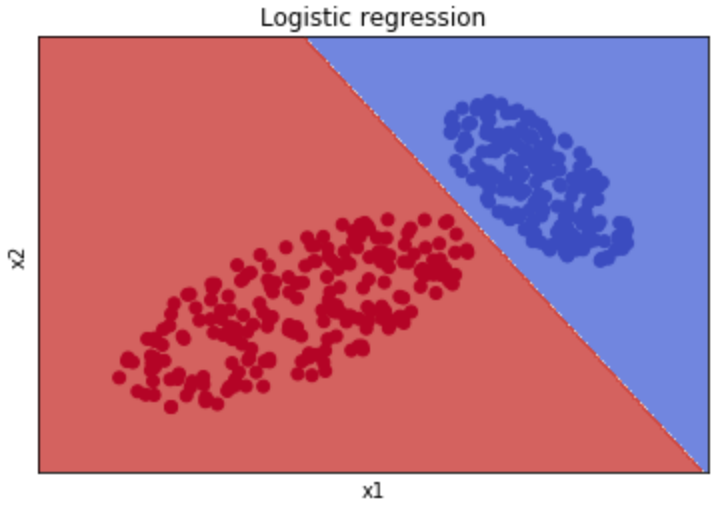
CS 155 PS 2

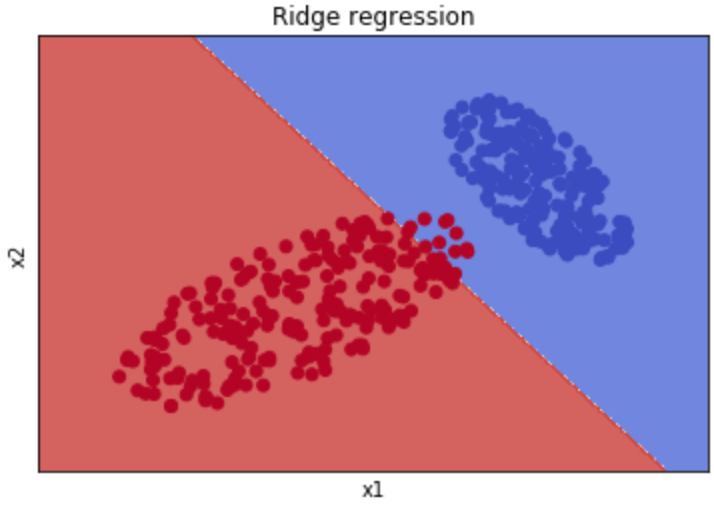
1 Comparing Different Loss Functions

Problem A

Squared loss is a terrible choice of loss function to train on for classification problems because it penalizes large values even if classified correctly. Since it only cares about how far the value of $\mathbf{w}^T\mathbf{x}$ is from the target (-1 or +1), points that are well classified but far from the target (i.e. y = +1, $\mathbf{w}^T\mathbf{x}$ = +5) will have high loss simply because of the distance.

Problem B

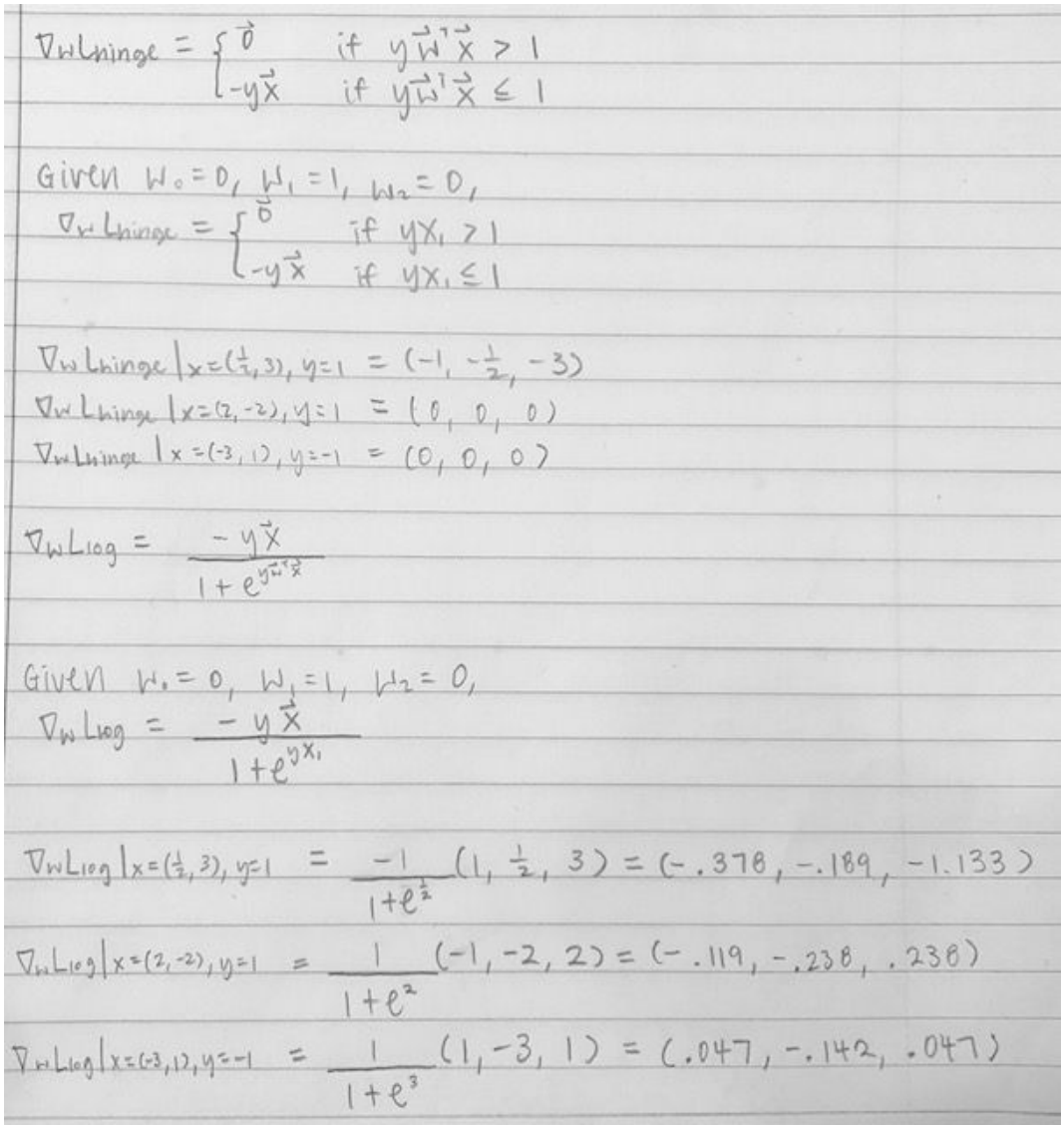




The decision boundary for logistic regression perfectly classifies the points, but the decision boundary for ridge regression is shifted to the bottom left, misclassifying some of the red points. This is because the squared loss function penalizes correctly classified points that are far from the boundary, causing the boundary to shift towards such points to decrease the loss values for them. On the other hand, log loss

function only penalizes misclassified values, according to their magnitude, and furthermore produces lower loss for correctly classified points farther from the boundary.

Problem C



Problem D

For the points with absolute value of x_1 greater than or equal to 1, the gradients for the hinge loss are zero (since they're all correctly classified). For the same points, the gradients for the log loss are small but not zero. The gradients for the log loss decrease as they get farther from the decision boundary (as long as classified correctly). The gradients for the hinge loss become zero as soon as y * $\mathbf{w}^T\mathbf{x}$ becomes greater than or equal to 1 (outside of margin on the correct side), while the gradients for the log loss converge to zero as y * $\mathbf{w}^T\mathbf{x}$ goes to infinity (as far from the margin as possible on the correct side). Scaling up the weight vector (including the bias term) will reduce the training error if the points are correctly classified,

while preserving the decision boundary, since it will increase the value of y * $\mathbf{w}^T\mathbf{x}$. The hinge loss will be completely eliminated if y * $\mathbf{w}^T\mathbf{x}$ is greater than or equal to 1, and the log loss will be reduced as y * $\mathbf{w}^T\mathbf{x}$ gets scaled up higher.

Problem E

Minimizing just L_{hinge} could be done by scaling up $\mathbf{w}$ without changing the decision boundary, which is what we want to control in order to classify the points correctly and maximize the margin. The penalty term addresses this issue by constraining the magnitude of $\mathbf{w}$.

2 Effects of Regularization

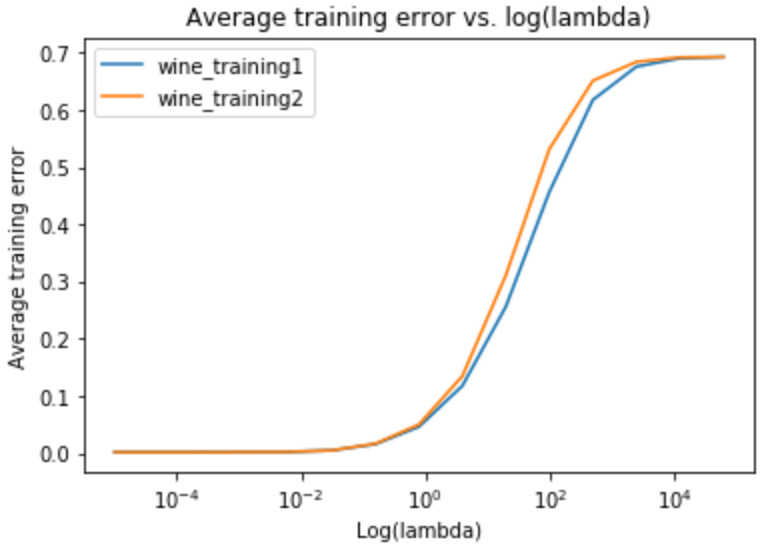
Problem A

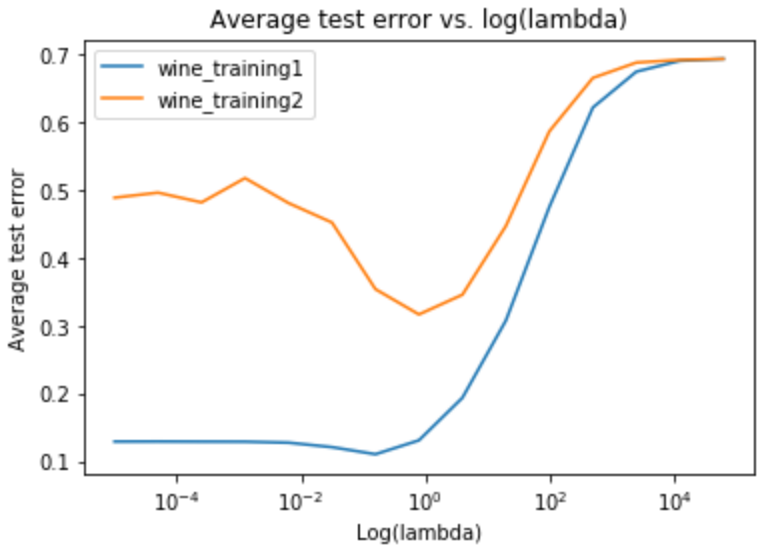
Adding the penalty term can't decrease the training error, since the term generally prevents the model from fitting perfectly to the training sample. The training error can only stay the same or increase due to a smaller hypothesis set. Adding a penalty term will decrease the out-of-sample errors only when there is overfitting (such as when the training set is small and model complexity is high). If the regularization parameter is too large, the out-of-sample errors can increase due to underfitting.

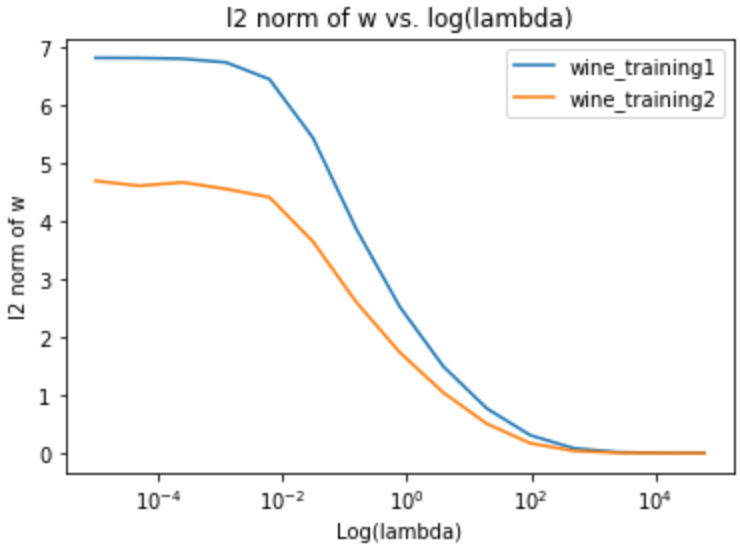
Problem B

The regularization term of the objective function for l_0 regularization is discontinuous and non-convex, making the optimization very difficult.

Problem C







Problem D

The difference between the training errors for the two training sets is minimal. As lambda increases, the training error for wine_training2 increases at a slightly faster rate than wine_training1. Since wine_training1 has fewer data points, its training error is more susceptible to the change (the limitation of the model) as the model gets more regularized. The test error for wine_training2 is much higher with low lambda, because its small sample size causes it to overfit. With the optimal amount of regularization, the test error for wine_training2 goes down, yet it is always higher (until the errors converge) than the test error for wine_training1 which is a better approximation of the true distribution.

Problem E

The training error for wine_training1 only increases with increasing lambda, showing the underfitting behavior (the model is limited in fitting the training data perfectly) as lambda increases. The test error for wine_training1 is quite low even with no regularization, and reaches its minimum at a lambda value

around 0.1. This means that there is not a lot of overfitting with this training data, and the model generalizes the best at $\lambda \cong 0.1$. The model underfits at higher lambda values, as the increasing test error shows.

Problem F

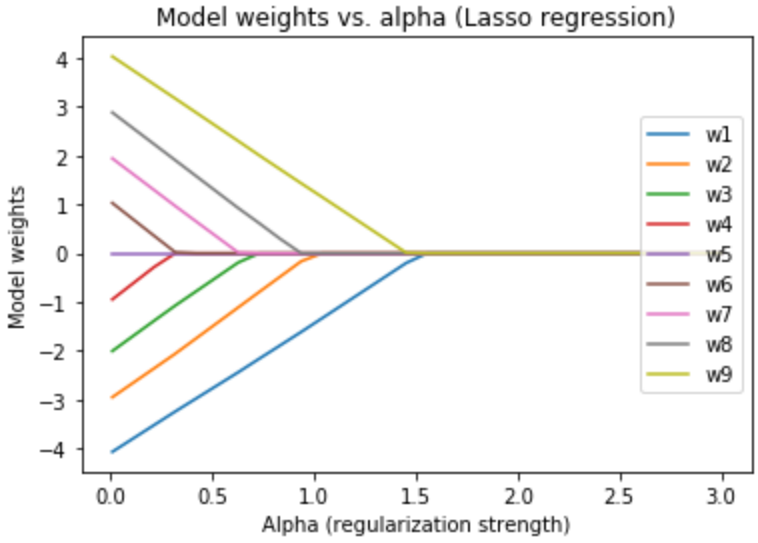
The l_2 norm of **w** decreases as lambda increases, because l_2 regularized training is equivalent to norm-constrained training using Lagrangian. As the regularization strength increases, the norm is constrained to smaller values.

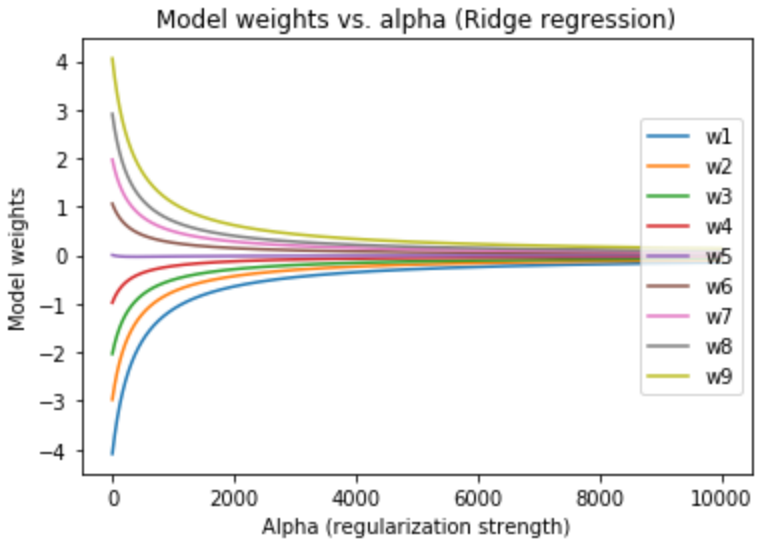
Problem G

As the test error plot shows, $\lambda \cong 1$ would be the optimal lambda value to prevent the overfitting behavior of wine_training2. Regularizing with $\lambda \cong 1$ would allow the model to generalize the best with the true distribution.

3 Lasso (l_1) vs. Ridge (l_2) Regularization

Problem A





As the regularization parameter increases, the number of model weights that are exactly zero increases with Lasso regression. The model weights for Ridge regression asymptotically reach zero as the regularization parameter increases, but none of the weights become exactly zero.

Problem B

i. To minimize $\|\vec{y} - \vec{x}w\|^2 + \lambda\|w\|_1$ with respect to w, we want to differentiate this function wrt. w and set it equal to 0. Since $\lambda\|w\|_1$ isn't differentiable, we will use its subgradient definition:

$$\nabla_w \lambda\|w\| = \begin{cases} -\lambda & \text{if } w < 0 \\ +\lambda & \text{if } w > 0 \\ [-\lambda, +\lambda] & \text{if } w = 0 \end{cases}$$

Thus $\frac{d}{dw}(\|\vec{y} - \vec{x}w\|^2 + \lambda\|w\|_1)$

$= -2\vec{x}^T(\vec{y} - \vec{x}w) + \lambda$ if $w > 0$, $\quad -2\vec{x}^T(\vec{y} - \vec{x}w) - \lambda$ if $w < 0$

Solving for w using $\frac{d}{dw}(\|\vec{y} - \vec{x}w\|^2 + \lambda\|w\|_1) = 0$ gives

$$w = (2\vec{x}^T\vec{y} - \lambda)(2\vec{x}^T\vec{x})^{-1} \quad \text{if } w > 0$$
$$w = (2\vec{x}^T\vec{y} + \lambda)(2\vec{x}^T\vec{x})^{-1} \quad \text{if } w < 0$$

$w > 0$ only if $2\vec{x}^T\vec{y} > \lambda$, given $\lambda \geq 0$
$w < 0$ only if $2\vec{x}^T\vec{y} < -\lambda$, given $\lambda \geq 0$
$w = 0$ only if $-\lambda \leq 2\vec{x}^T\vec{y} \leq \lambda$

Thus

$$w = \begin{cases} (2\vec{x}^T\vec{y} - \lambda)(2\vec{x}^T\vec{x})^{-1} & \text{if } 2\vec{x}^T\vec{y} > \lambda \\ (2\vec{x}^T\vec{y} + \lambda)(2\vec{x}^T\vec{x})^{-1} & \text{if } 2\vec{x}^T\vec{y} < -\lambda \\ 0 & \text{if } -\lambda \leq 2\vec{x}^T\vec{y} \leq \lambda \end{cases}$$

ii. Yes; as shown above, $w = 0$ if $-\lambda \leq 2\vec{x}^T\vec{y} \leq \lambda$
Thus the smallest value for λ s.t. $w = 0$ is $|2\vec{x}^T\vec{y}|$.
If negative λ values are allowed, $-\infty$ would be the smallest such value.

iii. To minimize $\|\vec{y} - \vec{X}\vec{w}\|^2 + \lambda \|\vec{w}\|_2^2$ wrt. $\vec{w}$, we want to take the gradient of this function wrt. $\vec{w}$ and set it equal to zero.

$\nabla_{\vec{w}}(\|\vec{y} - \vec{X}\vec{w}\|^2 + \lambda \|\vec{w}\|_2^2)$

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

solving for $\vec{w}$ gives

$\vec{w} = (\vec{X}^T\vec{X} + \lambda I_D)^{-1}\vec{X}^T y$

iv. Given $w \neq 0$ when $\lambda = 0$,

we know that $\vec{X} \neq 0$ and $\vec{y} \neq 0$, since

$w = \vec{x}^T\vec{y}(\vec{x}^T\vec{x})^{-1}$ when $\lambda = 0$

and either $\vec{x}$ or $\vec{y}$ being 0 causes w to be 0.

Therefore, regardless of λ value, w can't be 0.

Intuitively, λ is in the denominator and can't make the expression 0.

∴ There does not exist a value for $\lambda > 0$ s.t. $w = 0$.