Focal Loss

Explanation and Thoughts

Andrew Forrester April 26, 2019

Quick Summary & Thoughts

In computer-vision object detection, "focal loss" [1] is meant to focus learning on the worse-performing situations. It can be expressed as

$$\begin{aligned} \mathrm{FL}(p_t) &= -(1-p_t)^\gamma \ln(p_t) && \text{(focal loss, a modulated cross-entropy)} \\ &\text{or} \\ &= -\alpha_t(1-p_t)^\gamma \ln(p_t), && \text{(alpha-balanced or weighted focal loss)} \end{aligned}$$

where $p_t = \hat{y}_c$ is the probability of the ground truth class, or the score of the correct class, as described further below. It is a term of a training loss function and, during training of a detector, it focuses the adaptation of the detector toward the answers it gets very wrong, meaning both confident and wrong (misclassified as object/foreground or non-object/background).

One idea inspired by the focal loss paper is that the tuning focal parameter γ could be set high to start off (maybe 2 is high enough for many cases) and then be reduced as the detector gets better at the worse cases, to shift attention toward the less-worse cases. Other functions besides $f(\hat{y}_c) = (1-\hat{y}_c)^\gamma$ can be used, as mentioned below, to tune the attention in a different way if the need arises.

Context

detector

Suppose we'd like to build a neural network to act as an object detector that can take in an image x (eg, an array of RGB pixel values) and return (1) a bounding box[1] $\hat{b}$ for each detected object in the image and (2) a classification score vector $\hat{y}$ for each bounding box. The vector $\hat{y}$ gives a score for each possible category of object, where the scores in a single vector are normalized to add up to one. Although not technically a probability distribution in a frequentist sense, $\hat{y}$ can be roughly thought of as one in a Bayesian sense.

We'd like to build this detector to model and estimate the behavior of the ideal, best function that could perform this task with the given image data. In practice, an approximation to this best function is found and applied for a restricted set of image data using some kind of display (eg, a computer screen to turn the array x into a light transmission) and one to many humans (preferrably experts at spotting these categories of objects) who create a bounding box b and category label vector[2] y for each object in an image. Each label vector will act as a target value for the model, so that as the model is improved, the estimates $\hat{y}$ must collectively get closer to the targets y. The vectors y can also represent target scores that the best function *would* output but hasn't actually yet generated as labels. The model could potentially outperform expert humans, in which case the model or "estimator" actually becomes the best-available function, but it might take some investigation to determine that is occurring.

Figure 1 shows diagrams of detectors, simplified to illustrate single output score-vectors rather than the multiple output bounding boxes and score-vector for each box.

training the detector

During training, for the detector to learn not just from its positive calls (objects positively detected) but also from its negative calls (no objects detected), the output should additionally include the test bounding boxes with negative calls and their associated score-vectors *with* a probability assigned for "background" (no object). So in this case we should add one more component to $\hat{y}$ to represent the "object category" of *background*. Each score-vector including the background category has to add up to one, so this changes the normalization a bit from that considered above.

[1] (or, rather, an anchor box with shifts toward an expected better-accuracy bounding box)

[2] If the experts are absolutely certain of what they're seeing, this will be a "one-hot" encoding, with a probability of 1 for the correct class and zeros for the other classes, where 1 is a "hot" value and 0 is a "cold" value. However, if even the experts are uncertain about what they're seeing, this could be a non-singular probability/credence distribution.

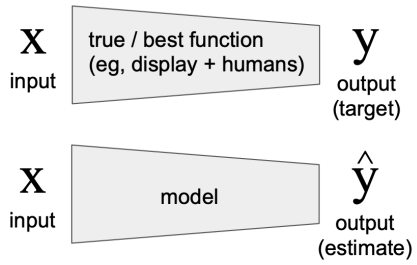

Figure 1: An object-detector model estimates the target scores/labels that are or can be generated by the best detectors (expert humans).

Actually, even for the positive calls and the output of the final model, including the probability of background is probably a good idea; it's a more complete expression of what the detector is determining.

We will consider the case of a binary categorization where all non-background objects are collapsed into one category called "foreground". In this case, we'll have a $\hat{y}$ with two components: one for background and one for foreground. This will be a useful perspective when considering issues of class imbalance between objects and non-objects.

data

We will analyze the situation where we have images with small numbers of objects compared to the number of bounding boxes that the network proposes and tests. That means there will be a class imbalance between objects and non-objects and hence foreground and background.

Cross-Entropy

The usual discrete cross-entropy, which can be interpreted as an asymmetric measure of the deviation of one discrete probability distribution q from another p, is

$$\mathrm{CE} \quad = \quad -\sum_i p_i \ln q_i.$$

In the context of our object-detector problem, where we have a target probability distribution y and a model-assessed probability distribution $\hat{y}$, we want a measure of the deviation of $\hat{y}$ away from the true distribution y:

$$\mathrm{CE} \quad = \quad -\sum_i y_i \ln \hat{y}_i,$$

where the sum is across all classes of objects. In the case of a one-hot encoding ("ohe"), meaning that the best/true classification expresses complete confidence, this formula simplifies:

$$\begin{aligned} \mathrm{CE}_{\mathrm{ohe}} &= -\sum_i y_i \ln \hat{y}_i \\ &= -(0)\ln \hat{y}_1 - (0)\ln \hat{y}_2 - \cdots - (1)\ln \hat{y}_{\mathrm{c}} - \cdots - (0)\ln \hat{y}_{N-1} - (0)\ln \hat{y}_N \\ &= -\ln \hat{y}_c, \end{aligned}$$

where $i = c$ is the correct class index out of N classes. Considered as a loss function (or part of one), this is a simple "log-loss".

Binary Cross-Entropy

If we take the binary "background" (no relevant object) and "foreground" (a relevant object) versions of y and $\hat{y}$, then we have $y = \langle y_{\mathrm{b}},\, y_{\mathrm{f}} \rangle = \langle 1 - y_{\mathrm{f}},\, y_{\mathrm{f}} \rangle$ and $\hat{y} = \langle \hat{y}_{\mathrm{b}},\, \hat{y}_{\mathrm{f}} \rangle = \langle 1 - \hat{y}_{\mathrm{f}},\, \hat{y}_{\mathrm{f}} \rangle$, where we've used the fact that the probabilities

sum to one. Thus the binary cross-entropy is

$$\begin{aligned} \mathrm{CE} &= -\sum_i y_i \ln \hat{y}_i \\ &= -y_\mathrm{b} \ln \hat{y}_\mathrm{b} - y_\mathrm{f} \ln \hat{y}_\mathrm{f} \\ &= -(1-y_\mathrm{f}) \ln(1-\hat{y}_\mathrm{f}) - y_\mathrm{f} \ln \hat{y}_\mathrm{f}. \end{aligned}$$

In the case of a one-hot encoding, we have

$$\mathrm{CE}_\mathrm{ohe} = \begin{cases} -\ln(\hat{y}_\mathrm{f}) & \text{if } y_\mathrm{f} = 1 \\ -\ln(1-\hat{y}_\mathrm{f}) & \text{if } y_\mathrm{f} = 0. \end{cases}$$

We can use an abbreviation as we did above, referring to the correct class c:

$$\hat{y}_c = \begin{cases} \hat{y}_\mathrm{f} & \text{if } y_\mathrm{f} = 1 \quad (y_\mathrm{b} = 0) \\ \hat{y}_\mathrm{b} = 1 - \hat{y}_\mathrm{f} & \text{if } y_\mathrm{f} = 0 \quad (y_\mathrm{b} = 1). \end{cases}$$

Then we can rewrite the one-hot cross-entropy just as we did before, as

$$\mathrm{CE}_\mathrm{ohe} = -\ln(\hat{y}_\mathrm{c}).$$

Modification 1: weighted cross entropy

To deal with foreground-background class imbalance and even out learning across the two classes, one can use "alpha balancing" with a weighting parameter $\alpha \in [0,1]$ for the foreground class and $1-\alpha$ for the background class. We can define α_c analogously to $\hat{y}_c$ as "the weight associated with the correct class".

$$\mathrm{CE}^\mathrm{w}_\mathrm{ohe} = -\alpha_c \ln(\hat{y}_\mathrm{c}).$$

For instance, if not enough learning is taking place for the foreground class, one can use a relatively large value of α, say 0.8, to magnify the *relative* loss compared with the background class.

Modification 2: focal loss

The basic idea of focal loss is to focus learning on the very wrong or worse scores to prioritize fixing them and improving performance that way. The strategy is to reduce the relative amount of loss for scores that are not very wrong, thus effectively raising the importance of the very wrong scores. More to the point, this increases the gradient of the loss for very wrong scores (really, the near-most-wrong-scores) and decreases the gradient for the many not-so-bad and okay cases, sending a bigger signal for adjusting network weight-parameters and learning on those very-wrong (near-very-wrong) scores. See Figure 2 and pay particular attention to the slopes of the curves. Focal loss helps the total loss to not be swamped out by the signals from the many not-so-bad and okay cases and instead focus on the worse cases.

This relative reduction of loss can be achieved with any function $f(\hat{y}_c)$ that is larger when $\hat{y}_c$ is small and smaller when $\hat{y}_c$ is large, preferrably that stays in the range $[0,1]$:

$$\mathrm{CE}'_\mathrm{ohe} = -f(\hat{y}_c)\ \ln(\hat{y}_\mathrm{c}).$$

Some functions that would work are $\cos(\hat{y}_c)$, $\exp(-\hat{y}_c)$, and $\exp(-{\hat{y}_c}^2)$, but we could instead simply use a power of $(1-\hat{y}_c)$:

$$f(\hat{y}_c) = (1-\hat{y}_c)^\gamma,$$

where the exponent is a tunable focusing paramter $\gamma \geq 0$. Tuning γ will select precisely how the focus is altered towards the worse-scores. This is the choice of function made by the team that coined the term "focal loss":

$$\begin{aligned} \mathrm{FL}(\hat{y}_c) &= \mathrm{CE}'_\mathrm{ohe} \\ &= -(1-\hat{y}_c)^\gamma\ \ln(\hat{y}_\mathrm{c}). \end{aligned}$$

Since the factor multiplying the one-hot cross-entropy changes with $\hat{y}_c$, instead of calling this a weighted cross-entropy, we can call it a modulated cross-entropy.

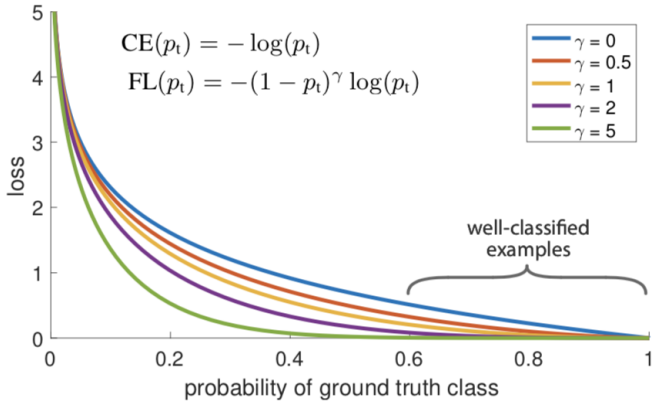

Figure 2: As they say in the focal loss paper [1], setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > 0.5$), putting more focus on hard, misclassified examples.

Modification 3: weighted focal loss

To simultaneously focus learning on very wrong scores *and* deal with imbalance in learning across the foreground-background classes, we can put the two previous modifications together:

$$\begin{aligned} \mathrm{FL}^{\mathrm{w}}(\hat{y}_c) &= \mathrm{CE}^{\mathrm{w}\prime}_{\mathrm{ohe}} \\ &= -\alpha_c(1-\hat{y}_c)^{\gamma}\,\ln(\hat{y}_c). \end{aligned}$$

In the focal loss paper, they confront a class imbalance with many more background instances than foreground instances. However, the effect of this imbalance on learning can actually be counteracted (and even reversed!) by using focal loss. As can be seen in Figure 3, the focal loss with $\gamma = 2$ dramatically concentrates loss and attention on few bad cases within the background class, indicating that there were relatively few very wrong scores in that class compared with the foreground class, which didn't get altered nearly as much. The best performance of the (RetinaNet) detector with $\gamma = 2$ was found with $\alpha = 0.25$, which is shifting attention *away* from the rare foreground class and *towards* the more common background class, whose net effects had apparently been reduced as they were concentrated into fewer contributions from worse-scoring instances.

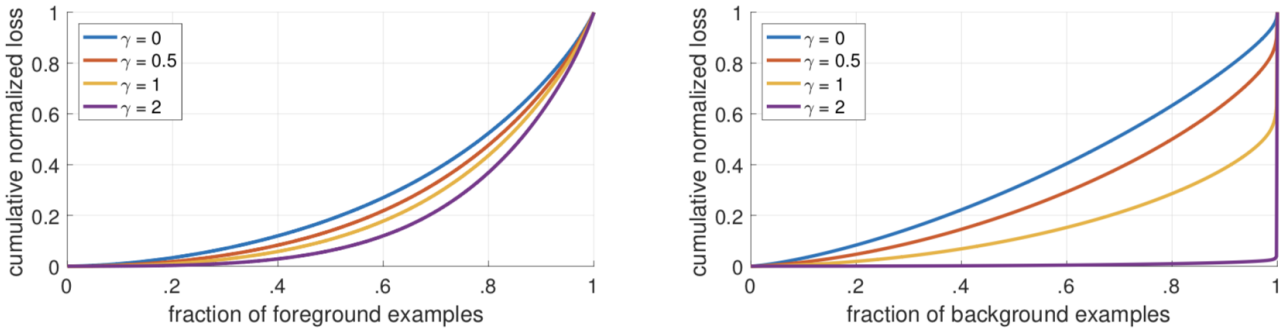

Figure 3: See the focal loss paper for explanation of the cumulative normalized loss. It's organized in such a way as to see how concentrated among the class instances the larger-contributions to the loss are (on the right portion of each graph). In the background instances, the larger-contributions are extremely concentrated when the $\gamma = 2$ focal loss is applied.

References

[1] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. CoRR, abs/1708.02002, 2017.