COMS 4721: Machine Learning for Data Science

Columbia University, Spring 2023

Homework 2: Due Sunday, March 26, 2023 by 11:59PM

Please read these instructions to ensure you receive full credit on your homework. Submit the written portion of your homework as a *single* PDF file through Courseworks (less than 5MB). In addition to your PDF write-up, submit all code written by you in their original extensions through Courseworks (e.g., .m, .r, .py, .c). Any coding language is acceptable, but do not submit notebooks. Also, do not wrap your files in .rar, .zip, .tar and do not submit your write-up in .doc or other file type. When resubmitting homeworks, please be sure to resubmit *all files*. Your grade will be based on the contents of one PDF file and the original source code. Additional files will be ignored. We will not run your code, so everything you are asked to show should be put in the PDF file. Show all work for full credit.

Late submission policy: Late homeworks will have 0.1% deducted from the final grade for each minute late. *Your homework submission time will be based on the time of your **last** submission to Courseworks. I will not revert to an earlier submission time!* Therefore, do not re-submit after midnight on the due date unless you are confident the new submission is significantly better to overcompensate for the points lost. Submission time is non-negotiable and will be based on the time you submitted your last file to Courseworks. The number of points deducted will be rounded to the nearest integer.

Problem 1 – 20 points

In this problem you will derive a naive Bayes classifier. For a labeled set of data $(y_1, x_1), \ldots, (y_n, x_n)$, where for this problem $y \in \{0, 1\}$ and x is a D-dimensional vector of counts, the Bayes classifier observes a new x_0 and predicts y_0 as

$$y_0 \;=\; \arg\max_y \; p(y_0 = y|\pi) \prod_{d=1}^{D} p(x_{0,d}|\lambda_{y,d}).$$

The distribution $p(y_0 = y|\pi) = \text{Bernoulli}(y|\pi)$. What is "naive" about this classifier is the assumption that all D dimensions of x are independent. Assume that each dimension of x is Poisson distributed with a Gamma prior. The full generative process is

$$\text{Data: } y_i \overset{iid}{\sim} \text{Bern}(\pi), \quad x_{i,d}|y_i \sim \text{Pois}(\lambda_{y_i,d}),\ d = 1, \ldots, D \qquad \text{Prior: } \lambda_{y,d} \overset{iid}{\sim} \text{Gamma}(2, 1)$$

Derive the solution for π and each $\lambda_{y,d}$ by maximizing

$$\widehat{\pi}, \widehat{\lambda}_{0,1:D}, \widehat{\lambda}_{1,1:D} \;=\; \arg \max_{\pi, \widehat{\lambda}_{0,1:D}, \widehat{\lambda}_{1,1:D}} \; \sum_{i=1}^{n} \ln p(y_i|\pi) + \sum_{d=1}^{D} \left(\ln p(\lambda_{0,d}) + \ln p(\lambda_{1,d}) + \sum_{i=1}^{n} \ln p(x_{i,d}|\lambda_{y_i,d}) \right).$$

Please separate your derivations as follows:

(a) Derive $\widehat{\pi}$ using the objective above.

(b) Derive $\widehat{\lambda}_{y,d}$ using the objective above, leaving y and d arbitrary in your notation.

Problem 2 – 30 points

In this problem you will implement the naive Bayes classifier derived in Problem 1 and the logistic regression algorithm. The data consists of examples of spam and non-spam emails, of which there are 4600 labeled examples. The feature vector x is a 54-dimensional vector extracted from the email and $y = 1$ indicates a spam email.[1]

In every experiment below, *randomly* partition the data into 10 groups and run the algorithm 10 different times so that each group is held out as a test set one time. The final result you show should be the cumulative result across these 10 groups.

(a) Implement the naive Bayes classifier described above. In a 2×2 table, write the number of times that you predicted a class y data point (ground truth) as a class y' data point (model prediction) in the (y, y')-th cell of the table, where y and y' can be either 0 or 1. There should be four values written in the table in your PDF. Next to your table, write the prediction accuracy—the sum of the diagonal divided by 4600. (The sum of all entries in the table should be 4600.)

(b) In one figure, show a stem plot (`stem()` in Matlab) of the 54 Poisson parameters for each class averaged across the 10 runs. (This average is only used for plotting purposes on this homework. In practice you would relearn these parameters using the entire data set to find their final values.) Use the README file to make an observation about dimensions 16 and 52.

Next run logistic regression on the same data set. ***Set every*** $y_i = 0$ ***to*** $y_i = -1$ ***for this part. Also, be sure to add a dimension equal to*** $+1$ ***to each data point.***

(c) Implement the steepest ascent algorithm discussed in class. Use a step size $\eta = \frac{0.01}{4600}$. Run your algorithm for 1,000 iterations and plot the logistic regression objective training function $\mathcal{L}$ per iteration for each of the 10 training runs. Plot this in the same figure.

(d) Finally, implement an algorithm called "Newton's method" for logistic regression as follows: At iteration t, approximate the function

$$\mathcal{L}(w) \approx \mathcal{L}'(w) \equiv \mathcal{L}(w_t) + (w - w_t)^T \nabla \mathcal{L}(w_t) + \frac{1}{2}(w - w_t)^T \nabla^2 \mathcal{L}(w_t)(w - w_t)$$

Then set $w_{t+1} = \arg\max_w \mathcal{L}'(w)$. Derive the update for w_{t+1} for the logistic regression problem and implement and run this algorithm. Plot the objective function $\mathcal{L}$ on the training data as a function of $t = 1, \ldots, 100$ for each of the 10 training runs. Plot this in the same figure.

(e) In a 2×2 table, show the testing results using Newton's method in the same way as shown in Problem 2(a).

[1] I've preprocessed the data already. Please use the data as-is for all of problem 2. More information about the meanings of the 54 dimensions of the data is provided in two accompanying files.

Problem 3 – 25 points

In this problem you will implement the Gaussian process model for regression. You will use the same data used for homework 1 to do this, which is again provided in the data zip file for this homework. Recall that the Gaussian process treats a set of N observations $(x_1, y_1), \ldots, (x_N, y_N)$, with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, as being generated from a multivariate Gaussian distribution as follows,

$$y \sim Normal(0, \sigma^2 I + K), \quad K_{ij} = K(x_i, x_j) \quad \Big(\text{use: } \exp\Big\{-\frac{1}{b}\|x_i - x_j\|^2\Big\}\Big).$$

Here, y is an N-dimensional vector of outputs and K is an $N \times N$ kernel matrix. For this problem use the Gaussian kernel indicated above. In the lecture slides, we discuss making predictions for a new y' given x', which was Gaussian with mean $\mu(x')$ and variance $\Sigma(x')$. The equations are shown in the slides.

There are two parameters that need to be set for this model as given above, σ^2 and b.

a) Write code to implement the Gaussian process and to make predictions on test data. For $b \in \{5, 7, 9, 11, 13, 15\}$ and $\sigma^2 \in \{.1, .2, .3, .4, .5, .6, .7, .8, .9, 1\}$—so 60 total pairs (b, σ^2)—calculate the RMSE on the 42 test points as you did in the first homework. Use the mean of the Gaussian process at the test point as your prediction. Show your results in a table.

b) Which value was the best and how does this compare with the first homework? What might be a drawback of using the approach in this homework compared with homework 1?

c) To better understand what the Gaussian process is doing through visualization, re-run the algorithm by using *only* the 4th dimension of x_i (car weight). Set $b = 5$ and $\sigma^2 = 2$. Show a scatter plot of the data (x[4] versus y for each point). Also, plot as a solid line the predictive mean of the Gaussian process at each point *in the training set*. You can think of this problem as asking you to create a test set by duplicating x_i[4] for each i in the training set and then to predict that test set.

Problem 1 – 20 points

In this problem you will derive a naive Bayes classifier. For a labeled set of data $(y_1, x_1), \ldots, (y_n, x_n)$, where for this problem $y \in \{0, 1\}$ and x is a D-dimensional vector of counts, the Bayes classifier observes a new x_0 and predicts y_0 as

$$y_0 \;=\; \arg\max_y \; p(y_0 = y|\pi) \prod_{d=1}^{D} p(x_{0,d}|\lambda_{y,d}).$$

The distribution $p(y_0 = y|\pi) = \text{Bernoulli}(y|\pi)$. What is "naive" about this classifier is the assumption that all D dimensions of x are independent. Assume that each dimension of x is Poisson distributed with a Gamma prior. The full generative process is

$$\text{Data: } \; y_i \overset{iid}{\sim} \text{Bern}(\pi), \quad x_{i,d}|y_i \sim \text{Pois}(\lambda_{y_i,d}), \; d = 1, \ldots, D \qquad \text{Prior: } \; \lambda_{y,d} \overset{iid}{\sim} \text{Gamma}(2, 1)$$

Derive the solution for π and each $\lambda_{y,d}$ by maximizing

$$\widehat{\pi}, \widehat{\lambda}_{0,1:D}, \widehat{\lambda}_{1,1:D} \;=\; \arg \max_{\pi, \widehat{\lambda}_{0,1:D}, \widehat{\lambda}_{1,1:D}} \sum_{i=1}^{n} \ln p(y_i|\pi) + \sum_{d=1}^{D} \left(\ln p(\lambda_{0,d}) + \ln p(\lambda_{1,d}) + \sum_{i=1}^{n} \ln p(x_{i,d}|\lambda_{y_i,d}) \right).$$

Please separate your derivations as follows:

(a) Derive $\widehat{\pi}$ using the objective above.

(b) Derive $\widehat{\lambda}_{y,d}$ using the objective above, leaving y and d arbitrary in your notation.

(a) $\hat{\pi}_{MAP} = \arg\max_{\pi} \sum_{i=1}^{n} \ln \text{Bern}(\pi)$

$$+ \sum_{d=1}^{D} \left(\ln \text{Gamma}(2,1) + \ln \text{Gamma}(2,1) + \sum_{i=1}^{n} \ln \text{Pois}(\lambda_{y_i,d}) \right)$$

$$= \arg\max_{\pi} \sum_{i=1}^{n} \ln \pi^{y_i} \cdot (1-\pi)^{(1-y_i)}$$

$$= \arg\max_{\pi} \sum_{i=1}^{n} \ln y_i \pi + \sum_{i=1}^{n} \ln (1-y_i)(1-\pi)$$

$$\frac{\partial L}{\partial \pi} = \sum_{i=1}^{n} \frac{y_i}{\pi} + \sum_{i=1}^{n} \frac{1-y_i}{1-\pi} = 0$$

$$\sum_{i=1}^{n} \frac{y_i - \pi}{\pi(1-\pi)} = 0$$

$$\sum_{i=1}^{n} y_i - n\pi = 0$$

$$\pi = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$\widehat{\pi}, \widehat{\lambda}_{0,1:D}, \widehat{\lambda}_{1,1:D} = \arg \max_{\pi, \widehat{\lambda}_{0,1:D}, \widehat{\lambda}_{1,1:D}} \sum_{i=1}^{n} \ln p(y_i|\pi) + \sum_{d=1}^{D} \left(\ln p(\lambda_{0,d}) + \ln p(\lambda_{1,d}) + \sum_{i=1}^{n} \ln p(x_{i,d}|\lambda_{y_i,d}) \right).$$

(b) Derive $\widehat{\lambda}_{y,d}$ using the objective above, leaving y and d arbitrary in your notation.

$\lambda_{0d} e^{-\lambda_{0d}}$

$$\hat{\lambda}_{MAP} = \operatorname{argmax}_{\lambda} \sum_{i=1}^{n} \ln \text{Bern}(\pi) + \sum_{d=1}^{D} (\ln \text{Gamma}(2,1) + \ln \text{Gamma}(2,1) + \sum_{i=1}^{n} \ln \text{Pois}(\lambda_{y_i,d}))$$

$$= \cdots$$

$$+ \sum_{d=1}^{D} (\ln \lambda e^{-\lambda_0} + \ln \lambda_1 e^{-\lambda_1} + \sum_{i=1}^{n} \ln \frac{\lambda_{y_i}^{x_{i,d}} e^{-\lambda_{y_i}}}{x_{i,d}!})$$

$$= \sum_{d=1}^{D} (\ln \lambda_{0,d} - \lambda_{0d} + \ln \lambda_{1,d} - \lambda_{1,d} + \sum_{i=1}^{n} (x_{i,d} \ln \lambda_{y_i,d} - \lambda_{y_i,d} - \ln x_{i,d}!))$$

Let $\frac{dl}{d\lambda_{y,d}} = 0$

$$\sum_{d=1}^{D} \left(\frac{1}{\lambda_{y,d}} - 1 + \frac{1}{\lambda_{y,d}} \sum_{i=1}^{n} x_{i,d} - \sum_{i=1, j=0}^{n} \mathbb{1}(y_i = j) \right) = 0$$

$$\frac{1}{\lambda_{y,d}} (1 + \sum_{i=1}^{n} x_{i,d}) = 1 + \sum_{i=1}^{n} \sum_{j=0} \mathbb{1}(y_i = j)$$

$$\lambda_{y,d} = \frac{1 + \sum_{i=1}^{n} x_{i,d}}{1 + \sum_{i=1}^{n} \sum_{j=0} \mathbb{1}(y_i = j)}$$

Problem 2 – 30 points

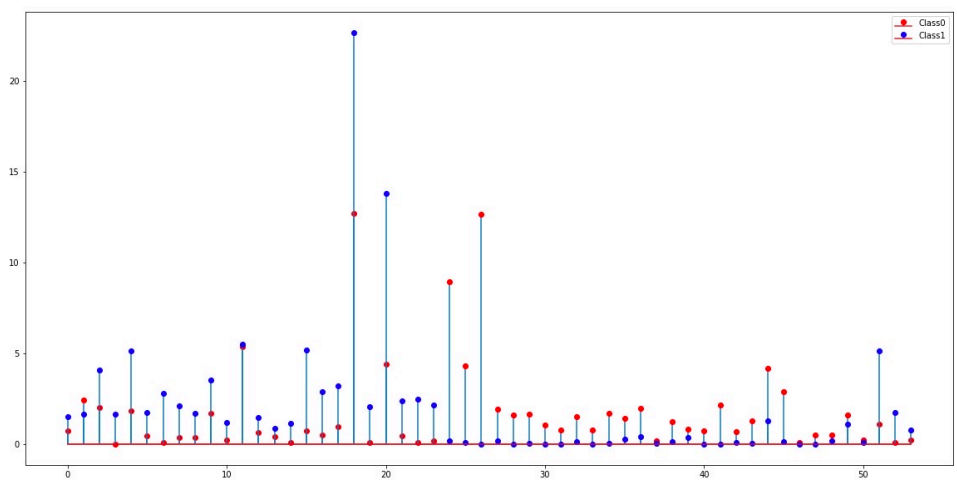
In this problem you will implement the naive Bayes classifier derived in Problem 1 and the logistic regression algorithm. The data consists of examples of spam and non-spam emails, of which there are 4600 labeled examples. The feature vector x is a 54-dimensional vector extracted from the email and $y = 1$ indicates a spam email.[1]

In every experiment below, *randomly* partition the data into 10 groups and run the algorithm 10 different times so that each group is held out as a test set one time. The final result you show should be the cumulative result across these 10 groups.

(a) Implement the naive Bayes classifier described above. In a 2×2 table, write the number of times that you predicted a class y data point (ground truth) as a class y' data point (model prediction) in the (y, y')-th cell of the table, where y and y' can be either 0 or 1. There should be four values written in the table in your PDF. Next to your table, write the prediction accuracy—the sum of the diagonal divided by 4600. (The sum of all entries in the table should be 4600.)

```
------------------------------------
              Actual 1      Actual 0
 Predicted 1     1714            490
 Predicted 0       99           2297
------------------------------------
Accuracy:  0.8719565217391304
```

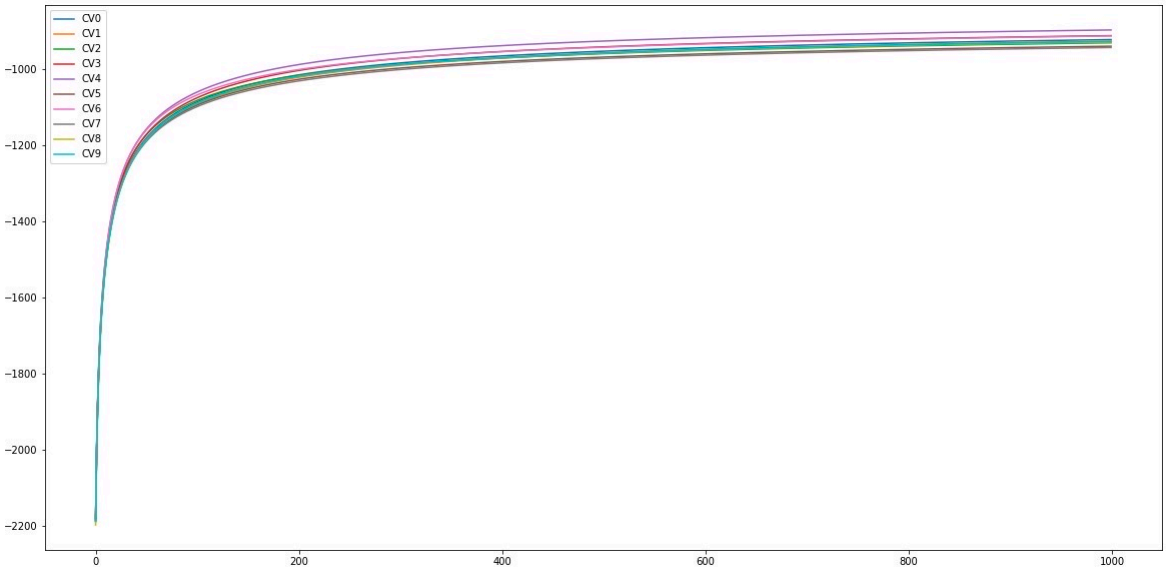
(b) In one figure, show a stem plot (`stem()` in Matlab) of the 54 Poisson parameters for each class averaged across the 10 runs. (This average is only used for plotting purposes on this homework. In practice you would relearn these parameters using the entire data set to find their final values.) Use the README file to make an observation about dimensions 16 and 52.



In feature 16 and 52 both have a higher value of lambda in class 1 and much lower value of lambda in class0, which means that the email contains "free" and "!" has a higher chance to be classified as a spam email.

Next run logistic regression on the same data set. ***Set every*** $y_i = 0$ ***to*** $y_i = -1$ ***for this part. Also, be sure to add a dimension equal to*** $+1$ ***to each data point.***

(c) Implement the steepest ascent algorithm discussed in class. Use a step size $\eta = \frac{0.01}{4600}$. Run your algorithm for 1,000 iterations and plot the logistic regression objective training function $\mathcal{L}$ per iteration for each of the 10 training runs. Plot this in the same figure.



(d) Finally, implement an algorithm called "Newton's method" for logistic regression as follows: At iteration t, approximate the function

$$\mathcal{L}(w) \approx \mathcal{L}'(w) \equiv \mathcal{L}(w_t) + (w - w_t)^T \nabla \mathcal{L}(w_t) + \frac{1}{2}(w - w_t)^T \nabla^2 \mathcal{L}(w_t)(w - w_t)$$

Then set $w_{t+1} = \arg\max_w \mathcal{L}'(w)$. Derive the update for w_{t+1} for the logistic regression problem and implement and run this algorithm. Plot the objective function $\mathcal{L}$ on the training data as a function of $t = 1, \ldots, 100$ for each of the 10 training runs. Plot this in the same figure.

$$\frac{\partial \mathcal{L}'(w)}{\partial w} = \nabla \mathcal{L}(w_t) + (w - w_t)\nabla^2 \mathcal{L}(w_t) = 0$$

$$w_{t+1} = w_t - \nabla \mathcal{L}(w_t) \cdot \nabla^2 \mathcal{L}(w_t)^{-1}$$

Gradient: $\nabla \mathcal{L}(w_t)$

$$\mathcal{L} = \prod_{i=1}^{n} \sigma_i(y_i w) \Rightarrow \nabla \mathcal{L}(w_t) = \frac{\partial \mathcal{L}}{\partial w}$$

$$= \sum_{i=1}^{n} \ln \sigma_i(y_i w) \qquad = \sum_{i=1}^{n} \frac{1}{\sigma_i(y_i w)} \sigma_i(y_i w)(1 - \sigma_i(y_i w)) y_i x_i$$

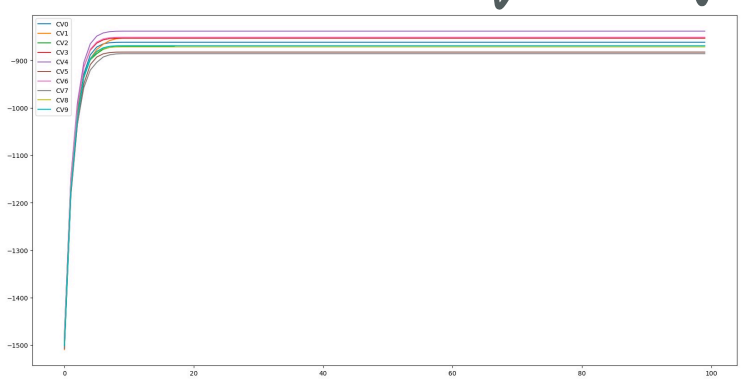
$$- \sigma(y_i w) \qquad = \sum_{i=1}^{n} (1 - \sigma(y_i w)) y_i x_i$$

$\nabla^2 \mathcal{L}(w_t)$:

$$\mathcal{L} = \sum_{i=1}^{n} \ln \sigma_i(y_i w) \Rightarrow \nabla^2 \mathcal{L}(w_t) = \frac{\partial \mathcal{L}(w)}{\partial w \partial w^T}$$

$$= -x_i^T x_i [\sigma(y_i w)(1 - \sigma(y_i w))] y_i^2$$

$$w_{t+1} = w_t + \frac{\sum_{i=1}^{n} (1 - \sigma(y_i w)) y_i x_i}{-x_i^T x_i [\sigma(y_i w)(1 - \sigma(y_i w))]}$$



(e) In a 2×2 table, show the testing results using Newton's method in the same way as shown in Problem 2(a).

```
Accuracy:  0.8882608695652173
-------------------------------------
               Actual 1      Actual 0
 Predicted 1     1430            131
 Predicted 0      383             2656
-------------------------------------
```

Problem 3 – 25 points

In this problem you will implement the Gaussian process model for regression. You will use the same data used for homework 1 to do this, which is again provided in the data zip file for this homework. Recall that the Gaussian process treats a set of N observations $(x_1, y_1), \ldots, (x_N, y_N)$, with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, as being generated from a multivariate Gaussian distribution as follows,

$$y \sim Normal(0, \sigma^2 I + K), \quad K_{ij} = K(x_i, x_j) \quad \Big(\text{use: } \exp\Big\{-\frac{1}{b}\|x_i - x_j\|^2\Big\}\Big).$$

Here, y is an N-dimensional vector of outputs and K is an $N \times N$ kernel matrix. For this problem use the Gaussian kernel indicated above. In the lecture slides, we discuss making predictions for a new y' given x', which was Gaussian with mean $\mu(x')$ and variance $\Sigma(x')$. The equations are shown in the slides.

There are two parameters that need to be set for this model as given above, σ^2 and b.

a) Write code to implement the Gaussian process and to make predictions on test data. For $b \in \{5, 7, 9, 11, 13, 15\}$ and $\sigma^2 \in \{.1, .2, .3, .4, .5, .6, .7, .8, .9, 1\}$—so 60 total pairs (b, σ^2)—calculate the RMSE on the 42 test points as you did in the first homework. Use the mean of the Gaussian process at the test point as your prediction. Show your results in a table.

	b/sigma	5	7	9	11	13	15
0	0.1	1.966278	1.920165	1.897651	1.890509	1.895850	1.909605
1	0.2	1.933137	1.904878	1.902521	1.914983	1.935588	1.959551
2	0.3	1.923422	1.908082	1.917650	1.938851	1.964600	1.990806
3	0.4	1.922200	1.915904	1.932517	1.957938	1.985504	2.011918
4	0.5	1.924771	1.924806	1.945702	1.973218	2.001316	2.027372
5	0.6	1.929215	1.933704	1.957237	1.985766	2.013881	2.039467
6	0.7	1.934636	1.942256	1.967406	1.996377	2.024313	2.049465
7	0.8	1.940585	1.950382	1.976494	2.005605	2.033309	2.058107
8	0.9	1.946822	1.958095	1.984743	2.013838	2.041320	2.065847
9	1.0	1.953215	1.965440	1.992344	2.021347	2.048644	2.072978

b) Which value was the best and how does this compare with the first homework? What might be a drawback of using the approach in this homework compared with homework 1?

```
Best Parameter:  (11, 0.1)   RMSE:  1.890509034564076
```

The best RMSE yielded by this model is better than the one in HW1.

Drawback:

1. The Gaussian Process method has a higher computational complexity which might take more memory and training time.

2. The choice of a "good" kernel might be a challenge.

c) To better understand what the Gaussian process is doing through visualization, re-run the algorithm by using *only* the 4th dimension of x_i (car weight). Set $b = 5$ and $\sigma^2 = 2$. Show a scatter plot of the data ($x[4]$ versus y for each point). Also, plot as a solid line the predictive mean of the Gaussian process at each point *in the training set*. You can think of this problem as asking you to create a test set by duplicating $x_i[4]$ for each i in the training set and then to predict that test set.