Neural Stack Writeup

October 29, 2017

Contents

```
emacs --version
```

```
GNU Emacs 25.3.1
Copyright (C) 2017 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of GNU Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
```

1 Structure

1.1 Introduction

What did you do? Why?

1.2 Methods

In this section, we describe the dataset used, experimental setup and details of models evaluated.

Unlike previous work, we don't rely on the performance of component models for estimating a set of *weights*. Instead, we use the predictions (probability distributions) from the components as input to the ensemble model and predict directly the output for the wILI prediction problem, skipping any intermediate weight estimation.

1.2.1 Component model inputs / outputs

1.2.2 Ensemble model inputs

The ensemble models *merge* probability distributions from input component models and predict probability distributions, in the same space, as output. During training, the mean categorical crossentropy loss between the output distribution and the actual wILI value, as one-hot distribution, is minimized. This loss minimization is equivalent to maximizing the log score as described by CDC [TODO: Elaborate more].

1.2.3 Models

We evaluate two neural network models for the stacking task. The first model (mixture density network) works by approximating the input probability distributions using a gaussian and the output as a mixture of gaussians. The second model (convolutional neural network) works directly on the probability distributions (as vector of bin values) from components as input and returns a vector of values as probability distribution as output. Both models are described below:

1. Model a: Mixture density network

   A mixture density network (CITE) is a simple feed forward neural network which outputs parameters for a mixture of distributions. The loss function here finds the crossentropy loss between this distribution generated by the mixture and one-hot representation of the truth.

   This model *assumes* a gaussian distribution input from the component models. It takes in the mean and variance (CHECK: std?) of the distribution from each of the component models and returns a mixture of gaussians by outputting a set of means (μ_i), variances (σ_i^2) and weights (w_i) for each distribution in the mixture. The final distribution using n component models is then given by:

$$F(x) = \sum_{i=1}^{n} w_i f(x, \mu_i, \sigma_i^2) \tag{1}$$

Where $f(x, \mu_i, \sigma_i^2)$ represents a gaussian with mean μ_i and variance σ_i^2.

2. Model b: Convolutional neural network

   This model puts less assumptions on the input and output distributions and uses a set of 1-dimensional convolutional layers over the discrete input distributions. As the output, it outputs the complete discrete probability distribution vector.

1.3 Results

What did you find?

1.4 Discussion

What does it all mean?

1.5 Conclusions