Learning to Round for Discrete Labeling Problems

Pritish Mohapatra
IIIT Hyderabad

C. V. Jawahar
IIIT Hyderabad

M. Pawan Kumar
University of Oxford

Abstract

Discrete labeling problems are often solved by formulating them as an integer program, and relaxing the integrality constraint to a continuous domain. While the continuous relaxation is closely related to the original integer program, its optimal solution is often fractional. Thus, the success of a relaxation depends crucially on the availability of an accurate rounding procedure. The problem of identifying an accurate rounding procedure has mainly been tackled in the theoretical computer science community through mathematical analysis of the worst-case. However, this approach is both onerous and ignores the distribution of the data encountered in practice. We present a novel interpretation of rounding procedures as sampling from a latent variable model, which opens the door to the use of powerful machine learning formulations in their design. Inspired by the recent success of deep latent variable models we parameterize rounding procedures as a neural network, which lends itself to efficient optimization via back-propagation. By minimizing the expected value of the objective of the discrete labeling problem over training samples, we learn a rounding procedure that is more suited to the task at hand. Using both synthetic and real world data sets, we demonstrate that our approach can outperform the state-of-the-art hand-designed rounding procedures.

1 Introduction

A discrete labeling problem is defined over a set of random variables, each of which needs to be assigned a value from a discrete label set. An assignment of values to all the random variables is referred to as a labeling. The large number of putative labelings (exponential in the number of random variables) are quantitatively distinguished from each other by means of an energy function. The goal of the discrete labeling problem is to compute the labeling with the minimum energy.

The discrete labeling problem plays a key role in many areas of computer science. For example, in machine learning it is required for maximum *a posteriori* estimation in graphical models. In theoretical computer science, several classical tasks such as vertex cover and graph partitioning can be viewed as discrete labeling problems. While special cases of the problem can be solved exactly in polynomial time [20], in general it is known to be NP-hard. A popular approach to obtain an approximate solution is to formulate the discrete labeling problem as an integer program, which can then be relaxed to obtain an easy-to-solve continuous optimization problem. While the continuous relaxation is closely related to the original integer program, its optimal solution can be fractional (it often is). Thus, a key requirement for using continuous relaxations is the availability of an accurate *rounding procedure*, that is, a method that can convert the optimal fractional solution to a feasible integer solution with low energy value. The most popular family of rounding procedures is based on randomized algorithms, which generally pose rounding as sampling from a distribution parameterized by the fractional solution.

The design of sampling based rounding procedures generally involves highly sophisticated mathematical analysis to establish the expected value of the energy of the rounded solution in the worst-case. In special cases such as uniform metric labeling, a lot of simple procedures, backed by strong mathematical analysis, have been proposed. While such an approach has important theoretical consequences (specifically, it establishes the computational complexity of various instances of discrete labeling problems), from a practitioners point of view it suffers from two major deficiencies. First, it is highly onerous and therefore cannot scale well as each new class of problems would require

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

expert knowledge to design an appropriate rounding procedure. Second, it focuses on the worst-case scenario (for ease of analysis), which often does not occur in practice.

In order to alleviate the aforementioned deficiencies, we propose a novel machine learning framework that learns to round the solutions of a continuous relaxation using a training data set. The key observation of our approach is that many randomized rounding procedures can be viewed as sampling from a joint distribution of two types of random variables: (i) variables whose marginal probability is provided by the optimal fractional solution of the relaxation; and (ii) a set of appropriately designed latent variables. Viewed in this way, the problem of rounding readily lends itself to machine learning techniques. In this work, we employ a deep neural network that consists of two stages. In the first stage, it projects a low-dimensional input (namely, the optimal fractional solution of a relaxation) to a potentially high-dimensional space of the aforementioned latent random variables. In the second stage, it projects the representation encoded by the latent random variables back to the original space of feasible fractional solutions, to give the output of the neural network. The integer solution of the problem is obtained by using the simplest rounding procedure on the output. The parameters of the deep neural network are estimated by optimizing a differentiable learning objective that minimizes the expected energy of the labeling for a given set of training samples.

Our approach can be readily applied to a large class of existing relaxations, including those based on linear programming [6, 13], quadratic programming [17] and second-order cone programming [15, 16]. By learning the neural network for a given set of samples that implicitly define the data distribution, we obtain more suitable rounding procedures for real-world problems compared to the ones designed for the worst-case. We demonstrate the efficacy of our approach on several instances of the discrete labeling problem, including uniform metric labeling and truncated linear and quadratic labeling by comparing them to the hand-designed rounding procedures proposed in the theoretical computer science literature.

2 Related Work

As mentioned earlier, most of the literature on randomized rounding procedures focuses on hand-designed algorithms for special instances of the discrete labeling problem. Examples include interval rounding and hierarchical rounding for linear programming relaxations of metric labeling [6, 12], hyperplane rounding for semidefinite relaxations of graph partitioning [8], and contention resolution for multilinear relaxations for submodular maximization [22]. While such an approach is of great theoretical importance, the complexity of mathematical analysis and the focus on the worst-case makes it less appealing in practice.

Our work exploits the close relationship between randomized rounding and sampling. Similar to rounding, there have been several hand-designed sampling algorithms that have been proposed in machine learning [2, 5]. Furthermore, there has also been some effort in learning to sample using a training data set [18, 25]. However, to the best of our knowledge, ours is the first approach to exploit the connection between sampling and rounding for discrete labeling problems. Moreover, unlike sampling in which the main objective is to maximize sample fidelity with the original distribution, our goal is to minimize a task specific energy function by learning to round.

There is also a rich history in machine learning for learning latent spaces for a particular problem [4, 9, 10]. The one most closely related to our approach is the variational auto-encoder (VAE) [11], which uses a deep neural network in order to obtain an expressive latent representation of the input space. However, there are several key differences between our framework and the VAE. First, our training objective is designed to maximize the accuracy rounding, instead of minimizing the reconstruction error. Second, while the VAE aims to learn the parameters of a neural network that generalizes well across the entire data distribution, our problem is concerned with learning the parameters for each individual instance of the problem.

Finally, there has been work on using reinforcement learning for function optimization [3, 23]. Reinforcement learning also optimizes an entropy regularized expected energy. However, in our case, a key difference is that we compute the expected energy and its exact gradient analytically instead of relying on estimates of the policy gradient. This reduces the variance in the gradient, thereby enabling efficient learning. Another key difference is that, unlike the reinforcement learning based methods, we also condition our model with respect to a primal marginal solution which allows a trained model to generalize to unseen energy functions.

3 Preliminaries

Discrete Labeling Problem. A discrete labeling problem is defined over a set of random variables $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$, each of which can take a value from the label set $\mathcal{L} = \{l_1, l_2, \cdots, l_h\}$. An assignment of values to all the random variables is called a labeling, and is denoted by $\mathbf{x} \in \mathcal{L}^n$. The labelings are quantitatively distinguished from each other by the means

of an energy function $E_\theta : \mathcal{L}^n \rightarrow \mathbb{R}$, which consists of a sum of potential functions. For the sake of clarity, we will restrict ourselves to a pairwise energy function. In other words, the energy of a labeling $\mathbf{x}$ is given by:

$$E_\theta(\mathbf{x}) = \left\{ \sum_{a \in [n]} \theta_a(x_a) + \sum_{(a,b) \in [n]^2} \theta_{ab}(x_a, x_b) \right\}. \quad (1)$$

Here, $[n] = \{1, ..., n\}$, and $\theta_a(x_a)$ and $\theta_{ab}(x_a, x_b)$ are short hand for $\theta_a(X_a = x_a)$ and $\theta_{ab}(X_a = x_a, X_b = x_b)$ respectively. While, $\theta_a(X_a = x_a)$ denotes the unary potential of assigning the label x_a to the random variable X_a, $\theta_{ab}(X_a = x_a, X_b = x_b)$ denotes the pairwise potential of assigning the labels x_a and x_b to random variables X_a and X_b respectively. The discrete labeling problem is specified as follows: $\mathbf{x}^* = \text{argmin}_{\mathbf{x} \in \mathcal{L}^n} E_\theta(\mathbf{x})$. The general discrete labeling problem allows for high-order potentials (that is, potentials that depend on the labels of an arbitrarily large subset of random variables). Our approach can be trivially generalized to handle such potentials by suitably modifying the training objective. We assume that the potentials are finite valued. In other words, all the labelings are valid, and the task is to identify the one with the minimum energy. This assumption is mainly due to the fact that we parameterize our framework via a neural network, which requires gradients for all possible parameters in order to utilize the back-propagation algorithm during training.

Integer Programming Formulation. The discrete labeling problem can be reformulated as an integer program using indicator variables $y_a(i) \in \{0, 1\}$ corresponding to all random variables $X_a \in \mathcal{X}$ and labels $l_i \in \mathcal{L}$. Each indicator variable $y_a(i) = 1$, if $X_a = l_i$ and 0 otherwise. Formally, the following program provides the minimum energy labeling:

$$\begin{aligned} \min \quad & \sum_{a \in [n]} \sum_{l_i \in \mathcal{L}} \theta_a(i) y_a(i) \\ & + \sum_{(a,b) \in [n]^2} \sum_{(l_i, l_j) \in \mathcal{L}^2} \theta_{ab}(i,j) y_a(i) y_b(j), \\ \text{s.t.} \quad & \sum_{l_i \in \mathcal{L}} y_a(i) = 1 \quad \forall a \in [n], \\ & y_a(i) \in \{0, 1\} \quad \forall a \in [n], \forall l_i \in \mathcal{L}. \end{aligned} \quad (2)$$

Here, the objective function is just a reformulation of the energy function in equation 1 using the binary indicator variables. The first constraint ensures that each random variable is assigned exactly one label, while the second constraint ensures that the optimization variables are binary.

Continuous Relaxations. The above integer program can be relaxed to obtain a continuous optimization problem. Several relaxations have been proposed in the literature, including linear programming [6, 13], quadratic programming [17] and second-order cone programming [15, 16]. All the aforementioned relaxations drop the integrality constraints, and instead enforce the variables $\mathbf{y}_a$ to belong to a probability simplex, that is, $y_a(i) \geq 0$ and $\sum_i y_a(i) = 1$. The resulting continuous problem is then solved to obtain an optimal fractional solution $\mathbf{y}^*$.

Randomized Rounding Procedures. Given a feasible fractional solution $\bar{\mathbf{y}}$, a rounding procedure treats $\bar{\mathbf{y}}_a(i)$ as the probability of assigning the label l_i to the random variable X_a. Several rounding procedures that satisfy this property have been proposed in the literature. We refer the interested reader to [21, 24] for examples. Although most often these algorithms are applied on optimal fractional solutions, the analysis applies to all feasible solutions. In our work, we will use the simplest rounding procedure to obtain integer solutions from the output of a deep neural network, which we refer to as complete rounding. The main steps of the complete rounding procedure are described in Algorithm 1. Complete rounding computes the cumulative distribution of $\mathbf{y}_a$ for each $X_a \in \mathcal{X}$ (step 3). It then samples from the cumulative distribution using the same real number $r \in [0, 1]$ for all the random variables (step 4). Note that the use of the same real number is important. Otherwise, the rounding procedure can be shown to produce arbitrarily bad labelings (see [14] for examples).

Algorithm 1 *The complete rounding algorithm.*

Input: A fractional solution $\mathbf{y}$ of a relaxation.

1: Sample a real number r from uniform distribution over $[0, 1]$.
2: **for** `all` $X_a \in \mathcal{X}$ **do**
3: `Define,` $Y_a(0) = 0$, $Y_a(i) = \sum_{j=1}^{i} y_a(j)$.
4: Assign label l_i to random variable X_a `if` $Y_a(i-1) \leq r < Y_a(i)$.
5: **end for**

4 Trainable Latent Variable Model For Rounding

Given a feasible (or optimal) fractional solution $\mathbf{y}$ of a continuous relaxation, randomized rounding procedures can be viewed as assigning a label to the set of discrete random variables $\mathcal{X}$ from the set $\mathcal{L}$ such that $\Pr(X_a = l_i) = y_a(i)$. There are several ways in which one can achieve this goal, including the simple complete rounding procedure described above. What separates a good rounding procedure from a bad one is the joint probability of the labeling of a subset of

random variables. Since we have restricted our description to pairwise energy functions, the key criterion for differentiating two rounding procedures is the joint probability of assigning two random variables X_a and X_b to the labels l_i and l_j respectively.

To illustrate the weakness of complete rounding, consider the following simple example of a uniform metric labeling problem (also referred to as the Potts model) defined over n random variables, each of which can take 1 of n possible labels. The unary potential for the a^{th} random variable X_a is defined as follows:

$$\theta_a(i) = \begin{cases} \infty & \text{if } i = a, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Every pair of random variables X_a and X_b are assumed to be connected by an edge and the pairwise potential between them is defined as follows: $\theta_{ab}(i,j) = \delta(i \neq j)$. Any labeling that assigns the same label to all but one of the random variables would be optimal for this discrete labeling problem.

On the other hand, the optimal fractional solution $\mathbf{y}$ obtained for the LP relaxation of this discrete problem can be defined as follows:

$$y_a(i) = \begin{cases} 1/(n-1) & \text{if } i \neq a, \\ 0 & \text{otherwise.} \end{cases} \quad ,\forall a \in \{1, \ldots, n\} \tag{4}$$

It can be verified that, when performing complete rounding on this optimal fractional solution, if the random number r is between $(i-1)/(n-1)$ and $i/(n-1)$ for $i = 1, 2, \cdots, n-1$, the first i variables take the label $i+1$, and the remaining ones take the label i. Specifically, it is impossible for it to do so when the random number $r \in [1/(n-1), (n-2)/(n-1)]$. Therefore, the probability of the complete rounding procedure to output an optimal integral solution (that is assigning the same label to all but 1 of the random variables) is only $2/(n-1)$. In order to overcome the deficiency of complete rounding, Kleinberg and Tardos [12] proposed a more suitable rounding procedure for uniform metric labeling. In what follows, we provide a novel interpretation of their procedure based on latent variable models, which will motivate our general learning based framework.

Consider an augmented label set $\mathcal{L}' = \{l_0\} \cup \mathcal{L}$, where the auxiliary label l_0 indicates that a random variable has not yet been assigned a label. We define a latent variable Z, which can take a value from the label set $\mathcal{L}$. The probability $\Pr(Z = l_i) = 1/h$ for all $l_i \in \mathcal{L}$. Furthermore, we define $\Pr(X_a|Z)$ as

$$\Pr(X_a = l_i | Z = l_j) = \begin{cases} y_a(i) & \text{if } i = j \neq 0, \\ 1 - y_a(i) & \text{if } i = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

In order to obtain a labeling, we use an iterative procedure. At each iteration, we first sample from the distribution $\Pr(Z)$ to fix the value of the latent variable. Next, we use complete rounding on the distributions $\Pr(X_a|Z)$ for all random variables $X_a \in \mathcal{X}$. If a random variable is assigned a label $l_i \in \mathcal{L}$ (that is, not the label l_0), then we fix its label to l_i. For all the unassigned random variables (that is, those with the label l_0) we repeat the above process until a valid labeling has been obtained. The above iterative procedure can be viewed as sampling from the joint distribution $\Pr(Z, X_a) = \sum_{l_i \in \mathcal{L}} \Pr(Z = l_i) \Pr(X_a | Z = l_i)$, and marginalizing out the latent variable. It can be verified that, in the case of the illustrative example for uniform metric labeling, the above procedure always outputs an optimal integral solution. It can do so because of the use of latent variables. This can thus result in an improved expected energy of the output labeling compared to complete rounding.

At first sight, the choice of the latent variable Z, its distribution $\Pr(Z)$ and the conditional distributions $\Pr(X_a|Z)$ may appear arbitrary. However, there are two factors that governed their design. First, they have to exploit the structure of the pairwise potentials, which encourages a pair of random variables to be assigned the same label. Second, the latent variable and the corresponding distributions have to be simple enough to lend themselves to worst-case mathematical analysis. We now consider each of the two aforementioned factors to motivate our methodology. The first factor implies that, for every different family of the discrete labeling problems, we would need to design a new rounding procedure. Indeed, for truncated linear and quadratic labeling, the random variable represents an interval of consecutive labels, instead of a single label [6]. Given the vast number of choices, it is clear that the process of hand-designing a rounding procedure is too tedious and would not scale well to meet the demands of an increasingly automated world. However, our novel interpretation of rounding procedures as latent variable models opens the door to the use of powerful machine learning frameworks. The second factor implies that the simple form of distributions is not a requirement in practice as we are not interested in worst-case analysis. Thus, inspired by the recent success of deep latent variable models such as VAE [11], we propose to parameterize rounding procedures for discrete labeling problems using a deep neural network. This allows us to learn highly complex latent variables and the corresponding distributions (which can depend on the fractional solution $\mathbf{y}$) using a set of training samples that implicitly define the data distribution of interest. In what follows, we describe our model and its end-to-end differentiable training objective in detail.

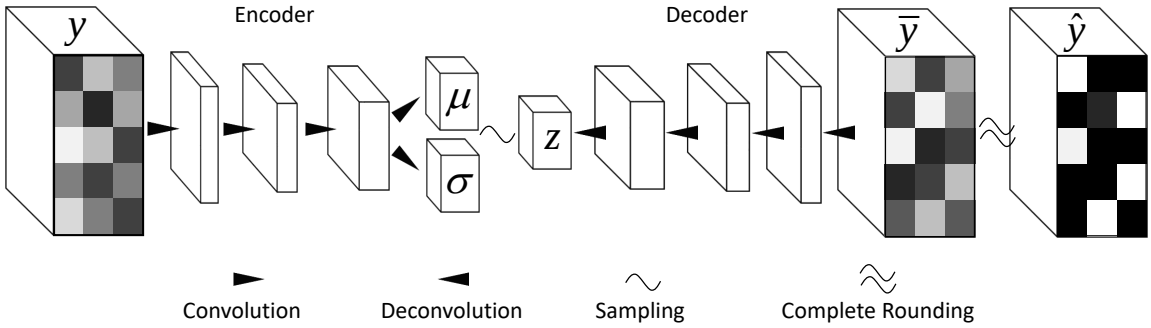

Figure 1: *Our deep latent variable model that includes an encoder, a latent layer and a decoder. The encoder takes in a feasible fractional solution* $\mathbf{y}$ *as input and computes the parameters* $[\boldsymbol{\mu}, \boldsymbol{\sigma}]$ *of the latent variable distribution* $\Pr(Z)$*. The decoder takes a sample from* $\Pr(Z)$ *and computes the output fractional solution* $\bar{\mathbf{y}}$*. Finally, approximate integral solution* $\hat{\mathbf{y}}$ *is obtained by performing complete rounding on* $\bar{\mathbf{y}}$*.*

4.1 Prediction Using Deep Latent Variable Model

We use deep neural networks to model both the distribution $\Pr(Z)$ associated with the latent random variables Z and the conditional distribution $\Pr(X|Z)$ that projects back into the space of feasible solutions. Overall, our network is composed of an encoder ($\mathcal{E}_{\boldsymbol{\alpha}}$), a layer of latent variables (Z) and a decoder ($\mathcal{D}_{\boldsymbol{\beta}}$). Here, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the learnable parameters of the encoder and decoder respectively. Figure1 shows a representation of our deep latent variable model. While we have shown a fully convolution network here, any network architecture can be used for this purpose.

The detailed description for rounding using our deep latent variable model is outlined in Algorithm 2. First, the parameters $\boldsymbol{\phi}$ of the latent variable distribution $\Pr(Z;\boldsymbol{\phi})$ are computed as outputs by the encoder $\mathcal{E}_{\boldsymbol{\alpha}}$, that is, $\boldsymbol{\phi}(\mathbf{y}) = \mathcal{E}_{\alpha}(\mathbf{y})$ (step 1). We then sample $\mathbf{z}$ from the distribution $\Pr(Z;\boldsymbol{\phi})$ (step 2) and pass it through the decoder $\mathcal{D}_{\boldsymbol{\beta}}$ to get an output fractional solution $\bar{\mathbf{y}} = \mathcal{D}_{\boldsymbol{\beta}}(\mathbf{z})$ (step 3). This output fractional solution $\bar{\mathbf{y}}$ parameterizes the distribution $\Pr(X = \hat{\mathbf{y}}|Z;\bar{\mathbf{y}})$, where $\hat{\mathbf{y}}$ is a feasible integral solution. We perform complete rounding as described in Algorithm 1 on $\bar{\mathbf{y}}$ to get our output $\hat{\mathbf{y}}$ (step 4). In practice, we perform several iterations of complete rounding on $\bar{\mathbf{y}}$ and choose $\hat{\mathbf{y}}$ to be the integral solution with the lowest energy $E_\theta(\hat{\mathbf{y}})$. In our experiments, we parameterize the latent variable distribution $\Pr(Z;\boldsymbol{\phi})$ as a diagonal Gaussian with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$, that is, $\boldsymbol{\phi} = [\boldsymbol{\mu}, \boldsymbol{\sigma}]$. However, one can also assume other parameterized models like the Bernoulli distribution for the latent variables.

4.2 Training Methodology

Our aim here is to learn the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of the encoder and the decoder respectively using training set which is assumed to contain independently sampled instances from the data distribution. The training data set D can consist of several instances of any particular discrete labeling problem that vary in terms of the values for the parameters of the energy function E_θ, while maintaining the same underlying structure. For example, in case of the problem of semantic segmentation of images, we can have different images as the samples of the training set while assuming the same structure for inter-pixel dependencies for all the images. To be specific, each instance s_t in the training set $D = \{s_1, \ldots, s_N\}$ should include the parameters θ_t of the energy function and a feasible fractional solution $\mathbf{y}_t$ from a continuous relaxation of the original discrete labeling problem.

Algorithm 2 *Rounding using deep latent variable model.*

Input: A feasible solution $\mathbf{y}$ of the continuous relaxation.

1: Compute parameters of the latent variable distribution by passing through the encoder: $\boldsymbol{\phi}(\mathbf{y}) = \mathcal{E}_{\alpha}(\mathbf{y})$.
2: Sample $\mathbf{z}$ from the latent variable distribution $P(Z;\boldsymbol{\phi})$.
3: Compute output fractional solution by passing $\mathbf{z}$ through the decoder: $\bar{\mathbf{y}} = \mathcal{D}_{\boldsymbol{\beta}}(\mathbf{z})$.
4: Perform complete rounding as described in Algorithm 1 to obtain an integral solution $\hat{\mathbf{y}}$ corresponding to $\bar{\mathbf{y}}$.

Output: An approximate integral solution $\hat{\mathbf{y}}$ to the original discrete labeling problem.

As has been discussed elaborately in the previous sections, our objective is to find the integral solution $\hat{\mathbf{y}}$ with the lowest possible energy $E_\theta(\hat{\mathbf{y}})$. Therefore, given the training data, we should try to minimize the energy $E_\theta(\hat{\mathbf{y}})$ of the output integral solutions for

all of the training examples. However, since ours is a stochastic model, it is appropriate for us to minimize the expected energy with respect to the distribution $\Pr(X; y_t, \boldsymbol{\alpha}, \boldsymbol{\beta})$ for all training samples $s_t \in \mathcal{D}$.

Given a training sample $s_t \in \mathcal{D}$ and a value $\mathbf{z}$ from the latent variable distribution $\Pr(Z; \boldsymbol{\phi}(y_t, \boldsymbol{\alpha}))$, we can sample several integral solutions using the complete rounding procedure. The expectation with respect to the distribution induced by the complete rounding procedure can be evaluated exactly by using the expression presented in *Lemma 11* of [14]. To be specific, the expected pairwise energy can be computed as,

$$\mathbb{E}(\theta_{ab}(\hat{\mathbf{y}}_a, \hat{\mathbf{y}}_b)) = \sum_{i=1}^{h-1} Y_a(i)\Theta_1(i) + \sum_{j=1}^{h-1} Y_b(j)\Theta_1(j) + \sum_{i=1}^{h-1}\sum_{j=1}^{h-1} |Y_a(i) - Y_b(j)|\Theta_2(i,j). \quad (6)$$

Here, Y_a and Y_b are the cumulative distributions with respect to $\mathbf{y}_a$ and $\mathbf{y}_b$, and Θ_1 and Θ_2 are defined as follows,

$$\Theta_1(i) = \tfrac{1}{2}(\theta_{ab}(l_i, l_1) + \theta_{ab}(l_i, l_h) - \theta_{ab}(l_{i+1}, l_1) - \theta_{ab}(l_{i+1}, l_h)), \forall i \in \{1, .., h-1\}, \quad (7)$$

$$\Theta_2(i,j) = \tfrac{1}{2}(\theta_{ab}(l_i, l_{j+1}) + \theta_{ab}(l_{i+1}, l_j) - \theta_{ab}(l_i, l_j) - \theta_{ab}(l_{i+1}, l_{j+1})), \forall i, j \in \{1, .., h-1\}. \quad (8)$$

The overall expected energy of the output integral solution $\hat{\mathbf{y}}$ with respect to complete rounding then can be computed as,

$$\mathop{\mathbb{E}}_{\hat{\mathbf{y}} \sim \Pr(X|Z;\boldsymbol{\beta})} [E_\theta(\hat{\mathbf{y}})] = \textstyle\sum_{X_a \in \mathcal{X}_a} \langle \theta_a(\hat{\mathbf{y}}_a), \mathbf{y}_a \rangle + \sum_{(X_a, X_b) \in \mathcal{X}^2} \mathbb{E}(\theta_{ab}(\hat{\mathbf{y}}_a, \hat{\mathbf{y}}_b)). \quad (9)$$

As discussed above, the expected energy of the output integral solution $\hat{\mathbf{y}}$, with respect to the distribution induced by the complete rounding procedure, has a closed form expression. This allows us to analytically compute the gradient of this expected energy with respect to the output fractional solution $\bar{\mathbf{y}}$ and thus have no variance in gradient estimation.

For our training objective, we further take expectation of the above computed expected energy with respect to the latent variable distribution $\Pr(Z; \boldsymbol{\phi})$ to obtain,

$$\mathop{\mathbb{E}}_{\hat{\mathbf{y}} \sim \Pr(X; y_t, \boldsymbol{\alpha}, \boldsymbol{\beta})} [E_{\theta_t}(\hat{\mathbf{y}})] = \mathop{\mathbb{E}}_{\mathbf{z} \sim \Pr(Z; \boldsymbol{\phi}(y_t, \boldsymbol{\alpha}))} \left[\mathop{\mathbb{E}}_{\hat{\mathbf{y}} \sim \Pr(X|Z=\mathbf{z}; \boldsymbol{\beta})} [E_{\theta_t}(\hat{\mathbf{y}})] \right]. \quad (10)$$

Since analytical computation of this expected energy is not feasible, we have to estimate it by using k samples from the latent variable distribution. When the latent variable distribution is chosen to be a Gaussian distribution, the gradient of this sampling based estimation of the expected energy with respect to the parameters of the distribution can be computed using the reparameterization trick. This results in very low variance for the gradient estimation. In the general case, the reinforce algorithm can be used to estimate the gradient.

Finally, our training objective is obtained by taking expectation of the above computed expected energy with respect to the data distribution $\mathcal{D}$. As in case of all machine learning frameworks, we estimate the expectation over the data distribution by using samples from the training data set D. Then the training objective function can be written as

$$J_E(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathop{\mathbb{E}}_{s_t \sim D} \left[\mathop{\mathbb{E}}_{\hat{\mathbf{y}} \sim \Pr(X; y_t, \boldsymbol{\alpha}, \boldsymbol{\beta})} [E_{\theta_t}(\hat{\mathbf{y}})] \right] \quad (11)$$

While optimizing the expected energy, the model might sometime get stuck in local minima corresponding to integral $\bar{\mathbf{y}}$'s that lead to bad approximate solutions. In order to avoid such bad local minima during optimization, we regularize the output fractional solutions $\bar{\mathbf{y}}$ to bias them towards having higher entropy. Specifically, we add the following entropy based regularization term to our objective function,

$$J_R(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathop{\mathbb{E}}_{s_t \sim D} \left[\mathop{\mathbb{E}}_{\mathbf{z} \sim \Pr(Z; \boldsymbol{\phi}(y_t, \boldsymbol{\alpha}))} \left[\sum_{a \in [n]} \bar{\mathbf{y}}_a(\mathbf{z}, \boldsymbol{\beta}) \log \bar{\mathbf{y}}_a(\mathbf{z}, \boldsymbol{\beta}) \right] \right]. \quad (12)$$

Overall, we optimize the objective function $J = J_E + \lambda J_R$ for learning the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of the encoder and the decoder respectively. Here, λ is a hyper-parameter which we fix appropriately. Since, this objective function is differentiable, we can use the standard back-propagation algorithm to train our network parameters. In order to back-propagate through the sampling process at the latent layer, we use the re-parameterization trick for Gaussian distribution as proposed in [11].

5 Experiments

We demonstrate the efficacy of our approach, described in the previous section, on discrete labeling problems using both synthetic as well as real world data.

5.1 Discrete Labeling Of Densely Connected Graphs

Problem. We consider the problem of labeling a set of 25 random variables from a discrete set of 21 labels.

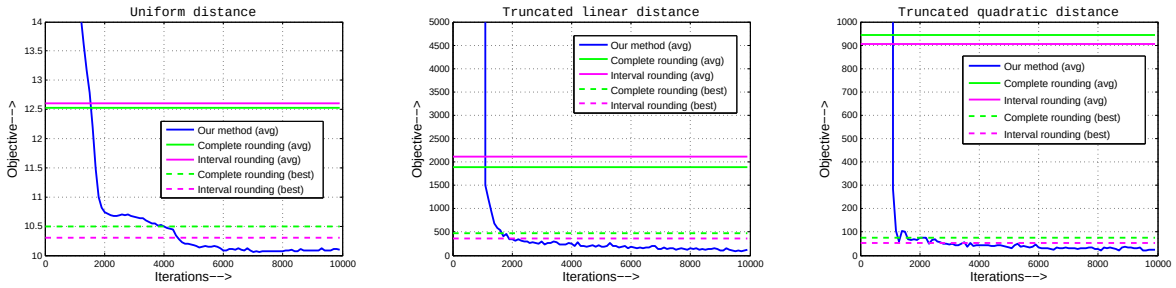

Figure 2: *Objective function Vs. iterations during optimization over the training set of the synthetic data set. Here, 'avg' refers to expected energy (E_{avg}) and 'best' refers to minimum energy (E_{min}).*

Labeling Metric	Max Rounding	Complete Rounding		Interval Rounding		Our method	
	E_{avg}=E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}
Uniform	12.204	12.532	10.435	12.437	10.327	10.180	10.103
Truncated Linear	74.423	1887.213	19.427	1936.361	17.787	153.801	14.762
Truncated Quadratic	152.532	904.821	84.640	876.610	72.418	71.469	65.359

Table 1: *Average (E_{avg}) and minimum (E_{min}) energy obtained by the different methods on the test set for the discrete labeling problem on synthetic data.*

We assume that all the random variables are dependent on each other and these dependencies are encoded by a densely connected graph. We consider three different types of distance metrics for encoding label compatibility, namely, the uniform, truncated linear and truncated quadratic distances. The uniform distance metric is also known as the Potts model and is defined as $d(i,j) = \delta(i \neq j)$. Where as, the truncated linear and truncated quadratic distances are defined as $d(i,j) = \min(|i-j|, M)$ and $d(i,j) = \min(|i-j|^2, M)$ respectively. We use $M = 10$ for our experiments. We also assume a spatial arrangement of the random variables in a 5×5 lattice such that each vertex v_a has a location coordinate $c_a = (x_a, y_a)$. As such, the pairwise potential associated with the edge between vertices v_a and v_b is defined as $\theta_{ab}(i,j) = w_{ab}d(i,j)$, where, $w_{ab} = \exp(-||c_a - c_b||_2^2/\sigma_s^2)$ is a spatial Gaussian weight with scaling factor σ_s.

Dataset. We use a synthetically generated collection of 100 graphs for this experiment. For each graph, we randomly generate the unary potential associated with each vertex from a uniform distribution over $[0, 1]$. We also set the parameters σ_s and a weighing factor τ for the pairwise potentials, from a uniform distribution over $[-10^5, 10^5]$. From our collection of 100 graphs, we use 50 for training and 50 for testing.

Methods. We train a deep latent variable model as described in Section 4.1 with both the encoder and the decoder being modeled by neural networks composed of fully connected layers. We use the efficient proximal LP solver proposed in [1] to obtain fractional solutions for training our model. The scaling factor for the entropy regularization term in our training objective function is fixed to $\lambda = 1$ for all our experiments. We compare our method with other standard rounding procedures like max-rounding, complete rounding and interval rounding[6, 12].

Results. Figure 2 shows the progression of the objective function when our model is being optimized on the training data set. As can be seen, our model swiftly learns to do accurate rounding and outperforms the hand designed rounding procedures like complete and interval rounding. Even though this performance corresponds to the training set, it is significant because unlike training for tasks like classification, we don't use any kind of ground-truth information. However, it is not necessary to train the model from scratch for each new sample. Instead, we only finetune the trained model for the new sample by optimizing over it for a very few iterations (20 in this case). Such finetuning is possible because we do not need any kind of ground-truth information for it. To evaluate, for each test sample, we use different randomized rounding methods, including ours, to sample 1000 integral solutions and compute the expected energy E_{avg} and the minimum energy E_{min} over this set. We report the mean values of E_{avg} and E_{min} computed over the entire test set. Table 1 shows the results for the uniform, truncated linear and truncated quadratic labeling distance metrics. As can be seen, our method outperforms the other rounding procedures in all the 3 cases.

Max Rounding	Complete Rounding		Interval Rounding		Our method		Our method (with finetuning)	
E_{avg}=E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}
803.417	742.596	696.452	744.249	691.179	658.936	653.255	655.547	652.836

Table 2: *Average (E_{avg}) and minimum (E_{min}) energy obtained by the different methods on the test set for the task of semantic segmentation on MSRC data set.*

Max Rounding	Complete Rounding		Interval Rounding		Our method		Our method (with finetuning)	
E_{avg}=E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}	E_{avg}	E_{min}
977.203	1131.940	977.242	1077.385	968.124	949.794	936.316	945.433	935.725

Table 3: *Average (E_{avg}) and minimum (E_{min}) energy obtained by the different methods on the test set for the task of semantic segmentation on MSRC data set when the input fractional solution corresponds to the negative exponentiated min-marginal provided by TRW-S.*

5.2 Semantic Segmentation of Images

Problem. We consider the problem of semantic segmentation of images which can be formulated as a labeling task in which each pixel has to be assigned a label from the set of semantic classes. We formulate this problem as a labeling problem over a grid graph in which each vertex v_a corresponds to a particular pixel and is connected to its four immediate neighbors via edges. The pairwise potential associated with an edge between vertices v_a and v_b is defined as $\theta_{ab}(i,j) = w_{ab}\delta(i \neq j)$. Here, $w_{ab} = \exp(-||f_a - f_b||_2^2/\sigma_c^2)$ is a color based Gaussian weight with f_a being the 3D color vector of the pixel a and σ_c being a scaling factor.

Dataset. We use the MSRC data set [19] for this experiment. It consists of a total of 591 images with ground-truth segmentation. We use the standard data set split of 276 training, 59 validation and 256 test samples, as specified in [19]. We use the texton based features proposed in [19] for unary potentials in our experiments.

Methods. For this task, the encoder and the decoder for our method are modeled as fully convolutional networks, in order to allow for variable input image sizes. The hyper-parameters σ_c and a weighing factor τ for the pairwise potentials is set by using the validation set. Here, we apply rounding to two different sets of fractional solutions: 1) Primal solutions obtained by solving a continuous relaxation [17] of the original primal problem, 2) Primal feasible solutions computed using expected min-marginals from obtained the TRW-S algorithm. Specifically, for each data set sample, we use fractional solutions obtained using the efficient QP solver proposed in [7] and those corresponding to the negative exponentiated min-marginal provided by TRW-S. We compare rounding procedures like max-rounding, complete rounding and interval rounding[6, 12].

Results. Similar to the previous experiment, we report the expected energy E_{avg} and the minimum energy E_{min} on the test set and compare our method with complete and interval rounding. For results shown in the second last column of Table 2 and Table 3, we do not fine-tune our model for the test samples and report results by simply forward passing through the network learned with the training set. However, our model is still able to generalize well to the test samples and outperforms the other rounding procedures for both types of fractional solutions. We also report results for the case when we fine-tune our model for each of the test samples. As can be seen from Table 2 and Table 3, fine-tuning slightly improves the results in both the cases.

6 Discussion

We proposed a novel framework for performing rounding for discrete labeling problems. Our approach views rounding as a method of sampling from a latent variable model and employs deep neural networks for this purpose. We showed that our method can adapt to different problem structures and outperforms hand designed rounding procedures on these tasks. Going ahead, we would like to explore approaches for performing rounding in presence of constraints. Another interesting direction for future research would be to try and marry different ideas developed in context of relaxation methods and rounding procedures with the aim of improving the overall performance.

7 Acknowledgement

This work is partially funded by the EPSRC grants EP/P020658/1 and TU/B/000048. Pritish was supported by a Google Travel Grant for attending this conference.

References

[1] Thalaiyasingam Ajanthan, Alban Desmaison, Rudy Bunel, Mathieu Salzmann, Philip Torr, and M Pawan Kumar. Efficient linear programming for dense CRFs. In *Computer Vision and Pattern Recognition*, 2017.

[2] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1), 2003.

[3] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations*, 2016.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 2013.

[5] Christopher Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[6] Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 2001.

[7] Alban Desmaison, Rudy Bunel, Pushmeet Kohli, Philip Torr, and M Pawan Kumar. Efficient continuous relaxations for dense CRFs. In *European Conference on Computer Vision*, 2016.

[8] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for max cut. *Random Structures & Algorithms*, 20(3), 2002.

[9] Mohammad Hosseini and Su-In Lee. Learning sparse gaussian graphical models with overlapping blocks. In *Advances in Neural Information Processing Systems*, 2016.

[10] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. In *ACM international conference on Web search and data mining*, 2014.

[11] Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[12] Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM*, 49(5), 2002.

[13] Arie Koster, Stan Van Hoesel, and Antoon Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations research letters*, 23(3), 1998.

[14] M Pawan Kumar and Puneet Dokania. Rounding-based moves for semi-metric labeling. *Journal of Machine Learning Research*, 17(91), 2016.

[15] M Pawan Kumar, Philip Torr, and Andrew Zisserman. Solving markov random fields using second order cone programming relaxations. In *Computer Vision and Pattern Recognition*, volume 1, 2006.

[16] Masakazu Muramatsu and Tsunehiro Suzuki. A new second-order cone programming relaxation for max-cut problems. *Journal of the Operations Research Society of Japan*, 46(2), 2003.

[17] Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, 2006.

[18] Tianlin Shi, Jacob Steinhardt, and Percy Liang. Learning where to sample in structured prediction. In *Artificial Intelligence and Statistics*, 2015.

[19] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 2009.

[20] Johan Thapper and Stanislav Živnỳ. The complexity of finite-valued CSPs. *Journal of the ACM*, 63(4), 2016.

[21] Vijay Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[22] Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *ACM symposium on Theory of computing*, 2011.

[23] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 1991.

[24] David Williamson and David Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[25] Cheng Zhang, Babak Shahbaba, and Hongkai Zhao. Hamiltonian monte carlo acceleration using surrogate functions with random bases. *Statistics and Computing*.