# Directed Factor Graph Notation for Generative Models

Laura Dietz
dietz@mpi-inf.mpg.de
Max Planck Institute for Informatics, Saarbrücken, Germany

We introduce the directed factor graph notation, a visual language for specifying the generative process of a probabilistic model. In contrast to boiler plate diagrams, directed factor graphs provide more information about the generative process, allowing to judge the complexity of the model at a glance.

## 1 Introduction

Generative probabilistic models encode a set of assumptions on how observed data is generated, which is called the generative process. The use of latent variables and parameters in these assumptions explains the data by some underlying unobserved states. It is possible to reverse the generative process to—given the data—estimate variables and parameters, which in turn gives rise to predictors in an unsupervised manner.

Parameter estimation procedures such as Gibbs sampling [1], variational message passing [2], or expectation propagation [3] provide templates for algorithms and need to be tailored to the generative model. Here the generative process plays the central role in determining the final learning algorithm.

The generative process is either specified as pseudocode or as a boiler plate diagram, where the former is more accurate, but the latter is better in visualizing the independence assumptions made.

This work introduces the directed factor graph notation, which enhances the visual notation of boiler plate diagrams towards the accuracy of the probabilistic pseudocode. It builds on the gates notation [4] which visualizes "conditioned on" relationships. Additionally it is based on the factor graph

notation [5], which depicts the factorization of the joint distribution. It provides a flexible visual notation with clear semantics for probabilistic models, such as the unified modelling language (UML [6]) provides for software.

Factor graphs are undirected. However in generative models, the factors are usually members of the exponential family and are used with their respective conjugate priors. Distributions in the exponential family, such as Gaussian, multinomial, and beta-Bernoulli distinguish between input parameters of the distribution and random variables drawn from this distribution (output variables). This input-output relationship is visualized by introducing directed edges to the factor graph. For many models, the directed factor graph is as expressive as the generative process in pseudocode notation.

**Outline.** The visual notation is introduced in Section 2. Examples of how models translate to directed factor graph notation are given in Section 3, with benefits dicussed. The paper concludes in Section 4.

# 2 Notation

In this section, we introduce the graphical primitives and how they relate to probabilistic concepts.

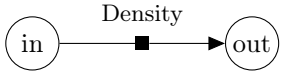
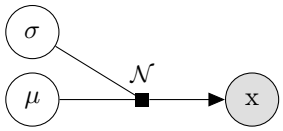
## 2.1 Variables and Constants

The factor graph consists of two kinds of nodes: variables and factors. Variables are depicted by a circle, which is shaded in the case of observed variables. Constants or hyperparameters are denoted without border.

| | Directed factor graph | Pseudocode |
|---|---|---|
| Llatent variable / latent parameter |   | |
| Observed variable |   | |
| Constant / hyper parameter | const | |

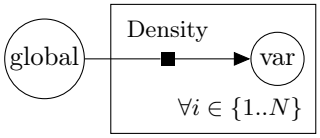
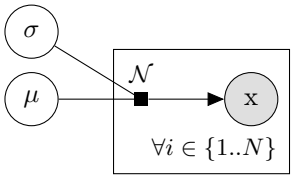
## 2.2 Factors and Densities

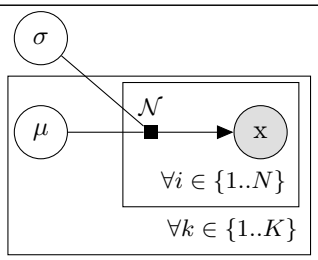
Factors are depicted by small filled boxes. The name of the density is written next to the box. Input parameters and output variables are connected by edges, with an arrow pointing towards the output variable.

| | Directed factor graph | Pseudocode |
|---|---|---|
| Factor with one input parameter |  | 1: **draw** out $\sim$ Density(in) |
| Example: Gaussian |  | 1: **draw** $x \sim \mathcal{N}(\mu, \sigma)$ |

## 2.3 Replication with Plates

If variables or factors are to be repeated, plates provide a notation for this. A template of variables, factors and connections that is to be iterated over, is placed inside the plate. Text in the lower right corner indicates how many replicas are created. Specifying an iteration variable induces an index for the repeated variables. Connections across the plate border represent multiple connections between each replica and the variable/factor outside the plate.

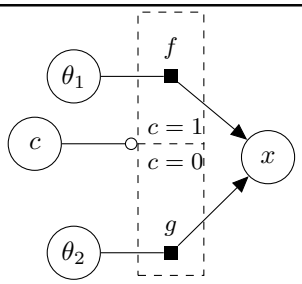
| | Directed factor graph | Pseudocode |
|---|---|---|
| Plate |  | 1: **for all** $\forall i \in \{1..N\}$ **do**<br>2: **draw** $\text{var}_i \sim$global |
| Example: repeated Gaussian |  | 1: **for all** $\forall i \in \{1..N\}$ **do**<br>2: **draw** $x_i \sim \mathcal{N}(\mu, \sigma)$ |

| | Directed factor graph | Pseudocode |
|---|---|---|
| Nested plates |  | 1: **for all** $\forall k \in \{1..K\}$ **do**<br>2: **for all** $\forall i \in \{1..N\}$ **do**<br>3: **draw** $x_{k,i} \sim \mathcal{N}(\mu_k, \sigma)$ |

## 2.4 Conditioning with Gates

If depending on the configuration of a variable, the model should do something slightly different, this can be represented by gates [4]. In contrast to a plate which represents replication, gates represent selection. Analogously to plates, a structural template of variables and factors is placed inside the gate. Text in the upper right corner of the gate specifies under which condition the structural template becomes active. Input variables to the conditions are attached to the gate boundary. Several gates of which only one may be active are placed adjacently to each other.

Whenever the generative pseudocode contains if-then-else or case constructs or a latent variable is used as an index, the analogue in directed factor graphs is a gate.

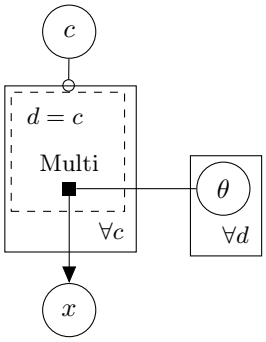
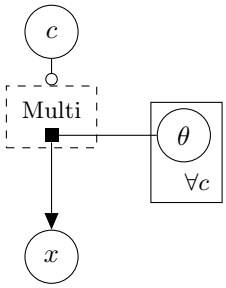
| | Directed factor graph | Pseudocode |
|---|---|---|
| Unrolled boolean gate |  | 1: **if** $c = 1$ **then**<br>2: **draw** $x \sim f(\theta_1)$<br>3: **else**<br>4: **draw** $x \sim g(\theta_2)$ |

If the gate is nested inside a plate, the plate replicates the gate and the condition may depend on the iterator variable of the plate as well. This is particularly useful, for instance, when the control variable is drawn from a multinomial and different input parameters are used for each possible outcome.

**Algorithm 1** Generative process of latent Dirichlet allocation.

1: **for all** $M$ documents **do**
2: **draw** $\theta \sim \mathrm{Dir}(\alpha)$
3: **for all** of the $N$ words $w_n$ **do**
4: **draw** a topic $z_n \sim \mathrm{Multi}(\theta)$
5: **draw** a word $w_n \sim \mathrm{Multi}(\beta_{z_n})$

For brevity, the plate around the single-box gate may be dropped when the iteration range is obvious. In cases, where the control variable itself is used to index the input variables, the gate condition description may be dropped as well.

| | Directed factor graph | Pseudocode |
|---|---|---|
| Replicated gate |   | 1: **draw** $x \sim \mathrm{Multi}(\theta_c)$ |
| Implicit notation for replicating gates |   | 1: **draw** $x \sim \mathrm{Multi}(\theta_c)$ |

## 3 Examples

The classical alternative are boiler plate diagrams, which do not explicitly name factors and usually do not use the gates notation.

Recent trends in generative models use different connections in the model

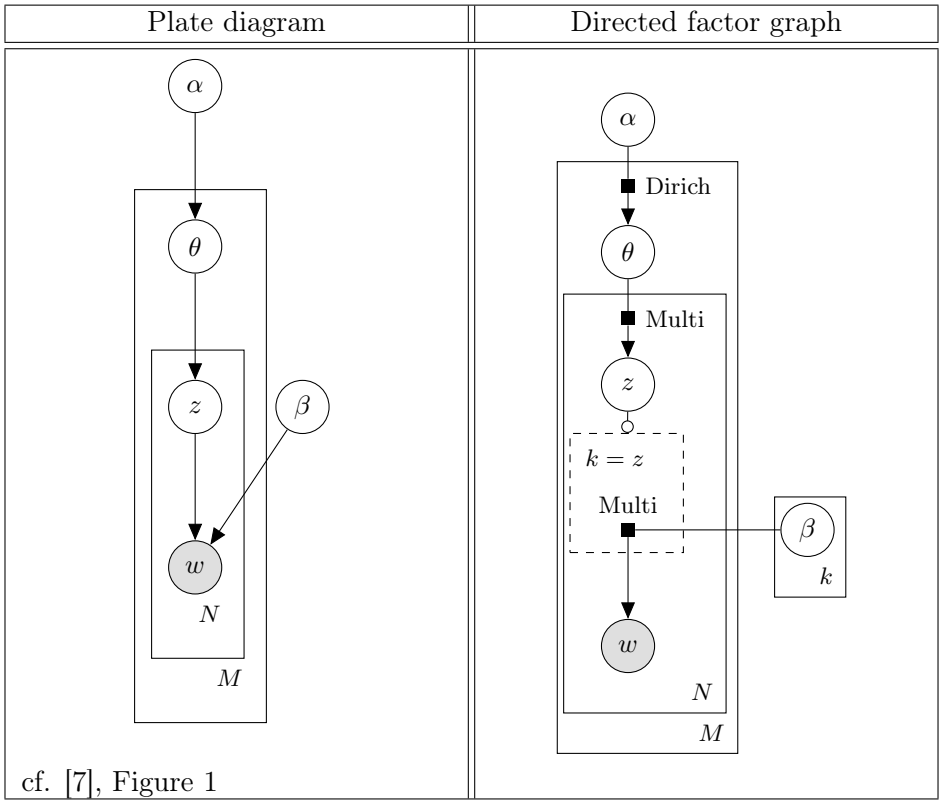



Figure 1: The latent Dirichlet allocation model as boiler plate diagram (left), directed factor graph (right).

**Algorithm 2** Citation influence model in pseudocode notation.

**for all** topics $t \in \{1..T\}$ **do**
  **draw** $\phi_t \sim \text{Dirich}(\alpha_\phi)$
**for all** cited documents $c' \in C$ **do**
  **draw** topic mixture $\theta_{c'} \sim \text{Dirich}(\alpha_\theta)$
  **for all** tokens $j$ **do**
    **draw** topic $t'_{c',j} \sim \text{Multi}(\theta_{c'})$
    **draw** word $w_{c',j} \sim \text{Multi}(\phi_{t'_{c',j}})$
**for all** citing documents $d \in D$ **do**
  **draw** citation mixture $\gamma_d \sim \text{Dirich}(\alpha_\gamma)$ ranging over citations $L(d)$
  **draw** innovation topic mixture $\psi_d \sim \text{Dirich}(\alpha_\psi)$
  **draw** mixing coin $\lambda_d \sim \text{Beta}(\alpha_{\lambda_\theta}, \alpha_{\lambda_\psi})$
  **for all** tokens $i$ **do**
    **draw** control variable $s_{d,i} \sim \text{Bern}(\lambda_d)$
    **if** $s_{d,i} = 0$ **then**
      **draw** cited document $c_{d,i} \sim \text{Multi}(\psi_d)$
      **draw** topic $t_{d,i} \sim \text{Multi}(\theta_{c_{d,i}})$ from the cited document
    **else**
      **draw** topic $t_{d,i} \sim \text{Multi}(\psi_d)$
    **draw** word $w_{d,i} \sim \text{Multi}(\phi_{t_{d,i}})$

depending on a latent variable. One simple example is latent Dirichlet allocation [7] in Figure 1 and Algorithm 1, where the gate clarifies that $w$ is not drawn from a complicated distribution, but using a different parameter $\beta$, conditioned on $z$. Furthermore, the conjugacy in the prior and likelihood density can be verified at a glance.

Another model, that looks complicated in boiler plate notation, but has a straight forward parameter estimation algorithm is the citation influence model [8]. The generative process is given as pseudocode in Algorithm 2. In the boiler plate diagram (Figure 2), it seems as if $t_{d,i}$ is drawn from a complicated distribution depending on four different parameters. The directed factor graph (Figure 3) clarifies, that $t_{d,i}$ is always drawn from a multinomial parameter with a Dirichlet prior. Further it clarifies that random variables $s_{d,i}$ and $c_{d,i}$ act as multiplexers between different multinomial parameters. Furthermore, it can be easily verified that only conjugate priors are used in the model, which allows for fast parameter estimation.

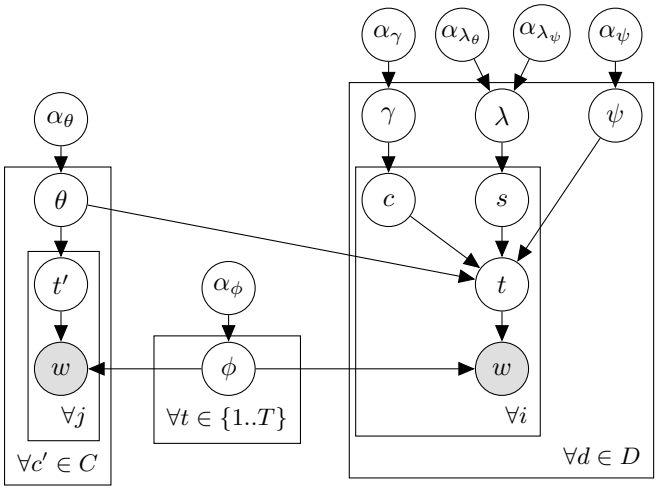



Figure 2: The citation influence model in boiler plate notation.

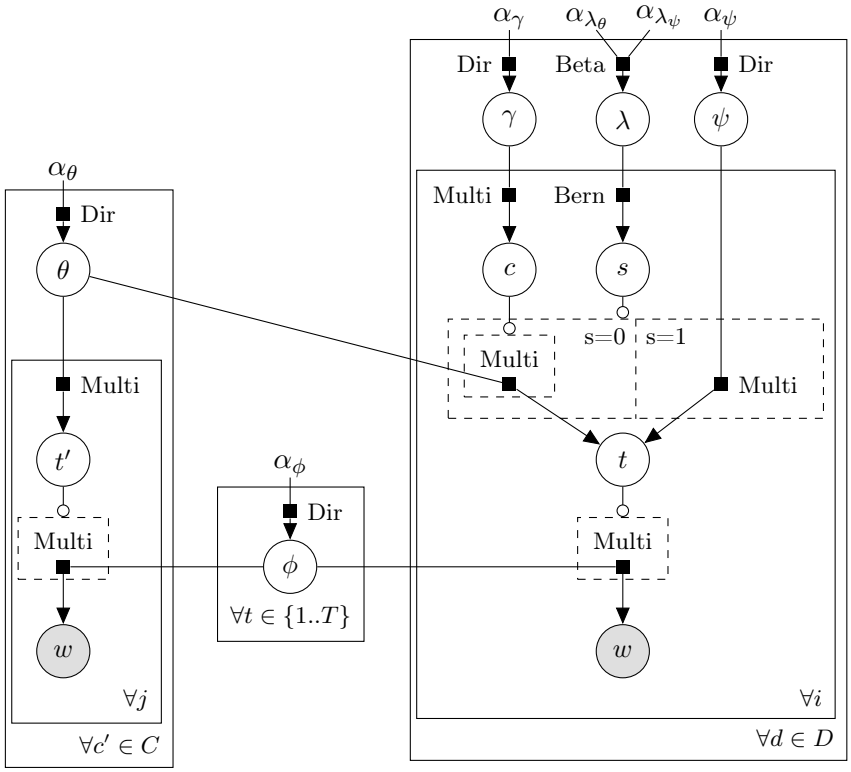



Figure 3: The citation influence model in directed factor graph notation.

## 4 Conclusion

This work introduces the directed factor graph notation, which incorporates features of boiler plate diagrams, factor graphs, and gates. Properties that determine the complexity of parameter inference are independence of variables, use of conjugate priors, and use of distributions from the exponential family. All these properties are visualized in directed factor graphs.

For many tractable probabilistic models, the directed factor graph representation is as rich as the pseudocode notation. Thus, directed factor graphs render the dual visualization of boiler plate and pseudocode unneccessary.

## References


[1] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, London, UK, 1996.

[2] John Winn, Christopher M. Bishop, and Tommi Jaakkola. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.

[3] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

[4] Tom Minka and John Winn. Gates. In *NIPS '08: Advances in Neural Information Processing Systems*, 2008.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, New York, NY, USA, 2007.

[6] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition)*. Addison-Wesley Professional, 3 edition, September 2003.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[8] Laura Dietz, Steffen Bickel, and Tobias Scheffer. Unsupervised prediction of citation influences. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, 2007.