Abstract

Probabilistic inference is a key computational challenge in statistical machine learning and artificial intelligence. The ability to represent complex high dimensional probability distributions in a compact form is the most important insight in the field of graphical models.

In this paper, we explore a novel way to exploit compact representations of high-dimensional probability distributions in approximate probabilistic inference algorithms. Our approach is based on discrete Fourier Representation of weighted Boolean Functions, complementing the classical method to exploit conditional independence between the variables. We show that a large class of probabilistic graphical models have a compact Fourier representation. This theoretical result opens up an entirely new way of approximating a probability distribution. We demonstrate the significance of this approach by applying it to the variable elimination algorithm and comparing the results with the bucket representation and other approximate inference algorithms, obtaining very encouraging results.

1 Introduction

Probabilistic inference is a key computational challenge in statistical machine learning and artificial intelligence. Inference methods have an enormous number of applications, from learning models to making predictions and informing decision-making using statistical models of data. Unfortunately, the problem is computationally intractable, and standard exact inference algorithms, such as variable elimination and junction tree algorithms have worst-case exponential complexity.

The ability to represent complex high dimensional probability distributions in a compact form is the most important insight in the field of graphical models. The fundamental idea is to exploit (conditional) independencies between the variables to achieve compact factored representations, where a complex global model is represented as a product of simpler, local models. Similar ideas have been considered in the analysis of Boolean functions and logical forms [4], as well as in physics with low rank tensor decompositions and matrix product states representations [6, 12, 13, 22].

Compact representations are also key for the development of efficient inference algorithms, including message-passing ones. Efficient algorithms can be developed when messages representing the interaction among many variables can be decomposed or approximated with the product of several smaller messages, each involving a subset of the original variables. Numerous approximate and exact inference algorithms are based on this idea [4, 5, 6, 8, 23, 3, 11, 10].

Conditional independence (and related factorizations) is not the only type of structure that can be exploited to achieve compactness. For example, consider the regression tree in Figure 1. No two variables in the probability distribution in Figure 1 are independent with each other. The probability
distribution cannot be represented by the product of simpler terms and requires a full probability table with $2^7 = 128$ entries to be represented exactly. Nevertheless, this table can be described exactly by 8 simple decision rules, each corresponding to a path from the root to a leaf in the tree.

In this paper, we explore a novel way to exploit compact representations of high-dimensional probability tables in (approximate) probabilistic inference algorithms. Our approach is based on a (discrete) Fourier representation of the tables, which can be interpreted as a change of basis. Crucially, tables that are dense in the canonical basis can have a sparse Fourier representation. Under certain conditions, probability tables can be represented (or well approximated) using a small number of Fourier coefficients. The Fourier representation has found numerous recent applications in PAC learning [18, 15, 1, 2], but these ideas have not been fully exploited in the fields of probabilistic inference and graphical models.

In general, a general factor over $n$ Boolean variables requires $O(2^n)$ entries to be specified, and similarly the corresponding Fourier representation is dense in general, i.e., it has $O(2^n)$ non-zero coefficients. However, a rather surprising fact which was first discovered by Linial [14] is that factors corresponding to fairly general classes of logical forms admit a compact Fourier representation. Linial discovered that formulas in Conjunctive Normal Form (CNF) and Disjunctive Normal Form (DNF) with bounded width (the number of variables in each clause) have compact Fourier representations.

In this paper, we introduce a novel approach for using approximate Fourier representations in the field of probabilistic inference. We generalize the work of Linial to the probability distributions (the weighted case where the entries are not necessarily 0 or 1), showing that a large class of probabilistic graphical models have compact Fourier representation, if they are composed with factors with bounded domain size and discriminative weights. This includes interesting graphical models, such as Markov Logic Networks [21] with discriminative weights. The proof extends the famous Hastad’s Switching Lemma [9] to the weighted case. At a high level, a compact Fourier representation often means the weighted probabilistic distribution can be captured by a small set of critical decision rules. Hence, this notion is closely related to decision trees with bounded depth.

Sparse (low-degree) Fourier representations provide an entirely new way of approximating a probability distribution. We demonstrate the power of this idea by applying it to the variable elimination algorithm and comparing the result with bucket representation and various approximate inference algorithms on a set of benchmarks, and obtain very encouraging results.

2 Preliminaries

2.1 Inference In Graphical Models

We consider a Boolean graphical model of $N = |V|$ boolean variables, where $V = \{x_i|i \in V\}$. We use bold typed variables to represent a vector of variables. For example, the vector of all boolean variables $x$ is defined as $x = (x_1, x_2, \ldots, x_N)^T$. Moreover, we use $x_S$ to represent the image of vector $x$ projected onto a subset of variables: $x_S = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})^T$ where $x_{i_1}, \ldots, x_{i_k} \in S$. A

![Decision Tree Example](image)

Figure 1: An example of a decision tree representing a function $f : \{x_1, \ldots, x_7\} \rightarrow \mathbb{R}^+$. 
pressed as a multilinear polynomial, \( \phi(x, y) = \frac{1}{4}(f_1 + f_2 + f_3 + f_4) \)

\[
\begin{array}{c|c|c}
 x & y & \phi(x, y) \\
-1 & -1 & f_1 \\
-1 & 1 & f_2 \\
1 & -1 & f_3 \\
1 & 1 & f_4 \\
\end{array}
\]

Table 1: (Left) Function \( \phi : \{-1, 1\}^2 \rightarrow \mathbb{R} \) is represented in a contingency table. (Middle) \( \phi \) is re-written using interpolation. (Right) The terms of the equation in the middle is re-arranged, which yields the Fourier expansion of function \( \phi \).

A probabilistic graphical model is defined as:

\[
Pr(x) = \frac{1}{Z} f(x) = \frac{1}{Z} \prod_{i=1}^{K} \psi_i(x_{S_i}).
\]

where each \( \psi : \{-1, 1\}^{\mid S_i \mid} \rightarrow \mathbb{R}^+ \) is called a factor, and is a function that depends on a subset of variables in \( S_i \). \( Z = \sum_{x} \prod_{i=1}^{K} \psi_i(x_{S_i}) \) is the normalization factor, and is often called the partition function. In this paper, we will use \(-1\) and \(1\) to represent false and true for Boolean functions. In this paper we consider two key probabilistic inference tasks: the computation of the partition function \( Z \) (PR) and marginal probabilities \( Pr(e) = \frac{1}{Z} \sum_{x \in e} f(x) \) (Marginal).

Variable Elimination Algorithm is an exact algorithm to compute marginals and the partition function for general graphical models. It starts with a variable ordering \( \pi \). In each iteration, it eliminates one variable by multiplying all factors involving that variable, and then summing that variable out. When all variables are eliminated, the factor remaining is a singleton, whose value corresponds to the partition function. The complexity of the VE algorithm depends on the size of largest factors generated during the elimination process, and is known to be exponential in the tree-width [7].

Detcher proposed the Mini-bucket Elimination Algorithm [4], which dynamically decomposes and approximates factors (when the domain of a product exceeds a threshold) with the product of smaller factors during the elimination process.

### 2.2 Hadmard-Fourier Transformation

Hadamard-Fourier transformation has recently attracted a lot of attention in PAC Learning Theory. Table[1] provides an example where a function \( \phi(x, y) \) is transformed into its Fourier representation. The transformation works by writing \( \phi(x, y) \) using interpolation, then re-arranging the terms to get a canonical term. The example can be generalized, and it can be shown that any function defined on a Boolean hypercube can be equivalently represented in Fourier space:

**Definition 1.** (Hadamard-Fourier Transformation) Every \( f : \{-1, 1\}^n \rightarrow \mathbb{R} \) can be uniquely expressed as a multilinear polynomial,

\[
f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i.
\]

where each \( c_S \in \mathbb{R} \). Following standard notation, we will write \( \hat{f}(S) \) to denote the coefficient \( c_S \) and \( \psi_S(x) \) for the function \( \prod_{i \in S} x_i \), and call \( f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \psi_S(x) \) the Hadmard-Fourier expansion of \( f \). Note \( \chi_0 = 1 \). We also call \( \hat{f}(S) \) a k-degree coefficient of \( f \) iff \( |S| = k \).

We re-iterate some classical results on Fourier expansion. First, as with the classical (inverse) Fast Fourier Transformation (FFT) in the continuous domain, there are similar divide-and-conquer algorithms (FFT and \( \text{invFFT} \)) which connect the contingency table representation of \( f \) with its Fourier
representation. Both FFT and $\text{invFFT}$ run in time $O(n \cdot 2^n)$ for a function involving $n$ variables. In fact, the vector of all function values and the vector of Fourier coefficients are connected by a $2^n$-by-$2^n$ matrix $H_n$, which is often called the $n$-th Hadamard-Fourier matrix. In addition, we have the Parseval’s identity for Boolean Functions as well: $E_x[f(x)^2] = \sum_S \hat{f}(S)^2$.

## 3 Low Degree Concentration of Fourier Coefficients

Fourier expansion replaces the complete contingency table representation of a weighted function with its Fourier coefficients. For a function with $n$ Boolean variables, the complete contingency table requires $2^n$ entries, and so does the full Fourier expansion. Interestingly, many natural functions can be approximated well with only a few Fourier coefficients. This raises a natural question: what type of functions can be well approximated by a compact Fourier expansion?

We first discuss which functions can be represented exactly in Fourier domain with coefficients up to degree $d$. To answer this question, we show a tight connection between Fourier representations with bounded degree with decision trees with bounded depth. We start by defining a weighted decision tree. A decision tree for a weighted Boolean function $f : \{-1,1\}^n \rightarrow \mathbb{R}$ is a tree in which each inner node is labelled with one variable, and two out-going edges, one labelled with $-1$, and other one with $1$. The leaf nodes are labelled with non-negative real values. When evaluating the value on an input $x = x_1 x_2 \ldots x_n$, we start from the root node, and follow the corresponding out-going edges by inspecting the value of one variable at each step, until we reach one of the leaf nodes. The value at the leaf node is the output for $f(x)$. The depth of the decision tree is defined as the longest path from the root node to one of the leaf nodes. Figure 1 provides a decision tree representation for a weighted Boolean function. One classical result [18] states that if a function can be captured by a decision tree with depth $d$, then it can be represented with Fourier coefficients up to degree $d$:

**Theorem 1.** Suppose $f : \{-1,1\}^n \rightarrow \mathbb{R}$ can be represented by a decision tree of depth $d$, then all the coefficients whose degree is larger than $d$ is zero in $f$’s Fourier expansion: $\hat{f}(S) = 0$ for all $S$ such that $|S| > d$.

We can also provide the converse of Theorem 1:

**Theorem 2.** Suppose $f : \{-1,1\}^n \rightarrow \mathbb{R}$ can be represented by a Fourier expansion whose coefficients are up to degree $d$, then $f$ can be represented by the sum of several decision trees, each of which has depth at most $d$.

From Theorem 1 and Theorem 2, we can see a tight connection between the Fourier expansion and decision trees. Notice that the Fourier representation complements the classical way of approximating weighted functions in compact forms exploiting (conditional) independencies. To see this, suppose there is a decision tree of the same structure as in Figure 1, but has depth $d$. According to Theorem 1, it can be represented exactly with Fourier coefficients up to degree $d$. In this example, the number of non-zero Fourier coefficients is $O(2^d)$. Nonetheless, no two variables in figure 1 are independent with each other. Therefore, it’s not possible to decompose this factor into a product of smaller factors (exploiting independencies). Notice that the full contingency table representation of this factor has $O(2^{2d})$ entries.

If we are willing to accept an approximate representation, low degree Fourier coefficients can capture an even wider class of functions. We follow the standard notion of $\epsilon$-concentration:

**Definition 2.** The Fourier spectrum of $f : \{-1,1\}^n \rightarrow \mathbb{R}$ is $\epsilon$-concentrated on degree up to $k$ if and only if $W_{>k}[f] = \sum_{S \subseteq [n], |S| > k} \hat{f}(S)^2 < \epsilon$.

We say a CNF (DNF) formula has bounded width $d$ if and only if every clause (term) of the CNF (DNF) has at most $d$ literals. Linial [14] proved the following key result:

**Theorem 3** (Linial). Suppose $f : \{-1,1\}^n \rightarrow \{-1,1\}$ is computable by a DNF (or CNF) of width $w$, then $f$’s Fourier spectrum is $\epsilon$-concentrated on degree up to $O(w \log(1/\epsilon))$.

Linial’s result demonstrates the power of Fourier representations, since bounded width CNF’s (or DNF’s) include a very rich class of functions. Interestingly, the bound does not depend on the number of clauses, even though the clause-variable ratio is believed to characterize the hardness of satisfiability problems.
As a contribution of this paper, we extend Linial’s results to a class of weighted probabilistic graphical models, which are contractive with gap $1 - \eta$ and have bounded width $w$. To our knowledge, our extension from the deterministic case to the probabilistic case is novel.

**Definition 3.** Suppose $f(x) : \{-1, 1\}^n \to \mathcal{R}^+$ is a weighted function, we say $f(x)$ has bounded width $w$ iff $|x| \leq w$. We say $f(x)$ is contractive if with gap $1 - \eta$ and only if (1) for all $x$, $f(x) \leq 1$; (2) $\max_x f(x) = 1$; (3) if $f(x_0) < 1$, then $f(x) \leq \eta$.

The first and second conditions are mild restrictions. For a graphical model, we can always rescale each factor properly to ensure its range is within $[0, 1]$ and the largest element is 1. The approximation bound we are going to prove depends on the gap $1 - \eta$. Ideally, we want $\eta$ to be small. The class of contractive functions with gap $1 - \eta$ still captures a wide class of interesting graphical models. For example, it captures Markov Logic Networks [21], when the weight of each clause is large. The main theorem we are going to prove is as follows:

**Theorem 4.** (Main) Suppose $f(x) = \prod_{i=1}^{m} f_i(x_i)$, in which every $f_i$ is contractive function with width $w$ and gap $1 - \eta$, then $f$’s Fourier spectrum is $\epsilon$-concentrated on degree up to $O(w \log(1/\epsilon) \log_\eta \epsilon)$ when $\eta < 1$ and $O(w \log(1/\epsilon))$ when $\eta = 1$.

The proof of theorem[4] relies on the notion of random restriction and our extension to the Hastad’s Switching Lemma[9] to the class of weighted functions defined above.

**Definition 4.** Let $f(x) : \{-1, 1\}^n \to \mathcal{R}$ and $J$ be subset of all the variables $x_1, \ldots, x_n$. Let $\mathbf{z}$ be an assignment to remaining variables $\mathcal{J} = \{-1, 1\}^n \setminus J$. Define $f|_{\mathbf{z}} : \{-1, 1\}^J \to \mathcal{R}$ to be the restricted function of $f$ on $J$ by setting all the remaining variables in $\mathcal{J}$ according to $\mathbf{z}$.

**Definition 5.** ($\delta$-random restriction) For a function $f(x) : \{-1, 1\}^n \to \mathcal{R}$, a $\delta$-random restriction of $f$ is defined as $f|_{\mathbf{z}}$, when elements in $J$ is selected randomly with probability $\delta$, and $\mathbf{z}$ is formed by randomly setting variables in $\mathcal{J}$ to either $-1$ or 1. We also say $J|\mathbf{z}$ is a $\delta$-random restrict set.

With these definitions, we proved our weighted extension to the Hastad’s Switching Lemma:

**Lemma 1.** (Weighted Hastad’s Switching Lemma) Suppose $f(x) = \prod_{i=1}^{m} f_i(x_i)$, in which every $f_i$ is contractive function with width $w$ and gap $1 - \eta$. Suppose $J|\mathbf{z}$ is a $\delta$-random restrict set, then

$$Pr\left( \exists \text{ decision tree } h \text{ with depth } t, \ |h - f|_{\mathbf{z}}|_\infty \leq \gamma \right) \geq 1 - \frac{1}{2} \left( \frac{\delta}{1 - \delta} \right)^{uw(t)}.$$

in which $u = \left[ \log_\eta \gamma \right] + 1$ if $\eta < 1$ or $u = 1$ if $\eta = 1$ and $|.|_\infty$ means $\max |.|$.

The formal proof of Lemma[1] is based on a clever generalization of the proof by Razborov for the unweighted case [20], and is deferred to the supplementary materials. At a high level, the general idea of the proof is as follows. We call $J|\mathbf{z}$ fix factor $f_i$ into a “compressing” branch, if under such restriction, $|f_i|_{\mathbf{z}}|_\infty \leq \eta$. Call $J|\mathbf{z}$ a bad restriction set, if there are no decision tree $h$ with depth $t$, such that $|h - f|_{\mathbf{z}}|_\infty \leq \gamma$. Intuitively, if a random restriction set is bad, it can at most fix $u$ different factors to their “compressing” branches, since otherwise, the $u + 1$ factor will render $|f_i|_{\mathbf{z}}|_\infty \leq \gamma$, and a trivial all-zero decision tree $h_0$ satisfies $|h_0 - f|_{\mathbf{z}}|_\infty \leq \gamma$. Because it is highly unlikely that a random restrict set only fixes at most $u$ factors to their “compressing” branches but leave other factors open, we can conclude that there cannot be too many bad restrictions, in which case we obtain a bound on the probability of encountering a bad restriction set.

**Lemma 2.** Suppose $f(x) : \{-1, 1\}^n \to \mathcal{R}$ and $f(x) \leq 1$. $J|\mathbf{z}$ is a $\delta$-random restrict set. $t \in \mathbb{N}^+$, $\gamma > 0$ and let $\epsilon_0 = Pr\{-\exists \text{ decision tree } h \text{ with depth } t \text{ such that } |f|_{\mathbf{z}} - h|_\infty \leq \gamma\}$, then the Fourier spectrum of $f$ is $4(\epsilon_0 + (1 - \epsilon_0)\gamma^2)$-concentrated on degree up to $2t/\delta$.

**Proof.** We first bound $E_{J|\mathbf{z}} \left[ \sum_{S \subseteq \mathcal{J}, |S| > t} \hat{f}_{J|\mathbf{z}}(S)^2 \right]$. According to Lemma[1] with probability $1 - \epsilon_0$, there is a decision tree $h$ with depth $t$ such that $|f - h|_\infty \leq \gamma$. In this scenario,

$$\sum_{S \subseteq \mathcal{J}, |S| > t} \hat{f}_{J|\mathbf{z}}(S)^2 = \sum_{S \subseteq \mathcal{J}, |S| > t} \left( \hat{f}_{J|\mathbf{z}}(S) - \hat{h}(S) \right)^2.$$

(1)
This is because due to Theorem 1, \( \hat{h}(S) = 0 \) for all \( S \) such that \( |S| > t \). Because \( |f|_{J|x}(x) - h(x)| \leq \gamma \) for all \( x \), hence the right side of Equation 1 must satisfy

\[
\sum_{S \subseteq [n], |S| > t} \left( \hat{f}|_{J|x}(S) - \hat{h}(S) \right)^2 \leq \sum_{S \subseteq [n]} \left( \hat{f}|_{J|x}(S) - \hat{h}(S) \right)^2 = \mathbb{E} \left[ (f|_{J|x}(x) - h(x))^2 \right] \leq \gamma^2. \tag{2}
\]

The second to the last equality of Equation 2 is due to the Parseval’s Identity. With probability \( \epsilon_0 \), there are no decision trees close to \( f|_{J|x} \). However, because \( |f|_{J|x} \leq 1 \), we must have \( \sum_{S \subseteq [n], |S| > t} \hat{f}|_{J|x}(S)^2 \leq 1 \). Summarizing these two points, we have:

\[
\mathbb{E}_{J|x} \left[ \sum_{S \subseteq [n], |S| > t} \hat{f}|_{J|x}(S)^2 \right] \leq (1 - \epsilon_0)\gamma^2 + \epsilon_0.
\]

Using a known result \( \mathbb{E}_{J|x} \left[ \hat{f}|_{J|x}(S)^2 \right] = \sum_{U \subseteq [n]} Pr\{U \cap J = S\} \cdot \hat{f}(U)^2 \), we have:

\[
\mathbb{E}_{J|x} \left[ \sum_{S \subseteq [n], |S| > t} \hat{f}|_{J|x}(S)^2 \right] = \sum_{S \subseteq [n], |S| > t} \mathbb{E}_{J|x} \left[ \hat{f}|_{J|x}(S)^2 \right] = \sum_{U \subseteq [n]} Pr\{U \cap J > t\} \cdot \hat{f}(U)^2.
\]

The distribution of random variable \( |U \cap J| \) is Binomial(|U|, \( \delta \)). When \( |U| \geq 2t/\delta \), this variable has mean at least \( 2t \), using Chernoff bound, \( Pr\{|U \cap J| \leq t\} \leq (2/e)^t < 3/4 \). Therefore,

\[
(1 - \epsilon_0)\gamma^2 + \epsilon_0 \geq \sum_{U \subseteq [n]} Pr\{|U \cap J| > t\} \cdot \hat{f}(U)^2 \geq \sum_{U \subseteq [n], |U| \geq 2t/\delta} Pr\{|U \cap J| > t\} \cdot \hat{f}(U)^2 \geq \sum_{U \subseteq [n], |U| \geq 2t/\delta} \left(1 - \frac{3}{4}\right) \cdot \hat{f}(U)^2.
\]

We get our claim \( \sum_{|U| \geq 2t/\delta} \hat{f}(U)^2 \leq 4((1 - \epsilon_0)\gamma^2 + \epsilon_0) \).

**Proof.** Now we are ready to prove Theorem 4. Firstly suppose \( \eta < 1 \), choose \( \gamma = \sqrt{\epsilon/8} \), which ensures \( 4(1 - \epsilon_0)\gamma^2 \leq 1/2 \cdot \epsilon \). Next choose \( \delta = 1/(16uw + 1) \), \( t = C \log(1/\epsilon) \), which ensures

\[
\epsilon_0 = \frac{1}{2} \left( \frac{\delta}{1 - \delta} \right)^t = \frac{1}{2} \epsilon C.
\]

Choose \( C \) large enough, such that \( 4 \cdot 1/2 \cdot \epsilon C \leq 1/2 \cdot \epsilon \). Now we have \( 4((1 - \epsilon_0)\gamma^2 + \epsilon_0) \leq \epsilon \). At the same time, \( 2t/\delta = C \log(1/\epsilon)(16uw + 1) = O(w \log(1/\epsilon) \log \eta \epsilon) \).

\( \eta = 1 \) case is similar to the classical CNF (or DNF) case.

**4 Variable Elimination in Fourier Domain**

We propose an inference algorithm which works like the classic Variable Elimination (VE) Algorithm, except for passing messages represented in the Fourier domain.

The classical VE algorithm consists of two basic steps – the multiplication step and the elimination step. The multiplication step takes \( f \) and \( g \), and returns \( f \cdot g \), while the elimination step sums out one variable \( x_i \) from \( f \) by returning \( \sum_{x_i} f \). Hence, the success of VE procedure in Fourier domain depends on efficient algorithms to carry out the aforementioned two steps. A naive approach is to transform the representation back to the value domain, carry out the two steps there, then transform it back to Fourier space. While correct, this strategy would eliminate all the benefits of Fourier representations.

Luckily, the elimination step can be carried out in Fourier domain in a rather straightforward way:
Suppose \( f \) has a Fourier expansion: \( f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x) \), \( x_0 \in x \) is a variable. Then the Fourier expansion for \( f' = \sum_{x_0} f \) is: \( \sum_{S \subseteq [n]} \hat{f}'(S) \chi_S(x) \), where \( \hat{f}'(S) = 2\hat{f}(S) \) if \( x_0 \not\in S \) and \( \hat{f}'(S) = 0 \) if \( x_0 \in S \).

From Theorem 5, one only needs a linear scan of all the Fourier coefficients of \( f \) in order to compute the Fourier expansion for \( \sum_{x_0} f \). Suppose \( f \) has \( m \) non-zero coefficients in its Fourier representation, then this linear scan takes time \( O(m) \).

There are several ways to implement the multiplication step. The first option is to use school book multiplication. To multiply functions \( f \) and \( g \) directly in Fourier domain, one multiplies every pair of their Fourier coefficients, and then combines similar terms. If \( f \) and \( g \) have \( m_f \) and \( m_g \) terms in their Fourier representations respectively, this operation takes time \( O(m_f m_g) \). Because we are working on models in which exact inference is intractable, sometimes we need truncate the Fourier representation to prevent an exponential explosion. We find it is a good idea to keep Fourier coefficients with large absolute values, compared with always keeping ones with low degree. Even though we proved results that most Fourier weights are concentrated on low degree coefficients, keeping coefficients with large absolute values give us extra flexibility, especially when the whole graphical model is dominated by a few key variables.

As a second option, one can convert \( f \) and \( g \) to their value domain, multiply corresponding entries in the their value domain, and then convert the result back to the Fourier domain. Suppose the union of the domains of \( f \) and \( g \) has \( n \) variables, then the conversion between the two domains dominates the complexity, which is \( O(n \cdot 2^n) \). Nonetheless, when \( f \) and \( g \) are relatively dense, this method could have a better time complexity than the school book multiplication. In our implementation, we trade the complexity between the aforementioned two options, and always use the one with lower time complexity.

## 5 Experiments

### 5.1 Weight Concentration on Low Degree Coefficients

We first validate our theoretical results on the sparsity of Fourier representations and the weight concentration on low-degree coefficients. We choose to evaluate our results on random weighted 3-SAT instances with 20 variables. We choose small instances because we need compute the full Fourier spectrum. The weighted 3-SAT instances is specified by a CNF and a weight \( \eta \). Each factor corresponds to a clause in the CNF. When the clause is satisfied, the corresponding factor evaluates to 1, otherwise evaluates to \( \eta \). For each setting, we random generate 100 instances. For each instance, we compute the squared sum weight at each degree: \( W_k[f] = \sum_{S \subseteq [n], |S| = k} \hat{f}(S)^2 \).

Figure 2 shows the median value of the squared sum weight over 100 instances. As seen from the figure, the weights are concentrated on low degree coefficients, regardless of \( \eta \).
5.2 The Performance When Applying to Variable Elimination

We integrate the Fourier representation into the variable elimination algorithm, and evaluate its performance as an approximate probabilistic inference scheme to estimate the partition function of undirected graphical models. Our main comparison is against Mini-Bucket Elimination, since the two algorithms are of the same type, with the only difference being the way in which the messages are approximated. We obtained the source code from the author of Mini-Bucket Elimination, which includes sophisticated heuristics for splitting factors. The versions we obtained are used for Maximum A Posteriori Estimation (MAP). We augment this version to compute the partition function by replacing the maximization operators by summation operators. We also compare our VE algorithm with MCMC and Loopy Belief Propagation. We implemented the classical Ogata-Tanemura scheme [19] to estimate the partition function in MCMC. We use the implementation in LibDAI [17] for belief propagation.

We first compare on small instances for which we can compute ground truth using Ace [3]. To do this, we run on 15-by-15 Ising models with mixed coupling strengths and various field strengths. We run 20 instances for each coupling strength. For a fair comparison, we fix the size of the messages for both Fourier VE and Mini-bucket to 1,024. Figure 3 shows the result. Clearly the Fourier VE Algorithm outperforms the MCMC and the Mini-bucket Elimination. It also outperforms the Belief Propagation when the field strength is relatively strong.

Next we evaluate their performance on a synthetically generated benchmark. For one instance of this benchmark, we randomly generate factors of size 3 with low coupling weights. We then add a backdoor structure to each instance, by enforcing coupling factors of size 3 in which the 3 variables of the factor must take the same value. For these instances, we can compute the expected value of the partition function and compare it with the output of the algorithms. We report the results on Figure 4. Here the message size for Fourier VE is 1,048,576. However, when doing multiplications, we will shrink the message size to 1,024, since the complexity of school book multiplication is quadratic in the number of coefficients. The Mini-bucket approach is not reported, as it performs very poorly on these instances. These results show that the Fourier approach outperforms both mcmc and bp, and suggest that it can perform arbitrarily better than both approaches as the size of the backdoor increases.
6 Conclusion

In this paper, we explore a novel way to exploit compact representations of high-dimensional probability distributions in approximate probabilistic inference algorithms. Our approach is based on discrete Fourier Representation of weighted Boolean Functions, complementing the classical method to exploit conditional independence between the variables. We show that a large class of weighted probabilistic graphical models have a compact Fourier representation. This theoretical result opens up an entirely new way of approximating a probability distribution. We demonstrate the significance of this approach by applying it to the variable elimination algorithm and comparing the results with the bucket representation and other approximate inference algorithms, obtaining very encouraging results.

References


7 Supplementary Materials

7.1 Proof of Lemma 1

Let $R_n^t$ be the collection of restrictions on $n$ Boolean variables $x_1, \ldots, x_n$. Each restriction in $R_n^t$ leaves a set of $l$ variables $J = \{x_i, \ldots, x_{i+t}\}$ open, while fix all other variables $x_i \notin J$ to either -1 or 1. It is easy to see that the size of $R_n^t$ is given by:

$$|R_n^t| = \binom{n}{l} \cdot 2^{n-l}. \quad (3)$$

For a restriction $J|\mathbf{z} \in R_n^t$, call $J|\mathbf{z}$ bad if and only if for all decision tree $h$ with depth $t$, there exists at least one input $\mathbf{x}_f$ such that $|h(\mathbf{x}_f) - f|J|\mathbf{z}(\mathbf{x}_f)| > \gamma$. Let $B_n^t$ be the set of all bad restrictions, ie: $B_n^t = \{J|\mathbf{z} \in R_n^t : J|\mathbf{z}$ is bad$\}$. To prove the lemma, it is sufficient to prove that

$$\left| \frac{|B_n^t|}{|R_n^t|} \right| \leq \frac{1}{2} \left( \frac{l}{n-l} \right)^t \cdot 8uw. \quad (4)$$

In the proof that follows, for every bad restriction $\rho \in B_n^t$, we establish a bijection between $\rho$ and $(\xi, s)$, in which $\xi$ is a restriction in $R_{n-t}$ and $s$ is a certificate from a witness set $A$. In this case, the number of distinct $\rho$'s is bounded by the number of $(\xi, s)$ pairs:

$$|B_n^t| \leq |R_{n-t}^t| \cdot |A|. \quad (5)$$

For a restriction $\rho$, we establish the canonical decision tree for $f|\rho$ under precision $\gamma$ as follows:

1. We start with a fixed order for the variables and another fixed order for the factors.
2. If $f|\rho$ is a constant function, or $||f|\rho||_\infty \leq \gamma$, stop.
3. Otherwise, under restriction $\rho$, some factors evaluate to fixed values (all variables in these factors are fixed or there are free variables, but all assignments to these free variables lead to value 1), while other factors do not. Examine the factors according to the fixed factor order until reaching the first factor that still does not evaluate to a fixed value.
4. Expand on the open variables of this factor, under the fixed variable order specified in step 1. The result will be a tree (The root branch is for the first open variable. The branches in the next level is for the second open variable, etc).
5. Each leaf of this tree corresponds to $f|\rho\pi_1$, in which $\pi_1$ is a value restriction for all open variables of the factor. Recursively apply step 2 to 5 for function $f|\rho\pi_1$, until the condition in step 2 holds. Then attach the resulting tree to this leaf.

Figure 5 provides a graphical demonstration of a canonical decision tree.

Now suppose restriction $\rho$ is bad. By definition, for any decision tree of depth $t$, there exists at least one input $\mathbf{x}$, such that $|h(\mathbf{x}) - f|\rho(\mathbf{x})| > \gamma$. The canonical decision tree is no exception. Therefore, there must be a path $l$ in the canonical decision tree of $f|\rho$, which has more than $t$ variables. Furthermore, these $t$ variables can be split into $k$ (1 ≤ $k$ ≤ $t$) segments, each of which corresponds to one factor. Let $f_{i}$ (i ∈ {1, ..., $k$}) be these factors, and let $\pi_i$ be the assignments of the free variables for $f_{i}$ in path $l$. Now for each factor $f_{i}$, by the definition of the canonical decision tree, under the restriction $\rho\pi_1 \ldots \pi_{i-1}$, $f_{i}|\rho\pi_1 \ldots \pi_{i-1}$ must have a branch whose value is no greater than $\eta_i$ (otherwise $f_{i}|\rho\pi_1 \ldots \pi_{i-1}$ all evaluates to 1). We call this branch the “compressing” branch for factor $f_{i}|\rho\pi_1 \ldots \pi_{i-1}$. Let the variable assignment which leads to this compressing branch for $f_{i}|\rho\pi_1 \ldots \pi_{i-1}$ be $\sigma_i$. Let $\sigma = \sigma_1 \ldots \sigma_k$. Then we map the bad restriction $\rho$ to $\rho\sigma$ and an auxiliary advice string that we are going to describe.

It is self-explanatory that we can map from any bad restriction $\rho$ to $\rho\sigma$. The auxiliary advice is used to establish the backward mapping, ie, the mapping from $\rho\sigma$ to $\rho$. When we look at the result of $f|\rho\sigma$, we will notice that at least one factor is set to its compressing branch (because we set $f_{i}$ to its compressing branch in the forward mapping). Now there could be other factors set at their compressing branches (because of $\rho$), but an important observation is that: the number of factors at
Figure 5: A graphical illustration of a canonical decision tree.

their compressing branches cannot exceed \( u = \lceil \log \eta \gamma \rceil + 1 \), because otherwise, the other \( u - 1 \) factors already render \( ||f|\rho||_\infty \leq \gamma \), and the canonical decision tree should stop on expanding this branch. We therefore could record the index number of \( f_1 \) out of the all factors which are fixed at their compressing branches in the auxiliary advice string, so we can find \( f_1 \) in the backward mapping. Notice that this index number will be between 1 and \( u \), so it takes \( \log u \) bits to store it.

Now with the auxiliary information, we can identify which factor is \( f_1 \). The next task is to identify which variables in \( f_1 \) are fixed by \( \rho \), and which are fixed by \( \sigma_1 \). Moreover, if one variable is fixed by \( \sigma_1 \), we would like to know its correct values in \( \pi_1 \). To do this, we introduce additional auxiliary information: for each factor \( f_i \), suppose it has \( r_i \) free variables under restriction \( f_i|\rho \pi_1 \ldots \pi_{i-1} \), we use \( r_i \) integers to mark the indices of these free variables. Notice that every integer of this type is between 1 and \( w \) (therefore can be stored in \( \log w \) bits), since each \( f_i \) is with width at most \( w \). Also, it requires \( t \) integers of this type in total to keep this information, because we have \( t \) free variables in total for \( f_1, \ldots, f_k \).

Notice that it is not sufficient to keep these integers. We further need \( k - 1 \) separators, which tell which integer belongs to which factor \( f_i \). Aligning these integers in a line, we need \( k - 1 \) separators to break the line into \( k \) segments. These separators can be represented by \( t - 1 \) bits, in which the \( i \)-th bit is 1 if and only if there is a separator between the \( i \)-th and \((i+1)\)-th integer (we have \( t \) integers at most). With these two pieces of information, we are able to know the locations of free variables set by \( \sigma_i \) for each factor \( f_i \).

We further need to know the values for each variable in \( \pi_i \). Therefore, we add in another \( t \)-bit string, each bit is either 0 or 1. 0 means the assignment of the corresponding variable in \( \pi_i \) is the same as the one in \( \sigma_i \), 1 means the opposite.

With all this auxiliary information, we can start from \( \rho \sigma \), find the first factor \( f_1 \), further identify which variables are set by \( \sigma_1 \) in \( f_1 \), and set its correct values in \( \pi_1 \). Then we start with \( f|\pi_1 \), we can find \( \pi_2 \) in the same process, and continue. Finally, we will find all variables in \( \sigma \) and back up the original restriction \( \rho \).

Now to count the length of the auxiliary information, the total length is \( t \log u + t \log w + 2t - 1 \) bits. Therefore, we can have a one-to-one mapping between elements in \( B_n^t \) and \( R_{n+t}^{t} \times A \), in which the size of \( A \) is bounded by \( 2^{t \log u + t \log w + 2t - 1} = (uw)^t \cdot 2^{2t-1} \).
In all,
\[ |B^l_n| \leq \binom{n}{l\cdot t} \frac{(n_l - t)}{2^{n_l - t}} \cdot 2^{2^t - 1} \]  
(6)
\[ = \binom{n}{l\cdot t} \frac{1}{2} (8u)^t \]  
(7)
\[ = \frac{l(l - 1) \cdots (l - t + 1)}{(n - l + 1) \cdots (n - l + t)} \frac{1}{2} (8u)^t \]  
(8)
\[ \leq \frac{1}{2} \left( \frac{l}{n - l} \right)^t 8u. \]  
(9)

### 7.2 Proof of Theorem 2

For each term in the Fourier expansion whose degree is less than or equal to \( d \), we can treat this term as a weighted function involving less than or equal to \( d \) variables. Therefore, it can be represented by a decision tree, in which each path of the tree involves no more than \( d \) variables (therefore the tree is at most at the depth of \( d \)). Because \( f \) is represented as the sum over a set of Fourier terms up to degree \( d \), it can be also represented as the sum of the corresponding decision trees.

### 7.3 Proof of Theorem 5

Let the Fourier expansion of \( f \) be: \( f(x) = \sum_S \hat{f}(S)\chi_S(x) \), we have:
\[ f'(x \setminus x_0) = f(x \setminus x_0, x_0 = +1) + f(x \setminus x_0, x_0 = -1) \]  
(10)
\[ = \sum_{S: x_0 \in S} \hat{f}(S) \cdot \chi_{S \setminus x_0}(x \setminus x_0) \cdot 1 + \sum_{S: x_0 \in S} \hat{f}(S) \cdot \chi_{S \setminus x_0}(x \setminus x_0) \cdot (-1) \]  
(11)
\[ + \sum_{S: x_0 \notin S} \hat{f}(S) \cdot \chi_{S \setminus x_0}(x \setminus x_0) + \sum_{S: x_0 \notin S} \hat{f}(S) \cdot \chi_{S \setminus x_0}(x \setminus x_0) \]  
(12)
\[ = \sum_{S: x_0 \notin S} 2 \cdot \hat{f}(S) \cdot \chi_{S \setminus x_0}(x \setminus x_0). \]  
(13)