CS-EJ3211 Machine Learning with Python

Session 3 - Model selection and validation

Shamsi Abdurakhmanova

Aalto University
FITech

10.02.22

Machine Learning

The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.

"Deep Learning" I. Goodfellow.

Machine Learning workflow

- Data (features & labels), Model, Loss
- Train (fit) model using all data (training set)
- Estimate how well model fit the training set (accuracy, MSE, MAE)

Does the model which fits best to training data will have good generalization ability?

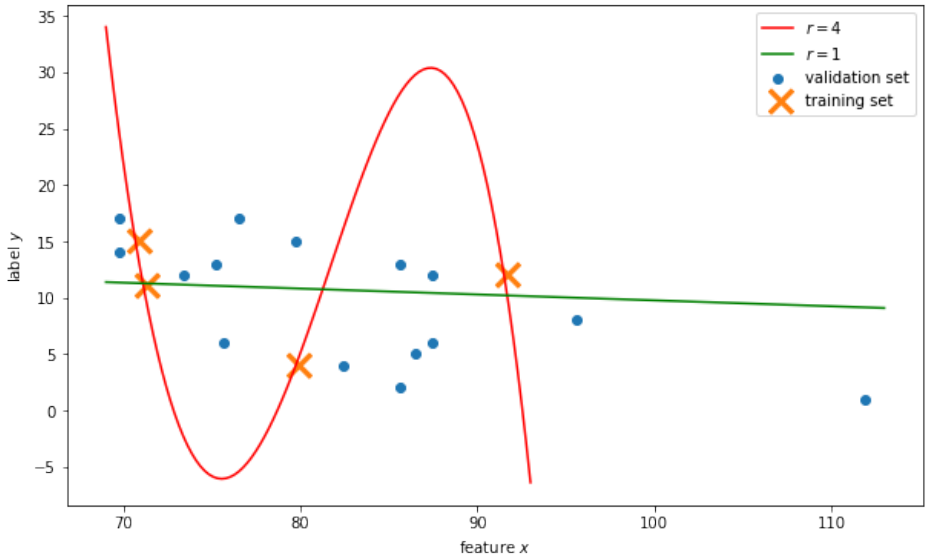
Machine Learning workflow

- Data (features & labels), Model, Loss
- Train (fit) model using all data (training set)
- Estimate how well model fit the training set (accuracy, MSE, MAE)

Does the model which fits best to training data will have good generalization ability?

Usually, model performs worse on unseen data (overfitting) $\rightarrow$ Need to choose model that fit unseen "new" data well.

Overfitting on training data



- Data: how representative is training set?
- Hypothesis: how complex is the model?

Ideally: n.o. data points $\gg$ n.o. model's parameters

Machine Learning workflow - updated

- Split Data into **training** and **validation** sets
- Train (fit) model using ONLY training set
- Estimate how well model fit validation set (accuracy, MSE, MAE)

Computed validation loss/ metrics is our estimate on how model will generalize to unseen data.

Machine Learning workflow - updated

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Split Data into training and validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# model
reg = LinearRegression()

# fit on training set
reg.fit(X_train, y_train)

# compute predictions
ypred_train = reg.predict(X_train)
ypred_val = reg.predict(X_val)

# compute mse
error_train = mean_squared_error(y_train, ypred_train)
error_val = mean_squared_error(y_val, ypred_val)
```

Hyperparameters tuning

Hyperparameters: degree of polynomial, number of units in ANN, parameter ε in Huber Loss, parameter α in Lasso, ... etc.

- Split Data into **training** and **validation** sets
- Fit models with different hyperparameter values to training set
- Estimate how well models fit validation set
- Select the model which fits best to validation set

Does the model which fits best to validation set will have good generalization ability?

Hyperparameters tuning

Hyperparameters: degree of polynomial, number of units in ANN, parameter ε in Huber Loss, parameter α in Lasso, ... etc.

- Split Data into **training** and **validation** sets
- Fit models with different hyperparameter values to training set
- Estimate how well models fit validation set
- Select the model which fits best to validation set

Does the model which fits best to validation set will have good generalization ability?

The model is selected based on validation set will overfit validation set $\rightarrow$ Use another validation (test) set for final evaluation.

Hyperparameters tuning - updated

- Split Data into **Training**, **Validation** and **Test** sets
- Fit models with different hyperparameter values to **Training** set
- Select the model which smallest **Validation** error
- Re-train chosen model on **Training** + **Validation** set
- Estimate model performance on **Test** set

Summary

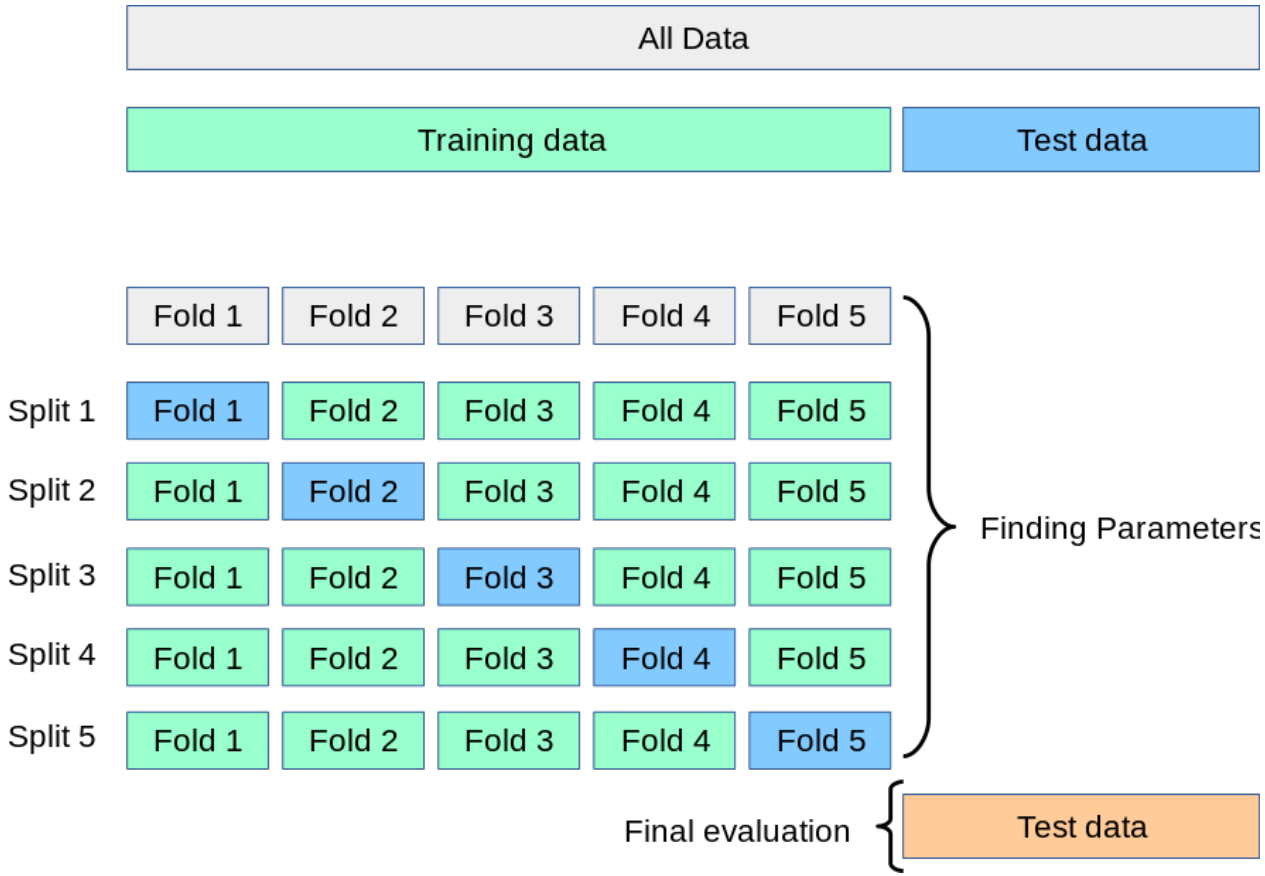
One model:

Data $\rightarrow$ Training, Test

Compare several models, hyperparameter tuning:

Data $\rightarrow$ Training, Validation, Test

Cross -Validation



Cross -Validation

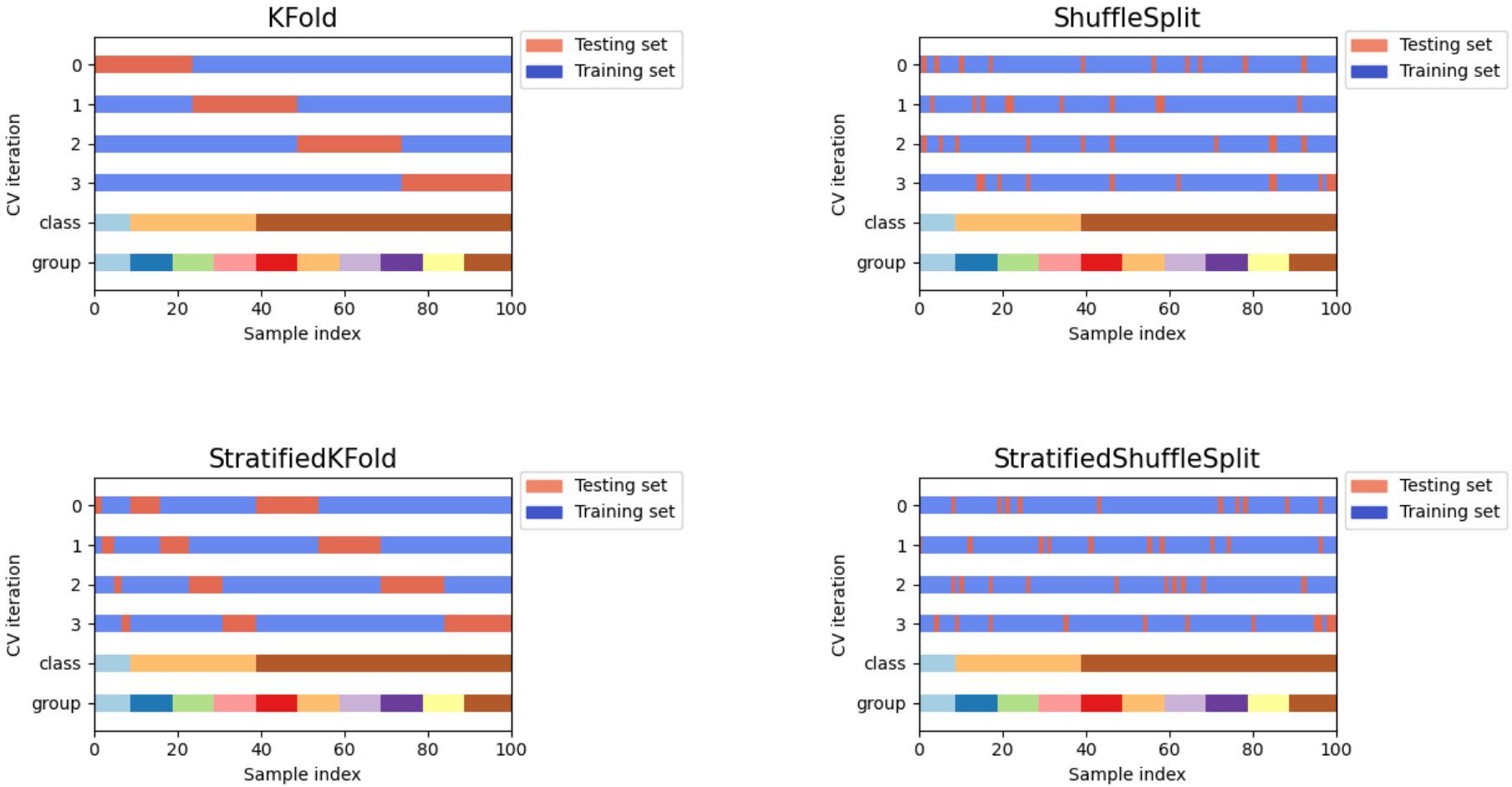


Figure: CV with sklearn

Student task. 5-Fold Cross Validation.

```
# For loop, iterate hyperparamter values:
    ...
    # For loop, iterate indices with kf object:

        # Create a model
        # Fit the model on the current training set

        # Calculate the predicted labels of the current training set
        # Calculate the predicted labels of the current val set

        # Add the training error to the list of errors
        # Add the val error to the list of errors

    # Compute the mean of training errors across splits
    # Compute the mean of validation errors across splits
```

GridSearchCV

```
# data
iris = datasets.load_iris()

# hyperparameters
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}

# cross-validation
kf = KFold(n_splits=6, shuffle=True)

# model
svc = svm.SVC()

# grid search
clf = GridSearchCV(svc, parameters, cv=kf)
clf.fit(iris.data, iris.target)
```

Penalized Optimization.

Ridge Regression:

$$\mathcal{E}(\mathbf{w}, w_0) = (1/m_t) \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathbb{X}^{(t)}} \left(y^{(i)} - w_0 - \mathbf{w}^T \mathbf{x}^{(i)}\right)^2 + \alpha \|\mathbf{w}\|_2^2. \quad (1)$$

Lasso Regression:

$$\mathcal{E}(\mathbf{w}, w_0) = (1/m_t) \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathbb{X}^{(t)}} \left(y^{(i)} - w_0 - \mathbf{w}^T \mathbf{x}^{(i)}\right)^2 + \alpha \|\mathbf{w}\|_1. \quad (2)$$