Notes Date / /

Nifty fifty - portfolio optimization project

Goal - Construct portfolio of Nifty 50 by leveraging GCN & Spectral Graph Theory

* GCN - learn relationship of stock's and use it to allocate weights.
- optimize return and risk

* GCN & Spectral Graph theory
→ 1.) Stock market as network
2.) Spectral graph theory

#. Stocks - nodes
#. Correlation - edges.

* Use spectral graph techniques to cluster (here simple k-mean's clustering) → to get diversified stocks.

* GCN for feature ~~engineering~~ (learning)!
1) aggregate info from stock neighbors
2) enable stock to $\boxed{\text{learn}}$ from Co-related peers

3) This capture pattern in stock relationship's and assign weight.

$\boxed{\text{Algorithm}}$:-

Step 1 :- Data
*) Fetch data for Nifty 50, (market cap);
*) Compute daily returns.
*) Analyze correlation matrix for daily returns.

Step 2 :- Graph.
→ *) Stock as node ; Edges - correlation b/w;
[apply threshold : computationally manageable]
→ essential for GCN.

Step 3 :- Spectral clustering :- (optional).
*) Graphs Laplacian (or adjacency) Cluster highly correlated using eigenvalue and eigen vectors, and find strong connections

Step 4 :- Train :- GCN to learn graphs

(*) ~~learn~~ layers that learn node (stock) embeddings by aggregating info from neighbouring nodes

(*) trained to output representation for each stock :-

(*) this output is used for portfolio weighting :-

→ GCN captures dependencies b/w stocks

→ learn a $\boxed{\text{representation}}$?

Process :- 1.) Train and validate.

Step 5 :- Generate Portfolio weights using node. embedding :-

(*) Extract $\boxed{\text{node embedding}}$ - from GCN (based on position & connections).

(*) Node embedding ~ proxy ~ weight's.

(*) Convert embedding to weights :

[take mean and normalise ... to 1 ; over 100%]

Process :- (1.) Use validation data, where weights are applied to get returns over the validation period

Step 6 :- Backtesting :-

(1) Calculate weighted portfolio return on validation data (from assigned weights).

(2) Evaluate performance metrics

→ Cumulative returns :- total growth
→ Annualized volatility → Risk.
→ Sharpe ratio → Risk adjusted return.

Notes :-

*) Spectral Analysis (Clustering)

→ Find laplacian matrix ; $L = D_{ii} -$ Adjacency matrix
or normalised laplacian //

→ Compute eigen values & eigen vectors

→ Smallest - k - eigen vectors (those corresponding
? to smallest non-zero eigen values) ~~to~~ will
represent the graph in lower dim space.

→ This eigen vector tells all properties.

→ Form a matrix U from this Small - eigen - vectors
row - node ; column - corresponds to an eigen vector
↳ low dim representation of a node
Captures graph -) (tells 'Strong connections).

→ Apply k-means clustering: on each row treating it as data point in lower dim Space

→ Resulting clusters corresponds to groups in original graph.

*) k-means Clustering:-

*) Unsupervised Machine learning algo
*) Create k-clusters.
*) Initial k-centroids ; calculate closed centroid using Euclidean distance and repeat.

Notes Date / /

*) GCN

→ trained to learn individual characteristic & corelation relationship

Individual Characteristics (Node features). : Return, Volatility, Indicator, Setiment Analysis.

→ Forward pass + Loss calculation using node embedding representation

Backpropogation + ←

(compute gradients using Adam).

(*) After training; each stock has a learned embedding

(*) Use this embedding to assign weights by normalising

*) Embedding :- Low dim → representation of high dim data. capturing key features in numerical form (stock characteristic)

* In this project :- vector form of self and correlations of stocks are embedding (outer layer)

(50 stocks, 8 nodes) : output layer. pattern //.

From here aggregate (take mean) -