SRIP

Graphics Visualisation and Computing Lab

Uncertainty Estimation and Analysis in Point Clouds

Authors:

Arjun Subhedar

IMT2021069

May-July, 2024

Contents

1 Objective

- The goal of this project is to estimate (quantify) uncertainty in classification of points in a point cloud and also make use of this uncertainty to improve classification models.
- This project also aims to explore how Bayesian Neural Networks can be used for classification and uncertainty estimation.

2 Theory - Key Points

2.1 Uncertainty

Uncertainty is of two types, which are Aleatoric Uncertainty and Epistemic Uncertainty.

- **Epistemic Uncertainty:** Caused due to lack of data points, can be resolved by using more data. This is inherent to the model.
- **Aleatoric Uncertainty:** The inherent noise/ambiguity in the dataset. Irreducible with more data. For example, multiple labels being observed at a given point. Aleatoric uncertainty is only meaningful in-distribution, as, by definition, it quantifies the level of ambiguity between the different classes.

2.2 Point Clouds

- In a point cloud, each point is associated with a class label, say, tree, pole, building etc.
- For classification of individual points in the point cloud, we need to get additional information (features) which are essential for determining the class of a point.
- As described in Weinmann's paper, there are 21 geometric features that can be extracted for points in a point cloud.
- The following is the sequence of steps that happens in for classification of points in a point cloud- Neighborhood selection, Features Extraction, Feature Selection, supervised Classification.

2.3 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) incorporate Bayesian inference into neural networks, providing a probabilistic approach to model uncertainty.

- In BNNs, unlike normal neural networks where we have fixed weights, the weights are represented as probability distributions. This means each neuron considers a range of values for each input, adding a layer of probabilistic reasoning.
- The major difference lies in how BNNs handle weights and biases. Instead of fixed values, BNNs use probability distributions, allowing them to express uncertainty and continuously update their beliefs. This allows the algorithm to assess the probability of each prediction and provide greater insight into the accuracy of the output.

- The variance of the predictions from different samples of the weight distribution provides a measure of uncertainty. High predictive variance indicates high uncertainty about the prediction. High predictive variance indicates that the model is uncertain about the prediction due to limited knowledge. As more data is gathered, the model can refine its weights, reducing this uncertainty.

3 Experiments

3.1 Dataset

The dataset used for all the experimentation is Vaihingen (Area 2 and Area3). The dataset has points with three coordinates and a label (1, 2, 5, or 8) associated with each point.

3.2 Feature Computation

- Initially, I was using the features computed by CloudCompare to train models. But, the classification results were not good enough.
- Next, I wrote code in PySpark to efficiently compute the features for each point by parallelizing the process. But, even this was taking a lot of time to compute features on my machine.
- Came across this python library on GitHub called **Jakteristics**. This library is easy to use, computes features very quickly and I compared the features computed through this library with the features that I manually computed for some points and they were exactly the same indicating the feature computation was accurate.

3.3 Classification

- We had written the code to estimate point-wise epistemic uncertainty (using Gaussian Discriminant Analysis) and aleatoric uncertainty (using softmax entropy loss) in last semester's PE/RE work. My first intuition was to use these as features in classification model to see if the accuracy improves. The features used in this experiment were the ones computed by CloudCompare.
  The accuracy of the the neural network for the classification task with only the features was around 65%. When I concatenated the epistemic uncertainty of every point as an additional feature, the accuracy went up to almost 70%.
  Since, the accuracy scores were not as good as I had expected, I realised the feature computation done by CloudCompare was not accurate.
- Next, I used the features computed by Jakteristics, and the accuracy of the neural network with just the fifteen features ('z', 'eigenvalue sum', 'omnivariance', 'eigenentropy', 'anisotropy', 'planarity', 'linearity', 'PCA1', 'PCA2', 'surface variation', 'sphericity', 'verticality', 'nx', 'ny', 'nz') went above 80%.
- Currently, I am trying to implement the Bayesian Neural Network for classification. A very simple model that I developed by using online resources.

- Lastly, I implemented a custom loss function designed in such a way that, the point with higher confidence value (complement of uncertainty) has more weight as compared to those points which have a lower confidence measure.

3.4 Results

All the following results are from the features computed through Jakteristics. I conducted four experiments for different neigbourhood sizes (0.10m, 0.15m, 0.20m, 0.25m) whose accuracy and loss (cross-validation scores) measures are listed below for every apporach and dataset.

3.4.1 Bayesian Neural Network

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy	40.2%	40.4%	41%	40.1%
Loss	1.333	1.302	1.301	1.312

Table 1: Validation accuracy and loss scores

Since the accuracy scores weren't good enough, I did not run the model for the Vaihingen (Area3) dataset.

3.4.2 Basic Neural Network

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy	76.55%	80.70%	80.46%	80.77%
Loss	0.54	0.46	0.47	0.46

Table 2: Accuracy and loss scores for Vaihingen (Area2)

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy(train)	81.76%	83.25%	83.36%	83.48%
Loss(train)	0.42	0.39	0.40	0.39

Table 3: Accuracy and loss scores for Vaihingen (Area3)

3.4.3 Using Epistemic Uncertainty as a Feature

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy	78.00%	80.31%	80.52%	81.77%
Loss	0.51	0.47	0.47	0.44

Table 4: Accuracy and loss scores for Vaihingen (Area2)

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy	81.53%	83.48%	83.88%	83.44%
Loss	0.43	0.38	0.38	0.39

Table 5: Accuracy and loss scores for Vaihingen (Area3)

3.4.4 Using a Custom Loss Function

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy	76.75%	80.33%	81.35%	82.00%
Loss	0.26	0.22	0.19	0.19

Table 6: Accuracy and loss scores for Vaihingen (Area2)

	0.10 m	0.15 m	0.20 m	0.25 m
Accuracy	81.82%	83.59%	83.77%	84.01%
Loss	0.24	0.18	0.17	0.18

Table 7: Accuracy and loss scores for Vaihingen (Area3)

4 Conclusion

- BNNs did not perform well on the point cloud data. Upon research, I found out that BNNs work most efficiently with data that a lesser number of data points. Traditional neural networks are data hungry, and overfit on small and noisy data. But such data is handled well by BNNs.
  Point clouds are not small datasets. Each of the Vaihingen datasets had over 200,000 points and each class had a good number of points lying in it, so it was balanced as well. Hence, using BNNs int this situation did not really make sense and in turn gave unsatisfactory results.

- As can be observed from the table, the basic classification neural network is almost as good as the other two subsequent models. But in both the datasets, the other two models are very slightly better in almost all neighborhood sizes for which the experiments were conducted.

- The loss values for the basic neural networks and the model where uncertainty was used a feature, are significantly more (almost double) than that of the last model which uses a custom loss function.
  Loss is a value that represents the summation of errors in our model. It measures how well (or bad) our model is doing. If the errors are high, the loss will be high, which means that the model does not do a good job.
  This implies, even though the accuracy scores for the three models are almost similar, the last one (weighted loss function) is the best amongst the three since it's commiting the least amount of errors.

5 Challenges

- The first challenge was the efficient feature computation. Brute force methods were able to get accurate features but then those methods also took a loot of time. CloudCompare was faster but not accurate. I faced a lot of issues and spent a lot of time with the PySpark code to parallelize the feature computation process but even that didn't work out in the end. Finally, Jakteristics was helpful.
- How to use the uncertainty measures to fine tune the classification models was another challenge. There was no reading material available for this.
- Writing the custom loss function also took a lot of time. Resolving errors is what took a huge chunk of time.

6 References

- **Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015)**
  - **Title:** Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers
  - **Source:** ISPRS Journal of Photogrammetry and Remote Sensing, 105, 286-304
- **Mukhoti, Jishnu, et al. "Deep deterministic uncertainty: A simple baseline." arXiv preprint (2021).**
  - **Title:** Deep Deterministic Uncertainty: A Simple Baseline
  - **Source:** ArXiv.org (pre-print)
- **Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, Yarin Gal (2020)**
  - **Title:** Uncertainty Estimation Using a Single Deep Deterministic Neural Network
  - **Source:** Proceedings of the 37th International Conference on Machine Learning, PMLR 119:9690-9700
- For Bayesian Neural Networks, I referred to the following websites:
  - https://www.databricks.com/glossary/bayesian-neural-network
  - https://medium.com/@costaleirbag/a-first-insight-into-bayesian-neural-networks-bnn-c767551e9526
  - https://towardsdatascience.com/why-you-should-use-bayesian-neural-network-aaf76732c150
  - https://www.tensorflow.org/probability