TSNE VISUALISATION

- TSNE stands for t-distributed stochastic Neighborhood Embedding
- It is the state of the art data visualisation technique developed by van der Maaten & Jefrey Hinton in 2008.
- nD-dim---->2D-dim
- pca preserves global shape or structure of the data and it won't preserves local structure where as tsne is a complex phenomenon tries to preserve local and global structure
- Neighborhood is the point which is closest and surronding our query point
- Embedding means nothing but projecting the points from nD plane to 2D plane
- Sometimes,it is impossible to preserve neighborhood distance,it is called "CROWDING PROBLEM".
- stochastic means not determenistic or probabilistic.
- t-sne is an iterative algorithm.
- t-sne algorithm adapts its notion of distance to regional density variation in data set.
- The parameters are:

  - perplexity
  - steps

- Relatie size of clusters can't be seen
- perplexity can be taken as how to balance attention between local and global aspects of data(generally lies between 5~50).
- Perplexity should always be less than no of points.
- No of iterations should be upto stable configuration.
- We must analyse multiple knots with multiple perplexities.
- t-sne tends to expand dense regions data.
- Distance between clusters might not give any Information.

In [0]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
data = pd.read_csv("/content/drive/My Drive/Data_Scientist/pca-tsne/sign_mnist_train.csv")
data.head()
```

Out[3]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pix
0	3	107	118	127	134	139	143	146	150	153	156	158	160	163	165	
1	6	155	157	156	156	156	157	156	158	158	157	158	156	154	154	
2	2	187	188	188	187	187	186	187	188	187	186	185	185	185	184	
3	2	211	211	212	212	211	210	211	210	210	211	209	207	208	207	
4	13	164	167	170	172	176	179	180	184	185	186	188	189	189	190	

5 rows × 785 columns

In [4]:
```
data.shape
```

Out[4]:
```
(27455, 785)
```

In [5]:
```
labels = data['label']
data = data.drop('label',axis= 1)
data.head()
```

Out[5]:

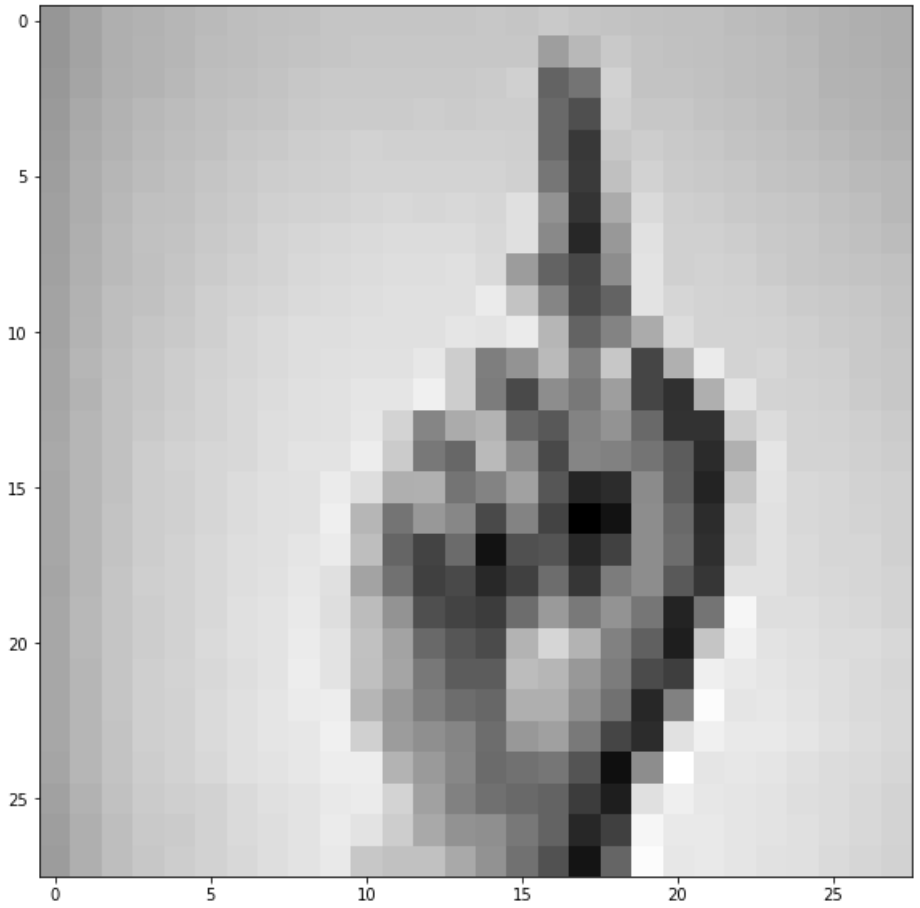
	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	
0	107	118	127	134	139	143	146	150	153	156	158	160	163	165	159	
1	155	157	156	156	156	157	156	158	158	157	158	156	154	154	153	
2	187	188	188	187	187	186	187	188	187	186	185	185	185	184	184	
3	211	211	212	212	211	210	211	210	210	211	209	207	208	207	206	
4	164	167	170	172	176	179	180	184	185	186	188	189	189	190	191	

5 rows × 784 columns

In [6]:
```
print(labels.shape)
```

```
(27455,)
```

In [7]:
```
plt.figure(figsize=(10,10))
ids = 24

data_matrix = data.iloc[ids].as_matrix().reshape(28,28)  # reshape from 1d to 2d pixel arra
plt.imshow(data_matrix, interpolation = "none", cmap = "gray")
plt.show()

print(labels[ids])
```




```
3
```

TSNE is complex algorithm so i consider it for 2k data points only.

In [12]:
```
data = data.head(2000)
labels = labels.head(2000)
print("The shape of the data becomes",data.shape," and labels become",labels.shape)
```

```
The shape of the data becomes (2000, 784)  and labels become (2000,)
```

In [13]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
standardised_data = scaler.fit_transform(data)
print("The standardised data shape is",standardised_data.shape)
```

```
The standardised data shape is (2000, 784)
```

In [0]:
```
words_dict ={
    0:'A',
    1:'B',
    2:'C',
    3:'D',
    4:'E',
    5:'F',
    6:'G',
    7:'H',
    8:'I',
    9:'J',
    10:'K',
    11:'L',
    12:'M',
    13:'N',
    14:'O',
    15:'P',
    16:'Q',
    17:'R',
    18:'S',
    19:'T',
    20:'U',
    21:'V',
    22:'W',
    23:'X',
    24:'Y',
    25:'Z'
}
```

In [15]:
```
words_dict[24]
```

Out[15]:
```
'Y'
```

In [16]:
```
for i in range(len(labels)) :
  if(labels[i] in words_dict.keys()) :
    labels[i] = words_dict[labels[i]]
labels = pd.Series(labels)
print(labels.head())
```

```
0    D
1    G
2    C
3    C
4    N
Name: label, dtype: object
```

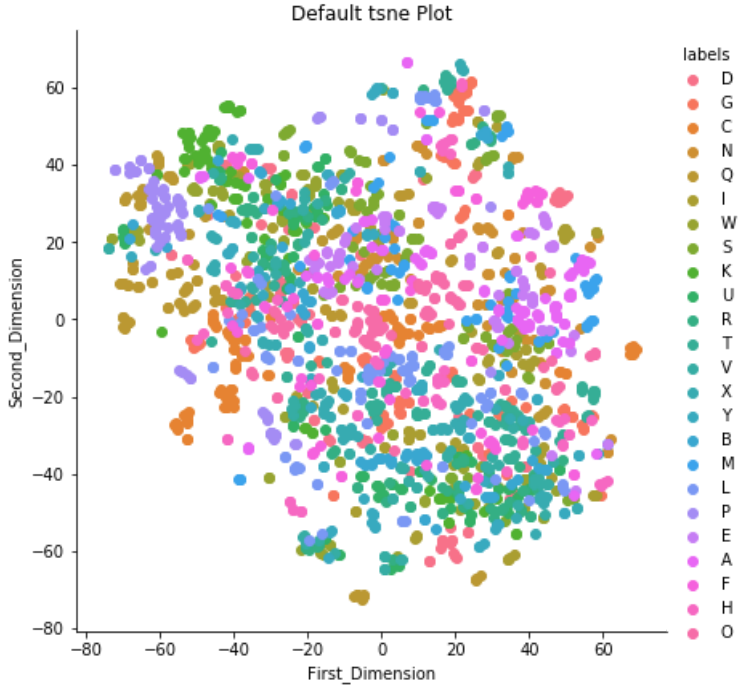
In [17]:
```
labels[24]
```

Out[17]:
```
'D'
```

TSNE parameter tuning

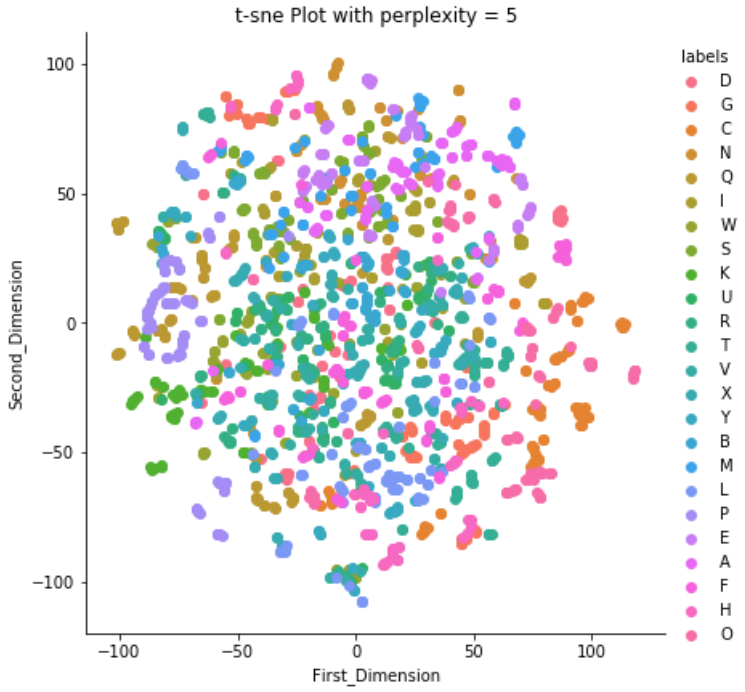
- ***Learning rate is set default to 200***
- ***Default perplexity is 30***
- ***Defalut No of iterations or steps is 1000***

In [19]:
```
from sklearn.manifold import TSNE


final_data = standardised_data
final_labels = labels

model = TSNE(n_components=2, random_state=0)

tsne_data = model.fit_transform(final_data)

# Creating pandas data frame by vertically stacking out tsne_data and labels

tsne_data = np.vstack((tsne_data.T, final_labels)).T
tsne_dataframe = pd.DataFrame(data=tsne_data, columns=("First_Dimension", "Second_Dimension

# Plotting using seaborn

sns.FacetGrid(tsne_dataframe, hue="labels", size=6).map(plt.scatter, 'First_Dimension', 'Se
plt.title("Default tsne Plot")
plt.show()
```



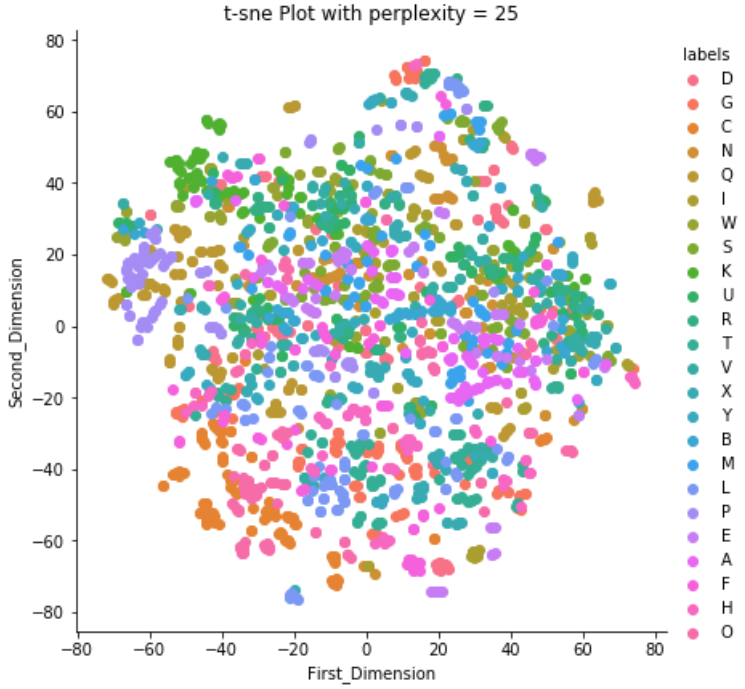
TSNE plot using perplexity=5

In [20]:
```
from sklearn.manifold import TSNE


final_data = standardised_data
final_labels = labels

model = TSNE(n_components=2, random_state=0,perplexity=5)

tsne_data = model.fit_transform(final_data)

# Creating pandas data frame by vertically stacking out tsne_data and labels

tsne_data = np.vstack((tsne_data.T, final_labels)).T
tsne_dataframe = pd.DataFrame(data=tsne_data, columns=("First_Dimension", "Second_Dimension

# Plotting using seaborn

sns.FacetGrid(tsne_dataframe, hue="labels", size=6).map(plt.scatter, 'First_Dimension', 'Se
plt.title("t-sne Plot with perplexity = 5")
plt.show()
```



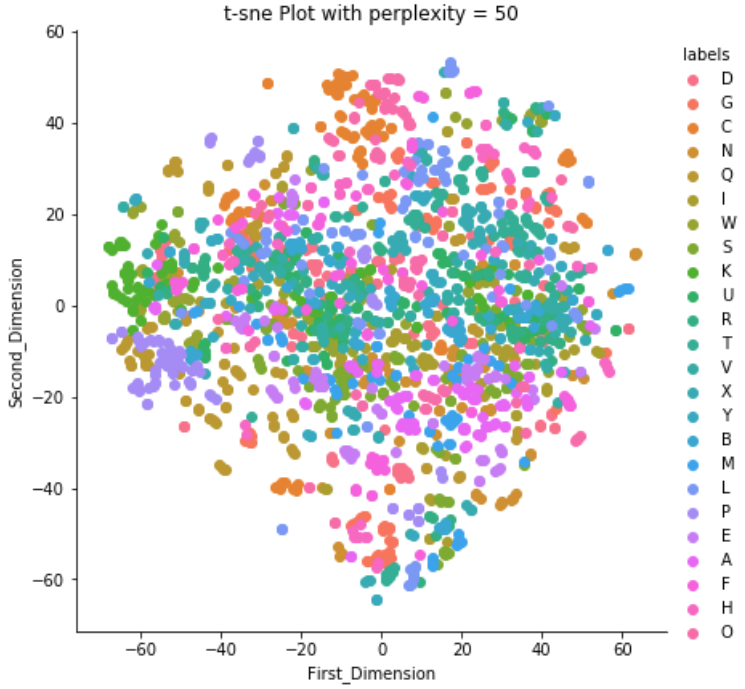
TSNE plot with perplexity = 25

In [21]:
```
from sklearn.manifold import TSNE


final_data = standardised_data
final_labels = labels

model = TSNE(n_components=2, random_state=0,perplexity=25)

tsne_data = model.fit_transform(final_data)

# Creating pandas data frame by vertically stacking out tsne_data and labels

tsne_data = np.vstack((tsne_data.T, final_labels)).T
tsne_dataframe = pd.DataFrame(data=tsne_data, columns=("First_Dimension", "Second_Dimension

# Plotting using seaborn

sns.FacetGrid(tsne_dataframe, hue="labels", size=6).map(plt.scatter, 'First_Dimension', 'Se
plt.title("t-sne Plot with perplexity = 25")
plt.show()
```



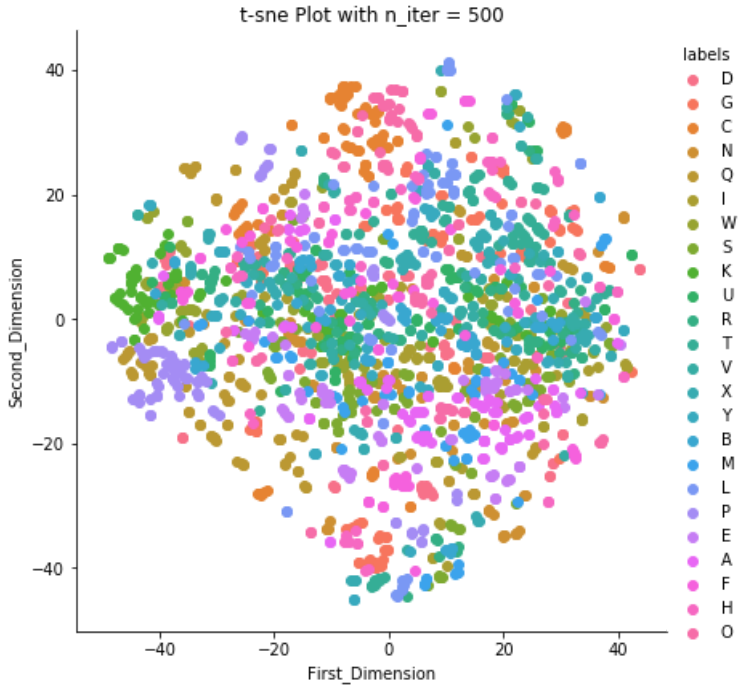
TSNE plot with perplexity=50

In [22]:
```
from sklearn.manifold import TSNE


final_data = standardised_data
final_labels = labels

model = TSNE(n_components=2, random_state=0,perplexity=50)

tsne_data = model.fit_transform(final_data)

# Creating pandas data frame by vertically stacking out tsne_data and labels

tsne_data = np.vstack((tsne_data.T, final_labels)).T
tsne_dataframe = pd.DataFrame(data=tsne_data, columns=("First_Dimension", "Second_Dimension

# Plotting using seaborn

sns.FacetGrid(tsne_dataframe, hue="labels", size=6).map(plt.scatter, 'First_Dimension', 'Se
plt.title("t-sne Plot with perplexity = 50")
plt.show()
```



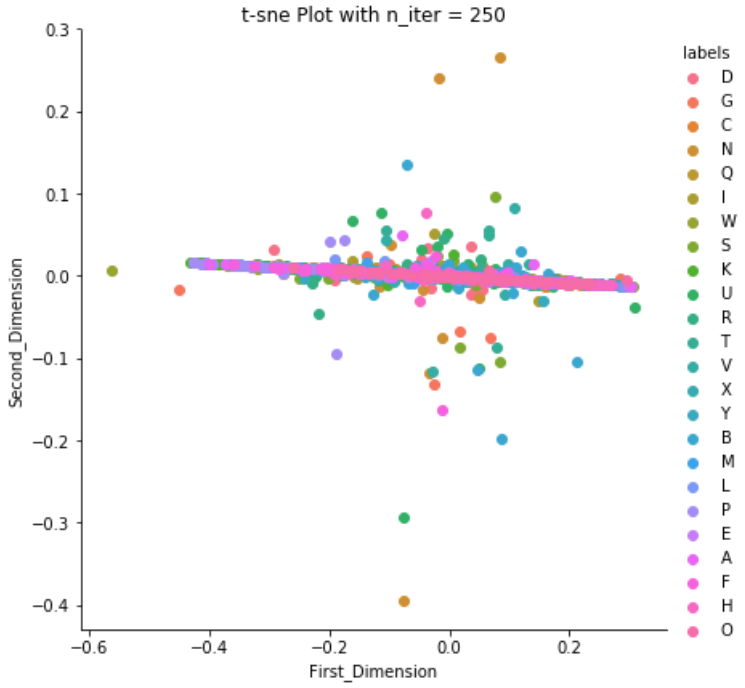
TSNE plot with n_iter = 500

In [23]:
```
from sklearn.manifold import TSNE


final_data = standardised_data
final_labels = labels

model = TSNE(n_components=2, random_state=0,perplexity=50,n_iter = 500)

tsne_data = model.fit_transform(final_data)

# Creating pandas data frame by vertically stacking out tsne_data and labels

tsne_data = np.vstack((tsne_data.T, final_labels)).T
tsne_dataframe = pd.DataFrame(data=tsne_data, columns=("First_Dimension", "Second_Dimension

# Plotting using seaborn

sns.FacetGrid(tsne_dataframe, hue="labels", size=6).map(plt.scatter, 'First_Dimension', 'Se
plt.title("t-sne Plot with n_iter = 500")
plt.show()
```



TSNE plot with n_iter = 250

In [24]:
```
from sklearn.manifold import TSNE


final_data = standardised_data
final_labels = labels

model = TSNE(n_components=2, random_state=0,perplexity=50,n_iter = 250)

tsne_data = model.fit_transform(final_data)

# Creating pandas data frame by vertically stacking out tsne_data and labels

tsne_data = np.vstack((tsne_data.T, final_labels)).T
tsne_dataframe = pd.DataFrame(data=tsne_data, columns=("First_Dimension", "Second_Dimension

# Plotting using seaborn

sns.FacetGrid(tsne_dataframe, hue="labels", size=6).map(plt.scatter, 'First_Dimension', 'Se
plt.title("t-sne Plot with n_iter = 250")
plt.show()
```



CONCLUSION

- ***The good parameters are perplexity=50 and n_iter=1000***
- t-sne->incredibly flexible & can often find structures where other dimensionality reductions techniques cannot.But,this flexibility makes it difficult to interpret.
- Algorithm makes all sort of adjustments that tidy up visualisations which sometimes make something like overfitting sometimes.