UMAP

Uniform Manifold Approximation and Projection for Dimension Reduction

---

The objective for creating this notebook is to understand UMAP intuitively, how it is better than T-SNE and where we can use it effectively.

Notebook Contents

Geometric_Intuition

- The two main parts of UMAP is to build the higher dimensional graph construction then enbedding this higher dimensional graph to lower dimension

In [1]:

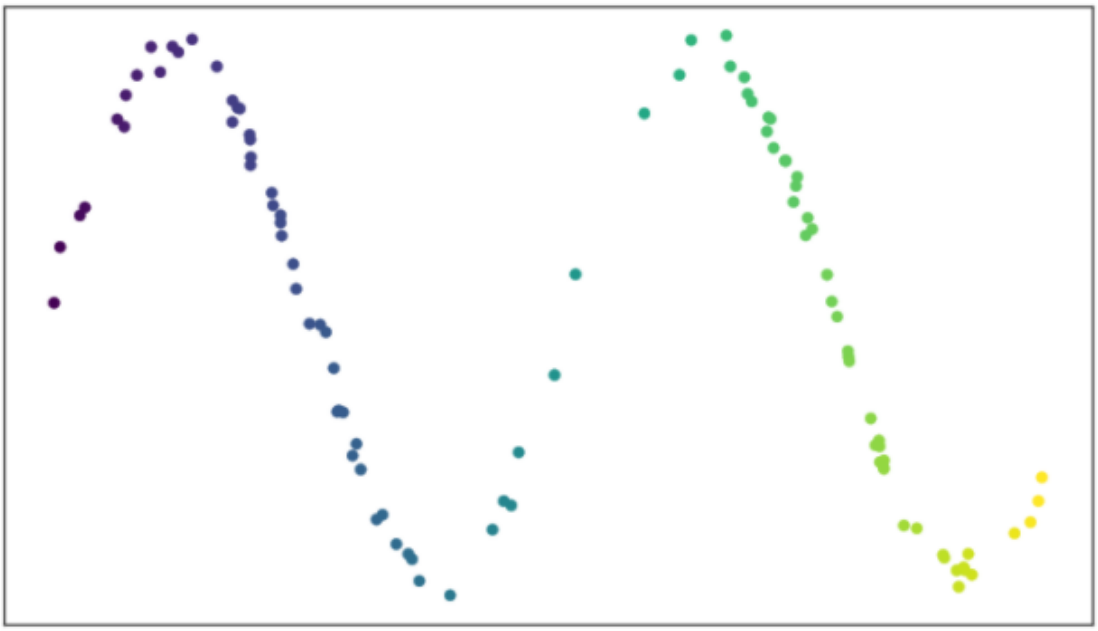
```
import numpy as np
from sklearn import decomposition
from sklearn import datasets
from babyplots import Babyplot
```

In [2]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img1.png",width=550,height=550)
```

Out[2]:

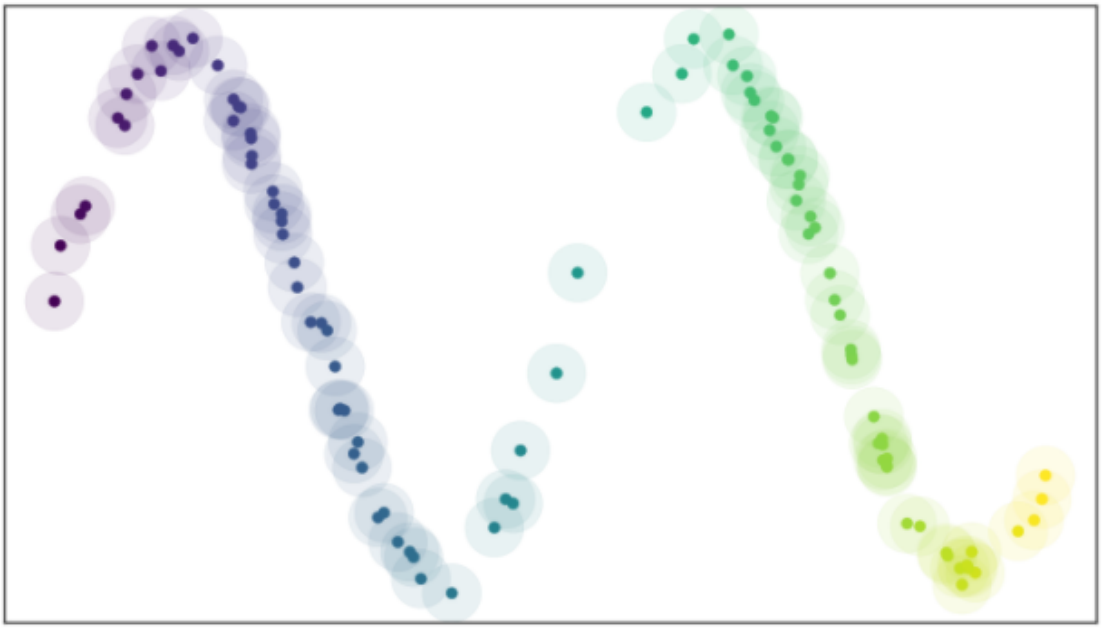


In [3]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img2.png",width=550,height=550)
```

Out[3]:



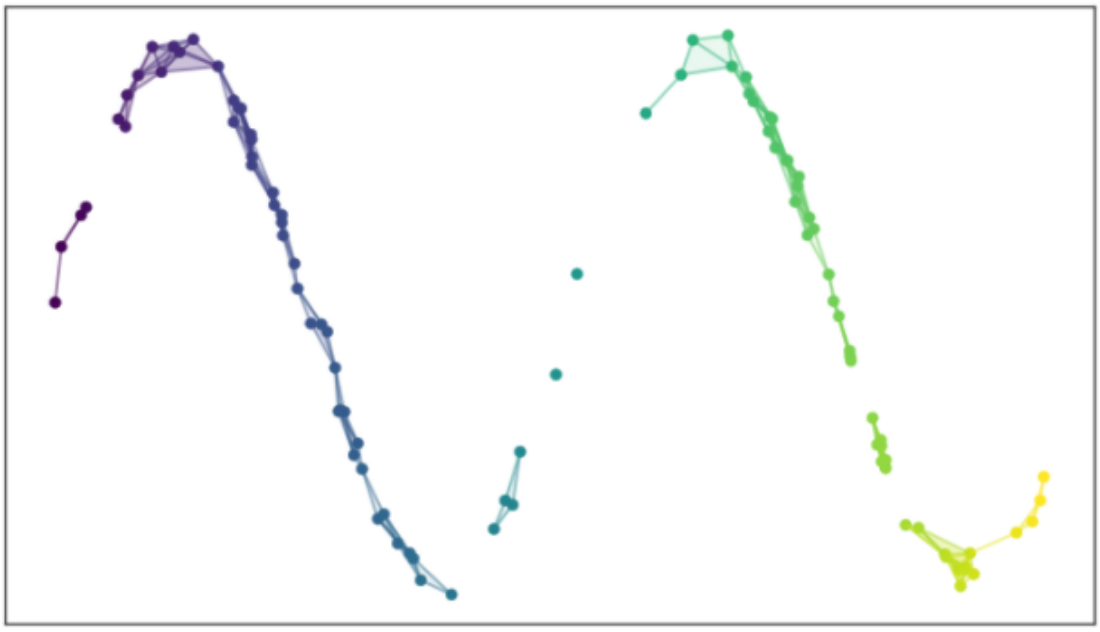
In [4]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img3.png",width=700,height=700)
```

Out[4]:

We can then depict the the simplicial complex of 0-, 1-, and 2-simplices as points, lines, and triangles



A simplicial complex built from the test data

It is harder to easily depict the higher dimensional simplices, but you can imagine how they would fit in. There are two things to note here: first, the simplicial complex does a reasonable job of starting to capture the fundamental topology of the dataset; second, most of the work is really done by the 0- and 1-simplices, which are easier to deal with computationally (it is just a graph, in the nodes and edges sense). The second observation motivates the Vietoris-Rips complex, which is similar to the Čech complex but is entirely determined by the 0- and 1-simplices. Vietoris-Rips complexes are much easier to work with computationally, especially for large datasets, and are one of the major tools of topological data analysis.

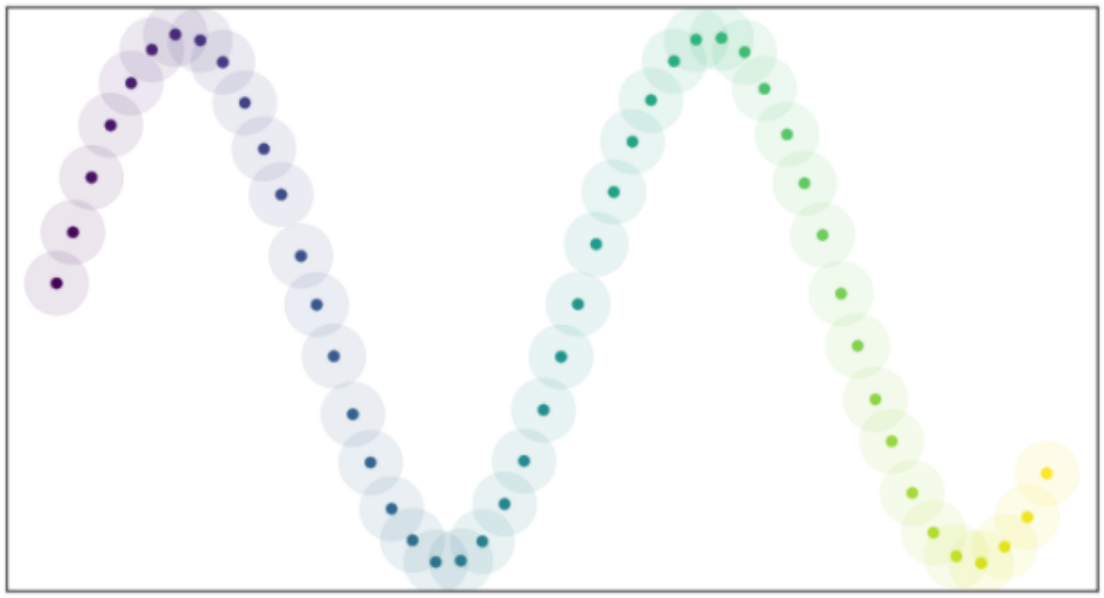
In [5]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img4.png",width=700,height=700)
```

Out[5]:

If we consider data that is uniformly distributed along the same manifold it is not hard to pick a good radius (a little above half the average distance between points) and the resulting open cover looks pretty good:



Open balls over uniformly_distributed_data

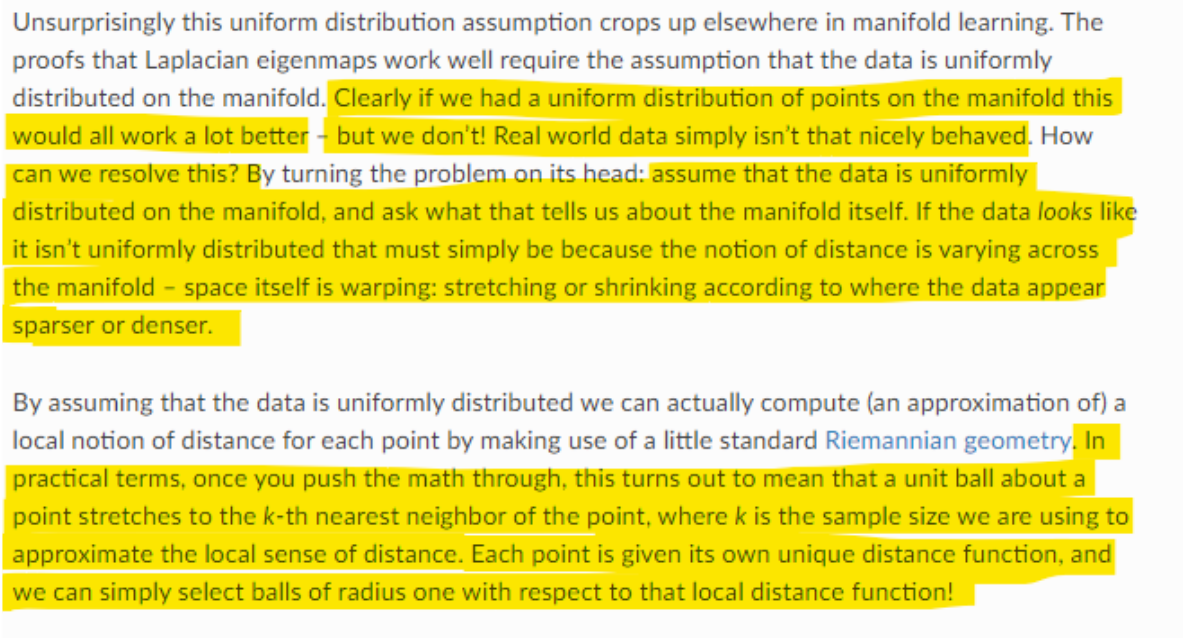
Because the data is evenly spread we actually cover the underlying manifold and don't end up with clumping. In other words, all this theory works well assuming that the data is uniformly distributed over the manifold.

In [6]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img5.png",width=700,height=700)
```

Out[6]:

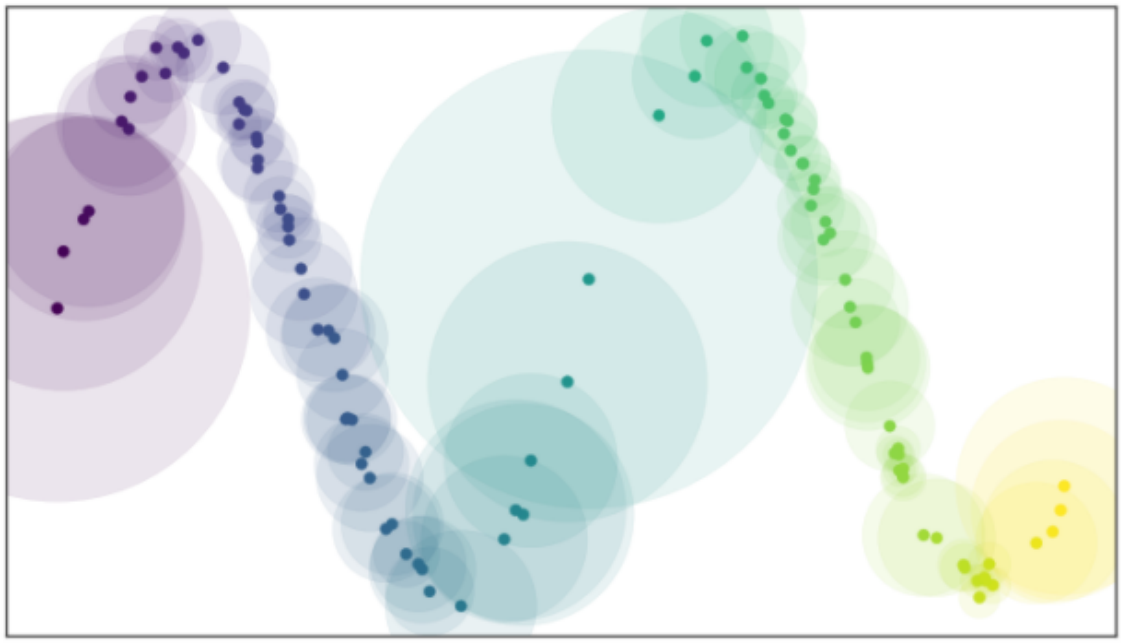


In [7]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img6.png",width=700,height=700)
```

Out[7]:



Open balls of radius one with a locally varying metric

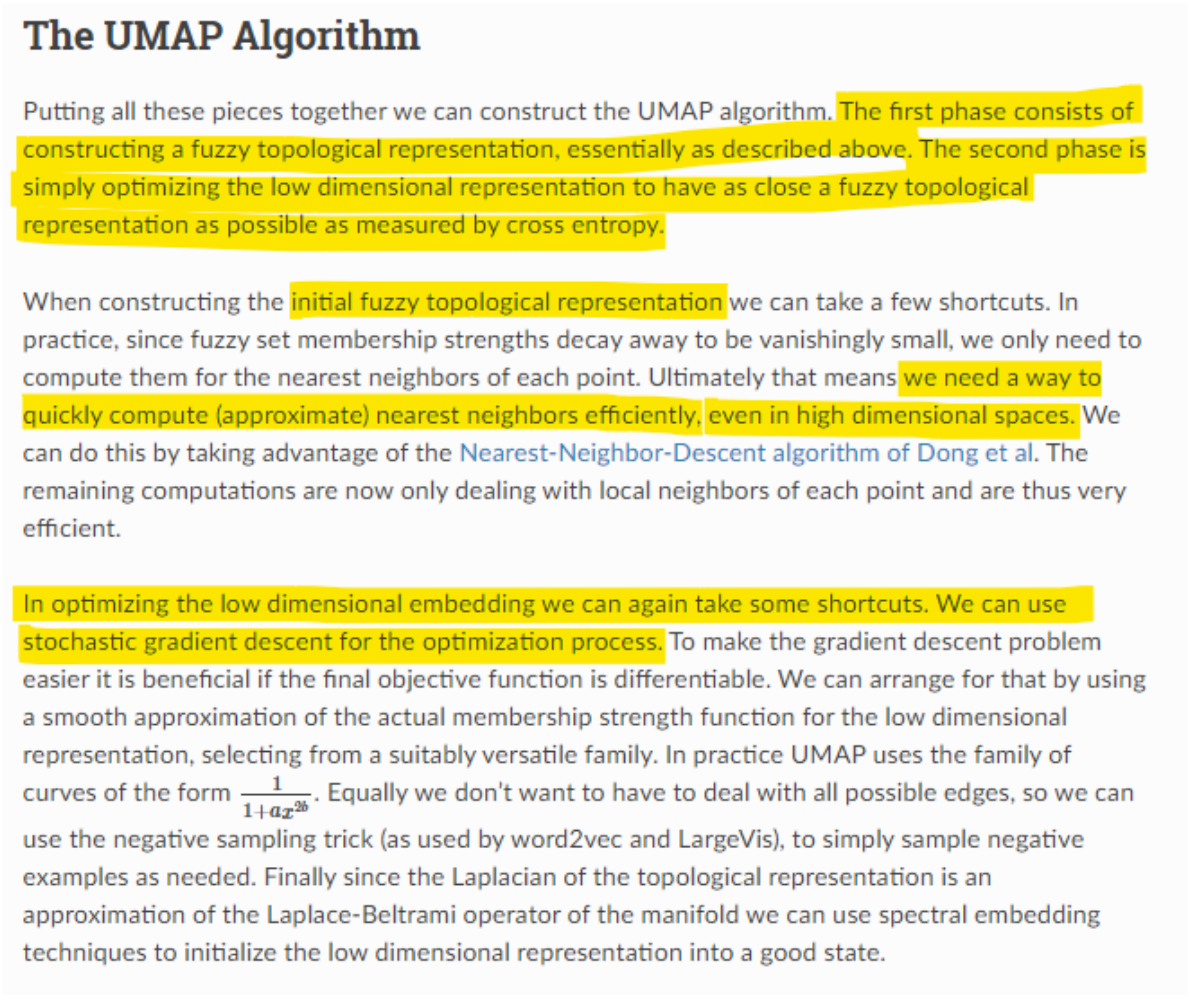
This theoretically derived result matches well with many traditional graph based algorithms: a standard approach for such algorithms is to use a k-neighbor graph instead of using balls of some fixed radius to define connectivity. What this means is that each point in the dataset is given an edge to each of its k nearest neighbors - the effective result of our locally varying metric with balls of radius one. Now, however, we can explain why this works in terms of simplicial complexes and the Nerve theorem.

In [8]:

```
from IPython.display import Image

Image("Refer_Notes\\UMAP_Images\\Img7.png",width=700,height=700)
```

Out[8]:



The UMAP Algorithm

Putting all these pieces together we can construct the UMAP algorithm. The first phase consists of constructing a fuzzy topological representation, essentially as described above. The second phase is simply optimizing the low dimensional representation to have as close a fuzzy topological representation as possible as measured by cross entropy.

When constructing the initial fuzzy topological representation we can take a few shortcuts. In practice, since fuzzy set membership strengths decay away to be vanishingly small, we only need to compute them for the nearest neighbors of each point. Ultimately that means we need a way to quickly compute (approximate) nearest neighbors efficiently, even in high dimensional spaces. We can do this by taking advantage of the Nearest-Neighbor-Descent algorithm of Dong et al. The remaining computations are now only dealing with local neighbors of each point and are thus very efficient.

In optimizing the low dimensional embedding we can again take some shortcuts. We can use stochastic gradient descent for the optimization process. To make the gradient descent problem easier it is beneficial if the final objective function is differentiable. We can arrange for that by using a smooth approximation of the actual membership strength function for the low dimensional representation, selecting from a suitably versatile family. In practice UMAP uses the family of curves of the form $\frac{1}{1+ax^{2b}}$. Equally we don't want to have to deal with all possible edges, so we can use the negative sampling trick (as used by word2vec and LargeVis), to simply sample negative examples as needed. Finally since the Laplacian of the topological representation is an approximation of the Laplace-Beltrami operator of the manifold we can use spectral embedding techniques to initialize the low dimensional representation into a good state.

Import_Packages

In [9]:

```
import umap
```

In [10]:

```
dir(umap)
```

Out[10]:

```
['UMAP',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 'distances',
 'layouts',
 'nndescent',
 'numba',
 'pkg_resources',
 'rp_tree',
 'sparse',
 'sparse_nndescent',
 'spectral',
 'umap_',
 'utils']
```

In [11]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy

from sklearn.datasets import load_breast_cancer as bc, load_iris as l_iris
from sklearn.preprocessing import StandardScaler as SS
from sklearn.manifold import TSNE

%matplotlib inline
```

UMAP_Usecase

Applying UMAP on Breast Cancer Dataset

In [12]:

```
cancer = bc()                                   ## Instantiating Breast Cancer Dataset objec
iris = l_iris()                                 ## Instantiating IRIS Datset object
ss = SS()                                       ## Instantiating Standard Scaler
tsne = TSNE(n_components=3, random_state=41)    ## Instantiating TSNE
```

In [13]:

```
cancer_df = pd.concat([pd.DataFrame(cancer.data,columns=cancer.feature_names),pd.DataFrame(
cancer_norm_df = pd.DataFrame(ss.fit_transform(cancer_df.iloc[:,0:-1]),columns=cancer.featu
cancer_norm_df.head(10)
```

Out[13]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493
5	-0.476375	-0.835335	-0.387148	-0.505650	2.237421	1.244335	0.866302	0.824656
6	1.170908	0.160649	1.138125	1.095295	-0.123136	0.088295	0.300072	0.646935
7	-0.118517	0.358450	-0.072867	-0.218965	1.604049	1.140102	0.061026	0.281950
8	-0.320167	0.588830	-0.184080	-0.384207	2.201839	1.684010	1.219096	1.150692
9	-0.473535	1.105439	-0.329482	-0.509063	1.582699	2.563358	1.738872	0.941760

10 rows × 30 columns

In [14]:

```
umap_cancer = umap.UMAP(n_neighbors=11,
                        n_components=3,
#                         metric=scipy.spatial.minkowski_distance,
#                         output_metric=scipy.spatial.minkowski_distance,
                        n_epochs=500,
                       learning_rate=0.5,
                       min_dist=1,
                       spread=2)
```

In [15]:

```
umap_cancer_cmps = pd.DataFrame(umap_cancer.fit_transform(cancer_norm_df),columns=['PC1','P
umap_cancer_cmps.head()
```

Out[15]:

	PC1	PC2	PC3
0	-2.509928	2.968512	6.418724
1	0.919889	7.963810	8.682273
2	-1.986632	4.028723	7.626415
3	0.697041	-0.168104	5.997721
4	0.023921	6.602288	6.438046

In [16]:

```
umap_cancer_cmps.shape
```

Out[16]:

```
(569, 3)
```

In [17]:

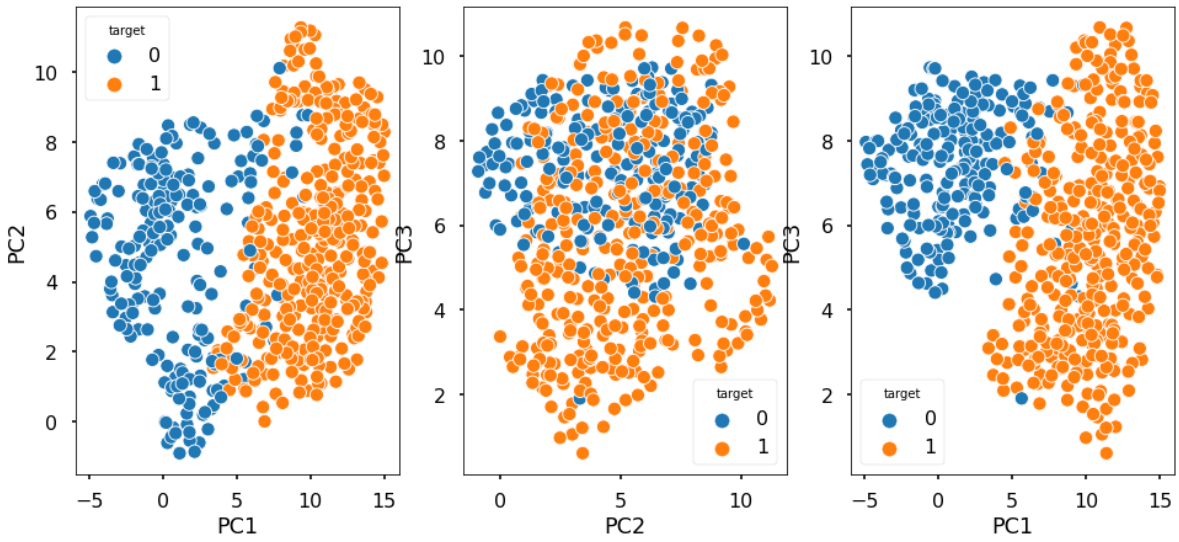
```
umap_cancer_cmps = pd.concat([umap_cancer_cmps,cancer_df['target']],axis=1)
umap_cancer_cmps.head()
```

Out[17]:

	PC1	PC2	PC3	target
0	-2.509928	2.968512	6.418724	0
1	0.919889	7.963810	8.682273	0
2	-1.986632	4.028723	7.626415	0
3	0.697041	-0.168104	5.997721	0
4	0.023921	6.602288	6.438046	0

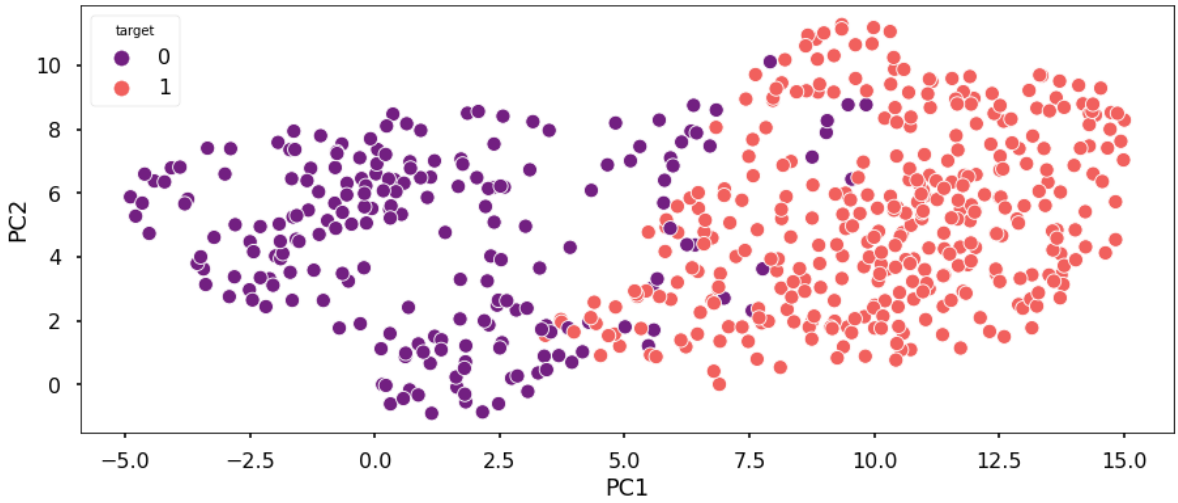
In [18]:

```
with plt.style.context('seaborn-poster'):
    fig, ax = plt.subplots(nrows=1,ncols=3,figsize=(16,7))
    sns.scatterplot(data=umap_cancer_cmps,x='PC1',y='PC2',hue='target',ax=ax[0])
    sns.scatterplot(data=umap_cancer_cmps,x='PC2',y='PC3',hue='target',ax=ax[1])
    sns.scatterplot(data=umap_cancer_cmps,x='PC1',y='PC3',hue='target',ax=ax[2])
    plt.show()
```



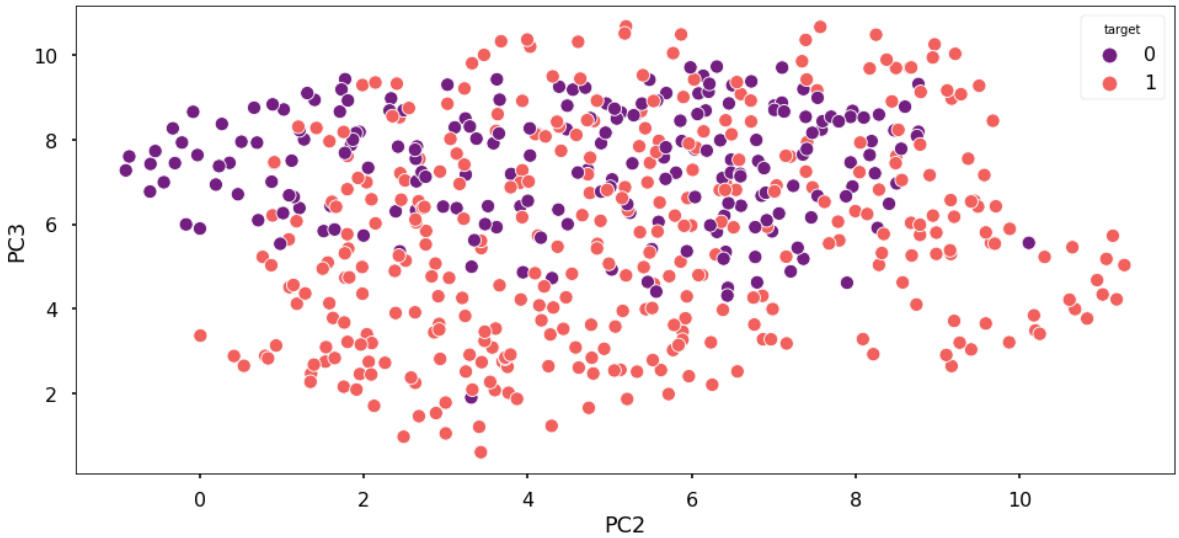
In [19]:

```
with plt.style.context('seaborn-poster'):
    fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(15,6))
    sns.scatterplot(data=umap_cancer_cmps,x='PC1',y='PC2',hue='target',ax=ax,palette='magma
    plt.show();
```



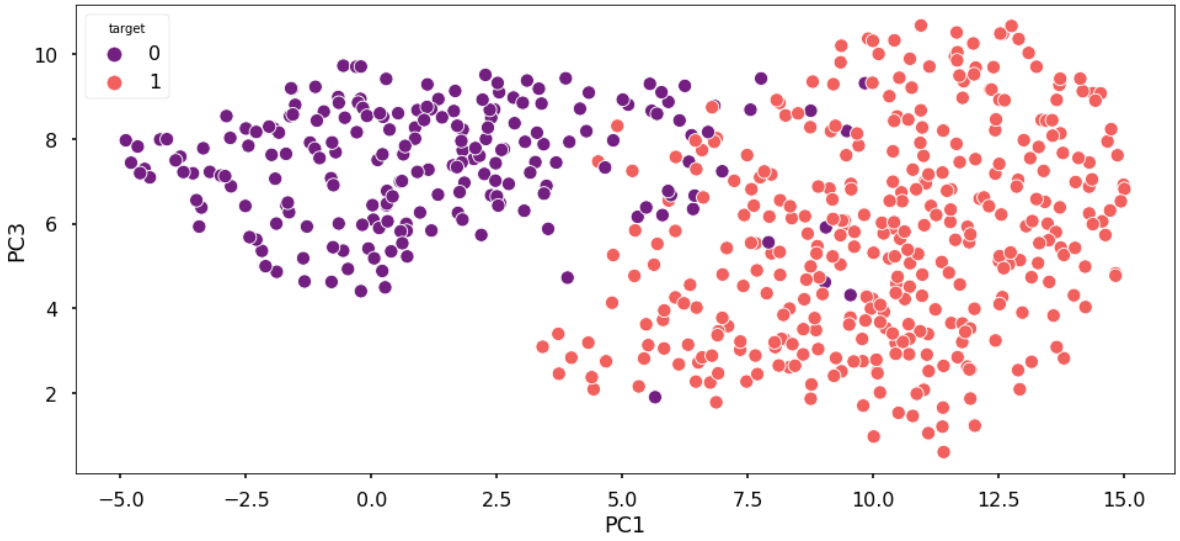
In [20]:

```
with plt.style.context('seaborn-poster'):
    fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(16,7))
    sns.scatterplot(data=umap_cancer_cmps,x='PC2',y='PC3',hue='target',ax=ax,palette='magma
    plt.show()
```



In [21]:

```
with plt.style.context('seaborn-poster'):
    fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(16,7))
    sns.scatterplot(data=umap_cancer_cmps,x='PC1',y='PC3',hue='target',ax=ax,palette='magma
    plt.show()
```



In [22]:

```
umap_cancer_cmps
```

Out[22]:

	PC1	PC2	PC3	target
0	-2.509928	2.968512	6.418724	0
1	0.919889	7.963810	8.682273	0
2	-1.986632	4.028723	7.626415	0
3	0.697041	-0.168104	5.997721	0
4	0.023921	6.602288	6.438046	0
...	...	...	...	...
564	-2.999779	6.595969	7.136393	0
565	-0.755743	7.239864	6.910088	0
566	4.657914	6.883886	7.327774	0
567	-2.912424	2.754453	7.123294	0
568	12.518476	8.742889	4.102712	1

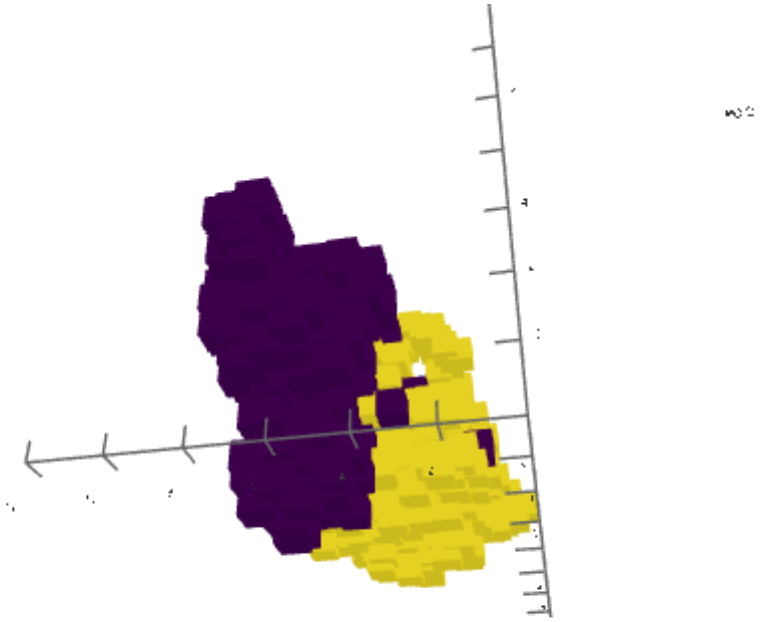
569 rows × 4 columns

In [41]:

```
X = umap_cancer_cmps.iloc[:,0:-1].values
y = umap_cancer_cmps.iloc[:,-1].values

# create the babyplots visualization
bp = Babyplot(width=550,height=400,show_ui=False,turntable=True,background_color='#f7ffff')
bp.add_plot(X.tolist(), "shapeCloud", "categories", y.tolist(), {"shape": "cone,box",
                                                                 "colorScale": "viridis",
                                                                 "showAxes": [True, True, T
                                                                 "showlegend":True,
                                                                 "fontsize":16,
                                                                 "axisLabels": ["PC 1", "PC
# show t
bp
```

Out[41]:



:: Reference Links ::

UMAP

- https://www.youtube.com/watch?v=6BPl81wGGP8 (https://www.youtube.com/watch?v=6BPl81wGGP8)
- https://www.sciencedirect.com/science/article/pii/S0010482521000585 (https://www.sciencedirect.com/science/article/pii/S0010482521000585)
- https://umap-learn.readthedocs.io/en/latest/parameters.html (https://umap-learn.readthedocs.io/en/latest/parameters.html)
- https://umap-learn.readthedocs.io/en/latest/reproducibility.html (https://umap-learn.readthedocs.io/en/latest/reproducibility.html)
- https://pair-code.github.io/understanding-umap/ (https://pair-code.github.io/understanding-umap/)

UMAP v/s T-SNE

- https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668#:~:text=We%20know%20that%20UMAP%20is,dimensions%20in%20the%20data%20set.&te (https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668#:~:text=We%20know%20that%20UMAP%20is,dimensions%20in%20the%20data%20set.&te

T-SNE

- https://lvdmaaten.github.io/tsne/ (https://lvdmaaten.github.io/tsne/)

Barnes-Hut T-SNE

- https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf (https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf)
- https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html (https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html)
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6402590/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6402590/)
- https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf (https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf)

UMAP uses PyNNDescent

- https://pypi.org/project/pynndescent/ (https://pypi.org/project/pynndescent/)