Randomized SPENs for Multi-Modal Prediction

David Belanger [1] **Andrew McCallum** [2]

Abstract

Energy-based prediction defines the mapping from an input $\mathbf{x}$ to an output $\mathbf{y}$ implicitly, via minimization of an energy function $E_{\mathbf{x}}(\mathbf{y})$. In certain machine learning tasks, such as information retrieval or recommender systems, we do not predict an individual value for $\mathbf{y}$, but instead a set of diverse predictions. Unfortunately, extending energy-based prediction to this context is non-trivial. One available approach is to first learn the energy function by conditional density estimation of $\mathbf{y}$ given $\mathbf{x}$, and then gather a set of predictions by sampling from the conditional distribution or finding its multiple modes. For structured prediction energy networks (Belanger & McCallum, 2016), density estimation is intractable, however, and approximate approaches require slow, difficult-to-tune MCMC. In response, this paper presents an alternative approach, where we directly learn a randomized predictor end-to-end such that it can be used to yield diverse sets of high-quality predictions.

1. Introduction

Let $\mathbf{x}$ be the input to a prediction problem and $\mathbf{y}$ be the output. In various machine learning applications, we predict multiple values for $\mathbf{y}$. For example, in information retrieval or recommender systems, the user is often presented with a small list of results. This is more effective than providing a single result, since it increases the chance of returning a relevant prediction and the user's experience is not diminished by the process of choosing among a few options. For problems where the distribution of high-quality $\mathbf{y}$ given $\mathbf{x}$ is multi-modal, predicting a set of results is particularly important. For example, the query 'jaguar' may refer to either a car or an animal.

Structured Prediction Energy Networks (SPENs) (Belanger & McCallum, 2016) are a useful energy-based approach to structured prediction. A deep network is used to define an energy function $E_{\mathbf{x}}(\mathbf{y})$ over candidate outputs and prediction is performed by gradient-based energy minimization. In Belanger et al. (2017), the authors present an accurate and user-friendly end-to-end training method. See Sec. 2 for an overview.

One approach to predicting a set of K values using a SPEN is to fit the conditional distribution $\mathbb{P}(\mathbf{y}|\mathbf{x})$ and then sample from it K times. Rather than sampling, we could also use a procedure that targets high-probability $\mathbf{y}$, such as finding the local modes of a multi-modal distribution.

For SPENs, it is natural to define the distribution

$$\mathbb{P}(\mathbf{y}|\mathbf{x}) \propto \exp(-E_{\mathbf{x}}(\mathbf{y})). \tag{1}$$

Since the energy can be a general deep architecture, this is sufficiently flexible to represent sophisticated multi-modal distributions. Unfortunately, exact conditional maximum likelihood estimation for (1) is typically intractable. In addition, MCMC-based approximate learning methods (Sec. 4) are often slow and difficult to tune. Similar difficulties are present when we seek to query this distribution at test time for samples or local modes.

In response, this paper introduces an alternative learning and prediction approach, where we train a randomized predictor end-to-end such that sets of predictions are high-quality (Sec. 5). This adapts the approach of (Guzman-Rivera et al., 2012) for training a fixed ensemble of independent predictors and can also be seen as a non-probabilistic method for fitting the randomized optimum model of Tarlow et al. (2012).

Our randomized method is defined by first uniformly sampling a value for $\mathbf{y}$ and then proceeding with deterministic gradient-based energy minimization that is initialized at this location. This can be seen as a discriminative method for learning energy functions such that optimization with random restarts is effective. Through simple preliminary experiments on a synthetic dataset with multi-modal $\mathbf{y}$, we demonstrate that our method performs better and is substantially easier to tune than an approach that requires density estimation.

[1]Google Brain [2]University of Massachusetts, Amherst. Correspondence to: David Belanger <dbelanger@google.com>.

Proceedings of the ICML 17 Workshop on Deep Structured Prediction, Sydney, Australia, PMLR 70, 2017.

2. Structured Prediction Energy Networks

A SPEN is a particular energy-based model (LeCun et al., 2006) that uses a deep neural network to define an energy function $E_{\mathbf{x}}(\mathbf{y})$ over structured $\mathbf{y}$. Prediction is performed by (approximately) minimizing the energy with respect to $\mathbf{y}$. Of course, Belanger & McCallum (2016) is not the earliest work to define deep energy functions for structured prediction. The principal novelty of Belanger & McCallum (2016) and Belanger et al. (2017) is that the energy function is treated as a black-box that only provides subroutines for forward and back-propagation. This contrasts with other works employing model-specific algorithms that exploit certain factorization structure of the energy.

Given this limited interface to the energy function, it is natural to perform (approximate) energy minimization using gradient descent. Consequently, SPENs are defined for continuous $\mathbf{y}$. For discrete prediction problems, the SPEN is defined on a convex relaxation.

SPENs can be learned end-to-end (Belanger et al., 2017). Here, a first-order energy minimization method is unrolled into a differentiable computation graph, which can then be trained using standard gradient-based learning methods for deep networks. Any hyperparameters of the energy mininization algorithm, such as per-iteration learning rates, can also be learned. Essentially, we train the energy such that gradient-based prediction yields high-quality predictions. A principal advantage of this training method is that it returns not just an energy function, but also an actual energy minimization procedure to be used at test time.

3. Evaluating Set-Valued Predictors

Consider a predictor that takes $\mathbf{x}$ as input and returns a set of predictions $\mathbf{y}_1, \ldots, \mathbf{y}_K$, where each y_k may be a univariate prediction or a structured object. Let $\mathbf{y}^*$ be the ground truth. Let $\Delta(\mathbf{y}, \mathbf{y}^*)$ be the cost function of predicting $\mathbf{y}$. Our experiments in Sec 6.1 employ the Oracle-K cost (Guzman-Rivera et al., 2012), defined as:

$$\Delta_K(\mathbf{y}_1, \ldots, \mathbf{y}_K, \mathbf{y}^*) = \min_k \Delta(\mathbf{y}_k, \mathbf{y}^*). \tag{2}$$

This is low whenever any of the predictions is close to $\mathbf{y}^*$. When $\mathbb{P}(\mathbf{y}|x)$ is multi-modal, the loss will be low only when the set of predictions is covers the high-density regions of $\mathbb{P}(\mathbf{y}|\mathbf{x})$. In many applications, (2) precisely quantifies test-time performance. For example in information retrieval, we may be interested in recall at K.

In future work, it may be interesting to employ loss functions that explicitly encouraging diversity among the predictions (Kulesza & Taskar, 2010; Guzman-Rivera et al., 2014). Our approach could also be used as a way to train the scoring model and sampler used for approximate minimum Bayes risk prediction (Premachandran et al., 2014).

4. Approximate Conditional Likelihood Learning for SPENs

Exact maximum likelihood (MLE) learning of the Gibbs distribution (1) is intractable in general. In response, we can perform stochastic MLE training (Younes, 1989) (SMLE). This is also known as *persistent contrastive divergence* (Tieleman, 2008). Let $\mathbf{x}_i, \mathbf{y}_i$ be a given training example. Let $\mathbf{y}_s$ be a sample from the conditional distribution (1) with $\mathbf{x} = \mathbf{x}_i$. Then, a stochastic gradient of the conditional log-likelihood of $\mathbf{y}_i$ given $\mathbf{x}_i$ is:

$$-\nabla E_{\mathbf{x}_i}(\mathbf{y}_i) + \nabla E_{\mathbf{x}_i}(\mathbf{y}_s), \tag{3}$$

where the gradient is taken with respect to the learned parameters of network defining $E_{\mathbf{x}_i}(\cdot)$. As noted in Sec. 2, we focus on the regime where the energy function is a black box that only provides subroutines for forward and back-propagation. Here, it is natural to sample from (1) using Hamiltonian Monte Carlo (HMC) sampling (Neal et al., 2011), an MCMC scheme that leverages gradients of the distribution's log-density. See (Swersky et al., 2010) for a derivation of SMLE, conditions under which it will converge to the true MLE estimate, and a discussion of its advantages vs. *contrastive divergence* (Hinton, 2002). Finally, Tarlow et al. (2012) present a method for approximate MLE for similar randomized predictors as those in the next section. Crucially, however, our method does not rely on the tractability of exact energy minimization.

5. End-to-End Learning for Randomized Set-Valued Predictors

While conditional density estimation is conceptually attractive, it may not yield good oracle cost (2) in practice. First of all, it does not directly provide a test-time prediction procedure, and thus we will need to separately tune our energy minimization method. Second, it requires accurate MCMC in the inner loop, and this may be very challenging to tune. Here, tuning is particularly difficult because the overall scale of the energy function may change tremendously over the course of learning, and thus the sampling hyperparameters will need to be updated on the fly.

This section proposes an alternative approach, where we directly minimize the Oracle-K cost (2) end-to-end. Given an energy function $E_{\mathbf{x}}(\cdot)$, let $\mathbb{P}_R(\mathbf{y}; E)$ be a distribution over randomized predictions. This may be completely different than the Gibbs distribution (1) induced by the energy. We assume that the cost Δ is differentiable in its first argument.

Furthermore, we assume that $\mathbb{P}_R(\mathbf{y}; E)$ supports the *reparametrization trick* (Kingma & Welling, 2014), where samples can be obtained by evaluating a differentiable, deterministic function $M(x, \epsilon)$ that depends on x and an external source of randomness $\epsilon \sim \mathbb{P}_\epsilon$. The advantage of the

reparametrization trick is that the distribution $\mathbb{P}_\epsilon$ we take an expectation with respect to his not part of our learned model. We have the expected Oracle-K cost:

$$\begin{aligned}&\mathbb{E}_{\mathbf{y}_k \sim \mathbb{P}_R(\mathbf{y}\ E)} \Delta_K(\mathbf{y}_1, \ldots, \mathbf{y}_K, \mathbf{y}^*) \\ &= \mathbb{E}_{\epsilon_1, \ldots, \epsilon_K \sim \mathbb{P}_\epsilon} \Delta(M(\mathbf{x}, \epsilon_1), \ldots, M(\mathbf{x}, \epsilon_1), \mathbf{y}^*) \end{aligned} \quad (4)$$

It is straightforward to compute a stochastic subgradient of the right hand side. We first sample K random values ϵ_k and evaluate the cost $\Delta(M(x, \epsilon_k), \mathbf{y}^*)$ for each. Then, we set ϵ to whichever ϵ_k yielded the lowest cost and peform back-propagation in $\Delta(M(x, \epsilon), \mathbf{y}^*)$.

For a given energy function, there are many ways to define a randomized sampling procedure that supports the reparametrization trick. First, we can unroll a fixed number of steps of HMC for the distribution (1), where we ignore the accept-reject step (Salimans et al., 2015). Similarly, we could unroll sampling by Langevin dynamics (Welling & Teh, 2011). Our experiments employ an even simpler predictor: we randomly sample the location for $\mathbf{y}$ where deterministic gradient-based energy minimization is initialized.

This randomized predictor is attractive because we can use the exact code used for end-to-end learning in Belanger et al. (2017), where we specify a certain number of gradient steps to be taken in advance, and we learn separate per-iteration learning rates. Note that we may learn a randomized prediction procedure that yields diverse predictions even if the associated energy is uni-modal, as an unrolled predictor performing truncated in-complete optimization will terminate at multiple locations.

6. Experiments

6.1. Experimental Setup

We consider synthetic data sampled from a conditional *Gaussian mixture model* (GMM) with 2 components:

$$\mathbb{P}(\mathbf{y}|\mathbf{x}) = \frac{1}{2} N(\mathbf{y}; \boldsymbol{\mu}_1, \sigma^2) + \frac{1}{2} N(\mathbf{y}; \boldsymbol{\mu}_2, \sigma^2), \quad (5)$$

where $N(\mathbf{y}; \boldsymbol{\mu}, \sigma^2)$ is the density for a multi-variate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance $\sigma^2 I$. We employ $\boldsymbol{\mu}_1 = A_1\mathbf{x}$ and $\boldsymbol{\mu}_2 = A_2\mathbf{x}$. We generate data by sampling a 5-dimensional $\mathbf{x}$ from $N(0, 1)$ and then sampling $\mathbf{y}$. We generate our dataset such that for each $\mathbf{x}$ we include 25 $(\mathbf{x}, \mathbf{y})$ pairs, where $\mathbf{y}$ is drawn from $\mathbb{P}(\mathbf{y}|\mathbf{x})$. Therefore, for a given $\mathbf{x}$ we can visualize both the data associated with it and also the conditional energy function $E_\mathbf{x}(\mathbf{y})$.

We choose the data-generating parameters such that samples nearly always occur within $[0, 1]^2$. When performing SMLE learning, we explicitly constrain the HMC iterates to this set. This can be achieved by performing unconstrained HMC in a reparametrized distribution over $\mathbb{R}^2$ with a density defined by multiplying (1) by the determinant of Jacobian of the sigmoid transformation from $\mathbb{R}^2$ to $[0, 1]^2$ (Stan Development Team, 2015, p. 391). For end-to-end learning, we do not constrain the iterates.

For energies fit using SMLE, we form predictions at test time using the same sampling approach used by the randomized unrolled optimizer we train end-to-end. We found that this is superior to sampling from (1). For SMLE, we find test performance is slightly improved by using backtracking line search during test-time energy minimization.

For SMLE, we first burn in HMC using 25 leapfrog trajectories for each $\mathbf{x}$-$\mathbf{y}$ pair. Then, we take 10 additional leapfrog trajectories, where each consists of 5 steps. For each sample, we compute a gradient with respect to the model parameters. Our HMC step size is incremented/decremented on the fly such that proposals are accepted approximately 75% of the time. For end-to-end learning, we unroll for 10 gradient steps and treat our per-step learning rates as trainable parameters. For all methods, the outer loop of optimization with respect to the model parameters is performed using Adam (Kingma & Ba, 2015).

6.2. SPEN Architectures

We employ two different architectures when fitting our GMM data. The first, which we refer to as a GMM energy, hard-codes the functional form of the conditional density of the true data-generating process:

$$E_\mathbf{x}(\mathbf{y}) = \log \sum_{i=1,2} \frac{1}{2} N(\mathbf{y}; \boldsymbol{\mu}_i, \sigma_i^2) \quad (6)$$

$$\boldsymbol{\mu}_i = A_i\mathbf{x} + b_i \quad (7)$$

Here, the trainable parameters are weights A_i, biases b_i, and σ_i. This architecture can be seen as a *mixture density network* (Bishop, 1994). We found that this energy is vulnerable in practice to collapsing modes: if the two mixture components ever coincide, learning will not be able to tease them apart afterwards. We avoid this by initializing the biases b_i to be well-separated.

Second, we employ a generic MLP energy. First, consider an MLP defined just on y.

$$E(\mathbf{y}) = a_3^\top g(A_2 g(A_1\mathbf{y} + b_1) + b_2). \quad (8)$$

Here, g is a coordinate-wise non-linearity.For SMLE we use a ReLU, and when performing end-to-end learning, this must be converted to a SoftPlus (Belanger et al., 2017).

We have found that (8) works well for un-conditional estimation of data drawn from a GMM. To extend this to be a conditional energy function, we can concatenate y and x as inputs to (8). Unfortunately, this does not work particularly well. Instead, it is significantly better to employ a *hypernetwork*, where a subnetwork predicts the weights to be used

in the main network. See Ha et al. (2017) for a thorough overview of the history of architectures and learning methods for hypernetworks. We employ an architecture that is identical to (8), except that the weights and biases in the first layer of the MLP are a function of x.

$$E_x(\mathbf{y}) = a_3^\top g(A_2 g(A_1(\mathbf{x})\mathbf{y} + b_1(\mathbf{x})) + b_2), \quad (9)$$

where $A_1(\mathbf{x})$ is a matrix defined by contracting a learned 3-dimensional tensor with x and $b_1(\mathbf{x})$ is a vector obtained by contracting a learned 2-dimensional tensor with x.

6.3. Results

First, in Fig. 1 we plot energy functions learned by SMLE. The columns correspond to different values of x. For each x, our dataset contains multiple samples for $\mathbf{y}$. These samples are red dots in the figures. The color scale of the figures is such that yellow is high energy and dark blue is low energy. In the top row, we fit a GMM energy function and in the bottom row we fit an MLP energy. In general, we found our learned MLP energies to be uni-modal. They generally capture the envelope of the data, but incorrectly assign low energy to the entire region between the two clusters of data. Consequently, energy minimization yields a prediction that is in the middle of the two modes. We do not plot the predictions made by energy minimization as they reliably end in the local minimia of the energy.

Next, in Fig. 2 we perform the same experiment, but estimate our energy function using end-to-end minimization of the Oracle-8 loss. The black lines are trajectories taken by our learned randomized optimizer. For both the GMM and MLP energies, the trajectories reliably end near one of the data's clusters. In some cases, the learned energy appears to be uni-modal, but the predictions are multi-modal. This is because we learned our optimizer hyperparameters alongside our energy function such that truncated energy minimization yields high-quality sets of predictions.

In Tab. 1 we evaluate our methods in terms of their Oracle-8 performance. We contrast SMLE vs. E2E-8 training and MLP vs. GMM energy functions. Remarkably, our best performance is obtained using an MLP energy with E2E-8 training. We struggle to achieve reasonable performance using an MLP energy with SMLE training, as the learned energy is typically unimodal (see bottom row of Tab. 1). As a result, randomized prediction always predicts a point that is mid-way between the two clusters. In general, we found E2E-8 training substantially easier to tune than HMC-based SMLE. We are unsure why we generally achieve worse performance with the GMM energy vs. the MLP energy. Perhaps the quadratic GMM energy is too steep, and this hinders adequate exploration.

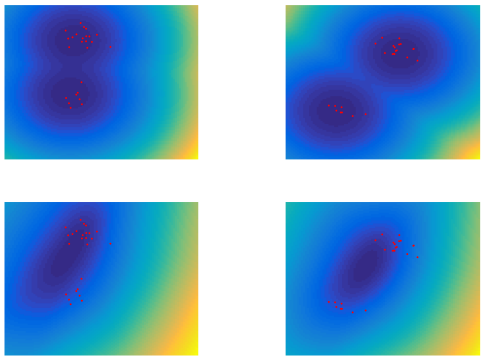

Figure 1. Learned energy functions from stochastic MLE training. Each column uses a different value of x, from which we draw multiple samples of y from a conditional GMM. Top row: GMM energy function. Bottom row: MLP energy function.

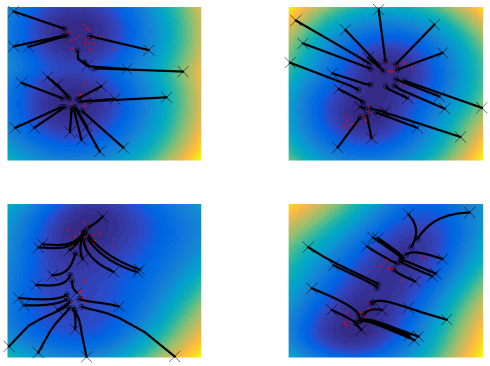

Figure 2. Energy functions learned by end-to-end minimization of Oracle-8 loss. Each column uses a different value of x, from which we draw multiple samples of y from a conditional GMM. Top row: GMM energy function. Bottom row: MLP energy function. Black lines are trajectories from the randomized optimizer.

Training Method	**MLE**	**MLE**	**E2E-8**	**E2E-8**
Architecture	**GMM**	**MLP**	**GMM**	**MLP**
Oracle-8 cost	0.007	0.040	0.004	0.002

Table 1. Oracle-8 performance

7. Conclusion

This paper presents preliminary experiments using a method for training a randomized energy-based predictor end-to-end. The technique is a conceptually-attractive alternative to conditional density estimation that is more straightforward to tune in practice and yields superior performance. In future work, we will apply our techniques to high-dimensional structured prediction problems. The hypernetwork approach of Sec. 6.2 may also be of interest for additional SPEN applications.

References

Belanger, D., Yang, B., and McCallum, A. End-to-end learning for structured prediction energy networks. In *ICML*, 2017.

Belanger, David and McCallum, Andrew. Structured prediction energy networks. In *ICML*, 2016.

Bishop, Christopher M. Mixture density networks. 1994.

Guzman-Rivera, Abner, Batra, Dhruv, and Kohli, Pushmeet. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems*, pp. 1799–1807, 2012.

Guzman-Rivera, Abner, Kohli, Pushmeet, Batra, Dhruv, and Rutenbar, Rob. Efficiently enforcing diversity in multi-output structured prediction. In *Artificial Intelligence and Statistics*, pp. 284–292, 2014.

Ha, David, Dai, Andrew, and Le, Quoc. Hypernetworks. 2017.

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

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *ICLR*, 2015.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *ICLR*, 2014.

Kulesza, Alex and Taskar, Ben. Structured determinantal point processes. In *Advances in neural information processing systems*, pp. 1171–1179, 2010.

LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, M, and Huang, F. A tutorial on energy-based learning. *Predicting Structured Data*, 1, 2006.

Neal, Radford M et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

Premachandran, Vittal, Tarlow, Daniel, and Batra, Dhruv. Empirical minimum bayes risk prediction: How to extract an extra few% performance from vision models with just three more parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1043–1050, 2014.

Salimans, Tim, Kingma, Diederik P, Welling, Max, et al. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Stan Development Team. Stan modeling language users guide and reference manual, Version 2.6.0, 2015. URL `http://mc-stan.org`.

Swersky, Kevin, Chen, Bo, Marlin, Ben, and De Freitas, Nando. A tutorial on stochastic approximation algorithms for training restricted boltzmann machines and deep belief nets. In *Information Theory and Applications Workshop (ITA), 2010*, pp. 1–10. IEEE, 2010.

Tarlow, Daniel, Adams, Ryan, and Zemel, Richard. Randomized optimum models for structured prediction. In *Artificial Intelligence and Statistics*, pp. 1221–1229, 2012.

Tieleman, Tijmen. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071. ACM, 2008.

Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Younes, Laurent. Parametric inference for imperfectly observed gibbsian fields. *Probability theory and related fields*, 82(4):625–645, 1989.