CS 7641 CSE/ISYE 6740 Homework 4

Yao Xie

Deadline: Monday March 19, 11:55pm

- Submit your answers as an electronic copy on T-square.
- No unapproved extension of deadline is allowed. Zero credit will be assigned for late submissions. Email request for late submission may not be replied.
- For typed answers with LaTeX (recommended) or word processors, extra credits will be given. If you handwrite, try to be clear as much as possible. No credit may be given to unreadable handwriting.
- Explicitly mention your collaborators if any.

1 SVM. [15 points]

1. Explain why can we set the margin $c = 1$ in SVM formulation?
2. Using Lagrangian dual, show that the the weight vector can be represented as

$$w = \sum_{i=1}^{m} \alpha_i y^i x^i.$$

   where α_i are the dual variables. What does this imply in terms of how to relate data to w?
3. Explain why only the data points on the "margin" will contribute to the sum above, i.e., playing a role in defining w. Hint: use the Lagrangian multiplier derivation and KKT condition we discussed in class.

2 Neural networks [10 points].

1. Consider a neural networks for a binary classification using sigmoid function for each unit. If the network has no hidden layer, explain why the model is equivalent to logistic regression.
2. Consider a simple two-layer network in the lecture slides. Given the cost function used to training the neural networks

$$\ell(w, \alpha, \beta) = \sum_{i=1}^{m} (y^i - \sigma(w^T z^i))^2$$

   where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function. Show the that the gradient is given by

$$\frac{\partial \ell(w, \alpha, \beta)}{\partial w} = \sum_{i=1}^{m} 2(y^i - \sigma(u^i))\sigma(u^i)(1 - \sigma(u^i))z^i.$$

   where $z_1^i = \sigma(\alpha^T x^i)$, $z_2^i = \sigma(\beta^T x^i)$. Also find the gradient of ℓ with respect to α and β.

3 Convolution [15 points].

Define the window function

$$u(t) = \begin{cases} 1, & u \in [0,1] \\ 0, & \text{otherwise.} \end{cases}$$

Compute the convolution

$$y(t) = (u(t+2) - u(t+1) + u(t-1) - u(t-2)) \star (u(t) - u(t-1)).$$

Hint: You can solve this problem graphically and then write the final expression.

4 Deriving weights α_t for AdaBoost [20 points].

Using the equation

$$\frac{\partial L(\alpha_t)}{\partial \alpha_t} = 0$$

to derive and show that the expression for the weight should

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right).$$

5 Programming Problem: Decision Trees and Random Forests [40 points]

This is the first programming question of the term in Python and it requires you to work with decision trees and random forests [1].

The homework source file is rfdigits.py. The data can be found in 555mnistData.tar.bz. This is a subset of the LeCun?s MNIST hand- written digit recognition dataset containing just the digits 0 through 5. The full dataset is available at `http://yann.lecun.com/exdb/mnist/` The dataset is split into training and testing pictures. Do all training on the training pictures, and reserve the testing pictures for classification.

You can read the documentation in the source files and the examples examplesDatatools.py, examplesLindisc.py, examplesTrees.py to get a better idea of what is going on.

Next, you are to implement a Random Forest classifier for classifying grayscale images of handwritten digits. For simplicity, we have provided a rfdigits.py starter file that contains the full skeleton of the random forest code (both learning and classification) as well as the functionality to load in the data and compute features on it.

You should implement the missing parts of the code as specified in the file. Two of the parts are for the classification side and one is for the learning side. All three are important to understanding the algorithm.

The features that have been defined in this code are too many to enumerate. So, during the randomized learning of the tree, you will have to dynamically instantiate a random set of, say 100, features and learn the best query based on them. Then, at the next node, you instantiate a new random set of features and so on. The actual features that we will use are simple weighted sums of pixel values (all of the images are 28-by-28 grayscale). The class `KSum_Feature` has been provided for you to easily compute these features. Finally, the class `DTQuery_KSum` has been provided to encapsulate a query based on these features; you do not need to use this one if you don't want, but it is recommended.

You should complete the `rfdigits.py` file, and execute it (by typing, for example, `python rfdigits.py` assuming your `PYTHONPATH` is set properly. Running this script will load a subset of the data, train the tree, and then compute the accuracy on the training set as well as the accuracy on the provided testing set.

[1] Courtesy: Dr. Jason J. Corso.

Consider what happens when you run the `rfdigits.py` file; i.e., what are you accuracies? How do the accuracies vary as you change the parameters of the tree (what if you use a featureDepth of 10 or 100, how does this change the accuracy?).

You are encouraged to explore the classifier and the problem further to, for example, display images of digits that were misclassified or visualize the paths down a tree given the selected type of feature.