A Feature Selection Layer Integrated Within A Deep Learning Framework

Alan Joseph Bekker 1 Jeremie Dreyfuss 1

Abstract
In a standard Industrial Internet-of-things (IoT) setting where large amount of sensors are monitoring a single process multiple problems may arise. Mis-relevancy between the feature variables and the target, large redundancy in different sensors and missing data points increase the importance of applying a feature selection phase. In our work we present a method that allows to train a deep neural network model and simultaneously learn which features are informative in the process. Our contribution differs from previous methods in a sense that the features selected don’t relay on any imposed non-linearity. Moreover, our method proposes to learn the neural net parameters and select the features in a single end to end training methodology.

1. Introduction
When solving a machine learning problem, feature selection methods may play a crucial role in the success of the endeavor. Reducing the number of features will decrease the processing power and storage requirements in the learning phase and might allow the inference process on the edge to become feasible (Guyon & Elisseeff, 2003). Reducing the feature space to a smaller dimensionality often allows to lessen the amount of noise entering the modeling phase, in classification or prediction scenarios. Moreover, obtaining sparse models occasionally make them easier to interpret thus providing valuable insights on the modeled data and on the source of modeling errors. Rich literature (Chandrashekar & Sahin, 2014) on feature selection exist in case where the relationship between the features and the target variable is linear whereas substantially fewer works have been done where the relation is assumed to be non-linear. While standard feature selection techniques such as Pearson correlation (Hall, 2000), mutual information (Estévez et al., 2009) and MRMR (Peng et al., 2005) are used for the ranking of the data attributes, thus helping the analyst to decide which features should be part of the second stage of modeling. The more sophisticated wrapper methods such as elastic-net (Zou & Hastie, 2005) incorporate the feature selection process as part of the model learning stage. However none of the methods mentioned above try to apply feature selection under non-linear dependency assumptions between the features and the target variable. (Weston et al., 2000) introduced a method feature selection method for Support Vector Machines which they based on finding the selected features that minimize bounds on the leave-one-out error. They showed that this method could be learned via a standard gradient ascent optimization. Applying the kernel trick de-facto enabled to select features under non-linear assumptions but only under the restriction of the used kernel function. (Verikas & Bacauskiene, 2002) Presented a neural network based approach for identifying informative features for classification in neural network models. The proposed method suggested to add an augmented cross-entropy error function which would force the neural network to keep low derivatives, thus reducing the output sensitivity to the input changes. The feature selection method they proposed is based on a cross-validation procedure which precludes from solving the classification problem and selecting the correct features together. In our work we present a method that allows to train a neural network model both for classification or regression problems and simultaneously learn which features are informative in the process. Our contribution differs from previous non-linear methods for feature selection two-folds. First, our framework could be applied to any type neural network topology which according to Universal approximation theorem (Hornik et al., 1989) can approximate any function thus relaxing the type of non-linearity imposed by the kernel trick (Bauer et al., 2010). Second, in contrast to (Verikas & Bacauskiene, 2002) our method proposes to learn the neural net parameters and select the features in a single end to end methodology. The rest of the paper is organized as follows, first we describe the problem setup which gave us the motivation for this work, second we derive our proposed method then we describe the data we used to evaluate our algorithm and finally we presented the obtained results.
Assume we want to train a multi-class neural-network soft-classifier $p(y = i|x; w)$ where $x \in \mathbb{R}^d$ is the feature vector and $w$ is the network parameter-set. We further assume that in the training phase we have a large amount of features and we need to select the most informative among them. In a classical Neural Network we would like to maximize the log likelihood function:

$$S(w, b) = \log p(y|x; w)$$  \hspace{1cm} (1)

were $p(y|x, w)$ is the soft-max function:

$$p(y = j|x; w) = \frac{\exp(hw_{0j})}{\sum_{l=1}^{y} \exp(hw_{0l})}$$  \hspace{1cm} (2)

Here, we denote the neural network non-linear function applied to the input $x$ by $h = h(x)$ which could be modeled by any type of network such as convolutional, recurrent or fully connected.

In our proposed model we add a feature selection layer such that the input of the network is the element-wise product between the feature vector $x$ and the feature selection layer weights $w_{fs}$.

$$h_0 = w_{fs} \ast x = [w_{fs}^1 x_1, w_{fs}^2 x_2, ..., w_{fs}^d x_d]$$  \hspace{1cm} (3)

Since our target is to maximize the log likelihood under the restriction that our model will have high input sparsity we add the lasso regularization term only to the feature selection layer as can be seen in the following cost function:

$$L(w, b, w_{fs}) = \sum_t log p(y_t|x_t; w, b) - \frac{\lambda}{d} \sum_i |w_{fs}^i|$$  \hspace{1cm} (4)

were $\lambda > 0$ is the regularization multiplier and $d$ is the number of NN input features.

Since we proposed a new type of layer we derive only the derivative of the cost function in respect to the FS layer. The partial derivatives of (4) respecting to the parameters of the FS layer are:

$$\frac{\partial S}{\partial w_{fs}} = \frac{\partial S}{\partial h_0} \frac{\partial h_0}{\partial w_{fs}} = \frac{\partial S}{\partial h_0} x$$  \hspace{1cm} (5)

were $h_0$ stands for the output of the first layer.

In the inference phase we use a threshold on the learned weights of the Feature Selection layer in order to determine which features will be removed of the model.

$$x \xrightarrow{w_{fs}} h_0 \xrightarrow{\text{ FS layer}} \xrightarrow{\text{neural-net}} y$$

Figure 2. Our proposed model consist of a FS layer defined by $h_0 = w_{fs} \ast x$ followed by any type of neural-net

### 4. Experiments

Due to Intel data policy restrictions we can’t publish our obtained results on our IoT scenario, instead we illustrate in this section our results obtained on MNIST data.
A Feature Selection Layer Integrated Within A Deep Learning Framework

Table 1. The feature selection deep-learning algorithm.

| Input: Data-points $x_1, ..., x_n \in \mathbb{R}^d$ with labels $y_1, ..., y_n \in \{1, ..., k\}$. |
| Output: Neural-network parameters $w$ and binary mask of features selected $w_{fs}$. |

The feed forward Algorithm:

- $h_0 = w_{fs} \ast x = [w_{fs}^1 x_1, w_{fs}^2 x_2, ..., w_{fs}^d x_d]$
- For $i=1, ..., L$
  - $z_i = w_i h_{i-1} + b_i$
  - $h_i = g(z_i)$
- $z_{oj} = w_{oj} h_L + b_{oj}$
- $\hat{y}_j = p(y = j | x; w, b) = \frac{\exp(z_{oj})}{\sum_{l=1}^{k} \exp(z_{ol})}$

and train a NN to find $w, w_{fs}$ that maximizes the following function:

$$L(w, b, w_{fs}) = \sum_t \log p(y_t | x_t; w, b) - \frac{\lambda}{d} \sum_i |w_{fs}^i|$$

trained a binary neural classifier derived form the MNIST 10 digits data. We took only the digits with labels '0' and '1' as can be seen in figure 4.

We trained the LeNet-1 topology composing of 4 convolutional hidden layers with dropout of 0.5. As described above, we optimize the cost-function in eq.4. We used sgd with momentum. Our initial learning-rate was set to 0.001. The code was implemented in theano.

The illustration we made in this section is visualizing the Feature Selection layer as function of the $\lambda$ parameter. The $\lambda$ tunes the aggressiveness of the selected features. As can be seen in 4 as far as we increase the $\lambda$ from 0.1 to 0.4 the number of pixels( features) needed to classify between the digits 0 and 1 is decrease.

5. Conclusions

In this work we presented a novel method for feature selection integrated within a neural network framework classifier. We introduced a feature selection layer which add a penalty term to the cost function using a Lasso regularisation method, thus enabling the end to end training as part of the regular optimization procedure. The proposed method specially fits to a IoT setting in which large amount of sensors monitor a single process and we would like to reduce the number of features for modeling. We illustrated the obtained results on a binary classification task derived from the MNIST digits dataset. We showed that as far as we increased the regularization penalty parameter the selected image features were reduced in a form of the intersection between the digits zero and one.
References


