CSCI-GA.2565-001 Machine Learning: Homework 1

Due 5pm EST, February 25, 2022 on Gradescope

We encourage LaTeX-typeset submissions but will accept quality scans of hand-written pages.

1 Regression

Consider a data generating process where $\mathbf{x} \in \mathbb{R}^D$ is drawn from $p(\mathbf{x})$, $w_1^{true} \in \mathbb{R}$, $\epsilon_y \sim \mathcal{N}(0,1)$, and $y = w_1^{true} x_1 + \epsilon_y$. We study the consequences of modeling y with all features $\mathbf{x}$ instead of just (correctly) using x_1 and how this affects estimation of w_1^{true}.

Let $\widehat{w_1}$ be the estimate of w_1^{true} using only x_1 and let $\widehat{w_1^{all}}$ be the estimate of w_1^{true} when using all of $\mathbf{x}$. In this problem we are interested in studying the relationships between $\mathbb{E}[\widehat{w_1^{all}}]$ and $\mathbb{E}[\widehat{w_1}]$, and $\text{Var}[\widehat{w_1^{all}}]$ and $\text{Var}[\widehat{w_1}]$. Rather than study these quantities analytically we will run simulations with PyTorch.

(A) Pick a value of w_1^{true} (e.g. with `torch.randn(1)`). Set $\sigma^2 = 1$. Set $N = 50$.

(B) Given N, D, and c, write a function to generate N samples. First generate $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$ where μ is a vector of D zeros and Σ is covariance matrix with all of the diagonal entries equal to σ^2 and all off-diagonal entries equal to c. Generate y as per the description above, which involves drawing N samples of $\mathcal{N}(0,1)$ noise ϵ_Y. Recall that y only depends on the first feature. Return (X, y).

(C) For a given (X, y), use least squares solutions to compute $\widehat{w_1^{all}}$ (using all features and returning the first coefficient) and $\widehat{w_1}$ (using only one feature).

(D) Write a function that does the previous step for $T = 100$ trials where each trial uses a new dataset (but same w_1^{true}). Save all T results of each estimator in a list and return the mean and standard deviation of this list for each estimator.

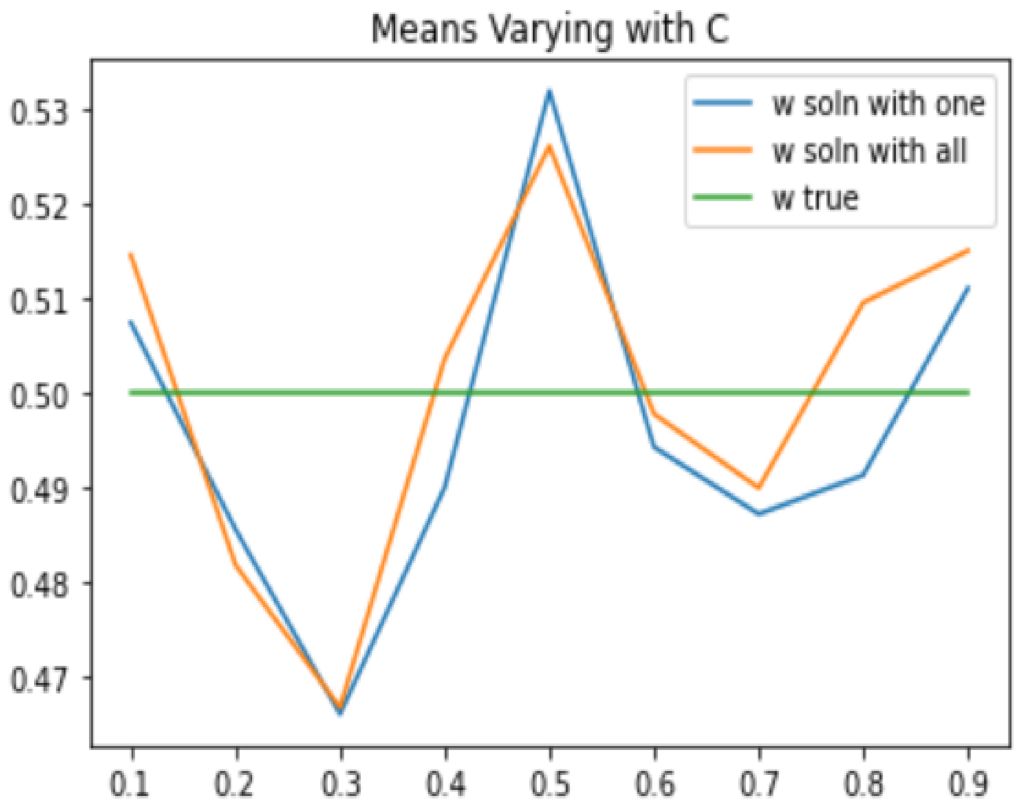
(E) Do the previous step for each value of $c \in \{0.1, 0.2, 0.3, \ldots, 0.9\}$. Plot the means and standard deviations as a function of c. Use one plot for both mean curves and one plot for both standard deviation curves.

(F) Finally, do all of the above varying $D \in \{2, 4, 8, 16, 32\}$. This means there will be two plots (each containing two curves) for each of the 5 choices of D.

(G) What do you observe with respect to c and D? What can explain this? Submit only your response to this last sub-question as well as a few (not all) of your generated plots that help you communicate your answer.

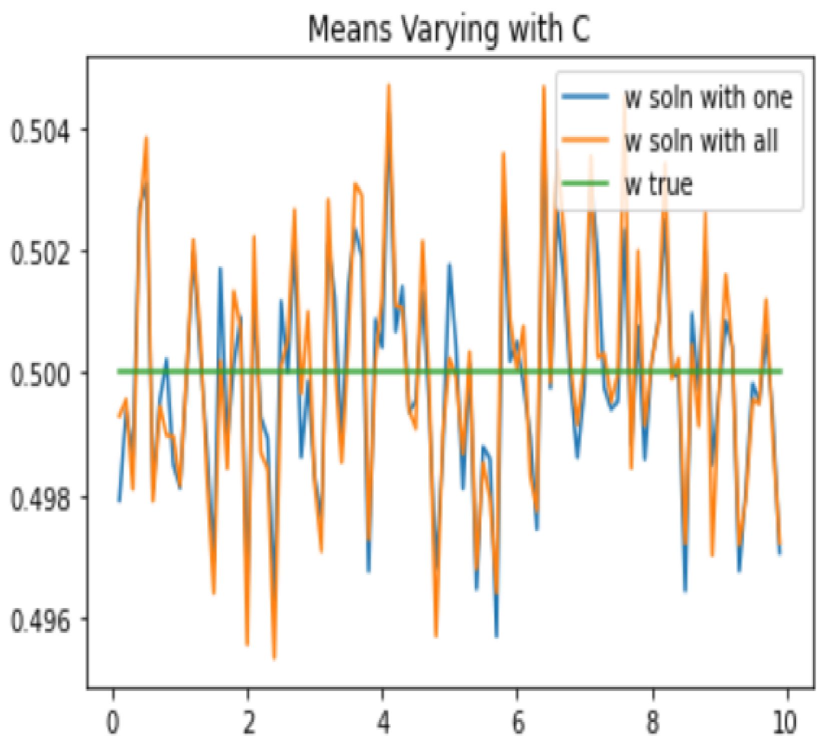
Solution. you can write your solutions here.

```
<matplotlib.legend.Legend at 0x1a9830c8d00>
```

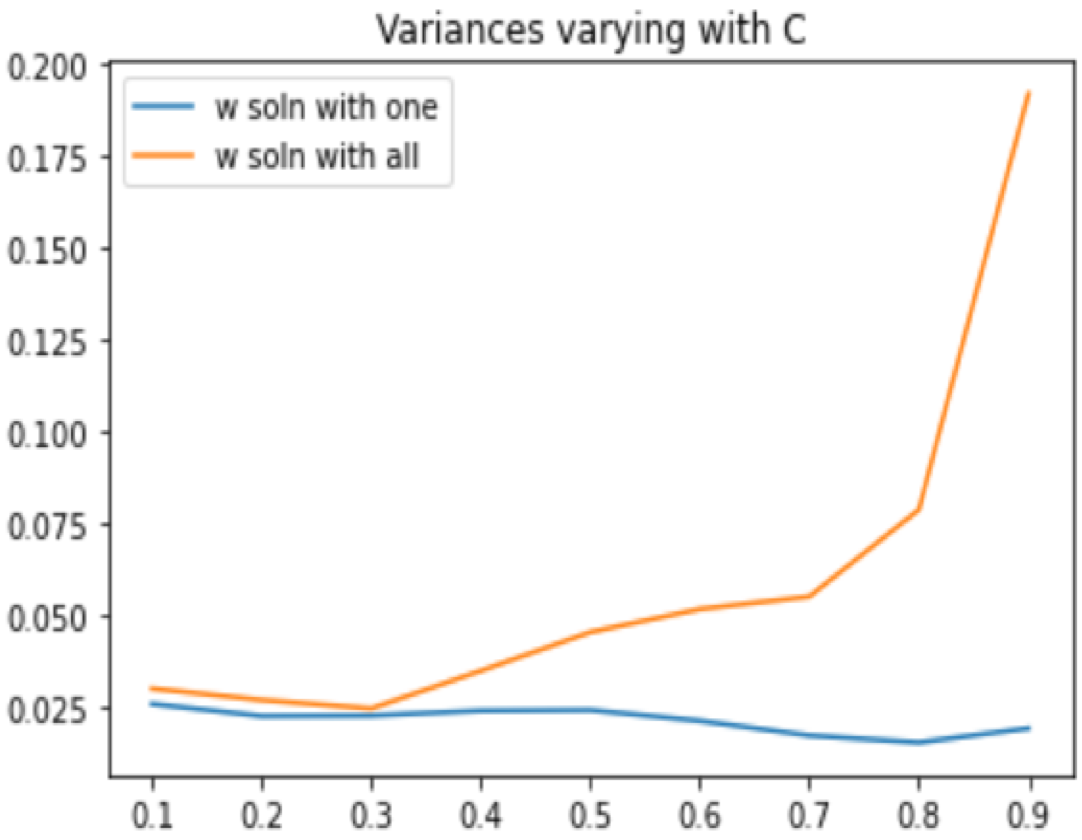


The upper graph 2.1(a) shows the varying means of w_1 for c going from 0.1 to 0.9 for sigma squared as 1.

```
Out[185]: <matplotlib.legend.Legend at 0x1a9ebe2a820>
```

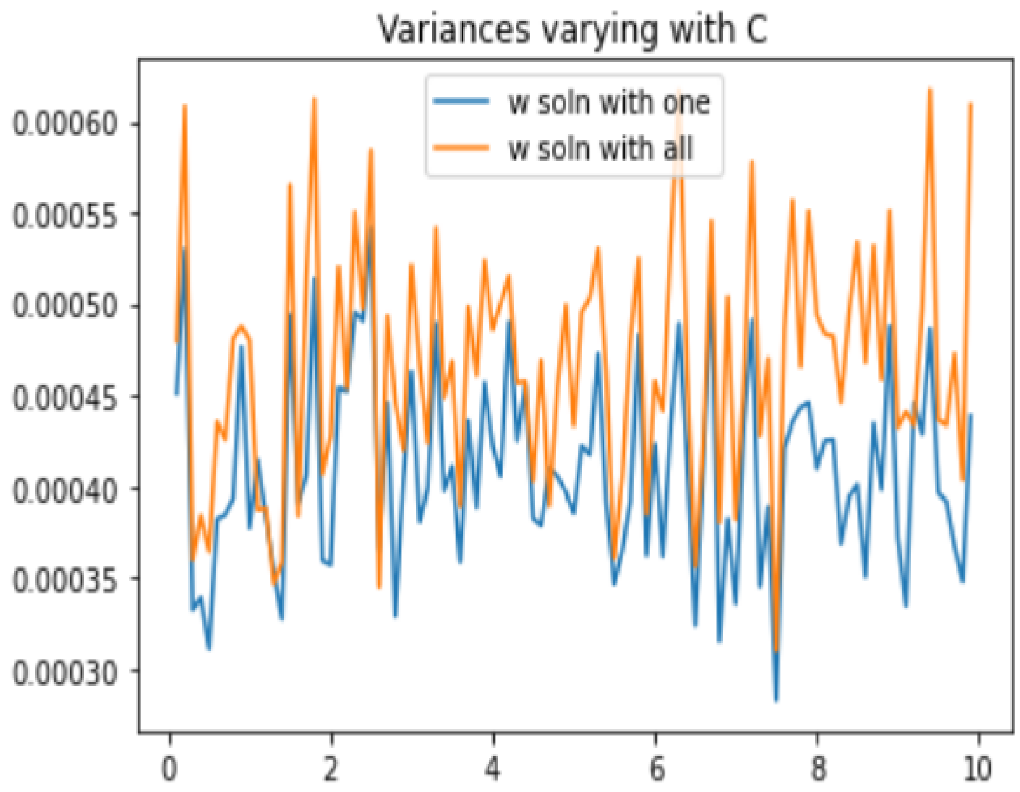


The upper graph 2.1(b) shows for the varying w_1 means c going from 0.1 to 9.9(increase in the range C for better grasping, similarly sigma squared value increased here for explanatory purposes in comparsion to the previous graph where sigma squared was 1).



The upper graph 2.2(a) shows for C going from 0.1 to 0.9, how the variance values of w_1 are varying with C.

```
<matplotlib.legend.Legend at 0x1a9818226a0>
```



The upper graph 2.2(b) shows for C going from 0.1 to 9.9(increase in the range for C and in the self variance sigma squared for explanatory purposes), how the variance values of w_1 are varying with C. 2.4 will show us the broader trend.

Our true value of w_1 is taken as 0.5. Here are the averages of the means as based estimations for w_1 in various trials over different c values.

```
sum(w1_one_means)/len(w1_one_means)
```

```
tensor(0.5000)
```

```
sum(w1_all_means)/len(w1_all_means)
```

```
tensor(0.5001)
```

Fig 2.3

For any trial we have a data set, we will get the value of w_1 from the the model with all features of x and the model with one feature of x. For T trails, we will get an expectation over w_1 values. We can find the variance of the w_1 values. Thinking of w as a function of the data X,y, we can write the following for our estimator $\tilde{w}_1$ considering w_1* as the true value, the bias and the variance in the estimate of w_1,

$$Bias(w_1) = E_{P(D)}[\tilde{w_1} - w_1^*]$$

Where,

$$\tilde{w_1} = f(X;y)$$

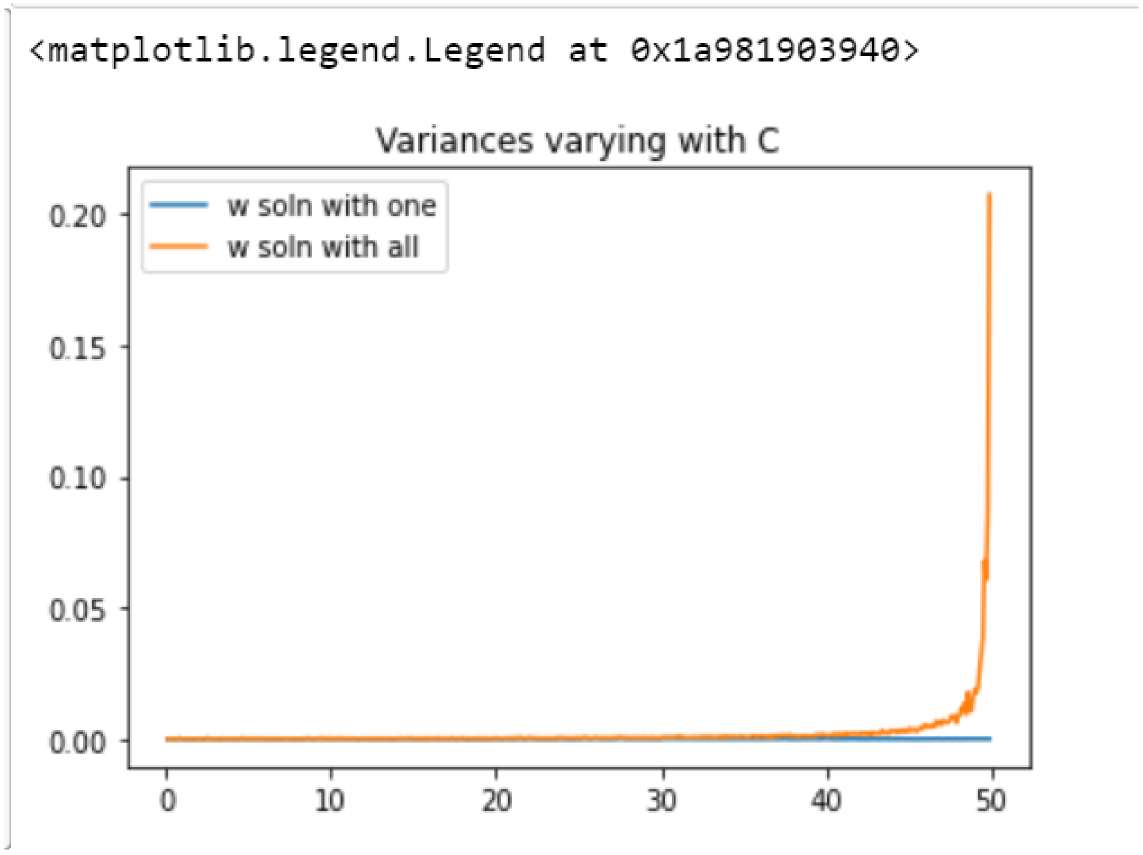
$$Variance(w_1) = E_{P(D)}[\tilde{w_1} - E_{P(D)}[\tilde{w_1}]]^2$$

Where,

$$\tilde{w_1} = f(X;y)$$

Our model M_1 uses a single feature and our model M_2 uses all the features in X. Model 2 therefore has more adaptive parameters. The corresponding model is of higher complexity. In our expectation, for model one, we will marginalize over other variables in X apart from x_1 and so we essentially have a univariate distribution and the corresponding expectation. Either the expectation for the bias or for the variance.

Consider the case for increasing C, for model 2, it has the structure set up to be capable of representing various functions which account for the X,y relationship in the data, including our function. We see that the expected value of the expected value of w_1 values learned through the trials tends to our true value of 0.5. The model 2 has relatively low bias. We can think of model 1 as a model of good, efficient representation and complexity as based on the information we have. It has a low bias. We can see the changes and the waving nature of the means across c values for both models and we can see that the variance in Model 2 is higher than the variance in model 1 across c values. It is because for a particular dataset, model 2 learns a set of parameters which could possibly fit the dataset well, but there is an overfitting of the parameters for the data set due to the model complexity and therefore across data sets, the set of parameters learned can be significantly different. For any c value, across T trials, the variance of Model 2 is higher than the variance of model 1. As c increases, we continue from fig 2.2 and extrapolate for larger values of c to see the following trend of an increasing variance,



```
<matplotlib.legend.Legend at 0x1a981903940>
```

Fig 2.4

As we increase the c value, the variance of model 2 increases rapidly in comparison with model 1. If we look at model complexity in terms of the degrees of freedom of the learned model,and think in terms of the function space. We know that x_1 can be correlated with y since x is linearly dependant on x. We know that now x_1 is correlated to a large extent with other x variables. It can be the case that the model can pick up for implicit correlations between other x variables and w. As a rough analogy, for linearly dependent columns of x, there can be multiple w vectors for the same linear combination. In a completely uncorrelated case, say if the x variables are independent, say orthogonal basis vectors in the input space, the model will tend to make w_1 values strongly depend on x_1 and the spread of different x variables across the input space could seldom affect the w_1 value. In the correlated case, the set of possible w_1 values that can be assigned in different regions of the probability space of X is thus a bit larger which could account for a higher variance with increasing c. Our arguments on model complexity, bias and variance hold. we look at the distribution of X and the probability of the data p(X,y),

First consider the graphs with respect to D.

```
<matplotlib.legend.Legend at 0x1a982e1e910>
```

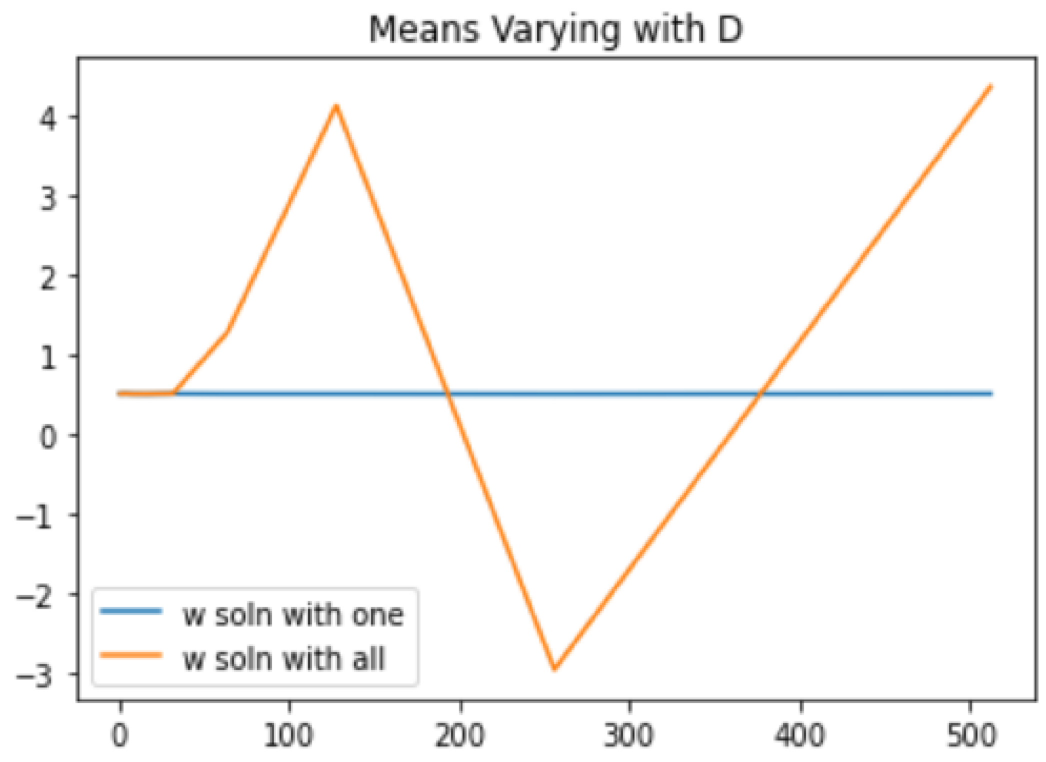


Fig 2.5

```
<matplotlib.legend.Legend at 0x1a982ce6910>
```

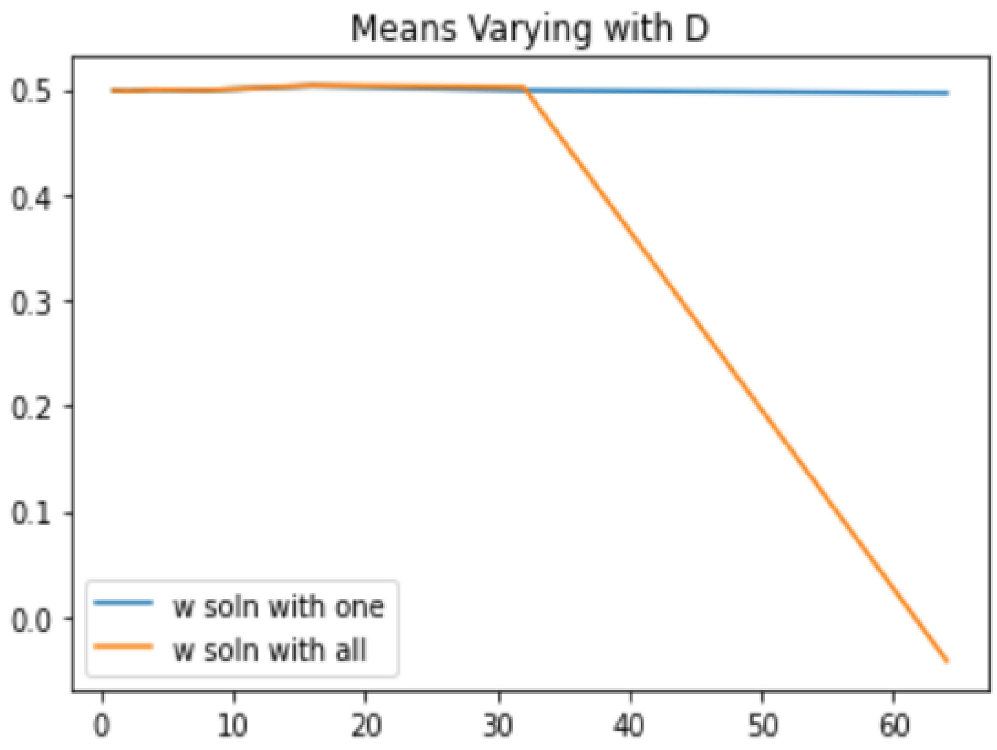

Means Varying with D
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50
60
w soln with one
w soln with all

Fig 2.6

```
<matplotlib.legend.Legend at 0x1a982c7a310>
```

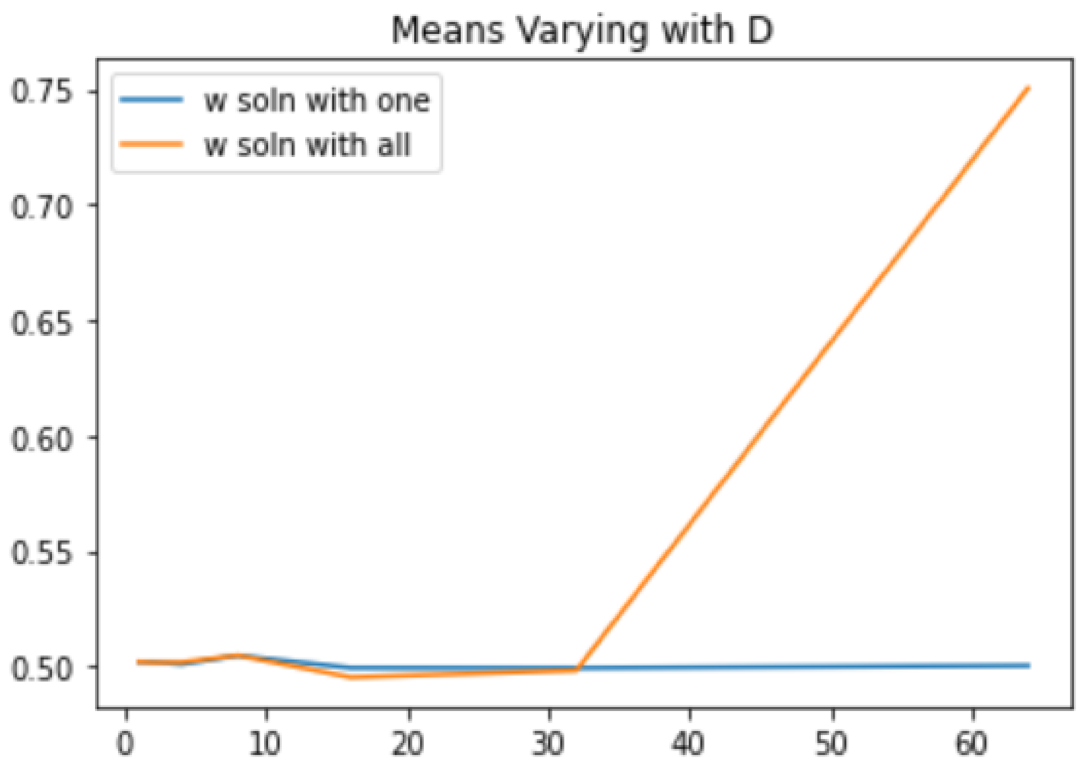


Fig 2.7

There is a clear trend of increase in the variances.

```
<matplotlib.legend.Legend at 0x1a982ea54c0>
```

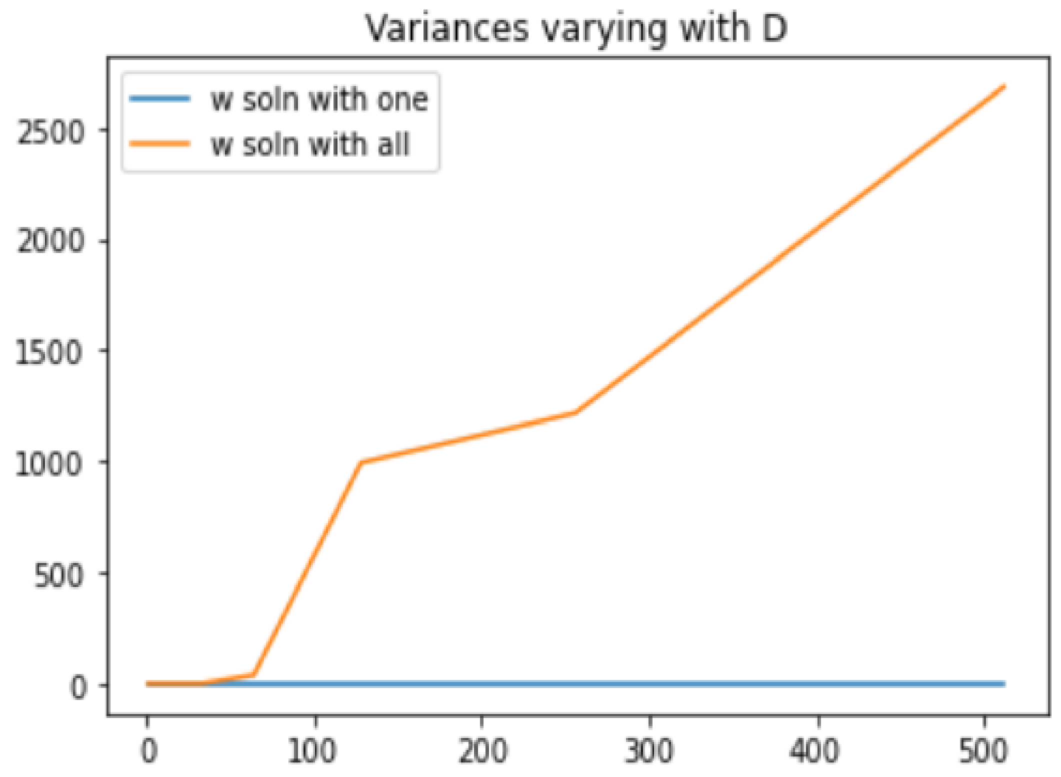

Variances varying with D
w soln with one
w soln with all
2500
2000
1500
1000
500
0
0
100
200
300
400
500

Fig 2.8

```
<matplotlib.legend.Legend at 0x1a982caf8e0>
```

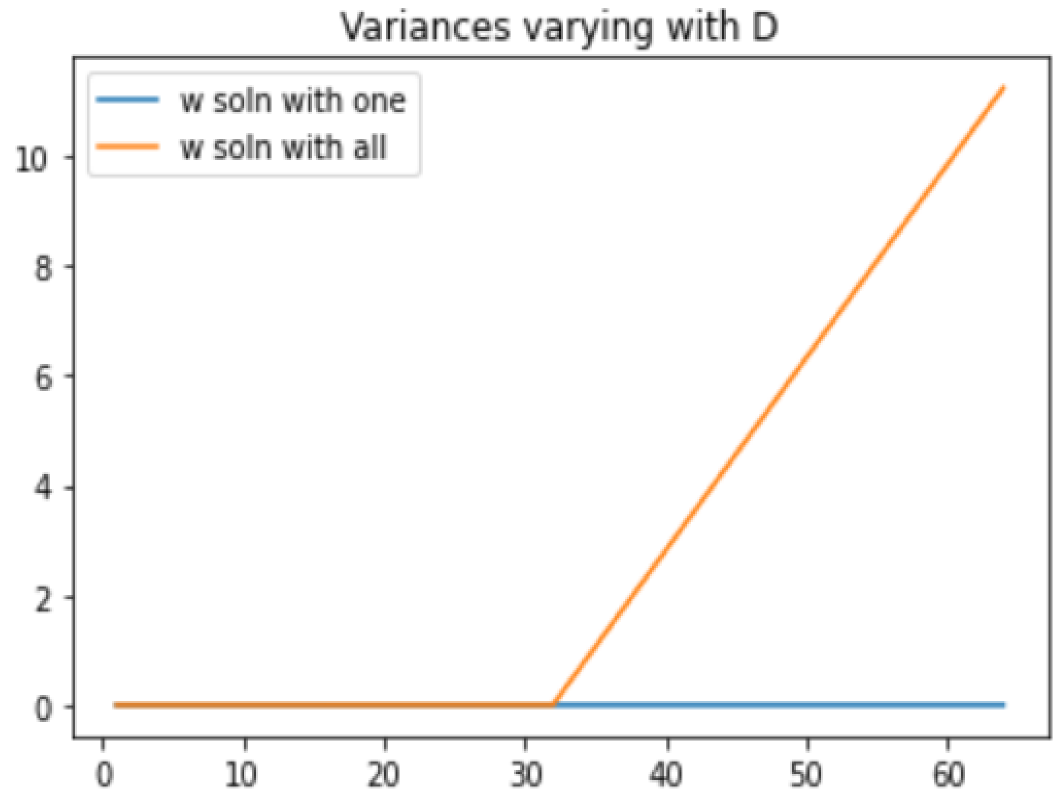

Variances varying with D
w soln with one
w soln with all
10
8
6
4
2
0
0
10
20
30
40
50
60

Fig 2.9

The probability distribution of the data p(X,y) is p(X)p(y/X).Here p(y/X) is given by the linear regression. For p(X), if we have to factor in the multivariate gaussian in our expecatation calculations, as the dimensionality of the distribution increases, so do the number of model parameters and the model complexity. A complex model can have its distribution spread across a larger region and thus affect computations. Again, the variance will be higher as the degrees of freedom for the learned model are more. Here, given our model complexity, if we increase the size of the data set, we can reduce our bias.

$\square$

2 Class-Conditional Gaussian Generative Model

Suppose we have input-output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, K\}$. Let us model this data by modeling the joint distribution using

$$p_\theta(\mathbf{x}, y) = p_\theta(\mathbf{x}|y)p_\theta(y)$$

where θ includes any parameters of the model.

(A) For a given value of θ, how would you predict the label for a given point $\mathbf{x}^\star$ using this joint model $p_\theta(\mathbf{x}, y)$?
We have been given,

$$p_\theta(x, y) = p_\theta(x/y)p_\theta(y)$$

$$p_\theta(x) = \int_y p_\theta(x, y)dy = \int_y p_\theta(x/y)p_\theta(y)dy$$

$$p_\theta p(y/x) = \frac{p_\theta(x/y)p_\theta(y)}{p_\theta(x)}$$

$$p_\theta(y/x) = \frac{p_\theta(x/y)p_\theta(y)}{\int_y p_\theta(x/y)p_\theta(y)dy}$$

we thus can have the conditional distribution of y given x. For some given x as x*, we can make a good prediction for y is we choose y* such that the probability that y* is the class given that x* is the data is maximized.

$$y* = \operatorname*{argmax}_{y=1,..,K} p_\theta(y/x*)$$

(B) Let us model y as a categorical variable where $p_\theta(y = k) \triangleq \text{Cat}(k; \pi) = \pi_k$ where $\pi = [\pi_1, \ldots, \pi_K]$ and $\forall k, \pi_k \geq 0$ and $\sum_k \pi_k = 1$. Write down the log-likelihood of the complete data in terms of $p_\theta(\mathbf{x}|y)$ and $p_\theta(y)$.
Let l be the likelihood. Let there be N data points. We have K classes.

$$l = \prod_{n=1}^{N} p_\theta(x, y) = \prod_{n=1}^{N} p_\theta(x/y)p_\theta(y)$$

$$l = \prod_{n=1}^{N}\prod_{k=1}^{K}[p_\theta(x/y = k)\pi_k]^{(y_n=k)}$$

$$\log l = \sum_{n=1}^{N}[\sum_{k=1}^{K}[I(y_n = k)\log p_\theta(x/y_n = k) + I(y_n = k)\log \pi_k]]$$

This is the log likelihood. We now take into consideration the constriants on π.

(C) Derive an expression for the maximum likelihood estimator (MLE) for π. Make sure to account for the constraints on π using Lagrange multipliers. For now we leave $p_\theta(\mathbf{x}|y)$ unspecified.
Using Lagrange multipliers, as π values denote a distribution, we get for new cost function L,

$$L(l; \lambda) = \sum_{n=1}^{N}[\sum_{k=1}^{K}[I(y_n = k)\log p_\theta(x/y_n = k) + I(y_n = k)\log \pi_k]] + \lambda(1 - \sum_{i=1}^{K}\pi_k)$$

$$\frac{\partial L}{\partial \pi_k} = \frac{1}{\pi_k}\sum_{n=1}^{N} I(y_n = k) - \lambda$$

Now setting,

$$\frac{\partial L}{\partial \pi_k} = 0$$

We get,

$$\pi_k = \frac{\sum_{n=1}^{N} I(y_n = k)}{\lambda}$$

Let N_k be the number of data points of class k,

$$\pi_k = \frac{N_k}{\lambda}$$

Now,

$$\frac{\partial L}{\partial \lambda} = (1 - \sum_{i=1}^{K} \pi_k) = 0$$

$$\sum_{i=1}^{K} \pi_k = 1$$

Therefore we get,

$$\sum_{i=1}^{K} \frac{N_k}{\lambda} = 1$$

$$\lambda = \sum_{k=1}^{K} N_k = N$$

Therefore,

$$\pi_k = \frac{N_k}{N}$$

This is our desired maximum likelihood estimate.

Let us model $\mathbf{x}|y$ with Gaussians $p_\theta(\mathbf{x}|y = k) = \mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)$ where for each k, $\mu_k \in \mathbb{R}^d$ and Σ_k is a $d \times d$ positive semi-definite covariance matrix. Let there only be $K = 2$ classes. This means that the total set of parameters are $\theta = \{\pi, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$.

(D) We now consider a case where the true data comes from this model. That is $y_i \sim \text{Cat}(\pi^{true})$ and $x_i|y_i = k \sim \mathcal{N}(\mu_k^{true}, \Sigma_k^{true})$. Let us try to predict y by directly modeling $p(y = k|\mathbf{x})$ with logistic regression. Recall that logistic regression models y given $\mathbf{x}$ as:

$$y|x \sim \text{Bernoulli}(\sigma(\mathbf{w}^\top \mathbf{x}))$$

where $\sigma(z)$ is the logistic sigmoid:

$$\sigma(z) = \frac{1}{1 + \exp[-z]}$$

Will logistic regression always be able to model the true data conditional $p(y = k|\mathbf{x})$? If so, why? If sometimes, when? And if there are any cases where logistic regression will not be able to model $p(y = k|\mathbf{x})$, are there any ways to fix it?

For classes K_1, K_2

$$p(K_1/x) = \frac{p(x/K_1)p(K_1)}{p(x/K_1)p(K_1) + p(X/K_2)p(K_2)}$$

We know,

$$p(K_1/x) = \sigma(w^T x)$$

$$\sigma(w^T x) = \frac{p(x/K_1)p(K_1)}{p(x/K_1)p(K_1) + p(X/K_2)p(K_2)}$$

-equation a.

$$p(x/K_1) \sim \mathcal{N}(\mu_1, \Sigma_1)$$
$$p(x/K_2) \sim \mathcal{N}(\mu_2, \Sigma_2)$$
$$p(K_2) = \pi_2$$
$$p(K_1) = \pi_1$$

If we know that $\sigma(z)$ is some expression, we can say something about z using properties of the inverse of the sigmoid (logit) function,

$$z = \log \frac{\sigma}{1 - \sigma}$$

Therefore,

$$w^T x = \log \frac{\sigma}{1 - \sigma}$$

Where $\sigma(w^T x)$ comes from equation a.

$1 - \sigma$ is essentially the probability of class K_2. When we take the subtraction of the two log factors, the denominator will cancel out. For each log factor, the log of the exponential will give us a quadratic form in x. The linear term(linear in x) in each quadratic form will have the inverse of the corresponding covariance matrix and the corresponding mean. For the quadratic term in x which will consist of the convariance matrix, if it has to vanish as we require in $w^T x$, we would want the covariance matrices to be the same as only that will ensure the subtraction to zero. If the covariance matrix is the same, for the linear term in w, we can compare forms and get w in terms of the inverse covariance matrix and the subtraction of the means. If we relax the assumption that the covariance matrices are not the same, there will be a quadratic discriminant, we will get quadratic decision boundaries and we know logistic regression can only learn linear decision boundaries. Since for a given w and x, the logistic function is one to one mapping from $w^T x$. The learned w can determine a linear boundary in the input space based on some threshold.

Thus, the datasets coming from our distribution of x given the class must be conducive to linear decision boundaries for logistic regression to work. This will require that the covariance matrix for the two classes be the same. Informally, if two classes have means close to each other and high variance and covariance magnitudes, it can be the case that our sampled data set looks like it doesn't have a linear separation boundary. If we want to learn non linear decision boundaries, we can use models which transform the input space and cast the data into a feature space, such that, in the feature space, the classification problem is linear. The aggregate non linear transformation can be learned by gradient methods using successive parameterized non linear transformations which can enable efficient gradient computation. We can use models like deep neural networks for classification.

3 Poisson Generalized Linear Models (GLM)

Suppose we have input-output pairs $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{N} = \{0, 1, 2, 3, \ldots\}$ for $i = 1, .., N$. Here, the number of features p is greater than the number of data pairs N. Our task is to train a Poisson GLM to model the data. Let θ denote the linear coefficients in the model.

(A) What is the log-likelihood function for Poisson GLM?
For the Possion GLM,

$$p(y/x;\theta) = \frac{e^{y\theta^T x} e^{-e^{\theta^T x}}}{y!}$$

$$p(y_1, .., y_N/x_1, .., x_N, \theta) = \prod_{i=1}^{N} \frac{e^{y_i\theta^T x - e^{-e^{\theta^T x}}}}{y_i!}$$

Therefore the log likelihood is given as,

$$\log(y/x;\theta) = \sum_{i=1}^{N} [y_i\theta^T x - e^{\theta^T x_i} - \log(y_i!)]$$

(B) Given a test point $\mathbf{x}^*$, and some estimate of the parameter $\hat{\boldsymbol{\theta}}$, how do you predict with Poisson GLM?
For a Poisson GLM, we know that, following from the log link function.

$$\log(E(Y/X)) = \theta^T x$$
$$E(Y/X)) = e^{\theta^T x}$$

The predicted value for x* will be $e^{\theta^T x*}$.

(C) Suppose the test point $\mathbf{x}^\star$ is orthogonal to the space generated by the training data. What is the distribution $y|\mathbf{x}^\star$ predicted by a Poisson GLM where the parameter is estimated through ℓ_2-regularized maximum likelihood estimation. Prove your answer.
Consider the log likelihood,

$$\log p(Y/X, w) = \frac{1}{N}\sum_{i=1}^{N} e^{-\theta^T x_i} + y_i\theta^T x_i - \log(y_i!) + \lambda||\theta||^2$$

$$\frac{\partial L}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N} y_i x_i - e^{\theta^T x_i} x_i + 2\lambda\theta = 0$$

$$\theta = \frac{\sum_{i=1}^{N} x_i[e^{\theta^T x_i} - y_i]}{2\lambda N}$$

Here we see the expression for θ through the maximum likelihood solution. Here we see that this θ obtained is a linear combination of the x vectors. It lies in the subspace spanned by x. For any x* orthogonal to the space spanned by x, we get that the dot product $\theta^T x^*$ will be zero. The predicted value of y will be $\theta^T x^*$ which will be one. Substituting 0 in the dot product,

$$p(y/x;\theta) = \frac{e^{y\theta^T x} e^{-e^{\theta^T x}}}{y!}$$

$$p(y_1, .., y_N/x_1, .., x_N, \theta) = \prod_{i=1}^{N} \frac{e^{y_i\theta^T x - e^{-e^{\theta^T x}}}}{y_i!}$$

$$p(y/x;\theta) = \frac{1}{ey!}$$

Since the sum of the reciprocals of factorials converges to e, this is a valid distribution.

(D) Use part (C) of this question to motivate ℓ_1-regularization when the number of training points is less than the features.
When the number of training points is less than the number of features, noting that the dimensionality of theta and x is the same, the data set of input vectors will not be able to span the input space. Here, the theta that is learned lies in the input space. There will be vectors in the inputs in the input space which will be orthogonal to the space spanned by the data vectors. For these vectors, the expected value of y given x is always predicted as 1. We need a better modelling choice and we can use L1 regularization instead of L2 regularization. The theta learned with L1 regularization will not be linear in x. Therefore, the corresponding dot product from the orthogonal subspace need not always to zero. This is a better modelling choice.

4 Alternative Losses

Suppose we are doing binary classification on dataset

$$\{(x_1, y_1), ..., (x_N, y_N)\}, \quad (x_i, y_i) \sim P.$$

where $x_i \in \mathbb{R}$ and $y_i \in \{0, 1\}$ for $i = 1, .., N$.

(A) Which function of x minimizes the expected squared loss on a new test sample $(x^*, y^*) \sim P$? How do you classify with this function? Is this a good solution to the classification problem? Why or why not?

Our expected squared error loss is given as,

$$E[L] = \sum_y \int_x p(f(x) - y)^2 p(x, y) dx$$

We can see from the calculus of variations that the conditional expectation of y given x will be the function f(x) which minimizes the squared error loss.

$$f(x) = \sum_y y.p(y/x)$$

For a new sample x*,

$$f(x*) = E_y[p(y/x*)]$$

We want our prediction to be either 0 or 1. The expected value wil not give us that. The prediction of f(x*) as y by itself is not a good solution to the classification problem.

(B) Now, which distribution maximizes the expected log likelihood? How do you classify with this distribution? Is this a good solution to the classification problem? Why or why not?
We a distribution q such that we want to maximize,

$$E_{(x*,y*)\sim P}[\log(y * /x*)]$$

$$E_{(x*,y*)\sim P}[\log(y * /x*)] = \sum_{y*} p(y * /x*) \log q(y * /x*)$$

Therefore, the problem is equivalent as finding q for minimizing,

$$L = -\sum_{y*} p(y * /x*) \log q(y * /x*)$$

We can write the KL divergence between p and q as,

$$KL(p||q) = -\sum_{y*} p(y * /x*) \log q(y * /x*) + \sum_{y*} p(y * /x*) \log p(y * /x*)$$

Comparing L wit $KL(p||q)$, We can see that the second term in $KL(p||q)$ is independent of q and the first term in $KL(p||q)$ is L. Therefore, minimizing the KL divergence wrt to q will be equivalent to minimizing L wrt q. We know that the KL divergence will be minimized when q is equal to p. Therefore our expected log likelihood will be maximized when q(y/x) is equal to p(y/x). Therefore, q(y*/x*) equal to p(y*/x*) is our desired distribution.

For the next sub-problems, consider the following model. Recall that the sigmoid maps real values to $(0, 1)$ with:

$$\sigma(z) = \frac{1}{(1 + \exp(-z))}$$

Then the model is:

$$P(y_i = 1|x_i) = \sigma(\beta x_i)$$
$$P(y_i = 0|x_i) = 1 - P(y_i = 1|x_i)$$

Now we investigate how an extreme point would affect different estimators. Suppose a new point ($x_{N+1} = 100, y_{N+1} = 0$) appears and we want to see how this point alters estimates by looking at derivatives of two different loss functions.

(C) Compute the derivative of the square loss for the new data point $(y_{N+1} - \sigma(\beta x_{N+1}))^2$ w.r.t β at $\beta = 3$.

$$Loss = L = (y_{N+1} - \sigma(\beta x_{N+1}))^2$$
$$\frac{\partial L}{\partial \beta} = 2 * (y_{N+1} - \sigma(\beta x_{N+1})) * \sigma(\beta x_{N+1}) * (1 - \sigma(\beta x_{N+1})) * x_{N+1}$$

Here, this is essentially 0, since the sigmoid function tends to be very close to 1, for numerical purposes, this is 0.

$$\frac{\partial L}{\partial \beta} = 0.0$$

(D) Compute the derivative of the negative log-likelihood of logistic regression for the new data point w.r.t β at $\beta = 3$.

$$Loss = L = -(1 - y_{N+1}) \log(1 - sigma(\beta x_{N+1})) - y_{N+1} \log(sigma(\beta x_{N+1}))$$

as $y_{N+1} = 0$,

$$L = -\log(1 - sigma(\beta x_{N+1}))$$
$$\frac{\partial L}{\partial \beta} = -\frac{1}{1 - sigma(\beta x_{N+1})} sigma(\beta x_{N+1})(1 - sigma(\beta x_{N+1})) * x_{N+1}$$

This comes out to be -100.0

(E) Compare the results of parts (3) and (4). What does it imply? Explain why. What does it tell you about how to choose losses in general?
The y value is 0 and the predicted value is close to 1, so we know that our loss is high. The squared error loss is 1 while the logistic regression loss tends to infinity. Here, the derivative of the sigmoid function for higher values of x tends to zero. Hence the aggregative derivative of the squared error loss function is also zero. If we are using gradient based methods for estimating beta based on the square error loss function, the training may not be very effective for very large and very small values of x.

In general, we would want losses which have a good probabilistic basis in a model set up suitable for the task. We can model a bernoulli distribution for y given x in binary classification whose parameter we predict using our generalized linear model of logistic regression. A gaussian distribution for y given x in a binary classification task does not make much modelling sense, our y is discrete binary. The negative log likelihood is the correct loss for the maximum likelihood estimate based on our logistic function use for prediction of the conditionals. A maximum likelihood estimate has desirable properties. When doing binary classification with logistic regression, the squared error loss does not follow from the maximum likelihood formulation. In comparison, the cross entropy loss follows from the minimization of KL divergence the true distribution of the classes given our x and our predicted distribution of the classes given x, which is also the minimization of the negative log likelihood. Also, the squared error loss for binary classification will lead to a non convex optimization problem. For classification, we should go with the cross entropy loss.