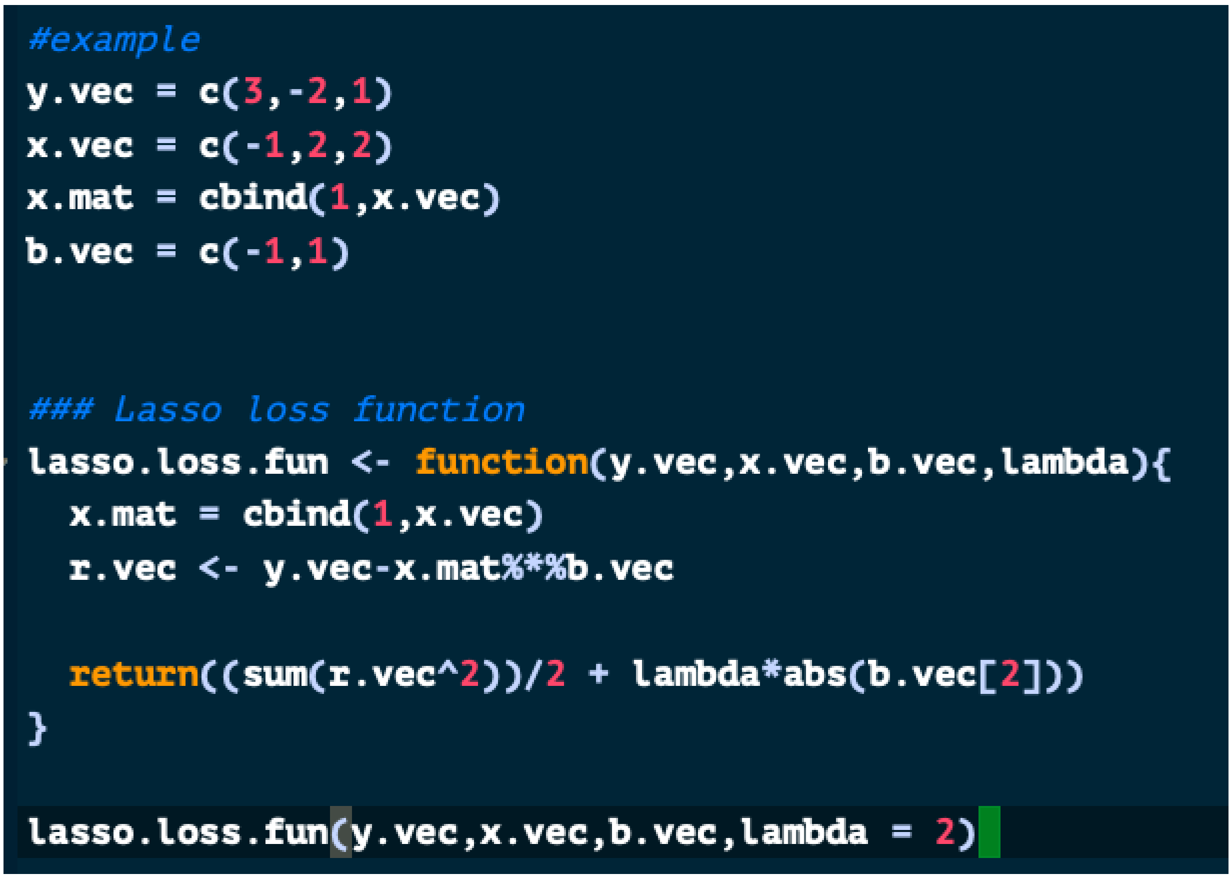
201111774 응용통계학과 박성진

(Homework)

1. Write an R code as above(LASSO)



```r
#example
y.vec = c(3,-2,1)
x.vec = c(-1,2,2)
x.mat = cbind(1,x.vec)
b.vec = c(-1,1)


### Lasso loss function
lasso.loss.fun <- function(y.vec,x.vec,b.vec,lambda){
  x.mat = cbind(1,x.vec)
  r.vec <- y.vec-x.mat%*%b.vec

  return((sum(r.vec^2))/2 + lambda*abs(b.vec[2]))
}

lasso.loss.fun(y.vec,x.vec,b.vec,lambda = 2)
```

2. Find and report some prorperties and algorithms for each loss(square, absolute, Huber, LASSO)

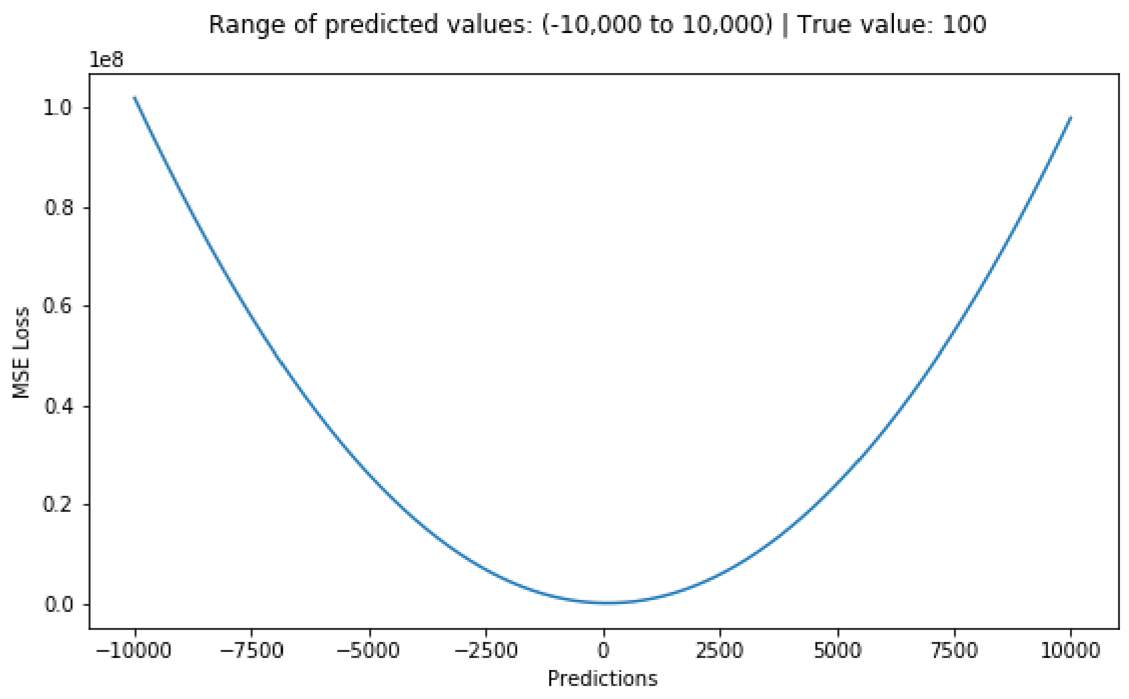
Various loss functions

- Popular choices for $L(B_0, B_1)$ for the simple linear regression model

---

square loss (least squares estimation = L2 loss)

$$L(B_0, B_1) = (y - B_0, B_1 x)^2$$

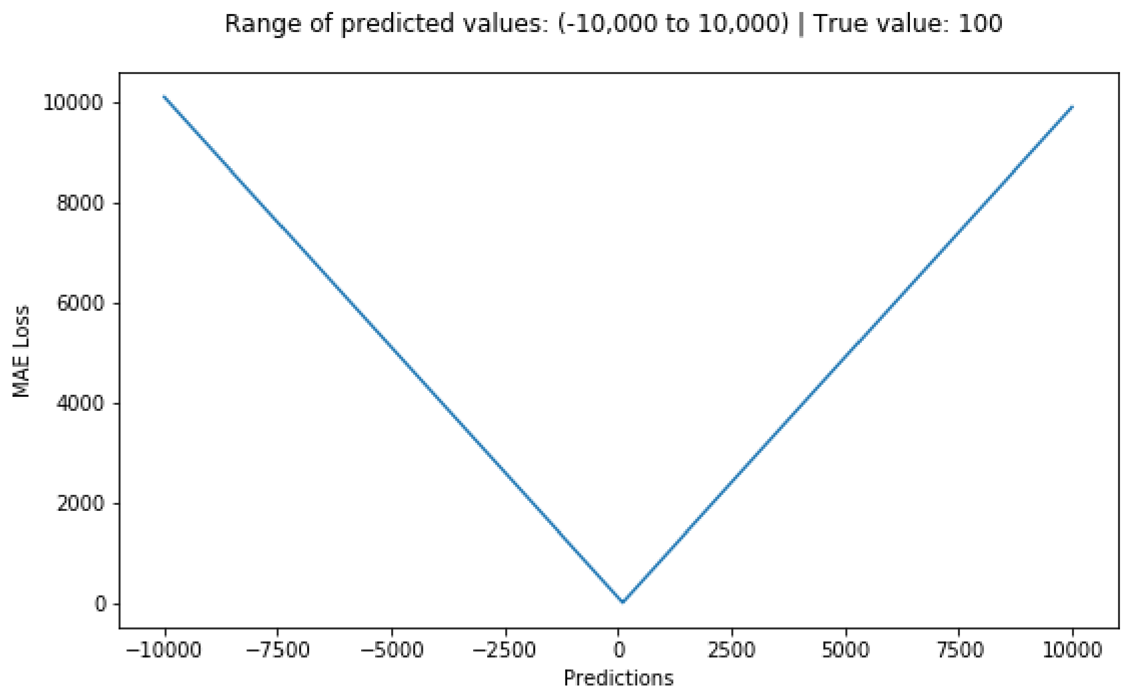


Plot of MSE Loss (Y-axis) vs. Predictions (X-axis)

- Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.
- It is always non-negative, and values closer to zero are better.
- The mathematical benefits of mean squared error are particularly evident in its use at analyzing the performance of linear regression, as it allows one to partition the variation in a dataset into variation explained by the model and variation explained by randomness.

absolute loss (Least absolute deviation = L1 loss)

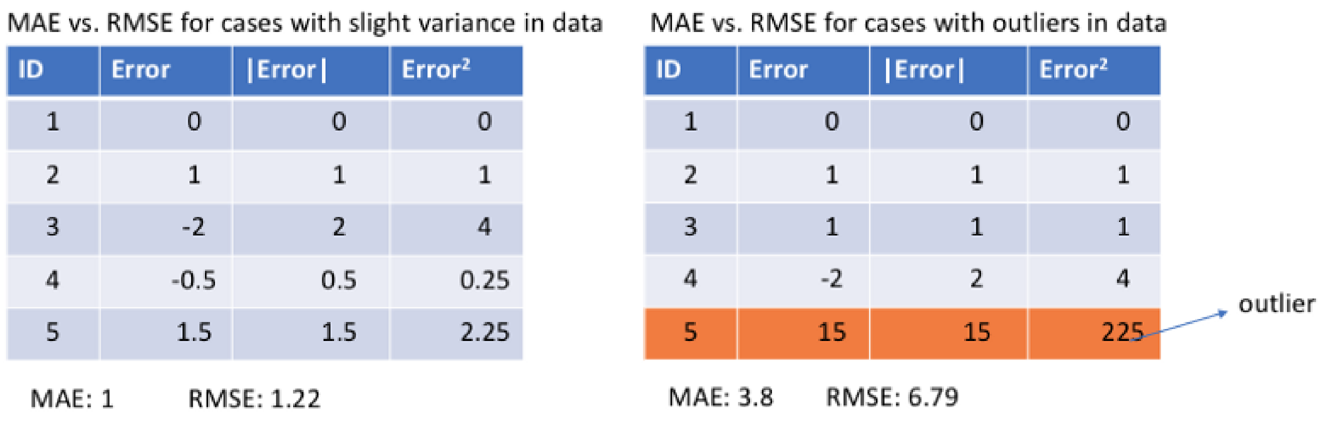
$$L(B_0, B_1) = |\, y - B_0 - B_1 x \,|$$



Plot of MAE Loss (Y-axis) vs. Predictions (X-axis)

- The method of least absolute deviations finds applications in many areas, due to its robustness compared to the least squares method.
- Least absolute deviations is robust in that it is resistant to outliers in the data.
- LAD gives equal emphasis to all observations, in contrast to ordinary least squares (OLS) which, by squaring the residuals, gives more weight to large residuals, that is, outliers in which predicted values are far from actual observations.
- This may be helpful in studies where outliers do not need to be given greater weight than other observations.
- If it is important to give greater weight to outliers, the method of least squares is a better choice.

L1 vs L2



MAE vs. RMSE for cases with slight variance in data

ID	Error	|Error|	Error^2
1	0	0	0
2	1	1	1
3	-2	2	4
4	-0.5	0.5	0.25
5	1.5	1.5	2.25

MAE: 1 RMSE: 1.22

MAE vs. RMSE for cases with outliers in data

ID	Error	|Error|	Error^2
1	0	0	0
2	1	1	1
3	1	1	1
4	-2	2	4
5	15	15	225

MAE: 3.8 RMSE: 6.79

Left: Errors are close to each other Right: One error is way off as compared to others

- L1 loss function is more robust and is generally not affected by outliers. On the contrary L2 loss function will try to adjust the model according to these outlier values, even on the expense of other samples. Hence, L2 loss function is highly sensitive to outliers in the dataset.
- With outliers in the dataset, a L2(Loss function) tries to adjust the model according to these outliers on the expense of other good-samples, since the squared-error is going to be huge for these outliers(for error > 1). On the other hand L1(Least absolute deviation) is quite resistant to outliers.

  As a result, L2 loss function may result in huge deviations in some of the samples which results in reduced accuracy.
- So, if you can ignore the ouliers in your dataset or you need them to be there, then you should be using a L1 loss function, on the other hand if you don’t want undesired outliers in the dataset and would like to use a stable solution then first of all you should try to remove the outliers and then use a L2 loss function. Or performance of a model with a L2 loss function may deteriorate badly due to the presence of outliers in the dataset.

summary

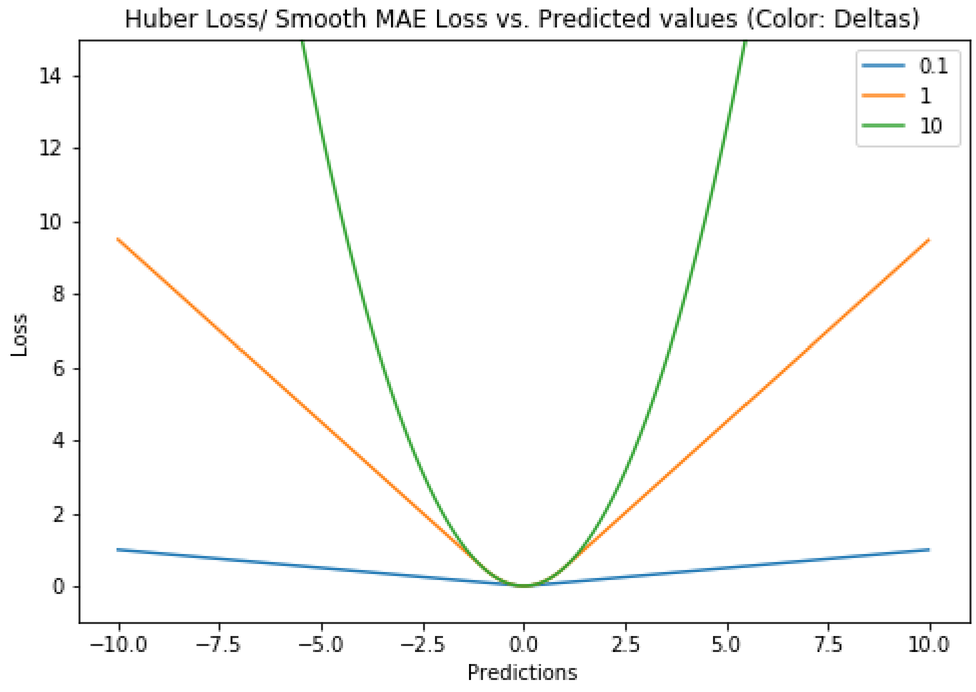
L1 loss is more robust to outliers, but its derivatives are not continuous, making it inefficient to find the solution. L2 loss is sensitive to outliers, but gives a more stable and closed form solution (by setting its derivative to 0.)

Huber loss (Huberiezed estimation)

$$L(B_0, B_1) = \begin{cases} t^2, & \text{if } |t| \le \delta \\ 2\delta|t| - \delta^2, & \text{otherwise} \end{cases}$$

$* \, t = y - B_0 - B_1 x$

$* \, \delta > 0$: Huber's tuning parameter (Huber's recommendation $\delta = 1.345$)



- **Problems with both(L1, L2)**: There can be cases where neither loss function gives desirable predictions. For example, if 90% of observations in our data have true target value of 150 and the remaining 10% have target value between 0–30.

  Then a model with MAE as loss might predict 150 for all observations, ignoring 10% of outlier cases, as it will try to go towards median value.

  In the same case, a model using MSE would give many predictions in the range of 0 to 30 as it will get skewed towards outliers. Both results are undesirable in many business cases.

- As defined above, the Huber loss function is strongly convex in a uniform neighborhood of its minimum t=0; at the boundary of this uniform neighborhood, the Huber loss function has a differentiable extension to an affine function at points t= δ and t= $-\delta$

- These properties allow it to combine much of the sensitivity of the mean-unbiased, minimum-variance estimator of the mean and the robustness of the median-unbiased estimator

Summary:

Huber loss is less sensitive to outliers in data than the squared error loss. It's also differentiable at 0. It's basically absolute error, which becomes quadratic when error is small.

How small that error has to be to make it quadratic depends on a hyperparameter, δ (delta), which can be tuned. Huber loss approaches MAE when δ ~ 0 and MSE when δ ~ ∞ (large numbers.)

---

references

- https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0 (https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0)
- https://nbviewer.jupyter.org/github/groverpr/Machine-Learning/blob/master/notebooks/05_Loss_Functions.ipynb (https://nbviewer.jupyter.org/github/groverpr/Machine-Learning/blob/master/notebooks/05_Loss_Functions.ipynb)

3. Normal equation(LSE) proof

i번째 sample (x_i, y_i)에 관한 회귀모형을 다음으로 표현할 때

- $$y_i = B_0 + B_1 x_i + \epsilon$$

Loss 함수는 $L(B_0, B_1) = (y - B_0 - B_1 x)^2$과 같아서

Risk 함수는 $R_n(B_0, B_1) = \sum_{i=1}^{n}(y_i - (B_0 + B_1 x_i))^2/2n$이다.

이를 최소로 하는 B_0과 B_1의 값을 이들의 추정값 $\hat{B}_0, \hat{B}_1$ 로 하는 방법이 **LSE** 이다.

$R_n(B_0, B_1)$을 최소화 시키는 B_0, B_1을 구하기 위하여 $R_n(B_0, B_1)$을 B_1과 B_0로 각각 편미분하여 다음의 결과를 얻는다

- $$\nabla R_n(B_0, B_1) = \begin{pmatrix} -\sum_{i=1}^{n}(y_i - (B_0 + B_1 x_i)/n \\ -\sum_{i=1}^{n} x_i(y_i - (B_0 + B_1 x_i)/n \end{pmatrix}$$

위의 편미분 값을 0으로 만드는 B_0, B_1을 $\hat{B}_0, \hat{B}_1$으로 대치하고 정리하면

- $$\hat{B}_0 n + \hat{B}_1 \sum x_i = \sum y_i$$
- $$\hat{B}_0 \sum x_i + \hat{B}_1 \sum x_i^2 = \sum x_i y_i$$

이 되는데 이 식을 **Normal equation**이라 부르고, 이를 $\hat{B}_0$과 $\hat{B}_1$에 대해 풀면,

- $\hat{B}_0 = \bar{y} - \hat{B}_1 \bar{x}$
- $\hat{B}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$ 이고,

표현을 간단히 하기 위하여

- $S_{xx} = \sum(x_i - \bar{x})^2$
- $S_{yy} = \sum(y_i - \bar{y})^2$
- $S_{xy} = \sum(x_i - \bar{x})(y_i - \bar{y})$

을 사용하기로 하면

- $\hat{B}_0 = \bar{y} - \hat{B}_1 \bar{x}$
- $\hat{B}_1 = \frac{S_{xy}}{S_{xx}}$

로 표현할 수 있다.

Implications for the Estimators

- $\hat{B}_1$ is normally distributed with mean B_1 and variance $\frac{\sigma^2}{S_{xx}}$

- $\hat{B}_0$ is normally distributed with mean B_0 and variance $\sigma^2 \frac{\sum_{i=1}^{n} x_i^2}{nS_{xx}}$

4. ANOVA TABLE

Source	**SS**	**DF**	**MS**	**F**	**P**
Factor	**SSR**	**1**	**SSR**	**MSR/MSE**	**F(1, n-2; a)**
Error	**SSE**	**n-2**	**MSE=** $\frac{SSE}{n-2}$		
Total	SST	n-1			

5. AIC(Akaike's Information Criterion)

- AIC는 여러 통계 모델들의 성능을 서로 비교할 수 있게 해줌.

- 데이터에 대한 여러 모델 세트가 주어지면 더 나은 모델은 최소 AIC 값을 갖는 모델.
- 따라서 AIC는 goodness of fit(가능도에 의해 구해진)을 보장하지만, 추정된 회귀계수의 수에 따라 증가하는 페널티도 포함.
- 모델에서 설명변수의 수를 늘리면 거의 goodness of fit이 향상하므로, 패널티는 overfitting을 방지함.
- AIC 값은 두 모델의 관측치 개수가 거의 동일할 때만 비교해야함.

proof (11)

$$\hat{\sigma}^2 = \hat{var}(\epsilon) = \frac{\sum_{i=1}^{n}(y_i - \hat{B}_0 - \hat{B}_1 x_i)^2}{n-2} \qquad (1)$$

$$\sigma^2 \approx \frac{(n-2)\hat{\sigma}^2}{n} = \frac{\sum_{i=1}^{n}(y_i - \hat{B}_0 - \hat{B}_1 x_i)^2}{n} \qquad (2)$$

$$AIC = AIC(k) = -2\log l(\hat{B}_0, \hat{B}_1) + \frac{2k}{n} \qquad (3)$$

(2)를 (3)에 적용하면

$$\log l(B_0, B_1) = -\frac{\log(2\pi)}{2} - \frac{\log \sigma^2}{2} - \frac{\sum_{i=1}^{n}(y_i - B_0 - B_1 x_i)^2}{2n\sigma^2} \qquad (4)$$

(4) 에 (2)을 대입하여

$$\log l(\hat{B}_0, \hat{B}_1) = -\frac{\log(2\pi)}{2} - \frac{\log(\frac{\sum_{i=1}^{n}(y_i - \hat{B}_0 - \hat{B}_1 x_i)}{n})^2}{2} - \frac{1}{2} \qquad (a)$$

변형된 AIC 는 (a)를 대입하여

$$\begin{aligned} AIC &= \log(2\pi) + \log(\frac{\sum_{i=1}^{n}(y_i - \hat{B}_0 - \hat{B}_1 x_i)^2}{n}) + 1 + \frac{2k}{n} \\ &\propto \log(\frac{\sum_{i=1}^{n}(y_i - \hat{B}_0 - \hat{B}_1 x_i)^2}{n}) + \frac{2k}{n} \end{aligned}$$