03_manifold_learning_tsne_umap

September 29, 2021

1 t-SNE and UMAP

[1]:
```
%matplotlib inline
from pathlib import Path
from os.path import join
import pandas as pd
import numpy as np
from numpy.random import choice, randint, uniform, randn
import seaborn as sns
import matplotlib.pyplot as plt
import ipyvolume as ipv
from sklearn.datasets import fetch_openml, make_swiss_roll, make_blobs
from sklearn.manifold import TSNE
import umap
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
import colorlover as cl
import warnings
```

[2]:
```
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
pd.options.display.float_format = '{:,.2f}'.format
init_notebook_mode(connected=True)
ipv_cmap = sns.color_palette("Paired", n_colors=10)
```

1.1 T-Stochastic Neighbor Embedding (TSNE): Parameter Settings

t-SNE is an award-winning algorithm developed in 2010 by Laurens van der Maaten and Geoff Hinton to detect patterns in high-dimensional data. It takes a probabilistic, non-linear approach to locating data on several different, but related low-dimensional manifolds. The algorithm emphasizes keeping similar points together in low dimensions, as opposed to maintaining the distance between points that are apart in high dimensions, which results from algorithms like PCA that minimize squared distances.

The algorithm proceeds by converting high-dimensional distances to (conditional) probabilities, where high probabilities imply low distance and reflect the likelihood of sampling two points based on similarity. It accomplishes this by positioning a normal distribution over each point and computing the density for a point and each neighbor, where the perplexity parameter controls the

effective number of neighbors. In a second step, it arranges points in low dimensions and uses similarly computed low-dimensional probabilities to match the high-dimensional distribution. It measures the difference between the distributions using the Kullback-Leibler divergence that puts a high penalty on misplacing similar points in low dimensions. The low-dimensional probabilities use a Student-t distribution with one degree of freedom because it has fatter tails that reduce the penalty of misplacing points that are more distant in high dimensions to manage the crowding problem.

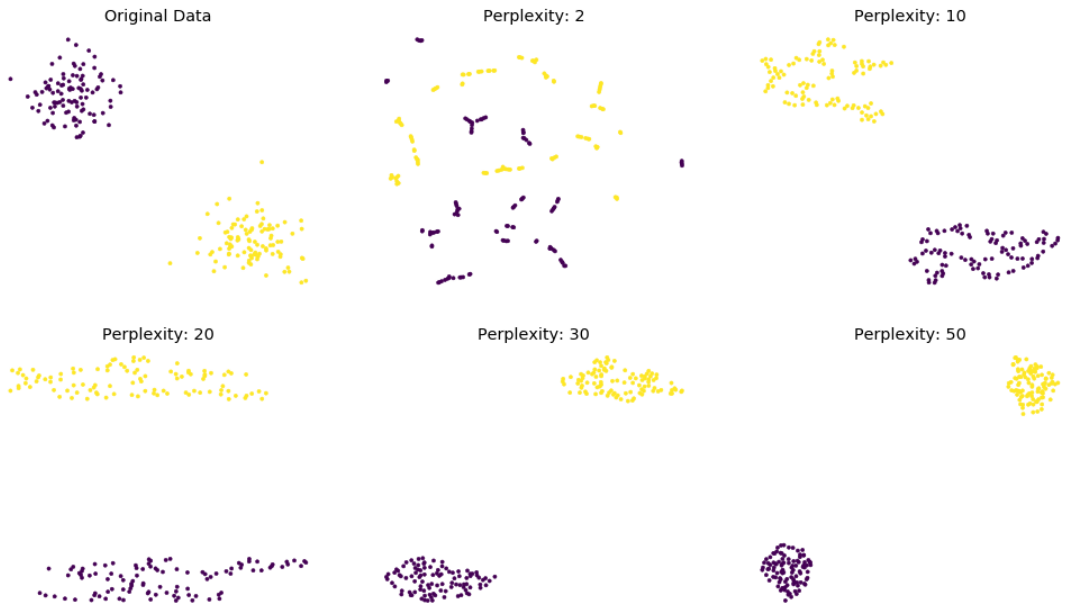
t-SNE is currently the state-of-the-art in high-dimensional data visualization. Weaknesses include the computational complexity that scales quadratically in the number n of points because it evaluates all pairwise distances, but a subsequent tree-based implementation has reduced the cost to n log n.

t-SNE does not facilitate the projection of new data points into the low-dimensional space. The compressed output is not a very useful input for distance- or density-based cluster algorithms because t-SNE treats small and large distances differently.

1.1.1 Perplexity: emphasis on local vs global structure

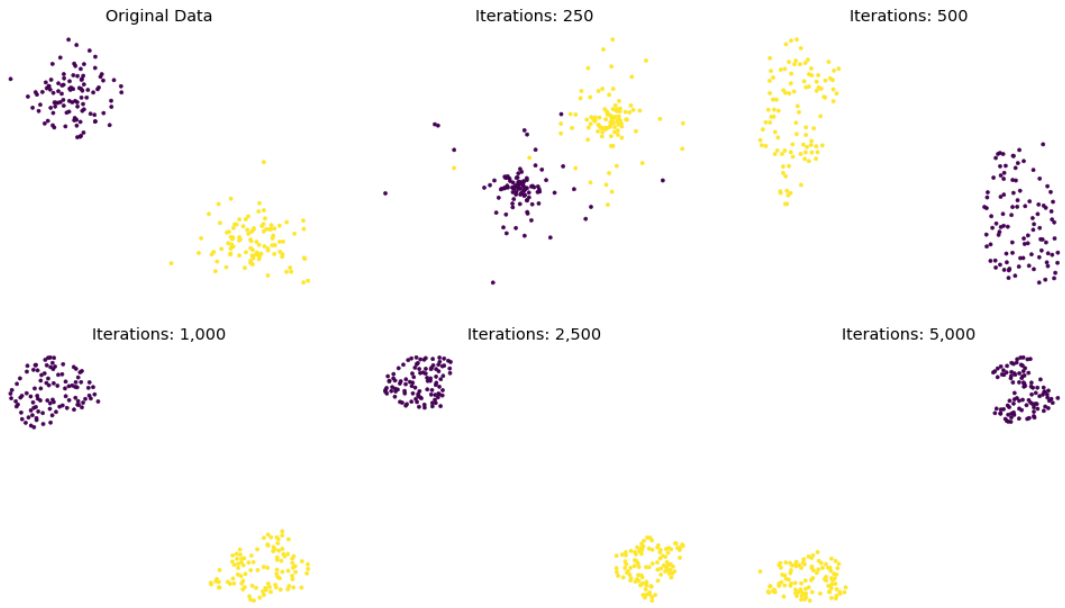
[3]:
```
data, label = make_blobs(n_samples=200, n_features=2, centers=2,␣
↪random_state=42)
```

[4]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()
axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, p in enumerate([2, 10, 20, 30, 50], 1):
    embedding = TSNE(perplexity=p, n_iter=5000).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Perplexity: {:.0f}'.format(p))
    axes[i].axis('off')
fig.tight_layout()
```



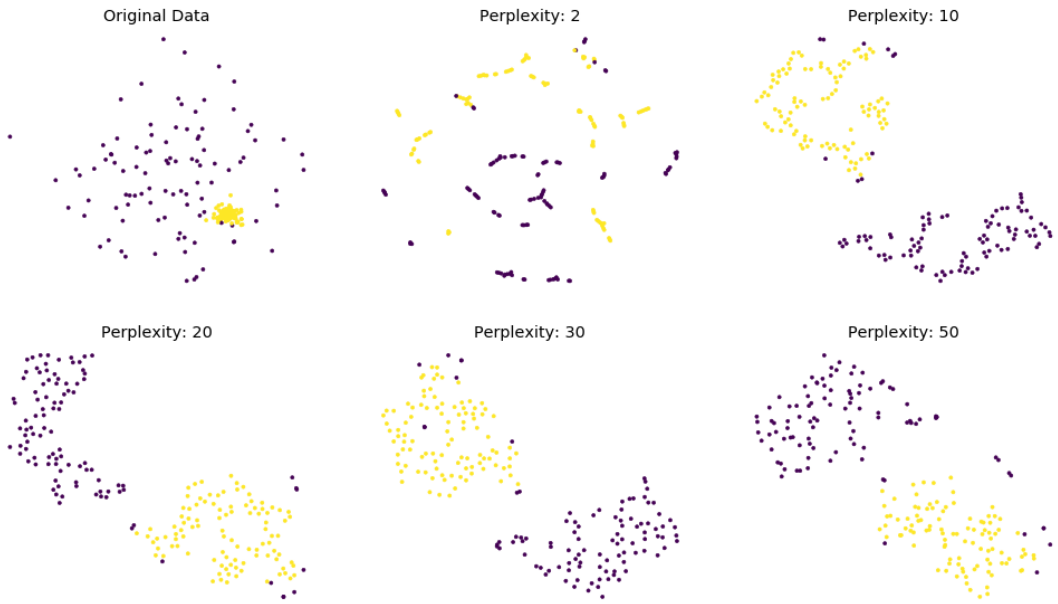
1.1.2 Convergence with `n_iter`

[5]:
```
data, label = make_blobs(n_samples=200, n_features=2, centers=2,␣
↪random_state=42)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()
axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, n in enumerate([250, 500, 1000, 2500, 5000], 1):
    embedding = TSNE(perplexity=30, n_iter=n).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Iterations: {:,.0f}'.format(n))
    axes[i].axis('off')
fig.tight_layout();
```



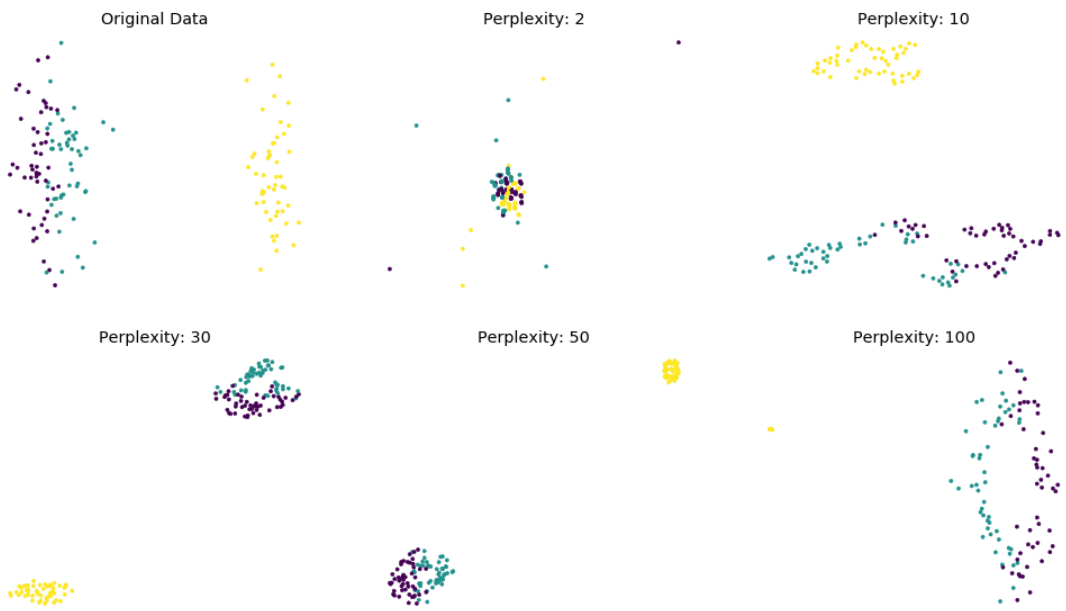
1.1.3 Different Cluster Sizes

[6]:
```
data, label = make_blobs(n_samples=200, n_features=2, cluster_std=[10, 1],␣
↪centers=2, random_state=42)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()
axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, p in enumerate([2,10, 20, 30, 50], 1):
    embedding = TSNE(perplexity=p, n_iter=5000).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Perplexity: {:.0f}'.format(p))
    axes[i].axis('off')
fig.tight_layout();
```



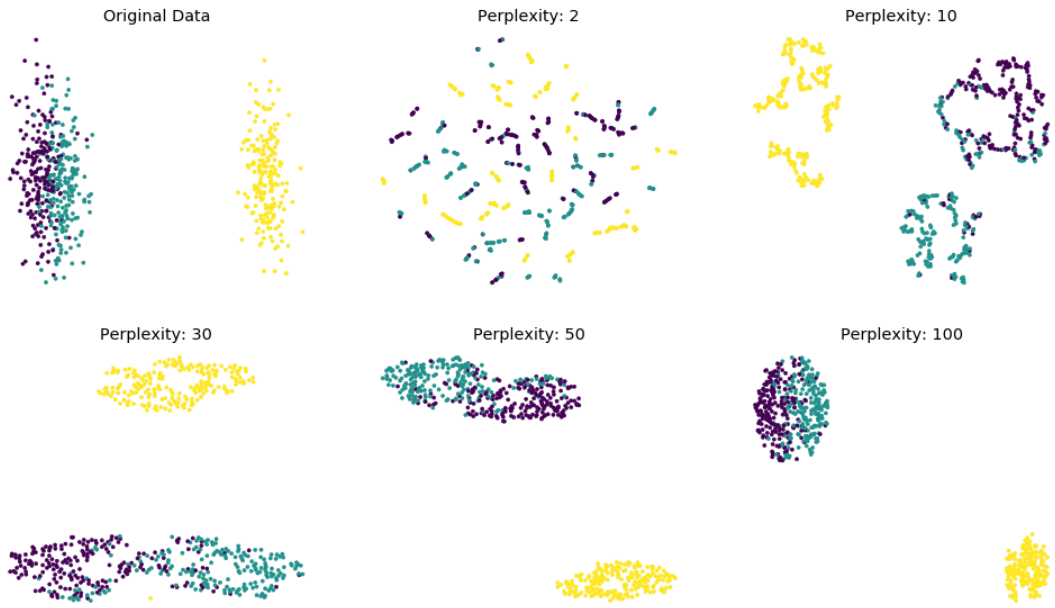
1.1.4 Different Cluster Distances

[7]:
```
data, label = make_blobs(n_samples=150, n_features=2, centers=[[-10, 0], [-8,␣
↪0], [10, 0]], random_state=2)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()
axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, p in enumerate([2,10, 30, 50, 100], 1):
    embedding = TSNE(perplexity=p, n_iter=5000).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Perplexity: {:.0f}'.format(p))
    axes[i].axis('off')
fig.tight_layout();
```



1.1.5 More points require higher perplexity

[8]:
```
data, label = make_blobs(n_samples=600, n_features=2, centers=[[-10, 0], [-8,␣
↪0], [10, 0]], random_state=2)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()
axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, p in enumerate([2,10, 30, 50, 100], 1):
    embedding = TSNE(perplexity=p, n_iter=5000).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Perplexity: {:.0f}'.format(p))
    axes[i].axis('off')
fig.tight_layout();
```



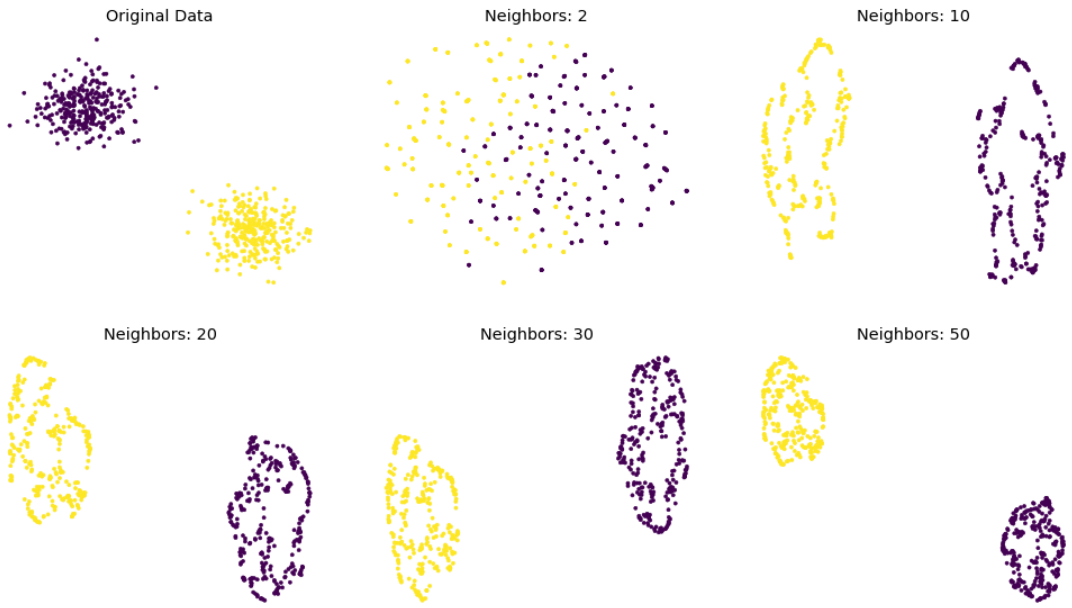
1.2 Uniform Manifold Approximation and Projection (UMAP): Parameter Settings

UMAP is a more recent algorithm for visualization and general dimensionality reduction. It assumes the data is uniformly distributed on a locally connected manifold and looks for the closest low-dimensional equivalent using fuzzy topology. It uses a neighbors parameter that impacts the result similarly as perplexity above.

It is faster and hence scales better to large datasets than t-SNE, and sometimes preserves global structure than better than t-SNE. It can also work with different distance functions, including, e.g., cosine similarity that is used to measure the distance between word count vectors.

1.2.1 Neighbors

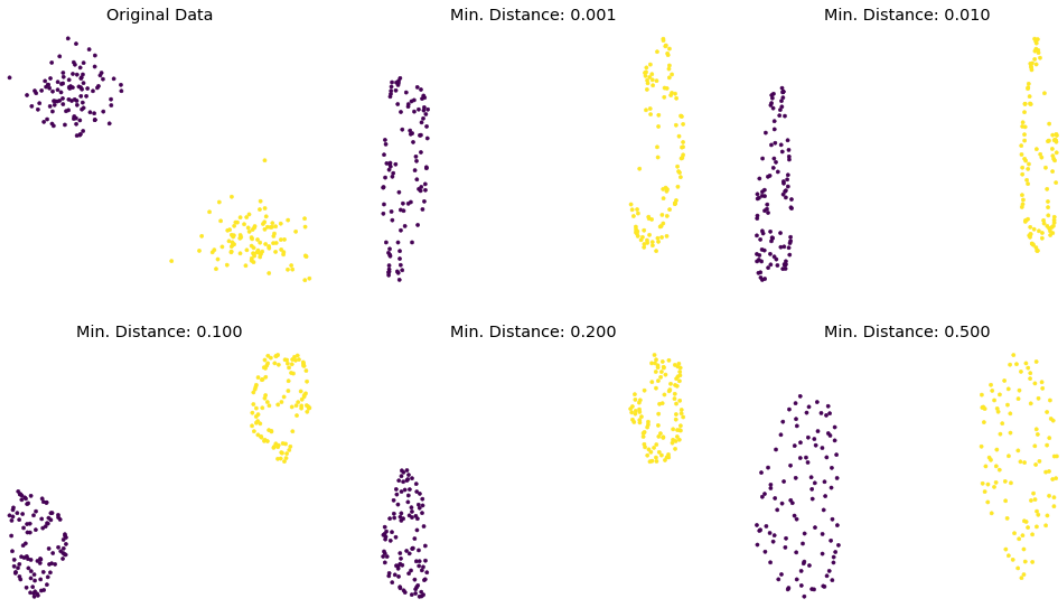
[9]:
```
data, label = make_blobs(n_samples=600, n_features=2, centers=2,␣
↪random_state=42)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()
axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, n in enumerate([2,10, 20, 30, 50], 1):
    embedding = umap.UMAP(n_neighbors=n, min_dist=0.1).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Neighbors: {:.0f}'.format(n))
    axes[i].axis('off')
fig.tight_layout();
```



1.2.2 Minimum Distance

[10]:
```
data, label = make_blobs(n_samples=200, n_features=2, centers=2,␣
↪random_state=42)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))
axes = axes.flatten()

axes[0].scatter(data[:, 0], data[:, 1], s=10, c=label)
axes[0].set_title('Original Data')
axes[0].axis('off')
for i, d in enumerate([.001, .01, .1, .2, .5], 1):
    embedding = umap.UMAP(n_neighbors=30, min_dist=d).fit_transform(data)
    axes[i].scatter(embedding[:, 0], embedding[:, 1], s=10, c=label)
    axes[i].set_title('Min. Distance: {:.3f}'.format(d))
    axes[i].axis('off')
fig.tight_layout();
```



1.3 Non-Linear Manifolds: Swiss Roll

[11]:
```
n_samples = 10000
palette = sns.color_palette('viridis', n_colors=n_samples)
```

[12]:
```
zeros = np.zeros(n_samples) + .5
swiss_3d, swiss_val = make_swiss_roll(
    n_samples=n_samples, noise=.1, random_state=42)

swiss_3d = swiss_3d[swiss_val.argsort()[::-1]]
x, y, z = swiss_3d.T
```

1.3.1 TSNE

Using pre-computed T-SNE and UMAP results due to the long running times, esp. for T-SNE.

[13]:
```
# pre-computed manifold results for the various datasets and algorithms, as␣
 ↪well as parameter settings:
with pd.HDFStore(Path('data', 'manifolds.h5')) as store:
    print(store.info())
```

```
<class 'pandas.io.pytables.HDFStore'>
File path: data/manifolds.h5
/fashion/labels                                 series       (shape->[12000])
/fashion/lle/2/10                               frame        (shape->[12000,2])
/fashion/lle/2/10/stats                         series       (shape->[2])
/fashion/lle/2/15                               frame        (shape->[12000,2])
```

```
/fashion/lle/2/15/stats                 series      (shape->[2])
/fashion/lle/2/20                       frame       (shape->[12000,2])
/fashion/lle/2/20/stats                 series      (shape->[2])
/fashion/lle/2/25                       frame       (shape->[12000,2])
/fashion/lle/2/25/stats                 series      (shape->[2])
/fashion/lle/2/30                       frame       (shape->[12000,2])
/fashion/lle/2/30/stats                 series      (shape->[2])
/fashion/lle/2/35                       frame       (shape->[12000,2])
/fashion/lle/2/35/stats                 series      (shape->[2])
/fashion/lle/2/40                       frame       (shape->[12000,2])
/fashion/lle/2/40/stats                 series      (shape->[2])
/fashion/lle/2/45                       frame       (shape->[12000,2])
/fashion/lle/2/45/stats                 series      (shape->[2])
/fashion/lle/2/5                        frame       (shape->[12000,2])
/fashion/lle/2/5/stats                  series      (shape->[2])
/fashion/lle/3/10                       frame       (shape->[12000,3])
/fashion/lle/3/10/stats                 series      (shape->[2])
/fashion/lle/3/15                       frame       (shape->[12000,3])
/fashion/lle/3/15/stats                 series      (shape->[2])
/fashion/lle/3/20                       frame       (shape->[12000,3])
/fashion/lle/3/20/stats                 series      (shape->[2])
/fashion/lle/3/25                       frame       (shape->[12000,3])
/fashion/lle/3/25/stats                 series      (shape->[2])
/fashion/lle/3/30                       frame       (shape->[12000,3])
/fashion/lle/3/30/stats                 series      (shape->[2])
/fashion/lle/3/35                       frame       (shape->[12000,3])
/fashion/lle/3/35/stats                 series      (shape->[2])
/fashion/lle/3/40                       frame       (shape->[12000,3])
/fashion/lle/3/40/stats                 series      (shape->[2])
/fashion/lle/3/45                       frame       (shape->[12000,3])
/fashion/lle/3/45/stats                 series      (shape->[2])
/fashion/lle/3/5                        frame       (shape->[12000,3])
/fashion/lle/3/5/stats                  series      (shape->[2])
/fashion/lle/modified/2/100             frame       (shape->[12000,2])
/fashion/lle/modified/2/100/stats       series      (shape->[2])
/fashion/lle/modified/2/200             frame       (shape->[12000,2])
/fashion/lle/modified/2/200/stats       series      (shape->[2])
/fashion/lle/modified/2/500             frame       (shape->[12000,2])
/fashion/lle/modified/2/500/stats       series      (shape->[2])
/fashion/lle/standard/2/10              frame       (shape->[12000,2])
/fashion/lle/standard/2/10/stats        series      (shape->[2])
/fashion/lle/standard/2/100             frame       (shape->[12000,2])
/fashion/lle/standard/2/100/stats       series      (shape->[2])
/fashion/lle/standard/2/15              frame       (shape->[12000,2])
/fashion/lle/standard/2/15/stats        series      (shape->[2])
/fashion/lle/standard/2/20              frame       (shape->[12000,2])
/fashion/lle/standard/2/20/stats        series      (shape->[2])
/fashion/lle/standard/2/200             frame       (shape->[12000,2])
```

```
/fashion/lle/standard/2/200/stats          series      (shape->[2])
/fashion/lle/standard/2/25                 frame       (shape->[12000,2])
/fashion/lle/standard/2/25/stats           series      (shape->[2])
/fashion/lle/standard/2/30                 frame       (shape->[12000,2])
/fashion/lle/standard/2/30/stats           series      (shape->[2])
/fashion/lle/standard/2/35                 frame       (shape->[12000,2])
/fashion/lle/standard/2/35/stats           series      (shape->[2])
/fashion/lle/standard/2/40                 frame       (shape->[12000,2])
/fashion/lle/standard/2/40/stats           series      (shape->[2])
/fashion/lle/standard/2/45                 frame       (shape->[12000,2])
/fashion/lle/standard/2/45/stats           series      (shape->[2])
/fashion/lle/standard/2/5                  frame       (shape->[12000,2])
/fashion/lle/standard/2/5/stats            series      (shape->[2])
/fashion/lle/standard/2/500                frame       (shape->[12000,2])
/fashion/lle/standard/2/500/stats          series      (shape->[2])
/fashion/lle/standard/3/10                 frame       (shape->[12000,3])
/fashion/lle/standard/3/10/stats           series      (shape->[2])
/fashion/lle/standard/3/15                 frame       (shape->[12000,3])
/fashion/lle/standard/3/15/stats           series      (shape->[2])
/fashion/lle/standard/3/20                 frame       (shape->[12000,3])
/fashion/lle/standard/3/20/stats           series      (shape->[2])
/fashion/lle/standard/3/25                 frame       (shape->[12000,3])
/fashion/lle/standard/3/25/stats           series      (shape->[2])
/fashion/lle/standard/3/30                 frame       (shape->[12000,3])
/fashion/lle/standard/3/30/stats           series      (shape->[2])
/fashion/lle/standard/3/35                 frame       (shape->[12000,3])
/fashion/lle/standard/3/35/stats           series      (shape->[2])
/fashion/lle/standard/3/40                 frame       (shape->[12000,3])
/fashion/lle/standard/3/40/stats           series      (shape->[2])
/fashion/lle/standard/3/45                 frame       (shape->[12000,3])
/fashion/lle/standard/3/45/stats           series      (shape->[2])
/fashion/lle/standard/3/5                  frame       (shape->[12000,3])
/fashion/lle/standard/3/5/stats            series      (shape->[2])
/fashion/tsne/2/10                         frame       (shape->[12000,2])
/fashion/tsne/2/10/stats                   series      (shape->[2])
/fashion/tsne/2/15                         frame       (shape->[12000,2])
/fashion/tsne/2/15/stats                   series      (shape->[2])
/fashion/tsne/2/20                         frame       (shape->[12000,2])
/fashion/tsne/2/20/stats                   series      (shape->[2])
/fashion/tsne/2/25                         frame       (shape->[12000,2])
/fashion/tsne/2/25/stats                   series      (shape->[2])
/fashion/tsne/2/30                         frame       (shape->[12000,2])
/fashion/tsne/2/30/stats                   series      (shape->[2])
/fashion/tsne/2/35                         frame       (shape->[12000,2])
/fashion/tsne/2/35/stats                   series      (shape->[2])
/fashion/tsne/2/40                         frame       (shape->[12000,2])
/fashion/tsne/2/40/stats                   series      (shape->[2])
/fashion/tsne/2/45                         frame       (shape->[12000,2])
```

```
/fashion/tsne/2/45/stats                series      (shape->[2])
/fashion/tsne/2/5                       frame       (shape->[12000,2])
/fashion/tsne/2/5/stats                 series      (shape->[2])
/fashion/tsne/2/50                      frame       (shape->[12000,2])
/fashion/tsne/2/50/stats                series      (shape->[2])
/fashion/tsne/3/10                      frame       (shape->[12000,3])
/fashion/tsne/3/10/stats                series      (shape->[2])
/fashion/tsne/3/15                      frame       (shape->[12000,3])
/fashion/tsne/3/15/stats                series      (shape->[2])
/fashion/tsne/3/20                      frame       (shape->[12000,3])
/fashion/tsne/3/20/stats                series      (shape->[2])
/fashion/tsne/3/25                      frame       (shape->[12000,3])
/fashion/tsne/3/25/stats                series      (shape->[2])
/fashion/tsne/3/30                      frame       (shape->[12000,3])
/fashion/tsne/3/30/stats                series      (shape->[2])
/fashion/tsne/3/35                      frame       (shape->[12000,3])
/fashion/tsne/3/35/stats                series      (shape->[2])
/fashion/tsne/3/40                      frame       (shape->[12000,3])
/fashion/tsne/3/40/stats                series      (shape->[2])
/fashion/tsne/3/45                      frame       (shape->[12000,3])
/fashion/tsne/3/45/stats                series      (shape->[2])
/fashion/tsne/3/5                       frame       (shape->[12000,3])
/fashion/tsne/3/5/stats                 series      (shape->[2])
/fashion/tsne/3/50                      frame       (shape->[12000,3])
/fashion/tsne/3/50/stats                series      (shape->[2])
/fashion/umap/2/15                      frame       (shape->[12000,2])
/fashion/umap/2/15/stats                series      (shape->[1])
/fashion/umap/2/25                      frame       (shape->[12000,2])
/fashion/umap/2/25/stats                series      (shape->[1])
/fashion/umap/2/35                      frame       (shape->[12000,2])
/fashion/umap/2/35/stats                series      (shape->[1])
/fashion/umap/2/5                       frame       (shape->[12000,2])
/fashion/umap/2/5/stats                 series      (shape->[1])
/mnist/labels                           series      (shape->[14000])
/mnist/lle/modified/2/100               frame       (shape->[14000,2])
/mnist/lle/modified/2/100/stats         series      (shape->[2])
/mnist/lle/modified/2/20                frame       (shape->[14000,2])
/mnist/lle/modified/2/200               frame       (shape->[14000,2])
/mnist/lle/modified/2/200/stats         series      (shape->[2])
/mnist/lle/modified/2/50                frame       (shape->[14000,2])
/mnist/lle/modified/3/20                frame       (shape->[14000,3])
/mnist/lle/standard/2/100               frame       (shape->[14000,2])
/mnist/lle/standard/2/100/stats         series      (shape->[2])
/mnist/lle/standard/2/20                frame       (shape->[14000,2])
/mnist/lle/standard/2/200               frame       (shape->[14000,2])
/mnist/lle/standard/2/200/stats         series      (shape->[2])
/mnist/lle/standard/2/5                 frame       (shape->[14000,2])
/mnist/lle/standard/2/50                frame       (shape->[14000,2])
```

/mnist/lle/standard/3/20	frame	(shape->[14000,3])
/mnist/lle/standard/3/5	frame	(shape->[14000,3])
/mnist/tsne/2/10	frame	(shape->[14000,2])
/mnist/tsne/2/15	frame	(shape->[14000,2])
/mnist/tsne/2/20	frame	(shape->[14000,2])
/mnist/tsne/2/25	frame	(shape->[14000,2])
/mnist/tsne/2/30	frame	(shape->[14000,2])
/mnist/tsne/2/35	frame	(shape->[14000,2])
/mnist/tsne/2/40	frame	(shape->[14000,2])
/mnist/tsne/2/45	frame	(shape->[14000,2])
/mnist/tsne/2/5	frame	(shape->[14000,2])
/mnist/tsne/2/50	frame	(shape->[14000,2])
/mnist/umap/2/15	frame	(shape->[14000,2])
/mnist/umap/2/15/stats	series	(shape->[1])
/mnist/umap/2/25	frame	(shape->[14000,2])
/mnist/umap/2/25/stats	series	(shape->[1])
/mnist/umap/2/35	frame	(shape->[14000,2])
/mnist/umap/2/35/stats	series	(shape->[1])
/mnist/umap/2/5	frame	(shape->[14000,2])
/mnist/umap/2/5/stats	series	(shape->[1])
/scurve/lle/modified/10	frame	(shape->[10000,2])
/scurve/lle/modified/20	frame	(shape->[10000,2])
/scurve/lle/modified/30	frame	(shape->[10000,2])
/scurve/lle/modified/40	frame	(shape->[10000,2])
/scurve/lle/modified/5	frame	(shape->[10000,2])
/scurve/lle/modified/50	frame	(shape->[10000,2])
/scurve/lle/standard/10	frame	(shape->[10000,2])
/scurve/lle/standard/20	frame	(shape->[10000,2])
/scurve/lle/standard/30	frame	(shape->[10000,2])
/scurve/lle/standard/40	frame	(shape->[10000,2])
/scurve/lle/standard/5	frame	(shape->[10000,2])
/scurve/lle/standard/50	frame	(shape->[10000,2])
/swiss/Cosine PCA	frame	(shape->[10000,2])
/swiss/Hession Eigenmap	frame	(shape->[10000,2])
/swiss/ICA	frame	(shape->[10000,3])
/swiss/IsoMap	frame	(shape->[10000,2])
/swiss/LLE	frame	(shape->[10000,2])
/swiss/MDS	frame	(shape->[10000,2])
/swiss/Modified LLE	frame	(shape->[10000,2])
/swiss/PCA	frame	(shape->[10000,3])
/swiss/Poly PCA	frame	(shape->[10000,2])
/swiss/RBF PCA	frame	(shape->[10000,2])
/swiss/Sigmoid PCA	frame	(shape->[10000,2])
/swiss/SpectralEmbedding	frame	(shape->[10000,2])
/swiss/label	series	(shape->[10000])
/swiss/lle/modified/10	frame	(shape->[10000,2])
/swiss/lle/modified/20	frame	(shape->[10000,2])
/swiss/lle/modified/25	frame	(shape->[10000,2])

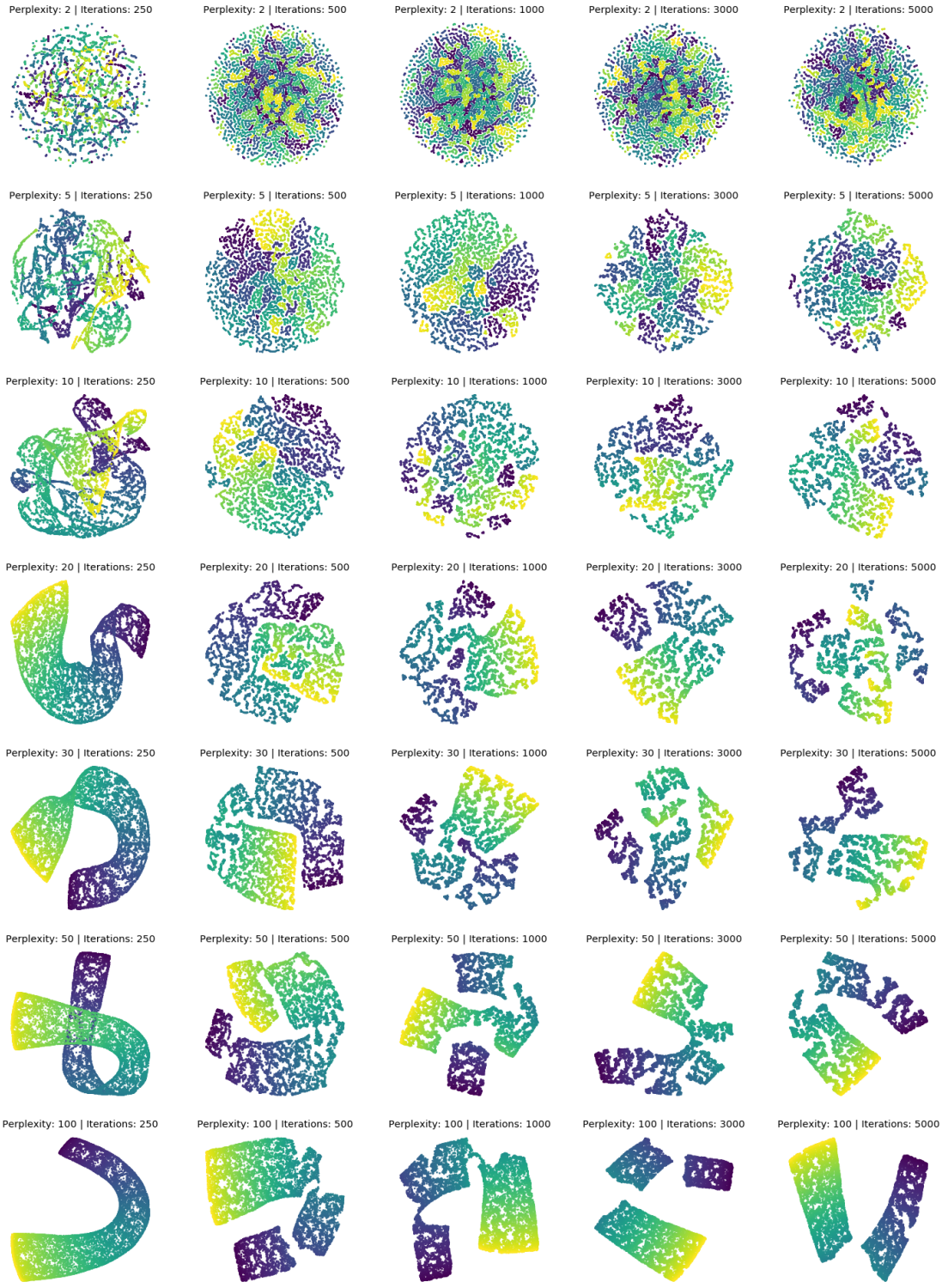
```
/swiss/lle/modified/30                  frame       (shape->[10000,2])
/swiss/lle/modified/40                  frame       (shape->[10000,2])
/swiss/lle/modified/5                   frame       (shape->[10000,2])
/swiss/lle/modified/50                  frame       (shape->[10000,2])
/swiss/lle/modified/stats               frame       (shape->[2,6])
/swiss/lle/standard/10                  frame       (shape->[10000,2])
/swiss/lle/standard/20                  frame       (shape->[10000,2])
/swiss/lle/standard/25                  frame       (shape->[10000,2])
/swiss/lle/standard/30                  frame       (shape->[10000,2])
/swiss/lle/standard/40                  frame       (shape->[10000,2])
/swiss/lle/standard/5                   frame       (shape->[10000,2])
/swiss/lle/standard/50                  frame       (shape->[10000,2])
/swiss/lle/standard/stats               frame       (shape->[2,12])
/swiss/tsne/10/1000                     frame       (shape->[10000,2])
/swiss/tsne/10/2000                     frame       (shape->[10000,2])
/swiss/tsne/10/250                      frame       (shape->[10000,2])
/swiss/tsne/10/3000                     frame       (shape->[10000,2])
/swiss/tsne/10/4000                     frame       (shape->[10000,2])
/swiss/tsne/10/500                      frame       (shape->[10000,2])
/swiss/tsne/10/5000                     frame       (shape->[10000,2])
/swiss/tsne/100/1000                    frame       (shape->[10000,2])
/swiss/tsne/100/2000                    frame       (shape->[10000,2])
/swiss/tsne/100/250                     frame       (shape->[10000,2])
/swiss/tsne/100/3000                    frame       (shape->[10000,2])
/swiss/tsne/100/4000                    frame       (shape->[10000,2])
/swiss/tsne/100/500                     frame       (shape->[10000,2])
/swiss/tsne/100/5000                    frame       (shape->[10000,2])
/swiss/tsne/2/1000                      frame       (shape->[10000,2])
/swiss/tsne/2/2000                      frame       (shape->[10000,2])
/swiss/tsne/2/250                       frame       (shape->[10000,2])
/swiss/tsne/2/3000                      frame       (shape->[10000,2])
/swiss/tsne/2/4000                      frame       (shape->[10000,2])
/swiss/tsne/2/500                       frame       (shape->[10000,2])
/swiss/tsne/2/5000                      frame       (shape->[10000,2])
/swiss/tsne/20/1000                     frame       (shape->[10000,2])
/swiss/tsne/20/2000                     frame       (shape->[10000,2])
/swiss/tsne/20/250                      frame       (shape->[10000,2])
/swiss/tsne/20/3000                     frame       (shape->[10000,2])
/swiss/tsne/20/4000                     frame       (shape->[10000,2])
/swiss/tsne/20/500                      frame       (shape->[10000,2])
/swiss/tsne/20/5000                     frame       (shape->[10000,2])
/swiss/tsne/30/1000                     frame       (shape->[10000,2])
/swiss/tsne/30/2000                     frame       (shape->[10000,2])
/swiss/tsne/30/250                      frame       (shape->[10000,2])
/swiss/tsne/30/3000                     frame       (shape->[10000,2])
/swiss/tsne/30/4000                     frame       (shape->[10000,2])
/swiss/tsne/30/500                      frame       (shape->[10000,2])
/swiss/tsne/30/5000                     frame       (shape->[10000,2])
```

```
/swiss/tsne/5/1000                      frame        (shape->[10000,2])
/swiss/tsne/5/2000                      frame        (shape->[10000,2])
/swiss/tsne/5/250                       frame        (shape->[10000,2])
/swiss/tsne/5/3000                      frame        (shape->[10000,2])
/swiss/tsne/5/4000                      frame        (shape->[10000,2])
/swiss/tsne/5/500                       frame        (shape->[10000,2])
/swiss/tsne/5/5000                      frame        (shape->[10000,2])
/swiss/tsne/50/1000                     frame        (shape->[10000,2])
/swiss/tsne/50/2000                     frame        (shape->[10000,2])
/swiss/tsne/50/250                      frame        (shape->[10000,2])
/swiss/tsne/50/3000                     frame        (shape->[10000,2])
/swiss/tsne/50/4000                     frame        (shape->[10000,2])
/swiss/tsne/50/500                      frame        (shape->[10000,2])
/swiss/tsne/50/5000                     frame        (shape->[10000,2])
/swiss/tsne/runtime                     series       (shape->[49])
/swiss/umap/10/1                        frame        (shape->[10000,2])
/swiss/umap/10/10                       frame        (shape->[10000,2])
/swiss/umap/10/20                       frame        (shape->[10000,2])
/swiss/umap/10/50                       frame        (shape->[10000,2])
/swiss/umap/2/1                         frame        (shape->[10000,2])
/swiss/umap/2/10                        frame        (shape->[10000,2])
/swiss/umap/2/20                        frame        (shape->[10000,2])
/swiss/umap/2/50                        frame        (shape->[10000,2])
/swiss/umap/25/1                        frame        (shape->[10000,2])
/swiss/umap/25/10                       frame        (shape->[10000,2])
/swiss/umap/25/20                       frame        (shape->[10000,2])
/swiss/umap/25/50                       frame        (shape->[10000,2])
/swiss/umap/5/1                         frame        (shape->[10000,2])
/swiss/umap/5/10                        frame        (shape->[10000,2])
/swiss/umap/5/20                        frame        (shape->[10000,2])
/swiss/umap/5/50                        frame        (shape->[10000,2])
/swiss/umap/50/1                        frame        (shape->[10000,2])
/swiss/umap/50/10                       frame        (shape->[10000,2])
/swiss/umap/50/20                       frame        (shape->[10000,2])
/swiss/umap/50/50                       frame        (shape->[10000,2])
/swiss/umap/runtime                     series       (shape->[20])
```

[14]:
```
fig, axes = plt.subplots(nrows=7, ncols=5, figsize=(20, 28))
method = 'tsne'
with pd.HDFStore(join('data', 'manifolds.h5')) as store:
    labels = store['/'.join(['swiss', 'label'])]
    for row, perplexity in enumerate([2, 5, 10, 20 , 30, 50, 100]):
        for col, n_iter in enumerate([250, 500, 1000, 3000, 5000]):
            x, y = store.get('/'.join(['swiss',  method, str(perplexity),␣
 ↪str(n_iter)])).T.values
            axes[row, col].scatter(x, y, c=palette, s=5)
```

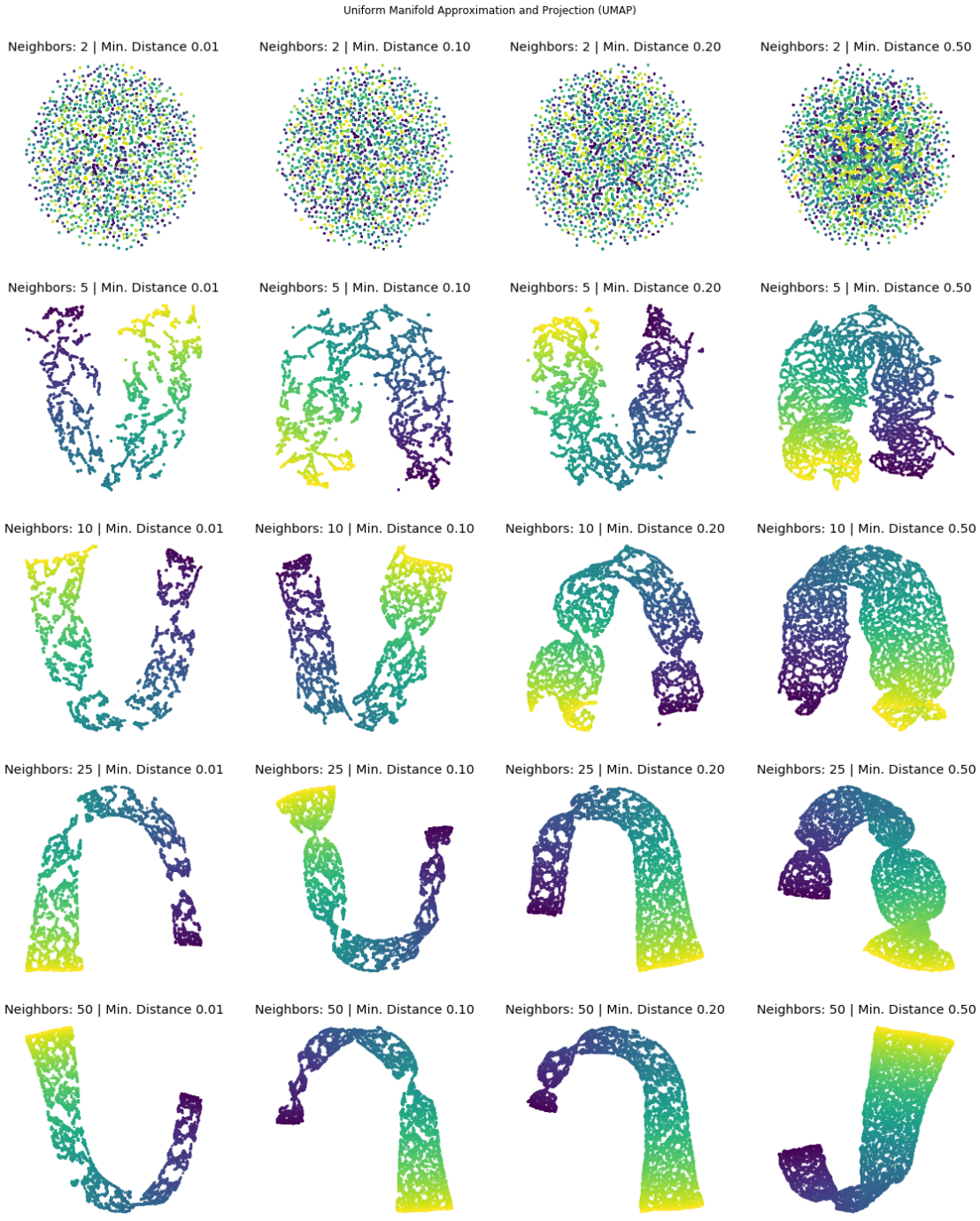
```
            axes[row, col].set_title('Perplexity: {} | Iterations: {}'.
 ↪format(perplexity, n_iter))
            axes[row, col].axis('off')
fig.tight_layout()
fig.suptitle('T-Stochastic Neighbor Embedding (TSNE)')
fig.subplots_adjust(top=.94)
```

T-Stochastic Neighbor Embedding (TSNE)



1.3.2 UMAP

[15]:
```
fig, axes = plt.subplots(nrows=5, ncols=4, figsize=(16, 20))
method = 'umap'
with pd.HDFStore(join('data', 'manifolds.h5')) as store:
    labels = store['swiss/label']
    for row, n_neighbors in enumerate([2, 5, 10, 25, 50]):
        for col, min_distance in enumerate([1, 10, 20, 50]):
            x, y = store.get('swiss/' + method + '/' + str(n_neighbors) + '/' +␣
 ↪str(min_distance)).T.values
            axes[row, col].scatter(x, y, c=palette, s=5)
            axes[row, col].set_title('Neighbors: {} | Min. Distance {:.2f}'.
 ↪format(n_neighbors, min_distance/100))
            axes[row, col].axis('off')
fig.tight_layout()
fig.suptitle('Uniform Manifold Approximation and Projection (UMAP)')
fig.subplots_adjust(top=.94)
```


Uniform Manifold Approximation and Projection (UMAP)
Neighbors: 2 | Min. Distance 0.01
Neighbors: 2 | Min. Distance 0.10
Neighbors: 2 | Min. Distance 0.20
Neighbors: 2 | Min. Distance 0.50
Neighbors: 5 | Min. Distance 0.01
Neighbors: 5 | Min. Distance 0.10
Neighbors: 5 | Min. Distance 0.20
Neighbors: 5 | Min. Distance 0.50
Neighbors: 10 | Min. Distance 0.01
Neighbors: 10 | Min. Distance 0.10
Neighbors: 10 | Min. Distance 0.20
Neighbors: 10 | Min. Distance 0.50
Neighbors: 25 | Min. Distance 0.01
Neighbors: 25 | Min. Distance 0.10
Neighbors: 25 | Min. Distance 0.20
Neighbors: 25 | Min. Distance 0.50
Neighbors: 50 | Min. Distance 0.01
Neighbors: 50 | Min. Distance 0.10
Neighbors: 50 | Min. Distance 0.20
Neighbors: 50 | Min. Distance 0.50

1.4 Handwritten Digits

[22]:
```
mnist = fetch_openml('mnist_784', data_home='.')
classes = sorted(np.unique(mnist.target).astype(int))
mnist.data.shape
```

[22]:
```
(70000, 784)
```

[23]:
```
ipv_cmap = sns.color_palette("Paired", n_colors=10)
print(classes)
sns.palplot(ipv_cmap)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1.4.1 Plot sample images

[24]:
```
image_size = int(np.sqrt(mnist.data.shape[1])) # 28 x 28 pixels
n_samples = 15
```

[35]:
```
fig, ax = plt.subplots()
mnist_sample = np.empty(
    shape=(image_size * len(classes), image_size * n_samples))
for row, label in enumerate(classes):
    label_data = np.squeeze(np.argwhere(mnist.target.astype(int) == label))
    samples = choice(label_data, size=n_samples, replace=False)
    i = row * image_size
    for col, sample in enumerate(samples):
        j = col * image_size
        mnist_sample[i:i+image_size, j:j +
                     image_size] = mnist.data[sample].reshape(image_size, -1)

ax.imshow(mnist_sample, cmap='Blues')
plt.title('Handwritten Digits')
plt.axis('off')
plt.tight_layout()
```

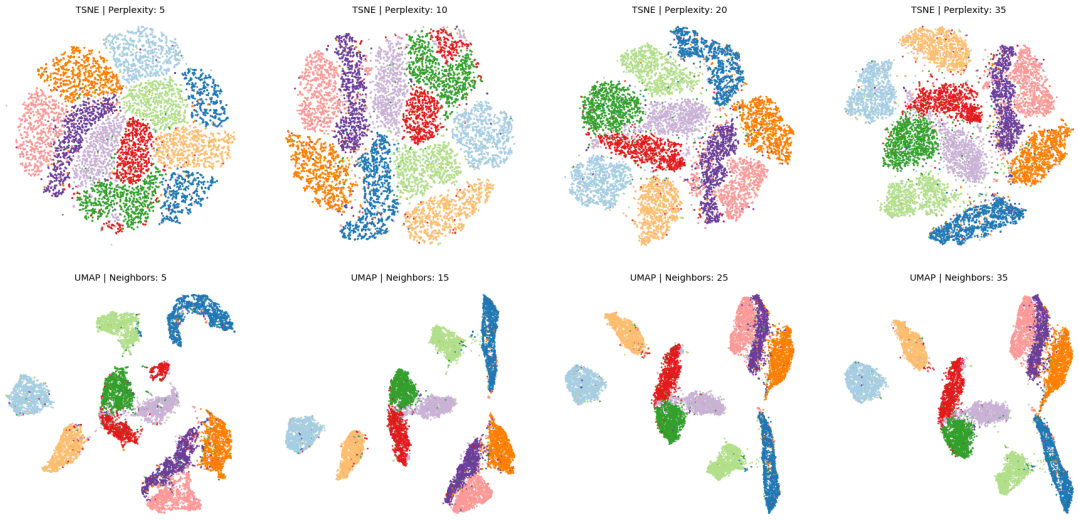
[36]:
```
plotly_cmap = cl.to_rgb( cl.scales['10']['qual']['Paired'])
def plotly_scatter(data, label, title, color, x='x', y='y'):
    fig = dict(
        data=[
            dict(
                type='scattergl',
                x=data[:, 0],
                y=data[:, 1],
                legendgroup="group",
                text=label.astype(int),
                mode='markers',
                marker=Marker(
                    size=5,
                    color=color,
                    autocolorscale=False,
                    showscale=False,
                    opacity=.9,
                    colorbar=ColorBar(
                        title='Class'
                    ),
                    line=dict(width=1))),
        ],
        layout=dict(title=title,
```

```
                    width=1200,
                    font=dict(color='white'),
                    xaxis=dict(
                        title=x,
                        hoverformat='.1f',
                        showgrid=False),
                    yaxis=dict(title=y,
                               hoverformat='.1f',
                               showgrid=False),
                    paper_bgcolor='rgba(0,0,0,0)',
                    plot_bgcolor='rgba(0,0,0,0)'
                    ))

    iplot(fig, show_link=False)
```

1.4.2 t-SNE and UMAP Visualization

[37]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(25, 12))
methods = ['tsne', 'umap']
params = {}
params['tsne'] = [5, 10, 20, 35]
params['umap'] = [5, 15, 25, 35]
param_labels = {'tsne': 'Perplexity', 'umap': 'Neighbors'}
with pd.HDFStore(join('data', 'manifolds.h5')) as store:
    labels = store['mnist/labels']
    color = [sns.color_palette('Paired', 10)[int(i)] for i in labels]
    for row, method in enumerate(methods):
        for col, param in enumerate(params[method]):
            x, y = store.get('mnist/' + method + '/2/' + str(param)).T.values
            axes[row, col].scatter(x, y, c=color, s=5)
            axes[row, col].set_title('{} | {}: {}'.format(method.upper(),␣
 ↪param_labels[method], param))
            axes[row, col].axis('off')
fig.tight_layout();
```



[38]:
```
def get_result(source, method, params):
    key = '/'.join([source, method, '/'.join([str(p) for p in params])])
    with pd.HDFStore(join('data', 'manifolds.h5')) as store:
        data = store[key].values
        labels = store['/'.join([source, 'labels'])]
    return data, labels
```

1.5 Load Fashion MNIST Data

[16]:
```
fashion_mnist = pd.read_csv(Path('data', 'fashion-mnist_train.csv.gz'))
fashion_label = fashion_mnist.label
fashion_data = fashion_mnist.drop('label', axis=1).values
classes = sorted(np.unique(fashion_label).astype(int))
```

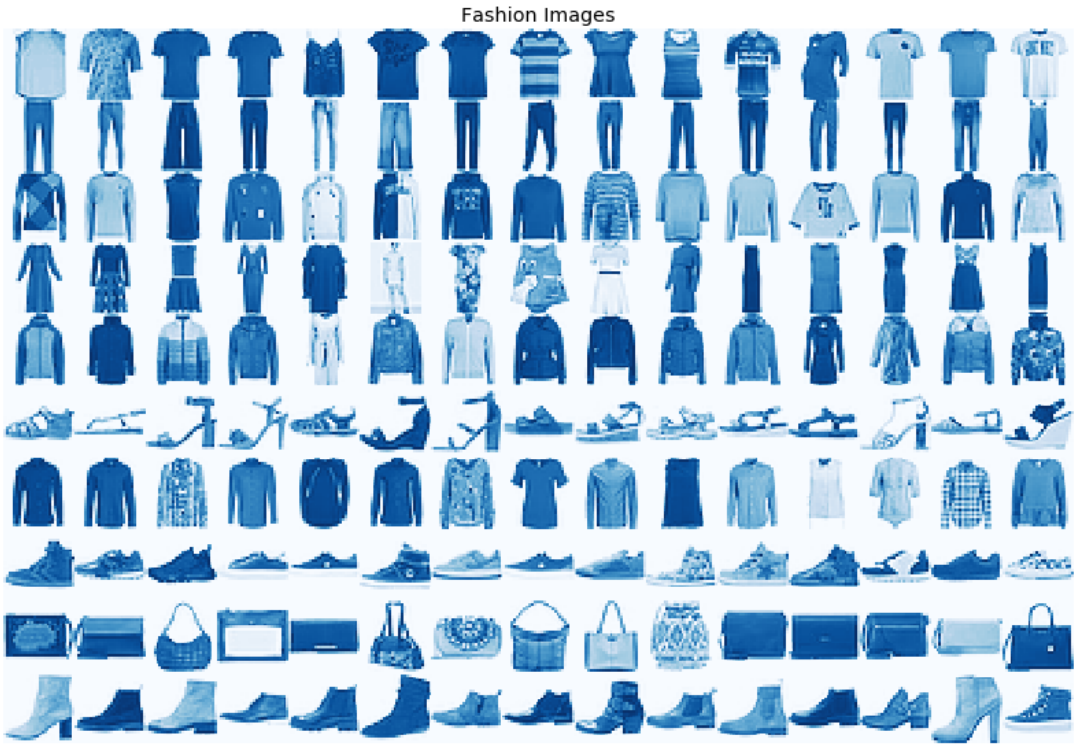
[17]:
```
image_size = int(np.sqrt(fashion_data.shape[1])) # 28 x 28 pixels
n_samples = 15
```

1.5.1 Plot sample images

[18]:
```
fig, ax = plt.subplots(figsize=(14,8))
fashion_sample = np.empty(shape=(image_size * len(classes),
                                 image_size * n_samples))
for row, label in enumerate(classes):
    label_data = np.squeeze(np.argwhere(fashion_label == label))
    samples = choice(label_data, size=n_samples, replace=False)
    i = row * image_size
    for col, sample in enumerate(samples):
        j = col * image_size
        fashion_sample[i:i+image_size,
```

```
                    j:j + image_size] = fashion_data[sample].
↪reshape(image_size, -1)

ax.imshow(fashion_sample, cmap='Blues')
plt.title('Fashion Images')
plt.axis('off')
plt.tight_layout();
```



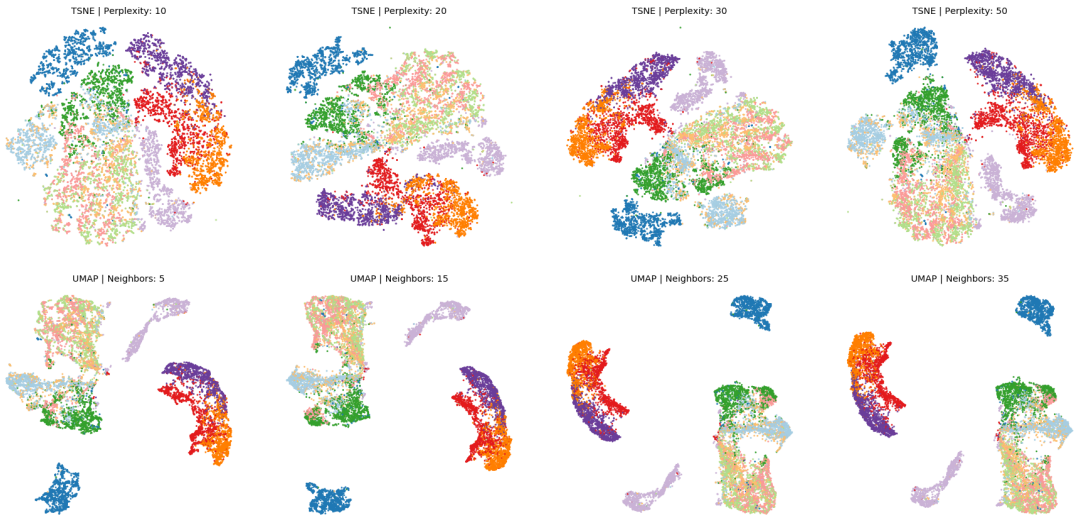
1.5.2 t-SNE and UMAP: Parameter Settings

The upper panels of the following chart show how t-SNE is able to differentiate between the image classes. A higher perplexity value increases the number of neighbors used to compute local structure and gradually results in more emphasis on global relationships.

The below figure illustrates how UMAP does indeed move the different clusters further apart, whereas t-SNE provides more granular insight into the local structure.

[19]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(25, 12))
source = 'fashion'
methods = ['tsne', 'umap']
params = {}
params['tsne'] = [10, 20, 30, 50]
params['umap'] = [5, 15, 25, 35]
```

```
param_labels = {'tsne': 'Perplexity', 'umap': 'Neighbors'}
with pd.HDFStore(join('data', 'manifolds.h5')) as store:
    labels = store[source + '/labels']
    color = [sns.color_palette('Paired', 10)[int(i)] for i in labels]
    for row, method in enumerate(methods):
        for col, param in enumerate(params[method]):
            x, y = store.get(source + '/' + method + '/2/' + str(param)).T.
 ↪values
            axes[row, col].scatter(x, y, c=color, s=5)
            axes[row, col].set_title('{} | {}: {}'.format(method.upper(),␣
 ↪param_labels[method], param))
            axes[row, col].axis('off')
fig.tight_layout();
```



[21]:
```
plotly_cmap = cl.to_rgb( cl.scales['10']['qual']['Paired'])
def plotly_scatter(data, label, title, color, x='x', y='y'):
    fig = dict(
        data=[
            dict(
                type='scattergl',
                x=data[:, 0],
                y=data[:, 1],
                legendgroup="group",
                text=label.astype(int),
                mode='markers',
                marker=dict(
                    size=5,
                    color=color,
                    autocolorscale=True,
```

```
                    showscale=False,
                    opacity=.9,
                    colorbar=go.ColorBar(
                        title='Class'
                    ),
                    line=dict(width=1))),
        ],
        layout=dict(title=title,
                    width=1200,
                    font=dict(color='white'),
                    xaxis=dict(
                        title=x,
                        hoverformat='.1f',
                        showgrid=False),
                    yaxis=dict(title=y,
                               hoverformat='.1f',
                               showgrid=False),
                    paper_bgcolor='rgba(0,0,0,0)',
                    plot_bgcolor='rgba(0,0,0,0)'
                    ))

    iplot(fig, show_link=False)
```

1.5.3 Plotly: t-SNE

[22]:
```
data, labels = get_result('fashion', 'tsne', [2, 25])
plotly_color = [plotly_cmap[int(i)] for i in labels]

plotly_scatter(data=data,
               title='MNIST TSNE Projection',
               label=labels,
               color=plotly_color)
```

1.5.4 Plotly UMAP

[23]:
```
data, labels = get_result('fashion', 'umap', [2, 15])
plotly_color = [plotly_cmap[int(i)] for i in labels]
plotly_scatter(data=data,
               title='MNIST UMAP Projection',
               label=labels,
               color=plotly_color)
```

1.5.5 t-SNE in 3D

[24]:
```
data, labels = get_result('fashion', 'tsne', [3, 25])
ipv_color = [ipv_cmap[int(t)] for t in labels]
ipv.quickscatter(*data.T, size=.5, color=ipv_color, marker='sphere')
```

```
VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0, position=(0.0, 0.0, 2.
↪0), quaternion=(0.0, 0.0, 0.0, …
```