

1 Aufgabe 1: Ridge Regression

1.1 Plot

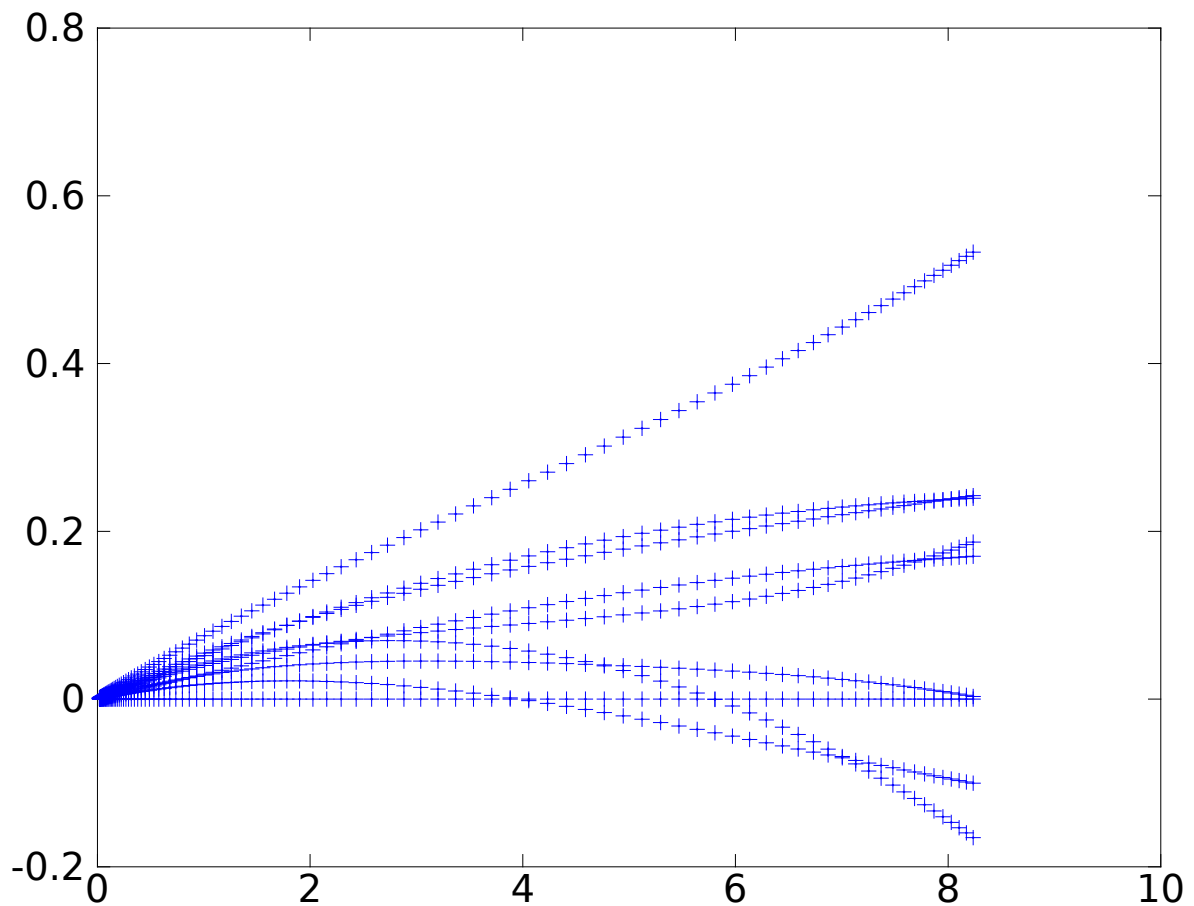


Figure 1: Darstellung der Features

1.2 Code

1.2.1 a1.m

```
1 % Daten laden
2 data = load('data.mod');
3 % Testdaten von Trainingsdaten trennen:
4 % Bei Trainingsdaten steht in der letzten Spalte eine 0,
5 % bei den Testdaten steht in der letzten Spalte eine 1.
6 % Außerdem weglassen der ersten Spalte, dort steht die Nummer
7 training = data(data(:,11) == 0,2:end-1);
8 testing = data(data(:,11) == 1,2:end-1);
```

```

9
10 cnt = size(training)(1)
11
12 % normalisiere trainingsdaten
13 meanTraining = mean(training)
14 training = training ./ repmat(meanTraining, cnt, 1);
15 varTraining = std(training,1)
16 normalizedTraining = training ./ repmat(varTraining, cnt, 1)
17
18 % Trenne abhängige von unabhängigen Daten
19 known = normalizedTraining(:,1:end-1); % lcavol, lweight, ..., pgg45
20
21 y = normalizedTraining(:,end); % lpsa
22
23 % Bestimme X durch Anfügen von 1-en vor die erste Spalte
24 X = augmentWithOnes(known);
25
26 hold on
27
28 % calculate singular value decomposition to get eigenvalues of X
29 d = svd(X);
30
31
32 for lambda = logspace(0.5,4.5,100) % chosen by looking
33     % Alpha bestimmen mit Formel für ridge Regression
34     alpha = (X' * X + lambda * eye(size(X' * X)))^(-1) * X' * y;
35     dl = sum(d.^2 ./ (d.^2 + lambda))
36     plot(repmat(dl, size(alpha)), alpha, "@")
37 end
38 hold off
39
40 print("a1.png");
41 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

```
1 % Daten laden
2 data = load('data.mod');
3 data = data(:,2:end-1);
4
5 % relevante Daten auswählen
6 sdata = data(:, [1 5 7 end]);
7
8 % hier speichern wir die Koeffizienten,
9 % ein Quadrupel von Koeffizienten je Zeile
10 coefficients = [];
11
12 % 100 Experimente durchführen
13 for x = 1:100
14     % 50 Datensätze auswählen
15     idx = randint(50, 1, [1, size(data, 1)]);
16     d = sdata(idx, :);
17     % Koeffizienten berechnen und abspeichern
18     alphas = linreg(d)';
19     coefficients = [coefficients; alphas];
20 end
21
22 % 10 Testdatensätze nehmen und Einsen anfügen
23 knownTest = sdata(1:10, 1:end-1);
24 Xtest = augmentWithOnes(knownTest);
25
26 % Vorhersagen mit allen Koeffizientenvektoren
27 predictions = Xtest * coefficients';
28 % Mittelwert und Standardabweichung ausrechnen
29 mus = mean(predictions, 2);
30 diffs = predictions - repmat(mus, 1, size(predictions, 2));
31 sigmas = sum(diffs.^2, 2) / size(predictions, 2);
32
```

```
33 % Ausgabeoptionen
34 output_max_field_width(3);
35
36 % für die Ausgabe transponieren :)
37 mus = mus'
38 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 nicht mehr ändert.

3.1 Code

3.1.1 a3.m

```
1 % Daten laden
2 trainingData = load("-ascii", "pendigits-training.txt");
3 % wir arbeiten mit der Acht
4 digit = 8;
5
6 % select all samples labeled with 'digit'
7 samples = trainingData(trainingData(:,17) == digit,:)(:,1:end - 1);
8
9 % compute mu and covariance matrix
10 [mu, cov] = gauss(samples);
11
12 % plot the mean vector
13 plotDigit(mu, int2str(digit), 'mean.pdf', 'mean.png')
14
15 % select a random vector as x0
16 x0 = normalize(rand(1, 16)');
17 % start with xk = x0
18 xk = x0;
19 for k = 0:14
20     clf;
21     % create a plot of the iteration
22     label = sprintf('iteration%d', k);
23     pdfname = sprintf('iteration%d.pdf', k);
24     pngname = sprintf('iteration%d.png', k);
25     plotDigit(xk, label, pdfname, pngname);
26     % multiply xk with the covariance matrix
27     xk = normalize(cov * xk);
28 end
```

3.2 Plots

Siehe unten

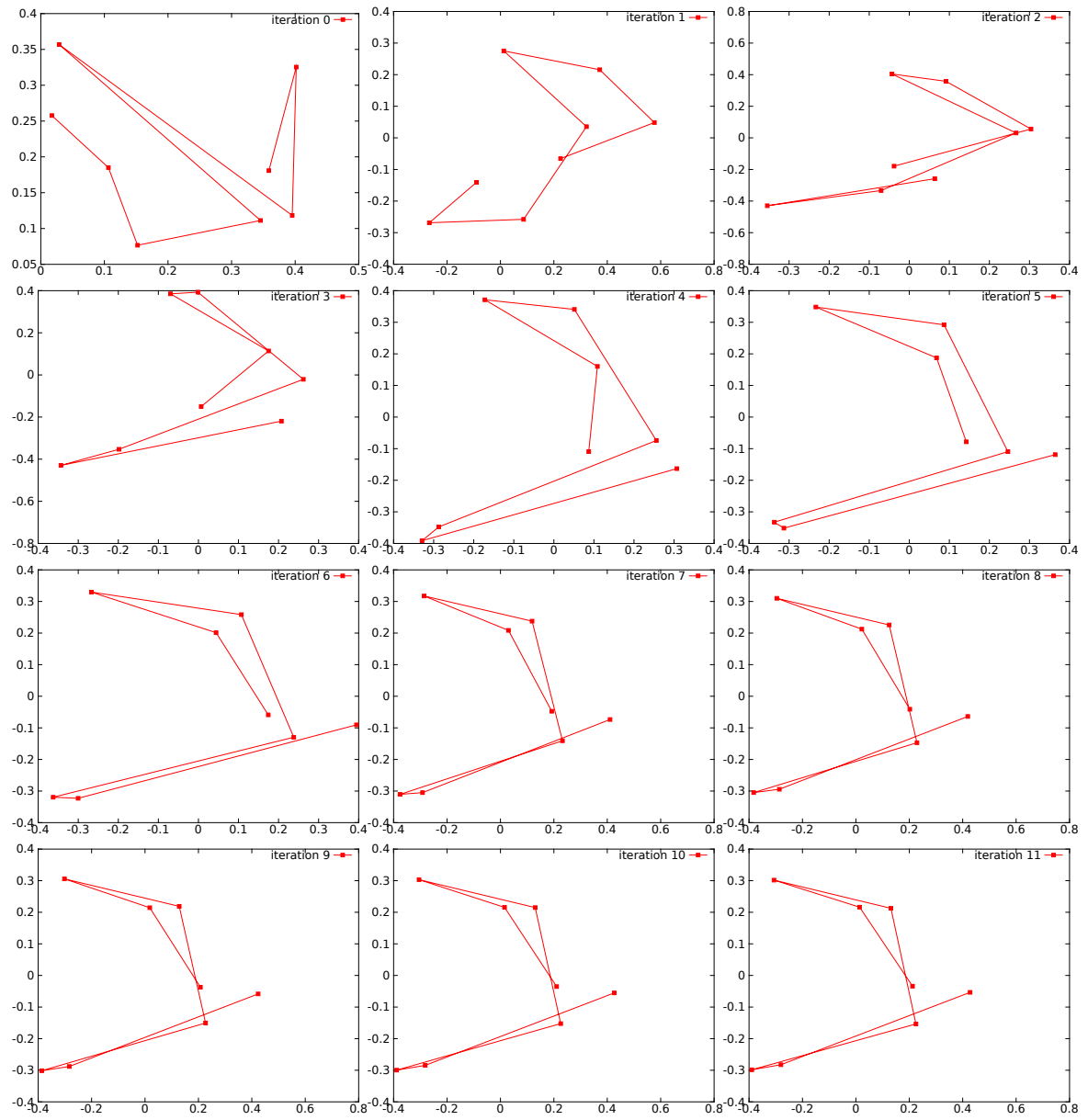


Figure 2: Darstellung der ersten x Iterationen