Predictive Modelling Tutorial 4: Model Validation Theory

Dr Liew How Hui

Jan 2021

Tut 4: Model Validation Theory

"Bias-Variance Decomposition" gives us the decomposition of expected test error as follows:

$$\underbrace{\mathbb{E}_{\boldsymbol{x},y,D}\left[\left(h_D(\boldsymbol{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{\mathbb{E}_{\boldsymbol{x},D}\left[\left(h_D(\boldsymbol{x})-\bar{h}(\boldsymbol{x})\right)^2\right]}_{\text{Variance}} +$$

$$\underbrace{\mathbb{E}_{\boldsymbol{x}}\left[\left(\bar{h}(\boldsymbol{x})-\bar{y}(\boldsymbol{x})\right)^2\right]}_{\text{Bias}^2} +$$

$$\underbrace{\mathbb{E}_{\boldsymbol{x},y}\left[\left(\bar{y}(\boldsymbol{x})-y\right)^2\right]}_{\text{Noise}}.$$

Bias-Variance Tradeoff

We study the example below following `https://daviddalpiaz.github.io/r4sl/biasvariance-tradeoff.html`:

$$Y = X^2 + \epsilon, \quad \epsilon \sim N(0, 0.3).$$

```
f = function(x) { x^2 }
gen_sim_data = function(f, sample_size = 100) {
  x = runif(n = sample_size, min = 0, max = 1)
  y = rnorm(n = sample_size, mean = f(x), sd = 0.3)
  data.frame(x, y)
}
```

Bias-Variance Tradeoff (cont)

```
set.seed(1)
n_sims = 250
n_models = 4
X = data.frame(x = 0.90) # fixed point at which we make predictions
predictions = matrix(0, nrow = n_sims, ncol = n_models)

for (sim in 1:n_sims) {
  sim_data = gen_sim_data(f)  # True model: Y = X^2 + e

  # Predictive models: We don't know the true model
  fit_0 = lm(y ~ 1,                   data = sim_data)
  fit_1 = lm(y ~ poly(x, degree = 1), data = sim_data)
  fit_2 = lm(y ~ poly(x, degree = 2), data = sim_data)
  fit_9 = lm(y ~ poly(x, degree = 9), data = sim_data)

  # Predictions by the predictive models
  predictions[sim, 1] = predict(fit_0, X)
  predictions[sim, 2] = predict(fit_1, X)
  predictions[sim, 3] = predict(fit_2, X)
  predictions[sim, 4] = predict(fit_9, X)
}
```

Bias-Variance Tradeoff (cont)

```
boxplot(predictions, names=c("deg=0","deg=1","deg=2", "deg=9"))

get_mse  = function(estimate, truth) { mean((estimate-truth)^2) }
get_bias = function(estimate, truth) { mean(estimate) - truth }
get_var  = function(estimate) { mean((estimate - mean(estimate))^2) }

bias     = apply(predictions, 2, get_bias, truth = f(x = 0.90))
variance = apply(predictions, 2, get_var)
mse      = apply(predictions, 2, get_mse, truth = f(x = 0.90))
```

Two things are immediately clear:

- As complexity $\nearrow$, bias $\searrow$. (The mean of a model's predictions is closer to the truth.)
- As complexity $\nearrow$, variance $\nearrow$. (The variance about the mean of a model's predictions increases.)

Tutorial 3, Q1

What are the advantages of k-fold cross validation relative to

(a) Validation set approach

(b) Leave-one-out cross validation (LOOCV)

Feature Importance / Selection

Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. Feature selection selects important features based on the “scores”.
Aim: remove noisy input.
Popular examples:

- statistical correlation scores (P&S I or II)
- linear model: p-value (Topic 3: LR & ARA/ASM), lasso, permutation importance scores (`mmpf`)
- decision trees’ Boruta(?), variable importance measure

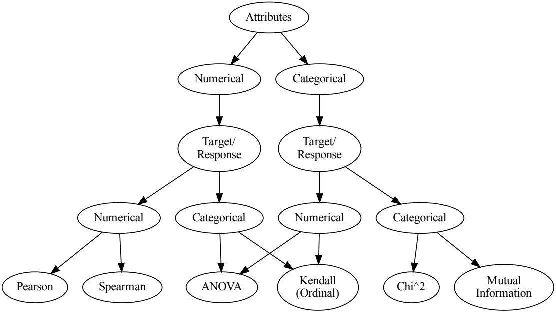
Feature Importance (cont)

Quote from Max Kuhn, Kjell Johnson (2013): Applied Predictive Modeling:

> *Often, we desire to quantify the strength of the relationship between the predictors and the outcome. [...] Ranking predictors [...] can be very useful when sifting through large amounts of data.*

Feature Importance (cont)

For statistical correlation scores (P&S I or II), `https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/` construct a nice summary:



Feature Importance (cont)

Find the correlations using R and Python

```
Normal    = c(56,56,65,65,50,25,87,44,35)
Hypervent = c(87,91,85,91,75,28,122,66,58)
```

Answer: cor, anova, chisq.test, `infotheo` library; np.corrcoef, stats.pearsonr, stats.spearmanr, stats.kendalltau, stats.f_oneway, stats.chisquare, stats.chi2_contingency, sklearn.metrics.mutual_info_score, sklearn.feature_selection.mutual_info_classif

Feature Importance (cont)

Linear model *p*-values (Topic 3: LR & ARA/ASM)
R — `summary(fit)$coefficients[,4]`
Python — `statmodels`, sklearn does not have it.

```
# linear regression feature importance
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
import numpy as np
from scipy import stats
# define dataset
X, y = make_regression(n_samples=1000, n_features=10,
         n_informative=5, random_state=1)
model = LinearRegression()
model.fit(X, y)
beta = np.append(model.intercept_,model.coef_)
```

Source: https://machinelearningmastery.com/calculate-feature-importance-with-python/

Feature Importance (cont)

Suggestion from
https://stackoverflow.com/questions/27928275/find-p-value-significance-in-scikit-learn-linearregression

```
yhat = model.predict(X)
n, p = X.shape[0], X.shape[1]
MSE = sum((y-yhat)**2)/(n-p-1)
newX = np.hstack( (np.ones((X.shape[0],1)), X) )
sd_b = np.sqrt(MSE*(np.linalg.inv(newX.T @ newX).diagonal()))
ts_b = beta / sd_b
p_values = [2*(1-stats.t.cdf(np.abs(i),(n-p-1))) for i in ts_b]

# summarize feature importance
for i in np.argsort(p_values):
    print('Feature: %s, p-value: %.5f' % (i,p_values[i]))
```

Feature Importance (cont)

If time permits, analyse the flame data `http://cs.joensuu.fi/sipu/datasets/flame.txt` use in lecture with an extra column, use the predictive models from Topics 2 and 3 to study the effect of feature to the classification using confusion matrix.