Feature Decorrelation - AI in Industry Project

Davide Perozzi, Fabian Vincenzi
{davide.perozzi, fabian.vincenzi}@studio.unibo.it

May 29, 2023

1 Introduction

Our task is to investigate the effect of feature decorrelation in model training, specifically the stability of the training. To achieve this, we built a dataset and its decorrelated version with three different standard data distribution graphs and two with noise and observed the difference in training with two different model architectures.

2 Our pipeline

2.1 Graphs and data generation

Using the Python library datagen we created three different graphs, two with noise and three without [1]:

- ```
  A = dg.normal(mu=0, sigma=1, name='A')
  B = dg.normal(mu=0, sigma=1, name='B')
  C = dg.descendant(A + B, name='C')
  D = dg.descendant(A + B - 3 * C, name='D')
  ```

- ```
  A = dg.normal(mu=0.5, sigma=0.5, name='A')
  B = dg.normal(mu=0.5, sigma=0.5, name='B')
  C = dg.descendant(A + B, name='C')
  D = dg.descendant(A + B - 3 * C, name='D')
  ```

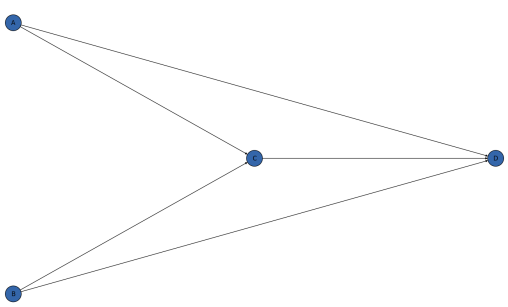
- ```
  A = dg.normal(mu=1.0, sigma=0.2, name='A')
  B = dg.normal(mu=0.8, sigma=0.4, name='B')
  C = dg.descendant(A + B, name='C')
  D = dg.descendant(A + B - 3 * C, name='D')
  ```

  With noise:

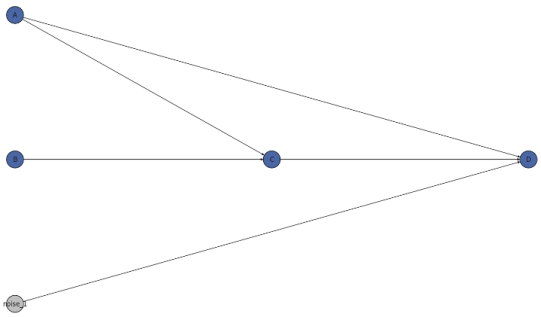
- ```
  A = dg.normal(mu=0, sigma=1, name='A')
  B = dg.normal(mu=0, sigma=1, name='B')
  C = dg.descendant(A + B, name='C')
  D = dg.descendant(A + B - 3 * C + dg.noise(), name='D')
  ```

- ```
  A = dg.normal(mu=0.5, sigma=0.5, name='A')
  B = dg.normal(mu=0.5, sigma=0.5, name='B')
  C = dg.descendant(A + B, name='C')
  D = dg.descendant(A + B - 3 * C + dg.noise(), name='D')
  ```

We generated a total of 10,000 samples using this graph. Next, with capture decorrelation, we compute the decorrelated feature C_prime.
  ```



(a) Structure of Graph #1, #2 and #3



(b) Structure of Graphs with noise

2.2 Decorrelation

We trained a NN model using Keras library. The model architecture was defined as follows [1]:

```
keras.Sequential([
        keras.layers.Dense(10, input_shape=[2], activation='relu'),
        keras.layers.Dense(1)
        ])
```

The goal of this model was to capture the correlation between features A, B, and C.
Next, we computed a new set of values for feature C, which we refer to as C_prime, for all samples in the dataset. Specifically, we computed C_prime as:

$$C' = C - M(A, B)$$

This allowed us to decorrelate feature C from features A and B in the dataset, and thus investigate the impact of feature decorrelation on model stability.

2.3 Trainings

We conducted experiments using two stochastic models from Keras and SciKit-Learn libraries to evaluate the impact of feature decorrelation [1]. Specifically, we used:

- Stochastic Linear Regression:

  ```
  keras.Sequential([Dense(1, input_dim=3, activation='linear')])
  ```

- Stochastic Decision Tree:

  ```
  sklearn.tree.DecisionTreeRegressor(splitter="random", random_state=seed)
  ```

To evaluate the performance of the Linear Regression and Decision Tree on the dataset, we trained both models with 25 different seeds. For each seed, at the end of training, we extracted the coefficients (feature importances) of different training sessions and computed the distance of the these from the mean coefficients using the L1 and L2 distance metrics.

3 Results

Our analysis, presented in the table, provides compelling evidence that the ratios of the average L1 and L2 values consistently fall below 0.35 for SDT and 0.94 for SLR. This consistent pattern strongly suggests that the implementation of feature decorrelation brings about a significant enhancement in training stability.

Remarkably, this improvement is particularly notable when utilizing the SDT (Stochastic Decision Tree) model. The combination of feature decorrelation and SDT methodology yields remarkable results in terms of training stability.

The great discovery is that this beaviour is also confirmed in presence of noise.

Graph	Model	Ratio L1 C'/C	Ratio L2 C'/C	R2 score C	R2 score C'
Graph 1	SLR	0.499974	0.864705	0.999999	0.999969
Graph 1	SDT	0.096018	0.097270	1.0	1.0
Graph 2	SLR	0.568582	0.864672	0.999696	0.991859
Graph 2	SDT	0.174069	0.170915	1.0	1.0
Graph 3	SLR	0.785187	0.931922	0.989710	0.956236
Graph 3	SDT	0.355453	0.339534	1.0	1.0
Graph w/ noise	SLR	0.498876	0.863772	0.888754	0.888872
Graph w/ noise	SDT	0.089377	0.091166	1.0	1.0
Graph w/ noise 2	SLR	0.568750	0.863956	0.668298	0.663281
Graph w/ noise 2	SDT	0.162088	0.165554	1.0	1.0

Table 1: Metrics comparison for different graphs and models.

4 Error Analysis

We also aimed to explore the potential impact on other performance indicators, specifically focusing on the train R2 loss metric. To conduct this analysis, we collected the R2 loss score of the predictions for each seed and calculated the average for each model.

The results of our investigation indicate that the implementation of feature decorrelation leads to a slightly increase in the training loss for SLR. This is evident from the finding that the C_prime R2 score is consistently lower or equal than the C R2 score. It is important to note this and carefully consider it during a comprehensive cost-benefit analysis.

5 Conclusion

In conclusion, our analysis reveals evidence supporting the effectiveness of feature decorrelation in enhancing training stability. Furthermore, the noteworthy finding is that the observed behavior holds true even in the presence of noise, indicating the robustness of feature decorrelation.

However, when examining the train R2 loss metric, we discovered a slight increase in training loss associated with the implementation of feature decorrelation. This disparity should be taken into consideration during a comprehensive cost-benefit analysis.

Overall, the results highlight the positive impact of feature decorrelation on training stability and performance, with the warning of a potential increase in training loss that needs to be carefully evaluated.

References

[1] Fabian Vincenzi Davide Perozzi. Feature decorrelation. https://colab.research.google.com/drive/1vCICGGyfzNFQXEtJT6qUlDx7Sc-ADeyy, 2023.