AMATH 582 Final Project

Tyrone DeSilva

February 21, 2020

Abstract

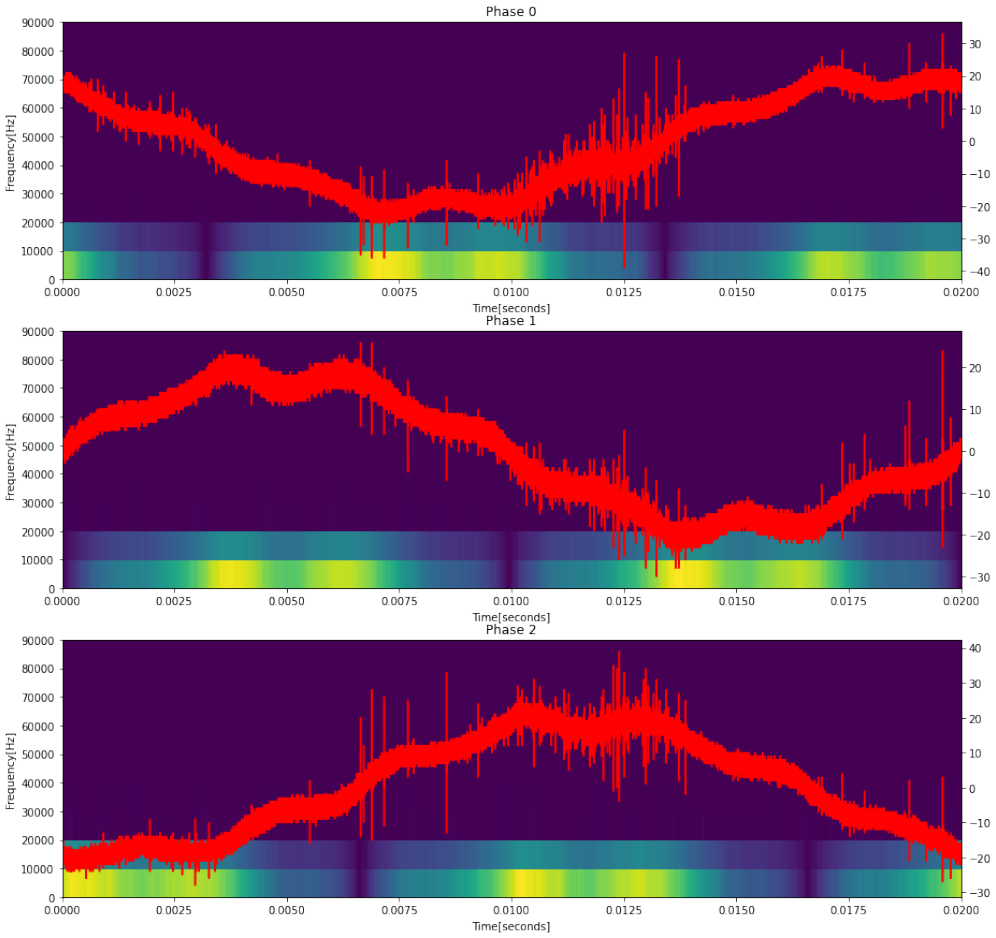

Figure 1: Periodograms for three phases along with original signals.

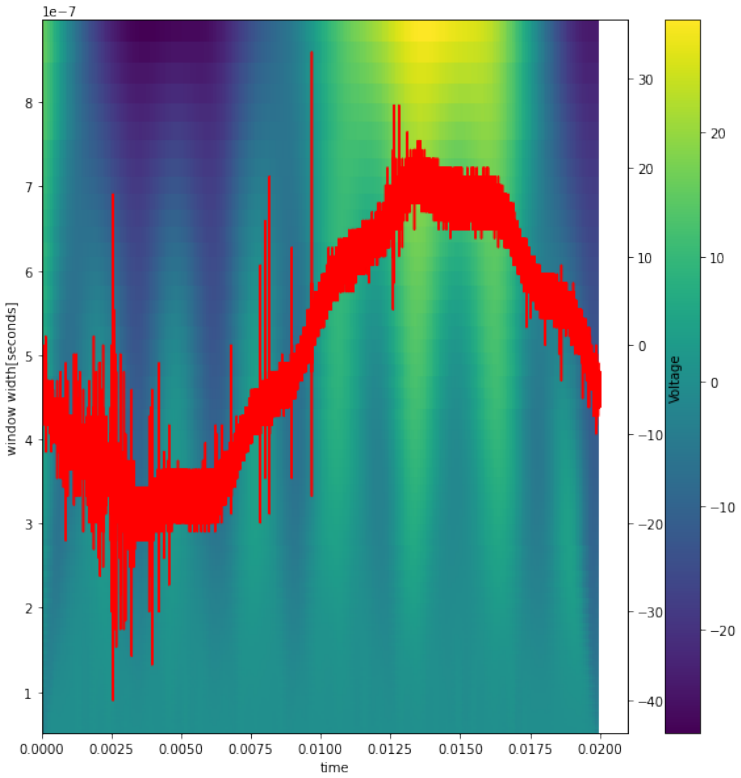

Figure 2: Scalogram with original signal.

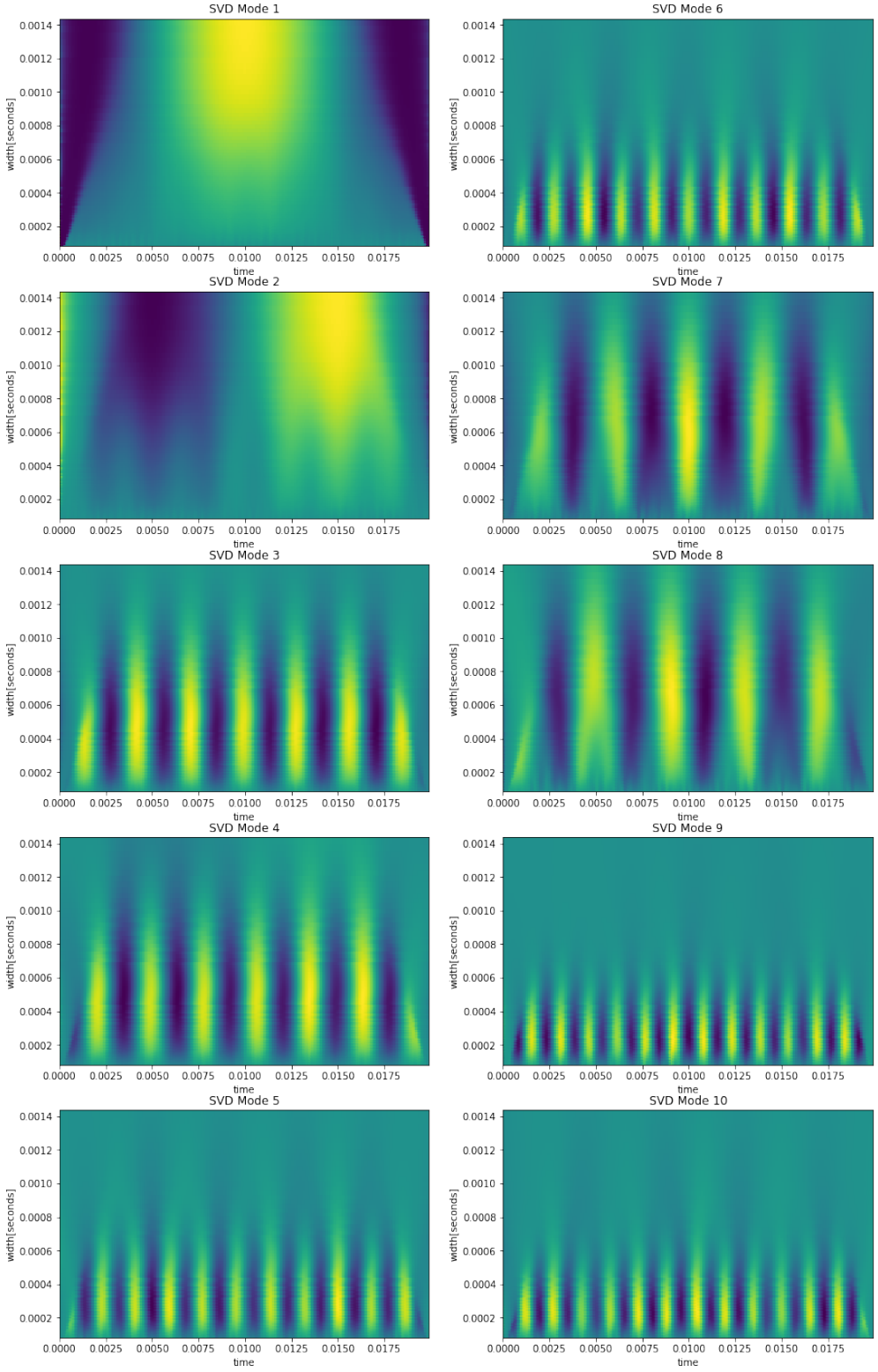

Figure 3: PCA Modes

Appendix A Python Functions

- Placeholder

Appendix B Python Code

B.1 preprocess.py

```
from scipy import signal
import load
import numpy as np
from tqdm import tqdm
import argparse


Fs = 40000000
n = int(Fs * 20e-3)
k = 24


def wavelet(s):
    downsampled = signal.resample(s, n // 2 // 1600)
    widths = [2 ** (j / k) for j in range(1, 101)]
    z = signal.cwt(downsampled, signal.ricker, widths)
    return z


def preprocess(loader, output):
    signals, meta = loader()
    result = np.zeros((int(25000), signals.shape[1]), dtype=np.float32)
    for i in tqdm(range(signals.shape[1])):
        z = wavelet(signals.iloc[:, i])
        z = np.ravel(z)
        result[:, i] = z.astype(np.float32)
    np.save(output, result)


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--train", action="store_true", default=False)
    parser.add_argument("--test", action="store_true", default=False)
    args = parser.parse_args()
    if args.train:
        preprocess(load.load_train, "data/preprocessed/train.npy")
    if args.test:
        preprocess(load.load_test, "data/preprocessed/test.npy")
```

B.2 load.py

```
import pandas as pd
import pyarrow.parquet as pq
import os
```

```
THIS_FILE_DIR = os.path.dirname(os.path.abspath(__file__))
DATA_DIR = os.path.join(THIS_FILE_DIR, "data/vsb-power-line-fault-detection")


def load_train(n_columns: int = None) -> (pd.DataFrame, pd.DataFrame):
    columns = None
    if n_columns:
        columns = [str(i) for i in range(n_columns)]
    train_data = pq.read_pandas(
        os.path.join(DATA_DIR, "train.parquet"), columns=columns
    ).to_pandas()
    train_meta = pd.read_csv(
        os.path.join(DATA_DIR, "metadata_train.csv"),
        index_col="signal_id",
        nrows=n_columns,
    )
    return train_data, train_meta


def load_test(n_columns: int = None) -> (pd.DataFrame, pd.DataFrame):
    columns = None
    if n_columns:
        columns = [str(i) for i in range(n_columns)]
    test_data = pq.read_pandas(
        os.path.join(DATA_DIR, "test.parquet"), columns=columns
    ).to_pandas()
    test_meta = pd.read_csv(
        os.path.join(DATA_DIR, "metadata_test.csv"),
        index_col="signal_id",
        nrows=n_columns,
    )
    return test_data, test_meta
```

B.3 pca_modes.py

```
import numpy as np
import matplotlib.pyplot as plt

data = np.load("../data/preprocessed/train.npy")
means = np.reshape(np.mean(data, axis=1), (-1, 1))
centered = data - means
std = np.reshape(np.std(centered, axis=1), (-1, 1))
scaled = data / std
u, s, vh = np.linalg.svd(scaled, full_matrices=False)
fig, ax = plt.subplots(5, 2, figsize=(15, 25))
widths = [2 ** (j / 24) for j in range(1, 101)]
dt = 20e-3 / 250
y = np.array(widths) * dt
x = np.array([j * dt for j in range(250)])
for j in range(10):
    pos = (j % 5, j // 5)
    a = ax[pos[0]][pos[1]]
    a.pcolormesh(x, y, np.reshape(u[:, j], (100, 250)))
    a.set_title(f"SVD Mode {j + 1}")
```

```
    a.set_xlabel("time")
    a.set_ylabel("width[seconds]")
```

B.4 pipeline.py

```
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_validate
from imblearn.pipeline import make_pipeline
from sklearn.metrics import (
    matthews_corrcoef,
    precision_score,
    recall_score,
    make_scorer,
)
from imblearn.over_sampling import SMOTE
from joblib import dump
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import GradientBoostingClassifier

train_wavelets = np.load("../data/preprocessed/train.npy")
train_meta = pd.read_csv(
    "../data/vsb-power-line-fault-detection/metadata_train.csv", index_col="signal_id",
)

x = train_wavelets.T
y = train_meta.target.values


gb_smote_pipe = make_pipeline(
    StandardScaler(), PCA(n_components=117), SMOTE(), GradientBoostingClassifier(),
)
scores = cross_validate(
    gb_smote_pipe,
    x,
    y,
    cv=5,
    scoring={
        "mcc": make_scorer(matthews_corrcoef),
        "precision": make_scorer(precision_score),
        "recall": make_scorer(recall_score),
    },
    return_train_score=True,
    n_jobs=-1,
)

print(scores)
gb_smote_pipe.fit(x, y)
dump(gb_smote_pipe, "../data/models/gb_smote.joblib")
```

B.5 plots.py

```
from scipy.signal import stft
from scipy import ndimage
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
from load import load_train

train, train_meta = load_train(6)
train.info()
train_meta.info()

Fs = 40000000
n = int(Fs / 10000)
overlap = None

fig, ax = plt.subplots(3, 1, figsize=(15, 15))
for i in range(3):
    f, t, z = stft(train.iloc[:, i].values, fs=Fs, nperseg=n, noverlap=overlap)
    ax[i].pcolormesh(t, f[:10], np.abs(z)[:10, :], vmin=0)
    ax[i].set_xlabel("Time[seconds]")
    ax[i].set_ylabel("Frequency[Hz]")
    ax[i].set_title(f"Phase {i}")
    ax2 = ax[i].twinx()
    ax2.plot([j / Fs for j in range(800000)], train.iloc[:, i], c="r")


downsampled = signal.resample(train.iloc[:, 4], 800000 // 2 // 1600)
k = 24
widths = [2 ** (j / k) for j in range(1, 101)]
z = signal.cwt(downsampled, signal.ricker, widths)
z_filt = z
plt.figure(figsize=(10, 10))
plt.pcolormesh(
    [1600 * j / (Fs / 2) for j in range(250)],
    [1 / (Fs / 2) * (j) for j in widths],
    z_filt,
)
plt.ylabel("window width[seconds]")
plt.xlabel("time")
plt.colorbar()
ax2 = plt.twinx()
ax2.plot([j / Fs for j in range(800000)], train.iloc[:, 4], c="r")
ax2.set_ylabel("Voltage")
```