A. Appendix

A.1. Analysis of Convergence Behavior

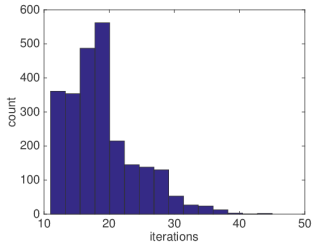


Figure 2. Histogram of #required iters for each example in a large batch to converge. Unnecessary computation is performed on already-converged examples while waiting for the final ones to converge.

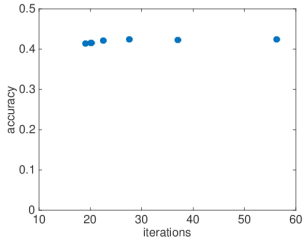


Figure 3. Accuracy vs. # required iters for varying convergence percentage. Optimization on a batch is terminated if X% of the examples in the batch have converged. This provides little decrease in accuracy, while providing an impressive speedup. We hypothesize that the many of the slow-converging examples were ones for which prediction was going to be incorrect anyway, so it is ok to terminate these early.

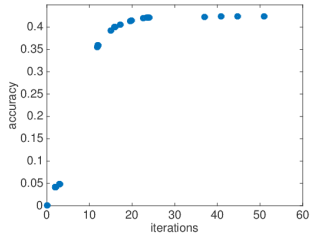
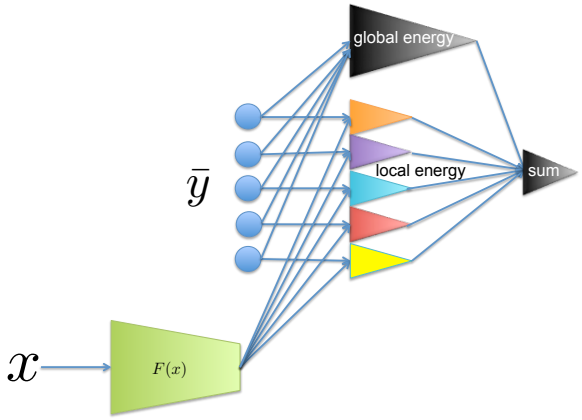


Figure 4. Accuracy vs. # required iters curve for varying convergence tolerance. By using a looseer convergence tolerance, we can sacrifice accuracy for speed.

A.2. SPEN Architecture for Multi-Label Classification



A.3. Deep Mean Field Predictor

Consider a fully-connected pairwise CRF for multi-label prediction. We have:

$$P(y|x) \propto \exp\left(\sum_{i,j} B_{ij}^{(x)}(y_i, y_j) + \sum_i U_i^{(x)}(y_i)\right) \quad (10)$$

Here, the dependence of the pairwise and unary potential functions B and U on x is arbitrary and we leave this dependence implicit going forward. There are 4 possible values for B_{ij}, dependin g on the values of y_i and y_j. Similarly, there are two possible values for U_i. Suppose that y is represented as a vector in $\{0,1\}^L$, then we can re-write (12) as

$$P(y|x) \propto \exp(y^\top A_1 y + (1-y)^\top A_2 y \quad (11)$$
$$+(1-y)^\top A_3(1-y) + C_1^\top y + C_2^\top(1-y)) \quad (12)$$

Here, A_1,A_2, and A_3 are matrices and C_1 and C_2 are vectors. Collecting terms, we obtain a matrix A and vector C such that

$$P(y|x) \propto \exp\left(y^\top A y + C^\top y + \text{constant}\right)$$

ie

$$P(y|x) \propto \exp\left(y^\top A y + C^\top y\right) \quad (13)$$

We seek to perform mean-field variational inference in this CRF. Let $\bar{y}^t \in [0,1]^L$ be the estimate for the marginals of y at timestep t. Define $\bar{y}_{i,0}^t$ to be a vector that is equa l to $\bar{y}$ everywhere but coordinate i, where it is 0 (ie we're conditioning the value of the ith label to be 0). Similarly, define $\bar{y}_{i,1}^t$ The mean-field updates set

$$\bar{y}^{y+1} = \frac{\exp(e_i^1)}{\exp(e_i^1) + \exp(e_i^0)} = \text{Sigmoid}(e_i^1 - e_i^0), \quad (14)$$

Algorithm 1 Vectorized Mean-Field Inference for Fully-Connected Pairwise CRF for Multi-Label Classification

Input: x, m
$A, C \leftarrow \text{GetPotentials}(x)$
Initialize $\bar{y}$ uniformly as $[0.5]^L$
$D \leftarrow \text{diag}(A)$
for $t = 1$ **to** m **do**
  $E \leftarrow A\bar{y} - D + C$
  $\bar{y} \leftarrow \text{Sigmoid}(E)$
end for

where:

$$e_i^1 = (\bar{y}_{i,1}^t)^\top A \bar{y}_{i,1}^t + C^\top \bar{y}_{i,1}^t$$

and

$$e_i^0 = (\bar{y}_{i,0}^t)^\top A \bar{y}_{i,0}^t + C^\top \bar{y}_{i,0}^t.$$

Define $s_i = \sum_{j \neq i} A_{i,j} \bar{y}^t$. Many terms in $e_i^1 - e_i^0$ cancel. We're left with

$$e_i^1 - e_i^0 = s_i + C_i.$$

A vectorized form of the mean-field updates is presented in Algorithm 1.

A.4. Details for Improving Efficiency and Accuracy of SPENs

Various tricks of the trade from the deep learning literature, such as momentum, can be applied to improve the prediction-time optimization performance of our entropic mirror descent approach described in Section 2, which are particularly important because $E_x(\bar{y})$ is generally non-convex.

We perform inference in minibatches in parallel on GPUs.

When 'soft' predictions are useful, it can be useful to augment $E_x(\bar{y})$ with an extra term for the entropy of $\bar{y}$. This can be handled at essentially no computational cost, by simply normalizing the iterates in entropic mirror descent at a certain 'temperature.' This is only done at test time, not in the inner loop of learning.

Typically, backpropagation computes the gradient of output with respect to the input and also computes the gradient of the output with respect to any parameters of the network. For us, however, we only care about gradients with respect to the inputs $\bar{y}$ during inference. Therefore, we can obtain a considerable speedup by avoiding computation of the parameter gradients.

We train the local energy network first, using a local label-wise prediction loss. Then, we clamp the parameters of the local energy network and train the global energy network. Finally, we perform an additional pass of training, where all parameters are updated using a small learning rate.

	#labels	#features	# train	% true labels
Bibtex	159	1836	4880	2.40
Delicious	983	500	12920	19.02
Bookmarks	208	2150	60000	2.03
Yeast	14	103	2417	30.3

Table 4. Properties of the datasets.

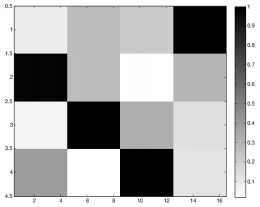


Figure 5. Structure learning on synthetic task using 10% of the data. The measurement matrix still recovers interactions between the labels characteristic of the data generating process

A.5. Hyperparameters

For prediction, both at test time and in the inner loop of learning, we ran gradient descent with momentum = 0.95, a learning rate of 0.1, and no learning rate decay. We terminated prediction when either the relative change in the objective was below a tolerance or the l_∞ change between iterates was below an absolute tolerance.

For training, we used sgd with momentum 0.9 with learning rate and learning rate decay tuned on development data. We use l2 regularization both when pre-training the features and net and during SSVM training, with l2 weights tuned on development data.

We did not tune the sizes of the hidden layers for the feature network. These were set based on intuition and the size of the data, the number of training examples, etc.