homework4

December 13, 2022

1 Homework 4

Subject sentence

1. a. Show that the level sets (the collection of values of β_1 and β_2 such that $Q(B) := \sum(y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 = k$) are in fact ellipses as claimed in class. According to https://www.maa.org/external_archive/joma/Volume8/Kalman/General.html, "the general equation for a rotated ellipse centered at (h, k) has the form $A(x-h)^2 + B(x-h)(y-k) + C(y-k)^2 = 1$, again where A and C are positive, and $B^2 - 4AC < 0$" and that a rotated ellipse has a non-zero xy term.

$\sum(y_i - \beta_1 x_{1i} - \beta_2 x_{2i}) = \sum (y_i^2 + \beta_1^2 x_{1i}^2 + \beta_2^2 x_{2i}^2 + \beta_1\beta_2 x_{1i}^2 x_{2i}^2 - y_i(\beta_1 x_{1i} + \beta_1^2 x_{2i}^2))$

These terms can then be rearranged into the general equation for a rotated ellipse.

b. State what would have to be true about the data in order for this to be a standard ellipse (major and minor axes aligned with coordinate axis) (see the coefficient of the xy term in equation 1).

If the major and minor axes of the ellipse are aligned with the coordinate axis, then the predictor variables of the data have to be ortogonal.

2. a. We drew the l_1 ball in 2 dimensions ($\{(x_1, x_2) : |x_1| + |x_2| \leq k\}$) in class, and showed that it has 4 corners. State and argue how many corners the l_1 ball in 3 dimensions has. Describe the regions of the boundary that give you "sparse" (not all $\widehat{\beta}_i \neq 0$) solutions.

The regions of the boundary that give you sparse solutions are the 6 corners $(k, 0, 0), (-k, 0, 0), (0, k, 0), (0, -k, 0), (0, 0, k), (0, 0, -k)$.

b. Compute the ratio of the radius of the inscribed l_2 ball to the circumscribed l_2 ball of the l_1-unit ball ($k = 1$) as a function of the dimension d and show that it is heading to 0 as $d \to \infty$. (Hint: What is the value closest to the origin in Euclidean distance. Check the 2-d plot to infer the general solution). Note that the corners of the d-ball, which give the sparsest solution, live on the circumscribed ball, while the non-sparse solution (all d $\widehat{\beta}$ are non-zero) lives on a hyper-plane that touches the inscribed ball.

Here's the picture in 2-d.

```
plot(c(-1,1), c(-1,1), t='n', asp=1)
lines(c(-1,0,1,0,-1),c(0,1,0,-1,0))
a <- seq(0,2*pi,.01)
r1 <- 1
r2 <- sqrt(2)/2
```

```
lines(r1*cos(a), r1*sin(a), col='red')
lines(r2*cos(a), r2*sin(a), col='blue')
```

The ratio of the radius of the inscribed l_2 ball to the circumscribed l_2 ball of the l_1-unit ball as a function of the dimension d is $\frac{1}{\sqrt{d}}$, which is going to 0 as $d \to \infty$.

3. Write a function that computes a ridge regression and returns the fitted values. Fitted values will be of the form $X\hat{B}$ where $\hat{B} = (X^T X + \lambda I)^{-1} X^T Y$, where X is a matrix who's columns are the basis vectors of the subspace we are projecting onto, i.e. a vector of all ones, and the data vectors.

The function will take in a vector of the response variable y, a matrix of predictors (without the intercept), and a value of λ.

Here's ordinary least squares to see how to work with matricies in R.

```
set.seed(47)
n <- 10
x <- runif(n)
y <- 3 + 5 * x + rnorm(n, 0, 1)
plot(x,y)
a <- rep(1,n)
X <- cbind(a,x) \#create the design matrix
beta <- solve(t(X) %*% X) %*% t(X) %*% y \#matrix multiplication in R and transpose of a matri
abline(beta, col='red')
```

The scikit-learn Python machine learning library provides an implementation of the Ridge Regression algorithm via the Ridge class. The lambda term can be configured via the "alpha" argument.

Given a response variable y, a matrix of predictors (without the intercept) X, and a value of λ, the following python code will compute a ridge regress, return the fitted values, and will even evaluate the model!

```
\# evaluate an ridge regression model on the dataset
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import Ridge
\# Here we are defining out ridge regression model
model = Ridge(alpha=1.0)
\# Here we are defining a model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
\# Here we are evaluating the model
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)
\# Here we force the scores to be positive and print
scores = np.absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

4. Using the *mtcars* data set as we did in Lecture 8 (removing mpg and qsec to form the matrix of predictors), build a model for fuel efficiency using a combination of feature engineering (non-linear transformations of existing variables) and the LASSO. See the notes and the vignette linked therein for using the glmnet package.

```
data(mtcars)
y <- mtcars$mpg
x <- as.matrix(mtcars[,-c(1,7)])
x <- cbind(x, x[,3]^2) \#add hp^2 to the data set.
x <- cbind(x, x[,3] * x[,4]) \#add hp * drat to the data set
```

Note: this seems clunky, but with simple features, we are likely to have a model that allows some interpretability (as opposed to adding infinitely many features, which we will do).

[5]:
```
import pandas as pd
from sklearn import linear_model

clf = linear_model.Lasso(alpha = 0.5)

# Import CSV mtcars
mtcars = pd.read_csv('https://gist.githubusercontent.com/ZeccaLehn/
 ↪4e06d2575eb9589dbe8c365d61cb056c/raw/
 ↪64f1660f38ef523b2a1a13be77b002b98665cdfe/mtcars.csv')
# Edit element of column header
mtcars.rename(columns={'Unnamed: 0':'brand'}, inplace=True)
y = mtcars['mpg']
X = mtcars.drop(columns=['brand','mpg', 'qsec'])
```

[4]:
```
clf.fit(X, y)
print(f'The coefficients of our model are {clf.coef_}')
print(f'The intercept of our model is {clf.intercept_}')
```

```
The coefficients of our model are [-0.13286461 -0.02288867 -0.01944688  0.
-0.99519437  0.
  0.          0.         -0.21000753]
The intercept of our model is 32.83869012090861
```