Mixture of Experts (MoE)

A Heterogeneous Data Modeling Approach

July 5, 2025

Introduction to Mixture of Experts

- Extension of traditional modeling techniques for heterogeneous data
- Combines multiple specialized models (experts) for different data segments
- Implements soft boundaries via probabilistic segment assignments
- Integrates clustering with predictive modeling
- Enhances robustness against distribution drift
- Provides interpretable predictions through functional ANOVA decomposition
- Functions as mixture of experts with specialized data region expertise

MoE Model Architecture

Core Concepts

- Overall prediction is a weighted combination of expert outputs
- Each expert specializes in different regions of the feature space
- Gating model determines segment memberships probabilistically
- Experts are trained with samples weighted by membership probabilities
- Model is fully interpretable through ANOVA-style decomposition

Key Innovations

- Combines segmentation and prediction in unified framework
- Allows for local expertise development within a global model
- Provides smooth transitions between expert regions
- Enhances model robustness against distribution shifts

Mathematical Foundation - Prediction Function

Overall Prediction Function

$$\hat{y}(x) = \sum_{k=1}^{K} \hat{p}_k(x) \cdot \hat{f}_k(x) \tag{1}$$

Components

- $x = (x_1, x_2, \ldots, x_d)$ - Feature vector
- $\hat{p}_k(x)$ - Membership probability from gating model for expert k
- $\hat{f}_k(x)$ - Prediction from expert model k
- K - Total number of experts/clusters

Ensemble Structure

- Weighted average of expert predictions
- Weights determined by cluster membership probabilities
- Each expert focuses on its specialized data region
- Graceful handling of boundary cases through soft assignments

Gating Model and Expert Networks

Gating Model Function

$$p_j(x) = \mu_j + \sum_i g_{ij}(x_i) + \sum_{ik} g_{ikj}(x_i, x_k) \tag{2}$$

Expert Model Function

$$f_j(x) = \mu_j + \sum_i f_{ij}(x_i) + \sum_{ik} f_{ikj}(x_i, x_k) \tag{3}$$

Components

- μ_j - Base membership/prediction for cluster j
- $g_{ij}(x_i)$ - Main effects for gating model
- $g_{ikj}(x_i, x_k)$ - Interaction effects for gating model
- $f_{ij}(x_i)$ - Main effects for expert model
- $f_{ikj}(x_i, x_k)$ - Interaction effects for expert model

Implementation Process

1. **Initial Clustering**
   - Define the number of clusters/experts (K)
   - Compute cluster centroids $\{c_k\}_{k=1}^{K}$ using k-means
   - Alternative: Use PSO for optimized centroid selection
2. **Gating Model Training**
   - Train gating XGBoost model (binary classification) $\hat{p}_k(x)$ for each cluster k
   - Use restricted tree depth (1-2) for interpretability
   - Employ probabilistic output for soft assignments
3. **Expert Model Training**
   - Train expert XGBoost model $\hat{f}_k(x)$ for each cluster k
   - Weight training samples by membership probabilities $\hat{p}_k(x)$
   - Focus experts on their relevant data regions

MoE in MoDeva Framework

```python
# For regression tasks
from modeva.models import MoMoERegressor
model_moe = MoMoERegressor(
    name="MOE_Regression",
    max_depth=2, n_clusters=2, n_estimators=100
)

# For classification tasks
from modeva.models import MoMoEClassifier
model_moe = MoMoEClassifier(
    name="MOE_Classification",
    max_depth=2, n_clusters=2, n_estimators=100
)
```

Key Parameters

- **n_clusters**: Number of experts (2-5 recommended)
- **max_depth**: Maximum tree depth (1-2 for interpretability)
- **n_estimators**: Trees per model (100+ recommended)

Training and Evaluation

```python
# Train model
model_moe.fit(ds.train_x, ds.train_y)

# Create TestSuite for evaluation
from modeva import TestSuite
ts = TestSuite(ds, model_moe)

# View model performance
result = ts.diagnose_accuracy_table()
result.table
```

TestSuite Features

- Comprehensive performance evaluation
- Multiple metrics for regression or classification
- Tools for model comparison and analysis
- Integrates with interpretation methods

Interpretability through Functional ANOVA

Decomposition of Prediction Function

The overall prediction function $f(x)$ is decomposed into additive components:

$$f(x) = f_0 + \sum_i f_i(x_i) + \sum_{i<j} f_{ij}(x_i, x_j) + \dots \tag{4}$$

- Baseline: Global mean prediction f_0
- Main Effects: For each feature x_j: $f_j(x_j)$
- Interaction Effects: For feature pairs (x_j, x_k): $f_{jk}(x_j, x_k)$

Global Model Understanding

Feature Importance Impact of features on predictions

```
result = ts.interpret_fi()
result.plot()
```

Importance Metrics

- Based on variance of marginal effects
- Normalized to sum to 1
- Higher values indicate stronger influence
- Accounts for feature scale differences

Effect Importance Contribution of ANOVA components

```
result = ts.interpret_ei()
result.plot()

# Main effect
result = ts.interpret_effects(
    features="hr")
result.plot()
```

Understanding Individual Predictions

Local Feature Importance Feature impact for specific samples

```
result = ts.interpret_local_fi(
    sample_index=10,
    centered=True)
result.plot()
```

Local Effect Importance Effect contribution for individuals

```
result = ts.interpret_local_ei(
    sample_index=10,
    centered=True)
result.plot()
```

Uncentered Analysis (centered=False):

- Raw feature contributions
- Direct interpretation
- May have identifiability issues

Centered Analysis (centered=True):

- Compares to population mean
- More stable interpretation
- Better for relative importance

Centroid Optimization via PSO

Particle Swarm Optimization

- Advanced technique for finding optimal cluster centroids
- Treats centroids as hyperparameters to be optimized
- Can significantly improve model performance
- Especially effective for complex data distributions

PSO Process

- **Particle Evaluation**: Retrain gating and expert models for candidate centroids
- **Particle Update**: Modify centroid positions based on performance
- **Selection**: Choose centroids with lowest validation loss
- **Iteration**: Repeat until convergence or maximum iterations

Implementing PSO for Centroid Optimization

```python
from modeva.models import ModelTunePSO, MoMoERegressor

# Define search space for centroids
n_clusters = 2
param_bounds = {
    'centroids': [
        np.repeat(ds.train_x.min(0).reshape(1, -1),
                  repeats=n_clusters, axis=0).ravel(),
        np.repeat(ds.train_x.max(0).reshape(1, -1),
                  repeats=n_clusters, axis=0).ravel()
    ]
}
# Run PSO for optimization
model = MoMoERegressor(n_clusters=n_clusters,
                       max_depth=1, n_estimators=100)
hpo = ModelTunePSO(dataset=ds, model=model)
hpo_result = hpo.run(param_bounds=param_bounds,
                     n_iter=10,
                     n_particles=5,
                     cv=3,
                     metric="MSE")
```

Advantages of the MoE Approach

Technical Benefits

- Adaptive learning of heterogeneous patterns
- Specialized experts for different data regions
- Enhanced predictive accuracy through local specialization
- Resilience against distribution shifts

Practical Advantages

- Clear data segmentation with actionable insights
- Deep understanding of segment-specific drivers
- Robust performance across varying data distributions
- Comprehensive interpretation capabilities

When to Use MoE Applications with heterogeneous data populations where different segments may be driven by different factors, especially when interpretability and robustness are important.

Key Takeaways

Mixture of Experts (MoE) Benefits

- Effectively handles heterogeneous data with multiple subpopulations
- Combines clustering and prediction in a unified framework
- Provides both local and global interpretability
- Enhances model robustness against distribution shifts
- Delivers actionable insights through segment-specific modeling

Implementation in MoDeva

- Simple API with comprehensive tools
- Flexible parameter configuration
- Advanced optimization options
- Rich interpretation capabilities
- Seamless integration with other modeling approaches