

Learning Graphical Models for Hypothesis Testing

SUJAY SANGHAVI, VINCENT Y. F. TAN AND ALAN S. WILLSKY[†]

Stochastic Systems Group, Laboratory for Information and Decision Systems
Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology
Cambridge, MA 02139, USA

[†]{sanghavi, vtan, willsky}@mit.edu



Introduction

- **Motivation:** Can sparse graphical models be learnt better if intended purpose is known?
- **Contribution:** Given training data from 2 distributions, we learn models to be used **specifically** for hypothesis testing.

The Problem

The Hypothesis Testing Problem

$X = (X_1, \dots, X_n)$ is a length- n vector of finite-valued random variables, generated from one of two hypotheses

$$H_0 : X \sim p \quad \text{or} \quad H_1 : X \sim q \quad (1)$$

p, q are not available *a priori*.

Given: Labeled training sets $\mathcal{T}_0, \mathcal{T}_1$, consisting of K samples each that are generated IID according to p, q respectively.

Task: Given a new test sample x , decide between H_0 and H_1 .

Natural Solutions

1. Generate empiricals p_e from \mathcal{T}_0 and q_e from \mathcal{T}_1 and do a Likelihood Ratio Test (LRT) using p_e, q_e .

Problem: When n is large, the number of possible sample vectors is large. For e.g. if each X_i is binary, this number is 2^n . Unless K is of the order 2^n , p_e, q_e will be poor approximations of p, q , and be inefficient at classification.

2. Learn sparse models \hat{p} from \mathcal{T}_0 and \hat{q} from \mathcal{T}_1 . Then do a LRT.

Problem: Each model uses information about only of the two hypotheses.

Our Approach

We learn \hat{p}, \hat{q} each using **both** \mathcal{T}_0 and \mathcal{T}_1 and then do a LRT.

$$\begin{array}{l} \text{declare } H_0 \\ \hat{p}(x) \geq \hat{q}(x) \\ \text{declare } H_1 \end{array} \quad (2)$$

Learning Graphical Models Specifically for Hypothesis Testing

Ideally, we would like to learn \hat{p}, \hat{q} to minimize $\Pr(\text{err})$, given by

$$P(e_0) = P(\text{declare } H_1 | H_0) = \sum_x p(x) \mathbf{1}_{\{\hat{q}(x) \geq \hat{p}(x)\}} \quad (3)$$

and $P(e_0)$. But we do not know p, q so instead we minimize

$$P(\tilde{e}_0) = \sum_x p_e(x) \mathbf{1}_{\{\hat{q}(x) \geq \hat{p}(x)\}} \quad (4)$$

and $P(\tilde{e}_1)$. $P(\tilde{e}_0)$ counts the fraction of samples in \mathcal{T}_0 that are misclassified. Learning \hat{p}, \hat{q} involves

1. Learning the **graph structure** of models \hat{p}, \hat{q} .
2. Learning the **parameters** of the models.

Learning Graph Structure

We first concentrate on learning the graph structures G_0, G_1 of the models \hat{p}, \hat{q} . The **projection** p_{G_0} of p onto G_0 is the closest distribution (in KL-divergence) to p that is Markov on G_0 .

$$D(p \| p_{G_0}) \leq D(p \| p') \quad (5)$$

for all p' that is Markov on G_0 . For structure learning, for any given G_0 assume that \hat{p} is projection of p_e onto G_0 .

For low $P(\tilde{e}_0)$ and $P(\tilde{e}_1)$, we would like $\log(\hat{p}(x)/\hat{q}(x)) > 0$ if $p_e(x) > q_e(x)$ and < 0 otherwise. Thus, **choose G_0 and G_1 to maximize**

$$\sum_x (p_e(x) - q_e(x)) \log \left(\frac{\hat{p}(x)}{\hat{q}(x)} \right) \quad (6)$$

This decomposes into two independent objectives with the first maximization problem being:

$$\max_{G_0} \left\{ \sum_x (p_e(x) - q_e(x)) \log \hat{p}(x) \right\} \equiv \min_{G_0} \left\{ D(p_e \| p_{G_0}) - D(q_e \| p_{G_0}) \right\}. \quad (7)$$

This is similar to $\min_{G_0} D(p \| p_{G_0})$ in the classical objective. So **any** classical learning method may be adapted to this objective. In this work, \hat{p} and \hat{q} are **Markov on trees**. Objective (7) becomes a **Max-Weight Spanning Tree** (MWST) problem with edge weights

$$w_{ij} = \sum_{(x_i, x_j)} (p_e(x_i, x_j) - q_e(x_i, x_j)) \log \left(\frac{p_e(x_i, x_j)}{p_e(x_i)p_e(x_j)} \right). \quad (8)$$

The space and time **complexity** of the tree-finding procedure is the **same** as the Chow-Liu procedure [1].

Learning Model Parameters

LRT can be done with \hat{p}, \hat{q} being the projections of the p_e, q_e onto G_0, G_1 . However, further reductions of $P(\tilde{e}_0)$ and $P(\tilde{e}_1)$ are possible by optimizing the **parameters** of \hat{p} and \hat{q} so that they remain Markov on the same graphs G_0, G_1 . For this, we upper bound

$$P(\tilde{e}_0) \leq \min_{\lambda \geq 0} \sum_x p_e(x) \left(\frac{\hat{q}(x)}{\hat{p}(x)} \right)^\lambda. \quad (9)$$

Consider \hat{p} and \hat{q} to be members of **exponential families**:

$$\hat{p}(x; \theta_{\hat{p}}) = \exp \left[(\theta_{\hat{p}}, \phi_p(x)) - \Phi(\theta_{\hat{p}}) \right], \quad (10)$$

where $\phi_p(x), \phi_q(x)$ are the (fixed) features of G_0, G_1 and determine the family. $\theta_{\hat{p}}, \theta_{\hat{q}}$ are the **exponential parameters** to which we optimize using **convex programming**. $\Phi(\cdot)$ is the log-partition function.

Lemma 1 Let $A_0(\theta_{\hat{p}}, \theta_{\hat{q}}, \lambda) \triangleq \sum_x p_e(x) \left(\frac{\hat{q}(x)}{\hat{p}(x)} \right)^\lambda$, then

1. A_0 is convex in $\theta_{\hat{p}}$ for fixed $(\theta_{\hat{q}}, \lambda)$.
2. A_0 is convex in λ for fixed $(\theta_{\hat{p}}, \theta_{\hat{q}})$.

We use an alternating minimizing procedure parameterized by a discrete set of λ 's (from 0 to 1) to find the optimal $\theta_{\hat{p}}^*, \theta_{\hat{q}}^*$.

For each value of λ , $\theta_{\hat{p}}, \theta_{\hat{q}}$ are initialized by transforming the mean parameters of p_e, q_e to exponential parameters. Finally, $\theta_{\hat{p}}^*, \theta_{\hat{q}}^*$ are chosen as the pair that **minimize the sum**:

$$A(\theta_{\hat{p}}, \theta_{\hat{q}}, \lambda) = A_0(\theta_{\hat{p}}, \theta_{\hat{q}}, \lambda) + A_1(\theta_{\hat{p}}, \theta_{\hat{q}}, 1 - \lambda). \quad (11)$$

Alternating-Minimization Algorithm

```

for  $\lambda \leftarrow 0$  to 1 do
  Evaluate  $\theta_{\hat{p}}^{(\lambda)}, \theta_{\hat{q}}^{(\lambda)}$  as follows ;
  repeat
     $\theta_{\hat{p}} = \text{argmin}_{\theta_{\hat{p}}} A_0(\theta_{\hat{p}}, \theta_{\hat{q}}, \lambda)$  ;
     $\theta_{\hat{q}} = \text{argmin}_{\theta_{\hat{q}}} A_1(\theta_{\hat{p}}, \theta_{\hat{q}}, 1 - \lambda)$  ;
  until convergence ;
endFor
 $\lambda_{\min} = \text{argmin}_{\lambda} [A_0(\theta_{\hat{p}}^{(\lambda)}, \theta_{\hat{q}}^{(\lambda)}, \lambda) + A_1(\theta_{\hat{p}}^{(\lambda)}, \theta_{\hat{q}}^{(\lambda)}, 1 - \lambda)]$  ;
 $\theta_{\hat{p}}^* = \theta_{\hat{p}}^{(\lambda_{\min})}$  ;
 $\theta_{\hat{q}}^* = \theta_{\hat{q}}^{(\lambda_{\min})}$  ;

```

Connection to Error Exponents

The parameter learning procedure above is related to the **error exponent** [2] of the hypothesis test. Suppose we are given M new samples. Let e_0^M be the (conditional) error event that H_1 is declared when the truth was H_0 . The error exponent is:

$$E_0(\hat{p}, \hat{q}) = \lim_{M \rightarrow \infty} -\frac{1}{M} P(e_0^M). \quad (12)$$

Lemma 2 $E_0(\hat{p}, \hat{q}) = \max_{\lambda \geq 0} -\log \left[\sum_x p(x) \left(\frac{\hat{q}(x)}{\hat{p}(x)} \right)^\lambda \right]$.

Note the close similarity between the above expression and that of A_0 in Lemma 1. If we replace p with p_e , then the right hand side is exactly $-\log A_0(\theta_{\hat{p}}, \theta_{\hat{q}}, \lambda)$. Thus **minimizing A_0 is equivalent to maximizing** (an empirical version of) the error exponent.

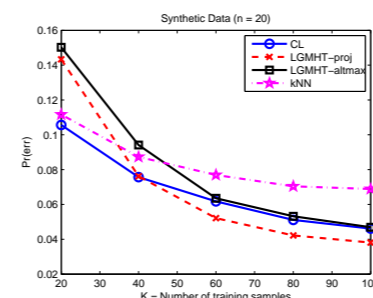
Numerical results

We perform simulations on synthetic examples and on a handwritten digit dataset. Each figure shows 4 curves.

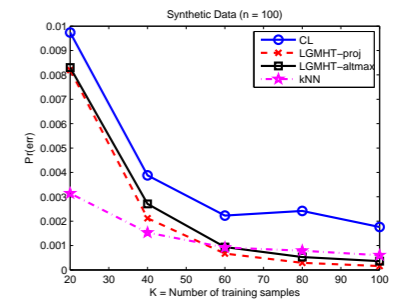
1. **CL:** Learning models according to the Chow-Liu [1] algorithm. (\hat{p} is learnt only from \mathcal{T}_0 and \hat{q} only from \mathcal{T}_1 .)
2. **LGMHT-proj:** \hat{p}, \hat{q} are simply the projections of p_e, q_e onto G_0, G_1 respectively.
3. **LGMHT-altmax:** $\theta_{\hat{p}}, \theta_{\hat{q}}$ are further optimized.
4. **kNN:** Using k -Nearest Neighbours with $k = 11$.

Synthetic examples

X is a **binary** vector of length n . The true distributions p and q are randomly chosen **tree-structured distributions**. Tree structures enable easy generation of training samples $\mathcal{T}_0, \mathcal{T}_1$.

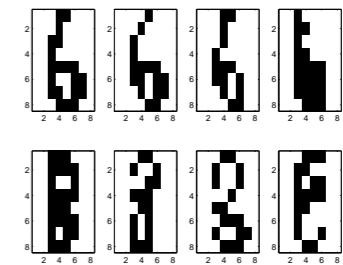


The $\Pr(\text{err})$ is plotted as a function of K for a $n = 20$ node synthetic example.

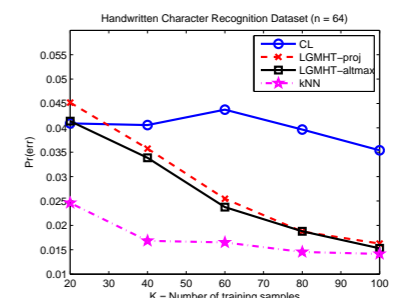


The $\Pr(\text{err})$ is plotted as a function of K for a $n = 100$ node synthetic example.

Handwritten Digits dataset



We used the MNIST handwritten digit dataset and classified the feature vectors corresponding to class labels 6 and 8. The pixel values are quantized to binary levels.



The 8×8 images are concatenated into length $n = 64$ vectors the $\Pr(\text{err})$ is plotted as a function of K .

Conclusion

- In general, learning the reduced-order distributions \hat{p}, \hat{q} from both datasets $\mathcal{T}_0, \mathcal{T}_1$ (instead of learning \hat{p} from \mathcal{T}_0 only as in CL) results in **lower $\Pr(\text{err})$** , especially on the handwritten digits dataset.
- The parameter optimization procedure gives lower $\Pr(\text{err})$ for the handwritten digits dataset but seems to be overfitting the synthetic datasets.
- In general, k NN performs better than CL and LGMHT but k NN requires distances to be calculated for **each** new test sample. LGMHT computes both distributions \hat{p}, \hat{q} offline.
- **Future work:** Examine how the idea of learning graphical models from both datasets can be applied to dimensionality reduction together with classification/clustering.

References

- [1] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–7, May 1968.
- [2] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.