gplvm-pyro experiments

January 2, 2024

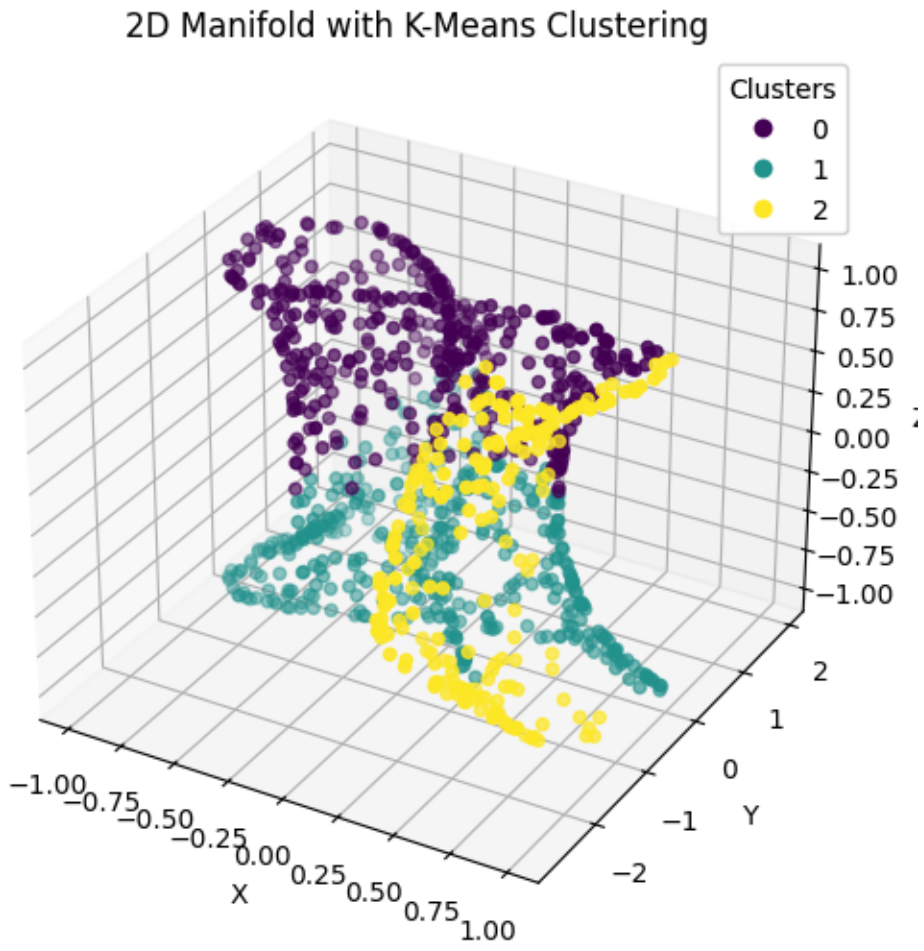
[24]:
```
import os
import matplotlib.pyplot as plt
import pandas as pd
import torch
from torch.nn import Parameter

%pip install pyro-ppl
import pyro
import pyro.contrib.gp as gp
import pyro.distributions as dist
import pyro.ops.stats as stats

smoke_test = ('CI' in os.environ)  # ignore; used to check code integrity in the␣
 ↪Pyro repo
assert pyro.__version__.startswith('1.8.6')
pyro.set_rng_seed(1)
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pyro-ppl in
./.local/perlmutter/pytorch1.13.1/lib/python3.9/site-packages (1.8.6)
Requirement already satisfied: pyro-api>=0.1.1 in
./.local/perlmutter/pytorch1.13.1/lib/python3.9/site-packages (from pyro-ppl)
(0.1.2)
Requirement already satisfied: tqdm>=4.36 in
/global/common/software/nersc/pm-2022q4/sw/pytorch/1.13.1/lib/python3.9/site-
packages (from pyro-ppl) (4.64.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
./.local/perlmutter/pytorch1.13.1/lib/python3.9/site-packages (from pyro-ppl)
(3.3.0)
Requirement already satisfied: numpy>=1.7 in
/global/common/software/nersc/pm-2022q4/sw/pytorch/1.13.1/lib/python3.9/site-
packages (from pyro-ppl) (1.23.4)
Requirement already satisfied: torch>=1.11.0 in
/global/common/software/nersc/pm-2022q4/sw/pytorch/1.13.1/lib/python3.9/site-
packages (from pyro-ppl) (1.13.1)
Requirement already satisfied: typing_extensions in
/global/common/software/nersc/pm-2022q4/sw/pytorch/1.13.1/lib/python3.9/site-
packages (from torch>=1.11.0->pyro-ppl) (4.4.0)
```

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

[43]:
```
import torch
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Function to generate 3D sphere points
def generate_2d_manifold_points(num_points, radius=1.0):
    phi = torch.rand(num_points) * 2 * torch.pi
    theta = torch.rand(num_points) * torch.pi

    x = radius * torch.cos(phi)/(1+torch.sin(theta)**2)
    y = radius * phi * torch.sin(phi)* torch.cos(phi)/(1+torch.cos(theta)**2)
    z = radius * torch.cos(theta)

    points_3d = torch.stack([x, y, z], dim=1)

    return points_3d

# Generate 1000 points on a 3D sphere with a radius of 1.0
num_points = 1000
sphere_points = generate_2d_manifold_points(num_points)

# Perform k-means clustering on the points
num_clusters = 3  # Adjust the number of clusters as needed
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
sphere_points_np = StandardScaler().fit_transform(sphere_points.numpy())  #␣
 ↪Standardize the data for k-means
labels = kmeans.fit_predict(sphere_points_np)

# Plot the 3D points using a scatter plot with coloring based on k-means clusters
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot with colored points
scatter = ax.scatter(sphere_points[:, 0], sphere_points[:, 1], sphere_points[:,␣
 ↪2], c=labels, cmap='viridis')

# Legend
legend = ax.legend(*scatter.legend_elements(), title="Clusters")
ax.add_artist(legend)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
```

```
ax.set_title('2D Manifold with K-Means Clustering')

plt.show()
```



[35]:
```
Y=sphere_points
# license: Copyright (c) 2014, the Open Data Science Initiative
# license: https://www.elsevier.com/legal/elsevier-website-terms-and-conditions
URL = "https://raw.githubusercontent.com/sods/ods/master/datasets/guo_qpcr.csv"
import pandas as pd
#df = pd.read_csv(URL, index_col=0)
df=pd.DataFrame(Y)
print("Data shape: {}\n{}\n".format(df.shape, "-" * 21))
print("Show a small subset of the data:")
df.head()
```

```
Data shape: (1000, 3)
---------------------
```

Show a small subset of the data:

[35]:
```
          0         1         2
0 -0.319155  1.434504  0.557880
1  0.013949  0.033985 -0.405001
2  0.376921  0.305555  0.689540
3  0.888790 -0.953122  0.968192
4 -0.510979 -0.618663  0.879811
```

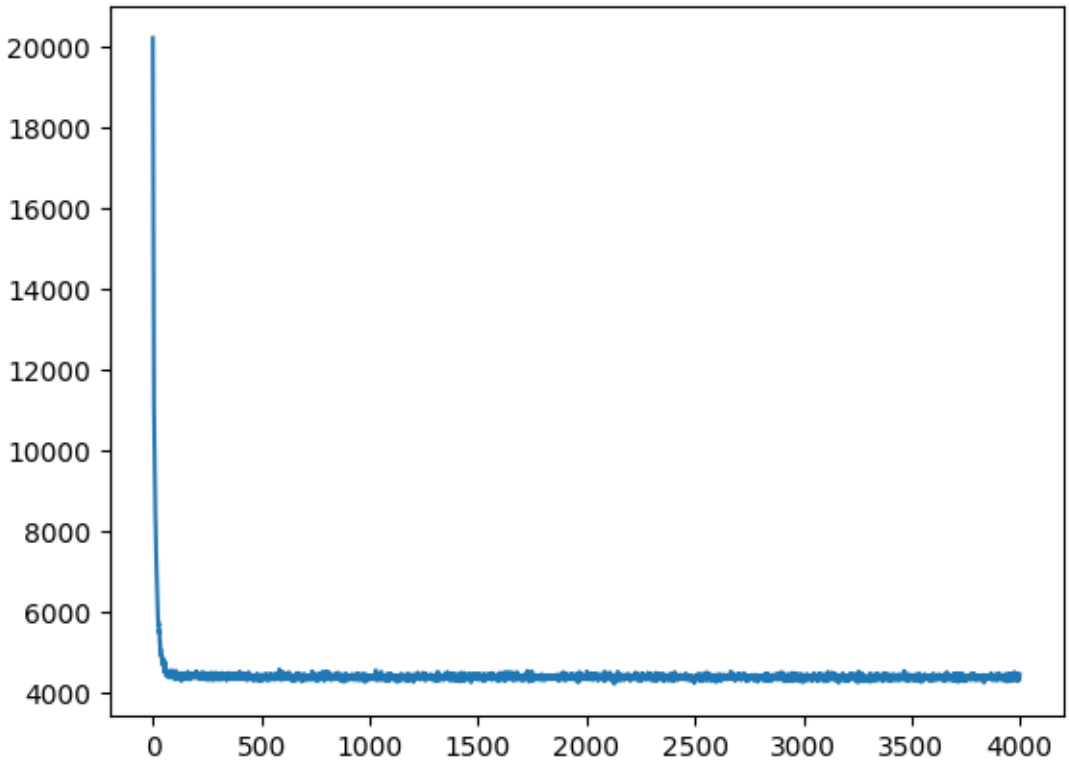
[36]:
```
data = torch.tensor(df.values, dtype=torch.get_default_dtype())
# we need to transpose data to correct its shape
y = data.t()
```

[37]:
```
#capture_time = y.new_tensor([int(cell_name.split(" ")[0]) for cell_name in df.
 ↪index.values])
# we scale the time into the interval [0, 1]
#time = capture_time.log2() / 6
y.size(1)
# we setup the mean of our prior over X
X_prior_mean = torch.zeros(y.size(1), 3)  # shape: 437 x 2
```

[38]:
```
kernel = gp.kernels.RBF(input_dim=2, lengthscale=torch.ones(2))

# we clone here so that we don't change our prior during the course of training
X = Parameter(X_prior_mean.clone())

# we will use SparseGPRegression model with num_inducing=32;
# initial values for Xu are sampled randomly from X_prior_mean
Xu = stats.resample(X_prior_mean.clone(), 100)
gplvm = gp.models.SparseGPRegression(X, y, kernel, Xu, noise=torch.tensor(0.01),␣
 ↪jitter=1e-5)
```

[39]:
```
# we use `.to_event()` to tell Pyro that the prior distribution for X has no␣
 ↪batch_shape
gplvm.X = pyro.nn.PyroSample(dist.Normal(X_prior_mean, 0.1).to_event())
gplvm.autoguide("X", dist.Normal)
```

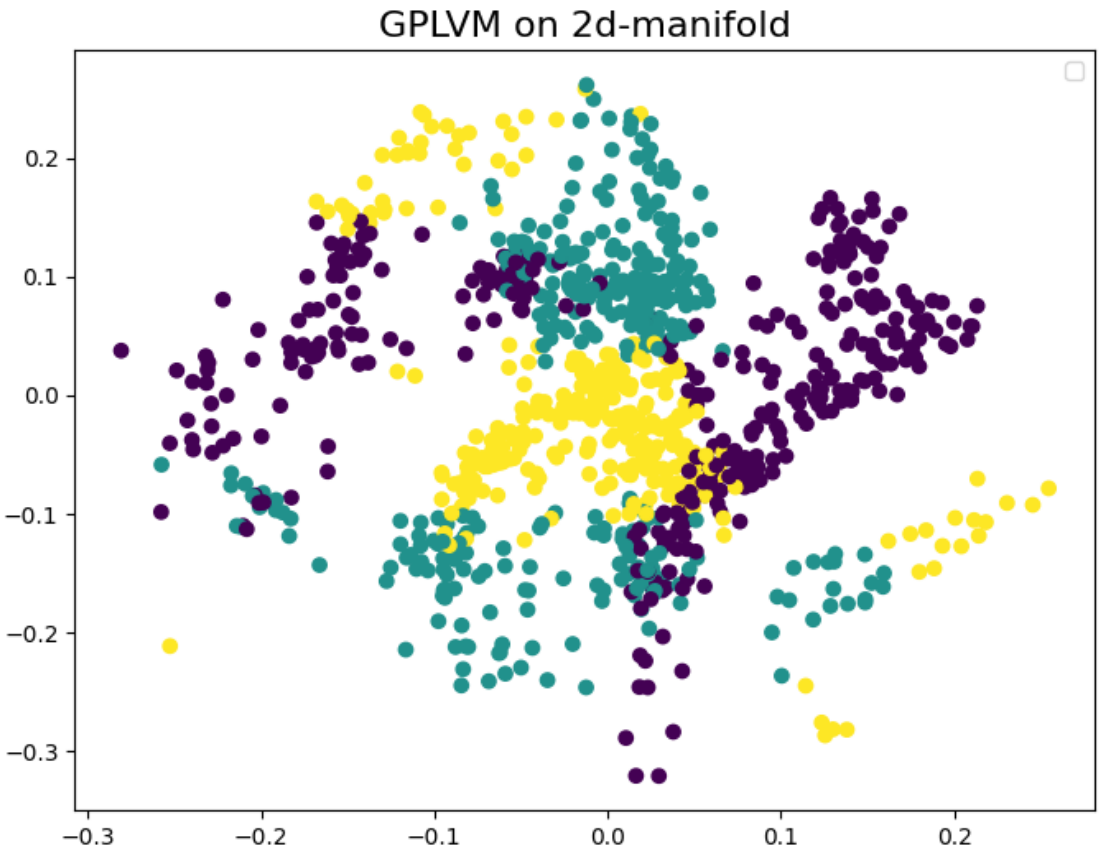
[40]:
```
# note that training is expected to take a minute or so
losses = gp.util.train(gplvm, num_steps=4000)

# let's plot the loss curve after 4000 steps of training
plt.plot(losses)
plt.show()
```



[41]:
```
gplvm.mode = "guide"
X = gplvm.X  # draw a sample from the guide of the variable X
```

[42]:
```
plt.figure(figsize=(8, 6))
labels = kmeans.fit_predict(sphere_points_np)
X = gplvm.X_loc.detach().numpy()
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')

plt.legend()
plt.title("GPLVM on 2d-manifold", fontsize=16)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```


GPLVM on 2d-manifold
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.3
−0.2
−0.1
0.0
0.1
0.2

[ ]: