scikit-learn

C. Travis Johnson
UNKNOWN
Mines Linux Users Group

Introduction

Machine Learning - What is it really?

- Goal: Extract Knowledge from Data
- Sometimes called predictive analysis or statistical learning
- Given a large matrix of observations X, fit a function $f(x)$ that maps observation x to a response variable y

Machine Learning - What is it really?

- Goal: Extract Knowledge from Data
- Sometimes called predictive analysis or statistical learning
- Given a large matrix of observations X, fit a function $f(x)$ that maps observation x to a response variable y

Machine Learning - What is it really?

- Goal: Extract Knowledge from Data
- Sometimes called predictive analysis or statistical learning
- Given a large matrix of observations X, fit a function $f(x)$ that maps observation x to a response variable y

Important Terms

Classifiers Algorithms that learn functions to map observations to a *discrete* response. E.g., is this tumor malignant or benign? Is this email spam or not?

Regressors Algorithms that learn functions to map observations to a *continuous* response. E.g., how much should this house cost?

Underfitting The learned function is too simple. "We barely studied for the exam."

Overfitting The learned function is too complex. "We memorized all the practice problems, but don't understand the material."

Generalization How well does the learned function extend to new observations?

Scikit-Learn: Machine Learning in Python

- Provides many machine learning algorithms with a common `Estimator` interface
- Built in helpers for common ML tasks (e.g., `metrics`, `preprocessing`)
- Easily combine algorithms to make a complex pipeline[1]
- Relies heavily on `numpy` and `scipy`, often used with `pandas`

[1] Sound familiar?

Scikit-Learn: Machine Learning in Python

- Provides many machine learning algorithms with a common `Estimator` interface
- Built in helpers for common ML tasks (e.g., `metrics`, `preprocessing`)
- Easily combine algorithms to make a complex pipeline[1]
- Relies heavily on `numpy` and `scipy`, often used with `pandas`

[1] Sound familiar?

Scikit-Learn: Machine Learning in Python

- Provides many machine learning algorithms with a common `Estimator` interface
- Built in helpers for common ML tasks (e.g., `metrics`, `preprocessing`)
- Easily combine algorithms to make a complex pipeline[1]
- Relies heavily on `numpy` and `scipy`, often used with `pandas`

[1] Sound familiar?

Scikit-Learn: Machine Learning in Python

- Provides many machine learning algorithms with a common `Estimator` interface
- Built in helpers for common ML tasks (e.g., `metrics`, `preprocessing`)
- Easily combine algorithms to make a complex pipeline[1]
- Relies heavily on `numpy` and `scipy`, often used with `pandas`

[1] Sound familiar?

Supervised Learning

Learning to Predict Breast Cancer

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

cancer = load_breast_cancer()    # Get some data
X_train, X_test, y_train, y_test = train_test_split(
        cancer.data, cancer.target,
        stratify=cancer.target, random_state=1337)

tree = DecisionTreeClassifier(random_state=7331)
tree.fit(X_train, y_train)  # Learn a Decision Function
```

Evaluating Accuracy of a Model

```
# How well did we do?
train_acc = tree.score(X_train, y_train)
test_acc = tree.score(X_test, y_test)
print("Training Accuracy: {:.3f}".format(train_acc))
print("Testing Accuracy: {:.3f}".format(test_acc))
# Training Accuracy: 1.000
# Testing Accuracy: 0.923
```

Other Supervised Learning Models

- Decision trees are a common first step, because they're easy to interpret and don't require much `preprocessing`
- Decision trees are prone to overfitting, so a good improvement is the `RandomForest`
- Support Vector Machines, Logistic/Linear Regression, and Artificial Neural Networks are commonly the first algorithms studied
- See the `scikit-learn` documentation for a comprehensive guide of available algorithms

Other Supervised Learning Models

- Decision trees are a common first step, because they're easy to interpret and don't require much `preprocessing`
- Decision trees are prone to overfitting, so a good improvement is the `RandomForest`
- Support Vector Machines, Logistic/Linear Regression, and Artificial Neural Networks are commonly the first algorithms studied
- See the `scikit-learn` documentation for a comprehensive guide of available algorithms

Other Supervised Learning Models

- Decision trees are a common first step, because they're easy to interpret and don't require much `preprocessing`
- Decision trees are prone to overfitting, so a good improvement is the `RandomForest`
- Support Vector Machines, Logistic/Linear Regression, and Artificial Neural Networks are commonly the first algorithms studied
- See the `scikit-learn` documentation for a comprehensive guide of available algorithms

Other Supervised Learning Models

- Decision trees are a common first step, because they're easy to interpret and don't require much `preprocessing`
- Decision trees are prone to overfitting, so a good improvement is the `RandomForest`
- Support Vector Machines, Logistic/Linear Regression, and Artificial Neural Networks are commonly the first algorithms studied
- See the `scikit-learn` documentation for a comprehensive guide of available algorithms

Unsupervised Learning

Distinction from Supervised Learning

Supervised Learning You tell the model what the correct answers are for training examples.

Unsupervised Learning You ask the model to extract information from a dataset.

Unsupervised Clustering Partition data into similar groups. Example: K-Means Clustering

Unsupervised Transformations Create new representations of data. Example: Principal Component Analysis

Model Evaluation and Improvement

Choice of Evaluation Metric

- Accuracy is not always the best metric for your system
- Plenty of others exist, pick the best for your business costs
- Look in the `sklearn.metrics` module for alternatives
- You can also use your own evaluation function!

Choice of Evaluation Metric

- Accuracy is not always the best metric for your system
- Plenty of others exist, pick the best for your business costs
- Look in the `sklearn.metrics` module for alternatives
- You can also use your own evaluation function!

Choice of Evaluation Metric

- Accuracy is not always the best metric for your system
- Plenty of others exist, pick the best for your business costs
- Look in the `sklearn.metrics` module for alternatives
- You can also use your own evaluation function!

Choice of Evaluation Metric

- Accuracy is not always the best metric for your system
- Plenty of others exist, pick the best for your business costs
- Look in the `sklearn.metrics` module for alternatives
- You can also use your own evaluation function!

Cross Validation

Grid Search with Cross Validation

Pipelines

Pipelines

Use `Pipeline` to combine multiple estimators into a single estimator. Two conveniences:

1. Convenience: You only have to call fit and predict once on your data to fit a whole sequence of estimators.
2. Joint parameter selection: You can grid search over parameters of all estimators in the pipeline at once.

A Simple Pipeline

```
>>> from sklearn.pipeline import Pipeline
>>> from sklearn.svm import SVC
>>> from sklearn.decomposition import PCA
>>> estimators = [('reduce_dim', PCA()), ('clf', SVC())]
>>> pipe = Pipeline(estimators)
>>> pipe
Pipeline(steps=[('reduce_dim', PCA(copy=True, iterated_power='auto',
    n_components=None, random_state=None, svd_solver='auto', tol=0.0,
    whiten=False)), ('clf', SVC(C=1.0, cache_size=200, class_weight=None,
        coef0=0.0, decision_function_shape=None, degree=3, gamma='auto',
        kernel='rbf', max_iter=-1, probability=False, random_state=None,
        shrinking=True, tol=0.001, verbose=False))])
```

Grid Search - Tuning a Complex Pipeline

```
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
pipe = make_pipeline(PCA(), SVC())
params = dict(pca__n_components=[2, 5, 10],
        svc__C=[0.1, 10, 100])
grid = GridSearchCV(pipe, param_grid=params)
# Next, call grid.fit on some training data
# This will use cross validation to estimation performance using each
# combination of parameters for pipeline in params dict

# With fitted model
print(grid.best_params_)
```

Questions?

Copyright Notice

Colorado School of Mines
Linux Users Group