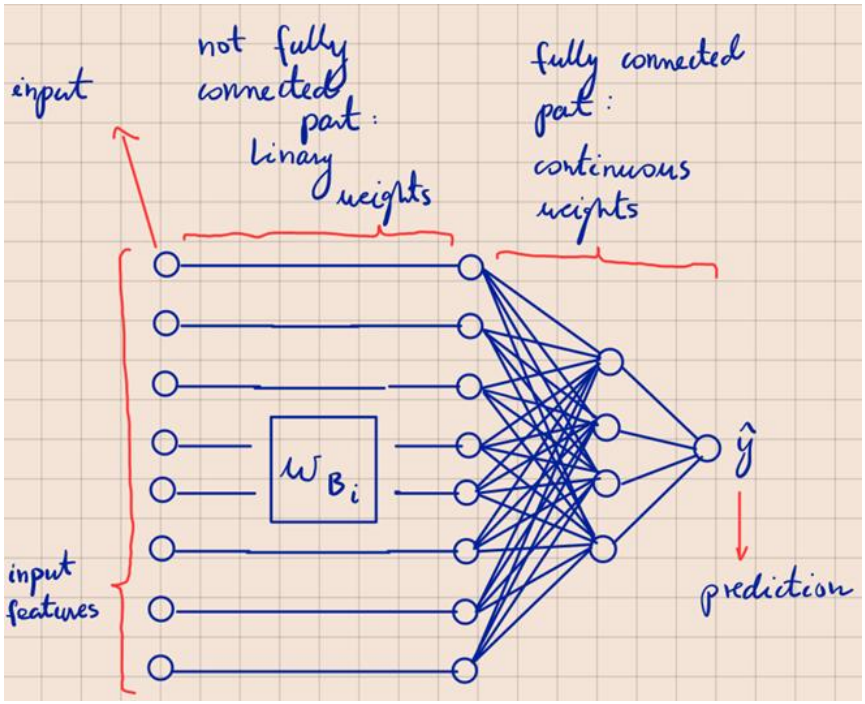
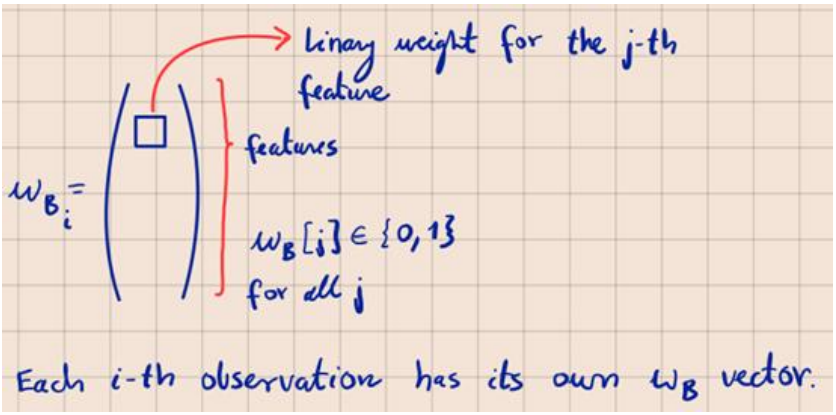
Motivation

All the known feature selection methods have a predefined number of features/dimensions which will be selected. For example, in PCA, tSNE or UMAP we define how many dimensions we want to select. In autoencoder-based feature selection, the dimensionality of the selected features is predefined by the architecture of the auto-encoder. As can be seen, there is no method which learns the number features based on data.

Method



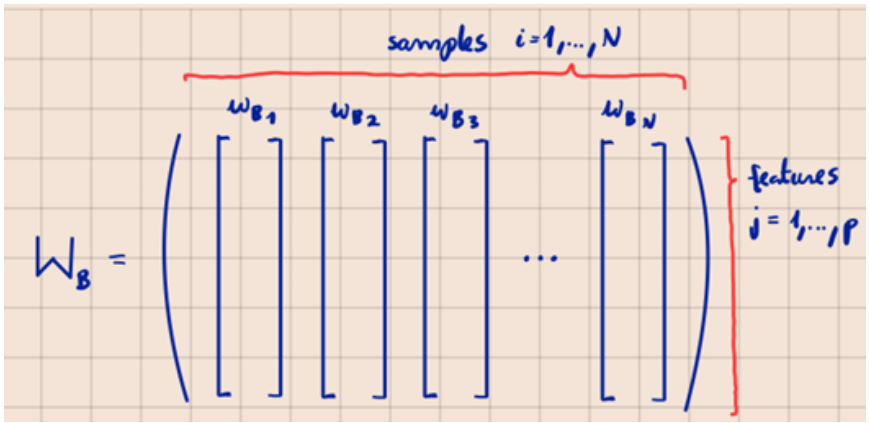
Make a Neural Network where the input and the 1st hidden layer are not fully connected (they are connected in the way as can be seen on the figure). This NN has as many input nodes as is the number of our original features. The 1st hidden layer has also the same number of nodes as the number of original features. The weights between the input and the 1st hidden layer are binary (denoted as wbi).



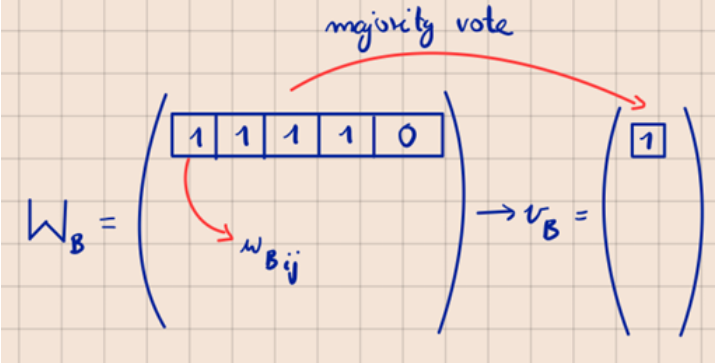
Notation

- i = 1,..,N → index for the sample
- j = 1,...,p → index for the feature

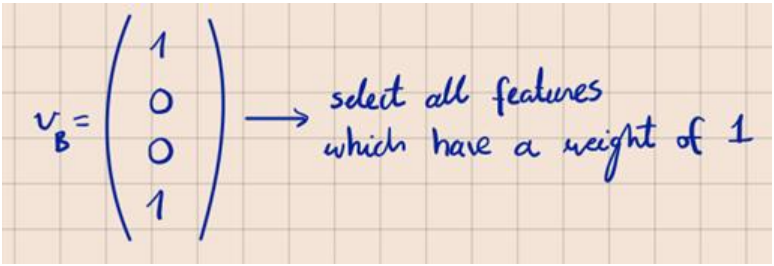
If wbij = 1 then the j-th feature for the i-th sample is included in the reduced features space, if wbij = 0, then the j-th feature for the i-th sample in not included in the reduced feature space. Each sample i has its own binary vector wbi, because the NN is trained using stochastic gradient descent and for example feature j might be important for sample i but not important for sample i+1.



So, when we sweep through all samples (i=1,..N), then all the wbi vectors will be concatenated into the binary Wb matrix of dimensions p x N.



On the figure above we focus on the 1st feature for all samples. As can be seen, for samples 1,2,3,4 the 1st feature was important, but for the 5th sample, the 1st feature was not important. But we need to decide in general if the 1st feature should be part of the reduced feature space or not. We do this by doing a majority vote. So, in this example, since we have more 1s than 0s, the 1st feature will get a weight of 1, setting it as important. The final weights for each feature will be stored in the vb binary vector.



When the vb vector is ready, we just select all the features for which vbj = 1. This set of features will be our reduced feature space.

Loss function

The loss function can't be a simple loss function that favors the highest prediction accuracy. Otherwise, it would lead the model to retain all its original features to yield the most accurate predictions. Therefore, our loss function must also have a penalty for the number of features. So, the loss function must be minimized by such a set of weights for which the highest prediction accuracy with the minimum number of features is achieved.