Support Vector Machine and PCA

Tasks

1. Implement the SVM algorithm using the unconstrained optimisation equations from the lectures and the `scipy.optimizers.minimize` method in Scipy, which is a black box optimiser.

   For this, you will need to implement a function that returns the value of the objective given the current α vector. See more in the documentation. Calculate the Q matrix beforehand, do not calculate it in every call of this function[1]!

   a. Using the algorithm, classify the dataset found in `svm_linear.npz`. Use C values of 1, 10 and 100.

   b. How many nonzero (accounting for rounding/computational errors) elements does the α vector have? I.e. how many support vectors did the algorithm find?

   c. Plot the data and the separator line (as usual), and mark the support vectors on your plot. What are the scores (i.e. $y(w^T x + b)$) for these samples? (You can use the `visualise_nonlin` function.)

   d. Print the α_is for the support vectors on your plot.

   e. Compare the result of the SVM with the PLA and the Logistic Regression. Which one had the best generalisation (in this case)?

2. *(Optional)* Using the kernel trick, classify the dataset found in `rings.npz` from Lab 4. I would suggest using a third degree polynomial kernel for this.

   What is the test/train accuracy of the classification? Plot the data and the separator curve. (For the plot, you can use the `visualise_nonlin` function.)

   Mark the support vectors again; how many of them are there?

3. Given the dataset `PCA.npz` that was generated by applying a linear transformation and some noise to a 2D dateset; apply the PCA algorithm to find the two-dimensional factors underlying the data!

   The final orientation should be apparent from visualising (a scatterplot of) the 2D data[2]. What was the original transformation matrix used to create the transformed dataset?

[1] Note that you can only do this if you do not have large amount of data, because it is quadratic in N.
[2] Note: I am left-handed.