STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Cross validation in finance

Why K-fold CV fails in finance

- Observations cannot be assumed to be drawn from an IID process
    - e.g., correlated feature X_t 's with labels Y_t 's that are formed on overlapping data
    - information is leaked from t to $t+1$; that is, $X_t \approx X_{t+1}$ and $Y_t \approx Y_{t+1}$
    - then, dependence between (Y_t, X_t) would be inherited by (Y_{t+1}, X_{t+1})
    - inflate performance, or even lead to false discovery
- Test set could be used multiple times in the process of developing a model

How to reduce info leakage

- Purged K-fold CV
  - Drop from the training set the observations with label Y_i's overlapped with those for the test set (unless the corresponding features are independent; see below)
- Avoid overfitting to take less advantage of the leaked information from the test set
  - Early stopping
  - Bagging with a maximum fraction of samples
  - Bagging with sequential bootstrap
- **Note!** For leakage to take place, it must occur that $(X_i, Y_i) \approx (X_j, Y_j)$. No leakage if only either $X_i \approx X_j$ or $Y_i \approx Y_j$.

Purged k-fold CV

- When leakage takes place, performance improves merely by increasing $k \to I$ (I is the total number of features bars), as the number of overlapping observations in the training set is increasing
- A solution: Purged k-fold CV
  - Let labels $Y_j = f(\Phi_j)$ and $Y_i = g(\Phi_i)$ from the test and training sets respectively, then remove Y_i from the training set if $\Phi_i \cap \Phi_j \neq \emptyset$
  - In many cases, purging suffices to prevent leakage, and a larger k allows the model to re-calibrate more often
  - However, when k is larger than a threshold, performance stops improving

Embargo

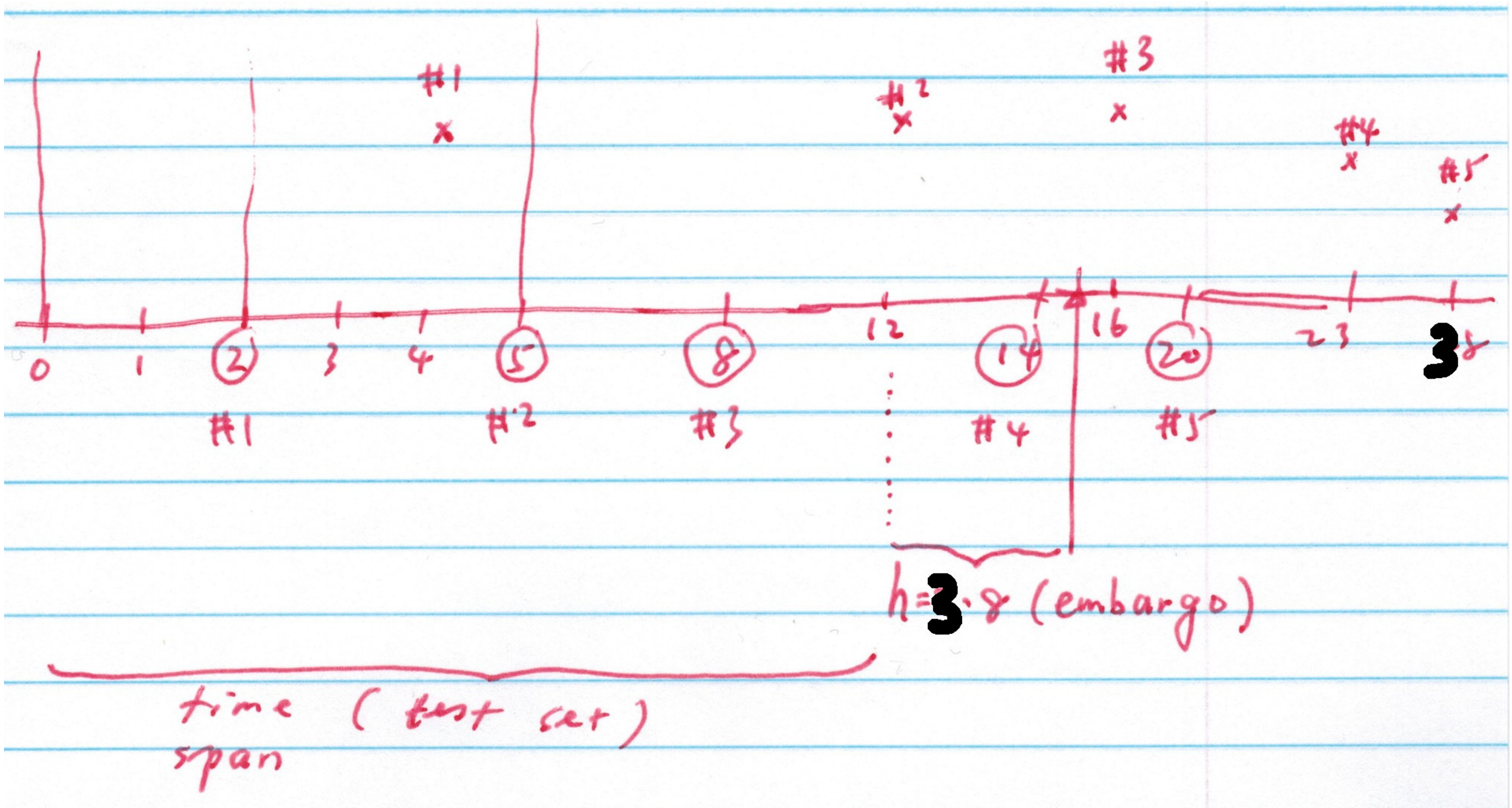
- When purging does not prevent all leakage, impose an embargo on training set right **after** every test set
- Further reduce the chance of information leakage **from the future**
- Not necessary for training sets before the test sets
- See FIGURE 7.3 of AFML

Embargo - Implementation

- Simply adding a small time window of width $h = \gamma T$ right after each test set, before conducting purging; that is, assume that the label of the last feature bar in the test set depends on info from $[t_{j,0}, t_{j,1} + h]$, where $[t_{j,0}, t_{j,1}]$ is the original bars covered.
- T is the total number of dollar/volume/etc bars
- A small γ such as $\gamma = 0.01$ suffices, but it really depends on how large I and T are, respectively

Purged CV with embargo - a toy example

```r
feaMat <- data.frame(Y=c(1,1,0,1,0), V=c(2,4,2,4,1),
                     tFea=c(2,5,8,14,20), tLabel=c(4,12,16,23,38))
```



Purged CV with embargo - a toy example

```
[[1]]$testSet
  Y V tFea tLabel
1 1 2    2      4
2 1 4    5     12
[[1]]$trainSet
  Y V tFea tLabel
5 0 1   20     38
```

```
[[2]]$testSet
  Y V tFea tLabel
3 0 2    8     16
4 1 4   14     23
5 0 1   20     38
[[2]]$trainSet
  Y V tFea tLabel
1 1 2    2      4
```

Purged CV with embargo - implementation

```r
#' @param feaMat:
#  a data.frame for feature matrix with the first column being the label
#   "tFea": time index for features bars,
#           i.e., the time index at the end of each features bars
#   "tLabel": time index where events occur (e.g., touching some barrier)
#' @param k: number of folds for k-fold CV
#' @param gam: gamma for embargo
#' @return a list of k data.frame, each containing a test set and a training set

purged_k_CV <- function(feaMat,k=5,gam=0.01){ ... }
```

Purged CV with embargo - implementation

- Divide the data (i.e., features bars and labels) into k folds

```
I <- nrow(feaMat)
nFold <- c(rep(wd <- floor(I/k), k-1), I - wd*(k-1)) # size of each fold
cumnFold <- c(0,cumsum(nFold))
out <- lapply(1:k, function(i){
testInx <- (cumnFold[i]+1):cumnFold[i+1]
testFold <- feaMat[testInx,]
trainFold <- feaMat[setdiff(1:I,testInx),]
...
}
```

Purged CV with embargo - implementation

- Purge the training sets and add embargo

```
TT <- max(feaMat$tLabel)
h <- gam*TT

# last time index of the test set + embargo
tEmbargo <- max(testFold$tLabel)+h

# first time index of the test set,
# +1 because the info at time 0 of the feature bar is not used for labeling
t0Test <- min(testFold$tFea)+1

trainFold <- subset(trainFold, (tLabel<t0Test) | (tFea>=tEmbargo))
```

- Try R
- Back to Course Scheduler