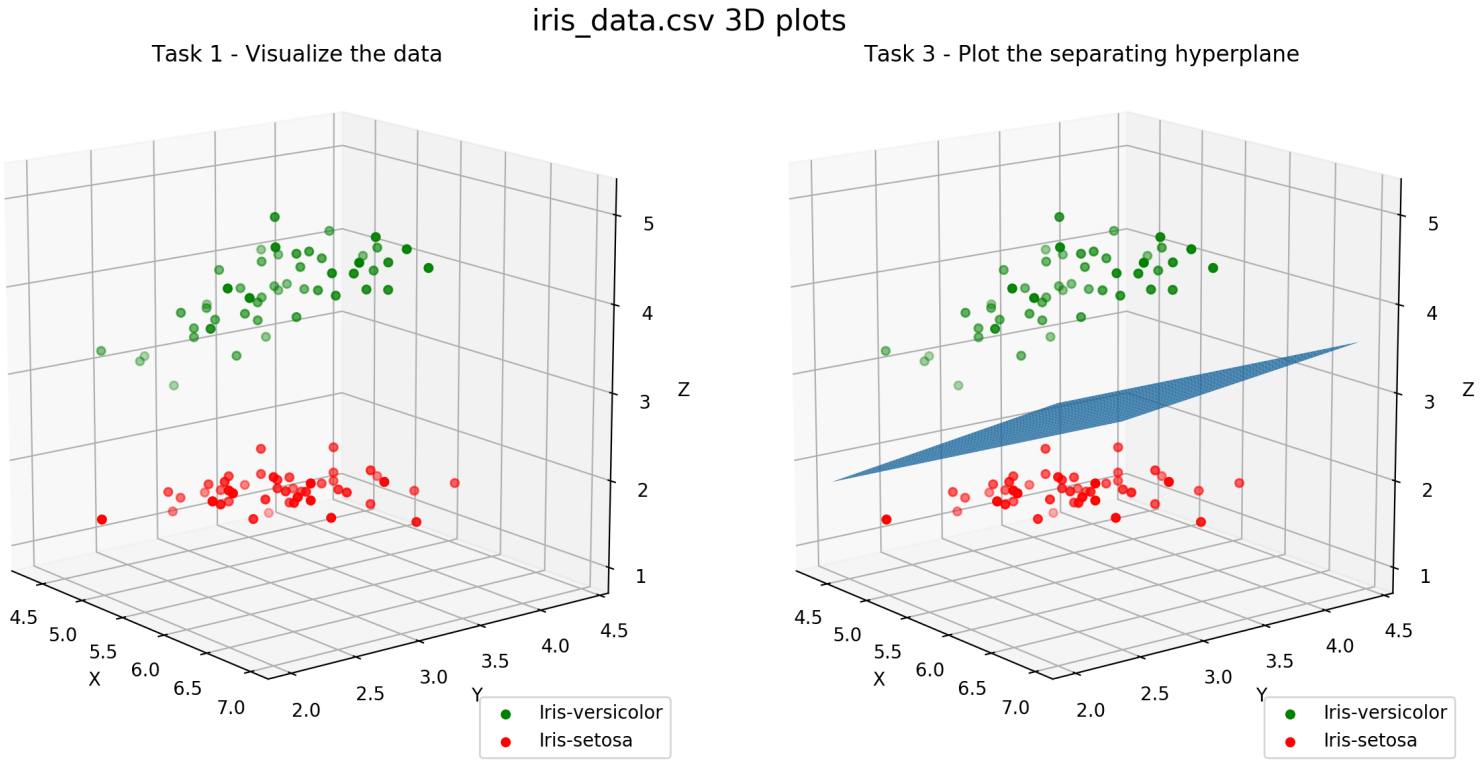
# Foundations of Data Analysis - SS20
# Lab Assignment:
# Supervised Learning
# Write-up

## Task 1 - Visualise the data &
## Task 3 - Plot the separating hyperplane

Below you can find the 3D scatter plots required by the Tasks 1 and 3. The Task 1 plot on the left produces a 3-d scatterplot of the lab_iris_data.csv dataset where the data points are coloured differently according to their class, red for Iris-setosa and green for Iris-versicolor with each of their dimensions corresponding to a feature. The Task 3 plot on the right extends the Task 1 plot with a separating hyperplane surface calculated using the findings in Task 2. The hyperplane surface is the decision boundary which represents all the points in the feature space in which the probability of being in class 1 is the same as that of class 0, when h_theta(x) = 0.5.




## Task 2 - Implement and train a regularised logistic regression model using stochastic gradient descent

The classifier is trained on the lab_iris_data.csv dataset using the algorithm provided in the assignment sheet to implement stochastic gradient descent for the regularised logistic regression model starting with values λ regularisation factor = 1, T number of iterations = 100, k batch size = 20, and ηinit (initial learning) = 0.1.

*Q1 Do these parameter values result in a well-trained model? how do you know?*

The starting values of the mentioned parameters result in a very well trained model with a 100% accuracy. This is because the data in the lab_iris_data.csv dataset is linearly separable thus the starting values of $\lambda = 1$, $T = 100$, $k = 20$, and $\eta init = 0.1$ results in a sufficiently trained model. For calculating this accuracy, the theta vector with given parameters was computed and then fed into the hypothesis function along with the features x of the dataset. The hypothesis function then returned the probability of each sample being in a certain class, classifies a sample x(i) in class 1 if $P(y = 1 \mid x(i)) = h\theta(x(i)) \geq 12$ and class 0 otherwise. The result of the hypothesis vector was then compared with the actual labels of the data points to calculate the accuracy.

### *Q2 Experiment by trying different values and see if you find a set that performs better or faster.*

Since the accuracy is already 100% the values result in a well-trained model. After experimenting with different values I observed the pattern described in Answer 3, based on that one way to make the performance faster would be by reducing the values of T and/or k (but not by too much), while increasing the ηinit value and keeping low regularisation factor. This would make the classifier function quicker because it would reduce the number of iterations required by it to calculate theta.

### *Q3 Explicitly describe how varying each of the four parameters affects training and the final model.*

I observed that reducing the ηinit value reduces the accuracy of the model, increasing the λ regularisation factor to large values decreases the accuracy, reducing the T and k values also reduces the accuracy.

## Task 4 - Further questions

*Q1 Given the above Note in Task 2 about non-stochastic Gradient Descent, why do you think the different variations of the algorithm exist, i.e. when is one more useful than the other?*

Stochastic gradient descent is a variation of gradient descent where it uses a batch of samples per iteration for theta calculation. In implementation of this assignment, we have used the stochastic gradient descent, our classifier approximates the true gradient by computing and updating θ after each batch of training samples in every iteration rather than summing over all samples before updating θ in case of non-stochastic Gradient Descent. The stochastic gradient exists and is useful when the size of the dataset is very large thus non stochastic gradient would be slow in those case scenarios since it would have to iterate through the full dataset and calculate the theta by summing over the samples, in such cases the Stochastic Gradient Descent would be much faster because we are only dealing with a batch of samples in each iteration. On the other hand the error function of the gradient descent is better minimised than the stochastic gradient descent.

*Q2 Under what conditions do they produce the same trained model?*

If the batch size is the same as the total size of the dataset we should have the same trained model.

*Q3 How (if at all) would you change the pseudocode in Algorithm 1 to use a different loss function?*

The gradient calculations (line 5) used in the Algorithm would change while keeping everything else the same. In our implementation the definition of the gradient function which is called in the classifier method will be needed to modified according to the new loss function.

*Q4 How does the regularised model we used here differ from the result if we had not used a regularisation term?*

The regularisation model is adopted to avoid overfitting by putting a control on the fitting parameters. If we had not used a regularisation term than it is possible that although our model appears to be sufficiently trained here, it might perform worse for new unseen samples.