DimredBySize

Due to the required computation time, the code for computing the dimensionality reduction performances is commented out. The obtained results are stored in the "Results" folder. The results are then loaded back into R for plotting.

Load libraries

```
library("parallel") # parallel processing
library("ggplot2") # general plotting
library("ggpubr") # combining plots
library("umap") # get default umap parameters to change to 1 component
```

Specify the experiment parameters

```
set.seed(17) # seed for reproducibility
ntimes <- 100 # number of noise replicates to investigate dimred performance
npoints <- c(25, 50, 100) # number of points in our ground truth data sets to be investigated
maxdim <- 10000 # maximal dimension of the data set to be investigated
dims <- round(exp(seq(log(2), log(maxdim), length.out=10))) # dimensions to study from log-scale
a <- 1.25 # magnitude of noise: per dimension we sample noise uniformly from [-a, a]
alpha <- 3 # factor controling the growth rate of the ground truth diameter
```

Construct the ground data sets according to the growth rates

```
datasets <- lapply(npoints, function(np){
  t <- seq(0, 1, length.out=np)
  if(alpha==Inf) factor <- rep(1, maxdim) else factor <- (1:maxdim)**(-1 / alpha)
  data.frame(sapply(factor, function(f) t * f))
})
```

Setup clusters for parallel experiments

```
# n.cores <- detectCores()
# clust <- makeCluster(n.cores)
# clusterExport(clust, c("datasets",  "maxdim", "dims", "npoints", "a" , "t"))
# clusterEvalQ(clust, library("diffusionMap")) # diffusion map dimensionality reduction
# clusterEvalQ(clust, library("umap")) # UMAP dimensionality reduction
# clusterEvalQ(clust, library("rpca")) # Robust PCA dimensionality reduction
# clusterEvalQ(clust, library("dimRed")) # Isomap dimensionality reduction
```

We study how high-dimensional noise affects the PCA dimensionality reduction

```
# set.seed(13) # seed for reproducibility
# cors_pca  <- Reduce("+", parLapply(clust, 1:ntimes, function(n){
#   N <- matrix(runif(maxdim * max(npoints), min=-a, max=a), ncol=maxdim)
#   sapply(datasets, function(X){
#     XN <- X + N[1:nrow(X),]
#     sapply(dims, function(this_dim){
#       PCA <- prcomp(XN[,1:this_dim], rank.=1)$x[,1]
#       max(cor(PCA, X[,1]), cor(PCA[rev(1:nrow(X))], X[,1]))
#     })
#   })
```

```r
# })) / ntimes
# cors_pca <- data.frame(size=rep(npoints, each=length(dims)),
#                         dim=rep(dims, length(npoints)),
#                         cor=as.numeric(cors_pca))
# saveRDS(cors_pca, file="Results/Size/PCA.rds") # store the results

cors_pca <- readRDS("Results/Size/PCA.rds") # load the results
P1 <- ggplot(data=cors_pca, aes(x=dim, y=cor, color=factor(size))) +
  geom_line() +
  geom_point() +
  scale_x_log10() +
  ylab("correlation") +
  labs(col="data size") +
  ggtitle("PCA") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

We study how high-dimensional noise affects the UMAP dimensionality reduction

```r
# custom.config <- umap.defaults
# custom.config$n_components <- 1
# clusterExport(clust, "custom.config")
# set.seed(13) # seed for reproducibility
# cors_umap  <- Reduce("+", parLapply(clust, 1:ntimes, function(n){
#   N <- matrix(runif(maxdim * max(npoints), min=-a, max=a), ncol=maxdim)
#   sapply(datasets, function(X){
#     XN <- X + N[1:nrow(X),]
#     sapply(dims, function(this_dim){
#       UMAP <- umap(XN[,1:this_dim], config=custom.config)$layout[,1]
#       max(cor(UMAP, X[,1]), cor(UMAP[rev(1:nrow(X))], X[,1]))
#     })
#   })
# })) / ntimes
# cors_umap <- data.frame(size=rep(npoints, each=length(dims)),
#                          dim=rep(dims, length(npoints)),
#                          cor=as.numeric(cors_umap))
# saveRDS(cors_umap, file="Results/Size/UMAP.rds") # store the results

cors_umap <- readRDS("Results/Size/UMAP.rds") # load the results
P2 <- ggplot(data=cors_umap, aes(x=dim, y=cor, color=factor(size))) +
  geom_line() +
  geom_point() +
  scale_x_log10() +
  ylab("correlation") +
  labs(col="data size") +
  ggtitle("UMAP") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

We study how high-dimensional noise affects the diffusion map dimensionality reduction

```r
# set.seed(13) # seed for reproducibility
# cors_diff  <- Reduce("+", parLapply(clust, 1:ntimes, function(n){
#   N <- matrix(runif(maxdim * max(npoints), min=-a, max=a), ncol=maxdim)
#   sapply(datasets, function(X){
```

```
#     XN <- X + N[1:nrow(X),]
#     sapply(dims, function(this_dim){
#       invisible(capture.output(DM <- suppressWarnings(diffuse(dist(XN[,1:this_dim]), maxdim=1)$X)))
#       max(cor(DM, X[,1]), cor(DM[rev(1:nrow(X))], X[,1]))
#     })
#   })
# })) / ntimes
# cors_diff <- data.frame(size=rep(npoints, each=length(dims)),
#                          dim=rep(dims, length(npoints)),
#                          cor=as.numeric(cors_diff))
# saveRDS(cors_diff, file="Results/Size/DIFFM.rds") # store the results

cors_diff <- readRDS("Results/Size/DIFFM.rds") # load the results
P3 <- ggplot(data=cors_diff, aes(x=dim, y=cor, color=factor(size))) +
  geom_line() +
  geom_point() +
  scale_x_log10() +
  ylab("correlation") +
  labs(col="data size") +
  ggtitle("DiffusionMap") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

We study how high-dimensional noise affects the robust PCA dimensionality reduction

```
# set.seed(13) # seed for reproducibility
# cors_rpca  <- Reduce("+", parLapply(clust, 1:ntimes, function(n){
#   N <- matrix(runif(maxdim * max(npoints), min=-a, max=a), ncol=maxdim)
#   sapply(datasets, function(X){
#     XN <- as.matrix(X + N[1:nrow(X),])
#     sapply(dims, function(this_dim){
#       RPCA <- rpca(XN[,1:this_dim])
#       RPCA <- RPCA$L.svd$u[,1] * RPCA$L.svd$d[1]
#       max(cor(RPCA, X[,1]), cor(RPCA[rev(1:nrow(X))], X[,1]))
#     })
#   })
# })) / ntimes
# cors_rpca <- data.frame(size=rep(npoints, each=length(dims)),
#                          dim=rep(dims, length(npoints)),
#                          cor=as.numeric(cors_rpca))
# saveRDS(cors_rpca, file="Results/Size/RPCA.rds") # store the results

cors_rpca <- readRDS("Results/Size/RPCA.rds") # load the results
P4 <- ggplot(data=cors_rpca, aes(x=dim, y=cor, color=factor(size))) +
  geom_line() +
  geom_point() +
  scale_x_log10() +
  ylab("correlation") +
  labs(col="data size") +
  ggtitle("Robust PCA") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

We study how high-dimensional noise affects the robust PCA dimensionality reduction

The experiments are conducted in Python and the results are loaded in R for plotting

```r
cors_auto <- read.table("Results/Size/AUTO_alpha2.csv", # load the results
                        sep=",", row.names=1, header=TRUE)
P5 <- ggplot(data=cors_auto, aes(x=dim, y=cor, color=factor(size))) +
  geom_line() +
  geom_point() +
  scale_x_log10() +
  ylab("correlation") +
  labs(col="data size") +
  ggtitle("AutoEncoder") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

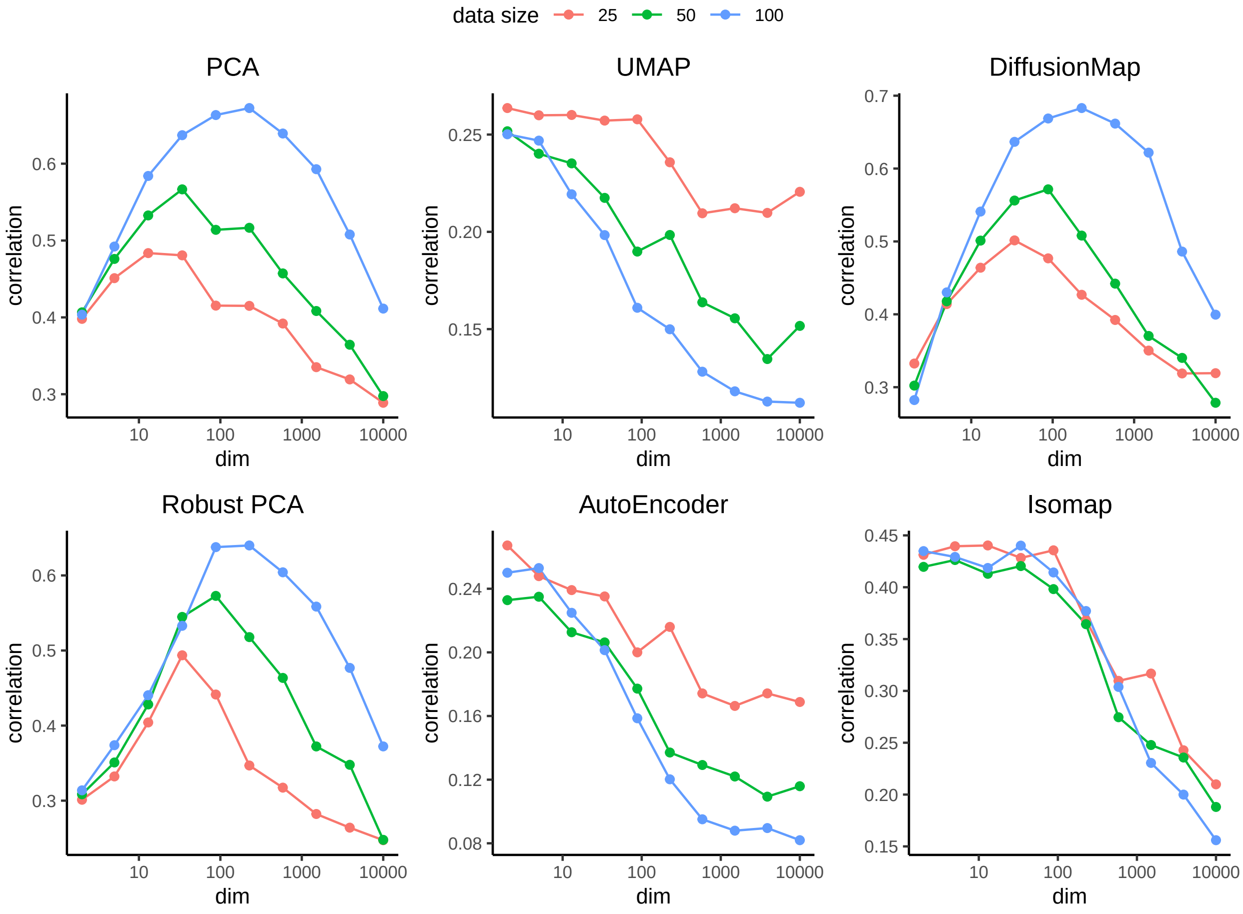
We study how high-dimensional noise affects the Isomap dimensionality reduction

```r
# set.seed(13) # seed for reproducibility
# cors_iso  <- Reduce("+", parLapply(clust, 1:ntimes, function(n){
#   N <- matrix(runif(maxdim * max(npoints), min=-a, max=a), ncol=maxdim)
#   sapply(datasets, function(X){
#     XN <- X + N[1:nrow(X),]
#     sapply(dims, function(this_dim){
#       ISO <- embed(XN[,1:this_dim], "Isomap", knn=10, ndim=1, .mute=c("message"))@data@data[,1]
#       max(cor(ISO, X[,1]), cor(ISO[rev(1:nrow(X))], X[,1]))
#     })
#   })
# })) / ntimes
# cors_iso <- data.frame(size=rep(npoints, each=length(dims)),
#                        dim=rep(dims, length(npoints)),
#                        cor=as.numeric(cors_iso))
# saveRDS(cors_iso, file="Results/Size/ISO.rds") # store the results

cors_iso <- readRDS("Results/Size/ISO.rds") # load the results
P6 <- ggplot(data=cors_iso, aes(x=dim, y=cor, color=factor(size))) +
  geom_line() +
  geom_point() +
  scale_x_log10() +
  ylab("correlation") +
  labs(col="data size") +
  ggtitle("Isomap") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

Finally, we combine the plots for comparison

```r
ggarrange(P1, P2, P3, P4, P5, P6, nrow=2, ncol=3, common.legend=TRUE)
```


data size
25
50
100
PCA
UMAP
DiffusionMap
Robust PCA
AutoEncoder
Isomap
correlation
dim
10
100
1000
10000
0.3
0.4
0.5
0.6
0.7
0.15
0.20
0.25
0.08
0.12
0.16
0.24
0.30
0.35
0.40
0.45