Negative_pattern_match_regex

May 17, 2019

[1]:
```
#Finding Putative NRF2 Binding Sites Using Motifs and then Visualize them
import time
import gzip
import shutil
import pandas as pd
import numpy as np
import re
from Bio import SeqIO
from itertools import islice
import matplotlib.pyplot as plt
import twobitreader as tbr
```

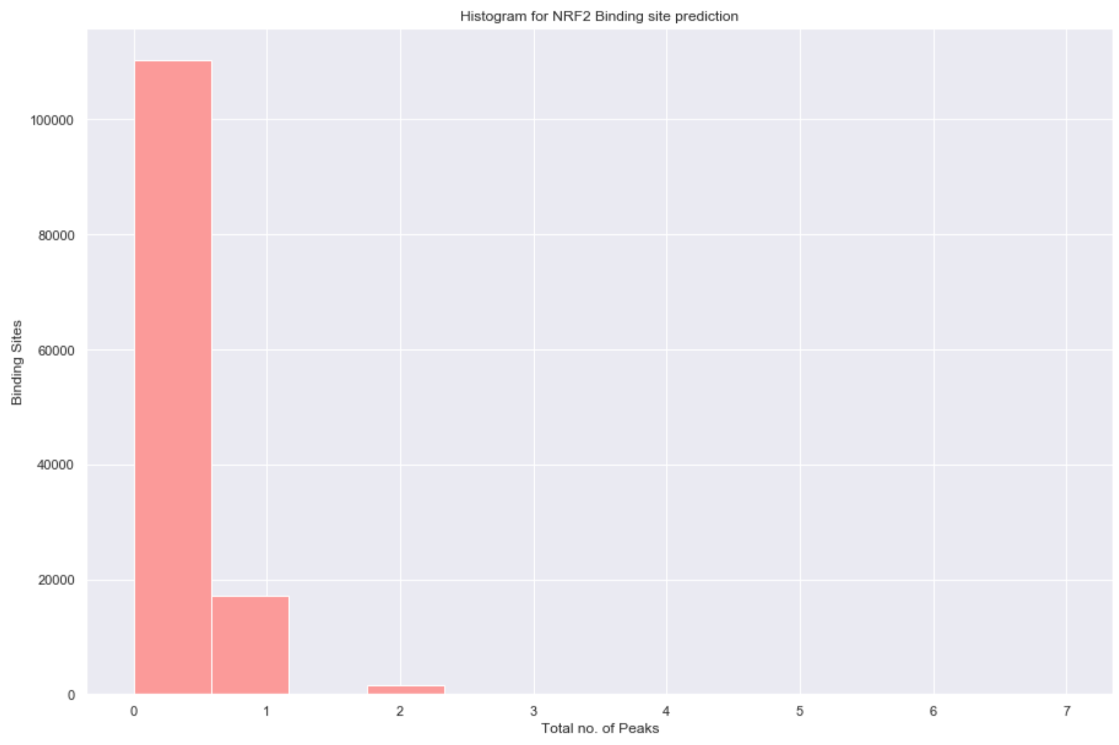
[2]:
```
#Creating Reverese Complements
def reverseComp(Seq):
    seq = Seq.upper()
    d = {'A':'T', 'T':'A', 'G':'C', 'C':'G'}
    try:
        seq = seq[::-1]
        rc_seq = "".join([d[nuc] for nuc in seq])
    except KeyError:
        return "Not Viable DNA Seq"
    return rc_seq
```

[30]:
```
motif = '[AGC]TGA[CTG][ATCG][GCAT][AGT]GC[ATCG]'
regBS = re.compile(motif)
motifDF = []
motifQuant = []
genome = tbr.TwoBitFile('/Users/kalyanidhusia/Downloads/hg19.2bit')
with open('/Users/kalyanidhusia/Desktop/nrf2_R/search2/min_score70/DNAse/
 ↪negativenrf2/sudin_negative_nrf2.bed') as f:
    for line in f:
        if line.startswith('track') == False:
            peak = list(line.split())
            seq = (genome[peak[0]][int(peak[1]):int(peak[2])]).upper()
            rSeq = reverseComp(seq)
            sequences = []
            sequences.extend(re.findall(regBS, seq))
            sequences.extend(re.findall(regBS, rSeq))
            if len(sequences) >= 0:
                seqs = pd.DataFrame({'binding':sequences, 'chrom':peak[0],␣
 ↪'chromstart':peak[1], 'chromend':peak[2], 'NR':'NRF2'})
                motifDF.append(seqs)
                motifQuant.append([peak[0], peak[1], peak[2], len(seqs),␣
 ↪len(seq)])
search_reg = pd.concat(motifDF)
names = ['chrom', 'chromstart', 'chromend', 'numOfMatches', 'lenSeq']
dist_reg = pd.DataFrame(motifQuant, columns=names)
search_reg.head()
n = 1
x = [len(i[6+n:-6-n]) for i in search_reg['binding']]
```

[10]:
```
dist_reg.head()
```

[10]:
```
  chrom  chromstart  chromend  numOfMatches  lenSeq
0  chr1       10422     10572             0     150
1  chr1      235628    235778             0     150
2  chr1      534194    534344             0     150
3  chr1      662533    662683             0     150
4  chr1      713255    713405             0     150
```

[51]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,10))
plt.title('Histogram for NRF2 Binding site prediction')
plt.xlabel('Total no. of Peaks')
plt.ylabel('Binding Sites')
plt.grid(True)
plt.hist(dist_reg['numOfMatches'], bins=12, color='#fb9a99')
plt.show()
```



[12]:
```
#Creating the LOGO for NRF2 Motif using weblogo
from Bio.Seq import Seq
from Bio import motifs
instances = search_reg['binding']
m = motifs.create(instances)
m.weblogo('trynoflankingneg.png')
```

[13]:
```
#One Hot Encoding Sequence
def oheSeq(DNAString):
    seq = DNAString.upper()
    nuc = 'ATGC'
    char2int = dict((c, i) for i, c in enumerate(nuc))
    int2char = dict((i, c) for i, c in enumerate(nuc))
    integer_encoded = [char2int[char] for char in seq]
    OHE = []
    for value in integer_encoded:
        letter = [0 for _ in range(len(nuc))]
        letter[value] = 1
        OHE.append(letter)
    seq_ohe = np.asarray(OHE)
    return seq_ohe
```

[14]:
```
#modified OHE for interspaced regions
def oheSeqMod(DNAString, flank_len):
    seq = DNAString.upper()
    flanks = seq[:6+flank_len] + seq[-6-flank_len:]
    nuc = 'ATGC'
    char2int = dict((c, i) for i, c in enumerate(nuc))
    int2char = dict((i, c) for i, c in enumerate(nuc))
    integer_encoded = [char2int[char] for char in flanks]
    OHE = []
    for value in integer_encoded:
        letter = [0 for _ in range(len(nuc))]
        letter[value] = 1
        OHE.extend(letter)
    OHE.append(len(seq[6+flank_len:-6-flank_len]))
    return OHE
```

[15]:
```
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import fetch_mldata
```

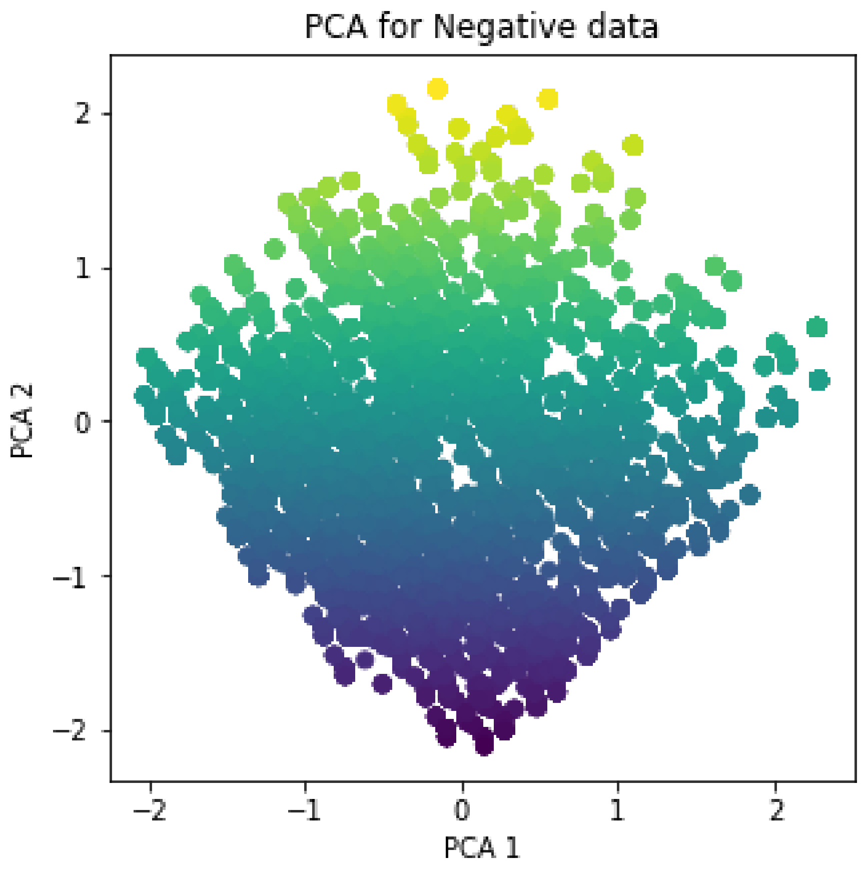
[16]:
```
#Dimensionality Reductions
#euc for euclidean
#Dimensionality reduction using PCA takes ~30m mins
euc_ohe =  np.array([oheSeqMod(i, 6) for i in search_reg['binding']])

pca = PCA(n_components=3)
pca_result = pca.fit_transform(euc_ohe)
print('Explained variation per principal component: {}'.format(pca.
 ↪explained_variance_ratio_))
```

```
Explained variation per principal component: [0.11961548 0.09545345 0.09405885]
```

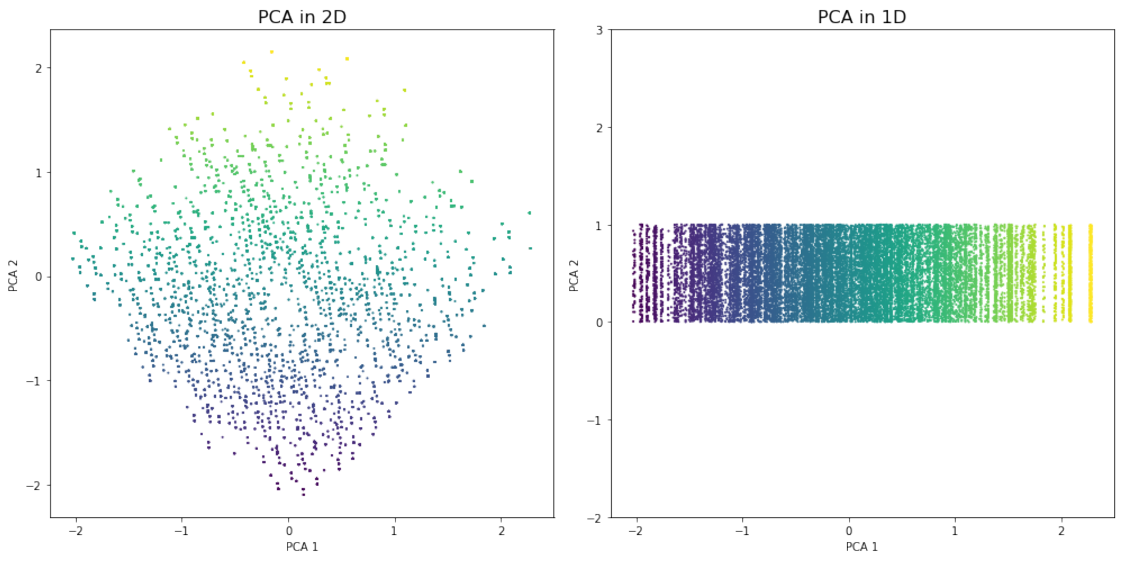
[17]:
```
pca_result
```

[17]:
```
array([[-1.04450656,  1.27934381,  1.09757867],
       [-0.23746919,  0.22021958, -0.26234606],
       [-0.23746919,  0.22021958, -0.26234606],
       ...,
       [ 2.28448836,  0.26589309, -0.52769797],
       [-0.46424685, -0.02274522, -0.96906135],
       [-0.89629829, -0.99526766, -0.33117659]])
```

[18]:
```
dim1 = pca_result[:,0]
dim2 = pca_result[:,1]
plt.figure(figsize=(5,5))
plt.scatter(dim1, dim2, c=dim2)
plt.title("PCA for Negative data")
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



[19]:
```
# Subsampling
#Showcase of various other options

plt.figure(figsize=(14,7))
plt.subplot(121)
plt.scatter(dim1, dim2,c=dim2, s=1)
plt.title('PCA in 2D', fontsize=16)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.subplot(122)
plt.scatter(pca_result[:,0], np.random.uniform(size=dim2.shape[0]), c=dim1, s=1)
plt.ylim([-2,3])
plt.title('PCA in 1D', fontsize=16)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.tight_layout()
```

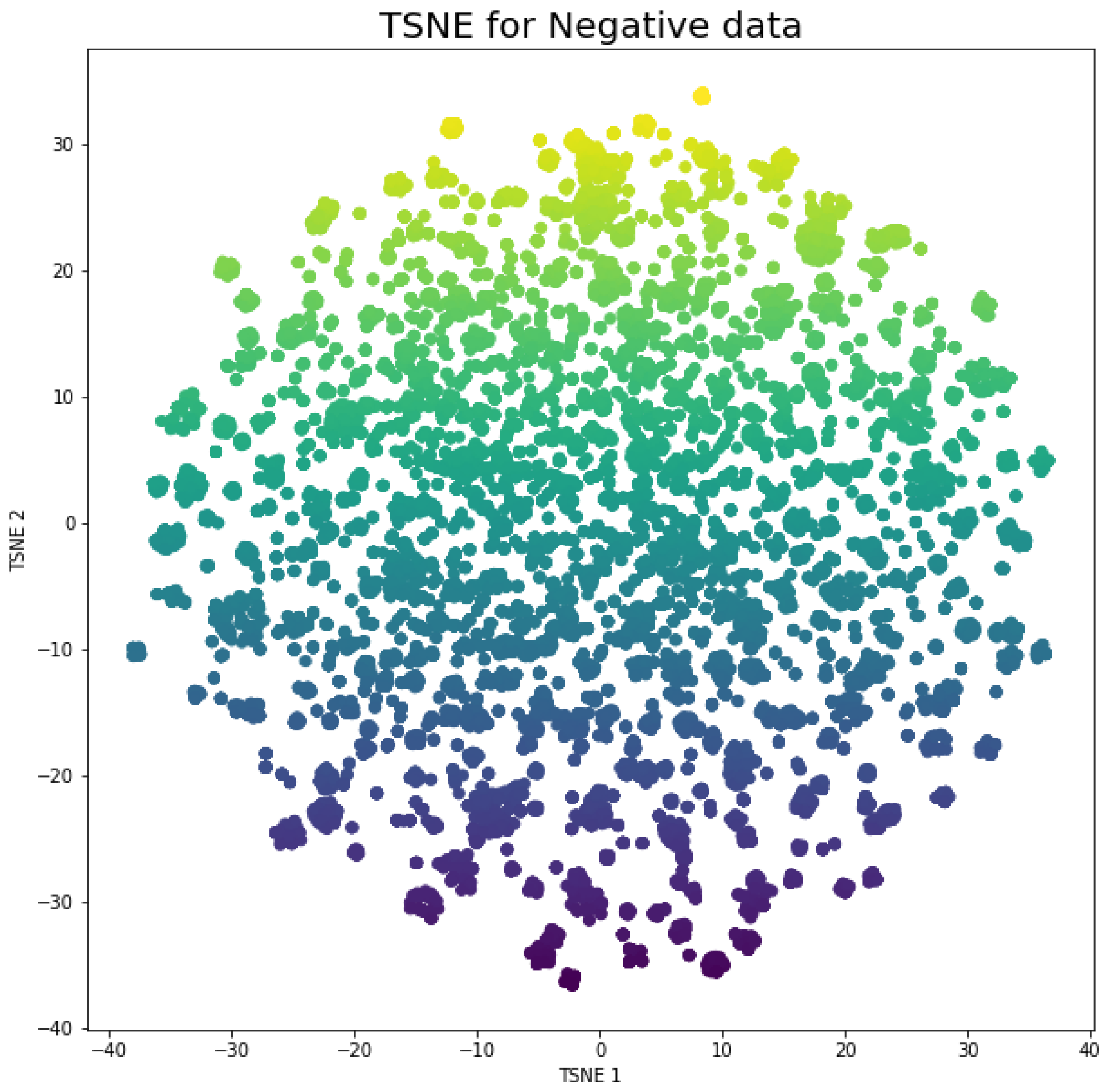


[20]:
```
#Dimensionality Reductions
#euc for euclidean
#euc_ohe =  np.array([oheSeqMod(i, 6) for i in search_reg['binding']])
#from sklearn.decomposition import PCA
#from sklearn.manifold import TSNE
#import umap
#Be careful with umap installation pip install umap will cause you to install␣
↪the wrong ver of umap
#(and will also break the real module)

#pca = PCA(n_components=2)
#pca.fit(euc_ohe)
#print(pca.explained_variance_ratio_)

#Takes a bit with larger datasets (scales n^2 in both compute time and memory)

X_embedded = TSNE(n_components=3).fit_transform(euc_ohe)
```

[21]:
```
#t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence␣
↪between two distributions:
#a distribution that measures pairwise similarities of the input objects and a␣
↪distribution that measures pairwise similarities of the corresponding␣
↪low-dimensional points in the embedding
X_embedded
```

[21]:
```
array([[-34.74552  ,   8.286813 ,   1.244146 ],
       [  6.0284348,  -7.734971 , -29.90522  ],
       [  6.0284896,  -7.735072 , -29.905285 ],
       ...,
       [ -6.4460535, -22.794235 , -11.632522 ],
       [-13.707052 ,  -3.1306846,  14.101543 ],
       [ -4.539845 ,   3.7606907,  -7.7787585]], dtype=float32)
```

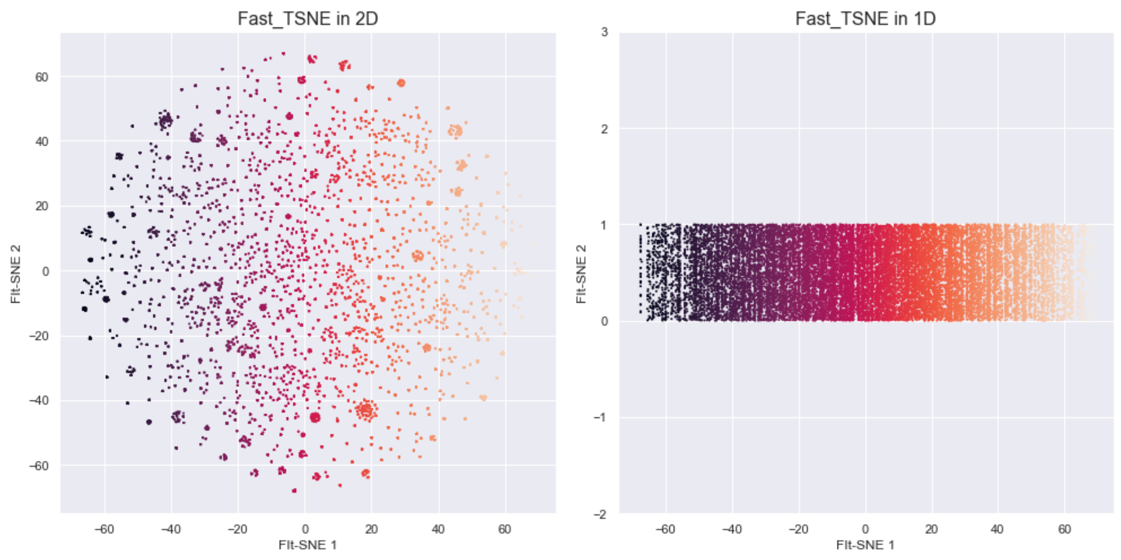
[22]:
```
#Graphing represntation for normal T-SNE
#Currently is takes ~15 to 20mins
dim1 = X_embedded[:, 0]
dim2 = X_embedded[:, 1]
plt.figure(figsize=(10,10))
plt.scatter(dim1, dim2, c=dim2)
plt.title("TSNE for Negative data", fontsize=20)
plt.xlabel('TSNE 1')
plt.ylabel('TSNE 2')
plt.show()
plt.savefig('TSNE_NRF2_negative')
```



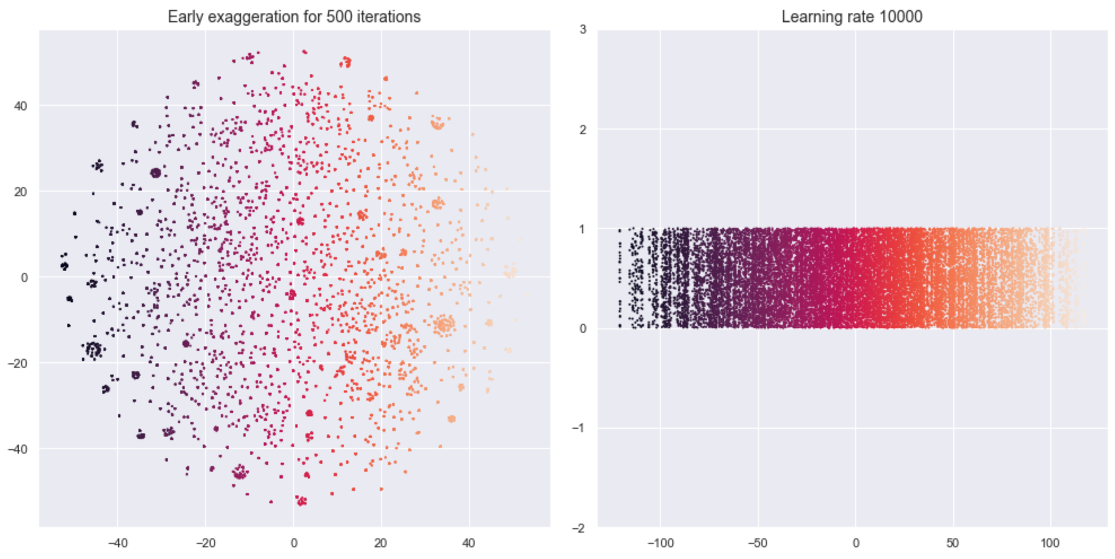
```
<Figure size 432x288 with 0 Axes>
```

[23]:
```python
import numpy as np
import pylab as plt
import seaborn as sns; sns.set()

# change the path for fast-TSNE!
import sys; sys.path.append('/Users/kalyanidhusia/Downloads/FIt-SNE-master/')
from fast_tsne import fast_tsne
```

[24]:
```python
Z1 = fast_tsne(X_embedded)

Z2 = fast_tsne(X_embedded)

plt.figure(figsize=(14,7))
plt.subplot(121)
plt.scatter(Z1[:,0], Z1[:,1], c=Z1[:,0], s=1)
plt.title('Fast_TSNE in 2D', fontsize=16)
plt.xlabel('FIt-SNE 1')
plt.ylabel('FIt-SNE 2')
plt.subplot(122)
plt.scatter(Z2[:,0], np.random.uniform(size=Z2.shape[0]), c=Z2[:,0], s=1)
plt.ylim([-2,3])
plt.title('Fast_TSNE in 1D', fontsize=16)
plt.xlabel('FIt-SNE 1')
plt.ylabel('FIt-SNE 2')
plt.tight_layout()
```



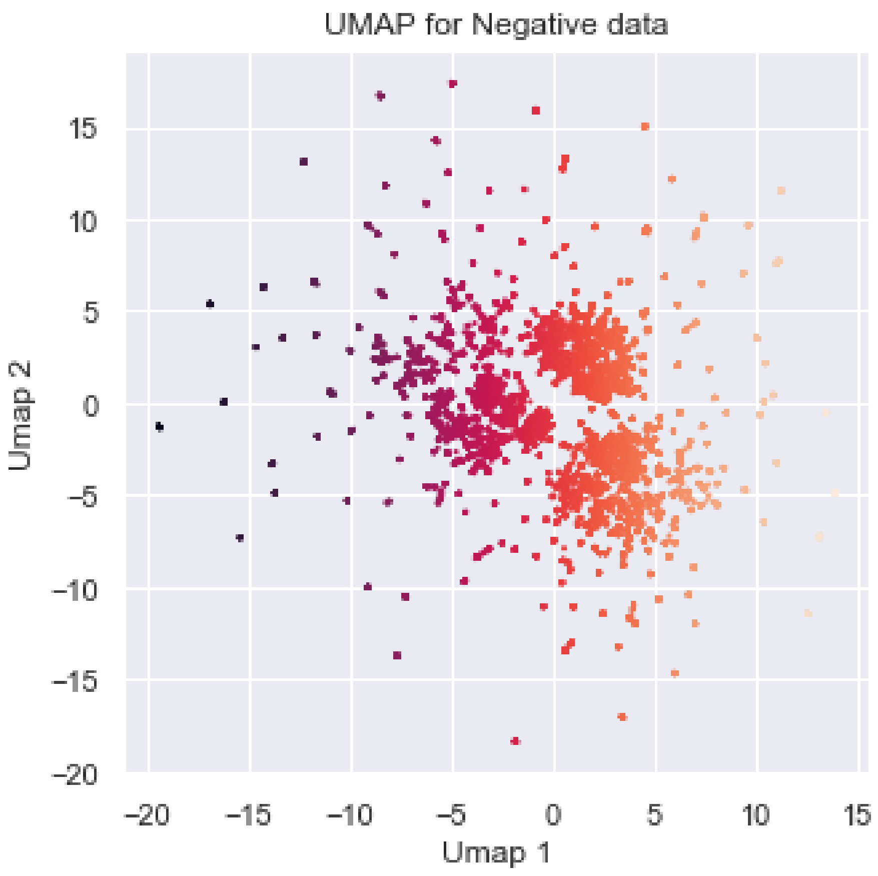
[25]:
```
# Two classes are separated into two parts on the above figure.
# This can be fixed if one uses stronger/longer early exaggeration (e.g.␣
↪stop_early_exag_iter=500)
# or higher learning rate (e.g. learning rate=1000)

Z1 = fast_tsne(X_embedded, stop_early_exag_iter=500, initialization=pca_result)

Z2 = fast_tsne(X_embedded, learning_rate=10000, initialization=pca_result)

plt.figure(figsize=(14,7))
plt.subplot(121)
plt.scatter(Z1[:,0], Z1[:,1], c=Z1[:,0], s=1)
plt.title('Early exaggeration for 500 iterations', fontsize=14)
plt.subplot(122)
plt.scatter(Z2[:,0], np.random.uniform(size=Z2.shape[0]), c=Z2[:,0], s=1)
plt.ylim([-2,3])
plt.title('Learning rate 10000', fontsize=14)
plt.tight_layout()
```



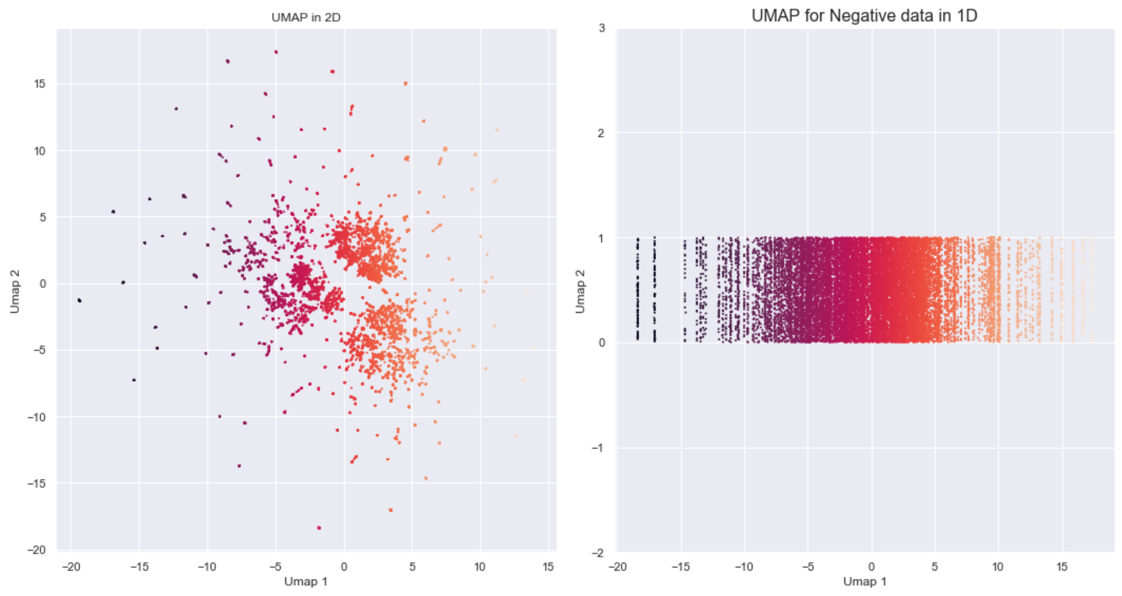
[26]:
```
#Also takes a bit (even though they claim it is faster than TSNE ~ <10m mins)
umapped = umap.UMAP().fit_transform(euc_ohe)
umapped
```

[26]:
```
array([[-19.353039  ,  -1.3077078 ],
       [  2.0771189 ,   0.21075149],
       [  2.1070747 ,   0.28643087],
       ...,
       [  7.474978  ,  10.041326  ],
       [ -5.9262986 ,   1.6900109 ],
       [  0.5149627 ,   1.6451739 ]], dtype=float32)
```

[27]:
```
#Graphing example
umap1 = umapped[:, 0]
umap2 = umapped[:, 1]
plt.figure(figsize=(5,5))
plt.scatter(umap1, umap2, c=umap1 , s=1)
plt.title("UMAP for Negative data")
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
plt.show()
#plt.savefig('OHE_NRF2_negative')
```



[28]:
```
#Making the kernel more heavy-tailed
plt.figure(figsize=(15,8))
plt.subplot(121)
plt.scatter(umap1, umap2, c=umap1 , s=1)
plt.title('UMAP in 2D')
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
plt.subplot(122)
plt.scatter(umap2, np.random.uniform(size=umap2.shape[0]), c=umap2, s=1)
plt.ylim([-2,3])
plt.title('UMAP for Negative data in 1D', fontsize=16)
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
plt.tight_layout()
#plt.savefig('OHE_NRF2_negative_kernel')
```



[37]:
```
!echo $PATH
```

```
/Users/kalyanidhusia/anaconda3/bin:/Users/kalyanidhusia/anaconda3/bin:/Users/kal
yanidhusia/anaconda3/condabin:/usr/bin:/bin:/usr/sbin:/sbin
```

[40]:
```
conda install -c anaconda nbconvert
```

```
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: /Users/kalyanidhusia/anaconda3

  added / updated specs:
    - nbconvert


The following packages will be downloaded:
    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2019.1.23  |                0         126 KB  anaconda
    certifi-2019.3.9           |           py36_0         155 KB  anaconda
    conda-4.6.14               |           py36_0         2.1 MB  anaconda
    nbconvert-5.5.0            |             py_0         381 KB  anaconda
    openssl-1.1.1              |       h1de35cc_0         4.6 MB  anaconda
    ------------------------------------------------------------
                                           Total:         7.4 MB

The following packages will be UPDATED:

  openssl              pkgs/main::openssl-1.1.1b-h1de35cc_1 -->
anaconda::openssl-1.1.1-h1de35cc_0

The following packages will be SUPERSEDED by a higher-priority channel:

  ca-certificates                                pkgs/main --> anaconda
  certifi                                        pkgs/main --> anaconda
  conda                                        conda-forge --> anaconda
  nbconvert                                    conda-forge --> anaconda



Downloading and Extracting Packages
ca-certificates-2019 | 126 KB    | ##################################### | 100%
conda-4.6.14         | 2.1 MB    | ##################################### | 100%
openssl-1.1.1        | 4.6 MB    | ##################################### | 100%
nbconvert-5.5.0      | 381 KB    | ##################################### | 100%
certifi-2019.3.9     | 155 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

[44]:
```
!export PATH=/Library/TeX/texbin/xelatex:$PATH
```

[45]:
```
jupyter nbconvert Negative_pattern_match_regex.ipynb --to pdf
```

```
          File "<ipython-input-45-8e9dcb1f9a2b>", line 1
        jupyter nbconvert Negative_pattern_match_regex.ipynb --to pdf
                        ^
    SyntaxError: invalid syntax
```

[ ]:
```
```