Mobile Robots Self-Driving Through Image Classification Using Discriminative Learning of Sum-Product Networks

Renato Lui Geh (student), Denis Deratani Mauá (advisor)
Institute of Mathematics and Statistics, University of São Paulo
{renatolg,ddm}@ime.usp.br

Motivation and Overview

Driving has proven to be a very difficult task for machines to emulate, not only due to the inherent complexity of the problem but also because of the need for accurate real-time predictions. Nonetheless, recent advances in computer vision and machine learning have shown promising results in the real-world. Mobile robots are low-cost miniature computers with limited processing power and memory. The problem of self-driving can be similarly applied to the mobile robot domain as a down-scaled version of the same task, with an additional hardware constraint. Sum-product networks are probabilistic graphical models capable of representing tractable probability distributions containing a great number of variables. Exact inference is asymptotically linear to the number of edges in the network's graph, and its deep architecture is capable of representing a wide range of distributions. In this work, we attempt to model autonomous driving by using sum-product networks on a small mobile robot. We model this task as an imitation learning problem through image classification. We present accuracy results on an artificial self-driving dataset for different sum-product network learning algorithms, providing a comparative study not only for different network architectures, but also discriminative and generative models. Finally, we provide a real-world mobile robot implementation on a miniature computer.

Sum-Product Networks

Sum-product networks are highly expressive deep probabilistic graphical models capable of linear time exact inference.

Definition 1. (Sum-product network; Gens and Domingos 2013)
A sum-product network (SPN) is a DAG where each node can be defined recursively as follows.

1. *A tractable univariate probability distribution is an SPN.*
2. *A product of SPNs with disjoint scopes is an SPN.*
3. *A weighted sum of SPNs with the same scope is an SPN, provided all weights are positive.*
4. *Nothing else is an SPN.*

Computing the probability of evidence of an SPN given some query variable X and evidence E is done by a backward pass through the graph, first computing leaf values and then internal nodes until the root is reached. The value of the root $S(X, E)$ is the probability $P(X, E)$ as long as all its sum nodes have normalized weights, i.e. they all sum to one.

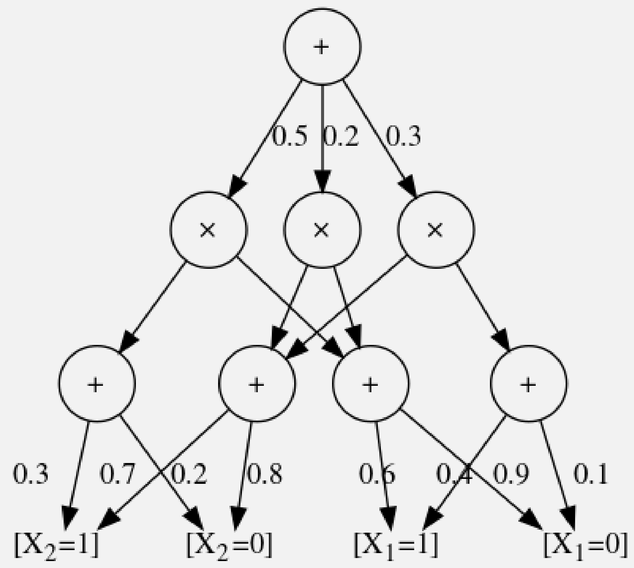

Figure 1: An SPN with indicator variables as leaves (degenerated univariate distributions).

This computation is linear in the number of edges, making SPNs an interesting model for measuring uncertainty in real-time.

Self-Driving in Mobile Robots

Lane tracking is a fundamental aspect of self-driving. The goal of lane tracking in self-driving is to correctly identify the road ahead by detecting track marks. Our approach to this problem is to make use of imitation learning through image classification, where an agent tries to reliably mimick human behavior by classification only. In our case, our agent's (i.e. robot's) goal is to remain within the established "road" by labelling input images as commands for going UP, LEFT or RIGHT. Thus, our objective in this project was to build an image classifier capable of accurate predictions in real-time on a miniaturized computer.

LEFT UP RIGHT

Figure 2: Dataset samples for lane tracking with their respective labels.

Our robot consists of two main processing units. The first, nicknamed the Berry, is a Raspberry Pi 3 Model B, a Quad Core 1.2GHz Broadcom BCM2837 64bit ARMv7 CPU with 1GB of RAM. Coupled with a USB webcam, it is responsible for computing inference and predicting labels. The second, named the Brick, is a Lego Mindstorm NXT v2 unit with an Atmel AT91SAM7S256 48MHz 32bit ARMv4 CPU and 64KB of RAM. It is a dedicated unit for receiving the Berry's commands and translating them into motor power.

Figure 3: Fully assembled robot.

Results

The Moraes and Salvatore 2018 dataset was used for training and testing. It consists of 80×45 RGB images of road following. We chose to apply some pre-processing to the images, namely Otsu's binarization, quantization (i.e. resolution downscaling) and histogram equalization.

Binarization (B) Quantization (Q_n) Equalization (E)

Figure 4: Image transformations done before training and inference.

We trained the SPNs with two structure learning algorithms: Gens and Domingos 2013 (**GD**), Dennis and Ventura 2012 (**DV**); and performed generative (**g**) and discriminative (**d**) gradient descent weight learning on both architectures. We additionally generated the structures with no weight learning (**s**) as a form of control.

(%, secs)	DV+g	DV+d	DV+s	GD+g	GD+d	GD+s
B	(78.8, 0.23)	(78.8, 0.25)	(78.8, 0.25)	(82.8, 0.38)	(83.8, 0.37)	(85.0, 0.31)
Q_2	(78.6, 0.22)	(78.0, 0.24)	(78.0, 0.23)	(78.6, 0.28)	(80.4, 0.34)	(79.4, 0.16)
$Q_2 + E$	(76.6, 0.22)	(76.6, 0.23)	(76.8, 0.23)	(79.6, 0.38)	(82.8, 0.30)	(81.8, 0.27)
Q_4	(78.2, 0.22)	(78.4, 0.22)	(78.2, 0.23)	(76.0, 0.16)	(78.2, 0.17)	(76.4, 0.13)
$Q_4 + E$	(76.6, 0.23)	(76.6, 0.27)	(76.8, 0.29)	(76.0, 0.13)	(74.6, 0.14)	(80.6, 0.13)
Q_6	(77.4, 0.23)	(78.4, 0.24)	(78.4, 0.23)	(75.2, 0.04)	(74.4, 0.05)	(72.0, 0.01)
$Q_6 + E$	(76.0, 0.22)	(76.4, 0.24)	(76.4, 0.28)	(73.0, 0.03)	(75.0, 0.04)	(73.6, 0.02)
$\emptyset$	(78.0, 0.22)	(78.4, 0.26)	(78.4, 0.23)	(62.4, 0.02)	(62.4, 0.01)	(62.4, 0.01)
E	(76.4, 0.23)	(76.4, 0.23)	(76.4, 0.22)	(60.4, 0.01)	(60.0, 0.01)	(61.2, 0.02)

Table 1: Accuracy values in percentage and inference time in seconds for each possible model permutation.

Table 1 shows results for each model and the pre-process filter applied beforehand. In this table we only show even quantizations.

We also applied the model to a real-world application: https://youtu.be/

Conclusions

Sum-product networks were able to somewhat accurately predict the labels. Because of fast inference, prediction error was smoothed out, as the model was capable of correcting itself every hundred milliseconds. Network complexity also played a huge factor in prediction, as a deeper, more complex model achieved better theoretical accuracy, but took longer to predict. This tradeoff can become troublesome, as a self-driving vehicle must be able to predict and correct itself at least as fast as a human. Overall, the model was able to navigate the track reasonably well for a primitive form of control.

References

Dennis, Aaron and Dan Ventura (2012). "Learning the Architecture of Sum-Product Networks Using Clustering on Variables". In: *Advances in Neural Information Processing Systems* 25.

Gens, Robert and Pedro Domingos (2013). "Learning the Structure of Sum-Product Networks". In: *International Conference on Machine Learning* 30.

Moraes, Paula and Felipe Salvatore (2018). *Self Driving Data*. URL: https://github.com/felipessalvatore/self_driving_data.