UNIVERSITY OF JEAN MONNET

ADVANCED MACHINE LEARNING

ADVANCED MACHINE LEARNING TP 1

Author:	*Student Number:*
Dimitrios Tsolakidis	16007165
Joseph Renner	16007158

Kernel-PCA

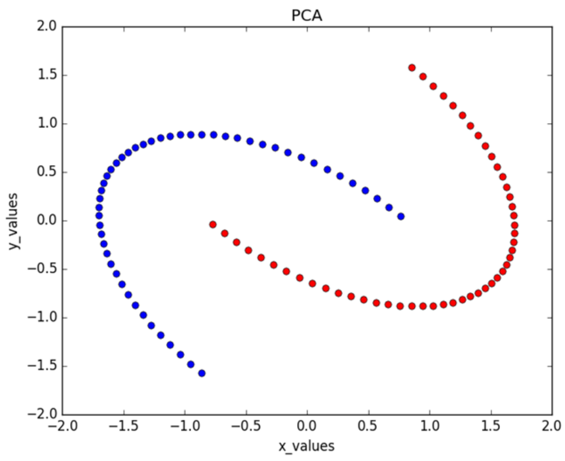
In these experiments, we illustrate the behavior, runtime, and classification pre-processing quality of a classic Principle Component Analysis and that of a kernel-PCA approach on multiple different benchmark datasets.

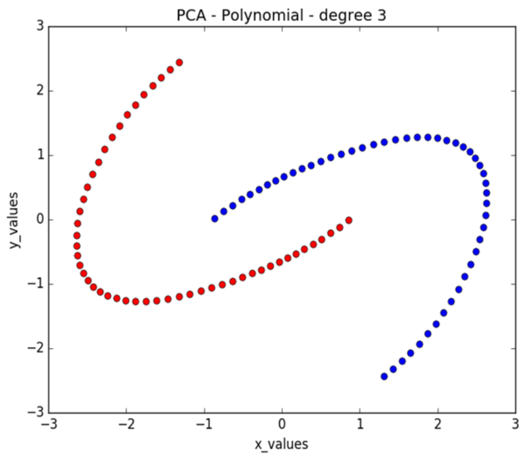
Experimental Setup

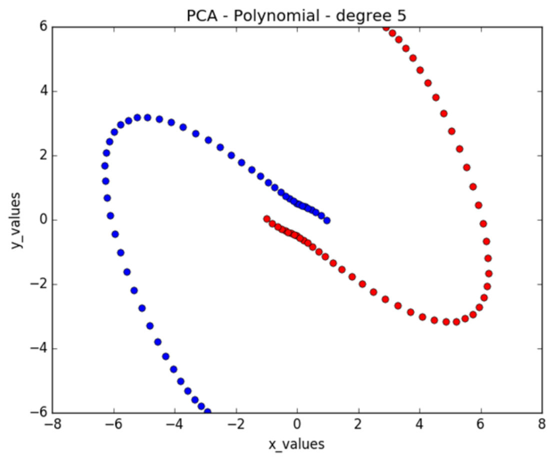
In order to demonstrate the different behaviors, we run a class PCA and a kernel PCA on 4 different benchmark datasets: moons, classification, circles, and swiss roll. Each dataset is generated using scikit learn. For each dataset, we use 2 principle components in order to effectively graph and visualize the results. We also use, for each dataset, 2 different kernels: polynomial kernel (tuning the gamma and degree hyperparameters) and rbf kernel (tuning the gamma parameter). We leave out the linear kernel as this gives the same results as the classic PCA. We will show the resulting visualizations for the PCA and each kernel PCA and parameter setting. Next, we will attempt to classify the resulting projections using a simple linear model to show the quality of the preprocessing for each algorithm, for each dataset. Finally, we will compare the running time for each algorithm, adjusting the number of samples to show how well they scale.

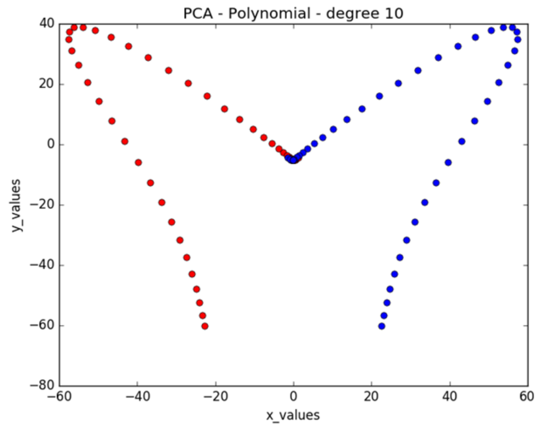
Moons

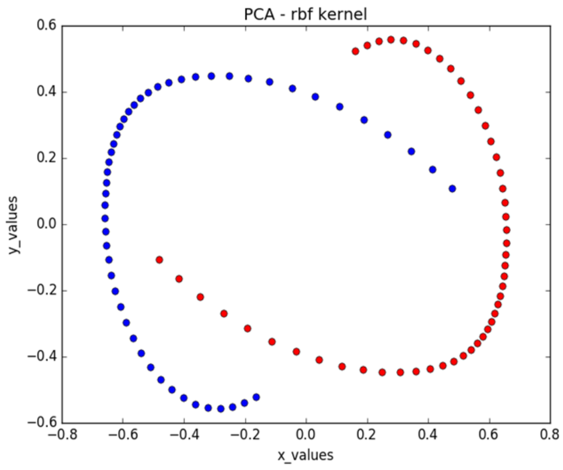
Here we see the resulting projections from the different algorithms on the moon dataset:

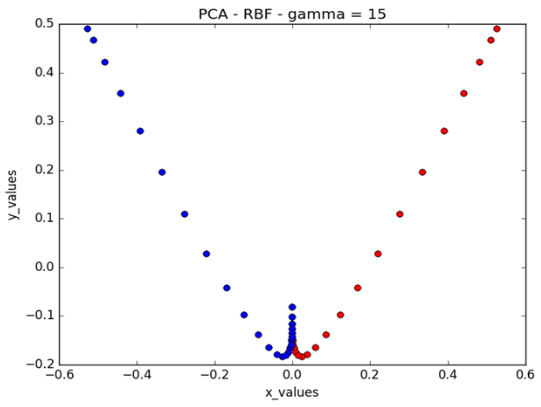










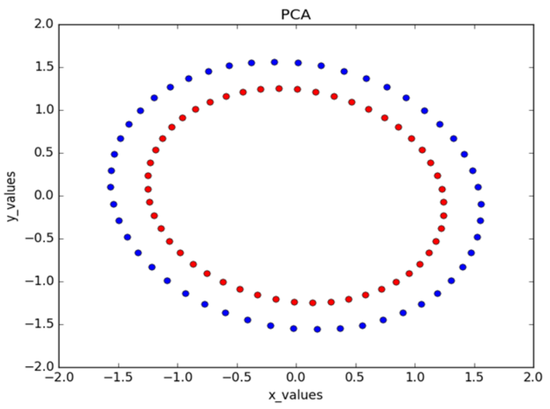


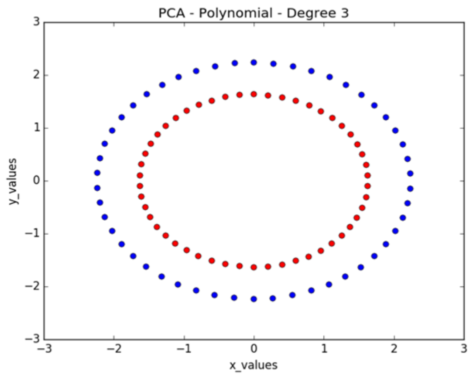
In terms of projecting the data points into a linearly separable space, the polynomial kernel with a high degree seems to do the best job. This can be seen in the accuracy when applying a simple logistic regression model to the resulting projections:

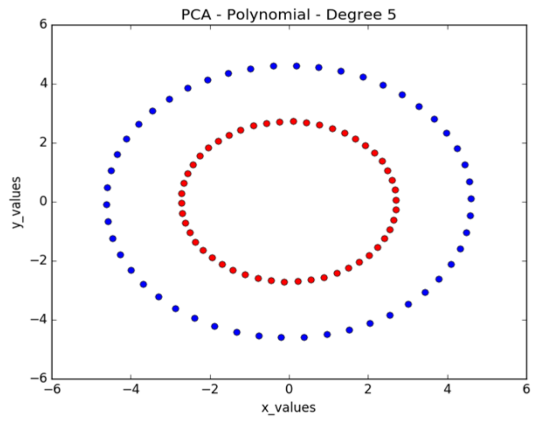
Algorithm	Test Set Accuracy
Classic PCA	0.833
PCA - Polynomial Kernel - Degree 3	0.7
PCA - Polynomial Kernel - Degree 5	0.866
PCA - Polynomial Kernel - Degree 10	0.933
PCA - RBF Kernel - Gamma 0.5	0.6
PCA - RBF Kernel - Gamma 15	0.7

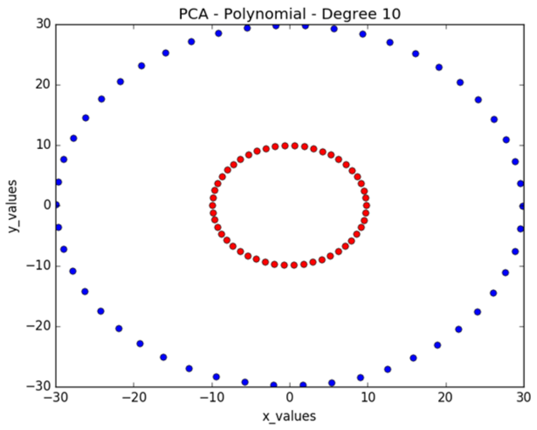
Circles

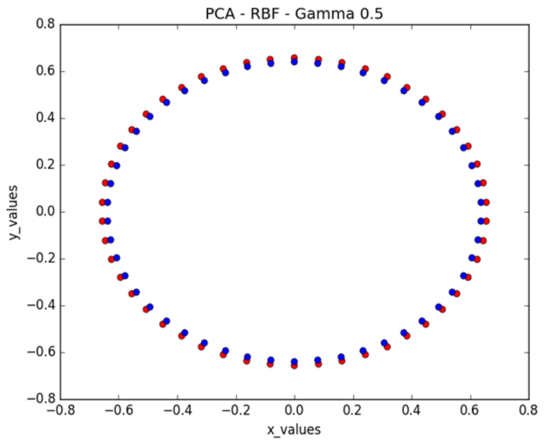
Here we see the resulting projections from the different algorithms for the circles dataset:

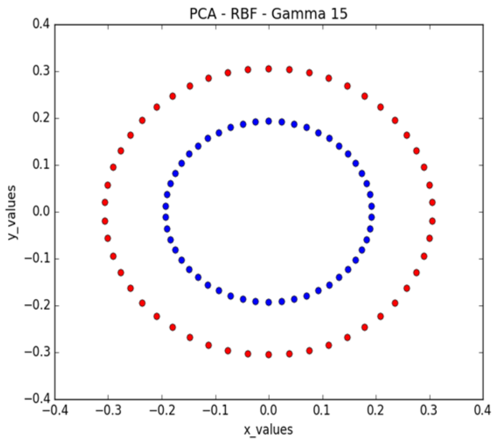










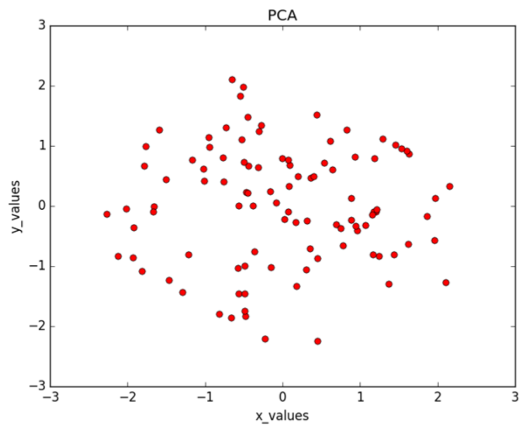


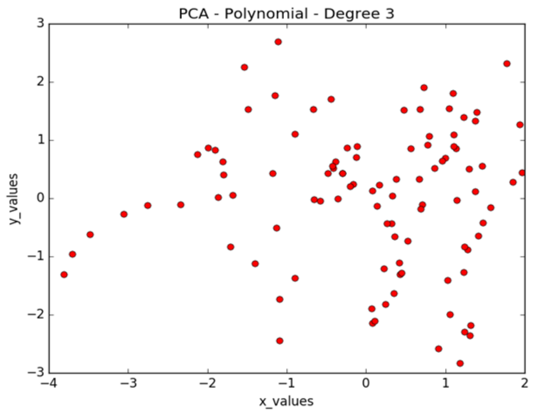
None of the algorithms seem to project the points into a linearly separable space, contrary the expected behavior. The linear classifier accuracies are shown below:

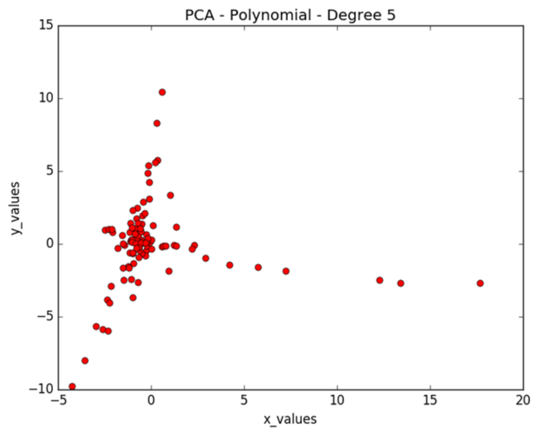
Algorithm	Test Set Accuracy
Classic PCA	0.533
PCA - Polynomial Kernel - Degree 3	0.433
PCA - Polynomial Kernel - Degree 5	0.566
PCA - Polynomial Kernel - Degree 10	0.433
PCA - RBF Kernel - Gamma 0.5	0.366
PCA - RBF Kernel - Gamma 15	0.566

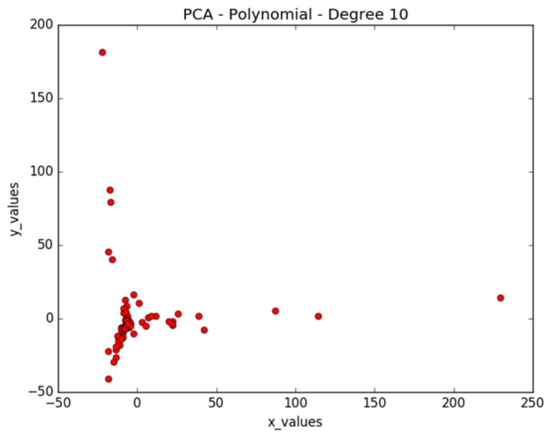
Swiss Roll

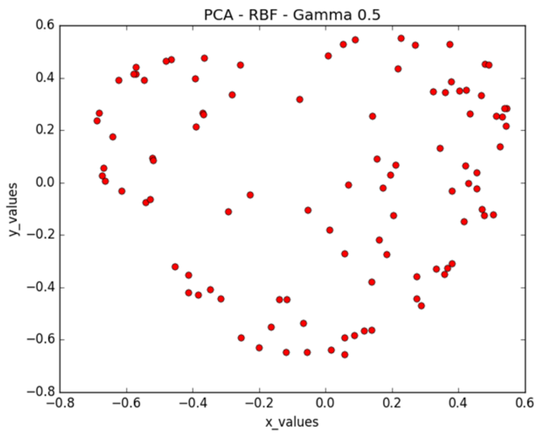
Here we see the resulting projections from the algorithms on a swiss roll dataset:

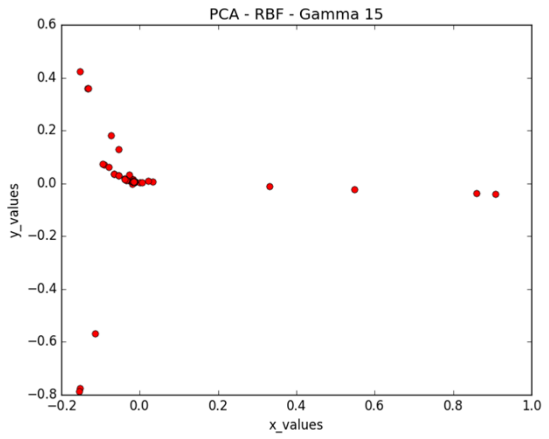






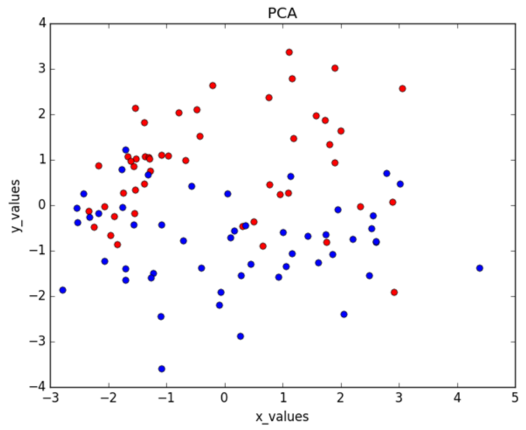


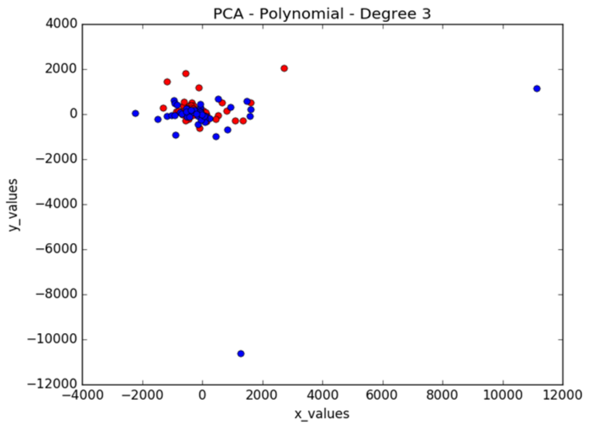


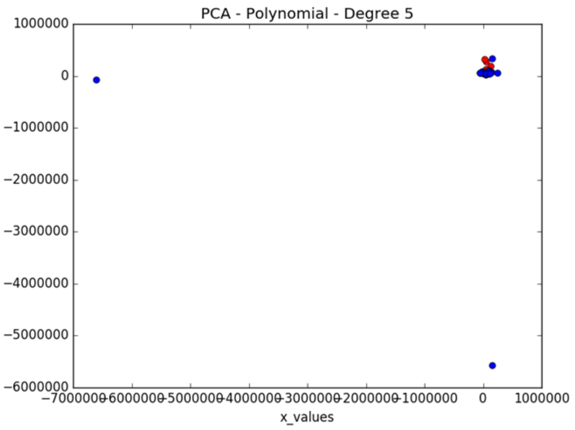


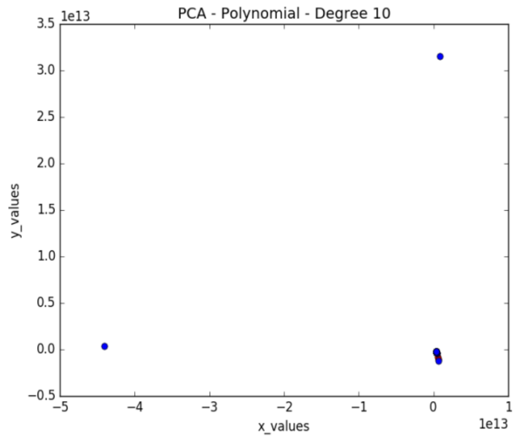
Classification

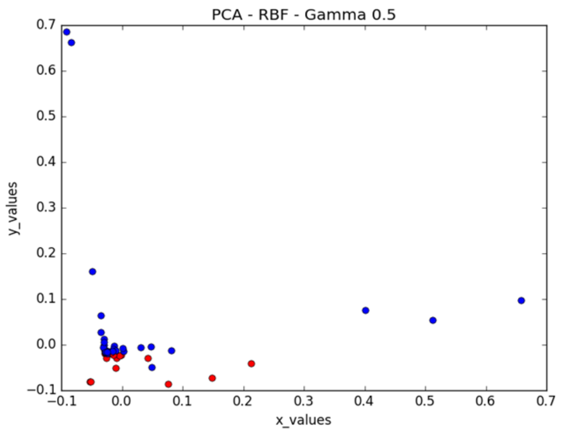
Here we see the resulting projections for the different algorithms on a classification dataset:

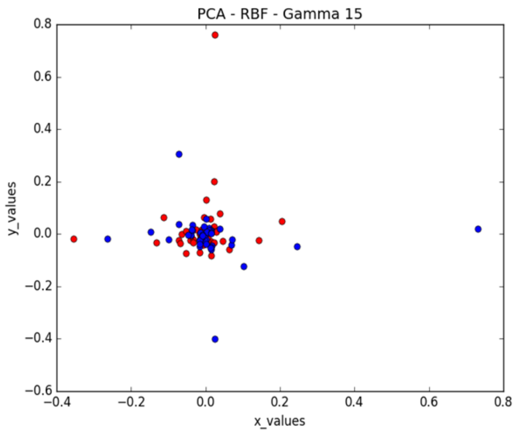












An RBF kernel with gamma=0.5 seems to do the best job of linearly separating the data points, and this is reflected in the accuracy scores of the classifier:

Algorithm	Test Set Accuracy
Classic PCA	0.666
PCA - Polynomial Kernel - Degree 3	0.633
PCA - Polynomial Kernel - Degree 5	0.466
PCA - Polynomial Kernel - Degree 10	0.566
PCA - RBF Kernel - Gamma 0.5	0.7
PCA - RBF Kernel - Gamma 15	0.4

Time and Scalability

We compare the time of the algorithms with different number of data points in the dataset (we use the moon dataset for this comparison):

Algorithm	Number of data points	Time (in seconds)
Classic PCA	100	0.002
PCA - Polynomial Kernel - Degree 3	100	0.039
PCA - RBF Kernel	100	0.036
Classic PCA	1000	0.005
PCA - Polynomial Kernel - Degree 3	1000	3.882
PCA - RBF Kernel	1000	4.119
Classic PCA	10000	0.012
PCA - Polynomial Kernel - Degree 3	10000	>30
PCA - RBF Kernel	10000	>30

As shown above, the kernel PCA methods do not scale well to larger number of data points. This most likely comes from the added computation of the gram matrix and projecting the data points to the new space, since for each new projection many passes over the other data points are needed.

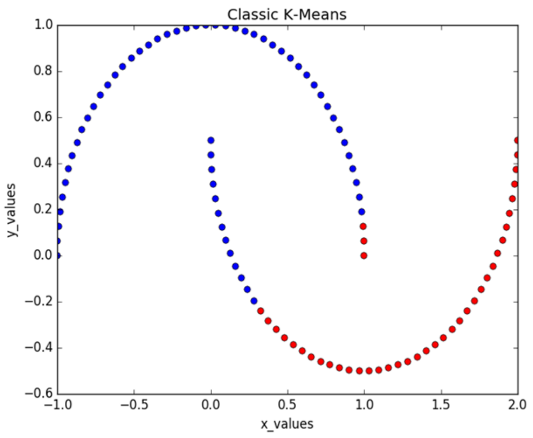
Kernel K-means

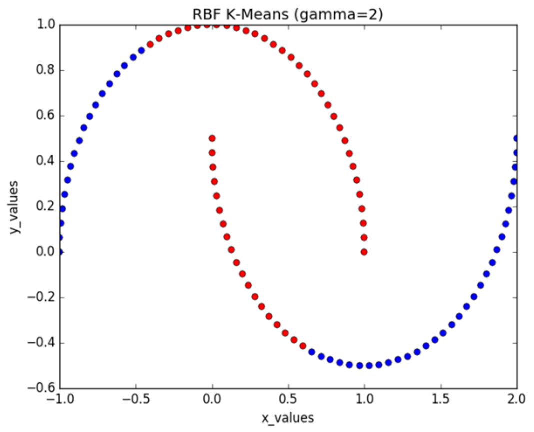
We next compare the behavior of the classic k-means algorithm and the kernel k-means approach

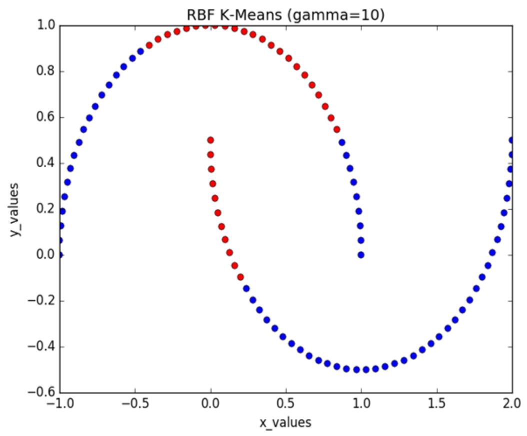
Experimental Setup

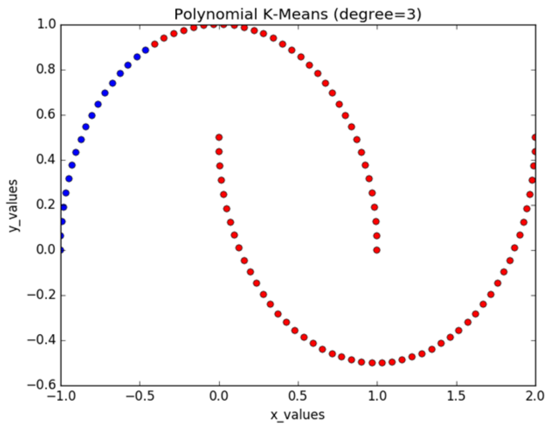
When comparing the two approaches, we use two different datasets that are easy to visualize: moons and circles. When run the two different algorithms and visualize the results. For the kernel k-means, we use an rbf kernel (varying gamma parameter) and a polynomial kernel (varying degree). So, for each algorithm, kernel, hyperparameter setting, and dataset we visualize the results and compare the runtimes of the algorithm. The value of k is set to 2 for each experiment, to conform with the datasets.

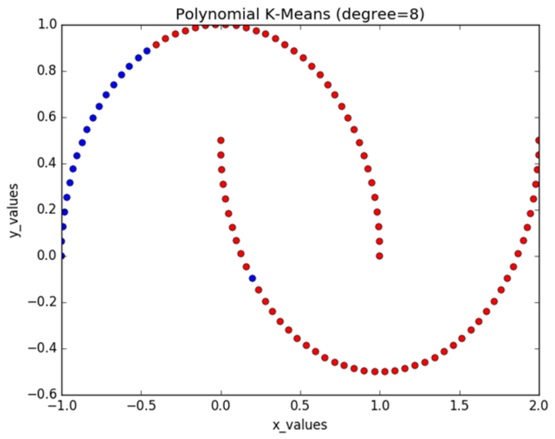
Moons Dataset



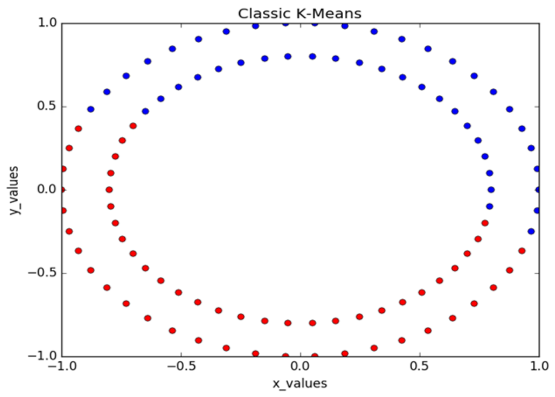


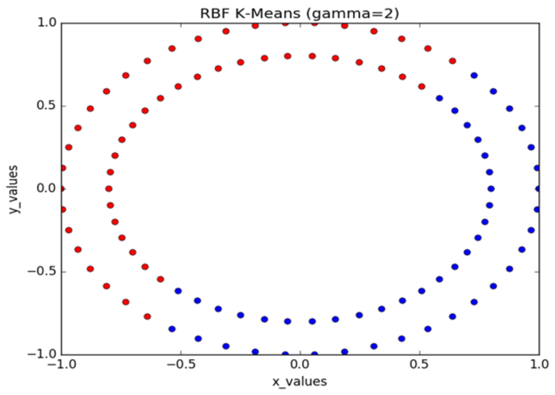


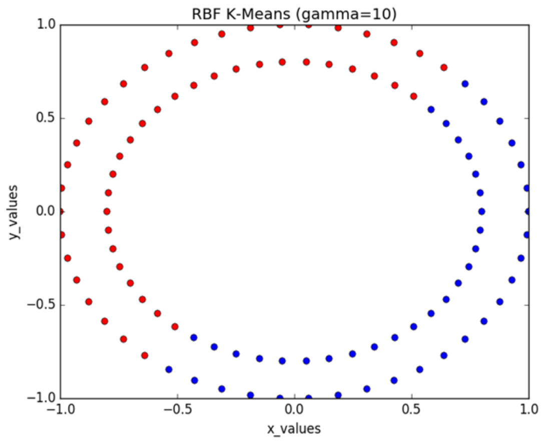


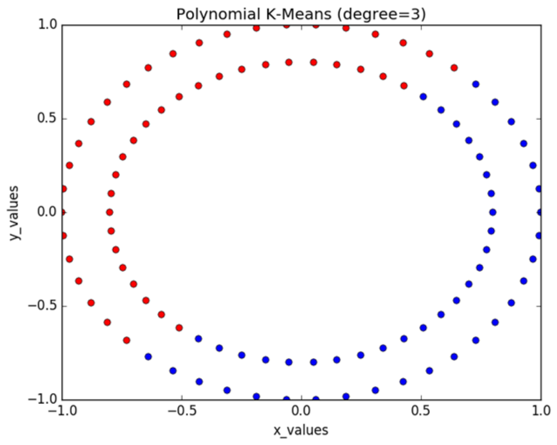


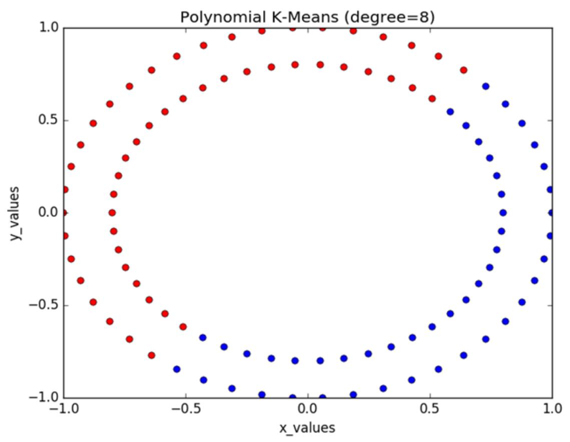
Circles Dataset











These results are somewhat unexpected in that the RBF kernel did not enhance the clustering of the circles, contrary to expectations.

Scalability

We test the scalability of the algorithms by timing their execution with varying number of examples to cluster.

Algorithm	Number of Examples	Time (In Seconds)
Classic K-Means	100	0.001
RBF Kernel K-Means	100	0.012
Polynomial Kernel K-Means	100	0.012
Classic K-Means	1000	0.01
RBF Kernel K-Means	1000	1.321
Polynomial Kernel K-Means	1000	1.33
Classic K-Means	10000	>30
RBF Kernel K-Means	10000	>30
Polynomial Kernel K-Means	10000	>30

Logistic Regression

Recall:

- Logistic Regression is a regression model where the outcome is categorical
- The outcome can be interpreted as a probability of belonging to a certain class
- Mathematically: $F(x) = \frac{1}{1+e^{-(w^T x)}}$, where *F(x)* is the prediction, *x* is the input feature vector, and *w* is the parameters of the model
- We can say that $w^T x$ is a **linear kernel**, and as such, logistic regression is a linear classifier
- Taking the Maximum-a-Posteriori Solution gives us:

$$w^* = \arg\min_w \sum_i^n \log\left(1 + e^{-y_i(w^T x_i)}\right) + \frac{\lambda}{2} w^T w$$

  which can be solved using gradient descent on w.

Kernelized:

- if we take the MAP solution above, take the gradient w.r.t. *w*, then set the gradient to 0, we can obtain:

$$w^* = \lambda^2 \sum_i^n y_i x_i (1 - F(x)) = \sum_i^n \alpha_i x_i$$

  where:

$$\alpha_i = \lambda^2 y_i (1 - F(x))$$

- Thus we can add a kernel and define the weights in terms of support vectors:

$$w = \sum_i^n \alpha_i \phi(x_i)$$

- This brings *F(x)* to:

$$F(x) = \frac{1}{1 + e^{\sum_i^n \alpha_i \phi(x_i)\phi(x)}} = \frac{1}{1 + e^{\sum_i^n \alpha_i K(x_i, x)}}$$

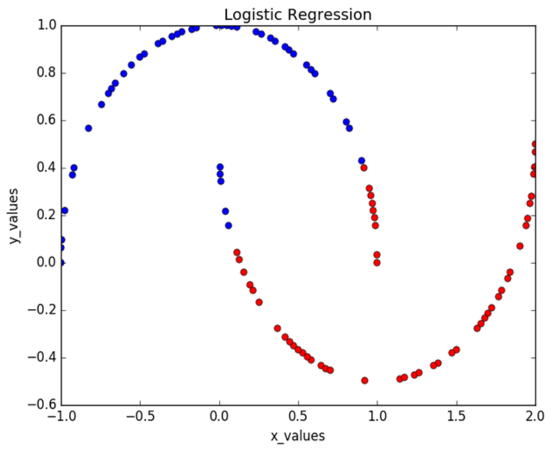
  Thus proving that logistic regression can be kernelized

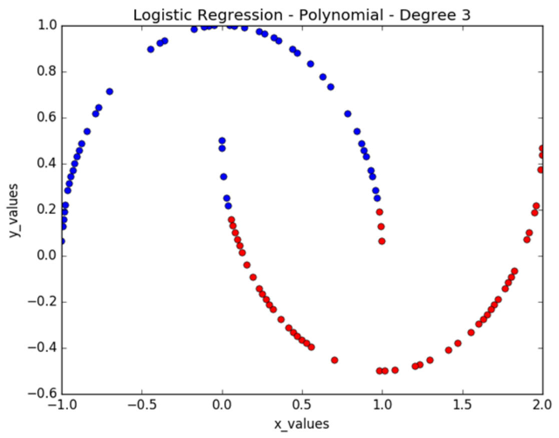
Experimental Study

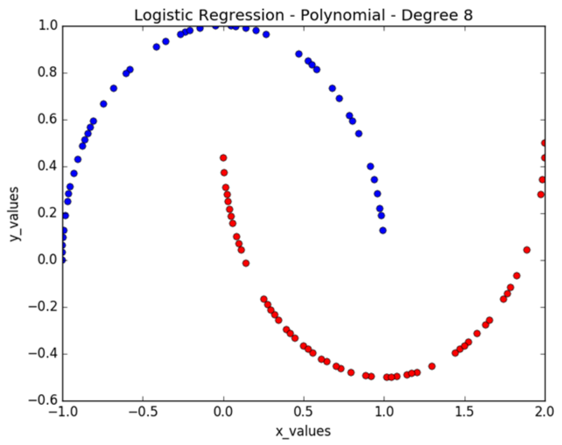
We test the behavior of the classic logistic regression and kernel logistic regression in a series of experiments. We generate moon and circle datasets using sklearn's datasets module. For each dataset, we run the algorithms and obtain the classification accuracy, and visualize the results. We use a polynomial kernel with different degree values, and an rbf kernel with different gamma values. Finally, we obtain the run time of each algorithm on a classification dataset with differing number of samples.

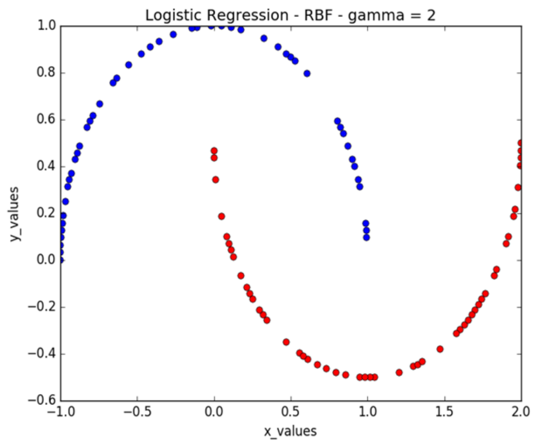
Circles Dataset

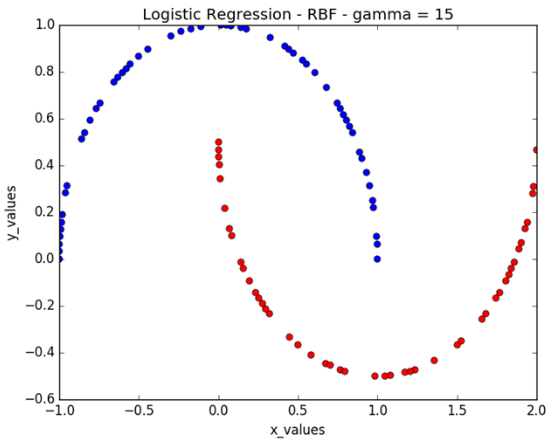
Algorithm	Accuracy
Logistic Regression	0.86
Polynomial kernel (degree=3) Log Reg	0.92
Polynomial kernel (degree=8) Log Reg	1.0
RBF kernel (gamma=2) Log Reg	1.0
RBF kernel (gamma=15) Log Reg	1.0







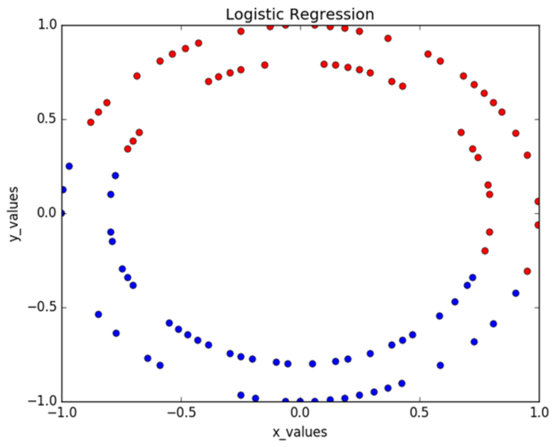


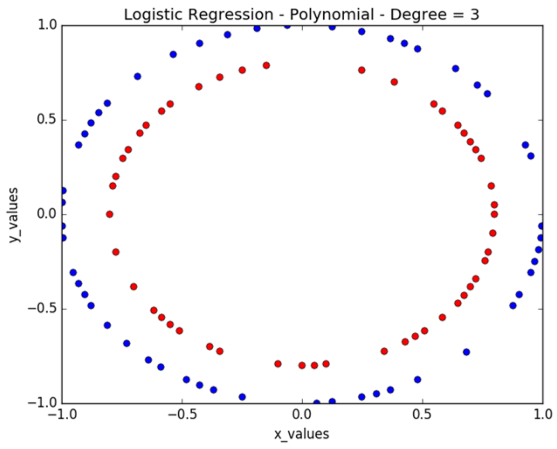


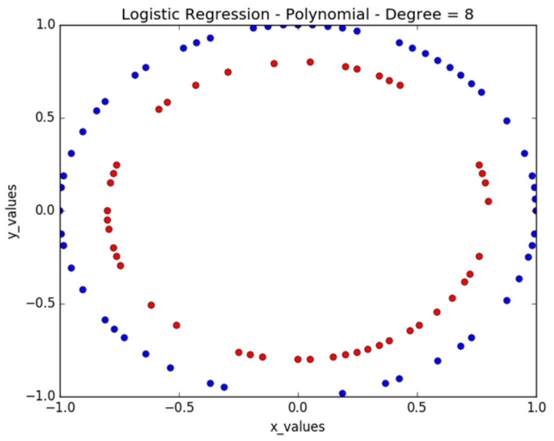
The accuracies and visualizations above show that the kernel logistic regression models do a better job of classifying the non-linearly separable moons dataset.

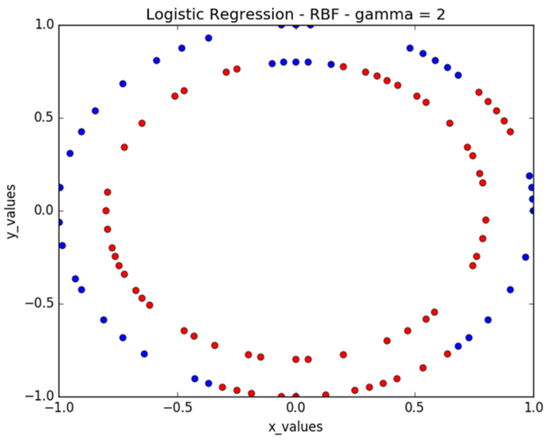
Circles Dataset

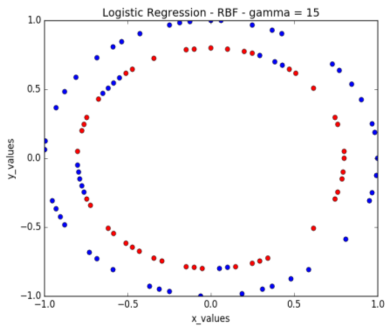
Algorithm	Accuracy
Logistic Regression	0.44
Polynomial kernel (degree=3) Log Reg	1.0
Polynomial kernel (degree=8) Log Reg	1.0
RBF kernel (gamma=2) Log Reg	0.78
RBF kernel (gamma=15) Log Reg	0.86











The above accuracies and visualizations show that the kernel logistic regression, especially with a polynomial kernel, work better than the classic logistic regression model for classifying the circles dataset.

Scalability

The run times for fitting a model and predicting on new data are shown in the below table. Note that for the kernel methods, fitting and predicting both require computing a gram matrix. For fitting, it is the training data against the training data; for predicting, it is the test data against the training data.

Algorithm	Number of Samples (train and test, each)	Fit time (in seconds)	Predict time (in seconds)
Log Reg	100	0.001	0.001
Log Reg - Polynomial	100	0.007	0.001
Log Reg - RBF	100	0.004	0.001
Log Reg	1000	0.001	0.001
Log Reg - Polynomial	1000	0.404	0.129
Log Reg - RBF	1000	0.316	0.034

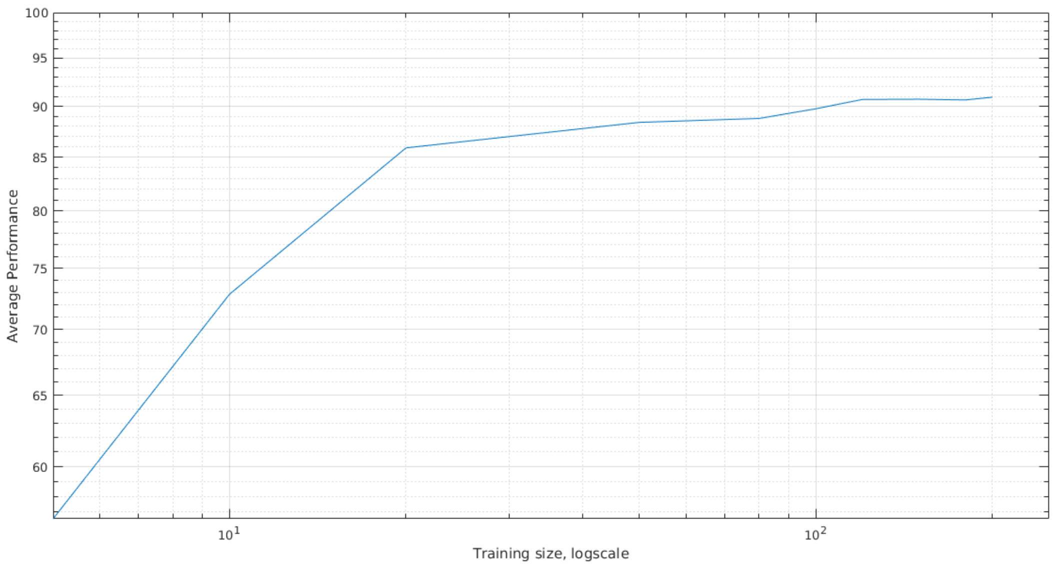
Classic logistic regression seems to scale better than the kernel variants, which is to be expected.

ONE-CLASS SVDD

In this section we conduct experiments using one-class SVDD. We validate the classifier using the Breast Cancer dataset, which is normally used for binary classification. For the experiment we select only positive examples at the training stage, and the resulting models are evaluated on a separated test set, composed of both positive and negative examples. Moreover, the size of the training set varies ([5 10 20 50 80 100 120 150 180 200]) in order to observe how many examples are required to be able to obtain a good performance. We also make use of pre-computed RBF and Polynomial kernels and then we compare the results.

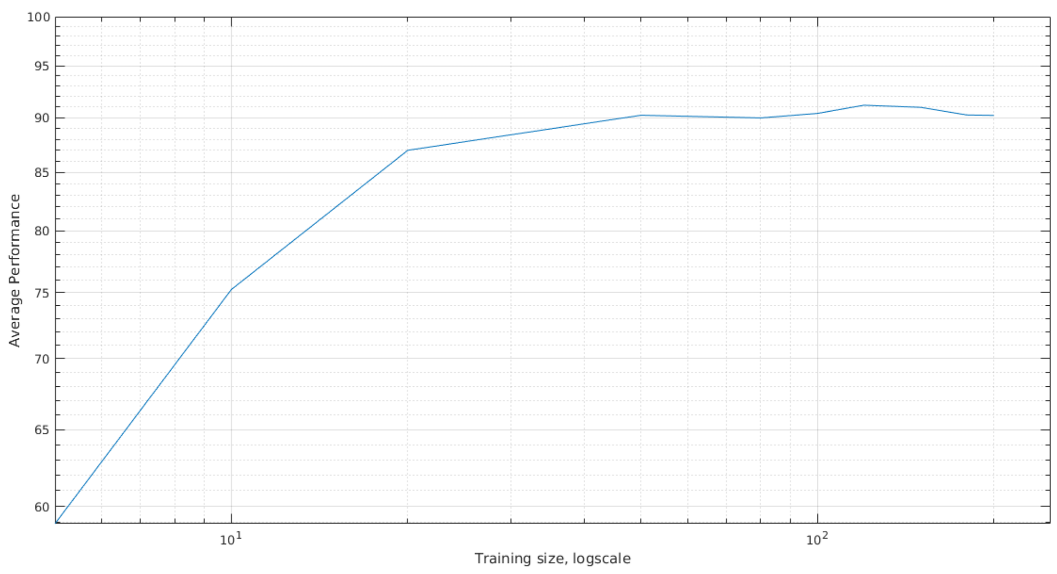
For each training set *gamma* and *C* are cross-validated. We repeat the process 10 times and we give the results on average. To allow some errors during the training phase, the hyper-parameter *C* must be greater than 0 but lower or equal than one. When *C* > 1 we are in the hard margin case.

For the RBF kernel we set the gamma values in the range [2^{-5}, …, 2^{7}] and we set the polynomial kernel up to 4th degree. We present the results obtained with the best *C* and best *gamma* value.



Average performance using different training sets with the best C and *gamma value for RBF kernel (of each training set).*

We observe when our training data consists of around 100 or more observations, there is not any significant difference in performance.



Average performance using different training sets with the best C and polynomial degree

Similar, when using polynomial kernels we observe that when our training set consist of 100 or more observations, there is not any significant difference in the average performance.

Moreover, we constructed our own 2D dataset in order to be able to visualize the performance of one-class SVDD using RBF kernel with different ranges. The *C* value is fixed to 0.1.

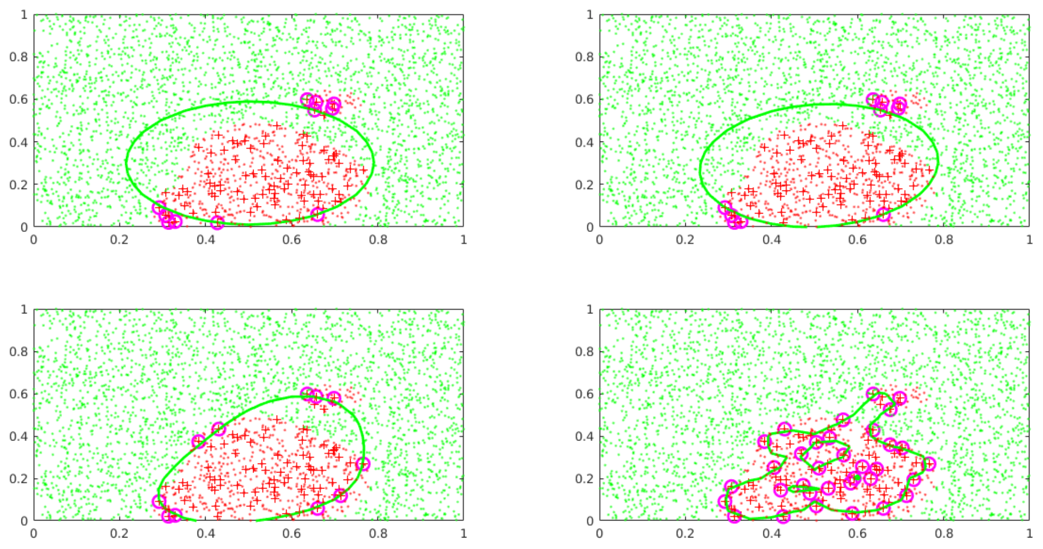


Illustration using RBF kernel with four different gamma values. Top-left gamma = 0.1, top-right gamma = 1, bottom-left gamma = 10, bottom-right gamma = 100.