

Experiment 6: GMM Clustering after Autoencoder Dimensionality Reduction

Imports and Configuration

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%cd ..
from src.models.autoencoder import Autoencoder
from src.models.gmm import GMM, Covariance
from src.utils.metrics import *
from src.utils.utils import *

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)

c:\Users\ziade\OneDrive\Desktop\Term 7\ML\Lab 4\Assignment4-
Unsupervised-Clustering-Analysis
```

Data Loading and Preprocessing

```
X_scaled, y_true =
load_scale_data(data_path='./data/breast_cancer.csv')

print(f"Dataset shape: {X_scaled.shape}")
print(f"Labels shape: {y_true.shape}")

Dataset shape: (569, 30)
Labels shape: (569,)
```

Comprehensive Grid Search: Dimensions & Covariance

```
bottleneck_dims = [2, 5, 10, 15, 20]
cov_types = [Covariance.FULL, Covariance.TIED, Covariance.DIAGONAL,
Covariance.SPHERICAL]
ae_gmm_results = []

for b_dim in bottleneck_dims:
    # 1. Train/Load Autoencoder for this dimension
    ae = Autoencoder(input_dim=X_scaled.shape[1], encoding_dims=[64,
32], bottleneck_dim=b_dim)
    ae.fit(X_scaled, verbose=False)
    X_latent = ae.encode(X_scaled)

    for cov_t in cov_types:
```

```

# 2. Fit GMM (k=2)
gmm = GMM(k=2, covariance_type=cov_t, max_iter=100)
gmm.fit(X_latent)
y_pred = gmm.predict(X_latent)

# 3. Collect Evaluation
metrics = evaluate_clustering(X_latent, y_true, y_pred, gmm)
metrics.update({'Dimensions': b_dim, 'Covariance':
cov_t.value})
ae_gmm_results.append(metrics)

results_df = pd.DataFrame(ae_gmm_results)
results_df

```

```

c:\Users\ziade\OneDrive\Desktop\Term 7\ML\Lab 4\Assignment4-
Unsupervised-Clustering-Analysis\src\utils\metrics.py:34:
RuntimeWarning: invalid value encountered in scalar divide
  s_scores[i] = (b_i - a_i) / max(a_i, b_i) if max(a_i, b_i) > 0 else
0

```

	ARI	NMI	Purity	Silhouette	Log-Likelihood	
BIC \						
0	0.000000	0.000000	0.627417	NaN	6420.872711	-
12771.962737						
1	0.000000	0.000000	0.627417	NaN	6420.872711	-
12790.994378						
2	0.000000	0.000000	0.627417	NaN	6420.872711	-
12784.650498						
3	0.000000	0.000000	0.627417	NaN	6420.872711	-
12797.338259						
4	-0.017620	0.030042	0.627417	-0.064229	11741.216963	-
23222.334828						
5	0.388045	0.391450	0.811951	0.591929	11301.470975	-
22438.001058						
6	-0.016529	0.028101	0.627417	-0.067312	11741.038667	-
23348.855844						
7	0.510800	0.460821	0.857645	0.619842	-662.577313	
1407.625072						
8	-0.005552	0.009376	0.627417	0.543051	10952.447198	-
21073.846059						
9	0.048242	0.038669	0.627417	0.095068	10799.539555	-
21116.944197						
10	0.261903	0.341524	0.757469	0.099023	12519.620450	-
24779.141803						
11	0.514098	0.394824	0.859402	0.466654	-6030.726997	
12207.363244						
12	0.587908	0.467511	0.884007	0.336311	11230.785141	-
20742.378684						
13	0.127130	0.083486	0.695958	0.327008	8361.641294	-
15765.356642						

14	0.514990	0.407329	0.859402	0.345780	8687.299832	-
15	0.677307	0.553601	0.912127	0.466862	-7757.470727	
16	0.243509	0.339005	0.748682	0.174401	14169.229310	-
17	-0.022562	0.011042	0.627417	-0.017639	6991.338021	-
18	0.244372	0.297746	0.748682	0.129326	9891.691566	-
19	-0.009206	0.058746	0.627417	0.178778	-7612.007214	

	AIC	Dimensions	Covariance
0	-12819.745422	2	full
1	-12825.745422	2	tied
2	-12823.745422	2	diagonal
3	-12827.745422	2	spherical
4	-23400.433925	5	full
5	-22550.941950	5	tied
6	-23440.077333	5	diagonal
7	1351.154626	5	spherical
8	-21642.894396	10	full
9	-21447.079110	10	tied
10	-24957.240900	10	diagonal
11	12107.453994	10	spherical
12	-21919.570282	15	full
13	-16421.282587	15	tied
14	-17252.599664	15	diagonal
15	15580.941454	15	spherical
16	-27416.458621	20	full
17	-13480.676043	20	tied
18	-19621.383131	20	diagonal
19	15310.014427	20	spherical

Analysis of Optimal Covariance Type

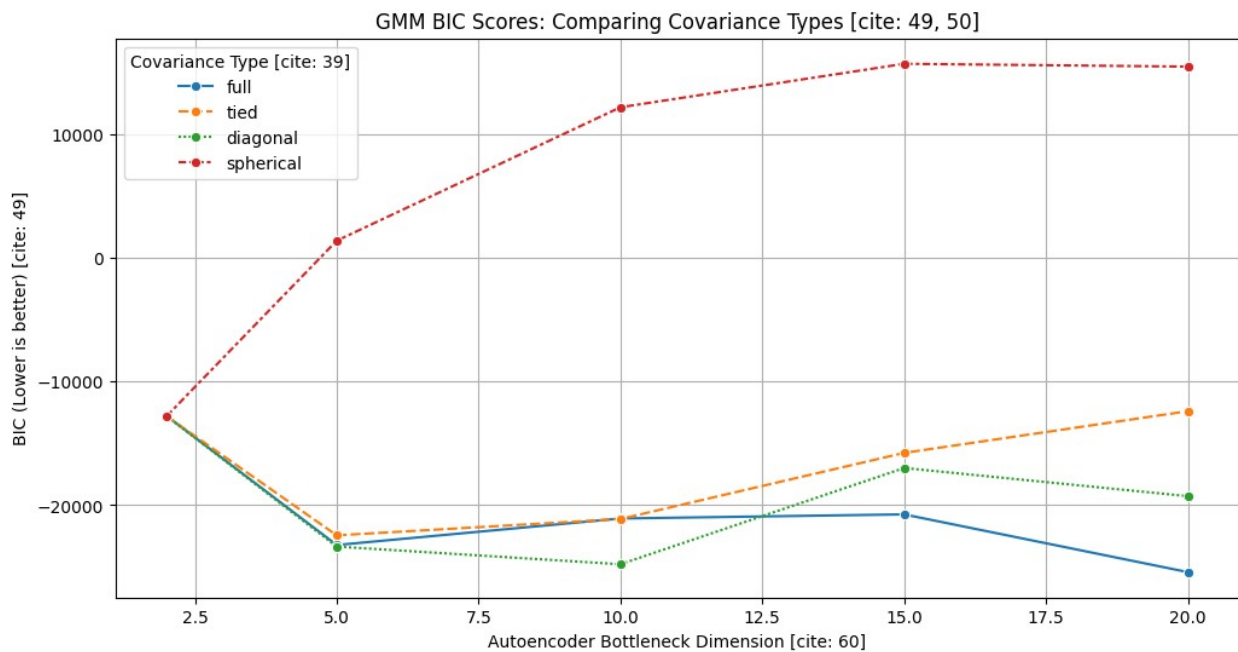
```
# Check if any covariance types resulted in NaN/Inf values
print(results_df.groupby('Covariance')['BIC'].describe())

plt.figure(figsize=(12, 6))
# Filter out non-finite values
clean_df = results_df[np.isfinite(results_df['BIC'])]

sns.lineplot(data=clean_df, x='Dimensions', y='BIC', hue='Covariance',
marker='o', style='Covariance')
plt.title("GMM BIC Scores: Comparing Covariance Types [cite: 49, 50]")
plt.ylabel("BIC (Lower is better) [cite: 49]")
plt.xlabel("Autoencoder Bottleneck Dimension [cite: 60]")
```

```
plt.grid(True, which="both", ls="-")
plt.legend(title="Covariance Type [cite: 39]")
plt.show()
```

	count	mean	std	min
25% \				
Covariance				
diagonal	5.0	-19433.959984	4850.832335	-24779.141803 -
23348.855844				
full	5.0	-20644.890410	4784.439699	-25413.929740 -
23222.334828				
spherical	5.0	6407.748170	12215.573460	-12797.338259
1407.625072				
tied	5.0	-16900.331666	4662.624011	-22438.001058 -
21116.944197				
	50%	75%	max	
Covariance				
diagonal	-19269.528816	-16987.622958	-12784.650498	
full	-21073.846059	-20742.378684	-12771.962737	
spherical	12207.363244	15496.801286	15724.289509	
tied	-15765.356642	-12790.994378	-12390.362054	



Visualizing Best Latent Clusters

```
# Select best performing config (highest ARI)
best_config = results_df.loc[results_df['ARI'].idxmax()]
print(f"Best Config: Dim={best_config['Dimensions']},
```

```

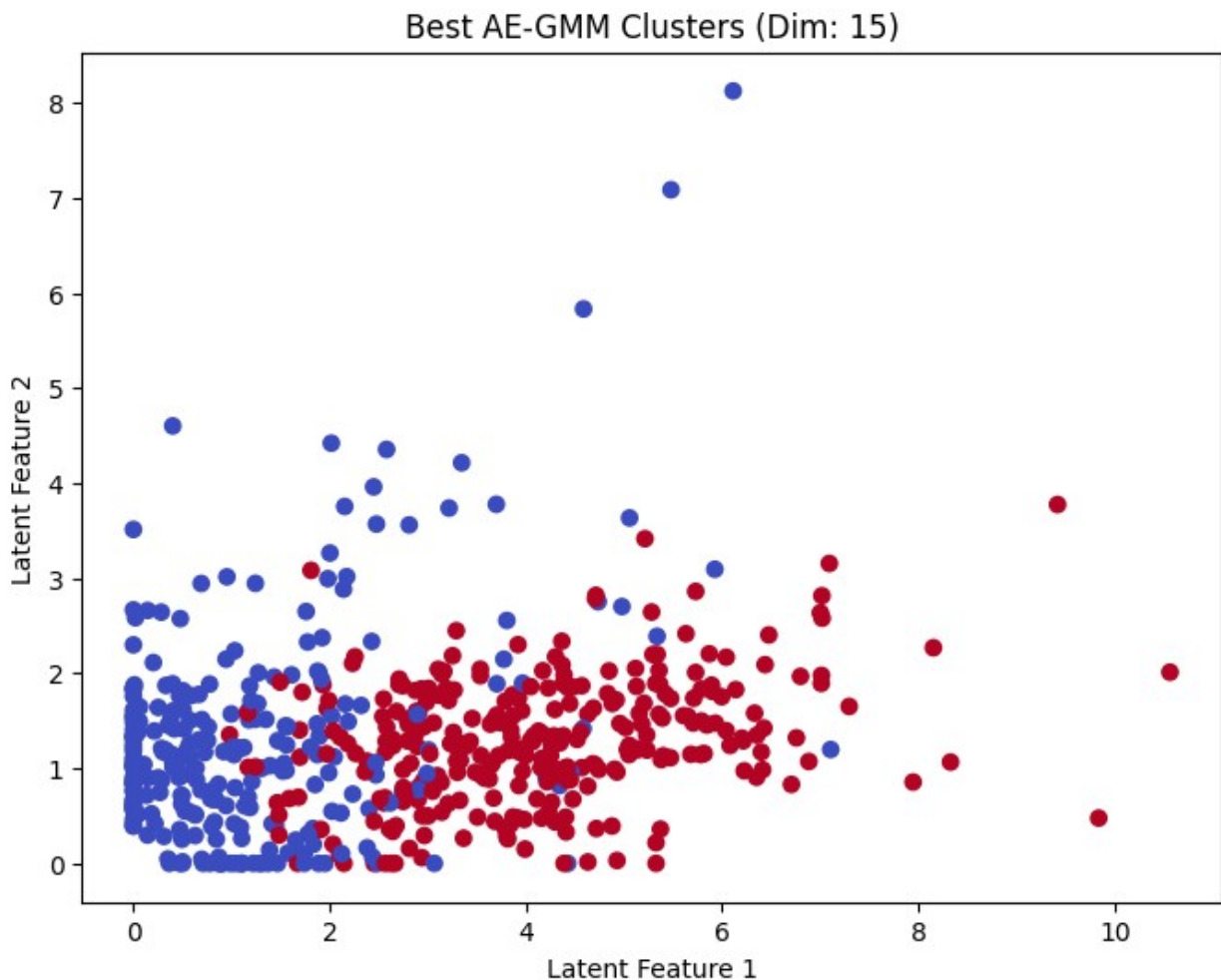
Cov={best_config['Covariance']})

# Re-run best for visualization
ae_best = Autoencoder(X_scaled.shape[1], [64, 32],
best_config['Dimensions']).fit(X_scaled, verbose=False)
X_best_latent = ae_best.encode(X_scaled)
gmm_best = GMM(k=2,
covariance_type=Covariance(best_config['Covariance']))
gmm_best.fit(X_best_latent)

plt.figure(figsize=(8, 6))
plt.scatter(X_best_latent[:, 0], X_best_latent[:, 1],
c=gmm_best.predict(X_best_latent), cmap='coolwarm')
plt.title(f"Best AE-GMM Clusters (Dim: {best_config['Dimensions']})")
plt.xlabel("Latent Feature 1")
plt.ylabel("Latent Feature 2")
plt.show()

Best Config: Dim=15, Cov=spherical

```

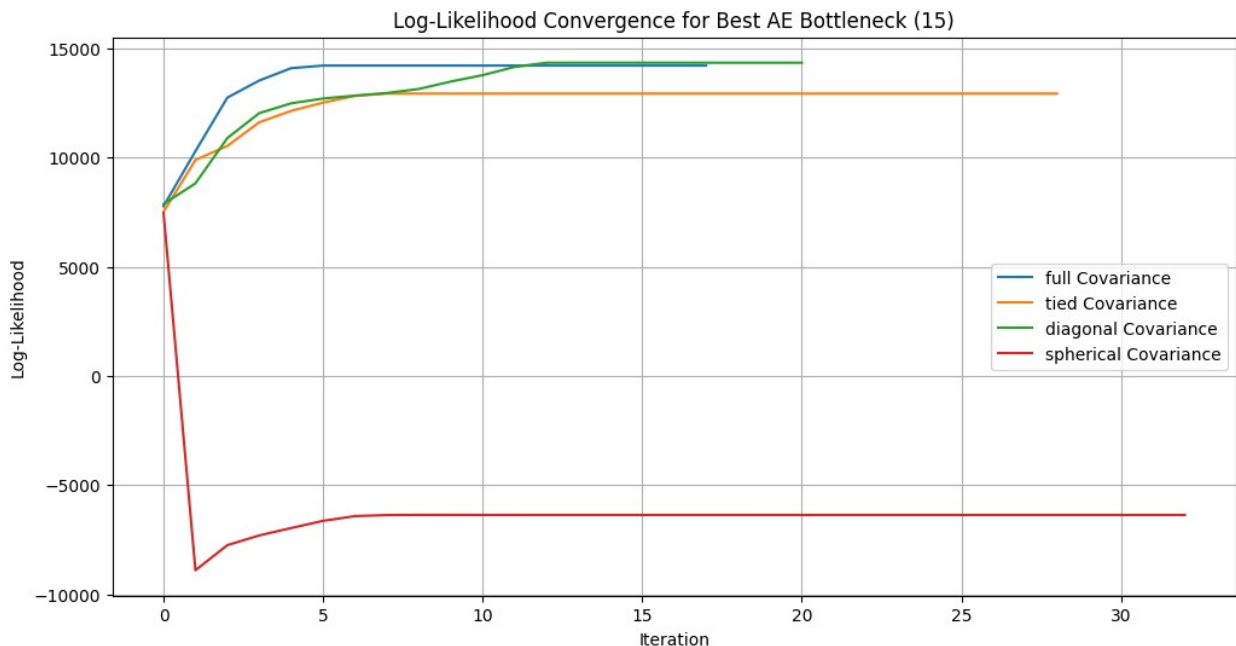


```

plt.figure(figsize=(12, 6))
for cov_t in cov_types:
    # Fit GMM again to get log-likelihood history
    gmm = GMM(k=2, covariance_type=cov_t, max_iter=100)
    gmm.fit(X_best_latent)
    plt.plot(gmm.log_likelihood_, label=f'{cov_t.value} Covariance')

plt.title(f"Log-Likelihood Convergence for Best AE Bottleneck
({best_config['Dimensions']})")
plt.xlabel("Iteration")
plt.ylabel("Log-Likelihood")
plt.legend()
plt.grid(True)
plt.show()

```

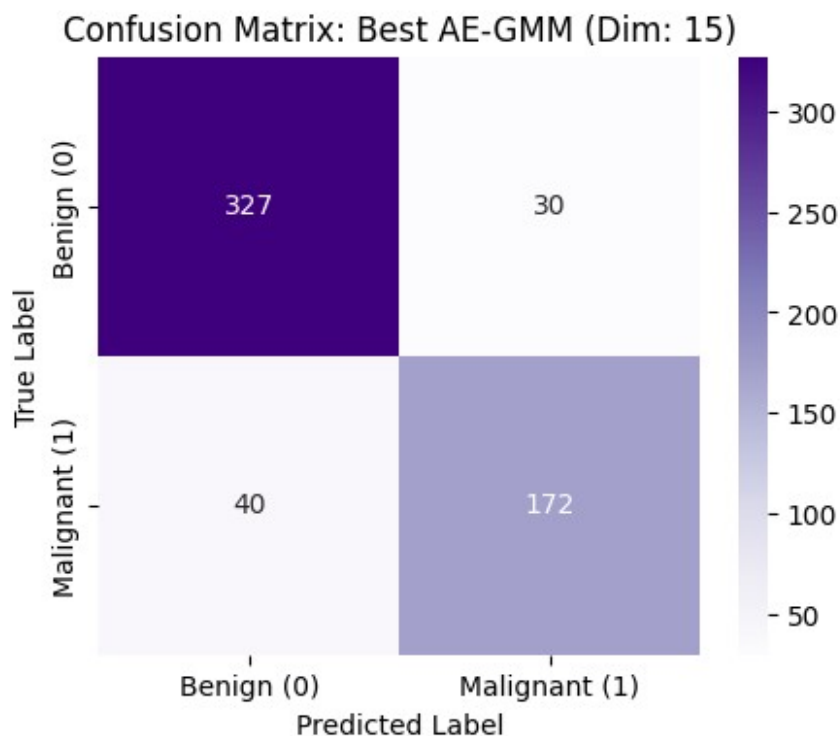


```

# Map clusters to true labels
y_best_pred = gmm_best.predict(X_best_latent)
y_best_aligned = align_clusters_with_labels(y_true, y_best_pred)

cm_ae_gmm = compute_confusion_matrix(y_true, y_best_aligned)
plt.figure(figsize=(5, 4))
sns.heatmap(cm_ae_gmm, annot=True, fmt='d', cmap='Purples',
            xticklabels=['Benign (0)', 'Malignant (1)'],
            yticklabels=['Benign (0)', 'Malignant (1)'])
plt.title(f"Confusion Matrix: Best AE-GMM (Dim:
{best_config['Dimensions']})")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()

```



Final Comparison Table

```
summary_comparison = results_df.sort_values(by='ARI',
ascending=False).head(10)
print("Top 10 AE-GMM Configurations:")
display(summary_comparison)
```

Top 10 AE-GMM Configurations:

	ARI	NMI	Purity	Silhouette	Log-Likelihood
BIC \					
15	0.677307	0.553601	0.912127	0.466862	-7757.470727
15724.289509					
12	0.587908	0.467511	0.884007	0.336311	11230.785141 -
20742.378684					
14	0.514990	0.407329	0.859402	0.345780	8687.299832 -
16987.622958					
11	0.514098	0.394824	0.859402	0.466654	-6030.726997
12207.363244					
7	0.510800	0.460821	0.857645	0.619842	-662.577313
1407.625072					
5	0.388045	0.391450	0.811951	0.591929	11301.470975 -
22438.001058					
10	0.261903	0.341524	0.757469	0.099023	12519.620450 -
24779.141803					
18	0.244372	0.297746	0.748682	0.129326	9891.691566 -
19269.528816					

16	0.243509	0.339005	0.748682	0.174401	14169.229310	-
25413.929740						
13	0.127130	0.083486	0.695958	0.327008	8361.641294	-
15765.356642						

	AIC	Dimensions	Covariance
15	15580.941454	15	spherical
12	-21919.570282	15	full
14	-17252.599664	15	diagonal
11	12107.453994	10	spherical
7	1351.154626	5	spherical
5	-22550.941950	5	tied
10	-24957.240900	10	diagonal
18	-19621.383131	20	diagonal
16	-27416.458621	20	full
13	-16421.282587	15	tied