Training and evaluation

Overview

Types of outputs:

- regression
- classification

Distribution of outputs:

- 1 per token **tagging**
- 1 per document **sentiment analysis, document/author/genre classification**
- n per document **summarization, machine translation**

sentiment regression where you have stuff labeled zero through 100 depending on how happy it is. for the output activation will often want to use linear activation.

Regression

if we use sigmoid function, it will never going to work because those outputs can not take values over one

output activation: for targets in $\mathbb{R}$, use linear activation function $f(x) = x$

loss function: commonly mean squared error (MSE), but there are many options

network example: train a network to do the cumulative something

Classification

binary classes

output activation: output is distribution over classes: use sigmoid/softmax activation function

multiple classes

loss function: commonly cross-entropy

Softmax activation

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

normalize part

[p0,p1,p2,p3], sum of p equals 1

we have x output classes which assign different probabilities
the hope is we have the highest probability assigned the the one associated with the correct outpus

the same form of the [p0,p1,p2,p3]distribution,
just load all probabilities to one bucket

one hot encoding for output label [1,0,0,0]
and the outputs received from the network look like a distribution
[0.7,0.15,0.1,0.05] then we assign it to the first class

Cross-entropy loss

are maximized when the two distributions are very far apart and minimized when they are close together

$$CELoss(y, p) = -\sum_i y_i \log(p_i)$$

y = [1,0,0,0] p = [p0,p1,p2,p3]

if yi= 1 and pi = 1, then it is the smallest(1*log(1)+0*log
if yi = 1 and pi=0, then you get infinitely large loss

RNNs

1-per-token "token classification" (POS tagging/NER)

- *predictors:* 1-hot-encoded words
- *target:* cell output at each input, 1-hot-encoded POS tag

1-per-document "document classification" (sentiment analysis/author identification)

- *predictors:* 1-hot-encoded words or word embeddings
- *target:* cell output at EOS input

end of sequence

n-per-document (machine translation/etc.)

- *predictors:* 1-hot-encoded words or word embeddings
- *target*: decoder outputs until EOS

Teacher forcing

At run time, we feed the previous output to the next decoder cell.

At train time, we can do better. We know the truth, so each cell is given the *correct* previous output.

Transformers, etc.

- No more EOS input tokens!
- Still need EOS on output.

because they are not processed sequentially. we don’t have to know where to stop, we just need to provide all the inputs

Evaluation metrics

when you write up the paper,what metric are you going to say are you going to provide that says this worked well or poorly?
We have an evaluation metric for a particular task and can use that as the loss function.
That's great. It's not always the case.

- *token/document classification:* precision/recall, etc.
- *regression:* (R)MSE, etc.
- *machine translation:* BLEU
- *"understanding":* GLUE

BLEU

Bilingual evaluation understudy

https://cloud.google.com/translate/automl/docs/evaluate#bleu-

GLUE

General Language Understanding Evaluation

https://gluebenchmark.com/

https://arxiv.org/abs/1804.07461