Latent Variables and Lossless Compression

James Townsend

Stanford IT Forum
28/10/2022

Talk outline

1. Latent variable methods
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

Talk outline

1. Latent variable *methods*
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

Talk outline

1. Latent variable *methods*
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

‘Latent variable methods’

We have data x, modelled with some distribution P(x).

Suppose modelling x is *difficult* in some way, it might be

- difficult to sample from P
- difficult to evaluate P
- difficult to compress x using P

‘Latent variable methods’

We have data x, modelled with some distribution P(x).

Suppose modelling x is *difficult* in some way, it might be

- difficult to sample from P
- difficult to evaluate P
- difficult to compress x using P

Old, useful idea:

Introduce extra randomness* z, correlated with x, such that $P(x) = \int P(x, z)dz$...

*aka entropy: $H(x, z) > H(x)$

‘Latent variable methods’

We have data x, modelled with some distribution P(x).

Suppose modelling x is *difficult* in some way, it might be

- difficult to sample from P
- difficult to evaluate P
- difficult to compress x using P

Old, useful idea:

Introduce extra randomness* z, correlated with x, such that $P(x) = \int P(x, z)dz$...

This is called a *latent* variable

*aka entropy: $H(x, z) > H(x)$

‘Latent variable methods’

We have data x, modelled with some distribution P(x).

Suppose modelling x is *difficult* in some way, it might be

- difficult to sample from P
- difficult to evaluate P
- difficult to compress x using P

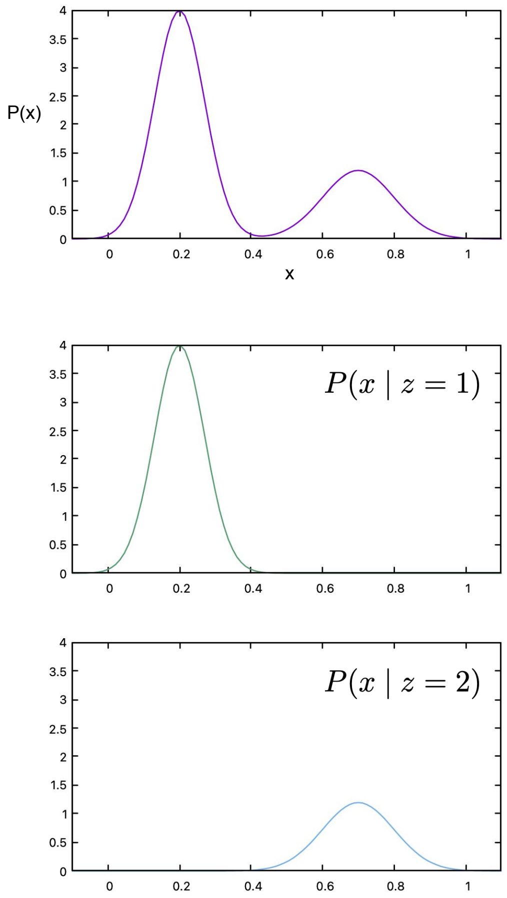
Old, useful idea:

Introduce extra randomness* z, correlated with x, such that $P(x) = \int P(x, z)dz$...

*aka entropy: $H(x, z) > H(x)$

‘Latent variable methods’

$$P(x) = c_1 N(x; \mu_1, \sigma_1) + c_2 N(x; \mu_2, \sigma_2)$$



We have data x, modelled with some distribution P(x).

Suppose modelling x is *difficult* in some way, it might be

- difficult to sample from P
- difficult to evaluate P
- difficult to compress x using P

Old, useful idea:

Introduce extra randomness* z, correlated with x, such that P(x) = ∫P(x, z)dz...

*aka entropy: H(x, z) > H(x)

'Latent variable methods'

We have data x, mod

Suppose modelling

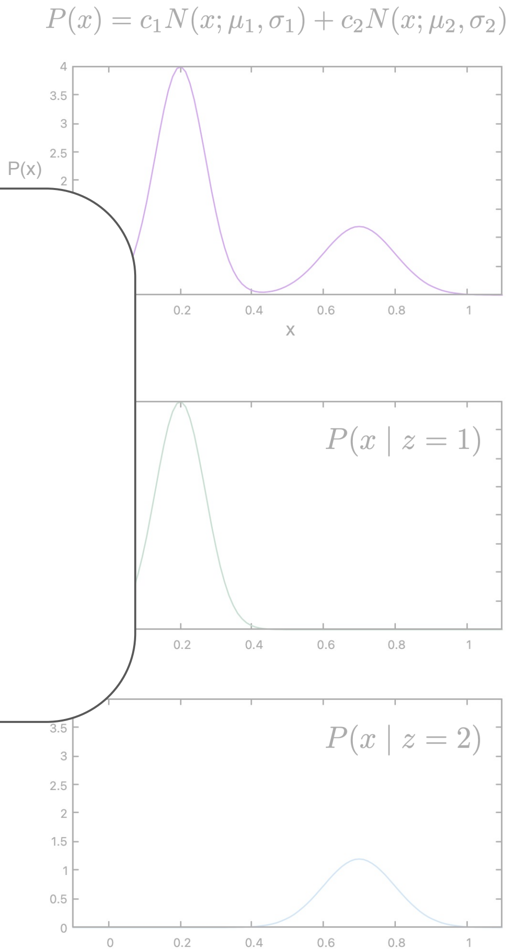
- difficult to samp
- difficult to evalu
- difficult to comp

Old, useful idea:

Introduce extra randomness* z, correlated with x, such that P(x) = ∫P(x, z)dz...

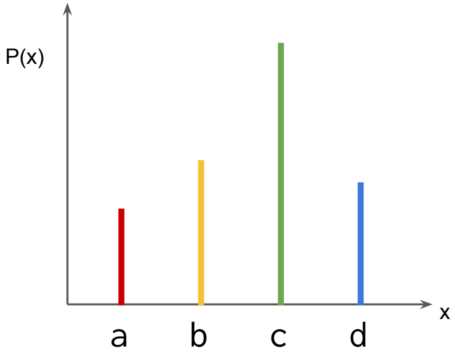
*aka entropy: H(x, z) > H(x)

Claim: this is a common pattern.
It's useful to notice it.



Example 1: sampling a discrete random variable

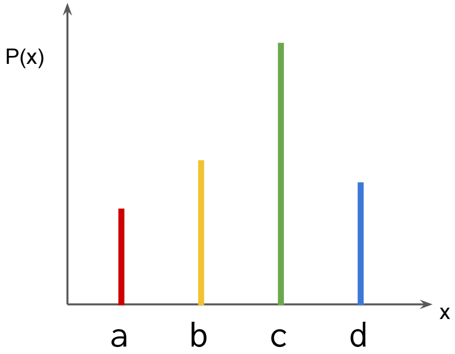
How do you sample a discrete random variable?



x	a	b	c	d
P(x)	0.17	0.24	0.40	0.19

Example 1: sampling a discrete random variable

How do you sample a discrete random variable?



x	a	b	c	d
P(x)	0.17	0.24	0.40	0.19

Answer: compute

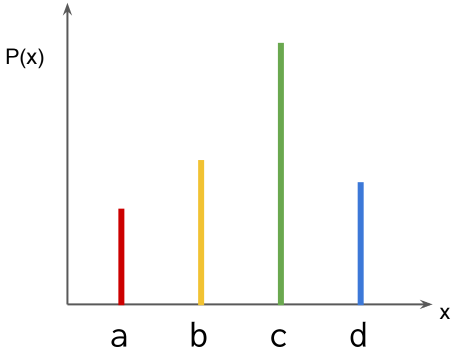
F(x)	0.00	0.17	0.41	0.81

Sample u ~ Uniform [0, 1)

Then *search* for $\arg\max_x F(x) < u$

Example 1: sampling a discrete random variable

How do you sample a discrete random variable?



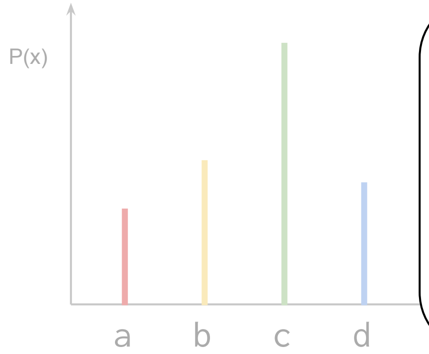
x	a	b	c	d
P(x)	0.17	0.24	0.40	0.19

In `numpy.random.choice()`:

```
cdf = p.cumsum()
cdf /= cdf[-1]
uniform_samples = self.random_sample(shape)
idx = cdf.searchsorted(uniform_samples, side='right')
# searchsorted returns a scalar
# force cast to int for LLP64
idx = np.array(idx, copy=False).astype(int, casting='unsafe')
```

Example 1: sampling a discrete random variable

How do you sample a discrete random variable?

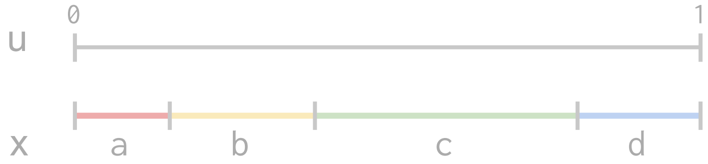


The variable u is a *latent variable*.

We have P(x) = ∫P(x, u) du
and H(x, u) > H(x).

b	c	d
0.24	0.40	0.19

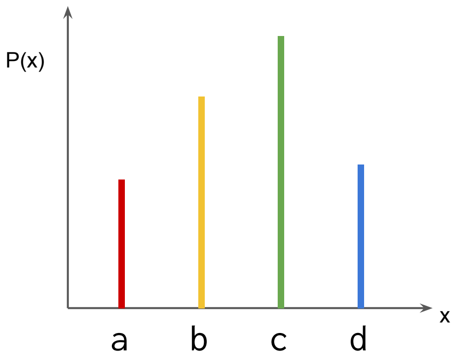
```
ce():
    cdf = p.cumsum()
    cdf /= cdf[-1]
    uniform_samples = self.random_sample(shape)
    idx = cdf.searchsorted(uniform_samples, side='right')
    # searchsorted returns a scalar
    # force cast to int for LLP64
    idx = np.array(idx, copy=False).astype(int, casting='unsafe')
```



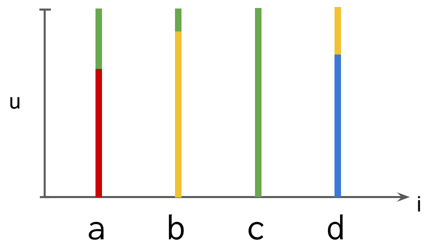
Example 2: the alias method (Walker, 1974)

Fast sampling from a categorical...

...2 table lookups + 2 samples from uniform distribution, *no search*.



x	a	b	c	d
P(x)	0.17	0.28	0.36	0.19



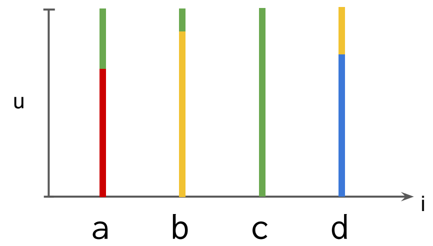
x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

Example 2: the alias method (Walker, 1974)

Fast sampling from a categorical...

...2 table lookups + 2 samples from uniform distribution, *no search*.

```
sample i ~ Uniform { a , b , c , d }
sample u ~ Uniform [0, 1)

if u < P(switch | i):
  x = alias(x)
else:
  x = i
```



x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

Example 2: the alias method (Walker, 1974)

Fast sampling from a categorical...

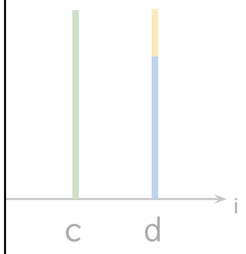
...2 table lookups + 2 samples from uniform distribution, *no search*

What's the latent variable this time?

It's the pair (u, i):

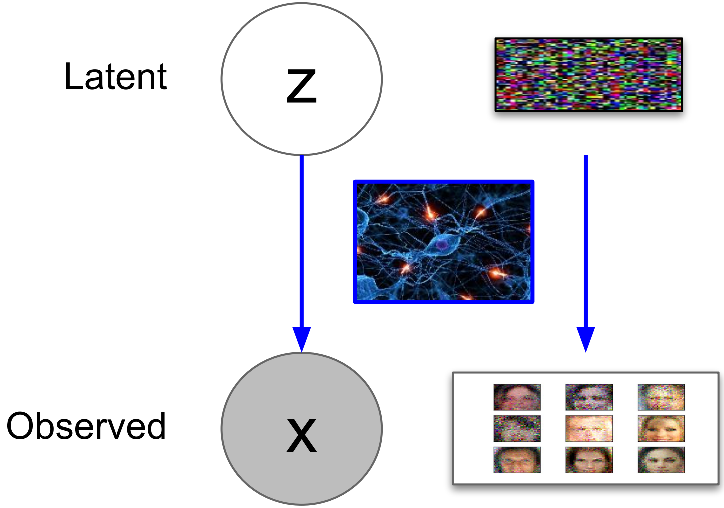
$$P(x) = \int P(x, u, i)\, du\, di$$
$$\text{and } H(x, u, i) > H(x)$$

```
sample i ~ Uniform {
sample u ~ Uniform [0

if u < P(switch | i):
  x = alias(x)
else:
  x = i
```

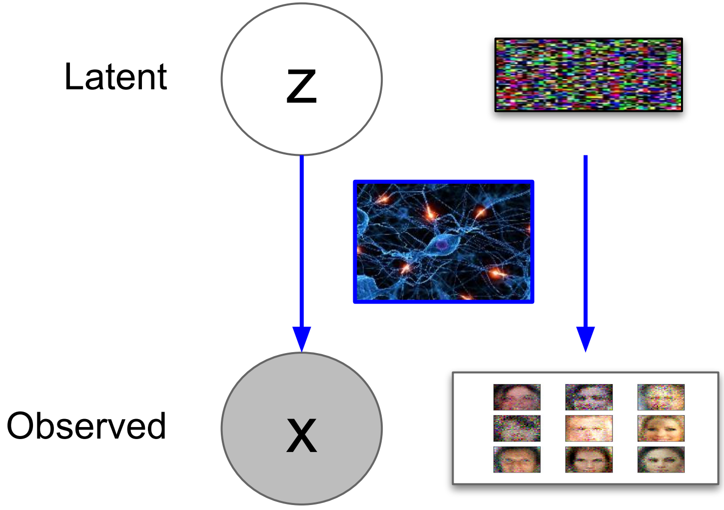


x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

Example 3: latent variable *models* (VAEs)



Example 3: latent variable *models* (VAEs)



Joint distribution p(x, z) = p(x | z)p(z) factors:

Prior over latent z

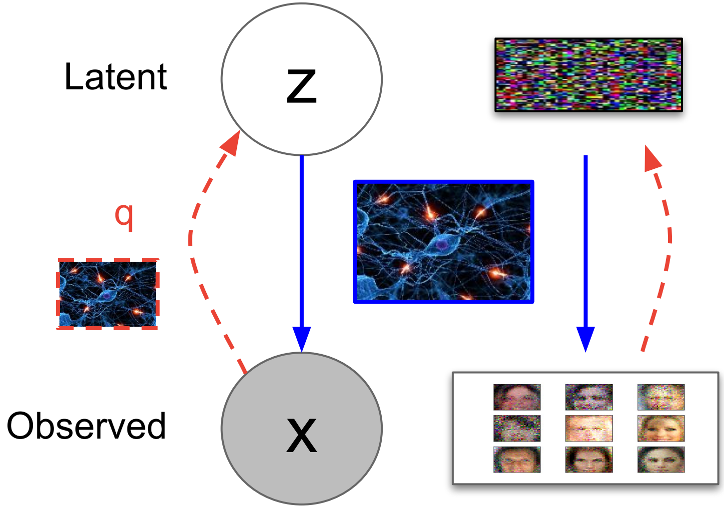
$$p(z) = N(z; 0, I)$$

Conditional over x

$$p(x \mid z; \theta) = m(x; \mu(z; \theta))$$

- μ is a **neural network,** θ parameters

Example 3: latent variable *models* (VAEs)



Optimize ***lower bound*** on marginal

$$\log p(x) \geq L(\theta, \varphi) \triangleq E_{q(z;\, x)}[\log p_\theta(x, z) - \log q_\varphi(z; x)]$$

$$q_\varphi(z; x) = N(z; \mu_q(x; \varphi), \Sigma_q(x; \varphi))$$

- q **approximates the posterior**

$$q(z; x) \approx p(z \mid x)$$

- $\Sigma_q(x; \varphi)$ usually **diagonal** (‘mean field’)

Talk outline

1. Latent variable *methods*
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

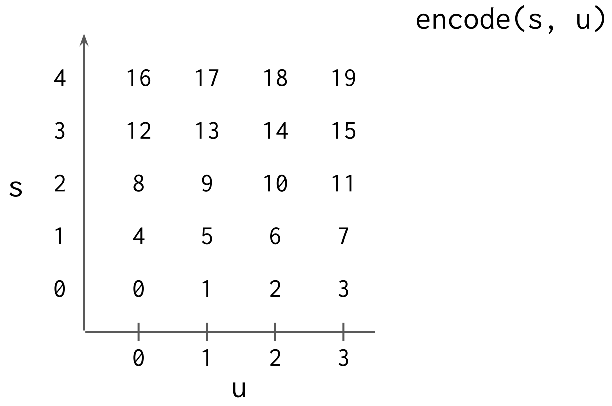
Talk outline

1. Latent variable *methods*
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

Compressing uniform data

A relatively simple ‘arithmetic coding’ method:

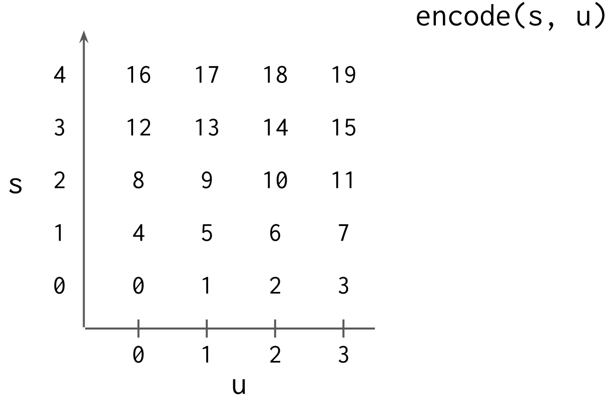
```
# Assuming s in {0, 1, ...} and u in {0, 1, ..., N - 1}
encode(s, u) := N * s + u

# Reverse operation:
decode(s’) := (s’ // N, s’ % N)
```



Compressing uniform data

A relatively simple ‘arithmetic coding’ method:

```
# Assuming s in {0, 1, ...} and u in {0, 1, ..., N - 1}
encode(s, u) := N * s + u

# Reverse operation:
decode(s’) := (s’ // N, s’ % N)
```



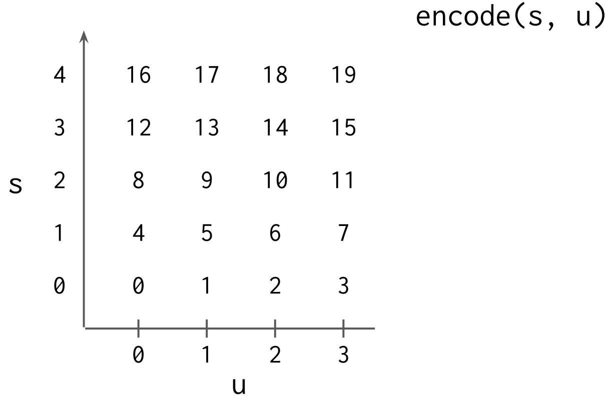
Can compress/decompress a list:

```
s = 0
for u in reversed(us):
  s = encode(s, u)

# to undo the above:
us = []
for i in range(length):
  s, u = decode(s)
  us.append(u)
```

Compressing uniform data

A relatively simple ‘arithmetic coding’ method:

```
# Assuming s in {0, 1, ...} and u in {0, 1, ..., N - 1}
encode(s, u) := N * s + u

# Reverse operation:
decode(s’) := (s’ // N, s’ % N)
```



Genius work by Duda (2009) showed that

- You can address the issue of `s` growing
- `size(encode(s, u)) ≈ size(s) + log2(N)`. Great if u is really uniform distributed, because then H(u) = log2(N).

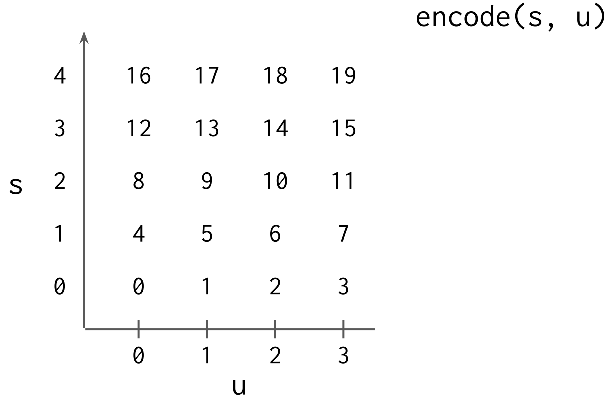
Intuitively: `log2(encode(s, u)) = log2(N * s + u) ≈ log2(s) + log2(N)`

Jarek Duda

Compressing uniform data

A relatively simple ‘arithmetic coding’ method:

```
# Assuming s in {0, 1, ...} and u in {0, 1, ..., N - 1}
encode(s, u) := N * s + u

# Reverse operation:
decode(s’) := (s’ // N, s’ % N)
```



Genius work by Duda (2009) showed that

- You can address the issue of `s` growing
- `size(encode(s, u)) ≈ s`
  really uniform distributed

This guy has done two PhDs!

Jarek Duda

Intuitively: `log2(encode(s, u)) = log2(N * s + u) ≈ log2(s) + log2(N)`

Talk outline

1. Latent variable *methods*
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

Talk outline

1. Latent variable *methods*
2. Lossless compression and asymmetric numeral systems
3. Combining 1 and 2

Compressing with latent variables ('bits-back' coding)

'Operationalize' the identity $H(X) = H(X, Z) - H(Z \mid X)$.

Compressing with latent variables (‘bits-back’ coding)

‘Operationalize’ the identity $H(X) = H(X, Z) - H(Z \mid X)$.

If we have

- an encoder/decoder for (X, Z), $\Delta\text{size}(s) = H(X, Z)$
- an encoder/decoder for $Z \mid X$, $\Delta\text{size}(s) = H(Z \mid X)$

Then we can build an encoder/decoder for x with $\Delta\text{size}(s) = H(X)$:

Compressing with latent variables (‘bits-back’ coding)

‘Operationalize’ the identity H(X) = H(X, Z) - H(Z | X).

If we have

- an encoder/decoder for (X, Z), Δsize(s) = H(X, Z)
- an encoder/decoder for Z | X, Δsize(s) = H(Z | X)

Then we can build an encoder/decoder for x with Δsize(s) = H(X):

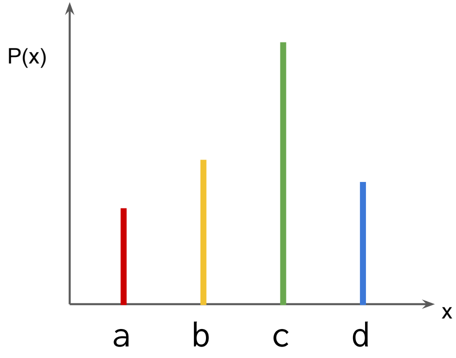
```
def encode_x(s, x):
  s, z = decode_zgx(x)(s)       # - log (1 / P(z | x))
  s    = encode_xz(s, (x, z))  # + log (1 / P(z, x))
  return s

def decode_x(s):
  s, (x, z) = decode_xz(s)
  s         = encode_zgx(x)(s, z)
  return s
```

= log 1 / P(x)

‘BB-ANS’
Townsend et al. (2019)

Example 1: sampling from a discrete random variable

How do you sample a discrete random variable?



x	a	b	c	d
P(x)	0.17	0.24	0.40	0.19

Answer: compute

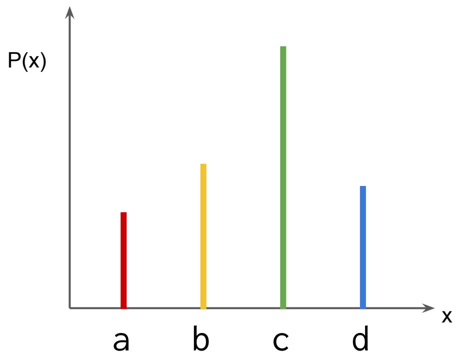
F(x)	0.00	0.17	0.41	0.81

Sample u ~ Uniform [0, 1)

Then *search* for $\arg\max_x F(x) < u$

Example 1: ~~sampling from~~ a discrete random variable

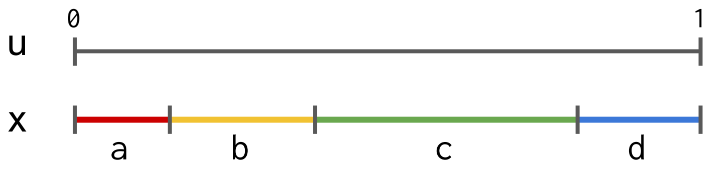
How do you ~~sample~~ a discrete random variable?



x	a	b	c	d
P(x)	0.17	0.24	0.40	0.19

Answer: compute

F(x)	0.00	0.17	0.41	0.81

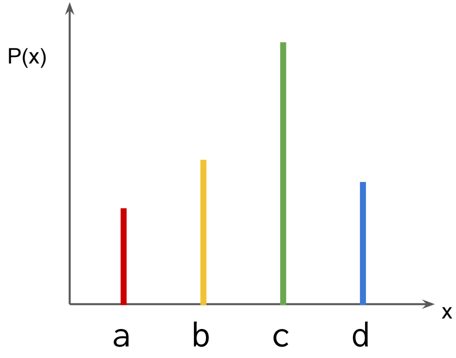


Sample u ~ Uniform [0, 1)

Then *search* for $\arg\max_x F(x) < u$

Example 1: compressing a discrete random variable

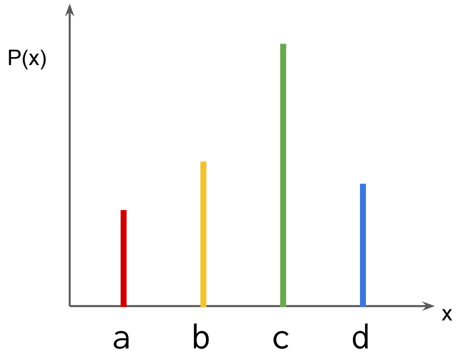
How do you compress a discrete random variable?



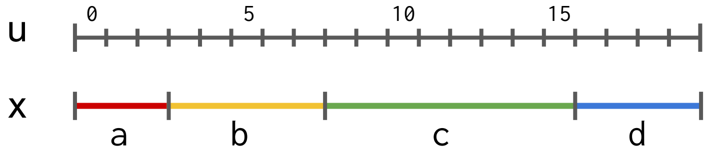
x	a	b	c	d
P(x)	0.17	0.24	0.40	0.19

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?

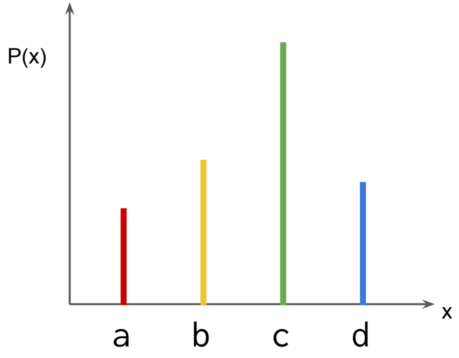


x	a	b	c	d
P(x)	3/20	5/20	8/20	4/20

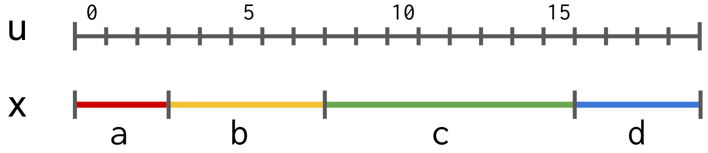


Example 1: compressing a discrete random variable

How do you compress a discrete random variable?

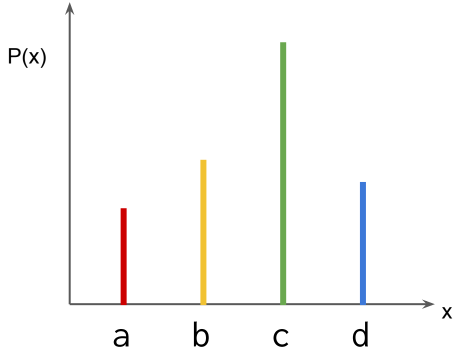


x	a	b	c	d
P(x)	3/20	5/20	8/20	4/20
M(x)	3	5	8	4

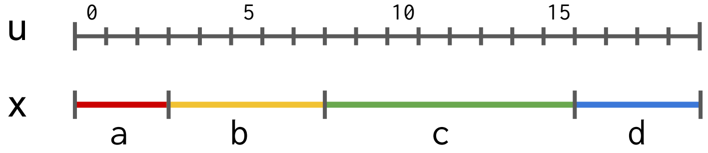


Example 1: compressing a discrete random variable

How do you compress a discrete random variable?

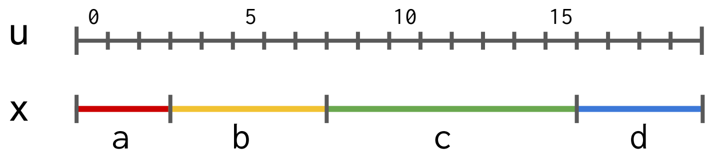


x	a	b	c	d
M(x)	3	5	8	4



Example 1: compressing a discrete random variable

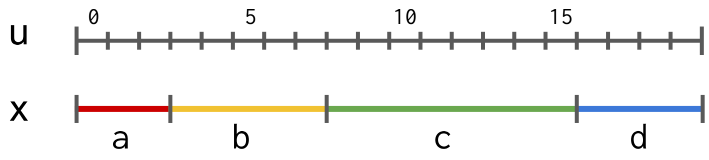
How do you compress a discrete random variable?



x	a	b	c	d
M(x)	3	5	8	4

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?

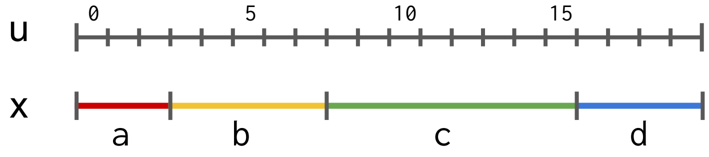


x	a	b	c	d
M(x)	3	5	8	4

```
def encode_x(s, x):
  s, u = decode_ugx(x)(s)
  s    = encode_xu(s, (x, u))
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?



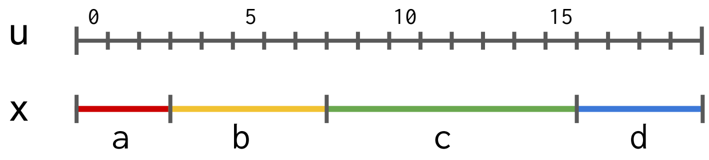
x	a	b	c	d
M(x)	3	5	8	4

```
def encode_x(s, x):
  s, u = decode_ugx(x)(s)
  s    = encode_xu(s, (x, u))
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

```
def encode_xu(s, (x , u)):
  # x deterministic given u, so only need to encode
  # u...
  return encode_uniform(20)(s, u)
```

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?



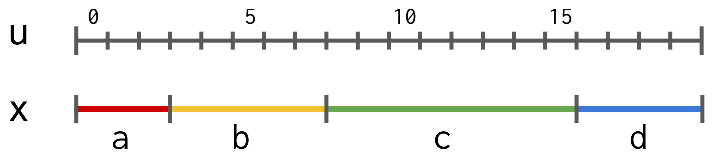
x	a	b	c	d
M(x)	3	5	8	4

```
def encode_x(s, x):
  s, u = decode_ugx(x)(s)
  s    = encode_xu(s, (x, u))
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

```
def encode_xu(s, (x , u)):
  # x deterministic given u, so only need to encode
  # u...
  return encode_uniform(20)(s, u)

def decode_xu(s):
  # First decode u
  s, u = decode_uniform(20)(s)
  # Then search
  x = cdf_lookup(u)
  return s, (x, u)
```

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?

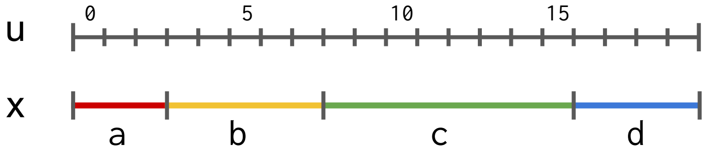


x	a	b	c	d
M(x)	3	5	8	4

```
def encode_x(s, x):
  s, u = decode_ugx(x)(s)
  s    = encode_xu(s, (x, u))
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?

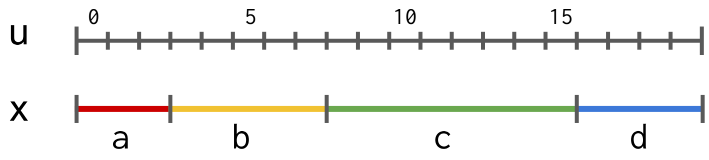


x	a	b	c	d
M(x)	3	5	8	4

```
def encode_x(s, x):
  s, u = decode_ugx(x)(s)
  s    = encode_xu(s, (x, u))
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?



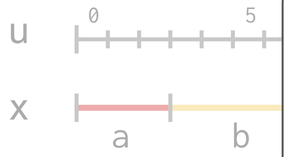
x	a	b	c	d
M(x)	3	5	8	4
F(x)	0	3	8	16

```
def encode_x(s, x):
  s, u = decode_ugx(x)(s)
  s    = encode_xu(s, (x, u))
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

```
def encode_ugx(x):
  def enc(s, u):
    # u ~ Uniform {F(x), F(x)+1, ..., F(x) + M(x)}
    return encode_uniform(M(x))(u - F(x))
  return enc

def decode_ugx(x):
  def dec(s):
    s, i = decode_uniform(M(x))(s)
    return s, i + F(x)
  return dec
```

Example 1: compressing a discrete random variable

How do you compress a discrete random variable?



c	d
8	4
8	16

This is the range variant of asymmetric numeral systems (rANS)!

```
def encode_x(s, x):
  s, u = decode_ugx(
  s    = encode_xu(s
  return s

def decode_x(s):
  s, (x, u) = decode_xu(s)
  s         = encode_ugx(x)(s, u)
  return s
```

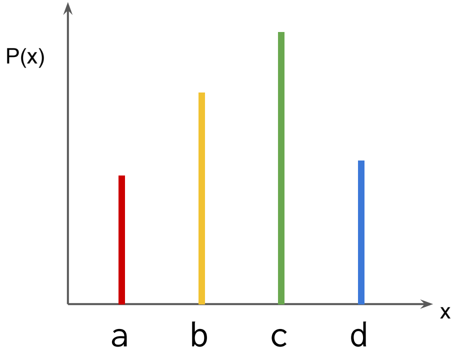
```
, F(x) + M(x)}
F(x))
```

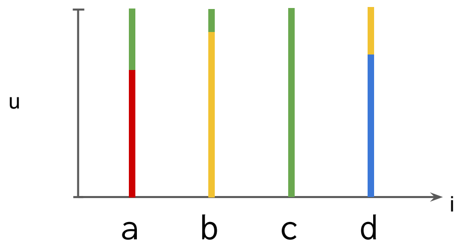
```
  def dec(s):
    s, i = decode_uniform(M(x))(s)
    return s, i + F(x)
  return dec
```

Example 2: the alias method (Walker, 1974)

Fast sampling from a categorical...

...2 table lookups + 2 samples from uniform distribution, *no search*.





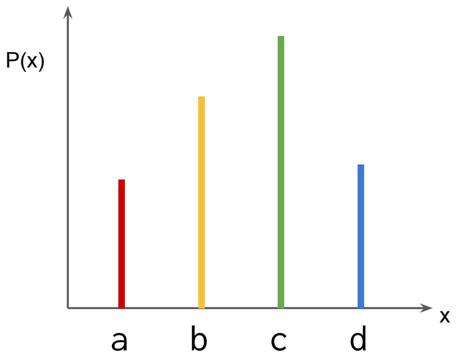
x	a	b	c	d
P(x)	0.17	0.28	0.36	0.19

x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

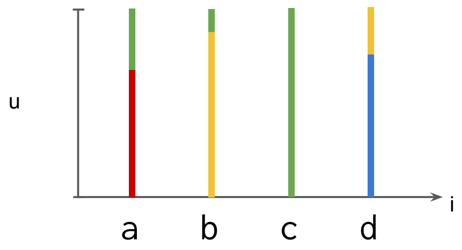
Example 2: the alias method ~~(Walker, 1974)~~

Fast ~~sampling~~ from a categorical...

...2 table lookups + 2 ~~samples~~ from uniform distribution, *no search*.



x	a	b	c	d
P(x)	0.17	0.28	0.36	0.19

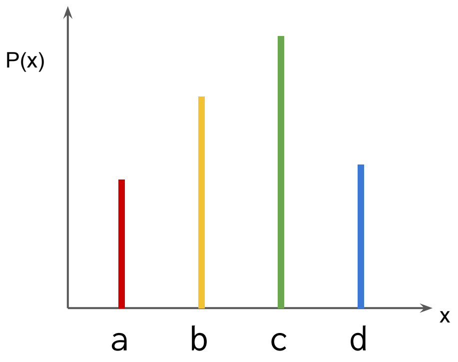


x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

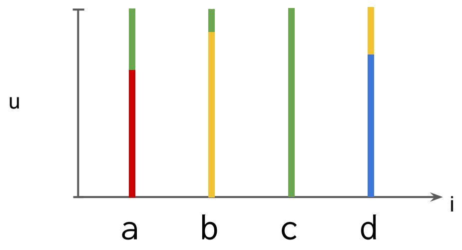
Example 2: the alias method (Giesen, 2014)

Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.



x	a	b	c	d
P(x)	0.17	0.28	0.36	0.19

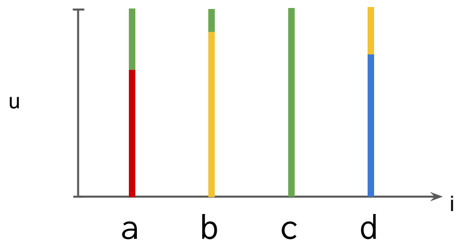


x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

Example 2: the alias method (Giesen, 2014)

Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.

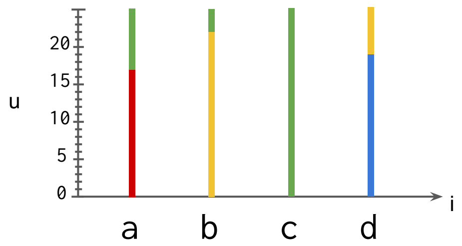


x	a	b	c	d
alias(x)	c	c	-	b
P(switch | i)	0.32	0.12	0.00	0.24

Example 2: the alias method (Giesen, 2014)

Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.



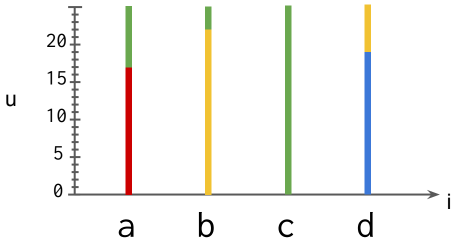
x	a	b	c	d
alias(x)	c	c	-	b
M(switch | i)	8	3	0	6

Example 2: the alias method (Giesen, 2014)

Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.

```
def encode_x(s, x):
  s, (u, i) = decode_uigx(x)(s)
  s         = encode_xui(s, (x, u, i))
  return s

def decode_x(s):
  s, (x, u, i) = decode_xui(s)
  s            = encode_uigx(x)(s, (u, i))
  return s
```



x	a	b	c	d
alias(x)	c	c	-	b
M(switch | i)	8	3	0	6

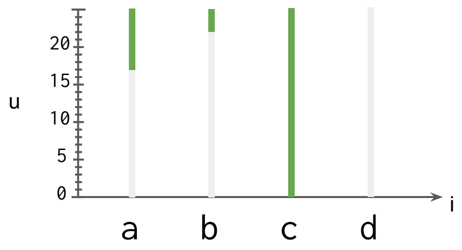
Example 2: the alias method (Giesen, 2014)

Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.

```
def encode_x(s, x):
  s, (u, i) = decode_uigx(x)(s)
  s         = encode_xui(s, (x, u, i))
  return s

def decode_x(s):
  s, (x, u, i) = decode_xui(s)
  s            = encode_uigx(x)(s, (u, i))
  return s
```

(u, i) | x = c

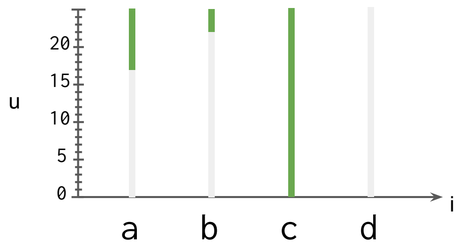


Example 2: the alias method (Giesen, 2014)

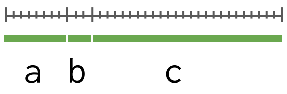
Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.

(u, i) | x = c




```
def encode_x(s, x):
  s, (u, i) = decode_uigx(x)(s)
  s         = encode_xui(s, (x, u, i))
  return s

def decode_x(s):
  s, (x, u, i) = decode_xui(s)
  s            = encode_uigx(x)(s, (u, i))
  return s
```



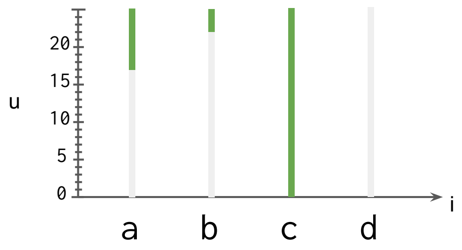
Example 2: the alias method (Giesen, 2014)

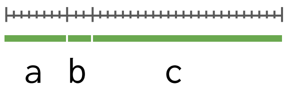
Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search*.

```
def encode_x(s, x):
  s, (u, i) = decode_uigx(x)(s)
  s         = encode_xui(s, (x, u, i))
  return s

def decode_x(s):
  s, (x, u, i) = decode_xui(s)
  s            = encode_uigx(x)(s, (u, i))
  return s
```

(u, i) | x = c





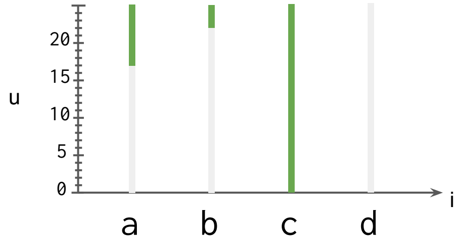
Decoding (u, i) | x requires search!

Example 2: the alias method (Giesen, 2014)

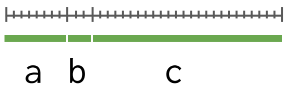
Fast decoding from a categorical...

...2 table lookups + 2 decodes from uniform distribution, *no search**.

(u, i) | x = c




```
def encode_x(s, x):
  s, (u, i) = decode_uigx(x)(s)
  s         = encode_xui(s, (x, u, i))
  return s

def decode_x(s):
  s, (x, u, i) = decode_xui(s)
  s            = encode_uigx(x)(s, (u, i))
  return s
```



Decoding (u, i) | x requires search!

*during decoding.

Example 2: the alias method (Giesen, 2014)

Fast decoding from a categorical...

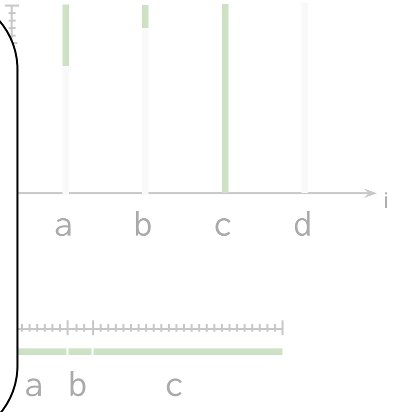
...2 table lookups + 2 decodes from uniform distribution, *no search**.

```
def encode_x(s, x):
  s, (u, i) = decode_uigx(x)
  s         = encode_xui(s,
  return s

def decode_x(s):
  s, (x, u, i) = decode_xui(
  s            = encode_uigx
  return s
```

Key point: alias method *moves* work from the decoder to the encoder...

total complexity of encoding + decoding stays the same
🤔

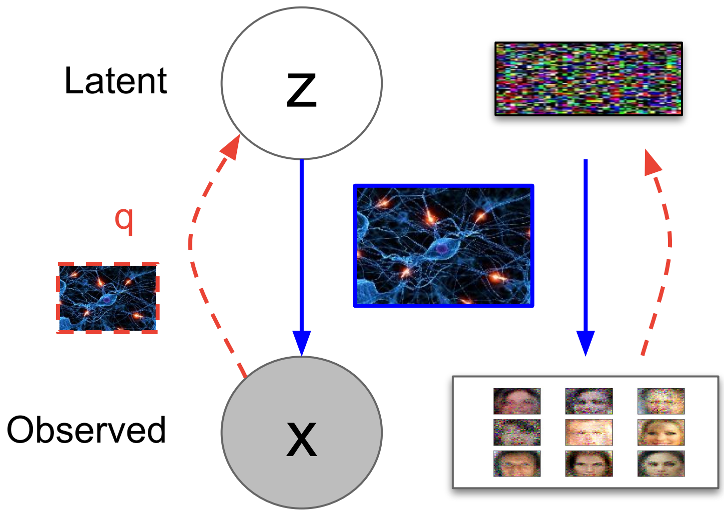
(u, i) | x = c



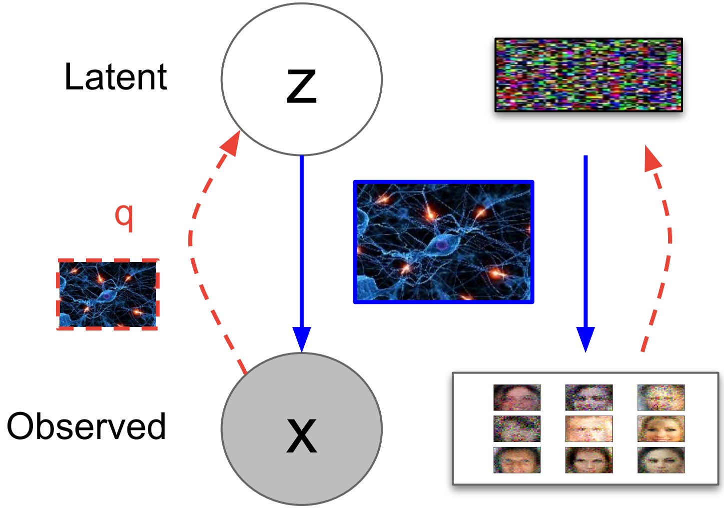
Decoding (u, i) | x requires search!

*during decoding.

Example 3: latent variable *models*



Example 3: latent variable *models*

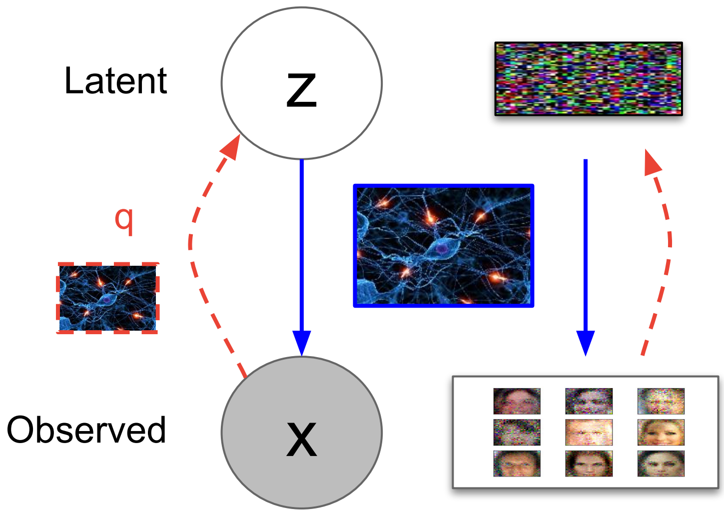


Train a VAE model on images...

...directly apply latent variable compression.

Result: a good (formerly SOTA) lossless image compression rate, with fast-ish decoding.

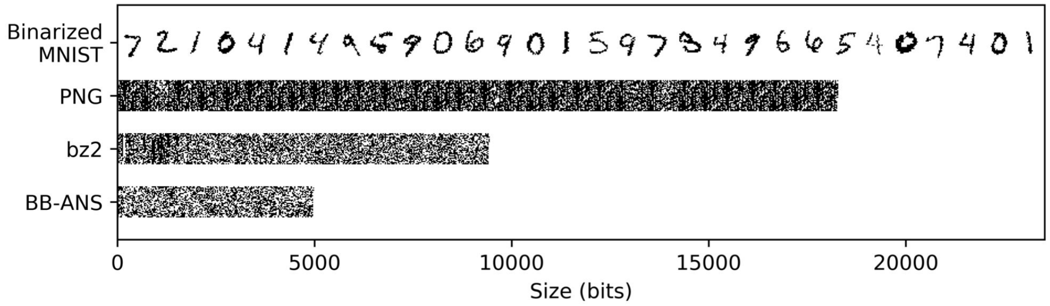
Example 3: latent variable *models*



Train a VAE model on images...

...directly apply latent variable compression.

Result: a good (formerly SOTA) lossless image compression rate, with fast-ish decoding.



Townsend et al. (2019)

More!

- Optimal compression of *multisets* (Severo et al., 2022)
- Optimal compression of *unlabelled random graphs* (unpublished)
- Compression with *latent state space models* (Townsend and Murray, 2021)
- Compression with *hierarchical LVMs* (Townsend et al., 2020; F. Kingma et al. 2020)
- Compression with *diffusion models* (D. Kingma et al., 2021)

More!

- Optimal compression of *multisets* (Severo et al., 2022)
- Optimal compression of *unlabelled random graphs* (unpublished)
- Compression with *latent state space models* (Townsend and Murray, 2021)
- Compression with *hierarchical LVMs* (Townsend et al., 2020; F. Kingma et al. 2020)
- Compression with *diffusion models* (D. Kingma et al., 2021)

All use the pattern discussed in this talk. There are probably more examples, still to be discovered.

The end. Thanks for listening.

James Townsend

Stanford IT Forum
28/10/2022

References

1. Walker, A. J. (April 1974). "New fast method for generating discrete random numbers with arbitrary frequency distributions". *Electronics Letters*. **10** (8): 127.
2. Duda, J. (2009). “Asymmetric numeral systems”. https://arxiv.org/abs/0902.0271.
3. Townsend, James, Thomas Bird, and David Barber. ‘Practical Lossless Compression with Latent Variables Using Bits Back Coding’, 2019. https://openreview.net/forum?id=ryE98iR5tm.
4. Daniel Severo*, James Townsend*, Ashish Khisti, Alireza Makhzani, and Karen Ullrich, Compressing Multisets with Large Alphabets, appearing at the Data Compression Conference (DCC), 2022.
5. James Townsend and Iain Murray, Lossless Compression with State Space Models Using Bits Back Coding, Neural Compression: From Information Theory to Applications -- Workshop @ ICLR 2021.
6. James Townsend*, Thomas Bird*, Julius Kunze, and David Barber, HiLLoC: Lossless Image Compression with Hierarchical Latent Variable Models, International Conference on Learning Representations (ICLR), 2020. *Equal contribution.
7. Friso Kingma, Pieter Abbeel, Jonathan Ho. Bit-Swap: Recursive Bits-Back Coding for Lossless Compression with Hierarchical Latent Variables. *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97:3408-3417, 2019.
8. Kingma, Diederik P., Tim Salimans, Ben Poole, and Jonathan Ho. ‘Variational Diffusion Models’, 2022. https://openreview.net/forum?id=2LdBqxc1Yv.