Peter Gruber
gruberpr@cs.umass.edu

Machine Learning Assignment 2

Question 1

Part 1.1

Part 1.2

$$\hat{A}_k = [U_1, \ldots, U_k] \cdot [\Sigma_1, \ldots, \Sigma_k] \cdot [V_1, \ldots V_k]^T$$

Proof:

$$= \begin{bmatrix} U_{1,1} & \cdots & U_{k,1} \\ \vdots & \ddots & \vdots \\ U_{1,m} & \cdots & U_{k,m} \end{bmatrix} \cdot \begin{bmatrix} \Sigma_{1,1} & \cdots & \Sigma_{n,1} \\ \vdots & \ddots & \vdots \\ \Sigma_{1,k} & \cdots & \Sigma_{n,k} \end{bmatrix} \cdot \begin{bmatrix} V_{1,1} & \cdots & V_{k,1} \\ \vdots & \ddots & \vdots \\ V_{1,n} & \cdots & V_{k,n} \end{bmatrix}^T$$

$$= \begin{bmatrix} \sum_{i=1}^{k} U_{i,1} \cdot \Sigma_{1,i} & \cdots & \sum_{i=1}^{k} U_{i,1} \cdot \Sigma_{n,i} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} U_{i,m} \cdot \Sigma_{1,k} & \cdots & \sum_{i=1}^{k} U_{i,m} \cdot \Sigma_{n,i} \end{bmatrix} \cdot \begin{bmatrix} V_{1,1} & \cdots & V_{k,1} \\ \vdots & \ddots & \vdots \\ V_{1,n} & \cdots & V_{k,n} \end{bmatrix}^T$$

$$= \begin{bmatrix} \sum_{i=1}^{k} U_{i,1} \cdot \Sigma_{1,i} & \cdots & \sum_{i=1}^{k} U_{i,1} \cdot \Sigma_{n,i} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} U_{i,m} \cdot \Sigma_{1,k} & \cdots & \sum_{i=1}^{k} U_{i,m} \cdot \Sigma_{n,i} \end{bmatrix} \cdot \begin{bmatrix} V_{1,1} & \cdots & V_{1,n} \\ \vdots & \ddots & \vdots \\ V_{k,1} & \cdots & V_{k,n} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^{k} V_{j,1} \cdot \sum_{i=1}^{k} U_{i,1} \cdot \Sigma_{1,i} & \cdots & \sum_{j=1}^{k} V_{j,n} \cdot \sum_{i=1}^{k} U_{i,1} \cdot \Sigma_{n,i} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{k} V_{j,1} \cdot \sum_{i=1}^{k} U_{i,m} \cdot \Sigma_{1,k} & \cdots & \sum_{j=1}^{k} V_{j,n} \cdot \sum_{i=1}^{k} U_{i,m} \cdot \Sigma_{n,i} \end{bmatrix}$$

$$\textit{and with} \quad \Sigma_{i,j} = \delta_{ij} \cdot \sigma_i \quad \textit{we get}$$

$$= \begin{bmatrix} \sum_{i=1}^{k} \sigma_i \cdot U_{i,1} \cdot V_{1,i} & \cdots & \sum_{i=1}^{k} \sigma_i \cdot U_{i,1} \cdot V_{k,i} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} \sigma_i \cdot U_{i,k} \cdot V_{1,i} & \cdots & \sum_{i=1}^{k} \sigma_i \cdot U_{i,k} \cdot V_{k,i} \end{bmatrix} = \sum_{i=1}^{k} \sigma_i \cdot \begin{bmatrix} U_{i,1} \cdot V_{1,i} & \cdots & U_{i,1} \cdot V_{k,i} \\ \vdots & \ddots & \vdots \\ U_{i,k} \cdot V_{1,i} & \cdots & U_{i,k} \cdot V_{k,i} \end{bmatrix}$$

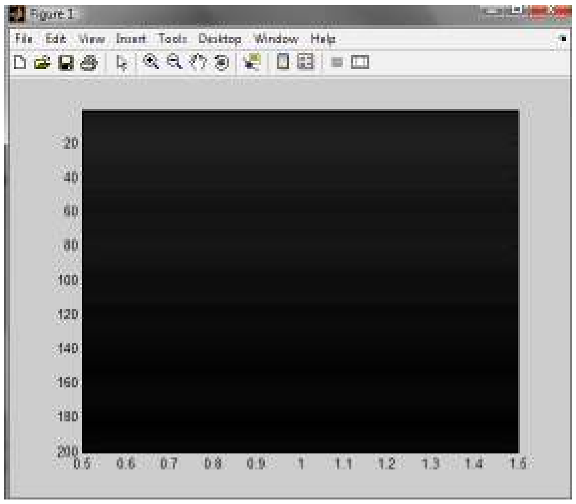
$$= \sum_{i=1}^{k} \sigma_i \cdot U_i \cdot V_i^T$$

Part 1.3

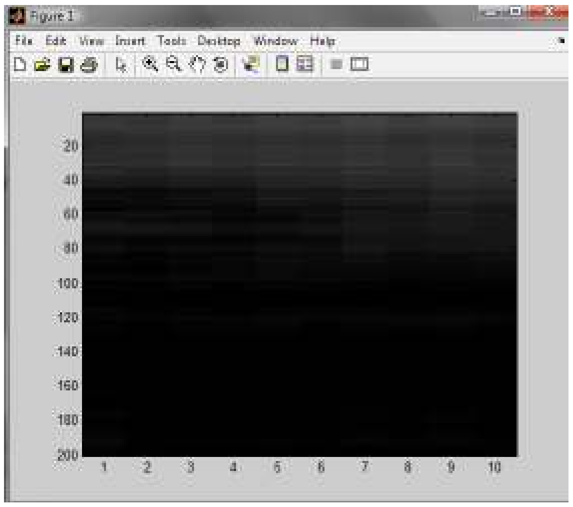
I wrote a little helper function to print the clown images with different k-rank approximations:

```
function [ A ] = clownimage( k )

load clown;
[U,S,V] = svd(X);

A=U(:,1:1:k)*S(1:1:k,:)*V(1:1:k,:)';
image(A); colormap(gray(256));
```

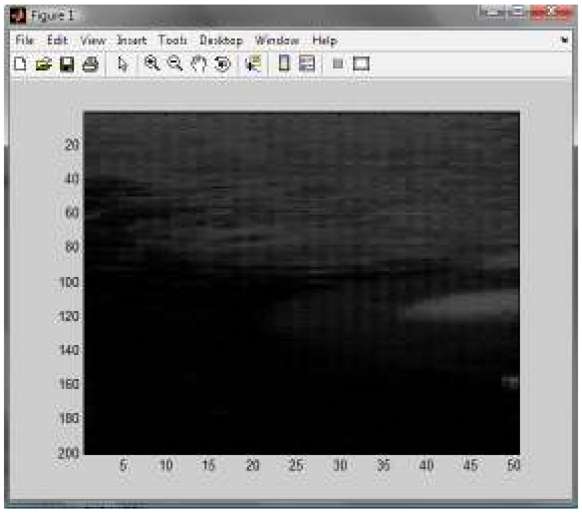
Here are the results:



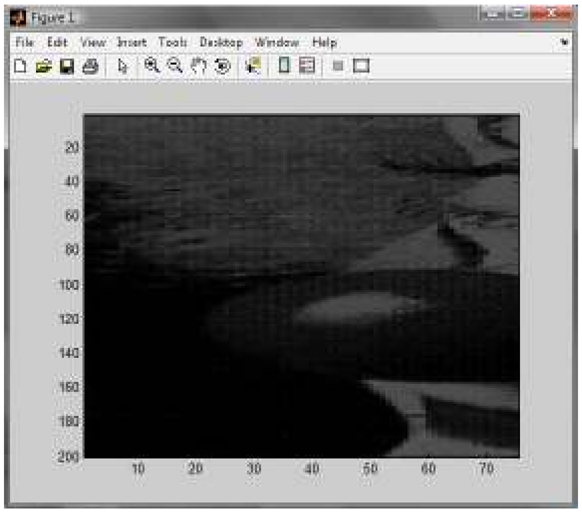
Plot with k=1



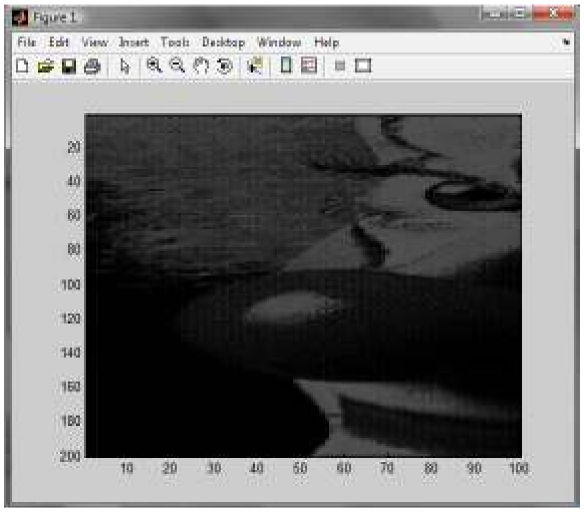
Plot with k=10



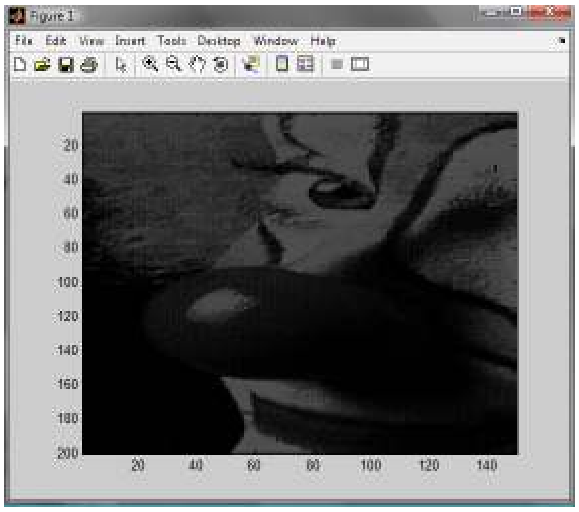
Plot with k=50



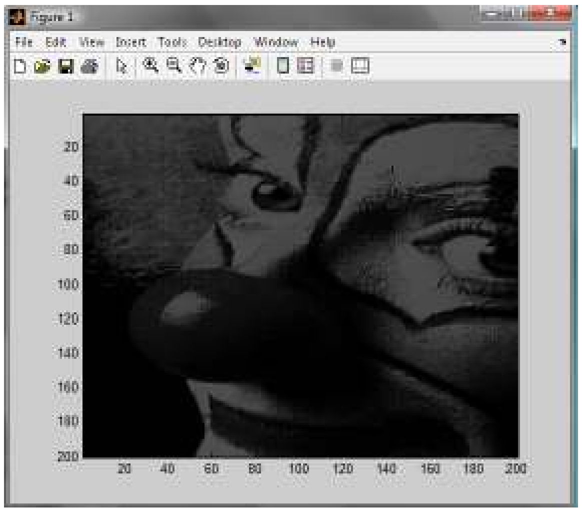
Plot with k=75



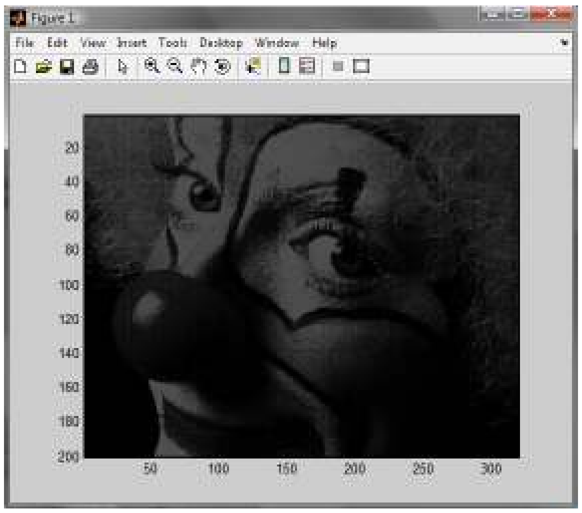
Plot with k=100



Plot with k=150



Plot with k=200

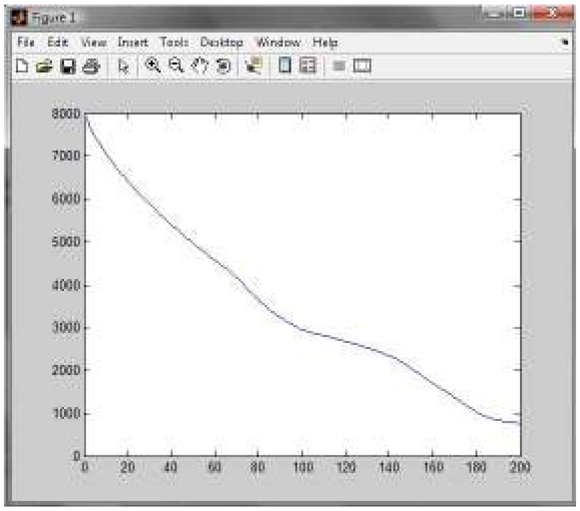


Original plot

The results of the comparison of the quality

$$quality := \left| \|X\|_F - \|A\|_F \right|$$

between the original image and the compressed ones is displayed here:



Plot of the difference of the Frobenius norms with varying k

Obviously, the higher we choose k, the better the quality gets.

Question 2

Part 2.1

Part 2.2

Question 3

Part 3.1

The PDF of a uniform distribution is in our case

$$P(x \mid \theta) = \begin{cases} \frac{1}{(\theta+1)-(\theta-1)} = \frac{1}{2} & for\ \theta-1 \le x \le \theta+1 \\ 0 \quad else \end{cases}$$

Since we have n IID random variables $X = \{x_1, \ldots, x_n\}$ and the PDF of IID random variables is

$$P(X \mid \theta) = \prod_{i=1}^{n} P(x_i \mid \theta)$$

And therefore in our case

$$P(X \mid \theta) = \left(\frac{1}{2}\right)^n = L(\theta \mid X)$$

Part 3.2

Part 3.3

Question 4

Part 4.1

Axioms for an inner product space:

(1) $\langle f, g \rangle = \langle g, f \rangle$

(2) $\langle \lambda \cdot f, g \rangle = \lambda \cdot \langle f, g \rangle$

(3) $\langle f + g, h \rangle = \langle f, h \rangle + \langle g, h \rangle$

(4) $\langle f, f \rangle > 0 \quad \forall f \neq 0$

Proof:

$$(1)\quad \langle f,g\rangle=\int_a^b f(t)\cdot g(t)\,dt=\int_a^b g(t)\cdot f(t)\,dt=\langle g,f\rangle$$

$$(2)\quad \langle \lambda\cdot f,g\rangle=\int_a^b \lambda\cdot f(t)\cdot g(t)\,dt=\lambda\cdot\int_a^b f(t)\cdot g(t)\,dt=\lambda\cdot\langle f,g\rangle$$

$$(3)\quad \langle f+g,h\rangle=\int_a^b [f(t)+g(t)]\cdot h(t)\,dt=\int_a^b f(t)\cdot h(t)+g(t)\cdot h(t)\,dt=\int_a^b f(t)\cdot h(t)\,dt+\int_a^b g(t)\cdot h(t)\,dt$$
$$=\langle f,h\rangle+\langle g,h\rangle$$

$$(4)\quad \langle f,f\rangle=\int_a^b f(t)\cdot f(t)\,dt=\int_a^b f^2(t)\,dt>0\quad \forall f\neq 0\in C_2[a,b]$$

Part 4.2

Axioms for a norm:

$$(1)\quad \|x\|=0\quad if\ x=0$$
$$(2)\quad \|\lambda\cdot x\|=|\lambda|\cdot\|x\|$$
$$(3)\quad \|x+y\|\le\|x\|+\|y\|$$

Proof:

$$(1)\quad \|x\|=\sqrt{\langle x,x\rangle}=\sqrt{0}\quad if\ x=0$$

$$(2)\quad \|\lambda\cdot x\|=\sqrt{\langle \lambda\cdot x,\lambda\cdot x\rangle}=\sqrt{\lambda\cdot\langle x,\lambda\cdot x\rangle}=\sqrt{\lambda\cdot\langle \lambda\cdot x,x\rangle}=\sqrt{\lambda^2\cdot\langle x,x\rangle}=\sqrt{\lambda^2}\cdot\sqrt{\langle x,x\rangle}=|\lambda|\cdot\|x\|$$

$$(3)\quad \|x+y\|^2=\sqrt{\langle x+y,x+y\rangle}^{\,2}=\langle x+y,x+y\rangle=\langle x,x+y\rangle+\langle y,x+y\rangle=\langle x,x\rangle+\langle x,y\rangle+\langle y,x\rangle+\langle y,y\rangle$$
$$=\|x\|^2+\langle x,y\rangle+\langle x,y\rangle+\|y\|^2=\|x\|^2+2\cdot\langle x,y\rangle+\|y\|^2\le\|x\|^2+2\cdot|\langle x,y\rangle|+\|y\|^2\le\|x\|^2+2\cdot\|x\|\cdot\|y\|+\|y\|^2$$
$$=(\|x\|+\|y\|)^2$$

Since the square root is a monotone function we get

$$\|x+y\|^2\le(\|x\|+\|y\|)^2\Rightarrow\sqrt{\|x+y\|^2}\le\sqrt{(\|x\|+\|y\|)^2}\Leftrightarrow\|x+y\|\le\|x\|+\|y\|$$

Unfortunately I found the same idea of proof on Wikipedia. I hope I still get some credit for my solution, even though I cannot prove I didn't simply copy it!
(http://en.wikipedia.org/wiki/Triangle_inequality#Proof)

Part 4.3

Proof:

$$\|x+y\|^2+\|x-y\|^2=\sqrt{\langle x+y,x+y\rangle}^2+\sqrt{\langle x-y,x-y\rangle}^2=\langle x+y,x+y\rangle+\langle x-y,x-y\rangle$$
$$=\langle x,x+y\rangle+\langle y,x+y\rangle+\langle x,x-y\rangle-\langle y,x-y\rangle$$
$$=\langle x,x\rangle+\langle x,y\rangle+\langle y,x\rangle+\langle y,y\rangle+\langle x,x\rangle-\langle x,y\rangle-\langle y,x\rangle+\langle y,y\rangle$$
$$=2\cdot\langle x,x\rangle+2\cdot\langle y,y\rangle+2\cdot\langle x,y\rangle-2\cdot\langle x,y\rangle=2\cdot\langle x,x\rangle+2\cdot\langle y,y\rangle$$
$$=2\cdot\sqrt{\langle x,x\rangle}^2+2\cdot\sqrt{\langle y,y\rangle}^2=2\cdot\|x\|^2+2\cdot\|y\|^2$$

Part 4.4

$$X=[a_1,\dots,a_n],\quad X\ni x=\sum_{i=1}^{n}\alpha_i\cdot a_i,\quad \alpha\in R^n,\quad a_i\ orthonormal$$

Proof:

$$\|x\|^2=\sqrt{\langle x,x\rangle}^2=\langle x,x\rangle=\left\langle\sum_{i=1}^{n}\alpha_i\cdot a_i,\sum_{i=1}^{n}\alpha_i\cdot a_i\right\rangle=\sum_{j=1}^{n}\left\langle\alpha_j\cdot a_j,\sum_{i=1}^{n}\alpha_i\cdot a_i\right\rangle=\sum_{j=1}^{n}\sum_{i=1}^{n}\langle\alpha_j\cdot a_j,\alpha_i\cdot a_i\rangle$$
$$=\sum_{j=1}^{n}\sum_{i=1}^{n}\alpha_j\cdot\alpha_i\cdot\langle a_j,a_i\rangle=\sum_{j=1}^{n}\sum_{i=1}^{n}\delta_{ij}\cdot\alpha_j\cdot\alpha_i=\sum_{i=1}^{n}\alpha_i^2=\sqrt{\langle\alpha,\alpha\rangle}^2=\|\alpha\|^2$$

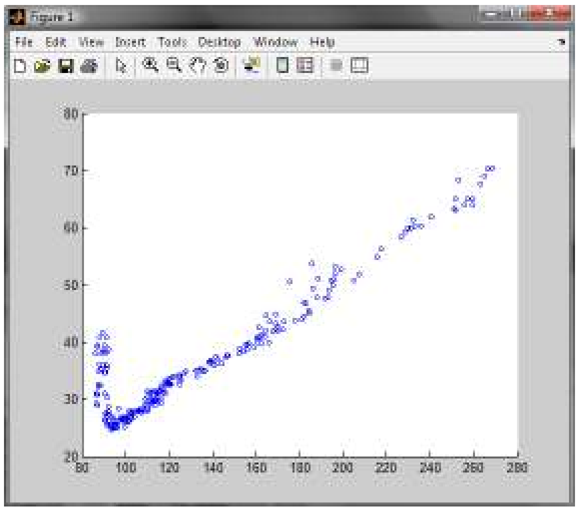
Question 5

Part 5.1

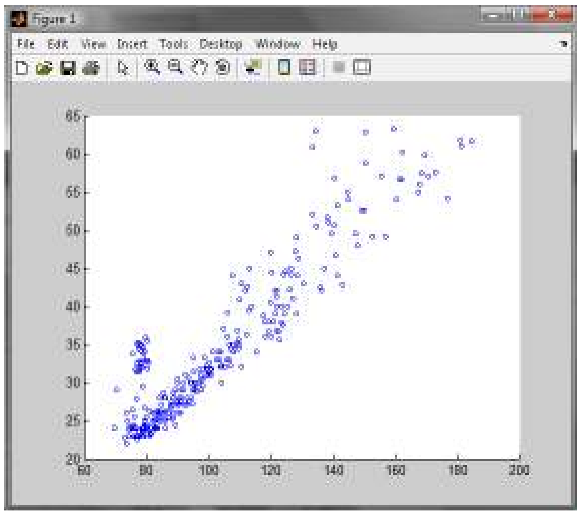
I wrote a little helper function do load the data and display it:

```
function [  ] = displayplot(  )

load g1.dat; load g2.dat; load g3.dat; load g4.dat; load g5.dat;
load g6.dat; load g7.dat; load g8.dat; load g9.dat; load g10.dat;
load g11.dat; load g12.dat; load g13.dat; load g14.dat; load g15.dat;

X = [];

for k=1:3
        X = [X,[g1(:,k);g2(:,k);g3(:,k);g4(:,k);g5(:,k);g6(:,k);...
        g7(:,k);g8(:,k);g9(:,k);g10(:,k);g11(:,k);g12(:,k);...
        g13(:,k);g14(:,k);g15(:,k)]];
end;

scatter(X(:,1),X(:,2),4);
```

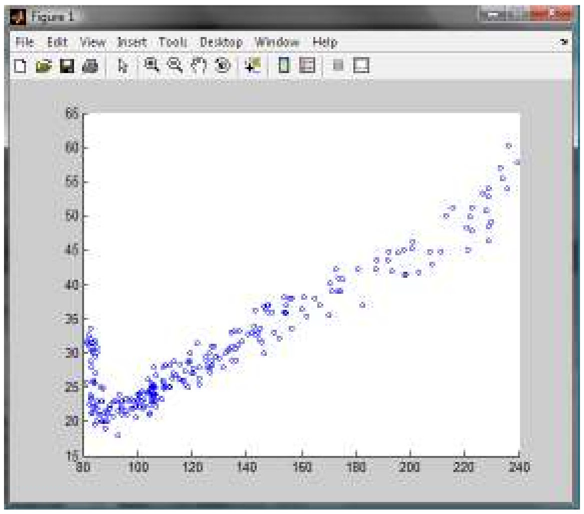
Here are the resulting plots:



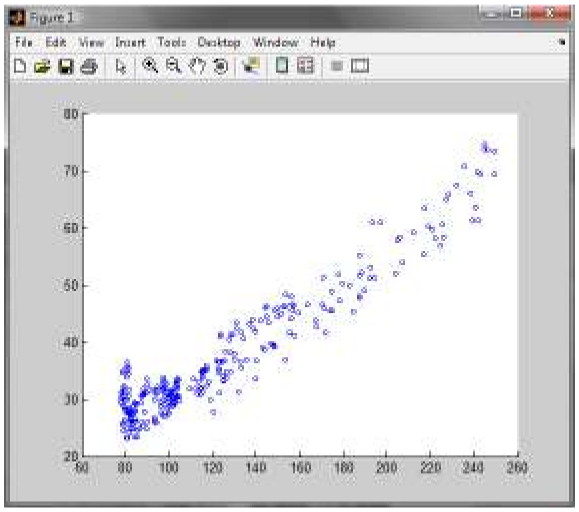
Person a



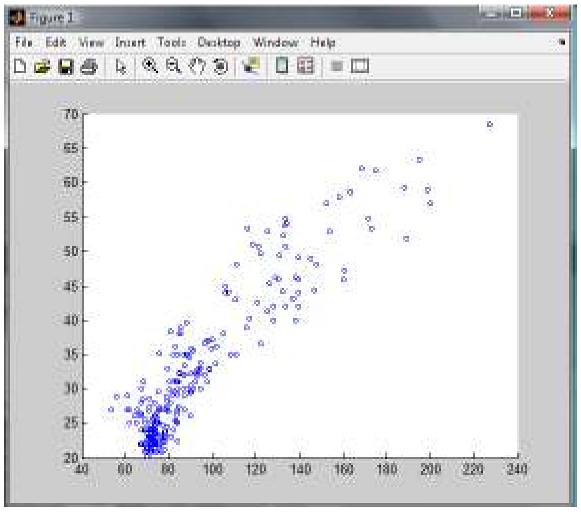
Person b



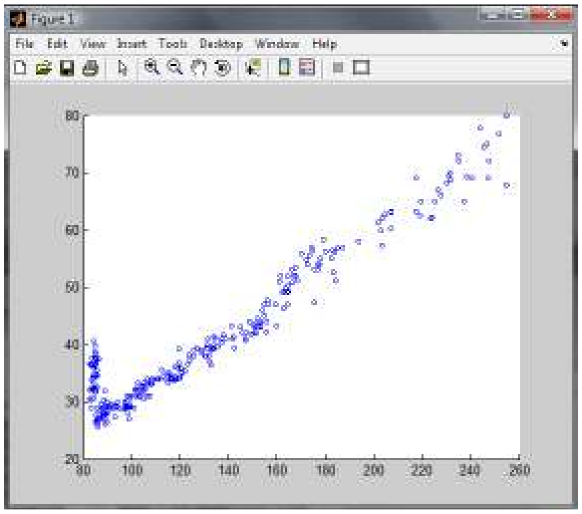
Person c



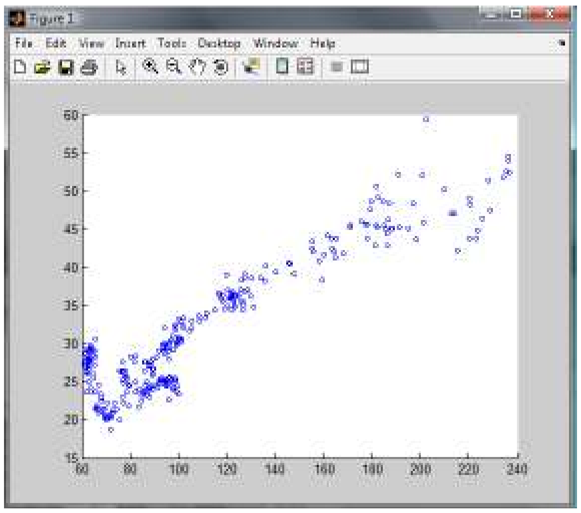
Person d



Person e



Person f



Person g

I did a scatter plot on the first two parameters (height and width) for each person. I would have done a threedimensional scatter plot involving the height/width parameter, but it turned out that without being able to twist and turn the plot you don't recognize that much.

The trajectories of persons c, d and f resemble each other very much. The other trajectories differ more from each other, but are still very much looking the same. It will be very hard to find discriminative boundaries between the trajectories.

A linear function will probably not allow for a good distinction between the different trajectories, but to me it still sounds like the most promising approach.

Part 5.2

I decided to use linear regression, since the plots suggested that a linear function could fit the data probably very well and way better than using k-means.

So our linear function looks like this

$$y = \alpha \cdot x + \beta$$

with parameters α and β. Our goal is to find these parameters in order to minimize the least squares error between our data points and the linear function.

This leads to the following set of equations

$$\left.\begin{array}{c} \beta + \alpha \cdot x_1 = y_1 \\ \beta + \alpha \cdot x_2 = y_2 \\ \vdots \\ \beta + \alpha \cdot x_m = y_m \end{array}\right\} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \cdot \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \text{ or } A \cdot x = y$$

Since this equation cannot be solved exactly as soon as m>2, we multiply the equation with A^T from both sides and get

$$A^T \cdot A \cdot x = A^T \cdot y$$

We can now solve the equation by multiplying both sides with $\left(A^T \cdot A\right)^{-1}$ and get

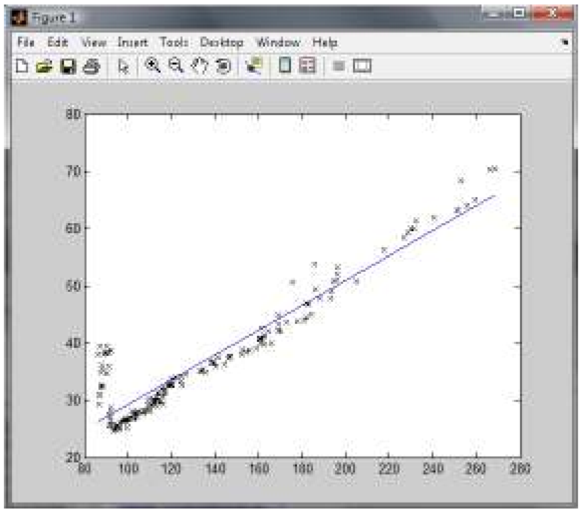
$$x = \left(A^T \cdot A\right)^{-1} \cdot A^T \cdot y$$

I chose the first 10 sets of data points out of the 15 sets to fit the linear curve.

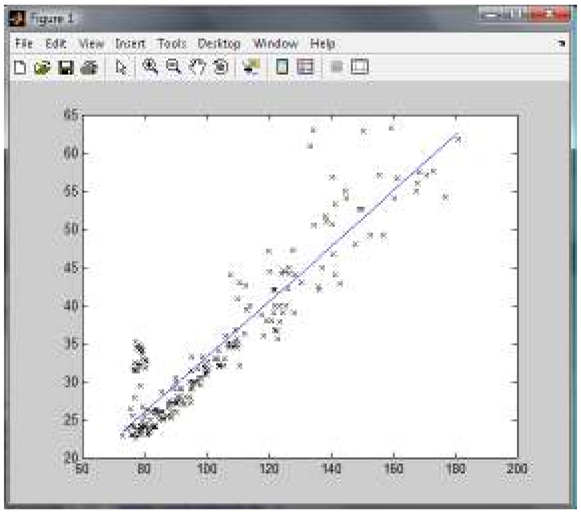
To accomplish this I used Matlab. I wrote another helper function, which solves the linear regression problem and plots the learning data and the fitting curve:

```
function [ out ] = linear_regression(  )

load a1.dat; load a2.dat; load a3.dat; load a4.dat; load a5.dat;
load a6.dat; load a7.dat; load a8.dat; load a9.dat; load a10.dat;

X = [a1(:,1);a2(:,1);a3(:,1);a4(:,1);a5(:,1);a6(:,1);a7(:,1);a8(:,1);a9(:,1);a10(:,1);];
Y = [a1(:,2);a2(:,2);a3(:,2);a4(:,2);a5(:,2);a6(:,2);a7(:,2);a8(:,2);a9(:,2);a10(:,2);];

A = [ones(size(X),1),X];

out = (A'*A)\A'*Y;

input = linspace(min(X),max(X),100);
func = @(x) out(1)+out(2).*x;

plot(X,Y,'kx',input,func(input));
```

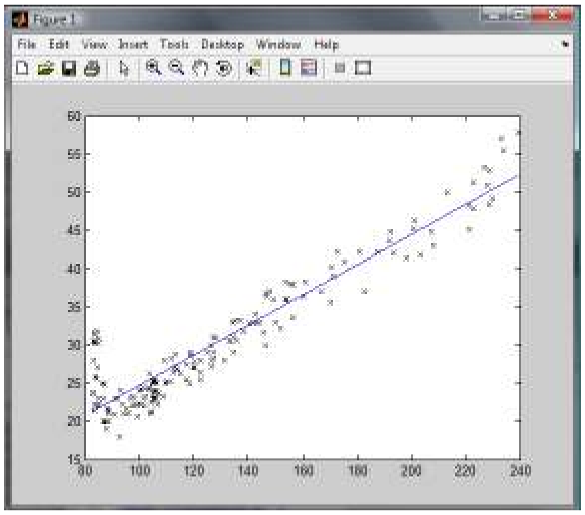
The results:



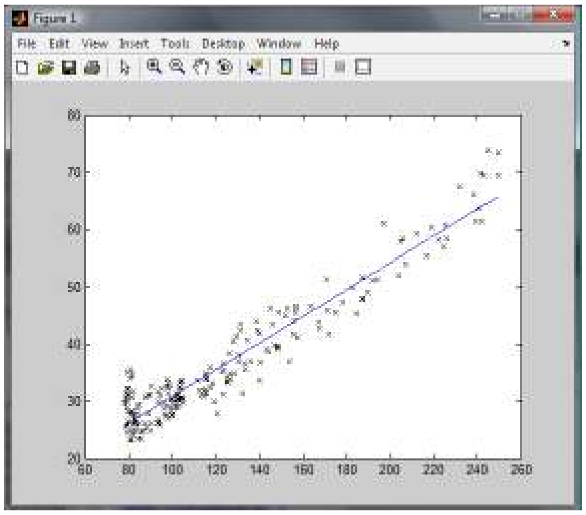
Person a, $\alpha = 0.2175, \beta = 7.3760$



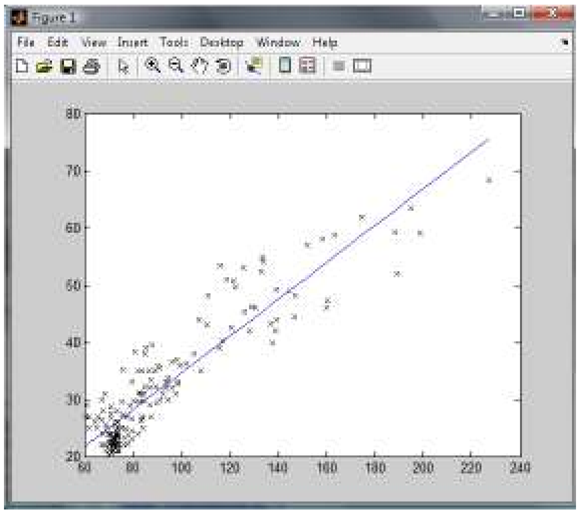
Person b, $\alpha = 0.3632, \beta = -3.2038$



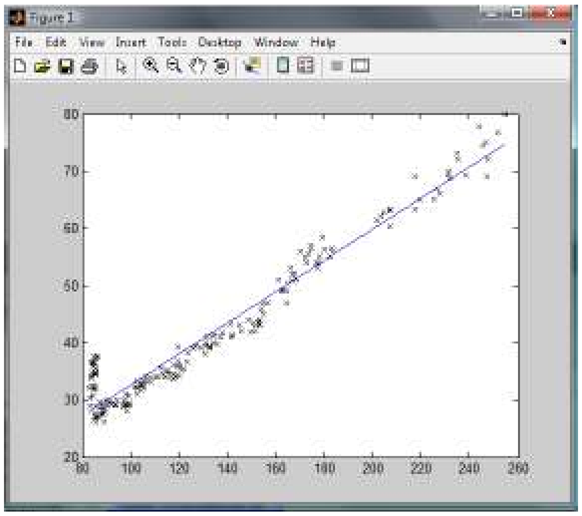
Person c, $\alpha = 0.1975, \beta = 4.9256$



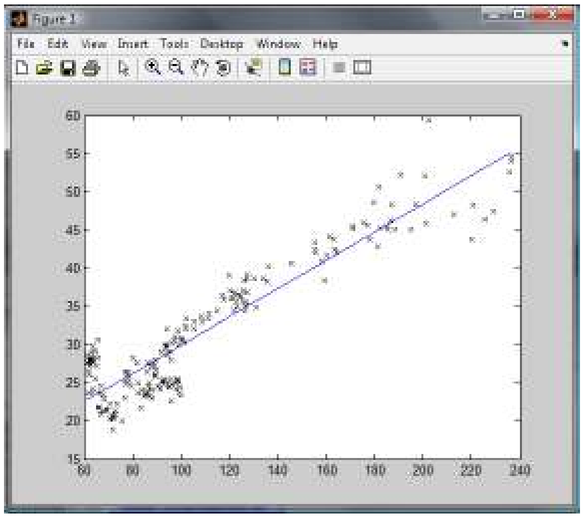
Person d, $\alpha = 0.2332, \beta = 7.4657$



Person e, $\alpha = 0.3222, \beta = 2.3314$



Person f, $\alpha = 0.2724, \beta = 5.2993$



Person g, $\alpha = 0.1850, \beta = 11.3742$

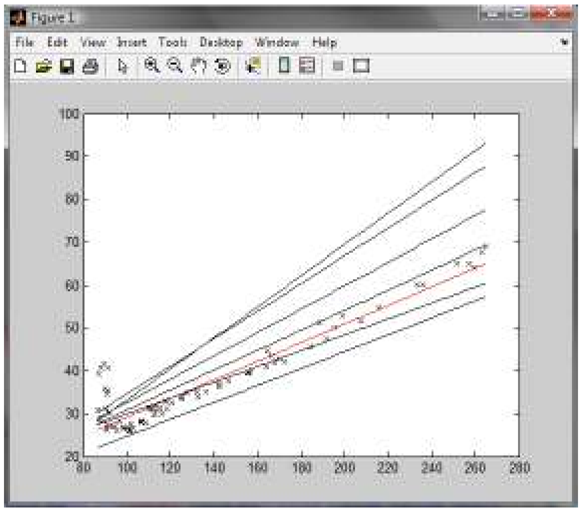
Part 5.3

The next step is to look at the remaining 5 data sets of each person and to check, if the distance to the linear function is minimal when compared to the other linear functions.

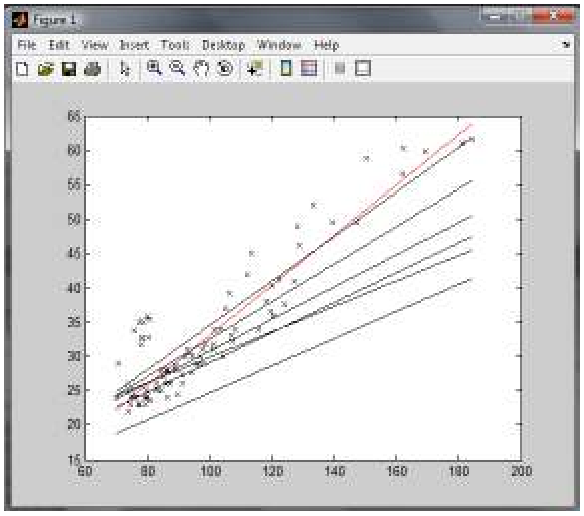
In order to do that I again wrote a little helper function which plots all functions, the function of one person in red, and the associated data points.

```
function [  ] = regression_check( )

load g11.dat; load g12.dat; load g13.dat; load g14.dat; load g15.dat;

X = [g11(:,1);g12(:,1);g13(:,1);g14(:,1);g15(:,1);];
Y = [g11(:,2);g12(:,2);g13(:,2);g14(:,2);g15(:,2);];

input = linspace(min(X),max(X),100);

outa = [7.3760,0.2175];
funca = @(x) outa(1)+outa(2).*x;
outb = [-3.2038,0.3632];
funcb = @(x) outb(1)+outb(2).*x;
outc = [4.9256,0.1975];
funcc = @(x) outc(1)+outc(2).*x;
outd = [7.4657,0.2332];
funcd = @(x) outd(1)+outd(2).*x;
oute = [2.3314,0.3222];
funce = @(x) oute(1)+oute(2).*x;
outf = [5.2993,0.2724];
funcf = @(x) outf(1)+outf(2).*x;
outg = [11.3742,0.1850];
funcg = @(x) outg(1)+outg(2).*x;

plot(X,Y,'kx',input,funca(input),'k',input,funcb(input),'k',input,funcc(input),'k',input,funcd(in
put),'k',input,funce(input),'k',input,funcf(input),'k',input,funcg(input),'r');
```

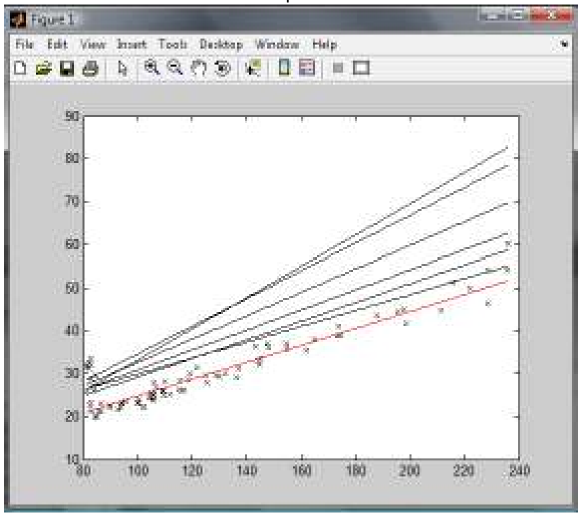
In order to get a feeling for how accurate the data fits the linear functions I plotted all functions and the corresponding data. The function the data should fit best to is always displayed in red.



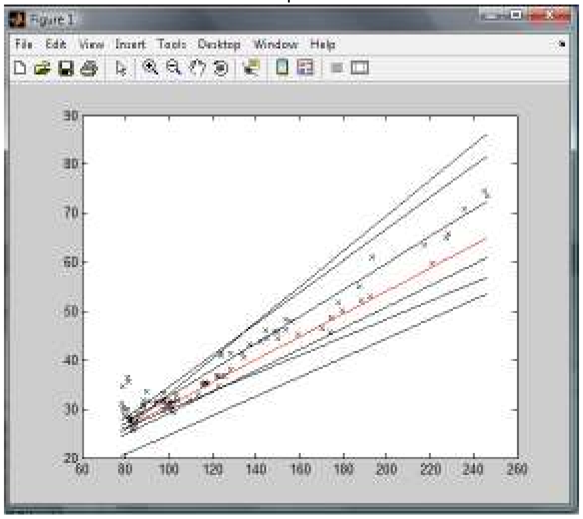
Plot for person a



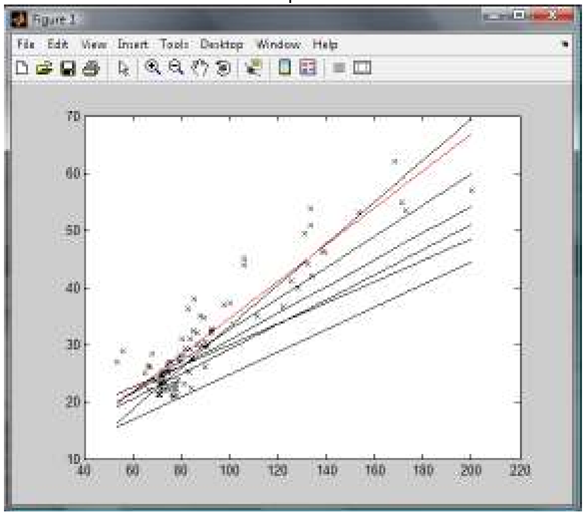
Plot for person b



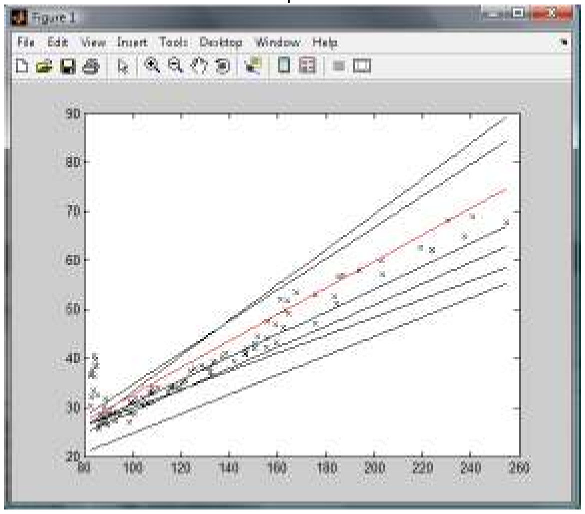
Plot for person c



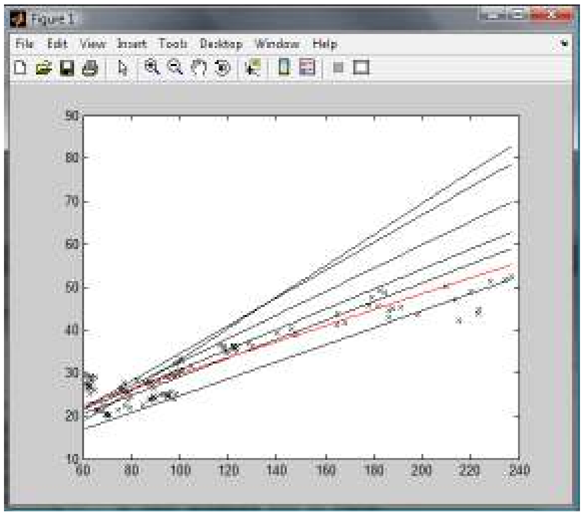
Plot for person d



Plot for person e



Plot for person f



Plot for person g

It turns out that in most cases the data would be assigned to the wrong person since the data points are nearer to other functions than to the one they would be supposed to be. The results are actually very poor. Maybe a polynomial approach would have fit the data a little bit better, but I guess we will learn about better methods.