LIFFTing the Top off the Black Box: Feature Selection for Neural Networks in Biology

Abstract

A common criticism of neural networks is their lack of interpretability or "black box" nature. This is a major barrier to adoption in application domains such as biology, where interpretability is important. Here we present DeepLIFFT (Learning Important Features From Training), a simple yet effective method for scoring the relative contributions of raw input features to the output of a deep neural network. DeepLIFFT leverages the piece-wise linear nature of common activation functions such as Rectified Linear Units (ReLUs) to decompose the inputs to a softmax classifier as a linear sum of terms corresponding to the raw features. It then applies a heuristic to score and rank features based on their importance for correctly classifying a user-defined subset of input examples. We apply DeepLIFFT to a key problem in functional genomics involving the identification of DNA sequence features that are predictive of different classes of genomic regulatory elements. We show that DeepLIFFT performs favorably compared to feature selection with L1 regularization and Random Forests. We further show that DeepLIFFT selects biologically meaningful features and can be used to obtain additional insights about sequence grammars encoded in regulatory elements.

1. Introduction

As neural networks become increasingly popular, their reputation as a "black box" presents a barrier to adoption in fields were interpretability is paramount. Existing approaches, such as those which identify the input that maximally activates a given neuron (Le et al., 2012), are geared towards image classification and require numerical optimization. Furthermore, while much research in the field has focused on unsupervised feature discovery, feature selection at the input level has not been well-studied, even though it is very valuable in fields such as biology.

In this paper, we assess the relevance of each input feature to the predicted class probabilities. In particular, we decompose the class inputs to the softmax probability function into linear combinations of input features, which is viable when the hidden unit activation functions consist of two piecewise linear regimes such as the Rectified Linear Unit (ReLU). Using this decomposition, we quantify the contribution from each feature to the final class probabilities using a heuristic; this is the DeepLIFFT feature importance score for an individual feature and input example. Finally, we aggregate these scores over a set of examples to obtain overall feature importance rankings.

2. DeepLIFFT Method

We study the case of neural network architectures where the hidden units are either pooling operations or contain two linear regimes, and the output layer is a softmax. First, we demonstrate how the inputs into the softmax layer can be decomposed into a linear combination of the raw features plus biases.

2.1. Linear Decomposition

Consider the case where the activation function $f(z)$ consists of two linear regimes:

$$f(z) = \begin{cases} a_1 z + b_1 & \text{if } z < z' \\ a_2 z + b_2 & \text{if } z \geq z' \end{cases}$$

We will use induction. Assume z can be decomposed into a linear combination of the raw features; let x_i denote raw feature i. If for some b and c_i,

$$z = b + \sum_i c_i x_i$$

then if we define

$$b' = \begin{cases} b_1 + a_1 b & \text{if } z < z' \\ b_2 + a_2 b & \text{if } z \geq z' \end{cases}$$

$$c_i' = \begin{cases} a_1 c_i & \text{if } z < z' \\ a_2 c_i & \text{if } z \geq z' \end{cases}$$

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

We have:

$$f(z) = b' + \sum_i c'_i x_i$$

Thus, if the input z can be decomposed into a linear combination of the raw input features plus a bias term, so can $f(z)$. We further note that in the hidden layers, the input z is a linear combination of the activations of the previous layer. Hence, we can conclude that if all activation functions consist of two piecewise linear regimes, at any layer the activation of a particular neuron can be decomposed into a linear combination of the original raw inputs.

In the case where the activation is a max-pool, we set the decomposition of the output to be the same as the decomposition of the maximum input. If the activation is an average pool, we average the coefficients c_i and bias terms b over all the inputs to get the decomposition.

2.2. Per-Input Feature Importance

Consider the multi-class classification case with a softmax output layer. Let the inputs into the softmax node for training example n be denoted z_j, where j is the class index. The softmax activation is defined as:

$$\sigma(\mathbf{z})_\mathbf{j} = \frac{\mathbf{e^{z_j}}}{\sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{e^{z_k}}}$$

Where K is the total number of classes. Let the correct class be j'. Also, let the linear decomposition of z_j be represented as:

$$z_j = b^{(j)} + \sum_i c_i^{(j)} x_i$$

The intuition for the approach is as follows: we sort the terms $c_i^{(j')} x_i$ of the correct class in descending order to get an ordering over the input features i. We then incrementally include the terms for the raw features according to this ordering and compute how much the softmax probability for the correct class changes with each term.

Let the vector representing the ordering of these feature indices after being sorted in descending order of $c_i^{(j')} x_i$ be called F. Let F_l represent the feature at index l of F. Define z_j^l as a quantity that considers only the terms corresponding to the raw features up to position l in F:

$$z_j^l = b^{(j)} + \sum_{l'=1}^{l} c_{F_{l'}}^{(j)} x_{F_{l'}}$$

Also define $\sigma(z)_j^l$ to be the softmax probability when only these terms are considered:

$$\sigma(z)_j^l = \frac{e^{z_j^l}}{\sum_{k=1}^{K} e^{z_k^l}}$$

The feature importance score $\phi_{F_l}^n$ for training example n and raw input F_l with correct label j' is:

$$\phi_{F_l}^n = \begin{cases} \sigma(z)_{j'}^l - \sigma(z)_{j'}^{l-1} & \text{if } l > 1 \\ \sigma(z)_{j'}^l - \frac{e^{b^{j'}}}{\sum_{k=1}^{K} e^{b^k}} & \text{if } l = 1 \end{cases}$$

For clarification, when $l = 1$, we simply consider what the change in the softmax probability is when the term corresponding to feature F_l is included, as compared to the probability when only the bias terms are considered.

Note that it is straightforward to adapt the method to assess the importance of individual neurons within the network by treating the hidden layer of the neuron of interest as though it is the input layer.

2.3. Aggregate Feature Importance

We can now compute two types of scores. If the goal is interpretability, we compute a feature importance for F_l for a specific class c by averaging the scores $\phi_{F_l}^n$ over all correctly classified inputs corresponding to class c; this will reveal how relevant a particular feature is for correctly classifying c. Formally, if $y(n)$ denotes the correct class of example n and $h(n)$ denotes the output of the neural net on example n, then the class-specific feature importance score $\phi_{F_l}^{(c)}$ is:

$$\phi_{F_l}^{(c)} = \frac{\sum_{n:y(n)=h(n)=c} \phi_{F_l}^n}{\sum_n 1\{y(n) = h(n) = c\}}$$

If the goal is to choose a subset of features that will give high classification accuracy, we can compute an overall feature importance Φ_{F_l} for feature F_l by averaging the absolute values of the scores $\phi_{F_l}^n$ over all correctly classified inputs. Formally:

$$\Phi_{F_l} = \frac{\sum_{n:y(n)=h(n)} |\phi_{F_l}^n|}{\sum_n 1\{y(n) = h(n)\}}$$

2.4. A note on taking the absolute value

A negative feature importance score indicates that a particular feature contributed adversely to the correct classification of the input example. This might lead one to think that to compute the overall feature importance score Φ_{F_l}, we should average the signed scores, rather than averaging the absolute values of the scores. However, consider the following situation: your inputs fall into classes A, B or C, and you have features 1 and 2. Class A is distinguished by the absence of feature 1 and feature 2. Class B is distinguished by the presence of feature 1 but not feature 2, and class C is distinguished by the presence of feature 2, but also occasionally has feature 1 present. Whenever feature 1 is present for class C, it will likely receive a negative

score because the occurrence of feature 1 favors classification as B. Thus, if we average the score of feature 1 over all training examples, we might obtain a net score near 0, even though feature 1 is important. The key is to realize that the only reason feature 1 obtains a negative score even on correctly classified examples from class C is because it is actually important for classification into a different class - class B. This is the justification for taking the absolute value, and also for restricting our attention to examples that are correctly classified. Note that when we compute feature importance scores $\phi_{F_l}^{(c)}$ for a specific class, we do not need to take the absolute value.

3. Data and Implementation

3.1. Data

The human genome encodes a large number of regulatory elements that control the activation and repression of genes. These regulatory elements can be broadly classified into 8 functional classes namely Promoters, Enhancers, Pure CTCF, Transcribed, Heterochromatin, Bivalent elements, Repressed Polycomb elements or Quiescent elements. Identifying the DNA sequence features that encode these distinct functional classes of elements is an important problem in functional genomics. We consider the problem of classifying regulatory elements in the GM12878 lymphoblastoid cell lines into the 8 functional classes. The regulatory elements and their labels were obtained using a method called ChromHMM (Ernst and Kellis, 2012) that combines multiple genome-wide tracks of biochemical marks to decipher and annotate regulatory elements. We study either the 8-class classification problem (where the classes are Promoter, Enhancer, Pure CTCF, Transcribed, Heterochromatin, Bivalent, Repressed Polycomb or Quiescent) or a one-versus-all classification problem (where the positive class is one of Promoter, Enhancer or Pure CTCF). The features we use are the counts of 512 sequence motifs (representing binding sites of regulatory proteins) in 2kb region centered around the regulatory element. When we apply logistic regression, the counts are normalized to be in the range 0 to 1. There are 54,495 examples (elements) in the training set, 13,577 in the validation set and 16,871 in the test set. The test set contains elements from a chromosome not used to construct the training or validation sets.

3.2. Network architecture

In every case described, the network architecture used consisted of two fully connected hidden layers, the first with 200 ReLUs and the second with 100 ReLUs. The final layer was a softmax output layer. The net was constructed using pylearn2 (Goodfellow et al., 2013), and was trained with an initial learning rate of 0.01 and momentum 0.1. The batch size was 100.

In each task, the net was trained once using all input features, and DeepLIFFT was performed on the validation set. The rankings obtained from DeepLIFFT were then used to restrict the feature space for various downstream algorithms in the benchmarking.

4. Benchmarking

4.1. Comparison with L1 regularization

To assess performance, we compared the features chosen by DeepLIFFT with the features chosen by L1 regularization for a logistic regression classifier. Since logistic regression is a binary classification task, we considered the three tasks "Promoter vs. All", "Enhancer vs. All" and "Pure CTCF vs. All". We trained a neural network on the input data and applied DeepLIFFT to obtain a feature ranking as described above. We then compared the performance of logistic regression on the top n features from DeepLIFFT (and no regularization) with the performance of logistic regression when only n weights are nonzero due to L1 regularization. As shown in Figure 1, DeepLIFFT outperforms L1 regularization when the number of features to be considered is small. This is particularly striking because the features selected by DeepLIFFT are intended to work with a neural network. Also shown in Figure 2 are the top 13 features selected by DeepLIFFT and by L1 regularization for the "Promoter vs. All" task.

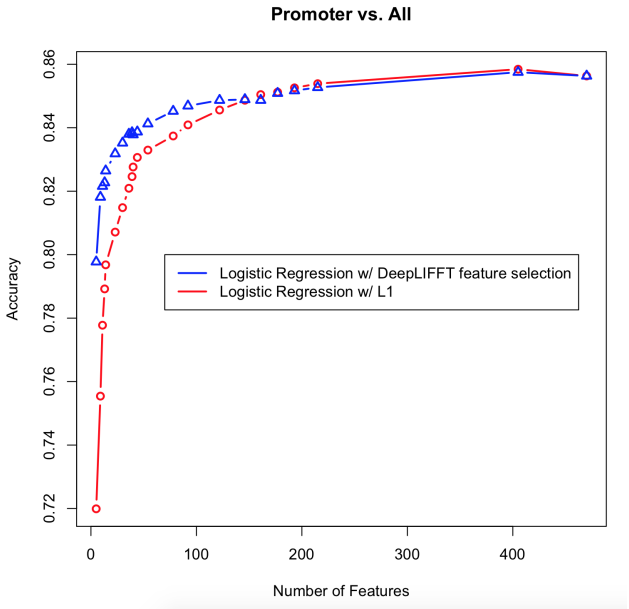

Figure 1. Performance of logistic regression with L1 and DeepLIFFT features on "Promoter vs. All" classification task. When the neural network is trained on this task, it obtains an accuracy of 87.1%. The plots for the "Enhancer vs. All" task and the "Pure CTCF vs. All" tasks look similar.

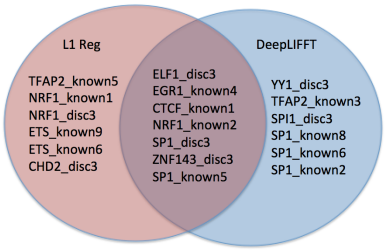

Figure 2. Top 13 features for "Promoter vs. All" classification selected by L1 regularization (red) and by DeepLIFFT (blue)

4.2. Comparison with Random Forest

We also benchmarked against a random forest. When the random forest was restricted to features chosen according to the DeepLIFFT rankings, it performed comparably to when it was restricted to features chosen according to the random forest's own feature importance scores. However, the converse was not true; when the neural network was restricted to the features chosen according to DeepLIFFT, it performed better than when restricted to the features chosen according to the Random Forest's importance scores, as shown in Figure 3.

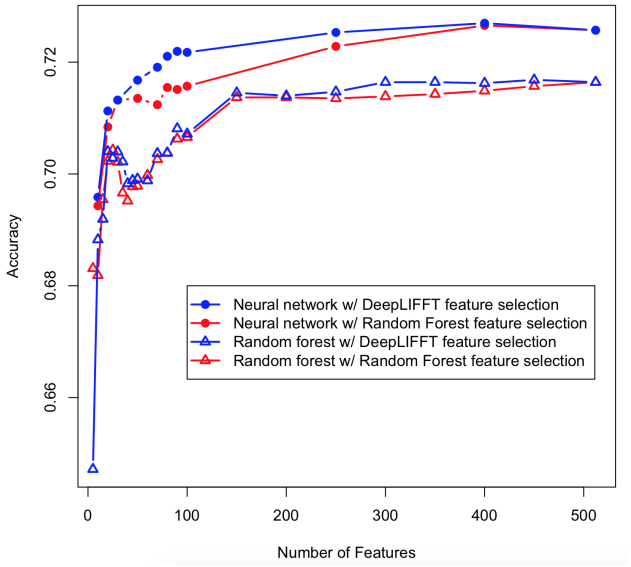

Figure 3. Performance of Random Forest and Neural Network on the 8-class classification task, with features chosen either according to the random forest or according to DeepLIFFT.

Furthermore, it is worth noting that the class-specific feature importance scores of DeepLIFFT are more biologically interpretable than the scores generated by a Random Forest. As an example, consider the top 5 features picked by the Random Forest for the "Pure CTCF vs. All" classification task shown in Table 1. The Pure CTCF elements should be exclusively enriched for motifs of CTCF and its primary co-factors namely RAD21; intuitively one would not expect to see the YY1_disc3 motif, but the Random Forest includes this feature because it is effective at identifying negative examples (Promoters in this case; looking at the YY1_disc3 motif - which is not the primary YY1 motif but was discovered at YY1 binding sites - we see that it is likely identifying CpG islands which are known to be enriched at Promoters). Unlike the Random Forest feature importance scores, the DeepLIFFT feature importance scores reveal that YY1 is actually negatively associated with the Pure CTCF class.

Motif	Logo	RF score	DL score	DL rank
CTCF_known1		0.183	0.193	1
CTCF_known2		0.087	0.060	2
YY1_disc3		0.008	-0.001	486
CTCF_disc3		0.008	0.005	4
RAD21_disc3		0.007	0.004	6

Table 1. Top 4 features chosen by Random Forest in "Pure CTCF vs. All" classification task, and corresponding DeepLIFFT class-specific scores and rank for the "Pure CTCF" class. There are 512 motifs in total. Note how the DeepLIFFT class-specific score reveals that YY1_disc3 is negatively associated with the Pure CTCF region; the Random Forest scores provide no such insight. Also keep in mind that the class-specific scores are distinct from DeepLIFFT's overall scores. The class-specific scores have the goal of interpretability; the overall scores have the goal of high performance with fewer features

5. Biological Interpretability and Correctness

As shown at the end of the previous section, a major advantage of DeepLIFFT is its ability to generate scores that allow the user to see which features are important for a specific class. For this task, we generated class specific scores for the Enhancer, Promoter and Pure CTCF classes, and our findings were consistent with known biology. The class-specific scores for the Enhancer, Promoter and Pure CTCF classes generated in the 8-class classification task are shown in Figures 4-6.

The top ranked motifs belong to DNA binding regulatory proteins that are known to be associated specifically with the respective classes of elements in lymphoblastoid cell-lines and B-cells. E.g. SPI1, PAX5, IRF and STAT are critical B-cell differentiation factors that have been shown to primarily bind distal enhancer elements using in-vivo protein-DNA binding experiments (Gerstein et al., 2012).

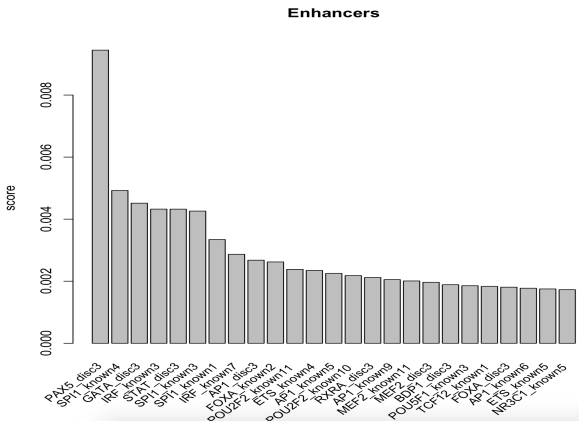

Figure 4. DeepLIFFT top ranked motifs for the Enhancer class.

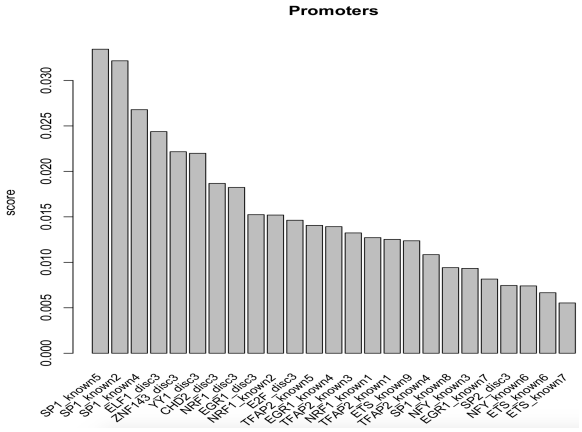

Figure 5. DeepLIFFT top ranked motifs for the Promoter class

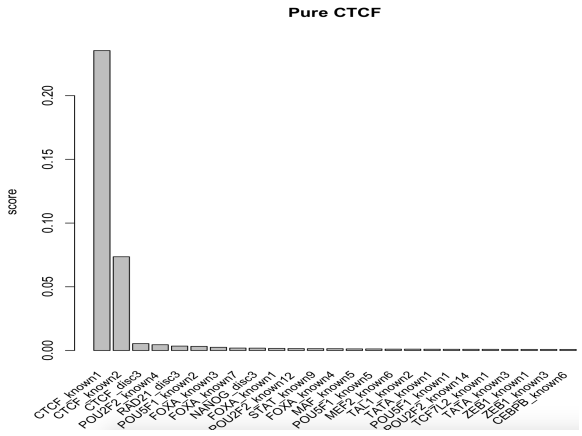

Figure 6. DeepLIFFT top ranked motifs for the Pure CTCF class.

On the other hand SP1, YY1, ELF, ETS and NRF have been shown to have a strong promoter bias in in-vivo binding experiments (Gerstein et al., 2012). As additional confirmation, we computed the proportion of in-vivo binding regions that fell within 2kb of a promoter; we found that our predicted promoter-associated proteins had a greater proportion of promoter-proximal sites compared to our predicted enhancer-associated proteins (EGR1, SP1 and ELF from the promoter-associated list have 67%, 51% and 63% of their sites near a promoter; PAX5, SPII and IRF have 36%, 20% and 22% respectively). Our results indicate that DeepLIFFT is able to precisely distinguish these class-specific features.

6. Revealing Input Heterogeneity

The ability to generate importance scores for individual features allows us to detect whether a particular class in our softmax output is actually comprised of multiple subclasses. We demonstrate this by doing k-means clustering on the feature importance scores of the top 10 motifs associated with the Enhancer class for all the correctly classified Enhancers in the full dataset. As shown in Figure 7, clustering these elements according to the DeepLIFFT feature importance scores reveals novel regulatory heterogeneity i.e. subclasses of enhancers with distinct sequence gram-

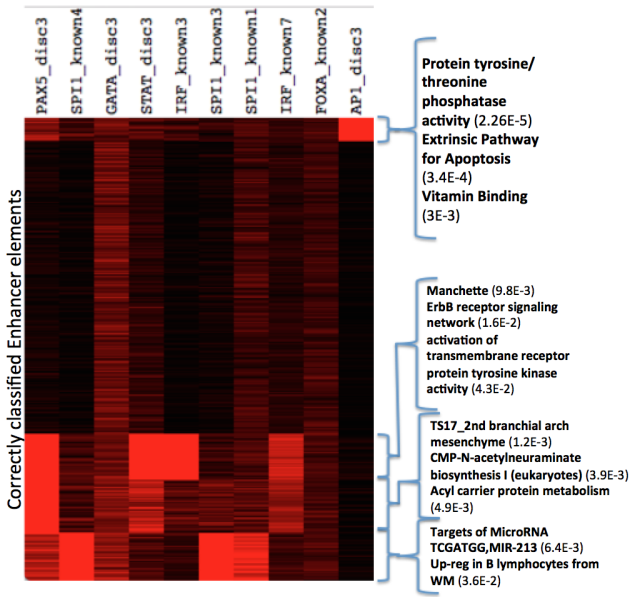

Figure 7. K=5 means clustering of feature importance scores for the top 10 Enhancer-associated motifs. Each row is a correctly classified Enhancer from the full dataset. Red indicates a high feature importance score. On the right are the top 3 terms produced by GREAT, sorted by the hypergeometric test FDR (shown in parentheses)

mars, such as an AP1_disc3 enriched class and an SPI1-enriched class. We also note interesting combinatorics with the PAX5_disc3 motif; the fourth cluster from the top is primarily enriched only for PAX5_disc3, while the third cluster from the top is enriched for PAX5_disc3 in conjunction with STAT_disc3 and IRF_known3. On the right we also list the top functional enrichments for genes associated with the regulatory elements in the cluster, obtained using GREAT (McLean et al., 2010). The background input into GREAT was the list of correctly classified Enhancers in our dataset.

7. Discussion

We showed that DeepLIFFT, while conceptually simple, is nevertheless an effective method of understanding which of the raw features are important for the accurate classification of a given input into the neural network. We further showed that the feature importance scores generated by DeepLIFFT can be used to do feature selection for other classifiers and can yield better results than L1 regularization. Finally, we have demonstrated that DeepLIFFT produces biologically meaningful results when applied to a real-world dataset, and that the feature importance scores for individual inputs can themselves be used in conjunction with techniques such as clustering to yield further insights about the data.

Future work would involve comparing DeepLIFFT with the results of stimulus optimization and investigating whether the central idea can be applied to domains such as image classification. It would also be interesting to see if the feature importance scores can be meaningfully computed even when the intermediate hidden layers involve activation functions that do not consist of two piecewise linear regimes - a linear decomposition may no longer be simple, but a heuristic might give sufficiently accurate results.

8. References

Le, Q., Ranzato, M., Monga, R., et al. (2012). Building high-level features using large scale unsupervised learning. Retrieved May 1, 2015, from http://arxiv.org/abs/1112.6209

Goodfellow, I., Warde-Farley, D., Lamblin, P., et al. (2013). Pylearn2: a machine learning research library http://arxiv.org/abs/1308.4214

Ernst J, Kellis M. ChromHMM: automating chromatin-state discovery and characterization. Nat Methods. 2012;9(3):215-6.

Gerstein MB, Kundaje A, Hariharan M, et al. Architecture of the human regulatory network derived from ENCODE data. Nature. 2012;489(7414):91-100.

Mclean CY, Bristor D, Hiller M, et al. GREAT improves functional interpretation of cis-regulatory regions. Nat Biotechnol. 2010;28(5):495-501.