

Q1 Support Vector Machines (60 Points)

Figure 1 shows a plot of the data in the file ‘HEIGHT-DATA.csv’. In this problem you will learn a support vector machine (SVM) classifier on the same data. Implementing SVM from scratch is non-trivial; so you will use *sklearn’s* svm package. Look it up to learn about it. Specifically, in the class we only covered linear SVMs i.e. which can find a linear boundary. For linear SVMs, *sklearn’s* svm package has a function ‘*LinearSVC*’. In general, for *sklearn*, first you create an object for a model (passing some parameters), and then call *.fit()* to fit the model. You may go ahead with the default parameter settings, but playing around with parameters such as *max_iter*, *C*, and *class_weight* can come in handy. The parameter *class_weight* helps you assign different costs for making mistakes while classifying a particular class *e.g.* in practice one might be okay with misclassifying one class while misclassifying another class could be disastrous. Please do not confuse *class_weight* with the classifier weights that you’re learning. *class_weight* specifies how costly it is to make error on each class *e.g.* if it’s twice as costly to misclassify class 1 as compared to class 2, the class weights for class 1 and 2 could be: 2 and 1 respectively, or 1 and 0.5, or 6 and 3 (only the ratio matters).

- (a) **(20 Points)** Using the default value for *class_weight*, train a linear SVM classifier for the height prediction problem. Report the learned weights and bias. Once the classifier has been learned (*i.e.* after you call *.fit()*), you may access several properties in your model object. The weight and bias can be accessed through ‘*model.coef_*’ and ‘*model.intercept_*’. Plot the separating line on your data. Is it different from the one you learned in assignment 2? If so, why?
- (b) **(40 Points)** By changing the *class_weight* parameter, find a setting that classifies all the points from ‘plus’ class correctly. To clarify, you are being asked to find a line which is not allowed to make *any* error while classifying the ‘plus’ class but still needs to minimize the errors for the ‘dot’ class, to the extent possible. Plot this new line and report the values for weight and bias. Report the values for class weights.

[**Hint 1**]: Since you are dealing with two classes, the parameter *class_weight* would be a dictionary object with two elements, with class labels as keys, and the corresponding weights as values.

[**Hint 2**]: You are not required to write a script to figure out the class weights to get all examples from ‘plus’ class right. Once you have implemented part (a), you can play around with *class_weights*, plot the line and visually see when the last ‘plus’ falls above the line. Another hint: Since the ratio of class weights is what actually matters, you may fix one weight and change the other.

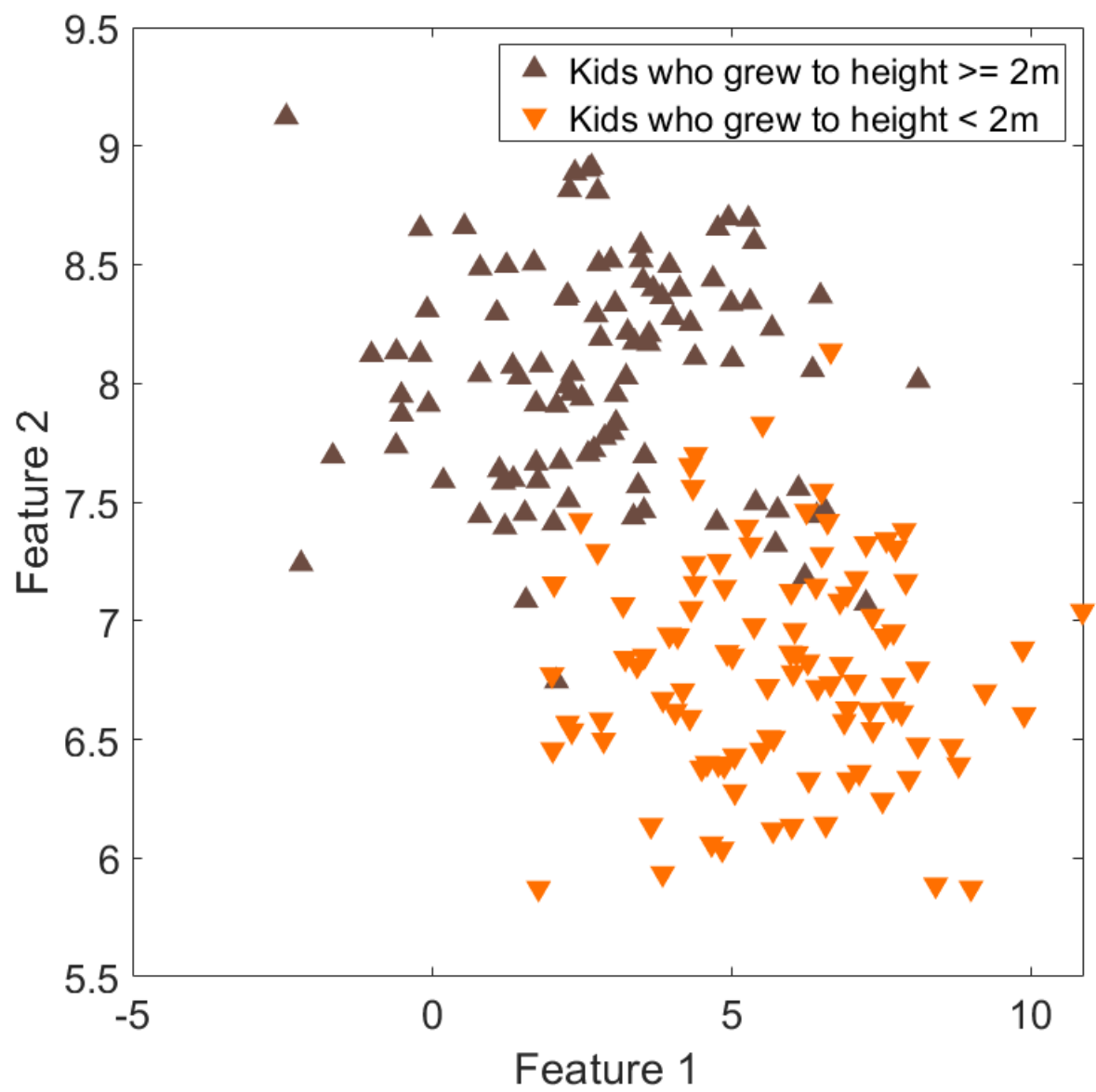


Figure 1: Training data for the height prediction problem

