TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Entropy and Compressibility

Rate-Distortion Autoencoders (RDAs)

Noisy Channel RDAs

Gaussian Variational Autoencoders (Gaussian VAEs)

Issue I: Measuring Cross Entropy

We typically cannot measure cross-entropy for a graphical model.

But we can measure cross-entropy of a compression model.

We write $\tilde{z}(y)$ for a lossless compression of y and write $|\tilde{z}_\Phi(y)|$ for the bit length of the compression.

$$\Phi^* = \operatorname*{argmin}_\Phi \; E_{y\sim\mathrm{Pop}} \; -\ln P_\Phi(y)$$

$$\Phi^* = \operatorname*{argmin}_\Phi \; E_{y\sim\mathrm{Pop}} \, |\tilde{z}_\Phi(y)| \quad \text{such that } \forall y \quad y = \tilde{y}_\Phi(\tilde{z}_\Phi(y))$$

Issue II: Avoiding Differential Entropy

For reasons discussed below, we would like to avoid differential entropy.

To avoid differential entropy we use a rate-distortion objective.

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{y\sim\mathrm{Pop}} \; -\ln P_\Phi(y) \quad y \text{ discrete}$$

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{y\sim\mathrm{Pop}} \; -\ln P_\Phi(\tilde{y}_\Phi(y)) + \lambda \mathrm{Dist}(y, \tilde{y}_\Phi(y)) \left\{ \begin{array}{l} y \text{ continuous} \\ \tilde{y} \text{ discrete} \end{array} \right.$$

Lossy Compression

Lossy compression combines compression for measuring cross-entropy with distortion for avoiding differential entropy.

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{y \sim \mathrm{Pop}} \; - \ln P_\Phi(y)$$

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{y \sim \mathrm{Pop}} \; |\tilde{z}_\Phi(y)| + \lambda \mathrm{Dist}(y, \tilde{y}_\Phi(\tilde{z}_\Phi(y)))$$

Entropy and Compression

For a distribution $P(y)$ on a discrete set $\mathcal{Y}$, the entropy $H(P)$, when measured using $\log_2$ rather than ln, gives the number of bits needed on average, when drawing from P, to represent the elements of $\mathcal{Y}$.

The cross-entropy $H(P,Q)$ give the number of bits used to code for items drawn from P but using the code defined by Q.

Cross-entropy gives the "data rate" when transmitting codes for items drawn from P but using the code defined by Q.

Entropy and Compressibility

Let S be a finite set.

Let z be a compression (or coding) function assigning a bit string $z(y)$ to each $y \in S$.

The compression function z is called *prefix-free* if for $y' \neq y$ we have that $z(y')$ is not a prefix of $z(y)$.

Prefix-Free Codes as Probabilities

A prefix-free code defines a binary branching tree — branch on the first code bit, then the second, and so on.

For a prefix-free code, only the leaves of this tree can be labeled with the elements of S.

The code defines a probability distribution on S by randomly selecting branches.

We have $P_z(y) = 2^{-|z(y)|}$.

Bits vs. Nats

We have that $|z(y)|$ is a number of bits.

We can define entropy in units of bits by

$$H_2(y) = E_y \; -\log_2 P(y) = H(y)/(\ln\ 2)$$

If y is uniformly distributed over 8 values then $H_2(y)$ is 3 bits.

We have that $H_2(y)$ is a number of bits while $H(y)$ is a number of "nats".

The Source Coding (compression) Theorem

(1) There exists a prefix-free code z such that

$$|z(y)| <= (-\log_2 \mathrm{Pop}(y)) + 1$$

and hence

$$E_{y\sim\mathrm{Pop}}|z(y)| \leq H_2(\mathrm{Pop}) + 1$$

(2) For any prefix-free code z

$$E_{y\sim\mathrm{Pop}}\,|z(y)| \geq H_2(\mathrm{Pop})$$

Code Construction

We construct a code by iterating over $y \in S$ in order of decreasing probability (most likely first).

For each y select a code word $z(y)$ (a tree leaf) with length (depth)

$$|z(y)| = \lceil -\log_2 \mathrm{Pop}(y) \rceil$$

and where $z(y)$ is not an extension of (under) any previously selected code word.

Code Existence Proof

At any point before coding all elements of S we have

$$\sum_{y \in \text{Defined}} 2^{-|z(y)|} \leq \sum_{y \in \text{Defined}} \text{Pop}(y) < 1$$

Therefore there exists an infinite descent into the tree that misses all previous code words.

Hence there exists a code word $z(x)$ not under any previous code word with $|z(x)| = \lceil -\log_2 \text{Pop}(y) \rceil$.

Furthermore $z(x)$ is at least as long as all previous code words and hence $z(x)$ is not a prefix of any previously selected code word.

No Better Code Exists

Let z be an arbirtary coding.

$$
\begin{aligned}
E_y\ |z(y)| &= E_y\ -\log_2 P_z(y) \\
&= H_2(\mathrm{Pop}, P_z) \\
&= H_2(\mathrm{Pop}) + KL_2(\mathrm{Pop}, P_z) \\
&\geq H_2(\mathrm{Pop})
\end{aligned}
$$

Huffman Coding

Maintain a list of trees $T_1, \ldots, T_N$.

Inititally each tree is just one root node labeled with an element of S.

Each tree T_i has a weight equal to the sum of the probabilities of the nodes on the leaves of that tree.

Repeatedly merge the two trees of lowest weight into a single tree until all trees are merged.

Optimality of Huffman Coding

Theorem: The Huffman code T for Pop is optimal — for any other tree T' we have $H(\mathrm{Pop}, T) \leq H(\mathrm{Pop}, T')$.

Proof: The algorithm maintains the invariant that there exists an optimal tree including all the subtrees on the list.

To prove that a merge operation maintains this invariant we consider any tree containing the given subtrees.

Consider the two subtrees T_i and T_j of minimal weight. Without loss of generality we can assume that T_i is at least as deep as T_j.

Lowering T_j to be the sibling of T_i while raising the old sibling of T_i to T_j's original position brings T_i and T_j together and can only improve the average depth.

Differential Entropy

Consider a continuous density $p(x)$. For example

$$p(x) = \frac{1}{\sqrt{2\pi}\ \sigma}\ e^{\frac{-x^2}{2\sigma^2}}$$

Differential entropy is often defined as

$$H(p) \doteq \int \left(\ln \frac{1}{p(x)} \right) p(x) dx$$

Differential Entropy Depends on the Choice of Units

$$
\begin{aligned}
H(\mathcal{N}(0,\sigma)) &= +\int \left(\ln(\sqrt{2\pi}\sigma) + \frac{x^2}{2\sigma^2}\right) p(x)dx \\
&= \ln(\sigma) + \ln(\sqrt{2\pi}) + \frac{1}{2}
\end{aligned}
$$

But if we take $y \doteq x/2$ we get $H(y) = H(x) - \ln 2$.

Also for $\sigma << 1$, we get $H(p) < 0$

Hence differential entropy then depends on the choice of units — a distributions on lengths will have a different entropy when measuring in inches than when measuring in feet.

More Problems with Differential Entropy

There are also other problems with continuous entropy and cross-entropy.

- Finite continuous entropy violates the source coding theorem — it takes an infinite number of bits to code a real number.

- Finite continuous entropy violates the data processing inequality that $H(f(x)) \leq H(x)$. For a continuous random variable x under finite continuous entropy we can have $H(f(x)) > H(x)$.

For these reasons it seems advisable to avoide differential entropy and differential cross entropy.

Differential KL-divergence is Independent of Units

$$KL(p,q) = \int \left(\ln \frac{p(x)}{q(x)}\right) p(x)dx$$

This integral can be computed by dividing the real numbers into bins and computing the KL divergence between the distributions on bins.

The KL divergence between the bin distribution often approaches a finite limit as the bin size goes to zero.

Rate-Distortion Autoencoders

Given a continuous signal y we can compress it into a (discrete) bit string $\tilde{z}_\Phi(y)$.

We let $\tilde{y}_\Phi(\tilde{z}_\Phi(y))$ be the decompression of $\tilde{z}_\Phi(y)$.

We can then define a rate-distortion loss.

$$\mathcal{L}(\Phi) = E_{y \sim \mathrm{Pop}}\, |\tilde{z}_\Phi(y)| + \lambda \mathrm{Dist}(y, \tilde{y}_\Phi(\tilde{z}_\Phi(y)))$$

The Rate-Distortion Tradeoff

$$\mathcal{L}(\Phi) = E_{y \sim \mathrm{Pop}}\, |\tilde{z}_\Phi(y)| + \lambda \mathrm{Dist}(y, \tilde{y}(\tilde{z}(y)))$$

The first term is as a rate measured in in bits per sample.

The meta-parameter λ has units of inverse distortion and controls the trade off between rate and distortion.

Common Distortion Functions

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \; E_{y\sim\mathrm{Pop}}\,|\tilde{z}_\Phi(y)| + \lambda \mathrm{Dist}(y, \tilde{y}_\Phi(y))$$

It is common to take

$$\mathrm{Dist}(y, \tilde{y}) = ||y - \tilde{y}||^2 \qquad (L_2)$$

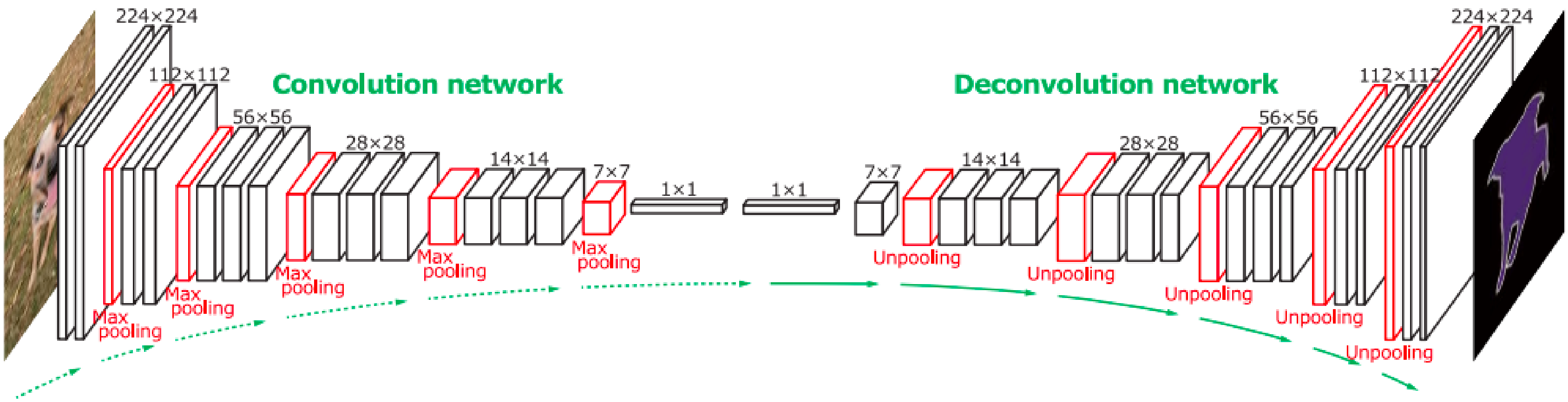
or

$$\mathrm{Dist}(y, \tilde{y}) = ||y - \tilde{y}||_1 \qquad (L_1)$$

A Case Study in Image Compression

End-to-End Optimized Image Compression, Balle, Laparra, Simoncelli, ICLR 2017.

y $\quad\quad$ $\tilde{z}_{\Psi}(y)$ $\quad\quad$ $\tilde{z}$ $\quad\quad$ $\tilde{y}_{\Phi}(\tilde{z})$ $\quad\quad$ $||y - \tilde{y}||^2$



JPEG at 4283 bytes or .121 bits per pixel

JPEG, 4283 bytes (0.121 bit/px), PSNR: 24.85 dB/29.23 dB, MS-SSIM: 0.8079

JPEG 2000 at 4004 bytes or .113 bits per pixel

JPEG 2000, 4004 bytes (0.113 bit/px), PSNR: 26.61 dB/33.88 dB, MS-SSIM: 0.8860

Deep Autoencoder at 3986 bytes or .113 bits per pixel

Proposed method, 3986 bytes (0.113 bit/px), PSNR: 27.01 dB/34.16 dB, MS-SSIM: 0.9039

A CNN Encoder

A three layer CNN is used as an encoder.

We let $z_\Phi(y)$ be the final layer of this CNN.

Each continuous value in the final layer $z_\Phi(y)$ is then rounded to a (small) integer giving a discrete encoding $\tilde{z}(y)$.

Rate-Distortion Autoencoders

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{y \sim \mathrm{Pop}} \, |\tilde{z}_\Phi(y)| + \lambda \mathrm{Dist}(y, y_\Phi(\tilde{z}_\Phi(y)))$$

Oops: Because of rounding, $\tilde{z}_\Phi(y)$ is discrete and the gradients are zero.

Rate-Distortion Autoencoders

$$\Phi^* = \operatorname*{argmin}_{\Phi} \mathcal{L}_{\mathrm{rate}}(\Phi) + \lambda \mathcal{L}_{\mathrm{dist}}(\Phi)$$

$$\mathcal{L}_{\mathrm{rate}}(\Phi) = E_{y \sim \mathrm{Pop}} \, |\tilde{z}_\Phi(y)|$$

$$\mathcal{L}_{\mathrm{dist}}(\Phi) = E_{y \sim \mathrm{Pop}} \, \mathrm{Dist}(y, y_\Phi(\tilde{z}_\Phi(y)))$$

We will consider differentiable approximations to both $\mathcal{L}_{\mathrm{rate}}$ and $\mathcal{L}_{\mathrm{dist}}$.

A Differentiable Approximation of $\mathcal{L}_{\text{rate}}$

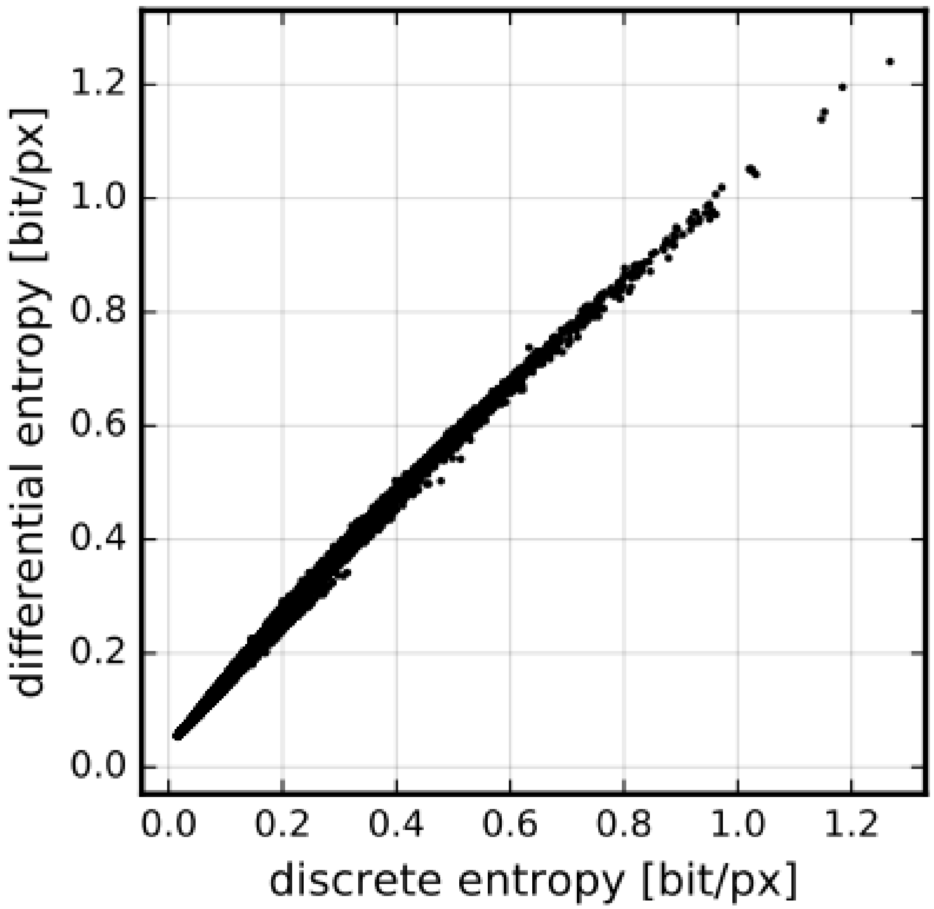
$$\mathcal{L}_{\text{rate}}(\Phi) = E_{y \sim \text{Pop}}\,|\tilde{z}_\Phi(y)|$$

Recall that $\tilde{z}_\Phi(y)$ is a rounding of a continuous encoding $z_\Phi(y)$.

We can make the cross-entropy loss differentiable by approximating discrete entropy with differential entropy.

$$|\tilde{z}_\Phi(y)| \approx -\ln p_\Psi(z_\Phi(y)) \qquad p_\Psi(z) \approx \sum_i \log_2 z_i$$

Differential Entropy vs. Discrete Entropy



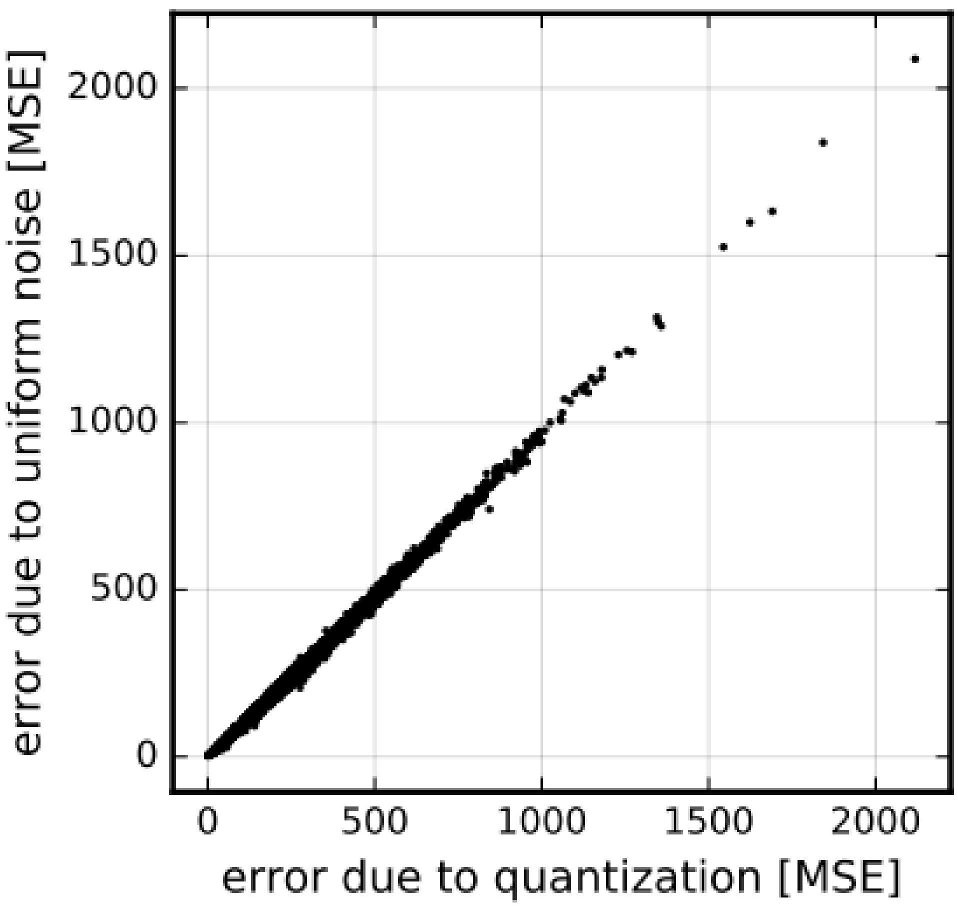
A Differentiable Approximation of $\mathcal{L}_{\mathrm{dist}}$

We can make the the distortion loss differentiable by modeling rounding as the addition of noise.

$$\begin{aligned}\mathcal{L}_{\mathrm{dist}}(\Phi) &= E_{y\sim\mathrm{Pop}}\ \mathrm{Dist}(y, y_\Phi(\tilde{z}_\Phi(y)))\\ &\approx E_{y,\epsilon}\ \mathrm{Dist}(y,\ y_\Phi(z_\Phi(y)+\epsilon))\end{aligned}$$

Here ϵ is a noise vector each component of which is drawn uniformly from $(-1/2, 1/2)$.

Noise vs. Rounding



Details

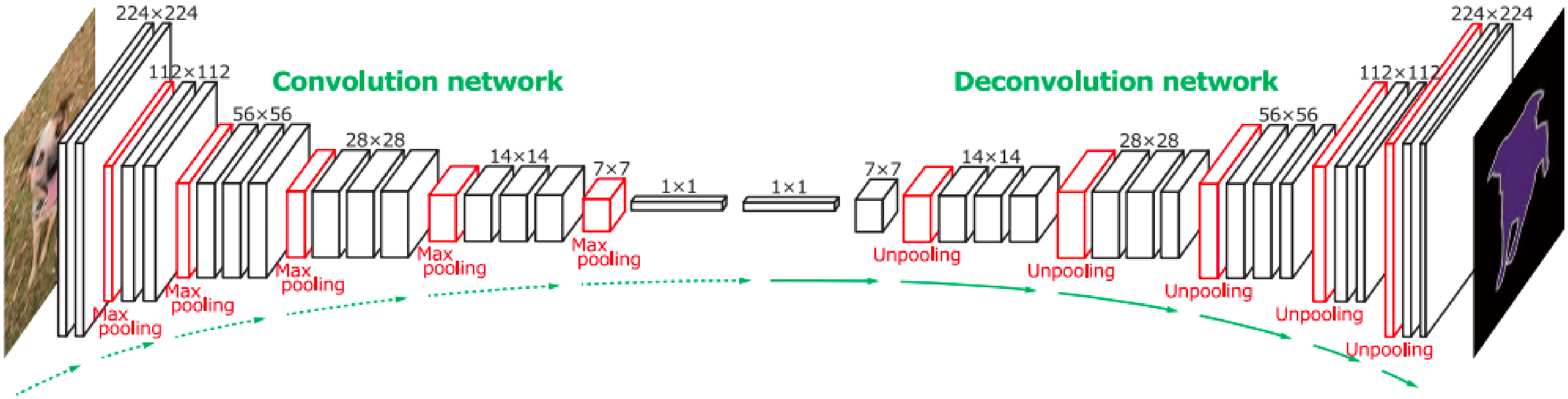
The first layer is computed stride 4.

The next two layers are computed stride 2.

Final image dimension is reduced by a factor of 16 with 192 channels per pixel (192 channels is for color images).

$$192 < 16 \times 16 \times 3 = 768$$

Increasing Spatial Dimension in Decoding



[Hyeonwoo Noh et al.]

In the ICLR 17 paper the deconvolution network has the shape as the input CNN but with independent parameters.

Increasing Spatial Dimension in Decoding

Consider a stride 2 convolution

$$L_{\ell+1}[x, y, j] = \sigma\left(\sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] L_\ell[2x + \Delta x, 2y + \Delta y, i]\right)$$

For deconvolution we use stride 1 with 4 times the features.

$$L'_\ell[x, y, i] = \sigma\left(\sum_{\Delta x, \Delta y, j} W[\Delta x, \Delta y, j, i] L'_{\ell+1}[x + \Delta x, y + \Delta y, j]\right)$$

The channels at each $L'_\ell[x, y]$ are divided among four higher resolution pixels.

This is done by a simple reshaping of $L'_\ell[x, y, i]$.

Noisy-Channel RDAs (TZ)

Consider the differentiable loss used in training.

$$\Phi^* = \underset{\Phi}{\mathrm{argmin}}\;\; E_{y\sim \mathrm{Pop}}\; -\ln p(z_\Phi(y)) + \lambda E_\epsilon \;\mathrm{Dist}(y, y_\Phi(z_\Phi(y)+\epsilon))$$

Intuitively, $-\ln p(z_\Phi(y))$ is a proxy for the number of bits used in the (intuitively rounded) encoding $z_\Phi(y) + \epsilon$.

By the channel capcacity theorem the number of bits that $z_\Phi(y) + \epsilon$ can carry about y is the mutual information between y and $z_\Phi(y) + \epsilon$.

$$I(y, z_\Phi(y) + \epsilon)$$

Noisy-Channel RDAs (TZ)

We now consider $p_\Phi(z|y)$ as a generalization of $z_\Phi(y) + \epsilon$.

The channel capacity theorem motivates

$$
\begin{aligned}
\Phi^* &= \operatorname*{argmin}_\Phi \left(\; I(y,z) + \; \lambda E_{y\sim \mathrm{Pop},z\sim p_\Phi(z|y)} \; \mathrm{Dist}(y, y_\Phi(z)) \; \right) \\
&= \operatorname*{argmin}_\Phi E_{y\sim \mathrm{Pop}} \; \left(\; KL(p_\Phi(z|y), p_\Phi(z)) + \lambda E_{z\sim p_\Phi(z|y)} \; \mathrm{Dist}(y, y_\Phi(z)) \; \right)
\end{aligned}
$$

A Variational Upper Bound

Unfortunately we cannot compute $p_\Phi(z) = E_{y\sim\mathrm{Pop}}\, p_\Phi(z|y)$.

We now replace $p_\Phi(z)$ by a friendly (variational) model $p_\Psi(z)$.

$$
\begin{aligned}
I_\Phi(y,z) &= E_{y\sim\mathrm{Pop}}\, KL(p_\Phi(z|y), p_\Phi(z)) \\
&= E_{y,z\sim P_\Phi(z|y)}\; \ln \frac{p_\Phi(z|y)}{p_\Psi(z)} + \ln \frac{p_\Psi(z)}{p_\Phi(z)} \\
&= E_y\, KL(p_\Phi(z|y), p_\Psi(z)) - KL(p_\Phi(z), p_\Psi(z)) \\
&\leq E_y\, KL(p_\Phi(z|y), p_\Psi(z))
\end{aligned}
$$

The Noisy-Channel RDA

$$\Phi^*, \Psi^* = \operatorname*{argmin}_{\Phi,\Psi} \; E_y \; KL(p_\Phi(z|y), P_\Psi(z)) \; + \; \lambda \; E_{z \sim p_\Phi(z|y)} \, \mathrm{Dist}(y, \; y_\Phi(z))$$

Here γ has the same units as distortion and controls the trade-off between rate and distortion.

Summary: Rate-Distortion

Rate-Distortion: y, continuous, $\tilde{z}$ a bit string,

$$\Phi^* = \operatorname*{argmin}_{\Phi} E_y \; |\tilde{z}_\Phi(y)| + \lambda \mathrm{Dist}(y, y_\Phi(\tilde{z}_\Phi(y)))$$

Noisy Channel: $\tilde{z} = z_\Phi(y) + \sigma_\Phi(y) \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$

$$\Phi^* = \operatorname*{argmin}_{\Phi} E_y \;\; KL(p_\Phi(\tilde{z}|y), \mathcal{N}(0, I)) + E_{\tilde{z} \sim p_\Phi(\tilde{z}|y)} \; \lambda \mathrm{Dist}(y, y_\Phi(\tilde{z}))$$

END