JSS30, Summer School, COMP5: Machine learning in inverse and ill-posed problems Course Project

Larisa Beilina, larisa@chalmers.se

Instructions

- You can work in groups by 2 persons.
- Sent final report for every computer assignment with description of your work together with Matlab or C++/PETSc programs to my e-mail before deadline. Report should have description of used techniques, tables and figures confirming your investigations. Analysis of obtained results is necessary to present in section "Numerical examples" and summarized results - in section "Conclusions". You can download latex or pdf-template for report from the course homepage.
- Matlab and C++ programs for examples in the book [1] are available for download from the course homepage: go to the link of the book [1] and click to "GitHub Page with MATLAB® Source Codes" on the bottom of this page, or copy the link below:

  `https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`

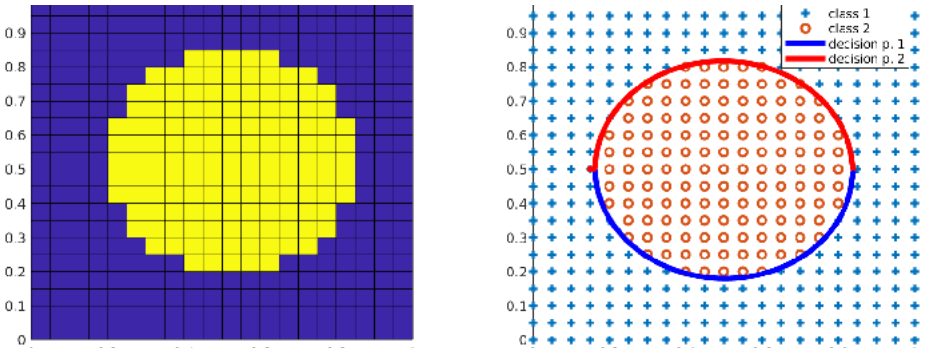

Figure 0.1: Classification of the computed solution for Poisson's equation on the unit square (see example 8.1.3 of [1]) for number of mesh points 21^2 using perceptron learning algorithm.

Course Project

Regularized Least squares and machine learning algorithms for classification problem

In this project we will study regularized versions of least squares and perceptron learning algorithms for solution of classification problem presented in the paper *Numerical analysis of least squares and perceptron learning for classification problems* which can be downloaded from the link

`https://arxiv.org/pdf/2004.01138.pdf`

Details about AI algorithms for classification together with machine learning techniques for choosing the reg.parameter can be found in [2, 3, 4].

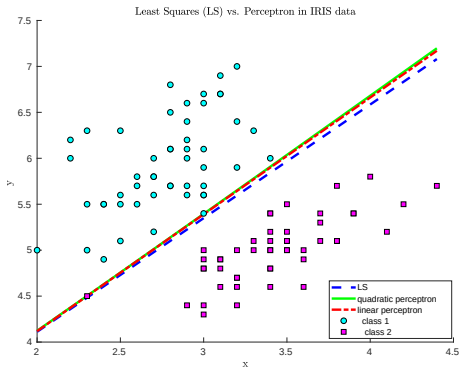

Figure 0.2: Decision lines computed by least squares and the perceptron learning algorithm for separation of two classes using Iris dataset. The dataset iris.csv is available for download from the course page.

Project assignments

Implement in MATLAB studied in the course regularized classification algorithms and present decision lines for following training sets:

- 0) Study non-regularized least squares and perceptron learning algorithms implemented in the Matlab code of section 1 for classification of IRIS flower data set used in the work []. The dataset can be downloaded from the link:

`https://en.wikipedia.org/wiki/Iris_flower_data_set`

- I) Classify IRIS flower data set into several classes using regularized versions of least squares and perceptron learning algorithms.
- II) Classify datapoints which are inside the circle $x^2 + y^2 = r^2$ for some $r > 0$, with code 1, and which are outside circle, with code -1 (choose by yourself number of datapoints which will belong to both classes). Determine decision line computed by the quadratic perceptron algorithm.
- III) Use support vector machines (SVM) to classify points generated in item II). Compare obtained decision line with the decision line computed by the quadratic perceptron.
- IV) Take some experimental data for classification from the link

  `https://archive.ics.uci.edu/ml/datasets.html`

  or the link for skin images:

  `https://www.isic-archive.com/#!/topWithHeader/wideContentTop/main`

  and classify them using regularized versions of least squares and perceptron learning algorithms.
- V) Compute missclassification rate E using the formula (see [4], p. 211-214):

$$\mathrm{E} = \frac{\sum_{i=1}^{K} N_{F,i}}{\sum_{i=1}^{K}(N_{T,i} + N_{F,i})}, \tag{0.1}$$

  where K is the number of classes, $N_{T,i}$ is the number of points of the class i which are classified correctly, $N_{F,i}$ is the number of points of the class i which are classified wrong. Precision for class i can be computed as

$$\mathrm{P}(i) = \frac{N_{T,i}}{N_{T,i} + N_{F,j}}. \tag{0.2}$$

Try answer to the following questions:

- Analyze effect of using different regularization strategies for classification.
- Analyze what happens with performance of perceptron learning algorithm if we take different learning rates $\eta \in (0, 1]$? For what values of η perceptron learning algorithm is more sensitive and when the iterative process is too slow?
- Analyze which one of the studied classification algorithms perform best and why?
- Try to explain why least squares approach can fail in the case when usual linear classifier is used.

Programs

1.1 Main Matlab program for classification

```
% Classification of Iris dataset  into 2 classes using linear least squares,
%  linear and quadratic perceptron learning  algorithms

 clear
 close all

%% Load the Iris data set
dataset_name = "Iris";
data = csvread("iris.csv");
Xiris = data(:, 1)';
Yiris = data(:, 2)';
Ciris = data(:, 3)';

x = Xiris(1, :);
y = Yiris(1, :);
class = Ciris(1, :);
d=1;  % degree of the polynomial

%number of discretization points or rows in the matrix A
m=size(x,2);

A=[];

classx1 = x(class==1);
classx2 = x(class==0);

classy1 = y(class==1);
classy2 = y(class==0);

hyp =   zeros(1,m);

%  LS for problem min ||A omega - t||   where A =[1; x; y], t = class
A = [ones(numel(x), 1) x' y'];
w = A\(class');

decision_linels = zeros(1,2);
coor(1)= min(x);
  coor(2)=max(x);
for i=1:1:2
decision_linels(i) =  (0.5-w(1))/w(3) - (w(2)/w(3))*coor(i);
end

%*************************************************************************
%******************  perceptron learning algorithm **********************
```

```
%************************************************************************
% init weights
weight= rand(3,1);
hyp =   zeros(1,m);

x = Xiris(1, :);
y = Yiris(1, :);
class = Ciris(1, :);


% init learning rate
eta = 0.5;
while      sum(class ~= hyp)
     count = 0;
     for i=1:1:m
        if   weight(1) + weight(2)*x(i)  + weight(3)*y(i) > 0
        hyp(i)=1;
    else
        hyp(i)=0;
        end

     %update weights

     weight(1) = weight(1) + eta*(class(i) - hyp(i));
     weight(2) = weight(2) + eta*(class(i) - hyp(i))*x(i);
     weight(3) = weight(3) + eta*(class(i) - hyp(i))*y(i);

end

count = count+1;
   % break out loop to avoid infnite loop
   if count > 1000
       count
       break
    end
end

decision_line = zeros(1,2);
coor(1)= min(x);
  coor(2)=max(x);
for i=1:1:2

decision_line(i) =  -weight(1)/weight(3) - (weight(2)/weight(3))*coor(i);
end

%************************************************************************
% *****  division into 2 classes for quadratic perceptron  ***************
%************************************************************************
```

```
hyp =   zeros(1,m);

x = Xiris(1, :);
y = Yiris(1, :);
class = Ciris(1, :);

%*****************  quadratic classification ******************************

% init weights
weight= rand(6,1);

% init learning rate
eta = 0.5;
count = 0;
   while      sum(class ~= hyp)

     for i=1:1:m
          if   weight(1) + weight(2)*x(i)  + weight(3)*y(i)  + ...
               weight(4)*x(i)*x(i)  + weight(5)*x(i)*y(i) + ...
               weight(6)*y(i)*y(i)  > 0
        hyp(i)=1;
    else
        hyp(i)=0;
        end

     %update weights

     weight(1) = weight(1) + eta*(class(i) - hyp(i));
     weight(2) = weight(2) + eta*(class(i) - hyp(i))*x(i);
     weight(3) = weight(3) + eta*(class(i) - hyp(i))*y(i);
     weight(4) = weight(4) + eta*(class(i) - hyp(i))*x(i)*x(i);
     weight(5) = weight(5) + eta*(class(i) - hyp(i))*x(i)*y(i);
     weight(6) = weight(6) + eta*(class(i) - hyp(i))*y(i)*y(i);

     end

     count = count+1;

   if count > 10000
     %  count
       break
    end
end


 a=weight(6);
ynew1=zeros(1,m);
```

```
ynew2=zeros(1,m);

x=sort(x);

 for i=1:1:m
bb= weight(5)*x(i) + weight(3);
cc = weight(1) + weight(2)*x(i) + weight(4)*x(i)*x(i);
D = bb*bb - 4*a*cc;
ynew1(i) = (-bb + sqrt(D))/(2*a);
ynew2(i) = (-bb - sqrt(D))/(2*a);
 end

%*********************************************************
% presenting results

figure
%plot(x,y,'o r', 'linewidth',1)
%hold on

% compute approximation to this exact polynomial with comp. coefficients c

%approx = A*c;


hold on
% plot decision line computed via Least squares
%plot(x,approx,'-- b', 'linewidth',3)
plot(coor,decision_linels,'-- b', 'linewidth',3);
% plot results of quadratic perceptron
  plot(x,ynew1,'- g', 'linewidth',3);
  %plot(x,ynew2,'* r', 'linewidth',2);
% plot results of  linear  perceptron
plot(coor,decision_line,' -. r', 'linewidth',3)
str_xlabel = ['poly.degree d=', num2str(d)];
xlabel('  x')

 plot(classx1, classy1,"ko", "MarkerSize", 7, "MarkerFaceColor", "c")
 plot(classx2,  classy2,"ks", "MarkerSize", 7, "MarkerFaceColor", "m")

%legend('exact  ',str_xlabel);
legend('LS', 'quadratic perceptron', 'linear perceptron', '  class 1', '   class 2');

%title(['LS vs. Perceptron, m= ',num2str(m),', ',str_xlabel])
hold off

title("Least Squares (LS) vs. Perceptron in IRIS data", "Interpreter", "Latex")
xlabel("x", "Interpreter", "Latex")
ylabel(" y ", "Interpreter", "Latex")
```

```
font_size = 12;
set(gca, "FontSize", font_size)

  % Export the plot
%set(gcf, "Units", "Inches", "Position", [0, 0, 7, 7], ...
%   "PaperUnits", "Inches", "PaperSize", [5, 5])
 %  saveas(gcf, sprintf("Iris.pdf"))
```

References

[1] L. Beilina, E. Karchevskii, M. Karchevskii, Numerical Linear Algebra: Theory and Applications, Springer, 2017.

[2] Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.

[3] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`

[4] Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.