Probabilistic Graphical Models

Raquel Urtasun and Tamir Hazan

TTI Chicago

May 23, 2011

Summary

Previously in class

- Representation of directed and undirected networks
- Inference in these networks
  - Exact inference in trees via message passing
  - Inference via sampling
  - Inference via optimization
- Two tasks of inference:
  - marginals
  - MAP assignment

The rest of this course:

- Parameter learning
- Structure learning (if time)

Today we will refresh your memory about what learning is.

How to acquire a model?

- Possible things to do:
  - Use expert knowledge to determine the graph and the potentials.
  - Use learning to determine the potentials, i.e., **parameter learning**.
  - Use learning to determine the graph, i.e., **structure learning**.
- Manual design is difficult to do and can take a long time for an expert.
- We usually have access to a set of examples from the distribution we wish to model, e.g., a set of images segmented by a labeler.
- We call this task of constructing a model from a set of instances **model learning**.

More rigorous definition

- Lets assume that the domain is governed by some underlying distribution P^*, which is induced by some network model $\mathcal{M}^* = (\mathcal{K}^*, \theta^*)$.
- We are given a dataset $\mathcal{D}$ of M samples from P^*.
- The standard assumption is that the data instances are **independent and identically distributed (IID)**.
- We are also given a family of models $\mathcal{M}$, and our task is to learn some model $\hat{\mathcal{M}}$ in this family that defines a distribution $P_{\hat{\mathcal{M}}}$.
- We can learn model parameters for fix structure, or structure and model parameters.
- We might be interested in returning a single model, a set of hypothesis that are likely, a probability distribution over models, or even a confidence of the model we return.

Goal of learning

- The goal of learning is to return a model $\hat{\mathcal{M}}$ that precisely captures the distribution P^* from which our data was sampled.
- This is in general not achievable because
  - computational reasons.
  - limited data only provides a rough approximation of the true underlying distribution.
- We need to select $\hat{\mathcal{M}}$ to construct the "best" approximation to $\mathcal{M}^*$.
- What is "best"?

What is "best"?

This depends on what we want to do

1. Density estimation: we are interested in marginals.
2. Specific prediction tasks: we are interested in conditional probabilities.
3. Structure or knowledge discovery: we are interested in the model itself.

1) Learning as density estimation

- We want to answer probabilistic inference queries.
- In this setting we can reformulate the learning problem as **density estimation**.
- We want to construct $\hat{\mathcal{M}}$ as "close" as possible to P^*.
- How do we evaluate "closeness"?
- **Relative entropy** is one possibility

$$\mathbf{D}(P^* || \hat{P}) = \mathbf{E}_{\xi \sim P^*} \left[\log \left(\frac{P^*(\xi)}{\hat{P}(\xi)} \right) \right]$$

- $\mathbf{D}(P^* || \hat{P}) = 0$ iff the two distributions are the same.
- It measures the "compression loss" (in bits) of using $\hat{P}$ instead of P^*.
- Problem: In general we do not know P^*.

Expected log-likelihood

- We can simplify this metric for any two distributions over $\mathcal{X}$

$$\mathbf{D}(P^* || \hat{P}) = \mathbf{E}_{\xi \sim P^*} \left[\log \left(\frac{P^*(\xi)}{\hat{P}(\xi)} \right) \right] = \mathbf{H}_P(\mathcal{X}) - \mathbf{E}_{\xi \sim P^*} \left[\log \hat{P}(\xi) \right]$$

- The first term does not depend on $\hat{P}$.
- We can then maximize the **expected log-likelihood**

$$\mathbf{E}_{\xi \sim P^*} \left[\log \hat{P}(\xi) \right]$$

- It assigns high probability to instances sampled from P^*, so to reflect the true distribution.
- We can now compare models, but since we are not computing $\mathbf{H}_P(\mathcal{X})$, we don't know how close we are to the optimum.

Likelihood, Loss and Risk

- We are interested in the (log) **likelihood** of the data given a model, called the **log-loss** $\log P(\mathcal{D}, \mathcal{M})$.
- This is an example of **loss function**.
- A **loss function** $loss(\xi, \mathcal{M})$ measures the loss that a model $\mathcal{M}$ makes on a particular instance ξ.
- When instances are sampled from some distribution P^*, our goal is to find the model that minimizes the **expected loss** or **risk**

$$\mathbf{E}_{\xi \sim P^*} [loss(\xi, \mathcal{M})]$$

- P^* is unknown, but we can approximate the expectation using the empirical average, i.e., **empirical risk**

$$\mathbf{E}_{\mathcal{D}} [loss(\xi, \mathcal{M})] = \frac{1}{|\mathcal{D}|} \sum_{\xi \in \mathcal{D}} loss(\xi, \mathcal{M})$$

- It is intuitive in the case of log loss, where

$$P(\mathcal{D}, \mathcal{M}) = \prod_{m=1}^{M} P(\xi_m, \mathcal{M})$$

2) Specific Prediction Task

- We want to predict a set of variables $\mathbf{Y}$ given some others $\mathbf{X}$, e.g., segmentation.
- We concentrate on predicting $P(\mathbf{Y}|\mathbf{X})$.
- A model trained should be able to produce $\hat{P}(Y|\mathbf{x})$ and the MAP assignment

$$\operatorname*{argmax}_{\mathbf{y}} \hat{P}(\mathbf{y}|\mathbf{x})$$

- An example of loss metric is the **classification error**

$$\mathbf{E}_{(\mathbf{x},\mathbf{y})\sim P^*}\left[\mathbb{1}\{\hat{P}(\mathbf{y}|\mathbf{x})\}\right]$$

  which is the probability over all $(\mathbf{x}, \mathbf{y})$ pairs sampled from P^* that our classifier selects the right label.
- This metric is not well suited for situations with multiple labels. Why?

Better Metrics

- **Hamming loss** counts the number of variables in which the MAP differs from the ground truth label.
- **Conditional log-likelihood** takes into account the confidence in the prediction

$$\mathbf{E}_{(\mathbf{x},\mathbf{y})\sim P^*}\left[\log \hat{P}(\mathbf{y}|\mathbf{x})\right]$$

- Unlike the density estimation, we do not have to predict the distribution over $\mathbf{X}$.
- We negate this expression to get a loss, and compute an empirical estimate by taking the average with respect to $\mathcal{D}$.
- Good choice if we know that we are only going to care about this task.

3) Knowledge Discovery

- We hope that looking at the learned model we can discover something about P^*
  - What are the direct and undirect dependencies.
  - Nature of the dependencies, e.g., positive or negative correlation.
- We may want to learn the structure of the model.
- Simple statistical models (e.g., looking at correlations) can be used.
- But the learned network can have a direct causal interpretation and reveal finer structure, e.g., distinguish between direct and undirect dependencies.
- In this setting we care about discovering the correct model $\mathcal{M}^*$, rather than a different model $\hat{\mathcal{M}}$ that induces a distribution similar to $\mathcal{M}^*$.
- Metric is in terms of the differences between $\mathcal{M}^*$ and $\hat{\mathcal{M}}$.

This is not always achievable

- The true model might not be identifiable
  - e.g., Bayesian network with several I-equivalent structures.
  - In this case the best we can hope is to discover an I-equivalent structure.
  - Problem is worst when the amount of data is limited and the relationships are weak.
- When the number of variables is large relative to the amount of training data: pairs that appear strongly correlated just by chance.
- In knowledge discovery it is very important to asses the confidence in a prediction.
- Taking into account the number of data available and the number of hypothesis.

Learning as optimization

- We define a numerical criteria that we would like to optimize.
- Learning is generally treated as an optimization problem where we have
  - **Hypothesis space**: a set of candidate models.
  - **Objective function**: a criterion for our preference for the models.
- We can formulate learning as finding a high-scoring model within our model class.
- Different approaches choose different hypothesis spaces and different objective functions.

Empirical Risk

- Choose a model $\mathcal{M}$ that optimizes the expectation of a particular loss

$$\mathbf{E}_{\xi \sim P^*} \left[loss(\xi, \mathcal{M}) \right]$$

- We don't know P^* so we use an empirical estimate by defining the empirical distribution

$$\hat{P}_{\mathcal{D}}(A) = \frac{1}{M} \sum_m \mathbb{1}\{\xi_m \in A\}$$

- The prob. of the event A is the fraction of training examples that satisfy A.
- $\hat{P}_{\mathcal{D}}$ is a prob. distribution.
- Let $\xi_1, \xi_2, \cdots$ be a sequence of IID samples from $P^*(\mathcal{X})$, and let $\mathcal{D}_M = \langle \xi_1, \cdots, \xi_M \rangle$, then

$$\lim_{M \to \infty} \hat{P}_{\mathcal{D}_M}(A) = P^*(A)$$

- For sufficiently large training set, $\hat{P}_{\mathcal{D}}$ is close to P^* with high probability.

Empirical Risk and Overfitting

- We can use $\hat{P}_{\mathcal{D}}$ as a proxy.
- Unfortunately a naive implementation will not work, e.g, consider the case of N random binary variables, and M number of training examples, e.g., $N = 100, M = 1000$
- Empirical risk minimization tends to **overfit** the data.
- Problem when using empirical risk as a surrogate for our true risk: Generalization to unseen examples.

Bias-Variance trade off

- If the hypothesis space is very limited, it might not be able to represent P^*, even with unlimited data.
- This type of limitation is called **bias**, as the learning is limited on how close it can approximate the target distribution.
- If we select a highly expressive hypothesis class, we might represent better the data.
- When we have small amount of data, multiple models can fit well, or even better than the true model.
- Moreover, small perturbations on $\mathcal{D}$ will result in very different estimates.
- This limitation is call the **variance**.
- There is an inherent **bias-variance trade off** when selecting the hypothesis class.
- Error due to both things: bias and variance.

How to avoid overfitting?

- Hard constraints, by selecting a less expressive hypothesis class
- Soft preference for simpler models: **Occam Razor**.
- Augment the objective function with **regularization**.

$$objective(\xi, \mathcal{M}) = loss(\xi, \mathcal{M}) + R(\mathcal{M})$$

Evaluating Generalization Performance

Procedure Evaluate (
  $\mathcal{M}$, // parameters to evaluate
  $\mathcal{D}$ // test data set
)

1 $loss \leftarrow 0$
2 **for** $m = 1, \ldots, M$
3 $loss \leftarrow loss + loss(\xi[m] : \mathcal{M})$
4 **return** $\frac{l}{M}$

Procedure Train-And-Test (
  LearnProc, // Learning procedure
  $\mathcal{D}_{train}$, // Training data
  $\mathcal{D}_{test}$, // Test data
)

1 $\mathcal{M} \leftarrow$ LearnProc($\mathcal{D}_{train}$)
2 **return** Evaluate($\mathcal{M}, \mathcal{D}_{test}$)

Procedure Holdout-Test (
  LearnProc, // Learning procedure
  $\mathcal{D}$, // Data set
  p_{test} // Fraction of data for testing
)

1 *Randomly reshuffle instances in* $\mathcal{D}$
2 $M_{train} \leftarrow$ round$(M \cdot (1 - p_{test}))$
3 $\mathcal{D}_{train} \leftarrow \{\xi[1], \ldots, \xi[M_{train}]\}$
4 $\mathcal{D}_{test} \leftarrow \{\xi[M_{train} + 1], \ldots, \xi[M]\}$
5 **return** Train-And-Test(LearnProc, $\mathcal{D}_{train}, \mathcal{D}_{test}$)

Procedure Cross-Validation (
  LearnProc, // Learning procedure
  $\mathcal{D}$, // Data set
  K, // number of cross validation folds
)

1 *Randomly reshuffle instances in* $\mathcal{D}$
2 *Partition* $\mathcal{D}$ *into* K *disjoint datasets* $\mathcal{D}_1, \ldots, \mathcal{D}_K$
3 $loss \leftarrow 0$
4 **for** $k = 1, \ldots, K$
5 $\mathcal{D}_{-k} \leftarrow \mathcal{D} - \mathcal{D}_k$
6 $loss \leftarrow loss +$ Train-And-Test(LearnProc, $\mathcal{D}_{-k}, \mathcal{D}_k$)
7 **return** $\frac{l}{K}$

Goodness of fit

- Cross-validation and hold out test do not allow us to evaluate whether our learned model captures everything that we need in the distribution.
- In statistics, **goodness of fit**.
- After learning the model parameters, we can evaluate if the data behaves as if it was sampled from the this distribution.
- Compare properties of the training data $f(\mathcal{D}_{train})$ and of datasets generated from the model of the same size $f(\mathcal{D})$.
- Many choices of f, e.g., empirical log-loss $\mathbf{E}_{\mathcal{D}}[loss(\xi, \mathcal{M})]$ is the entropy for the model.
- Look at the tales to compute the significance.
- This can be approximate with the variance of the log-loss as a function of M.

PAC bounds I

- We hope that a model that achieves low training loss also achieves low expected loss (risk).
- We cannot guarantee with certainty the quality of our learned model.
- This is because the data is sample stochastically from P^*, and it might be unlucky sample.
- The goal is to prove that the model is approximately correct: for most $\mathcal{D}$, the learning procedure returns a model whose error is low.
- Assume that we have the relative entropy to the true distribution as our loss function.
- Let P^*_M be the distribution over datasets $\mathcal{D}$ of size M sampled IID from P^*.
- Assume that we have a learner L that given $\mathcal{D}$ returns $\mathcal{M}_{L(\mathcal{D})}$.

PAC bounds II

- We want to prove results of the form: with M large enough

$$P_M^*(\{\mathcal{D} : \mathbf{D}(P^* || P_{\mathcal{M}_{L(\mathcal{D})}}) \leq \epsilon\}) \geq 1 - \delta$$

  with $\epsilon > 0$ the approximation parameter and δ our confidence parameter.
- For sufficiently large M, for most datasets $\mathcal{D}$ of size M sampled from P^*, the learning procedure applied to $\mathcal{D}$ will learn a close approximation to P^*.
- This bound can only be obtained if the hypothesis class can be correctly represent P^*.
- Such a setting is called **consistent**.
- In many cases this is not included in the hypothesis class.
- In this case, the best we can hope to get error at most ϵ worse than the lowest error found within our hypothesis space.
- The expected loss beyond the minimal possible error is called the **excess risk**.

Generative vs Discriminative Training I

- We often know in advance that we want to perform a particular task, e.g., predicting $\mathbf{Y}$ from $\mathbf{X}$.
- The training procedure we have described is to compute the joint distribution $P^*(\mathbf{Y}, \mathbf{X})$.
- This is called **generative training** as we train the model to generate all the variables.
- We can also do **discriminative training**, where the goal is to get $\hat{P}(\mathbf{Y}|\mathbf{X})$ as close as possible to $P^*(\mathbf{Y}|\mathbf{X})$.
- The model that is trained generatively can be used for the prediction task.
- However, the discriminatively trained model does not model $P(\mathbf{X})$, and cannot say anything about these variables.
- Discriminative training in BN changes the meaning of the parameters and they no longer correspond to conditional distributions of P^*.
- Discriminative training is done in the context of undirected models, i.e., conditional random fields (CRFs).

Generative vs Discriminative Training II

- Generative models have a higher *bias*, as they make assumptions about the form of $\hat{P}(\mathbf{X})$.
- Discriminative models make assumptions only about $\hat{P}(\mathbf{Y}|\mathbf{X})$.
- The bias reduces the ability of the model to overfit the data, and thus generative models work usually better with small training sets.
- Discriminative models make less assumptions and thus they are less impacted by their incorrect assumptions, and work better with larger training sets.
- Discriminative models can make use of a much reacher feature set. This can result in much higher classification performance, e.g., segmentation.

Learning tasks

The **input** to the learning is

- Some prior knowledge, or constraints about $\hat{\mathcal{M}}$.
- A set $\mathcal{D}$ of IID samples.

The **output** of the learning is a model $\hat{\mathcal{M}}$, which may include the structure, the parameters or both.

The **learning** problem varies along 3 axis

- The output: type of graphical model we are trying to learn, i.e, BN or Markov network.
- The extent of the constraints we are given on $\hat{\mathcal{X}}$.
- The extent to which the data in our training set is fully observed.

Model Constraints

Extent that our input constrains the hypothesis space, i.e., the class of models that we are allow to learn.

- We are given a graph structure, and we have to learn only (some of) the parameters.
- Learn both the parameters and the structure.
- We might not even know the complete set of variables over which P^* is defined, i.e., we might only observe some subset of variables.

The less prior knowledge we are given, the larger the hypothesis space. This complexity depends on

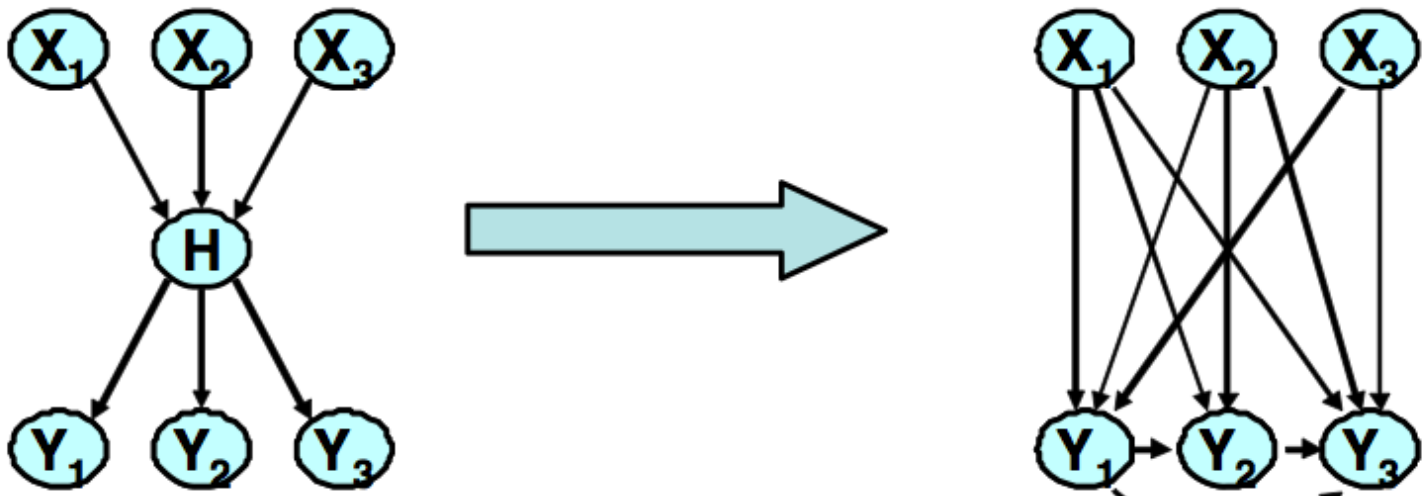
- **statistical**: If we restrict too much it might be unable to represent P^*. If the model is too flexible, we might have models with high score and bad fit to P^*.
- **computational**: the richer the hypothesis class, the more difficult to search.

Data observability

- **Fully observed**: each training instance sees all the variables.
- **Partially observed:** In each training instance, some variables are not observed, e.g., patients and medical tests.
- **Hidden variables:** The value of certain variables is never observed in any of the training instances. This arrives if we don't know all the variables, but also if we simple don't have observations for some.

Missing data

- If the data is missing, we have to hypothesize their value.
- The larger the number of these variables, the less reliable we can hypothesize.
- For the task of knowledge discovery the hidden variables might be very important.



Taxonomy of Learning Tasks in BN

	Complete data	**Missing data**	**Hidden variables**
Known structure	Closed form solution	• Iterated optimization to local maximum, • inference on network multiple times	• Symmetrical solutions • Infinite # of solutions
Unknown structure, known variables	• Combinatorial optimization over structures • score has closed form	• Inference over multiple different network structures • no closed form for score	
Unknown vars	N/A	N/A	• infinite number of possible solutions

Taxonomy of Learning Tasks in Markov Networks

	Complete data	**Missing data**	**Hidden variables**
Known structure	• Convex optimization problem solved optimally via numerical optimization • inference on network multiple times	• Non-convex problem • Iterated optimization to local maximum • inference on network multiple times	• Symmetrical solutions • Infinite # of solutions
Unknown structure, known variables	• Combinatorial and numerical formulations • Can be solved via convex optimization • Inference over multiple different network structures		
Unknown vars	N/A	N/A	• Infinite number of possible solutions