CPSC 340 Assignment 3

1 Vectors, Matrices, and Quadratic Functions

1.1 Basic Operations

Using the definitions below,

$$\alpha = 5, \quad x = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad z = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & -2 \end{bmatrix},$$

evaluate the following expressions (show your work, but you may use answers from previous parts to simplify calculations):

1. $x^Tx = \begin{bmatrix} 2 & -3 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ -3 \end{bmatrix} = \begin{bmatrix} 13 \end{bmatrix}$
2. $\|x\|^2 = 2^2 + (-3)^2 = 13$
3. $x^T(x + \alpha y) = \begin{bmatrix} 2 & -3 \end{bmatrix} \cdot \begin{bmatrix} 2 + 5 \cdot 1 \\ -3 + 5 \cdot 4 \end{bmatrix} = \begin{bmatrix} -37 \end{bmatrix}$
4. $Ax = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & -2 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ -3 \end{bmatrix} = \begin{bmatrix} -4 \\ -5 \\ 12 \end{bmatrix}$
5. $z^TAx = \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -4 \\ -5 \\ 12 \end{bmatrix} = \begin{bmatrix} 4 \end{bmatrix}$
6. $A^TA = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & -2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & -2 \end{bmatrix} = \begin{bmatrix} 14 & 2 \\ 2 & 9 \end{bmatrix}$

If $\{\alpha, \beta\}$ are scalars, $\{x, y, z\}$ are real-valued column-vectors of length d, and $\{A, B, C\}$ are real-valued $d \times d$ matrices, state whether each of the below statements is true or false in general and give a short explanation.

7. $yy^Ty = \|y\|^2y$.
   False. $\|y\|^2$ is a 1x1 matrix, while y is a dx1 matrix. It is not possible to perform matrix multiplication on the RHS. Furthermore, $\|y\|^2 = y^Ty$ and not yy^T.
8. $x^TA^T(Ay + Az) = x^TA^TAy + z^TA^TAx$.
   True. After expanding the brackets, we will get the first term. To manipulate the second term to the RHS, we can use $x^TA^TAz = (Ax)^TAz = (Az)^TAx = z^TA^TAx$.

9. $x^T(B+C) = Bx + Cx$.
   False. After expanding the bracket, there is no way to manipulate x^TB to Bx, unless $B^T = B$. The same holds for C and C^T.

10. $(A+BC)^T = A^T + C^TB^T$.
    True. After expanding the brackets, we get $A^T + (BC)^T = A^T + C^TB^T$.

11. $(x-y)^T(x-y) = \|x\|^2 - x^Ty + \|y\|^2$.
    False. After expansion of the LHS, we obtain $\|x\|^2 - x^Ty - y^Tx + \|y\|^2$. There is an extra term $-y^Tx$ that is not in the RHS.

12. $(x-y)^T(x+y) = \|x\|^2 - \|y\|^2$.
    False. After expansion, we get 4 $1x1$ matrices, $\|x\|^2 - x^Ty - y^Tx + \|y\|^2$. With the exception of the edge case where both x^Ty and y^Tx are $\begin{bmatrix} 0 \end{bmatrix}$, it is not possible to remove both terms.

1.2 Converting to Matrix/Vector/Norm Notation

Using our standard supervised learning notation (X, y, w) express the following functions in terms of vectors, matrices, and norms (there should be no summations or maximums).

1. $\|Xw - y\|$
2. $\log \| \exp |Xw - y|\| + \frac{\lambda}{2}\|w\|^2$
3. $(Xw-y)^TZ(Xw-y) + \lambda\|w\|$

You can use Z to denote a diagonal matrix that has the values z_i along the diagonal.

1.3 Minimizing Quadratic Functions as Linear Systems

Rubric: {reasoning:3}

Write finding a minimizer w of the functions below as a system of linear equations (using vector/matrix notation and simplifying as much as possible). Note that all the functions below are convex so finding a w with $\nabla f(w) = 0$ is sufficient to minimize the functions (but show your work in getting to this point).

1. $f(w) = \frac{1}{2}\|w - v\|^2$.
   $f'(w) = w - v$. By setting $f'(w)$ to zero, we find that the minimizer is when $w = v$.

2. $f(w) = \frac{1}{2}\|w\|^2 + w^TX^Ty$.
   $f'(w) = 0.5w + X^Ty$. Setting $f'(w)$ to zero, we find that the minimizer is when $w = -2X^Ty$.

3. $f(w) = \frac{1}{2}\sum_{i=1}^n z_i(w^Tx_i - y_i)^2$.
   $f'(w) = X^TZ(Xw - y)$. Setting $f'(w)$ to zero, we find the minimizer by solving $X^TZ(Xw - y) = 0$.

Above we assume that v is a $d \times 1$ vector.

Hint: Once you convert to vector/matrix notation, you can use the results from class to quickly compute these quantities term-wise. As a sanity check for your derivation, make sure that your results have the right dimensions.

2 Robust Regression and Gradient Descent

2.1 Weighted Least Squares in One Dimension

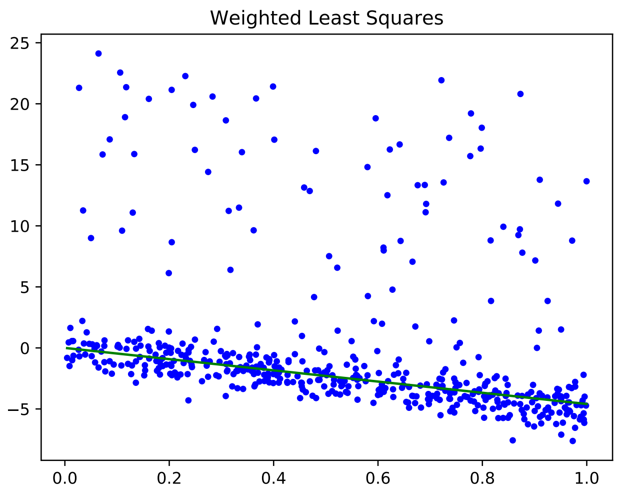


Figure 1: Updated Weighted Least Squares Plot

Completed in *linear_model.py*.

2.2 Smooth Approximation to the L1-Norm

$$f(w)=\sum_{i=1}^{n} \log\left(\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)\right).$$

$$\frac{\partial f}{\partial w_j} for j = 1, \ldots, d=\sum_{i=1}^{n} \frac{x_{ij}\exp(w^T x_i - y_i) - x_{ij}\exp(y_i - w^T x_i)}{\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)}$$

2.3 Robust Regression

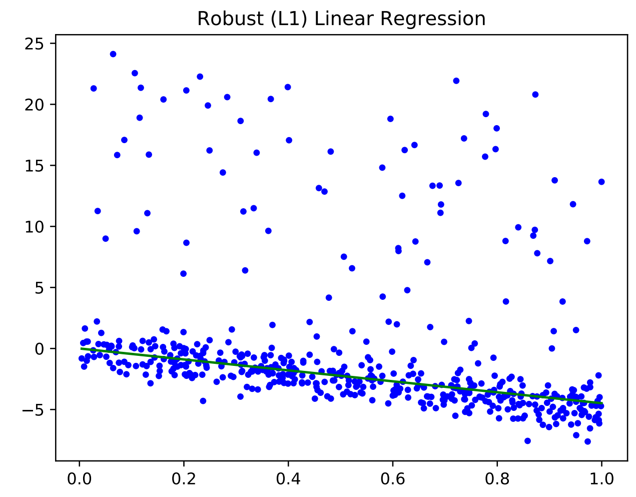


Figure 2: Updated Robust Regression Plot

Completed in *linear_model.py*.

3 Linear Regression and Nonlinear Bases

3.1 Adding a Bias Variable

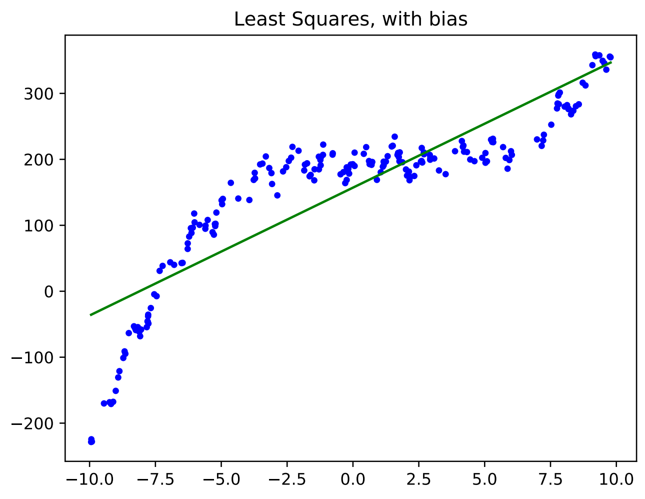


Figure 3: Updated least squares regression with bias

Code completed in *linear_model.py*
Updated training error: 3551.3
Updated test error: 3393.9

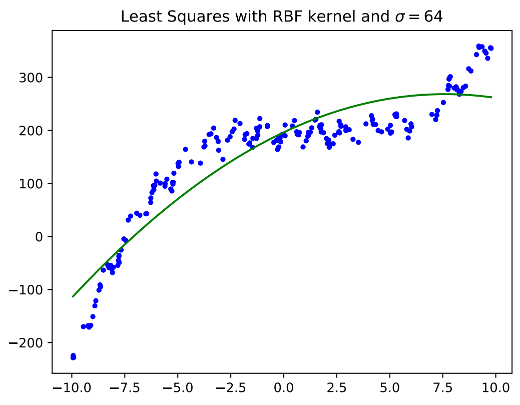
3.2 Polynomial Basis

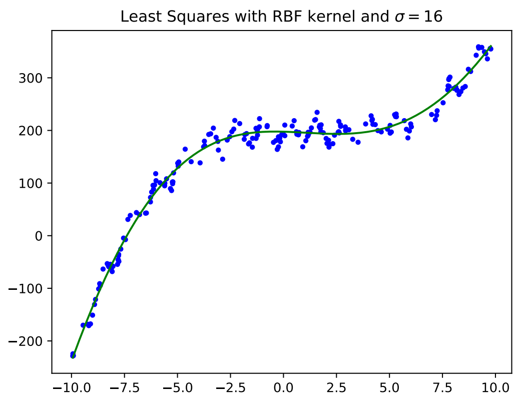
p value	Training error	Test error
0	15480.5	14390.8
1	3551.3	3393.9
2	2168.0	2480.7
3	252.0	242.8
4	251.5	242.1
5	251.1	239.5
6	248.6	246.0
7	247.0	242.9
8	241.3	246.0
9	235.8	259.3
10	235.1	256.3

Code completed in *linear_model.py*
We can see that as the p value increases, the training error decreases with each increment. As p increases, the model is able to learn non-linear relationships up to the p-th power and fit the training set better. However, as the p value increases, the test error initially decreases before increasing. Initially as p increases, the model is able to learn the non-linear relationships between X and y. However, as p gets overly large, it overfits the training set and performs poorly on the test set.

4 Non-Parametric Bases and Cross-Validation

4.1 Proper Training and Validation Sets





RBF without shuffling

RBF with shuffling

Comparison with and without shuffling of data

-	Training error	Test error
Before shuffling	2184.1	2495.9
After shuffling	253.1	237.0

After performing shuffling on the training and validation set, both the training and test error are reduced. Furthermore, from the graph plots, we can see that the model is able to better fit the data.

4.2 Cross-Validation

Now that we've dealt with the randomization, something's still a bit disturbing: if you run the script more than once it might choose different values of σ. This variability would be reduced if we had a larger "train" and "validation" set, and one way to simulate this is with *cross-validation*.

1. 1 is the most common, followed by 16.
2. Code completed in *main.py*. Value selected is mostly 16.

4.3 Cost of Non-Parametric Bases

Under the linear basis, the cost of training the model on n examples and d features is $O(nd^2 + d^3)$. This is because forming X^TX costs $O(nd^2)$ and solving the resulting d x d linear system costs $O(d^3)$. The cost of classifying t new examples is $O(dt)$, since each prediction is simply a sum of the multiplication of the d features with the corresponding weights.

Under the Gaussian RBF, X is replaced by Z, which is a nxn matrix. Forming Z^TZ costs $O(n^3)$ and solving the resulting nxn linear system also costs $O(n^3)$. Thus, solving the cost of training the model on n examples and d features is $O(n^3)$. The cost of classifying t new examples is $O(nt)$, since each prediction is a sum of the multiplication of the n features with the corresponding weights. RBFs are thus cheaper to train and test when $n < d$.

5 Very-Short Answer Questions

5.1 Essentials

1. Regression deals with continuous data which are not restricted to defined separate values, but can occupy any value over a continuous range. It would not be useful to judge the accuracy of a prediction to whether it is exactly that of the result, as this would result in the dismissal of results that are close to ideal. Instead, the "closeness" of a prediction to the result is more useful a gauge and is thus used.
2. A situation would be when one column of X is a linear combination of another / others. For example, if X = $\begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \end{bmatrix}$, where X[1] is twice of X[0].
3. With 1 feature and a polynomial basis of degree p, the matrix Z to replace X will be n x p. The complexity to computing a solution is $O(p)$, since all that is required is multiplying the p features to their weights.
4. A regression tree with linear regression at the leaves would be a better choice when the data has non-linear relationships which cannot be modeled by the least squares regression. It allows us to model each split separately and model non-linear relationships with linear models.
5. If the loss function is convex, then all stationary points are global optima, ie. any minimizers obtained will give the global minimum loss. If the loss function is non-convex, finding the optimal solution will be a NP-hard optimization problem.
6. The time complexity for the normal equation is $O(nd^2+d^3)$, which increases polynomially as dimensions d increases. Hence, gradient descent would perform better for larger d as its has a significantly better time complexity of $O(ndt)$ for t iterations.
7. Computing the partial derivatives of the loss function is non-trivial, because each partial derivative depends on the other partial values of w, and cannot be solved by simply setting the gradient to zero.
8. The normal equations are only applicable to linear least squares, but in robust regression we use a different loss function.
9. Too small a learning rate will result in slow / no progress where the loss function is not moving towards the minimum. In the case of a non-convex function, it could also lead to the optimizer getting stuck in a local minimum.
10. Too large a learning rate may result in a strong fluctuation to the error function without a steady decrease to the minimum. In some cases, gradient descent could lead to going further away from the intended minimum.

5.2 These ones are optional and not for marks

1. We take the gradient by hand to better understand the function that we are calculating, and use the approximate fprime to check our results. I would believe that calculating the gradient by hand would also have a smaller time complexity than using approximation.
2. It will be harder to achieve convergence as the gradients do not get smaller near a minimizer.