GRAPHICAL LASSO (GLASSO) - Penalization for Undirected Graphs:

The Graphical Lasso (Glasso) algorithm is used for sparse precision matrix (inverse covariance) estimation in undirected graphical methods. It applies L1 (Lasso) regularization to encourage sparsity, helping identify conditional independence relationships between variables.

Problem statement:

Given a dataset X with p variables, the goal is to estimate the precision matrix $\Theta = \Sigma^{-1}$, where Σ is the covariance matrix. The Graphical Lasso involves:

$$\boxed{\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \left(tr(\Sigma\Theta) - \log\det(\Theta) + \lambda \sum_{i \neq j} |\Theta_{ij}| \right)}$$

where:

→ $tr(\Sigma\Theta)$ ensures the solution matches observed covariance structure.

→ $-\log\det(\Theta)$ ensures positive definiteness.

→ $\lambda \sum_{i \neq j} |\Theta_{ij}|$ is the L_1 penalty, shrinking small elements to zero, forcing sparsity.

Python Example:

Estimating a sparse Precision Matrix:

```
# import libraries:
import numpy as np
from sklearn.covariance import GraphicalLasso
```

Semester : VIII Subject : AIFB Academic Year: 2024-25

```
# Generate synthetic data (100 samples, 5 variables)
np.random.seed(42)
X = np.random.randn(100,5)

# Apply Graphical Lasso.
glasso = GraphicalLasso(alpha=0.1) # λ alpha controls sparsity.
glass.fit(X)

# Estimated precision matrix (Θ)
print("Estimated Precision Matrix : \n", glasso.precision)
```

~~# Visuali~~ Output:

Estimated Precision Matrix :

```
[[1.249   0.056   -0.     0.     0.022]
 [0.056   1.045   -0.019  -0.    -0.   ]
 [-0.     -0.019  1.043   -0.    -0.   ]
 [0.      -0.     -0.     1.097  -0.   ]
 [0.022   -0.     -0.     -0.     0.878]
```

Understanding the Output:-

* Non-zero elements in the precision matrix indicate direct relationships between variables.
* Zero elements imply conditional independence (eg. if $\Theta_{ij} = 0$, variables i and j are independent given all other variables).
* Higher λ (alpha) → More sparsity → Fewer dependencies.
* Lower λ (alpha) → Less sparsity → More dependencies.