先端データ解析論 (杉山将先生・本多淳也先生)
第 9 回レポート

ashiato45

2017 年 6 月 13 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 を得た。

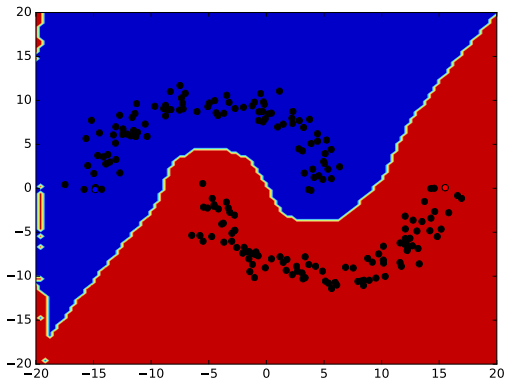

図 1

宿題 2

$$J(\pi) = 2\mathbb{E}_{x' \sim p_{\mathrm{test}}, x \sim q_\pi} \|x' - x\| - \mathbb{E}_{x, \widetilde{x} \sim q_\pi} \|x - \widetilde{x}\| + \mathrm{Const.} \tag{1}$$

が成り立つ。それぞれの項を計算する。

$$\begin{aligned}
\mathbb{E}_{x' \sim p_{\mathrm{test}}, x \sim q_\pi} \|x' - x\| &= \pi \mathbb{E}_{x' \sim p_{\mathrm{test}}, x \sim p_{\mathrm{train}}(x|y=+1)} \|x' - x\| + (1-\pi) \mathbb{E}_{x' \sim p_{\mathrm{test}}, x \sim p_{\mathrm{train}}(x|y=-1)} \|x' - x\| && (2) \\
&= \pi b_{+1} - \pi b_{-1} + \mathrm{Const.} && (3)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_{x, \widetilde{x} \sim q_\pi} \|x - \widetilde{x}\| &= \pi^2 \mathbb{E}_{x \sim p_{\mathrm{train}}(x|y=+1), \widetilde{x} \sim p_{\mathrm{train}}(x|y=+1)} \|x - \widetilde{x}\| && (4) \\
&\quad + 2\pi(1-\pi) \mathbb{E}_{x \sim p_{\mathrm{train}}(x|y=+1), \widetilde{x} \sim p_{\mathrm{train}}(x|y=-1)} \|x - \widetilde{x}\| && (5) \\
&\quad + (1-\pi)^2 \mathbb{E}_{x \sim p_{\mathrm{train}}(x|y=-1), \widetilde{x} \sim p_{\mathrm{train}}(x|y=-1)} \|x - \widetilde{x}\| && (6) \\
&= \pi^2 A_{+1,+1} + 2\pi A_{+1,-1} - 2\pi^2 A_{+1,-1} - 2\pi A_{-1,-1} + \pi^2 A_{-1,-1} + \mathrm{Const.} && (7)
\end{aligned}$$

これらをまとめて、

$$J(\pi) = (2A_{+1,-1} - A_{+1,+1} - A_{-1,-1})\pi^2 - 2(A_{+1,-1} - A_{-1,-1} - b_{+1} + b_{-1})\pi + \text{Const.} \tag{8}$$

を得る。

宿題 3

C++ と Eigen による実装 (表示には Python を用いた) は付録 2 にある。結果、図 2,3 を得た。図 2 は学習データとそれに対する重みなし最小二乗法の結果であり、図 3 はテストデータとそれに対するクラス比重み付き最小二乗法の結果である。

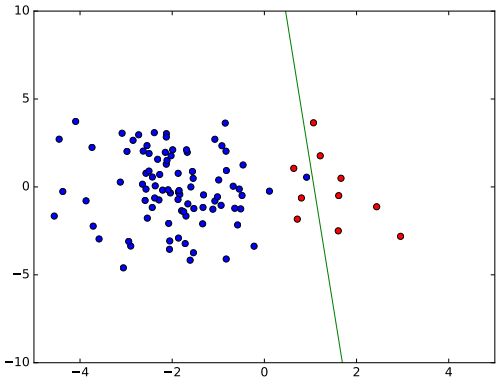

図 2

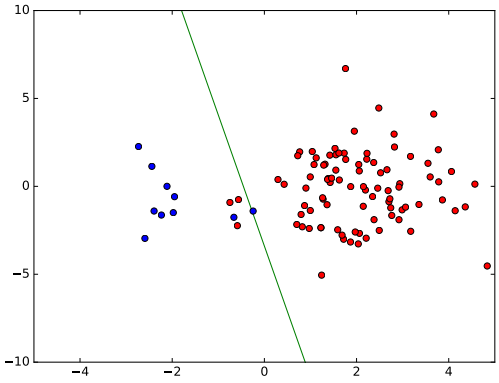

図 3

付録 1

学習プログラム

```
#include <iostream>
#include <Eigen/Dense>
#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;

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

float k(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);

  MatrixXd a(100, 1);
  for(int i=0; i < 100; i++){
    a(i) = M_PI*i/100;
  }
  MatrixXd x(2, 200);
  for(int i=0; i < 100; i++){
      x(0, i) = -10*(std::cos(a(i)) + 0.5) + dist(engine);
      x(1, i) = 10*std::sin(a(i)) + dist(engine);
      x(0, i + 100) = -10*(std::cos(a(i)) - 0.5) + dist(engine);
      x(1, i + 100) = -10*std::sin(a(i)) + dist(engine);
  }
  MatrixXd y(1, 200);
  y(0) = -1;
  y(199) = 1;
  MatrixXd yy(1, 2);
  yy(0) = -1;
  yy(1) = 1;


  /* Learn */
  MatrixXd phi(200, 200);
  for(int i=0; i < 200; i++){
    for(int j=0; j < 200; j++){
      phi(i, j) =  k(x.col(i), x.col(j));
    }
  }
  MatrixXd phit(2, 200);
  for(int j=0; j < 200; j++){
    phit(0, j) = k(x.col(0), x.col(j));
    phit(1, j) = k(x.col(199), x.col(j));
  }
  MatrixXd w = phi;
  MatrixXd d(200, 200);
  for(int i=0; i < 200; i++){
    d(i, i) = w.col(i).sum();
  }
  MatrixXd l = d - w;

  MatrixXd A = phit.transpose()*phit + MatrixXd::Identity(200, 200) + 2*phi.transpose()*l*phi;
  MatrixXd b = phit.transpose()*yy.transpose();
  MatrixXd theta = A.colPivHouseholderQr().solve(b);

  /* Output */
  MatrixXd out(100, 100);
  for(int i=0; i < 100; i++){
    for(int j=0; j < 100; j++){
      float mx = -20*(1.0 - i/100.0) + 20*(i/100.0);
      float my = -20*(1.0 - j/100.0) + 20*(j/100.0);
      for(int kk = 0; kk < 200; kk++){
        MatrixXd xx(2, 1);
        xx << mx, my;
        out(j, i) += theta(kk)*k(x.col(kk), xx);
      }
    }
  }

  std::ofstream file("x");
  file << x << std::endl;
  std::ofstream fileout("out");
  fileout << out << std::endl;
  //std::cout << m << std::endl;
}
```

表示プログラム

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

x = np.loadtxt("x")

v2 = np.loadtxt("out")

plt.axis([-20, 20, -20, 20])
lsp = np.linspace(-20, 20, 100)
plt.contourf(lsp, lsp, np.sign(v2))
plt.plot(x[0, :], x[1, :], 'ko')
plt.plot(x[0, 0], x[1, 0], 'bo')
plt.plot(x[0, 199], x[1, 199], 'ro')
plt.show()
```

付録 2

学習プログラム

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

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

using Eigen::MatrixXd;

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);

  /* Making data */
  MatrixXd px(3, 100);
  MatrixXd py(1, 100);
  for(int i=0; i < 90; i++){
    px(0, i) = dist(engine) - 2;
    px(1, i) = 2*dist(engine);
    px(2, i) = 1;
    py(i) = -1;
  }
  for(int i=90; i < 100; i++){
    px(0, i) = dist(engine) + 2;
    px(1, i) = 2*dist(engine);
    px(2, i) = 1;
    py(i) = 1;
  }
  MatrixXd tx(3, 100);
  for(int i=0; i < 10; i++){
    tx(0, i) = dist(engine) - 2;
    tx(1, i) = 2*dist(engine);
    tx(2, i) = 1;
  }
  for(int i=10; i < 100; i++){
    tx(0, i) = dist(engine) + 2;
    tx(1, i) = 2*dist(engine);
    tx(2, i) = 1;
  }

  /* Learn */
  float hat_a_11 = 0;
  int nn_11 = 0;
  float hat_a_12 = 0;
  int nn_12 = 0;
  float hat_a_22 = 0;
  int nn_22 = 0;
  for(int i=0; i < 100; i++){
    for(int j=0; j < 100; j++){
      if(py(i) == -1 && py(j) == -1){
        hat_a_11 += (px.col(i) - px.col(j)).norm();
        nn_11++;
      }else if(py(i) == -1 && py(j) == 1){
        hat_a_12 += (px.col(i) - px.col(j)).norm();
        nn_12++;
      }else if(py(i) == 1 && py(j) == 1){
        hat_a_22 += (px.col(i) - px.col(j)).norm();
        nn_22++;
      }
    }
  }
  hat_a_11 /= (float)nn_11;
  hat_a_12 /= (float)nn_12;
  hat_a_22 /= (float)nn_22;
  float hat_b_1 = 0;
  int ndn_1 = 0;
  float hat_b_2 = 0;
  int ndn_2 = 0;
  for(int i=0; i < 100; i++){
    for(int j=0; j < 100; j++){
      if(py(j) == -1){
        hat_b_1 += (tx.col(i) - px.col(j)).norm();
        ndn_1++;
      }else if(py(j) == 1){
        hat_b_2 += (tx.col(i) - px.col(j)).norm();
        ndn_2++;
      }
    }
  }
  hat_b_1 /= (float)ndn_1;
  hat_b_2 /= (float)ndn_2;

  float til_pi = (hat_a_12 - hat_a_11 - hat_b_2 + hat_b_1)/(2*hat_a_12 - hat_a_22 - hat_a_11);
  float hat_pi = std::min(1.0f, std::max(0.0f, til_pi));

  /* Learn */
  std::cout << hat_pi << std::endl;
  MatrixXd til_w(100, 100);
  for(int i=0; i < 100; i++){
    if(py(i) == 1){
      til_w(i, i) = hat_pi;
    }else{
      til_w(i, i) = 1.0 - hat_pi;
    }
```

```
  }
  MatrixXd phi = px.transpose();
  MatrixXd theta = (phi.transpose()*til_w*phi).colPivHouseholderQr().solve(phi.transpose()*til_w*py.transpose());
  MatrixXd theta2 = (phi.transpose()*phi).colPivHouseholderQr().solve(phi.transpose()*py.transpose());


  /* Output */
  std::ofstream pxfile("px");
  pxfile << px << std::endl;
  std::ofstream txfile("tx");
  txfile << tx << std::endl;
  std::ofstream thetafile("theta");
  thetafile << theta << std::endl;
  std::ofstream theta2file("theta2");
  theta2file << theta2 << std::endl;
  //std::cout << m << std::endl;
}
```

表示プログラム

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

theta = np.loadtxt("theta")
theta2 = np.loadtxt("theta2")

tx = np.loadtxt("tx")
px = np.loadtxt("px")

plt.axis([-5, 5, -10, 10])
p = np.linspace(-5, 5, 10)
plt.plot(p, -(theta2[2] + p*theta2[0])/theta2[1], 'g-')
plt.plot(px[0, 0:90], px[1, 0:90], 'bo')
plt.plot(px[0, 90:], px[1, 90:], 'ro')
plt.show()

plt.axis([-5, 5, -10, 10])
p = np.linspace(-5, 5, 10)
plt.plot(p, -(theta[2] + p*theta[0])/theta[1], 'g-')
plt.plot(tx[0, 0:10], tx[1, 0:10], 'bo')
plt.plot(tx[0, 10:], tx[1, 10:], 'ro')
plt.show()
```