Prediction Extrapolation

Charles Zheng

May 19, 2016

Run this code

Use Rstudio: `https://www.rstudio.com` . Go to `https://github.com/snarles/fmri/blob/master/extrapolation/simulation.Rmd` .

Source code

Generate data

Create synthetic data from Gaussian mixture model.

```
library(pracma)
p <- 10
sigma <- 1 # noise around cluster
k <- 20 # initial number of classes
K <- 50 # final number of classes
r1 <- 20 # number of training repeats
r2 <- 20 # number of test repeats
gen.data <- function(p, sigma, K, r1, r2) {
  mus <- randn(K, p) # cluster centroids
  Z1 <- rep(1:K, each = r1)
  Z2 <- rep(1:K, each = r2)
  Ytr <- mus[Z1, ] + sigma * randn(K * r1, p) # final training data
  Yte <- mus[Z2, ] + sigma * randn(K * r2, p) # final test data
  list(Ytr = Ytr, Yte = Yte, Z1 = Z1, Z2 = Z2)
}
synth_data <- gen.data(p, sigma, K, r1, r2)
```

Train models

Train Gaussian mixture model (equivalent to naive Bayes), quadratic discriminant analysis, multinomial logistic regression, ϵ-nearest neighbors, and single-layer neural network.

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.4
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:pracma':
##
##     expm, lu, tril, triu
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
library(MASS)
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 3.2.4
```

```
library(nnet)
epsilon_nn <- 0.1 # epsilon for epsilon-NN
## get log probs for first k classes
pred_submodel <- function(Ytr, Yte, Z1, Z2, k) {
  Ytr_sub <- Ytr[Z1 %in% 1:k, ] # subset training data
  Yte_sub <- Yte[Z2 %in% 1:k, ] # subset test data
  Z1_sub <- Z1[Z1 %in% 1:k]
  Z2_sub <- Z2[Z2 %in% 1:k]
  ## gmm
  mu_hat <- t(sapply(1:k, function(i) {
    colMeans(Ytr_sub[Z1_sub == i, ])
  }))
  dist_gmm <- pdist2(mu_hat, Yte_sub)
  pred_gmm <- -t(dist_gmm)
  ## QDA
  res_qda <- qda(x = Ytr_sub, grouping = Z1_sub)
  pred_qda <- predict(res_qda, Yte_sub)$posterior
  ## multinomial logistic
  res_glmnet <- glmnet(Ytr_sub, Z1_sub, family = "multinomial")
  pred_glmnet <- predict(res_glmnet, Yte_sub, s = 0)[, , 1]
  ## eps-NN
  df_train <- data.frame(Z = as.factor(Z1_sub), Y = Ytr_sub)
  df_test <- data.frame(Z = as.factor(Z2_sub * 0 + 1), Y = Yte_sub)
  res_enn <- kknn::kknn(Z ~ ., train = df_train, test = df_test,
                        k = floor(epsilon_nn * length(Z1_sub)))
  pred_enn <- res_enn$prob
  ## nnet
  res_nnet <- nnet(Z ~ ., data = df_train, size = 10, trace = FALSE)
  pred_nnet <- predict(res_nnet, df_test)
  list(pred_gmm = pred_gmm, pred_qda = pred_qda, pred_glmnet = pred_glmnet,
       pred_enn = pred_enn, pred_nnet = pred_nnet)
}

preds_sub <- pred_submodel(synth_data$Ytr, synth_data$Yte,
                           synth_data$Z1, synth_data$Z2, k)
preds_full <- pred_submodel(synth_data$Ytr, synth_data$Yte,
                            synth_data$Z1, synth_data$Z2, K)
```

Get Vij

Compute the statistics V_{ij} needed for prediction extrapolation, and compute sub/full accuracies.

```
get_vij <- function(pred, Z2) {
  rankconv <- t(apply(pred, 1, function(v) rank(v, ties.method = "random"))) - 1
  Z2_sub <- Z2[1:nrow(pred)]
  Vs <- rankconv[cbind(1:nrow(pred), Z2_sub)]
  Vs
}

V_subs <- lapply(preds_sub, get_vij, Z2 =synth_data$Z2)
lapply(preds_sub, function(v) table(get_vij(v, synth_data$Z2)))
```

```
## $pred_gmm
##
##  12  14  15  16  17  18  19
##   1   3   5  12  18  58 303
##
## $pred_qda
##
##   2   3   4   5   6   7   9  10  11  12  13  14  15  16  17  18  19
##   1   1   2   1   1   1   6   5   5   7  10  11  15  31  33  70 200
##
## $pred_glmnet
##
##   7   8   9  10  11  12  13  14  15  16  17  18  19
##   1   1   2   1   5   1   4   4   5  11  25  69 271
##
## $pred_enn
##
##   8  10  12  13  14  15  16  17  18  19
##   1   2   4   4   4  14  10  24  62 275
##
## $pred_nnet
##
##   3   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
##   1   4   1   4   5   2   2   6   4  12  12  21  23  36  56 211
```

```
acc_sub <- lapply(preds_sub, function(v) mean(get_vij(v, synth_data$Z2) == (k -
1)))
unlist(acc_sub)
```

```
##    pred_gmm    pred_qda pred_glmnet    pred_enn   pred_nnet
##      0.7575      0.5000      0.6775      0.6875      0.5275
```

```
acc_full <- lapply(preds_full, function(v) mean(get_vij(v, synth_data$Z2) == (K -
1)))
unlist(acc_full)
```

```
##    pred_gmm    pred_qda pred_glmnet    pred_enn   pred_nnet
##       0.630       0.383       0.590       0.604       0.495
```

Exponential extrapolation

```
library(nnls)
expmix <- function(ws, as, xs) {
  as.numeric(ws %*% exp(t(t(as)) %*%  t(xs)))
}
expbasis <- function(as, xs) {
  t(exp(t(t(as)) %*%  t(xs)))
}
fit_expmix <- function(as, xs, y) {
  X <- expbasis(as, xs)
  res <- nnls(X, y)
  sol <- res$x
  sol[sol < 1e-10] <- 0
  fit_a <- as[sol > 0]
  fit_w <- sol[sol > 0]
  ff <- function(xs) expmix(fit_w, fit_a, xs)
  list(a = fit_a, w = fit_w, f = ff)
}
binmom <- function(succ, tot, k) {
  choose(succ, k)/choose(tot, k)
}
expmix_binmom <- function(Vs, k, K, plot = FALSE) {
  momks <- sapply(1:k, function(x) mean(binmom(Vs, k -1, x - 1)))
  res <- fit_expmix(-seq(0, 5, 0.01), 1:k, momks)
  if (plot) {
    plot(1:max(K), res$f(1:max(K)), type = "l", ylim = c(0, 1))
    points(1:k, momks)
  }
  res$f(K)
}
c(expmix_binmom(V_subs$pred_gmm, k, K), acc_full$pred_gmm)
```

```
## [1] 0.5893481 0.6300000
```

```
c(expmix_binmom(V_subs$pred_qda, k, K), acc_full$pred_qda)
```

```
## [1] 0.3224499 0.3830000
```

```
c(expmix_binmom(V_subs$pred_glmnet, k, K), acc_full$pred_glmnet)
```

```
## [1] 0.5085619 0.5900000
```

```
c(expmix_binmom(V_subs$pred_enn, k, K), acc_full$pred_enn)
```

```
## [1] 0.5348675 0.6040000
```

```
c(expmix_binmom(V_subs$pred_nnet, k, K), acc_full$pred_nnet)
```

```
## [1] 0.4261939 0.4950000
```

Pseudolikelihood

Fit MPLE (mple), MPLE + monotonic constraint (mono), MPLE + moment constraint (mom), or MPLE + both constraints (mm).

```
library(nloptr)
us = seq(0, 1, 0.02) # change the discretization level

fit_pm_models <- function(Ys, k, us = seq(0, 1, 0.02), gu_init = rep(1/length(us),
length(us)),
                          K = k) {
  Ys <- as.numeric(Ys)
  (ws <- sapply(0:k, function(i) sum(Ys == i)))
  (momK <- mean(binmom(Ys, k - 1, k - 1)))
  usk <- us^(k-1)
  binprobs <- matrix(0, k + 1, length(us))
  for (i in 1:length(us)) binprobs[, i] <- dbinom(0:k, k, us[i])
  of_gu <- function(gu) {
    ft <- binprobs %*% gu
    -sum(ws * log(ft))
  }
  gof <- function(gu) {
    ft <- binprobs %*% gu
    -as.numeric(t(binprobs) %*% (ws/ft))
  }
  ## MPLE unconstrained
  t1 <- proc.time()
  res <- nloptr(gu_init, of_gu, eval_grad_f = gof,
                lb = 0 * us, ub = 0 * us + 1,
                eval_g_ineq = function(gu) sum(gu) - 1,
                eval_jac_g_ineq = function(gu) 0 * gu + 1,
                opts = list(algorithm = "NLOPT_LD_SLSQP",
                            xtol_rel = 1.0e-8,
                            print_level = 0,
                            check_derivatives = FALSE,maxeval = 1e4))
  (t2u <- proc.time() - t1)
  # print(res)
  gu_mple <- res$solution
  # disp_solution(res$solution); title("uncon")


  ## MPLE moment constraint
  t1 <- proc.time()
  res <- nloptr(gu_init, of_gu, eval_grad_f = gof,
                lb = 0 * us, ub = 0 * us + 1,
                eval_g_ineq = function(gu) sum(gu) - 1,
                eval_jac_g_ineq = function(gu) 0 * gu + 1,
                eval_g_eq = function(gu) sum(gu * usk) - momK,
                eval_jac_g_eq = function(gu) usk,
                opts = list(algorithm = "NLOPT_LD_SLSQP",
                            xtol_rel = 1.0e-8,
                            print_level = 0,
                            check_derivatives = FALSE,maxeval = 1e4))
  (t2mom <- proc.time() - t1)
  # print(res)
```

```
gu_mom <- res$solution

## MPLE monotonic constraint
ll <- length(us)
mat <- matrix(0, ll - 1, ll)
cmat <- (row(mat) == col(mat)) - (row(mat) == (col(mat) - 1))
eval_g_ineq_mono = function(gu) c(sum(gu) - 1, gu[-ll] - gu[-1])
eval_jac_g_ineq_mono = function(gu) rbind(0 * gu + 1, cmat)

t1 <- proc.time()
res <- nloptr(gu_init, of_gu, eval_grad_f = gof,
              lb = 0 * us, ub = 0 * us + 1,
              eval_g_ineq = eval_g_ineq_mono,
              eval_jac_g_ineq = eval_jac_g_ineq_mono,
              # eval_g_eq = function(gu) sum(gu * usk) - momK,
              # eval_jac_g_eq = function(gu) usk,
              opts = list(algorithm = "NLOPT_LD_SLSQP",
                          xtol_rel = 1.0e-8,
                          print_level = 0,
                          check_derivatives = FALSE,maxeval = 1e4))
(t2mono <- proc.time() - t1)
# print(res)
gu_mono <- res$solution

## MPLE 2 constraint
ll <- length(us)
mat <- matrix(0, ll - 1, ll)
cmat <- (row(mat) == col(mat)) - (row(mat) == (col(mat) - 1))
eval_g_ineq_mono = function(gu) c(sum(gu) - 1, gu[-ll] - gu[-1])
eval_jac_g_ineq_mono = function(gu) rbind(0 * gu + 1, cmat)

t1 <- proc.time()
res <- nloptr(gu_init, of_gu, eval_grad_f = gof,
              lb = 0 * us, ub = 0 * us + 1,
              eval_g_ineq = eval_g_ineq_mono,
              eval_jac_g_ineq = eval_jac_g_ineq_mono,
              eval_g_eq = function(gu) sum(gu * usk) - momK,
              eval_jac_g_eq = function(gu) usk,
              opts = list(algorithm = "NLOPT_LD_SLSQP",
                          xtol_rel = 1.0e-8,
                          print_level = 0,check_derivatives = FALSE,
                          maxeval = 1e4))
(t2c2 <- proc.time() - t1)
# print(res)
gu_mm <- res$solution

list(gu_mple = gu_mple, gu_mom = gu_mom, gu_mono = gu_mono, gu_mm = gu_mm,
     mple_est = sum(gu_mple * us^(K  - 1)),
     mom_est = sum(gu_mom * us^(K  - 1)),
     mono_est = sum(gu_mono * us^(K  - 1)),
     mm_est = sum(gu_mm * us^(K  - 1)))
```

```
}

pmle_gmm <- fit_pm_models(V_subs$pred_gmm, k, us, K = K)
c(true = acc_full$pred_gmm, unlist(pmle_gmm[5:8]))
```

```
##       true   mple_est    mom_est   mono_est     mm_est
## 0.63000000 0.03551845 0.54651552 0.03098945 0.57602920
```

```
pmle_qda <- fit_pm_models(V_subs$pred_qda, k, us, K = K)
c(true = acc_full$pred_qda, unlist(pmle_qda[5:8]))
```

```
##       true   mple_est    mom_est   mono_est     mm_est
## 0.38300000 0.01151813 0.24141417 0.10712073 0.29016531
```

```
pmle_glmnet <- fit_pm_models(V_subs$pred_glmnet, k, us, K = K)
c(true = acc_full$pred_glmnet, unlist(pmle_glmnet[5:8]))
```

```
##       true   mple_est    mom_est   mono_est     mm_est
## 0.59000000 0.02158243 0.42709501 0.03098945 0.47102245
```

```
pmle_enn <- fit_pm_models(V_subs$pred_enn, k, us, K = K)
c(true = acc_full$pred_enn, unlist(pmle_enn[5:8]))
```

```
##       true   mple_est    mom_est   mono_est     mm_est
## 0.60400000 0.02297476 0.43623682 0.03098945 0.48664740
```

```
pmle_nnet <- fit_pm_models(V_subs$pred_nnet, k, us, K = K)
c(true = acc_full$pred_nnet, unlist(pmle_nnet[5:8]))
```

```
##       true   mple_est    mom_est   mono_est     mm_est
## 0.49500000 0.01344954 0.26682531 0.13046145 0.33732096
```

High-dimensional asymptotics

```
meanexp <- function (v)
{
    vm <- max(v)
    mean(exp(v - vm)) * exp(vm)
}

piK <- function (mus, K, mc.reps = 10000)
{
    samp <- qnorm(((1:mc.reps) - 0.5)/mc.reps)
    sampmat <- repmat(t(samp), length(mus), 1) - mus
    temp <- log(1 - pnorm(sampmat))
    1 - apply((K - 1) * temp, 1, meanexp)
}

inv_piK <- function (p, K, upper = 20, res = 1000)
{
    if (p == 0)
        return(Inf)
    xs <- seq(0, upper, length.out = res + 1)
    ps <- piK(xs, K)
    xs[order(abs(ps - p))[1]]
}

extrapolate_hd <- function(acck, k, K) {
  1-piK(inv_piK(1-acck, k), K)
}

sapply(acc_sub, extrapolate_hd, k = k, K = K)
```

```
##    pred_gmm    pred_qda pred_glmnet    pred_enn   pred_nnet
##   0.6370327   0.3601287   0.5458248   0.5530054   0.3877789
```

```
unlist(acc_full)
```

```
##    pred_gmm    pred_qda pred_glmnet    pred_enn   pred_nnet
##       0.630       0.383       0.590       0.604       0.495
```

Build table

```
extrapolation_methods <- c("acc_sub", "acc_full", "exp",
                           "pmle", "pmle_mom", "pmle_mono", "pmle_mm",
                           "hd")
classifiers <- c("gmm", "qda", "glmnet", "enn", "nnet")

build_extrapolation_table <- function(synth_data, k, K) {

  tab <- matrix(NA, length(classifiers), length(extrapolation_methods),
                dimnames = list(classifiers, extrapolation_methods))
  preds_sub <- pred_submodel(synth_data$Ytr, synth_data$Yte,
                             synth_data$Z1, synth_data$Z2, k)
  preds_full <- pred_submodel(synth_data$Ytr, synth_data$Yte,
                              synth_data$Z1, synth_data$Z2, K)
  V_subs <- lapply(preds_sub, get_vij, Z2 =synth_data$Z2)
  acc_sub <- sapply(preds_sub, function(v) mean(get_vij(v, synth_data$Z2) == (k -
1)))
  acc_full <- sapply(preds_full, function(v) mean(get_vij(v, synth_data$Z2) == (K
- 1)))
  tab[, "acc_full"] <- acc_full
  tab[, "acc_sub"] <- acc_sub
  tab[, "exp"] <- sapply(V_subs, expmix_binmom, k = k, K = K)
  pmle_fits <- lapply(V_subs, fit_pm_models, k = k, us = us, K = K)
  for (i in 1:length(pmle_fits)) {
    tab[i, c("pmle", "pmle_mom", "pmle_mono", "pmle_mm")] <-
      unlist(pmle_fits[[i]][5:8])
  }
  tab[, "hd"] <- sapply(acc_sub, extrapolate_hd, k = k, K = K)
  tab
}
```

Simulations

Results for different noise levels.

```
## high noise
synth_data <- gen.data(p = 10, sigma = 3, K = 50, r1 = 20, r2 = 30)
build_extrapolation_table(synth_data, k = 20, K = 50)
```

```
##          acc_sub   acc_full        exp         pmle   pmle_mom  pmle_mono
## gmm    0.2050000 0.09800000 0.09508230 0.0021620503 0.06346643 0.03098945
## qda    0.1250000 0.05666667 0.08473500 0.0006800516 0.03848636 0.03098945
## glmnet 0.1900000 0.09333333 0.09348126 0.0014105723 0.05984712 0.03098945
## enn    0.1516667 0.07400000 0.09108530 0.0007040623 0.04566405 0.05587846
## nnet   0.1233333 0.07733333 0.04934708 0.0006184623 0.04499259 0.04861940
##           pmle_mm         hd
## gmm    0.11167272 0.11463271
## qda    0.06607235 0.06019360
## glmnet 0.10161142 0.10427227
## enn    0.07990312 0.07726179
## nnet   0.06467659 0.06019360
```

```
## low noise
synth_data <- gen.data(p = 10, sigma = 0.5, K = 50, r1 = 20, r2 = 30)
build_extrapolation_table(synth_data, k = 20, K = 50)
```

```
##          acc_sub  acc_full       exp       pmle  pmle_mom  pmle_mono
## gmm    0.9850000 0.9820000 0.9758202 0.04822418 0.9704382 0.26294357
## qda    0.8883333 0.8573333 0.8351186 0.04741848 0.7835500 0.03098945
## glmnet 0.9566667 0.9513333 0.9267394 0.04822418 0.9143357 0.26294357
## enn    0.9683333 0.9740000 0.9304755 0.04822418 0.9375743 0.26294357
## nnet   0.8650000 0.7860000 0.7596413 0.04730613 0.7392244 0.24201815
##          pmle_mm        hd
## gmm    0.9704299 0.9701820
## qda    0.7870464 0.8133798
## glmnet 0.9145464 0.9203284
## enn    0.9375743 0.9392982
## nnet   0.7568171 0.7831407
```

```
## very low noise
synth_data <- gen.data(p = 10, sigma = 0.3, K = 50, r1 = 20, r2 = 30)
build_extrapolation_table(synth_data, k = 20, K = 50)
```

```
##          acc_sub  acc_full       exp       pmle  pmle_mom pmle_mono
## gmm    1.0000000 0.9993333 1.0000000 0.04822418 1.0000000 0.2629436
## qda    0.9916667 0.9813333 0.9838733 0.04822418 0.9835721 0.2629436
## glmnet 0.9950000 0.9960000 0.9916444 0.04822418 0.9901433 0.2629436
## enn    1.0000000 0.9993333 1.0000000 0.04822418 1.0000000 0.2629436
## nnet   0.9683333 0.9153333 0.9429881 0.04822418 0.9376264 0.2629436
##          pmle_mm        hd
## gmm    0.9999957 1.0000000
## qda    0.9835722 0.9826077
## glmnet 0.9901432 0.9893390
## enn    0.9999957 1.0000000
## nnet   0.9376922 0.9392982
```