Neural Belief Reasoner

Haifeng Qian
IBM Research, Yorktown Heights, NY, USA
qianhaifeng@us.ibm.com

Abstract

This paper proposes a new generative model called neural belief reasoner (NBR). It differs from previous models in that it specifies a belief function rather than a probability distribution. Its implementation consists of neural networks, fuzzy-set operations and belief-function operations, and query-answering, sample-generation and training algorithms are presented. This paper studies NBR in two tasks. The first is a synthetic unsupervised-learning task, which demonstrates NBR's ability to perform multi-hop reasoning, reasoning with uncertainty and reasoning about conflicting information. The second is supervised learning: a robust MNIST classifier for 4 and 9, which is the most challenging pair of digits. This classifier needs no adversarial training, and it substantially exceeds the state of the art in adversarial robustness as measured by the L_2 metric, while at the same time maintains 99.1% accuracy on natural images.

1 Introduction

It is a widely held hypothesis that bridging the gap between machine learning and rule-based reasoning would bring benefits to both domains. For rule-based systems, this could provide an elegant solution to automatically discover rules from observations, and could provide new approaches to reasoning with uncertainty. On the other hand, despite the phenomenal successes of deep learning, neural networks tend to have poor robustness and interpretability, both of which are nonissue in rule-based systems. The robustness issue has recently been highlighted by the existence of adversarial examples in many systems. For example, even for the well-studied MNIST task, the state of the art in L_2 robustness is far from satisfactory and comes at a cost to accuracy on natural images.

Some of the works at the intersection of machine learning and reasoning will be reviewed in Section 5. A prominent one is Boltzmann machine and its variants [Salakhutdinov and Hinton, 2009], which combine neural networks and Markov random fields. A recent example is differentiable inductive logic programming [Evans and Grefenstette, 2018], which combines neural networks and logic programs.

This paper presents a new approach called neural belief reasoner (NBR), which combines neural networks and belief functions. Belief functions are a generalization of probability functions [Shafer, 1976], and have the advantage of modeling epistemic uncertainty, i.e., the lack of knowledge, and an elegant way of combining multiple sources of information. Despite these advantages, belief functions have seen much less adoption than mainstream methods like Bayesian networks and Markov random fields. NBR is built on two innovations: 1) using neural networks to represent fuzzy sets, and 2) using fuzzy sets to specify a belief function. From the machine-learning perspective, NBR is a new generative model that specifies a belief function rather than a probability distribution. From the rule-based perspective, NBR is a new method of reasoning with uncertainty that enables automatic discovery of non-symbolic rules from observations and that uses belief functions to model uncertainty.

The next section will define the model and present query-answering, sample-generation and training algorithms. Then Sections 3 and 4 demonstrate NBR's capabilities through two tasks. The first task is unsupervised learning: in a synthetic 11-bit world where only partial observations are available, an NBR model is trained and then answers queries. The queries involve multi-hop reasoning, reasoning with uncertainty and reasoning about conflicting information. The second task is supervised learning: a robust MNIST classifier for 4 and 9, which is the most challenging pair of digits. It sets a new state of the art in adversarial robustness as measured by the L_2 metric and maintains 99.1% accuracy on natural images.

2 Neural Belief Reasoner

Let's start with a restricted framework of reasoning in Section 2.1, which serves as a stepping stone towards NBR's definitions in Section 2.2 and algorithms after that.

2.1 Prototype with Classical Sets

Let U denote the sample space, i.e., the set of all possibilities. Consider a reasoning framework where a model is composed of K sets, $R_1, \cdots, R_K \subseteq U$, and each R_i is annotated with a scalar $0 \leq b_i \leq 1$. Let's interpret each R_i as a logic rule: an outcome $x \in U$ is said to satisfy R_i if and only if $x \in R_i$. Let's interpret b_i as the belief in R_i.

Define vector $\mathbf{y} \triangleq (y_1, \cdots, y_K)$ where entries are 0 or 1. Define intersection sets $S_{\mathbf{y}} \triangleq \bigcap_{1 \leq i \leq K | y_i = 1} R_i$, and define

$S_{(0,\cdots,0)} \triangleq U$. Intuitively each $S_{\mathbf{y}}$ contains outcomes that satisfy a subset of the K rules as selected by $\mathbf{y}$. Define scalar function $p(\mathbf{y}) \triangleq \prod_{i=1}^{K} (b_i \cdot y_i + (1-b_i)\cdot(1-y_i))$.

Let 2^U denote the power set of U. Define the following function from 2^U to $\mathbb{R}$: $m(\emptyset) \triangleq 0$; for $A \neq \emptyset$,

$$m(A) \triangleq \frac{\sum_{\mathbf{y}|S_{\mathbf{y}}=A} p(\mathbf{y})}{1-\sum_{\mathbf{y}|S_{\mathbf{y}}=\emptyset} p(\mathbf{y})} \tag{1}$$

It is straightforward to verify that $\sum_{A\subseteq U} m(A) = 1$. Therefore this function $m(\cdot)$ satisfies the requirements to be a basic probability assignment [Shafer, 1976]. Hence this function $m(\cdot)$ uniquely specifies a belief function over U [Shafer, 1976]: $\mathrm{Bel}(A) = \sum_{B\subseteq A} m(B), \forall A \subseteq U$.

Intuitively this framework considers 2^K possible worlds: each world corresponds to each $\mathbf{y}$, and in each world a subset of the K rules, as selected by $\mathbf{y}$, exist. Each satisfiable world, i.e., where $S_{\mathbf{y}} \neq \emptyset$, is assigned a mass that is proportional to $p(\mathbf{y})$, the product of b_i's for rules that are present and $(1-b_i)$'s for rules that are absent. Each unsatisfiable world, i.e., where $S_{\mathbf{y}} = \emptyset$, is assigned a mass of zero. The total mass is one, which is achieved through the denominator in (1) that is essentially Dempster's rule of combination [Shafer, 1976].

With a belief function defined, this framework is able to answer queries. Similar to conditional probabilities in traditional models, it answers with conditional belief functions. Given a condition $C \subseteq U$ and a proposition $Q \subseteq U$, the conditional belief and conditional plausibility are

$$\begin{aligned}\mathrm{Bel}(Q \mid C) &= \frac{\mathrm{Bel}\left(Q\cup\overline{C}\right)-\mathrm{Bel}\left(\overline{C}\right)}{1-\mathrm{Bel}\left(\overline{C}\right)}\\ &= 1-\frac{\sum_{\mathbf{y}|S_{\mathbf{y}}\cap C\cap\overline{Q}\neq\emptyset} p(\mathbf{y})}{\sum_{\mathbf{y}|S_{\mathbf{y}}\cap C\neq\emptyset} p(\mathbf{y})}\\ \mathrm{Pl}(Q\mid C) &= 1-\mathrm{Bel}\left(\overline{Q}\mid C\right)\\ &= \frac{\sum_{\mathbf{y}|S_{\mathbf{y}}\cap C\cap Q\neq\emptyset} p(\mathbf{y})}{\sum_{\mathbf{y}|S_{\mathbf{y}}\cap C\neq\emptyset} p(\mathbf{y})}\end{aligned} \tag{2}$$

Intuitively, $\mathrm{Bel}(\cdot)$ quantifies the evidence that supports a proposition and $1-\mathrm{Pl}(\cdot)$ quantifies evidence against it.

2.2 Model Definitions

I now define the full form of NBR by generalizing the previous section in a number of ways: R's are replaced by fuzzy sets represented by neural networks; U becomes the latent space which is separated from observation space.

An NBR model has the following components and Figure 1 illustrates the architecture.

- Function $\mathbf{x} = F(\mathbf{z})$ where vector $\mathbf{x}$ is the observation variables and vector $\mathbf{z}$ is the latent variables.
- Function $\mathbf{r} = R(\mathbf{z})$ with output values in range $[0,1]$.
- Bernoulli variables $\mathbf{Y} = (Y_1, Y_2, \cdots, Y_K)$, where K is the dimension of $\mathbf{r}$.

The F and R functions can be implemented by neural networks. The parameters of an NBR model are the parameters of functions F and R, and $b_i = P_{Y_i}(1)$ for $i = 1, 2, \cdots, K$.

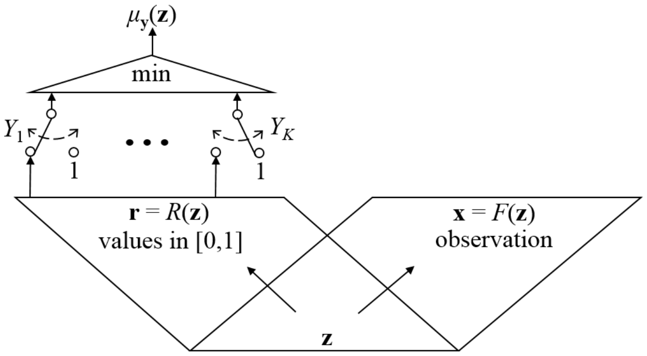
$\mu_{\mathbf{y}}(\mathbf{z})$
min
Y_1
1
Y_K
1
$\mathbf{r} = R(\mathbf{z})$
values in [0,1]
$\mathbf{x} = F(\mathbf{z})$
observation
$\mathbf{z}$

Figure 1: Architecture of NBR.

Each $R_i(\mathbf{z})$ is interpreted as the membership function of a fuzzy set over $\mathbf{z}$ space [Zadeh, 1965]. I consider the same 2^K possible worlds as in Section 2.1, but the intersection set $S_{\mathbf{y}}$ for each world now becomes the intersection of fuzzy sets and has the following membership function:

$$\mu_{\mathbf{y}}(\mathbf{z}) = \min_{i=1}^{K} (y_i \cdot R_i(\mathbf{z}) + 1 - y_i) \tag{3}$$

Consequently, the satisfiability of each world is no longer a binary property, but a degree in $[0,1]$: $\max_{\mathbf{z}} \mu_{\mathbf{y}}(\mathbf{z})$. Therefore, the mass assigned to each world should be proportional to $p(\mathbf{y}) \cdot \max_{\mathbf{z}} \mu_{\mathbf{y}}(\mathbf{z})$. I still need to ensure that the total mass is one, and that leads to the following formula which replaces (1) as the new basic probability assignment:

$$m\left(S'_{\mathbf{y}}\right) \triangleq \frac{p(\mathbf{y})\cdot\max_{\mathbf{z}}\mu_{\mathbf{y}}(\mathbf{z})}{\sum_{\mathbf{y}'}\left(p(\mathbf{y}')\cdot\max_{\mathbf{z}}\mu_{\mathbf{y}'}(\mathbf{z})\right)} \tag{4}$$

where $S'_{\mathbf{y}}$ is a fuzzy set with the membership function of $\mu_{\mathbf{y}}(\mathbf{z}) / \max_{\mathbf{z}'} \mu_{\mathbf{y}}(\mathbf{z}')$: $S'_{\mathbf{y}}$ is scaled $S_{\mathbf{y}}$ such that at least one point in the $\mathbf{z}$ space is fully included. Considering that $p(\cdot)$ is exactly the probability function of the Bernoulli vector $\mathbf{Y}$, the above has a more concise form:

$$m\left(S'_{\mathbf{y}}\right) \triangleq \frac{p(\mathbf{y})\cdot\max_{\mathbf{z}}\mu_{\mathbf{y}}(\mathbf{z})}{\mathrm{E}\left[\max_{\mathbf{z}}\mu_{\mathbf{Y}}(\mathbf{z})\right]} \tag{5}$$

With this new $m(\cdot)$ function, a belief function is specified over the $\mathbf{z}$ space: for any fuzzy set A,

$$\mathrm{Bel}(A) \triangleq \sum_{\mathbf{y}} m\left(S'_{\mathbf{y}}\right)\cdot\left(1-\max_{\mathbf{z}}\mu_{S'_{\mathbf{y}}\cap\overline{A}}(\mathbf{z})\right) \tag{6}$$

where $\mu_{S'_{\mathbf{y}}\cap\overline{A}}(\mathbf{z}) \triangleq \min\left(\mu_{\mathbf{y}}(\mathbf{z}) / \max_{\mathbf{z}'}\mu_{\mathbf{y}}(\mathbf{z}'), 1-\mu_A(\mathbf{z})\right)$ and where $\mu_A(\mathbf{z})$ is the membership function of A.

2.3 Query Answering

The general form of a query is a conditional belief function given a condition function $C(\mathbf{z})$ which outputs a scalar in range $[0,1]$. The condition may come as a function of $\mathbf{x}$, and since $\mathbf{x}$ is a deterministic function of $\mathbf{z}$, $C(\mathbf{z})$ is the general form and is the membership function of a fuzzy set in $\mathbf{z}$ space.

To answer a query, I add $C(\mathbf{z})$ as an extra entry to $\mathbf{r}$, and add an extra entry of constant 1 to $\mathbf{Y}$. Intuitively, the NBR has an additional rule that always exists. After such additions, formulas (5)(6) specify a conditional belief function.

Let us consider a special type of queries that are the most common in practice: given a Boolean function $C(\mathbf{z})$ as condition, compute the conditional belief and plausibility of another Boolean function $Q(\mathbf{z})$. In other words, I look for the replacement of (2). The following formulas can be derived from (5)(6) and the proof is omitted due to space limit.

$$\mathrm{Bel}\left(Q(\cdot) \mid C(\cdot)\right) = 1 - \frac{\mathrm{E}\left[\max\limits_{\mathbf{z}|C(\mathbf{z})=1, Q(\mathbf{z})=0} \mu_{\mathbf{Y}}(\mathbf{z})\right]}{\mathrm{E}\left[\max\limits_{\mathbf{z}|C(\mathbf{z})=1} \mu_{\mathbf{Y}}(\mathbf{z})\right]} \quad (7)$$

$$\mathrm{Pl}\left(Q(\cdot) \mid C(\cdot)\right) = \frac{\mathrm{E}\left[\max\limits_{\mathbf{z}|C(\mathbf{z})=1, Q(\mathbf{z})=1} \mu_{\mathbf{Y}}(\mathbf{z})\right]}{\mathrm{E}\left[\max\limits_{\mathbf{z}|C(\mathbf{z})=1} \mu_{\mathbf{Y}}(\mathbf{z})\right]} \quad (8)$$

2.4 Sample Generation

A belief function allows sample generation only if it is also a probability function. When combining two belief functions where one of the two is a probability function, by Dempster's rule of combination [Shafer, 1976], the resulting belief function is always a probability function. Therefore, to generate observation samples like traditional generative models, I combine the belief function of an NBR with a probability function over the $\mathbf{x}$ space. This probability function is referred to as the *prior-knowledge distribution*. Intuitively, the prior-knowledge distribution represents assumptions or knowledge that are not included in this NBR. For example, if one only knows the range of $\mathbf{x}$, a uniform prior-knowledge distribution can be used; if one knows the mean and variance of $\mathbf{x}$, a Gaussian distribution can be used. As will become evident later, I only require the ability to draw samples from it. The flexibility to combine an NBR with various prior-knowledge distributions is analogous to applying the same knowledge in multiple environments.

For clarity of presentation, let us focus on the scenario where the $\mathbf{x}$ space is discrete. For a sample value $\tilde{\mathbf{x}}$, let $P_0(\tilde{\mathbf{x}})$ denote its probability in the prior-knowledge distributions. Its probability after combining with an NBR is

$$P(\mathbf{x} = \tilde{\mathbf{x}}) = \frac{P_0(\tilde{\mathbf{x}}) \cdot \mathrm{Pl}(\mathbf{x} = \tilde{\mathbf{x}})}{\sum_{\mathbf{x}'} \left(P_0(\mathbf{x}') \cdot \mathrm{Pl}(\mathbf{x} = \mathbf{x}')\right)} \quad (9)$$

where the plausibilities are given by (8) with C being always true. To generate samples according to (9), I draw samples from the prior-knowledge distribution and randomly keep or discard a sample, such that the probability to keep sample $\tilde{\mathbf{x}}$ is proportional to $\mathrm{Pl}(\mathbf{x} = \tilde{\mathbf{x}})$. Note that the denominator in (8) does not change with Q and hence is the same for all $\tilde{\mathbf{x}}$ values; therefore I can simply use the numerator. In summary, the probability to keep a sample $\tilde{\mathbf{x}}$ is:

$$P_{\mathrm{keep}}(\tilde{\mathbf{x}}) = \mathrm{E}\left[\max_{\mathbf{z}|F(\mathbf{z})=\tilde{\mathbf{x}}} \mu_{\mathbf{Y}}(\mathbf{z})\right]. \quad (10)$$

When the $\mathbf{x}$ space is continuous, the generation procedure is the same: draw samples from a continuous prior-knowledge distribution and randomly keep or discard a sample according to the keep probability of (10).

2.5 Training

For unsupervised learning, an NBR is trained by maximizing the likelihood of observations in the sample generation process of the previous section. Therefore one needs to choose a prior-knowledge distribution for training. This choice defines what should be learned by the NBR: new knowledge that is present in the observations yet that is not already encoded in the prior-knowledge distribution. For example, samples generated by an existing NBR can be used as the prior-knowledge distribution to train a new NBR, and then this new NBR would be trained to learn and only learn new knowledge that is not in that existing NBR. Note that, after an NBR is trained, the query-answering process of Section 2.3 is independent of the prior-knowledge distribution used in training. In other words, an NBR's answers are based on only the knowledge contained in itself, and this enables modular and transferable knowledge representation.

Given observations $\mathbf{x_1}, \cdots, \mathbf{x_n}$, the likelihood loss is

$$\begin{aligned}\mathcal{L} &= -\frac{1}{n}\sum_{i=1}^{n} \log P(\mathbf{x} = \mathbf{x_i}) \\ &= -\frac{1}{n}\sum_{i=1}^{n} \log \frac{P_0(\mathbf{x_i}) \cdot P_{\mathrm{keep}}(\mathbf{x_i})}{\sum_{\mathbf{x}'}\left(P_0(\mathbf{x}') \cdot P_{\mathrm{keep}}(\mathbf{x}')\right)} \\ &= -\frac{1}{n}\sum_{i=1}^{n} \log P_0(\mathbf{x_i}) - \frac{1}{n}\sum_{i=1}^{n} \log P_{\mathrm{keep}}(\mathbf{x_i}) \\ &\quad + \log \sum_{\mathbf{x}'}\left(P_0(\mathbf{x}') \cdot P_{\mathrm{keep}}(\mathbf{x}')\right)\end{aligned} \quad (11)$$

The first term is a constant and can be removed; the last term can be shortened by letting $\mathbf{X}$ denote a random vector that has the prior-knowledge distribution. The loss function becomes:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\mathrm{keep}}(\mathbf{x_i}) + \log \mathrm{E}\left[P_{\mathrm{keep}}(\mathbf{X})\right] \quad (12)$$

Each training iteration uses a batch of observations to approximate the first term and a batch of samples from the prior-knowledge distribution to approximate the second term. One implementation issue is that the expectation is before log in the second term and small batch size causes bias in gradient estimation. In practice, I use the following loss instead:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\mathrm{keep}}(\mathbf{x_i}) + \mathrm{E}\left[P_{\mathrm{keep}}(\mathbf{X})\right]/\alpha \quad (13)$$

where α is a constant that gets updated once every certain number of batches, and its value is an estimate of $\mathrm{E}\left[P_{\mathrm{keep}}(\mathbf{X})\right]$ on a large number of samples. It is straightforward to verify that, with a large batch size, (12) and (13) result in asymptotically the same gradients with respect to model parameters. (13) is linear with respect to the expectation in the second term and hence small batch size can be used without causing bias in gradient estimation.

3 Unsupervised Learning: a Synthetic Task

This section gives a first demonstration of NBR on an unsupervised-learning task. Source code for training and in-

C	Q	belief	plausibility
$x_0 = 1$	x_{10}	0	0.33
$x_0 = 0$	x_{10}	0.67	1
$x_0 = 1, x_{1-4} = 0$	x_5	0.89	1
$x_0 = 1, x_{1-4} = 0, x_{10} = 1$	x_5	0.67	1
$x_0 = 1, x_{1-4} = 0, x_{6-10} = 1$	x_5	0.67	0.75

Table 1: Answers to example queries by NBR.

ference is available at
http://researcher.watson.ibm.com/group/10228

Consider a world with 11 bits. Partial observations are available that observe either the first 10 bits or the last 10 bits. In observations of the first 10 bits, the first bit is the majority function of the middle 9 bits for 90% of the cases, and the inverse for 10% of the cases. In observations of the last 10 bits, the last bit is the majority function of the middle 9 bits for 20% of the cases, and the inverse for 80%.

An NBR is trained on these observations: K is 2; $F(\cdot)$ is identity function; $R(\cdot)$ is two three-layer ReLU networks, where each network is followed by a sigmoid unit, and where one network takes the first 10 bits as input and the other takes the last 10 bits. The loss (13) is used, and the prior-knowledge distribution is uniform over the 2^{11} possibilities. For a partial observation $\mathbf{x_i}$, let $\mathbf{x_{i,0}}$ and $\mathbf{x_{i,1}}$ be the two possible full observations. Substituting $P(\mathbf{x} = \mathbf{x_i}) = P(\mathbf{x} = \mathbf{x_{i,0}}) + P(\mathbf{x} = \mathbf{x_{i,1}})$ into (11) and following the same derivation to (13), it is straightforward to see that I simply need to compute $P_{\text{keep}}(\mathbf{x_i})$ in (13) as $P_{\text{keep}}(\mathbf{x_{i,0}}) + P_{\text{keep}}(\mathbf{x_{i,1}})$.

Table 1 lists NBR's answers to five queries. The belief values are computed by (7) and the plausibility values are by (8). In the first query, the condition is the first bit being 1 and the query is on the last bit. NBR answers with belief zero and plausibility 0.33, which means that it has some evidence to support x_{10} being 0 but no evidence to support x_{10} being 1. This answer is intuitive: with $x_0 = 1$, it is likely that the majority of the middle 9 bits is 1, and consequently it is likely that x_{10} is 0. Recall that during training the NBR has never seen an observation that simultaneously shows x_0 and x_{10}, and it answers the query by performing multi-hop reasoning with uncertainty. The second query is the opposite: with $x_0 = 0$, NBR has some evidence to support x_{10} being 1 but no evidence to support x_{10} being 0. There is an interesting comparison between the third and fourth queries: NBR's belief decreases when $x_{10} = 1$ is added to the condition. The reason is that $x_0 = 1$ and $x_{10} = 1$ are two conflicting pieces of information, and as a result the denominator in (7) is reduced to less than 1 for the fourth query, which in turn causes the belief value to decrease. This is consistent with human intuition when facing conflicting information. The fifth query is a similar case of conflicting information where all bits but x_5 are fixed. NBR's answer means that it has evidence to support both possibilities for x_5 and that the evidence for x_5 being 1 is stronger than the evidence for 0. This is again consistent with human intuition based on observations of this world. It's worth noting that the gap between belief and plausibility narrows for the fifth query, which reflects more information about the world from the condition, however the gap still exists, which reflects epistemic uncertainty, – the fact that NBR does not have complete knowledge about the world. The gap only closes when an NBR has complete knowledge regarding a query, in which case the answer is a probability function.

To demonstrate the separation between latent space and observation space, a second NBR model is trained with nontrivial $F(\cdot)$. First an autoencoder is trained with a latent space of dimension 8. I then use the decoder part as $F(\cdot)$ and fix it as non-trainable during NBR training. When I need to evaluate the max operation in (10), I feed $\tilde{\mathbf{x}}$ into the encoder part to compute $\mathbf{z}$ as a cheap surrogate operation. The resulting NBR gives the same answers as in Table 1.

4 Supervised Learning: a Robust Classifier

This section demonstrates NBR for supervised learning, and specifically discusses a robust MNIST classifier for 4 and 9.

4.1 Using NBR for Classification

It is possible to convert a classification task to unsupervised learning by treating labels as part of the observation. However, that is not the most efficient way to use NBR for classification, and this section presents a better approach. Most discussions are applicable to classification in general.

Given an MNIST image, consider a world with 10 possibilities, – one of the ten labels is true. Recall from Section 2.2 that the role of each entry in $\mathbf{r}$ is to specify a fuzzy set. In a world with 10 possibilities, a fuzzy set is defined by 10 grades of membership, i.e., 10 numbers between 0 and 1. Therefore, each entry in $\mathbf{r}$ can be implemented by an arbitrary MNIST classifier, and I simply add 10 sigmoid units at the end to convert logits to values in $[0, 1]$. Function $F(\cdot)$ is not used. Therefore, an NBR classifier is composed of K classifiers with sigmoids added and Bernoulli variables $\mathbf{Y}$.

Now let's define its outputs. With (7)(8), the belief and plausibility of each label can be computed; let them be Bel_j, Pl_j, $0 \le j \le 9$. The output vector in the implementation is:

$$\mathbf{o} = (\log \text{Pl}_0, \cdots, \log \text{Pl}_9) \tag{14}$$

and its argmax is NBR's output label. The values in (14) are the negative of weights of evidence against each label [Shafer, 1976]. There are other choices: for example, $\log \text{Pl}_j - \log(1 - \text{Bel}_j)$ is another reasonable choice which combines weights of evidence for and against label j.

4.2 Distinct Bodies of Evidence

The intuition behind robust classification is to divide a classification task into simpler tasks, each of which is so simple that it can be solved robustly by a single $R_i(\cdot)$, and then NBR combines the K sources to solve the overall task. Dempster's rule requires that sources represent entirely distinct bodies of evidence [Shafer, 1976]. I achieve this by using different frames of discernment and dividing the training data.

To explain by example, consider a rule with this frame of discernment: {0} {5,6} {7}. It is built as a classifier with 3 classes and specifies a fuzzy set with these grades of membership: $v_1, 1, 1, 1, 1, v_2, v_2, v_3, 1, 1$. Note that the grades for {1,2,3,4,8,9} are always 1; intuitively this rule makes no judgment about labels outside its frame of discernment. Also note that the grades for 5 and 6 are always the same; intuitively this rule does not distinguish between them. With different frames of discernment, an NBR can gather enough knowledge for the overall task.

However, some minimal frame of discernment, for example $\{4\}$ $\{9\}$, is still too complex to solve robustly with a single neural network. I will focus on this pair and will build an NBR with 2632 rules to classify 4 and 9. The first 14 rules are neural networks that are trained on different subsets of the training data, while the rest are memorization rules.

Let T_4 be the set of training images with label 4, and let T_9 be that for 9. Let $T_{4,i}$, $1 \le i \le 8$ be mutually exclusive and collectively exhaustive subsets of T_4, and let $T_{9,i}$, $1 \le i \le 8$ be such subsets of T_9. For $1 \le i \le 7$, the i^{th} rule is a binary classifier that is trained to distinguish $T_{4,i}$ versus T_9. For $8 \le i \le 14$, the i^{th} rule is a binary classifier that is trained to distinguish $T_{9,i-7}$ versus T_4.

All rules have the form of $\text{sigmoid}\left(s_i \cdot G_i\left(\text{image}\right)\right), 1 \le i \le 2632$, where s_i is a trainable scalar and $G_i\left(\cdot\right)$ is a trainable function that outputs a scalar. For the first 14 rules, $G_i\left(\cdot\right)$ is a L_2-nonexpansive neural network (L2NNN) [Qian and Wegman, 2019]. Each $T_{4,1\le i\le 7}$ is $\{\mathbf{t} \in T_4 | G_i\left(\mathbf{t}\right) \ge \gamma \,\text{and}\, G_i\left(\mathbf{t}\right) \ge G_j\left(\mathbf{t}\right), \forall 1 \le j \le 7\}$, where γ is a hyperparameter. Intuitively, each digit 4 is assigned to the $G_i\left(\cdot\right)$ that classifies it the most robustly, and it is assigned to $T_{4,8}$ if none of the seven $G_i\left(\cdot\right)$'s reaches threshold γ. The $T_{9,i}$ subsets are similarly defined. All subsets are periodically updated during the training process.

After the first 14 rules are trained, $T_{4,8}$ contains 653 images and $T_{9,8}$ contains 1965. These images are handled by memorization rules[1]: 653 rules to distinguish each image in $T_{4,8}$ versus T_9, and another 1965 rules for $T_{9,8}$ against T_4. For these rules, $G_i\left(\mathbf{x}\right) \triangleq d_i - \|\mathbf{x} - \mathbf{t_i}\|_2, 15 \le i \le 2632$, where $\mathbf{t_i}$ is the image to memorize and d_i is a trainable scalar; note that this function is nonexpansive with respect to L_2.

A final adjustment is needed to produce $R_i\left(\cdot\right)$. Let T_i' and T_i'' be the two sets of training images that the i^{th} rule is trained to distinguish. For example, T_i' may be one of the $T_{4,i}$'s while T_i'' may be T_9. If the sigmoid outputs 0 for image $\mathbf{t}$, the knowledge obtained is that $\mathbf{t}$ is dissimilar to those specific 4's in T_i', not all digits 4; hence this rule should not put the plausibility of label 4 to zero. Therefore I have $R_i\left(\mathbf{t}\right) \triangleq \text{sigmoid}\left(s_i \cdot G_i\left(\mathbf{t}\right)\right)$ if $G_i\left(\mathbf{t}\right) \ge 0$, and otherwise $R_i\left(\mathbf{t}\right) \triangleq 0.5 - \frac{|T_i'|}{|T_4|} \cdot \left(0.5 - \text{sigmoid}\left(s_i \cdot G_i\left(\mathbf{t}\right)\right)\right)$.

4.3 Scaling Trick for Linear Complexity

Let us consider a single rule and let $\mathbf{v} \triangleq \left(v_1, \cdots, v_J\right)$ be the grades of membership that this rule computes for a particular image over its frame of discernment. Let $v_{\max}$ and $v_{\min}$ denote the max and min grades among them. Let b denote the corresponding b_i parameter. If I pretend that this is a single-rule NBR and apply (5), I effectively replace $\mathbf{v}$ with $\mathbf{v}' \triangleq \left(v_1/v_{\max}, \cdots, v_J/v_{\max}\right)$ and replace b with $b' \triangleq b \cdot v_{\max} / \left(1 - b + b \cdot v_{\max}\right)$. Note that the max grade in $\mathbf{v}'$ is always 1. These replacements do not modify the single-rule NBR at all: (6) for any set A computes the same value before and after the replacements.

Let us push one step further and also scale the min grade to zero. Specifically, I replace $\mathbf{v}$ with $\mathbf{v}'' \triangleq \left(\frac{v_1 - v_{\min}}{v_{\max} - v_{\min}}, \cdots, \frac{v_J - v_{\min}}{v_{\max} - v_{\min}}\right)$ and replace b with $b'' \triangleq b \cdot \left(v_{\max} - v_{\min}\right) / \left(1 - b + b \cdot v_{\max}\right)$. These replacements subtly modify the single-rule NBR: (6) is not affected for any classical set A, however there is no guarantee if A is a fuzzy set. It is arguable that such changes are acceptable.

The benefit of using $\mathbf{v}''$ and b'' is that, if $J = 2$, i.e., if the frame of discernment is binary, this rule becomes a classical rule. If all rules in an NBR are classical, (7)(8) become the same as (2) and can be evaluated by the original Dempster's rule. Since Dempster's rule is associative [Shafer, 1976], the worst-case complexity is $O\left(K \cdot 2^L\right)$ where K is the number of rules and L is the number of classes. Consequently NBR can use a large K. If L is large, a classification task can be done hierarchically. Note that using only binary frames of discernments is not a strong restriction: for example, $\{1,4,7,9\}$ $\{2,3,5,8\}$ is a binary frame and is expressive.

4.4 Loss Functions for Robustness

The training process has three steps. In the first step, $G_i\left(\cdot\right)$ functions in the first 14 rules are learned. The first seven $G_i\left(\cdot\right)$'s are trained jointly with the following loss function:

$$\begin{aligned} \mathcal{L}_{1-7} = &-\sum_{i=1}^{7} \sum_{\mathbf{t} \in T_{4,i}} \log\left(\text{sigmoid}\left(s \cdot \left(G_i\left(\mathbf{t}\right) - \beta\right)\right)\right) \\ &- \sum_{\mathbf{t} \in T_9} \log\left(1 - \text{sigmoid}\left(s \cdot \left(\max_{i=1}^{7} G_i\left(\mathbf{t}\right) + \beta\right)\right)\right) \end{aligned} \tag{15}$$

where s and β are hyperparameters. With the definition of $T_{4,1\le i\le 7}$ from Section 4.2, (15) can be viewed as the cross-entropy loss of the classifier of $\text{sigmoid}\left(s \cdot \max_{i=1}^{7} G_i\left(\mathbf{t}\right)\right)$ with a twist: I reduce $G_i\left(\cdot\right)$'s by β for digits 4 and increase them by β for digits 9. Intuitively, because $G_i\left(\cdot\right)$'s are L2NNNs, these adjustments realize the worst-case scenario of an adversarial attack of L_2 distortion of β. This technique is computationally much cheaper than adversarial training, and I refer to it as *poor man's adversarial training*. Functions $G_i\left(\cdot\right)$ for $8 \le i \le 14$ are trained with a similar loss.

In the second step, $s_i, 1 \le i \le 2632$ and $d_i, 15 \le i \le 2632$ are learned. The loss function for the i^{th} rule is:

$$\begin{aligned} \mathcal{L}_i = &-\sum_{\mathbf{t} \in T_i'} \log\left(\text{sigmoid}\left(s_i \cdot \left(G_i\left(\mathbf{t}\right) - \beta\right)\right)\right) \\ &- \sum_{\mathbf{t} \in T_i''} \log\left(1 - \text{sigmoid}\left(s_i \cdot \left(G_i\left(\mathbf{t}\right) + \beta\right)\right)\right) \end{aligned} \tag{16}$$

where again T_i' and T_i'' are the two sets of training images that this rule distinguishes. (16) is also the cross-entropy loss with poor man's adversarial training.

In the third step, parameters $b_i, 1 \le i \le 2632$ are learned jointly. I again use poor man's adversarial training but, instead of a fixed β, I apply an image-dependent amount of

[1]Effects of the memorization rules: if they are removed, the nominal accuracy of the NBR classifier drops from 99.1% to 98.0%, while its robust accuracy increases from 55.3% to 59.1%.

	natural	PGD	BA	CW	SCW	robust
Vanilla	99.7%	48.8%	0%	0%	0%	0%
Madry *et al.*	98.6%	97.8%	10.6%	47.8%	17.6%	1.3%
Wong&Kolter	98.9%	97.7%	15.9%	60.4%	28.3%	12.0%
L2NNN	99.1%	94.4%	42.7%	41.3%	41.3%	41.3%
NBR	99.1%	92.5%	57.2%	55.5%	55.3%	55.3%

Table 2: Accuracies on natural and adversarial test images of 4 and 9 where the L_2-norm limit of distortion is 2. Accuracies in the last column are under the best of four attacks for each image.

adversarial adjustment on $G_i(\cdot)$'s. Let $\beta_{\mathbf{t}}$ denote the adjustment amount for training image $\mathbf{t}$. Let $\mathbf{o}_{\text{ori}}(\mathbf{t})$ denote (14) without adjustment and let $\mathbf{o}_{\text{adv}}(\mathbf{t})$ denote that with adjustments of $\beta_{\mathbf{t}}$. The $\beta_{\mathbf{t}}$ value is chosen such that $\mathbf{o}_{\text{adv}}(\mathbf{t})$ barely classifies correctly, and it is periodically updated during the training process. The loss function of the third step is:

$$\begin{aligned}\mathcal{L} =&\text{avg}_{\mathbf{t}}\left(\text{softmax-cross-entropy}\left(\mathbf{o}_{\text{adv}}(\mathbf{t}), \text{label}_{\mathbf{t}}\right)\right)\\ &+\omega\cdot\text{avg}_{\mathbf{t}}\left(\text{softmax-cross-entropy}\left(\mathbf{o}_{\text{ori}}(\mathbf{t}), \text{label}_{\mathbf{t}}\right)\right)\end{aligned} \tag{17}$$

where ω is a hyperparameter. Intuitively, the image-dependent adversarial adjustments cause a uniform push to classify all images more robustly. It's worth noting that, applying (9), it can be shown that the softmax cross entropy of $\mathbf{o}(\mathbf{t})$ is equal to the log likelihood of generating the label of $\mathbf{t}$ if assuming a uniform prior-knowledge distribution.

4.5 Results

The pre-trained NBR MNIST 4-9 classifier is available at http://researcher.watson.ibm.com/group/10228

Table 2 compares the NBR MNIST classifier against those in [Madry *et al.*, 2018; Wong and Kolter, 2018; Qian and Wegman, 2019], which are publicly available. I simply use their two logits for 4 and 9 to form binary classifiers. Among them, the L2NNN classifier from [Qian and Wegman, 2019] is the state of the art in robustness as measured by L_2 metric.

To choose a meaningful L_2 ε, I measure $d(\mathbf{t})$, which is the distance from $\mathbf{t}$ to the nearest training image with a different label, and the *oracle robustness* for $\mathbf{t}$ is L_2 radius $d(\mathbf{t})/2$. Over the MNIST training set, the oracle robustness radius is above 3 for 51% of images, above 2.5 for 79%, and above 2 for 96%. Consequently $\varepsilon = 2$ is the meaningful L_2 threshold when quantifying robustness of MNIST classifiers.

Robustness is measured by running four attacks: projected gradient descent (PGD) [Madry *et al.*, 2018], boundary attack (BA) [Brendel *et al.*, 2018], Carlini & Wagner (CW) attack [Carlini and Wagner, 2017b] and seeded CW (SCW). Foolbox [Rauber *et al.*, 2017] is used for PGD and BA; CW is original code from [Carlini and Wagner, 2017b]; SCW is a CW search with a starting point that is provided by a transfer attack, and is a straightforward variation of the CW code. Iteration limit is 100 for PGD, 50K for BA, and 10K for CW and SCW. A classifier is considered robust on an image if it remains correct under all four attacks. Table 2 shows that the NBR classifier has the best robustness, and also the best natural accuracy among all but the non-robust vanilla model.

Table 3 presents empirical comparisons between NBR and other ensemble methods, by replacing belief-function arithmetic with Markov random fields (MRF) and Gaussian naive Bayes (GNB). The difference between MRF #1 and MRF #2 is that the former uses $-\text{sigmoid}(s_i\cdot G_i(\cdot))$ as energy functions while the latter uses $-\log(\text{sigmoid}(s_i\cdot G_i(\cdot)))$; parameters s_i are re-trained together with MRF's weight parameters; the same poor man's adversarial training of (17) is applied in training MRF models. The difference between GNB #1 and GNB #2 is that the former uses $\text{sigmoid}(s_i\cdot G_i(\cdot))$ as features while the latter uses $G_i(\cdot)$.

	natural	robust
NBR	99.1%	55.3%
Markov random field #1	97.6%	52.0%
Markov random field #2	90.0%	47.8%
Gaussian naive Bayes #1	98.6%	53.9%
Gaussian naive Bayes #2	69.4%	33.7%

Table 3: Accuracies of ablation-study models.

5 Related Work

As a formalism for reasoning with uncertainty, belief functions [Shafer, 1976] have two distinct advantages: explicit modeling of the lack of knowledge and an elegant mechanism of combining multiple sources of information. Reasoning frameworks based on belief functions have been proposed [Gordon and Shortliffe, 1985; Baldwin, 1986; Lowrance *et al.*, 1986; Laskey and Lehner, 1988; D'Ambrosio, 1988; Wan and Kifer, 2009], and they have varying degrees of similarity to the restricted framework of Section 2.1. There are another set of works that combine belief functions and fuzzy sets [Zadeh, 1979; Yen, 1990; Denœux, 2000], and the commonality between them and NBR is that my equation (6), the mapping from a basic probability assignment to a belief function, coincides with that in [Zadeh, 1979].

These early works share some common weaknesses: where do rules come from, and where do uncertainty quantifications on the rules come from. Relying on manual inputs is clearly not scalable. To be fair, mainstream methods based on Bayesian networks [Pearl, 1988] or Markov random fields [Kindermann and Snell, 1980] often have the same weaknesses. Some have addressed the second weakness: for example, Markov logic networks [Richardson and Domingos, 2006] learn the weights on clauses. Attempts to address the first weakness, e.g., by inductive logic programming [Muggleton and De Raedt, 1994; De Raedt and Kersting, 2008], are often limited.

Progresses in addressing the first weakness via automatic discovery of rules have emerged from the machine learning field. In [Hinton, 2002; Salakhutdinov and Hinton, 2009], Markov random fields, which can be viewed as compositions of non-symbolic rules, are learned from data. In [França *et al.*, 2014; Evans and Grefenstette, 2018], symbolic logic programs are learned from examples. Another example is [Serafini and Garcez, 2016] which defines a formalism of real-valued logic and thereby enables learning non-symbolic rules. The training algorithm of NBR represents a new approach on this front.

Adversarial robustness is a well-known difficult problem [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015; Carlini and

Wagner, 2017b], and many remedies have been tried and failed [Carlini and Wagner, 2017a; Athalye *et al.*, 2018]. For MNIST, if distortion is measured by the L_∞ distance, there are a number of approaches [Wong and Kolter, 2018; Raghunathan *et al.*, 2018; Schott *et al.*, 2019] and in particular [Madry *et al.*, 2018] achieves good L_∞ robustness by adversarial training. For L_2 robustness which is less understood and perhaps more difficult, before this work the state of the art is an L2NNN from [Qian and Wegman, 2019] with adversarial training. It's worth noting that adversarial training alone does not work well for L_2 robustness [Schott *et al.*, 2019; Qian and Wegman, 2019; Tsipras *et al.*, 2019]. My hypothesis is that reasoning is a missing piece in previous works, and the NBR classifier is a demonstration that it's possible to break a classification task into simpler and smaller tasks, each of which is robustly solvable by an L2NNN, and that NBR can reason about the resulting many sources of evidence to reach a robust conclusion.

6 Conclusions and Future Work

This paper presents neural belief reasoner, which is a new generative model and a new approach to combine learning and reasoning. Its properties are studied through two tasks: an unsupervised-learning task of reasoning with uncertainty, and a supervised-learning task of robust classification. In the latter task, the MNIST classifier for 4 and 9 sets a new state of the art in L_2 robustness while maintaining over 99% nominal accuracy.

An important future direction is improving the scalability of unsupervised learning. For example, the first task uses an NBR with $F(\cdot)$ being identity function, and the complexity of exact calculus in unsupervised learning is exponential with respect to the number of rules. To unlock its full potential, innovations would be needed for efficient inference and training algorithms, including but not limited to Monte Carlo methods, as well as efficient constraint-programming solvers. There are also open questions from the application perspective, e.g., how to take advantage of NBR's sample-generation capability and how to leverage NBR for interpretability.

References

[Athalye *et al.*, 2018] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.

[Baldwin, 1986] James F. Baldwin. Support logic programming. In *Fuzzy sets theory and applications*, pages 133–170. Springer, 1986.

[Brendel *et al.*, 2018] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[Carlini and Wagner, 2017a] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[Carlini and Wagner, 2017b] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[D'Ambrosio, 1988] Bruce D'Ambrosio. A hybrid approach to reasoning under uncertainty. *International Journal of Approximate Reasoning*, 2(1):29–45, 1988.

[De Raedt and Kersting, 2008] Luc De Raedt and Kristian Kersting. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming: Theory and Applications*, pages 1–27. Springer, 2008.

[Denœux, 2000] Thierry Denœux. Modeling vague beliefs using fuzzy-valued belief structures. *Fuzzy Sets and Systems*, 116(2):167–199, 2000.

[Evans and Grefenstette, 2018] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

[França *et al.*, 2014] Manoel VM França, Gerson Zaverucha, and Artur d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104, 2014.

[Goodfellow *et al.*, 2015] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[Gordon and Shortliffe, 1985] Jean Gordon and Edward H. Shortliffe. A method for managing evidential reasoning in a hierarchical hypothesis space. *Artificial Intelligence*, 26(3):323–357, 1985.

[Hinton, 2002] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[Kindermann and Snell, 1980] Ross Kindermann and J. Laurie Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

[Laskey and Lehner, 1988] Kathryn B. Laskey and Paul E. Lehner. Belief maintenance: An integrated approach to uncertainty management. In *AAAI Conference on Artificial Intelligence*, pages 210–214, 1988.

[Lowrance *et al.*, 1986] John D. Lowrance, Thomas D. Garvey, and Thomas M. Strat. A framework for evidential-reasoning systems. In *AAAI Conference on Artificial Intelligence*, pages 896–901, 1986.

[Madry *et al.*, 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[Muggleton and De Raedt, 1994] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Qian and Wegman, 2019] Haifeng Qian and Mark N. Wegman. L2-nonexpansive neural networks. In *International Conference on Learning Representations*, 2019.

[Raghunathan *et al.*, 2018] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

[Rauber *et al.*, 2017] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[Salakhutdinov and Hinton, 2009] Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[Schott *et al.*, 2019] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.

[Serafini and Garcez, 2016] Luciano Serafini and Artur d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

[Shafer, 1976] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[Tsipras *et al.*, 2019] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[Wan and Kifer, 2009] Hui Wan and Michael Kifer. Belief logic programming: uncertainty reasoning with correlation of evidence. In *Logic Programming and Nonmonotonic Reasoning, Lecture Notes in Computer Science*, pages 316–328, 2009.

[Wong and Kolter, 2018] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

[Yen, 1990] John Yen. Generalizing the Dempster-Shafer theory to fuzzy sets. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(3):559–570, 1990.

[Zadeh, 1965] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[Zadeh, 1979] Lotfi A. Zadeh. Fuzzy sets and information granularity. *Advances in Fuzzy Set Theory and Applications*, 11:3–18, 1979.