Information Bottleneck as Regularizer for Decision Trees

Niels Warncke

Technische Universität Berlin

niels.warncke@gmail.com

July 29, 2020

Overview

Learning Problems: Classification and Regression

Definition

Let $\mathbb{X}, \mathbb{Y}$ be random variables with an unknown joint probability distribution $P_{\mathbb{X},\mathbb{Y}}$, $\mathcal{X}, \mathcal{Y}$ their domain and $X \in \mathbb{X}^N$, $Y \in \mathbb{Y}^N$ be observed samples. Finding a function f such that $E_{\mathbb{X},\mathbb{Y}}[J(f(\mathbb{X}), \mathbb{Y})]$ is small is called a classification or regression problem. [1]

- Classification: $|\mathcal{Y}| \in \mathbb{N}$ - the target variable represents a category
- Regression: $|\mathcal{Y}| \in \mathbb{R}^{D_y}$ - the target can be any vector
- Example: Predict if an image shows a cat, predict the age of a person shown in an image.

[1] This definition can be seen as a special case of the definition by [Mit]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance in task T. measured by P, improves with experience E."

Decision Trees

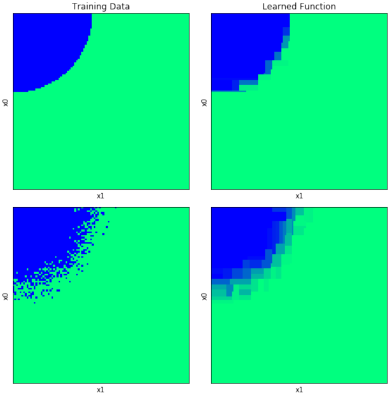
Decision tree inducers provide an algorithm to solve classification and regression problems.

- A binary decision tree T splits the data: $s_T(x) = x_{i_T} \leq t_T$ unless T is a leaf
- If T is a leaf, it predicts $T(x) = c_T$ for some constant c_T
- If T is not a leaf, it creates two subtrees T_{left} and T_{right}, and predicts:

$$T(x) = \begin{cases} T_{left}(x) & \text{if } s_T(x) = \text{TRUE} \\ T_{right}(x) & \text{if } s_T(x) = \text{FALSE} \end{cases}$$

- This defines the capacity of the model

Decision Trees - Example



Left: Training data of an artificial classification problem. *Right:* Learned function of a decision tree

Decision Trees - Learning

- Fitting a tree is an optimization problem
- Many exact solutions are NP hard: e.g. finding a minimal tree that fits the data
- Instead of exact solutions: greedy algorithms (bottom up vs top down)
- Different loss functions have been proposed: InformationGain, Gini Index, Likelihood-Ratio Chi–Squared Statistics, DKM Criterion, Gain Ratio, ...- see [RM]
- This project evaluates the Information Bottleneck as loss function

Decision Trees - Top Down

```
class DecisionTree():
  def fit(self, X, Y):
    best_loss = infinity
    for d in range(X.shape[1]): # O(D)
      loss, thresh, left_split, right_split = \
        best_split(X[:,d], Y) # O(N)
      if loss_d < best_loss:
        update best_loss, X_l, Y_l, X_r, Y_r, t_T, i_T
    if stopping_criterion is fulfilled:
      self.c_T = best_constant_estimator(Y)
      return self
    self.left = DecisionTree().fit(X_l, Y_l)
    self.right = DecisionTree().fit(X_r, Y_r)
    self.prune()
    return self
```

The Information Bottleneck

Let $\mathbb{X}, \mathbb{Y}$ be random variables with a known joint probability distribution $P_{\mathbb{X},\mathbb{Y}}$, $\mathcal{X}, \mathcal{Y}$ their domain and $|\mathcal{X}| \in \mathbb{N}, |\mathcal{Y}| \in \mathbb{N}$.

Intuition: we want to encode a message $\mathbb{X}$ such that we keep as much information about $\mathbb{Y}$ as possible, while compressing $\mathbb{X}$ as much as possible.

We achieve this by finding a soft partioning of $\mathbb{X}$ defined by a mapping $P_{\hat{X}|\mathbb{X}}$ such that $I(\hat{X}, \mathbb{X})$ is minimized while $I(\hat{X}, \mathbb{Y})$ is maximized.

The solution is the minima of the functional:

$$P_{\hat{X}|\mathbb{X}} = argmin_{p(\hat{X}|\mathbb{X})} I(\hat{X}, \mathbb{X}) - \beta I(\hat{X}, \mathbb{Y}) \quad (1)$$

This was introduced by [TPB].

The Information Bottleneck Iterative Algorithm

Iterative algorithm that converges to the optimal $P_{\hat{X}|\mathbb{X}}$ for the IB problem, similar to the Blahut-Arimoto Algorithm ([Bla] [Ari])

$$p_t(\hat{x}|x) = \frac{p_t(\hat{x})}{Z_t(x,\beta)} exp(-\beta d(x,\hat{x}))$$

$$p_{t+1}(\hat{x}) = \sum_x p_t(\hat{x}|x) P_{\mathbb{X}}(x)$$

$$p_{t+1}(y|\hat{x}) = \frac{p_t(y,\hat{x})}{p_{t+1}(\hat{x})} = \frac{\sum_x P_{\mathbb{X},\mathbb{Y}}(x,y) p_t(\hat{x}|x)}{p_{t+1}(\hat{x})}$$

Where $d(x,\hat{x}) = D_{KL}(P_{\mathbb{Y}|\mathbb{X}=x} || P_{\mathbb{Y}|\hat{X}=\hat{x}})$ [2]

[2] In equation (31) of [TPB], the update rule is stated as $p_{t+1}(y|\hat{x}) = \sum_y P_{\mathbb{Y}|\mathbb{X}}(y|x) p_t(x|\hat{x})$, which cannot be correct since it does not depend on y of the lefthand side of the equation.

Comparison: Decision Trees and Information Bottleneck Iterative Algorithm

Common

- Prediction:
  Trees: $X \to \mathit{leaf} \to Y$
  IB Iterative Algorithm: $X \to \hat{X} \to Y$
- usage of loss function that can be expressed as expectation over $\mathbb{X}, \mathbb{Y}$ (next slide)

Differences

- IB assumes knowledge of $P_{\mathbb{X},\mathbb{Y}}$
- IB requires finite $\mathcal{X}$ or explicit quantization
- DT is sensitive to the parameterization of $\mathbb{X}$
- IB finds an optimal solution, DT can and will get stuck in local optima

Information Bottleneck in Decision Trees

$J_{IB;\beta}$ can be estimated in a decision tree:

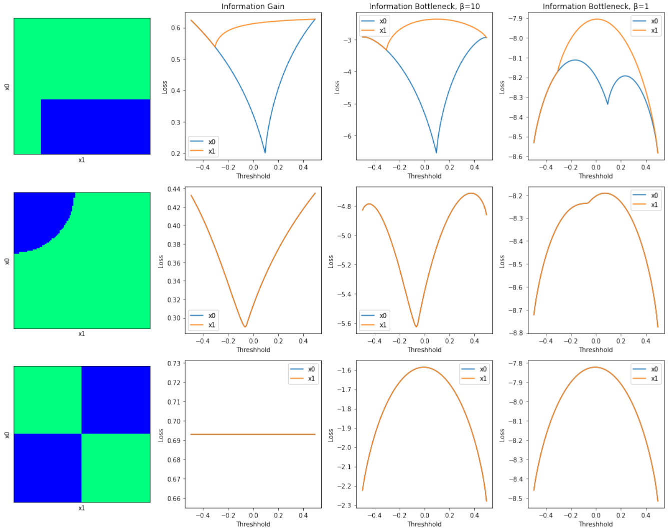
$$\begin{aligned}
J_{IB;\beta}(Y) &= I(\mathbb{X}, \hat{X}) - \beta I(\hat{X}, \mathbb{Y}) \\
&= H(\mathbb{X}) - H(\mathbb{X}|\hat{X}) - \beta(H(\mathbb{Y}) - H(\mathbb{Y}|\hat{X})) \\
&= const - H(\mathbb{X}|\hat{X}) + \beta H(\mathbb{Y}|\hat{X}) \\
&= const - E_{\mathbb{X},\hat{X}}[-logP(\mathbb{X}|\hat{X})] + \beta E_{\hat{X},\mathbb{Y}}[-log(P(\mathbb{Y}|\hat{X}))] \\
&\simeq const - 1/N \sum_i log|\{x'|x' \in X, leaf(x') = leaf(x_i)\}| \\
&+ \beta/N \sum_i D_{KL}(y_i || T(x_i))
\end{aligned}$$

where $T(x)$ is the empirical distribution of Y given $leaf(x)$ in the train set.

This can be greedily optimized:

$$J_{IB;\beta}(Y) = \frac{|Y_{left}|}{|Y|} J_{IB;\beta}(Y_{left}) + \frac{|Y_{right}|}{|Y|} J_{IB;\beta}(Y_{right})$$

Information Bottleneck in Decision Trees



Information Bottleneck in Decision Trees

- Similar to InformationGain + regularizer
- Time and space complexity of the algorithm not affected, but no more pruning necessary
- IB in neural networks: $I(\mathbb{X}, \hat{X})$ and $I(\hat{X}, \mathbb{Y})$ are hard to estimate
  - Information Dropout [AS]: add multiplicative noise to intermediate activations
  - MINE [BBR+]: general purpose estimator for mututal information using neural networks, which can then be used to train another network with the IB objective

Experiments

Open notebook on localhost

or

Open notebook on google colab

Conclusion

- Analogy drawn between Information Bottleneck and decision trees
- Decision trees can be trained with IB inspired loss
- On a small example of handwritten digit classification, the generalization accuracy improved compared to two baselines:
  - Pure InformationGain
  - Default implementation of sklearn, which uses another loss function (gini impurity)
- Open questions:
  - How does an ensemble like random forest of these models perform?
  - Is there a bound for the expected generalization loss?

References I

- S. Arimoto, *An algorithm for computing the capacity of arbitrary discrete memoryless channels*, no. 1, 14–20.
- Alessandro Achille and Stefano Soatto, *Information dropout: Learning optimal representations through noisy computation*.
- Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R. Devon Hjelm, *MINE: Mutual information neural estimation*.
- R. Blahut, *Computation of channel capacity and rate-distortion functions*, no. 4, 460–473, Conference Name: IEEE Transactions on Information Theory.
- Thomas M. Mitchell, *Machine learning*, 1 ed., McGraw-Hill, Inc.

References II

- Lior Rokach and Oded Maimon, *Decision trees*, Data Mining and Knowledge Discovery Handbook (Oded Maimon and Lior Rokach, eds.), Springer US, pp. 165–192.
- Naftali Tishby, Fernando C. Pereira, and William Bialek, *The information bottleneck method*.

The End