Complete Guide: L1 Regularization, Matrix Factorization and Spiking Neural Networks in City Networks

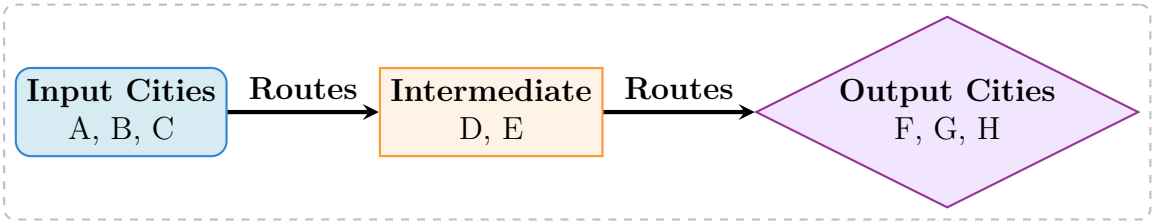
Based on Detailed Discussion

March 28, 2026

Contents

1 Complete Mind Map: City Network with L1 Regularization

2 Problem Statement: City Network Example



2.1 Problem Overview

- **Input Features:** Cities A, B, C (with attributes: Traffic, Distance, Cost)
- **Intermediate Nodes:** D, E (hidden transformations)
- **Output Targets:** F, G, H (destinations)
- **Goal:** Predict unknown routes F,G,H $\rightarrow$ D,E using known routes F,G,H $\rightarrow$ A,B,C

2.2 Known Data Matrix

$$X = \begin{bmatrix} 0.9 & 0.8 & 0.7 \\ 0.6 & 0.7 & 0.5 \\ 0.4 & 0.5 & 0.6 \end{bmatrix} \tag{1}$$

Where rows = F,G,H (sources) and columns = A,B,C (destinations)

2.3 Unknown Routes to Predict

$$\begin{bmatrix} ? & ? \\ ? & ? \\ ? & ? \end{bmatrix} \text{ for F,G,H} \rightarrow \text{D,E} \tag{2}$$

3 Data Preprocessing Pipeline

3.1 Raw Data Collection

City	Traffic (cars/hr)	Distance (km)	Cost ( )
A	100	10	200
B	30	60	180
C	20	100	150

Table 1: Raw data for cities

3.2 Normalization (Min-Max Scaling)

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3}$$

City	Traffic	Distance	Cost
A	1.0	0.1	0.5
B	0.3	0.6	0.4
C	0.2	1.0	0.3

Table 2: Normalized features

3.3 Feature Combination Formula

$$x = w_1 \cdot \text{Traffic} + w_2 \cdot (1 - \text{Distance}) + w_3 \cdot (1 - \text{Cost}) \tag{4}$$

For our example:

$$x = 0.5(\text{Traffic}) + 0.3(1 - \text{Distance}) + 0.2(1 - \text{Cost}) \tag{5}$$

3.4 Final Input Vector

$$X = [1, 0.3, 0.2] \tag{6}$$

4 Linear Regression with L1 Regularization

4.1 Linear Model

$$\hat{Y} = XW + b \tag{7}$$

where $X = [A, B, C]$, W = weights, b = bias

4.2 L1 Regularization (Lasso)

$$\text{Loss} = \text{MSE} + \lambda \sum_{i=1}^{n} |w_i| \tag{8}$$

In matrix form:

$$\text{Loss} = ||Y - XW||^2 + \lambda ||W||_1 \tag{9}$$

4.3 Gradient Descent Update Rule

$$W_{\text{new}} = W_{\text{old}} - \eta \left(\frac{\partial \text{MSE}}{\partial W} + \lambda \cdot \text{sign}(W) \right) \tag{10}$$

4.3.1 Understanding Each Term

- η = learning rate
- $\frac{\partial \text{MSE}}{\partial W} = X^T(XW - Y)$ = error gradient
- $\lambda \cdot \text{sign}(W)$ = L1 penalty gradient

4.4 Why $X^T(XW - Y)$?

$$X^T(XW - Y) = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{11}$$

$$= \begin{bmatrix} A_1e_1 + A_2e_2 + A_3e_3 \\ B_1e_1 + B_2e_2 + B_3e_3 \\ C_1e_1 + C_2e_2 + C_3e_3 \end{bmatrix} \tag{12}$$

Intuition: This calculates how much each feature contributed to the error - like assigning blame to students based on their scores and mistakes!

5 L1 Regularization: Detailed Mechanics

5.1 How Weights Become Zero

Step	Weight A (large)	Weight B (small)	Weight C (very small)
Initial	2.0	0.2	-0.1
Step 1	1.9	0.1	0
Step 2	1.8	0	0
Step 3	1.7	0	0
⋮	⋮	⋮	⋮
Final	1.5	0	0

Table 3: Iterative weight shrinkage with $\eta\lambda = 0.1$

5.2 Soft Thresholding Formula

$$w_{\text{new}} = \begin{cases} w - \lambda & \text{if } w > \lambda \\ w + \lambda & \text{if } w < -\lambda \\ 0 & \text{if } |w| \leq \lambda \end{cases} \tag{13}$$

5.3 Penalty Concept

Penalty = extra cost imposed on the model when weights are large

$$\text{Loss}_{\text{with L1}} = \underbrace{\text{Error}}_{\text{prediction mistake}} + \underbrace{\lambda \sum |w_i|}_{\text{complexity penalty}} \tag{14}$$

5.4 Sign Function Role

$$\text{sign}(w) = \begin{cases} +1 & \text{if } w > 0 \\ 0 & \text{if } w = 0 \\ -1 & \text{if } w < 0 \end{cases} \tag{15}$$

Purpose: Ensures weights are always pushed toward zero regardless of sign

- Positive weights: subtract λ
- Negative weights: add λ

6 City Example: L1 Regularization in Action

6.1 Initial Network

$$D = w_{AD}A + w_{BD}B + w_{CD}C \tag{16}$$
$$E = w_{AE}A + w_{BE}B + w_{CE}C \tag{17}$$

6.2 Initial Weights

$$w_{AD} = 2.0 \quad \text{(strong road)} \tag{18}$$
$$w_{BD} = 0.2 \quad \text{(weak road)} \tag{19}$$
$$w_{CD} = -0.1 \quad \text{(weak negative)} \tag{20}$$

6.3 After L1 Regularization ($\eta\lambda = 0.1$)

Connection	**Before**	**After**
A $\rightarrow$ D	2.0	1.9
B $\rightarrow$ D	0.2	0
C $\rightarrow$ D	-0.1	0

Table 4: Weight changes after L1

6.4 Final Sparse Model

$$D = 1.9A + 0B + 0C \tag{21}$$
$$E = 0.9A + 0B + 0C \quad \text{(assuming similar for E)} \tag{22}$$

6.5 Analogy: Toll Tax on Roads

- **L1** = toll tax on every road
- **Small road** (B $\rightarrow$ D): toll feels expensive $\rightarrow$ close the road (weight = 0)
- **Highway** (A $\rightarrow$ D): toll manageable $\rightarrow$ road survives

7 Matrix Factorization

7.1 Basic Concept

$$X \approx U \cdot V^T \tag{23}$$

where:

- U = source city nature (F,G,H)
- V = destination city nature (A,B,C,D,E)

7.2 Learned Hidden Factors (Example)

Source Cities Nature (U):

City	Fast Routes	Cheap Routes
F	0.9	0.2
G	0.6	0.5
H	0.3	0.8

Destination Cities Nature (V):

City	Fast Routes	Cheap Routes
A	0.8	0.3
B	0.7	0.4
C	0.6	0.5
D	0.9	0.1
E	0.2	0.9

7.3 Predicting Missing Routes

$$X(F, D) = U_F \cdot V_D = [0.9, 0.2] \cdot [0.9, 0.1] \tag{24}$$
$$= 0.9 \times 0.9 + 0.2 \times 0.1 = 0.81 + 0.02 = 0.83 \tag{25}$$

$$X(F, E) = U_F \cdot V_E = [0.9, 0.2] \cdot [0.2, 0.9] \tag{26}$$
$$= 0.18 + 0.18 = 0.36 \tag{27}$$

7.4 Interpretation

- **F → D = 0.83**: High score means good route
- **F → E = 0.36**: Low score means poor route
- **Netflix Analogy**: Known movies (A,B,C) → learn user taste (U) → recommend new movies (D,E)

8 Eigenvectors and Eigenvalues

8.1 Core Equation

$$\Sigma v = \lambda v \tag{28}$$

where:

- Σ = covariance matrix of data
- v = eigenvector (direction of maximum variance)
- λ = eigenvalue (importance of that direction)

8.2 Covariance Matrix

$$\Sigma = \frac{1}{n} X^T X \tag{29}$$

8.3 City Example Interpretation

$$v = \begin{bmatrix} 0.8 \\ -0.5 \\ -0.1 \end{bmatrix} \tag{30}$$

Meaning:

- Traffic $\uparrow \rightarrow$ score $\uparrow$ (0.8)
- Distance $\uparrow \rightarrow$ score $\downarrow$ (-0.5)
- Cost $\rightarrow$ negligible effect (-0.1)

8.4 Eigenvalue Meaning

$$\lambda = 4.5 \tag{31}$$

Interpretation: This direction captures 4.5 units of variance - very important!

8.5 Multiple Eigenvectors

Eigenvector	Meaning	Eigenvalue
v_1	Best route direction	4.5
v_2	Second best direction	1.2
v_3	Useless direction	0.1

8.6 PCA Decision

- Keep v_1 (most important)
- Optionally keep v_2
- Drop v_3 (useless)

8.7 Geometric Intuition

- Data points form an ellipse in 3D space
- **Eigenvector** = longest axis of the ellipse
- **Eigenvalue** = length of that axis

9 Spiking Neural Networks (SNN)

9.1 Basic Firing Rule

$$\text{Spike} = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

where u = membrane potential, θ = threshold

9.2 City Example with SNN

Input: $A = 1$ (strong), $B = 0.3$ (weak), $C = 0.2$ (weak)

Weights:

$$A \to D = 0.8 \tag{33}$$
$$B \to D = 0.5 \tag{34}$$
$$C \to E = 0.6 \tag{35}$$

Threshold: $\theta = 0.7$

9.3 Neuron Calculations

D neuron:

$$u_D = 1 \times 0.8 + 0.3 \times 0.5 = 0.8 + 0.15 = 0.95 \tag{36}$$
$$0.95 > 0.7 \quad \Rightarrow \quad \text{FIRES!} \tag{37}$$

E neuron:

$$u_E = 0.2 \times 0.6 = 0.12 \tag{38}$$
$$0.12 < 0.7 \quad \Rightarrow \quad \text{SILENT} \tag{39}$$

9.4 Result: Sparse Activity

- Only D fires
- E silent
- F activates, G,H remain inactive

9.5 Comparison: L1 vs SNN

Aspect	**L1 Regularization**	**SNN**
What's sparse?	Weights	Neuron firing
Mechanism	$\lambda \cdot \text{sign}(w)$ in gradient	Threshold: fire if $u > \theta$
Result	Fewer connections	Fewer spikes
Analogy	Close unnecessary roads	Use roads only when needed

10 L1 + SNN: Ultra-Efficient Model

10.1 Combined Architecture

$$\text{L1:} \quad \text{Remove unnecessary connections} \tag{40}$$
$$\text{SNN:} \quad \text{Fire only when input strong} \tag{41}$$

10.2 Mathematical Formulation

$$\min_W \mathcal{L}_{\text{task}}(s^{(2)}, y) + \lambda \|W\|_1 \quad \text{s.t.} \quad s = H(u - \theta) \tag{42}$$

where H is the Heaviside step function.

10.3 Complete Update Rules

Output layer:

$$\frac{\partial L}{\partial W^{(2)}} = (s^{(2)} - y) \odot \sigma'(u^{(2)} - \theta^{(2)}) \cdot (s^{(1)})^\top + \lambda_2 \cdot \text{sign}(W^{(2)}) \tag{43}$$

Hidden layer:

$$\frac{\partial L}{\partial W^{(1)}} = [((s^{(2)} - y) \odot \sigma'(\cdot))W^{(2)\top} \odot \sigma'(u^{(1)} - \theta^{(1)})] \cdot x^\top + \lambda_1 \cdot \text{sign}(W^{(1)}) \tag{44}$$

10.4 Efficiency Comparison

Model	Connections	Active Neurons	Efficiency
Without L1+SNN	10	8	Low
With L1+SNN	3	1	Ultra-efficient

10.5 Real-World Applications

- **Brain-inspired AI**: Brain uses sparse connections + sparse firing
- **Edge devices / IoT**: Low power consumption
- **Neuromorphic hardware**: Chips like Intel Loihi

10.6 Key Insight

"L1 network (connections)
SNN network (firing)
: + = ultra-efficient AI"

11 Sparse vs Dense Algorithms

11.1 Algorithms Using Sparse Matrices

- **Lasso Regression (L1)**: Weights become zero
- **Sparse PCA**: Eigenvectors contain zeros
- **Matrix Factorization**: User-item matrices are sparse
- **PageRank**: Graph matrices are sparse
- **Linear SVM (text)**: TF-IDF is sparse
- **Naive Bayes**: Works well with sparse text data

11.2 Algorithms NOT Using Sparse Matrices

- **Standard PCA**: Eigenvectors are dense
- **Neural Networks (traditional)**: Weights mostly non-zero
- **K-Means Clustering**: Distance calculations assume dense vectors
- **Gaussian Mixture Models**: Covariance matrices are dense

11.3 Hybrid Cases

Some algorithms can work with both:

- Gradient Descent (SGD)
- Linear Regression
- Logistic Regression

Adding L1 regularization makes them sparse!

11.4 City Example Sparse Matrix

$$X = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{45}$$

This sparse matrix is ideal for:

- Matrix Factorization
- Sparse PCA
- L1 Regression

12 When to Use L1 Regularization

12.1 Ideal Situations

- **High Dimensional Data**: 1000 features, only 20 useful
- **Feature Selection Needed**: "Which routes actually matter?"
- **Sparse Model Desired**: Simple, fast, interpretable
- **Explainability Important**: "Why did model choose this?"

12.2 Real-World Applications

- **Healthcare**: Identify important medical tests from 1000 parameters
- **Finance**: Select key stock indicators from 500 features
- **Bioinformatics**: Find important genes from 10,000+
- **NLP**: Choose important words from millions of features

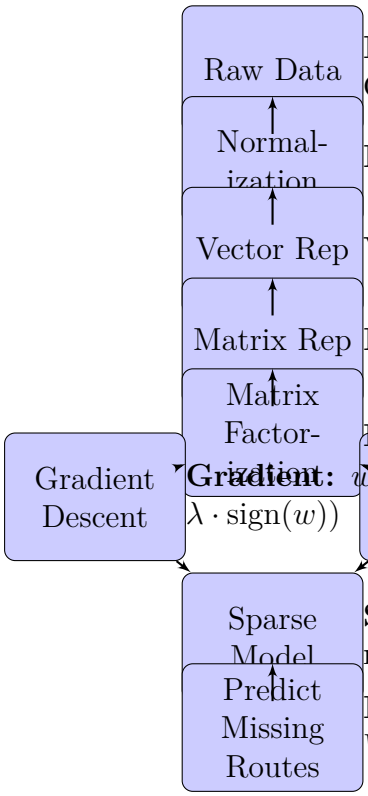
12.3 When NOT to Use L1

- **Highly Correlated Features**: L1 picks one, discards others (use L2/Ridge)
- **All Features Important**: L1 may zero out useful features

12.4 L1 vs L2 Comparison

Situation	L1 (Lasso)	L2 (Ridge)
Feature selection		
Correlated features		
Simple model		
Stability		

13 Complete Solution Pipeline



Raw Data: Traffic, Distance, Cost for each city

Normalization: $x' = \frac{x - \min}{\max - \min}$

Vector: $[1, 0.3, 0.2]$

Matrix: $X = \begin{bmatrix} 0.9 & 0.8 & 0.7 \\ 0.6 & 0.7 & 0.5 \\ 0.4 & 0.5 & 0.6 \end{bmatrix}$

Factorization: $X \approx UV^T$

Gradient: $w = w - \eta(\nabla \text{MSE} + \lambda \cdot \text{sign}(w))$ **L1:** $\lambda \sum |w_i|$ pushes small weights to zero

Sparse: $w = [0.7, 0, 0]$ (only A matters)

Prediction: $X(F, D) = U_F \cdot V_D = 0.83$

14 Key Insights Summary

14.1 Why L1 Makes Weights Zero

$$w_{\text{new}} = w_{\text{old}} - \eta(\nabla \text{MSE} + \lambda \cdot \text{sign}(w)) \tag{46}$$

- $\lambda \cdot \text{sign}(w)$ gives constant push toward zero
- Small weights ($|w| < \lambda$) get pushed to zero quickly
- Large weights shrink slowly but survive

14.2 Why $X^T(XW - Y)$ is Important

$$X^T(XW - Y) = \begin{bmatrix} A_1e_1 + A_2e_2 + A_3e_3 \\ B_1e_1 + B_2e_2 + B_3e_3 \\ C_1e_1 + C_2e_2 + C_3e_3 \end{bmatrix} \tag{47}$$

Intuition: Blame score for each feature - like multiplying student's score $\times$ mistake to find who's responsible

14.3 Eigenvectors vs L1

- **Eigenvectors**: Find best NEW directions (PCA)
- **L1**: Selects best EXISTING features (Lasso)
- Both find sparsity, but in different ways

14.4 L1 vs SNN

- **L1** = sparse WEIGHTS (fewer connections)
- **SNN** = sparse ACTIVITY (fewer spikes)
- **Together** = ultra-efficient (fewer roads + less traffic)

15 Mathematical Summary Table

Concept	**Formula**	**Meaning**
Linear Model	$\hat{Y} = XW$	Prediction
L1 Loss	$\text{Loss} = \text{MSE} + \lambda \sum \lvert w_i \rvert$	Error + Penalty
Gradient Update	$w = w - \eta(\nabla L + \lambda \cdot \text{sign}(w))$	Learning
Soft Thresholding	$w_{\text{new}} = \begin{cases} w - \lambda & w > \lambda \\ w + \lambda & w < -\lambda \\ 0 & \lvert w \rvert \leq \lambda \end{cases}$	Zero mechanism
Matrix Factorization	$X \approx UV^T$	Hidden factors
Eigen Equation	$\Sigma v = \lambda v$	Principal directions
SNN Firing	$s = H(u - \theta)$	Threshold-based
Covariance	$\Sigma = \frac{1}{n} X^T X$	Data structure
Normalization	$x' = \frac{x - \min}{\max - \min}$	Scaling

Table 5: Key mathematical formulas

16 Final One-Line Clarity

**complex city data (traffic, distance, cost)
hidden patterns (U,V)
gradient descent + L1 regularization ,
sparse model
important features (A) depend
unknown routes (D,E)
- — SNN efficient**

**Machine Learning Goal:
Complex Raw Data → Simple Hidden
Structure → Accurate Predictions**

Mathematics Power:
L1 = Feature Selection
Gradient Descent = Learning
Matrix Factorization = Hidden Structure
Eigenvectors = Principal Directions
SNN = Efficient Activity

Appendix: How to Read the Mind Map

- **Start from the top** - Understand the city network problem (A,B,C → D,E → F,G,H)
- **Follow the colored branches:**
  - **Blue**: Known data and preprocessing
  - **Green**: Unknown routes and predictions
  - **Purple**: Mathematical concepts (Matrix Factorization, Eigenvectors)
  - **Orange**: Learning mechanisms (Gradient Descent, L1)
  - **Red**: Advanced concepts (SNN, Combined models)
  - **Brown**: Data preprocessing steps
- **Dashed boxes** group related concepts together
- **Arrows** show the flow of information and relationships
- **Mathematical formulas** are included at each node for clarity

Key to Mathematical Symbols

Symbol	Meaning
X	Data matrix
W	Weight matrix
λ	Regularization strength
η	Learning rate
$\text{sign}(w)$	Direction of weight (+1 or -1)
Σ	Covariance matrix
v	Eigenvector
λ (in eigen)	Eigenvalue
U, V	Factor matrices
θ	SNN threshold
$H(\cdot)$	Heaviside step function

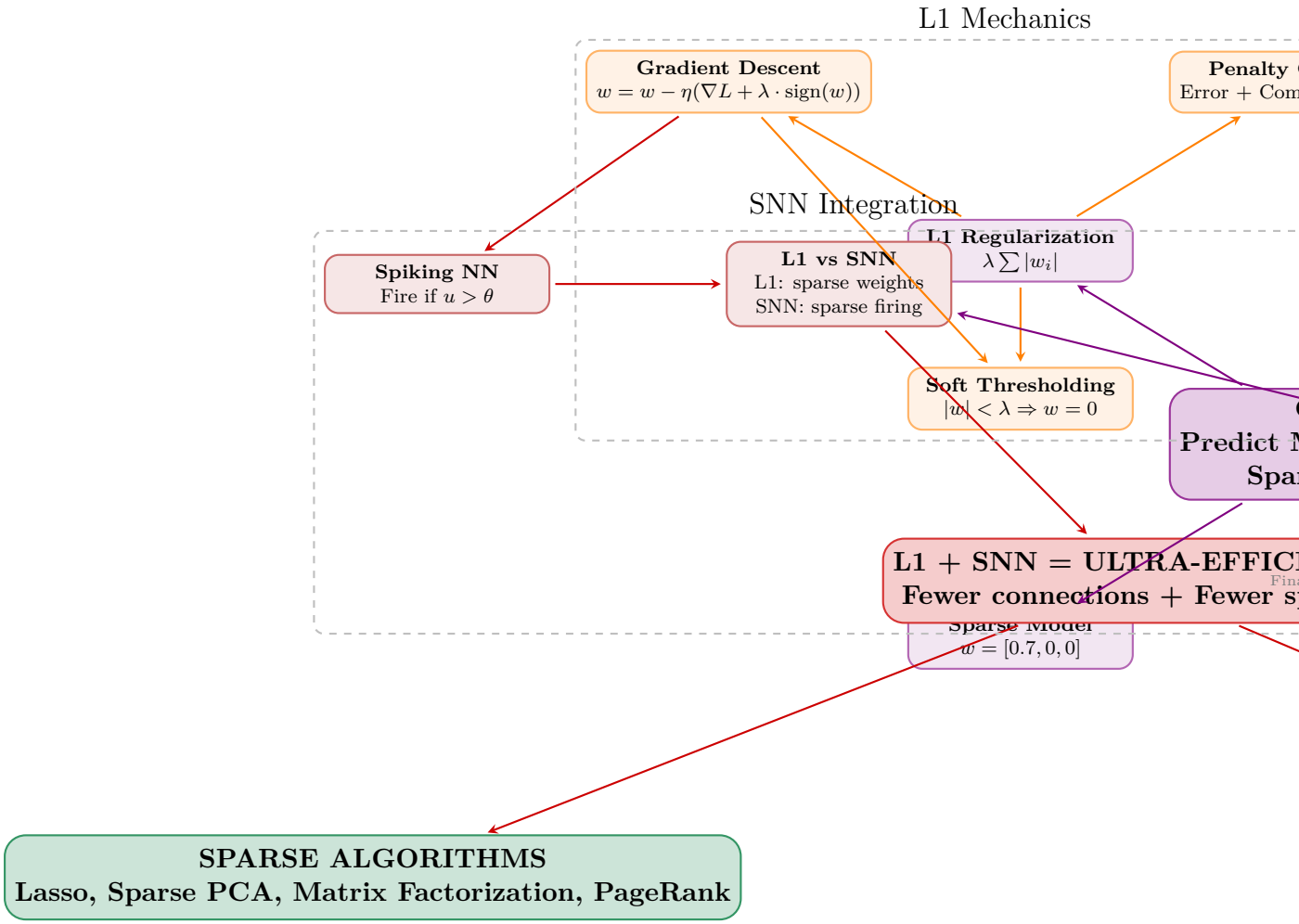


Figure 1: Complete Mind Map: City Network with L1 Regularization and Related Concepts