STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Performance measures

Baseline Performance

- Baseline performance is the performance that can be achieved by a much simpler model
- Use baseline performance to assess whether a machine learning model is at all useful
- For regression models, use conditional mean or median, or some basic linear regression models
- For classification models, use guesses of the following three types:
  - Random guess: based on the total number of classes in the training set, randomly assign a class for each observation from the test set.
  - Educated guess: based on the percentage of each class in the training set, assign a class accordingly for each observation from the test set.
  - Majority guess: assign the class that appears most often in the training set to all the observations from the test set.
  - use `fmlr::acc_lucky()`

Confusion matrix

- See FIGURE 3.2 of AFML
- True pos rate, Recall, Power, Sensitivity, 1 - Type II err = TP / left rectangle
- False pos rate, 1 - Specificity, Type I err = FP / right rectangle
- Precision = TP / ellipse
- Accuracy = (TP+TN) / all

ROC / AUC / F1 score

- Receiver Operating Characteristic (ROC) curve:
- AUC: area under the ROC curve
- F1 score: harmonic mean of precision and recall
  - $F_1 = \frac{2}{1/\text{precision}+1/\text{recall}}$
  - $F_1 \in [0, 1]$, the higher the better
- In order to have higher values for **both** precision and recall
- Try R

Sample weights and Sequential bootstrap

Sample weights

- Overlapping outcomes
  - For some $i < j$, both y_i and y_j may depend on the return over $[t_{j,0}, \min\{t_{i,1}, t_{j,1}\}]$
  - $\{y_i\}, i = 1, \ldots, I$, are not IID if there exists i s.t. $t_{i,1} > t_{i+1,0}$
  - Avoiding overlaps leads to loss of information

Number of concurrent labels

- Two labels y_i and y_j are concurrent at t when both are a function of at least one common return
- Compute the number of labels that are a function of a given return $r_{t-1,t}$
  - For each $t = 1, \ldots, T$, form $\{1_{t,i}\}, i = 1, \ldots, I$, where $1_{t,i} \in \{0, 1\}$
  - Variable $1_{t,i} = 1$ if and only if $[t_{i,0}, t_{i,1}]$ overlaps with $[t-1, t]$
  - Number of concurrent at t: $c_t = \sum_{i=1}^{I} 1_{t,i}$.

Average uniqueness of a label

- A label's degree of non-overlapping can be estimated as its average uniqueness over its lifespan
- The uniqueness of a label i at time t is $u_{t,i} = 1_{t,i}/c_t$
  - 0: label i does not overlap with $[t-1, t]$
  - 1: **Only** label i overlaps with $[t-1, t]$
- Average uniqueness: $\bar{u}_i = (\sum_{t=1}^{T} u_{t,i})/(\sum_{t=1}^{T} 1_{t,i})$
  - $\bar{u}_i \in (0, 1)$
  - The higher the $\bar{u}_i$, the lower the degree of overlapping/dependence between label i and the other labels
  - See a histogram

Construct indicator matrix

```
t1_Fea <- c(2,5,11)
t1 <- c(9,11,15)
I <- length(t1_Fea) # number of features bars
t1_max <- max(t1) # same as the total number of bars
indMat <- matrix(0, t1_max, I) # indicator matrix 1_{t,i}
for(i in 1:I){indMat[(t1_Fea[i]+1):t1[i],i] <- 1}
t(indMat)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    0    0    1    1    1    1    1    1    1     0     0     0     0
## [2,]    0    0    0    0    0    1    1    1    1     1     1     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     1     1
##      [,14] [,15]
## [1,]     0     0
## [2,]     0     0
## [3,]     1     1
```

Get average uniqueness

```
sampled <- NULL # or something like: sampled <- c(1,3,3)
if(is.null(sampled)){sampled <- 1:ncol(indMat)}
indMat_ <- indMat[,sampled]
c_t <- apply(indMat_, 1, sum)
u_ti <- indMat_ / c_t
u_ti[c_t==0,] <- 0 # some c_t is 0, and need to change NaN to 0
ubar_i <- apply(u_ti,2,sum) / apply(indMat_,2,sum)
ubar_i
```

```
## [1] 0.7142857 0.6666667 1.0000000
```

```
c(mean(c(1,1,1,0.5,0.5,0.5,0.5)), mean(c(0.5,0.5,0.5,0.5,1,1)),
  mean(c(1,1,1,1)))
```

```
## [1] 0.7142857 0.6666667 1.0000000
```

Bagging classifiers and uniqueness

- Bootstrap aggregating (i.e., bagging) samples **with** replacement
- Let $N = \sum_{i=1}^{I} 1_i$, and $1_i = 1$ if i is selected after I draws. Then $E[N] = I \times E[1_i] = I \times [1 - (1 - 1/I)^I]$
- $1 - (1 - 1/I)^I \to (1 - e^{-1}) \approx 2/3$ as $I \to \infty$
- Getting worse if only $K(< I)$ features are non-overlapping: $N_1 = \sum_{i=1}^{K} 1_i$, and $E[N_1] = K \times E[1_i] = K \times [1 - (1 - 1/K)^I]$
- $[1 - (1 - 1/K)^I] \approx 1 - e^{-I/K}$ when K is large, and numerical comparison suggests that $E[N_1] < E[N]$.

Issues for bagging

- When $\sum_{i=1}^{I} \bar{u}_i / I \ll 1$, i.e., the overall average uniqueness is too small,
  - in-bag observations are redundant to each other
    - Different models are too similar to each other
  - in-bag observations are very similar to out-of-bag observations
    - out-of-bag accuracy is inflated

Some solutions

- Do not drop overlapping outcomes, and dropping **partial overlaps** results in extreme loss of information
- The in-bag observations are not sampled at a frequency much higher than their uniqueness
- Sequential bootstrap: draws are made according to a changing probability that controls for redundancy

Sequential bootstrap

1. Randomly choose $X_i, i \in 1, \dots, I$; let φ be the sequence of draws, then $\varphi^{(1)} = \{i\}$
2. Then the uniqueness of j at time t is $u_{t,j}^{(2)} = 1_{t,j}(1 + \sum_{k \in \varphi^{(1)}} 1_{t,k})^{-1}$, and the average uniqueness over the lifespan is $\bar{u}_j^{(2)} = (\sum_{t=1}^{T} u_{t,j}^{(2)})(\sum_{t=1}^{T} 1_{t,j})^{-1}$
3. Make a 2nd draw based on the updated probabilities:
   - $\delta_j^{(2)} = \bar{u}_j^{(2)}(\sum_{k=1}^{I} \bar{u}_k^{(2)})^{-1}$
4. Update $\varphi^{(2)}$, and draw the 3rd one based on $\delta_j^{(3)}$, and so on.

Sequential bootstrap - toy example

- For example

$$\{1_{t,i}\} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

- Randomly select 1, 2, and 3, and 2 is randomly selected.
- Then $\bar{u}_1^{(2)} = (1 + 1 + 1/2)/3 = 5/6$, $\bar{u}_2^{(2)} = (1/2 + 1/2)/2 = 1/2$, $\bar{u}_3^{(2)} = 1$.
- Then $\delta^{(2)} = \{5/14, 3/14, 6/14\}$.

Sequential bootstrap - implementation

- See Github for any updated implementation

```r
# phi: the vector used to store the sampled features bars index
# sLen: how many draws, the default is length(t1_Fea)
phi <- sample(1:sLen, 1)

for(s in 2:sLen) # the loop for sequential bootstrap
{
  # update average uniqueness based on phi
  c_t <- apply(as.matrix(indMat[,phi],nrow=t1_max), 1, sum)
  u_ti <- indMat / (c_t+1)
  ubar_i <- apply(u_ti,2,sum) / apply(indMat,2,sum)

  # sample the next one based on updated average uniqueness
  phi <- c(phi, sample(1:sLen, 1, prob = ubar_i / sum(ubar_i)))
}
```

- Try R

Some other weights

Return attribution

- Motivation: labels with large absolute returns should be given more weights.
- $\tilde{w}_i = \left|\sum_{t=t_{i,0}}^{t_{i,1}} \frac{r_{t-1,t}}{c_t}\right|$, where $r_{t-1,t} = \ln(p_t) - \ln(p_{t-1})$ is log-return.
- Based on the absolute **log returns** that can be attributed **uniquely** to it

Some other weights

Time decay

- Motivation: labels with new observations should be given more weights.
- Decay takes place according to cumulative uniqueness $x \in [0, \sum_{i=1}^{I} \bar{u}_i]$
- Let $d(x)$ be the weight due to time decay, eg., $d(x) = \max(0, ax + b)$ with the boundary conditions:
  - $d(\sum_{i=1}^{I} \bar{u}_i) = 1$
  - $d(x) = 0$ for x less than some threshold

Some other weights

Class weights

- Motivation: labels for underrepresented classes should be given more weights.
- Back to Course Scheduler