Statistical Models & Computing Methods

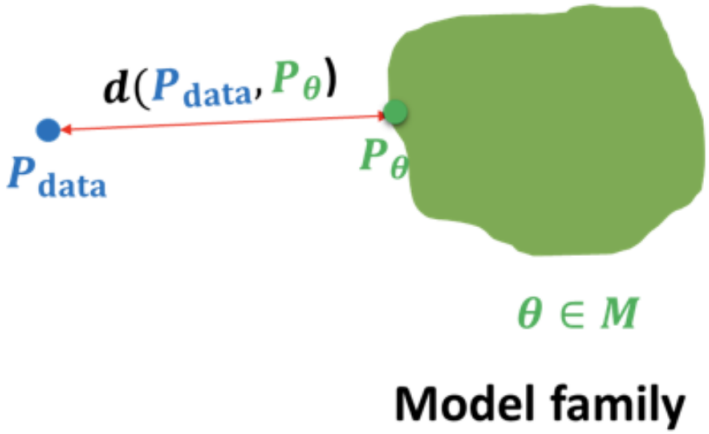
Lecture 12: Generative Adversarial Nets and Bayesian Phylogenetic Inference

Cheng Zhang

School of Mathematical Sciences, Peking University

December 24, 2020

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



- Model families
  - Autoregressive Models: $p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_{<i})$
  - Variational Autoencoders: $p_\theta(x) = \int_z p_\theta(x, z) dz$
  - Normalizing Flow Models:

    $$p_X(x; \theta) = p_Z(f_\theta^{-1}(x)) \left| \det \left(\frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|$$
- All the above families are based on maximizing likelihoods (or approximations, e.g., lower bound)
- Is the likelihood a good indicator of the quality of samples generated by the model?

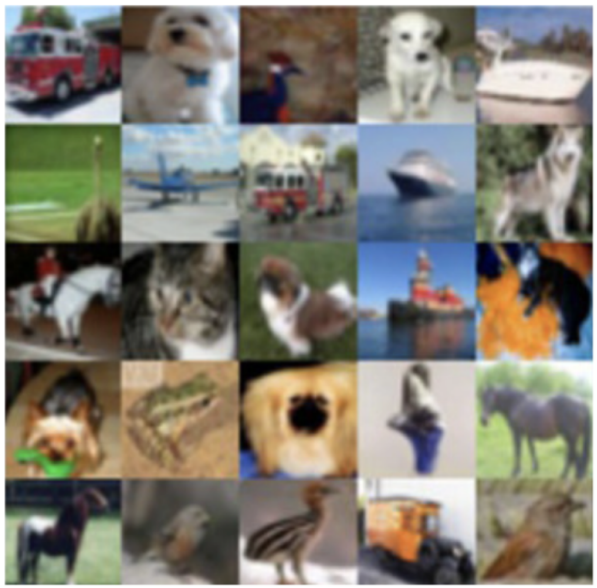
Sample Quality and Likelihood

- Optimal generative model will give best sample quality and highest test log-likelihood. However, in practice, high log-likelihoods $\neq$ good sample quality (Theis et al., 2016)
- Case 1: great test log-likelihoods, poor samples. Consider a mixture model $p_\theta(x) = 0.01 p_{\text{data}}(x) + 0.99 p_{\text{noise}}(x)$, we have

  $$\mathbb{E}_{p_{\text{data}}} \log p_{\text{data}}(x) \geq \mathbb{E}_{p_{\text{data}}} \log p_\theta(x) \geq \mathrm{E}_{p_{\text{data}}} \log p_{\text{data}}(x) - \log 100$$

  This means $\mathbb{E}_{p_{\text{data}}} \log p_\theta(x) \approx \mathrm{E}_{p_{\text{data}}} \log p_{\text{data}}(x)$ when the dimension of x is large.
- Case 2: great samples, poor test log-likelihoods. E.g., memorizing training set: samples look exactly like the training set; test set will have zero probability
- The above cases suggest that it might be useful to disentangle likelihoods and samples $\Rightarrow$ likelihood-free learning!

Comparing Distributions via Samples



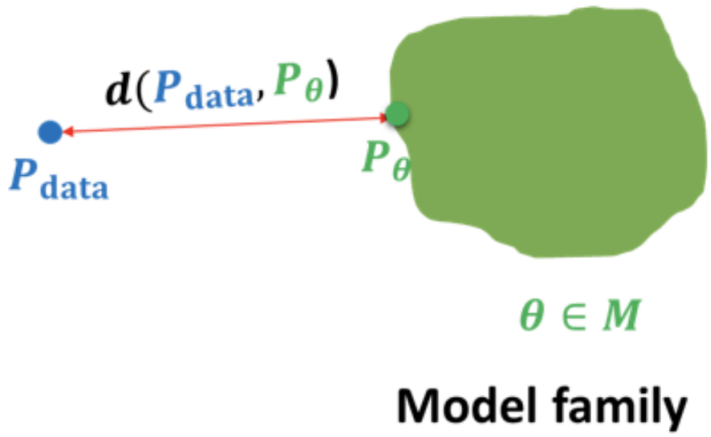
vs.

$S_1 = \{\mathbf{x} \sim P\}$ $S_2 = \{\mathbf{x} \sim Q\}$

Given samples from two distributions $S_1 = \{x \sim P\}$ and $S_2 = \{x \sim Q\}$, how can we tell if these samples are from the same distribution? (i.e., $P = Q$?)

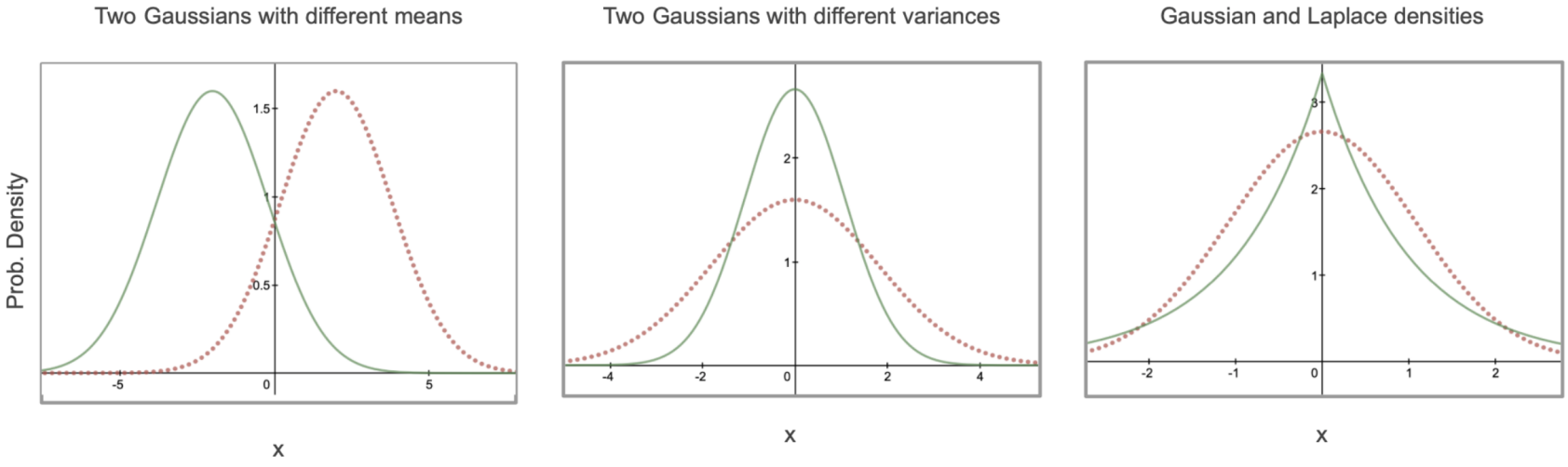
- Given $S_1 = \{x \sim P\}$ and $S_2 = \{x \sim Q\}$, a two-sample test considers the following hypotheses
  - Null hypothesis $H_0 : P = Q$
  - Alternative hypothesis $H_1 : p \neq Q$
- Test statistic T compares S_1 and S_2, e.g., difference in means, variances of the two sets of samples
- If T is less than a threshold α, the accept H_0 else reject it
- Key observation: Test statistics is likelihood-free since it does not involve the densities P or Q (only samples)

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



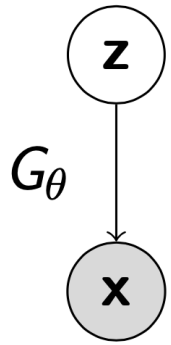
- Suppose we have direct access to the data set $S_1 = \mathcal{D} = \{x \sim p_{\text{data}}\}$
- Now assume that the model distribution p_θ permits efficient sampling (e.g., directed models). Let $S_2 = \{x \sim p_\theta\}$
- Use a two-sample test objective to measure the distance between distributions and train the generative model p_θ to minimize this distance between S_1 and S_2

Two-Sample Test via a Discriminator



- Finding a two-sample test objective in high dimensions is non-trivial
- In the generative model setup, we know that S_1 and S_2 come from different distributions p_{data} and p_θ respectively
- Key idea: Learn a statistic that maximizes a suitable notion of distance between the two sets of samples S_1 and S_2

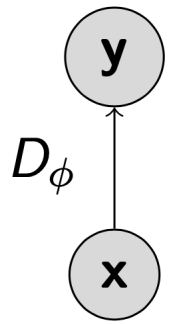
The **generator** and **discriminator** play a minimax game!



Generator

- Directed, latent variable model with a deterministic mapping between z and x given by G_θ
- Minimizes a two-sample test objective (in support of the null hypothesis $p_{\text{data}} = p_\theta$

The **generator** and **discriminator** play a minimax game!



Discriminator

- Any function (e.g., neural network) which tries to distinguish "real" samples from the dataset and "fake" sampels generated from the model
- Maximizes the two-sample test objectivee (in support of the alternative hypothesis $p_{\text{data}} \neq p_\theta$)

Discriminator Training Objective

- Training objective for discriminator:

$$\max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x \sim p_G} \log(1 - D(x))$$

- For a fixed generator G, the discriminator is performing binary classification with the cross entropy objective
  - Assign probability 1 to true data points $x \sim p_{\text{data}}$
  - Assign probability 0 to fake samples $x \sim p_G$
- Optimal discriminator

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

Generator Training Objective

- Training Objective for generator:

$$\min_G V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x\sim p_G} \log(1-D(x))$$

- For the optimal discriminator $D_G^*(\cdot)$, we have

$$\begin{aligned} V(G,D_G^*) &= \mathbb{E}_{x\sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)} + \mathbb{E}_{x\sim p_G} \log \frac{p_G(x)}{p_{\text{data}}(x)+p_G(x)} \\ &= \mathbb{E}_{x\sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x)+p_G(x)}{2}} + \mathbb{E}_{x\sim p_G} \log \frac{p_G(x)}{\frac{p_{\text{data}}(x)+p_G(x)}{2}} - \log 4 \\ &= \mathrm{KL}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}}+p_G}{2}\right) + \mathrm{KL}\left(p_G \middle\| \frac{p_{\text{data}}+p_G}{2}\right) - \log 4 \end{aligned}$$

- The sum of KL in the above equation is known as Jensen-Shannon divergence (JSD)

$$\mathrm{JSD}(p,q) = \mathrm{KL}\left(p \middle\| \frac{p+q}{2}\right) + \mathrm{KL}\left(q \middle\| \frac{p+q}{2}\right)$$

- Properties
  - $\mathrm{JSD}(p,q) \geq 0$
  - $\mathrm{JSD}(p,q) = 0$ iff $p = q$
  - $\mathrm{JSD}(p,q) = \mathrm{JSD}(q,p)$
  - $\sqrt{\mathrm{JSD}(p,q)}$ satisfies triangle inequality
- Optimal generator for the JSD GAN

$$p_G = p_{\text{data}}$$

- For the optimal discriminator $D^*_{G^*}(\cdot)$ and generator $G^*(\cdot)$, we have

$$V(G^*, D^*_{G^*}(x)) = -\log 4$$

Alternating Optimization in GAN

$$\min_{\theta}\max_{\phi} V(G_\theta, D_\phi) = \mathbb{E}_{x\sim p_{\text{data}}} \log D_\phi(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_\phi(G_\theta(z)))$$

- sample m training points $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ from $\mathcal{D}$
- sample m noise vectors $z^{(1)}, z^{(2)}, \ldots, z^{(m)}$ from p_z
- generator parameters θ update: stochastic gradient descent

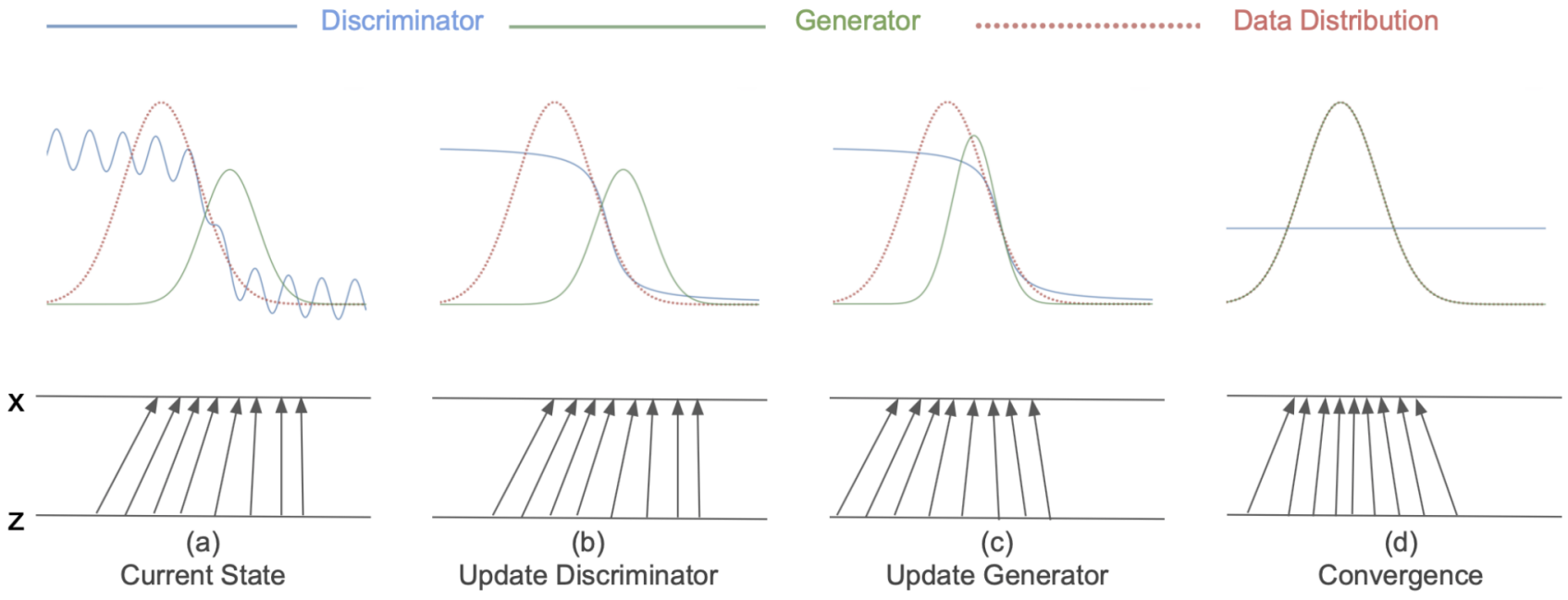
$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(z^{(i)})))$$

- discriminator parameters ϕ update: stochastic gradient ascent

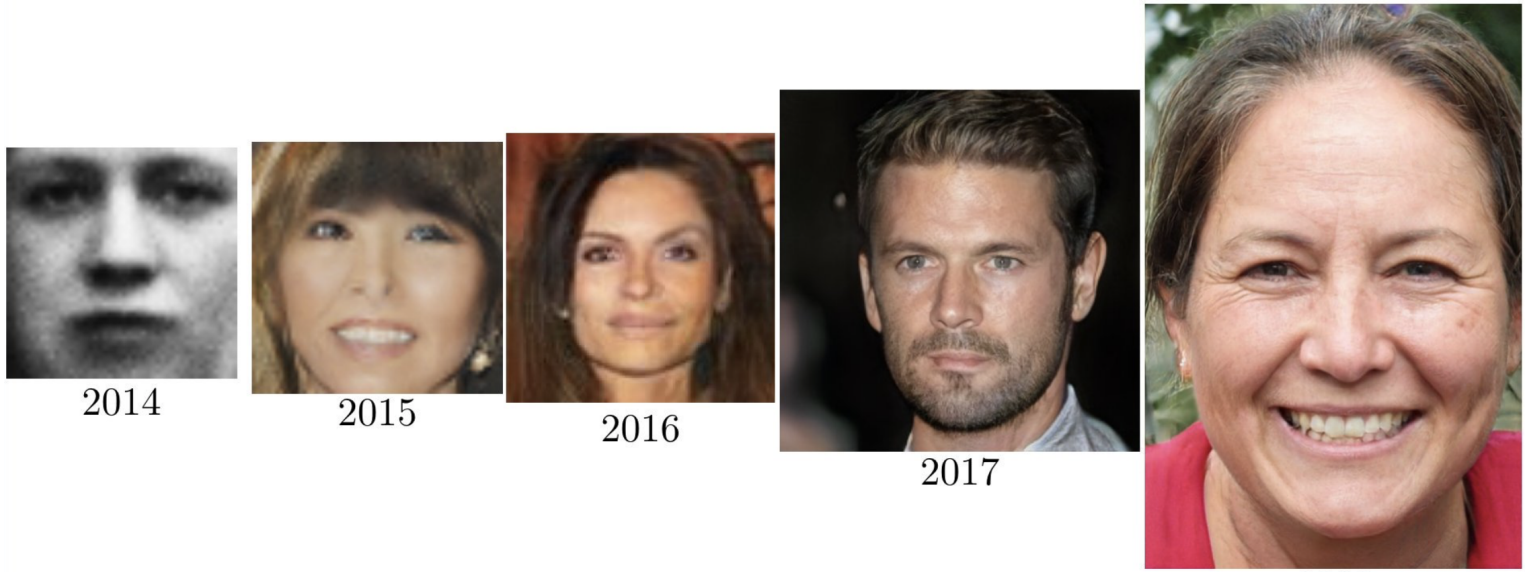
$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\phi \sum_{i=1}^{m} \log D_\phi(x^{(i)}) + \log(1 - D_\phi(G_\theta(z^{(i)})))$$

- Repeat for fixed number of epochs

A Toy Example



Adapted from Goodfellow, 2014

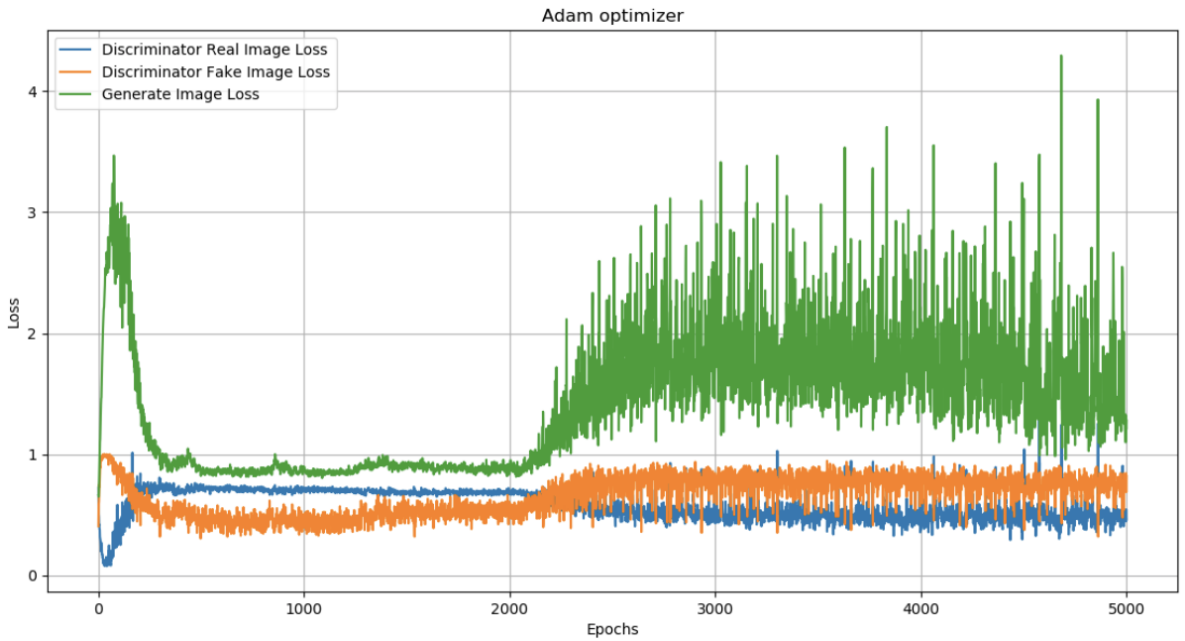


- GANs have been successfully applied to several domains and tasks
- However, working with GANs can be very challenging in practice: unstable optimization/mode collapse/evaluation
- Many bag of tricks applied to train GANs successfully

Image source: Ian Goodfellow. Samples from Goodfellow et al., 2014, Radford et al., 2015, Liu et al., 2016, Karras et al., 2017, Karras et al., 2018

Optimization Challenges

- Theorem: If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution
- Unrealistic assumptions! In practice, the generator and discriminator loss keeps oscillating during GAN training



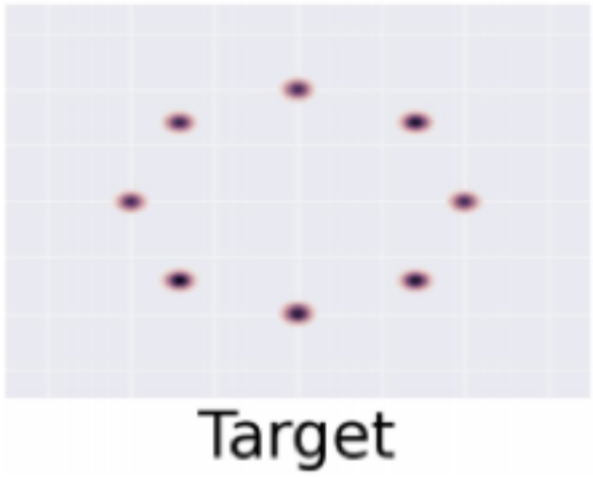
- No robust stopping criteria in practice (unlike MLE)

Mode Collapse

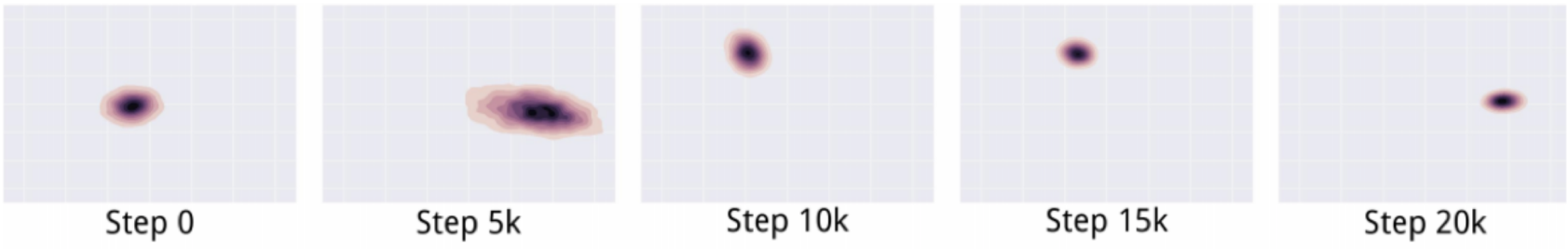
- GANs are notorious for suffering from mode collapse
- Intuitively, this refers to the phenomena where the generator of a GAN collapse to one or few samples (i.e., "modes")

Arjovsky et al., 2017

Mode Collapse



- True distribution is a mixture of Gaussians



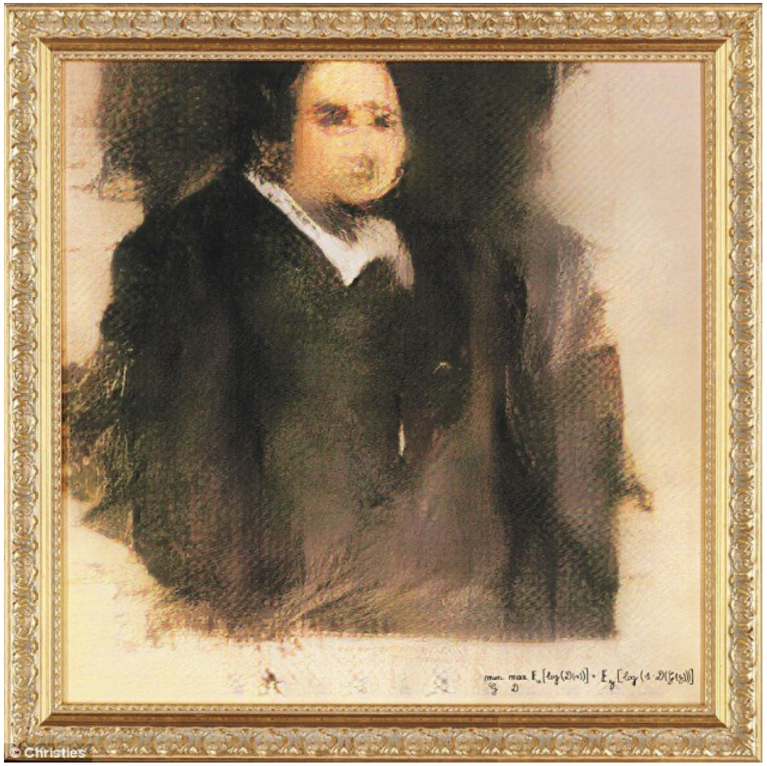
Source: Metz et al., 2017

- The generator distribution keeps oscillating between different models

Mode Collapse

Source: Metz et al., 2017

- Fixes to mode collapse are mostly empirically driven: alternate architectures, adding regularization terms, injecting small noise perturbations etc.
- Tips and tricks to make GAN work by Soumith Chintala: `https://github.com/soumith/ganhacks`



Source: Robbie Barrat, Obvious

GAN generated art auctioned at Christie's.
Expected Price: \$7,000 – \$10,000
True Price: \$432,500

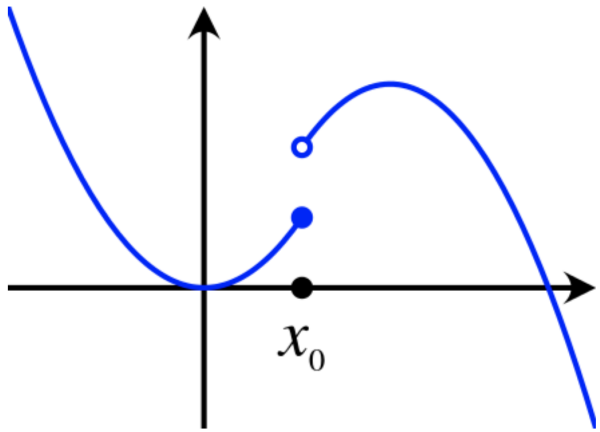
- The GAN Zoo:
  `https://github.com/hindupuravinash/the-gan-zoo`
- Examples
  - Rich class of likelihood-free objectives
  - Combination with latent representations
  - Application: Image-to-image translation, etc.

- Given two densities p and q, the $f-$ divergence is given by

$$D_f(p\|q) = \mathbb{E}_{x\sim q}\, f\left(\frac{p(x)}{q(x)}\right)$$

where f is any convex, lower-semicontinuous function with $f(1) = 0$

- Lower-semicontinuous: function value at any pint x_0 is close to $f(x_0)$ or greater than $f(x_0)$



- Example: KL divergence with $f(u) = u \log u$

Many more f-divergence!

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2 \mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)} + q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2} + u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)} + q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x - \log(4)$	$u\log u-(u+1)\log(u+1)$
α-divergence ($\alpha\notin\{0,1\}$)	$\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^{\alpha}-1\right]-\alpha(q(x)-p(x))\right)\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}\left(u^{\alpha}-1-\alpha(u-1)\right)$

Source: Nowozin et al., 2016

- To use f-divergences as a two-sample test objective for likelihood-free learning, we need to be able to estimate it only via samples
- Fenchel conjugate: For any function $f(\cdot)$, its convex conjugate is defined as

$$f^*(t) = \sup_{u \in \mathrm{dom}_f} ut - f(u)$$

- Duallity: $f^{**} = f$. When $f(\cdot)$ is convex, lower semicontinuous, so is $f^*(\cdot)$

$$f(u) = \sup_{t \in \mathrm{dom}_{f^*}} tu - f^*(t)$$

- We can obtain a lower bound to any f-divergence via its Fenchel conjugate

$$
\begin{aligned}
D_f(p\|q) &= \mathbb{E}_{x\sim q}\, f\left(\frac{p(x)}{q(x)}\right) \\
&= \mathbb{E}_{x\sim q} \sup_{t\in \mathrm{dom}_{f^*}} \left(t\frac{p(x)}{q(x)} - f^*(t)\right) \\
&\geq \mathbb{E}_{x\sim q}\, t(x)\frac{p(x)}{q(x)} - f^*(t(x)) \\
&= \int_{\mathcal{X}} t(x)p(x) - f^*(t(x))q(x)dx \\
&= \mathbb{E}_{x\sim p}\, t(x) - \mathbb{E}_{x\sim q}\, f^*(t(x))
\end{aligned}
$$

for any function $t : \mathcal{X} \mapsto \mathrm{dom}_{f^*}$

- Variational lower bound

$$D_f(p\|q) \geq \sup_{t\in\mathcal{T}}(\mathbb{E}_{x\sim p}\, t(x) - \mathbb{E}_{x\sim q}\, f^*(t(x)))$$

- Choose any f-divergence
- Let $p = p_{\text{data}}$ and $q = p_G$
- Parameterize t by ϕ and G by θ
- Consider the following f-GAN objective

$$\min_\theta \max_\phi F(\theta,\phi) = \mathbb{E}_{x\sim p_{\text{data}}}\, t_\phi(x) - \mathbb{E}_{x\sim p_{G_\theta}} f^*(t_\phi(x))$$

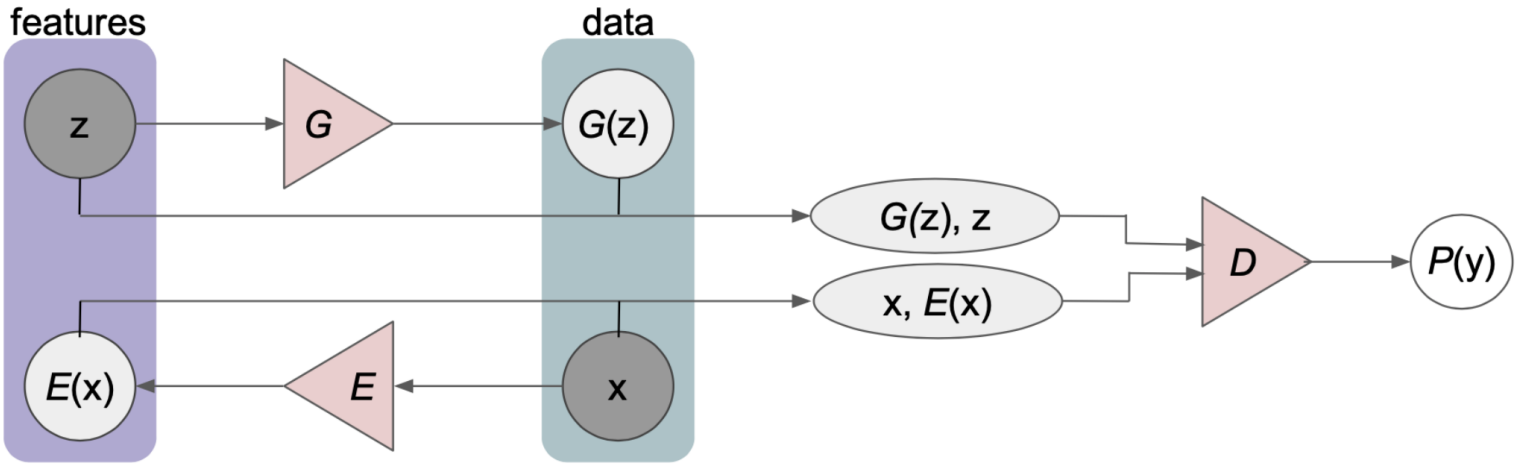
- Generator G_θ tries to minimize the divergence estimate and discriminator t_ϕ tries to tighten the lower bound

Inferring Latent Representation in GANs

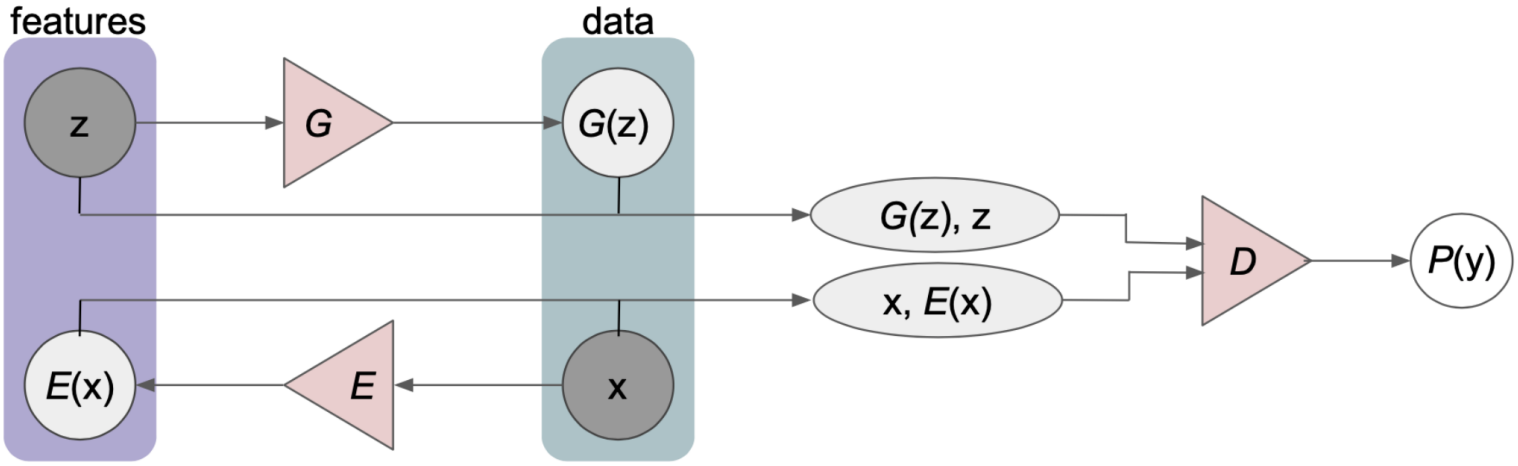
- The generator of a GAN is typically a directed, latent variable model with latent variable z and observed variables x. How can we infer the latent feature representations in a GAN?
- Unlike a normalizing flow model, the mapping $G : z \mapsto x$ need not to be invertible
- Unlike a variational autoencoder, there is no inference network $q(\cdot)$ which can learn a variational posterior over latent variables
- Solution 1: For any point x, use the activations of the prefinal layer of a discriminator as a feature representation
- Intuition: similar to supervised deep neural networks, the discriminator would have learned useful representations for x while distinguishing real and fake x

Inferring Latent Representation in GANs

- If we want to directly learn the latent representation of x, we need a different learning algorithm
- A regular GAN optimizes a two-sample test objective that compares samples of x from the generator and the data distribution
- Solution 2: To infer latent representations, we will compare samples of x, z from joint distributions of observed and latent variables as per the model and the data distribution
- For any x generated via the model, we have access to z (sampled from a simple prior $p(z)$)
- For any x from the data distribution, the z is however unobserved (latent)

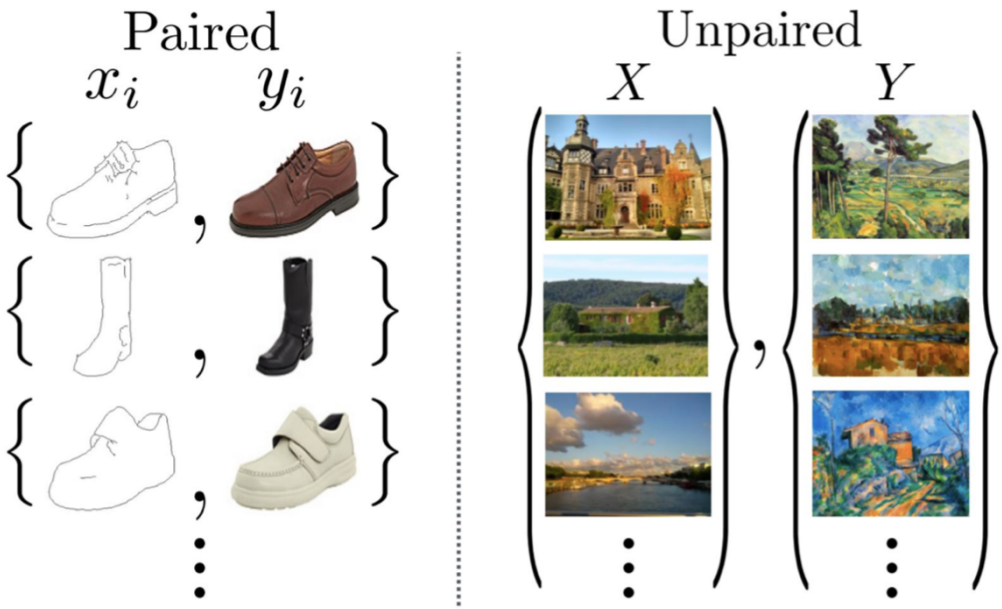


- In a BiGAN, we have an encoder network E in addtion to the generator network G
- The encoder network only observes $x \sim p_{\text{data}}(x)$ during training to learn a mapping $E : x \mapsto z$
- As before, the generator network only observes the samples from the prior $z \sim p(z)$ during training to learn a mapping $G : z \mapsto x$



- The discriminator D observes samples from the generative model $z, G(z)$ and encoding distribution $E(x), x$
- The goal of the discriminator is the maximize the two-sample test objective between $z, G(z)$ and $E(x), x$
- After training is complete, new samples are generated via G and latent representations are inferred via E

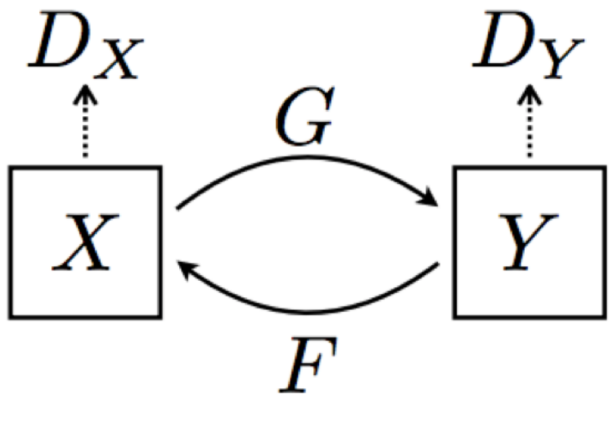
- Image-to-image translation: we are given image from two domains, $\mathcal{X}$ and $\mathcal{Y}$
- Paired vs. unpaired examples



Source: Zhu et al., 2016

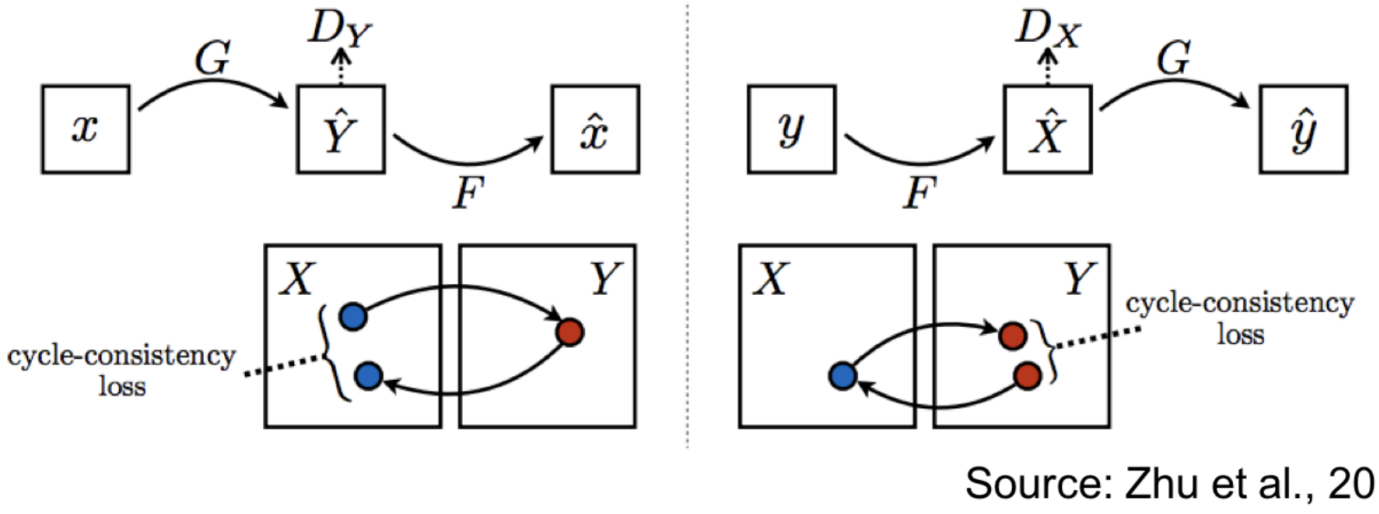
- Paired examples can be expensive to obtain. Can we translate from $\mathcal{X} \Leftrightarrow \mathcal{Y}$ in an unsupervised manner?

- To match the two distributions, we learn two parameterized conditional generative models $G : \mathcal{X} \mapsto \mathcal{Y}$ and $F : \mathcal{Y} \mapsto \mathcal{X}$
- G maps an element of $\mathcal{X}$ to an element of $\mathcal{Y}$. A discriminator $D_{\mathcal{Y}}$ compares the observed dataset Y and the generated samples $\hat{Y} = G(X)$
- Similarly, F maps an element of $\mathcal{Y}$ to an element of $\mathcal{X}$. A discriminator $D_{\mathcal{X}}$ compares the observed dataset X and the generated samples $\hat{X} = F(Y)$



Source: Zhu et al., 2016

- Cycle consistency: If we can go from X to $\hat{Y}$ via G, then it should also be possible to go from $\hat{Y}$ back to X via F
  - $F(G(X)) \approx X$
  - Similarly, vice versa: $G(F(Y)) \approx Y$

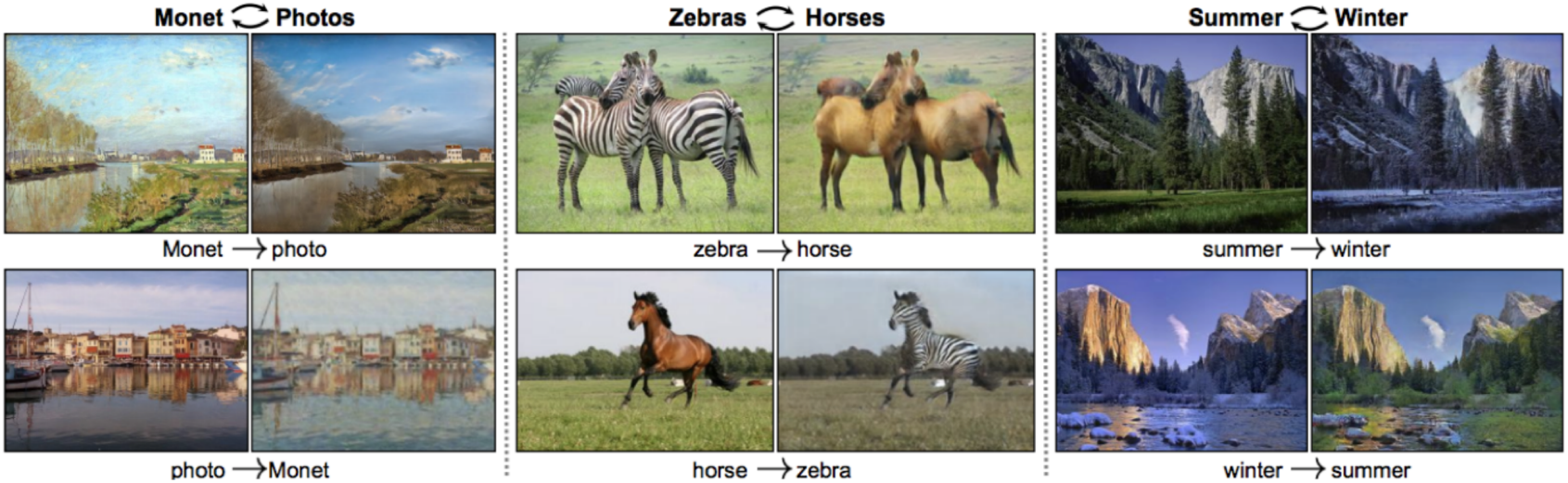


Source: Zhu et al., 2016

- Overall loss function

$$\mathcal{L}_{\text{GAN}}(G, D_{\mathcal{Y}}, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_{\mathcal{X}}, X, Y)$$
$$+\lambda(\mathbb{E}_X \|F(G(X)) - X\|_1 + \mathbb{E}_Y \|G(F(Y)) - Y\|_1)$$

CycleGAN in Practice



Source: Zhu et al., 2016

Summary of Generative Adversarial Networks

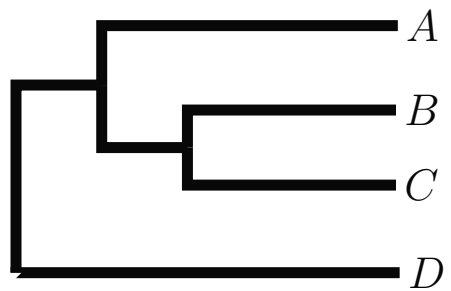
- Key observation: Samples and likelihoods are not correlated in practice
- Two-sample test objectives allow for learning generative mdoels only via samples (likelihood-free)
- Wide range of two-sample test objectives covering f-divergences (and more)
- Latent representations can be inferred via BiGAN (and other GANs with similar autoencoder structures)
- Cycle-consistent domain translations via CycleGAN and other variants

- While modern statistical approaches have been quite successful in many application areas, there are still challenging areas where the complex model structures make it difficult to apply those methods.
- In what follows, we will discuss some of the recent advancement on statistical approaches for computational biology, with an emphasis on evolutionary models.

The goal of **phylogenetic inference** is to reconstruct the evolution history (e.g., *phylogenetic trees*) from molecular sequence data (e.g., DNA, RNA or protein sequences)

Taxa	Characters
Species A	ATGAACAT
Species B	ATGCACAC
Species C	ATGCATAT
Species D	ATGCATGC

Molecular Sequence Data



Phylogenetic Tree

Lots of modern biological and medical applications: *predict the evolution of influenza viruses and help vaccine design, etc.*

BBC Sign in | Home | News | Sport | Reel | Worklife | T

NEWS

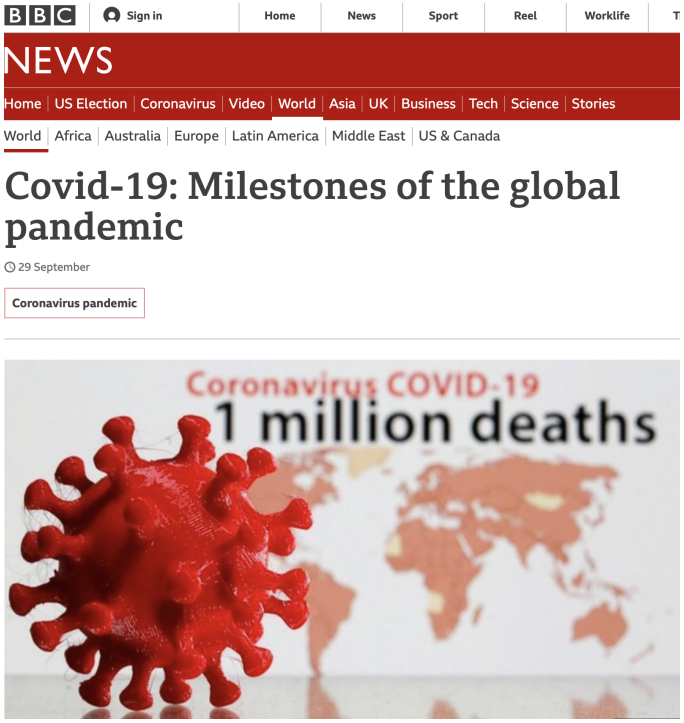
Home | US Election | Coronavirus | Video | World | Asia | UK | Business | Tech | Science | Stories

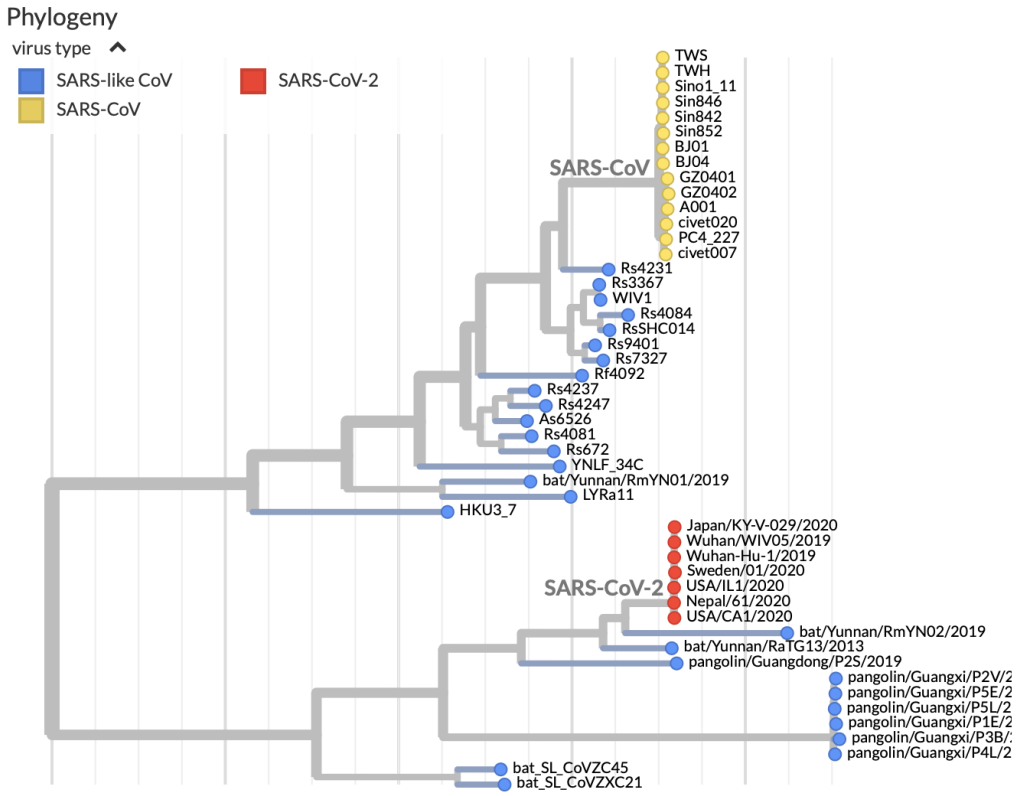
World | Africa | Australia | Europe | Latin America | Middle East | US & Canada

Covid-19: Milestones of the global pandemic

29 September

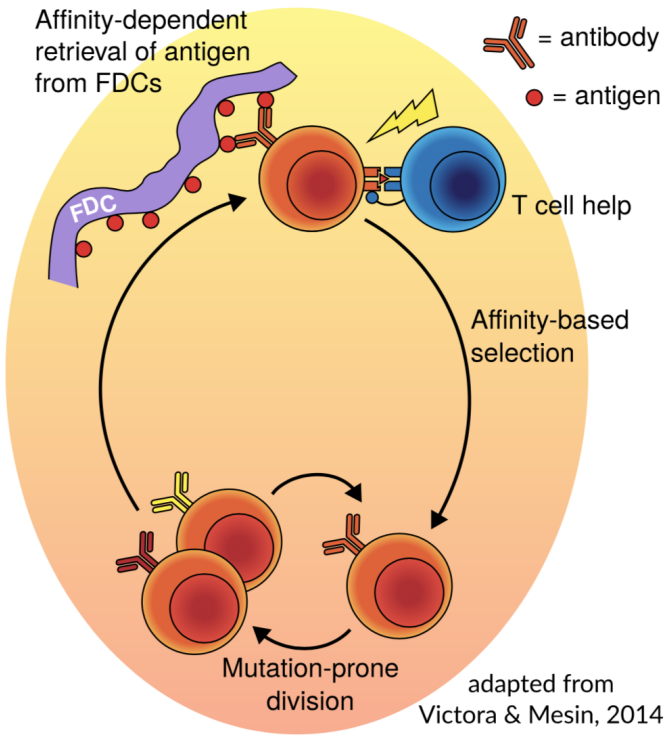
Coronavirus pandemic


BBC
Sign in
Home
News
Sport
Reel
Worklife
NEWS
Home | US Election | Coronavirus | Video | World | Asia | UK | Business | Tech | Science | Stories
World | Africa | Australia | Europe | Latin America | Middle East | US & Canada
Covid-19: Milestones of the global pandemic
29 September
Coronavirus pandemic
Coronavirus COVID-19
1 million deaths


Phylogeny
virus type
SARS-like CoV
SARS-CoV
SARS-CoV-2
SARS-CoV
TWS
TWH
Sino1_11
Sin846
Sin842
Sin852
BJ01
BJ04
GZ0401
GZ0402
A001
civet020
PC4_227
civet007
Rs4231
Rs3367
WIV1
Rs4084
RsSHC014
Rs9401
Rs7327
Rf4092
Rs4237
Rs4247
As6526
Rs4081
Rs672
YNLF_34C
bat/Yunnan/RmYN01/2019
LYRa11
HKU3_7
SARS-CoV-2
Japan/KY-V-029/2020
Wuhan/WIV05/2019
Wuhan-Hu-1/2019
Sweden/01/2020
USA/IL1/2020
Nepal/61/2020
USA/CA1/2020
bat/Yunnan/RmYN02/2019
bat/Yunnan/RaTG13/2013
pangolin/Guangdong/P2S/2019
pangolin/Guangxi/P2V/2
pangolin/Guangxi/P5E/2
pangolin/Guangxi/P5L/2
pangolin/Guangxi/P1E/2
pangolin/Guangxi/P3B/
pangolin/Guangxi/P4L/2
bat_SL_CoVZC45
bat_SL_CoVZXC21

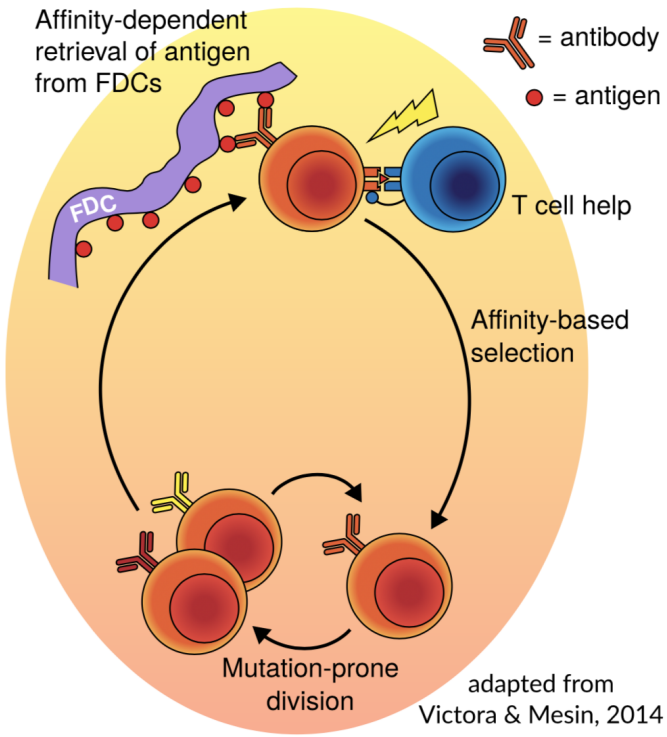
Example: B Cell Evolution

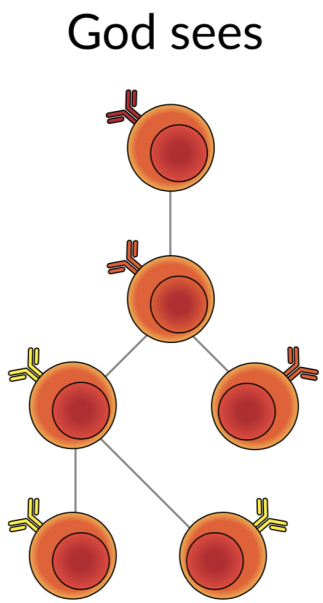
This happens inside of you!



Example: B Cell Evolution

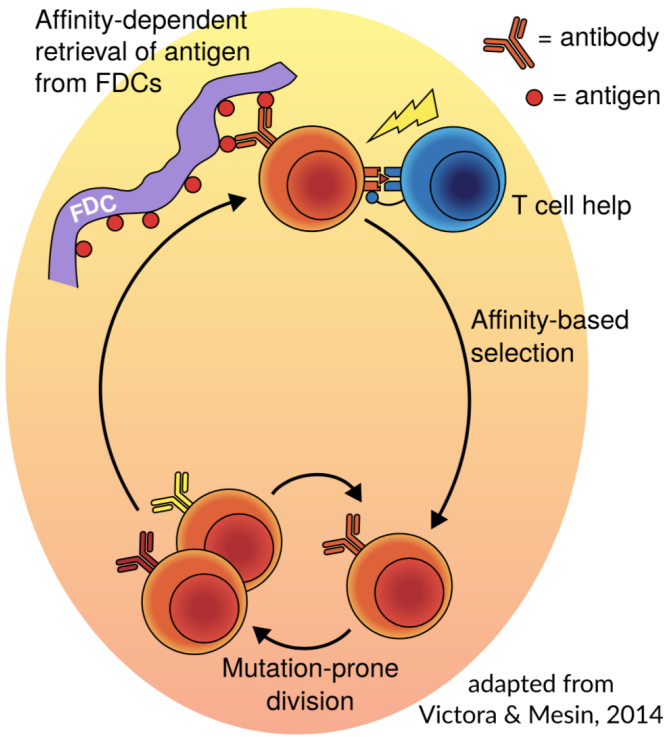
This happens inside of you!

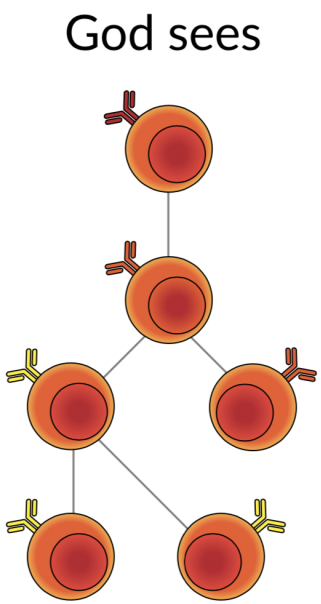


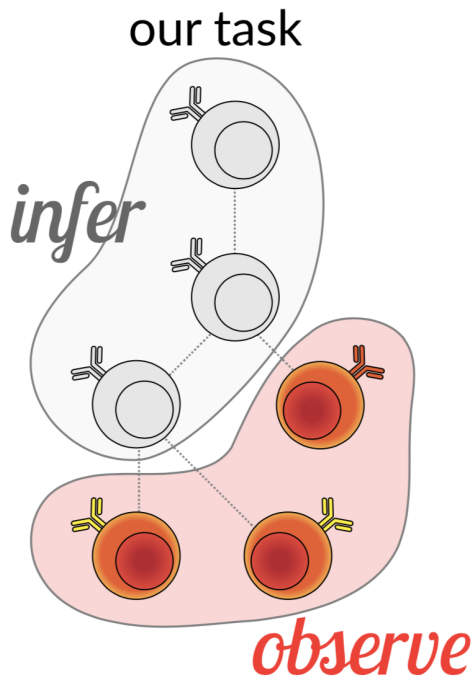


Example: B Cell Evolution

This happens inside of you!

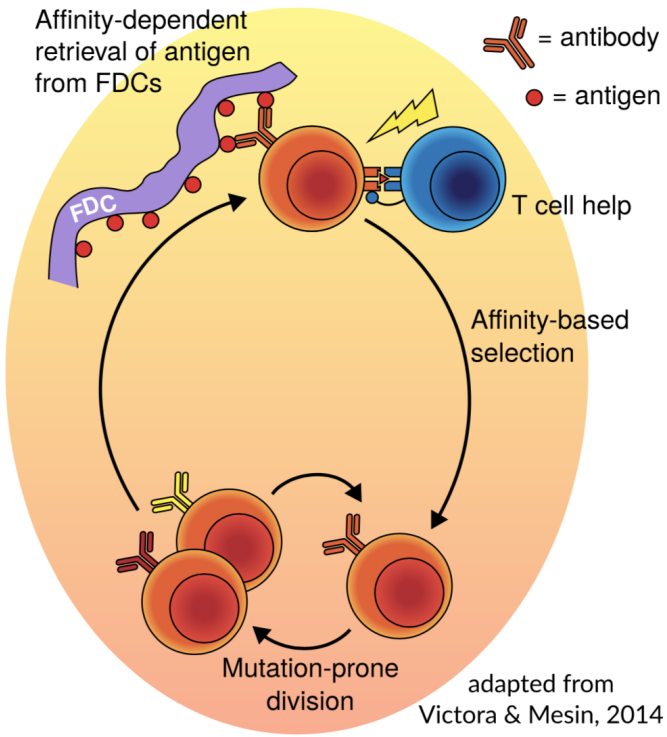


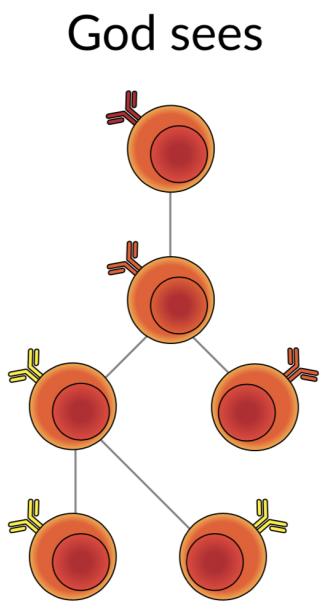


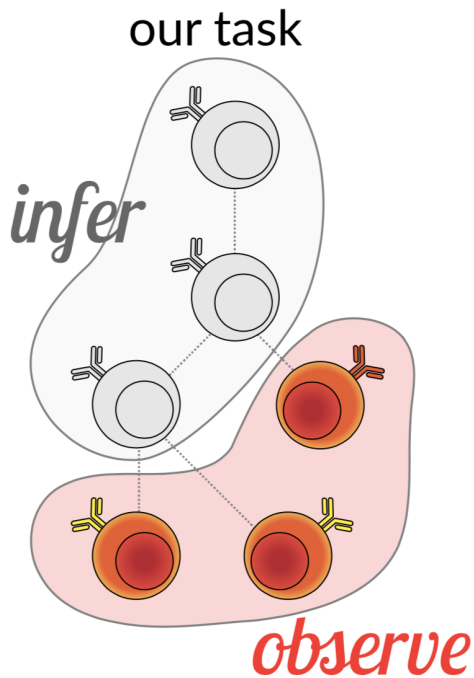


Example: B Cell Evolution

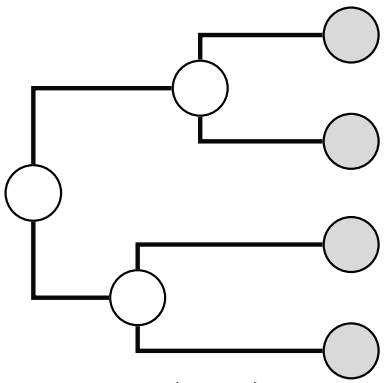
This happens inside of you!







These inferences guide rational vaccine design.



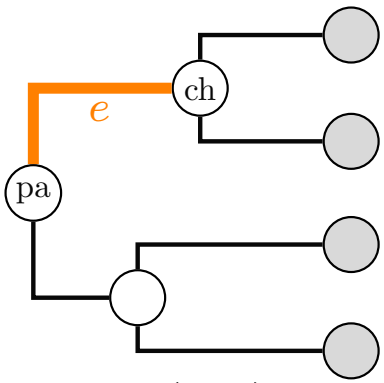
A T G A A C $\cdots$

A T G C A C $\cdots$

A T G C A T $\cdots$

A T G C A T $\cdots$

$y_1\, y_2\, y_3\, y_4\, y_5\, y_6$



$(\tau, \boldsymbol{q})$

A T G A A C $\cdots$

A T G C A C $\cdots$

A T G C A T $\cdots$

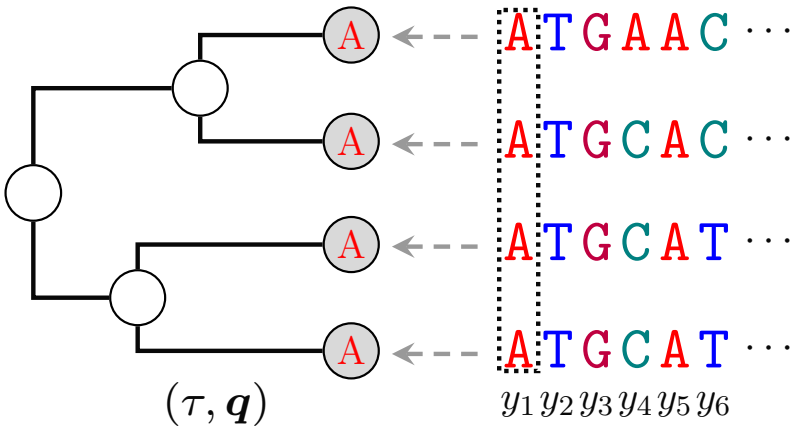
A T G C A T $\cdots$

$y_1\, y_2\, y_3\, y_4\, y_5\, y_6$

Evolution model:

$$p(\text{ch}|\text{pa}, q_e)$$

q_e: amount of evolution on e.



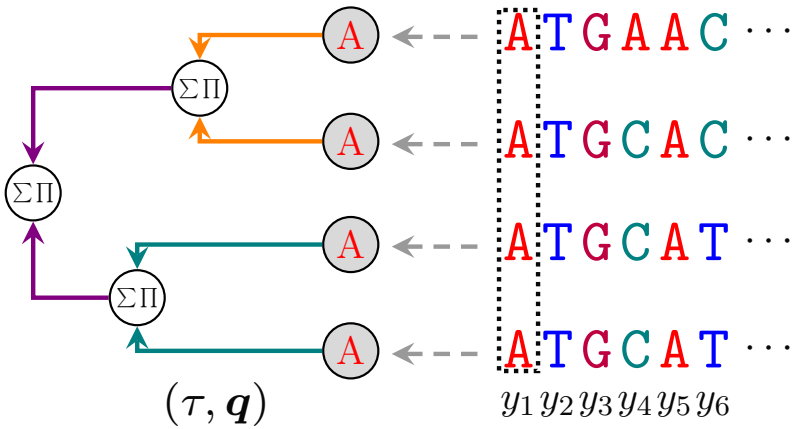
Evolution model:

$$p(\text{ch}|\text{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \eta(a_\rho^i) \prod_{(u,v)\in E(\tau)} P_{a_u^i a_v^i}(q_{uv})$$



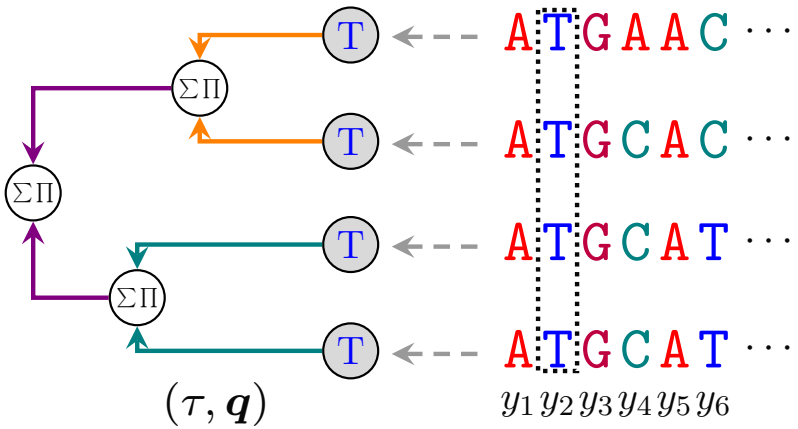
Evolution model:

$$p(\mathrm{ch}|\mathrm{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$



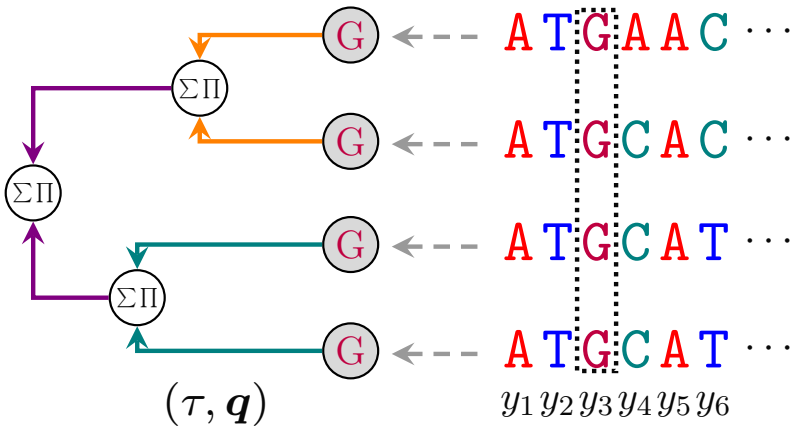
Evolution model:

$$p(\mathrm{ch}|\mathrm{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$



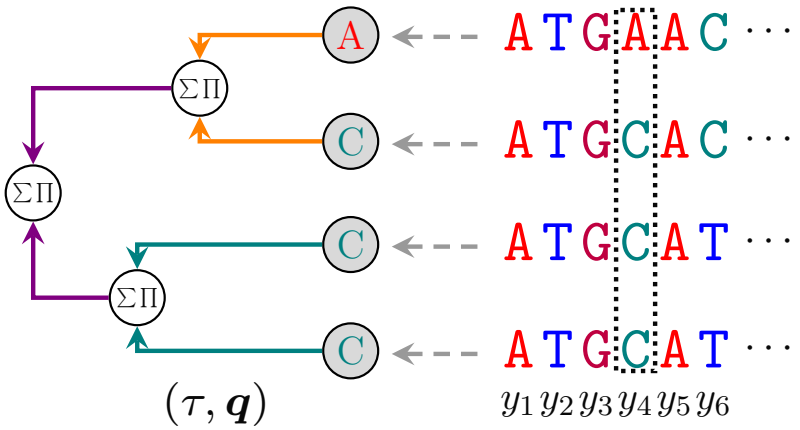
Evolution model:

$$p(\mathrm{ch}|\mathrm{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$



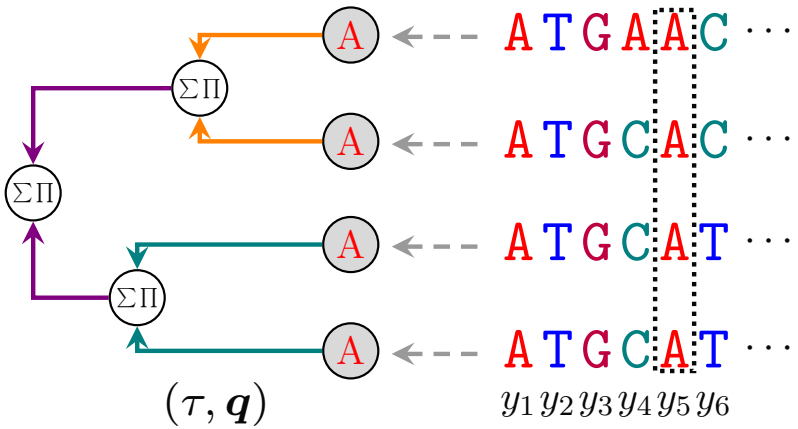
Evolution model:

$$p(\text{ch}|\text{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$



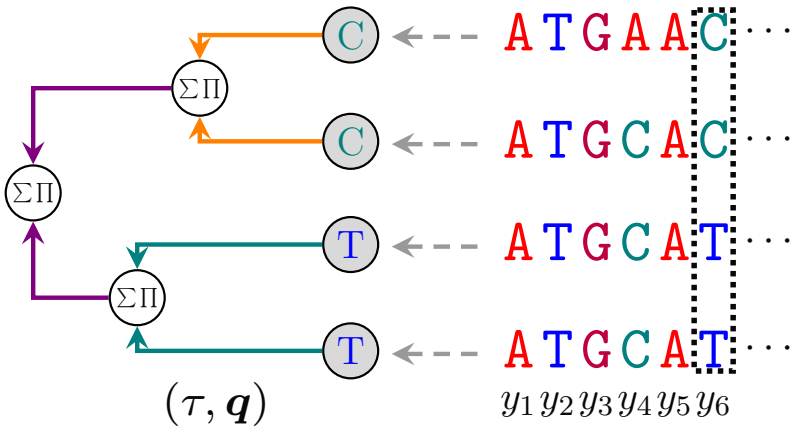
Evolution model:

$$p(\mathrm{ch}|\mathrm{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a_\rho^i) \prod_{(u,v)\in E(\tau)} P_{a_u^i a_v^i}(q_{uv})$$



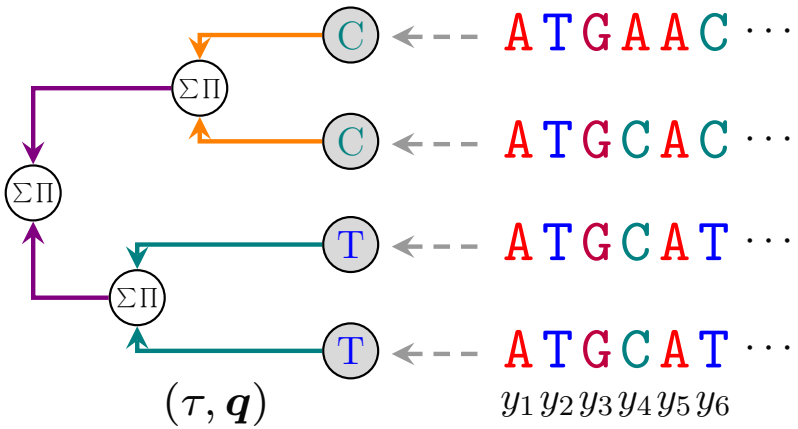
Evolution model:

$$p(\mathrm{ch}|\mathrm{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$



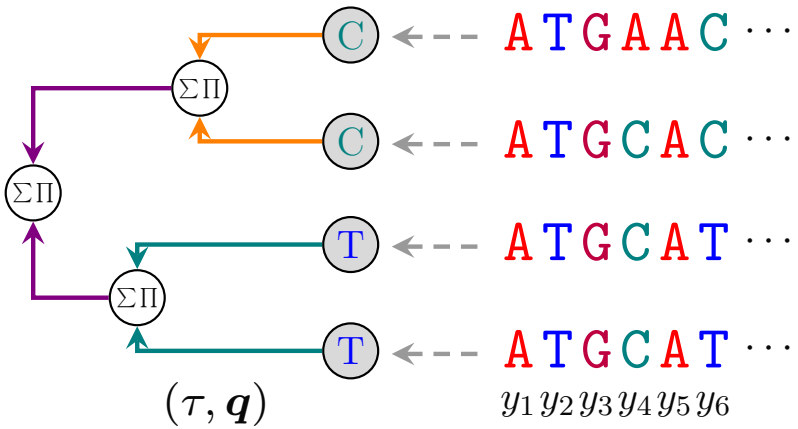
Evolution model:

$$p(\text{ch}|\text{pa}, q_e)$$

q_e: amount of evolution on e.

Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$



Evolution model:

$$p(\text{ch}|\text{pa}, q_e)$$

q_e: amount of evolution on e.

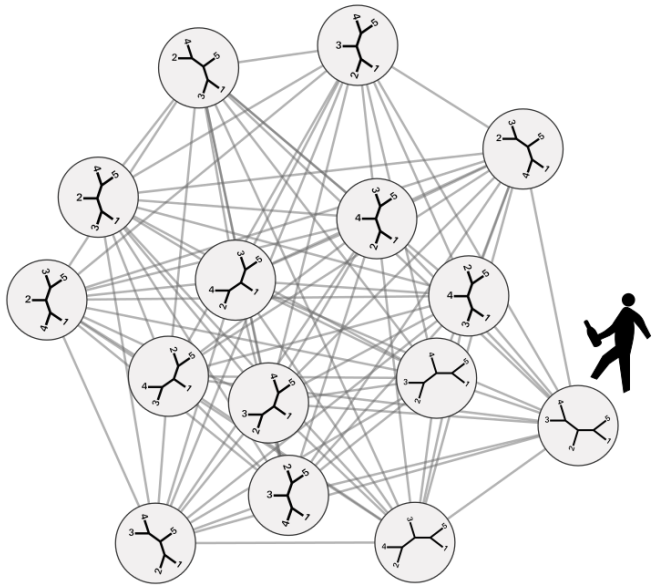
Likelihood

$$p(\boldsymbol{Y}|\tau, \boldsymbol{q}) = \prod_{i=1}^{M} \sum_{a^i} \eta(a^i_\rho) \prod_{(u,v)\in E(\tau)} P_{a^i_u a^i_v}(q_{uv})$$

Given a proper prior distribution $p(\tau, \boldsymbol{q})$, the posterior is

$$p(\tau, \boldsymbol{q}|\boldsymbol{Y}) \propto p(\boldsymbol{Y}|\tau, \boldsymbol{q}) p(\tau, \boldsymbol{q}).$$

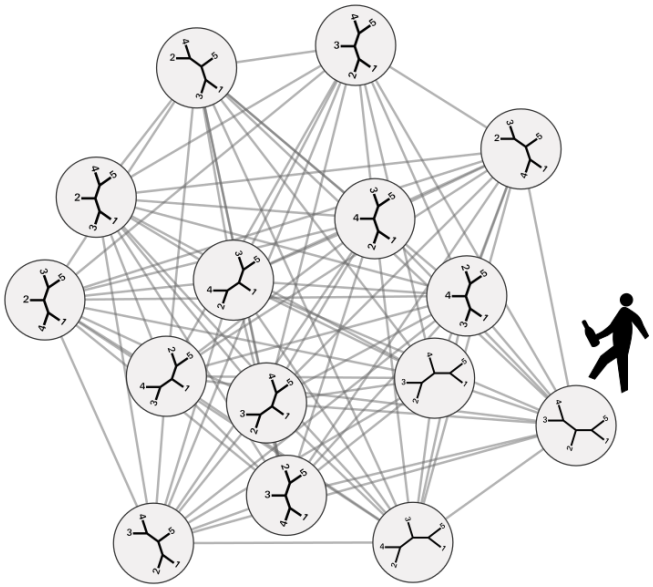
Random-walk MCMC (MrBayes, BEAST):



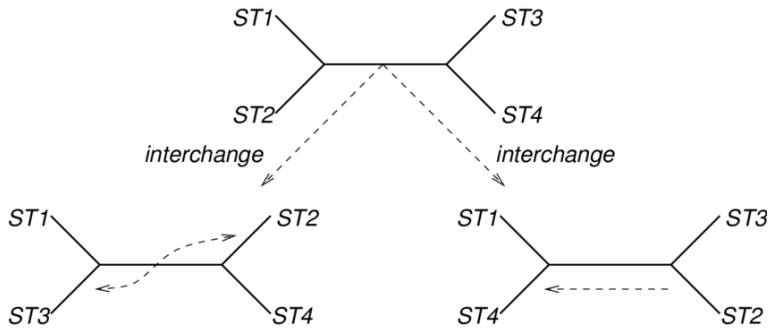
Challenges for MCMC

- Large search space: $(2n-5)!!$ unrooted trees (n taxa)

Random-walk MCMC (MrBayes, BEAST):



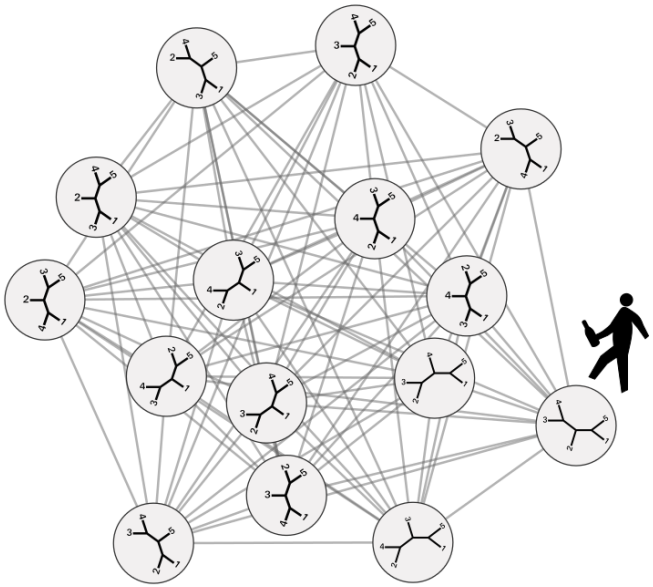
local tree transform

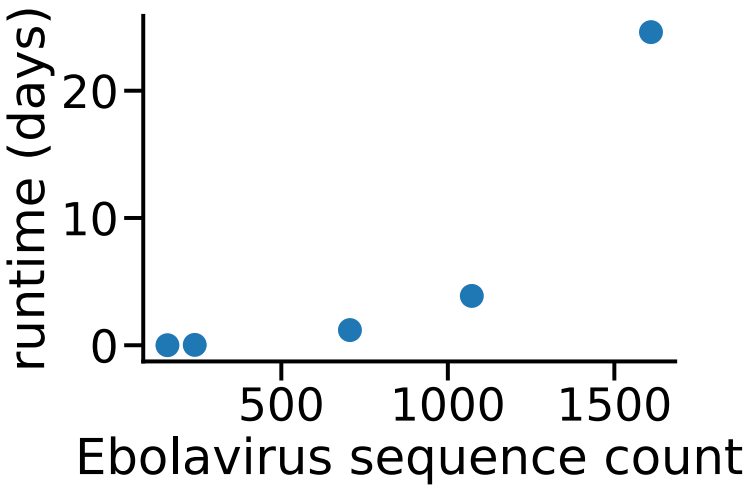


Challenges for MCMC

- Large search space: $(2n-5)!!$ unrooted trees (n taxa)
- Intertwined parameter space, low acceptance rate, hard to scale to data sets with many sequences.

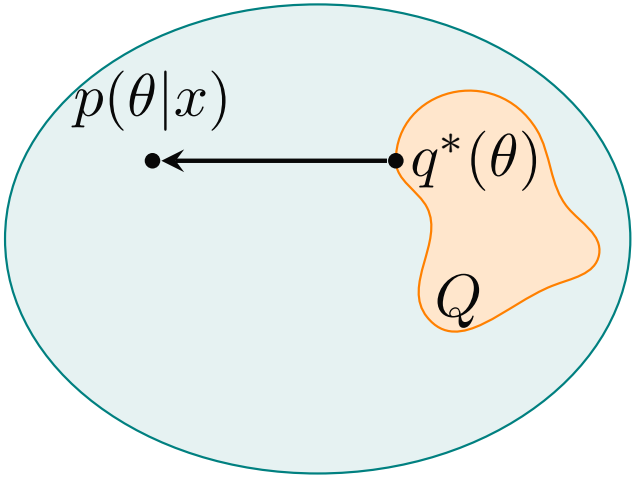
Random-walk MCMC (MrBayes, BEAST):





Challenges for MCMC

- Large search space: $(2n-5)!!$ unrooted trees (n taxa)
- Intertwined parameter space, low acceptance rate, hard to scale to data sets with many sequences.



$$q^*(\theta) = \arg\min_{q \in Q} \mathrm{KL}\left(q(\theta) \| p(\theta|x)\right)$$

$$= \arg\min_{q \in Q} \mathbb{E}_{q(\theta)} \log \frac{p(x,\theta)}{q(\theta)}$$

- VI turns inference into optimization
- Specify a variational family of distributions over the model parameters

$$Q = \{q_\phi(\theta); \phi \in \Phi\}$$

- Fit the variational parameters ϕ to minimize the distance (often in terms of KL divergence) to the exact posterior

- Approximating Distribution:

$$Q_{\boldsymbol{\phi},\boldsymbol{\psi}}(\tau, \boldsymbol{q}) \triangleq \underbrace{Q_{\boldsymbol{\phi}}(\tau)}_{\text{tree topology}} \cdot \underbrace{Q_{\boldsymbol{\psi}}(\boldsymbol{q}|\tau)}_{\text{branch length}}$$

- Multi-sample Lower Bound:

$$L^K(\boldsymbol{\phi}, \boldsymbol{\psi}) = \mathbb{E}_{Q_{\boldsymbol{\phi},\boldsymbol{\psi}}(\tau^{1:K}, \boldsymbol{q}^{1:K})} \log \left(\frac{1}{K} \sum_{i=1}^{K} \frac{p(\boldsymbol{Y}|\tau^i, \boldsymbol{q}^i) p(\tau^i, \boldsymbol{q}^i)}{Q_{\boldsymbol{\phi}}(\tau^i) Q_{\boldsymbol{\psi}}(\boldsymbol{q}^i|\tau^i)} \right)$$

- Use **stochastic gradient ascent** (SGA) to maximize the lower bound:

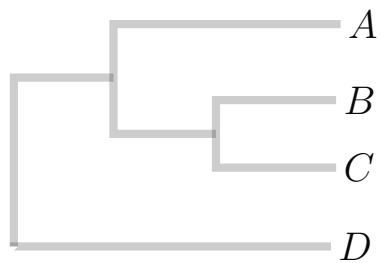
$$\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}} = \arg\max_{\boldsymbol{\phi},\boldsymbol{\psi}} \; L^K(\boldsymbol{\phi}, \boldsymbol{\psi})$$

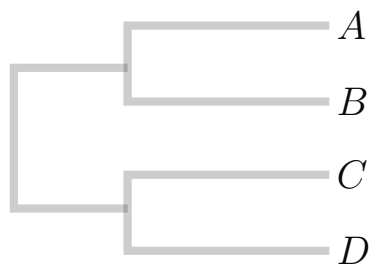
Stochastic gradient estimators for $\boldsymbol{\phi}, \boldsymbol{\psi}$

ϕ : VIMCO/RWS, ψ : The Reparameterization Trick

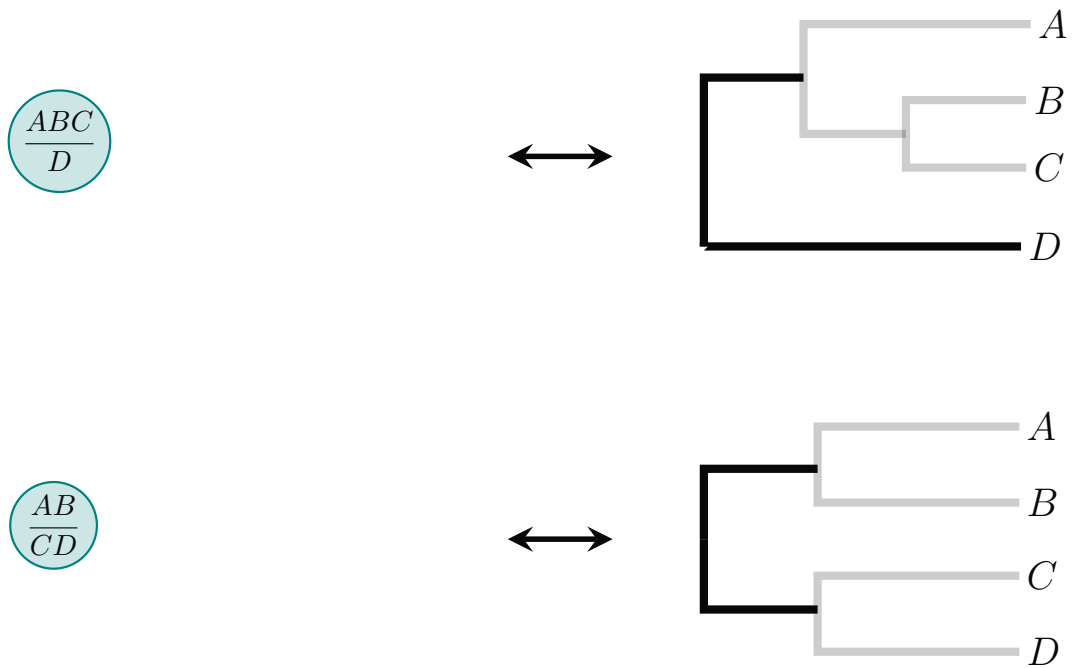
[Zhang and Matsen IV, ICLR 2019]

Inspired by previous works (Höhna and Drummond 2012, Larget 2013), we can decompose trees into local structures and encode the tree topology space via Bayesian networks! [Zhang and Matsen IV, NeurIPS 2018]

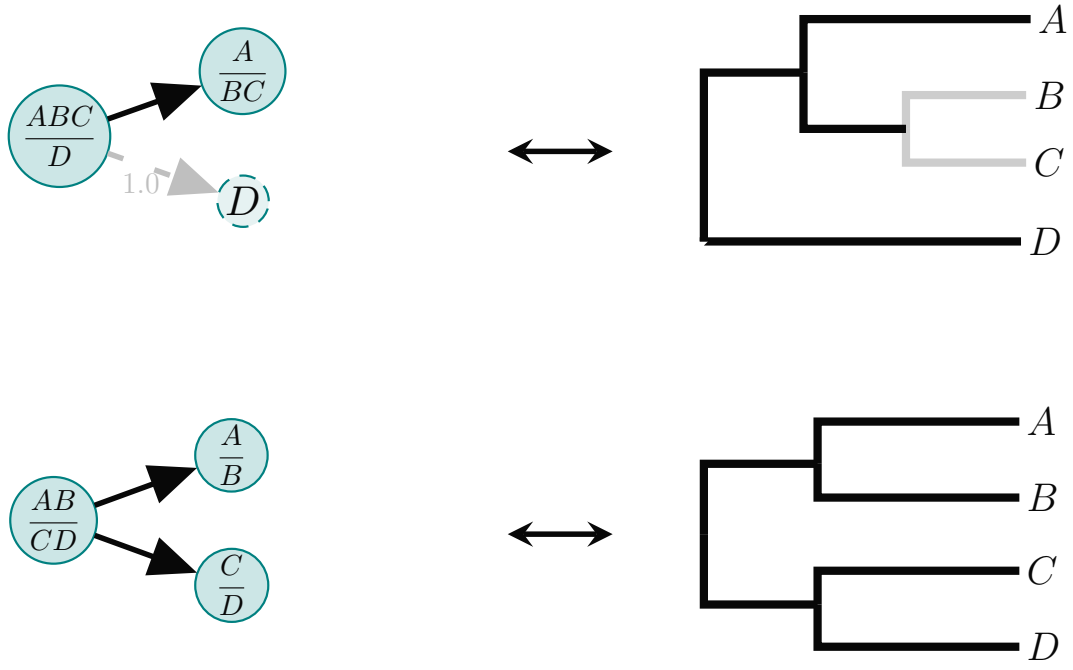




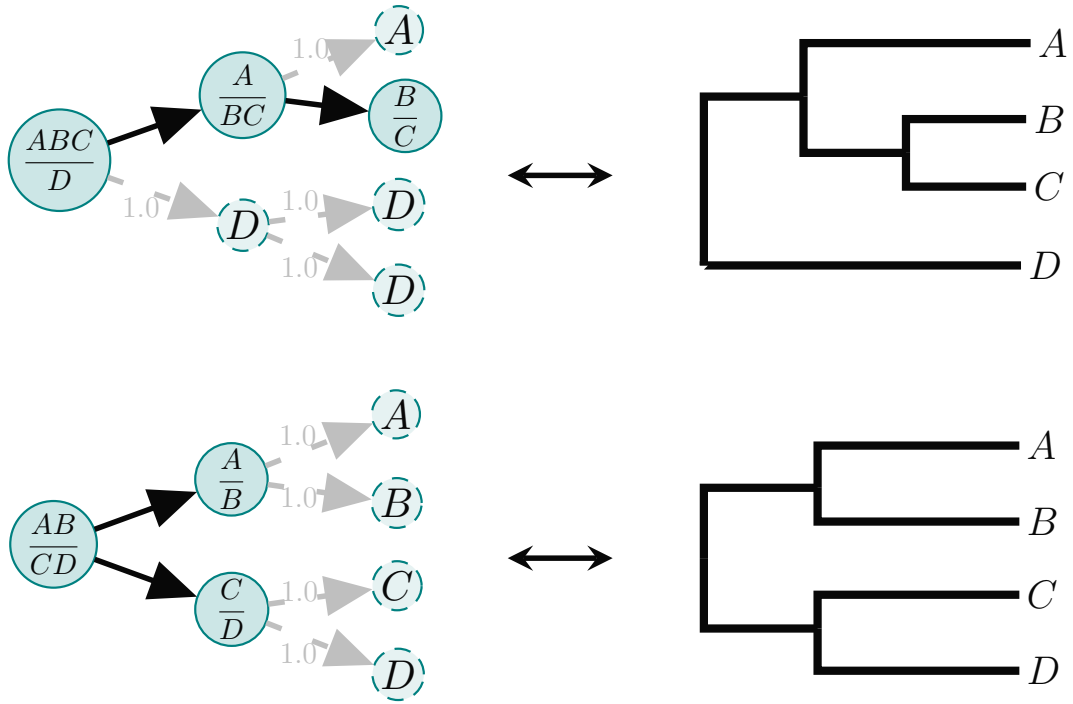
Inspired by previous works (Höhna and Drummond 2012, Larget 2013), we can decompose trees into local structures and encode the tree topology space via Bayesian networks! [Zhang and Matsen IV, NeurIPS 2018]



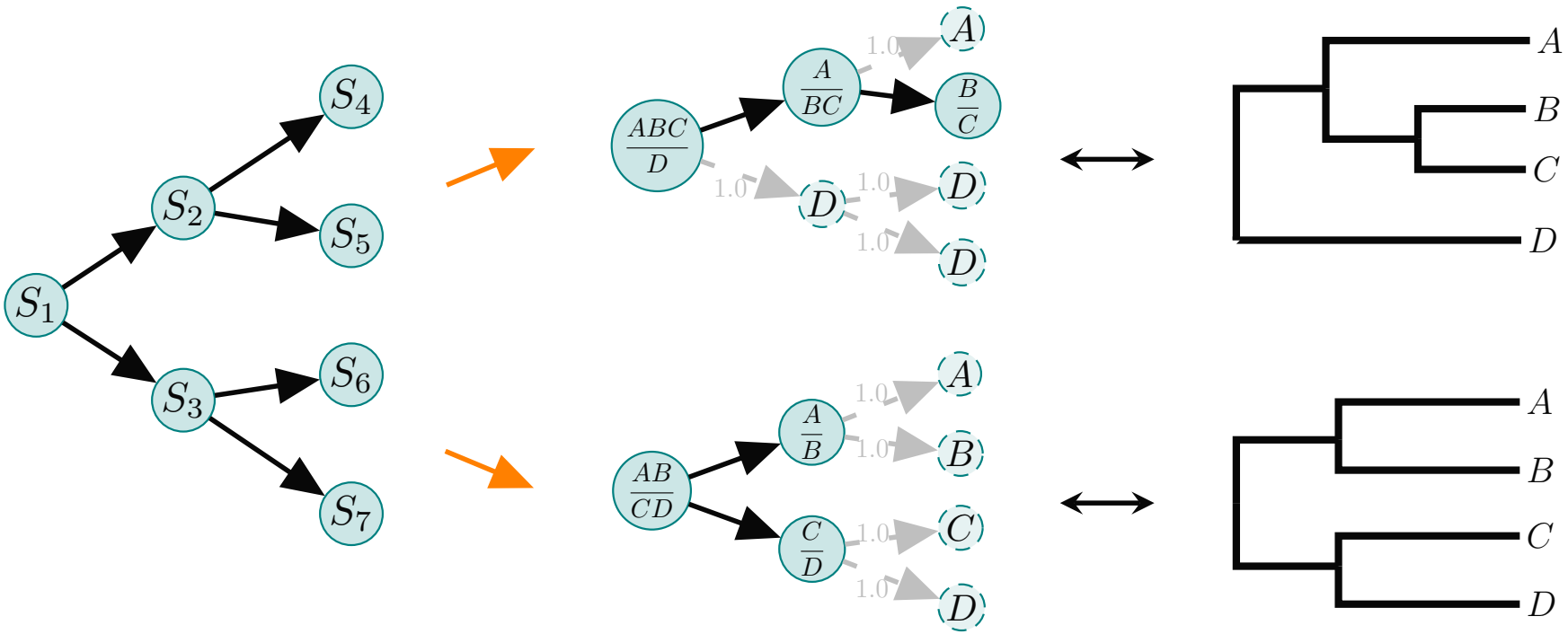
Inspired by previous works (Höhna and Drummond 2012, Larget 2013), we can decompose trees into local structures and encode the tree topology space via Bayesian networks! [Zhang and Matsen IV, NeurIPS 2018]

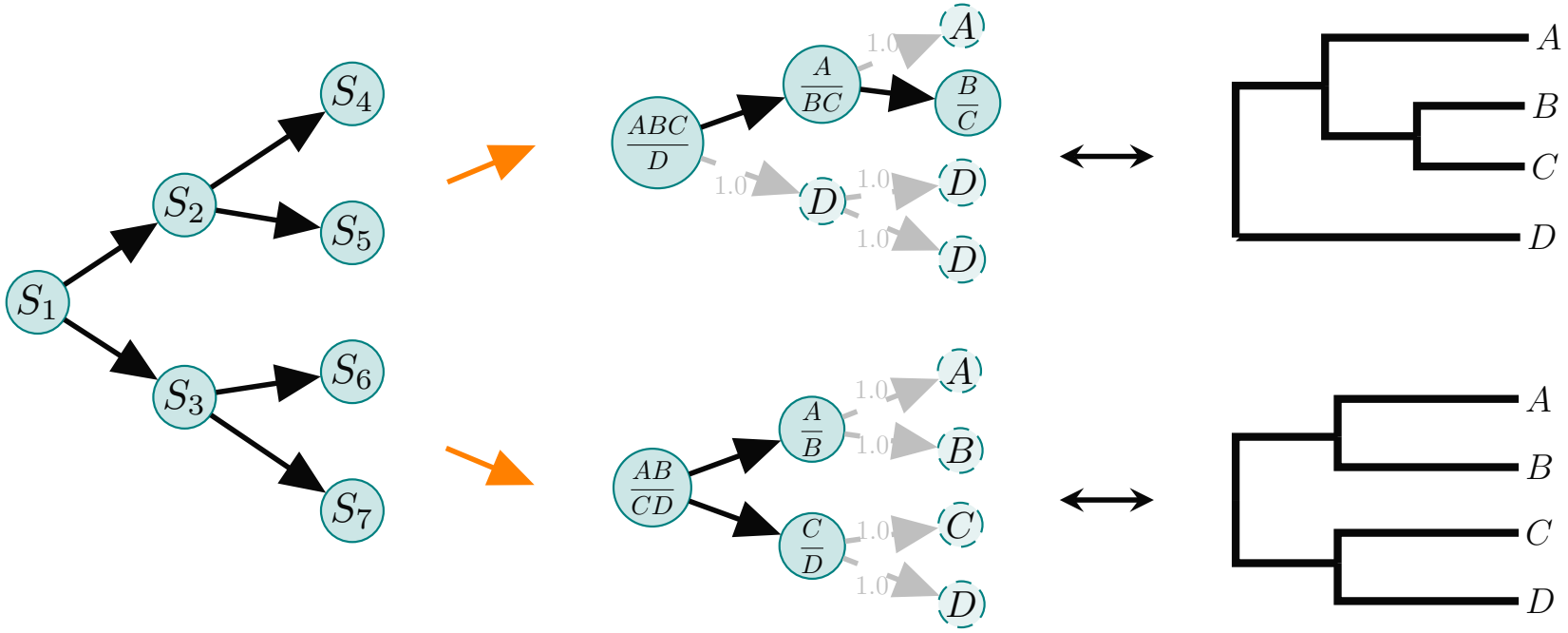


Inspired by previous works (Höhna and Drummond 2012, Larget 2013), we can decompose trees into local structures and encode the tree topology space via Bayesian networks! [Zhang and Matsen IV, NeurIPS 2018]



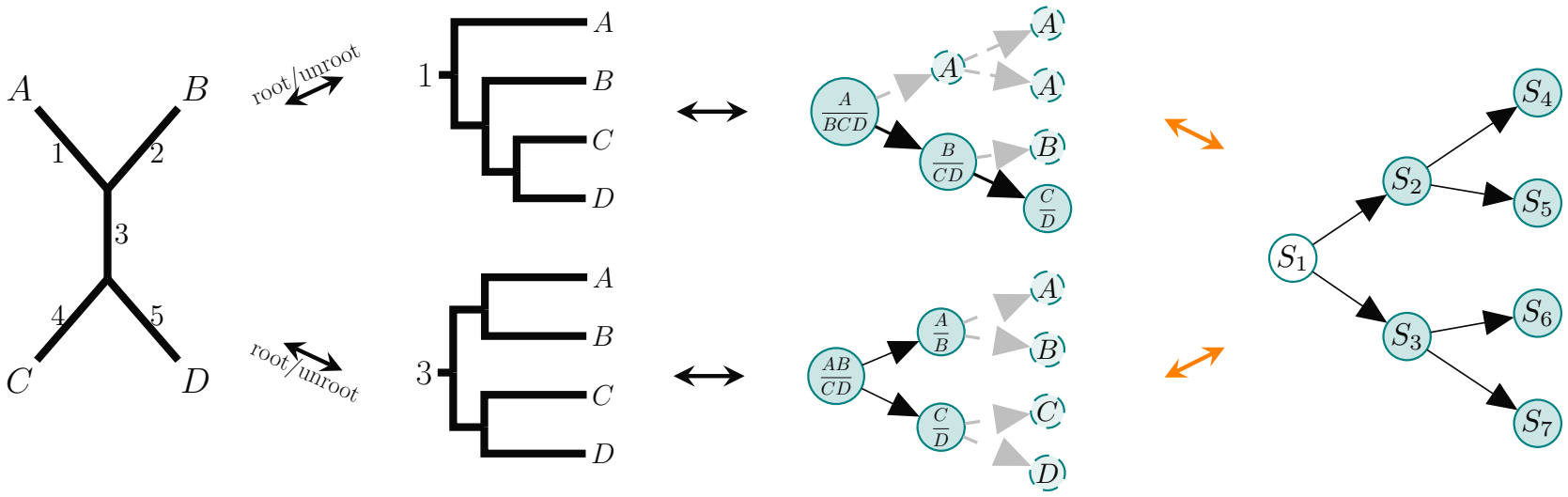
Inspired by previous works (Höhna and Drummond 2012, Larget 2013), we can decompose trees into local structures and encode the tree topology space via Bayesian networks! [Zhang and Matsen IV, NeurIPS 2018]





Rooted Trees

$$p_{\text{sbn}}(T=\tau) = p(S_1 = s_1) \prod_{i>1} p(S_i = s_i | S_{\pi_i} = s_{\pi_i}).$$



Unrooted Trees:

$$p_{\text{sbn}}(T^{\text{u}} = \tau) = \sum_{s_1 \sim \tau} p(S_1 = s_1) \prod_{i>1} p(S_i = s_i | S_{\pi_i} = s_{\pi_i}).$$

Remark: can use a two-pass algorithm, computation cost is only doubled compared to rooted trees.

SBNs can be used to learn a probability distribution based on a collection of trees $T = \{T_1, \cdots, T_K\}$.

$$T_k = \{S_i = s_{i,k},\ i \geq 1\}, \quad k = 1, \ldots, K$$

Rooted Trees

- **Maximum Likelihood Estimates**: relative frequencies.

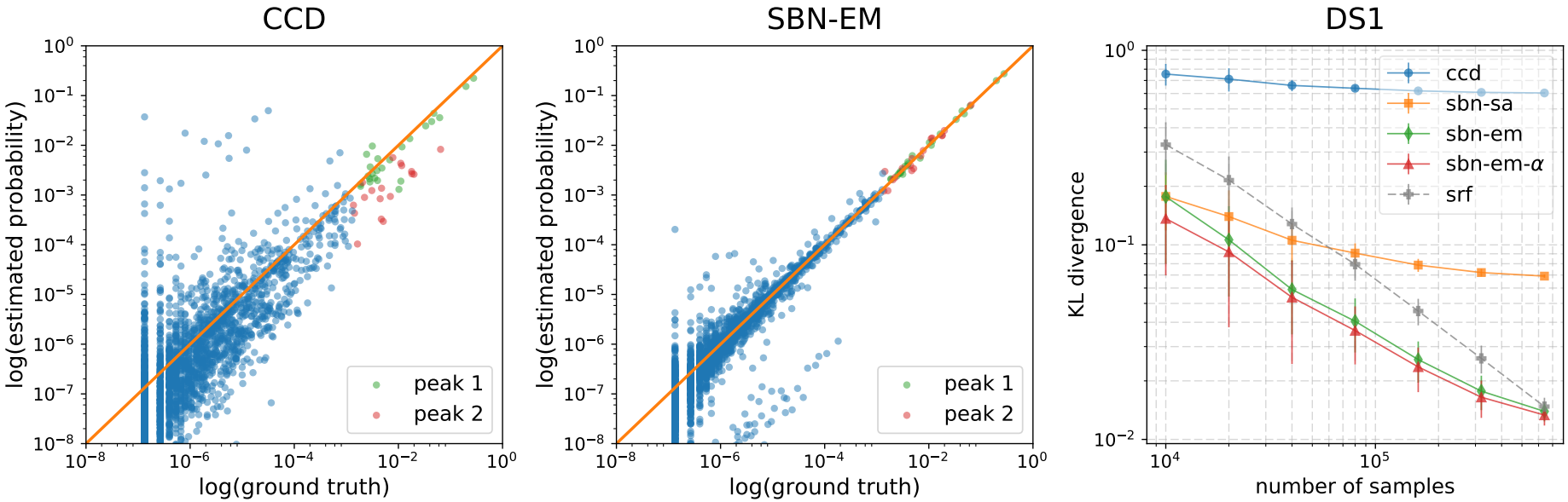
$$\hat{p}^{\mathrm{MLE}}(S_1 = s_1) = \frac{m_{s_1}}{K}, \quad \hat{p}^{\mathrm{MLE}}(S_i = s_i | S_{\pi_i} = t_i) = \frac{m_{s_i,t_i}}{\sum_{s \in \mathbb{C}_i} m_{s,t_i}}$$

Unrooted Trees

- **Expectation Maximization**

$$\hat{p}^{\mathrm{EM},(n+1)} = \arg\max_p \mathbb{E}_{p(S_1|T,\hat{p}^{\mathrm{EM},(\mathrm{n})})} \left(\log p(S_1) + \sum_{i>1} \log p(S_i|S_{\pi_i}) \right)$$

Example: Phylogenetic Posterior Estimation



[Zhang and Matsen, NeurIPS 2018]

- Compared to a previous method CCD (Larget, 2013), SBNs significantly reduce the biases for both high probability and low probability trees.
- SBNs perform better in the weak data regime.

Example: Phylogenetic Posterior Estimation

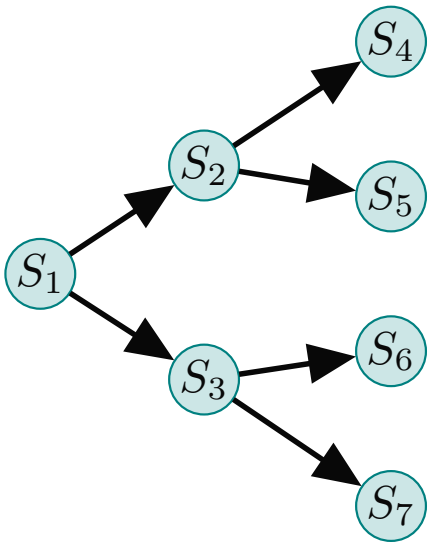
Data set	(#Taxa, #Sites)	Tree space size	Sampled trees	KL divergence to ground truth				
				SRF	CCD	**SBN-SA**	**SBN-EM**	**SBN-EM-α**
DS1	(27, 1949)	5.84×10^{32}	1228	0.0155	0.6027	0.0687	0.0136	**0.0130**
DS2	(29, 2520)	1.58×10^{35}	7	**0.0122**	0.0218	0.0218	0.0199	0.0128
DS3	(36, 1812)	4.89×10^{47}	43	0.3539	0.2074	0.1152	0.1243	**0.0882**
DS4	(41, 1137)	1.01×10^{57}	828	0.5322	0.1952	0.1021	0.0763	**0.0637**
DS5	(50, 378)	2.84×10^{74}	33752	11.5746	1.3272	0.8952	0.8599	**0.8218**
DS6	(50, 1133)	2.84×10^{74}	35407	10.0159	0.4526	**0.2613**	0.3016	0.2786
DS7	(59, 1824)	4.36×10^{92}	1125	1.2765	0.3292	0.2341	0.0483	**0.0399**
DS8	(64, 1008)	1.04×10^{103}	3067	2.1653	0.4149	0.2212	0.1415	**0.1236**

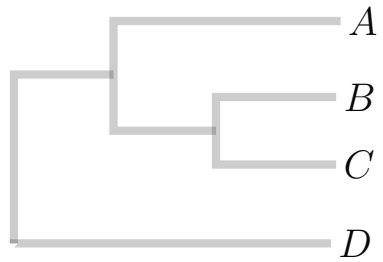
[Zhang and Matsen, NeurIPS 2018]

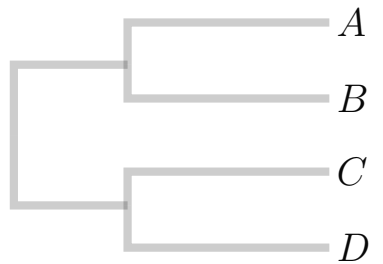
Remark: Unlike previous methods, SBNs are flexible enough to provide accurate approximations to real data posteriors!

Rooted Trees: ancestral sampling

Rooted Trees: ancestral sampling

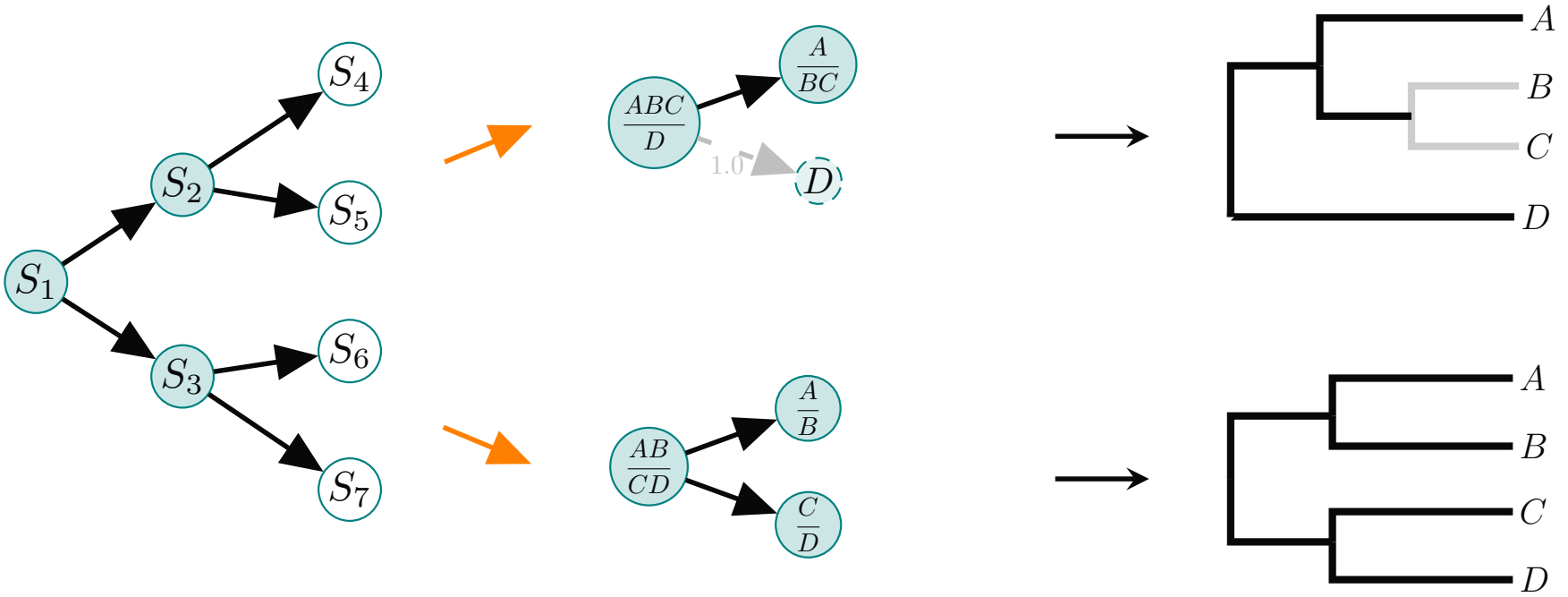




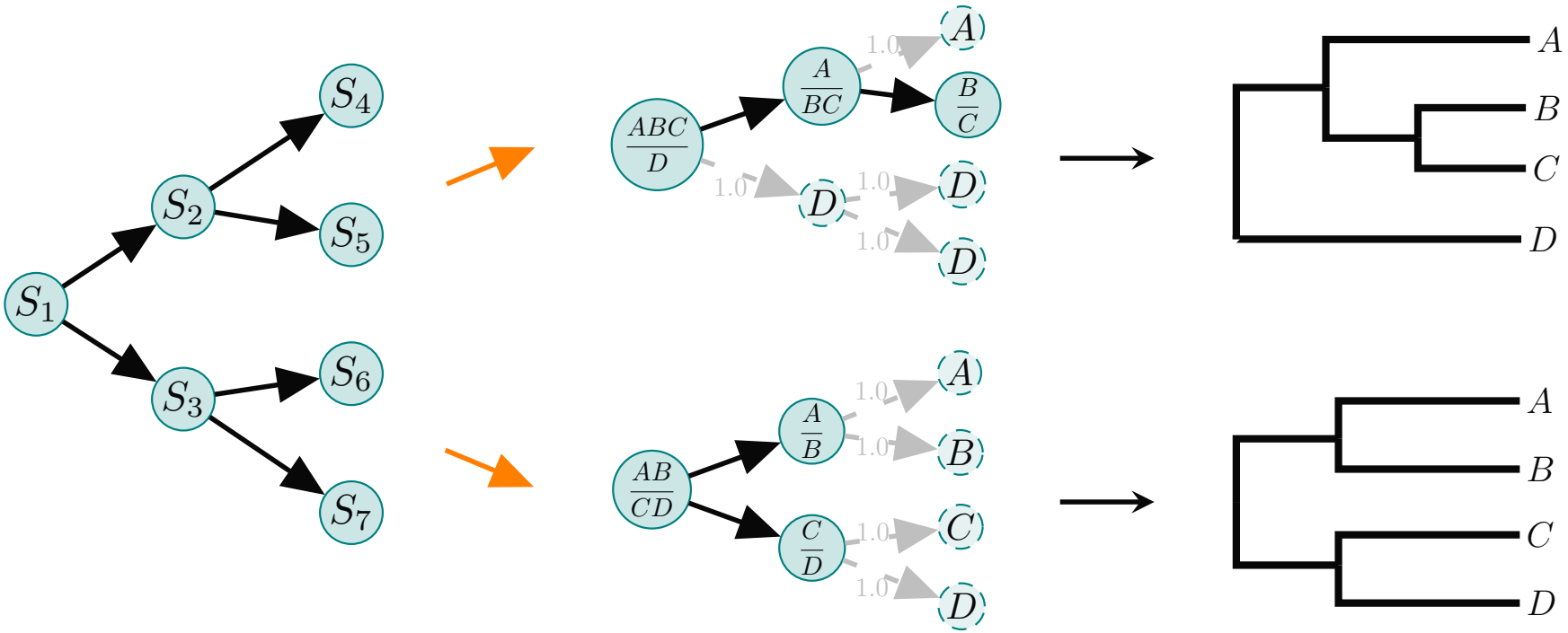


Rooted Trees: ancestral sampling

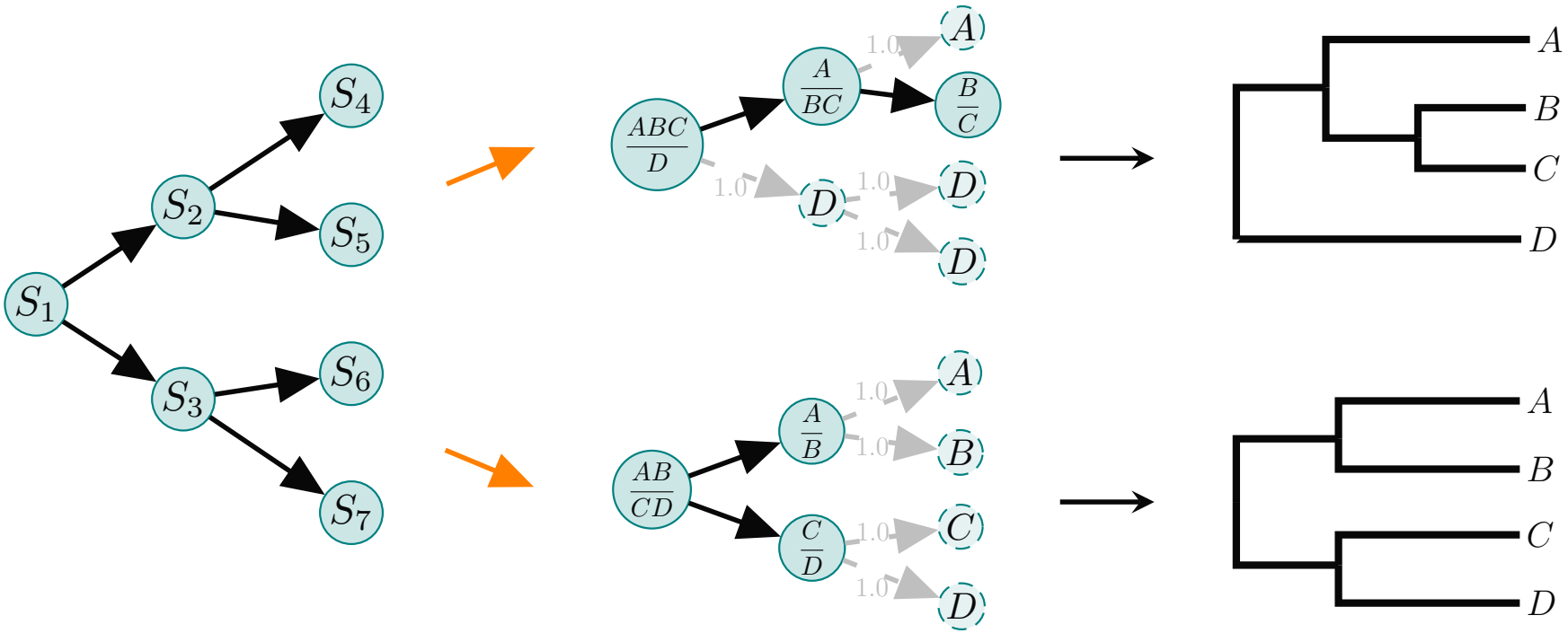
Rooted Trees: ancestral sampling



Rooted Trees: ancestral sampling



Rooted Trees: ancestral sampling



Unrooted Trees: sample as rooted trees, then remove the roots

SBNs Parameters

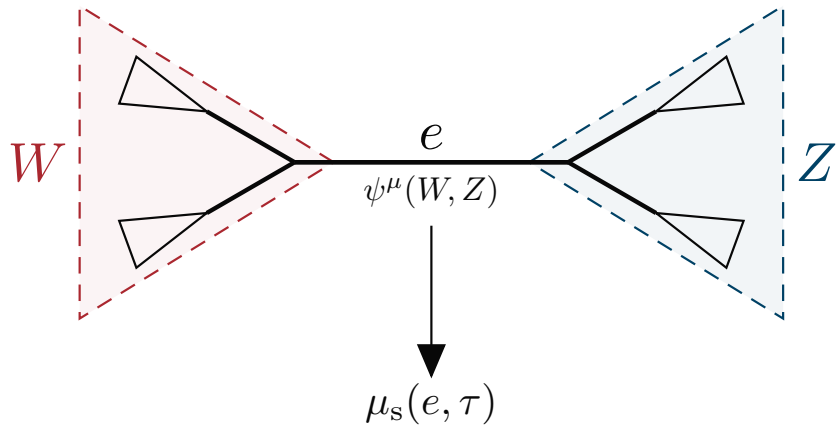
$$p(S_1 = s_1) = \frac{\exp(\phi_{s_1})}{\sum_{s_r \in \mathbb{S}_r} \exp(\phi_{s_r})}, \quad p(S_i = s | S_{\pi_i} = t) = \frac{\exp(\phi_{s|t})}{\sum_{s \in \mathbb{S}_{\cdot|t}} \exp(\phi_{s|t})}$$

Branch Length Parameters

$$Q_{\boldsymbol{\psi}}(\boldsymbol{q}|\tau) = \prod_{e \in E(\tau)} p^{\text{Lognormal}}\left(q_e \mid \mu(e,\tau), \sigma(e,\tau)\right)$$

- *Simple Split*

$$\mu_{\text{s}}(e,\tau) = \psi^{\mu}_{e/\tau}, \ \sigma_{\text{s}}(e,\tau) = \psi^{\sigma}_{e/\tau}.$$



SBNs Parameters

$$p(S_1 = s_1) = \frac{\exp(\phi_{s_1})}{\sum_{s_r \in \mathbb{S}_r} \exp(\phi_{s_r})}, \quad p(S_i = s | S_{\pi_i} = t) = \frac{\exp(\phi_{s|t})}{\sum_{s \in \mathbb{S}_{\cdot|t}} \exp(\phi_{s|t})}$$

Branch Length Parameters

$$Q_{\boldsymbol{\psi}}(\boldsymbol{q}|\tau) = \prod_{e \in E(\tau)} p^{\text{Lognormal}}\left(q_e \mid \mu(e,\tau), \sigma(e,\tau)\right)$$

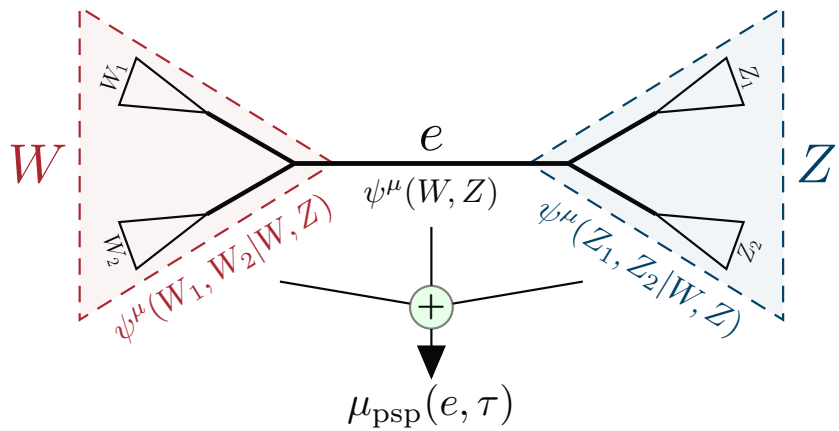
- *Simple Split*

  $$\mu_{\text{s}}(e,\tau) = \psi^{\mu}_{e/\tau}, \ \sigma_{\text{s}}(e,\tau) = \psi^{\sigma}_{e/\tau}.$$

- *Primary Subsplit Pair* (PSP)

  $$\mu_{\text{psp}}(e,\tau) = \psi^{\mu}_{e/\tau} + \sum_{s \in e/\!/\tau} \psi^{\mu}_{s}$$

  $$\sigma_{\text{psp}}(e,\tau) = \psi^{\sigma}_{e/\tau} + \sum_{s \in e/\!/\tau} \psi^{\sigma}_{s}.$$



SBNs Parameters $\boldsymbol{\phi}$. With $\tau^j, \boldsymbol{q}^j \overset{\text{iid}}{\sim} Q_{\boldsymbol{\phi},\boldsymbol{\psi}}(\tau,\ \boldsymbol{q})$

- *VIMCO*. [Minh and Rezende, ICML 2016]

$$\nabla_{\boldsymbol{\phi}} L^K(\boldsymbol{\phi},\boldsymbol{\psi}) \simeq \sum_{j=1}^{K} \left(\hat{L}^K_{j|-j}(\boldsymbol{\phi},\boldsymbol{\psi}) - \tilde{w}^j\right) \nabla_{\boldsymbol{\phi}} \log Q_{\boldsymbol{\phi}}(\tau^j).$$

- *RWS*. [Bornschein and Bengio, ICLR 2015]

$$\nabla_{\boldsymbol{\phi}} L^K(\boldsymbol{\phi},\boldsymbol{\psi}) \simeq \sum_{j=1}^{K} \tilde{w}^j \nabla_{\boldsymbol{\phi}} \log Q_{\boldsymbol{\phi}}(\tau^j).$$

Branch Length Parameters $\boldsymbol{\psi}$. $g_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}|\tau) = \exp(\boldsymbol{\mu}_{\boldsymbol{\psi},\tau} + \boldsymbol{\sigma}_{\boldsymbol{\psi},\tau} \odot \boldsymbol{\epsilon})$.

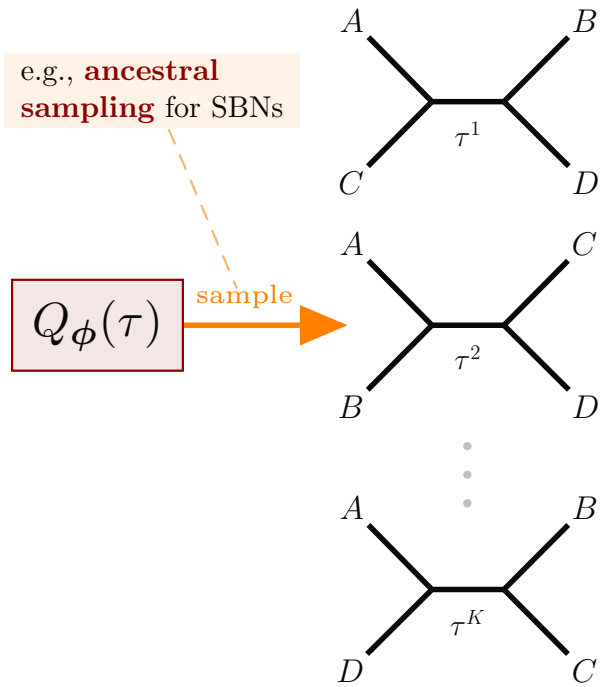
- *Reparameterization Trick*. Let $f_{\boldsymbol{\phi},\boldsymbol{\psi}}(\tau,\boldsymbol{q}) = \frac{p(\boldsymbol{Y}|\tau,\boldsymbol{q})p(\tau,\boldsymbol{q})}{Q_{\boldsymbol{\phi}}(\tau)Q_{\boldsymbol{\psi}}(\boldsymbol{q}|\tau)}$.

$$\nabla_{\boldsymbol{\psi}} L^K(\boldsymbol{\phi},\boldsymbol{\psi}) \simeq \sum_{j=1}^{K} \tilde{w}^j \nabla_{\boldsymbol{\psi}} \log f_{\boldsymbol{\phi},\boldsymbol{\psi}}(\tau^j, g_{\boldsymbol{\psi}}(\boldsymbol{\epsilon}^j|\tau^j))$$

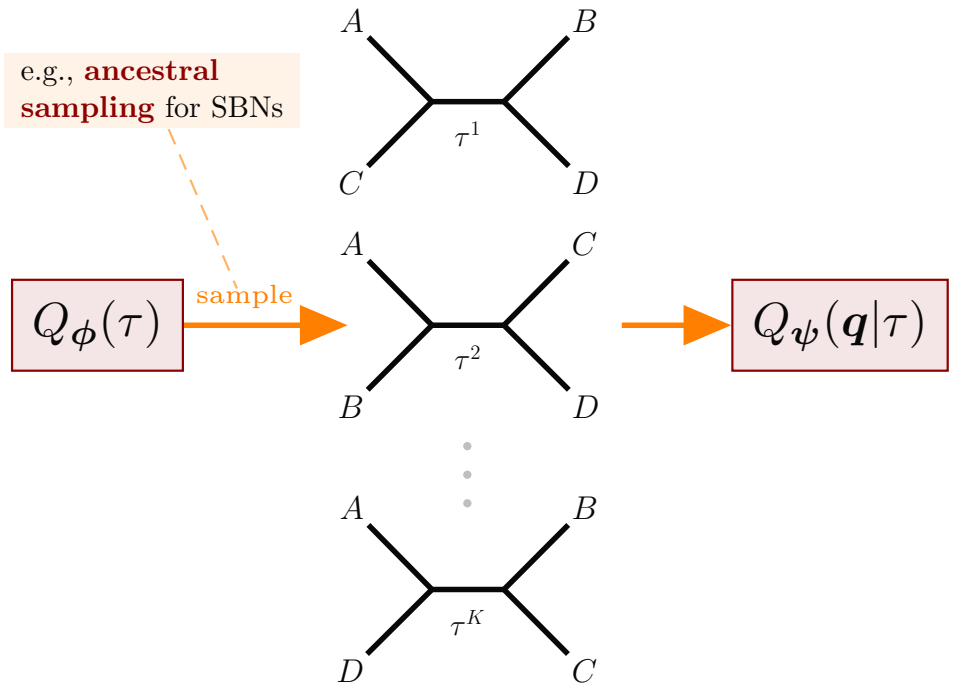
where $\tau^j \overset{\text{iid}}{\sim} Q_{\boldsymbol{\phi}}(\tau)$, $\boldsymbol{\epsilon}^j \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$.

$Q_{\phi}(\tau)$

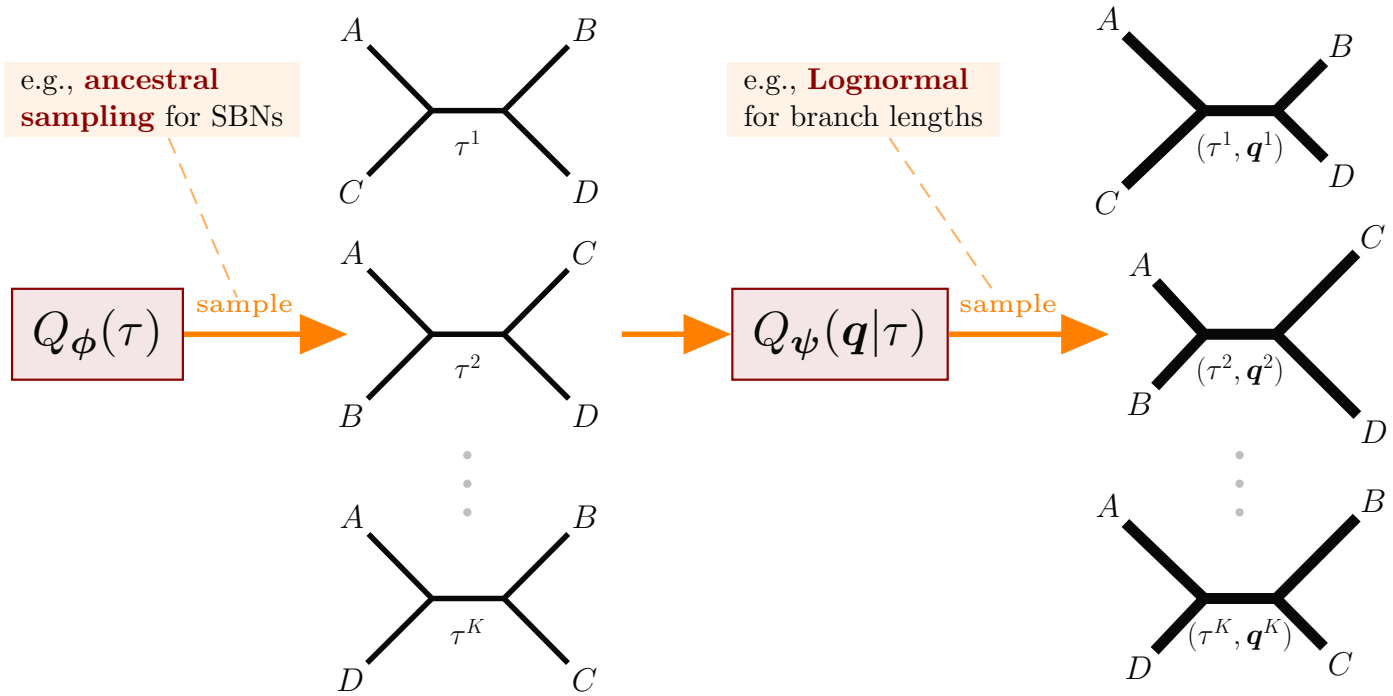
The VBPI Pipeline



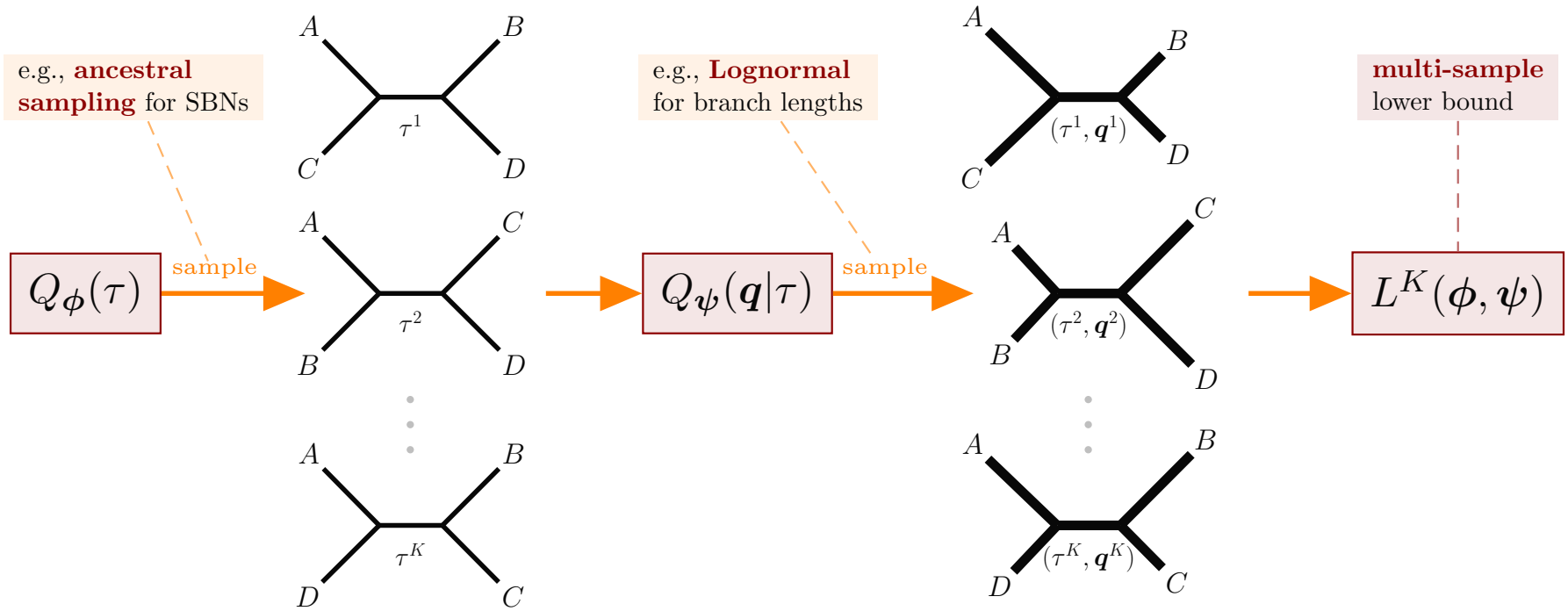
The VBPI Pipeline



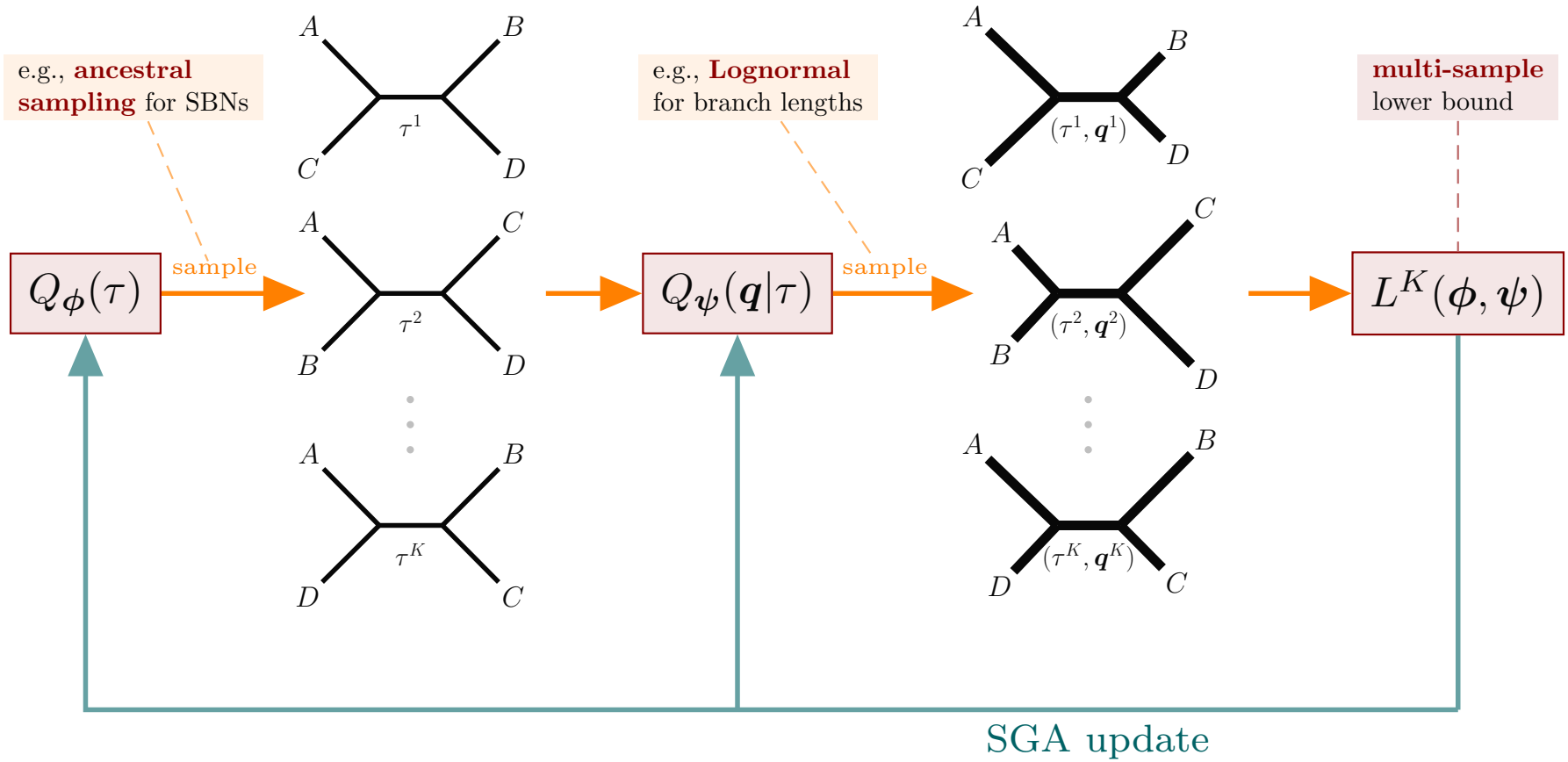
The VBPI Pipeline



The VBPI Pipeline

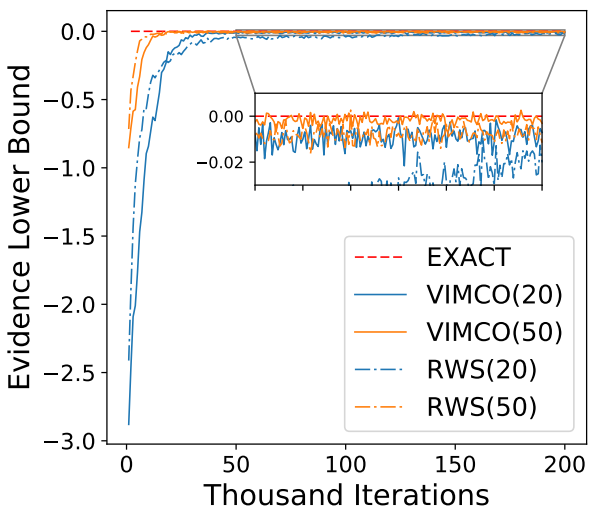


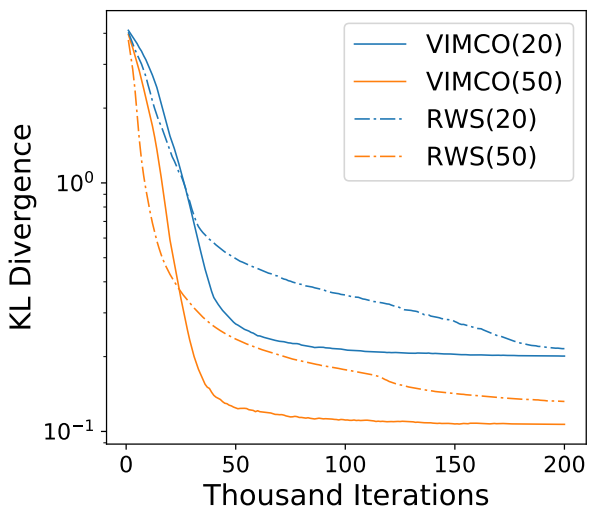
The VBPI Pipeline

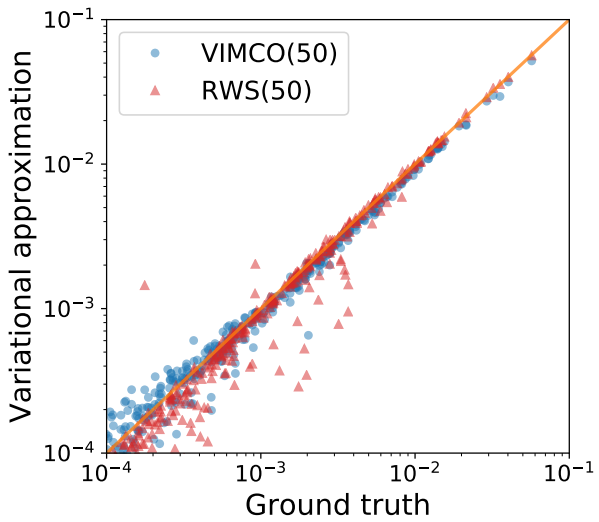


Performance on Synthetic Data

A simulated study on unrooted phylogenetic trees with 8 leaves (10395 trees). The target distribution is a random sample from the symmetric Dirichlet distribution $\mathrm{Dir}(\beta\mathbf{1})$, $\beta = 0.008$





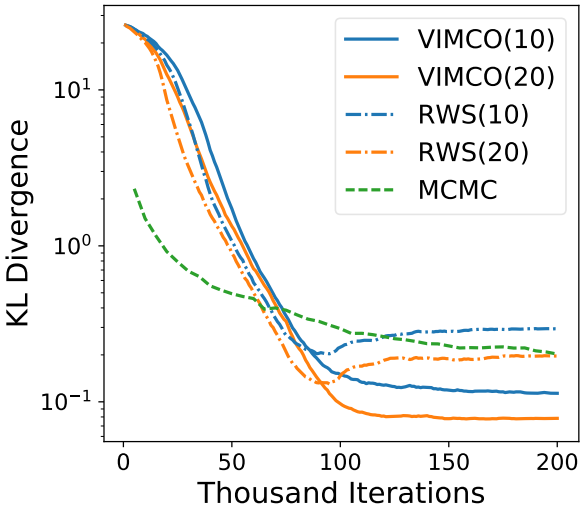


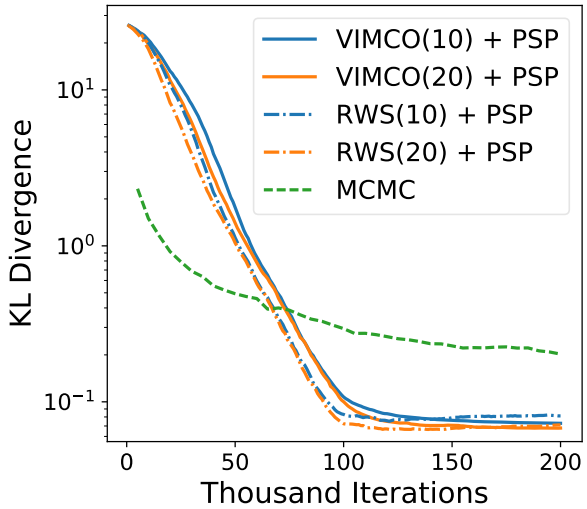
[Zhang and Matsen, ICLR 2019]

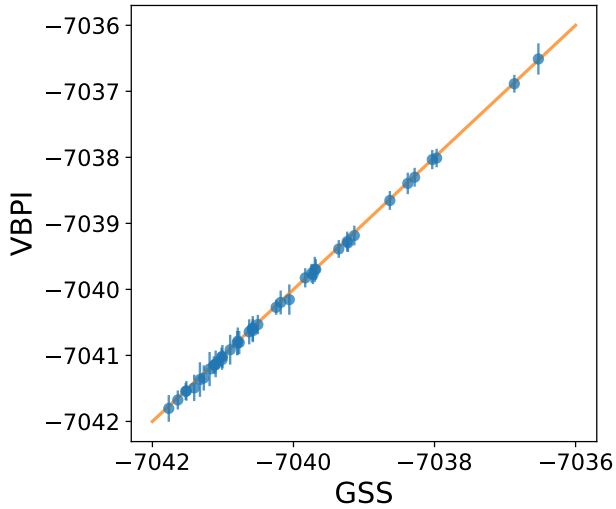
ELBOs approach 0 quickly $\Rightarrow$ SBNs approximations are flexible.

More samples in the multi-sample ELBOs could be helpful.

Performance on Real Data







[Zhang and Matsen, ICLR 2019]

- More samples ⇒ better exploration ⇒ better approximation
- More flexible branch length distributions across tree topologies (PSP) ease training and improve approximation
- Outperform MCMC via much more efficient tree space exploration and branch length updates

Performance on Real Data

Data set	Marginal Likelihood (NATs)				
	VIMCO(10)	**VIMCO(20)**	**VIMCO(10)+PSP**	**VIMCO(20)+PSP**	SS
DS1	-7108.43(0.26)	-7108.35(0.21)	-7108.41(0.16)	**-7108.42(0.10)**	-7108.42(0.18)
DS2	-26367.70(0.12)	-26367.71(0.09)	**-26367.72(0.08)**	-26367.70(0.10)	-26367.57(0.48)
DS3	-33735.08(0.11)	-33735.11(0.11)	**-33735.10(0.09)**	-33735.07(0.11)	-33735.44(0.50)
DS4	-13329.90(0.31)	-13329.98(0.20)	**-13329.94(0.18)**	-13329.93(0.22)	-13330.06(0.54)
DS5	-8214.36(0.67)	-8214.74(0.38)	-8214.61(0.38)	-8214.55(0.43)	**-8214.51(0.28)**
DS6	-6723.75(0.68)	-6723.71(0.65)	-6724.09(0.55)	**-6724.34(0.45)**	-6724.07(0.86)
DS7	-37332.03(0.43)	-37331.90(0.49)	-37331.90(0.32)	**-37332.03(0.23)**	-37332.76(2.42)
DS8	-8653.34(0.55)	-8651.54(0.80)	**-8650.63(0.42)**	-8650.55(0.46)	-8649.88(1.75)

[Zhang and Matsen, ICLR 2019]

- Competitive to state-of-the-art (stepping-stone), dramatically reducing cost at test time: VBPI(1000) vs SS(100,000)
- PSP alleviates the demand for large samples, reducing computation while maintaining approximation accuracy

Improve Branch Length Approximation

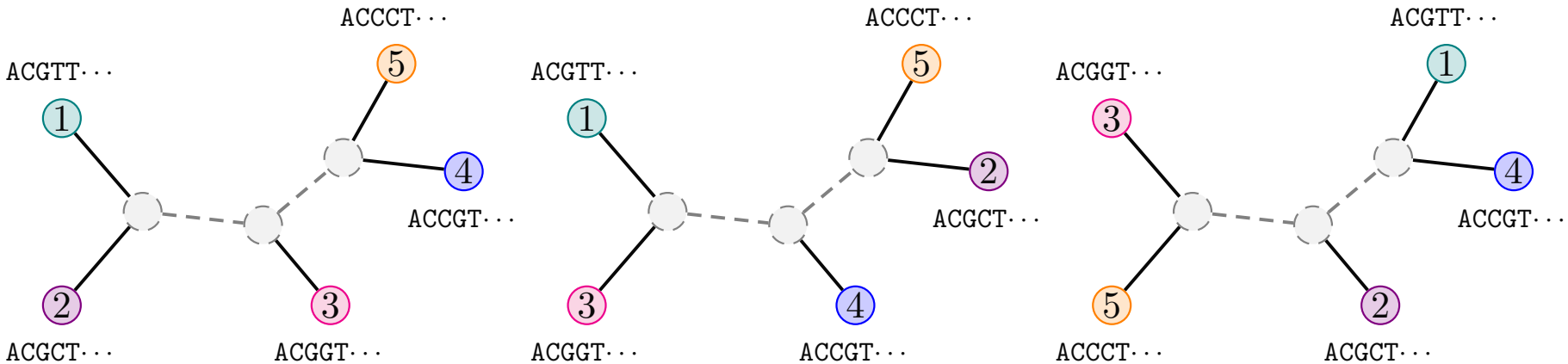
- Vanilla VBPI uses the diagonal Lognormal branch length approximation that completely ignored the correlation between branch lengths

Improve Branch Length Approximation

- Vanilla VBPI uses the diagonal Lognormal branch length approximation that completely ignored the correlation between branch lengths
- Improve the branch length approximation via normalizing flows?

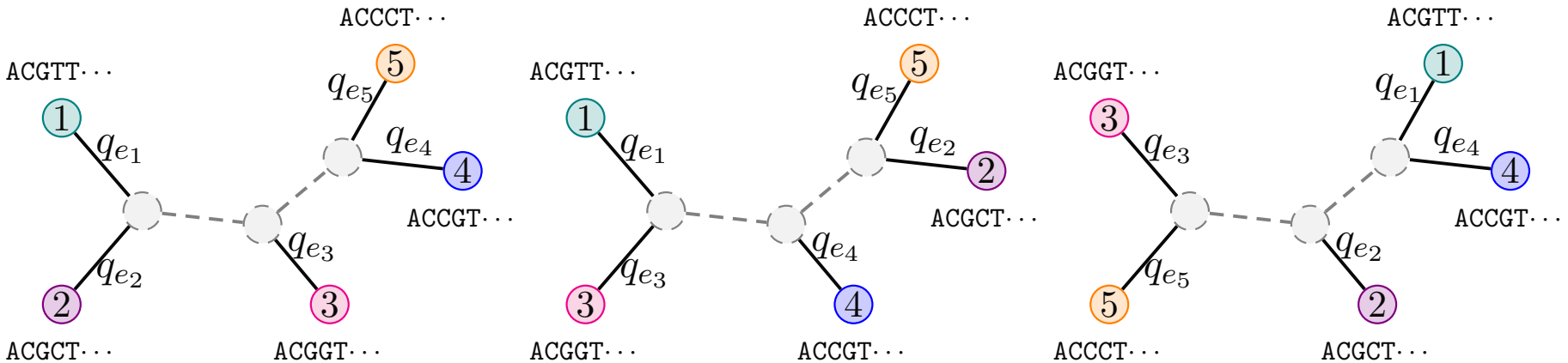
Improve Branch Length Approximation

- Vanilla VBPI uses the diagonal Lognormal branch length approximation that completely ignored the correlation between branch lengths
- Improve the branch length approximation via normalizing flows?
- Hard to align the branch length vectors consistently across tree topologies! Non-Euclidean branch length space over different tree topologies!



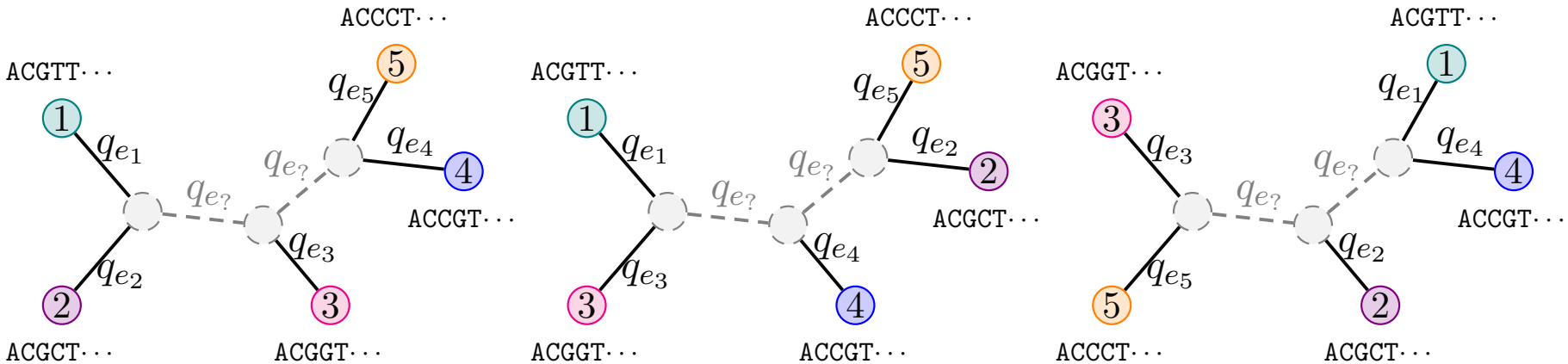
Improve Branch Length Approximation

- Vanilla VBPI uses the diagonal Lognormal branch length approximation that completely ignored the correlation between branch lengths
- Improve the branch length approximation via normalizing flows?
- Hard to align the branch length vectors consistently across tree topologies! Non-Euclidean branch length space over different tree topologies!



Improve Branch Length Approximation

- Vanilla VBPI uses the diagonal Lognormal branch length approximation that completely ignored the correlation between branch lengths
- Improve the branch length approximation via normalizing flows?
- Hard to align the branch length vectors consistently across tree topologies! Non-Euclidean branch length space over different tree topologies!



- Standard planar transformation

$$z_i = x_i + \gamma_i a \left(\sum\nolimits_j w_j x_j + b \right), \quad i = 1, \ldots, d$$

- Structured planar transformation on phylogenetic trees

$$z_e = \tilde{q}_e + \gamma_e a \left(\sum\nolimits_{e' \in E(\tau)} w_{e'} \tilde{q}_{e'} + b \right), \quad \forall\, e \in E(\tau)$$

where

$$\gamma_e = \psi^{\gamma}_{e/\tau} + \sum_{s \in e /\!\!/ \tau} \psi^{\gamma}_s, \quad w_e = \psi^{w}_{e/\tau} + \sum_{s \in e /\!\!/ \tau} \psi^{w}_s$$

- The above planar transformation is permutation equivariant.

- Standard affine coupling transformation

$$z_i = x_i,\ i \in S^c. \quad z_i = x_i \exp(\alpha_i(\boldsymbol{x}_{S^c})) + \beta_i(\boldsymbol{x}_{S^c}),\ i \in S.$$

- Structured affine coupling transformation on phylogenetic trees

$$z_e = \tilde{q}_e,\ e \in S^c. \quad z_e = \tilde{q}_e \exp\left(\alpha_e(\tilde{\boldsymbol{q}}_{S^c})\right) + \beta_e(\tilde{\boldsymbol{q}}_{S^c}),\ e \in S.$$

with a consistent pendant and interior bipartition $S \cup S^c$ of the edges across tree topologies, and permutation invariant α_e and β_e.

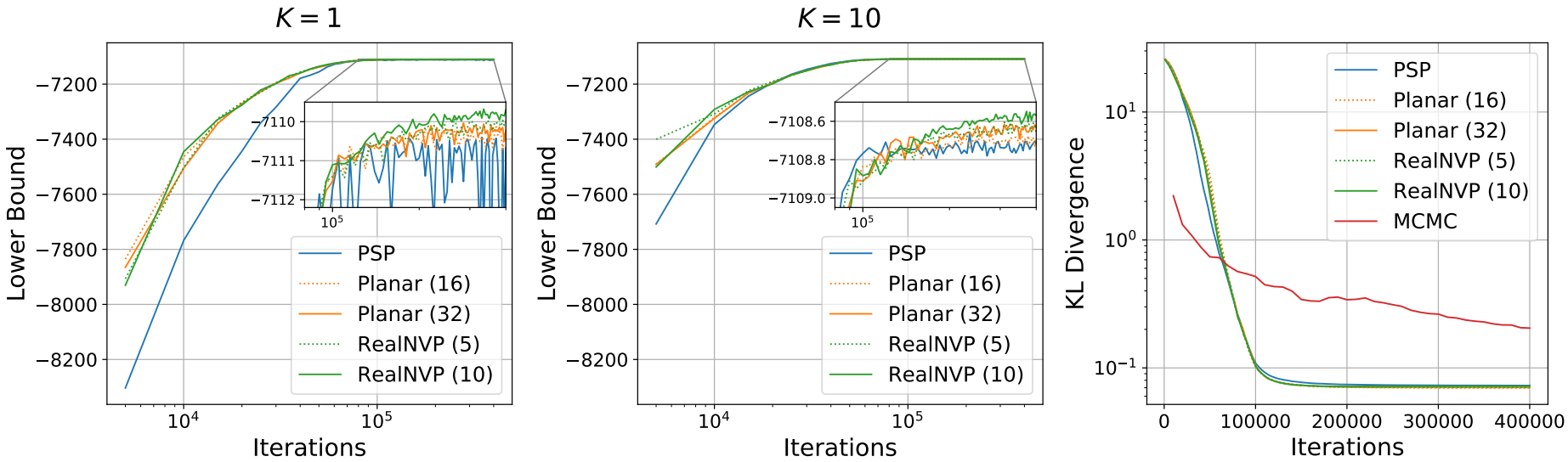
$$\begin{bmatrix} \alpha_e(\tilde{\boldsymbol{q}}_{S^c}) \\ \beta_e(\tilde{\boldsymbol{q}}_{S^c}) \end{bmatrix} = \begin{bmatrix} (\boldsymbol{w}_e^{\alpha})^T \\ \left(\boldsymbol{w}_e^{\beta}\right)^T \end{bmatrix} \rho\left(\sum_{e' \in S^c} \tilde{q}_{e'} \boldsymbol{w}_{e'} + \boldsymbol{b}\right) + \begin{bmatrix} b_e^{\alpha} \\ b_e^{\beta} \end{bmatrix}$$

- The above affine coupling transformation is permutation equivariant.

Lower Bounds and Marginal Likelihoods

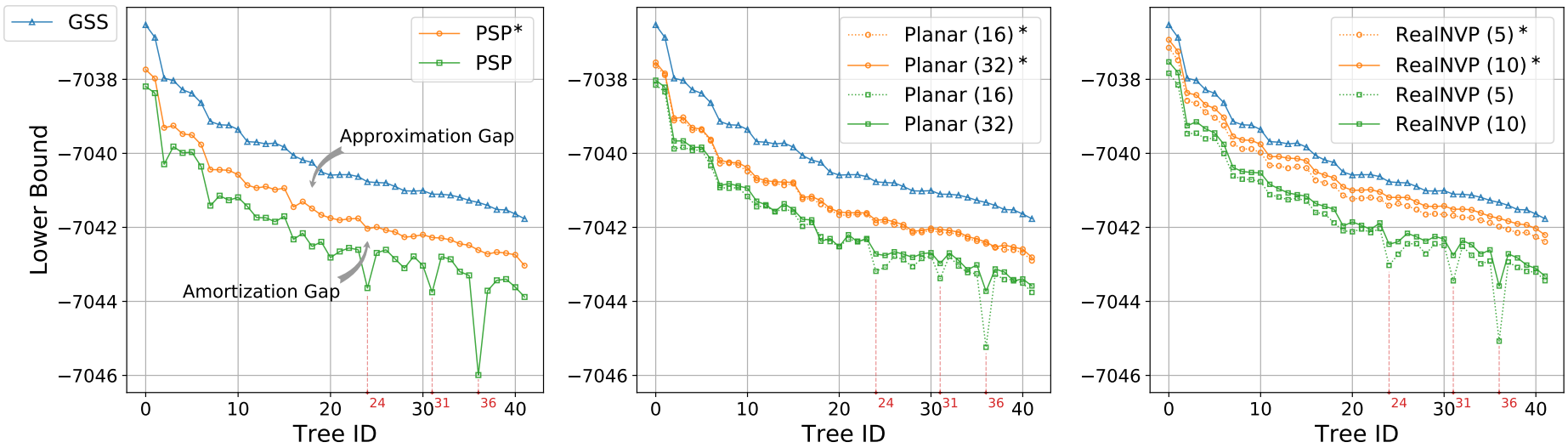
	Data set	DS1	DS2	DS3	DS4	DS5	DS6	DS7	DS8
	# Taxa	27	29	36	41	50	50	59	64
	# Sites	1949	2520	1812	1137	378	1133	1824	1008
LB (K=1)	PSP	-7111.23(1.04)	-26369.63(0.69)	-33736.60(0.33)	-13332.37(0.54)	-8218.35(0.20)	-6729.27(0.50)	-37335.15(0.11)	-8655.48(0.38)
	Planar(16)	-7110.33(0.16)	-26368.80(0.27)	-33736.14(0.14)	-13331.92(0.11)	-8217.98(0.13)	-6728.89(0.18)	-37334.78(0.11)	-8655.15(0.17)
	Planar(32)	-7110.22(0.17)	-26368.69(0.23)	-33736.02(0.21)	-13331.73(0.12)	-8217.90(0.14)	-6728.68(0.19)	-37334.60(0.12)	-8654.97(0.16)
	RealNVP(5)	-7110.12(0.13)	-26368.75(0.24)	-33735.86(0.10)	-13331.71(0.11)	-8217.80(0.14)	-6728.54(0.15)	-37334.44(0.11)	-8654.62(0.13)
	RealNVP(10)	**-7109.80(0.11)**	**-26368.59(0.23)**	**-33735.81(0.12)**	**-13331.39(0.08)**	**-8217.56(0.12)**	**-6728.04(0.14)**	**-37333.94(0.09)**	**-8654.02(0.12)**
LB (K=10)	PSP	-7108.73(0.02)	-26367.88(0.02)	-33735.29(0.02)	-13330.34(0.03)	-8215.57(0.04)	-6725.48(0.04)	-37332.69(0.03)	-8651.88(0.04)
	Planar(16)	-7108.70(0.02)	-26367.80(0.01)	-33735.21(0.01)	-13330.28(0.02)	-8215.44(0.04)	-6725.42(0.04)	-37332.50(0.03)	-8651.80(0.04)
	Planar(32)	-7108.64(0.02)	-26367.77(0.01)	-33735.17(0.01)	-13330.22(0.02)	-8215.37(0.03)	-6725.32(0.04)	-37332.43(0.03)	-8651.72(0.04)
	RealNVP(5)	-7108.63(0.02)	-26367.77(0.01)	-33735.18(0.01)	-13330.22(0.02)	-8215.36(0.03)	-6725.33(0.04)	-37332.42(0.03)	-8651.62(0.04)
	RealNVP(10)	**-7108.58(0.02)**	**-26367.75(0.01)**	**-33735.16(0.01)**	**-13330.16(0.02)**	**-8215.29(0.03)**	**-6725.18(0.04)**	**-37332.30(0.02)**	**-8651.41(0.03)**
ML	PSP	-7108.39(0.18)	-26367.71(0.08)	-33735.09(0.10)	-13329.93(0.21)	-8214.44(0.48)	-6724.13(0.48)	-37331.92(0.32)	-8650.12(0.58)
	Planar(16)	-7108.39(0.15)	-26367.70(0.07)	-33735.09(0.07)	-13329.93(0.17)	-8214.49(0.42)	-6724.25(0.45)	-37331.91(0.26)	-8650.42(0.52)
	Planar(32)	-7108.40(0.14)	-26367.70(0.06)	**-33735.09(0.05)**	-13329.93(0.16)	-8214.50(0.38)	-6724.19(0.44)	-37331.93(0.23)	-8650.40(0.50)
	RealNVP(5)	-7108.40(0.14)	**-26367.71(0.04)**	-33735.09(0.06)	-13329.92(0.16)	-8214.50(0.38)	-6724.28(0.39)	**-37331.92(0.22)**	-8650.46(0.44)
	RealNVP(10)	**-7108.39(0.11)**	**-26367.71(0.04)**	**-33735.09(0.05)**	**-13329.92(0.13)**	-8214.51(0.36)	**-6724.25(0.37)**	**-37331.90(0.22)**	**-8650.42(0.41)**
	SS	-7108.42(0.18)	-26367.57(0.48)	-33735.44(0.50)	-13330.06(0.54)	**-8214.51(0.28)**	-6724.07(0.86)	-37332.76(2.42)	-8649.88(1.75)

Computational Complexity and Convergence



- Achieve comparable approximation quality when PSP converges, and quickly surpass PSP as the number of iterations increases.
- Maintain the speed advantage of PSP when compared to MCMC.

Approximation and Amortization Gaps



Gap	PSP		Planar (16)		Planar (32)		RealNVP (5)		RealNVP (10)	
	Tree 36	All	Tree 36	All	Tree 36	All	Tree 36	All	Tree 36	All
Approximation	1.29	1.21	1.12	1.08	1.07	1.02	0.65	0.62	**0.43**	**0.40**
Amortization	3.37	0.84	2.80	0.82	**1.33**	**0.72**	3.10	0.98	1.83	0.93
Inference	4.66	2.05	3.92	1.90	2.40	1.74	3.75	1.60	**2.26**	**1.33**

- We introduced **VBPI**, a general variational framework for Bayesian phylogenetic inference.
- **VBPI** allows efficient learning on both tree topology and branch lengths, providing competitive performance to MCMC while requiring much less computation.
- Can be used for further statistical analysis (e.g., marginal likelihood estimation) via importance sampling.
- Can be improved with more advanced deep learning techniques (e.g., normalizing flows)
- There are many extensions, including more flexible branch length distributions, more flexible amortization architectures, more general models, etc.

- L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In ICLR, 2016
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.
- Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.
- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

- L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163, 2016.
- S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In Advances in neural information processing systems, pages 271–279, 2016.
- Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. arXiv preprint arXiv:1605.09782, 2016.
- Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In ICCV, 2017.

References

- Sebastian Hóhna and Alexei J. Drummond. Guided tree topology proposals for Bayesian phyloge- netic inference. Syst. Biol., 61(1):1–11, January 2012.
- Bret Larget. The estimation of tree posterior probabilities using conditional clade probability distributions. Syst. Biol., 62(4):501–511, July 2013.
- Zhang, C. and Matsen F. A., Generalizing Tree Probability Estimation via Bayesian Networks. In Advances in Neural Information Processing Systems, 2018.
- Zhang, C. and Matsen F. A., Variational Bayesian Phylogenetic Inference. In Proceedings of the 7th International Conference on Learning Representations, 2019.

References

- Zhang, C., Improved Variational Bayesian Phylogenetic Inference with Normalizing Flows. In Advances in Neural Information Processing Systems, 2020.