CPSC 340 Assignment 3

1 Vectors, Matrices, and Quadratic Functions

1.1 Basic Operations

1. $x^T x = 2^2 + 3^2 = 13$
2. $\|x\|^2 = x^T x = 13$
3. $x^T(x + \alpha y) = \begin{bmatrix} 2 & -3 \end{bmatrix} (\begin{bmatrix} 2 \\ -3 \end{bmatrix} + \begin{bmatrix} 5 \\ 20 \end{bmatrix}) = \begin{bmatrix} 2 & -3 \end{bmatrix} \begin{bmatrix} 7 \\ 17 \end{bmatrix} = 2(7) + -3(17) = -37$
4. $Ax = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & -2 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \end{bmatrix} = \begin{bmatrix} 1(2) + 2(-3) \\ 2(2) + 3(-3) \\ 3(2) + -2(-3) \end{bmatrix} = \begin{bmatrix} -4 \\ -5 \\ 12 \end{bmatrix}$
5. $z^T Ax = \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} -4 \\ -5 \\ 12 \end{bmatrix} = 2(-4) + 0(-5) + 1(12) = 4$
6. $A^T A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & -2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & -2 \end{bmatrix} = \begin{bmatrix} 1(1) + 2(2) + 3(3) & 1(2) + 2(3) + 3(-2) \\ 2(1) + 3(2) + -2(3) & 2(2) + 3(3) + -2(-2) \end{bmatrix} = \begin{bmatrix} 14 & 2 \\ 2 & 17 \end{bmatrix}$
7. True: $yy^T y = y(y^T y) = y\|y\|^2 = \|y\|^2 y$. Since $\|y\|^2$ is a scalar, order does not matter.
8. True: $x^T A^T(Ay + Az) = x^T A^T Ay + x^T A^T Az = x^T A^T Ay + (Ax)^T(Az) = x^T A^T Ay + (Az)^T(Ax) = x^T A^T Ay + z^T A^T Ax$
9. False: $x^T(B + C) = x^T B + x^T C$ since matrix-vector multiplication is not commutative.
10. True: $(A + BC)^T = A^T + (BC)^T = A^T + C^T B^T$
11. False: $(x - y)^T(x - y) = x^T x - x^T y - y^T x + y^T y = x^T x - x^T y - x^T y + y^T y = \|x\|^2 + \|y\|^2 - 2x^T y$
12. True: $(x - y)^T(x + y) = x^T x + x^T y - y^T x - y^T y = x^T x + x^T y - x^T y - y^T y = \|x\|^2 - \|y\|^2$

1.2 Converting to Matrix/Vector/Norm Notation

1. $\sum_{i=1}^n |w^T x_i - y_i| = \|Xw - y\|_1$
2. $\max_{i \in \{1,2,\ldots,n\}} |w^T x_i - y_i| + \frac{\lambda}{2} \sum_{j=1}^d w_j^2 = \|Xw - y\|_\infty + \frac{\lambda}{2}\|w\|^2$
3. $\sum_{i=1}^n z_i(w^T x_i - y_i)^2 + \lambda \sum_{j=1}^d |w_j| = Z\|Xw - y\|^2 + \lambda\|w\|_1$

1.3 Minimizing Quadratic Functions as Linear Systems

1. $f(w) = \frac{1}{2}\|w - v\|^2 = \frac{1}{2}\|w\|^2 + \frac{1}{2}\|v\|^2 - w^T v$
   $\nabla f(w) = w - v = 0$
   $\therefore$ When $w = v, \nabla f(w) = 0$

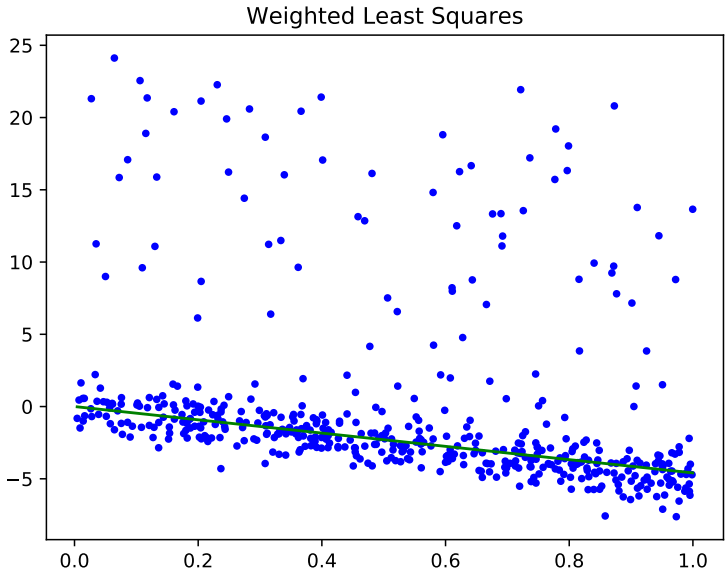
2. $f(w) = \frac{1}{2}\|w\|^2 + w^T X^T y$
   $\nabla f(w) = w + X^T y = 0$
   $\therefore$ When $w = -X^T y, \nabla f(w) = 0$

3. $f(w) = \frac{1}{2}\sum_{i=1}^{n} z_i (w^T x_i - y_i)^2 = Z\|Xw - y\|^2 + \lambda\|w\|_1 = \frac{1}{2} w^T X^T Z X w + \frac{1}{2} y^T Z y - w^T X^T Z y$
   $\nabla f(w) = X^T Z X w - X^T Z y = 0$
   $\therefore$ When $X^T Z X w = X^T Z y, \nabla f(w) = 0$

2 Robust Regression and Gradient Descent

2.1 Weighted Least Squares in One Dimension

Code: Linked in README, `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_c5u0b_a3/blob/master/code/linear_model.py`

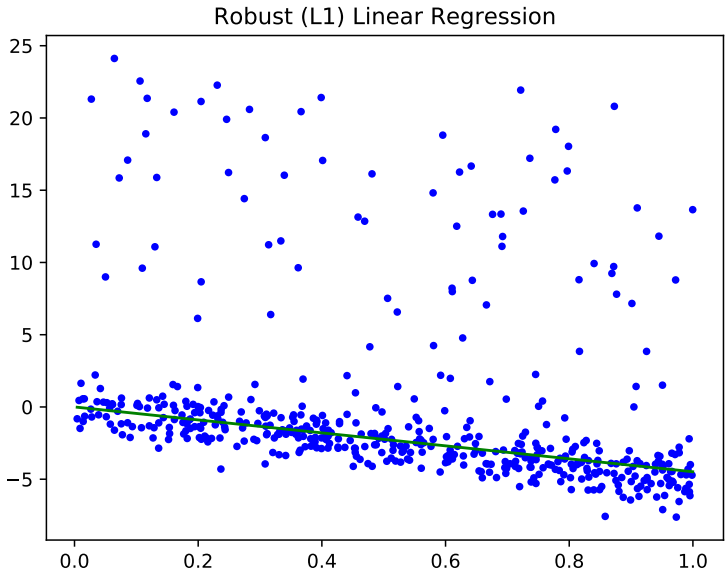


2.2 Smooth Approximation to the L1-Norm

$f(w) = \sum_{i=1}^{n} \log\left(\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)\right)$
$\nabla f(w) = \sum_{i=1}^{n} \frac{\partial}{\partial w} \log\left(\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)\right)$
$\nabla f(w) = \sum_{i=1}^{n} \frac{\frac{\partial}{\partial w}\left(\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)\right)}{\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)}$
$\nabla f(w) = \sum_{i=1}^{n} \frac{\exp(w^T x_i - y_i)\frac{\partial}{\partial w}(w^T x_i - y_i) \exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)(-1)\frac{\partial}{\partial w}(w^T x_i - y_i)}{\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)}$
$\nabla f(w) = \sum_{i=1}^{n} \frac{\exp(w^T x_i - y_i) x_{ij} - \exp(y_i - w^T x_i) x_{ij}}{\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)}$
$\nabla f(w) = \sum_{i=1}^{n} x_{ij} \frac{\exp(w^T x_i - y_i) - \exp(y_i - w^T x_i)}{\exp(w^T x_i - y_i) + \exp(y_i - w^T x_i)}$

2.3 Robust Regression

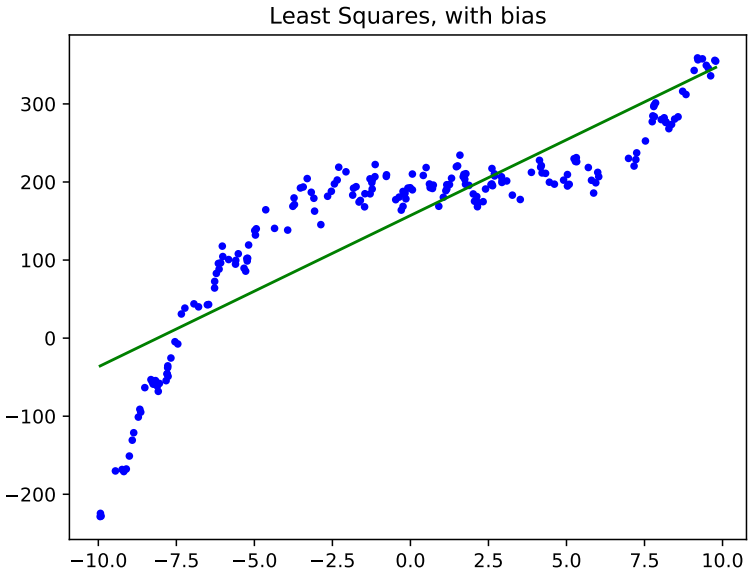
Code: Linked in README, `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_c5u0b_a3/blob/master/code/linear_model.py`



3 Linear Regression and Nonlinear Bases

3.1 Adding a Bias Variable

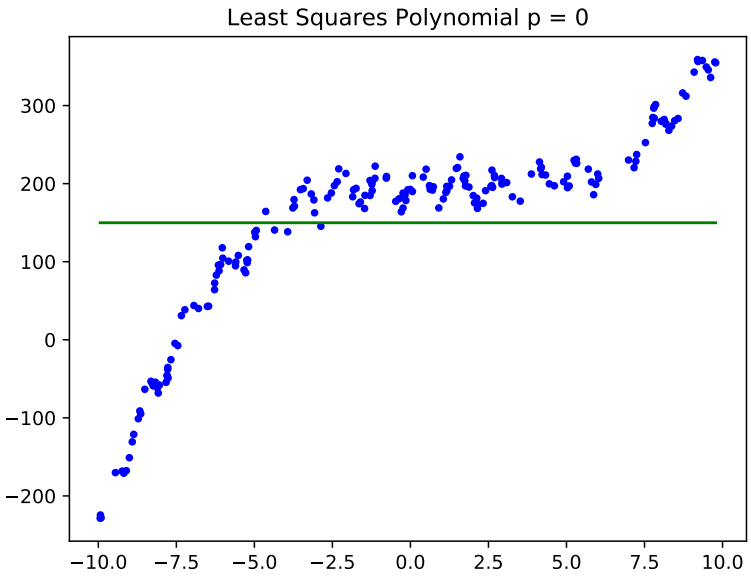
Code: Linked in README, `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_c5u0b_a3/blob/master/code/linear_model.py`



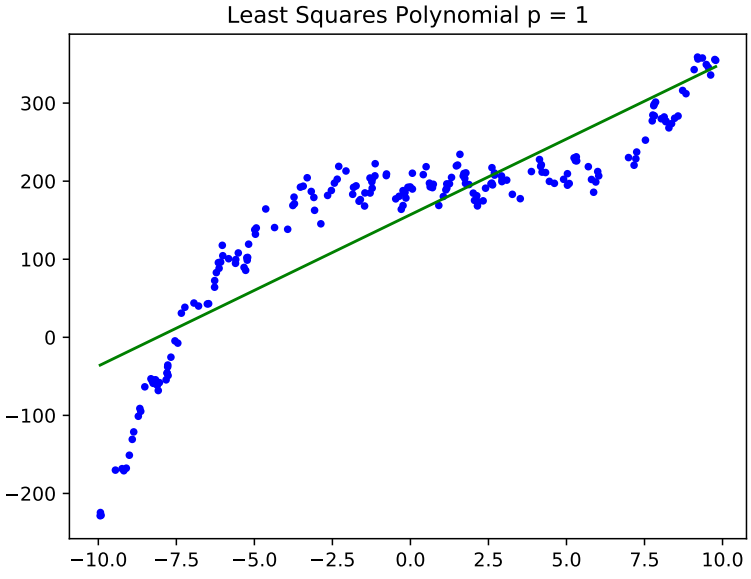
training error = 3551.346
test error = 3393.869

3.2 Polynomial Basis

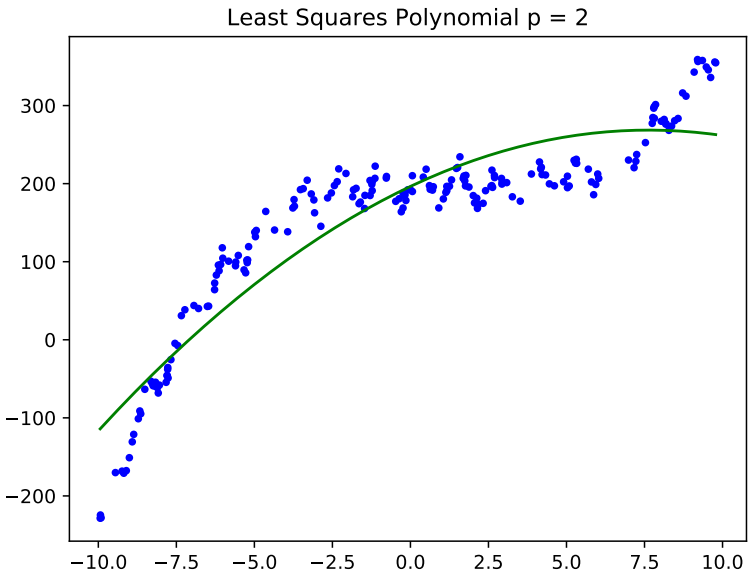
Code: Linked in README, `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_c5u0b_a3/blob/master/code/linear_model.py`



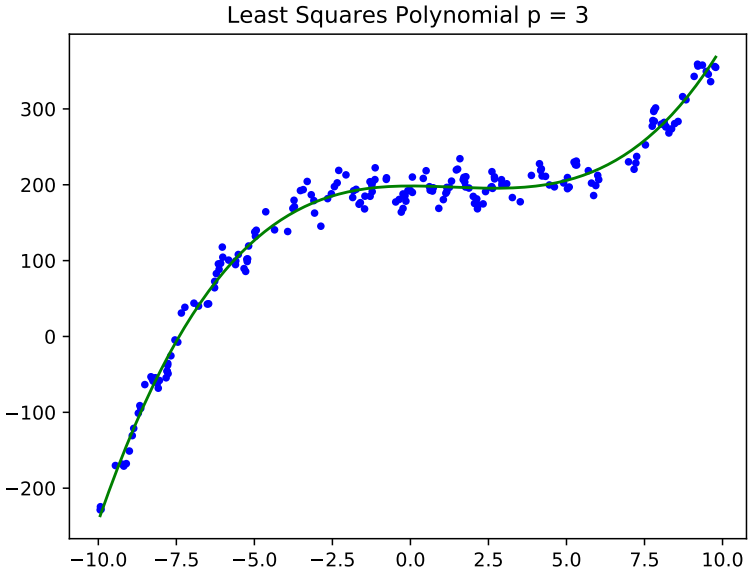
training error = 15480.5
test error = 14390.8



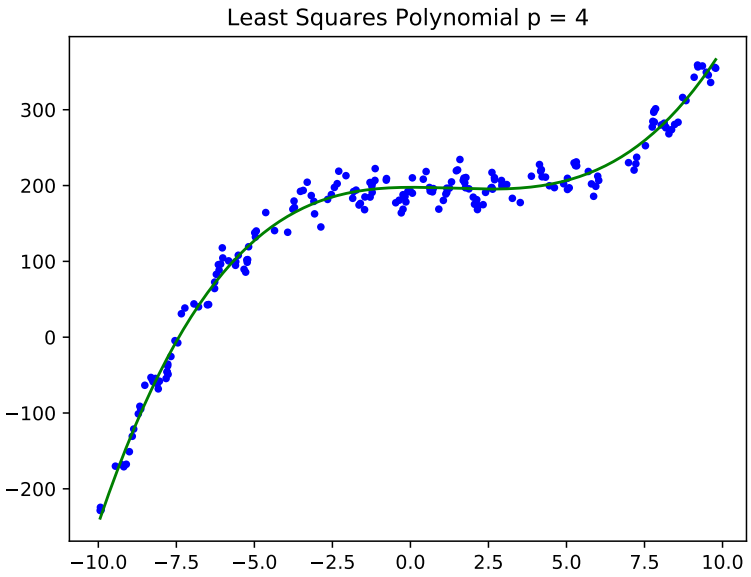
training error = 3551.3
test error = 3393.9



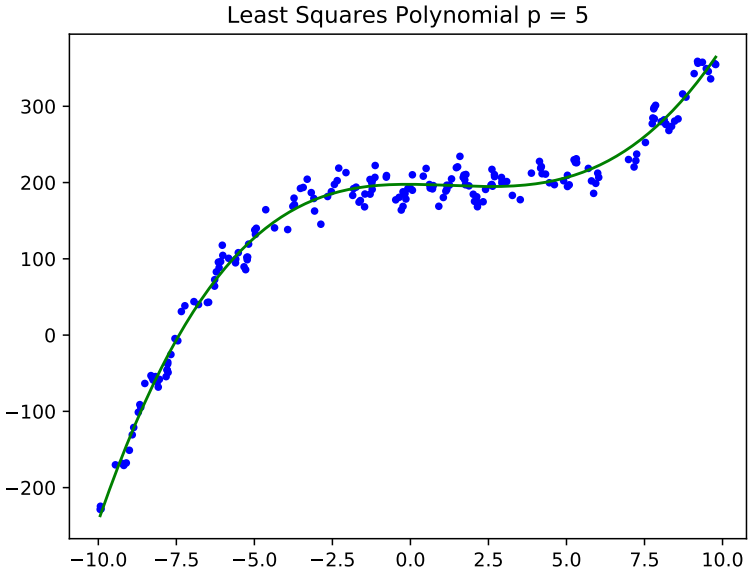
training error = 2168.0
test error = 2480.7



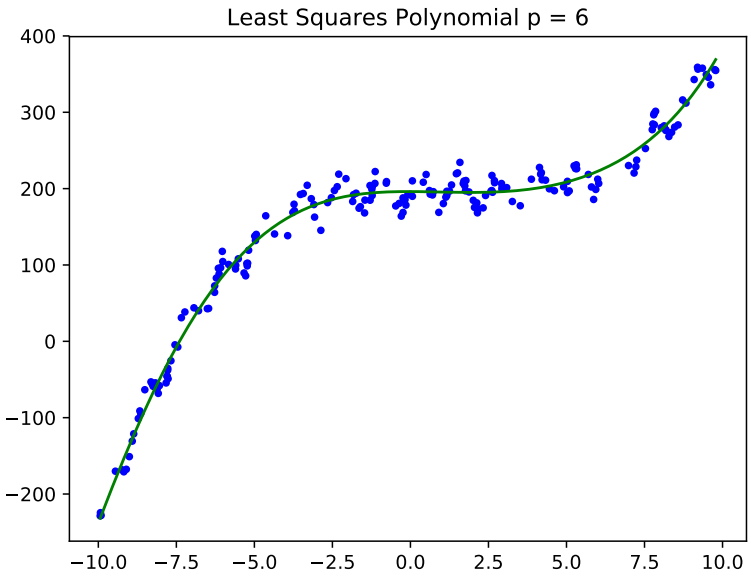
training error = 252.0
test error = 242.8



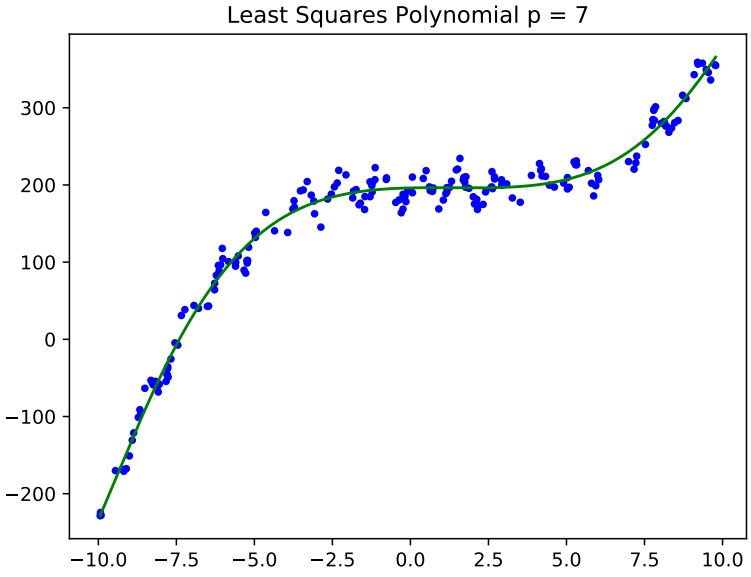
training error = 251.5
test error = 242.1



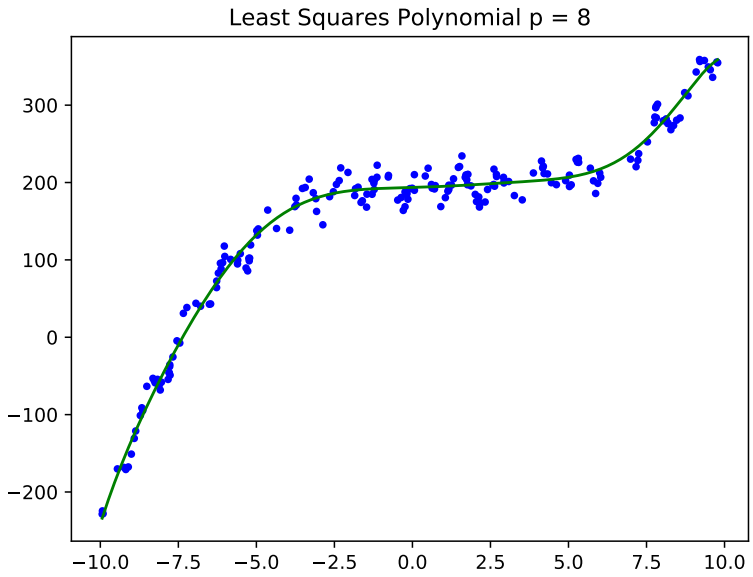
training error = 251.1
test error = 239.5



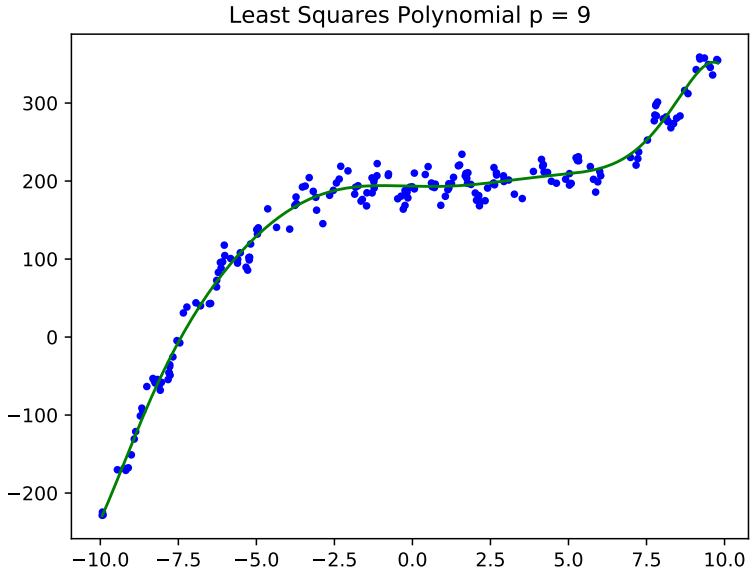
training error = 247.0
test error = 242.9



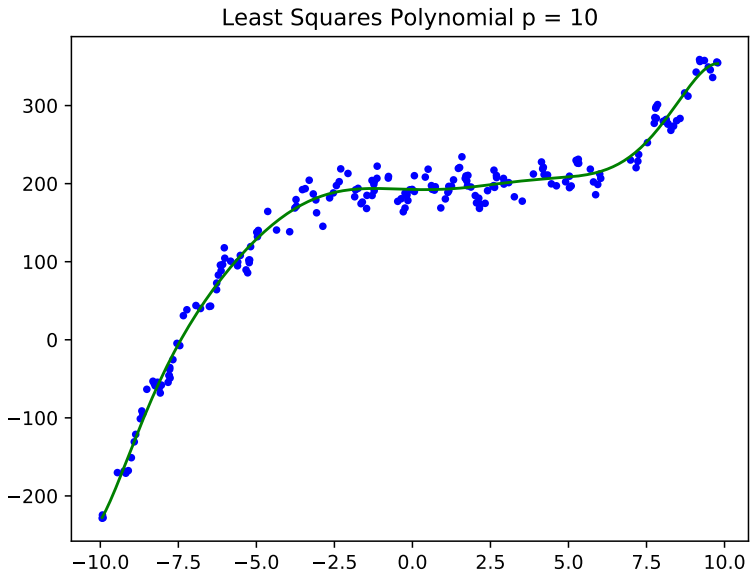
training error = 241.3
test error = 246.0



training error = 241.3
test error = 246.0



training error = 235.8
test error = 259.3



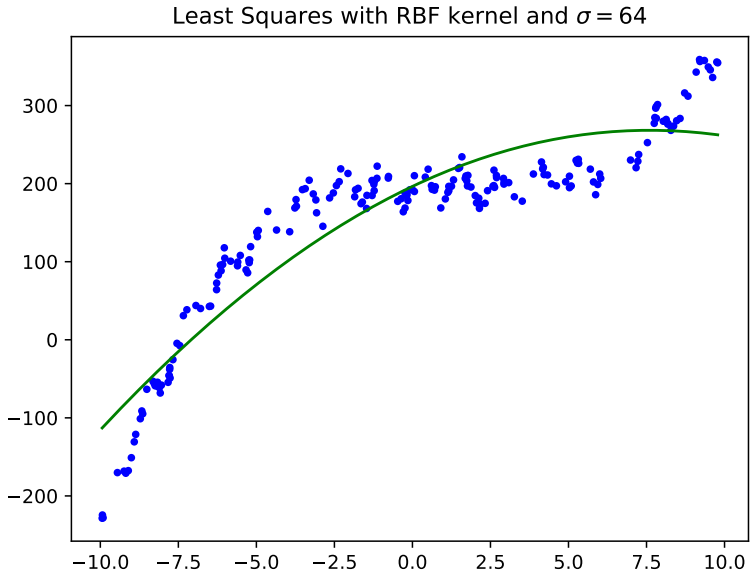
training error = 235.1
test error = 256.3

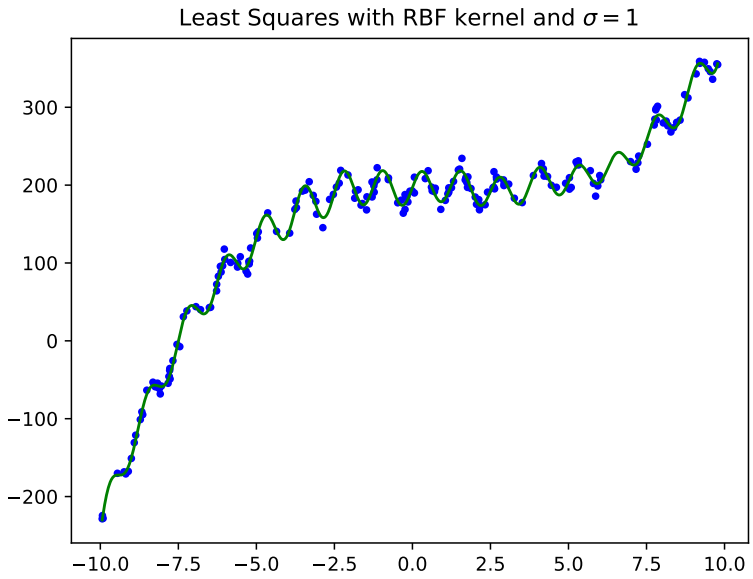
As p increases, the training error decreases. However, the test error decreases initially up to $p = 5$ but then increases as p increases, probably due to overfitting.

4 Non-Parametric Bases and Cross-Validation

4.1 Proper Training and Validation Sets

Without shuffling and data splitting, the train and test errors were 2184.1 and 2495.9 respectively. After shuffling, the train and test errors were 39.5 and 71.2 respectively. The change is around a 50 times decrease in training error and a 35 times decrease in test error.





4.2 Cross-Validation

1. We observed that $\sigma = 1$ and 4 resulted in the lowest errors for both train and test. Surprisingly, we found that $\sigma = 1$ was the most common result when we ran the script multiple times.
2. Code: Linked in README, `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_c5u0b_a3/blob/master/code/main.py`. This procedure typically selects $\sigma = 1$.

4.3 Cost of Non-Parametric Bases

(a) Linear basis:
Training cost: X^TX costs $O(nd^2)$. For the inverse $(X^TX)^{-1}$ costs $O(d^3)$. The total cost is $O(nd^2 + d^3)$.
Classifying cost: Each test costs $O(d)$ to compute $\hat{y}$. For t test examples, the total cost is $O(td)$.

(b) Gaussian RBF:
Training cost: It takes $O(n^2)$ to construct Z, and $O(n^3)$ to compute Z^TZ and the inverse of Z^TZ. The total cost is $O(n^3)$.
Classifying cost: It takes $O(tnd)$ to compute $\hat{Z}$. For each element in $\hat{y}$, it costs $O(n)$ to add the inner products. Since $\hat{y}$ has t entries, the cost is $O(tn)$. In total, classifying t examples is $O(tnd + tn) = O(tnd)$.

For RBFs, since the number of features is equal to n, it will be cheaper to train with RBFs where $n < d$. However, since testing with a linear basis costs $O(td)$ and testing with RBFs costs $O(tnd)$, and $n > 1$, RBFs are rarely cheaper to test than the linear case.

5 Very-Short Answer Questions

5.1 Essentials

1. We compute the squared error $(y_i - \hat{y}_i)^2$ because it gives us a measure of how far apart our data is from the actual value rather than just telling us whether the prediction was exactly correct or not.
2. $A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \end{bmatrix}$
3. $O(n)$
4. When there is multiple clusters of data in the same dataset, a regression tree would be able to isolate those clusters and then linear regression could be run on each of the clusters to produce a more accurate result.
5. With a convex loss function, we know that any minimum found is a global minimum, thus making is easier to solve.
6. When d is very large, gradient descent can be much faster than using the normal equations.
7. When we have datasets in multiple dimensions, it's hard to solve the equation that results in order to set the gradient to 0. Also, some of those equations may not have a solution.
8. The normal equations only work in very specific situations; gradient descent can be applied to any loss function, such as the sum of absolute values.
9. With too small a learning rate, the number of iterations required to reach a solution will be high and thus, computational cost will be high.
10. With too high a learning rate, there is a chance the algorithm will overstep the solution and never be able to actually reach it.