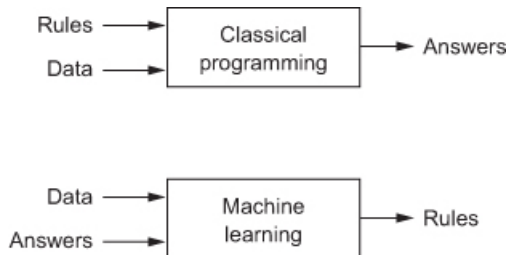


Intro to Machine Learning

Elliott Ash

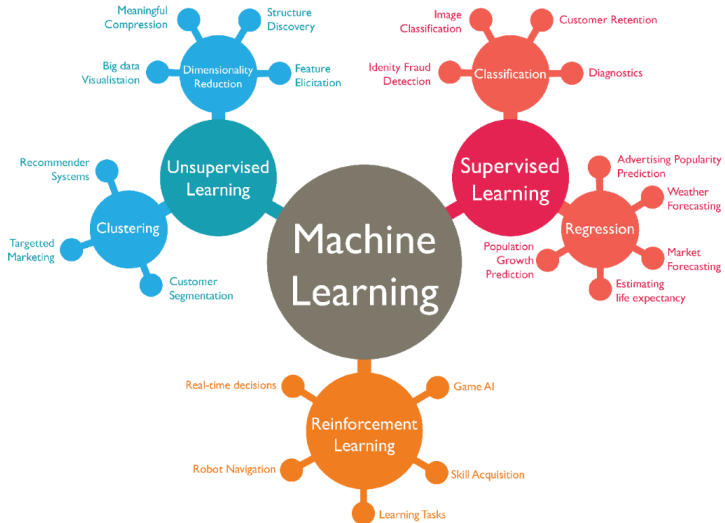
Text Data Course, Bocconi 2018

What is machine learning?



- ▶ In classical computer programming, humans input the rules and the data, and the computer provides answers.
- ▶ In machine learning, humans input the data and the answers, and the computer learns the rules.

The Machine Learning Landscape



A Machine Learning Project, End-to-End

Aurelien Geron, *Hands-on machine learning with Scikit-Learn & TensorFlow*, Chapter 2:

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Our First Data Set

```
import pandas as pd
df1 = pd.read_csv('death-penalty-cases.csv')

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(stop_words='english',
                             max_features=4)
X = vectorizer.fit_transform(df1['snippet'])
X

X = X.todense()
X = X / X.sum(axis=1) # counts to frequencies
words = vectorizer.get_feature_names()
for i, word in enumerate(words):
    column = X[:, i]
    df1['x_' + word] = column
df1.head()
```


Inspecting Data

```
import numpy as np
df1[ 'logcites' ] = np.log(1+df1[ 'citeCount' ])

features = [ 'x_'+x for x in words ]
keepcols = [ 'logcites' ] + features
df1 = df1[ keepcols ]

corr_matrix = df1.corr()
corr_matrix[ 'logcites' ].sort_values(ascending=False)

from pandas.plotting import scatter_matrix
scatter_matrix(df1)

df1.plot(kind='scatter', x='x_death', y='logcites',
```

Select a performance measure

- ▶ A typical performance measure for regression problems is Root Mean Squared Error (RMSE):

$$\text{RMSE}(X, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(x_i) - y_i)^2}$$

- ▶ m , the number of rows/observations
 - ▶ X , the feature set, with row x_i
 - ▶ Y , the outcome, with item y_i
 - ▶ $h(x_i)$ the model prediction (hypothesis)
- ▶ In econometrics, we are familiar with RMSE because that is the cost function that motivates the OLS estimator.
 - ▶ it corresponds to the Euclidian norm or L2 norm, notated as $\|\cdot\|_2$

Other cost functions

- ▶ We will see that in machine learning, while RMSE is a good baseline, other cost functions are sometimes used.
 - ▶ for regression tasks, I personally like R^2 .
- ▶ For example, Mean Absolute Error:

$$\text{MAE}(X, h) = \frac{1}{m} \sum_{i=1}^m |h(x_i) - y_i|$$

which corresponds to the L1 norm or quantile regression.

- ▶ More generally, the k -norm for a vector v is

$$\|v\| = \left(\sum_{i=1}^m |v_i|^k \right)^{\frac{1}{k}}$$

where a higher norm index focuses on large values and neglects small ones.

- ▶ e.g., the L2 norm is more sensitive to outliers than the L1 norm.
- ▶ The L0 norm gives the number of non-zero elements in the vector, while the $L-\infty$ norm gives the maximum absolute value in the vector.

Create a Test Set

```
from sklearn.model_selection import train_test_split  
train_set, test_set = train_test_split(df1, test_size=0.2)
```

Our first machine learning model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
Xtrain = train[features]
Ytrain = train['logcites']
lin_reg.fit(Xtrain, Ytrain)
lin_reg.coef_
```

In-Sample Performance

```
from sklearn.metrics import mean_squared_error  
Ytrain_pred = lin_reg.predict(Xtrain)  
train_mse = mean_squared_error(Ytrain, Ytrain_pred)  
train_mse
```

Out-of-Sample Performance

```
Xtest = test[features]  
Ytest = test['logcites']  
Ytest_pred = lin_reg.predict(Xtest)  
test_mse = mean_squared_error(Ytest, Ytest_pred)  
test_mse
```

Data Prep for Machine Learning (1)

- ▶ Not too different from data prep for econometrics
 - ▶ See Geron Chapter 2, pp. 61-68 for pandas and sklearn syntax.
- ▶ Missing values:

```
judge.fillna(0,inplace=True)  
from sklearn.preprocessing import Imputer
```

- ▶ Feature scaling
 - ▶ This is often necessary for ML models to work (e.g. lasso/ridge require standardizing)

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X = scaler.fit_transform(df1)  
df1 = pd.DataFrame(X,columns=df1.columns)
```

Data Prep for Machine Learning (2)

- ▶ Categorical variables:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
judge_fes = enc.fit_transform(judge.values.reshape(-1,1))
```

- ▶ Custom data prep pipelines
 - ▶ use for multiple steps, e.g. fill missing, then make dummies, then scale, etc.

```
from sklearn.pipeline import Pipeline
```

Scikit-Learn Design Principles

► **Consistency:**

- *Estimator*: An object that can estimate parameters. Estimation is performed by `fit()` method. Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer*: An object that transforms a data set. Transformation is performed by the `transform()` method. The convenience method `fit_transform()` both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set. The `predict()` method forms the predictions. It also has a `score()` method that measures the quality of the predictions given a test set.

- **Inspection**: Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)

- **Non-proliferation of classes**: Use native Python data types; existing building blocks are used as much as possible.

- **Sensible defaults**: Provides reasonable default values for hyperparameters – easy to get a good baseline up and running.

Cross-Validation

```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor()

from sklearn.model_selection import cross_val_score
scores = cross_val_score(forest_reg ,
                          df1[features] ,
                          df1['anycites'] ,

print(scores.mean() , scores.std())
```

Grid Search

```
from sklearn.model_selection import GridSearchCV
param_grid = {'n_estimators': [3, 10, 30],
              'max_features': [2, 4],
              'bootstrap': [True, False]}

grid_search = GridSearchCV(forest_reg ,
                           param_grid ,
                           cv=3)
grid_search.fit(df1[features], df1['logcites'])

grid_search.best_params_
grid_search.best_score_
```

Saving and Loading Models

```
from sklearn.externals import joblib  
joblib.dump(forest_reg, 'forest_reg.pkl')  
forest_reg = joblib.load('forest_reg.pkl')
```