Homework5_DS-GA1013

March 14, 2021

Name: Zhuoyuan Xu

NetID: zx1137

Due Date: March 14, 2021

Problem 1

(a) Given $x : R \to C$, $T = 1$ and x is continuously differentiable, for $k \neq 0$,

$$\begin{aligned}
\hat{x}[k] &= \int_0^1 x(t)e^{-i2\pi kt}\,dt \\
&= \int_0^1 x(t)\frac{d}{dt}(\frac{e^{-i2\pi kt}}{-i2\pi k})\,dt \\
&= [x(t)\frac{e^{-i2\pi kt}}{-i2\pi k}]_0^1 - \int_0^1 x(t)'\frac{e^{-i2\pi kt}}{-i2\pi k}\,dt \\
&= -\int_0^1 x(t)'\frac{e^{-i2\pi kt}}{-i2\pi k}\,dt
\end{aligned}$$

Also note that $|\int_0^1 f(t)\,dt| \leq \int_0^1 |f(t)|\,dt$

Therefore,

$$\begin{aligned}
|\hat{x}[k]| &= |-\int_0^1 x(t)'\frac{e^{-i2\pi kt}}{-i2\pi k}\,dt| \\
&\leq \int_0^1 |x(t)'\frac{e^{-i2\pi kt}}{-i2\pi k}|\,dt \\
&= \frac{1}{|k|}\int_0^1 |x(t)'\frac{e^{-i2\pi kt}}{-i2\pi}|\,dt
\end{aligned}$$

Note the value of $\frac{e^{-2\pi kt}}{-i2\pi} \leq 1$ because it is a periodic, complex sinusoid with max absolute amplitude smaller than 1, this expression can be further simplified as

$$|\hat{x}[k]| \leq \frac{1}{|k|}\int_0^1 |x(t)'|\,dt$$

and since $\hat{x}$ is continuously differentiable, the integral is finite.

Thus,

$$|\hat{x}[k]| \leq \frac{C_1}{|k|}$$

(b) Given x is twice continuously differentiable, the expression from the previous question can be further written as

$$\begin{aligned}\hat{x}[k] &= -\int_0^1 x(t)'\frac{e^{-i2\pi kt}}{-i2\pi k}\,dt \\ &= -\int_0^1 x(t)'\frac{d}{dt}(\frac{e^{-i2\pi kt}}{4\pi^2k^2})\,dt \\ &= -[x(t)'\frac{e^{-i2\pi kt}}{4\pi^2k^2}]_0^1 + \int_0^1 x(t)''(\frac{e^{-i2\pi kt}}{4\pi^2k^2})\,dt \\ &= \int_0^1 x(t)''(\frac{e^{-i2\pi kt}}{4\pi^2k^2})\,dt\end{aligned}$$

Therefore,

$$\begin{aligned}|\hat{x}[k]| &= |\int_0^1 x(t)''(\frac{e^{-i2\pi kt}}{4\pi^2k^2})\,dt| \\ &\leq \int_0^1 |x(t)''(\frac{e^{-i2\pi kt}}{4\pi^2k^2})|\,dt \\ &\leq \frac{1}{|k|^2}\int_0^1 |x(t)''|\,dt\end{aligned}$$

Since $\hat{x}$ is twice continuously differentiable, the integral is finite.

$$|\hat{x}[k]| \leq \frac{C_2}{|k|^2}$$

Problem 2

(a) Given signal x belonging to the unit interval with form

$$x(t) = a_1 e^{i2\pi k_1 t} + a_2 e^{i2\pi k_2 t}$$

It fits into the bandlimited signal equation, with only $\hat{x}[k_1] = a_1$ and $\hat{x}[k_2] = a_2$ and the cut-off frequency $\frac{k_c}{T} = max\{|k_1|, |k_2|\}$

By the Nyquist-Shannon-Kotelnikov Sampling Theorem, any bandlimited signal with T>0 and cut-off frequency $\frac{k_c}{T}$ can be recovered from N uniformly spaced samples as long as

$$N \geq 2k_c + 1$$

In this case,

$$N \geq 2max\{|k_1|, |k_2|\} + 1$$

(b) If we define $x_{[N]}$ as the vector containing the N samples, we have

$$x_{[N]} = \frac{1}{T} \sum_{k=-k_c}^{k_c} \hat{x}_k \psi_k$$

In this problem we have

$$\begin{aligned} x_N &= \hat{x}[k_1]\psi_{k_1} + \hat{x}[k_2]\psi_{k_2} \\ &= a_1\psi_{k_1} + a_2\psi_{k_2} \end{aligned}$$

Note ψ_{k_1} and ψ_{k_2} are defined as the vectors of discrete complex sinusoids with integer frequencies k_1 and k_2.

It can also be written as

$$\begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_N \end{bmatrix} = \begin{bmatrix} e^{i2\pi k_1 0} & e^{i2\pi k_2 0} \\ e^{\frac{i2\pi k_1}{N}} & e^{\frac{i2\pi k_2}{N}} \\ ... & \\ e^{\frac{i2\pi k_1 (N-1)}{N}} & e^{\frac{i2\pi k_2 (N-1)}{N}} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

(c) To possibly recover the coefficients (amplitudes) from the equations, we should have the number of entries N in x_N greater than or equal to the number of coefficients, which is 2 in this problem, i.e. $N >= 2$.

Meanwhile, ψ_{k_1} and ψ_{k_2} should be linearly independent. To satisfy this condition, $k_1 \neq k_2$ and $(k_2 - k_1) mod N \neq 0$.

The Sampling Theorem from the previous problem determines that

$$N \geq 2max\{|k_1|, |k_2|\} + 1$$

so it is possible for N to be smaller than that of Sampling Theorem because in this situation we only have 2 nonzero terms at frequencies k_1 and k_2. If we ever has any k_1 or k_2 larger than 0.5, then the N we propose can be smaller than that from the Theorem. For example, if we have $k_1 = 1$ and $k_2 = 2$, then the system of equations becomes

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ e^{i\pi} & e^{i2\pi} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

(d) This approach assumes that (1) we know the exact equation of the signal, and (2) the signals are bandlimited, but signal are never exactly bandlimited. Such assumptions makes this approach unrealistic. The estimated Fourier coefficients are not equal to the true coefficients due to other interferences, i.e. aliasing.

Problem 3

(a) Given the bandpass signal x belonging to the unit interval with nonzero Fourier series coefficients between $[k_1, k_2]$, if we can still assume x is bandlimited, then by the Sampling Theorem,

$$N \geq 2k_c + 1$$

in this case, we can set k_c as k_2

(b) If we can assume that $k_2 = k_1 + 2\tilde{k_c}$ where $\tilde{k_c}$ is a positive integer, then all the nonzero Fourier series coefficients are in $[k_1, k_1 + 2\tilde{k_c}]$ with mid point at $k_1 + \tilde{k_c}$.

To recover the signal from the equation system

$$x_{[N]} = \frac{1}{T} \sum_{k=-k_c}^{k_c} \hat{x}_k \psi_k$$

in which k goes from $-k_c$ to k_c, we know in this problem it can be re-written as going from k_1 to $k_1 + 2\tilde{k_c}$. This shows this equation system will have at most $2\tilde{k_c} + 1$ coefficients, and thus

$$N \geq 2\tilde{k_c} + 1$$

(c) In this problem, if we want to reconstruct the signal x, we should be able to recover the corresponding Fourier coefficients. On the other hand, we should avoid aliasing for which we have the reconstructed coefficient as

$$\hat{x}^{rec}[k] = \sum_{(s-k) mod N = 0} \hat{x}[s]$$

Assume that $k_2 = k_1 + 2\tilde{k_c}$, $N \geq 2\tilde{k_c} + 1$ and $mN = k_1 + \tilde{k_c}$, note x only has nonzero Fourier coefficients between k_1 and k_2, Since

$$\psi_{-\tilde{k_c}} = \psi_{-\tilde{k_c}+mN} = \psi_{-\tilde{k_c}+k_1+\tilde{k_c}} = \psi_{k_1}$$
$$\psi_{\tilde{k_c}} = \psi_{\tilde{k_c}+mN} = \psi_{\tilde{k_c}+k_1+\tilde{k_c}} = \psi_{k_2}$$

The reconstruction coefficients are

$$\hat{x}^{rec}[k] = \frac{1}{N}\langle \psi_k, x_N \rangle$$

or we can write x_N as

$$x_N = \sum_{k=-\tilde{k_c}}^{\tilde{k_c}} \hat{x}_k \psi_k$$

which has $-\tilde{k_c} \leq k \leq k_c$

Then to reconstruct coefficients we have

$$\hat{x}^{rec}[k] = \sum_{(s-k) mod N = 0} \hat{x}[s]$$

which has $k_1 \leq s \leq k_2$

Based on the assumptions we have

$$(k_1 - (-k_c)) mod N = 0$$

$$(k_2 - (k_c)) mod N = 0$$

These allow us to have only one $\hat{x}[s]$ map to each $\hat{x}^{rec}[k]$, i.e. 1-to-1 mapping. Thus,

$$\hat{x}^{rec}[-k_c] = \hat{x}[k_1]$$
$$...$$
$$\hat{x}^{rec}[k_c] = \hat{x}[k_2]$$

Then we can reconstruct the signal x correctly from $\hat{x}$.

Problem 4

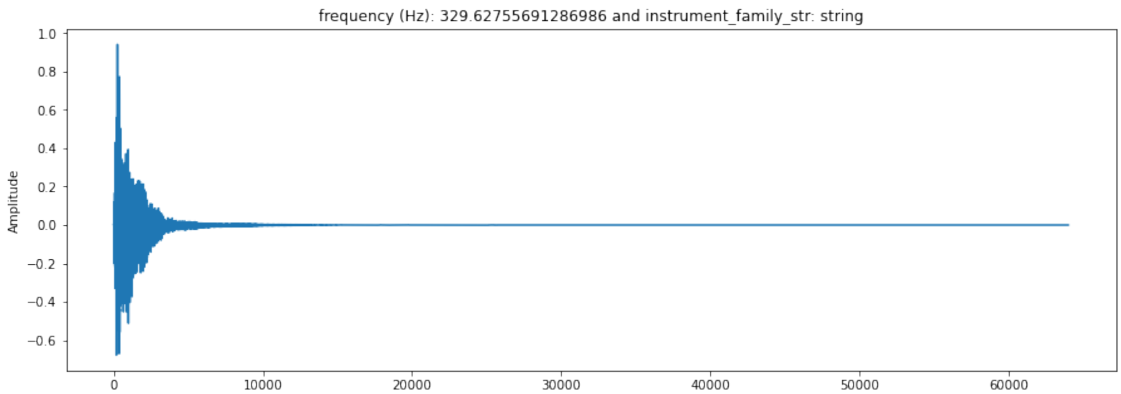
[26]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import pandas as pd
import sounddevice as sd
import soundfile
from music_tools import *
```

[27]:
```
#(a)
df_train,df_test = load_df()
print(df_train.head())
print("Number of training examples: %d"%len(df_train))
print("Number of test examples: %d"%len(df_test))
sigs_train,sigs_test = load_signals(df_train),load_signals(df_test)
y_train,y_test = df_train['pitch'].values,df_test['pitch'].values
all_pitches = sorted({p for p in y_train})
print('Pitches:',all_pitches)
print({s for s in df_train['instrument_family_str']})
```

```
                          filename   frequency  instrument_family  \
0      string_acoustic_014-064-127  329.627557                  8
1  keyboard_electronic_001-065-127  349.228231                  4
2       bass_synthetic_034-065-127  349.228231                  0
3      guitar_acoustic_010-064-100  329.627557                  3
4  keyboard_electronic_001-060-075  261.625565                  4

  instrument_family_str  pitch  sample_rate
0                string     64        16000
1              keyboard     65        16000
2                  bass     65        16000
3                guitar     64        16000
4              keyboard     60        16000
Number of training examples: 1000
Number of test examples: 283
Pitches: [60, 62, 64, 65, 67, 69, 71, 72]
{'keyboard', 'flute', 'guitar', 'brass', 'bass', 'string', 'organ', 'reed',
'mallet', 'vocal'}
```

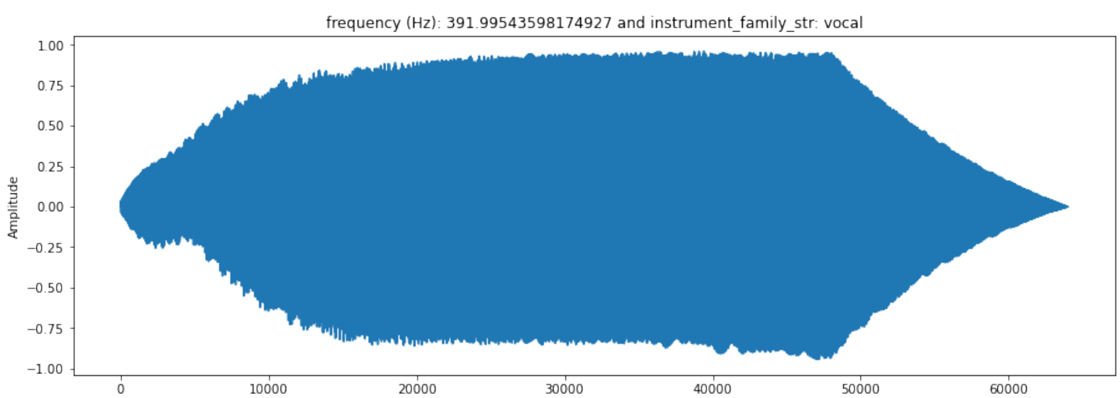
[28]:
```
fig, ax = plt.subplots(figsize=(15, 5))
ax.plot(sigs_train[0])
ax.set_title('frequency (Hz): {} and instrument_family_str: {}'.
 ↪format(df_train['frequency'][0], df_train['instrument_family_str'][0]))
ax.set_ylabel('Amplitude')
#play_signal(sigs_train[0], df_train['frequency'][0])
```

[28]:
```
Text(0, 0.5, 'Amplitude')
```



[29]:
```
vocal_idx = df_train[df_train['instrument_family_str'] == 'vocal'].index[0]
fig, ax = plt.subplots(figsize=(15, 5))
ax.plot(sigs_train[vocal_idx])
ax.set_ylabel('Amplitude')
ax.set_title('frequency (Hz): {} and instrument_family_str: {}'.
 ↪format(df_train['frequency'][vocal_idx],␣
 ↪df_train['instrument_family_str'][vocal_idx]))
```

[29]:
```
Text(0.5, 1.0, 'frequency (Hz): 391.99543598174927 and instrument_family_str:
vocal')
```



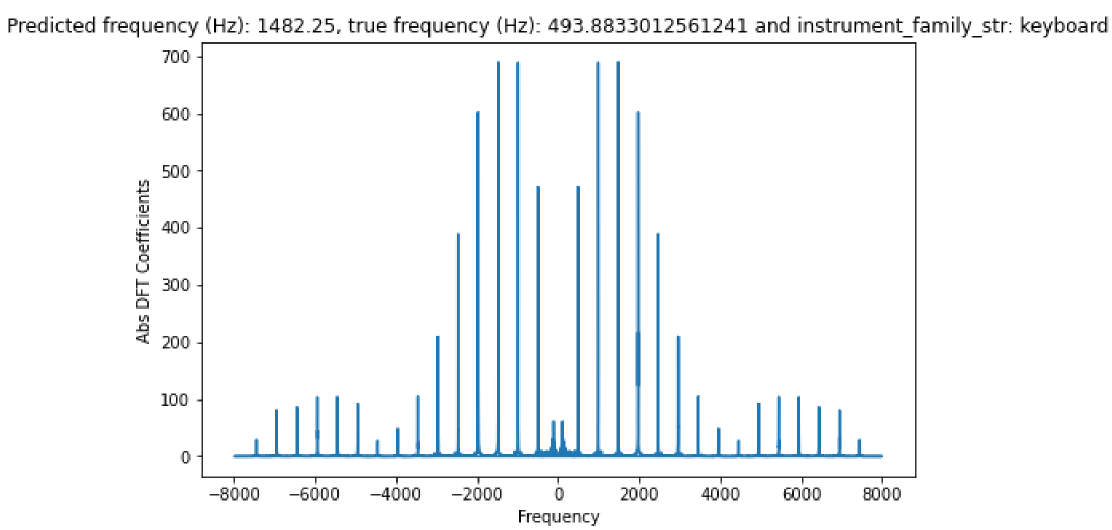
[42]:
```
# (b)
pred_freq = []
for i in range(sigs_test.shape[0]):
    signal = sigs_test[i]
    fourier = np.fft.fft(signal)
    freq = np.fft.fftfreq(len(fourier), 1/16000)
```

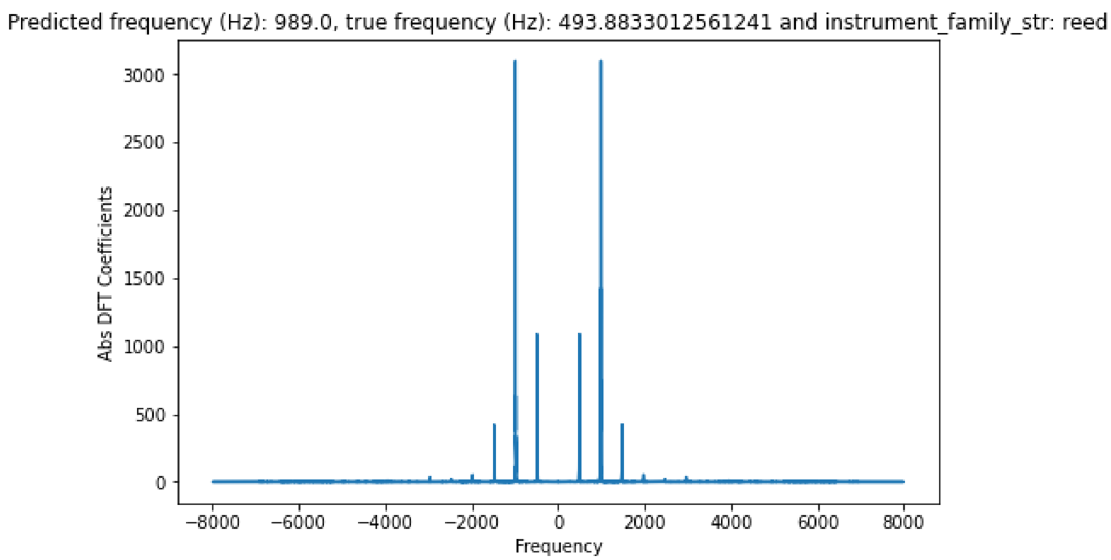
```
    idx = np.where(freq > 0)
    freq = freq[idx[0]]
    fourier = fourier[idx[0]]
    a = np.argmax(np.abs(fourier))
    pred_freq.append(freq[a])

pred_pitch = [freq2pitch(freq) for freq in pred_freq]
check = [x == y for x, y in zip(pred_pitch, y_test)]
accuracy = np.sum(check)/len(check)
```

[43]:
```
print('The overall accuracy using this method is', accuracy)
```

```
The overall accuracy using this method is 0.7208480565371025
```

[83]:
```
wrong_pred = [i for i, x in enumerate(check) if not x]
for i in [0, 1]:
    fig, ax = plt.subplots(figsize = (8, 5))
    signal = sigs_test[wrong_pred[i]]
    fourier = np.abs(np.fft.fft(signal))
    freq = np.fft.fftfreq(len(fourier), 1/16000)
    ax.plot(freq, fourier)
    ax.set_xlabel('Frequency')
    ax.set_ylabel('Abs DFT Coefficients')
    ax.set_title('Predicted frequency (Hz): {}, true frequency (Hz): {} and␣
 ↪instrument_family_str: {}'.format(pred_freq[wrong_pred[i]], df_test.
 ↪loc[1000+wrong_pred[i], 'frequency'], df_test.loc[1000+wrong_pred[i],␣
 ↪'instrument_family_str']))
```





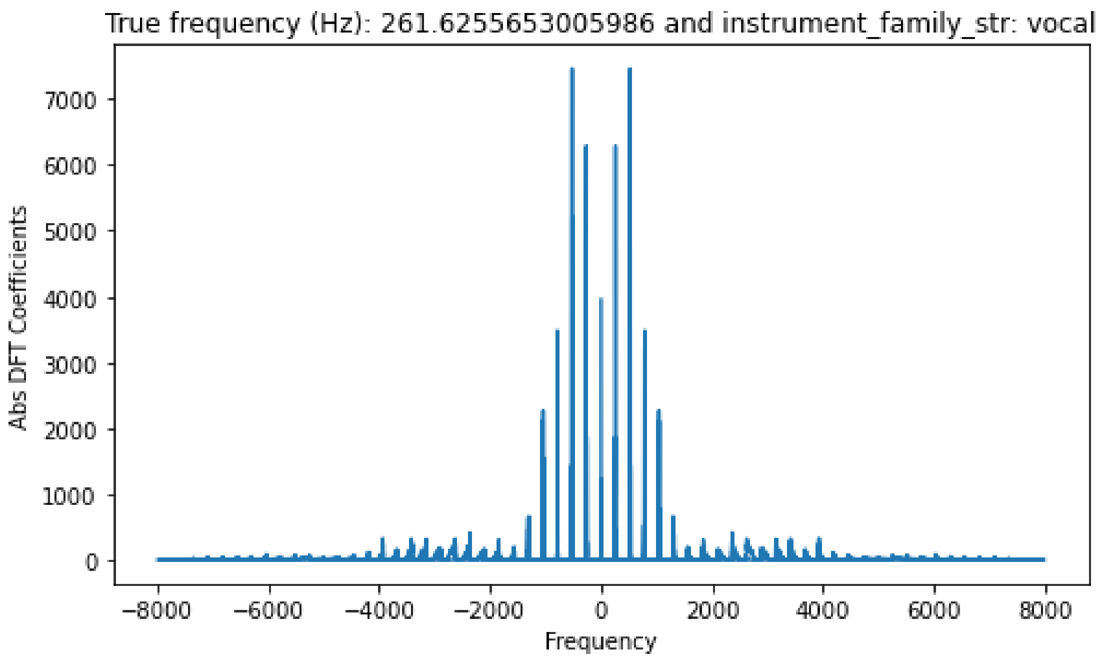
[84]:
```
instr_fam = list(set(np.array(df_test['instrument_family_str'])))
incorrect = []
for fam in instr_fam:
    idx = df_test.index[df_test['instrument_family_str']==fam]
    fam_check = 1-np.sum([check[i-1000] for i in idx])/len(idx)
    incorrect.append(fam_check)
```

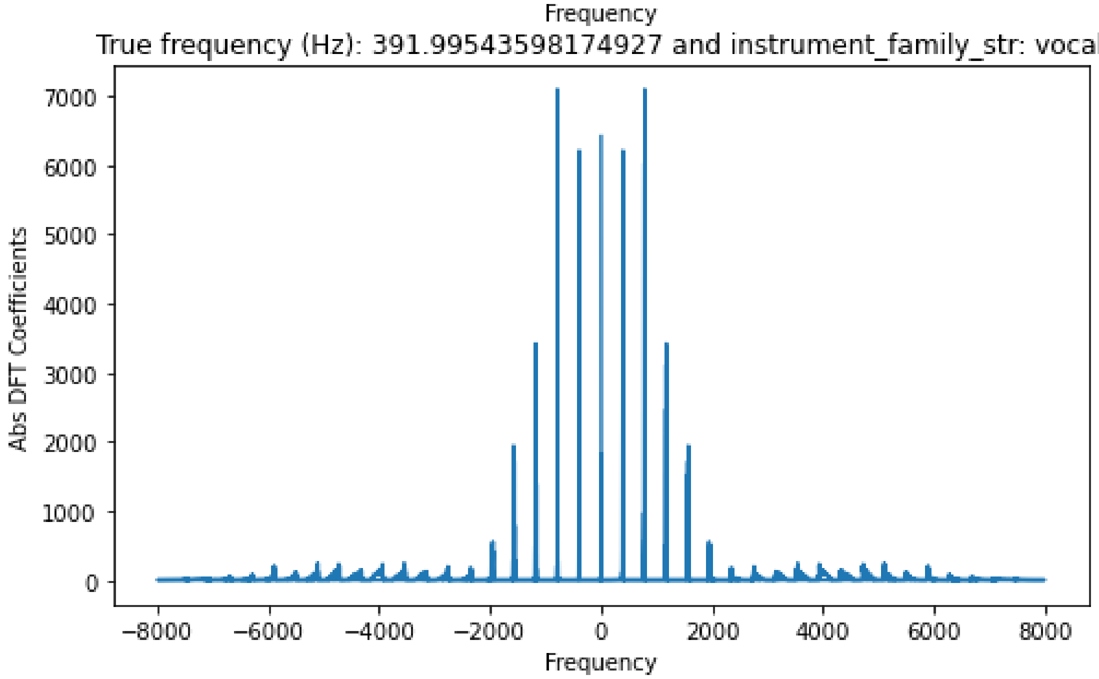
[85]:
```
nameidx = np.argmax(incorrect)
print('The instrument family with hightest incorrect prediction rate is:',␣
↪instr_fam[nameidx])
```

```
The instrument family with hightest incorrect prediction rate is: vocal
```

The previous answer makes sense because human voices have more overtones and variations while instruments produce pure frequencies.

[86]:
```
fig, ax = plt.subplots(2, figsize = (8, 10))
idx = df_test.index[df_test['instrument_family_str']=='vocal']
for i in [0, 1]:
    signal = sigs_test[idx[i]-1000]
    fourier = np.abs(np.fft.fft(signal))
    freq = np.fft.fftfreq(len(fourier), 1/16000)
    ax[i].plot(freq, fourier)
    ax[i].set_xlabel('Frequency')
    ax[i].set_ylabel('Abs DFT Coefficients')
    ax[i].set_title('True frequency (Hz): {} and instrument_family_str: {}'.
↪format(df_test.loc[idx[i], 'frequency'], df_test.loc[idx[i],␣
↪'instrument_family_str']))
```





[87]:
```
#(c)
model = LogisticRegression(multi_class='multinomial',solver='lbfgs')

coef = []
for i in range(sigs_train.shape[0]):
    coef.append(np.abs(np.fft.fft(sigs_train[i])))
```

```
coef = np.array(coef)
model.fit(coef, df_train['pitch'])
```

```
C:\Users\kalle\Anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[87]:
```
LogisticRegression(multi_class='multinomial')
```

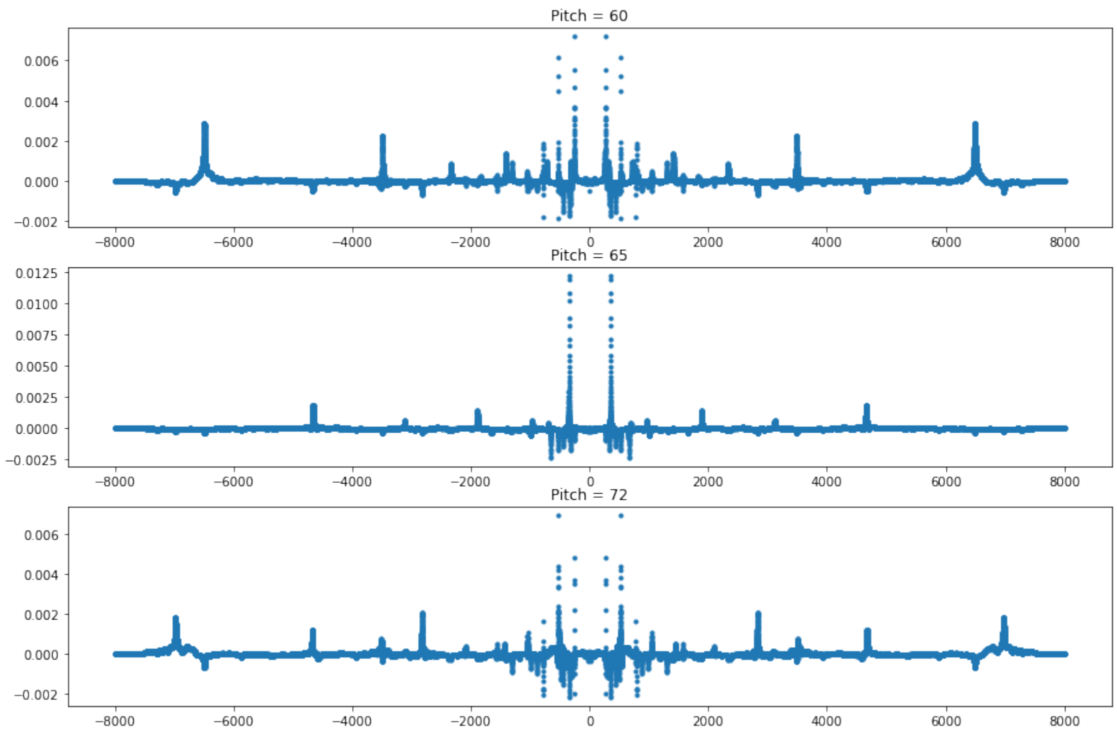
[48]:
```
coef_test = []
for i in range(sigs_test.shape[0]):
    coef_test.append(np.abs(np.fft.fft(sigs_test[i])))

coef_test = np.array(coef_test)
sc = model.score(coef_test, df_test['pitch'])
print('Score on the test set computed by the model: {}'.format(sc))
```

```
Score on the test set computed by the model: 0.9964664310954063
```

[49]:
```
coef_lr = model.coef_
classes = model.classes_
```

[50]:
```
fig, ax = plt.subplots(3, figsize=(15,10))
ax[0].plot(np.fft.fftfreq(len(coef_lr[0]), 1/16000), coef_lr[0], '.')
ax[0].set_title('Pitch = 60')
ax[1].plot(np.fft.fftfreq(len(coef_lr[0]), 1/16000), coef_lr[3], '.')
ax[1].set_title('Pitch = 65')
ax[2].plot(np.fft.fftfreq(len(coef_lr[0]), 1/16000), coef_lr[-1], '.')
ax[2].set_title('Pitch = 72')

#i = 0
#for c in [60, 65, 72]:
#    idx = np.where(classes == c)
#    print(idx)
#    ax[i].plot(coef[idx[0]])
#    i += 1
```

[50]:
```
Text(0.5, 1.0, 'Pitch = 72')
```



(c) The graphs from the previous problem plot the logistic regression coefficients against fft frequencies. Generally for frequencies with lower magnitudes, the coefficients vary more and can have high magnitudes. Meanwhile, the graphs are symmetric around x=0.