ML Part 2 tutorial

Dimensionality reduction & cross-validation

Jérôme Dockès & Nikhil Bhagwat

QLS course 2021-07-30

Problem setting

$$Y = f(X) + E \tag{1}$$

- $Y \in \mathbb{R}$: output (a.k.a. target, dependent variable) to predict
- $X \in \mathbb{R}^p$: features (a.k.a. inputs, regressors, descriptors, independent variables)
- $E \in \mathbb{R}$: unmodelled noise
- f: the function we try to approximate

Parameter estimation a.k.a. model fitting

Minimize a sum of:

- the empirical risk: error on training data
- a regularization term

Example: logistic regression

$$\underset{\boldsymbol{\beta}, \beta_0}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 + C \sum_{i=1}^{n} \log(\exp(-y_i\,(\mathbf{X}_i^T\,\boldsymbol{\beta} + \beta_0)) + 1) \qquad (2)$$

- $\boldsymbol{\beta}, \beta_0$: parameters to be *estimated*
- C: hyperparameter, *chosen* prior to learning (controls amount of regularization)

`sklearn.linear_model.LogisticRegression`

scikit-learn "estimator API": `fit; predict`

```
estimator = LogisticRegression(C=1)
estimator.fit(X_train, y_train)
predictions = estimator.predict(X_test)
```

https://scikit-learn.org/stable/getting_started.html
sklearn.linear_model.LogisticRegression

Evaluating performance with sklearn.metrics

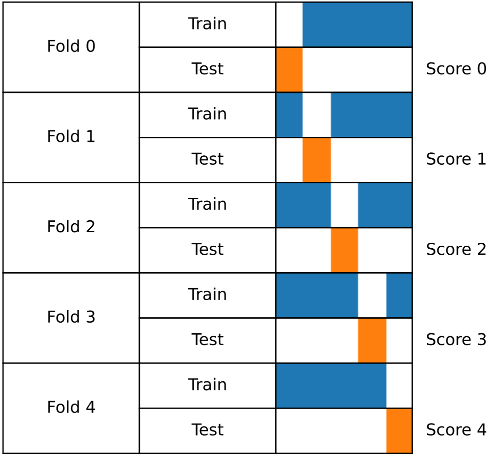
```
estimator = LogisticRegression(C=1)
estimator.fit(X_train, y_train)
predictions = estimator.predict(X_test)

accuracy = metrics.accuracy_score(y_test, predictions)
```

https://scikit-learn.org/stable/getting_started.html
sklearn.linear_model.LogisticRegression
sklearn.metrics more info on model evaluation

ex_01_fit_predict.py

Cross-validation



`scikit-learn.org/stable/modules/cross_validation.html`
`sklearn.model_selection.cross_validate`
`ex_02_cross_validate.py`

Dataset transformations

Typical pipeline

Example: for autism prediction with fMRI from ML part 1

scikit-learn "transformer API": `fit; transform`

```
transformer = StandardScaler()
transformer.fit(X_train)
transformed_X = transformer.transform(X_train)
```

can also be written:

```
transformer = StandardScaler()
transformed_X = transformer.fit_transform(X_train)
```

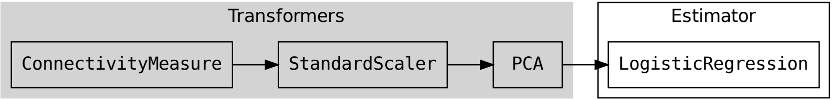
sklearn.preprocessing.StandardScaler
scikit-learn "getting started"
scikit-learn "user guide"

`ex_03_transformer.py`

scikit-learn "transformer API": `fit; transform`

```
transformer = StandardScaler()
transformed_X = transformer.fit_transform(X_train)

transformed_X_test = transformer.transform(X_test)
```



`sklearn.preprocessing.StandardScaler`
scikit-learn "getting started"
scikit-learn "user guide"

Example: `preprocessing.StandardScaler`

`fit:`
Compute mean and standard deviation of each column

`transform:`
Subtract mean and divide by standard deviation

`sklearn.preprocessing.StandardScaler`

Example: feature_selection.SelectKBest

fit:

- compute ANOVA or correlation for each column of X
- Remember the indices of the k columns with highest scores

transform:

- Index input to keep only the k selected columns

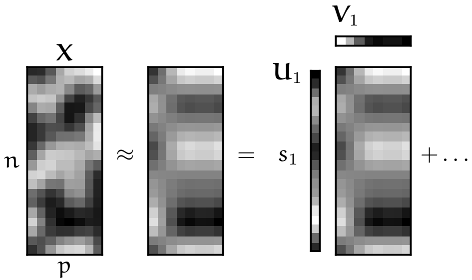
sklearn.feature_selection.SelectKBest
https://scikit-learn.org/stable/modules/feature_selection.html
ex_04_feature_selection.py

Example: `decomposition.PCA`

`fit:`

- Compute Singular Value Decomposition of $\mathbf{X}$

$$\mathbf{X} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\top} \tag{3}$$



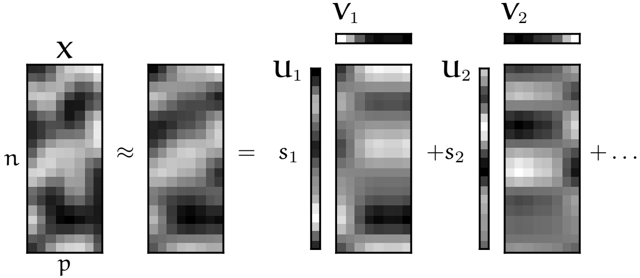
Explained variance: 0.53

Example: `decomposition.PCA`

`fit:`

- Compute Singular Value Decomposition of X

$$X = U\,S\,V^\top \tag{4}$$



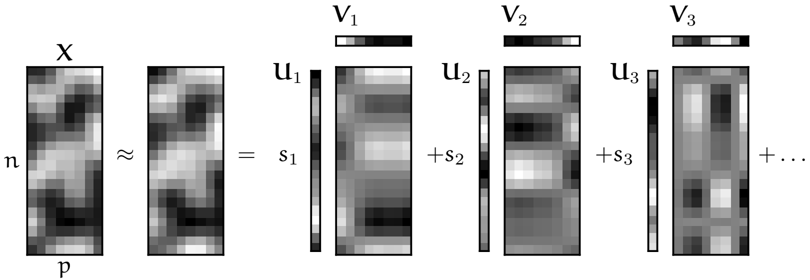
Explained variance: 0.84

Example: `decomposition.PCA`

`fit:`

- Compute Singular Value Decomposition of X

$$X = U S V^{\top} \tag{5}$$



Explained variance: 0.97

Example: `decomposition.PCA`

`fit:`

- Compute Singular Value Decomposition of $\mathbf{X}$

$$\mathbf{X} = \mathbf{U}\, \mathbf{S}\, \mathbf{V}^{\mathsf{T}}$$

- store $\mathbf{V}$

`transform:`

Compute projection on column space of $\mathbf{V}$: simply multiply by $\mathbf{V}^{\mathsf{T}}$

Notes

- `fit_transform`: simply return $\mathbf{U}\, \mathbf{S}$
- $\mathbf{V}^{\mathsf{T}}$ is the 'components_' attribute of a fitted 'PCA' instance

[sklearn.decomposition.PCA]

Chaining transformations

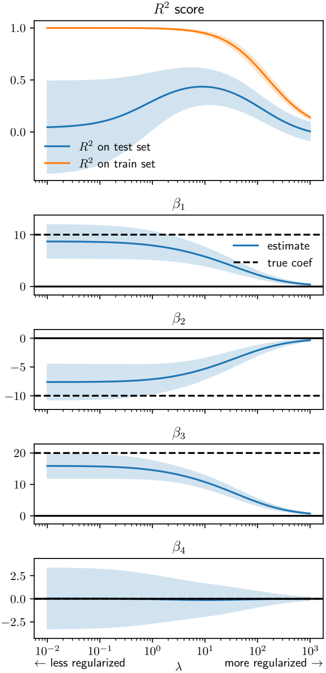
Use `sklearn.pipeline.Pipeline` or `sklearn.pipeline.make_pipeline`:

```
pipe = make_pipeline(
    standardizer, dim_reductor, estimator
)
pipe.fit(X, y)
```

Example:

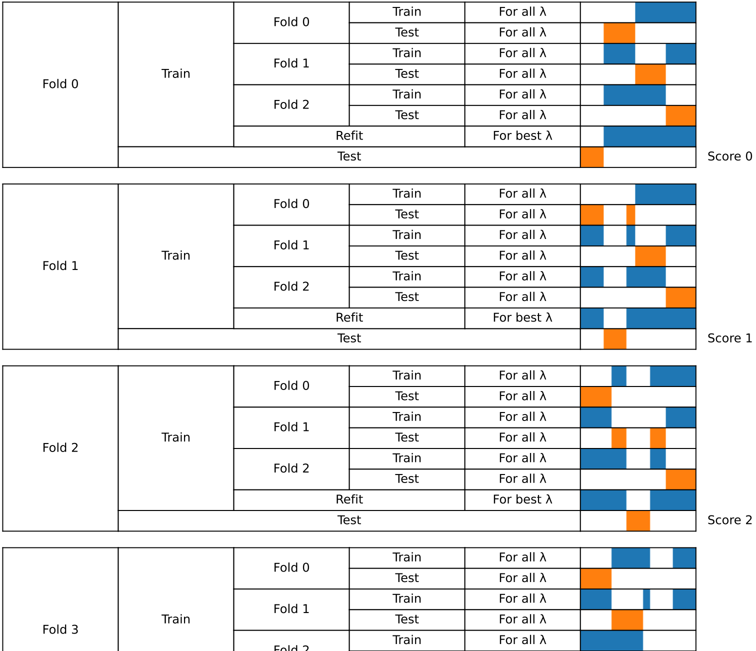
```
make_pipeline(
    StandardScaling(), PCA(), LogisticRegression()
)
```

$$\mathrm{Var}(\hat{\beta}_i) = \mathbb{E}(\hat{\beta}_i - \mathbb{E}(\hat{\beta}_i))^2$$



$$\mathrm{Bias}(\hat{\beta}_i) = \mathbb{E}(\hat{\beta}_i) - \beta_i$$

Nested cross-validation

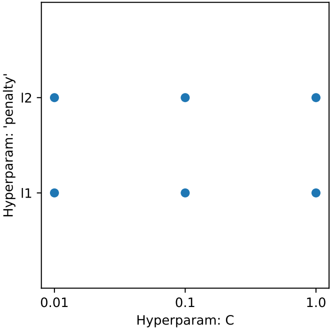


Implementing nested CV

`ex_05_nested_cross_validation.py`

Cross-validation and hyperparameter selection in scikit-learn

- `sklearn.pipeline.Pipeline` or `sklearn.pipeline.make_pipeline`
- `sklearn.model_selection.GridSearchCV`
- `sklearn.model_selection.cross_validate`
- use `*CV` estimators! `RidgeCV`, `LogisticRegressionCV`, ...



Cross-validation pitfalls

- fitting part of the pipeline on the whole data: use `Pipeline`
- ignoring some dependencies in the data: use the appropriate cv iterator: https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators
- good cv scores on one dataset do not guarantee generalization to new data