Лабораторная работа № 7

Вариант № 4

Распознавание образов с использованием машины опорных векторов

Цель работы

Исследовать алгоритмы распознавания образов на основе аппарата машины опорных векторов (Support Vector Machine).

Задание

Воспользовавшись классификатором SVM, определите вероятности ошибок классификации линейно НЕразделимых выборок двух классов для следующих типов ядер: квадратичная функция, полиномиальная функция. Определите оптимальную функцию ядра.

Код программы (внесённые изменения в шаблон кода выделены)

```
% Пример 1. Получение экспериментальной матрицы ошибок по svm
% Вывести матрицы ошибок: теоретическую, экспериментальную (из лабы 3) и
% экспериментальную по svm
%Файл pr53_rec_gaus_uneq. Синтез и анализ алгоритмов распознавания ГСВ с
%различными матрицами ковариации
clear all;
close all;
% Построить график зависимости суммарной экспериментальной ошибки
% первого рода ( для третьего класса) от числа испытаний (объема выборки ).
% Сравнить с теоретическим значением.
KK = 1000;
% Добавляется цикл по объемам выборки
for tt = 1 : numel(KK) % цикл по объемам выборки
    %1.Задание исходных данных
    n=2;M=3;%%размерность признакового пространства и число классов
    K = KK(tt); % количество статистических испытаний
    %Априорные вероятности, математические ожидания и матрицы ковариации классов
    dm=2.0;%расстояние между математическими ожиданиями классов по координатным осям
    C=zeros(n,n,M); C_=C;%матрица ковариации вектора признаков различных классов
    pw=[0.4 0.6 0.5];
    pw=pw/sum(pw);
    D=3*eye(2);
    m=[2 1; -1 -2; -2 1]';
    C(:,:,1)=[3 -1; -1 3];
    C(:,:,2)=[5 3; 3 5];
    C(:,:,3)=[5 -3; -3 5];
    for k=1:M,
        C_(:,:,k)=C(:,:,k)^-1;
    end;
    np=sum(pw); pw=pw/np; %исключение некорректного задания априорных вероятностей
    % 1.1. Генерация обучающих выборок классов
```

```
    % Объемы выборок каждого класса
    Ks = fix(K * pw);
    Ks(end) = K - sum(Ks(1 : end - 1));
    for i=1:M,%цикл по классам
        XN{i} = repmat(m(:,i), [1, Ks(i)]) + randncor(n,Ks(i),C(:,:,i)); %генерация
Ks(i) образов i-го класса
    end;
    % 1.2. Сначала обучаем классификаторы, чтобы на эксперименте дёргать уже
обученные.
    % Поскольку классов может быть 3, а svm осуществляет только попарное
    % сравнение, то обучаем свой классификатор для каждой пары классов
    % (для сравнения 1го и 2го, 2го и 3го, 1го и 3го классов)
    % Обучаем svm-классификаторы для каждой пары классов show = true; % визуализация
результатов обучения (если n=3, можно установить false - 3d пространство не
визуализируется) r = 0.5; % параметр регуляризации (балансирует положение разделяющей
границы относительно положения опорных векторов и "объему" заступов за границу)
    svm_strs = cell(M);
    for i=1:M-1, % цикл по парам классов (как в 4й лабе с 3мя буквами)
        for j=i+1:M,
            % Формирование смешанной обучающей выборки
            data = [XN{i}'; XN{j}']; % помещаем образы
            groups = [true(Ks(i), 1); false(Ks(j), 1)]; % метки классов
            if show, figure; end % создаём графическое окно, если нужно
            % % Классификатор для линейно неразделимых данных (разделяющая граница -
кривая)
            svm_strs{i,j} = svmtrain(data, groups, 'Method', 'QP', 'Kernel_Function',
'rbf', 'rbf_sigma', 0.5, 'showplot', show);
            if show, title(sprintf('class %d and %d', i, j)); end % подпись, если
включена визуализация
        end;
    end;
    %2.Расчет матриц вероятностей ошибок распознавания
    PIJ=zeros(M); PIJB=zeros(M); mg=zeros(M); Dg=zeros(M); l0_=zeros(M);
    for i=1:M,
        for j=i+1:M,
            dmij=m(:,i)-m(:,j);
            l0_(i,j)=log(pw(j)/pw(i));
            dti=det(C(:,:,i)); dtj=det(C(:,:,j));
            trij=trace(C_(:,:,j)*C(:,:,i)-eye(n)); trji=trace(eye(n)-
C_(:,:,i)*C(:,:,j));
            mg1=0.5*(trij+dmij'*C_(:,:,j)*dmij-log(dti/dtj));
            Dg1=0.5*trij^2+dmij'*C_(:,:,j)*C(:,:,i)*C_(:,:,j)*dmij;
            mg2=0.5*(trji-dmij'*C_(:,:,i)*dmij+log(dtj/dti));
            Dg2=0.5*trji^2+dmij'*C_(:,:,i)*C(:,:,j)*C_(:,:,i)*dmij;
            sD1=sqrt(Dg1); sD2=sqrt(Dg2);
            PIJ(i,j)=normcdf(l0_(i,j),mg1,sD1); PIJ(j,i)=1-normcdf(l0_(i,j),mg2,sD2);
            mu2=(1/8)*dmij'*((C(:,:,i)/2+C(:,:,j)/2)^-1)*dmij...
                +0.5*log((dti+dtj)/(2*sqrt(dti*dtj)));%расстояние Бхатачария
            PIJB(i,j)=sqrt(pw(j)/pw(i))*exp(-mu2);PIJB(j,i)=sqrt(pw(i)/pw(j))*exp(-
mu2);%границы Чернова
        end;
        PIJB(i,i)=1-sum(PIJB(i,:));
        PIJ(i,i)=1-sum(PIJ(i,:));%нижняя граница вероятности правильного
распознавания
    end;
    %3.Тестирование алгоритма методом статистических испытаний
    Pcv=zeros(M); % инициализация экспериментальной матрицы ошибок
    x=ones(n,1); u=zeros(M,1);
    Pc_=zeros(M);%экспериментальная матрица вероятностей ошибок из 3й лабы
    for k=1:K,%цикл по числу испытаний
        for i=1:M,%цикл по классам
            [x,px]=randncor(n,1,C(:,:,i)); x=x+m(:,i);%генерация образа i-го класса
```

```
            for j=1:M,%вычисление значения разделяющих функций из 3й лабы
                u(j)=-0.5*(x-m(:,j))'*C_(:,:,j)*(x-m(:,j))-
0.5*log(det(C(:,:,j)))+log(pw(j));
            end;
            [ui,iai]=max(u);%определение максимума
            Pc_(i,iai)=Pc_(i,iai)+1;%фиксация результата распознавания
            % Прогон по всем классификаторам в том же порядке
            % как мы их обучали
            iais = []; % массив для фиксации результатов от разных классификаторов
            for ii=1:M-1 % цикл по парам классов
                for jj=ii+1:M
                    % Вызываем классификатор для сравнения ii-го и jj-го
                    % классов из массива svm_strs и распознаём им образ x
                    cl = svmclassify(svm_strs{ii,jj}, x', 'showplot', false);
                    if cl % cl будет true, если класс ii-й
                        iai = ii;
                    else % cl будет false, если класс jj-й (потому что так
формировался массив groups при обучении)
                        iai = jj;
                    end
                    iais = [iais, iai]; % фиксируем результат распознавания
                end
            end
            iai = mode(iais); % выбираем класс, за который проголосовало большинство
классификаторов
            Pcv(i,iai)=Pcv(i,iai)+1;% %фиксация результата распознавания
        end; %цикл по классам
    end;
    Pc_=Pc_/K;
    Pcv=Pcv/K; % нормировка экспериментальной матрицы по svm на число испытаний
end % конец цикла по объемам выборки
% Вывести матрицы ошибок
disp('Теоретическая матрица вероятностей ошибок');disp(PIJ);
% disp('Матрица вероятностей ошибок на основе границы Чернова');disp(PIJB);
disp('Экспериментальная матрица вероятностей ошибок');disp(Pc_);
disp('Экспериментальная матрица вероятностей ошибок');disp(Pcv);
```

Результаты выполнения задания

Результаты тестирования алгоритма на основе метода SVM при линейно не разделимых классах:

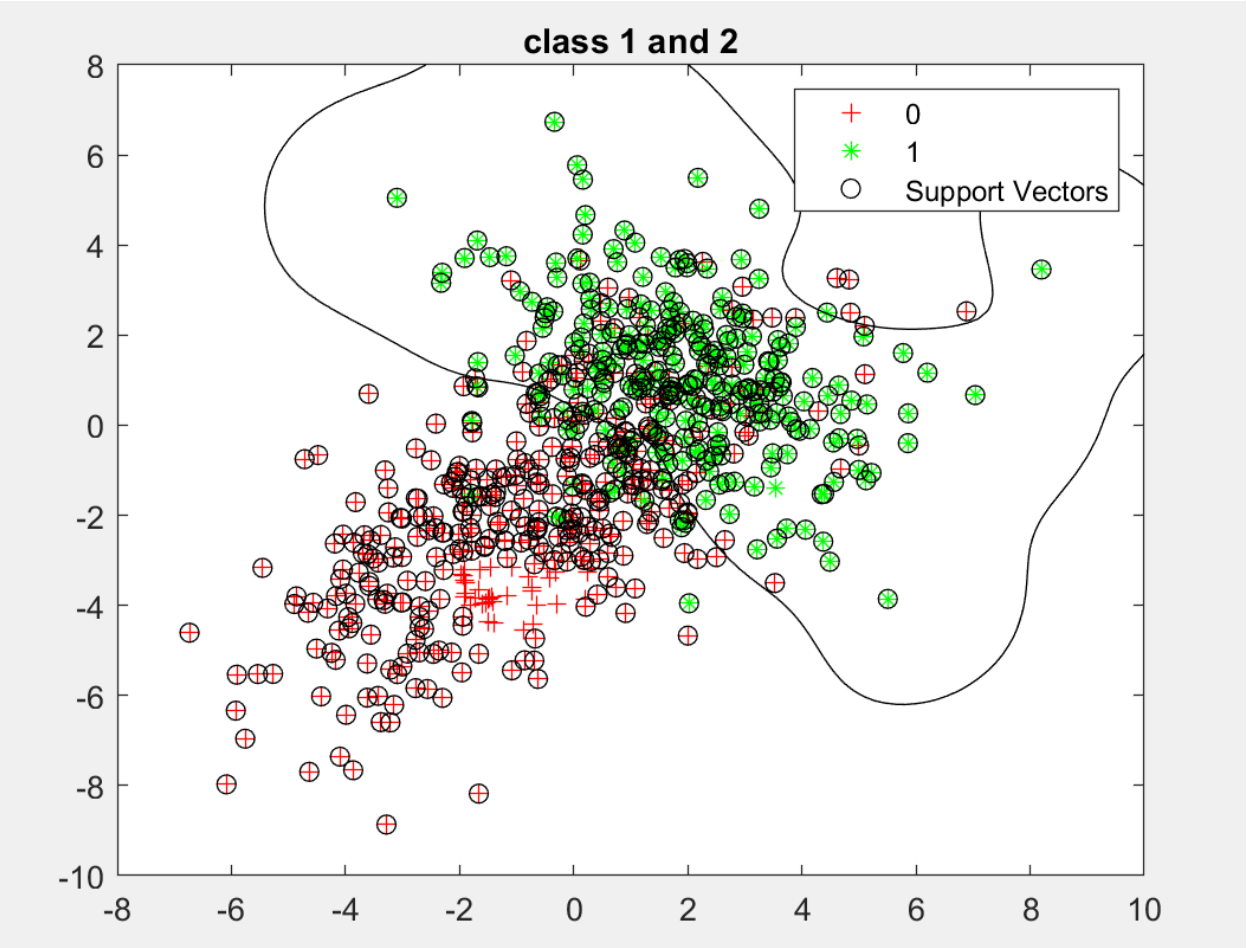

Рисунок 1 – Классы 1 и 2

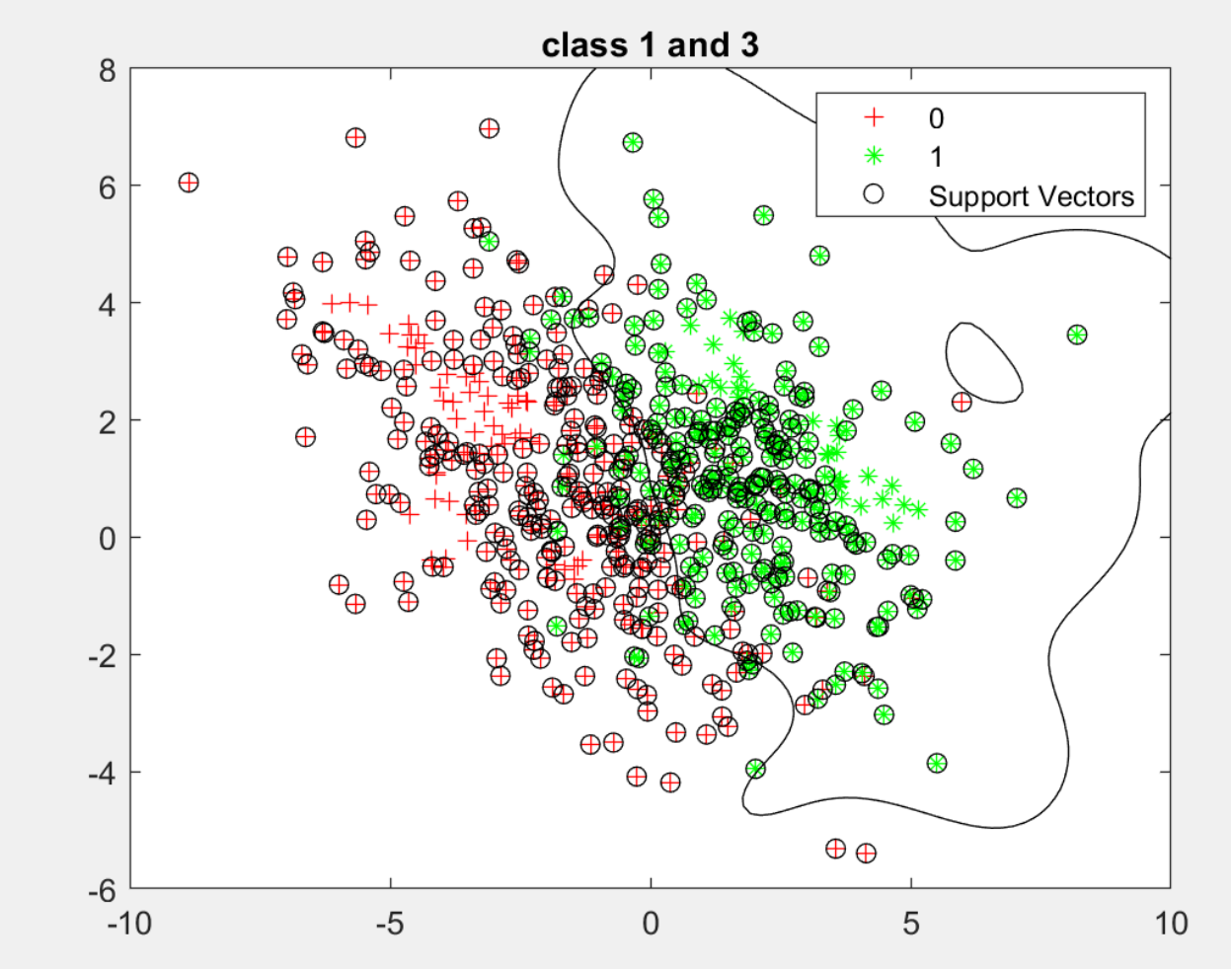

Рисунок 2 – Классы 1 и 3

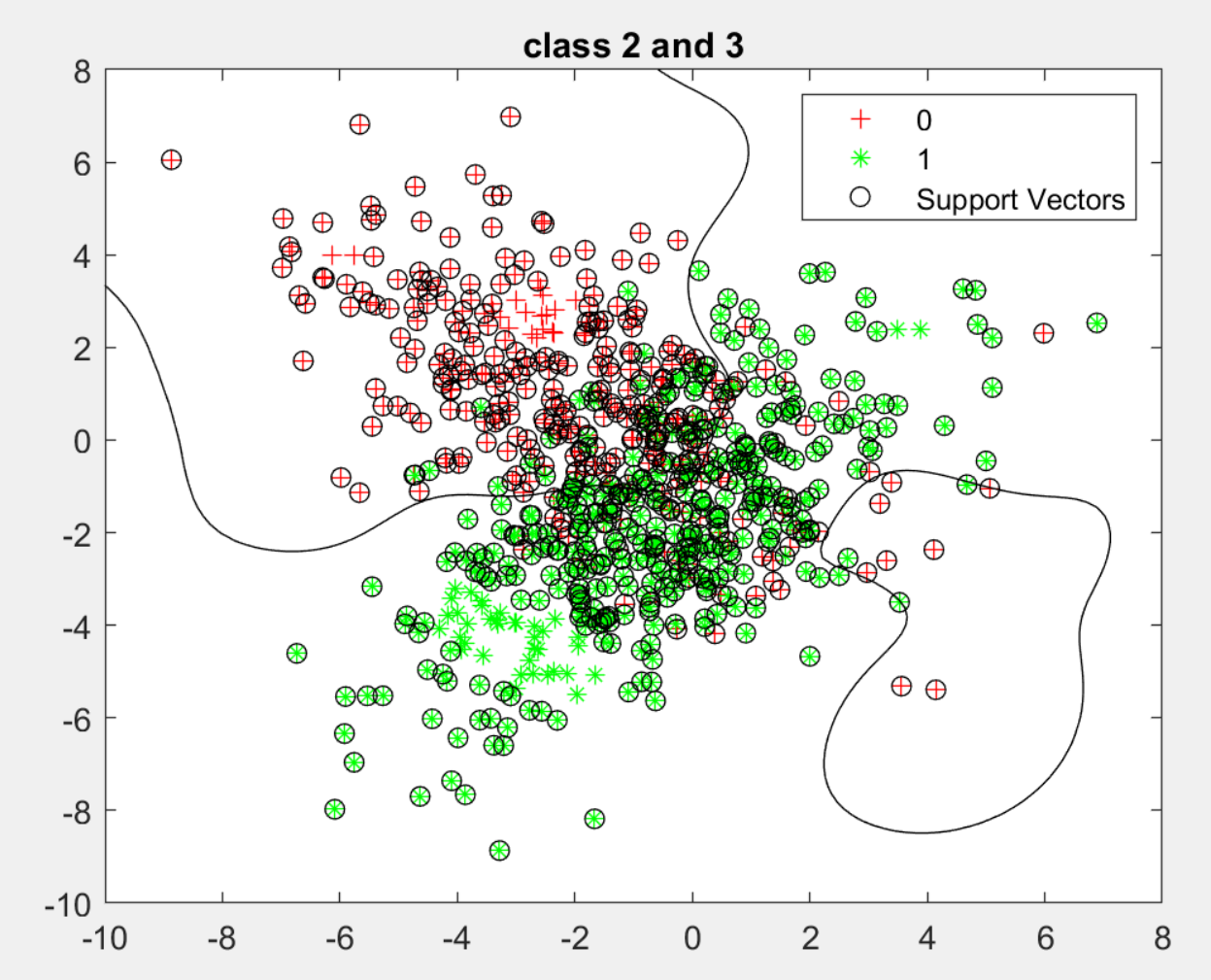

Рисунок 3 – Классы 2 и 3

Теоретическая матрица вероятностей ошибок

0.8040	0.0611	0.1349
0.1765	0.6357	0.1878
0.1234	0.2384	0.6382

Экспериментальная матрица вероятностей ошибок

0.8240	0.0920	0.0840
0.1580	0.7590	0.0830
0.0880	0.2450	0.6670

Экспериментальная матрица вероятностей ошибок

0.8400	0.0660	0.0940
0.1700	0.6920	0.1380
0.1000	0.1950	0.7050

Рисунок 4 – Матрицы вероятностей ошибок

Выводы

1. Какой смысл имеет параметр регуляризации C?

Параметр регуляризации C в методе опорных векторов (Support Vector Machines, SVM) контролирует баланс между достижением точности на обучающем наборе данных и уменьшением сложности модели. Он используется для предотвращения переобучения.

Параметр C является обратным коэффициентом регуляризации: чем меньше значение C, тем сильнее регуляризация, и модель будет более простой, предпочитая выбрать больше ошибок на обучающем наборе, но с более простой гиперплоскостью. С другой стороны, при больших значениях C модель будет стремиться к правильному классифицированию каждого обучающего примера, что может привести к более сложным моделям.

Метод SVM в большинстве задач обладает хорошей эффективностью. В тоже время отмечаются его недостатки, в том числе неустойчивость к шумовым искажениям обучающих данных, необходимость подбора параметра C, необходимость обоснования или подбора вида ядра.

2. Какой вид функции ядра обеспечивает наилучшее качество оценивания?

Видим более высокую эффективность алгоритма, использующего функцию ядра в виде радиально-базисной функции

RBF-ядро обычно считается более универсальным и широко используется. Оно подходит для различных типов данных и способно работать с нелинейными структурами.