02_manifold_learning_lle

September 29, 2021

1 Manifold Learning with Local Linear Embedding

Several techniques approximate a lower dimensional manifold. One example is locally-linear embedding (LLE) that was developed in 2000 by Sam Roweis and Lawrence Saul.

This notebook demonstrates how LLE 'unrolls' the swiss roll, and how it performs on other data.

For each data point, LLE identifies a given number of nearest neighbors and computes weights that represent each point as a linear combination of its neighbors. It finds a lower-dimensional embedding by linearly projecting each neighborhood on global internal coordinates on the lower-dimensional manifold and can be thought of as a sequence of PCA applications.

1.1 Imports & Settings

[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
from matplotlib import cm
import ipyvolume as ipv
import ipyvolume.pylab as p3
from ipywidgets import HBox
from sklearn.datasets import fetch_mldata, make_swiss_roll, make_s_curve
from sklearn.manifold import locally_linear_embedding
from sklearn.decomposition import PCA
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
import colorlover as cl
```

[2]:
```
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)
```

1.2 Manifold Examples

1.2.1 Linear Manifold: Ellipse in 3D

[3]:
```
n_points, noise = 1000, 0.1
angles = uniform(low=-np.pi, high=np.pi, size=n_points)
x = 2 * np.cos(angles) + noise * randn(n_points)
y = np.sin(angles) + noise * randn(n_points)

theta = np.pi/4 # 45 degree rotation
rotation_matrix = np.array([[np.cos(theta), -np.sin(theta)],
                            [np.sin(theta), np.cos(theta)]])

rotated = np.column_stack((x, y)).dot(rotation_matrix)
x, y = rotated[:, 0], rotated[:, 1]

z = .2 * x  + .2 * y + noise * randn(n_points)
ellipse3d = np.vstack((x, y, z)).T
```

1.2.2 PCA: Linear Dimensionality Reduction

[4]:
```
def get_2d_projection(data, pc):
    min_, max_ = data[:, :2].min(0), data[:, :2].max(0)
    X, Y = np.meshgrid(np.linspace(min_[0], max_[0], 50),
                       np.linspace(min_[1], max_[1], 50))

    nv = np.cross(pc.components_[0], pc.components_[1])
    d = -pc.mean_.dot(nv)
    Z = (-nv[0] * X - nv[1] * Y - d) * 1 / nv[2]
    factor = max(data[:, -1].min() / Z.min(), data[:, -1].max()/Z.max())
    return X, Y, Z * factor
```

[5]:
```
pca = PCA(n_components=2)
ellipse2d = pca.fit_transform(ellipse3d)
```

[6]:
```
znorm = z - z.min()
znorm /= znorm.ptp()
color = cm.viridis(znorm)

xs, ys, zs = get_2d_projection(ellipse3d, pca)
p3.figure(width=600, height=600)
p3.plot_wireframe(xs, ys, zs, color="black")
p3.scatter(x, y, z, marker='sphere', color=color[:,0:3], size=1)
p3.view(azimuth=45, elevation=75)
p3.show()
```

```
VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0, position=(0.
↪3660254037844386, 1.9318516525781366, 0.3…
```

1.2.3 Swiss Roll Example

[7]:
```
n_samples = 10000
palette = sns.color_palette('viridis', n_colors=n_samples)
zeros = np.zeros(n_samples) + .5
```

[8]:
```
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
```

[9]:
```
p3.figure()
p3.scatter(np.sort(swiss_val), y, zeros, marker='sphere', color=palette, size=1)
p3.xlim(swiss_val.min(), swiss_val.max())
fig = p3.gcc()
```

[11]:
```
HBox([
    ipv.quickscatter(x, y, z, size=1, color=palette, marker='sphere'),
    fig
])
```

```
HBox(children=(VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0,␣
↪position=(0.0, 0.0, 2.0), quaternion=(…
```

Linear cuts along the axes

[12]:
```
p3.figure(width=600, height=600)
p3.scatter(zeros, y, z, marker='sphere', color=palette, size=1)
p3.view(azimuth=15, elevation=45)
fig1 = p3.gcc()
```

[13]:
```
p3.figure(width=600, height=600)
p3.scatter(x, zeros, z, marker='sphere', color=palette, size=1)
p3.view(azimuth=15, elevation=45)
fig2 = p3.gcc()
```

[14]:
```
p3.figure(width=600, height=600)
p3.scatter(x, y, zeros, marker='sphere', color=palette, size=1)
p3.view(azimuth=15, elevation=45)
fig3 = p3.gcc()
```

[15]:
```
HBox([
    fig1, fig2, fig3]
)
```

```
HBox(children=(VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0,␣
↪position=(0.36602540378443865, 1.41421…
```

Principal Component Analysis

[16]:
```
pca = PCA(n_components=2)
swiss_2d = pca.fit_transform(swiss_3d)
```

[17]:
```
p3.figure(width=600, height=600)
xs, ys, zs = get_2d_projection(swiss_3d, pca)
p3.plot_wireframe(xs, ys, zs, color='black')
p3.scatter(*swiss_3d.T, marker='sphere', color=palette, size=1)
p3.view(azimuth=15, elevation=45)
fig1 = p3.gcc()
```

[18]:
```
p3.figure(width=600, height=600)

min_2d, max_2d = swiss_2d[:, :2].min(0), swiss_2d[:, :2].max(0)
x2d, y2d = np.meshgrid(np.linspace(min_2d[0], max_2d[0], 100),
                     np.linspace(min_2d[1], max_2d[1], 100))
p3.plot_wireframe(x2d, y2d, np.zeros(shape=(100, 100)) + .5, color='black'),

p3.scatter(*np.c_[swiss_2d, np.zeros(n_samples) + .5].T,
           marker='sphere', color=palette, size=1)
p3.view(azimuth=45, elevation=45)
fig2 = p3.gcc()
```

[19]:
```
HBox([
fig1, fig2]
)
```

```
HBox(children=(VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0,␣
↪position=(0.36602540378443865, 1.41421…
```

1.2.4 But will manifold learning simplify the task at hand?

[20]:
```
cpos, cneg = cm.viridis(0)[:3], cm.viridis(.999)[:3]
positive_class = swiss_3d[:, 0] > 4
X_pos = swiss_3d[positive_class]
X_neg = swiss_3d[~positive_class]
```

[21]:
```
p3.figure(width=600, height=600)
p3.scatter(*X_pos.T, marker='sphere', color=cpos, size=1)
p3.scatter(*X_neg.T, marker='sphere', color=cneg, size=1)
p3.view(azimuth=15, elevation=45)
fig1 = p3.gcc()
```

[22]:
```
p3.figure(width=600, height=600)
p3.scatter(np.sort(swiss_val)[positive_class], X_pos[:, 1],␣
↪zeros,marker='sphere', color=cpos, size=1)
```

```
p3.scatter(np.sort(swiss_val)[~positive_class], X_neg[:, 1],␣
↪zeros,marker='sphere', color=cneg, size=1)
p3.view(azimuth=15, elevation=45)
fig2 = p3.gcc()
```

[23]:
```
HBox([fig1, fig2])
```

```
HBox(children=(VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0,␣
↪position=(0.36602540378443865, 1.41421…
```

[24]:
```
positive_class = 2 * (np.sort(swiss_val) - 4) > swiss_3d[:, 1]
X_pos = swiss_3d[positive_class]
X_neg = swiss_3d[~positive_class]
```

[25]:
```
p3.figure(width=600, height=600)
p3.scatter(*X_pos.T, marker='sphere', color=cpos, size=1)
p3.scatter(*X_neg.T, marker='sphere', color=cneg, size=1)
p3.view(azimuth=15, elevation=45)
fig1 = p3.gcc()
```

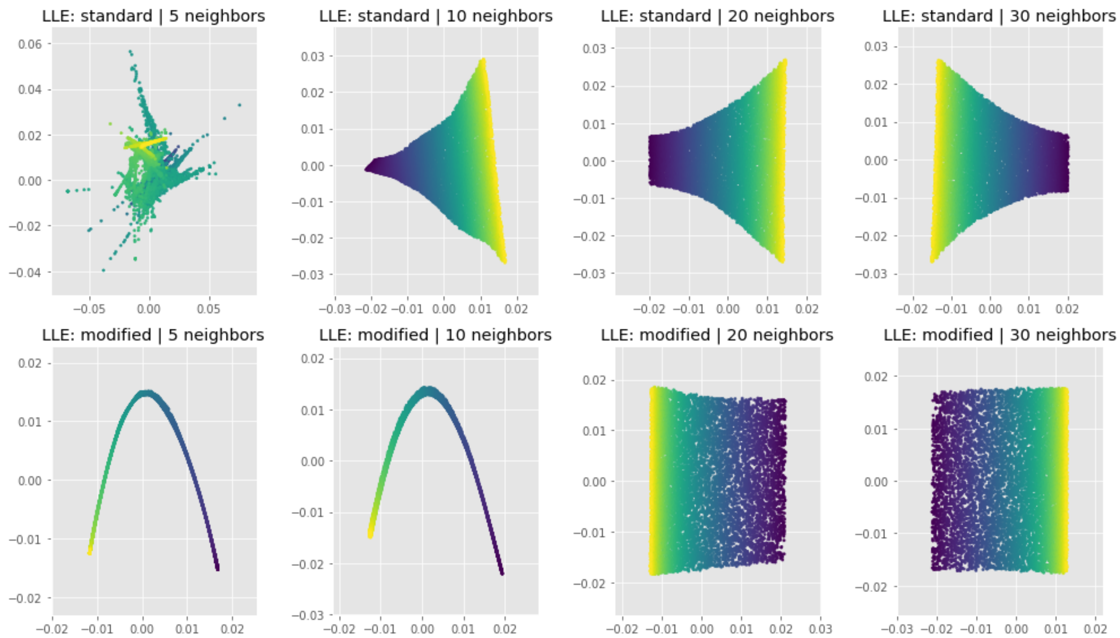
[26]:
```
p3.figure(width=600, height=600)
p3.scatter(np.sort(swiss_val)[positive_class], X_pos[:, 1],␣
↪zeros,marker='sphere', color=cpos, size=1)
p3.scatter(np.sort(swiss_val)[~positive_class], X_neg[:, 1],␣
↪zeros,marker='sphere', color=cneg, size=1)
p3.view(azimuth=15, elevation=45)
fig2 = p3.gcc()
```

[27]:
```
HBox([fig1, fig2])
```

```
HBox(children=(VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0,␣
↪position=(0.36602540378443865, 1.41421…
```

1.3 Local-Linear Embedding

[28]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(14, 8))
for row, method in enumerate(['standard', 'modified']):
    for col, n_neighbors in enumerate([5, 10, 20, 30]):
        embedded, err = locally_linear_embedding(swiss_3d,␣
↪n_neighbors=n_neighbors, n_components=2,
                                                 method=method, random_state=42)
        axes[row, col].scatter(*embedded.T, c=palette, s=5)
        axes[row, col].set_title('LLE: {} | {} neighbors'.format(method,␣
↪n_neighbors))
fig.tight_layout()
```



1.3.1 S-Curve Example

[29]:
```
scurve_3d, scurve_val = make_s_curve(
    n_samples=n_samples, noise=.05, random_state=42)
scurve_3d = scurve_3d[scurve_val.argsort()[::-1]]
scurve_3d[:, 1] *= 10
```

[30]:
```
pca = PCA(n_components=2)
scurve_2d = pca.fit_transform(scurve_3d)
```

[31]:
```
p3.figure(width=600, height=600)
xs, ys, zs = get_2d_projection(scurve_3d, pca)
p3.plot_wireframe(xs, ys, zs, color='black')
p3.scatter(*scurve_3d.T, marker='sphere', color=palette, size=1)
p3.view(azimuth=15, elevation=45)
fig1 = p3.gcc()
```

[32]:
```
p3.figure(width=600, height=600)

min_2d, max_2d = scurve_2d[:, :2].min(0), scurve_2d[:, :2].max(0)
x2d, y2d = np.meshgrid(np.linspace(min_2d[0], max_2d[0], 100),
                       np.linspace(min_2d[1], max_2d[1], 100))
p3.plot_wireframe(x2d, y2d, np.zeros(shape=(100, 100)) + .5, color='black'),

p3.scatter(*np.c_[scurve_2d, np.zeros(n_samples) + .5].T,
           marker='sphere', color=palette, size=1)
```

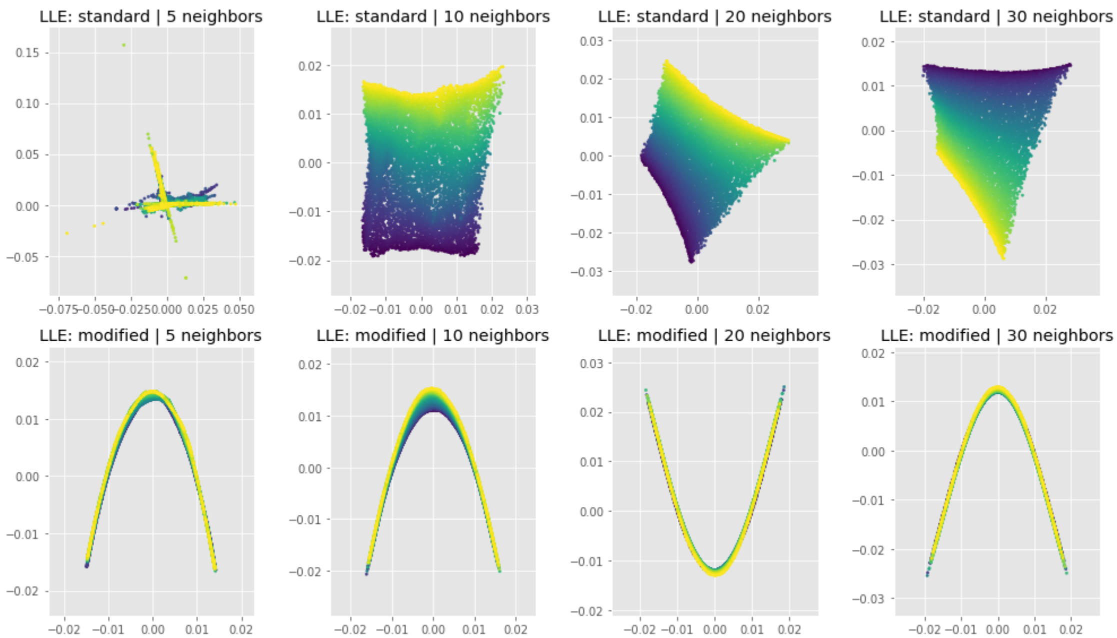
```
p3.view(azimuth=45, elevation=45)
fig2 = p3.gcc()
```

[33]:
```
HBox([
fig1, fig2]
)
```

```
HBox(children=(VBox(children=(Figure(camera=PerspectiveCamera(fov=46.0,␣
↪position=(0.36602540378443865, 1.41421…
```

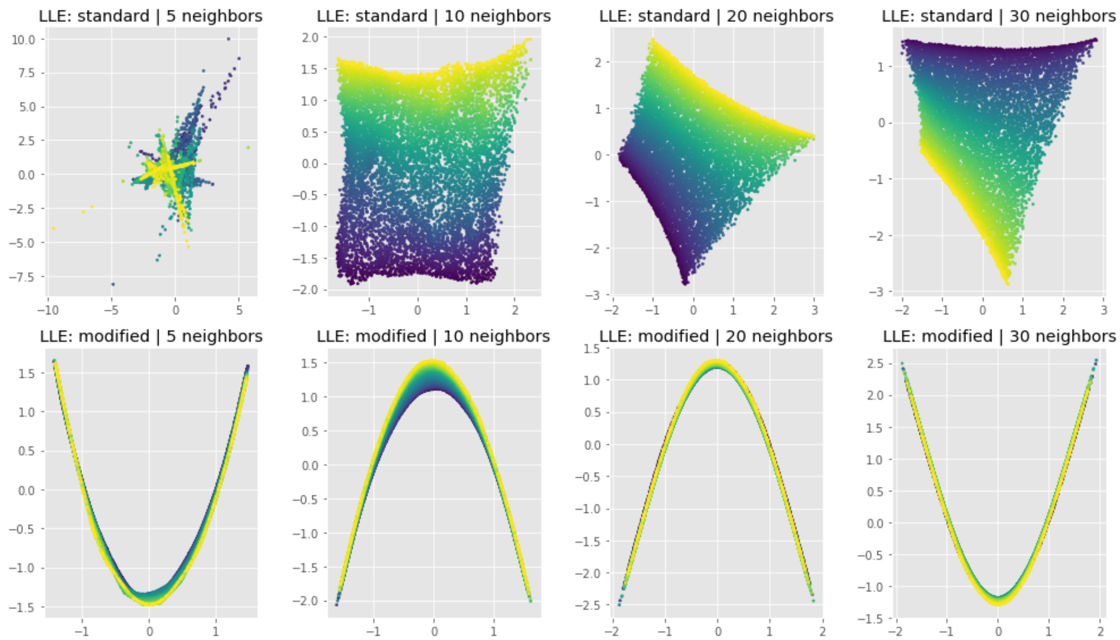
Local-Linear Embedding

[34]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(14, 8))
for row, method in enumerate(['standard', 'modified']):
    for col, n_neighbors in enumerate([5, 10, 20, 30]):
        embedded, err = locally_linear_embedding(scurve_3d,
                                                  n_neighbors=n_neighbors,
                                                  n_components=2,
                                                  method=method,
                                                  random_state=42)
        axes[row, col].scatter(*embedded.T, c=palette, s=5)
        axes[row, col].set_title('LLE: {} | {} neighbors'.format(method,␣
↪n_neighbors))
fig.tight_layout()
```



[36]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(14, 8))
with pd.HDFStore('/'.join(['data', 'manifolds.h5'])) as store:
```

```
    for row, method in enumerate(['standard', 'modified']):
        for col, n_neighbors in enumerate([5, 10, 20, 30]):
            x, y = store.get('/'.join(['scurve', 'lle', method,␣
↪str(n_neighbors)])).T.values * 100
            axes[row, col].scatter(x, y, c=palette, s=5)
            axes[row, col].set_title('LLE: {} | {} neighbors'.format(method,␣
↪n_neighbors))
fig.tight_layout()
```



[54]:
```
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.scattergl.marker.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)
```

Local Linear Embedding: Standard The following `locally_linear_embedding` on `mnist.data` takes fairly long to run, hence we are providing pre-computed results so you can explore the visualizations regardless of your hardware setup.

[36]:
```
# the pre-computed manifold results for the various datasets and numerous␣
 ↪parameter settings are here:
with pd.HDFStore(join('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])
```

[40]:

```
# commented out to avoid long run time
# lle, err = locally_linear_embedding(X=mnist.data, n_components=2,␣
 ↪n_neighbors=20, method='standard')
```

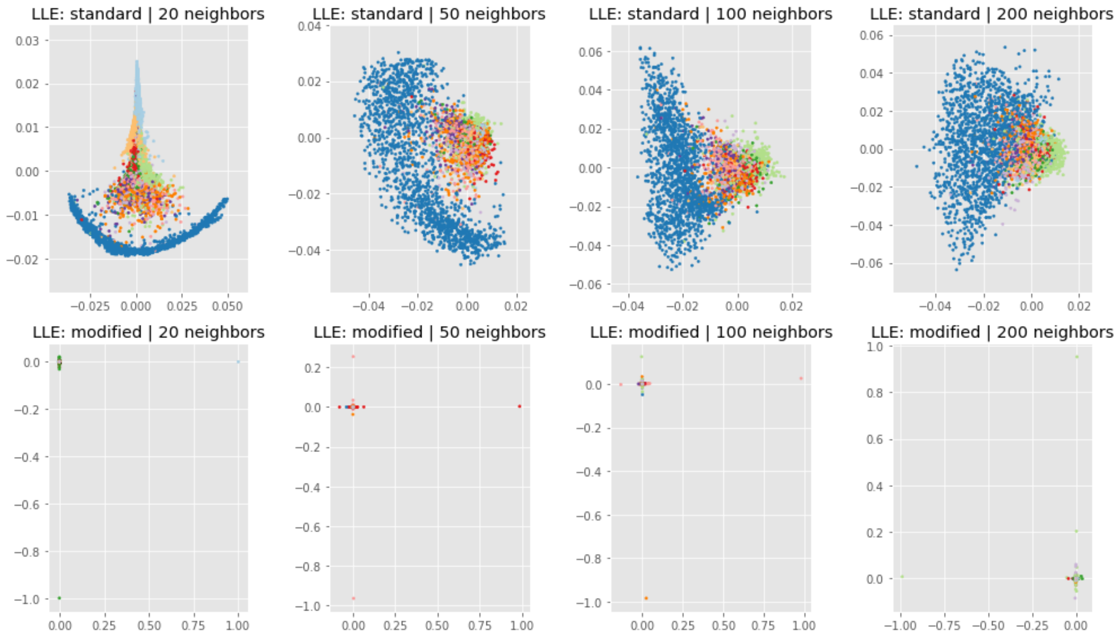
[37]:

```
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
```

[38]:

```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(14, 8))
with pd.HDFStore(join('data', 'manifolds.h5')) as store:
    labels = store.get('/'.join(['mnist', 'labels']))
    color = [sns.color_palette('Paired', 10)[int(i)] for i in labels]
    for row, method in enumerate(['standard', 'modified']):
        for col, n_neighbors in enumerate([20, 50, 100, 200]):
            try:
                x, y = store.get('/'.join(['mnist', 'lle', method, '2',␣
 ↪str(n_neighbors)])).T.values
            except:
```

```
                x, y = store.get('/'.join(['mnist', 'lle', '2',␣
↪str(n_neighbors)])).T.values
            axes[row, col].scatter(x, y, c=color, s=5)
            axes[row, col].set_title('LLE: {} | {} neighbors'.format(method,␣
↪n_neighbors))
fig.tight_layout()
```



[55]:
```
params = ['standard', 2, 100]
embedding, labels = get_result('mnist', 'lle', params)
color = [plotly_cmap[int(i)] for i in labels]
plotly_scatter(embedding, labels, color=color, title='Local Linear Embedding␣
↪(Standard) | 100 Neighbors')
```

1.3.2 Load Fashion MNIST Data

[47]:
```
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))
```

[48]:
```
image_size = int(np.sqrt(fashion_data.shape[1])) # 28 x 28 pixels
n_samples = 15
```

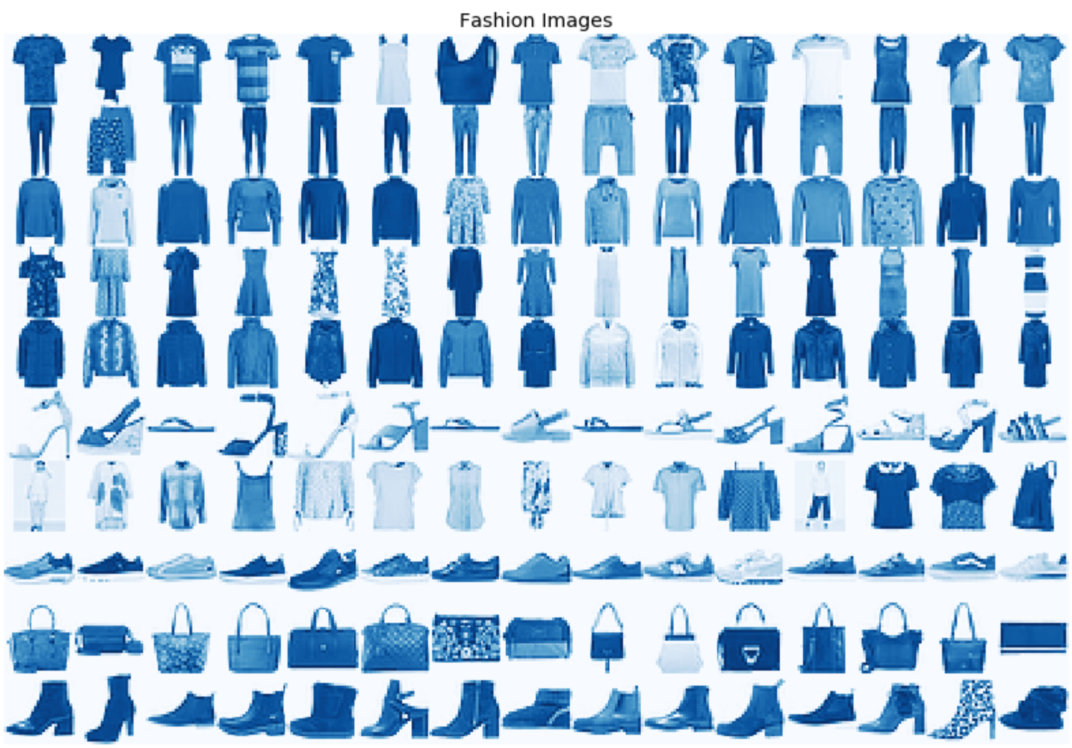
[49]:
```
fig, ax = plt.subplots(figsize=(14, 8))
fashion_sample = np.empty(shape=(image_size * len(classes),
                                 image_size * n_samples))
```

```
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();
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-
packages/numpy/core/fromnumeric.py:56: FutureWarning:

Series.nonzero() is deprecated and will be removed in a future version.Use
Series.to_numpy().nonzero() instead
```



[50]:
```
pca = PCA(n_components=2)
fashion_pca_2d = pca.fit_transform(fashion_data)
ev = pca.explained_variance_ratio_
pd.Series(ev)
```

[50]:
```
0    0.29
1    0.18
dtype: float64
```

[56]:
```
plotly_color = [plotly_cmap[int(i)] for i in fashion_label]
plotly_scatter(data=fashion_pca_2d,
               title='Fashion MNIST PCA Projection',
               label=fashion_label,
               color=plotly_color,
               x='1st Principal Component: {:.2%}'.format(ev[0]),
               y='Second Principal Component: {:.2%}'.format(ev[1]))
```

[57]:
```
pca = PCA(n_components=3)
fashion_3d = pca.fit_transform(fashion_data)
pd.Series(pca.explained_variance_ratio_)
```

[57]:
```
0    0.29
1    0.18
2    0.06
dtype: float64
```

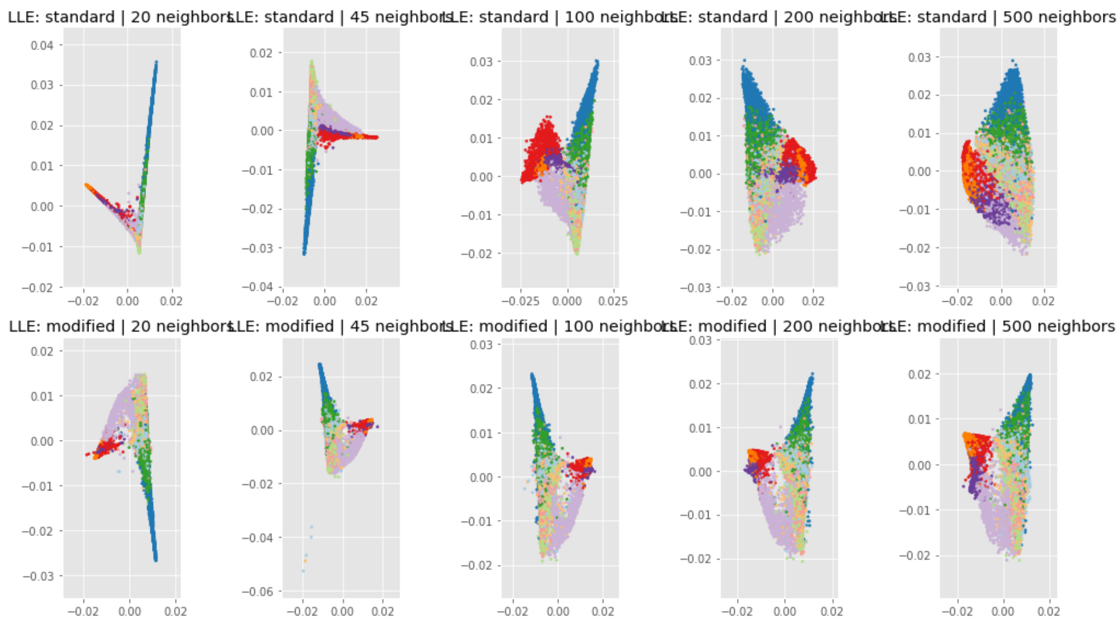
[58]:
```
ipv_color = [ipv_cmap[int(t)] for t in fashion_label]
ipv.quickscatter(*fashion_3d.T, size=.5, color=ipv_color, marker='sphere')
```

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

1.3.3 Local Linear Embedding

[59]:
```
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(14,8))
with pd.HDFStore('/'.join(['data', 'manifolds.h5'])) as store:
    labels = store.get('/'.join(['fashion', 'labels']))
    color = [sns.color_palette('Paired', 10)[int(i)] for i in labels]
    for row, method in enumerate(['standard', 'modified']):
        for col, n_neighbors in enumerate([20, 45, 100, 200, 500]):
            try:
                x, y = store.get('/'.join(['fashion', 'lle', method, '2',␣
↪str(n_neighbors)])).T.values
            except:
                x, y = store.get('/'.join(['fashion', 'lle', '2',␣
↪str(n_neighbors)])).T.values
            axes[row, col].scatter(x, y, c=color, s=5)
```

```
            axes[row, col].set_title('LLE: {} | {} neighbors'.format(method,␣
 ↪n_neighbors))
fig.tight_layout()
```



[60]:
```
params = ['standard', 2, 200]
embedding, labels = get_result('fashion', 'lle', params)
color = [plotly_cmap[int(i)] for i in labels]
plotly_scatter(embedding, labels, color=color, title='Local Linear Embedding␣
 ↪(Standard) | 200 Neighbors' )
```