Presentation at Research Group

DALE: Differential Accumulated Local Effects for efficient and accurate global explanations

Vasilis Gkolemis[!,*]

[!]ATHENA Research and Innovation Center

[*]Harokopio University of Athens

March 2023

Who we are

- Vasilis Gkolemis:
  - Research Assistant at ATHENA Research Center (ATHENA RC)
  - First-year PhD at Harokopio University of Athens (HUA)
  - Main focus: Explainability under uncertainty
- Supervisors:
  - Christos Diou (HUA) → Generalization, Few(Zero)-shot learning
  - Eirini Ntoutsi (UniBw-M) → Explainability, Fairness
  - Theodore Dalamagas (ATHENA) → Databases, data semantics
- Paper I will present
  - DALE: Differential Accumulated Local Effects for efficient and accurate global explanations
  - Accepted at Asian Conference Machine Learning (ACML) 2022

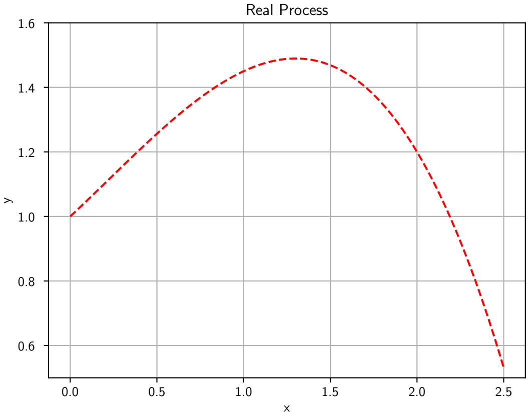
Questions

- Why did the model make a specific decision? local XAI
- What could we change so that the model will make a different decision? counterfactual
- Can we summarize the model's behavior? global XAI
- If models are knowledge extractors, what have they learned? global XAI

Feature Effect: global, model-agnostic, outputs a $1D$ plot

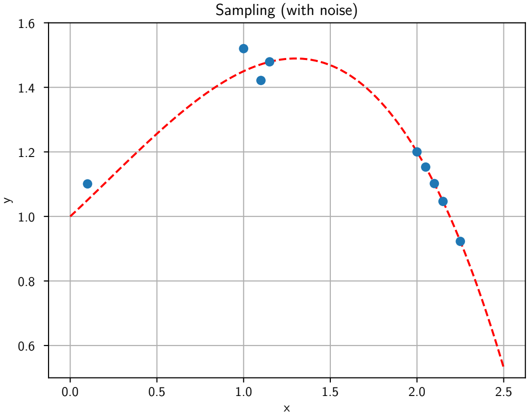
Example

Consider the following mapping $x \to y$



Example

Process unknown $\rightarrow$ we only have samples



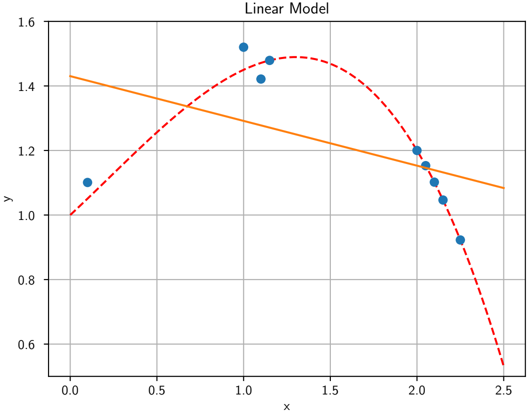
Example

Our goal is to model the process using the available samples (regression)

Example

Linear model → Underfiting!

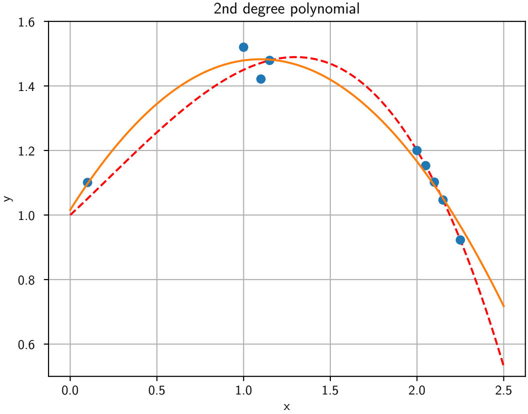
$$y = w_1 \cdot x + w_0$$



Example

2^{nd} degree polynomial $\rightarrow$ Decent Fit!

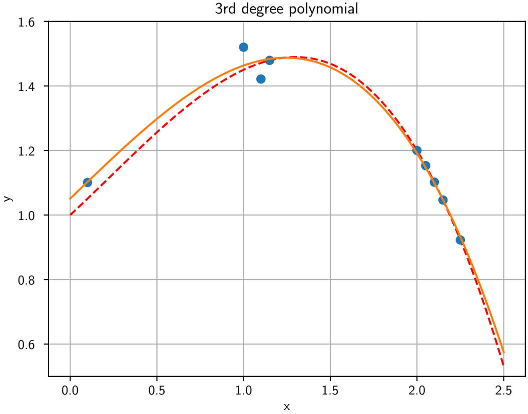
$$y = w_2 \cdot x^2 + w_1 \cdot x + w_0$$



Example

3^{rd} degree polynomial $\rightarrow$ Good Fit!

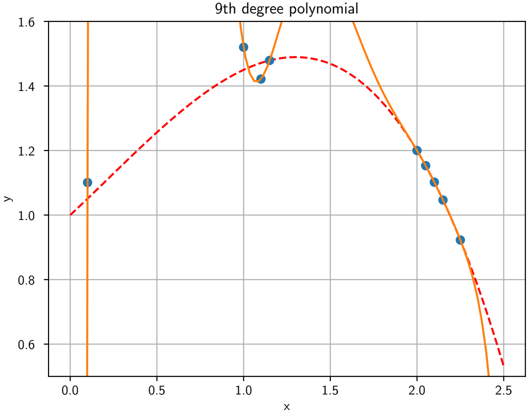
$$y = w_3 \cdot x^3 + w_2 \cdot x^2 + w_1 \cdot x + w_0$$



Example

9^{th} degree polynomial → Overfitting!

$$y = \sum_{i=0}^{9} w_i \cdot x^i$$



Problem diagnosis

- Model behavior is **explained** by the shape of the function
- Overfitting, Underfitting are easily diagnosed
- If the input has multiple dimensions D?
  - We often have tens or hundreds of features
  - Images and signals: Several thousands of input dimensions
- Example: Risk Factors for Cervical Cancer Dataset
  - input: 15 features (smoker, years of hormonal contraceptives, age)
  - output: predict whether a woman will get cervical cancer

Feature Effect

$y = f(x_s) \rightarrow$ plot showing the effect of x_s on the output y

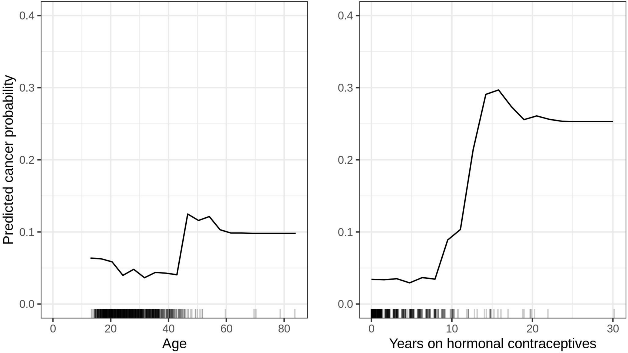


Figure: Image taken from Interpretable ML book (Molnar, 2022)

Feature Effect is simple and intuitive.

Feature Effect Methods

- $x_s \rightarrow$ feature of interest, $\boldsymbol{x_c} \rightarrow$ other features
- Isolating the effect of x_s is a difficult task:
  - features are correlated
  - f has learned complex interactions
- Three well-known methods:
  - Partial Dependence Plots (PDP)
  - M-Plots
  - Accumulated Local Effects (ALE)

Partial Dependence Plots (PDP)

- Proposed by J. Friedman on 2001[1] and is the marginal **effect** of a feature to the model output:

$$f_s(x_s) = \mathbb{E}_{\boldsymbol{x_c}}\left[f(x_s, \boldsymbol{x_c})\right] = \int f(x_s, \boldsymbol{x_c}) p(\boldsymbol{x_c}) d\boldsymbol{x_c}$$

  where:
  - x_s is the feature whose effect we wish to compute
  - $\boldsymbol{x_c}$ are the rest of the features
- Approximation:

$$\hat{f}_s(x_s) = \frac{1}{n} \sum_{i=1}^{n} f(x_s, \mathbf{x}_c^{(i)})$$

[1] J. Friedman. "Greedy function approximation: A gradient boosting machine." Annals of statistics (2001): 1189-1232

Issues with PDPs

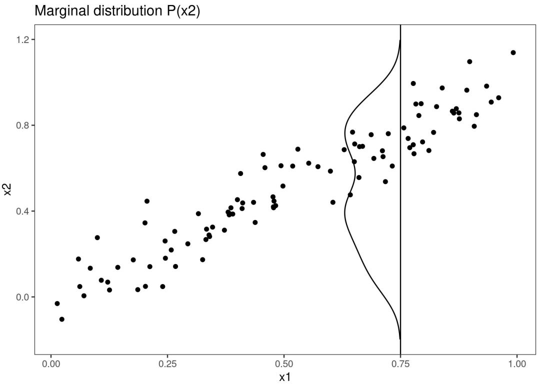


Figure: C. Molnar, IML book

Issues with PDPs

- Correlated features
  - To compute the effect of $x_{\mathtt{age}} = 20$ on the output (cancer probability) it will integrate over all $x_{\mathtt{years\_contraceptives}}$ values, e.g., $[0, 50]$
  - f can have weird behavior when $x_{\mathtt{age}} = 20, x_{\mathtt{years\_contraceptives}} = 20$ (out of distribution)
  - As a result, we have a wrong estimation of the feature effect

MPlots

- We use the value of x_s as a condition, so we integrate over $\mathbf{x}_c | x_s$

$$f(x_s) = \mathbb{E}_{\mathbf{x}_c|x_s}[f(x_s, \mathbf{x}_c)] = \int f(x_s, \mathbf{x}_c) p(\boldsymbol{x_c}|x_s) d\boldsymbol{x_c}$$

  where:
  - x_s is the feature whose effect we wish to compute
  - $\boldsymbol{x_c}$ the rest of the features
- Approximation:

$$f_s(x_s) = \frac{1}{n} \sum_{i:x_s^{(i)} \approx x_s} f(x_s, \mathbf{x}_c^{(i)})$$

MPlots

In the previous example

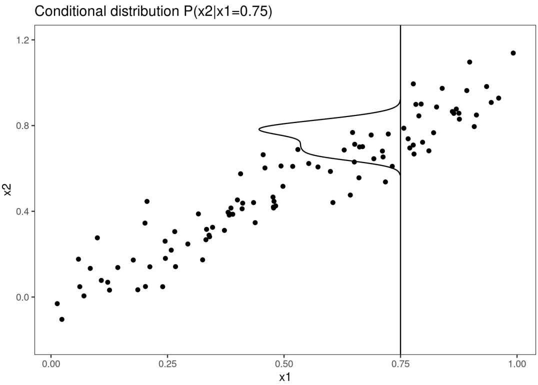


Figure: C. Molnar, IML book

Issues with M-Plots

- Aggregated effect symptom $\rightarrow$ the calculated effects result from the combination of all (correlated) features
- Real effect:
  - $x_{\texttt{age}} = 50 \rightarrow 10$
  - $x_{\texttt{years\_contraceptives}} = 20 \rightarrow 10$
  - aggregated effect close to 20
- Because $x_{\texttt{age}}, x_{\texttt{years\_contraceptives}}$ are correlated, MPlot may assign:
  - $x_{\texttt{age}} = 50 \rightarrow 17 \approx$ aggregated effect
  - $x_{\texttt{years\_contraceptives}} = 20 \rightarrow 17 \approx$ aggregated effect

Accumulated Local Effects (ALE)[2]

- Resolves problems that result from the feature correlation by computing differences over a (small) window

$$f(x_s) = \int_{x_{min}}^{x_s} \mathbb{E}_{\underbrace{\mathbf{x}_c|z}_{realistic}} \underbrace{[\frac{\partial f}{\partial x_s}(z, \mathbf{x}_c)]}_{isolates} \partial z$$

[2] D. Apley and J. Zhu. "Visualizing the effects of predictor variables in black box supervised learning models." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 82.4 (2020): 1059-1086.

ALE approximation

ALE definition: $f(x_s) = \int_{x_{s,min}}^{x_s} \mathbb{E}_{\mathbf{x_c}|z}[\frac{\partial f}{\partial x_s}(z, \mathbf{x_c})]\partial z$

ALE approximation: $f(x_s) = \sum_k^{k_x} \underbrace{\frac{1}{|\mathcal{S}_k|} \sum_{i:\mathbf{x}^i \in \mathcal{S}_k} \underbrace{[f(z_k, \mathbf{x_c^i}) - f(z_{k-1}, \mathbf{x_c^i})]}_{\text{point effect}}}_{\text{bin effect}}$

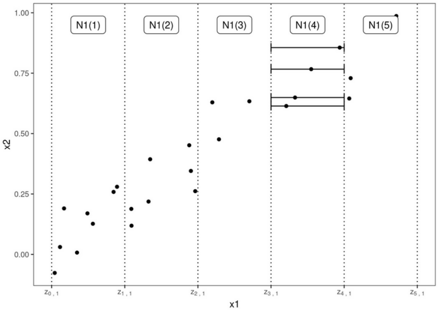


Figure: Image taken from Interpretable ML book (Molnar, 2022)

ALE approximation - weaknesses

$$f(x_s) = \sum_{k}^{k_x} \underbrace{\frac{1}{|\mathcal{S}_k|} \sum_{i:\mathbf{x}^i \in \mathcal{S}_k} \underbrace{[f(z_k, \mathbf{x_c^i}) - f(z_{k-1}, \mathbf{x_c^i})]}_{\text{point effect}}}_{\text{bin effect}}$$

- Point Effect $\Rightarrow$ evaluation at bin limits
  - 2 evaluations of f per point $\rightarrow$ slow
  - change bin limits, pay again $2 * N$ evaluations of f $\rightarrow$ restrictive
  - broad bins may create out of distribution (OOD) samples $\rightarrow$ not-robust in wide bins

---

ALE approximation has some weaknesses

Recap!

- PDP $\rightarrow$ problems with correlated features $\rightarrow$ Unrealistic instances
- MPlot $\rightarrow$ problems with correlated features $\rightarrow$ Aggregated effects
- ALE $\rightarrow$ resolves both issues! But:
- ALE approximation (estimation of ALE from the training set)
  - slow when there are many features
  - unrealistic instances when bins become bigger
- Differential ALE (DALE)!

Our proposal: Differential ALE

$$f(x_s) = \Delta x \sum_{k}^{k_x} \underbrace{\frac{1}{|\mathcal{S}_k|} \sum_{i:\mathbf{x}^i \in \mathcal{S}_k} \underbrace{[\frac{\partial f}{\partial x_s}(x_s^i, \mathbf{x_c^i})]}_{\text{point effect}}}_{\text{bin effect}}$$

- Point Effect ⇒ evaluation on instances
  - Fast → use of auto-differentiation, all derivatives in a single pass
  - Versatile → point effects computed once, change bins without cost
  - Secure → does not create artificial instances

For differentiable models, DALE resolves ALE weaknesses

DALE is faster and versatile - theory

$$f(x_s) = \Delta x \sum_{k}^{k_x} \underbrace{\frac{1}{|\mathcal{S}_k|} \sum_{i:\boldsymbol{x}^i \in \mathcal{S}_k} \underbrace{[\frac{\partial f}{\partial x_s}(x_s^i, \boldsymbol{x_c^i})]}_{\text{point effect}}}_{\text{bin effect}}$$

- Faster
  - gradients wrt all features $\nabla_{\boldsymbol{x}} f(\boldsymbol{x^i})$ in a single pass
  - auto-differentiation must be available (deep learning)
- Versatile
  - Change bin limits, with near zero computational cost

DALE is faster and allows redefining bin-limits

DALE is faster and versatile - Experiments

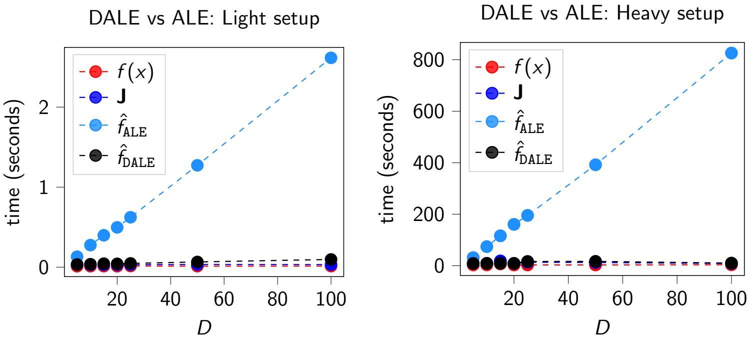


Figure: Light setup; small dataset ($N = 10^2$ instances), light f. Heavy setup; big dataset ($N = 10^5$ instances), heavy f

DALE considerably accelerates the estimation

DALE uses on-distribution samples - Theory

$$f(x_s) = \sum_{k}^{k_x} \underbrace{\frac{1}{|\mathcal{S}_k|} \sum_{i:\boldsymbol{x}^i \in \mathcal{S}_k} \underbrace{[\frac{\partial f}{\partial x_s}(x_s^i, \boldsymbol{x_c^i})]}_{\text{point effect}}}_{\text{bin effect}}$$

- point effect independent of bin limits
  - $\frac{\partial f}{\partial x_s}(x_s^i, \boldsymbol{x_c^i})$ computed on real instances $\boldsymbol{x}^i = (x_s^i, \boldsymbol{x_c^i})$
- bin limits affect only the resolution of the plot
  - wide bins $\rightarrow$ low resolution plot, bin estimation from more points
  - narrow bins $\rightarrow$ high resolution plot, bin estimation from less points

---

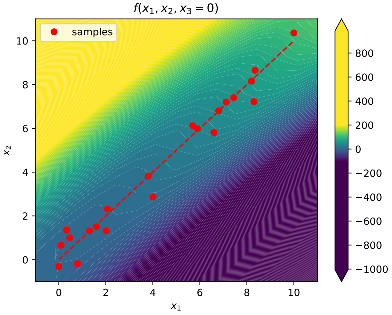
DALE enables wide bins without creating out of distribution instances

DALE uses on-distribution samples - Experiments

$$f(x_1, x_2, x_3) = x_1 x_2 + x_1 x_3 \pm g(x)$$

$$x_1 \in [0, 10], x_2 \sim x_1 + \epsilon, x_3 \sim \mathcal{N}(0, \sigma^2)$$

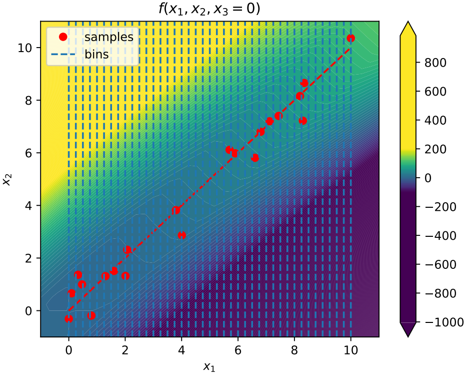
$$f_{\mathtt{ALE}}(x_1) = \frac{x_1^2}{2}$$



- point effects affected by $(x_1 x_3)$ (σ is large)
- bin estimation is noisy (samples are few)

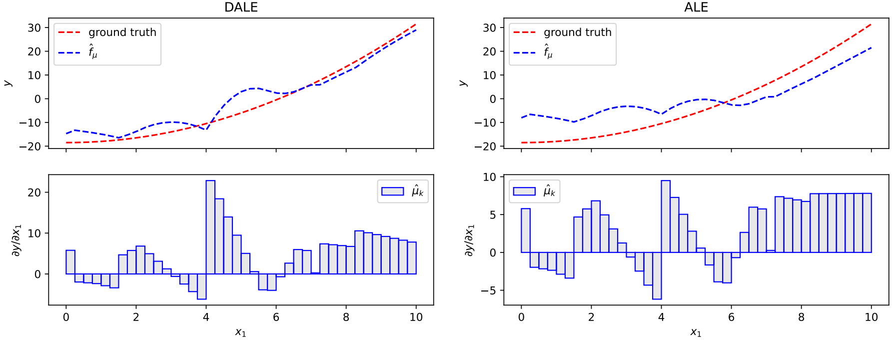
Intuition: we need wider bins (more samples per bin)

DALE vs ALE - 40 Bins



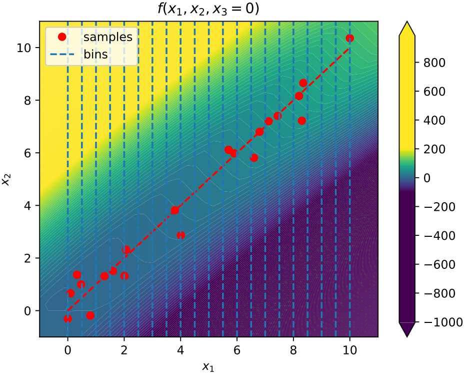
- DALE: on-distribution, noisy bin effect → poor estimation
- ALE: on-distribution, noisy bin effect → poor estimation

DALE vs ALE - 40 Bins



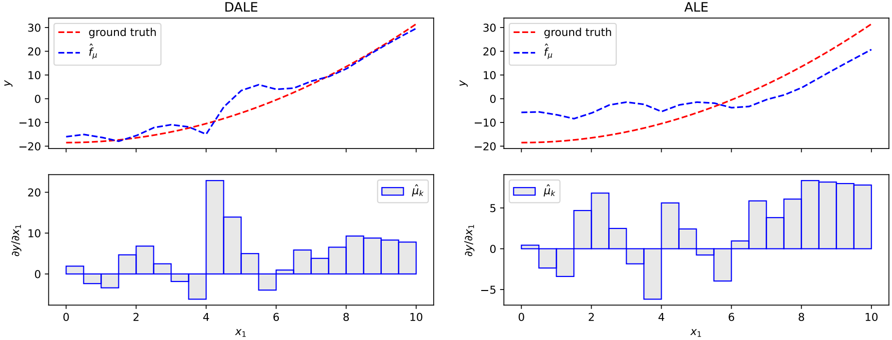
- DALE: on-distribution, noisy bin effect $\rightarrow$ poor estimation
- ALE: on-distribution, noisy bin effect $\rightarrow$ poor estimation

DALE vs ALE - 20 Bins



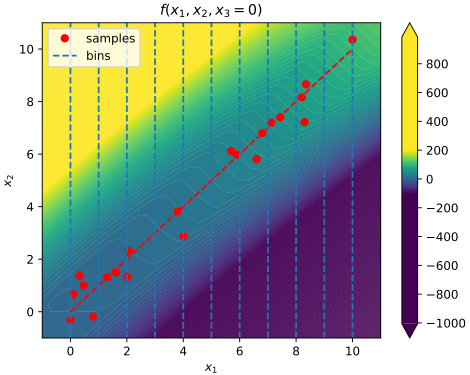
- DALE: on-distribution, noisy bin effect → poor estimation
- ALE: on-distribution, noisy bin effect → poor estimation

DALE vs ALE - 20 Bins



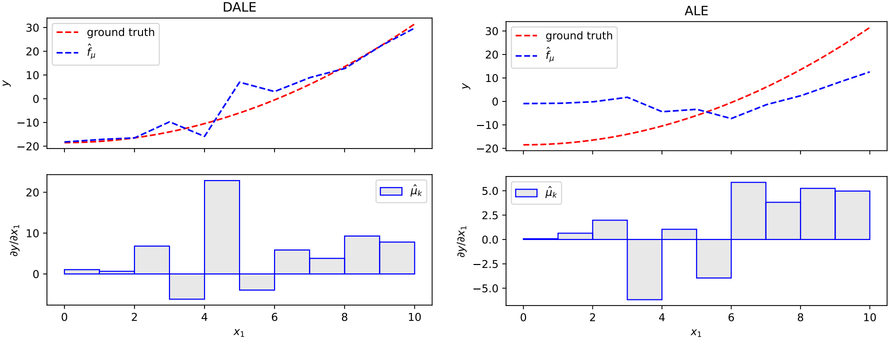
- DALE: on-distribution, noisy bin effect → poor estimation
- ALE: on-distribution, noisy bin effect → poor estimation

DALE vs ALE - 10 Bins



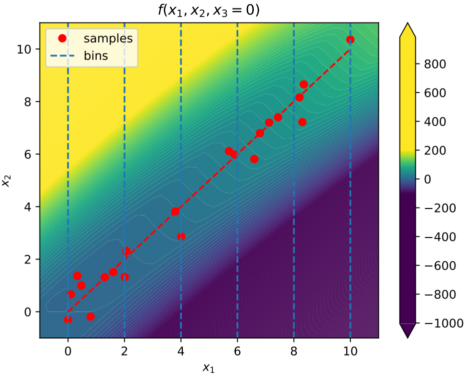
- DALE: on-distribution, noisy bin effect → poor estimation
- ALE: starts being OOD, noisy bin effect → poor estimation

DALE vs ALE - 10 Bins



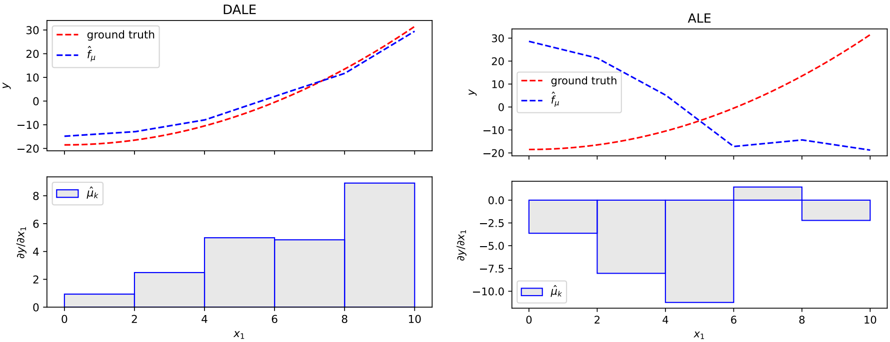
- DALE: on-distribution, noisy bin effect → poor estimation
- ALE: starts being OOD, noisy bin effect → poor estimation

DALE vs ALE - 5 Bins



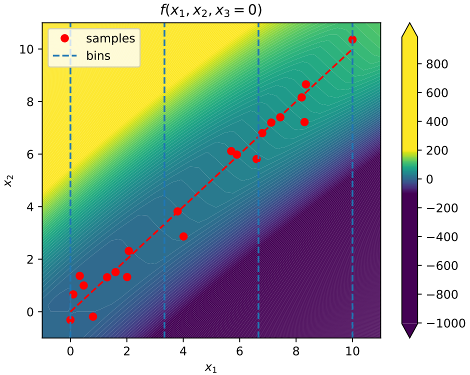
- DALE: on-distribution, robust bin effect → good estimation
- ALE: completely OOD, robust bin effect → poor estimation

DALE vs ALE - 5 Bins



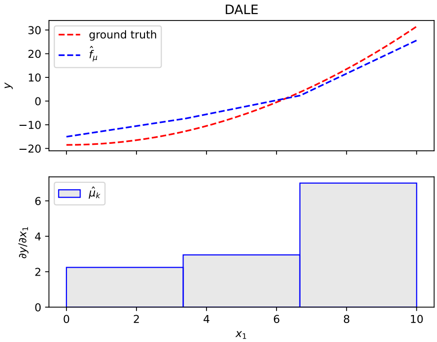
- DALE: on-distribution, robust bin effect → good estimation
- ALE: completely OOD, robust bin effect → poor estimation

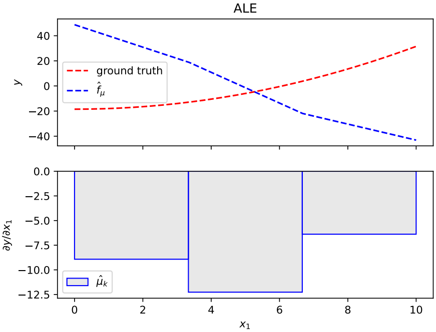
DALE vs ALE - 3 Bins



- DALE: on-distribution, robust bin effect → good estimation
- ALE: completely OOD, robust bin effect → poor estimation

DALE vs ALE - 3 Bins





- DALE: on-distribution, robust bin effect $\rightarrow$ good estimation
- ALE: completely OOD, robust bin effect $\rightarrow$ poor estimation

Future Ideas (1)

PDPs use ICE plots, for exhibiting heterogeneity

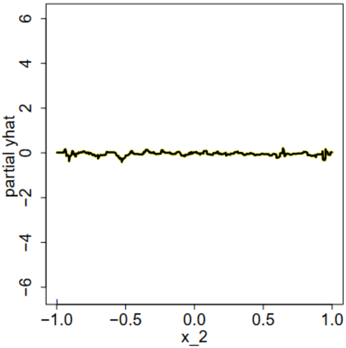


Figure: PDP plot, taken from Goldstein et. al

Interpretation? Maybe $y \perp\!\!\!\perp x_2$

Future Ideas (2)

PDPs use ICE plots, for exhibiting heterogeneity

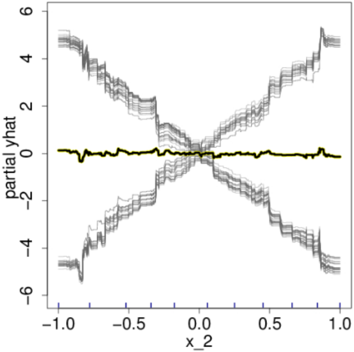


Figure: PDP-ICE plot, taken from Goldstein et. al

Interpretation now? Maybe $y \approx \pm 6x_2$ depending on a condition

Future Ideas (3)

- Could ALE plots do the same?
- Variance inside each bin?

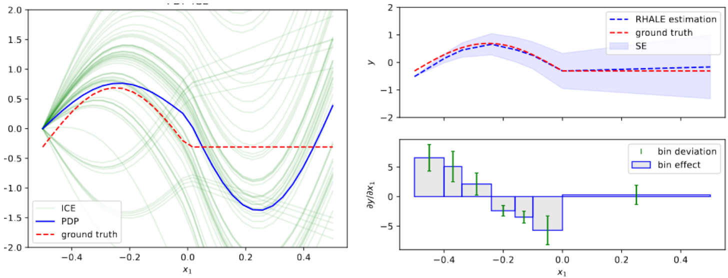


Figure: (Left) PDP-ICE (Right) ALE with heterogeneity

Future Ideas (4) - Regional Effect plots

- Heterogeneity → subspaces with homogeneous effects

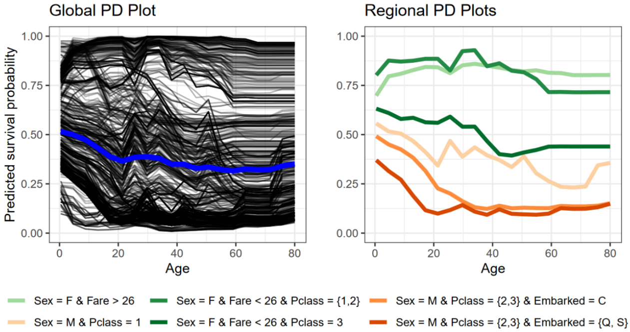


Figure: REPID: Regional Effect plots, taken from Herbinger et. al

Same idea on ALE?

Thank you

- Questions?

References I

Fanaee-T, Hadi and Joao Gama (2013). “Event labeling combining ensemble detectors and background knowledge”. In: *Progress in Artificial Intelligence*, pp. 1–15. ISSN: 2192-6352. DOI: 10.1007/s13748-013-0040-3. URL: [WebLink].

Molnar, Christoph (2022). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. URL: https://christophm.github.io/interpretable-ml-book.