Complete Guide: From City Network to Neural Architectures

Comprehensive Pipeline Document with Mind Maps

March 28, 2026

Contents

1 City Network Problem: The Foundation **3**
- 1.1 Problem Statement . . . 3
- 1.2 Real-World Analogy: Logistics Network . . . 3
- 1.3 Data Encoding . . . 3

2 Complete Mind Map: All Models and Concepts **4**

3 Linear Models Pipeline **6**
- 3.1 Mathematical Core of Linear Models . . . 6
- 3.2 Linear Models Comparison Mind Map . . . 7
- 3.3 Numerical Example with City Data . . . 8

4 Neural Network Pipeline **9**
- 4.1 Architecture for City Network . . . 9
- 4.2 Neural Network Mathematical Structure . . . 9
- 4.3 Numerical Example Forward Pass . . . 9
  - 4.3.1 Weights . . . 9
  - 4.3.2 Forward Pass for City A . . . 9

5 Activation Functions Pipeline **10**
- 5.1 Softmax: Converting to Probabilities . . . 11

6 Attention and Transformer Pipeline **12**
- 6.1 Attention Mathematical Core . . . 12
- 6.2 Components of Attention . . . 12
- 6.3 Transformer Architecture Mind Map . . . 13
- 6.4 Attention Numerical Example . . . 14

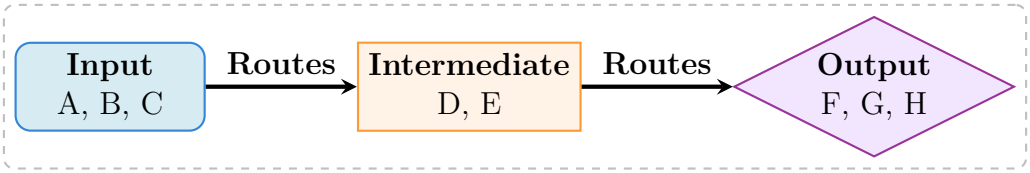
7 Tree-Based Models Pipeline **15**
- 7.1 Tree Models Mind Map . . . 16
- 7.2 Decision Tree Structure . . . 17
- 7.3 Random Forest Formula . . . 17

8 Optimization Pipeline **18**
8.1 Gradient Descent Core 18
8.2 Optimization Methods Mind Map 19
8.3 Comparison of Optimizers 20

9 Complete Model Comparison **21**

10 Key Insights Summary **22**
10.1 Unified View of All Models 22
10.2 The City Network Story 22
10.3 Final Mathematical Truth 22
10.4 One-Line Summary 23

1 City Network Problem: The Foundation



1.1 Problem Statement

- **Start Cities:** A, B, C (Top layer - Input)
- **Middle Hubs:** D, E (Hidden layer)
- **Destination Cities:** F, G, H (Bottom layer - Output)
- **Goal:** Learn optimal routing from A,B,C through D,E to F,G,H

1.2 Real-World Analogy: Logistics Network

- **A, B, C** = Warehouses
- **D, E** = Distribution Hubs
- **F, G, H** = Retail Stores
- **Routes** = Transportation paths

1.3 Data Encoding

City	Input Vector
A	[1, 0, 0]
B	[0, 1, 0]
C	[0, 0, 1]
D	[0, 0, 0, 1, 0]
E	[0, 0, 0, 0, 1]
F, G, H	Output targets

Table 1: One-hot encoding for cities

2 Complete Mind Map: All Models and Concepts

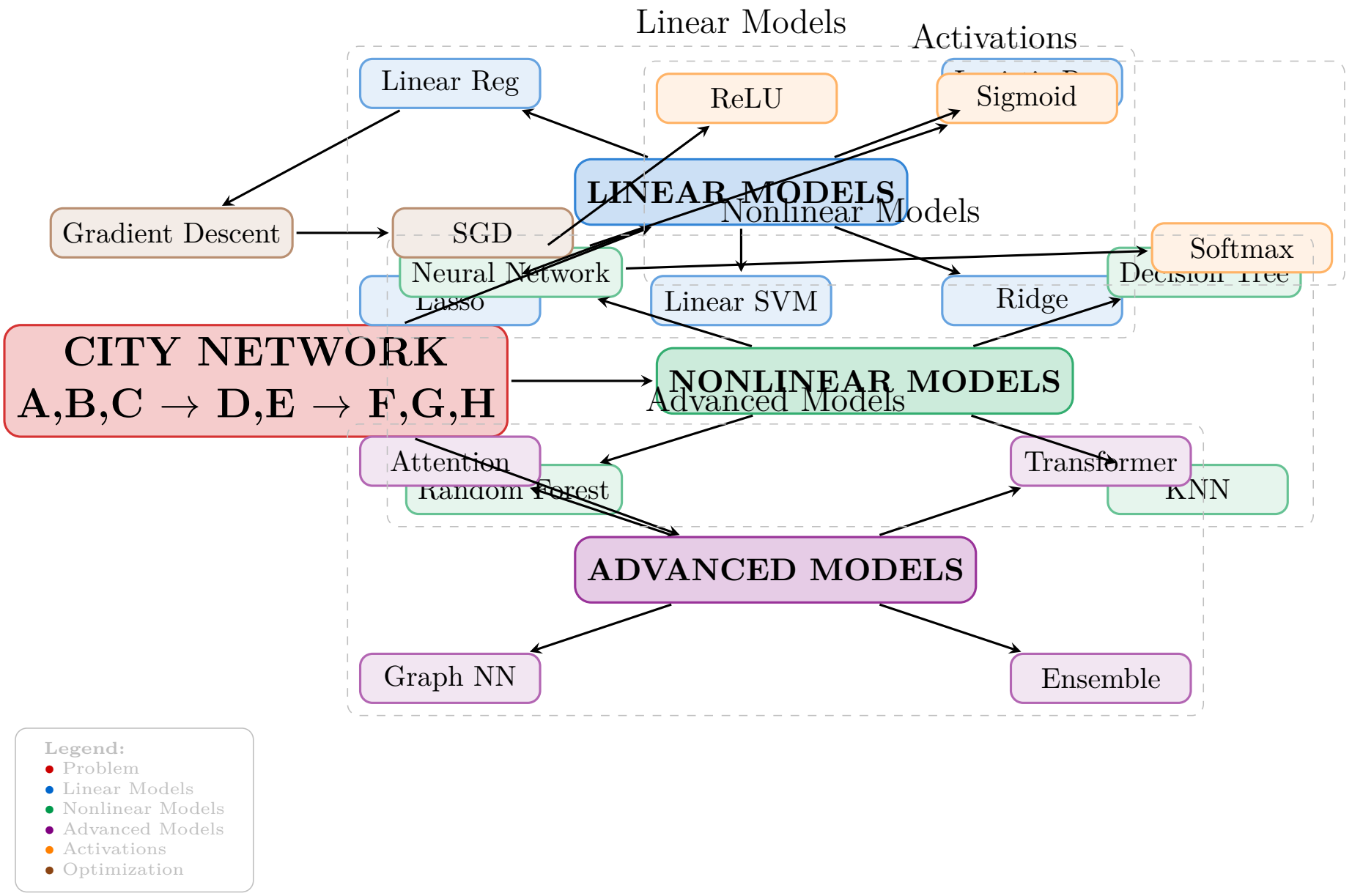


Figure 1: Complete Mind Map: All ML Models in City Network Context

3 Linear Models Pipeline

3.1 Mathematical Core of Linear Models

All linear models share the same foundation:

$$z = Wx + b = \sum_{i=1}^{n} w_i x_i + b \tag{1}$$

3.2 Linear Models Comparison Mind Map

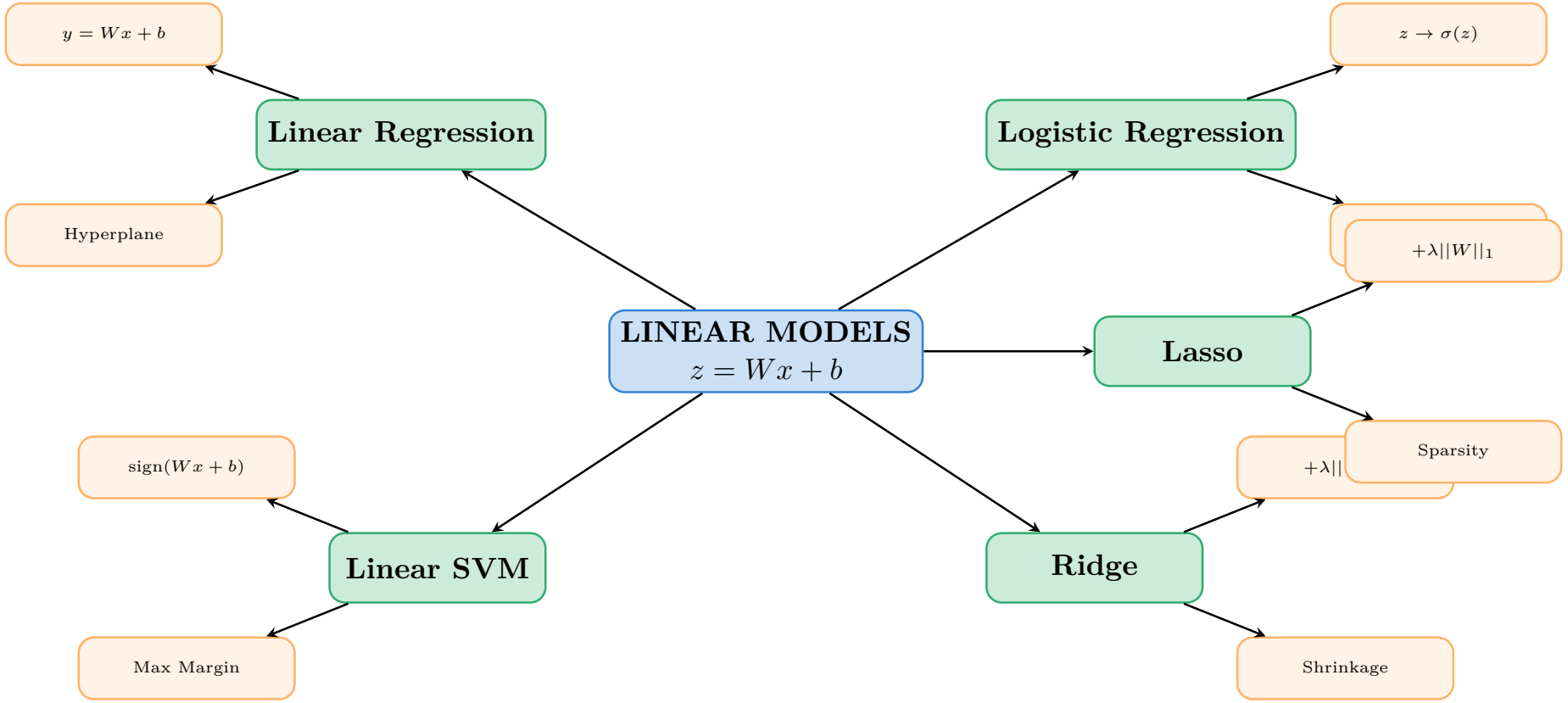


Figure 2: Linear Models Family Tree

3.3 Numerical Example with City Data

Using our city network with weights:

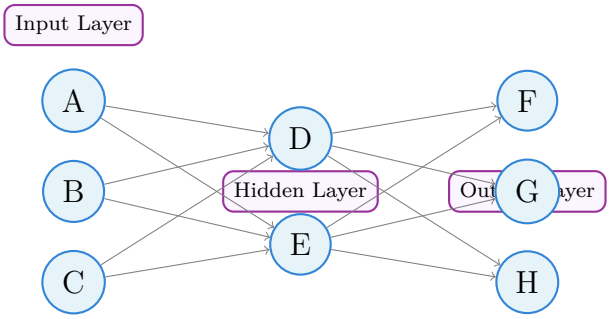
$$W = [1, 1], \quad b = -5 \tag{2}$$

City	Input	Linear Reg	Logistic	SVM
A	(1,1)	-3	0.047	0
B	(2,1)	-2	0.119	0
C	(2,2)	-1	0.269	0
F	(6,5)	6	0.997	1
G	(7,6)	8	0.9997	1
H	(8,6)	9	0.9999	1

Table 2: All linear models applied to city data

4 Neural Network Pipeline

4.1 Architecture for City Network



4.2 Neural Network Mathematical Structure

$$\text{Hidden Layer:} \quad h = \text{Activation}(W_1 x + b_1) \tag{3}$$

$$\text{Output Layer:} \quad y = \text{Activation}(W_2 h + b_2) \tag{4}$$

4.3 Numerical Example Forward Pass

4.3.1 Weights

Input → Hidden (W1):

$$W_1 = \begin{bmatrix} 0.9 & 0.2 \\ 0.3 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}, \quad b_1 = [0, 0] \tag{5}$$

Hidden → Output (W2):

$$W_2 = \begin{bmatrix} 0.9 & 0.2 & 0.3 \\ 0.1 & 0.8 & 0.7 \end{bmatrix}, \quad b_2 = [-0.5] \tag{6}$$

4.3.2 Forward Pass for City A

Input: $x = [1, 0, 0]$

Hidden Layer:

$$h_D = 1 \times 0.9 + 0 \times 0.3 + 0 \times 0.4 = 0.9 \tag{7}$$

$$h_E = 1 \times 0.2 + 0 \times 0.8 + 0 \times 0.6 = 0.2 \tag{8}$$

Output Layer:

$$y_F = 0.9 \times 0.9 + 0.2 \times 0.1 - 0.5 = 0.33 \tag{9}$$

$$y_G = 0.9 \times 0.2 + 0.2 \times 0.8 - 0.5 = -0.16 \tag{10}$$

$$y_H = 0.9 \times 0.3 + 0.2 \times 0.7 - 0.5 = -0.09 \tag{11}$$

After Softmax (probabilities):

$$P = [0.83, 0.34, 0.41] \quad \rightarrow \text{Highest} = \text{F} \tag{12}$$

5 Activation Functions Pipeline

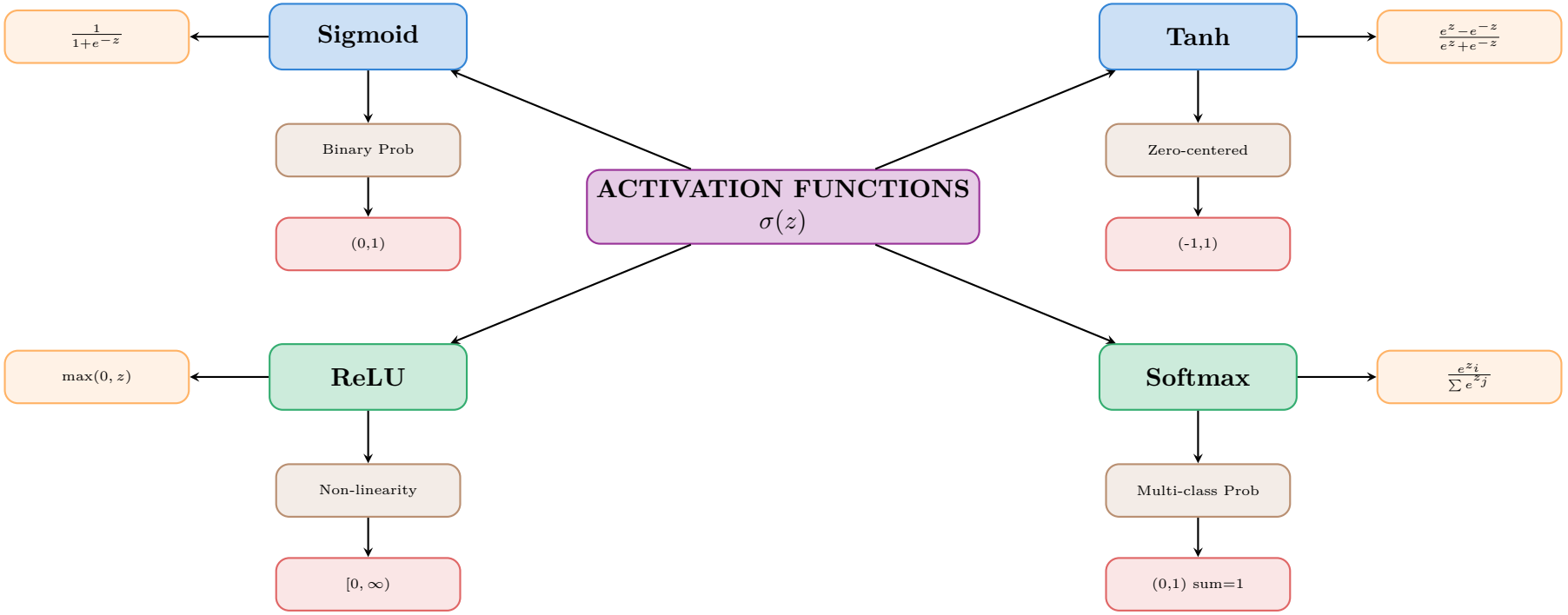


Figure 3: Activation Functions: Purpose and Properties

5.1 Softmax: Converting to Probabilities

Example: Attention scores $[2.0, 1.0, 0.1]$

Step 1: Exponentiate

$$e^{2.0} = 7.39 \tag{13}$$
$$e^{1.0} = 2.71 \tag{14}$$
$$e^{0.1} = 1.10 \tag{15}$$

Step 2: Sum $= 7.39 + 2.71 + 1.10 = 11.20$

Step 3: Normalize

$$p_1 = 7.39/11.20 = 0.66 \tag{16}$$
$$p_2 = 2.71/11.20 = 0.24 \tag{17}$$
$$p_3 = 1.10/11.20 = 0.10 \tag{18}$$

Result: $[0.66, 0.24, 0.10]$ (Probabilities sum to 1)

6 Attention and Transformer Pipeline

6.1 Attention Mathematical Core

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{19}$$

6.2 Components of Attention

- **Q (Query):** What is being looked for
- **K (Key):** What each token offers
- **V (Value):** Information content
- **Softmax:** Convert scores to probabilities
- **Weighted Sum:** Combine information

6.3 Transformer Architecture Mind Map

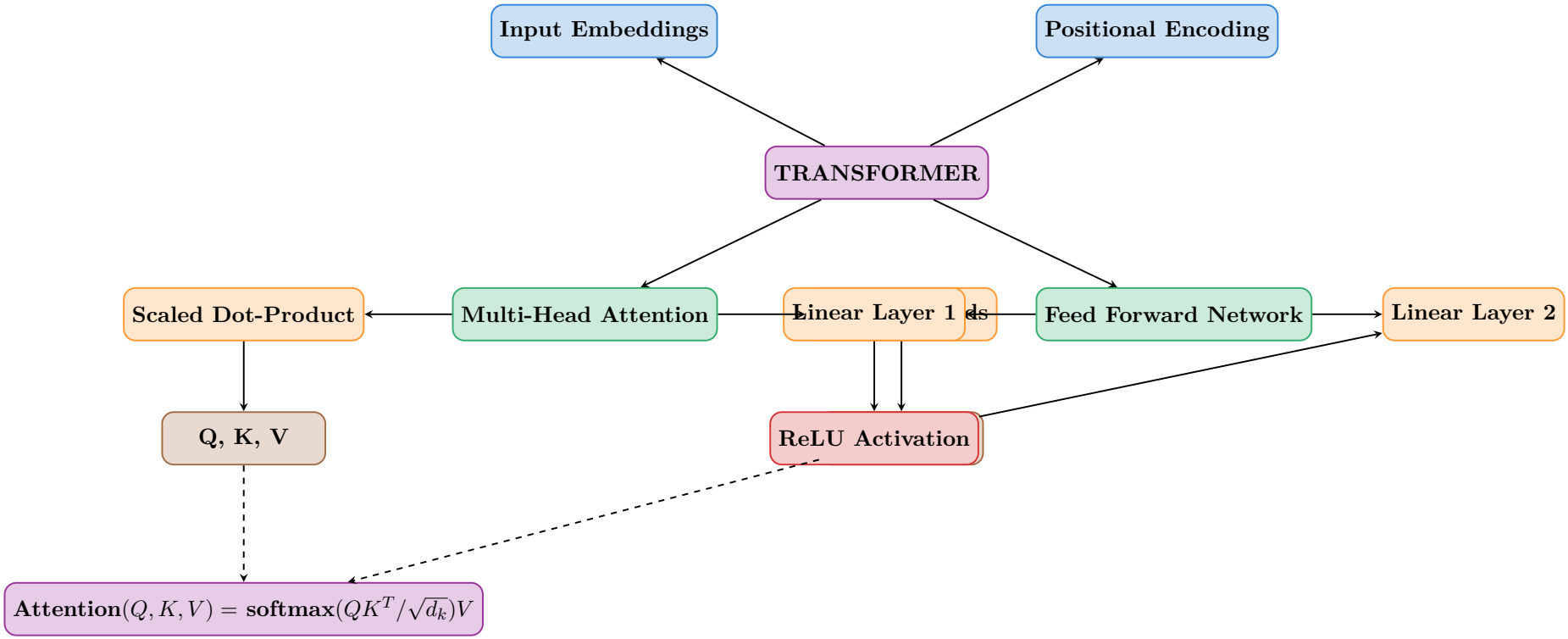


Figure 4: Transformer Architecture Breakdown

6.4 Attention Numerical Example

Value Vectors:

$$V_{\text{animal}} = [2, 3, 1] \tag{20}$$
$$V_{\text{street}} = [0, 1, 4] \tag{21}$$
$$V_{\text{tired}} = [5, 2, 0] \tag{22}$$

Attention Weights:

$$w_{\text{animal}} = 0.7 \tag{23}$$
$$w_{\text{street}} = 0.2 \tag{24}$$
$$w_{\text{tired}} = 0.1 \tag{25}$$

Weighted Sum:

$$\begin{aligned} \text{Output} &= 0.7 \times [2, 3, 1] + 0.2 \times [0, 1, 4] + 0.1 \times [5, 2, 0] && (26)\\ &= [1.4, 2.1, 0.7] + [0, 0.2, 0.8] + [0.5, 0.2, 0] && (27)\\ &= [1.9, 2.5, 1.5] && (28) \end{aligned}$$

7 Tree-Based Models Pipeline

7.1 Tree Models Mind Map

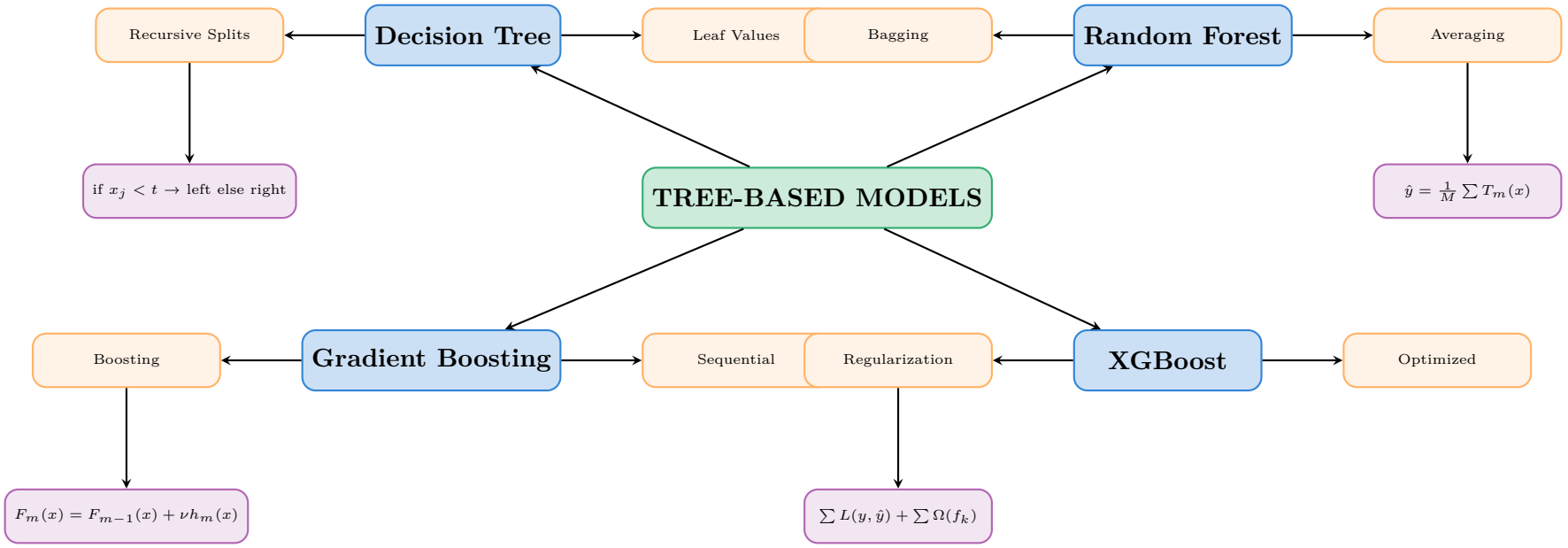


Figure 5: Tree-Based Models Family

7.2 Decision Tree Structure

$$\text{Tree}(x) = \sum_{m=1}^{M} c_m \cdot I(x \in R_m) \tag{29}$$

7.3 Random Forest Formula

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} T_m(x) \tag{30}$$

8 Optimization Pipeline

8.1 Gradient Descent Core

$$\theta_{new} = \theta_{old} - \eta \nabla L(\theta) \tag{31}$$

8.2 Optimization Methods Mind Map

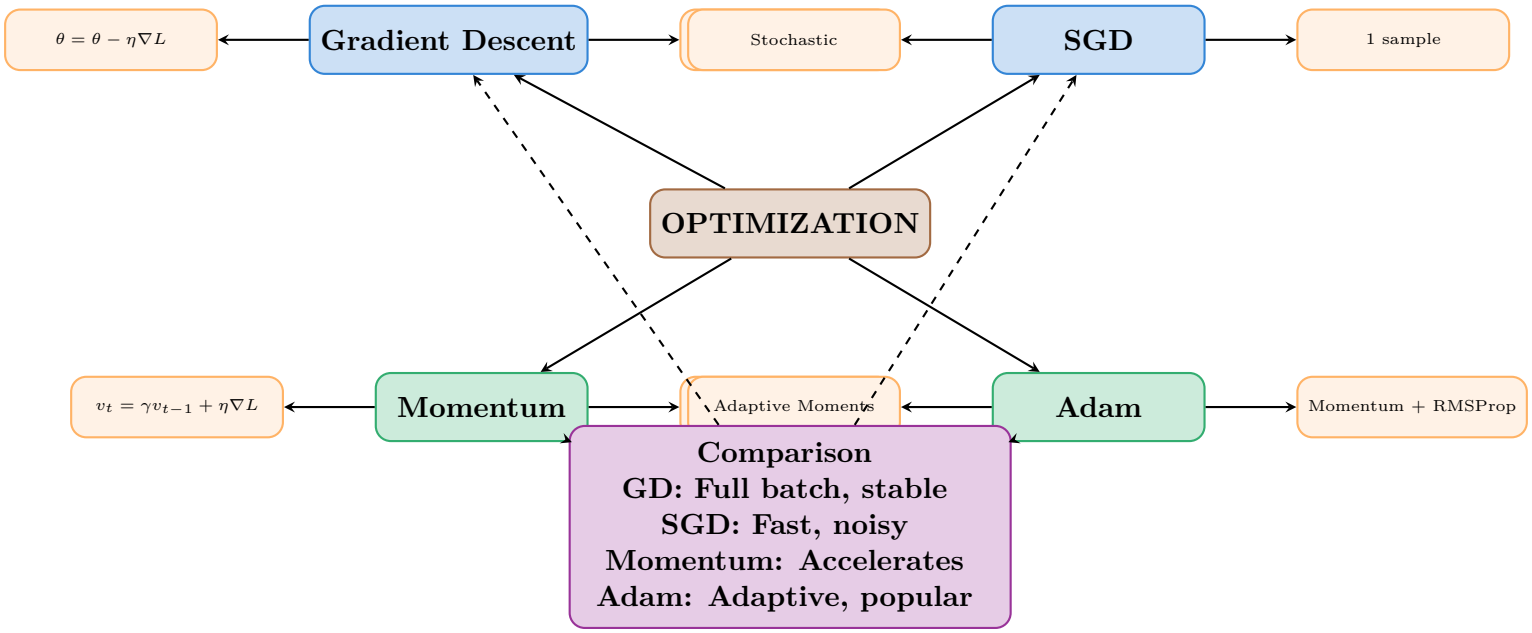


Figure 6: Optimization Algorithms Family

8.3 Comparison of Optimizers

Optimizer	**Update Rule**	**Pros/Cons**
GD	$\theta = \theta - \eta \nabla L$	Stable, but slow
SGD	$\theta = \theta - \eta \nabla L_i$	Fast, but noisy
Momentum	$v = \gamma v + \eta \nabla L$	Accelerates
Adam	Adaptive moments	Popular, adaptive

Table 3: Optimization algorithms comparison

9 Complete Model Comparison

Model	**Mathematical Core**	**City Network Interpretation**	**Key Property**
Linear Regression	$y = Wx + b$	One straight highway	Hyperplane
Logistic Regression	$y = \sigma(Wx + b)$	Probability gate at border	Binary probability
Linear SVM	$y = \text{sign}(Wx + b)$	Max margin separation	Margin optimization
Neural Network	$h = \sigma(W_1 x), y = \sigma(W_2 h)$	Multi-layer routing	Universal approximator
Decision Tree	if-else rules	Route decision points	Interpretable
Random Forest	$\frac{1}{M}\sum T_m(x)$	Multiple route experts	Robust
KNN	majority of k nearest	Local route voting	Instance-based
Attention	$\text{softmax}(QK^T)V$	Dynamic route weighting	Context-aware
Transformer	Attention + FFN	Complex route system	Sequential learning
Ridge	$+\lambda\|\|W\|\|_2^2$	Smooth weight shrinkage	Stability
Lasso	$+\lambda\|\|W\|\|_1$	Feature selection	Sparsity

Table 4: Complete model comparison with city network interpretation

10 Key Insights Summary

10.1 Unified View of All Models

Every ML model is either:

Category	Examples
Linear Transformations	Linear, Logistic, SVM, Ridge, Lasso
Nonlinear Transformations	Neural Networks, Attention
Space Partitioning	Decision Trees, Random Forest
Distance-Based	KNN, Similarity methods
Aggregation Methods	Ensemble, Boosting, Bagging
Optimization Methods	Gradient Descent, Adam

10.2 The City Network Story

- **Linear Models** → One straight highway
- **Neural Networks** → Complex road network with intersections
- **Decision Trees** → Sign posts with if-else rules
- **Random Forest** → Multiple route experts voting
- **KNN** → Ask nearby cities for direction
- **Attention** → Dynamically focus on important routes
- **Transformer** → Full traffic prediction system
- **Ridge/Lasso** → Penalize complex routes

10.3 Final Mathematical Truth

All models fundamentally solve:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \cdot \Omega(f) \tag{32}$$

Where:

- f = model function
- $\mathcal{F}$ = hypothesis space
- L = loss function
- Ω = regularization
- λ = penalty strength

10.4 One-Line Summary

All ML models are just mathematical transformations of the same city network problem: Find the best route from A,B,C through D,E to F,G,H