Direct Use of Nonparametric Outlier Detection

Introduction

This example demonstrates nonparametric outlier detection using kernel density estimation (KDE). The algorithm learns a nonparametric probability density function from undamaged baseline data and identifies outliers as points with low probability density.

Data from the **3-story structure** dataset are used to extract AR model features, which are then analyzed using kernel density estimation for damage detection.

Key Concepts:

- **Kernel Density Estimation**: Nonparametric density estimation using various kernel functions
- **Bandwidth Selection**: Automatic methods for optimal smoothing parameter selection
- **Threshold Determination**: Statistical approach using normal distribution fitting
- **Multiple Kernel Functions**: Comparison of different kernel shapes (Gaussian, Epanechnikov, etc.)

References:

Figueiredo, E., Park, G., Figueiras, J., Farrar, C., & Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons using Standard Data Sets. Los Alamos National Laboratory Report: LA-14393.

SHMTools functions used:

- `ar_model_shm`
- `learn_kernel_density_shm`
- `score_kernel_density_shm`
- `roc_shm`
- `epanechnikov_kernel_shm`

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
import sys
from scipy import stats

# Add shmtools to path - handle different execution contexts
current_dir = Path.cwd()
notebook_dir = Path(__file__).parent if '__file__' in globals() else current
```

```
# Try different relative paths to find shmtools
possible_paths = [
    notebook_dir.parent.parent.parent,  # From examples/notebooks/advanced/
    current_dir.parent.parent,          # From examples/notebooks/
    current_dir,                        # From project root
    Path('/Users/eric/repo/shm/shmtools-python')  # Absolute fallback
]

shmtools_found = False
for path in possible_paths:
    if (path / 'shmtools').exists():
        if str(path) not in sys.path:
            sys.path.insert(0, str(path))
        shmtools_found = True
        print(f"Found shmtools at: {path}")
        break

if not shmtools_found:
    print("Warning: Could not find shmtools module")

from shmtools.utils.data_loading import load_3story_data
from shmtools.features.time_series import ar_model_shm
from shmtools.classification.nonparametric import (
    learn_kernel_density_shm,
    score_kernel_density_shm,
    epanechnikov_kernel_shm,
    gaussian_kernel_shm,
    roc_shm
)

# Set up plotting
plt.style.use('default')
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 10

# Set random seed for reproducibility
np.random.seed(42)
```

```
Found shmtools at: /Users/eric/repo/shm/shmtools-python
```

```
/Users/eric/repo/shm/shmtools-python/venv/lib/python3.9/site-packages/urllib
3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1
+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: http
s://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
```

Load data

In [2]:
```
data = load_3story_data()
dataset = data['dataset']
states = data['damage_states']
```

In [3]:
```
time_data = np.zeros((2048, 5, 680))
time_data_states = np.zeros(680)
```

```
for i in range(4):
    start_idx = 2048 * i
    end_idx = 2048 * (i + 1)
    time_data[:, :, i::4] = dataset[start_idx:end_idx, :, :]
    time_data_states[i::4] = states
```

In [4]:
```
N = 400
np.random.seed(42)
idx = np.random.permutation(time_data.shape[2])[:N]
X_data = ar_model_shm(time_data[:, :, idx])[1]
X_states = time_data_states[idx]
```

In [5]:
```
idx = np.isin(X_states, range(1, 10))
X_undamaged = X_data[idx, :]
n_undamaged = X_undamaged.shape[0]
n_train = round(0.8 * n_undamaged)
X_train = X_undamaged[:n_train, :]
X_test = np.vstack([X_undamaged[n_train:, :], X_data[~idx, :]])
n_test = X_test.shape[0]
```

In [6]:
```
n_test_0 = n_undamaged - n_train
```

In [7]:
```
test_labels = np.concatenate([np.zeros(n_test_0), np.ones(n_test - n_test_0)
```

Train a model over the undamaged data

In [8]:
```
kernel_fun = epanechnikov_kernel_shm
H = None
bs_method = 2
d_model = learn_kernel_density_shm(X_train, H, kernel_fun, bs_method)
```

Pick a threshold from the training data

In [9]:
```
likelihoods = score_kernel_density_shm(X_train, d_model)
```

In [10]:
```
model_p = stats.norm.fit(likelihoods)
```

In [11]:
```
confidence = 0.9
threshold = stats.norm.ppf(1 - confidence, model_p[0], model_p[1])
```

Test the detector

In [12]:
```
scores = score_kernel_density_shm(X_test, d_model)
```

In [13]:
```
results = scores <= threshold
```

Report the detector's performance

In [14]:
```
total_err = np.sum(results != test_labels) / n_test
false_positive_err = np.sum(results[:n_test_0] != 0) / n_test_0
false_negative_err = np.sum(results[n_test_0:] != 1) / (n_test - n_test_0)
print(f'\n Total error: {total_err:.2f}\n False Positive rate: {false_positi
```

```
Total error: 0.10
False Positive rate: 0.18
False Negative rate: 0.08
```

In [15]:
```
true_positives, false_positives = roc_shm(scores, test_labels)
```

In [16]:
```
plt.figure()
plt.plot(false_positives, true_positives)
plt.xlabel('falsePositives')
plt.ylabel('truePositives')
plt.title('ROC curve')
plt.show()
```