Bayesuvius, a small visual dictionary of Bayesian Networks

Robert R. Tucci
www.ar-tiste.xyz

March 25, 2021

Figure 1: View of Mount Vesuvius from Pompeii

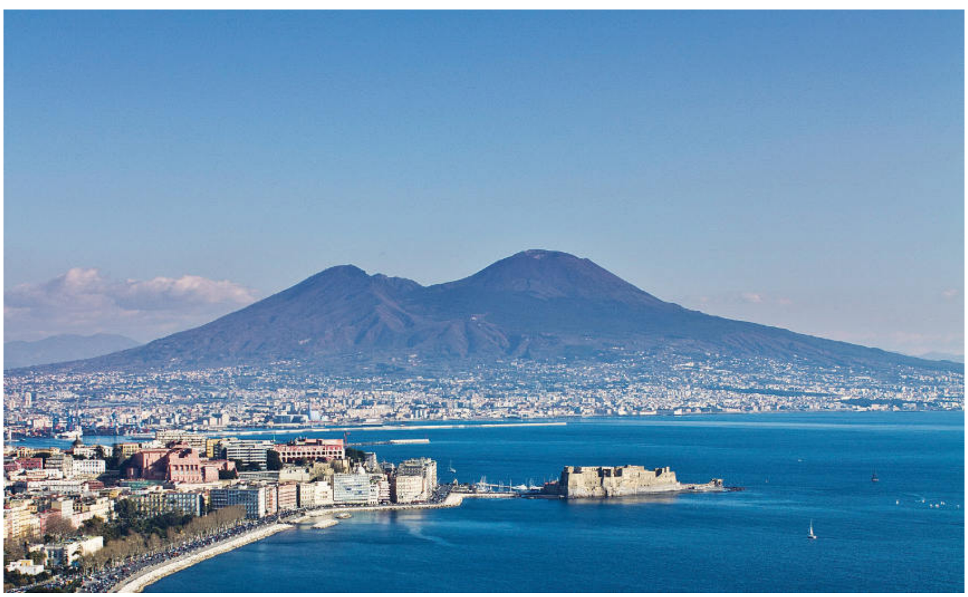

Figure 2: Mount Vesuvius and Bay of Naples

Contents

Chapter 41

Recurrent Neural Networks

This chapter is mostly based on Ref.[19].

This chapter assumes you are familiar with the material and notation of Chapter 35 on plain Neural Nets.

$$
\begin{tikzcd}
\underline{x}(\cdot) \arrow[d] \arrow[dr] \arrow[drr] & & \\
\underline{x}(0) \arrow[d] & \underline{x}(1) \arrow[d] & \underline{x}(2) \arrow[d] \\
\underline{h}(0) \arrow[r] \arrow[d] & \underline{h}(1) \arrow[r] \arrow[d] & \underline{h}(2) \arrow[d] \\
\underline{Y}(0) & \underline{Y}(1) & \underline{Y}(2)
\end{tikzcd}
$$

Figure 41.1: Simple example of RNN witb $T = 3$

Suppose
T is a positive integer.
$t = 0, 1, \dots, T-1$,
$\underline{x}_i(t) \in \mathbb{R}$ for $i = 0, 1, \dots, numx - 1$,
$\underline{h}_i(t) \in \mathbb{R}$ for $i = 0, 1, \dots, numh - 1$,
$\underline{Y}_i(t) \in \mathbb{R}$ for $i = 0, 1, \dots, numy - 1$,
$W^{h|x} \in \mathbb{R}^{numh \times numx}$,
$W^{h|h} \in \mathbb{R}^{numh \times numh}$,
$W^{y|h} \in \mathbb{R}^{numy \times numh}$,
$b^y \in \mathbb{R}^{numy}$,
$b^h \in \mathbb{R}^{numh}$.
Henceforth, $x(\cdot)$ will mean the array of $x(t)$ for all t.

The simplest kind of recurrent neural network (RNN) has the bnet Fig.41.1 with arbitrary T. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(x(\cdot)) = \text{given} \tag{41.1}$$

$$P(x(t)) = \delta(x(t), [x(\cdot)]_t) \tag{41.2}$$

$$P(h(t) \mid h(t-1), x(t)) = \delta(h(t), \mathcal{A}(W^{h|x}x(t) + W^{h|h}h(t-1) + b^h)) \;, \tag{41.3}$$

where $h(-1) = 0$.

$$P(Y(t) \mid h(t)) = \delta(Y(t), \mathcal{A}(W^{y|h}h(t) + b^y)) \tag{41.4}$$

Define

$$W^h = [W^{h|x}, W^{h|h}, b^h] \;, \tag{41.5}$$

and

$$W^y = [W^{y|h}, b^y] \;. \tag{41.6}$$

The bnet of Fig.41.1 can be used for classification once its parameters W^h and W^y have been optimized. To optimize those parameters via gradient descent, one can use the bnet of Fig.41.2.

Let $\sigma = 0, 1, \ldots, nsam(\vec{x}) - 1$ be the labels for a minibatch of samples. The node TPMs, printed in blue, for bnet Fig.41.2, are as follows.

$$P(x(\cdot)[\sigma]) = \text{given} \tag{41.7}$$

$$P(x(t)[\sigma]) = \delta(x(t)[\sigma], [x(\cdot)]_t[\sigma]) \tag{41.8}$$

$$P(h(t)[\sigma] \mid h(t-1)[\sigma], x(t)[\sigma]) = \delta(h(t)[\sigma], \mathcal{A}(W^{h|x}x(t)[\sigma] + W^{h|h}h(t-1)[\sigma] + b^h) \tag{41.9}$$

$$P(Y(t)[\sigma] \mid h(t-1)[\sigma]) = \delta(Y(t)[\sigma], \mathcal{A}(W^{y|h}h(t-1)[\sigma] + b^y) \tag{41.10}$$

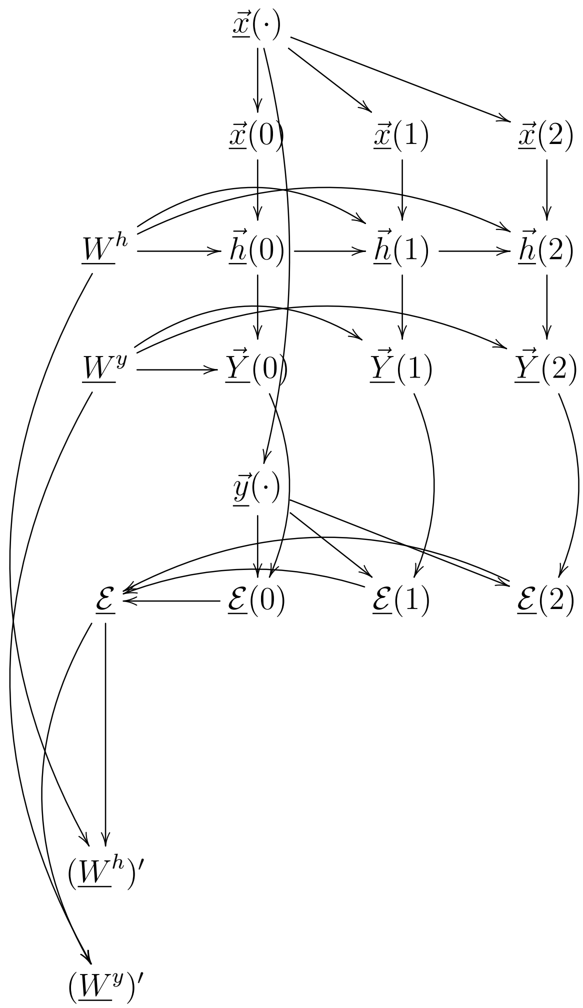

Figure 41.2: RNN bnet used to optimize parameters W^h and W^y of RNN bnet Fig.41.1.

$$P(y(\cdot)[\sigma] \mid x(\cdot)[\sigma]) = \text{given} \tag{41.11}$$

$$P(\mathcal{E}(t) \mid \vec{y}(\cdot), \vec{Y}(t)) = \frac{1}{nsam(\vec{x})} \sum_\sigma d(y(t)[\sigma], Y(t)[\sigma]) \;, \tag{41.12}$$

where

$$d(y, Y) = |y - Y|^2 \;. \tag{41.13}$$

If $y, Y \in [0,1]$, one can use this instead

$$d(y, Y) = XE(y \to Y) = -y \ln Y - (1-y)\ln(1-Y) \; . \tag{41.14}$$

$$P(\mathcal{E} \mid [\mathcal{E}(t)]_{\forall t}) = \delta(\mathcal{E}, \sum_t \mathcal{E}(t)) \tag{41.15}$$

For $a = h, y$,

$$P(W^a) = \text{given} \; . \tag{41.16}$$

The first time it is used, W^a is fairly arbitrary. Afterwards, it is determined by previous horizontal stage.

$$P((W^a)'|\mathcal{E}, W^a) = \delta((W^a)', W^a - \eta^a \partial_{W^a}\mathcal{E}) \; . \tag{41.17}$$

$\eta^a > 0$ is the learning rate for W^a.

41.1 Language Sequence Modeling

Figs.41.1, and 41.2 with arbirary T can be used as follows to do Language Sequence Modeling.

For this usecase, one must train with the following TPM for node $\vec{y}(\cdot)$:

$$P(y(\cdot)[\sigma] \mid x(\cdot)[\sigma]) = \prod_t \mathbb{1}(\quad y(t)[\sigma] = P(x(t)[\sigma] \mid [x(t')[\sigma]]_{t'<t}) \quad) \tag{41.18}$$

With such training, one gets

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = P(x(t) \mid [x(t')]_{t'<t}) \quad) \; . \tag{41.19}$$

Therefore,

$$Y(0) = P(x(0)) \; , \tag{41.20}$$

$$Y(1) = P(x(1)|x(0)) \; , \tag{41.21}$$

$$Y(2) = P(x(2)|x(0), x(1)) \; , \tag{41.22}$$

and so on.

We can use this to:

- predict the probability of a sentence,

  example: Get $P(x(0), x(1), x(2))$.

- predict the most likely next word in a sentence,
  example: Get $P(x(2)|x(0), x(1))$.
- generate fake sentences.
  example:
  Get $x(0) \sim P(x(0))$.
  Next get $x(1) \sim P(x(1)|x(0))$.
  Next get $x(2) \sim P(x(2)|x(0), x(1))$.

41.2 Other types of RNN

$$
\begin{tikzcd}
\underline{x}(\cdot) \arrow[d] \arrow[dr] & & & \\
\underline{x}(0) \arrow[d] & \underline{x}(1) \arrow[d] & & \\
\underline{h}(0) \arrow[r] & \underline{h}(1) \arrow[r] & \underline{h}(2) \arrow[r] \arrow[d] & \underline{h}(3) \arrow[d] \\
& & \underline{Y}(2) & \underline{Y}(3)
\end{tikzcd}
$$

Figure 41.3: RNN bnet of the many to many kind. This one can be used for translation. $x(0)$ and $x(1)$ might denote two words of an English sentence, and $Y(2)$ and $Y(3)$ might be their Italian translation.

Let $\mathcal{T} = \{0, 1, \ldots, T-1\}$, and $\mathcal{T}^x, \mathcal{T}^y \subset \mathcal{T}$. Above, we assumed that $\underline{x}(t)$ and $\underline{Y}(t)$ were both defined for all $t \in \mathcal{T}$. More generally, they might be defined only for subsets of $\mathcal{T}$: $\underline{x}(t)$ for $t \in \mathcal{T}^x$ and $\underline{Y}(t)$ for $t \in \mathcal{T}^y$. If $|\mathcal{T}^x| = 1$ and $|\mathcal{T}^y| > 1$, we say the RNN bnet is of the 1 to many kind. In general, can have **1 to 1, 1 to many, many to 1, many to many** RNN bnets.

Plain RNNs can suffer from the **vanishing or exploding gradients problem**. There are various ways to mitigate this (good choice of initial W^h and W^y, good choice of activation functions, regularization). Or by using GRU or LSTM (discussed below). **GRU and LSTM** were designed to mitigate the vanishing or exploding gradients problem. They are very popular in NLP (Natural Language Processing).

41.2.1 Long Short Term Memory (LSTM) unit (1997)

This section is based on Wikipedia article Ref.[63]. In this section, $\odot$ will denote the Hadamard matrix product (elementwise product).

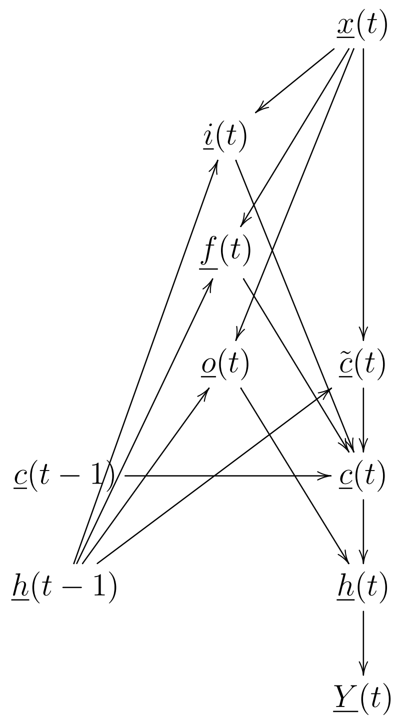

Figure 41.4: bnet for a Long Short Term Memory (LSTM) unit.

Let
$\underline{x}(t) \in \mathbb{R}^{numx}$: input vector to the LSTM unit
$\underline{f}(t) \in \mathbb{R}^{numh}$: forget gate's activation vector
$\underline{i}(t) \in \mathbb{R}^{numh}$: input/update gate's activation vector
$\underline{o}(t) \in \mathbb{R}^{numh}$: output gate's activation vector
$\underline{h}(t) \in \mathbb{R}^{numh}$: hidden state vector also known as output vector of the LSTM unit
$\underline{\tilde{c}}(t) \in \mathbb{R}^{numh}$: cell input activation vector
$\underline{c}(t) \in \mathbb{R}^{numh}$: cell state vector
$\underline{Y}(t) \in \mathbb{R}^{numy}$: classification of $x(t)$.
$W \in \mathbb{R}^{numh \times numx}$, $U \in \mathbb{R}^{numh \times numh}$ and $b \in \mathbb{R}^{numh}$: weight matrices and bias vectors, parameters learned by training.
$\mathcal{W}^{y|h} \in \mathbb{R}^{numy \times numh}$: weight matrix

Fig.41.4 is a bnet net for a LSTM unit. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(f(t)|x(t),h(t-1)) = \mathbb{1}(\quad f(t) = \text{sig}(W^{f|x}x(t) + U^{f|h}h(t-1) + b^f) \quad), \tag{41.23}$$

where $h(-1) = 0$.

$$P(i(t)|x(t),h(t-1)) = \mathbb{1}(\quad i(t) = \text{sig}(W^{i|x}x(t) + U^{i|h}h(t-1) + b^i) \quad) \tag{41.24}$$

$$P(o(t)|x(t),h(t-1)) = \mathbb{1}(\quad o(t) = \text{sig}(W^{o|x}x(t) + U^{o|h}h(t-1) + b^o) \quad) \tag{41.25}$$

$$P(\tilde{c}(t)|x(t),h(t-1)) = \mathbb{1}(\quad \tilde{c}(t) = \tanh(W^{c|x}x(t) + U^{c|h}h(t-1) + b^c) \quad) \tag{41.26}$$

$$P(c(t)|f(t),c(t-1),i(t),\tilde{c}(t)) = \mathbb{1}(\quad c(t) = f(t) \odot c(t-1) + i(t) \odot \tilde{c}(t) \quad) \tag{41.27}$$

$$P(h(t)|o(t),c(t)) = \mathbb{1}(\qquad h(t) = o(t) \odot \tanh(c(t)) \quad) \tag{41.28}$$

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = \mathcal{A}(\mathcal{W}^{y|h}h(t) + b^y) \quad) \tag{41.29}$$

41.2.2 Gated Recurrence Unit (GRU) (2014)

This section is based on Wikipedia article Ref.[52]. In this section, $\odot$ will denote the Hadamard matrix product (elementwise product).

GRU is a more recent (17 years later) attempt at simplifying LSTM unit.

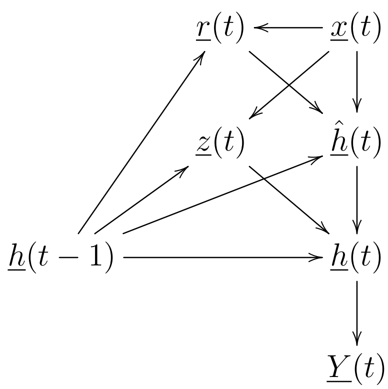

Figure 41.5: bnet for a Gated Recurrent Unit (GRU).

Let

$\underline{x}(t) \in \mathbb{R}^{numx}$: input vector

$\underline{h}(t) \in \mathbb{R}^{numh}$: output vector

$\underline{\hat{h}}(t) \in \mathbb{R}^{numh}$: candidate activation vector

$\underline{z}(t) \in \mathbb{R}^{numh}$: update gate vector

$\underline{r}(t) \in \mathbb{R}^{numh}$: reset gate vector

$\underline{Y}(t) \in \mathbb{R}^{numy}$: classification of $x(t)$.

$W \in \mathbb{R}^{numh \times numx}$, $U \in \mathbb{R}^{numh \times numh}$ and $b \in \mathbb{R}^{numh}$: weight matrices and bias vectors, parameters learned by training.

$\mathcal{W}^{y|h} \in \mathbb{R}^{numy \times numh}$: weight matrix

Fig.41.5 is a bnet net for a GRU. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(z(t)|x(t), h(t-1)) = \mathbb{1}(\quad z(t) = \text{sig}(W^{z|x}x(t) + U^{z|h}h(t-1) + b^z) \quad) \;, \tag{41.30}$$

where $h(-1) = 0$.

$$P(r(t)|x(t), h(t-1)) = \mathbb{1}(\quad r(t) = \text{sig}(W^{r|x}x(t) + U^{r|h}h(t-1) + b^r) \quad) \tag{41.31}$$

$$P(\hat{h}(t)|x(t), r(t), h(t-1)) = \mathbb{1}(\quad \hat{h}(t) = \tanh(W^{h|x}x(t) + U^{h|h}(r(t) \odot h(t-1)) + b^h) \quad) \tag{41.32}$$

$$P(h(t)|z(t), h(t-1), \hat{h}(t)) = \mathbb{1}(\quad h(t) = (1 - z(t)) \odot h(t-1) + z(t) \odot \hat{h}(t) \quad) \tag{41.33}$$

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = \mathcal{A}(\mathcal{W}^{y|h} h(t) + b^y) \quad) \tag{41.34}$$

Bibliography

[1] Dan Bendel. Metropolis-Hastings: A comprehensive overview and proof. `https://similarweb.engineering/mcmc/`.

[2] Alexandra M Carvalho. Scoring functions for learning Bayesian networks. `http://www.lx.it.pt/~asmc/pub/talks/09-TA/ta_pres.pdf`.

[3] Scott Cunningham. *Causal inference: The mixtape*. Yale University Press, 2021. `https://mixtape.scunning.com/index.html`.

[4] Robin J. Evans. Graphical methods for inequality constraints in marginalized DAGs. `https://arxiv.org/abs/1209.2978`.

[5] Matheus Facure Alves. *Causal Inference for The Brave and True*. 2021. `https://matheusfacure.github.io/python-causality-handbook/landing-page.html`.

[6] George Fei. Modeling uplift directly: Uplift decision tree with kl divergence and euclidean distance as splitting criteria. `https://tinyurl.com/yhnzwj58`.

[7] Charles Fox, Neil Girdhar, and Kevin Gurney. A causal Bayesian network view of reinforcement learning. `https://www.aaai.org/Papers/FLAIRS/2008/FLAIRS08-030.pdf"`.

[8] Bruno Gonçalves. Model testing and causal search. blog post `https://medium.com/data-for-science/causal-inference-part-vii-model-testing-and-causal-search-536b796f0384`.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, David Warde-Farley Bing Xu, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. `https://arxiv.org/abs/1406.2661`.

[10] Pierre Gutierrez and Jean-Yves Grardy. Causal inference and uplift modelling: A review of the literature. In *Proceedings of The 3rd International Conference on Predictive Applications and APIs*, pages 1–13, 2017. `http://proceedings.mlr.press/v67/gutierrez17a.html`.

[11] Christina Heinze-Deml. Causality, spring semester 2019 at ETH Zurich. https://stat.ethz.ch/lectures/ss19/causality.php#course_materials.

[12] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International journal of approximate reasoning*, 15(3):225–263, 1996. http://www.ar-tiste.com/Huang-Darwiche1996.pdf.

[13] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988. http://www.eecis.udel.edu/~shatkay/Course/papers/Lauritzen1988.pdf.

[14] Sergey Levine. Course CS 285 at UC Berkeley, Deep reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse/.

[15] Dimitris Margaritis. Learning Bayesian network model structure from data (thesis, 2003, Carnegie Mellon Univ). https://apps.dtic.mil/sti/citations/ADA461103.

[16] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. In *BMC bioinformatics*, volume 7, page S7. Springer, 2006. https://link.springer.com/article/10.1186/1471-2105-7-S1-S7.

[17] Robert J. McEliece, David J. C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearls belief propagation algorithm. http://authors.library.caltech.edu/6938/1/MCEieeejstc98.pdf.

[18] Richard E Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall, 2004.

[19] Andrew Ng. Lecture at deeplearning.ai on recurrent neural networks. http://www.ar-tiste.com/ng-lec-rnn.pdf.

[20] Gregory Nuel. Tutorial on exact belief propagation in Bayesian networks: from messages to algorithms. https://arxiv.org/abs/1201.4724.

[21] Judea Pearl. Linear models: A useful microscope for causal analysis. https://ftp.cs.ucla.edu/pub/stat_ser/r409-corrected-reprint.pdf.

[22] Judea Pearl. Mediating instrumental variables. https://ftp.cs.ucla.edu/pub/stat_ser/r210.pdf.

[23] Judea Pearl. On the testability of causal models with latent and instrumental variables. https://arxiv.org/abs/1302.4976.

[24] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. https://www.aaai.org/Papers/AAAI/1982/AAAI82-032.pdf, 1982.

[25] Judea Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, 1988.

[26] Judea Pearl. *Causality: Models, Reasoning, and Inference, Second Edition*. Cambridge University Press, 2013.

[27] Judea Pearl. Causal and counterfactual inference. *The Handbook of Rationality*, pages 1–41, 2019. https://ftp.cs.ucla.edu/pub/stat_ser/r485.pdf.

[28] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[29] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[30] ReliaSoft. System analysis reference. http://reliawiki.org/index.php/System_Analysis_Reference.

[31] Piotr Rzepakowski and Szymon Jaroszewicz. Decision trees for uplift modeling with single and multiple treatments. *Knowledge and Information Systems*, 32(2):303–327, 2012. https://link.springer.com/content/pdf/10.1007/s10115-011-0434-0.pdf.

[32] Marco Scutari. bnlearn. https://www.bnlearn.com/.

[33] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019. https://arxiv.org/abs/1805.11908.

[34] Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[35] Masayoshi Takahashi. Statistical inference in missing data by MCMC and non-MCMC multiple imputation algorithms: Assessing the effects of between-imputation iterations. *Data Science Journal*, 16, 2017. https://datascience.codata.org/articles/10.5334/dsj-2017-037/.

[36] theinvestorsbook.com. Pert analysis. https://theinvestorsbook.com/pert-analysis.html.

[37] Robert R. Tucci. Bell's inequalities for Bayesian statisticians. blog post in blog Quantum Bayesian Networks, https://qbnets.wordpress.com/2008/09/19/bells-inequaties-for-bayesian-statistician/.

[38] Robert R. Tucci. Quantum Fog. https://github.com/artiste-qb-net/quantum-fog.

[39] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook nureg-0492. https://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/.

[40] Wikipedia. Belief propagation. https://en.wikipedia.org/wiki/Belief_propagation.

[41] Wikipedia. Berkson's paradox. https://en.wikipedia.org/wiki/Berkson%27s_paradox.

[42] Wikipedia. Beta function. https://en.wikipedia.org/wiki/Beta_function.

[43] Wikipedia. Binary decision diagram. https://en.wikipedia.org/wiki/Binary_decision_diagram.

[44] Wikipedia. Boolean algebra. https://en.wikipedia.org/wiki/Boolean_algebra.

[45] Wikipedia. Categorical distribution. https://en.wikipedia.org/wiki/Categorical_distribution.

[46] Wikipedia. Chow-Liu tree. https://en.wikipedia.org/wiki/Chow%E2%80%93Liu_tree.

[47] Wikipedia. Data processing inequality. https://en.wikipedia.org/wiki/Data_processing_inequality.

[48] Wikipedia. Dirichlet distribution. https://en.wikipedia.org/wiki/Dirichlet_distribution.

[49] Wikipedia. Errors in variables models. https://en.wikipedia.org/wiki/Errors-in-variables_models.

[50] Wikipedia. Expectation maximization. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

[51] Wikipedia. Gamma function. https://en.wikipedia.org/wiki/Gamma_function.

[52] Wikipedia. Gated recurrent unit. https://en.wikipedia.org/wiki/Gated_recurrent_unit.

[53] Wikipedia. Gibbs sampling. https://en.wikipedia.org/wiki/Gibbs_sampling.

[54] Wikipedia. Hidden Markov model. https://en.wikipedia.org/wiki/Hidden_Markov_model.

[55] Wikipedia. Importance sampling. https://en.wikipedia.org/wiki/Importance_sampling.

[56] Wikipedia. Instrumental variables estimation. https://en.wikipedia.org/wiki/Instrumental_variables_estimation.

[57] Wikipedia. Inverse transform sampling. https://en.wikipedia.org/wiki/Inverse_transform_sampling.

[58] Wikipedia. Junction tree algorithm. https://en.wikipedia.org/wiki/Junction_tree_algorithm.

[59] Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering.

[60] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter.

[61] Wikipedia. Least squares. https://en.wikipedia.org/wiki/Least_squares.

[62] Wikipedia. Linear regression. https://en.wikipedia.org/wiki/Linear_regression.

[63] Wikipedia. Long short term memory. https://en.wikipedia.org/wiki/Long_short-term_memory.

[64] Wikipedia. Markov blanket. https://en.wikipedia.org/wiki/Markov_blanket.

[65] Wikipedia. Metropolis-Hastings method. https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm.

[66] Wikipedia. Minimum spanning tree. https://en.wikipedia.org/wiki/Minimum_spanning_tree.

[67] Wikipedia. Monte Carlo methods. https://en.wikipedia.org/wiki/Category:Monte_Carlo_methods.

[68] Wikipedia. Multinomial distribution. https://en.wikipedia.org/wiki/Multinomial_distribution.

[69] Wikipedia. Multinomial theorem. https://en.wikipedia.org/wiki/Multinomial_theorem.

[70] Wikipedia. Multivariate normal distribution. https://en.wikipedia.org/wiki/Multivariate_normal_distribution.

[71] Wikipedia. Natural experiment. https://en.wikipedia.org/wiki/Natural_experiment.

[72] Wikipedia. Non-negative matrix factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.

[73] Wikipedia. Ordinary least squares. https://en.wikipedia.org/wiki/Ordinary_least_squares.

[74] Wikipedia. Program evaluation and review technique. https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique.

[75] Wikipedia. Receiver operating characteristic. https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[76] Wikipedia. Rejection sampling. https://en.wikipedia.org/wiki/Rejection_sampling.

[77] Wikipedia. Simple linear regression. https://en.wikipedia.org/wiki/Simple_linear_regression.

[78] Wikipedia. Simpson's paradox. https://en.wikipedia.org/wiki/Simpson's_paradox.

[79] Wikipedia. Spring system. https://en.wikipedia.org/wiki/Spring_system.

[80] Wikipedia. Uplift modelling. https://en.wikipedia.org/wiki/Uplift_modelling.

[81] Wikipedia. Variational Bayesian methods. https://en.wikipedia.org/wiki/Variational_Bayesian_methods.

[82] Hao Wu and Zhaohui Steve Qin. course notes, BIOS731: Advanced statistical computing, 2016 Emory Univ. http://web1.sph.emory.edu/users/hwu30/teaching/statcomp/statcomp.html.