Kate Lassiter

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.

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

$$y_0 = argmax \sum_{i=1}^{n} lnp(y_i|\pi) + \sum_{d=1}^{D} (lnp(\lambda_{0,d}) + lnp(\lambda_{1,d}) + \sum_{i=1}^{n} lnp(x_{i,d}|\lambda_{y_i,d}))$$

$$y_i \sim_{i.i.d} Bernoulli(\pi) \rightarrow p^k (1 - p)^{1-k} \rightarrow k \leq \{0, 1\}$$

Recall:

$$f(x_1, \ldots, x_n) = \pi^{x_1}(1 - \pi)^{1-x_1} \times \pi^{x_2}(1 - \pi)^{1-x_2} \times \ldots \times \pi^{x_n}(1 - \pi)^{1-x_n}$$

$$f(x_1, \ldots, x_n) = \pi^{\sum_{i=1}^{n} x_i} (1 - \pi)^{n - \sum_{i=1}^{n} x_i}$$

Thus:

$$y_0 = argmax \sum_{i=1}^{n} lnp(y_i|\pi) + \sum_{d=1}^{D} (lnp(\lambda_{0,d}) + lnp(\lambda_{1,d}) + \sum_{i=1}^{n} lnp(x_{i,d}|\lambda_{y_i,d}))$$

$$y_0 = argmax \sum_{i=1}^{n} lnp(y_i|\pi) + \ldots$$

$$y_0 = ln(\pi^{\sum_{i=1}^{n} y_i} (1 - \pi)^{n - \sum_{i=1}^{n} y_i})$$

$$y_0 = ln\pi^{\sum_{i=1}^{n} y_i} + ln(1 - \pi)^{n - \sum_{i=1}^{n} y_i}$$

$$y_0 = \sum_{i=1}^{n} y_i ln\pi + (n - \sum_{i=1}^{n} y_i) ln(1 - \pi)$$

$$\frac{d}{d\pi} y_0 = \frac{\sum_{i=1}^{n} y_i}{\pi} + \frac{(n - \sum_{i=1}^{n} y_i)}{(1-\pi)}$$

$$\frac{d}{d\pi} y_0 = 0$$

Kate Lassiter

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

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

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

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

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

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

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

$$y_0 = argmax \sum_{i=1}^{n} lnp(y_i|\pi) + \sum_{d=1}^{D} (lnp(\lambda_{0,d}) + lnp(\lambda_{1,d}) + \sum_{i=1}^{n} lnp(x_{i,d}|\lambda_{y_i,d}))$$

Recall:

$x_i \sim_{i.i.d} p(\lambda_{y_i})$

$$p(x|\lambda_{y_i}) = \frac{\lambda_{y_i}^{x}}{x!}e^{-\lambda} \rightarrow Poisson\, r.v$$

Thus:

$$p(x_1,..., x_n|\lambda_{y_i}) = \frac{\lambda_{y_1}^{x_1}}{x_1!}e^{-\lambda_{y_1}} \times \ldots \times \frac{\lambda_{y_n}^{x_n}}{x_n!}e^{-\lambda_{y_n}}$$

$$p(x_1,..., x_n|\lambda_{y_i}) = \frac{\lambda_{y_i}^{\sum_{i=1}^{n} x_i}}{x_1! \times \ldots \times x_n!}e^{-n\lambda_{y_i}}$$

$$lnp(x_1,..., x_n|\lambda_{y_i}) = ln\lambda_{y_i}^{\sum_{i=1}^{n} x_i} + lne^{-n\lambda_{y_i}} - ln\,(x_1! \times \ldots \times x_n!)$$

$$lnp(x_1,..., x_n|\lambda_{y_i}) = \sum_{i=1}^{n} x_i ln\lambda_{y_i} - n\lambda_{y_i} - Constant$$

$$lnp(x_1,..., x_n|\lambda_{y_i}) = \sum_{i=1}^{n} x_i ln\lambda_{y_i,d} - n\lambda_{y_i,d}$$

Recall:

$\lambda_{y_i,d} \sim_{i.i.d} Gamma(2, 1)$

$$p(\lambda) = \frac{\lambda^{a-1}e^{-b\lambda}b^{a}}{\Gamma(a)} \rightarrow Gamma\, r.v$$

$$p(\lambda) = \frac{\lambda^{2-1}e^{-1\lambda}1^{2}}{\Gamma(2)}$$

$$p(\lambda) = \frac{\lambda\, e^{-\lambda}}{(2-1)1!}$$

$$p(\lambda) = \lambda\, e^{-\lambda}$$

Thus:

$$p(\lambda_{y_i,d}) = \lambda_{y_i,d}e^{-\lambda_{y_i,d}}$$

Kate Lassiter

$$p(\lambda_{y_i,1},...,\lambda_{y_i,D}) = \lambda_{y_i,1}e^{-\lambda_{y_i,1}} \times \ ... \ \times \lambda_{y_i,D}e^{-\lambda_{y_i,D}}$$

$$p(\lambda_{y_i,1},...,\lambda_{y_i,D}) = d\lambda_{y_i}e^{-d\lambda_{y_i}}$$

$$lnp(\lambda_{y_i,1},...,\lambda_{y_i,D}) = lnd\lambda_{y_i} - d\lambda_{y_i}$$

$$\boldsymbol{\lambda} = argmax \sum_{i=1}^{n} lnp(y_i|\pi) + \sum_{d=1}^{D}(lnp(\lambda_{0,d}) + lnp(\lambda_{1,d}) + \sum_{i=1}^{n} lnp(x_{i,d}|\lambda_{y_i,d}))$$

$$\boldsymbol{\lambda} = \ ... \ + \sum_{d=1}^{D}(lnp(\lambda_{0,d}) + lnp(\lambda_{1,d}) + \sum_{i=1}^{n} lnp(x_{i,d}|\lambda_{y_i,d}))$$

$$\boldsymbol{\lambda} = \ ... \ + \sum_{d=1}^{D}(lnp(\lambda_{0,d}) + lnp(\lambda_{1,d}) + \sum_{i=1}^{n}(y_i lnp(x_{i,d}|\lambda_{y_i,d}) + (1-y_i)lnp(x_{i,d}|\lambda_{y_i,d})$$

$$ln\boldsymbol{\lambda} = \ ... \ + lnd\lambda_{0,\,d} - d\lambda_{0,d} + lnd\lambda_{1,d} - d\lambda_{1,d} + \sum_{i=1}^{n}(y_i x_i ln\lambda_{1,d} - n\lambda_{1,d} + (1-y_i)x_i ln\lambda_{0,d} - n\lambda_{0,d})$$

Solve for $\lambda_{1,d}$:

$$\frac{d}{d\lambda_{1,d}} ln\boldsymbol{\lambda} = \ ... \ + lnd\lambda_{1,d} - d\lambda_{1,d} + \sum_{i=1}^{n}(y_i x_i ln\lambda_{1,d} - n\lambda_{1,d})$$

$$\frac{d}{d\lambda_{1,d}} ln\boldsymbol{\lambda} = \frac{1}{\lambda_{1,d}} - 1 + \sum_{i=1}^{n}(y_i x_i \frac{1}{\lambda_{1,d}}) - \sum_{i=1}^{n} y_i$$

$$\frac{d}{d\lambda_{1,d}} ln\boldsymbol{\lambda} = 0$$

$$0 = \frac{1}{\lambda_{1,d}} - 1 + \sum_{i=1}^{n}(y_i x_i \frac{1}{\lambda_{1,d}}) - \sum_{i=1}^{n} y_i$$

$$1 + \sum_{i=1}^{n} y_i = \frac{1}{\lambda_{1,d}} + \frac{\sum_{i=1}^{n} y_i x_i}{\lambda_{1,d}}$$

$$1 + \sum_{i=1}^{n} y_i = \frac{\sum_{i=1}^{n} y_i x_i + 1}{\lambda_{1,d}}$$

$$\frac{1}{1+\sum_{i=1}^{n} y_i} = \frac{\lambda_{1,d}}{\sum_{i=1}^{n} y_i x_i + 1}$$

$$\boxed{\frac{\sum_{i=1}^{n} y_i x_i + 1}{1+\sum_{i=1}^{n} y_i} = \lambda_{1,d}}$$

Solve for $\lambda_{0,d}$:

$$\frac{d}{d\lambda_{0,d}} ln\boldsymbol{\lambda} = \ ... \ + lnd\lambda_{0,d} - d\lambda_{0,d} + \sum_{i=1}^{n}(1-y_i)x_i ln\lambda_{0,d} - n\lambda_{0,d})$$

$$\frac{d}{d\lambda_{1,d}} ln\boldsymbol{\lambda} = \frac{1}{\lambda_{0,d}} - 1 + \sum_{i=1}^{n}(1-y_i)x_i \frac{1}{\lambda_{0,d}}) - \sum_{i=1}^{n}(1-y_i)$$

$$\frac{d}{d\lambda_{1,d}} ln\boldsymbol{\lambda} = 0$$

$$0 = \frac{1}{\lambda_{0,d}} - 1 + \sum_{i=1}^{n}(1-y_i)x_i \frac{1}{\lambda_{0,d}}) - \sum_{i=1}^{n}(1-y_i)$$

$$1 + \sum_{i=1}^{n}(1-y_i) = \frac{1}{\lambda_{0,d}} + \frac{\sum_{i=1}^{n}(1-y_i)x_i}{\lambda_{0,d}}$$

$$1 + \sum_{i=1}^{n}(1-y_i) = \frac{\sum_{i=1}^{n}(1-y_i)x_i + 1}{\lambda_{0,d}}$$

Kate Lassiter

$$\frac{1}{1+\sum_{i=1}^{n}(1-y_i)} = \frac{\lambda_{0,d}}{\sum_{i=1}^{n}(1-y_i)x_i + 1}$$

$$\boxed{\frac{\sum_{i=1}^{n}(1-y_i)x_i + 1}{1+\sum_{i=1}^{n}(1-y_i)} = \lambda_{0,d}}$$

$$\boxed{\lambda_{y_i,d} = (1-y_i)\left(\frac{\sum_{i=1}^{n}(1-y_i)x_{i,d} + 1}{1+\sum_{i=1}^{n}(1-y_i)}\right) + y_i\left(\frac{\sum_{i=1}^{n}y_i x_{i,d} + 1}{1+\sum_{i=1}^{n}y_i}\right)}$$

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]

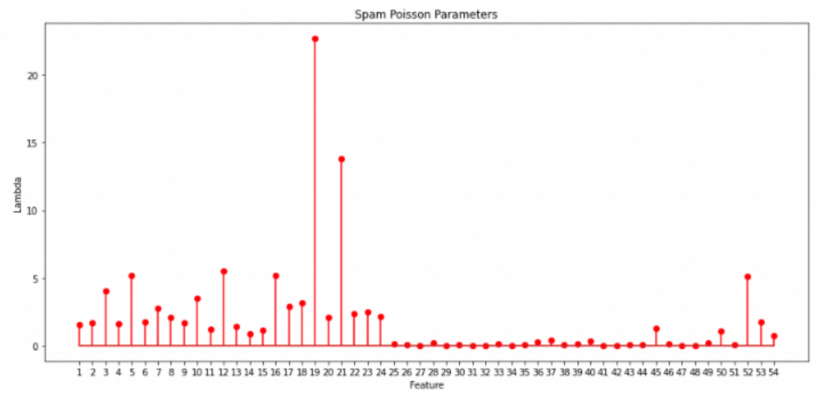
In every experiment below, *randomly* partition the data into 10 groups and run the algorithm 10 different times so that each group is held out as a test set one time. The final result you show should be the cumulative result across these 10 groups.

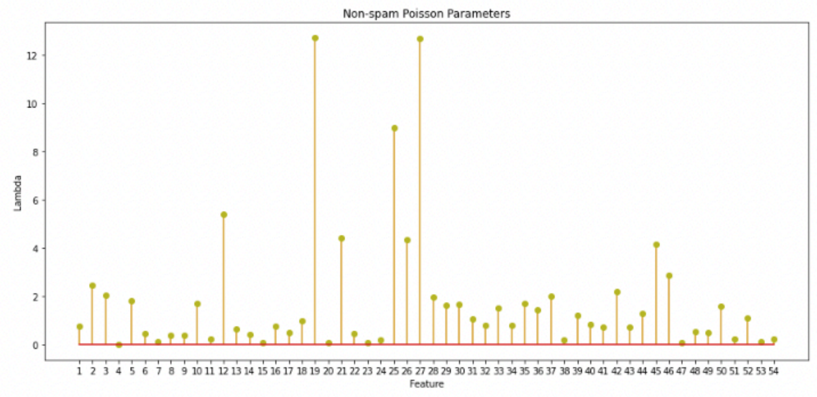
(a) Implement the naive Bayes classifier described above. In a 2×2 table, write the number of times that you predicted a class y data point (ground truth) as a class y' data point (model prediction) in the (y, y')-th cell of the table, where y and y' can be either 0 or 1. There should be four values 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.)

```
-  ----  ----
      0     1
0  2293   106
1   494  1707
-  ----  ----
prediction accuracy: 0.8695652173913043
```

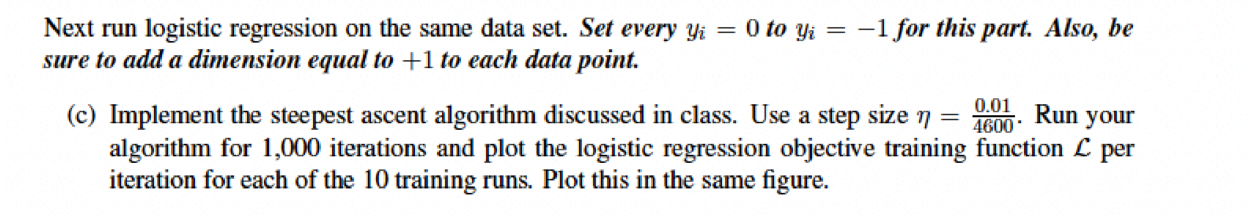
(b) In one figure, show a stem plot (`stem()` in Matlab) of the 54 Poisson parameters for each class averaged across the 10 runs. (This average is only used for plotting purposes on this homework. In practice you would relearn these parameters using the entire data set to find their final values.) Use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

Kate Lassiter



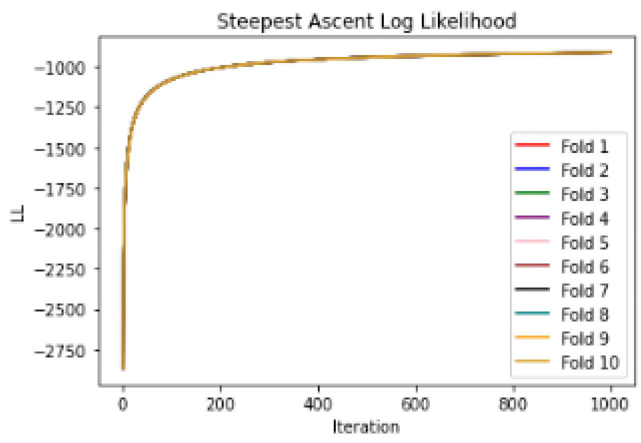


For dimensions 16 and 52, which correspond to the word "free" and character "!" respectively, these are noticeably higher for spam emails versus non-spam emails. The word free appears on average about five times for spam emails, corresponding to a lambda averaged around five. For non-spam emails, this average is about once per email. The same results can be seen for the "!" character, with an average around five for spam emails as opposed to around one for non-spam emails.

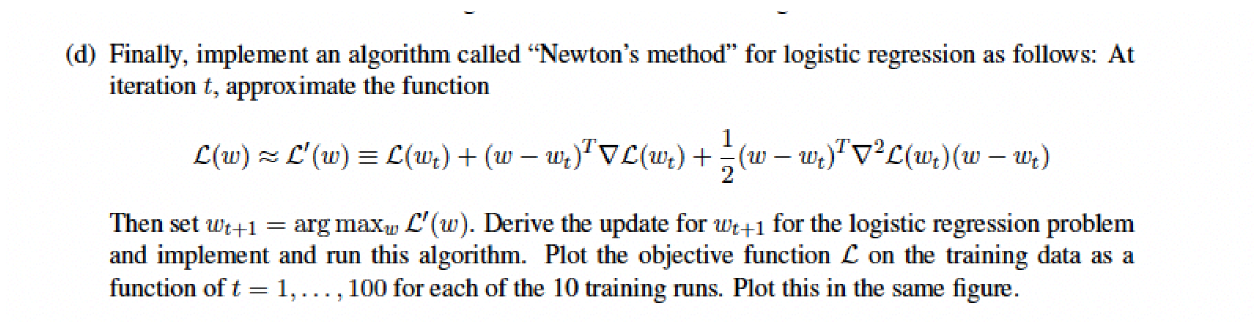


Next run logistic regression on the same data set. ***Set every $y_i = 0$ to $y_i = -1$ for this part. Also, be sure to add a dimension equal to $+1$ to each data point.***

(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.



Kate Lassiter



(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.

Derive update:

$f(w) \approx f(z) + (w - z)^T \nabla f(z) + \frac{1}{2}(w - z)^T (\nabla^2 f(z))(w - z)$

$\frac{d}{dw} f(w) \approx \nabla f(z) + (\nabla^2 f(z))(w - z)$

$\frac{d}{dw} f(w) \approx 0$

$0 \approx \nabla f(z) + (\nabla^2 f(z))(w - z)$

$0 \approx \nabla f(z) + (\nabla^2 f(z))w - (\nabla^2 f(z))z$

$-(\nabla^2 f(z))w \approx \nabla f(z) - (\nabla^2 f(z))z$

$\frac{-(\nabla^2 f(z))w}{(\nabla^2 f(z))} \approx \frac{\nabla f(z) - (\nabla^2 f(z))z}{(\nabla^2 f(z))}$

$-w \approx \frac{\nabla f(z)}{(\nabla^2 f(z))} - z$

$w \approx -\frac{\nabla f(z)}{(\nabla^2 f(z))} + z$

Matrix notation:

$w \approx -\nabla f(z)(\nabla^2 f(z))^{-1} + z$

Update:

$$\boxed{w_{t+1} \approx -\nabla f(w_t)(\nabla^2 f(w_t))^{-1} + w_t}$$

Log likelihood:

$p(y_1, \ldots, y_n | x_1, \ldots, x_n, w) = \prod_{t=1}^{n} \sigma_i(y_i \times w)$

Recall:

$\sigma_i(y_i \times w) = \frac{e^{y_i x_i^T w}}{1 + e^{y_i x_i^T w}}$

Thus:

$p(y_1, \ldots, y_n | x_1, \ldots, x_n, w) = \prod_{i=1}^{n} \frac{e^{y_i x_i^T w}}{1 + e^{y_i x_i^T w}}$

$\ln p(y_1, \ldots, y_n | x_1, \ldots, x_n, w) = \sum_{i=1}^{n} (\ln e^{y_i x_i^T w} - \ln(1 + e^{y_i x_i^T w}))$

$\ln p(y_1, \ldots, y_n | x_1, \ldots, x_n, w) = \sum_{i=1}^{n} (y_i x_i^T w - \ln(1 + e^{y_i x_i^T w}))$

Recall:

$$\frac{d}{dx}[f(g(x))] = f'(g(x))g'(x)$$

$$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{(g(x))^2}$$

Thus:

$$\nabla f(w_t) = \sum_{i=1}^{n} \left(y_i x_i^T - \left(\frac{1}{1+e^{y_i x_i^T w}}\right) y_i x_i^T e^{y_i x_i^T w}\right)$$

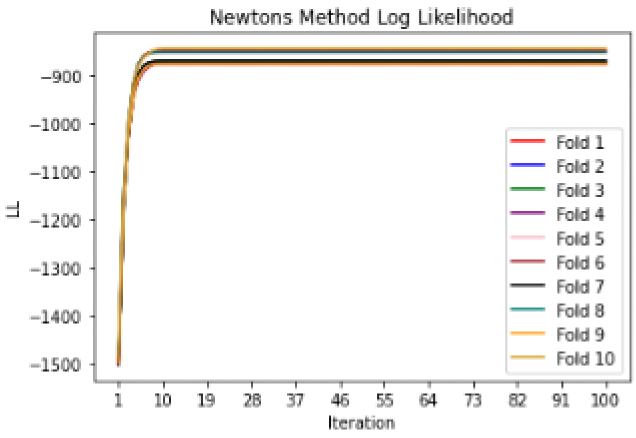
$$\nabla f(w_t) = \sum_{i=1}^{n} \left(y_i x_i^T - \frac{y_i x_i^T e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}\right)$$

$$\nabla^2 f(w_t) = \sum_{i=1}^{n} \left(\frac{(1+e^{y_i x_i^T w})(y_i x_i^T \bullet y_i x_i^T \bullet e^{y_i x_i^T w}) - (y_i x_i^T e^{y_i x_i^T w})(y_i x_i^T e^{y_i x_i^T w})}{(1+e^{y_i x_i^T w})^2}\right)$$

$$\nabla^2 f(w_t) = \sum_{i=1}^{n} \left(\frac{(1+e^{y_i x_i^T w})(e^{y_i x_i^T w}) - (e^{y_i x_i^T w})(e^{y_i x_i^T w})}{(1+e^{y_i x_i^T w})^2}\right)(y_i x_i^T \bullet y_i x_i^T)$$

$$\nabla^2 f(w_t) = \sum_{i=1}^{n} \left(\frac{e^{y_i x_i^T w} + (e^{y_i x_i^T w})^2 - (e^{y_i x_i^T w})^2}{(1+e^{y_i x_i^T w})^2}\right)(y_i x_i^T \bullet y_i x_i^T)$$

$$\boxed{\nabla^2 f(w_t) = \sum_{i=1}^{n} \left(\frac{e^{y_i x_i^T w}}{(1+e^{y_i x_i^T w})^2}\right)(y_i x_i^T \bullet y_i x_i^T)}$$



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

```
-  ----  ----
      0     1
0  2646   211
1   141  1602
-  ----  ----
prediction accuracy: 0.9234782608695652
```

Kate Lassiter

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 $(x_1, y_1), \dots, (x_N, y_N)$, 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.

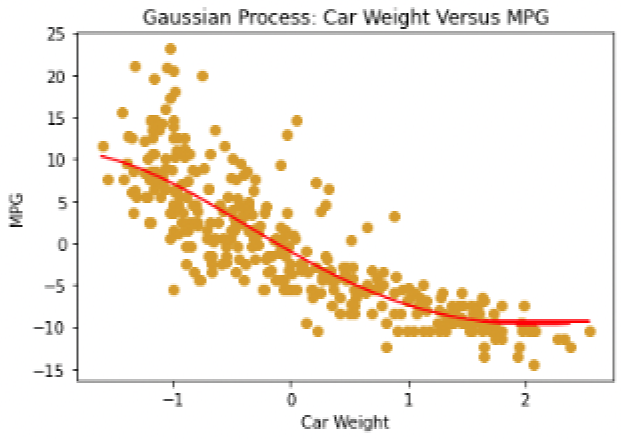
```
           5        7        9       11       13       15
---  -------  -------  -------  -------  -------  -------
0.1  1.96628  1.92017  1.89765  1.89051  1.89585  1.90961
0.2  1.93314  1.90488  1.90252  1.91498  1.93559  1.95955
0.3  1.92342  1.90808  1.91765  1.93885  1.9646   1.99081
0.4  1.9222   1.9159   1.93252  1.95794  1.9855   2.01192
0.5  1.92477  1.92481  1.9457   1.97322  2.00132  2.02737
0.6  1.92921  1.9337   1.95724  1.98577  2.01388  2.03947
0.7  1.93464  1.94226  1.96741  1.99638  2.02431  2.04947
0.8  1.94059  1.95038  1.97649  2.00561  2.03331  2.05811
0.9  1.94682  1.9581   1.98474  2.01384  2.04132  2.06585
1    1.95321  1.96544  1.99234  2.02135  2.04864  2.07298
```

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?

The best value found was variance=0.1 and b=11 for the Gaussian process, which resulted in a root mean square of 1.890509. The testing error for this process was much less that of the last homework, where standard linear regression, ridge regression, as well as regularized polynomial regression's RMSE remained above 2. While this Gaussian process has a better performance, much of the interpretability of the regression model is lost by using this approach. It becomes less clear why individual parameters are chosen and what the kernel functions mean, and model explainability is often key in presenting machine learning results.

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.

Kate Lassiter


Gaussian Process: Car Weight Versus MPG
MPG
Car Weight