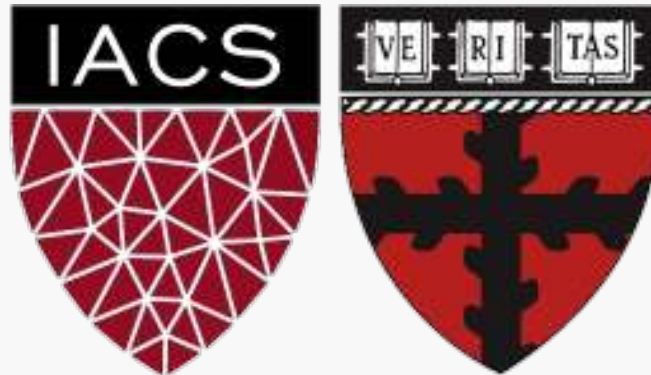


Variational Auto-Encoders

Part One

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner



Outline

- Motivation for Variational Autoencoders (VAE)
- Inference in Neural Networks
 - Bayesian Linear Regression
 - Bayesian Neural Networks
 - Introduction to variational methods
 - Variational Autoencoder as an inference model
- Variational Autoencoders as generative model
 - Separability of VAE
 - Tips & tricks
 - Other generative models

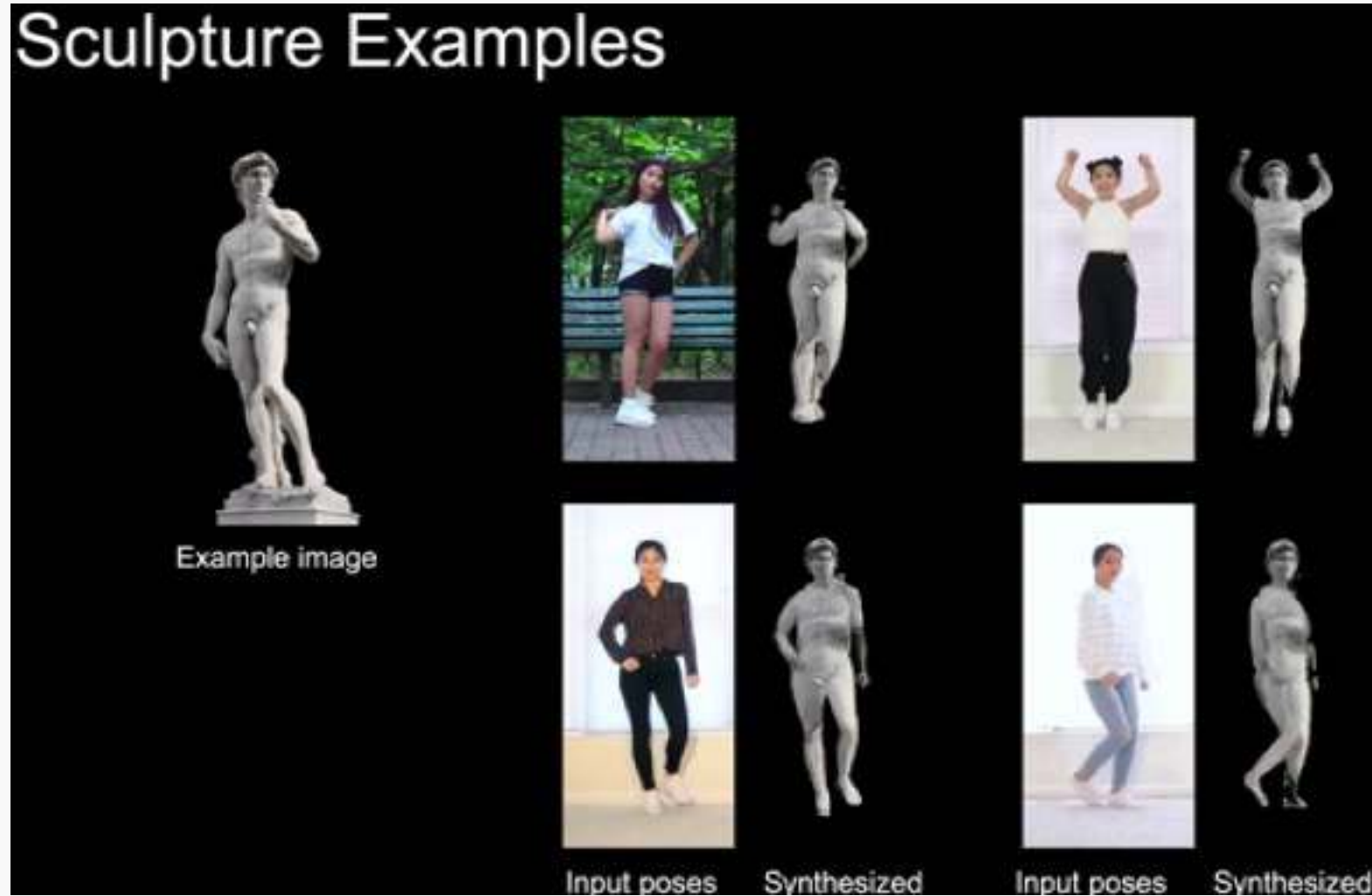


Outline: Part 1

- Motivation for Variational Autoencoders (VAE)
- Inference in Neural Networks
 - Bayesian Linear Regression
 - Bayesian Neural Networks
 - Introduction to Variational methods
 - Variational Autoencoder as an inference model
- Variational Autoencoders as generative model
 - Separability of VAE
 - Tips & tricks
 - Other generative models



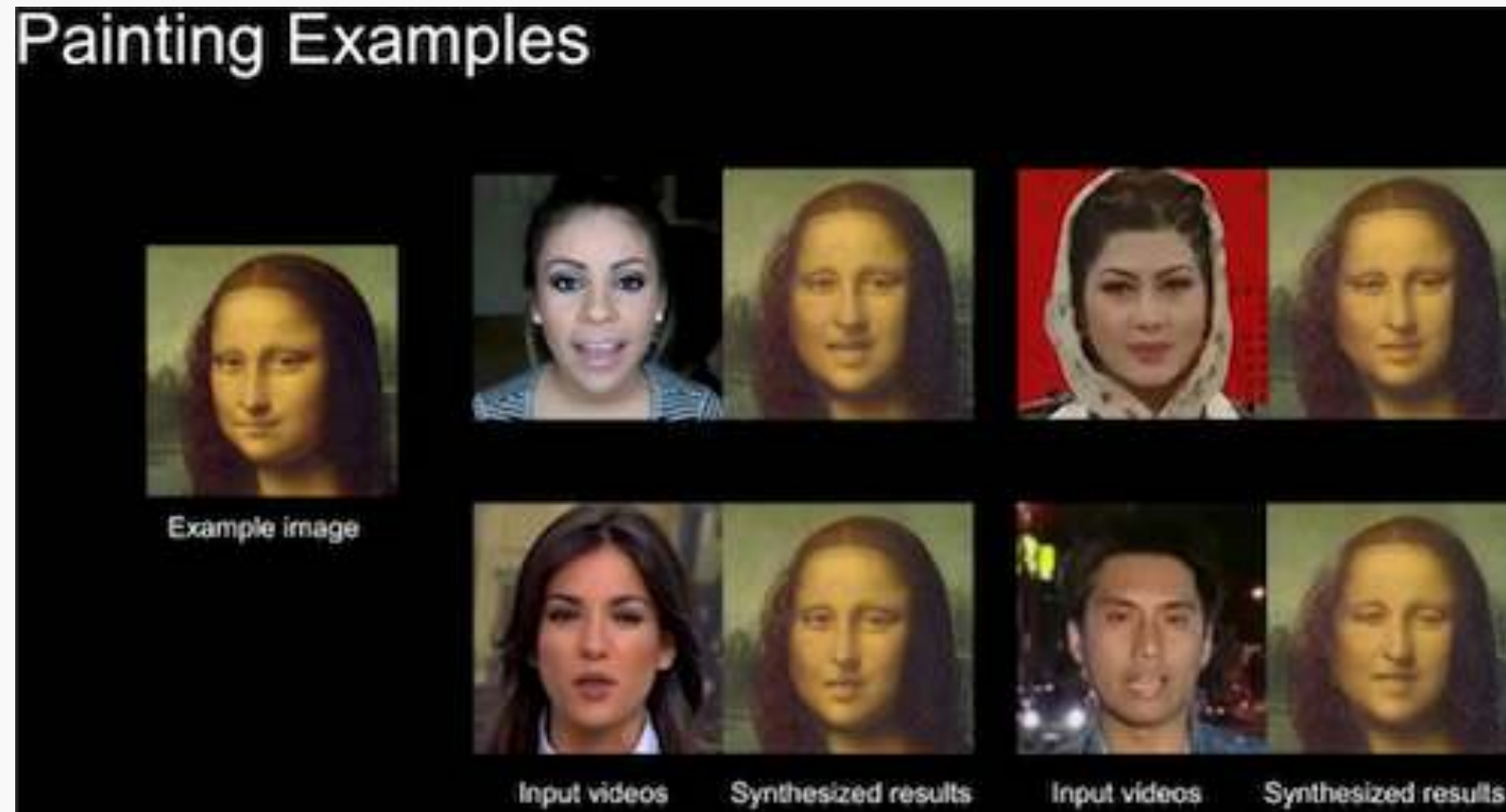
State of the Art in AI – sans NLP



<https://nvlabs.github.io/few-shot-vid2vid/>



State of the Art in AI – sans NLP



<https://nvlabs.github.io/few-shot-vid2vid/>



Generative Modeling



https://github.com/tkarras/progressive_growing_of_gans

Generative Modeling



https://github.com/tkarras/progressive_growing_of_gans

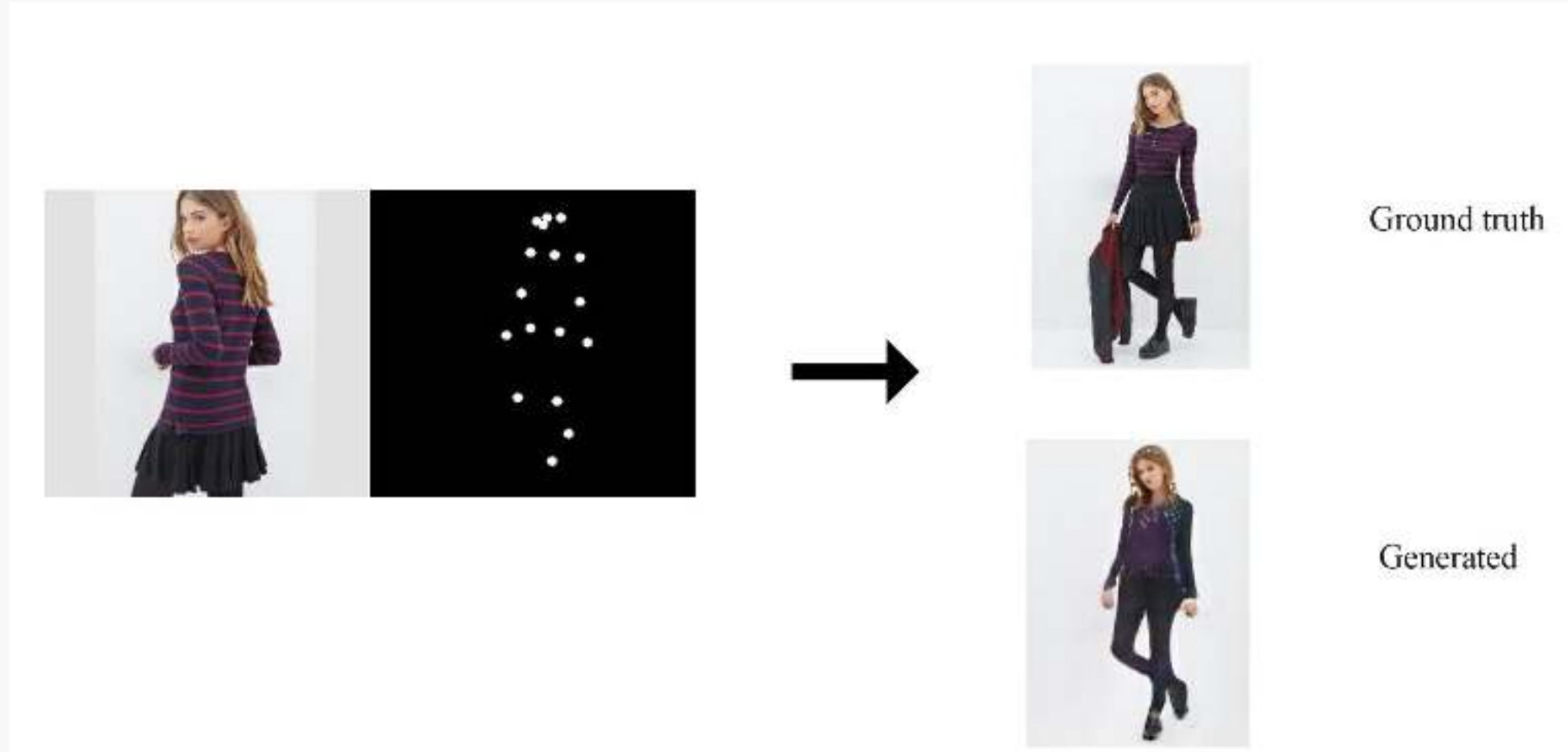


Generative Modeling



Figure 7: Generated samples

Generative Modeling



Generative Modeling

Zebras \leftrightarrow Horses



zebra \rightarrow horse



horse \rightarrow zebra



(a) MidiNet model 1



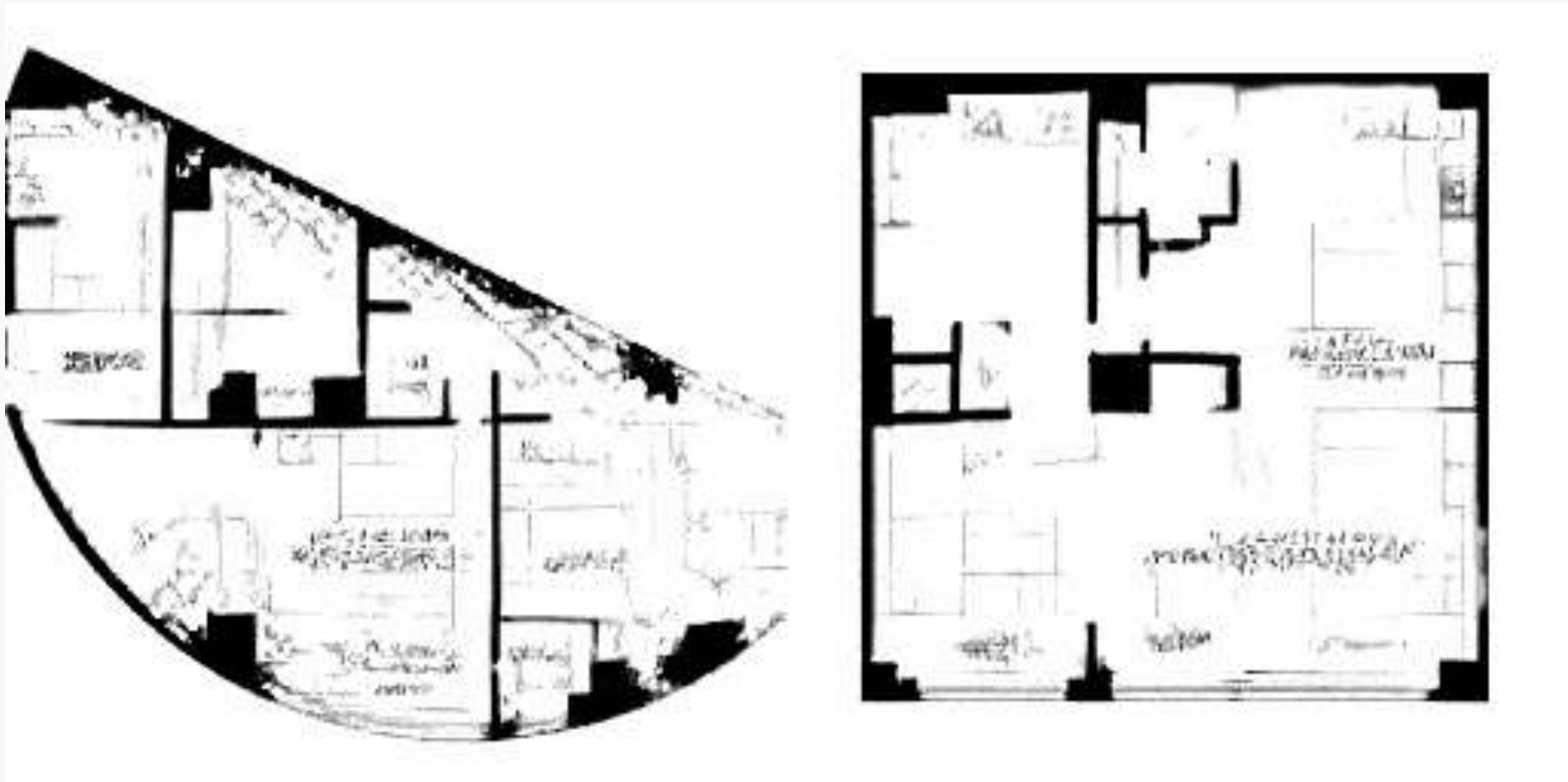
(b) MidiNet model 2



(c) MidiNet model 3

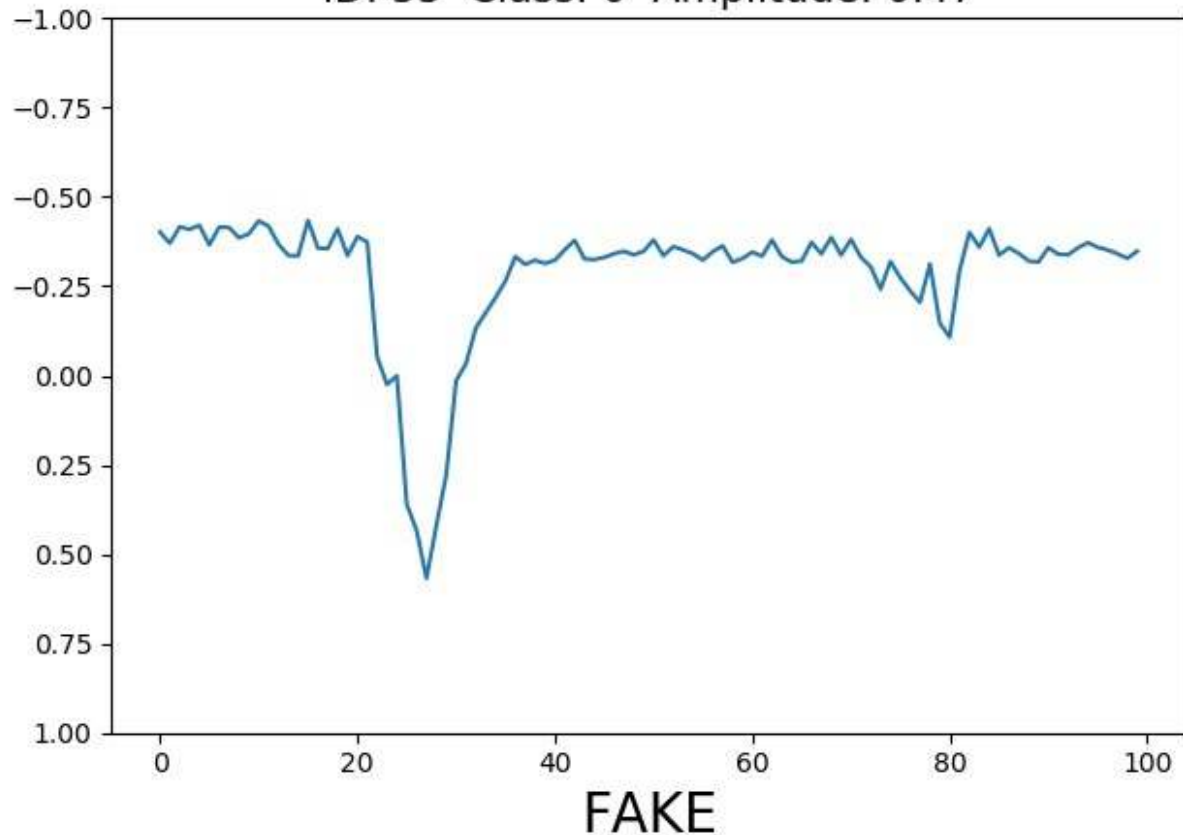
Figure 3. Example result of the melodies (of 8 bars) generated by different implementations of MidiNet.

Another use of generating new data is to give us ideas and options. Suppose we're planning a house. We can give the computer the space we have available, and its location. From this, the computer can give us some ideas.

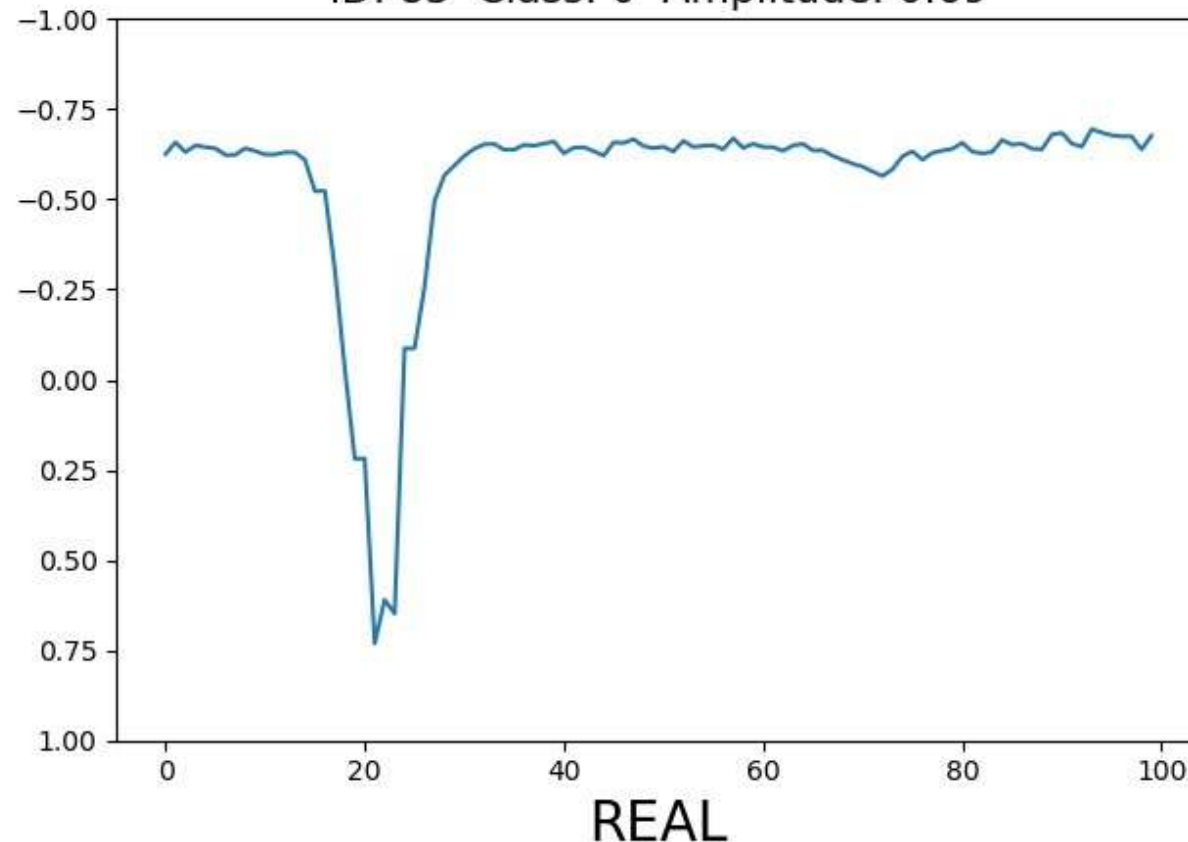


Big networks require big data, and getting high-quality, labeled data is difficult. If we're generating that data our selves, we can make as much of it as we like.

ID: 55 Class: 0 Amplitude: 0.47



ID: 83 Class: 0 Amplitude: 0.69



To summarize:
VAEs are a form of generative models



Variational AutoEncoder

Popular explanations of VAE as generative models

Using Variational Autoencoder (VAE) to Generate New Images

An advancement of traditional autoencoder.



Muhammad Ardi [Follow](#)

Oct 19, 2020 · 14 min read



This article introduces everything you need to know about VAE models. We provide a step-by-step guide on large image datasets and use

And why they're so useful in creating your own generative text, art and even music

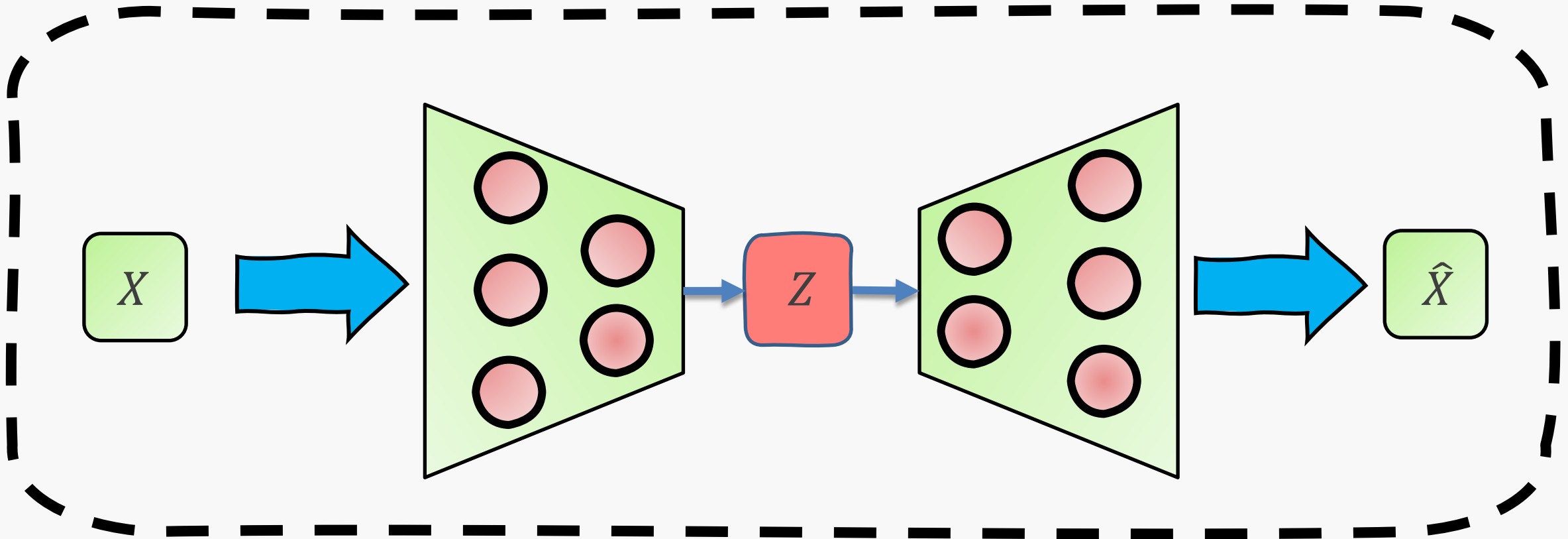


Irhum Shafkat · Feb 4, 2018 · 9 min read

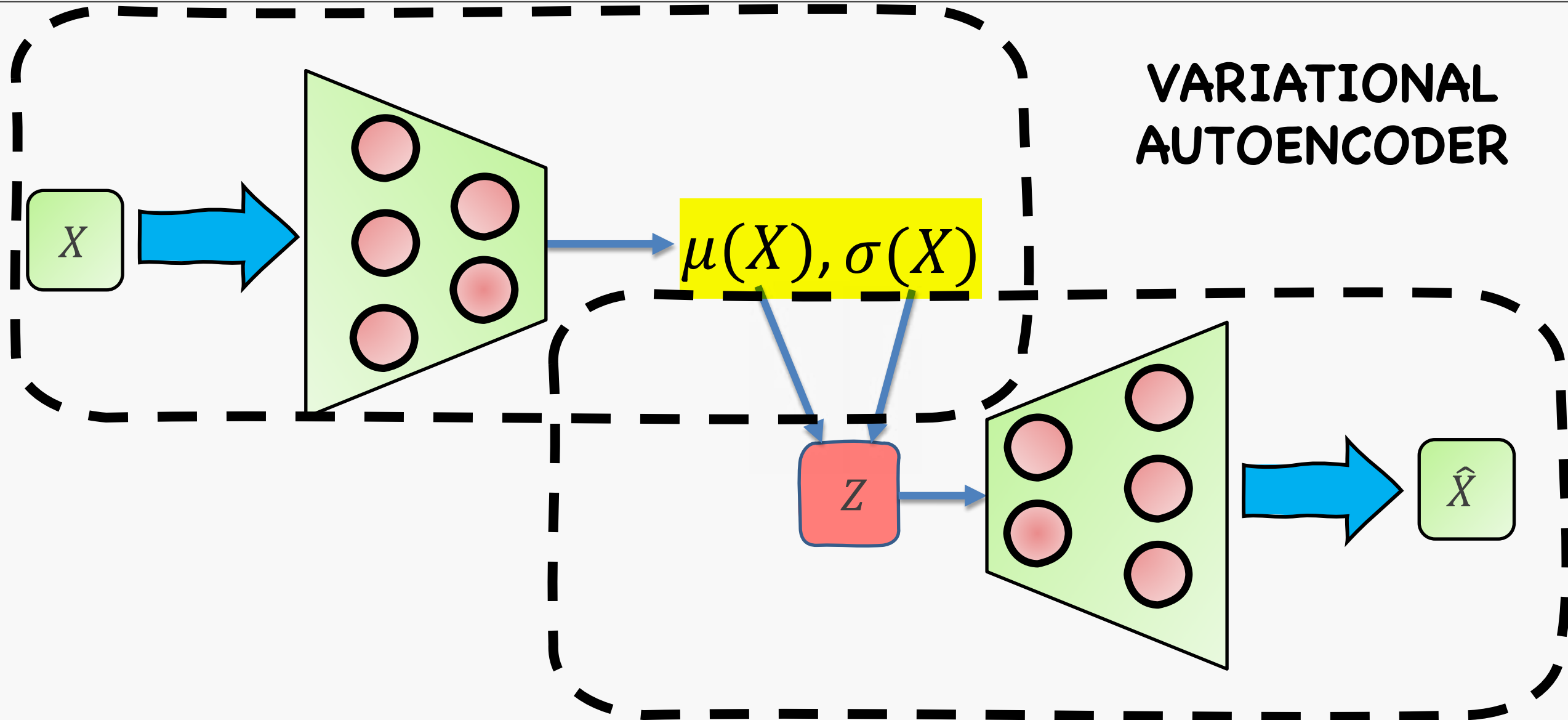


RECAP: Vanilla AutoEncoder

AUTOENCODER



Variational AutoEncoder



Variational Autoencoders

Typical explanations include:

- We want a continuous latent space representation
- We use a simple Reparameterization “trick”
- We add an Estimated Lower Bound (ELBO) error to reconstruction term



Why are Variational Autoencoders built this way?



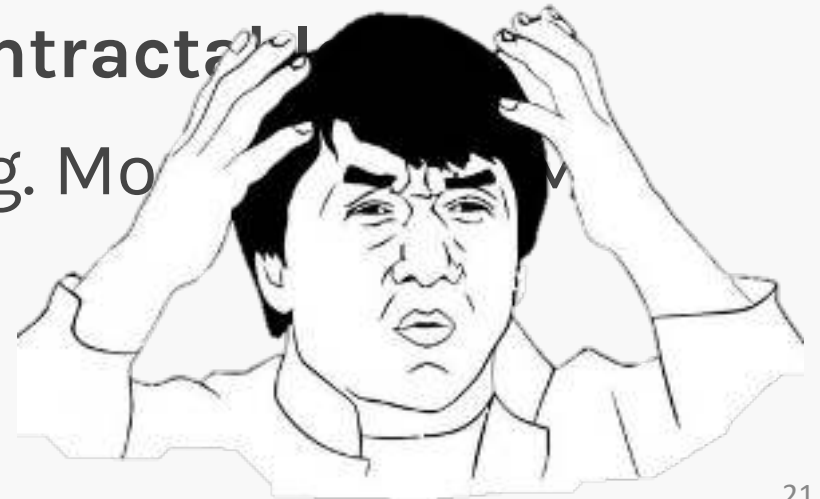
Variational AutoEncoder: Original Paper

Let us consider some dataset $X = \{x^{(i)}\}_{i=1}^N$ consisting of N i.i.d. samples of some continuous or discrete variable x . We assume that the data are generated by some random process, involving an unobserved continuous random variable z .

...

We are interested in a general algorithm that works in case of:

1. *Intractability*: $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$ is **intractable**
2. *Large Dataset*: Sampling based solutions eg. MCMC would be too slow



[Auto-Encoding Variational Bayes \(Diederik P. Kingma et al\)](#)

Variational Autoencoders as Inference models

The story of Variational Auto-encoders is like Coca-Cola

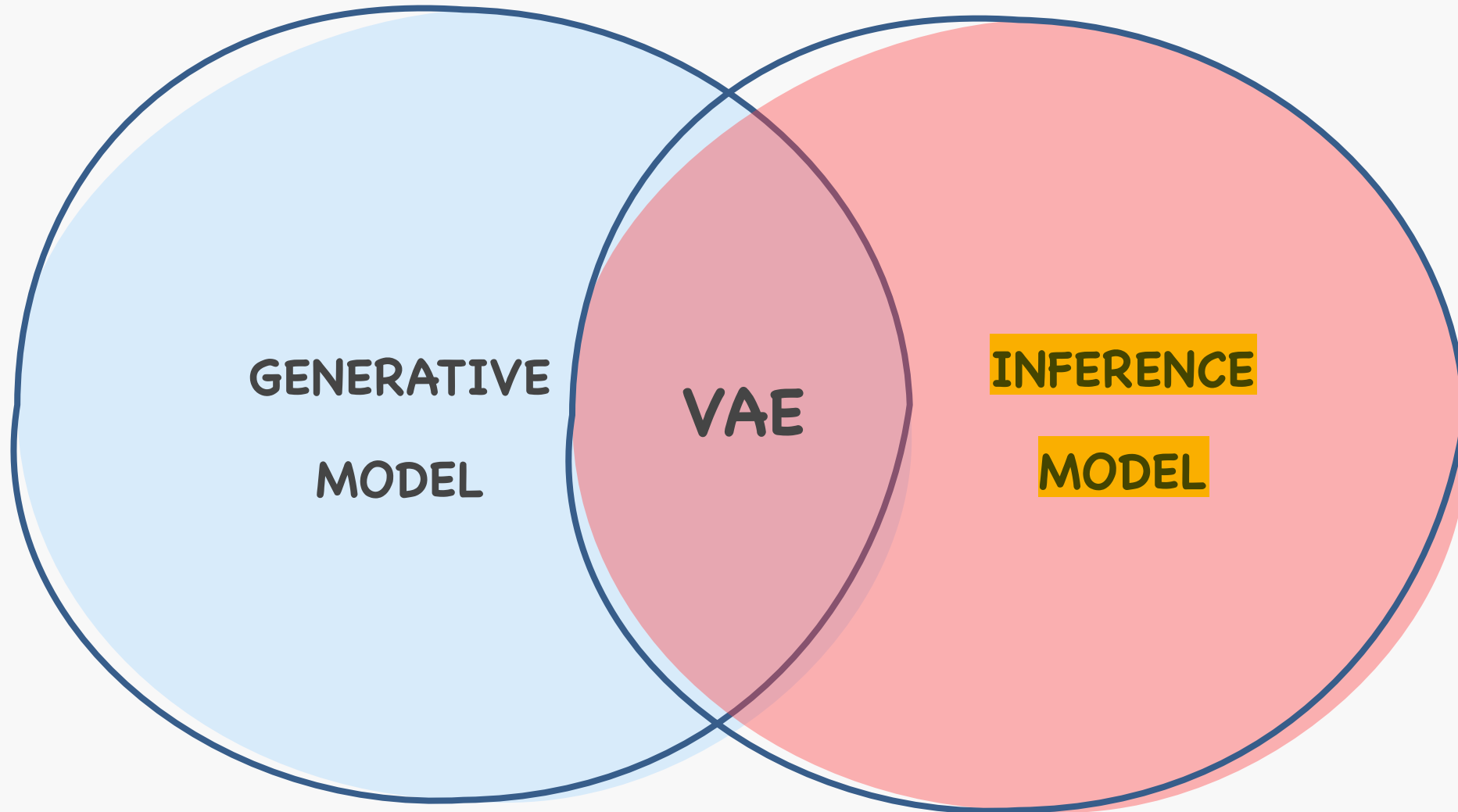
- Originally invented by John Pemberton to counter his own morphine addiction.
- Eventually found commercial success as a sugary drink and now is synonymous with soft beverages.



Source : <https://en.wikipedia.org/wiki/Coca-Cola>



Variational AutoEncoders



Are you ready?



BAYESIAN
LINEAR REGRESSION

BAYESIAN
NEURAL NETWORKS



VARIATIONAL
AUTO-ENCODERS

VARIATIONAL
METHODS



Bayesian Linear Regression



Inference Review

Bootstrap combined with Maximum Likelihood can give us the distribution of the coefficients.

Such a method is:

- Easy to understand
- Easy to implement
- Statistically equivalent to analytical solution



Inference Review

Let us begin by considering the case of linear regression:

$$y = f(x) + \epsilon \quad \text{where} \quad f(x) = \beta_0 + \beta_1 x$$

We interpret the ϵ term to be noise introduced by random variations in nature, or imprecisions of our scientific instruments and everything else.

If we knew the exact form of $f(x)$, for example, $f(x) = \beta_0 + \beta_1 x$, and there was no noise in the data, then estimating the $\hat{\beta}$'s would have been exact.

Sometimes ϵ is considered as “catch-it-all” term



Confidence intervals for the predictors estimates (cont)

However, two things happen, which result in mistrust of the values of $\hat{\beta}'s$:

- observational error is always there – this is called **aleatoric** error, or **irreducible** error.
- we do not know the exact form of $f(x)$ - this is called **misspecification** error and it is part of the **epistemic** error.

We will put everything into the catch-it-all term ε .

Because of ε , every time we measure the response y for a fix value of x , we will obtain a different observation, and hence a different estimate of $\hat{\beta}'s$.

Confidence intervals for the predictors estimates (cont)

So, if we just have one set of measurements of $\{X, Y\}$, our estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are just for this particular **realization**.

Question: If this is just one realization of the reality how do we know the truth? How do we deal with this conundrum?

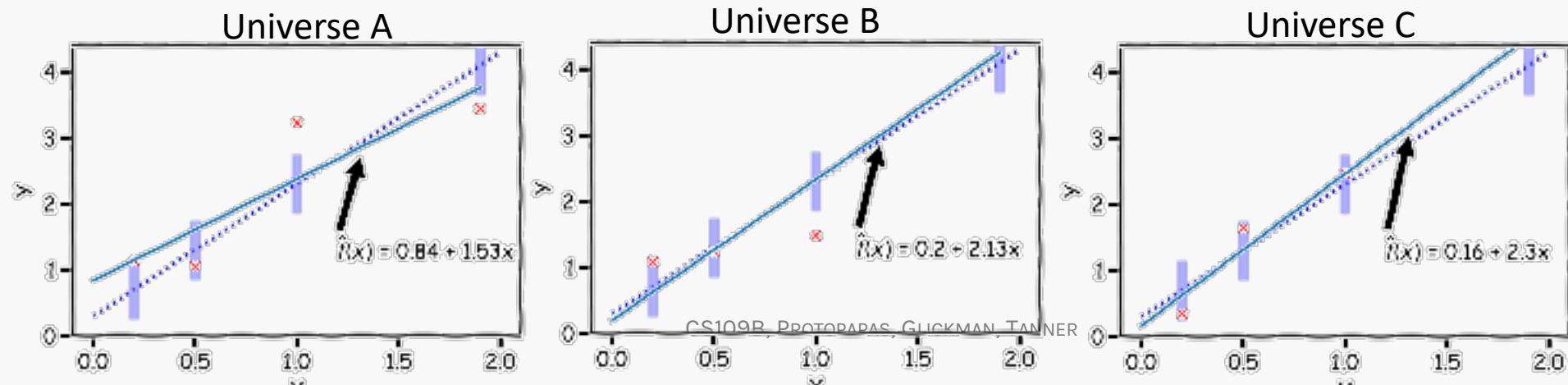


Confidence intervals for the predictors estimates (cont)

So, if we just have one set of measurements of $\{X, Y\}$, our estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are just for this particular realization.

Question: If this is just one realization of the reality how do we know the truth? How do we deal with this conundrum?

Imagine (magic realism) we have parallel universes, and we repeat this experiment on each of the other universes.



Confidence intervals for the predictors estimates (cont)

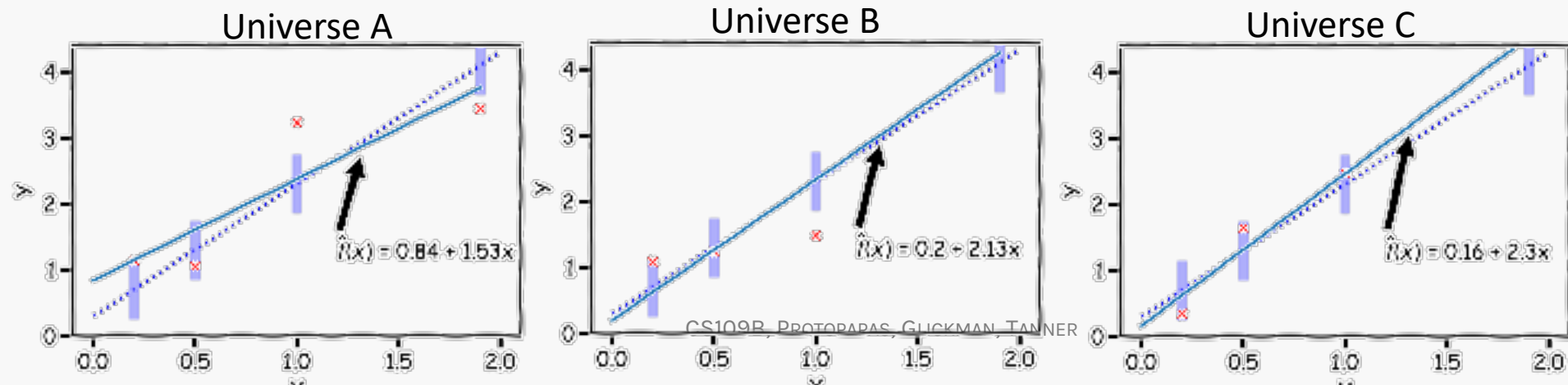
So, if we just have one set of measurements of $\{X, Y\}$, our estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are just for this particular realization

Question: If this is just one realization of the truth? How do we deal with this?

We use **bootstrap** to create different dataset.

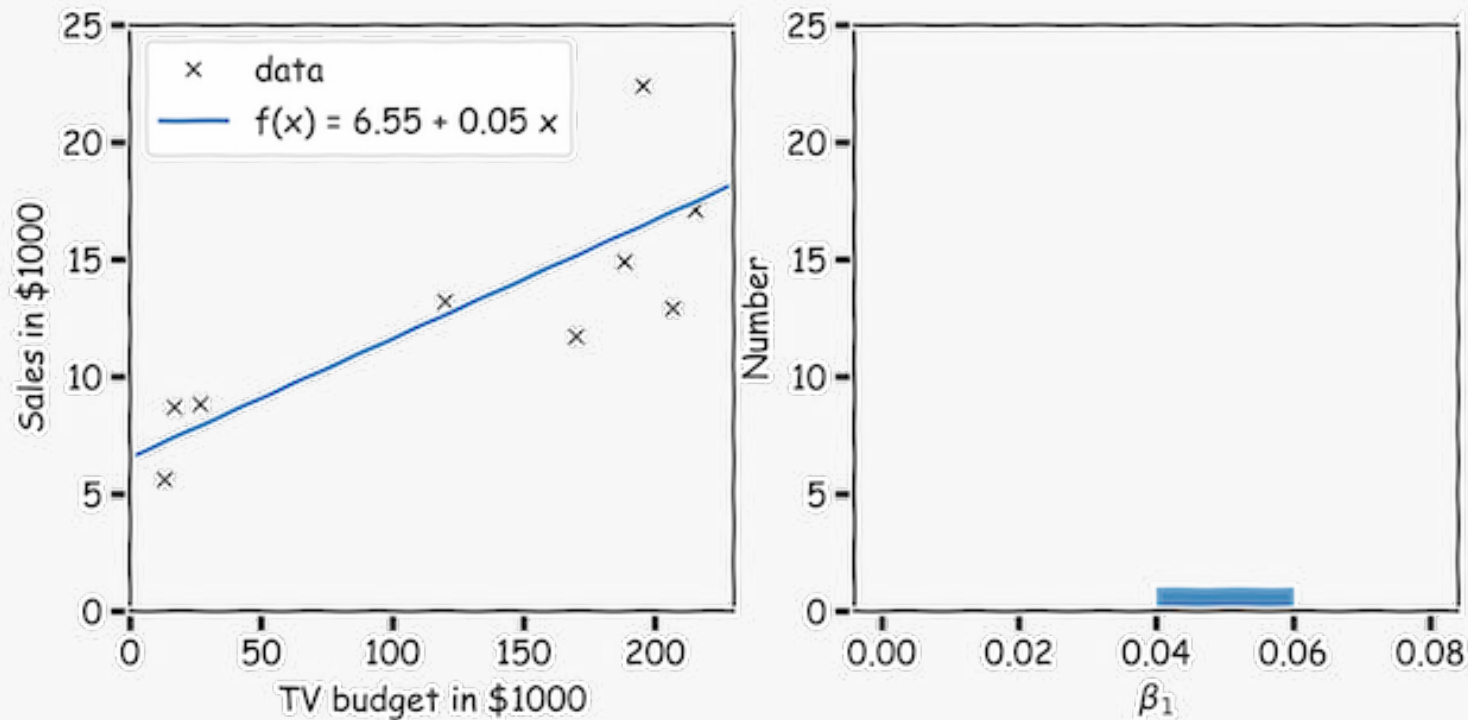
Bootstrap is sampling with replacement. Bootstrapping was covered **extensively** in CS109A.

Imagine (magic realism) we have parallel universes, and we repeat this experiment on each of the other universes.



Confidence intervals for the predictors estimates (cont)

In our magical realisms, we can now sample multiple times. One universe, one sample, one set of estimates for $\hat{\beta}_0, \hat{\beta}_1$

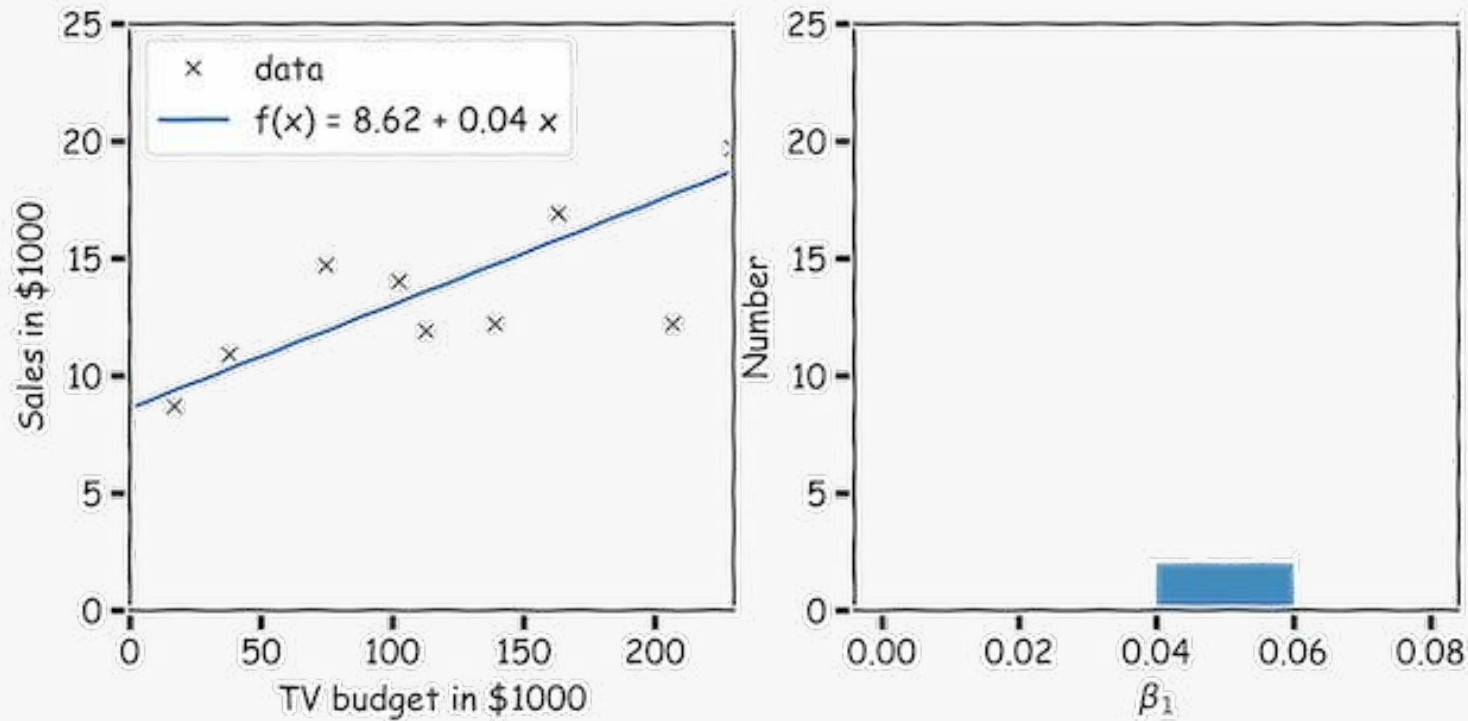


There will be an equivalent plot for $\hat{\beta}_0$ which we don't show here for simplicity

BOOTSTRAP

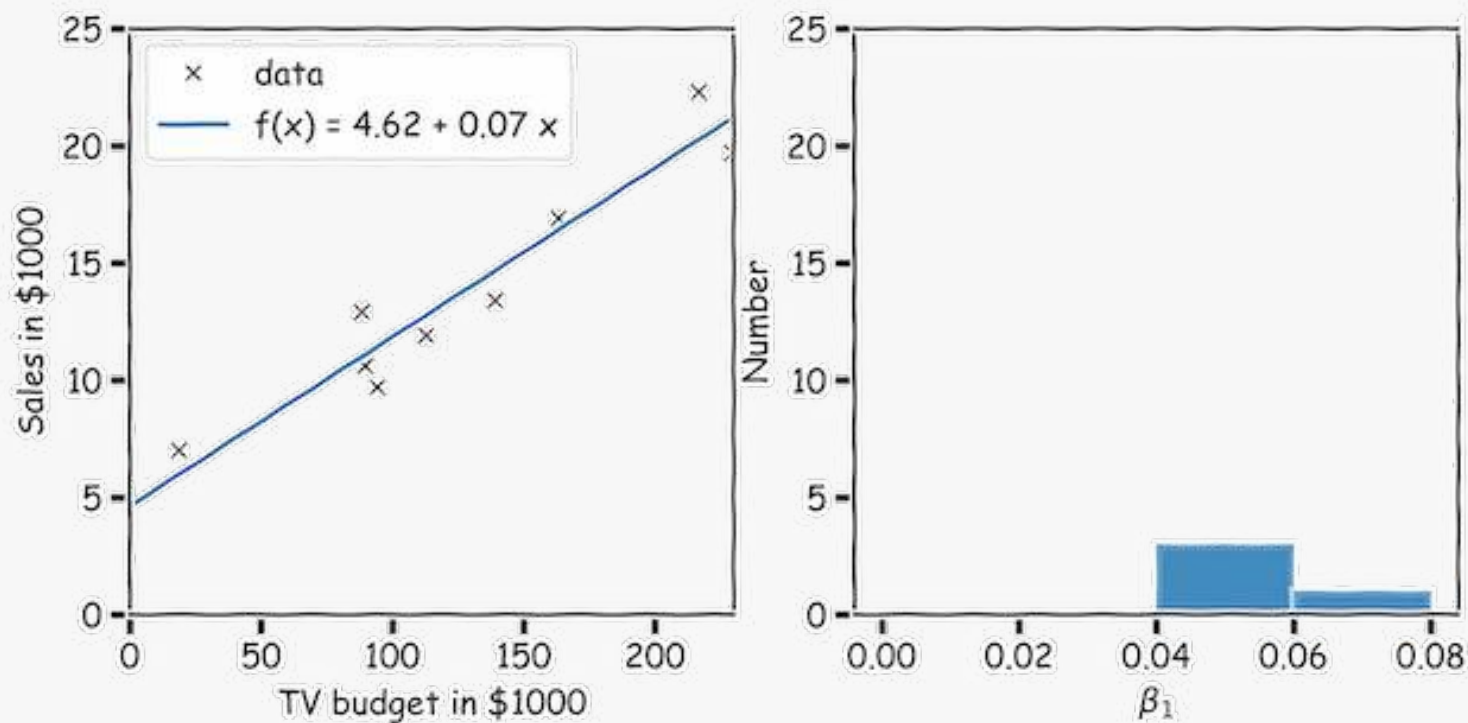
Confidence intervals for the predictors estimates (cont)

Another sample, another estimate of $\hat{\beta}_0, \hat{\beta}_1$



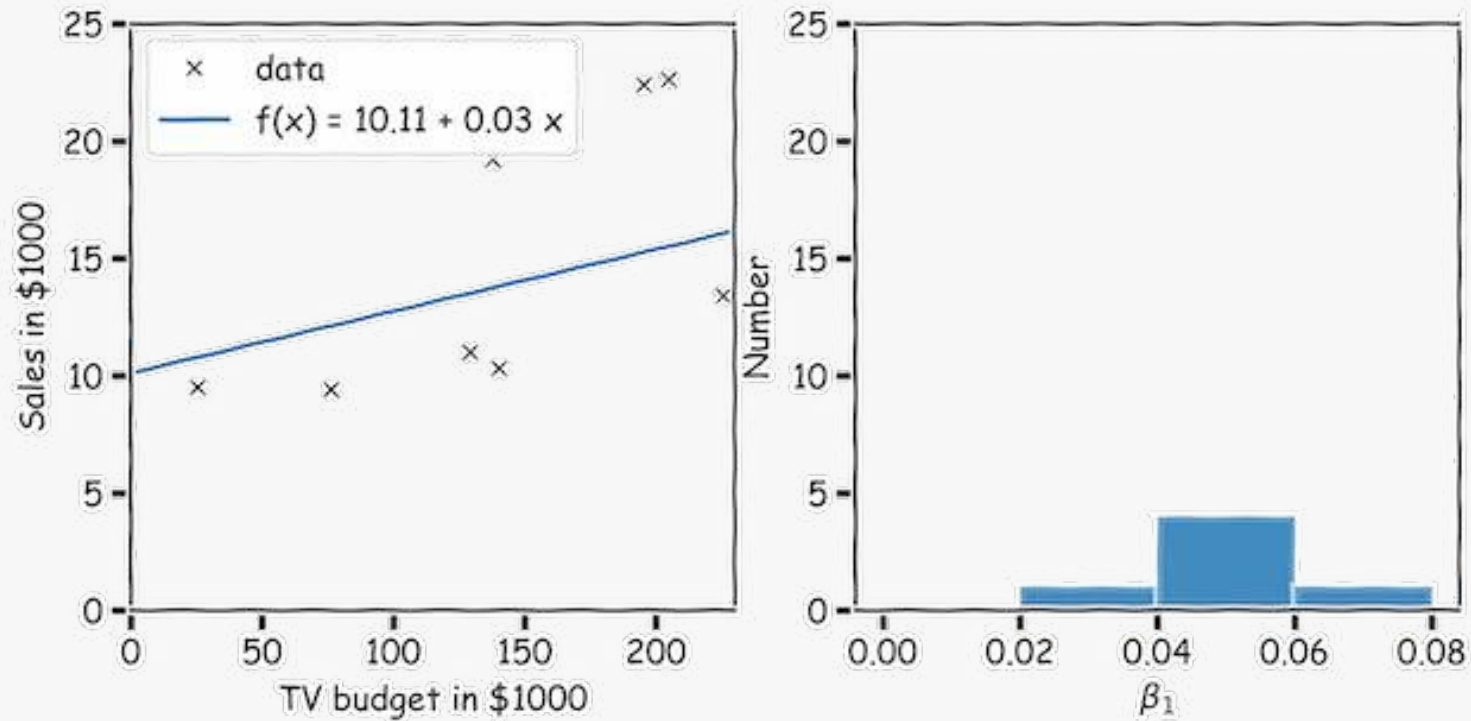
Confidence intervals for the predictors estimates (cont)

again



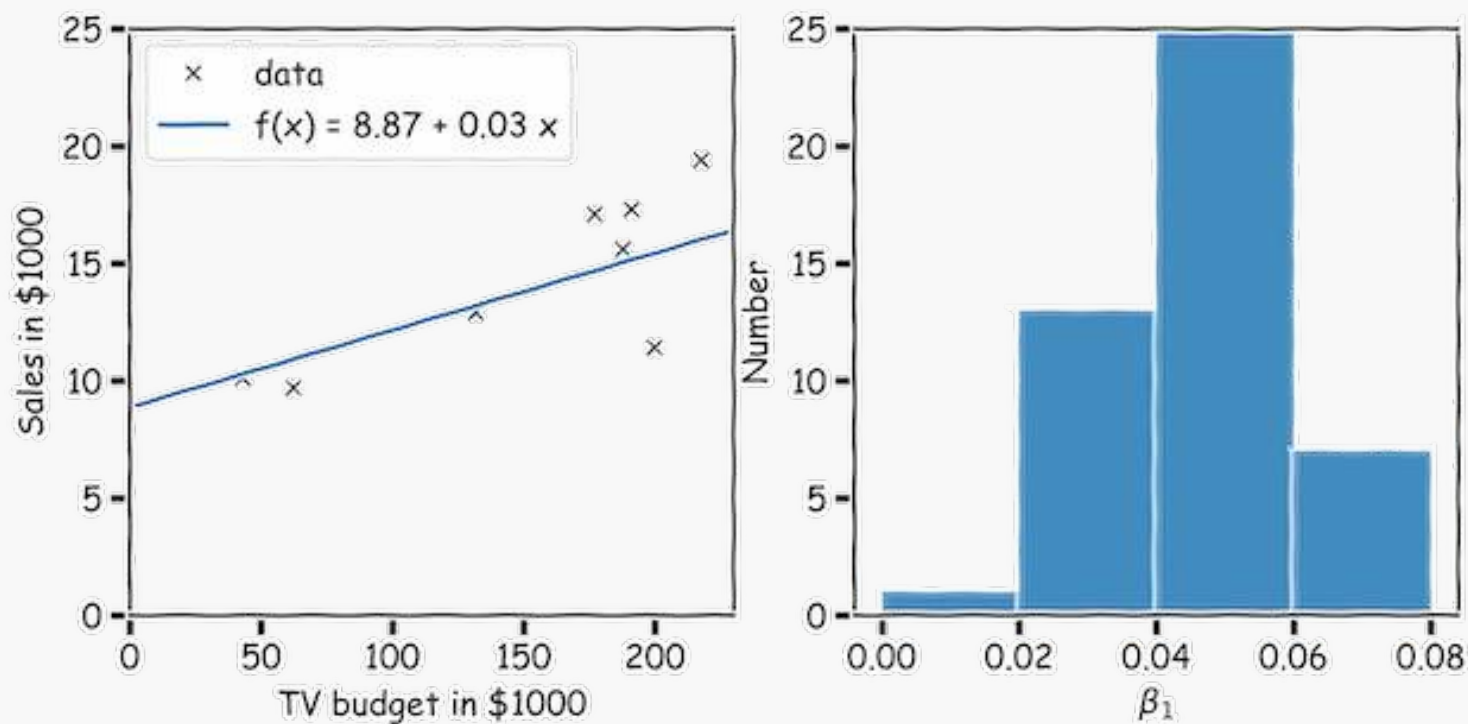
Confidence intervals for the predictors estimates (cont)

and again



Confidence intervals for the predictors estimates (cont)

repeat this for 100 times, until we have enough samples of $\hat{\beta}_0, \hat{\beta}_1$.

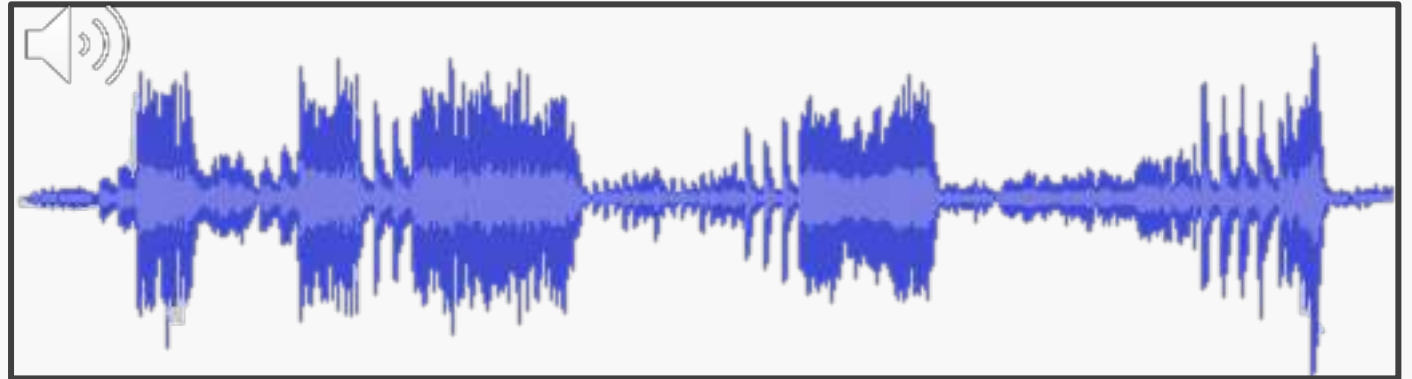


BOOTSTRAP ISSUES?

- Although easy to implement, it is computationally expensive.
- The output distribution is sensitive to input data.
- As it uses maximum likelihood point estimates, it is not a natural candidate for inference.



The marriage of Figaro



The composition is a “collection” of notes, as seen by the waveform above

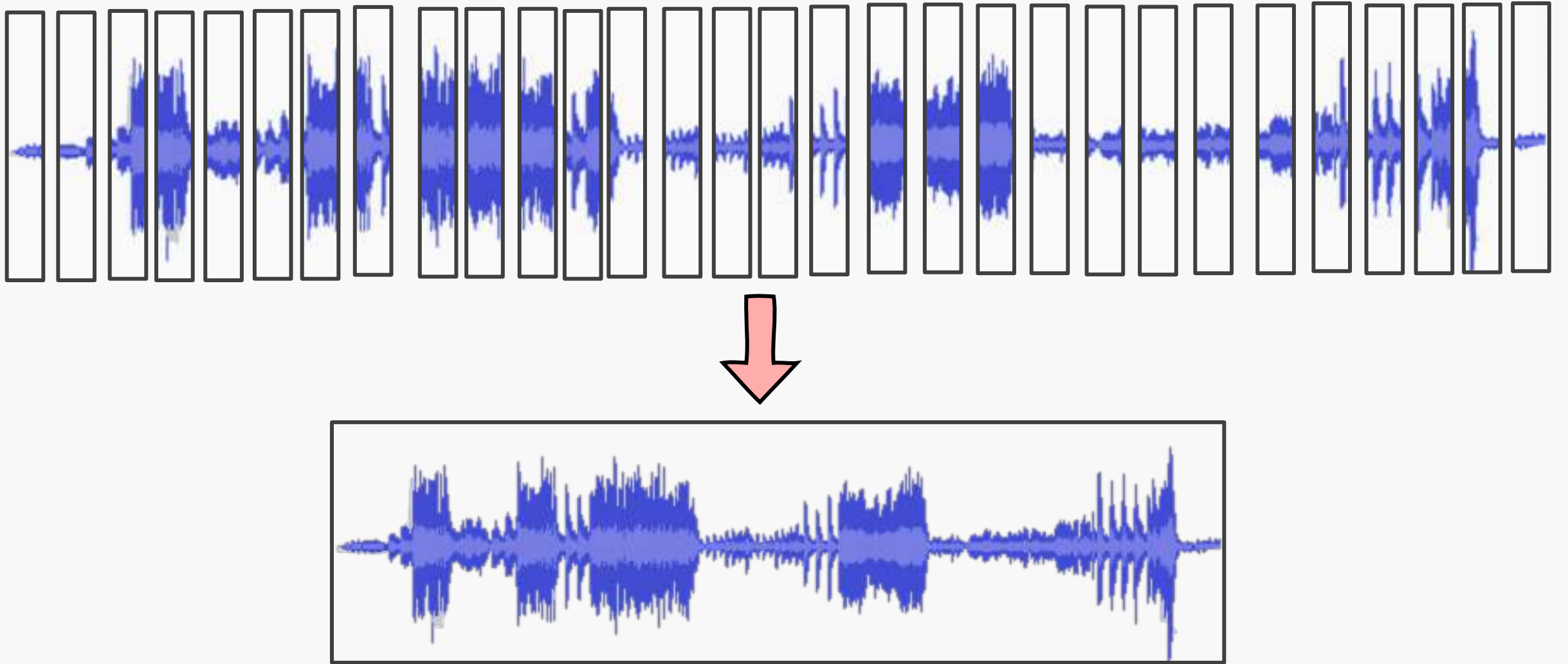


The Marriage of Figaro – Bootstrap edition



MysteryGuitarMan – [The marriage of Figaro](#)

The Marriage of Figaro – Bootstrap edition



What we want

INFERENCE WISHLIST?

- We want the full probability distribution of the parameters given the data, i.e., let $\beta = [\beta_0, \beta_1]$, then we are looking for $P(\beta | data)$.
- We want the **posterior predictive** distribution of the output given the data. i.e., $P(Y|X)$.

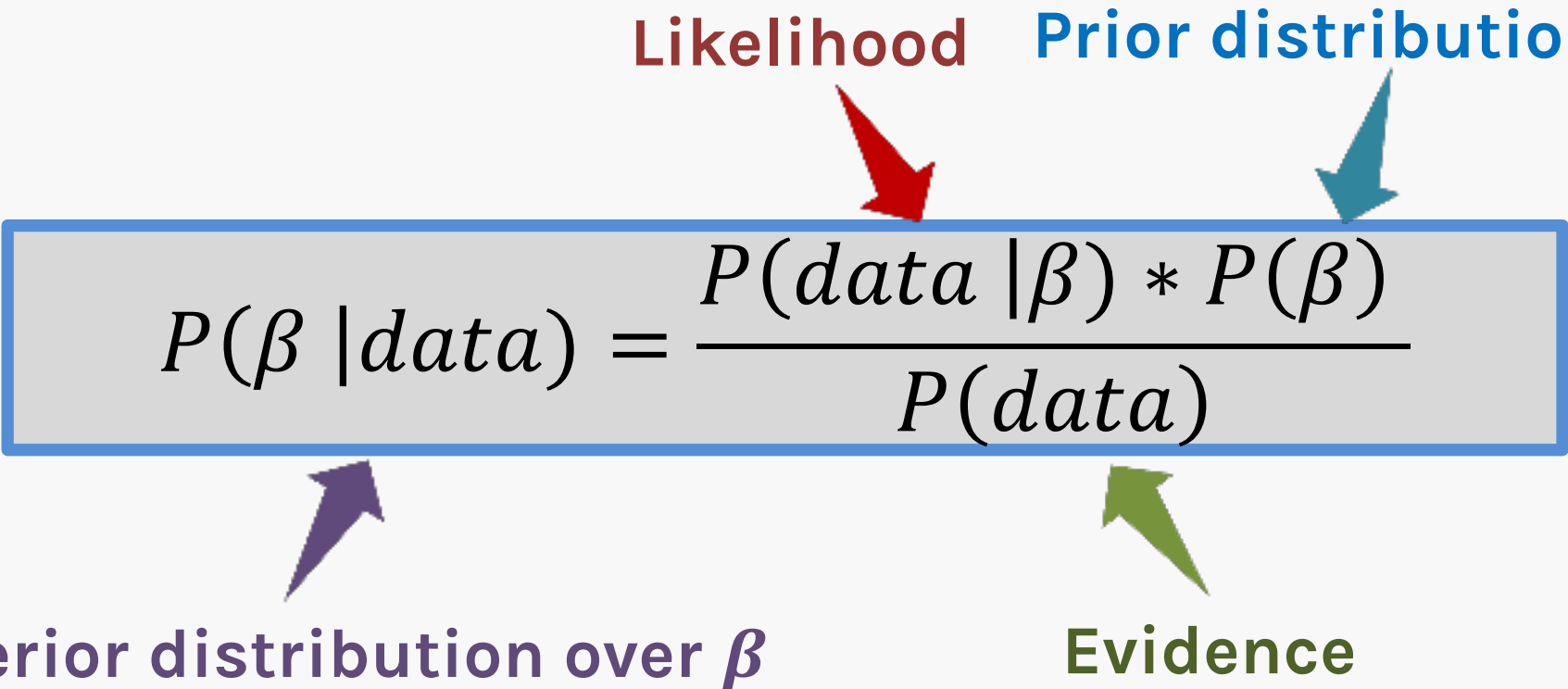
We could use Bayes Theorem

$$P(\beta | data) = \frac{P(data | \beta) * P(\beta)}{P(data)}$$



RECAP: Bayes Theorem

Likelihood Prior distribution over β


$$P(\beta | data) = \frac{P(data | \beta) * P(\beta)}{P(data)}$$

Posterior distribution over β

Evidence

Evidence

$$P(data) = \int P(data|\beta)P(\beta)d\beta$$

ISSUES WITH ANALYTICAL FORM

- Unless for very simple distributions, the **evidence**, $P(data)$ is **hard to compute**.
- It is difficult to express the posterior as a closed form, which we will need to perform inference.

$$P(\beta|data) = \frac{P(data|\beta) * P(\beta)}{\int P(data|\beta)P(\beta)d\beta}$$


This product may not have a closed form solution

This integral is very hard to compute



ISSUES WITH ANALYTICAL FORM

- Unless for very simple distributions, the **evidence**, $P(data)$ is **hard to compute**.
- It is difficult to express the posterior as a closed form, which we will need to perform inference.



This is a problem!


$$P(\beta|data) = \frac{P(data|\beta) * P(\beta)}{\int P(data|\beta)P(\beta)d\beta}$$

This product may not have a closed form solution

This integral is very hard to compute

ISSUES WITH ANALYTICAL FORM

- Unless for very simple distributions, the **evidence**, $P(data)$ is **hard to compute**.
- It is difficult to express the posterior as a closed form, which we will need to perform inference.



Maybe you
should look at
my notes

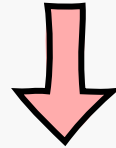
$$P(\beta|data) = \frac{P(data|\beta) * P(\beta)}{\int P(data|\beta)P(\beta)d\beta}$$

This product may
not have a closed
form solution

This integral is very hard to
compute

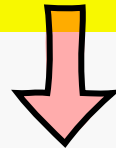
RECAP: Bayes Theorem

$$P(\beta|data) = \frac{P(data|\beta) * P(\beta)}{\int P(data|\beta)P(\beta)d\beta}$$



$$P(\beta|data) \propto P(data|\beta) * P(\beta)$$

For any **given** value of β_0, β_1 we can find the posterior probability **up to a proportionality constant**:

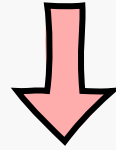


$$P(\beta|data) = C * P(data|\beta) * P(\beta)$$



RECAP: Bayes Theorem

For any **given** value of β_0, β_1 we can find the posterior probability **up to a proportionality constant**:



$$P(\beta|data) = C * P(data|\beta) * P(\beta)$$

LINEAR REGRESSION EXAMPLE

let: $y = f(x) + \epsilon$

where: $f(x) = \beta_0 + \beta_1 x$

Assume β_0 & β_1 have normally distributed priors $\sim \mathcal{N}(0, \sigma_{0,1})$
where $\sigma_{0,1}$ are arbitrarily large enough variance.



RECAP: Bayes Linear Regression

We assume that the likelihood $P(data|\beta)$ is also normal, $\mathcal{N}(\beta_0 + \beta_1 x, \sigma_D)$

Hence, for given values of parameters, β_0, β_1 , and for each $(x_i, y_i) \in data$

$$P((x_n, y_n)|\boldsymbol{\beta}) = \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\left(\frac{(y_n - (\beta_0 + \beta_1 x_n))^2}{2\sigma_D^2}\right)}$$

$$P(data|\boldsymbol{\beta}) = \prod_n \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\left(\frac{(y_n - (\beta_0 + \beta_1 x_n))^2}{2\sigma_D^2}\right)}$$



RECAP: Bayes Linear Regression

Calculating the prior distribution is much easier for the chosen values of β_0 & β_1

$$P(\beta) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\beta_0^2}{2\sigma_0^2}} * \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{\beta_1^2}{2\sigma_1^2}}$$

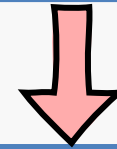
TECHNICAL DETAIL

Here we assume that the parameters β_0 & β_1 are independent random variables, hence the joint distribution (β_0, β_1) is the product of the individual distributions



Putting it all together

$$P(\beta|data) = C * P(data|\beta^i) * P(\beta^i)$$



$$P(\beta|data) = \prod_{n=1}^N \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\frac{(y_n - (\beta_0 + \beta_1 x_n))^2}{2\sigma_D^2}} * \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\beta_0^2}{2\sigma_0^2}} * \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{\beta_1^2}{2\sigma_1^2}}$$

Now we can crunch some numbers

Proportionality constant

Likelihood

Prior over β



INFERENCE WISHLIST?

Even if we cannot get the analytical solution for the parameters, we can compare how *likely* a given value of $\beta^{(i)}$ is to another value $\beta^{(j)}$ using the equations from before

$$\frac{P(\beta^{(i)} | data)}{P(\beta^{(j)} | data)} = \frac{P(data | \beta^{(i)}) * P(\beta^{(i)})}{P(data | \beta^{(j)}) * P(\beta^{(j)})}$$

This way, we are still using Bayesian inference, and we can work with equations that are computable.



A (very) brief recap of sampling

$$\frac{P(\beta^{(i)}|data)}{P(\beta^{(j)}|data)} = \frac{P(data|\beta^{(i)}) * P(\beta^{(i)})}{P(data|\beta^{(j)}) * P(\beta^{(j)})}$$

- Sampling methods using Bayesian inference with **relative comparison** between candidate points are popular methods to calculate parameter distributions.
- Popular sampling methods include **Monte Carlo sampling** and **Markov Chain Monte Carlo (MCMC)** sampling methods.
- We will consider a much popular variant of MCMC, **Metropolis aka random walk** for our discussion, but other variants can also be used to do posterior distributions.

But how can we find the parameter distributions by such a comparison?



MCMC Sampling



MCMC - Metropolis Hastings

Metropolis-Hastings is very powerful and a widely-used sampler.

If we are looking for the distribution for some parameter, θ , it reduces the problem of sampling from a difficult distribution $p(\theta|data)$ to:

- making proposals: $q(\theta^{(j)}|\theta^{(j-1)})$
- evaluating ratios: $\frac{p(\theta^*|data)/q(\theta^*|\theta^{(j-1)})}{p(\theta^{(j-1)}|data)/q(\theta^{(j-1)}|\theta^*)}$

The proposals can be trivial, for e.g., random walk (choose $\theta^{(j)}$ from a normal distribution centered at $\theta^{(j-1)}$).

Note: Only efficiency, not correctness is affected by the proposal.



The Metropolis-Hastings algorithm is outlined below:

1. Select an initial value θ_0

2. For $i = 1, \dots, m$ repeat:

a) Draw a candidate $\theta^* \sim q(\theta^* | \theta^{(j-1)})$

$$b) \quad \alpha = \frac{p(\theta^* | data) / q(\theta^* | \theta^{(j-1)})}{p(\theta^{(j-1)} | data) / q(\theta^{(j-1)} | \theta^*)} = \frac{p(\theta^* | data) q(\theta^{(j-1)} | \theta^*)}{p(\theta^{(j-1)} | data) q(\theta^* | \theta^{(j-1)})}$$

c) If $\alpha \geq 1$, accept θ^* & set $\theta^{(j)} \leftarrow \theta^*$

Else if $0 < \alpha < 1$ accept θ^* & set $\theta^{(j)} \leftarrow \theta^*$ with probability α

reject θ^* & set $\theta^{(j)} \leftarrow \theta^{(j-1)}$ with probability $1 - \alpha$



The Metropolis-Hastings algorithm is outlined below:

1. Select an initial value θ_0

2. For $i = 1, \dots, m$ repeat:

a) Draw a candidate $\theta^* \sim q(\theta^* | \theta^{(j-1)})$

b)
$$\alpha = \frac{p(\theta^* | \text{data}) / q(\theta^* | \theta^{(j-1)})}{p(\theta^{(j-1)} | \text{data}) / q(\theta^{(j-1)} | \theta^*)} = \frac{p(\theta^* | \text{data}) q(\theta^{(j-1)} | \theta^*)}{p(\theta^{(j-1)} | \text{data}) q(\theta^* | \theta^{(j-1)})}$$

c) If $\alpha \geq 1$, accept θ^* & set $\theta^{(j)} \leftarrow \theta^*$

Else if $0 < \alpha < 1$ accept θ^* & set $\theta^{(j)} \leftarrow \theta^*$ with probability α
reject θ^* with probability $1 - \alpha$

$q(\cdot)$ is the proposal. For example,
 $\theta^* \sim N(\theta^{(j-1)}, \sigma)$

How? Draw a random number, ρ .
If $\rho > (1 - \alpha)$ accept.



Metropolis Hastings – Random Walk

A special case of the Metropolis-Hastings algorithm is the **Metropolis** where the distribution q is symmetric i.e.,

$$q(\theta^* | \theta^{(j-1)}) = q(\theta^{(j-1)} | \theta^*)$$

This makes the algorithm much simpler, as

$$\alpha = \frac{p(\theta^* | data) q(\theta^{(j-1)} | \theta^*)}{p(\theta^{(j-1)} | data) q(\theta^* | \theta^{(j-1)})}$$



Metropolis Hastings – Random Walk

A special case of the Metropolis-Hastings algorithm is the **Metropolis** where the distribution q is symmetric i.e.,

$$q(\theta^* | \theta^{(j-1)}) = q(\theta^{(j-1)} | \theta^*)$$

This makes the algorithm much simpler, as

$$\alpha = \frac{p(\theta^* | data) q(\theta^{(j-1)} | \theta^*)}{p(\theta^{(j-1)} | data) q(\theta^* | \theta^{(j-1)})}$$

Metropolis Hastings – Random Walk

A special case of the Metropolis-Hastings algorithm is the **Metropolis** where the distribution q is symmetric i.e.,

$$q(\theta^* | \theta^{(j-1)}) = q(\theta^{(j-1)} | \theta^*)$$

This makes the algorithm much simpler, as

$$\alpha = \frac{p(\theta^* | data) q(\theta^{(j-1)} | \theta^*)}{p(\theta^{(j-1)} | data) q(\theta^* | \theta^{(j-1)})} = \frac{p(\theta^* | data)}{p(\theta^{(j-1)} | data)}$$

Metropolis Hastings – Random Walk

A special case of the Metropolis-Hastings algorithm is the **Metropolis** where the distribution q is symmetric i.e.,

$$q(\theta^* | \theta^{(j-1)}) = q(\theta^{(j-1)} | \theta^*)$$

This makes the algorithm much simpler, as

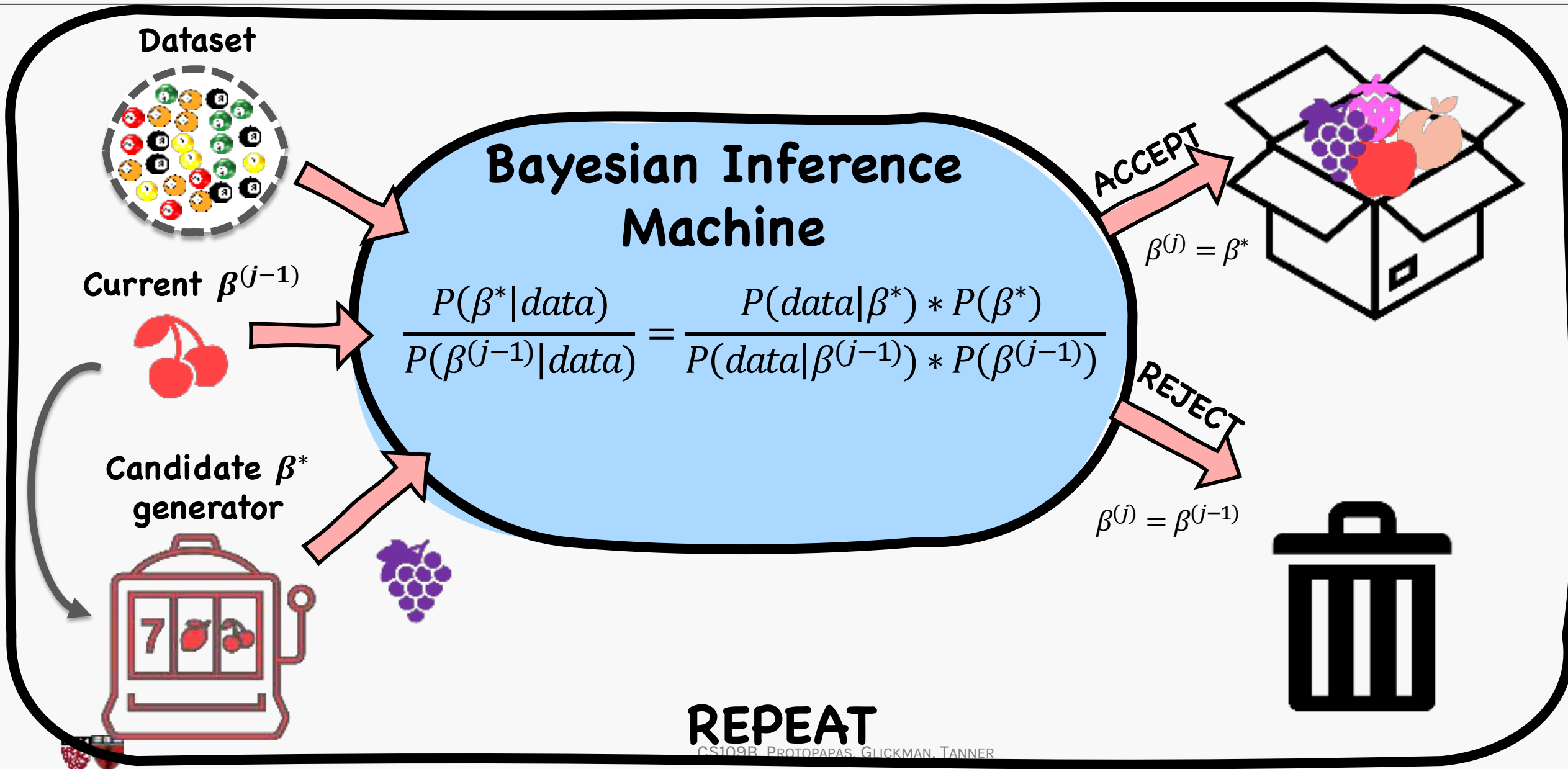
$$\alpha = \frac{p(\theta^* | data) q(\theta^{(j-1)} | \theta^*)}{p(\theta^{(j-1)} | data) q(\theta^* | \theta^{(j-1)})} = \frac{p(\theta^* | data)}{p(\theta^{(j-1)} | data)}$$

We can show using Bayes theorem:

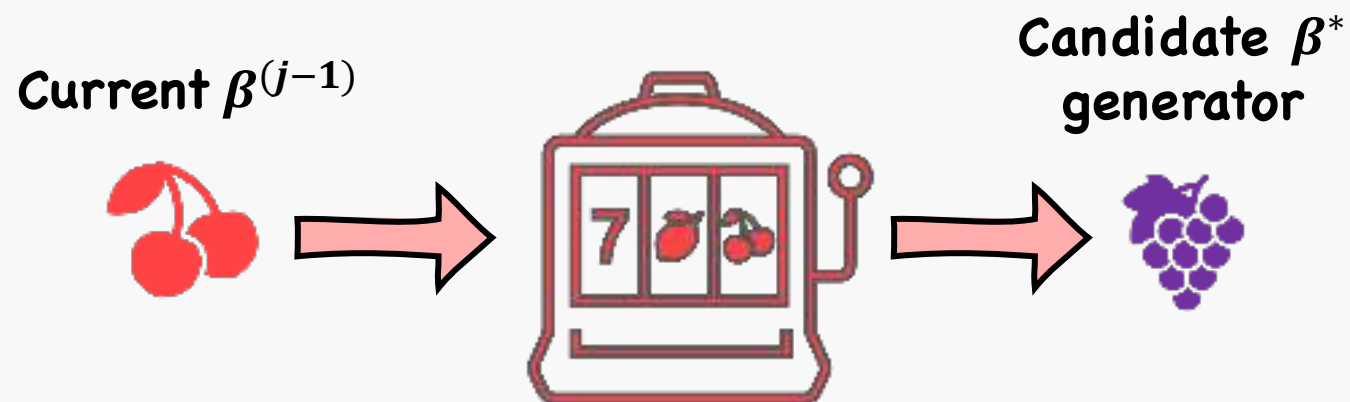
$$\alpha = \frac{p(data | \theta^*)}{p(data | \theta^{(j-1)})} \frac{p(\theta^*)}{p(\theta^{(j-1)})}$$



MCMC Factory



MCMC Factory - What is a generator



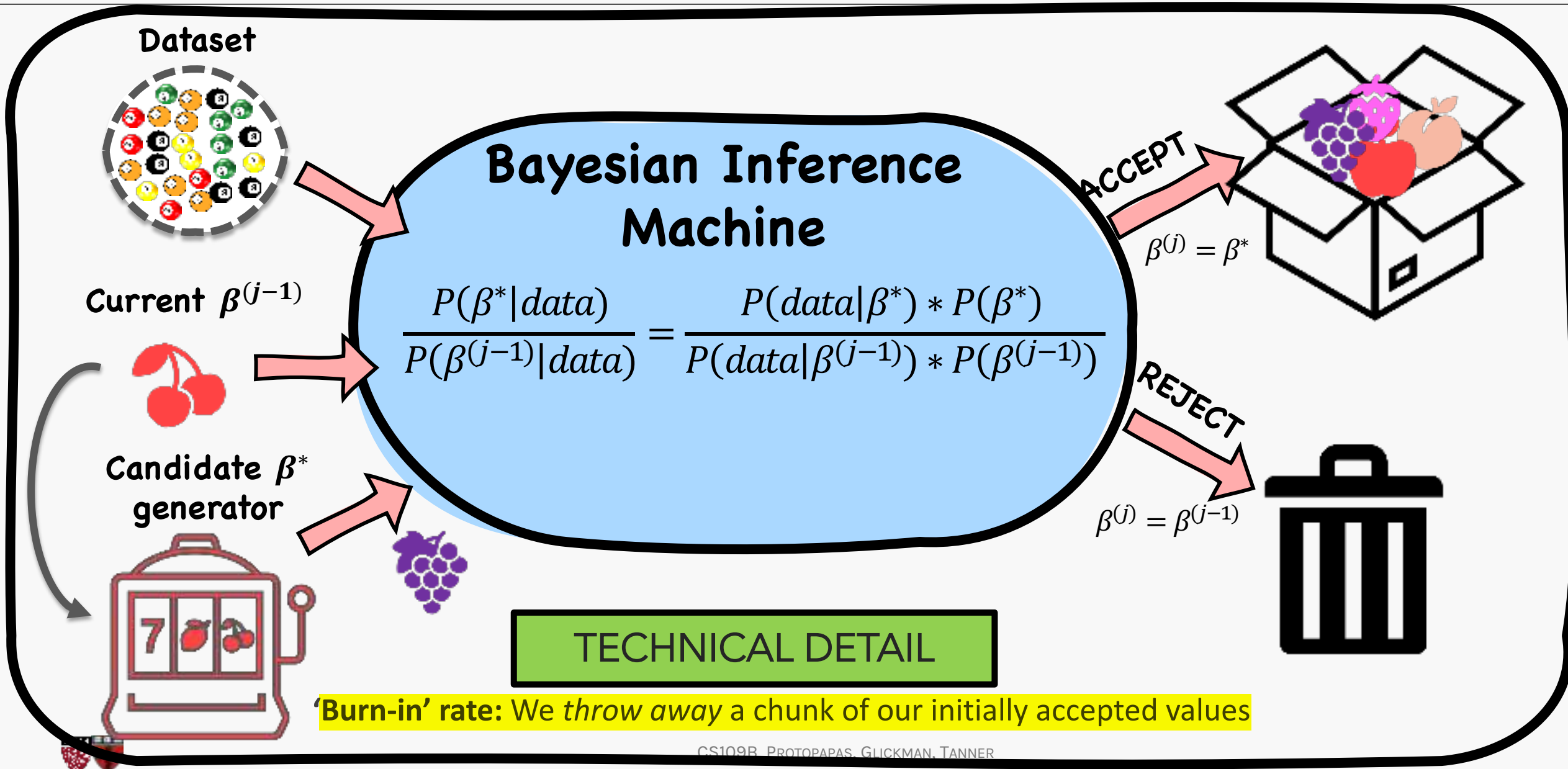
- The candidate β^* generator takes the currently accepted $\beta^{(j-1)}$ value and draws a value from a distribution, for example:

$$\beta^* \sim \mathcal{N}(\beta^{(j-1)}, \sigma)$$

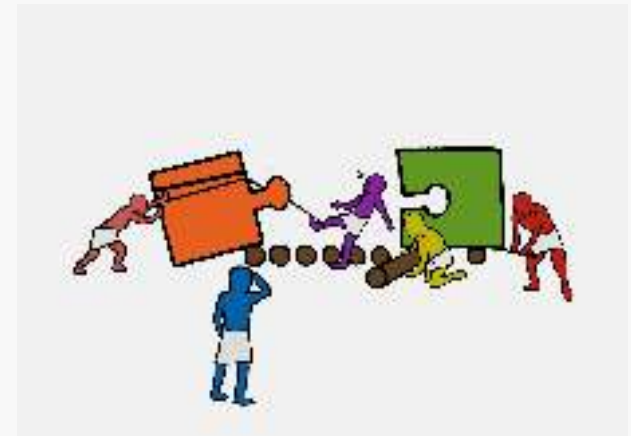
- This ensures that we don't get wildly crazy candidates, but ones quite similar to the current β .



MCMC Factory

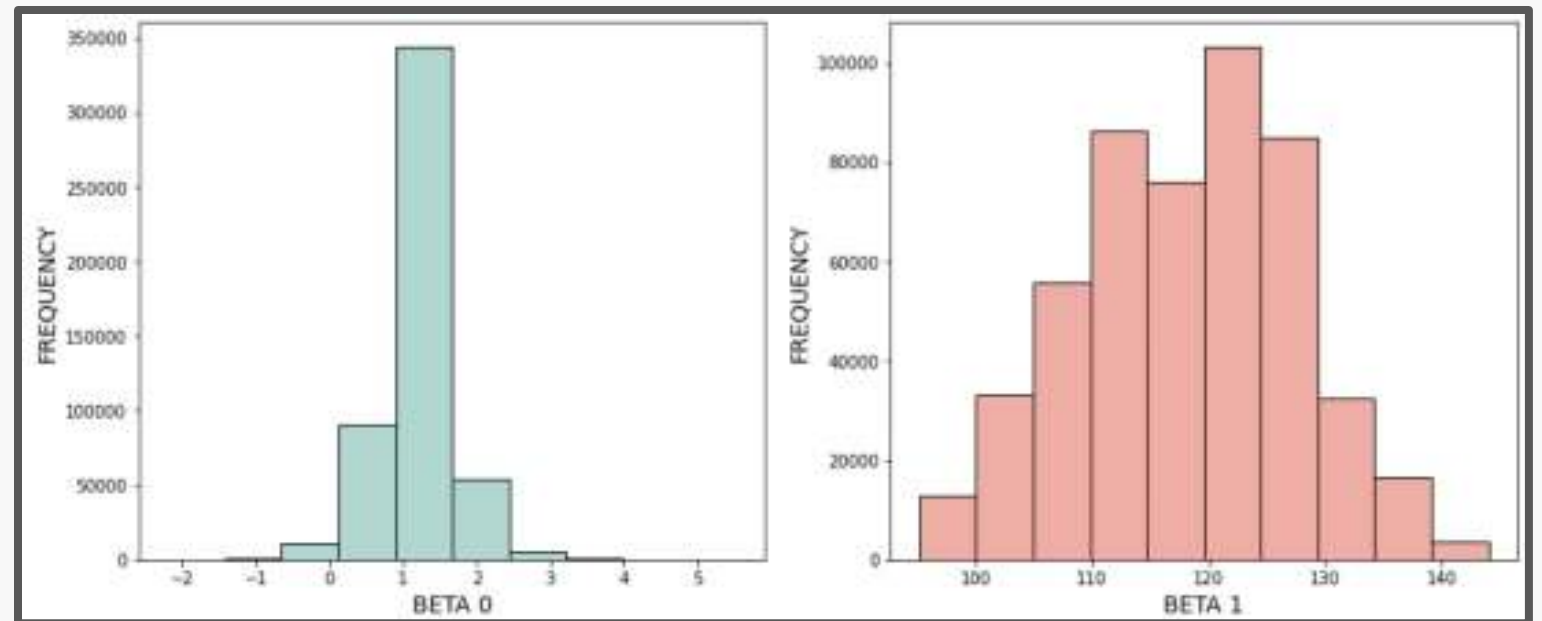


Exercise: Linear Regression MCMC from scratch



The aim of this exercise is to perform Monte Carlo Markov Chain (MCMC) from scratch for linear regression.

On completing the exercise you should be able to see the following distribution. One for each of the beta value:



Time to scale the hills of
Bayesian NNs!

BAYESIAN
LINEAR REGRESSION

BAYESIAN
NEURAL NETWORKS



Bayesian Neural Network



Bayesian Linear Regression

We assume that the likelihood $P(data|\beta)$ is also normal, $\mathcal{N}(\beta_0 + \beta_1 x, \sigma_D)$

Hence, for given values of parameters, β_0, β_1 , and for each $(x_i, y_i) \in data$

$$P(\{x_n, y_n\}|\boldsymbol{\beta}) = \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\left(\frac{(y_n - (\beta_0 + \beta_1 x_n))^2}{2\sigma_D^2}\right)}$$

$$P(data|\boldsymbol{\beta}) = \prod_n \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\left(\frac{(y_n - (\beta_0 + \beta_1 x_n))^2}{2\sigma_D^2}\right)}$$



Bayesian **Neural Network**

We assume that the likelihood $P(data|\beta)$ is also normal, $\mathcal{N}(NN_W(x), \sigma_D)$

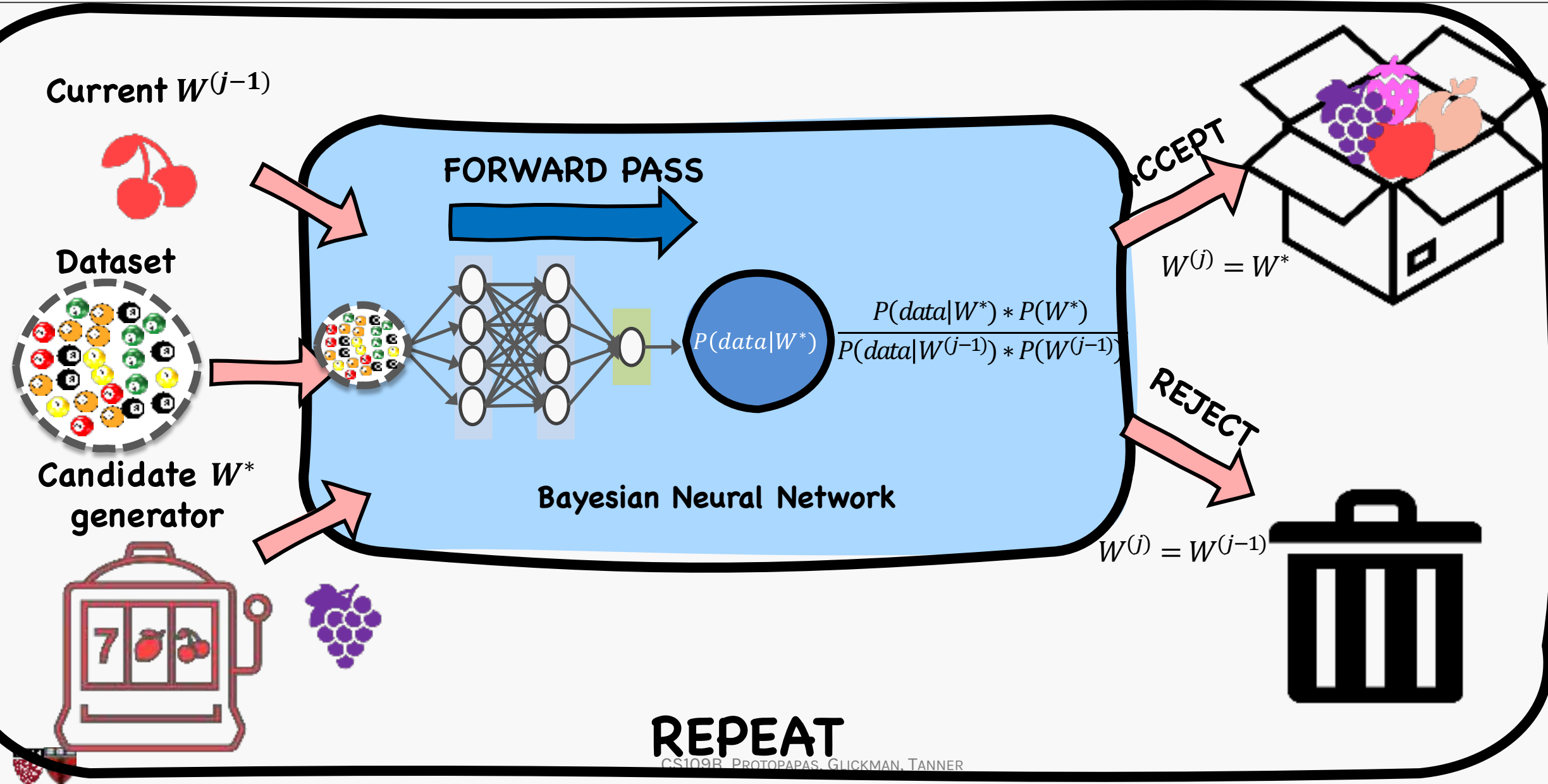
Hence, for given values of parameters, W , and for each $(x_i, y_i) \in data$

$$P(\{x_n, y_n\}|W) = \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\left(\frac{(y_n - NN_W(x_n))^2}{2\sigma_D^2}\right)}$$

$$P(data|W) = \prod_n \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\left(\frac{(y_n - NN_W(x_n))^2}{2\sigma_D^2}\right)}$$



Bayesian Neural Network

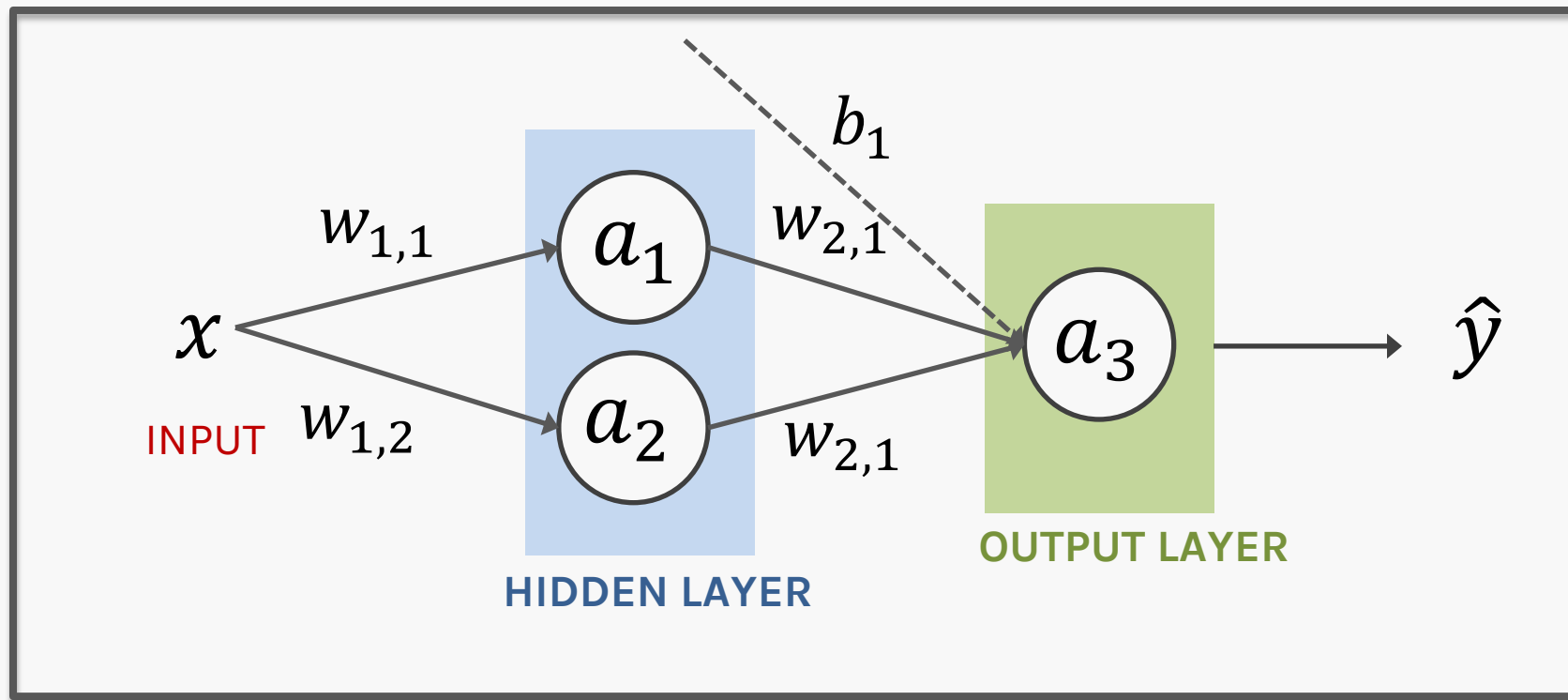




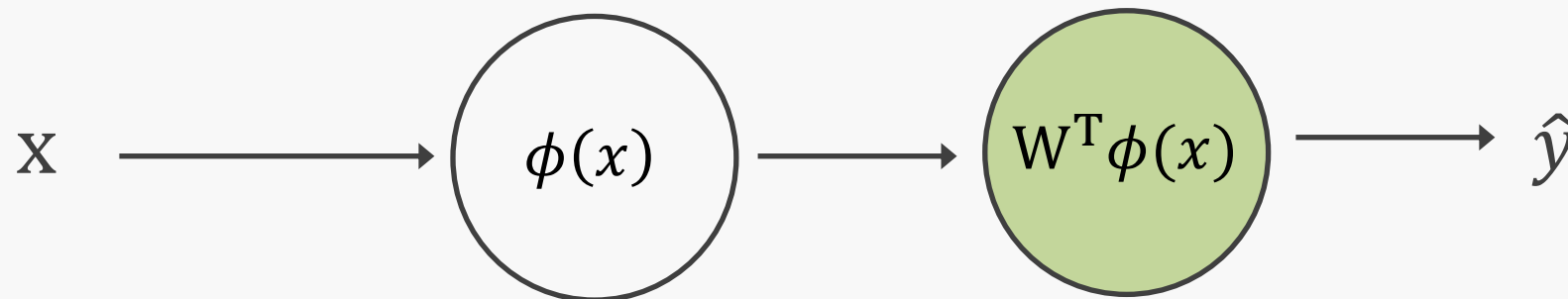
FORWARD PASS ONLY

THAT'S IT?

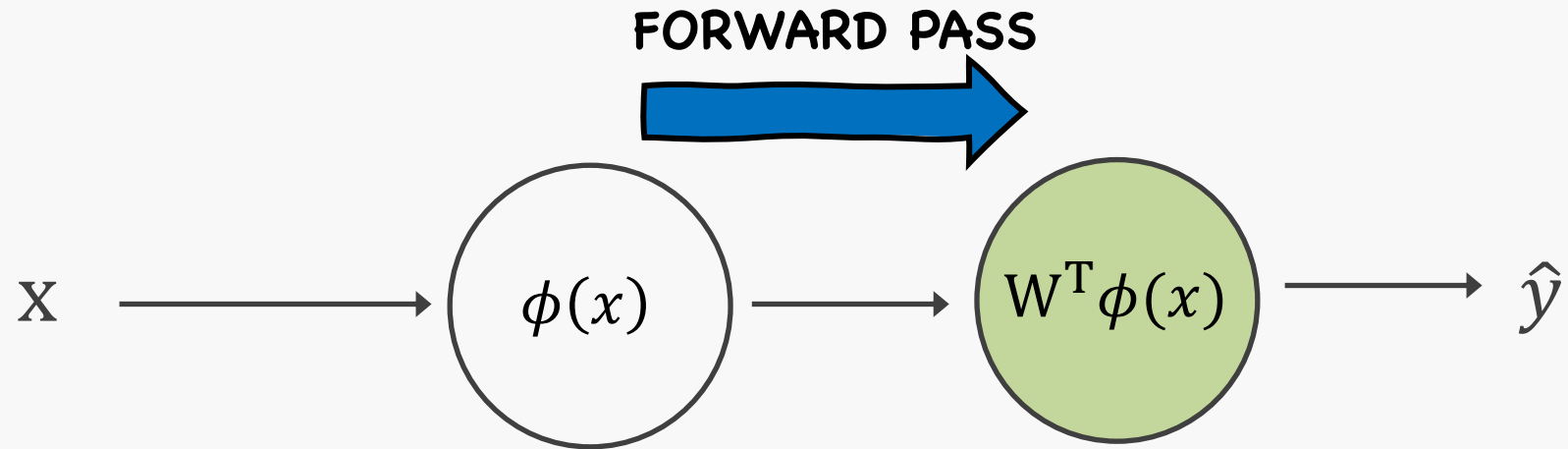
Example: A simple Bayesian Neural Network (BNN)



Neural Network Architecture



Bayesian Neural Network (BNN)



1. Select an initial value $W^{(0)}$ (this represent all weights in the network)
2. For $i = 1, \dots, m$ repeat:

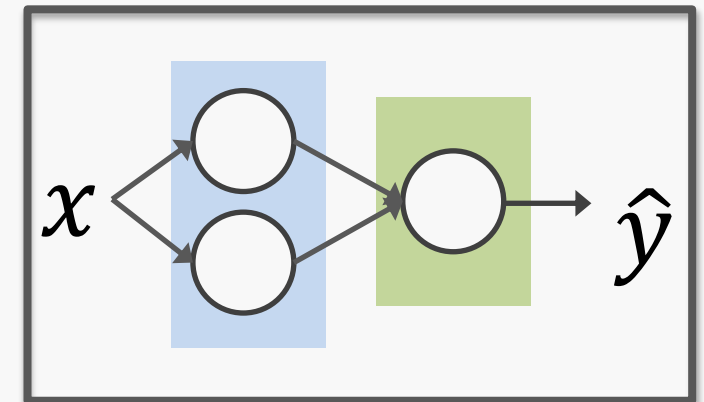
a) Draw a candidate $W^* \sim q(W^* | W^{(j-1)})$

b)
$$\alpha = \frac{p(\text{data} | W^*)}{p(\text{data} | W^{(j-1)})} \frac{p(W^*)}{p(W^{(j-1)})}$$

c) If $\alpha \geq 1$, accept W^* & set $W^{(j)} \leftarrow W^*$

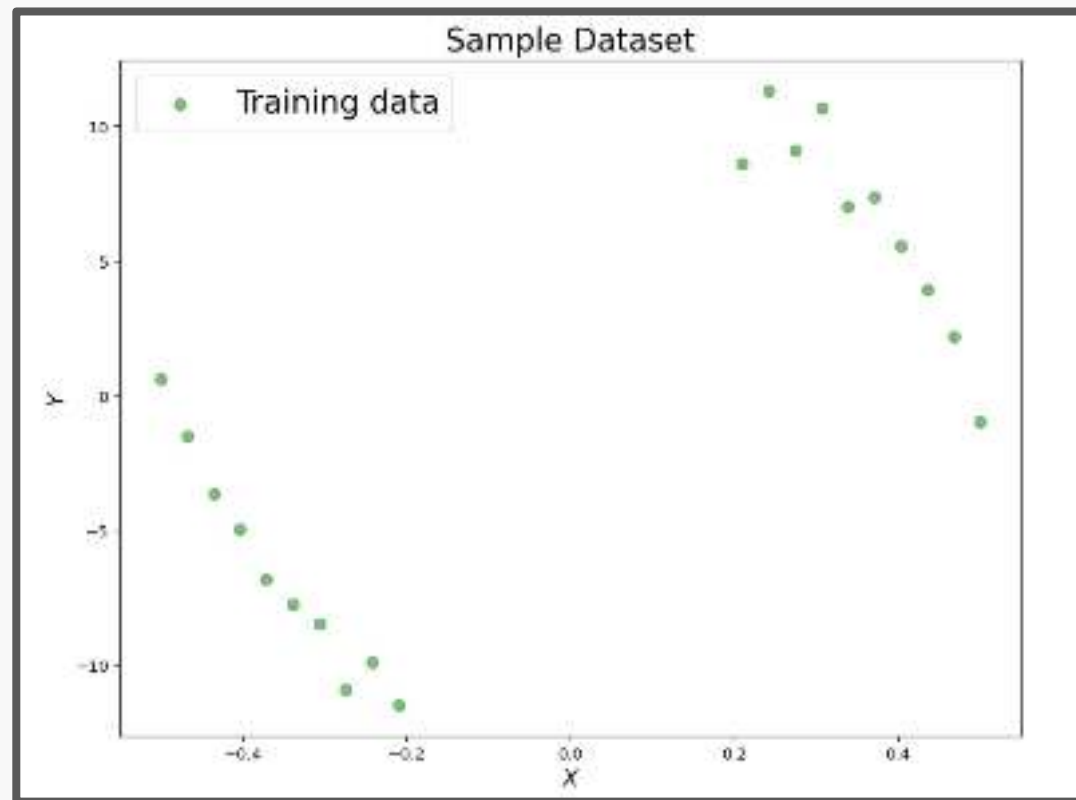
else if $0 < \alpha < 1$ **accept** W^* & set $W^{(j)} \leftarrow W^*$ with probability α

reject θ^* probability $1 - \alpha$



Bayesian Neural Network

Let us consider the dataset below for regression

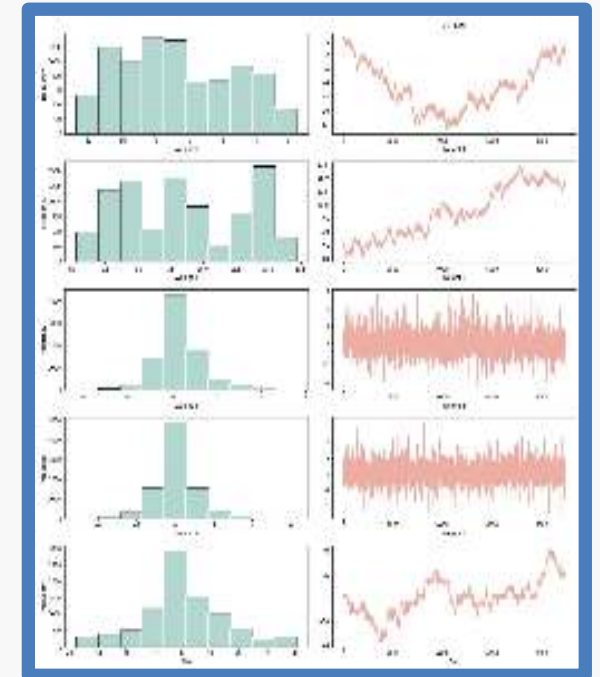
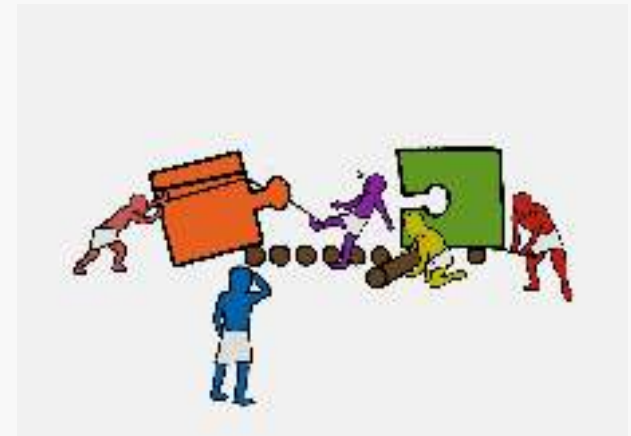


Exercise: MCMC from Scratch for Neural Networks

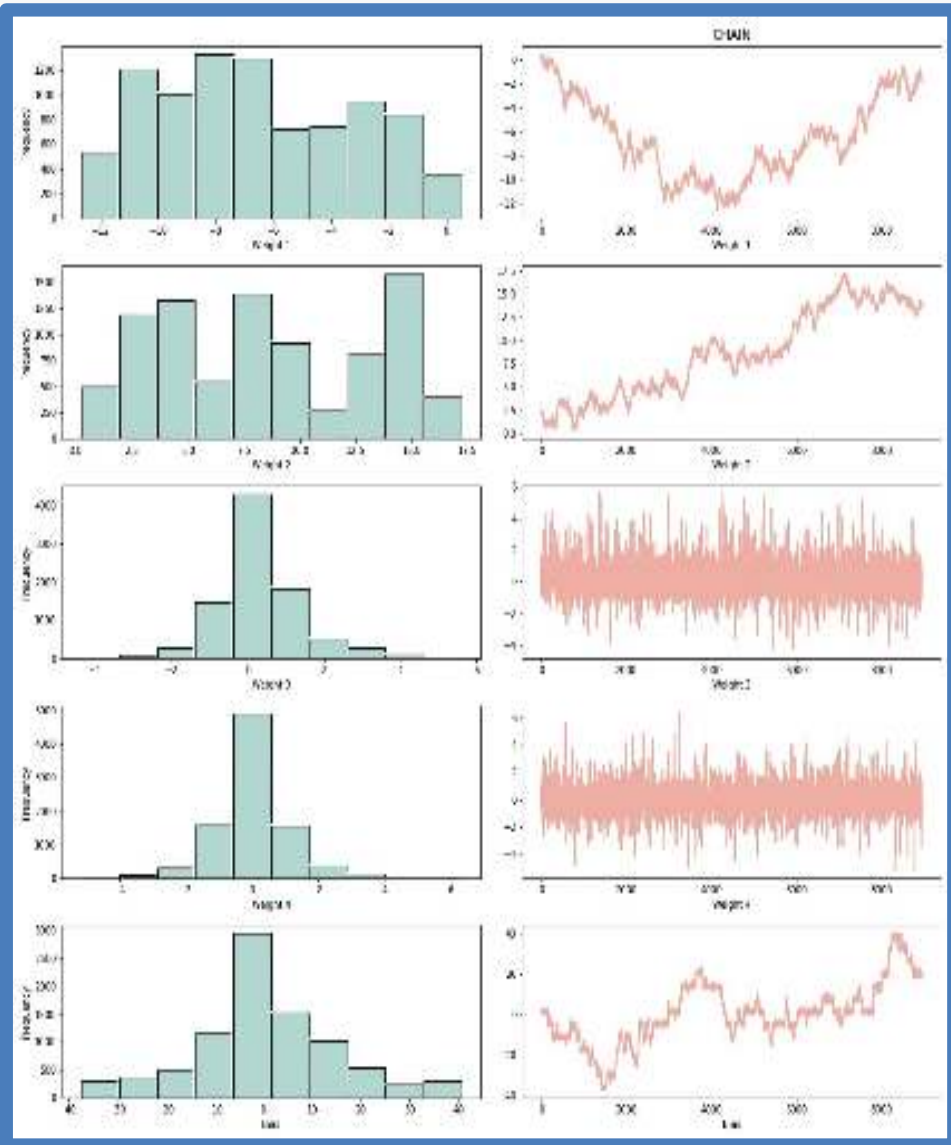
The aim of this exercise is to perform Metropolis Monte Carlo Markov Chain (MCMC) from scratch (as in exercise 1) for a simple neural network.

On completing the exercise, you should be able to see the following distribution. One for each of the beta value:

Warning: This is not going to converge unless we start very near the mode of the distribution!



FULL MCMC ISSUES



MCMC we will not work for NN's with more than a few dozens of parameters.

Why?

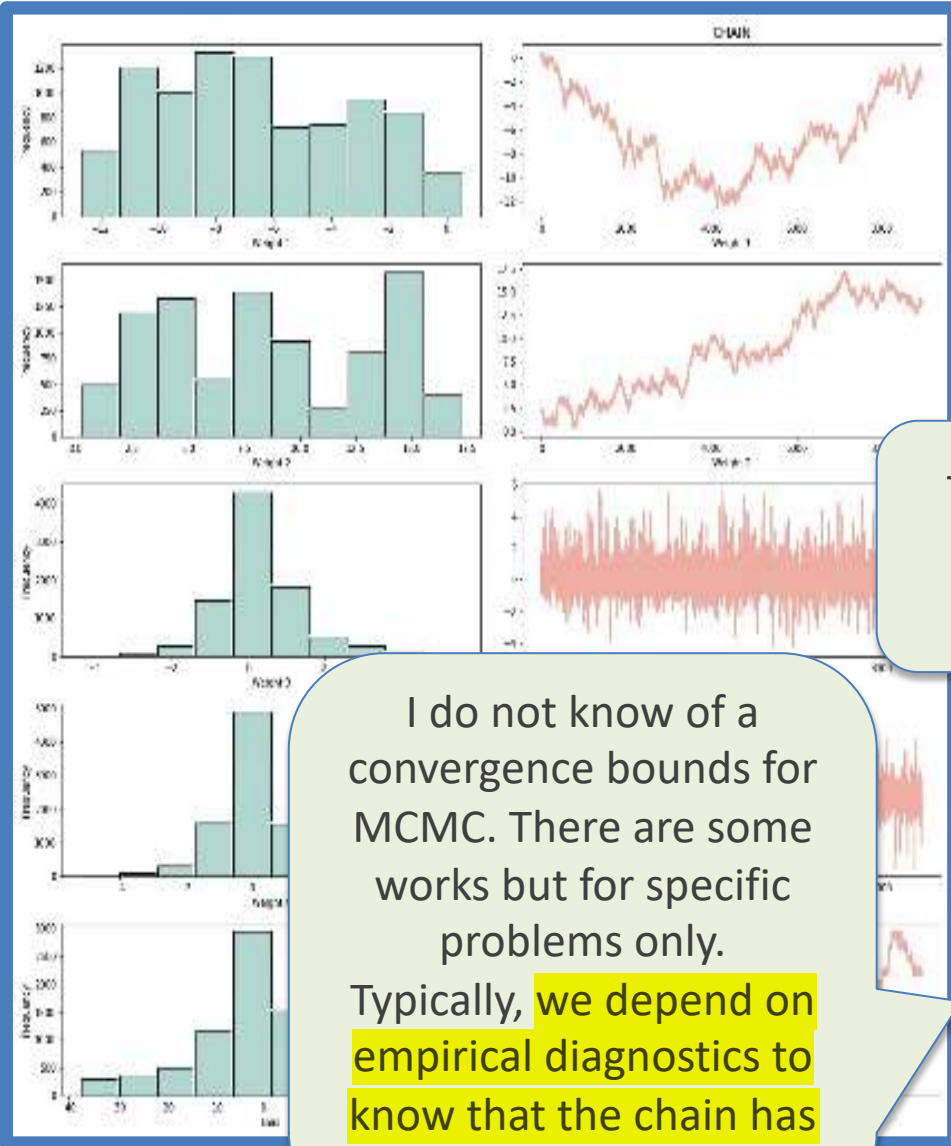
For each parameter (weight) we sample and calculate the likelihood $5 * n$ times, where n being the length of the chain.

We also throw away a significant number of samples.

Sufficient n also grows with the number of parameters and complexity of the posterior.



FULL MCMC ISSUES



I do not know of a convergence bounds for MCMC. There are some works but for specific problems only. Typically, we depend on empirical diagnostics to know that the chain has converged

MCMC we will not work for NN's with more than a few dozens of parameters.

Why?

For each parameter (we have n), we sample and calculate the likelihood $5 * n$ times, where n being the number of parameters in the chain. Typically, we throw the first 10-20% of the samples for burn in.

Five times because we usually have acceptance rate of $\sim 20\%$.

We also throw away a significant number of samples.

Sufficient n also grows with the number of parameters and complexity of the posterior.



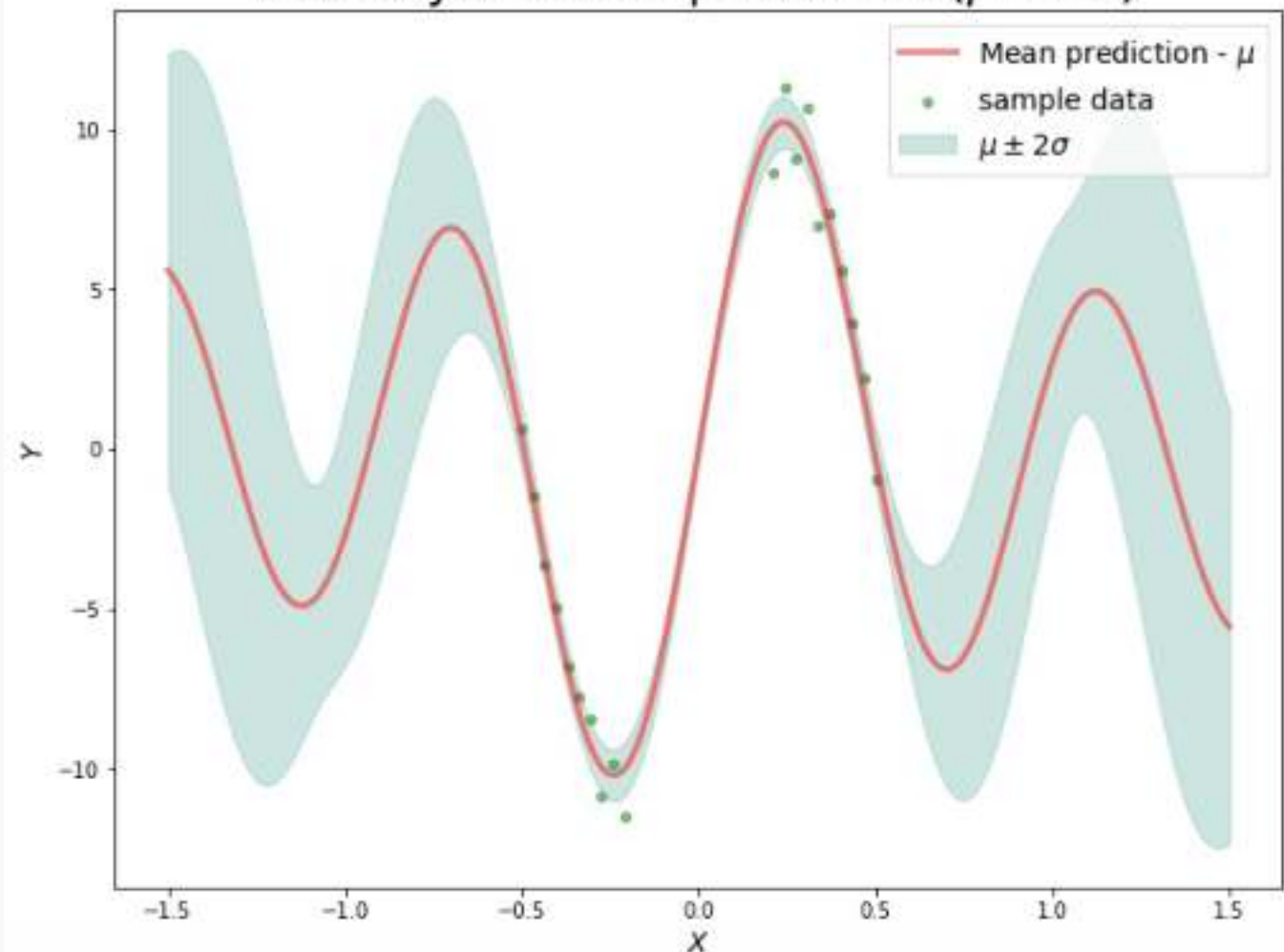
Bayesian Neural Network: Full MCMC with HMC

Instead of the simple Metropolis MCMC, we could use a more sophisticated sampler **HMC**, as we did in HW3 and lab.

Here with Tensorflow Probability

POSTERIOR DISTRIBUTION y

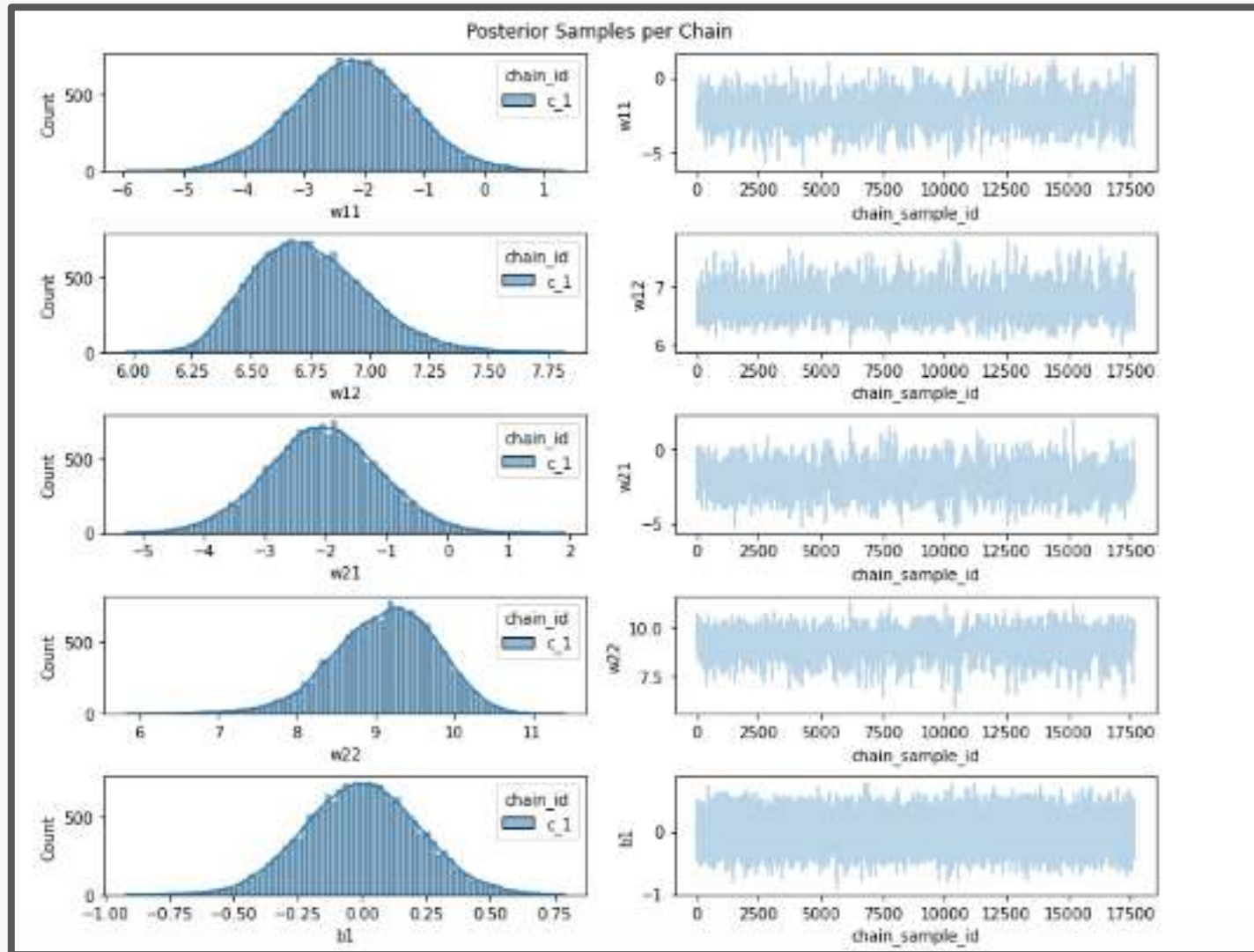
Full Bayesian NN prediction ($\mu \pm 2\sigma$)



Bayesian Neural Network: Full MCMC with **HMC**

NEURAL NET WEIGHT DISTRIBUTIONS

- The first set of plots on the right represents the distribution of the individual weights.
- The second set of plots on the right show the individual values of the weights considered while running the sampling algorithm.



Bayesian Neural Network: Full MCMC with HMC

- The first set of plots on the right represents the distribution of the individual weights.
- The second set of plots on the right show the individual values of the weights considered while running the sampling algorithm.

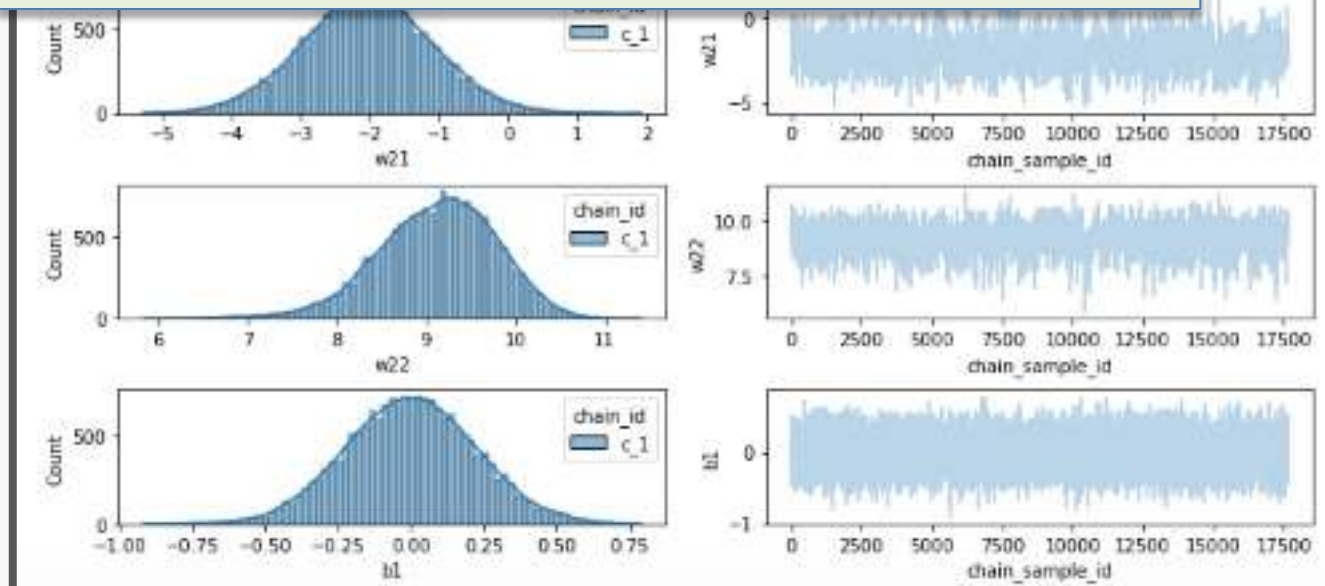
Activation: $\sin(\cdot)$

Burn rate: 30%

NEURAL NET WEIGHT DISTRIBUTIONS

HMC is a smart way of proposing which is much more efficient. It is based on introducing a “fake” variable called momentum and follow Hamiltonian mechanics to propose new variables.

https://arogozhnikov.github.io/2016/12/19/mar_kov_chain_monte_carlo.html



Bayesian Neural Network – Roadblocks

MCMC doesn't work for higher dimension

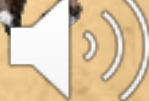




BAYESIAN
INFERENCE



BAYESIAN
NEURAL NETWORKS



Guess we'll need
an alternate route!



VARIATIONAL
AUTO-ENCODERS



End of Part 1



Bayesian Neural Network: Hacks



Pavlos
Idea #3622



We could 'estimate'
some weights, or try
other tricks

W_{11} & W_{12} ML estimate

