In our data we have 108 patients and for each patient a feature vector of 171 dimensions. Our goal is to assign weights to each dimension of the feature vectors, so that the distances between feature vectors of patients from the same survival group become small while the distances between two patients, one from each survival group become large.

Here I want to study the feasibility of learning parameter weights for feature vectors by simulating data. I assume that we have 100 patients given (50 patients below and 50 above the median) and for each patient we have a 150 dimensional feature vector. I simulate the feature vectors for the patients as follows: I assume that there are a number of *meaningful* dimensions in the feature space and a number of *non-meaningful* dimensions. For the *meaningful* dimensions, the entries in the feature vectors need to be similar for pairs of patients that both come from the same survival group (below median, above median) and different for pairs of patients, where one patient comes from one group and the other patient from the other group. Assume for example that I want to simulate 30 *meaningful* dimensions: I take two gaussian distributions, one with mean -2 and the other with mean 2 (both with standard deviation 2). For the first group of patients, I sample the first 15 of the *meaningful* dimensions from the first distribution (with mean -2) and the last 15 dimensions from the second distribution (with mean 2). For the second group of patients I sample the 30 dimensions vice versa (first 15 dimensions from the second distribution and last 15 dimensions from the first distribution). For the remaining dimensions (the 120 *non-meaningful* dimensions) I sample from a third distribution with mean 0 and variance 1 for both groups of patients. If I visualize the patient data as before with these feature vectors I get a clear separation between the two groups. To hide the *meaningful* dimensions I divide these dimensions by scalars. (for 30 *meaningful* dimensions, I generate 30 random numbers $r_1, \ldots, r_{30}$ and divide *meaningful* dimension 1 by r_1, dimension 2 by r_2, and so on). See Figure 1, middle for the visualization of the patients after hiding the meaningful dimensions.

Now I try to recover the *meaningful* dimensions by learning the parameter weights. I worked with the matlab functions *fmincon* (very fast) and *fminsearch* (slower) where f is the objective function. For f I used ((mean distance among pairs of patients median)+(mean distance among pairs of patients median))/(mean distance among all pairs with one patient from each group).

See Figure 1 for the visualization of the patient features after the three steps (generating the features, hiding the meaningful dimensions and recovering the meaningful dimensions by learning the weights).

In Figure 1 we see that weights work well for 30 *meaningful* dimensions. Now I assume that my data has only 10 *meaningful* dimensions. Figure 2 shows the visualizaion before and after applying the learned weights. We see that the visualization after applying the weights is not as well separated as

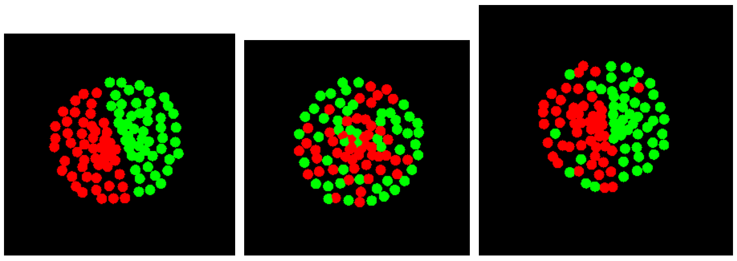

Figure 1: 30 meaningful dimensions. left: visualization before hiding meaningful dimensions. middle: after hiding meaningful dimensions. right: after applying the learned weights.

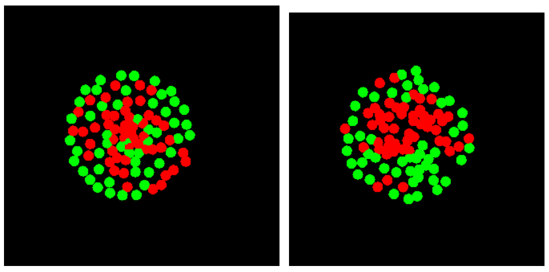

Figure 2: 10 meaningful dimensions. left: before applying weights. right: after applying weights.

for the 30 dimensions but we still get a god clustering. In Figure 3, I choose 0 *meaningful* dimensions and as expected I cant get a good clustering with the learned weights. For a larger number of *meaningful* dimensions the weights work well. Figure 4 shows the clustering for 60 *meaningful* dimensions.

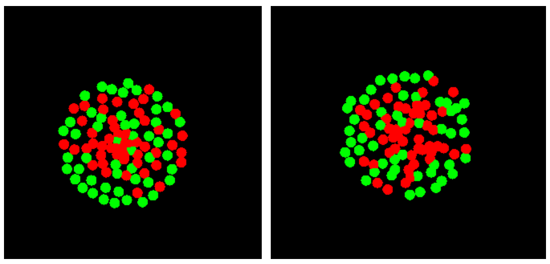

Figure 3: 0 meaningful dimensions. left: before applying weights. right: after applying weights.

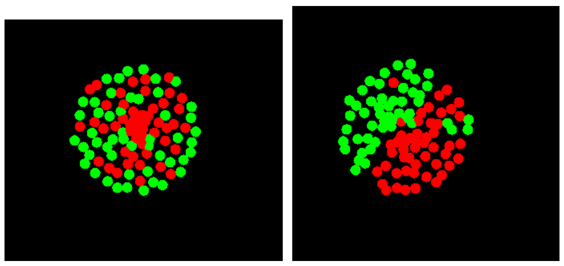

Figure 4: 60 meaningful dimensions. left: before applying weights. right: after applying weights.