Cvičení 8a

Převzorkování a křížová validace

Jan Přikryl

ČVUT FD

14. dubna 2020

Příklad 1

Trénovací a validační množina

Nejprve vyzkoušíme postup odhadu testovací chyby lineárního modelu na trénovací a validační množině dat.

Výběr trénovacích a testovacích dat je randomizovaný, pro sjednocení výstupů nastavíme nejprve

```
rng(1)
```

Vytvoříme trénovací a testovací podmnožinu dat `islr_auto.csv`. Nejprve je ale musíme vyčistit:

```
auto = readtable('islr_auto.csv','TreatAsEmpty','?')
invalid = isnan(auto.horsepower);
auto(invalid,:)=[];
```

Příklad 1

Pokračování

Rozdělíme data na testovací a trénovací. S výhodou lze využít funkce `randperm()` a logického indexování:

```
num_rows = size(auto,1);
train_rows = randperm(num_rows,floor(num_rows/2));
train_idx = boolean(zeros(1,num_rows)); % num_rows FALSE hodnot
train_idx(train_rows) = true; % a TRUE pro trenovaci data
```

Na trénovací data se nyní můžeme odkazovat

```
auto_train = auto(train_idx,:);
```

a na testovací

```
auto_test = auto(~train_idx,:); % vybere radky, kde ‘train‘ je FALSE
```

Příklad 1

Pokračování

Natrénujeme lineární model závislosti `mpg` na `horsepower` a spočteme testovací MSE:

```
autofit = fitlm(auto_train, 'mpg~horsepower');
autodif = auto_train.mpg - predict(autofit, auto_test);
mean(autodif.*autodif)
```

Natrénujeme kvadratický model:

```
autofit2 = fitlm(auto_train, 'mpg~horsepower^2');
autodif2 = auto_train.mpg - predict(autofit2, auto_test);
mean(autodif2.*autodif2)
```

Sami udělejte kubický model.

Příklad 1

Pokračování

Pokračujeme pokusem s jiným dělením na testovací a trénovací množinu:

```
train_rowsb = randperm(num_rows,floor(num_rows/2));
train_idxb = boolean(zeros(1,num_rows));
train_idxb(train_rowsb) = true;
autofitb = fitlm(auto(train_idxb,:), 'mpg~horsepower')
autodifb = auto.mpg(~train_idxb) - predict(autofitb,
auto(~train_idxb,:));
mean(autodifb.*autodifb)
```

Porovnejte, zda výsledek bude totožný s

```
mean(autodif.*autodif)
```

Příklad 2

Leave one out cross-validation

Vytvoříme objekt reprezentující LOOCV v Matlabu. K tomu slouží funkce `cvpartition()`:

```
cvp_loocv = cvpartition(num_rows, 'LeaveOut');
```

Dále musíme napsat funkci `cv_auto_linear(xtrain,xtest)` pro vyhodnocení testovací MSE modelu, jenž nejprve natrénujeme na datech z `xtrain`, a pak ověříme na datech z `xtest`. Model bude standardně `'mpg horsepower'`.
Do následující šablony doplňte správné příkazy místo `...`:

```
function mse=cv_auto_linear(xtrain, xtest)
  % Train the model
  mdl = fitlm(...);
  % Compute the test mse
  dv = ... - ...;
  mse = mean(...);
```

Příklad 2

Pokračování

Samotná křížová validace pomocí `crossval()` vrátí vektor všech vypočtených hodnot `mse`:

```
crossval(@cv_auto_linear,auto,'partition',cvp_loocv)
```

lépe tedy

```
mse_loocv = mean(crossval(@cv_auto_linear, auto,
                          'partition', cvp_loocv))
```

Dtto s 10násobnou křížovou validací

```
cvp_10fcv = cvpartition(num_rows, 'KFold', 10);
mse_10fcv = mean(crossval(@cv_auto_linear, auto,
                          'partition', cvp_10fcv))
```