CS 6782: Fall 2010
Probabilistic Graphical Models

Guozhang Wang

December 10, 2010

1 Introduction to Probabilistic Graphical Models

In a probabilistic graphical model, each node represents a random variable, and the links express probabilistic relationships between these variables. The structure that graphical models exploit is the independence properties that exist in many real-world phenomena. The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables. Directed graphs are useful for expressing causal relationships between random variables, whereas undirected graphs are better suited to expressing soft constraints between random variables.

When we apply a graphical model to a problem in machine learning problem, we will typically set some of the random variables to specific values, as *observed variables*. Other unobserved variables would be *latent variables*. The primary role of the latent variables is to allow a complicated distribution over the observed variables to be represented in terms of a model constructed from simpler (typically exponential family) conditional distributions. Generally speaking, with no independence captured in the graph (i.e., the graph is complete), the parameter size would be exponential to the number of latent variables. There are several ways to reduce the independent parameter dimensionality: 1) add independence assumptions, i.e., remove links in the graph, 2) share parameters, also known as *tying* of parameters, 3) use parameterized models for the conditional distributions instead of complete tables of conditional probability values.

1.1 Directed and Undirected Graph

For undirected graphs, the local functions can no longer be chosen as conditional probabilities since they may not be consistent to each other. Further, we can show that the local functions should not be defined on domains of nodes that extend beyond the boundaries of *cliques*. Given that all cliques are subsets of one or more maximal cliques, we can restrict ourselves to

maximal cliques without loss of generality, since an arbitrary function on the maximal cliques already captures all possible dependencies on the nodes.

We can convert the model specified using a directed graph to an undirected graph by "marrying the parents" of all the nodes in the directed graph. This process is known as *moralization*. We saw that in going from a directed to an undirected representation we had to discard some conditional independence properties from the graph. The process of moralization adds the fewest extra links and so retains the maximum number of independence properties. Note that there are some distributions that can be represented as a perfect map using an undirected graph, but not through a directed graph, and vice versa.

1.2 Conditional Independence

Conditional independence properties play an important role in using probabilistic models by simplifying both structure of a model and the computations needed to perform inference and learning under that model. Moreover, conditional independence properties of the joint distribution can be read directly from the graph. The general framework for achieving this is called *d-separation*. In sum in the directed graphs, a tail-to-tail node or a head-to-tail node leaves a path unblocked unless it is observed in which case it blocks the path. By contrast, a head-to-head node blocks a path if it is unobserved, but once the node, and/or at least one of its descendants, is observed the path becomes unblocked. On the other hand in the undirected graphs, the Markov blanket of the node consists of the set of neighboring nodes. We can therefore define the factors in the decomposition of the joint distribution to be functions of the variables in the cliques.

Note that we do not restrict the choice of potential functions to those that have a specific probabilistic interpretation as marginal or conditional distributions. However, one consequence of the generality of the potential functions is that their product will in general not be correctly normalized. We therefore have to introduce an explicit normalization factor. The presence of this normalization constant is one of the major limitations of undirected graphs.

Finally, we can prove that both the factorization and the conditional independence are equal in both directed and undirected graphs.

2 Inference in Graphical Models

We now turn to the problem of inference in graphical models, in which some of the nodes in a graph are clamped to observed values, and we wish to compute the posterior distributions of one or more subsets of other nodes. As we shall see, we can exploit the graphical structure both to find effi-

cient algorithms for inference, and to make the structure of those algorithms transparent.

2.1 The Elimination Algorithm

We will firstly show that the conditional independencies encoded in a graph can be exploited for efficient computation of conditional and marginal probabilities. By taking advantage of the factorization we can safely move the summation over a subset over the nodes in the graph, and thus largely reduce the computation costs. We can introduce some intermediate factors that arise when performing these sums, and computing these factors will eliminate the summation node from further consideration in the computation.

The limiting step in the algorithm is the computation of each potential, therefore the overall computation complexity of the elimination algorithm is exponential in the size of the largest elimination clique. Although the general problem of finding the best elimination ordering of a graph, that is the elimination ordering that achieves the treewidth, turns out to be NP-hard, a number of useful heuristics for finding good elimination orders.

For directed graph, we can firstly moralize it into an undirected graph, and then decide the elimination ordering. Therefore we have seen one example of the important role that undirected graphical models play in designing and analyzing inference algorithms.

A serious limitation of the basic elimination methodology is the restriction to a single query node. Therefore we would like a general procedure for avoiding redundant computation, which will be presented in the next subsection.

2.2 Belief Propagation and Sum-Product Algorithm

The *sum-product algorithm* can efficiently compute all marginals in the special case of trees. From the point of view of graphical model representation and inference there is little significant difference between directed trees and undirected trees. A directed tree and the corresponding undirected tree make exactly the same set of conditional independence assertions.

On the undirected trees, we can define the message going out of each node as the summation of the multiplication of the messages going into that node with the functions on the node and the edges that each message comes along. After all the message for each edge has been calculated, the marginal of a certain node can be defined as a normalized product over all incoming messages. As a result, we can avoid computing the same messages which scales linearly with the size of the tree over and over again, as did in the elimination algorithm.

2.2.1 Factor Graphs

The *factor graph* is an alternative graphical representation of probabilities that is of particular value in the context of the sum-product algorithm. The factor graph approach provides an elegant way to handle various general "tree-like" graphs, including "polytrees", a class of directed graphical models in which nodes have multiple parents.

Factor graphs are closely related to directed and undirected graphical models, but start with factorization rather than with conditional independence. Factor graphs also provide a gateway to factor analysis, probabilistic PCA, and Kalman filters, etc. Sum-Product algorithm only need minor changes to be applied to factor trees. The marginal is then the produce of all the incoming messages arriving at the node from its neighbor factor nodes.

2.2.2 The Max-Sum Algorithm

Two other common tasks other than finding marginals are to find a setting of the variables that has the largest probability and to find the value of that probability. Typically the *argnax* problem is of greater interest than the *max* problem, but the two are closely related. This can be addressed by a closely related algorithm called *max-sum*, which can be viewed as an application of dynamic programming in the context of graphical models. This is no-longer a general inference problem but an prediction problem, when a single best solution is desired.

2.3 Linear Regression and Linear Classification

Linear regression and classification is not specific to graphical models, but their statistical concepts are highly related to them. Therefore they are both elementary building blocks for graphical models.

2.3.1 Linear Regression

In a *regression model* the goal is to model the dependence of a *response* or *output* variable Y on a *covariate* or *input* variable X. We could estimate the joint density $P(X, Y)$ to treat the regression problem, but this usually requires to model the dependencies within X. Therefore it is usually preferable to work with conditional densities $P(Y|X)$. If we view each data point as imposing a linear constraint on the parameters then we can treat parameter estimation in the regression as a (deterministic) constraint satisfaction problem. And still we can show that there is a natural correspondence between the (Euclidean) geometry underly the constraint satisfaction formulation and the statistical assumptions alluded.

If we take the whole set of the data points as a "batch" presentation, in which data are available as a block, then by driving the normal equations or the sum of least square cost functions will give us the same expression of the parameters. In an "online" setting, we can use the gradient of the sum-of-squares function to adjust the values of the parameters. This leads to a *gradient descent* algorithm, which is equivalent to the "LMS" algorithm.

In terms of probabilistic interpretation, we can show that if the data are independent and the conditional density is Gaussian, then maximizing the log likelihood of the conditional density is equal to minimizing the least squares cost. And moreover, the normal equations characterize the parameters to maximize the conditional densities likelihood.

2.3.2 Linear Classification

Classification problems are related to regression problems in that they involve pairs of variables. The distinguish feature of classification is that the response variables range over a finite set, a seemingly minor issue that has important applications. Accordingly, the covariate or input variable X is often called the *feature vector*.

There are two basic approaches to classification problems. The first approach is *generative*, meaning that we model the class-conditional density $P(X|Y)$ and the marginal probability of classes $P(Y)$ so that we can "invert the arrow" to compute posterior probability $P(Y|X)$ using the Bayes' rule. This approach requires modeling the data to get accurate classification results, thus usually need a large and complete set of training data. The second is *discriminative*, where we directly estimate the conditional probability $P(Y|X)$. This approach can optimize directly for the task of interest, but the conditional probability may be harder to interpret in some cases. However, the assumptions underlying linear regression are clearly not met in the classification setting; in particular, the assumption that the variable Y is Gaussian is clearly false. This makes linear regression methods infeasible for linear classification problems, which can be viewed as providing a partitioning of the feature space into regions corresponding to the class labels.

For generative approaches, we can assume that the class-conditional density follows a Gaussian distribution, and thus derive a posterior distribution which is the logistic function of a linear function of the feature vector. Furthermore, we can use *Maximum Likelihood Estimation* to compute the parameters of the class-conditional density's parameters.

Discriminative approaches do not require conditional independence of features since it directly computes the conditional densities of the classes. This is of course the same setting as that of regression, and indeed the methods that we discuss here are closely related to regression. For example, we can use logistic regression methods, which assume Y is a bernoulli random

variable whose probability is a linear-logistic function of X. This will yield the conditional density to also have a logistic form. By using the online gradient descent algorithm, we can update the parameters following the same form as the LMS algorithm. In the batch setting, we can use the *iterative reweighted least squares (IRLS)* algorithm that actually iteratively solves a weighted least squares problem which is reweighted per iteration.

However, there are other choices of class-conditional probabilities that do not yield the logistic-linear form for the posterior probability.

2.4 Exponential Family

Many common densities can be expressed in general *exponential family* form, which has four components: a sufficient statistic $T(x)$ (meaning that there is no information in X regarding certain parameter θ beyond that in $T(X)$), a natural parameter η, a partition function $A(\eta)$, and an underlying function $h(x)$. The exponential family of probability distribution is a large family that includes the multinomial, Gaussian, Bernoulli, Poisson, Gamma, and Dirichlet distributions, etc. We can show that as long as we use exponential family distributions as class-conditional densities for generative model based classification, then the posterior probability will always be the logistic function of a linear function of x; for multiway classification it will be a softmax function of a linear function of x.

Another appearling feature of the exponential family representation is that we can obtain moments of the distribution by taking derivatives of $A(\eta)$. More specifically, we can show the in general the first derivative of $A(\eta)$ is equal to the mean of the sufficient statistics, and the second derivative of $A(\eta)$ is equal to the variance of the sufficient statistics, etc. In addition, we can show that the relationship between η and the mean of the distribution μ can be invertible. This argument implies that a distribution in the exponential family can be *parameterized* not only by η, which is the canonical parameterization, but also by μ, which is the moment parameterization. In fact, many distributions are traditionally parameterized using the moment parameterization.

2.5 Generalized Linear Models

Generalized linear models (GLIMs) are a general category of models that include linear regression and linear classification models as special cases.

A common feature of both the linear regression and discriminative linear classification models is a particular choice of representation for the conditional expectation of Y. Letting μ denote the modeled value of the conditional expectation, we can summarize the structural component of both types of models by: $\mu = f(\theta^T x)$. In the case of linear regression the function the *response function* f is the identity function, and in the case of linear

classification the function f can be logistic function or simple threshold function, etc. In addition, we endow Y with a particular conditional probability distribution, having μ as a parameter. So we need the following two assumptions: $\theta^T x$ is a linear combination of x, the conditional probability of Y based on μ is from the exponential family distributions.

Usually the choice of exponential family is strongly constrained by the type of data, so the main choice to make is the form of the response function. The default canonical response function is given by $f(\eta) = \Psi^{-1}(\eta)$, thus $\eta = \theta^T x$ and the natural parameter is a linear function of the data. And with canonical response functions, the ML estimation has a simple form with sufficient statistic $\Sigma x_n y_n$.

3 Learning in Graphical Models

Given the parameter values and the observed nodes, inference can tell us various conditional and marginal probabilities. However, usually the parameter values are not known in advance, and thus we need to estimate their values using the training data, called *parameter estimation*.

Furthermore, sometimes the graph topology (structure) is not completely known in advance. It is possible to learn both of these from data. However, learning structure is much harder than learning parameters. Also, learning when some of the nodes are hidden, or we have missing data, is much harder than when everything is observed. This gives rise to 4 cases:

- Known structure, full observability: Maximum Likelihood Estimation.
- Known structure, partial observability: EM (or gradient ascent).
- Unknown structure, full observability: Search through model space.
- Unknown structure, partial observability: EM + search through model space.

We start with the simplest case, where the graph structure is known, and the variables are completely observed.

3.1 Completely Observed Graphical Models

When the variables are completely observed, we can estimate the parameters by maximum likelihood estimation. Thus, by "learning" we essentially mean frequentist statistical inference with respect to a parametric distribution.

Let's begin with a simple case, where the graphical model is directed, with discrete random variables and separate parameter set for each local conditional distribution. We can show that the maximization of the log

likelihood of the observed variables can be decomposed into separate maximizations for the local conditional distributions. Furthermore, if each local conditional probability model is an exponential family distribution, the likelihood of each local conditional distribution can be computed from the sufficient statistics, which is the *marginal count* that the certain variable values are observed. If the parameters are shared across local conditional distributions, all associated counts must be considered in estimation.

In domains such as vision and information retrieval, which involve large collections of variables that do not necessarily have a natural ordering, undirected models are often the preferred choice. For undirected models, we can still show that for each clique, the model marginals much be equal to the empirical marginals. For *decomposable graphs*, the maximum likelihood estimation problem decouples, and we can write down maximum likelihood estimates by inspection.

We have seen that sufficiency, exponential family, GLMs, will allow the problem to be addressed in a very general way. Let's now take a look at a general way for maximum likelihood estimation in undirected models when it cannot be decomposable to local probabilities.

3.1.1 Iterative Proportional Fitting

The main idea of *iterative proportional fitting (IPF)* is to "iterate", updating parameters of each potential and hoping that the iterations converge. The update rule is simple, updating one clique at a time, cycling through the cliques until convergence. Though simple, the IPF update rule has two interesting properties: 1) the marginal is equal to the empirical marginal; 2) the normalization factor remains constant across IPF updates.

In addition, IPF is equivalent to "coordinate ascent algorithm" where potentials are coordinates, and also equivalent to minimizing component of KL divergence corresponding to one potential.

3.2 Mixture Models and EM

Now we consider a more different learning problem, where the models have *latent* or *hidden* variables. Latent variables are simply random variables whose values are not specified in the observed data. Latent variables are useful to capture *distinctions* or *components* even if they are not fully understood. Also they allow for a simple, low-dimensional representation of seemingly complex correlations: models with latent variables can often be simpler than models without latent variables. In general, mixture modeling allows problems to be broken into subproblems, and thus can be viewed as a "divide-and-conquer" approach to statistical modeling.

However, estimating the parameters through maximum likelihood will generate a tangled "logs of sums" nonlinear function of the incomplete

marginal log likelihood for the observed variables to maximize.

One approach to maximizing the likelihood is to hand it to a nonlinear optimization algorithm such as conjugate gradient or Newton-Raphson. On the other hand, we can use an alternative approach to maximizing the likelihood known as the Expectation-Maximization (EM) algorithm. Its important virtue in the graphical model setting is that it allows us to take full advantage of the graphical structure underlying the likelihood; in particular, we will be able to exploit the inference algorithms discussed before. The main idea of EM is that to firstly "fill in" latent variables with best guess according to current parameters (the E step), then update the parameters by solving the easier complete-data estimation (i.e., the expected complete log likelihood) problem (the M step), and iterate until convergence. By relating the problem of parameter estimation and the problem of efficient inference, the EM algorithm brings together two of our major themes. And in addition, this is particularly useful when the complete data optimization problem is convex.

3.2.1 K-means Clustering Algorithm

To motivate the EM algorithm, it is useful to step briefly outside of the Gaussian mixture framework to consider an even simpler approach to clustering. The *K-means clustering algorithm* represents each cluster with a single vector, which we refer to as a "cluster mean". The basic idea is to assign data points to clusters by finding the nearest cluster mean and assigning the data point to that cluster. Now we have a chicken-and-egg problem: given the mean of the cluster, we can easily assign the points to clusters; given the points of the cluster, its mean can be easily designed. And the solution to this problem is by iterations of guessing and iterating. In other words, we keep track of the means and the assignments, each being used to update the other. We iterate these two steps until convergence.

Therefore K-means can be viewed as a *coordinate descent* algorithm, which implicitly minimizes a cost function which sums up the within-cluster distances of the clusters.

3.2.2 EM for Gaussian Mixtures

K-means makes a hard assignment of points to clusters. Suppose instead we we make a *soft* (probabilistic) assignment. We can think of the assignment variables as latent and the data points as observed. As before we iteratively update the assignments and the means, now we use *posterior probabilities* of latent variables to make *weighted* assignments (the "Expectation" step) through inference algorithms. And the means (and possibly variances) are the *parameters* which can be updated through learning algorithms such as MLE (the "Maximization" step). Note that the M step based on the ex-

pected sufficient statistics instead of the observed ones. These are all we need for the EM algorithm. Indeed, the EM approach goes hand-in-glove with general graphical model machinery, taking advantage of the conditional independence structure of graphical models in a systematic way.

Generally speaking, the sufficient statistics for EM will be the posterior expected values of the sufficient statistics for a fully observed model. The E step involves computing these expected values, typically by summing across examples using sum-product algorithm, etc. The M step is the same as ML estimation in fully observed models but with the expected sufficient statistics instead of the observed ones. And a common stop criteria is based on the difference between the log likelihoods of two consecutive parameters: if it does not change to much, then break.

3.2.3 EM for Conditional Mixture Models

In fact, both the K-means and Gaussian mixture decomposition are iterative and unsupervised versions of *Linear Discriminant Analysis (LDA)*, which are *unconditional* mixture models, where latent variables are source nodes in the graph. It may be useful to introduce latent variables that are dependent on observed values, giving rise to a *conditional Mixture Models*. Thus, latent variables can be incorporated into regression or classification problems, and EM is a natural choice for parameter estimation in this case. The difference here, is that usually the likelihood in the M step can be decomposed, and therefore maximizing it can be done by some other iterative methods other than ML estimation.

Furthermore, we can show that the EM updates reflect properties that must hold at every "stationary" point of the cost function. Thus if indeed the algorithm converges, it will have found a stationary point. So what is left to be concerned is the convergence of the algorithm. Therefore two forms of log likelihoods: *Expected Complete Log Likelihood (ECLL)* and *Incomplete Log Likelihood (ILL)*. In addition, there is a *quantity* $L(q, \theta)$ which is the upper bound of ECLL (difference is the entropy of the posterior probability) and the lower bound of ILL (difference is the KL-divergence between the posterior and the parameterized average probability). EM can be thought of as coordinate ascent on the function $L(q, \theta)$: E-step fix θ, and trying to maximize the function w.r.t. hidden variable q, so that $L(q, \theta)$ approaches to ILL by eliminating the KL-divergence; M-step fix q and trying to maximize the function w.r.t. parameters θ, so that ECLL and $L(q, \theta)$ are increased with the same portion (entropy unchanged), and ILL will be increased by a larger portion (KL-divergence reappear). In the M-step, if the ECLL cannot be increased anymore, then ILL reaches a (possibly local) maximum. So in conclusion, we can show that EM **will** finally convergence. In fact, EM can also be seen as coordinate minimization of the KL-Divergence of the empirical and model distributions.

Note the the proof of convergence only requires ECLL increases on each iteration, not that it is maximized. Therefore when it is hard to maximize the log likelihood in the M-Step, you can solve it partially. For example, you can partition parameters and optimize groups separately. Also the E-Step can also be approximated by using "viterbi" EM to hard-assign hidden variables instead of soft-assign posterior probabilities.

4 Special Graphical Models

In this section we investigate some specific graphical models which are most popular in many different tasks.

4.1 Markov Chains and HMM

So far we have focused on set of data points that were assumed to be *IID*. Sometimes this assumption is inadequate but assumptions of *partial independence* may be reasonable. A particular important case is sequential data, where data points are generated in a sequence, and the data points can be dependent on its previous data in the sequence but conditionally independent with all others given its previous data. *Markov Chain* is designed to capture such sequential conditional independence, and its key property is that the past and the future are independent given the present, which is also called the *memoryless* property. In particular, the Markov chain is defined by a vector of begin probabilities, and a matrix of transition probabilities. For *First-order* Markov Chains, the matrix has only two dimensions: the data point depends only on its previous data point in the sequence. An N-th order Markov chain has a larger but still limited memory (N steps). It can capture a more complex correlation structure as long as it is *local*.

Simple Markov models are too restrictive for many problems of interest. However, a rich, flexible family of models can be derived by assuming observed variables are generated from latent Markov chains, such as *Hidden Markov Models*.

HMMs are state space models in which the Markov chain has a discrete, finite state space. Thus, they are finite mixture models with correlated mixture components (i.e., in a Markov chain). Like Markov chains, HMMs can be thought of as probabilistic finite state machines, but with probabilistic *emissions* as well as *transitions*.

HMMs are not the richest possible models for sequences, but in many cases they strike an ideal balance between simplicity and expressiveness Both training and inference can be done simply and efficiently. In addition, HMMs are inherently modular they allow complex models to be constructed from simpler pieces

4.1.1 Key Questions and Algorithms

One of the key question is that given the model and a sequence of the observed data, what is the likelihood of the sequence? A related question is to ask what is the most likely path? These two questions are categorized as inference problems. The *Forward Algorithm* is used for the first problem, which is similar to the sum-product algorithm, and the *Viterbi Algorithm* is used for the second problem, which is similar to the max-product algorithm.

Another category of inference problems ask about posterior probabilities. For example, what is the posterior distribution for the latent variable sequence given the observed sequence, and what is the maximum likelihood estimate of all parameters? This can be answered by the combination of the forward algorithm and the *Backward Algorithm*. The backward algorithm is also similar to the sum-product algorithm. In all these algorithms, the propagation of messages in neatly captured by a simple recurrence, because of the highly regular structure of the HMM model.

Other questions arise in an online setting. For example, filtering questions ask for the probability of the current state given the sequence observed so far, predication questions ask for the probability of a future state, and smoothing questions ask for the probability of a previous state. These can also be handled by the forward and backward algorithms.

Now let's talk about the learning of HMMs, which is estimating the transition and emission distribution parameters. If the path of the latent variables Z is observed, maximum likelihood parameter estimation is straightforward. The emission parameters can be estimated separately for each state, as with any finite mixture model. Estimation of the transition parameters then reduces to simple multinominal estimation.

If the latent path is unobserved, however, we need to use EM. And for HMM there is a special-case EM algorithm called the *Baum-Welch* algorithm, where the expected values of the sufficient statistics are needed. For emissions, we need the marginal posterior probabilities to compute the sufficient statistics (we can get from forward-backward algorithms); for transitions, we need *joint pairwise* state configurations are needed (this can be done by slightly modifying the forward-backward algorithm, adding one estimated transition probability and estimated emission probability).

So in a word, we only need to do the forward and backward algorithm once using the estimated parameters in each iteration of EM, and then compute the marginal and pairwise probabilities, and then get the sufficient statistics, which will be used to update the estimation of the parameters.

4.2 Factor analysis

Mixture of Gaussian models and HMMs are those probabilistic models having discrete latent variables. Now we explore some models in which some,

or all, of the latent variables are continuous. The simplest continuous latent variable model assumes Gaussian distributions for both the latent and observed variables and makes use of a linear-Gaussian dependence of the observed variables on the state of the latent variables. This leads to a probabilistic formulation of the well-known technique of principle component analysis (PCA), as well as to a related model called factor analysis.

We will begin with a standard, nonprobabilistic treatment of PCA, then we show factor analysis and how the probabilistic PCA is related to it. In fact, the probabilistic reformulation brings many advantages, such as the use of EM for parameter estimation, principled extensions to mixtures of PCA models, and Bayesian formulations that allow the number of principal components to be determined automatically from the data.

4.2.1 Principle Component Analysis

Principle Component Analysis (PCA) is well-known for dimension reduction in many data analysis tasks. Its goal is to project data points into a lower dimensional space so that the variance of the projected data is maximized. This goal comes from the intuition that we want to keep as much as possible information (measured by the variance) of the original data when we reduce them into lower dimensional space. We can show that maximizing the projected variance is equal to finding the largest eigenvectors as the projection base vectors, since their corresponding eigenvalues are actually the projected variance.

We can further show that maximizing the projected variance is equal to minimizing the projection error. In other words, if the original data space has D dimensions, and we want to reduce to M dimensional space using PCA, then the sum of the first M largest eigenvalues is the projected variance, and the sum of the $D - M$ smallest eigenvalues is the squared distance between the original data points and their approximation.

In implementation, PCA is equivalently defined by the *singular value decomposition* of X^T.

4.2.2 Factor Analysis and Probabilistic PCA

In many cases we can treat the high-dimensional data Y by first sampling a point x in the low-dimensional subspace, then sampling a second point y conditional on the first. Thus the data Y can be described by a continuous mixture model, with lower-dimensional latent variable X and $Y|X$.

Factor analysis arises when the subspace is *linear* to the data space (meaning that the first point will first be mapped using matrix Λ into a subspace of the higher-dimensional data space) and both X and Y—X (conditional on the mapped point on the mapped subspace) are Gaussian. Therefore factor analysis is also a latent variable model where the latent variable

is a continuous random vector.

Because X and $Y|X$ are both Gaussian, the unconditional distribution of Y is also Gaussian, with mean and variance derivable. Therefore that we do not need estimate X but only use it as a dimension reduction method. This is to say, instead of estimating the mean as on the high-dimension data space, we only need to estimate Λ which only has the number of base vectors as the low dimension.

We now show how to express PCA as the maximum likelihood solution of a probabilistic latent variable model. This reformulation of PCA, known as *probabilistic PCA*, can be treated as a special case of factor analysis. In probabilistic PCA, like factor analysis, all of the marginal and conditional distributions are Gaussian. The difference is that in the conditional distributions, the covariance is not arbitrary but diagonal which is controlled by only one parameter. Note that this framework is based on a mapping from latent space to data space, in contrast to the more conventional view of PCA discussed above.

Probabilistic PCA represents a constrained form of the Gaussian distribution in which the number of free parameters can be restricted while still allowing the model to capture the dominant correlations in a data set. Compared to factor analysis, it has a reduced parameterization of covariance matrix to reveal dimensions along which most variance occurs. Compared to standard PCA, its posterior gives a similar "projection" into the lower dimensional space. In other words, this model correctly captures the variance of the data along the principal axes, and approximates the variance in all remaining directions with a single average value, which is the single parameter. Because of that, although factor analysis does not have a closed form for its parameter estimation, probabilistic PCA has a closed-form maximum likelihood estimations. In addition, if we project the data points using the posterior probability of the latent variables, we have the same form as the equations for regularized linear regression and is a consequence of maximizing the likelihood function for a linear Gaussian model.

Although probabilistic PCA has a closed-form MLE, in spaces of high dimensionality, there may be computational advantages in using an iterative EM procedure rather than working directly with the sample covariance matrix. This EM procedure can also be extended to the factor analysis model, for which there is no closed-form solution. Finally, it allows missing data to be handled in a principled way.

We could employ cross-validation to determine the value of dimensionality by selecting the largest log likelihood on a validation data set. Such an approach, however, can become computationally costly. Remember that a probabilistic view of PCA allows for a fully Bayesian version, with priors on model parameters. In this way we can introduce a hyperparameter for each of the dimensions, and estimate its posterior values from likelihoods.

4.2.3 Kalman Filtering

As HMM can be treated as correlated versions of the finite mixture models, Kalman filtering model can be treated as correlated versions of the factor analysis models. Kalman filtering model are widely used in many noise reduction problems such as radar tracking, robotic control, computer vision, etc. Like HMM, it has a transition probability and emission probability. However, both of them are Gaussian, as the transition probability in HMM is discrete. In addition the prior for the initial state is also Gaussian. So the joint distribution for all variables and all marginals and conditionals are also Gaussian.

Two online inference problems are are particular interest of Kalman filtering model: 1) Filtering, where we want to estimate probability of latent variable x_t based on observed $y_0, ..., y_t$ as $p(x_t|y_0, ..., y_t)$; 2) Smoothing, there we want to estimate probability of of latent variable x_t based on observed $y_0, ..., y_T$ where $T > t$ as $p(x_t|y_0, ..., y_t, ..., y_T)$.

For the filtering problem, as with the forward algorithm, we look for a recursion linking between $p(x_t|y_0, ..., y_t)$ and $p(x_{t+1}|y_0, ..., y_{t+1})$. It turns out to be useful to decompose the recursive update into two parts: the time update from $p(x_t|y_0, ..., y_t)$ to $p(x_{t+1}|y_0, ..., y_t)$, and the measurement update from $p(x_{t+1}|y_0, ..., y_t)$ to $p(x_{t+1}|y_0, ..., y_{t+1})$. Since all the probabilities are Gaussian, the time update is straightforward: we only need to update the mean and covariance. For measurement update, we need to firstly compute the joint distribution of x_{t+1} and y_{t+1}, and the decompose the Gaussian to get the conditional distribution. In fact, the time update does not change the mean, but only results in "diffusion" to $t + 1$ with increased variance, while the measurement update "corrects" mean and reduces variance based on observed value. And the full update for the mean has an "error correction" interpretation, which is reminiscent of the LMS algorithm.

For the smoothing problem, the solution is similar to forward/backward algorithm. There are mainly two approaches: the RTS algorithm, and the two-pass algorithm. For the RTS algorithm, the result will be a recurrence linking $p(x_t|x_{t+1}, y_0, ..., y_T)$ and $p(x_{t+1}|x_{t+2}, y_0, ..., y_T)$, which can be iterated from x_T back to x_t using the conditional independence structure of the Kalman filtering model. More specifically, we can make use of the structure that the conditional probability of $p(x_t|x_{t+1}, y_0, ..., y_T)$ is equal to $p(x_t|x_{t+1}, y_0, ..., y_t)$, which can be derived from the joint probability $p(x_t x_{t+1}|y_0, ..., y_t)$.

The smoothing problem is highly related to the parameter estimation problem of Kalman filtering which can be solved by EM. Like factor analysis models, the form of EM can break to individual linear regression models. And also as in FA, efficient inference depends strongly on Gaussian assumptions; sampling or approximate inference is needed under other assumptions.

4.3 Conditional Random Fields

4.3.1 Directed v.s. Undirected Models

The HMM model is a generative model, where the full joint distribution can be factorized into transmission and emission probabilities. Another note is that HMM is also a directed model, where the conditional distribution of the observed variables is modeled by the emission probabilities. There is, however, a problem called the *Label Bias* with directed models due to local normalization. Since each "factor" of a directed model must be a conditional probability distribution, its values must sum up to one. Therefore, HMM can decide *where* to pass their probability mass but not *how much in total*: the transitions leaving a given state compete only against each other leaving that state locally, rather than against all other transitions in the model. This cause a bias toward states with fewer outgoing transitions.

An alternative solution to this problem is to instead use an undirected model. For example, Markov Random Fields (MRF) is such an undirected model, where the sequential MRF share the same structure as HMM but instead modeling the factors as probabilities, they just use a factor function for each factor. The functions are also usually called *features*.

4.3.2 Generative v.s. Discriminative Models

On the other hand, recall there are two major strategies in building (graphical) models for machine learning problems. The generative approach models full probability distributions for the classes and data, while the classes are inferred by the Bayes rule. HMM is actually a generative model which requires modeling the data. The discriminative approach directly models the conditional probability distribution of the classes given the data. This approach will be optimized directly for the task of interest. An discriminative version of the HMM is the Maximum Entropy Markov Model (MEMM), where the emission arrows are "reversed". As a result, we can directly model the conditional probability distribution of the class given the data. Such modification from HMM to MEMM can also be observed from Naive Bayes to Logistic Regression, etc. However, like HMM, MEMM is also a directed model that suffers from the above label bias problems.

4.3.3 Undirected and Discriminative Come Together

Conditional Random Fields (CRF) model extends HMM in two ways: first it is a discriminative model like MEMM, relaxing the dependency assumptions between observations and labels; Second it is an undirected model like MRF, whose conditional distribution is globally based on the whole observation sequence (thus no state bias problem). Here we need to clarify that only Linear-Chain CRF is related to HMM, but not general CRFs. General CRFs

are undirected graphical models in which a set of variables Y is conditioned on another set of (observed) variables X. As usual, X and Y correspond to nodes in an undirected graph, which defines a factorized joint probability In this case, however, X consists of input variables or covariates, and the conditional distribution $p(Y|X)$ is of interest. Usually we factorize $p(Y|X)$ according to its factors $\{\psi_C\}$, with each factor modeled as a *feature function* $\psi_C(y_C, x)$. Note that here x can be all the observed variables/nodes in the model, and therefore as we said before, no conditional independence is assumed in CRF. Typically each feature function picks out particular "features" of X.

For the inference tasks, despite the richness of the model, the Viterbi, Forward/Backward algorithm are essentially unchanged for Linear-Chain CRFs. More generally, as long as Y is defined by a polytree, a slighted adapted sum/max-product algorithm can be used; For more general graphs, the junction tree algorithm, approximate methods, or sampling can be used.

For the learning tasks, parameter estimation is accomplished by conditional maximum likelihood. However, as other undirected models, it is complicated by the normalization tern $Z(X)$ in the conditional likelihood. Therefore maximization is typically accomplished by a quasi-Newton or conjugate gradients methods, or iterative scaling.

5 Approximate Inference

So far we have focused on models for which exact inference is possible. In general, however, this will not be true. For most probabilistic models of practical interest, exact inference is intractable, e.g., for models with non-Gaussian continuous distributions or large clique sizes. There are two main options available in such cases: 1) inference algorithms based on deterministic approximations, including methods such as variational inference and expectation propagation; 2) inference methods based on numerical sampling.

5.1 Sampling

For most situations the posterior distributions of interest is required primarily for the purpose of evaluating expectations. And such expectations can be estimated by N samples if N is sufficiently large. Note that the samples need not be independent, but if not then N must be larger.

5.1.1 Basic Sampling Methods

There are several simple sampling methods for complex distributions: Inversion method, Transformation methods, Rejection method, Adaptive Rejection method, Importance method, etc. However, some of these methods (inverse, transformation) can only be applied to standard distributions,

while others (rejection, importance) suffer from high dimensionality of data. This class of sampling methods is by firstly get random, independent samples from approximation distribution and then adjust to the target distribution. Therefore since a high dimensional distribution is often concentrated in a small region of the state space known as its typical set (for example, Gaussian) which dominate the value of the expected function value, if the approximation distribution is not very close to the target sampling, it will take exponentially long time to get enough samples from this typical set regions as the dimension increases.

5.1.2 Markov Chain Monte Carlo

The reason that this class of sampling fails to overcome this curse of dimensionality is that they use total independent random sampling, which is analogous to random walks. Since random walk is very slowly in exploring the state space, it takes a long time to get to the typical set region of a highly concentrated distribution. Therefore we turn to a very general and powerful framework called Markov chain Monte Carlo (MCMC), which eliminates this random walk behavior, allows sampling from a large class of distributions, and scales well with the dimensionality of the sample space.

The key idea of MCMC is that the distribution of the current sample is conditioned on the previous sample, controlled by the *proposal distribution*. As a result, it "guides" the semi-random walk in the space. And assuming the density of the sample can be easily evaluated within a multiplicative normalizing constant (which we do not know), we will decide to reject the current sample or not based on the density ratio between the current sample and previous sample. However, instead of discarding the current sample if rejected in the rejection sampling, we will use the previous sample as the current sample, leading to multiple copies of samples. As mentioned before, the samples from MCMC are no longer independent and thus we need more samples to converge.

So the question is, how can we set the transition probabilities such that the stationary distribution is the posterior we want, without knowing what the posterior is? To prove a MCMC will converge with certain proposal density, we need the target distribution an *invariant/stationary distribution* of the chains, and the chain itself be *ergoidc*. The first property ensures you that once you enters this distribution, you will never "jump" out of it again. A sufficient (but not necessary) condition for this is to choose the proposal transition probabilities to satisfy the *reversibility/detailed balance* property. The example *Metropolis-Hastings* sampling method ensures this with respect to the posterior simply be evaluating ratios of densities.

Note that a given Markov chain may have more than one invariant distribution. And to guarantee that there is only one such distribution (which is of course the target distribution), we need the second property that the

chain is ergodic. That is, the chain must be 1) *irreducible*: from any state to any other state the transition probability is positive; 2) *aperiodic*: must not cycle through states deterministically; 3) non-transient: must always be able to return to a state after visiting it. Among them, irreducibility is typically the critical property. If ergodicity is true, then the only invariant distribution is given by the eigenvector corresponding to the largest eigenvalue of the transition matrix.

5.2 Variational Inference

As discussed before, intractable inference problems can also be approached with deterministic approximation schemes besides the stochastic techniques such as MCMC. Stochastic techniques generally have the property that given infinite computational resource, they can generate exact results, and the approximation arises from the use of a finite amount of processor time. Deterministic methods, on the other hand, are based on analytical approximations to the posterior distribution, for example by assuming that it factorizes in a particular way or that it has a specific parametric form. As such, they can never generate exact results. These approaches include methods based on the *Laplace approximation*, *variational inference*, and *Expectation propagation*, etc.

Variational methods are named for the *calculus of variations*, which is a kind of "meta-calculus" concerned with *functionals* (which take functions as arguments) rather than *functions* (which take numbers). Many problems can be expressed in terms of an optimization problem in which the quantity being optimized is a functional. Functionals naturally lend themselves to finding approximate solutions. This is done by restricting the range of functions over which the optimization is performed. In the case of applications to probabilistic inference, the restriction may for example take the form of factorization assumptions. And this is the method called variational inference, which is used to find a tractable (factorized) distribution that is as close as possible to the desired intractable distribution, in terms of the KL divergence.

One important application of variational inference is Variational EM. In standard EM, in the expectation step, we need to compute the expected posterior to get the sufficient statistics from the parameter values. This computation, in many cases, could be intractable for many indirected models (e.g., Markov Random Field). Therefore, we seek an approximate distribution that minimize the KL divergence.

Basically, the variational inference is used when exact inference is intractable, by approximately deriving the dependence of the latent variables, and iteratively fixing other variables and updating the left one using the dependencies.

5.2.1 Loopy Belief Propagation

Remember that the sum-product algorithm can be described in terms of message passing: node j sends a message to node i when it has received messages from all other neighbors. *Loopy Belief Propagation (LBP)* is the heuristic procedure of performing message passing on a graph with cycles where nodes are initialized with starting "beliefs". The algorithm is not guaranteed to converge but often seems to perform remarkably well in practice. In addition, when LBP does converge, it effectively acts as a form of variational inference, using the Bethe free energy.

Expectation Propagation unifies and generalizes two techniques: assumed-density filtering, an extension of the Kalman filter, and loopy belief propagation, an extension of belief propagation in Bayesian networks. The unification shows how both of these algorithms can be viewed as approximating the true posterior distribution with a simpler distribution, which is close in the sense of KL-divergence.

5.3 Junction Tree Algorithms

For most inference algorithms, the Markov properties of the underlying graphical model provide the formal machinery to justify the dependencies. The *Junction Tree Algorithm* makes systematic use of the Markov properties of graphical models. All of the examples that we have treated until now emerge as special cases. The general idea is to use the Markov properties of graphical models to find ways to decompose a general probabilistic calculation into a linked set of local computations. The junction tree makes explicit the important relationship between graph-theoretic locality and efficient probabilistic inference. After the algorithm runs, the potential of each clique in the graph will be equal to the unnormalized version of the desired conditional probability, where the normalization constant is *locally* obtained by summing or integrating the desired latent variables inside the clique.

- **Clique Tree.** A clique tree is a singly-connected graph whose nodes represent members of the predefined clique set C. Note that although C can be defined arbitrarily, we generally require these cliques to be maximal, so that no member of C is a subset of another member. We can initialize the clique potentials so that we obtain a representation of the joint or conditional probability as the product of the potentials. In addition separator nodes are introduced as intersections of connected cliques.

- **Introduce Evidence**. Evidence can be introduced by slicing potentials, and normalize locally.

- **Message-Passing Algorithm.** In the two-pass algorithm which follows the message-passing protocol we can guarantee 1) local consis-

tency, 2) invariant joint distribution (supported by introducing the separator nodes). Note that although the algorithm looks like approximate inferencing, it is still exponential in clique sizes and generates exact results (unless the graph gets triangulated).

- **Junction Tree.** If the graph is *triangulated*, then we can get clique trees which obey the *junction tree property* by applying the MST algorithm based on maximizing separator cardinalities. Such trees are also called junction tree. If the tree is a junction tree, we can further guarantee that the algorithm 1) improves local consistency to global consistency, 2) is correct in generating marginals for the cliques and separators.

Therefore, the complete algorithm would be 1) moralize the directed graph to indirected graph, 2) introduce evidence, 3) triangulate (where approximation is introduced), 4) build junction tree by first select maximal cliques, then select separators as MST, 5) performance message passing by selecting root, collection and distribution, 5) extract single-node marginals.