Reliable Decision Support using Counterfactual Models

Critiqued by: Aditya Jain and Manish Reddy

Short Summary:

- Problem:
  - Supervised Learning Algorithms might be learning about data-set generating policies and not general rules.
  - We need to be able ask and answer Counterfactual questions to ensure we didn't simply learn data-generating policies.
- Solution:
  - Model the data with a general outcome model: $\underbrace{p^*(y \mid t, z_y)}_{\text{[B] Outcome model (GP)}}$
  - Feasible under 4 very important assumptions.
  - Simplify the problem to regression, and use rich-GPs to represent the curve.
  - Finally, use MLE to get the parameters.

#1 Experimental procedure.

- “NIPS - Not Even Wrong?: A Systematic Review of Empirically Complete Demonstrations of Algorithmic Effectiveness in the Machine Learning and Artificial Intelligence Literature “ Reviews multiple things wrong with argumentative chain in NIPS (NeurIPS now!) papers.

ID	Citation	Synthetic	Real-World	Uninformed	Gold Standard	Explanation	Alternative	CIs	CI Reference	Comparison	Generalisation
6767	[106]	Y	Y	N	N	N	Y	N	NA	N	N

- What the paper is missing:
  - Uninformed Comparisons: A very general method un-specific to the problems of the current setting.
  - CIs: Confidence Intervals are absent.
  - Comparison: Statistical comparison between previous and proposed algorithms.

#2 Assumption about confounders.

- A common flaw with most causality research: Ignoring unmeasured confounders.
- Confounders Review: Factors that can affect both history, action and future in a correlated way.
    - Eg: Let’s say we want to find the efficacy of a lung-treatment. We randomly give/not give treatments to all groups so as to isolate the sole effect of the treatment.
    - However, despite averaging across varied groups, the ‘gender’ factor will confound our observations, since Males are more likely to smoke than Females.
    - When we ignore unmeasured confounders, we assume that our X contains all such influences.
- It is usually impossibly hard to mitigate confounding: We would need to know the underlying ancestors in the causal graph of every part of the universe interacting with our process.

#3 Untestable Assumptions ⇒ Unreliability

Assumption 3 (Continuous-Time NUC). *For all times t and all histories $\mathcal{H}_{t^-}$, the densities $\lambda^*(t)$, $p^*(z_u, z_a \mid t)$, and $p^*(a \mid y, t, z_a)$ do not depend on $Y_s[\mathbf{a}]$ for all times $s > t$ and all actions $\mathbf{a}$.*

Assumption 4 (Non-Informative Measurement Times). *For all times t and any history $\mathcal{H}_{t^-}$, the following holds: $p^*(y \mid t, z_y = 1)\,\mathrm{d}y = P(Y_t \in \mathrm{d}y \mid \mathcal{H}_{t^-})$.*

Assumptions 3 & 4 are necessary to establish equivalence between Potential Outcome model and the Counterfactual Gaussian Process **which in general are not testable. Violated assumptions leads to worse performance (in terms of Kendall's score) and unreliability which the paper aims to avoid.**

	Regime A		Regime B		Regime C	
	Baseline GP	CGP	Baseline GP	CGP	Baseline GP	CGP
Risk Score Δ from A	0.000	0.000	0.083	0.001	0.162	0.128
Kendall's τ from A	1.000	1.000	0.857	0.998	0.640	0.562
AUC	0.853	0.872	0.832	0.872	0.806	0.829

What are some advantages of using Gaussian Process Models vs Neural Networks?

Answer · Follow · 127 · Request · 1

5 Answers

Yoshua Bengio, My lab has been one of the three that started the deep learning approach, back in 2006, along with Hinton's...

Answered Apr 6, 2011 · Upvoted by David Warde-Farley, Machine Learning PhD Student, Practitioner/Researcher for 7 years and Anant Raj, Ph.D. Machine Learning & Mathematical Optimization, Max Planck Institute for Intelligent Systems (2...

I would add the following to David Warde-Farley's excellent answers. An advantage of Gaussian Processes is that, like other kernel methods, they can be optimized exactly, given the values of their hyper-parameters (such as the weight decay and the spread of a Gaussian kernel), and this often allows a fine and precise trade-off between fitting the data and smoothing. On small datasets they are very good because of this well-tuned smoothing and because they are still computationally affordable. They are my method of choice for small regression datasets (less than 1000 or 2000 examples). On the other hand, if you want to capture a complicated function (with many many ups and downs, i.e., not necessarily very smooth), then you need a model that can scale to large datasets and that can generalize non-locally (which kernel machines with standard generic kernels, typically local, do not provide). Modern variants of neural networks (so-called Deep Learning, Deep Learning) are more attractive with respect to these two properties, so I would prefer them for larger datasets where there is a lot of structure to be extracted from the data (the target function is not smooth).

#4 Limited Scalability of Gaussian Process limits model usage

- Gaussian processes handle small datasets (1000-2000 points) well but are **computationally inefficient for larger datasets**

- The complexity of Gaussian Processes is generally **$O(n^3)$** where n is the number of data points.