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 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.

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 models [29], \( x \) might be body motion data captured by an accelerometer, and \( y \) might be activity labels (walking, swimming, etc). We seek to make good predictions of labels \( y \) from data \( x \), while simultaneously learning a good model of the data \( x \) itself that is informed by task labels \( y \). Strong generative models are valuable because they allow analysts to visualize recovered temporal patterns and query the model for missing data. Moreover, our approach enables semi-supervised time-series learning from data where only a small subset of data sequences have label annotations.

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.

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: \( x_n = [x_{n1}, x_{n2} \ldots x_{n,T_n}] \).

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 \( z_n = [z_{n1}, z_{n2} \ldots z_{nT_n}] \) from a Markov process, and then drawing observed data \( x_{nt} \) conditioned on these assignments. Specifically, we generate the sequence \( z_n \) as

\[
    z_{n1} \sim \text{Cat}(\pi_0), \quad z_{nt} \sim \text{Cat}(\pi_{z_{nt-1}}),
\]

where \( \pi_0 \) is the initial state distribution, and \( \pi = \{ \pi_k \}_{k=0}^K \) denotes the probabilities of transitioning from state \( j \) to state

---

1UC Irvine 2Tufts University 3Harvard University. Correspondence to: Gabriel Hope <hopej@uci.edu>.
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 $b_{ntk}(x_n, \pi, \phi) \triangleq p(z_{nt} = k \mid x_n, \pi, \phi)$, 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 $O(T_n K^2)$. The probabilities $b_{ntk}(x_n, \pi, \phi)$ 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 $b^{\pi}_{ntk}(x_n, \pi, \phi) \triangleq p(z_{nt} = k \mid x_{n1}, \ldots, x_{nt}, \pi, \phi)$ 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 $b_{ntk}$ 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:

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

(1)

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

(2)

where $b_t(x_n, \pi, \phi) = p(z_{nt} \mid x_n, \pi, \phi)$, $\eta$ is a vector of regression coefficients, and $f(\cdot)$ is an appropriate link function (e.g., a logistic function $f(w) = 1/(1 + e^{-w})$ 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:

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

(3)

where $\eta$ are parameters of differentiable function $f(\cdot; \eta)$.

#### 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 $b_{ntk}$ at times $t_{w_{nt}} : t_{w_{nd}}$ in a window around $t$ as features for a regression or classification model with parameters $\eta$:

$$\hat{y}_{nt} \triangleq \hat{y}_t(x_n, \pi, \phi, \eta) = f(b_{t_{w_{nt}} : t_{w_{nd}}}(x_n, \pi, \phi); \eta).$$

(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 $b^{\pi}_{ntk}$.

### 3 Learning via Prediction Constraints

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

Suppose we observe a set $L$ of labeled sequences $\{x_n, y_n\}_{n=1}^{N_L}$ together with a set $U$ 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:

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

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

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

$$\sum_{n \in L} \sum_{t=1}^{T_n} \text{loss}(y_{nt}, \hat{y}_t(x_n, \pi, \phi, \eta)) \leq \epsilon.$$
We fit the model parameters \( \pi, \phi \) by using the KKT conditions to define an equivalent unconstrained objective that penalizes inaccurate label predictions:

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

Here \( \lambda \) is a positive multiplier chosen to ensure that the target prediction constraint is achieved; smaller tolerances \( \epsilon \) require larger penalty multipliers \( \lambda \). The quadratic regularizaton constant \( \rho \), which is scaled by \( \lambda \) to improve interpretability, may be chosen via validation data.

The PC objective of Eq. (7) is differentiable with respect to the model parameters \( \pi, \phi, \eta \) 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.

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 \( x \).
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 the unsupervised likelihood of the data, via expectation maximization (EM) [28]. This HMM baseline first fits \( \pi, \phi \) given only the data \( x \), then trains a second-stage predictor with parameters \( \eta \) 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.

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


Prediction-Constrained Hidden Markov Models for Semi-Supervised Classification


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 \{ gru, 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 \( \pi, \phi \) given only the data \( x \), then trains a second-stage predictor with parameters \( \eta \) 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 \( \pi_k \), we transform to an unconstrained vector \( \tau_k \in \mathbb{R}^K \) via the natural logarithm \( \pi_k = [\log \pi_{k1} \ldots \log \pi_{kK}] \), and use the softmax function to map \( \tau \) to probabilities \( \pi_k \) that sum to one. For the covariance matrices \( \Sigma_k \) 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 \( \phi_k \) 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 \( x \) 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] pre-train 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 classifiers, 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.