先端データ解析論 (杉山将先生・本多淳也先生)
第 11 回レポート

ashiato45

2017 年 7 月 4 日

宿題 1

C++ と Eigen による実装 (表示には Python を用いた) は付録 1 にある。結果、図 1 と図 2 を得た。

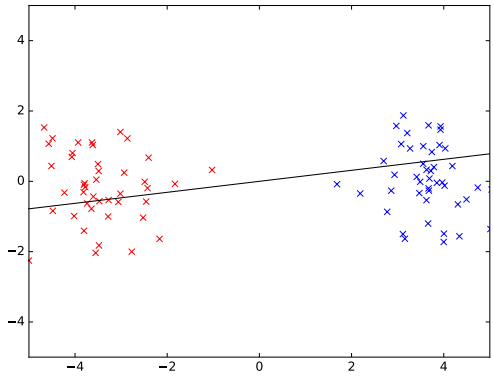

図 1

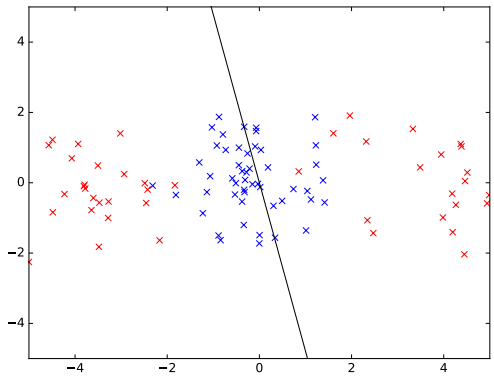

図 2

宿題 2

前半を示す。

$$S^{(w)} = \sum_{y=1}^{c} \sum_{i\,:\,y_i=y} (x_i - \mu_y)(x_i - \mu_y)^\top \tag{1}$$

$$= \sum_{y} \sum_{i} \left(x_i x_i^\top - x_i \sum_{j\,:\,y_j=y} \frac{1}{n_y} x_j^\top - \sum_{k\,:\,y_k=y} \frac{1}{n_y} x_k x_i^\top + \frac{1}{n_y^2} \sum_{j} \sum_{k} x_j x_k^\top \right) \tag{2}$$

$$= \sum_{y} \left(\sum_{i} x_i x_i^\top - \frac{2}{n_y} \sum_{i,j} x_i x_j^\top + \frac{1}{n_y^2} \sum_{i,j,k} x_j x_k^\top \right) \tag{3}$$

$$= \sum_{y} \left(\sum_{i} x_i x_i^\top - \frac{1}{n_y} \sum_{i,j} x_i x_j^\top \right). \tag{4}$$

一方、

$$\frac{1}{2} \sum_{i,j=1}^{n} Q_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^\top = \frac{1}{2} \sum_{y=1}^{c} \sum_{i\,:\,y_i=y} \sum_{z=1}^{c} \sum_{j\,:\,y_j=z} Q_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^\top \tag{5}$$

$$= \frac{1}{2} \sum_{y=1}^{c} \sum_{i,j} \frac{1}{n_y} (x_i - x_j)(x_i - x_j)^\top \tag{6}$$

$$= \frac{1}{2} \sum_{y=1}^{c} \sum_{i,j} \frac{1}{n_y} (x_i x_i^\top - x_i x_j^\top - x_j x_i^\top + x_j x_j^\top) \tag{7}$$

$$= \sum_{y=1}^{c} \left(\sum_{i} x_i x_i^\top - \frac{1}{n_y} \sum_{i,j} x_i x_j^\top \right). \tag{8}$$

よって、

$$S^{(w)} = \frac{1}{2} \sum_{i,j=1}^{n} Q_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^\top. \tag{9}$$

前半は示された。

後半を示す。

$$\frac{1}{2} \sum_{i,j=1}^{n} Q_{i,j}^{(b)} (x_i - x_j)(x_i - x_j)^\top = \frac{1}{2} \sum_{i,j=1}^{n} (1/n - Q_{i,j}^{(b)})(x_i - x_j)(x_i - x_j)^\top \tag{10}$$

$$= \frac{1}{2n} \sum_{i,j=1}^{n} (x_i - x_j)(x_i - x_j)^\top - S^{(w)} \tag{11}$$

$$= \sum_{i=1}^{n} x_i x_i^\top - S^{(w)} \tag{12}$$

$$= S^{(b)}. \tag{13}$$

示された。

付録 1

学習プログラム

```
#include <iostream>
```

```
#include <Eigen/Dense>
#include <Eigen/Eigenvalues>
#include <cmath>
#include <random>
#include <string>
#include <fstream>
// #define M_PI 3.1416

#define print(var)  \
  std::cout<<#var"= "<<std::endl<<(var)<<std::endl

using Eigen::MatrixXd;
using Eigen::MatrixXf;
using Eigen::GeneralizedEigenSolver;

void save_mat(std::string name, const MatrixXd* mat){
  std::ofstream file(name);
  file << *mat << std::endl << std::endl;
}

float gk(const MatrixXd& x, const MatrixXd& c){
  float hh = std::pow(2*1, 2);
  auto d = (x - c).norm();
  return std::exp(-d*d/hh);
}


int main()
{
  std::mt19937 engine;
  std::normal_distribution<> dist(0, 1);
  std::uniform_real_distribution<> unif(0, 1);

  /* make points */
  MatrixXd matX(2, 100);
  MatrixXd sumX(2, 1);
  MatrixXd vecMu(2, 2);
  // for(int i=0; i < 100; i++){
  //   matX(0, i) = dist(engine);
  //   if(i < 50){
  //     matX(0, i) -= 4;
  //   }else{
  //     matX(0, i) += 4;
  //   }
  //   matX(1, i) = dist(engine);
  // }
  for(int i=0; i < 100; i++){
    matX(0, i) = dist(engine);
    if(i < 25){
      matX(0, i) -= 4;
    }else if(25 <= i && i < 50){
      matX(0, i) += 4;
    }
    matX(1, i) = dist(engine);
  }

  /* centralize */
  for(int i=0; i < 100; i++){
    sumX += matX.col(i);
  }
  for(int i=0; i < 100; i++){
    matX.col(i) -= sumX/100;
  }

  vecMu(0, 0) = 0;
  vecMu(0, 1) = 0;
  vecMu(1, 0) = 0;
  vecMu(1, 1) = 0;
  std::cout << vecMu << std::endl;
  for(int i=0; i < 100; i++){
    if(i < 50){
      vecMu(0, 0) += matX(0, i);
      vecMu(1, 0) += matX(1, i);
    }else{
      vecMu(0, 1) += matX(0, i);
      vecMu(1, 1) += matX(1, i);
    }
    std::cout << i << "," << vecMu << std::endl;
  }
  // vecMu /= 50;

  MatrixXd matSw(2, 2);
  // std::cout << matSw << std::endl;
  for(int i=0; i < 100; i++){
    if(i < 50){
      matSw += (matX.col(i) - vecMu.col(0))*(matX.col(i) - vecMu.col(0)).transpose();
    }else{
      matSw += (matX.col(i) - vecMu.col(1))*(matX.col(i) - vecMu.col(1)).transpose();
    }
      // std::cout << i << "," << matSw << std::endl;
  }

  MatrixXd matSb(2, 2);
  matSb = 50*vecMu.col(0)*vecMu.col(0).transpose() + 50*vecMu.col(1)*vecMu.col(1).transpose();




  // MatrixXd matW(100, 100);
  // for(int i=0; i < 100; i++){
  //   for(int j=0; j < 100; j++){
  //     matW(i, j) = gk(matX.col(i), matX.col(j));
  //   }
  // }

  // MatrixXd matD(100, 100);
  // for(int i=0; i < 100; i++){
  //   matD(i, i) = matW.col(i).sum();
  // }

  // MatrixXd matL = matD - matW;

  GeneralizedEigenSolver<MatrixXd> ges;
  ges.compute(matSb, matSw, true);
  MatrixXd matTlpp(2, 2); //= ges.eigenvectors().alpha();
```

```
  for(int i=0; i < 2; i++){
    for(int j=0; j < 2; j++){
      matTlpp(i, j) = ges.eigenvectors()(i, j).real();
    }
  }
  MatrixXd bvec(2, 1);
  if(ges.eigenvalues()(0).real() > ges.eigenvalues()(1).real()){
    bvec = matTlpp.col(0);
  }else{
    bvec = matTlpp.col(1);
  }

  // MatrixXd matRed = matTlpp*matX;

  // // out
  // std::ofstream file("matTlpp");
  // file << matTlpp << std::endl;
  // std::ofstream filew("matW");
  // filew << matW << std::endl;
  // std::ofstream filed("matD");
  // filed << matD << std::endl;
  // std::ofstream filep("points");
  // filep << matX << std::endl;
  // std::ofstream fileeig("eig");
  // fileeig << ges.eigenvalues() << std::endl << ges.eigenvectors() << std::endl;
  // std::ofstream filered("matRed");
  // filered << matRed << std::endl;
  // std::ofstream filewvec("matX");
  // filewvec << matX << std::endl;
  // std::ofstream filebvec("bestVec");
  // filebvec << bvec << std::endl;
  save_mat(std::string("matSb"), &matSb);
  save_mat(std::string("matSw"), &matSw);
  save_mat(std::string("vecMu"), &vecMu);
  save_mat(std::string("matTlpp"), &matTlpp);
  save_mat(std::string("matX"), &matX);
  save_mat(std::string("bVec"), &bvec);
}
```

表示プログラム

```
import numpy as np
import matplotlib.pyplot as plt

wvec = np.loadtxt("bVec")

matx = np.loadtxt("matX")

plt.axis([-5, 5, -5, 5])
lsp = np.linspace(-20, 20, 100)
plt.plot(matx[0, :50], matx[1, :50], 'rx')
plt.plot(matx[0, 50:], matx[1, 50:], 'bx')
plt.plot(lsp, lsp*(wvec[1]/wvec[0]), 'k-')
plt.show()
```