**Exercises for 10/5 (heavily based on HOML Ch. 5-6 exercises)**

*Submission instructions:*

* *Make a copy of this document and rename it with your FIRSTNAME.LASTNAME.056*
* *Share it with* [*denys.katerenchuk@gmail.com*](mailto:denys.katerenchuk@gmail.com) *and* [***rebeccalevitan@share.brooklyn.edu***](mailto:rebeccalevitan@share.brooklyn.edu)*Do not share it with other levitan email addresses*
* *Submission closes on* ***Sunday 12pm***

1. What is the fundamental idea behind Support Vector Machines?

SVM’s are meant to split the two different classes via the support vectors while providing as much wiggle room as possible for predictions. This way when you get an unknown instance, you can be confident in your prediction if it’s far enough away from the decision boundary (aka not on the street). [Note: for problems where generalization is more important than a perfect split for all instances, you can widen the street further, thereby including a few instances on it, called margin violations].

1. What is a support vector?

A support vector is an instance on the street [after training], which are the only instances that actually have an effect on the decision boundary [for predictions].

1. Why is it important to scale the inputs when using SVMs?

If it’s not scaled then calculating the widest street will take heavier consideration for the larger-valued features, as they will have the most effect on the distance from the decision boundary.

1. Can an SVM classifier output a confidence score when it classifies an instance? What about a probability?

You can use the distance between the instance you’re predicting and the decision boundary as a confidence score (the further the better). You can also enable probability functions in sklearn by passing that parameter as true, and use them during your predictions.

1. Suppose you have trained an SVM classifier with an RBF kernel, but it seems to underfit the training set. Should you increase or decrease gamma? What about C?

For both gamma and C lower values are more regularized, so you increase them to relax the regularization [and help prevent underfitting].

1. Train an SVM classifier on the MNIST dataset. Since SVM classifiers are binary classifiers, you will need to use one-versus-the-rest to classify all 10 digits. This will take a long time, so tune the hyperparameters using small validation sets to speed up the process. What accuracy can you reach? Submit a colab notebook with the code using the naming/sharing instructions outlined above.

See shared ipynb file (should run error free but randomized-search is taking its time…)

1. What is the approximate depth of a decision tree trained (without restrictions) on a training set with one million instances?

log2(1,000,000) ~ 20

1. If a decision tree is overfitting the training set, is it a good idea to try decreasing *max\_depth*?

Yes; the less depth it is allowed the less freedom it has to try to fit individual training instances.

1. If it takes one hour to train a decision tree on a training set containing 1 million instances, approximately how long will it take to train another decision tree on a training set containing 10 million instances?
   * Given: n \* 1,000,000 \* log2(1,000,000) = 1 hr
   * Isolate n: n = 1 hr / (1,000,000 \* log2(1,000,000))
   * Solve: n \* 10,000,000 \* log2(10,000,000) = ?
   * Substitute n: 1 hr / (1,000,000 \* log2(1,000,000)) \* 10,000,000 \* log2(10,000,000)
   * = 1 hr \* (10,000,000 / 1,000,000) \* (log2(10,000,000) / log2(1,000,000))
   * = 11.67 hrs