Using Feature Selection to Find Inputs that Work Better as Extra Outputs

Rich Caruana
Just Research and
Carnegie Mellon University
Pittsburgh, PA 15213
caruana@cs.cmu.edu

Virginia R. de Sa
Sloan Center for Theoretical
Neurobiology, Dept. Physiology
UCSF, San Francisco CA 94143-0444
desa@phy.ucsf.edu

Abstract

In supervised learning there is usually a clear distinction between inputs and outputs — inputs are what you measure, outputs are what you predict from those measurements. The distinction between inputs and outputs is not this simple. Previously, we demonstrated that on synthetic problems some input features are more useful when used as extra *outputs* than when used as inputs[6]. This paper shows the same effect on a real problem, and presents a means of determining what features can be used as extra outputs. We show that the feature selection method devised by Koller and Sahami[11] can be used to select features to use as extra outputs, and that using some features as as extra outputs instead of as inputs yields better performance on the DNA splice-junction domain.

1 MOTIVATION

The goal in supervised learning is to learn functions that map inputs to outputs with high predictive accuracy. The common practice in backprop nets is to use all features that will be available for test cases as inputs, and use as outputs only features that need to be predicted. On real problems, where there may be many redundant or irrelevant features, using all the available features as inputs is often suboptimal. Many algorithms learn better given a carefully selected subset of the features to use inputs[3, 10, 11].

If feature selection is used to find the features to use as inputs, what should be done with the features not selected? Usually, features not selected for use as inputs are discarded. But, there are other ways to benefit from features without using them as inputs. One way to benefit from features not used as inputs is multitask learning. Multitask learning (MTL) is an inductive transfer method where extra tasks are learned in parallel with the main task while using a shared representation. Because the extra tasks share a hidden layer with the main task, internal representations learned for the extra tasks can be used by the main task outputs, often improving performance on the main task.

MTL in backprop nets is well documented[13, 1, 2, 4, 8, 9, 7]. Most applications of MTL are to problems where some features available for the training set will not be available for future test cases[5]. We recently demonstrated that there are problems where some features that *could be used as inputs* would be

more useful if used as extra outputs instead[6]. This demonstration was made on problems carefully engineered to demonstrate this effect. It was clear in those problems which features would be more useful as outputs than as inputs. Do real problems have features that would benefit learning more if used as extra outputs than if used as inputs? If they do, how would we determine which features to use as outputs?

This paper shows that the DNA splice-junction domain contains features that yield better recognition accuracy when they are used as extra outputs than when they are used as inputs. We use the feature selection method developed by Koller and Sahami [11] to select which features to use as inputs. They showed naive bayes and C4.5 achieve better accuracy using just the selected features as inputs than using all the features as inputs. We show a similar result for backprop nets. We then show that it is better to use some of the features not used as inputs as extra outputs instead of ignoring them. Best performance is achieved by moving some of the input features to the output side of the net. We suspect other real-world problems would benefit from a similar combination of feature selection and multitask learning.

2 FEATURE SELECTION

In real-world problems often there are many irrelevant and redundant features. Because many learning algorithms have difficulty coping with large numbers of redundant and irrelevant features, performance often improves when the learning method is given only the subset of features most useful for the learning task. Feature selection selects from the available features which ones to use as inputs for learning.

Here we use the feature selection method developed by Koller and Sahami. It uses information theoretic measures of feature importance to select the features most likely to be useful as inputs. The theoretical motivation behind the algorithm is to remove attributes which have Markov blankets in the remaining attributes with respect to the prediction task. If an attribute has a Markov blanket in the other attributes, this attribute provides no additional information and the class decision is independent of the value of this attribute conditioned on the values of the other attributes in the blanket. Finding Markov blankets is not practical given a large number of attributes. The Koller-Sahami algorithm makes several simplifying approximations that allow the degree to which an attribute is blanketed by other attributes be estimated in reasonable time.

The Koller-Sahami algorithm is a greedy feature selector. The preferred way of using it is backward-elimination. Start with all features in the set, and remove attributes one-at-a-time, at each step removing the attribute most covered by the other attributes remaining in the set (i.e., remove the attribute that provides the least additional information for the class given the other remaining attributes).

3 DNA SPLICE-JUNCTION PROBLEM

DNA contains coded information that is used by cells to construct proteins. In the process of building a message RNA template to use as a scaffold on which to build a protein, large sections of the original DNA coding are ignored. The coded sequences that are used are known as "exons", while the ignored sequences are known as "introns". The nature of the boundaries between exons and introns, known as "splice junctions", is a subject of active research.

The DNA Splice-Junction Problem we use is from the UCI machine learning repository[12]. For each case in the database there is a sequence of 60 nucleotides. The goal is to predict if the center of the nucleotide sequence codes for an exon-to-intron boundary, an intron-to-exon boundary, or neither. 25% of the cases are EI boundaries, 25% are IE boundaries, and 50% are neither EI or IE boundaries. For compatibility, we use the nucleotide coding scheme used by Koller and Sahami, which codes each of the 60 nucleotides using 3 bits. This yields 180 boolean attributes that can be used as inputs. Typical performance on this problem is 92-94% accuracy when trained on training sets containing 1000-2000 cases.

4 EXPERIMENTS

We've run two experiments with DNA Splice-Junction. In the first, we determine what size net performs best. In the second, we determine if using some features as extra outputs improves performance. All our experiments use backprop nets composed of sigmoid units. We train the nets using conjugate gradient and use an independent halt set to determine when to stop training. The performance of the net is then measured on an independent test set not used for training or early stopping. The dataset contains 2000 cases. We randomly split this set into train, halt, and test sets containing 667, 666, and 667 cases, respectively. We repeatedly sample the dataset this way to generate multiple trials.

Our coding for the main splice-junction task uses three outputs, one for IE, one for EI, and one for neither. We use a normalized cross-entropy loss function for the outputs for the main task. The classification of the net prediction is done by finding which of the three outputs has the highest activation. When boolean attributes are used as extra outputs, we use a non-normalized cross-entropy loss function to train them.

4.1 Experiment 1: Performance vs. Net Size

The purpose of this experiment is to determine what net size yields best performance. We tried nets containing 5, 20, 80, 320, and 1280 hidden units. The nets have 3 outputs that code for the main task.

Figure 1a shows the test set cross-entropy error for nets of different sizes trained with all 180 input features, and trained with the 30 input features selected by the Koller-Sahami feature selector. (These are the same 30 features

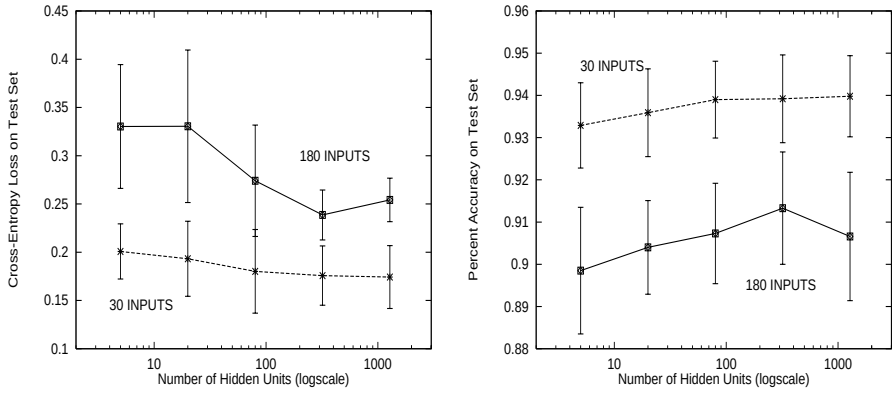

Figure 1: Cross-Entropy (left) and Accuracy (right) of Different Size Nets With All 180 Inputs and 30 Selected Inputs.

Koller and Sahami selected for this problem.) Figure 1b shows the prediction accuracy of the nets. Each point is the average of 12 trials; the vertical bars are 95% confidence intervals.

Better performance is achieved with 80, 320, or 1280 hidden units. Also, performance is significantly better for nets that use only 30 selected features as inputs as compared to nets that use all 180 features as inputs. We conclude from this experiment that the optimal net size is about 320 hidden units, and that training nets with the 30 features selected with the Koller-Sahami feature selector yields better performance than using all 180 inputs.

4.2 Experiment 2: Unused Features as Extra Outputs

In the previous experiment, the 150 features *not* selected for use as inputs were thrown away. In this section we use some of the features not used as inputs as extra outputs for multitask learning.

Because the Koller-Sahami feature selector is a greedy algorithm that removes attributes one-at-a-time and is told how many features to remove, it can be used to order the attributes. We ran Koller-Sahami and had it remove all 180 attributes in the DNA problem. As above, we use the last 30 attributes removed as inputs. Rather than ignore the remaining 150 attributes, we use the *next* 30 attributes as extra outputs for multitask learning. These are the attributes of the remaining 150 that the feature selector considers most useful.

The multitask net has 30 inputs, 3 outputs for the main task, and an additional 30 outputs for 30 attributes not selected for use as inputs. The multitask net has 800 hidden units (instead of 320) because it is learning many more tasks. We have not attempted to find the optimal number of hidden units for the multitask net, and suspect it would perform better with more than 800 hidden units. This biases our experiments in favor of nets that do not use extra outputs because we use a near optimal number of hidden units on those nets.

Table 1 shows the cross-entropy and accuracy for four nets. Net1 has 320 hidden units and uses all 180 inputs. This is a traditional net trained without feature selection. Net2 also has 320 hidden units, but uses only the 30 features selected by feature selection as inputs. Net3 is the MTL net. It has 800 hidden units and uses the same 30 inputs as Net2. Net3, however, also uses the next best 30 attributes as extra outputs. These are attributes ignored by Net2. Net4 has 320 hidden units and uses as inputs both the 30 attributes used by Net2 as inputs, and the 30 attributes used by Net3 as extra outputs. Net4 uses all attributes used by Net3, but Net4 uses them as inputs. It has no extra outputs.

Table 1: Cross-Entropy and Accuracy of Different Combinations of Inputs and Outputs. Net3 is the MTL Net that Uses Some Features as Extra Outputs.

Net	**Inputs**	**XtraOutputs**	**CrossEntropy**	**StdErr**	**Accuracy**
Net1	180	0	0.257	0.006	90.98%
Net2	30	0	0.180	0.009	94.16%
Net3	30	30	0.167	0.006	94.32%
Net4	60	0	0.187	0.006	93.66%

Net1 is clearly the worst performer. Using all 180 attributes as inputs is not the best thing to do. Using only the 30 features selected by Koller-Sahami as inputs (Net2) yields significantly better performance. Net3, however, performs even better. It is best to use some of the features not used as inputs as extra outputs instead of ignoring them. Net4, which uses all features used by Net3 *as inputs,* does not perform as well as Net3 (nor even as well as Net2).[1] *In DNA splice-junction it is better to use some of the features as extra outputs than as inputs. Moreover, in this domain the Koller-Sahami feature selector is an effective way of selecting which features to use as inputs and which features are candidates for use as extra outputs.*

5 DISCUSSION

This paper shows that DNA splice-junction contains features that improve recognition accuracy if used as extra outputs instead of as inputs. This confirms our expectation from previous work that there are real problems where some features could be better used as extra outputs than as inputs. This shows that the benefit of using a feature as an extra output is different from the benefit of using that feature as an input. As an input, the net has access to the feature's values on the training and test cases. As an output, however, the net is instead biased to learn a mapping from the other input features in to that output. What is learned for this mapping is sometimes more useful than the feature value itself, particularly if the value of the feature as an *additional* input is marginal or harmful.

[1] Although there is overlap between the 95% confidence intervals for Net2 and Net3, paired t-tests show Net3 performs better than Net2 at $p = 0.05$. Much of the variance results from different trials. Net3 consistently outperforms Net2.

The Koller-Sahami feature selector does not automatically determine how many features to use as inputs. We use cross validation to find the appropriate number of inputs. We do not know how many of the features not used as inputs should be used as outputs and are currently examining if this number depends on the number of unused features used as extra outputs. We are also trying to develop an *output selection* algorithm that would do feature selection for outputs instead of inputs. We suspect that many real-world problems would benefit from a similar combination of feature selection and multitask learning.

Acknowledgements

R. Caruana was supported by ARPA grant F33615-93-1-1330, NSF grant BES-9315428, and Agency for Health Care Policy grant HS06468. V. de Sa was supported by post-doctoral fellowship from the Sloan Foundation. We thank the University of Toronto for the Xerion Simulator, and D. Koller and M. Sahami for their feature selector.

References

[1] Abu-Mostafa, Y. S., "Learning from Hints in Neural Networks," *Journal of Complexity,* 1990, 6(2), pp. 192–198.

[2] Baxter, J., "Learning Internal Representations," *COLT-95,* Santa Cruz, CA, 1995.

[3] Caruana, R. and Freitag, D., "Greedy Attribute Selection," *ICML-94,* 1994, Rutgers, NJ, pp. 28-36.

[4] Caruana, R., "Learning Many Related Tasks at the Same Time with Backpropagation," *NIPS-94,* 1995, pp. 656-664.

[5] Caruana, R., "Applications and Algorithms for Multitask Learning," *ICML-96,* Bari, Italy, 1996, pp. 87-95.

[6] Caruana, R. and de Sa, V. R., "Promoting Poor Features to Supervisors: Some Inputs Work Better As Outputs," *NIPS-96,* 1997.

[7] Caruana, R., "Multitask Learning," Ph.D. thesis, Carnegie Mellon University, CMU-CS-97-203, 1997.

[8] Dietterich, T. G., Hild, H., and Bakiri, G., "A Comparison of ID3 and Backpropagation for English Text-to-speech Mapping," *Machine Learning,* 18(1), 1995, pp. 51-80.

[9] Ghosn, J. and Bengio, Y., "Multi-Task Learning for Stock Selection," *NIPS-96,* 1997.

[10] John, G., Kohavi, R. and Pfleger, K., "Irrelevant Features and the Subset Selection Problem," *ICML-94,* 1994, Rutgers, NJ, pp. 121-129.

[11] Koller, D. and Sahami, M., "Towards Optimal Feature Selection," *ICML-96,* Bari, Italy, 1996, pp. 284-292.

[12] Noordewier, M., Towell, G. and Shavlik, J., "Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences," *NIPS-90,* 1990.

[13] Suddarth, S. C. and Holden, A. D. C., "Symbolic-neural Systems and the Use of Hints for Developing Complex Systems," *International Journal of Man-Machine Studies,* 1991, 35(3), pp. 291-311.