STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Hyper-Parameter Tuning with CV

Grid search

- Exhaustive search on the parameters space to maximize CV performance according to some user-defined score functions
- Use F1 score in the context of meta-labeling applications
    - For example, for a sample with a lot more "0" than "1", a classifier that predicts all cases to be "0" will achieve high "accuracy", even if it has not learned from the features how to discriminate between these cases
    - F1 scores correct such performance inflation
- For other (non-meta-labeling) applications, it is fine to use "accuracy", because we are equally interested in predicting all cases.

Randomized search CV

- When there are many parameters to tune
- Sample each parameter from a distribution
    - Control for the number of combinations to reduce costs
    - Exclude parameters' values that are relatively irrelevant performance-wise
    - Depending on the potential impacts from the parameters, try to search "uniformly"
    - See a log-uniform example in AFML
        - $\log p \sim \mathcal{U}(\log(a), \log(b))$

Scoring and hyper-parameter tuning

- For **meta labeling**, accuracy does not provide a realistic scoring of the classifier's performance. A toy example:

	predicted	
observed	0	1
0	40	2
1	2	1

- accuracy = 91% (41/45)
- The no-information rate is 93% (42/45), the largest proportion of the observed classes
- The model has not learned how to discriminate between cases although accuracy is very high

Scoring and hyper-parameter tuning

- Use F1 scores for meta-labeling applications
- For the previous example, F1 = 33% (2/(1/(1/3) + 1/(1/3)))
- Try R

Scoring and hyper-parameter tuning

- For **non meta-labeling**, where both sides are important, one can use either accuracy or cross entropy loss, and cross entropy loss is preferred as accuracy does not account for the confidence of predictions.
    - Recall that given a cutoff of 0.5 for a binary classification, any probability larger than 0.5 can predict '1'. However, 'accuracy' omits the confidence of being '1'.
- See FIGURE 9.2 from AFML

Cross entropy loss - revisit

- Entropy
  - Average amount of information of a sample drawn from a given distribution: higher entropy more unpredictable
  - Entropy of a probability measure $X \sim F$ with density/mass functions f:
    - $H(F) = E(-\log f(X))$
      - Discrete case: $-\sum_x f(x) \log f(x)$
      - Continuous case: $-\int_x f(x) \log f(x) dx$
- For example:
  - $X \sim \text{Bernoulli}(p)$, then $H(F) = -p \log p - (1-p) \log(1-p)$
  - $p \in \{0, 1\} \Rightarrow H(F) = 0$ with $0 \log(0) = 0$
  - $p = 1/2 \Rightarrow H(F) = \log(2)$

Cross entropy loss - revisit

- Cross entropy
  - Cross entropy between two probability measures $X \sim F$ and $Y \sim G$ with density/mass functions f and g, respectively:
    - $H(F, G) = E(-\log g(X))$
      - Discrete case: $-\sum_x f(x) \log g(x)$
      - Continuous case: $-\int_x f(x) \log g(x) dx$
  - If $f = g$, then $H(F, G) = H(F) = H(G)$
  - Negative log-likelihood is a cross entropy between the empirical distribution (f) and the model (g)

Cross entropy loss - revisit

- $H(F, G) - H(F) = E(-\log g(X) + \log f(X)) = E(\log\left(\frac{f(X)}{g(X)}\right))$
- Kullback–Leibler divergence: $D_{KL}(F||G) = E(\log\left(\frac{f(X)}{g(X)}\right)) \geq 0$ (use $\log x \leq x - 1$ for $x > 0$)
- MLE is to minimize cross entropy, i.e., to minimize KL-divergence with G the model to be estimated

Cross entropy loss - revisit

- Recall: Classification tree
  - cross entropy = $\sum_{m=1}^{|T|} N_m \left(-\sum_k \hat{p}_{mk} \log \hat{p}_{mk}\right)$, where N_m is number of observations in R_m, and $\hat{p}_{mk} = N_{mk}/N_m$ is the proportion of kth class observations in R_m. Therefore,

$$\begin{aligned}\text{cross entropy} &= -\sum_{m=1}^{|T|}\left(\sum_k \left[N_{mk} \log \hat{p}_{mk}\right]\right) \\ &= -\sum_k \left(\sum_{m=1}^{|T|}\sum_{x_i \in R_m} \left[1_{\{y_i=k\}} \log \hat{p}_{mk}\right]\right).\end{aligned}$$

  - Look at $\hat{p}_{mk}$ as the probability mass based on the tree model, and cross entropy is just the negative log-likelihood.

Cross entropy loss vs Accuracy

```
library(MLmetrics)
LogLoss
```

```
## function (y_pred, y_true)
## {
##     eps <- 1e-15
##     y_pred <- pmax(pmin(y_pred, 1 - eps), eps)
##     LogLoss <- -mean(y_true * log(y_pred) + (1 - y_true) * log(1 -
##         y_pred))
##     return(LogLoss)
## }
## <bytecode: 0x55a2880e5608>
## <environment: namespace:MLmetrics>
```

Cross entropy loss vs Accuracy

Now, we construct two predictions that have the same accuracy and confusion matrix, but different cross entropy losses.

```
library(SDMTools)
set.seed(1)
obs <- rep(0,45); obs[one_obs <- sample(1:45,20,replace = F)] <- 1
pre1 <- pre2 <- runif(45)
pre1[one_pre <- sample(one_obs, 10, replace = F)] <- runif(10, 0.7,0.9)
pre2[one_pre] <- runif(10, 0.51,0.7)
```

Clearly, 'pre1' outperforms 'pre2'. However …

Cross entropy loss vs Accuracy

```
t(confusion.matrix(obs, pre1, threshold = 0.5))
```

```
##    pred
## obs  0  1
##   0 16  9
##   1  4 16
## attr(,"class")
## [1] "confusion.matrix"
```

```
t(confusion.matrix(obs, pre2, threshold = 0.5))
```

```
##    pred
## obs  0  1
##   0 16  9
##   1  4 16
## attr(,"class")
## [1] "confusion.matrix"
```

Cross entropy loss vs Accuracy

```
LogLoss(pre1, obs)
```

```
## [1] 0.6827992
```

```
LogLoss(pre2, obs)
```

```
## [1] 0.751262
```

- Back to Course Scheduler