TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Information Theory Revisited

Shannon's Source Coding Theorem

Avoiding Differential Entropy

Measuring Cross Entropy of an Exponential Softmax

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

Although we can train using pseduo-likelihood, it remains unclear how to measure the resulting cross-entropy loss.

One approach is to construct a compression algorithm.

Let $|z_\Phi(y)|$ be the bit length of the compression of y.

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

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

Sparse Labeling Compression (TZ)

After training a graphical model Φ on semantic segmentations we can code a segmentation y by a sparse segmentation $z_\Phi(y)$ assigning a label to only a small fraction of the pixels.

We then define the decoding $y_\Phi(z_\Phi(y))$ to be the result of running deterministic local search over the labels of the unspecified pixels to find a locally best-scoring full semantic segmentation.

We can define $z_\Phi(y)$ by some heuristic approximation to

$$z_\Phi(y) = \underset{z:\, y_\Phi(z)=y}{\mathrm{argmin}} \; |z|$$

where $|z|$ is the number of pixels assigned by z.

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 $z(y)$.

Null-terminated byte strings are prefix-free bit strings.

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

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.

Avoiding 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 the numerical value of σ depends on the choice of units.

A distributions on lengths will have a different entropy when measuring in inches than when measuring in feet.

Also, for σ small we get $H(\mathcal{N}(0,\sigma)) < 0$

More Problems with Differential Entropy

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

- Differential entropy violates the source coding theorem — it takes an infinite number of bits to code a real number.
- Differential 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 avoid differential entropy and differential cross entropy.

Differential KL-divergence is Independent of Units

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

If x has units of length then $p(x)$ has units of probability mass per length.

In this case $p(x)/q(x)$ is dimensionless.

Avoiding Differential Entropy

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

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

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

Lossy Compression

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

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

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

END