CS 335
Assignment 2

Aakriti

Due 26th September 2021

Contents

1 Perceptron **2**
1.1 CS 337: Theory . . . 2
(1) . . . 2
(2) . . . 2
(3) . . . 3
(4) . . . 4
(5) . . . 5
1.2 CS 335: Lab . . . 5

2 LASSO and ISTA **7**
2.1 CS 337: Theory . . . 7
(1) . . . 7
(2) . . . 8
(3) . . . 9
2.2 CS 337: Lab . . . 11
(1) . . . 11
(2) . . . 12
(3) . . . 12

3 Bias Variance Trade-off **15**
3.1 CS 337: Theory . . . 15
(1) . . . 15
(2) . . . 15
3.2 CS 337: Lab . . . 16
(1) . . . 16
(2) . . . 17
(3) . . . 18

1 Perceptron

A Perceptron keeps a weight vector w_y corresponding to each class y. Given a feature vector f, we score each class y with

$$score(f, y) = \sum_i f_i w_{y_i}$$

where i iterates over the dimensions in the data. Then we choose the class with the highest score as the predicted label for data instance f.

1.1 CS 337: Theory

(1)

1-vs-1 Perceptron compares every class with all the other classes at once and then classifies. So every class is compared with the rest $N-1$ classes. Hence, if there are N classes, then this classification requires $\binom{N}{2}$ binary models for classification.

1-vs-Rest Perceptron tries to compares a point for all the possible classes and see which scores the maximum. One particular class is separated from the rest at a time, while all the other classes are consider to be one and then it follows the simple 2-class classification. If there are N classes, then this classification requires **N** binary models for classification.

Advantage : 1-vs-1 is less prone to creating an imbalance in the dataset due to dominance in specific classes as compared to 1-vs-Rest.
If we have 10 classes and 100 data points belonging to each, in 1-vs-1, both the classes fed into classifier have 100 data points each while in 1-vs-rest classifier, one class has 100 data points while other has 900 so, clear imbalance!

Disadvantage : 1-vs-Rest Perceptron trains on less number of classifiers which makes it faster than the 1-vs-1 Perceptron approach. 1-vs-1 has $\binom{n}{2}$ models to train, which will take more time and resources.
Also, there is no probabilistic interpretation here, unlike 1-vs-rest perceptron.

(2)

A data point (f, y) got misclassified as y'.
The update rule for f we have is the following:

$$w_y = w_y + f$$
$$w_{y'} = w_{y'} - f$$

- **To Prove:** $score(f, y)$ **increases**

$$\begin{aligned} score(f, y) &= \sum_i f_i w_{y_i} = \sum_i f_i (w_{y_i} + f_i) \\ &= \sum_i f_i w_{y_i} + \sum_i f_i^2 \\ &> \sum_i f_i w_{y_i} \end{aligned}$$

As we calculated, *updated* $score(f, y) = \sum_i f_i w_{y_i} + \sum_i f_i^2$. Here, the first term $(\sum_i f_i w_{y_i})$ corresponds to the old $score(f, y)$ value and the second term $(\sum_i f_i^2)$ is a positive quantity which makes the value of updated score $score(f, y)$ greater than the old score.
Thus, $score(f, y)$ increases.

- **To Prove:** $score(f, y')$ **decreases**

$$\begin{aligned} score(f, y') &= \sum_i f_i w_{y'_i} \\ &= \sum_i f_i (w_{y'_i} - f_i) \\ &= \sum_i f_i w_{y'_i} - \sum_i f_i^2 \end{aligned}$$

As we calculated, *updated* $score(f, y') = \sum_i f_i w_{y'_i} - \sum_i f_i^2$. Here, the first term $(\sum_i f_i w_{y_i})$ corresponds to the old $score(f, y)$ value and the second term $(\sum_i f_i^2)$ is a positive quantity which makes the value of updated score $score(f, y)$ lesser than the old score.
Thus, $score(f, y')$ decreases.

(3)

Assume a 2-class dataset with $y \in \{-1, +1\}$. Let us define the loss function as $\mathcal{L}(f, y) = max(0, -yf^T w)$ and learn w using gradient descent algorithm.

Gradient Update Rule

The gradient of loss function wrt w is calculated as follows:

- Case (i): $yf^T w > 0$, i.e., point is classified correctly

$$\mathcal{L}(f, y) = 0 \implies \nabla_w \mathcal{L}(f, y) = 0$$

- Case (i): $yf^T w < 0$, i.e., point is misclassified

$$\mathcal{L}(f, y) = -yf^T w \implies \nabla_w \mathcal{L}(f, y) = -yf$$

Therefore, the update rule is given as-

$$w \leftarrow w - \nabla_w \mathcal{L}(f, y)$$

$$w \leftarrow w + yf^T \qquad if \;\; yf^T w < 0$$

Proof of convergence

We are given that the dataset is given linearly separable. We need to show the convergence, which we will proof using the upper bound on the number of maximum iterations that will take for the algorithm to obtain the optimal results. Note, that after taking $\eta = 1$, this gradient update rule is the same as we have done in class, so, we prove the same theorem which were presented in slides.

As the data is linearly separable, there exists a weight vector u, such that $yu^T f > 0$. WLOG, we can assume, learning rate $\eta = 1$, $\|u\| = 1$ and $\|f\| \leq 1 \;\; \forall f \in \mathcal{D}$. Thus, u is a unit length vector and all data points are scaled to lie the Euclidean ball of radius 1. Now, we prove the following theorem.

Let for dataset $\mathcal{D}, \gamma = min_{x \in \mathcal{D}} |u^T f|$

Theorem: If there exist a u such that $\forall y, f \in \mathcal{D}$, $yu^T f > 0$, then the number of weight updates made by the perceptron algorithm is at most $\dfrac{1}{\gamma^2}$

We observe two quantities $\|w\|^2$ and $w^T u$

Let's check $w_{i+1}^T u$ for a positive misclassified point at iteration i.

$$\begin{aligned} w_{i+1}^T u &= (w_i + f)^T u \\ (w_i + f)^T u &= w_i^T u + f^T u \\ (w_i + f)^T u &\geq w_i^T u + \gamma \end{aligned}$$

Let's check $\|w_{i+1}^T\|^2$ for a positive misclassified point at iteration i.

$$\begin{aligned} \|w_{i+1}\|^2 &= \|(w_i + f)\|^2 \\ &= (w_i + f)^T (w_i + f) \\ &= \|w_i\|^2 + \|f\|^2 + 2w^T f \\ &\leq \|w_i\|^2 + 1 \end{aligned}$$

This is because, we know, $2w^T f < 0$ and $\|f\|^2 < 1$, we have $\|f\|^2 + 2w^T f < 1$

Hence, after k updates we have: $\|w_k\|^2 \leq k$ and $u^T w_k \geq k\gamma$

Now, we apply the Cauchy Schwarz Inequality as follows:

$$\begin{aligned} u^T w_k &\leq \|u\| \times \|w_k\| \\ k\gamma &\leq 1 \times \sqrt{k} \\ \sqrt{k}\gamma &\leq 1 \\ k &\leq \frac{1}{\gamma^2} \end{aligned}$$

Thus, proved!

(4)

If we change the perceptron update rules as follows:

$$\begin{aligned} w_y &= w_y + 0.5f \\ w'_y &= w'_y - 0.5f \end{aligned}$$

Previously, we had considered $\eta = 1$, reiterating proof for general η as follows -

Let's check $w_{i+1}^T u$ for a positive misclassified point at iteration i.

$$\begin{aligned} w_{i+1}^T u &= (w_i + \eta f)u \\ (w_i + f)^T u &\geq w_i^T u + \eta\gamma \end{aligned}$$

Let's check $\|w_{i+1}^T\|^2$ for a positive misclassified point at iteration i.

$$\begin{aligned} \|w_{i+1}^T\|^2 &= \|(w_i + \eta f)\|^2 \\ &= \|w_i\|^2 + \eta^2\|f\|^2 + 2\eta w^T f \\ &\leq \|w_i\|^2 + \eta^2 \end{aligned}$$

Again, we know, $2w^T f < 0$ and $\|f\|^2 < 1$, we have $\|f\|^2 + 2w^T f < 1$.

Hence, after k updates we have: $\|w_k\|^2 \leq k\eta^2$ and $u^T w_k \geq k\eta\gamma$

Now, we apply the Cauchy Schwarz Inequality as follows:

$$\begin{aligned} u^T w_k &\leq \|u\| \times \|w_k\| \\ k\eta\gamma &\leq 1 \times \sqrt{k}\eta \\ \sqrt{k}\gamma &\leq 1 \\ k &\leq \frac{1}{\gamma^2} \end{aligned}$$

The upper bound on number of iterations will **NOT change**.

(5)

To calculate: Upper bound on number of iterations for the `AND` dataset given by the AND function.

Let us derive the maximum iterations step by step :

(i) The data points are contained in a sphere of radius $\sqrt{3}$ including bias

(ii) The separating line which ensures AB is the minimum distance to the data points from the separating line is u = (2, 2, -3).
Normalizing the weight vector we get :

$$u = \frac{1}{\sqrt{17}}(2, 2, -3)$$

(iii) Now, $min\|u^T x\| = \frac{1}{\sqrt{17}}$

Therefore,

$$(w + x)^T u \geq w^T u + \lambda$$

This gives :

$$\begin{aligned} u^T w^k &\geq k\lambda \\ \|w + x\|_2^2 &= \|w\|_2^2 + 3 \\ \|w^k\|_2^2 &\leq 3k \end{aligned}$$

$\therefore$ we have, $\sqrt{3k} \geq \|w^k\| \geq u^T w^k \geq k\sqrt{17}$

Thus $\boxed{k \leq 51}$
Maximum number of iterations are 51.

1.2 CS 335: Lab

The `perceptron()` algorithm completed. Classification report presented below -

	precision	recall	f1-score	support
class 0	1.00	1.00	1.00	15
class 1	1.00	1.00	1.00	12
class 2	1.00	1.00	1.00	13
accuracy			1.00	40
macro avg	1.00	1.00	1.00	40
weighted avg	1.00	1.00	1.00	40

2 LASSO and ISTA

2.1 CS 337: Theory

(1)

To Prove: LASSO is the MAP estimate of Linear regression subject to the Laplacian prior on weights.
LASSO cost function -

$$\mathcal{L}(w) = \left(\sum_{i=1}^{n} \|y - x_i \cdot w\|_2^2 + \lambda \|w\|_1^1 \right)$$

LASSO estimation of w -

$$\hat{w}_{LASSO} = \arg\min_w \left(\sum_{i=1}^{n} \|y - x_i \cdot w\|_2^2 + \lambda \|w\|_1^1 \right) \tag{1}$$

Consider $\mathcal{D} \triangleq x_i, y_i$ where $i \in [n]$ and $y_i = f(x_i) + \varepsilon$. Here, $f(\cdot)$ is the true function, and we assume gassian noise, $\varepsilon \sim \mathcal{N}(0, \sigma)$.
We have, likelihood as -

$$P(y_i|w, x_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y_i - x_i \cdot w)^2}{2\sigma^2} \right)$$

$$P(D|w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y_i - x_i \cdot w)^2}{2\sigma^2} \right) \tag{2}$$

We also have laplacian prior on w with mean μ and variance $2b^2$. Therefore,

$$P(w|b) = \frac{1}{2b} \exp\left(\frac{-|w - \mu|}{b} \right) \tag{3}$$

From (1) and (2), posterior is given by -

$$P(w|D) \propto P(D|w) \times P(w)$$
$$P(w|D) \propto \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y_i - x_i \cdot w)^2}{2\sigma^2} \right) \times \frac{1}{2b} \exp\left(\frac{-|w - \mu|}{b} \right)$$

For the Laplace prior on weights, we take the assumption that $\mu = 0$.

The negative log likelihood is given by -

$$-log(P(w|D) = \sum_{i=1}^{n} \frac{(y_i - x_i \cdot w)^2}{2\sigma^2} + \frac{\|w\|_1}{b} + constants$$

Now, we know that,

$$\begin{aligned}
\hat{w}_{MAP} &= \arg\max_w P(w|D) \\
&= \arg\max_w log(P(w|D)) \\
&= \arg\min_w -log(P(w|D)) \\
&= \arg\min_w \left(\sum_{i=1}^{n} \frac{(y_i - x_i \cdot w)^2}{2\sigma^2} + \frac{\|w\|_1}{b} + constants \right)
\end{aligned}$$

Ignoring constants and refactoring the equation we get -

$$\hat{w}_{MAP} = \arg\min_{w} \left(\sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 + \frac{2\sigma^2}{b} \|w\|_1^1 \right) \tag{4}$$

Taking $\lambda = \dfrac{2\sigma^2}{b}$,

$$\hat{w}_{MAP} = \arg\min_{w} \left(\sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 + \lambda \|w\|_1^1 \right) \tag{5}$$

The above equation minimizes the LASSO cost function. That is, on comparing equations (1) and (5), we get that $\boxed{\hat{w}_{LASSO} = \hat{w}_{MAP}}$ Hence, proved!

(2)

The ridge regression cost function is given by -

$$w = \arg\min_{w} \sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 + \lambda \|w\|_2^2$$

The lasso cost function is given by -

$$w = \arg\min_{w} \sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 + \lambda \|w\|_1^1$$

Since, the only difference is in the regularization term, we can rewrite these penalised cost cost functions as constraint cost functions. That is,

$$w_{Lasso} = \arg\min_{w} \|w\|_1 \;\; s.t. \sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 < \epsilon_1$$

$$w_{Ridge} = \arg\min_{w} \|w\|_2 \;\; s.t. \sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 < \epsilon_2$$

for some $\epsilon_1 > 0$ and $\epsilon_2 > 0$.

Now, since for both, equation under constraint is the same, we can take both ϵ_1 and ϵ_2 to be the same ϵ.
We consider $\sum_{i=1}^{n} \|y_1 - x_i \cdot w\| < \epsilon$. Let us assume it looks the following in w space -

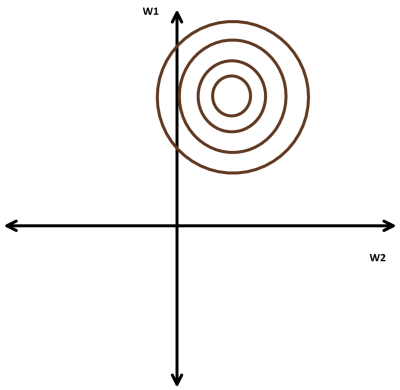


Figure 1: ϵ in w space

Now, the contours of $\|w\|_1$ and $\|w\|_2$ would look like -

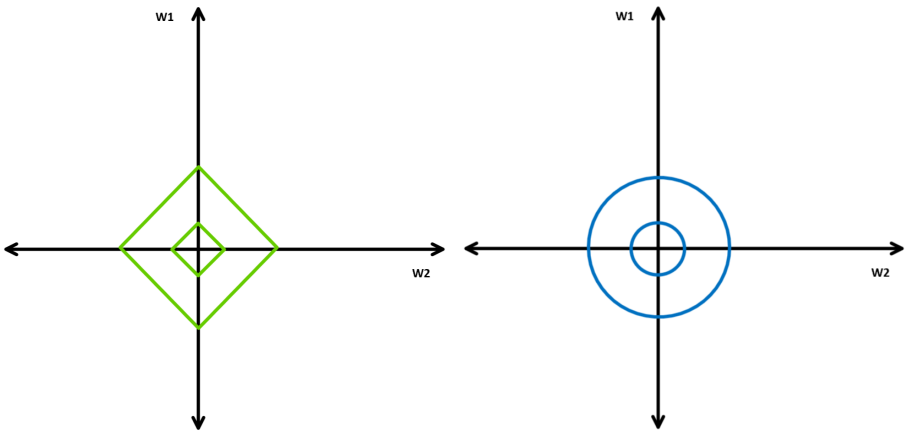


Figure 2: $\|w\|_1$ and $\|w\|_2$ in w space

Now, $\|w\|_1$ and $\|w\|_2$ are minimized with the given constraint only when the green and blues curves touch the brown curve of ϵ. It looks like -

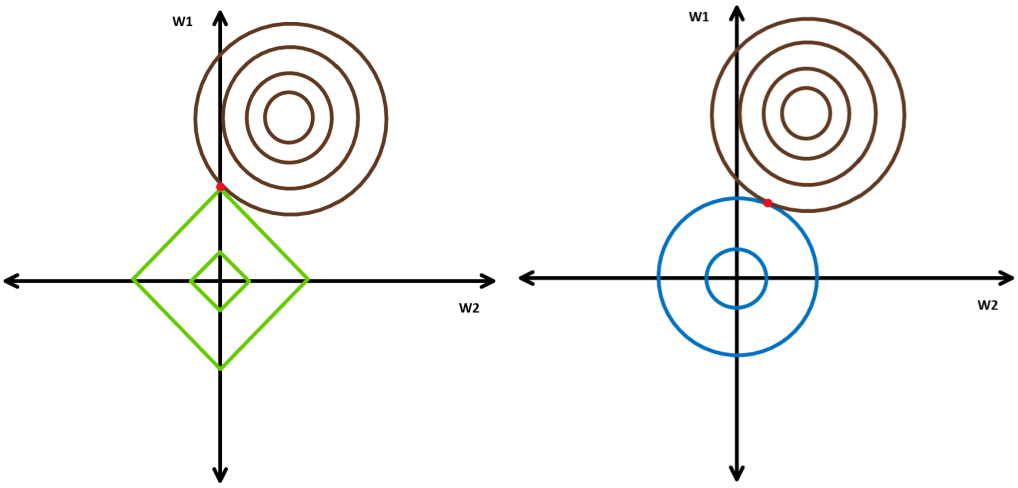


Figure 3: $\|w\|_1$ and $\|w\|_2$ touching ϵ in w space

Notice that the solution of LASSO has a tendency to lie on the axes due the pointy nature of $\|x\|_1$ contours in contrast to the bulgy contours of $\|x\|_2$. This is the reason for sparse solutions in LASSO. In conclusion, ridge regression shrinks all regression coefficients to be close to zero; the lasso tends to give a set of zero regression coefficients and leads to a sparse solution.

The penalty term associated with a Lasso regression forms a square constraint region for sets of coefficients. Because the RSS contours are likely to intercept the constraint region at the extremas we are likely to see certain coefficients remain larger and others be forced to zero.

(3)

In general, the LASSO lacks a closed form solution because the objective function is not differentiable. However, it is possible to obtain closed form solutions for the special case of an orthonormal design matrix i.e when $X^T X = I$

$$L(w) = \sum_{i=1}^{n} \|y_i - x_i \cdot w\|_2^2 + \lambda \|w\|_1^1$$
$$\frac{\partial L(w)}{\partial w} = -X^T(Y - X \cdot w) + \lambda sign(w) = 0$$

Now, $\frac{\partial \|w\|_1}{\partial w}$ is defined when any entry of w is not 0

We have, $X^TY = (X^TX)w + \lambda sign(w)$

This equation can be solved when $X^TX = I$ i.e., X is orthogonal.

Suppose $X^TX = I$, then $X^TY = w + \lambda sign(w)$

The solution to $y = x + \lambda sign(x)$ is given by -

$$x = soft(y; \lambda)$$

where,

$$soft(z; \nu) = \begin{cases} z - \nu & z \geq \nu \\ z + \nu & z \leq \nu \\ 0 & |z| < \nu \end{cases}$$

$\therefore$ **Closed form of Lasso exists when $X^TX = I$ and it is given by** $\boxed{w = soft(X^TY; \lambda)}$

Note:

We had, $X^TY = (X^TX)w + \lambda sign(w)$. When X^TX is diagonal, we have -

$$\forall i \in [n], \quad X^TY[i] = (X^TX)[i]w[i] + \lambda sign(w[i])$$

If $\forall i \in [n], \quad X^TX[i] \neq 0$, we can write -

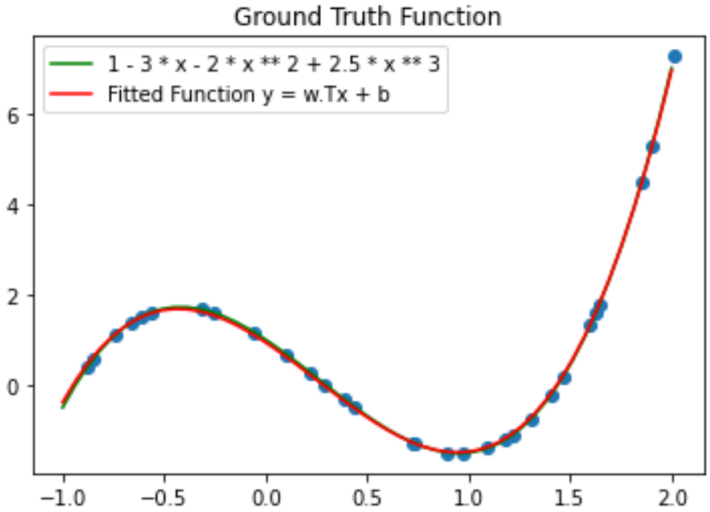
$$\forall i \in [n], \quad \frac{X^TY[i]}{(X^TX)[i]} = w[i] + \frac{\lambda}{(X^TX)[i]} sign(w[i])$$
$$\implies \forall i \in [n], \quad w[i] = soft\left(\frac{X^TY[i]}{(X^TX)[i]}, \frac{\lambda}{(X^TX)[i]}\right)$$

$\therefore$ Closed form also exists when X^TX is diagonal and $\forall i \in [n], \quad X^TX[i] \neq 0$.

2.2 CS 337: Lab

(1)

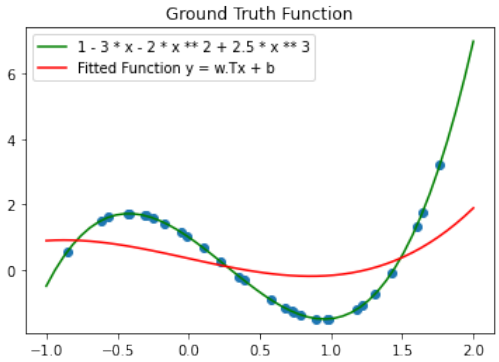
```
Validation loss if 0.002257888051649537
Training Loss loss if 0.0016779944173161564
(5,) 0.9360936906328226 (30, 5) (30,)
```



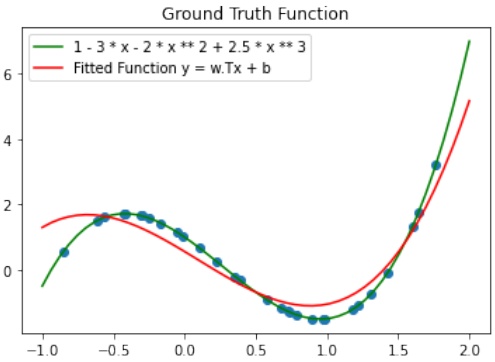
Parameters : lambda $\lambda = 0.01$, epochs $= 1000$

Figure 4: **Best plot using ISTA for LASSO**

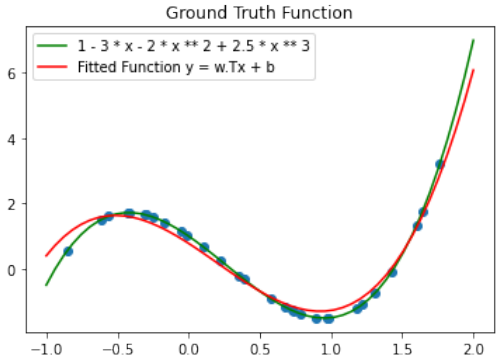
Some other plots at different values of λ -



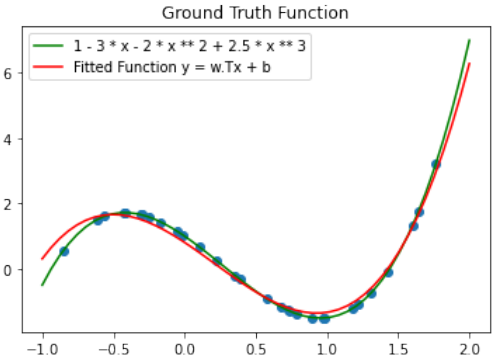
(a) $\lambda = 0.5$, epochs $= 1000$



(b) $\lambda = 0.1$, epochs $= 1000$



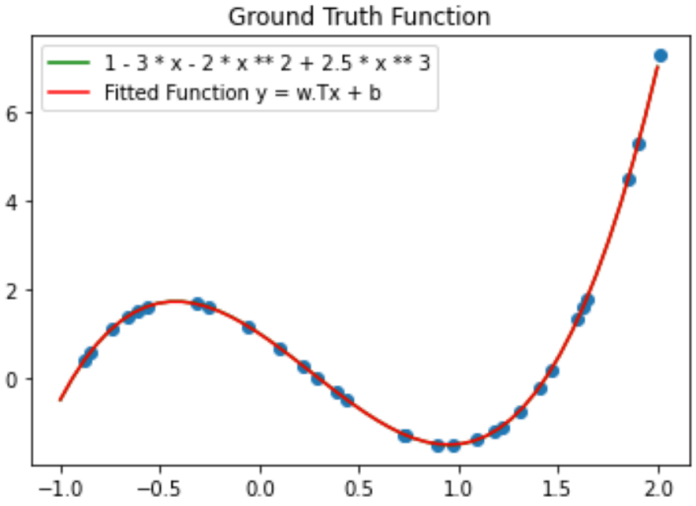
(a) $\lambda = 0.05$, epochs $= 1000$



(b) $\lambda = 0.025$, epochs $= 1000$

(2)

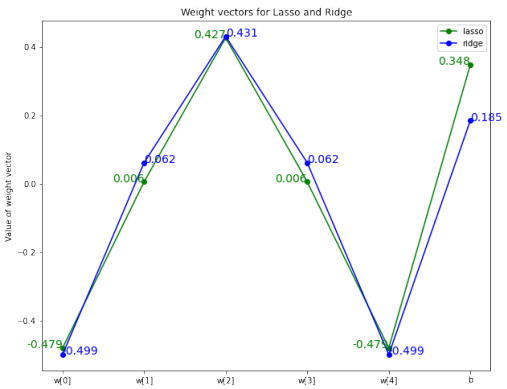
```
Validation loss if 1.751109722161578e-05
Training Loss loss if 1.3756077672482561e-05
```



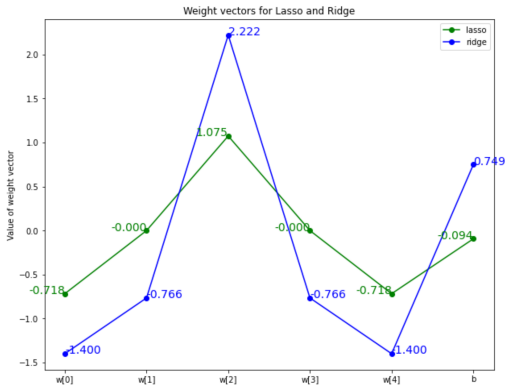
Parameters : lambda $\lambda = 0.01$

Figure 7: **Best plot for closed form for Ridge regression**

(3)



(a) lambda $\lambda = 0.01$, epochs $= 1000$



(b) lambda $\lambda = 0.5$, epochs $= 1000$

At low values of λ i.e. $\lambda \sim 0.01$, we didn't observe any w_i to be zero in both ridge and lasso and both gives can fit the given function with very low error. As we increase the value of λ i.e. $\lambda \approx 0.5$, we can see that lasso and ridge perfoem differently and both give poorer fit for thid=s dataset. We can compare ridge and lasso in feature selection, as increasing λ gives more weight to regularization term, for ridge regression, only the magnitude of some coefficients get affected, but for lasso the magnitude of some coefficients becomes 0 resulting stricter feature selection. this can be seen in the figures below where the the weight vector is printed below the curve plot.

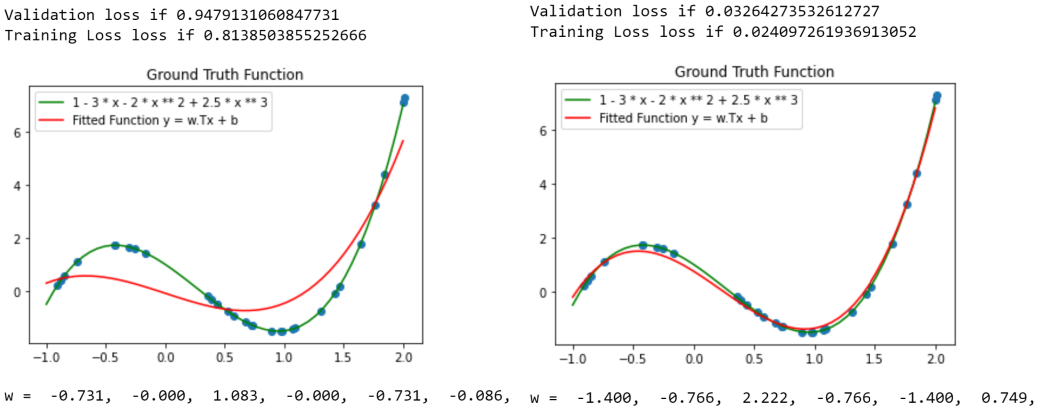


Figure 9: Lasso and Ridge Functions respectively with $\lambda = 0.5$

We can clearly observe that in case of Lasso, 2 values of weight vector turn out to 0 but this does not happen in ridge regression case.

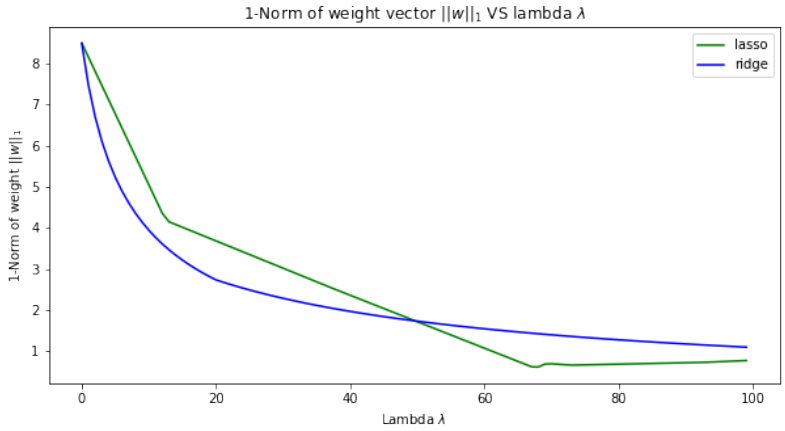


Figure 10: 1-norm of w ($\|w\|_1$) VS λ

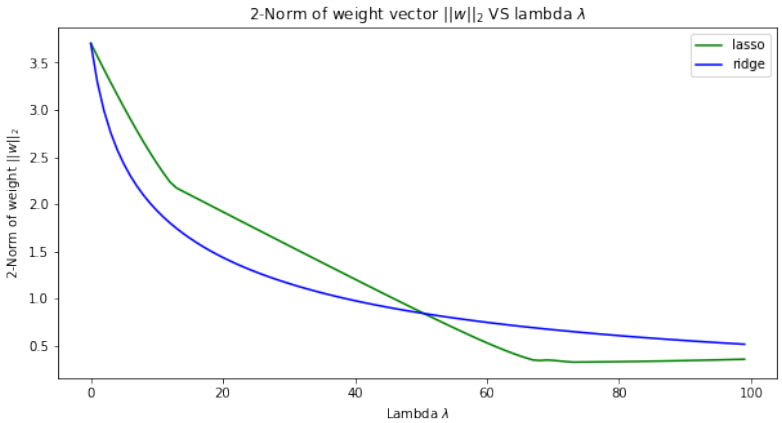


Figure 11: 2-norm of w ($\|w\|_2$) VS λ

The above plots also show that as lambda λ increases, Lasso drives some of the values of weight vector w to 0, while Ridge decreases the magnitude of every value w.

3 Bias Variance Trade-off

3.1 CS 337: Theory

(1)

(a) When lambda λ increases

- bias *increases*
- variance *decreases*

When lambda λ increases in LASSO Regression then the weights are penalised more and hence the model complexity does not increase much. More importance is given to the regularisation term in comparison to noise, so model becomes less susceptible to noise and becomes flatter. However, as λ becomes larger and larger, the coefficients will be smaller than they ought to be to get the best predictive accuracy. All the coefficients are forced to be arbitrarily close to 0 regardless of the data, so the model is maximally underfit. This leads to increase in bias due to poorer predictions by model. However, in this case the model would generalize better leading to decrease in variance.

(b) When model complexity increases

- bias *decreases*
- variance *increases*

Increasing model complexity by by adding more features of high degree leads to overfitting of the data and hence the model has a higher variance due to poor prediction on new data. This would however lead to lower bias as the model can learn more complicated features in the dataset (even the noise) and can predict values from the training set with higher accuracy.

(c) When we reduce dimension by choosing only those subset of features which are of more importance

- bias *increases*
- variance *decreases*

Reducing dimension by choosing only those subset of features which are of more importance leads to decrease in model complexity so consequences would be opposite too that of part (b). Trained model will have better generalization on new data which causes the bias of the model to increases as the predicted values on training data would be farther away from the actual values which have noise added however, since the model would generalize better the variance would be lower.

(2)

Given irreducible noise $\epsilon \sim \mathcal{N}(\mu = 0, \sigma^2)$. Let $\hat{f(x)}, x \in$ test Set, be the function which approximates the true function, $f(x)$ so, for x corresponding $y = f(x) + \epsilon$

To Prove : MSE = Variance($\hat{f(x)}$) + Bias($\hat{f(x)}$)$^2 + \sigma^2$

Proof :

$$
\begin{aligned}
\text{MSE} &= \mathbb{E}[(f(\hat{x}) - y)^2] \\
&= \mathbb{E}[(f(\hat{x}))^2 + y^2 - 2f(\hat{x})y] \\
&= \mathbb{E}[(f(\hat{x}))^2] + \mathbb{E}[y^2] - \mathbb{E}[2f(\hat{x})y] \\
&= \mathbb{E}[(f(\hat{x}))^2] + \mathbb{E}[y^2] - 2\mathbb{E}[f(\hat{x})]\mathbb{E}[y] && \because f(\hat{x}) \text{ and } y \text{ are independent} \\
&= \mathbb{E}[(f(\hat{x}))^2] + \mathbb{E}[y^2] - 2\mathbb{E}[f(\hat{x})]f(x) && \because \mathbb{E}[y] = \mathbb{E}[f(x) + \epsilon] = f(x) \ (\textit{constant})
\end{aligned}
$$

Now, using the fact that $\mathbb{E}\Big[(x - \mathbb{E}[x])^2\Big] + \Big(\mathbb{E}[x]\Big)^2 = \mathbb{E}[x^2]$ on $\mathbb{E}[(f(\hat{x}))^2]$ and $\mathbb{E}[y^2]$

$$
\begin{aligned}
\text{MSE} &= \mathbb{E}[(f(\hat{x}))^2] + \mathbb{E}[y^2] - 2\mathbb{E}[f(\hat{x})]f(x) \\
&= \mathbb{E}\Big[\Big(f(\hat{x}) - \mathbb{E}[f(\hat{x})]\Big)^2\Big] + \Big(\mathbb{E}[f(\hat{x})]\Big)^2 + \mathbb{E}\Big[\Big(y - \mathbb{E}[y]\Big)^2\Big] + \Big(\mathbb{E}[y]\Big)^2 - 2\Big(\mathbb{E}[f(\hat{x})]f(x)\Big) \\
&= \mathbb{E}\Big[\Big(f(\hat{x}) - \mathbb{E}[f(\hat{x})]\Big)^2\Big] + \Big(\mathbb{E}[f(\hat{x})]\Big)^2 + \mathbb{E}\Big[\Big(y - f(x)\Big)^2\Big] + \Big(f(x)\Big)^2 - 2\Big(\mathbb{E}[f(\hat{x})]f(x)\Big) \\
&= \mathbb{E}\Big[\Big(f(\hat{x}) - \mathbb{E}[f(\hat{x})]\Big)^2\Big] + \Big(\mathbb{E}[f(\hat{x})]\Big)^2 + \Big(f(x)\Big)^2 - 2\Big(\mathbb{E}[f(\hat{x})]f(x)\Big) + \mathbb{E}\Big[\Big(y - f(x)\Big)^2\Big] \quad \textit{(Rearranging)} \\
&= \mathbb{E}\Big[\Big(f(\hat{x}) - \mathbb{E}[f(\hat{x})]\Big)^2\Big] + \mathbb{E}\Big[\Big(\mathbb{E}[f(\hat{x})] - f(x)\Big)^2\Big] + \mathbb{E}\Big[\Big(y - f(x)\Big)^2\Big] \\
&= \mathbb{E}\Big[\Big(f(\hat{x}) - \mathbb{E}[f(\hat{x})]\Big)^2\Big] + \Big(\mathbb{E}[f(\hat{x})] - f(x)\Big)^2 + \mathbb{E}\Big[\Big(y - f(x)\Big)^2\Big] \quad \because \mathbb{E}[f(\hat{x})] - f(x) \text{ is a constant} \\
&= Variance(f(\hat{x})) + (Bias(f(\hat{x}))^2 + \sigma^2
\end{aligned}
$$

where,

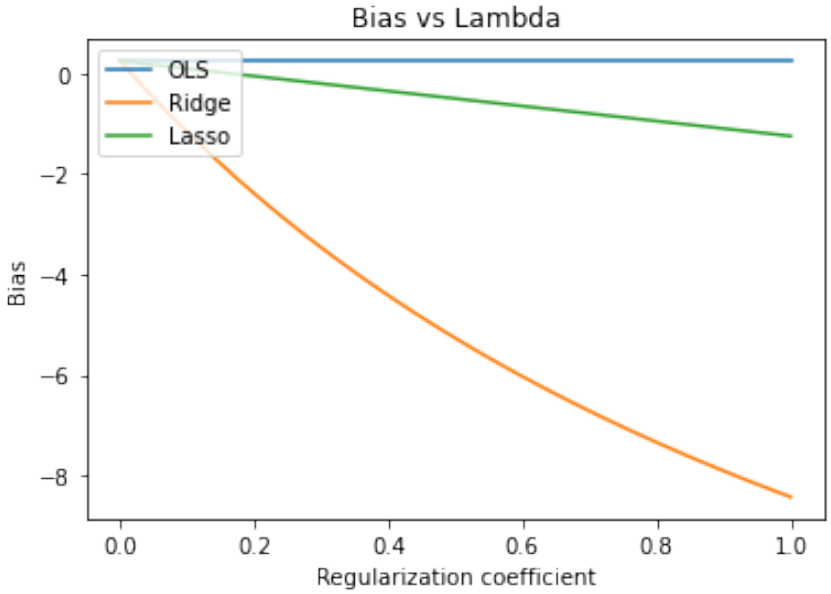
$$
\begin{aligned}
Variance(f(\hat{x})) &= \mathbb{E}\Big[\Big(f(\hat{x}) - \mathbb{E}[f(\hat{x})]\Big)^2\Big] \\
Bias(f(\hat{x})) &= \Big(\mathbb{E}[f(\hat{x})] - f(x)\Big) \\
\mathbb{E}[y - f(x)] &= \mathbb{E}[\epsilon^2] = \sigma^2
\end{aligned}
$$

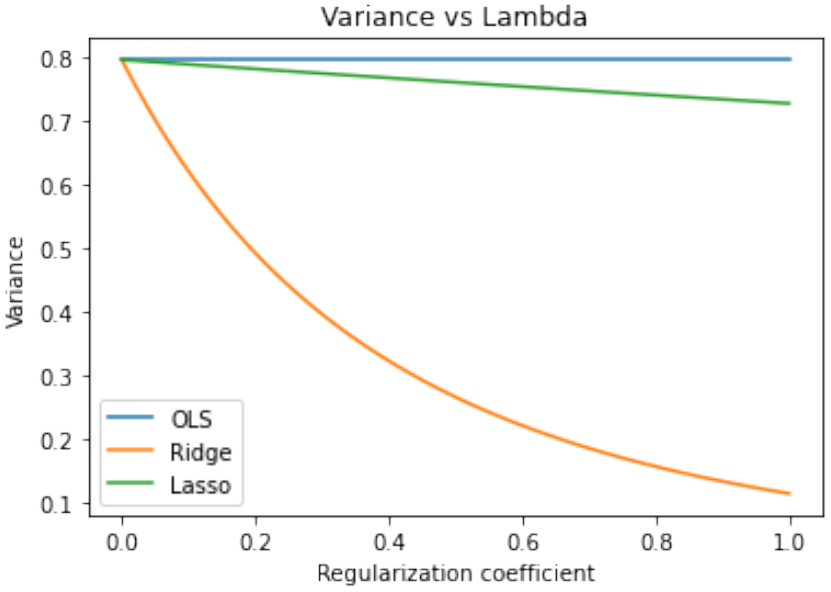
3.2 CS 337: Lab

(1)

Bias and Variance calculated using the formula mentioned above.

(2)





Observation w.r.t bias-variance trade-off for OLS, Ridge and Lasso are as follows -

(i) As expected the OLS bias variance does not change with λ as it is independent of λ

(ii) The bias of both Ridge and lasso increases with λ as discussed and shown above in part **1-(a)**. Ass coefficients become smaller than they ought to be for optimal prediction, bias increases due to underfitting of model on data.

(iii) The variance of both Ridge and lasso decreases with λ as discussed and shown above in part **1-(a)**. Since, more importance is gievn to the regularisation term, model's susceptibility to noise decreases so less variance.

(iv) Ridge regression has a more profound effect on bias and variance with variation in lambda compared to lasso. This is evident from the fact that we take squared term $\|w\|^2$ in ridge

regression whereas we only take magnitude $\|w\|$ in lasso.

(v) The main observation in both Lasso and Ridge regression is that as magnitude of bias increases, magnitude of variance decreases. This depicts the bias variance trade-off!

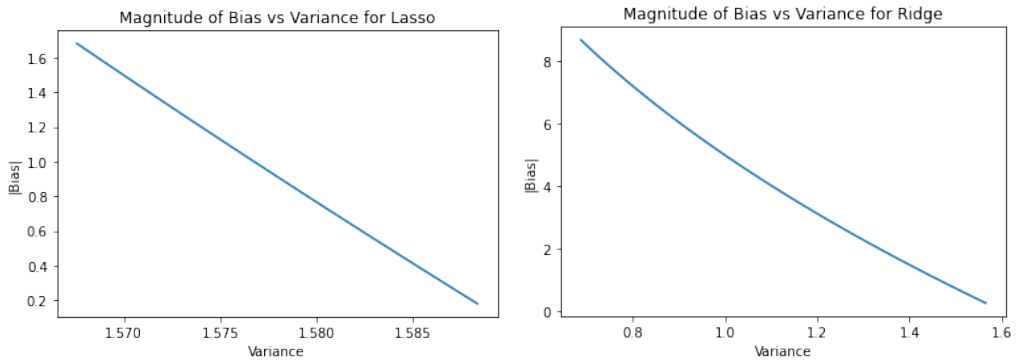


Figure 12: Visualisation of Bias - Variance trade-off

(3)

The `ols`, `lasso` and `ridge` regression functions completed in notebook. `ISTA` implemented for lasso.