Interpreting Black-box Models Through Sufficient Input Subsets

by

Brandon M. Carter

S.B., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
May 16, 2019

Certified by .....
David K. Gifford
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Interpreting Black-box Models Through Sufficient Input Subsets

by

Brandon M. Carter

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Recent progress in machine learning has come at the cost of interpretability, earning the field a reputation of producing opaque, "black-box" models. While deep neural networks are often able to achieve superior predictive accuracy over traditional models, the functions and representations they learn are usually highly nonlinear and difficult to interpret. This lack of interpretability hinders adoption of deep learning methods in fields such as medicine where understanding *why* a model made a decision is crucial. Existing techniques for explaining the decisions by black-box models are often restricted to either a specific type of predictor or are undesirably sensitive to factors unrelated to the model's decision-making process. In this thesis, we propose *sufficient input subsets*, minimal subsets of input features whose values form the basis for a model's decision. Our technique can rationalize decisions made by a black-box function on individual inputs and can also explain the basis for misclassifications. Moreover, general principles that globally govern a model's decision-making can be revealed by searching for clusters of such input patterns across many data points. Our approach is conceptually straightforward, entirely model-agnostic, simply implemented using instance-wise backward selection, and able to produce more concise rationales than existing techniques. We demonstrate the utility of our interpretation method on various neural network models trained on text, genomic, and image data.

Thesis Supervisor: David K. Gifford
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

This work would not have been possible without the support of countless people throughout my time at MIT.

First, I am enormously grateful to my advisor, David Gifford, for his guidance and mentorship throughout my research endeavours. Dave has always been receptive, available, and kind, while encouragingly offering many directions for improvement. I appreciate Dave's high standard of scientific rigor and am constantly inspired by his passion for science.

I am fortunate to have worked closely with Jonas Mueller, who has been a great collaborator and mentor, and to have been surrounded by a great group of labmates: Siddhartha Jain, Haoyang Zeng, Saber Liu, and the rest of the Gifford Lab. I also thank Tommi Jaakkola for a number of helpful discussions during my research.

In Fall 2017, I had the pleasure to teach MIT's machine learning course (6.036) alongside Profs. Leslie Kaelbling, Tomas Lozano-Perez, Tommi Jaakkola, and Jacob White. Teaching with them was both enjoyable and immensely rewarding.

I am thankful to have had fantastic academic advisors in the EECS department at MIT: Albert Meyer, Chris Terman, and Joel Emur. I also thank Anne Hunter and the Undergraduate EECS Office, as well as Linda Lynch, for their guidance and support.

Throughout my time at MIT, I was surrounded by a number of great friends and colleagues, with whom I share many fond memories. In particular, I want to thank Ken Leidal, who was a frequent course project partner and offered helpful suggestions throughout my work.

Finally, to my parents, my grandparents, and my family, thank you for enabling all of my successes and for your unconditional support – I could not have made it here without you. To Angie, thank you for helping me grow in too many ways to count, for your constant feedback and advice on my work and research, and for challenging and motivating me every day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, machine learning (ML) has seen widespread adoption in a large number of applications. ML methods are employed to understand natural language, speech, and images, to make medical diagnoses and propose treatment interventions, and as tools in scientific research. Machine learning algorithms are powerful because they are able to capture complex patterns that exist in observed data and exploit them to learn functions governing the data. These functions can then be used to make predictions about data that has not yet been observed. Deep learning methods (e.g. neural networks) are particularly powerful because they act hierarchically, where learned representations are expressed in terms of other, simpler, representations, which are also learned by the model. For example, deep learning systems in computer vision detect objects in images by combining increasingly complex representations, starting from pixels and edges and eventually assembling these into higher-level patterns. Deep learning has become extremely popular because of the state of the art performance it achieves in many domains, enabled by growing availability of data and by the decreasing cost of large amounts of computing power. However, compared to more traditional ML algorithms, deep learning models are generally poorly understood and have consequently earned reputations as opaque "black-boxes." While the complex, nonlinear functions learned by these models enable them to attain superior predictive accuracy, the performance gains come at the cost of limited interpretability.

The lack of interpretability of ML models is concerning because it means that

practitioners are unable to understand *how* their models make decisions. In many settings, it is crucial that the decisions made by an ML model can be interpretably rationalized. For example, consider ML models used by physicians to screen patients for a particular disease. Given patient data, a black-box model simply outputs *Disease* or *No Disease*, but without also providing an explanation (or rationale) for this decision, the model cannot be trusted. In such applications, it is imperative that model outputs can be justified. Interpretability is necessary to ensure that the prediction was not made as a result of spurious correlations or biases that may exist in the training data.

Interpretability has a number of other practical use cases in ML as well. Model interpretation is a useful tool for debugging, where an understanding of why a model makes certain *misclassifications* enables experts to revise the model and improve predictive accuracy. Such insights can also be used to aid research of new model architectures. Similarly, interpretability of model behavior can be used by practitioners to address the problem of model selection: which model to select from a large number of models trained on the same task. An understanding of the representations learned by the different models is useful to determine how the models differ in order to decide which is best suited for adoption in a particular system. Finally, interpretability can be leveraged in scientific settings, where highly accurate predictive models can be interpreted to extract new scientific principles underlying the observed data.

Current approaches for interpretable ML are limited in nature. One approach is to restrict the class of learned models to those which are inherently interpretable such as linear models or decision trees. Attention models and the generator-encoder approach of Lei et al. [26] are example variants of deep neural network architectures which aim to be inherently interpretable. However, in many applications, these architectures are either not suitable or are unable to attain supreme accuracy. Hence, there is often a trade-off between a model's predictive power and the ability to interpret its decisions. Additionally, these approaches cannot be used to explain decisions made by pre-trained models whose architectures may not even be known to the practitioner.

Other approaches for interpretability include a variety of attribution methods,

which produce an importance score for each individual input feature. However, many of these methods either require the model to be differentiable (e.g. gradient-based saliency methods) or necessitate an auxiliary explanation model (e.g. LIME [36]) in order to produce the attributions. Moreover, these attributions do not account for combinatorial interactions between features, for instance where a feature is only important when another feature takes on a certain value. Chapter 2 contains further discussion of existing approaches for model interpretability.

Another framework for interpreting model behavior involves identification of *rationales*, which are subsets of a particular input which can be considered the basis for the model's decision on that input. Suppose that a function f outputs $f(\mathbf{x})$ for some input $\mathbf{x}$ and that $\mathbf{x}$ contains a set of p features, $\mathbf{x} = [x_1, x_2, \ldots, x_p]$ (each $x_i \in \mathbb{R}^d$). For example, in a natural language application, $\mathbf{x}$ may be a sequence of text and x_i is the i^{th} word in the sequence (or more precisely, an embedding representation of word i), and f decides whether the text $\mathbf{x}$ expresses positive or negative sentiment (as in Figures 2-3 and 2-4). In this thesis, we propose a method which explains a decision $f(\mathbf{x})$ by identifying minimal subsets of the features in $\mathbf{x}$ that alone suffice for f to arrive at the same decision (even with the values of all the other features missing). These *sufficient input subsets* can then be understood as the basis for the model's decision on the given input.

In **Chapter 2** we introduce the sufficient input subsets (SIS) method for model interpretability. Our approach is entirely model-agnostic, requiring only black-box access to the function f. We make no assumptions about f so that our method is not restricted to interpreting specific classes of functions. Our approach is completely faithful to the underlying function f as no additional explanatory model is used, there is no additional training, and the sufficiency of the rationales is guaranteed (i.e. the model arrives at the same decision using only the values of features in the rationale as on the original input). SIS is based on a straightforward backward selection strategy that preserves interactions that may exist among input features. Finally, our method is simple and can be easily understood and applied in practice, even by non-experts. We explore and validate the SIS method by interpreting neural network models in

three different settings: a natural language sentiment analysis task, a biological task of predicting transcription factor binding, and a vision task classifying handwritten digits.

Chapter 3 extends the SIS method to not only explain a model's decisions on particular examples (locally), but also to gain a global understanding of the general principles governing the model's behavior. In particular, we cluster the sufficient input subsets stemming from many examples on which the model made the same decision. The resulting clusters provide global insights into the distinct feature patterns the model has learned to associate with the decision, such as the features learned by a CNN to discriminate a handwritten digit 4 from other digits (as in Section 3.2.3). We demonstrate the SIS clustering approach in each of our natural language, genomic, and image classification applications. Further, we adopt this SIS clustering methodology to reveal differences between two models trained on the same task. Jointly clustering the SIS stemming from both models allows us to inspect differences in how the models operate. We show that two models that achieve similar accuracy may make the same decisions for different reasons. This technique can be used to determine which of a set of models may be most desirable to use in a particular system. The methods developed in this thesis enable ML practitioners to discover such insights.

1.1 Thesis Outline

The remainder of this thesis is organized as follows. **Chapter 2** introduces the sufficient input subsets (SIS) method for interpreting black-box models. We apply the method in a number of domains and show that it outperforms alternative methods for explaining model behavior. In **Chapter 3**, we show how the local explanations of model-decision making can be aggregated over many examples to gain insight into global model behavior and to contrast the behavior of different models trained on the same task. **Chapter 4** concludes the thesis and discusses various opportunities for future work. Appendix A contains supplemental results and experimental details.

1.2 Bibliographic Note

Content in this thesis has appeared in other published work and was done in conjunction with a number of coauthors who were key in shaping this research. Chapters 2 and 3 are published in Carter et al. [7], whose citation is duplicated here:

Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. What made you do this? Understanding black-box decisions with sufficient input subsets. In *Artificial Intelligence and Statistics*, 2019.

Code for the experiments presented in this thesis is publicly available at:
`https://github.com/b-carter/SufficientInputSubsets`

Chapter 2

Sufficient Input Subsets for Interpretability

2.1 Introduction

The rise of neural networks and nonparametric methods in machine learning (ML) has driven enormous improvements in prediction capabilities, while simultaneously earning the field a reputation of producing complex black-box models. Vital applications, which could benefit most from improved prediction, are often deemed too sensitive for opaque learning systems. Such applications include the use of ML models to reject loan applicants [42], deny defendants' bail [24], or diagnose disease [18]. For ML models to be used in these systems, it is imperative that the decisions they make can be interpretably rationalized. Interpretability is also crucial in scientific applications, where it is hoped that underlying principles may be extracted from accurate predictive models [14, 28].

One simple explanation for *why* a particular black-box decision is reached may be obtained via a sparse subset of the input features whose values form the basis for the model's decision – a *rationale*. For text (or image) data, a rationale might consist of a subset of positions in the document (or image) together with the words (or pixel-values) occurring at these positions (examples shown in Figures 2-3 and 2-17). Here, we consider rationales that do not attempt to summarize the (potentially complex)

operations carried out within a black-box model, but instead point to the relevant features used by the model to arrive at a decision on that particular input. This property ensures that the interpretations remain faithful to any arbitrary model. Additionally, we desire that rationales are sparse, which facilitates interpretability when inputs are high-dimensional [26].

In this work, we develop a local explanation framework to produce rationales for a learned model that has been trained to map inputs $\mathbf{x} \in \mathcal{X}$ via some arbitrary learned function $f : \mathcal{X} \to \mathbb{R}$. Unlike many other interpretability techniques, our approach is not restricted to vector-valued data and does not require gradients or differentiability of f. Rather, each input example is solely presumed to have a set of indexable features $\mathbf{x} = [x_1, \ldots, x_p]$, where each $x_i \in \mathbb{R}^d$ for $i \in [p] = \{1, \ldots, p\}$. Our method can be applied to interpret decisions made on inputs $\mathbf{x}$ whose features are unordered (set-valued inputs) or for which the number of features p can vary (e.g. variable-length sequences). A rationale corresponds to a sparse subset of these indices $S \subseteq [p]$ together with the specific values of the features in this subset.

To understand why a certain decision was made for a given input $\mathbf{x}$, we propose a particular rationale called a *sufficient input subset* (SIS). Each SIS consists of a minimal input pattern present in $\mathbf{x}$ that alone suffices for f to produce the same decision, even if provided no other information about the rest of $\mathbf{x}$. Presuming the decision is based on $f(\mathbf{x})$ exceeding some prespecified threshold $\tau \in \mathbb{R}$, we seek to find a minimal-cardinality subset S of the input features such that $f(\mathbf{x}_S) \geqslant \tau$. Throughout, we use $\mathbf{x}_S \in \mathcal{X}$ to denote a modified input example in which all information about the values of features outside subset S has been masked, with features in S remaining at their original values. Thus, each SIS characterizes a particular standalone input pattern that drives the model toward the decision, providing sufficient justification for this choice from the model's perspective, even without any information about the values of the other features in $\mathbf{x}$.

In classification settings, f might represent the predicted probability of class C where we decide to assign the input to class C if $f(\mathbf{x}) \geqslant \tau$, where τ can be chosen based on precision/recall considerations. Each SIS in such an application corresponds

to a small input pattern that on its own is highly indicative of class C, according to the model. Note that by suitably defining f and τ with respect to the predictor outputs, any particular decision for input $\mathbf{x}$ can be precisely identified with the occurrence of $f(\mathbf{x}) \geqslant \tau$, such that greater values of f are associated with greater confidence in the decision.

For a given input $\mathbf{x}$ for which $f(\mathbf{x}) \geqslant \tau$, this work presents a simple method to find a complete collection of sufficient input subsets, each satisfying $f(\mathbf{x}_S) \geqslant \tau$, such that there exists no additional SIS outside of this collection. Each SIS may be understood as a disjoint piece of evidence that would lead the model to reach the same decision, and why this decision was reached for $\mathbf{x}$ can be unequivocally attributed to the SIS-collection. Furthermore, global insight on the general principles underlying the model's decision-making process may be gleaned by clustering the types of SIS extracted across different data points (Chapter 3). Such insights allow us to compare models based not only on their accuracy, but also on human-determined relevance of the concepts they target. Our method's simplicity facilitates its utilization by non-experts who may know very little about the models they wish to interrogate.

2.2 Related Work

Certain neural network variants such as attention mechanisms [39] and the generator-encoder of Lei et al. [26] have been proposed as powerful yet human-interpretable learners. Other interpretability efforts have tailored decompositions to certain convolutional/recurrent networks [31, 32, 33, 45], but these approaches are model-specific and only suited for ML experts. Many applications necessitate a model outside of these families, either to ensure supreme accuracy, or if training is done separately with access restricted to a black-box API [8, 47]. Thus, much recent research aims to address the critical need for methods which enable non-ML experts to rationalize the predictions of any type of model. One general approach entails fitting a separate explanation model to the outputs of f over the same training data, for example a feature-selector [27] or surrogate decision tree [17, 53, 49]. However, such a strategy

may not be generalizable to out of sample examples (which are crucial for understanding how f would behave in certain counterfactual settings).

An alternative model-agnostic approach to interpretability produces local explanations of f for a particular input $\mathbf{x}$. Local explanation often relies on attribution methods that quantify how much each feature influences the output of f at $\mathbf{x}$. Examples include LIME, which locally approximates f [36], saliency maps based on gradients of f [2, 41], Layer-wise Relevance Propagation [1], as well as the discrete DeepLIFT approach [40] and its continuous variant – Integrated Gradients (IG) [46], developed to ensure attributions reflect the cumulative difference in f at $\mathbf{x}$ vs. a reference input. A separate class of input-signal-based explanation techniques such as DeConvNet [51], Guided Backprop [44], and PatternNet [22] employ gradients of f in order to identify input patterns that cause f to output large values. However, many gradient-based saliency methods have been deemed unreliable, depending not only on the learned function f, but also on its specific architectural implementation and how inputs are scaled [21, 22]. More like our approach, recent techniques from Dabkowski and Gal [12], Kim et al. [20], Chen et al. [9] also aim to identify input patterns that best explain certain decisions, but additionally require either a predefined set of such patterns or an auxiliary neural network trained to identify them.

In comparison with the aforementioned methods, our SIS approach is: conceptually simple, entirely faithful to any type of model, and requires neither gradients of f nor auxiliary training of the underlying model f or a surrogate explanation model. Also related to our subset-selection methodology are the ideas of Li et al. [27] and Fong and Vedaldi [16], which for a particular input seek a small feature subset whose omission causes a substantial drop in f such that a different decision would be reached. However, this objective can produce adversarial artifacts that are hard to interpret. In contrast, we focus on identifying small subsets of input features whose values suffice to ensure f outputs significantly positive predictions, even in the absence of any other information about the rest of the input. While the techniques of Li et al. [27] and Fong and Vedaldi [16] produce rationales that remain highly dependent on the rest of the input outside of the selected feature subset, each rationale identified by our

SIS approach is independently considered by f as an entirely sufficient justification for a particular decision in the absence of other information.

2.3 Sufficient Input Subsets Method

Our approach to rationalizing why a particular black-box decision is reached only applies to input examples $\mathbf{x} \in \mathcal{X}$ that meet the decision criterion $f(\mathbf{x}) \geqslant \tau$. For such an input $\mathbf{x}$, we aim to identify a SIS-collection of disjoint feature subsets $S_1, \ldots, S_K \subseteq [p]$ that satisfy the following criteria:

(1) $f(\mathbf{x}_{S_k}) \geqslant \tau$ for each $k = 1, \ldots, K$

(2) There exists no feature subset $S' \subset S_k$ for some $k = 1, \ldots, K$ such that $f(\mathbf{x}_{S'}) \geqslant \tau$

(3) $f(\mathbf{x}_R) < \tau$ for $R = [p] \setminus \bigcup_{k=1}^{K} S_k$ (the remaining features outside of the SIS-collection)

Criterion (1) ensures that for any SIS S_k, the values of the features in this subset alone suffice to justify the decision in the absence of any information regarding the values of the other features. To ensure information that is not vital to reach the decision is not included within the SIS, criterion (2) encourages each SIS to contain a minimal number of features, which facilitates interpretability. Finally, we require that our SIS-collection satisfies a notion of completeness via criterion (3), which states that the same decision is no longer reached for the input after the entire SIS-collection has been masked. This implies the remaining feature values of the input no longer contain sufficient evidence for the same decision. Figures 2-4 and 2-17 show SIS-collections found in text and image inputs, respectively.

Recall that $\mathbf{x}_S \in \mathcal{X}$ denotes a modified input in which the information about the values of features outside subset S is considered to be missing. We construct $\mathbf{x}_S$ as new input whose values on features in S are identical to those in the original $\mathbf{x}$, and whose remaining features $x_i \in [p] \setminus S$ are each replaced by a special mask $z_i \in \mathbb{R}^{d_i}$ used

to represent a missing observation. While certain models are specially adapted to handle inputs with missing observations [43], this is generally not the case. To ensure our approach is applicable to all models, we draw inspiration from data imputation techniques which are a common way to represent missing data [38].

Two popular strategies include hot-deck imputation, in which unobserved values are sampled from their marginal feature distribution, and mean imputation, in which each z_i is simply fixed to the average value of feature i in the data. Note that for a linear model, these two strategies are expected to produce an identical change in prediction $f(\mathbf{x}) - f(\mathbf{x}_S)$. We find in practice that the change in predictions resulting from either masking strategy is roughly equivalent even for nonlinear models such as neural networks (Section A.1.1, Figure A-1). In this work, we favor the mean-imputation approach over sampling-based imputation, which would be computationally-expensive and nondeterministic (undesirable for facilitating interpretability). One may also view $\mathbf{z}$ as the *baseline* input value used by feature attribution methods [46, 40], a value which should not lead to particularly noteworthy decisions. Since our interests primarily lie in rationalizing atypical decisions, the average input arising from mean imputation serves as a suitable baseline. Zeros have also been used to mask image and categorical data [27], but empirically, this mask appears undesirably more informative than the mean (predictions more affected by zero-masking).

For an arbitrarily complex function f over inputs with many features p, the combinatorial search to identify sets which satisfy objectives (1)-(3) is computationally infeasible. To find a SIS-collection in practice, we employ a straightforward backward selection strategy, which is here applied separately on an example-by-example basis (unlike standard statistical tools which perform backward selection globally to find a fixed set of features for all inputs). The **SIScollection** algorithm details our straightforward procedure to identify disjoint SIS subsets that satisfy (1)-(3) approximately for an input $\mathbf{x} \in \mathcal{X}$ where $f(\mathbf{x}) \geqslant \tau$. Disjointness of the sufficient input subsets in a SIS-collection is crucial to ensure computational tractability and that the number of SIS per example does not grow huge and hard to interpret. For a more rigorous evaluation of the properties of SIS, see Section 3.1 of Carter et al. [7].

SIScollection(f, $\mathbf{x}$, τ)

$S = [p]$
for $k = 1, 2, \ldots$ **do**
    $R =$ **BackSelect**$(f, \mathbf{x}, S)$
    $S_k =$ **FindSIS**$(f, \mathbf{x}, \tau, R)$
    $S \leftarrow S \backslash S_k$
    if $f(\mathbf{x}_S) < \tau$: **return** S_1,...,S_{k-1}

BackSelect(f, $\mathbf{x}$, S)

$R =$ empty stack
while $S \neq \varnothing$ **do**
    $i^* = \text{argmax}_{i \in S} f(\mathbf{x}_{S \backslash \{i\}})$
    Update $S \leftarrow S \backslash \{i^*\}$
    Push i^* onto top of R
return R

FindSIS(f, $\mathbf{x}$, τ, R)

$S = \varnothing$
while $f(\mathbf{x}_S) < \tau$ **do**
    Pop i from top of R
    Update $S \leftarrow S \cup \{i\}$
if $f(\mathbf{x}_S) \geqslant \tau$: **return** S
else: **return** *None*

Our overall strategy is to find a SIS subset S_k (via **BackSelect** and **FindSIS**), mask it out, and then repeat these two steps restricting each search for the next SIS solely to features disjoint from the currently found SIS-collection $S_1, \ldots, S_k$, until the decision of interest is no longer supported by the remaining feature values. In the **BackSelect** procedure, $S \subset [p]$ denotes the set of remaining unmasked features that are to be considered during backward selection. For the current subset S, step 3 in

BackSelect identifies which remaining feature $i \in S$ produces the *minimal* reduction in $f(\mathbf{x}_S) - f(\mathbf{x}_{S\setminus\{i\}})$ (meaning it least reduces the output of f if additionally masked), a question trivially answered by running each of the remaining possibilities through the model. This strategy aims to gradually mask out the least important features in order to reveal the core input pattern that is perceived by the model as sufficient evidence for its decision. Finally, we build our SIS up from the last ℓ features omitted during the backward selection, selecting a ℓ value just large enough to meet our sufficiency criterion (1). Because we desire minimality of the SIS as specified by (2), it is not appropriate to terminate the backward elimination in **BackSelect** as soon as the sufficiency condition $f(\mathbf{x}_S) \geqslant \tau$ is violated, due to the possible presence of local minima in f along the path of subsets encountered during backward selection (as shown in Figure 2-19).

Because this approach always queries a prediction over the joint set of remaining features S, it is better suited to account for interactions between these features and ensure their sufficiency (i.e. that $f(\mathbf{x}_S) \geqslant \tau$) compared to a forward selection in the opposite direction which builds the SIS upwards one feature at a time by greedily maximizing marginal gains. Throughout its execution, **BackSelect** attempts to maintain the sufficiency of $\mathbf{x}_S$ as the set S shrinks. Given p input features, our algorithm requires $\mathcal{O}(p^2k)$ evaluations of f to identify k SIS, but we can achieve $\mathcal{O}(pk)$ by parallelizing each argmax in **BackSelect** (for example, by batching on GPU).

2.4 Experimental Overview

In the following sections, we apply our SIS method to analyze neural networks in three settings: (1) a natural language task involving multi-aspect sentiment analysis on beer reviews, (2) predicting transcription factor binding in biological data, and (3) image classification on handwritten digits. **SIScollection** is compared with alternative methods for producing rationales (details in Section 2.4.1). Note that our **BackSelect** procedure determines an ordering of elements, R, subsequently used to construct the SIS. Depictions of each SIS are shaded based on the feature order in R (darker =

later), which can indicate relative feature importance within the SIS.

In the "Suff. IG," "Suff. LIME," and "Suff. Perturb." (*sufficiency constrained*) methods, we instead compute the ordering of elements R according to the feature attribution values output by integrated gradients [46], LIME [36], or a perturbative approach that measures the change in prediction when individually masking each feature (see Section 2.4.1). The rationale subset S produced under each method is subsequently assembled using **FindSIS** exactly as in our approach and thus is guaranteed to satisfy $f(\mathbf{x}_S) \geqslant \tau$. In the "IG," "LIME," and "Perturb." (*length constrained*) methods, we use the same previously described ordering R, but always select the same number of features in the rationale as in the SIS produced by our method (per example). We also compare against the additional "Top IG" method, in which top features from R are added into the rationale until sum of integrated gradients attributions suggests that the rationale has met our sufficiency criterion (see Section 2.4.1).

2.4.1 Details of Baseline Methods

Throughout this chapter, we employ a number of alternative methods for identifying rationales for comparison with SIS. Here, we provide detailed descriptions and implementation details of these baseline methods.

We use methods based on integrated gradients [46], LIME [36], and feature perturbation. Note that integrated gradients is an attribution method which assigns a numerical score to each input feature. LIME likewise assigns a weight to each feature using a local linear regression model for f around $\mathbf{x}$. In the perturbative approach, we compute the change in prediction when each feature is individually masked, as in Equation 2.1 (of Section 2.5.3). Each of these feature orderings R is used to construct a rationale using the **FindSIS** procedure (Section 2.3) for the "Suff. IG," "Suff. LIME," and "Suff. Perturb." (*sufficiency constrained*) methods.

Note that our text classification architecture (described in Section 2.5.1) encodes discrete words as 100-dimensional continuous word embeddings. The integrated gradients method returns attribution scores for each coordinate of each word embedding. For each word embedding $x_i \in \mathbf{x}$ (where each $x_i \in \mathbb{R}^{100}$), we summarize the attribu-

tions along the corresponding embedding into a single score y_i using the L_1 norm: $y_i = \sum_d |x_{id}|$ and compute the ordering R by sorting the y_i values.

We use an implementation of integrated gradients for Keras-based models from `https://github.com/hiranumn/IntegratedGradients`. In the case of the beer review dataset (Section 2.5), we use the mean embedding vector as a baseline for computing integrated gradients. In the case of TF binding (Section 2.6), we use the $[0.25, 0.25, 0.25, 0.25]$ uniform mean vector as the baseline reference value. As suggested in Sundararajan et al. [46], we verified that the prediction at the baseline and the integrated gradients sum to approximately the prediction of the input.

For LIME and our beer reviews dataset, we use the approach described in Ribeiro et al. [36] for textual data, where individual words are removed entirely from the input sequence. In our TF binding dataset, LIME replaces bases with the unknown `N` base (represented as the uniform-distribution $[0.25, 0.25, 0.25, 0.25]$). We use the implementation of LIME at: `https://github.com/marcotcr/lime`. The `LimeTextExplainer` module is used with default parameters, except we set the maximal number of features used in the regression to be the full input length so we can order all input features.

Additionally, we explore methods in which we use the same ordering R by these alternative methods but select the same number of input features in the rationale to be the median SIS length in the SIS-collection computed by our method on each example: the "IG," "LIME," and "Perturb." (*length constrained*) methods. In the TF binding models, we use a baseline of zero vectors such that the integrated gradients result along the encoded sequence is also one-hot. We compute the feature ordering based on the absolute value of the non-zero integrated gradient attributions.

In TF binding data (Section 2.6), we add an additional method, "Top IG," in which we compute integrated gradients using an all-zeros baseline and order features by attribution magnitude (as in the length constrained IG method). But, we select elements for the rationale by finding the minimum number of elements necessary such that the sum of integrated gradients of those features equals $\tau - f(\mathbf{0})$, where $\mathbf{0}$ is the all-zeros baseline for integrated gradients. Note that for the length constrained and

Top IG methods, there is no guarantee of sufficiency $f(\mathbf{x}_S) \geqslant \tau$ for any input subset S.

2.5 Sentiment Analysis of Reviews

We first consider a dataset of beer reviews from McAuley et al. [30] where beers receive text reviews along with numerical ratings of aspects like aroma, appearance, and palate. Taking the text of the review as input, three different LSTM recurrent neural networks [19] are trained to predict the continuous-valued sentiment toward each aspect. We apply our SIS method to interpret and validate the decisions made by these LSTMs.

In this section, we present results interpreting the LSTM trained to predict sentiment toward the *aroma* aspect in particular. Results for the *appearance* and *palate* aspects are similar and can be found in the Supplementary Material of Carter et al. [7].

2.5.1 Dataset and Model Details

Following Lei et al. [26], we use a preprocessed version of the BeerAdvocate[1] dataset[2] which contains decorrelated numerical ratings toward three aspects: *aroma*, *appearance*, and *palate* (each normalized to $[0, 1]$). Dataset statistics can be found in Table 2.1. Reviews are tokenized by converting to lowercase and filtering punctuation, and we use a vocabulary containing the top 10,000 most common words. McAuley et al. [30] also provide a subset of human-annotated reviews, in which humans manually selected full sentences in each review that describe the relevant aspects. This annotated set is never seen during training and used solely as part of our evaluation.

Long short-term memory (LSTM) networks [19] are commonly employed for natural language tasks such as sentiment analysis [48, 35]. In these experiments, we use a recurrent neural network (RNN) architecture with two stacked LSTMs as follows:

[1] `https://www.beeradvocate.com/`

[2] `http://snap.stanford.edu/data/web-BeerAdvocate.html`

Table 2.1: Summary and performance statistics (mean squared error (MSE) and Pearson correlation coefficient ρ) for LSTM models on beer reviews data.

Aspect	Fold	Size	MSE	Pearson ρ
Appearance	Train	80,000	0.016	0.864
	Validation	3,000	0.024	0.783
	Test	7,000	0.023	0.801
	Annotation	994	0.020	0.563
Aroma	Train	70,000	0.014	0.873
	Validation	3,000	0.024	0.767
	Test	7,000	0.025	0.756
	Annotation	994	0.021	0.598
Palate	Train	70,000	0.016	0.835
	Validation	3,000	0.029	0.680
	Test	7,000	0.028	0.694
	Annotation	994	0.016	0.592

1. **Input/Embeddings Layer**: Sequence with 500 timesteps, the word at each timestep is represented by a (learned) 100-dimensional embedding

2. **LSTM Layer 1**: 200-unit recurrent layer with LSTM (forward direction only)

3. **LSTM Layer 2**: 200-unit recurrent layer with LSTM (forward direction only)

4. **Dense**: 1 neuron (sentiment output), sigmoid activation

Taking the text of a review as input, different LSTM networks are trained to predict user-provided numerical ratings of each aspect. We train the models using the Adam optimizer [23] to minimize mean squared error (MSE) on the training set. We use a held-out set of 3,000 examples for validation (sampled at random from the pre-defined test set from Lei et al. [26]). Our test set consists of the remaining 7,000 test examples. Training results are shown in Table 2.1.

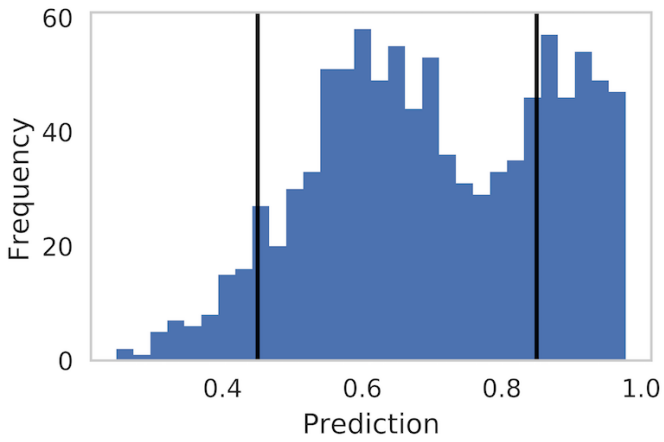


Figure 2-1: Predictive distribution on the annotation set (held-out) using the LSTM model for aroma. Vertical lines indicate decision thresholds ($\tau_+ = 0.85$, $\tau_- = 0.45$) selected for **SIScollection**.

2.5.2 Applying SIS to Interpret Sentiment Predictors

We apply SIS to interpret the LSTM's decisions on the set of reviews containing sentence-level annotations (Annotation fold in Table 2.1). Note that these reviews (and the human annotations) were not seen during training. Tokens in the input sequence are masked by replacement with a mean embedding taken over the learned vocabulary (see Appendix A.1.1 for further discussion of our mean imputation approach). In our formulation (Section 2.3), we apply the SIS method to inputs $\mathbf{x}$ for which $f(\mathbf{x}) \geqslant \tau$. For the sentiment analysis task, we analogously apply our method for both $f(\mathbf{x}) \geqslant \tau_+$ and $-f(\mathbf{x}) \geqslant -\tau_-$, where the model predicts either strong positive or strong negative sentiment, respectively. We choose thresholds $\tau_+ = 0.85$, $\tau_- = 0.45$ and extract the complete set of sufficient input subsets using our method. These thresholds were set empirically such that they were sufficiently apart, based on the predictive distribution on the held-out annotated set (shown in Figure 2-1). For most reviews, **SIScollection** outputs a collection containing just one or two sufficient input subsets (Figure 2-2).

Figure 2-3 shows a sample beer review in which we highlight the SIS identified for the LSTMs that predict each aspect. In this example, the SIS-collection for each of the three LSTMs only contained a single sufficient input subset. We see that each

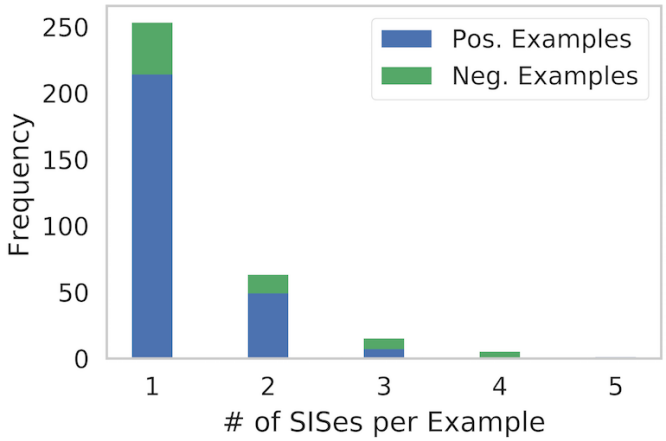


Figure 2-2: Number of sufficient input subsets for aroma identified by **SIScollection** per example.

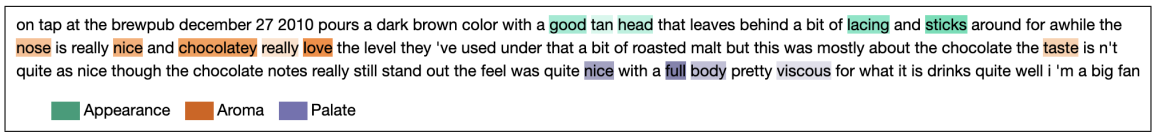


Figure 2-3: Beer review with one sufficient input subset identified for the prediction of each aspect.

SIS only captures sentiment toward the relevant aspect (as compared to just general positive sentiment), revealing that the LSTMs have learned to make predictions based on context-specific features.

Figure 2-4 depicts the SIS-collection identified from a review the LSTM decided to flag for positive aroma. Here, the SIS-collection for this review is comprised of three sufficient input subsets. From this example, we can see that the aroma LSTM is generally making predictions based on disjoint pieces of evidence in the text suggesting positive sentiment toward the aroma aspect. However, the example also shows how this type of analysis may be used to debug or improve the model. While the rationales generally seem sound, a practitioner may desire that the models not include tokens such as "t" or "s" (which are likely artifacts of tokenization) in the rationales for its decisions.

To gain further insight into the behavior of the LSTM models, we next analyze the predictor model's output following the elimination of each feature in the **BackSelect** procedure (Section 2.3). Figure 2-5 shows the LSTM output on the remaining

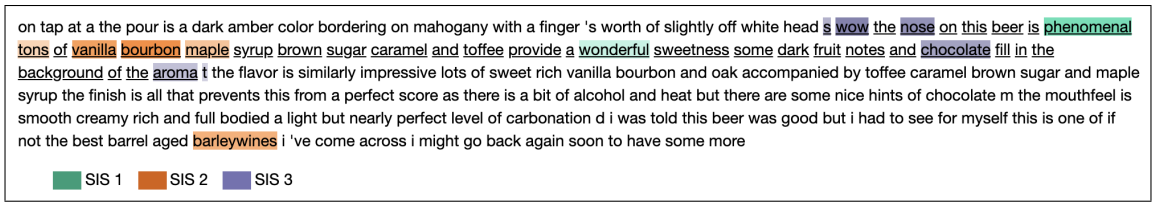


Figure 2-4: Beer review with three disjoint SIS S_1, S_2, S_3 identified for a positive aroma prediction. Underlined are sentences that human labelers manually annotated as capturing the aroma sentiment.

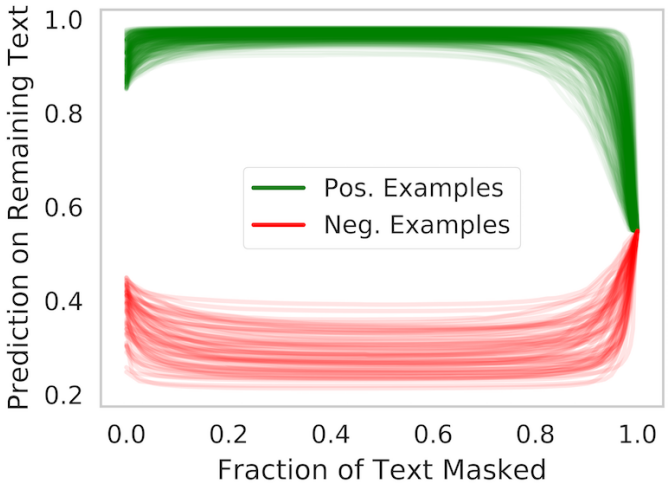


Figure 2-5: Prediction history on remaining (unmasked) text at each step of the **BackSelect** procedure, for examples where aroma sentiment is predicted.

unmasked text $f(\mathbf{x}_{S\setminus\{i^*\}})$ at each iteration of **BackSelect**, for all examples. This figure reveals that only a small number of features are needed by the model in order to make a strong prediction (most features can be removed without changing the prediction). We see that as those final, critical features are removed, there is a rapid, monotonic decrease in output values. Finally, we see that the first features to be removed by **BackSelect** are those which generally provide negative evidence against the decision. The prediction becomes more positive (or negative in the case of strong negative sentiment reviews [red]) as the first features are eliminated. Note, however, that **BackSelect** may exhibit different behavior when applied to other models or architectures (see Figure 2-19 for one such example).

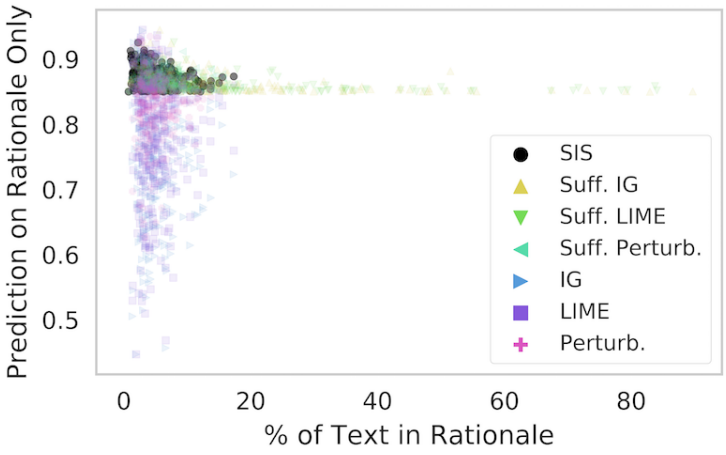


Figure 2-6: Prediction on rationales only vs. rationale length for various methods in reviews with positive aroma prediction ($\tau = 0.85$).

2.5.3 Comparing SIS to Baseline Methods

We next compare rationales produced by **SIScollection** to those from the baseline methods described in Section 2.4.1. Figure 2-6 shows the prediction on the rationale only (all other words masked) vs. length of rationale for the rationales produced by these various methods on the same set of beer reviews on which the LSTM predicts positive aroma. From this figure, we see that when the alternative methods are length constrained, the rationales they produce often badly fail to meet our sufficiency criterion. Thus, even though the same number of feature values are preserved in the rationale and these alternative methods select the features to which they have assigned the largest attribution values, their rationales lead to significantly reduced f outputs compared to our SIS subsets. When the lengths of these alternative rationales is allowed to grow large enough to ensure our sufficiency criterion is met, the rationales become unnecessarily long. If the sufficiency constraint is instead enforced for these alternative methods, the rationales they identify become significantly larger than those produced by **SIScollection**, and they also tend to contain a larger number of unimportant features (as shown in Table 2.2 and Figures 2-7 and 2-8, detailed below). Thus, our SIS method effectively extracts rationales that are sparse yet suffice for a strong prediction by the model.

We also compare the rationales from our SIS method those from the baseline meth-

Table 2.2: Statistics for rationale length and feature importance in aroma prediction. For rationale length, median and max indicate percentage of input text in the rationale. For marginal perturbed feature importance, we indicate the median importance of features in rationales and features from the other (non-rationale) text. p-values are computed using a Wilcoxon rank-sum test (comparing each distribution to that from SIS).

Method	Rationale Length (% of text)			Marg. Perturbed Feat. Import.		
	Med.	Max	p	Rationale	Other	p
SIS	**3.9%**	**17.3%**	–	0.0112	1.50e-05	–
Suff. IG	7.7%	89.7%	5e-26	0.0068	1.85e-05	3e-42
Suff. LIME	7.2%	84.0%	4e-23	0.0075	1.87e-05	1e-35
Suff. Perturb.	5.1%	18.3%	1e-06	0.0209	1.90e-05	1e-72

ods by comparing the importance of features that comprise the rationales. For each word i in an input sequence $\mathbf{x}$, we quantify its marginal importance by individually perturbing only this word:

$$\text{Feature Importance}(i) = f(\mathbf{x}) - f(\mathbf{x} \backslash \{i\}) \tag{2.1}$$

Note that these marginal Feature Importance scores are identical to those of the Perturb. method described in Section 2.4.1.

For rationales computed by the various methods on these beer reviews, we compute the marginal Feature Importance of features in the rationales, which are summarized in Table 2.2 and Figure 2-7. Compared to the Suff. IG and Suff. LIME methods, our **SIScollection** technique produces rationales that are much shorter and contain fewer irrelevant (i.e. not marginally important) features (Table 2.2, Figures 2-7 and 2-8). Note that by construction, the rationales of the Suff. Perturb. method contain features with the greatest Feature Importance, since this precisely how the ranking in Suff. Perturb. is defined.

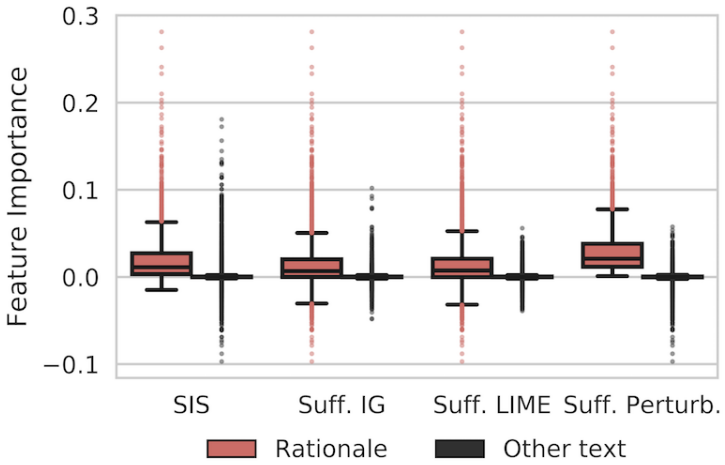


Figure 2-7: Importance of individual features in the rationales for aroma prediction in beer reviews.

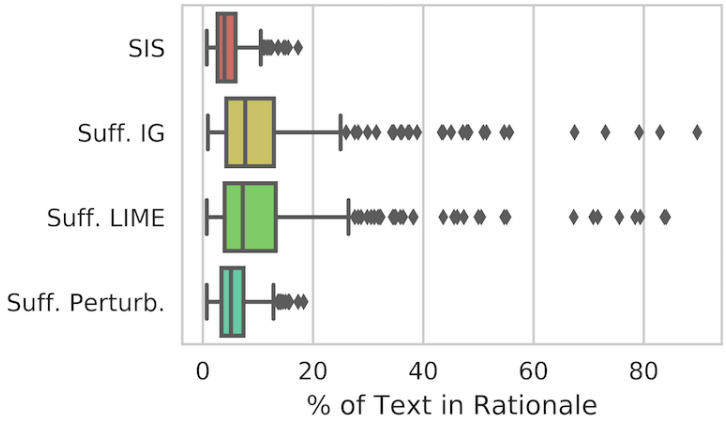


Figure 2-8: Length of rationales for aroma prediction.

2.5.4 Evaluation of SIS Rationales

Benchmarking interpretability methods is difficult because a learned f may behave counterintuitively such that seemingly unreasonable model explanations are in fact faithful descriptions of a model's decision-making process. For some reviews in our dataset, a human annotator has manually selected which sentences carry the relevant sentiment for the aspect of interest (Section 2.5.1, see examples in the underlined text of Figures 2-4 and 2-10), so we treat these annotations as an alternative rationale for the LSTM prediction. For a review $\mathbf{x}$ whose true and predicted aroma exceed our decision threshold, we define the *quality of human-selected sentences for model explanation* (QHS) as

$$\text{QHS} = f(\mathbf{x}_S) - f(\mathbf{x}) \tag{2.2}$$

where S here is the human-selected-subset of words in the review (as opposed to a sufficient input subset).

Figure 2-9 shows the relationship between QHS and the fraction of the SIS that falls inside the human-selected sentences. There is a positive correlation between the two variables (Pearson $\rho = 0.491$, $p = 1.5\text{e}{-25}$). High variability of QHS in the annotated reviews indicates the human rationales often do not contain sufficient information to preserve the LSTM's decision. As the model diverges from alignment with the human-selected sentences (and those sentences are not necessary for prediction), fewer words in the sufficient input subsets lie within those sentences (lower left of Figure 2-9). Additionally, as the human-selected sentences become more sufficient for prediction (QHS $\rightarrow 0$), almost the entirety of the sufficient input subsets identified by our method end up lying within those sentences (upper right of Figure 2-9). Figure 2-10 provides examples from both extremes of alignment (SIS has good alignment with human-selected sentences, where QHS ≈ 0, and SIS and human-selected sentences have poor alignment, where QHS < 0). The bottom panel of Figure 2-10 illustrates an example where the LSTM is able to predict positive sentiment from features that diverge from what a human would expect, which may suggest overfitting.

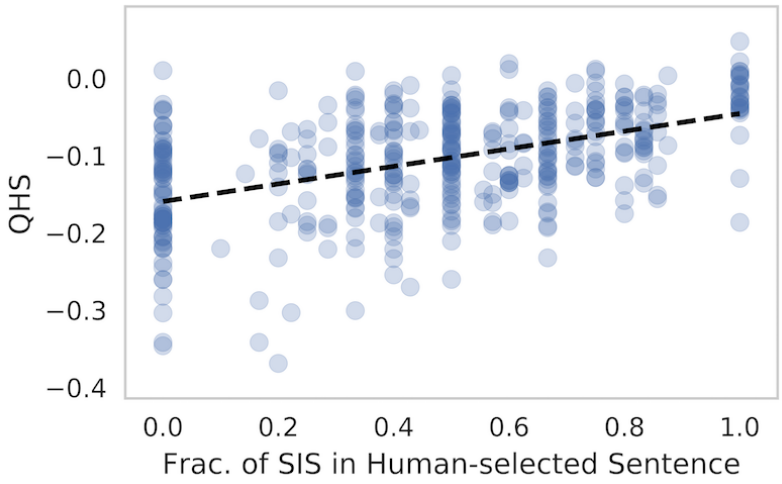


Figure 2-9: QHS (Equation 2.2) vs. similarity between SIS and annotation in the reviews with positive aroma sentiment (Pearson $\rho = 0.491$, p-value = 1.5e−25).

Good Alignment with Human-selection: QHS = 0.00296, Full Sequence Prediction = 0.900

poured from a 24oz bottle into a large appearance this pours a deep bronze amber in color this beer has some of the best head formation and retention that i have ever seen along with lots and lots of sticky lacing smell tons of piney resinous evergreen abound in this ale cant get enough of this beer its the best smelling harvest ale i 've ever had the pleasure of enjoying taste huge flavor profile with lots of bitterness and only a little malt sweetness as this beer warms up the bitterness really dominates the flavor i 'm tasting lot of subtle orange aromas mouthfeel full body beer with alot of carbonation overall i love this beer in my opinion its nevada 's best beer overall the price is i bought my for 3 99 so much flavor and biting bitterness this harvest ale has no equal

Poor Alignment with Human-selection: QHS = -0.3037, Full Sequence Prediction = 0.928

like having a bowl of honey nut of malt for breakfast this beer makes me as happy as a pearly going all from tree to tree the fat is actually pretty interesting especially when compared to most brown ales its aromas and flavor are a perfectly balanced blend of honey hazelnut toast walnut and a tad smidgen of vanilla the mouthfeel is light and smooth not hoppy in the slightest quite sweet too damn drinkable for our own good

Figure 2-10: Beer reviews (aroma) in which human-selected sentences (underlined) are aligned well (top) and poorly (bottom) with predictive model. Fraction of SIS in the human sentences corresponds accordingly. In the bottom example (poor alignment between human-selection and predictive model), our procedure has surfaced a case where the LSTM has learned features that diverge from what a human would expect (and may suggest overfitting).

2.6 Transcription Factor Binding

We next analyze SIS in the context of convolutional neural networks (CNNs) trained to classify whether a given transcription factor (TF) will bind to a specific DNA sequence [52]. This setting also provides us with ground truth motifs containing known binding sites, which we use to evaluate the ability of SIS to recover such motifs (see Section 2.6.3).

2.6.1 Dataset and Model Details

We use the *motif occupancy* datasets[3] from Zeng et al. [52], where each dataset originates from a ChIP-seq experiment from the ENCODE project [11]. Each of the 422 datasets studies a particular transcription factor, containing between 600 and 700,000 (median 50,000) 101 base-pair DNA sequences (inputs) each associated with a binary label based on whether the sequence is bound by the TF or not. Each dataset also contains a test set ranging between 150 and 170,000 sequences (median 12,000). Here, the positive and negative classes in each dataset are balanced, and we filter out all sequences containing the unknown base (`N`). The nucleotide occurring at base position (`A`, `C`, `G`, `T`) is encoded as a one-hot representation which is fed into the CNN. Zeng et al. [52] showed that convolutional neural network architectures outperform other models for this TF binding prediction task.

For each of the 422 prediction tasks, we employ the optimal "1layer_128motif" architecture from Zeng et al. [52], defined as follows:

1. **Input**: (101 x 4) sequence encoding
2. **Convolutional Layer 1**: Applies 128 kernels of window size 24, with ReLU activation
3. **Global Max Pooling Layer 1**: Performs global max pooling
4. **Dense Layer 1**: 32 neurons, with ReLU activation and dropout probability 0.5

[3] available at `http://cnn.csail.mit.edu`

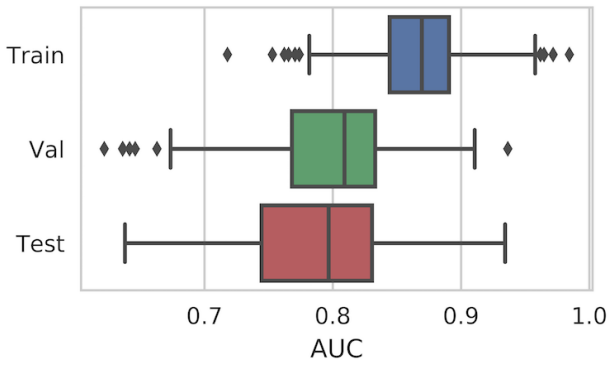


Figure 2-11: Median area under the receiver operating curve (AUC) for all 422 transcription factor binding motif occupancy datasets. The validation set is held-out at training but used to choose model parameters; the test set is not seen until after training.

5. **Dense Layer 2**: 1 neuron (output probability), with sigmoid activation

We hold out 1/8 of each train set for validation and minimize binary cross-entropy using the Adadelta optimizer [50] with default parameter settings in Keras [10]. We train each model on each of the 422 datasets for 10 epochs (using batch size 128) with early-stopping based on validation loss. Figure 2-11 shows the area under the receiver operating curve (AUC) over the 422 datasets, and we note that the performance of our models closely resembles that in Zeng et al. [52].

2.6.2 Applying SIS to Interpret TF Binding Classifiers

From each of the 422 different datasets of DNA sequences bound-or-not by different TFs (and 422 different CNN models, see Section 2.6.1), we extract SIS-collections from sequences in the test set with high (top 10%) predicted binding affinity for the TF profiled in each dataset. The distribution of threshold τ over the 422 datasets is shown in Figure 2-12. Since `A`, `C`, `G`, `T` nucleotides all occur with similar frequency in this data, our SIS analysis simply masks each base using a uniform embedding ($[0.25, 0.25, 0.25, 0.25]$). This is also the standard strategy to represent unknown `N` nucleotides in DNA sequences that typically arise from issues in read quality. We generally find that there is only a single SIS per example for the sequences in these

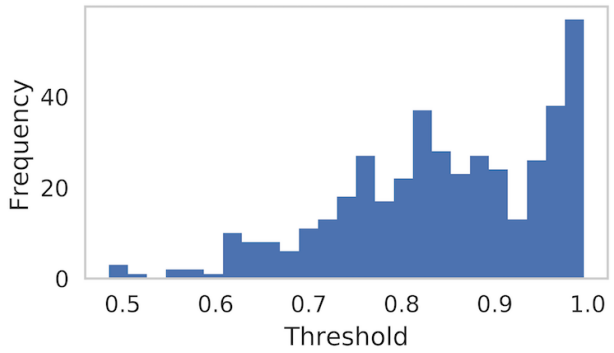


Figure 2-12: Thresholds τ used for identifying sufficient input subsets in TF binding datasets. In each dataset, the threshold is defined as the 90th percentile of the predictive test distribution.

CACTGTCATTCTCTTGGTCAGCCCTGGACATCCCTGGAAAGGATGACTCAGCTGTCCGTTTTAAACAGGGTAGTTCAGAAGAATACATTCCTGGTTATTCA
TTTTTTTCTCCCTTCGATTTCCACTATGATTTGTATTTCCTTTGTTCTGCTGACTTTGCAATTTCGGTTGTTTTTTCTAAATTTCTTAGGGTGAAAACTGA

Figure 2-13: Two DNA sequences that receive positive TF binding predictions for the MAFF factor (SIS is shaded).

datasets. Figure 2-13 depicts two examples of input DNA sequences and the corresponding sufficient input subsets identified by our **SIScollection** procedure.

We evaluate the minimality and sufficiency of the rationales produced by SIS to those produced by the alternative baseline methods we explored (see Section 2.4.1). On each dataset, we compute the median rationale length (as number of bases in the rationale). The distribution of median rationale length over all datasets by the various methods is shown in Figure 2-14. Note that for the IG, LIME, and Perturb. methods, rationale length was constrained to the length of the rationales produced by our method, as described in Section 2.4.1. For the Top IG method, neither sufficiency nor length constraints are enforced. We see that when the sufficiency constraint is enforced in alternative methods (Suff. IG), the rationales are significantly longer than those identified by SIS. Moreover, as shown in Figure 2-15, when the sufficiency constraint is not enforced (or the rationale lengths are constrained to the length of SIS rationales) in alternative methods, the rationales have significantly less predictive power, often not satisfying $f(\mathbf{x}_S) \geqslant \tau$. The rationales produced via our SIS approach are shorter and better at preserving large f-values than rationales from other methods

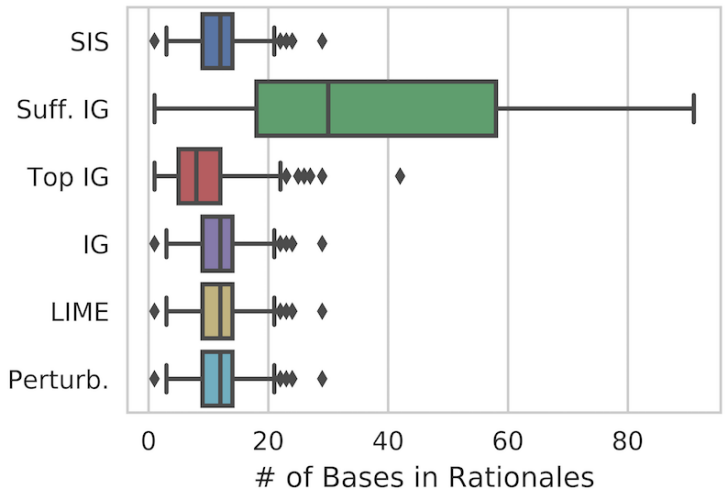


Figure 2-14: Length (number of bases) of rationales identified by various methods. Note that the sufficiency constraint ($f(\mathbf{x}_S) \geqslant \tau$) is only enforced for SIS and Suff. IG. The lengths of IG, LIME, and Perturb. rationales are constrained to the length of SIS rationales.

(Figures 2-14 and 2-15).

2.6.3 Evaluation of the Quality of TF Rationales

To predict binding so accurately (Figure 2-11), the CNN must faithfully reflect the biological mechanisms that relate the DNA sequence to the probability of TF occupancy. We evaluate the rationales found by SIS and our baseline methods (see Section 2.4.1) against known TF binding motifs from JASPAR [29] as the ground truth. We adopt KL divergence between the known motif and each proposed rationale as a measure of quality of the rationale. Each rationale is padded with "`N`" (unknown) bases to the length of a full input sequence (101 bases) and optimally aligned with the known motif[4] according to the likelihood criterion. The aligned motif is then also padded to the same length, and we compute the divergence between

[4]A JASPAR motif is a $n \times 4$ right stochastic matrix M. The columns represent the ACGT DNA bases and the rows a DNA sequence. It represents the marginal probability of the base j at position i being present with probability M_{ij}. The unknown base "N" receives uniform 1/4 probability for each of ACGT. An example JASPAR motif is shown in Figure 3-1.

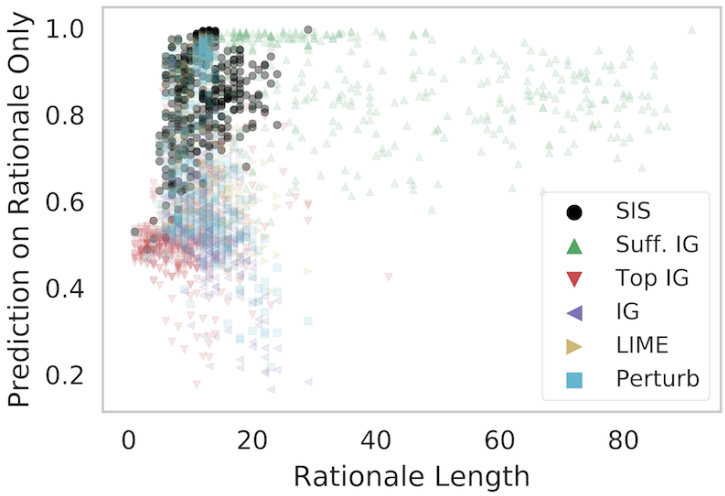


Figure 2-15: Prediction on rationale only (all other bases masked) vs. rationale length (number of bases) for various methods in the TF binding task.

between the rationale R and known motif M as:

$$\text{Div}(R, M) = \sum_i D_{\text{KL}}(R_i || M_i)$$

where $D_{\text{KL}}(R_i||M_i) = \sum_j R_i(j) \log \frac{R_i(j)}{M_i(j)}$ is the Kullback-Leibler (KL) divergence from M_i to R_i, and M_i and R_i are distributions over bases (`A`, `C`, `G`, `T`) at position i. Note that as R and M become more dissimilar, $\text{Div}(R, M)$ increases. We ensure $M_{ij} > 0 \; \forall \; i, j$ so D_{KL} is always finite. Figure 2-16 shows the divergence of rationales produced by **SIScollection** is significantly lower than that of rationales identified using other methods (Wilcoxon $p \leqslant 1\text{e}{-5}$ in all cases). SIS is thus more effective at uncovering these underlying biological principles than the alternative methods we explored.

2.7 MNIST Digit Classification

In this section, we apply SIS to interpret a 10-way convolutional neural network (CNN) classifier trained on the MNIST handwritten digits data [25]. In addition to interpreting the classifier's decisions on correctly classified examples, we see how

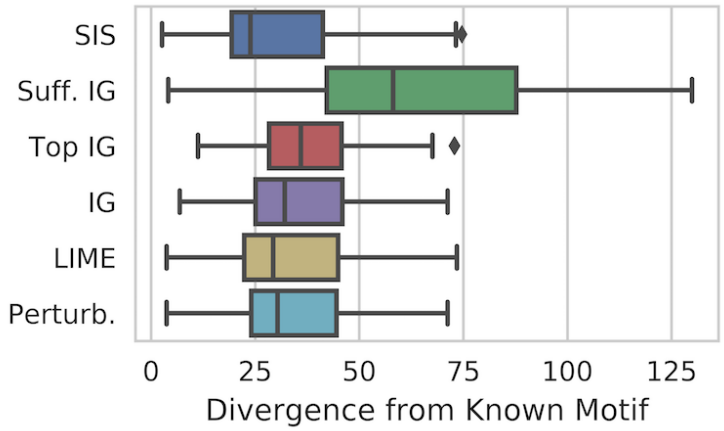


Figure 2-16: KL divergence between JASPAR motifs (known ground truth) and rationales found via various methods. Shown are results for 422 TF datasets (each one summarized by median divergence).

.

SIS can be further employed to understand the basis for the CNN's misclassifications (Section 2.7.2). In this application, we also observe the effect of local minima in the backward selection phase of the SIS procedure and show how our method facilitates minimality of the resulting rationales (Section 2.7.3).

2.7.1 Dataset and Model Details

The MNIST database of handwritten digits contains 60k training images and 10k test images [25]. All images are 28x28 grayscale, and we normalize them such that all pixel values are between 0 and 1. We train a simple 10-way CNN to classify the images using the same architecture as that provided in the Keras MNIST CNN example.[5] The architecture is defined as follows:

1. **Input**: (28 x 28 x 1) image, all values $\in [0, 1]$
2. **Convolutional Layer 1**: Applies 32 3x3 filters with ReLU activation
3. **Convolutional Layer 2**: Applies 64 3x3 filters, with ReLU activation

[5] `http://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py`

4. **Pooling Layer 1**: Performs max pooling with a 2x2 filter and dropout probability 0.25

5. **Dense Layer 1**: 128 neurons, with ReLU activation and dropout probability 0.5

6. **Dense Layer 2**: 10 neurons (one per digit class), with softmax activation

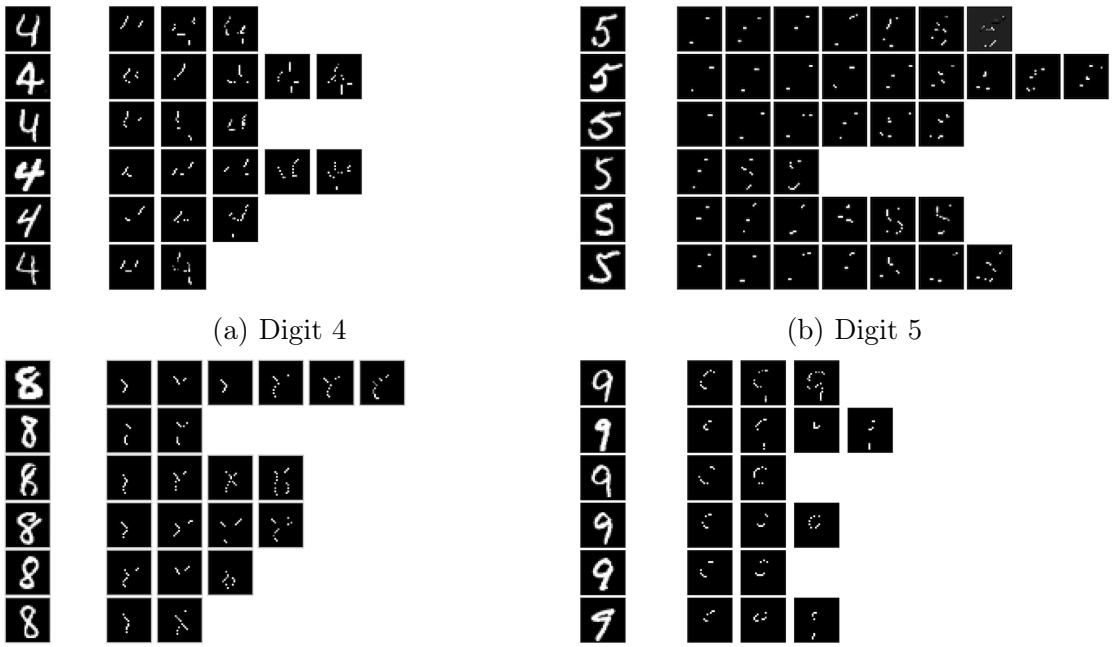
The Adadelta optimizer [50] is used to minimize cross-entropy loss on the training set. The final model achieves 99.7% accuracy on the train set and 99.1% accuracy on the held-out test set.

2.7.2 Applying SIS to Interpret Image Classifiers

When applying SIS to interpret the CNN's classification of MNIST handwritten digits, we only consider predicted probabilities for one class of interest at a time and always set $\tau = 0.7$ as the probability threshold for deciding that an image belongs to the class. We then extract the SIS-collections of all corresponding MNIST test set examples. Examples of the complete SIS-collection corresponding to randomly chosen digits are shown in Figure 2-17.

We also employ the SIS procedure to rationalize the CNN's misclassifications. We explore misclassifications of natural images in the MNIST test set as well as adversarially modified images. Figure 2-18a shows two (unmodified) MNIST digits whose true labels are 5 but which are misclassified by the CNN as 6 and 0, respectively. The SIS-collections depicted for these digits immediately enable us to understand the basis for why the misclassifications occur.

Figure 2-18b illustrates how the SIS-collection drastically changes for an example of a correctly-classified 9 that has been adversarially manipulated [6] to become confidently classified as the digit 4. Although a visual inspection of the perturbed image does not really reveal exactly how it has been manipulated, it becomes immediately clear from the SIS-collection for the adversarial image. The SIS-collections show that the perturbation modifies pixels in such a way that input patterns similar



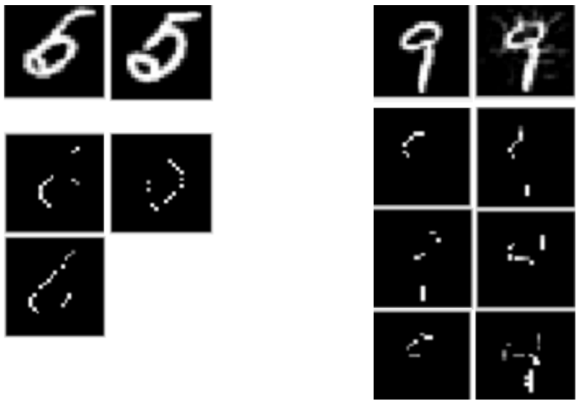
(a) Digit 4

(b) Digit 5

(c) Digit 8

(d) Digit 9

Figure 2-17: Visualization of SIS-collections identified from MNIST digits **(a)** 4, **(b)** 5, **(c)** 8, and **(d)** 9 that are confidently classified by the CNN. For each class, six examples were chosen randomly. For each example, we show the original image (left) and the complete set of sufficient input subsets identified for that example (remaining images in each row). Each individual SIS depicted satisfies $f(\mathbf{x}_S) \geqslant 0.7$ for that class.



(a) (b)

Figure 2-18: **(a)** SIS for digits 5 that are misclassified as 6 (1st column) and as 0 (2nd column). **(b)** SIS for correctly classified 9 (1st column) and when adversarially perturbed toward class 4 (2nd column).

to the typical SIS-collection for a 4 (Figure 3-2) become embedded in the image. The adversarial manipulation was done using the Carlini-Wagner L_2 (CW2) attack[6] [6] with a confidence parameter of 10. The CW2 attack tries to find the minimal change to the image, with respect to the L_2 norm, that will lead the image to be misclassified. Carlini and Wagner [5] demonstrate it to be one of the strongest extant adversarial attacks.

2.7.3 Local Minima in Backward Selection

Figure 2-19 demonstrates an example MNIST digit for which there exists a local minimum in the backward selection phase of our algorithm to identify the initial SIS. Note that if we were to terminate the backward selection as soon as predictions drop below the decision threshold, the resulting SIS would be overly large, violating our minimality criterion. It is also evident from Figure 2-19 that the smaller-cardinality SIS in (d), found after the initial local optimum in (c), presents a more interpretable input pattern that enables better understanding of the core motifs influencing our classifier's decisions. To avoid suboptimal results, it is important to run a complete backward selection sweep until the entire input is masked before building the SIS

[6]Implemented in the cleverhans library of Papernot et al. [34]

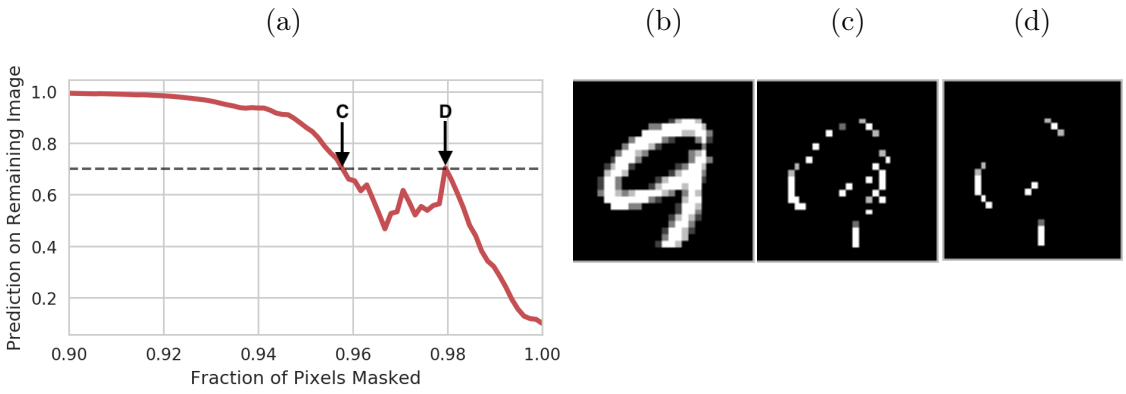


Figure 2-19: **(a)** Prediction on remaining image as pixels are masked during backward selection, when our CNN classifier is fed the MNIST digit in (b). The dashed line depicts the threshold $\tau = 0.7$. **(b)** Original image (class 9). **(c)** SIS if backward selection were to terminate the first time prediction on remaining image drops below 0.7, corresponding to point **C** in (a) (CNN predicts class 9 with probability 0.700 on this SIS). **(d)** Actual SIS produced by our **FindSIS** algorithm, corresponding to point **D** in (a) (CNN predicts class 9 with probability 0.704 on this SIS).

upward, as done in our **SIScollection** procedure (Section 2.3).

2.8 Discussion

In this chapter, we introduce the idea of interpreting black-box decisions on the basis of *sufficient input subsets* – minimal input patterns that alone provide sufficient evidence to justify a particular decision. Our SIS method allows a practitioner to obtain explanations for a machine learning model's decisions and to interrogate its behavior. Our methodology is easily understood by non-experts, applicable to all ML models without any additional training steps, and remains fully faithful to the underlying model without making approximations. Comparing to existing methods, we show that our SIS method produces rationales that are both minimal *and* meet our sufficiency criterion, ensuring that the model arrives at the same prediction given just the rationale. Further, we show how our method can be used not only to explain decisions that were made correctly, but also to understand the basis for a model's misclassifications. In Chapter 3, we adopt the SIS method to gain global insights into the general principles governing the decisions made by an ML model.

Chapter 3

Clustering SIS for Global Insights

3.1 Introduction

In Chapter 2, we introduced the sufficient input subsets (SIS) interpretability method, which produces local explanations for the decisions made by black-box functions. The explanations are comprised of sparse subsets of the features in the original input whose values form the basis for the model's decision. These local explanations are useful to understand why a model makes certain decisions on singular examples. In this chapter, we extend the SIS approach to extract global insights into the behavior of the model. In particular, we cluster the sufficient input subsets stemming from many examples on which the model makes the same decision. The resulting clusters enable us to visualize the distinct feature patterns the model has learned to associate with the decision (e.g. the features learned by a CNN to discriminate a handwritten digit 4 from other digits, as in Section 3.2.3).

Furthermore, we can adopt this SIS clustering methodology to reveal differences between two models trained on the same task. Jointly clustering the SIS stemming from both models allows us to inspect differences in how the models operate. We show that two models that achieve similar accuracy may make the same decisions for wildly different reasons. This kind of analysis can be used to decide which of a set of models may be most desirable to select for a system in practice. Our technique enables practitioners to discover such insights.

3.2 Clustering SIS for General Insights

Identifying the different input patterns that justify a decision can help us better grasp the general operating principles of a model. To this end, we cluster all of the sufficient input subsets produced by our SIS method applied across a large number of examples that receive the same decision by a particular model. In our work, we cluster the sufficient input subsets using DBSCAN [15], a widely applicable algorithm that only requires specifying pairwise distances between the SIS. This allows us to choose a suitable distance metric between sufficient input subsets depending on the particular domain.

3.2.1 Clustering SIS from Sentiment Predictors

We first cluster the sufficient input subsets found across held-out[1] beer reviews (Test fold in Table 2.1) that received positive aroma predictions from our LSTM model (model details in Section 2.5.1). The distance between two SIS is taken here as the Jaccard (intersection over union) distance between their bag of words representations, S_1 and S_2:

$$D(S_1, S_2) = 1 - \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

Table 3.1 shows three resulting clusters containing phrases that the LSTM has learned to associate with positive aromas in the absence of other context. The full clustering for SIS from beer reviews with strong positive predicted sentiment can be found in Table A.1 (strong negative predicted sentiment in Table A.2).

3.2.2 Clustering SIS from TF Binding Classifiers

We next apply our clustering procedure to the sufficient input subsets found by our method across all test-set DNA sequences which the CNN model (Section 2.6.1) predicts would be bound by some transcription factor (see Section 2.6). In this application, the pairwise distance between two sufficient input subsets is taken to be

[1]For experiments involving clustering and/or comparing different models, we use examples drawn from the Test fold (instead of Annotation fold, see Table 2.1) to consider a larger number of examples.

Table 3.1: Three clusters of SIS extracted from beer reviews with positive CNN aroma predictions. Each row shows four most frequent unique SIS in a cluster (each SIS shown as ordered word list with text-positions omitted). Each unique SIS can be present many times in one cluster.

Cluster	**SIS #1**	**SIS #2**	**SIS #3**	**SIS #4**
C_1	smell amazing wonderful	nice wonderful nose	wonderful amazing	amazing amazing
C_2	grapefruit mango pineapple	pineapple grapefruit pineapple grapefruit	hops grapefruit pineapple floyds	mango pineapple incredible
C_3	creme brulee brulee	creme brulee decadent	incredible creme brulee	creme brulee exceptional

the Levenshtein (edit) distance between the string representations of the masked sequences (where non-SIS characters are masked with `N` as in Section 2.6.1).

Figure 3-1 shows the clusters for a particular transcription factor (MAFF) for which two SIS clusters were found, aligned with the known motif from JASPAR [29] for this TF (discussion of JASPAR motifs in Section 2.6.3). Additional SIS in each of the clusters are given in Table 3.2. Notably, we find that despite contiguity not being enforced in our algorithm, each cluster is comprised of short sequences that clearly capture different aspects of the underlying DNA motif known to bind this TF. This result suggests that when the models are expected to behave according to some underlying scientific principles (e.g. those governing DNA transcription factor binding, as captured by the motif), the SIS clustering approach presented here is able to recover them. Had the motif not been known *a priori*, our approach would have enabled us to gain insight into which DNA sequence positions are critical for DNA-TF binding to occur.

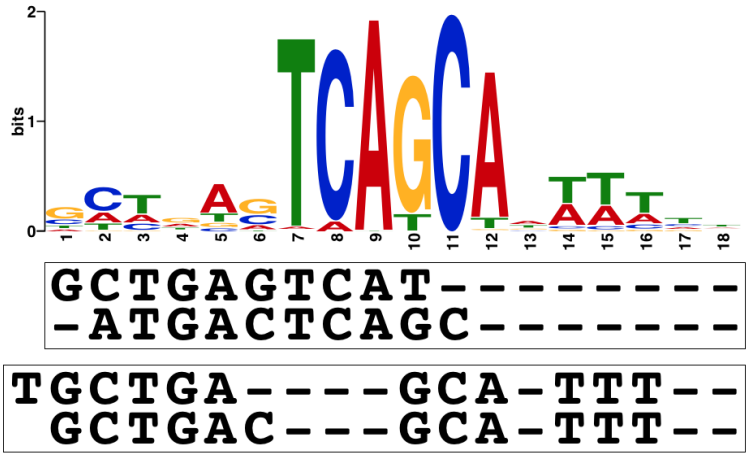


Figure 3-1: Known JASPAR motif (top) and alignment with cluster modes (bottom) for two SIS clusters identified by DBSCAN for one particular TF (MAFF). Additional items in clusters given in Table 3.2.

Table 3.2: Two clusters of SIS resulting from DBSCAN clustering on SIS stemming from CNN predicting binding for a particular transcription factor (MAFF). Most frequently-occurring SIS are each shown for each cluster. Frequency indicates number of times that SIS was observed.

SIS	Freq.
GCTGAGTCAT	197
ATGACTCAGC	185
GCTGAGTCA-C	83
GCTGAGTCAC	53
GCTGACTCAGCA	42

SIS	Freq.
TGCTGA--GCA-TTT	12
GCTGAC--GCA-TTT	8
TGCTGAC--GCA-TT	6
TGCTGAC--GCA-AA	5
TGCTGAC--GCA-AT	4

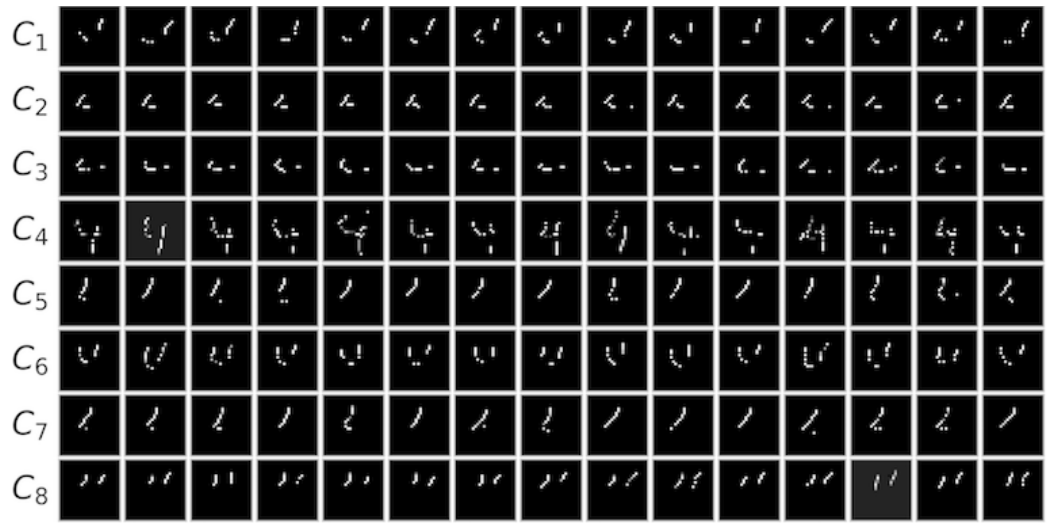


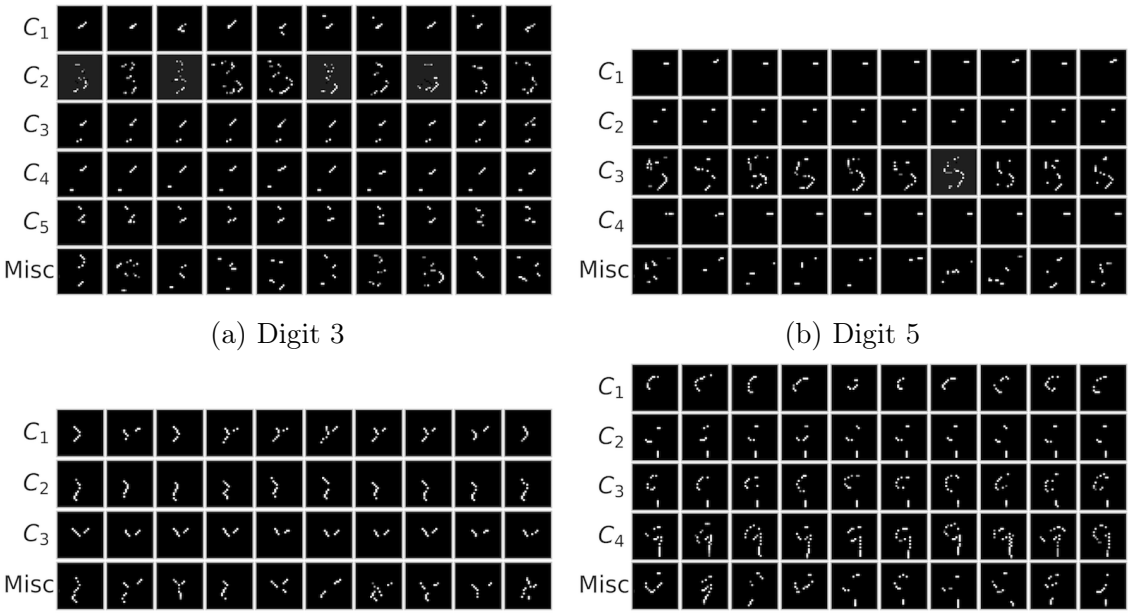
Figure 3-2: Eight clusters of SIS identified from examples of digit 4. Each row contains fifteen random SIS from a single cluster.

3.2.3 Clustering SIS from MNSIT Digit Classifiers

Finally, we apply our clustering methodology the sufficient input subsets found across all MNIST test set examples that are confidently identified by the CNN (Section 2.7) as a particular class. Pairwise distances are here defined as the *energy distance* [37] over pixel locations between two sufficient input subsets (see Section A.2.1 for details). Figure 3-2 depicts the SIS clusters identified for digit 4. These clusters reveal distinct feature patterns learned by the CNN to distinguish digit 4 from other digits, which are clearly present in the vast majority of test set images confidently classified as a 4. For example, cluster C_8 depicts parallel slanted lines, a pattern that never occurs in other digits. We repeat this analysis for additional digit classes, and results are shown in Figure 3-3.

3.3 Understanding Differences Between Models

The general insights revealed by our SIS clustering methodology can also be used to compare and contrast the operating behaviors of different models trained for the same task. In this section, we demonstrate this approach in two of our settings: training a text CNN to compare to our existing LSTM that predicts sentiment in beer reviews



(a) Digit 3

(b) Digit 5

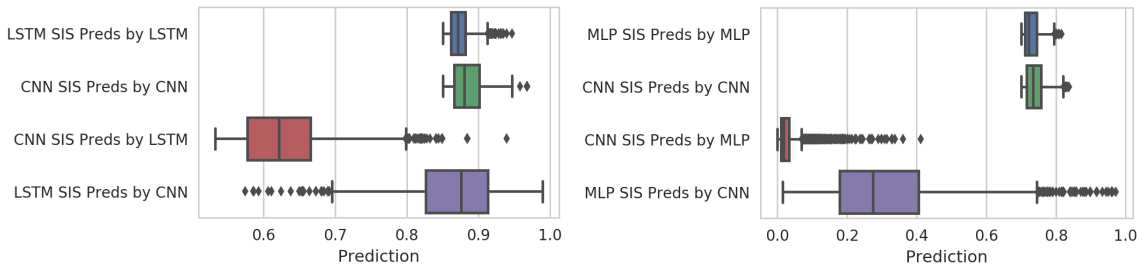
(c) Digit 8

(d) Digit 9

Figure 3-3: Clustering all the SIS found for digits **(a)** 3, **(b)** 5, **(c)** 8, and **(d)** 9 under the CNN model. Each row contains images drawn from one cluster. The bottom row ("Misc") contains a sample of miscellaneous SIS not assigned to any cluster by DBSCAN.

(Section 2.5) and training a simple feed-forward neural network to compare to our existing CNN to classify MNIST digits (Section 2.7).

Both networks exhibit similar performance (i.e. accuracy) in each task, so it is not immediately clear which model would be preferable to use in practice. We first determine whether the SIS extracted under one model are sufficient for the other model to arrive at the same prediction. Figure 3-4 shows the SIS extracted under one model are typically insufficient to receive the same decision from the other model, suggesting that these models base their positive predictions on different evidence. We next adopt our joint SIS clustering methodology to expose the differences in the SIS from each of the architectures in each of these applications in turn.



(a) (b)

Figure 3-4: Predictions by one model on the SIS extracted from the other model in: **(a)** beer reviews with positive LSTM/CNN aroma predictions, and **(b)** MNIST digits confidently classified as 4 by CNN/MLP.

3.3.1 Understanding Differences Between Sentiment Predictors

In addition to the LSTM (see Section 2.5.1), we train a convolutional neural network (CNN) on the same sentiment analysis task (on the aroma aspect). The text CNN architecture is as follows:

1. **Input/Embeddings Layer**: Sequence with 500 timesteps, the word at each timestep is represented by a (learned) 100-dimensional embedding
2. **Convolutional Layer 1**: Applies 128 filters of window size 3 over the sequence, with ReLU activation
3. **Max Pooling Layer 1**: Max-over-time pooling, followed by flattening, to produce a $(128,)$ representation
4. **Dense**: 1 neuron (sentiment output), sigmoid activation

Note that a new set of embeddings is learned with the CNN. As with the LSTM model, we use Adam [23] to minimize MSE on the training set. For the aroma aspect, this CNN achieves 0.016 (0.850), 0.025 (0.748), 0.026 (0.741), 0.014 (0.662) MSE (and Pearson ρ) on the Train, Validation, Test, and Annotation sets, respectively. We note that this performance is very similar to that from the LSTM (Table 2.1).

We apply our **SIScollection** procedure to extract the SIS-collections from all applicable test examples using the text CNN, as in Section 2.5. Figure 3-4a shows

Table 3.3: Joint clustering of the SIS from beer reviews predicted to have positive aroma by LSTM or CNN. Dashes are used in clusters with under four unique SIS. Percentages quantify SIS per cluster stemming from the LSTM.

Cluster	**LSTM**	**SIS #1**	**SIS #2**	**SIS #3**	**SIS #4**
C_1	0%	delicious	-	-	-
C_2	0%	very nice	-	-	-
C_3	20%	rich chocolate	very rich	chocolate complex	smells rich
C_4	33%	oak chocolate	chocolate raisins raisins oak bourbon	chocolate oak	raisins chocolate
C_5	70%	complex aroma	aroma complex peaches complex	aroma complex interesting cherries	aroma complex

the predictions from one model (LSTM or CNN) when fed input examples that are SIS extracted with respect to the *other* model (for reviews predicted to have positive sentiment toward the aroma aspect). Since the word embeddings are model-specific, we embed each SIS using the embeddings of the model making the prediction (note that while the embeddings are different, the vocabulary is the same across the models).

In Table 3.3, we show five example clusters (and cluster composition) resulting from clustering the combined set of all sufficient input subsets extracted by the LSTM and CNN on reviews in the test set for which a model predicts positive sentiment toward the aroma aspect. The complete clustering on reviews receiving positive sentiment predictions is shown in Table A.3 (Table A.4 for reviews receiving negative sentiment predictions). These results suggest that the text CNN tends to learn localized (unigram/bigram) word patterns, while the LSTM identifies more complex multi-word interactions that seem more relevant to the target aroma value. Many SIS from the CNN are simply phrases with universally-positive sentiment, indicating this model may be less able to distinguish between positive sentiment toward aroma vs. other aspects such as appearance or palate.

3.3.2 Understanding Differences Between MNIST Classifiers

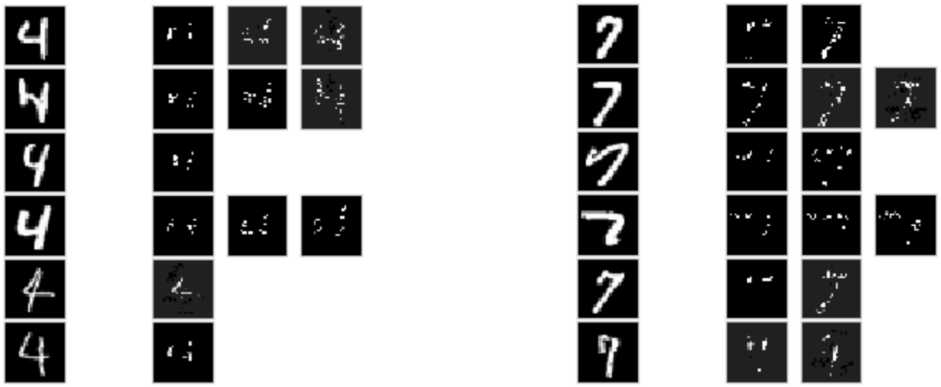
We next use SIS and our clustering procedure to understand and visualize differences in features learned by two different models trained on the same MNIST digit classification task. In addition to the previously described CNN model (see Section 2.7.1), we also trained a simple multilayer perceptron (MLP) on the same task. The MLP architecture is as follows:

1. **Input**: 784-dimensional (flattened) image, all values $\in [0, 1]$
2. **Dense Layer 1**: 250 neurons, ReLU activation, and dropout probability 0.2
3. **Dense Layer 2**: 250 neurons, ReLU activation, and dropout probability 0.2
4. **Dense Layer 3**: 10 neurons (one per digit class), with softmax activation

As with the CNN, Adadelta [50] is used to minimize cross-entropy loss on the training set. The final MLP model achieves 99.7% accuracy on the train set and 98.3% accuracy on the test set, which is close to the performance of the CNN (see Section 2.7.1).

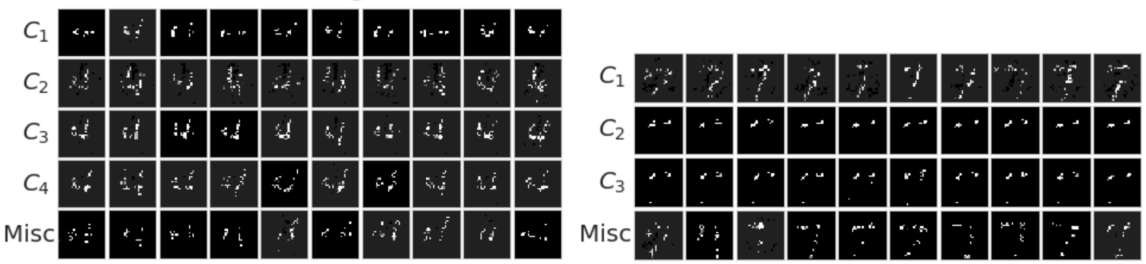
We apply the same procedure as in Section 2.7 to extract the SIS-collections from MNIST test images using the MLP. Figure 3-5 shows some examples of SIS-collections extracted for MNIST digits 4 and 8 from the MLP architecture. We also cluster the SIS-collections extracted from the MLP (as in Section 3.2.3). Clusters for two classes are shown in Figure 3-6.

To understand and visualize the differences between the features learned by each model to classify MNIST digits, we combine all SIS (from both models, for a particular class) and apply our joint SIS clustering procedure. In the resulting clustering (for digit 4 as shown in Figure 3-7), we list what percentage of the SIS in each cluster stem from the CNN vs. the MLP. Most clusters contain examples purely from a single model, indicating the two models have learned to associate different feature patterns with class 4. Evidently, the CNN bases its confidence primarily on spatially-contiguous strokes comprising only a small portion of each image. Classifications by the MLP instead seem to be based on pixels located throughout the digit, demonstrating this model relies more on the global shape of the handwriting. Thus, this



(a) Digit 4 (b) Digit 7

Figure 3-5: Visualization of SIS-collections identified for MNIST digits **(a)** 4 and **(b)** 7 under the MLP model. For each class, six examples were chosen randomly. For each example, we show the original image (left) and the complete set of sufficient input subsets identified for that example (remaining images in each row). Note that each individual SIS satisfies $f(\mathbf{x}_S) \geqslant \tau$ for that class. Compare to the SIS extracted from the CNN architecture (Figure 2-17).



(a) Digit 4 (b) Digit 7

Figure 3-6: Clustering all the SIS identified by our method on digits **(a)** 4 and **(b)** 7 under the MLP model (Section 3.3.2). Each row contains images drawn from one cluster. The bottom row ("Misc") contains a sample of miscellaneous SIS not assigned to any cluster by DBSCAN. Compare to the SIS clustering from our CNN model (Figure 3-3).

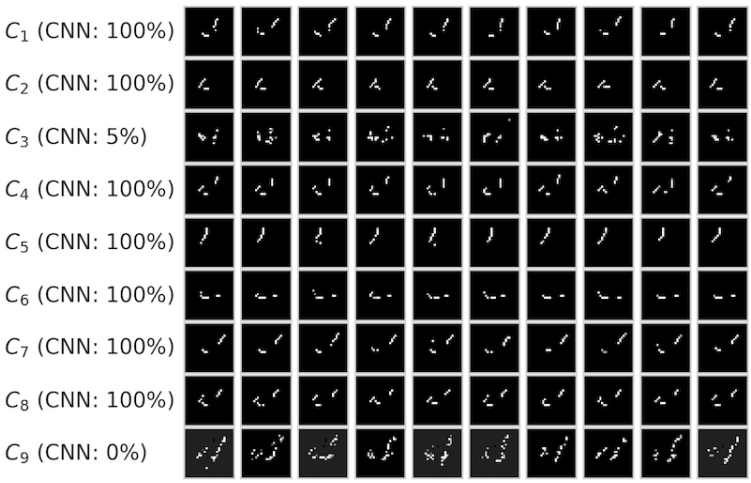


Figure 3-7: Jointly clustering the MNIST digit 4 SIS from CNN and MLP. We list the percentage of SIS in each cluster stemming from the CNN (rest from MLP).

result suggests that the CNN may be more susceptible to mistaking other (non-digit) handwritten characters for 4s if they happen to share some of the same strokes.

3.4 Discussion

In this chapter, we show how SIS can be clustered to gain global insights into a model's general operating behavior. The resulting clusters reveal distinct feature patterns that the model has learned to associate with a particular decision. Moreover, given multiple models of comparable accuracy, we show how these clusters can be used to uncover critical operating differences between the models. For example, our analysis can reveal which model is more susceptible to spurious training data correlations or may generalize worse to counterfactual inputs that lie outside the data distribution. As with the SIS procedure, our clustering methodology can be easily adopted into a wide range of applications.

Chapter 4

Conclusion

This chapter contains a summary of the contributions of this thesis as well as directions for future work in this area.

4.1 Summary of Contributions

In this thesis, we contribute a novel interpretability method based on the premise of *sufficient input subsets*. In this framework, we presume that a model f makes a decision $f(\mathbf{x})$ on a multi-dimensional input $\mathbf{x}$, such as a sequence of characters. Our method finds a collection of sparse, minimal subsets of the features in $\mathbf{x}$ such that the model is able to make the same prediction using only the features in any of these subsets alone, without information about any other features' values. These sufficient input subsets can be thought of as explanations (rationales) for the model's decisions.

In **Chapter 2**, we introduce the sufficient input subsets (SIS) method for black-box model interpretability. We describe the **SIScollection** algorithm which produces rationales for a decision $f(\mathbf{x})$ by applying backward selection locally to the features in $\mathbf{x}$. We apply SIS to interpret the decision-making of neural network models in three different domains: sentiment prediction in natural language, a computational biology task of predicting transcription factor binding, and a vision task involving handwritten digit classification. We show how the method can be used to explain the models' decisions that were made correctly and to understand the basis of misclassifications.

Unlike many other methods, SIS can be applied to interpret *any* black-box function f with no gradient information or assumptions of differentiability. Further, no auxiliary explanation model is required to produce the explanations, and hence, the method is completely faithful to the underlying function f. The SIS method is widely applicable to a number of domains where producing explanations for decisions made by black-box models is imperative. These applications include critical decisions made by machine learning models in screening loan applicants [42], in recidivism prediction [24], and in determining whether patients have a particular disease [18].

We compare SIS to a number of alternative methods for explaining these models and show that our method more effectively produces subsets of features that are both minimal *and* sufficient to allow the model to reach the same decision given that subset alone. Moreover, we show that in the biological task where ground-truth for the model's behavior is known, subsets from our method more accurately reflect the underlying biological principles governing transcription factor and DNA binding than rationales produced by the alternative methods. This finding suggests that our SIS approach can be applied to discover new scientific insights: highly accurate machine learning models trained on scientific datasets may be interpreted in order to learn about the underlying principles.

In **Chapter 3**, we show how sufficient input subsets can not only allow practitioners to explain model behavior locally (on particular examples), but also gain insight into global behavior governing the model's decision-making. Our approach is based on clustering the sufficient input subsets taken from a large number of examples on which the model makes the same prediction. We demonstrate the utility of this clustering methodology in our three application domains (sentiment prediction, transcription factor binding, and digit classification). Furthermore, we adopt this methodology to reveal operating differences between two models trained on the same task. We show that two models that achieve similar accuracy may make the same decisions for very different reasons. Our techniques enable practitioners to uncover and visualize these differences in model behavior.

Together, the methods introduced in this thesis are broadly applicable to interpret

and compare machine learning models in a wide range of domains in practice. Such applications of this work include determining whether a model can be trusted in critical applications, as a tool in scientific research, for developing new ML architectures, and as a basis for model selection.

4.2 Future Work

The arena of model interpretability is ripe with opportunities for future work. Such work along the lines of the SIS method presented in Chapter 2 includes algorithmic improvements to decrease the complexity of the search procedure as well as extensions to overlapping sufficient input subsets. Other work could include seeking theoretical guarantees about the subsets found by the algorithm, which are currently only known for backward selection in certain linear settings [13]. Finally, one may try to use SIS along with a known ground truth in an active learning setting, encouraging a model to make decisions in accordance with the known factors. SIS can provide a way to measure the model's alignment with the ground truth.

There are also many applications and extensions of the SIS clustering methodology presented in Chapter 3. For example, one may evaluate the robustness of an architecture in a particular task by retraining an identical model with different random initialization of parameters. Consistency in SIS revealed by our SIS clustering approach would suggest that the rationales behind the architecture's decision-making are robust to retraining the model. Similar work can also involve extending the SIS clustering technique as a way to quantify model uncertainty. Finally, as we show that the backward selection strategy employed by SIS is well-suited to biological data, future work can involve adopting SIS to understand and critique state of the art neural networks trained on biological data (e.g. [3, 4]).

Appendix A

Details for Sufficient Input Subsets Experiments

This appendix contains additional details and results for the material in Chapters 2 and 3. Further experiments and results can be found in the Supplementary Information of Carter et al. [7].

A.1 Additional Details of Sentiment Analysis Experiments

Here, we provide additional details of our experiments applying SIS to interpret LSTM models predicting sentiment in beer reviews (Section 2.5).

A.1.1 Imputation Strategies: Mean vs. Hot-deck

In Section 2.3, we discuss the problem of masking input features. Here, we show that the mean-imputation approach (in which missing inputs are masked with a mean embedding, taken over the entire vocabulary) produces a nearly identical change in prediction to a nondeterministic hot-deck approach (in which missing inputs are replaced by randomly sampling feature-values from the data). Figure A-1 shows the change in prediction $f(\mathbf{x}\backslash\{i\}) - f(\mathbf{x})$ by both imputation techniques after drawing

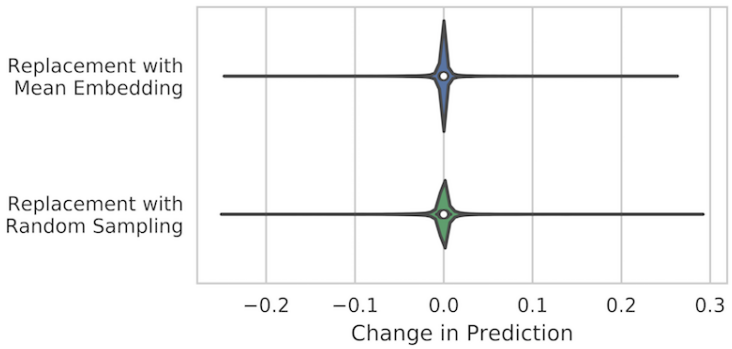


Figure A-1: Change in prediction ($f(\mathbf{x}\backslash\{i\}) - f(\mathbf{x})$) after masking a randomly chosen word with mean imputation or hot-deck imputation. 10,000 replacements were sampled from the aroma beer reviews training set.

a training example $\mathbf{x}$ and word $x_i \in \mathbf{x}$ (both uniformly at random) and replacing x_i with either the mean embedding or a randomly selected word (drawn from the vocabulary, based on counts in the training corpus). This procedure is repeated 10,000 times. Both resulting distributions have mean near zero ($\mu_{\text{mean-embedding}} = -7.0\text{e}{-4}$, $\mu_{\text{hot-deck}} = -7.4\text{e}{-4}$), and the distribution for mean embedding is slightly narrower ($\sigma_{\text{mean-embedding}} = 0.013$, $\sigma_{\text{hot-deck}} = 0.018$). Because we find that these two imputation approaches perform equally well on average, we elect to use mean-imputation as our preferred method for masking information about features' values when applying SIS.

We also explored other options for masking word information, e.g. replacement with a zero embedding, replacement with the learned <PAD> embedding, and simply removing the word entirely from the input sequence, but each of these alternative options led to undesirably larger changes in predicted values as a result of masking, indicating they appear more informative to f than replacement via the feature-mean.

A.1.2 Additional Results for Aroma Aspect

This section includes additional results applying our SIS clustering methodology (Chapter 3) to interpret LSTM sentiment predictors. Here, we present the full SIS clustering for both reviews with strong positive and strong negative predicted sentiment (Section 3.2.1).

Table A.1: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have positive aroma by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	smell amazing wonderful	2	nice wonderful nose	2	wonderful amazing	2	amazing amazing	2
C_2	grapefruit mango pineapple	2	pineapple grapefruit pineapple grapefruit	1	hops grapefruit pineapple floyds	1	mango pineapple incredible	1
C_3	nice smell citrus nice grapefruit taste	1	smell great complex ripe taste	1	nice smell nice hop smell pine taste	1	love nice nice smell bliss taste	1
C_4	fresh great fantastic taste	1	rich great fantastic hoped	1	fantastic cherries fantastic	1	everyone great snifters fantastic	1
C_5	awesome bounds	1	awesome grapefruit awesome	1	awesome awesome pleasing	1	awesome nailed nailed	1
C_6	creme brulee brulee	3	creme brulee decadent	1	incredible creme brulee	1	creme brulee exceptional	1
C_7	oak vanilla chocolate cinnamon vanilla oak love	1	dose oak chocolate vanilla acidic	1	vanilla figs oak thinner great	1	chocolate aroma oak vanilla dessert	1

Table A.2: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have negative aroma by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	awful	15	skunky skunky	9	skunky t	7	skunky taste	6
C_2	garbage	3	taste garbage	1	garbage avoid	1	garbage rice	1
C_3	vomit	16	-	-	-	-	-	-
C_4	gross rotten	1	rotten forte	1	awkward rotten	1	rotten offputting	1
C_5	rancid horrid	1	rancid t	1	rancid	1	rancid avoid	1
C_6	rice t rice	2	rice rice	1	rice tasteless	1	budweiser rice	1

A.1.3 Understanding Differences Between Sentiment Predictors

We also include the full joint SIS clustering (clustering SIS from LSTM and text CNN models together) for reviews with strong positive and strong negative predicted sentiment (Section 3.3.1).

Table A.3: Joint clustering of the SIS extracted from beer reviews predicted to have positive aroma by LSTM or CNN model. Frequency indicates the number of occurrences of the SIS in the cluster. Percentages quantify SIS per cluster from the LSTM. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1 (LSTM: 20%)	rich chocolate	13	very rich	9	chocolate complex	5	smells rich	4
C_2 (LSTM: 21%)	great	248	amazing	119	wonderful	112	fantastic	75
C_3 (LSTM: 47%)	best smelling	23	pineapple mango	6	mango pineapple	6	pineapple grapefruit	5
C_4 (LSTM: 5%)	excellent	42	excellent flemish flemish	1	excellent excellent phenomenal	1	-	-
C_5 (LSTM: 33%)	oak chocolate	2	chocolate raisins raisins oak bourbon	1	chocolate oak	1	raisins chocolate	1
C_6 (LSTM: 5%)	goodness	19	watering goodness	1	-	-	-	-
C_7 (LSTM: 24%)	pumpkin pie	25	huge pumpkin aroma pumpkin pie	1	aroma perfect pumpkin pie taste	1	smell pumpkin nutmeg cinnamon pie	1
C_8 (LSTM: 5%)	jd	13	tremendous	8	tremendous jd	1	-	-
C_9 (LSTM: 40%)	brulee	14	creme brulee brulee	3	creme creme	1	creme brulee amazing	1
C_{10} (LSTM: 0%)	s wow	20	-	-	-	-	-	-
C_{11} (LSTM: 0%)	delicious	56	-	-	-	-	-	-
C_{12} (LSTM: 0%)	very nice	23	-	-	-	-	-	-
C_{13} (LSTM: 70%)	complex aroma	5	aroma complex peaches complex	1	aroma complex interesting cherries	1	aroma complex	1

Table A.4: Joint clustering of the SIS extracted from beer reviews predicted to have negative aroma by LSTM or CNN model. Frequency indicates the number of occurrences of the SIS in the cluster. Percentages quantify SIS per cluster from the LSTM. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1 (LSTM: 29%)	not	247	no	105	bad	104	macro	94
C_2 (LSTM: 100%)	gross rotten	1	-	-	-	-	-	-
C_3 (LSTM: 100%)	rotten garbage	1	-	-	-	-	-	-
C_4 (LSTM: 62%)	vomit	26	-	-	-	-	-	-
C_5 (LSTM: 21%)	budweiser	22	sewage budweiser	1	metal budweiser	1	budweiser budweiser budweiser	1
C_6 (LSTM: 100%)	garbage rice	1	-	-	-	-	-	-
C_7 (LSTM: 3%)	n't	19	adjuncts	14	n't adjuncts	1	-	-
C_8 (LSTM: 0%)	faint	82	-	-	-	-	-	-
C_9 (LSTM: 0%)	adjunct	42	-	-	-	-	-	-

A.2 Additional Details of MNIST Experiments

This section includes additional details of our SIS clustering experiments of MNIST digits (Sections 3.2.3 and 3.3.2).

A.2.1 Energy Distance Between MNIST Image SIS

To cluster SIS from the image data, we compute the pairwise distance between two sufficient input subsets S_1 and S_2 as the energy distance [37] between two distributions over the image pixel coordinates that comprise the SIS, X_1 and $X_2 \in \mathbb{R}^2$:

$$D(X_1, X_2) = 2 \cdot \mathbb{E}\,||X_1 - X_2|| - \mathbb{E}\,||X_1 - X_1'|| - \mathbb{E}\,||X_2 - X_2'|| \geqslant 0$$

Here, X_i is uniformly distributed over the pixels that are selected as part of the SIS subset S_i, X_i' is an i.i.d. copy of X_i, and $||\cdot||$ represents the Euclidean norm. Unlike a Euclidean distance between images, our usage of the energy distance takes into account distances between the similar pixel coordinates that comprise each SIS. The energy distance offers a more efficiently computable integral probability metric than the optimal transport distance, which has been widely adopted as an appropriate measure of distance between images.

Bibliography

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One*, 10(7): e0130140, 2015.

[2] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11:1803–1831, 2010.

[3] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*, 2019.

[4] Maxwell L Bileschi, David Belanger, Drew H Bryant, Theo Sanderson, Brandon Carter, D Sculley, Mark L DePristo, and Lucy J Colwell. Using deep learning to annotate the protein universe. *bioRxiv*, page 626507, 2019.

[5] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[7] Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. In *Artificial Intelligence and Statistics*, 2019.

[8] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[9] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, 2018.

[10] François Chollet et al. Keras. `https://keras.io`, 2015.

[11] ENCODE Project Consortium et al. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57, 2012.

[12] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, 2017.

[13] Abhimanyu Das and David Kempe. Algorithms for subset selection in linear regression. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008.

[14] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1996.

[16] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[17] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML*, 2017.

[18] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, 2018.

[21] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *NIPS Workshop: Interpreting, Explaining and Visualizing Deep Learning - Now what?*, 2017.

[22] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. In *International Conference on Learning Representations*, 2018.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[24] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. *The Quarterly Journal of Economics*, 133(1):237–293, 2018.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Empirical Methods in Natural Language Processing*, 2016.

[27] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv:1612.08220*, 2017.

[28] Z. C. Lipton. The mythos of model interpretability. In *ICML Workshop on Human Interpretability of Machine Learning*, 2016.

[29] Anthony Mathelier, Oriol Fornes, David J Arenillas, Chih-yu Chen, Grégoire Denay, Jessica Lee, Wenqiang Shi, Casper Shyr, Ge Tan, Rebecca Worsley-Hunt, et al. Jaspar 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic acids research*, 44(D1):D110–D115, 2015.

[30] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *IEEE International Conference on Data Mining*, pages 1020–1025, 2012.

[31] W. James Murdoch, Peter J. Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. In *International Conference on Learning Representations*, 2018.

[32] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007.

[33] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010.

[34] Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, Abhibhav Garg, and Yen-Chen Lin. cleverhans v2.0.0: an adversarial machine learning library. *arXiv:1610.00768*, 2017.

[35] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv:1704.01444*, 2017.

[36] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[37] Maria L Rizzo and Gábor J Székely. Energy distance. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(1):27–38, 2016.

[38] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[39] Ying Sha and May D. Wang. Interpretable predictions of clinical outcomes with an attention-based recurrent neural network. In *ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics*, 2017.

[40] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, 2017.

[41] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*, 2014.

[42] Justin A. Sirignano, Apaar Sadhwani, and Kay Giesecke. Deep learning for mortgage risk. *arXiv:1607.02470*, 2018.

[43] Alexander J Smola, SVN Vishwanathan, and Thomas Hofmann. Kernel methods for missing variables. In *Artificial Intelligence and Statistics*, 2005.

[44] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*, 2015.

[45] H. Strobelt, S. Gehrmann, H. Pfister, and A.M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, pages 667–676, 2018.

[46] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

[47] Florian Tramer, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, 2016.

[48] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[49] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *NIPS TIML Workshop*, 2017.

[50] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv:1212.5701*, 2012.

[51] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 2014.

[52] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121, 2016.

[53] Quanshi Zhang, Yu Yang, Ying Nian Wu, and Song-Chun Zhu. Interpreting CNNs via decision trees. *arXiv:1802.00121*, 2018.