Note: 19 late hours were used for this set.

1 Comparing Different Loss Functions [30 Points]

Relevant materials: lecture 3

We've discussed three loss functions for linear models so far:

- Squared loss: $L_{\text{squared}} = (1 - y\mathbf{w}^T\mathbf{x})^2$
- Hinge loss: $L_{\text{hinge}} = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$
- Log loss: $L_{\text{log}} = \ln(1 + e^{-y\mathbf{w}^T\mathbf{x}})$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of the model parameters, $y \in \{-1, 1\}$ is the class label for datapoint $\mathbf{x} \in \mathbb{R}^n$, and we're including a bias term in $\mathbf{x}$ and $\mathbf{w}$. The model classifies points according to sign$(\mathbf{w}^T\mathbf{x})$.

Performing gradient descent on any of these loss functions will train a model to classify more points correctly, but the choice of loss function has a significant impact on the model that is learned.

Problem A [3 points]: Squared loss is often a terrible choice of loss function to train on for classification problems. Why?

Squared loss is often a terrible choice of loss function to train on for classification problems because the squared loss function penalizes models for having classification values that are much higher than the minimum classification values, resulting in having a loss function that often penalizes for a model having correct classifications but with values that are not close enough to the classification boundary value.

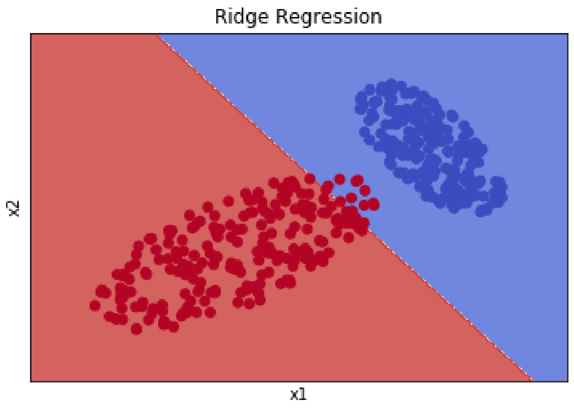
Problem B [9 points]: A dataset is included with your problem set: `problem1data1.txt`. The first two columns represent x_1, x_2, and the last column represents the label, $y \in \{-1, +1\}$.

On this dataset, train both a logistic regression model and a ridge regression model to classify the points. (In other words, on each dataset, train one linear classifier using L_{log} as the loss, and another linear classifier using L_{squared} as the loss.) For this problem, you should use the logistic regression and ridge regression implementations provided within scikit-learn (logistic regression documentation) (Ridge regression documentation) instead of your own implementations. Use the default parameters for these classifiers except for setting the regularization parameters so that very little regularization is applied.

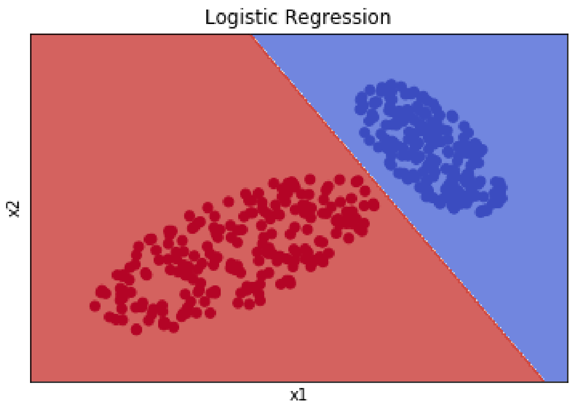
For each loss function/model, plot the data points as a scatter plot and overlay them with the decision boundary defined by the weights of the trained linear classifier. Include both plots in your submission. The template notebook for this problem contains a helper function for producing plots given a trained classifier.

What differences do you see in the decision boundaries learned using the different loss functions? Provide a qualitative explanation for this behavior.

Plot of ridge regression decision boundary:



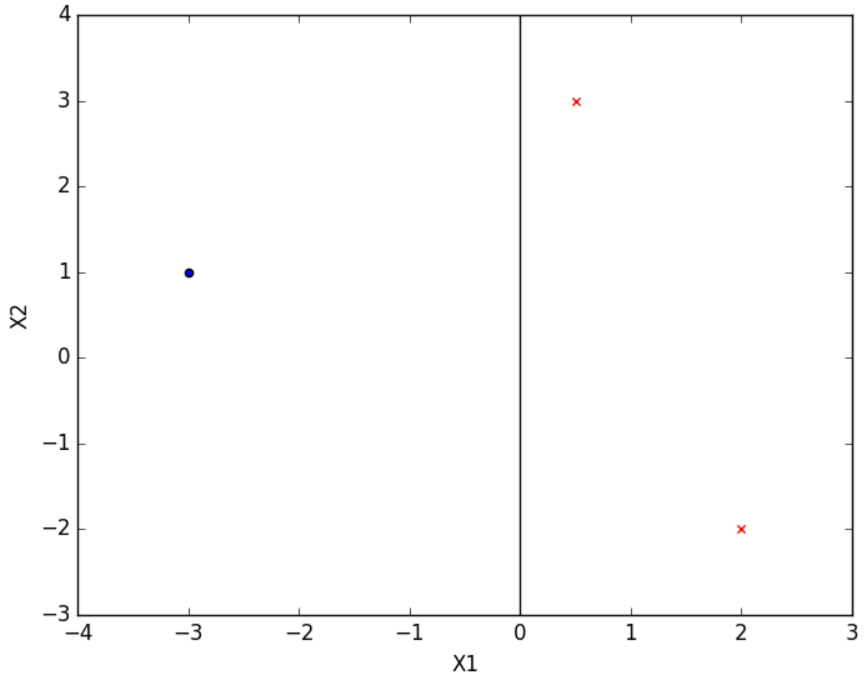
Plot of logistic regression decision boundary:



The decision boundary for the logistic regression completely separates the points according to their classifications, while the decision boundary for the ridge regression does not (it passes within the space of red points (+1 classification) points). The reason why the ridge regression does not separate the points in the dataset is because the hinge loss function only penalizes for incorrect classifications, but does not penalize for weight vectors for classifications that are very close to the classification threshold but are still correct. This results in a decision boundary that classifies data slightly worse than the logistic regression, which penalizes for weights that give correct classifications but are still too close to the classification threshold.

Problem C [9 points]: Leaving squared loss behind, let's focus on log loss and hinge loss. Consider the set of points $S = \{(\frac{1}{2}, 3), (2, -2), (-3, 1)\}$ in 2D space, shown below, with labels $(1, 1, -1)$ respectively.

Given a linear model with weights $w_0 = 0, w_1 = 1, w_2 = 0$ (where w_0 corresponds to the bias term), compute the gradients $\nabla_w L_{\text{hinge}}$ and $\nabla_w L_{\text{log}}$ of the hinge loss and log loss, and calculate their values for each point in S.



The example dataset and decision boundary described above. Positive instances are represented by red x's, while negative instances appear as blue dots.

For a linear model, the hinge loss function is given by $L_{\text{hinge}}(\mathbf{x}, y|\mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$.

Therefore, the gradient of the hinge loss function for the linear model is

$$\nabla_w L_{\text{hinge}}(\mathbf{x}, y|\mathbf{w}) = \nabla_{\mathbf{w}} \max(0, 1 - y\mathbf{w}^T\mathbf{x}) = \begin{cases} \nabla_{\mathbf{w}} 0 & L_{\text{hinge}}(\mathbf{x}, y|\mathbf{w}) \leq 0 \\ \nabla_{\mathbf{w}}(1 - y\mathbf{w}^T\mathbf{x})) & L_{\text{hinge}}(\mathbf{x}, y|\mathbf{w}) > 0 \end{cases}$$

$$= \begin{cases} 0 & L_{\text{hinge}}(\mathbf{x}, y|\mathbf{w}) = 0 \\ -y\mathbf{x} & L_{\text{hinge}}(\mathbf{x}, y|\mathbf{w}) > 0 \end{cases}.$$

For the points in $S = \{(\frac{1}{2}, 3), (2, -2), (-3, 1)\}$ (with $S = \{(1, \frac{1}{2}, 3), (1, 2, -2), (1, -3, 1)\}$ including the x_0 bias terms) with labels $(1, 1, -1)$ respectively, their loss function values are $\frac{1}{2}$, 0, and 0 respectively. Since the point $(\frac{1}{2}, 3)$ has a positive loss function value, the gradient of this point is $-y\mathbf{x} = (-1, -\frac{1}{2}, -3)$. Since the last two points have loss function values of 0, the gradient values for both these points ($(2, -2)$ and $(-3, 1)$)

are 0.

For a linear model, the logistic loss function is given by
$L_{\text{log}}(\mathbf{x}, y|\mathbf{w}) = -\ln\Big(\frac{e^{\frac{1}{2}yf(\mathbf{x}|\mathbf{w})}}{e^{\frac{1}{2}yf(\mathbf{x}|\mathbf{w})} + e^{-\frac{1}{2}yf(\mathbf{x}|\mathbf{w})}}\Big) = -\ln\Big(\frac{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}\Big)$.

Therefore, the gradient of the logistic loss function for the linear model is

$$\nabla_{\mathbf{w}} L_{\text{log}}(\mathbf{x}, y|\mathbf{w}) = \nabla_{\mathbf{w}}\Big(-\ln\Big(\frac{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}\Big)\Big) = -\frac{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}\nabla_{\mathbf{w}}\Big(\frac{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}\Big)$$

$$= -\frac{e^{\frac{1}{2}y\mathbf{w}^T x} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}\frac{(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})\nabla_{\mathbf{w}} e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} - e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}\nabla_{\mathbf{w}}(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})}{(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})^2}$$

$$= -\frac{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}{e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}}\frac{(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})(\frac{1}{2}y\mathbf{x})e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} - e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}(\frac{1}{2}y\mathbf{x})(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} - e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})}{(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})^2}.$$

For the points in $S = \{(\frac{1}{2}, 3), (2, -2), (-3, 1)\}$ (with $S = \{(1, \frac{1}{2}, 3), (1, 2, -2), (1, -3, 1)\}$ including the x_0 bias terms) with labels $(1, 1, -1)$ respectively (and with weights $w_0 = 0, w_1 = 1, w_2 = 0$), their gradient values are $(-0.377541, -0.18877, -1.13262)$, $(-0.119203, -0.238406, 0.238406)$, and $(-0.952574, 2.85772, -0.952574)$ respectively.

Problem D [4 points]: Compare the gradients resulting from log loss to those resulting from hinge loss. When (if ever) will these gradients converge to 0? For a linearly separable dataset, is there any way to reduce or altogether eliminate training error without changing the decision boundary?

While the gradients of log loss above are all nonzero, two of the three hinge loss gradients are zero. Additionally, the nonzero gradient of hinge loss above $(-1, -\frac{1}{2}, -3)$ has a larger magnitude than any of the three logistic loss gradients above.

For the nonzero gradient in hinge loss above $((-1, -\frac{1}{2}, -3))$, that gradient will converge to zero once $\mathbf{y}\mathbf{w}^T\mathbf{x} > 1$, since then the hinge loss will become 0 and the gradient will be 0 as well.

The gradients of logistic loss will converge to zero as the term $y\mathbf{w}^T\mathbf{x}$ increases. This is because the denominator of the gradient of the logistic loss function is $e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}}(e^{\frac{1}{2}y\mathbf{w}^T\mathbf{x}} + e^{-\frac{1}{2}y\mathbf{w}^T\mathbf{x}})^2$, so as this denominator increases, the gradients of logistic loss will converge to zero.

For a linearly separable dataset, it is possible to reduce or altogether eliminate training error without changing the decision boundary, by changing the magnitude of the weight vector used by a machine learning model to linearly separate data based on their classifications.

Problem E [5 points]: Based on your answer to the previous question, explain why for an SVM to be a "maximum margin" classifier, its learning objective must not be to minimize just L_{hinge}, but to minimize $L_{\text{hinge}} + \lambda\|w\|^2$ for some $\lambda > 0$.

(You don't need to prove that minimizing $L_{\text{hinge}} + \lambda\|w\|^2$ results in a maximum margin classifier; just show that the additional penalty term addresses the issues of minimizing just L_{hinge}.)

By minimizing the $\lambda||w||^2$ term in the SVM learning objective, this minimizes the term $||w||^2$, which is equivalent to maximizing $1/||w||^2$. Since $1/||w||^2$ is maximizing the margin with a linear separating model, this explains why for an SVM to be a "maximum margin" classifier, its learning objective must not be to minimize just L_{hinge}, but to minimize $L_{\text{hinge}} + \lambda\|w\|^2$ for some $\lambda > 0$.

2 Effects of Regularization

Relevant materials: Lecture 4

For this problem, you are required to implement everything yourself and submit code.

Problem A [4 points]: In order to prevent over-fitting in the least-squares linear regression problem, we add a regularization penalty term. Can adding the penalty term decrease the training (in-sample) error? Will adding a penalty term always decrease the out-of-sample errors? Please justify your answers. Think about the case when there is over-fitting while training the model.

Adding the penalty term cannot decrease the training (in-sample) error, because the regularization term adds an augmented error to the loss function, making the training error greater than or equal to what the training error would be without regularization.

Adding a penalty term will not always decrease the out-of-sample errors. When over-fitting to the training model without regularization, adding an optimal regularization term penalty will decrease the out-of-sample errors. However, adding an non-optimal regularization term penalty (with $lambda$ too small or too large) will not decrease the out-of-sample errors.

Problem B [4 points]: ℓ_1 regularization is sometimes favored over ℓ_2 regularization due to its ability to generate a sparse w (more zero weights). In fact, ℓ_0 regularization (using ℓ_0 norm instead of ℓ_1 or ℓ_2 norm) can generate an even sparser w, which seems favorable in high-dimensional problems. However, it is rarely used. Why?

ℓ_0 regularization is rarely used because ℓ_0 regularization loss function is not continuous (the function is flat and discontinuous), making it very difficult to optimize over.

Implementation of ℓ_2 regularization:

We are going to experiment with regression for the Red Wine Quality Rating data set. The data set is uploaded on the course website, and you can read more about it here: `https://archive.ics.uci.edu/ml/datasets/Wine`. The data relates 13 different factors (last 13 columns) to wine type (the first column). Each column of data represents a different factor, and they are all continuous features. Note that the original data set has three classes, but one was removed to make this a binary classification problem.

Download the data for training and testing. There are two training sets, wine_training1.txt (100 data points) and wine_training2.txt (a proper subset of wine_training1.txt containing only 40 data points), and one test set, wine_testing.txt (30 data points). You will use the wine_testing.txt dataset to evaluate your models.

We will train a *ℓ_2-regularized logistic regression* model on this data. Recall that the unregularized logistic error (a.k.a. log loss) is

$$E = -\sum_{i=1}^{N} \log(p(y_i|\mathbf{x}_i))$$

where $p(y_i = -1|\mathbf{x}_i)$ is

$$\frac{1}{1+e^{\mathbf{w}^T\mathbf{x}_i}}$$

and $p(y_i = 1|\mathbf{x}_i)$ is

$$\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}_i}},$$

where as usual we assume that all $\mathbf{x}_i$ contain a bias term. The ℓ_2-regularized logistic error is

$$\begin{aligned} E &= -\sum_{i=1}^{N} \log(p(y_i|\mathbf{x}_i)) + \lambda\mathbf{w}^T\mathbf{w} \\ &= -\sum_{i=1}^{N} \log\left(\frac{1}{1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}}\right) + \lambda\mathbf{w}^T\mathbf{w} \\ &= -\sum_{i=1}^{N} \left(\log\left(\frac{1}{1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}}\right) - \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}\right). \end{aligned}$$

Implement SGD to train a model that minimizes the ℓ_2-regularized logistic error, i.e. train an ℓ_2-regularized logistic regression model. Train the model with 15 different values of λ starting with $\lambda_0 = 0.00001$ and increasing by a factor of 5, i.e.

$$\lambda_0 = 0.00001, \lambda_1 = 0.00005, \lambda_2 = 0.00025, ..., \lambda_{14} = 61,035.15625.$$

Some important notes: Terminate the SGD process after 20,000 epochs, where each epoch performs one SGD iteration for each point in the training dataset. You should shuffle the order of the points before each epoch such that you go through the points in a random order (hint: use `numpy.random.permutation`). Use a learning rate of $5e-4$, and initialize your weights to small random numbers.

You may run into numerical instability issues (overflow or underflow). One way to deal with these issues is by normalizing the input data X. Given the column for the jth feature, $X_{:,j}$, you can normalize it by setting $X_{ij} = \frac{X_{ij}-\overline{X_{:,j}}}{\sigma(X_{:,j})}$ where $\sigma(X_{:,j})$ is the standard deviation of the jth column's entries, and $\overline{X_{:,j}}$ is the mean of the jth column's entries. Normalization may change the optimal choice of λ; the λ range given above corresponds to data that has been normalized in this manner. If you treat the input data differently, simply plot enough choices of λ to see any trends.

Problem C [16 points]: Do the following for both training data sets (wine_training1.txt and wine_training2.txt) and attach your plots in the homework submission (use a log-scale on the horizontal axis):

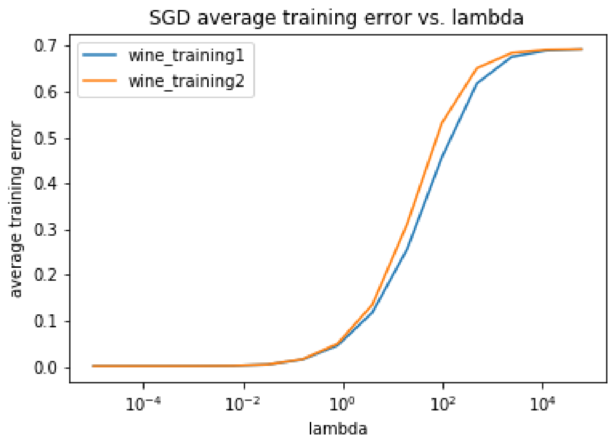
i. Plot the average training error (E_{in}) versus different λs.

ii. Plot the average test error (E_{out}) versus different λs using wine_testing.txt as the test set.

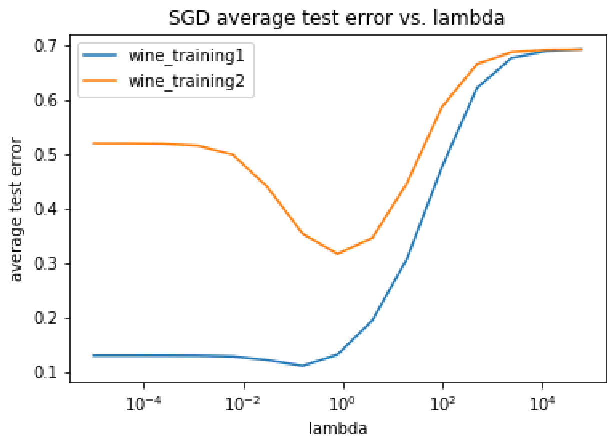
iii. Plot the ℓ_2 norm of $\mathbf{w}$ versus different λs.

You should end up with three plots, with two series (one for wine_training1.txt and one for wine_training2.txt) on each plot. Note that the E_{in} and E_{out} values you plot should not include the regularization penalty — the penalty is only included when performing gradient descent.

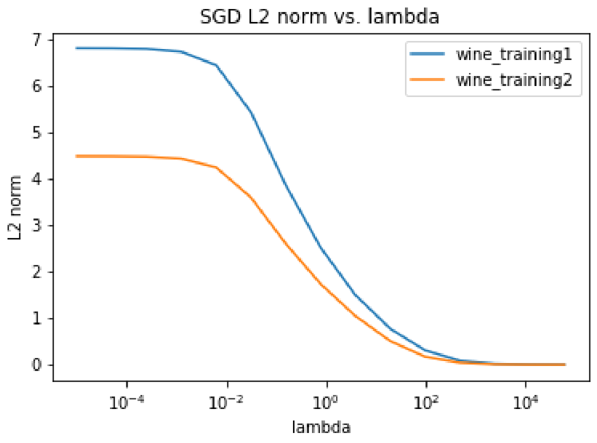
Plot of the average training error (E_{in}) versus different λs:



Plot of the average test error (E_{out}) versus different λs:



Plot of the ℓ_2 norm of $\mathbf{w}$ versus different λs:



Problem D [4 points]: Given that the data in wine_training2.txt is a subset of the data in wine_training1.txt, compare errors (training and test) resulting from training with wine_training1.txt (100 data points) versus wine_training2.txt (40 data points). Briefly explain the differences.

For the wine_training1.txt and wine_training2.txt training errors, over all λs, the training errors are very close if not about equal to each other. This is because SGD is able to find near-zero error with both training sets, and for large λs, the regularization term in the SGD gradient dominates the behavior of SGD resulting in similar training errors between the two datasets for large λs ($\lambda \geq 1$).

For the wine_training1.txt and wine_training2.txt test errors, for $\lambda < 1000$, wine_training2.txt has higher test errors than wine_training1.txt. This is because wine_training2.txt contains less data than wine_training1.txt, resulting in SGD overfitting to the limited data in wine_training2.txt.

Problem E [4 points]: Briefly explain the qualitative behavior (i.e. over-fitting and under-fitting) of the training and test errors with different λs while training with data in wine_training1.txt.

For training errors (with the dataset wine_training1.txt) corresponding to large λs, under-fitting occurs, because as λ increases, the bias of the model increases, causing the training and test errors for large λs to increase as well. For $\lambda < 0.15625 = \lambda_6$, slight over-fitting occurs, because while training error for wine_training1.txt for these λ values is very close to zero, test errors for these λ values are slightly higher than that of $\lambda = 0.15625$.

Problem F [4 points]: Briefly explain the qualitative behavior of the ℓ_2 norm of **w** with different λs while training with the data in wine_training1.txt.

The ℓ_2 norm of **w** while training with the data in wine_training1.txt is constant for small λs ($\lambda < 0.001$), then the ℓ_2 norm of **w** decreases exponentially for increasing λ for $\lambda >= 0.001$.

Problem G [4 points]: If the model were trained with wine_training2.txt, which λ would you choose to train your final model? Why?

If the model were trained with wine_training2.txt, $\lambda = \lambda_7 = 0.78125$ would be choosen to train the final model because this value of lambda results in the model having the minimum test error when trained with the wine_training2.txt dataset.

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

Relevant materials: Lecture 3

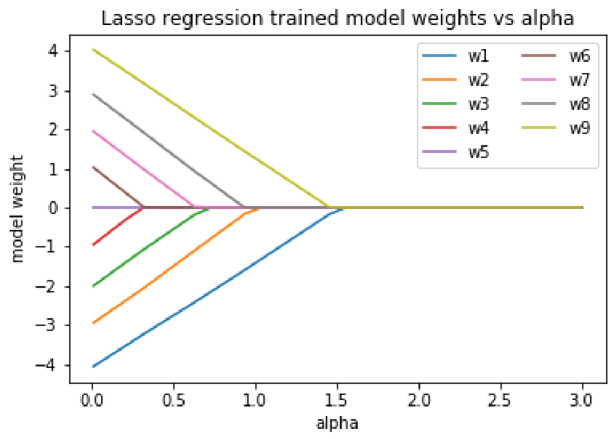
For this problem, you may use the scikit-learn (or other Python package) implementation of Lasso and Ridge regression — you don't have to code it yourself.

The two most commonly-used regularized regression models are Lasso (ℓ_1) regression and Ridge (ℓ_2) regression. Although both enforce "simplicity" in the models they learn, only Lasso regression results in sparse weight vectors. This problem compares the effect of the two methods on the learned model parameters.

Problem A [12 points]: The tab-delimited file problem3data.txt on the course website contains 1000 9-dimensional datapoints. The first 9 columns contain $x_1, \ldots, x_9$, and the last column contains the target value y.

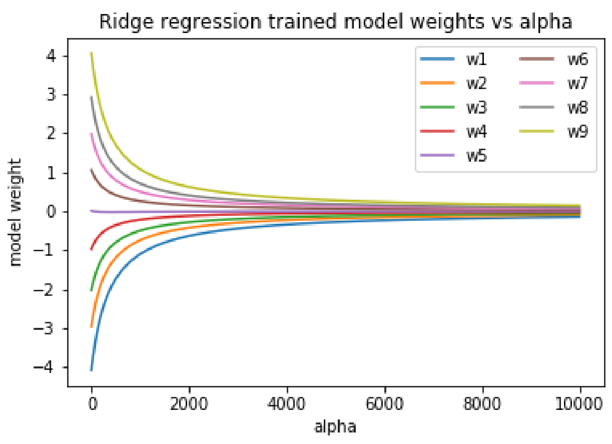
i. Train a linear regression model on the problem3data.txt data with Lasso regularization for regularization strengths α in the vector given by `numpy.linspace(0.01, 3, 30)`. On a single plot, plot each of the model weights $w_1, ..., w_9$ (ignore the bias/intercept) as a function of α.

Plot of each of the model weights $w_1, ..., w_9$ as a function of α (with Lasso regression):



ii. Repeat **i.** with Ridge regression, and this time using regularization strengths $\alpha \in \{1, 2, 3, \ldots, 1e4\}$.

Plot of each of the model weights $w_1, ..., w_9$ as a function of α (with Ridge regression):



iii. As the regularization parameter increases, what happens to the number of model weights that are exactly zero with Lasso regression? What happens to the number of model weights that are exactly zero with Ridge regression?

As the regularization parameter increases, the number of model weights that are exactly zero increases with Lasso regression.

As the regularization parameter increases, the number of model weights that are exactly zero mostly stays constant with Ridge regression (w_5 starts out nonzero at $\alpha = 0$ and then becomes 0 before $\alpha = 1000$, but this is the only weight that becomes 0 for Ridge regression, although all the model weights approach zero as α increases).

Problem B [18 points]:

i. In the case of 1-dimensional data, Lasso regression admits a closed-form solution. Given a dataset containing N datapoints each with d features, where $d = 1$, solve for

$$\arg\min_w \|\mathbf{y} - \mathbf{x}w\|^2 + \lambda\|w\|_1,$$

where $\mathbf{x} \in \mathbb{R}^N$ is the vector of datapoints and $\mathbf{y} \in \mathbb{R}^N$ is the vector of all output values corresponding to these datapoints. Just consider the case where $d = 1$, $\lambda \geq 0$, and the weight w is a scalar.

This is linear regression with Lasso regularization.

To solve for $\arg\min_w \|\mathbf{y} - \mathbf{x}w\|^2 + \lambda\|w\|_1$, set the derivative of the expression $\|\mathbf{y} - \mathbf{x}w\|^2 + \lambda\|w\|_1$ with respect to w equal to 0, since the minimum value of the given expression is at a location such that the derivative at the location with respect to w is 0.

$$\|\mathbf{y} - \mathbf{x}w\|^2 + \lambda\|w\|_1 = (\mathbf{y} - \mathbf{x}w)^T(\mathbf{y} - \mathbf{x}w) + \lambda\|w\|_1 = (\mathbf{y}^T - \mathbf{x}^T w)(\mathbf{y} - \mathbf{x}w) + \lambda\|w\|_1$$

$$\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{x}w - \mathbf{x}^T w\mathbf{y} + \mathbf{x}^T\mathbf{x}w^2 + \lambda\|w\|_1.$$

Therefore,

$$\frac{\partial}{\partial w}(\|\mathbf{y}-\mathbf{x}w\|^2+\lambda\|w\|_1)=0 \implies \frac{\partial}{\partial w}(\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{x}w-\mathbf{x}^Tw\mathbf{y}+\mathbf{x}^T\mathbf{x}w^2+\lambda\|w\|_1)=0$$
$$\implies 0-\mathbf{y}^T\mathbf{x}-\mathbf{x}^T\mathbf{y}+2\mathbf{x}^T\mathbf{x}w+\lambda\text{sign}(w)=0.$$

Since $\mathbf{y}^T\mathbf{x}=\mathbf{x}^T\mathbf{y}$,

$$0-\mathbf{y}^T\mathbf{x}-\mathbf{x}^T\mathbf{y}+2\mathbf{x}^T\mathbf{x}w+\lambda\text{sign}(w)=0 \implies -2\mathbf{y}^T\mathbf{x}+2\mathbf{x}^T\mathbf{x}w+\lambda\text{sign}(w)=0 \implies 2\mathbf{x}^T\mathbf{x}w=2\mathbf{y}^T\mathbf{x}-\lambda\text{sign}(w)$$
$$\implies w=\frac{2\mathbf{y}^T\mathbf{x}-\lambda\text{sign}(w)}{2\mathbf{x}^T\mathbf{x}},$$

where

$$\text{sign}(w)=\begin{cases}-1 & w<0\\ 1 & w>0\\ [-1,1] & w=0\end{cases}.$$

ii. In this question, we continue to consider Lasso regularization in 1-dimension. Now, suppose that when $\lambda=0$, $w\neq 0$. Does there exist a value for λ such that $w=0$? If so, what is the smallest such value?

Using the expression for w obtained in part i and setting $w>0$,

$$w=\frac{2\mathbf{y}^T\mathbf{x}-\lambda(1)}{2\mathbf{x}^T\mathbf{x}} \implies w=\frac{2\mathbf{y}^T\mathbf{x}-\lambda}{2\mathbf{x}^T\mathbf{x}}.$$

Since $w>0$ and $\mathbf{x}^T\mathbf{x}>0$, $2\mathbf{y}^T\mathbf{x}-\lambda>0 \implies 2\mathbf{y}^T\mathbf{x}>\lambda$.

Using the expression for w obtained in part i and setting $w<0$,

$$w=\frac{2\mathbf{y}^T\mathbf{x}-\lambda(-1)}{2\mathbf{x}^T\mathbf{x}} \implies w=\frac{2\mathbf{y}^T\mathbf{x}+\lambda}{2\mathbf{x}^T\mathbf{x}}.$$

Since $w<0$ and $\mathbf{x}^T\mathbf{x}>0$, $2\mathbf{y}^T\mathbf{x}+\lambda>0 \implies 2\mathbf{y}^T\mathbf{x}<-\lambda$.

Since $2\mathbf{y}^T\mathbf{x}>\lambda$ for $w>0$ and $2\mathbf{y}^T\mathbf{x}<-\lambda$ for $w<0$, for $w=0$, $-\lambda\leq 2\mathbf{y}^T\mathbf{x}\leq\lambda \implies \lambda\geq 2|\mathbf{y}^T\mathbf{x}|$. Therefore, the smallest value of λ such that $w=0$ is $\lambda=2|\mathbf{y}^T\mathbf{x}|$.

iii. Given a dataset containing N datapoints each with d features, solve for

$$\arg\min_{\mathbf{w}}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2+\lambda\|\mathbf{w}\|_2^2$$

where $\mathbf{X}\in\mathbb{R}^{N\times d}$ is the matrix of datapoints and $\mathbf{y}\in\mathbb{R}^N$ is the vector of all output values for these datapoints. Do so for arbitrary d and $\lambda\geq 0$.

This is linear regression with Ridge regularization.

To solve for $\arg\min_{\mathbf{w}}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2+\lambda\|\mathbf{w}\|_2^2$, set the derivative of the expression $\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2+\lambda\|\mathbf{w}\|_2^2$ with respect to $\mathbf{w}$ equal to 0, since the minimum value of the given expression is at a location such that the derivative at the location with respect to $\mathbf{w}$ is 0.

$$\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2+\lambda\|\mathbf{w}\|_2^2=(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w})+\lambda\mathbf{w}^T\mathbf{w}=(\mathbf{y}^T-\mathbf{w}^T\mathbf{X}^T)(\mathbf{y}-\mathbf{X}\mathbf{w})+\lambda\mathbf{w}^T\mathbf{w}$$

$$\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}^T\mathbf{w}.$$

Using the matrix derivative rules $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T\mathbf{a}) = \frac{\partial}{\partial \mathbf{w}}(\mathbf{a}^T\mathbf{w}) = \mathbf{a}$ and $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T\mathbf{A}\mathbf{w}) = \mathbf{A}^T\mathbf{w} + \mathbf{A}\mathbf{w} = (\mathbf{A}^T + \mathbf{A})\mathbf{w}$,

$$\frac{\partial}{\partial \mathbf{w}}(\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|_2^2) = 0 \implies \frac{\partial}{\partial \mathbf{w}}(\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}^T\mathbf{w}) = 0$$

$$\implies 0 - \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w} + 2\lambda\mathbf{w} = 0 \implies \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y} \implies (\mathbf{X}^T\mathbf{X} + \lambda I)\mathbf{w} = \mathbf{X}^T\mathbf{y},$$

where I is the identity matrix (with dimensions equal to those of $\mathbf{X}^T\mathbf{X}$). By the invertible matrix theorem, $(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}$ exists. Therefore, $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$.

iv. In this question, we consider Ridge regularization in 1-dimension. Suppose that when $\lambda = 0$, $w \neq 0$. Does there exist a value for $\lambda > 0$ such that $w = 0$? If so, what is the smallest such value?

From part iii, in 1-dimension, $w = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$. Given that when $\lambda = 0$, $w \neq 0$,
$w = (\mathbf{X}^T\mathbf{X} + (0)I)^{-1}\mathbf{X}^T\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. By the invertible matrix theorem, $(\mathbf{X}^T\mathbf{X})^{-1}$ has an inverse, so $(\mathbf{X}^T\mathbf{X})^{-1}$ is a nonzero matrix. Since w is nonzero and $(\mathbf{X}^T\mathbf{X})^{-1}$ is nonzero in $w = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, $\mathbf{X}^T\mathbf{y}$ must also be nonzero. For $\lambda > 0$, by the invertible matrix theorem, $(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}$ has an inverse, so $(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}$ must be nonzero, and since $\mathbf{X}^T\mathbf{y}$ is also nonzero, $w = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$ must also be nonzero for all $\lambda > 0$. Therefore, there does not exist a value for $\lambda > 0$ such that $w = 0$.