Prediction-Constrained Hidden Markov Models for Semi-Supervised Classification

Gabriel Hope [1] **Michael C. Hughes** [2] **Finale Doshi-Velez** [3] **Erik B. Sudderth** [1]

Abstract

We develop a new framework for training hidden Markov models that balances generative and discriminative goals. Our approach requires likelihood-based or Bayesian learning to meet task-specific prediction quality constraints, preventing model misspecification from leading to poor subsequent predictions. When users specify appropriate loss functions to constrain predictions, our approach can enhance semi-supervised learning when labeled sequences are rare and boost accuracy when data has unbalanced labels.Via automatic differentiation we backpropagate gradients through dynamic programming computation of the marginal likelihood, making training feasible without auxiliary bounds or approximations. Our approach is effective for human activity modeling and healthcare intervention forecasting, delivering accuracies competitive with well-tuned neural networks for fully labeled data, and substantially better for partially labeled data. Simultaneously, our learned generative model illuminates the dynamical states driving predictions.

1 Introduction

We develop broadly applicable methods for learning models of data sequences x, some of which are annotated with task-specific labels y. For example, in human activity tasks [29], x might be body motion data captured by an accelerometer, and y might be activity labels (walking, swimming, etc). We seek good predictions of labels y from data x, while simultaneously learning a good model of x itself that is *informed* by task labels y. Strong generative models are valuable because they allow analysts to visualize recovered patterns and query for missing data. Moreover, our approach enables semi-supervised time-series learning from data where only a small subset of data sequences have labels.

[1]UC Irvine [2]Tufts University [3]Harvard University . Correspondence to: Gabriel Hope <hopej@uci.edu>.

Accepted Paper at the *Time Series Workshop at ICML 2021*.

We consider the broad family of *hidden Markov models* (HMMs), for which a wide range of training methods have been previously proposed. Unsupervised learning for HMMs such as the EM algorithm [27] do not use task labels y to inform the learned state structure. They thus may have poor predictive power. In contrast, discriminative methods for training HMMs [5; 32; 12] have been widely used for problems like speech recognition to learn states informed by labels y. These methods have several shortcomings, including restrictions on the loss function used for label prediction, and a failure to allow users to select a task-specific tradeoff between generative and discriminative performance. See App. B for detailed related work discussion.

This paper develops a framework for training *prediction constrained hidden Markov models* (PC-HMMs) that minimize application-motivated loss functions in the prediction of task labels y, while simultaneously learning good quality generative models of the raw sequence data x. We demonstrate that PC-HMM training leads to prediction performance competitive with modern recurrent network architectures on fully-labeled corpora, and excels over these baselines when given semi-supervised corpora where labels are rare. At the same time, PC-HMM training also learns latent states that enable exploratory interpretation, visualization, and simulation.

2 Hidden Markov Models for Prediction

In this section, we review HMMs. We consider a dataset of N sequences x_n, some of which may have labels y_n. Each sequence x_n has T_n timesteps: $x_n = [x_{n1}, x_{n2} \dots x_{n,T_n}]$. At each timestep t we observe a vector $x_{nt} \in \mathbb{R}^D$.

2.1 Hidden Markov Models

Standard unsupervised HMMs [27] assume that the N observed sequences are generated by a common model with K hidden, discrete states. The sequences are generated by drawing a sequence of per-timestep state assignments $z_n = [z_{n1}, z_{n2} \dots z_{nT_n}]$ from a Markov process, and then drawing observed data x_n conditioned on these assignments. Specifically, we generate the sequence z_n as

$$z_{n1} \sim \text{Cat}(\pi_0), \qquad z_{nt} \sim \text{Cat}(\pi_{z_{nt-1}}),$$

where π_0 is the initial state distribution, and $\pi = \{\pi_k\}_{k=0}^K$ denotes the probabilities of transitioning from state j to state k: $\pi_{jk} = p(z_t = k \mid z_{t-1} = j)$. Next the observations x_n

are sampled such that the distribution of x_{nt} depends only on z_{nt}. In this work we consider Gaussian emissions with a state-specific mean and covariance,

$$p(x_{nt} \mid z_{nt} = k, \phi_k = \{\mu_k, \Sigma_k\}) = \mathcal{N}(x_{nt} \mid \mu_k, \Sigma_k)$$

as well as a first-order autoregressive Gaussian emissions:

$$p(x_{nt} \mid x_{nt-1}, z_{nt} = k, \phi_k = \{A_k, \mu_k, \Sigma_k\}) = \mathcal{N}(x_{nt} \mid A_k x_{nt-1} + \mu_k, \Sigma_k).$$

To simplify notation, we assume that there exists an unmodeled observation x_{n0} at time zero.

Given the observed data x_n, we can efficiently compute posterior marginal probabilities, or *beliefs*, for the latent states z_n via the belief propagation or forward-backward algorithm [27]. We denote these probabilities by $b_{tk}(x_n, \pi, \phi) \triangleq p(z_{nt} = k \mid x_n, \pi, \phi)$, and note that these beliefs are a deterministic function of x_n (which will be important for our end-to-end optimization) with computational cost $\mathcal{O}(T_n K^2)$. The probabilities $b_{tk}(x_n, \pi, \phi)$ take into account the full sequence x_n, including future timesteps $x_{nt'}$, $t' > t$. In some applications, predictions must be made only on the data up until time t. These beliefs $b^{\rightarrow}_{tk}(x_n, \pi, \phi) \triangleq p(z_{nt} = k \mid x_{n1}, \ldots, x_{nt}, \pi, \phi)$ are computed by the forward pass of belief propagation.

2.2 Predicting Labels from Beliefs

Now we consider the prediction of labels y given data x. Because they capture uncertainty in the hidden states z_n, the beliefs b_{tk} are succinct (and computationally efficient) summary statistics for the data. We use beliefs as features for the prediction of labels y_n from data x_n in two scenarios: per-sequence classification, where the entire sequence n has a single binary or categorical label y_n, and per-timestep classification, where each timestep has its own event label.

Sequence classification. In the sequence classification scenario, we seek to assign a scalar label y_n to the entire sequence. Below, we provide two classification functions given the beliefs. In the first, we use the *average* amount of time spent in each state as our feature:

$$\bar{b}(x_n, \pi, \phi) \triangleq \frac{1}{T_n} \sum_{t=1}^{T_n} \mathrm{b}_t(x_n, \pi, \phi), \quad (1)$$

$$\hat{y}_n \triangleq \hat{y}(x_n, \pi, \phi, \eta) = f(\eta^T \bar{b}(x_n, \pi, \phi)). \quad (2)$$

where $\mathrm{b}_t(x_n, \pi, \phi) = p(z_{nt} \mid x_n, \pi, \phi)$, η is a vector of regression coefficients, and $f(\cdot)$ is an appropriate link function (e.g., a logistic $f(w) = 1/(1 + e^{-w})$ for binary labels or a softmax for categorical labels).

For some tasks it may be beneficial to use a more flexible, non-linear model to predict labels from belief states. In these cases, we can replace the linear model based on averaged belief states in Eq. (2) with a general function that takes in the sequence of belief states, and outputs a prediction:

$$\hat{y}_n \triangleq \hat{y}(x_n, \pi, \phi, \eta) = f(b_{1:T}(x_n, \pi, \phi); \eta), \quad (3)$$

where η are parameters of differentiable function $f(\cdot; \eta)$. In one of the sequence classification tasks in Sec. 4, we find that a prediction function incorporating a convolutional transformation of the belief sequence, followed by local max-pooling, leads to improved accuracy. The convolutional structure allows for predictions to depend on belief patterns that span several time-steps, while localized pooling preserves high-level temporal structures while removing some sensitivity to the temporal alignment of sequences.

Event detection. In other applications, we seek to densely label the events occurring at each timestep of a sequence, such as the prediction of medical events from hourly observations of patients in a hospital. To predict the label y_{nt} at time t, we use the beliefs b_{tk} at times $t_{w_{st.}} : t_{w_{end}}$ in a window around t as features for a regression or classification model with parameters η:

$$\hat{y}_{nt} \triangleq \hat{y}_t(x_n, \pi, \phi, \eta) = f(\mathrm{b}_{t_{w_{st.}}:t_{w_{end}}}(x_n, \pi, \phi); \eta). \quad (4)$$

Here $f(\cdot; \eta)$ could either be a generalized linear model based on the average state frequencies in the local time window, or a more complicated non-linear model as discussed for sequence classification.

Finally, we note that many prediction tasks are *offline*: the prediction is needed for post-hoc analysis and thus can incorporate information from the full data sequence. When a prediction needs to happen *online*, for example in forecasting applications, we instead use as regression features only the forward-belief representation $b^{\rightarrow}_{tk}$.

3 Learning via Prediction Constraints

We now develop an overall optimization objective and algorithm to estimate parameters π, ϕ, η given (possibly partially) labeled training data. Our goal is to *jointly* learn how to model the sequence data x_n (via estimated transition parameters π and emission parameters ϕ) and how to predict the labels y_n (via estimated regression parameters η), so that predictions may be high quality even when the model is misspecified.

Suppose we observe a set $\mathcal{L}$ of labeled sequences $\{x_n, y_n\}_{n=1}^{N_L}$ together with a set $\mathcal{U}$ of unlabeled sequences $\{x_n\}_{n=1}^{N_U}$. Our goal is to both explain the sequences x_n, by achieving a high $p(x \mid \pi, \phi)$, while also making predictions $\hat{y}_n$ that minimize some task-specific loss (e.g., hinge loss or logistic loss). Our *prediction-constrained* (PC) training objective for both sequence classification scenarios is:

$$\min_{\pi, \phi, \eta} \quad -\sum_{n \in \mathcal{L} \cup \mathcal{U}} \log p(x_n \mid \pi, \phi) \; - \log p(\pi, \phi), \quad (5)$$

$$\text{subject to:} \quad \sum_{n \in \mathcal{L}} \mathrm{loss}\big(y_n, \hat{y}(x_n, \pi, \phi, \eta)\big) \leq \epsilon.$$

Here, $p(\pi, \phi)$ is an optional prior density to regularize parameters. For event detection, the constraint becomes:

$$\sum_{n \in \mathcal{L}} \sum_{t=1}^{T_n} \mathrm{loss}\big(y_{nt}, \hat{y}_t(x_n, \pi, \phi, \eta)\big) \leq \epsilon. \quad (6)$$

Our constraint ensures that any feasible solution will deliver aggregate loss less than ϵ on the labeled training set, where both the loss function and scalar threshold $\epsilon > 0$ can be set

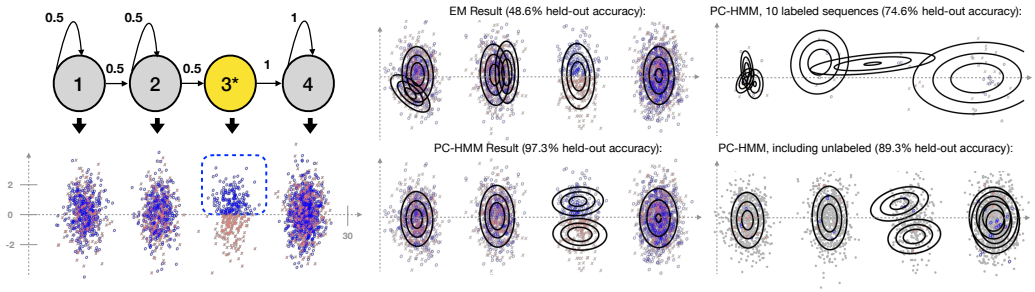

Figure 1: Comparison of our PC-HMM to conventional unsupervised HMM features on a synthetic binary classification task. *Left:* Many state sequences z_n of length 8 are drawn from the illustrated Markov chain. 2D observations x_n are then drawn from evenly-spaced state-specific Gaussians. Each sequence is assigned a positive label y_n only if it has any observation x_{nt} that falls into the marked box. *Center: Fully-labeled task.* We compare a supervised PC-HMM (bottom) to the conventional approach (top) of first training an unsupervised HMM with the EM algorithm, and subsequently predicting labels given average belief state features with a linear classifier. *Right: Semi-supervised task.* Only 10 randomly selected training sequences (out of 350) are labeled (colored observations).

in practice to reflect task-specific needs (e.g., medical interventions must have a certain false positive rate or overall accuracy). Here the loss function may be arbitrary, and need not equal the log-likelihood of discrete labels as assumed by previous work with topic models [10].

We fit the model parameters π, ϕ, η by using the KKT conditions to define an equivalent unconstrained objective that penalizes inaccurate label predictions:

$$\min_{\pi,\phi,\eta} \sum_{n \in \mathcal{L} \cup \mathcal{U}} -\log p(x_n \mid \pi, \phi) - \log p(\pi, \phi) \qquad (7)$$
$$+\lambda \sum_{n \in \mathcal{L}} \text{loss}(y_n, \hat{y}(x_n, \pi, \phi, \eta)) + \lambda \rho ||\eta||_2^2.$$

Here λ is a positive multiplier chosen to ensure that the target prediction constraint is achieved; smaller tolerances ϵ require larger penalty multipliers λ. The quadratic regularization constant ρ, which is scaled by λ to improve interpretability, may be chosen via validation data.

The PC objective of Eq. (7) is differentiable with respect to the model parameters π, ϕ, η and can thus be minimized via standard (stochastic) gradient descent algorithms. Efficient computation of the data log-likelihood $\log p(x_n \mid \pi, \phi)$ is possible by accumulating the log-normalizers of the forward messages underlying the belief propagation algorithm [27]. Thus via a single call to the forward-backward algorithm of Sec. 2.1, we can compute all terms in Eq. (7) and their gradients, with cost *linear* in the number of time steps.

The major hyperparameter required for PC training is the Lagrange multiplier value $\lambda > 0$. There are two noteworthy special cases. Setting $\lambda = 0$ results in unsupervised maximum likelihood training (or MAP training, if priors are used). Setting $\lambda = 1$ and choosing a probabilistic loss $-\log p(y \mid x)$ results in maximum likelihood training of a joint supervised model $p(x, y)$. We refer to this special case as a *supervised* HMM (sHMM). We recommend $\lambda > 1$ to achieve stronger prediction performance. In practice, we select the best of a handful of values on a validation set.

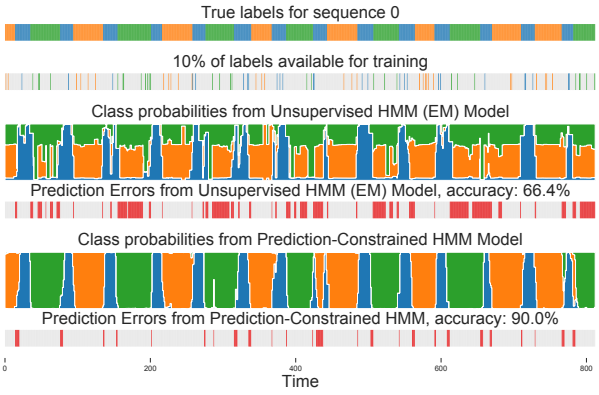

Figure 2: Per-timestep label completion task on the dancing bee dataset. Each method must predict the missing behavior labels for all 6 bee sequences given 10% of the labels selected at random. We visualize the learned states from our PC-HMM and an unsupervised HMM on one representative sequence. Across all 6 sequences, PC-HMM is more accurate (87.2% vs. 61.1%).

Our framework supports any differentiable loss function. For unbalanced data, we recommend reweighting the cross-entropy to equalize contributions from different classes [3]. While balancing does not arise from generative models, it can naturally be incorporated into our PC framework.

4 Experiments

We now assess how well our proposed PC training achieves our two key goals: accurate prediction of labels y given data x and useful generative models of the sequential data x. Details of all protocols can be found in App. A.

Baselines. To establish a competitive baseline for some prediction tasks, we consider modern deep recurrent neural networks [4; 9]. To demonstrate that PC optimization is necessary to learn HMM states useful for prediction, we also compare to a baseline that trains an HMM to maximize

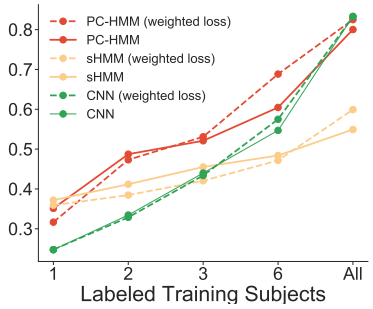

Figure 3: Accuracy on the *activities of daily living* task, evaluated using "leave-one-subject-out" cross-validation over the 30 test subjects. We report the average accuracy on held-out data in predicting the 9 available classes. We plot the performance for the PC-HMM model varying the number of labeled test subjects available in each fold of the training data (each fold uses unique labeled subjects). We compare these results against a CNN for this task [19] that we found strictly outperformed our RNN baseline. Using a weighted loss that balances the influence of all classes, the PC-HMM outperforms the CNN especially when labeled data is rare. Our PC-HMM with $\lambda > 1$ clearly outperforms the sHMM (equivalent to $\lambda = 1$), which maximizes the joint likelihood $p(y, x)$ without extra emphasis on discrimination.

the unsupervised likelihood of the data, via *expectation maximization* (EM) [28]. This HMM baseline first fits π, ϕ given only the data x, then trains a second-stage predictor with parameters η given belief states from the fixed HMM and labels y. We also include our sHMM as a baseline.

Per-Sequence Classification: *Toy Data*. We construct a dataset whose features x are drawn from a known toy HMM, but whose labels y are then assigned so that a correct generative model will not predict well (see illustration in Fig. 1). We see substantial gains for our PC-HMM approach (97.3% accuracy) compared to post-hoc prediction from an unsupervised HMM (48.5%). The learned generative parameters from semi-supervised PC-HMM often differ only slightly from the ideal (unsupervised) generative models.

Per-Timestep Label Completion: *Dancing Bee*. Honey bees communicate the location of food sources to other members of their hive through "waggle dances." Oh et al. [23] tracked 6 different bees performing these dances and identified 3 distinct behaviors: *turn left*, *turn right*, and *waggle* (moving straight while waggling its body). We consider the task of per-timestep prediction of these behaviors using the bee's tracked position and orientation. Fig. 2 shows results of label completion for one sequence. Across all sequences, the PC-HMM achieves 87.2% accuracy at label completion, compared to 61.1% for the unsupervised HMM.

Per-Sequence Classification: *Human Activities*. We recognize human activities using the *activities of daily living* task from the UniMiB SHAR dataset [21]. This dataset consists of 7759 short sequences of 3-axis accelerometer measurements captured from 30 subjects performing 9 everyday activities like *walking*, *sitting*, and *climbing stairs*.

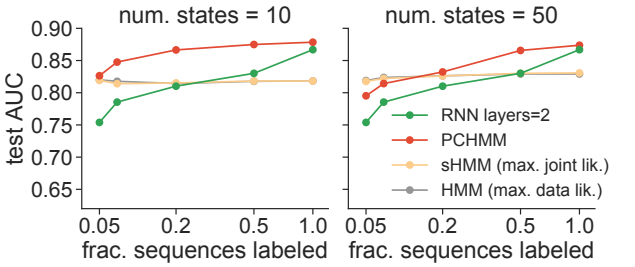

Figure 4: Performance on ICU task: model vital signs x and predict need-for-ventilator y, using 5%, 10%, 20%, 50%, and 100% of labels. AUC scores (prediction quality, higher is better) are shown for a heldout test set, for HMMs with $K = 10$ and $K = 50$. HMM and sHMM models perform similarly, and inferior to PC-HMMs.

Fig. 3 shows that our best model for all 30 subjects using non-linear prediction achieves an accuracy of **83.0%**. This substantially improves the 73.2% previously reported for random forests [21]. A PC-HMM using the simpler linear prediction model of Eq. (2) achieves **72.2%** accuracy using first-order AR emission model, which jumps to **78.6%** using a *second-order* AR model.

Per-Sequence Prediction: *Ventilator Need in the ICU*. We consider an intervention prediction task using 16492 train, 2007 validation, and 4582 test sequences of vital signs and lab results available from the MIMIC-III public dataset of patient stays in an intensive care unit (ICU) [13]. The per-sequence binary outcome y_n is the need for a mechanical ventilator. Each sequence x_n contains 18 hourly measurements of vital signs and lab test results.

Fig. 4 shows that with 100% examples labeled, our 10-state PC-HMM achieves an AUC of 0.878, as good or better than the RNN's 0.867. Greater advantage is seen when only 10% of examples are labeled: the 10-state PC-HMM achieves **0.848** AUC, only slightly worse than fully-labeled and superior to the RNN's 0.785 and plain HMM's 0.817.

5 Conclusion

We have developed a new optimization framework for training hidden Markov models to balance discriminative and generative goals. Across human activity and critical care applications, our PC-HMM delivers superior predictions when labels are scarce and competitive performance even in fully labeled cases. The PC approach is an antidote to model misspecification: the constraint prevents the model from underperforming at the discriminative task, while still allowing learning from unlabeled time series.

Acknowledgements. FDV acknowledges support from NSF CAREER 1750358, and EBS acknowledges support from NSF CAREER 1758028.

References

[1] M. Aczon, D. Ledbetter, L. Ho, A. Gunny, A. Flynn, J. Williams, and R. Wetzel. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *arXiv preprint arXiv:1701.06675*, 2017.

[2] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[3] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann. The balanced accuracy and its posterior distribution. In *International Conference on Pattern Recognition*, 2010.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Empirical Methods in Natural Language Processing*, 2014. URL http://arxiv.org/abs/1406.1078.

[5] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing*, 2002.

[6] B. Dhariyal, T. Le Nguyen, S. Gsponer, and G. Ifrim. An examination of the state-of-the-art for multivariate time series classification. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 243–250. IEEE, 2020.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[8] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. A sticky HDP-HMM with application to speaker diarization. *Annals of Applied Statistics*, 5 (2A):1020–1056, 2011.

[9] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997. URL https://www.bioinf.jku.at/publications/older/2604.pdf.

[10] M. C. Hughes, G. Hope, L. Weiner, T. H. McCoy, R. H. Perlis, E. B. Sudderth, and F. Doshi-Velez. Semi-supervised prediction-constrained topic models. In *Artificial Intelligence and Statistics*, 2018.

[11] T. S. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Neural Information Processing Systems*, 1999.

[12] S. Ji, L. T. Watson, and L. Carin. Semisupervised learning of hidden markov models via a homotopy method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):275–287, 2009.

[13] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

[14] R. Johnson and T. Zhang. Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings. In *International Conference on Machine Learning*, 2016. URL http://arxiv.org/abs/1602.02373.

[15] F. Karim, S. Majumdar, H. Darabi, and S. Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[16] O. Koller, S. Zargaran, and H. Ney. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] H. Kuehne, A. Richard, and J. Gall. A hybrid rnn-hmm approach for weakly supervised temporal action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. doi: 10.1109/TPAMI.2018.2884469.

[18] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, 2001.

[19] F. Li, K. Shirahama, M. Nisar, L. Köping, and M. Grzegorzek. Comparison of feature learning methods for human activity recognition using wearable sensors. *Sensors*, 18(2):679, 2018.

[20] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[21] D. Micucci, M. Mobilio, and P. Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied*

Sciences, 7(10), 2017. ISSN 2076-3417. doi: 10.3390/app7101101. URL http://www.mdpi.com/2076-3417/7/10/1101.

[22] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

[23] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, 2008.

[24] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *arXiv:1804.09170 [cs, stat]*, 2018. URL http://arxiv.org/abs/1804.09170.

[25] W. Ping, Q. Liu, and A. Ihler. Marginal structured svm with hidden variables. In *International Conference on Machine Learning*, pages 190–198, 2014.

[26] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(10), 2007.

[27] L. R. Rabiner and B.-H. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1): 4–16, 1986.

[28] L. R. Rabiner and B.-H. Juang. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1): 4–16, 1986.

[29] J. L. Reyes-Ortiz, A. Ghio, X. Parra, D. Anguita, J. Cabestany, and A. Catala. Human activity and motion disorder recognition: towards smarter interactive cognitive environments. In *ESANN*. Citeseer, 2013.

[30] A. Schmaltz, Y. Kim, A. M. Rush, and S. M. Shieber. Adapting sequence models for sentence correction. *arXiv preprint arXiv:1707.09067*, 2017.

[31] A. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction with latent variables for general graphical models. In *International Conference on Machine Learning*, 2012.

[32] F. Sha and L. K. Saul. Large margin hidden markov models for automatic speech recognition. In *Neural Information Processing Systems*, pages 1249–1256, 2007.

[33] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[34] C. Sutton and A. McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267—-373, 2012.

[35] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing Systems*, pages 25–32, 2004.

[36] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, pages 104–112, 2004.

[37] L. Wei and E. Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–753, 2006.

[38] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] S. W. Yoon, J. Seo, and J. Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, pages 7115–7123. PMLR, 2019.

[40] C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning*, pages 1169–1176. ACM, 2009.

[41] A. Zhang, J. Zhu, and B. Zhang. Max-margin infinite hidden markov models. In *International Conference on Machine Learning*, pages 315–323, 2014.

[42] X. Zhu. Semi-Supervised Learning Literature Survey. Technical Report Technical Report 1530, Department of Computer Science, University of Wisconsin Madison., 2005.

A Experimental Protocol

Here we provide additional experimental details which did not fit into the primary paper.

RNN Baselines. To establish a competitive baseline for some prediction tasks, we consider modern deep recurrent neural networks [4; 9]. We train RNNs via an aggressive randomized grid search over many possible architectures with 2 recurrent layers, varying the number of hidden units in each layer $\{10, 25, 50, 100\}$, the recurrent unit type $\{\text{gru}, \text{lstm}\}$, the activation function, and the number of dense output layers $\{1, 2\}$, batch size, learning rate and L_2 regularization strength on all weight parameters. The RNN models are trained to optimize a *class-balanced* cross-entropy loss, using RMSprop stochastic gradient descent for up to 200 epochs with early stopping triggered whenever validation loss stops improving. For each possible model size (number of hidden units), we select the best of 50 possible hyperparameter configurations according to the validation set area under the ROC curve (AUC).

HMM Baselines. To demonstrate that PC optimization is necessary to learn HMM states useful for prediction, we compare to a baseline that trains an HMM to maximize the unsupervised likelihood of the data, via *expectation maximization* (EM) [28]. This HMM baseline first fits π, ϕ given only the data x, then trains a second-stage predictor with parameters η given belief states from the fixed HMM and labels y. An alternative supervised HMM (sHMM) baseline optimizes the $\lambda = 1$ special case of our PC objective. Across all HMM-based methods, to mitigate sensitivity to local optima we select the best of many independent runs from random initializations.

Transformation to Unconstrained Parameters. Modern software tools for automatic differentiation make it feasible to compute gradients without substantial engineering effort. However, before applying any iterative update algorithm that uses gradients, we need to transform all transition and emission parameters to unconstrained vectors, so that additive gradient updates do not violate constraints. For each transition probability vector π_k, we transform to an unconstrained vector $\tau_k \in \mathbb{R}^K$ via the natural logarithm $\tau_k = [\log \pi_{k1} \ \ldots \ \log \pi_{kK}]$, and use the softmax function to map τ to probabilities π_k that sum to one. For the covariance matrices Σ_k arising in Gaussian likelihoods, we first Cholesky factorize to obtain a lower-triangular matrix L_k such that $\Sigma_k = L_k L_k^T$. We then apply a logarithm to the diagonal entries of L_k, keeping off-diagonal entries unchanged, to obtain an unconstrained parameter.

State Space Selection. For each task, we fit models with a large number of HMM states K, deliberately selecting more than the true number of states if known. Including extra states makes it more likely that our non-convex optimization will return all needed states. If we could afford many more initializations, using the "true" number of states would give similar results.

Prior Specification. We set the starting-state Dirichlet prior to a uniform with $\alpha_{0k} = 10$ (encouraging usage of all states). We choose "sticky" priors over transition probabilities [8] to encourage high-probability self-transitions: $\alpha_{jj} = \alpha + \kappa$ and $\alpha_{jk} = \alpha$ if $j \neq k$; $\alpha = 1$ and $\kappa \approx 100$. In our target tasks, we expect ideal temporal segmentations to switch states infrequently. Selecting the prior on the emission parameters ϕ_k requires some task-dependent choices. For our AR-Gaussian likelihoods, we use a conjugate matrix-normal-inverse-Wishart prior [8] with parameters chosen via grid search to maximize validation set performance.

B Related Work: Discriminative Models for Sequential Data

There is an extensive literature on the discriminative prediction of per-timestep or per-sequence labels y given corresponding observation sequences x. We briefly review two broad categories of competitive models: structured prediction models and neural networks.

Structured prediction methods. Many competitive sequential prediction models are variants of *conditional random fields* (CRFs) [18] or *structural support vector machines* (SSVMs) [36; 35]. These models typically assume the labels y are available for all training sequences x, and are trained to minimize a loss (log-likelihood for CRFs, hinge loss for SSVMs) in the prediction of y given x. Surveys have highlighted applications to natural language [34] and image data [22].

The discriminative CRFs and SSVMs above typically do not employ latent variables: the conditional distribution of the sequence or label y is directly parameterized via features $\phi(x)$ of the observed sequence. However, in many cases it is helpful to introduce a latent sequence z that is never directly observed, but is useful in summarizing aspects of the sequence x relevant to the prediction of y. For example, if y indicates times that a patient's blood pressure drops too low, it may be helpful to have a latent sequence z that tracks statistics related to the blood pressure across a long observation history x. Previous work has developed learning algorithms for CRFs with latent variables [26], with applications including the recognition of gestures y from video sequences x. Structural SVMs with latent variables have also been proposed [40; 7; 31; 25], but those previous applications have focused on models where states z lack sequential or temporal structure. The *max-margin infinite HMM* [41] adapts maximum entropy discrimination [11] to incorporate (via a hinge loss) label prediction accuracy when learning a Bayesian nonparametric HMM.

Neural network models. Recently, excellent prediction accuracy has often been achieved via *recurrent neural networks* (RNNs) for non-linear sequence-to-sequence and sequence-to-label prediction [33]. RNNs are the state-of-the-art for many text processing applications [30; 38] and have been used in a variety of settings, including acuity prediction in the ICU [1], and have performed well on some general timeseries classification benchmarks [15]. While flexible, discriminative neural networks (like CRFs and SSVMs) can only make use of sequences x for which labels y are available; they also cannot be used to generate new sequences x or otherwise understand the structure of x itself. Because our prediction constrained approach retains a generative model of observations x alongside a discriminative model of labels y, we can leverage large collections of unlabeled data, visualize structure in x, and gracefully handle missing values in irregularly sampled time series. Moreover recent work [6] has shown that deep-network based models can often be outperformed by simple statistical-feature based classifiers. [6] provides an overview and comparison of recent deep-learning based timeseries classifiers to other state-of-the-art methods, evaluating on the UEA Multivariate Time Series Classification Archive [2].

Previous HMM-Neural hybrids. Several previous efforts have integrated HMMs and deep neural networks. Kuehne et al. [17] develop a per-frame activity classifier for videos where an RNN produces fine-grained likelihoods which are then fed into an HMM to infer smoothed segmentations over longer time-scales. Related efforts explore cooking videos [20] and sign-language sequences [16] using a similar neural likelihood approach. In contrast, our work applies an HMM to raw data and then feeds beliefs into a learned discriminator (possibly a NN). Our approach allows us to make predictions even when some data x_t is missing, and further performs end-to-end training to optimize all parameters at once to balance generative and discriminative goals, rather than the iterative alignment in Kuehne et al. [17].

Previous semi-supervised methods. Semi-supervised learning methods attempt to improve predictors learned from a small set of labeled examples with a large set of unlabeled examples. Despite decades of work [42], recent surveys highlight how semi-supervised predictors can struggle to outperform well-tuned discriminative methods that use only the smaller labeled dataset [24]. Many existing methods for semi-supervised learning given sequential data x use unlabeled data only for "pretraining". For example, in work specialized to text data Johnson and Zhang [14] pretrain RNN embeddings of words on large unlabeled corpora of sentences before fine-tuning these on a smaller labeled corpus. We emphasize that our approach trains a model simultaneously on labeled and unlabeled data. Other methods [37] have adapted simple time-series classifers, such as the one-nearest-neighbor classifier, to the semi-supervised domain.

The recently introduced Tapnet [39] model allows for semi-supervised learning with a neural network-based model, but shows only modest improvement over a fully-supervised approach when labels are sparse.