CS8803: STR, Lab 3: Supervised Learning

Yujia Liu & Xiang Liu

April 2015

1 Objective

In this project, we are trying to implement some of the supervised learning algorithms to classify laser points (3D point-clouds of Oakland in Pittsburgh) into one of five categories: ground (supporting surface), vegetation, facade, pole, and wire. These algorithms includes Gaussian process regression with kernels (a RBF kernel and a SAM kernel), Bayesian linear regression, and kernel logistic regression. Each algorithm is implemented in Matlab and the result is visualize in python. All Matlab codes are shown in the appendix.

2 Methods

2.1 Gaussian Process Regression with RBF Kernel

In the Gaussian process regression, we implement the RBF kernel

$$K(x_i, x_j) = \exp\{\frac{-(x_i - x_j)^2}{2\gamma^2}\}$$

where $\gamma = 10$. And we compute the posterior mean of the new data point with

$$f(\hat{x^*}) = f(x)(K(x,x) + \sigma^2 I)^{-1} K(x, x^*)$$

where $\sigma = 10^-3$.

2.2 Bayesian Linear Regression

In Bayesian linear regression, we first derive the relationship between the moments parameter and the natural parameters

$$P_i = \Sigma_i^{-1}$$
$$J_i = P_i \mu_i$$

and the update equations for the natural parameters

$$J_i = \frac{y_i x_i}{\gamma^2} + J_{i-1}$$
$$P_i = \frac{x_i x_i^T}{\gamma^2} + P_{i-1}$$

where $\gamma = 0.01$. Then we have $f(\hat{x^*}) = x^* \mu_i = x^* P_i^{-1} J_i$ as the boundary to classify new data. This algorithm works online, i.e., every time new data comes in, J and P update and the variance decrease.

2.3 Kernel Logistic Regression

In kernel logistic regression, we try to minimize the loss function

$$L[f] = \sum_i \log(1 + \exp(-y_i f(x_i)) + \frac{\lambda}{2}\|f\|^2$$

We take the gradient of the function above

$$\nabla L[f] = \sum_i (\frac{-y_i \exp(-y_i f(x_i))}{1 + \exp(-y_i f(x_i))}) K(x_i, \cdot) + \lambda f$$

and update the coefficients $alpha_i$ corresponding to every $K(x_i, \cdot)$.

$$\alpha_i \leftarrow (1 - \eta\lambda)\alpha_i - \eta\gamma_i$$

where $\eta = 0.01$, $\lambda = 1$, and $\gamma_i = \frac{-y_i \exp(-y_i f(x_i))}{1+\exp(-y_i f(x_i))}$. The kernel we use here is the same as in the guassian process regression.

2.4 Gaussian Process Regression with SAM Kernel

A SAM kernel is defined as follows.

$$K(x_i, x_j) = \exp\{\frac{-acos(x_i * x_j^T)^2}{2\gamma^2}\}$$

.
where $\gamma = 10$. The posterior mean of a new data point is calculated as

$$f(\hat{x^*}) = f(x)(K(x, x) + \sigma^2 I)^{-1} K(x, x^*)$$

where $\sigma = 10^-3$.

3 Experiment & Result

The algorithms are implemented on Matlab for simulation. For first two learners, we randomly choose 1000 samples for training and 500 samples for testing. For the kernel logistic classifier, the training sample number is 3000 and training sample number is 1000.

a. How well did it perform for online learning? Does it perform well on the held-out data?

   (a) Gaussian process with RBF kernel achieves an accuracy larger than 95% for each pair of classes, and classification between ground and other classes are especially successful, since ground is easier to identify.

   (b) Bayesian linear regression achieves a 95% accuracy for ground classification, and a 85% accuracy for others, due to the linearity constraint of the method

   (c) Kernel logistic regression achieves an accuracy of 95% to classify all five classes.

   (d) Gaussian process with SAM kernel performs worse than with RBF kernel. Although sometimes an accuracy of 95% can be achieved, it falls to 70% from time to time.

b. Are there any classes that did not get classified well? Why do you think that is?
   For all four methods we used, poles and wires are more difficult to be classified. Especially when we classify them among five classes. This is largely due to the unbalanced data we have. Since in our dataset, the data labeled as ground and facade is much more than other labeled data. This leads to a higher accuracy for ground and facade classification than for poles and wires.

c. How easy was the learner to implement?
All the algorithms are easy to implement. The code scripts are programmed in 10 lines.

d. How long does the learner take for training and prediction?
In order to compare the processing time, we take 1000 and 500 points for training and prediction for all four algorithms.

(a) Gaussian process with RBF kernel: Training time: 0.047067, Prediction time: 0.000056.

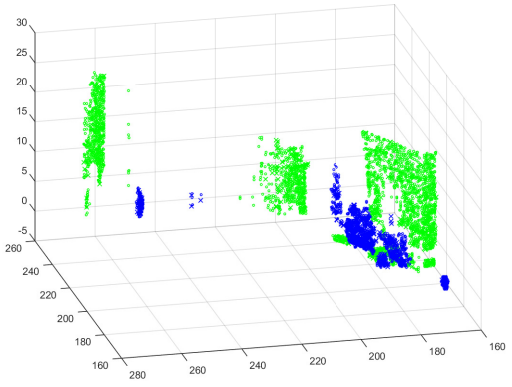
(b) Bayesian linear regression: Training time: 0.015910, Prediction time: 0.000610.

(c) Kernel logistic regression: Training time: 0.654493, Prediction time: 0.006187.

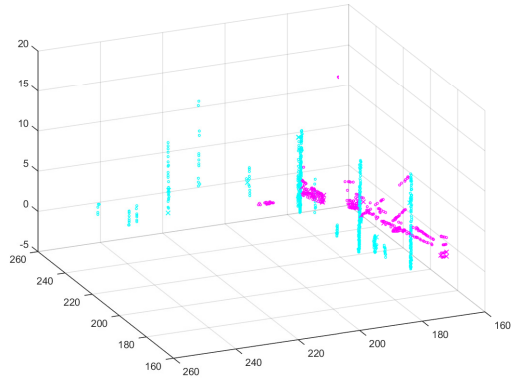
(d) Gaussian process with SAM kernel: Training time: 0.095446, Prediction time: 0.000717.

e. Show images of the classified data.
We designate blue for 1004, magenta for 1100, cyan for 1103, red for 1200 and green for 1400. We use 'o' to represent correctly classified data and 'x' to represent incorrect ones. The result computed by each algorithm is shown in the following figures.

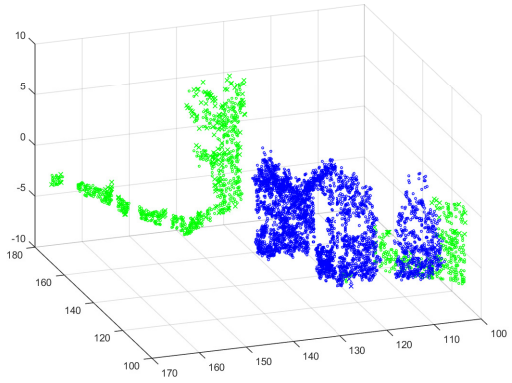
(a) Gaussian process with RBF kernel



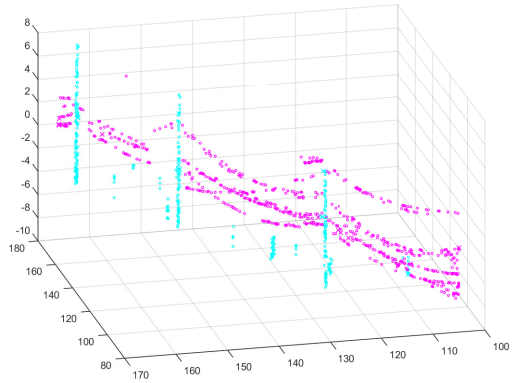
am:1004 & 1400



am:1100 & 1103

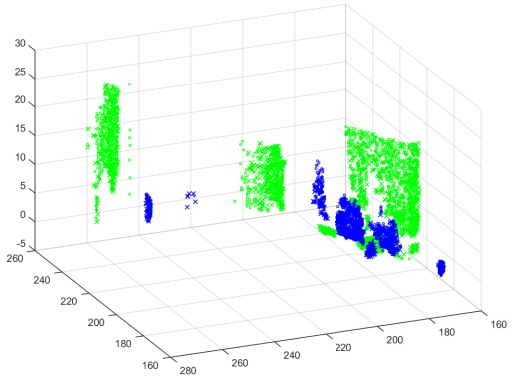


an:1004 & 1400

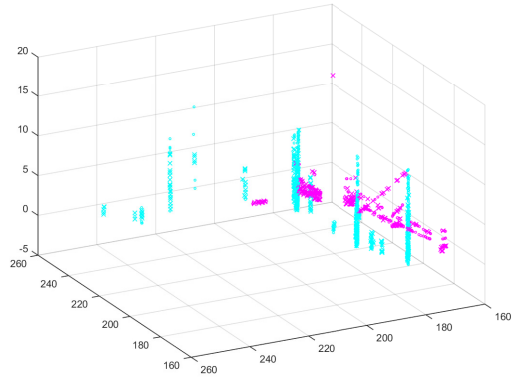


an:1100 & 1103

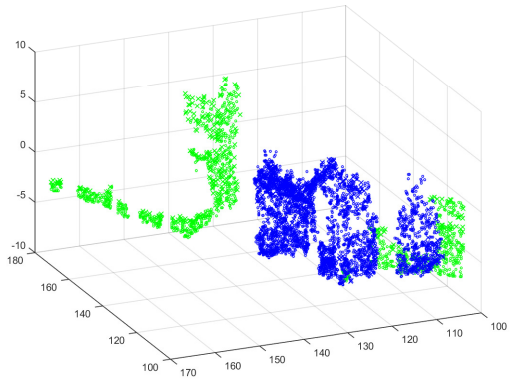
(b) Bayesian linear regression



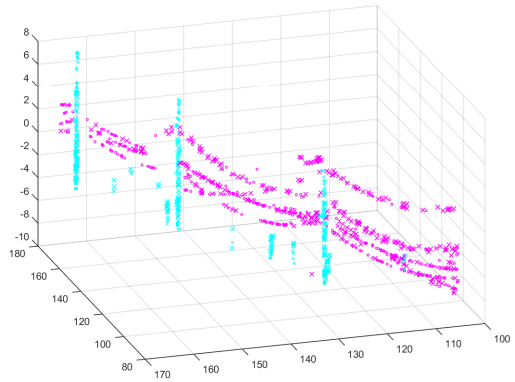
am:1004 & 1400



am:1100 & 1103

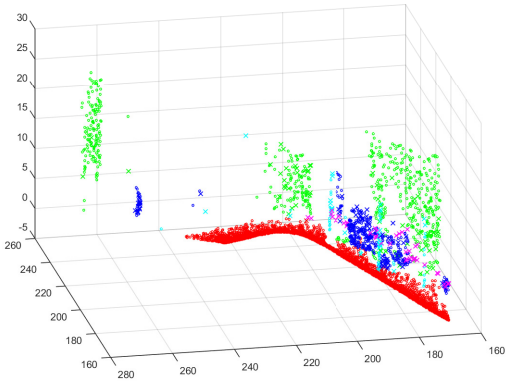


an:1004 & 1400

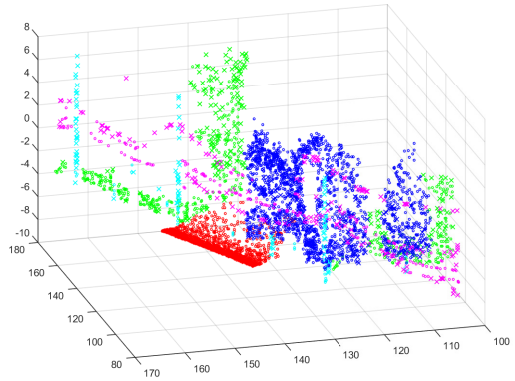


an:1100 & 1103

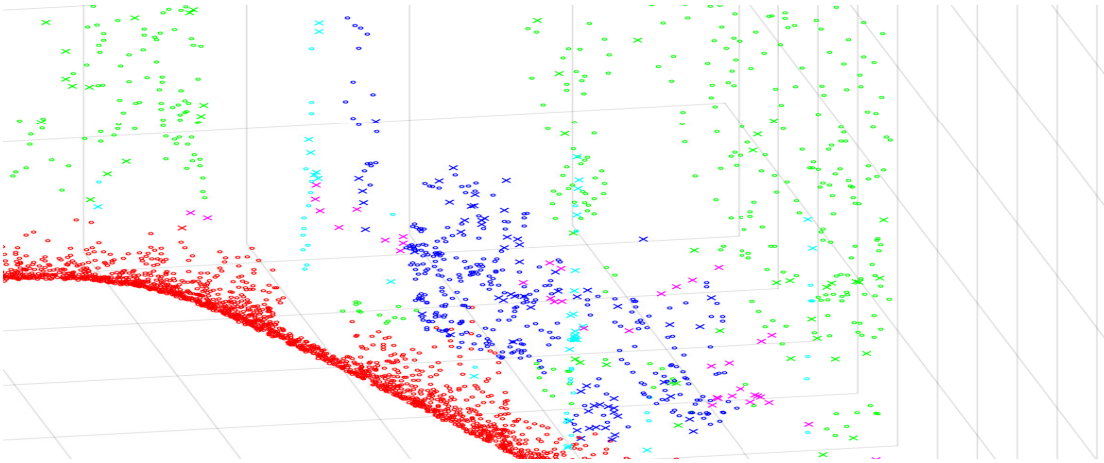
(c) Kernel logistic regression



am

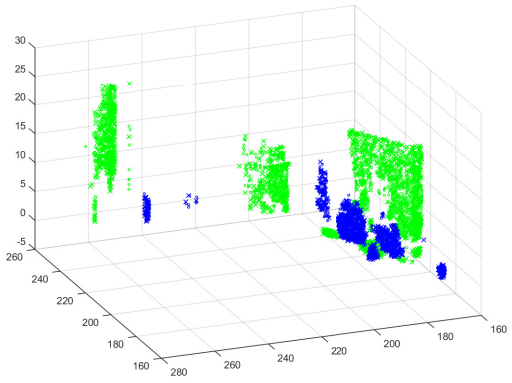


an

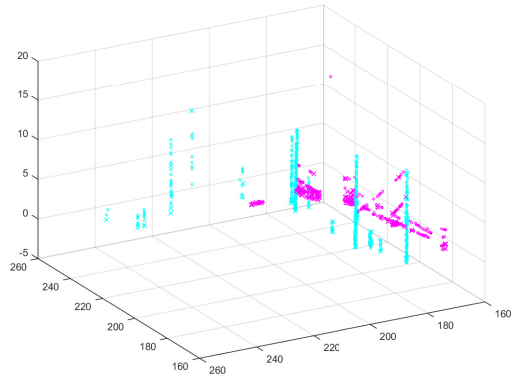


am: zoom in
o : correct x: incorrect

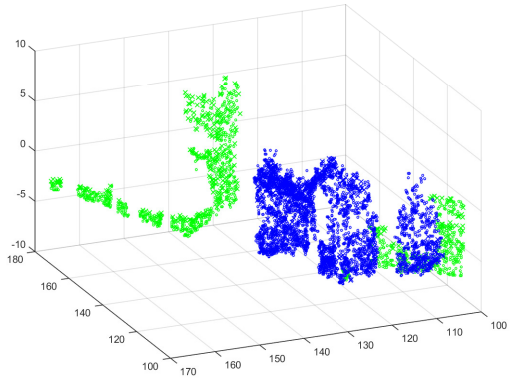
(d) Gaussian process with SAM kernel



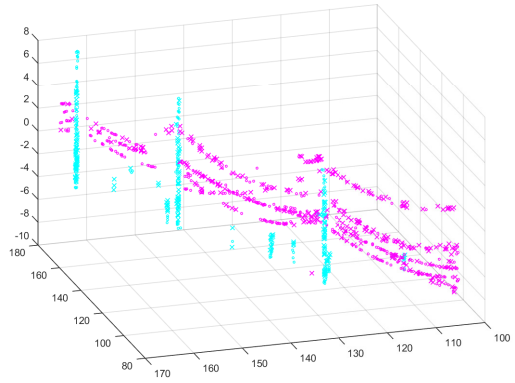
am:1004 & 1400



am:1100 & 1103



an:1004 & 1400



an:1100 & 1103

f. How did you choose (hyper)parameters.
The hyper-parameters in implementation are chosen by trial and error. As the four methods fit much on the dataset, tuning the parameters is not costing.

g. How robust is this algorithm to noise?

(a) Add a large number of random features.

i. Gaussian process. This kind of noise does not influence the performance of the Gaussian process method. The accuracy is more than 95%.

ii. Bayesian linear regression. The accuracy decreases about 3% under this kind of noise.

iii. Kernel logistic regression: The accuracy is reduced by about 3% under the noise.

iv. Gaussian process with SAM kernel: The noise influences the performance much. The accuracy is reduced by 25%.

(b) Add a large number of features that are noise corrupted versions of the features already in the dataset.

i. Gaussian process. This kind of noise does not influence the performance of the Gaussian process method. The accuracy is more than 95%.

ii. Bayesian linear regression: Similar to the first case, the accuracy is reduced by about 1% under this noise.

iii. Kernel logistic regression: The accuracy is reduced by about 1% under the noise.

iv. Gaussian process with SAM kernel: The kernel is not robust to this kind of noise. The accuracy decreases by about 20%.

4 Analysis

From the result we obtain above, we conclude that the Gaussian process regression performs the best among the four learners that we choose to implement. And Bayesian linear regression performs the worst, due to its linearity constraint. On the other hand, Bayesian linear regression is most efficient and cache-friendly, given our experiment result.

RBF kernel is definitely helpful on this dataset. However the SAM kernel is not good as much as RBF.

In practice, we might want to choose Gaussian process regression as our tool to classify this type of data when the size of data is not that large.

5 Future work

In our implementation, we didn't normalized or whitening our features before we use them. To take more advantage of those learners, pre-processing might be a good idea.

Given more than 80,000 data points, we only use several thousand of them to train our learners, which is a great waste of resource. However, we cannot process the whole dataset at one time due to the limit memory. This problem leads to some variant method of approximating kernel matrices and reduce the computational cost. Some approximation we discussed in class might come in handy in the future implementation.