Nearest Neighbor Autocovariates: With R Package

A. Ian McLeod, M. Shahidul Islam
Department of Statistical and Actuarial Sciences
The University of Western Ontario
London, Ontario, Canada

Abstract

A fast R (R Development Core Team 2010) algorithm is given for computing the nearest neighbor autocovariates (Holmes and Adams 2003). These autocovariates may be used to select k for kNN classifiers using a maximum pseudolikelihood estimate for k. More general logistic regression and multinomial regression models may be developed using nearest neighbor autocovariates. Computation of nearest neighbor autocovariates and pseudolikelihood estimation of k are implemented in our **nnc** package (McLeod and Islam 2010).

Keywords: classification, k-nearest neighbors, logistic regression, multinomial regression, nearest neighbor autocovariates, nonlinearity.

1. Introduction

The k-nearest neighbor (kNN) classification algorithm is often considered among the most important methods in data mining (Wu and Kumar 2009; Hastie, Tibshirani, and Friedman 2009). The nearest neighbor autocovariates defined by Holmes and Adams (2003) allows one to determine the parameter k, the number of nearest neighbors in kNN, using maximum pseudolikelihood estimation. Furthermore, Holmes and Adams (2003) showed that by including several nearest neighbor autocovariates in a logistic or multilogistic model followed by a best subset or stepwise selection, a better model may be obtained that includes non-linear covariates along with the usual inputs. Empirical evidence presented in Holmes and Adams (2003, Section 3) suggests that this method works better than kNN by itself. Moreover, it was also found that the pseudolikelihood method tends to choose a larger value k for kNN

than cross-validation methods[1]. Even if kNN is used by itself, the pseudolikelihood method for choosing k seems to work better than leave-one-out cross-validation (Holmes and Adams 2003).

2. Nearest neighbor autocovariate for two classes

2.1. Pseudolikelihood function

Initially we consider the problem with two output[2] classes, denoted by $\{-1, 1\}$, and p inputs. Assume that there are n data values $\{y_i, x_i\}$, where $x_i \in \Re^p$, $y_i \in \{-1, 1\}$, $i = 1, \ldots, n$. The nearest neighbor autocovariate corresponding to the ith data value is defined by

$$z_i^{(k)} = \frac{1}{k} \sum_{\substack{k \\ \ell \sim i}} \{\mathcal{I}(y_\ell = 1) - \mathcal{I}(y_\ell = -1)\}, \tag{1}$$

where the summation is over the k-nearest neighbors of x_i in the set x_1, x_2, …, x_{i-1}, x_{i+1}, …, x_n and $\mathcal{I}$ is the indicator function. The autocovariate $z_i^{(k)} \in [-1, 1]$ represents the relative share of class 1's with respect to class -1's in the k nearest neighbors of x_i. Values of $z_i^{(k)}$ close to 1, indicate that 1's are more abundant in the kth order neighborhood of x_i while values of $z_i^{(k)}$ close to -1, indicate that -1's are more abundant.

The logistic regression model containing the covariates $z_i^{(k)}$ can be written as

$$\begin{aligned} \eta_i &= \Pr(y_i = 1) \\ &= \frac{\exp(\alpha_k z_i^{(k)})}{1 + \exp(\alpha_k z_i^{(k)})}. \end{aligned} \tag{2}$$

For fixed k, the pseudolikelihood function for α_k is,

$$L_k(\alpha_k) = \prod_{i=1}^{n} \eta_i^{u_i} (1 - \eta_i)^{1 - u_i}, \tag{3}$$

where $u_i = (z_i^{(k)} + 1)/2$. It is called a pseudolikelihood because the input, $z_i^{(k)}$, has been determined using the output, y_i. For fixed k, the maximum pseudolikelihood estimate, $\hat{\alpha}_k$, using the `glm` function for logistic regression is obtained. Holmes and Adams (2003) note that the predictions of the logistic regression defined in eqn (2) with $\alpha_k = \hat{\alpha}_k$ are identical to the predictions obtained from kNN with neighborhood size k.

Holmes and Adams (2003) suggest maximizing the profile pseudolikelihood,

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, L_k(\hat{\alpha}_k), \tag{4}$$

to obtain the optimal k, $\hat{k}$.

[1] Larger k, corresponds to a more parsimonious model since there are few effective parameters

[2] Following Hastie *et al.* (2009) the response/explanatory variables are also called output/input variables respectively.

2.2. R algorithm

The nearest neighbor autocovariate is easily computed using the `knn.cv` function in the standard R package **class**. The `knn.cv` function implements leave-one-out cross-validation and, as a by-product, we may obtain $\hat{y}_i$, the leave-one-out prediction for the ith output and p_i, the proportion of votes for the winning class. Then $(p_i k - (k - p_i k))/k$ is the required autocovariate. The following script implements this computation for `synth.tr`, the training sample in the synthetic mixture dataset in **MASS**.

```
R> library("MASS")
R> library("class")
R> X <- synth.tr[, 1:2]
R> y <- 2 * synth.tr[, 3] - 1
R> k <- 10
R> ans <- knn.cv(train = X, cl = as.factor(y), k = k, prob = TRUE)
R> pr <- attr(ans, "prob")
R> yfit <- as.numeric(as.character(ans))
R> NumberOfVotesForWinner <- pr * k
R> NumberOfVotesForLoser <- k - NumberOfVotesForWinner
R> ConsensusProportion <- (NumberOfVotesForWinner - NumberOfVotesForLoser)/k
R> z <- yfit * ConsensusProportion
R> str(z)

 num [1:250] -0.8 -1 -1 -0.8 0.6 0 -1 -1 -0.8 -0.6 ...
```

The variable `z` displayed is the nearest neighbor autocovariate when $k = 10$. In general the nearest neighbor autocovariate for any k may be computed using the function `nnc` in our package **nnc**.

2.3. Application to synthetic mixture dataset

When using the `glm` function, it is more convenient to work with the deviance. We use the `nnc` function to plot the deviance for various k.

```
R> library("nnc")
R> X <- synth.tr[, 1:2]
R> y <- synth.tr[, 3]
R> KMAX <- 100
R> L <- numeric(KMAX)
R> for (k in 1:KMAX) {
+     z <- nnc(X = X, Y = y, k = k)
+     L[k] <- deviance(glm.fit(x = z, y = y, family = binomial(link = "logit")))
+ }
R> khat <- which.min(L)
R> plot(L, xlab = "k", ylab = "deviance")
R> points(khat, L[khat], col = "blue", pch = 16, cex = 1.5)
R> title(sub = bquote(hat(k) == .(khat)))
```

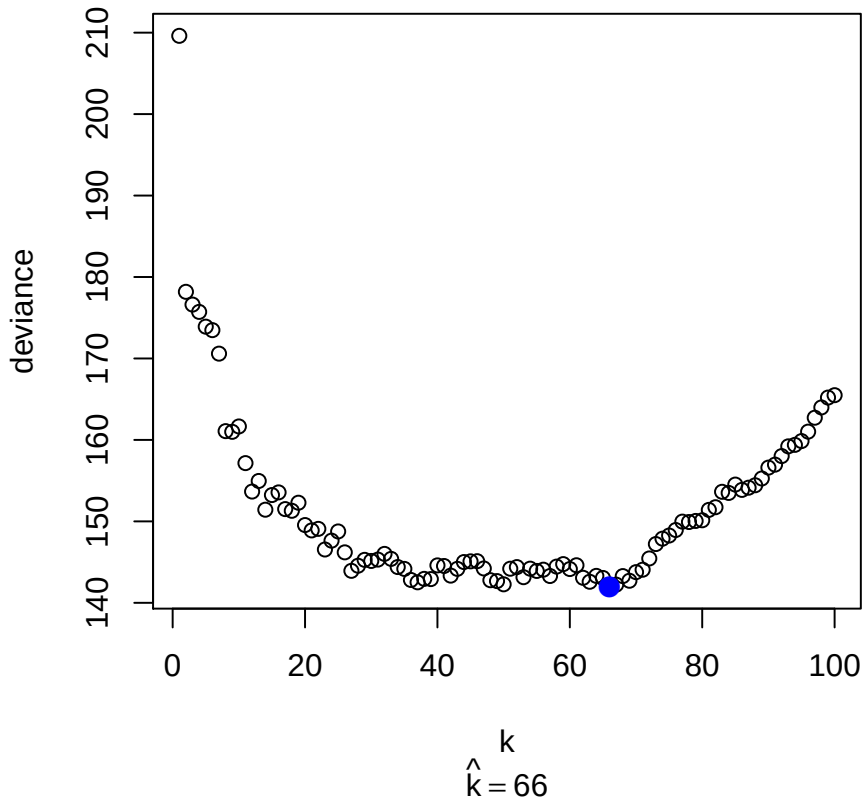

Figure 1: Deviance for pseudo-logistic regression and mle for k.

The result $\hat{k} = 66$ agrees with computation reported by Holmes and Adams (2003) for this dataset.

3. Nearest neighbor autocovariate for multiclass case

In the multiclass case we have Q classes, $\mathfrak{C}_1, \mathfrak{C}_2, \ldots, \mathfrak{C}_Q$ and the multiclass autocovariate for class $j = 2, \ldots, Q$, is defined by,

$$z_{i,j}^{(k)} = \frac{1}{k} \sum_{\substack{k \\ \ell \sim i}} \{\mathcal{I}(y_\ell = \mathfrak{C}_j) - \mathcal{I}(y_\ell = \mathfrak{C}_1)\}, \tag{5}$$

where $i = 1, \ldots, n$ and $z_{i,j}^{(k)} \in [-1, 1]$. So the autocovariate, $z_{i,j}^{(k)}$, has the interpretation as the relative proportion of $\mathfrak{C}_j$ to $\mathfrak{C}_1$ in a neighborhood of size k about the ith data point. The k nearest neighbors excluding the ith data point itself are used. Positive values of $z_{i,j}^{(k)}$ indicate $\mathfrak{C}_j$ is more frequent than $\mathfrak{C}_1$. Similarly, negative values of $z_{i,j}^{(k)}$ indicate $\mathfrak{C}_1$ is more numerous than $\mathfrak{C}_j$.

The multiclass autocovariate $z_{i,j}^{(k)}$ may be computed conveniently in R using two passes of the algorithm for computing autocovariates with two classes. Two new response or output vectors are defined. For both output vectors, the class values corresponding to $\mathfrak{C}_1$ are set to -1 and for the jth autocovariate, the values corresponding to $\mathfrak{C}_j$ are set to 1. In the first output vector, values corresponding to all other classes set to -1 and in the second output vector these values are set to 1. The `nnc` algorithm for binary classes using each of these output vectors as response variables is used to obtain two autocovariate vectors. The average of these two vectors produced by `nnc` is the required jth multiclass autocovariate, $z_{i,j}^{(k)}, i = 1, \ldots, n$. Doing this for $j = 2, \ldots, Q$, produces all $Q - 1$ autocovariates.

The code snippet below illustrates how this works for the built-in dataset `iris`.

```
R> k <- 17
R> X <- iris[, 1:4]
R> Y <- iris[, 5]
R> n <- length(Y)
R> y <- numeric(n)
R> classes <- unique(Y)
R> Q <- length(classes)
R> ind1 <- Y == classes[1]
R> y[ind1] <- -1
R> z <- matrix(numeric(n * (Q - 1)), nrow = n)
R> for (j in 2:Q) {
+     indk <- Y == classes[j]
+     indOther <- !(ind1 | indk)
+     y[indk] <- 1
+     y[indOther] <- -1
+     zA <- nnc(X = X, Y = y, k = k)
+     y[indOther] <- 1
+     zB <- nnc(X, y, k)
+     z[, j - 1] <- (zA + zB)/2
+ }
R> tail(z, 5)
```

```
             [,1]      [,2]
[146,] 0.05882353 0.9411765
[147,] 0.23529412 0.7647059
[148,] 0.05555556 0.9444444
[149,] 0.00000000 1.0000000
[150,] 0.17647059 0.8235294
```

The last 5 values for the nearest neighbor autocovariate corresponding to $z_{i,j}^{(k)}$, $i = 146, \ldots, 150$, $j = 2, 3$ and $k = 17$ are displayed.

As a further check, we have implemented the nearest neighbor autocovariate computation for the multiclass case in Mathematica (Wolfram Research, Inc 2008). This algorithm uses the definition in eqn (5) directly so the resulting algorithmic details are completely different but the numerical results agree with our R function `nnc`. This Mathematica package is included along with sample output in the documentation with our R package **nnc**.

Setting,

$$\begin{aligned} \eta_{i,j} &= \Pr(y_i = \mathfrak{C}_j) \\ &= \frac{\exp(\alpha_j z_{i,j}^{(k)})}{\sum_{j=1}^{Q} \exp(\alpha_j z_{i,j}^{(k)})}, \end{aligned} \tag{6}$$

where we set $\alpha_1 = 0$ and $z_{i,1} = 0, i = 1, \ldots, n$. We see that the model has the form of a multinomial regression (Hastie *et al.* 2009). In R, for fixed k, maximum pseudolikelihood estimates for $\alpha_2, \ldots, \alpha_Q$ may be obtained using the `multinom` function in the package **nnet**.[3] Using the pseudo maximum likelihood estimates for $\alpha_2, \ldots, \alpha_Q$ the model in eqn (6) gives identical predictions as the kNN algorithm with parameter k (Holmes and Adams 2003).

Since `multinom` prints convergence information and since we need to call this function many times, it is useful to use `sink`, as in the code snippet below to omit unnecessary output.

```
R> library("nnet")
R> kmax <- 50
R> d <- numeric(kmax)
R> for (k in 1:kmax) {
+     z <- nnc(X = X, Y = Y, k = k)
+     sink("junk.txt")
+     d[k] <- multinom(Y ~ z)$deviance
+     sink()
+ }
R> unlink("junk.txt")
R> khat <- which.min(d)
R> plot(d, xlab = "k", ylab = "deviance")
R> points(khat, d[khat], col = "blue", pch = 16, cex = 1.5)
R> title(sub = bquote(hat(k) == .(khat)))
```

[3] The function `multinom` seems to have better convergence properties in the non-penalized case than `glmnet`. The non-penalized case is obtained by setting the argument `lambda=0` in `glmnet`.

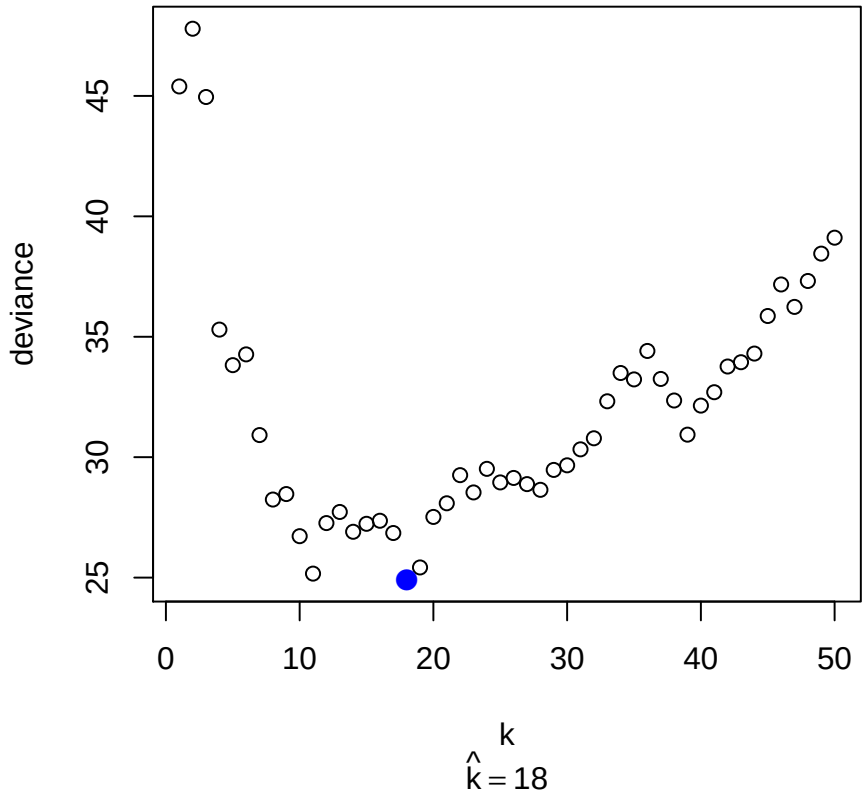

Figure 2: Estimate for the k parameter in kNN for iris dataset. Fisher linear discriminant analysis uses 5 parameters for this problem, the kNN-pseudolikelihood method produces $\hat{k} = 18$ which is equivalent to about $\nu \approx 8$ effective parameters.

Our function `nnc` handles the multiclass case and we have also implemented a function `khat` that uses the profile pseudolikelihood to estimate k. When $Q = 2$, the profile pseudolikelihood is obtained using `glm` and for $Q > 2$, `multinom` is used.

In the vignette accompanying our R package **nnc**, we provide a script for a cross-validation comparison of Fisher linear discriminant analysis and kNN for the iris dataset. The delete-d cross-validation method with 10^4 randomly chosen test samples was used. The computations took about 27 minutes on a Mac Pro running eight threads. The observed misclassification rates are 0.18088 and 0.01978 for kNN and Fisher linear discriminant analysis respectively. It was not surprising the Fisher linear discriminant analysis outperformed kNN for this dataset since an exploratory analysis of the iris dataset does not suggest nonlinearity is an issue. More details are given in the vignette.

4. Application to logistic regression

Holmes and Adams (2003) suggest including nearest neighbor autocovariates as inputs in logistic regression models to allow for nonlinear effects. By using these autocovariates corresponding to various size neighborhoods, we can incorporate a greater range of nonlinear effects than simply using one neighborhood.

To illustrate this method, we use the kyphosis data that is included in the **rpart** library. The output is a binary indicator of the presence or absence of the condition after treatment and there are three inputs, `Age`, `Number` and `Start`. First we estimate k for pure kNN and we find $\hat{k} = 8$.

```
R> library("rpart")
R> X <- kyphosis[, 2:4]
R> Y <- kyphosis[, 1]
R> kHat <- khat(X = X, Y = Y, plot = FALSE)
R> kHat

[1] 8
```

This result suggests using the first eight autocovariates as additional inputs. We fit with a linear logistic regression followed by stagewise model selection using the AIC criterion.

```
R> Z <- sapply(1:8, function(k) nnc(X = X, Y = Y, k = k))
R> dimnames(Z)[[2]] <- paste("z", 1:8, sep = "")
R> kdf <- data.frame(Z, kyphosis)
R> ans <- glm(Kyphosis ~ ., family = binomial(link = "logit"), data = kdf)
R> step(ans, trace = 0)$coefficients

 (Intercept)           z1           z5           z6           z7           z8
-4.12166880  -2.35111478  11.07409030 -11.07094427  -6.80680382   9.85025556
         Age       Number        Start
 0.02371589   0.90752239  -0.39572491
```

Instead of using the built-in R function `step` for stagewise optimization, an exhaustive best subset approach using `bestglm` (McLeod and Xu 2009) is available when the number of inputs is not too large.

```
R> library("bestglm")
R> Xy <- data.frame(X, Z, Kyphosis = as.numeric(Y) - 1)
R> ans <- bestglm(Xy = Xy, family = binomial(link = "logit"), IC = "AIC")

Morgan-Tatar search since family is non-gaussian.

R> ans$BestModel$coefficients

 (Intercept)           Age        Number         Start           z1           z5
-4.12166880   0.02371589   0.90752239  -0.39572491  -2.35111478  11.07409030
          z6           z7           z8
-11.07094427  -6.80680382   9.85025556
```

In this case the exhaustive subset method produced the same answer.

It might be even better to use a regularization approach for logistic and multinomial regression as is implemented in the R package **glmnet** (Friedman, Hastie, and Tibshirani 2010b,a). This method can not only handle a large number of inputs but it also produces even better predictions. Recent work with non-concave penalty methods for regularization is also very promising (Fan and Li 2001; Zhang 2010).

Nearest neighbor autocovariates may easily be incorporated into multinomial regression as well as many types of classification algorithms (Islam 2008). In particular, Islam (2008) showed that nearest neighbor autocovariates could be used to improve predictions with regularized discriminant analysis in microarray data (Guo, Hastie, and Tibshirani 2007).

References

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**, 1348–1360.

Friedman J, Hastie T, Tibshirani R (2010a). *glmnet: Lasso and elastic-net regularized generalized linear models*. URL http://CRAN.R-project.org/package=glmnet.

Friedman J, Hastie T, Tibshirani R (2010b). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22.

Guo Y, Hastie T, Tibshirani R (2007). "Regularized Linear Discriminant Analysis and its Application in Microarrays." *Biostatistics*, **8**(1), 86–100.

Hastie T, Tibshirani R, Friedman JH (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. 2nd edition. Springer-Verlag, New York.

Holmes CC, Adams NM (2003). "Likelihood Inference in Nearest-Neighbour Classification Models." *Biometrika*, **90**(1), 99–112.

Islam MS (2008). *Periodicity, Change Detection and Prediction in Microarrays*. Ph.D. thesis, The University of Western Ontario.

McLeod AI, Islam MS (2010). *nnc: Computes nearest neighbor autocovariates*. URL http://CRAN.R-project.org/package=nnc.

McLeod AI, Xu C (2009). *bestglm: Best Subset GLM*. URL http://CRAN.R-project.org/package=bestglm.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Wolfram Research, Inc (2008). *Mathematica Edition: Version 7.0*. Wolfram Research, Inc., Champaign, Illinois. URL http://www.wolfram.com/.

Wu X, Kumar V (2009). *The Top Ten Algorithms in Data Mining*. CRC/Chapman-Hall, Boca Raton.