|  |
| --- |
| **Homework 3****COSC 6342: Machine Learning****University of Houston****Department of Computer Science****Sent on: Oct. 27, 2017; Due: Nov. 10, 2017** |

|  |
| --- |
| Names:  Sidharth \_\_\_\_\_\_\_ PPSID 1\_\_\_\_\_\_\_\_  Chieh Chen PPSID 0837931 |

**You may work individually or with 1 partner. No groups of more than 2 people are allowed.**

You will make use of various python libraries and their dependencies, such as:

Scikit-learn: <http://scikit-learn.org/stable/>

Matplotlib: <https://matplotlib.org/>

Numpy: <http://www.numpy.org/>

You can install them manually, compile from source, obtain using pip (easier):

<https://pypi.python.org/pypi/pip>

Or just get everything at once with Anaconda:

<https://www.anaconda.com/>

Most likely you have everything already, because of doing homework 2.

**FAQ:** For all questions, do not just say ‘Yes’ or ‘No’. Provide your reasoning and give some examples; draw a picture if you want. For all mathematical concepts, show proofs and formultations. Do not just give a number. **Why** is more important that **How**. Showing a good understanding of the question and sound reasoning that results in a wrong answer will still get you nearly full credit. Do not just print-screen and copy paste; produce nice figures. Attach your source code into an Appendix. Any results, graphs, measurements that are relevant should be part of the project; additional resources go into an Appendix. If asked to implement something that takes a long time to train, please attached model files. **Your report should be submitted in a pdf format. It should be styled like a normal conference submission.**

**Support Vector Machines**

**(and some Learning Theory)**

**Questions 1-3: Foundations of SVM. Explicit Projections.**

*For the following 3 questions, an explicit mapping means introducing a new feature that makes the data classifiable by projecting it into a different space. You are asked to imagine a possible projection; given a dataset D; pretend you know what the “real” D that lives in a different, higher dimensional space looks like. Think of the following: What would it look like if it were linearly separable there? Can it be, while still looking the same as in the original space?*

**Question 1: (2 points)**



Figure 1: Binary classification in 1D space

Draw (by hand is fine) a transformation to a higher dimensional space that will make this example linearly separable. *In other words, draw a sketch of how an SVM would be able to project and separate these samples.*

**Question 2: (4 points)**

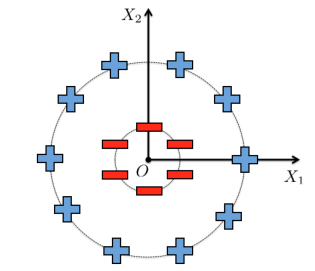


Figure 2: Circles in 2D space

How could the above data be made linearly separable? (Remember, we are not using a kernel trick!) (1 point) Draw the resulting projection. (1 point). Give the formula for the new feature in terms of X1 and X2.(1 point).Give the (general) equation for the hyperplane separating the two classes in this space. (1 point)

*(Hint: think of various transformations of X1 and X2 that would result in enlarged space, e.g. X1X2 or X12)*

**Question 3: (4 points)** Project the hyperplane separating the classes from higher-dimensional space back into R2 by drawing the new decision boundary in 2D (you can do so on the figure above) (1 point). How is it possible that a linear classifier draws a nonlinear boundary? (2 points) Explain the difference between feature space and input space. (1 point)

**Subsection Total = 10 points**

The approach used in Questions 1-3 serves as a simple illustration to partially understand SVMs, but it does not scale well. Imagine having thousands of features and tens of thousands of samples. An approximation by sampling can be used, but it is only occasionally accurate. Hence, we need another solution.

**Questions 4-6: Introduction to Kernels**

**Recap:** A very simple and intuitive way of thinking about kernels (at least for SVMs) is as a similarity function. Given two objects, the kernel outputs some similarity score.

Mathematically, a Kernel is a way of computing the dot product of two vectors **x** and **y** in some (possibly very high dimensional) feature space, which is why kernel functions are sometimes called "a generalized dot product".

Suppose we have a mapping *φ* : R*n*→R*m* that brings our vectors in Rn to some feature space Rm. Then the dot product of **x** and **y** in this space is *φ*(**x**)t *φ*(**y**). A kernel is a function *k* that corresponds to this dot product, i.e. *k*(**x**,**y**)=*φ*(**x**)t *φ*(**y**).

Why is this useful?

Kernels give a way to compute dot products in some feature space without even knowing what this space is and what is *φ* - and without really caring too much as to how many dimensions are there.

**Question 4: (4 points)** Explain in a paragraph or so what a kernel trick is. (2 points) What is the difference between the explicit mapping we did in Questions 1-3 and the kernel trick utilized by Support Vector Machines? (2 points)

**Question 5: (6 points, 3 points per question)** For each of the following, prove whether it can be a valid kernel.

1. *k*(*x, y*)= (*xy* + 1)2
2. *k*(*x, y*)= (*xy* - 1)3

*Note: A yes/no answer here will be marked as wrong and no partial credit will be given. Formal proof is required for full credit. Informal but sound reasoning is sufficient for partial credit.*

**Question 6 (8 points):** Consider a Gaussian kernel *k*(**x**,**y**)=exp(−*γ*∥**x**−**y**∥2).

1. Use Taylor-expansion to find the dimensionality of the corresponding *φ.*  (3 pts)
2. What is the VC-dimension of an SVM classifier using this kernel and why? (2 pts)
3. In general, what does a high VC dimension tell us about an algorithm as opposed to a low one? When would we want to pick one over the other? For which family of classifiers is high VC dimension not really a problem and why? (3 pts)

**Question 10: Practical Considerations (5 points, 1 each)**

1. Can a properly tuned Gaussian Kernel ever perform worse than a Linear SVM? Why?
2. In practice, imagine you have a dataset with a huge number of features, such as some text classification data. Which kernel would you choose and why?
3. Why are SVMs especially sensitive to data being normalized or not? In the other words, why is it so important to do so?
4. Explain the importance of the C parameter of SVMs. What does it do?
5. Why are classical SVMs particularly memory intensive?

**Question 7: (7 points)**

Take the following code as your base:

<http://scikit-learn.org/stable/auto_examples/exercises/plot_iris_exercise.html#sphx-glr-auto-examples-exercises-plot-iris-exercise-py>

Modify the feature selection part. Whereas in the original code all but the first 2 features are dropped, you want to pick 2 features intelligently. See here (basically merge the codes given):

<http://scikit-learn.org/stable/auto_examples/plot_compare_reduction.html#sphx-glr-auto-examples-plot-compare-reduction-py>

Then run a comparison of linear, polynomial and rbf kernels. Attach the graph and the code.

**Subsection Total = 30 points Running Total = 40 points**

**Question 8: Design and Implement your own Kernel (30 points)**

Consider the problem in Figure 2. You have already performed an explicit mapping for it. Now your task is to design and implement a custom kernel.

*Note: We understand that a Gaussian kernel would work, but you cannot use it or any modification of it. Do not use any of the standard kernels; write your own, it’s OK if it does not do as well. You are not being graded on accuracy. This of course does not mean “some random formula”. Show how you arrived at your choice, either mathematically or empirically (then you would need to give me results of your experiments like graphs or put in a table with various ones you tried and their accuracy numbers, etc.)*

In scikit-learn, you can define your own kernels by defining it as a python function and to the keyword kernel in the constructor. Your kernel must take as arguments two matrices of shape (n\_samples\_1, n\_features), (n\_samples\_2, n\_features)and return a kernel matrix of shape (n\_samples\_1, n\_samples\_2). The following code defines a linear kernel and creates a classifier that will use that kernel:

**import** **numpy** **as** **np**  
**from** **sklearn** **import** svm

**def** my\_kernel(X, Y):  
 **return** np.dot(X, Y.T)  
  
clf = svm.SVC(kernel=my\_kernel)

Recall **Figure 2** and the solution you produced.

1. Use <http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html#sklearn.datasets.make_circles> to generate a dataset similar to that. (2 points)
2. Is this dataset PAC Learnable? Show why yes or why not. (3 pts)
3. Implement a python function that will be your new kernel. Use your custom SVM to classify. Graph (feel free to re-use code from Question 7 for graphing). Attach your code. (10 pts)
4. How did you arrive from your answer to Question 3 to the kernel you implemented? Explain **in detail** (learning theory might come in useful)**.** *Cannot use popular existing kernels, or just take one and modify it. (10 pts)*
5. Now use scikit-kit’s cross-validation tools to run your SVM against linear SVM, 10-fold. Report the results. (5 pts).

**Subsection Total = 30 points Running Total = 70 points**

**Ensemble Learning**

**(and some bias and variance)**

**Recap:** The bias of x‘ is the difference between the expected value of a hypothesis and the true value f(x‘). The variance of x‘ is how far out the outcomes of a set of hypotheses (learned on different datasets) are spread out from their mean.

**Recap:** Given a set of training samples, bootstrap size, and a learning algorithm, the bagging method returns a set of hypothesis learned on each of the bootstrap samples hb.

**Recap:** TheAdaboost method is one of the commonly used ensemble method. Adaboost learns weights for both training samples and the classifiers. Unlike the Bagging method, Adaboost makes a final decision by a weighted voting over the classifiers. At every iteration, Adaboost updates the weights on the training samples so that the classifier can focus on learning samples that are difficult for classification.

**Question 9 Bias/Variance Decomposition for Regression (10 pts):** Show that error can be expressed as bias2 + variance + noise by performing the decomposition. Show all the major steps involved. For each step, write in one or two short sentences your understanding of it.

**Question 11: (20 points)**

1. Get lonosphere: <https://archive.ics.uci.edu/ml/datasets/Ionosphere>
2. Choose any other dataset that is sufficiently large and complex.
3. Using scikit-learn, run classification on it first with Bagging and then with AdaBoost. For each, select 3 algorithms you are going to compare, ideally you should pick weak learners like Decision Tree. (So, you will be using 6 classifiers in total). Perform your classification using K-fold cross validation, with K being incremented from 2 to 4 to 6 to 8 to 10. Record the test and train error rates as K changes. Plot them. (10 points)
4. Is anything interesting happening as K changes? Explain in terms of Bias/Variance and overfitting/underfitting. How do you think Bagging and Boosting compare? Which component of the error does each affect and how? (8 points)
5. When would you choose AdaBoost over Bagging? What about the other way around? (2 points)

**Section Total: 30 points**

**Total: 70+30=100 points**