HW07: K-NN and Logistic Regression

Stat 154, Fall 2019

Problem 1

Consider a regression model with one response and one predictor: $y_i = f(x_i) + \epsilon_i$, and assumme that ϵ_i is a zero-mean noise term with variance σ^2. Likewise suppose that $f(x)$ is estimated with $\hat{f}(x)$. As you know, the bias-variance decomposition of the *expected test MSE* of $\hat{f}(x)$ becomes:

$$\mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}^{(\mathcal{D})}(x_o) - y_o\right)^2\right] = \text{bias}^2 + \text{var} + \sigma^2$$

where x_0 is a test point (out-of-sample), $\mathcal{D}$ is a training set, and the expectation is taken over all possible training sets.

If we denote $\hat{f}_K(x)$ the predicted value using K-Nearest-Neighbors, **show that the (theoretical) bias-variance decomposition for K-NN is**:

$$\mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}_K^{(\mathcal{D})}(x_o) - y_o\right)^2\right] = \underbrace{\left(f(x_0) - \mathbb{E}\left[\hat{f}_K(x_0)\right]\right)^2}_{\text{bias}^2} + \underbrace{\frac{\sigma^2}{k}}_{\text{variance}} + \underbrace{\sigma^2}_{\text{noise}}$$

Hint: recall that $\hat{f}_K(x)$

$$\hat{f}_K(x) = \frac{1}{K}\sum_{i=1}^{K} y_i; \qquad \text{for } i \in N_k$$

Problem 2

In lab-07 we discussed K-NN without paying any attention to the relative distance of the K nearest examples to the query point x_0. In other words, we let the K neighbors have equal influence on predictions irrespective of their relative distance from the query point. An alternative approach is to use arbitrarily large values of K (if not the entire training set) with more importance given to cases closest to the query point. This is achieved using so-called **distance weighting**.

Since K-NN predictions are based on the intuitive assumption that objects close in distance are potentially similar, it makes sense to differentiate between the K nearest neighbors when making predictions, i.e. let the closest points among the K nearest neighbors have a larger influence in affecting the outcome of the query point. This can be achieved by introducing a

set of weights $w_i(x_0)$, one for each nearest neighbor, defined by the relative closeness of each neighbor x_i with respect to the query point x_0. Thus:

$$w_i(x_0) = \text{weight}(x_i, x_0) = \frac{exp\{-d(x_i, x_0)\}}{\sum_{i=1}^{K} exp\{-d(x_i, x_0)\}}$$

where $d(x_i, x_0)$ is the distance between the query point x_0 and the i-th case x_i of the training set. The weights $w_i(x_0)$ defined in this manner above will satisfy:

$$\sum_{i=1}^{K} w_i(x_0) = 1$$

Thus, for regression problems we have:

$$\hat{y}_0 = \sum_{i=1}^{K} w_i(x_0) y_i$$

that is, the prediction $\hat{y}_0$ is a weighted average of the y_i-values from the K nearest neighbors x_i to the query point x_0.

K-NN with Distance Weighting

Write code to implement K-NN with distance weighting.

Implement a function `kNNW(x, y, z, k, p)` that finds the set of indices $\{i_1, ..., i_k\}$ of the k nearest points to $x_0 = z$, and returns $\hat{y}_0 = \sum_{i=1}^{K} w_i(z) y_i$. The arguments of the function are:

- `x` input vector
- `y` response vector
- `z` query point (or vector of query points)
- `k` number of neighbors
- `p` value for minkowski distance

The distance measure to be used is based on Minkowski distance:

$$d(i, l) = (|x_i - x_l|^p)^{1/p}$$

A value of $p = 1$ corresponds to the so-called Manhattan distance, which is what you used in lab-07. A value of $p = 2$ corresponds to the Euclidean distance.

- Use different values for $K = 10, 30, 50, 70, 100$
- Use values $p = 1$ and $p = 2$ for the type of distances

- Use the synthetic data provided in lab-07
- Plot the $kNNW$ function for all K values (see above) over a reasonable range of z values
- Here's how you should be able to invoke `kNNW()`

```
# synthetic data
set.seed(12345)
x <- runif(-2, 2, n = 100)
f <- function(u) {
  return(sin(pi*u) + u^2)
}
y <- f(x) + rnorm(length(x), sd = 0.2)

# one query point, k = 10 neighbors, manhattan distance
yhat = kNNW(x, y, z = 0, k = 10, p = 1)

# one query point, k = 50 neighbors, manhattan distance
yhat = kNNW(x, y, z = 0, k = 10, p = 1)

# various query points, k = 50 neighbors, manhattan distance
yhat = kNNW(x, y, z = c(-0.5, 0, 0.5), k = 10, p = 1)
```

Problem 3

Refer to the description of the Algorithm 7.1 "Local Regression" described in *ISL*, page 282—screenshot of the pseudo-code shown below. By the way, this is just one of several versions of local regression algortihms.

Algorithm 7.1 *Local Regression At $X = x_0$*

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.
3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

As you can tell, local regression involves fitting a **weighted least squares regression**. To simplify notation, let $w_i = K(x_i, x_0)$, and consider the Weighted Mean Squared Error (WMSE):

$$\text{WMSE} = \frac{1}{n}\sum_{i=1}^{n} w_i(b_0 + b_1 x_i - y_i)^2$$

Show that, for a given x_0, the vector of coefficients $\mathbf{b} = (b_0, b_1)$ that minimizes WMSE is given by:

$$\mathbf{b} = (\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{y}$$

where:

- $\mathbf{X}$ is an $n \times 2$ matrix with a constant column of ones, and another column for the predictor X.
- $\mathbf{W}$ is a diagonal $(n \times n)$ matrix with weights w_i on the diagonal.

Problem 4

As we saw in lecture, logistic regression involves modeling the conditional probability $P(y_i|\mathbf{x_i})$ as follows:

$$P(y_i|\mathbf{x}_i) = \begin{cases} h(\mathbf{x_i}) & \text{for} \quad y_i = 1 \\ \\ 1 - h(\mathbf{x_i}) & \text{for} \quad y_i = 0 \end{cases}$$

where the probability function $h(\mathbf{x_i})$ takes the form of the logistic function:

$$\text{logistic function:} \qquad \phi(s) = \frac{e^s}{1 + e^s}$$

In other words, the logistic model involves fitting the conditional probabilities $P(y_i|\mathbf{x}_i)$ by using logistic transformations, $\phi()$, of the linear combination of predictors $\mathbf{b}^\mathsf{T}\mathbf{x_i}$:

$$P(y_i|\mathbf{x}_i) = \begin{cases} h(\mathbf{x_i}) = \phi(\mathbf{b}^\mathsf{T}\mathbf{x_i}) & \text{for} \quad y_i = 1 \\ \\ 1 - h(\mathbf{x_i}) = 1 - \phi(\mathbf{b}^\mathsf{T}\mathbf{x_i}) & \text{for} \quad y_i = 0 \end{cases}$$

Note that $\mathbf{x_i}$ is a row-vector (for the i-th individual). The vector of regression coefficients $\mathbf{b}$ is obtained by Maximum Likelihood, where the likelihood $L(\mathbf{b})$ is:

$$L(\mathbf{b}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) = \prod_{i=1}^{n} h(\mathbf{x_i})^{y_i}(1 - h(\mathbf{x_i}))^{1-y_i}$$

Show that the gradient of the log-likelihood $l(\mathbf{b})$ is:

$$\nabla l(\mathbf{b}) = \sum_{i=1}^{n} \Big(y_i - \phi(\mathbf{b}^\mathsf{T}\mathbf{x_i})\Big)\, \mathbf{x_i}$$

Problem 5

As we also saw in lecture, logistic regression can be approached by encoding the response values with $y_i = \pm 1$ as follows:

$$P(y_i|\mathbf{x}_i) = \begin{cases} h(\mathbf{x_i}) & \text{for} \quad y_i = +1 \\ \\ 1 - h(\mathbf{x_i}) & \text{for} \quad y_i = -1 \end{cases}$$

where the probability function $h(\mathbf{x_i})$ takes the form of the logistic function:

$$\text{logistic function:} \qquad \phi(s) = \frac{e^s}{1 + e^s}$$

As stated in the previous problem, the logistic model involves fitting the conditional probabilities $P(y_i|\mathbf{x}_i)$ by using logistic transformations, $\phi()$, of the linear combination of predictors $\mathbf{b}^\mathsf{T}\mathbf{x_i}$:

$$P(y_i|\mathbf{x}_i) = \begin{cases} h(\mathbf{x_i}) = \phi(\mathbf{b}^\mathsf{T}\mathbf{x_i}) & \text{for} \quad y_i = +1 \\ \\ 1 - h(\mathbf{x_i}) = 1 - \phi(\mathbf{b}^\mathsf{T}\mathbf{x_i}) & \text{for} \quad y_i = -1 \end{cases}$$

Notice that both cases above can be merged into a single term: $\phi(y_i\mathbf{b}^\mathsf{T}\mathbf{x_i})$. This allows you to write the Likelihood $L(\mathbf{b})$ as:

$$L(\mathbf{b}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) = \prod_{i=1}^{n} \phi(y_i\mathbf{b}^\mathsf{T}\mathbf{x_i})$$

In this case, maximizing the likelihood $L(\mathbf{b})$ is equivalent to minimizing the "cross-entropy error" $E_{in}(\mathbf{b})$:

$$E_{in}(\mathbf{b}) = \frac{1}{n}\sum_{i=1}^{n} log\left(1 + e^{-y_i\mathbf{b}^\mathsf{T}\mathbf{x_i}}\right)$$

Obtain the derivation of the cross-entropy error and show that its gradient is:

$$\nabla E_{in}(\mathbf{b}) = -\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i\mathbf{x_i}}{1 + e^{y_i\mathbf{b}^\mathsf{T}\mathbf{x_i}}}\right)$$

Problem 6

This is problem 9 in Chapter 7, *ISL*. This question uses the variables `dis` (the weighted mean of distances to five Boston employment centers) and `nox` (nitrogen oxides concen- tration in parts per 10 million) from the `Boston` data. We will treat `dis` as the predictor and `nox` as the response.

a) Use the `poly()` function to fit a cubic polynomial regression to predict `nox` using `dis`. Report the regression output, and plot the resulting data and polynomial fits.

b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

c) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

d) Use the `bs()` function to fit a regression spline to predict `nox` using `dis`. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

f) Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.