CSC2515 - HW2
Kopal Garg, 1003063221
Team members: Rohan Ajwani, Yujie Chen

1. Bias and Variance Decomposition for the ℓ_2-regularized Mean Estimator – 35pts. This exercise helps you become more comfortable with the bias and variance calculations and decomposition. It focuses on the simplified setting of the mean estimator.

Consider a r.v. Y with an unknown distribution p. This random variable has an (unknown) mean $\mu = \mathbb{E}[Y]$ and variance $\sigma^2 = \mathrm{Var}[Y] = \mathbb{E}\big[(Y-\mu)^2\big]$. Consider a dataset $\mathcal{D} = \{Y_1, \ldots, Y_n\}$ with independently sampled $Y_i \sim p$.

(a) **[5pt]** Show that the sample average estimator $h_{\text{avg}} = \frac{1}{n}\sum_{i=1}^n Y_i$ is the solution of the following optimization problem:

$$h_{\text{avg}}(\mathcal{D}) \leftarrow \underset{m\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^n |Y_i - m|^2.$$

a). sample avg. estimator, $h_{avg} = \frac{1}{n}\sum_{i=1}^{n} Y_i$

$$h_{avg}(D) \leftarrow \underset{m\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |Y_i - m|^2$$

$$= \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=1}^{n}|Y_i - m|^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\sum_{i=1}^{n}|Y_i - m|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial m}|Y_i - m|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial m}(Y_i - m)^2$$

Applying the chain rule:

$$= \frac{1}{n}\sum_{i=1}^{n}\left[2(Y_i - m)\frac{\partial}{\partial m}(Y_i - m)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[(2(Y_i - m)(-1)\right]$$

$$= \frac{-2}{n}\sum_{i=1}^{n}[Y_i - m]$$

$$= \frac{1}{n}\sum_{i=1}^{n} Y_i - \frac{1}{n}\sum_{i=1}^{n} m$$

$$= \frac{1}{n}\sum_{i=1}^{n} Y_i - \frac{m \cdot n}{n}$$

$$= \frac{1}{n}\sum_{i=1}^{n} Y_i - m$$

$$m = h_{avg} = \boxed{\frac{1}{n}\sum_{i=1}^{n} Y_i}$$

$\therefore$ havg minimizes the optimization problem

(b) **[5pt]** Compute the bias and variance of $h_{\text{avg}}(\mathcal{D})$.

$$h_{avg}(D) = \underset{m \in \mathbb{R}}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} |y_i - m|^2 = \frac{1}{n} \sum_{i=1}^{n} y_i$$

$$\text{bias} = |E_D[h(D)] - \mu|^2$$

$$= \left|E_D\left[\frac{1}{n} \sum_{i=1}^{n} y_i\right] - \mu\right|^2$$

$$= \left|\frac{1}{n} E_D\left[\sum_{i=1}^{n} y_i\right] - \mu\right|^2$$

$$= \left|\frac{1}{n} \sum_{i=1}^{n} E_D[y_i] - \mu\right|^2$$

$$= \left|\frac{1}{n} \sum_{i=1}^{n} \mu - \mu\right|^2$$

$$= ||\mu - \mu|$$

$$= \boxed{0}$$

$$\text{Variance} = E\left[|h(D) - E[h(D)]|^2\right]$$

$$= E_D\left[\left|\frac{1}{n} \sum_{i=1}^{n} y_i - E\left[\frac{1}{n} \sum_{i=1}^{n} y_i\right]\right|^2\right]$$

$$= E_D\left[\left|\frac{1}{n} \sum_{i=1}^{n} (y_i - \mu)\right|^2\right]$$

$$= \frac{1}{n^2} \sum_{i=1}^{n} E_D[(y_i - \mu)^2]$$

$$= \frac{1}{n^2} n \sigma^2 = \boxed{\frac{\sigma^2}{n}}$$

(c) **[5pt]** Consider the ℓ_2-regularized mean estimator $h_\lambda(\mathcal{D})$ defined for any $\lambda \geq 0$.

$$h_\lambda(\mathcal{D}) \leftarrow \underset{m\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |Y_i - m|^2 + \lambda |m|^2.$$

Notice that this is similar to the ridge regression, but only for a single random variable. Provide an explicit formula for this estimator in terms of the sample average and λ.

$$h_\lambda(D) \leftarrow \underset{m\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |Y_i - m|^2 + \lambda|m|^2 \quad , \lambda \geq 0$$

$$\frac{1}{n}\frac{\partial}{\partial m}\sum_{i=1}^{n}|Y_i - m|^2 + \frac{\partial}{\partial m}\lambda|m|^2 = 0$$

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial m}(Y_i^2 + m^2 - 2Y_i m) + \frac{\partial}{\partial m}\lambda|m|^2 = 0$$

$$\frac{1}{n}\sum_{i=1}^{n}(0 + 2m - 2Y_i) + 2\lambda m = 0$$

$$\frac{\cancel{2}}{n}\sum_{i=1}^{n}(m) - \frac{\cancel{2}}{n}\sum_{i=1}^{n}(Y_i) + \cancel{2}\lambda m = 0$$

$$m - \frac{1}{n}\sum_{i=1}^{n}Y_i + \lambda m = 0$$

$$m = h_\lambda(D) = \boxed{\left(\frac{1}{n}\sum_{i=1}^{n}Y_i\right)\frac{1}{1+\lambda}}$$

This minimizes the optimization problem

(d) **[10pt]** Compute the bias and variance of this regularized estimator.

Based on the derivation of the estimator $h_\lambda(D)$ from the previous part: $\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right)\frac{1}{1+\lambda}$

Bias: $|E_D[h_\lambda(D)] - \mu|^2$

$= \left|E\left[\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right)\frac{1}{1+\lambda}\right] - \mu\right|^2$

$= \left|E\left[\frac{1}{n}\sum_{i=1}^{n} y_i \left(1-\frac{\lambda}{1+\lambda}\right)\right] - \mu\right|^2$

$= \left|E\left[\frac{1}{n}\sum_{i=1}^{n} y_i\right] - E\left[\frac{1}{n}\sum_{i=1}^{n}(y_i)\left(\frac{\lambda}{1+\lambda}\right)\right] - \mu\right|^2$

$= \left|E\left[\frac{1}{n}\sum_{i=1}^{n} y_i\right] - \mu - E\left[\frac{1}{n}\sum_{i=1}^{n}(y_i)\left(\frac{\lambda}{1+\lambda}\right)\right]\right|^2$

$= \left|-E\left[\frac{1}{n}\sum_{i=1}^{n} y_i\left(\frac{\lambda}{1+\lambda}\right)\right]\right|^2$

$= \left|-\mu\left(\frac{\lambda}{1+\lambda}\right)\right|^2 = \boxed{\left(\frac{\mu\lambda}{1+\lambda}\right)^2}$

Variance: $E_D[|h_\lambda(D) - E[h_\lambda(D)]|^2]$

$= E_D\left[\left|\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right)\left(\frac{1}{1+\lambda}\right) - E\left[\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right)\left(\frac{1}{1+\lambda}\right)\right]\right|^2\right]$

$= \left(\frac{1}{1+\lambda}\right)^2 \sum_{i=1}^{n} E_D[|h_{avg}(D) - E[h_{avg}(D)]|^2]$

$= \boxed{\left(\frac{1}{1+\lambda}\right)^2 \cdot \frac{\sigma^2}{n}}$

Therefore, compared to linear regression without regularization, penalized regression trades increase in bias for decrease in variance.

(e) **[5pt]** Visualize $\mathbb{E}_{\mathcal{D}}\left[|h_\lambda(\mathcal{D}) - \mu|^2\right]$, the bias, and the variance terms for a range of λ. As a starting point, you can choose $\mu = 1$, $\sigma^2 = 9$, and $n = 10$.

Based on Piazza @83, I fixed μ=1, σ2=9, and n=10, and visualized how the E_D[|$h_λ$(D)−μ|^2], bias and variance terms vary as functions of λ (where λ>=0).

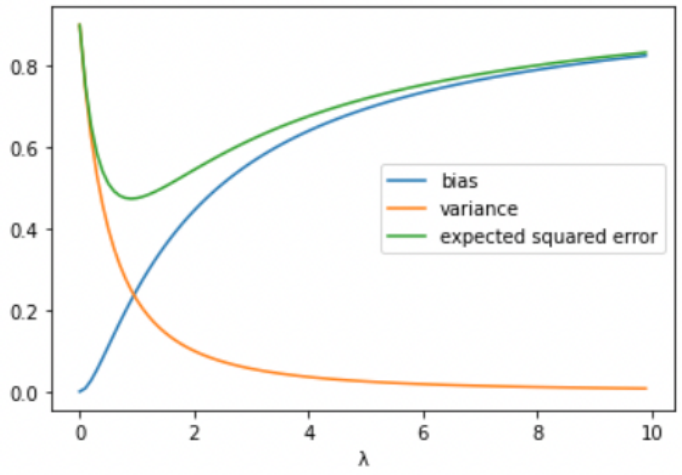
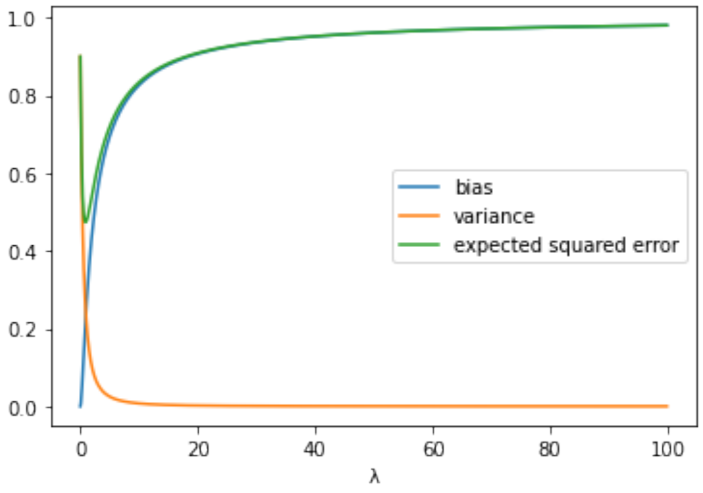


Figure 1. I E_D[|$h_λ$(D)−μ|^2], bias and variance as functions of λ, range (1-10).



II E_D[|$h_λ$(D)−μ|^2], bias and variance as functions of λ, range (1-100).

```
min error = 0.4736842105263158, λ = 0.9
```

The minimum error 0.47 occurs at λ value of 0.9.

(f) **[5pt]** Explain the visualization from the previous part, in terms of the effect of bias and variance on the expected squared error.

In previous parts, I decomposed the error term of a classifier into 2 interpretable terms – variance and bias. Variance captures how much a classifier changes if you we train on a different training set. Bias is inherent to our model; it checks if a classifier is biased towards a particular kind of solution even with an infinite amount of training data.

In general, more complex models tend to overfit to training samples, while simplest models may be underfit. The model starts off (λ=0) with a high variance and low bias and hence a high expected error. The cause of our model's poor performance is that the training error is most likely lower than the testing error (i.e., overfitting to noise or unrepresentative training examples). In comparison, with high λ values, the model has a high bias and a low variance. In this case, it may underfit the training data, perhaps because the model underfits the target and overlooks the regularities of the data.

Expected squared error is the sum of the bias and variance error. As lambda increases, the expected squared error increases. The best value of λ occurs around 0.9 and the minimum expected squared error with the given initial conditions is 0.47.

Q2.

a. **Data loading function is found in HW2q2.py.**
b. **Data summarization**

```
Number of data points: (506,)

Dimensions of the dataset (506, 13)

Features: ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX'
'PTRATIO' 'B' 'LSTAT']

Target min: 5.0

Target max: 50.0

Target mean: 22.532806324110677 standard deviation: 9.188011545278203
```

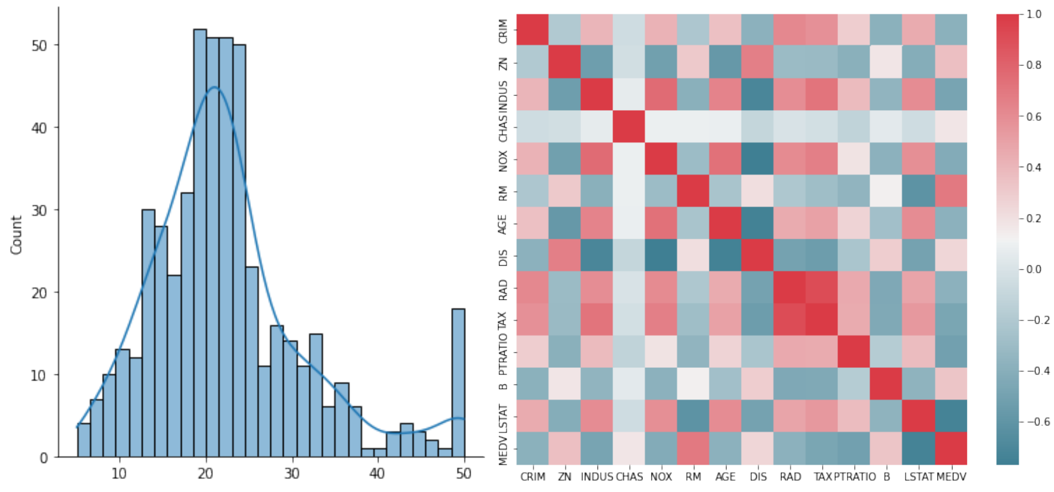


Figure 2. I. Target value distribution II. Pairwise Pearson's correlation coefficients between features and target.

c. Feature visualization

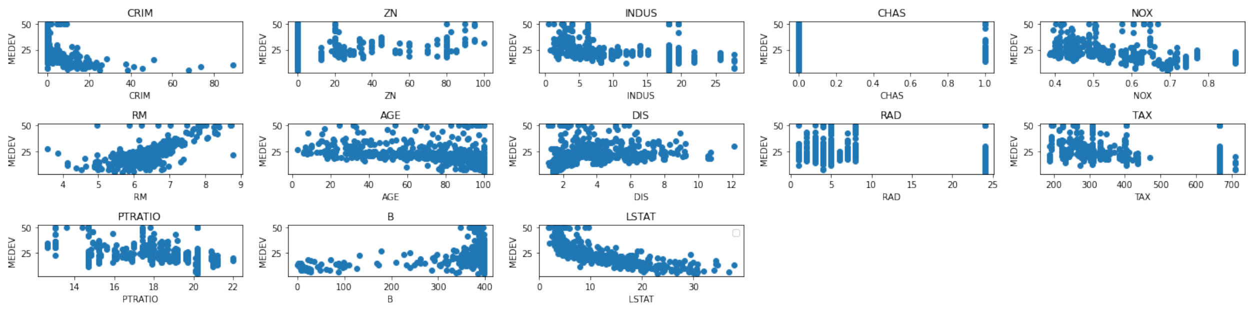


Figure 3. Scatter plot of features against target.

Next, X and y were split into training (70%) and testing arrays (30%).

d. Linear regression

I stacked a numpy array of one's to X to account for bias.

$f(x) = w_0 + w_1 x_1 + \cdots + w_d x_d$, where $x \in R^d$. This can be re-written as $f(x) = x^T w$, where $w \in R^{d+1}$.

By solving the linear system of equations (Lecture 3, Slide 21) from the direct solution for linear models, we get that the optimal weights are:

$$w^{LS} = (X^T X)^{-1} X^T y$$

numpy.linalg.solve was used to solve the above equation.

e. Features along with associated weights

Table 1. Features and associated weights.

CRIM	-0.115457
ZN	0.061744
INDUS	-0.005507
CHAS	2.270499
NOX	-17.957172
RM	3.338829
AGE	0.014306
DIS	-1.579986
RAD	0.365537
TAX	-0.015470
PTRATIO	-0.882199
B	0.007193
LSTAT	-0.562588

The third feature in the table, INDUS, refers to the proportion of non-retail business acres per town. It has an absolute value of approx. 0.006 and a negative sign, implying it is negatively related with median value of owner-occupied homes. Looking at the scatter plot for INDUS vs. MEDV in Figure 2., I don't see an apparent positive or negative relation between the two variables. Looking at the pair-wise Pearson's correlation in Figure 1., I see that INDUS and MEDV have a slightly negative correlation coefficient. I would expect for houses in neighborhoods with more businesses to have higher prices. Therefore, the sign doesn't match my expectations. However, the weight is small (and upon several iterations it fluctuated from very small negative values to 0 to very small positive values) so this feature seems to have a little effect on the MEDV.

f. Test fitted model and calculate MSE

30% of the entire dataset was set aside for testing purposes.

```
Test set performance:
   MSE: 22.337011563626145,
   RMSE: 4.726204773772095,
   MAE: 3.383743754540305
```

g. 2 more error measurement metrics

```
Test set performance:
   MSE: 22.337011563626145,
   RMSE: 4.726204773772095,
   MAE: 3.383743754540305
```

I chose RMSE and MAE.

$$RMSE = \sqrt{MSE}$$

MSE is biased for high values. RMSE is better when dealing with larger values and is also more interpretable. It represents the square root of the mean difference between predicted values and observed values. It can be larger or equal to 0. The smaller the RMSE the better the fit.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_{hat} - y|$$

MAE indicates how much the predicted values deviate from the target value, and can range from 0 to infinity. A smaller MAE implies a better model fit. The data seems to have a few outliers. MAE is less sensitive to outliers than RMSE.

h. Feature selection

Table 2. Top-5 Features sorted by absolute weight in descending order.

```
NOX       17.957172
RM         3.338829
CHAS       2.270499
DIS        1.579986
PTRATIO    0.882199
```

The most significant features are the ones with the highest absolute weight. The feature with a larger absolute value of weight can explain a larger proportion of model variance compared to feature with less a smaller weight. The most significant features include NOX, CHAS, RM, DIS and PTRATIO.

Q3.

(a) **[10pt]** Given dataset $\{(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(N)}, y^{(N)})\}$ and positive weights $a^{(1)}, ..., a^{(N)}$ show that the solution to the *weighted* least square problem

$$\mathbf{w}^* = \arg\min \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 \tag{3.1}$$

is given by the formula

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{A}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{A}\mathbf{y} \tag{3.2}$$

where $\mathbf{X}$ is the design matrix (defined in class) and $\mathbf{A}$ is a diagonal matrix where $\mathbf{A}_{ii} = a^{(i)}$

$$w^* = \text{argmin } \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - w^T x^{(i)})^2$$

$$= \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - w^T x^{(i)})^2$$

$$= \frac{1}{2} A(Y - W^T X)^2$$

$$= \frac{1}{2}(Y - XW)^T A(Y - XW) \Big] ①$$

$$= \frac{1}{2}(Y^T - X^T W^T)(AY - AXW)$$

$$= \frac{1}{2}\left[Y^T AY + X^T W^T AXW - \overset{②}{\overbrace{X^T W^T AY}} - Y^T AXW\right]$$

$$= \frac{1}{2}\left[Y^T AY + \underbrace{X^T W^T AXW}_{③} - 2X^T W^T AY\right]$$

$$\nabla(w^*) = 0 = \left[0 + 2X^T AXw^* - 2X^T AY\right]$$

$$X^T AXw^* = X^T AY$$

$$w^* = (X^T AX)^{-1} X^T AY$$

① If we rewrite the error term as $a^{(i)}b^2$, where $b = y^{(i)} - w^T x^{(i)}$ inner product $= b^T A b$

② $(X^T W^T AY)^T = Y^T AXW$
This is a 1X1 matrix, so, $X^T W^T AY = Y^T AXW$

③ Property:
$\frac{d}{dx} x^T a x = 2ax$

(b) **[10pt]** Locally weighted least squares regression algorithm combines ideas from k-NN and ordinary linear regression. For each new test example $\mathbf{x}$ it computes distance-based weights for each training example

$$a^{(i)} = \frac{\exp(-||\mathbf{x} - \mathbf{x}^{(i)}||^2/2\tau^2)}{\sum_j \exp(-||\mathbf{x} - \mathbf{x}^{(j)}||^2/2\tau^2)},$$

and then computes

$$\mathbf{w}^* = \arg\min \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2,$$

and predicts

$$\hat{y} = \mathbf{x}^T\mathbf{w}^*.$$

$$w^* = \text{argmin } \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - w^T x^{(i)})^2 + \frac{\lambda}{2} \|w\|^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - w^T x^{(i)})^2 + \frac{\lambda}{2} \|w\|^2$$

$$= \frac{1}{2} A (Y - W^T X)^2 + \frac{\lambda}{2} w^2$$

$$= \frac{1}{2} (Y - XW)^T A (Y - XW) + \frac{\lambda}{2} w^2 \Big]$$

① If we rewrite the error term as $a^{(i)} b^2$, where $b = y^{(i)} - w^T x^{(i)}$ inner product $= b^T A b$

$$= \frac{1}{2} (Y^T - X^T W^T)(AY - AXW) + \frac{\lambda}{2} w^2$$

$$= \frac{1}{2} \Big[Y^T A Y + X^T W^T A X W - \underbrace{X^T W^T A Y - Y^T A X W}_{②} \Big] + \frac{\lambda}{2} w^2$$

$$= \frac{1}{2} \Big[Y^T A Y + \underbrace{X^T W^T A X W}_{③} - 2 X^T W^T A Y \Big] + \frac{\lambda}{2} w^2$$

② $(X^T W^T A Y)^T = Y^T A X W$ This is a 1×1 matrix, so, $X^T W^T A Y = Y^T A X W$

$$\nabla(w^*) = 0 = \frac{1}{\cancel{2}} \Big[0 + \cancel{2} X^T A X w^* - \cancel{2} X^T A Y \Big] + \lambda I w^*$$

③ Property: $\frac{d}{dx} x^T a x = 2ax$

$$- X^T A X w^* - \lambda I w^* = - X^T A Y$$

$$w^* (X^T A X + \lambda I) = X^T A Y$$

$$w^* = \boxed{(X^T A X + \lambda I)^{-1} (X^T A Y)}$$

Please see the code for implementation of locally weighted least squares regression algorithm.

(c) **[5pt]** Use k-fold cross-validation to compute the average loss for different values of τ in the range [10,1000] when performing regression on the Boston Houses dataset. Plot these loss values for each choice of τ (For this question fix $\lambda = 10^{-5}$).

Please see attached code.

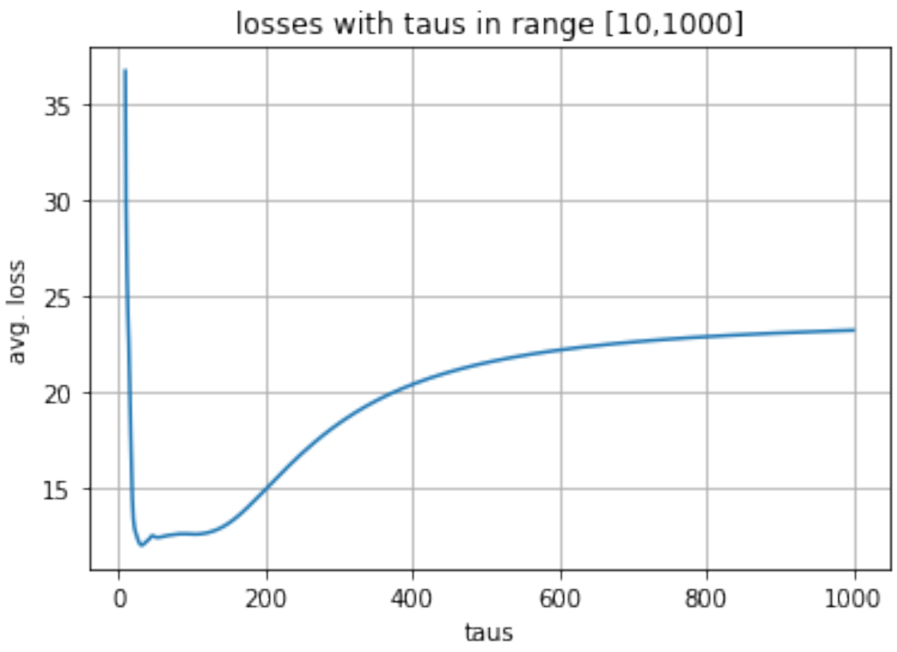


Figure 4. Average loss against τ ranging from 10-1000, using 5-fold cross validation

(d) **[5pt]** How does this algorithm behave when $\tau \to \infty$? When $\tau \to 0$?

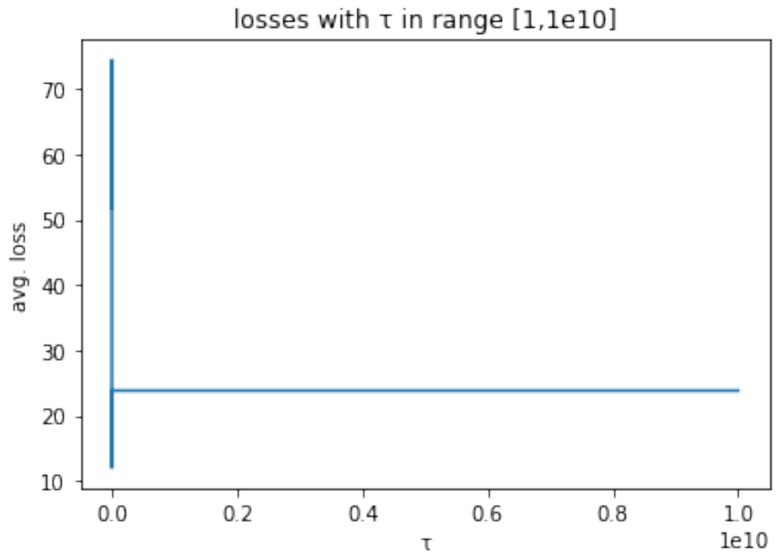


Figure 5. I. Average loss against τ ranging from 0 to 1e10, using 5-fold cross validation

From the plot it looks like as tau goes to infinity, the average loss stabilizes at a value close to 25. If we compute this, it stabilizes at 1/N (where N = # data points):

$$\frac{e^{-(\|x-x^{(i)}\|/2\tau^2)}}{\sum_j e^{-(\|x-x^{(j)}\|/2\tau^2)}}$$

$$= \frac{\frac{1}{e^{(\|x-x^{(i)}\|/2\tau^2)}}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\tau^2)}}}$$

As $\tau \to \infty$:

$$= \frac{1/e^0}{\sum_j 1/e^0} = \frac{1}{\sum_j 1} = \frac{1}{N}, \text{ where N is the \# training examples}$$

Numerically computing the value as tau goes to 0 is difficult as it leads to an undefined value.

As tau goes to 0, the average loss displays an asymptotic behavior. The influence of points closest to the test datum grow very large. This leads to an unstable behavior in the validation/testing sets, but for the training set, test datum is predicted based on the value of point closest to it (which in the training set, would be the point itself), so the error would be 0. But since the model is overfit, the validation error fluctuates to high values.

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

Tuning parameter, τ, controls how quickly the weight of a training example falls off with its distance to the query point x.

The smaller the value, the more local and wigglier our fit will be.

A very large value will lead to a global fit to the data using all the training observations.

If we think of the above simplified function as a bell-shaped curve, very large τ increases the span or width of the bell shape curve and makes further points have more weight.

(e) **[5pt]** Mention two advantages and two disadvantages of the locally weighted linear regression compared to ordinary linear regression.

Advantages of using locally weighted linear regression compared to linear regression:

- Locally weighted linear regression is a non-parametric algorithm. The model does not learn a fixed set of parameters as is done in ordinary linear regression. Rather parameters are computed individually for each query point. A higher weightage is assigned to the points in the training set lying closer than the points lying far away from the test point.
- If we are modeling a highly nonlinear relationships between the independent variables and the dependent variable, then a locally weighted linear regression would be a better choice than linear regression, as linear regression tends to under fit the training data of such kind.
- It doesn’t require us to specify a function to fit a model to all sample data points. Instead.

Disadvantages of using locally weighted linear regression compared to linear regression:

- We need the entire training set to make future predictions because every time we try to make a prediction, we are constructing a regression line that’s local to the data point of our interest. This process is also computationally intensive in comparison to linear regression as a regression model is computed for each data point
- It depends on local data structure when performing local fits, so we need densely sampled data sets.