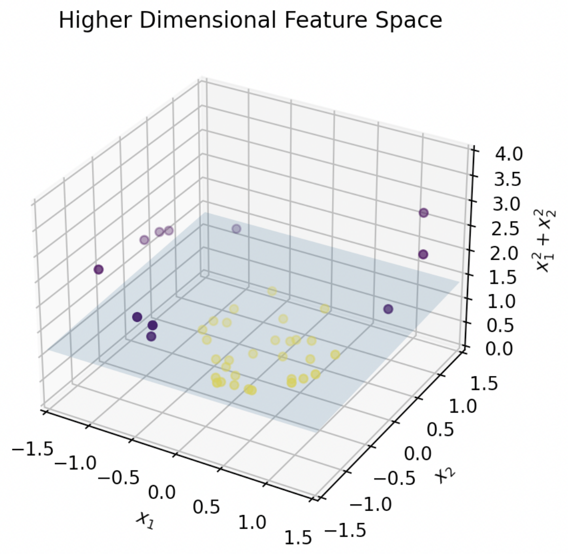
Problem 1

All the plots as follows:

Figure 1: Part 1



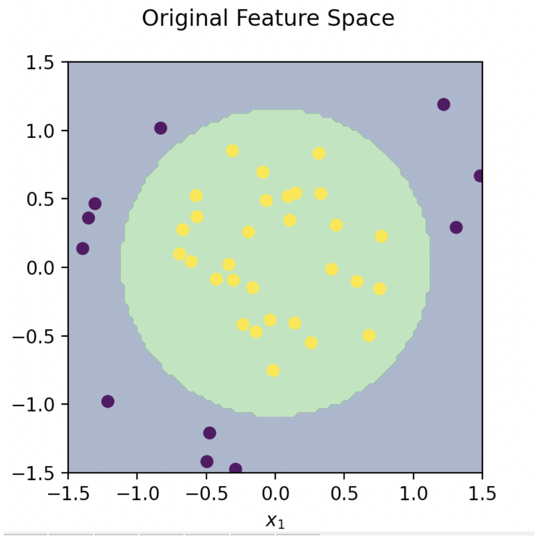
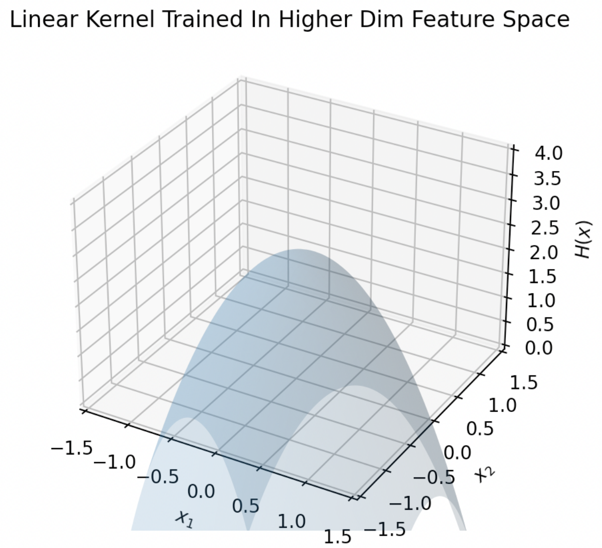


Figure 2: Part 2



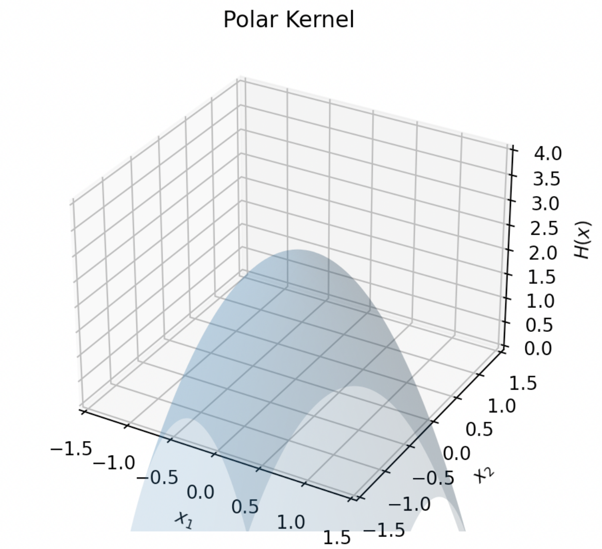


Figure 3: Part 3

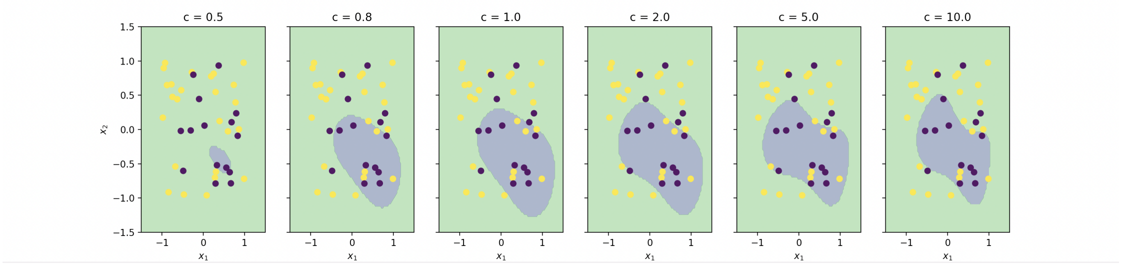
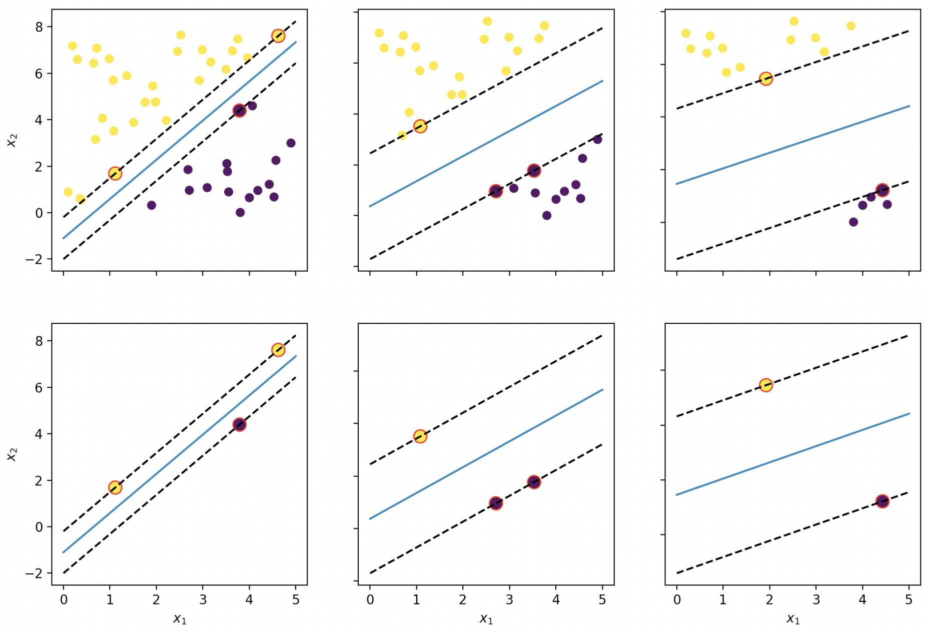


Figure 4: Part 4



Problem 2

In part 1, we saw that the embedding, $\phi((x1, x2)) \rightarrow (x1, x2, x1^2 + x2^2)$, was able to make our non linearly separable data in $\mathbb{R}^2$ perfectly linearly separable in the higher dimensional feature space. Give a non-rigorous explanation using the embedding function and the nature of the data regarding why the polar embedding was able to make this particular dataset perfectly linearly separable. Would this embedding be be able to make any data set linearly separable? Feel free to use the plots generated in part 1 to support your answer.

From the second image we can observe that the data points are perfectly separable by a circle. Therefore, in the new 3-dimensional feature space, the polar embedding function creates a boundary in the form of a cylinder centered at the origin. The third dimension in the new feature space corresponds to the distance of the data point from the origin. By introducing this additional dimension, the polar embedding effectively separates the data points that are closer to the origin from those that are farther away. Since the original data points that belong to different classes are likely to be at different distances from the origin, the cylinder can be oriented to separate the two classes by aligning its axis with the direction of maximum separation between them. While this particular polar embedding was effective in making the specific dataset linearly separable, it may not work for others since it is possible that some datasets have data points that share the same distance to the origin, making it not linear separable by a circle or in 3-dimension as well.

Problem 3

In part 2, we visually confirmed the solution found by the SVM by manually embedding and training a linear SVM vs training a SVM using a corresponding kernel were the same. In lecture we mathematically proved that they are the same. If this is the case, then why would we use a kernel function rather than manually embedding? Give two reasons why this is the case.

There are many reasons. First, using a kernel function allows us to compute the inner products between pairs of data points without explicitly computing the embedded feature space. This can be much faster, especially if the kernel function has a closed-form expression that can be efficiently evaluated. Second, manually embedding the data requires domain knowledge and extra effort to design an effective embedding function. In contrast, kernel functions are often designed to be general-purpose and can be applied to a wide variety of problems. Moreover, using a kernel function allows us to work with complex and non-linear relationships between the input features without explicitly defining a corresponding feature space.

Problem 4

In part 3, what trend do you see regarding the complexity of the decision boundaries as C increases? When would we want to use low C values or high C values?

We can observe that as C increases, the complexity of the decision boundaries also increases. We would want to use a low C value when we want a less complex decision boundary, for instance, when we observe data over-fitting in the original decision boundary and a low testing accuracy, we want to make it less complex to reduce over-fitting. In contrast, We could use a high C value when we want more comnplex decision boundary. For example, when there's a under-fitting in data, causing the training and testing accuracy to be both very low, we can pick a high C value.

Problem 5

In part 4, does the plot of the SVM trained on the entire dataset vs the plot of the SVM trained on only the support vectors match your expectations? Explain why you had those expectations. It's fine if the plots contradicted your expectations.

The plots matches my expectations. My expectation is that the non support vectors does not make contribution to the decision boundary, and only the support vectors determines the position of the decision boundary. The support vectors are the data points that lie on or closest to the decision boundary, the non-support vectors, on the other hand, are farther away from the boundary. Thus, changing the position of a non-support vector outside the margin will not affect the location of the decision boundary or the margin size. The plot corresponds to this hypothesis, since we get exactly the same set of decision boundaries and margins with or without the non support vectors.

Problem 6

In part 4, comment on the margins as we remove support vectors. Explain why this is the case using the optimization problem presented in lecture, and the definition of support vectors.

We can observe that the margin gets larger as we remove support vectors. The reason is that, when we train a SVM, the optimization algorithm seeks to find the decision boundary that maximizes the margin between the support vectors of the two classes. When we remove support vectors and retrain the SVM, we remove the constraints that correspond to those data points and loosen the constraints on the remaining data points. This means that the optimization algorithm is able to find a new decision boundary that maximizes the margin between the remaining support vectors, which are now farther away from the boundary than the original set of support vectors. As a result, the margins tend to get bigger as we remove support vectors and retrain the SVM.