UNIVERSITY OF CAMBRIDGE
Department of Computer Science and Technology

Ain't Nobody Got Time For That: Budget-aware Concept Intervention Policies

Thomas Yuan

Downing College

May 2024

Submitted in partial fulfillment of the requirements for the
Computer Science Tripos, Part III

Total page count: 21

Main chapters (excluding front-matter, references and appendix): 12 pages (pp 8–19)

Main chapters word count: 467

Methodology used to generate that word count:

```
$ make wordcount
gs -q -dSAFER -sDEVICE=txtwrite -o - \
   -dFirstPage=6 -dLastPage=11 report-submission.pdf | \
egrep '[A-Za-z]{3}' | wc -w
467
```

Declaration

I, Thomas Yuan of Downing College, being a candidate for the Computer Science Tripos, Part III, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed: Thomas Yuan

Date: May 10, 2024 -

Abstract

Regular supervised learning Machine Learning models learn to predict the labels of inputs. Concept Bottleneck Models (CBMs) are ML models designed to increase the interpretability of model predictions by decomposing a model into two submodels, splitting the original process into predicting a set of human-interpretable concepts / features present in the input, then predicting the label using these concepts. Since these concepts are human-interpretable, it is much more easier to understand the reasoning behind the predicted labels, thus mitigating some of the potential dangerous downsides associated with using ML models as "black-box" models, especially in fields where these predictions can have significant consequences to human life, such as medicine, criminal justice, autonomous vehicles, etc.

During inference time, professionals can intervene on CBMs by correcting the predicted concepts leading to more accurate predicted labels. Due to the costs associated with performing such an intervention, the question of what concepts to intervene on in order to maximize the accuracy of the model becomes an important research question. This project focuses on answering this research question. This project attempts to model the costs associated with using experts to perform interventions as a budget, and thus the main research question of this project is "How can we determine the concepts to intervene on for a given budget for a set of inputs and the corresponding model predictions?"

This project focuses on answering the above research questions. It first investigates the differences between greedy and non-greedy models, showing that non-greedy algorithms can outperform its greedy counterparts. It then investigates the performance of greedy models, building on top of existing methods by incorporating surrogate models to model the distribution of concepts. The output of these surrogate models are then used by an ML model that learns to predict the next concept to intervene on in each step. The project then investigates using Reinforcement Learning and these surrogate models to train a non-greedy model that learns to predict an entire sequence of interventions for given inputs and corresponding CBM outputs. Lastly, the project then investigates if it is necessary to train the prediction model simultaneously with the CBMs.

Acknowledgements

This project would not have been possible without the wonderful support of my lovely supervisors Mateo Espinosa Zarlenga, Dr Mateja Jamnik and Dr. Zohreh Shams. I would also like to thank my friends and family for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Concept Bottleneck Models

Machine Learning (ML) models are universal approximation solutions to problems and have been traditionally viewed and used as ”black-box” solutions, where users simply query the model and receive an answer without knowing the reasoning process behind it. Over the past decade, the increased application of ML models along with this ”black-box” property has raised many concerns, especially in areas where decisions made are critical to human safety, such as in medicine or automated driving. To increase interpretability, researchers developed Concept Bottleneck Models (CBMs) [3] that learns to predict a set of human-interpretable concepts from input, and then uses these concepts to predict labels. This increases the interpretability of models as humans can understand the basis of the ML model predictions via the high-level intermediate concepts that the model is trained to predict.

Another additional benefit of these models is that when used in practice, experts can intervene by correcting incorrect intermediate concept predictions to generate more accurate final label predictions. Given that experts have limited time, determining the order of concepts to query experts to intervene on, to maximize the accuracy of the model, becomes an important problem. Since the objective behind developing these models is to use them in real life, this project utilizes different datasets that reflect properties of problems encountered in real life to measure the performance. This is further discussed in Section **??**.

1.2 CEM and IntCEM

Current research has made significant progress on optimizing these models for interventions, including numerous studies on developing models to learn the optimal order of concepts to intervene on [2], most notably Intervention-aware Concept Embedding Models (IntCEMs) [8]. IntCEMs build on Concept Embedding Models [7], a variant of CBMs that utilize embeddings to represent the intermediate concepts. IntCEMs augment CEMs with an additional model that learns to predict the next concept to intervene on given the current state of the CBM, which is also used during training to increase the CBM's sensitivity to interventions. IntCEMs achieve state-of-the-art performance on the performance of interventions while still maintaining similar performance when no interventions are performed.

1.3 Reinforcement Learning

Despite the above-mentioned improvements, there is still a big gap between the performance of these intervention policies versus the best possible performance, i.e. the performance of intervention policies that have access to the ground truth output labels. Additionally, existing approaches such as IntCEM are greedy, which means that they learn to predict concepts that maximize the performance at each step. We speculate that non-greedy methods may outperform these existing greedy methods, with the objective being maximizing performance after intervening a certain number of concepts rather than maximizing performance at each step. One such approach that may be used to generate non-greedy intervention policies is Reinforcement Learning (RL) [6].

This project focuses specifically on trying to solve the question of finding a good intervention policy for a given budget using RL, and Surrogate models to model conditional probabilities to guide the RL model, taking inspiration from an approach [4] in a similar setting of Active Feature Acquisition (AFA) [5].

This project successfully develops a novel RL-based method that when used in conjunction with existing methods from IntCEM to increase sensitivity to interventions, is able to learn a non-greedy intervention policy for a Concept Bottleneck Model that outperforms existing greedy intervention policies for different budgets, while maintaining similar performance when no interventions are performed. To our knowledge, this is the first approach that utilises Reinforcement Learning to learn an intervention policy, building on top of the joint optimization approach from IntCEM to learn better intervention policies that offers notably better intervention performances throughout the intervention trajectory.

Chapter 2

Background

2.1 CBM

Concept Bottleneck Models (CBMs), initially proposed by Koh et al. [3], are a class of models that consist of a model g that learns a mapping from input $\mathbf{x}$ to a concept vector $\mathbf{c} = g(\mathbf{x})$, where $\mathbf{c}$ is a multi-hot encoding of the concepts present in the input, and a model f that learns a mapping from such concepts vector c to the output label $\mathbf{y} = f(\mathbf{c})$, as shown in Figure 2.1. These types of model can be created by simply adding a new layer in traditional models with the same number of neurons as the number of concepts, where this layer is referred to as the "CBM Bottleneck". Henceforth g is referred to as the "concept predictor $\mathbf{x} \rightarrow \mathbf{c}$ model" and f as the "label $\mathbf{c} \rightarrow \mathbf{y}$ predictor model". Training such a model requires a dataset of inputs $\mathbf{x}$ annotated with the corresponding concepts $\mathbf{c}$ and labels $\mathbf{y}$.

2.1.1 Intervention Policy

Another key advantage of using CBMs is having access to run-time interventions, which is the idea of utilizing professionals to modify incorrect concept predictions to improve the performance of the model, as illustrated in Figure 2.2 For simplicity, we omit the possibility of incorrect interventions, i.e. the scenario where the professional misjudges and changes the concept prediction to be incorrect, and assume that all interventions modify the concepts correctly. Thus an intervention can be defined as the following function, where the predicted concepts $\hat{\mathbf{c}}$ and the true concepts $\mathbf{c}$ are interpolated via the intervention vector $\boldsymbol{\mu}$, a multi-hot encoded vector.

$$I(\hat{\mathbf{c}}, \mathbf{c}, \boldsymbol{\mu}) = \boldsymbol{\mu}\, \mathbf{c} + (1 - \boldsymbol{\mu})\, \hat{\mathbf{c}} \qquad \hat{\mathbf{c}}, \mathbf{c}, \boldsymbol{\mu} \in \{0, 1\}^k$$

To formalize an intervention policy for this project, we define an intervention policy $\mathcal{P}$ to be a policy, either learnt or heuristic-based, that determines the order of concepts

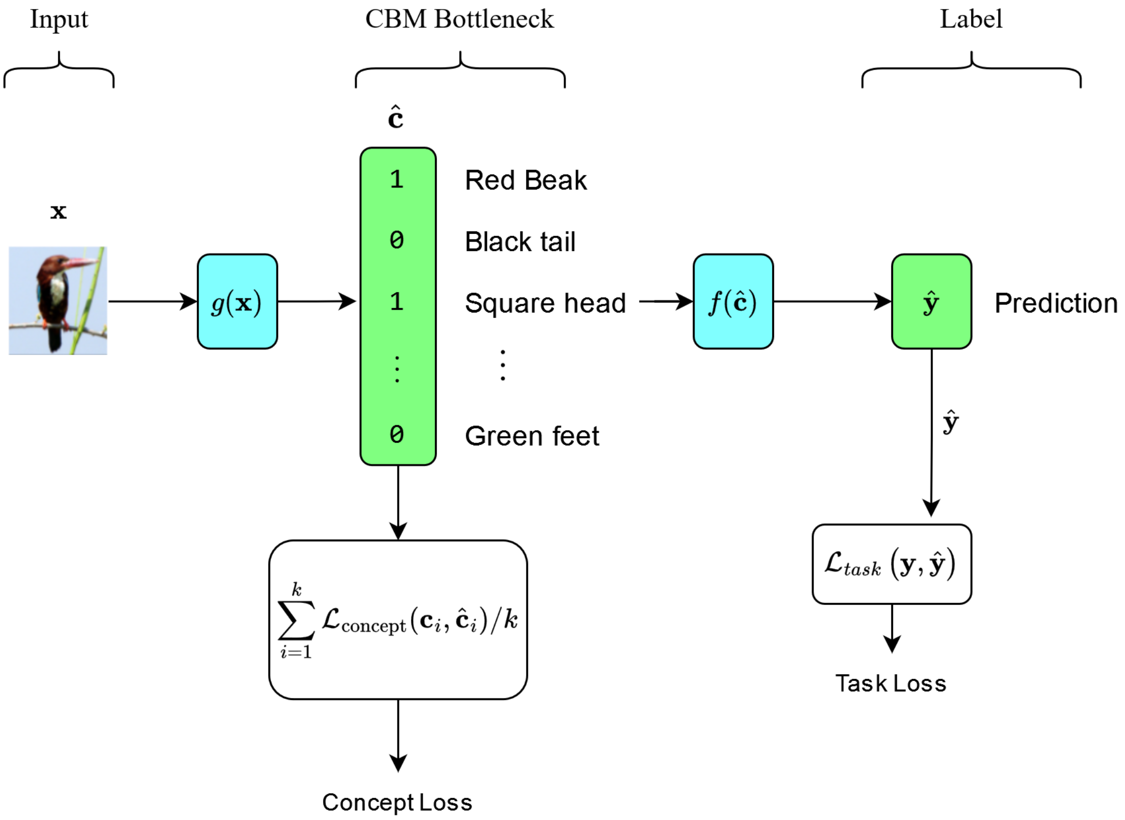

Figure 2.1: The CBM Architecture [3].

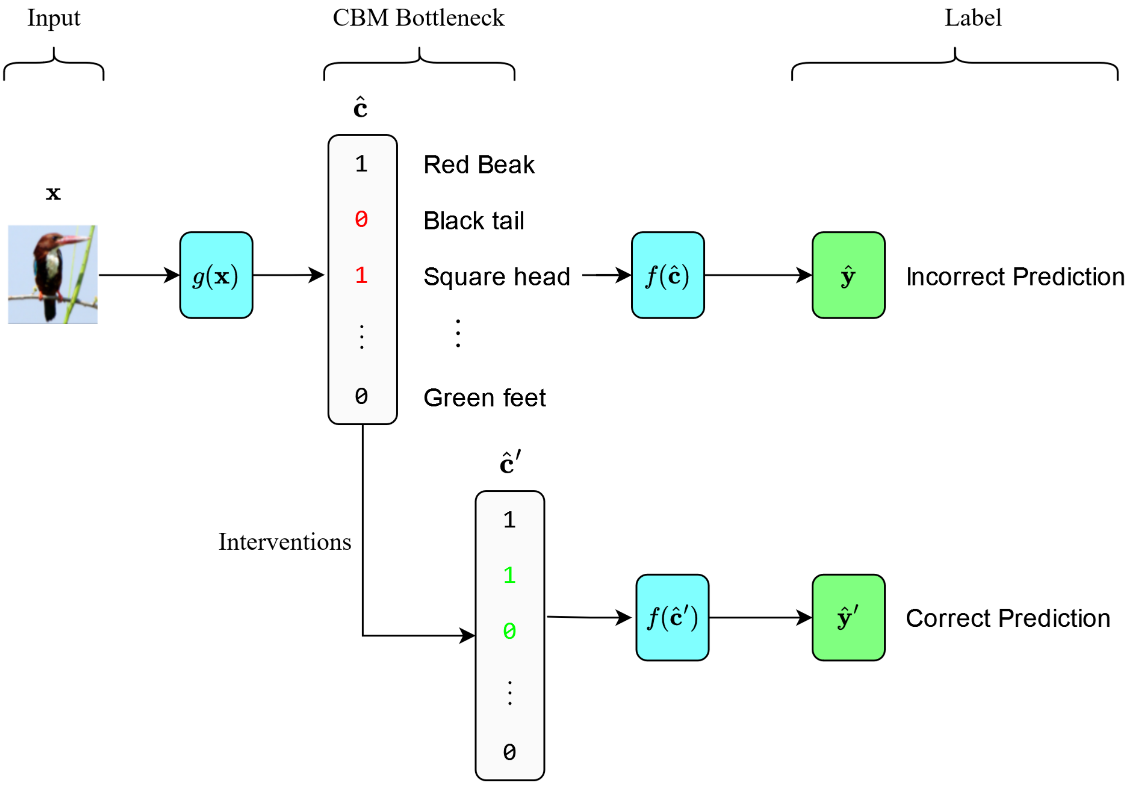

Figure 2.2: An illustration of intervening on the concepts predicted by a CBM.

to intervene on with the goal of maximizing the performance after interventions. If the performance of a model is measured by minimizing some loss L_{task}, a greedy intervention policy is a collection of policies $\mathcal{P}_i$, each outputting an index to intervene on at step i. The policy aims to minimizes the following:

$$\hat{\mathcal{P}} = \bigcup_{i=1}^{k} \underset{\mathcal{P}_i}{\text{argmin}}\, L_{\text{task}}(\hat{g}(\hat{\mathbf{c}}_{\mathcal{P}_j}), \mathbf{y})$$

$$\hat{\mathbf{c}}_{\mathcal{P}_0} = \hat{\mathbf{c}}, \hat{\mathbf{c}}_{\mathcal{P}_j} = I(\hat{\mathbf{c}}_{\mathcal{P}_{j-1}}, \mathbf{c}, \mathcal{P}_j(\hat{\mathbf{c}}_{\mathcal{P}_{j-1}}))$$

Which minimizes the loss at each step j sequentially for all k concepts. At each step, $\hat{\mathbf{c}}_{j-1}$ is the concept after the previous $j-1$ interventions, Similar to above, the task loss L_{task} is used to minimize the discrepancy of $\hat{g}(\hat{\mathbf{c}}_{\mathcal{P}_j})$, the output of the label predictor model on the intervened concepts, and label $\mathbf{y}$.

Compared to a greedy intervention policy, a non-greedy intervention policy outputs a set of concept to intervene on for a given budget j, which we want to maximize the performance of the label predictor model on. This is equivalent to maximizing the following function:

$$\hat{\mathcal{P}} = \underset{\mathcal{P}}{\text{argmin}} \sum_{j=1}^{k} L_{\text{task}}(\hat{g}(\hat{\mathbf{c}}_{\mathcal{P}_j}), \mathbf{y})$$

$$\hat{\mathbf{c}}_{\mathcal{P}_j} = I(\hat{\mathbf{c}}, \mathbf{c}, \mathcal{P}(\hat{\mathbf{c}}, j))$$

Note that the notion of a budget, defined as the number of concepts allowed to intervene on for simplicity, is only important to non-greedy policies. While non-greedy policies may have different concepts to intervene on for different budgets to maximize performance, greedy policies always select the same concepts and thus the current or remaining budget should not affect the concept selected by the policy.

2.1.2 Training CBMs

There are several different ways to train a CBM. If we let the concept loss L_{concept} be a loss function that measures the discrepancy between the predicted concepts $\hat{\mathbf{c}}$ and the actual concepts $\mathbf{c}$, and similarly the label loss L_{label} measuring the discrepancy between the predicted concepts $\hat{\mathbf{y}}$ and the actual concept $\mathbf{y}$, both losses as illustrated in Figure 2.1. There are the following ways to train a CBM as proposed in [3].

1. Independent: Training the two models independently by minimizing $L_{\text{concept}}(g(\mathbf{x}), \mathbf{c})$ and $L_{\text{label}}(f(\mathbf{c}), \mathbf{y})$ independently.

2. Sequential: Training the models one by one, first learning $\hat{g}$ by minimizing

   $L_{\text{concept}}(g(\mathbf{x}), \mathbf{c})$, then learning f by minimizing $L_{\text{label}}(f(\hat{\mathbf{g}}(\mathbf{x})), \mathbf{y})$

3. Joint: The model is trained via a weighted sum of the losses given by

$\lambda_{\text{concept}} L_{\text{concept}}(g(\mathbf{x}), \mathbf{c}) + \lambda_{\text{label}} L_{\text{label}}(f(g(\mathbf{x})), \mathbf{y})$
such that both losses are minimised simultaneously.

It has been shown experimentally that while the joint models perform the best without interventions, followed by the sequential model, and the independent model performs the worst. This is because the sequential model allows the $\mathbf{c} \to \mathbf{y}$ model to learn a mapping from the concepts produced by the $\mathbf{x} \to \mathbf{c}$ model to label $\mathbf{y}$, where the concepts produced by the $\mathbf{x} \to \mathbf{c}$ model is often different from the true concepts, an underlying requirement for the independent model to perform well. Additionally, the joint model allows the $\mathbf{x} \to \mathbf{c}$ model to simultaneously learn to output a representation of concepts that allow for best performance of the $\mathbf{c} \to \mathbf{y}$ model [3].

When comparing performance under interventions, independent models outperform the two models. They are more sensitive to interventions and each successive intervention step leads to a bigger increase in performance compared to the other two, with better performances after the same number of interventions. The reason behind this is that the independent model learns a mapping from the true concepts to the label, whereas the other two learn a mapping from the predicted concepts to the label. Each intervention modifies the predicted concepts to be closer to the true concepts, which is what the $\mathbf{c} \to \mathbf{y}$ independent model is trained to do [3].

2.2 CEM

While CBMs proved to be useful in achieving machine learning models with high interpretability, they do not perform as well as traditional models that learn a direct mapping from input to labels. This is because the CBM label prediction model relies only on a set of human-interpretable concepts, which limits the performance of the model as traditional models can extract information outside of these concepts that are potentially not human-interpretable [7]. This is even more apparent if the dataset does not contain a complete set of concepts that cover all features present in the input, which is very common in real-life scenarios. As such, there is a trade-off between performance and interpretability, where researchers have developed methods such as extending the CBM bottleneck with a set of unsupervised neurons to increase accuracy at a cost of decreasing interpretability [1].

To overcome this trade-off, Concept Embedding Models (CEMs) were proposed by Zarlenga et al. [7], these are CBMs that further add an additional layer of learnable embeddings before the original bottleneck, learning two embedding vectors for each concept: one for when the concept is and is not present. The architecture of CEMs is shown in Figure **??**. An intermediate scoring function ϕ_i is learnt for each concept i, and the embedding assigned to the bottleneck is an interpolation of the two embeddings based on the scoring function predicting the possibility of the concept to be present.

This architecture also allows for interventions during run-time. By simply modifying the output of the scoring function to be that of the true concept, the bottleneck can be modified similarly to CBMs and improve the performance of the model. Additionally to increase the performance of CEMs, the authors utilised observations mentioned in Section 2.1, where models trained on the true concepts are more sensitive to interventions. They proposed RandInt, a method to randomly intervene during training with $\lambda_{\text{int}} = 0.25$ probability of intervening on a concept. They show that this effectively boosts the performance of the model under interventions during test time without notable effects to the performance without interventions [7].

CEMs successfully solves the trade-off problem between performance and interpretability, allowing for similar performance to traditional models while maintaining the interpretability, along with high concept accuracy. This is because the embedding structure for CEMs allow for encoding of more information in the concept representations and is more machine-interpretable, where the additional information in the bottleneck compared to scalar representations in CBMs lead to a better performing label predictor model. Additionally CEMs are still trained on the same set of human-interpretable concepts as CBM via a similar concept loss, which leads to high interpretability and good intervention performance. It has been shown experimentally that CEMs are able to provide better performance for concept-incomplete dataset tasks (where the concepts do not cover all features present in input), and these learnt concept embedding representations effectively represent the true concepts measured by an alignment score [7].

2.3 IntCEM

Building on top of CEMs, Zarlenga et al. [8] introduced Intervention-aware CEM (IntCEM), which are CEMs that are augmented with a learnable concept intervention policy model. IntCEMs' novelty lies in framing the problem of training a CEM and finding an intervention policy as a joint optimization problem by augmenting existing CEMs with a trainable intervention policy model. This approach offers significant improvements in performance after interventions while maintaining similar performance without interventions. IntCEM achieves this because the intervention policy model learn a good intervention policy specific to the CEM, and the CEM also learns to be more sensitive to interventions by the model, through the introduction of an intervention loss L_{int} and an intervention task loss $L_{\text{int-task}}$.

2.4 RL

2.5 AFA

2.6 Problem Setting

2.6.1 Surrogate Models

2.7 Research Questions

Chapter 3

Related Work

Chapter 4

Design and implementation

Chapter 5

Evaluation

Chapter 6

Summary and conclusions

Bibliography

[1] *Promises and Pitfalls of Black-Box Concept Learning Models*, volume 1, 2021.

[2] Kushal Chauhan, Rishabh Tiwari, Jan Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37:5948–5955, 06 2023.

[3] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 13–18 Jul 2020.

[4] Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6450–6459. PMLR, 18–24 Jul 2021.

[5] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 483–486, 2004.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA, 2018.

[7] Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, and Mateja Jamnik. Concept embedding models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[8] Mateo Espinosa Zarlenga, Katherine M. Collins, Krishnamurthy Dj Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help: Intervention-aware concept embedding models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Appendix A

Technical Details