Variational Oblique Predictive Clustering Trees

Viktor Andonovikj

Jožef Stefan Institute & Jožef Stefan International Postgraduate School

September 26, 2024

Outline

- Key Idea
- Introduction to Structured Output Prediction
- Variational SPYCT
- Experimental Setting
- Data
- Results
- Interpretability
- Conclusion

Key Idea

- We focus on **decision tree** type of models - popular due to their interpretability and simplicity.
- We aim to combine the **predictive power** of **ensemble** methods with the **interpretability** of a **single** decision tree.
- **Variational SPYCT** incorporates Bayesian inference to enhance both predictive performance and decision-making transparency.

Introduction to Structured Output Prediction (SOP)

- **Structured Output Prediction (SOP)** involves predicting multiple interdependent outputs.
- SOP tasks include simultaneous prediction of:
  - Multiple continuous values.
  - Multiple discrete values.
  - Hierarchically organized discrete values.
- **Real-world applications**:
  - Drug discovery: predicting multiple biological properties of molecules.
  - Recommender systems: predicting user preferences across multiple items.
  - Genomics: predicting gene functions based on interdependent traits.

Predictive Clustering Framework

- **Predictive Clustering**: Combines clustering and prediction by treating prediction as a hierarchical clustering task where similar instances are grouped and predictions are made for each cluster.
- **Characteristics**:
  - Supports both supervised and semi-supervised learning.
  - State-of-the-art performance via ensemble learning.
  - Offers interpretable models through feature importance analysis.
  - Provides a unified framework for predictive modeling across multiple tasks.

Predictive Clustering Trees (PCT) vs Oblique Predictive Clustering Trees (SPYCT)

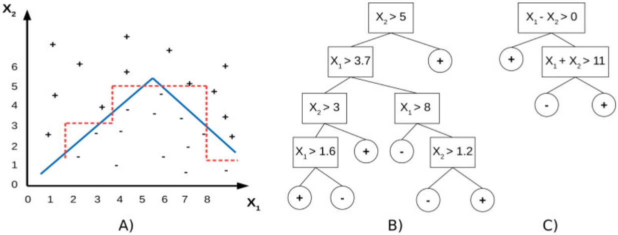

Figure: A) Learned split - SPYCT in blue, PCT in red B) PCT C) SPYCT [1]

- **PCT**: Uses axis-aligned splits (based on single features), fully interpretable models.
- **SPYCT**: Uses oblique splits (linear combinations of features), more flexibility for high-dimensional and sparse data, suited for more complex tasks with intricate decision boundaries.

Introduction to Variational SPYCT

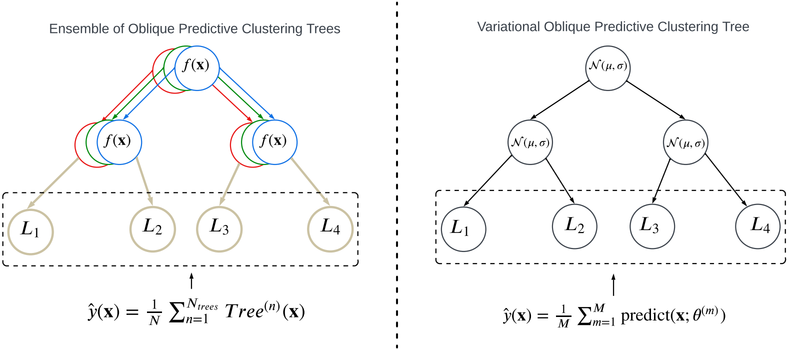

Figure: Ensemble of SPYCTs vs Variational SPYCT

Introduction to Variational SPYCT

- **Motivation**: Variational SPYCT (VSPYCT) integrates variational Bayes for improved decision-making within a single model, eliminating the need for ensembles.
- **Uncertainty quantification**: Embeds Bayesian inference directly into the decision tree structure, providing insight into decision processes and confidence levels.
- **Novelty**: VSPYCT introduces probabilistic treatment of oblique splits, offering a paradigm shift toward interpretable, and reliable machine learning models.

Optimization through Variational Bayes

- **Bayes' Theorem**:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

- The primary computational challenge lies in calculating the posterior $p(\theta|x)$, where the denominator $p(x)$ requires:

$$p(x) = \int_\theta p(x|\theta)p(\theta)\, d\theta$$

- **Variational Bayes (VB)** approximates the posterior with $q_{\omega^*}(\theta)$, minimizing the KL divergence:

$$\omega^* = \arg\min_{\omega\in\Omega} KL(q_\omega(\theta) \parallel p(\theta|x))$$

- The Evidence Lower Bound (ELBO) is maximized to improve the approximation:

$$KL(q_\omega(\theta) \parallel p(\theta|x)) = -\mathbb{E}_q[\log p(x,\theta) - \log q_\omega(\theta)] + \log p(x)$$

- VB's computational efficiency surpasses methods like MCMC, though it introduces bias based on the choice of the variational family $\mathcal{Q}$.

Optimization through Variational Bayes

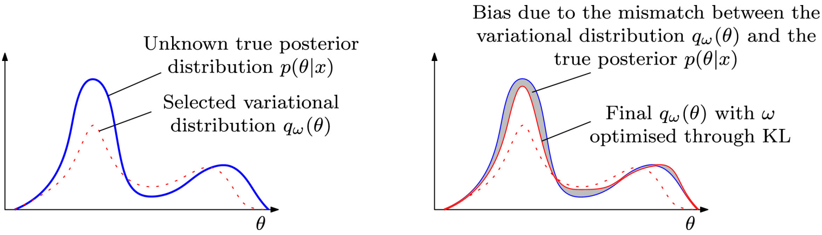

Figure: Optimisation process of finding the closest variational distribution $q_\omega(\theta)$ over the set of latent variables ω.

Methodology Overview

- VSPYCT follows SPYCT's tree-based architecture, but replaces fixed parameters with random variables.
- **Key difference**: VSPYCT uses variational Bayes (VB) for probabilistic split optimization, improving handling of noisy data and uncertainty.
- Split parameters (weights $\mathbf{w}$ and bias b) are modeled as random variables, allowing uncertainty estimation.

$$f(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

Learning Splits through Variational Bayes

- Weights **w** and bias b have Gaussian priors:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad b \sim \mathcal{N}(0, 1)$$

- Variational Bayes approximates the posterior distribution with:

$$q(\mathbf{w}, b|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)\mathcal{N}(b|\mu_b, \sigma_b^2)$$

- ELBO maximization:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}_{q(\mathbf{w},b|\mathcal{D})}[\log p(\mathcal{D}|\mathbf{w}, b)] - KL(q(\mathbf{w}, b|\mathcal{D}) \parallel p(\mathbf{w}, b))$$

Algorithm for Learning a Split

Algorithm 1 Variational Learning of Split Parameters

Input: $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, $\theta = \{\mathbf{w}_0, b_0\}$, E (epochs), λ (learning rate), β (batch size), σ (selection probability)

Output: $\Theta = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mu_b, \sigma_b^2\}$ (variational parameters)

procedure $\textsc{LearnSplit}(\mathcal{D}, \theta, E, \lambda, \beta, \sigma)$
- Initialize $\Theta = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mu_b, \sigma_b^2\}$
- **for** $e \in \{1, \ldots, E\}$ **do**
  - **for** each mini-batch $\mathcal{B} \subseteq \mathcal{D}$ of size β **do**
    - Sample $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, $b \sim \mathcal{N}(\mu_b, \sigma_b^2)$
    - Compute impurity $\Omega(\mathbf{X}, \mathbf{Y}; \mathbf{w}, b)$
    - Compute ELBO $\mathcal{L}(\mathcal{B}; \Theta)$
    - Update Θ
  - **end for**
- **end for**
- **return** Θ

end procedure

Deriving the Impurity Function

- For a given split node i, we define the fuzzy membership:

$$FM_i = \sigma(\mathbf{x}^\top \mathbf{w_i} + b_i), \quad 1 - FM_i = \text{right group membership}$$

- The split fitness function is the following:

$$f(\mathbf{w_i}, b_i) = Z \cdot imp(FM) + (L + U - Z) \cdot imp(1 - FM)$$

- Impurity is given by the variances of the target Y and features X:

$$imp(FM) = \sum_{k=1}^{N} \sigma^2_{\bar{X}_k} + \sum_{k=1}^{T} \sigma^2_{\bar{Y}_k}$$

- In VSPYCT, we minimize this impurity by observing a target impurity of *impurity*/2 at each step using Variational Bayes.

Making a Prediction

- Prediction involves traversing the tree, making probabilistic decisions at each node.
- The split is determined by evaluating the function:

$$f(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

- The final prediction $\hat{y}$ is the prototype value at the reached leaf:

$$\hat{y} = \frac{1}{|\mathbf{Y}|} \sum_{i=1}^{|\mathbf{Y}|} y_i$$

Prediction Process in VSPYCT

Algorithm 2 Prediction using Monte Carlo Sampling

Input: Feature vector $\mathbf{x}$, tree T, samples M
Output: Prediction $\hat{y}$
Initialize $\hat{y}_{\text{sum}} = 0$
for $m = 1$ to M **do**
    Traverse T from root to leaf with sampled $\mathbf{w}^{(m)}, b^{(m)}$
    $\hat{y}_{\text{sum}} \leftarrow \hat{y}_{\text{sum}} +$ prediction at leaf
end for
return $\hat{y} = \frac{\hat{y}_{\text{sum}}}{M}$

Feature Importance

- Feature importance in VSPYCT: Calculated as the influence of features across all splits.
- The importance of a feature is determined by:

$$\mathrm{Imp}(T) = \sum_{s \in T} \left(\frac{s_n}{N} \right) \left(\frac{\mathbb{E}[\mathbf{w}_s]}{|\mathbb{E}[\mathbf{w}_s]|_1} \right)$$

- Weights $\mathbf{w}$ are sampled from the posterior distribution, and importance is aggregated over all splits.

Time Complexity Analysis

- Time complexity of learning a split in VSPYCT:

$$\mathcal{O}(MNI_{vb}(D+K))$$

- M: Number of Monte Carlo samples, N: Data points, D: Features, K: Clustering attributes, I_{VB}: Number of optimization iterations.

Experimental Setting

- **Comparison**: VSPYCT vs SPYCT (single tree and ensemble).
- **Tasks**: Single-target and multi-target regression, binary and multi-class classification.
- **Goal**: Evaluate predictive performance of VSPYCT across different modeling scenarios.

Data

- Datasets cover regression and classification tasks.
- Number of examples (N), features (D), targets (T), and classes (C).

Table: Summary of the datasets used in the experiments.

Dataset	**Type of Task**	**N**	**D**	**T**	**C**
rf1 [2]	Multi-target regression	9125	64	8	–
rf2 [2]	Multi-target regression	9125	576	8	–
atp1d [2]	Multi-target regression	337	411	6	–
atp7d [2]	Multi-target regression	296	411	6	–
scm1d [2]	Multi-target regression	9803	280	16	–
house_8L [3]	Single-target regression	22784	8	1	–
puma8NH [3]	Single-target regression	8192	8	1	–
diabetes [3]	Binary classification	768	8	1	2
banknote [3]	Binary classification	1372	4	1	2
gas-drift [3]	Multi-class classification	13910	128	1	6
balance [3]	Multi-class classification	625	4	1	3

Evaluation Metrics

- **Regression**: Mean Absolute Error (MAE).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

- **Classification**: F1 score (macro-averaged for multi-class tasks).
- **Cross-validation**: 5-fold cross-validation used to reduce bias.

Regression Results

Table: *MAE* Scores for Regression Datasets

Dataset	SPYCT-single tree	SPYCT-ensemble	VSPYCT
rf1	29.38	**29.34**	29.37
rf2	29.37	29.49	**29.36**
atp1d	95.17	**70.93**	77.24
atp7d	115.54	**73.08**	91.28
scm1d	205.99	206.01	**205.72**
house_8L	25378.02	**19817.56**	22992.24
puma8NH	2.80	**2.62**	2.74

- VSPYCT outperforms or closely matches the SPYCT ensemble on several datasets (rf2, scm1d).
- Ensemble methods generally achieve lower MAE, but VSPYCT provides competitive results.

Classification Results

Table: F1 Scores for Classification Datasets

Dataset	SPYCT-single tree	SPYCT-ensemble	VSPYCT
diabetes	0.53	**0.60**	0.59
banknote	0.97	**0.99**	0.98
gas-drift	0.98	0.99	**0.99**
balance	0.60	0.66	**0.73**

- VSPYCT achieves competitive performance across datasets, often matching or surpassing ensembles.
- In the balance dataset, VSPYCT achieves the best F1 score.

Interpretability - An example with real dataset

- **Dataset:** Unemployment dataset from the Slovenian Public Employment Service (PES), consisting of 74,086 anonymized instances.
- **Features:** Age, gender, education, work experience, and other personal/professional characteristics.
- **Target:** Time until the jobseeker becomes employed or exits the study (measured in days). Some records are right-censored (event not observed).
- **Goal:** Predict the time-to-event (employment) with censored data, using a semi-supervised learning approach for handling the inherent missing information.
- **Challenges:** Diverse attribute types (categorical, numerical, temporal) and the presence of censored data.

Structure of the learned VSPYCT model

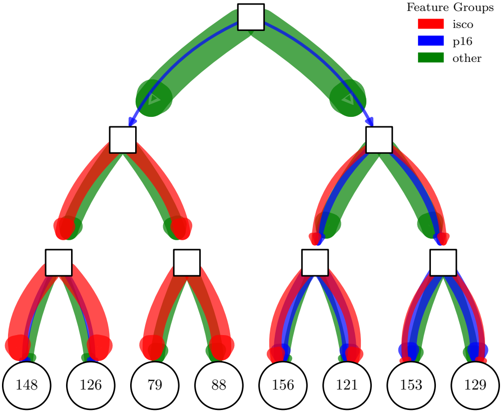

Figure: VSPYCT tree structure. Colors represent different feature groups (e.g., education, profession).

Predicted Survival Curve

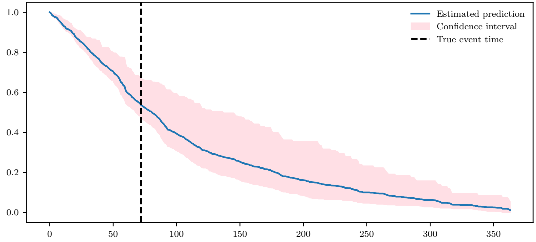

Figure: Predicted survival curve for a jobseeker. Blue: mean prediction; shaded area: confidence interval; vertical line: true event time.

- The confidence interval captures the uncertainty behind the predictions.

Feature Importance

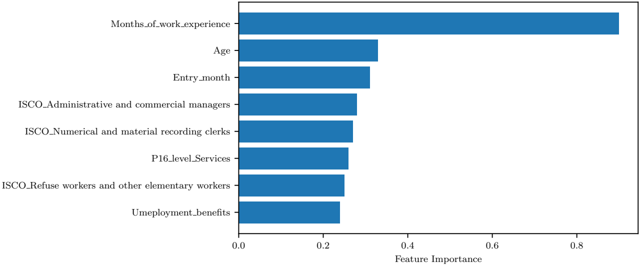

Figure: Feature importance from the VSPYCT model. "Months of work experience" is the most important factor in the model's decisions.

Conclusion

- **VSPYCT Model:** Combines oblique splits, variational Bayes, and Bayesian inference in an interpretable predictive framework.
- **Key Characteristic:** Balances the performance of ensemble models with the interpretability of single decision trees, while introducing uncertainty quantification.
- **Results:** Competitive performance, sometimes surpassing ensembles of SPYCT.
- **Uncertainty Quantification:** Enhances decision-making reliability, making the model applicable to domains where certainty in predictions is critical.
- **Future Work:** Further performance improvements and extending applications to diverse predictive tasks.

References I

[1] T. Stepišnik and D. Kocev, "Oblique predictive clustering trees," *Knowledge-Based Systems*, vol. 227, p. 107 228, Sep. 2021, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2021.107228. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2021.107228.

[2] Mulan, *Mulan: A java library for multi-label learning*, http://mulan.sourceforge.net/datasets.html, Accessed: 2020-04-15.

[3] OpenML, *Openml*, https://www.openml.org, Accessed: 2020-04-15.

Thank You

Questions?