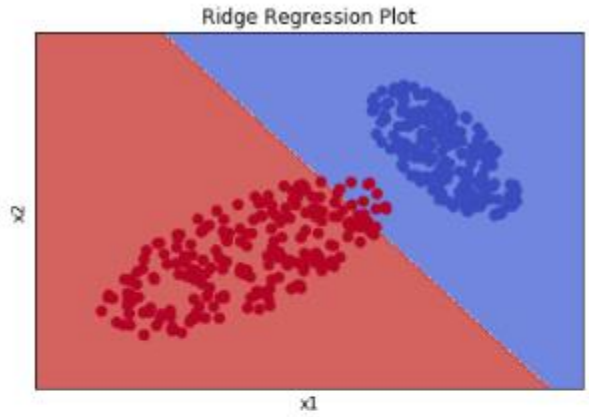
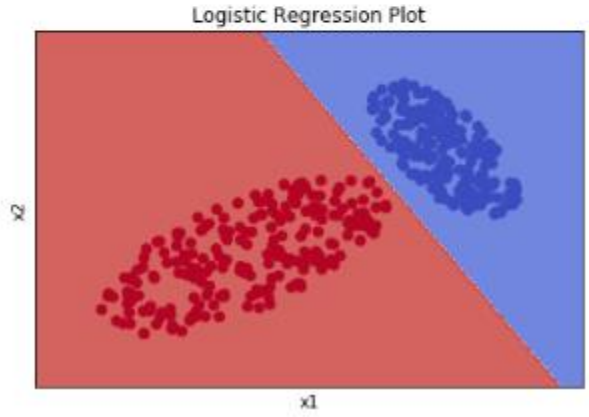
1a. Squared loss is a terrible choice because for classification since it heavily punishes outliers and points that yield a large w^Tx value, even if they are classified correctly. The error value is squared, so an outlier would create an enormous error relative to other values. A large w^Tx value, even classified correctly (have the same sign as the corresponding y value), will still become squared into a large error value, and a large error will shift the decision boundary greatly to compensate for the huge difference. This process makes squared loss very inefficient for classification.

1b. The difference in the graphs is the position of the decision boundary between the two chunks of data points (there is a slight difference in angle but not significant in the terms of the question). The ridge regression, using the squared loss, is shifted to the left, which shows how the decision boundary is affected as described in part 1a. Since the red, left data is not clumped, the squared loss moves to the left to compensate for the data far on the left. The logistic regression does not do this, which shows why squared loss should not be used for classification.





1c. Taking the derivatives of the loss with respect to w:

$$\nabla_w L_{hinge} = \begin{cases} 0, & yw^Tx \geq 1 \\ -yx, & yw^Tx < 1 \end{cases}$$

$$\nabla_w L_{log} = \frac{-xy}{e^{yw^T x} + 1}$$

x_0	x_1	x_2	y	hinge	log
1	½	3	1	(-1, -½ , -3)	(-0.38, -0.19, -1.1)
1	2	-2	1	(0, 0, 0)	(-0.12, -0.24, 0.24)
1	-3	1	-1	(0, 0, 0)	(0.047, -0.14, 0.047)

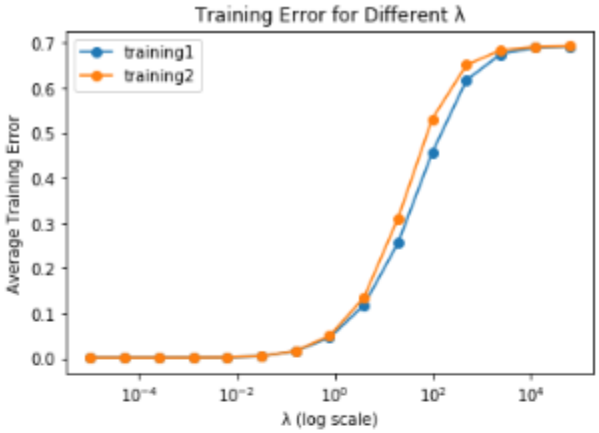
1d. Whenever $yw^Tx < 1$, the hinge loss returns a gradient much larger than the log loss gradient; however, the hinge loss did not update for correctly classified points when the log loss does. The log loss always has a gradient, even if it is a very small decimal, while the hinge loss makes larger updates when the points are misclassified. The hinge gradient will converge to zero when all the points are correctly classified, since the hinge gradient is only nonzero when $yw^Tx < 1$. The log gradient will converge to zero when yw^Tx gets infinitely large, either by having a large, positive value with y = 1 or a large, negative value with y = -1. We can reduce training error without changing the decision boundary by maximizing the margin.

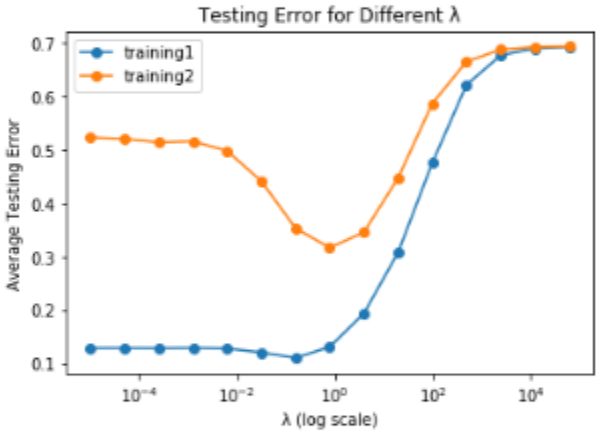
1e. If we just try to minimize L_{hinge}, we will just create a decision boundary that correctly classifies all the points, but it does not address the issue of maximizing the margin. In order to maximize the margin, we must add the $\lambda\|w\|^2$ $(\lambda > 0)$ to the loss and minimize it altogether. This will then create a decision boundary with correct classification of points and a maximum margin between nearest points. The addition of the new term forces the decision boundary to fit to the maximum margin position.

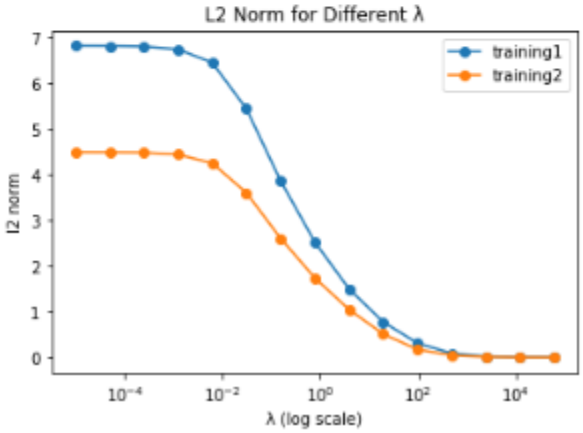
2a. Adding the penalty term cannot decrease the training error any more than the unregularized version. Without the penalty term, the unregularized version fits to the model that minimizes the training error, so the regularized version cannot decrease the in-sample error anymore. The penalty term is meant to deal with out of sample error. Adding the penalty term will not always decrease the out-of-sample error either. When there is overfitting, regularization can do a good job of reducing variance, but it may affect the model too much and increase the bias, causing underfitting.

2b. The l_0 regularization is rarely used because of the properties of the model. The model is discontinuous and undifferentiable which are needed for minimization in regularization. Also, this model heavily penalizes values, even if they are very close to zero. The l_0 norm adds one to the sum if the value is not zero, so even small values are penalized heavily. This makes this method less effective.

2c. The plots given from my code:







2d. The training error is very similar for both training sets relative to the changing λ. This makes sense because a stronger regularizer makes it more difficult for the model to perfectly fit any data set. Therefore, the training error increases in the same quantity for data sets of different sizes.

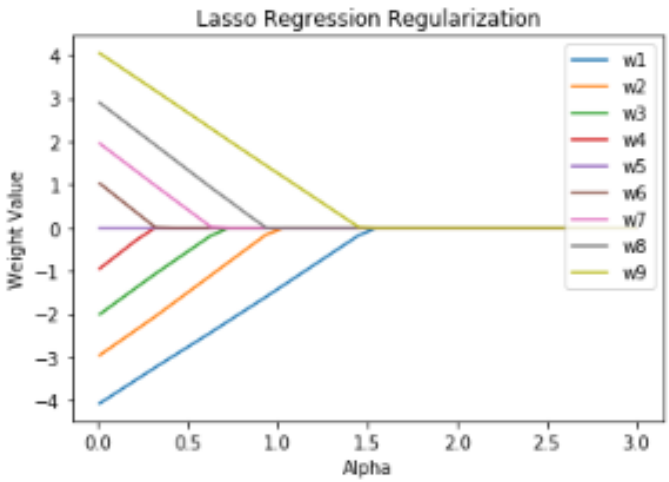
The testing error differentiates greatly, and this can be accounted for by the overfitting of the smaller data set. Training set 2 overcompensates on the small data set and causes overfitting, which results in a higher testing error than training set 1. However, as regularization increases, the testing error becomes similar between the two sets for similar reasons stated for the training error.

2e. For low values of λ, we can see slight overfitting. The training error is incredibly low, while we have some testing error. This is a clear indication of overfitting. This makes sense because the regularization term does practically nothing at these values. As λ increases, we approach a situation where the testing error decreases while the training error is also low. This is an idea value for the first data set to be trained at. After this point, the training error and testing error both increase drastically which is due to underfitting. Heavy regularization causes the model to fit the data too much which increases both errors, hence underfitting.

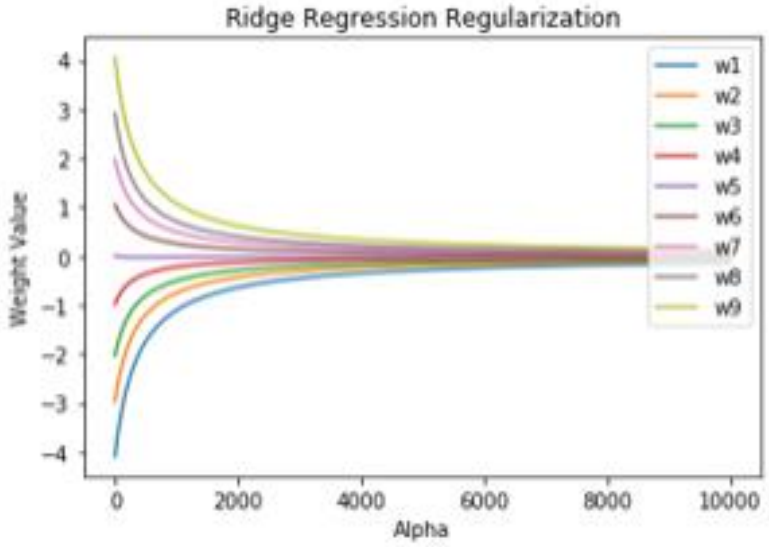
2f. The l_2 norm is strictly decreasing with increasing λ. This is because a higher λ imposes a higher penalty for large l_2 norm, so there is a greater emphasis on the regularization term. For large λ values, the minimization problem becomes minimizing the regularization term.

2g. We would use the λ that would give the lowest testing error. This comes at $\lambda_7 = 0.78125$.

3ai. Code is in the Jupyter notebook:



3aii. Code is in the Jupyter notebook:



3aiii. As the regularization parameter increases, the number of model weights that are exactly zero increases with Lasso regression until all the weights are zero. With Ridge regression, the number of model weights that are exactly zero is zero. The model weights get close to zero, but none of them become zero.

3bi. We need to differentiate and set equal to zero to find w:

$$\nabla_w \left|\left|y - X^T w\right|\right|^2 + \lambda \left|\left|w\right|\right|_1$$

$$= -2X^T(y - X^T w) + \lambda = 0$$

$$w = \frac{2X^T y - \lambda}{2X^T X}$$

However, we know that $\nabla_w \lambda \left|\left|w\right|\right|$ has three cases depending on w; therefore, this means $\nabla_w \lambda \left|\left|w\right|\right|$ can range from $[-\lambda, \lambda]$. Therefore, we can write the solution as:

$$w = \frac{2X^T y + n\lambda}{2X^T X}$$

where n is a value on the interval $[-1, 1]$ depending on w.

3bii. Yes, we can set the numerator equal to zero to get

$$\lambda = \frac{2X^T y}{n}$$

and our smallest value for λ that gives zero is when n = 1 or n = -1:

$$\lambda = |2X^T y|$$

3biii. We take the derivate again:

$$\nabla_w \left||y - X^T w|\right|^2 + \lambda \left||w|\right|^2_2$$

$$= -2X^T(y - X^T w) + 2\lambda w = 0$$

$$w = \frac{X^T y}{\lambda I + X^T X}$$

where I is the appropriate identity matrix

3biv. First, we must consider the $\lambda = 0, w \neq 0$ case:

$$0 \neq \frac{X^T y}{X^T X}$$

We know that X^TX is positive by nature, so

$$0 \neq X^T y$$

So, to have

$$0 = \frac{X^T y}{\lambda I + X^T X}$$

we must have

$$0 = \frac{1}{\lambda I + X^T X}$$

However, we know that X^TX is positive, and we have restricted $\lambda > 0$. This is not possible algebraically. There is no value for positive λ where w = 0.