Report

Feiyang Huang

December 21, 2022

1 Data doppelgängers

1.1 Definition

Data doppelgängers are pairs of data samples that are very similar to each other in terms of their measurements. They can occur when data are independently derived but are still very similar due to chance or other factors. Data doppelgängers can have a confounding effect on the performance of machine learning (ML) models, causing them to perform well regardless of how they are trained. This is known as the doppelgänger effect. In the context of drug development, data doppelgängers can cause problems when evaluating the performance of ML models that are used to identify potential drug targets or to repurpose existing drugs to treat other diseases. Therefore, it is important to identify and address data doppelgängers in order to ensure the reliability of the model's performance.

1.2 Data doppelgängers in different fields

The doppelganger effect can exist in any type of data. In the context of analyzing news text by natural language processing (NLP), the "doppelganger effect" could refer to the occurrence of duplicate or highly similar news in a dataset. This could happen when the same text is entered multiple times due to errors or mistakes in data entry, or when the same text is present in multiple sources without being properly linked or merged. When this text analysis algorithm is designed to count the number of occurrences of a particular word or phrase in a dataset, it may produce an inflated count if there are duplicate or highly similar text present.

2 Methods to avoid Data doppelgängers

2.1 Proper Model

According to this paper, choosing the proper machine learning method can reduce the impact of the Data doppelgängers. One example is that kNN and Naïve Bayes are more affected by the Data doppelgängers than decision tree and logistic regression models. One possible reason why decision tree and logistic regression models may be less affected by the Data doppelgängers is that they are less sensitive to small differences in the data compared to other models. Naïve Bayes and k-nearest neighbors (KNN) are models sensitive to individual data points and can be affected by noise. Naïve Bayes is a probabilistic model that makes predictions based on the probability of a particular class given the input features. It assumes that the input features are independent of each other, which means that each feature contributes to the prediction independently of the others. This makes Naïve Bayes sensitive to individual data points because a single data point can significantly impact the probability estimates. KNN is a non-parametric model that makes predictions based on the "k" closest training examples to a given sample. It is sensitive to individual data points because the model's prediction is heavily influenced by the characteristics of the "k" nearest neighbors, which can be affected by the presence or absence of individual data points. On the other hand, decision trees and logistic regression are both parametric models that are less affected by individual data points. This is because they make predictions based on a fixed set of parameters that are learned from the training data rather than being influenced by the characteristics of specific individual data points. This makes them less sensitive to noise in the data and generally more robust to the presence or absence of individual data points.

2.2 Meta data

In this paper, the authors suggest using meta-data to identify potential data doppelgängers to prevent the doppelgänger effect. Meta-data refers to data about data. It is information that describes the characteristics and context of a dataset but is not part of the actual data itself. Meta-data can be used to anticipate the range of values that might be expected for certain metrics, such as the Pearson product-moment correlation coefficient (PPCC). By identifying potential data doppelgängers based on meta-data, it may be possible to assort them into either the training or validation sets, effectively preventing the doppelgänger effect and allowing for a more objective evaluation of the performance of machine learning models.

2.3 Data stratification

Data stratification is a method for evaluating the performance of a machine learning model by dividing the data into different groups or strata based on certain characteristics, such as similarity. This paper stratifies the data into strata based on Pearson product-moment correlation coefficient (PPCC) data doppelgängers and non-PPCC data doppelgängers and evaluate the model performance on each stratum separately. By assuming that each stratum represents a known proportion of the real-world population, it is possible to understand the classifier's real-world performance by considering each stratum's real-world prevalence. This approach can also identify gaps in the model's performance by highlighting strata where the model performs poorly. In this case, poor performance on papillary renal cell carcinoma (RCC) samples, which make up 10% of kidney cancer cells, would indicate an area of weakness for the classifier that would need to be addressed.

2.4 Meta Pseudo labels

For small data sets, the impact of data doppelgängers is more severe, and direct deletion of data can lead to insufficient data volume. One way to address the lack of data volume can be to combine a small amount of labeled data with a large amount of unlabeled data for training.

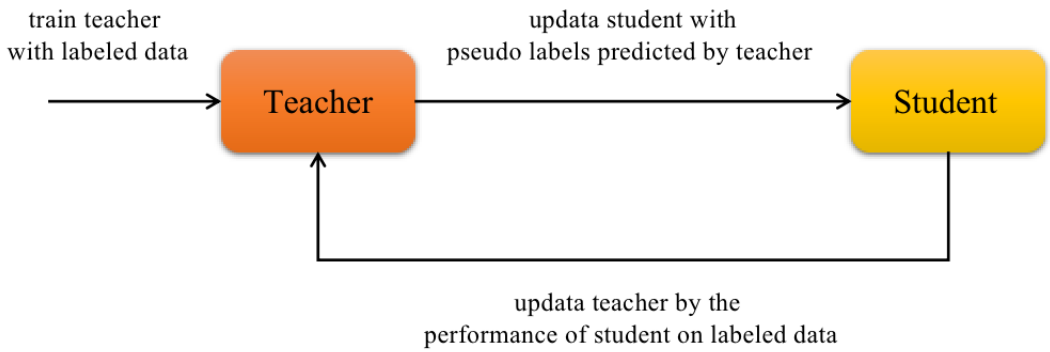

Figure 1: Meta Pseudo Labels

Meta pseudo labels are a technique used in machine learning to improve a model's performance by using one model's predictions to label the data for another model. This can be useful when there is a large amount of unlabeled data and only a small amount of labeled data. The basic idea behind meta pseudo labels is to use a Teacher model that has already been trained on a labeled dataset to make predictions on the unlabeled data. These predictions are then treated as "pseudo labels" and used to train a Student model. The Student model is then fine-tuned on the labeled data to improve its performance. There are several benefits to using meta pseudo labels. One benefit is that it allows you to use large amounts of unlabeled data, which can help improve your model's overall performance. Additionally, using meta pseudo labels can help to reduce the amount of labeled data needed to train a model, which can be helpful if obtaining labeled data is expensive or time-consuming.

The student network is first trained using the pseudo labels generated by the teacher network for

the unlabeled data, and then it is trained using the true labels for the labeled data. In these two steps, the loss will be then back-propagated through the student network to update its parameters. The teacher network is then trained using the predicted labels for the labeled data and the pseudo labels for the unlabeled data. The progress of the training is tracked using meters for the student and teacher losses. It is worth noting that the training loop begins by iterating over a specified number of steps. At each step, the gradients of the student and teacher networks are reset.

Here is the detailed procedure of Meta Pseudo Labes:

Algorithm 1 Meta Pseudo Label Algorithm

1: Input labelled dat x_l, y_l and unlabelled data x_u
2: Initialize $\theta_T^{(0)}$ and $\theta_S^{(0)}$, the parameters for Teacher model and Student model
3: **for** episode = 0 to N **do**
4: Generate pseudo label from Teacher model $\hat{y}_l \sim P(\cdot|x_u, \theta_T)$
5: Update student model using $\hat{y}_l$

$$\theta_S^{(t+1)} = \theta_S^{(t)} - \eta_S \nabla_{\theta_S} \mathrm{CE}\left(\widehat{y}_u,\, S\left(x_u; \theta_S\right)|_{\theta_S=\theta_S^{(t)}}\right.$$

6: Compute feedback coefficient and gradient for teacher model based on the the cross-entropy of student on labelled data

$$h = \eta_S \cdot \left(\left(\nabla_{\theta_S'} \mathrm{CE}\left(y_l, S\left(x_l; \theta_S^{(t+1)}\right)\right)^\top \cdot \nabla_{\theta_S} \mathrm{CE}\left(\widehat{y}_u, S\left(x_u; \theta_S^{(t)}\right)\right)\right)\right.$$

$$g_T^{(t)} = h \cdot \nabla_{\theta_T} \mathrm{CE}\left(\widehat{y}_u, T\left(x_u; \theta_T\right)\right)|_{\theta_T=\theta_T^{(t)}}$$

7: Compute the gradient of teacher model on labelled data

$$g_{T,\text{ supervised}}^{(t)} = \nabla_{\theta_T} \mathrm{CE}\left(y_l, T\left(x_l; \theta_T\right)\right)|_{\theta_T=\theta_T^{(t)}}$$

8: Compute the the gradient of teacher model on the UDA loss with unlabelled data

$$g_{T,\mathrm{UDA}}^{(t)} = \nabla_{\theta_T} \text{ CE } \left(\text{StopGradient}\left(T\left(x_l\right); \theta_T\right), T\left(\text{ RandAugment } \left(x_l\right); \theta_T\right)\right)|_{\theta_T=\theta_T^{(t)}}$$

9: Update the parameter of teacher model

$$\theta_T^{(t+1)} = \theta_T^{(t)} - \eta_T \cdot \left(g_T^{(t)} + g_{T,\text{ supervised}}^{(t)} + g_{T,\mathrm{UDA}}^{(t)}\right)$$

10: **end for**