Stat 154 Homework 7

Sharon Hui

3/23/2020

Problem 1 Ridge regression

For the true/false questions, you must provide a justification, but you may cite prior lecture/section materials.

Let $\mathbf{X}$ be an $n * p$ matrix.

Problem 1a

(a) The eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ are nonnegative. True or false?

We know that $||\mathbf{Xv}|| \geq 0$ and equivalently $||\mathbf{Xv}||^2 \geq 0$.

Let λ be an eigenvalue of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Let $\mathbf{v}$ be an eigenvector of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

$$||\mathbf{Xv}||^2 \geq 0$$
$$||\mathbf{Xv}||^2 = (\mathbf{Xv})^{\mathrm{T}}(\mathbf{Xv}) = \mathbf{v}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Xv} \geq 0$$

This means that

$$\mathbf{X}^{\mathrm{T}}\mathbf{Xv} \geq 0$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{Xv} = \lambda\mathbf{v} \geq 0$$

Since we know $\mathbf{v}$ is a nonzero vector (as it is an eigenvector). This means that $\lambda \geq 0$.

In other words, this tells us that the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ are nonnegative.

TRUE

Problem 1b

(b) The eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ are [strictly] positive. True or false? Let λ be a positive real number.

We know that $||\mathbf{Xv}||^2 \geq 0$ is true.

$$||\mathbf{Xv}||^2 \geq 0$$
$$||\mathbf{Xv}||^2 = (\mathbf{Xv})^{\mathrm{T}}(\mathbf{Xv}) = \mathbf{v}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Xv} \geq 0$$

This means that

$$\mathbf{X}^{\mathrm{T}}\mathbf{Xv} \geq 0$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{Xv} = \lambda\mathbf{v} \geq 0$$

Since we know $\mathbf{v}$ is a nonzero vector (as it is an eigenvector). This means that $\lambda \geq 0$.

The eigenvalue can be 0. The eigenvalue is not strictly positive.

FALSE

Problem 1c

(c) Suppose λ_1, λ_2,..., λ_p are the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Determine the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}}$.

Let the SVD of $\mathbf{X}$ be $\mathbf{UDV}^{\mathrm{T}}$

$$\begin{aligned}\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}} &= (\mathbf{UDV}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{UDV}^{\mathrm{T}}) + \lambda\mathbf{I}_{\mathrm{p}} = \mathbf{VD}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{UDV}^{\mathrm{T}} + \lambda\mathbf{I}_{\mathrm{p}} \\ &= \mathbf{VD}^{\mathrm{T}}\mathbf{DV}^{\mathrm{T}} + \lambda\mathbf{I}_{\mathrm{p}} = \mathbf{VD}^{\mathrm{T}}\mathbf{DV}^{\mathrm{T}} + \lambda\mathbf{VV}^{\mathrm{T}} = \mathbf{V}(\mathbf{D}^{\mathrm{T}}\mathbf{D} + \lambda\mathbf{I}_{\mathrm{p}})\mathbf{V}^{\mathrm{T}}\end{aligned}$$

We know that $\mathbf{V}$ and $\mathbf{V}^{\mathrm{T}}$ are orthonormal (multiplying the two matrices together is the identity matrix is important since it tells us that $\mathbf{V}$ and $\mathbf{V}^{\mathrm{T}}$ are inverses of each other). Thus, the diagonal of $\mathbf{D}^{\mathrm{T}}\mathbf{D} + \lambda\mathbf{I}_{\mathrm{p}}$ have the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

This means that

$$\mathbf{D}^{\mathrm{T}}\mathbf{D} + \lambda\mathbf{I}_{\mathrm{p}} = \begin{bmatrix} \lambda_1 + \lambda & 0 & \cdots & 0 \\ 0 & \lambda_2 + \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_p + \lambda \end{bmatrix}$$

The eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}}$ are $\lambda_i^* = \lambda_i + \lambda$.

Alternatively, let λ_a be the eigenvalue of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}}$ and λ_b be the eigenvalue of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Let $\mathbf{v}_i$ be the eigenvector of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}}$

$$\begin{aligned}(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}})\mathbf{v}_i &= \lambda_a\mathbf{v}_i \\ = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{v}_i + \lambda\mathbf{v}_i &= \lambda_a\mathbf{v}_i \\ \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{v}_i = (\lambda_a - \lambda)\mathbf{v}_i &= \lambda_b\mathbf{v}_i\end{aligned}$$

This means $\lambda_a = \lambda_b + \lambda$.

This means that the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{\mathrm{p}}$, λ_a is $\lambda_b + \lambda$

Problem 1d

(d) Use the previous parts to conclude that the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ are strictly positive. Why does this imply $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ is invertible?

Since we know that all of the eigenvalues of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ are strictly positive since λ_b is at least 0 and λ is more than 0, this means that $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ is full rank the nullity is 0. We know that this also means that the only vector in the null space of $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ is the $\mathbf{0}$ vector.

Since $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ is full rank, this implies $\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_p$ is invertible.

Problem 2

It is often said the LASSO performs feature selection. That is, it will select a few important variables andset the coefficients of all others to zero - it encourages sparsity. Why does this happen? One answer is that the l^1 norm (denoted as $||\cdot||_1$) is a "good" surrogate for the sparsity-enforcing l^0 norm (denoted as $||\cdot||_0$). The l^0 "norm" (not strictly a norm) for a vector $\mathbf{x} \in \mathbb{R}^n$ is defined by

$$||x||_0 \equiv \text{number of non-zero entries in } \mathbf{x}$$

Why is l^1 a "good" surrogate? One answer is that it is convex. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if it satisfies the following inequality:

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \text{for all}\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$$

Problem 2a

a. For the one-dimensional case ($n = 1$), show that the l^1 norm is convex.

Recall that by the triangle inequality $|x_1 + ...x_n| \leq |x_1| + ... + |x_n|$ and we know that since $\lambda \in [0,1]$ then $\lambda > 0$ and $1 - \lambda > 0$

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) = |\lambda\mathbf{x} + (1-\lambda)\mathbf{y}| \leq (|\lambda\mathbf{x}| + |(1-\lambda)\mathbf{y}|) = \lambda|\mathbf{x}| + (1-\lambda)|\mathbf{y}| = \lambda f(\mathbf{x}) + (1-\lambda)\mathbf{y}$$

This means that $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)\mathbf{y}$ is satisfied for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$

Furthermore, notice that we can realize that since $n = 1$, the function of the l_1 norm is an absolute value function centered at 0, which is convex graphically.

Problem 2b

b. For the one-dimensional case ($n = 1$), show that the l^0 norm is *not* convex.

Let a be the number of non-zero elements in vector $\mathbf{x}$. Let b be the number of non-zero elements in vector $\mathbf{y}$.

Let's make a further assumption that $\mathbf{x}$ and $\mathbf{y}$ have positive elements with no overlapping (as in $\mathbf{x}$ does not have non-zero elements where $\mathbf{y}$ does have non-zero elements and vice versa) and $\lambda > 0$

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) = ||(\lambda\mathbf{x} + (1-\lambda)\mathbf{y})||_0 = a + b$$

$$\lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) = \lambda a + (1-\lambda)b = b + \lambda(a-b)$$

Also, notice that if $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \text{for all}\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$ holds true then, $a + b \leq b + \lambda(a-b)$.

However,

$$(a+b) - (b + \lambda(a-b)) = a + b - b - \lambda(a-b) = a - \lambda a + b > 0$$

which implies that $(a+b) \geq (b + \lambda(a-b))$.

This is a contradiction that $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \text{for all}\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$. This implies that it can be concave.

In another case, let's assume that there is some overlap.

Let a be the number of non-zero elements in vector $\mathbf{x}$. Let b be the number of non-zero elements in vector $\mathbf{y}$. Furthermore, let's assume that $\mathbf{x}$ and $\mathbf{y}$ have positive elements that overlap in their position and assume that $a \geq b$ This means that

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) = ||(\lambda\mathbf{x} + (1-\lambda)\mathbf{y})||_0 = a$$

and

$$\lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) = \lambda a + (1-\lambda)b = b + \lambda(a-b)$$

Also, notice that if $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \text{for all}\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$ holds true then, $a \leq b + \lambda(a-b)$.

However,

$$a - (b + \lambda(a-b)) = a - b - \lambda a + \lambda b = (a - \lambda a) - (b - \lambda b) = a(1-\lambda) - b(1-\lambda) = (a-b)(1-\lambda) > 0$$

This is also a contradiction that $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \text{for all}\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$. This implies that it can be concave.

Furthermore, notice that we can realize that since $n = 1$, for all values of $\mathbf{x}$ and $\mathbf{y}$, this is just a line of $y = 1$, which is not convex.

More concretely, take the case that $\lambda = 0.25$ and $x = 0$ and $y = 1$

This tells us

$$||(\lambda\mathbf{x} + (1-\lambda)\mathbf{y})||_0 = ||(.25 * 0 + (1-0.25) * 1)||_0 = ||.75||_0 = 1$$

however,

$$\lambda||(\mathbf{x})||_0 + (1-\lambda)||(\mathbf{y})||_0 = .25 * 0 + (1-.25) * 1 = 0.75$$

and we know $1 < 0.75$

Thus, l^0 norm is *not* convex.

Problem 2c

c. For the one-dimensional case, let f be another convex function such that $f(x) \leq ||x||_0 \text{for all}\, x \in [-1,1]$. Show that $f(x) \leq ||x||_1 \text{for all}\, x \in [-1,1]$.

We know the following:

$$f(1) \leq ||1||_0 = 1$$
$$f(-1) \leq ||1||_0 = -1$$
$$f(0) \leq ||0||_0 = 0$$

Consider the case when $x \in (0,1)$

$$f(x) = f(x * 1 + (1-x) * 0) < xf(1) + (1-x)f(0) = x$$

Consider the case when $-x \in (-1,0)$

$$f(-x) = f(x * (-1) + (1-x) * 0) < xf(-1) + (1-x)f(0) = -x$$

This means

$$f(x) < x \,\text{and}\, f(-x) < -x$$

Combining these two cases, for $x \in [-1, 1]$,

$$f(x) \leq |x| = ||x||_1$$

We have just shown that on the interval $[0, 1]$, the l^1 norm is the largest convex function that underestimates the l^0 norm: formally, the l^1 norm is the lower convex envelope of the l^0 norm on the interval $[-1, 1]$. Onecan think of this as the l^1 norm being the "best" convex approximation to the‘0norm, at least locally. It turns out that this is true for the multi-dimensional case too. In part because the l^1 norm is convex, it can be shown that minimizing $||\mathbf{y} - \mathbf{X}\beta||^2 + \lambda||\beta||_1$ is the same as solving the constrained optimization problem:

$$\begin{gathered} \min||\mathbf{y} - \mathbf{X}\beta||^2 \\ \text{subject to}\, ||\beta||_1 \leq \mu \end{gathered}$$

or some appropriate μ. Since l^1 norm is the "closest" convex approximation of the l^0 norm, this l^1 constrained optimization problem is "close" to the l^0 constrained optimization problem. This is why the Lasso tends to do feature selection.

Problem 3 Explicit solutions for ridge and LASSO in a special scenario

We consider a very simplified situation where both the ridge regression and LASSO optimization problemsdecompose nicely. For the rest of this question, consider the special case $n = p$ and

$$\mathbf{X} = \mathbf{I}_\mathrm{p}$$

(No intercept.) Let $\mathbf{y} \in \mathbb{R}^p$ be the response variable.

Problem 3a

a) Determine the OLS coefficients (solution to $\min_\mathrm{b} ||\mathbf{y} - \mathbf{X}\mathbf{b}||_2^2$).

$$\mathbf{b} = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y} = ((\mathbf{I}_\mathrm{p})^\mathrm{T}\mathbf{I}_\mathrm{p})^{-1}(\mathbf{I}_\mathrm{p})^\mathrm{T}\mathbf{y} = \mathbf{y}$$

The OLS coefficients are $\mathbf{y}$. $\mathbf{b} = \mathbf{y}$

Problem 3b

(b) Explicitly write down the solution to the ridge regression optimization problem

$$\min_b ||\mathbf{y} - \mathbf{X}\mathbf{b}||_2^2 + \lambda||\mathbf{b}||_2^2$$

where $\lambda > 0$ Explain why the solution is a "shrinkage" of the OLS coefficients, with the shrinkage effect being stronger as λ is larger.

Given:

$$\min_b ||\mathbf{y} - \mathbf{X}\mathbf{b}||_2^2 + \lambda||\mathbf{b}||_2^2$$

$$(\mathbf{y} - \mathbf{X}\mathbf{b})^\mathrm{T}(\mathbf{y} - \mathbf{X}\mathbf{b}) + \lambda\mathbf{b}^\mathrm{T}\mathbf{b} = \mathbf{y}^\mathrm{T}\mathbf{y} - 2\mathbf{b}^\mathrm{T}\mathbf{X}^\mathrm{T}\mathbf{y} + \mathbf{b}^\mathrm{T}\mathbf{X}^\mathrm{T}\mathbf{X}\mathbf{b} + \lambda\mathbf{b}^\mathrm{T}\mathbf{b}$$

Take the derivative and set it equal to 0.

$$-2\mathbf{X}^\mathrm{T}\mathbf{y} + 2\mathbf{X}^\mathrm{T}\mathbf{X}\mathbf{b} + 2\lambda\mathbf{b} = 0$$
$$\mathbf{X}^\mathrm{T}\mathbf{y} = \mathbf{X}^\mathrm{T}\mathbf{X}\mathbf{b} + \lambda\mathbf{b} = (\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I}_p)\mathbf{b}$$

$$\mathbf{b} = (\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\mathrm{T}\mathbf{y}$$

As λ becomes larger and larger, $(\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I}_p)$ gets larger but it becomes closer to the matrix $\lambda\mathbf{I}_p$ since λ will dominate. This further tells us that $(\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I}_p)^{-1}$ will get closer to $(\lambda\mathbf{I}_p)^{-1}$.

$\mathbf{b} = (\mathbf{X}^\mathrm{T}\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\mathrm{T}\mathbf{y} = (\mathbf{I}_p + \lambda\mathbf{I}_p)^{-1}\mathbf{I}_p\mathbf{y} = (\mathbf{I}_p + \lambda\mathbf{I}_p)^{-1}\mathbf{y}$.

More explicitly, this tells us if λ becomes larger and larger, then

$$\mathbf{b} = (\mathbf{I}_p{+}\lambda\mathbf{I}_p)^{-1}\mathbf{y} = \begin{bmatrix} 1+\lambda & 0 & \cdots & 0 \\ 0 & 1+\lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1+\lambda \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \frac{1}{(\lambda+1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{(\lambda+1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \frac{1}{(\lambda+1)} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \frac{1}{(\lambda+1)} * y_1 \\ \frac{1}{(\lambda+1)} * y_2 \\ \vdots \\ \frac{1}{(\lambda+1)} * y_n \end{bmatrix}$$

this means that $\mathbf{b}$ will become $\mathbf{0}$. The larger the λ , the closer the ridge coefficients are to zero.

Problem 3c

(c) Consider the LASSO optimization problem

$$\min_{b} ||\mathbf{y} - \mathbf{X}\mathbf{b}||_2^2 + \lambda||\mathbf{b}||_1$$

Show that the jth entry of the LASSO coefficients $\mathbf{b}_{\text{LASSO}}$ can be found by solving the following optimization problem.

$$\min_{b_j} (y_j - b_j)^2 + \lambda|b_j|$$

(Actually solving the latter problem will be discussed in the next part.)

First, notice that:

$$||\mathbf{y} - \mathbf{X}\mathbf{b}||_2^2 + \lambda||\mathbf{b}||_1 = (\mathbf{y} - \mathbf{X}\mathbf{b})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\mathbf{b}) + \lambda\sum_{j=1}^{p}|b_j$$

$$= \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{b}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{b}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{b} + \lambda\sum_{j=1}^{p}|b_j| = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{b}^{\mathrm{T}}\mathbf{I}_{\mathrm{p}}\mathbf{y} + \mathbf{b}^{\mathrm{T}}\mathbf{I}_{\mathrm{p}}\mathbf{b} + \lambda\sum_{j=1}^{p}|b_j|$$

$$= \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{b}^{\mathrm{T}}\mathbf{y} + \mathbf{b}^{\mathrm{T}}\mathbf{b} + \lambda\sum_{j=1}^{p}|b_j| = (\mathbf{y} - \mathbf{b})^{\mathrm{T}}(\mathbf{y} - \mathbf{b}) + \lambda\sum_{j=1}^{p}|b_j| = (y_j - b_j)^2 + \lambda|b_j|$$

Since the LASSO solution is iunique this tells us that to minimize the $(\mathbf{y} - \mathbf{b})^{\mathrm{T}}(\mathbf{y} - \mathbf{b}) + \lambda\sum_{j=1}^{p}|b_j|$, we must can choose a b_j that minimizes $(y_j - b_j)^2 + \lambda|b_j|$.

Problem 3d

(d) For a parameter $t > 0$, we define the **soft-thresholding function** S_t by

$$S_t(u) = \begin{cases} u - t & u > t \\ 0 & |u| \leq t \\ u + t & u < -t \end{cases}$$

Check that the solution to the optimization problem (from the previous question) is $S_{\lambda/2}(y_j)$. (You may assume that a solution exists and is unique. You are only asked to check that $S_{\lambda/2}(y_j)$ is the solution.) Hint: the non-differentiability of the absolute value makes the usual approach not possible. To get around this, consider separately the cases where the solution to optimization problem (from the previous question) is positive, is negative, and is zero.

This is saying:

$$S_{\lambda/2}(y_j) = \begin{cases} y_j - \lambda/2 & y_j > \lambda/2 \\ 0 & |y_j| \leq \lambda/2 \\ y_j + \lambda/2 & y_j < -\lambda/2 \end{cases}$$

Take the deriviative of $(y_j - b_j)^2 + \lambda|b_j|$ with respect to b_j

$$\frac{\partial}{\partial b_j}((y_j - b_j)^2 + \lambda|b_j|) = \frac{\partial}{\partial b_j}(y_j - 2b_j y_j + b_j + \lambda b_j) = -2y_j + 2b_j + \lambda\frac{b_j}{|b_j|} = 0$$

When $b_j > 0$, this means $y_j > \lambda/2$ and:

$$-2y_j + 2b_j + \lambda\frac{b_j}{|b_j|} = -2y_j + 2(y_j - \lambda/2) + \lambda\frac{y_j - \lambda/2}{|y_j - \lambda/2|} = -2y_j + 2y_j - \lambda + \lambda * (1) = (-2y_j + 2y_j) + (-\lambda + \lambda) = 0$$

When $b_j < 0$, this means $y_j < -\lambda/2$ and:

$$-2y_j + 2(y_j + \lambda/2) + \lambda\frac{y_j + \lambda/2}{|y_j + \lambda/2|} = -2y_j + 2y_j + \lambda + \lambda * (-1) = (-2y_j + 2y_j) + (\lambda - \lambda) = 0$$

The middle case considers what happens when $b_j = 0$. We know that when $b_j \not< 0$ and $b_j \not> 0$ which tells us that $y_j \not> \lambda/2$ and $y_j \not< \lambda/2$. In other words, that tells us $|y_j| \leq \lambda/2$.

Problem 3e

(e) Explain why the solution $\mathbf{b}_{LASSO} = (S_{\lambda/2}(y_1), ..., S_{\lambda/2}(y_p))$ to the LASSO optimization problem is a "shrinkage" of the OLS solution. Which components of the LASSO solution are exactly zero? Which components of the ridge regression solution are exactly zero?

$$S_{\lambda/2}(y_j) = \begin{cases} y_j - \lambda/2 & y_j > \lambda/2 \\ 0 & |y_j| \leq \lambda/2 \\ y_j + \lambda/2 & y_j < -\lambda/2 \end{cases}$$

In the first case, $y_j > \lambda/2$, thus $y_j - \lambda/2 = b_j$ is positive. As we increase λ, $y_j - \lambda/2$ gets smaller and smaller so the b_j coefficient will shrink towards zero.

In the second case, when $|y_j| \leq \lambda/2$ the b_j coefficient will be exactly zero.

In the thid case, $y_j < -\lambda/2$, thus $y_j - \lambda/2 = b_j$ is negative. As we increase λ, $y_j - \lambda/2$ gets less and less negative (moving towards 0) so the b_j coefficient will shrink towards zero.

From the cases, we can see that $lim_{\lambda \to \infty} S_{\lambda/2}(y_i) = 0$. From this, we can see that this shrinks the coefficient values towards 0 as we increase λ. Compared to ridge regression, lasso tends to give a set of zero regression coefficients and leads to a sparse solution.

From part b for Ridge, we know that b_j will be 0 when $\frac{1}{(\lambda+1)} * y_j$ which only occurs when $y_j = 0$. Ridge yielding $b_j = 0$ has a smaller set compared to LASSO where we have $b_j = 0$ when $|y_j| \leq \lambda/2$. This tells us that LASSO will tend to set b to 0 more often than ridge.

Problem 4

Consider the data generating model:

$y = f(x) + \epsilon = x^3 - 3x + \epsilon;\ \ \epsilon \sim N(0,1),\ \ x \sim Unif[-2,2]$

Problem 4a

a. Implement a function that will perform kNN regression. It should have four arguments: k: the number of neighbors, z: a vector containing the points on which we want to predict, and x/y: the input/response data.

```
distance <- function(a, b){
  return( abs(a-b))
}
```

```
knn <- function(num_neighbor, point_predict, input, response){
  point_predict = matrix(point_predict, ncol = 1)
  input = matrix(input, ncol = 1)
  stored_resp = c()
  for (i in 1:nrow(point_predict)){
    bunch_dist = c()
    for (j in 1:nrow(input)){
      bunch_dist = c(bunch_dist, distance(point_predict[i, 1], input[j, 1])) #get all distances of z_i
    }
    temp_mat = data.frame(input, response, bunch_dist) #df of input, resp, dist
    new_temp_mat = temp_mat[order(temp_mat$bunch_dist, decreasing = F),] #order by distance
    result_resp = head(new_temp_mat, num_neighbor)$response
    stored_resp = c(stored_resp, mean(result_resp))
  }
  return(stored_resp)
}
```

In my `knn` function, I calculate all the distances of z_i to the input values and choose the k smallest distances of z_i and my inputs. Recall that k is the number of neighbhors that I want. Then, I will obtain the responses corresponding the the input that yielded the smallest distances. Then I average their responses based. I repeat this for all z_i which are specified by `point_predict`

Problem 4b

b. Run the following experiment 100 times.

c. Generate a dataset of 30 points

ii. Using kNN regression for $k = 4$, compute and save both the predictions $(\hat{f}k(x) - f(x))$ at points $-1, 0, 1$.

```
set.seed(22)
obs = 30
z_4 = c(-1,0,1)
k_4 = 4
times = 100
true_z_resp = c(2, 0, -2) # actual response values from the true function

predictions = list()
error = list()
```

```r
for (i in 1:times){
  x_4 = runif(obs, min=-2, max = 2)
  y_4 = (x_4)^3-3*(x_4) + rnorm(obs, mean = 0, sd = 1)
  temp_results = knn(k_4, z_4, x_4, y_4)
  predictions[[i]] = temp_results
  error[[i]] = temp_results - true_z_resp
}
```

From the above experiment, estimate the point-wise squared bias and variance at each of the above points. Is there a point that kNN performs particularly well for?

```r
error_1st = c()
for (i in 1:length(error)){
  error_1st = c(error_1st, error[[i]][1])
}
(mean(error_1st))^2
```

```
## [1] 0.01663193
```

```r
predictions_1st = c()
for (i in 1:length(predictions)){
  predictions_1st = c(predictions_1st, predictions[[i]][1])
}
(sd(predictions_1st))^2
```

```
## [1] 0.2577642
```

```r
error_2nd = c()
for (i in 1:length(error)){
  error_2nd = c(error_2nd, error[[i]][2])
}
(mean(error_2nd))^2
```

```
## [1] 0.002417144
```

```r
predictions_2nd = c()
for (i in 1:length(predictions)){
  predictions_2nd = c(predictions_2nd, predictions[[i]][2])
}
(sd(predictions_2nd))^2
```

```
## [1] 0.3300612
```

```r
error_3rd = c()
for (i in 1:length(error)){
  error_3rd = c(error_3rd, error[[i]][3])
}
(mean(error_3rd))^2
```

```
## [1] 0.003073335
```

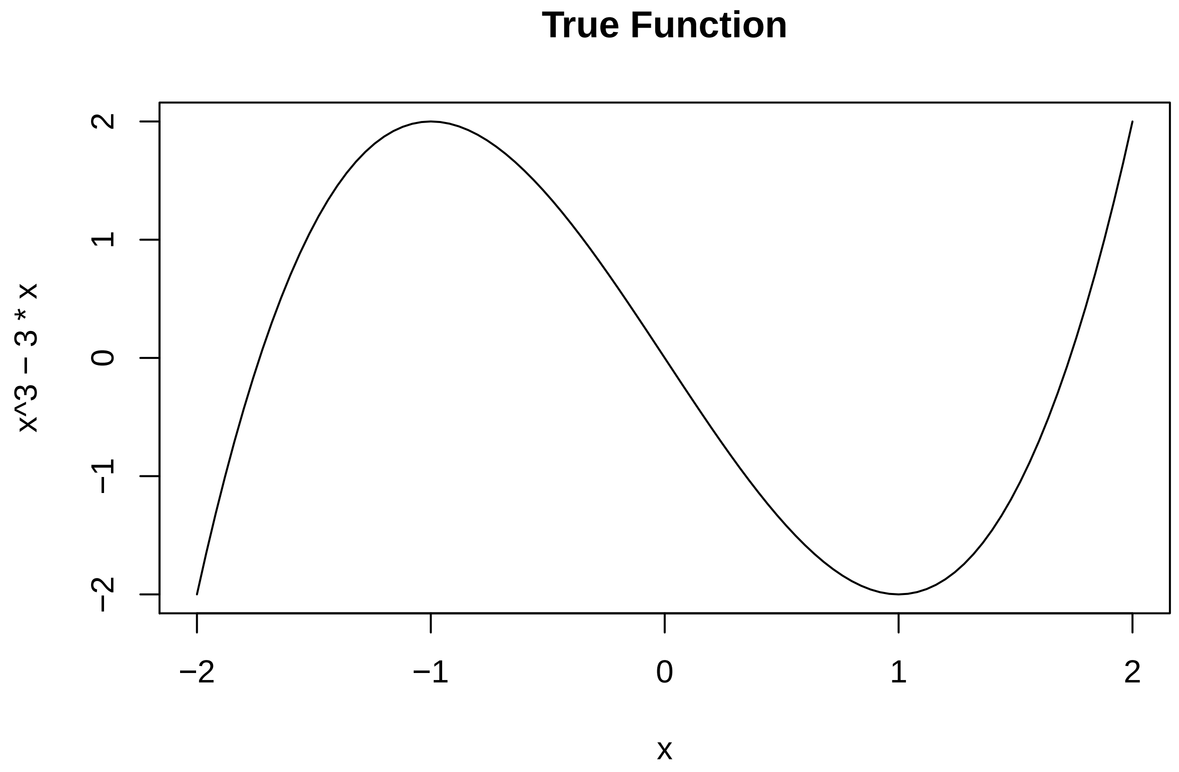
```r
predictions_3rd = c()
for (i in 1:length(predictions)){
  predictions_3rd = c(predictions_3rd, predictions[[i]][3])
}
(sd(predictions_3rd))^2
```

```
## [1] 0.3107504
```

```
df1 = data.frame("Squared Bias" = c((mean(error_1st))^2,
                              (mean(error_2nd))^2, (mean(error_3rd))^2),
                 "Variances" = c((sd(predictions_1st))^2, (sd(predictions_2nd))^2, (sd(predictions_3rd)
rownames(df1) = c("-1", "0", "1")
df1
```

```
##    Squared.Bias Variances
## -1  0.016631929 0.2577642
## 0   0.002417144 0.3300612
## 1   0.003073335 0.3107504
```

```
curve(x^3-3*x, -2, 2, main = "True Function")
```



Notice that the point at 0 has a particularly low bias compared to the other numbers. This is the point that kNN performs particularly well for.

The reason why this might make sense is that around close values of 0, we can notice a linear trend. Meanwhile at nearby points -1 and 1, the function is difficult to approximate with linear functions. As stated in part 4d, the point is unbiased when we have a function that is linear. The function is more complex at -1 and 1.

Problem 4c

c. Repeat part b for $k = 1$ and $k = 10$. How does k affect the point-wise bias and variance?

```
set.seed(22)
obs = 30
z_1 = c(-1,0,1)
k_1 = 1
k_10 = 10
times = 100
true_z_resp = c(2, 0, -2) # actual response values from the true function
```

```r
predictions1 = list()
predictions10 = list()
for (i in 1:times){
  x_1 = runif(obs, min=-2, max = 2)
  y_1 = (x_1)^3-3*(x_1) + rnorm(obs, mean = 0, sd = 1)
  predictions1[[i]] = knn(k_1, z_1, x_1, y_1)
  predictions10[[i]] = knn(k_10, z_1, x_1, y_1)
}

error = list()
error10 = list()
for (i in 1:length(predictions1)){
  error[[i]] = predictions1[[i]] - true_z_resp
  error10[[i]] = predictions10[[i]] - true_z_resp
}
```

```r
error_1st2 = c()
for (i in 1:length(error)){
  error_1st2 = c(error_1st2, error[[i]][1])
}
(mean(error_1st2))^2
```

```
## [1] 0.006304114
```

```r
predictions1_1st = c()
for (i in 1:length(predictions1)){
  predictions1_1st = c(predictions1_1st, predictions1[[i]][1])
}
(sd(predictions1_1st))^2
```

```
## [1] 0.977348
```

```r
error_2nd2 = c()
for (i in 1:length(error)){
  error_2nd2 = c(error_2nd2, error[[i]][2])
}
(mean(error_2nd2))^2
```

```
## [1] 0.01915848
```

```r
predictions1_2nd = c()
for (i in 1:length(predictions1)){
  predictions1_2nd = c(predictions1_2nd, predictions1[[i]][2])
}
(sd(predictions1_2nd))^2
```

```
## [1] 0.8577229
```

```r
error_3rd2 = c()
for (i in 1:length(error)){
  error_3rd2 = c(error_3rd2, error[[i]][3])
}
(mean(error_3rd2))^2
```

```
## [1] 0.01740109
```

```
predictions1_3rd = c()
for (i in 1:length(predictions1)){
  predictions1_3rd = c(predictions1_3rd, predictions1[[i]][3])
}
(sd(predictions1_3rd))^2
```

```
## [1] 1.091456
```

```
df2 = data.frame("Squared Bias" = c((mean(error_1st2))^2,
                             (mean(error_2nd2))^2, (mean(error_3rd2))^2),
                 "Variances" = c((sd(predictions1_1st))^2, (sd(predictions1_2nd))^2, (sd(predictions1_3r
rownames(df2) = c("-1", "0", "1")
df2
```

```
##    Squared.Bias Variances
## -1  0.006304114 0.9773480
## 0   0.019158480 0.8577229
## 1   0.017401091 1.0914561
```

```
error_1st = c()
for (i in 1:length(error10)){
  error_1st = c(error_1st, error10[[i]][1])
}
(mean(error_1st))^2
```

```
## [1] 0.2745368
```

```
predictions10_1st = c()
for (i in 1:length(predictions10)){
  predictions10_1st = c(predictions10_1st, predictions10[[i]][1])
}
(sd(predictions10_1st))^2
```

```
## [1] 0.1639617
```

```
error_2nd = c()
for (i in 1:length(error10)){
  error_2nd = c(error_2nd, error10[[i]][2])
}
(mean(error_2nd))^2
```

```
## [1] 0.0005302271
```

```
predictions10_2nd = c()
for (i in 1:length(predictions10)){
  predictions10_2nd = c(predictions10_2nd, predictions10[[i]][2])
}
(sd(predictions10_2nd))^2
```

```
## [1] 0.2124065
```

```
error_3rd = c()
for (i in 1:length(error10)){
  error_3rd = c(error_3rd, error10[[i]][3])
}
(mean(error_3rd))^2
```

```
## [1] 0.2427293
```

```
predictions10_3rd = c()
for (i in 1:length(predictions10)){
  predictions10_3rd = c(predictions10_3rd, predictions10[[i]][3])
}
(sd(predictions10_3rd))^2
```

```
## [1] 0.2051505
```

```
df3 = data.frame("Squared Bias" = c((mean(error_1st))^2,
                             (mean(error_2nd))^2,
                             (mean(error_3rd))^2),
                 "Variances" = c((sd(predictions10_1st))^2, (sd(predictions10_2nd))^2, (sd(predictions1
rownames(df2) = c("-1", "0", "1")
df3
```

```
##   Squared.Bias Variances
## 1 0.2745367573 0.1639617
## 2 0.0005302271 0.2124065
## 3 0.2427293114 0.2051505
```

As expected, as we increase k, the variance tends to fall but at the expense of increasing bias. Large values for k result in more inflexible fits (i.e. more bias), but also less variance. Conversely, small values for k result in more flexible fits (i.e. less bias), at the expense of increasing the variance.

From parts b and c, we found that the kNN regression performed particularly well at 0 regardless of our choice of k . Why is that? One answer is approximate linearity.

Problem 4d

d. Suppose we have data (x_i, y_i) for $i = 1, ..., n$, where $y_i = f(x_i) + \epsilon$, where ϵ is $i.i.d.$ $N(0,1)$ noise, and where $f(x) = mx + b$ is a linear function.

Suppose the k neighbors of a new point x_0 happen to be $x_1, ..., x_k$, and that $\frac{1}{k}\sum_{i=1}^{k} x_i = x_0$ (the neighbors are centered around x_0). Compute the expected value of the predicted value $\hat{f}_k(x_0)$ that kNN regression produces. (Note that unlike the previous parts, we are considering the x_i values as non-random in this question.) In this scenario, is kNN unbiased?

$$\hat{f}_k(x_0) = \frac{1}{k}\sum_{i=1}^{k} y_i$$

So,

$$\mathbb{E}(\hat{f}_k(x_0)) = \mathbb{E}\left(\frac{1}{k}\sum_{i=1}^{k} y_i\right) = \frac{1}{k}\mathbb{E}(y_1 + ...y_k) = \frac{1}{k} * (\sum_{i=1}^{k}\mathbb{E}(y_i)) = \frac{1}{k} * (\sum_{i=1}^{k}\mathbb{E}(f(x_i) + \epsilon))$$

$$= \frac{1}{k} * (\sum_{i=1}^{k}\mathbb{E}(mx_i + b + \epsilon)) = \frac{1}{k} * (\sum_{i=1}^{k}(mx_i + b)) = m * \frac{1}{k}\sum_{i=1}^{k}(x_i) + \frac{1}{k} * \sum_{i=1}^{k} b = mx_0 + b$$

We also know:

$$f_k(x_0) = mx_0 + b$$

Since $\mathbb{E}(\hat{f}_k(x_0)) = mx_0 + b = f_k(x_0)$, this tells us in this scenario, kNN is unbiased.

Problem 4e

e. In the setting of the previous sub-question, compute the variance of $\hat{f}_k(x_0)$.

$$var(\hat{f}_k(x_0)) = var\left(\frac{1}{k}\sum_{i=1}^{k} y_i\right) = \frac{1}{k^2}var(\sum_{i=1}^{k} y_i) = \frac{1}{k^2}\sum_{i=1}^{k} var((f(x_i)+\epsilon))$$

$$= \frac{1}{k^2}\sum_{i=1}^{k} var((mx_i + b + \epsilon)) = \frac{1}{k^2}\sum_{i=1}^{k}(1) = \frac{k}{k^2} = \frac{1}{k}$$

Since the polynomial $x^3 - 3x$ is approximately linear about 0 but very "curvy" around 1 and -1, kNN regression performs much better at 0. That is, kNN peforms well at points where the trend function f is locally linear.