11MAMY – Cvičení 10

Křížová validace
Výběr modelu

Jan Přikryl

ČVUT FD

úterý 12. dubna 2021

verze: 2022-04-12 12:28

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, protože obsahují neplatné hodnoty:

```
auto = readtable(...,...,...)
invalid = isnan(...); % Některé prvky auto.horsepower jsou NaN
auto(invalid,:)=...; % Vymažeme je
```

Příklad 1

Pokračování

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

```
num_rows = size(...,...); % kolik máme záznamů v databázi?
train_rows = randperm(...,...); % viz dokumentace, chceme 1/2 true
train_idx = boolean(...); % ‘num_rows‘ false hodnot
train_idx(...) = true; % a true pro trénovací data
```

Na trénovací data se nyní můžeme odkazovat

```
auto_train = auto(...);
```

a na testovací

```
auto_test = auto(...); % vybere řádky, kde ‘train‘ je false
```

Příklad 1

Pokračování

Na trénovací sadě natrénujeme lineární model závislosti `mpg` na `horsepower` a spočteme testovací MSE:

```
autofit1 = fitlm(..., ...); % Model na trénovacích datech
autodif1 = ....mpg - ...predict(...); % Rozdíl testovacích předpovědí
automse1 = mean(....*...)
```

Stejně natrénujeme a vyhodnotíme kvadratický model:

```
autofit2 = fitlm(..., ...); % Model na trénovacích datech
autodif2 = ....mpg - ...predict(...); % Rozdíl testovacích předpovědí
automse2 = mean(....*...)
```

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` pro všech K rozdělení množiny dat:

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

Lépe je tedy použít něco jako

```
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))
```

Příklad 3

Hřebenová regrese

Použijeme sadu statistik z baseballu. První sloupec v `islr_hitters.csv` jsou jména hráčů, proto `'ReadRowNames'`. U platů občas není uveden plat, je tam `'NA'`. Tyto hodnoty je třeba převést na NaN, proto `'TreatAsEmpty'`

```
hitters = readtable('islr_hitters.csv', 'ReadRowNames', true,
                    'TreatAsEmpty', 'NA');
```

Informace o datech

```
summary(hitters)
size(hitters)
```

Příklad 1

Hřebenová regrese – pokračování

Zjistíme, kolik tam máme hráčů bez uvedeného platu a odstraníme všechny neúplné záznamy

```
sum(isnan(hitters.Salary))
missing = ismissing(hitters);
hitters2 = hitters(~any(missing,2),:);
size(hitters2)
sum(isnan(hitters2.Salary))
```

Bohužel, Matlab neumí zpracovávat hřebenovou regresí data, uložená v tabulce, vstupem funkce `ridge()` je matice regresorů a vektor odezev. Provedeme proto úpravu formátu použitých dat.

Příklad 3

Hřebenová regrese – převod na numerické hodnoty

Vše číselné, tedy `League`, `NewLeague` a `Division` musíme konvertovat na čísla:

```
hitters2.League = double(categorical(hitters2.League))-1;
hitters2.Division = double(categorical(hitters2.Division))-1;
hitters2.NewLeague = double(categorical(hitters2.NewLeague))-1;
```

Hřebenová regrese potřebuje vektor výstupu a matici vstupu, ve vstupech nesmí být plat:

```
X = table2array(hitters2);
X(:,end-1) = []; % Smazat platy
y = hitters2.Salary;
```

Příklad 3

Hřebenová regrese – vlastní idetifikace modelů

Vygenerujeme vektor 100 možných hodnot λ od 10^{10} do 10^{-2}

```
rr.lambda = 10.^linspace(10,-2,100);
```

A identifikujeme 100 modelů s danými hodnotami regularizačního koeficientu a absolutním členem

```
rr.coefs = ridge(y,X,rr.lambda,0); % Chceme absolutni clen
```

Čtyřicátá lambda a suma čtverců koeficientů

```
rr.lambda(40)
rr.coefs(:,40)
sqrt(sum(rr.coefs(:,40).^2))
```

Šedesátou a osmdesátou lambdu si udělejte sami. **Q:** Co koeficienty?

Příklad 3

Hřebenová regrese – trénovací a testovací sada

Rozdělení na trénovací a testovací sadu opět 50-50 a pomocí logického indexování

```
rng(2); % Aby nám to vyšlo všem stejně
num_rows = length(y); % Kolik je dat
train_idx = randsample(num_rows, floor(num_rows/2)); % Náhodně 50%
X_train = X(train_idx,:); % Trénovací prediktory
y_train = y(train_idx); % Trénovací výstupy
X_test = X(~train_idx,:); % Testovací prediktory
y_test = y(~train_idx); % Testovací výstupy
```

A proložíme pro nějaké tři hodnoty λ model trénovacími daty:

```
mdl.lambda = [10^10, 4, 0.001];
mdl.coefs = ridge(y_train, X_train, 0); % Zase včetně abs. členu
```

Příklad 3

Hřebenová regrese – predikce na testovací sadě

Jak budou vypadat testové predikce modelu pro dané $\lambda \in$ `mdl.lambda`?

Data v `mdl.coefs` jsou matice po sloupcích uložených koeficientů. První koeficient je absolutní člen, ten násobíme jedničkou, pokud jsou tedy koeficienty ve sloupcovém vektoru β a data v matici $[\mathbf{1}\ \mathbf{X}]$ jsou jsou orientována po řádcích, bude výstup pro jednotlivá pozorování prostě

$$\hat{\mathbf{y}} = [\mathbf{1}\ \mathbf{X}] \cdot \beta.$$

Výstup je opět sloupcový vektor.

Příklad 3

Hřebenová regrese – predikce na testovací sadě

Rozšíříme `X_test`:

```
X_test1 = [ ones(size(X_test)) X_test ];
```

a

```
y_pred_1 = X_testr * mdl.coefs(:,1);
y_pred_2 = X_testr * mdl.coefs(:,2);
y_pred_3 = X_testr * mdl.coefs(:,3);
```

MSE těch predikcí

```
mset_1 = mean((y_pred_1-y_test).^2)
mset_2 = mean((y_pred_2-y_test).^2)
mset_3 = mean((y_pred_3-y_test).^2)
```

Na doma: křížová validace.