Experiment Notes on RLCP

Latest Update: 2024 年 10 月 9 日

摘要

An implementation for localized conformal prediction.

- Author: Rohan Hore, Rina Foygel Barber[HB24].
- RLCP stands for randomly-localized conformal prediction.
- Localized CP: Guan(2023, biometrika)
- Weighted CP: Tibshirani et al. (2019)

1 Simulations

For the following experiments we replicate, our aim will be to achieve 95% coverage, i.e., we choose $\alpha = 0.05$. The feature spce is given bt $\mathcal{X} = \mathbb{R}^d$. and we can choose a kernel, such as a radial basis keneral.

1.1 Univariate setting

We begin with experiments in a univariate setting, i.e., $d = 1$. We consider the following two data generating distributions.

- Setting 1: $X \sim \mathcal{N}(0,1), Y \Big| X \sim \mathcal{N}\left(\frac{X}{2}, |\sin(X)|\right)$
- Setting 2: $X \sim \mathcal{N}(0,1), Y \Big| X \sim \mathcal{N}\left(\frac{X}{2}, \frac{4}{3}\phi\left(\frac{2X}{3}\right)\right)$, where $\phi(\cdot)$ is the density of a standard Gaussian.

We can use `suppressPackageStartupMessages` to suppress the package startup messages while library the packages.

```r
suppressPackageStartupMessages(library(doParallel))
suppressPackageStartupMessages(library(MASS))
suppressPackageStartupMessages(library(mvtnorm))
suppressPackageStartupMessages(library(ggplot2))
```

The following general code generates the data for the settings.

```r
#---------------------------------------------------
#---------------simulation settings-----------------
#---------------------------------------------------
simulation=function(n,d,setting){
  X=rmvnorm(n,mean=rep(0,d),sigma=diag(d))

  ##setting 1
  if(setting==1){
    Y=0.5*apply(X,1,mean)+apply(abs(sin(X)),1,sum)*rnorm(n,0,1)
  }

  ##setting 2
  if(setting==2){
    Y=0.5*apply(X,1,mean)+apply(2*dnorm(X,0,1.5),1,sum)*rnorm(n,0,1)
  }

  ##setting 3 : P_X uniform on cube
  if(setting==3){
    X=matrix(runif(n*d,-3,3),nrow=n,ncol=d)
    Y=0.5*apply(X,1,mean)+apply(abs(sin(X)),1,sum)*rnorm(n,0,1)
  }

  data=as.data.frame(cbind(Y,X))
  colnames(data)=c("Y",paste0("X",1:d))
  return(data)
}
```

The two settings share the same feature distribution, $P_X = \mathcal{N}(0,1)$. The difference lies in the conditional distribution of Y, $P_{Y|X}$.

In setting 2, the response has more variance for values of X near the center of the distribution. In contrast, the variance has more variance for values of X lying away from the mean.

```r
#--------------------------------------------------
#-------Simulating Y|X on a grid of ---------------
#--------feature points in (-3,3)------------------
#--------------------------------------------------
conditional_simulation=function(n=100,setting){
  X=as.matrix(seq(-3,3,by=0.01))
  N=rnorm(length(X),0,1)

  ##setting 1
  if(setting==1){Y=0.5*apply(X,1,mean)+abs(sin(X))*N}
  ##setting 2
  if(setting==2){Y=0.5*apply(X,1,mean)+2*dnorm(X,0,1.5)*N}

  data=as.data.frame(cbind(Y,X));d=1
  colnames(data)=c("Y",paste0("X",1:d))
  return(data)
}
```

For both settings, our score function is given by $s(x,y)=|y-\hat{f}(x)|$, where $\hat{f}$ is a pretrained predictive model. For each setting, we implement the methods using localization kernel H at five different bandwidths, $h\in\{0.1,0.2,0.4,0.8,1.6\}$; the lowest value represents a highly local H, while the highest value leads to a kernel H that is essentially flat over the bulk of the feature distribution. The methods are run with sample size $n=2000$, and evaluated on 2000 test points. The entire experiment is repeated for 50 independent trials.

1.2 Generating $\tilde{X}$ for box kernel

Consider a new feature variable $\widetilde{X}_{n+1}$, generated as $\widetilde{X}_{n+1}\mid X_{n+1}\sim P_X\circ H\left(\cdot,X_{n+1}\right)$, where for any function $g:\mathcal{X}\rightarrow\mathbb{R}_{\geq 0}$ with $0<\mathbb{E}_{P_X}[g(X)]<\infty$, we define $P_X\circ g$ as the distribution P_X reweighted by g, i.e.,

$$\left(P_X\circ g\right)(A)=\frac{\int_A g(x)\mathrm{d}P_X(x)}{\int_{\mathcal{X}} g(x)\mathrm{d}P_X(x)}\text{ for all } A\subseteq\mathcal{X}$$

If the localization kernel H has a reasonably small bandwidth, then the reweighted distribution $P_X\circ\, H\left(\cdot,X_{n+1}\right)$ places most of its mass near X_{n+1}. As a result, we can interpret $\widetilde{X}_{n+1}$ as a sort of "synthetic sample" designed to be similar to the test feature X_{n+1}-in a setting where each data point is a patient, we are generating new patient data $\widetilde{X}_{n+1}$ for a patient that is similar (in feature space) to the test patient for whom we want to perform prediction.

The only difference between RLCP and baseLCP, then, is that the weights are given by the $\widetilde{w}_i$'s, computed with the kernel H centered at $\widetilde{X}_{n+1}$, rather than the original w_i 's, which were computed with H centered at X_{n+1}.

$$\text{calLCP}:\quad w_{i,j}=\frac{H\left(X_j,X_i\right)}{\sum_{k=1}^{n+1}H\left(X_k,X_i\right)},i,j\in[n+1]$$

$$\text{RLCP}:\quad \widetilde{w}_i=\frac{H\left(X_i,\widetilde{X}_{n+1}\right)}{\sum_{j=1}^{n+1}H\left(X_j,\widetilde{X}_{n+1}\right)},\quad i\in[n+1]$$

```
#-------------------------------------------------
#--------generating X_tilde for box kernel---------
#-------------------------------------------------
runifball=function(n,center,radius){
  d=length(center)
  data=matrix(0,nrow=n,ncol=d)

  U=runif(n,min=0,max=1)
```

```r
  Z=matrix(rnorm(n*d),nrow=n,ncol=d)
  for (i in 1:n){
    data[i,]=center+radius*U[i]^(1/d)*Z[i,]/sqrt(sum(Z[i,]^2))}
  return(data)
}
```

2 Base LCP

```r
#--------------------------------------------------
#---------------------baseLCP----------------------
#--------------------------------------------------
baseLCP=function(Xcalib,scores_calib,Xtest,scores_test,kernel,h,alpha){
  ntest=dim(Xtest)[1] ## number of test points
  d=dim(Xtest)[2] ## dimension of the feature space
  coverage=score_threshold=rep(0,ntest) ## initializing coverage and score_threshold

  #sorting with respect to the order of calibration scores.
  # scores_calib = s(x,y) = |y - \hat{f}(x)|
  Xcalib=as.matrix(Xcalib[order(scores_calib),])
  scores_calib=sort(scores_calib)

  #finding unique scores and the indices where each of these unique scores have been repeated.

  scores=c(scores_calib,Inf)
  indices=list();j=1;i=1
  scores_unique=vector()
  while(i<=length(scores)){
    scores_unique=c(scores_unique,scores[i])
    indices[[j]]=which(scores==scores[i])
    i=i+sum(scores==scores[i]);j=j+1
  }

  for(i in 1:ntest){
    xtest=Xtest[i,];test_score=scores_test[i]
    cov_data=rbind(Xcalib,xtest)

    #finding the weights and the score threshold
    if(kernel=="gaussian"){
      weights=dmvnorm(cov_data,mean=xtest,sigma=diag(d)*h^2)
      result=smoothed_weighted_quantile(scores_unique,alpha,weights,indices)
    }
    if(kernel=="box"){
      weights=apply(cov_data,1,FUN=function(x){(euclid_distance(x,xtest)<=h)+0})
      result=smoothed_weighted_quantile(scores_unique,alpha,weights,indices)
    }
    score_threshold[i]=result[1] #score_threshold
    closed=result[2]    #whether it's a closed interval

    #coverage
    coverage[i]=(test_score<score_threshold[i])+0
    if(closed==TRUE){coverage[i]=(test_score<=score_threshold[i])+0}
  }
  return(cbind(coverage,score_threshold))
}
```

This approach compute a quantile $\widehat{q}(x)$ that places more weight on the hold out points X_i that lie near x.

- We first choose some *localization kernel* given by a function $H : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$.
- For instance, for $\mathcal{X} = \mathbb{R}^d$, common choices include a Gaussian kernel with bandwidth h given by $H(x, x') = \exp\left(-\|x - x'\|^2/h^2\right)$.
- Or a box kernel with bandwidth h given by $H(x, x') = 1\{\|x - x'\| \leq h\}$.
- The local quantile is then computed as

$$\widehat{q}_{1-\alpha}(X_{n+1}) = \text{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n} w_i \delta_{s(X_i,Y_i)} + w_{n+1}\delta_\infty\right).$$

- δ_s is the point mass at s.
- weights are given by

$$w_i = \frac{H(X_i, X_{n+1})}{\sum_{j=1}^{n+1} H(H_j, X_{n+1})}, \quad i \in [n+1].$$

- The resulting prediction interval is defined as

$$\widehat{C}_n^{\text{baseLCP}}\left(X_{n+1}\right) = \left\{y \in \mathcal{Y} : s\left(X_{n+1}, y\right) \leq \widehat{q}_{1-\alpha}\left(X_{n+1}\right)\right\}$$

- We choose the trivial localization kernel $H(x, x') = 1$, namely, no localization, then we would have $\widehat{q}_{1-\alpha}(x)$ constant over x.

2.1 Smooted Weighted Quantile

```
#---------------------------------------------------
#----computing smoothed weighted quantile-----------
#---------------------------------------------------
##----here v consists of the unique scores, while indices contain the data-indices that
```

```r
##----has led to same score, w is the weight vector.
smoothed_weighted_quantile=function(v,alpha,w,indices){
  w=w/sum(w) ## normalizing the weights by definition
  U=runif(1,min=0,max=1)

  #finding the weights corresponding to each unique score.
  v_tilde=v ## unique scores
  w_tilde=rep(0,length(v))
  for(i in 1:length(v)){
    w_tilde[i]=sum(w[indices[[i]]]) ## summing the weights corresponding to each unique score.
  }

  #computing p-value at points in between the calibration scores.
  p_values=rep(0,length(v_tilde))
  for(i in 1:length(v_tilde)-1){
    p_values[i]=sum(w_tilde[i:(length(v_tilde)-1)])+U*(tail(w_tilde,1))
  }
  #computing p-value at a point higher than all calibration scores.
  p_values[length(v_tilde)]=U*(tail(w_tilde,1))

  #if pvalue is never greater than alpha, we output the empty set.
  if(sum((p_values>alpha))>0){
    id=max(which(p_values>alpha))
    #now we check, whether the prediction interval will be a closed interval or open.
    quantile=v_tilde[id]
    if(id<length(v_tilde)-1){closed=(sum(w_tilde[(id+1):(length(v_tilde)-1)])+U*(w_tilde[id]+ta
    if(id==length(v_tilde)){closed=FALSE}
    if(id==length(v_tilde)-1){closed=(U*(w_tilde[id]+tail(w_tilde,1))>alpha)}}
  else{quantile=-Inf;closed=FALSE}
  return(c(quantile,closed))
}
```

References

[HB24] Rohan Hore and Rina Foygel Barber. *Conformal prediction with local weights: randomization enables local guarantees*. 2024. arXiv: 2310.07850 [stat.ME]. URL: https://arxiv.org/abs/2310.07850.