TTIC 31230, Fundamentals of Deep Learning

David McAllester, April 2017

Information Theory and Distribution Modeling

Why do we model distributions and conditional distributions using the following objective functions?

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \, \mathrm{E}_{x \sim D} \left[\ln \frac{1}{P_\Theta(x)} \right]$$

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \, \mathrm{E}_{(x,y) \sim D} \left[\ln \frac{1}{P_\Theta(y|x)} \right]$$

Why is "bits per word" the natural measure of the performance of a language model?

How is "bits per sample" related to actual data compression?

Shannon's Source Coding (Compression) Theorem

Consider a data distribution D such as the "natural" distribution on sentences.

Shannon's theorem states that the average compressed size (in bits) under optimal compression when drawing x from D is the entropy $H(D)$

$$H(D) = \mathrm{E}_{x \sim D}\left[\log_2 \frac{1}{D(x)}\right]$$

Note that if D is the uniform distribution on 2^N items then it takes N bits to name one of the items.

Shannon's Source Coding (Compression) Theorem

Consider a probability distribution D on a finite set $\mathcal{X}$.

We define a tree $\mathcal{T}$ over $\mathcal{X}$ to be a binary branching tree whose leaves are labeled with (all) the elements of $\mathcal{X}$.

Let $d(x; T)$ be the depth of the leaf that is labeled with x.

We can name each element with a bit string of length $d(x; T)$.

Define $d(T; D) = \mathrm{E}_{x \sim D}\left[d(x; T)\right] =$ average compressed size.

Theorem:

$$\forall T\ d(T; D) \geq H(D)$$
$$\exists T\ d(T; D) \leq H(D) + 1$$

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 $\mathcal{X}$.

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 D is optimal — for any other tree T' we have $d(T; D) \leq d(T'; D)$.

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.

Swapping the sibling of T_i for T_j brings T_i and T_j together and can only improve the average depth.

Modeling a Distribution

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \; H(D, P_\Theta)$$

$$H(D, P_\Theta) = \textbf{ cross entropy} = \mathrm{E}_{x \sim D}\left[\log_2 \frac{1}{P_\Theta(x)}\right]$$

Distribution Modeling and Data Compression

Theorem: For any P_Θ there exists a code T such that for all $x \in \mathcal{X}$

$$\log_2 \frac{1}{P_\Theta(x)} \leq d(x;T) \leq \left(\log_2 \frac{1}{P_\Theta(x)}\right) + 1$$

Optimal average compressed size is achieved by

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, H(D, P_\Theta) = \underset{\Theta}{\operatorname{argmin}}\ \mathrm{E}_{x\sim D}\left[\log_2 \frac{1}{P_\Theta(x)}\right]$$

Minimizing Cross-Entropy is **the same** as optimizing data compression is **the same** as distribution modeling.

Cross Entropy vs. Entropy

An LSTM language models allow us to calculate the probability of given sentence.

This allows us to measure $H(D, P_\Theta)$ by sampling.

While we can measure the cross-entropy $H(D, P_\Theta)$ we cannot measure the true entropy of the source $H(D)$ which, for language, presumably involves semantic truth.

But we can show

$$H(D) \leq H(D, P)$$

The cross cross entropy to the model upper bounds the true data source entropy.

KL Divergence

The KL divergence is

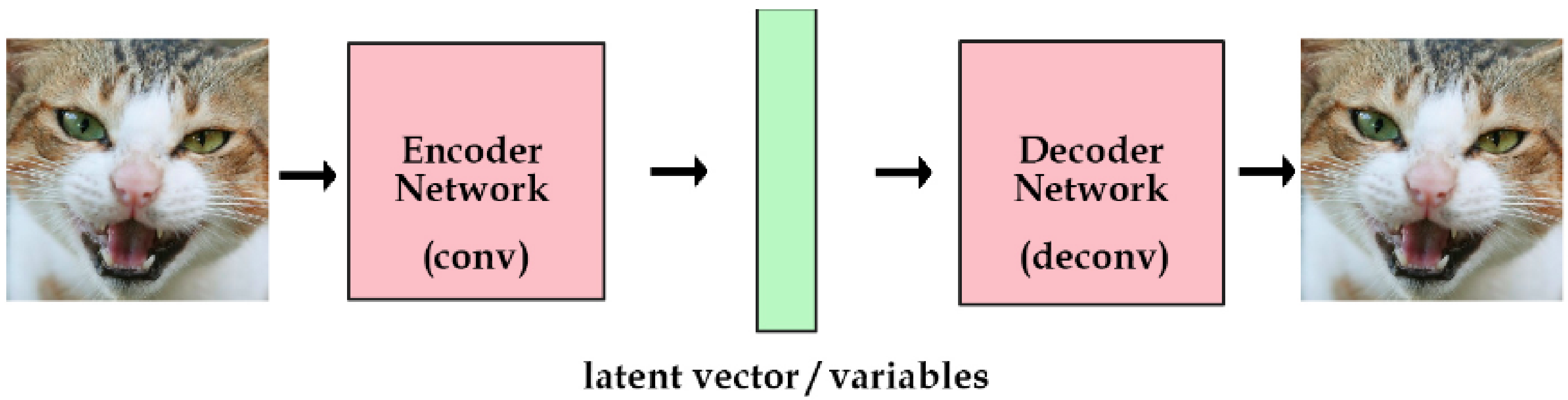
$$KL(D, P) = H(D, P) - H(D) = \mathrm{E}_{x \sim D}\left[\log_2 \frac{D(x)}{P(x)}\right]$$

We can show $KL(D, P) \geq 0$ using Jensen's inequality applied to the convexity of the negative of the log function.

KL Divergence

$$
\begin{aligned}
-KL(D,P) &= \mathrm{E}_{x\sim D}\left[\log\frac{P(x)}{D(x)}\right] \\
&\leq \log \mathrm{E}_{x\sim D}\left[\frac{P(x)}{D(x)}\right] \\
&= \log\sum_x D(x)\frac{P(x)}{D(x)} \\
&= \log\sum_x P(x) = 0 \\
KL(D,P) &\geq 0
\end{aligned}
$$

Rate-Distortion Autoencoders



[Kevin Frans]

Rate-Distortion Autoencoders

Rate-distortion theory addresses lossy compression. We assume

- An encoder (compression) network $z_\Phi(x)$ where $z_\Phi(x)$ is a bit string in a prefix-free code (a code corresponding to the leaves of a binary tree). We write $|z|$ for the number of bits in the string z.
- A decoder (decompression) network $\hat{x}_\Psi(z)$
- A distortion function $L(x, \hat{x})$

$$\Phi^*, \Psi^* = \underset{\Phi,\Psi}{\operatorname{argmin}}\; \mathrm{E}_{x\sim D}\left[\, |z_\Phi(x)| + \lambda L(x, \hat{x}_\Psi(z_\Phi(x)))\,\right]$$

Summary of Distribution Modeling

Distribution modeling is important when the distribution being modeled ($D(x)$ or $D(y|x)$) is highly distributed and precise prediction is impossible.

Mathematically, distribution modeling (minimizing cross entropy) is the same as optimizing data compression.

Summary of Distribution Modeling

$$\Theta^* = \mathrm{argmin}_\Theta \, H(D, P_\Theta)$$

Conditional version:

$$\Theta^* = \mathrm{argmin}_\Theta \, \mathrm{E}_{x \sim D} \; H(D(y|x), P_\Theta(y|x))$$

$$H(D,P) = \mathrm{E}_{x \sim D} \left[\log_2 \frac{1}{P(x)} \right]$$

$$H(D) = \mathrm{E}_{x \sim D} \left[\log_2 \frac{1}{D(x)} \right]$$

$$H(D,P) \geq H(D)$$

$$KL(D,P) = H(D,P) - H(D) = \mathrm{E}_{x \sim D} \left[\log_2 \frac{D(x)}{P(x)} \right] \geq 0$$

Summary of Distribution Modeling

$\Theta^* = \mathrm{argmin}_\Theta \, H(D, P_\Theta)$

Consistency:

If there exists Θ with $P_\Theta = D$ then $P_{\Theta^*} = D$.

This follows from

$$H(D, D) = H(D) \leq H(D, P)$$

Methods of Modeling Distributions

Structured Prediction.

$$P(y|x) = \operatorname*{softmax}_y \; W_\Theta(x) \cdot \Phi(y)$$

where this is an **exponential softmax**.

Rate-Distortion Autoencoding.

Variational Autoencoding.

Generative Adversarial Networks.

END