conţin perechi (x,y).
2. Afişaţi punctele pe o imagine albă de dimensiune 500x500. Pentru vizibilitate mai bună se poate trasa un cerc, sau un pătrat în jurul fiecărui punct. Aveţi în vedere convenţia pentru sistemul de coordonate al imaginii. Unele puncte pot avea coordonate negative. Acestea ori nu se afişează ori se translatează graficul. Metoda în sine nu este afectată de faptul că punctele au coordonate negative.
3. Opţional, utilizaţi modelul 1 cu gradient descent. Vizualizaţi poziţia liniei la fiecare pas şi tipăriţi valorile funcţiei de cost. Trebuie să alegeţi o rată de învăţare care asigura descreşterea funcţiei de cost.
4. Folosiţi *modelul 1* şi formulele pentru a calcula parametri. Vizualizaţi linia şi comparaţi rezultatul cu soluţia iterativă de la pasul anterior.
5. Opţional, utilizaţi modelul 2 şi gradient descent pentru a găsi parametri. Vizualizaţi poziţia liniei la fiecare pas şi tipăriţi valorile funcţiei de cost. Trebuie să alegeţi o rată de învăţare care asigură descreşterea funcţiei de cost.
6. Folosiţi *modelul 2* şi formulele pentru a calcula parametri. Vizualizaţi linia şi comparaţi rezultatul cu soluţia iterativă de la pasul anterior.
7. Opţional, desenaţi erorile de la puncte ca şi segmente perpendiculare pe linia găsită.
8. Opţional, utilizaţi modelul 3 pentru a găsi ecuaţia dreptei.

6. Referinţe

[1] Stanford Machine Learning - course notes 1 – http://cs229.stanford.edu/notes/cs229-notes1.pdf
[2] Tomas Svoboda - Least-squares solution of Homogeneous Equations - http://cmp.felk.cvut.cz/cmp/courses/XE33PVR/WS20072008/Lectures/Supporting/constrained_lsq.pdf