Mustererkennung WiSe 12/13
Übung 5

Lutz Freitag, Sebastian Kürten

1 Aufgabe 1: Ridge Regression

1.1 Plot

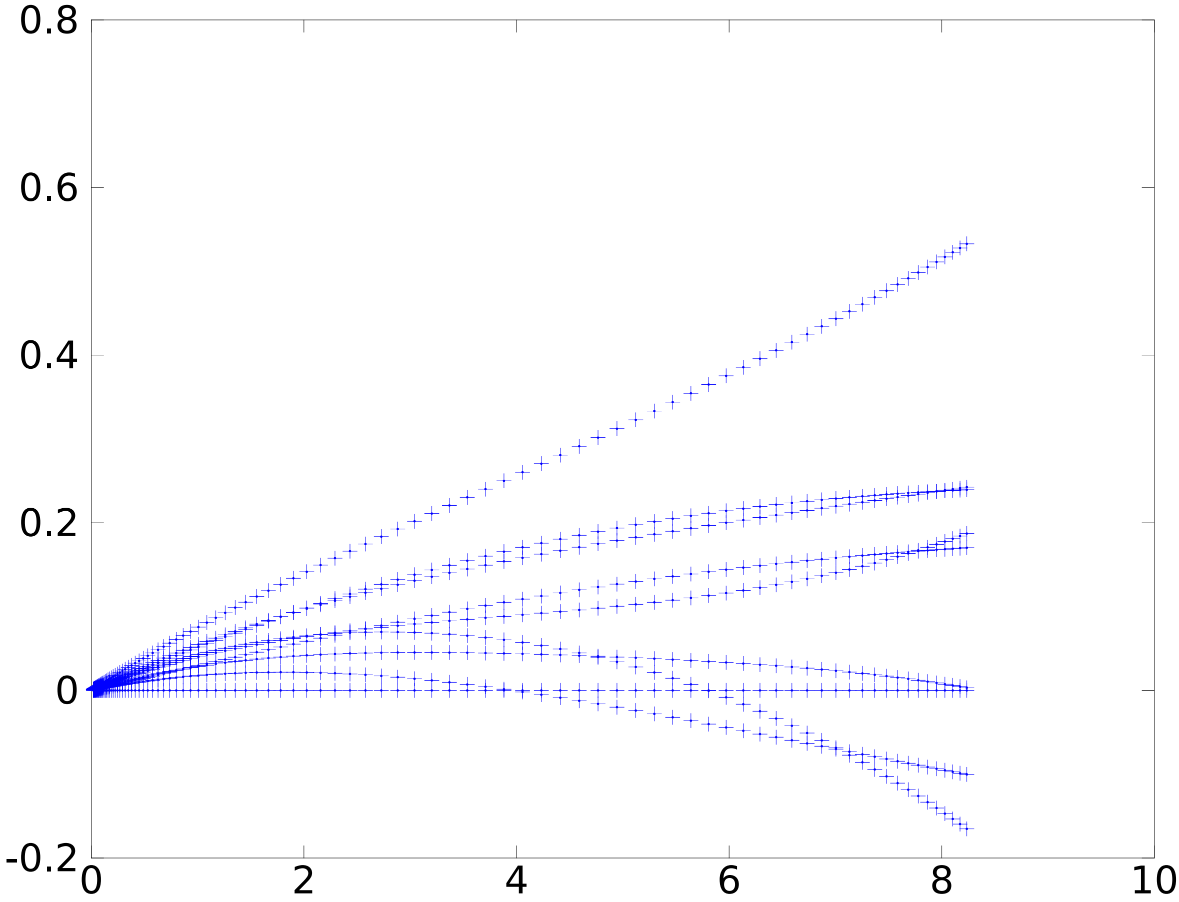

Figure 1: Darstellung der Features

1.2 Code

1.2.1 a1.m

```
% Daten laden
data = load('data.mod');
% Testdaten von Trainingdaten trennen:
% Bei Trainingsdaten steht in der letzten Spalte eine 0,
% bei den Testdaten steht in der letzten Spalte eine 1.
% Außerdem weglassen der ersten Spalte, dort steht die Nummer
training = data(data(:,11) == 0,2:end-1);
testing  = data(data(:,11) == 1,2:end-1);
```

```
cnt = size(training)(1)

% normalisiere trainingsdaten
meanTraining = mean(training)
training = training .- repmat(meanTraining, cnt, 1);
varTraining = std(training,1)
normalizedTraining = training ./ repmat(varTraining, cnt, 1)

% Trenne abhängige von unabhängigen Daten
known = normalizedTraining(:,1:end-1); % lcavol, lweight, ..., pgg45

y = normalizedTraining(:,end); % lpsa

% Bestimme X durch Anfügen von 1-en vor die erste Spalte
X = augmentWithOnes(known);

hold on

% calculate singular value decomposition to get eigenvalues of X
d = svd(X);


for lambda = logspace(0.5,4.5,100) % chosen by looking
        % Alpha bestimmen mit Formel für ridge Regression
        alpha = (X' * X + lambda * eye(size(X' * X)))^(-1) * X' * y;
        dl = sum(d .^2 ./ (d .^2 + lambda))
        plot(repmat(dl, size(alpha)), alpha, "@")
end
hold off

print("a1.png");
print("a1.pdf");
```

2 Aufgabe 2: Bootstrap

Die Ausgabe sind Mittelwerte und Standardabweichung der ersten zehn Datensätze bezogen auf die 100 vorhergesagten Werte mit Hilfe linearer Regression auf Basis zufällig gewählter Trainingsgruppen der Größe 50.

2.1 Ausgabe

```
mus =

   1.159   0.921   1.279   0.800   1.925   0.889   1.917   1.892   1.046   1.621

sigmas =

 Columns 1 through 8:

   0.0597   0.0828   0.0618   0.0966   0.0246   0.0863   0.0246   0.0249

 Columns 9 and 10:

   0.0699   0.0314
```

2.2 Code

2.2.1 a2.m

```
% Daten laden
data = load('data.mod');
data = data(:,2:end-1);

% relevante Daten auswählen
sdata = data(:, [1 5 7 end]);

% hier speichern wir die Koeffizienten,
% ein Quadrupel von Koeffizienten je Zeile
coefficients = [];

% 100 Experimente durchführen
for x = 1:100
        % 50 Datensätze auswählen
        idx = randint(50, 1, [1,size(data, 1)]);
        d = sdata(idx, :);
        % Koeffizenten berechnen und abspeichern
        alphas = linreg(d)';
        coefficients = [coefficients; alphas];
end

% 10 Testdatensätze nehmen und Einsen anfügen
knownTest = sdata(1:10,1:end-1);
Xtest = augmentWithOnes(knownTest);

% Vorhersagen mit allen Koeffizientenvektoren
predictions = Xtest * coefficients';
% Mittelwert und Standardabweichung ausrechnen
mus = mean(predictions, 2);
diffs = predictions - repmat(mus, 1, size(predictions, 2));
sigmas = sum(diffs.^2, 2) / size(predictions, 2);

```

```
% Ausgabeoptionen
output_max_field_width(3);

% für die Ausgabe transponieren :)
mus = mus'
sigmas = sigmas'
```

3 Aufgabe 3: Experiment

Wir haben hier mit der 8 gearbeitet. Die Beobachtung ist, dass die x_k konvergieren, d.h. dass sich die Richtung der Vektoren nich mehr ändert.

3.1 Code

3.1.1 a3.m

```
% Daten laden
trainingData = load("-ascii", "pendigits-training.txt");
% wir arbeiten mit der Acht
digit = 8;

% select all samples labeled with 'digit'
samples = trainingData(trainingData(:,17) == digit,:)(:,1:end - 1);

% compute mu and covanriance matrix
[mu, cov] = gauss(samples);

% plot the mean vector
plotDigit(mu, int2str(digit), 'mean.pdf', 'mean.png')

% select a random vector as x0
x0 = normalize(rand(1, 16)');
% start with xk = x0
xk = x0;
for k = 0:14
        clf;
        % create a plot of the iteration
        label = sprintf('iteration %d', k);
        pdfname = sprintf('iteration%d.pdf', k);
        pngname = sprintf('iteration%d.png', k);
        plotDigit(xk, label, pdfname, pngname);
        % multiply xk with the covariance matrix
        xk = normalize(cov * xk);
end
```

3.2 Plots

Siehe unten

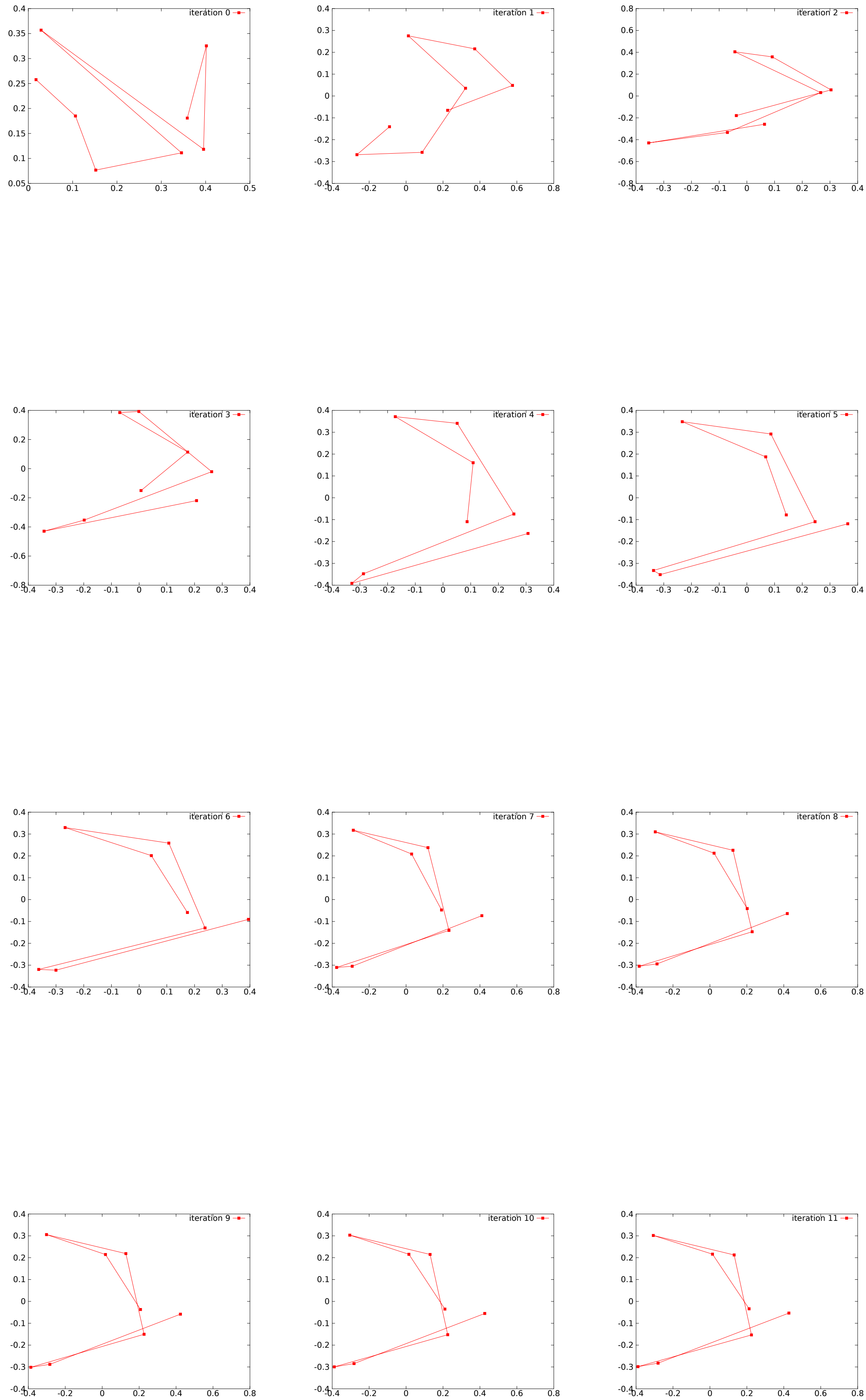

Figure 2: Darstellung der ersten x Iterationen