CS8803: STR, Spring 2015. Lab 3: Supervised Learning

You may work in groups of up to 3.

Due: Wednesday, April 8, beginning of class

Assignment

This assignment gives you a chance to implement some of the supervised learning algorithms discussed in class on real data.

Automated interpretation of ladar data is crucial for outdoor vehicle operation. Your goal is to apply algorithms we discussed in class to this problem and in particular to compare and contrast the various methods. You will be building a system to classify ladar points into one of five categories: ground (supporting surface), vegetation, facade, pole, and wire.

You must **implement** 3 learners we discussed in class. (Talk to me if you're interested in implementing other variants instead.)

- Gaussian Process Regression with a Gaussian RBF kernel (choose two classes)
- Bayesian Linear Regression (choose two classes)
- Bayesian Logistic Regression (via an EKF) **OR** Kernel Logistic Regression (via gradient descent)
- **Extra Credit**: method of your choosing (SVM, Neural Networks, Decision Trees/Forests, Gaussian process with another nonlinear kernel, etc.).

Data

The datasets you should use are available in the data folder. They are 3D point-clouds of Oakland in Pittsburgh (see `http://www.cs.cmu.edu/~vmr/datasets/oakland_3d/cvpr09/doc/`). Features are provided courtesy of Dan Munoz, but you are welcome to come up with new features to use (please explain if you do). You should not distribute the data without permission.

There are five classes, their labels values are:

- 1004: Veg
- 1100: Wire
- 1103: Pole
- 1200: Ground
- 1400: Facade

Run each algorithm on both datasets and report on the performance.

What to turn in

I am interested in a short performance summary and nice visualizations. You should turn in your *code* and a 2-3 page *report* via email. For each learner you implemented, report on the following:

1. How well did it perform for online learning? Does it perform well on the held-out data?
2. Are there any classes that did not get classified well? Why do you think that is?
3. How easy was the learner to implement?
4. How long does the learner take (in terms of data points, dimensions, classes, etc...) for training and prediction?
5. Show images/movies of the classified data. Note that MATLAB is not very good at displaying thousands of 3D points; you might want to use VRML or python.
6. How did you choose (hyper)parameters (priors, kernel width, noise variance, prior variance, learning rate, etc. . .)?
7. How robust is this algorithm to noise? Take the current feature set and:
   - Add a large number of random features
   - Add a large number of features that are noise corrupted versions of the features already in the data-set.

You should also compare the learners' performance to each other. Did kernels help on this data set? Which one would you use on your robot? What would future work include?