先端データ解析論 (杉山将先生・本多淳也先生)
第 10 回レポート

ashiato45

2017 年 6 月 20 日

宿題 1

$$\sum_{i,i'} W_{i,i'} \|Tx_i - Tx_{i'}\|^2 = \sum_{i,i'} W_{i,i'} (Tx_i - Tx_{i'})^\top (Tx_i - Tx_{i'}) \tag{1}$$

$$= \sum_{i,i'} W_{i,i'} (x_i^\top T^\top T x_i - x_i^\top T^\top T x_{i'} - x_{i'}^\top T^\top T x_i + x_{i'}^\top T^\top T x_{i'}) \tag{2}$$

$$= 2 \left(\sum_{i,i'} W_{i,i'} x_i^\top T^\top T x_i - \sum_{i,i'} W_{i,i'} x_i^\top T^\top T x_{i'} \right) \tag{3}$$

$$= 2 \left(\underbrace{\sum_{i,i',a,b,c} W_{i,i'} X_{ai} T_{ba} T_{bc} X_{ci}}_{\alpha :=} - \underbrace{\sum_{i,i'a,b,c} W_{i,i'} X_{ai} T_{ba} T_{bc} X_{ci'}}_{\beta :=} \right) \tag{4}$$

$$\tag{5}$$

$$(TXDX^\top T^\top)_{ij} = \sum_{a,b,c,d} T_{ia} X_{ab} D_{bc} (X^\top)_{cd} (T^\top)_{dj} \tag{6}$$

$$= \sum_{a,b,c,d} T_{ia} X_{ab} D_{bc} X_{dc} T_{jd} \tag{7}$$

$$= \sum_{a,b,d} T_{ia} X_{ab} (\sum_{i'} W_{b,i'}) X_{db} T_{jd}. \tag{8}$$

よって、

$$\mathrm{tr}(TXDX^\top T^\top) = \sum_{i,i'a,b,d} T_{ia} X_{ab} W_{b,i'} X_{db} T_{id} \tag{9}$$

$$= \alpha. \tag{10}$$

$$(TXWX^\top T^\top)_{ij} = \sum_{a,b,c,d} T_{ia} X_{ab} W_{bc} (X^\top)_{cd} (T^\top)_{dj} \tag{11}$$

$$= \sum_{a,b,c,d} T_{ia} X_{ab} W_{bc} X_{dc} T_{jd}. \tag{12}$$

よって、

$$\mathrm{tr}(TXWX^\top T^\top) = \sum_{a,b,c,d,i} T_{ia} X_{ab} W_{bc} X_{dc} T_{id} \tag{13}$$

$$= \beta. \tag{14}$$

よって、

$$\sum_{i,i'} W_{i,i'} \|Tx_i - Tx_{i'}\|^2 = 2\left(\mathrm{tr}(TXDX^\top T^\top) - \mathrm{tr}(TXWX^\top T^\top)\right) \tag{15}$$

$$= 2\left(\mathrm{tr}(TX(D-W)X^\top T^\top)\right) \tag{16}$$

$$= 2\left(\mathrm{tr}(TXLX^\top T^\top)\right). \tag{17}$$

宿題 2

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。

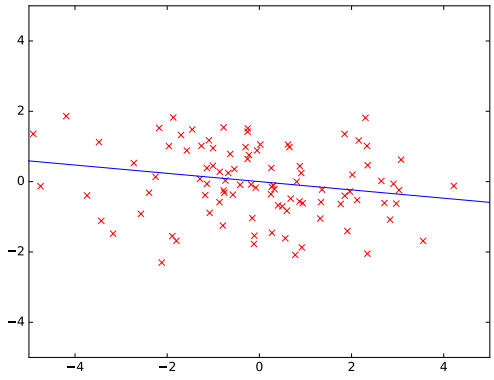

図 1

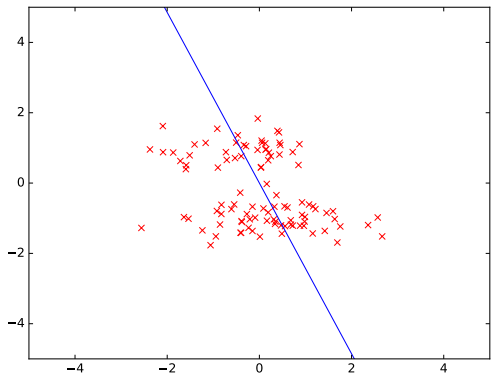

図 2

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;
}

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);

  // MatrixXd matX(2, 100);
  // for(int i=0; i < 100; i++){
  //   matX(0, i) = 2*dist(engine);
  //   matX(1, i) = dist(engine);
  // }
  MatrixXd matX(2, 100);
  for(int i=0; i < 100; i++){
    matX(0, i) = dist(engine);
    matX(1, i) = 2*std::round(unif(engine)) - 1 + dist(engine)/3;
  }

  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(matX*matL*matX.transpose(), matX*matD*matX.transpose(), 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 wvec(2, 1);
  if(ges.eigenvalues()(0).real() < ges.eigenvalues()(1).real()){
    wvec = matTlpp.col(0);
  }else{
    wvec = 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("worstVec");
  filewvec << wvec << std::endl;
}
```

表示プログラム

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

wvec = np.loadtxt("worstVec")

matx = np.loadtxt("points")
```

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